# OpenShift Container Platform 4.19

## Installing on AWS

Installing OpenShift Container Platform on Amazon Web Services

Last Updated: 2026-01-15

# OpenShift Container Platform 4.19 Installing on AWS

Installing OpenShift Container Platform on Amazon Web Services

## Abstract

This document describes how to install OpenShift Container Platform on Amazon Web Services.

# Table of Contents

# CHAPTER 1. INSTALLATION METHODS

You can install OpenShift Container Platform on Amazon Web Services using installer-provisioned, user-provisioned infrastructure, or on a single node, depending on the needs of your use case.

The default installation type uses installer-provisioned infrastructure, where the installation program provisions the underlying infrastructure for the cluster.

You can also install OpenShift Container Platform on infrastructure that you provision. If you do not use infrastructure that the installation program provisions, you must manage and maintain the cluster resources yourself.

You can also install OpenShift Container Platform on a single node, which is a specialized installation method that is ideal for edge computing environments.

## 1.1. INSTALLING A CLUSTER ON INSTALLER-PROVISIONED INFRASTRUCTURE

You can install a cluster on AWS infrastructure that is provisioned by the OpenShift Container Platform installation program, by using one of the following methods:

You can install OpenShift Container Platform on AWS infrastructure that is provisioned by the OpenShift Container Platform installation program. You can install a cluster quickly by using the default configuration options.

You can install a customized cluster on AWS infrastructure that the installation program provisions. You can also customize your OpenShift Container Platform network configuration during installation, so that your cluster can coexist with your existing IP address allocations and adhere to your network requirements. The installation program allows for some customization to be applied at the installation stage. Many other customization options are available post-installation.

You can install OpenShift Container Platform on AWS on installer-provisioned infrastructure by using an internal mirror of the installation release content. You can use this method to install a cluster that does not require an active internet connection to obtain the software components.

You can install OpenShift Container Platform on an existing AWS Virtual Private Cloud (VPC). You can use this installation method if you have constraints set by the guidelines of your company, such as limits when creating new accounts or infrastructure.

You can install a private cluster on an existing AWS VPC. You can use this method to deploy OpenShift Container Platform on an internal network that is not visible to the internet.

OpenShift Container Platform can be deployed into AWS regions that are specifically designed for US government agencies at the federal, state, and local level, as well as contractors, educational institutions, and other US customers that must run sensitive workloads in the cloud.

## 1.2. INSTALLING A CLUSTER ON USER-PROVISIONED INFRASTRUCTURE

You can install a cluster on AWS in one of two ways: on infrastructure that you provide or infrastructure that you provide by using an internal mirror of the installation release content.

To install OpenShift Container Platform on AWS infrastructure that you provide, you can use the provided CloudFormation templates to create stacks of AWS resources that represent each of the components required for an OpenShift Container Platform installation.

To install a cluster that does not require an active internet connection to obtain the software components, install OpenShift Container Platform on AWS infrastructure that you provide by using an internal mirror of the installation release content. You can also use this installation method to ensure that your clusters only use container images that satisfy your organizational controls on external content. While you can install OpenShift Container Platform by using the mirrored content, your cluster still requires internet access to use the AWS APIs.

## 1.3. INSTALLING A CLUSTER ON A SINGLE NODE

Installing OpenShift Container Platform on a single node alleviates some of the requirements for high availability and large scale clusters. However, you must address requirements for installing on a single node, and the additional requirements for installing single-node OpenShift on a cloud provider.

After addressing the requirements for single node installation, use the installing a customized cluster on AWS procedure to install the cluster. The installing single-node OpenShift manually section contains an exemplary **install-config.yaml** file when installing an OpenShift Container Platform cluster on a single node.

## 1.4. ADDITIONAL RESOURCES

- Installing a cluster quickly on AWS
- Installing a customized cluster on AWS
- Post-installation
- Installing a cluster on AWS in a restricted network
- Installing a cluster on an existing Virtual Private Cloud
- Installing a private cluster on an existing VPC
- Installing a cluster on AWS infrastructure that you provide
- Installing a cluster on AWS in a restricted network with user-provisioned infrastructure
- Installation process

# CHAPTER 2. CONFIGURING AN AWS ACCOUNT

Before you can install OpenShift Container Platform, you must configure an Amazon Web Services (AWS) account.

## 2.1. CONFIGURING ROUTE 53

To install OpenShift Container Platform, the Amazon Web Services (AWS) account you use must have a dedicated public hosted zone in your Route 53 service. This zone must be authoritative for the domain. The Route 53 service provides cluster DNS resolution and name lookup for external connections to the cluster.

### Procedure

1. Identify your domain, or subdomain, and registrar. You can transfer an existing domain and registrar or obtain a new one through AWS or another source.

   NOTE

   If you purchase a new domain through AWS, it takes time for the relevant DNS changes to propagate. For more information about purchasing domains through AWS, see Registering Domain Names Using Amazon Route 53 in the AWS documentation.

2. If you are using an existing domain and registrar, migrate its DNS to AWS. See Making Amazon Route 53 the DNS Service for an Existing Domain in the AWS documentation.

3. Create a public hosted zone for your domain or subdomain. See Creating a Public Hosted Zone in the AWS documentation.
   Use an appropriate root domain, such as **openshiftcorp.com**, or subdomain, such as **clusters.openshiftcorp.com**.

4. Extract the new authoritative name servers from the hosted zone records. See Getting the Name Servers for a Public Hosted Zone in the AWS documentation.

5. Update the registrar records for the AWS Route 53 name servers that your domain uses. For example, if you registered your domain to a Route 53 service in a different accounts, see the following topic in the AWS documentation: Adding or Changing Name Servers or Glue Records.

6. If you are using a subdomain, add its delegation records to the parent domain. This gives Amazon Route 53 responsibility for the subdomain. Follow the delegation procedure outlined by the DNS provider of the parent domain. See Creating a subdomain that uses Amazon Route 53 as the DNS service without migrating the parent domain in the AWS documentation for an example high level procedure.

### 2.1.1. Ingress Operator endpoint configuration for AWS Route 53

Configure Ingress Operator endpoints for OpenShift Container Platform clusters in Amazon Web Services (AWS) GovCloud (US) regions. Verifying these settings helps to ensure that your cluster connects to the correct API endpoints.

If you install in either AWS GovCloud (US) US-West or US-East region, the Ingress Operator uses **us-gov-west-1** region for Route53 and tagging API clients.

The Ingress Operator uses **https://tagging.us-gov-west-1.amazonaws.com** as the tagging API endpoint if a tagging custom endpoint is configured that includes the string 'us-gov-east-1'.

For more information on AWS GovCloud (US) endpoints, see the Service Endpoints in the AWS documentation about GovCloud (US).

IMPORTANT

Private, disconnected installations are not supported for AWS GovCloud when you install in the **us-gov-east-1** region.

### Example Route 53 configuration

```
platform:
  aws:
    region: us-gov-west-1
    serviceEndpoints:
    - name: ec2
      url: https://ec2.us-gov-west-1.amazonaws.com
    - name: elasticloadbalancing
      url: https://elasticloadbalancing.us-gov-west-1.amazonaws.com
    - name: route53
      url: https://route53.us-gov.amazonaws.com
    - name: tagging
      url: https://tagging.us-gov-west-1.amazonaws.com
```

where:

**https://route53.us-gov.amazonaws.com**

Defaults to **https://route53.us-gov.amazonaws.com** for both AWS GovCloud (US) regions.

**https://tagging.us-gov-west-1.amazonaws.com**

Only the US-West region has endpoints for tagging. Omit this parameter if your cluster is in another region.

## 2.2. AWS ACCOUNT LIMITS

The OpenShift Container Platform cluster uses several Amazon Web Services (AWS) components, and the default Service Limits affect your ability to install OpenShift Container Platform clusters.

If you use certain cluster configurations, deploy your cluster in certain AWS regions, or run multiple clusters from your account, you might need to request additional resources for your AWS account.

The following table summarizes the AWS components whose limits can impact your ability to install and run OpenShift Container Platform clusters.

| Component | Number of clusters available by default | Default AWS limit | Description |
|---|---|---|---|

| Component | Number of clusters available by default | Default AWS limit | Description |
|---|---|---|---|
| Instance Limits | Varies | Varies | By default, each cluster creates the following instances:<br>• One bootstrap machine, which is removed after installation<br>• Three control plane nodes<br>• Three worker nodes<br>These instance type counts are within a new account's default limit. To deploy more worker nodes, enable autoscaling, deploy large workloads, or use a different instance type, review your account limits to ensure that your cluster can deploy the machines that you need.<br>In most regions, the worker machines use an **m6i.large** instance and the bootstrap and control plane machines use **m6i.xlarge** instances. In some regions, including all regions that do not support these instance types, **m5.large** and **m5.xlarge** instances are used instead. |
| Elastic IPs (EIPs) | 0 to 1 | 5 EIPs per account | To provision the cluster in a highly available configuration, the installation program creates a public and private subnet for each availability zone within a region. Each private subnet requires aNAT Gateway, and each NAT gateway requires a separate elastic IP. Review the AWS region map to determine how many availability zones are in each region. To take advantage of the default high availability, install the cluster in a region with at least three availability zones. To install a cluster in a region with more than five availability zones, you must increase the EIP limit.<br>**IMPORTANT**<br>To use the **us-east-1** region, you must increase the EIP limit for your account. |
| Virtual Private Clouds (VPCs) | 5 | 5 VPCs per region | Each cluster creates its own VPC. |

| Component | Number of clusters available by default | Default AWS limit | Description |
|---|---|---|---|
| Elastic Load Balancing (ELB/NLB) | 3 | 20 per region | By default, each cluster creates internal and external network load balancers for the master API server and a single Classic Load Balancer for the router. Deploying more Kubernetes **Service** objects with type **LoadBalancer** will create additional load balancers. |
| NAT Gateways | 5 | 5 per availability zone | The cluster deploys one NAT gateway in each availability zone. |
| Elastic Network Interfaces (ENIs) | At least 12 | 350 per region | The default installation creates 21 ENIs and an ENI for each availability zone in your region. For example, the **us-east-1** region contains six availability zones, so a cluster that is deployed in that zone uses 27 ENIs. Review the AWS region map to determine how many availability zones are in each region.<br><br>Additional ENIs are created for additional machines and ELB load balancers that are created by cluster usage and deployed workloads. |
| VPC Gateway | 20 | 20 per account | Each cluster creates a single VPC Gateway for S3 access. |
| S3 buckets | 99 | 100 buckets per account | Because the installation process creates a temporary bucket and the registry component in each cluster creates a bucket, you can create only 99 OpenShift Container Platform clusters per AWS account. |
| Security Groups | 250 | 2,500 per account | Each cluster creates 10 distinct security groups. |

## 2.3. REQUIRED AWS PERMISSIONS FOR THE IAM USER

To deploy all components of an OpenShift Container Platform cluster, you must grant the all the required permissions to the IAM user that you create in Amazon Web Services (AWS).

**NOTE**

Your IAM user must have the permission **tag:GetResources** in the region **us-east-1** to delete the base cluster resources. As part of the AWS API requirement, the OpenShift Container Platform installation program performs various actions in this region.

When you attach the **AdministratorAccess** policy to the IAM user that you create in Amazon Web Services (AWS), you grant that user all of the required permissions. To deploy all components of an OpenShift Container Platform cluster, the IAM user requires the following permissions:

Example 2.1. Required EC2 permissions for installation

- **ec2:AttachNetworkInterface**
- **ec2:AuthorizeSecurityGroupEgress**
- **ec2:AuthorizeSecurityGroupIngress**
- **ec2:CopyImage**
- **ec2:CreateNetworkInterface**
- **ec2:CreateSecurityGroup**
- **ec2:CreateTags**
- **ec2:CreateVolume**
- **ec2:DeleteSecurityGroup**
- **ec2:DeleteSnapshot**
- **ec2:DeleteTags**
- **ec2:DeregisterImage**
- **ec2:DescribeAccountAttributes**
- **ec2:DescribeAddresses**
- **ec2:DescribeAvailabilityZones**
- **ec2:DescribeDhcpOptions**
- **ec2:DescribeImages**
- **ec2:DescribeInstanceAttribute**
- **ec2:DescribeInstanceCreditSpecifications**
- **ec2:DescribeInstances**
- **ec2:DescribeInstanceTypes**
- **ec2:DescribeInstanceTypeOfferings**
- **ec2:DescribeInternetGateways**
- **ec2:DescribeKeyPairs**
- **ec2:DescribeNatGateways**

- **ec2:DescribeNetworkAcls**
- **ec2:DescribeNetworkInterfaces**
- **ec2:DescribePrefixLists**
- **ec2:DescribePublicIpv4Pools** (only required if **publicIpv4Pool** is specified in **install-config.yaml**)
- **ec2:DescribeRegions**
- **ec2:DescribeRouteTables**
- **ec2:DescribeSecurityGroupRules**
- **ec2:DescribeSecurityGroups**
- **ec2:DescribeSubnets**
- **ec2:DescribeTags**
- **ec2:DescribeVolumes**
- **ec2:DescribeVpcAttribute**
- **ec2:DescribeVpcClassicLink**
- **ec2:DescribeVpcClassicLinkDnsSupport**
- **ec2:DescribeVpcEndpoints**
- **ec2:DescribeVpcs**
- **ec2:DisassociateAddress** (only required if **publicIpv4Pool** is specified in **install-config.yaml**)
- **ec2:GetEbsDefaultKmsKeyId**
- **ec2:ModifyInstanceAttribute**
- **ec2:ModifyNetworkInterfaceAttribute**
- **ec2:RevokeSecurityGroupEgress**
- **ec2:RevokeSecurityGroupIngress**
- **ec2:RunInstances**
- **ec2:TerminateInstances**

Example 2.2. Required permissions for creating network resources during installation

- **ec2:AllocateAddress**
- **ec2:AssociateAddress**

- **ec2:AssociateDhcpOptions**
- **ec2:AssociateRouteTable**
- **ec2:AttachInternetGateway**
- **ec2:CreateDhcpOptions**
- **ec2:CreateInternetGateway**
- **ec2:CreateNatGateway**
- **ec2:CreateRoute**
- **ec2:CreateRouteTable**
- **ec2:CreateSubnet**
- **ec2:CreateVpc**
- **ec2:CreateVpcEndpoint**
- **ec2:ModifySubnetAttribute**
- **ec2:ModifyVpcAttribute**

NOTE

If you use an existing Virtual Private Cloud (VPC), your account does not require these permissions for creating network resources.

Example 2.3. Required Elastic Load Balancing permissions (ELB) for installation

- **elasticloadbalancing:AddTags**
- **elasticloadbalancing:ApplySecurityGroupsToLoadBalancer**
- **elasticloadbalancing:AttachLoadBalancerToSubnets**
- **elasticloadbalancing:ConfigureHealthCheck**
- **elasticloadbalancing:CreateListener**
- **elasticloadbalancing:CreateLoadBalancer**
- **elasticloadbalancing:CreateLoadBalancerListeners**
- **elasticloadbalancing:CreateTargetGroup**
- **elasticloadbalancing:DeleteLoadBalancer**
- **elasticloadbalancing:DeregisterInstancesFromLoadBalancer**
- **elasticloadbalancing:DeregisterTargets**

- **elasticloadbalancing:DescribeInstanceHealth**
- **elasticloadbalancing:DescribeListeners**
- **elasticloadbalancing:DescribeLoadBalancerAttributes**
- **elasticloadbalancing:DescribeLoadBalancers**
- **elasticloadbalancing:DescribeTags**
- **elasticloadbalancing:DescribeTargetGroupAttributes**
- **elasticloadbalancing:DescribeTargetHealth**
- **elasticloadbalancing:ModifyLoadBalancerAttributes**
- **elasticloadbalancing:ModifyTargetGroup**
- **elasticloadbalancing:ModifyTargetGroupAttributes**
- **elasticloadbalancing:RegisterInstancesWithLoadBalancer**
- **elasticloadbalancing:RegisterTargets**
- **elasticloadbalancing:SetLoadBalancerPoliciesOfListener**
- **elasticloadbalancing:SetSecurityGroups**

IMPORTANT

OpenShift Container Platform uses both the ELB and ELBv2 API services to provision load balancers. The permission list shows permissions required by both services. A known issue exists in the AWS web console where both services use the same **elasticloadbalancing** action prefix but do not recognize the same actions. You can ignore the warnings about the service not recognizing certain **elasticloadbalancing** actions.

Example 2.4. Required IAM permissions for installation

- **iam:AddRoleToInstanceProfile**
- **iam:CreateInstanceProfile**
- **iam:CreateRole**
- **iam:DeleteInstanceProfile**
- **iam:DeleteRole**
- **iam:DeleteRolePolicy**
- **iam:GetInstanceProfile**
- **iam:GetRole**

- **iam:GetRolePolicy**
- **iam:GetUser**
- **iam:ListInstanceProfilesForRole**
- **iam:ListRoles**
- **iam:ListUsers**
- **iam:PassRole**
- **iam:PutRolePolicy**
- **iam:RemoveRoleFromInstanceProfile**
- **iam:SimulatePrincipalPolicy**
- **iam:TagInstanceProfile**
- **iam:TagRole**

NOTE

- If you specify an existing IAM role in the **install-config.yaml** file, the following IAM permissions are not required: **iam:CreateRole**,**iam:DeleteRole**, **iam:DeleteRolePolicy**, and **iam:PutRolePolicy**.
- If you have not created a load balancer in your AWS account, the IAM user also requires the **iam:CreateServiceLinkedRole** permission.

Example 2.5. Required Route 53 permissions for installation

- **route53:ChangeResourceRecordSets**
- **route53:ChangeTagsForResource**
- **route53:CreateHostedZone**
- **route53:DeleteHostedZone**
- **route53:GetChange**
- **route53:GetHostedZone**
- **route53:ListHostedZones**
- **route53:ListHostedZonesByName**
- **route53:ListResourceRecordSets**
- **route53:ListTagsForResource**
- **route53:UpdateHostedZoneComment**

Example 2.6. Required Amazon Simple Storage Service (S3) permissions for installation

- **s3:CreateBucket**
- **s3:DeleteBucket**
- **s3:GetAccelerateConfiguration**
- **s3:GetBucketAcl**
- **s3:GetBucketCors**
- **s3:GetBucketLocation**
- **s3:GetBucketLogging**
- **s3:GetBucketObjectLockConfiguration**
- **s3:GetBucketPolicy**
- **s3:GetBucketRequestPayment**
- **s3:GetBucketTagging**
- **s3:GetBucketVersioning**
- **s3:GetBucketWebsite**
- **s3:GetEncryptionConfiguration**
- **s3:GetLifecycleConfiguration**
- **s3:GetReplicationConfiguration**
- **s3:ListBucket**
- **s3:PutBucketAcl**
- **s3:PutBucketPolicy**
- **s3:PutBucketTagging**
- **s3:PutEncryptionConfiguration**

Example 2.7. S3 permissions that cluster Operators require

- **s3:DeleteObject**
- **s3:GetObject**
- **s3:GetObjectAcl**

- **s3:GetObjectTagging**
- **s3:GetObjectVersion**
- **s3:PutObject**
- **s3:PutObjectAcl**
- **s3:PutObjectTagging**

Example 2.8. Required permissions to delete base cluster resources

- **autoscaling:DescribeAutoScalingGroups**
- **ec2:DeleteNetworkInterface**
- **ec2:DeletePlacementGroup**
- **ec2:DeleteVolume**
- **elasticloadbalancing:DeleteTargetGroup**
- **elasticloadbalancing:DescribeTargetGroups**
- **iam:DeleteAccessKey**
- **iam:DeleteUser**
- **iam:DeleteUserPolicy**
- **iam:ListAttachedRolePolicies**
- **iam:ListInstanceProfiles**
- **iam:ListRolePolicies**
- **iam:ListUserPolicies**
- **s3:DeleteObject**
- **s3:ListBucketVersions**
- **tag:GetResources**

Example 2.9. Required permissions to delete network resources

- **ec2:DeleteDhcpOptions**
- **ec2:DeleteInternetGateway**
- **ec2:DeleteNatGateway**
- **ec2:DeleteRoute**

- **ec2:DeleteRouteTable**
- **ec2:DeleteSubnet**
- **ec2:DeleteVpc**
- **ec2:DeleteVpcEndpoints**
- **ec2:DetachInternetGateway**
- **ec2:DisassociateRouteTable**
- **ec2:ReleaseAddress**
- **ec2:ReplaceRouteTableAssociation**

NOTE

If you use an existing VPC, your account does not require these permissions to delete network resources. Instead, your account only requires the **tag:UntagResources** permission to delete network resources.

Example 2.10. Optional permissions for installing a cluster with a custom Key Management Service (KMS) key

- **kms:CreateGrant**
- **kms:Decrypt**
- **kms:DescribeKey**
- **kms:Encrypt**
- **kms:GenerateDataKey**
- **kms:GenerateDataKeyWithoutPlainText**
- **kms:ListGrants**
- **kms:RevokeGrant**

Example 2.11. Required permissions to delete a cluster with shared instance roles

- **iam:UntagRole**

Example 2.12. Required permissions to delete a cluster with shared instance profiles

- **tag:UntagResources**

Example 2.13. Additional IAM and S3 permissions that are required to create manifests

- **iam:GetUserPolicy**
- **iam:ListAccessKeys**
- **iam:PutUserPolicy**
- **iam:TagUser**
- **s3:AbortMultipartUpload**
- **s3:GetBucketPublicAccessBlock**
- **s3:ListBucket**
- **s3:ListBucketMultipartUploads**
- **s3:PutBucketPublicAccessBlock**
- **s3:PutLifecycleConfiguration**

NOTE

If you are managing your cloud provider credentials with mint mode, the IAM user also requires the **iam:CreateAccessKey** and **iam:CreateUser** permissions.

Example 2.14. Optional permissions for instance and quota checks for installation

- **servicequotas:ListAWSDefaultServiceQuotas**

Example 2.15. Optional permissions for the cluster owner account when installing a cluster on a shared VPC

- **sts:AssumeRole**

Example 2.16. Required permissions for enabling Bring your own public IPv4 addresses (BYOIP) feature for installation

- **ec2:DescribePublicIpv4Pools**
- **ec2:DisassociateAddress**

## 2.4. CREATING AN IAM USER

Before you install OpenShift Container Platform, you must create a secondary IAM administrative user and assign permissions to create the cluster.

Each Amazon Web Services (AWS) account contains a root user account that is based on the email address you used to create the account.

IMPORTANT

This is a highly-privileged account, and it is recommended to use it for only initial account and billing configuration, creating an initial set of users, and securing the account.

As you complete the Creating an IAM User in Your AWS Account procedure in the Amazon Web Services (AWS) documentation, set the following options:

**Procedure**

1. Specify the IAM user name and select **Programmatic access**.
2. Attach the **AdministratorAccess** policy to ensure that the account has sufficient permission to create the cluster. This policy provides the cluster with the ability to grant credentials to each OpenShift Container Platform component. The cluster grants the components only the credentials that they require.

   NOTE

   While it is possible to create a policy that grants the all of the required AWS permissions and attach it to the user, this is not the preferred option. The cluster will not have the ability to grant additional credentials to individual components, so the same credentials are used by all components.

3. Optional: Add metadata to the user by attaching tags.
4. Confirm that the user name that you specified is granted the **AdministratorAccess** policy.
5. Record the access key ID and secret access key values. You must use these values when you configure your local machine to run the installation program.

   IMPORTANT

   You cannot use a temporary session token that you generated while using a multi-factor authentication device to authenticate to AWS when you deploy a cluster. The cluster continues to use your current AWS credentials to create AWS resources for the entire life of the cluster, so you must use key-based, long-term credentials.

## 2.5. IAM POLICIES AND AWS AUTHENTICATION

You can specify your own IAM roles if required. By default, the installation program creates instance profiles for the bootstrap, control plane, and compute instances with the necessary permissions for the cluster to operate.

NOTE

To enable pulling images from the Amazon Elastic Container Registry (ECR) as a postinstallation task in a single-node OpenShift cluster, you must add the **AmazonEC2ContainerRegistryReadOnly** policy to the IAM role associated with the cluster's control plane role.

However, you can create your own IAM roles and specify them as part of the installation process. You might need to specify your own roles to deploy the cluster or to manage the cluster after installation. For example:

- Your organization's security policies require that you use a more restrictive set of permissions to install the cluster.
- After the installation, the cluster is configured with an Operator that requires access to additional services.

If you choose to specify your own IAM roles, you can take the following steps:

- Begin with the default policies and adapt as required. For more information, see "Default permissions for IAM instance profiles".
- To create a policy template that is based on the cluster's activity, see "Using AWS IAM Analyzer to create policy templates".

### 2.5.1. Default permissions for IAM instance profiles

To ensure your cluster operates with the correct security permissions in OpenShift Container Platform, review the default IAM instance profiles created by the installation program.

By default, the installation program creates IAM instance profiles for the bootstrap, control plane, and compute instances with the necessary permissions for the cluster to operate.

The following lists specify the default permissions for control plane and compute machines:

**Default IAM role permissions for control plane instance profiles**

- **ec2:AttachVolume**
- **ec2:AuthorizeSecurityGroupIngress**
- **ec2:CreateSecurityGroup**
- **ec2:CreateTags**
- **ec2:CreateVolume**
- **ec2:DeleteSecurityGroup**
- **ec2:DeleteVolume**
- **ec2:Describe***
- **ec2:DetachVolume**
- **ec2:ModifyInstanceAttribute**

- **ec2:ModifyVolume**
- **ec2:RevokeSecurityGroupIngress**
- **elasticloadbalancing:AddTags**
- **elasticloadbalancing:AttachLoadBalancerToSubnets**
- **elasticloadbalancing:ApplySecurityGroupsToLoadBalancer**
- **elasticloadbalancing:CreateListener**
- **elasticloadbalancing:CreateLoadBalancer**
- **elasticloadbalancing:CreateLoadBalancerPolicy**
- **elasticloadbalancing:CreateLoadBalancerListeners**
- **elasticloadbalancing:CreateTargetGroup**
- **elasticloadbalancing:ConfigureHealthCheck**
- **elasticloadbalancing:DeleteListener**
- **elasticloadbalancing:DeleteLoadBalancer**
- **elasticloadbalancing:DeleteLoadBalancerListeners**
- **elasticloadbalancing:DeleteTargetGroup**
- **elasticloadbalancing:DeregisterInstancesFromLoadBalancer**
- **elasticloadbalancing:DeregisterTargets**
- **elasticloadbalancing:Describe***
- **elasticloadbalancing:DetachLoadBalancerFromSubnets**
- **elasticloadbalancing:ModifyListener**
- **elasticloadbalancing:ModifyLoadBalancerAttributes**
- **elasticloadbalancing:ModifyTargetGroup**
- **elasticloadbalancing:ModifyTargetGroupAttributes**
- **elasticloadbalancing:RegisterInstancesWithLoadBalancer**
- **elasticloadbalancing:RegisterTargets**
- **elasticloadbalancing:SetLoadBalancerPoliciesForBackendServer**
- **elasticloadbalancing:SetLoadBalancerPoliciesOfListener**
- **kms:DescribeKey**

**Default IAM role permissions for compute instance profiles**

- **ec2:DescribeInstances**
- **ec2:DescribeRegions**

### 2.5.2. Specifying an existing IAM role

Instead of allowing the installation program to create IAM instance profiles with the default permissions, you can use the **install-config.yaml** file to specify an existing IAM role for control plane and compute instances.

**Prerequisites**

- You have an existing **install-config.yaml** file.

**Procedure**

1. Update **compute.platform.aws.iamRole** with an existing role for the compute machines.

   **Sample install-config.yaml file with an IAM role for compute instances**

   ```
   compute:
   - hyperthreading: Enabled
     name: worker
     platform:
       aws:
         iamRole: ExampleRole
   ```

2. Update **controlPlane.platform.aws.iamRole** with an existing role for the control plane machines.

   **Sample install-config.yaml file with an IAM role for control plane instances**

   ```
   controlPlane:
     hyperthreading: Enabled
     name: master
     platform:
       aws:
         iamRole: ExampleRole
   ```

3. Save the file and reference it when installing the OpenShift Container Platform cluster.

**NOTE**

To change or update an IAM account after the cluster has been installed, see RHOCP 4 AWS cloud-credentials access key is expired (Red Hat Knowledgebase).

**Additional resources**

- Quickly install a cluster

- Install a cluster with cloud customizations on installer-provisioned infrastructure
- Installing a cluster on user-provisioned infrastructure in AWS by using CloudFormation templates

# CHAPTER 3. INSTALLER-PROVISIONED INFRASTRUCTURE

## 3.1. PREPARING TO INSTALL A CLUSTER ON AWS

To install an OpenShift Container Platform cluster on Amazon Web Services (AWS), you must verify your internet connectivity, download the installation program, install the OpenShift CLI (**oc**), and generate an SSH key pair.

If required, you also need to manually create long-term credentials for AWS or configure an AWS cluster to use short-term credentials with Amazon Web Services Security Token Service (AWS STS).

The following list outlines in detail the steps to prepare to install an OpenShift Container Platform cluster on AWS:

- Verifying internet connectivity for your cluster.
- Configuring an AWS account.
- Downloading the installation program.

  **NOTE**

  If you are installing in a disconnected environment, you extract the installation program from the mirrored content. For more information, see Mirroring images for a disconnected installation.

- Installing the OpenShift CLI (**oc**).

  **NOTE**

  If you are installing in a disconnected environment, install **oc** to the mirror host.

- Generating an SSH key pair. You can use this key pair to authenticate into the OpenShift Container Platform cluster's nodes after it is deployed.
- If the cloud identity and access management (IAM) APIs are not accessible in your environment, or if you do not want to store an administrator-level credential secret in the **kube-system** namespace, manually creating long-term credentials for AWS or configuring an AWS cluster to use short-term credentials with (AWS STS).

### 3.1.1. Internet access for OpenShift Container Platform

In OpenShift Container Platform 4.19, you require access to the internet to install your cluster.

You must have internet access to perform the following actions:

- Access OpenShift Cluster Manager to download the installation program and perform subscription management. If the cluster has internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access Quay.io to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.

**IMPORTANT**

If your cluster cannot have direct internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the required content and use it to populate a mirror registry with the installation packages. With some installation types, the environment that you install your cluster in will not require internet access. Before you update the cluster, you update the content of the mirror registry.

### 3.1.2. Obtaining the installation program

Before you install OpenShift Container Platform, download the installation file on the host you are using for installation.

#### Prerequisites

- You have a computer that runs Linux or macOS, with 500 MB of local disk space.

#### Procedure

1. Go to the Cluster Type page on the Red Hat Hybrid Cloud Console. If you have a Red Hat account, log in with your credentials. If you do not, create an account.

   **TIP**

   You can also download the binaries for a specific OpenShift Container Platform release .

2. Select your infrastructure provider from the **Run it yourself** section of the page.

3. Select your host operating system and architecture from the dropdown menus under **OpenShift Installer** and click **Download Installer**.

4. Place the downloaded file in the directory where you want to store the installation configuration files.

   **IMPORTANT**

   - The installation program creates several files on the computer that you use to install your cluster. You must keep the installation program and the files that the installation program creates after you finish installing the cluster. Both of the files are required to delete the cluster.
   - Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. To remove your cluster, complete the OpenShift Container Platform uninstallation procedures for your specific cloud provider.

5. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

   ```
   $ tar -xvf openshift-install-linux.tar.gz
   ```

6. Download your installation pull secret from Red Hat OpenShift Cluster Manager . This pull secret allows you to authenticate with the services that are provided by the included authorities,

including Quay.io, which serves the container images for OpenShift Container Platform components.

TIP

Alternatively, you can retrieve the installation program from the Red Hat Customer Portal, where you can specify a version of the installation program to download. However, you must have an active subscription to access this page.

### 3.1.3. Installing the OpenShift CLI on Linux

To manage your cluster and deploy applications from the command line, install the OpenShift CLI (**oc**) binary on Linux.

IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform.

Download and install the new version of **oc**.

#### Procedure

1. Navigate to the Download OpenShift Container Platform page on the Red Hat Customer Portal.
2. Select the architecture from the **Product Variant** list.
3. Select the appropriate version from the **Version** list.
4. Click **Download Now** next to the **OpenShift v4.19 Linux Clients** entry and save the file.
5. Unpack the archive:

```
$ tar xvf <file>
```

6. Place the **oc** binary in a directory that is on your **PATH**.
   To check your **PATH**, execute the following command:

```
$ echo $PATH
```

#### Verification

- After you install the OpenShift CLI, it is available using the **oc** command:

```
$ oc <command>
```

### 3.1.4. Installing the OpenShift CLI on Windows

To manage your cluster and deploy applications from the command line, install OpenShift CLI (**oc**) binary on Windows.

**IMPORTANT**

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform.

Download and install the new version of **oc**.

**Procedure**

1. Navigate to the Download OpenShift Container Platform page on the Red Hat Customer Portal.
2. Select the appropriate version from the **Version** list.
3. Click **Download Now** next to the **OpenShift v4.19 Windows Client** entry and save the file.
4. Extract the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH** variable.
   To check your **PATH** variable, open the command prompt and execute the following command:

```
C:\> path
```

**Verification**

- After you install the OpenShift CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

### 3.1.5. Installing the OpenShift CLI on macOS

To manage your cluster and deploy applications from the command line, install the OpenShift CLI (**oc**) binary on macOS.

**IMPORTANT**

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform.

Download and install the new version of **oc**.

**Procedure**

1. Navigate to the Download OpenShift Container Platform page on the Red Hat Customer Portal.
2. Select the architecture from the **Product Variant** list.
3. Select the appropriate version from the **Version** list.
4. Click **Download Now** next to the **OpenShift v4.19 macOS Clients** entry and save the file.

   **NOTE**

   For macOS arm64, choose the **OpenShift v4.19 macOS arm64 Client** entry.

5. Unpack and unzip the archive.

6. Move the **oc** binary to a directory on your **PATH** variable.
   To check your **PATH** variable, open a terminal and execute the following command:

```
$ echo $PATH
```

**Verification**

- Verify your installation by using an **oc** command:

```
$ oc <command>
```

## 3.1.6. Generating a key pair for cluster node SSH access

To enable secure, passwordless SSH access to your cluster nodes, provide an SSH public key during the OpenShift Container Platform installation. This ensures that the installation program automatically configures the Red Hat Enterprise Linux CoreOS (RHCOS) nodes for remote authentication through the **core** user.

The SSH public key gets added to the **~/.ssh/authorized_keys** list for the **core** user on each node. After the key is passed to the Red Hat Enterprise Linux CoreOS (RHCOS) nodes through their Ignition config files, you can use the key pair to SSH in to the RHCOS nodes as the user **core**. To access the nodes through SSH, the private key identity must be managed by SSH for your local user.

If you want to SSH in to your cluster nodes to perform installation debugging or disaster recovery, you must provide the SSH public key during the installation process. The **./openshift-install gather** command also requires the SSH public key to be in place on the cluster nodes.

**IMPORTANT**

Do not skip this procedure in production environments, where disaster recovery and debugging is required.

**NOTE**

You must use a local key, not one that you configured with platform-specific approaches.

**Procedure**

1. If you do not have an existing SSH key pair on your local machine to use for authentication onto your cluster nodes, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t ed25519 -N '' -f <path>/<file_name>
```

   Specifies the path and file name, such as **~/.ssh/id_ed25519**, of the new SSH key. If you have an existing key pair, ensure your public key is in the your **~/.ssh** directory.

NOTE

If you plan to install an OpenShift Container Platform cluster that uses the RHEL cryptographic libraries that have been submitted to NIST for FIPS 140-2/140-3 Validation on only the **x86_64**, **ppc64le**, and **s390x** architectures, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. View the public SSH key:

```
$ cat <path>/<file_name>.pub
```

For example, run the following to view the **~/.ssh/id_ed25519.pub** public key:

```
$ cat ~/.ssh/id_ed25519.pub
```

3. Add the SSH private key identity to the SSH agent for your local user, if it has not already been added. SSH agent management of the key is required for password-less SSH authentication onto your cluster nodes, or if you want to use the **./openshift-install gather** command.

NOTE

On some distributions, default SSH private key identities such as **~/.ssh/id_rsa** and **~/.ssh/id_dsa** are managed automatically.

a. If the **ssh-agent** process is not already running for your local user, start it as a background task:

```
$ eval "$(ssh-agent -s)"
```

**Example output**

```
Agent pid 31874
```

NOTE

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

4. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name>
```

Specifies the path and file name for your SSH private key, such as **~/.ssh/id_ed25519**

**Example output**

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

**Next steps**

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

### 3.1.7. Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.19, the Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, requires internet access. If your cluster is connected to the internet, Telemetry runs automatically, and your cluster is registered to OpenShift Cluster Manager.

After you confirm that your OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually by using OpenShift Cluster Manager, use subscription watch to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

**Additional resources**

- See About remote health monitoring for more information about the Telemetry service

## 3.2. INSTALLING A CLUSTER ON AWS

In OpenShift Container Platform version 4.19, you can install a cluster on Amazon Web Services (AWS) that uses the default configuration options.

### 3.2.1. Prerequisites

- You reviewed details about the OpenShift Container Platform installation and update processes.
- You read the documentation on selecting a cluster installation method and preparing it for users.
- You configured an AWS account to host the cluster.

  **IMPORTANT**

  If you have an AWS profile stored on your computer, it must not use a temporary session token that you generated while using a multi-factor authentication device. The cluster continues to use your current AWS credentials to create AWS resources for the entire life of the cluster, so you must use key-based, long-term credentials. To generate appropriate keys, see Managing Access Keys for IAM Users in the AWS documentation. You can supply the keys when you run the installation program.

- If you use a firewall, you configured it to allow the sites that your cluster requires access to.

### 3.2.2. Deploying the cluster

You can install OpenShift Container Platform on a compatible cloud platform.

**IMPORTANT**

You can run the **create cluster** command of the installation program only once, during initial installation.

### Prerequisites

- You have configured an account with the cloud platform that hosts your cluster.
- You have the OpenShift Container Platform installation program and the pull secret for your cluster.
- You have verified that the cloud provider account on your host has the correct permissions to deploy the cluster. An account with incorrect permissions causes the installation process to fail with an error message that displays the missing permissions.

### Procedure

1. In the directory that contains the installation program, initialize the cluster deployment by running the following command:

   ```
   $ ./openshift-install create cluster --dir <installation_directory> \ 1
       --log-level=info 2
   ```

   1 For **<installation_directory>**, specify the directory name to store the files that the installation program creates.

   2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

   When specifying the directory:

   - Verify that the directory has the **execute** permission. This permission is required to run Terraform binaries under the installation directory.
   - Use an empty directory. Some installation assets, such as bootstrap X.509 certificates, have short expiration intervals, therefore you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

2. Provide values at the prompts:

   a. Optional: Select an SSH key to use to access your cluster machines.

      **NOTE**

      For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

   b. Select **aws** as the platform to target.

   c. If you do not have an Amazon Web Services (AWS) profile stored on your computer, enter the AWS access key ID and secret access key for the user that you configured to run the installation program.

NOTE

The AWS access key ID and secret access key are stored in **~/.aws/credentials** in the home directory of the current user on the installation host. You are prompted for the credentials by the installation program if the credentials for the exported profile are not present in the file. Any credentials that you provide to the installation program are stored in the file.

d. Select the AWS region to deploy the cluster to.

e. Select the base domain for the Route 53 service that you configured for your cluster.

f. Enter a descriptive name for your cluster.

g. Paste the pull secret from Red Hat OpenShift Cluster Manager .

3. Optional: Remove or disable the **AdministratorAccess** policy from the IAM account that you used to install the cluster.

NOTE

The elevated permissions provided by the **AdministratorAccess** policy are required only during installation.

## Verification

When the cluster deployment completes successfully:

- The terminal displays directions for accessing your cluster, including a link to the web console and credentials for the **kubeadmin** user.
- Credential information also outputs to **<installation_directory>/.openshift_install.log**.

IMPORTANT

Do not delete the installation program or the files that the installation program creates. Both are required to delete the cluster.

## Example output

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```

IMPORTANT

- The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.
- It is recommended that you use Ignition config files within 12 hours after they are generated because the 24-hour certificate rotates from 16 to 22 hours after the cluster is installed. By using the Ignition config files within 12 hours, you can avoid installation failure if the certificate update runs during installation.

### Additional resources

- Configuration and credential file settings (AWS documentation)

### 3.2.3. Logging in to the cluster by using the CLI

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

#### Prerequisites

- You deployed an OpenShift Container Platform cluster.
- You installed the OpenShift CLI (**oc**).

#### Procedure

1. Export the **kubeadmin** credentials by running the following command:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration by running the following command:

```
$ oc whoami
```

**Example output**

```
system:admin
```

### 3.2.4. Logging in to the cluster by using the web console

The **kubeadmin** user exists by default after an OpenShift Container Platform installation. You can log in to your cluster as the **kubeadmin** user by using the OpenShift Container Platform web console.

**Prerequisites**

- You have access to the installation host.
- You completed a cluster installation and all cluster Operators are available.

**Procedure**

1. Obtain the password for the **kubeadmin** user from the **kubeadmin-password** file on the installation host:

   ```
   $ cat <installation_directory>/auth/kubeadmin-password
   ```

   **NOTE**

   Alternatively, you can obtain the **kubeadmin** password from the **<installation_directory>/.openshift_install.log** log file on the installation host.

2. List the OpenShift Container Platform web console route:

   ```
   $ oc get routes -n openshift-console | grep 'console-openshift'
   ```

   **NOTE**

   Alternatively, you can obtain the OpenShift Container Platform route from the **<installation_directory>/.openshift_install.log** log file on the installation host.

   **Example output**

   ```
   console     console-openshift-console.apps.<cluster_name>.<base_domain>            console
   https   reencrypt/Redirect   None
   ```

3. Navigate to the route detailed in the output of the preceding command in a web browser and log in as the **kubeadmin** user.

**Additional resources**

- Accessing the web console

## 3.2.5. Next steps

- Validating an installation.
- Customize your cluster.
- If necessary, you can Remote health reporting.
- If necessary, you can remove cloud provider credentials.

## 3.3. INSTALLING A CLUSTER ON AWS WITH CUSTOMIZATIONS

In OpenShift Container Platform version 4.19, you can install a customized cluster on infrastructure that the installation program provisions on Amazon Web Services (AWS). To customize the installation, you modify parameters in the **install-config.yaml** file before you install the cluster.

NOTE

The scope of the OpenShift Container Platform installation configurations is intentionally narrow. It is designed for simplicity and ensured success. You can complete many more OpenShift Container Platform configuration tasks after an installation completes.

### 3.3.1. Prerequisites

- You reviewed details about the OpenShift Container Platform installation and update processes.
- You read the documentation on selecting a cluster installation method and preparing it for users.
- You configured an AWS account to host the cluster.

  IMPORTANT

  If you have an AWS profile stored on your computer, it must not use a temporary session token that you generated while using a multi-factor authentication device. The cluster continues to use your current AWS credentials to create AWS resources for the entire life of the cluster, so you must use long-term credentials. To generate appropriate keys, see Managing Access Keys for IAM Users in the AWS documentation. You can supply the keys when you run the installation program.

- If you use a firewall, you configured it to allow the sites that your cluster requires access to.

### 3.3.2. Obtaining an AWS Marketplace image

If you are deploying an OpenShift Container Platform cluster using an AWS Marketplace image, you must first subscribe through AWS. Subscribing to the offer provides you with the AMI ID that the installation program uses to deploy compute nodes.

NOTE

You should only modify the RHCOS image for compute machines to use an AWS Marketplace image. Control plane machines and infrastructure nodes do not require an OpenShift Container Platform subscription and use the public RHCOS default image by default, which does not incur subscription costs on your AWS bill. Therefore, you should not modify the cluster default boot image or the control plane boot images. Applying the AWS Marketplace image to them will incur additional licensing costs that cannot be recovered.

#### Prerequisites

- You have an AWS account to purchase the offer. This account does not have to be the same account that is used to install the cluster.

**Procedure**

1. Complete the OpenShift Container Platform subscription from the AWS Marketplace.

2. Record the AMI ID for your specific AWS Region. As part of the installation process, you must update the **install-config.yaml** file with this value before deploying the cluster.

**Sample install-config.yaml file with AWS Marketplace compute nodes**

```
apiVersion: v1
baseDomain: example.com
compute:
- hyperthreading: Enabled
  name: worker
  platform:
    aws:
      amiID: ami-06c4d345f7c207239 (1)
      type: m5.4xlarge
  replicas: 3
metadata:
  name: test-cluster
platform:
  aws:
    region: us-east-2 (2)
sshKey: ssh-ed25519 AAAA...
pullSecret: '{"auths": ...}'
```

(1) The AMI ID from your AWS Marketplace subscription.

(2) Your AMI ID is associated with a specific AWS Region. When creating the installation configuration file, ensure that you select the same AWS Region that you specified when configuring your subscription.

### 3.3.3. Creating the installation configuration file

You can customize the OpenShift Container Platform cluster you install on Amazon Web Services (AWS).

**Prerequisites**

- You have the OpenShift Container Platform installation program and the pull secret for your cluster.

**Procedure**

1. Create the **install-config.yaml** file.

   a. Change to the directory that contains the installation program and run the following command:

   ```
   $ ./openshift-install create install-config --dir <installation_directory>
   ```

- **<installation_directory>**: For **<installation_directory>**, specify the directory name to store the files that the installation program creates.
  When specifying the directory:

- Verify that the directory has the **execute** permission. This permission is required to run Terraform binaries under the installation directory.

- Use an empty directory. Some installation assets, such as bootstrap X.509 certificates, have short expiration intervals, therefore you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

b. At the prompts, provide the configuration details for your cloud:

   i. Optional: Select an SSH key to use to access your cluster machines.

   NOTE

   For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

   ii. Select **AWS** as the platform to target.

   iii. If you do not have an Amazon Web Services (AWS) profile stored on your computer, enter the AWS access key ID and secret access key for the user that you configured to run the installation program.

   iv. Select the AWS region to deploy the cluster to.

   v. Select the base domain for the Route 53 service that you configured for your cluster.

   vi. Enter a descriptive name for your cluster.

2. Modify the **install-config.yaml** file. You can find more information about the available parameters in the "Installation configuration parameters" section.

   NOTE

   If you are installing a three-node cluster, be sure to set the **compute.replicas** parameter to **0**. This ensures that the cluster's control planes are schedulable. For more information, see "Installing a three-node cluster on AWS".

3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.

   IMPORTANT

   The **install-config.yaml** file is consumed during the installation process. If you want to reuse the file, you must back it up now.

**Additional resources**

- Installation configuration parameters for AWS

#### 3.3.3.1. Minimum resource requirements for cluster installation

Each created cluster must meet minimum requirements so that the cluster runs as expected.

**Table 3.1. Minimum resource requirements**

| Machine | Operating System | vCPU [1] | Virtual RAM | Storage | Input/Output Per Second (IOPS)[2] |
|---|---|---|---|---|---|
| Bootstrap | RHCOS | 4 | 16 GB | 100 GB | 300 |
| Control plane | RHCOS | 4 | 16 GB | 100 GB | 300 |
| Compute | RHCOS, RHEL 8.6 and later [3] | 2 | 8 GB | 100 GB | 300 |

1. One vCPU is equivalent to one physical core when simultaneous multithreading (SMT), or Hyper-Threading, is not enabled. When enabled, use the following formula to calculate the corresponding ratio: (threads per core × cores) × sockets = vCPUs.

2. OpenShift Container Platform and Kubernetes are sensitive to disk performance, and faster storage is recommended, particularly for etcd on the control plane nodes which require a 10 ms p99 fsync duration. Note that on many cloud platforms, storage size and IOPS scale together, so you might need to over-allocate storage volume to obtain sufficient performance.

3. As with all user-provisioned installations, if you choose to use RHEL compute machines in your cluster, you take responsibility for all operating system life cycle management and maintenance, including performing system updates, applying patches, and completing all other required tasks. Use of RHEL 7 compute machines is deprecated and has been removed in OpenShift Container Platform 4.10 and later.

**NOTE**

For OpenShift Container Platform version 4.19, RHCOS is based on RHEL version 9.6, which updates the micro-architecture requirements. The following list contains the minimum instruction set architectures (ISA) that each architecture requires:

- x86-64 architecture requires x86-64-v2 ISA
- ARM64 architecture requires ARMv8.0-A ISA
- IBM Power architecture requires Power 9 ISA
- s390x architecture requires z14 ISA

For more information, see Architectures (RHEL documentation).

If an instance type for your platform meets the minimum requirements for cluster machines, it is supported to use in OpenShift Container Platform.

**Additional resources**

- Optimizing storage

### 3.3.3.2. Tested instance types for AWS

The following Amazon Web Services (AWS) instance types have been tested with OpenShift Container Platform.

**NOTE**

Use the machine types included in the following charts for your AWS instances. If you use an instance type that is not listed in the chart, ensure that the instance size you use matches the minimum resource requirements that are listed in the section named "Minimum resource requirements for cluster installation".

**Example 3.1. Machine types based on 64-bit x86 architecture**

- **c4.***
- **c5.***
- **c5a.***
- **i3.***
- **m4.***
- **m5.***
- **m5a.***
- **m6a.***
- **m6i.***
- **r4.***
- **r5.***
- **r5a.***
- **r6i.***
- **t3.***
- **t3a.***

### 3.3.3.3. Tested instance types for AWS on 64-bit ARM infrastructures

The following Amazon Web Services (AWS) 64-bit ARM instance types have been tested with OpenShift Container Platform.

NOTE

Use the machine types included in the following charts for your AWS ARM instances. If you use an instance type that is not listed in the chart, ensure that the instance size you use matches the minimum resource requirements that are listed in "Minimum resource requirements for cluster installation".

**Example 3.2. Machine types based on 64-bit ARM architecture**

- **c6g.***
- **c7g.***
- **m6g.***
- **m7g.***
- **r8g.***

### 3.3.3.4. Sample customized install-config.yaml file for AWS

You can customize the installation configuration file (**install-config.yaml**) to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.

IMPORTANT

This sample YAML file is provided for reference only. You must obtain your **install-config.yaml** file by using the installation program and modify it. For a full list and description of all installation configuration parameters, see *Installation configuration parameters for AWS*.

#### Sample **install-config.yaml** file for AWS

```
apiVersion: v1 1
baseDomain: example.com
sshKey: ssh-ed25519 AAAA...
pullSecret: '{"auths": ...}'
metadata:
  name: example-cluster
controlPlane: 2
  name: master
  platform:
    aws:
      type: m6i.xlarge
  replicas: 3
compute: 3
-  name: worker
  platform:
    aws:
      type: c5.4xlarge
  replicas: 3
networking: 4
```

```
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
platform: 5
  aws:
    region: us-west-2
```

1. Parameters at the first level of indentation apply to the cluster globally.

2. The **controlPlane** stanza applies to control plane machines.

3. The **compute** stanza applies to compute machines.

4. The **networking** stanza applies to the cluster networking configuration. If you do not provide networking values, the installation program provides default values.

5. The **platform** stanza applies to the infrastructure platform that hosts the cluster.

### Additional resources

- Installation configuration parameters for AWS

#### 3.3.3.5. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

### Prerequisites

- You have an existing **install-config.yaml** file.

- You reviewed the sites that your cluster requires access to and determined whether any of them need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. You added sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.

NOTE

The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.

For installations on Amazon Web Services (AWS), Google Cloud, Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

### Procedure

1. Edit your **install-config.yaml** file and add the proxy settings. For example:

```
apiVersion: v1
```

```
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: ec2.<aws_region>.amazonaws.com,elasticloadbalancing.
<aws_region>.amazonaws.com,s3.<aws_region>.amazonaws.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1 A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**.

2 A proxy URL to use for creating HTTPS connections outside the cluster.

3 A comma-separated list of destination domain names, IP addresses, or other network CIDRs to exclude from proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use * to bypass the proxy for all destinations. If you have added the Amazon **EC2**,**Elastic Load Balancing**, and **S3** VPC endpoints to your VPC, you must add these endpoints to the **noProxy** field.

4 If provided, the installation program generates a config map that is named **user-ca-bundle** in the **openshift-config** namespace that contains one or more additional CA certificates that are required for proxying HTTPS connections. The Cluster Network Operator then creates a **trusted-ca-bundle** config map that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this config map is referenced in the **trustedCA** field of the **Proxy** object. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle.

5 Optional: The policy to determine the configuration of the **Proxy** object to reference the **user-ca-bundle** config map in the **trustedCA** field. The allowed values are **Proxyonly** and **Always**. Use **Proxyonly** to reference the **user-ca-bundle** config map only when **http/https** proxy is configured. Use **Always** to always reference the **user-ca-bundle** config map. The default value is **Proxyonly**.

NOTE

The installation program does not support the proxy **readinessEndpoints** field.

NOTE

If the installer times out, restart and then complete the deployment by using the **wait-for** command of the installer. For example:

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. Save the file and reference it when installing OpenShift Container Platform.

The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster Proxy** object is still created, but it will have a nil **spec**.

NOTE

Only the **Proxy** object named **cluster** is supported, and no additional proxies can be created.

### 3.3.4. Alternatives to storing administrator-level secrets in the kube-system project

By default, administrator secrets are stored in the **kube-system** project. If you configured the **credentialsMode** parameter in the **install-config.yaml** file to **Manual**, you must use one of the following alternatives:

- To manage long-term cloud credentials manually, follow the procedure in Manually creating long-term credentials.
- To implement short-term credentials that are managed outside the cluster for individual components, follow the procedures in Configuring an AWS cluster to use short-term credentials.

#### 3.3.4.1. Manually creating long-term credentials

The Cloud Credential Operator (CCO) can be put into manual mode prior to installation in environments where the cloud identity and access management (IAM) APIs are not reachable, or the administrator prefers not to store an administrator-level credential secret in the cluster **kube-system** namespace.

**Procedure**

1. If you did not set the **credentialsMode** parameter in the **install-config.yaml** configuration file to **Manual**, modify the value as shown:

   **Sample configuration file snippet**

   ```
   apiVersion: v1
   baseDomain: example.com
   credentialsMode: Manual
   # ...
   ```

2. If you have not previously created installation manifest files, do so by running the following command:

   ```
   $ openshift-install create manifests --dir <installation_directory>
   ```

   where **<installation_directory>** is the directory in which the installation program creates files.

3. Set a **$RELEASE_IMAGE** variable with the release image from your installation file by running the following command:

   ```
   $ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
   ```

4. Extract the list of **CredentialsRequest** custom resources (CRs) from the OpenShift Container Platform release image by running the following command:

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

1 The **--included** parameter includes only the manifests that your specific cluster configuration requires.

2 Specify the location of the **install-config.yaml** file.

3 Specify the path to the directory where you want to store the **CredentialsRequest** objects. If the specified directory does not exist, this command creates it.

This command creates a YAML file for each **CredentialsRequest** object.

**Sample CredentialsRequest object**

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
    - effect: Allow
      action:
      - iam:GetUser
      - iam:GetUserPolicy
      - iam:ListAccessKeys
      resource: "*"
  ...
```

5. Create YAML files for secrets in the **openshift-install** manifests directory that you generated previously. The secrets must be stored using the namespace and secret name defined in the **spec.secretRef** for each **CredentialsRequest** object.

**Sample CredentialsRequest object with secrets**

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
```

```
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
    - effect: Allow
      action:
      - s3:CreateBucket
      - s3:DeleteBucket
      resource: "*"
      ...
  secretRef:
    name: <component_secret>
    namespace: <component_namespace>
  ...
```

**Sample Secret object**

```
apiVersion: v1
kind: Secret
metadata:
  name: <component_secret>
  namespace: <component_namespace>
data:
  aws_access_key_id: <base64_encoded_aws_access_key_id>
  aws_secret_access_key: <base64_encoded_aws_secret_access_key>
```

IMPORTANT

Before upgrading a cluster that uses manually maintained credentials, you must ensure that the CCO is in an upgradeable state.

#### 3.3.4.2. Configuring an AWS cluster to use short-term credentials

To install a cluster that is configured to use the AWS Security Token Service (STS), you must configure the CCO utility and create the required AWS resources for your cluster.

##### 3.3.4.2.1. Configuring the Cloud Credential Operator utility

To create and manage cloud credentials from outside of the cluster when the Cloud Credential Operator (CCO) is operating in manual mode, extract and prepare the CCO utility (**ccoctl**) binary.

NOTE

The **ccoctl** utility is a Linux binary that must run in a Linux environment.

**Prerequisites**

- You have access to an OpenShift Container Platform account with cluster administrator access.
- You have installed the OpenShift CLI (**oc**).
- You have created an AWS account for the **ccoctl** utility to use with the following permissions:

Required **iam** permissions

- **iam:CreateOpenIDConnectProvider**
- **iam:CreateRole**
- **iam:DeleteOpenIDConnectProvider**
- **iam:DeleteRole**
- **iam:DeleteRolePolicy**
- **iam:GetOpenIDConnectProvider**
- **iam:GetRole**
- **iam:GetUser**
- **iam:ListOpenIDConnectProviders**
- **iam:ListRolePolicies**
- **iam:ListRoles**
- **iam:PutRolePolicy**
- **iam:TagOpenIDConnectProvider**
- **iam:TagRole**

Required **s3** permissions

- **s3:CreateBucket**
- **s3:DeleteBucket**
- **s3:DeleteObject**
- **s3:GetBucketAcl**
- **s3:GetBucketTagging**
- **s3:GetObject**
- **s3:GetObjectAcl**
- **s3:GetObjectTagging**
- **s3:ListBucket**
- **s3:PutBucketAcl**
- **s3:PutBucketPolicy**
- **s3:PutBucketPublicAccessBlock**

- **s3:PutBucketTagging**
- **s3:PutObject**
- **s3:PutObjectAcl**
- **s3:PutObjectTagging**

Required **cloudfront** permissions

- **cloudfront:ListCloudFrontOriginAccessIdentities**
- **cloudfront:ListDistributions**
- **cloudfront:ListTagsForResource**

- If you plan to store the OIDC configuration in a private S3 bucket that is accessed by the IAM identity provider through a public CloudFront distribution URL, the AWS account that runs the **ccoctl** utility requires the following additional permissions:
  - **cloudfront:CreateCloudFrontOriginAccessIdentity**
  - **cloudfront:CreateDistribution**
  - **cloudfront:DeleteCloudFrontOriginAccessIdentity**
  - **cloudfront:DeleteDistribution**
  - **cloudfront:GetCloudFrontOriginAccessIdentity**
  - **cloudfront:GetCloudFrontOriginAccessIdentityConfig**
  - **cloudfront:GetDistribution**
  - **cloudfront:TagResource**
  - **cloudfront:UpdateDistribution**

NOTE

These additional permissions support the use of the **--create-private-s3-bucket** option when processing credentials requests with the **ccoctl aws create-all** command.

Procedure

1. Set a variable for the OpenShift Container Platform release image by running the following command:

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. Obtain the CCO container image from the OpenShift Container Platform release image by running the following command:

```
$ CCO_IMAGE=$(oc adm release info --image-for='cloud-credential-operator'
$RELEASE_IMAGE -a ~/.pull-secret)
```

NOTE

Ensure that the architecture of the **$RELEASE_IMAGE** matches the architecture of the environment in which you will use the **ccoctl** tool.

3. Extract the **ccoctl** binary from the CCO container image within the OpenShift Container Platform release image by running the following command:

```
$ oc image extract $CCO_IMAGE \
  --file="/usr/bin/ccoctl.<rhel_version>" \ 1
  -a ~/.pull-secret
```

1 For **<rhel_version>**, specify the value that corresponds to the version of Red Hat Enterprise Linux (RHEL) that the host uses. If no value is specified, **ccoctl.rhel8** is used by default. The following values are valid:

- **rhel8**: Specify this value for hosts that use RHEL 8.
- **rhel9**: Specify this value for hosts that use RHEL 9.

NOTE

The **ccoctl** binary is created in the directory from where you executed the command and not in **/usr/bin/**. You must rename the directory or move the **ccoctl.<rhel_version>** binary to **ccoctl**.

4. Change the permissions to make **ccoctl** executable by running the following command:

```
$ chmod 775 ccoctl
```

### Verification

- To verify that **ccoctl** is ready to use, display the help file. Use a relative file name when you run the command, for example:

```
$ ./ccoctl
```

#### Example output

```
OpenShift credentials provisioning tool

Usage:
  ccoctl [command]

Available Commands:
  aws         Manage credentials objects for AWS cloud
  azure       Manage credentials objects for Azure
  gcp         Manage credentials objects for Google cloud
  help        Help about any command
  ibmcloud     Manage credentials objects for {ibm-cloud-title}
  nutanix     Manage credentials objects for Nutanix

Flags:
  -h, --help   help for ccoctl

Use "ccoctl [command] --help" for more information about a command.
```

#### 3.3.4.2.2. Creating AWS resources with the Cloud Credential Operator utility

You have the following options when creating AWS resources:

- You can use the **ccoctl aws create-all** command to create the AWS resources automatically. This is the quickest way to create the resources. See Creating AWS resources with a single command.
- If you need to review the JSON files that the **ccoctl** tool creates before modifying AWS resources, or if the process the **ccoctl** tool uses to create AWS resources automatically does not meet the requirements of your organization, you can create the AWS resources individually. See Creating AWS resources individually .

##### 3.3.4.2.2.1. Creating AWS resources with a single command

If the process the **ccoctl** tool uses to create AWS resources automatically meets the requirements of your organization, you can use the **ccoctl aws create-all** command to automate the creation of AWS resources.

Otherwise, you can create the AWS resources individually. For more information, see "Creating AWS resources individually".

> **NOTE**
>
> By default, **ccoctl** creates objects in the directory in which the commands are run. To create the objects in a different directory, use the **--output-dir** flag. This procedure uses **<path_to_ccoctl_output_dir>** to refer to this directory.

### Prerequisites

You must have:

- Extracted and prepared the **ccoctl** binary.

### Procedure

1. Set a **$RELEASE_IMAGE** variable with the release image from your installation file by running the following command:

   ```
   $ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
   ```

2. Extract the list of **CredentialsRequest** objects from the OpenShift Container Platform release image by running the following command:

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

1. The **--included** parameter includes only the manifests that your specific cluster configuration requires.
2. Specify the location of the **install-config.yaml** file.
3. Specify the path to the directory where you want to store the **CredentialsRequest** objects. If the specified directory does not exist, this command creates it.

NOTE

This command might take a few moments to run.

3. Use the **ccoctl** tool to process all **CredentialsRequest** objects by running the following command:

```
$ ccoctl aws create-all \
  --name=<name> \ 1
  --region=<aws_region> \ 2
  --credentials-requests-dir=<path_to_credentials_requests_directory> \ 3
  --output-dir=<path_to_ccoctl_output_dir> \ 4
  --create-private-s3-bucket 5
```

1. Specify the name used to tag any cloud resources that are created for tracking.
2. Specify the AWS region in which cloud resources will be created.
3. Specify the directory containing the files for the component **CredentialsRequest** objects.
4. Optional: Specify the directory in which you want the **ccoctl** utility to create objects. By default, the utility creates objects in the directory in which the commands are run.
5. Optional: By default, the **ccoctl** utility stores the OpenID Connect (OIDC) configuration files in a public S3 bucket and uses the S3 URL as the public OIDC endpoint. To store the OIDC configuration in a private S3 bucket that is accessed by the IAM identity provider through a public CloudFront distribution URL instead, use the **--create-private-s3-bucket** parameter.

NOTE

If your cluster uses Technology Preview features that are enabled by the **TechPreviewNoUpgrade** feature set, you must include the **--enable-tech-preview** parameter.

**Verification**

- To verify that the OpenShift Container Platform secrets are created, list the files in the **<path_to_ccoctl_output_dir>/manifests** directory:

  ```
  $ ls <path_to_ccoctl_output_dir>/manifests
  ```

  **Example output**

  ```
  cluster-authentication-02-config.yaml
  openshift-cloud-credential-operator-cloud-credential-operator-iam-ro-creds-credentials.yaml
  openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
  openshift-cluster-api-capa-manager-bootstrap-credentials-credentials.yaml
  openshift-cluster-csi-drivers-ebs-cloud-credentials-credentials.yaml
  openshift-image-registry-installer-cloud-credentials-credentials.yaml
  openshift-ingress-operator-cloud-credentials-credentials.yaml
  openshift-machine-api-aws-cloud-credentials-credentials.yaml
  ```

  You can verify that the IAM roles are created by querying AWS. For more information, refer to AWS documentation on listing IAM roles.

##### 3.3.4.2.2.2. Creating AWS resources individually

You can use the **ccoctl** tool to create AWS resources individually. This option might be useful for an organization that shares the responsibility for creating these resources among different users or departments.

Otherwise, you can use the **ccoctl aws create-all** command to create the AWS resources automatically. For more information, see "Creating AWS resources with a single command".

> **NOTE**
>
> By default, **ccoctl** creates objects in the directory in which the commands are run. To create the objects in a different directory, use the **--output-dir** flag. This procedure uses **<path_to_ccoctl_output_dir>** to refer to this directory.
>
> Some **ccoctl** commands make AWS API calls to create or modify AWS resources. You can use the **--dry-run** flag to avoid making API calls. Using this flag creates JSON files on the local file system instead. You can review and modify the JSON files and then apply them with the AWS CLI tool using the **--cli-input-json** parameters.

**Prerequisites**

- Extract and prepare the **ccoctl** binary.

**Procedure**

1. Generate the public and private RSA key files that are used to set up the OpenID Connect provider for the cluster by running the following command:

   ```
   $ ccoctl aws create-key-pair
   ```

   **Example output**

   ```
   2021/04/13 11:01:02 Generating RSA keypair
   ```

```
2021/04/13 11:01:03 Writing private key to /<path_to_ccoctl_output_dir>/serviceaccount-signer.private
2021/04/13 11:01:03 Writing public key to /<path_to_ccoctl_output_dir>/serviceaccount-signer.public
2021/04/13 11:01:03 Copying signing key for use by installer
```

where **serviceaccount-signer.private** and **serviceaccount-signer.public** are the generated key files.

This command also creates a private key that the cluster requires during installation in **/<path_to_ccoctl_output_dir>/tls/bound-service-account-signing-key.key**.

2. Create an OpenID Connect identity provider and S3 bucket on AWS by running the following command:

```
$ ccoctl aws create-identity-provider \
  --name=<name> \ 1
  --region=<aws_region> \ 2
  --public-key-file=<path_to_ccoctl_output_dir>/serviceaccount-signer.public 3
```

1 **<name>** is the name used to tag any cloud resources that are created for tracking.

2 **<aws-region>** is the AWS region in which cloud resources will be created.

3 **<path_to_ccoctl_output_dir>** is the path to the public key file that the **ccoctl aws create-key-pair** command generated.

### Example output

```
2021/04/13 11:16:09 Bucket <name>-oidc created
2021/04/13 11:16:10 OpenID Connect discovery document in the S3 bucket <name>-oidc at .well-known/openid-configuration updated
2021/04/13 11:16:10 Reading public key
2021/04/13 11:16:10 JSON web key set (JWKS) in the S3 bucket <name>-oidc at keys.json updated
2021/04/13 11:16:18 Identity Provider created with ARN: arn:aws:iam::<aws_account_id>:oidc-provider/<name>-oidc.s3.<aws_region>.amazonaws.com
```

where **openid-configuration** is a discovery document and **keys.json** is a JSON web key set file.

This command also creates a YAML configuration file in **/<path_to_ccoctl_output_dir>/manifests/cluster-authentication-02-config.yaml**. This file sets the issuer URL field for the service account tokens that the cluster generates, so that the AWS IAM identity provider trusts the tokens.

3. Create IAM roles for each component in the cluster:

   a. Set a **$RELEASE_IMAGE** variable with the release image from your installation file by running the following command:

   ```
   $ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
   ```

b. Extract the list of **CredentialsRequest** objects from the OpenShift Container Platform release image:

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \
2
  --to=<path_to_directory_for_credentials_requests> 3
```

1 The **--included** parameter includes only the manifests that your specific cluster configuration requires.

2 Specify the location of the **install-config.yaml** file.

3 Specify the path to the directory where you want to store the **CredentialsRequest** objects. If the specified directory does not exist, this command creates it.

c. Use the **ccoctl** tool to process all **CredentialsRequest** objects by running the following command:

```
$ ccoctl aws create-iam-roles \
  --name=<name> \
  --region=<aws_region> \
  --credentials-requests-dir=<path_to_credentials_requests_directory> \
  --identity-provider-arn=arn:aws:iam::<aws_account_id>:oidc-provider/<name>-oidc.s3.
<aws_region>.amazonaws.com
```

NOTE

For AWS environments that use alternative IAM API endpoints, such as GovCloud, you must also specify your region with the **--region** parameter.

If your cluster uses Technology Preview features that are enabled by the **TechPreviewNoUpgrade** feature set, you must include the **--enable-tech-preview** parameter.

For each **CredentialsRequest** object, **ccoctl** creates an IAM role with a trust policy that is tied to the specified OIDC identity provider, and a permissions policy as defined in each **CredentialsRequest** object from the OpenShift Container Platform release image.

## Verification

- To verify that the OpenShift Container Platform secrets are created, list the files in the **<path_to_ccoctl_output_dir>/manifests** directory:

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

### Example output

```
cluster-authentication-02-config.yaml
```

```
openshift-cloud-credential-operator-cloud-credential-operator-iam-ro-creds-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capa-manager-bootstrap-credentials-credentials.yaml
openshift-cluster-csi-drivers-ebs-cloud-credentials-credentials.yaml
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-aws-cloud-credentials-credentials.yaml
```

You can verify that the IAM roles are created by querying AWS. For more information, refer to AWS documentation on listing IAM roles.

#### 3.3.4.2.3. Incorporating the Cloud Credential Operator utility manifests

To implement short-term security credentials managed outside the cluster for individual components, you must move the manifest files that the Cloud Credential Operator utility (**ccoctl**) created to the correct directories for the installation program.

##### Prerequisites

- You have configured an account with the cloud platform that hosts your cluster.
- You have configured the Cloud Credential Operator utility (**ccoctl**).
- You have created the cloud provider resources that are required for your cluster with the **ccoctl** utility.

##### Procedure

1. If you did not set the **credentialsMode** parameter in the **install-config.yaml** configuration file to **Manual**, modify the value as shown:

   **Sample configuration file snippet**

   ```
   apiVersion: v1
   baseDomain: example.com
   credentialsMode: Manual
   # ...
   ```

2. If you have not previously created installation manifest files, do so by running the following command:

   ```
   $ openshift-install create manifests --dir <installation_directory>
   ```

   where **<installation_directory>** is the directory in which the installation program creates files.

3. Copy the manifests that the **ccoctl** utility generated to the **manifests** directory that the installation program created by running the following command:

   ```
   $ cp /<path_to_ccoctl_output_dir>/manifests/* ./manifests/
   ```

4. Copy the **tls** directory that contains the private key to the installation directory:

   ```
   $ cp -a /<path_to_ccoctl_output_dir>/tls .
   ```

### 3.3.5. Deploying the cluster

You can install OpenShift Container Platform on a compatible cloud platform.

IMPORTANT

You can run the **create cluster** command of the installation program only once, during initial installation.

#### Prerequisites

- You have configured an account with the cloud platform that hosts your cluster.
- You have the OpenShift Container Platform installation program and the pull secret for your cluster.
- You have verified that the cloud provider account on your host has the correct permissions to deploy the cluster. An account with incorrect permissions causes the installation process to fail with an error message that displays the missing permissions.

#### Procedure

1. In the directory that contains the installation program, initialize the cluster deployment by running the following command:

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 For **<installation_directory>**, specify the location of your customized **./install-config.yaml** file.

2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

2. Optional: Remove or disable the **AdministratorAccess** policy from the IAM account that you used to install the cluster.

NOTE

The elevated permissions provided by the **AdministratorAccess** policy are required only during installation.

#### Verification

When the cluster deployment completes successfully:

- The terminal displays directions for accessing your cluster, including a link to the web console and credentials for the **kubeadmin** user.
- Credential information also outputs to **<installation_directory>/.openshift_install.log**.

IMPORTANT

Do not delete the installation program or the files that the installation program creates. Both are required to delete the cluster.

### Example output

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```

IMPORTANT

- The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

- It is recommended that you use Ignition config files within 12 hours after they are generated because the 24-hour certificate rotates from 16 to 22 hours after the cluster is installed. By using the Ignition config files within 12 hours, you can avoid installation failure if the certificate update runs during installation.

## 3.3.6. Logging in to the cluster by using the CLI

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

### Prerequisites

- You deployed an OpenShift Container Platform cluster.

- You installed the OpenShift CLI (**oc**).

### Procedure

1. Export the **kubeadmin** credentials by running the following command:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration by running the following command:

```
$ oc whoami
```

**Example output**

```
system:admin
```

### 3.3.7. Logging in to the cluster by using the web console

The **kubeadmin** user exists by default after an OpenShift Container Platform installation. You can log in to your cluster as the **kubeadmin** user by using the OpenShift Container Platform web console.

**Prerequisites**

- You have access to the installation host.
- You completed a cluster installation and all cluster Operators are available.

**Procedure**

1. Obtain the password for the **kubeadmin** user from the **kubeadmin-password** file on the installation host:

```
$ cat <installation_directory>/auth/kubeadmin-password
```

NOTE

Alternatively, you can obtain the **kubeadmin** password from the **<installation_directory>/.openshift_install.log** log file on the installation host.

2. List the OpenShift Container Platform web console route:

```
$ oc get routes -n openshift-console | grep 'console-openshift'
```

NOTE

Alternatively, you can obtain the OpenShift Container Platform route from the **<installation_directory>/.openshift_install.log** log file on the installation host.

**Example output**

```
console     console-openshift-console.apps.<cluster_name>.<base_domain>            console
https   reencrypt/Redirect   None
```

3. Navigate to the route detailed in the output of the preceding command in a web browser and log in as the **kubeadmin** user.

**Additional resources**

- Accessing the web console

### 3.3.8. Next steps

- Validating an installation.
- Customize your cluster.
- If necessary, you can Remote health reporting.
- If necessary, you can remove cloud provider credentials.

## 3.4. INSTALLING A CLUSTER ON AWS WITH NETWORK CUSTOMIZATIONS

In OpenShift Container Platform version 4.19, you can install a cluster on Amazon Web Services (AWS) with customized network configuration options. By customizing your network configuration, your cluster can coexist with existing IP address allocations in your environment and integrate with existing MTU and VXLAN configurations.

You must set most of the network configuration parameters during installation, and you can modify only **kubeProxy** configuration parameters in a running cluster.

### 3.4.1. Prerequisites

- You reviewed details about the OpenShift Container Platform installation and update processes.
- You read the documentation on selecting a cluster installation method and preparing it for users.
- You configured an AWS account to host the cluster.

  **IMPORTANT**

  If you have an AWS profile stored on your computer, it must not use a temporary session token that you generated while using a multi-factor authentication device. The cluster continues to use your current AWS credentials to create AWS resources for the entire life of the cluster, so you must use key-based, long-term credentials. To generate appropriate keys, see Managing Access Keys for IAM Users in the AWS documentation. You can supply the keys when you run the installation program.

- If you use a firewall, you configured it to allow the sites that your cluster requires access to.

### 3.4.2. Network configuration phases

There are two phases prior to OpenShift Container Platform installation where you can customize the network configuration.

**Phase 1**

You can customize the following network-related fields in the **install-config.yaml** file before you create the manifest files:

- **networking.networkType**
- **networking.clusterNetwork**
- **networking.serviceNetwork**
- **networking.machineNetwork**
- **nodeNetworking**
  For more information, see "Installation configuration parameters".

NOTE

Set the **networking.machineNetwork** to match the Classless Inter-Domain Routing (CIDR) where the preferred subnet is located.

IMPORTANT

The CIDR range **172.17.0.0/16** is reserved by **libVirt**. You cannot use any other CIDR range that overlaps with the **172.17.0.0/16** CIDR range for networks in your cluster.

Phase 2

After creating the manifest files by running **openshift-install create manifests**, you can define a customized Cluster Network Operator manifest with only the fields you want to modify. You can use the manifest to specify an advanced network configuration.

During phase 2, you cannot override the values that you specified in phase 1 in the **install-config.yaml** file. However, you can customize the network plugin during phase 2.

## 3.4.3. Creating the installation configuration file

You can customize the OpenShift Container Platform cluster you install on Amazon Web Services (AWS).

### Prerequisites

- You have the OpenShift Container Platform installation program and the pull secret for your cluster.

### Procedure

1. Create the **install-config.yaml** file.

   a. Change to the directory that contains the installation program and run the following command:

   ```
   $ ./openshift-install create install-config --dir <installation_directory>
   ```

   - **<installation_directory>**: For **<installation_directory>**, specify the directory name to store the files that the installation program creates.
     When specifying the directory:

- Verify that the directory has the **execute** permission. This permission is required to run Terraform binaries under the installation directory.
- Use an empty directory. Some installation assets, such as bootstrap X.509 certificates, have short expiration intervals, therefore you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

b. At the prompts, provide the configuration details for your cloud:

i. Optional: Select an SSH key to use to access your cluster machines.

NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

ii. Select **AWS** as the platform to target.

iii. If you do not have an Amazon Web Services (AWS) profile stored on your computer, enter the AWS access key ID and secret access key for the user that you configured to run the installation program.

iv. Select the AWS region to deploy the cluster to.

v. Select the base domain for the Route 53 service that you configured for your cluster.

vi. Enter a descriptive name for your cluster.

2. Modify the **install-config.yaml** file. You can find more information about the available parameters in the "Installation configuration parameters" section.

3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.

IMPORTANT

The **install-config.yaml** file is consumed during the installation process. If you want to reuse the file, you must back it up now.

**Additional resources**

- Installation configuration parameters for AWS

### 3.4.3.1. Minimum resource requirements for cluster installation

Each created cluster must meet minimum requirements so that the cluster runs as expected.

**Table 3.2. Minimum resource requirements**

| Machine | Operating System | vCPU [1] | Virtual RAM | Storage | Input/Output Per Second (IOPS)[2] |
|---|---|---|---|---|---|
| Bootstrap | RHCOS | 4 | 16 GB | 100 GB | 300 |
| Control plane | RHCOS | 4 | 16 GB | 100 GB | 300 |
| Compute | RHCOS, RHEL 8.6 and later [3] | 2 | 8 GB | 100 GB | 300 |

1. One vCPU is equivalent to one physical core when simultaneous multithreading (SMT), or Hyper-Threading, is not enabled. When enabled, use the following formula to calculate the corresponding ratio: (threads per core × cores) × sockets = vCPUs.

2. OpenShift Container Platform and Kubernetes are sensitive to disk performance, and faster storage is recommended, particularly for etcd on the control plane nodes which require a 10 ms p99 fsync duration. Note that on many cloud platforms, storage size and IOPS scale together, so you might need to over-allocate storage volume to obtain sufficient performance.

3. As with all user-provisioned installations, if you choose to use RHEL compute machines in your cluster, you take responsibility for all operating system life cycle management and maintenance, including performing system updates, applying patches, and completing all other required tasks. Use of RHEL 7 compute machines is deprecated and has been removed in OpenShift Container Platform 4.10 and later.

**NOTE**

For OpenShift Container Platform version 4.19, RHCOS is based on RHEL version 9.6, which updates the micro-architecture requirements. The following list contains the minimum instruction set architectures (ISA) that each architecture requires:

- x86-64 architecture requires x86-64-v2 ISA
- ARM64 architecture requires ARMv8.0-A ISA
- IBM Power architecture requires Power 9 ISA
- s390x architecture requires z14 ISA

For more information, see Architectures (RHEL documentation).

If an instance type for your platform meets the minimum requirements for cluster machines, it is supported to use in OpenShift Container Platform.

**Additional resources**

- Optimizing storage

#### 3.4.3.2. Tested instance types for AWS

The following Amazon Web Services (AWS) instance types have been tested with OpenShift Container Platform.

NOTE

Use the machine types included in the following charts for your AWS instances. If you use an instance type that is not listed in the chart, ensure that the instance size you use matches the minimum resource requirements that are listed in the section named "Minimum resource requirements for cluster installation".

**Example 3.3. Machine types based on 64-bit x86 architecture**

- **c4.***
- **c5.***
- **c5a.***
- **i3.***
- **m4.***
- **m5.***
- **m5a.***
- **m6a.***
- **m6i.***
- **r4.***
- **r5.***
- **r5a.***
- **r6i.***
- **t3.***
- **t3a.***

### 3.4.3.3. Tested instance types for AWS on 64-bit ARM infrastructures

The following Amazon Web Services (AWS) 64-bit ARM instance types have been tested with OpenShift Container Platform.

NOTE

Use the machine types included in the following charts for your AWS ARM instances. If you use an instance type that is not listed in the chart, ensure that the instance size you use matches the minimum resource requirements that are listed in "Minimum resource requirements for cluster installation".

**Example 3.4. Machine types based on 64-bit ARM architecture**

- **c6g.***
- **c7g.***
- **m6g.***
- **m7g.***
- **r8g.***

#### 3.4.3.4. Sample customized install-config.yaml file for AWS

You can customize the installation configuration file (**install-config.yaml**) to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.

IMPORTANT

This sample YAML file is provided for reference only. You must obtain your **install-config.yaml** file by using the installation program and modify it. For a full list and description of all installation configuration parameters, see *Installation configuration parameters for AWS*.

**Sample install-config.yaml file for AWS**

```
apiVersion: v1 1
baseDomain: example.com
sshKey: ssh-ed25519 AAAA...
pullSecret: '{"auths": ...}'
metadata:
  name: example-cluster
controlPlane: 2
  name: master
  platform:
    aws:
      type: m6i.xlarge
  replicas: 3
compute: 3
-  name: worker
  platform:
    aws:
      type: c5.4xlarge
  replicas: 3
networking: 4
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
platform: 5
  aws:
    region: us-west-2
```

1 Parameters at the first level of indentation apply to the cluster globally.

2 The **controlPlane** stanza applies to control plane machines.

3 The **compute** stanza applies to compute machines.

4 The **networking** stanza applies to the cluster networking configuration. If you do not provide networking values, the installation program provides default values.

5 The **platform** stanza applies to the infrastructure platform that hosts the cluster.

#### Additional resources

- Installation configuration parameters for AWS

### 3.4.3.5. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

#### Prerequisites

- You have an existing **install-config.yaml** file.
- You reviewed the sites that your cluster requires access to and determined whether any of them need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. You added sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.

  NOTE

  The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.

  For installations on Amazon Web Services (AWS), Google Cloud, Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

#### Procedure

1. Edit your **install-config.yaml** file and add the proxy settings. For example:

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: ec2.<aws_region>.amazonaws.com,elasticloadbalancing.
<aws_region>.amazonaws.com,s3.<aws_region>.amazonaws.com 3
additionalTrustBundle: | 4
```

```
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**.

2. A proxy URL to use for creating HTTPS connections outside the cluster.

3. A comma-separated list of destination domain names, IP addresses, or other network CIDRs to exclude from proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use * to bypass the proxy for all destinations. If you have added the Amazon **EC2**,**Elastic Load Balancing**, and **S3** VPC endpoints to your VPC, you must add these endpoints to the **noProxy** field.

4. If provided, the installation program generates a config map that is named **user-ca-bundle** in the **openshift-config** namespace that contains one or more additional CA certificates that are required for proxying HTTPS connections. The Cluster Network Operator then creates a **trusted-ca-bundle** config map that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this config map is referenced in the **trustedCA** field of the **Proxy** object. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle.

5. Optional: The policy to determine the configuration of the **Proxy** object to reference the **user-ca-bundle** config map in the **trustedCA** field. The allowed values are **Proxyonly** and **Always**. Use **Proxyonly** to reference the **user-ca-bundle** config map only when **http/https** proxy is configured. Use **Always** to always reference the **user-ca-bundle** config map. The default value is **Proxyonly**.

NOTE

The installation program does not support the proxy **readinessEndpoints** field.

NOTE

If the installer times out, restart and then complete the deployment by using the **wait-for** command of the installer. For example:

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. Save the file and reference it when installing OpenShift Container Platform.

The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster Proxy** object is still created, but it will have a nil **spec**.

NOTE

Only the **Proxy** object named **cluster** is supported, and no additional proxies can be created.

### 3.4.4. Alternatives to storing administrator-level secrets in the kube-system project

By default, administrator secrets are stored in the **kube-system** project. If you configured the **credentialsMode** parameter in the **install-config.yaml** file to **Manual**, you must use one of the following alternatives:

- To manage long-term cloud credentials manually, follow the procedure in Manually creating long-term credentials.
- To implement short-term credentials that are managed outside the cluster for individual components, follow the procedures in Configuring an AWS cluster to use short-term credentials.

#### 3.4.4.1. Manually creating long-term credentials

The Cloud Credential Operator (CCO) can be put into manual mode prior to installation in environments where the cloud identity and access management (IAM) APIs are not reachable, or the administrator prefers not to store an administrator-level credential secret in the cluster **kube-system** namespace.

**Procedure**

1. If you did not set the **credentialsMode** parameter in the **install-config.yaml** configuration file to **Manual**, modify the value as shown:

   **Sample configuration file snippet**

   ```
   apiVersion: v1
   baseDomain: example.com
   credentialsMode: Manual
   # ...
   ```

2. If you have not previously created installation manifest files, do so by running the following command:

   ```
   $ openshift-install create manifests --dir <installation_directory>
   ```

   where **<installation_directory>** is the directory in which the installation program creates files.

3. Set a **$RELEASE_IMAGE** variable with the release image from your installation file by running the following command:

   ```
   $ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
   ```

4. Extract the list of **CredentialsRequest** custom resources (CRs) from the OpenShift Container Platform release image by running the following command:

   ```
   $ oc adm release extract \
     --from=$RELEASE_IMAGE \
     --credentials-requests \
     --included \ 1
     --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
     --to=<path_to_directory_for_credentials_requests> 3
   ```

1 The **--included** parameter includes only the manifests that your specific cluster configuration requires.

2 Specify the location of the **install-config.yaml** file.

3 Specify the path to the directory where you want to store the **CredentialsRequest** objects. If the specified directory does not exist, this command creates it.

This command creates a YAML file for each **CredentialsRequest** object.

**Sample CredentialsRequest object**

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
    - effect: Allow
      action:
      - iam:GetUser
      - iam:GetUserPolicy
      - iam:ListAccessKeys
      resource: "*"
  ...
```

5. Create YAML files for secrets in the **openshift-install** manifests directory that you generated previously. The secrets must be stored using the namespace and secret name defined in the **spec.secretRef** for each **CredentialsRequest** object.

**Sample CredentialsRequest object with secrets**

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
    - effect: Allow
      action:
      - s3:CreateBucket
      - s3:DeleteBucket
      resource: "*"
      ...
  secretRef:
    name: <component_secret>
    namespace: <component_namespace>
  ...
```

**Sample Secret object**

```
apiVersion: v1
kind: Secret
metadata:
  name: <component_secret>
  namespace: <component_namespace>
data:
  aws_access_key_id: <base64_encoded_aws_access_key_id>
  aws_secret_access_key: <base64_encoded_aws_secret_access_key>
```

IMPORTANT

Before upgrading a cluster that uses manually maintained credentials, you must ensure that the CCO is in an upgradeable state.

### 3.4.4.2. Configuring an AWS cluster to use short-term credentials

To install a cluster that is configured to use the AWS Security Token Service (STS), you must configure the CCO utility and create the required AWS resources for your cluster.

#### 3.4.4.2.1. Configuring the Cloud Credential Operator utility

To create and manage cloud credentials from outside of the cluster when the Cloud Credential Operator (CCO) is operating in manual mode, extract and prepare the CCO utility (**ccoctl**) binary.

NOTE

The **ccoctl** utility is a Linux binary that must run in a Linux environment.

**Prerequisites**

- You have access to an OpenShift Container Platform account with cluster administrator access.
- You have installed the OpenShift CLI (**oc**).
- You have created an AWS account for the **ccoctl** utility to use with the following permissions:
  **Required iam permissions**
  - **iam:CreateOpenIDConnectProvider**
  - **iam:CreateRole**
  - **iam:DeleteOpenIDConnectProvider**
  - **iam:DeleteRole**
  - **iam:DeleteRolePolicy**

- **iam:GetOpenIDConnectProvider**
- **iam:GetRole**
- **iam:GetUser**
- **iam:ListOpenIDConnectProviders**
- **iam:ListRolePolicies**
- **iam:ListRoles**
- **iam:PutRolePolicy**
- **iam:TagOpenIDConnectProvider**
- **iam:TagRole**

Required **s3** permissions

- **s3:CreateBucket**
- **s3:DeleteBucket**
- **s3:DeleteObject**
- **s3:GetBucketAcl**
- **s3:GetBucketTagging**
- **s3:GetObject**
- **s3:GetObjectAcl**
- **s3:GetObjectTagging**
- **s3:ListBucket**
- **s3:PutBucketAcl**
- **s3:PutBucketPolicy**
- **s3:PutBucketPublicAccessBlock**
- **s3:PutBucketTagging**
- **s3:PutObject**
- **s3:PutObjectAcl**
- **s3:PutObjectTagging**

Required **cloudfront** permissions

- **cloudfront:ListCloudFrontOriginAccessIdentities**

- **cloudfront:ListDistributions**
- **cloudfront:ListTagsForResource**

- If you plan to store the OIDC configuration in a private S3 bucket that is accessed by the IAM identity provider through a public CloudFront distribution URL, the AWS account that runs the **ccoctl** utility requires the following additional permissions:
  - **cloudfront:CreateCloudFrontOriginAccessIdentity**
  - **cloudfront:CreateDistribution**
  - **cloudfront:DeleteCloudFrontOriginAccessIdentity**
  - **cloudfront:DeleteDistribution**
  - **cloudfront:GetCloudFrontOriginAccessIdentity**
  - **cloudfront:GetCloudFrontOriginAccessIdentityConfig**
  - **cloudfront:GetDistribution**
  - **cloudfront:TagResource**
  - **cloudfront:UpdateDistribution**

  NOTE

  These additional permissions support the use of the **--create-private-s3-bucket** option when processing credentials requests with the **ccoctl aws create-all** command.

Procedure

1. Set a variable for the OpenShift Container Platform release image by running the following command:

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. Obtain the CCO container image from the OpenShift Container Platform release image by running the following command:

```
$ CCO_IMAGE=$(oc adm release info --image-for='cloud-credential-operator'
$RELEASE_IMAGE -a ~/.pull-secret)
```

   NOTE

   Ensure that the architecture of the **$RELEASE_IMAGE** matches the architecture of the environment in which you will use the **ccoctl** tool.

3. Extract the **ccoctl** binary from the CCO container image within the OpenShift Container Platform release image by running the following command:

```
$ oc image extract $CCO_IMAGE \
--file="/usr/bin/ccoctl.<rhel_version>" \ 1
-a ~/.pull-secret
```

1 For **<rhel_version>**, specify the value that corresponds to the version of Red Hat Enterprise Linux (RHEL) that the host uses. If no value is specified, **ccoctl.rhel8** is used by default. The following values are valid:

- **rhel8**: Specify this value for hosts that use RHEL 8.
- **rhel9**: Specify this value for hosts that use RHEL 9.

NOTE

The **ccoctl** binary is created in the directory from where you executed the command and not in **/usr/bin/**. You must rename the directory or move the **ccoctl.<rhel_version>** binary to **ccoctl**.

4. Change the permissions to make **ccoctl** executable by running the following command:

```
$ chmod 775 ccoctl
```

### Verification

- To verify that **ccoctl** is ready to use, display the help file. Use a relative file name when you run the command, for example:

```
$ ./ccoctl
```

#### Example output

```
OpenShift credentials provisioning tool

Usage:
  ccoctl [command]

Available Commands:
  aws          Manage credentials objects for AWS cloud
  azure        Manage credentials objects for Azure
  gcp          Manage credentials objects for Google cloud
  help         Help about any command
  ibmcloud     Manage credentials objects for {ibm-cloud-title}
  nutanix      Manage credentials objects for Nutanix

Flags:
  -h, --help   help for ccoctl

Use "ccoctl [command] --help" for more information about a command.
```

#### 3.4.4.2.2. Creating AWS resources with the Cloud Credential Operator utility

You have the following options when creating AWS resources:

- You can use the **ccoctl aws create-all** command to create the AWS resources automatically. This is the quickest way to create the resources. See Creating AWS resources with a single command.
- If you need to review the JSON files that the **ccoctl** tool creates before modifying AWS resources, or if the process the **ccoctl** tool uses to create AWS resources automatically does not meet the requirements of your organization, you can create the AWS resources individually. See Creating AWS resources individually .

###### 3.4.4.2.2.1. Creating AWS resources with a single command

If the process the **ccoctl** tool uses to create AWS resources automatically meets the requirements of your organization, you can use the **ccoctl aws create-all** command to automate the creation of AWS resources.

Otherwise, you can create the AWS resources individually. For more information, see "Creating AWS resources individually".

NOTE

By default, **ccoctl** creates objects in the directory in which the commands are run. To create the objects in a different directory, use the **--output-dir** flag. This procedure uses **<path_to_ccoctl_output_dir>** to refer to this directory.

#### Prerequisites

You must have:

- Extracted and prepared the **ccoctl** binary.

#### Procedure

1. Set a **$RELEASE_IMAGE** variable with the release image from your installation file by running the following command:

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. Extract the list of **CredentialsRequest** objects from the OpenShift Container Platform release image by running the following command:

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

1 The **--included** parameter includes only the manifests that your specific cluster configuration requires.

2 Specify the location of the **install-config.yaml** file.

3

Specify the path to the directory where you want to store the **CredentialsRequest** objects. If the specified directory does not exist, this command creates it.

> **NOTE**
>
> This command might take a few moments to run.

3. Use the **ccoctl** tool to process all **CredentialsRequest** objects by running the following command:

```
$ ccoctl aws create-all \
  --name=<name> \ 1
  --region=<aws_region> \ 2
  --credentials-requests-dir=<path_to_credentials_requests_directory> \ 3
  --output-dir=<path_to_ccoctl_output_dir> \ 4
  --create-private-s3-bucket 5
```

1 Specify the name used to tag any cloud resources that are created for tracking.

2 Specify the AWS region in which cloud resources will be created.

3 Specify the directory containing the files for the component **CredentialsRequest** objects.

4 Optional: Specify the directory in which you want the **ccoctl** utility to create objects. By default, the utility creates objects in the directory in which the commands are run.

5 Optional: By default, the **ccoctl** utility stores the OpenID Connect (OIDC) configuration files in a public S3 bucket and uses the S3 URL as the public OIDC endpoint. To store the OIDC configuration in a private S3 bucket that is accessed by the IAM identity provider through a public CloudFront distribution URL instead, use the **--create-private-s3-bucket** parameter.

> **NOTE**
>
> If your cluster uses Technology Preview features that are enabled by the **TechPreviewNoUpgrade** feature set, you must include the **--enable-tech-preview** parameter.

## Verification

- To verify that the OpenShift Container Platform secrets are created, list the files in the **<path_to_ccoctl_output_dir>/manifests** directory:

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

### Example output

```
cluster-authentication-02-config.yaml
openshift-cloud-credential-operator-cloud-credential-operator-iam-ro-creds-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capa-manager-bootstrap-credentials-credentials.yaml
```

```
openshift-cluster-csi-drivers-ebs-cloud-credentials-credentials.yaml
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-aws-cloud-credentials-credentials.yaml
```

You can verify that the IAM roles are created by querying AWS. For more information, refer to AWS documentation on listing IAM roles.

##### 3.4.4.2.2.2. Creating AWS resources individually

You can use the **ccoctl** tool to create AWS resources individually. This option might be useful for an organization that shares the responsibility for creating these resources among different users or departments.

Otherwise, you can use the **ccoctl aws create-all** command to create the AWS resources automatically. For more information, see "Creating AWS resources with a single command".

NOTE

By default, **ccoctl** creates objects in the directory in which the commands are run. To create the objects in a different directory, use the **--output-dir** flag. This procedure uses **<path_to_ccoctl_output_dir>** to refer to this directory.

Some **ccoctl** commands make AWS API calls to create or modify AWS resources. You can use the **--dry-run** flag to avoid making API calls. Using this flag creates JSON files on the local file system instead. You can review and modify the JSON files and then apply them with the AWS CLI tool using the **--cli-input-json** parameters.

**Prerequisites**

- Extract and prepare the **ccoctl** binary.

**Procedure**

1. Generate the public and private RSA key files that are used to set up the OpenID Connect provider for the cluster by running the following command:

```
$ ccoctl aws create-key-pair
```

**Example output**

```
2021/04/13 11:01:02 Generating RSA keypair
2021/04/13 11:01:03 Writing private key to /<path_to_ccoctl_output_dir>/serviceaccount-signer.private
2021/04/13 11:01:03 Writing public key to /<path_to_ccoctl_output_dir>/serviceaccount-signer.public
2021/04/13 11:01:03 Copying signing key for use by installer
```

where **serviceaccount-signer.private** and **serviceaccount-signer.public** are the generated key files.

This command also creates a private key that the cluster requires during installation in **/<path_to_ccoctl_output_dir>/tls/bound-service-account-signing-key.key**.

2. Create an OpenID Connect identity provider and S3 bucket on AWS by running the following command:

```
$ ccoctl aws create-identity-provider \
  --name=<name> \ 1
  --region=<aws_region> \ 2
  --public-key-file=<path_to_ccoctl_output_dir>/serviceaccount-signer.public 3
```

1 **<name>** is the name used to tag any cloud resources that are created for tracking.

2 **<aws-region>** is the AWS region in which cloud resources will be created.

3 **<path_to_ccoctl_output_dir>** is the path to the public key file that the **ccoctl aws create-key-pair** command generated.

**Example output**

```
2021/04/13 11:16:09 Bucket <name>-oidc created
2021/04/13 11:16:10 OpenID Connect discovery document in the S3 bucket <name>-oidc at
.well-known/openid-configuration updated
2021/04/13 11:16:10 Reading public key
2021/04/13 11:16:10 JSON web key set (JWKS) in the S3 bucket <name>-oidc at keys.json
updated
2021/04/13 11:16:18 Identity Provider created with ARN: arn:aws:iam::
<aws_account_id>:oidc-provider/<name>-oidc.s3.<aws_region>.amazonaws.com
```

where **openid-configuration** is a discovery document and **keys.json** is a JSON web key set file.

This command also creates a YAML configuration file in **/<path_to_ccoctl_output_dir>/manifests/cluster-authentication-02-config.yaml**. This file sets the issuer URL field for the service account tokens that the cluster generates, so that the AWS IAM identity provider trusts the tokens.

3. Create IAM roles for each component in the cluster:

   a. Set a **$RELEASE_IMAGE** variable with the release image from your installation file by running the following command:

   ```
   $ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
   ```

   b. Extract the list of **CredentialsRequest** objects from the OpenShift Container Platform release image:

   ```
   $ oc adm release extract \
     --from=$RELEASE_IMAGE \
     --credentials-requests \
     --included \ 1
     --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \
   2
     --to=<path_to_directory_for_credentials_requests> 3
   ```

   1 The **--included** parameter includes only the manifests that your specific cluster configuration requires.

2 Specify the location of the **install-config.yaml** file.

3 Specify the path to the directory where you want to store the **CredentialsRequest** objects. If the specified directory does not exist, this command creates it.

c. Use the **ccoctl** tool to process all **CredentialsRequest** objects by running the following command:

```
$ ccoctl aws create-iam-roles \
  --name=<name> \
  --region=<aws_region> \
  --credentials-requests-dir=<path_to_credentials_requests_directory> \
  --identity-provider-arn=arn:aws:iam::<aws_account_id>:oidc-provider/<name>-oidc.s3.
<aws_region>.amazonaws.com
```

NOTE

For AWS environments that use alternative IAM API endpoints, such as GovCloud, you must also specify your region with the **--region** parameter.

If your cluster uses Technology Preview features that are enabled by the **TechPreviewNoUpgrade** feature set, you must include the **--enable-tech-preview** parameter.

For each **CredentialsRequest** object, **ccoctl** creates an IAM role with a trust policy that is tied to the specified OIDC identity provider, and a permissions policy as defined in each **CredentialsRequest** object from the OpenShift Container Platform release image.

### Verification

- To verify that the OpenShift Container Platform secrets are created, list the files in the **<path_to_ccoctl_output_dir>/manifests** directory:

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

#### Example output

```
cluster-authentication-02-config.yaml
openshift-cloud-credential-operator-cloud-credential-operator-iam-ro-creds-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capa-manager-bootstrap-credentials-credentials.yaml
openshift-cluster-csi-drivers-ebs-cloud-credentials-credentials.yaml
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-aws-cloud-credentials-credentials.yaml
```

You can verify that the IAM roles are created by querying AWS. For more information, refer to AWS documentation on listing IAM roles.

#### 3.4.4.2.3. Incorporating the Cloud Credential Operator utility manifests

To implement short-term security credentials managed outside the cluster for individual components, you must move the manifest files that the Cloud Credential Operator utility (**ccoctl**) created to the correct directories for the installation program.

### Prerequisites

- You have configured an account with the cloud platform that hosts your cluster.
- You have configured the Cloud Credential Operator utility (**ccoctl**).
- You have created the cloud provider resources that are required for your cluster with the **ccoctl** utility.

### Procedure

1. If you did not set the **credentialsMode** parameter in the **install-config.yaml** configuration file to **Manual**, modify the value as shown:

   **Sample configuration file snippet**

   ```
   apiVersion: v1
   baseDomain: example.com
   credentialsMode: Manual
   # ...
   ```

2. If you have not previously created installation manifest files, do so by running the following command:

   ```
   $ openshift-install create manifests --dir <installation_directory>
   ```

   where **<installation_directory>** is the directory in which the installation program creates files.

3. Copy the manifests that the **ccoctl** utility generated to the **manifests** directory that the installation program created by running the following command:

   ```
   $ cp /<path_to_ccoctl_output_dir>/manifests/* ./manifests/
   ```

4. Copy the **tls** directory that contains the private key to the installation directory:

   ```
   $ cp -a /<path_to_ccoctl_output_dir>/tls .
   ```

## 3.4.5. Cluster Network Operator configuration

The configuration for the cluster network is specified as part of the Cluster Network Operator (CNO) configuration and stored in a custom resource (CR) object that is named **cluster**. The CR specifies the fields for the **Network** API in the **operator.openshift.io** API group.

The CNO configuration inherits the following fields during cluster installation from the **Network** API in the **Network.config.openshift.io** API group:

**clusterNetwork**

IP address pools from which pod IP addresses are allocated.

**serviceNetwork**

IP address pool for services.

**defaultNetwork.type**

Cluster network plugin. **OVNKubernetes** is the only supported plugin during installation.

You can specify the cluster network plugin configuration for your cluster by setting the fields for the **defaultNetwork** object in the CNO object named **cluster**.

### 3.4.5.1. Cluster Network Operator configuration object

The fields for the Cluster Network Operator (CNO) are described in the following table:

Table 3.3. Cluster Network Operator configuration object

| Field | Type | Description |
|---|---|---|
| **metadata.name** | **string** | The name of the CNO object. This name is always **cluster**. |
| **spec.clusterNetwork** | **array** | A list specifying the blocks of IP addresses from which pod IP addresses are allocated and the subnet prefix length assigned to each individual node in the cluster. For example: <br>`spec:`<br>`  clusterNetwork:`<br>`  - cidr: 10.128.0.0/19`<br>`    hostPrefix: 23`<br>`  - cidr: 10.128.32.0/19`<br>`    hostPrefix: 23` |
| **spec.serviceNetwork** | **array** | A block of IP addresses for services. The OVN-Kubernetes network plugin supports only a single IP address block for the service network. For example: <br>`spec:`<br>`  serviceNetwork:`<br>`  - 172.30.0.0/14`<br>You can customize this field only in the **install-config.yaml** file before you create the manifests. The value is read-only in the manifest file. |
| **spec.defaultNetwork** | **object** | Configures the network plugin for the cluster network. |

| Field | Type | Description |
|---|---|---|
| **spec.additionalRoutingCapabilities.providers** | **array** | This setting enables a dynamic routing provider. The FRR routing capability provider is required for the route advertisement feature. The only supported value is **FRR**.<br><br>• **FRR**: The FRR routing provider<br><br>`spec:`<br>`  additionalRoutingCapabilities:`<br>`    providers:`<br>`    - FRR` |

IMPORTANT

For a cluster that needs to deploy objects across multiple networks, ensure that you specify the same value for the **clusterNetwork.hostPrefix** parameter for each network type that is defined in the **install-config.yaml** file. Setting a different value for each **clusterNetwork.hostPrefix** parameter can impact the OVN-Kubernetes network plugin, where the plugin cannot effectively route object traffic among different nodes.

#### 3.4.5.1.1. defaultNetwork object configuration

The values for the **defaultNetwork** object are defined in the following table:

Table 3.4. **defaultNetwork** object

| Field | Type | Description |
|---|---|---|
| **type** | **string** | **OVNKubernetes**. The Red Hat OpenShift Networking network plugin is selected during installation. This value cannot be changed after cluster installation.<br><br>NOTE<br><br>OpenShift Container Platform uses the OVN-Kubernetes network plugin by default. |
| **ovnKubernetesConfig** | **object** | This object is only valid for the OVN-Kubernetes network plugin. |

##### 3.4.5.1.1.1. Configuration for the OVN-Kubernetes network plugin

The following table describes the configuration fields for the OVN-Kubernetes network plugin:

Table 3.5. **ovnKubernetesConfig** object

| Field | Type | Description |
|---|---|---|
| **mtu** | **integer** | The maximum transmission unit (MTU) for the Geneve (Generic Network Virtualization Encapsulation) overlay network. This is detected automatically based on the MTU of the primary network interface. You do not normally need to override the detected MTU.<br><br>If the auto-detected value is not what you expect it to be, confirm that the MTU on the primary network interface on your nodes is correct. You cannot use this option to change the MTU value of the primary network interface on the nodes.<br><br>If your cluster requires different MTU values for different nodes, you must set this value to **100** less than the lowest MTU value in your cluster. For example, if some nodes in your cluster have an MTU of **9001**, and some have an MTU of**1500**, you must set this value to **1400**. |
| **genevePort** | **integer** | The port to use for all Geneve packets. The default value is **6081**. This value cannot be changed after cluster installation. |
| **ipsecConfig** | **object** | Specify a configuration object for customizing the IPsec configuration. |
| **ipv4** | **object** | Specifies a configuration object for IPv4 settings. |
| **ipv6** | **object** | Specifies a configuration object for IPv6 settings. |
| **policyAuditConfig** | **object** | Specify a configuration object for customizing network policy audit logging. If unset, the defaults audit log settings are used. |
| **routeAdvertisements** | **string** | Specifies whether to advertise cluster network routes. The default value is **Disabled**.<br><br>• **Enabled**: Import routes to the cluster network and advertise cluster network routes as configured in **RouteAdvertisements** objects.<br><br>• **Disabled**: Do not import routes to the cluster network or advertise cluster network routes. |

| Field | Type | Description |
|---|---|---|
| **gatewayConfig** | **object** | Optional: Specify a configuration object for customizing how egress traffic is sent to the node gateway. Valid values are **Shared** and **Local**. The default value is **Shared**. In the default setting, the Open vSwitch (OVS) outputs traffic directly to the node IP interface. In the **Local** setting, it traverses the host network; consequently, it gets applied to the routing table of the host.<br><br>NOTE<br><br>While migrating egress traffic, you can expect some disruption to workloads and service traffic until the Cluster Network Operator (CNO) successfully rolls out the changes. |

Table 3.6. **ovnKubernetesConfig.ipv4** object

| Field | Type | Description |
|---|---|---|
| **internalTransitSwitchSubnet** | string | If your existing network infrastructure overlaps with the **100.88.0.0/16** IPv4 subnet, you can specify a different IP address range for internal use by OVN-Kubernetes. The subnet for the distributed transit switch that enables east-west traffic. This subnet cannot overlap with any other subnets used by OVN-Kubernetes or on the host itself. It must be large enough to accommodate one IP address per node in your cluster.<br><br>The default value is **100.88.0.0/16**. |
| **internalJoinSubnet** | string | If your existing network infrastructure overlaps with the **100.64.0.0/16** IPv4 subnet, you can specify a different IP address range for internal use by OVN-Kubernetes. You must ensure that the IP address range does not overlap with any other subnet used by your OpenShift Container Platform installation. The IP address range must be larger than the maximum number of nodes that can be added to the cluster. For example, if the **clusterNetwork.cidr** value is **10.128.0.0/14** and the **clusterNetwork.hostPrefix** value is **/23**, then the maximum number of nodes is **2^(23-14)=512**.<br><br>The default value is **100.64.0.0/16**. |

Table 3.7. **ovnKubernetesConfig.ipv6** object

| Field | Type | Description |
|---|---|---|

| Field | Type | Description |
|---|---|---|
| **internalTransitSwitchSubnet** | string | If your existing network infrastructure overlaps with the **fd97::/64** IPv6 subnet, you can specify a different IP address range for internal use by OVN-Kubernetes. The subnet for the distributed transit switch that enables east-west traffic. This subnet cannot overlap with any other subnets used by OVN-Kubernetes or on the host itself. It must be large enough to accommodate one IP address per node in your cluster.<br><br>The default value is **fd97::/64**. |
| **internalJoinSubnet** | string | If your existing network infrastructure overlaps with the **fd98::/64** IPv6 subnet, you can specify a different IP address range for internal use by OVN-Kubernetes. You must ensure that the IP address range does not overlap with any other subnet used by your OpenShift Container Platform installation. The IP address range must be larger than the maximum number of nodes that can be added to the cluster.<br><br>The default value is **fd98::/64**. |

Table 3.8. **policyAuditConfig** object

| Field | Type | Description |
|---|---|---|
| **rateLimit** | integer | The maximum number of messages to generate every second per node. The default value is **20** messages per second. |
| **maxFileSize** | integer | The maximum size for the audit log in bytes. The default value is **50000000** or 50 MB. |
| **maxLogFiles** | integer | The maximum number of log files that are retained. |
| **destination** | string | One of the following additional audit log targets:<br><br>**libc**<br>The libc **syslog()** function of the journald process on the host.<br>**udp:<host>:<port>**<br>A syslog server. Replace **<host>:<port>** with the host and port of the syslog server.<br>**unix:<file>**<br>A Unix Domain Socket file specified by **<file>**.<br>**null**<br>Do not send the audit logs to any additional target. |
| **syslogFacility** | string | The syslog facility, such as **kern**, as defined by RFC5424. The default value is **local0**. |

Table 3.9. **gatewayConfig** object

| Field | Type | Description |
|---|---|---|
| **routingViaHost** | **boolean** | Set this field to **true** to send egress traffic from pods to the host networking stack. For highly-specialized installations and applications that rely on manually configured routes in the kernel routing table, you might want to route egress traffic to the host networking stack. By default, egress traffic is processed in OVN to exit the cluster and is not affected by specialized routes in the kernel routing table. The default value is **false**.<br><br>This field has an interaction with the Open vSwitch hardware offloading feature. If you set this field to **true**, you do not receive the performance benefits of the offloading because egress traffic is processed by the host networking stack. |
| **ipForwarding** | **object** | You can control IP forwarding for all traffic on OVN-Kubernetes managed interfaces by using the **ipForwarding** specification in the **Network** resource. Specify **Restricted** to only allow IP forwarding for Kubernetes related traffic. Specify **Global** to allow forwarding of all IP traffic. For new installations, the default is **Restricted**. For updates to OpenShift Container Platform 4.14 or later, the default is **Global**.<br><br>NOTE<br><br>The default value of **Restricted** sets the IP forwarding to drop. |
| **ipv4** | **object** | Optional: Specify an object to configure the internal OVN-Kubernetes masquerade address for host to service traffic for IPv4 addresses. |
| **ipv6** | **object** | Optional: Specify an object to configure the internal OVN-Kubernetes masquerade address for host to service traffic for IPv6 addresses. |

Table 3.10. **gatewayConfig.ipv4** object

| Field | Type | Description |
|---|---|---|

| Field | Type | Description |
|---|---|---|
| **internalMasqueradeSubnet** | **string** | The masquerade IPv4 addresses that are used internally to enable host to service traffic. The host is configured with these IP addresses as well as the shared gateway bridge interface. The default value is **169.254.169.0/29**.<br><br>IMPORTANT<br><br>For OpenShift Container Platform 4.17 and later versions, clusters use **169.254.0.0/17** as the default masquerade subnet. For upgraded clusters, there is no change to the default masquerade subnet. |

Table 3.11. **gatewayConfig.ipv6** object

| Field | Type | Description |
|---|---|---|
| **internalMasqueradeSubnet** | **string** | The masquerade IPv6 addresses that are used internally to enable host to service traffic. The host is configured with these IP addresses as well as the shared gateway bridge interface. The default value is **fd69::/125**.<br><br>IMPORTANT<br><br>For OpenShift Container Platform 4.17 and later versions, clusters use **fd69::/112** as the default masquerade subnet. For upgraded clusters, there is no change to the default masquerade subnet. |

Table 3.12. **ipsecConfig** object

| Field | Type | Description |
|---|---|---|
| **mode** | **string** | Specifies the behavior of the IPsec implementation. Must be one of the following values:<br>• **Disabled**: IPsec is not enabled on cluster nodes.<br>• **External**: IPsec is enabled for network traffic with external hosts.<br>• **Full**: IPsec is enabled for pod traffic and network traffic with external hosts. |

### Example OVN-Kubernetes configuration with IPSec enabled

```
defaultNetwork:
  type: OVNKubernetes
  ovnKubernetesConfig:
    mtu: 1400
    genevePort: 6081
    ipsecConfig:
      mode: Full
```

## 3.4.6. Specifying advanced network configuration

You can use advanced network configuration for your network plugin to integrate your cluster into your existing network environment.

You can specify advanced network configuration only before you install the cluster.

**IMPORTANT**

Customizing your network configuration by modifying the OpenShift Container Platform manifest files created by the installation program is not supported. Applying a manifest file that you create, as in the following procedure, is supported.

### Prerequisites

- You have created the **install-config.yaml** file and completed any modifications to it.

### Procedure

1. Change to the directory that contains the installation program and create the manifests:

   ```
   $ ./openshift-install create manifests --dir <installation_directory> (1)
   ```

   (1) **<installation_directory>** specifies the name of the directory that contains the **install-config.yaml** file for your cluster.

2. Create a stub manifest file for the advanced network configuration that is named **cluster-network-03-config.yml** in the **<installation_directory>/manifests/** directory:

   ```
   apiVersion: operator.openshift.io/v1
   kind: Network
   metadata:
     name: cluster
   spec:
   ```

3. Specify the advanced network configuration for your cluster in the **cluster-network-03-config.yml** file, such as in the following example:

   #### Enable IPsec for the OVN-Kubernetes network provider

   ```
   apiVersion: operator.openshift.io/v1
   kind: Network
   ```

```
metadata:
  name: cluster
spec:
  defaultNetwork:
    ovnKubernetesConfig:
      ipsecConfig:
        mode: Full
```

4. Optional: Back up the **manifests/cluster-network-03-config.yml** file. The installation program consumes the **manifests/** directory when you create the Ignition config files.

5. Remove the Kubernetes manifest files that define the control plane machines and compute **MachineSets**:

   ```
   $ rm -f openshift/99_openshift-cluster-api_master-machines-*.yaml openshift/99_openshift-cluster-api_worker-machineset-*.yaml
   ```

   Because you create and manage these resources yourself, you do not have to initialize them.

   - You can preserve the **MachineSet** files to create compute machines by using the machine API, but you must update references to them to match your environment.

NOTE

For more information on using a Network Load Balancer (NLB) on AWS, see Configuring Ingress cluster traffic on AWS using a Network Load Balancer.

### 3.4.7. Configuring an Ingress Controller Network Load Balancer on a new AWS cluster

You can create an Ingress Controller backed by an Amazon Web Services Network Load Balancer (NLB) on a new cluster in situations where you need more transparent networking capabilities.

#### Prerequisites

- Create and edit the **install-config.yaml** file. For instructions, see "Creating the installation configuration file" in the *Additonal resources* section.

#### Procedure

1. Change to the directory that contains the installation program and create the manifests:

   ```
   $ ./openshift-install create manifests --dir <installation_directory>
   ```

   - For **<installation_directory>**, specify the name of the directory that contains the **install-config.yaml** file for your cluster.

2. Create a file that is named **cluster-ingress-default-ingresscontroller.yaml** in the **<installation_directory>/manifests/** directory:

   ```
   $ touch <installation_directory>/manifests/cluster-ingress-default-ingresscontroller.yaml
   ```

   **<installation_directory>**

Specifies the directory name that contains the **manifests/** directory for your cluster.

3. Check the several network configuration files that exist in the **manifests/** directory by entering the following command:

```
$ ls <installation_directory>/manifests/cluster-ingress-default-ingresscontroller.yaml
```

**Example output**

```
cluster-ingress-default-ingresscontroller.yaml
```

4. Open the **cluster-ingress-default-ingresscontroller.yaml** file in an editor and enter a custom resource (CR) that describes the Operator configuration you want:

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  creationTimestamp: null
  name: default
  namespace: openshift-ingress-operator
spec:
  endpointPublishingStrategy:
    loadBalancer:
      scope: External
      providerParameters:
        type: AWS
        aws:
          type: NLB
    type: LoadBalancerService
```

5. Save the **cluster-ingress-default-ingresscontroller.yaml** file and quit the text editor.

6. Optional: Back up the **manifests/cluster-ingress-default-ingresscontroller.yaml** file because the installation program deletes the **manifests/** directory during cluster creation.

### 3.4.8. Configuring hybrid networking with OVN-Kubernetes

You can configure your cluster to use hybrid networking with the OVN-Kubernetes network plugin. This allows a hybrid cluster that supports different node networking configurations.

**NOTE**

This configuration is necessary to run both Linux and Windows nodes in the same cluster.

**Prerequisites**

- You defined **OVNKubernetes** for the **networking.networkType** parameter in the **install-config.yaml** file. See the installation documentation for configuring OpenShift Container Platform network customizations on your chosen cloud provider for more information.

**Procedure**

1. Change to the directory that contains the installation program and create the manifests:

```
$ ./openshift-install create manifests --dir <installation_directory>
```

where:

**<installation_directory>**

Specifies the name of the directory that contains the **install-config.yaml** file for your cluster.

2. Create a stub manifest file for the advanced network configuration that is named **cluster-network-03-config.yml** in the **<installation_directory>/manifests/** directory:

```
$ cat <<EOF > <installation_directory>/manifests/cluster-network-03-config.yml
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
EOF
```

where:

**<installation_directory>**

Specifies the directory name that contains the **manifests/** directory for your cluster.

3. Open the **cluster-network-03-config.yml** file in an editor and configure OVN-Kubernetes with hybrid networking, as in the following example:

### Specify a hybrid networking configuration

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    ovnKubernetesConfig:
      hybridOverlayConfig:
        hybridClusterNetwork: 1
        - cidr: 10.132.0.0/14
          hostPrefix: 23
        hybridOverlayVXLANPort: 9898 2
```

1 Specify the CIDR configuration used for nodes on the additional overlay network. The **hybridClusterNetwork** CIDR must not overlap with the **clusterNetwork** CIDR.

2 Specify a custom VXLAN port for the additional overlay network. This is required for running Windows nodes in a cluster installed on vSphere, and must not be configured for any other cloud provider. The custom port can be any open port excluding the default **6081** port. For more information on this requirement, see Pod-to-pod connectivity between hosts is broken in the Microsoft documentation.

NOTE

Windows Server Long-Term Servicing Channel (LTSC): Windows Server 2019 is not supported on clusters with a custom **hybridOverlayVXLANPort** value because this Windows server version does not support selecting a custom VXLAN port.

4. Save the **cluster-network-03-config.yml** file and quit the text editor.

5. Optional: Back up the **manifests/cluster-network-03-config.yml** file. The installation program deletes the **manifests/** directory when creating the cluster.

NOTE

For more information about using Linux and Windows nodes in the same cluster, see Understanding Windows container workloads.

### 3.4.9. Deploying the cluster

You can install OpenShift Container Platform on a compatible cloud platform.

IMPORTANT

You can run the **create cluster** command of the installation program only once, during initial installation.

#### Prerequisites

- You have configured an account with the cloud platform that hosts your cluster.
- You have the OpenShift Container Platform installation program and the pull secret for your cluster.
- You have verified that the cloud provider account on your host has the correct permissions to deploy the cluster. An account with incorrect permissions causes the installation process to fail with an error message that displays the missing permissions.

#### Procedure

1. In the directory that contains the installation program, initialize the cluster deployment by running the following command:

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 For **<installation_directory>**, specify the location of your customized **./install-config.yaml** file.

2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

2. Optional: Remove or disable the **AdministratorAccess** policy from the IAM account that you used to install the cluster.

NOTE

The elevated permissions provided by the **AdministratorAccess** policy are required only during installation.

### Verification

When the cluster deployment completes successfully:

- The terminal displays directions for accessing your cluster, including a link to the web console and credentials for the **kubeadmin** user.
- Credential information also outputs to **<installation_directory>/.openshift_install.log**.

IMPORTANT

Do not delete the installation program or the files that the installation program creates. Both are required to delete the cluster.

### Example output

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```

IMPORTANT

- The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.
- It is recommended that you use Ignition config files within 12 hours after they are generated because the 24-hour certificate rotates from 16 to 22 hours after the cluster is installed. By using the Ignition config files within 12 hours, you can avoid installation failure if the certificate update runs during installation.

## 3.4.10. Logging in to the cluster by using the CLI

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

### Prerequisites

- You deployed an OpenShift Container Platform cluster.
- You installed the OpenShift CLI (**oc**).

**Procedure**

1. Export the **kubeadmin** credentials by running the following command:

   ```
   $ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
   ```

   1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration by running the following command:

   ```
   $ oc whoami
   ```

   **Example output**

   ```
   system:admin
   ```

### 3.4.11. Logging in to the cluster by using the web console

The **kubeadmin** user exists by default after an OpenShift Container Platform installation. You can log in to your cluster as the **kubeadmin** user by using the OpenShift Container Platform web console.

**Prerequisites**

- You have access to the installation host.
- You completed a cluster installation and all cluster Operators are available.

**Procedure**

1. Obtain the password for the **kubeadmin** user from the **kubeadmin-password** file on the installation host:

   ```
   $ cat <installation_directory>/auth/kubeadmin-password
   ```

   **NOTE**

   Alternatively, you can obtain the **kubeadmin** password from the **<installation_directory>/.openshift_install.log** log file on the installation host.

2. List the OpenShift Container Platform web console route:

   ```
   $ oc get routes -n openshift-console | grep 'console-openshift'
   ```

NOTE

Alternatively, you can obtain the OpenShift Container Platform route from the **<installation_directory>/.openshift_install.log** log file on the installation host.

**Example output**

```
console     console-openshift-console.apps.<cluster_name>.<base_domain>          console
https   reencrypt/Redirect   None
```

3. Navigate to the route detailed in the output of the preceding command in a web browser and log in as the **kubeadmin** user.

**Additional resources**

- See Accessing the web console for more details about accessing and understanding the OpenShift Container Platform web console.

### 3.4.12. Next steps

- Validating an installation.
- Customize your cluster.
- If necessary, you can Remote health reporting.
- If necessary, you can remove cloud provider credentials.

## 3.5. INSTALLING A CLUSTER ON AWS IN A DISCONNECTED ENVIRONMENT

In OpenShift Container Platform version 4.19, you can install a cluster on Amazon Web Services (AWS) in a restricted network by creating an internal mirror of the installation release content on an existing Amazon Virtual Private Cloud (VPC).

### 3.5.1. Prerequisites

- You reviewed details about the OpenShift Container Platform installation and update processes.
- You read the documentation on selecting a cluster installation method and preparing it for users.
- You mirrored the images for a disconnected installation to your registry and obtained the **imageContentSources** data for your version of OpenShift Container Platform.

IMPORTANT

Because the installation media is on the mirror host, you can use that computer to complete all installation steps.

- You have an existing VPC in AWS. When installing to a restricted network using installer-provisioned infrastructure, you cannot use the installer-provisioned VPC. You must use a user-provisioned VPC that satisfies one of the following requirements:
  - Contains the mirror registry
  - Has firewall rules or a peering connection to access the mirror registry hosted elsewhere
- You configured an AWS account to host the cluster.

> **IMPORTANT**
>
> If you have an AWS profile stored on your computer, it must not use a temporary session token that you generated while using a multi-factor authentication device. The cluster continues to use your current AWS credentials to create AWS resources for the entire life of the cluster, so you must use key-based, long-term credentials. To generate appropriate keys, see Managing Access Keys for IAM Users in the AWS documentation. You can supply the keys when you run the installation program.

- You downloaded the AWS CLI and installed it on your computer. See Install the AWS CLI Using the Bundled Installer (Linux, macOS, or UNIX) in the AWS documentation.
- If you use a firewall and plan to use the Telemetry service, you configured the firewall to allow the sites that your cluster requires access to.

> **NOTE**
>
> If you are configuring a proxy, be sure to also review this site list.

### 3.5.2. About installations in restricted networks

In OpenShift Container Platform 4.19, you can perform an installation that does not require an active connection to the internet to obtain software components. Restricted network installations can be completed using installer-provisioned infrastructure or user-provisioned infrastructure, depending on the cloud platform to which you are installing the cluster.

If you choose to perform a restricted network installation on a cloud platform, you still require access to its cloud APIs. Some cloud functions, like Amazon Web Service's Route 53 DNS and IAM services, require internet access. Depending on your network, you might require less internet access for an installation on bare metal hardware, Nutanix, or on VMware vSphere.

To complete a restricted network installation, you must create a registry that mirrors the contents of the OpenShift image registry and contains the installation media. You can create this registry on a mirror host, which can access both the internet and your closed network, or by using other methods that meet your restrictions.

#### 3.5.2.1. Additional limits

Clusters in restricted networks have the following additional limitations and restrictions:

- The **ClusterVersion** status includes an **Unable to retrieve available updates** error.
- By default, you cannot use the contents of the Developer Catalog because you cannot access the required image stream tags.

### 3.5.3. About using a custom VPC

In OpenShift Container Platform 4.19, you can deploy a cluster into existing subnets in an existing Amazon Virtual Private Cloud (VPC) in Amazon Web Services (AWS). By deploying OpenShift Container Platform into an existing AWS VPC, you might be able to avoid limit constraints in new accounts or more easily abide by the operational constraints that your company's guidelines set. If you cannot obtain the infrastructure creation permissions that are required to create the VPC yourself, use this installation option.

Because the installation program cannot know what other components are also in your existing subnets, it cannot choose subnet CIDRs and so forth on your behalf. You must configure networking for the subnets that you install your cluster to yourself.

#### 3.5.3.1. Requirements for using your VPC

The installation program no longer creates the following components:

- Internet gateways
- NAT gateways
- Subnets
- Route tables
- VPCs
- VPC DHCP options
- VPC endpoints

**NOTE**

The installation program requires that you use the cloud-provided DNS server. Using a custom DNS server is not supported and causes the installation to fail.

If you use a custom VPC, you must correctly configure it and its subnets for the installation program and the cluster to use. See Create a VPC in the Amazon Web Services documentation for more information about AWS VPC console wizard configurations and creating and managing an AWS VPC.

The installation program cannot:

- Subdivide network ranges for the cluster to use.
- Set route tables for the subnets.
- Set VPC options like DHCP.

You must complete these tasks before you install the cluster. See VPC networking components and Route tables for your VPC for more information on configuring networking in an AWS VPC.

Your VPC must meet the following characteristics:

- The VPC must not use the **kubernetes.io/cluster/.*: owned**, **Name**, and **openshift.io/cluster** tags.
  The installation program modifies your subnets to add the **kubernetes.io/cluster/.*: shared**

tag, so your subnets must have at least one free tag slot available for it. See Tag Restrictions in the AWS documentation to confirm that the installation program can add a tag to each subnet that you specify. You cannot use a **Name** tag, because it overlaps with the EC2 **Name** field and the installation fails.

- If you want to extend your OpenShift Container Platform cluster into an AWS Outpost and have an existing Outpost subnet, the existing subnet must use the **kubernetes.io/cluster/unmanaged: true** tag. If you do not apply this tag, the installation might fail due to the Cloud Controller Manager creating a service load balancer in the Outpost subnet, which is an unsupported configuration.

- You must enable the **enableDnsSupport** and **enableDnsHostnames** attributes in your VPC, so that the cluster can use the Route 53 zones that are attached to the VPC to resolve cluster's internal DNS records. See DNS Support in Your VPC in the AWS documentation.
  If you prefer to use your own Route 53 hosted private zone, you must associate the existing hosted zone with your VPC prior to installing a cluster. You can define your hosted zone using the **platform.aws.hostedZone** and **platform.aws.hostedZoneRole** fields in the **install-config.yaml** file. You can use a private hosted zone from another account by sharing it with the account where you install the cluster. If you use a private hosted zone from another account, you must use the **Passthrough** or **Manual** credentials mode.

If you are working in a disconnected environment, you are unable to reach the public IP addresses for EC2, ELB, and S3 endpoints. Depending on the level to which you want to restrict internet traffic during the installation, the following configuration options are available:

#### 3.5.3.1.1. Option 1: Create VPC endpoints

Create a VPC endpoint and attach it to the subnets that the clusters are using. Name the endpoints as follows:

- **ec2.<aws_region>.amazonaws.com**
- **elasticloadbalancing.<aws_region>.amazonaws.com**
- **s3.<aws_region>.amazonaws.com**

With this option, network traffic remains private between your VPC and the required AWS services.

#### 3.5.3.1.2. Option 2: Create a proxy without VPC endpoints

As part of the installation process, you can configure an HTTP or HTTPS proxy. With this option, internet traffic goes through the proxy to reach the required AWS services.

#### 3.5.3.1.3. Option 3: Create a proxy with VPC endpoints

As part of the installation process, you can configure an HTTP or HTTPS proxy with VPC endpoints. Create a VPC endpoint and attach it to the subnets that the clusters are using. Name the endpoints as follows:

- **ec2.<aws_region>.amazonaws.com**
- **elasticloadbalancing.<aws_region>.amazonaws.com**
- **s3.<aws_region>.amazonaws.com**

When configuring the proxy in the **install-config.yaml** file, add these endpoints to the **noProxy** field. With this option, the proxy prevents the cluster from accessing the internet directly. However, network traffic remains private between your VPC and the required AWS services.

### Required VPC components

You must provide a suitable VPC and subnets that allow communication to your machines.

| Component | AWS type | Description |
|---|---|---|
| VPC | • **AWS::EC2::VPC**<br>• **AWS::EC2::VPCEndpoint** | You must provide a public VPC for the cluster to use. The VPC uses an endpoint that references the route tables for each subnet to improve communication with the registry that is hosted in S3. |
| Public subnets | • **AWS::EC2::Subnet**<br>• **AWS::EC2::SubnetNetworkAclAssociation** | Your VPC must have public subnets for between 1 and 3 availability zones and associate them with appropriate Ingress rules. |
| Internet gateway | • **AWS::EC2::InternetGateway**<br>• **AWS::EC2::VPCGatewayAttachment**<br>• **AWS::EC2::RouteTable**<br>• **AWS::EC2::Route**<br>• **AWS::EC2::SubnetRouteTableAssociation**<br>• **AWS::EC2::NatGateway**<br>• **AWS::EC2::EIP** | You must have a public internet gateway, with public routes, attached to the VPC. In the provided templates, each public subnet has a NAT gateway with an EIP address. These NAT gateways allow cluster resources, like private subnet instances, to reach the internet and are not required for some restricted network or proxy scenarios. |
| Network access control | • **AWS::EC2::NetworkAcl**<br>• **AWS::EC2::NetworkAclEntry** | You must allow the VPC to access the following ports: |

| Port | Reason |
|---|---|
| **80** | Inbound HTTP traffic |
| **443** | Inbound HTTPS traffic |
| **22** | Inbound SSH traffic |

| Component | AWS type | Description | |
|---|---|---|---|
| | | **1024 - 65535** | Inbound ephemeral traffic |
| | | **0 - 65535** | Outbound ephemeral traffic |
| Private subnets | • **AWS::EC2::Subnet** <br> • **AWS::EC2::RouteTable** <br> • **AWS::EC2::SubnetRouteTableAssociation** | Your VPC can have private subnets. The provided CloudFormation templates can create private subnets for between 1 and 3 availability zones. If you use private subnets, you must provide appropriate routes and tables for them. | |

### 3.5.3.2. VPC validation

To ensure that the subnets that you provide are suitable, the installation program confirms the following data:

- All the subnets that you specify exist.
- You provide private subnets.
- The subnet CIDRs belong to the machine CIDR that you specified.
- You provide subnets for each availability zone. Each availability zone contains no more than one public and one private subnet. If you use a private cluster, provide only a private subnet for each availability zone. Otherwise, provide exactly one public and private subnet for each availability zone.
- You provide a public subnet for each private subnet availability zone. Machines are not provisioned in availability zones that you do not provide private subnets for.

If you destroy a cluster that uses an existing VPC, the VPC is not deleted. When you remove the OpenShift Container Platform cluster from a VPC, the **kubernetes.io/cluster/.*: shared** tag is removed from the subnets that it used.

### 3.5.3.3. Division of permissions

Starting with OpenShift Container Platform 4.3, you do not need all of the permissions that are required for an installation program-provisioned infrastructure cluster to deploy a cluster. This change mimics the division of permissions that you might have at your company: some individuals can create different resource in your clouds than others. For example, you might be able to create application-specific items, like instances, buckets, and load balancers, but not networking-related components such as VPCs, subnets, or ingress rules.

The AWS credentials that you use when you create your cluster do not need the networking permissions that are required to make VPCs and core networking components within the VPC, such as subnets, routing tables, internet gateways, NAT, and VPN. You still need permission to make the application resources that the machines within the cluster require, such as ELBs, security groups, S3 buckets, and nodes.

### 3.5.3.4. Isolation between clusters

If you deploy OpenShift Container Platform to an existing network, the isolation of cluster services is reduced in the following ways:

- You can install multiple OpenShift Container Platform clusters in the same VPC.
- ICMP ingress is allowed from the entire network.
- TCP 22 ingress (SSH) is allowed to the entire network.
- Control plane TCP 6443 ingress (Kubernetes API) is allowed to the entire network.
- Control plane TCP 22623 ingress (MCS) is allowed to the entire network.

## 3.5.4. Creating the installation configuration file

You can customize the OpenShift Container Platform cluster you install on Amazon Web Services (AWS).

**Prerequisites**

- You have the OpenShift Container Platform installation program and the pull secret for your cluster. For a restricted network installation, these files are on your mirror host.
- You have the **imageContentSources** values that were generated during mirror registry creation.
- You have obtained the contents of the certificate for your mirror registry.

**Procedure**

1. Create the **install-config.yaml** file.

   a. Change to the directory that contains the installation program and run the following command:

   ```
   $ ./openshift-install create install-config --dir <installation_directory>
   ```

   - **<installation_directory>**: For **<installation_directory>**, specify the directory name to store the files that the installation program creates.
     When specifying the directory:
   - Verify that the directory has the **execute** permission. This permission is required to run Terraform binaries under the installation directory.
   - Use an empty directory. Some installation assets, such as bootstrap X.509 certificates, have short expiration intervals, therefore you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

   b. At the prompts, provide the configuration details for your cloud:

      i. Optional: Select an SSH key to use to access your cluster machines.

NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

ii. Select **AWS** as the platform to target.

iii. If you do not have an Amazon Web Services (AWS) profile stored on your computer, enter the AWS access key ID and secret access key for the user that you configured to run the installation program.

iv. Select the AWS region to deploy the cluster to.

v. Select the base domain for the Route 53 service that you configured for your cluster.

vi. Enter a descriptive name for your cluster.

2. Edit the **install-config.yaml** file to give the additional information that is required for an installation in a restricted network.

   a. Update the **pullSecret** value to contain the authentication information for your registry:

```
pullSecret: '{"auths":{"<mirror_host_name>:5000": {"auth": "<credentials>","email":
"you@example.com"}}}'
```

   For **<mirror_host_name>**, specify the registry domain name that you specified in the certificate for your mirror registry, and for **<credentials>**, specify the base64-encoded user name and password for your mirror registry.

   b. Add the **additionalTrustBundle** parameter and value.

```
additionalTrustBundle: |
  -----BEGIN CERTIFICATE-----

ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
```

   The value must be the contents of the certificate file that you used for your mirror registry. The certificate file can be an existing, trusted certificate authority, or the self-signed certificate that you generated for the mirror registry.

   c. Define the subnets for the VPC to install the cluster in, as in the following example:

```
platform:
  aws:
    vpc:
      subnets:
        - id: subnet-<id_1>
        - id: subnet-<id_2>
        - id: subnet-<id_3>
```

   d. Add the image content resources, which resemble the following YAML excerpt:

```
imageContentSources:
```

```
- mirrors:
  - <mirror_host_name>:5000/<repo_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <mirror_host_name>:5000/<repo_name>/release
  source: registry.redhat.io/ocp/release
```

For these values, use the **imageContentSources** that you recorded during mirror registry creation.

e. Set the publishing strategy to **Internal**:

```
publish: Internal
```

By setting this option, you create an internal Ingress Controller and a private load balancer.

3. Make any other modifications to the **install-config.yaml** file that you require.
   For more information about the parameters, see "Installation configuration parameters".

4. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.

IMPORTANT

The **install-config.yaml** file is consumed during the installation process. If you want to reuse the file, you must back it up now.

### Additional resources

- Installation configuration parameters for AWS

### 3.5.4.1. Minimum resource requirements for cluster installation

Each created cluster must meet minimum requirements so that the cluster runs as expected.

Table 3.13. Minimum resource requirements

| Machine | Operating System | vCPU [1] | Virtual RAM | Storage | Input/Output Per Second (IOPS)[2] |
|---|---|---|---|---|---|
| Bootstrap | RHCOS | 4 | 16 GB | 100 GB | 300 |
| Control plane | RHCOS | 4 | 16 GB | 100 GB | 300 |
| Compute | RHCOS, RHEL 8.6 and later [3] | 2 | 8 GB | 100 GB | 300 |

1. One vCPU is equivalent to one physical core when simultaneous multithreading (SMT), or Hyper-Threading, is not enabled. When enabled, use the following formula to calculate the corresponding ratio: (threads per core × cores) × sockets = vCPUs.

2. OpenShift Container Platform and Kubernetes are sensitive to disk performance, and faster storage is recommended, particularly for etcd on the control plane nodes which require a 10 ms p99 fsync duration. Note that on many cloud platforms, storage size and IOPS scale together, so you might need to over-allocate storage volume to obtain sufficient performance.

3. As with all user-provisioned installations, if you choose to use RHEL compute machines in your cluster, you take responsibility for all operating system life cycle management and maintenance, including performing system updates, applying patches, and completing all other required tasks. Use of RHEL 7 compute machines is deprecated and has been removed in OpenShift Container Platform 4.10 and later.

NOTE

For OpenShift Container Platform version 4.19, RHCOS is based on RHEL version 9.6, which updates the micro-architecture requirements. The following list contains the minimum instruction set architectures (ISA) that each architecture requires:

- x86-64 architecture requires x86-64-v2 ISA
- ARM64 architecture requires ARMv8.0-A ISA
- IBM Power architecture requires Power 9 ISA
- s390x architecture requires z14 ISA

For more information, see Architectures (RHEL documentation).

If an instance type for your platform meets the minimum requirements for cluster machines, it is supported to use in OpenShift Container Platform.

### Additional resources

- Optimizing storage

### 3.5.4.2. Sample customized install-config.yaml file for AWS

You can customize the installation configuration file (**install-config.yaml**) to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.

IMPORTANT

This sample YAML file is provided for reference only. You must obtain your **install-config.yaml** file by using the installation program and modify it. For a full list and description of all installation configuration parameters, see *Installation configuration parameters for AWS*.

### Sample **install-config.yaml** file for AWS

```
apiVersion: v1 1
baseDomain: example.com
sshKey: ssh-ed25519 AAAA...
pullSecret: '{"auths": ...}'
metadata:
  name: example-cluster
```

```
controlPlane: 2
  name: master
  platform:
    aws:
      type: m6i.xlarge
  replicas: 3
compute: 3
-  name: worker
  platform:
    aws:
      type: c5.4xlarge
  replicas: 3
networking: 4
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
platform: 5
  aws:
    region: us-west-2
```

1. Parameters at the first level of indentation apply to the cluster globally.
2. The **controlPlane** stanza applies to control plane machines.
3. The **compute** stanza applies to compute machines.
4. The **networking** stanza applies to the cluster networking configuration. If you do not provide networking values, the installation program provides default values.
5. The **platform** stanza applies to the infrastructure platform that hosts the cluster.

**Additional resources**

- Installation configuration parameters for AWS

### 3.5.4.3. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

**Prerequisites**

- You have an existing **install-config.yaml** file.
- You reviewed the sites that your cluster requires access to and determined whether any of them need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. You added sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.

NOTE

The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.

For installations on Amazon Web Services (AWS), Google Cloud, Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

## Procedure

1. Edit your **install-config.yaml** file and add the proxy settings. For example:

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: ec2.<aws_region>.amazonaws.com,elasticloadbalancing.
<aws_region>.amazonaws.com,s3.<aws_region>.amazonaws.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1 A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**.

2 A proxy URL to use for creating HTTPS connections outside the cluster.

3 A comma-separated list of destination domain names, IP addresses, or other network CIDRs to exclude from proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use * to bypass the proxy for all destinations. If you have added the Amazon **EC2**,**Elastic Load Balancing**, and **S3** VPC endpoints to your VPC, you must add these endpoints to the **noProxy** field.

4 If provided, the installation program generates a config map that is named **user-ca-bundle** in the **openshift-config** namespace that contains one or more additional CA certificates that are required for proxying HTTPS connections. The Cluster Network Operator then creates a **trusted-ca-bundle** config map that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this config map is referenced in the **trustedCA** field of the **Proxy** object. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle.

5 Optional: The policy to determine the configuration of the **Proxy** object to reference the **user-ca-bundle** config map in the **trustedCA** field. The allowed values are **Proxyonly** and **Always**. Use **Proxyonly** to reference the **user-ca-bundle** config map only when **http/https** proxy is configured. Use **Always** to always reference the **user-ca-bundle** config map. The default value is **Proxyonly**.

NOTE

The installation program does not support the proxy **readinessEndpoints** field.

NOTE

If the installer times out, restart and then complete the deployment by using the **wait-for** command of the installer. For example:

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. Save the file and reference it when installing OpenShift Container Platform.

The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster** **Proxy** object is still created, but it will have a nil **spec**.

NOTE

Only the **Proxy** object named **cluster** is supported, and no additional proxies can be created.

### 3.5.5. Alternatives to storing administrator-level secrets in the kube-system project

By default, administrator secrets are stored in the **kube-system** project. If you configured the **credentialsMode** parameter in the **install-config.yaml** file to **Manual**, you must use one of the following alternatives:

- To manage long-term cloud credentials manually, follow the procedure in Manually creating long-term credentials.
- To implement short-term credentials that are managed outside the cluster for individual components, follow the procedures in Configuring an AWS cluster to use short-term credentials.

#### 3.5.5.1. Manually creating long-term credentials

The Cloud Credential Operator (CCO) can be put into manual mode prior to installation in environments where the cloud identity and access management (IAM) APIs are not reachable, or the administrator prefers not to store an administrator-level credential secret in the cluster **kube-system** namespace.

**Procedure**

1. If you did not set the **credentialsMode** parameter in the **install-config.yaml** configuration file to **Manual**, modify the value as shown:

**Sample configuration file snippet**

```
apiVersion: v1
baseDomain: example.com
credentialsMode: Manual
# ...
```

–

2. If you have not previously created installation manifest files, do so by running the following command:

   ```
   $ openshift-install create manifests --dir <installation_directory>
   ```

   where **<installation_directory>** is the directory in which the installation program creates files.

3. Set a **$RELEASE_IMAGE** variable with the release image from your installation file by running the following command:

   ```
   $ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
   ```

4. Extract the list of **CredentialsRequest** custom resources (CRs) from the OpenShift Container Platform release image by running the following command:

   ```
   $ oc adm release extract \
     --from=$RELEASE_IMAGE \
     --credentials-requests \
     --included \ 1
     --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
     --to=<path_to_directory_for_credentials_requests> 3
   ```

   1 The **--included** parameter includes only the manifests that your specific cluster configuration requires.

   2 Specify the location of the **install-config.yaml** file.

   3 Specify the path to the directory where you want to store the **CredentialsRequest** objects. If the specified directory does not exist, this command creates it.

   This command creates a YAML file for each **CredentialsRequest** object.

   **Sample CredentialsRequest object**

   ```
   apiVersion: cloudcredential.openshift.io/v1
   kind: CredentialsRequest
   metadata:
     name: <component_credentials_request>
     namespace: openshift-cloud-credential-operator
     ...
   spec:
     providerSpec:
       apiVersion: cloudcredential.openshift.io/v1
       kind: AWSProviderSpec
       statementEntries:
       - effect: Allow
         action:
         - iam:GetUser
         - iam:GetUserPolicy
         - iam:ListAccessKeys
         resource: "*"
     ...
   ```

5. Create YAML files for secrets in the **openshift-install** manifests directory that you generated previously. The secrets must be stored using the namespace and secret name defined in the **spec.secretRef** for each **CredentialsRequest** object.

   **Sample CredentialsRequest object with secrets**

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
    - effect: Allow
      action:
      - s3:CreateBucket
      - s3:DeleteBucket
      resource: "*"
      ...
  secretRef:
    name: <component_secret>
    namespace: <component_namespace>
  ...
```

   **Sample Secret object**

```
apiVersion: v1
kind: Secret
metadata:
  name: <component_secret>
  namespace: <component_namespace>
data:
  aws_access_key_id: <base64_encoded_aws_access_key_id>
  aws_secret_access_key: <base64_encoded_aws_secret_access_key>
```

IMPORTANT

Before upgrading a cluster that uses manually maintained credentials, you must ensure that the CCO is in an upgradeable state.

### 3.5.5.2. Configuring an AWS cluster to use short-term credentials

To install a cluster that is configured to use the AWS Security Token Service (STS), you must configure the CCO utility and create the required AWS resources for your cluster.

#### 3.5.5.2.1. Configuring the Cloud Credential Operator utility

To create and manage cloud credentials from outside of the cluster when the Cloud Credential Operator (CCO) is operating in manual mode, extract and prepare the CCO utility (**ccoctl**) binary.

NOTE

The **ccoctl** utility is a Linux binary that must run in a Linux environment.

Prerequisites

- You have access to an OpenShift Container Platform account with cluster administrator access.
- You have installed the OpenShift CLI (**oc**).
- You have created an AWS account for the **ccoctl** utility to use with the following permissions:
  Required **iam** permissions
  - **iam:CreateOpenIDConnectProvider**
  - **iam:CreateRole**
  - **iam:DeleteOpenIDConnectProvider**
  - **iam:DeleteRole**
  - **iam:DeleteRolePolicy**
  - **iam:GetOpenIDConnectProvider**
  - **iam:GetRole**
  - **iam:GetUser**
  - **iam:ListOpenIDConnectProviders**
  - **iam:ListRolePolicies**
  - **iam:ListRoles**
  - **iam:PutRolePolicy**
  - **iam:TagOpenIDConnectProvider**
  - **iam:TagRole**

  Required **s3** permissions
  - **s3:CreateBucket**
  - **s3:DeleteBucket**
  - **s3:DeleteObject**
  - **s3:GetBucketAcl**
  - **s3:GetBucketTagging**
  - **s3:GetObject**
  - **s3:GetObjectAcl**

  - **s3:GetObjectTagging**
  - **s3:ListBucket**
  - **s3:PutBucketAcl**
  - **s3:PutBucketPolicy**
  - **s3:PutBucketPublicAccessBlock**
  - **s3:PutBucketTagging**
  - **s3:PutObject**
  - **s3:PutObjectAcl**
  - **s3:PutObjectTagging**

  Required **cloudfront** permissions

  - **cloudfront:ListCloudFrontOriginAccessIdentities**
  - **cloudfront:ListDistributions**
  - **cloudfront:ListTagsForResource**

- If you plan to store the OIDC configuration in a private S3 bucket that is accessed by the IAM identity provider through a public CloudFront distribution URL, the AWS account that runs the **ccoctl** utility requires the following additional permissions:
  - **cloudfront:CreateCloudFrontOriginAccessIdentity**
  - **cloudfront:CreateDistribution**
  - **cloudfront:DeleteCloudFrontOriginAccessIdentity**
  - **cloudfront:DeleteDistribution**
  - **cloudfront:GetCloudFrontOriginAccessIdentity**
  - **cloudfront:GetCloudFrontOriginAccessIdentityConfig**
  - **cloudfront:GetDistribution**
  - **cloudfront:TagResource**
  - **cloudfront:UpdateDistribution**

  NOTE

  These additional permissions support the use of the **--create-private-s3-bucket** option when processing credentials requests with the **ccoctl aws create-all** command.

Procedure

1. Set a variable for the OpenShift Container Platform release image by running the following command:

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. Obtain the CCO container image from the OpenShift Container Platform release image by running the following command:

```
$ CCO_IMAGE=$(oc adm release info --image-for='cloud-credential-operator'
$RELEASE_IMAGE -a ~/.pull-secret)
```

NOTE

Ensure that the architecture of the **$RELEASE_IMAGE** matches the architecture of the environment in which you will use the **ccoctl** tool.

3. Extract the **ccoctl** binary from the CCO container image within the OpenShift Container Platform release image by running the following command:

```
$ oc image extract $CCO_IMAGE \
  --file="/usr/bin/ccoctl.<rhel_version>" \ 1
  -a ~/.pull-secret
```

1 For **<rhel_version>**, specify the value that corresponds to the version of Red Hat Enterprise Linux (RHEL) that the host uses. If no value is specified, **ccoctl.rhel8** is used by default. The following values are valid:

- **rhel8**: Specify this value for hosts that use RHEL 8.
- **rhel9**: Specify this value for hosts that use RHEL 9.

NOTE

The **ccoctl** binary is created in the directory from where you executed the command and not in **/usr/bin/**. You must rename the directory or move the **ccoctl.<rhel_version>** binary to **ccoctl**.

4. Change the permissions to make **ccoctl** executable by running the following command:

```
$ chmod 775 ccoctl
```

## Verification

- To verify that **ccoctl** is ready to use, display the help file. Use a relative file name when you run the command, for example:

```
$ ./ccoctl
```

**Example output**

```
OpenShift credentials provisioning tool

Usage:
  ccoctl [command]

Available Commands:
  aws          Manage credentials objects for AWS cloud
  azure        Manage credentials objects for Azure
  gcp          Manage credentials objects for Google cloud
  help         Help about any command
  ibmcloud     Manage credentials objects for {ibm-cloud-title}
  nutanix      Manage credentials objects for Nutanix

Flags:
  -h, --help   help for ccoctl

Use "ccoctl [command] --help" for more information about a command.
```

#### 3.5.5.2.2. Creating AWS resources with the Cloud Credential Operator utility

You have the following options when creating AWS resources:

- You can use the **ccoctl aws create-all** command to create the AWS resources automatically. This is the quickest way to create the resources. See Creating AWS resources with a single command.
- If you need to review the JSON files that the **ccoctl** tool creates before modifying AWS resources, or if the process the **ccoctl** tool uses to create AWS resources automatically does not meet the requirements of your organization, you can create the AWS resources individually. See Creating AWS resources individually.

##### 3.5.5.2.2.1. Creating AWS resources with a single command

If the process the **ccoctl** tool uses to create AWS resources automatically meets the requirements of your organization, you can use the **ccoctl aws create-all** command to automate the creation of AWS resources.

Otherwise, you can create the AWS resources individually. For more information, see "Creating AWS resources individually".

NOTE

By default, **ccoctl** creates objects in the directory in which the commands are run. To create the objects in a different directory, use the **--output-dir** flag. This procedure uses **<path_to_ccoctl_output_dir>** to refer to this directory.

### Prerequisites

You must have:

- Extracted and prepared the **ccoctl** binary.

### Procedure

1. Set a **$RELEASE_IMAGE** variable with the release image from your installation file by running the following command:

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. Extract the list of **CredentialsRequest** objects from the OpenShift Container Platform release image by running the following command:

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

1 The **--included** parameter includes only the manifests that your specific cluster configuration requires.

2 Specify the location of the **install-config.yaml** file.

3 Specify the path to the directory where you want to store the **CredentialsRequest** objects. If the specified directory does not exist, this command creates it.

NOTE

This command might take a few moments to run.

3. Use the **ccoctl** tool to process all **CredentialsRequest** objects by running the following command:

```
$ ccoctl aws create-all \
  --name=<name> \ 1
  --region=<aws_region> \ 2
  --credentials-requests-dir=<path_to_credentials_requests_directory> \ 3
  --output-dir=<path_to_ccoctl_output_dir> \ 4
  --create-private-s3-bucket 5
```

1 Specify the name used to tag any cloud resources that are created for tracking.

2 Specify the AWS region in which cloud resources will be created.

3 Specify the directory containing the files for the component **CredentialsRequest** objects.

4 Optional: Specify the directory in which you want the **ccoctl** utility to create objects. By default, the utility creates objects in the directory in which the commands are run.

5 Optional: By default, the **ccoctl** utility stores the OpenID Connect (OIDC) configuration files in a public S3 bucket and uses the S3 URL as the public OIDC endpoint. To store the OIDC configuration in a private S3 bucket that is accessed by the IAM identity provider through a public CloudFront distribution URL instead, use the **--create-private-s3-bucket** parameter.

NOTE

If your cluster uses Technology Preview features that are enabled by the **TechPreviewNoUpgrade** feature set, you must include the **--enable-tech-preview** parameter.

### Verification

- To verify that the OpenShift Container Platform secrets are created, list the files in the **<path_to_ccoctl_output_dir>/manifests** directory:

  ```
  $ ls <path_to_ccoctl_output_dir>/manifests
  ```

  **Example output**

  ```
  cluster-authentication-02-config.yaml
  openshift-cloud-credential-operator-cloud-credential-operator-iam-ro-creds-credentials.yaml
  openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
  openshift-cluster-api-capa-manager-bootstrap-credentials-credentials.yaml
  openshift-cluster-csi-drivers-ebs-cloud-credentials-credentials.yaml
  openshift-image-registry-installer-cloud-credentials-credentials.yaml
  openshift-ingress-operator-cloud-credentials-credentials.yaml
  openshift-machine-api-aws-cloud-credentials-credentials.yaml
  ```

  You can verify that the IAM roles are created by querying AWS. For more information, refer to AWS documentation on listing IAM roles.

#### 3.5.5.2.2.2. Creating AWS resources individually

You can use the **ccoctl** tool to create AWS resources individually. This option might be useful for an organization that shares the responsibility for creating these resources among different users or departments.

Otherwise, you can use the **ccoctl aws create-all** command to create the AWS resources automatically. For more information, see "Creating AWS resources with a single command".

NOTE

By default, **ccoctl** creates objects in the directory in which the commands are run. To create the objects in a different directory, use the **--output-dir** flag. This procedure uses **<path_to_ccoctl_output_dir>** to refer to this directory.

Some **ccoctl** commands make AWS API calls to create or modify AWS resources. You can use the **--dry-run** flag to avoid making API calls. Using this flag creates JSON files on the local file system instead. You can review and modify the JSON files and then apply them with the AWS CLI tool using the **--cli-input-json** parameters.

### Prerequisites

- Extract and prepare the **ccoctl** binary.

### Procedure

1. Generate the public and private RSA key files that are used to set up the OpenID Connect provider for the cluster by running the following command:

```
$ ccoctl aws create-key-pair
```

**Example output**

```
2021/04/13 11:01:02 Generating RSA keypair
2021/04/13 11:01:03 Writing private key to /<path_to_ccoctl_output_dir>/serviceaccount-signer.private
2021/04/13 11:01:03 Writing public key to /<path_to_ccoctl_output_dir>/serviceaccount-signer.public
2021/04/13 11:01:03 Copying signing key for use by installer
```

where **serviceaccount-signer.private** and **serviceaccount-signer.public** are the generated key files.

This command also creates a private key that the cluster requires during installation in **/<path_to_ccoctl_output_dir>/tls/bound-service-account-signing-key.key**.

2. Create an OpenID Connect identity provider and S3 bucket on AWS by running the following command:

```
$ ccoctl aws create-identity-provider \
  --name=<name> \ 1
  --region=<aws_region> \ 2
  --public-key-file=<path_to_ccoctl_output_dir>/serviceaccount-signer.public 3
```

1 **<name>** is the name used to tag any cloud resources that are created for tracking.

2 **<aws-region>** is the AWS region in which cloud resources will be created.

3 **<path_to_ccoctl_output_dir>** is the path to the public key file that the **ccoctl aws create-key-pair** command generated.

**Example output**

```
2021/04/13 11:16:09 Bucket <name>-oidc created
2021/04/13 11:16:10 OpenID Connect discovery document in the S3 bucket <name>-oidc at .well-known/openid-configuration updated
2021/04/13 11:16:10 Reading public key
2021/04/13 11:16:10 JSON web key set (JWKS) in the S3 bucket <name>-oidc at keys.json updated
2021/04/13 11:16:18 Identity Provider created with ARN: arn:aws:iam::<aws_account_id>:oidc-provider/<name>-oidc.s3.<aws_region>.amazonaws.com
```

where **openid-configuration** is a discovery document and **keys.json** is a JSON web key set file.

This command also creates a YAML configuration file in **/<path_to_ccoctl_output_dir>/manifests/cluster-authentication-02-config.yaml**. This file sets the issuer URL field for the service account tokens that the cluster generates, so that the AWS IAM identity provider trusts the tokens.

3. Create IAM roles for each component in the cluster:

   a. Set a **$RELEASE_IMAGE** variable with the release image from your installation file by running the following command:

   ```
   $ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
   ```

   b. Extract the list of **CredentialsRequest** objects from the OpenShift Container Platform release image:

   ```
   $ oc adm release extract \
     --from=$RELEASE_IMAGE \
     --credentials-requests \
     --included \ 1
     --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \
   2
     --to=<path_to_directory_for_credentials_requests> 3
   ```

   1 The **--included** parameter includes only the manifests that your specific cluster configuration requires.

   2 Specify the location of the **install-config.yaml** file.

   3 Specify the path to the directory where you want to store the **CredentialsRequest** objects. If the specified directory does not exist, this command creates it.

   c. Use the **ccoctl** tool to process all **CredentialsRequest** objects by running the following command:

   ```
   $ ccoctl aws create-iam-roles \
     --name=<name> \
     --region=<aws_region> \
     --credentials-requests-dir=<path_to_credentials_requests_directory> \
     --identity-provider-arn=arn:aws:iam::<aws_account_id>:oidc-provider/<name>-oidc.s3.
   <aws_region>.amazonaws.com
   ```

   NOTE

   For AWS environments that use alternative IAM API endpoints, such as GovCloud, you must also specify your region with the **--region** parameter.

   If your cluster uses Technology Preview features that are enabled by the **TechPreviewNoUpgrade** feature set, you must include the **--enable-tech-preview** parameter.

   For each **CredentialsRequest** object, **ccoctl** creates an IAM role with a trust policy that is tied to the specified OIDC identity provider, and a permissions policy as defined in each **CredentialsRequest** object from the OpenShift Container Platform release image.

**Verification**

- To verify that the OpenShift Container Platform secrets are created, list the files in the **<path_to_ccoctl_output_dir>/manifests** directory:

  ```
  $ ls <path_to_ccoctl_output_dir>/manifests
  ```

  **Example output**

  ```
  cluster-authentication-02-config.yaml
  openshift-cloud-credential-operator-cloud-credential-operator-iam-ro-creds-credentials.yaml
  openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
  openshift-cluster-api-capa-manager-bootstrap-credentials-credentials.yaml
  openshift-cluster-csi-drivers-ebs-cloud-credentials-credentials.yaml
  openshift-image-registry-installer-cloud-credentials-credentials.yaml
  openshift-ingress-operator-cloud-credentials-credentials.yaml
  openshift-machine-api-aws-cloud-credentials-credentials.yaml
  ```

  You can verify that the IAM roles are created by querying AWS. For more information, refer to AWS documentation on listing IAM roles.

#### 3.5.5.2.3. Incorporating the Cloud Credential Operator utility manifests

To implement short-term security credentials managed outside the cluster for individual components, you must move the manifest files that the Cloud Credential Operator utility (**ccoctl**) created to the correct directories for the installation program.

**Prerequisites**

- You have configured an account with the cloud platform that hosts your cluster.
- You have configured the Cloud Credential Operator utility (**ccoctl**).
- You have created the cloud provider resources that are required for your cluster with the **ccoctl** utility.

**Procedure**

1. If you did not set the **credentialsMode** parameter in the **install-config.yaml** configuration file to **Manual**, modify the value as shown:

   **Sample configuration file snippet**

   ```
   apiVersion: v1
   baseDomain: example.com
   credentialsMode: Manual
   # ...
   ```

2. If you have not previously created installation manifest files, do so by running the following command:

   ```
   $ openshift-install create manifests --dir <installation_directory>
   ```

   where **<installation_directory>** is the directory in which the installation program creates files.

3. Copy the manifests that the **ccoctl** utility generated to the **manifests** directory that the installation program created by running the following command:

```
$ cp /<path_to_ccoctl_output_dir>/manifests/* ./manifests/
```

4. Copy the **tls** directory that contains the private key to the installation directory:

```
$ cp -a /<path_to_ccoctl_output_dir>/tls .
```

### 3.5.6. Deploying the cluster

You can install OpenShift Container Platform on a compatible cloud platform.

IMPORTANT

You can run the **create cluster** command of the installation program only once, during initial installation.

#### Prerequisites

- You have configured an account with the cloud platform that hosts your cluster.
- You have the OpenShift Container Platform installation program and the pull secret for your cluster.
- You have verified that the cloud provider account on your host has the correct permissions to deploy the cluster. An account with incorrect permissions causes the installation process to fail with an error message that displays the missing permissions.

#### Procedure

1. In the directory that contains the installation program, initialize the cluster deployment by running the following command:

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 For **<installation_directory>**, specify the location of your customized **./install-config.yaml** file.

2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

2. Optional: Remove or disable the **AdministratorAccess** policy from the IAM account that you used to install the cluster.

NOTE

The elevated permissions provided by the **AdministratorAccess** policy are required only during installation.

#### Verification

When the cluster deployment completes successfully:

- The terminal displays directions for accessing your cluster, including a link to the web console and credentials for the **kubeadmin** user.
- Credential information also outputs to **<installation_directory>/.openshift_install.log**.

IMPORTANT

Do not delete the installation program or the files that the installation program creates. Both are required to delete the cluster.

### Example output

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```

IMPORTANT

- The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.
- It is recommended that you use Ignition config files within 12 hours after they are generated because the 24-hour certificate rotates from 16 to 22 hours after the cluster is installed. By using the Ignition config files within 12 hours, you can avoid installation failure if the certificate update runs during installation.

## 3.5.7. Logging in to the cluster by using the CLI

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

### Prerequisites

- You deployed an OpenShift Container Platform cluster.
- You installed the OpenShift CLI (**oc**).

### Procedure

1. Export the **kubeadmin** credentials by running the following command:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

   1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration by running the following command:

```
$ oc whoami
```

   **Example output**

```
system:admin
```

### 3.5.8. Disabling the default OperatorHub catalog sources

Operator catalogs that source content provided by Red Hat and community projects are configured for OperatorHub by default during an OpenShift Container Platform installation. In a restricted network environment, you must disable the default catalogs as a cluster administrator.

**Procedure**

- Disable the sources for the default catalogs by adding **disableAllDefaultSources: true** to the **OperatorHub** object:

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

**TIP**

Alternatively, you can use the web console to manage catalog sources. From the **Administration → Cluster Settings → Configuration → OperatorHub** page, click the **Sources** tab, where you can create, update, delete, disable, and enable individual sources.

### 3.5.9. Next steps

- Validate an installation.
- Customize your cluster.
- Configure image streams for the Cluster Samples Operator and the **must-gather** tool.
- Learn how to use Operator Lifecycle Manager in disconnected environments .
- If the mirror registry that you used to install your cluster has a trusted CA, add it to the cluster by configuring additional trust stores.
- If necessary, you can Remote health reporting .

## 3.6. INSTALLING A CLUSTER ON AWS INTO AN EXISTING VPC

In OpenShift Container Platform version 4.19, you can install a cluster into an existing Amazon Virtual Private Cloud (VPC) on Amazon Web Services (AWS). The installation program provisions the rest of the required infrastructure, which you can further customize. To customize the installation, you modify parameters in the **install-config.yaml** file before you install the cluster.

### 3.6.1. Prerequisites

- You reviewed details about the OpenShift Container Platform installation and update processes.
- You read the documentation on selecting a cluster installation method and preparing it for users.
- You configured an AWS account to host the cluster.
- If the existing VPC is owned by a different account than the cluster, you shared the VPC between accounts.

  **IMPORTANT**

  If you have an AWS profile stored on your computer, it must not use a temporary session token that you generated while using a multi-factor authentication device. The cluster continues to use your current AWS credentials to create AWS resources for the entire life of the cluster, so you must use long-term credentials. To generate appropriate keys, see Managing Access Keys for IAM Users in the AWS documentation. You can supply the keys when you run the installation program.

- If you use a firewall, you configured it to allow the sites that your cluster requires access to.

### 3.6.2. About using a custom VPC

In OpenShift Container Platform 4.19, you can deploy a cluster into existing subnets in an existing Amazon Virtual Private Cloud (VPC) in Amazon Web Services (AWS). By deploying OpenShift Container Platform into an existing AWS VPC, you might be able to avoid limit constraints in new accounts or more easily abide by the operational constraints that your company's guidelines set. If you cannot obtain the infrastructure creation permissions that are required to create the VPC yourself, use this installation option.

Because the installation program cannot know what other components are also in your existing subnets, it cannot choose subnet CIDRs and so forth on your behalf. You must configure networking for the subnets that you install your cluster to yourself.

#### 3.6.2.1. Requirements for using your VPC

The installation program no longer creates the following components:

- Internet gateways
- NAT gateways
- Subnets

- Route tables
- VPCs
- VPC DHCP options
- VPC endpoints

NOTE

The installation program requires that you use the cloud-provided DNS server. Using a custom DNS server is not supported and causes the installation to fail.

If you use a custom VPC, you must correctly configure it and its subnets for the installation program and the cluster to use. See Create a VPC in the Amazon Web Services documentation for more information about AWS VPC console wizard configurations and creating and managing an AWS VPC.

The installation program cannot:

- Subdivide network ranges for the cluster to use.
- Set route tables for the subnets.
- Set VPC options like DHCP.

You must complete these tasks before you install the cluster. See VPC networking components and Route tables for your VPC for more information on configuring networking in an AWS VPC.

Your VPC must meet the following characteristics:

- Create a public and private subnet for each availability zone that your cluster uses. Each availability zone can contain no more than one public and one private subnet. For an example of this type of configuration, see VPC with public and private subnets (NAT) in the AWS documentation.
  Record each subnet ID. Completing the installation requires that you enter these values in the **platform** section of the **install-config.yaml** file. See Finding a subnet ID in the AWS documentation.
- The VPC's CIDR block must contain the **Networking.MachineCIDR** range, which is the IP address pool for cluster machines. The subnet CIDR blocks must belong to the machine CIDR that you specify.
- The VPC must have a public internet gateway attached to it. For each availability zone:
  - The public subnet requires a route to the internet gateway.
  - The public subnet requires a NAT gateway with an EIP address.
  - The private subnet requires a route to the NAT gateway in public subnet.
- The VPC must not use the **kubernetes.io/cluster/.*: owned**, **Name**, and **openshift.io/cluster** tags.
  The installation program modifies your subnets to add the **kubernetes.io/cluster/.*: shared** tag, so your subnets must have at least one free tag slot available for it. See Tag Restrictions in the AWS documentation to confirm that the installation program can add a tag to each subnet

that you specify. You cannot use a **Name** tag, because it overlaps with the EC2 **Name** field and the installation fails.

- If you want to extend your OpenShift Container Platform cluster into an AWS Outpost and have an existing Outpost subnet, the existing subnet must use the **kubernetes.io/cluster/unmanaged: true** tag. If you do not apply this tag, the installation might fail due to the Cloud Controller Manager creating a service load balancer in the Outpost subnet, which is an unsupported configuration.

- You must enable the **enableDnsSupport** and **enableDnsHostnames** attributes in your VPC, so that the cluster can use the Route 53 zones that are attached to the VPC to resolve cluster's internal DNS records. See DNS Support in Your VPC in the AWS documentation.
  If you prefer to use your own Route 53 hosted private zone, you must associate the existing hosted zone with your VPC prior to installing a cluster. You can define your hosted zone using the **platform.aws.hostedZone** and **platform.aws.hostedZoneRole** fields in the **install-config.yaml** file. You can use a private hosted zone from another account by sharing it with the account where you install the cluster. If you use a private hosted zone from another account, you must use the **Passthrough** or **Manual** credentials mode.

If you are working in a disconnected environment, you are unable to reach the public IP addresses for EC2, ELB, and S3 endpoints. Depending on the level to which you want to restrict internet traffic during the installation, the following configuration options are available:

#### 3.6.2.1.1. Option 1: Create VPC endpoints

Create a VPC endpoint and attach it to the subnets that the clusters are using. Name the endpoints as follows:

- **ec2.<aws_region>.amazonaws.com**
- **elasticloadbalancing.<aws_region>.amazonaws.com**
- **s3.<aws_region>.amazonaws.com**

With this option, network traffic remains private between your VPC and the required AWS services.

#### 3.6.2.1.2. Option 2: Create a proxy without VPC endpoints

As part of the installation process, you can configure an HTTP or HTTPS proxy. With this option, internet traffic goes through the proxy to reach the required AWS services.

#### 3.6.2.1.3. Option 3: Create a proxy with VPC endpoints

As part of the installation process, you can configure an HTTP or HTTPS proxy with VPC endpoints. Create a VPC endpoint and attach it to the subnets that the clusters are using. Name the endpoints as follows:

- **ec2.<aws_region>.amazonaws.com**
- **elasticloadbalancing.<aws_region>.amazonaws.com**
- **s3.<aws_region>.amazonaws.com**

When configuring the proxy in the **install-config.yaml** file, add these endpoints to the **noProxy** field. With this option, the proxy prevents the cluster from accessing the internet directly. However, network traffic remains private between your VPC and the required AWS services.

### Required VPC components

You must provide a suitable VPC and subnets that allow communication to your machines.

| Component | AWS type | Description |
|---|---|---|
| VPC | • **AWS::EC2::VPC**<br>• **AWS::EC2::VPCEndpoint** | You must provide a public VPC for the cluster to use. The VPC uses an endpoint that references the route tables for each subnet to improve communication with the registry that is hosted in S3. |
| Public subnets | • **AWS::EC2::Subnet**<br>• **AWS::EC2::SubnetNetworkAclAssociation** | Your VPC must have public subnets for between 1 and 3 availability zones and associate them with appropriate Ingress rules. |
| Internet gateway | • **AWS::EC2::InternetGateway**<br>• **AWS::EC2::VPCGatewayAttachment**<br>• **AWS::EC2::RouteTable**<br>• **AWS::EC2::Route**<br>• **AWS::EC2::SubnetRouteTableAssociation**<br>• **AWS::EC2::NatGateway**<br>• **AWS::EC2::EIP** | You must have a public internet gateway, with public routes, attached to the VPC. In the provided templates, each public subnet has a NAT gateway with an EIP address. These NAT gateways allow cluster resources, like private subnet instances, to reach the internet and are not required for some restricted network or proxy scenarios. |
| Network access control | • **AWS::EC2::NetworkAcl**<br>• **AWS::EC2::NetworkAclEntry** | You must allow the VPC to access the following ports:<br><br>**Port** – **Reason**<br>**80** – Inbound HTTP traffic<br>**443** – Inbound HTTPS traffic<br>**22** – Inbound SSH traffic |

| Component | AWS type | Description | |
|---|---|---|---|
| | | **1024 - 65535** | Inbound ephemeral traffic |
| | | **0 - 65535** | Outbound ephemeral traffic |
| Private subnets | • **AWS::EC2::Subnet**<br>• **AWS::EC2::RouteTable**<br>• **AWS::EC2::SubnetRouteTableAssociation** | Your VPC can have private subnets. The provided CloudFormation templates can create private subnets for between 1 and 3 availability zones. If you use private subnets, you must provide appropriate routes and tables for them. | |

### 3.6.2.2. VPC validation

To ensure that the subnets that you provide are suitable, the installation program confirms the following data:

- All the subnets that you specify exist.
- You provide private subnets.
- The subnet CIDRs belong to the machine CIDR that you specified.
- You provide subnets for each availability zone. Each availability zone contains no more than one public and one private subnet. If you use a private cluster, provide only a private subnet for each availability zone. Otherwise, provide exactly one public and private subnet for each availability zone.
- You provide a public subnet for each private subnet availability zone. Machines are not provisioned in availability zones that you do not provide private subnets for.

If you destroy a cluster that uses an existing VPC, the VPC is not deleted. When you remove the OpenShift Container Platform cluster from a VPC, the **kubernetes.io/cluster/.*: shared** tag is removed from the subnets that it used.

### 3.6.2.3. Division of permissions

Starting with OpenShift Container Platform 4.3, you do not need all of the permissions that are required for an installation program-provisioned infrastructure cluster to deploy a cluster. This change mimics the division of permissions that you might have at your company: some individuals can create different resource in your clouds than others. For example, you might be able to create application-specific items, like instances, buckets, and load balancers, but not networking-related components such as VPCs, subnets, or ingress rules.

The AWS credentials that you use when you create your cluster do not need the networking permissions that are required to make VPCs and core networking components within the VPC, such as subnets, routing tables, internet gateways, NAT, and VPN. You still need permission to make the application resources that the machines within the cluster require, such as ELBs, security groups, S3 buckets, and nodes.

### 3.6.2.4. Isolation between clusters

If you deploy OpenShift Container Platform to an existing network, the isolation of cluster services is reduced in the following ways:

- You can install multiple OpenShift Container Platform clusters in the same VPC.
- ICMP ingress is allowed from the entire network.
- TCP 22 ingress (SSH) is allowed to the entire network.
- Control plane TCP 6443 ingress (Kubernetes API) is allowed to the entire network.
- Control plane TCP 22623 ingress (MCS) is allowed to the entire network.

### 3.6.2.5. Optional: AWS security groups

By default, the installation program creates and attaches security groups to control plane and compute machines. The rules associated with the default security groups cannot be modified.

However, you can apply additional existing AWS security groups, which are associated with your existing VPC, to control plane and compute machines. Applying custom security groups can help you meet the security needs of your organization, in such cases where you need to control the incoming or outgoing traffic of these machines.

As part of the installation process, you apply custom security groups by modifying the **install-config.yaml** file before deploying the cluster.

For more information, see "Applying existing AWS security groups to the cluster".

### 3.6.2.6. Modifying trust policy when installing into a shared VPC

If you install your cluster using a shared VPC, you can use the **Passthrough** or **Manual** credentials mode. You must add the IAM role used to install the cluster as a principal in the trust policy of the account that owns the VPC.

If you use **Passthrough** mode, add the Amazon Resource Name (ARN) of the account that creates the cluster, such as **arn:aws:iam::123456789012:user/clustercreator**, to the trust policy as a principal.

If you use **Manual** mode, add the ARN of the account that creates the cluster as well as the ARN of the ingress operator role in the cluster owner account, such as **arn:aws:iam::123456789012:role/<cluster-name>-openshift-ingress-operator-cloud-credentials**, to the trust policy as principals.

You must add the following actions to the policy:

**Example 3.5. Required actions for shared VPC installation**

- **route53:ChangeResourceRecordSets**
- **route53:ListHostedZones**
- **route53:ListHostedZonesByName**
- **route53:ListResourceRecordSets**
- **route53:ChangeTagsForResource**

- **route53:GetAccountLimit**
- **route53:GetChange**
- **route53:GetHostedZone**
- **route53:ListTagsForResource**
- **route53:UpdateHostedZoneComment**
- **tag:GetResources**
- **tag:UntagResources**

### 3.6.3. Creating the installation configuration file

You can customize the OpenShift Container Platform cluster you install on Amazon Web Services (AWS).

#### Prerequisites

- You have the OpenShift Container Platform installation program and the pull secret for your cluster.

#### Procedure

1. Create the **install-config.yaml** file.
    a. Change to the directory that contains the installation program and run the following command:

    ```
    $ ./openshift-install create install-config --dir <installation_directory>
    ```

    - **<installation_directory>**: For **<installation_directory>**, specify the directory name to store the files that the installation program creates.
      When specifying the directory:
    - Verify that the directory has the **execute** permission. This permission is required to run Terraform binaries under the installation directory.
    - Use an empty directory. Some installation assets, such as bootstrap X.509 certificates, have short expiration intervals, therefore you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

    b. At the prompts, provide the configuration details for your cloud:
        i. Optional: Select an SSH key to use to access your cluster machines.

NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

ii. Select **AWS** as the platform to target.

iii. If you do not have an Amazon Web Services (AWS) profile stored on your computer, enter the AWS access key ID and secret access key for the user that you configured to run the installation program.

iv. Select the AWS region to deploy the cluster to.

v. Select the base domain for the Route 53 service that you configured for your cluster.

vi. Enter a descriptive name for your cluster.

2. Modify the **install-config.yaml** file. You can find more information about the available parameters in the "Installation configuration parameters" section.

3. Back up the **install-config.yaml** file so that you can use it to install multiple clusters.

IMPORTANT

The **install-config.yaml** file is consumed during the installation process. If you want to reuse the file, you must back it up now.

**Additional resources**

- Installation configuration parameters for AWS

### 3.6.3.1. Minimum resource requirements for cluster installation

Each created cluster must meet minimum requirements so that the cluster runs as expected.

**Table 3.14. Minimum resource requirements**

| Machine | Operating System | vCPU [1] | Virtual RAM | Storage | Input/Output Per Second (IOPS)[2] |
|---|---|---|---|---|---|
| Bootstrap | RHCOS | 4 | 16 GB | 100 GB | 300 |
| Control plane | RHCOS | 4 | 16 GB | 100 GB | 300 |
| Compute | RHCOS, RHEL 8.6 and later [3] | 2 | 8 GB | 100 GB | 300 |

1. One vCPU is equivalent to one physical core when simultaneous multithreading (SMT), or Hyper-Threading, is not enabled. When enabled, use the following formula to calculate the corresponding ratio: (threads per core × cores) × sockets = vCPUs.

2. OpenShift Container Platform and Kubernetes are sensitive to disk performance, and faster storage is recommended, particularly for etcd on the control plane nodes which require a 10 ms p99 fsync duration. Note that on many cloud platforms, storage size and IOPS scale together, so you might need to over-allocate storage volume to obtain sufficient performance.

3. As with all user-provisioned installations, if you choose to use RHEL compute machines in your cluster, you take responsibility for all operating system life cycle management and maintenance, including performing system updates, applying patches, and completing all other required tasks. Use of RHEL 7 compute machines is deprecated and has been removed in OpenShift Container Platform 4.10 and later.

NOTE

For OpenShift Container Platform version 4.19, RHCOS is based on RHEL version 9.6, which updates the micro-architecture requirements. The following list contains the minimum instruction set architectures (ISA) that each architecture requires:

- x86-64 architecture requires x86-64-v2 ISA
- ARM64 architecture requires ARMv8.0-A ISA
- IBM Power architecture requires Power 9 ISA
- s390x architecture requires z14 ISA

For more information, see Architectures (RHEL documentation).

If an instance type for your platform meets the minimum requirements for cluster machines, it is supported to use in OpenShift Container Platform.

**Additional resources**

- Optimizing storage

### 3.6.3.2. Tested instance types for AWS

The following Amazon Web Services (AWS) instance types have been tested with OpenShift Container Platform.

NOTE

Use the machine types included in the following charts for your AWS instances. If you use an instance type that is not listed in the chart, ensure that the instance size you use matches the minimum resource requirements that are listed in the section named "Minimum resource requirements for cluster installation".

**Example 3.6. Machine types based on 64-bit x86 architecture**

- **c4.***
- **c5.***
- **c5a.***

- **i3.***
- **m4.***
- **m5.***
- **m5a.***
- **m6a.***
- **m6i.***
- **r4.***
- **r5.***
- **r5a.***
- **r6i.***
- **t3.***
- **t3a.***

### 3.6.3.3. Tested instance types for AWS on 64-bit ARM infrastructures

The following Amazon Web Services (AWS) 64-bit ARM instance types have been tested with OpenShift Container Platform.

NOTE

Use the machine types included in the following charts for your AWS ARM instances. If you use an instance type that is not listed in the chart, ensure that the instance size you use matches the minimum resource requirements that are listed in "Minimum resource requirements for cluster installation".

**Example 3.7. Machine types based on 64-bit ARM architecture**

- **c6g.***
- **c7g.***
- **m6g.***
- **m7g.***
- **r8g.***

### 3.6.3.4. Sample customized install-config.yaml file for AWS

You can customize the installation configuration file (**install-config.yaml**) to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.

IMPORTANT

This sample YAML file is provided for reference only. You must obtain your **install-config.yaml** file by using the installation program and modify it. For a full list and description of all installation configuration parameters, see *Installation configuration parameters for AWS*.

### Sample **install-config.yaml** file for AWS

```
apiVersion: v1 1
baseDomain: example.com
sshKey: ssh-ed25519 AAAA...
pullSecret: '{"auths": ...}'
metadata:
  name: example-cluster
controlPlane: 2
  name: master
  platform:
    aws:
      type: m6i.xlarge
  replicas: 3
compute: 3
-  name: worker
  platform:
    aws:
      type: c5.4xlarge
  replicas: 3
networking: 4
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
platform: 5
  aws:
    region: us-west-2
```

1. Parameters at the first level of indentation apply to the cluster globally.
2. The **controlPlane** stanza applies to control plane machines.
3. The **compute** stanza applies to compute machines.
4. The **networking** stanza applies to the cluster networking configuration. If you do not provide networking values, the installation program provides default values.
5. The **platform** stanza applies to the infrastructure platform that hosts the cluster.

#### Additional resources

- Installation configuration parameters for AWS

### 3.6.3.5. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

### Prerequisites

- You have an existing **install-config.yaml** file.
- You reviewed the sites that your cluster requires access to and determined whether any of them need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. You added sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.

NOTE

The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.

For installations on Amazon Web Services (AWS), Google Cloud, Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

### Procedure

1. Edit your **install-config.yaml** file and add the proxy settings. For example:

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: ec2.<aws_region>.amazonaws.com,elasticloadbalancing.
<aws_region>.amazonaws.com,s3.<aws_region>.amazonaws.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1 A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**.

2 A proxy URL to use for creating HTTPS connections outside the cluster.

3 A comma-separated list of destination domain names, IP addresses, or other network CIDRs to exclude from proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use * to bypass the proxy for all destinations. If you have added the Amazon **EC2**,**Elastic Load Balancing**, and **S3** VPC endpoints to your VPC, you must add these endpoints to the **noProxy** field.

4 If provided, the installation program generates a config map that is named **user-ca-bundle** in the **openshift-config** namespace that contains one or more additional CA certificates

that are required for proxying HTTPS connections. The Cluster Network Operator then creates a **trusted-ca-bundle** config map that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this config map is referenced in the **trustedCA** field of the **Proxy** object. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle.

5 Optional: The policy to determine the configuration of the **Proxy** object to reference the **user-ca-bundle** config map in the **trustedCA** field. The allowed values are **Proxyonly** and **Always**. Use **Proxyonly** to reference the **user-ca-bundle** config map only when **http/https** proxy is configured. Use **Always** to always reference the **user-ca-bundle** config map. The default value is **Proxyonly**.

NOTE

The installation program does not support the proxy **readinessEndpoints** field.

NOTE

If the installer times out, restart and then complete the deployment by using the **wait-for** command of the installer. For example:

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. Save the file and reference it when installing OpenShift Container Platform.

The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster Proxy** object is still created, but it will have a nil **spec**.

NOTE

Only the **Proxy** object named **cluster** is supported, and no additional proxies can be created.

### 3.6.3.6. Applying existing AWS security groups to the cluster

Applying existing AWS security groups to your control plane and compute machines can help you meet the security needs of your organization, in such cases where you need to control the incoming or outgoing traffic of these machines.

#### Prerequisites

- You have created the security groups in AWS. For more information, see the AWS documentation about working with security groups.
- The security groups must be associated with the existing VPC that you are deploying the cluster to. The security groups cannot be associated with another VPC.
- You have an existing **install-config.yaml** file.

#### Procedure

1. In the **install-config.yaml** file, edit the **compute.platform.aws.additionalSecurityGroupIDs** parameter to specify one or more custom security groups for your compute machines.

2. Edit the **controlPlane.platform.aws.additionalSecurityGroupIDs** parameter to specify one or more custom security groups for your control plane machines.

3. Save the file and reference it when deploying the cluster.

**Sample install-config.yaml file that specifies custom security groups**

```
# ...
compute:
- hyperthreading: Enabled
  name: worker
  platform:
    aws:
      additionalSecurityGroupIDs:
        - sg-1 (1)
        - sg-2
  replicas: 3
controlPlane:
  hyperthreading: Enabled
  name: master
  platform:
    aws:
      additionalSecurityGroupIDs:
        - sg-3
        - sg-4
  replicas: 3
platform:
  aws:
    region: us-east-1
    subnets: (2)
      - subnet-1
      - subnet-2
      - subnet-3
```

(1) Specify the name of the security group as it appears in the Amazon EC2 console, including the **sg** prefix.

(2) Specify subnets for each availability zone that your cluster uses.

### 3.6.4. Alternatives to storing administrator-level secrets in the kube-system project

By default, administrator secrets are stored in the **kube-system** project. If you configured the **credentialsMode** parameter in the **install-config.yaml** file to **Manual**, you must use one of the following alternatives:

- To manage long-term cloud credentials manually, follow the procedure in Manually creating long-term credentials.

- To implement short-term credentials that are managed outside the cluster for individual components, follow the procedures in Configuring an AWS cluster to use short-term credentials.

#### 3.6.4.1. Manually creating long-term credentials

The Cloud Credential Operator (CCO) can be put into manual mode prior to installation in environments where the cloud identity and access management (IAM) APIs are not reachable, or the administrator prefers not to store an administrator-level credential secret in the cluster **kube-system** namespace.

**Procedure**

1. If you did not set the **credentialsMode** parameter in the **install-config.yaml** configuration file to **Manual**, modify the value as shown:

   **Sample configuration file snippet**

   ```
   apiVersion: v1
   baseDomain: example.com
   credentialsMode: Manual
   # ...
   ```

2. If you have not previously created installation manifest files, do so by running the following command:

   ```
   $ openshift-install create manifests --dir <installation_directory>
   ```

   where **<installation_directory>** is the directory in which the installation program creates files.

3. Set a **$RELEASE_IMAGE** variable with the release image from your installation file by running the following command:

   ```
   $ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
   ```

4. Extract the list of **CredentialsRequest** custom resources (CRs) from the OpenShift Container Platform release image by running the following command:

   ```
   $ oc adm release extract \
     --from=$RELEASE_IMAGE \
     --credentials-requests \
     --included \ 1
     --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
     --to=<path_to_directory_for_credentials_requests> 3
   ```

   1 The **--included** parameter includes only the manifests that your specific cluster configuration requires.

   2 Specify the location of the **install-config.yaml** file.

   3 Specify the path to the directory where you want to store the **CredentialsRequest** objects. If the specified directory does not exist, this command creates it.

   This command creates a YAML file for each **CredentialsRequest** object.

   **Sample CredentialsRequest object**

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
    - effect: Allow
      action:
      - iam:GetUser
      - iam:GetUserPolicy
      - iam:ListAccessKeys
      resource: "*"
  ...
```

5. Create YAML files for secrets in the **openshift-install** manifests directory that you generated previously. The secrets must be stored using the namespace and secret name defined in the **spec.secretRef** for each **CredentialsRequest** object.

   **Sample CredentialsRequest object with secrets**

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
    - effect: Allow
      action:
      - s3:CreateBucket
      - s3:DeleteBucket
      resource: "*"
      ...
  secretRef:
    name: <component_secret>
    namespace: <component_namespace>
  ...
```

   **Sample Secret object**

```
apiVersion: v1
kind: Secret
metadata:
  name: <component_secret>
  namespace: <component_namespace>
data:
```

```
aws_access_key_id: <base64_encoded_aws_access_key_id>
aws_secret_access_key: <base64_encoded_aws_secret_access_key>
```

IMPORTANT

Before upgrading a cluster that uses manually maintained credentials, you must ensure that the CCO is in an upgradeable state.

#### 3.6.4.2. Configuring an AWS cluster to use short-term credentials

To install a cluster that is configured to use the AWS Security Token Service (STS), you must configure the CCO utility and create the required AWS resources for your cluster.

##### 3.6.4.2.1. Configuring the Cloud Credential Operator utility

To create and manage cloud credentials from outside of the cluster when the Cloud Credential Operator (CCO) is operating in manual mode, extract and prepare the CCO utility (**ccoctl**) binary.

NOTE

The **ccoctl** utility is a Linux binary that must run in a Linux environment.

**Prerequisites**

- You have access to an OpenShift Container Platform account with cluster administrator access.
- You have installed the OpenShift CLI (**oc**).
- You have created an AWS account for the **ccoctl** utility to use with the following permissions:
  Required **iam** permissions
  - **iam:CreateOpenIDConnectProvider**
  - **iam:CreateRole**
  - **iam:DeleteOpenIDConnectProvider**
  - **iam:DeleteRole**
  - **iam:DeleteRolePolicy**
  - **iam:GetOpenIDConnectProvider**
  - **iam:GetRole**
  - **iam:GetUser**
  - **iam:ListOpenIDConnectProviders**
  - **iam:ListRolePolicies**
  - **iam:ListRoles**
  - **iam:PutRolePolicy**

- **iam:TagOpenIDConnectProvider**
- **iam:TagRole**

Required **s3** permissions

- **s3:CreateBucket**
- **s3:DeleteBucket**
- **s3:DeleteObject**
- **s3:GetBucketAcl**
- **s3:GetBucketTagging**
- **s3:GetObject**
- **s3:GetObjectAcl**
- **s3:GetObjectTagging**
- **s3:ListBucket**
- **s3:PutBucketAcl**
- **s3:PutBucketPolicy**
- **s3:PutBucketPublicAccessBlock**
- **s3:PutBucketTagging**
- **s3:PutObject**
- **s3:PutObjectAcl**
- **s3:PutObjectTagging**

Required **cloudfront** permissions

- **cloudfront:ListCloudFrontOriginAccessIdentities**
- **cloudfront:ListDistributions**
- **cloudfront:ListTagsForResource**

- If you plan to store the OIDC configuration in a private S3 bucket that is accessed by the IAM identity provider through a public CloudFront distribution URL, the AWS account that runs the **ccoctl** utility requires the following additional permissions:
  - **cloudfront:CreateCloudFrontOriginAccessIdentity**
  - **cloudfront:CreateDistribution**
  - **cloudfront:DeleteCloudFrontOriginAccessIdentity**

- **cloudfront:DeleteDistribution**
- **cloudfront:GetCloudFrontOriginAccessIdentity**
- **cloudfront:GetCloudFrontOriginAccessIdentityConfig**
- **cloudfront:GetDistribution**
- **cloudfront:TagResource**
- **cloudfront:UpdateDistribution**

NOTE

These additional permissions support the use of the **--create-private-s3-bucket** option when processing credentials requests with the **ccoctl aws create-all** command.

### Procedure

1. Set a variable for the OpenShift Container Platform release image by running the following command:

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. Obtain the CCO container image from the OpenShift Container Platform release image by running the following command:

```
$ CCO_IMAGE=$(oc adm release info --image-for='cloud-credential-operator'
$RELEASE_IMAGE -a ~/.pull-secret)
```

NOTE

Ensure that the architecture of the **$RELEASE_IMAGE** matches the architecture of the environment in which you will use the **ccoctl** tool.

3. Extract the **ccoctl** binary from the CCO container image within the OpenShift Container Platform release image by running the following command:

```
$ oc image extract $CCO_IMAGE \
  --file="/usr/bin/ccoctl.<rhel_version>" \ 1
  -a ~/.pull-secret
```

1 For **<rhel_version>**, specify the value that corresponds to the version of Red Hat Enterprise Linux (RHEL) that the host uses. If no value is specified, **ccoctl.rhel8** is used by default. The following values are valid:

- **rhel8**: Specify this value for hosts that use RHEL 8.
- **rhel9**: Specify this value for hosts that use RHEL 9.

NOTE

The **ccoctl** binary is created in the directory from where you executed the command and not in /**usr**/**bin**/. You must rename the directory or move the **ccoctl.<rhel_version>** binary to **ccoctl**.

4. Change the permissions to make **ccoctl** executable by running the following command:

```
$ chmod 775 ccoctl
```

**Verification**

- To verify that **ccoctl** is ready to use, display the help file. Use a relative file name when you run the command, for example:

  ```
  $ ./ccoctl
  ```

  **Example output**

  ```
  OpenShift credentials provisioning tool

  Usage:
    ccoctl [command]

  Available Commands:
    aws          Manage credentials objects for AWS cloud
    azure        Manage credentials objects for Azure
    gcp          Manage credentials objects for Google cloud
    help         Help about any command
    ibmcloud     Manage credentials objects for {ibm-cloud-title}
    nutanix      Manage credentials objects for Nutanix

  Flags:
    -h, --help   help for ccoctl

  Use "ccoctl [command] --help" for more information about a command.
  ```

#### 3.6.4.2.2. Creating AWS resources with the Cloud Credential Operator utility

You have the following options when creating AWS resources:

- You can use the **ccoctl aws create-all** command to create the AWS resources automatically. This is the quickest way to create the resources. See Creating AWS resources with a single command.
- If you need to review the JSON files that the **ccoctl** tool creates before modifying AWS resources, or if the process the **ccoctl** tool uses to create AWS resources automatically does not meet the requirements of your organization, you can create the AWS resources individually. See Creating AWS resources individually .

##### 3.6.4.2.2.1. Creating AWS resources with a single command

If the process the **ccoctl** tool uses to create AWS resources automatically meets the requirements of your organization, you can use the **ccoctl aws create-all** command to automate the creation of AWS resources.

Otherwise, you can create the AWS resources individually. For more information, see "Creating AWS resources individually".

NOTE

By default, **ccoctl** creates objects in the directory in which the commands are run. To create the objects in a different directory, use the **--output-dir** flag. This procedure uses **<path_to_ccoctl_output_dir>** to refer to this directory.

## Prerequisites

You must have:

- Extracted and prepared the **ccoctl** binary.

### Procedure

1. Set a **$RELEASE_IMAGE** variable with the release image from your installation file by running the following command:

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. Extract the list of **CredentialsRequest** objects from the OpenShift Container Platform release image by running the following command:

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

1 The **--included** parameter includes only the manifests that your specific cluster configuration requires.

2 Specify the location of the **install-config.yaml** file.

3 Specify the path to the directory where you want to store the **CredentialsRequest** objects. If the specified directory does not exist, this command creates it.

NOTE

This command might take a few moments to run.

3. Use the **ccoctl** tool to process all **CredentialsRequest** objects by running the following command:

```
$ ccoctl aws create-all \
--name=<name> \ (1)
--region=<aws_region> \ (2)
--credentials-requests-dir=<path_to_credentials_requests_directory> \ (3)
--output-dir=<path_to_ccoctl_output_dir> \ (4)
--create-private-s3-bucket (5)
```

1. Specify the name used to tag any cloud resources that are created for tracking.
2. Specify the AWS region in which cloud resources will be created.
3. Specify the directory containing the files for the component **CredentialsRequest** objects.
4. Optional: Specify the directory in which you want the **ccoctl** utility to create objects. By default, the utility creates objects in the directory in which the commands are run.
5. Optional: By default, the **ccoctl** utility stores the OpenID Connect (OIDC) configuration files in a public S3 bucket and uses the S3 URL as the public OIDC endpoint. To store the OIDC configuration in a private S3 bucket that is accessed by the IAM identity provider through a public CloudFront distribution URL instead, use the **--create-private-s3-bucket** parameter.

NOTE

If your cluster uses Technology Preview features that are enabled by the **TechPreviewNoUpgrade** feature set, you must include the **--enable-tech-preview** parameter.

Verification

- To verify that the OpenShift Container Platform secrets are created, list the files in the **<path_to_ccoctl_output_dir>/manifests** directory:

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

Example output

```
cluster-authentication-02-config.yaml
openshift-cloud-credential-operator-cloud-credential-operator-iam-ro-creds-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capa-manager-bootstrap-credentials-credentials.yaml
openshift-cluster-csi-drivers-ebs-cloud-credentials-credentials.yaml
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-aws-cloud-credentials-credentials.yaml
```

You can verify that the IAM roles are created by querying AWS. For more information, refer to AWS documentation on listing IAM roles.

###### 3.6.4.2.2.2. Creating AWS resources individually

You can use the **ccoctl** tool to create AWS resources individually. This option might be useful for an organization that shares the responsibility for creating these resources among different users or departments.

Otherwise, you can use the **ccoctl aws create-all** command to create the AWS resources automatically. For more information, see "Creating AWS resources with a single command".

NOTE

By default, **ccoctl** creates objects in the directory in which the commands are run. To create the objects in a different directory, use the **--output-dir** flag. This procedure uses **<path_to_ccoctl_output_dir>** to refer to this directory.

Some **ccoctl** commands make AWS API calls to create or modify AWS resources. You can use the **--dry-run** flag to avoid making API calls. Using this flag creates JSON files on the local file system instead. You can review and modify the JSON files and then apply them with the AWS CLI tool using the **--cli-input-json** parameters.

### Prerequisites

- Extract and prepare the **ccoctl** binary.

### Procedure

1. Generate the public and private RSA key files that are used to set up the OpenID Connect provider for the cluster by running the following command:

   ```
   $ ccoctl aws create-key-pair
   ```

   **Example output**

   ```
   2021/04/13 11:01:02 Generating RSA keypair
   2021/04/13 11:01:03 Writing private key to /<path_to_ccoctl_output_dir>/serviceaccount-signer.private
   2021/04/13 11:01:03 Writing public key to /<path_to_ccoctl_output_dir>/serviceaccount-signer.public
   2021/04/13 11:01:03 Copying signing key for use by installer
   ```

   where **serviceaccount-signer.private** and **serviceaccount-signer.public** are the generated key files.

   This command also creates a private key that the cluster requires during installation in **/<path_to_ccoctl_output_dir>/tls/bound-service-account-signing-key.key**.

2. Create an OpenID Connect identity provider and S3 bucket on AWS by running the following command:

   ```
   $ ccoctl aws create-identity-provider \
     --name=<name> \ (1)
     --region=<aws_region> \ (2)
     --public-key-file=<path_to_ccoctl_output_dir>/serviceaccount-signer.public (3)
   ```

   (1) **<name>** is the name used to tag any cloud resources that are created for tracking.

2 **<aws-region>** is the AWS region in which cloud resources will be created.

3 **<path_to_ccoctl_output_dir>** is the path to the public key file that the **ccoctl aws create-key-pair** command generated.

**Example output**

```
2021/04/13 11:16:09 Bucket <name>-oidc created
2021/04/13 11:16:10 OpenID Connect discovery document in the S3 bucket <name>-oidc at
.well-known/openid-configuration updated
2021/04/13 11:16:10 Reading public key
2021/04/13 11:16:10 JSON web key set (JWKS) in the S3 bucket <name>-oidc at keys.json
updated
2021/04/13 11:16:18 Identity Provider created with ARN: arn:aws:iam::
<aws_account_id>:oidc-provider/<name>-oidc.s3.<aws_region>.amazonaws.com
```

where **openid-configuration** is a discovery document and **keys.json** is a JSON web key set file.

This command also creates a YAML configuration file in **/<path_to_ccoctl_output_dir>/manifests/cluster-authentication-02-config.yaml**. This file sets the issuer URL field for the service account tokens that the cluster generates, so that the AWS IAM identity provider trusts the tokens.

3. Create IAM roles for each component in the cluster:

   a. Set a **$RELEASE_IMAGE** variable with the release image from your installation file by running the following command:

   ```
   $ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
   ```

   b. Extract the list of **CredentialsRequest** objects from the OpenShift Container Platform release image:

   ```
   $ oc adm release extract \
     --from=$RELEASE_IMAGE \
     --credentials-requests \
     --included \ 1
     --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \
   2
     --to=<path_to_directory_for_credentials_requests> 3
   ```

   1 The **--included** parameter includes only the manifests that your specific cluster configuration requires.

   2 Specify the location of the **install-config.yaml** file.

   3 Specify the path to the directory where you want to store the **CredentialsRequest** objects. If the specified directory does not exist, this command creates it.

   c. Use the **ccoctl** tool to process all **CredentialsRequest** objects by running the following command:

```
$ ccoctl aws create-iam-roles \
  --name=<name> \
  --region=<aws_region> \
  --credentials-requests-dir=<path_to_credentials_requests_directory> \
  --identity-provider-arn=arn:aws:iam::<aws_account_id>:oidc-provider/<name>-oidc.s3.
<aws_region>.amazonaws.com
```

NOTE

For AWS environments that use alternative IAM API endpoints, such as GovCloud, you must also specify your region with the **--region** parameter.

If your cluster uses Technology Preview features that are enabled by the **TechPreviewNoUpgrade** feature set, you must include the **--enable-tech-preview** parameter.

For each **CredentialsRequest** object, **ccoctl** creates an IAM role with a trust policy that is tied to the specified OIDC identity provider, and a permissions policy as defined in each **CredentialsRequest** object from the OpenShift Container Platform release image.

#### Verification

- To verify that the OpenShift Container Platform secrets are created, list the files in the **<path_to_ccoctl_output_dir>/manifests** directory:

  ```
  $ ls <path_to_ccoctl_output_dir>/manifests
  ```

  **Example output**

  ```
  cluster-authentication-02-config.yaml
  openshift-cloud-credential-operator-cloud-credential-operator-iam-ro-creds-credentials.yaml
  openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
  openshift-cluster-api-capa-manager-bootstrap-credentials-credentials.yaml
  openshift-cluster-csi-drivers-ebs-cloud-credentials-credentials.yaml
  openshift-image-registry-installer-cloud-credentials-credentials.yaml
  openshift-ingress-operator-cloud-credentials-credentials.yaml
  openshift-machine-api-aws-cloud-credentials-credentials.yaml
  ```

  You can verify that the IAM roles are created by querying AWS. For more information, refer to AWS documentation on listing IAM roles.

### 3.6.4.2.3. Incorporating the Cloud Credential Operator utility manifests

To implement short-term security credentials managed outside the cluster for individual components, you must move the manifest files that the Cloud Credential Operator utility (**ccoctl**) created to the correct directories for the installation program.

#### Prerequisites

- You have configured an account with the cloud platform that hosts your cluster.
- You have configured the Cloud Credential Operator utility (**ccoctl**).

- You have created the cloud provider resources that are required for your cluster with the **ccoctl** utility.

#### Procedure

1. If you did not set the **credentialsMode** parameter in the **install-config.yaml** configuration file to **Manual**, modify the value as shown:

   **Sample configuration file snippet**

   ```
   apiVersion: v1
   baseDomain: example.com
   credentialsMode: Manual
   # ...
   ```

2. If you have not previously created installation manifest files, do so by running the following command:

   ```
   $ openshift-install create manifests --dir <installation_directory>
   ```

   where **<installation_directory>** is the directory in which the installation program creates files.

3. Copy the manifests that the **ccoctl** utility generated to the **manifests** directory that the installation program created by running the following command:

   ```
   $ cp /<path_to_ccoctl_output_dir>/manifests/* ./manifests/
   ```

4. Copy the **tls** directory that contains the private key to the installation directory:

   ```
   $ cp -a /<path_to_ccoctl_output_dir>/tls .
   ```

### 3.6.5. Deploying the cluster

You can install OpenShift Container Platform on a compatible cloud platform.

> **IMPORTANT**
>
> You can run the **create cluster** command of the installation program only once, during initial installation.

#### Prerequisites

- You have configured an account with the cloud platform that hosts your cluster.
- You have the OpenShift Container Platform installation program and the pull secret for your cluster.
- You have verified that the cloud provider account on your host has the correct permissions to deploy the cluster. An account with incorrect permissions causes the installation process to fail with an error message that displays the missing permissions.

#### Procedure

1. In the directory that contains the installation program, initialize the cluster deployment by running the following command:

   ```
   $ ./openshift-install create cluster --dir <installation_directory> \ 1
       --log-level=info 2
   ```

   1 For **<installation_directory>**, specify the location of your customized **./install-config.yaml** file.

   2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

2. Optional: Remove or disable the **AdministratorAccess** policy from the IAM account that you used to install the cluster.

   NOTE

   The elevated permissions provided by the **AdministratorAccess** policy are required only during installation.

### Verification

When the cluster deployment completes successfully:

- The terminal displays directions for accessing your cluster, including a link to the web console and credentials for the **kubeadmin** user.
- Credential information also outputs to **<installation_directory>/.openshift_install.log**.

IMPORTANT

Do not delete the installation program or the files that the installation program creates. Both are required to delete the cluster.

### Example output

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```

IMPORTANT

- The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

- It is recommended that you use Ignition config files within 12 hours after they are generated because the 24-hour certificate rotates from 16 to 22 hours after the cluster is installed. By using the Ignition config files within 12 hours, you can avoid installation failure if the certificate update runs during installation.

### 3.6.6. Logging in to the cluster by using the CLI

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

#### Prerequisites

- You deployed an OpenShift Container Platform cluster.

- You installed the OpenShift CLI (**oc**).

#### Procedure

1. Export the **kubeadmin** credentials by running the following command:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration by running the following command:

```
$ oc whoami
```

#### Example output

```
system:admin
```

### 3.6.7. Logging in to the cluster by using the web console

The **kubeadmin** user exists by default after an OpenShift Container Platform installation. You can log in to your cluster as the **kubeadmin** user by using the OpenShift Container Platform web console.

#### Prerequisites

- You have access to the installation host.
- You completed a cluster installation and all cluster Operators are available.

#### Procedure

1. Obtain the password for the **kubeadmin** user from the **kubeadmin-password** file on the installation host:

   ```
   $ cat <installation_directory>/auth/kubeadmin-password
   ```

   > **NOTE**
   >
   > Alternatively, you can obtain the **kubeadmin** password from the **<installation_directory>/.openshift_install.log** log file on the installation host.

2. List the OpenShift Container Platform web console route:

   ```
   $ oc get routes -n openshift-console | grep 'console-openshift'
   ```

   > **NOTE**
   >
   > Alternatively, you can obtain the OpenShift Container Platform route from the **<installation_directory>/.openshift_install.log** log file on the installation host.

   **Example output**

   ```
   console     console-openshift-console.apps.<cluster_name>.<base_domain>            console
   https   reencrypt/Redirect   None
   ```

3. Navigate to the route detailed in the output of the preceding command in a web browser and log in as the **kubeadmin** user.

#### Additional resources

- Accessing the web console

### 3.6.8. Next steps

- Validating an installation.
- Customize your cluster.
- If necessary, you can Remote health reporting.
- If necessary, you can remove cloud provider credentials.
- After installing a cluster on AWS into an existing VPC, you can extend the AWS VPC cluster into an AWS Outpost.

## 3.7. INSTALLING A PRIVATE CLUSTER ON AWS

In OpenShift Container Platform version 4.19, you can install a private cluster into an existing VPC on Amazon Web Services (AWS). The installation program provisions the rest of the required infrastructure, which you can further customize. To customize the installation, you modify parameters in the **install-config.yaml** file before you install the cluster.

### 3.7.1. Prerequisites

- You reviewed details about the OpenShift Container Platform installation and update processes.
- You read the documentation on selecting a cluster installation method and preparing it for users.
- You configured an AWS account to host the cluster.

  **IMPORTANT**

  If you have an AWS profile stored on your computer, it must not use a temporary session token that you generated while using a multi-factor authentication device. The cluster continues to use your current AWS credentials to create AWS resources for the entire life of the cluster, so you must use long-term credentials. To generate appropriate keys, see Managing Access Keys for IAM Users in the AWS documentation. You can supply the keys when you run the installation program.

- If you use a firewall, you configured it to allow the sites that your cluster requires access to.

### 3.7.2. Private clusters

You can deploy a private OpenShift Container Platform cluster that does not expose external endpoints. Private clusters are accessible from only an internal network and are not visible to the internet.

By default, OpenShift Container Platform is provisioned to use publicly-accessible DNS and endpoints. A private cluster sets the DNS, Ingress Controller, and API server to private when you deploy your cluster. This means that the cluster resources are only accessible from your internal network and are not visible to the internet.

**IMPORTANT**

If the cluster has any public subnets, load balancer services created by administrators might be publicly accessible. To ensure cluster security, verify that these services are explicitly annotated as private.

To deploy a private cluster, you must:

- Use existing networking that meets your requirements. Your cluster resources might be shared between other clusters on the network.
- Deploy from a machine that has access to:
  - The API services for the cloud to which you provision.

  - The hosts on the network that you provision.
  - The internet to obtain installation media.

You can use any machine that meets these access requirements and follows your company's guidelines. For example, this machine can be a bastion host on your cloud network or a machine that has access to the network through a VPN.

### 3.7.2.1. Private clusters in AWS

To create a private cluster on Amazon Web Services (AWS), you must provide an existing private VPC and subnets to host the cluster. The installation program must also be able to resolve the DNS records that the cluster requires. The installation program configures the Ingress Operator and API server for access from only the private network.

The cluster still requires access to internet to access the AWS APIs.

The following items are not required or created when you install a private cluster:

- Public subnets
- Public load balancers, which support public ingress
- A public Route 53 zone that matches the **baseDomain** for the cluster

The installation program does use the **baseDomain** that you specify to create a private Route 53 zone and the required records for the cluster. The cluster is configured so that the Operators do not create public records for the cluster and all cluster machines are placed in the private subnets that you specify.

#### 3.7.2.1.1. Limitations

The ability to add public functionality to a private cluster is limited.

- You cannot make the Kubernetes API endpoints public after installation without taking additional actions, including creating public subnets in the VPC for each availability zone in use, creating a public load balancer, and configuring the control plane security groups to allow traffic from the internet on 6443 (Kubernetes API port).
- If you use a public Service type load balancer, you must tag a public subnet in each availability zone with **kubernetes.io/cluster/<cluster-infra-id>: shared** so that AWS can use them to create public load balancers.

## 3.7.3. About using a custom VPC

In OpenShift Container Platform 4.19, you can deploy a cluster into existing subnets in an existing Amazon Virtual Private Cloud (VPC) in Amazon Web Services (AWS). By deploying OpenShift Container Platform into an existing AWS VPC, you might be able to avoid limit constraints in new accounts or more easily abide by the operational constraints that your company's guidelines set. If you cannot obtain the infrastructure creation permissions that are required to create the VPC yourself, use this installation option.

Because the installation program cannot know what other components are also in your existing subnets, it cannot choose subnet CIDRs and so forth on your behalf. You must configure networking for the subnets that you install your cluster to yourself.

#### 3.7.3.1. Requirements for using your VPC

The installation program no longer creates the following components:

- Internet gateways
- NAT gateways
- Subnets
- Route tables
- VPCs
- VPC DHCP options
- VPC endpoints

NOTE

The installation program requires that you use the cloud-provided DNS server. Using a custom DNS server is not supported and causes the installation to fail.

If you use a custom VPC, you must correctly configure it and its subnets for the installation program and the cluster to use. See Create a VPC in the Amazon Web Services documentation for more information about AWS VPC console wizard configurations and creating and managing an AWS VPC.

The installation program cannot:

- Subdivide network ranges for the cluster to use.
- Set route tables for the subnets.
- Set VPC options like DHCP.

You must complete these tasks before you install the cluster. See VPC networking components and Route tables for your VPC for more information on configuring networking in an AWS VPC.

Your VPC must meet the following characteristics:

- The VPC must not use the **kubernetes.io/cluster/.*: owned**, **Name**, and **openshift.io/cluster** tags.
  The installation program modifies your subnets to add the **kubernetes.io/cluster/.*: shared** tag, so your subnets must have at least one free tag slot available for it. See Tag Restrictions in the AWS documentation to confirm that the installation program can add a tag to each subnet that you specify. You cannot use a **Name** tag, because it overlaps with the EC2 **Name** field and the installation fails.

- If you want to extend your OpenShift Container Platform cluster into an AWS Outpost and have an existing Outpost subnet, the existing subnet must use the **kubernetes.io/cluster/unmanaged: true** tag. If you do not apply this tag, the installation might fail due to the Cloud Controller Manager creating a service load balancer in the Outpost subnet, which is an unsupported configuration.

- You must enable the **enableDnsSupport** and **enableDnsHostnames** attributes in your VPC, so that the cluster can use the Route 53 zones that are attached to the VPC to resolve cluster's internal DNS records. See DNS Support in Your VPC in the AWS documentation.
  If you prefer to use your own Route 53 hosted private zone, you must associate the existing hosted zone with your VPC prior to installing a cluster. You can define your hosted zone using the **platform.aws.hostedZone** and **platform.aws.hostedZoneRole** fields in the **install-config.yaml** file. You can use a private hosted zone from another account by sharing it with the account where you install the cluster. If you use a private hosted zone from another account, you must use the **Passthrough** or **Manual** credentials mode.

If you are working in a disconnected environment, you are unable to reach the public IP addresses for EC2, ELB, and S3 endpoints. Depending on the level to which you want to restrict internet traffic during the installation, the following configuration options are available:

#### 3.7.3.1.1. Option 1: Create VPC endpoints

Create a VPC endpoint and attach it to the subnets that the clusters are using. Name the endpoints as follows:

- **ec2.<aws_region>.amazonaws.com**
- **elasticloadbalancing.<aws_region>.amazonaws.com**
- **s3.<aws_region>.amazonaws.com**

With this option, network traffic remains private between your VPC and the required AWS services.

#### 3.7.3.1.2. Option 2: Create a proxy without VPC endpoints

As part of the installation process, you can configure an HTTP or HTTPS proxy. With this option, internet traffic goes through the proxy to reach the required AWS services.

#### 3.7.3.1.3. Option 3: Create a proxy with VPC endpoints

As part of the installation process, you can configure an HTTP or HTTPS proxy with VPC endpoints. Create a VPC endpoint and attach it to the subnets that the clusters are using. Name the endpoints as follows:

- **ec2.<aws_region>.amazonaws.com**
- **elasticloadbalancing.<aws_region>.amazonaws.com**
- **s3.<aws_region>.amazonaws.com**

When configuring the proxy in the **install-config.yaml** file, add these endpoints to the **noProxy** field. With this option, the proxy prevents the cluster from accessing the internet directly. However, network traffic remains private between your VPC and the required AWS services.

### Required VPC components

You must provide a suitable VPC and subnets that allow communication to your machines.

| Component | AWS type | Description |
|---|---|---|
| VPC | • **AWS::EC2::VPC**<br>• **AWS::EC2::VPCEndpoint** | You must provide a public VPC for the cluster to use. The VPC uses an endpoint that references the route tables for each subnet to improve communication with the registry that is hosted in S3. |
| Public subnets | • **AWS::EC2::Subnet**<br>• **AWS::EC2::SubnetNetworkAclAssociation** | Your VPC must have public subnets for between 1 and 3 availability zones and associate them with appropriate Ingress rules. |
| Internet gateway | • **AWS::EC2::InternetGateway**<br>• **AWS::EC2::VPCGatewayAttachment**<br>• **AWS::EC2::RouteTable**<br>• **AWS::EC2::Route**<br>• **AWS::EC2::SubnetRouteTableAssociation**<br>• **AWS::EC2::NatGateway**<br>• **AWS::EC2::EIP** | You must have a public internet gateway, with public routes, attached to the VPC. In the provided templates, each public subnet has a NAT gateway with an EIP address. These NAT gateways allow cluster resources, like private subnet instances, to reach the internet and are not required for some restricted network or proxy scenarios. |
| Network access control | • **AWS::EC2::NetworkAcl**<br>• **AWS::EC2::NetworkAclEntry** | You must allow the VPC to access the following ports: |

| Port | Reason |
|---|---|
| **80** | Inbound HTTP traffic |
| **443** | Inbound HTTPS traffic |
| **22** | Inbound SSH traffic |
| **1024 - 65535** | Inbound ephemeral traffic |
| **0 - 65535** | Outbound ephemeral traffic |

| Component | AWS type | Description |
|---|---|---|
| Private subnets | • **AWS::EC2::Subnet**<br>• **AWS::EC2::RouteTable**<br>• **AWS::EC2::SubnetRouteTableAssociation** | Your VPC can have private subnets. The provided CloudFormation templates can create private subnets for between 1 and 3 availability zones. If you use private subnets, you must provide appropriate routes and tables for them. |

#### 3.7.3.2. VPC validation

To ensure that the subnets that you provide are suitable, the installation program confirms the following data:

- All the subnets that you specify exist.
- You provide private subnets.
- The subnet CIDRs belong to the machine CIDR that you specified.
- You provide subnets for each availability zone. Each availability zone contains no more than one public and one private subnet. If you use a private cluster, provide only a private subnet for each availability zone. Otherwise, provide exactly one public and private subnet for each availability zone.
- You provide a public subnet for each private subnet availability zone. Machines are not provisioned in availability zones that you do not provide private subnets for.

If you destroy a cluster that uses an existing VPC, the VPC is not deleted. When you remove the OpenShift Container Platform cluster from a VPC, the **kubernetes.io/cluster/.*: shared** tag is removed from the subnets that it used.

#### 3.7.3.3. Division of permissions

Starting with OpenShift Container Platform 4.3, you do not need all of the permissions that are required for an installation program-provisioned infrastructure cluster to deploy a cluster. This change mimics the division of permissions that you might have at your company: some individuals can create different resource in your clouds than others. For example, you might be able to create application-specific items, like instances, buckets, and load balancers, but not networking-related components such as VPCs, subnets, or ingress rules.

The AWS credentials that you use when you create your cluster do not need the networking permissions that are required to make VPCs and core networking components within the VPC, such as subnets, routing tables, internet gateways, NAT, and VPN. You still need permission to make the application resources that the machines within the cluster require, such as ELBs, security groups, S3 buckets, and nodes.

#### 3.7.3.4. Isolation between clusters

If you deploy OpenShift Container Platform to an existing network, the isolation of cluster services is reduced in the following ways:

- You can install multiple OpenShift Container Platform clusters in the same VPC.
- ICMP ingress is allowed from the entire network.
- TCP 22 ingress (SSH) is allowed to the entire network.
- Control plane TCP 6443 ingress (Kubernetes API) is allowed to the entire network.
- Control plane TCP 22623 ingress (MCS) is allowed to the entire network.

#### 3.7.3.5. Optional: AWS security groups

By default, the installation program creates and attaches security groups to control plane and compute machines. The rules associated with the default security groups cannot be modified.

However, you can apply additional existing AWS security groups, which are associated with your existing VPC, to control plane and compute machines. Applying custom security groups can help you meet the security needs of your organization, in such cases where you need to control the incoming or outgoing traffic of these machines.

As part of the installation process, you apply custom security groups by modifying the **install-config.yaml** file before deploying the cluster.

For more information, see "Applying existing AWS security groups to the cluster".

### 3.7.4. Manually creating the installation configuration file

To customise your OpenShift Container Platform deployment and meet specific network requirements, manually create the installation configuration file. This ensures that the installation program uses your tailored settings rather than default values during the setup process.

**Prerequisites**

- You have an SSH public key on your local machine for use with the installation program. You can use the key for SSH authentication onto your cluster nodes for debugging and disaster recovery.
- You have obtained the OpenShift Container Platform installation program and the pull secret for your cluster.

**Procedure**

1. Create an installation directory to store your required installation assets in:

```
$ mkdir <installation_directory>
```

IMPORTANT

You must create a directory. Some installation assets, such as bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

2. Customize the provided sample **install-config.yaml** file template and save the file in the **<installation_directory>**.

   NOTE

   You must name this configuration file **install-config.yaml**.

3. Back up the **install-config.yaml** file so that you can use it to install many clusters.

   IMPORTANT

   Back up the **install-config.yaml** file now, because the installation process consumes the file in the next step.

**Additional resources**

- Installation configuration parameters for AWS

### 3.7.4.1. Minimum resource requirements for cluster installation

Each created cluster must meet minimum requirements so that the cluster runs as expected.

**Table 3.15. Minimum resource requirements**

| Machine | Operating System | vCPU [1] | Virtual RAM | Storage | Input/Output Per Second (IOPS)[2] |
|---|---|---|---|---|---|
| Bootstrap | RHCOS | 4 | 16 GB | 100 GB | 300 |
| Control plane | RHCOS | 4 | 16 GB | 100 GB | 300 |
| Compute | RHCOS, RHEL 8.6 and later [3] | 2 | 8 GB | 100 GB | 300 |

1. One vCPU is equivalent to one physical core when simultaneous multithreading (SMT), or Hyper-Threading, is not enabled. When enabled, use the following formula to calculate the corresponding ratio: (threads per core × cores) × sockets = vCPUs.

2. OpenShift Container Platform and Kubernetes are sensitive to disk performance, and faster storage is recommended, particularly for etcd on the control plane nodes which require a 10 ms p99 fsync duration. Note that on many cloud platforms, storage size and IOPS scale together, so you might need to over-allocate storage volume to obtain sufficient performance.

3. As with all user-provisioned installations, if you choose to use RHEL compute machines in your cluster, you take responsibility for all operating system life cycle management and maintenance, including performing system updates, applying patches, and completing all other required tasks. Use of RHEL 7 compute machines is deprecated and has been removed in OpenShift Container Platform 4.10 and later.

NOTE

For OpenShift Container Platform version 4.19, RHCOS is based on RHEL version 9.6, which updates the micro-architecture requirements. The following list contains the minimum instruction set architectures (ISA) that each architecture requires:

- x86-64 architecture requires x86-64-v2 ISA
- ARM64 architecture requires ARMv8.0-A ISA
- IBM Power architecture requires Power 9 ISA
- s390x architecture requires z14 ISA

For more information, see Architectures (RHEL documentation).

If an instance type for your platform meets the minimum requirements for cluster machines, it is supported to use in OpenShift Container Platform.

**Additional resources**

- Optimizing storage

### 3.7.4.2. Tested instance types for AWS

The following Amazon Web Services (AWS) instance types have been tested with OpenShift Container Platform.

NOTE

Use the machine types included in the following charts for your AWS instances. If you use an instance type that is not listed in the chart, ensure that the instance size you use matches the minimum resource requirements that are listed in the section named "Minimum resource requirements for cluster installation".

**Example 3.8. Machine types based on 64-bit x86 architecture**

- **c4.***
- **c5.***
- **c5a.***
- **i3.***
- **m4.***
- **m5.***
- **m5a.***
- **m6a.***
- **m6i.***

- **r4.***
- **r5.***
- **r5a.***
- **r6i.***
- **t3.***
- **t3a.***

### 3.7.4.3. Tested instance types for AWS on 64-bit ARM infrastructures

The following Amazon Web Services (AWS) 64-bit ARM instance types have been tested with OpenShift Container Platform.

NOTE

Use the machine types included in the following charts for your AWS ARM instances. If you use an instance type that is not listed in the chart, ensure that the instance size you use matches the minimum resource requirements that are listed in "Minimum resource requirements for cluster installation".

**Example 3.9. Machine types based on 64-bit ARM architecture**

- **c6g.***
- **c7g.***
- **m6g.***
- **m7g.***
- **r8g.***

### 3.7.4.4. Sample customized install-config.yaml file for AWS

You can customize the installation configuration file (**install-config.yaml**) to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.

IMPORTANT

This sample YAML file is provided for reference only. You must obtain your **install-config.yaml** file by using the installation program and modify it. For a full list and description of all installation configuration parameters, see *Installation configuration parameters for AWS*.

**Sample install-config.yaml file for AWS**

```
apiVersion: v1 (1)
baseDomain: example.com
sshKey: ssh-ed25519 AAAA...
pullSecret: '{"auths": ...}'
metadata:
  name: example-cluster
controlPlane: (2)
  name: master
  platform:
    aws:
      type: m6i.xlarge
  replicas: 3
compute: (3)
- name: worker
  platform:
    aws:
      type: c5.4xlarge
  replicas: 3
networking: (4)
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
platform: (5)
  aws:
    region: us-west-2
```

1. Parameters at the first level of indentation apply to the cluster globally.
2. The **controlPlane** stanza applies to control plane machines.
3. The **compute** stanza applies to compute machines.
4. The **networking** stanza applies to the cluster networking configuration. If you do not provide networking values, the installation program provides default values.
5. The **platform** stanza applies to the infrastructure platform that hosts the cluster.

**Additional resources**

- Installation configuration parameters for AWS

### 3.7.4.5. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

**Prerequisites**

- You have an existing **install-config.yaml** file.
- You reviewed the sites that your cluster requires access to and determined whether any of them need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to

hosting cloud provider APIs. You added sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.

NOTE

The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.

For installations on Amazon Web Services (AWS), Google Cloud, Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

### Procedure

1. Edit your **install-config.yaml** file and add the proxy settings. For example:

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: ec2.<aws_region>.amazonaws.com,elasticloadbalancing.
<aws_region>.amazonaws.com,s3.<aws_region>.amazonaws.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1 A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**.

2 A proxy URL to use for creating HTTPS connections outside the cluster.

3 A comma-separated list of destination domain names, IP addresses, or other network CIDRs to exclude from proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use * to bypass the proxy for all destinations. If you have added the Amazon **EC2**,**Elastic Load Balancing**, and **S3** VPC endpoints to your VPC, you must add these endpoints to the **noProxy** field.

4 If provided, the installation program generates a config map that is named **user-ca-bundle** in the **openshift-config** namespace that contains one or more additional CA certificates that are required for proxying HTTPS connections. The Cluster Network Operator then creates a **trusted-ca-bundle** config map that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this config map is referenced in the **trustedCA** field of the **Proxy** object. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle.

5 Optional: The policy to determine the configuration of the **Proxy** object to reference the **user-ca-bundle** config map in the **trustedCA** field. The allowed values are **Proxyonly** and **Always**. Use **Proxyonly** to reference the **user-ca-bundle** config map only when

**http/https** proxy is configured. Use **Always** to always reference the **user-ca-bundle** config map. The default value is **Proxyonly**.

NOTE

The installation program does not support the proxy **readinessEndpoints** field.

NOTE

If the installer times out, restart and then complete the deployment by using the **wait-for** command of the installer. For example:

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. Save the file and reference it when installing OpenShift Container Platform.

The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster Proxy** object is still created, but it will have a nil **spec**.

NOTE

Only the **Proxy** object named **cluster** is supported, and no additional proxies can be created.

### 3.7.4.6. Applying existing AWS security groups to the cluster

Applying existing AWS security groups to your control plane and compute machines can help you meet the security needs of your organization, in such cases where you need to control the incoming or outgoing traffic of these machines.

#### Prerequisites

- You have created the security groups in AWS. For more information, see the AWS documentation about working with security groups.
- The security groups must be associated with the existing VPC that you are deploying the cluster to. The security groups cannot be associated with another VPC.
- You have an existing **install-config.yaml** file.

#### Procedure

1. In the **install-config.yaml** file, edit the **compute.platform.aws.additionalSecurityGroupIDs** parameter to specify one or more custom security groups for your compute machines.
2. Edit the **controlPlane.platform.aws.additionalSecurityGroupIDs** parameter to specify one or more custom security groups for your control plane machines.
3. Save the file and reference it when deploying the cluster.

#### Sample install-config.yaml file that specifies custom security groups

```
# ...
compute:
- hyperthreading: Enabled
  name: worker
  platform:
    aws:
      additionalSecurityGroupIDs:
        - sg-1 (1)
        - sg-2
  replicas: 3
controlPlane:
  hyperthreading: Enabled
  name: master
  platform:
    aws:
      additionalSecurityGroupIDs:
        - sg-3
        - sg-4
  replicas: 3
platform:
  aws:
    region: us-east-1
    subnets: (2)
      - subnet-1
      - subnet-2
      - subnet-3
```

1. Specify the name of the security group as it appears in the Amazon EC2 console, including the **sg** prefix.

2. Specify subnets for each availability zone that your cluster uses.

### 3.7.5. Alternatives to storing administrator-level secrets in the kube-system project

By default, administrator secrets are stored in the **kube-system** project. If you configured the **credentialsMode** parameter in the **install-config.yaml** file to **Manual**, you must use one of the following alternatives:

- To manage long-term cloud credentials manually, follow the procedure in Manually creating long-term credentials.

- To implement short-term credentials that are managed outside the cluster for individual components, follow the procedures in Configuring an AWS cluster to use short-term credentials.

#### 3.7.5.1. Manually creating long-term credentials

The Cloud Credential Operator (CCO) can be put into manual mode prior to installation in environments where the cloud identity and access management (IAM) APIs are not reachable, or the administrator prefers not to store an administrator-level credential secret in the cluster **kube-system** namespace.

**Procedure**

1. If you did not set the **credentialsMode** parameter in the **install-config.yaml** configuration file to **Manual**, modify the value as shown:

   **Sample configuration file snippet**

   ```
   apiVersion: v1
   baseDomain: example.com
   credentialsMode: Manual
   # ...
   ```

2. If you have not previously created installation manifest files, do so by running the following command:

   ```
   $ openshift-install create manifests --dir <installation_directory>
   ```

   where **<installation_directory>** is the directory in which the installation program creates files.

3. Set a **$RELEASE_IMAGE** variable with the release image from your installation file by running the following command:

   ```
   $ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
   ```

4. Extract the list of **CredentialsRequest** custom resources (CRs) from the OpenShift Container Platform release image by running the following command:

   ```
   $ oc adm release extract \
     --from=$RELEASE_IMAGE \
     --credentials-requests \
     --included \ 1
     --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
     --to=<path_to_directory_for_credentials_requests> 3
   ```

   1 The **--included** parameter includes only the manifests that your specific cluster configuration requires.

   2 Specify the location of the **install-config.yaml** file.

   3 Specify the path to the directory where you want to store the **CredentialsRequest** objects. If the specified directory does not exist, this command creates it.

   This command creates a YAML file for each **CredentialsRequest** object.

   **Sample CredentialsRequest object**

   ```
   apiVersion: cloudcredential.openshift.io/v1
   kind: CredentialsRequest
   metadata:
     name: <component_credentials_request>
     namespace: openshift-cloud-credential-operator
     ...
   spec:
     providerSpec:
       apiVersion: cloudcredential.openshift.io/v1
   ```

```
    kind: AWSProviderSpec
    statementEntries:
    - effect: Allow
      action:
      - iam:GetUser
      - iam:GetUserPolicy
      - iam:ListAccessKeys
      resource: "*"
  ...
```

5. Create YAML files for secrets in the **openshift-install** manifests directory that you generated previously. The secrets must be stored using the namespace and secret name defined in the **spec.secretRef** for each **CredentialsRequest** object.

   **Sample CredentialsRequest object with secrets**

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
    - effect: Allow
      action:
      - s3:CreateBucket
      - s3:DeleteBucket
      resource: "*"
      ...
  secretRef:
    name: <component_secret>
    namespace: <component_namespace>
  ...
```

   **Sample Secret object**

```
apiVersion: v1
kind: Secret
metadata:
  name: <component_secret>
  namespace: <component_namespace>
data:
  aws_access_key_id: <base64_encoded_aws_access_key_id>
  aws_secret_access_key: <base64_encoded_aws_secret_access_key>
```

IMPORTANT

Before upgrading a cluster that uses manually maintained credentials, you must ensure that the CCO is in an upgradeable state.

### 3.7.5.2. Configuring an AWS cluster to use short-term credentials

To install a cluster that is configured to use the AWS Security Token Service (STS), you must configure the CCO utility and create the required AWS resources for your cluster.

#### 3.7.5.2.1. Configuring the Cloud Credential Operator utility

To create and manage cloud credentials from outside of the cluster when the Cloud Credential Operator (CCO) is operating in manual mode, extract and prepare the CCO utility (**ccoctl**) binary.

NOTE

The **ccoctl** utility is a Linux binary that must run in a Linux environment.

**Prerequisites**

- You have access to an OpenShift Container Platform account with cluster administrator access.
- You have installed the OpenShift CLI (**oc**).
- You have created an AWS account for the **ccoctl** utility to use with the following permissions:
  **Required iam permissions**
  - **iam:CreateOpenIDConnectProvider**
  - **iam:CreateRole**
  - **iam:DeleteOpenIDConnectProvider**
  - **iam:DeleteRole**
  - **iam:DeleteRolePolicy**
  - **iam:GetOpenIDConnectProvider**
  - **iam:GetRole**
  - **iam:GetUser**
  - **iam:ListOpenIDConnectProviders**
  - **iam:ListRolePolicies**
  - **iam:ListRoles**
  - **iam:PutRolePolicy**
  - **iam:TagOpenIDConnectProvider**
  - **iam:TagRole**

  **Required s3 permissions**
  - **s3:CreateBucket**

- **s3:DeleteBucket**
- **s3:DeleteObject**
- **s3:GetBucketAcl**
- **s3:GetBucketTagging**
- **s3:GetObject**
- **s3:GetObjectAcl**
- **s3:GetObjectTagging**
- **s3:ListBucket**
- **s3:PutBucketAcl**
- **s3:PutBucketPolicy**
- **s3:PutBucketPublicAccessBlock**
- **s3:PutBucketTagging**
- **s3:PutObject**
- **s3:PutObjectAcl**
- **s3:PutObjectTagging**

Required **cloudfront** permissions

- **cloudfront:ListCloudFrontOriginAccessIdentities**
- **cloudfront:ListDistributions**
- **cloudfront:ListTagsForResource**

- If you plan to store the OIDC configuration in a private S3 bucket that is accessed by the IAM identity provider through a public CloudFront distribution URL, the AWS account that runs the **ccoctl** utility requires the following additional permissions:
  - **cloudfront:CreateCloudFrontOriginAccessIdentity**
  - **cloudfront:CreateDistribution**
  - **cloudfront:DeleteCloudFrontOriginAccessIdentity**
  - **cloudfront:DeleteDistribution**
  - **cloudfront:GetCloudFrontOriginAccessIdentity**
  - **cloudfront:GetCloudFrontOriginAccessIdentityConfig**
  - **cloudfront:GetDistribution**

- **cloudfront:TagResource**
- **cloudfront:UpdateDistribution**

NOTE

These additional permissions support the use of the **--create-private-s3-bucket** option when processing credentials requests with the **ccoctl aws create-all** command.

Procedure

1. Set a variable for the OpenShift Container Platform release image by running the following command:

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. Obtain the CCO container image from the OpenShift Container Platform release image by running the following command:

```
$ CCO_IMAGE=$(oc adm release info --image-for='cloud-credential-operator'
$RELEASE_IMAGE -a ~/.pull-secret)
```

NOTE

Ensure that the architecture of the **$RELEASE_IMAGE** matches the architecture of the environment in which you will use the **ccoctl** tool.

3. Extract the **ccoctl** binary from the CCO container image within the OpenShift Container Platform release image by running the following command:

```
$ oc image extract $CCO_IMAGE \
  --file="/usr/bin/ccoctl.<rhel_version>" \ 1
  -a ~/.pull-secret
```

1 For **<rhel_version>**, specify the value that corresponds to the version of Red Hat Enterprise Linux (RHEL) that the host uses. If no value is specified, **ccoctl.rhel8** is used by default. The following values are valid:

- **rhel8**: Specify this value for hosts that use RHEL 8.
- **rhel9**: Specify this value for hosts that use RHEL 9.

NOTE

The **ccoctl** binary is created in the directory from where you executed the command and not in **/usr/bin/**. You must rename the directory or move the **ccoctl.<rhel_version>** binary to **ccoctl**.

4. Change the permissions to make **ccoctl** executable by running the following command:

```
$ chmod 775 ccoctl
```

**Verification**

- To verify that **ccoctl** is ready to use, display the help file. Use a relative file name when you run the command, for example:

  ```
  $ ./ccoctl
  ```

  **Example output**

  ```
  OpenShift credentials provisioning tool

  Usage:
    ccoctl [command]

  Available Commands:
    aws          Manage credentials objects for AWS cloud
    azure        Manage credentials objects for Azure
    gcp          Manage credentials objects for Google cloud
    help         Help about any command
    ibmcloud     Manage credentials objects for {ibm-cloud-title}
    nutanix      Manage credentials objects for Nutanix

  Flags:
    -h, --help   help for ccoctl

  Use "ccoctl [command] --help" for more information about a command.
  ```

#### 3.7.5.2.2. Creating AWS resources with the Cloud Credential Operator utility

You have the following options when creating AWS resources:

- You can use the **ccoctl aws create-all** command to create the AWS resources automatically. This is the quickest way to create the resources. See Creating AWS resources with a single command.

- If you need to review the JSON files that the **ccoctl** tool creates before modifying AWS resources, or if the process the **ccoctl** tool uses to create AWS resources automatically does not meet the requirements of your organization, you can create the AWS resources individually. See Creating AWS resources individually .

##### 3.7.5.2.2.1. Creating AWS resources with a single command

If the process the **ccoctl** tool uses to create AWS resources automatically meets the requirements of your organization, you can use the **ccoctl aws create-all** command to automate the creation of AWS resources.

Otherwise, you can create the AWS resources individually. For more information, see "Creating AWS resources individually".

NOTE

By default, **ccoctl** creates objects in the directory in which the commands are run. To create the objects in a different directory, use the **--output-dir** flag. This procedure uses **<path_to_ccoctl_output_dir>** to refer to this directory.

### Prerequisites

You must have:

- Extracted and prepared the **ccoctl** binary.

#### Procedure

1. Set a **$RELEASE_IMAGE** variable with the release image from your installation file by running the following command:

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. Extract the list of **CredentialsRequest** objects from the OpenShift Container Platform release image by running the following command:

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

1 The **--included** parameter includes only the manifests that your specific cluster configuration requires.

2 Specify the location of the **install-config.yaml** file.

3 Specify the path to the directory where you want to store the **CredentialsRequest** objects. If the specified directory does not exist, this command creates it.

NOTE

This command might take a few moments to run.

3. Use the **ccoctl** tool to process all **CredentialsRequest** objects by running the following command:

```
$ ccoctl aws create-all \
  --name=<name> \ 1
  --region=<aws_region> \ 2
  --credentials-requests-dir=<path_to_credentials_requests_directory> \ 3
  --output-dir=<path_to_ccoctl_output_dir> \ 4
  --create-private-s3-bucket 5
```

1. Specify the name used to tag any cloud resources that are created for tracking.
2. Specify the AWS region in which cloud resources will be created.
3. Specify the directory containing the files for the component **CredentialsRequest** objects.
4. Optional: Specify the directory in which you want the **ccoctl** utility to create objects. By default, the utility creates objects in the directory in which the commands are run.
5. Optional: By default, the **ccoctl** utility stores the OpenID Connect (OIDC) configuration files in a public S3 bucket and uses the S3 URL as the public OIDC endpoint. To store the OIDC configuration in a private S3 bucket that is accessed by the IAM identity provider through a public CloudFront distribution URL instead, use the **--create-private-s3-bucket** parameter.

NOTE

If your cluster uses Technology Preview features that are enabled by the **TechPreviewNoUpgrade** feature set, you must include the **--enable-tech-preview** parameter.

### Verification

- To verify that the OpenShift Container Platform secrets are created, list the files in the **<path_to_ccoctl_output_dir>/manifests** directory:

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

#### Example output

```
cluster-authentication-02-config.yaml
openshift-cloud-credential-operator-cloud-credential-operator-iam-ro-creds-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capa-manager-bootstrap-credentials-credentials.yaml
openshift-cluster-csi-drivers-ebs-cloud-credentials-credentials.yaml
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-aws-cloud-credentials-credentials.yaml
```

You can verify that the IAM roles are created by querying AWS. For more information, refer to AWS documentation on listing IAM roles.

#### 3.7.5.2.2.2. Creating AWS resources individually

You can use the **ccoctl** tool to create AWS resources individually. This option might be useful for an organization that shares the responsibility for creating these resources among different users or departments.

Otherwise, you can use the **ccoctl aws create-all** command to create the AWS resources automatically. For more information, see "Creating AWS resources with a single command".

NOTE

By default, **ccoctl** creates objects in the directory in which the commands are run. To create the objects in a different directory, use the **--output-dir** flag. This procedure uses **<path_to_ccoctl_output_dir>** to refer to this directory.

Some **ccoctl** commands make AWS API calls to create or modify AWS resources. You can use the **--dry-run** flag to avoid making API calls. Using this flag creates JSON files on the local file system instead. You can review and modify the JSON files and then apply them with the AWS CLI tool using the **--cli-input-json** parameters.

### Prerequisites

- Extract and prepare the **ccoctl** binary.

### Procedure

1. Generate the public and private RSA key files that are used to set up the OpenID Connect provider for the cluster by running the following command:

```
$ ccoctl aws create-key-pair
```

#### Example output

```
2021/04/13 11:01:02 Generating RSA keypair
2021/04/13 11:01:03 Writing private key to /<path_to_ccoctl_output_dir>/serviceaccount-signer.private
2021/04/13 11:01:03 Writing public key to /<path_to_ccoctl_output_dir>/serviceaccount-signer.public
2021/04/13 11:01:03 Copying signing key for use by installer
```

where **serviceaccount-signer.private** and **serviceaccount-signer.public** are the generated key files.

This command also creates a private key that the cluster requires during installation in **/<path_to_ccoctl_output_dir>/tls/bound-service-account-signing-key.key**.

2. Create an OpenID Connect identity provider and S3 bucket on AWS by running the following command:

```
$ ccoctl aws create-identity-provider \
  --name=<name> \ 1
  --region=<aws_region> \ 2
  --public-key-file=<path_to_ccoctl_output_dir>/serviceaccount-signer.public 3
```

1 **<name>** is the name used to tag any cloud resources that are created for tracking.

2 **<aws-region>** is the AWS region in which cloud resources will be created.

3 **<path_to_ccoctl_output_dir>** is the path to the public key file that the **ccoctl aws create-key-pair** command generated.

#### Example output

```
2021/04/13 11:16:09 Bucket <name>-oidc created
2021/04/13 11:16:10 OpenID Connect discovery document in the S3 bucket <name>-oidc at
.well-known/openid-configuration updated
2021/04/13 11:16:10 Reading public key
2021/04/13 11:16:10 JSON web key set (JWKS) in the S3 bucket <name>-oidc at keys.json
updated
2021/04/13 11:16:18 Identity Provider created with ARN: arn:aws:iam::
<aws_account_id>:oidc-provider/<name>-oidc.s3.<aws_region>.amazonaws.com
```

where **openid-configuration** is a discovery document and **keys.json** is a JSON web key set file.

This command also creates a YAML configuration file in **/<path_to_ccoctl_output_dir>/manifests/cluster-authentication-02-config.yaml**. This file sets the issuer URL field for the service account tokens that the cluster generates, so that the AWS IAM identity provider trusts the tokens.

3. Create IAM roles for each component in the cluster:

   a. Set a **$RELEASE_IMAGE** variable with the release image from your installation file by running the following command:

   ```
   $ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
   ```

   b. Extract the list of **CredentialsRequest** objects from the OpenShift Container Platform release image:

   ```
   $ oc adm release extract \
     --from=$RELEASE_IMAGE \
     --credentials-requests \
     --included \ 1
     --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \
   2
     --to=<path_to_directory_for_credentials_requests> 3
   ```

   1 The **--included** parameter includes only the manifests that your specific cluster configuration requires.

   2 Specify the location of the **install-config.yaml** file.

   3 Specify the path to the directory where you want to store the **CredentialsRequest** objects. If the specified directory does not exist, this command creates it.

   c. Use the **ccoctl** tool to process all **CredentialsRequest** objects by running the following command:

   ```
   $ ccoctl aws create-iam-roles \
     --name=<name> \
     --region=<aws_region> \
     --credentials-requests-dir=<path_to_credentials_requests_directory> \
     --identity-provider-arn=arn:aws:iam::<aws_account_id>:oidc-provider/<name>-oidc.s3.
   <aws_region>.amazonaws.com
   ```

NOTE

For AWS environments that use alternative IAM API endpoints, such as GovCloud, you must also specify your region with the **--region** parameter.

If your cluster uses Technology Preview features that are enabled by the **TechPreviewNoUpgrade** feature set, you must include the **--enable-tech-preview** parameter.

For each **CredentialsRequest** object, **ccoctl** creates an IAM role with a trust policy that is tied to the specified OIDC identity provider, and a permissions policy as defined in each **CredentialsRequest** object from the OpenShift Container Platform release image.

### Verification

- To verify that the OpenShift Container Platform secrets are created, list the files in the **<path_to_ccoctl_output_dir>/manifests** directory:

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

#### Example output

```
cluster-authentication-02-config.yaml
openshift-cloud-credential-operator-cloud-credential-operator-iam-ro-creds-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capa-manager-bootstrap-credentials-credentials.yaml
openshift-cluster-csi-drivers-ebs-cloud-credentials-credentials.yaml
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-aws-cloud-credentials-credentials.yaml
```

You can verify that the IAM roles are created by querying AWS. For more information, refer to AWS documentation on listing IAM roles.

### 3.7.5.2.3. Incorporating the Cloud Credential Operator utility manifests

To implement short-term security credentials managed outside the cluster for individual components, you must move the manifest files that the Cloud Credential Operator utility (**ccoctl**) created to the correct directories for the installation program.

### Prerequisites

- You have configured an account with the cloud platform that hosts your cluster.
- You have configured the Cloud Credential Operator utility (**ccoctl**).
- You have created the cloud provider resources that are required for your cluster with the **ccoctl** utility.

### Procedure

1. If you did not set the **credentialsMode** parameter in the **install-config.yaml** configuration file to **Manual**, modify the value as shown:

**Sample configuration file snippet**

```
apiVersion: v1
baseDomain: example.com
credentialsMode: Manual
# ...
```

2. If you have not previously created installation manifest files, do so by running the following command:

```
$ openshift-install create manifests --dir <installation_directory>
```

where **<installation_directory>** is the directory in which the installation program creates files.

3. Copy the manifests that the **ccoctl** utility generated to the **manifests** directory that the installation program created by running the following command:

```
$ cp /<path_to_ccoctl_output_dir>/manifests/* ./manifests/
```

4. Copy the **tls** directory that contains the private key to the installation directory:

```
$ cp -a /<path_to_ccoctl_output_dir>/tls .
```

## 3.7.6. Deploying the cluster

You can install OpenShift Container Platform on a compatible cloud platform.

**IMPORTANT**

You can run the **create cluster** command of the installation program only once, during initial installation.

**Prerequisites**

- You have configured an account with the cloud platform that hosts your cluster.
- You have the OpenShift Container Platform installation program and the pull secret for your cluster.
- You have verified that the cloud provider account on your host has the correct permissions to deploy the cluster. An account with incorrect permissions causes the installation process to fail with an error message that displays the missing permissions.

**Procedure**

1. In the directory that contains the installation program, initialize the cluster deployment by running the following command:

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1. For **<installation_directory>**, specify the location of your customized **./install-config.yaml** file.

2. To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

2. Optional: Remove or disable the **AdministratorAccess** policy from the IAM account that you used to install the cluster.

NOTE

The elevated permissions provided by the **AdministratorAccess** policy are required only during installation.

### Verification

When the cluster deployment completes successfully:

- The terminal displays directions for accessing your cluster, including a link to the web console and credentials for the **kubeadmin** user.
- Credential information also outputs to **<installation_directory>/.openshift_install.log**.

IMPORTANT

Do not delete the installation program or the files that the installation program creates. Both are required to delete the cluster.

### Example output

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```

IMPORTANT

- The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.
- It is recommended that you use Ignition config files within 12 hours after they are generated because the 24-hour certificate rotates from 16 to 22 hours after the cluster is installed. By using the Ignition config files within 12 hours, you can avoid installation failure if the certificate update runs during installation.

### 3.7.7. Logging in to the cluster by using the CLI

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

#### Prerequisites

- You deployed an OpenShift Container Platform cluster.
- You installed the OpenShift CLI (**oc**).

#### Procedure

1. Export the **kubeadmin** credentials by running the following command:

   ```
   $ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
   ```

   1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration by running the following command:

   ```
   $ oc whoami
   ```

   **Example output**

   ```
   system:admin
   ```

### 3.7.8. Logging in to the cluster by using the web console

The **kubeadmin** user exists by default after an OpenShift Container Platform installation. You can log in to your cluster as the **kubeadmin** user by using the OpenShift Container Platform web console.

#### Prerequisites

- You have access to the installation host.
- You completed a cluster installation and all cluster Operators are available.

#### Procedure

1. Obtain the password for the **kubeadmin** user from the **kubeadmin-password** file on the installation host:

   ```
   $ cat <installation_directory>/auth/kubeadmin-password
   ```

NOTE

Alternatively, you can obtain the **kubeadmin** password from the **<installation_directory>/.openshift_install.log** log file on the installation host.

2. List the OpenShift Container Platform web console route:

```
$ oc get routes -n openshift-console | grep 'console-openshift'
```

NOTE

Alternatively, you can obtain the OpenShift Container Platform route from the **<installation_directory>/.openshift_install.log** log file on the installation host.

**Example output**

```
console     console-openshift-console.apps.<cluster_name>.<base_domain>            console
https   reencrypt/Redirect   None
```

3. Navigate to the route detailed in the output of the preceding command in a web browser and log in as the **kubeadmin** user.

**Additional resources**

- Accessing the web console

### 3.7.9. Next steps

- Validating an installation.
- Customize your cluster.
- If necessary, you can Remote health reporting.
- If necessary, you can remove cloud provider credentials.

## 3.8. INSTALLING A CLUSTER ON AWS INTO A GOVERNMENT REGION

In OpenShift Container Platform version 4.19, you can install a cluster on Amazon Web Services (AWS) into a government region. To configure the region, modify parameters in the **install-config.yaml** file before you install the cluster.

### 3.8.1. Prerequisites

- You reviewed details about the OpenShift Container Platform installation and update processes.
- You read the documentation on selecting a cluster installation method and preparing it for users.
- You configured an AWS account to host the cluster.

IMPORTANT

If you have an AWS profile stored on your computer, it must not use a temporary session token that you generated while using a multi-factor authentication device. The cluster continues to use your current AWS credentials to create AWS resources for the entire life of the cluster, so you must use long-term credentials. To generate appropriate keys, see Managing Access Keys for IAM Users in the AWS documentation. You can supply the keys when you run the installation program.

- If you use a firewall, you configured it to allow the sites that your cluster requires access to.

### 3.8.2. AWS government regions

OpenShift Container Platform supports deploying a cluster to an AWS GovCloud (US) region.

The following AWS GovCloud partitions are supported:

- **us-gov-east-1**
- **us-gov-west-1**

### 3.8.3. Installation requirements

Before you can install the cluster, you must:

- Provide an existing private AWS VPC and subnets to host the cluster.
  Public zones are not supported in Route 53 in AWS GovCloud. As a result, clusters must be private when you deploy to an AWS government region.
- Manually create the installation configuration file (**install-config.yaml**).

### 3.8.4. Private clusters

You can deploy a private OpenShift Container Platform cluster that does not expose external endpoints. Private clusters are accessible from only an internal network and are not visible to the internet.

NOTE

Public zones are not supported in Route 53 in an AWS GovCloud Region. Therefore, clusters must be private if they are deployed to an AWS GovCloud Region.

By default, OpenShift Container Platform is provisioned to use publicly-accessible DNS and endpoints. A private cluster sets the DNS, Ingress Controller, and API server to private when you deploy your cluster. This means that the cluster resources are only accessible from your internal network and are not visible to the internet.

IMPORTANT

If the cluster has any public subnets, load balancer services created by administrators might be publicly accessible. To ensure cluster security, verify that these services are explicitly annotated as private.

To deploy a private cluster, you must:

- Use existing networking that meets your requirements. Your cluster resources might be shared between other clusters on the network.
- Deploy from a machine that has access to:
  - The API services for the cloud to which you provision.
  - The hosts on the network that you provision.
  - The internet to obtain installation media.

You can use any machine that meets these access requirements and follows your company's guidelines. For example, this machine can be a bastion host on your cloud network or a machine that has access to the network through a VPN.

#### 3.8.4.1. Private clusters in AWS

To create a private cluster on Amazon Web Services (AWS), you must provide an existing private VPC and subnets to host the cluster. The installation program must also be able to resolve the DNS records that the cluster requires. The installation program configures the Ingress Operator and API server for access from only the private network.

The cluster still requires access to internet to access the AWS APIs.

The following items are not required or created when you install a private cluster:

- Public subnets
- Public load balancers, which support public ingress
- A public Route 53 zone that matches the **baseDomain** for the cluster

The installation program does use the **baseDomain** that you specify to create a private Route 53 zone and the required records for the cluster. The cluster is configured so that the Operators do not create public records for the cluster and all cluster machines are placed in the private subnets that you specify.

##### 3.8.4.1.1. Limitations

The ability to add public functionality to a private cluster is limited.

- You cannot make the Kubernetes API endpoints public after installation without taking additional actions, including creating public subnets in the VPC for each availability zone in use, creating a public load balancer, and configuring the control plane security groups to allow traffic from the internet on 6443 (Kubernetes API port).
- If you use a public Service type load balancer, you must tag a public subnet in each availability zone with **kubernetes.io/cluster/<cluster-infra-id>: shared** so that AWS can use them to create public load balancers.

### 3.8.5. About using a custom VPC

In OpenShift Container Platform 4.19, you can deploy a cluster into existing subnets in an existing Amazon Virtual Private Cloud (VPC) in Amazon Web Services (AWS). By deploying OpenShift Container Platform into an existing AWS VPC, you might be able to avoid limit constraints in new accounts or more easily abide by the operational constraints that your company's guidelines set. If you cannot obtain the infrastructure creation permissions that are required to create the VPC yourself, use this installation option.

Because the installation program cannot know what other components are also in your existing subnets, it cannot choose subnet CIDRs and so forth on your behalf. You must configure networking for the subnets that you install your cluster to yourself.

### 3.8.5.1. Requirements for using your VPC

The installation program no longer creates the following components:

- Internet gateways
- NAT gateways
- Subnets
- Route tables
- VPCs
- VPC DHCP options
- VPC endpoints

**NOTE**

The installation program requires that you use the cloud-provided DNS server. Using a custom DNS server is not supported and causes the installation to fail.

If you use a custom VPC, you must correctly configure it and its subnets for the installation program and the cluster to use. See Create a VPC in the Amazon Web Services documentation for more information about AWS VPC console wizard configurations and creating and managing an AWS VPC.

The installation program cannot:

- Subdivide network ranges for the cluster to use.
- Set route tables for the subnets.
- Set VPC options like DHCP.

You must complete these tasks before you install the cluster. See VPC networking components and Route tables for your VPC for more information on configuring networking in an AWS VPC.

Your VPC must meet the following characteristics:

- The VPC must not use the **kubernetes.io/cluster/.*: owned**, **Name**, and **openshift.io/cluster** tags.
  The installation program modifies your subnets to add the **kubernetes.io/cluster/.*: shared** tag, so your subnets must have at least one free tag slot available for it. See Tag Restrictions in the AWS documentation to confirm that the installation program can add a tag to each subnet that you specify. You cannot use a **Name** tag, because it overlaps with the EC2 **Name** field and the installation fails.

- If you want to extend your OpenShift Container Platform cluster into an AWS Outpost and have an existing Outpost subnet, the existing subnet must use the **kubernetes.io/cluster/unmanaged: true** tag. If you do not apply this tag, the installation might fail due to the Cloud Controller Manager creating a service load balancer in the Outpost subnet, which is an unsupported configuration.

- You must enable the **enableDnsSupport** and **enableDnsHostnames** attributes in your VPC, so that the cluster can use the Route 53 zones that are attached to the VPC to resolve cluster's internal DNS records. See DNS Support in Your VPC in the AWS documentation.
  If you prefer to use your own Route 53 hosted private zone, you must associate the existing hosted zone with your VPC prior to installing a cluster. You can define your hosted zone using the **platform.aws.hostedZone** and **platform.aws.hostedZoneRole** fields in the **install-config.yaml** file. You can use a private hosted zone from another account by sharing it with the account where you install the cluster. If you use a private hosted zone from another account, you must use the **Passthrough** or **Manual** credentials mode.

If you are working in a disconnected environment, you are unable to reach the public IP addresses for EC2, ELB, and S3 endpoints. Depending on the level to which you want to restrict internet traffic during the installation, the following configuration options are available:

#### 3.8.5.1.1. Option 1: Create VPC endpoints

Create a VPC endpoint and attach it to the subnets that the clusters are using. Name the endpoints as follows:

- **ec2.<aws_region>.amazonaws.com**
- **elasticloadbalancing.<aws_region>.amazonaws.com**
- **s3.<aws_region>.amazonaws.com**

With this option, network traffic remains private between your VPC and the required AWS services.

#### 3.8.5.1.2. Option 2: Create a proxy without VPC endpoints

As part of the installation process, you can configure an HTTP or HTTPS proxy. With this option, internet traffic goes through the proxy to reach the required AWS services.

#### 3.8.5.1.3. Option 3: Create a proxy with VPC endpoints

As part of the installation process, you can configure an HTTP or HTTPS proxy with VPC endpoints. Create a VPC endpoint and attach it to the subnets that the clusters are using. Name the endpoints as follows:

- **ec2.<aws_region>.amazonaws.com**
- **elasticloadbalancing.<aws_region>.amazonaws.com**
- **s3.<aws_region>.amazonaws.com**

When configuring the proxy in the **install-config.yaml** file, add these endpoints to the **noProxy** field. With this option, the proxy prevents the cluster from accessing the internet directly. However, network traffic remains private between your VPC and the required AWS services.

### Required VPC components

You must provide a suitable VPC and subnets that allow communication to your machines.

| Component | AWS type | Description |
|---|---|---|
| VPC | • **AWS::EC2::VPC**<br>• **AWS::EC2::VPCEndpoint** | You must provide a public VPC for the cluster to use. The VPC uses an endpoint that references the route tables for each subnet to improve communication with the registry that is hosted in S3. |
| Public subnets | • **AWS::EC2::Subnet**<br>• **AWS::EC2::SubnetNetworkAclAssociation** | Your VPC must have public subnets for between 1 and 3 availability zones and associate them with appropriate Ingress rules. |
| Internet gateway | • **AWS::EC2::InternetGateway**<br>• **AWS::EC2::VPCGatewayAttachment**<br>• **AWS::EC2::RouteTable**<br>• **AWS::EC2::Route**<br>• **AWS::EC2::SubnetRouteTableAssociation**<br>• **AWS::EC2::NatGateway**<br>• **AWS::EC2::EIP** | You must have a public internet gateway, with public routes, attached to the VPC. In the provided templates, each public subnet has a NAT gateway with an EIP address. These NAT gateways allow cluster resources, like private subnet instances, to reach the internet and are not required for some restricted network or proxy scenarios. |
| Network access control | • **AWS::EC2::NetworkAcl**<br>• **AWS::EC2::NetworkAclEntry** | You must allow the VPC to access the following ports: |

| Port | Reason |
|---|---|
| **80** | Inbound HTTP traffic |
| **443** | Inbound HTTPS traffic |
| **22** | Inbound SSH traffic |
| **1024 - 65535** | Inbound ephemeral traffic |

| Compone nt | AWS type | Description | |
|---|---|---|---|
| | | **0 - 65535** | Outbound ephemeral traffic |
| Private subnets | • **AWS::EC2::Subnet**<br>• **AWS::EC2::RouteTable**<br>• **AWS::EC2::SubnetRouteTableAssociation** | Your VPC can have private subnets. The provided CloudFormation templates can create private subnets for between 1 and 3 availability zones. If you use private subnets, you must provide appropriate routes and tables for them. | |

### 3.8.5.2. VPC validation

To ensure that the subnets that you provide are suitable, the installation program confirms the following data:

- All the subnets that you specify exist.
- You provide private subnets.
- The subnet CIDRs belong to the machine CIDR that you specified.
- You provide subnets for each availability zone. Each availability zone contains no more than one public and one private subnet. If you use a private cluster, provide only a private subnet for each availability zone. Otherwise, provide exactly one public and private subnet for each availability zone.
- You provide a public subnet for each private subnet availability zone. Machines are not provisioned in availability zones that you do not provide private subnets for.

If you destroy a cluster that uses an existing VPC, the VPC is not deleted. When you remove the OpenShift Container Platform cluster from a VPC, the **kubernetes.io/cluster/.*: shared** tag is removed from the subnets that it used.

### 3.8.5.3. Division of permissions

Starting with OpenShift Container Platform 4.3, you do not need all of the permissions that are required for an installation program-provisioned infrastructure cluster to deploy a cluster. This change mimics the division of permissions that you might have at your company: some individuals can create different resource in your clouds than others. For example, you might be able to create application-specific items, like instances, buckets, and load balancers, but not networking-related components such as VPCs, subnets, or ingress rules.

The AWS credentials that you use when you create your cluster do not need the networking permissions that are required to make VPCs and core networking components within the VPC, such as subnets, routing tables, internet gateways, NAT, and VPN. You still need permission to make the application resources that the machines within the cluster require, such as ELBs, security groups, S3 buckets, and nodes.

### 3.8.5.4. Isolation between clusters

If you deploy OpenShift Container Platform to an existing network, the isolation of cluster services is reduced in the following ways:

- You can install multiple OpenShift Container Platform clusters in the same VPC.
- ICMP ingress is allowed from the entire network.
- TCP 22 ingress (SSH) is allowed to the entire network.
- Control plane TCP 6443 ingress (Kubernetes API) is allowed to the entire network.
- Control plane TCP 22623 ingress (MCS) is allowed to the entire network.

#### 3.8.5.5. Optional: AWS security groups

By default, the installation program creates and attaches security groups to control plane and compute machines. The rules associated with the default security groups cannot be modified.

However, you can apply additional existing AWS security groups, which are associated with your existing VPC, to control plane and compute machines. Applying custom security groups can help you meet the security needs of your organization, in such cases where you need to control the incoming or outgoing traffic of these machines.

As part of the installation process, you apply custom security groups by modifying the **install-config.yaml** file before deploying the cluster.

For more information, see "Applying existing AWS security groups to the cluster".

### 3.8.6. Obtaining an AWS Marketplace image

If you are deploying an OpenShift Container Platform cluster using an AWS Marketplace image, you must first subscribe through AWS. Subscribing to the offer provides you with the AMI ID that the installation program uses to deploy compute nodes.

**NOTE**

You should only modify the RHCOS image for compute machines to use an AWS Marketplace image. Control plane machines and infrastructure nodes do not require an OpenShift Container Platform subscription and use the public RHCOS default image by default, which does not incur subscription costs on your AWS bill. Therefore, you should not modify the cluster default boot image or the control plane boot images. Applying the AWS Marketplace image to them will incur additional licensing costs that cannot be recovered.

**Prerequisites**

- You have an AWS account to purchase the offer. This account does not have to be the same account that is used to install the cluster.

**Procedure**

1. Complete the OpenShift Container Platform subscription from the AWS Marketplace.
2. Record the AMI ID for your specific AWS Region. As part of the installation process, you must update the **install-config.yaml** file with this value before deploying the cluster.

**Sample install-config.yaml file with AWS Marketplace compute nodes**

```
apiVersion: v1
baseDomain: example.com
compute:
- hyperthreading: Enabled
  name: worker
  platform:
    aws:
      amiID: ami-06c4d345f7c207239 (1)
      type: m5.4xlarge
  replicas: 3
metadata:
  name: test-cluster
platform:
  aws:
    region: us-east-2 (2)
sshKey: ssh-ed25519 AAAA...
pullSecret: '{"auths": ...}'
```

(1) The AMI ID from your AWS Marketplace subscription.

(2) Your AMI ID is associated with a specific AWS Region. When creating the installation configuration file, ensure that you select the same AWS Region that you specified when configuring your subscription.

### 3.8.7. Manually creating the installation configuration file

To customise your OpenShift Container Platform deployment and meet specific network requirements, manually create the installation configuration file. This ensures that the installation program uses your tailored settings rather than default values during the setup process.

#### Prerequisites

- You have an SSH public key on your local machine for use with the installation program. You can use the key for SSH authentication onto your cluster nodes for debugging and disaster recovery.
- You have obtained the OpenShift Container Platform installation program and the pull secret for your cluster.

#### Procedure

1. Create an installation directory to store your required installation assets in:

```
$ mkdir <installation_directory>
```

IMPORTANT

You must create a directory. Some installation assets, such as bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

2. Customize the provided sample **install-config.yaml** file template and save the file in the **<installation_directory>**.

NOTE

You must name this configuration file **install-config.yaml**.

3. Back up the **install-config.yaml** file so that you can use it to install many clusters.

IMPORTANT

Back up the **install-config.yaml** file now, because the installation process consumes the file in the next step.

**Additional resources**

- Installation configuration parameters for AWS

### 3.8.7.1. Minimum resource requirements for cluster installation

Each created cluster must meet minimum requirements so that the cluster runs as expected.

**Table 3.16. Minimum resource requirements**

| Machine | Operating System | vCPU [1] | Virtual RAM | Storage | Input/Output Per Second (IOPS)[2] |
|---|---|---|---|---|---|
| Bootstrap | RHCOS | 4 | 16 GB | 100 GB | 300 |
| Control plane | RHCOS | 4 | 16 GB | 100 GB | 300 |
| Compute | RHCOS, RHEL 8.6 and later [3] | 2 | 8 GB | 100 GB | 300 |

1. One vCPU is equivalent to one physical core when simultaneous multithreading (SMT), or Hyper-Threading, is not enabled. When enabled, use the following formula to calculate the corresponding ratio: (threads per core × cores) × sockets = vCPUs.

2. OpenShift Container Platform and Kubernetes are sensitive to disk performance, and faster storage is recommended, particularly for etcd on the control plane nodes which require a 10 ms

p99 fsync duration. Note that on many cloud platforms, storage size and IOPS scale together, so you might need to over-allocate storage volume to obtain sufficient performance.

3. As with all user-provisioned installations, if you choose to use RHEL compute machines in your cluster, you take responsibility for all operating system life cycle management and maintenance, including performing system updates, applying patches, and completing all other required tasks. Use of RHEL 7 compute machines is deprecated and has been removed in OpenShift Container Platform 4.10 and later.

NOTE

For OpenShift Container Platform version 4.19, RHCOS is based on RHEL version 9.6, which updates the micro-architecture requirements. The following list contains the minimum instruction set architectures (ISA) that each architecture requires:

- x86-64 architecture requires x86-64-v2 ISA
- ARM64 architecture requires ARMv8.0-A ISA
- IBM Power architecture requires Power 9 ISA
- s390x architecture requires z14 ISA

For more information, see Architectures (RHEL documentation).

If an instance type for your platform meets the minimum requirements for cluster machines, it is supported to use in OpenShift Container Platform.

**Additional resources**

- Optimizing storage

### 3.8.7.2. Tested instance types for AWS

The following Amazon Web Services (AWS) instance types have been tested with OpenShift Container Platform.

NOTE

Use the machine types included in the following charts for your AWS instances. If you use an instance type that is not listed in the chart, ensure that the instance size you use matches the minimum resource requirements that are listed in the section named "Minimum resource requirements for cluster installation".

**Example 3.10. Machine types based on 64-bit x86 architecture**

- **c4.***
- **c5.***
- **c5a.***
- **i3.***

- **m4.***
- **m5.***
- **m5a.***
- **m6a.***
- **m6i.***
- **r4.***
- **r5.***
- **r5a.***
- **r6i.***
- **t3.***
- **t3a.***

### 3.8.7.3. Tested instance types for AWS on 64-bit ARM infrastructures

The following Amazon Web Services (AWS) 64-bit ARM instance types have been tested with OpenShift Container Platform.

NOTE

Use the machine types included in the following charts for your AWS ARM instances. If you use an instance type that is not listed in the chart, ensure that the instance size you use matches the minimum resource requirements that are listed in "Minimum resource requirements for cluster installation".

**Example 3.11. Machine types based on 64-bit ARM architecture**

- **c6g.***
- **c7g.***
- **m6g.***
- **m7g.***
- **r8g.***

### 3.8.7.4. Sample customized install-config.yaml file for AWS

You can customize the installation configuration file (**install-config.yaml**) to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.

IMPORTANT

This sample YAML file is provided for reference only. You must obtain your **install-config.yaml** file by using the installation program and modify it. For a full list and description of all installation configuration parameters, see *Installation configuration parameters for AWS*.

### Sample **install-config.yaml** file for AWS

```
apiVersion: v1 1
baseDomain: example.com
sshKey: ssh-ed25519 AAAA...
pullSecret: '{"auths": ...}'
metadata:
  name: example-cluster
controlPlane: 2
  name: master
  platform:
    aws:
      type: m6i.xlarge
  replicas: 3
compute: 3
-  name: worker
  platform:
    aws:
      type: c5.4xlarge
  replicas: 3
networking: 4
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
platform: 5
  aws:
    region: us-west-2
```

1 Parameters at the first level of indentation apply to the cluster globally.

2 The **controlPlane** stanza applies to control plane machines.

3 The **compute** stanza applies to compute machines.

4 The **networking** stanza applies to the cluster networking configuration. If you do not provide networking values, the installation program provides default values.

5 The **platform** stanza applies to the infrastructure platform that hosts the cluster.

#### Additional resources

- Installation configuration parameters for AWS

### 3.8.7.5. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

## Prerequisites

- You have an existing **install-config.yaml** file.
- You reviewed the sites that your cluster requires access to and determined whether any of them need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. You added sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.

NOTE

The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.

For installations on Amazon Web Services (AWS), Google Cloud, Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

## Procedure

1. Edit your **install-config.yaml** file and add the proxy settings. For example:

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: ec2.<aws_region>.amazonaws.com,elasticloadbalancing.
<aws_region>.amazonaws.com,s3.<aws_region>.amazonaws.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1 A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**.

2 A proxy URL to use for creating HTTPS connections outside the cluster.

3 A comma-separated list of destination domain names, IP addresses, or other network CIDRs to exclude from proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use * to bypass the proxy for all destinations. If you have added the Amazon **EC2**,**Elastic Load Balancing**, and **S3** VPC endpoints to your VPC, you must add these endpoints to the **noProxy** field.

4 If provided, the installation program generates a config map that is named **user-ca-bundle** in the **openshift-config** namespace that contains one or more additional CA certificates

that are required for proxying HTTPS connections. The Cluster Network Operator then creates a **trusted-ca-bundle** config map that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this config map is referenced in the **trustedCA** field of the **Proxy** object. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle.

5 Optional: The policy to determine the configuration of the **Proxy** object to reference the **user-ca-bundle** config map in the **trustedCA** field. The allowed values are **Proxyonly** and **Always**. Use **Proxyonly** to reference the **user-ca-bundle** config map only when **http/https** proxy is configured. Use **Always** to always reference the **user-ca-bundle** config map. The default value is **Proxyonly**.

NOTE

The installation program does not support the proxy **readinessEndpoints** field.

NOTE

If the installer times out, restart and then complete the deployment by using the **wait-for** command of the installer. For example:

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. Save the file and reference it when installing OpenShift Container Platform.

The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster Proxy** object is still created, but it will have a nil **spec**.

NOTE

Only the **Proxy** object named **cluster** is supported, and no additional proxies can be created.

#### 3.8.7.6. Applying existing AWS security groups to the cluster

Applying existing AWS security groups to your control plane and compute machines can help you meet the security needs of your organization, in such cases where you need to control the incoming or outgoing traffic of these machines.

**Prerequisites**

- You have created the security groups in AWS. For more information, see the AWS documentation about working with security groups.
- The security groups must be associated with the existing VPC that you are deploying the cluster to. The security groups cannot be associated with another VPC.
- You have an existing **install-config.yaml** file.

**Procedure**

1. In the **install-config.yaml** file, edit the **compute.platform.aws.additionalSecurityGroupIDs** parameter to specify one or more custom security groups for your compute machines.

2. Edit the **controlPlane.platform.aws.additionalSecurityGroupIDs** parameter to specify one or more custom security groups for your control plane machines.

3. Save the file and reference it when deploying the cluster.

**Sample install-config.yaml file that specifies custom security groups**

```
# ...
compute:
- hyperthreading: Enabled
  name: worker
  platform:
    aws:
      additionalSecurityGroupIDs:
        - sg-1 (1)
        - sg-2
  replicas: 3
controlPlane:
  hyperthreading: Enabled
  name: master
  platform:
    aws:
      additionalSecurityGroupIDs:
        - sg-3
        - sg-4
  replicas: 3
platform:
  aws:
    region: us-east-1
    subnets: (2)
      - subnet-1
      - subnet-2
      - subnet-3
```

(1) Specify the name of the security group as it appears in the Amazon EC2 console, including the **sg** prefix.

(2) Specify subnets for each availability zone that your cluster uses.

### 3.8.8. Alternatives to storing administrator-level secrets in the kube-system project

By default, administrator secrets are stored in the **kube-system** project. If you configured the **credentialsMode** parameter in the **install-config.yaml** file to **Manual**, you must use one of the following alternatives:

- To manage long-term cloud credentials manually, follow the procedure in Manually creating long-term credentials.

- To implement short-term credentials that are managed outside the cluster for individual components, follow the procedures in Incorporating the Cloud Credential Operator utility manifests.

#### 3.8.8.1. Manually creating long-term credentials

The Cloud Credential Operator (CCO) can be put into manual mode prior to installation in environments where the cloud identity and access management (IAM) APIs are not reachable, or the administrator prefers not to store an administrator-level credential secret in the cluster **kube-system** namespace.

**Procedure**

1. If you did not set the **credentialsMode** parameter in the **install-config.yaml** configuration file to **Manual**, modify the value as shown:

   **Sample configuration file snippet**

   ```
   apiVersion: v1
   baseDomain: example.com
   credentialsMode: Manual
   # ...
   ```

2. If you have not previously created installation manifest files, do so by running the following command:

   ```
   $ openshift-install create manifests --dir <installation_directory>
   ```

   where **<installation_directory>** is the directory in which the installation program creates files.

3. Set a **$RELEASE_IMAGE** variable with the release image from your installation file by running the following command:

   ```
   $ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
   ```

4. Extract the list of **CredentialsRequest** custom resources (CRs) from the OpenShift Container Platform release image by running the following command:

   ```
   $ oc adm release extract \
     --from=$RELEASE_IMAGE \
     --credentials-requests \
     --included \ 1
     --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
     --to=<path_to_directory_for_credentials_requests> 3
   ```

   1 The **--included** parameter includes only the manifests that your specific cluster configuration requires.

   2 Specify the location of the **install-config.yaml** file.

   3 Specify the path to the directory where you want to store the **CredentialsRequest** objects. If the specified directory does not exist, this command creates it.

   This command creates a YAML file for each **CredentialsRequest** object.

   **Sample CredentialsRequest object**

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
    - effect: Allow
      action:
      - iam:GetUser
      - iam:GetUserPolicy
      - iam:ListAccessKeys
      resource: "*"
  ...
```

5. Create YAML files for secrets in the **openshift-install** manifests directory that you generated previously. The secrets must be stored using the namespace and secret name defined in the **spec.secretRef** for each **CredentialsRequest** object.

   **Sample CredentialsRequest object with secrets**

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
    - effect: Allow
      action:
      - s3:CreateBucket
      - s3:DeleteBucket
      resource: "*"
      ...
  secretRef:
    name: <component_secret>
    namespace: <component_namespace>
  ...
```

   **Sample Secret object**

```
apiVersion: v1
kind: Secret
metadata:
  name: <component_secret>
  namespace: <component_namespace>
data:
```

```
aws_access_key_id: <base64_encoded_aws_access_key_id>
aws_secret_access_key: <base64_encoded_aws_secret_access_key>
```

IMPORTANT

Before upgrading a cluster that uses manually maintained credentials, you must ensure that the CCO is in an upgradeable state.

### 3.8.8.2. Configuring an AWS cluster to use short-term credentials

To install a cluster that is configured to use the AWS Security Token Service (STS), you must configure the CCO utility and create the required AWS resources for your cluster.

#### 3.8.8.2.1. Configuring the Cloud Credential Operator utility

To create and manage cloud credentials from outside of the cluster when the Cloud Credential Operator (CCO) is operating in manual mode, extract and prepare the CCO utility (**ccoctl**) binary.

NOTE

The **ccoctl** utility is a Linux binary that must run in a Linux environment.

**Prerequisites**

- You have access to an OpenShift Container Platform account with cluster administrator access.
- You have installed the OpenShift CLI (**oc**).
- You have created an AWS account for the **ccoctl** utility to use with the following permissions:
  **Required iam permissions**
  - **iam:CreateOpenIDConnectProvider**
  - **iam:CreateRole**
  - **iam:DeleteOpenIDConnectProvider**
  - **iam:DeleteRole**
  - **iam:DeleteRolePolicy**
  - **iam:GetOpenIDConnectProvider**
  - **iam:GetRole**
  - **iam:GetUser**
  - **iam:ListOpenIDConnectProviders**
  - **iam:ListRolePolicies**
  - **iam:ListRoles**
  - **iam:PutRolePolicy**

- **iam:TagOpenIDConnectProvider**
- **iam:TagRole**

Required **s3** permissions

- **s3:CreateBucket**
- **s3:DeleteBucket**
- **s3:DeleteObject**
- **s3:GetBucketAcl**
- **s3:GetBucketTagging**
- **s3:GetObject**
- **s3:GetObjectAcl**
- **s3:GetObjectTagging**
- **s3:ListBucket**
- **s3:PutBucketAcl**
- **s3:PutBucketPolicy**
- **s3:PutBucketPublicAccessBlock**
- **s3:PutBucketTagging**
- **s3:PutObject**
- **s3:PutObjectAcl**
- **s3:PutObjectTagging**

Required **cloudfront** permissions

- **cloudfront:ListCloudFrontOriginAccessIdentities**
- **cloudfront:ListDistributions**
- **cloudfront:ListTagsForResource**

- If you plan to store the OIDC configuration in a private S3 bucket that is accessed by the IAM identity provider through a public CloudFront distribution URL, the AWS account that runs the **ccoctl** utility requires the following additional permissions:
  - **cloudfront:CreateCloudFrontOriginAccessIdentity**
  - **cloudfront:CreateDistribution**
  - **cloudfront:DeleteCloudFrontOriginAccessIdentity**

- **cloudfront:DeleteDistribution**
- **cloudfront:GetCloudFrontOriginAccessIdentity**
- **cloudfront:GetCloudFrontOriginAccessIdentityConfig**
- **cloudfront:GetDistribution**
- **cloudfront:TagResource**
- **cloudfront:UpdateDistribution**

NOTE

These additional permissions support the use of the **--create-private-s3-bucket** option when processing credentials requests with the **ccoctl aws create-all** command.

Procedure

1. Set a variable for the OpenShift Container Platform release image by running the following command:

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. Obtain the CCO container image from the OpenShift Container Platform release image by running the following command:

```
$ CCO_IMAGE=$(oc adm release info --image-for='cloud-credential-operator'
$RELEASE_IMAGE -a ~/.pull-secret)
```

NOTE

Ensure that the architecture of the **$RELEASE_IMAGE** matches the architecture of the environment in which you will use the **ccoctl** tool.

3. Extract the **ccoctl** binary from the CCO container image within the OpenShift Container Platform release image by running the following command:

```
$ oc image extract $CCO_IMAGE \
  --file="/usr/bin/ccoctl.<rhel_version>" \ 1
  -a ~/.pull-secret
```

1 For **<rhel_version>**, specify the value that corresponds to the version of Red Hat Enterprise Linux (RHEL) that the host uses. If no value is specified, **ccoctl.rhel8** is used by default. The following values are valid:

- **rhel8**: Specify this value for hosts that use RHEL 8.
- **rhel9**: Specify this value for hosts that use RHEL 9.

NOTE

The **ccoctl** binary is created in the directory from where you executed the command and not in **/usr/bin/**. You must rename the directory or move the **ccoctl.<rhel_version>** binary to **ccoctl**.

4. Change the permissions to make **ccoctl** executable by running the following command:

```
$ chmod 775 ccoctl
```

### Verification

- To verify that **ccoctl** is ready to use, display the help file. Use a relative file name when you run the command, for example:

```
$ ./ccoctl
```

#### Example output

```
OpenShift credentials provisioning tool

Usage:
  ccoctl [command]

Available Commands:
  aws          Manage credentials objects for AWS cloud
  azure        Manage credentials objects for Azure
  gcp          Manage credentials objects for Google cloud
  help         Help about any command
  ibmcloud     Manage credentials objects for {ibm-cloud-title}
  nutanix      Manage credentials objects for Nutanix

Flags:
  -h, --help   help for ccoctl

Use "ccoctl [command] --help" for more information about a command.
```

#### 3.8.8.2.2. Creating AWS resources with the Cloud Credential Operator utility

You have the following options when creating AWS resources:

- You can use the **ccoctl aws create-all** command to create the AWS resources automatically. This is the quickest way to create the resources. See Creating AWS resources with a single command.

- If you need to review the JSON files that the **ccoctl** tool creates before modifying AWS resources, or if the process the **ccoctl** tool uses to create AWS resources automatically does not meet the requirements of your organization, you can create the AWS resources individually. See Creating AWS resources individually .

##### 3.8.8.2.2.1. Creating AWS resources with a single command

If the process the **ccoctl** tool uses to create AWS resources automatically meets the requirements of your organization, you can use the **ccoctl aws create-all** command to automate the creation of AWS resources.

Otherwise, you can create the AWS resources individually. For more information, see "Creating AWS resources individually".

NOTE

By default, **ccoctl** creates objects in the directory in which the commands are run. To create the objects in a different directory, use the **--output-dir** flag. This procedure uses **<path_to_ccoctl_output_dir>** to refer to this directory.

### Prerequisites

You must have:

- Extracted and prepared the **ccoctl** binary.

### Procedure

1. Set a **$RELEASE_IMAGE** variable with the release image from your installation file by running the following command:

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. Extract the list of **CredentialsRequest** objects from the OpenShift Container Platform release image by running the following command:

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

1 The **--included** parameter includes only the manifests that your specific cluster configuration requires.

2 Specify the location of the **install-config.yaml** file.

3 Specify the path to the directory where you want to store the **CredentialsRequest** objects. If the specified directory does not exist, this command creates it.

NOTE

This command might take a few moments to run.

3. Use the **ccoctl** tool to process all **CredentialsRequest** objects by running the following command:

```
$ ccoctl aws create-all \
--name=<name> \ 1
--region=<aws_region> \ 2
--credentials-requests-dir=<path_to_credentials_requests_directory> \ 3
--output-dir=<path_to_ccoctl_output_dir> \ 4
--create-private-s3-bucket 5
```

1. Specify the name used to tag any cloud resources that are created for tracking.
2. Specify the AWS region in which cloud resources will be created.
3. Specify the directory containing the files for the component **CredentialsRequest** objects.
4. Optional: Specify the directory in which you want the **ccoctl** utility to create objects. By default, the utility creates objects in the directory in which the commands are run.
5. Optional: By default, the **ccoctl** utility stores the OpenID Connect (OIDC) configuration files in a public S3 bucket and uses the S3 URL as the public OIDC endpoint. To store the OIDC configuration in a private S3 bucket that is accessed by the IAM identity provider through a public CloudFront distribution URL instead, use the **--create-private-s3-bucket** parameter.

NOTE

If your cluster uses Technology Preview features that are enabled by the **TechPreviewNoUpgrade** feature set, you must include the **--enable-tech-preview** parameter.

Verification

- To verify that the OpenShift Container Platform secrets are created, list the files in the **<path_to_ccoctl_output_dir>/manifests** directory:

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

Example output

```
cluster-authentication-02-config.yaml
openshift-cloud-credential-operator-cloud-credential-operator-iam-ro-creds-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capa-manager-bootstrap-credentials-credentials.yaml
openshift-cluster-csi-drivers-ebs-cloud-credentials-credentials.yaml
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-aws-cloud-credentials-credentials.yaml
```

You can verify that the IAM roles are created by querying AWS. For more information, refer to AWS documentation on listing IAM roles.

#### 3.8.8.2.2.2. Creating AWS resources individually

You can use the **ccoctl** tool to create AWS resources individually. This option might be useful for an organization that shares the responsibility for creating these resources among different users or departments.

Otherwise, you can use the **ccoctl aws create-all** command to create the AWS resources automatically. For more information, see "Creating AWS resources with a single command".

NOTE

By default, **ccoctl** creates objects in the directory in which the commands are run. To create the objects in a different directory, use the **--output-dir** flag. This procedure uses **<path_to_ccoctl_output_dir>** to refer to this directory.

Some **ccoctl** commands make AWS API calls to create or modify AWS resources. You can use the **--dry-run** flag to avoid making API calls. Using this flag creates JSON files on the local file system instead. You can review and modify the JSON files and then apply them with the AWS CLI tool using the **--cli-input-json** parameters.

### Prerequisites

- Extract and prepare the **ccoctl** binary.

### Procedure

1. Generate the public and private RSA key files that are used to set up the OpenID Connect provider for the cluster by running the following command:

```
$ ccoctl aws create-key-pair
```

#### Example output

```
2021/04/13 11:01:02 Generating RSA keypair
2021/04/13 11:01:03 Writing private key to /<path_to_ccoctl_output_dir>/serviceaccount-signer.private
2021/04/13 11:01:03 Writing public key to /<path_to_ccoctl_output_dir>/serviceaccount-signer.public
2021/04/13 11:01:03 Copying signing key for use by installer
```

where **serviceaccount-signer.private** and **serviceaccount-signer.public** are the generated key files.

This command also creates a private key that the cluster requires during installation in **/<path_to_ccoctl_output_dir>/tls/bound-service-account-signing-key.key**.

2. Create an OpenID Connect identity provider and S3 bucket on AWS by running the following command:

```
$ ccoctl aws create-identity-provider \
  --name=<name> \ 1
  --region=<aws_region> \ 2
  --public-key-file=<path_to_ccoctl_output_dir>/serviceaccount-signer.public 3
```

1 **<name>** is the name used to tag any cloud resources that are created for tracking.

2 **<aws-region>** is the AWS region in which cloud resources will be created.

3 **<path_to_ccoctl_output_dir>** is the path to the public key file that the **ccoctl aws create-key-pair** command generated.

### Example output

```
2021/04/13 11:16:09 Bucket <name>-oidc created
2021/04/13 11:16:10 OpenID Connect discovery document in the S3 bucket <name>-oidc at
.well-known/openid-configuration updated
2021/04/13 11:16:10 Reading public key
2021/04/13 11:16:10 JSON web key set (JWKS) in the S3 bucket <name>-oidc at keys.json
updated
2021/04/13 11:16:18 Identity Provider created with ARN: arn:aws:iam::
<aws_account_id>:oidc-provider/<name>-oidc.s3.<aws_region>.amazonaws.com
```

where **openid-configuration** is a discovery document and **keys.json** is a JSON web key set file.

This command also creates a YAML configuration file in **/<path_to_ccoctl_output_dir>/manifests/cluster-authentication-02-config.yaml**. This file sets the issuer URL field for the service account tokens that the cluster generates, so that the AWS IAM identity provider trusts the tokens.

3. Create IAM roles for each component in the cluster:

   a. Set a **$RELEASE_IMAGE** variable with the release image from your installation file by running the following command:

   ```
   $ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
   ```

   b. Extract the list of **CredentialsRequest** objects from the OpenShift Container Platform release image:

   ```
   $ oc adm release extract \
     --from=$RELEASE_IMAGE \
     --credentials-requests \
     --included \ 1
     --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \
   2
     --to=<path_to_directory_for_credentials_requests> 3
   ```

   1 The **--included** parameter includes only the manifests that your specific cluster configuration requires.

   2 Specify the location of the **install-config.yaml** file.

   3 Specify the path to the directory where you want to store the **CredentialsRequest** objects. If the specified directory does not exist, this command creates it.

   c. Use the **ccoctl** tool to process all **CredentialsRequest** objects by running the following command:

```
$ ccoctl aws create-iam-roles \
  --name=<name> \
  --region=<aws_region> \
  --credentials-requests-dir=<path_to_credentials_requests_directory> \
  --identity-provider-arn=arn:aws:iam::<aws_account_id>:oidc-provider/<name>-oidc.s3.
<aws_region>.amazonaws.com
```

NOTE

For AWS environments that use alternative IAM API endpoints, such as GovCloud, you must also specify your region with the **--region** parameter.

If your cluster uses Technology Preview features that are enabled by the **TechPreviewNoUpgrade** feature set, you must include the **--enable-tech-preview** parameter.

For each **CredentialsRequest** object, **ccoctl** creates an IAM role with a trust policy that is tied to the specified OIDC identity provider, and a permissions policy as defined in each **CredentialsRequest** object from the OpenShift Container Platform release image.

#### Verification

- To verify that the OpenShift Container Platform secrets are created, list the files in the **<path_to_ccoctl_output_dir>/manifests** directory:

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

Example output

```
cluster-authentication-02-config.yaml
openshift-cloud-credential-operator-cloud-credential-operator-iam-ro-creds-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capa-manager-bootstrap-credentials-credentials.yaml
openshift-cluster-csi-drivers-ebs-cloud-credentials-credentials.yaml
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-aws-cloud-credentials-credentials.yaml
```

You can verify that the IAM roles are created by querying AWS. For more information, refer to AWS documentation on listing IAM roles.

### 3.8.8.2.3. Incorporating the Cloud Credential Operator utility manifests

To implement short-term security credentials managed outside the cluster for individual components, you must move the manifest files that the Cloud Credential Operator utility (**ccoctl**) created to the correct directories for the installation program.

#### Prerequisites

- You have configured an account with the cloud platform that hosts your cluster.
- You have configured the Cloud Credential Operator utility (**ccoctl**).

- You have created the cloud provider resources that are required for your cluster with the **ccoctl** utility.

Procedure

1. If you did not set the **credentialsMode** parameter in the **install-config.yaml** configuration file to **Manual**, modify the value as shown:

   Sample configuration file snippet

   ```
   apiVersion: v1
   baseDomain: example.com
   credentialsMode: Manual
   # ...
   ```

2. If you have not previously created installation manifest files, do so by running the following command:

   ```
   $ openshift-install create manifests --dir <installation_directory>
   ```

   where **<installation_directory>** is the directory in which the installation program creates files.

3. Copy the manifests that the **ccoctl** utility generated to the **manifests** directory that the installation program created by running the following command:

   ```
   $ cp /<path_to_ccoctl_output_dir>/manifests/* ./manifests/
   ```

4. Copy the **tls** directory that contains the private key to the installation directory:

   ```
   $ cp -a /<path_to_ccoctl_output_dir>/tls .
   ```

### 3.8.9. Deploying the cluster

You can install OpenShift Container Platform on a compatible cloud platform.

IMPORTANT

You can run the **create cluster** command of the installation program only once, during initial installation.

Prerequisites

- You have configured an account with the cloud platform that hosts your cluster.
- You have the OpenShift Container Platform installation program and the pull secret for your cluster.
- You have verified that the cloud provider account on your host has the correct permissions to deploy the cluster. An account with incorrect permissions causes the installation process to fail with an error message that displays the missing permissions.

Procedure

1. In the directory that contains the installation program, initialize the cluster deployment by running the following command:

   ```
   $ ./openshift-install create cluster --dir <installation_directory> \ 1
       --log-level=info 2
   ```

   1 For **<installation_directory>**, specify the location of your customized **./install-config.yaml** file.

   2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

2. Optional: Remove or disable the **AdministratorAccess** policy from the IAM account that you used to install the cluster.

   NOTE

   The elevated permissions provided by the **AdministratorAccess** policy are required only during installation.

### Verification

When the cluster deployment completes successfully:

- The terminal displays directions for accessing your cluster, including a link to the web console and credentials for the **kubeadmin** user.
- Credential information also outputs to **<installation_directory>/.openshift_install.log**.

IMPORTANT

Do not delete the installation program or the files that the installation program creates. Both are required to delete the cluster.

### Example output

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```

IMPORTANT

- The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.
- It is recommended that you use Ignition config files within 12 hours after they are generated because the 24-hour certificate rotates from 16 to 22 hours after the cluster is installed. By using the Ignition config files within 12 hours, you can avoid installation failure if the certificate update runs during installation.

## 3.8.10. Logging in to the cluster by using the CLI

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

### Prerequisites

- You deployed an OpenShift Container Platform cluster.
- You installed the OpenShift CLI (**oc**).

### Procedure

1. Export the **kubeadmin** credentials by running the following command:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration by running the following command:

```
$ oc whoami
```

**Example output**

```
system:admin
```

## 3.8.11. Logging in to the cluster by using the web console

The **kubeadmin** user exists by default after an OpenShift Container Platform installation. You can log in to your cluster as the **kubeadmin** user by using the OpenShift Container Platform web console.

### Prerequisites

- You have access to the installation host.
- You completed a cluster installation and all cluster Operators are available.

### Procedure

1. Obtain the password for the **kubeadmin** user from the **kubeadmin-password** file on the installation host:

   ```
   $ cat <installation_directory>/auth/kubeadmin-password
   ```

   **NOTE**

   Alternatively, you can obtain the **kubeadmin** password from the **<installation_directory>/.openshift_install.log** log file on the installation host.

2. List the OpenShift Container Platform web console route:

   ```
   $ oc get routes -n openshift-console | grep 'console-openshift'
   ```

   **NOTE**

   Alternatively, you can obtain the OpenShift Container Platform route from the **<installation_directory>/.openshift_install.log** log file on the installation host.

   **Example output**

   ```
   console     console-openshift-console.apps.<cluster_name>.<base_domain>            console
   https   reencrypt/Redirect   None
   ```

3. Navigate to the route detailed in the output of the preceding command in a web browser and log in as the **kubeadmin** user.

### Additional resources

- See Accessing the web console for more details about accessing and understanding the OpenShift Container Platform web console.

## 3.8.12. Next steps

- Validating an installation.
- Customize your cluster.
- Remote health reporting.
- Remove cloud provider credentials.

## 3.9. INSTALLING A CLUSTER ON AWS INTO A SECRET OR TOP SECRET REGION

In OpenShift Container Platform version 4.19, you can install a cluster on Amazon Web Services (AWS) into the following secret regions:

- Secret Commercial Cloud Services (SC2S)
- Commercial Cloud Services (C2S)

To configure a cluster in either region, you change parameters in the **install config.yaml** file before you install the cluster.

WARNING

In OpenShift Container Platform 4.19, the installation program uses Cluster API instead of Terraform to provision cluster infrastructure during installations on AWS. Installing a cluster on AWS into a secret or top-secret region by using the Cluster API implementation has not been tested as of the release of OpenShift Container Platform 4.19. This document will be updated when installation into a secret region has been tested.

There is a known issue with Network Load Balancers' support for security groups in secret or top secret regions that causes installations in these regions to fail. For more information, see OCPBUGS-33311.

### 3.9.1. Prerequisites

- You reviewed details about the OpenShift Container Platform installation and update processes.
- You read the documentation on selecting a cluster installation method and preparing it for users.
- You configured an AWS account to host the cluster.

  IMPORTANT

  If you have an AWS profile stored on your computer, it must not use a temporary session token that you generated while using a multifactor authentication device. The cluster continues to use your current AWS credentials to create AWS resources for the entire life of the cluster, so you must use long-term credentials. To generate appropriate keys, see Managing Access Keys for IAM Users in the AWS documentation. You can supply the keys when you run the installation program.

- If you use a firewall, you configured it to allow the sites that your cluster requires access to.

### 3.9.2. AWS secret regions

The following AWS secret partitions are supported:

- **us-isob-east-1** (SC2S)
- **us-iso-east-1** (C2S)

NOTE

The maximum supported MTU in an AWS SC2S and C2S Regions is not the same as AWS commercial. For more information about configuring MTU during installation, see the *Cluster Network Operator configuration object* section in *Installing a cluster on AWS with network customizations*

### 3.9.3. Installation requirements

Red Hat does not publish a Red Hat Enterprise Linux CoreOS (RHCOS) Amzaon Machine Image for the AWS Secret and Top Secret Regions.

Before you can install the cluster, you must:

- Upload a custom RHCOS AMI.
- Manually create the installation configuration file (**install-config.yaml**).
- Specify the AWS region, and the accompanying custom AMI, in the installation configuration file.

You cannot use the OpenShift Container Platform installation program to create the installation configuration file. The installer does not list an AWS region without native support for an RHCOS AMI.

IMPORTANT

You must also define a custom CA certificate in the **additionalTrustBundle** field of the **install-config.yaml** file because the AWS API requires a custom CA trust bundle. To allow the installation program to access the AWS API, the CA certificates must also be defined on the machine that runs the installation program. You must add the CA bundle to the trust store on the machine, use the **AWS_CA_BUNDLE** environment variable, or define the CA bundle in the **ca_bundle** field of the AWS config file.

### 3.9.4. Private clusters

You can deploy a private OpenShift Container Platform cluster that does not expose external endpoints. Private clusters are accessible from only an internal network and are not visible to the internet.

NOTE

Public zones are not supported in Route 53 in an AWS Top Secret Region. Therefore, clusters must be private if they are deployed to an AWS Top Secret Region.

By default, OpenShift Container Platform is provisioned to use publicly-accessible DNS and endpoints. A private cluster sets the DNS, Ingress Controller, and API server to private when you deploy your cluster. This means that the cluster resources are only accessible from your internal network and are not visible to the internet.

IMPORTANT

If the cluster has any public subnets, load balancer services created by administrators might be publicly accessible. To ensure cluster security, verify that these services are explicitly annotated as private.

To deploy a private cluster, you must:

- Use existing networking that meets your requirements. Your cluster resources might be shared between other clusters on the network.
- Deploy from a machine that has access to:
  - The API services for the cloud to which you provision.
  - The hosts on the network that you provision.
  - The internet to obtain installation media.

You can use any machine that meets these access requirements and follows your company's guidelines. For example, this machine can be a bastion host on your cloud network or a machine that has access to the network through a VPN.

#### 3.9.4.1. Private clusters in AWS

To create a private cluster on Amazon Web Services (AWS), you must provide an existing private VPC and subnets to host the cluster. The installation program must also be able to resolve the DNS records that the cluster requires. The installation program configures the Ingress Operator and API server for access from only the private network.

The cluster still requires access to internet to access the AWS APIs.

The following items are not required or created when you install a private cluster:

- Public subnets
- Public load balancers, which support public ingress
- A public Route 53 zone that matches the **baseDomain** for the cluster

The installation program does use the **baseDomain** that you specify to create a private Route 53 zone and the required records for the cluster. The cluster is configured so that the Operators do not create public records for the cluster and all cluster machines are placed in the private subnets that you specify.

##### 3.9.4.1.1. Limitations

The ability to add public functionality to a private cluster is limited.

- You cannot make the Kubernetes API endpoints public after installation without taking additional actions, including creating public subnets in the VPC for each availability zone in use, creating a public load balancer, and configuring the control plane security groups to allow traffic from the internet on 6443 (Kubernetes API port).
- If you use a public Service type load balancer, you must tag a public subnet in each availability zone with **kubernetes.io/cluster/<cluster-infra-id>: shared** so that AWS can use them to create public load balancers.

### 3.9.5. About using a custom VPC

In OpenShift Container Platform 4.19, you can deploy a cluster into existing subnets in an existing Amazon Virtual Private Cloud (VPC) in Amazon Web Services (AWS). By deploying OpenShift Container Platform into an existing AWS VPC, you might be able to avoid limit constraints in new accounts or more easily abide by the operational constraints that your company's guidelines set. If you cannot obtain the infrastructure creation permissions that are required to create the VPC yourself, use this installation option.

Because the installation program cannot know what other components are also in your existing subnets, it cannot choose subnet CIDRs and so forth on your behalf. You must configure networking for the subnets that you install your cluster to yourself.

#### 3.9.5.1. Requirements for using your VPC

The installation program no longer creates the following components:

- Internet gateways
- NAT gateways
- Subnets
- Route tables
- VPCs
- VPC DHCP options
- VPC endpoints

**NOTE**

The installation program requires that you use the cloud-provided DNS server. Using a custom DNS server is not supported and causes the installation to fail.

If you use a custom VPC, you must correctly configure it and its subnets for the installation program and the cluster to use. See Create a VPC in the Amazon Web Services documentation for more information about AWS VPC console wizard configurations and creating and managing an AWS VPC.

The installation program cannot:

- Subdivide network ranges for the cluster to use.
- Set route tables for the subnets.
- Set VPC options like DHCP.

You must complete these tasks before you install the cluster. See VPC networking components and Route tables for your VPC for more information on configuring networking in an AWS VPC.

Your VPC must meet the following characteristics:

- The VPC must not use the **kubernetes.io/cluster/.*: owned**, **Name**, and **openshift.io/cluster** tags.
  The installation program modifies your subnets to add the **kubernetes.io/cluster/.*: shared**

tag, so your subnets must have at least one free tag slot available for it. See Tag Restrictions in the AWS documentation to confirm that the installation program can add a tag to each subnet that you specify. You cannot use a **Name** tag, because it overlaps with the EC2 **Name** field and the installation fails.

- If you want to extend your OpenShift Container Platform cluster into an AWS Outpost and have an existing Outpost subnet, the existing subnet must use the **kubernetes.io/cluster/unmanaged: true** tag. If you do not apply this tag, the installation might fail due to the Cloud Controller Manager creating a service load balancer in the Outpost subnet, which is an unsupported configuration.

- You must enable the **enableDnsSupport** and **enableDnsHostnames** attributes in your VPC, so that the cluster can use the Route 53 zones that are attached to the VPC to resolve cluster's internal DNS records. See DNS Support in Your VPC in the AWS documentation.
  If you prefer to use your own Route 53 hosted private zone, you must associate the existing hosted zone with your VPC prior to installing a cluster. You can define your hosted zone using the **platform.aws.hostedZone** and **platform.aws.hostedZoneRole** fields in the **install-config.yaml** file. You can use a private hosted zone from another account by sharing it with the account where you install the cluster. If you use a private hosted zone from another account, you must use the **Passthrough** or **Manual** credentials mode.

A cluster in an SC2S or C2S Region is unable to reach the public IP addresses for the EC2, ELB, and S3 endpoints. Depending on the level to which you want to restrict internet traffic during the installation, the following configuration options are available:

#### 3.9.5.1.1. Option 1: Create VPC endpoints

Create a VPC endpoint and attach it to the subnets that the clusters are using. Name the endpoints as follows:

SC2S

- **elasticloadbalancing.<aws_region>.sc2s.sgov.gov**
- **ec2.<aws_region>.sc2s.sgov.gov**
- **s3.<aws_region>.sc2s.sgov.gov**

C2S

- **elasticloadbalancing.<aws_region>.c2s.ic.gov**
- **ec2.<aws_region>.c2s.ic.gov**
- **s3.<aws_region>.c2s.ic.gov**

With this option, network traffic remains private between your VPC and the required AWS services.

#### 3.9.5.1.2. Option 2: Create a proxy without VPC endpoints

As part of the installation process, you can configure an HTTP or HTTPS proxy. With this option, internet traffic goes through the proxy to reach the required AWS services.

#### 3.9.5.1.3. Option 3: Create a proxy with VPC endpoints

As part of the installation process, you can configure an HTTP or HTTPS proxy with VPC endpoints. Create a VPC endpoint and attach it to the subnets that the clusters are using. Name the endpoints as follows:

SC2S

- **elasticloadbalancing.<aws_region>.sc2s.sgov.gov**
- **ec2.<aws_region>.sc2s.sgov.gov**
- **s3.<aws_region>.sc2s.sgov.gov**

C2S

- **elasticloadbalancing.<aws_region>.c2s.ic.gov**
- **ec2.<aws_region>.c2s.ic.gov**
- **s3.<aws_region>.c2s.ic.gov**

When configuring the proxy in the **install-config.yaml** file, add these endpoints to the **noProxy** field. With this option, the proxy prevents the cluster from accessing the internet directly. However, network traffic remains private between your VPC and the required AWS services.

### Required VPC components

You must provide a suitable VPC and subnets that allow communication to your machines.

| Component | AWS type | Description |
|---|---|---|
| VPC | • **AWS::EC2::VPC**<br>• **AWS::EC2::VPCEndpoint** | You must provide a public VPC for the cluster to use. The VPC uses an endpoint that references the route tables for each subnet to improve communication with the registry that is hosted in S3. |
| Public subnets | • **AWS::EC2::Subnet**<br>• **AWS::EC2::SubnetNetworkAclAssociation** | Your VPC must have public subnets for between 1 and 3 availability zones and associate them with appropriate Ingress rules. |

| Component | AWS type | Description |
|---|---|---|
| Internet gateway | • **AWS::EC2::InternetGateway**<br>• **AWS::EC2::VPCGatewayAttachment**<br>• **AWS::EC2::RouteTable**<br>• **AWS::EC2::Route**<br>• **AWS::EC2::SubnetRouteTableAssociation**<br>• **AWS::EC2::NatGateway**<br>• **AWS::EC2::EIP** | You must have a public internet gateway, with public routes, attached to the VPC. In the provided templates, each public subnet has a NAT gateway with an EIP address. These NAT gateways allow cluster resources, like private subnet instances, to reach the internet and are not required for some restricted network or proxy scenarios. |
| Network access control | • **AWS::EC2::NetworkAcl**<br>• **AWS::EC2::NetworkAclEntry** | You must allow the VPC to access the following ports:<br>Port / Reason<br>**80** / Inbound HTTP traffic<br>**443** / Inbound HTTPS traffic<br>**22** / Inbound SSH traffic<br>**1024 - 65535** / Inbound ephemeral traffic<br>**0 - 65535** / Outbound ephemeral traffic |
| Private subnets | • **AWS::EC2::Subnet**<br>• **AWS::EC2::RouteTable**<br>• **AWS::EC2::SubnetRouteTableAssociation** | Your VPC can have private subnets. The provided CloudFormation templates can create private subnets for between 1 and 3 availability zones. If you use private subnets, you must provide appropriate routes and tables for them. |

### 3.9.5.2. VPC validation

To ensure that the subnets that you provide are suitable, the installation program confirms the following data:

- All the subnets that you specify exist.
- You provide private subnets.
- The subnet CIDRs belong to the machine CIDR that you specified.
- You provide subnets for each availability zone. Each availability zone contains no more than one public and one private subnet. If you use a private cluster, provide only a private subnet for each availability zone. Otherwise, provide exactly one public and private subnet for each availability zone.
- You provide a public subnet for each private subnet availability zone. Machines are not provisioned in availability zones that you do not provide private subnets for.

If you destroy a cluster that uses an existing VPC, the VPC is not deleted. When you remove the OpenShift Container Platform cluster from a VPC, the **kubernetes.io/cluster/.*: shared** tag is removed from the subnets that it used.

#### 3.9.5.3. Division of permissions

Starting with OpenShift Container Platform 4.3, you do not need all of the permissions that are required for an installation program-provisioned infrastructure cluster to deploy a cluster. This change mimics the division of permissions that you might have at your company: some individuals can create different resource in your clouds than others. For example, you might be able to create application-specific items, like instances, buckets, and load balancers, but not networking-related components such as VPCs, subnets, or ingress rules.

The AWS credentials that you use when you create your cluster do not need the networking permissions that are required to make VPCs and core networking components within the VPC, such as subnets, routing tables, internet gateways, NAT, and VPN. You still need permission to make the application resources that the machines within the cluster require, such as ELBs, security groups, S3 buckets, and nodes.

#### 3.9.5.4. Isolation between clusters

If you deploy OpenShift Container Platform to an existing network, the isolation of cluster services is reduced in the following ways:

- You can install multiple OpenShift Container Platform clusters in the same VPC.
- ICMP ingress is allowed from the entire network.
- TCP 22 ingress (SSH) is allowed to the entire network.
- Control plane TCP 6443 ingress (Kubernetes API) is allowed to the entire network.
- Control plane TCP 22623 ingress (MCS) is allowed to the entire network.

#### 3.9.5.5. Optional: AWS security groups

By default, the installation program creates and attaches security groups to control plane and compute machines. The rules associated with the default security groups cannot be modified.

However, you can apply additional existing AWS security groups, which are associated with your existing VPC, to control plane and compute machines. Applying custom security groups can help you meet the security needs of your organization, in such cases where you need to control the incoming or outgoing traffic of these machines.

As part of the installation process, you apply custom security groups by modifying the **install-config.yaml** file before deploying the cluster.

For more information, see "Applying existing AWS security groups to the cluster".

### 3.9.6. Uploading a custom RHCOS AMI in AWS

If you are deploying to a custom Amazon Web Services (AWS) region, you must upload a custom Red Hat Enterprise Linux CoreOS (RHCOS) Amazon Machine Image (AMI) that belongs to that region.

#### Prerequisites

- You configured an AWS account.
- You created an Amazon S3 bucket with the required IAM service role.
- You uploaded your RHCOS VMDK file to Amazon S3. The RHCOS VMDK file must be the highest version that is less than or equal to the OpenShift Container Platform version you are installing.
- You downloaded the AWS CLI and installed it on your computer. See Install the AWS CLI Using the Bundled Installer.

#### Procedure

1. Export your AWS profile as an environment variable:

```
$ export AWS_PROFILE=<aws_profile> 1
```

2. Export the region to associate with your custom AMI as an environment variable:

```
$ export AWS_DEFAULT_REGION=<aws_region> 1
```

3. Export the version of RHCOS you uploaded to Amazon S3 as an environment variable:

```
$ export RHCOS_VERSION=<version> 1
```

1 1 1 The RHCOS VMDK version, like **4.19.0**.

4. Export the Amazon S3 bucket name as an environment variable:

```
$ export VMIMPORT_BUCKET_NAME=<s3_bucket_name>
```

5. Create the **containers.json** file and define your RHCOS VMDK file:

```
$ cat <<EOF > containers.json
{
  "Description": "rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64",
  "Format": "vmdk",
  "UserBucket": {
    "S3Bucket": "${VMIMPORT_BUCKET_NAME}",
      "S3Key": "rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64.vmdk"
   }
}
EOF
```

6. Import the RHCOS disk as an Amazon EBS snapshot:

```
$ aws ec2 import-snapshot --region ${AWS_DEFAULT_REGION} \
     --description "<description>" \ (1)
     --disk-container "file://<file_path>/containers.json" (2)
```

(1) The description of your RHCOS disk being imported, like **rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64**.

(2) The file path to the JSON file describing your RHCOS disk. The JSON file should contain your Amazon S3 bucket name and key.

7. Check the status of the image import:

```
$ watch -n 5 aws ec2 describe-import-snapshot-tasks --region ${AWS_DEFAULT_REGION}
```

**Example output**

```
{
    "ImportSnapshotTasks": [
        {
            "Description": "rhcos-4.7.0-x86_64-aws.x86_64",
            "ImportTaskId": "import-snap-fh6i8uil",
            "SnapshotTaskDetail": {
                "Description": "rhcos-4.7.0-x86_64-aws.x86_64",
                "DiskImageSize": 819056640.0,
                "Format": "VMDK",
                "SnapshotId": "snap-06331325870076318",
                "Status": "completed",
                "UserBucket": {
                    "S3Bucket": "external-images",
                    "S3Key": "rhcos-4.7.0-x86_64-aws.x86_64.vmdk"
                }
            }
        }
    ]
}
```

Copy the **SnapshotId** to register the image.

8. Create a custom RHCOS AMI from the RHCOS snapshot:

```
$ aws ec2 register-image \
   --region ${AWS_DEFAULT_REGION} \
   --architecture x86_64 \ (1)
   --description "rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64" \ (2)
   --ena-support \
   --name "rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64" \ (3)
```

```
    --virtualization-type hvm \
    --root-device-name '/dev/xvda' \
    --block-device-mappings 'DeviceName=/dev/xvda,Ebs=
{DeleteOnTermination=true,SnapshotId=<snapshot_ID>}' 4
```

1. The RHCOS VMDK architecture type, like **x86_64**, **aarch64**, **s390x**, or **ppc64le**.
2. The **Description** from the imported snapshot.
3. The name of the RHCOS AMI.
4. The **SnapshotID** from the imported snapshot.

To learn more about these APIs, see the AWS documentation for importing snapshots and creating EBS-backed AMIs.

## 3.9.7. Manually creating the installation configuration file

To customise your OpenShift Container Platform deployment and meet specific network requirements, manually create the installation configuration file. This ensures that the installation program uses your tailored settings rather than default values during the setup process.

### Prerequisites

- You have uploaded a custom RHCOS AMI.
- You have an SSH public key on your local machine for use with the installation program. You can use the key for SSH authentication onto your cluster nodes for debugging and disaster recovery.
- You have obtained the OpenShift Container Platform installation program and the pull secret for your cluster.

### Procedure

1. Create an installation directory to store your required installation assets in:

```
$ mkdir <installation_directory>
```

IMPORTANT

You must create a directory. Some installation assets, such as bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

2. Customize the provided sample **install-config.yaml** file template and save the file in the **<installation_directory>**.

NOTE

You must name this configuration file **install-config.yaml**.

3. Back up the **install-config.yaml** file so that you can use it to install many clusters.

IMPORTANT

Back up the **install-config.yaml** file now, because the installation process consumes the file in the next step.

Additional resources

- Installation configuration parameters for AWS

### 3.9.7.1. Tested instance types for AWS

The following Amazon Web Services (AWS) instance types have been tested with OpenShift Container Platform.

NOTE

Use the machine types included in the following charts for your AWS instances. If you use an instance type that is not listed in the chart, ensure that the instance size you use matches the minimum resource requirements that are listed in the section named "Minimum resource requirements for cluster installation".

Example 3.12. Machine types based on 64-bit x86 architecture for secret regions

- **c4.***
- **c5.***
- **i3.***
- **m4.***
- **m5.***
- **r4.***
- **r5.***
- **t3.***

### 3.9.7.2. Sample customized install-config.yaml file for AWS

You can customize the installation configuration file (**install-config.yaml**) to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.

IMPORTANT

This sample YAML file is provided for reference only. You must obtain your **install-config.yaml** file by using the installation program and modify it. For a full list and description of all installation configuration parameters, see *Installation configuration parameters for AWS*.

### Sample **install-config.yaml** file for AWS

```
apiVersion: v1 1
baseDomain: example.com
sshKey: ssh-ed25519 AAAA...
pullSecret: '{"auths": ...}'
metadata:
  name: example-cluster
controlPlane: 2
  name: master
  platform:
    aws:
      type: m6i.xlarge
  replicas: 3
compute: 3
- name: worker
  platform:
    aws:
      type: c5.4xlarge
  replicas: 3
networking: 4
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
platform: 5
  aws:
    region: us-west-2
```

1 Parameters at the first level of indentation apply to the cluster globally.

2 The **controlPlane** stanza applies to control plane machines.

3 The **compute** stanza applies to compute machines.

4 The **networking** stanza applies to the cluster networking configuration. If you do not provide networking values, the installation program provides default values.

5 The **platform** stanza applies to the infrastructure platform that hosts the cluster.

### Additional resources

- Installation configuration parameters for AWS

#### 3.9.7.3. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

## Prerequisites

- You have an existing **install-config.yaml** file.
- You reviewed the sites that your cluster requires access to and determined whether any of them need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. You added sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.

NOTE

The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.

For installations on Amazon Web Services (AWS), Google Cloud, Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

## Procedure

1. Edit your **install-config.yaml** file and add the proxy settings. For example:

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: ec2.<aws_region>.amazonaws.com,elasticloadbalancing.
<aws_region>.amazonaws.com,s3.<aws_region>.amazonaws.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1 A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**.

2 A proxy URL to use for creating HTTPS connections outside the cluster.

3 A comma-separated list of destination domain names, IP addresses, or other network CIDRs to exclude from proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use * to bypass the proxy for all destinations. If you have added the Amazon **EC2**,**Elastic Load Balancing**, and **S3** VPC endpoints to your VPC, you must add these endpoints to the **noProxy** field.

4 If provided, the installation program generates a config map that is named **user-ca-bundle** in the **openshift-config** namespace that contains one or more additional CA certificates

that are required for proxying HTTPS connections. The Cluster Network Operator then creates a **trusted-ca-bundle** config map that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this config map is referenced in the **trustedCA** field of the **Proxy** object. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle.

5 Optional: The policy to determine the configuration of the **Proxy** object to reference the **user-ca-bundle** config map in the **trustedCA** field. The allowed values are **Proxyonly** and **Always**. Use **Proxyonly** to reference the **user-ca-bundle** config map only when **http/https** proxy is configured. Use **Always** to always reference the **user-ca-bundle** config map. The default value is **Proxyonly**.

NOTE

The installation program does not support the proxy **readinessEndpoints** field.

NOTE

If the installer times out, restart and then complete the deployment by using the **wait-for** command of the installer. For example:

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. Save the file and reference it when installing OpenShift Container Platform.

The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster Proxy** object is still created, but it will have a nil **spec**.

NOTE

Only the **Proxy** object named **cluster** is supported, and no additional proxies can be created.

### 3.9.7.4. Applying existing AWS security groups to the cluster

Applying existing AWS security groups to your control plane and compute machines can help you meet the security needs of your organization, in such cases where you need to control the incoming or outgoing traffic of these machines.

**Prerequisites**

- You have created the security groups in AWS. For more information, see the AWS documentation about working with security groups.
- The security groups must be associated with the existing VPC that you are deploying the cluster to. The security groups cannot be associated with another VPC.
- You have an existing **install-config.yaml** file.

**Procedure**

1. In the **install-config.yaml** file, edit the **compute.platform.aws.additionalSecurityGroupIDs** parameter to specify one or more custom security groups for your compute machines.

2. Edit the **controlPlane.platform.aws.additionalSecurityGroupIDs** parameter to specify one or more custom security groups for your control plane machines.

3. Save the file and reference it when deploying the cluster.

**Sample install-config.yaml file that specifies custom security groups**

```
# ...
compute:
- hyperthreading: Enabled
  name: worker
  platform:
    aws:
      additionalSecurityGroupIDs:
        - sg-1 (1)
        - sg-2
  replicas: 3
controlPlane:
  hyperthreading: Enabled
  name: master
  platform:
    aws:
      additionalSecurityGroupIDs:
        - sg-3
        - sg-4
  replicas: 3
platform:
  aws:
    region: us-east-1
    subnets: (2)
      - subnet-1
      - subnet-2
      - subnet-3
```

(1) Specify the name of the security group as it appears in the Amazon EC2 console, including the **sg** prefix.

(2) Specify subnets for each availability zone that your cluster uses.

### 3.9.8. Alternatives to storing administrator-level secrets in the kube-system project

By default, administrator secrets are stored in the **kube-system** project. If you configured the **credentialsMode** parameter in the **install-config.yaml** file to **Manual**, you must use one of the following alternatives:

- To manage long-term cloud credentials manually, follow the procedure in Manually creating long-term credentials.

- To implement short-term credentials that are managed outside the cluster for individual components, follow the procedures in Configuring an AWS cluster to use short-term credentials.

### 3.9.8.1. Manually creating long-term credentials

The Cloud Credential Operator (CCO) can be put into manual mode prior to installation in environments where the cloud identity and access management (IAM) APIs are not reachable, or the administrator prefers not to store an administrator-level credential secret in the cluster **kube-system** namespace.

#### Procedure

1. If you did not set the **credentialsMode** parameter in the **install-config.yaml** configuration file to **Manual**, modify the value as shown:

   **Sample configuration file snippet**

   ```
   apiVersion: v1
   baseDomain: example.com
   credentialsMode: Manual
   # ...
   ```

2. If you have not previously created installation manifest files, do so by running the following command:

   ```
   $ openshift-install create manifests --dir <installation_directory>
   ```

   where **<installation_directory>** is the directory in which the installation program creates files.

3. Set a **$RELEASE_IMAGE** variable with the release image from your installation file by running the following command:

   ```
   $ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
   ```

4. Extract the list of **CredentialsRequest** custom resources (CRs) from the OpenShift Container Platform release image by running the following command:

   ```
   $ oc adm release extract \
     --from=$RELEASE_IMAGE \
     --credentials-requests \
     --included \ 1
     --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
     --to=<path_to_directory_for_credentials_requests> 3
   ```

   1 The **--included** parameter includes only the manifests that your specific cluster configuration requires.

   2 Specify the location of the **install-config.yaml** file.

   3 Specify the path to the directory where you want to store the **CredentialsRequest** objects. If the specified directory does not exist, this command creates it.

   This command creates a YAML file for each **CredentialsRequest** object.

   **Sample CredentialsRequest object**

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
    - effect: Allow
      action:
      - iam:GetUser
      - iam:GetUserPolicy
      - iam:ListAccessKeys
      resource: "*"
  ...
```

5. Create YAML files for secrets in the **openshift-install** manifests directory that you generated previously. The secrets must be stored using the namespace and secret name defined in the **spec.secretRef** for each **CredentialsRequest** object.

   **Sample CredentialsRequest object with secrets**

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
    - effect: Allow
      action:
      - s3:CreateBucket
      - s3:DeleteBucket
      resource: "*"
      ...
  secretRef:
    name: <component_secret>
    namespace: <component_namespace>
  ...
```

   **Sample Secret object**

```
apiVersion: v1
kind: Secret
metadata:
  name: <component_secret>
  namespace: <component_namespace>
data:
```

```
aws_access_key_id: <base64_encoded_aws_access_key_id>
aws_secret_access_key: <base64_encoded_aws_secret_access_key>
```

IMPORTANT

Before upgrading a cluster that uses manually maintained credentials, you must ensure that the CCO is in an upgradeable state.

#### 3.9.8.2. Configuring an AWS cluster to use short-term credentials

To install a cluster that is configured to use the AWS Security Token Service (STS), you must configure the CCO utility and create the required AWS resources for your cluster.

##### 3.9.8.2.1. Configuring the Cloud Credential Operator utility

To create and manage cloud credentials from outside of the cluster when the Cloud Credential Operator (CCO) is operating in manual mode, extract and prepare the CCO utility (**ccoctl**) binary.

NOTE

The **ccoctl** utility is a Linux binary that must run in a Linux environment.

**Prerequisites**

- You have access to an OpenShift Container Platform account with cluster administrator access.
- You have installed the OpenShift CLI (**oc**).
- You have created an AWS account for the **ccoctl** utility to use with the following permissions:
  **Required iam permissions**
  - **iam:CreateOpenIDConnectProvider**
  - **iam:CreateRole**
  - **iam:DeleteOpenIDConnectProvider**
  - **iam:DeleteRole**
  - **iam:DeleteRolePolicy**
  - **iam:GetOpenIDConnectProvider**
  - **iam:GetRole**
  - **iam:GetUser**
  - **iam:ListOpenIDConnectProviders**
  - **iam:ListRolePolicies**
  - **iam:ListRoles**
  - **iam:PutRolePolicy**

- **iam:TagOpenIDConnectProvider**
- **iam:TagRole**

Required **s3** permissions

- **s3:CreateBucket**
- **s3:DeleteBucket**
- **s3:DeleteObject**
- **s3:GetBucketAcl**
- **s3:GetBucketTagging**
- **s3:GetObject**
- **s3:GetObjectAcl**
- **s3:GetObjectTagging**
- **s3:ListBucket**
- **s3:PutBucketAcl**
- **s3:PutBucketPolicy**
- **s3:PutBucketPublicAccessBlock**
- **s3:PutBucketTagging**
- **s3:PutObject**
- **s3:PutObjectAcl**
- **s3:PutObjectTagging**

Required **cloudfront** permissions

- **cloudfront:ListCloudFrontOriginAccessIdentities**
- **cloudfront:ListDistributions**
- **cloudfront:ListTagsForResource**

- If you plan to store the OIDC configuration in a private S3 bucket that is accessed by the IAM identity provider through a public CloudFront distribution URL, the AWS account that runs the **ccoctl** utility requires the following additional permissions:
  - **cloudfront:CreateCloudFrontOriginAccessIdentity**
  - **cloudfront:CreateDistribution**
  - **cloudfront:DeleteCloudFrontOriginAccessIdentity**

- **cloudfront:DeleteDistribution**
- **cloudfront:GetCloudFrontOriginAccessIdentity**
- **cloudfront:GetCloudFrontOriginAccessIdentityConfig**
- **cloudfront:GetDistribution**
- **cloudfront:TagResource**
- **cloudfront:UpdateDistribution**

NOTE

These additional permissions support the use of the **--create-private-s3-bucket** option when processing credentials requests with the **ccoctl aws create-all** command.

Procedure

1. Set a variable for the OpenShift Container Platform release image by running the following command:

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. Obtain the CCO container image from the OpenShift Container Platform release image by running the following command:

```
$ CCO_IMAGE=$(oc adm release info --image-for='cloud-credential-operator'
$RELEASE_IMAGE -a ~/.pull-secret)
```

NOTE

Ensure that the architecture of the **$RELEASE_IMAGE** matches the architecture of the environment in which you will use the **ccoctl** tool.

3. Extract the **ccoctl** binary from the CCO container image within the OpenShift Container Platform release image by running the following command:

```
$ oc image extract $CCO_IMAGE \
  --file="/usr/bin/ccoctl.<rhel_version>" \ 1
  -a ~/.pull-secret
```

1 For **<rhel_version>**, specify the value that corresponds to the version of Red Hat Enterprise Linux (RHEL) that the host uses. If no value is specified, **ccoctl.rhel8** is used by default. The following values are valid:

- **rhel8**: Specify this value for hosts that use RHEL 8.
- **rhel9**: Specify this value for hosts that use RHEL 9.

NOTE

The **ccoctl** binary is created in the directory from where you executed the command and not in **/usr/bin/**. You must rename the directory or move the **ccoctl.<rhel_version>** binary to **ccoctl**.

4. Change the permissions to make **ccoctl** executable by running the following command:

```
$ chmod 775 ccoctl
```

#### Verification

- To verify that **ccoctl** is ready to use, display the help file. Use a relative file name when you run the command, for example:

```
$ ./ccoctl
```

##### Example output

```
OpenShift credentials provisioning tool

Usage:
  ccoctl [command]

Available Commands:
  aws          Manage credentials objects for AWS cloud
  azure        Manage credentials objects for Azure
  gcp          Manage credentials objects for Google cloud
  help         Help about any command
  ibmcloud     Manage credentials objects for {ibm-cloud-title}
  nutanix      Manage credentials objects for Nutanix

Flags:
  -h, --help   help for ccoctl

Use "ccoctl [command] --help" for more information about a command.
```

#### 3.9.8.2.2. Creating AWS resources with the Cloud Credential Operator utility

You have the following options when creating AWS resources:

- You can use the **ccoctl aws create-all** command to create the AWS resources automatically. This is the quickest way to create the resources. See Creating AWS resources with a single command.

- If you need to review the JSON files that the **ccoctl** tool creates before modifying AWS resources, or if the process the **ccoctl** tool uses to create AWS resources automatically does not meet the requirements of your organization, you can create the AWS resources individually. See Creating AWS resources individually .

##### 3.9.8.2.2.1. Creating AWS resources with a single command

If the process the **ccoctl** tool uses to create AWS resources automatically meets the requirements of your organization, you can use the **ccoctl aws create-all** command to automate the creation of AWS resources.

Otherwise, you can create the AWS resources individually. For more information, see "Creating AWS resources individually".

NOTE

By default, **ccoctl** creates objects in the directory in which the commands are run. To create the objects in a different directory, use the **--output-dir** flag. This procedure uses **<path_to_ccoctl_output_dir>** to refer to this directory.

### Prerequisites

You must have:

- Extracted and prepared the **ccoctl** binary.

### Procedure

1. Set a **$RELEASE_IMAGE** variable with the release image from your installation file by running the following command:

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. Extract the list of **CredentialsRequest** objects from the OpenShift Container Platform release image by running the following command:

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

1 The **--included** parameter includes only the manifests that your specific cluster configuration requires.

2 Specify the location of the **install-config.yaml** file.

3 Specify the path to the directory where you want to store the **CredentialsRequest** objects. If the specified directory does not exist, this command creates it.

NOTE

This command might take a few moments to run.

3. Use the **ccoctl** tool to process all **CredentialsRequest** objects by running the following command:

```
$ ccoctl aws create-all \
--name=<name> \ 1
--region=<aws_region> \ 2
--credentials-requests-dir=<path_to_credentials_requests_directory> \ 3
--output-dir=<path_to_ccoctl_output_dir> \ 4
--create-private-s3-bucket 5
```

1. Specify the name used to tag any cloud resources that are created for tracking.
2. Specify the AWS region in which cloud resources will be created.
3. Specify the directory containing the files for the component **CredentialsRequest** objects.
4. Optional: Specify the directory in which you want the **ccoctl** utility to create objects. By default, the utility creates objects in the directory in which the commands are run.
5. Optional: By default, the **ccoctl** utility stores the OpenID Connect (OIDC) configuration files in a public S3 bucket and uses the S3 URL as the public OIDC endpoint. To store the OIDC configuration in a private S3 bucket that is accessed by the IAM identity provider through a public CloudFront distribution URL instead, use the **--create-private-s3-bucket** parameter.

NOTE

If your cluster uses Technology Preview features that are enabled by the **TechPreviewNoUpgrade** feature set, you must include the **--enable-tech-preview** parameter.

Verification

- To verify that the OpenShift Container Platform secrets are created, list the files in the **<path_to_ccoctl_output_dir>/manifests** directory:

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

**Example output**

```
cluster-authentication-02-config.yaml
openshift-cloud-credential-operator-cloud-credential-operator-iam-ro-creds-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capa-manager-bootstrap-credentials-credentials.yaml
openshift-cluster-csi-drivers-ebs-cloud-credentials-credentials.yaml
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-aws-cloud-credentials-credentials.yaml
```

You can verify that the IAM roles are created by querying AWS. For more information, refer to AWS documentation on listing IAM roles.

###### 3.9.8.2.2.2. Creating AWS resources individually

You can use the **ccoctl** tool to create AWS resources individually. This option might be useful for an organization that shares the responsibility for creating these resources among different users or departments.

Otherwise, you can use the **ccoctl aws create-all** command to create the AWS resources automatically. For more information, see "Creating AWS resources with a single command".

NOTE

By default, **ccoctl** creates objects in the directory in which the commands are run. To create the objects in a different directory, use the **--output-dir** flag. This procedure uses **<path_to_ccoctl_output_dir>** to refer to this directory.

Some **ccoctl** commands make AWS API calls to create or modify AWS resources. You can use the **--dry-run** flag to avoid making API calls. Using this flag creates JSON files on the local file system instead. You can review and modify the JSON files and then apply them with the AWS CLI tool using the **--cli-input-json** parameters.

**Prerequisites**

- Extract and prepare the **ccoctl** binary.

**Procedure**

1. Generate the public and private RSA key files that are used to set up the OpenID Connect provider for the cluster by running the following command:

```
$ ccoctl aws create-key-pair
```

**Example output**

```
2021/04/13 11:01:02 Generating RSA keypair
2021/04/13 11:01:03 Writing private key to /<path_to_ccoctl_output_dir>/serviceaccount-
signer.private
2021/04/13 11:01:03 Writing public key to /<path_to_ccoctl_output_dir>/serviceaccount-
signer.public
2021/04/13 11:01:03 Copying signing key for use by installer
```

where **serviceaccount-signer.private** and **serviceaccount-signer.public** are the generated key files.

This command also creates a private key that the cluster requires during installation in **/<path_to_ccoctl_output_dir>/tls/bound-service-account-signing-key.key**.

2. Create an OpenID Connect identity provider and S3 bucket on AWS by running the following command:

```
$ ccoctl aws create-identity-provider \
  --name=<name> \ (1)
  --region=<aws_region> \ (2)
  --public-key-file=<path_to_ccoctl_output_dir>/serviceaccount-signer.public (3)
```

(1) **<name>** is the name used to tag any cloud resources that are created for tracking.

2 **<aws-region>** is the AWS region in which cloud resources will be created.

3 **<path_to_ccoctl_output_dir>** is the path to the public key file that the **ccoctl aws create-key-pair** command generated.

### Example output

```
2021/04/13 11:16:09 Bucket <name>-oidc created
2021/04/13 11:16:10 OpenID Connect discovery document in the S3 bucket <name>-oidc at
.well-known/openid-configuration updated
2021/04/13 11:16:10 Reading public key
2021/04/13 11:16:10 JSON web key set (JWKS) in the S3 bucket <name>-oidc at keys.json
updated
2021/04/13 11:16:18 Identity Provider created with ARN: arn:aws:iam::
<aws_account_id>:oidc-provider/<name>-oidc.s3.<aws_region>.amazonaws.com
```

where **openid-configuration** is a discovery document and **keys.json** is a JSON web key set file.

This command also creates a YAML configuration file in **/<path_to_ccoctl_output_dir>/manifests/cluster-authentication-02-config.yaml**. This file sets the issuer URL field for the service account tokens that the cluster generates, so that the AWS IAM identity provider trusts the tokens.

3. Create IAM roles for each component in the cluster:

   a. Set a **$RELEASE_IMAGE** variable with the release image from your installation file by running the following command:

   ```
   $ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
   ```

   b. Extract the list of **CredentialsRequest** objects from the OpenShift Container Platform release image:

   ```
   $ oc adm release extract \
     --from=$RELEASE_IMAGE \
     --credentials-requests \
     --included \ 1
     --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \
   2
     --to=<path_to_directory_for_credentials_requests> 3
   ```

   1 The **--included** parameter includes only the manifests that your specific cluster configuration requires.

   2 Specify the location of the **install-config.yaml** file.

   3 Specify the path to the directory where you want to store the **CredentialsRequest** objects. If the specified directory does not exist, this command creates it.

   c. Use the **ccoctl** tool to process all **CredentialsRequest** objects by running the following command:

```
$ ccoctl aws create-iam-roles \
  --name=<name> \
  --region=<aws_region> \
  --credentials-requests-dir=<path_to_credentials_requests_directory> \
  --identity-provider-arn=arn:aws:iam::<aws_account_id>:oidc-provider/<name>-oidc.s3.
<aws_region>.amazonaws.com
```

NOTE

For AWS environments that use alternative IAM API endpoints, such as GovCloud, you must also specify your region with the **--region** parameter.

If your cluster uses Technology Preview features that are enabled by the **TechPreviewNoUpgrade** feature set, you must include the **--enable-tech-preview** parameter.

For each **CredentialsRequest** object, **ccoctl** creates an IAM role with a trust policy that is tied to the specified OIDC identity provider, and a permissions policy as defined in each **CredentialsRequest** object from the OpenShift Container Platform release image.

#### Verification

- To verify that the OpenShift Container Platform secrets are created, list the files in the **<path_to_ccoctl_output_dir>/manifests** directory:

  ```
  $ ls <path_to_ccoctl_output_dir>/manifests
  ```

  **Example output**

  ```
  cluster-authentication-02-config.yaml
  openshift-cloud-credential-operator-cloud-credential-operator-iam-ro-creds-credentials.yaml
  openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
  openshift-cluster-api-capa-manager-bootstrap-credentials-credentials.yaml
  openshift-cluster-csi-drivers-ebs-cloud-credentials-credentials.yaml
  openshift-image-registry-installer-cloud-credentials-credentials.yaml
  openshift-ingress-operator-cloud-credentials-credentials.yaml
  openshift-machine-api-aws-cloud-credentials-credentials.yaml
  ```

  You can verify that the IAM roles are created by querying AWS. For more information, refer to AWS documentation on listing IAM roles.

### 3.9.8.2.3. Incorporating the Cloud Credential Operator utility manifests

To implement short-term security credentials managed outside the cluster for individual components, you must move the manifest files that the Cloud Credential Operator utility (**ccoctl**) created to the correct directories for the installation program.

#### Prerequisites

- You have configured an account with the cloud platform that hosts your cluster.
- You have configured the Cloud Credential Operator utility (**ccoctl**).

- You have created the cloud provider resources that are required for your cluster with the **ccoctl** utility.

**Procedure**

1. If you did not set the **credentialsMode** parameter in the **install-config.yaml** configuration file to **Manual**, modify the value as shown:

   **Sample configuration file snippet**

   ```
   apiVersion: v1
   baseDomain: example.com
   credentialsMode: Manual
   # ...
   ```

2. If you have not previously created installation manifest files, do so by running the following command:

   ```
   $ openshift-install create manifests --dir <installation_directory>
   ```

   where **<installation_directory>** is the directory in which the installation program creates files.

3. Copy the manifests that the **ccoctl** utility generated to the **manifests** directory that the installation program created by running the following command:

   ```
   $ cp /<path_to_ccoctl_output_dir>/manifests/* ./manifests/
   ```

4. Copy the **tls** directory that contains the private key to the installation directory:

   ```
   $ cp -a /<path_to_ccoctl_output_dir>/tls .
   ```

### 3.9.9. Deploying the cluster

You can install OpenShift Container Platform on a compatible cloud platform.

**IMPORTANT**

You can run the **create cluster** command of the installation program only once, during initial installation.

**Prerequisites**

- You have configured an account with the cloud platform that hosts your cluster.
- You have the OpenShift Container Platform installation program and the pull secret for your cluster.
- You have verified that the cloud provider account on your host has the correct permissions to deploy the cluster. An account with incorrect permissions causes the installation process to fail with an error message that displays the missing permissions.

**Procedure**

1. In the directory that contains the installation program, initialize the cluster deployment by running the following command:

   ```
   $ ./openshift-install create cluster --dir <installation_directory> \ 1
       --log-level=info 2
   ```

   1 For **<installation_directory>**, specify the location of your customized **./install-config.yaml** file.

   2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

2. Optional: Remove or disable the **AdministratorAccess** policy from the IAM account that you used to install the cluster.

   NOTE

   The elevated permissions provided by the **AdministratorAccess** policy are required only during installation.

## Verification

When the cluster deployment completes successfully:

- The terminal displays directions for accessing your cluster, including a link to the web console and credentials for the **kubeadmin** user.
- Credential information also outputs to **<installation_directory>/.openshift_install.log**.

IMPORTANT

Do not delete the installation program or the files that the installation program creates. Both are required to delete the cluster.

## Example output

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```

IMPORTANT

- The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

- It is recommended that you use Ignition config files within 12 hours after they are generated because the 24-hour certificate rotates from 16 to 22 hours after the cluster is installed. By using the Ignition config files within 12 hours, you can avoid installation failure if the certificate update runs during installation.

### 3.9.10. Logging in to the cluster by using the CLI

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

#### Prerequisites

- You deployed an OpenShift Container Platform cluster.

- You installed the OpenShift CLI (**oc**).

#### Procedure

1. Export the **kubeadmin** credentials by running the following command:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration by running the following command:

```
$ oc whoami
```

**Example output**

```
system:admin
```

### 3.9.11. Logging in to the cluster by using the web console

The **kubeadmin** user exists by default after an OpenShift Container Platform installation. You can log in to your cluster as the **kubeadmin** user by using the OpenShift Container Platform web console.

### Prerequisites

- You have access to the installation host.
- You completed a cluster installation and all cluster Operators are available.

### Procedure

1. Obtain the password for the **kubeadmin** user from the **kubeadmin-password** file on the installation host:

```
$ cat <installation_directory>/auth/kubeadmin-password
```

NOTE

Alternatively, you can obtain the **kubeadmin** password from the **<installation_directory>/.openshift_install.log** log file on the installation host.

2. List the OpenShift Container Platform web console route:

```
$ oc get routes -n openshift-console | grep 'console-openshift'
```

NOTE

Alternatively, you can obtain the OpenShift Container Platform route from the **<installation_directory>/.openshift_install.log** log file on the installation host.

#### Example output

```
console     console-openshift-console.apps.<cluster_name>.<base_domain>            console
https   reencrypt/Redirect   None
```

3. Navigate to the route detailed in the output of the preceding command in a web browser and log in as the **kubeadmin** user.

### Additional resources

- Accessing the web console

### 3.9.12. Next steps

- Validating an installation.
- Customize your cluster.
- Remote health reporting.
- Remove cloud provider credentials.

## 3.10. INSTALLING A CLUSTER ON AWS CHINA

In OpenShift Container Platform version 4.19, you can install a cluster to the following Amazon Web Services (AWS) China regions:

- **cn-north-1** (Beijing)
- **cn-northwest-1** (Ningxia)

### 3.10.1. Prerequisites

- You have an Internet Content Provider (ICP) license.
- You reviewed details about the OpenShift Container Platform installation and update processes.
- You read the documentation on selecting a cluster installation method and preparing it for users.
- You configured an AWS account to host the cluster.
- If you use a firewall, you configured it to allow the sites that your cluster requires access to.

IMPORTANT

If you have an AWS profile stored on your computer, it must not use a temporary session token that you generated while using a multi-factor authentication device. The cluster continues to use your current AWS credentials to create AWS resources for the entire life of the cluster, so you must use long-term credentials. To generate appropriate keys, see Managing Access Keys for IAM Users in the AWS documentation. You can supply the keys when you run the installation program.

### 3.10.2. Installation requirements

Red Hat does not publish a Red Hat Enterprise Linux CoreOS (RHCOS) Amazon Machine Image (AMI) for the AWS China regions.

Before you can install the cluster, you must:

- Upload a custom RHCOS AMI.
- Manually create the installation configuration file (**install-config.yaml**).
- Specify the AWS region, and the accompanying custom AMI, in the installation configuration file.

You cannot use the OpenShift Container Platform installation program to create the installation configuration file. The installer does not list an AWS region without native support for an RHCOS AMI.

### 3.10.3. Private clusters

You can deploy a private OpenShift Container Platform cluster that does not expose external endpoints. Private clusters are accessible from only an internal network and are not visible to the internet.

By default, OpenShift Container Platform is provisioned to use publicly-accessible DNS and endpoints. A private cluster sets the DNS, Ingress Controller, and API server to private when you deploy your cluster. This means that the cluster resources are only accessible from your internal network and are not visible to the internet.

IMPORTANT

If the cluster has any public subnets, load balancer services created by administrators might be publicly accessible. To ensure cluster security, verify that these services are explicitly annotated as private.

To deploy a private cluster, you must:

- Use existing networking that meets your requirements. Your cluster resources might be shared between other clusters on the network.
- Deploy from a machine that has access to:
  - The API services for the cloud to which you provision.
  - The hosts on the network that you provision.
  - The internet to obtain installation media.

You can use any machine that meets these access requirements and follows your company's guidelines. For example, this machine can be a bastion host on your cloud network.

NOTE

AWS China does not support a VPN connection between the VPC and your network. For more information about the Amazon VPC service in the Beijing and Ningxia regions, see Amazon Virtual Private Cloud in the AWS China documentation.

### 3.10.3.1. Private clusters in AWS

To create a private cluster on Amazon Web Services (AWS), you must provide an existing private VPC and subnets to host the cluster. The installation program must also be able to resolve the DNS records that the cluster requires. The installation program configures the Ingress Operator and API server for access from only the private network.

The cluster still requires access to internet to access the AWS APIs.

The following items are not required or created when you install a private cluster:

- Public subnets
- Public load balancers, which support public ingress
- A public Route 53 zone that matches the **baseDomain** for the cluster

The installation program does use the **baseDomain** that you specify to create a private Route 53 zone and the required records for the cluster. The cluster is configured so that the Operators do not create public records for the cluster and all cluster machines are placed in the private subnets that you specify.

#### 3.10.3.1.1. Limitations

The ability to add public functionality to a private cluster is limited.

- You cannot make the Kubernetes API endpoints public after installation without taking additional actions, including creating public subnets in the VPC for each availability zone in use, creating a public load balancer, and configuring the control plane security groups to allow traffic from the internet on 6443 (Kubernetes API port).
- If you use a public Service type load balancer, you must tag a public subnet in each availability zone with **kubernetes.io/cluster/<cluster-infra-id>: shared** so that AWS can use them to create public load balancers.

### 3.10.4. About using a custom VPC

In OpenShift Container Platform 4.19, you can deploy a cluster into existing subnets in an existing Amazon Virtual Private Cloud (VPC) in Amazon Web Services (AWS). By deploying OpenShift Container Platform into an existing AWS VPC, you might be able to avoid limit constraints in new accounts or more easily abide by the operational constraints that your company's guidelines set. If you cannot obtain the infrastructure creation permissions that are required to create the VPC yourself, use this installation option.

Because the installation program cannot know what other components are also in your existing subnets, it cannot choose subnet CIDRs and so forth on your behalf. You must configure networking for the subnets that you install your cluster to yourself.

#### 3.10.4.1. Requirements for using your VPC

The installation program no longer creates the following components:

- Internet gateways
- NAT gateways
- Subnets
- Route tables
- VPCs
- VPC DHCP options
- VPC endpoints

**NOTE**

The installation program requires that you use the cloud-provided DNS server. Using a custom DNS server is not supported and causes the installation to fail.

If you use a custom VPC, you must correctly configure it and its subnets for the installation program and the cluster to use. See Create a VPC in the Amazon Web Services documentation for more information about AWS VPC console wizard configurations and creating and managing an AWS VPC.

The installation program cannot:

- Subdivide network ranges for the cluster to use.
- Set route tables for the subnets.
- Set VPC options like DHCP.

You must complete these tasks before you install the cluster. See VPC networking components and Route tables for your VPC for more information on configuring networking in an AWS VPC.

Your VPC must meet the following characteristics:

- The VPC must not use the **kubernetes.io/cluster/.*: owned**, **Name**, and **openshift.io/cluster** tags.
  The installation program modifies your subnets to add the **kubernetes.io/cluster/.*: shared** tag, so your subnets must have at least one free tag slot available for it. See Tag Restrictions in the AWS documentation to confirm that the installation program can add a tag to each subnet that you specify. You cannot use a **Name** tag, because it overlaps with the EC2 **Name** field and the installation fails.

- If you want to extend your OpenShift Container Platform cluster into an AWS Outpost and have an existing Outpost subnet, the existing subnet must use the **kubernetes.io/cluster/unmanaged: true** tag. If you do not apply this tag, the installation might fail due to the Cloud Controller Manager creating a service load balancer in the Outpost subnet, which is an unsupported configuration.

- You must enable the **enableDnsSupport** and **enableDnsHostnames** attributes in your VPC, so that the cluster can use the Route 53 zones that are attached to the VPC to resolve cluster's internal DNS records. See DNS Support in Your VPC in the AWS documentation.
  If you prefer to use your own Route 53 hosted private zone, you must associate the existing hosted zone with your VPC prior to installing a cluster. You can define your hosted zone using the **platform.aws.hostedZone** and **platform.aws.hostedZoneRole** fields in the **install-config.yaml** file. You can use a private hosted zone from another account by sharing it with the account where you install the cluster. If you use a private hosted zone from another account, you must use the **Passthrough** or **Manual** credentials mode.

If you are working in a disconnected environment, you are unable to reach the public IP addresses for EC2, ELB, and S3 endpoints. Depending on the level to which you want to restrict internet traffic during the installation, the following configuration options are available:

###### 3.10.4.1.1. Option 1: Create VPC endpoints

Create a VPC endpoint and attach it to the subnets that the clusters are using. Name the endpoints as follows:

- **ec2.<aws_region>.amazonaws.com.cn**
- **elasticloadbalancing.<aws_region>.amazonaws.com**
- **s3.<aws_region>.amazonaws.com**

With this option, network traffic remains private between your VPC and the required AWS services.

###### 3.10.4.1.2. Option 2: Create a proxy without VPC endpoints

As part of the installation process, you can configure an HTTP or HTTPS proxy. With this option, internet traffic goes through the proxy to reach the required AWS services.

###### 3.10.4.1.3. Option 3: Create a proxy with VPC endpoints

As part of the installation process, you can configure an HTTP or HTTPS proxy with VPC endpoints. Create a VPC endpoint and attach it to the subnets that the clusters are using. Name the endpoints as follows:

- **ec2.<aws_region>.amazonaws.com.cn**
- **elasticloadbalancing.<aws_region>.amazonaws.com**
- **s3.<aws_region>.amazonaws.com**

When configuring the proxy in the **install-config.yaml** file, add these endpoints to the **noProxy** field. With this option, the proxy prevents the cluster from accessing the internet directly. However, network traffic remains private between your VPC and the required AWS services.

### Required VPC components

You must provide a suitable VPC and subnets that allow communication to your machines.

| Component | AWS type | Description |
|---|---|---|
| VPC | • **AWS::EC2::VPC**<br>• **AWS::EC2::VPCEndpoint** | You must provide a public VPC for the cluster to use. The VPC uses an endpoint that references the route tables for each subnet to improve communication with the registry that is hosted in S3. |
| Public subnets | • **AWS::EC2::Subnet**<br>• **AWS::EC2::SubnetNetworkAclAssociation** | Your VPC must have public subnets for between 1 and 3 availability zones and associate them with appropriate Ingress rules. |
| Internet gateway | • **AWS::EC2::InternetGateway**<br>• **AWS::EC2::VPCGatewayAttachment**<br>• **AWS::EC2::RouteTable**<br>• **AWS::EC2::Route**<br>• **AWS::EC2::SubnetRouteTableAssociation**<br>• **AWS::EC2::NatGateway**<br>• **AWS::EC2::EIP** | You must have a public internet gateway, with public routes, attached to the VPC. In the provided templates, each public subnet has a NAT gateway with an EIP address. These NAT gateways allow cluster resources, like private subnet instances, to reach the internet and are not required for some restricted network or proxy scenarios. |
| Network access control | • **AWS::EC2::NetworkAcl**<br>• **AWS::EC2::NetworkAclEntry** | You must allow the VPC to access the following ports:<br><br>**Port** \| **Reason** |

| Compone nt | AWS type | Description | |
|---|---|---|---|
| | | **80** | Inbound HTTP traffic |
| | | **443** | Inbound HTTPS traffic |
| | | **22** | Inbound SSH traffic |
| | | **1024 - 65535** | Inbound ephemeral traffic |
| | | **0 - 65535** | Outbound ephemeral traffic |
| Private subnets | • **AWS::EC2::Subnet**<br>• **AWS::EC2::RouteTable**<br>• **AWS::EC2::SubnetRouteTableAssociation** | Your VPC can have private subnets. The provided CloudFormation templates can create private subnets for between 1 and 3 availability zones. If you use private subnets, you must provide appropriate routes and tables for them. | |

### 3.10.4.2. VPC validation

To ensure that the subnets that you provide are suitable, the installation program confirms the following data:

- All the subnets that you specify exist.
- You provide private subnets.
- The subnet CIDRs belong to the machine CIDR that you specified.
- You provide subnets for each availability zone. Each availability zone contains no more than one public and one private subnet. If you use a private cluster, provide only a private subnet for each availability zone. Otherwise, provide exactly one public and private subnet for each availability zone.
- You provide a public subnet for each private subnet availability zone. Machines are not provisioned in availability zones that you do not provide private subnets for.

If you destroy a cluster that uses an existing VPC, the VPC is not deleted. When you remove the OpenShift Container Platform cluster from a VPC, the **kubernetes.io/cluster/.*: shared** tag is removed from the subnets that it used.

### 3.10.4.3. Division of permissions

Starting with OpenShift Container Platform 4.3, you do not need all of the permissions that are required for an installation program-provisioned infrastructure cluster to deploy a cluster. This change mimics the division of permissions that you might have at your company: some individuals can create different resource in your clouds than others. For example, you might be able to create application-specific items, like instances, buckets, and load balancers, but not networking-related components such as VPCs, subnets, or ingress rules.

The AWS credentials that you use when you create your cluster do not need the networking permissions that are required to make VPCs and core networking components within the VPC, such as subnets, routing tables, internet gateways, NAT, and VPN. You still need permission to make the application resources that the machines within the cluster require, such as ELBs, security groups, S3 buckets, and nodes.

#### 3.10.4.4. Isolation between clusters

If you deploy OpenShift Container Platform to an existing network, the isolation of cluster services is reduced in the following ways:

- You can install multiple OpenShift Container Platform clusters in the same VPC.
- ICMP ingress is allowed from the entire network.
- TCP 22 ingress (SSH) is allowed to the entire network.
- Control plane TCP 6443 ingress (Kubernetes API) is allowed to the entire network.
- Control plane TCP 22623 ingress (MCS) is allowed to the entire network.

#### 3.10.4.5. Optional: AWS security groups

By default, the installation program creates and attaches security groups to control plane and compute machines. The rules associated with the default security groups cannot be modified.

However, you can apply additional existing AWS security groups, which are associated with your existing VPC, to control plane and compute machines. Applying custom security groups can help you meet the security needs of your organization, in such cases where you need to control the incoming or outgoing traffic of these machines.

As part of the installation process, you apply custom security groups by modifying the **install-config.yaml** file before deploying the cluster.

For more information, see "Applying existing AWS security groups to the cluster".

### 3.10.5. Uploading a custom RHCOS AMI in AWS

If you are deploying to a custom Amazon Web Services (AWS) region, you must upload a custom Red Hat Enterprise Linux CoreOS (RHCOS) Amazon Machine Image (AMI) that belongs to that region.

**Prerequisites**

- You configured an AWS account.
- You created an Amazon S3 bucket with the required IAM service role.
- You uploaded your RHCOS VMDK file to Amazon S3. The RHCOS VMDK file must be the highest version that is less than or equal to the OpenShift Container Platform version you are installing.

- You downloaded the AWS CLI and installed it on your computer. See Install the AWS CLI Using the Bundled Installer.

**Procedure**

1. Export your AWS profile as an environment variable:

```
$ export AWS_PROFILE=<aws_profile> 1
```

1 The AWS profile name that holds your AWS credentials, like **beijingadmin**.

2. Export the region to associate with your custom AMI as an environment variable:

```
$ export AWS_DEFAULT_REGION=<aws_region> 1
```

1 The AWS region, like **cn-north-1**.

3. Export the version of RHCOS you uploaded to Amazon S3 as an environment variable:

```
$ export RHCOS_VERSION=<version> 1
```

1 The RHCOS VMDK version, like **4.19.0**.

4. Export the Amazon S3 bucket name as an environment variable:

```
$ export VMIMPORT_BUCKET_NAME=<s3_bucket_name>
```

5. Create the **containers.json** file and define your RHCOS VMDK file:

```
$ cat <<EOF > containers.json
{
  "Description": "rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64",
  "Format": "vmdk",
  "UserBucket": {
    "S3Bucket": "${VMIMPORT_BUCKET_NAME}",
    "S3Key": "rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64.vmdk"
  }
}
EOF
```

6. Import the RHCOS disk as an Amazon EBS snapshot:

```
$ aws ec2 import-snapshot --region ${AWS_DEFAULT_REGION} \
     --description "<description>" \ 1
     --disk-container "file://<file_path>/containers.json" 2
```

1 The description of your RHCOS disk being imported, like **rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64**.

2 The file path to the JSON file describing your RHCOS disk. The JSON file should contain your Amazon S3 bucket name and key.

your Amazon S3 bucket name and key.

7. Check the status of the image import:

```
$ watch -n 5 aws ec2 describe-import-snapshot-tasks --region ${AWS_DEFAULT_REGION}
```

**Example output**

```
{
    "ImportSnapshotTasks": [
        {
            "Description": "rhcos-4.7.0-x86_64-aws.x86_64",
            "ImportTaskId": "import-snap-fh6i8uil",
            "SnapshotTaskDetail": {
                "Description": "rhcos-4.7.0-x86_64-aws.x86_64",
                "DiskImageSize": 819056640.0,
                "Format": "VMDK",
                "SnapshotId": "snap-06331325870076318",
                "Status": "completed",
                "UserBucket": {
                    "S3Bucket": "external-images",
                    "S3Key": "rhcos-4.7.0-x86_64-aws.x86_64.vmdk"
                }
            }
        }
    ]
}
```

Copy the **SnapshotId** to register the image.

8. Create a custom RHCOS AMI from the RHCOS snapshot:

```
$ aws ec2 register-image \
   --region ${AWS_DEFAULT_REGION} \
   --architecture x86_64 \ 1
   --description "rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64" \ 2
   --ena-support \
   --name "rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64" \ 3
   --virtualization-type hvm \
   --root-device-name '/dev/xvda' \
   --block-device-mappings 'DeviceName=/dev/xvda,Ebs=
{DeleteOnTermination=true,SnapshotId=<snapshot_ID>}' 4
```

1 The RHCOS VMDK architecture type, like **x86_64**, **aarch64**, **s390x**, or **ppc64le**.

2 The **Description** from the imported snapshot.

3 The name of the RHCOS AMI.

4 The **SnapshotID** from the imported snapshot.

To learn more about these APIs, see the AWS documentation for importing snapshots and creating EBS-backed AMIs.

### 3.10.6. Manually creating the installation configuration file

To customise your OpenShift Container Platform deployment and meet specific network requirements, manually create the installation configuration file. This ensures that the installation program uses your tailored settings rather than default values during the setup process.

#### Prerequisites

- You have uploaded a custom RHCOS AMI.
- You have an SSH public key on your local machine for use with the installation program. You can use the key for SSH authentication onto your cluster nodes for debugging and disaster recovery.
- You have obtained the OpenShift Container Platform installation program and the pull secret for your cluster.

#### Procedure

1. Create an installation directory to store your required installation assets in:

```
$ mkdir <installation_directory>
```

IMPORTANT

You must create a directory. Some installation assets, such as bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

2. Customize the provided sample **install-config.yaml** file template and save the file in the **<installation_directory>**.

NOTE

You must name this configuration file **install-config.yaml**.

3. Back up the **install-config.yaml** file so that you can use it to install many clusters.

IMPORTANT

Back up the **install-config.yaml** file now, because the installation process consumes the file in the next step.

#### Additional resources

- Installation configuration parameters for AWS

#### 3.10.6.1. Sample customized install-config.yaml file for AWS

You can customize the installation configuration file (**install-config.yaml**) to specify more details about your OpenShift Container Platform cluster's platform or modify the values of the required parameters.

IMPORTANT

This sample YAML file is provided for reference only. You must obtain your **install-config.yaml** file by using the installation program and modify it. For a full list and description of all installation configuration parameters, see *Installation configuration parameters for AWS*.

### Sample install-config.yaml file for AWS

```
apiVersion: v1 (1)
baseDomain: example.com
sshKey: ssh-ed25519 AAAA...
pullSecret: '{"auths": ...}'
metadata:
  name: example-cluster
controlPlane: (2)
  name: master
  platform:
    aws:
      type: m6i.xlarge
  replicas: 3
compute: (3)
-  name: worker
  platform:
    aws:
      type: c5.4xlarge
  replicas: 3
networking: (4)
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
platform: (5)
  aws:
    region: us-west-2
```

1. Parameters at the first level of indentation apply to the cluster globally.
2. The **controlPlane** stanza applies to control plane machines.
3. The **compute** stanza applies to compute machines.
4. The **networking** stanza applies to the cluster networking configuration. If you do not provide networking values, the installation program provides default values.
5. The **platform** stanza applies to the infrastructure platform that hosts the cluster.

#### Additional resources

- Installation configuration parameters for AWS

#### 3.10.6.2. Minimum resource requirements for cluster installation

Each created cluster must meet minimum requirements so that the cluster runs as expected.

**Table 3.17. Minimum resource requirements**

| Machine | Operating System | vCPU [1] | Virtual RAM | Storage | Input/Output Per Second (IOPS)[2] |
|---|---|---|---|---|---|
| Bootstrap | RHCOS | 4 | 16 GB | 100 GB | 300 |
| Control plane | RHCOS | 4 | 16 GB | 100 GB | 300 |
| Compute | RHCOS, RHEL 8.6 and later [3] | 2 | 8 GB | 100 GB | 300 |

1. One vCPU is equivalent to one physical core when simultaneous multithreading (SMT), or Hyper-Threading, is not enabled. When enabled, use the following formula to calculate the corresponding ratio: (threads per core × cores) × sockets = vCPUs.

2. OpenShift Container Platform and Kubernetes are sensitive to disk performance, and faster storage is recommended, particularly for etcd on the control plane nodes which require a 10 ms p99 fsync duration. Note that on many cloud platforms, storage size and IOPS scale together, so you might need to over-allocate storage volume to obtain sufficient performance.

3. As with all user-provisioned installations, if you choose to use RHEL compute machines in your cluster, you take responsibility for all operating system life cycle management and maintenance, including performing system updates, applying patches, and completing all other required tasks. Use of RHEL 7 compute machines is deprecated and has been removed in OpenShift Container Platform 4.10 and later.

**NOTE**

For OpenShift Container Platform version 4.19, RHCOS is based on RHEL version 9.6, which updates the micro-architecture requirements. The following list contains the minimum instruction set architectures (ISA) that each architecture requires:

- x86-64 architecture requires x86-64-v2 ISA
- ARM64 architecture requires ARMv8.0-A ISA
- IBM Power architecture requires Power 9 ISA
- s390x architecture requires z14 ISA

For more information, see Architectures (RHEL documentation).

If an instance type for your platform meets the minimum requirements for cluster machines, it is supported to use in OpenShift Container Platform.

**Additional resources**

- Optimizing storage

### 3.10.6.3. Tested instance types for AWS

The following Amazon Web Services (AWS) instance types have been tested with OpenShift Container Platform.

NOTE

Use the machine types included in the following charts for your AWS instances. If you use an instance type that is not listed in the chart, ensure that the instance size you use matches the minimum resource requirements that are listed in the section named "Minimum resource requirements for cluster installation".

**Example 3.13. Machine types based on 64-bit x86 architecture**

- **c4.***
- **c5.***
- **c5a.***
- **i3.***
- **m4.***
- **m5.***
- **m5a.***
- **m6a.***
- **m6i.***
- **r4.***
- **r5.***
- **r5a.***
- **r6i.***
- **t3.***
- **t3a.***

### 3.10.6.4. Tested instance types for AWS on 64-bit ARM infrastructures

The following Amazon Web Services (AWS) 64-bit ARM instance types have been tested with OpenShift Container Platform.

NOTE

Use the machine types included in the following charts for your AWS ARM instances. If you use an instance type that is not listed in the chart, ensure that the instance size you use matches the minimum resource requirements that are listed in "Minimum resource requirements for cluster installation".

**Example 3.14. Machine types based on 64-bit ARM architecture**

- **c6g.***
- **c7g.***
- **m6g.***
- **m7g.***
- **r8g.***

### 3.10.6.5. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

#### Prerequisites

- You have an existing **install-config.yaml** file.
- You reviewed the sites that your cluster requires access to and determined whether any of them need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. You added sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.

  NOTE

  The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.

  For installations on Amazon Web Services (AWS), Google Cloud, Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

#### Procedure

1. Edit your **install-config.yaml** file and add the proxy settings. For example:

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> (1)
  httpsProxy: https://<username>:<pswd>@<ip>:<port> (2)
  noProxy: ec2.<aws_region>.amazonaws.com,elasticloadbalancing.
<aws_region>.amazonaws.com,s3.<aws_region>.amazonaws.com (3)
additionalTrustBundle: | (4)
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> (5)
```

1. A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**.
2. A proxy URL to use for creating HTTPS connections outside the cluster.
3. A comma-separated list of destination domain names, IP addresses, or other network CIDRs to exclude from proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use * to bypass the proxy for all destinations. If you have added the Amazon **EC2**,**Elastic Load Balancing**, and **S3** VPC endpoints to your VPC, you must add these endpoints to the **noProxy** field.
4. If provided, the installation program generates a config map that is named **user-ca-bundle** in the **openshift-config** namespace that contains one or more additional CA certificates that are required for proxying HTTPS connections. The Cluster Network Operator then creates a **trusted-ca-bundle** config map that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this config map is referenced in the **trustedCA** field of the **Proxy** object. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle.
5. Optional: The policy to determine the configuration of the **Proxy** object to reference the **user-ca-bundle** config map in the **trustedCA** field. The allowed values are **Proxyonly** and **Always**. Use **Proxyonly** to reference the **user-ca-bundle** config map only when **http/https** proxy is configured. Use **Always** to always reference the **user-ca-bundle** config map. The default value is **Proxyonly**.

NOTE

The installation program does not support the proxy **readinessEndpoints** field.

NOTE

If the installer times out, restart and then complete the deployment by using the **wait-for** command of the installer. For example:

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. Save the file and reference it when installing OpenShift Container Platform.

The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster Proxy** object is still created, but it will have a nil **spec**.

NOTE

Only the **Proxy** object named **cluster** is supported, and no additional proxies can be created.

#### 3.10.6.6. Applying existing AWS security groups to the cluster

Applying existing AWS security groups to your control plane and compute machines can help you meet the security needs of your organization, in such cases where you need to control the incoming or outgoing traffic of these machines.

##### Prerequisites

- You have created the security groups in AWS. For more information, see the AWS documentation about working with security groups.
- The security groups must be associated with the existing VPC that you are deploying the cluster to. The security groups cannot be associated with another VPC.
- You have an existing **install-config.yaml** file.

##### Procedure

1. In the **install-config.yaml** file, edit the **compute.platform.aws.additionalSecurityGroupIDs** parameter to specify one or more custom security groups for your compute machines.
2. Edit the **controlPlane.platform.aws.additionalSecurityGroupIDs** parameter to specify one or more custom security groups for your control plane machines.
3. Save the file and reference it when deploying the cluster.

##### Sample **install-config.yaml** file that specifies custom security groups

```
# ...
compute:
- hyperthreading: Enabled
  name: worker
  platform:
    aws:
      additionalSecurityGroupIDs:
        - sg-1 (1)
        - sg-2
  replicas: 3
controlPlane:
  hyperthreading: Enabled
  name: master
  platform:
    aws:
      additionalSecurityGroupIDs:
        - sg-3
        - sg-4
  replicas: 3
platform:
  aws:
    region: us-east-1
```

```
subnets: (2)
  - subnet-1
  - subnet-2
  - subnet-3
```

(1) Specify the name of the security group as it appears in the Amazon EC2 console, including the **sg** prefix.

(2) Specify subnets for each availability zone that your cluster uses.

### 3.10.7. Alternatives to storing administrator-level secrets in the kube-system project

By default, administrator secrets are stored in the **kube-system** project. If you configured the **credentialsMode** parameter in the **install-config.yaml** file to **Manual**, you must use one of the following alternatives:

- To manage long-term cloud credentials manually, follow the procedure in Manually creating long-term credentials.
- To implement short-term credentials that are managed outside the cluster for individual components, follow the procedures in Configuring an AWS cluster to use short-term credentials.

#### 3.10.7.1. Manually creating long-term credentials

The Cloud Credential Operator (CCO) can be put into manual mode prior to installation in environments where the cloud identity and access management (IAM) APIs are not reachable, or the administrator prefers not to store an administrator-level credential secret in the cluster **kube-system** namespace.

**Procedure**

1. If you did not set the **credentialsMode** parameter in the **install-config.yaml** configuration file to **Manual**, modify the value as shown:

   **Sample configuration file snippet**

   ```
   apiVersion: v1
   baseDomain: example.com
   credentialsMode: Manual
   # ...
   ```

2. If you have not previously created installation manifest files, do so by running the following command:

   ```
   $ openshift-install create manifests --dir <installation_directory>
   ```

   where **<installation_directory>** is the directory in which the installation program creates files.

3. Set a **$RELEASE_IMAGE** variable with the release image from your installation file by running the following command:

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

4. Extract the list of **CredentialsRequest** custom resources (CRs) from the OpenShift Container Platform release image by running the following command:

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

1 The **--included** parameter includes only the manifests that your specific cluster configuration requires.

2 Specify the location of the **install-config.yaml** file.

3 Specify the path to the directory where you want to store the **CredentialsRequest** objects. If the specified directory does not exist, this command creates it.

This command creates a YAML file for each **CredentialsRequest** object.

### Sample CredentialsRequest object

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
    - effect: Allow
      action:
      - iam:GetUser
      - iam:GetUserPolicy
      - iam:ListAccessKeys
      resource: "*"
  ...
```

5. Create YAML files for secrets in the **openshift-install** manifests directory that you generated previously. The secrets must be stored using the namespace and secret name defined in the **spec.secretRef** for each **CredentialsRequest** object.

### Sample CredentialsRequest object with secrets

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <component_credentials_request>
  namespace: openshift-cloud-credential-operator
  ...
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
    - effect: Allow
      action:
      - s3:CreateBucket
      - s3:DeleteBucket
      resource: "*"
      ...
  secretRef:
    name: <component_secret>
    namespace: <component_namespace>
  ...
```

**Sample Secret object**

```
apiVersion: v1
kind: Secret
metadata:
  name: <component_secret>
  namespace: <component_namespace>
data:
  aws_access_key_id: <base64_encoded_aws_access_key_id>
  aws_secret_access_key: <base64_encoded_aws_secret_access_key>
```

IMPORTANT

Before upgrading a cluster that uses manually maintained credentials, you must ensure that the CCO is in an upgradeable state.

### 3.10.7.2. Configuring an AWS cluster to use short-term credentials

To install a cluster that is configured to use the AWS Security Token Service (STS), you must configure the CCO utility and create the required AWS resources for your cluster.

#### 3.10.7.2.1. Configuring the Cloud Credential Operator utility

To create and manage cloud credentials from outside of the cluster when the Cloud Credential Operator (CCO) is operating in manual mode, extract and prepare the CCO utility (**ccoctl**) binary.

NOTE

The **ccoctl** utility is a Linux binary that must run in a Linux environment.

**Prerequisites**

- You have access to an OpenShift Container Platform account with cluster administrator access.
- You have installed the OpenShift CLI (**oc**).

- You have created an AWS account for the **ccoctl** utility to use with the following permissions:
  Required **iam** permissions
  - **iam:CreateOpenIDConnectProvider**
  - **iam:CreateRole**
  - **iam:DeleteOpenIDConnectProvider**
  - **iam:DeleteRole**
  - **iam:DeleteRolePolicy**
  - **iam:GetOpenIDConnectProvider**
  - **iam:GetRole**
  - **iam:GetUser**
  - **iam:ListOpenIDConnectProviders**
  - **iam:ListRolePolicies**
  - **iam:ListRoles**
  - **iam:PutRolePolicy**
  - **iam:TagOpenIDConnectProvider**
  - **iam:TagRole**

  Required **s3** permissions
  - **s3:CreateBucket**
  - **s3:DeleteBucket**
  - **s3:DeleteObject**
  - **s3:GetBucketAcl**
  - **s3:GetBucketTagging**
  - **s3:GetObject**
  - **s3:GetObjectAcl**
  - **s3:GetObjectTagging**
  - **s3:ListBucket**
  - **s3:PutBucketAcl**
  - **s3:PutBucketPolicy**
  - **s3:PutBucketPublicAccessBlock**

- **s3:PutBucketTagging**
- **s3:PutObject**
- **s3:PutObjectAcl**
- **s3:PutObjectTagging**

Required **cloudfront** permissions

- **cloudfront:ListCloudFrontOriginAccessIdentities**
- **cloudfront:ListDistributions**
- **cloudfront:ListTagsForResource**

- If you plan to store the OIDC configuration in a private S3 bucket that is accessed by the IAM identity provider through a public CloudFront distribution URL, the AWS account that runs the **ccoctl** utility requires the following additional permissions:
  - **cloudfront:CreateCloudFrontOriginAccessIdentity**
  - **cloudfront:CreateDistribution**
  - **cloudfront:DeleteCloudFrontOriginAccessIdentity**
  - **cloudfront:DeleteDistribution**
  - **cloudfront:GetCloudFrontOriginAccessIdentity**
  - **cloudfront:GetCloudFrontOriginAccessIdentityConfig**
  - **cloudfront:GetDistribution**
  - **cloudfront:TagResource**
  - **cloudfront:UpdateDistribution**

NOTE

These additional permissions support the use of the **--create-private-s3-bucket** option when processing credentials requests with the **ccoctl aws create-all** command.

Procedure

1. Set a variable for the OpenShift Container Platform release image by running the following command:

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. Obtain the CCO container image from the OpenShift Container Platform release image by running the following command:

```
$ CCO_IMAGE=$(oc adm release info --image-for='cloud-credential-operator'
$RELEASE_IMAGE -a ~/.pull-secret)
```

NOTE

Ensure that the architecture of the **$RELEASE_IMAGE** matches the architecture of the environment in which you will use the **ccoctl** tool.

3. Extract the **ccoctl** binary from the CCO container image within the OpenShift Container Platform release image by running the following command:

```
$ oc image extract $CCO_IMAGE \
  --file="/usr/bin/ccoctl.<rhel_version>" \ 1
  -a ~/.pull-secret
```

1 For **<rhel_version>**, specify the value that corresponds to the version of Red Hat Enterprise Linux (RHEL) that the host uses. If no value is specified, **ccoctl.rhel8** is used by default. The following values are valid:

- **rhel8**: Specify this value for hosts that use RHEL 8.
- **rhel9**: Specify this value for hosts that use RHEL 9.

NOTE

The **ccoctl** binary is created in the directory from where you executed the command and not in **/usr/bin/**. You must rename the directory or move the **ccoctl.<rhel_version>** binary to **ccoctl**.

4. Change the permissions to make **ccoctl** executable by running the following command:

```
$ chmod 775 ccoctl
```

## Verification

- To verify that **ccoctl** is ready to use, display the help file. Use a relative file name when you run the command, for example:

```
$ ./ccoctl
```

### Example output

```
OpenShift credentials provisioning tool

Usage:
  ccoctl [command]

Available Commands:
  aws         Manage credentials objects for AWS cloud
  azure       Manage credentials objects for Azure
  gcp         Manage credentials objects for Google cloud
  help        Help about any command
  ibmcloud     Manage credentials objects for {ibm-cloud-title}
  nutanix      Manage credentials objects for Nutanix

Flags:
  -h, --help   help for ccoctl

Use "ccoctl [command] --help" for more information about a command.
```

#### 3.10.7.2.2. Creating AWS resources with the Cloud Credential Operator utility

You have the following options when creating AWS resources:

- You can use the **ccoctl aws create-all** command to create the AWS resources automatically. This is the quickest way to create the resources. See Creating AWS resources with a single command.
- If you need to review the JSON files that the **ccoctl** tool creates before modifying AWS resources, or if the process the **ccoctl** tool uses to create AWS resources automatically does not meet the requirements of your organization, you can create the AWS resources individually. See Creating AWS resources individually .

##### 3.10.7.2.2.1. Creating AWS resources with a single command

If the process the **ccoctl** tool uses to create AWS resources automatically meets the requirements of your organization, you can use the **ccoctl aws create-all** command to automate the creation of AWS resources.

Otherwise, you can create the AWS resources individually. For more information, see "Creating AWS resources individually".

NOTE

By default, **ccoctl** creates objects in the directory in which the commands are run. To create the objects in a different directory, use the **--output-dir** flag. This procedure uses **<path_to_ccoctl_output_dir>** to refer to this directory.

### Prerequisites

You must have:

- Extracted and prepared the **ccoctl** binary.

### Procedure

1. Set a **$RELEASE_IMAGE** variable with the release image from your installation file by running the following command:

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. Extract the list of **CredentialsRequest** objects from the OpenShift Container Platform release image by running the following command:

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \ 2
  --to=<path_to_directory_for_credentials_requests> 3
```

1 The **--included** parameter includes only the manifests that your specific cluster configuration requires.

2 Specify the location of the **install-config.yaml** file.

3 Specify the path to the directory where you want to store the **CredentialsRequest** objects. If the specified directory does not exist, this command creates it.

NOTE

This command might take a few moments to run.

3. Use the **ccoctl** tool to process all **CredentialsRequest** objects by running the following command:

```
$ ccoctl aws create-all \
  --name=<name> \ 1
  --region=<aws_region> \ 2
  --credentials-requests-dir=<path_to_credentials_requests_directory> \ 3
  --output-dir=<path_to_ccoctl_output_dir> \ 4
  --create-private-s3-bucket 5
```

1 Specify the name used to tag any cloud resources that are created for tracking.

2 Specify the AWS region in which cloud resources will be created.

3 Specify the directory containing the files for the component **CredentialsRequest** objects.

4 Optional: Specify the directory in which you want the **ccoctl** utility to create objects. By default, the utility creates objects in the directory in which the commands are run.

5 Optional: By default, the **ccoctl** utility stores the OpenID Connect (OIDC) configuration files in a public S3 bucket and uses the S3 URL as the public OIDC endpoint. To store the OIDC configuration in a private S3 bucket that is accessed by the IAM identity provider through a public CloudFront distribution URL instead, use the **--create-private-s3-bucket** parameter.

NOTE

If your cluster uses Technology Preview features that are enabled by the **TechPreviewNoUpgrade** feature set, you must include the **--enable-tech-preview** parameter.

Verification

- To verify that the OpenShift Container Platform secrets are created, list the files in the **<path_to_ccoctl_output_dir>/manifests** directory:

  ```
  $ ls <path_to_ccoctl_output_dir>/manifests
  ```

  **Example output**

  ```
  cluster-authentication-02-config.yaml
  openshift-cloud-credential-operator-cloud-credential-operator-iam-ro-creds-credentials.yaml
  openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
  openshift-cluster-api-capa-manager-bootstrap-credentials-credentials.yaml
  openshift-cluster-csi-drivers-ebs-cloud-credentials-credentials.yaml
  openshift-image-registry-installer-cloud-credentials-credentials.yaml
  openshift-ingress-operator-cloud-credentials-credentials.yaml
  openshift-machine-api-aws-cloud-credentials-credentials.yaml
  ```

  You can verify that the IAM roles are created by querying AWS. For more information, refer to AWS documentation on listing IAM roles.

###### 3.10.7.2.2.2. Creating AWS resources individually

You can use the **ccoctl** tool to create AWS resources individually. This option might be useful for an organization that shares the responsibility for creating these resources among different users or departments.

Otherwise, you can use the **ccoctl aws create-all** command to create the AWS resources automatically. For more information, see "Creating AWS resources with a single command".

**NOTE**

By default, **ccoctl** creates objects in the directory in which the commands are run. To create the objects in a different directory, use the **--output-dir** flag. This procedure uses **<path_to_ccoctl_output_dir>** to refer to this directory.

Some **ccoctl** commands make AWS API calls to create or modify AWS resources. You can use the **--dry-run** flag to avoid making API calls. Using this flag creates JSON files on the local file system instead. You can review and modify the JSON files and then apply them with the AWS CLI tool using the **--cli-input-json** parameters.

**Prerequisites**

- Extract and prepare the **ccoctl** binary.

**Procedure**

1. Generate the public and private RSA key files that are used to set up the OpenID Connect provider for the cluster by running the following command:

   ```
   $ ccoctl aws create-key-pair
   ```

   **Example output**

   ```
   2021/04/13 11:01:02 Generating RSA keypair
   ```

```
2021/04/13 11:01:03 Writing private key to /<path_to_ccoctl_output_dir>/serviceaccount-signer.private
2021/04/13 11:01:03 Writing public key to /<path_to_ccoctl_output_dir>/serviceaccount-signer.public
2021/04/13 11:01:03 Copying signing key for use by installer
```

where **serviceaccount-signer.private** and **serviceaccount-signer.public** are the generated key files.

This command also creates a private key that the cluster requires during installation in **/<path_to_ccoctl_output_dir>/tls/bound-service-account-signing-key.key**.

2. Create an OpenID Connect identity provider and S3 bucket on AWS by running the following command:

```
$ ccoctl aws create-identity-provider \
  --name=<name> \ 1
  --region=<aws_region> \ 2
  --public-key-file=<path_to_ccoctl_output_dir>/serviceaccount-signer.public 3
```

1. **<name>** is the name used to tag any cloud resources that are created for tracking.
2. **<aws-region>** is the AWS region in which cloud resources will be created.
3. **<path_to_ccoctl_output_dir>** is the path to the public key file that the **ccoctl aws create-key-pair** command generated.

### Example output

```
2021/04/13 11:16:09 Bucket <name>-oidc created
2021/04/13 11:16:10 OpenID Connect discovery document in the S3 bucket <name>-oidc at .well-known/openid-configuration updated
2021/04/13 11:16:10 Reading public key
2021/04/13 11:16:10 JSON web key set (JWKS) in the S3 bucket <name>-oidc at keys.json updated
2021/04/13 11:16:18 Identity Provider created with ARN: arn:aws:iam::<aws_account_id>:oidc-provider/<name>-oidc.s3.<aws_region>.amazonaws.com
```

where **openid-configuration** is a discovery document and **keys.json** is a JSON web key set file.

This command also creates a YAML configuration file in **/<path_to_ccoctl_output_dir>/manifests/cluster-authentication-02-config.yaml**. This file sets the issuer URL field for the service account tokens that the cluster generates, so that the AWS IAM identity provider trusts the tokens.

3. Create IAM roles for each component in the cluster:

   a. Set a **$RELEASE_IMAGE** variable with the release image from your installation file by running the following command:

   ```
   $ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
   ```

b. Extract the list of **CredentialsRequest** objects from the OpenShift Container Platform release image:

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --included \ 1
  --install-config=<path_to_directory_with_installation_configuration>/install-config.yaml \
2
  --to=<path_to_directory_for_credentials_requests> 3
```

1. The **--included** parameter includes only the manifests that your specific cluster configuration requires.
2. Specify the location of the **install-config.yaml** file.
3. Specify the path to the directory where you want to store the **CredentialsRequest** objects. If the specified directory does not exist, this command creates it.

c. Use the **ccoctl** tool to process all **CredentialsRequest** objects by running the following command:

```
$ ccoctl aws create-iam-roles \
  --name=<name> \
  --region=<aws_region> \
  --credentials-requests-dir=<path_to_credentials_requests_directory> \
  --identity-provider-arn=arn:aws:iam::<aws_account_id>:oidc-provider/<name>-oidc.s3.
<aws_region>.amazonaws.com
```

NOTE

For AWS environments that use alternative IAM API endpoints, such as GovCloud, you must also specify your region with the **--region** parameter.

If your cluster uses Technology Preview features that are enabled by the **TechPreviewNoUpgrade** feature set, you must include the **--enable-tech-preview** parameter.

For each **CredentialsRequest** object, **ccoctl** creates an IAM role with a trust policy that is tied to the specified OIDC identity provider, and a permissions policy as defined in each **CredentialsRequest** object from the OpenShift Container Platform release image.

### Verification

- To verify that the OpenShift Container Platform secrets are created, list the files in the **<path_to_ccoctl_output_dir>/manifests** directory:

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

#### Example output

```
cluster-authentication-02-config.yaml
```

```
openshift-cloud-credential-operator-cloud-credential-operator-iam-ro-creds-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capa-manager-bootstrap-credentials-credentials.yaml
openshift-cluster-csi-drivers-ebs-cloud-credentials-credentials.yaml
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-aws-cloud-credentials-credentials.yaml
```

You can verify that the IAM roles are created by querying AWS. For more information, refer to AWS documentation on listing IAM roles.

#### 3.10.7.2.3. Incorporating the Cloud Credential Operator utility manifests

To implement short-term security credentials managed outside the cluster for individual components, you must move the manifest files that the Cloud Credential Operator utility (**ccoctl**) created to the correct directories for the installation program.

**Prerequisites**

- You have configured an account with the cloud platform that hosts your cluster.
- You have configured the Cloud Credential Operator utility (**ccoctl**).
- You have created the cloud provider resources that are required for your cluster with the **ccoctl** utility.

**Procedure**

1. If you did not set the **credentialsMode** parameter in the **install-config.yaml** configuration file to **Manual**, modify the value as shown:

   **Sample configuration file snippet**

   ```
   apiVersion: v1
   baseDomain: example.com
   credentialsMode: Manual
   # ...
   ```

2. If you have not previously created installation manifest files, do so by running the following command:

   ```
   $ openshift-install create manifests --dir <installation_directory>
   ```

   where **<installation_directory>** is the directory in which the installation program creates files.

3. Copy the manifests that the **ccoctl** utility generated to the **manifests** directory that the installation program created by running the following command:

   ```
   $ cp /<path_to_ccoctl_output_dir>/manifests/* ./manifests/
   ```

4. Copy the **tls** directory that contains the private key to the installation directory:

   ```
   $ cp -a /<path_to_ccoctl_output_dir>/tls .
   ```

### 3.10.8. Deploying the cluster

You can install OpenShift Container Platform on a compatible cloud platform.

IMPORTANT

You can run the **create cluster** command of the installation program only once, during initial installation.

#### Prerequisites

- You have configured an account with the cloud platform that hosts your cluster.
- You have the OpenShift Container Platform installation program and the pull secret for your cluster.
- You have verified that the cloud provider account on your host has the correct permissions to deploy the cluster. An account with incorrect permissions causes the installation process to fail with an error message that displays the missing permissions.

#### Procedure

1. In the directory that contains the installation program, initialize the cluster deployment by running the following command:

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 For **<installation_directory>**, specify the location of your customized **./install-config.yaml** file.

2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

2. Optional: Remove or disable the **AdministratorAccess** policy from the IAM account that you used to install the cluster.

NOTE

The elevated permissions provided by the **AdministratorAccess** policy are required only during installation.

#### Verification

When the cluster deployment completes successfully:

- The terminal displays directions for accessing your cluster, including a link to the web console and credentials for the **kubeadmin** user.
- Credential information also outputs to **<installation_directory>/.openshift_install.log**.

IMPORTANT

Do not delete the installation program or the files that the installation program creates. Both are required to delete the cluster.

### Example output

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```

IMPORTANT

- The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

- It is recommended that you use Ignition config files within 12 hours after they are generated because the 24-hour certificate rotates from 16 to 22 hours after the cluster is installed. By using the Ignition config files within 12 hours, you can avoid installation failure if the certificate update runs during installation.

## 3.10.9. Logging in to the cluster by using the CLI

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

### Prerequisites

- You deployed an OpenShift Container Platform cluster.

- You installed the OpenShift CLI (**oc**).

### Procedure

1. Export the **kubeadmin** credentials by running the following command:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration by running the following command:

   ```
   $ oc whoami
   ```

   **Example output**

   ```
   system:admin
   ```

### 3.10.10. Logging in to the cluster by using the web console

The **kubeadmin** user exists by default after an OpenShift Container Platform installation. You can log in to your cluster as the **kubeadmin** user by using the OpenShift Container Platform web console.

**Prerequisites**

- You have access to the installation host.
- You completed a cluster installation and all cluster Operators are available.

**Procedure**

1. Obtain the password for the **kubeadmin** user from the **kubeadmin-password** file on the installation host:

   ```
   $ cat <installation_directory>/auth/kubeadmin-password
   ```

   NOTE

   Alternatively, you can obtain the **kubeadmin** password from the **<installation_directory>/.openshift_install.log** log file on the installation host.

2. List the OpenShift Container Platform web console route:

   ```
   $ oc get routes -n openshift-console | grep 'console-openshift'
   ```

   NOTE

   Alternatively, you can obtain the OpenShift Container Platform route from the **<installation_directory>/.openshift_install.log** log file on the installation host.

   **Example output**

   ```
   console     console-openshift-console.apps.<cluster_name>.<base_domain>            console
   https   reencrypt/Redirect   None
   ```

3. Navigate to the route detailed in the output of the preceding command in a web browser and log in as the **kubeadmin** user.

**Additional resources**

- Accessing the web console

### 3.10.11. Next steps

- Validating an installation.
- Customize your cluster.
- If necessary, you can Remote health reporting.
- If necessary, you can remove cloud provider credentials.

## 3.11. INSTALLING A CLUSTER WITH COMPUTE NODES ON AWS LOCAL ZONES

You can quickly install an OpenShift Container Platform cluster on Amazon Web Services (AWS) Local Zones by setting the zone names in the edge compute pool of the **install-config.yaml** file, or install a cluster in an existing Amazon Virtual Private Cloud (VPC) with Local Zone subnets.

AWS Local Zones is an infrastructure that place Cloud Resources close to metropolitan regions. For more information, see the AWS Local Zones Documentation.

### 3.11.1. Infrastructure prerequisites

- You reviewed details about OpenShift Container Platform installation and update processes.
- You are familiar with Selecting a cluster installation method and preparing it for users.
- You configured an AWS account to host the cluster.

> **WARNING**
>
> If you have an AWS profile stored on your computer, it must not use a temporary session token that you generated while using a multifactor authentication device. The cluster continues to use your current AWS credentials to create AWS resources for the entire life of the cluster, so you must use key-based, long-term credentials. To generate appropriate keys, see Managing Access Keys for IAM Users in the AWS documentation. You can supply the keys when you run the installation program.

- You downloaded the AWS CLI and installed it on your computer. See Install the AWS CLI Using the Bundled Installer (Linux, macOS, or UNIX) in the AWS documentation.
- If you use a firewall, you configured it to allow the sites that your cluster must access.
- You noted the region and supported AWS Local Zones locations to create the network resources in.
- You read the AWS Local Zones features in the AWS documentation.

- You added permissions for creating network resources that support AWS Local Zones to the Identity and Access Management (IAM) user or role. The following example enables a zone group that can give a user or role access for creating network resources that support AWS Local Zones.

  **Example of an additional IAM policy with the ec2:ModifyAvailabilityZoneGroup permission attached to an IAM user or role.**

  ```
  {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Action": [
          "ec2:ModifyAvailabilityZoneGroup"
        ],
        "Effect": "Allow",
        "Resource": "*"
      }
    ]
  }
  ```

### 3.11.2. About AWS Local Zones and edge compute pool

Read the following sections to understand infrastructure behaviors and cluster limitations in an AWS Local Zones environment.

#### 3.11.2.1. Cluster limitations in AWS Local Zones

Some limitations exist when you try to deploy a cluster with a default installation configuration in an Amazon Web Services (AWS) Local Zone.

IMPORTANT

The following list details limitations when deploying a cluster in a pre-configured AWS zone:

- The maximum transmission unit (MTU) between an Amazon EC2 instance in a zone and an Amazon EC2 instance in the Region is **1300**. This causes the cluster-wide network MTU to change according to the network plugin that is used with the deployment.
- Network resources such as Network Load Balancer (NLB), Classic Load Balancer, and Network Address Translation (NAT) Gateways are not globally supported.
- For an OpenShift Container Platform cluster on AWS, the AWS Elastic Block Storage (EBS) **gp3** type volume is the default for node volumes and the default for the storage class. This volume type is not globally available on zone locations. By default, the nodes running in zones are deployed with the **gp2** EBS volume. The **gp2-csi StorageClass** parameter must be set when creating workloads on zone nodes.

If you want the installation program to automatically create Local Zone subnets for your OpenShift Container Platform cluster, specific configuration limitations apply with this method.

IMPORTANT

The following configuration limitation applies when you set the installation program to automatically create subnets for your OpenShift Container Platform cluster:

- When the installation program creates private subnets in AWS Local Zones, the program associates each subnet with the route table of its parent zone. This operation ensures that each private subnet can route egress traffic to the internet by way of NAT Gateways in an AWS Region.
- If the parent-zone route table does not exist during cluster installation, the installation program associates any private subnet with the first available private route table in the Amazon Virtual Private Cloud (VPC). This approach is valid only for AWS Local Zones subnets in an OpenShift Container Platform cluster.

#### 3.11.2.2. About edge compute pools

Edge compute nodes are tainted compute nodes that run in AWS Local Zones locations.

When deploying a cluster that uses Local Zones, consider the following points:

- Amazon EC2 instances in the Local Zones are more expensive than Amazon EC2 instances in the Availability Zones.
- The latency is lower between the applications running in AWS Local Zones and the end user. A latency impact exists for some workloads if, for example, ingress traffic is mixed between Local Zones and Availability Zones.

IMPORTANT

Generally, the maximum transmission unit (MTU) between an Amazon EC2 instance in a Local Zones and an Amazon EC2 instance in the Region is 1300. The cluster network MTU must be always less than the EC2 MTU to account for the overhead. The specific overhead is determined by the network plugin. For example: OVN-Kubernetes has an overhead of **100 bytes**.

The network plugin can provide additional features, such as IPsec, that also affect the MTU sizing.

For more information, see How Local Zones work in the AWS documentation.

OpenShift Container Platform 4.12 introduced a new compute pool, *edge*, that is designed for use in remote zones. The edge compute pool configuration is common between AWS Local Zones locations. Because of the type and size limitations of resources like EC2 and EBS on Local Zones resources, the default instance type can vary from the traditional compute pool.

The default Elastic Block Store (EBS) for Local Zones locations is **gp2**, which differs from the non-edge compute pool. The instance type used for each Local Zones on an edge compute pool also might differ from other compute pools, depending on the instance offerings on the zone.

The edge compute pool creates new labels that developers can use to deploy applications onto AWS Local Zones nodes. The new labels are:

- **node-role.kubernetes.io/edge=''**
- **machine.openshift.io/zone-type=local-zone**

- **machine.openshift.io/zone-group=$ZONE_GROUP_NAME**

By default, the machine sets for the edge compute pool define the taint of **NoSchedule** to prevent other workloads from spreading on Local Zones instances. Users can only run user workloads if they define tolerations in the pod specification.

### Additional resources

- MTU value selection
- Changing the MTU for the cluster network
- Understanding taints and tolerations
- Storage classes
- Ingress Controller sharding

## 3.11.3. Installation prerequisites

Before you install a cluster in an AWS Local Zones environment, you must configure your infrastructure so that it can adopt Local Zone capabilities.

### 3.11.3.1. Opting in to an AWS Local Zones

If you plan to create subnets in AWS Local Zones, you must opt in to each zone group separately.

#### Prerequisites

- You have installed the AWS CLI.
- You have determined an AWS Region for where you want to deploy your OpenShift Container Platform cluster.
- You have attached a permissive IAM policy to a user or role account that opts in to the zone group.

#### Procedure

1. List the zones that are available in your AWS Region by running the following command:

   **Example command for listing available AWS Local Zones in an AWS Region**

   ```
   $ aws --region "<value_of_AWS_Region>" ec2 describe-availability-zones \
       --query 'AvailabilityZones[].[{ZoneName: ZoneName, GroupName: GroupName, Status: OptInStatus}]' \
       --filters Name=zone-type,Values=local-zone \
       --all-availability-zones
   ```

   Depending on the AWS Region, the list of available zones might be long. The command returns the following fields:

   **ZoneName**

   The name of the Local Zones.

**GroupName**

The group that comprises the zone. To opt in to the Region, save the name.

**Status**

The status of the Local Zones group. If the status is **not-opted-in**, you must opt in the **GroupName** as described in the next step.

2. Opt in to the zone group on your AWS account by running the following command:

```
$ aws ec2 modify-availability-zone-group \
    --group-name "<value_of_GroupName>" \ 1
    --opt-in-status opted-in
```

1 Replace **<value_of_GroupName>** with the name of the group of the Local Zones where you want to create subnets. For example, specify **us-east-1-nyc-1** to use the zone **us-east-1-nyc-1a** (US East New York).

### 3.11.3.2. Obtaining an AWS Marketplace image

If you are deploying an OpenShift Container Platform cluster using an AWS Marketplace image, you must first subscribe through AWS. Subscribing to the offer provides you with the AMI ID that the installation program uses to deploy compute nodes.

NOTE

You should only modify the RHCOS image for compute machines to use an AWS Marketplace image. Control plane machines and infrastructure nodes do not require an OpenShift Container Platform subscription and use the public RHCOS default image by default, which does not incur subscription costs on your AWS bill. Therefore, you should not modify the cluster default boot image or the control plane boot images. Applying the AWS Marketplace image to them will incur additional licensing costs that cannot be recovered.

#### Prerequisites

- You have an AWS account to purchase the offer. This account does not have to be the same account that is used to install the cluster.

#### Procedure

1. Complete the OpenShift Container Platform subscription from the AWS Marketplace.

2. Record the AMI ID for your specific AWS Region. As part of the installation process, you must update the **install-config.yaml** file with this value before deploying the cluster.

**Sample install-config.yaml file with AWS Marketplace compute nodes**

```
apiVersion: v1
baseDomain: example.com
compute:
- hyperthreading: Enabled
  name: worker
  platform:
    aws:
      amiID: ami-06c4d345f7c207239 1
      type: m5.4xlarge
  replicas: 3
metadata:
  name: test-cluster
platform:
  aws:
    region: us-east-2 2
sshKey: ssh-ed25519 AAAA...
pullSecret: '{"auths": ...}'
```

1. The AMI ID from your AWS Marketplace subscription.
2. Your AMI ID is associated with a specific AWS Region. When creating the installation configuration file, ensure that you select the same AWS Region that you specified when configuring your subscription.

### 3.11.4. Preparing for the installation

Before you extend nodes to Local Zones, you must prepare certain resources for the cluster installation environment.

#### 3.11.4.1. Minimum resource requirements for cluster installation

Each created cluster must meet minimum requirements so that the cluster runs as expected.

**Table 3.18. Minimum resource requirements**

| Machine | Operating System | vCPU [1] | Virtual RAM | Storage | Input/Output Per Second (IOPS)[2] |
|---|---|---|---|---|---|
| Bootstrap | RHCOS | 4 | 16 GB | 100 GB | 300 |
| Control plane | RHCOS | 4 | 16 GB | 100 GB | 300 |
| Compute | RHCOS, RHEL 8.6 and later [3] | 2 | 8 GB | 100 GB | 300 |

1. One vCPU is equivalent to one physical core when simultaneous multithreading (SMT), or Hyper-Threading, is not enabled. When enabled, use the following formula to calculate the corresponding ratio: (threads per core × cores) × sockets = vCPUs.
2. OpenShift Container Platform and Kubernetes are sensitive to disk performance, and faster storage is recommended, particularly for etcd on the control plane nodes which require a 10 ms p99 fsync duration. Note that on many cloud platforms, storage size and IOPS scale together, so you might need to over-allocate storage volume to obtain sufficient performance.
3. As with all user-provisioned installations, if you choose to use RHEL compute machines in your cluster, you take responsibility for all operating system life cycle management and maintenance,

including performing system updates, applying patches, and completing all other required tasks. Use of RHEL 7 compute machines is deprecated and has been removed in OpenShift Container Platform 4.10 and later.

NOTE

For OpenShift Container Platform version 4.19, RHCOS is based on RHEL version 9.6, which updates the micro-architecture requirements. The following list contains the minimum instruction set architectures (ISA) that each architecture requires:

- x86-64 architecture requires x86-64-v2 ISA
- ARM64 architecture requires ARMv8.0-A ISA
- IBM Power architecture requires Power 9 ISA
- s390x architecture requires z14 ISA

For more information, see Architectures (RHEL documentation).

If an instance type for your platform meets the minimum requirements for cluster machines, it is supported to use in OpenShift Container Platform.

#### 3.11.4.2. Tested instance types for AWS

The following Amazon Web Services (AWS) instance types have been tested with OpenShift Container Platform for use with AWS Local Zones.

NOTE

Use the machine types included in the following charts for your AWS instances. If you use an instance type that is not listed in the chart, ensure that the instance size you use matches the minimum resource requirements that are listed in the section named "Minimum resource requirements for cluster installation".

**Example 3.15. Machine types based on 64-bit x86 architecture for AWS Local Zones**

- **c5.***
- **c5d.***
- **m6i.***
- **m5.***
- **r5.***
- **t3.***

**Additional resources**

- AWS Local Zones features (AWS documentation)

### 3.11.4.3. Creating the installation configuration file

Generate and customize the installation configuration file that the installation program needs to deploy your cluster.

#### Prerequisites

- You obtained the OpenShift Container Platform installation program for user-provisioned infrastructure and the pull secret for your cluster.
- You checked that you are deploying your cluster to an AWS Region with an accompanying Red Hat Enterprise Linux CoreOS (RHCOS) AMI published by Red Hat. If you are deploying to an AWS Region that requires a custom AMI, such as an AWS GovCloud Region, you must create the **install-config.yaml** file manually.

#### Procedure

1. Create the **install-config.yaml** file.
   a. Change to the directory that contains the installation program and run the following command:

      ```
      $ ./openshift-install create install-config --dir <installation_directory> (1)
      ```

      (1) For **<installation_directory>**, specify the directory name to store the files that the installation program creates.

      IMPORTANT

      Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

   b. At the prompts, provide the configuration details for your cloud:
      i. Optional: Select an SSH key to use to access your cluster machines.

         NOTE

         For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

      ii. Select **aws** as the platform to target.
      iii. If you do not have an AWS profile stored on your computer, enter the AWS access key ID and secret access key for the user that you configured to run the installation program.

NOTE

The AWS access key ID and secret access key are stored in **~/.aws/credentials** in the home directory of the current user on the installation host. You are prompted for the credentials by the installation program if the credentials for the exported profile are not present in the file. Any credentials that you provide to the installation program are stored in the file.

iv. Select the AWS Region to deploy the cluster to.

v. Select the base domain for the Route 53 service that you configured for your cluster.

vi. Enter a descriptive name for your cluster.

vii. Paste the pull secret from Red Hat OpenShift Cluster Manager .

2. Optional: Back up the **install-config.yaml** file.

IMPORTANT

The **install-config.yaml** file is consumed during the installation process. If you want to reuse the file, you must back it up now.

### 3.11.4.4. Examples of installation configuration files with edge compute pools

The following examples show **install-config.yaml** files that contain an edge machine pool configuration.

#### Configuration that uses an edge pool with a custom instance type

```
apiVersion: v1
baseDomain: devcluster.openshift.com
metadata:
  name: ipi-edgezone
compute:
- name: edge
  platform:
    aws:
      type: r5.2xlarge
platform:
  aws:
    region: us-west-2
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...
```

Instance types differ between locations. To verify availability in the Local Zones in which the cluster runs, see the AWS documentation.

#### Configuration that uses an edge pool with a custom Amazon Elastic Block Store (EBS) type

```
apiVersion: v1
baseDomain: devcluster.openshift.com
metadata:
  name: ipi-edgezone
```

```
compute:
- name: edge
  platform:
    aws:
      zones:
      - us-west-2-lax-1a
      - us-west-2-lax-1b
      - us-west-2-phx-2a
      rootVolume:
        type: gp3
        size: 120
platform:
  aws:
    region: us-west-2
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...
```

Elastic Block Storage (EBS) types differ between locations. Check the AWS documentation to verify availability in the Local Zones in which the cluster runs.

**Configuration that uses an edge pool with custom security groups**

```
apiVersion: v1
baseDomain: devcluster.openshift.com
metadata:
  name: ipi-edgezone
compute:
- name: edge
  platform:
    aws:
      additionalSecurityGroupIDs:
        - sg-1 1
        - sg-2
platform:
  aws:
    region: us-west-2
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...
```

1 Specify the name of the security group as it is displayed on the Amazon EC2 console. Ensure that you include the **sg** prefix.

#### 3.11.4.5. Customizing the cluster network MTU

Before you deploy a cluster on AWS, you can customize the cluster network maximum transmission unit (MTU) for your cluster network to meet the needs of your infrastructure.

By default, when you install a cluster with supported Local Zones capabilities, the MTU value for the cluster network is automatically adjusted to the lowest value that the network plugin accepts.

**IMPORTANT**

Setting an unsupported MTU value for EC2 instances that operate in the Local Zones infrastructure can cause issues for your OpenShift Container Platform cluster.

If the Local Zone supports higher MTU values in between EC2 instances in the Local Zone and the AWS Region, you can manually configure the higher value to increase the network performance of the cluster network.

You can customize the MTU for a cluster by specifying the **networking.clusterNetworkMTU** parameter in the **install-config.yaml** configuration file.

IMPORTANT

All subnets in Local Zones must support the higher MTU value, so that each node in that zone can successfully communicate with services in the AWS Region and deploy your workloads.

### Example of overwriting the default MTU value

```
apiVersion: v1
baseDomain: devcluster.openshift.com
metadata:
  name: edge-zone
networking:
  clusterNetworkMTU: 8901
compute:
- name: edge
  platform:
    aws:
      zones:
      - us-west-2-lax-1a
      - us-west-2-lax-1b
platform:
  aws:
    region: us-west-2
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...
```

### Additional resources

- For more information about the maximum supported maximum transmission unit (MTU) value, see AWS resources supported in Local Zones in the AWS documentation.

## 3.11.5. Cluster installation options for an AWS Local Zones environment

Choose one of the following installation options to install an OpenShift Container Platform cluster on AWS with edge compute nodes defined in Local Zones:

- Fully automated option: Installing a cluster to quickly extend compute nodes to edge compute pools, where the installation program automatically creates infrastructure resources for the OpenShift Container Platform cluster.
- Existing VPC option: Installing a cluster on AWS into an existing VPC, where you supply Local Zones subnets to the **install-config.yaml** file.

### Next steps

Choose one of the following options to install an OpenShift Container Platform cluster in an AWS Local Zones environment:

- Installing a cluster quickly in AWS Local Zones
- Installing a cluster in an existing VPC with defined AWS Local Zone subnets

### 3.11.6. Install a cluster quickly in AWS Local Zones

For OpenShift Container Platform 4.19, you can quickly install a cluster on Amazon Web Services (AWS) to extend compute nodes to Local Zones locations. By using this installation route, the installation program automatically creates network resources and Local Zones subnets for each zone that you defined in your configuration file. To customize the installation, you must modify parameters in the **install-config.yaml** file before you deploy the cluster.

#### 3.11.6.1. Modifying an installation configuration file to use AWS Local Zones

Modify an **install-config.yaml** file to include AWS Local Zones.

**Prerequisites**

- You have configured an AWS account.
- You added your AWS keys and AWS Region to your local AWS profile by running **aws configure**.
- You are familiar with the configuration limitations that apply when you specify the installation program to automatically create subnets for your OpenShift Container Platform cluster.
- You opted in to the Local Zones group for each zone.
- You created an **install-config.yaml** file by using the procedure "Creating the installation configuration file".

**Procedure**

1. Modify the **install-config.yaml** file by specifying Local Zones names in the **platform.aws.zones** property of the edge compute pool.

```
# ...
platform:
  aws:
    region: <region_name> 1
compute:
- name: edge
  platform:
    aws:
      zones: 2
      - <local_zone_name>
#...
```

1 The AWS Region name.

2 The list of Local Zones names that you use must exist in the same AWS Region specified in the **platform.aws.region** field.

**Example of a configuration to install a cluster in the us-west-2 AWS Region that extends edge nodes to Local Zones in Los Angeles and Las Vegas locations**

```
apiVersion: v1
baseDomain: example.com
metadata:
  name: cluster-name
platform:
  aws:
    region: us-west-2
compute:
- name: edge
  platform:
    aws:
      zones:
      - us-west-2-lax-1a
      - us-west-2-lax-1b
      - us-west-2-las-1a
pullSecret: '{"auths": ...}'
sshKey: 'ssh-ed25519 AAAA...'
#...
```

2. Deploy your cluster.

### Additional resources

- Creating the installation configuration file
- Cluster limitations in AWS Local Zones

### Next steps

- Deploying the cluster

## 3.11.7. Installing a cluster in an existing VPC that has Local Zone subnets

You can install a cluster into an existing Amazon Virtual Private Cloud (VPC) on Amazon Web Services (AWS). The installation program provisions the rest of the required infrastructure, which you can further customize. To customize the installation, change parameters in the **install-config.yaml** file before you install the cluster.

Installing a cluster on AWS into an existing VPC requires extending compute nodes to the edge of the Cloud Infrastructure by using AWS Local Zones.

Local Zone subnets extend regular compute nodes to edge networks. Each edge compute nodes runs a user workload. After you create an Amazon Web Service (AWS) Local Zone environment, and you deploy your cluster, you can use edge compute nodes to create user workloads in Local Zone subnets.

NOTE

If you want to create private subnets, you must either change the provided CloudFormation template or create your own template.

You can use a provided CloudFormation template to create network resources. Additionally, you can change a template to customize your infrastructure or use the information that they contain to create AWS resources according to your company's policies.

IMPORTANT

The documentation provides the steps for performing an installer-provisioned infrastructure installation for example purposes only. Installing a cluster in an existing VPC requires that you have knowledge of the cloud provider and the installation process of OpenShift Container Platform. You can use a CloudFormation template to assist you with completing these steps or to help model your own cluster installation. Instead of using the CloudFormation template to create resources, you can decide to use other methods for generating these resources.

### 3.11.7.1. Creating a VPC in AWS

You can create a Virtual Private Cloud (VPC), and subnets for all Local Zones locations, in Amazon Web Services (AWS) for your OpenShift Container Platform cluster to extend compute nodes to edge locations. You can further customize your VPC to meet your requirements, including a VPN and route tables. You can also add new Local Zones subnets not included at initial deployment.

You can use the provided CloudFormation template and a custom parameter file to create a stack of AWS resources that represent the VPC.

NOTE

If you do not use the provided CloudFormation template to create your AWS infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

#### Prerequisites

- You configured an AWS account.
- You added your AWS keys and AWS Region to your local AWS profile by running **aws configure**.
- You opted in to the AWS Local Zones on your AWS account.

#### Procedure

1. Create a JSON file that contains the parameter values that the CloudFormation template requires:

```
[
  {
    "ParameterKey": "VpcCidr", (1)
    "ParameterValue": "10.0.0.0/16" (2)
  },
  {
    "ParameterKey": "AvailabilityZoneCount", (3)
    "ParameterValue": "3" (4)
  },
  {
   "ParameterKey": "SubnetBits", 5
   "ParameterValue": "12" 6
  }
]
```

1 The CIDR block for the VPC.

2 Specify a CIDR block in the format **x.x.x.x/16-24**.

3 The number of availability zones to deploy the VPC in.

4 Specify an integer between **1** and **3**.

5 The size of each subnet in each availability zone.

6 Specify an integer between **5** and **13**, where **5** is **/27** and **13** is **/19**.

2. Go to the section of the documentation named "CloudFormation template for the VPC", and then copy the syntax from the provided template. Save the copied template syntax as a YAML file on your local system. This template describes the VPC that your cluster requires.

3. Launch the CloudFormation template to create a stack of AWS resources that represent the VPC by running the following command:

**IMPORTANT**

You must enter the command on a single line.

```
$ aws cloudformation create-stack --stack-name <name> \ 1
     --template-body file://<template>.yaml \ 2
     --parameters file://<parameters>.json 3
```

1 **<name>** is the name for the CloudFormation stack, such as **cluster-vpc**. You need the name of this stack if you remove the cluster.

2 **<template>** is the relative path to and name of the CloudFormation template YAML file that you saved.

3 **<parameters>** is the relative path and the name of the CloudFormation parameters JSON file.

### Example output

```
arn:aws:cloudformation:us-east-1:123456789012:stack/cluster-vpc/dbedae40-2fd3-11eb-
820e-12a48460849f
```

4. Confirm that the template components exist by running the following command:

```
$ aws cloudformation describe-stacks --stack-name <name>
```

After the **StackStatus** displays **CREATE_COMPLETE**, the output displays values for the following parameters. You must provide these parameter values to the other CloudFormation templates that you run to create your cluster.

| | |
|---|---|
| **VpcId** | The ID of your VPC. |
| **PublicSubnetIds** | The IDs of the new public subnets. |
| **PrivateSubnetIds** | The IDs of the new private subnets. |
| **PublicRouteTableId** | The ID of the new public route table ID. |

### 3.11.7.2. CloudFormation template for the VPC

You can use the following CloudFormation template to deploy the VPC that you need for your OpenShift Container Platform cluster.

**Example 3.16. CloudFormation template for the VPC**

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for Best Practice VPC with 1-3 AZs

Parameters:
  VpcCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-
4][0-9]|25[0-5])(\/(1[6-9]|2[0-4]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/16-24.
    Default: 10.0.0.0/16
    Description: CIDR block for VPC.
    Type: String
  AvailabilityZoneCount:
    ConstraintDescription: "The number of availability zones. (Min: 1, Max: 3)"
    MinValue: 1
    MaxValue: 3
    Default: 1
    Description: "How many AZs to create VPC subnets for. (Min: 1, Max: 3)"
    Type: Number
  SubnetBits:
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/19-27.
    MinValue: 5
    MaxValue: 13
    Default: 12
    Description: "Size of each subnet to create within the availability zones. (Min: 5 = /27, Max: 13 =
/19)"
    Type: Number

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcCidr
      - SubnetBits
    - Label:
        default: "Availability Zones"
      Parameters:
      - AvailabilityZoneCount
    ParameterLabels:
      AvailabilityZoneCount:
        default: "Availability Zone Count"
      VpcCidr:
        default: "VPC CIDR"
      SubnetBits:
        default: "Bits Per Subnet"

Conditions:
  DoAz3: !Equals [3, !Ref AvailabilityZoneCount]
  DoAz2: !Or [!Equals [2, !Ref AvailabilityZoneCount], Condition: DoAz3]

Resources:
  VPC:
    Type: "AWS::EC2::VPC"
    Properties:
      EnableDnsSupport: "true"
      EnableDnsHostnames: "true"
      CidrBlock: !Ref VpcCidr
  PublicSubnet:
    Type: "AWS::EC2::Subnet"
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [0, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 0
      - Fn::GetAZs: !Ref "AWS::Region"
  PublicSubnet2:
    Type: "AWS::EC2::Subnet"
    Condition: DoAz2
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [1, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 1
      - Fn::GetAZs: !Ref "AWS::Region"
  PublicSubnet3:
    Type: "AWS::EC2::Subnet"
    Condition: DoAz3
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [2, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 2
      - Fn::GetAZs: !Ref "AWS::Region"
  InternetGateway:
    Type: "AWS::EC2::InternetGateway"
  GatewayToInternet:
    Type: "AWS::EC2::VPCGatewayAttachment"
    Properties:
      VpcId: !Ref VPC
      InternetGatewayId: !Ref InternetGateway
  PublicRouteTable:
    Type: "AWS::EC2::RouteTable"
    Properties:
      VpcId: !Ref VPC
  PublicRoute:
    Type: "AWS::EC2::Route"
    DependsOn: GatewayToInternet
    Properties:
      RouteTableId: !Ref PublicRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId: !Ref InternetGateway
  PublicSubnetRouteTableAssociation:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Properties:
      SubnetId: !Ref PublicSubnet
      RouteTableId: !Ref PublicRouteTable
  PublicSubnetRouteTableAssociation2:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Condition: DoAz2
    Properties:
      SubnetId: !Ref PublicSubnet2
      RouteTableId: !Ref PublicRouteTable
  PublicSubnetRouteTableAssociation3:
    Condition: DoAz3
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Properties:
      SubnetId: !Ref PublicSubnet3
      RouteTableId: !Ref PublicRouteTable
  PrivateSubnet:
    Type: "AWS::EC2::Subnet"
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [3, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 0
      - Fn::GetAZs: !Ref "AWS::Region"
  PrivateRouteTable:
    Type: "AWS::EC2::RouteTable"
    Properties:
      VpcId: !Ref VPC
  PrivateSubnetRouteTableAssociation:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Properties:
      SubnetId: !Ref PrivateSubnet
      RouteTableId: !Ref PrivateRouteTable
  NAT:
    DependsOn:
    - GatewayToInternet
    Type: "AWS::EC2::NatGateway"
    Properties:
      AllocationId:
        "Fn::GetAtt":
```

```
        - EIP
        - AllocationId
      SubnetId: !Ref PublicSubnet
  EIP:
    Type: "AWS::EC2::EIP"
    Properties:
      Domain: vpc
  Route:
    Type: "AWS::EC2::Route"
    Properties:
      RouteTableId:
        Ref: PrivateRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId:
        Ref: NAT
  PrivateSubnet2:
    Type: "AWS::EC2::Subnet"
    Condition: DoAz2
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [4, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 1
      - Fn::GetAZs: !Ref "AWS::Region"
  PrivateRouteTable2:
    Type: "AWS::EC2::RouteTable"
    Condition: DoAz2
    Properties:
      VpcId: !Ref VPC
  PrivateSubnetRouteTableAssociation2:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Condition: DoAz2
    Properties:
      SubnetId: !Ref PrivateSubnet2
      RouteTableId: !Ref PrivateRouteTable2
  NAT2:
    DependsOn:
    - GatewayToInternet
    Type: "AWS::EC2::NatGateway"
    Condition: DoAz2
    Properties:
      AllocationId:
        "Fn::GetAtt":
        - EIP2
        - AllocationId
      SubnetId: !Ref PublicSubnet2
  EIP2:
    Type: "AWS::EC2::EIP"
    Condition: DoAz2
    Properties:
      Domain: vpc
  Route2:
    Type: "AWS::EC2::Route"
    Condition: DoAz2
    Properties:
      RouteTableId:
```

```
      Ref: PrivateRouteTable2
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId:
      Ref: NAT2
PrivateSubnet3:
  Type: "AWS::EC2::Subnet"
  Condition: DoAz3
  Properties:
    VpcId: !Ref VPC
    CidrBlock: !Select [5, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
    AvailabilityZone: !Select
    - 2
    - Fn::GetAZs: !Ref "AWS::Region"
PrivateRouteTable3:
  Type: "AWS::EC2::RouteTable"
  Condition: DoAz3
  Properties:
    VpcId: !Ref VPC
PrivateSubnetRouteTableAssociation3:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Condition: DoAz3
  Properties:
    SubnetId: !Ref PrivateSubnet3
    RouteTableId: !Ref PrivateRouteTable3
NAT3:
  DependsOn:
  - GatewayToInternet
  Type: "AWS::EC2::NatGateway"
  Condition: DoAz3
  Properties:
    AllocationId:
      "Fn::GetAtt":
      - EIP3
      - AllocationId
    SubnetId: !Ref PublicSubnet3
EIP3:
  Type: "AWS::EC2::EIP"
  Condition: DoAz3
  Properties:
    Domain: vpc
Route3:
  Type: "AWS::EC2::Route"
  Condition: DoAz3
  Properties:
    RouteTableId:
      Ref: PrivateRouteTable3
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId:
      Ref: NAT3
S3Endpoint:
  Type: AWS::EC2::VPCEndpoint
  Properties:
    PolicyDocument:
      Version: 2012-10-17
      Statement:
      - Effect: Allow
```

```
            Principal: '*'
            Action:
            - '*'
            Resource:
            - '*'
          RouteTableIds:
          - !Ref PublicRouteTable
          - !Ref PrivateRouteTable
          - !If [DoAz2, !Ref PrivateRouteTable2, !Ref "AWS::NoValue"]
          - !If [DoAz3, !Ref PrivateRouteTable3, !Ref "AWS::NoValue"]
          ServiceName: !Join
          - ''
          - - com.amazonaws.
            - !Ref 'AWS::Region'
            - .s3
          VpcId: !Ref VPC

Outputs:
  VpcId:
    Description: ID of the new VPC.
    Value: !Ref VPC
  PublicSubnetIds:
    Description: Subnet IDs of the public subnets.
    Value:
      !Join [
        ",",
        [!Ref PublicSubnet, !If [DoAz2, !Ref PublicSubnet2, !Ref "AWS::NoValue"], !If [DoAz3, !Ref
PublicSubnet3, !Ref "AWS::NoValue"]]
      ]
  PrivateSubnetIds:
    Description: Subnet IDs of the private subnets.
    Value:
      !Join [
        ",",
        [!Ref PrivateSubnet, !If [DoAz2, !Ref PrivateSubnet2, !Ref "AWS::NoValue"], !If [DoAz3, !Ref
PrivateSubnet3, !Ref "AWS::NoValue"]]
      ]
  PublicRouteTableId:
    Description: Public Route table ID
    Value: !Ref PublicRouteTable
  PrivateRouteTableIds:
    Description: Private Route table IDs
    Value:
      !Join [
        ",",
        [
          !Join ["=", [
            !Select [0, "Fn::GetAZs": !Ref "AWS::Region"],
            !Ref PrivateRouteTable
          ]],
          !If [DoAz2,
               !Join ["=", [!Select [1, "Fn::GetAZs": !Ref "AWS::Region"], !Ref PrivateRouteTable2]],
               !Ref "AWS::NoValue"
          ],
          !If [DoAz3,
               !Join ["=", [!Select [2, "Fn::GetAZs": !Ref "AWS::Region"], !Ref PrivateRouteTable3]],
              !Ref "AWS::NoValue"
            ]
          ]
        ]
```

### 3.11.7.3. Creating subnets in Local Zones

Before you configure a machine set for edge compute nodes in your OpenShift Container Platform cluster, you must create the subnets in Local Zones. Complete the following procedure for each Local Zone that you want to deploy compute nodes to.

You can use the provided CloudFormation template and create a CloudFormation stack. You can then use this stack to custom provision a subnet.

NOTE

If you do not use the provided CloudFormation template to create your AWS infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

**Prerequisites**

- You configured an AWS account.
- You added your AWS keys and region to your local AWS profile by running **aws configure**.
- You opted in to the Local Zones group.

**Procedure**

1. Go to the section of the documentation named "CloudFormation template for the VPC subnet", and copy the syntax from the template. Save the copied template syntax as a YAML file on your local system. This template describes the VPC that your cluster requires.

2. Run the following command to deploy the CloudFormation template, which creates a stack of AWS resources that represent the VPC:

```
$ aws cloudformation create-stack --stack-name <stack_name> \ 1
  --region ${CLUSTER_REGION} \
  --template-body file://<template>.yaml \ 2
  --parameters \
    ParameterKey=VpcId,ParameterValue="${VPC_ID}" \ 3
    ParameterKey=ClusterName,ParameterValue="${CLUSTER_NAME}" \ 4
    ParameterKey=ZoneName,ParameterValue="${ZONE_NAME}" \ 5
    ParameterKey=PublicRouteTableId,ParameterValue="${ROUTE_TABLE_PUB}" \ 6
    ParameterKey=PublicSubnetCidr,ParameterValue="${SUBNET_CIDR_PUB}" \ 7
    ParameterKey=PrivateRouteTableId,ParameterValue="${ROUTE_TABLE_PVT}" \ 8
    ParameterKey=PrivateSubnetCidr,ParameterValue="${SUBNET_CIDR_PVT}" 9
```

1 **<stack_name>** is the name for the CloudFormation stack, such as **cluster-wl-<local_zone_shortname>**. You need the name of this stack if you remove the cluster.

2 **<template>** is the relative path and the name of the CloudFormation template YAML file that you saved.

3 **${VPC_ID}** is the VPC ID, which is the value **VpcID** in the output of the CloudFormation template for the VPC.

4 **${ZONE_NAME}** is the value of Local Zones name to create the subnets.

5 **${CLUSTER_NAME}** is the value of **ClusterName** to be used as a prefix of the new AWS resource names.

6 **${SUBNET_CIDR_PUB}** is a valid CIDR block that is used to create the public subnet. This block must be part of the VPC CIDR block **VpcCidr**.

7 **${ROUTE_TABLE_PVT}** is the **PrivateRouteTableId** extracted from the output of the VPC's CloudFormation stack.

8 **${SUBNET_CIDR_PVT}** is a valid CIDR block that is used to create the private subnet. This block must be part of the VPC CIDR block **VpcCidr**.

### Example output

```
arn:aws:cloudformation:us-east-1:123456789012:stack/<stack_name>/dbedae40-820e-11eb-2fd3-
12a48460849f
```

### Verification

- Confirm that the template components exist by running the following command:

  ```
  $ aws cloudformation describe-stacks --stack-name <stack_name>
  ```

  After the **StackStatus** displays **CREATE_COMPLETE**, the output displays values for the following parameters. Ensure that you provide these parameter values to the other CloudFormation templates that you run to create for your cluster.

| | |
|---|---|
| **PublicSubnetId** | The IDs of the public subnet created by the CloudFormation stack. |
| **PrivateSubnetId** | The IDs of the private subnet created by the CloudFormation stack. |

## 3.11.7.4. CloudFormation template for the VPC subnet

You can use the following CloudFormation template to deploy the private and public subnets in a zone on Local Zones infrastructure.

**Example 3.17. CloudFormation template for VPC subnets**

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for Best Practice Subnets (Public and Private)
```

```
Parameters:
  VpcId:
    Description: VPC ID that comprises all the target subnets.
    Type: String
    AllowedPattern: ^(?:(?:vpc)(?:-[a-zA-Z0-9]+)?\b|(?:[0-9]{1,3}\.){3}[0-9]{1,3})$
    ConstraintDescription: VPC ID must be with valid name, starting with vpc-.*.
  ClusterName:
    Description: Cluster name or prefix name to prepend the Name tag for each subnet.
    Type: String
    AllowedPattern: ".+"
    ConstraintDescription: ClusterName parameter must be specified.
  ZoneName:
    Description: Zone Name to create the subnets, such as us-west-2-lax-1a.
    Type: String
    AllowedPattern: ".+"
    ConstraintDescription: ZoneName parameter must be specified.
  PublicRouteTableId:
    Description: Public Route Table ID to associate the public subnet.
    Type: String
    AllowedPattern: ".+"
    ConstraintDescription: PublicRouteTableId parameter must be specified.
  PublicSubnetCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-
4][0-9]|25[0-5])(\/(1[6-9]|2[0-4]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/16-24.
    Default: 10.0.128.0/20
    Description: CIDR block for public subnet.
    Type: String
  PrivateRouteTableId:
    Description: Private Route Table ID to associate the private subnet.
    Type: String
    AllowedPattern: ".+"
    ConstraintDescription: PrivateRouteTableId parameter must be specified.
  PrivateSubnetCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-
4][0-9]|25[0-5])(\/(1[6-9]|2[0-4]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/16-24.
    Default: 10.0.128.0/20
    Description: CIDR block for private subnet.
    Type: String


Resources:
  PublicSubnet:
    Type: "AWS::EC2::Subnet"
    Properties:
      VpcId: !Ref VpcId
      CidrBlock: !Ref PublicSubnetCidr
      AvailabilityZone: !Ref ZoneName
      Tags:
      - Key: Name
        Value: !Join ['-', [!Ref ClusterName, "public", !Ref ZoneName]]

  PublicSubnetRouteTableAssociation:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Properties:
      SubnetId: !Ref PublicSubnet
      RouteTableId: !Ref PublicRouteTableId

  PrivateSubnet:
    Type: "AWS::EC2::Subnet"
    Properties:
      VpcId: !Ref VpcId
      CidrBlock: !Ref PrivateSubnetCidr
      AvailabilityZone: !Ref ZoneName
      Tags:
      - Key: Name
        Value: !Join ['-', [!Ref ClusterName, "private", !Ref ZoneName]]

  PrivateSubnetRouteTableAssociation:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Properties:
      SubnetId: !Ref PrivateSubnet
      RouteTableId: !Ref PrivateRouteTableId

Outputs:
  PublicSubnetId:
    Description: Subnet ID of the public subnets.
    Value:
      !Join ["", [!Ref PublicSubnet]]

  PrivateSubnetId:
    Description: Subnet ID of the private subnets.
    Value:
      !Join ["", [!Ref PrivateSubnet]]
```

#### Additional resources

- You can view details about the CloudFormation stacks that you create by navigating to the AWS CloudFormation console.

### 3.11.7.5. Modifying an installation configuration file to use AWS Local Zones subnets

Modify your **install-config.yaml** file to include Local Zones subnets.

#### Prerequisites

- You created subnets by using the procedure "Creating subnets in Local Zones".
- You created an **install-config.yaml** file by using the procedure "Creating the installation configuration file".

#### Procedure

- Modify the **install-config.yaml** configuration file by specifying Local Zones subnets in the **platform.aws.subnets** parameter.

  **Example installation configuration file with Local Zones subnets**

```
# ...
platform:
  aws:
    region: us-west-2
    subnets: 1
    - publicSubnetId-1
    - publicSubnetId-2
    - publicSubnetId-3
    - privateSubnetId-1
    - privateSubnetId-2
    - privateSubnetId-3
    - publicSubnetId-LocalZone-1
# ...
```

1 List of subnet IDs created in the zones: Availability and Local Zones.

**Additional resources**

- For more information about viewing the CloudFormation stacks that you created, see AWS CloudFormation console.
- For more information about AWS profile and credential configuration, see Configuration and credential file settings in the AWS documentation.

**Next steps**

- Deploying the cluster

### 3.11.8. Optional: AWS security groups

By default, the installation program creates and attaches security groups to control plane and compute machines. The rules associated with the default security groups cannot be modified.

However, you can apply additional existing AWS security groups, which are associated with your existing VPC, to control plane and compute machines. Applying custom security groups can help you meet the security needs of your organization, in such cases where you need to control the incoming or outgoing traffic of these machines.

As part of the installation process, you apply custom security groups by modifying the **install-config.yaml** file before deploying the cluster.

For more information, see "Edge compute pools and AWS Local Zones".

### 3.11.9. Optional: Assign public IP addresses to edge compute nodes

If your workload requires deploying the edge compute nodes in public subnets on Local Zones infrastructure, you can configure the machine set manifests when installing a cluster.

AWS Local Zones infrastructure accesses the network traffic in a specified zone, so applications can take advantage of lower latency when serving end users that are closer to that zone.

The default setting that deploys compute nodes in private subnets might not meet your needs, so consider creating edge compute nodes in public subnets when you want to apply more customization to your infrastructure.

IMPORTANT

By default, OpenShift Container Platform deploy the compute nodes in private subnets. For best performance, consider placing compute nodes in subnets that have their Public IP addresses attached to the subnets.

You must create additional security groups, but ensure that you only open the groups' rules over the internet when you really need to.

**Procedure**

1. Change to the directory that contains the installation program and generate the manifest files. Ensure that the installation manifests get created at the **openshift** and **manifests** directory level.

```
$ ./openshift-install create manifests --dir <installation_directory>
```

2. Edit the machine set manifest that the installation program generates for the Local Zones, so that the manifest gets deployed in public subnets. Specify **true** for the **spec.template.spec.providerSpec.value.publicIP** parameter.

#### Example machine set manifest configuration for installing a cluster quickly in Local Zones

```
spec:
  template:
    spec:
      providerSpec:
        value:
          publicIp: true
          subnet:
            filters:
              - name: tag:Name
                values:
                  - ${INFRA_ID}-public-${ZONE_NAME}
```

#### Example machine set manifest configuration for installing a cluster in an existing VPC that has Local Zones subnets

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  name: <infrastructure_id>-edge-<zone>
  namespace: openshift-machine-api
spec:
  template:
    spec:
      providerSpec:
        value:
          publicIp: true
```

### 3.11.10. Deploying the cluster

You can install OpenShift Container Platform on a compatible cloud platform.

IMPORTANT

You can run the **create cluster** command of the installation program only once, during initial installation.

## Prerequisites

- You have configured an account with the cloud platform that hosts your cluster.
- You have the OpenShift Container Platform installation program and the pull secret for your cluster.
- You have verified that the cloud provider account on your host has the correct permissions to deploy the cluster. An account with incorrect permissions causes the installation process to fail with an error message that displays the missing permissions.

## Procedure

1. In the directory that contains the installation program, initialize the cluster deployment by running the following command:

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 For **<installation_directory>**, specify the location of your customized **./install-config.yaml** file.

2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

2. Optional: Remove or disable the **AdministratorAccess** policy from the IAM account that you used to install the cluster.

NOTE

The elevated permissions provided by the **AdministratorAccess** policy are required only during installation.

## Verification

When the cluster deployment completes successfully:

- The terminal displays directions for accessing your cluster, including a link to the web console and credentials for the **kubeadmin** user.
- Credential information also outputs to **<installation_directory>/.openshift_install.log**.

IMPORTANT

Do not delete the installation program or the files that the installation program creates. Both are required to delete the cluster.

## Example output

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```

IMPORTANT

- The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

- It is recommended that you use Ignition config files within 12 hours after they are generated because the 24-hour certificate rotates from 16 to 22 hours after the cluster is installed. By using the Ignition config files within 12 hours, you can avoid installation failure if the certificate update runs during installation.

## 3.11.11. Verifying the status of the deployed cluster

Verify that your OpenShift Container Platform successfully deployed on AWS Local Zones.

### 3.11.11.1. Logging in to the cluster by using the CLI

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

#### Prerequisites

- You deployed an OpenShift Container Platform cluster.

- You installed the OpenShift CLI (**oc**).

#### Procedure

1. Export the **kubeadmin** credentials by running the following command:

   ```
   $ export KUBECONFIG=<installation_directory>/auth/kubeconfig (1)
   ```

   (1) For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration by running the following command:

```
$ oc whoami
```

**Example output**

```
system:admin
```

#### 3.11.11.2. Logging in to the cluster by using the web console

The **kubeadmin** user exists by default after an OpenShift Container Platform installation. You can log in to your cluster as the **kubeadmin** user by using the OpenShift Container Platform web console.

**Prerequisites**

- You have access to the installation host.
- You completed a cluster installation and all cluster Operators are available.

**Procedure**

1. Obtain the password for the **kubeadmin** user from the **kubeadmin-password** file on the installation host:

```
$ cat <installation_directory>/auth/kubeadmin-password
```

NOTE

Alternatively, you can obtain the **kubeadmin** password from the **<installation_directory>/.openshift_install.log** log file on the installation host.

2. List the OpenShift Container Platform web console route:

```
$ oc get routes -n openshift-console | grep 'console-openshift'
```

NOTE

Alternatively, you can obtain the OpenShift Container Platform route from the **<installation_directory>/.openshift_install.log** log file on the installation host.

**Example output**

```
console    console-openshift-console.apps.<cluster_name>.<base_domain>          console
https   reencrypt/Redirect   None
```

3. Navigate to the route detailed in the output of the preceding command in a web browser and log in as the **kubeadmin** user.

**Additional resources**

- Accessing the web console

### 3.11.11.3. Verifying nodes that were created with edge compute pool

After you install a cluster that uses AWS Local Zones infrastructure, check the status of the machine that was created by the machine set manifests created during installation.

1. To check the machine sets created from the subnet you added to the **install-config.yaml** file, run the following command:

   ```
   $ oc get machineset -n openshift-machine-api
   ```

   **Example output**

   ```
   NAME                              DESIRED   CURRENT   READY   AVAILABLE   AGE
   cluster-7xw5g-edge-us-east-1-nyc-1a   1         1         1       1           3h4m
   cluster-7xw5g-worker-us-east-1a       1         1         1       1           3h4m
   cluster-7xw5g-worker-us-east-1b       1         1         1       1           3h4m
   cluster-7xw5g-worker-us-east-1c       1         1         1       1           3h4m
   ```

2. To check the machines that were created from the machine sets, run the following command:

   ```
   $ oc get machines -n openshift-machine-api
   ```

   **Example output**

   ```
   NAME                                      PHASE     TYPE          REGION      ZONE               AGE
   cluster-7xw5g-edge-us-east-1-nyc-1a-wbclh   Running   c5d.2xlarge   us-east-1   us-east-1-
   nyc-1a   3h
   cluster-7xw5g-master-0                    Running   m6i.xlarge    us-east-1   us-east-1a         3h4m
   cluster-7xw5g-master-1                    Running   m6i.xlarge    us-east-1   us-east-1b         3h4m
   cluster-7xw5g-master-2                    Running   m6i.xlarge    us-east-1   us-east-1c         3h4m
   cluster-7xw5g-worker-us-east-1a-rtp45       Running   m6i.xlarge    us-east-1   us-east-1a
   3h
   cluster-7xw5g-worker-us-east-1b-glm7c       Running   m6i.xlarge    us-east-1   us-east-1b
   3h
   cluster-7xw5g-worker-us-east-1c-qfvz4       Running   m6i.xlarge    us-east-1   us-east-1c
   3h
   ```

3. To check nodes with edge roles, run the following command:

   ```
   $ oc get nodes -l node-role.kubernetes.io/edge
   ```

   **Example output**

   ```
   NAME                          STATUS   ROLES         AGE    VERSION
   ip-10-0-207-188.ec2.internal   Ready    edge,worker   172m   v1.25.2+d2e245f
   ```

**Next steps**

- Validating an installation.
- If necessary, you can Remote health reporting.

## 3.12. INSTALLING A CLUSTER WITH COMPUTE NODES ON AWS WAVELENGTH ZONES

You can quickly install an OpenShift Container Platform cluster on Amazon Web Services (AWS) Wavelength Zones by setting the zone names in the edge compute pool of the **install-config.yaml** file, or install a cluster in an existing Amazon Virtual Private Cloud (VPC) with Wavelength Zone subnets.

AWS Wavelength Zones is an infrastructure that AWS configured for mobile edge computing (MEC) applications.

A Wavelength Zone embeds AWS compute and storage services within the 5G network of a communication service provider (CSP). By placing application servers in a Wavelength Zone, the application traffic from your 5G devices can stay in the 5G network. The application traffic of the device reaches the target server directly, making latency a non-issue.

**Additional resources**

- Wavelength Zones(AWS documentation)

### 3.12.1. Infrastructure prerequisites

- You reviewed details about OpenShift Container Platform installation and update processes.
- You are familiar with Selecting a cluster installation method and preparing it for users .
- You configured an AWS account to host the cluster.

> **WARNING**
>
> If you have an AWS profile stored on your computer, it must not use a temporary session token that you generated while using a multi-factor authentication device. The cluster continues to use your current AWS credentials to create AWS resources for the entire life of the cluster, so you must use key-based, long-term credentials. To generate appropriate keys, see Managing Access Keys for IAM Users in the AWS documentation. You can supply the keys when you run the installation program.

- You downloaded the AWS CLI and installed it on your computer. See Install the AWS CLI Using the Bundled Installer (Linux, macOS, or UNIX) in the AWS documentation.
- If you use a firewall, you configured it to allow the sites that your cluster must access.
- You noted the region and supported AWS Wavelength Zone locations to create the network resources in.
- You read AWS Wavelength features in the AWS documentation.
- You read the Quotas and considerations for Wavelength Zones in the AWS documentation.

- You added permissions for creating network resources that support AWS Wavelength Zones to the Identity and Access Management (IAM) user or role. For example:

  **Example of an additional IAM policy that attached `ec2:ModifyAvailabilityZoneGroup`, `ec2:CreateCarrierGateway`, and `ec2:DeleteCarrierGateway` permissions to a user or role**

  ```
  {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": [
          "ec2:DeleteCarrierGateway",
          "ec2:CreateCarrierGateway"
        ],
        "Resource": "*"
      },
      {
        "Action": [
          "ec2:ModifyAvailabilityZoneGroup"
        ],
        "Effect": "Allow",
        "Resource": "*"
      }
    ]
  }
  ```

## 3.12.2. About AWS Wavelength Zones and edge compute pool

Read the following sections to understand infrastructure behaviors and cluster limitations in an AWS Wavelength Zones environment.

### 3.12.2.1. Cluster limitations in AWS Wavelength Zones

Some limitations exist when you try to deploy a cluster with a default installation configuration in an Amazon Web Services (AWS) Wavelength Zone.

IMPORTANT

The following list details limitations when deploying a cluster in a pre-configured AWS zone:

- The maximum transmission unit (MTU) between an Amazon EC2 instance in a zone and an Amazon EC2 instance in the Region is **1300**. This causes the cluster-wide network MTU to change according to the network plugin that is used with the deployment.
- Network resources such as Network Load Balancer (NLB), Classic Load Balancer, and Network Address Translation (NAT) Gateways are not globally supported.
- For an OpenShift Container Platform cluster on AWS, the AWS Elastic Block Storage (EBS) **gp3** type volume is the default for node volumes and the default for the storage class. This volume type is not globally available on zone locations. By default, the nodes running in zones are deployed with the **gp2** EBS volume. The **gp2-csi StorageClass** parameter must be set when creating workloads on zone nodes.

If you want the installation program to automatically create Wavelength Zone subnets for your OpenShift Container Platform cluster, specific configuration limitations apply with this method. The following note details some of these limitations. For other limitations, ensure that you read the "Quotas and considerations for Wavelength Zones" document that Red Hat provides in the "Infrastructure prerequisites" section.

IMPORTANT

The following configuration limitation applies when you set the installation program to automatically create subnets for your OpenShift Container Platform cluster:

- When the installation program creates private subnets in AWS Wavelength Zones, the program associates each subnet with the route table of its parent zone. This operation ensures that each private subnet can route egress traffic to the internet by way of NAT Gateways in an AWS Region.
- If the parent-zone route table does not exist during cluster installation, the installation program associates any private subnet with the first available private route table in the Amazon Virtual Private Cloud (VPC). This approach is valid only for AWS Wavelength Zones subnets in an OpenShift Container Platform cluster.

#### 3.12.2.2. About edge compute pools

Edge compute nodes are tainted compute nodes that run in AWS Wavelength Zones locations.

When deploying a cluster that uses Wavelength Zones, consider the following points:

- Amazon EC2 instances in the Wavelength Zones are more expensive than Amazon EC2 instances in the Availability Zones.
- The latency is lower between the applications running in AWS Wavelength Zones and the end user. A latency impact exists for some workloads if, for example, ingress traffic is mixed between Wavelength Zones and Availability Zones.

IMPORTANT

Generally, the maximum transmission unit (MTU) between an Amazon EC2 instance in a Wavelength Zones and an Amazon EC2 instance in the Region is 1300. The cluster network MTU must be always less than the EC2 MTU to account for the overhead. The specific overhead is determined by the network plugin. For example: OVN-Kubernetes has an overhead of **100 bytes**.

The network plugin can provide additional features, such as IPsec, that also affect the MTU sizing.

For more information, see How AWS Wavelength work in the AWS documentation.

OpenShift Container Platform 4.12 introduced a new compute pool, *edge*, that is designed for use in remote zones. The edge compute pool configuration is common between AWS Wavelength Zones locations. Because of the type and size limitations of resources like EC2 and EBS on Wavelength Zones resources, the default instance type can vary from the traditional compute pool.

The default Elastic Block Store (EBS) for Wavelength Zones locations is **gp2**, which differs from the non-edge compute pool. The instance type used for each Wavelength Zones on an edge compute pool also might differ from other compute pools, depending on the instance offerings on the zone.

The edge compute pool creates new labels that developers can use to deploy applications onto AWS Wavelength Zones nodes. The new labels are:

- **node-role.kubernetes.io/edge=''**
- **machine.openshift.io/zone-type=wavelength-zone**
- **machine.openshift.io/zone-group=$ZONE_GROUP_NAME**

By default, the machine sets for the edge compute pool define the taint of **NoSchedule** to prevent other workloads from spreading on Wavelength Zones instances. Users can only run user workloads if they define tolerations in the pod specification.

### Additional resources

- MTU value selection
- Changing the MTU for the cluster network
- Understanding taints and tolerations
- Storage classes
- Ingress Controller sharding

### 3.12.3. Installation prerequisites

Before you install a cluster in an AWS Wavelength Zones environment, you must configure your infrastructure so that it can adopt Wavelength Zone capabilities.

#### 3.12.3.1. Opting in to an AWS Wavelength Zones

If you plan to create subnets in AWS Wavelength Zones, you must opt in to each zone group separately.

**Prerequisites**

- You have installed the AWS CLI.
- You have determined an AWS Region for where you want to deploy your OpenShift Container Platform cluster.
- You have attached a permissive IAM policy to a user or role account that opts in to the zone group.

**Procedure**

1. List the zones that are available in your AWS Region by running the following command:

   **Example command for listing available AWS Wavelength Zones in an AWS Region**

   ```
   $ aws --region "<value_of_AWS_Region>" ec2 describe-availability-zones \
       --query 'AvailabilityZones[].[{ZoneName: ZoneName, GroupName: GroupName, Status: OptInStatus}]' \
       --filters Name=zone-type,Values=wavelength-zone \
       --all-availability-zones
   ```

   Depending on the AWS Region, the list of available zones might be long. The command returns the following fields:

   **ZoneName**

   The name of the Wavelength Zones.

   **GroupName**

   The group that comprises the zone. To opt in to the Region, save the name.

   **Status**

   The status of the Wavelength Zones group. If the status is **not-opted-in**, you must opt in the **GroupName** as described in the next step.

2. Opt in to the zone group on your AWS account by running the following command:

   ```
   $ aws ec2 modify-availability-zone-group \
       --group-name "<value_of_GroupName>" \ (1)
       --opt-in-status opted-in
   ```

   (1) Replace **<value_of_GroupName>** with the name of the group of the Wavelength Zones where you want to create subnets. As an example for Wavelength Zones, specify **us-east-1-wl1** to use the zone **us-east-1-wl1-nyc-wlz-1** (US East New York).

### 3.12.3.2. Obtaining an AWS Marketplace image

If you are deploying an OpenShift Container Platform cluster using an AWS Marketplace image, you must first subscribe through AWS. Subscribing to the offer provides you with the AMI ID that the installation program uses to deploy compute nodes.

NOTE

You should only modify the RHCOS image for compute machines to use an AWS Marketplace image. Control plane machines and infrastructure nodes do not require an OpenShift Container Platform subscription and use the public RHCOS default image by default, which does not incur subscription costs on your AWS bill. Therefore, you should not modify the cluster default boot image or the control plane boot images. Applying the AWS Marketplace image to them will incur additional licensing costs that cannot be recovered.

**Prerequisites**

- You have an AWS account to purchase the offer. This account does not have to be the same account that is used to install the cluster.

**Procedure**

1. Complete the OpenShift Container Platform subscription from the AWS Marketplace.
2. Record the AMI ID for your specific AWS Region. As part of the installation process, you must update the **install-config.yaml** file with this value before deploying the cluster.

**Sample install-config.yaml file with AWS Marketplace compute nodes**

```
apiVersion: v1
baseDomain: example.com
compute:
- hyperthreading: Enabled
  name: worker
  platform:
    aws:
      amiID: ami-06c4d345f7c207239 (1)
      type: m5.4xlarge
  replicas: 3
metadata:
  name: test-cluster
platform:
  aws:
    region: us-east-2 (2)
sshKey: ssh-ed25519 AAAA...
pullSecret: '{"auths": ...}'
```

(1) The AMI ID from your AWS Marketplace subscription.

(2) Your AMI ID is associated with a specific AWS Region. When creating the installation configuration file, ensure that you select the same AWS Region that you specified when configuring your subscription.

## 3.12.4. Preparing for the installation

Before you extend nodes to Wavelength Zones, you must prepare certain resources for the cluster installation environment.

#### 3.12.4.1. Minimum resource requirements for cluster installation

Each created cluster must meet minimum requirements so that the cluster runs as expected.

**Table 3.19. Minimum resource requirements**

| Machine | Operating System | vCPU [1] | Virtual RAM | Storage | Input/Output Per Second (IOPS)[2] |
|---|---|---|---|---|---|
| Bootstrap | RHCOS | 4 | 16 GB | 100 GB | 300 |
| Control plane | RHCOS | 4 | 16 GB | 100 GB | 300 |
| Compute | RHCOS, RHEL 8.6 and later [3] | 2 | 8 GB | 100 GB | 300 |

1. One vCPU is equivalent to one physical core when simultaneous multithreading (SMT), or Hyper-Threading, is not enabled. When enabled, use the following formula to calculate the corresponding ratio: (threads per core × cores) × sockets = vCPUs.

2. OpenShift Container Platform and Kubernetes are sensitive to disk performance, and faster storage is recommended, particularly for etcd on the control plane nodes which require a 10 ms p99 fsync duration. Note that on many cloud platforms, storage size and IOPS scale together, so you might need to over-allocate storage volume to obtain sufficient performance.

3. As with all user-provisioned installations, if you choose to use RHEL compute machines in your cluster, you take responsibility for all operating system life cycle management and maintenance, including performing system updates, applying patches, and completing all other required tasks. Use of RHEL 7 compute machines is deprecated and has been removed in OpenShift Container Platform 4.10 and later.

**NOTE**

For OpenShift Container Platform version 4.19, RHCOS is based on RHEL version 9.6, which updates the micro-architecture requirements. The following list contains the minimum instruction set architectures (ISA) that each architecture requires:

- x86-64 architecture requires x86-64-v2 ISA
- ARM64 architecture requires ARMv8.0-A ISA
- IBM Power architecture requires Power 9 ISA
- s390x architecture requires z14 ISA

For more information, see Architectures (RHEL documentation).

If an instance type for your platform meets the minimum requirements for cluster machines, it is supported to use in OpenShift Container Platform.

#### 3.12.4.2. Tested instance types for AWS

The following Amazon Web Services (AWS) instance types have been tested with OpenShift Container Platform for use with AWS Wavelength Zones.

NOTE

Use the machine types included in the following charts for your AWS instances. If you use an instance type that is not listed in the chart, ensure that the instance size you use matches the minimum resource requirements that are listed in the section named "Minimum resource requirements for cluster installation".

**Example 3.18. Machine types based on 64-bit x86 architecture for AWS Wavelength Zones**

- **r5.***
- **t3.***

**Additional resources**

- AWS Wavelength features(AWS documentation)

### 3.12.4.3. Creating the installation configuration file

Generate and customize the installation configuration file that the installation program needs to deploy your cluster.

**Prerequisites**

- You obtained the OpenShift Container Platform installation program and the pull secret for your cluster.
- You checked that you are deploying your cluster to an AWS Region with an accompanying Red Hat Enterprise Linux CoreOS (RHCOS) AMI published by Red Hat. If you are deploying to an AWS Region that requires a custom AMI, such as an AWS GovCloud Region, you must create the **install-config.yaml** file manually.

**Procedure**

1. Create the **install-config.yaml** file.
   a. Change to the directory that contains the installation program and run the following command:

```
$ ./openshift-install create install-config --dir <installation_directory> 1
```

1 For **<installation_directory>**, specify the directory name to store the files that the installation program creates.

IMPORTANT

Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

b. At the prompts, provide the configuration details for your cloud:

i. Optional: Select an SSH key to use to access your cluster machines.

NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

ii. Select **aws** as the platform to target.

iii. If you do not have an AWS profile stored on your computer, enter the AWS access key ID and secret access key for the user that you configured to run the installation program.

NOTE

The AWS access key ID and secret access key are stored in **~/.aws/credentials** in the home directory of the current user on the installation host. You are prompted for the credentials by the installation program if the credentials for the exported profile are not present in the file. Any credentials that you provide to the installation program are stored in the file.

iv. Select the AWS Region to deploy the cluster to.

v. Select the base domain for the Route 53 service that you configured for your cluster.

vi. Enter a descriptive name for your cluster.

vii. Paste the pull secret from Red Hat OpenShift Cluster Manager .

2. Optional: Back up the **install-config.yaml** file.

IMPORTANT

The **install-config.yaml** file is consumed during the installation process. If you want to reuse the file, you must back it up now.

### 3.12.4.4. Examples of installation configuration files with edge compute pools

The following examples show **install-config.yaml** files that contain an edge machine pool configuration.

### Configuration that uses an edge pool with a custom instance type

```
apiVersion: v1
baseDomain: devcluster.openshift.com
metadata:
  name: ipi-edgezone
compute:
- name: edge
  platform:
    aws:
      type: r5.2xlarge
platform:
  aws:
    region: us-west-2
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...
```

Instance types differ between locations. To verify availability in the Wavelength Zones in which the cluster runs, see the AWS documentation.

### Configuration that uses an edge pool with custom security groups

```
apiVersion: v1
baseDomain: devcluster.openshift.com
metadata:
  name: ipi-edgezone
compute:
- name: edge
  platform:
    aws:
      additionalSecurityGroupIDs:
        - sg-1 (1)
        - sg-2
platform:
  aws:
    region: us-west-2
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...
```

(1) Specify the name of the security group as it is displayed on the Amazon EC2 console. Ensure that you include the **sg** prefix.

## 3.12.5. Cluster installation options for an AWS Wavelength Zones environment

Choose one of the following installation options to install an OpenShift Container Platform cluster on AWS with edge compute nodes defined in Wavelength Zones:

- Fully automated option: Installing a cluster to quickly extend compute nodes to edge compute pools, where the installation program automatically creates infrastructure resources for the OpenShift Container Platform cluster.
- Existing VPC option: Installing a cluster on AWS into an existing VPC, where you supply Wavelength Zones subnets to the **install-config.yaml** file.

### Next steps

Choose one of the following options to install an OpenShift Container Platform cluster in an AWS Wavelength Zones environment:

- Installing a cluster quickly in AWS Wavelength Zones
- Modifying an installation configuration file to use AWS Wavelength Zones

## 3.12.6. Install a cluster quickly in AWS Wavelength Zones

For OpenShift Container Platform 4.19, you can quickly install a cluster on Amazon Web Services (AWS) to extend compute nodes to Wavelength Zones locations. By using this installation route, the installation program automatically creates network resources and Wavelength Zones subnets for each zone that you defined in your configuration file. To customize the installation, you must modify parameters in the **install-config.yaml** file before you deploy the cluster.

### 3.12.6.1. Modifying an installation configuration file to use AWS Wavelength Zones

Modify an **install-config.yaml** file to include AWS Wavelength Zones.

#### Prerequisites

- You have configured an AWS account.
- You added your AWS keys and AWS Region to your local AWS profile by running **aws configure**.
- You are familiar with the configuration limitations that apply when you specify the installation program to automatically create subnets for your OpenShift Container Platform cluster.
- You opted in to the Wavelength Zones group for each zone.
- You created an **install-config.yaml** file by using the procedure "Creating the installation configuration file".

#### Procedure

1. Modify the **install-config.yaml** file by specifying Wavelength Zones names in the **platform.aws.zones** property of the edge compute pool.

```
# ...
platform:
  aws:
    region: <region_name> (1)
compute:
- name: edge
  platform:
    aws:
      zones: (2)
      - <wavelength_zone_name>
#...
```

(1) The AWS Region name.

2 The list of Wavelength Zones names that you use must exist in the same AWS Region specified in the **platform.aws.region** field.

**Example of a configuration to install a cluster in the us-west-2 AWS Region that extends edge nodes to Wavelength Zones in Los Angeles and Las Vegas locations**

```
apiVersion: v1
baseDomain: example.com
metadata:
  name: cluster-name
platform:
  aws:
    region: us-west-2
compute:
- name: edge
  platform:
    aws:
      zones:
      - us-west-2-wl1-lax-wlz-1
      - us-west-2-wl1-las-wlz-1
pullSecret: '{"auths": ...}'
sshKey: 'ssh-ed25519 AAAA...'
#...
```

2. Deploy your cluster.

**Additional resources**

- Creating the installation configuration file
- Cluster limitations in AWS Wavelength Zones

**Next steps**

- Deploying the cluster

## 3.12.7. Installing a cluster in an existing VPC that has Wavelength Zone subnets

You can install a cluster into an existing Amazon Virtual Private Cloud (VPC) on Amazon Web Services (AWS). The installation program provisions the rest of the required infrastructure, which you can further customize. To customize the installation, modify parameters in the **install-config.yaml** file before you install the cluster.

Installing a cluster on AWS into an existing VPC requires extending compute nodes to the edge of the Cloud Infrastructure by using AWS Wavelength Zones.

You can use a provided CloudFormation template to create network resources. Additionally, you can modify a template to customize your infrastructure or use the information that they contain to create AWS resources according to your company’s policies.

IMPORTANT

The steps for performing an installer-provisioned infrastructure installation are provided for example purposes only. Installing a cluster in an existing VPC requires that you have knowledge of the cloud provider and the installation process of OpenShift Container Platform. You can use a CloudFormation template to assist you with completing these steps or to help model your own cluster installation. Instead of using the CloudFormation template to create resources, you can decide to use other methods for generating these resources.

### 3.12.7.1. Creating a VPC in AWS

You can create a Virtual Private Cloud (VPC), and subnets for all Wavelength Zones locations, in Amazon Web Services (AWS) for your OpenShift Container Platform cluster to extend compute nodes to edge locations. You can further customize your VPC to meet your requirements, including a VPN and route tables. You can also add new Wavelength Zones subnets not included at initial deployment.

You can use the provided CloudFormation template and a custom parameter file to create a stack of AWS resources that represent the VPC.

NOTE

If you do not use the provided CloudFormation template to create your AWS infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

#### Prerequisites

- You configured an AWS account.
- You added your AWS keys and AWS Region to your local AWS profile by running **aws configure**.
- You opted in to the AWS Wavelength Zones on your AWS account.

#### Procedure

1. Create a JSON file that contains the parameter values that the CloudFormation template requires:

```
[
  {
    "ParameterKey": "VpcCidr", (1)
    "ParameterValue": "10.0.0.0/16" (2)
  },
  {
    "ParameterKey": "AvailabilityZoneCount", (3)
    "ParameterValue": "3" (4)
  },
  {
    "ParameterKey": "SubnetBits", (5)
    "ParameterValue": "12" (6)
  }
]
```

(1) The CIDR block for the VPC.

(2) Specify a CIDR block in the format **x.x.x.x/16-24**.

(3) The number of availability zones to deploy the VPC in.

(4) Specify an integer between **1** and **3**.

(5) The size of each subnet in each availability zone.

(6) Specify an integer between **5** and **13**, where **5** is **/27** and **13** is **/19**.

2. Go to the section of the documentation named "CloudFormation template for the VPC", and then copy the syntax from the provided template. Save the copied template syntax as a YAML file on your local system. This template describes the VPC that your cluster requires.

3. Launch the CloudFormation template to create a stack of AWS resources that represent the VPC by running the following command:

**IMPORTANT**

You must enter the command on a single line.

```
$ aws cloudformation create-stack --stack-name <name> \ (1)
    --template-body file://<template>.yaml \ (2)
    --parameters file://<parameters>.json (3)
```

(1) **<name>** is the name for the CloudFormation stack, such as **cluster-vpc**. You need the name of this stack if you remove the cluster.

(2) **<template>** is the relative path to and name of the CloudFormation template YAML file that you saved.

(3) **<parameters>** is the relative path and the name of the CloudFormation parameters JSON file.

### Example output

```
arn:aws:cloudformation:us-east-1:123456789012:stack/cluster-vpc/dbedae40-2fd3-11eb-
820e-12a48460849f
```

4. Confirm that the template components exist by running the following command:

```
$ aws cloudformation describe-stacks --stack-name <name>
```

After the **StackStatus** displays **CREATE_COMPLETE**, the output displays values for the following parameters. You must provide these parameter values to the other CloudFormation templates that you run to create your cluster.

| | |
|---|---|
| **VpcId** | The ID of your VPC. |
| **PublicSubnetIds** | The IDs of the new public subnets. |
| **PrivateSubnetIds** | The IDs of the new private subnets. |
| **PublicRouteTableId** | The ID of the new public route table ID. |

### 3.12.7.2. CloudFormation template for the VPC

You can use the following CloudFormation template to deploy the VPC that you need for your OpenShift Container Platform cluster.

**Example 3.19. CloudFormation template for the VPC**

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for Best Practice VPC with 1-3 AZs

Parameters:
  VpcCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-
4][0-9]|25[0-5])(\/(1[6-9]|2[0-4]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/16-24.
    Default: 10.0.0.0/16
    Description: CIDR block for VPC.
    Type: String
  AvailabilityZoneCount:
    ConstraintDescription: "The number of availability zones. (Min: 1, Max: 3)"
    MinValue: 1
    MaxValue: 3
    Default: 1
    Description: "How many AZs to create VPC subnets for. (Min: 1, Max: 3)"
    Type: Number
  SubnetBits:
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/19-27.
    MinValue: 5
    MaxValue: 13
    Default: 12
    Description: "Size of each subnet to create within the availability zones. (Min: 5 = /27, Max: 13 =
/19)"
    Type: Number

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcCidr
      - SubnetBits
    - Label:
        default: "Availability Zones"
      Parameters:
      - AvailabilityZoneCount
    ParameterLabels:
      AvailabilityZoneCount:
        default: "Availability Zone Count"
      VpcCidr:
        default: "VPC CIDR"
      SubnetBits:
        default: "Bits Per Subnet"

Conditions:
  DoAz3: !Equals [3, !Ref AvailabilityZoneCount]
  DoAz2: !Or [!Equals [2, !Ref AvailabilityZoneCount], Condition: DoAz3]

Resources:
  VPC:
    Type: "AWS::EC2::VPC"
    Properties:
      EnableDnsSupport: "true"
      EnableDnsHostnames: "true"
      CidrBlock: !Ref VpcCidr
  PublicSubnet:
    Type: "AWS::EC2::Subnet"
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [0, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 0
      - Fn::GetAZs: !Ref "AWS::Region"
  PublicSubnet2:
    Type: "AWS::EC2::Subnet"
    Condition: DoAz2
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [1, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 1
      - Fn::GetAZs: !Ref "AWS::Region"
  PublicSubnet3:
    Type: "AWS::EC2::Subnet"
    Condition: DoAz3
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [2, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 2
      - Fn::GetAZs: !Ref "AWS::Region"
  InternetGateway:
    Type: "AWS::EC2::InternetGateway"
  GatewayToInternet:
    Type: "AWS::EC2::VPCGatewayAttachment"
    Properties:
      VpcId: !Ref VPC
      InternetGatewayId: !Ref InternetGateway
```

```
PublicRouteTable:
  Type: "AWS::EC2::RouteTable"
  Properties:
    VpcId: !Ref VPC
PublicRoute:
  Type: "AWS::EC2::Route"
  DependsOn: GatewayToInternet
  Properties:
    RouteTableId: !Ref PublicRouteTable
    DestinationCidrBlock: 0.0.0.0/0
    GatewayId: !Ref InternetGateway
PublicSubnetRouteTableAssociation:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Properties:
    SubnetId: !Ref PublicSubnet
    RouteTableId: !Ref PublicRouteTable
PublicSubnetRouteTableAssociation2:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Condition: DoAz2
  Properties:
    SubnetId: !Ref PublicSubnet2
    RouteTableId: !Ref PublicRouteTable
PublicSubnetRouteTableAssociation3:
  Condition: DoAz3
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Properties:
    SubnetId: !Ref PublicSubnet3
    RouteTableId: !Ref PublicRouteTable
PrivateSubnet:
  Type: "AWS::EC2::Subnet"
  Properties:
    VpcId: !Ref VPC
    CidrBlock: !Select [3, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
    AvailabilityZone: !Select
    - 0
    - Fn::GetAZs: !Ref "AWS::Region"
PrivateRouteTable:
  Type: "AWS::EC2::RouteTable"
  Properties:
    VpcId: !Ref VPC
PrivateSubnetRouteTableAssociation:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Properties:
    SubnetId: !Ref PrivateSubnet
    RouteTableId: !Ref PrivateRouteTable
NAT:
  DependsOn:
  - GatewayToInternet
  Type: "AWS::EC2::NatGateway"
  Properties:
    AllocationId:
      "Fn::GetAtt":
      - EIP
      - AllocationId
    SubnetId: !Ref PublicSubnet
EIP:
    Type: "AWS::EC2::EIP"
    Properties:
      Domain: vpc
  Route:
    Type: "AWS::EC2::Route"
    Properties:
      RouteTableId:
        Ref: PrivateRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId:
        Ref: NAT
  PrivateSubnet2:
    Type: "AWS::EC2::Subnet"
    Condition: DoAz2
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [4, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 1
      - Fn::GetAZs: !Ref "AWS::Region"
  PrivateRouteTable2:
    Type: "AWS::EC2::RouteTable"
    Condition: DoAz2
    Properties:
      VpcId: !Ref VPC
  PrivateSubnetRouteTableAssociation2:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Condition: DoAz2
    Properties:
      SubnetId: !Ref PrivateSubnet2
      RouteTableId: !Ref PrivateRouteTable2
  NAT2:
    DependsOn:
    - GatewayToInternet
    Type: "AWS::EC2::NatGateway"
    Condition: DoAz2
    Properties:
      AllocationId:
        "Fn::GetAtt":
        - EIP2
        - AllocationId
      SubnetId: !Ref PublicSubnet2
  EIP2:
    Type: "AWS::EC2::EIP"
    Condition: DoAz2
    Properties:
      Domain: vpc
  Route2:
    Type: "AWS::EC2::Route"
    Condition: DoAz2
    Properties:
      RouteTableId:
        Ref: PrivateRouteTable2
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId:
        Ref: NAT2
```

```
PrivateSubnet3:
  Type: "AWS::EC2::Subnet"
  Condition: DoAz3
  Properties:
    VpcId: !Ref VPC
    CidrBlock: !Select [5, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
    AvailabilityZone: !Select
    - 2
    - Fn::GetAZs: !Ref "AWS::Region"
PrivateRouteTable3:
  Type: "AWS::EC2::RouteTable"
  Condition: DoAz3
  Properties:
    VpcId: !Ref VPC
PrivateSubnetRouteTableAssociation3:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Condition: DoAz3
  Properties:
    SubnetId: !Ref PrivateSubnet3
    RouteTableId: !Ref PrivateRouteTable3
NAT3:
  DependsOn:
  - GatewayToInternet
  Type: "AWS::EC2::NatGateway"
  Condition: DoAz3
  Properties:
    AllocationId:
      "Fn::GetAtt":
      - EIP3
      - AllocationId
    SubnetId: !Ref PublicSubnet3
EIP3:
  Type: "AWS::EC2::EIP"
  Condition: DoAz3
  Properties:
    Domain: vpc
Route3:
  Type: "AWS::EC2::Route"
  Condition: DoAz3
  Properties:
    RouteTableId:
      Ref: PrivateRouteTable3
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId:
      Ref: NAT3
S3Endpoint:
  Type: AWS::EC2::VPCEndpoint
  Properties:
    PolicyDocument:
      Version: 2012-10-17
      Statement:
      - Effect: Allow
        Principal: '*'
        Action:
        - '*'
        Resource:
```

```
        - '*'
      RouteTableIds:
      - !Ref PublicRouteTable
      - !Ref PrivateRouteTable
      - !If [DoAz2, !Ref PrivateRouteTable2, !Ref "AWS::NoValue"]
      - !If [DoAz3, !Ref PrivateRouteTable3, !Ref "AWS::NoValue"]
      ServiceName: !Join
      - ''
      - - com.amazonaws.
        - !Ref 'AWS::Region'
        - .s3
      VpcId: !Ref VPC

Outputs:
  VpcId:
    Description: ID of the new VPC.
    Value: !Ref VPC
  PublicSubnetIds:
    Description: Subnet IDs of the public subnets.
    Value:
      !Join [
        ",",
        [!Ref PublicSubnet, !If [DoAz2, !Ref PublicSubnet2, !Ref "AWS::NoValue"], !If [DoAz3, !Ref
PublicSubnet3, !Ref "AWS::NoValue"]]
      ]
  PrivateSubnetIds:
    Description: Subnet IDs of the private subnets.
    Value:
      !Join [
        ",",
        [!Ref PrivateSubnet, !If [DoAz2, !Ref PrivateSubnet2, !Ref "AWS::NoValue"], !If [DoAz3, !Ref
PrivateSubnet3, !Ref "AWS::NoValue"]]
      ]
  PublicRouteTableId:
    Description: Public Route table ID
    Value: !Ref PublicRouteTable
  PrivateRouteTableIds:
    Description: Private Route table IDs
    Value:
      !Join [
        ",",
        [
          !Join ["=", [
            !Select [0, "Fn::GetAZs": !Ref "AWS::Region"],
            !Ref PrivateRouteTable
          ]],
          !If [DoAz2,
               !Join ["=", [!Select [1, "Fn::GetAZs": !Ref "AWS::Region"], !Ref PrivateRouteTable2]],
               !Ref "AWS::NoValue"
          ],
          !If [DoAz3,
               !Join ["=", [!Select [2, "Fn::GetAZs": !Ref "AWS::Region"], !Ref PrivateRouteTable3]],
               !Ref "AWS::NoValue"
          ]
        ]
      ]
```

### 3.12.7.3. Creating a VPC carrier gateway

To use public subnets in your OpenShift Container Platform cluster that runs on Wavelength Zones, you must create the carrier gateway and associate the carrier gateway to the VPC. Subnets are useful for deploying load balancers or edge compute nodes.

To create edge nodes or internet-facing load balancers in Wavelength Zones locations for your OpenShift Container Platform cluster, you must create the following required network components:

- A carrier gateway that associates to the existing VPC.
- A carrier route table that lists route entries.
- A subnet that associates to the carrier route table.

Carrier gateways exist for VPCs that only contain subnets in a Wavelength Zone.

The following list explains the functions of a carrier gateway in the context of an AWS Wavelength Zones location:

- Provides connectivity between your Wavelength Zone and the carrier network, which includes any available devices from the carrier network.
- Performs Network Address Translation (NAT) functions, such as translating IP addresses that are public IP addresses stored in a network border group, from Wavelength Zones to carrier IP addresses. These translation functions apply to inbound and outbound traffic.
- Authorizes inbound traffic from a carrier network that is located in a specific location.
- Authorizes outbound traffic to a carrier network and the internet.

NOTE

No inbound connection configuration exists from the internet to a Wavelength Zone through the carrier gateway.

You can use the provided CloudFormation template to create a stack of the following AWS resources:

- One carrier gateway that associates to the VPC ID in the template.
- One public route table for the Wavelength Zone named as **<ClusterName>-public-carrier**.
- Default IPv4 route entry in the new route table that targets the carrier gateway.
- VPC gateway endpoint for an AWS Simple Storage Service (S3).

NOTE

If you do not use the provided CloudFormation template to create your AWS infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

## Prerequisites

- You configured an AWS account.
- You added your AWS keys and region to your local AWS profile by running **aws configure**.

## Procedure

1. Go to the next section of the documentation named "CloudFormation template for the VPC Carrier Gateway", and then copy the syntax from the **CloudFormation template for VPC Carrier Gateway** template. Save the copied template syntax as a YAML file on your local system. This template describes the VPC that your cluster requires.

2. Run the following command to deploy the CloudFormation template, which creates a stack of AWS resources that represent the VPC:

```
$ aws cloudformation create-stack --stack-name <stack_name> \ 1
  --region ${CLUSTER_REGION} \
  --template-body file://<template>.yaml \ 2
  --parameters \//
    ParameterKey=VpcId,ParameterValue="${VpcId}" \ 3
    ParameterKey=ClusterName,ParameterValue="${ClusterName}" 4
```

1 **<stack_name>** is the name for the CloudFormation stack, such as **clusterName-vpc-carrier-gw**. You need the name of this stack if you remove the cluster.

2 **<template>** is the relative path and the name of the CloudFormation template YAML file that you saved.

3 **<VpcId>** is the VPC ID extracted from the CloudFormation stack output created in the section named "Creating a VPC in AWS".

4 **<ClusterName>** is a custom value that prefixes to resources that the CloudFormation stack creates. You can use the same name that is defined in the **metadata.name** section of the **install-config.yaml** configuration file.

### Example output

```
arn:aws:cloudformation:us-east-1:123456789012:stack/<stack_name>/dbedae40-2fd3-11eb-
820e-12a48460849f
```

## Verification

- Confirm that the CloudFormation template components exist by running the following command:

```
$ aws cloudformation describe-stacks --stack-name <stack_name>
```

After the **StackStatus** displays **CREATE_COMPLETE**, the output displays values for the following parameter. Ensure that you provide the parameter value to the other CloudFormation templates that you run to create for your cluster.

| | |
|---|---|
| **PublicRouteTableId** | The ID of the Route Table in the Carrier infrastructure. |

### Additional resources

- See Amazon S3 in the AWS documentation.

#### 3.12.7.4. CloudFormation template for the VPC Carrier Gateway

You can use the following CloudFormation template to deploy the Carrier Gateway on AWS Wavelength infrastructure.

**Example 3.20. CloudFormation template for VPC Carrier Gateway**

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for Creating Wavelength Zone Gateway (Carrier Gateway).

Parameters:
  VpcId:
    Description: VPC ID to associate the Carrier Gateway.
    Type: String
    AllowedPattern: ^(?:(?:vpc)(?:-[a-zA-Z0-9]+)?\b|(?:[0-9]{1,3}\.){3}[0-9]{1,3})$
    ConstraintDescription: VPC ID must be with valid name, starting with vpc-.*.
  ClusterName:
    Description: Cluster Name or Prefix name to prepend the tag Name for each subnet.
    Type: String
    AllowedPattern: ".+"
    ConstraintDescription: ClusterName parameter must be specified.

Resources:
  CarrierGateway:
    Type: "AWS::EC2::CarrierGateway"
    Properties:
      VpcId: !Ref VpcId
      Tags:
      - Key: Name
        Value: !Join ['-', [!Ref ClusterName, "cagw"]]

  PublicRouteTable:
    Type: "AWS::EC2::RouteTable"
    Properties:
      VpcId: !Ref VpcId
      Tags:
      - Key: Name
        Value: !Join ['-', [!Ref ClusterName, "public-carrier"]]

  PublicRoute:
    Type: "AWS::EC2::Route"
    DependsOn: CarrierGateway
    Properties:
      RouteTableId: !Ref PublicRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      CarrierGatewayId: !Ref CarrierGateway
  S3Endpoint:
    Type: AWS::EC2::VPCEndpoint
    Properties:
      PolicyDocument:
        Version: 2012-10-17
        Statement:
        - Effect: Allow
          Principal: '*'
          Action:
          - '*'
          Resource:
          - '*'
      RouteTableIds:
      - !Ref PublicRouteTable
      ServiceName: !Join
      - ''
      - - com.amazonaws.
        - !Ref 'AWS::Region'
        - .s3
      VpcId: !Ref VpcId

Outputs:
  PublicRouteTableId:
    Description: Public Route table ID
    Value: !Ref PublicRouteTable
```

### 3.12.7.5. Creating subnets in Wavelength Zones

Before you configure a machine set for edge compute nodes in your OpenShift Container Platform cluster, you must create the subnets in Wavelength Zones. Complete the following procedure for each Wavelength Zone that you want to deploy compute nodes to.

You can use the provided CloudFormation template and create a CloudFormation stack. You can then use this stack to custom provision a subnet.

**NOTE**

If you do not use the provided CloudFormation template to create your AWS infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

**Prerequisites**

- You configured an AWS account.
- You added your AWS keys and region to your local AWS profile by running **aws configure**.
- You opted in to the Wavelength Zones group.

**Procedure**

1. Go to the section of the documentation named "CloudFormation template for the VPC subnet", and copy the syntax from the template. Save the copied template syntax as a YAML file on your local system. This template describes the VPC that your cluster requires.

2. Run the following command to deploy the CloudFormation template, which creates a stack of AWS resources that represent the VPC:

```
$ aws cloudformation create-stack --stack-name <stack_name> \ 1
  --region ${CLUSTER_REGION} \
  --template-body file://<template>.yaml \ 2
  --parameters \
    ParameterKey=VpcId,ParameterValue="${VPC_ID}" \ 3
    ParameterKey=ClusterName,ParameterValue="${CLUSTER_NAME}" \ 4
    ParameterKey=ZoneName,ParameterValue="${ZONE_NAME}" \ 5
    ParameterKey=PublicRouteTableId,ParameterValue="${ROUTE_TABLE_PUB}" \ 6
    ParameterKey=PublicSubnetCidr,ParameterValue="${SUBNET_CIDR_PUB}" \ 7
    ParameterKey=PrivateRouteTableId,ParameterValue="${ROUTE_TABLE_PVT}" \ 8
    ParameterKey=PrivateSubnetCidr,ParameterValue="${SUBNET_CIDR_PVT}" 9
```

1 **<stack_name>** is the name for the CloudFormation stack, such as **cluster-wl-<wavelength_zone_shortname>**. You need the name of this stack if you remove the cluster.

2 **<template>** is the relative path and the name of the CloudFormation template YAML file that you saved.

3 **${VPC_ID}** is the VPC ID, which is the value **VpcID** in the output of the CloudFormation template for the VPC.

4 **${ZONE_NAME}** is the value of Wavelength Zones name to create the subnets.

5 **${CLUSTER_NAME}** is the value of **ClusterName** to be used as a prefix of the new AWS resource names.

6 **${ROUTE_TABLE_PUB}** is the **PublicRouteTableId** extracted from the output of the VPC's carrier gateway CloudFormation stack.

7 **${SUBNET_CIDR_PUB}** is a valid CIDR block that is used to create the public subnet. This block must be part of the VPC CIDR block **VpcCidr**.

8 **${ROUTE_TABLE_PVT}** is the **PrivateRouteTableId** extracted from the output of the VPC's CloudFormation stack.

9 **${SUBNET_CIDR_PVT}** is a valid CIDR block that is used to create the private subnet. This block must be part of the VPC CIDR block **VpcCidr**.

**Example output**

```
arn:aws:cloudformation:us-east-1:123456789012:stack/<stack_name>/dbedae40-820e-11eb-2fd3-
12a48460849f
```

**Verification**

- Confirm that the template components exist by running the following command:

  ```
  $ aws cloudformation describe-stacks --stack-name <stack_name>
  ```

  After the **StackStatus** displays **CREATE_COMPLETE**, the output displays values for the following parameters. Ensure that you provide these parameter values to the other CloudFormation templates that you run to create for your cluster.

| | |
|---|---|
| **PublicSubnetId** | The IDs of the public subnet created by the CloudFormation stack. |
| **PrivateSubnetId** | The IDs of the private subnet created by the CloudFormation stack. |

### 3.12.7.6. CloudFormation template for the VPC subnet

You can use the following CloudFormation template to deploy the private and public subnets in a zone on Wavelength Zones infrastructure.

**Example 3.21. CloudFormation template for VPC subnets**

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for Best Practice Subnets (Public and Private)

Parameters:
  VpcId:
    Description: VPC ID that comprises all the target subnets.
    Type: String
    AllowedPattern: ^(?:(?:vpc)(?:-[a-zA-Z0-9]+)?\b|(?:[0-9]{1,3}\.){3}[0-9]{1,3})$
    ConstraintDescription: VPC ID must be with valid name, starting with vpc-.*.
  ClusterName:
    Description: Cluster name or prefix name to prepend the Name tag for each subnet.
    Type: String
    AllowedPattern: ".+"
    ConstraintDescription: ClusterName parameter must be specified.
  ZoneName:
    Description: Zone Name to create the subnets, such as us-west-2-lax-1a.
    Type: String
    AllowedPattern: ".+"
    ConstraintDescription: ZoneName parameter must be specified.
  PublicRouteTableId:
    Description: Public Route Table ID to associate the public subnet.
    Type: String
    AllowedPattern: ".+"
    ConstraintDescription: PublicRouteTableId parameter must be specified.
  PublicSubnetCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-
4][0-9]|25[0-5])(\/(1[6-9]|2[0-4]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/16-24.
    Default: 10.0.128.0/20
    Description: CIDR block for public subnet.
    Type: String
  PrivateRouteTableId:
```

```
    Description: Private Route Table ID to associate the private subnet.
    Type: String
    AllowedPattern: ".+"
    ConstraintDescription: PrivateRouteTableId parameter must be specified.
  PrivateSubnetCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-
4][0-9]|25[0-5])(\/(1[6-9]|2[0-4]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/16-24.
    Default: 10.0.128.0/20
    Description: CIDR block for private subnet.
    Type: String


Resources:
  PublicSubnet:
    Type: "AWS::EC2::Subnet"
    Properties:
      VpcId: !Ref VpcId
      CidrBlock: !Ref PublicSubnetCidr
      AvailabilityZone: !Ref ZoneName
      Tags:
      - Key: Name
        Value: !Join ['-', [!Ref ClusterName, "public", !Ref ZoneName]]

  PublicSubnetRouteTableAssociation:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Properties:
      SubnetId: !Ref PublicSubnet
      RouteTableId: !Ref PublicRouteTableId

  PrivateSubnet:
    Type: "AWS::EC2::Subnet"
    Properties:
      VpcId: !Ref VpcId
      CidrBlock: !Ref PrivateSubnetCidr
      AvailabilityZone: !Ref ZoneName
      Tags:
      - Key: Name
        Value: !Join ['-', [!Ref ClusterName, "private", !Ref ZoneName]]

  PrivateSubnetRouteTableAssociation:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Properties:
      SubnetId: !Ref PrivateSubnet
      RouteTableId: !Ref PrivateRouteTableId

Outputs:
  PublicSubnetId:
    Description: Subnet ID of the public subnets.
    Value:
      !Join ["", [!Ref PublicSubnet]]

  PrivateSubnetId:
    Description: Subnet ID of the private subnets.
    Value:
      !Join ["", [!Ref PrivateSubnet]]
```

#### 3.12.7.7. Modifying an installation configuration file to use AWS Wavelength Zones subnets

Modify your **install-config.yaml** file to include Wavelength Zones subnets.

**Prerequisites**

- You created subnets by using the procedure "Creating subnets in Wavelength Zones".
- You created an **install-config.yaml** file by using the procedure "Creating the installation configuration file".

**Procedure**

- Modify the **install-config.yaml** configuration file by specifying Wavelength Zones subnets in the **platform.aws.subnets** parameter.

  **Example installation configuration file with Wavelength Zones subnets**

  ```
  # ...
  platform:
    aws:
      region: us-west-2
      subnets: (1)
      - publicSubnetId-1
      - publicSubnetId-2
      - publicSubnetId-3
      - privateSubnetId-1
      - privateSubnetId-2
      - privateSubnetId-3
      - publicOrPrivateSubnetID-Wavelength-1
  # ...
  ```

  (1) List of subnet IDs created in the zones: Availability and Wavelength Zones.

**Additional resources**

- For more information about viewing the CloudFormation stacks that you created, see AWS CloudFormation console.
- For more information about AWS profile and credential configuration, see Configuration and credential file settings in the AWS documentation.

**Next steps**

- Deploying the cluster

### 3.12.8. Optional: Assign public IP addresses to edge compute nodes

If your workload requires deploying the edge compute nodes in public subnets on Wavelength Zones infrastructure, you can configure the machine set manifests when installing a cluster.

AWS Wavelength Zones infrastructure accesses the network traffic in a specified zone, so applications can take advantage of lower latency when serving end users that are closer to that zone.

The default setting that deploys compute nodes in private subnets might not meet your needs, so consider creating edge compute nodes in public subnets when you want to apply more customization to your infrastructure.

IMPORTANT

By default, OpenShift Container Platform deploy the compute nodes in private subnets. For best performance, consider placing compute nodes in subnets that have their Public IP addresses attached to the subnets.

You must create additional security groups, but ensure that you only open the groups' rules over the internet when you really need to.

**Procedure**

1. Change to the directory that contains the installation program and generate the manifest files. Ensure that the installation manifests get created at the **openshift** and **manifests** directory level.

```
$ ./openshift-install create manifests --dir <installation_directory>
```

2. Edit the machine set manifest that the installation program generates for the Wavelength Zones, so that the manifest gets deployed in public subnets. Specify **true** for the **spec.template.spec.providerSpec.value.publicIP** parameter.

**Example machine set manifest configuration for installing a cluster quickly in Wavelength Zones**

```
spec:
  template:
    spec:
      providerSpec:
        value:
          publicIp: true
          subnet:
            filters:
              - name: tag:Name
                values:
                  - ${INFRA_ID}-public-${ZONE_NAME}
```

**Example machine set manifest configuration for installing a cluster in an existing VPC that has Wavelength Zones subnets**

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  name: <infrastructure_id>-edge-<zone>
  namespace: openshift-machine-api
spec:
  template:
    spec:
```

```
providerSpec:
  value:
    publicIp: true
```

### 3.12.9. Deploying the cluster

You can install OpenShift Container Platform on a compatible cloud platform.

IMPORTANT

You can run the **create cluster** command of the installation program only once, during initial installation.

**Prerequisites**

- You have configured an account with the cloud platform that hosts your cluster.
- You have the OpenShift Container Platform installation program and the pull secret for your cluster.
- You have verified that the cloud provider account on your host has the correct permissions to deploy the cluster. An account with incorrect permissions causes the installation process to fail with an error message that displays the missing permissions.

**Procedure**

1. In the directory that contains the installation program, initialize the cluster deployment by running the following command:

```
$ ./openshift-install create cluster --dir <installation_directory> \ 1
    --log-level=info 2
```

1 For **<installation_directory>**, specify the location of your customized **./install-config.yaml** file.

2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

2. Optional: Remove or disable the **AdministratorAccess** policy from the IAM account that you used to install the cluster.

NOTE

The elevated permissions provided by the **AdministratorAccess** policy are required only during installation.

**Verification**

When the cluster deployment completes successfully:

- The terminal displays directions for accessing your cluster, including a link to the web console and credentials for the **kubeadmin** user.
- Credential information also outputs to **<installation_directory>/.openshift_install.log**.

IMPORTANT

Do not delete the installation program or the files that the installation program creates. Both are required to delete the cluster.

**Example output**

```
...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 36m22s
```

IMPORTANT

- The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.
- It is recommended that you use Ignition config files within 12 hours after they are generated because the 24-hour certificate rotates from 16 to 22 hours after the cluster is installed. By using the Ignition config files within 12 hours, you can avoid installation failure if the certificate update runs during installation.

## 3.12.10. Verifying the status of the deployed cluster

Verify that your OpenShift Container Platform successfully deployed on AWS Wavelength Zones.

### 3.12.10.1. Logging in to the cluster by using the CLI

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

**Prerequisites**

- You deployed an OpenShift Container Platform cluster.
- You installed the OpenShift CLI (**oc**).

**Procedure**

1. Export the **kubeadmin** credentials by running the following command:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration by running the following command:

```
$ oc whoami
```

**Example output**

```
system:admin
```

### 3.12.10.2. Logging in to the cluster by using the web console

The **kubeadmin** user exists by default after an OpenShift Container Platform installation. You can log in to your cluster as the **kubeadmin** user by using the OpenShift Container Platform web console.

**Prerequisites**

- You have access to the installation host.
- You completed a cluster installation and all cluster Operators are available.

**Procedure**

1. Obtain the password for the **kubeadmin** user from the **kubeadmin-password** file on the installation host:

```
$ cat <installation_directory>/auth/kubeadmin-password
```

**NOTE**

Alternatively, you can obtain the **kubeadmin** password from the **<installation_directory>/.openshift_install.log** log file on the installation host.

2. List the OpenShift Container Platform web console route:

```
$ oc get routes -n openshift-console | grep 'console-openshift'
```

**NOTE**

Alternatively, you can obtain the OpenShift Container Platform route from the **<installation_directory>/.openshift_install.log** log file on the installation host.

**Example output**

```
console     console-openshift-console.apps.<cluster_name>.<base_domain>            console
https   reencrypt/Redirect   None
```

3. Navigate to the route detailed in the output of the preceding command in a web browser and log in as the **kubeadmin** user.

#### Additional resources

- Accessing the web console

### 3.12.10.3. Verifying nodes that were created with edge compute pool

After you install a cluster that uses AWS Wavelength Zones infrastructure, check the status of the machine that was created by the machine set manifests created during installation.

1. To check the machine sets created from the subnet you added to the **install-config.yaml** file, run the following command:

```
$ oc get machineset -n openshift-machine-api
```

**Example output**

```
NAME                                  DESIRED   CURRENT   READY   AVAILABLE   AGE
cluster-7xw5g-edge-us-east-1-wl1-nyc-wlz-1   1         1         1       1           3h4m
cluster-7xw5g-worker-us-east-1a             1         1         1       1           3h4m
cluster-7xw5g-worker-us-east-1b             1         1         1       1           3h4m
cluster-7xw5g-worker-us-east-1c             1         1         1       1           3h4m
```

2. To check the machines that were created from the machine sets, run the following command:

```
$ oc get machines -n openshift-machine-api
```

**Example output**

```
NAME                                  PHASE     TYPE        REGION      ZONE               AGE
cluster-7xw5g-edge-us-east-1-wl1-nyc-wlz-1-wbclh   Running   c5d.2xlarge   us-east-1   us-
east-1-wl1-nyc-wlz-1   3h
cluster-7xw5g-master-0                      Running   m6i.xlarge    us-east-1   us-east-1a
3h4m
cluster-7xw5g-master-1                      Running   m6i.xlarge    us-east-1   us-east-1b
3h4m
cluster-7xw5g-master-2                      Running   m6i.xlarge    us-east-1   us-east-1c
3h4m
cluster-7xw5g-worker-us-east-1a-rtp45           Running   m6i.xlarge    us-east-1   us-east-1a
3h
cluster-7xw5g-worker-us-east-1b-glm7c           Running   m6i.xlarge    us-east-1   us-east-1b
3h
cluster-7xw5g-worker-us-east-1c-qfvz4           Running   m6i.xlarge    us-east-1   us-east-1c
3h
```

3. To check nodes with edge roles, run the following command:

```
$ oc get nodes -l node-role.kubernetes.io/edge
```

**Example output**

```
NAME                        STATUS   ROLES         AGE    VERSION
ip-10-0-207-188.ec2.internal   Ready    edge,worker   172m   v1.25.2+d2e245f
```

**Next steps**

- Validating an installation.
- If necessary, you can Remote health reporting.

## 3.13. EXTENDING AN AWS VPC CLUSTER INTO AN AWS OUTPOST

In OpenShift Container Platform version 4.14, you could install a cluster on Amazon Web Services (AWS) with compute nodes running in AWS Outposts as a Technology Preview. As of OpenShift Container Platform version 4.15, this installation method is no longer supported. Instead, you can install a cluster on AWS into an existing VPC, and provision compute nodes on AWS Outposts as a postinstallation configuration task.

After installing a cluster on Amazon Web Services (AWS) into an existing Amazon Virtual Private Cloud (VPC), you can create a compute machine set that deploys compute machines in AWS Outposts. AWS Outposts is an AWS edge compute service that enables using many features of a cloud-based AWS deployment with the reduced latency of an on-premise environment. For more information, see the AWS Outposts documentation.

### 3.13.1. AWS Outposts on OpenShift Container Platform requirements and limitations

You can manage the resources on your AWS Outpost similarly to those on a cloud-based AWS cluster if you configure your OpenShift Container Platform cluster to accommodate the following requirements and limitations:

- To extend an OpenShift Container Platform cluster on AWS into an Outpost, you must have installed the cluster into an existing Amazon Virtual Private Cloud (VPC).
- The infrastructure of an Outpost is tied to an availability zone in an AWS region and uses a dedicated subnet. Edge compute machines deployed into an Outpost must use the Outpost subnet and the availability zone that the Outpost is tied to.
- When the AWS Kubernetes cloud controller manager discovers an Outpost subnet, it attempts to create service load balancers in the Outpost subnet. AWS Outposts do not support running service load balancers. To prevent the cloud controller manager from creating unsupported services in the Outpost subnet, you must include the **kubernetes.io/cluster/unmanaged** tag in the Outpost subnet configuration. This requirement is a workaround in OpenShift Container Platform version 4.19. For more information, see OCPBUGS-30041.
- OpenShift Container Platform clusters on AWS include the **gp3-csi** and **gp2-csi** storage classes. These classes correspond to Amazon Elastic Block Store (EBS) gp3 and gp2 volumes. OpenShift Container Platform clusters use the **gp3-csi** storage class by default, but AWS Outposts does not support EBS gp3 volumes.
- This implementation uses the **node-role.kubernetes.io/outposts** taint to prevent spreading regular cluster workloads to the Outpost nodes. To schedule user workloads in the Outpost, you must specify a corresponding toleration in the **Deployment** resource for your application. Reserving the AWS Outpost infrastructure for user workloads avoids additional configuration requirements, such as updating the default CSI to **gp2-csi** so that it is compatible.

- To create a volume in the Outpost, the CSI driver requires the Outpost Amazon Resource Name (ARN). The driver uses the topology keys stored on the **CSINode** objects to determine the Outpost ARN. To ensure that the driver uses the correct topology values, you must set the volume binding mode to **WaitForConsumer** and avoid setting allowed topologies on any new storage classes that you create.

- When you extend an AWS VPC cluster into an Outpost, you have two types of compute resources. The Outpost has edge compute nodes, while the VPC has cloud-based compute nodes. The cloud-based AWS Elastic Block volume cannot attach to Outpost edge compute nodes, and the Outpost volumes cannot attach to cloud-based compute nodes.
  As a result, you cannot use CSI snapshots to migrate applications that use persistent storage from cloud-based compute nodes to edge compute nodes or directly use the original persistent volume. To migrate persistent storage data for applications, you must perform a manual backup and restore operation.

- AWS Outposts does not support AWS Network Load Balancers or AWS Classic Load Balancers. You must use AWS Application Load Balancers to enable load balancing for edge compute resources in the AWS Outposts environment.
  To provision an Application Load Balancer, you must use an Ingress resource and install the AWS Load Balancer Operator. If your cluster contains both edge and cloud-based compute instances that share workloads, additional configuration is required.

  For more information, see "Using the AWS Load Balancer Operator in an AWS VPC cluster extended into an Outpost".

**Additional resources**

- Using the AWS Load Balancer Operator in an AWS VPC cluster extended into an Outpost

### 3.13.2. Obtaining information about your environment

To extend an AWS VPC cluster to your Outpost, you must provide information about your OpenShift Container Platform cluster and your Outpost environment. You use this information to complete network configuration tasks and configure a compute machine set that creates compute machines in your Outpost. You can use command-line tools to gather the required details.

#### 3.13.2.1. Obtaining information from your OpenShift Container Platform cluster

You can use the OpenShift CLI (**oc**) to obtain information from your OpenShift Container Platform cluster.

**TIP**

You might find it convenient to store some or all of these values as environment variables by using the **export** command.

**Prerequisites**

- You have installed an OpenShift Container Platform cluster into a custom VPC on AWS.

- You have access to the cluster using an account with **cluster-admin** permissions.

- You have installed the OpenShift CLI (**oc**).

**Procedure**

1. List the infrastructure ID for the cluster by running the following command. Retain this value.

```
$ oc get -o jsonpath='{.status.infrastructureName}{"\n"}' infrastructures.config.openshift.io
cluster
```

2. Obtain details about the compute machine sets that the installation program created by running the following commands:

   a. List the compute machine sets on your cluster:

   ```
   $ oc get machinesets.machine.openshift.io -n openshift-machine-api
   ```

   **Example output**

   ```
   NAME                            DESIRED   CURRENT   READY   AVAILABLE   AGE
   <compute_machine_set_name_1>   1         1         1       1           55m
   <compute_machine_set_name_2>   1         1         1       1           55m
   ```

   b. Display the Amazon Machine Image (AMI) ID for one of the listed compute machine sets. Retain this value.

   ```
   $ oc get machinesets.machine.openshift.io <compute_machine_set_name_1> \
     -n openshift-machine-api \
     -o jsonpath='{.spec.template.spec.providerSpec.value.ami.id}'
   ```

   c. Display the subnet ID for the AWS VPC cluster. Retain this value.

   ```
   $ oc get machinesets.machine.openshift.io <compute_machine_set_name_1> \
     -n openshift-machine-api \
     -o jsonpath='{.spec.template.spec.providerSpec.value.subnet.id}'
   ```

#### 3.13.2.2. Obtaining information from your AWS account

You can use the AWS CLI (**aws**) to obtain information from your AWS account.

**TIP**

You might find it convenient to store some or all of these values as environment variables by using the **export** command.

**Prerequisites**

- You have an AWS Outposts site with the required hardware setup complete.
- Your Outpost is connected to your AWS account.
- You have access to your AWS account by using the AWS CLI (**aws**) as a user with permissions to perform the required tasks.

**Procedure**

1. List the Outposts that are connected to your AWS account by running the following command:

   ```
   $ aws outposts list-outposts
   ```

2. Retain the following values from the output of the **aws outposts list-outposts** command:

   - The Outpost ID.
   - The Amazon Resource Name (ARN) for the Outpost.
   - The Outpost availability zone.

   NOTE

   The output of the **aws outposts list-outposts** command includes two values related to the availability zone: **AvailabilityZone** and **AvailabilityZoneId**. You use the **AvailablilityZone** value to configure a compute machine set that creates compute machines in your Outpost.

3. Using the value of the Outpost ID, show the instance types that are available in your Outpost by running the following command. Retain the values of the available instance types.

   ```
   $ aws outposts get-outpost-instance-types \
     --outpost-id <outpost_id_value>
   ```

4. Using the value of the Outpost ARN, show the subnet ID for the Outpost by running the following command. Retain this value.

   ```
   $ aws ec2 describe-subnets \
     --filters Name=outpost-arn,Values=<outpost_arn_value>
   ```

### 3.13.3. Configuring your network for your Outpost

To extend your VPC cluster into an Outpost, you must complete the following network configuration tasks:

- Change the Cluster Network MTU.
- Create a subnet in your Outpost.

#### 3.13.3.1. Changing the cluster network MTU to support AWS Outposts

During installation, the maximum transmission unit (MTU) for the cluster network is detected automatically based on the MTU of the primary network interface of nodes in the cluster. You might need to decrease the MTU value for the cluster network to support an AWS Outposts subnet.

IMPORTANT

You cannot roll back an MTU value for nodes during the MTU migration process, but you can roll back the value after the MTU migration process completes.

The migration is disruptive and nodes in your cluster might be temporarily unavailable as the MTU update takes effect.

For more details about the migration process, including important service interruption considerations, see "Changing the MTU for the cluster network" in the additional resources for this procedure.

**Prerequisites**

- You have installed the OpenShift CLI (**oc**).
- You have access to the cluster using an account with **cluster-admin** permissions.
- You have identified the target MTU for your cluster. The MTU for the OVN-Kubernetes network plugin must be set to **100** less than the lowest hardware MTU value in your cluster.
- If your nodes are physical machines, ensure that the cluster network and the connected network switches support jumbo frames.
- If your nodes are virtual machines (VMs), ensure that the hypervisor and the connected network switches support jumbo frames.

#### 3.13.3.1.1. Checking the current cluster MTU value

Use the following procedure to obtain the current maximum transmission unit (MTU) for the cluster network.

**Procedure**

- To obtain the current MTU for the cluster network, enter the following command:

```
$ oc describe network.config cluster
```

**Example output**

```
...
Status:
  Cluster Network:
    Cidr:               10.217.0.0/22
    Host Prefix:        23
  Cluster Network MTU:  1400
  Network Type:         OVNKubernetes
  Service Network:
    10.217.4.0/23
...
```

#### 3.13.3.1.2. Beginning the MTU migration

Use the following procedure to start the MTU migration.

**Procedure**

1. To begin the MTU migration, specify the migration configuration by entering the following command. The Machine Config Operator performs a rolling reboot of the nodes in the cluster in preparation for the MTU change.

```
$ oc patch Network.operator.openshift.io cluster --type=merge --patch \
  '{"spec": { "migration": { "mtu": { "network": { "from": <overlay_from>, "to": <overlay_to> } ,
"machine": { "to" : <machine_to> } } } } }'
```

where:

**<overlay_from>**

Specifies the current cluster network MTU value.

**<overlay_to>**

Specifies the target MTU for the cluster network. This value is set relative to the value of **<machine_to>**. For OVN-Kubernetes, this value must be **100** less than the value of **<machine_to>**.

**<machine_to>**

Specifies the MTU for the primary network interface on the underlying host network.

```
$ oc patch Network.operator.openshift.io cluster --type=merge --patch \
  '{"spec": { "migration": { "mtu": { "network": { "from": 1400, "to": 1000 } , "machine": { "to" :
1100} } } } }'
```

2. As the Machine Config Operator updates machines in each machine config pool, the Operator reboots each node one by one. You must wait until all the nodes are updated. Check the machine config pool status by entering the following command:

```
$ oc get machineconfigpools
```

A successfully updated node has the following status: **UPDATED=true**, **UPDATING=false**, **DEGRADED=false**.

NOTE

By default, the Machine Config Operator updates one machine per pool at a time, causing the total time the migration takes to increase with the size of the cluster.

#### 3.13.3.1.3. Verifying the machine configuration

Use the following procedure to verify the machine configuration.

**Procedure**

- Confirm the status of the new machine configuration on the hosts:

  a. To list the machine configuration state and the name of the applied machine configuration, enter the following command:

```
$ oc describe node | egrep "hostname|machineconfig"
```

**Example output**

```
kubernetes.io/hostname=master-0
machineconfiguration.openshift.io/currentConfig: rendered-master-
```

```
c53e221d9d24e1c8bb6ee89dd3d8ad7b
machineconfiguration.openshift.io/desiredConfig: rendered-master-
c53e221d9d24e1c8bb6ee89dd3d8ad7b
machineconfiguration.openshift.io/reason:
machineconfiguration.openshift.io/state: Done
```

b. Verify that the following statements are true:

   - The value of **machineconfiguration.openshift.io/state** field is **Done**.
   - The value of the **machineconfiguration.openshift.io/currentConfig** field is equal to the value of the **machineconfiguration.openshift.io/desiredConfig** field.

c. To confirm that the machine config is correct, enter the following command:

```
$ oc get machineconfig <config_name> -o yaml | grep ExecStart
```

where:

**<config_name>**

Specifies the name of the machine config from the **machineconfiguration.openshift.io/currentConfig** field.

The machine config must include the following update to the systemd configuration:

```
ExecStart=/usr/local/bin/mtu-migration.sh
```

#### 3.13.3.1.4. Finalizing the MTU migration

Use the following procedure to finalize the MTU migration.

##### Procedure

1. To finalize the MTU migration, enter the following command for the OVN-Kubernetes network plugin:

```
$ oc patch Network.operator.openshift.io cluster --type=merge --patch \
  '{"spec": { "migration": null, "defaultNetwork":{ "ovnKubernetesConfig": { "mtu": <mtu> }}}}'
```

where:

**<mtu>**

Specifies the new cluster network MTU that you specified with **<overlay_to>**.

2. After finalizing the MTU migration, each machine config pool node is rebooted one by one. You must wait until all the nodes are updated. Check the machine config pool status by entering the following command:

```
$ oc get machineconfigpools
```

A successfully updated node has the following status: **UPDATED=true**, **UPDATING=false**, **DEGRADED=false**.

Verification

- Verify that the node in your cluster uses the MTU that you specified by entering the following command:

```
$ oc describe network.config cluster
```

#### Additional resources

- Changing the MTU for the cluster network

### 3.13.3.2. Creating subnets for AWS edge compute services

Before you configure a machine set for edge compute nodes in your OpenShift Container Platform cluster, you must create a subnet in AWS Outposts.

You can use the provided CloudFormation template and create a CloudFormation stack. You can then use this stack to custom provision a subnet.

NOTE

If you do not use the provided CloudFormation template to create your AWS infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

#### Prerequisites

- You configured an AWS account.
- You added your AWS keys and region to your local AWS profile by running **aws configure**.
- You have obtained the required information about your environment from your OpenShift Container Platform cluster, Outpost, and AWS account.

#### Procedure

1. Go to the section of the documentation named "CloudFormation template for the VPC subnet", and copy the syntax from the template. Save the copied template syntax as a YAML file on your local system. This template describes the VPC that your cluster requires.

2. Run the following command to deploy the CloudFormation template, which creates a stack of AWS resources that represent the VPC:

```
$ aws cloudformation create-stack --stack-name <stack_name> \ (1)
  --region ${CLUSTER_REGION} \
  --template-body file://<template>.yaml \ (2)
  --parameters \
    ParameterKey=VpcId,ParameterValue="${VPC_ID}" \ (3)
    ParameterKey=ClusterName,ParameterValue="${CLUSTER_NAME}" \ (4)
    ParameterKey=ZoneName,ParameterValue="${ZONE_NAME}" \ (5)
    ParameterKey=PublicRouteTableId,ParameterValue="${ROUTE_TABLE_PUB}" \ (6)
    ParameterKey=PublicSubnetCidr,ParameterValue="${SUBNET_CIDR_PUB}" \ (7)
    ParameterKey=PrivateRouteTableId,ParameterValue="${ROUTE_TABLE_PVT}" \ (8)
```

```
ParameterKey=PrivateSubnetCidr,ParameterValue="${SUBNET_CIDR_PVT}" \ 9
ParameterKey=PrivateSubnetLabel,ParameterValue="private-outpost" \
ParameterKey=PublicSubnetLabel,ParameterValue="public-outpost" \
ParameterKey=OutpostArn,ParameterValue="${OUTPOST_ARN}" 10
```

1. **<stack_name>** is the name for the CloudFormation stack, such as **cluster-<outpost_name>**.
2. **<template>** is the relative path and the name of the CloudFormation template YAML file that you saved.
3. **${VPC_ID}** is the VPC ID, which is the value **VpcID** in the output of the CloudFormation template for the VPC.
4. **${CLUSTER_NAME}** is the value of **ClusterName** to be used as a prefix of the new AWS resource names.
5. **${ZONE_NAME}** is the value of AWS Outposts name to create the subnets.
6. **${ROUTE_TABLE_PUB}** is the Public Route Table ID created in the **${VPC_ID}** used to associate the public subnets on Outposts. Specify the public route table to associate the Outpost subnet created by this stack.
7. **${SUBNET_CIDR_PUB}** is a valid CIDR block that is used to create the public subnet. This block must be part of the VPC CIDR block **VpcCidr**.
8. **${ROUTE_TABLE_PVT}** is the Private Route Table ID created in the **${VPC_ID}** used to associate the private subnets on Outposts. Specify the private route table to associate the Outpost subnet created by this stack.
9. **${SUBNET_CIDR_PVT}** is a valid CIDR block that is used to create the private subnet. This block must be part of the VPC CIDR block **VpcCidr**.
10. **${OUTPOST_ARN}** is the Amazon Resource Name (ARN) for the Outpost.

### Example output

```
arn:aws:cloudformation:us-east-1:123456789012:stack/<stack_name>/dbedae40-820e-
11eb-2fd3-12a48460849f
```

## Verification

- Confirm that the template components exist by running the following command:

```
$ aws cloudformation describe-stacks --stack-name <stack_name>
```

After the **StackStatus** displays **CREATE_COMPLETE**, the output displays values for the following parameters:

| | |
|---|---|
| **PublicSubnetId** | The IDs of the public subnet created by the CloudFormation stack. |

| | |
|---|---|
| **PrivateSubnetId** | The IDs of the private subnet created by the CloudFormation stack. |

Ensure that you provide these parameter values to the other CloudFormation templates that you run to create for your cluster.

### 3.13.3.3. CloudFormation template for the VPC subnet

You can use the following CloudFormation template to deploy the Outpost subnet.

**Example 3.22. CloudFormation template for VPC subnets**

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for Best Practice Subnets (Public and Private)

Parameters:
  VpcId:
    Description: VPC ID that comprises all the target subnets.
    Type: String
    AllowedPattern: ^(?:(?:vpc)(?:-[a-zA-Z0-9]+)?\b|(?:[0-9]{1,3}\.){3}[0-9]{1,3})$
    ConstraintDescription: VPC ID must be with valid name, starting with vpc-.*.
  ClusterName:
    Description: Cluster name or prefix name to prepend the Name tag for each subnet.
    Type: String
    AllowedPattern: ".+"
    ConstraintDescription: ClusterName parameter must be specified.
  ZoneName:
    Description: Zone Name to create the subnets, such as us-west-2-lax-1a.
    Type: String
    AllowedPattern: ".+"
    ConstraintDescription: ZoneName parameter must be specified.
  PublicRouteTableId:
    Description: Public Route Table ID to associate the public subnet.
    Type: String
    AllowedPattern: ".+"
    ConstraintDescription: PublicRouteTableId parameter must be specified.
  PublicSubnetCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-
4][0-9]|25[0-5])(\/(1[6-9]|2[0-4]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/16-24.
    Default: 10.0.128.0/20
    Description: CIDR block for public subnet.
    Type: String
  PrivateRouteTableId:
    Description: Private Route Table ID to associate the private subnet.
    Type: String
    AllowedPattern: ".+"
    ConstraintDescription: PrivateRouteTableId parameter must be specified.
  PrivateSubnetCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-
4][0-9]|25[0-5])(\/(1[6-9]|2[0-4]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/16-24.
    Default: 10.0.128.0/20
    Description: CIDR block for private subnet.
    Type: String
  PrivateSubnetLabel:
    Default: "private"
    Description: Subnet label to be added when building the subnet name.
    Type: String
  PublicSubnetLabel:
    Default: "public"
    Description: Subnet label to be added when building the subnet name.
    Type: String
  OutpostArn:
    Default: ""
    Description: OutpostArn when creating subnets on AWS Outpost.
    Type: String

Conditions:
  OutpostEnabled: !Not [!Equals [!Ref "OutpostArn", ""]]

Resources:
  PublicSubnet:
    Type: "AWS::EC2::Subnet"
    Properties:
      VpcId: !Ref VpcId
      CidrBlock: !Ref PublicSubnetCidr
      AvailabilityZone: !Ref ZoneName
      OutpostArn: !If [ OutpostEnabled, !Ref OutpostArn, !Ref "AWS::NoValue"]
      Tags:
      - Key: Name
        Value: !Join ['-', [ !Ref ClusterName, !Ref PublicSubnetLabel, !Ref ZoneName]]
      - Key: kubernetes.io/cluster/unmanaged 1
        Value: true

  PublicSubnetRouteTableAssociation:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Properties:
      SubnetId: !Ref PublicSubnet
      RouteTableId: !Ref PublicRouteTableId

  PrivateSubnet:
    Type: "AWS::EC2::Subnet"
    Properties:
      VpcId: !Ref VpcId
      CidrBlock: !Ref PrivateSubnetCidr
      AvailabilityZone: !Ref ZoneName
      OutpostArn: !If [ OutpostEnabled, !Ref OutpostArn, !Ref "AWS::NoValue"]
      Tags:
      - Key: Name
        Value: !Join ['-', [!Ref ClusterName, !Ref PrivateSubnetLabel, !Ref ZoneName]]
      - Key: kubernetes.io/cluster/unmanaged 2
        Value: true

  PrivateSubnetRouteTableAssociation:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Properties:
      SubnetId: !Ref PrivateSubnet
      RouteTableId: !Ref PrivateRouteTableId
```

```
Outputs:
  PublicSubnetId:
    Description: Subnet ID of the public subnets.
    Value:
      !Join ["", [!Ref PublicSubnet]]

  PrivateSubnetId:
    Description: Subnet ID of the private subnets.
    Value:
      !Join ["", [!Ref PrivateSubnet]]
```

1 You must include the **kubernetes.io/cluster/unmanaged** tag in the public subnet configuration for AWS Outposts.

2 You must include the **kubernetes.io/cluster/unmanaged** tag in the private subnet configuration for AWS Outposts.

### 3.13.4. Creating a compute machine set that deploys edge compute machines on an Outpost

To create edge compute machines on AWS Outposts, you must create a new compute machine set with a compatible configuration.

#### Prerequisites

- You have an AWS Outposts site.
- You have installed an OpenShift Container Platform cluster into a custom VPC on AWS.
- You have access to the cluster using an account with **cluster-admin** permissions.
- You have installed the OpenShift CLI (**oc**).

#### Procedure

1. List the compute machine sets in your cluster by running the following command:

   ```
   $ oc get machinesets.machine.openshift.io -n openshift-machine-api
   ```

   **Example output**

   ```
   NAME                          DESIRED   CURRENT   READY   AVAILABLE   AGE
   <original_machine_set_name_1>   1         1         1       1           55m
   <original_machine_set_name_2>   1         1         1       1           55m
   ```

2. Record the names of the existing compute machine sets.
3. Create a YAML file that contains the values for a new compute machine set custom resource (CR) by using one of the following methods:

- Copy an existing compute machine set configuration into a new file by running the following command:

  ```
  $ oc get machinesets.machine.openshift.io <original_machine_set_name_1> \
    -n openshift-machine-api -o yaml > <new_machine_set_name_1>.yaml
  ```

  You can edit this YAML file with your preferred text editor.

- Create an empty YAML file named **<new_machine_set_name_1>.yaml** with your preferred text editor and include the required values for your new compute machine set. If you are not sure which value to set for a specific field, you can view values of an existing compute machine set CR by running the following command:

  ```
  $ oc get machinesets.machine.openshift.io <original_machine_set_name_1> \
    -n openshift-machine-api -o yaml
  ```

  **Example output**

  ```
  apiVersion: machine.openshift.io/v1beta1
  kind: MachineSet
  metadata:
    labels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
    name: <infrastructure_id>-<role>-<availability_zone> 2
    namespace: openshift-machine-api
  spec:
    replicas: 1
    selector:
      matchLabels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id>
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-
  <availability_zone>
    template:
      metadata:
        labels:
          machine.openshift.io/cluster-api-cluster: <infrastructure_id>
          machine.openshift.io/cluster-api-machine-role: <role>
          machine.openshift.io/cluster-api-machine-type: <role>
          machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>-
  <availability_zone>
      spec:
        providerSpec: 3
  # ...
  ```

  1 The cluster infrastructure ID.

  2 A default node label. For AWS Outposts, you use the **outposts** role.

  3 The omitted **providerSpec** section includes values that must be configured for your Outpost.

4. Configure the new compute machine set to create edge compute machines in the Outpost by editing the **<new_machine_set_name_1>.yaml** file:

### Example compute machine set for AWS Outposts

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
  name: <infrastructure_id>-outposts-<availability_zone> 2
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-outposts-
<availability_zone>
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id>
        machine.openshift.io/cluster-api-machine-role: outposts
        machine.openshift.io/cluster-api-machine-type: outposts
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-outposts-
<availability_zone>
    spec:
      metadata:
        labels:
          node-role.kubernetes.io/outposts: ""
          location: outposts
      providerSpec:
        value:
          ami:
            id: <ami_id> 3
          apiVersion: machine.openshift.io/v1beta1
          blockDevices:
            - ebs:
                volumeSize: 120
                volumeType: gp2 4
          credentialsSecret:
            name: aws-cloud-credentials
          deviceIndex: 0
          iamInstanceProfile:
            id: <infrastructure_id>-worker-profile
          instanceType: m5.xlarge 5
          kind: AWSMachineProviderConfig
          placement:
            availabilityZone: <availability_zone>
            region: <region> 6
          securityGroups:
            - filters:
              - name: tag:Name
                values:
                  - <infrastructure_id>-worker-sg
          subnet:
            id: <subnet_id> 7
          tags:
            - name: kubernetes.io/cluster/<infrastructure_id>
              value: owned
          userDataSecret:
            name: worker-user-data
      taints: 8
        - key: node-role.kubernetes.io/outposts
          effect: NoSchedule
```

1. Specifies the cluster infrastructure ID.
2. Specifies the name of the compute machine set. The name is composed of the cluster infrastructure ID, the **outposts** role name, and the Outpost availability zone.
3. Specifies the Amazon Machine Image (AMI) ID.
4. Specifies the EBS volume type. AWS Outposts requires gp2 volumes.
5. Specifies the AWS instance type. You must use an instance type that is configured in your Outpost.
6. Specifies the AWS region in which the Outpost availability zone exists.
7. Specifies the dedicated subnet for your Outpost.
8. Specifies a taint to prevent workloads from being scheduled on nodes that have the **node-role.kubernetes.io/outposts** label. To schedule user workloads in the Outpost, you must specify a corresponding toleration in the **Deployment** resource for your application.

5. Save your changes.
6. Create a compute machine set CR by running the following command:

```
$ oc create -f <new_machine_set_name_1>.yaml
```

### Verification

- To verify that the compute machine set is created, list the compute machine sets in your cluster by running the following command:

```
$ oc get machinesets.machine.openshift.io -n openshift-machine-api
```

**Example output**

```
NAME                            DESIRED   CURRENT   READY   AVAILABLE   AGE
<new_machine_set_name_1>        1         1         1       1           4m12s
<original_machine_set_name_1>   1         1         1       1           55m
<original_machine_set_name_2>   1         1         1       1           55m
```

- To list the machines that are managed by the new compute machine set, run the following command:

```
$ oc get -n openshift-machine-api machines.machine.openshift.io \
  -l machine.openshift.io/cluster-api-machineset=<new_machine_set_name_1>
```

**Example output**

```
NAME                     PHASE         TYPE        REGION      ZONE         AGE
<machine_from_new_1>     Provisioned   m5.xlarge   us-east-1   us-east-1a   25s
<machine_from_new_2>     Provisioning  m5.xlarge   us-east-1   us-east-1a   25s
```

- To verify that a machine created by the new compute machine set has the correct configuration, examine the relevant fields in the CR for one of the new machines by running the following command:

```
$ oc describe machine <machine_from_new_1> -n openshift-machine-api
```

### 3.13.5. Creating user workloads in an Outpost

After you extend an OpenShift Container Platform in an AWS VPC cluster into an Outpost, you can use edge compute nodes with the label **node-role.kubernetes.io/outposts** to create user workloads in the Outpost.

**Prerequisites**

- You have extended an AWS VPC cluster into an Outpost.
- You have access to the cluster using an account with **cluster-admin** permissions.
- You have installed the OpenShift CLI (**oc**).
- You have created a compute machine set that deploys edge compute machines compatible with the Outpost environment.

**Procedure**

1. Configure a **Deployment** resource file for an application that you want to deploy to the edge compute node in the edge subnet.

   **Example Deployment manifest**

```
kind: Namespace
apiVersion: v1
metadata:
  name: <application_name> (1)
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <application_name>
  namespace: <application_namespace> (2)
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  storageClassName: gp2-csi 3
  volumeMode: Filesystem
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: <application_name>
  namespace: <application_namespace>
spec:
  selector:
    matchLabels:
      app: <application_name>
  replicas: 1
  template:
    metadata:
      labels:
        app: <application_name>
        location: outposts 4
    spec:
      securityContext:
        seccompProfile:
          type: RuntimeDefault
      nodeSelector: 5
        node-role.kubernetes.io/outpost: ''
      tolerations: 6
      - key: "node-role.kubernetes.io/outposts"
        operator: "Equal"
        value: ""
        effect: "NoSchedule"
      containers:
        - image: openshift/origin-node
          command:
           - "/bin/socat"
          args:
            - TCP4-LISTEN:8080,reuseaddr,fork
            - EXEC:'/bin/bash -c \"printf \\\"HTTP/1.0 200 OK\r\n\r\n\\\"; sed -e \\\"/^\r/q\\\"\"'
          imagePullPolicy: Always
          name: <application_name>
          ports:
            - containerPort: 8080
          volumeMounts:
            - mountPath: "/mnt/storage"
              name: data
      volumes:
      - name: data
        persistentVolumeClaim:
          claimName: <application_name>
```

1 Specify a name for your application.

2 Specify a namespace for your application. The application namespace can be the same as the application name.

3. Specify the storage class name. For an edge compute configuration, you must use the **gp2-csi** storage class.

4. Specify a label to identify workloads deployed in the Outpost.

5. Specify the node selector label that targets edge compute nodes.

6. Specify tolerations that match the **key** and **effects** taints in the compute machine set for your edge compute machines. Set the **value** and **operator** tolerations as shown.

2. Create the **Deployment** resource by running the following command:

```
$ oc create -f <application_deployment>.yaml
```

3. Configure a **Service** object that exposes a pod from a targeted edge compute node to services that run inside your edge network.

**Example Service manifest**

```
apiVersion: v1
kind: Service (1)
metadata:
  name:  <application_name>
  namespace: <application_namespace>
spec:
  ports:
    - port: 80
      targetPort: 8080
      protocol: TCP
  type: NodePort
  selector: (2)
    app: <application_name>
```

1. Defines the **service** resource.

2. Specify the label type to apply to managed pods.

4. Create the **Service** CR by running the following command:

```
$ oc create -f <application_service>.yaml
```

### 3.13.6. Scheduling workloads on edge and cloud-based AWS compute resources

When you extend an AWS VPC cluster into an Outpost, the Outpost uses edge compute nodes and the VPC uses cloud-based compute nodes. The following load balancer considerations apply to an AWS VPC cluster extended into an Outpost:

- Outposts cannot run AWS Network Load Balancers or AWS Classic Load Balancers, but a Classic Load Balancer for a VPC cluster extended into an Outpost can attach to the Outpost edge compute nodes. For more information, see Using AWS Classic Load Balancers in an AWS VPC cluster extended into an Outpost.

- To run a load balancer on an Outpost instance, you must use an AWS Application Load Balancer. You can use the AWS Load Balancer Operator to deploy an instance of the AWS Load Balancer Controller. The controller provisions AWS Application Load Balancers for Kubernetes Ingress resources. For more information, see Using the AWS Load Balancer Operator in an AWS VPC cluster extended into an Outpost.

### 3.13.6.1. Using AWS Classic Load Balancers in an AWS VPC cluster extended into an Outpost

AWS Outposts infrastructure cannot run AWS Classic Load Balancers, but Classic Load Balancers in the AWS VPC cluster can target edge compute nodes in the Outpost if edge and cloud-based subnets are in the same availability zone. As a result, Classic Load Balancers on the VPC cluster might schedule pods on either of these node types.

Scheduling the workloads on edge compute nodes and cloud-based compute nodes can introduce latency. If you want to prevent a Classic Load Balancer in the VPC cluster from targeting Outpost edge compute nodes, you can apply labels to the cloud-based compute nodes and configure the Classic Load Balancer to only schedule on nodes with the applied labels.

NOTE

If you do not need to prevent a Classic Load Balancer in the VPC cluster from targeting Outpost edge compute nodes, you do not need to complete these steps.

#### Prerequisites

- You have extended an AWS VPC cluster into an Outpost.
- You have access to the cluster using an account with **cluster-admin** permissions.
- You have installed the OpenShift CLI (**oc**).
- You have created a user workload in the Outpost with tolerations that match the taints for your edge compute machines.

#### Procedure

1. Optional: Verify that the edge compute nodes have the **location=outposts** label by running the following command and verifying that the output includes only the edge compute nodes in your Outpost:

```
$ oc get nodes -l location=outposts
```

2. Label the cloud-based compute nodes in the VPC cluster with a key-value pair by running the following command:

```
$ for NODE in $(oc get node -l node-role.kubernetes.io/worker --no-headers | grep -v
outposts | awk '{print$1}'); do oc label node $NODE <key_name>=<value>; done
```

where **<key_name>=<value>** is the label you want to use to distinguish cloud-based compute nodes.

**Example output**

```
node1.example.com labeled
node2.example.com labeled
node3.example.com labeled
```

3. Optional: Verify that the cloud-based compute nodes have the specified label by running the following command and confirming that the output includes all cloud-based compute nodes in your VPC cluster:

```
$ oc get nodes -l <key_name>=<value>
```

**Example output**

```
NAME                STATUS   ROLES    AGE     VERSION
node1.example.com   Ready    worker   7h      v1.32.3
node2.example.com   Ready    worker   7h      v1.32.3
node3.example.com   Ready    worker   7h      v1.32.3
```

4. Configure the Classic Load Balancer service by adding the cloud-based subnet information to the **annotations** field of the **Service** manifest:

**Example service configuration**

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app: <application_name>
  name: <application_name>
  namespace: <application_namespace>
  annotations:
    service.beta.kubernetes.io/aws-load-balancer-subnets: <aws_subnet> (1)
    service.beta.kubernetes.io/aws-load-balancer-target-node-labels: <key_name>=<value>
(2)
spec:
  ports:
  - name: http
    port: 80
    protocol: TCP
    targetPort: 8080
  selector:
    app: <application_name>
  type: LoadBalancer
```

(1) Specify the subnet ID for the AWS VPC cluster.

(2) Specify the key-value pair that matches the pair in the node label.

5. Create the **Service** CR by running the following command:

```
$ oc create -f <file_name>.yaml
```

**Verification**

1. Verify the status of the **service** resource to show the host of the provisioned Classic Load Balancer by running the following command:

   ```
   $ HOST=$(oc get service <application_name> -n <application_namespace> --
   template='{{(index .status.loadBalancer.ingress 0).hostname}}')
   ```

2. Verify the status of the provisioned Classic Load Balancer host by running the following command:

   ```
   $ curl $HOST
   ```

3. In the AWS console, verify that only the labeled instances appear as the targeted instances for the load balancer.

### 3.13.6.2. Using the AWS Load Balancer Operator in an AWS VPC cluster extended into an Outpost

You can configure the AWS Load Balancer Operator to provision an AWS Application Load Balancer in an AWS VPC cluster extended into an Outpost. AWS Outposts does not support AWS Network Load Balancers. As a result, the AWS Load Balancer Operator cannot provision Network Load Balancers in an Outpost.

You can create an AWS Application Load Balancer either in the cloud subnet or in the Outpost subnet. An Application Load Balancer in the cloud can attach to cloud-based compute nodes and an Application Load Balancer in the Outpost can attach to edge compute nodes. You must annotate Ingress resources with the Outpost subnet or the VPC subnet, but not both.

#### Prerequisites

- You have extended an AWS VPC cluster into an Outpost.
- You have installed the OpenShift CLI (**oc**).
- You have installed the AWS Load Balancer Operator and created the AWS Load Balancer Controller.

#### Procedure

- Configure the **Ingress** resource to use a specified subnet:

  **Example Ingress resource configuration**

  ```
  apiVersion: networking.k8s.io/v1
  kind: Ingress
  metadata:
    name: <application_name>
    annotations:
      alb.ingress.kubernetes.io/subnets: <subnet_id> (1)
  spec:
    ingressClassName: alb
    rules:
      - http:
          paths:
            - path: /
  ```

```
            pathType: Exact
            backend:
              service:
                name: <application_name>
                port:
                  number: 80
```

1. Specifies the subnet to use.
   - To use the Application Load Balancer in an Outpost, specify the Outpost subnet ID.
   - To use the Application Load Balancer in the cloud, you must specify at least two subnets in different availability zones.

**Additional resources**

- Creating the AWS Load Balancer Controller

### 3.13.7. Additional resources

- Installing a cluster on AWS into an existing VPC

## 3.14. INSTALLING A CLUSTER WITH THE SUPPORT FOR CONFIGURING MULTI-ARCHITECTURE COMPUTE MACHINES

An OpenShift Container Platform cluster with multi-architecture compute machines supports compute machines with different architectures.

> **NOTE**
>
> When you have nodes with multiple architectures in your cluster, the architecture of your image must be consistent with the architecture of the node. You must ensure that the pod is assigned to the node with the appropriate architecture and that it matches the image architecture. For more information on assigning pods to nodes, Scheduling workloads on clusters with multi-architecture compute machines.

You can install an Amazon Web Services (AWS) cluster with the support for configuring multi-architecture compute machines. After installing the cluster, you can add multi-architecture compute machines to the cluster in the following ways:

- Adding 64-bit x86 compute machines to a cluster that uses 64-bit ARM control plane machines and already includes 64-bit ARM compute machines. In this case, 64-bit x86 is considered the secondary architecture.
- Adding 64-bit ARM compute machines to a cluster that uses 64-bit x86 control plane machines and already includes 64-bit x86 compute machines. In this case, 64-bit ARM is considered the secondary architecture.

NOTE

Before adding a secondary architecture node to your cluster, it is recommended to install the Multiarch Tuning Operator, and deploy a **ClusterPodPlacementConfig** custom resource. For more information, see "Managing workloads on multi-architecture clusters by using the Multiarch Tuning Operator".

### 3.14.1. Installing a cluster with multi-architecture support

You can install a cluster with the support for configuring multi-architecture compute machines.

#### Prerequisites

- You installed the OpenShift CLI (**oc**).
- You have the OpenShift Container Platform installation program.
- You downloaded the pull secret for your cluster.

#### Procedure

1. Check that the **openshift-install** binary is using the **multi** payload by running the following command:

```
$ ./openshift-install version
```

**Example output**

```
./openshift-install 4.19.0
built from commit abc123etc
release image quay.io/openshift-release-dev/ocp-release@sha256:abc123wxyzetc
release architecture multi
default architecture amd64
```

The output must contain **release architecture multi** to indicate that the **openshift-install** binary is using the **multi** payload.

2. Update the **install-config.yaml** file to configure the architecture for the nodes.

**Sample install-config.yaml file with multi-architecture configuration**

```
apiVersion: v1
baseDomain: example.openshift.com
compute:
- architecture: amd64 (1)
  hyperthreading: Enabled
  name: worker
  platform: {}
  replicas: 3
controlPlane:
  architecture: arm64 (2)
  name: master
  platform: {}
  replicas: 3
```

```
# ...
```

1. Specify the architecture of the worker node. You can set this field to either **arm64** or **amd64**.

2. Specify the control plane node architecture. You can set this field to either **arm64** or **amd64**.

### Next steps

- Deploying the cluster

### Additional resources

- Managing workloads on multi-architecture clusters by using the Multiarch Tuning Operator

# CHAPTER 4. USER-PROVISIONED INFRASTRUCTURE

## 4.1. PREPARING TO INSTALL A CLUSTER ON AWS

You prepare to install an OpenShift Container Platform cluster on AWS by completing the following steps:

- Verifying internet connectivity for your cluster.
- Configuring an AWS account .
- Downloading the installation program.

  NOTE

  If you are installing in a disconnected environment, you extract the installation program from the mirrored content. For more information, see Mirroring images for a disconnected installation.

- Installing the OpenShift CLI (**oc**).

  NOTE

  If you are installing in a disconnected environment, install **oc** to the mirror host.

- Generating an SSH key pair. You can use this key pair to authenticate into the OpenShift Container Platform cluster's nodes after it is deployed.
- Preparing the user-provisioned infrastructure.
- If the cloud identity and access management (IAM) APIs are not accessible in your environment, or if you do not want to store an administrator-level credential secret in the **kube-system** namespace, manually creating long-term credentials for AWS or configuring an AWS cluster to use short-term credentials with Amazon Web Services Security Token Service (AWS STS).

### 4.1.1. Internet access for OpenShift Container Platform

In OpenShift Container Platform 4.19, you require access to the internet to install your cluster.

You must have internet access to perform the following actions:

- Access OpenShift Cluster Manager to download the installation program and perform subscription management. If the cluster has internet access and you do not disable Telemetry, that service automatically entitles your cluster.
- Access Quay.io to obtain the packages that are required to install your cluster.
- Obtain the packages that are required to perform cluster updates.

IMPORTANT

If your cluster cannot have direct internet access, you can perform a restricted network installation on some types of infrastructure that you provision. During that process, you download the required content and use it to populate a mirror registry with the installation packages. With some installation types, the environment that you install your cluster in will not require internet access. Before you update the cluster, you update the content of the mirror registry.

### 4.1.2. Obtaining the installation program

Before you install OpenShift Container Platform, download the installation file on the host you are using for installation.

**Prerequisites**

- You have a computer that runs Linux or macOS, with 500 MB of local disk space.

**Procedure**

1. Go to the Cluster Type page on the Red Hat Hybrid Cloud Console. If you have a Red Hat account, log in with your credentials. If you do not, create an account.

   TIP

   You can also download the binaries for a specific OpenShift Container Platform release .

2. Select your infrastructure provider from the **Run it yourself** section of the page.

3. Select your host operating system and architecture from the dropdown menus under **OpenShift Installer** and click **Download Installer**.

4. Place the downloaded file in the directory where you want to store the installation configuration files.

   IMPORTANT

   - The installation program creates several files on the computer that you use to install your cluster. You must keep the installation program and the files that the installation program creates after you finish installing the cluster. Both of the files are required to delete the cluster.

   - Deleting the files created by the installation program does not remove your cluster, even if the cluster failed during installation. To remove your cluster, complete the OpenShift Container Platform uninstallation procedures for your specific cloud provider.

5. Extract the installation program. For example, on a computer that uses a Linux operating system, run the following command:

   ```
   $ tar -xvf openshift-install-linux.tar.gz
   ```

6. Download your installation pull secret from Red Hat OpenShift Cluster Manager . This pull secret allows you to authenticate with the services that are provided by the included authorities,

including Quay.io, which serves the container images for OpenShift Container Platform components.

**TIP**

Alternatively, you can retrieve the installation program from the Red Hat Customer Portal, where you can specify a version of the installation program to download. However, you must have an active subscription to access this page.

### 4.1.3. Installing the OpenShift CLI on Linux

To manage your cluster and deploy applications from the command line, install the OpenShift CLI (**oc**) binary on Linux.

**IMPORTANT**

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform.

Download and install the new version of **oc**.

**Procedure**

1. Navigate to the Download OpenShift Container Platform page on the Red Hat Customer Portal.
2. Select the architecture from the **Product Variant** list.
3. Select the appropriate version from the **Version** list.
4. Click **Download Now** next to the **OpenShift v4.19 Linux Clients** entry and save the file.
5. Unpack the archive:

```
$ tar xvf <file>
```

6. Place the **oc** binary in a directory that is on your **PATH**.
   To check your **PATH**, execute the following command:

```
$ echo $PATH
```

**Verification**

- After you install the OpenShift CLI, it is available using the **oc** command:

```
$ oc <command>
```

### 4.1.4. Installing the OpenShift CLI on Windows

To manage your cluster and deploy applications from the command line, install OpenShift CLI (**oc**) binary on Windows.

IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform.

Download and install the new version of **oc**.

### Procedure

1. Navigate to the Download OpenShift Container Platform page on the Red Hat Customer Portal.
2. Select the appropriate version from the **Version** list.
3. Click **Download Now** next to the **OpenShift v4.19 Windows Client** entry and save the file.
4. Extract the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH** variable.
   To check your **PATH** variable, open the command prompt and execute the following command:

```
C:\> path
```

### Verification

- After you install the OpenShift CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

## 4.1.5. Installing the OpenShift CLI on macOS

To manage your cluster and deploy applications from the command line, install the OpenShift CLI (**oc**) binary on macOS.

IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in OpenShift Container Platform.

Download and install the new version of **oc**.

### Procedure

1. Navigate to the Download OpenShift Container Platform page on the Red Hat Customer Portal.
2. Select the architecture from the **Product Variant** list.
3. Select the appropriate version from the **Version** list.
4. Click **Download Now** next to the **OpenShift v4.19 macOS Clients** entry and save the file.

   NOTE

   For macOS arm64, choose the **OpenShift v4.19 macOS arm64 Client** entry.

5. Unpack and unzip the archive.

6. Move the **oc** binary to a directory on your **PATH** variable.
   To check your **PATH** variable, open a terminal and execute the following command:

   ```
   $ echo $PATH
   ```

**Verification**

- Verify your installation by using an **oc** command:

  ```
  $ oc <command>
  ```

### 4.1.6. Generating a key pair for cluster node SSH access

To enable secure, passwordless SSH access to your cluster nodes, provide an SSH public key during the OpenShift Container Platform installation. This ensures that the installation program automatically configures the Red Hat Enterprise Linux CoreOS (RHCOS) nodes for remote authentication through the **core** user.

The SSH public key gets added to the **~/.ssh/authorized_keys** list for the **core** user on each node. After the key is passed to the Red Hat Enterprise Linux CoreOS (RHCOS) nodes through their Ignition config files, you can use the key pair to SSH in to the RHCOS nodes as the user **core**. To access the nodes through SSH, the private key identity must be managed by SSH for your local user.

If you want to SSH in to your cluster nodes to perform installation debugging or disaster recovery, you must provide the SSH public key during the installation process. The **./openshift-install gather** command also requires the SSH public key to be in place on the cluster nodes.

IMPORTANT

Do not skip this procedure in production environments, where disaster recovery and debugging is required.

NOTE

You must use a local key, not one that you configured with platform-specific approaches.

**Procedure**

1. If you do not have an existing SSH key pair on your local machine to use for authentication onto your cluster nodes, create one. For example, on a computer that uses a Linux operating system, run the following command:

   ```
   $ ssh-keygen -t ed25519 -N '' -f <path>/<file_name>
   ```

   Specifies the path and file name, such as **~/.ssh/id_ed25519**, of the new SSH key. If you have an existing key pair, ensure your public key is in the your **~/.ssh** directory.

NOTE

If you plan to install an OpenShift Container Platform cluster that uses the RHEL cryptographic libraries that have been submitted to NIST for FIPS 140-2/140-3 Validation on only the **x86_64**, **ppc64le**, and **s390x** architectures, do not create a key that uses the **ed25519** algorithm. Instead, create a key that uses the **rsa** or **ecdsa** algorithm.

2. View the public SSH key:

```
$ cat <path>/<file_name>.pub
```

For example, run the following to view the **~/.ssh/id_ed25519.pub** public key:

```
$ cat ~/.ssh/id_ed25519.pub
```

3. Add the SSH private key identity to the SSH agent for your local user, if it has not already been added. SSH agent management of the key is required for password-less SSH authentication onto your cluster nodes, or if you want to use the **./openshift-install gather** command.

NOTE

On some distributions, default SSH private key identities such as **~/.ssh/id_rsa** and **~/.ssh/id_dsa** are managed automatically.

a. If the **ssh-agent** process is not already running for your local user, start it as a background task:

```
$ eval "$(ssh-agent -s)"
```

**Example output**

```
Agent pid 31874
```

NOTE

If your cluster is in FIPS mode, only use FIPS-compliant algorithms to generate the SSH key. The key must be either RSA or ECDSA.

4. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name>
```

Specifies the path and file name for your SSH private key, such as **~/.ssh/id_ed25519**

**Example output**

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

**Next steps**

- When you install OpenShift Container Platform, provide the SSH public key to the installation program.

### 4.1.7. Telemetry access for OpenShift Container Platform

In OpenShift Container Platform 4.19, the Telemetry service, which runs by default to provide metrics about cluster health and the success of updates, requires internet access. If your cluster is connected to the internet, Telemetry runs automatically, and your cluster is registered to OpenShift Cluster Manager.

After you confirm that your OpenShift Cluster Manager inventory is correct, either maintained automatically by Telemetry or manually by using OpenShift Cluster Manager, use subscription watch to track your OpenShift Container Platform subscriptions at the account or multi-cluster level.

**Additional resources**

- See About remote health monitoring for more information about the Telemetry service.

## 4.2. INSTALLATION REQUIREMENTS FOR USER-PROVISIONED INFRASTRUCTURE ON AWS

Before you begin an installation on infrastructure that you provision, be sure that your AWS environment meets the following installation requirements.

For a cluster that contains user-provisioned infrastructure, you must deploy all of the required machines.

### 4.2.1. Required machines for cluster installation

You must specify the minimum required machines or hosts for your cluster so that your cluster remains stable if a node fails.

The smallest OpenShift Container Platform clusters require the following hosts:

**IMPORTANT**

For a cluster that contains user-provisioned infrastructure, you must deploy all of the required machines.

**Table 4.1. Minimum required hosts**

| Hosts | Description |
|---|---|
| One temporary bootstrap machine | The cluster requires the bootstrap machine to deploy the OpenShift Container Platform cluster on the three control plane machines. You can remove the bootstrap machine after you install the cluster. |
| Three control plane machines | The control plane machines run the Kubernetes and OpenShift Container Platform services that form the control plane. |
| At least two compute machines, which are also known as worker machines. | The workloads requested by OpenShift Container Platform users run on the compute machines. |

IMPORTANT

To maintain high availability of your cluster, use separate physical hosts for these cluster machines.

The bootstrap and control plane machines must use Red Hat Enterprise Linux CoreOS (RHCOS) as the operating system. However, the compute machines can choose between Red Hat Enterprise Linux CoreOS (RHCOS), Red Hat Enterprise Linux (RHEL) 8.6 and later.

Note that RHCOS is based on Red Hat Enterprise Linux (RHEL) 9.2 and inherits all of its hardware certifications and requirements. See Red Hat Enterprise Linux technology capabilities and limits .

#### 4.2.1.1. Minimum resource requirements for cluster installation

Each created cluster must meet minimum requirements so that the cluster runs as expected.

**Table 4.2. Minimum resource requirements**

| Machine | Operating System | vCPU [1] | Virtual RAM | Storage | Input/Output Per Second (IOPS)[2] |
|---|---|---|---|---|---|
| Bootstrap | RHCOS | 4 | 16 GB | 100 GB | 300 |
| Control plane | RHCOS | 4 | 16 GB | 100 GB | 300 |
| Compute | RHCOS, RHEL 8.6 and later [3] | 2 | 8 GB | 100 GB | 300 |

1. One vCPU is equivalent to one physical core when simultaneous multithreading (SMT), or Hyper-Threading, is not enabled. When enabled, use the following formula to calculate the corresponding ratio: (threads per core × cores) × sockets = vCPUs.

2. OpenShift Container Platform and Kubernetes are sensitive to disk performance, and faster storage is recommended, particularly for etcd on the control plane nodes which require a 10 ms p99 fsync duration. Note that on many cloud platforms, storage size and IOPS scale together, so you might need to over-allocate storage volume to obtain sufficient performance.

3. As with all user-provisioned installations, if you choose to use RHEL compute machines in your cluster, you take responsibility for all operating system life cycle management and maintenance, including performing system updates, applying patches, and completing all other required tasks. Use of RHEL 7 compute machines is deprecated and has been removed in OpenShift Container Platform 4.10 and later.

NOTE

For OpenShift Container Platform version 4.19, RHCOS is based on RHEL version 9.6, which updates the micro-architecture requirements. The following list contains the minimum instruction set architectures (ISA) that each architecture requires:

- x86-64 architecture requires x86-64-v2 ISA
- ARM64 architecture requires ARMv8.0-A ISA
- IBM Power architecture requires Power 9 ISA
- s390x architecture requires z14 ISA

For more information, see Architectures (RHEL documentation).

If an instance type for your platform meets the minimum requirements for cluster machines, it is supported to use in OpenShift Container Platform.

**Additional resources**

- Optimizing storage

#### 4.2.1.2. Tested instance types for AWS

The following Amazon Web Services (AWS) instance types have been tested with OpenShift Container Platform.

NOTE

Use the machine types included in the following charts for your AWS instances. If you use an instance type that is not listed in the chart, ensure that the instance size you use matches the minimum resource requirements that are listed in the section named "Minimum resource requirements for cluster installation".

**Example 4.1. Machine types based on 64-bit x86 architecture**

- **c4.***
- **c5.***
- **c5a.***
- **i3.***
- **m4.***
- **m5.***
- **m5a.***
- **m6a.***
- **m6i.***

- **r4.***
- **r5.***
- **r5a.***
- **r6i.***
- **t3.***
- **t3a.***

#### 4.2.1.3. Tested instance types for AWS on 64-bit ARM infrastructures

The following Amazon Web Services (AWS) 64-bit ARM instance types have been tested with OpenShift Container Platform.

NOTE

Use the machine types included in the following charts for your AWS ARM instances. If you use an instance type that is not listed in the chart, ensure that the instance size you use matches the minimum resource requirements that are listed in "Minimum resource requirements for cluster installation".

Example 4.2. Machine types based on 64-bit ARM architecture

- **c6g.***
- **c7g.***
- **m6g.***
- **m7g.***
- **r8g.***

### 4.2.2. Certificate signing requests management

On user-provisioned infrastructure, you must provide a mechanism for approving cluster certificate signing requests (CSRs) after installation when your cluster has limited access to automatic machine management.

The **kube-controller-manager** only approves the kubelet client CSRs. The **machine-approver** cannot guarantee the validity of a serving certificate that is requested by using kubelet credentials because it cannot confirm that the correct machine issued the request. You must determine and implement a method of verifying the validity of the kubelet serving certificate requests and approving them.

### 4.2.3. Required AWS infrastructure components

To install OpenShift Container Platform on user-provisioned infrastructure in Amazon Web Services (AWS), you must manually create both the machines and their supporting infrastructure.

For more information about the integration testing for different platforms, see the OpenShift Container Platform 4.x Tested Integrations page.

By using the provided CloudFormation templates, you can create stacks of AWS resources that represent the following components:

- An AWS Virtual Private Cloud (VPC)
- Networking and load balancing components
- Security groups and roles
- An OpenShift Container Platform bootstrap node
- OpenShift Container Platform control plane nodes
- An OpenShift Container Platform compute node

Alternatively, you can manually create the components or you can reuse existing infrastructure that meets the cluster requirements. Review the CloudFormation templates for more details about how the components interrelate.

#### 4.2.3.1. Other infrastructure components

- A VPC
- DNS entries
- Load balancers (classic or network) and listeners
- A public and a private Route 53 zone
- Security groups
- IAM roles
- S3 buckets

If you are working in a disconnected environment, you are unable to reach the public IP addresses for EC2, ELB, and S3 endpoints. Depending on the level to which you want to restrict internet traffic during the installation, the following configuration options are available:

##### 4.2.3.1.1. Option 1: Create VPC endpoints

Create a VPC endpoint and attach it to the subnets that the clusters are using. Name the endpoints as follows:

- **ec2.<aws_region>.amazonaws.com**
- **elasticloadbalancing.<aws_region>.amazonaws.com**
- **s3.<aws_region>.amazonaws.com**

With this option, network traffic remains private between your VPC and the required AWS services.

##### 4.2.3.1.2. Option 2: Create a proxy without VPC endpoints

As part of the installation process, you can configure an HTTP or HTTPS proxy. With this option, internet traffic goes through the proxy to reach the required AWS services.

#### 4.2.3.1.3. Option 3: Create a proxy with VPC endpoints

As part of the installation process, you can configure an HTTP or HTTPS proxy with VPC endpoints. Create a VPC endpoint and attach it to the subnets that the clusters are using. Name the endpoints as follows:

- **ec2.<aws_region>.amazonaws.com**
- **elasticloadbalancing.<aws_region>.amazonaws.com**
- **s3.<aws_region>.amazonaws.com**

When configuring the proxy in the **install-config.yaml** file, add these endpoints to the **noProxy** field. With this option, the proxy prevents the cluster from accessing the internet directly. However, network traffic remains private between your VPC and the required AWS services.

### Required VPC components

You must provide a suitable VPC and subnets that allow communication to your machines.

| Component | AWS type | Description |
|---|---|---|
| VPC | • **AWS::EC2::VPC**<br>• **AWS::EC2::VPCEndpoint** | You must provide a public VPC for the cluster to use. The VPC uses an endpoint that references the route tables for each subnet to improve communication with the registry that is hosted in S3. |
| Public subnets | • **AWS::EC2::Subnet**<br>• **AWS::EC2::SubnetNetworkAclAssociation** | Your VPC must have public subnets for between 1 and 3 availability zones and associate them with appropriate Ingress rules. |
| Internet gateway | • **AWS::EC2::InternetGateway**<br>• **AWS::EC2::VPCGatewayAttachment**<br>• **AWS::EC2::RouteTable**<br>• **AWS::EC2::Route**<br>• **AWS::EC2::SubnetRouteTableAssociation**<br>• **AWS::EC2::NatGateway**<br>• **AWS::EC2::EIP** | You must have a public internet gateway, with public routes, attached to the VPC. In the provided templates, each public subnet has a NAT gateway with an EIP address. These NAT gateways allow cluster resources, like private subnet instances, to reach the internet and are not required for some restricted network or proxy scenarios. |

| Compone nt | AWS type | Description | |
|---|---|---|---|
| Network access control | • **AWS::EC2::NetworkAcl**<br>• **AWS::EC2::NetworkAclEntry** | You must allow the VPC to access the following ports: | |
| | | Port | Reason |
| | | **80** | Inbound HTTP traffic |
| | | **443** | Inbound HTTPS traffic |
| | | **22** | Inbound SSH traffic |
| | | **1024 - 65535** | Inbound ephemeral traffic |
| | | **0 - 65535** | Outbound ephemeral traffic |
| Private subnets | • **AWS::EC2::Subnet**<br>• **AWS::EC2::RouteTable**<br>• **AWS::EC2::SubnetRouteTableAssociation** | Your VPC can have private subnets. The provided CloudFormation templates can create private subnets for between 1 and 3 availability zones. If you use private subnets, you must provide appropriate routes and tables for them. | |

## Required DNS and load balancing components

Your DNS and load balancer configuration needs to use a public hosted zone and can use a private hosted zone similar to the one that the installation program uses if it provisions the cluster's infrastructure. You must create a DNS entry that resolves to your load balancer. An entry for **api.<cluster_name>.<domain>** must point to the external load balancer, and an entry for **api-int.<cluster_name>.<domain>** must point to the internal load balancer.

The cluster also requires load balancers and listeners for port 6443, which are required for the Kubernetes API and its extensions, and port 22623, which are required for the Ignition config files for new machines. The targets will be the control plane nodes. Port 6443 must be accessible to both clients external to the cluster and nodes within the cluster. Port 22623 must be accessible to nodes within the cluster.

| Component | AWS type | Description |
|---|---|---|

| Component | AWS type | Description |
|---|---|---|
| DNS | **AWS::Route53::HostedZone** | The hosted zone for your internal DNS. |
| Public load balancer | **AWS::ElasticLoadBalancingV2::LoadBalancer** | The load balancer for your public subnets. |
| External API server record | **AWS::Route53::RecordSetGroup** | Alias records for the external API server. |
| External listener | **AWS::ElasticLoadBalancingV2::Listener** | A listener on port 6443 for the external load balancer. |
| External target group | **AWS::ElasticLoadBalancingV2::TargetGroup** | The target group for the external load balancer. |
| Private load balancer | **AWS::ElasticLoadBalancingV2::LoadBalancer** | The load balancer for your private subnets. |
| Internal API server record | **AWS::Route53::RecordSetGroup** | Alias records for the internal API server. |
| Internal listener | **AWS::ElasticLoadBalancingV2::Listener** | A listener on port 22623 for the internal load balancer. |
| Internal target group | **AWS::ElasticLoadBalancingV2::TargetGroup** | The target group for the internal load balancer. |
| Internal listener | **AWS::ElasticLoadBalancingV2::Listener** | A listener on port 6443 for the internal load balancer. |

| Component | AWS type | Description |
|---|---|---|
| Internal target group | **AWS::ElasticLoadBalancingV2::TargetGroup** | The target group for the internal load balancer. |

## Security groups

The control plane and worker machines require access to the following ports:

| Group | Type | IP Protocol | Port range |
|---|---|---|---|
| **MasterSecurityGroup** | **AWS::EC2::SecurityGroup** | **icmp** | **0** |
| | | **tcp** | **22** |
| | | **tcp** | **6443** |
| | | **tcp** | **22623** |
| **WorkerSecurityGroup** | **AWS::EC2::SecurityGroup** | **icmp** | **0** |
| | | **tcp** | **22** |
| **BootstrapSecurityGroup** | **AWS::EC2::SecurityGroup** | **tcp** | **22** |
| | | **tcp** | **19531** |

## Control plane Ingress

The control plane machines require the following Ingress groups. Each Ingress group is a **AWS::EC2::SecurityGroupIngress** resource.

| Ingress group | Description | IP protocol | Port range |
|---|---|---|---|
| **MasterIngressEtcd** | etcd | **tcp** | **2379- 2380** |
| **MasterIngressVxlan** | Vxlan packets | **udp** | **6081** |
| **MasterIngressWorkerVxlan** | Vxlan packets | **udp** | **6081** |

| Ingress group | Description | IP protocol | Port range |
|---|---|---|---|
| **MasterIngressInternal** | Internal cluster communication and Kubernetes proxy metrics | **tcp** | **9000 - 9999** |
| **MasterIngressWorkerInternal** | Internal cluster communication | **tcp** | **9000 - 9999** |
| **MasterIngressKube** | Kubernetes kubelet, scheduler and controller manager | **tcp** | **10250 - 10259** |
| **MasterIngressWorkerKube** | Kubernetes kubelet, scheduler and controller manager | **tcp** | **10250 - 10259** |
| **MasterIngressIngressServices** | Kubernetes Ingress services | **tcp** | **30000 - 32767** |
| **MasterIngressWorkerIngressServices** | Kubernetes Ingress services | **tcp** | **30000 - 32767** |
| **MasterIngressGeneve** | Geneve packets | **udp** | **6081** |
| **MasterIngressWorkerGeneve** | Geneve packets | **udp** | **6081** |
| **MasterIngressIpsecIke** | IPsec IKE packets | **udp** | **500** |
| **MasterIngressWorkerIpsecIke** | IPsec IKE packets | **udp** | **500** |
| **MasterIngressIpsecNat** | IPsec NAT-T packets | **udp** | **4500** |
| **MasterIngressWorkerIpsecNat** | IPsec NAT-T packets | **udp** | **4500** |
| **MasterIngressIpsecEsp** | IPsec ESP packets | **50** | **All** |

| Ingress group | Description | IP protocol | Port range |
|---|---|---|---|
| **MasterIngressWorkerIpsecEsp** | IPsec ESP packets | **50** | **All** |
| **MasterIngressInternalUDP** | Internal cluster communication | **udp** | **9000 - 9999** |
| **MasterIngressWorkerInternalUDP** | Internal cluster communication | **udp** | **9000 - 9999** |
| **MasterIngressIngressServicesUDP** | Kubernetes Ingress services | **udp** | **30000 - 32767** |
| **MasterIngressWorkerIngressServicesUDP** | Kubernetes Ingress services | **udp** | **30000 - 32767** |

## Worker Ingress

The worker machines require the following Ingress groups. Each Ingress group is a **AWS::EC2::SecurityGroupIngress** resource.

| Ingress group | Description | IP protocol | Port range |
|---|---|---|---|
| **WorkerIngressVxlan** | Vxlan packets | **udp** | **6081** |
| **WorkerIngressWorkerVxlan** | Vxlan packets | **udp** | **6081** |
| **WorkerIngressInternal** | Internal cluster communication | **tcp** | **9000 - 9999** |
| **WorkerIngressWorkerInternal** | Internal cluster communication | **tcp** | **9000 - 9999** |
| **WorkerIngressKube** | Kubernetes kubelet, scheduler, and controller manager | **tcp** | **10250** |
| **WorkerIngressWorkerKube** | Kubernetes kubelet, scheduler, and controller manager | **tcp** | **10250** |

| Ingress group | Description | IP protocol | Port range |
|---|---|---|---|
| **WorkerIngressIngressServices** | Kubernetes Ingress services | **tcp** | **30000 - 32767** |
| **WorkerIngressWorkerIngressServices** | Kubernetes Ingress services | **tcp** | **30000 - 32767** |
| **WorkerIngressGeneve** | Geneve packets | **udp** | **6081** |
| **WorkerIngressMasterGeneve** | Geneve packets | **udp** | **6081** |
| **WorkerIngressIpsecIke** | IPsec IKE packets | **udp** | **500** |
| **WorkerIngressMasterIpsecIke** | IPsec IKE packets | **udp** | **500** |
| **WorkerIngressIpsecNat** | IPsec NAT-T packets | **udp** | **4500** |
| **WorkerIngressMasterIpsecNat** | IPsec NAT-T packets | **udp** | **4500** |
| **WorkerIngressIpsecEsp** | IPsec ESP packets | **50** | **All** |
| **WorkerIngressMasterIpsecEsp** | IPsec ESP packets | **50** | **All** |
| **WorkerIngressInternalUDP** | Internal cluster communication | **udp** | **9000 - 9999** |
| **WorkerIngressMasterInternalUDP** | Internal cluster communication | **udp** | **9000 - 9999** |

| Ingress group | Description | IP protocol | Port range |
|---|---|---|---|
| **WorkerIngressIngressServicesUDP** | Kubernetes Ingress services | **udp** | **30000 - 32767** |
| **WorkerIngressMasterIngressServicesUDP** | Kubernetes Ingress services | **udp** | **30000 - 32767** |

### Roles and instance profiles

You must grant the machines permissions in AWS. The provided CloudFormation templates grant the machines **Allow** permissions for the following **AWS::IAM::Role** objects and provide a **AWS::IAM::InstanceProfile** for each set of roles. If you do not use the templates, you can grant the machines the following broad permissions or the following individual permissions.

| Role | Effect | Action | Resource |
|---|---|---|---|
| Master | **Allow** | **ec2:*** | * |
|  | **Allow** | **elasticloadbalancing:*** | * |
|  | **Allow** | **iam:PassRole** | * |
|  | **Allow** | **s3:GetObject** | * |
| Worker | **Allow** | **ec2:Describe*** | * |
| Bootstrap | **Allow** | **ec2:Describe*** | * |
|  | **Allow** | **ec2:AttachVolume** | * |
|  | **Allow** | **ec2:DetachVolume** | * |

#### 4.2.3.2. Cluster machines

You need **AWS::EC2::Instance** objects for the following machines:

- A bootstrap machine. This machine is required during installation, but you can remove it after your cluster deploys.
- Three control plane machines. The control plane machines are not governed by a control plane machine set.
- Compute machines. You must create at least two compute machines, which are also known as worker machines, during installation. These machines are not governed by a compute machine set.

### 4.2.4. Required AWS permissions for the IAM user

To deploy all components of an OpenShift Container Platform cluster, you must grant the all the required permissions to the IAM user that you create in Amazon Web Services (AWS).

NOTE

Your IAM user must have the permission **tag:GetResources** in the region **us-east-1** to delete the base cluster resources. As part of the AWS API requirement, the OpenShift Container Platform installation program performs various actions in this region.

When you attach the **AdministratorAccess** policy to the IAM user that you create in Amazon Web Services (AWS), you grant that user all of the required permissions. To deploy all components of an OpenShift Container Platform cluster, the IAM user requires the following permissions:

**Example 4.3. Required EC2 permissions for installation**

- **ec2:AttachNetworkInterface**
- **ec2:AuthorizeSecurityGroupEgress**
- **ec2:AuthorizeSecurityGroupIngress**
- **ec2:CopyImage**
- **ec2:CreateNetworkInterface**
- **ec2:CreateSecurityGroup**
- **ec2:CreateTags**
- **ec2:CreateVolume**
- **ec2:DeleteSecurityGroup**
- **ec2:DeleteSnapshot**
- **ec2:DeleteTags**
- **ec2:DeregisterImage**
- **ec2:DescribeAccountAttributes**
- **ec2:DescribeAddresses**
- **ec2:DescribeAvailabilityZones**
- **ec2:DescribeDhcpOptions**
- **ec2:DescribeImages**
- **ec2:DescribeInstanceAttribute**
- **ec2:DescribeInstanceCreditSpecifications**

- **ec2:DescribeInstances**
- **ec2:DescribeInstanceTypes**
- **ec2:DescribeInstanceTypeOfferings**
- **ec2:DescribeInternetGateways**
- **ec2:DescribeKeyPairs**
- **ec2:DescribeNatGateways**
- **ec2:DescribeNetworkAcls**
- **ec2:DescribeNetworkInterfaces**
- **ec2:DescribePrefixLists**
- **ec2:DescribePublicIpv4Pools** (only required if **publicIpv4Pool** is specified in **install-config.yaml**)
- **ec2:DescribeRegions**
- **ec2:DescribeRouteTables**
- **ec2:DescribeSecurityGroupRules**
- **ec2:DescribeSecurityGroups**
- **ec2:DescribeSubnets**
- **ec2:DescribeTags**
- **ec2:DescribeVolumes**
- **ec2:DescribeVpcAttribute**
- **ec2:DescribeVpcClassicLink**
- **ec2:DescribeVpcClassicLinkDnsSupport**
- **ec2:DescribeVpcEndpoints**
- **ec2:DescribeVpcs**
- **ec2:DisassociateAddress** (only required if **publicIpv4Pool** is specified in **install-config.yaml**)
- **ec2:GetEbsDefaultKmsKeyId**
- **ec2:ModifyInstanceAttribute**
- **ec2:ModifyNetworkInterfaceAttribute**
- **ec2:RevokeSecurityGroupEgress**

- **ec2:RevokeSecurityGroupIngress**
- **ec2:RunInstances**
- **ec2:TerminateInstances**

Example 4.4. Required permissions for creating network resources during installation

- **ec2:AllocateAddress**
- **ec2:AssociateAddress**
- **ec2:AssociateDhcpOptions**
- **ec2:AssociateRouteTable**
- **ec2:AttachInternetGateway**
- **ec2:CreateDhcpOptions**
- **ec2:CreateInternetGateway**
- **ec2:CreateNatGateway**
- **ec2:CreateRoute**
- **ec2:CreateRouteTable**
- **ec2:CreateSubnet**
- **ec2:CreateVpc**
- **ec2:CreateVpcEndpoint**
- **ec2:ModifySubnetAttribute**
- **ec2:ModifyVpcAttribute**

NOTE

If you use an existing Virtual Private Cloud (VPC), your account does not require these permissions for creating network resources.

Example 4.5. Required Elastic Load Balancing permissions (ELB) for installation

- **elasticloadbalancing:AddTags**
- **elasticloadbalancing:ApplySecurityGroupsToLoadBalancer**
- **elasticloadbalancing:AttachLoadBalancerToSubnets**
- **elasticloadbalancing:ConfigureHealthCheck**

- **elasticloadbalancing:CreateListener**
- **elasticloadbalancing:CreateLoadBalancer**
- **elasticloadbalancing:CreateLoadBalancerListeners**
- **elasticloadbalancing:CreateTargetGroup**
- **elasticloadbalancing:DeleteLoadBalancer**
- **elasticloadbalancing:DeregisterInstancesFromLoadBalancer**
- **elasticloadbalancing:DeregisterTargets**
- **elasticloadbalancing:DescribeInstanceHealth**
- **elasticloadbalancing:DescribeListeners**
- **elasticloadbalancing:DescribeLoadBalancerAttributes**
- **elasticloadbalancing:DescribeLoadBalancers**
- **elasticloadbalancing:DescribeTags**
- **elasticloadbalancing:DescribeTargetGroupAttributes**
- **elasticloadbalancing:DescribeTargetHealth**
- **elasticloadbalancing:ModifyLoadBalancerAttributes**
- **elasticloadbalancing:ModifyTargetGroup**
- **elasticloadbalancing:ModifyTargetGroupAttributes**
- **elasticloadbalancing:RegisterInstancesWithLoadBalancer**
- **elasticloadbalancing:RegisterTargets**
- **elasticloadbalancing:SetLoadBalancerPoliciesOfListener**
- **elasticloadbalancing:SetSecurityGroups**

IMPORTANT

OpenShift Container Platform uses both the ELB and ELBv2 API services to provision load balancers. The permission list shows permissions required by both services. A known issue exists in the AWS web console where both services use the same **elasticloadbalancing** action prefix but do not recognize the same actions. You can ignore the warnings about the service not recognizing certain **elasticloadbalancing** actions.

Example 4.6. Required IAM permissions for installation

- **iam:AddRoleToInstanceProfile**

- **iam:CreateInstanceProfile**
- **iam:CreateRole**
- **iam:DeleteInstanceProfile**
- **iam:DeleteRole**
- **iam:DeleteRolePolicy**
- **iam:GetInstanceProfile**
- **iam:GetRole**
- **iam:GetRolePolicy**
- **iam:GetUser**
- **iam:ListInstanceProfilesForRole**
- **iam:ListRoles**
- **iam:ListUsers**
- **iam:PassRole**
- **iam:PutRolePolicy**
- **iam:RemoveRoleFromInstanceProfile**
- **iam:SimulatePrincipalPolicy**
- **iam:TagInstanceProfile**
- **iam:TagRole**

NOTE

- If you specify an existing IAM role in the **install-config.yaml** file, the following IAM permissions are not required: **iam:CreateRole**,**iam:DeleteRole**, **iam:DeleteRolePolicy**, and **iam:PutRolePolicy**.
- If you have not created a load balancer in your AWS account, the IAM user also requires the **iam:CreateServiceLinkedRole** permission.

Example 4.7. Required Route 53 permissions for installation

- **route53:ChangeResourceRecordSets**
- **route53:ChangeTagsForResource**
- **route53:CreateHostedZone**
- **route53:DeleteHostedZone**

- **route53:GetChange**
- **route53:GetHostedZone**
- **route53:ListHostedZones**
- **route53:ListHostedZonesByName**
- **route53:ListResourceRecordSets**
- **route53:ListTagsForResource**
- **route53:UpdateHostedZoneComment**

Example 4.8. Required Amazon Simple Storage Service (S3) permissions for installation

- **s3:CreateBucket**
- **s3:DeleteBucket**
- **s3:GetAccelerateConfiguration**
- **s3:GetBucketAcl**
- **s3:GetBucketCors**
- **s3:GetBucketLocation**
- **s3:GetBucketLogging**
- **s3:GetBucketObjectLockConfiguration**
- **s3:GetBucketPolicy**
- **s3:GetBucketRequestPayment**
- **s3:GetBucketTagging**
- **s3:GetBucketVersioning**
- **s3:GetBucketWebsite**
- **s3:GetEncryptionConfiguration**
- **s3:GetLifecycleConfiguration**
- **s3:GetReplicationConfiguration**
- **s3:ListBucket**
- **s3:PutBucketAcl**
- **s3:PutBucketPolicy**
- **s3:PutBucketTagging**

- **s3:PutEncryptionConfiguration**

Example 4.9. S3 permissions that cluster Operators require

- **s3:DeleteObject**
- **s3:GetObject**
- **s3:GetObjectAcl**
- **s3:GetObjectTagging**
- **s3:GetObjectVersion**
- **s3:PutObject**
- **s3:PutObjectAcl**
- **s3:PutObjectTagging**

Example 4.10. Required permissions to delete base cluster resources

- **autoscaling:DescribeAutoScalingGroups**
- **ec2:DeleteNetworkInterface**
- **ec2:DeletePlacementGroup**
- **ec2:DeleteVolume**
- **elasticloadbalancing:DeleteTargetGroup**
- **elasticloadbalancing:DescribeTargetGroups**
- **iam:DeleteAccessKey**
- **iam:DeleteUser**
- **iam:DeleteUserPolicy**
- **iam:ListAttachedRolePolicies**
- **iam:ListInstanceProfiles**
- **iam:ListRolePolicies**
- **iam:ListUserPolicies**
- **s3:DeleteObject**
- **s3:ListBucketVersions**
- **tag:GetResources**

Example 4.11. Required permissions to delete network resources

- **ec2:DeleteDhcpOptions**
- **ec2:DeleteInternetGateway**
- **ec2:DeleteNatGateway**
- **ec2:DeleteRoute**
- **ec2:DeleteRouteTable**
- **ec2:DeleteSubnet**
- **ec2:DeleteVpc**
- **ec2:DeleteVpcEndpoints**
- **ec2:DetachInternetGateway**
- **ec2:DisassociateRouteTable**
- **ec2:ReleaseAddress**
- **ec2:ReplaceRouteTableAssociation**

NOTE

If you use an existing VPC, your account does not require these permissions to delete network resources. Instead, your account only requires the **tag:UntagResources** permission to delete network resources.

Example 4.12. Optional permissions for installing a cluster with a custom Key Management Service (KMS) key

- **kms:CreateGrant**
- **kms:Decrypt**
- **kms:DescribeKey**
- **kms:Encrypt**
- **kms:GenerateDataKey**
- **kms:GenerateDataKeyWithoutPlainText**
- **kms:ListGrants**
- **kms:RevokeGrant**

Example 4.13. Required permissions to delete a cluster with shared instance roles

- **iam:UntagRole**

Example 4.14. Required permissions to delete a cluster with shared instance profiles

- **tag:UntagResources**

Example 4.15. Additional IAM and S3 permissions that are required to create manifests

- **iam:GetUserPolicy**
- **iam:ListAccessKeys**
- **iam:PutUserPolicy**
- **iam:TagUser**
- **s3:AbortMultipartUpload**
- **s3:GetBucketPublicAccessBlock**
- **s3:ListBucket**
- **s3:ListBucketMultipartUploads**
- **s3:PutBucketPublicAccessBlock**
- **s3:PutLifecycleConfiguration**

NOTE

If you are managing your cloud provider credentials with mint mode, the IAM user also requires the **iam:CreateAccessKey** and **iam:CreateUser** permissions.

Example 4.16. Optional permissions for instance and quota checks for installation

- **servicequotas:ListAWSDefaultServiceQuotas**

Example 4.17. Optional permissions for the cluster owner account when installing a cluster on a shared VPC

- **sts:AssumeRole**

Example 4.18. Required permissions for enabling Bring your own public IPv4 addresses (BYOIP) feature for installation

- **ec2:DescribePublicIpv4Pools**

- **ec2:DisassociateAddress**

### 4.2.5. Obtaining an AWS Marketplace image

If you are deploying an OpenShift Container Platform cluster using an AWS Marketplace image, you must first subscribe through AWS. Subscribing to the offer provides you with the AMI ID that the installation program uses to deploy compute nodes.

NOTE

You should only modify the RHCOS image for compute machines to use an AWS Marketplace image. Control plane machines and infrastructure nodes do not require an OpenShift Container Platform subscription and use the public RHCOS default image by default, which does not incur subscription costs on your AWS bill. Therefore, you should not modify the cluster default boot image or the control plane boot images. Applying the AWS Marketplace image to them will incur additional licensing costs that cannot be recovered.

**Prerequisites**

- You have an AWS account to purchase the offer. This account does not have to be the same account that is used to install the cluster.

**Procedure**

1. Complete the OpenShift Container Platform subscription from the AWS Marketplace.

## 4.3. INSTALLING A CLUSTER ON USER-PROVISIONED INFRASTRUCTURE IN AWS BY USING CLOUDFORMATION TEMPLATES

In OpenShift Container Platform version 4.19, you can install a cluster on Amazon Web Services (AWS) that uses infrastructure that you provide.

One way to create this infrastructure is to use the provided CloudFormation templates. You can modify the templates to customize your infrastructure or use the information that they contain to create AWS objects according to your company's policies.

IMPORTANT

The steps for performing a user-provisioned infrastructure installation are provided as an example only. Installing a cluster with infrastructure you provide requires knowledge of the cloud provider and the installation process of OpenShift Container Platform. Several CloudFormation templates are provided to assist in completing these steps or to help model your own. You are also free to create the required resources through other methods; the templates are just an example.

### 4.3.1. Prerequisites

- You reviewed details about the OpenShift Container Platform installation and update processes.

- You read the documentation on selecting a cluster installation method and preparing it for users.
- You configured an AWS account to host the cluster.

IMPORTANT

If you have an AWS profile stored on your computer, it must not use a temporary session token that you generated while using a multi-factor authentication device. The cluster continues to use your current AWS credentials to create AWS resources for the entire life of the cluster, so you must use key-based, long-term credentials. To generate appropriate keys, see Managing Access Keys for IAM Users in the AWS documentation. You can supply the keys when you run the installation program.

- You prepared the user-provisioned infrastructure.
- You downloaded the AWS CLI and installed it on your computer. See Install the AWS CLI Using the Bundled Installer (Linux, macOS, or UNIX) in the AWS documentation.
- If you use a firewall, you configured it to allow the sites that your cluster requires access to.

NOTE

Be sure to also review this site list if you are configuring a proxy.

- If the cloud identity and access management (IAM) APIs are not accessible in your environment, or if you do not want to store an administrator-level credential secret in the **kube-system** namespace, you can manually create and maintain long-term credentials.

### 4.3.2. Creating the installation files for AWS

To install OpenShift Container Platform on Amazon Web Services using user-provisioned infrastructure, you must generate the files that the installation program needs to deploy your cluster and modify them so that the cluster creates only the machines that it will use. You generate and customize the **install-config.yaml** file, Kubernetes manifests, and Ignition config files. You also have the option to first set up a separate **var** partition during the preparation phases of installation.

#### 4.3.2.1. Optional: Creating a separate /var partition

It is recommended that disk partitioning for OpenShift Container Platform be left to the installer. However, there are cases where you might want to create separate partitions in a part of the filesystem that you expect to grow.

OpenShift Container Platform supports the addition of a single partition to attach storage to either the **/var** partition or a subdirectory of **/var**. For example:

- **/var/lib/containers**: Holds container-related content that can grow as more images and containers are added to a system.
- **/var/lib/etcd**: Holds data that you might want to keep separate for purposes such as performance optimization of etcd storage.
- **/var**: Holds data that you might want to keep separate for purposes such as auditing.

Storing the contents of a /**var** directory separately makes it easier to grow storage for those areas as needed and reinstall OpenShift Container Platform at a later date and keep that data intact. With this method, you will not have to pull all your containers again, nor will you have to copy massive log files when you update systems.

Because /**var** must be in place before a fresh installation of Red Hat Enterprise Linux CoreOS (RHCOS), the following procedure sets up the separate /**var** partition by creating a machine config manifest that is inserted during the **openshift-install** preparation phases of an OpenShift Container Platform installation.

IMPORTANT

If you follow the steps to create a separate /**var** partition in this procedure, it is not necessary to create the Kubernetes manifest and Ignition config files again as described later in this section.

### Procedure

1. Create a directory to hold the OpenShift Container Platform installation files:

```
$ mkdir $HOME/clusterconfig
```

2. Run **openshift-install** to create a set of files in the **manifest** and **openshift** subdirectories. Answer the system questions as you are prompted:

```
$ openshift-install create manifests --dir $HOME/clusterconfig
```

#### Example output

```
? SSH Public Key ...
INFO Credentials loaded from the "myprofile" profile in file "/home/myuser/.aws/credentials"
INFO Consuming Install Config from target directory
INFO Manifests created in: $HOME/clusterconfig/manifests and
$HOME/clusterconfig/openshift
```

3. Optional: Confirm that the installation program created manifests in the **clusterconfig/openshift** directory:

```
$ ls $HOME/clusterconfig/openshift/
```

#### Example output

```
99_kubeadmin-password-secret.yaml
99_openshift-cluster-api_master-machines-0.yaml
99_openshift-cluster-api_master-machines-1.yaml
99_openshift-cluster-api_master-machines-2.yaml
...
```

4. Create a Butane config that configures the additional partition. For example, name the file **$HOME/clusterconfig/98-var-partition.bu**, change the disk device name to the name of the storage device on the **worker** systems, and set the storage size as appropriate. This example places the /**var** directory on a separate partition:

```
variant: openshift
version: 4.19.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/disk/by-id/<device_name> (1)
    partitions:
    - label: var
      start_mib: <partition_start_offset> (2)
      size_mib: <partition_size> (3)
      number: 5
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] (4)
      with_mount_unit: true
```

1. The storage device name of the disk that you want to partition.
2. When adding a data partition to the boot disk, a minimum value of 25000 MiB (Mebibytes) is recommended. The root file system is automatically resized to fill all available space up to the specified offset. If no value is specified, or if the specified value is smaller than the recommended minimum, the resulting root file system will be too small, and future reinstalls of RHCOS might overwrite the beginning of the data partition.
3. The size of the data partition in mebibytes.
4. The **prjquota** mount option must be enabled for filesystems used for container storage.

NOTE

When creating a separate **/var** partition, you cannot use different instance types for worker nodes, if the different instance types do not have the same device name.

5. Create a manifest from the Butane config and save it to the **clusterconfig/openshift** directory. For example, run the following command:

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o $HOME/clusterconfig/openshift/98-var-partition.yaml
```

6. Run **openshift-install** again to create Ignition configs from a set of files in the **manifest** and **openshift** subdirectories:

```
$ openshift-install create ignition-configs --dir $HOME/clusterconfig
```

```
$ ls $HOME/clusterconfig/
auth  bootstrap.ign  master.ign  metadata.json  worker.ign
```

You can now use the Ignition config files as input to the installation procedures to install Red Hat Enterprise Linux CoreOS (RHCOS) systems.

#### 4.3.2.2. Creating the installation configuration file

Generate and customize the installation configuration file that the installation program needs to deploy your cluster.

##### Prerequisites

- You obtained the OpenShift Container Platform installation program for user-provisioned infrastructure and the pull secret for your cluster.
- You checked that you are deploying your cluster to an AWS Region with an accompanying Red Hat Enterprise Linux CoreOS (RHCOS) AMI published by Red Hat. If you are deploying to an AWS Region that requires a custom AMI, such as an AWS GovCloud Region, you must create the **install-config.yaml** file manually.

##### Procedure

1. Create the **install-config.yaml** file.

   a. Change to the directory that contains the installation program and run the following command:

   ```
   $ ./openshift-install create install-config --dir <installation_directory> 1
   ```

   1 For **<installation_directory>**, specify the directory name to store the files that the installation program creates.

   IMPORTANT

   Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

   b. At the prompts, provide the configuration details for your cloud:

      i. Optional: Select an SSH key to use to access your cluster machines.

      NOTE

      For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

      ii. Select **aws** as the platform to target.

iii. If you do not have an AWS profile stored on your computer, enter the AWS access key ID and secret access key for the user that you configured to run the installation program.

> NOTE
>
> The AWS access key ID and secret access key are stored in **~/.aws/credentials** in the home directory of the current user on the installation host. You are prompted for the credentials by the installation program if the credentials for the exported profile are not present in the file. Any credentials that you provide to the installation program are stored in the file.

iv. Select the AWS Region to deploy the cluster to.

v. Select the base domain for the Route 53 service that you configured for your cluster.

vi. Enter a descriptive name for your cluster.

vii. Paste the pull secret from Red Hat OpenShift Cluster Manager .

2. If you are installing a three-node cluster, modify the **install-config.yaml** file by setting the **compute.replicas** parameter to **0**. This ensures that the cluster's control planes are schedulable. For more information, see "Installing a three-node cluster on AWS".

3. Optional: Back up the **install-config.yaml** file.

> IMPORTANT
>
> The **install-config.yaml** file is consumed during the installation process. If you want to reuse the file, you must back it up now.

### Additional resources

- See Configuration and credential file settings in the AWS documentation for more information about AWS profile and credential configuration.

### 4.3.2.3. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

### Prerequisites

- You have an existing **install-config.yaml** file.

- You reviewed the sites that your cluster requires access to and determined whether any of them need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. You added sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.

NOTE

The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.

For installations on Amazon Web Services (AWS), Google Cloud, Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

## Procedure

1. Edit your **install-config.yaml** file and add the proxy settings. For example:

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: ec2.<aws_region>.amazonaws.com,elasticloadbalancing.
<aws_region>.amazonaws.com,s3.<aws_region>.amazonaws.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1 A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**.

2 A proxy URL to use for creating HTTPS connections outside the cluster.

3 A comma-separated list of destination domain names, IP addresses, or other network CIDRs to exclude from proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use * to bypass the proxy for all destinations. If you have added the Amazon **EC2**,**Elastic Load Balancing**, and **S3** VPC endpoints to your VPC, you must add these endpoints to the **noProxy** field.

4 If provided, the installation program generates a config map that is named **user-ca-bundle** in the **openshift-config** namespace that contains one or more additional CA certificates that are required for proxying HTTPS connections. The Cluster Network Operator then creates a **trusted-ca-bundle** config map that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this config map is referenced in the **trustedCA** field of the **Proxy** object. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle.

5 Optional: The policy to determine the configuration of the **Proxy** object to reference the **user-ca-bundle** config map in the **trustedCA** field. The allowed values are **Proxyonly** and **Always**. Use **Proxyonly** to reference the **user-ca-bundle** config map only when **http/https** proxy is configured. Use **Always** to always reference the **user-ca-bundle** config map. The default value is **Proxyonly**.

NOTE

The installation program does not support the proxy **readinessEndpoints** field.

NOTE

If the installer times out, restart and then complete the deployment by using the **wait-for** command of the installer. For example:

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. Save the file and reference it when installing OpenShift Container Platform.

The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster Proxy** object is still created, but it will have a nil **spec**.

NOTE

Only the **Proxy** object named **cluster** is supported, and no additional proxies can be created.

### 4.3.2.4. Creating the Kubernetes manifest and Ignition config files

Because you must modify some cluster definition files and manually start the cluster machines, you must generate the Kubernetes manifest and Ignition config files that the cluster needs to configure the machines.

The installation configuration file transforms into the Kubernetes manifests. The manifests wrap into the Ignition configuration files, which are later used to configure the cluster machines.

IMPORTANT

- The Ignition config files that the OpenShift Container Platform installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

- It is recommended that you use Ignition config files within 12 hours after they are generated because the 24-hour certificate rotates from 16 to 22 hours after the cluster is installed. By using the Ignition config files within 12 hours, you can avoid installation failure if the certificate update runs during installation.

#### Prerequisites

- You obtained the OpenShift Container Platform installation program.
- You created the **install-config.yaml** installation configuration file.

## Procedure

1. Change to the directory that contains the OpenShift Container Platform installation program and generate the Kubernetes manifests for the cluster:

```
$ ./openshift-install create manifests --dir <installation_directory> 1
```

1 For **<installation_directory>**, specify the installation directory that contains the **install-config.yaml** file you created.

2. Remove the Kubernetes manifest files that define the control plane machines:

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

By removing these files, you prevent the cluster from automatically generating control plane machines.

3. Remove the Kubernetes manifest files that define the control plane machine set:

```
$ rm -f <installation_directory>/openshift/99_openshift-machine-api_master-control-plane-machine-set.yaml
```

4. Remove the Kubernetes manifest files that define the worker machines:

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

IMPORTANT

If you disabled the **MachineAPI** capability when installing a cluster on user-provisioned infrastructure, you must remove the Kubernetes manifest files that define the worker machines. Otherwise, your cluster fails to install.

Because you create and manage the worker machines yourself, you do not need to initialize these machines.

WARNING

If you are installing a three-node cluster, skip the following step to allow the control plane nodes to be schedulable.

IMPORTANT

When you configure control plane nodes from the default unschedulable to schedulable, additional subscriptions are required. This is because control plane nodes then become compute nodes.

5. Check that the **mastersSchedulable** parameter in the

**<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes manifest file is set to **false**. This setting prevents pods from being scheduled on the control plane machines:

a. Open the **<installation_directory>/manifests/cluster-scheduler-02-config.yml** file.

b. Locate the **mastersSchedulable** parameter and ensure that it is set to **false**.

c. Save and exit the file.

6. Optional: If you do not want the Ingress Operator to create DNS records on your behalf, remove the **privateZone** and **publicZone** sections from the **<installation_directory>/manifests/cluster-dns-02-config.yml** DNS configuration file:

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: (1)
    id: mycluster-100419-private-zone
  publicZone: (2)
    id: example.openshift.com
status: {}
```

(1) (2) Remove this section completely.

If you do so, you must add ingress DNS records manually in a later step.

7. To create the Ignition configuration files, run the following command from the directory that contains the installation program:

```
$ ./openshift-install create ignition-configs --dir <installation_directory> (1)
```

(1) For **<installation_directory>**, specify the same installation directory.

Ignition config files are created for the bootstrap, control plane, and compute nodes in the installation directory. The **kubeadmin-password** and **kubeconfig** files are created in the **./<installation_directory>/auth** directory:

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

### 4.3.3. Extracting the infrastructure name

The Ignition config files contain a unique cluster identifier that you can use to uniquely identify your cluster in Amazon Web Services. The infrastructure name is also used to locate the appropriate AWS resources during an OpenShift Container Platform installation. The provided CloudFormation templates contain references to this infrastructure name, so you must extract it.

#### Prerequisites

- You obtained the OpenShift Container Platform installation program and the pull secret for your cluster.
- You generated the Ignition config files for your cluster.
- You installed the **jq** package.

#### Procedure

- To extract and view the infrastructure name from the Ignition config file metadata, run the following command:

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

#### Example output

```
openshift-vw9j6 1
```

1 The output of this command is your cluster name and a random string.

### 4.3.4. Creating a VPC in AWS

You must create a Virtual Private Cloud (VPC) in Amazon Web Services (AWS) for your OpenShift Container Platform cluster to use. You can customize the VPC to meet your requirements, including VPN and route tables.

You can use the provided CloudFormation template and a custom parameter file to create a stack of AWS resources that represent the VPC.

NOTE

If you do not use the provided CloudFormation template to create your AWS infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

#### Prerequisites

- You configured an AWS account.
- You added your AWS keys and region to your local AWS profile by running **aws configure**.

- You generated the Ignition config files for your cluster.

**Procedure**

1. Create a JSON file that contains the parameter values that the template requires:

```
[
  {
    "ParameterKey": "VpcCidr", 1
    "ParameterValue": "10.0.0.0/16" 2
  },
  {
    "ParameterKey": "AvailabilityZoneCount", 3
    "ParameterValue": "1" 4
  },
  {
    "ParameterKey": "SubnetBits", 5
    "ParameterValue": "12" 6
  }
]
```

1 The CIDR block for the VPC.

2 Specify a CIDR block in the format **x.x.x.x/16-24**.

3 The number of availability zones to deploy the VPC in.

4 Specify an integer between **1** and **3**.

5 The size of each subnet in each availability zone.

6 Specify an integer between **5** and **13**, where **5** is **/27** and **13** is **/19**.

2. Copy the template from the **CloudFormation template for the VPC**section of this topic and save it as a YAML file on your computer. This template describes the VPC that your cluster requires.

3. Launch the CloudFormation template to create a stack of AWS resources that represent the VPC:

IMPORTANT

You must enter the command on a single line.

```
$ aws cloudformation create-stack --stack-name <name> 1
     --template-body file://<template>.yaml 2
     --parameters file://<parameters>.json 3
```

1 **<name>** is the name for the CloudFormation stack, such as **cluster-vpc**. You need the name of this stack if you remove the cluster.

2

**<template>** is the relative path to and name of the CloudFormation template YAML file that you saved.

3 **<parameters>** is the relative path to and name of the CloudFormation parameters JSON file.

#### Example output

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-vpc/dbedae40-2fd3-11eb-
820e-12a48460849f
```

4. Confirm that the template components exist:

```
$ aws cloudformation describe-stacks --stack-name <name>
```

After the **StackStatus** displays **CREATE_COMPLETE**, the output displays values for the following parameters. You must provide these parameter values to the other CloudFormation templates that you run to create your cluster:

| | |
|---|---|
| **VpcId** | The ID of your VPC. |
| **PublicSubnetIds** | The IDs of the new public subnets. |
| **PrivateSubnetIds** | The IDs of the new private subnets. |

### 4.3.4.1. CloudFormation template for the VPC

You can use the following CloudFormation template to deploy the VPC that you need for your OpenShift Container Platform cluster.

**Example 4.19. CloudFormation template for the VPC**

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for Best Practice VPC with 1-3 AZs

Parameters:
  VpcCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-
4][0-9]|25[0-5])(\/(1[6-9]|2[0-4]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/16-24.
    Default: 10.0.0.0/16
    Description: CIDR block for VPC.
    Type: String
  AvailabilityZoneCount:
    ConstraintDescription: "The number of availability zones. (Min: 1, Max: 3)"
    MinValue: 1
    MaxValue: 3
    Default: 1
    Description: "How many AZs to create VPC subnets for. (Min: 1, Max: 3)"
    Type: Number
  SubnetBits:
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/19-27.
    MinValue: 5
    MaxValue: 13
    Default: 12
    Description: "Size of each subnet to create within the availability zones. (Min: 5 = /27, Max: 13 =
/19)"
    Type: Number

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcCidr
      - SubnetBits
    - Label:
        default: "Availability Zones"
      Parameters:
      - AvailabilityZoneCount
    ParameterLabels:
      AvailabilityZoneCount:
        default: "Availability Zone Count"
      VpcCidr:
        default: "VPC CIDR"
      SubnetBits:
        default: "Bits Per Subnet"

Conditions:
  DoAz3: !Equals [3, !Ref AvailabilityZoneCount]
  DoAz2: !Or [!Equals [2, !Ref AvailabilityZoneCount], Condition: DoAz3]

Resources:
  VPC:
    Type: "AWS::EC2::VPC"
    Properties:
      EnableDnsSupport: "true"
      EnableDnsHostnames: "true"
      CidrBlock: !Ref VpcCidr
  PublicSubnet:
    Type: "AWS::EC2::Subnet"
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [0, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 0
      - Fn::GetAZs: !Ref "AWS::Region"
  PublicSubnet2:
    Type: "AWS::EC2::Subnet"
    Condition: DoAz2
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [1, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 1
      - Fn::GetAZs: !Ref "AWS::Region"
  PublicSubnet3:
    Type: "AWS::EC2::Subnet"
    Condition: DoAz3
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [2, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 2
      - Fn::GetAZs: !Ref "AWS::Region"
  InternetGateway:
    Type: "AWS::EC2::InternetGateway"
  GatewayToInternet:
    Type: "AWS::EC2::VPCGatewayAttachment"
    Properties:
      VpcId: !Ref VPC
      InternetGatewayId: !Ref InternetGateway
  PublicRouteTable:
    Type: "AWS::EC2::RouteTable"
    Properties:
      VpcId: !Ref VPC
  PublicRoute:
    Type: "AWS::EC2::Route"
    DependsOn: GatewayToInternet
    Properties:
      RouteTableId: !Ref PublicRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId: !Ref InternetGateway
  PublicSubnetRouteTableAssociation:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Properties:
      SubnetId: !Ref PublicSubnet
      RouteTableId: !Ref PublicRouteTable
  PublicSubnetRouteTableAssociation2:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Condition: DoAz2
    Properties:
      SubnetId: !Ref PublicSubnet2
      RouteTableId: !Ref PublicRouteTable
  PublicSubnetRouteTableAssociation3:
    Condition: DoAz3
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Properties:
      SubnetId: !Ref PublicSubnet3
      RouteTableId: !Ref PublicRouteTable
  PrivateSubnet:
    Type: "AWS::EC2::Subnet"
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [3, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 0
      - Fn::GetAZs: !Ref "AWS::Region"
  PrivateRouteTable:
    Type: "AWS::EC2::RouteTable"
    Properties:
      VpcId: !Ref VPC
  PrivateSubnetRouteTableAssociation:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Properties:
      SubnetId: !Ref PrivateSubnet
      RouteTableId: !Ref PrivateRouteTable
  NAT:
    DependsOn:
    - GatewayToInternet
    Type: "AWS::EC2::NatGateway"
    Properties:
      AllocationId:
        "Fn::GetAtt":
        - EIP
        - AllocationId
      SubnetId: !Ref PublicSubnet
  EIP:
    Type: "AWS::EC2::EIP"
    Properties:
      Domain: vpc
  Route:
    Type: "AWS::EC2::Route"
    Properties:
      RouteTableId:
        Ref: PrivateRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId:
        Ref: NAT
  PrivateSubnet2:
    Type: "AWS::EC2::Subnet"
    Condition: DoAz2
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [4, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 1
      - Fn::GetAZs: !Ref "AWS::Region"
  PrivateRouteTable2:
    Type: "AWS::EC2::RouteTable"
    Condition: DoAz2
    Properties:
      VpcId: !Ref VPC
  PrivateSubnetRouteTableAssociation2:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Condition: DoAz2
    Properties:
      SubnetId: !Ref PrivateSubnet2
      RouteTableId: !Ref PrivateRouteTable2
  NAT2:
    DependsOn:
    - GatewayToInternet
    Type: "AWS::EC2::NatGateway"
    Condition: DoAz2
    Properties:
      AllocationId:
          "Fn::GetAtt":
          - EIP2
          - AllocationId
        SubnetId: !Ref PublicSubnet2
    EIP2:
      Type: "AWS::EC2::EIP"
      Condition: DoAz2
      Properties:
        Domain: vpc
    Route2:
      Type: "AWS::EC2::Route"
      Condition: DoAz2
      Properties:
        RouteTableId:
          Ref: PrivateRouteTable2
        DestinationCidrBlock: 0.0.0.0/0
        NatGatewayId:
          Ref: NAT2
    PrivateSubnet3:
      Type: "AWS::EC2::Subnet"
      Condition: DoAz3
      Properties:
        VpcId: !Ref VPC
        CidrBlock: !Select [5, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
        AvailabilityZone: !Select
        - 2
        - Fn::GetAZs: !Ref "AWS::Region"
    PrivateRouteTable3:
      Type: "AWS::EC2::RouteTable"
      Condition: DoAz3
      Properties:
        VpcId: !Ref VPC
    PrivateSubnetRouteTableAssociation3:
      Type: "AWS::EC2::SubnetRouteTableAssociation"
      Condition: DoAz3
      Properties:
        SubnetId: !Ref PrivateSubnet3
        RouteTableId: !Ref PrivateRouteTable3
    NAT3:
      DependsOn:
      - GatewayToInternet
      Type: "AWS::EC2::NatGateway"
      Condition: DoAz3
      Properties:
        AllocationId:
          "Fn::GetAtt":
          - EIP3
          - AllocationId
        SubnetId: !Ref PublicSubnet3
    EIP3:
      Type: "AWS::EC2::EIP"
      Condition: DoAz3
      Properties:
        Domain: vpc
    Route3:
      Type: "AWS::EC2::Route"
    Condition: DoAz3
    Properties:
      RouteTableId:
        Ref: PrivateRouteTable3
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId:
        Ref: NAT3
  S3Endpoint:
    Type: AWS::EC2::VPCEndpoint
    Properties:
      PolicyDocument:
        Version: 2012-10-17
        Statement:
        - Effect: Allow
          Principal: '*'
          Action:
          - '*'
          Resource:
          - '*'
      RouteTableIds:
      - !Ref PublicRouteTable
      - !Ref PrivateRouteTable
      - !If [DoAz2, !Ref PrivateRouteTable2, !Ref "AWS::NoValue"]
      - !If [DoAz3, !Ref PrivateRouteTable3, !Ref "AWS::NoValue"]
      ServiceName: !Join
      - ''
      - - com.amazonaws.
        - !Ref 'AWS::Region'
        - .s3
      VpcId: !Ref VPC

Outputs:
  VpcId:
    Description: ID of the new VPC.
    Value: !Ref VPC
  PublicSubnetIds:
    Description: Subnet IDs of the public subnets.
    Value:
      !Join [
        ",",
        [!Ref PublicSubnet, !If [DoAz2, !Ref PublicSubnet2, !Ref "AWS::NoValue"], !If [DoAz3, !Ref
PublicSubnet3, !Ref "AWS::NoValue"]]
      ]
  PrivateSubnetIds:
    Description: Subnet IDs of the private subnets.
    Value:
      !Join [
        ",",
        [!Ref PrivateSubnet, !If [DoAz2, !Ref PrivateSubnet2, !Ref "AWS::NoValue"], !If [DoAz3, !Ref
PrivateSubnet3, !Ref "AWS::NoValue"]]
      ]
  PublicRouteTableId:
    Description: Public Route table ID
    Value: !Ref PublicRouteTable
  PrivateRouteTableIds:
    Description: Private Route table IDs
```

```
Value:
  !Join [
    ",",
    [
      !Join ["=", [
        !Select [0, "Fn::GetAZs": !Ref "AWS::Region"],
        !Ref PrivateRouteTable
      ]],
      !If [DoAz2,
          !Join ["=", [!Select [1, "Fn::GetAZs": !Ref "AWS::Region"], !Ref PrivateRouteTable2]],
          !Ref "AWS::NoValue"
      ],
      !If [DoAz3,
          !Join ["=", [!Select [2, "Fn::GetAZs": !Ref "AWS::Region"], !Ref PrivateRouteTable3]],
          !Ref "AWS::NoValue"
      ]
    ]
  ]
```

**Additional resources**

- You can view details about the CloudFormation stacks that you create by navigating to the AWS CloudFormation console.

### 4.3.5. Creating networking and load balancing components in AWS

You must configure networking and classic or network load balancing in Amazon Web Services (AWS) that your OpenShift Container Platform cluster can use.

You can use the provided CloudFormation template and a custom parameter file to create a stack of AWS resources. The stack represents the networking and load balancing components that your OpenShift Container Platform cluster requires. The template also creates a hosted zone and subnet tags.

You can run the template multiple times within a single Virtual Private Cloud (VPC).

**NOTE**

If you do not use the provided CloudFormation template to create your AWS infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

**Prerequisites**

- You configured an AWS account.
- You added your AWS keys and region to your local AWS profile by running **aws configure**.
- You generated the Ignition config files for your cluster.
- You created and configured a VPC and associated subnets in AWS.

## Procedure

1. Obtain the hosted zone ID for the Route 53 base domain that you specified in the **install-config.yaml** file for your cluster. You can obtain details about your hosted zone by running the following command:

   ```
   $ aws route53 list-hosted-zones-by-name --dns-name <route53_domain> (1)
   ```

   (1) For the **<route53_domain>**, specify the Route 53 base domain that you used when you generated the **install-config.yaml** file for the cluster.

   **Example output**

   ```
   mycluster.example.com. False 100
   HOSTEDZONES 65F8F38E-2268-B835-E15C-AB55336FCBFA
   /hostedzone/Z21IXYZABCZ2A4 mycluster.example.com. 10
   ```

   In the example output, the hosted zone ID is **Z21IXYZABCZ2A4**.

2. Create a JSON file that contains the parameter values that the template requires:

   ```
   [
     {
       "ParameterKey": "ClusterName", (1)
       "ParameterValue": "mycluster" (2)
     },
     {
       "ParameterKey": "InfrastructureName", (3)
       "ParameterValue": "mycluster-<random_string>" (4)
     },
     {
       "ParameterKey": "HostedZoneId", (5)
       "ParameterValue": "<random_string>" (6)
     },
     {
       "ParameterKey": "HostedZoneName", (7)
       "ParameterValue": "example.com" (8)
     },
     {
       "ParameterKey": "PublicSubnets", (9)
       "ParameterValue": "subnet-<random_string>" (10)
     },
     {
       "ParameterKey": "PrivateSubnets", (11)
       "ParameterValue": "subnet-<random_string>" (12)
     },
     {
       "ParameterKey": "VpcId", (13)
       "ParameterValue": "vpc-<random_string>" (14)
     }
   ]
   ```

1 A short, representative cluster name to use for hostnames, etc.

2 Specify the cluster name that you used when you generated the **install-config.yaml** file for the cluster.

3 The name for your cluster infrastructure that is encoded in your Ignition config files for the cluster.

4 Specify the infrastructure name that you extracted from the Ignition config file metadata, which has the format **<cluster-name>-<random-string>**.

5 The Route 53 public zone ID to register the targets with.

6 Specify the Route 53 public zone ID, which as a format similar to **Z21IXYZABCZ2A4**. You can obtain this value from the AWS console.

7 The Route 53 zone to register the targets with.

8 Specify the Route 53 base domain that you used when you generated the **install-config.yaml** file for the cluster. Do not include the trailing period (.) that is displayed in the AWS console.

9 The public subnets that you created for your VPC.

10 Specify the **PublicSubnetIds** value from the output of the CloudFormation template for the VPC.

11 The private subnets that you created for your VPC.

12 Specify the **PrivateSubnetIds** value from the output of the CloudFormation template for the VPC.

13 The VPC that you created for the cluster.

14 Specify the **VpcId** value from the output of the CloudFormation template for the VPC.

3. Copy the template from the **CloudFormation template for the network and load balancers** section of this topic and save it as a YAML file on your computer. This template describes the networking and load balancing objects that your cluster requires.

   IMPORTANT

   If you are deploying your cluster to an AWS government or secret region, you must update the **InternalApiServerRecord** in the CloudFormation template to use **CNAME** records. Records of type **ALIAS** are not supported for AWS government regions.

4. Launch the CloudFormation template to create a stack of AWS resources that provide the networking and load balancing components:

   IMPORTANT

   You must enter the command on a single line.

```
$ aws cloudformation create-stack --stack-name <name> (1)
    --template-body file://<template>.yaml (2)
    --parameters file://<parameters>.json (3)
    --capabilities CAPABILITY_NAMED_IAM (4)
```

1. **<name>** is the name for the CloudFormation stack, such as **cluster-dns**. You need the name of this stack if you remove the cluster.
2. **<template>** is the relative path to and name of the CloudFormation template YAML file that you saved.
3. **<parameters>** is the relative path to and name of the CloudFormation parameters JSON file.
4. You must explicitly declare the **CAPABILITY_NAMED_IAM** capability because the provided template creates some **AWS::IAM::Role** resources.

### Example output

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-dns/cd3e5de0-2fd4-11eb-
5cf0-12be5c33a183
```

5. Confirm that the template components exist:

```
$ aws cloudformation describe-stacks --stack-name <name>
```

After the **StackStatus** displays **CREATE_COMPLETE**, the output displays values for the following parameters. You must provide these parameter values to the other CloudFormation templates that you run to create your cluster:

| | |
|---|---|
| **PrivateHostedZoneId** | Hosted zone ID for the private DNS. |
| **ExternalApiLoadBalancerName** | Full name of the external API load balancer. |
| **InternalApiLoadBalancerName** | Full name of the internal API load balancer. |
| **ApiServerDnsName** | Full hostname of the API server. |
| **RegisterNlbIpTargetsLambda** | Lambda ARN useful to help register/deregister IP targets for these load balancers. |

| | |
|---|---|
| **ExternalApiTargetGroupArn** | ARN of external API target group. |
| **InternalApiTargetGroupArn** | ARN of internal API target group. |
| **InternalServiceTargetGroupArn** | ARN of internal service target group. |

### 4.3.5.1. CloudFormation template for the network and load balancers

You can use the following CloudFormation template to deploy the networking objects and load balancers that you need for your OpenShift Container Platform cluster.

**Example 4.20. CloudFormation template for the network and load balancers**

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Network Elements (Route53 & LBs)

Parameters:
  ClusterName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Cluster name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, representative cluster name to use for host names and other identifying
names.
    Type: String
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, unique cluster ID used to tag cloud resources and identify items owned or
used by the cluster.
    Type: String
  HostedZoneId:
    Description: The Route53 public zone ID to register the targets with, such as
Z21IXYZABCZ2A4.
    Type: String
  HostedZoneName:
    Description: The Route53 zone to register the targets with, such as example.com. Omit the
trailing period.
    Type: String
    Default: "example.com"
  PublicSubnets:
    Description: The internet-facing subnets.
    Type: List<AWS::EC2::Subnet::Id>
  PrivateSubnets:
    Description: The internal subnets.
    Type: List<AWS::EC2::Subnet::Id>
  VpcId:
    Description: The VPC-scoped resources will belong to this VPC.
    Type: AWS::EC2::VPC::Id

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - ClusterName
      - InfrastructureName
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcId
      - PublicSubnets
      - PrivateSubnets
    - Label:
        default: "DNS"
      Parameters:
      - HostedZoneName
      - HostedZoneId
    ParameterLabels:
      ClusterName:
        default: "Cluster Name"
      InfrastructureName:
        default: "Infrastructure Name"
      VpcId:
        default: "VPC ID"
      PublicSubnets:
        default: "Public Subnets"
      PrivateSubnets:
        default: "Private Subnets"
      HostedZoneName:
        default: "Public Hosted Zone Name"
      HostedZoneId:
        default: "Public Hosted Zone ID"

Resources:
  ExtApiElb:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Name: !Join ["-", [!Ref InfrastructureName, "ext"]]
      IpAddressType: ipv4
      Subnets: !Ref PublicSubnets
      Type: network

  IntApiElb:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Name: !Join ["-", [!Ref InfrastructureName, "int"]]
      Scheme: internal
      IpAddressType: ipv4
      Subnets: !Ref PrivateSubnets
      Type: network

  IntDns:
    Type: "AWS::Route53::HostedZone"
    Properties:
      HostedZoneConfig:
        Comment: "Managed by CloudFormation"
      Name: !Join [".", [!Ref ClusterName, !Ref HostedZoneName]]
      HostedZoneTags:
      - Key: Name
        Value: !Join ["-", [!Ref InfrastructureName, "int"]]
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "owned"
      VPCs:
      - VPCId: !Ref VpcId
        VPCRegion: !Ref "AWS::Region"

  ExternalApiServerRecord:
    Type: AWS::Route53::RecordSetGroup
    Properties:
      Comment: Alias record for the API server
      HostedZoneId: !Ref HostedZoneId
      RecordSets:
      - Name:
          !Join [
            ".",
            ["api", !Ref ClusterName, !Join ["", [!Ref HostedZoneName, "."]]],
          ]
        Type: A
        AliasTarget:
          HostedZoneId: !GetAtt ExtApiElb.CanonicalHostedZoneID
          DNSName: !GetAtt ExtApiElb.DNSName

  InternalApiServerRecord:
    Type: AWS::Route53::RecordSetGroup
    Properties:
      Comment: Alias record for the API server
      HostedZoneId: !Ref IntDns
      RecordSets:
      - Name:
          !Join [
            ".",
            ["api", !Ref ClusterName, !Join ["", [!Ref HostedZoneName, "."]]],
          ]
        Type: A
        AliasTarget:
          HostedZoneId: !GetAtt IntApiElb.CanonicalHostedZoneID
          DNSName: !GetAtt IntApiElb.DNSName
      - Name:
          !Join [
            ".",
          ["api-int", !Ref ClusterName, !Join ["", [!Ref HostedZoneName, "."]]],
        ]
      Type: A
      AliasTarget:
        HostedZoneId: !GetAtt IntApiElb.CanonicalHostedZoneID
        DNSName: !GetAtt IntApiElb.DNSName

ExternalApiListener:
  Type: AWS::ElasticLoadBalancingV2::Listener
  Properties:
    DefaultActions:
    - Type: forward
      TargetGroupArn:
        Ref: ExternalApiTargetGroup
    LoadBalancerArn:
      Ref: ExtApiElb
    Port: 6443
    Protocol: TCP

ExternalApiTargetGroup:
  Type: AWS::ElasticLoadBalancingV2::TargetGroup
  Properties:
    HealthCheckIntervalSeconds: 10
    HealthCheckPath: "/readyz"
    HealthCheckPort: 6443
    HealthCheckProtocol: HTTPS
    HealthyThresholdCount: 2
    UnhealthyThresholdCount: 2
    Port: 6443
    Protocol: TCP
    TargetType: ip
    VpcId:
      Ref: VpcId
    TargetGroupAttributes:
    - Key: deregistration_delay.timeout_seconds
      Value: 60

InternalApiListener:
  Type: AWS::ElasticLoadBalancingV2::Listener
  Properties:
    DefaultActions:
    - Type: forward
      TargetGroupArn:
        Ref: InternalApiTargetGroup
    LoadBalancerArn:
      Ref: IntApiElb
    Port: 6443
    Protocol: TCP

InternalApiTargetGroup:
  Type: AWS::ElasticLoadBalancingV2::TargetGroup
  Properties:
    HealthCheckIntervalSeconds: 10
    HealthCheckPath: "/readyz"
    HealthCheckPort: 6443
    HealthCheckProtocol: HTTPS
```

```
      HealthyThresholdCount: 2
      UnhealthyThresholdCount: 2
      Port: 6443
      Protocol: TCP
      TargetType: ip
      VpcId:
        Ref: VpcId
      TargetGroupAttributes:
      - Key: deregistration_delay.timeout_seconds
        Value: 60

  InternalServiceInternalListener:
    Type: AWS::ElasticLoadBalancingV2::Listener
    Properties:
      DefaultActions:
      - Type: forward
        TargetGroupArn:
          Ref: InternalServiceTargetGroup
      LoadBalancerArn:
        Ref: IntApiElb
      Port: 22623
      Protocol: TCP

  InternalServiceTargetGroup:
    Type: AWS::ElasticLoadBalancingV2::TargetGroup
    Properties:
      HealthCheckIntervalSeconds: 10
      HealthCheckPath: "/healthz"
      HealthCheckPort: 22623
      HealthCheckProtocol: HTTPS
      HealthyThresholdCount: 2
      UnhealthyThresholdCount: 2
      Port: 22623
      Protocol: TCP
      TargetType: ip
      VpcId:
        Ref: VpcId
      TargetGroupAttributes:
      - Key: deregistration_delay.timeout_seconds
        Value: 60

  RegisterTargetLambdaIamRole:
    Type: AWS::IAM::Role
    Properties:
      RoleName: !Join ["-", [!Ref InfrastructureName, "nlb", "lambda", "role"]]
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Principal:
            Service:
            - "lambda.amazonaws.com"
          Action:
          - "sts:AssumeRole"
      Path: "/"
      Policies:
          - PolicyName: !Join ["-", [!Ref InfrastructureName, "master", "policy"]]
            PolicyDocument:
              Version: "2012-10-17"
              Statement:
              - Effect: "Allow"
                Action:
                  [
                    "elasticloadbalancing:RegisterTargets",
                    "elasticloadbalancing:DeregisterTargets",
                  ]
                Resource: !Ref InternalApiTargetGroup
              - Effect: "Allow"
                Action:
                  [
                    "elasticloadbalancing:RegisterTargets",
                    "elasticloadbalancing:DeregisterTargets",
                  ]
                Resource: !Ref InternalServiceTargetGroup
              - Effect: "Allow"
                Action:
                  [
                    "elasticloadbalancing:RegisterTargets",
                    "elasticloadbalancing:DeregisterTargets",
                  ]
                Resource: !Ref ExternalApiTargetGroup

  RegisterNlbIpTargets:
    Type: "AWS::Lambda::Function"
    Properties:
      Handler: "index.handler"
      Role:
        Fn::GetAtt:
        - "RegisterTargetLambdaIamRole"
        - "Arn"
      Code:
        ZipFile: |
          import json
          import boto3
          import cfnresponse
          def handler(event, context):
            elb = boto3.client('elbv2')
            if event['RequestType'] == 'Delete':
              elb.deregister_targets(TargetGroupArn=event['ResourceProperties']
['TargetArn'],Targets=[{'Id': event['ResourceProperties']['TargetIp']}])
            elif event['RequestType'] == 'Create':
              elb.register_targets(TargetGroupArn=event['ResourceProperties']['TargetArn'],Targets=
[{'Id': event['ResourceProperties']['TargetIp']}])
            responseData = {}
            cfnresponse.send(event, context, cfnresponse.SUCCESS, responseData,
event['ResourceProperties']['TargetArn']+event['ResourceProperties']['TargetIp'])
      Runtime: "python3.11"
      Timeout: 120

  RegisterSubnetTagsLambdaIamRole:
    Type: AWS::IAM::Role
    Properties:
      RoleName: !Join ["-", [!Ref InfrastructureName, "subnet-tags-lambda-role"]]
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Principal:
            Service:
            - "lambda.amazonaws.com"
          Action:
          - "sts:AssumeRole"
      Path: "/"
      Policies:
      - PolicyName: !Join ["-", [!Ref InfrastructureName, "subnet-tagging-policy"]]
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
          - Effect: "Allow"
            Action:
              [
                "ec2:DeleteTags",
                "ec2:CreateTags"
              ]
            Resource: "arn:aws:ec2:*:*:subnet/*"
          - Effect: "Allow"
            Action:
              [
                "ec2:DescribeSubnets",
                "ec2:DescribeTags"
              ]
            Resource: "*"

  RegisterSubnetTags:
    Type: "AWS::Lambda::Function"
    Properties:
      Handler: "index.handler"
      Role:
        Fn::GetAtt:
        - "RegisterSubnetTagsLambdaIamRole"
        - "Arn"
      Code:
        ZipFile: |
          import json
          import boto3
          import cfnresponse
          def handler(event, context):
            ec2_client = boto3.client('ec2')
            if event['RequestType'] == 'Delete':
              for subnet_id in event['ResourceProperties']['Subnets']:
                ec2_client.delete_tags(Resources=[subnet_id], Tags=[{'Key': 'kubernetes.io/cluster/' +
event['ResourceProperties']['InfrastructureName']}]);
            elif event['RequestType'] == 'Create':
              for subnet_id in event['ResourceProperties']['Subnets']:
                ec2_client.create_tags(Resources=[subnet_id], Tags=[{'Key': 'kubernetes.io/cluster/' +
event['ResourceProperties']['InfrastructureName'], 'Value': 'shared'}]);
            responseData = {}
            cfnresponse.send(event, context, cfnresponse.SUCCESS, responseData,
event['ResourceProperties']['InfrastructureName']+event['ResourceProperties']['Subnets'][0])
      Runtime: "python3.11"
      Timeout: 120

  RegisterPublicSubnetTags:
    Type: Custom::SubnetRegister
    Properties:
      ServiceToken: !GetAtt RegisterSubnetTags.Arn
      InfrastructureName: !Ref InfrastructureName
      Subnets: !Ref PublicSubnets

  RegisterPrivateSubnetTags:
    Type: Custom::SubnetRegister
    Properties:
      ServiceToken: !GetAtt RegisterSubnetTags.Arn
      InfrastructureName: !Ref InfrastructureName
      Subnets: !Ref PrivateSubnets

Outputs:
  PrivateHostedZoneId:
    Description: Hosted zone ID for the private DNS, which is required for private records.
    Value: !Ref IntDns
  ExternalApiLoadBalancerName:
    Description: Full name of the external API load balancer.
    Value: !GetAtt ExtApiElb.LoadBalancerFullName
  InternalApiLoadBalancerName:
    Description: Full name of the internal API load balancer.
    Value: !GetAtt IntApiElb.LoadBalancerFullName
  ApiServerDnsName:
    Description: Full hostname of the API server, which is required for the Ignition config files.
    Value: !Join [".", ["api-int", !Ref ClusterName, !Ref HostedZoneName]]
  RegisterNlbIpTargetsLambda:
    Description: Lambda ARN useful to help register or deregister IP targets for these load
balancers.
    Value: !GetAtt RegisterNlbIpTargets.Arn
  ExternalApiTargetGroupArn:
    Description: ARN of the external API target group.
    Value: !Ref ExternalApiTargetGroup
  InternalApiTargetGroupArn:
    Description: ARN of the internal API target group.
    Value: !Ref InternalApiTargetGroup
  InternalServiceTargetGroupArn:
    Description: ARN of the internal service target group.
    Value: !Ref InternalServiceTargetGroup
```

IMPORTANT

If you are deploying your cluster to an AWS government or secret region, you must update the **InternalApiServerRecord** to use **CNAME** records. Records of type **ALIAS** are not supported for AWS government regions. For example:

```
Type: CNAME
TTL: 10
ResourceRecords:
- !GetAtt IntApiElb.DNSName
```

**Additional resources**

- You can view details about the CloudFormation stacks that you create by navigating to the AWS CloudFormation console.
- You can view details about your hosted zones by navigating to the AWS Route 53 console .
- Listing public hosted zones(AWS documentation)

### 4.3.6. Creating security group and roles in AWS

You must create security groups and roles in Amazon Web Services (AWS) for your OpenShift Container Platform cluster to use.

You can use the provided CloudFormation template and a custom parameter file to create a stack of AWS resources. The stack represents the security groups and roles that your OpenShift Container Platform cluster requires.

NOTE

If you do not use the provided CloudFormation template to create your AWS infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

**Prerequisites**

- You configured an AWS account.
- You added your AWS keys and region to your local AWS profile by running **aws configure**.
- You generated the Ignition config files for your cluster.
- You created and configured a VPC and associated subnets in AWS.

**Procedure**

1. Create a JSON file that contains the parameter values that the template requires:

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
    "ParameterKey": "VpcCidr", (3)
    "ParameterValue": "10.0.0.0/16" (4)
  },
  {
    "ParameterKey": "PrivateSubnets", (5)
    "ParameterValue": "subnet-<random_string>" (6)
  },
  {
    "ParameterKey": "VpcId", (7)
    "ParameterValue": "vpc-<random_string>" (8)
  }
]
```

1. The name for your cluster infrastructure that is encoded in your Ignition config files for the cluster.
2. Specify the infrastructure name that you extracted from the Ignition config file metadata, which has the format **<cluster-name>-<random-string>**.
3. The CIDR block for the VPC.
4. Specify the CIDR block parameter that you used for the VPC that you defined in the form **x.x.x.x/16-24**.
5. The private subnets that you created for your VPC.
6. Specify the **PrivateSubnetIds** value from the output of the CloudFormation template for the VPC.
7. The VPC that you created for the cluster.
8. Specify the **VpcId** value from the output of the CloudFormation template for the VPC.

2. Copy the template from the **CloudFormation template for security objects**section of this topic and save it as a YAML file on your computer. This template describes the security groups and roles that your cluster requires.

3. Launch the CloudFormation template to create a stack of AWS resources that represent the security groups and roles:

IMPORTANT

You must enter the command on a single line.

```
$ aws cloudformation create-stack --stack-name <name> (1)
     --template-body file://<template>.yaml (2)
     --parameters file://<parameters>.json (3)
     --capabilities CAPABILITY_NAMED_IAM (4)
```

1. **<name>** is the name for the CloudFormation stack, such as **cluster-sec**. You need the name of this stack if you remove the cluster.

2 **<template>** is the relative path to and name of the CloudFormation template YAML file that you saved.

3 **<parameters>** is the relative path to and name of the CloudFormation parameters JSON file.

4 You must explicitly declare the **CAPABILITY_NAMED_IAM** capability because the provided template creates some **AWS::IAM::Role** and **AWS::IAM::InstanceProfile** resources.

**Example output**

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-sec/03bd4210-2ed7-11eb-
6d7a-13fc0b61e9db
```

4. Confirm that the template components exist:

```
$ aws cloudformation describe-stacks --stack-name <name>
```

After the **StackStatus** displays **CREATE_COMPLETE**, the output displays values for the following parameters. You must provide these parameter values to the other CloudFormation templates that you run to create your cluster:

| | |
|---|---|
| **MasterSecurityGroupId** | Master Security Group ID |
| **WorkerSecurityGroupId** | Worker Security Group ID |
| **MasterInstanceProfile** | Master IAM Instance Profile |
| **WorkerInstanceProfile** | Worker IAM Instance Profile |

### 4.3.6.1. CloudFormation template for security objects

You can use the following CloudFormation template to deploy the security objects that you need for your OpenShift Container Platform cluster.

**Example 4.21. CloudFormation template for security objects**

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Security Elements (Security Groups & IAM)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, unique cluster ID used to tag cloud resources and identify items owned or
used by the cluster.
    Type: String
  VpcCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-
4][0-9]|25[0-5])(\/(1[6-9]|2[0-4]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/16-24.
    Default: 10.0.0.0/16
    Description: CIDR block for VPC.
    Type: String
  VpcId:
    Description: The VPC-scoped resources will belong to this VPC.
    Type: AWS::EC2::VPC::Id
  PrivateSubnets:
    Description: The internal subnets.
    Type: List<AWS::EC2::Subnet::Id>

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - InfrastructureName
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcId
      - VpcCidr
      - PrivateSubnets
    ParameterLabels:
      InfrastructureName:
        default: "Infrastructure Name"
      VpcId:
        default: "VPC ID"
      VpcCidr:
        default: "VPC CIDR"
      PrivateSubnets:
        default: "Private Subnets"

Resources:
  MasterSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Cluster Master Security Group
      SecurityGroupIngress:
      - IpProtocol: icmp
        FromPort: 0
        ToPort: 0
        CidrIp: !Ref VpcCidr
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: !Ref VpcCidr
      - IpProtocol: tcp
        ToPort: 6443
        FromPort: 6443
        CidrIp: !Ref VpcCidr
      - IpProtocol: tcp
        FromPort: 22623
        ToPort: 22623
        CidrIp: !Ref VpcCidr
      VpcId: !Ref VpcId

  WorkerSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Cluster Worker Security Group
      SecurityGroupIngress:
      - IpProtocol: icmp
        FromPort: 0
        ToPort: 0
        CidrIp: !Ref VpcCidr
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: !Ref VpcCidr
      VpcId: !Ref VpcId

  MasterIngressEtcd:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: etcd
      FromPort: 2379
      ToPort: 2380
      IpProtocol: tcp

  MasterIngressVxlan:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Vxlan packets
      FromPort: 4789
      ToPort: 4789
      IpProtocol: udp

  MasterIngressWorkerVxlan:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Vxlan packets
      FromPort: 4789
      ToPort: 4789
      IpProtocol: udp

  MasterIngressGeneve:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Geneve packets
      FromPort: 6081
      ToPort: 6081
      IpProtocol: udp

  MasterIngressWorkerGeneve:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Geneve packets
      FromPort: 6081
      ToPort: 6081
      IpProtocol: udp

  MasterIngressIpsecIke:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: IPsec IKE packets
      FromPort: 500
      ToPort: 500
      IpProtocol: udp

  MasterIngressIpsecNat:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: IPsec NAT-T packets
      FromPort: 4500
      ToPort: 4500
      IpProtocol: udp

  MasterIngressIpsecEsp:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: IPsec ESP packets
      IpProtocol: 50

  MasterIngressWorkerIpsecIke:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: IPsec IKE packets
      FromPort: 500
      ToPort: 500
      IpProtocol: udp

  MasterIngressWorkerIpsecNat:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: IPsec NAT-T packets
      FromPort: 4500
      ToPort: 4500
      IpProtocol: udp

  MasterIngressWorkerIpsecEsp:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: IPsec ESP packets
      IpProtocol: 50

  MasterIngressInternal:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: tcp

  MasterIngressWorkerInternal:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: tcp

  MasterIngressInternalUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: udp

  MasterIngressWorkerInternalUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: udp

  MasterIngressKube:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Kubernetes kubelet, scheduler and controller manager
      FromPort: 10250
      ToPort: 10259
      IpProtocol: tcp

  MasterIngressWorkerKube:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Kubernetes kubelet, scheduler and controller manager
      FromPort: 10250
      ToPort: 10259
      IpProtocol: tcp

  MasterIngressIngressServices:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Kubernetes ingress services
      FromPort: 30000
      ToPort: 32767
      IpProtocol: tcp

  MasterIngressWorkerIngressServices:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Kubernetes ingress services
      FromPort: 30000
      ToPort: 32767
      IpProtocol: tcp

  MasterIngressIngressServicesUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Kubernetes ingress services
      FromPort: 30000
      ToPort: 32767
      IpProtocol: udp
```

```
MasterIngressWorkerIngressServicesUDP:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Kubernetes ingress services
    FromPort: 30000
    ToPort: 32767
    IpProtocol: udp

WorkerIngressVxlan:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Vxlan packets
    FromPort: 4789
    ToPort: 4789
    IpProtocol: udp

WorkerIngressMasterVxlan:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Vxlan packets
    FromPort: 4789
    ToPort: 4789
    IpProtocol: udp

WorkerIngressGeneve:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Geneve packets
    FromPort: 6081
    ToPort: 6081
    IpProtocol: udp

WorkerIngressMasterGeneve:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Geneve packets
    FromPort: 6081
    ToPort: 6081
    IpProtocol: udp

WorkerIngressIpsecIke:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
```

```
      Description: IPsec IKE packets
      FromPort: 500
      ToPort: 500
      IpProtocol: udp

  WorkerIngressIpsecNat:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: IPsec NAT-T packets
      FromPort: 4500
      ToPort: 4500
      IpProtocol: udp

  WorkerIngressIpsecEsp:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: IPsec ESP packets
      IpProtocol: 50

  WorkerIngressMasterIpsecIke:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: IPsec IKE packets
      FromPort: 500
      ToPort: 500
      IpProtocol: udp

  WorkerIngressMasterIpsecNat:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: IPsec NAT-T packets
      FromPort: 4500
      ToPort: 4500
      IpProtocol: udp

  WorkerIngressMasterIpsecEsp:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: IPsec ESP packets
      IpProtocol: 50

  WorkerIngressInternal:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: tcp

  WorkerIngressMasterInternal:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: tcp

  WorkerIngressInternalUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: udp

  WorkerIngressMasterInternalUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: udp

  WorkerIngressKube:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Kubernetes secure kubelet port
      FromPort: 10250
      ToPort: 10250
      IpProtocol: tcp

  WorkerIngressWorkerKube:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Internal Kubernetes communication
      FromPort: 10250
      ToPort: 10250
      IpProtocol: tcp

  WorkerIngressIngressServices:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Kubernetes ingress services
      FromPort: 30000
      ToPort: 32767
      IpProtocol: tcp

  WorkerIngressMasterIngressServices:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Kubernetes ingress services
      FromPort: 30000
      ToPort: 32767
      IpProtocol: tcp

  WorkerIngressIngressServicesUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Kubernetes ingress services
      FromPort: 30000
      ToPort: 32767
      IpProtocol: udp

  WorkerIngressMasterIngressServicesUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Kubernetes ingress services
      FromPort: 30000
      ToPort: 32767
      IpProtocol: udp

  MasterIamRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Principal:
            Service:
            - "ec2.amazonaws.com"
          Action:
          - "sts:AssumeRole"
      Policies:
      - PolicyName: !Join ["-", [!Ref InfrastructureName, "master", "policy"]]
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
```

```
        - Effect: "Allow"
          Action:
          - "ec2:AttachVolume"
          - "ec2:AuthorizeSecurityGroupIngress"
          - "ec2:CreateSecurityGroup"
          - "ec2:CreateTags"
          - "ec2:CreateVolume"
          - "ec2:DeleteSecurityGroup"
          - "ec2:DeleteVolume"
          - "ec2:Describe*"
          - "ec2:DetachVolume"
          - "ec2:ModifyInstanceAttribute"
          - "ec2:ModifyVolume"
          - "ec2:RevokeSecurityGroupIngress"
          - "elasticloadbalancing:AddTags"
          - "elasticloadbalancing:AttachLoadBalancerToSubnets"
          - "elasticloadbalancing:ApplySecurityGroupsToLoadBalancer"
          - "elasticloadbalancing:CreateListener"
          - "elasticloadbalancing:CreateLoadBalancer"
          - "elasticloadbalancing:CreateLoadBalancerPolicy"
          - "elasticloadbalancing:CreateLoadBalancerListeners"
          - "elasticloadbalancing:CreateTargetGroup"
          - "elasticloadbalancing:ConfigureHealthCheck"
          - "elasticloadbalancing:DeleteListener"
          - "elasticloadbalancing:DeleteLoadBalancer"
          - "elasticloadbalancing:DeleteLoadBalancerListeners"
          - "elasticloadbalancing:DeleteTargetGroup"
          - "elasticloadbalancing:DeregisterInstancesFromLoadBalancer"
          - "elasticloadbalancing:DeregisterTargets"
          - "elasticloadbalancing:Describe*"
          - "elasticloadbalancing:DetachLoadBalancerFromSubnets"
          - "elasticloadbalancing:ModifyListener"
          - "elasticloadbalancing:ModifyLoadBalancerAttributes"
          - "elasticloadbalancing:ModifyTargetGroup"
          - "elasticloadbalancing:ModifyTargetGroupAttributes"
          - "elasticloadbalancing:RegisterInstancesWithLoadBalancer"
          - "elasticloadbalancing:RegisterTargets"
          - "elasticloadbalancing:SetLoadBalancerPoliciesForBackendServer"
          - "elasticloadbalancing:SetLoadBalancerPoliciesOfListener"
          - "kms:DescribeKey"
          Resource: "*"

  MasterInstanceProfile:
    Type: "AWS::IAM::InstanceProfile"
    Properties:
      Roles:
      - Ref: "MasterIamRole"

  WorkerIamRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Principal:
            Service:
            - "ec2.amazonaws.com"
          Action:
          - "sts:AssumeRole"
      Policies:
      - PolicyName: !Join ["-", [!Ref InfrastructureName, "worker", "policy"]]
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
          - Effect: "Allow"
            Action:
            - "ec2:DescribeInstances"
            - "ec2:DescribeRegions"
            Resource: "*"

  WorkerInstanceProfile:
    Type: "AWS::IAM::InstanceProfile"
    Properties:
      Roles:
      - Ref: "WorkerIamRole"

Outputs:
  MasterSecurityGroupId:
    Description: Master Security Group ID
    Value: !GetAtt MasterSecurityGroup.GroupId

  WorkerSecurityGroupId:
    Description: Worker Security Group ID
    Value: !GetAtt WorkerSecurityGroup.GroupId

  MasterInstanceProfile:
    Description: Master IAM Instance Profile
    Value: !Ref MasterInstanceProfile

  WorkerInstanceProfile:
    Description: Worker IAM Instance Profile
    Value: !Ref WorkerInstanceProfile
```

**Additional resources**

- You can view details about the CloudFormation stacks that you create by navigating to the AWS CloudFormation console.

### 4.3.7. Accessing RHCOS AMIs with stream metadata

In OpenShift Container Platform, *stream metadata* provides standardized metadata about RHCOS in the JSON format and injects the metadata into the cluster. Stream metadata is a stable format that supports multiple architectures and is intended to be self-documenting for maintaining automation.

You can use the **coreos print-stream-json** sub-command of **openshift-install** to access information about the boot images in the stream metadata format. This command provides a method for printing stream metadata in a scriptable, machine-readable format.

For user-provisioned installations, the **openshift-install** binary contains references to the version of RHCOS boot images that are tested for use with OpenShift Container Platform, such as the AWS AMI.

### Procedure

To parse the stream metadata, use one of the following methods:

- From a Go program, use the official **stream-metadata-go** library at https://github.com/coreos/stream-metadata-go. You can also view example code in the library.
- From another programming language, such as Python or Ruby, use the JSON library of your preferred programming language.
- From a command-line utility that handles JSON data, such as **jq**:
  - Print the current **x86_64** or **aarch64** AMI for an AWS region, such as **us-west-1**:

    **For x86_64**

    ```
    $ openshift-install coreos print-stream-json | jq -r
    '.architectures.x86_64.images.aws.regions["us-west-1"].image'
    ```

    **Example output**

    ```
    ami-0d3e625f84626bbda
    ```

    **For aarch64**

    ```
    $ openshift-install coreos print-stream-json | jq -r
    '.architectures.aarch64.images.aws.regions["us-west-1"].image'
    ```

    **Example output**

    ```
    ami-0af1d3b7fa5be2131
    ```

    The output of this command is the AWS AMI ID for your designated architecture and the **us-west-1** region. The AMI must belong to the same region as the cluster.

## 4.3.8. RHCOS AMIs for the AWS infrastructure

Red Hat provides Red Hat Enterprise Linux CoreOS (RHCOS) AMIs that are valid for the various AWS regions and instance architectures that you can manually specify for your OpenShift Container Platform nodes.

**NOTE**

By importing your own AMI, you can also install to regions that do not have a published RHCOS AMI.

**Table 4.3. x86_64 RHCOS AMIs**

| AWS zone | AWS AMI |
|---|---|
| **af-south-1** | **ami-0163621ea085783d8** |
| **ap-east-1** | **ami-033db3b659641feea** |
| **ap-northeast-1** | **ami-0baf16f8c6bd53f63** |
| **ap-northeast-2** | **ami-01a92be7f419359cc** |
| **ap-northeast-3** | **ami-0f16895f6f50e656e** |
| **ap-south-1** | **ami-0272be2f6528576f3** |
| **ap-south-2** | **ami-0311119df2ebc0bbc** |
| **ap-southeast-1** | **ami-0637678b0ad540477** |
| **ap-southeast-2** | **ami-0b67b492c091ac746** |
| **ap-southeast-3** | **ami-0a9e63bf1df36a936** |
| **ap-southeast-4** | **ami-0f153b95673592039** |
| **ap-southeast-5** | **ami-025944207bb28ae8f** |
| **ap-southeast-7** | **ami-0b5e29c2ae4aaa66d** |
| **ca-central-1** | **ami-03263f0cfdfa8bbdb** |
| **ca-west-1** | **ami-0254620c2dc7dcacc** |
| **eu-central-1** | **ami-0a0a87862b24395d8** |
| **eu-central-2** | **ami-015c8ca32f5d8300a** |
| **eu-north-1** | **ami-0c4404a6ae5921a1b** |
| **eu-south-1** | **ami-0e0724943dd915bb2** |
| **eu-south-2** | **ami-0e6cac787a21b221d** |
| **eu-west-1** | **ami-0355d4c968e466965** |
| **eu-west-2** | **ami-0e079f8742280b034** |

| AWS zone | AWS AMI |
|---|---|
| **eu-west-3** | **ami-06702aad076acda7b** |
| **il-central-1** | **ami-0094ac2722d41c18c** |
| **me-central-1** | **ami-03680a3dcecfbe79d** |
| **me-south-1** | **ami-04e14a3c4be812ac7** |
| **mx-central-1** | **ami-0eac1c8d4154a417f** |
| **sa-east-1** | **ami-07abd63bb465f89b6** |
| **us-east-1** | **ami-0e8fd9094e487d1ff** |
| **us-east-2** | **ami-0d4a7b7677c0c883f** |
| **us-gov-east-1** | **ami-0b67e7ffd11a17645** |
| **us-gov-west-1** | **ami-041e18a76f42c752c** |
| **us-west-1** | **ami-0167f257577d883cc** |
| **us-west-2** | **ami-0b29d41f2ed6b8c94** |

Table 4.4. aarch64 RHCOS AMIs

| AWS zone | AWS AMI |
|---|---|
| **af-south-1** | **ami-009231bfc2490c6f9** |
| **ap-east-1** | **ami-0a23fad8fb25f5bb7** |
| **ap-northeast-1** | **ami-0754a269f165f227c** |
| **ap-northeast-2** | **ami-0d81f596571ce27d8** |
| **ap-northeast-3** | **ami-01eb2f8b176229523** |
| **ap-south-1** | **ami-0be3b34441044e437** |
| **ap-south-2** | **ami-02a86359661950bb0** |
| **ap-southeast-1** | **ami-0c70d35c9b5b190be** |

| AWS zone | AWS AMI |
|---|---|
| **ap-southeast-2** | **ami-0310f2acbeca636ed** |
| **ap-southeast-3** | **ami-04db4055063382442** |
| **ap-southeast-4** | **ami-0e2e40cc31633d7d6** |
| **ap-southeast-5** | **ami-0cf0c9ee9f324f763** |
| **ap-southeast-7** | **ami-04cdafcdc85bf9040** |
| **ca-central-1** | **ami-0aee20271a9396925** |
| **ca-west-1** | **ami-03ca778cd4265aad9** |
| **eu-central-1** | **ami-0281dddee0884d9f0** |
| **eu-central-2** | **ami-00fc4e5e3926530af** |
| **eu-north-1** | **ami-0696b5b31d326ccc6** |
| **eu-south-1** | **ami-04090792b7bdb9e0f** |
| **eu-south-2** | **ami-0d45a2586055d5daa** |
| **eu-west-1** | **ami-02f08479c3613ed0e** |
| **eu-west-2** | **ami-0ef2fc25f02a2d475** |
| **eu-west-3** | **ami-0ba5d0a0e5d796da8** |
| **il-central-1** | **ami-0e5b8f3b8e71961e7** |
| **me-central-1** | **ami-0d13d6a91da2ba547** |
| **me-south-1** | **ami-0183dab9f96845e3f** |
| **mx-central-1** | **ami-072535d81a5de8e76** |
| **sa-east-1** | **ami-0977fa46dff272ba9** |
| **us-east-1** | **ami-083de3282c55be3f7** |
| **us-east-2** | **ami-02f30107e3441227b** |

| AWS zone | AWS AMI |
|---|---|
| **us-gov-east-1** | **ami-0abaadf7322cfc258** |
| **us-gov-west-1** | **ami-0ca27128d77d732aa** |
| **us-west-1** | **ami-05a9426ae7c35740c** |
| **us-west-2** | **ami-0cd6ec50e0480b3a3** |

#### 4.3.8.1. AWS regions without a published RHCOS AMI

You can deploy an OpenShift Container Platform cluster to Amazon Web Services (AWS) regions without native support for a Red Hat Enterprise Linux CoreOS (RHCOS) Amazon Machine Image (AMI) or the AWS software development kit (SDK). If a published AMI is not available for an AWS region, you can upload a custom AMI prior to installing the cluster.

If you are deploying to a region not supported by the AWS SDK and you do not specify a custom AMI, the installation program copies the **us-east-1** AMI to the user account automatically. Then the installation program creates the control plane machines with encrypted EBS volumes using the default or user-specified Key Management Service (KMS) key. This allows the AMI to follow the same process workflow as published RHCOS AMIs.

A region without native support for an RHCOS AMI is not available to select from the terminal during cluster creation because it is not published. However, you can install to this region by configuring the custom AMI in the **install-config.yaml** file.

#### 4.3.8.2. Uploading a custom RHCOS AMI in AWS

If you are deploying to a custom Amazon Web Services (AWS) region, you must upload a custom Red Hat Enterprise Linux CoreOS (RHCOS) Amazon Machine Image (AMI) that belongs to that region.

##### Prerequisites

- You configured an AWS account.
- You created an Amazon S3 bucket with the required IAM service role.
- You uploaded your RHCOS VMDK file to Amazon S3. The RHCOS VMDK file must be the highest version that is less than or equal to the OpenShift Container Platform version you are installing.
- You downloaded the AWS CLI and installed it on your computer. See Install the AWS CLI Using the Bundled Installer.

##### Procedure

1. Export your AWS profile as an environment variable:

```
$ export AWS_PROFILE=<aws_profile> 1
```

2. Export the region to associate with your custom AMI as an environment variable:

```
$ export AWS_DEFAULT_REGION=<aws_region> 1
```

3. Export the version of RHCOS you uploaded to Amazon S3 as an environment variable:

```
$ export RHCOS_VERSION=<version> 1
```

1 1 1 The RHCOS VMDK version, like **4.19.0**.

4. Export the Amazon S3 bucket name as an environment variable:

```
$ export VMIMPORT_BUCKET_NAME=<s3_bucket_name>
```

5. Create the **containers.json** file and define your RHCOS VMDK file:

```
$ cat <<EOF > containers.json
{
  "Description": "rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64",
  "Format": "vmdk",
  "UserBucket": {
    "S3Bucket": "${VMIMPORT_BUCKET_NAME}",
    "S3Key": "rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64.vmdk"
  }
}
EOF
```

6. Import the RHCOS disk as an Amazon EBS snapshot:

```
$ aws ec2 import-snapshot --region ${AWS_DEFAULT_REGION} \
    --description "<description>" \ 1
    --disk-container "file://<file_path>/containers.json" 2
```

1 The description of your RHCOS disk being imported, like **rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64**.

2 The file path to the JSON file describing your RHCOS disk. The JSON file should contain your Amazon S3 bucket name and key.

7. Check the status of the image import:

```
$ watch -n 5 aws ec2 describe-import-snapshot-tasks --region ${AWS_DEFAULT_REGION}
```

**Example output**

```
{
    "ImportSnapshotTasks": [
        {
            "Description": "rhcos-4.7.0-x86_64-aws.x86_64",
            "ImportTaskId": "import-snap-fh6i8uil",
            "SnapshotTaskDetail": {
                "Description": "rhcos-4.7.0-x86_64-aws.x86_64",
                "DiskImageSize": 819056640.0,
                "Format": "VMDK",
                "SnapshotId": "snap-06331325870076318",
                "Status": "completed",
                "UserBucket": {
                    "S3Bucket": "external-images",
                    "S3Key": "rhcos-4.7.0-x86_64-aws.x86_64.vmdk"
                }
            }
        }
    ]
}
```

Copy the **SnapshotId** to register the image.

8. Create a custom RHCOS AMI from the RHCOS snapshot:

```
$ aws ec2 register-image \
   --region ${AWS_DEFAULT_REGION} \
   --architecture x86_64 \ 1
   --description "rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64" \ 2
   --ena-support \
   --name "rhcos-${RHCOS_VERSION}-x86_64-aws.x86_64" \ 3
   --virtualization-type hvm \
   --root-device-name '/dev/xvda' \
   --block-device-mappings 'DeviceName=/dev/xvda,Ebs=
{DeleteOnTermination=true,SnapshotId=<snapshot_ID>}' 4
```

1 The RHCOS VMDK architecture type, like **x86_64**, **aarch64**, **s390x**, or **ppc64le**.

2 The **Description** from the imported snapshot.

3 The name of the RHCOS AMI.

4 The **SnapshotID** from the imported snapshot.

To learn more about these APIs, see the AWS documentation for importing snapshots and creating EBS-backed AMIs.

### 4.3.9. Creating the bootstrap node in AWS

You must create the bootstrap node in Amazon Web Services (AWS) to use during OpenShift Container Platform cluster initialization. You do this by:

- Providing a location to serve the **bootstrap.ign** Ignition config file to your cluster. This file is located in your installation directory. The provided CloudFormation Template assumes that the Ignition config files for your cluster are served from an S3 bucket. If you choose to serve the files from another location, you must modify the templates.

- Using the provided CloudFormation template and a custom parameter file to create a stack of AWS resources. The stack represents the bootstrap node that your OpenShift Container Platform installation requires.

NOTE

If you do not use the provided CloudFormation template to create your bootstrap node, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

**Prerequisites**

- You configured an AWS account.
- You added your AWS keys and region to your local AWS profile by running **aws configure**.
- You generated the Ignition config files for your cluster.
- You created and configured a VPC and associated subnets in AWS.
- You created and configured DNS, load balancers, and listeners in AWS.
- You created the security groups and roles required for your cluster in AWS.

**Procedure**

1. Create the bucket by running the following command:

```
$ aws s3 mb s3://<cluster-name>-infra 1
```

1 **<cluster-name>-infra** is the bucket name. When creating the **install-config.yaml** file, replace **<cluster-name>** with the name specified for the cluster.

   You must use a presigned URL for your S3 bucket, instead of the **s3://** schema, if you are:

   - Deploying to a region that has endpoints that differ from the AWS SDK.
   - Deploying a proxy.
   - Providing your own custom endpoints.

2. Upload the **bootstrap.ign** Ignition config file to the bucket by running the following command:

```
$ aws s3 cp <installation_directory>/bootstrap.ign s3://<cluster-name>-infra/bootstrap.ign 1
```

1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

3. Verify that the file uploaded by running the following command:

```
$ aws s3 ls s3://<cluster-name>-infra/
```

**Example output**

```
2019-04-03 16:15:16     314878 bootstrap.ign
```

**NOTE**

The bootstrap Ignition config file does contain secrets, like X.509 keys. The following steps provide basic security for the S3 bucket. To provide additional security, you can enable an S3 bucket policy to allow only certain users, such as the OpenShift IAM user, to access objects that the bucket contains. You can avoid S3 entirely and serve your bootstrap Ignition config file from any address that the bootstrap machine can reach.

4. Create a JSON file that contains the parameter values that the template requires:

```
[
 {
  "ParameterKey": "InfrastructureName", 1
  "ParameterValue": "mycluster-<random_string>" 2
 },
 {
  "ParameterKey": "RhcosAmi", 3
  "ParameterValue": "ami-<random_string>" 4
 },
 {
  "ParameterKey": "AllowedBootstrapSshCidr", 5
  "ParameterValue": "0.0.0.0/0" 6
 },
 {
  "ParameterKey": "PublicSubnet", 7
  "ParameterValue": "subnet-<random_string>" 8
 },
 {
  "ParameterKey": "MasterSecurityGroupId", 9
  "ParameterValue": "sg-<random_string>" 10
 },
 {
  "ParameterKey": "VpcId", 11
  "ParameterValue": "vpc-<random_string>" 12
 },
 {
  "ParameterKey": "BootstrapIgnitionLocation", 13
  "ParameterValue": "s3://<bucket_name>/bootstrap.ign" 14
 },
 {
  "ParameterKey": "AutoRegisterELB", 15
  "ParameterValue": "yes" 16
 },
 {
  "ParameterKey": "RegisterNlbIpTargetsLambdaArn", 17
  "ParameterValue": "arn:aws:lambda:<aws_region>:<account_number>:function:
<dns_stack_name>-RegisterNlbIpTargets-<random_string>" 18
 },
 {
  "ParameterKey": "ExternalApiTargetGroupArn", 19
  "ParameterValue": "arn:aws:elasticloadbalancing:<aws_region>:
<account_number>:targetgroup/<dns_stack_name>-Exter-<random_string>" 20
```

```
  },
  {
    "ParameterKey": "InternalApiTargetGroupArn", 21
    "ParameterValue": "arn:aws:elasticloadbalancing:<aws_region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 22
  },
  {
    "ParameterKey": "InternalServiceTargetGroupArn", 23
    "ParameterValue": "arn:aws:elasticloadbalancing:<aws_region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 24
  }
]
```

1 The name for your cluster infrastructure that is encoded in your Ignition config files for the cluster.

2 Specify the infrastructure name that you extracted from the Ignition config file metadata, which has the format **<cluster-name>-<random-string>**.

3 Current Red Hat Enterprise Linux CoreOS (RHCOS) AMI to use for the bootstrap node based on your selected architecture.

4 Specify a valid **AWS::EC2::Image::Id** value.

5 CIDR block to allow SSH access to the bootstrap node.

6 Specify a CIDR block in the format **x.x.x.x/16-24**.

7 The public subnet that is associated with your VPC to launch the bootstrap node into.

8 Specify the **PublicSubnetIds** value from the output of the CloudFormation template for the VPC.

9 The master security group ID (for registering temporary rules)

10 Specify the **MasterSecurityGroupId** value from the output of the CloudFormation template for the security group and roles.

11 The VPC created resources will belong to.

12 Specify the **VpcId** value from the output of the CloudFormation template for the VPC.

13 Location to fetch bootstrap Ignition config file from.

14 Specify the S3 bucket and file name in the form **s3://<bucket_name>/bootstrap.ign**.

15 Whether or not to register a network load balancer (NLB).

16 Specify **yes** or **no**. If you specify **yes**, you must provide a Lambda Amazon Resource Name (ARN) value.

17 The ARN for NLB IP target registration lambda group.

18 Specify the **RegisterNlbIpTargetsLambda** value from the output of the CloudFormation template for DNS and load balancing. Use **arn:aws-us-gov** if deploying the cluster to an AWS GovCloud region.

19 The ARN for external API load balancer target group.

20 Specify the **ExternalApiTargetGroupArn** value from the output of the CloudFormation template for DNS and load balancing. Use **arn:aws-us-gov** if deploying the cluster to an AWS GovCloud region.

21 The ARN for internal API load balancer target group.

22 Specify the **InternalApiTargetGroupArn** value from the output of the CloudFormation template for DNS and load balancing. Use **arn:aws-us-gov** if deploying the cluster to an AWS GovCloud region.

23 The ARN for internal service load balancer target group.

24 Specify the **InternalServiceTargetGroupArn** value from the output of the CloudFormation template for DNS and load balancing. Use **arn:aws-us-gov** if deploying the cluster to an AWS GovCloud region.

5. Copy the template from the **CloudFormation template for the bootstrap machine**section of this topic and save it as a YAML file on your computer. This template describes the bootstrap machine that your cluster requires.

6. Optional: If you are deploying the cluster with a proxy, you must update the ignition in the template to add the **ignition.config.proxy** fields. Additionally, If you have added the Amazon EC2, Elastic Load Balancing, and S3 VPC endpoints to your VPC, you must add these endpoints to the **noProxy** field.

7. Launch the CloudFormation template to create a stack of AWS resources that represent the bootstrap node:

IMPORTANT

You must enter the command on a single line.

```
$ aws cloudformation create-stack --stack-name <name> 1
     --template-body file://<template>.yaml 2
     --parameters file://<parameters>.json 3
     --capabilities CAPABILITY_NAMED_IAM 4
```

1 **<name>** is the name for the CloudFormation stack, such as **cluster-bootstrap**. You need the name of this stack if you remove the cluster.

2 **<template>** is the relative path to and name of the CloudFormation template YAML file that you saved.

3 **<parameters>** is the relative path to and name of the CloudFormation parameters JSON file.

4 You must explicitly declare the **CAPABILITY_NAMED_IAM** capability because the provided template creates some **AWS::IAM::Role** and **AWS::IAM::InstanceProfile** resources.

**Example output**

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-bootstrap/12944486-2add-
11eb-9dee-12dace8e3a83
```

8. Confirm that the template components exist:

```
$ aws cloudformation describe-stacks --stack-name <name>
```

After the **StackStatus** displays **CREATE_COMPLETE**, the output displays values for the following parameters. You must provide these parameter values to the other CloudFormation templates that you run to create your cluster:

| | |
|---|---|
| **Bootstrap InstanceId** | The bootstrap Instance ID. |
| **Bootstrap PublicIp** | The bootstrap node public IP address. |
| **Bootstrap PrivateIp** | The bootstrap node private IP address. |

### 4.3.9.1. CloudFormation template for the bootstrap machine

You can use the following CloudFormation template to deploy the bootstrap machine that you need for your OpenShift Container Platform cluster.

**Example 4.22. CloudFormation template for the bootstrap machine**

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Bootstrap (EC2 Instance, Security Groups and IAM)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, unique cluster ID used to tag cloud resources and identify items owned or
used by the cluster.
    Type: String
  RhcosAmi:
    Description: Current Red Hat Enterprise Linux CoreOS AMI to use for bootstrap.
    Type: AWS::EC2::Image::Id
  AllowedBootstrapSshCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-
4][0-9]|25[0-5])(\/([0-9]|1[0-9]|2[0-9]|3[0-2]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/0-32.
    Default: 0.0.0.0/0
    Description: CIDR block to allow SSH access to the bootstrap node.
    Type: String
  PublicSubnet:
    Description: The public subnet to launch the bootstrap node into.
    Type: AWS::EC2::Subnet::Id
  MasterSecurityGroupId:
    Description: The master security group ID for registering temporary rules.
    Type: AWS::EC2::SecurityGroup::Id
  VpcId:
    Description: The VPC-scoped resources will belong to this VPC.
    Type: AWS::EC2::VPC::Id
  BootstrapIgnitionLocation:
    Default: s3://my-s3-bucket/bootstrap.ign
    Description: Ignition config file location.
    Type: String
  AutoRegisterELB:
    Default: "yes"
    AllowedValues:
    - "yes"
    - "no"
    Description: Do you want to invoke NLB registration, which requires a Lambda ARN parameter?
    Type: String
  RegisterNlbIpTargetsLambdaArn:
    Description: ARN for NLB IP target registration lambda.
    Type: String
  ExternalApiTargetGroupArn:
    Description: ARN for external API load balancer target group.
    Type: String
  InternalApiTargetGroupArn:
    Description: ARN for internal API load balancer target group.
    Type: String
  InternalServiceTargetGroupArn:
    Description: ARN for internal service load balancer target group.
    Type: String
  BootstrapInstanceType:
    Description: Instance type for the bootstrap EC2 instance
    Default: "i3.large"
    Type: String

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - InfrastructureName
    - Label:
        default: "Host Information"
      Parameters:
      - RhcosAmi
      - BootstrapIgnitionLocation
      - MasterSecurityGroupId
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcId
      - AllowedBootstrapSshCidr
      - PublicSubnet
      - Label:
          default: "Load Balancer Automation"
        Parameters:
        - AutoRegisterELB
        - RegisterNlbIpTargetsLambdaArn
        - ExternalApiTargetGroupArn
        - InternalApiTargetGroupArn
        - InternalServiceTargetGroupArn
      ParameterLabels:
        InfrastructureName:
          default: "Infrastructure Name"
        VpcId:
          default: "VPC ID"
        AllowedBootstrapSshCidr:
          default: "Allowed SSH Source"
        PublicSubnet:
          default: "Public Subnet"
        RhcosAmi:
          default: "Red Hat Enterprise Linux CoreOS AMI ID"
        BootstrapIgnitionLocation:
          default: "Bootstrap Ignition Source"
        MasterSecurityGroupId:
          default: "Master Security Group ID"
        AutoRegisterELB:
          default: "Use Provided ELB Automation"

  Conditions:
    DoRegistration: !Equals ["yes", !Ref AutoRegisterELB]

  Resources:
    BootstrapIamRole:
      Type: AWS::IAM::Role
      Properties:
        AssumeRolePolicyDocument:
          Version: "2012-10-17"
          Statement:
          - Effect: "Allow"
            Principal:
              Service:
              - "ec2.amazonaws.com"
            Action:
            - "sts:AssumeRole"
        Path: "/"
        Policies:
        - PolicyName: !Join ["-", [!Ref InfrastructureName, "bootstrap", "policy"]]
          PolicyDocument:
            Version: "2012-10-17"
            Statement:
            - Effect: "Allow"
              Action: "ec2:Describe*"
              Resource: "*"
            - Effect: "Allow"
              Action: "ec2:AttachVolume"
              Resource: "*"
            - Effect: "Allow"
              Action: "ec2:DetachVolume"
        Resource: "*"
      - Effect: "Allow"
        Action: "s3:GetObject"
        Resource: "*"

BootstrapInstanceProfile:
  Type: "AWS::IAM::InstanceProfile"
  Properties:
    Path: "/"
    Roles:
    - Ref: "BootstrapIamRole"

BootstrapSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Cluster Bootstrap Security Group
    SecurityGroupIngress:
    - IpProtocol: tcp
      FromPort: 22
      ToPort: 22
      CidrIp: !Ref AllowedBootstrapSshCidr
    - IpProtocol: tcp
      ToPort: 19531
      FromPort: 19531
      CidrIp: 0.0.0.0/0
    VpcId: !Ref VpcId

BootstrapInstance:
  Type: AWS::EC2::Instance
  Properties:
    ImageId: !Ref RhcosAmi
    IamInstanceProfile: !Ref BootstrapInstanceProfile
    InstanceType: !Ref BootstrapInstanceType
    NetworkInterfaces:
    - AssociatePublicIpAddress: "true"
      DeviceIndex: "0"
      GroupSet:
      - !Ref "BootstrapSecurityGroup"
      - !Ref "MasterSecurityGroupId"
      SubnetId: !Ref "PublicSubnet"
    UserData:
      Fn::Base64: !Sub
      - '{"ignition":{"config":{"replace":{"source":"${S3Loc}"}},"version":"3.1.0"}}'
      - {
        S3Loc: !Ref BootstrapIgnitionLocation
      }

RegisterBootstrapApiTarget:
  Condition: DoRegistration
  Type: Custom::NLBRegister
  Properties:
    ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
    TargetArn: !Ref ExternalApiTargetGroupArn
    TargetIp: !GetAtt BootstrapInstance.PrivateIp

RegisterBootstrapInternalApiTarget:
      Condition: DoRegistration
      Type: Custom::NLBRegister
      Properties:
        ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
        TargetArn: !Ref InternalApiTargetGroupArn
        TargetIp: !GetAtt BootstrapInstance.PrivateIp

    RegisterBootstrapInternalServiceTarget:
      Condition: DoRegistration
      Type: Custom::NLBRegister
      Properties:
        ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
        TargetArn: !Ref InternalServiceTargetGroupArn
        TargetIp: !GetAtt BootstrapInstance.PrivateIp

  Outputs:
    BootstrapInstanceId:
      Description: Bootstrap Instance ID.
      Value: !Ref BootstrapInstance

    BootstrapPublicIp:
      Description: The bootstrap node public IP address.
      Value: !GetAtt BootstrapInstance.PublicIp

    BootstrapPrivateIp:
      Description: The bootstrap node private IP address.
      Value: !GetAtt BootstrapInstance.PrivateIp
```

**Additional resources**

- You can view details about the CloudFormation stacks that you create by navigating to the AWS CloudFormation console.
- RHCOS AMIs for the AWS infrastructure

### 4.3.10. Creating the control plane machines in AWS

You must create the control plane machines in Amazon Web Services (AWS) that your cluster will use.

You can use the provided CloudFormation template and a custom parameter file to create a stack of AWS resources that represent the control plane nodes.

**IMPORTANT**

The CloudFormation template creates a stack that represents three control plane nodes.

**NOTE**

If you do not use the provided CloudFormation template to create your control plane nodes, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

### Prerequisites

- You configured an AWS account.
- You added your AWS keys and region to your local AWS profile by running **aws configure**.
- You generated the Ignition config files for your cluster.
- You created and configured a VPC and associated subnets in AWS.
- You created and configured DNS, load balancers, and listeners in AWS.
- You created the security groups and roles required for your cluster in AWS.
- You created the bootstrap machine.

### Procedure

1. Create a JSON file that contains the parameter values that the template requires:

```
[
  {
    "ParameterKey": "InfrastructureName", (1)
    "ParameterValue": "mycluster-<random_string>" (2)
  },
  {
    "ParameterKey": "RhcosAmi", (3)
    "ParameterValue": "ami-<random_string>" (4)
  },
  {
    "ParameterKey": "AutoRegisterDNS", (5)
    "ParameterValue": "yes" (6)
  },
  {
    "ParameterKey": "PrivateHostedZoneId", (7)
    "ParameterValue": "<random_string>" (8)
  },
  {
    "ParameterKey": "PrivateHostedZoneName", (9)
    "ParameterValue": "mycluster.example.com" (10)
  },
  {
    "ParameterKey": "Master0Subnet", (11)
    "ParameterValue": "subnet-<random_string>" (12)
  },
  {
    "ParameterKey": "Master1Subnet", (13)
    "ParameterValue": "subnet-<random_string>" (14)
  },
  {
    "ParameterKey": "Master2Subnet", (15)
    "ParameterValue": "subnet-<random_string>" (16)
  },
  {
    "ParameterKey": "MasterSecurityGroupId", (17)
    "ParameterValue": "sg-<random_string>" (18)
  },
  {
    "ParameterKey": "IgnitionLocation", (19)
    "ParameterValue": "https://api-int.<cluster_name>.<domain_name>:22623/config/master"
(20)
  },
  {
    "ParameterKey": "CertificateAuthorities", (21)
    "ParameterValue": "data:text/plain;charset=utf-8;base64,ABC...xYz==" (22)
  },
  {
    "ParameterKey": "MasterInstanceProfileName", (23)
    "ParameterValue": "<roles_stack>-MasterInstanceProfile-<random_string>" (24)
  },
  {
    "ParameterKey": "MasterInstanceType", (25)
    "ParameterValue": "" (26)
  },
  {
    "ParameterKey": "AutoRegisterELB", (27)
    "ParameterValue": "yes" (28)
  },
  {
    "ParameterKey": "RegisterNlbIpTargetsLambdaArn", (29)
    "ParameterValue": "arn:aws:lambda:<aws_region>:<account_number>:function:
<dns_stack_name>-RegisterNlbIpTargets-<random_string>" (30)
  },
  {
    "ParameterKey": "ExternalApiTargetGroupArn", (31)
    "ParameterValue": "arn:aws:elasticloadbalancing:<aws_region>:
<account_number>:targetgroup/<dns_stack_name>-Exter-<random_string>" (32)
  },
  {
    "ParameterKey": "InternalApiTargetGroupArn", (33)
    "ParameterValue": "arn:aws:elasticloadbalancing:<aws_region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" (34)
  },
  {
    "ParameterKey": "InternalServiceTargetGroupArn", (35)
    "ParameterValue": "arn:aws:elasticloadbalancing:<aws_region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" (36)
  }
]
```

1. The name for your cluster infrastructure that is encoded in your Ignition config files for the cluster.

2. Specify the infrastructure name that you extracted from the Ignition config file metadata, which has the format **<cluster-name>-<random-string>**.

3. Current Red Hat Enterprise Linux CoreOS (RHCOS) AMI to use for the control plane machines based on your selected architecture.

machines based on your selected architecture.

4 Specify an **AWS::EC2::Image::Id** value.

5 Whether or not to perform DNS etcd registration.

6 Specify **yes** or **no**. If you specify **yes**, you must provide hosted zone information.

7 The Route 53 private zone ID to register the etcd targets with.

8 Specify the **PrivateHostedZoneId** value from the output of the CloudFormation template for DNS and load balancing.

9 The Route 53 zone to register the targets with.

10 Specify **<cluster_name>.<domain_name>** where **<domain_name>** is the Route 53 base domain that you used when you generated **install-config.yaml** file for the cluster. Do not include the trailing period (.) that is displayed in the AWS console.

11 13 15 A subnet, preferably private, to launch the control plane machines on.

12 14 16 Specify a subnet from the **PrivateSubnets** value from the output of the CloudFormation template for DNS and load balancing.

17 The master security group ID to associate with control plane nodes.

18 Specify the **MasterSecurityGroupId** value from the output of the CloudFormation template for the security group and roles.

19 The location to fetch control plane Ignition config file from.

20 Specify the generated Ignition config file location, **https://api-int.<cluster_name>.<domain_name>:22623/config/master**.

21 The base64 encoded certificate authority string to use.

22 Specify the value from the **master.ign** file that is in the installation directory. This value is the long string with the format **data:text/plain;charset=utf-8;base64,ABC…xYz==**.

23 The IAM profile to associate with control plane nodes.

24 Specify the **MasterInstanceProfile** parameter value from the output of the CloudFormation template for the security group and roles.

25 The type of AWS instance to use for the control plane machines based on your selected architecture.

26 The instance type value corresponds to the minimum resource requirements for control plane machines. For example **m6i.xlarge** is a type for AMD64 and **m6g.xlarge** is a type for ARM64.

27 Whether or not to register a network load balancer (NLB).

28 Specify **yes** or **no**. If you specify **yes**, you must provide a Lambda Amazon Resource Name (ARN) value.

29 The ARN for NLB IP target registration lambda group.

30 Specify the **RegisterNlbIpTargetsLambda** value from the output of the CloudFormation template for DNS and load balancing. Use **arn:aws-us-gov** if deploying the cluster to an

31 The ARN for external API load balancer target group.

32 Specify the **ExternalApiTargetGroupArn** value from the output of the CloudFormation template for DNS and load balancing. Use **arn:aws-us-gov** if deploying the cluster to an AWS GovCloud region.

33 The ARN for internal API load balancer target group.

34 Specify the **InternalApiTargetGroupArn** value from the output of the CloudFormation template for DNS and load balancing. Use **arn:aws-us-gov** if deploying the cluster to an AWS GovCloud region.

35 The ARN for internal service load balancer target group.

36 Specify the **InternalServiceTargetGroupArn** value from the output of the CloudFormation template for DNS and load balancing. Use **arn:aws-us-gov** if deploying the cluster to an AWS GovCloud region.

2. Copy the template from the **CloudFormation template for control plane machines**section of this topic and save it as a YAML file on your computer. This template describes the control plane machines that your cluster requires.

3. If you specified an **m5** instance type as the value for **MasterInstanceType**, add that instance type to the **MasterInstanceType.AllowedValues** parameter in the CloudFormation template.

4. Launch the CloudFormation template to create a stack of AWS resources that represent the control plane nodes:

IMPORTANT

You must enter the command on a single line.

```
$ aws cloudformation create-stack --stack-name <name> 1
    --template-body file://<template>.yaml 2
    --parameters file://<parameters>.json 3
```

1 **<name>** is the name for the CloudFormation stack, such as **cluster-control-plane**. You need the name of this stack if you remove the cluster.

2 **<template>** is the relative path to and name of the CloudFormation template YAML file that you saved.

3 **<parameters>** is the relative path to and name of the CloudFormation parameters JSON file.

**Example output**

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-control-plane/21c7e2b0-2ee2-
11eb-c6f6-0aa34627df4b
```

NOTE

The CloudFormation template creates a stack that represents three control plane nodes.

5. Confirm that the template components exist:

```
$ aws cloudformation describe-stacks --stack-name <name>
```

#### 4.3.10.1. CloudFormation template for control plane machines

You can use the following CloudFormation template to deploy the control plane machines that you need for your OpenShift Container Platform cluster.

**Example 4.23. CloudFormation template for control plane machines**

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Node Launch (EC2 master instances)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, unique cluster ID used to tag nodes for the kubelet cloud provider.
    Type: String
  RhcosAmi:
    Description: Current Red Hat Enterprise Linux CoreOS AMI to use for bootstrap.
    Type: AWS::EC2::Image::Id
  AutoRegisterDNS:
    Default: ""
    Description: unused
    Type: String
  PrivateHostedZoneId:
    Default: ""
    Description: unused
    Type: String
  PrivateHostedZoneName:
    Default: ""
    Description: unused
    Type: String
  Master0Subnet:
    Description: The subnets, recommend private, to launch the master nodes into.
    Type: AWS::EC2::Subnet::Id
  Master1Subnet:
    Description: The subnets, recommend private, to launch the master nodes into.
    Type: AWS::EC2::Subnet::Id
  Master2Subnet:
    Description: The subnets, recommend private, to launch the master nodes into.
    Type: AWS::EC2::Subnet::Id
  MasterSecurityGroupId:
    Description: The master security group ID to associate with master nodes.
    Type: AWS::EC2::SecurityGroup::Id
  IgnitionLocation:
    Default: https://api-int.$CLUSTER_NAME.$DOMAIN:22623/config/master
    Description: Ignition config file location.
    Type: String
  CertificateAuthorities:
    Default: data:text/plain;charset=utf-8;base64,ABC...xYz==
    Description: Base64 encoded certificate authority string to use.
    Type: String
  MasterInstanceProfileName:
    Description: IAM profile to associate with master nodes.
    Type: String
  MasterInstanceType:
    Default: m5.xlarge
    Type: String

  AutoRegisterELB:
    Default: "yes"
    AllowedValues:
    - "yes"
    - "no"
    Description: Do you want to invoke NLB registration, which requires a Lambda ARN parameter?
    Type: String
  RegisterNlbIpTargetsLambdaArn:
    Description: ARN for NLB IP target registration lambda. Supply the value from the cluster
infrastructure or select "no" for AutoRegisterELB.
    Type: String
  ExternalApiTargetGroupArn:
    Description: ARN for external API load balancer target group. Supply the value from the cluster
infrastructure or select "no" for AutoRegisterELB.
    Type: String
  InternalApiTargetGroupArn:
    Description: ARN for internal API load balancer target group. Supply the value from the cluster
infrastructure or select "no" for AutoRegisterELB.
    Type: String
  InternalServiceTargetGroupArn:
    Description: ARN for internal service load balancer target group. Supply the value from the
cluster infrastructure or select "no" for AutoRegisterELB.
    Type: String

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - InfrastructureName
    - Label:
        default: "Host Information"
      Parameters:
      - MasterInstanceType
      - RhcosAmi
      - IgnitionLocation
      - CertificateAuthorities
      - MasterSecurityGroupId
      - MasterInstanceProfileName
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcId
      - AllowedBootstrapSshCidr
      - Master0Subnet
      - Master1Subnet
      - Master2Subnet
    - Label:
        default: "Load Balancer Automation"
      Parameters:
      - AutoRegisterELB
      - RegisterNlbIpTargetsLambdaArn
      - ExternalApiTargetGroupArn
      - InternalApiTargetGroupArn
      - InternalServiceTargetGroupArn
    ParameterLabels:
      InfrastructureName:
        default: "Infrastructure Name"
      VpcId:
        default: "VPC ID"
      Master0Subnet:
        default: "Master-0 Subnet"
      Master1Subnet:
        default: "Master-1 Subnet"
      Master2Subnet:
        default: "Master-2 Subnet"
      MasterInstanceType:
        default: "Master Instance Type"
      MasterInstanceProfileName:
        default: "Master Instance Profile Name"
      RhcosAmi:
        default: "Red Hat Enterprise Linux CoreOS AMI ID"
      BootstrapIgnitionLocation:
        default: "Master Ignition Source"
      CertificateAuthorities:
        default: "Ignition CA String"
      MasterSecurityGroupId:
        default: "Master Security Group ID"
      AutoRegisterELB:
        default: "Use Provided ELB Automation"

Conditions:
  DoRegistration: !Equals ["yes", !Ref AutoRegisterELB]

Resources:
  Master0:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref RhcosAmi
      BlockDeviceMappings:
      - DeviceName: /dev/xvda
        Ebs:
          VolumeSize: "120"
          VolumeType: "gp2"
      IamInstanceProfile: !Ref MasterInstanceProfileName
      InstanceType: !Ref MasterInstanceType
      NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "MasterSecurityGroupId"
        SubnetId: !Ref "Master0Subnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"merge":[{"source":"${SOURCE}"}]},"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}"}]}},"version":"3.1.0"}}'
        - {
          SOURCE: !Ref IgnitionLocation,
          CA_BUNDLE: !Ref CertificateAuthorities,
        }
      Tags:
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "shared"

  RegisterMaster0:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref ExternalApiTargetGroupArn
      TargetIp: !GetAtt Master0.PrivateIp

  RegisterMaster0InternalApiTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalApiTargetGroupArn
      TargetIp: !GetAtt Master0.PrivateIp

  RegisterMaster0InternalServiceTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalServiceTargetGroupArn
      TargetIp: !GetAtt Master0.PrivateIp

  Master1:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref RhcosAmi
      BlockDeviceMappings:
      - DeviceName: /dev/xvda
        Ebs:
          VolumeSize: "120"
          VolumeType: "gp2"
      IamInstanceProfile: !Ref MasterInstanceProfileName
      InstanceType: !Ref MasterInstanceType
      NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "MasterSecurityGroupId"
        SubnetId: !Ref "Master1Subnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"merge":[{"source":"${SOURCE}"}]},"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}"}]}},"version":"3.1.0"}}'
        - {
          SOURCE: !Ref IgnitionLocation,
          CA_BUNDLE: !Ref CertificateAuthorities,
        }
      Tags:
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "shared"

  RegisterMaster1:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref ExternalApiTargetGroupArn
      TargetIp: !GetAtt Master1.PrivateIp

  RegisterMaster1InternalApiTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalApiTargetGroupArn
      TargetIp: !GetAtt Master1.PrivateIp

  RegisterMaster1InternalServiceTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalServiceTargetGroupArn
      TargetIp: !GetAtt Master1.PrivateIp

  Master2:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref RhcosAmi
      BlockDeviceMappings:
      - DeviceName: /dev/xvda
        Ebs:
          VolumeSize: "120"
          VolumeType: "gp2"
      IamInstanceProfile: !Ref MasterInstanceProfileName
      InstanceType: !Ref MasterInstanceType
      NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "MasterSecurityGroupId"
        SubnetId: !Ref "Master2Subnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"merge":[{"source":"${SOURCE}"}]},"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}"}]}},"version":"3.1.0"}}'
        - {
          SOURCE: !Ref IgnitionLocation,
          CA_BUNDLE: !Ref CertificateAuthorities,
        }
      Tags:
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "shared"

  RegisterMaster2:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref ExternalApiTargetGroupArn
      TargetIp: !GetAtt Master2.PrivateIp

  RegisterMaster2InternalApiTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalApiTargetGroupArn
      TargetIp: !GetAtt Master2.PrivateIp

  RegisterMaster2InternalServiceTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalServiceTargetGroupArn
      TargetIp: !GetAtt Master2.PrivateIp

Outputs:
  PrivateIPs:
    Description: The control-plane node private IP addresses.
    Value:
      !Join [
        ",",
        [!GetAtt Master0.PrivateIp, !GetAtt Master1.PrivateIp, !GetAtt Master2.PrivateIp]
      ]
```

**Additional resources**

- You can view details about the CloudFormation stacks that you create by navigating to the AWS CloudFormation console.

### 4.3.11. Creating the worker nodes in AWS

You can create worker nodes in Amazon Web Services (AWS) for your cluster to use.

NOTE

If you are installing a three-node cluster, skip this step. A three-node cluster consists of three control plane machines, which also act as compute machines.

You can use the provided CloudFormation template and a custom parameter file to create a stack of AWS resources that represent a worker node.

IMPORTANT

The CloudFormation template creates a stack that represents one worker node. You must create a stack for each worker node.

NOTE

If you do not use the provided CloudFormation template to create your worker nodes, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

### Prerequisites

- You configured an AWS account.
- You added your AWS keys and region to your local AWS profile by running **aws configure**.
- You generated the Ignition config files for your cluster.
- You created and configured a VPC and associated subnets in AWS.
- You created and configured DNS, load balancers, and listeners in AWS.
- You created the security groups and roles required for your cluster in AWS.
- You created the bootstrap machine.
- You created the control plane machines.

### Procedure

1. Create a JSON file that contains the parameter values that the CloudFormation template requires:

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
    "ParameterKey": "RhcosAmi", 3
    "ParameterValue": "ami-<random_string>" 4
  },
  {
    "ParameterKey": "Subnet", 5
    "ParameterValue": "subnet-<random_string>" (6)
  },
  {
    "ParameterKey": "WorkerSecurityGroupId", (7)
    "ParameterValue": "sg-<random_string>" (8)
  },
  {
    "ParameterKey": "IgnitionLocation", (9)
    "ParameterValue": "https://api-int.<cluster_name>.<domain_name>:22623/config/worker"
(10)
  },
  {
    "ParameterKey": "CertificateAuthorities", (11)
    "ParameterValue": "data:text/plain;charset=utf-8;base64,ABC...xYz==" (12)
  },
  {
    "ParameterKey": "WorkerInstanceProfileName", (13)
    "ParameterValue": "<roles_stack>-WorkerInstanceProfile-<random_string>" (14)
  },
  {
    "ParameterKey": "WorkerInstanceType", (15)
    "ParameterValue": "" (16)
  }
]
```

1. The name for your cluster infrastructure that is encoded in your Ignition config files for the cluster.

2. Specify the infrastructure name that you extracted from the Ignition config file metadata, which has the format **<cluster-name>-<random-string>**.

3. Current Red Hat Enterprise Linux CoreOS (RHCOS) AMI to use for the worker nodes based on your selected architecture.

4. Specify an **AWS::EC2::Image::Id** value.

5. A subnet, preferably private, to start the worker nodes on.

6. Specify a subnet from the **PrivateSubnets** value from the output of the CloudFormation template for DNS and load balancing.

7. The worker security group ID to associate with worker nodes.

8. Specify the **WorkerSecurityGroupId** value from the output of the CloudFormation template for the security group and roles.

9. The location to fetch the bootstrap Ignition config file from.

10. Specify the generated Ignition config location, **https://api-int.<cluster_name>.<domain_name>:22623/config/worker**.

11. Base64 encoded certificate authority string to use.

12. Specify the value from the **worker.ign** file that is in the installation directory. This value is the long string with the format **data:text/plain;charset=utf-8;base64,ABC…xYz==**.

13 The IAM profile to associate with worker nodes.

14 Specify the **WorkerInstanceProfile** parameter value from the output of the CloudFormation template for the security group and roles.

15 The type of AWS instance to use for the compute machines based on your selected architecture.

16 The instance type value corresponds to the minimum resource requirements for compute machines. For example **m6i.large** is a type for AMD64 and **m6g.large** is a type for ARM64.

2. Copy the template from the **CloudFormation template for worker machines** section of this topic and save it as a YAML file on your computer. This template describes the networking objects and load balancers that your cluster requires.

3. Optional: If you specified an **m5** instance type as the value for **WorkerInstanceType**, add that instance type to the **WorkerInstanceType.AllowedValues** parameter in the CloudFormation template.

4. Optional: If you are deploying with an AWS Marketplace image, update the **Worker0.type.properties.ImageID** parameter with the AMI ID that you obtained from your subscription.

5. Use the CloudFormation template to create a stack of AWS resources that represent a worker node:

IMPORTANT

You must enter the command on a single line.

```
$ aws cloudformation create-stack --stack-name <name> 1
    --template-body file://<template>.yaml \ 2
    --parameters file://<parameters>.json 3
```

1 **<name>** is the name for the CloudFormation stack, such as **cluster-worker-1**. You need the name of this stack if you remove the cluster.

2 **<template>** is the relative path to and name of the CloudFormation template YAML file that you saved.

3 **<parameters>** is the relative path to and name of the CloudFormation parameters JSON file.

### Example output

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-worker-1/729ee301-1c2a-
11eb-348f-sd9888c65b59
```

NOTE

The CloudFormation template creates a stack that represents one worker node.

6. Confirm that the template components exist:

```
$ aws cloudformation describe-stacks --stack-name <name>
```

7. Continue to create worker stacks until you have created enough worker machines for your cluster. You can create additional worker stacks by referencing the same template and parameter files and specifying a different stack name.

**IMPORTANT**

You must create at least two worker machines, so you must create at least two stacks that use this CloudFormation template.

### 4.3.11.1. CloudFormation template for compute machines

You can deploy the compute machines that you need for your OpenShift Container Platform cluster by using the following CloudFormation template.

**Example 4.24. CloudFormation template for compute machines**

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Node Launch (EC2 worker instance)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, unique cluster ID used to tag nodes for the kubelet cloud provider.
    Type: String
  RhcosAmi:
    Description: Current Red Hat Enterprise Linux CoreOS AMI to use for bootstrap.
    Type: AWS::EC2::Image::Id
  Subnet:
    Description: The subnets, recommend private, to launch the worker nodes into.
    Type: AWS::EC2::Subnet::Id
  WorkerSecurityGroupId:
    Description: The worker security group ID to associate with worker nodes.
    Type: AWS::EC2::SecurityGroup::Id
  IgnitionLocation:
    Default: https://api-int.$CLUSTER_NAME.$DOMAIN:22623/config/worker
    Description: Ignition config file location.
    Type: String
  CertificateAuthorities:
    Default: data:text/plain;charset=utf-8;base64,ABC...xYz==
    Description: Base64 encoded certificate authority string to use.
    Type: String
  WorkerInstanceProfileName:
    Description: IAM profile to associate with worker nodes.
    Type: String
  WorkerInstanceType:
    Default: m5.large
    Type: String
```

```
Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - InfrastructureName
    - Label:
        default: "Host Information"
      Parameters:
      - WorkerInstanceType
      - RhcosAmi
      - IgnitionLocation
      - CertificateAuthorities
      - WorkerSecurityGroupId
      - WorkerInstanceProfileName
    - Label:
        default: "Network Configuration"
      Parameters:
      - Subnet
    ParameterLabels:
      Subnet:
        default: "Subnet"
      InfrastructureName:
        default: "Infrastructure Name"
      WorkerInstanceType:
        default: "Worker Instance Type"
      WorkerInstanceProfileName:
        default: "Worker Instance Profile Name"
      RhcosAmi:
        default: "Red Hat Enterprise Linux CoreOS AMI ID"
      IgnitionLocation:
        default: "Worker Ignition Source"
      CertificateAuthorities:
        default: "Ignition CA String"
      WorkerSecurityGroupId:
        default: "Worker Security Group ID"

Resources:
  Worker0:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref RhcosAmi
      BlockDeviceMappings:
      - DeviceName: /dev/xvda
        Ebs:
          VolumeSize: "120"
          VolumeType: "gp2"
      IamInstanceProfile: !Ref WorkerInstanceProfileName
      InstanceType: !Ref WorkerInstanceType
      NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "WorkerSecurityGroupId"
        SubnetId: !Ref "Subnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"merge":[{"source":"${SOURCE}"}]},"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}"}]}},"version":"3.1.0"}}'
        - {
          SOURCE: !Ref IgnitionLocation,
          CA_BUNDLE: !Ref CertificateAuthorities,
        }
      Tags:
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "shared"

Outputs:
  PrivateIP:
    Description: The compute node private IP address.
    Value: !GetAtt Worker0.PrivateIp
```

**Additional resources**

- You can view details about the CloudFormation stacks that you create by navigating to the AWS CloudFormation console.

### 4.3.11.2. Creating the CloudFormation stack for compute machines

You can create a stack of AWS resources for the compute machines by using the CloudFormation template that was previously shared.

**IMPORTANT**

When you use the CloudFormation template for the control plane machines, the template provisions all three control plane machines with a single stack; however, when you use the CloudFormation template to deploy the compute machines, you must create the number of stacks based on the number that you defined in the **install-config.yaml** file. Each stack is provisioned once for each machine. To provision a new compute machine, you must change the stack name.

**Procedure**

- To create the CloudFormation stack for compute machines, run the following command:

```
$ aws cloudformation create-stack --stack-name <name> \ 1
     --template-body file://<template>.yaml \ 2
     --parameters file://<parameters>.json 3
```

1. Specify the **<name>** with the name for the CloudFormation stack, such as **cluster-worker-1**. You need the name of this stack if you remove the cluster.

2. Specify the relative path and the name of the CloudFormation template YAML file that you saved.

3. Specify the relative path and the name of the JSON file for the CloudFormation parameters.

**Example output**

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-worker-1/729ee301-1c2a-
11eb-348f-sd9888c65b59
```

### 4.3.12. Initializing the bootstrap sequence on AWS with user-provisioned infrastructure

After you create all of the required infrastructure in Amazon Web Services (AWS), you can start the bootstrap sequence that initializes the OpenShift Container Platform control plane.

**Prerequisites**

- You configured an AWS account.
- You added your AWS keys and region to your local AWS profile by running **aws configure**.
- You generated the Ignition config files for your cluster.
- You created and configured a VPC and associated subnets in AWS.
- You created and configured DNS, load balancers, and listeners in AWS.
- You created the security groups and roles required for your cluster in AWS.
- You created the bootstrap machine.
- You created the control plane machines.
- You created the worker nodes.

**Procedure**

1. Change to the directory that contains the installation program and start the bootstrap process that initializes the OpenShift Container Platform control plane:

```
$ ./openshift-install wait-for bootstrap-complete --dir <installation_directory> \ 1
    --log-level=info 2
```

1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

**Example output**

```
INFO Waiting up to 20m0s for the Kubernetes API at
https://api.mycluster.example.com:6443...
INFO API v1.32.3 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
INFO Time elapsed: 1s
```

If the command exits without a **FATAL** warning, your OpenShift Container Platform control plane has initialized.

NOTE

After the control plane initializes, it sets up the compute nodes and installs additional services in the form of Operators.

### Additional resources

- See Monitoring installation progress for details about monitoring the installation, bootstrap, and control plane logs as an OpenShift Container Platform installation progresses.
- See Gathering bootstrap node diagnostic data for information about troubleshooting issues related to the bootstrap process.
- You can view details about the running instances that are created by using the AWS EC2 console.

## 4.3.13. Logging in to the cluster by using the CLI

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

### Prerequisites

- You deployed an OpenShift Container Platform cluster.
- You installed the OpenShift CLI (**oc**).

### Procedure

1. Export the **kubeadmin** credentials by running the following command:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

   1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration by running the following command:

```
$ oc whoami
```

   **Example output**

```
system:admin
```

## 4.3.14. Approving the certificate signing requests for your machines

When you add machines to a cluster, two pending certificate signing requests (CSRs) are generated for each machine that you added. You must confirm that these CSRs are approved or, if necessary, approve them yourself. The client requests must be approved first, followed by the server requests.

### Prerequisites

- You added machines to your cluster.

### Procedure

1. Confirm that the cluster recognizes the machines:

   ```
   $ oc get nodes
   ```

   #### Example output

   ```
   NAME      STATUS    ROLES   AGE  VERSION
   master-0  Ready     master  63m  v1.32.3
   master-1  Ready     master  63m  v1.32.3
   master-2  Ready     master  64m  v1.32.3
   ```

   The output lists all of the machines that you created.

   NOTE

   The preceding output might not include the compute nodes, also known as worker nodes, until some CSRs are approved.

2. Review the pending CSRs and ensure that you see the client requests with the **Pending** or **Approved** status for each machine that you added to the cluster:

   ```
   $ oc get csr
   ```

   #### Example output

   ```
   NAME        AGE     REQUESTOR                                                                   CONDITION
   csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
   csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
   ...
   ```

   In this example, two machines are joining the cluster. You might see more approved CSRs in the list.

3. If the CSRs were not approved, after all of the pending CSRs for the machines you added are in **Pending** status, approve the CSRs for your cluster machines:

NOTE

Because the CSRs rotate automatically, approve your CSRs within an hour of adding the machines to the cluster. If you do not approve them within an hour, the certificates will rotate, and more than two certificates will be present for each node. You must approve all of these certificates. After the client CSR is approved, the Kubelet creates a secondary CSR for the serving certificate, which requires manual approval. Then, subsequent serving certificate renewal requests are automatically approved by the **machine-approver** if the Kubelet requests a new certificate with identical parameters.

NOTE

For clusters running on platforms that are not machine API enabled, such as bare metal and other user-provisioned infrastructure, you must implement a method of automatically approving the kubelet serving certificate requests (CSRs). If a request is not approved, then the **oc exec**, **oc rsh**, and **oc logs** commands cannot succeed, because a serving certificate is required when the API server connects to the kubelet. Any operation that contacts the Kubelet endpoint requires this certificate approval to be in place. The method must watch for new CSRs, confirm that the CSR was submitted by the **node-bootstrapper** service account in the **system:node** or **system:admin** groups, and confirm the identity of the node.

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

NOTE

Some Operators might not become available until some CSRs are approved.

4. Now that your client requests are approved, you must review the server requests for each machine that you added to the cluster:

```
$ oc get csr
```

**Example output**

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

–

5. If the remaining CSRs are not approved, and are in the **Pending** status, approve the CSRs for your cluster machines:

    - To approve them individually, run the following command for each valid CSR:

      ```
      $ oc adm certificate approve <csr_name> 1
      ```

      1 **<csr_name>** is the name of a CSR from the list of current CSRs.

    - To approve all pending CSRs, run the following command:

      ```
      $ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
      {{end}}{{end}}' | xargs oc adm certificate approve
      ```

6. After all client and server CSRs have been approved, the machines have the **Ready** status. Verify this by running the following command:

   ```
   $ oc get nodes
   ```

   **Example output**

   ```
   NAME      STATUS   ROLES   AGE  VERSION
   master-0  Ready    master  73m  v1.32.3
   master-1  Ready    master  73m  v1.32.3
   master-2  Ready    master  74m  v1.32.3
   worker-0  Ready    worker  11m  v1.32.3
   worker-1  Ready    worker  11m  v1.32.3
   ```

   NOTE

   It can take a few minutes after approval of the server CSRs for the machines to transition to the **Ready** status.

**Additional information**

- Certificate Signing Requests

### 4.3.15. Initial Operator configuration

After the control plane initializes, you must immediately configure some Operators so that they all become available.

**Prerequisites**

- Your control plane has initialized.

**Procedure**

1. Watch the cluster components come online:

   ```
   $ watch -n5 oc get clusteroperators
   ```

–

**Example output**

```
NAME                                      VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.19.0    True       False         False      19m
baremetal                                  4.19.0    True       False         False      37m
cloud-credential                           4.19.0    True       False         False      40m
cluster-autoscaler                         4.19.0    True       False         False      37m
config-operator                            4.19.0    True       False         False      38m
console                                    4.19.0    True       False         False      26m
csi-snapshot-controller                    4.19.0    True       False         False      37m
dns                                        4.19.0    True       False         False      37m
etcd                                       4.19.0    True       False         False      36m
image-registry                             4.19.0    True       False         False      31m
ingress                                    4.19.0    True       False         False      30m
insights                                   4.19.0    True       False         False      31m
kube-apiserver                             4.19.0    True       False         False      26m
kube-controller-manager                    4.19.0    True       False         False      36m
kube-scheduler                             4.19.0    True       False         False      36m
kube-storage-version-migrator              4.19.0    True       False         False      37m
machine-api                                4.19.0    True       False         False      29m
machine-approver                           4.19.0    True       False         False      37m
machine-config                             4.19.0    True       False         False      36m
marketplace                                4.19.0    True       False         False      37m
monitoring                                 4.19.0    True       False         False      29m
network                                    4.19.0    True       False         False      38m
node-tuning                                4.19.0    True       False         False      37m
openshift-apiserver                        4.19.0    True       False         False      32m
openshift-controller-manager               4.19.0    True       False         False      30m
openshift-samples                          4.19.0    True       False         False      32m
operator-lifecycle-manager                 4.19.0    True       False         False      37m
operator-lifecycle-manager-catalog         4.19.0    True       False         False      37m
operator-lifecycle-manager-packageserver   4.19.0    True       False         False      32m
service-ca                                 4.19.0    True       False         False      38m
storage                                    4.19.0    True       False         False      37m
```

2. Configure the Operators that are not available.

### 4.3.15.1. Image registry storage configuration

Amazon Web Services provides default storage, which means the Image Registry Operator is available after installation. However, if the Registry Operator cannot create an S3 bucket and automatically configure storage, you must manually configure registry storage.

Instructions are shown for configuring a persistent volume, which is required for production clusters. Where applicable, instructions are shown for configuring an empty directory as the storage location, which is available for only non-production clusters.

Additional instructions are provided for allowing the image registry to use block storage types by using the **Recreate** rollout strategy during upgrades.

You can configure registry storage for user-provisioned infrastructure in AWS to deploy OpenShift Container Platform to hidden regions. See Configuring the registry for AWS user-provisioned infrastructure for more information.

#### 4.3.15.1.1. Configuring registry storage for AWS with user-provisioned infrastructure

During installation, your cloud credentials are sufficient to create an Amazon S3 bucket and the Registry Operator will automatically configure storage.

If the Registry Operator cannot create an S3 bucket and automatically configure storage, you can create an S3 bucket and configure storage with the following procedure.

> **WARNING**
>
> To secure your registry images in AWS, block public access to the S3 bucket.

**Prerequisites**

- You have a cluster on AWS with user-provisioned infrastructure.
- For Amazon S3 storage, the secret is expected to contain two keys:
  - **REGISTRY_STORAGE_S3_ACCESSKEY**
  - **REGISTRY_STORAGE_S3_SECRETKEY**

**Procedure**

1. Set up a Bucket Lifecycle Policy to abort incomplete multipart uploads that are one day old.
2. Fill in the storage configuration in **configs.imageregistry.operator.openshift.io/cluster**:

```
$ oc edit configs.imageregistry.operator.openshift.io/cluster
```

**Example configuration**

```
apiVersion: imageregistry.operator.openshift.io/v1
kind: Config
metadata:
  name: cluster
spec:
  storage:
    s3:
      bucket: <bucket_name>
      region: <region_name>
```

#### 4.3.15.1.2. Configuring storage for the image registry in non-production clusters

You must configure storage for the Image Registry Operator. For non-production clusters, you can set the image registry to an empty directory. If you do so, all images are lost if you restart the registry.

**Procedure**

- To set the image registry storage to an empty directory:

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":
{"storage":{"emptyDir":{}}}}'
```

> **WARNING**
>
> Configure this option for only non-production clusters.

If you run this command before the Image Registry Operator initializes its components, the **oc patch** command fails with the following error:

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

Wait a few minutes and run the command again.

## 4.3.16. Deleting the bootstrap resources

After you complete the initial Operator configuration for the cluster, remove the bootstrap resources from Amazon Web Services (AWS).

### Prerequisites

- You completed the initial Operator configuration for your cluster.

### Procedure

1. Delete the bootstrap resources. If you used the CloudFormation template, delete its stack:
   - Delete the stack by using the AWS CLI:

     ```
     $ aws cloudformation delete-stack --stack-name <name> 1
     ```

     1 **<name>** is the name of your bootstrap stack.

   - Delete the stack by using the AWS CloudFormation console.

## 4.3.17. Creating the Ingress DNS Records

If you removed the DNS Zone configuration, manually create DNS records that point to the Ingress load balancer. You can create either a wildcard record or specific records. While the following procedure uses A records, you can use other record types that you require, such as CNAME or alias.

### Prerequisites

- You deployed an OpenShift Container Platform cluster on Amazon Web Services (AWS) that uses infrastructure that you provisioned.
- You installed the OpenShift CLI (**oc**).

- You installed the **jq** package.
- You downloaded the AWS CLI and installed it on your computer. See Install the AWS CLI Using the Bundled Installer (Linux, macOS, or Unix).

**Procedure**

1. Determine the routes to create.
   - To create a wildcard record, use ***.apps.<cluster_name>.<domain_name>**, where **<cluster_name>** is your cluster name, and **<domain_name>** is the Route 53 base domain for your OpenShift Container Platform cluster.
   - To create specific records, you must create a record for each route that your cluster uses, as shown in the output of the following command:

     ```
     $ oc get --all-namespaces -o jsonpath='{range .items[*]}{range .status.ingress[*]}{.host}
     {"\n"}{end}{end}' routes
     ```

     **Example output**

     ```
     oauth-openshift.apps.<cluster_name>.<domain_name>
     console-openshift-console.apps.<cluster_name>.<domain_name>
     downloads-openshift-console.apps.<cluster_name>.<domain_name>
     alertmanager-main-openshift-monitoring.apps.<cluster_name>.<domain_name>
     prometheus-k8s-openshift-monitoring.apps.<cluster_name>.<domain_name>
     ```

2. Retrieve the Ingress Operator load balancer status and note the value of the external IP address that it uses, which is shown in the **EXTERNAL-IP** column:

   ```
   $ oc -n openshift-ingress get service router-default
   ```

   **Example output**

   ```
   NAME           TYPE           CLUSTER-IP      EXTERNAL-IP                            PORT(S)
   AGE
   router-default   LoadBalancer   172.30.62.215   ab3...28.us-east-2.elb.amazonaws.com
   80:31499/TCP,443:30693/TCP   5m
   ```

3. Locate the hosted zone ID for the load balancer:

   ```
   $ aws elb describe-load-balancers | jq -r '.LoadBalancerDescriptions[] | select(.DNSName ==
   "<external_ip>").CanonicalHostedZoneNameID' (1)
   ```

   (1) For **<external_ip>**, specify the value of the external IP address of the Ingress Operator load balancer that you obtained.

   **Example output**

   ```
   Z3AADJGX6KTTL2
   ```

   The output of this command is the load balancer hosted zone ID.

4. Obtain the public hosted zone ID for your cluster's domain:

```
$ aws route53 list-hosted-zones-by-name \
        --dns-name "<domain_name>" \ (1)
        --query 'HostedZones[? Config.PrivateZone != `true` && Name ==
`<domain_name>.`].Id' (2)
        --output text
```

(1) (2) For **<domain_name>**, specify the Route 53 base domain for your OpenShift Container Platform cluster.

**Example output**

```
/hostedzone/Z3URY6TWQ91KVV
```

The public hosted zone ID for your domain is shown in the command output. In this example, it is **Z3URY6TWQ91KVV**.

5. Add the alias records to your private zone:

```
$ aws route53 change-resource-record-sets --hosted-zone-id "<private_hosted_zone_id>" --
change-batch '{ (1)
>   "Changes": [
>     {
>       "Action": "CREATE",
>       "ResourceRecordSet": {
>         "Name": "\\052.apps.<cluster_domain>", (2)
>         "Type": "A",
>         "AliasTarget":{
>           "HostedZoneId": "<hosted_zone_id>", (3)
>           "DNSName": "<external_ip>.", (4)
>           "EvaluateTargetHealth": false
>         }
>       }
>     }
>   ]
> }'
```

(1) For **<private_hosted_zone_id>**, specify the value from the output of the CloudFormation template for DNS and load balancing.

(2) For **<cluster_domain>**, specify the domain or subdomain that you use with your OpenShift Container Platform cluster.

(3) For **<hosted_zone_id>**, specify the public hosted zone ID for the load balancer that you obtained.

(4) For **<external_ip>**, specify the value of the external IP address of the Ingress Operator load balancer. Ensure that you include the trailing period (**.**) in this parameter value.

6. Add the records to your public zone:

```
$ aws route53 change-resource-record-sets --hosted-zone-id "<public_hosted_zone_id>"" --
change-batch '{ 1
>   "Changes": [
>     {
>       "Action": "CREATE",
>       "ResourceRecordSet": {
>         "Name": "\\052.apps.<cluster_domain>", 2
>         "Type": "A",
>         "AliasTarget":{
>           "HostedZoneId": "<hosted_zone_id>", 3
>           "DNSName": "<external_ip>.", 4
>           "EvaluateTargetHealth": false
>         }
>       }
>     }
>   ]
> }'
```

1. For **<public_hosted_zone_id>**, specify the public hosted zone for your domain.
2. For **<cluster_domain>**, specify the domain or subdomain that you use with your OpenShift Container Platform cluster.
3. For **<hosted_zone_id>**, specify the public hosted zone ID for the load balancer that you obtained.
4. For **<external_ip>**, specify the value of the external IP address of the Ingress Operator load balancer. Ensure that you include the trailing period (**.**) in this parameter value.

## 4.3.18. Completing an AWS installation on user-provisioned infrastructure

After you start the OpenShift Container Platform installation on Amazon Web Service (AWS) user-provisioned infrastructure, monitor the deployment to completion.

### Prerequisites

- You removed the bootstrap node for an OpenShift Container Platform cluster on user-provisioned AWS infrastructure.
- You installed the **oc** CLI.

### Procedure

- From the directory that contains the installation program, complete the cluster installation:

  ```
  $ ./openshift-install --dir <installation_directory> wait-for install-complete 1
  ```

  1. For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

  #### Example output

```
INFO Waiting up to 40m0s for the cluster at https://api.mycluster.example.com:6443 to
initialize...
INFO Waiting up to 10m0s for the openshift-console route to be created...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 1s
```

IMPORTANT

- The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

- It is recommended that you use Ignition config files within 12 hours after they are generated because the 24-hour certificate rotates from 16 to 22 hours after the cluster is installed. By using the Ignition config files within 12 hours, you can avoid installation failure if the certificate update runs during installation.

### 4.3.19. Logging in to the cluster by using the web console

The **kubeadmin** user exists by default after an OpenShift Container Platform installation. You can log in to your cluster as the **kubeadmin** user by using the OpenShift Container Platform web console.

#### Prerequisites

- You have access to the installation host.
- You completed a cluster installation and all cluster Operators are available.

#### Procedure

1. Obtain the password for the **kubeadmin** user from the **kubeadmin-password** file on the installation host:

   ```
   $ cat <installation_directory>/auth/kubeadmin-password
   ```

   NOTE

   Alternatively, you can obtain the **kubeadmin** password from the **<installation_directory>/.openshift_install.log** log file on the installation host.

2. List the OpenShift Container Platform web console route:

```
$ oc get routes -n openshift-console | grep 'console-openshift'
```

NOTE

Alternatively, you can obtain the OpenShift Container Platform route from the **<installation_directory>/.openshift_install.log** log file on the installation host.

**Example output**

```
console     console-openshift-console.apps.<cluster_name>.<base_domain>            console
https   reencrypt/Redirect   None
```

3. Navigate to the route detailed in the output of the preceding command in a web browser and log in as the **kubeadmin** user.

**Additional resources**

- Accessing the web console

### 4.3.20. Additional resources

- Working with stacks(AWS documentation)

### 4.3.21. Next steps

- Validating an installation.
- Customize your cluster.
- If necessary, you can Remote health reporting.
- If necessary, you can remove cloud provider credentials.

## 4.4. INSTALLING A CLUSTER ON AWS IN A DISCONNECTED ENVIRONMENT WITH USER-PROVISIONED INFRASTRUCTURE

In OpenShift Container Platform version 4.19, you can install a cluster on Amazon Web Services (AWS) using infrastructure that you provide and an internal mirror of the installation release content.

IMPORTANT

While you can install an OpenShift Container Platform cluster by using mirrored installation release content, your cluster still requires internet access to use the AWS APIs.

One way to create this infrastructure is to use the provided CloudFormation templates. You can modify the templates to customize your infrastructure or use the information that they contain to create AWS objects according to your company's policies.

IMPORTANT

The steps for performing a user-provisioned infrastructure installation are provided as an example only. Installing a cluster with infrastructure you provide requires knowledge of the cloud provider and the installation process of OpenShift Container Platform. Several CloudFormation templates are provided to assist in completing these steps or to help model your own. You are also free to create the required resources through other methods; the templates are just an example.

### 4.4.1. Prerequisites

- You reviewed details about the OpenShift Container Platform installation and update processes.
- You read the documentation on selecting a cluster installation method and preparing it for users.
- You created a mirror registry on your mirror host and obtained the **imageContentSources** data for your version of OpenShift Container Platform.

  IMPORTANT

  Because the installation media is on the mirror host, you can use that computer to complete all installation steps.

- You configured an AWS account to host the cluster.

  IMPORTANT

  If you have an AWS profile stored on your computer, it must not use a temporary session token that you generated while using a multi-factor authentication device. The cluster continues to use your current AWS credentials to create AWS resources for the entire life of the cluster, so you must use key-based, long-term credentials. To generate appropriate keys, see Managing Access Keys for IAM Users in the AWS documentation. You can supply the keys when you run the installation program.

- You prepared the user-provisioned infrastructure.
- You downloaded the AWS CLI and installed it on your computer. See Install the AWS CLI Using the Bundled Installer (Linux, macOS, or UNIX) in the AWS documentation.
- If you use a firewall and plan to use the Telemetry service, you configured the firewall to allow the sites that your cluster requires access to.

  NOTE

  Be sure to also review this site list if you are configuring a proxy.

- If the cloud identity and access management (IAM) APIs are not accessible in your environment, or if you do not want to store an administrator-level credential secret in the **kube-system** namespace, you can manually create and maintain long-term credentials.

### 4.4.2. About installations in restricted networks

In OpenShift Container Platform 4.19, you can perform an installation that does not require an active connection to the internet to obtain software components. Restricted network installations can be completed using installer-provisioned infrastructure or user-provisioned infrastructure, depending on the cloud platform to which you are installing the cluster.

If you choose to perform a restricted network installation on a cloud platform, you still require access to its cloud APIs. Some cloud functions, like Amazon Web Service's Route 53 DNS and IAM services, require internet access. Depending on your network, you might require less internet access for an installation on bare metal hardware, Nutanix, or on VMware vSphere.

To complete a restricted network installation, you must create a registry that mirrors the contents of the OpenShift image registry and contains the installation media. You can create this registry on a mirror host, which can access both the internet and your closed network, or by using other methods that meet your restrictions.

IMPORTANT

Because of the complexity of the configuration for user-provisioned installations, consider completing a standard user-provisioned infrastructure installation before you attempt a restricted network installation using user-provisioned infrastructure. Completing this test installation might make it easier to isolate and troubleshoot any issues that might arise during your installation in a restricted network.

#### 4.4.2.1. Additional limits

Clusters in restricted networks have the following additional limitations and restrictions:

- The **ClusterVersion** status includes an **Unable to retrieve available updates** error.
- By default, you cannot use the contents of the Developer Catalog because you cannot access the required image stream tags.

### 4.4.3. Creating the installation files for AWS

To install OpenShift Container Platform on Amazon Web Services using user-provisioned infrastructure, you must generate the files that the installation program needs to deploy your cluster and modify them so that the cluster creates only the machines that it will use. You generate and customize the **install-config.yaml** file, Kubernetes manifests, and Ignition config files. You also have the option to first set up a separate **var** partition during the preparation phases of installation.

#### 4.4.3.1. Optional: Creating a separate /var partition

It is recommended that disk partitioning for OpenShift Container Platform be left to the installer. However, there are cases where you might want to create separate partitions in a part of the filesystem that you expect to grow.

OpenShift Container Platform supports the addition of a single partition to attach storage to either the **/var** partition or a subdirectory of **/var**. For example:

- **/var/lib/containers**: Holds container-related content that can grow as more images and containers are added to a system.

- **/var/lib/etcd**: Holds data that you might want to keep separate for purposes such as performance optimization of etcd storage.
- **/var**: Holds data that you might want to keep separate for purposes such as auditing.

Storing the contents of a **/var** directory separately makes it easier to grow storage for those areas as needed and reinstall OpenShift Container Platform at a later date and keep that data intact. With this method, you will not have to pull all your containers again, nor will you have to copy massive log files when you update systems.

Because **/var** must be in place before a fresh installation of Red Hat Enterprise Linux CoreOS (RHCOS), the following procedure sets up the separate **/var** partition by creating a machine config manifest that is inserted during the **openshift-install** preparation phases of an OpenShift Container Platform installation.

IMPORTANT

If you follow the steps to create a separate **/var** partition in this procedure, it is not necessary to create the Kubernetes manifest and Ignition config files again as described later in this section.

### Procedure

1. Create a directory to hold the OpenShift Container Platform installation files:

```
$ mkdir $HOME/clusterconfig
```

2. Run **openshift-install** to create a set of files in the **manifest** and **openshift** subdirectories. Answer the system questions as you are prompted:

```
$ openshift-install create manifests --dir $HOME/clusterconfig
```

#### Example output

```
? SSH Public Key ...
INFO Credentials loaded from the "myprofile" profile in file "/home/myuser/.aws/credentials"
INFO Consuming Install Config from target directory
INFO Manifests created in: $HOME/clusterconfig/manifests and
$HOME/clusterconfig/openshift
```

3. Optional: Confirm that the installation program created manifests in the **clusterconfig/openshift** directory:

```
$ ls $HOME/clusterconfig/openshift/
```

#### Example output

```
99_kubeadmin-password-secret.yaml
99_openshift-cluster-api_master-machines-0.yaml
99_openshift-cluster-api_master-machines-1.yaml
99_openshift-cluster-api_master-machines-2.yaml
...
```

4. Create a Butane config that configures the additional partition. For example, name the file **$HOME/clusterconfig/98-var-partition.bu**, change the disk device name to the name of the storage device on the **worker** systems, and set the storage size as appropriate. This example places the **/var** directory on a separate partition:

```
variant: openshift
version: 4.19.0
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-var-partition
storage:
  disks:
  - device: /dev/disk/by-id/<device_name> 1
    partitions:
    - label: var
      start_mib: <partition_start_offset> 2
      size_mib: <partition_size> 3
      number: 5
  filesystems:
    - device: /dev/disk/by-partlabel/var
      path: /var
      format: xfs
      mount_options: [defaults, prjquota] 4
      with_mount_unit: true
```

1 The storage device name of the disk that you want to partition.

2 When adding a data partition to the boot disk, a minimum value of 25000 MiB (Mebibytes) is recommended. The root file system is automatically resized to fill all available space up to the specified offset. If no value is specified, or if the specified value is smaller than the recommended minimum, the resulting root file system will be too small, and future reinstalls of RHCOS might overwrite the beginning of the data partition.

3 The size of the data partition in mebibytes.

4 The **prjquota** mount option must be enabled for filesystems used for container storage.

NOTE

When creating a separate **/var** partition, you cannot use different instance types for worker nodes, if the different instance types do not have the same device name.

5. Create a manifest from the Butane config and save it to the **clusterconfig/openshift** directory. For example, run the following command:

```
$ butane $HOME/clusterconfig/98-var-partition.bu -o $HOME/clusterconfig/openshift/98-var-partition.yaml
```

6. Run **openshift-install** again to create Ignition configs from a set of files in the **manifest** and **openshift** subdirectories:

```
$ openshift-install create ignition-configs --dir $HOME/clusterconfig
```

```
$ ls $HOME/clusterconfig/
auth  bootstrap.ign  master.ign  metadata.json  worker.ign
```

You can now use the Ignition config files as input to the installation procedures to install Red Hat Enterprise Linux CoreOS (RHCOS) systems.

### 4.4.3.2. Creating the installation configuration file

Generate and customize the installation configuration file that the installation program needs to deploy your cluster.

#### Prerequisites

- You obtained the OpenShift Container Platform installation program for user-provisioned infrastructure and the pull secret for your cluster. For a restricted network installation, these files are on your mirror host.
- You checked that you are deploying your cluster to an AWS Region with an accompanying Red Hat Enterprise Linux CoreOS (RHCOS) AMI published by Red Hat. If you are deploying to an AWS Region that requires a custom AMI, such as an AWS GovCloud Region, you must create the **install-config.yaml** file manually.

#### Procedure

1. Create the **install-config.yaml** file.

   a. Change to the directory that contains the installation program and run the following command:

   ```
   $ ./openshift-install create install-config --dir <installation_directory> 1
   ```

   1 For **<installation_directory>**, specify the directory name to store the files that the installation program creates.

   **IMPORTANT**

   Specify an empty directory. Some installation assets, like bootstrap X.509 certificates have short expiration intervals, so you must not reuse an installation directory. If you want to reuse individual files from another cluster installation, you can copy them into your directory. However, the file names for the installation assets might change between releases. Use caution when copying installation files from an earlier OpenShift Container Platform version.

   b. At the prompts, provide the configuration details for your cloud:

      i. Optional: Select an SSH key to use to access your cluster machines.

NOTE

For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.

ii. Select **aws** as the platform to target.

iii. If you do not have an AWS profile stored on your computer, enter the AWS access key ID and secret access key for the user that you configured to run the installation program.

NOTE

The AWS access key ID and secret access key are stored in **~/.aws/credentials** in the home directory of the current user on the installation host. You are prompted for the credentials by the installation program if the credentials for the exported profile are not present in the file. Any credentials that you provide to the installation program are stored in the file.

iv. Select the AWS Region to deploy the cluster to.

v. Select the base domain for the Route 53 service that you configured for your cluster.

vi. Enter a descriptive name for your cluster.

vii. Paste the pull secret from Red Hat OpenShift Cluster Manager .

2. Edit the **install-config.yaml** file to give the additional information that is required for an installation in a restricted network.

   a. Update the **pullSecret** value to contain the authentication information for your registry:

```
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email":
"you@example.com"}}}'
```

   For **<local_registry>**, specify the registry domain name, and optionally the port, that your mirror registry uses to serve content. For example **registry.example.com** or **registry.example.com:5000**. For **<credentials>**, specify the base64-encoded user name and password for your mirror registry.

   b. Add the **additionalTrustBundle** parameter and value. The value must be the contents of the certificate file that you used for your mirror registry. The certificate file can be an existing, trusted certificate authority or the self-signed certificate that you generated for the mirror registry.

```
additionalTrustBundle: |
  -----BEGIN CERTIFICATE-----

ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
```

   c. Add the image content resources:

```
imageContentSources:
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

Use the **imageContentSources** section from the output of the command to mirror the repository or the values that you used when you mirrored the content from the media that you brought into your restricted network.

d. Optional: Set the publishing strategy to **Internal**:

```
publish: Internal
```

By setting this option, you create an internal Ingress Controller and a private load balancer.

3. Optional: Back up the **install-config.yaml** file.

IMPORTANT

The **install-config.yaml** file is consumed during the installation process. If you want to reuse the file, you must back it up now.

### Additional resources

- See Configuration and credential file settings in the AWS documentation for more information about AWS profile and credential configuration.

#### 4.4.3.3. Configuring the cluster-wide proxy during installation

Production environments can deny direct access to the internet and instead have an HTTP or HTTPS proxy available. You can configure a new OpenShift Container Platform cluster to use a proxy by configuring the proxy settings in the **install-config.yaml** file.

### Prerequisites

- You have an existing **install-config.yaml** file.
- You reviewed the sites that your cluster requires access to and determined whether any of them need to bypass the proxy. By default, all cluster egress traffic is proxied, including calls to hosting cloud provider APIs. You added sites to the **Proxy** object's **spec.noProxy** field to bypass the proxy if necessary.

NOTE

The **Proxy** object **status.noProxy** field is populated with the values of the **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, and **networking.serviceNetwork[]** fields from your installation configuration.

For installations on Amazon Web Services (AWS), Google Cloud, Microsoft Azure, and Red Hat OpenStack Platform (RHOSP), the **Proxy** object **status.noProxy** field is also populated with the instance metadata endpoint (**169.254.169.254**).

## Procedure

1. Edit your **install-config.yaml** file and add the proxy settings. For example:

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: ec2.<aws_region>.amazonaws.com,elasticloadbalancing.
<aws_region>.amazonaws.com,s3.<aws_region>.amazonaws.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1 A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be **http**.

2 A proxy URL to use for creating HTTPS connections outside the cluster.

3 A comma-separated list of destination domain names, IP addresses, or other network CIDRs to exclude from proxying. Preface a domain with **.** to match subdomains only. For example, **.y.com** matches **x.y.com**, but not **y.com**. Use * to bypass the proxy for all destinations. If you have added the Amazon **EC2**,**Elastic Load Balancing**, and **S3** VPC endpoints to your VPC, you must add these endpoints to the **noProxy** field.

4 If provided, the installation program generates a config map that is named **user-ca-bundle** in the **openshift-config** namespace that contains one or more additional CA certificates that are required for proxying HTTPS connections. The Cluster Network Operator then creates a **trusted-ca-bundle** config map that merges these contents with the Red Hat Enterprise Linux CoreOS (RHCOS) trust bundle, and this config map is referenced in the **trustedCA** field of the **Proxy** object. The **additionalTrustBundle** field is required unless the proxy's identity certificate is signed by an authority from the RHCOS trust bundle.

5 Optional: The policy to determine the configuration of the **Proxy** object to reference the **user-ca-bundle** config map in the **trustedCA** field. The allowed values are **Proxyonly** and **Always**. Use **Proxyonly** to reference the **user-ca-bundle** config map only when **http/https** proxy is configured. Use **Always** to always reference the **user-ca-bundle** config map. The default value is **Proxyonly**.

NOTE

The installation program does not support the proxy **readinessEndpoints** field.

NOTE

If the installer times out, restart and then complete the deployment by using the **wait-for** command of the installer. For example:

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. Save the file and reference it when installing OpenShift Container Platform.

The installation program creates a cluster-wide proxy that is named **cluster** that uses the proxy settings in the provided **install-config.yaml** file. If no proxy settings are provided, a **cluster Proxy** object is still created, but it will have a nil **spec**.

NOTE

Only the **Proxy** object named **cluster** is supported, and no additional proxies can be created.

#### 4.4.3.4. Creating the Kubernetes manifest and Ignition config files

Because you must modify some cluster definition files and manually start the cluster machines, you must generate the Kubernetes manifest and Ignition config files that the cluster needs to configure the machines.

The installation configuration file transforms into the Kubernetes manifests. The manifests wrap into the Ignition configuration files, which are later used to configure the cluster machines.

IMPORTANT

- The Ignition config files that the OpenShift Container Platform installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

- It is recommended that you use Ignition config files within 12 hours after they are generated because the 24-hour certificate rotates from 16 to 22 hours after the cluster is installed. By using the Ignition config files within 12 hours, you can avoid installation failure if the certificate update runs during installation.

**Prerequisites**

- You obtained the OpenShift Container Platform installation program. For a restricted network installation, these files are on your mirror host.

- You created the **install-config.yaml** installation configuration file.

### Procedure

1. Change to the directory that contains the OpenShift Container Platform installation program and generate the Kubernetes manifests for the cluster:

   ```
   $ ./openshift-install create manifests --dir <installation_directory> 1
   ```

   1 For **<installation_directory>**, specify the installation directory that contains the **install-config.yaml** file you created.

2. Remove the Kubernetes manifest files that define the control plane machines:

   ```
   $ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
   ```

   By removing these files, you prevent the cluster from automatically generating control plane machines.

3. Remove the Kubernetes manifest files that define the control plane machine set:

   ```
   $ rm -f <installation_directory>/openshift/99_openshift-machine-api_master-control-plane-machine-set.yaml
   ```

4. Remove the Kubernetes manifest files that define the worker machines:

   ```
   $ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
   ```

   IMPORTANT

   If you disabled the **MachineAPI** capability when installing a cluster on user-provisioned infrastructure, you must remove the Kubernetes manifest files that define the worker machines. Otherwise, your cluster fails to install.

   Because you create and manage the worker machines yourself, you do not need to initialize these machines.

5. Check that the **mastersSchedulable** parameter in the **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes manifest file is set to **false**. This setting prevents pods from being scheduled on the control plane machines:

   a. Open the **<installation_directory>/manifests/cluster-scheduler-02-config.yml** file.

   b. Locate the **mastersSchedulable** parameter and ensure that it is set to **false**.

   c. Save and exit the file.

6. Optional: If you do not want the Ingress Operator to create DNS records on your behalf, remove the **privateZone** and **publicZone** sections from the **<installation_directory>/manifests/cluster-dns-02-config.yml** DNS configuration file:

   ```
   apiVersion: config.openshift.io/v1
   kind: DNS
   metadata:
   ```

```
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
  publicZone: 2
    id: example.openshift.com
status: {}
```

1 2 Remove this section completely.

If you do so, you must add ingress DNS records manually in a later step.

7. To create the Ignition configuration files, run the following command from the directory that contains the installation program:

```
$ ./openshift-install create ignition-configs --dir <installation_directory> 1
```

1 For **<installation_directory>**, specify the same installation directory.

Ignition config files are created for the bootstrap, control plane, and compute nodes in the installation directory. The **kubeadmin-password** and **kubeconfig** files are created in the **./<installation_directory>/auth** directory:

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

### Additional resources

- Manually creating long-term credentials

## 4.4.4. Extracting the infrastructure name

The Ignition config files contain a unique cluster identifier that you can use to uniquely identify your cluster in Amazon Web Services. The infrastructure name is also used to locate the appropriate AWS resources during an OpenShift Container Platform installation. The provided CloudFormation templates contain references to this infrastructure name, so you must extract it.

### Prerequisites

- You obtained the OpenShift Container Platform installation program and the pull secret for your cluster.
- You generated the Ignition config files for your cluster.
- You installed the **jq** package.

**Procedure**

- To extract and view the infrastructure name from the Ignition config file metadata, run the following command:

  ```
  $ jq -r .infraID <installation_directory>/metadata.json 1
  ```

  1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

  **Example output**

  ```
  openshift-vw9j6 1
  ```

  1 The output of this command is your cluster name and a random string.

## 4.4.5. Creating a VPC in AWS

You must create a Virtual Private Cloud (VPC) in Amazon Web Services (AWS) for your OpenShift Container Platform cluster to use. You can customize the VPC to meet your requirements, including VPN and route tables.

You can use the provided CloudFormation template and a custom parameter file to create a stack of AWS resources that represent the VPC.

NOTE

If you do not use the provided CloudFormation template to create your AWS infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

**Prerequisites**

- You configured an AWS account.
- You added your AWS keys and region to your local AWS profile by running **aws configure**.
- You generated the Ignition config files for your cluster.

**Procedure**

1. Create a JSON file that contains the parameter values that the template requires:

   ```
   [
     {
       "ParameterKey": "VpcCidr", 1
       "ParameterValue": "10.0.0.0/16" 2
     },
     {
       "ParameterKey": "AvailabilityZoneCount", 3
       "ParameterValue": "1" 4
   ```

```
  },
  {
    "ParameterKey": "SubnetBits", 5
    "ParameterValue": "12" 6
  }
]
```

1 The CIDR block for the VPC.

2 Specify a CIDR block in the format **x.x.x.x/16-24**.

3 The number of availability zones to deploy the VPC in.

4 Specify an integer between **1** and **3**.

5 The size of each subnet in each availability zone.

6 Specify an integer between **5** and **13**, where **5** is **/27** and **13** is **/19**.

2. Copy the template from the **CloudFormation template for the VPC**section of this topic and save it as a YAML file on your computer. This template describes the VPC that your cluster requires.

3. Launch the CloudFormation template to create a stack of AWS resources that represent the VPC:

   IMPORTANT

   You must enter the command on a single line.

```
$ aws cloudformation create-stack --stack-name <name> 1
     --template-body file://<template>.yaml 2
     --parameters file://<parameters>.json 3
```

1 **<name>** is the name for the CloudFormation stack, such as **cluster-vpc**. You need the name of this stack if you remove the cluster.

2 **<template>** is the relative path to and name of the CloudFormation template YAML file that you saved.

3 **<parameters>** is the relative path to and name of the CloudFormation parameters JSON file.

### Example output

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-vpc/dbedae40-2fd3-11eb-
820e-12a48460849f
```

4. Confirm that the template components exist:

```
$ aws cloudformation describe-stacks --stack-name <name>
```

After the **StackStatus** displays **CREATE_COMPLETE**, the output displays values for the following parameters. You must provide these parameter values to the other CloudFormation templates that you run to create your cluster:

| | |
|---|---|
| **VpcId** | The ID of your VPC. |
| **PublicSubnetIds** | The IDs of the new public subnets. |
| **PrivateSubnetIds** | The IDs of the new private subnets. |

### 4.4.5.1. CloudFormation template for the VPC

You can use the following CloudFormation template to deploy the VPC that you need for your OpenShift Container Platform cluster.

**Example 4.25. CloudFormation template for the VPC**

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for Best Practice VPC with 1-3 AZs

Parameters:
  VpcCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])(\/(1[6-9]|2[0-4]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/16-24.
    Default: 10.0.0.0/16
    Description: CIDR block for VPC.
    Type: String
  AvailabilityZoneCount:
    ConstraintDescription: "The number of availability zones. (Min: 1, Max: 3)"
    MinValue: 1
    MaxValue: 3
    Default: 1
    Description: "How many AZs to create VPC subnets for. (Min: 1, Max: 3)"
    Type: Number
  SubnetBits:
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/19-27.
    MinValue: 5
    MaxValue: 13
    Default: 12
    Description: "Size of each subnet to create within the availability zones. (Min: 5 = /27, Max: 13 = /19)"
    Type: Number

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcCidr
      - SubnetBits
    - Label:
        default: "Availability Zones"
      Parameters:
      - AvailabilityZoneCount
    ParameterLabels:
      AvailabilityZoneCount:
        default: "Availability Zone Count"
      VpcCidr:
        default: "VPC CIDR"
      SubnetBits:
        default: "Bits Per Subnet"

Conditions:
  DoAz3: !Equals [3, !Ref AvailabilityZoneCount]
  DoAz2: !Or [!Equals [2, !Ref AvailabilityZoneCount], Condition: DoAz3]

Resources:
  VPC:
    Type: "AWS::EC2::VPC"
    Properties:
      EnableDnsSupport: "true"
      EnableDnsHostnames: "true"
      CidrBlock: !Ref VpcCidr
  PublicSubnet:
    Type: "AWS::EC2::Subnet"
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [0, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 0
      - Fn::GetAZs: !Ref "AWS::Region"
  PublicSubnet2:
    Type: "AWS::EC2::Subnet"
    Condition: DoAz2
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [1, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 1
      - Fn::GetAZs: !Ref "AWS::Region"
  PublicSubnet3:
    Type: "AWS::EC2::Subnet"
    Condition: DoAz3
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [2, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 2
      - Fn::GetAZs: !Ref "AWS::Region"
  InternetGateway:
    Type: "AWS::EC2::InternetGateway"
  GatewayToInternet:
    Type: "AWS::EC2::VPCGatewayAttachment"
    Properties:
      VpcId: !Ref VPC
      InternetGatewayId: !Ref InternetGateway
  PublicRouteTable:
    Type: "AWS::EC2::RouteTable"
    Properties:
      VpcId: !Ref VPC
  PublicRoute:
    Type: "AWS::EC2::Route"
    DependsOn: GatewayToInternet
    Properties:
      RouteTableId: !Ref PublicRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId: !Ref InternetGateway
  PublicSubnetRouteTableAssociation:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Properties:
      SubnetId: !Ref PublicSubnet
      RouteTableId: !Ref PublicRouteTable
  PublicSubnetRouteTableAssociation2:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Condition: DoAz2
    Properties:
      SubnetId: !Ref PublicSubnet2
      RouteTableId: !Ref PublicRouteTable
  PublicSubnetRouteTableAssociation3:
    Condition: DoAz3
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Properties:
      SubnetId: !Ref PublicSubnet3
      RouteTableId: !Ref PublicRouteTable
  PrivateSubnet:
    Type: "AWS::EC2::Subnet"
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [3, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 0
      - Fn::GetAZs: !Ref "AWS::Region"
  PrivateRouteTable:
    Type: "AWS::EC2::RouteTable"
    Properties:
      VpcId: !Ref VPC
  PrivateSubnetRouteTableAssociation:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Properties:
      SubnetId: !Ref PrivateSubnet
      RouteTableId: !Ref PrivateRouteTable
  NAT:
    DependsOn:
    - GatewayToInternet
    Type: "AWS::EC2::NatGateway"
    Properties:
      AllocationId:
        "Fn::GetAtt":
        - EIP
        - AllocationId
      SubnetId: !Ref PublicSubnet
```

```
EIP:
  Type: "AWS::EC2::EIP"
  Properties:
    Domain: vpc
Route:
  Type: "AWS::EC2::Route"
  Properties:
    RouteTableId:
      Ref: PrivateRouteTable
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId:
      Ref: NAT
PrivateSubnet2:
  Type: "AWS::EC2::Subnet"
  Condition: DoAz2
  Properties:
    VpcId: !Ref VPC
    CidrBlock: !Select [4, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
    AvailabilityZone: !Select
    - 1
    - Fn::GetAZs: !Ref "AWS::Region"
PrivateRouteTable2:
  Type: "AWS::EC2::RouteTable"
  Condition: DoAz2
  Properties:
    VpcId: !Ref VPC
PrivateSubnetRouteTableAssociation2:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Condition: DoAz2
  Properties:
    SubnetId: !Ref PrivateSubnet2
    RouteTableId: !Ref PrivateRouteTable2
NAT2:
  DependsOn:
  - GatewayToInternet
  Type: "AWS::EC2::NatGateway"
  Condition: DoAz2
  Properties:
    AllocationId:
      "Fn::GetAtt":
      - EIP2
      - AllocationId
    SubnetId: !Ref PublicSubnet2
EIP2:
  Type: "AWS::EC2::EIP"
  Condition: DoAz2
  Properties:
    Domain: vpc
Route2:
  Type: "AWS::EC2::Route"
  Condition: DoAz2
  Properties:
    RouteTableId:
      Ref: PrivateRouteTable2
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId:
      Ref: NAT2
  PrivateSubnet3:
    Type: "AWS::EC2::Subnet"
    Condition: DoAz3
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [5, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 2
      - Fn::GetAZs: !Ref "AWS::Region"
  PrivateRouteTable3:
    Type: "AWS::EC2::RouteTable"
    Condition: DoAz3
    Properties:
      VpcId: !Ref VPC
  PrivateSubnetRouteTableAssociation3:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Condition: DoAz3
    Properties:
      SubnetId: !Ref PrivateSubnet3
      RouteTableId: !Ref PrivateRouteTable3
  NAT3:
    DependsOn:
    - GatewayToInternet
    Type: "AWS::EC2::NatGateway"
    Condition: DoAz3
    Properties:
      AllocationId:
        "Fn::GetAtt":
        - EIP3
        - AllocationId
      SubnetId: !Ref PublicSubnet3
  EIP3:
    Type: "AWS::EC2::EIP"
    Condition: DoAz3
    Properties:
      Domain: vpc
  Route3:
    Type: "AWS::EC2::Route"
    Condition: DoAz3
    Properties:
      RouteTableId:
        Ref: PrivateRouteTable3
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId:
        Ref: NAT3
  S3Endpoint:
    Type: AWS::EC2::VPCEndpoint
    Properties:
      PolicyDocument:
        Version: 2012-10-17
        Statement:
        - Effect: Allow
          Principal: '*'
          Action:
          - '*'
          Resource:
          - '*'
        RouteTableIds:
        - !Ref PublicRouteTable
        - !Ref PrivateRouteTable
        - !If [DoAz2, !Ref PrivateRouteTable2, !Ref "AWS::NoValue"]
        - !If [DoAz3, !Ref PrivateRouteTable3, !Ref "AWS::NoValue"]
        ServiceName: !Join
        - ''
        - - com.amazonaws.
          - !Ref 'AWS::Region'
          - .s3
        VpcId: !Ref VPC

Outputs:
  VpcId:
    Description: ID of the new VPC.
    Value: !Ref VPC
  PublicSubnetIds:
    Description: Subnet IDs of the public subnets.
    Value:
      !Join [
        ",",
        [!Ref PublicSubnet, !If [DoAz2, !Ref PublicSubnet2, !Ref "AWS::NoValue"], !If [DoAz3, !Ref
PublicSubnet3, !Ref "AWS::NoValue"]]
      ]
  PrivateSubnetIds:
    Description: Subnet IDs of the private subnets.
    Value:
      !Join [
        ",",
        [!Ref PrivateSubnet, !If [DoAz2, !Ref PrivateSubnet2, !Ref "AWS::NoValue"], !If [DoAz3, !Ref
PrivateSubnet3, !Ref "AWS::NoValue"]]
      ]
  PublicRouteTableId:
    Description: Public Route table ID
    Value: !Ref PublicRouteTable
  PrivateRouteTableIds:
    Description: Private Route table IDs
    Value:
      !Join [
        ",",
        [
          !Join ["=", [
            !Select [0, "Fn::GetAZs": !Ref "AWS::Region"],
            !Ref PrivateRouteTable
          ]],
          !If [DoAz2,
               !Join ["=", [!Select [1, "Fn::GetAZs": !Ref "AWS::Region"], !Ref PrivateRouteTable2]],
               !Ref "AWS::NoValue"
          ],
          !If [DoAz3,
               !Join ["=", [!Select [2, "Fn::GetAZs": !Ref "AWS::Region"], !Ref PrivateRouteTable3]],
               !Ref "AWS::NoValue"
        ]
      ]
    ]
```

### 4.4.6. Creating networking and load balancing components in AWS

You must configure networking and classic or network load balancing in Amazon Web Services (AWS) that your OpenShift Container Platform cluster can use.

You can use the provided CloudFormation template and a custom parameter file to create a stack of AWS resources. The stack represents the networking and load balancing components that your OpenShift Container Platform cluster requires. The template also creates a hosted zone and subnet tags.

You can run the template multiple times within a single Virtual Private Cloud (VPC).

**NOTE**

If you do not use the provided CloudFormation template to create your AWS infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

#### Prerequisites

- You configured an AWS account.
- You added your AWS keys and region to your local AWS profile by running **aws configure**.
- You generated the Ignition config files for your cluster.
- You created and configured a VPC and associated subnets in AWS.

#### Procedure

1. Obtain the hosted zone ID for the Route 53 base domain that you specified in the **install-config.yaml** file for your cluster. You can obtain details about your hosted zone by running the following command:

   ```
   $ aws route53 list-hosted-zones-by-name --dns-name <route53_domain> 1
   ```

   1 For the **<route53_domain>**, specify the Route 53 base domain that you used when you generated the **install-config.yaml** file for the cluster.

   **Example output**

   ```
   mycluster.example.com. False 100
   HOSTEDZONES 65F8F38E-2268-B835-E15C-AB55336FCBFA
   /hostedzone/Z21IXYZABCZ2A4 mycluster.example.com. 10
   ```

   In the example output, the hosted zone ID is **Z21IXYZABCZ2A4**.

2. Create a JSON file that contains the parameter values that the template requires:

```
[
  {
   "ParameterKey": "ClusterName", (1)
   "ParameterValue": "mycluster" (2)
  },
  {
   "ParameterKey": "InfrastructureName", (3)
   "ParameterValue": "mycluster-<random_string>" (4)
  },
  {
   "ParameterKey": "HostedZoneId", (5)
   "ParameterValue": "<random_string>" (6)
  },
  {
   "ParameterKey": "HostedZoneName", (7)
   "ParameterValue": "example.com" (8)
  },
  {
   "ParameterKey": "PublicSubnets", (9)
   "ParameterValue": "subnet-<random_string>" (10)
  },
  {
   "ParameterKey": "PrivateSubnets", (11)
   "ParameterValue": "subnet-<random_string>" (12)
  },
  {
   "ParameterKey": "VpcId", (13)
   "ParameterValue": "vpc-<random_string>" (14)
  }
]
```

(1) A short, representative cluster name to use for hostnames, etc.

(2) Specify the cluster name that you used when you generated the **install-config.yaml** file for the cluster.

(3) The name for your cluster infrastructure that is encoded in your Ignition config files for the cluster.

(4) Specify the infrastructure name that you extracted from the Ignition config file metadata, which has the format **<cluster-name>-<random-string>**.

(5) The Route 53 public zone ID to register the targets with.

(6) Specify the Route 53 public zone ID, which as a format similar to **Z21IXYZABCZ2A4**. You can obtain this value from the AWS console.

(7) The Route 53 zone to register the targets with.

(8) Specify the Route 53 base domain that you used when you generated the **install-config.yaml** file for the cluster. Do not include the trailing period (.) that is displayed in the AWS console.

9 The public subnets that you created for your VPC.

10 Specify the **PublicSubnetIds** value from the output of the CloudFormation template for the VPC.

11 The private subnets that you created for your VPC.

12 Specify the **PrivateSubnetIds** value from the output of the CloudFormation template for the VPC.

13 The VPC that you created for the cluster.

14 Specify the **VpcId** value from the output of the CloudFormation template for the VPC.

3. Copy the template from the **CloudFormation template for the network and load balancers** section of this topic and save it as a YAML file on your computer. This template describes the networking and load balancing objects that your cluster requires.

IMPORTANT

If you are deploying your cluster to an AWS government or secret region, you must update the **InternalApiServerRecord** in the CloudFormation template to use **CNAME** records. Records of type **ALIAS** are not supported for AWS government regions.

4. Launch the CloudFormation template to create a stack of AWS resources that provide the networking and load balancing components:

IMPORTANT

You must enter the command on a single line.

```
$ aws cloudformation create-stack --stack-name <name> 1
     --template-body file://<template>.yaml 2
     --parameters file://<parameters>.json 3
     --capabilities CAPABILITY_NAMED_IAM 4
```

1 **<name>** is the name for the CloudFormation stack, such as **cluster-dns**. You need the name of this stack if you remove the cluster.

2 **<template>** is the relative path to and name of the CloudFormation template YAML file that you saved.

3 **<parameters>** is the relative path to and name of the CloudFormation parameters JSON file.

4 You must explicitly declare the **CAPABILITY_NAMED_IAM** capability because the provided template creates some **AWS::IAM::Role** resources.

### Example output

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-dns/cd3e5de0-2fd4-11eb-
5cf0-12be5c33a183
```

5. Confirm that the template components exist:

```
$ aws cloudformation describe-stacks --stack-name <name>
```

After the **StackStatus** displays **CREATE_COMPLETE**, the output displays values for the following parameters. You must provide these parameter values to the other CloudFormation templates that you run to create your cluster:

| | |
|---|---|
| **PrivateHostedZoneId** | Hosted zone ID for the private DNS. |
| **ExternalApiLoadBalancerName** | Full name of the external API load balancer. |
| **InternalApiLoadBalancerName** | Full name of the internal API load balancer. |
| **ApiServerDnsName** | Full hostname of the API server. |
| **RegisterNlbIpTargetsLambda** | Lambda ARN useful to help register/deregister IP targets for these load balancers. |
| **ExternalApiTargetGroupArn** | ARN of external API target group. |
| **InternalApiTargetGroupArn** | ARN of internal API target group. |
| **InternalServiceTargetGroupArn** | ARN of internal service target group. |

### 4.4.6.1. CloudFormation template for the network and load balancers

You can use the following CloudFormation template to deploy the networking objects and load balancers that you need for your OpenShift Container Platform cluster.

**Example 4.26. CloudFormation template for the network and load balancers**

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Network Elements (Route53 & LBs)

Parameters:
  ClusterName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Cluster name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, representative cluster name to use for host names and other identifying
names.
    Type: String
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, unique cluster ID used to tag cloud resources and identify items owned or
used by the cluster.
    Type: String
  HostedZoneId:
    Description: The Route53 public zone ID to register the targets with, such as
Z21IXYZABCZ2A4.
    Type: String
  HostedZoneName:
    Description: The Route53 zone to register the targets with, such as example.com. Omit the
trailing period.
    Type: String
    Default: "example.com"
  PublicSubnets:
    Description: The internet-facing subnets.
    Type: List<AWS::EC2::Subnet::Id>
  PrivateSubnets:
    Description: The internal subnets.
    Type: List<AWS::EC2::Subnet::Id>
  VpcId:
    Description: The VPC-scoped resources will belong to this VPC.
    Type: AWS::EC2::VPC::Id

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - ClusterName
      - InfrastructureName
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcId
      - PublicSubnets
```

```
        - PrivateSubnets
      - Label:
          default: "DNS"
        Parameters:
        - HostedZoneName
        - HostedZoneId
      ParameterLabels:
        ClusterName:
          default: "Cluster Name"
        InfrastructureName:
          default: "Infrastructure Name"
        VpcId:
          default: "VPC ID"
        PublicSubnets:
          default: "Public Subnets"
        PrivateSubnets:
          default: "Private Subnets"
        HostedZoneName:
          default: "Public Hosted Zone Name"
        HostedZoneId:
          default: "Public Hosted Zone ID"

Resources:
  ExtApiElb:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Name: !Join ["-", [!Ref InfrastructureName, "ext"]]
      IpAddressType: ipv4
      Subnets: !Ref PublicSubnets
      Type: network

  IntApiElb:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Name: !Join ["-", [!Ref InfrastructureName, "int"]]
      Scheme: internal
      IpAddressType: ipv4
      Subnets: !Ref PrivateSubnets
      Type: network

  IntDns:
    Type: "AWS::Route53::HostedZone"
    Properties:
      HostedZoneConfig:
        Comment: "Managed by CloudFormation"
      Name: !Join [".", [!Ref ClusterName, !Ref HostedZoneName]]
      HostedZoneTags:
      - Key: Name
        Value: !Join ["-", [!Ref InfrastructureName, "int"]]
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "owned"
      VPCs:
      - VPCId: !Ref VpcId
        VPCRegion: !Ref "AWS::Region"

  ExternalApiServerRecord:
    Type: AWS::Route53::RecordSetGroup
    Properties:
      Comment: Alias record for the API server
      HostedZoneId: !Ref HostedZoneId
      RecordSets:
      - Name:
          !Join [
            ".",
            ["api", !Ref ClusterName, !Join ["", [!Ref HostedZoneName, "."]]],
          ]
        Type: A
        AliasTarget:
          HostedZoneId: !GetAtt ExtApiElb.CanonicalHostedZoneID
          DNSName: !GetAtt ExtApiElb.DNSName

  InternalApiServerRecord:
    Type: AWS::Route53::RecordSetGroup
    Properties:
      Comment: Alias record for the API server
      HostedZoneId: !Ref IntDns
      RecordSets:
      - Name:
          !Join [
            ".",
            ["api", !Ref ClusterName, !Join ["", [!Ref HostedZoneName, "."]]],
          ]
        Type: A
        AliasTarget:
          HostedZoneId: !GetAtt IntApiElb.CanonicalHostedZoneID
          DNSName: !GetAtt IntApiElb.DNSName
      - Name:
          !Join [
            ".",
            ["api-int", !Ref ClusterName, !Join ["", [!Ref HostedZoneName, "."]]],
          ]
        Type: A
        AliasTarget:
          HostedZoneId: !GetAtt IntApiElb.CanonicalHostedZoneID
          DNSName: !GetAtt IntApiElb.DNSName

  ExternalApiListener:
    Type: AWS::ElasticLoadBalancingV2::Listener
    Properties:
      DefaultActions:
      - Type: forward
        TargetGroupArn:
          Ref: ExternalApiTargetGroup
      LoadBalancerArn:
        Ref: ExtApiElb
      Port: 6443
      Protocol: TCP

  ExternalApiTargetGroup:
    Type: AWS::ElasticLoadBalancingV2::TargetGroup
    Properties:
      HealthCheckIntervalSeconds: 10
      HealthCheckPath: "/readyz"
      HealthCheckPort: 6443
      HealthCheckProtocol: HTTPS
      HealthyThresholdCount: 2
      UnhealthyThresholdCount: 2
      Port: 6443
      Protocol: TCP
      TargetType: ip
      VpcId:
        Ref: VpcId
      TargetGroupAttributes:
      - Key: deregistration_delay.timeout_seconds
        Value: 60

  InternalApiListener:
    Type: AWS::ElasticLoadBalancingV2::Listener
    Properties:
      DefaultActions:
      - Type: forward
        TargetGroupArn:
          Ref: InternalApiTargetGroup
      LoadBalancerArn:
        Ref: IntApiElb
      Port: 6443
      Protocol: TCP

  InternalApiTargetGroup:
    Type: AWS::ElasticLoadBalancingV2::TargetGroup
    Properties:
      HealthCheckIntervalSeconds: 10
      HealthCheckPath: "/readyz"
      HealthCheckPort: 6443
      HealthCheckProtocol: HTTPS
      HealthyThresholdCount: 2
      UnhealthyThresholdCount: 2
      Port: 6443
      Protocol: TCP
      TargetType: ip
      VpcId:
        Ref: VpcId
      TargetGroupAttributes:
      - Key: deregistration_delay.timeout_seconds
        Value: 60

  InternalServiceInternalListener:
    Type: AWS::ElasticLoadBalancingV2::Listener
    Properties:
      DefaultActions:
      - Type: forward
        TargetGroupArn:
          Ref: InternalServiceTargetGroup
      LoadBalancerArn:
        Ref: IntApiElb
      Port: 22623
      Protocol: TCP
```

```
InternalServiceTargetGroup:
  Type: AWS::ElasticLoadBalancingV2::TargetGroup
  Properties:
    HealthCheckIntervalSeconds: 10
    HealthCheckPath: "/healthz"
    HealthCheckPort: 22623
    HealthCheckProtocol: HTTPS
    HealthyThresholdCount: 2
    UnhealthyThresholdCount: 2
    Port: 22623
    Protocol: TCP
    TargetType: ip
    VpcId:
      Ref: VpcId
    TargetGroupAttributes:
    - Key: deregistration_delay.timeout_seconds
      Value: 60

RegisterTargetLambdaIamRole:
  Type: AWS::IAM::Role
  Properties:
    RoleName: !Join ["-", [!Ref InfrastructureName, "nlb", "lambda", "role"]]
    AssumeRolePolicyDocument:
      Version: "2012-10-17"
      Statement:
      - Effect: "Allow"
        Principal:
          Service:
          - "lambda.amazonaws.com"
        Action:
        - "sts:AssumeRole"
    Path: "/"
    Policies:
    - PolicyName: !Join ["-", [!Ref InfrastructureName, "master", "policy"]]
      PolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Action:
            [
              "elasticloadbalancing:RegisterTargets",
              "elasticloadbalancing:DeregisterTargets",
            ]
          Resource: !Ref InternalApiTargetGroup
        - Effect: "Allow"
          Action:
            [
              "elasticloadbalancing:RegisterTargets",
              "elasticloadbalancing:DeregisterTargets",
            ]
          Resource: !Ref InternalServiceTargetGroup
        - Effect: "Allow"
          Action:
            [
              "elasticloadbalancing:RegisterTargets",
              "elasticloadbalancing:DeregisterTargets",
          ]
        Resource: !Ref ExternalApiTargetGroup

  RegisterNlbIpTargets:
    Type: "AWS::Lambda::Function"
    Properties:
      Handler: "index.handler"
      Role:
        Fn::GetAtt:
        - "RegisterTargetLambdaIamRole"
        - "Arn"
      Code:
        ZipFile: |
          import json
          import boto3
          import cfnresponse
          def handler(event, context):
            elb = boto3.client('elbv2')
            if event['RequestType'] == 'Delete':
              elb.deregister_targets(TargetGroupArn=event['ResourceProperties']
['TargetArn'],Targets=[{'Id': event['ResourceProperties']['TargetIp']}])
            elif event['RequestType'] == 'Create':
              elb.register_targets(TargetGroupArn=event['ResourceProperties']['TargetArn'],Targets=
[{'Id': event['ResourceProperties']['TargetIp']}])
            responseData = {}
            cfnresponse.send(event, context, cfnresponse.SUCCESS, responseData,
event['ResourceProperties']['TargetArn']+event['ResourceProperties']['TargetIp'])
      Runtime: "python3.11"
      Timeout: 120

  RegisterSubnetTagsLambdaIamRole:
    Type: AWS::IAM::Role
    Properties:
      RoleName: !Join ["-", [!Ref InfrastructureName, "subnet-tags-lambda-role"]]
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Principal:
            Service:
            - "lambda.amazonaws.com"
          Action:
          - "sts:AssumeRole"
      Path: "/"
      Policies:
      - PolicyName: !Join ["-", [!Ref InfrastructureName, "subnet-tagging-policy"]]
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
          - Effect: "Allow"
            Action:
              [
                "ec2:DeleteTags",
                "ec2:CreateTags"
              ]
            Resource: "arn:aws:ec2:*:*:subnet/*"
          - Effect: "Allow"
            Action:
              [
                "ec2:DescribeSubnets",
                "ec2:DescribeTags"
              ]
            Resource: "*"

  RegisterSubnetTags:
    Type: "AWS::Lambda::Function"
    Properties:
      Handler: "index.handler"
      Role:
        Fn::GetAtt:
        - "RegisterSubnetTagsLambdaIamRole"
        - "Arn"
      Code:
        ZipFile: |
          import json
          import boto3
          import cfnresponse
          def handler(event, context):
            ec2_client = boto3.client('ec2')
            if event['RequestType'] == 'Delete':
              for subnet_id in event['ResourceProperties']['Subnets']:
                ec2_client.delete_tags(Resources=[subnet_id], Tags=[{'Key': 'kubernetes.io/cluster/' +
event['ResourceProperties']['InfrastructureName']}]);
            elif event['RequestType'] == 'Create':
              for subnet_id in event['ResourceProperties']['Subnets']:
                ec2_client.create_tags(Resources=[subnet_id], Tags=[{'Key': 'kubernetes.io/cluster/' +
event['ResourceProperties']['InfrastructureName'], 'Value': 'shared'}]);
            responseData = {}
            cfnresponse.send(event, context, cfnresponse.SUCCESS, responseData,
event['ResourceProperties']['InfrastructureName']+event['ResourceProperties']['Subnets'][0])
      Runtime: "python3.11"
      Timeout: 120

  RegisterPublicSubnetTags:
    Type: Custom::SubnetRegister
    Properties:
      ServiceToken: !GetAtt RegisterSubnetTags.Arn
      InfrastructureName: !Ref InfrastructureName
      Subnets: !Ref PublicSubnets

  RegisterPrivateSubnetTags:
    Type: Custom::SubnetRegister
    Properties:
      ServiceToken: !GetAtt RegisterSubnetTags.Arn
      InfrastructureName: !Ref InfrastructureName
      Subnets: !Ref PrivateSubnets

Outputs:
  PrivateHostedZoneId:
    Description: Hosted zone ID for the private DNS, which is required for private records.
    Value: !Ref IntDns
  ExternalApiLoadBalancerName:
      Description: Full name of the external API load balancer.
      Value: !GetAtt ExtApiElb.LoadBalancerFullName
    InternalApiLoadBalancerName:
      Description: Full name of the internal API load balancer.
      Value: !GetAtt IntApiElb.LoadBalancerFullName
    ApiServerDnsName:
      Description: Full hostname of the API server, which is required for the Ignition config files.
      Value: !Join [".", ["api-int", !Ref ClusterName, !Ref HostedZoneName]]
    RegisterNlbIpTargetsLambda:
      Description: Lambda ARN useful to help register or deregister IP targets for these load
balancers.
      Value: !GetAtt RegisterNlbIpTargets.Arn
    ExternalApiTargetGroupArn:
      Description: ARN of the external API target group.
      Value: !Ref ExternalApiTargetGroup
    InternalApiTargetGroupArn:
      Description: ARN of the internal API target group.
      Value: !Ref InternalApiTargetGroup
    InternalServiceTargetGroupArn:
      Description: ARN of the internal service target group.
      Value: !Ref InternalServiceTargetGroup
```

IMPORTANT

If you are deploying your cluster to an AWS government or secret region, you must update the **InternalApiServerRecord** to use **CNAME** records. Records of type **ALIAS** are not supported for AWS government regions. For example:

```
Type: CNAME
TTL: 10
ResourceRecords:
- !GetAtt IntApiElb.DNSName
```

**Additional resources**

- Listing public hosted zones(AWS documentation)

### 4.4.7. Creating security group and roles in AWS

You must create security groups and roles in Amazon Web Services (AWS) for your OpenShift Container Platform cluster to use.

You can use the provided CloudFormation template and a custom parameter file to create a stack of AWS resources. The stack represents the security groups and roles that your OpenShift Container Platform cluster requires.

NOTE

If you do not use the provided CloudFormation template to create your AWS infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

### Prerequisites

- You configured an AWS account.
- You added your AWS keys and region to your local AWS profile by running **aws configure**.
- You generated the Ignition config files for your cluster.
- You created and configured a VPC and associated subnets in AWS.

### Procedure

1. Create a JSON file that contains the parameter values that the template requires:

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
    "ParameterKey": "VpcCidr", 3
    "ParameterValue": "10.0.0.0/16" 4
  },
  {
    "ParameterKey": "PrivateSubnets", 5
    "ParameterValue": "subnet-<random_string>" 6
  },
  {
    "ParameterKey": "VpcId", 7
    "ParameterValue": "vpc-<random_string>" 8
  }
]
```

1 The name for your cluster infrastructure that is encoded in your Ignition config files for the cluster.

2 Specify the infrastructure name that you extracted from the Ignition config file metadata, which has the format **<cluster-name>-<random-string>**.

3 The CIDR block for the VPC.

4 Specify the CIDR block parameter that you used for the VPC that you defined in the form **x.x.x.x/16-24**.

5 The private subnets that you created for your VPC.

6 Specify the **PrivateSubnetIds** value from the output of the CloudFormation template for the VPC.

7 The VPC that you created for the cluster.

8 Specify the **VpcId** value from the output of the CloudFormation template for the VPC.

2. [illegible]

2. Copy the template from the **CloudFormation template for security objects**section of this topic and save it as a YAML file on your computer. This template describes the security groups and roles that your cluster requires.

3. Launch the CloudFormation template to create a stack of AWS resources that represent the security groups and roles:

   IMPORTANT

   You must enter the command on a single line.

```
$ aws cloudformation create-stack --stack-name <name> 1
    --template-body file://<template>.yaml 2
    --parameters file://<parameters>.json 3
    --capabilities CAPABILITY_NAMED_IAM 4
```

   1 **<name>** is the name for the CloudFormation stack, such as **cluster-sec**. You need the name of this stack if you remove the cluster.

   2 **<template>** is the relative path to and name of the CloudFormation template YAML file that you saved.

   3 **<parameters>** is the relative path to and name of the CloudFormation parameters JSON file.

   4 You must explicitly declare the **CAPABILITY_NAMED_IAM** capability because the provided template creates some **AWS::IAM::Role** and **AWS::IAM::InstanceProfile** resources.

   **Example output**

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-sec/03bd4210-2ed7-11eb-
6d7a-13fc0b61e9db
```

4. Confirm that the template components exist:

```
$ aws cloudformation describe-stacks --stack-name <name>
```

   After the **StackStatus** displays **CREATE_COMPLETE**, the output displays values for the following parameters. You must provide these parameter values to the other CloudFormation templates that you run to create your cluster:

| | |
|---|---|
| **MasterSecurityGroupId** | Master Security Group ID |
| **WorkerSecurityGroupId** | Worker Security Group ID |

| | |
|---|---|
| **MasterInstanceProfile** | Master IAM Instance Profile |
| **WorkerInstanceProfile** | Worker IAM Instance Profile |

### 4.4.7.1. CloudFormation template for security objects

You can use the following CloudFormation template to deploy the security objects that you need for your OpenShift Container Platform cluster.

**Example 4.27. CloudFormation template for security objects**

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Security Elements (Security Groups & IAM)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, unique cluster ID used to tag cloud resources and identify items owned or
used by the cluster.
    Type: String
  VpcCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-
4][0-9]|25[0-5])(\/(1[6-9]|2[0-4]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/16-24.
    Default: 10.0.0.0/16
    Description: CIDR block for VPC.
    Type: String
  VpcId:
    Description: The VPC-scoped resources will belong to this VPC.
    Type: AWS::EC2::VPC::Id
  PrivateSubnets:
    Description: The internal subnets.
    Type: List<AWS::EC2::Subnet::Id>

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - InfrastructureName
    - Label:
        default: "Network Configuration"
      Parameters:
```

```
      - VpcId
      - VpcCidr
      - PrivateSubnets
    ParameterLabels:
      InfrastructureName:
        default: "Infrastructure Name"
      VpcId:
        default: "VPC ID"
      VpcCidr:
        default: "VPC CIDR"
      PrivateSubnets:
        default: "Private Subnets"

Resources:
  MasterSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Cluster Master Security Group
      SecurityGroupIngress:
      - IpProtocol: icmp
        FromPort: 0
        ToPort: 0
        CidrIp: !Ref VpcCidr
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: !Ref VpcCidr
      - IpProtocol: tcp
        ToPort: 6443
        FromPort: 6443
        CidrIp: !Ref VpcCidr
      - IpProtocol: tcp
        FromPort: 22623
        ToPort: 22623
        CidrIp: !Ref VpcCidr
      VpcId: !Ref VpcId

  WorkerSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Cluster Worker Security Group
      SecurityGroupIngress:
      - IpProtocol: icmp
        FromPort: 0
        ToPort: 0
        CidrIp: !Ref VpcCidr
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: !Ref VpcCidr
      VpcId: !Ref VpcId

  MasterIngressEtcd:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: etcd
      FromPort: 2379
      ToPort: 2380
      IpProtocol: tcp

  MasterIngressVxlan:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Vxlan packets
      FromPort: 4789
      ToPort: 4789
      IpProtocol: udp

  MasterIngressWorkerVxlan:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Vxlan packets
      FromPort: 4789
      ToPort: 4789
      IpProtocol: udp

  MasterIngressGeneve:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Geneve packets
      FromPort: 6081
      ToPort: 6081
      IpProtocol: udp

  MasterIngressWorkerGeneve:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Geneve packets
      FromPort: 6081
      ToPort: 6081
      IpProtocol: udp

  MasterIngressIpsecIke:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: IPsec IKE packets
      FromPort: 500
      ToPort: 500
      IpProtocol: udp
```

```
MasterIngressIpsecNat:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: IPsec NAT-T packets
    FromPort: 4500
    ToPort: 4500
    IpProtocol: udp

MasterIngressIpsecEsp:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: IPsec ESP packets
    IpProtocol: 50

MasterIngressWorkerIpsecIke:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: IPsec IKE packets
    FromPort: 500
    ToPort: 500
    IpProtocol: udp

MasterIngressWorkerIpsecNat:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: IPsec NAT-T packets
    FromPort: 4500
    ToPort: 4500
    IpProtocol: udp

MasterIngressWorkerIpsecEsp:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: IPsec ESP packets
    IpProtocol: 50

MasterIngressInternal:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Internal cluster communication
    FromPort: 9000
    ToPort: 9999
    IpProtocol: tcp
```

```
MasterIngressWorkerInternal:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Internal cluster communication
    FromPort: 9000
    ToPort: 9999
    IpProtocol: tcp

MasterIngressInternalUDP:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Internal cluster communication
    FromPort: 9000
    ToPort: 9999
    IpProtocol: udp

MasterIngressWorkerInternalUDP:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Internal cluster communication
    FromPort: 9000
    ToPort: 9999
    IpProtocol: udp

MasterIngressKube:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Kubernetes kubelet, scheduler and controller manager
    FromPort: 10250
    ToPort: 10259
    IpProtocol: tcp

MasterIngressWorkerKube:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Kubernetes kubelet, scheduler and controller manager
    FromPort: 10250
    ToPort: 10259
    IpProtocol: tcp

MasterIngressIngressServices:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Kubernetes ingress services
```

```
      FromPort: 30000
      ToPort: 32767
      IpProtocol: tcp

  MasterIngressWorkerIngressServices:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Kubernetes ingress services
      FromPort: 30000
      ToPort: 32767
      IpProtocol: tcp

  MasterIngressIngressServicesUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Kubernetes ingress services
      FromPort: 30000
      ToPort: 32767
      IpProtocol: udp

  MasterIngressWorkerIngressServicesUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Kubernetes ingress services
      FromPort: 30000
      ToPort: 32767
      IpProtocol: udp

  WorkerIngressVxlan:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Vxlan packets
      FromPort: 4789
      ToPort: 4789
      IpProtocol: udp

  WorkerIngressMasterVxlan:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Vxlan packets
      FromPort: 4789
      ToPort: 4789
      IpProtocol: udp

  WorkerIngressGeneve:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Geneve packets
      FromPort: 6081
      ToPort: 6081
      IpProtocol: udp

  WorkerIngressMasterGeneve:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Geneve packets
      FromPort: 6081
      ToPort: 6081
      IpProtocol: udp

  WorkerIngressIpsecIke:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: IPsec IKE packets
      FromPort: 500
      ToPort: 500
      IpProtocol: udp

  WorkerIngressIpsecNat:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: IPsec NAT-T packets
      FromPort: 4500
      ToPort: 4500
      IpProtocol: udp

  WorkerIngressIpsecEsp:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: IPsec ESP packets
      IpProtocol: 50

  WorkerIngressMasterIpsecIke:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: IPsec IKE packets
      FromPort: 500
      ToPort: 500
      IpProtocol: udp
```

```
WorkerIngressMasterIpsecNat:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: IPsec NAT-T packets
    FromPort: 4500
    ToPort: 4500
    IpProtocol: udp

WorkerIngressMasterIpsecEsp:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: IPsec ESP packets
    IpProtocol: 50

WorkerIngressInternal:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Internal cluster communication
    FromPort: 9000
    ToPort: 9999
    IpProtocol: tcp

WorkerIngressMasterInternal:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Internal cluster communication
    FromPort: 9000
    ToPort: 9999
    IpProtocol: tcp

WorkerIngressInternalUDP:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Internal cluster communication
    FromPort: 9000
    ToPort: 9999
    IpProtocol: udp

WorkerIngressMasterInternalUDP:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Internal cluster communication
    FromPort: 9000
    ToPort: 9999
```

```
      IpProtocol: udp

  WorkerIngressKube:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Kubernetes secure kubelet port
      FromPort: 10250
      ToPort: 10250
      IpProtocol: tcp

  WorkerIngressWorkerKube:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Internal Kubernetes communication
      FromPort: 10250
      ToPort: 10250
      IpProtocol: tcp

  WorkerIngressIngressServices:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Kubernetes ingress services
      FromPort: 30000
      ToPort: 32767
      IpProtocol: tcp

  WorkerIngressMasterIngressServices:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Kubernetes ingress services
      FromPort: 30000
      ToPort: 32767
      IpProtocol: tcp

  WorkerIngressIngressServicesUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Kubernetes ingress services
      FromPort: 30000
      ToPort: 32767
      IpProtocol: udp

  WorkerIngressMasterIngressServicesUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
```

```
        SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
        Description: Kubernetes ingress services
        FromPort: 30000
        ToPort: 32767
        IpProtocol: udp

    MasterIamRole:
      Type: AWS::IAM::Role
      Properties:
        AssumeRolePolicyDocument:
          Version: "2012-10-17"
          Statement:
          - Effect: "Allow"
            Principal:
              Service:
              - "ec2.amazonaws.com"
            Action:
            - "sts:AssumeRole"
        Policies:
        - PolicyName: !Join ["-", [!Ref InfrastructureName, "master", "policy"]]
          PolicyDocument:
            Version: "2012-10-17"
            Statement:
            - Effect: "Allow"
              Action:
              - "ec2:AttachVolume"
              - "ec2:AuthorizeSecurityGroupIngress"
              - "ec2:CreateSecurityGroup"
              - "ec2:CreateTags"
              - "ec2:CreateVolume"
              - "ec2:DeleteSecurityGroup"
              - "ec2:DeleteVolume"
              - "ec2:Describe*"
              - "ec2:DetachVolume"
              - "ec2:ModifyInstanceAttribute"
              - "ec2:ModifyVolume"
              - "ec2:RevokeSecurityGroupIngress"
              - "elasticloadbalancing:AddTags"
              - "elasticloadbalancing:AttachLoadBalancerToSubnets"
              - "elasticloadbalancing:ApplySecurityGroupsToLoadBalancer"
              - "elasticloadbalancing:CreateListener"
              - "elasticloadbalancing:CreateLoadBalancer"
              - "elasticloadbalancing:CreateLoadBalancerPolicy"
              - "elasticloadbalancing:CreateLoadBalancerListeners"
              - "elasticloadbalancing:CreateTargetGroup"
              - "elasticloadbalancing:ConfigureHealthCheck"
              - "elasticloadbalancing:DeleteListener"
              - "elasticloadbalancing:DeleteLoadBalancer"
              - "elasticloadbalancing:DeleteLoadBalancerListeners"
              - "elasticloadbalancing:DeleteTargetGroup"
              - "elasticloadbalancing:DeregisterInstancesFromLoadBalancer"
              - "elasticloadbalancing:DeregisterTargets"
              - "elasticloadbalancing:Describe*"
              - "elasticloadbalancing:DetachLoadBalancerFromSubnets"
              - "elasticloadbalancing:ModifyListener"
              - "elasticloadbalancing:ModifyLoadBalancerAttributes"
          - "elasticloadbalancing:ModifyTargetGroup"
          - "elasticloadbalancing:ModifyTargetGroupAttributes"
          - "elasticloadbalancing:RegisterInstancesWithLoadBalancer"
          - "elasticloadbalancing:RegisterTargets"
          - "elasticloadbalancing:SetLoadBalancerPoliciesForBackendServer"
          - "elasticloadbalancing:SetLoadBalancerPoliciesOfListener"
          - "kms:DescribeKey"
          Resource: "*"

  MasterInstanceProfile:
    Type: "AWS::IAM::InstanceProfile"
    Properties:
      Roles:
      - Ref: "MasterIamRole"

  WorkerIamRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Principal:
            Service:
            - "ec2.amazonaws.com"
          Action:
          - "sts:AssumeRole"
      Policies:
      - PolicyName: !Join ["-", [!Ref InfrastructureName, "worker", "policy"]]
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
          - Effect: "Allow"
            Action:
            - "ec2:DescribeInstances"
            - "ec2:DescribeRegions"
            Resource: "*"

  WorkerInstanceProfile:
    Type: "AWS::IAM::InstanceProfile"
    Properties:
      Roles:
      - Ref: "WorkerIamRole"

Outputs:
  MasterSecurityGroupId:
    Description: Master Security Group ID
    Value: !GetAtt MasterSecurityGroup.GroupId

  WorkerSecurityGroupId:
    Description: Worker Security Group ID
    Value: !GetAtt WorkerSecurityGroup.GroupId

  MasterInstanceProfile:
    Description: Master IAM Instance Profile
    Value: !Ref MasterInstanceProfile
```

```
WorkerInstanceProfile:
  Description: Worker IAM Instance Profile
  Value: !Ref WorkerInstanceProfile
```

### 4.4.8. Accessing RHCOS AMIs with stream metadata

In OpenShift Container Platform, *stream metadata* provides standardized metadata about RHCOS in the JSON format and injects the metadata into the cluster. Stream metadata is a stable format that supports multiple architectures and is intended to be self-documenting for maintaining automation.

You can use the **coreos print-stream-json** sub-command of **openshift-install** to access information about the boot images in the stream metadata format. This command provides a method for printing stream metadata in a scriptable, machine-readable format.

For user-provisioned installations, the **openshift-install** binary contains references to the version of RHCOS boot images that are tested for use with OpenShift Container Platform, such as the AWS AMI.

#### Procedure

To parse the stream metadata, use one of the following methods:

- From a Go program, use the official **stream-metadata-go** library at https://github.com/coreos/stream-metadata-go. You can also view example code in the library.
- From another programming language, such as Python or Ruby, use the JSON library of your preferred programming language.
- From a command-line utility that handles JSON data, such as **jq**:
  - Print the current **x86_64** or **aarch64** AMI for an AWS region, such as **us-west-1**:

    **For x86_64**

    ```
    $ openshift-install coreos print-stream-json | jq -r
    '.architectures.x86_64.images.aws.regions["us-west-1"].image'
    ```

    **Example output**

    ```
    ami-0d3e625f84626bbda
    ```

    **For aarch64**

    ```
    $ openshift-install coreos print-stream-json | jq -r
    '.architectures.aarch64.images.aws.regions["us-west-1"].image'
    ```

    **Example output**

    ```
    ami-0af1d3b7fa5be2131
    ```

    The output of this command is the AWS AMI ID for your designated architecture and the **us-west-1** region. The AMI must belong to the same region as the cluster.

### 4.4.9. RHCOS AMIs for the AWS infrastructure

Red Hat provides Red Hat Enterprise Linux CoreOS (RHCOS) AMIs that are valid for the various AWS regions and instance architectures that you can manually specify for your OpenShift Container Platform nodes.

**NOTE**

By importing your own AMI, you can also install to regions that do not have a published RHCOS AMI.

Table 4.5. x86_64 RHCOS AMIs

| AWS zone | AWS AMI |
|---|---|
| **af-south-1** | **ami-0163621ea085783d8** |
| **ap-east-1** | **ami-033db3b659641feea** |
| **ap-northeast-1** | **ami-0baf16f8c6bd53f63** |
| **ap-northeast-2** | **ami-01a92be7f419359cc** |
| **ap-northeast-3** | **ami-0f16895f6f50e656e** |
| **ap-south-1** | **ami-0272be2f6528576f3** |
| **ap-south-2** | **ami-0311119df2ebc0bbc** |
| **ap-southeast-1** | **ami-0637678b0ad540477** |
| **ap-southeast-2** | **ami-0b67b492c091ac746** |
| **ap-southeast-3** | **ami-0a9e63bf1df36a936** |
| **ap-southeast-4** | **ami-0f153b95673592039** |
| **ap-southeast-5** | **ami-025944207bb28ae8f** |
| **ap-southeast-7** | **ami-0b5e29c2ae4aaa66d** |
| **ca-central-1** | **ami-03263f0cfdfa8bbdb** |
| **ca-west-1** | **ami-0254620c2dc7dcacc** |
| **eu-central-1** | **ami-0a0a87862b24395d8** |

| AWS zone | AWS AMI |
|---|---|
| eu-central-2 | ami-015c8ca32f5d8300a |
| eu-north-1 | ami-0c4404a6ae5921a1b |
| eu-south-1 | ami-0e0724943dd915bb2 |
| eu-south-2 | ami-0e6cac787a21b221d |
| eu-west-1 | ami-0355d4c968e466965 |
| eu-west-2 | ami-0e079f8742280b034 |
| eu-west-3 | ami-06702aad076acda7b |
| il-central-1 | ami-0094ac2722d41c18c |
| me-central-1 | ami-03680a3dcecfbe79d |
| me-south-1 | ami-04e14a3c4be812ac7 |
| mx-central-1 | ami-0eac1c8d4154a417f |
| sa-east-1 | ami-07abd63bb465f89b6 |
| us-east-1 | ami-0e8fd9094e487d1ff |
| us-east-2 | ami-0d4a7b7677c0c883f |
| us-gov-east-1 | ami-0b67e7ffd11a17645 |
| us-gov-west-1 | ami-041e18a76f42c752c |
| us-west-1 | ami-0167f257577d883cc |
| us-west-2 | ami-0b29d41f2ed6b8c94 |

Table 4.6. aarch64 RHCOS AMIs

| AWS zone | AWS AMI |
|---|---|
| af-south-1 | ami-009231bfc2490c6f9 |
| ap-east-1 | ami-0a23fad8fb25f5bb7 |

| AWS zone | AWS AMI |
|---|---|
| **ap-northeast-1** | **ami-0754a269f165f227c** |
| **ap-northeast-2** | **ami-0d81f596571ce27d8** |
| **ap-northeast-3** | **ami-01eb2f8b176229523** |
| **ap-south-1** | **ami-0be3b34441044e437** |
| **ap-south-2** | **ami-02a86359661950bb0** |
| **ap-southeast-1** | **ami-0c70d35c9b5b190be** |
| **ap-southeast-2** | **ami-0310f2acbeca636ed** |
| **ap-southeast-3** | **ami-04db4055063382442** |
| **ap-southeast-4** | **ami-0e2e40cc31633d7d6** |
| **ap-southeast-5** | **ami-0cf0c9ee9f324f763** |
| **ap-southeast-7** | **ami-04cdafcdc85bf9040** |
| **ca-central-1** | **ami-0aee20271a9396925** |
| **ca-west-1** | **ami-03ca778cd4265aad9** |
| **eu-central-1** | **ami-0281dddee0884d9f0** |
| **eu-central-2** | **ami-00fc4e5e3926530af** |
| **eu-north-1** | **ami-0696b5b31d326ccc6** |
| **eu-south-1** | **ami-04090792b7bdb9e0f** |
| **eu-south-2** | **ami-0d45a2586055d5daa** |
| **eu-west-1** | **ami-02f08479c3613ed0e** |
| **eu-west-2** | **ami-0ef2fc25f02a2d475** |
| **eu-west-3** | **ami-0ba5d0a0e5d796da8** |
| **il-central-1** | **ami-0e5b8f3b8e71961e7** |

| AWS zone | AWS AMI |
|---|---|
| **me-central-1** | **ami-0d13d6a91da2ba547** |
| **me-south-1** | **ami-0183dab9f96845e3f** |
| **mx-central-1** | **ami-072535d81a5de8e76** |
| **sa-east-1** | **ami-0977fa46dff272ba9** |
| **us-east-1** | **ami-083de3282c55be3f7** |
| **us-east-2** | **ami-02f30107e3441227b** |
| **us-gov-east-1** | **ami-0abaadf7322cfc258** |
| **us-gov-west-1** | **ami-0ca27128d77d732aa** |
| **us-west-1** | **ami-05a9426ae7c35740c** |
| **us-west-2** | **ami-0cd6ec50e0480b3a3** |

### 4.4.10. Creating the bootstrap node in AWS

You must create the bootstrap node in Amazon Web Services (AWS) to use during OpenShift Container Platform cluster initialization. You do this by:

- Providing a location to serve the **bootstrap.ign** Ignition config file to your cluster. This file is located in your installation directory. The provided CloudFormation Template assumes that the Ignition config files for your cluster are served from an S3 bucket. If you choose to serve the files from another location, you must modify the templates.
- Using the provided CloudFormation template and a custom parameter file to create a stack of AWS resources. The stack represents the bootstrap node that your OpenShift Container Platform installation requires.

NOTE

If you do not use the provided CloudFormation template to create your bootstrap node, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

**Prerequisites**

- You configured an AWS account.
- You added your AWS keys and region to your local AWS profile by running **aws configure**.

- You generated the Ignition config files for your cluster.
- You created and configured a VPC and associated subnets in AWS.
- You created and configured DNS, load balancers, and listeners in AWS.
- You created the security groups and roles required for your cluster in AWS.

## Procedure

1. Create the bucket by running the following command:

   ```
   $ aws s3 mb s3://<cluster-name>-infra (1)
   ```

   (1) **<cluster-name>-infra** is the bucket name. When creating the **install-config.yaml** file, replace **<cluster-name>** with the name specified for the cluster.

   You must use a presigned URL for your S3 bucket, instead of the **s3://** schema, if you are:

   - Deploying to a region that has endpoints that differ from the AWS SDK.
   - Deploying a proxy.
   - Providing your own custom endpoints.

2. Upload the **bootstrap.ign** Ignition config file to the bucket by running the following command:

   ```
   $ aws s3 cp <installation_directory>/bootstrap.ign s3://<cluster-name>-infra/bootstrap.ign (1)
   ```

   (1) For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

3. Verify that the file uploaded by running the following command:

   ```
   $ aws s3 ls s3://<cluster-name>-infra/
   ```

   **Example output**

   ```
   2019-04-03 16:15:16     314878 bootstrap.ign
   ```

   **NOTE**

   The bootstrap Ignition config file does contain secrets, like X.509 keys. The following steps provide basic security for the S3 bucket. To provide additional security, you can enable an S3 bucket policy to allow only certain users, such as the OpenShift IAM user, to access objects that the bucket contains. You can avoid S3 entirely and serve your bootstrap Ignition config file from any address that the bootstrap machine can reach.

4. Create a JSON file that contains the parameter values that the template requires:

   ```
   [
   ```

```
 {
  "ParameterKey": "InfrastructureName", 1
  "ParameterValue": "mycluster-<random_string>" 2
 },
 {
  "ParameterKey": "RhcosAmi", 3
  "ParameterValue": "ami-<random_string>" 4
 },
 {
  "ParameterKey": "AllowedBootstrapSshCidr", 5
  "ParameterValue": "0.0.0.0/0" 6
 },
 {
  "ParameterKey": "PublicSubnet", 7
  "ParameterValue": "subnet-<random_string>" 8
 },
 {
  "ParameterKey": "MasterSecurityGroupId", 9
  "ParameterValue": "sg-<random_string>" 10
 },
 {
  "ParameterKey": "VpcId", 11
  "ParameterValue": "vpc-<random_string>" 12
 },
 {
  "ParameterKey": "BootstrapIgnitionLocation", 13
  "ParameterValue": "s3://<bucket_name>/bootstrap.ign" 14
 },
 {
  "ParameterKey": "AutoRegisterELB", 15
  "ParameterValue": "yes" 16
 },
 {
  "ParameterKey": "RegisterNlbIpTargetsLambdaArn", 17
  "ParameterValue": "arn:aws:lambda:<aws_region>:<account_number>:function:
<dns_stack_name>-RegisterNlbIpTargets-<random_string>" 18
 },
 {
  "ParameterKey": "ExternalApiTargetGroupArn", 19
  "ParameterValue": "arn:aws:elasticloadbalancing:<aws_region>:
<account_number>:targetgroup/<dns_stack_name>-Exter-<random_string>" 20
 },
 {
  "ParameterKey": "InternalApiTargetGroupArn", 21
  "ParameterValue": "arn:aws:elasticloadbalancing:<aws_region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 22
 },
 {
  "ParameterKey": "InternalServiceTargetGroupArn", 23
  "ParameterValue": "arn:aws:elasticloadbalancing:<aws_region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 24
 }
]
```

1 The name for your cluster infrastructure that is encoded in your Ignition config files for the cluster.

2 Specify the infrastructure name that you extracted from the Ignition config file metadata, which has the format **<cluster-name>-<random-string>**.

3 Current Red Hat Enterprise Linux CoreOS (RHCOS) AMI to use for the bootstrap node based on your selected architecture.

4 Specify a valid **AWS::EC2::Image::Id** value.

5 CIDR block to allow SSH access to the bootstrap node.

6 Specify a CIDR block in the format **x.x.x.x/16-24**.

7 The public subnet that is associated with your VPC to launch the bootstrap node into.

8 Specify the **PublicSubnetIds** value from the output of the CloudFormation template for the VPC.

9 The master security group ID (for registering temporary rules)

10 Specify the **MasterSecurityGroupId** value from the output of the CloudFormation template for the security group and roles.

11 The VPC created resources will belong to.

12 Specify the **VpcId** value from the output of the CloudFormation template for the VPC.

13 Location to fetch bootstrap Ignition config file from.

14 Specify the S3 bucket and file name in the form **s3://<bucket_name>/bootstrap.ign**.

15 Whether or not to register a network load balancer (NLB).

16 Specify **yes** or **no**. If you specify **yes**, you must provide a Lambda Amazon Resource Name (ARN) value.

17 The ARN for NLB IP target registration lambda group.

18 Specify the **RegisterNlbIpTargetsLambda** value from the output of the CloudFormation template for DNS and load balancing. Use **arn:aws-us-gov** if deploying the cluster to an AWS GovCloud region.

19 The ARN for external API load balancer target group.

20 Specify the **ExternalApiTargetGroupArn** value from the output of the CloudFormation template for DNS and load balancing. Use **arn:aws-us-gov** if deploying the cluster to an AWS GovCloud region.

21 The ARN for internal API load balancer target group.

22 Specify the **InternalApiTargetGroupArn** value from the output of the CloudFormation template for DNS and load balancing. Use **arn:aws-us-gov** if deploying the cluster to an AWS GovCloud region.

23 The ARN for internal service load balancer target group.

24 Specify the **InternalServiceTargetGroupArn** value from the output of the CloudFormation template for DNS and load balancing. Use **arn:aws-us-gov** if deploying the cluster to an AWS GovCloud region.

5. Copy the template from the **CloudFormation template for the bootstrap machine**section of this topic and save it as a YAML file on your computer. This template describes the bootstrap machine that your cluster requires.

6. Optional: If you are deploying the cluster with a proxy, you must update the ignition in the template to add the **ignition.config.proxy** fields. Additionally, If you have added the Amazon EC2, Elastic Load Balancing, and S3 VPC endpoints to your VPC, you must add these endpoints to the **noProxy** field.

7. Launch the CloudFormation template to create a stack of AWS resources that represent the bootstrap node:

   IMPORTANT

   You must enter the command on a single line.

```
$ aws cloudformation create-stack --stack-name <name> 1
     --template-body file://<template>.yaml 2
     --parameters file://<parameters>.json 3
     --capabilities CAPABILITY_NAMED_IAM 4
```

1 **<name>** is the name for the CloudFormation stack, such as **cluster-bootstrap**. You need the name of this stack if you remove the cluster.

2 **<template>** is the relative path to and name of the CloudFormation template YAML file that you saved.

3 **<parameters>** is the relative path to and name of the CloudFormation parameters JSON file.

4 You must explicitly declare the **CAPABILITY_NAMED_IAM** capability because the provided template creates some **AWS::IAM::Role** and **AWS::IAM::InstanceProfile** resources.

### Example output

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-bootstrap/12944486-2add-
11eb-9dee-12dace8e3a83
```

8. Confirm that the template components exist:

```
$ aws cloudformation describe-stacks --stack-name <name>
```

After the **StackStatus** displays **CREATE_COMPLETE**, the output displays values for the following parameters. You must provide these parameter values to the other CloudFormation templates that you run to create your cluster:

| | |
|---|---|
| **BootstrapInstanceId** | The bootstrap Instance ID. |
| **BootstrapPublicIp** | The bootstrap node public IP address. |
| **BootstrapPrivateIp** | The bootstrap node private IP address. |

### 4.4.10.1. CloudFormation template for the bootstrap machine

You can use the following CloudFormation template to deploy the bootstrap machine that you need for your OpenShift Container Platform cluster.

**Example 4.28. CloudFormation template for the bootstrap machine**

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Bootstrap (EC2 Instance, Security Groups and IAM)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a maximum of 27 characters.
    Description: A short, unique cluster ID used to tag cloud resources and identify items owned or used by the cluster.
    Type: String
  RhcosAmi:
    Description: Current Red Hat Enterprise Linux CoreOS AMI to use for bootstrap.
    Type: AWS::EC2::Image::Id
  AllowedBootstrapSshCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])(\/([0-9]|1[0-9]|2[0-9]|3[0-2]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/0-32.
    Default: 0.0.0.0/0
    Description: CIDR block to allow SSH access to the bootstrap node.
    Type: String
  PublicSubnet:
    Description: The public subnet to launch the bootstrap node into.
    Type: AWS::EC2::Subnet::Id
  MasterSecurityGroupId:
    Description: The master security group ID for registering temporary rules.
    Type: AWS::EC2::SecurityGroup::Id
  VpcId:
    Description: The VPC-scoped resources will belong to this VPC.
    Type: AWS::EC2::VPC::Id
  BootstrapIgnitionLocation:
    Default: s3://my-s3-bucket/bootstrap.ign
    Description: Ignition config file location.
    Type: String
  AutoRegisterELB:
    Default: "yes"
    AllowedValues:
    - "yes"
    - "no"
    Description: Do you want to invoke NLB registration, which requires a Lambda ARN parameter?
    Type: String
  RegisterNlbIpTargetsLambdaArn:
    Description: ARN for NLB IP target registration lambda.
    Type: String
  ExternalApiTargetGroupArn:
    Description: ARN for external API load balancer target group.
    Type: String
  InternalApiTargetGroupArn:
    Description: ARN for internal API load balancer target group.
    Type: String
  InternalServiceTargetGroupArn:
    Description: ARN for internal service load balancer target group.
    Type: String
  BootstrapInstanceType:
    Description: Instance type for the bootstrap EC2 instance
    Default: "i3.large"
    Type: String

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - InfrastructureName
    - Label:
        default: "Host Information"
      Parameters:
      - RhcosAmi
      - BootstrapIgnitionLocation
      - MasterSecurityGroupId
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcId
      - AllowedBootstrapSshCidr
      - PublicSubnet
    - Label:
        default: "Load Balancer Automation"
      Parameters:
      - AutoRegisterELB
      - RegisterNlbIpTargetsLambdaArn
      - ExternalApiTargetGroupArn
      - InternalApiTargetGroupArn
      - InternalServiceTargetGroupArn
    ParameterLabels:
      InfrastructureName:
```

```
      default: "Infrastructure Name"
    VpcId:
      default: "VPC ID"
    AllowedBootstrapSshCidr:
      default: "Allowed SSH Source"
    PublicSubnet:
      default: "Public Subnet"
    RhcosAmi:
      default: "Red Hat Enterprise Linux CoreOS AMI ID"
    BootstrapIgnitionLocation:
      default: "Bootstrap Ignition Source"
    MasterSecurityGroupId:
      default: "Master Security Group ID"
    AutoRegisterELB:
      default: "Use Provided ELB Automation"

Conditions:
  DoRegistration: !Equals ["yes", !Ref AutoRegisterELB]

Resources:
  BootstrapIamRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Principal:
            Service:
            - "ec2.amazonaws.com"
          Action:
          - "sts:AssumeRole"
      Path: "/"
      Policies:
      - PolicyName: !Join ["-", [!Ref InfrastructureName, "bootstrap", "policy"]]
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
          - Effect: "Allow"
            Action: "ec2:Describe*"
            Resource: "*"
          - Effect: "Allow"
            Action: "ec2:AttachVolume"
            Resource: "*"
          - Effect: "Allow"
            Action: "ec2:DetachVolume"
            Resource: "*"
          - Effect: "Allow"
            Action: "s3:GetObject"
            Resource: "*"

  BootstrapInstanceProfile:
    Type: "AWS::IAM::InstanceProfile"
    Properties:
      Path: "/"
      Roles:
```

```
      - Ref: "BootstrapIamRole"

  BootstrapSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Cluster Bootstrap Security Group
      SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: !Ref AllowedBootstrapSshCidr
      - IpProtocol: tcp
        ToPort: 19531
        FromPort: 19531
        CidrIp: 0.0.0.0/0
      VpcId: !Ref VpcId

  BootstrapInstance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref RhcosAmi
      IamInstanceProfile: !Ref BootstrapInstanceProfile
      InstanceType: !Ref BootstrapInstanceType
      NetworkInterfaces:
      - AssociatePublicIpAddress: "true"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "BootstrapSecurityGroup"
        - !Ref "MasterSecurityGroupId"
        SubnetId: !Ref "PublicSubnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"replace":{"source":"${S3Loc}"}},"version":"3.1.0"}}'
        - {
          S3Loc: !Ref BootstrapIgnitionLocation
        }

  RegisterBootstrapApiTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref ExternalApiTargetGroupArn
      TargetIp: !GetAtt BootstrapInstance.PrivateIp

  RegisterBootstrapInternalApiTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalApiTargetGroupArn
      TargetIp: !GetAtt BootstrapInstance.PrivateIp

  RegisterBootstrapInternalServiceTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
```

```
      Properties:
        ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
        TargetArn: !Ref InternalServiceTargetGroupArn
        TargetIp: !GetAtt BootstrapInstance.PrivateIp

Outputs:
  BootstrapInstanceId:
    Description: Bootstrap Instance ID.
    Value: !Ref BootstrapInstance

  BootstrapPublicIp:
    Description: The bootstrap node public IP address.
    Value: !GetAtt BootstrapInstance.PublicIp

  BootstrapPrivateIp:
    Description: The bootstrap node private IP address.
    Value: !GetAtt BootstrapInstance.PrivateIp
```

**Additional resources**

- RHCOS AMIs for the AWS infrastructure(AWS documentation)

#### 4.4.10.2. Creating the control plane machines in AWS

You must create the control plane machines in Amazon Web Services (AWS) that your cluster will use.

You can use the provided CloudFormation template and a custom parameter file to create a stack of AWS resources that represent the control plane nodes.

IMPORTANT

The CloudFormation template creates a stack that represents three control plane nodes.

NOTE

If you do not use the provided CloudFormation template to create your control plane nodes, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

**Prerequisites**

- You configured an AWS account.
- You added your AWS keys and region to your local AWS profile by running **aws configure**.
- You generated the Ignition config files for your cluster.
- You created and configured a VPC and associated subnets in AWS.
- You created and configured DNS, load balancers, and listeners in AWS.
- You created the security groups and roles required for your cluster in AWS.

- You created the bootstrap machine.

**Procedure**

1. Create a JSON file that contains the parameter values that the template requires:

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
    "ParameterKey": "RhcosAmi", 3
    "ParameterValue": "ami-<random_string>" 4
  },
  {
    "ParameterKey": "AutoRegisterDNS", 5
    "ParameterValue": "yes" 6
  },
  {
    "ParameterKey": "PrivateHostedZoneId", 7
    "ParameterValue": "<random_string>" 8
  },
  {
    "ParameterKey": "PrivateHostedZoneName", 9
    "ParameterValue": "mycluster.example.com" 10
  },
  {
    "ParameterKey": "Master0Subnet", 11
    "ParameterValue": "subnet-<random_string>" 12
  },
  {
    "ParameterKey": "Master1Subnet", 13
    "ParameterValue": "subnet-<random_string>" 14
  },
  {
    "ParameterKey": "Master2Subnet", 15
    "ParameterValue": "subnet-<random_string>" 16
  },
  {
    "ParameterKey": "MasterSecurityGroupId", 17
    "ParameterValue": "sg-<random_string>" 18
  },
  {
    "ParameterKey": "IgnitionLocation", 19
    "ParameterValue": "https://api-int.<cluster_name>.<domain_name>:22623/config/master"
20
  },
  {
    "ParameterKey": "CertificateAuthorities", 21
    "ParameterValue": "data:text/plain;charset=utf-8;base64,ABC...xYz==" 22
  },
  {
    "ParameterKey": "MasterInstanceProfileName", (23)
    "ParameterValue": "<roles_stack>-MasterInstanceProfile-<random_string>" (24)
  },
  {
    "ParameterKey": "MasterInstanceType", (25)
    "ParameterValue": "" (26)
  },
  {
    "ParameterKey": "AutoRegisterELB", (27)
    "ParameterValue": "yes" (28)
  },
  {
    "ParameterKey": "RegisterNlbIpTargetsLambdaArn", (29)
    "ParameterValue": "arn:aws:lambda:<aws_region>:<account_number>:function:
<dns_stack_name>-RegisterNlbIpTargets-<random_string>" (30)
  },
  {
    "ParameterKey": "ExternalApiTargetGroupArn", (31)
    "ParameterValue": "arn:aws:elasticloadbalancing:<aws_region>:
<account_number>:targetgroup/<dns_stack_name>-Exter-<random_string>" (32)
  },
  {
    "ParameterKey": "InternalApiTargetGroupArn", (33)
    "ParameterValue": "arn:aws:elasticloadbalancing:<aws_region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" (34)
  },
  {
    "ParameterKey": "InternalServiceTargetGroupArn", (35)
    "ParameterValue": "arn:aws:elasticloadbalancing:<aws_region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" (36)
  }
]
```

1 The name for your cluster infrastructure that is encoded in your Ignition config files for the cluster.

2 Specify the infrastructure name that you extracted from the Ignition config file metadata, which has the format **<cluster-name>-<random-string>**.

3 Current Red Hat Enterprise Linux CoreOS (RHCOS) AMI to use for the control plane machines based on your selected architecture.

4 Specify an **AWS::EC2::Image::Id** value.

5 Whether or not to perform DNS etcd registration.

6 Specify **yes** or **no**. If you specify **yes**, you must provide hosted zone information.

7 The Route 53 private zone ID to register the etcd targets with.

8 Specify the **PrivateHostedZoneId** value from the output of the CloudFormation template for DNS and load balancing.

9 The Route 53 zone to register the targets with.

10 Specify **<cluster_name>.<domain_name>** where **<domain_name>** is the Route 53 base domain that you used when you generated **install-config.yaml** file for the cluster. Do not

11 13 15 A subnet, preferably private, to launch the control plane machines on.

12 14 16 Specify a subnet from the **PrivateSubnets** value from the output of the CloudFormation template for DNS and load balancing.

17 The master security group ID to associate with control plane nodes.

18 Specify the **MasterSecurityGroupId** value from the output of the CloudFormation template for the security group and roles.

19 The location to fetch control plane Ignition config file from.

20 Specify the generated Ignition config file location, **https://api-int.<cluster_name>.<domain_name>:22623/config/master**.

21 The base64 encoded certificate authority string to use.

22 Specify the value from the **master.ign** file that is in the installation directory. This value is the long string with the format **data:text/plain;charset=utf-8;base64,ABC…xYz==**.

23 The IAM profile to associate with control plane nodes.

24 Specify the **MasterInstanceProfile** parameter value from the output of the CloudFormation template for the security group and roles.

25 The type of AWS instance to use for the control plane machines based on your selected architecture.

26 The instance type value corresponds to the minimum resource requirements for control plane machines. For example **m6i.xlarge** is a type for AMD64 and **m6g.xlarge** is a type for ARM64.

27 Whether or not to register a network load balancer (NLB).

28 Specify **yes** or **no**. If you specify **yes**, you must provide a Lambda Amazon Resource Name (ARN) value.

29 The ARN for NLB IP target registration lambda group.

30 Specify the **RegisterNlbIpTargetsLambda** value from the output of the CloudFormation template for DNS and load balancing. Use **arn:aws-us-gov** if deploying the cluster to an AWS GovCloud region.

31 The ARN for external API load balancer target group.

32 Specify the **ExternalApiTargetGroupArn** value from the output of the CloudFormation template for DNS and load balancing. Use **arn:aws-us-gov** if deploying the cluster to an AWS GovCloud region.

33 The ARN for internal API load balancer target group.

34 Specify the **InternalApiTargetGroupArn** value from the output of the CloudFormation

35 The ARN for internal service load balancer target group.

36 Specify the **InternalServiceTargetGroupArn** value from the output of the CloudFormation template for DNS and load balancing. Use **arn:aws-us-gov** if deploying the cluster to an AWS GovCloud region.

2. Copy the template from the **CloudFormation template for control plane machines**section of this topic and save it as a YAML file on your computer. This template describes the control plane machines that your cluster requires.

3. If you specified an **m5** instance type as the value for **MasterInstanceType**, add that instance type to the **MasterInstanceType.AllowedValues** parameter in the CloudFormation template.

4. Launch the CloudFormation template to create a stack of AWS resources that represent the control plane nodes:

IMPORTANT

You must enter the command on a single line.

```
$ aws cloudformation create-stack --stack-name <name> 1
     --template-body file://<template>.yaml 2
     --parameters file://<parameters>.json 3
```

1 **<name>** is the name for the CloudFormation stack, such as **cluster-control-plane**. You need the name of this stack if you remove the cluster.

2 **<template>** is the relative path to and name of the CloudFormation template YAML file that you saved.

3 **<parameters>** is the relative path to and name of the CloudFormation parameters JSON file.

**Example output**

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-control-plane/21c7e2b0-2ee2-
11eb-c6f6-0aa34627df4b
```

NOTE

The CloudFormation template creates a stack that represents three control plane nodes.

5. Confirm that the template components exist:

```
$ aws cloudformation describe-stacks --stack-name <name>
```

#### 4.4.10.3. CloudFormation template for control plane machines

You can use the following CloudFormation template to deploy the control plane machines that you need for your OpenShift Container Platform cluster.

**Example 4.29. CloudFormation template for control plane machines**

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Node Launch (EC2 master instances)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a maximum of 27 characters.
    Description: A short, unique cluster ID used to tag nodes for the kubelet cloud provider.
    Type: String
  RhcosAmi:
    Description: Current Red Hat Enterprise Linux CoreOS AMI to use for bootstrap.
    Type: AWS::EC2::Image::Id
  AutoRegisterDNS:
    Default: ""
    Description: unused
    Type: String
  PrivateHostedZoneId:
    Default: ""
    Description: unused
    Type: String
  PrivateHostedZoneName:
    Default: ""
    Description: unused
    Type: String
  Master0Subnet:
    Description: The subnets, recommend private, to launch the master nodes into.
    Type: AWS::EC2::Subnet::Id
  Master1Subnet:
    Description: The subnets, recommend private, to launch the master nodes into.
    Type: AWS::EC2::Subnet::Id
  Master2Subnet:
    Description: The subnets, recommend private, to launch the master nodes into.
    Type: AWS::EC2::Subnet::Id
  MasterSecurityGroupId:
    Description: The master security group ID to associate with master nodes.
    Type: AWS::EC2::SecurityGroup::Id
  IgnitionLocation:
    Default: https://api-int.$CLUSTER_NAME.$DOMAIN:22623/config/master
    Description: Ignition config file location.
    Type: String
  CertificateAuthorities:
    Default: data:text/plain;charset=utf-8;base64,ABC...xYz==
    Description: Base64 encoded certificate authority string to use.
    Type: String
  MasterInstanceProfileName:
    Description: IAM profile to associate with master nodes.
    Type: String
  MasterInstanceType:
    Default: m5.xlarge
    Type: String

  AutoRegisterELB:
    Default: "yes"
    AllowedValues:
    - "yes"
    - "no"
    Description: Do you want to invoke NLB registration, which requires a Lambda ARN parameter?
    Type: String
  RegisterNlbIpTargetsLambdaArn:
    Description: ARN for NLB IP target registration lambda. Supply the value from the cluster
infrastructure or select "no" for AutoRegisterELB.
    Type: String
  ExternalApiTargetGroupArn:
    Description: ARN for external API load balancer target group. Supply the value from the cluster
infrastructure or select "no" for AutoRegisterELB.
    Type: String
  InternalApiTargetGroupArn:
    Description: ARN for internal API load balancer target group. Supply the value from the cluster
infrastructure or select "no" for AutoRegisterELB.
    Type: String
  InternalServiceTargetGroupArn:
    Description: ARN for internal service load balancer target group. Supply the value from the
cluster infrastructure or select "no" for AutoRegisterELB.
    Type: String

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - InfrastructureName
    - Label:
        default: "Host Information"
      Parameters:
      - MasterInstanceType
      - RhcosAmi
      - IgnitionLocation
      - CertificateAuthorities
      - MasterSecurityGroupId
      - MasterInstanceProfileName
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcId
      - AllowedBootstrapSshCidr
      - Master0Subnet
      - Master1Subnet
      - Master2Subnet
    - Label:
        default: "Load Balancer Automation"
      Parameters:
      - AutoRegisterELB
      - RegisterNlbIpTargetsLambdaArn
      - ExternalApiTargetGroupArn
      - InternalApiTargetGroupArn
      - InternalServiceTargetGroupArn
    ParameterLabels:
      InfrastructureName:
        default: "Infrastructure Name"
      VpcId:
        default: "VPC ID"
      Master0Subnet:
        default: "Master-0 Subnet"
      Master1Subnet:
        default: "Master-1 Subnet"
      Master2Subnet:
        default: "Master-2 Subnet"
      MasterInstanceType:
        default: "Master Instance Type"
      MasterInstanceProfileName:
        default: "Master Instance Profile Name"
      RhcosAmi:
        default: "Red Hat Enterprise Linux CoreOS AMI ID"
      BootstrapIgnitionLocation:
        default: "Master Ignition Source"
      CertificateAuthorities:
        default: "Ignition CA String"
      MasterSecurityGroupId:
        default: "Master Security Group ID"
      AutoRegisterELB:
        default: "Use Provided ELB Automation"

Conditions:
  DoRegistration: !Equals ["yes", !Ref AutoRegisterELB]

Resources:
  Master0:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref RhcosAmi
      BlockDeviceMappings:
      - DeviceName: /dev/xvda
        Ebs:
          VolumeSize: "120"
          VolumeType: "gp2"
      IamInstanceProfile: !Ref MasterInstanceProfileName
      InstanceType: !Ref MasterInstanceType
      NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "MasterSecurityGroupId"
        SubnetId: !Ref "Master0Subnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"merge":[{"source":"${SOURCE}"}]},"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}"}]}},"version":"3.1.0"}}'
        - {
          SOURCE: !Ref IgnitionLocation,
          CA_BUNDLE: !Ref CertificateAuthorities,
        }
      Tags:
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "shared"

  RegisterMaster0:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref ExternalApiTargetGroupArn
      TargetIp: !GetAtt Master0.PrivateIp

  RegisterMaster0InternalApiTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalApiTargetGroupArn
      TargetIp: !GetAtt Master0.PrivateIp

  RegisterMaster0InternalServiceTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalServiceTargetGroupArn
      TargetIp: !GetAtt Master0.PrivateIp

  Master1:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref RhcosAmi
      BlockDeviceMappings:
      - DeviceName: /dev/xvda
        Ebs:
          VolumeSize: "120"
          VolumeType: "gp2"
      IamInstanceProfile: !Ref MasterInstanceProfileName
      InstanceType: !Ref MasterInstanceType
      NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "MasterSecurityGroupId"
        SubnetId: !Ref "Master1Subnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"merge":[{"source":"${SOURCE}"}]},"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}"}]}},"version":"3.1.0"}}'
        - {
          SOURCE: !Ref IgnitionLocation,
          CA_BUNDLE: !Ref CertificateAuthorities,
        }
      Tags:
```

```
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "shared"

  RegisterMaster1:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref ExternalApiTargetGroupArn
      TargetIp: !GetAtt Master1.PrivateIp

  RegisterMaster1InternalApiTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalApiTargetGroupArn
      TargetIp: !GetAtt Master1.PrivateIp

  RegisterMaster1InternalServiceTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalServiceTargetGroupArn
      TargetIp: !GetAtt Master1.PrivateIp

  Master2:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref RhcosAmi
      BlockDeviceMappings:
      - DeviceName: /dev/xvda
        Ebs:
          VolumeSize: "120"
          VolumeType: "gp2"
      IamInstanceProfile: !Ref MasterInstanceProfileName
      InstanceType: !Ref MasterInstanceType
      NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "MasterSecurityGroupId"
        SubnetId: !Ref "Master2Subnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"merge":[{"source":"${SOURCE}"}]},"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}"}]}},"version":"3.1.0"}}'
        - {
          SOURCE: !Ref IgnitionLocation,
          CA_BUNDLE: !Ref CertificateAuthorities,
        }
      Tags:
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "shared"
  RegisterMaster2:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref ExternalApiTargetGroupArn
      TargetIp: !GetAtt Master2.PrivateIp

  RegisterMaster2InternalApiTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalApiTargetGroupArn
      TargetIp: !GetAtt Master2.PrivateIp

  RegisterMaster2InternalServiceTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalServiceTargetGroupArn
      TargetIp: !GetAtt Master2.PrivateIp

Outputs:
  PrivateIPs:
    Description: The control-plane node private IP addresses.
    Value:
      !Join [
        ",",
        [!GetAtt Master0.PrivateIp, !GetAtt Master1.PrivateIp, !GetAtt Master2.PrivateIp]
      ]
```

### 4.4.11. Creating the worker nodes in AWS

You can create worker nodes in Amazon Web Services (AWS) for your cluster to use.

You can use the provided CloudFormation template and a custom parameter file to create a stack of AWS resources that represent a worker node.

**IMPORTANT**

The CloudFormation template creates a stack that represents one worker node. You must create a stack for each worker node.

**NOTE**

If you do not use the provided CloudFormation template to create your worker nodes, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

**Prerequisites**

- You configured an AWS account.
- You added your AWS keys and region to your local AWS profile by running **aws configure**.
- You generated the Ignition config files for your cluster.
- You created and configured a VPC and associated subnets in AWS.
- You created and configured DNS, load balancers, and listeners in AWS.
- You created the security groups and roles required for your cluster in AWS.
- You created the bootstrap machine.
- You created the control plane machines.

**Procedure**

1. Create a JSON file that contains the parameter values that the CloudFormation template requires:

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
    "ParameterKey": "RhcosAmi", 3
    "ParameterValue": "ami-<random_string>" 4
  },
  {
    "ParameterKey": "Subnet", 5
    "ParameterValue": "subnet-<random_string>" 6
  },
  {
    "ParameterKey": "WorkerSecurityGroupId", 7
    "ParameterValue": "sg-<random_string>" 8
  },
  {
    "ParameterKey": "IgnitionLocation", 9
    "ParameterValue": "https://api-int.<cluster_name>.<domain_name>:22623/config/worker"
10
  },
  {
    "ParameterKey": "CertificateAuthorities", 11
    "ParameterValue": "data:text/plain;charset=utf-8;base64,ABC...xYz==" 12
  },
  {
    "ParameterKey": "WorkerInstanceProfileName", 13
    "ParameterValue": "<roles_stack>-WorkerInstanceProfile-<random_string>" 14
  },
  {
    "ParameterKey": "WorkerInstanceType", 15
    "ParameterValue": "" (16)
  }
]
```

(1) The name for your cluster infrastructure that is encoded in your Ignition config files for the cluster.

(2) Specify the infrastructure name that you extracted from the Ignition config file metadata, which has the format **<cluster-name>-<random-string>**.

(3) Current Red Hat Enterprise Linux CoreOS (RHCOS) AMI to use for the worker nodes based on your selected architecture.

(4) Specify an **AWS::EC2::Image::Id** value.

(5) A subnet, preferably private, to start the worker nodes on.

(6) Specify a subnet from the **PrivateSubnets** value from the output of the CloudFormation template for DNS and load balancing.

(7) The worker security group ID to associate with worker nodes.

(8) Specify the **WorkerSecurityGroupId** value from the output of the CloudFormation template for the security group and roles.

(9) The location to fetch the bootstrap Ignition config file from.

(10) Specify the generated Ignition config location, **https://api-int.<cluster_name>.<domain_name>:22623/config/worker**.

(11) Base64 encoded certificate authority string to use.

(12) Specify the value from the **worker.ign** file that is in the installation directory. This value is the long string with the format **data:text/plain;charset=utf-8;base64,ABC…xYz==**.

(13) The IAM profile to associate with worker nodes.

(14) Specify the **WorkerInstanceProfile** parameter value from the output of the CloudFormation template for the security group and roles.

(15) The type of AWS instance to use for the compute machines based on your selected architecture.

(16) The instance type value corresponds to the minimum resource requirements for compute machines. For example **m6i.large** is a type for AMD64 and **m6g.large** is a type for ARM64.

2. Copy the template from the **CloudFormation template for worker machines** section of this topic and save it as a YAML file on your computer. This template describes the networking objects and load balancers that your cluster requires.

3. Optional: If you specified an **m5** instance type as the value for **WorkerInstanceType**, add that instance type to the **WorkerInstanceType.AllowedValues** parameter in the CloudFormation template.

4. Optional: If you are deploying with an AWS Marketplace image, update the **Worker0.type.properties.ImageID** parameter with the AMI ID that you obtained from your subscription.

5. Use the CloudFormation template to create a stack of AWS resources that represent a worker node:

IMPORTANT

You must enter the command on a single line.

```
$ aws cloudformation create-stack --stack-name <name> (1)
     --template-body file://<template>.yaml \ (2)
     --parameters file://<parameters>.json (3)
```

(1) **<name>** is the name for the CloudFormation stack, such as **cluster-worker-1**. You need the name of this stack if you remove the cluster.

(2) **<template>** is the relative path to and name of the CloudFormation template YAML file that you saved.

(3) **<parameters>** is the relative path to and name of the CloudFormation parameters JSON file.

**Example output**

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-worker-1/729ee301-1c2a-
11eb-348f-sd9888c65b59
```

NOTE

The CloudFormation template creates a stack that represents one worker node.

6. Confirm that the template components exist:

```
$ aws cloudformation describe-stacks --stack-name <name>
```

7. Continue to create worker stacks until you have created enough worker machines for your cluster. You can create additional worker stacks by referencing the same template and parameter files and specifying a different stack name.

IMPORTANT

You must create at least two worker machines, so you must create at least two stacks that use this CloudFormation template.

#### 4.4.11.1. CloudFormation template for compute machines

You can deploy the compute machines that you need for your OpenShift Container Platform cluster by using the following CloudFormation template.

**Example 4.30. CloudFormation template for compute machines**

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Node Launch (EC2 worker instance)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, unique cluster ID used to tag nodes for the kubelet cloud provider.
    Type: String
  RhcosAmi:
    Description: Current Red Hat Enterprise Linux CoreOS AMI to use for bootstrap.
    Type: AWS::EC2::Image::Id
  Subnet:
    Description: The subnets, recommend private, to launch the worker nodes into.
    Type: AWS::EC2::Subnet::Id
  WorkerSecurityGroupId:
    Description: The worker security group ID to associate with worker nodes.
    Type: AWS::EC2::SecurityGroup::Id
  IgnitionLocation:
    Default: https://api-int.$CLUSTER_NAME.$DOMAIN:22623/config/worker
    Description: Ignition config file location.
    Type: String
  CertificateAuthorities:
    Default: data:text/plain;charset=utf-8;base64,ABC...xYz==
    Description: Base64 encoded certificate authority string to use.
    Type: String
  WorkerInstanceProfileName:
    Description: IAM profile to associate with worker nodes.
    Type: String
  WorkerInstanceType:
    Default: m5.large
    Type: String

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - InfrastructureName
    - Label:
        default: "Host Information"
      Parameters:
      - WorkerInstanceType
      - RhcosAmi
      - IgnitionLocation
      - CertificateAuthorities
      - WorkerSecurityGroupId
      - WorkerInstanceProfileName
    - Label:
        default: "Network Configuration"
      Parameters:
      - Subnet
    ParameterLabels:
      Subnet:
        default: "Subnet"
      InfrastructureName:
        default: "Infrastructure Name"
      WorkerInstanceType:
        default: "Worker Instance Type"
      WorkerInstanceProfileName:
        default: "Worker Instance Profile Name"
      RhcosAmi:
        default: "Red Hat Enterprise Linux CoreOS AMI ID"
      IgnitionLocation:
        default: "Worker Ignition Source"
      CertificateAuthorities:
        default: "Ignition CA String"
      WorkerSecurityGroupId:
        default: "Worker Security Group ID"

Resources:
  Worker0:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref RhcosAmi
      BlockDeviceMappings:
      - DeviceName: /dev/xvda
        Ebs:
          VolumeSize: "120"
          VolumeType: "gp2"
      IamInstanceProfile: !Ref WorkerInstanceProfileName
      InstanceType: !Ref WorkerInstanceType
      NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "WorkerSecurityGroupId"
        SubnetId: !Ref "Subnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"merge":[{"source":"${SOURCE}"}]},"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}"}]}},"version":"3.1.0"}}'
        - {
          SOURCE: !Ref IgnitionLocation,
          CA_BUNDLE: !Ref CertificateAuthorities,
        }
      Tags:
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "shared"

Outputs:
  PrivateIP:
    Description: The compute node private IP address.
    Value: !GetAtt Worker0.PrivateIp
```

### 4.4.11.2. Creating the CloudFormation stack for compute machines

You can create a stack of AWS resources for the compute machines by using the CloudFormation template that was previously shared.

**IMPORTANT**

When you use the CloudFormation template for the control plane machines, the template provisions all three control plane machines with a single stack; however, when you use the CloudFormation template to deploy the compute machines, you must create the number of stacks based on the number that you defined in the **install-config.yaml** file. Each stack is provisioned once for each machine. To provision a new compute machine, you must change the stack name.

**Procedure**

- To create the CloudFormation stack for compute machines, run the following command:

```
$ aws cloudformation create-stack --stack-name <name> \ 1
    --template-body file://<template>.yaml \ 2
    --parameters file://<parameters>.json 3
```

1 Specify the **<name>** with the name for the CloudFormation stack, such as **cluster-worker-1**. You need the name of this stack if you remove the cluster.

2 Specify the relative path and the name of the CloudFormation template YAML file that you saved.

3 Specify the relative path and the name of the JSON file for the CloudFormation parameters.

**Example output**

```
arn:aws:cloudformation:us-east-1:269333783861:stack/cluster-worker-1/729ee301-1c2a-
11eb-348f-sd9888c65b59
```

## 4.4.12. Initializing the bootstrap sequence on AWS with user-provisioned infrastructure

After you create all of the required infrastructure in Amazon Web Services (AWS), you can start the bootstrap sequence that initializes the OpenShift Container Platform control plane.

**Prerequisites**

- You configured an AWS account.
- You added your AWS keys and region to your local AWS profile by running **aws configure**.
- You generated the Ignition config files for your cluster.
- You created and configured a VPC and associated subnets in AWS.
- You created and configured DNS, load balancers, and listeners in AWS.

- You created the security groups and roles required for your cluster in AWS.
- You created the bootstrap machine.
- You created the control plane machines.
- You created the worker nodes.

### Procedure

1. Change to the directory that contains the installation program and start the bootstrap process that initializes the OpenShift Container Platform control plane:

```
$ ./openshift-install wait-for bootstrap-complete --dir <installation_directory> \ 1
    --log-level=info 2
```

1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2 To view different installation details, specify **warn**, **debug**, or **error** instead of **info**.

#### Example output

```
INFO Waiting up to 20m0s for the Kubernetes API at
https://api.mycluster.example.com:6443...
INFO API v1.32.3 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
INFO Time elapsed: 1s
```

If the command exits without a **FATAL** warning, your OpenShift Container Platform control plane has initialized.

NOTE

After the control plane initializes, it sets up the compute nodes and installs additional services in the form of Operators.

### Additional resources

- See Monitoring installation progress for details about monitoring the installation, bootstrap, and control plane logs as an OpenShift Container Platform installation progresses.
- See Gathering bootstrap node diagnostic data for information about troubleshooting issues related to the bootstrap process.

## 4.4.13. Approving the certificate signing requests for your machines

When you add machines to a cluster, two pending certificate signing requests (CSRs) are generated for each machine that you added. You must confirm that these CSRs are approved or, if necessary, approve them yourself. The client requests must be approved first, followed by the server requests.

### Prerequisites

- You added machines to your cluster.

## Procedure

1. Confirm that the cluster recognizes the machines:

   ```
   $ oc get nodes
   ```

   ### Example output

   ```
   NAME      STATUS    ROLES   AGE  VERSION
   master-0  Ready     master  63m  v1.32.3
   master-1  Ready     master  63m  v1.32.3
   master-2  Ready     master  64m  v1.32.3
   ```

   The output lists all of the machines that you created.

   NOTE

   The preceding output might not include the compute nodes, also known as worker nodes, until some CSRs are approved.

2. Review the pending CSRs and ensure that you see the client requests with the **Pending** or **Approved** status for each machine that you added to the cluster:

   ```
   $ oc get csr
   ```

   ### Example output

   ```
   NAME        AGE     REQUESTOR                                                                   CONDITION
   csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-
   bootstrapper   Pending
   csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-
   bootstrapper   Pending
   ...
   ```

   In this example, two machines are joining the cluster. You might see more approved CSRs in the list.

3. If the CSRs were not approved, after all of the pending CSRs for the machines you added are in **Pending** status, approve the CSRs for your cluster machines:

   NOTE

   Because the CSRs rotate automatically, approve your CSRs within an hour of adding the machines to the cluster. If you do not approve them within an hour, the certificates will rotate, and more than two certificates will be present for each node. You must approve all of these certificates. After the client CSR is approved, the Kubelet creates a secondary CSR for the serving certificate, which requires manual approval. Then, subsequent serving certificate renewal requests are automatically approved by the **machine-approver** if the Kubelet requests a new certificate with identical parameters.

NOTE

For clusters running on platforms that are not machine API enabled, such as bare metal and other user-provisioned infrastructure, you must implement a method of automatically approving the kubelet serving certificate requests (CSRs). If a request is not approved, then the **oc exec**, **oc rsh**, and **oc logs** commands cannot succeed, because a serving certificate is required when the API server connects to the kubelet. Any operation that contacts the Kubelet endpoint requires this certificate approval to be in place. The method must watch for new CSRs, confirm that the CSR was submitted by the **node-bootstrapper** service account in the **system:node** or **system:admin** groups, and confirm the identity of the node.

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

NOTE

Some Operators might not become available until some CSRs are approved.

4. Now that your client requests are approved, you must review the server requests for each machine that you added to the cluster:

```
$ oc get csr
```

**Example output**

```
NAME        AGE     REQUESTOR                                                    CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. If the remaining CSRs are not approved, and are in the **Pending** status, approve the CSRs for your cluster machines:

- To approve them individually, run the following command for each valid CSR:

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** is the name of a CSR from the list of current CSRs.

- To approve all pending CSRs, run the following command:

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. After all client and server CSRs have been approved, the machines have the **Ready** status. Verify this by running the following command:

```
$ oc get nodes
```

**Example output**

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.32.3
master-1  Ready     master  73m  v1.32.3
master-2  Ready     master  74m  v1.32.3
worker-0  Ready     worker  11m  v1.32.3
worker-1  Ready     worker  11m  v1.32.3
```

NOTE

It can take a few minutes after approval of the server CSRs for the machines to transition to the **Ready** status.

**Additional information**

- Certificate Signing Requests

## 4.4.14. Initial Operator configuration

After the control plane initializes, you must immediately configure some Operators so that they all become available.

**Prerequisites**

- Your control plane has initialized.

**Procedure**

1. Watch the cluster components come online:

```
$ watch -n5 oc get clusteroperators
```

**Example output**

```
NAME                                 VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                        4.19.0    True        False         False      19m
baremetal                            4.19.0    True        False         False      37m
cloud-credential                      4.19.0    True        False         False      40m
cluster-autoscaler                     4.19.0    True        False         False      37m
config-operator                       4.19.0    True        False         False      38m
```

```
console                                    4.19.0    True       False        False      26m
csi-snapshot-controller                    4.19.0    True       False        False      37m
dns                                        4.19.0    True       False        False      37m
etcd                                       4.19.0    True       False        False      36m
image-registry                             4.19.0    True       False        False      31m
ingress                                    4.19.0    True       False        False      30m
insights                                   4.19.0    True       False        False      31m
kube-apiserver                             4.19.0    True       False        False      26m
kube-controller-manager                    4.19.0    True       False        False      36m
kube-scheduler                             4.19.0    True       False        False      36m
kube-storage-version-migrator              4.19.0    True       False        False      37m
machine-api                                4.19.0    True       False        False      29m
machine-approver                           4.19.0    True       False        False      37m
machine-config                             4.19.0    True       False        False      36m
marketplace                                4.19.0    True       False        False      37m
monitoring                                 4.19.0    True       False        False      29m
network                                    4.19.0    True       False        False      38m
node-tuning                                4.19.0    True       False        False      37m
openshift-apiserver                        4.19.0    True       False        False      32m
openshift-controller-manager               4.19.0    True       False        False      30m
openshift-samples                          4.19.0    True       False        False      32m
operator-lifecycle-manager                 4.19.0    True       False        False      37m
operator-lifecycle-manager-catalog         4.19.0    True       False        False      37m
operator-lifecycle-manager-packageserver   4.19.0    True       False        False      32m
service-ca                                 4.19.0    True       False        False      38m
storage                                    4.19.0    True       False        False      37m
```

2. Configure the Operators that are not available.

### 4.4.14.1. Disabling the default OperatorHub catalog sources

Operator catalogs that source content provided by Red Hat and community projects are configured for OperatorHub by default during an OpenShift Container Platform installation. In a restricted network environment, you must disable the default catalogs as a cluster administrator.

#### Procedure

- Disable the sources for the default catalogs by adding **disableAllDefaultSources: true** to the **OperatorHub** object:

```
$ oc patch OperatorHub cluster --type json \
    -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

#### TIP

Alternatively, you can use the web console to manage catalog sources. From the **Administration → Cluster Settings → Configuration → OperatorHub** page, click the **Sources** tab, where you can create, update, delete, disable, and enable individual sources.

### 4.4.14.2. Image registry storage configuration

Amazon Web Services provides default storage, which means the Image Registry Operator is available after installation. However, if the Registry Operator cannot create an S3 bucket and automatically configure storage, you must manually configure registry storage.

Instructions are shown for configuring a persistent volume, which is required for production clusters. Where applicable, instructions are shown for configuring an empty directory as the storage location, which is available for only non-production clusters.

Additional instructions are provided for allowing the image registry to use block storage types by using the **Recreate** rollout strategy during upgrades.

#### 4.4.14.2.1. Configuring registry storage for AWS with user-provisioned infrastructure

During installation, your cloud credentials are sufficient to create an Amazon S3 bucket and the Registry Operator will automatically configure storage.

If the Registry Operator cannot create an S3 bucket and automatically configure storage, you can create an S3 bucket and configure storage with the following procedure.

> **WARNING**
>
> To secure your registry images in AWS, block public access to the S3 bucket.

**Prerequisites**

- You have a cluster on AWS with user-provisioned infrastructure.
- For Amazon S3 storage, the secret is expected to contain two keys:
  - **REGISTRY_STORAGE_S3_ACCESSKEY**
  - **REGISTRY_STORAGE_S3_SECRETKEY**

**Procedure**

1. Set up a Bucket Lifecycle Policy to abort incomplete multipart uploads that are one day old.
2. Fill in the storage configuration in **configs.imageregistry.operator.openshift.io/cluster**:

```
$ oc edit configs.imageregistry.operator.openshift.io/cluster
```

**Example configuration**

```
apiVersion: imageregistry.operator.openshift.io/v1
kind: Config
metadata:
  name: cluster
spec:
  storage:
    s3:
      bucket: <bucket_name>
      region: <region_name>
```

#### 4.4.14.2.2. Configuring storage for the image registry in non-production clusters

You must configure storage for the Image Registry Operator. For non-production clusters, you can set the image registry to an empty directory. If you do so, all images are lost if you restart the registry.

**Procedure**

- To set the image registry storage to an empty directory:

  ```
  $ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":
  {"storage":{"emptyDir":{}}}}'
  ```

  > **WARNING**
  >
  > Configure this option for only non-production clusters.

  If you run this command before the Image Registry Operator initializes its components, the **oc patch** command fails with the following error:

  ```
  Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
  ```

  Wait a few minutes and run the command again.

### 4.4.15. Deleting the bootstrap resources

After you complete the initial Operator configuration for the cluster, remove the bootstrap resources from Amazon Web Services (AWS).

**Prerequisites**

- You completed the initial Operator configuration for your cluster.

**Procedure**

1. Delete the bootstrap resources. If you used the CloudFormation template, delete its stack:
   - Delete the stack by using the AWS CLI:

     ```
     $ aws cloudformation delete-stack --stack-name <name> 1
     ```

     1 **<name>** is the name of your bootstrap stack.

   - Delete the stack by using the AWS CloudFormation console.

### 4.4.16. Creating the Ingress DNS Records

If you removed the DNS Zone configuration, manually create DNS records that point to the Ingress load balancer. You can create either a wildcard record or specific records. While the following procedure uses A records, you can use other record types that you require, such as CNAME or alias.

### Prerequisites

- You deployed an OpenShift Container Platform cluster on Amazon Web Services (AWS) that uses infrastructure that you provisioned.
- You installed the OpenShift CLI (**oc**).
- You installed the **jq** package.
- You downloaded the AWS CLI and installed it on your computer. See Install the AWS CLI Using the Bundled Installer (Linux, macOS, or Unix).

### Procedure

1. Determine the routes to create.
   - To create a wildcard record, use ***.apps.<cluster_name>.<domain_name>**, where **<cluster_name>** is your cluster name, and **<domain_name>** is the Route 53 base domain for your OpenShift Container Platform cluster.
   - To create specific records, you must create a record for each route that your cluster uses, as shown in the output of the following command:

     ```
     $ oc get --all-namespaces -o jsonpath='{range .items[*]}{range .status.ingress[*]}{.host}
     {"\n"}{end}{end}' routes
     ```

     **Example output**

     ```
     oauth-openshift.apps.<cluster_name>.<domain_name>
     console-openshift-console.apps.<cluster_name>.<domain_name>
     downloads-openshift-console.apps.<cluster_name>.<domain_name>
     alertmanager-main-openshift-monitoring.apps.<cluster_name>.<domain_name>
     prometheus-k8s-openshift-monitoring.apps.<cluster_name>.<domain_name>
     ```

2. Retrieve the Ingress Operator load balancer status and note the value of the external IP address that it uses, which is shown in the **EXTERNAL-IP** column:

   ```
   $ oc -n openshift-ingress get service router-default
   ```

   **Example output**

   ```
   NAME             TYPE           CLUSTER-IP      EXTERNAL-IP                            PORT(S)
   AGE
   router-default   LoadBalancer   172.30.62.215   ab3...28.us-east-2.elb.amazonaws.com
   80:31499/TCP,443:30693/TCP   5m
   ```

3. Locate the hosted zone ID for the load balancer:

   ```
   $ aws elb describe-load-balancers | jq -r '.LoadBalancerDescriptions[] | select(.DNSName ==
   "<external_ip>").CanonicalHostedZoneNameID' 1
   ```

1 For **<external_ip>**, specify the value of the external IP address of the Ingress Operator load balancer that you obtained.

**Example output**

```
Z3AADJGX6KTTL2
```

The output of this command is the load balancer hosted zone ID.

4. Obtain the public hosted zone ID for your cluster's domain:

```
$ aws route53 list-hosted-zones-by-name \
            --dns-name "<domain_name>" \ 1
            --query 'HostedZones[? Config.PrivateZone != `true` && Name ==
`<domain_name>.`].Id' 2
            --output text
```

1 2 For **<domain_name>**, specify the Route 53 base domain for your OpenShift Container Platform cluster.

**Example output**

```
/hostedzone/Z3URY6TWQ91KVV
```

The public hosted zone ID for your domain is shown in the command output. In this example, it is **Z3URY6TWQ91KVV**.

5. Add the alias records to your private zone:

```
$ aws route53 change-resource-record-sets --hosted-zone-id "<private_hosted_zone_id>" --
change-batch '{ 1
>   "Changes": [
>     {
>       "Action": "CREATE",
>       "ResourceRecordSet": {
>         "Name": "\\052.apps.<cluster_domain>", 2
>         "Type": "A",
>         "AliasTarget":{
>           "HostedZoneId": "<hosted_zone_id>", 3
>           "DNSName": "<external_ip>.", 4
>           "EvaluateTargetHealth": false
>         }
>       }
>     }
>   ]
> }'
```

1 For **<private_hosted_zone_id>**, specify the value from the output of the CloudFormation template for DNS and load balancing.

2

For **<cluster_domain>**, specify the domain or subdomain that you use with your OpenShift Container Platform cluster.

3 For **<hosted_zone_id>**, specify the public hosted zone ID for the load balancer that you obtained.

4 For **<external_ip>**, specify the value of the external IP address of the Ingress Operator load balancer. Ensure that you include the trailing period (**.**) in this parameter value.

6. Add the records to your public zone:

```
$ aws route53 change-resource-record-sets --hosted-zone-id "<public_hosted_zone_id>"" --
change-batch '{ 1
>   "Changes": [
>     {
>       "Action": "CREATE",
>       "ResourceRecordSet": {
>         "Name": "\\052.apps.<cluster_domain>", 2
>         "Type": "A",
>         "AliasTarget":{
>           "HostedZoneId": "<hosted_zone_id>", 3
>           "DNSName": "<external_ip>.", 4
>           "EvaluateTargetHealth": false
>         }
>       }
>     }
>   ]
> }'
```

1 For **<public_hosted_zone_id>**, specify the public hosted zone for your domain.

2 For **<cluster_domain>**, specify the domain or subdomain that you use with your OpenShift Container Platform cluster.

3 For **<hosted_zone_id>**, specify the public hosted zone ID for the load balancer that you obtained.

4 For **<external_ip>**, specify the value of the external IP address of the Ingress Operator load balancer. Ensure that you include the trailing period (**.**) in this parameter value.

### 4.4.17. Completing an AWS installation on user-provisioned infrastructure

After you start the OpenShift Container Platform installation on Amazon Web Service (AWS) user-provisioned infrastructure, monitor the deployment to completion.

#### Prerequisites

- You removed the bootstrap node for an OpenShift Container Platform cluster on user-provisioned AWS infrastructure.
- You installed the **oc** CLI.

### Procedure

1. From the directory that contains the installation program, complete the cluster installation:

```
$ ./openshift-install --dir <installation_directory> wait-for install-complete 1
```

1 For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

#### Example output

```
INFO Waiting up to 40m0s for the cluster at https://api.mycluster.example.com:6443 to
initialize...
INFO Waiting up to 10m0s for the openshift-console route to be created...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/home/myuser/install_dir/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.mycluster.example.com
INFO Login to the console with user: "kubeadmin", and password: "password"
INFO Time elapsed: 1s
```

IMPORTANT

- The Ignition config files that the installation program generates contain certificates that expire after 24 hours, which are then renewed at that time. If the cluster is shut down before renewing the certificates and the cluster is later restarted after the 24 hours have elapsed, the cluster automatically recovers the expired certificates. The exception is that you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. See the documentation for *Recovering from expired control plane certificates* for more information.

- It is recommended that you use Ignition config files within 12 hours after they are generated because the 24-hour certificate rotates from 16 to 22 hours after the cluster is installed. By using the Ignition config files within 12 hours, you can avoid installation failure if the certificate update runs during installation.

2. Register your cluster on the Cluster registration page.

### 4.4.18. Logging in to the cluster by using the CLI

You can log in to your cluster as a default system user by exporting the cluster **kubeconfig** file. The **kubeconfig** file contains information about the cluster that is used by the CLI to connect a client to the correct cluster and API server. The file is specific to a cluster and is created during OpenShift Container Platform installation.

### Prerequisites

- You deployed an OpenShift Container Platform cluster.

- You installed the OpenShift CLI (**oc**).

### Procedure

1. Export the **kubeadmin** credentials by running the following command:

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig (1)
```

(1) For **<installation_directory>**, specify the path to the directory that you stored the installation files in.

2. Verify you can run **oc** commands successfully using the exported configuration by running the following command:

```
$ oc whoami
```

#### Example output

```
system:admin
```

### 4.4.19. Logging in to the cluster by using the web console

The **kubeadmin** user exists by default after an OpenShift Container Platform installation. You can log in to your cluster as the **kubeadmin** user by using the OpenShift Container Platform web console.

#### Prerequisites

- You have access to the installation host.
- You completed a cluster installation and all cluster Operators are available.

#### Procedure

1. Obtain the password for the **kubeadmin** user from the **kubeadmin-password** file on the installation host:

```
$ cat <installation_directory>/auth/kubeadmin-password
```

**NOTE**

Alternatively, you can obtain the **kubeadmin** password from the **<installation_directory>/.openshift_install.log** log file on the installation host.

2. List the OpenShift Container Platform web console route:

```
$ oc get routes -n openshift-console | grep 'console-openshift'
```

**NOTE**

Alternatively, you can obtain the OpenShift Container Platform route from the **<installation_directory>/.openshift_install.log** log file on the installation host.

**Example output**

```
console     console-openshift-console.apps.<cluster_name>.<base_domain>            console
https    reencrypt/Redirect    None
```

3. Navigate to the route detailed in the output of the preceding command in a web browser and log in as the **kubeadmin** user.

**Additional resources**

- Accessing the web console

**Additional resources**

- See About remote health monitoring for more information about the Telemetry service

### 4.4.20. Additional resources

- Working with stacks (AWS documentation)

### 4.4.21. Next steps

- Validate an installation.
- Customize your cluster.
- Configure image streams for the Cluster Samples Operator and the **must-gather** tool.
- Learn how to use Operator Lifecycle Manager in disconnected environments .
- If the mirror registry that you used to install your cluster has a trusted CA, add it to the cluster by configuring additional trust stores.
- If necessary, you can Remote health reporting .
- If necessary, see Registering your disconnected cluster
- If necessary, you can remove cloud provider credentials .

## 4.5. INSTALLING A CLUSTER WITH THE SUPPORT FOR CONFIGURING MULTI-ARCHITECTURE COMPUTE MACHINES

An OpenShift Container Platform cluster with multi-architecture compute machines supports compute machines with different architectures.

**NOTE**

When you have nodes with multiple architectures in your cluster, the architecture of your image must be consistent with the architecture of the node. You must ensure that the pod is assigned to the node with the appropriate architecture and that it matches the image architecture. For more information on assigning pods to nodes, see Scheduling workloads on clusters with multi-architecture compute machines.

You can install an AWS cluster with the support for configuring multi-architecture compute machines. After installing the AWS cluster, you can add multi-architecture compute machines to the cluster in the following ways:

- Adding 64-bit x86 compute machines to a cluster that uses 64-bit ARM control plane machines and already includes 64-bit ARM compute machines. In this case, 64-bit x86 is considered the secondary architecture.
- Adding 64-bit ARM compute machines to a cluster that uses 64-bit x86 control plane machines and already includes 64-bit x86 compute machines. In this case, 64-bit ARM is considered the secondary architecture.

NOTE

Before adding a secondary architecture node to your cluster, it is recommended to install the Multiarch Tuning Operator, and deploy a **ClusterPodPlacementConfig** custom resource. For more information, see "Managing workloads on multi-architecture clusters by using the Multiarch Tuning Operator".

## 4.5.1. Installing a cluster with multi-architecture support

You can install a cluster with the support for configuring multi-architecture compute machines.

### Prerequisites

- You installed the OpenShift CLI (**oc**).
- You have the OpenShift Container Platform installation program.
- You downloaded the pull secret for your cluster.

### Procedure

1. Check that the **openshift-install** binary is using the **multi** payload by running the following command:

```
$ ./openshift-install version
```

**Example output**

```
./openshift-install 4.19.0
built from commit abc123etc
release image quay.io/openshift-release-dev/ocp-release@sha256:abc123wxyzetc
release architecture multi
default architecture amd64
```

The output must contain **release architecture multi** to indicate that the **openshift-install** binary is using the **multi** payload.

2. Update the **install-config.yaml** file to configure the architecture for the nodes.

**Sample install-config.yaml file with multi-architecture configuration**

```
apiVersion: v1
```

```
baseDomain: example.openshift.com
compute:
- architecture: amd64 1
  hyperthreading: Enabled
  name: worker
  platform: {}
  replicas: 3
controlPlane:
  architecture: arm64 2
  name: master
  platform: {}
  replicas: 3
# ...
```

1 Specify the architecture of the worker node. You can set this field to either **arm64** or **amd64**.

2 Specify the control plane node architecture. You can set this field to either **arm64** or **amd64**.

### Next steps

- Deploying the cluster

### Additional resources

- Managing workloads on multi-architecture clusters by using the Multiarch Tuning Operator

# CHAPTER 5. INSTALLING A THREE-NODE CLUSTER ON AWS

In OpenShift Container Platform version 4.19, you can install a three-node cluster on Amazon Web Services (AWS). A three-node cluster consists of three control plane machines, which also act as compute machines.

This type of cluster provides a smaller, more resource efficient cluster, for cluster administrators and developers to use for testing, development, and production.

You can install a three-node cluster using either installer-provisioned or user-provisioned infrastructure.

NOTE

Deploying a three-node cluster using an AWS Marketplace image is not supported.

## 5.1. CONFIGURING A THREE-NODE CLUSTER

To configure a three-node cluster, set the number of worker nodes to **0** in the **install-config.yaml** file before you deploy the cluster.

Setting the number of worker nodes to **0** ensures that the control plane machines are schedulable. This allows application workloads to be scheduled to run from the control plane nodes.

NOTE

Because application workloads run from control plane nodes, additional subscriptions are required, as the control plane nodes are considered to be compute nodes.

### Prerequisites

- You have an existing **install-config.yaml** file.

### Procedure

1. Set the number of compute replicas to **0** in your **install-config.yaml** file, as shown in the following **compute** stanza:

   **Example install-config.yaml file for a three-node cluster**

   ```
   apiVersion: v1
   baseDomain: example.com
   compute:
   - name: worker
     platform: {}
     replicas: 0
   # ...
   ```

2. If you are deploying a cluster with user-provisioned infrastructure:
   - After you create the Kubernetes manifest files, make sure that the **spec.mastersSchedulable** parameter is set to **true** in **cluster-scheduler-02-config.yml** file. You can locate this file in **<installation_directory>/manifests**. For more information,

see "Creating the Kubernetes manifest and Ignition config files" in "Installing a cluster on user-provisioned infrastructure in AWS by using CloudFormation templates".

- Do not create additional worker nodes.

**Example cluster-scheduler-02-config.yml file for a three-node cluster**

```
apiVersion: config.openshift.io/v1
kind: Scheduler
metadata:
  creationTimestamp: null
  name: cluster
spec:
  mastersSchedulable: true
  policy:
    name: ""
status: {}
```

## 5.2. ADDITIONAL RESOURCES

- Installing a cluster on AWS with customizations
- Installing a cluster on user-provisioned infrastructure in AWS by using CloudFormation templates

# CHAPTER 6. UNINSTALLING A CLUSTER ON AWS

You can remove a cluster that you deployed to Amazon Web Services (AWS).

## 6.1. REMOVING A CLUSTER THAT USES INSTALLER-PROVISIONED INFRASTRUCTURE

You can remove a cluster that uses installer-provisioned infrastructure that you provisioned from your cloud platform.

NOTE

After uninstallation, check your cloud provider for any resources that were not removed properly, especially with user-provisioned infrastructure clusters. Some resources might exist because either the installation program did not create the resource or could not access the resource.

### Prerequisites

- You have a copy of the installation program that you used to deploy the cluster.
- You have the files that the installation program generated when you created your cluster.

### Procedure

1. From the directory that has the installation program on the computer that you used to install the cluster, run the following command:

   ```
   $ ./openshift-install destroy cluster \
   --dir <installation_directory> --log-level info
   ```

   where:

   **<installation_directory>**
   Specify the path to the directory that you stored the installation files in.

   **--log-level info**
   To view different details, specify **warn**, **debug**, or **error** instead of **info**.

   NOTE

   You must specify the directory that includes the cluster definition files for your cluster. The installation program requires the **metadata.json** file in this directory to delete the cluster.

2. Optional: Delete the **<installation_directory>** directory and the OpenShift Container Platform installation program.

## 6.2. DELETING AMAZON WEB SERVICES RESOURCES WITH THE CLOUD CREDENTIAL OPERATOR UTILITY

After uninstalling an OpenShift Container Platform cluster that uses short-term credentials managed outside the cluster, you can use the CCO utility (**ccoctl**) to remove the Amazon Web Services resources that **ccoctl** created during installation.

### Prerequisites

- Extract and prepare the **ccoctl** binary.
- Uninstall an OpenShift Container Platform cluster on AWS that uses short-term credentials.

### Procedure

- Delete the AWS resources that **ccoctl** created by running the following command:

  ```
  $ ccoctl aws delete \
    --name=<name> \
  ifdef::aws-sts
  [  --region=<aws_region>]
  ```

  where:

  **<name>**
  : Matches the name that was originally used to create and tag the cloud resources.

  **<aws_region>**
  : is the AWS region in which to delete cloud resources.

  #### Example output

  ```
  2021/04/08 17:50:41 Identity Provider object .well-known/openid-configuration deleted
  from the bucket <name>-oidc
  2021/04/08 17:50:42 Identity Provider object keys.json deleted from the bucket <name>-
  oidc
  2021/04/08 17:50:43 Identity Provider bucket <name>-oidc deleted
  2021/04/08 17:51:05 Policy <name>-openshift-cloud-credential-operator-cloud-credential-
  o associated with IAM Role <name>-openshift-cloud-credential-operator-cloud-credential-
  o deleted
  2021/04/08 17:51:05 IAM Role <name>-openshift-cloud-credential-operator-cloud-
  credential-o deleted
  2021/04/08 17:51:07 Policy <name>-openshift-cluster-csi-drivers-ebs-cloud-credentials
  associated with IAM Role <name>-openshift-cluster-csi-drivers-ebs-cloud-credentials
  deleted
  2021/04/08 17:51:07 IAM Role <name>-openshift-cluster-csi-drivers-ebs-cloud-
  credentials deleted
  2021/04/08 17:51:08 Policy <name>-openshift-image-registry-installer-cloud-credentials
  associated with IAM Role <name>-openshift-image-registry-installer-cloud-credentials
  deleted
  2021/04/08 17:51:08 IAM Role <name>-openshift-image-registry-installer-cloud-
  credentials deleted
  2021/04/08 17:51:09 Policy <name>-openshift-ingress-operator-cloud-credentials
  associated with IAM Role <name>-openshift-ingress-operator-cloud-credentials deleted
  2021/04/08 17:51:10 IAM Role <name>-openshift-ingress-operator-cloud-credentials
  deleted
  2021/04/08 17:51:11 Policy <name>-openshift-machine-api-aws-cloud-credentials
  associated with IAM Role <name>-openshift-machine-api-aws-cloud-credentials deleted
  ```

```
2021/04/08 17:51:11 IAM Role <name>-openshift-machine-api-aws-cloud-credentials
deleted
2021/04/08 17:51:39 Identity Provider with ARN arn:aws:iam::<aws_account_id>:oidc-
provider/<name>-oidc.s3.<aws_region>.amazonaws.com deleted
```

**Verification**

- To verify that the resources are deleted, query AWS. For more information, refer to AWS documentation.

## 6.3. DELETING A CLUSTER WITH A CONFIGURED AWS LOCAL ZONE INFRASTRUCTURE

After you install a cluster on Amazon Web Services (AWS) into an existing Virtual Private Cloud (VPC), and you set subnets for each Local Zone location, you can delete the cluster and any AWS resources associated with it.

The example in the procedure assumes that you created a VPC and its subnets by using a CloudFormation template.

**Prerequisites**

- You know the name of the CloudFormation stacks, **<local_zone_stack_name>** and **<vpc_stack_name>**, that were used during the creation of the network. You need the name of the stack to delete the cluster.
- You have access rights to the directory that contains the installation files that were created by the installation program.
- Your account includes a policy that provides you with permissions to delete the CloudFormation stack.

**Procedure**

1. Change to the directory that contains the stored installation program, and delete the cluster by using the **destroy cluster** command:

   ```
   $ ./openshift-install destroy cluster --dir <installation_directory> \
     --log-level=debug
   ```

   where:

   **<installation_directory>**

   Specify the directory that stored any files created by the installation program.

   **--log-level=debug**

   To view different log details, specify **error**, **info**, or **warn** instead of **debug**.

2. Delete the CloudFormation stack for the Local Zone subnet:

   ```
   $ aws cloudformation delete-stack --stack-name <local_zone_stack_name>
   ```

3. Delete the stack of resources that represent the VPC:

```
$ aws cloudformation delete-stack --stack-name <vpc_stack_name>
```

**Verification**

- Check that you removed the stack resources by issuing the following commands in the AWS CLI. The AWS CLI outputs that no template component exists.

  ```
  $ aws cloudformation describe-stacks --stack-name <local_zone_stack_name>
  ```

  ```
  $ aws cloudformation describe-stacks --stack-name <vpc_stack_name>
  ```

## 6.4. ADDITIONAL RESOURCES

- Working with stacks(AWS documentation)
- Opt into AWS Local Zones(AWS documentation)
- AWS Local Zones available locations(AWS documentation)
- AWS Local Zones features(AWS documentation)

# CHAPTER 7. INSTALLATION CONFIGURATION PARAMETERS FOR AWS

Before you deploy an OpenShift Container Platform cluster on AWS, you provide parameters to customize your cluster and the platform that hosts it. When you create the **install-config.yaml** file, you provide values for the required parameters through the command line. You can then modify the **install-config.yaml** file to customize your cluster further.

## 7.1. AVAILABLE INSTALLATION CONFIGURATION PARAMETERS FOR AWS

The following tables specify the required, optional, and AWS-specific installation configuration parameters that you can set as part of the installation process.

IMPORTANT

After installation, you cannot change these parameters in the **install-config.yaml** file.

### 7.1.1. Required configuration parameters

Required installation configuration parameters are described in the following table:

**Table 7.1. Required parameters**

| Parameter | Description |
|---|---|
| `apiVersion:` | The API version for the **install-config.yaml** content. The current version is **v1**. The installation program might also support older API versions.<br><br>**Value:** String |
| `baseDomain:` | The base domain of your cloud provider. The base domain is used to create routes to your OpenShift Container Platform cluster components. The full DNS name for your cluster is a combination of the **baseDomain** and **metadata.name** parameter values that uses the **<metadata.name>.<baseDomain>** format.<br><br>**Value:** A fully-qualified domain or subdomain name, such as **example.com**. |
| `metadata:` | Kubernetes resource **ObjectMeta**, from which only the **name** parameter is consumed.<br><br>**Value:** Object |

| Parameter | Description |
|---|---|
| `metadata: name:` | The name of the cluster. DNS records for the cluster are all subdomains of **{{.metadata.name}}.{{.baseDomain}}**.<br><br>**Value:** String of lowercase letters, hyphens (**-**), and periods (**.**), such as **dev**. |
| `platform:` | The configuration for the specific platform upon which to perform the installation: **aws**, **baremetal**, **azure**, **gcp**, **ibmcloud**, **nutanix**, **openstack**, **powervs**, **vsphere**, or **{}**. For additional information about **platform.<platform>** parameters, consult the table for your specific platform that follows.<br><br>**Value:** Object |
| `pullSecret:` | Get a pull secret from Red Hat OpenShift Cluster Manager to authenticate downloading container images for OpenShift Container Platform components from services such as Quay.io.<br><br>**Value:**<br>`{ "auths":{ "cloud.openshift.com":{ "auth":"b3Blb=", "email":"you@example.com" }, "quay.io":{ "auth":"b3Blb=", "email":"you@example.com" } } }` |

### 7.1.2. Network configuration parameters

You can customize your installation configuration based on the requirements of your existing network infrastructure. For example, you can expand the IP address block for the cluster network or configure different IP address blocks than the defaults.

Only IPv4 addresses are supported.

**Table 7.2. Network parameters**

| Parameter | Description |
|---|---|
| `networking:` | The configuration for the cluster network.<br><br>**Value:** Object<br><br>NOTE<br><br>You cannot change parameters specified by the **networking** object after installation. |
| `networking:`<br>`  networkType:` | The Red Hat OpenShift Networking network plugin to install.<br><br>**Value:OVNKubernetes**. **OVNKubernetes** is a Container Network Interface (CNI) plugin for Linux networks and hybrid networks that contain both Linux and Windows servers. The default value is **OVNKubernetes**. |
| `networking:`<br>`  clusterNetwork:` | The IP address blocks for pods.<br><br>The default value is **10.128.0.0/14** with a host prefix of /**23**.<br><br>If you specify multiple IP address blocks, the blocks must not overlap.<br><br>**Value:** An array of objects. For example:<br><br>`networking:`<br>`  clusterNetwork:`<br>`  - cidr: 10.128.0.0/14`<br>`    hostPrefix: 23` |
| `networking:`<br>`  clusterNetwork:`<br>`    cidr:` | Required if you use **networking.clusterNetwork**. An IP address block.<br><br>An IPv4 network.<br><br>**Value:** An IP address block in Classless Inter-Domain Routing (CIDR) notation. The prefix length for an IPv4 block is between **0** and **32**. |

| Parameter | Description |
|---|---|
| `networking:`<br>`  clusterNetwork:`<br>`    hostPrefix:` | The subnet prefix length to assign to each individual node. For example, if **hostPrefix** is set to **23** then each node is assigned a **/23** subnet out of the given **cidr**. A **hostPrefix** value of **23** provides 510 (2^(32 - 23) - 2) pod IP addresses.<br><br>**Value:** A subnet prefix.<br><br>The default value is **23**. |
| `networking:`<br>`  serviceNetwork:` | The IP address block for services. The default value is **172.30.0.0/16**.<br><br>The OVN-Kubernetes network plugins supports only a single IP address block for the service network.<br><br>**Value:** An array with an IP address block in CIDR format. For example:<br><br>`networking:`<br>`  serviceNetwork:`<br>`   - 172.30.0.0/16` |
| `networking:`<br>`  machineNetwork:` | The IP address blocks for machines.<br><br>If you specify multiple IP address blocks, the blocks must not overlap.<br><br>**Value:** An array of objects. For example:<br><br>`networking:`<br>`  machineNetwork:`<br>`  - cidr: 10.0.0.0/16` |

| Parameter | Description |
|---|---|
| `networking:`<br>`  machineNetwork:`<br>`    cidr:` | Required if you use **networking.machineNetwork**. An IP address block. The default value is **10.0.0.0/16** for all platforms other than libvirt and IBM Power® Virtual Server. For libvirt, the default value is **192.168.126.0/24**. For IBM Power® Virtual Server, the default value is **192.168.0.0/24**.<br><br>**Value:** An IP network block in CIDR notation.<br><br>For example, **10.0.0.0/16**.<br><br>NOTE<br><br>Set the **networking.machineNetwork** to match the CIDR that the preferred NIC resides in. |
| `networking:`<br>`  ovnKubernetesConfig:`<br>`    ipv4:`<br>`      internalJoinSubnet:` | Configures the IPv4 join subnet that is used internally by **ovn-kubernetes**. This subnet must not overlap with any other subnet that OpenShift Container Platform is using, including the node network. The size of the subnet must be larger than the number of nodes. You cannot change the value after installation.<br><br>**Value:** An IP network block in CIDR notation. The default value is **100.64.0.0/16**. |

## 7.1.3. Optional configuration parameters

Optional installation configuration parameters are described in the following table:

Table 7.3. Optional parameters

| Parameter | Description |
|---|---|
| `additionalTrustBundle:` | A PEM-encoded X.509 certificate bundle that is added to the nodes' trusted certificate store. This trust bundle might also be used when a proxy has been configured.<br><br>**Value:** String |

| Parameter | Description |
|---|---|
| `capabilities:` | Controls the installation of optional core cluster components. You can reduce the footprint of your OpenShift Container Platform cluster by disabling optional components. For more information, see the "Cluster capabilities" page in *Installing*.<br><br>**Value:** String array |
| `capabilities:`<br>`  baselineCapabilitySet:` | Selects an initial set of optional capabilities to enable. Valid values are **None**, **v4.11**, **v4.12** and **vCurrent**. The default value is **vCurrent**.<br><br>**Value:** String |
| `capabilities:`<br>`  additionalEnabledCapabilities:` | Extends the set of optional capabilities beyond what you specify in **baselineCapabilitySet**. You can specify multiple capabilities in this parameter.<br><br>**Value:** String array |
| `cpuPartitioningMode:` | Enables workload partitioning, which isolates OpenShift Container Platform services, cluster management workloads, and infrastructure pods to run on a reserved set of CPUs. You can only enable workload partitioning during installation. You cannot disable it after installation. While this field enables workload partitioning, it does not configure workloads to use specific CPUs. For more information, see the *Workload partitioning* page in the *Scalability and Performance* section.<br><br>**Value:** **None** or **AllNodes**. **None** is the default value. |
| `compute:` | The configuration for the machines that comprise the compute nodes.<br><br>**Value:** Array of **MachinePool** objects. |

| Parameter | Description |
|---|---|
| `compute:`<br>`  architecture:` | Determines the instruction set architecture of the machines in the pool. Currently, clusters with varied architectures are not supported. All pools must specify the same architecture. Valid values are **amd64** and **arm64**.<br><br>Not all installation options support the 64-bit ARM architecture. To verify if your installation option is supported on your platform, see *Supported installation methods for different platforms* in *Selecting a cluster installation method and preparing it for users*.<br><br>**Value:** String |
| `compute:`<br>`  hyperthreading:` | Whether to enable or disable simultaneous multithreading, or **hyperthreading**, on compute machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores.<br><br>IMPORTANT<br><br>If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance.<br><br>**Value:** **Enabled** or **Disabled** |
| `compute:`<br>`  name:` | Required if you use **compute**. The name of the machine pool.<br><br>**Value:** **worker** |
| `compute:`<br>`  platform:` | Required if you use **compute**. Use this parameter to specify the cloud provider to host the worker machines. This parameter value must match the **controlPlane.platform** parameter value.<br><br>**Value:** **aws**, **azure**, **gcp**, **ibmcloud**, **nutanix**, **openstack**, **powervs**, **vsphere**, or **{}** |
| `compute:`<br>`  replicas:` | The number of compute machines, which are also known as worker machines, to provision.<br><br>**Value:** A positive integer greater than or equal to **2**. The default value is **3**. |

| Parameter | Description |
|---|---|
| `featureSet:` | Enables the cluster for a feature set. A feature set is a collection of OpenShift Container Platform features that are not enabled by default. For more information about enabling a feature set during installation, see "Enabling features using feature gates". **Value:** String. The name of the feature set to enable, such as **TechPreviewNoUpgrade**. |
| `controlPlane:` | The configuration for the machines that form the control plane. **Value:** Array of **MachinePool** objects. |
| `controlPlane: architecture:` | Determines the instruction set architecture of the machines in the pool. Currently, clusters with varied architectures are not supported. All pools must specify the same architecture. Valid values are **amd64** and **arm64**. Not all installation options support the 64-bit ARM architecture. To verify if your installation option is supported on your platform, see *Supported installation methods for different platforms* in *Selecting a cluster installation method and preparing it for users*. **Value:** String |
| `controlPlane: hyperthreading:` | Whether to enable or disable simultaneous multithreading, or **hyperthreading**, on control plane machines. By default, simultaneous multithreading is enabled to increase the performance of your machines' cores. **IMPORTANT** If you disable simultaneous multithreading, ensure that your capacity planning accounts for the dramatically decreased machine performance. **Value:** **Enabled** or **Disabled** |
| `controlPlane: name:` | Required if you use **controlPlane**. The name of the machine pool. **Value:** **master** |

| Parameter | Description |
|---|---|
| `controlPlane:`<br>`  platform:` | Required if you use **controlPlane**. Use this parameter to specify the cloud provider that hosts the control plane machines. This parameter value must match the **compute.platform** parameter value.<br><br>**Value:** **aws**, **azure**, **gcp**, **ibmcloud**, **nutanix**, **openstack**, **powervs**, **vsphere**, or **{}** |
| `controlPlane:`<br>`  replicas:` | The number of control plane machines to provision.<br><br>**Value:** Supported values are **3**, or **1** when deploying single-node OpenShift. |
| `credentialsMode:` | The Cloud Credential Operator (CCO) mode. If no mode is specified, the CCO dynamically tries to determine the capabilities of the provided credentials, with a preference for mint mode on the platforms where multiple modes are supported.<br><br>NOTE<br><br>Not all CCO modes are supported for all cloud providers. For more information about CCO modes, see the "Managing cloud provider credentials" entry in the *Authentication and authorization* content.<br><br>**Value:** **Mint**, **Passthrough**, **Manual** or an empty string (**""**). |

| Parameter | Description |
|---|---|
| `fips:` | Enable or disable FIPS mode. The default is **false** (disabled). If you enable FIPS mode, the Red Hat Enterprise Linux CoreOS (RHCOS) machines that OpenShift Container Platform runs on bypass the default Kubernetes cryptography suite and use the cryptography modules that RHCOS provides instead.<br><br>IMPORTANT<br><br>To enable FIPS mode for your cluster, you must run the installation program from a Red Hat Enterprise Linux (RHEL) computer configured to operate in FIPS mode. For more information about configuring FIPS mode on RHEL, see Switching RHEL to FIPS mode.<br><br>When running Red Hat Enterprise Linux (RHEL) or Red Hat Enterprise Linux CoreOS (RHCOS) booted in FIPS mode, OpenShift Container Platform core components use the RHEL cryptographic libraries that have been submitted to NIST for FIPS 140-2/140-3 Validation on only the x86_64, ppc64le, and s390x architectures.<br><br>IMPORTANT<br><br>If you are using Azure File storage, you cannot enable FIPS mode.<br><br>**Value:** **false** or **true** |
| `imageContentSources:` | Sources and repositories for the release-image content.<br><br>**Value:** Array of objects. Includes a**source** and, optionally, **mirrors**, as described in the following rows of this table. |
| `imageContentSources:`<br>`  source:` | Required if you use **imageContentSources**. Specify the repository that users refer to, for example, in image pull specifications.<br><br>**Value:** String |
| `imageContentSources:`<br>`  mirrors:` | Specify one or more repositories that might also contain the same images.<br><br>**Value:** Array of strings |

| Parameter | Description |
|---|---|
| `platform:`<br>`  aws:`<br>`    lbType:` | Required to set the NLB load balancer type in AWS. Valid values are **Classic** or **NLB**. If no value is specified, the installation program defaults to **Classic**. The installation program sets the value provided here in the ingress cluster configuration object. If you do not specify a load balancer type for other Ingress Controllers, they use the type set in this parameter.<br><br>**Value:** **Classic** or **NLB**. The default value is **Classic**. |
| `publish:` | How to publish or expose the user-facing endpoints of your cluster, such as the Kubernetes API, OpenShift routes.<br><br>**Value:** **Internal** or **External**. To deploy a private cluster that cannot be accessed from the internet, set the **publish** parameter to **Internal**. The default value is **External**. |
| `sshKey:` | The SSH key to authenticate access to your cluster machines.<br><br>NOTE<br><br>For production OpenShift Container Platform clusters on which you want to perform installation debugging or disaster recovery, specify an SSH key that your **ssh-agent** process uses.<br><br>**Value:** For example, **sshKey: ssh-ed25519 AAAA..**. |

NOTE

If your AWS account has service control policies (SCP) enabled, you must configure the **credentialsMode** parameter to **Mint**, **Passthrough**, or **Manual**.

IMPORTANT

Setting this parameter to **Manual** enables alternatives to storing administrator-level secrets in the **kube-system** project, which require additional configuration steps. For more information, see "Alternatives to storing administrator-level secrets in the kube-system project".

### 7.1.4. Optional AWS configuration parameters

Optional AWS configuration parameters are described in the following table:

**Table 7.4. Optional AWS parameters**

| Parameter | Description |
|---|---|
| compute:<br>  platform:<br>    aws:<br>      amiID: | The AWS AMI used to boot compute machines for the cluster. This is required for regions that require a custom RHCOS AMI.<br><br>**Value:** Any published or custom RHCOS AMI that belongs to the set AWS region. See *RHCOS AMIs for AWS infrastructure* for available AMI IDs. |
| compute:<br>  platform:<br>    aws:<br>      iamProfile: | The name of the IAM instance profile that you use for the machine. If you want the installation program to create the IAM instance profile for you, do not use the **iamProfile** parameter. You can specify either the **iamProfile** or **iamRole** parameter, but you cannot specify both.<br><br>**Value:** String |
| compute:<br>  platform:<br>    aws:<br>      iamRole: | The name of the IAM instance role that you use for the machine. When you specify an IAM role, the installation program creates an instance profile. If you want the installation program to create the IAM instance role for you, do not select the **iamRole** parameter. You can specify either the **iamRole** or **iamProfile** parameter, but you cannot specify both.<br><br>**Value:** String |
| compute:<br>  platform:<br>    aws:<br>      rootVolume:<br>        iops: | The Input/Output Operations Per Second (IOPS) that is reserved for the root volume.<br><br>**Value:** Integer, for example **4000**. |
| compute:<br>  platform:<br>    aws:<br>      rootVolume:<br>        size: | The size in GiB of the root volume.<br><br>**Value:** Integer, for example **500**. |

| Parameter | Description |
|---|---|
| `compute:`<br>`  platform:`<br>`    aws:`<br>`      rootVolume:`<br>`        type:` | The type of the root volume.<br><br>**Value:** Valid AWS EBS volume type, such as **io1**. |
| `compute:`<br>`  platform:`<br>`    aws:`<br>`      rootVolume:`<br>`        kmsKeyARN:` | The Amazon Resource Name (key ARN) of a KMS key. This is required to encrypt operating system volumes of worker nodes with a specific KMS key.<br><br>**Value:** Valid key ID or the key ARN |
| `compute:`<br>`  platform:`<br>`    aws:`<br>`      type:` | The EC2 instance type for the compute machines.<br><br>**Value:** Valid AWS instance type, such as **m4.2xlarge**. See the "Tested instance types for AWS" table on the "Installing a cluster on AWS with customizations" page. |
| `compute:`<br>`  platform:`<br>`    aws:`<br>`      zones:` | The availability zones where the installation program creates machines for the compute machine pool. If you provide your own VPC, you must provide a subnet in that availability zone.<br><br>**Value:** A list of valid AWS availability zones, such as **us-east-1c**, in a YAML sequence. |
| `controlPlane:`<br>`  platform:`<br>`    aws:`<br>`      amiID:` | The AWS AMI used to boot control plane machines for the cluster. This is required for regions that require a custom RHCOS AMI.<br><br>**Value:** Any published or custom RHCOS AMI that belongs to the set AWS region. See *RHCOS AMIs for AWS infrastructure* for available AMI IDs. |
| `controlPlane:`<br>`  platform:`<br>`    aws:`<br>`      iamProfile:` | The name of the IAM instance profile that you use for the machine. If you want the installation program to create the IAM instance profile for you, do not use the **iamProfile** parameter. You can specify either the **iamProfile** or **iamRole** parameter, but you cannot specify both.<br><br>**Value:** String |

| Parameter | Description |
|---|---|
| `controlPlane: platform: aws: iamRole:` | The name of the IAM instance role that you use for the machine. When you specify an IAM role, the installation program creates an instance profile. If you want the installation program to create the IAM instance role for you, do not use the **iamRole** parameter. You can specify either the **iamRole** or **iamProfile** parameter, but you cannot specify both. **Value:** String |
| `controlPlane: platform: aws: rootVolume: iops:` | The Input/Output Operations Per Second (IOPS) that is reserved for the root volume on control plane machines. **Value:** Integer, for example **4000**. |
| `controlPlane: platform: aws: rootVolume: size:` | The size in GiB of the root volume for control plane machines. **Value:** Integer, for example **500**. |
| `controlPlane: platform: aws: rootVolume: type:` | The type of the root volume for control plane machines. **Value:** Valid AWS EBS volume type, such as **io1**. |
| `controlPlane: platform: aws: rootVolume: kmsKeyARN:` | The Amazon Resource Name (key ARN) of a KMS key. This is required to encrypt operating system volumes of control plane nodes with a specific KMS key. **Value:** Valid key ID and the key ARN. |
| `controlPlane: platform: aws: type:` | The EC2 instance type for the control plane machines. **Value:** Valid AWS instance type, such as **m6i.xlarge**. See the "Tested instance types for AWS" table on the "Installing a cluster on AWS with customizations" page. |

| Parameter | Description |
|---|---|
| <pre>controlPlane:<br>  platform:<br>    aws:<br>      zones:</pre> | The availability zones where the installation program creates machines for the control plane machine pool.<br><br>**Value:** A list of valid AWS availability zones, such as **us-east-1c**, in a YAML sequence. |
| <pre>platform:<br>  aws:<br>    amiID:</pre> | The AWS AMI used to boot all machines for the cluster. If set, the AMI must belong to the same region as the cluster. This is required for regions that require a custom RHCOS AMI.<br><br>**Value:** Any published or custom RHCOS AMI that belongs to the set AWS region. See *RHCOS AMIs for AWS infrastructure* for available AMI IDs. |
| <pre>platform:<br>  aws:<br>    hostedZone:</pre> | An existing Route 53 private hosted zone for the cluster. You can only use a pre-existing hosted zone when also supplying your own VPC. The hosted zone must already be associated with the user-provided VPC before installation. Also, the domain of the hosted zone must be the cluster domain or a parent of the cluster domain. If undefined, the installation program creates a new hosted zone.<br><br>**Value:** String, for example **Z3URY6TWQ91KVV**. |
| <pre>platform:<br>  aws:<br>    hostedZoneRole:</pre> | An Amazon Resource Name (ARN) for an existing IAM role in the account containing the specified hosted zone. The installation program and cluster operators assume this role when performing operations on the hosted zone. Use this parameter only when you are installing a cluster into a shared VPC.<br><br>**Value:** String, for example **arn:aws:iam::1234567890:role/shared-vpc-role**. |

| Parameter | Description |
|---|---|
| `platform:`<br>`  aws:`<br>`    region:` | The AWS region that the installation program creates all cluster resources in.<br><br>**Value:** Any valid AWS region, such as **us-east-1**. You can use the AWS CLI to access the regions available based on your selected instance type by running the following command:<br><br>`$ aws ec2 describe-instance-type-offerings --filters Name=instance-type,Values=c7g.xlarge`<br><br>**IMPORTANT**<br><br>When running on ARM based AWS instances, ensure that you enter a region where AWS Graviton processors are available. See Global availability map in the AWS documentation. Currently, AWS Graviton3 processors are only available in some regions. |
| `platform:`<br>`  aws:`<br>`    serviceEndpoints:`<br>`      - name:`<br>`        url:` | The AWS service endpoint name and URL. Custom endpoints are only required for cases where alternative AWS endpoints, such as FIPS, must be used. Custom API endpoints can be specified for EC2, S3, IAM, Elastic Load Balancing, Tagging, Route 53, and STS AWS services.<br><br>**Value:** Valid AWS service endpoint name and valid AWS service endpoint URL. |
| `platform:`<br>`  aws:`<br>`    userTags:` | A map of keys and values that the installation program adds as tags to all resources that it creates.<br><br>**Value:** Any valid YAML map, such as key value pairs in the **<key>: <value>** format. For more information about AWS tags, see Tagging Your Amazon EC2 Resources in the AWS documentation.<br><br>**NOTE**<br><br>You can add up to 25 user-defined tags during installation. The remaining 25 tags are reserved for OpenShift Container Platform. |

| Parameter | Description |
|---|---|
| `platform:`<br>`  aws:`<br>`    propagateUserTags:` | A flag that directs in-cluster Operators to include the specified user tags in the tags of the AWS resources that the Operators create.<br><br>**Value:** Boolean values, for example **true** or **false**. |
| `platform:`<br>`  aws:`<br>`    publicIpv4Pool:` | The public IPv4 pool ID that is used to allocate Elastic IPs (EIPs) when **publish** is set to **External**. You must provision and advertise the pool in the same AWS account and region of the cluster. You must ensure that you have 2n + 1 IPv4 addresses available in the pool where *n* is the total number of AWS zones used to deploy the Network Load Balancer (NLB) for API, NAT gateways, and bootstrap node. For more information about bring your own IP addresses (BYOIP) in AWS, see Onboard your BYOIP.<br><br>**Value:** A valid public IPv4 pool id<br><br>NOTE<br><br>You can enable BYOIP only for customized installations that do not have any network restrictions. |
| `platform:`<br>`  aws:`<br>`    preserveBootstrapIgnition:` | Prevents the S3 bucket from being deleted after completion of bootstrapping.<br><br>**Value: true** or **false**. The default value is **false**, which results in the S3 bucket being deleted. |

| Parameter | Description |
|---|---|
| `platform:`<br>`  aws:`<br>`    vpc:`<br>`      subnets:` | A list of subnets in an existing VPC to be used in place of automatically created subnets. You specify a subnet by providing the subnet ID and an optional list of roles that apply to that subnet. If you specify subnet IDs but do not specify roles for any subnet, the subnets' roles are decided automatically. If you do not specify any roles, you must ensure that any other subnets in your VPC have the **kubernetes.io/cluster/.: .** or **kubernetes.io/cluster/unmanaged: true** tags.<br><br>The subnets must be part of the same **machineNetwork[].cidr** ranges that you specify.<br><br>For a public cluster, specify a public and a private subnet for each availability zone.<br><br>For a private cluster, specify a private subnet for each availability zone.<br><br>For clusters that use AWS Local Zones, you must add AWS Local Zone subnets to this list to ensure edge machine pool creation.<br><br>**Value:** List of pairs of**id** and **roles** parameters. |
| `platform:`<br>`  aws:`<br>`    vpc:`<br>`      subnets:`<br>`      - id:` | The ID of an existing subnet to be used in place of a subnet created by the installation program.<br><br>**Value:** String. The subnet ID must be a unique ID containing only alphanumeric characters, beginning with "subnet-". The ID must be exactly 24 characters long. |

| Parameter | Description |
|---|---|
| <pre>platform:<br>  aws:<br>    vpc:<br>      subnets:<br>      - id:<br>        roles:<br>        - type:</pre> | One or more roles that apply to the subnet specified by **platform.aws.vpc.subnets.id**. If you specify a role for any subnet, each subnet must have at least one assigned role, and the **ClusterNode**, **IngressControllerLB**, **ControlPlaneExternalLB**, **BootstrapNode**, and **ControlPlaneInternalLB** roles must be assigned to at least one subnet. However, if the cluster scope is internal, then the **ControlPlaneExternalLB** role is not required.<br><br>You can only assign the **EdgeNode** role to subnets in AWS Local Zones.<br><br>**Value:** List of one or more role types. Valid values include **ClusterNode**, **EdgeNode**, **BootstrapNode**, **IngressControllerLB**, **ControlPlaneExternalLB**, and **ControlPlaneInternalLB**. |

# CHAPTER 8. AWS LOCAL ZONE OR WAVELENGTH ZONE TASKS

After installing OpenShift Container Platform on Amazon Web Services (AWS), you can further configure AWS Local Zones or Wavelength Zones and an edge compute pool.

## 8.1. EXTEND EXISTING CLUSTERS TO USE AWS LOCAL ZONES OR WAVELENGTH ZONES

As a post-installation task, you can extend an existing OpenShift Container Platform cluster on Amazon Web Services (AWS) to use AWS Local Zones or Wavelength Zones.

Extending nodes to Local Zones or Wavelength Zones locations comprises the following steps:

- Adjusting the cluster-network maximum transmission unit (MTU).
- Opting in the Local Zones or Wavelength Zones group to AWS Local Zones or Wavelength Zones.
- Creating a subnet in the existing VPC for a Local Zones or Wavelength Zones location.

  **IMPORTANT**

  Before you extend an existing OpenShift Container Platform cluster on AWS to use Local Zones or Wavelength Zones, check that the existing VPC contains available Classless Inter-Domain Routing (CIDR) blocks. These blocks are needed for creating the subnets.

- Creating the machine set manifest, and then creating a node in each Local Zone or Wavelength Zone location.
- Local Zones only: Adding the permission **ec2:ModifyAvailabilityZoneGroup** to the Identity and Access Management (IAM) user or role, so that the required network resources can be created. For example:

  **Example of an additional IAM policy for AWS Local Zones deployments**

  ```
  {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Action": [
          "ec2:ModifyAvailabilityZoneGroup"
        ],
        "Effect": "Allow",
        "Resource": "*"
      }
    ]
  }
  ```

- Wavelength Zone only: Adding the permissions **ec2:ModifyAvailabilityZoneGroup**, **ec2:CreateCarrierGateway**, and **ec2:DeleteCarrierGateway** to the Identity and Access Management (IAM) user or role, so that the required network resources can be created. For example:

**Example of an additional IAM policy for AWS Wavelength Zones deployments**

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DeleteCarrierGateway",
        "ec2:CreateCarrierGateway"
      ],
      "Resource": "*"
    },
    {
      "Action": [
        "ec2:ModifyAvailabilityZoneGroup"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

**Additional resources**

- AWS Local Zones features (AWS documentation)
- AWS Wavelength features (AWS documentation)

### 8.1.1. About edge compute pools

Edge compute nodes are tainted compute nodes that run in AWS Local Zones or Wavelength Zones locations.

When deploying a cluster that uses Local Zones or Wavelength Zones, consider the following points:

- Amazon EC2 instances in the Local Zones or Wavelength Zones are more expensive than Amazon EC2 instances in the Availability Zones.
- The latency is lower between the applications running in AWS Local Zones or Wavelength Zones and the end user. A latency impact exists for some workloads if, for example, ingress traffic is mixed between Local Zones or Wavelength Zones and Availability Zones.

IMPORTANT

Generally, the maximum transmission unit (MTU) between an Amazon EC2 instance in a Local Zones or Wavelength Zones and an Amazon EC2 instance in the Region is 1300. The cluster network MTU must be always less than the EC2 MTU to account for the overhead. The specific overhead is determined by the network plugin. For example: OVN-Kubernetes has an overhead of **100 bytes**.

The network plugin can provide additional features, such as IPsec, that also affect the MTU sizing.

You can access the following resources to learn more about a respective zone type:

- See How Local Zones work in the AWS documentation.
- See How AWS Wavelength work in the AWS documentation.

OpenShift Container Platform 4.12 introduced a new compute pool, *edge*, that is designed for use in remote zones. The edge compute pool configuration is common between AWS Local Zones or Wavelength Zones locations. Because of the type and size limitations of resources like EC2 and EBS on Local Zones or Wavelength Zones resources, the default instance type can vary from the traditional compute pool.

The default Elastic Block Store (EBS) for Local Zones or Wavelength Zones locations is **gp2**, which differs from the non-edge compute pool. The instance type used for each Local Zones or Wavelength Zones on an edge compute pool also might differ from other compute pools, depending on the instance offerings on the zone.

The edge compute pool creates new labels that developers can use to deploy applications onto AWS Local Zones or Wavelength Zones nodes. The new labels are:

- **node-role.kubernetes.io/edge=''**
- Local Zones only: **machine.openshift.io/zone-type=local-zone**
- Wavelength Zones only: **machine.openshift.io/zone-type=wavelength-zone**
- **machine.openshift.io/zone-group=$ZONE_GROUP_NAME**

By default, the machine sets for the edge compute pool define the taint of **NoSchedule** to prevent other workloads from spreading on Local Zones or Wavelength Zones instances. Users can only run user workloads if they define tolerations in the pod specification.

## 8.2. CHANGING THE CLUSTER NETWORK MTU TO SUPPORT LOCAL ZONES OR WAVELENGTH ZONES

You might need to change the maximum transmission unit (MTU) value for the cluster network so that your cluster infrastructure can support Local Zones or Wavelength Zones subnets.

### 8.2.1. About the cluster MTU

During installation, the cluster network MTU is set automatically based on the primary network interface MTU of cluster nodes. You do not usually need to override the detected MTU.

You might want to change the MTU of the cluster network for one of the following reasons:

- The MTU detected during cluster installation is not correct for your infrastructure.
- Your cluster infrastructure now requires a different MTU, such as from the addition of nodes that need a different MTU for optimal performance.

Only the OVN-Kubernetes network plugin supports changing the MTU value.

#### 8.2.1.1. Service interruption considerations

When you initiate a maximum transmission unit (MTU) change on your cluster the following effects might impact service availability:

- At least two rolling reboots are required to complete the migration to a new MTU. During this time, some nodes are not available as they restart.
- Specific applications deployed to the cluster with shorter timeout intervals than the absolute TCP timeout interval might experience disruption during the MTU change.

#### 8.2.1.2. MTU value selection

When planning your maximum transmission unit (MTU) migration there are two related but distinct MTU values to consider.

- **Hardware MTU**: This MTU value is set based on the specifics of your network infrastructure.
- **Cluster network MTU**: This MTU value is always less than your hardware MTU to account for the cluster network overlay overhead. The specific overhead is determined by your network plugin. For OVN-Kubernetes, the overhead is **100** bytes.

If your cluster requires different MTU values for different nodes, you must subtract the overhead value for your network plugin from the lowest MTU value that is used by any node in your cluster. For example, if some nodes in your cluster have an MTU of **9001**, and some have an MTU of **1500**, you must set this value to **1400**.

IMPORTANT

To avoid selecting an MTU value that is not acceptable by a node, verify the maximum MTU value (**maxmtu**) that is accepted by the network interface by using the **ip -d link** command.

#### 8.2.1.3. How the migration process works

The following table summarizes the migration process by segmenting between the user-initiated steps in the process and the actions that the migration performs in response.

**Table 8.1. Live migration of the cluster MTU**

| User-initiated steps | OpenShift Container Platform activity |
|---|---|
| Set the following values in the Cluster Network Operator configuration:<br>• **spec.migration.mtu.machine.to**<br>• **spec.migration.mtu.network.from**<br>• **spec.migration.mtu.network.to** | **Cluster Network Operator (CNO)**: Confirms that each field is set to a valid value.<br>• The **mtu.machine.to** must be set to either the new hardware MTU or to the current hardware MTU if the MTU for the hardware is not changing. This value is transient and is used as part of the migration process. If you set a hardware MTU different from the current value, you must manually configure it to persist. Use methods such as a machine config, DHCP setting, or kernel command line.<br>• The **mtu.network.from** field must equal the **network.status.clusterNetworkMTU** field, which is the current MTU of the cluster network.<br>• The **mtu.network.to** field must be set to the target cluster network MTU. It must be lower than the hardware MTU to allow for the overlay overhead of the network plugin. For OVN-Kubernetes, the overhead is **100** bytes.<br><br>If the values provided are valid, the CNO writes out a new temporary configuration with the MTU for the cluster network set to the value of the **mtu.network.to** field.<br><br>**Machine Config Operator (MCO)**: Performs a rolling reboot of each node in the cluster. |
| Reconfigure the MTU of the primary network interface for the nodes on the cluster. You can use one of the following methods to accomplish this:<br>• Deploying a new NetworkManager connection profile with the MTU change<br>• Changing the MTU through a DHCP server setting<br>• Changing the MTU through boot parameters | N/A |
| Set the **mtu** value in the CNO configuration for the network plugin and set **spec.migration** to **null**. | **Machine Config Operator (MCO)**: Performs a rolling reboot of each node in the cluster with the new MTU configuration. |

### 8.2.2. Changing the cluster network MTU

As a cluster administrator, you can increase or decrease the maximum transmission unit (MTU) for your cluster.

IMPORTANT

You cannot roll back an MTU value for nodes during the MTU migration process, but you can roll back the value after the MTU migration process completes.

The migration is disruptive and nodes in your cluster might be temporarily unavailable as the MTU update takes effect.

**Prerequisites**

- You have installed the OpenShift CLI (**oc**).
- You have access to the cluster using an account with **cluster-admin** permissions.
- You have identified the target MTU for your cluster. The MTU for the OVN-Kubernetes network plugin must be set to **100** less than the lowest hardware MTU value in your cluster.
- If your nodes are physical machines, ensure that the cluster network and the connected network switches support jumbo frames.
- If your nodes are virtual machines (VMs), ensure that the hypervisor and the connected network switches support jumbo frames.

#### 8.2.2.1. Checking the current cluster MTU value

Use the following procedure to obtain the current maximum transmission unit (MTU) for the cluster network.

**Procedure**

- To obtain the current MTU for the cluster network, enter the following command:

```
$ oc describe network.config cluster
```

**Example output**

```
...
Status:
  Cluster Network:
    Cidr:              10.217.0.0/22
    Host Prefix:       23
  Cluster Network MTU:  1400
  Network Type:         OVNKubernetes
  Service Network:
    10.217.4.0/23
...
```

#### 8.2.2.2. Beginning the MTU migration

Use the following procedure to start the MTU migration.

### Procedure

1. To begin the MTU migration, specify the migration configuration by entering the following command. The Machine Config Operator performs a rolling reboot of the nodes in the cluster in preparation for the MTU change.

   ```
   $ oc patch Network.operator.openshift.io cluster --type=merge --patch \
     '{"spec": { "migration": { "mtu": { "network": { "from": <overlay_from>, "to": <overlay_to> } ,
   "machine": { "to" : <machine_to> } } } } }'
   ```

   where:

   **<overlay_from>**

   Specifies the current cluster network MTU value.

   **<overlay_to>**

   Specifies the target MTU for the cluster network. This value is set relative to the value of **<machine_to>**. For OVN-Kubernetes, this value must be **100** less than the value of **<machine_to>**.

   **<machine_to>**

   Specifies the MTU for the primary network interface on the underlying host network.

   ```
   $ oc patch Network.operator.openshift.io cluster --type=merge --patch \
     '{"spec": { "migration": { "mtu": { "network": { "from": 1400, "to": 9000 } , "machine": { "to" :
   9100} } } } }'
   ```

2. As the Machine Config Operator updates machines in each machine config pool, the Operator reboots each node one by one. You must wait until all the nodes are updated. Check the machine config pool status by entering the following command:

   ```
   $ oc get machineconfigpools
   ```

   A successfully updated node has the following status: **UPDATED=true**, **UPDATING=false**, **DEGRADED=false**.

   **NOTE**

   By default, the Machine Config Operator updates one machine per pool at a time, causing the total time the migration takes to increase with the size of the cluster.

#### 8.2.2.3. Verifying the machine configuration

Use the following procedure to verify the machine configuration.

### Procedure

- Confirm the status of the new machine configuration on the hosts:

  a. To list the machine configuration state and the name of the applied machine configuration, enter the following command:

     ```
     $ oc describe node | egrep "hostname|machineconfig"
     ```

**Example output**

```
kubernetes.io/hostname=master-0
machineconfiguration.openshift.io/currentConfig: rendered-master-
c53e221d9d24e1c8bb6ee89dd3d8ad7b
machineconfiguration.openshift.io/desiredConfig: rendered-master-
c53e221d9d24e1c8bb6ee89dd3d8ad7b
machineconfiguration.openshift.io/reason:
machineconfiguration.openshift.io/state: Done
```

b. Verify that the following statements are true:

- The value of **machineconfiguration.openshift.io/state** field is **Done**.
- The value of the **machineconfiguration.openshift.io/currentConfig** field is equal to the value of the **machineconfiguration.openshift.io/desiredConfig** field.

c. To confirm that the machine config is correct, enter the following command:

```
$ oc get machineconfig <config_name> -o yaml | grep ExecStart
```

where:

**<config_name>**

Specifies the name of the machine config from the **machineconfiguration.openshift.io/currentConfig** field.

The machine config must include the following update to the systemd configuration:

```
ExecStart=/usr/local/bin/mtu-migration.sh
```

### 8.2.2.4. Finalizing the MTU migration

Use the following procedure to finalize the MTU migration.

**Procedure**

1. To finalize the MTU migration, enter the following command for the OVN-Kubernetes network plugin:

```
$ oc patch Network.operator.openshift.io cluster --type=merge --patch \
  '{"spec": { "migration": null, "defaultNetwork":{ "ovnKubernetesConfig": { "mtu": <mtu> }}}}'
```

where:

**<mtu>**

Specifies the new cluster network MTU that you specified with **<overlay_to>**.

2. After finalizing the MTU migration, each machine config pool node is rebooted one by one. You must wait until all the nodes are updated. Check the machine config pool status by entering the following command:

```
$ oc get machineconfigpools
```

A successfully updated node has the following status: **UPDATED=true**, **UPDATING=false**, **DEGRADED=false**.

#### Verification

- Verify that the node in your cluster uses the MTU that you specified by entering the following command:

```
$ oc describe network.config cluster
```

### 8.2.3. Opting in to AWS Local Zones or Wavelength Zones

If you plan to create subnets in AWS Local Zones or Wavelength Zones, you must opt in to each zone group separately.

#### Prerequisites

- You have installed the AWS CLI.
- You have determined an AWS Region for where you want to deploy your OpenShift Container Platform cluster.
- You have attached a permissive IAM policy to a user or role account that opts in to the zone group.

#### Procedure

1. List the zones that are available in your AWS Region by running the following command:

**Example command for listing available AWS Local Zones in an AWS Region**

```
$ aws --region "<value_of_AWS_Region>" ec2 describe-availability-zones \
    --query 'AvailabilityZones[].[{ZoneName: ZoneName, GroupName: GroupName, Status: OptInStatus}]' \
    --filters Name=zone-type,Values=local-zone \
    --all-availability-zones
```

**Example command for listing available AWS Wavelength Zones in an AWS Region**

```
$ aws --region "<value_of_AWS_Region>" ec2 describe-availability-zones \
    --query 'AvailabilityZones[].[{ZoneName: ZoneName, GroupName: GroupName, Status: OptInStatus}]' \
    --filters Name=zone-type,Values=wavelength-zone \
    --all-availability-zones
```

Depending on the AWS Region, the list of available zones might be long. The command returns the following fields:

**ZoneName**
: The name of the Local Zones or Wavelength Zones.

**GroupName**
: The group that comprises the zone. To opt in to the Region, save the name.

**Status**

The status of the Local Zones or Wavelength Zones group. If the status is **not-opted-in**, you must opt in the **GroupName** as described in the next step.

2. Opt in to the zone group on your AWS account by running the following command:

```
$ aws ec2 modify-availability-zone-group \
  --group-name "<value_of_GroupName>" \ 1
  --opt-in-status opted-in
```

1 Replace **<value_of_GroupName>** with the name of the group of the Local Zones or Wavelength Zones where you want to create subnets.

### 8.2.4. Create network requirements in an existing VPC that uses AWS Local Zones or Wavelength Zones

If you want a Machine API to create an Amazon EC2 instance in a remote zone location, you must create a subnet in a Local Zones or Wavelength Zones location. You can use any provisioning tool, such as Ansible or Terraform, to create subnets in the existing Virtual Private Cloud (VPC).

You can configure the CloudFormation template to meet your requirements. The following subsections include steps that use CloudFormation templates to create the network requirements that extend an existing VPC to use an AWS Local Zones or Wavelength Zones.

Extending nodes to Local Zones requires that you create the following resources:

- 2 VPC Subnets: public and private. The public subnet associates to the public route table for the regular Availability Zones in the Region. The private subnet associates to the provided route table ID.

Extending nodes to Wavelength Zones requires that you create the following resources:

- 1 VPC Carrier Gateway associated to the provided VPC ID.
- 1 VPC Route Table for Wavelength Zones with a default route entry to VPC Carrier Gateway.
- 2 VPC Subnets: public and private. The public subnet associates to the public route table for an AWS Wavelength Zone. The private subnet associates to the provided route table ID.

IMPORTANT

Considering the limitation of NAT Gateways in Wavelength Zones, the provided CloudFormation templates support only associating the private subnets with the provided route table ID. A route table ID is attached to a valid NAT Gateway in the AWS Region.

### 8.2.5. Wavelength Zones only: Creating a VPC carrier gateway

To use public subnets in your OpenShift Container Platform cluster that runs on Wavelength Zones, you must create the carrier gateway and associate the carrier gateway to the VPC. Subnets are useful for deploying load balancers or edge compute nodes.

To create edge nodes or internet-facing load balancers in Wavelength Zones locations for your OpenShift Container Platform cluster, you must create the following required network components:

- A carrier gateway that associates to the existing VPC.
- A carrier route table that lists route entries.
- A subnet that associates to the carrier route table.

Carrier gateways exist for VPCs that only contain subnets in a Wavelength Zone.

The following list explains the functions of a carrier gateway in the context of an AWS Wavelength Zones location:

- Provides connectivity between your Wavelength Zone and the carrier network, which includes any available devices from the carrier network.
- Performs Network Address Translation (NAT) functions, such as translating IP addresses that are public IP addresses stored in a network border group, from Wavelength Zones to carrier IP addresses. These translation functions apply to inbound and outbound traffic.
- Authorizes inbound traffic from a carrier network that is located in a specific location.
- Authorizes outbound traffic to a carrier network and the internet.

NOTE

No inbound connection configuration exists from the internet to a Wavelength Zone through the carrier gateway.

You can use the provided CloudFormation template to create a stack of the following AWS resources:

- One carrier gateway that associates to the VPC ID in the template.
- One public route table for the Wavelength Zone named as **<ClusterName>-public-carrier**.
- Default IPv4 route entry in the new route table that targets the carrier gateway.
- VPC gateway endpoint for an AWS Simple Storage Service (S3).

NOTE

If you do not use the provided CloudFormation template to create your AWS infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

**Prerequisites**

- You configured an AWS account.
- You added your AWS keys and region to your local AWS profile by running **aws configure**.

**Procedure**

1. Go to the next section of the documentation named "CloudFormation template for the VPC Carrier Gateway", and then copy the syntax from the **CloudFormation template for VPC Carrier Gateway** template. Save the copied template syntax as a YAML file on your local

system. This template describes the VPC that your cluster requires.

2. Run the following command to deploy the CloudFormation template, which creates a stack of AWS resources that represent the VPC:

```
$ aws cloudformation create-stack --stack-name <stack_name> \ 1
  --region ${CLUSTER_REGION} \
  --template-body file://<template>.yaml \ 2
  --parameters \//
   ParameterKey=VpcId,ParameterValue="${VpcId}" \ 3
   ParameterKey=ClusterName,ParameterValue="${ClusterName}" 4
```

1. **<stack_name>** is the name for the CloudFormation stack, such as **clusterName-vpc-carrier-gw**. You need the name of this stack if you remove the cluster.

2. **<template>** is the relative path and the name of the CloudFormation template YAML file that you saved.

3. **<VpcId>** is the VPC ID extracted from the CloudFormation stack output created in the section named "Creating a VPC in AWS".

4. **<ClusterName>** is a custom value that prefixes to resources that the CloudFormation stack creates. You can use the same name that is defined in the **metadata.name** section of the **install-config.yaml** configuration file.

#### Example output

```
arn:aws:cloudformation:us-east-1:123456789012:stack/<stack_name>/dbedae40-2fd3-11eb-
820e-12a48460849f
```

### Verification

- Confirm that the CloudFormation template components exist by running the following command:

```
$ aws cloudformation describe-stacks --stack-name <stack_name>
```

After the **StackStatus** displays **CREATE_COMPLETE**, the output displays values for the following parameter. Ensure that you provide the parameter value to the other CloudFormation templates that you run to create for your cluster.

| | |
|---|---|
| **PublicRouteTableId** | The ID of the Route Table in the Carrier infrastructure. |

### 8.2.6. Wavelength Zones only: CloudFormation template for the VPC Carrier Gateway

You can use the following CloudFormation template to deploy the Carrier Gateway on AWS Wavelength infrastructure.

**Example 8.1. CloudFormation template for VPC Carrier Gateway**

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for Creating Wavelength Zone Gateway (Carrier Gateway).

Parameters:
  VpcId:
    Description: VPC ID to associate the Carrier Gateway.
    Type: String
    AllowedPattern: ^(?:(?:vpc)(?:-[a-zA-Z0-9]+)?\b|(?:[0-9]{1,3}\.){3}[0-9]{1,3})$
    ConstraintDescription: VPC ID must be with valid name, starting with vpc-.*.
  ClusterName:
    Description: Cluster Name or Prefix name to prepend the tag Name for each subnet.
    Type: String
    AllowedPattern: ".+"
    ConstraintDescription: ClusterName parameter must be specified.

Resources:
  CarrierGateway:
    Type: "AWS::EC2::CarrierGateway"
    Properties:
      VpcId: !Ref VpcId
      Tags:
      - Key: Name
        Value: !Join ['-', [!Ref ClusterName, "cagw"]]

  PublicRouteTable:
    Type: "AWS::EC2::RouteTable"
    Properties:
      VpcId: !Ref VpcId
      Tags:
      - Key: Name
        Value: !Join ['-', [!Ref ClusterName, "public-carrier"]]

  PublicRoute:
    Type: "AWS::EC2::Route"
    DependsOn: CarrierGateway
    Properties:
      RouteTableId: !Ref PublicRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      CarrierGatewayId: !Ref CarrierGateway

  S3Endpoint:
    Type: AWS::EC2::VPCEndpoint
    Properties:
      PolicyDocument:
        Version: 2012-10-17
        Statement:
        - Effect: Allow
          Principal: '*'
          Action:
          - '*'
          Resource:
          - '*'
      RouteTableIds:
      - !Ref PublicRouteTable
      ServiceName: !Join
      - ''
```

```
          - - com.amazonaws.
            - !Ref 'AWS::Region'
            - .s3
          VpcId: !Ref VpcId

Outputs:
  PublicRouteTableId:
    Description: Public Route table ID
    Value: !Ref PublicRouteTable
```

### 8.2.7. Creating subnets for AWS edge compute services

Before you configure a machine set for edge compute nodes in your OpenShift Container Platform cluster, you must create a subnet in Local Zones or Wavelength Zones. Complete the following procedure for each Wavelength Zone that you want to deploy compute nodes to.

You can use the provided CloudFormation template and create a CloudFormation stack. You can then use this stack to custom provision a subnet.

**NOTE**

If you do not use the provided CloudFormation template to create your AWS infrastructure, you must review the provided information and manually create the infrastructure. If your cluster does not initialize correctly, you might have to contact Red Hat support with your installation logs.

**Prerequisites**

- You configured an AWS account.
- You added your AWS keys and region to your local AWS profile by running **aws configure**.
- You opted in to the Local Zones or Wavelength Zones group.

**Procedure**

1. Go to the section of the documentation named "CloudFormation template for the VPC subnet", and copy the syntax from the template. Save the copied template syntax as a YAML file on your local system. This template describes the VPC that your cluster requires.

2. Run the following command to deploy the CloudFormation template, which creates a stack of AWS resources that represent the VPC:

```
$ aws cloudformation create-stack --stack-name <stack_name> \ (1)
  --region ${CLUSTER_REGION} \
  --template-body file://<template>.yaml \ (2)
  --parameters \
    ParameterKey=VpcId,ParameterValue="${VPC_ID}" \ (3)
    ParameterKey=ClusterName,ParameterValue="${CLUSTER_NAME}" \ (4)
    ParameterKey=ZoneName,ParameterValue="${ZONE_NAME}" \ (5)
    ParameterKey=PublicRouteTableId,ParameterValue="${ROUTE_TABLE_PUB}" \ (6)
```

```
ParameterKey=PublicSubnetCidr,ParameterValue="${SUBNET_CIDR_PUB}" \ 7
ParameterKey=PrivateRouteTableId,ParameterValue="${ROUTE_TABLE_PVT}" \ 8
ParameterKey=PrivateSubnetCidr,ParameterValue="${SUBNET_CIDR_PVT}" 9
```

1. **<stack_name>** is the name for the CloudFormation stack, such as **cluster-wl-<local_zone_shortname>** for Local Zones and **cluster-wl-<wavelength_zone_shortname>** for Wavelength Zones. You need the name of this stack if you remove the cluster.
2. **<template>** is the relative path and the name of the CloudFormation template YAML file that you saved.
3. **${VPC_ID}** is the VPC ID, which is the value **VpcID** in the output of the CloudFormation template for the VPC.
4. **${CLUSTER_NAME}** is the value of **ClusterName** to be used as a prefix of the new AWS resource names.
5. **${ZONE_NAME}** is the value of Local Zones or Wavelength Zones name to create the subnets.
6. **${ROUTE_TABLE_PUB}** is the Public Route Table Id extracted from the CloudFormation template. For Local Zones, the public route table is extracted from the VPC CloudFormation Stack. For Wavelength Zones, the value must be extracted from the output of the VPC's carrier gateway CloudFormation stack.
7. **${SUBNET_CIDR_PUB}** is a valid CIDR block that is used to create the public subnet. This block must be part of the VPC CIDR block **VpcCidr**.
8. **${ROUTE_TABLE_PVT}** is the **PrivateRouteTableId** extracted from the output of the VPC's CloudFormation stack.
9. **${SUBNET_CIDR_PVT}** is a valid CIDR block that is used to create the private subnet. This block must be part of the VPC CIDR block **VpcCidr**.

#### Example output

```
arn:aws:cloudformation:us-east-1:123456789012:stack/<stack_name>/dbedae40-820e-
11eb-2fd3-12a48460849f
```

### Verification

- Confirm that the template components exist by running the following command:

```
$ aws cloudformation describe-stacks --stack-name <stack_name>
```

After the **StackStatus** displays **CREATE_COMPLETE**, the output displays values for the following parameters:

| | |
|---|---|
| **PublicSubnetId** | The IDs of the public subnet created by the CloudFormation stack. |

| | |
|---|---|
| **PrivateSubnetId** | The IDs of the private subnet created by the CloudFormation stack. |

Ensure that you provide these parameter values to the other CloudFormation templates that you run to create for your cluster.

## 8.2.8. CloudFormation template for the VPC subnet

You can use the following CloudFormation template to deploy the private and public subnets in a zone on Local Zones or Wavelength Zones infrastructure.

**Example 8.2. CloudFormation template for VPC subnets**

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for Best Practice Subnets (Public and Private)

Parameters:
  VpcId:
    Description: VPC ID that comprises all the target subnets.
    Type: String
    AllowedPattern: ^(?:(?:vpc)(?:-[a-zA-Z0-9]+)?\b|(?:[0-9]{1,3}\.){3}[0-9]{1,3})$
    ConstraintDescription: VPC ID must be with valid name, starting with vpc-.*.
  ClusterName:
    Description: Cluster name or prefix name to prepend the Name tag for each subnet.
    Type: String
    AllowedPattern: ".+"
    ConstraintDescription: ClusterName parameter must be specified.
  ZoneName:
    Description: Zone Name to create the subnets, such as us-west-2-lax-1a.
    Type: String
    AllowedPattern: ".+"
    ConstraintDescription: ZoneName parameter must be specified.
  PublicRouteTableId:
    Description: Public Route Table ID to associate the public subnet.
    Type: String
    AllowedPattern: ".+"
    ConstraintDescription: PublicRouteTableId parameter must be specified.
  PublicSubnetCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])(\/(1[6-9]|2[0-4]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/16-24.
    Default: 10.0.128.0/20
    Description: CIDR block for public subnet.
    Type: String
  PrivateRouteTableId:
    Description: Private Route Table ID to associate the private subnet.
    Type: String
    AllowedPattern: ".+"
    ConstraintDescription: PrivateRouteTableId parameter must be specified.
  PrivateSubnetCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])(\/(1[6-9]|2[0-4]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/16-24.
    Default: 10.0.128.0/20
    Description: CIDR block for private subnet.
    Type: String


Resources:
  PublicSubnet:
    Type: "AWS::EC2::Subnet"
    Properties:
      VpcId: !Ref VpcId
      CidrBlock: !Ref PublicSubnetCidr
      AvailabilityZone: !Ref ZoneName
      Tags:
      - Key: Name
        Value: !Join ['-', [!Ref ClusterName, "public", !Ref ZoneName]]

  PublicSubnetRouteTableAssociation:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Properties:
      SubnetId: !Ref PublicSubnet
      RouteTableId: !Ref PublicRouteTableId

  PrivateSubnet:
    Type: "AWS::EC2::Subnet"
    Properties:
      VpcId: !Ref VpcId
      CidrBlock: !Ref PrivateSubnetCidr
      AvailabilityZone: !Ref ZoneName
      Tags:
      - Key: Name
        Value: !Join ['-', [!Ref ClusterName, "private", !Ref ZoneName]]

  PrivateSubnetRouteTableAssociation:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Properties:
      SubnetId: !Ref PrivateSubnet
      RouteTableId: !Ref PrivateRouteTableId

Outputs:
  PublicSubnetId:
    Description: Subnet ID of the public subnets.
    Value:
      !Join ["", [!Ref PublicSubnet]]

  PrivateSubnetId:
    Description: Subnet ID of the private subnets.
    Value:
      !Join ["", [!Ref PrivateSubnet]]
```

### 8.2.9. Creating a machine set manifest for an AWS Local Zones or Wavelength Zones node

After you create subnets in AWS Local Zones or Wavelength Zones, you can create a machine set manifest.

The installation program sets the following labels for the **edge** machine pools at cluster installation time:

- **machine.openshift.io/parent-zone-name: <value_of_ParentZoneName>**
- **machine.openshift.io/zone-group: <value_of_ZoneGroup>**
- **machine.openshift.io/zone-type: <value_of_ZoneType>**

The following procedure details how you can create a machine set configuraton that matches the **edge** compute pool configuration.

### Prerequisites

- You have created subnets in AWS Local Zones or Wavelength Zones.

### Procedure

- Manually preserve **edge** machine pool labels when creating the machine set manifest by gathering the AWS API. To complete this action, enter the following command in your command-line interface (CLI):

```
$ aws ec2 describe-availability-zones --region <value_of_Region> \ (1)
    --query 'AvailabilityZones[].{
 ZoneName: ZoneName,
 ParentZoneName: ParentZoneName,
 GroupName: GroupName,
 ZoneType: ZoneType}' \
    --filters Name=zone-name,Values=<value_of_ZoneName> \ (2)
    --all-availability-zones
```

(1) For **<value_of_Region>**, specify the name of the region for the zone.

(2) For **<value_of_ZoneName>**, specify the name of the Local Zones or Wavelength Zones.

#### Example output for Local Zone **us-east-1-nyc-1a**

```
[
    {
        "ZoneName": "us-east-1-nyc-1a",
        "ParentZoneName": "us-east-1f",
        "GroupName": "us-east-1-nyc-1",
        "ZoneType": "local-zone"
    }
]
```

#### Example output for Wavelength Zone **us-east-1-wl1**

```
[
    {
        "ZoneName": "us-east-1-wl1-bos-wlz-1",
        "ParentZoneName": "us-east-1a",
        "GroupName": "us-east-1-wl1",
        "ZoneType": "wavelength-zone"
    }
]
```

### 8.2.9.1. Sample YAML for a compute machine set custom resource on AWS

This sample YAML defines a compute machine set that runs in the **us-east-1-nyc-1a** Amazon Web Services (AWS) zone and creates nodes that are labeled with **node-role.kubernetes.io/edge: ""**.

NOTE

If you want to reference the sample YAML file in the context of Wavelength Zones, ensure that you replace the AWS Region and zone information with supported Wavelength Zone values.

In this sample, **<infrastructure_id>** is the infrastructure ID label that is based on the cluster ID that you set when you provisioned the cluster, and **<edge>** is the node label to add.

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
  name: <infrastructure_id>-edge-<zone> 2
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-edge-<zone>
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id>
        machine.openshift.io/cluster-api-machine-role: edge 3
        machine.openshift.io/cluster-api-machine-type: edge
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-edge-<zone>
    spec:
      metadata:
        labels:
          machine.openshift.io/parent-zone-name: <value_of_ParentZoneName>
          machine.openshift.io/zone-group: <value_of_GroupName>
          machine.openshift.io/zone-type: <value_of_ZoneType>
          node-role.kubernetes.io/edge: ""
      providerSpec:
        value:
          ami:
            id: ami-046fe691f52a953f9 4
          apiVersion: machine.openshift.io/v1beta1
          blockDevices:
            - ebs:
                iops: 0
                volumeSize: 120
```

```
           volumeType: gp2
        credentialsSecret:
          name: aws-cloud-credentials
        deviceIndex: 0
        iamInstanceProfile:
          id: <infrastructure_id>-worker-profile
        instanceType: m6i.large
        kind: AWSMachineProviderConfig
        placement:
          availabilityZone: <zone> 5
          region: <region> 6
        securityGroups:
          - filters:
            - name: tag:Name
              values:
                - <infrastructure_id>-node
          - filters:
            - name: tag:Name
              values:
                - <infrastructure_id>-lb
        subnet:
            id: <value_of_PublicSubnetIds> 7
        publicIp: true
        tags:
          - name: kubernetes.io/cluster/<infrastructure_id>
            value: owned
          - name: <custom_tag_name> 8
            value: <custom_tag_value>
        userDataSecret:
          name: worker-user-data
      taints: 9
       - key: node-role.kubernetes.io/edge
        effect: NoSchedule
```

1. Specify the infrastructure ID that is based on the cluster ID that you set when you provisioned the cluster. If you have the OpenShift CLI installed, you can obtain the infrastructure ID by running the following command:

    ```
    $ oc get -o jsonpath='{.status.infrastructureName}{"\n"}' infrastructure cluster
    ```

2. Specify the infrastructure ID, **edge** role node label, and zone name.

3. Specify the **edge** role node label.

4. Specify a valid Red Hat Enterprise Linux CoreOS (RHCOS) Amazon Machine Image (AMI) for your AWS zone for your OpenShift Container Platform nodes. If you want to use an AWS Marketplace image, you must complete the OpenShift Container Platform subscription from the AWS Marketplace to obtain an AMI ID for your region.

    ```
    $ oc -n openshift-machine-api \
        -o jsonpath='{.spec.template.spec.providerSpec.value.ami.id}{"\n"}' \
        get machineset/<infrastructure_id>-<role>-<zone>
    ```

5. Specify the zone name, for example, **us-east-1-nyc-1a**.

6 Specify the region, for example, **us-east-1**.

7 The ID of the public subnet that you created in AWS Local Zones or Wavelength Zones. You created this public subnet ID when you finished the procedure for "Creating a subnet in an AWS zone".

8 Optional: Specify custom tag data for your cluster. For example, you might add an admin contact email address by specifying a **name:value** pair of **Email:admin-email@example.com**.

NOTE

Custom tags can also be specified during installation in the **install-config.yml** file. If the **install-config.yml** file and the machine set include a tag with the same **name** data, the value for the tag from the machine set takes priority over the value for the tag in the **install-config.yml** file.

9 Specify a taint to prevent user workloads from being scheduled on **edge** nodes.

NOTE

After adding the **NoSchedule** taint on the infrastructure node, existing DNS pods running on that node are marked as **misscheduled**. You must either delete or add toleration on **misscheduled** DNS pods.

### 8.2.9.2. Creating a compute machine set

In addition to the compute machine sets created by the installation program, you can create your own to dynamically manage the machine compute resources for specific workloads of your choice.

#### Prerequisites

- Deploy an OpenShift Container Platform cluster.
- Install the OpenShift CLI (**oc**).
- Log in to **oc** as a user with **cluster-admin** permission.

#### Procedure

1. Create a new YAML file that contains the compute machine set custom resource (CR) sample and is named **<file_name>.yaml**.
   Ensure that you set the **<clusterID>** and **<role>** parameter values.
2. Optional: If you are not sure which value to set for a specific field, you can check an existing compute machine set from your cluster.
   a. To list the compute machine sets in your cluster, run the following command:

```
$ oc get machinesets -n openshift-machine-api
```

#### Example output

```
NAME                                DESIRED   CURRENT   READY   AVAILABLE   AGE
agl030519-vplxk-worker-us-east-1a   1         1         1       1           55m
agl030519-vplxk-worker-us-east-1b   1         1         1       1           55m
agl030519-vplxk-worker-us-east-1c   1         1         1       1           55m
agl030519-vplxk-worker-us-east-1d   0         0                             55m
agl030519-vplxk-worker-us-east-1e   0         0                             55m
agl030519-vplxk-worker-us-east-1f   0         0                             55m
```

b. To view values of a specific compute machine set custom resource (CR), run the following command:

```
$ oc get machineset <machineset_name> \
  -n openshift-machine-api -o yaml
```

**Example output**

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
  name: <infrastructure_id>-<role> 2
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id>
        machine.openshift.io/cluster-api-machine-role: <role>
        machine.openshift.io/cluster-api-machine-type: <role>
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
    spec:
      providerSpec: 3
        ...
```

1 The cluster infrastructure ID.

2 A default node label.

**NOTE**

For clusters that have user-provisioned infrastructure, a compute machine set can only create **worker** and **infra** type machines.

3 The values in the **<providerSpec>** section of the compute machine set CR are platform-specific. For more information about **<providerSpec>** parameters in the CR, see the sample compute machine set CR configuration for your provider.

3. Create a **MachineSet** CR by running the following command:

```
$ oc create -f <file_name>.yaml
```

### Verification

- View the list of compute machine sets by running the following command:

```
$ oc get machineset -n openshift-machine-api
```

**Example output**

```
NAME                                    DESIRED   CURRENT   READY   AVAILABLE   AGE
agl030519-vplxk-edge-us-east-1-nyc-1a      1         1        1       1          11m
agl030519-vplxk-worker-us-east-1a        1         1        1       1          55m
agl030519-vplxk-worker-us-east-1b        1         1        1       1          55m
agl030519-vplxk-worker-us-east-1c        1         1        1       1          55m
agl030519-vplxk-worker-us-east-1d        0         0                           55m
agl030519-vplxk-worker-us-east-1e        0         0                           55m
agl030519-vplxk-worker-us-east-1f        0         0                           55m
```

When the new compute machine set is available, the **DESIRED** and **CURRENT** values match. If the compute machine set is not available, wait a few minutes and run the command again.

- Optional: To check nodes that were created by the edge machine, run the following command:

```
$ oc get nodes -l node-role.kubernetes.io/edge
```

**Example output**

```
NAME                          STATUS   ROLES         AGE    VERSION
ip-10-0-207-188.ec2.internal   Ready    edge,worker   172m   v1.25.2+d2e245f
```

### Additional resources

- Installing a cluster on AWS with compute nodes on AWS Local Zones
- Installing a cluster on AWS with compute nodes on AWS Wavelength Zones

## 8.3. CREATING USER WORKLOADS IN AWS LOCAL ZONES OR WAVELENGTH ZONES

After you create an Amazon Web Service (AWS) Local Zones or Wavelength Zones infrastructure and deploy your cluster, you can use edge compute nodes to create user workloads in Local Zones or Wavelength Zones subnets.

When you use the installation program to create a cluster, the installation program automatically specifies a taint effect of **NoSchedule** to each edge compute node. This means that a scheduler does not add a new pod, or deployment, to a node if the pod does not match the specified tolerations for a taint. You can modify the taint for better control over how nodes create workloads in each Local Zones or Wavelength Zones subnet.

The installation program creates the compute machine set manifests file with **node-role.kubernetes.io/edge** and **node-role.kubernetes.io/worker** labels applied to each edge compute node that is located in a Local Zones or Wavelength Zones subnet.

NOTE

The examples in the procedure are for a Local Zones infrastructure. If you are working with a Wavelength Zones infrastructure, ensure you adapt the examples to what is supported in this infrastructure.

### Prerequisites

- You have access to the OpenShift CLI (**oc**).
- You deployed your cluster in a Virtual Private Cloud (VPC) with defined Local Zones or Wavelength Zones subnets.
- You ensured that the compute machine set for the edge compute nodes on Local Zones or Wavelength Zones subnets specifies the taints for **node-role.kubernetes.io/edge**.

### Procedure

1. Create a **deployment** resource YAML file for an example application to be deployed in the edge compute node that operates in a Local Zones subnet. Ensure that you specify the correct tolerations that match the taints for the edge compute node.

#### Example of a configured **deployment** resource for an edge compute node that operates in a Local Zone subnet

```
kind: Namespace
apiVersion: v1
metadata:
  name: <local_zone_application_namespace>
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <pvc_name>
  namespace: <local_zone_application_namespace>
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  storageClassName: gp2-csi (1)
  volumeMode: Filesystem
---
apiVersion: apps/v1
kind: Deployment (2)
metadata:
  name: <local_zone_application> (3)
  namespace: <local_zone_application_namespace> (4)
spec:
```

```
selector:
  matchLabels:
    app: <local_zone_application>
replicas: 1
template:
  metadata:
    labels:
      app: <local_zone_application>
      zone-group: ${ZONE_GROUP_NAME} 5
  spec:
    securityContext:
      seccompProfile:
        type: RuntimeDefault
    nodeSelector: 6
      machine.openshift.io/zone-group: ${ZONE_GROUP_NAME}
    tolerations: 7
    - key: "node-role.kubernetes.io/edge"
      operator: "Equal"
      value: ""
      effect: "NoSchedule"
    containers:
      - image: openshift/origin-node
        command:
         - "/bin/socat"
        args:
          - TCP4-LISTEN:8080,reuseaddr,fork
          - EXEC:'/bin/bash -c \"printf \\\"HTTP/1.0 200 OK\r\n\r\n\\\"; sed -e \\\"/^\r/q\\\"\"'
        imagePullPolicy: Always
        name: echoserver
        ports:
          - containerPort: 8080
        volumeMounts:
          - mountPath: "/mnt/storage"
            name: data
    volumes:
    - name: data
      persistentVolumeClaim:
        claimName: <pvc_name>
```

1. **storageClassName**: For the Local Zone configuration, you must specify **gp2-csi**.
2. **kind**: Defines the **deployment** resource.
3. **name**: Specifies the name of your Local Zone application. For example, **local-zone-demo-app-nyc-1**.
4. **namespace:** Defines the namespace for the AWS Local Zone where you want to run the user workload. For example: **local-zone-app-nyc-1a**.
5. **zone-group**: Defines the group to where a zone belongs. For example, **us-east-1-iah-1**.
6. **nodeSelector**: Targets edge compute nodes that match the specified labels.
7. **tolerations**: Sets the values that match with the **taints** defined on the **MachineSet** manifest for the Local Zone node.

2. Create a **service** resource YAML file for the node. This resource exposes a pod from a targeted edge compute node to services that run inside your Local Zone network.

**Example of a configured service resource for an edge compute node that operates in a Local Zone subnet**

```
apiVersion: v1
kind: Service (1)
metadata:
  name:  <local_zone_application>
  namespace: <local_zone_application_namespace>
spec:
  ports:
    - port: 80
      targetPort: 8080
      protocol: TCP
  type: NodePort
  selector: (2)
    app: <local_zone_application>
```

(1) **kind**: Defines the **service** resource.

(2) **selector:** Specifies the label type applied to managed pods.

### Additional resources

- Installing a cluster on AWS with compute nodes on AWS Local Zones
- Installing a cluster on AWS with compute nodes on AWS Wavelength Zones
- Understanding taints and tolerations

## 8.4. NEXT STEPS

- Optional: Use the AWS Load Balancer (ALB) Operator to expose a pod from a targeted edge compute node to services that run inside of a Local Zones or Wavelength Zones subnet from a public network. See Installing the AWS Load Balancer Operator.