Red Hat OpenShift Service on AWS 4

教程

Red Hat OpenShift Service on AWS 指南

Last Updated: 2024-07-02

Red Hat OpenShift Service on AWS 4 教程

Red Hat OpenShift Service on AWS 指南

法律通告

摘要

创建第一个 Red Hat OpensShift Service on AWS (ROSA)集群的教程。

目录

第 1 章 教程概述

红帽专家逐步教程，以帮助您充分利用受管 OpenShift 集群。

为了快速提供此云专家教程内容，可能还没有在每个支持的配置中进行测试。

第 2 章 教程：使用 HCP 激活和帐户链接的 ROSA

本教程介绍了在部署第一个集群前，使用托管的 control plane (HCP)激活 Red Hat OpenShift Service on AWS (ROSA)和链接到 AWS 帐户的过程。

重要

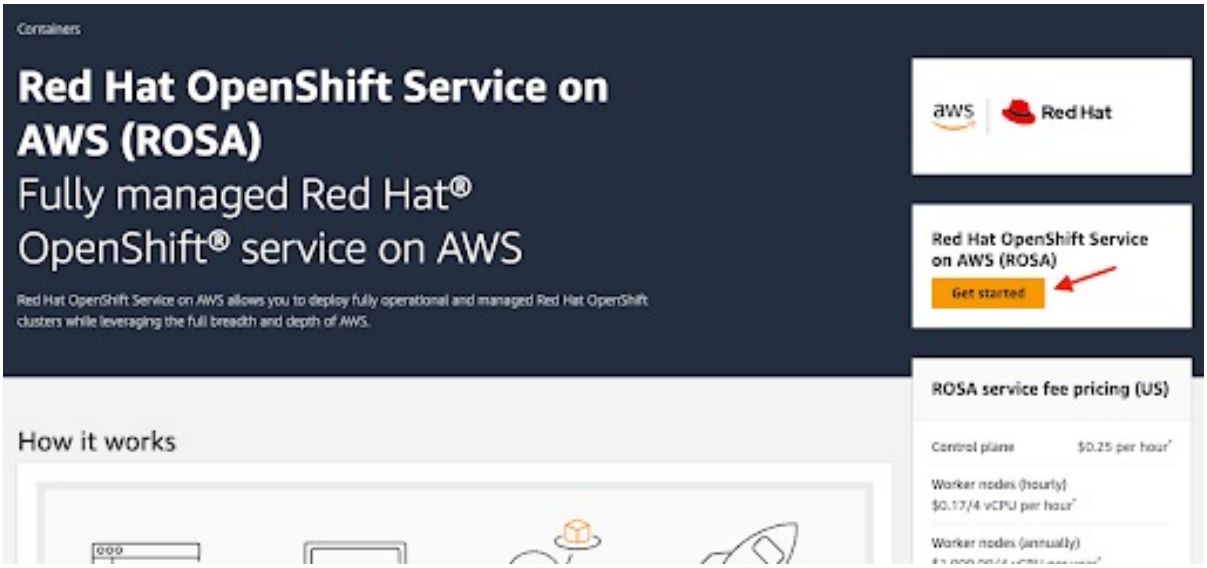
如果您收到产品的私有产品，请确保按照本教程前提供私有提供的说明进行操作。私有产品在产品已激活时（替换有效订阅或首次激活时）被设计。

2.1. 前提条件

- 确保登录到您计划与在前面的步骤中激活 ROSA 和 HCP 的 AWS 帐户关联的红帽帐户。
- 只能将用于服务账单的 AWS 帐户与红帽帐户相关联。通常，具有其他 AWS 帐户（如开发人员帐户）的机构 AWS 帐户通常是被计费的，而不是单独的 AWS 最终用户帐户。
- 属于同一红帽机构的红帽帐户将与同一 AWS 帐户相关联。因此，您可以管理谁有权使用红帽机构帐户级别使用 HCP 集群创建 ROSA。

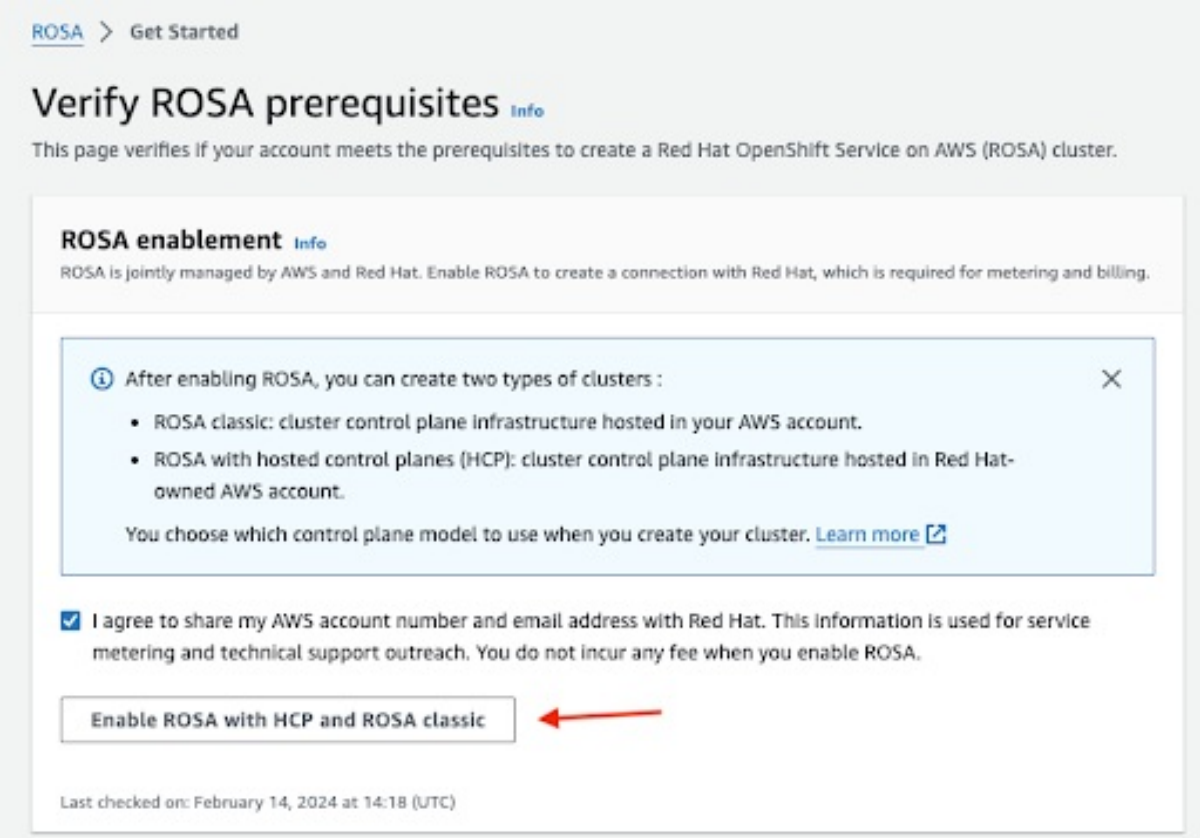
2.2. 订阅启用和 AWS 帐户设置

1. 点 **Get started** 按钮在 AWS 控制台页面中使用 HCP 产品激活 ROSA：



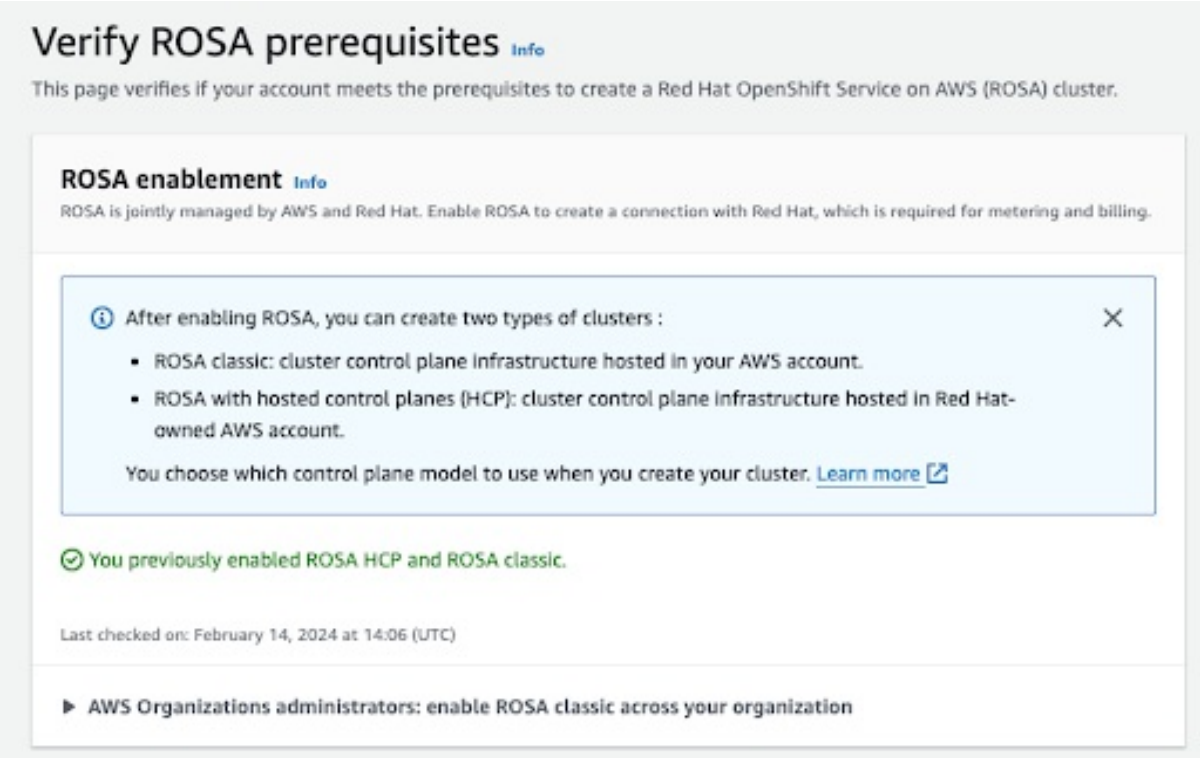
   如果您在完成此过程前已激活了 ROSA，您可以点该按钮并完成帐户链接，如以下步骤所述。

2. 确认您希望与红帽共享联系信息并启用该服务：

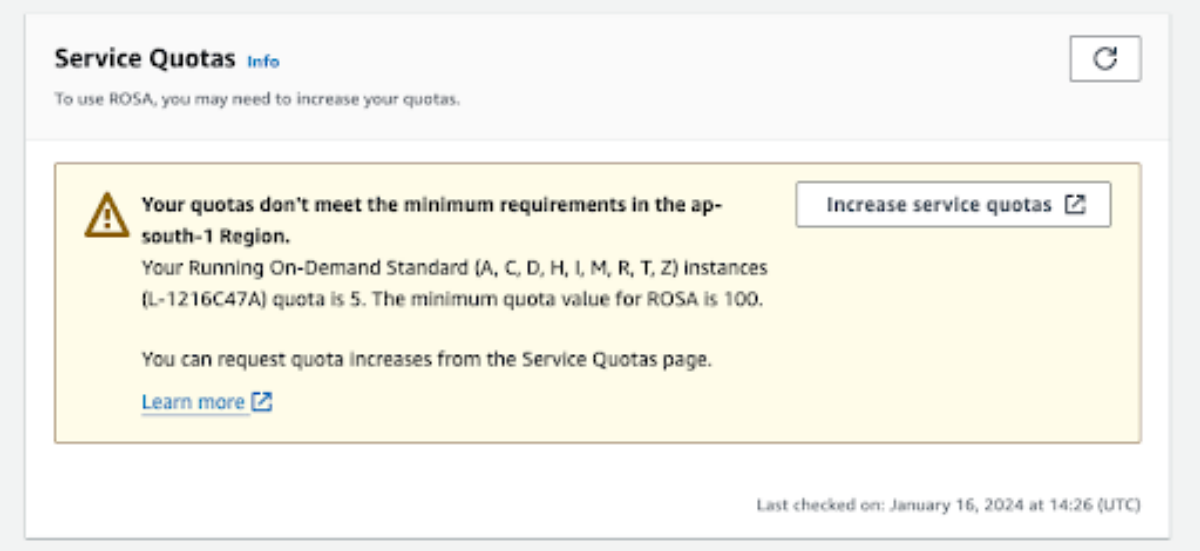


- 在此步骤中启用该服务不会收取。该连接是为在部署第一个集群后发生的计费和计量。这可能需要几分钟时间。

3. 过程完成后，您将看到确认：

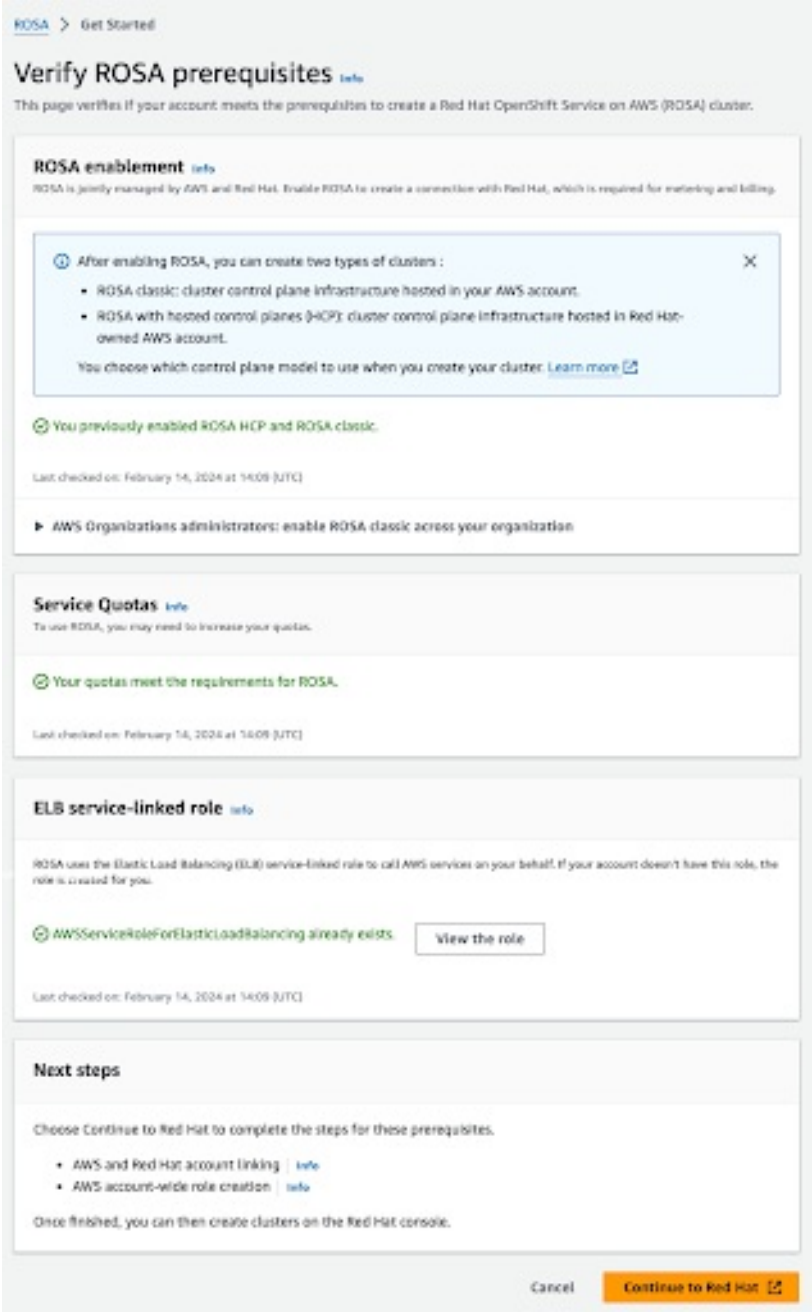


4. 此验证页面中的其他部分显示额外先决条件的状态。如果没有满足任何这些先决条件，则会显示相应的信息。以下是所选区域中配额不足的示例：



a. 单击 **增加服务配额** 按钮，或使用 **了解更多** 链接来获取有关如何管理服务配额的更多信息。如果配额不足，请注意配额是特定于区域的。您可以使用 web 控制台右上角的 region switcher 来重新运行您感兴趣的任何区域的配额检查，然后根据需要提交服务配额增加请求。

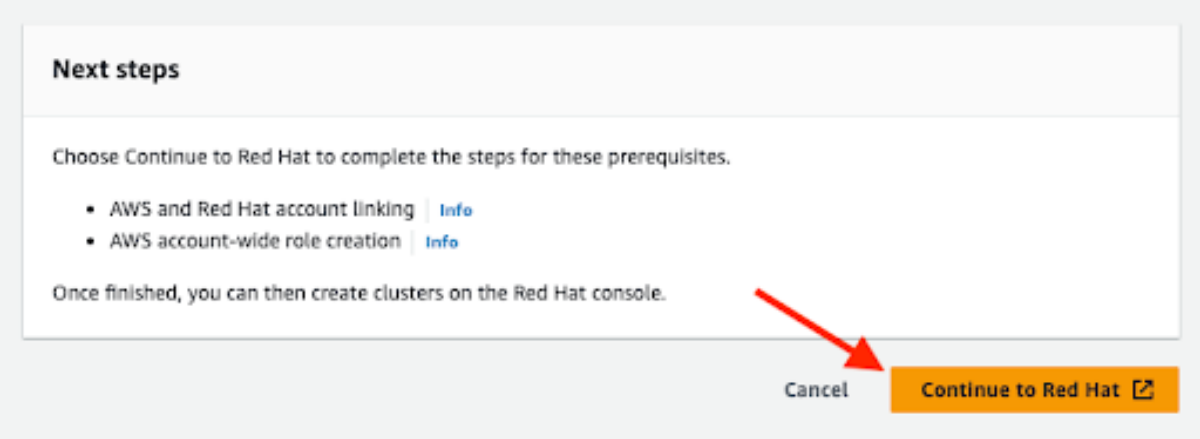
5. 如果满足所有先决条件，页面将类似如下：



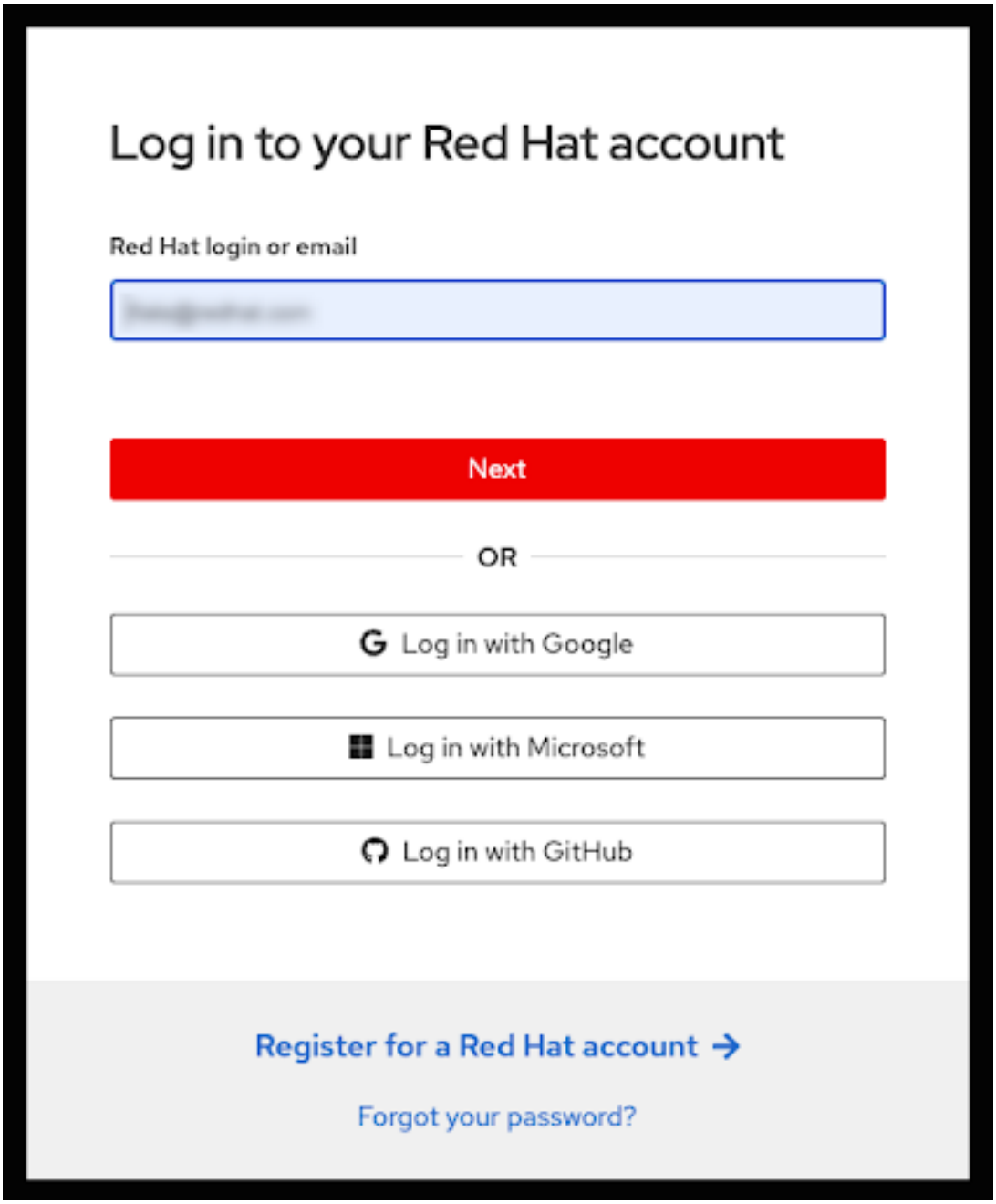
自动为您创建 ELB 服务链接的角色。您可以单击任何小的 **Info** blue 链接来获取上下文帮助和资源。

2.3. AWS 和红帽帐户和订阅链接

1. 点 orange **Continue to Red Hat**按钮继续帐户链接：



2. 如果您还没有在当前浏览器的会话中登录到您的红帽帐户，则会要求您登录到您的帐户：



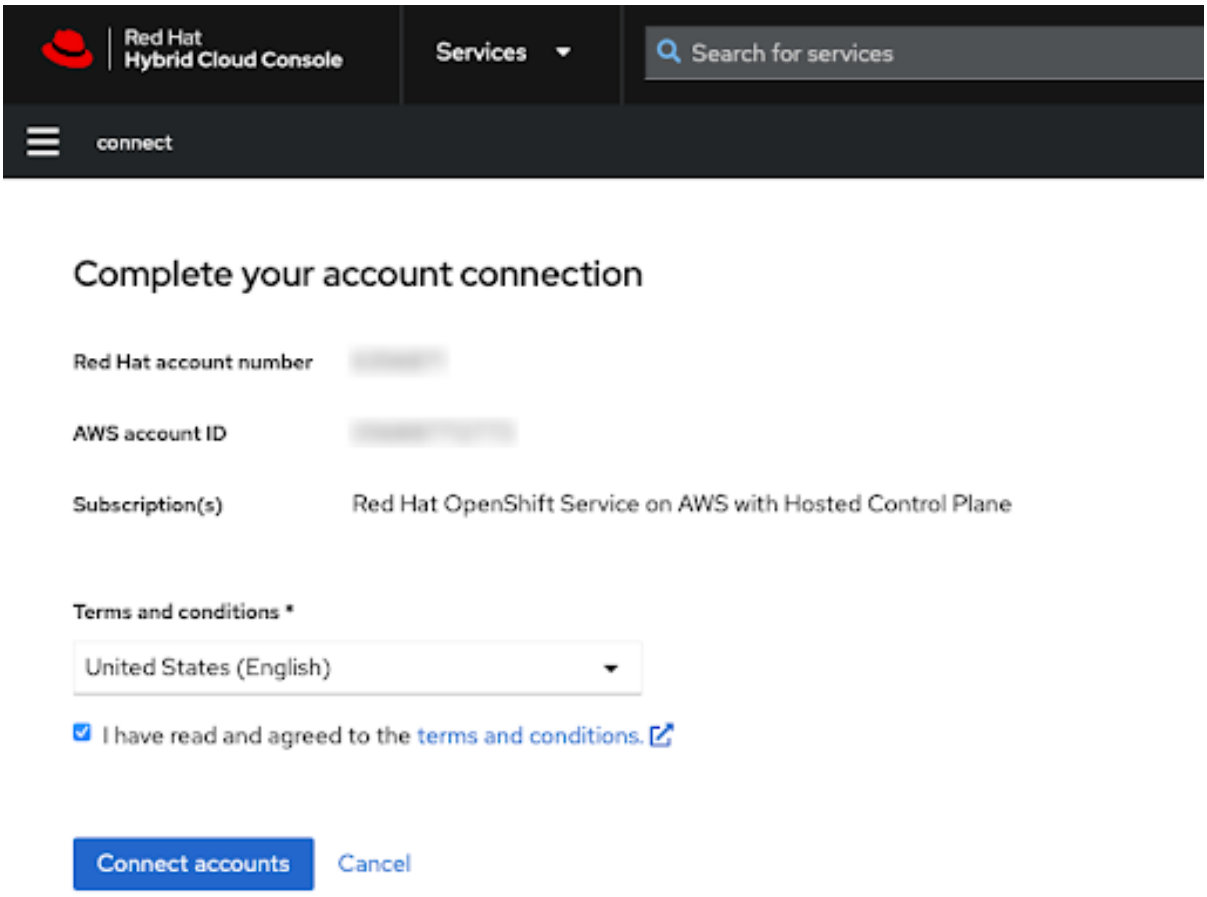
- 您还可以注册新的红帽帐户或在此页面中重置密码。
- 确保登录到您计划与在前面的步骤中激活 ROSA 和 HCP 的 AWS 帐户关联的红帽帐户。

- 只能将用于服务账单的 AWS 帐户与红帽帐户相关联。通常，具有其他 AWS 帐户（如开发人员帐户）的机构 AWS 帐户通常是被计费的，而不是单独的 AWS 最终用户帐户。
- 属于同一红帽机构的红帽帐户将与同一 AWS 帐户相关联。因此，您可以管理谁有权使用红帽机构帐户级别使用 HCP 集群创建 ROSA。

3. 查看条款和条件后完成红帽帐户链接：

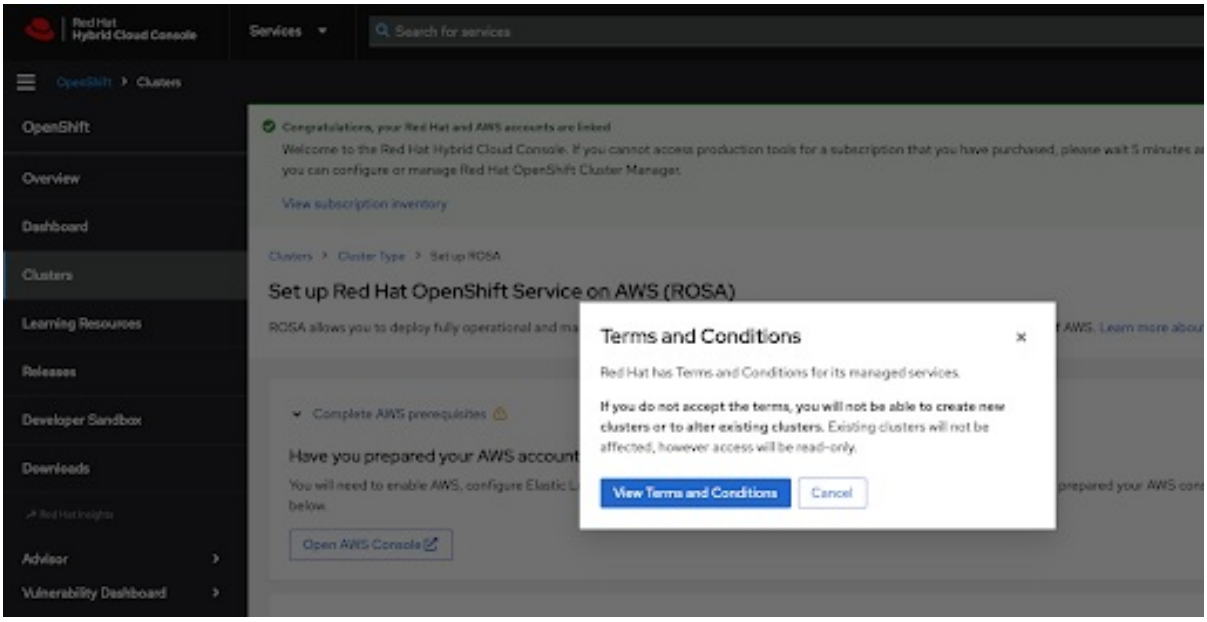
注意

只有在已登录的红帽帐户或管理红帽帐户的红帽机构之前没有链接到 AWS 帐户时，才提供这个步骤。

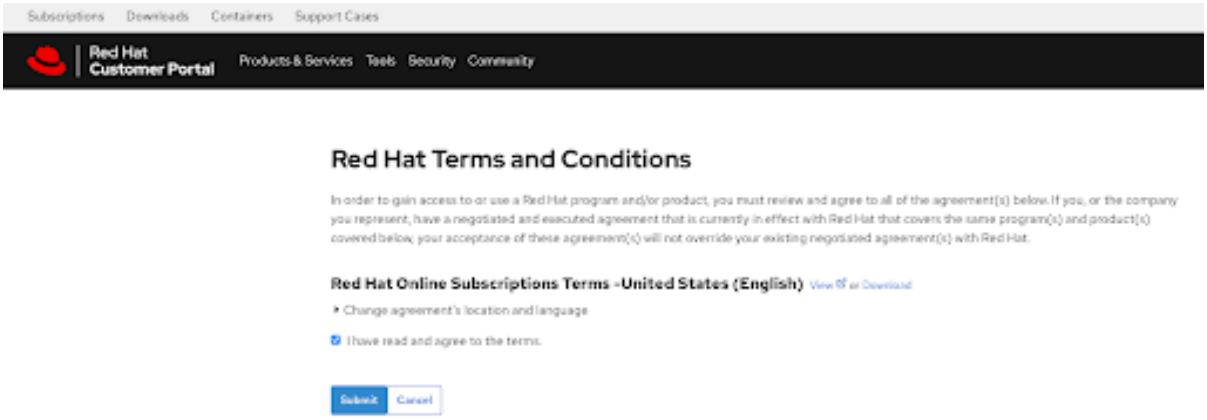


此屏幕中会显示 Red Hat 和 AWS 账户号。

4. 如果您同意了服务条款，请单击"**连接帐户** "按钮。
如果您是第一次使用 Red Hat Hybrid Cloud 控制台，在创建第一个 ROSA 集群前，您需要同意常规受管服务条款和条件：

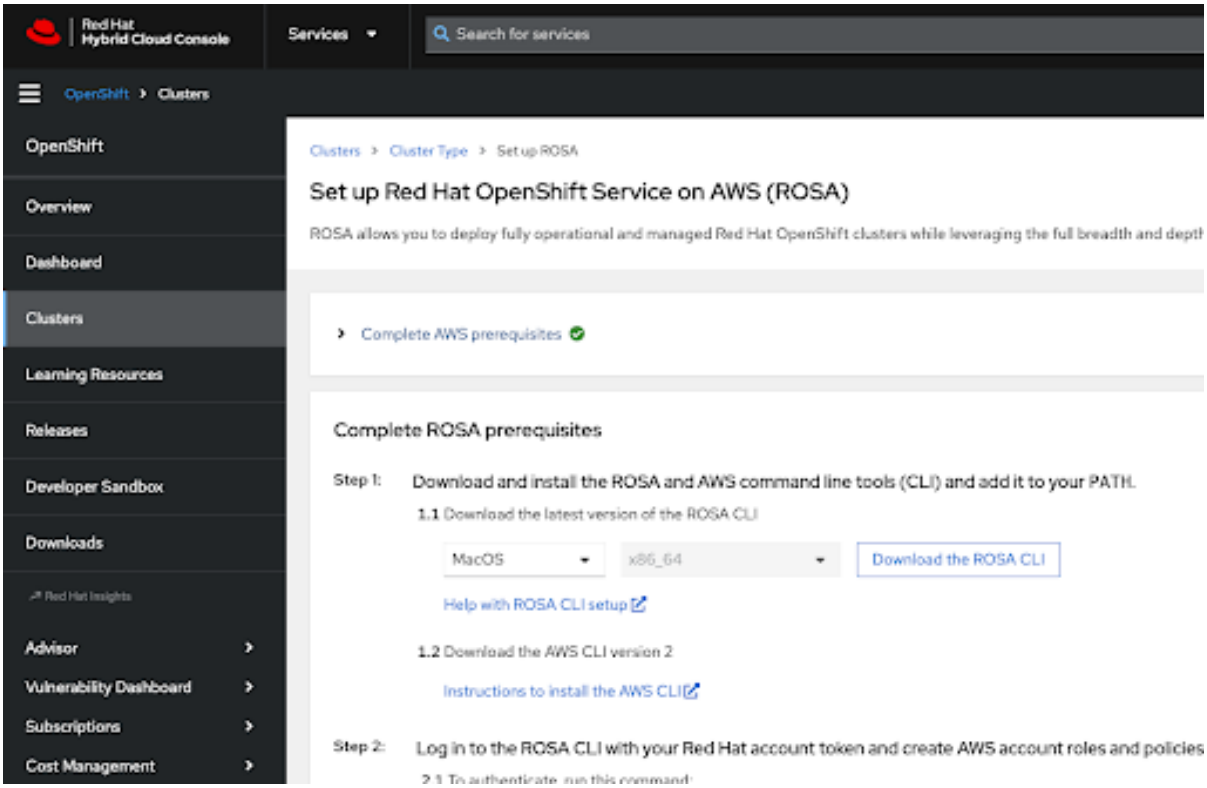


点 **View Conditions 和 Conditions 按钮后会显示需要检查和接受的其他条款**：

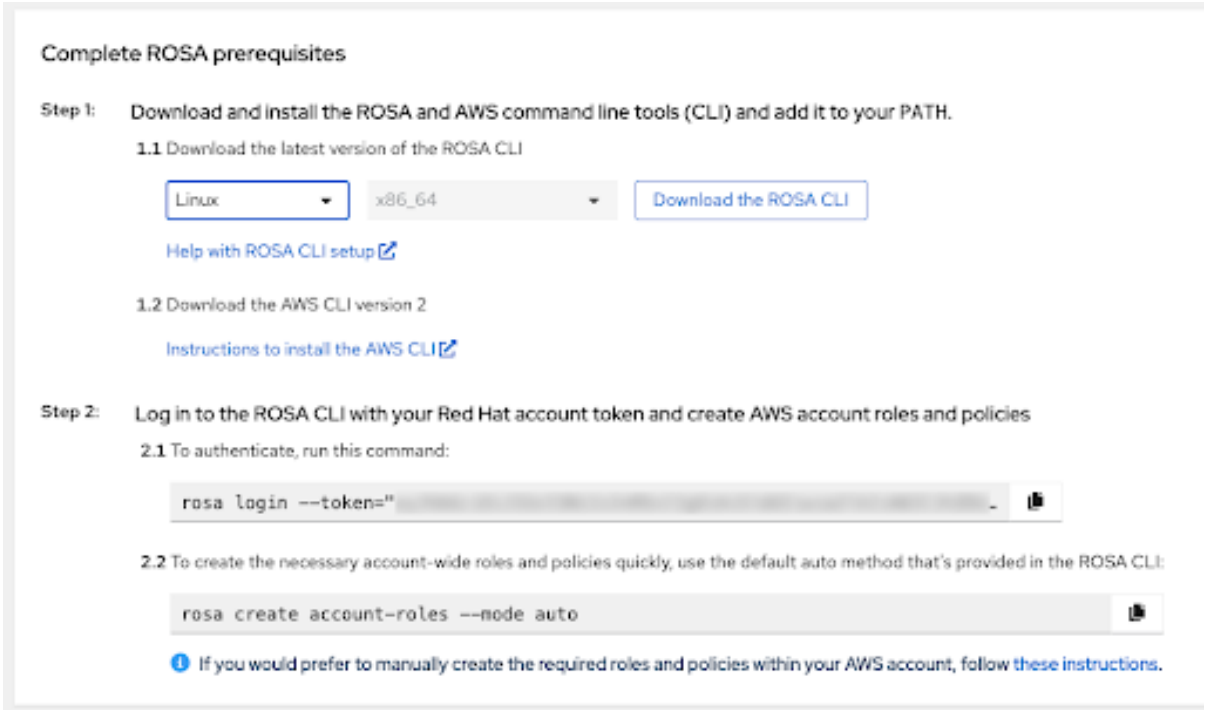


在此时，您审核的任何其他条款后，提交您的协议。

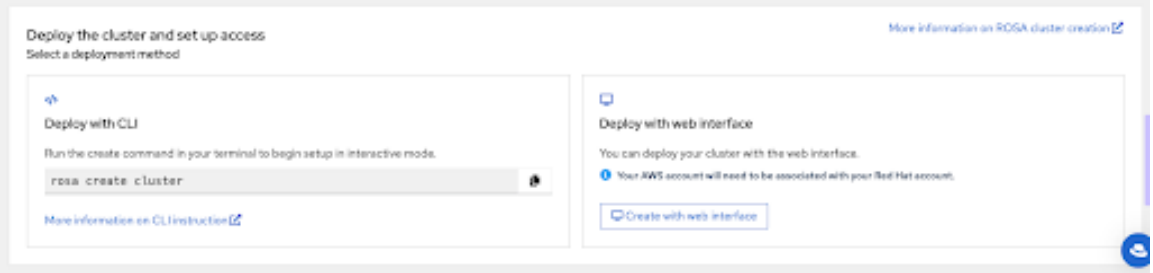
5. Hybrid Cloud Console 提供一个确认 AWS 的先决条件已完成，并列出了集群部署所需的第一步：



6. 与集群的技术部署相关的以下步骤：



- 这些步骤可能在不同于服务启用和帐户链接完成的不同机器上执行。
- 如前文所述，任何属于红帽机构（属于激活 ROSA 服务的 AWS 帐户）的红帽帐户都可以访问创建集群，并可以选择之前在 Red Hat 机构下链接的账单 AWS 帐户。
  本页的最后一部分显示集群部署选项，可以使用 **rosa** CLI 或 Web 控制台：



- 以下步骤使用 **rosa** CLI 描述集群部署。
- 如果您只对使用 Web 控制台进行部署感兴趣，您可以使用 *web 控制台部分跳过使用 HCP 集群部署的 ROSA*。但请注意，某些任务需要 **rosa** CLI，如创建帐户角色。如果您要第一次部署 ROSA，请按照此 CLI 步骤操作，直到运行 **rosa whoami** 命令，然后再跳到 web 控制台部署步骤。

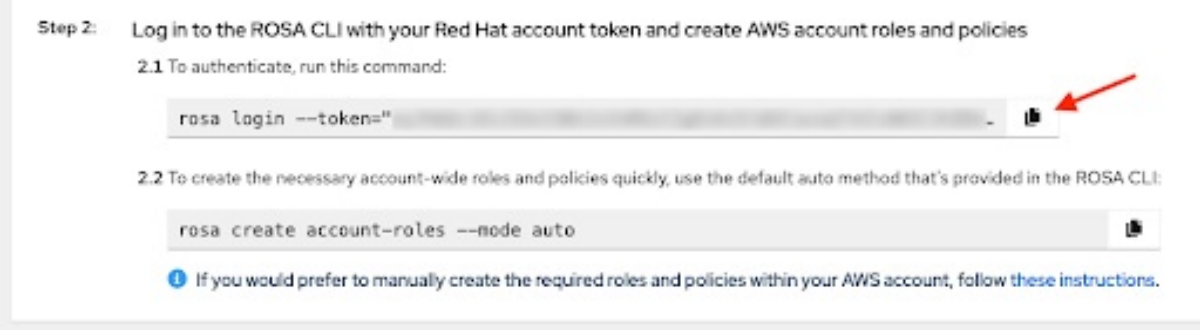
2.4. 使用 CLI 使用 HCP 集群部署的 ROSA

1. 点 **Download the ROSA CLI**按钮为您的操作系统下载 ROSA 命令行界面(CLI)并设置它，如 ROSA CLI 设置帮助 中所述。

重要

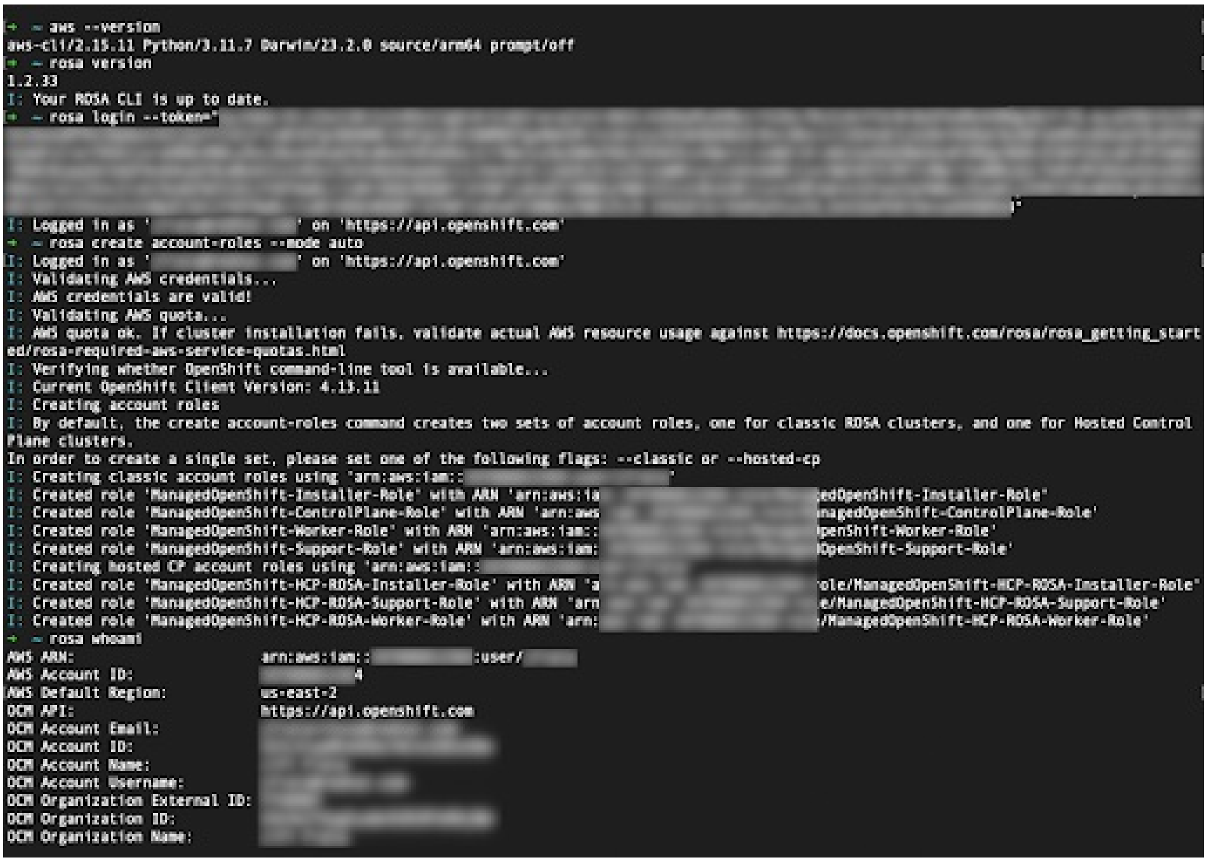
确保安装了最新的 AWS CLI。如需更多信息，请参阅安装 AWS CLI。

2. 完成前面的步骤后，您可以通过运行 **rosa 版本** 来验证两个 CLI 都可用。如果您在终端中使用旧版本和 **aws -version** 命令，这个命令会显示更新通知。
3. 使用 HCP 集群创建 ROSA 的先决条件是通过个人化命令使用 **rosa** cli 登录，并在第 *2.1 步中显示的唯一令牌。要进行身份验证，请在 web 控制台中运行这个命令*。使用 **复制** 按钮轻松复制和粘贴带有完整令牌的命令：

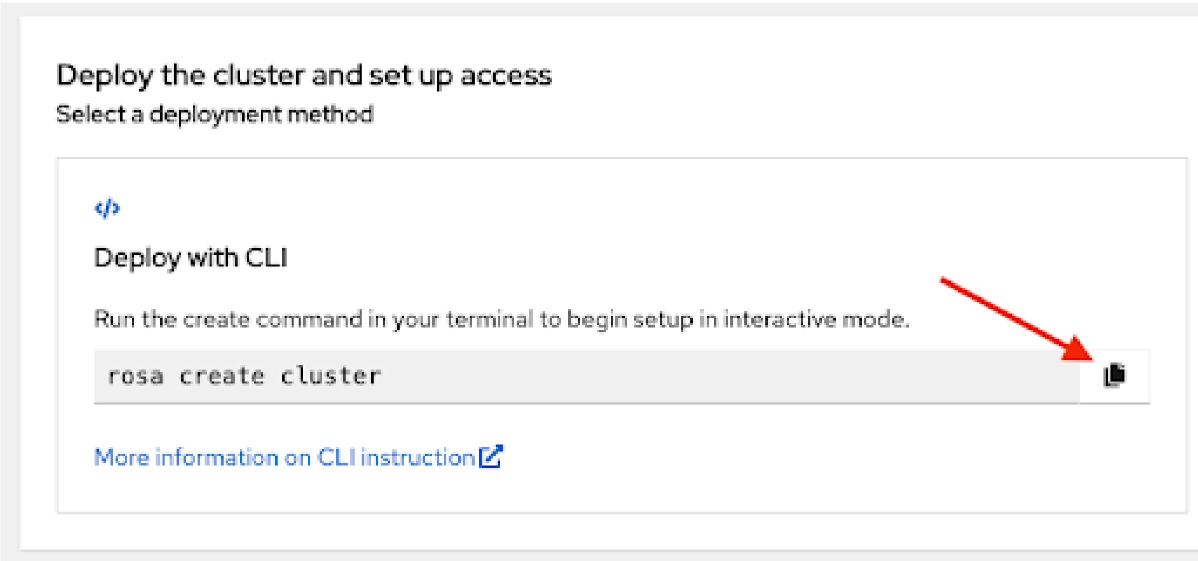


不要共享您的唯一令牌。

4. 在第一个集群部署前的最后一个先决条件，确保创建了必要的集群范围的角色和策略。**rosa** CLI 可以使用第 *2.2 步下显示的命令来解决此问题。在 web 控制台中快速创建必要的集群范围的角色和策略...*。手动创建这些角色和策略的替代选择。
5. 登录后，创建帐户角色并使用 **rosa whoami** 命令验证您的身份，您的终端将类似如下：



6. 使用提供的命令启动集群部署。您可以再次点 **复制** 按钮，并在终端中粘贴命令：



7. 要使用自定义 AWS 配置集，在 **~/.aws/credentials** 中指定的非默认配置集之一，您可以将 **-profile <profile_name>** 选择器添加到 rosa create cluster 命令中，以便命令类似 rosa create cluster **-profile stage**。如果没有使用这个选项指定 AWS CLI 配置集，则默认 AWS CLI 配置集将决定集群部署到的 AWS 基础架构配置集。账单 AWS 配置集在以下步骤之一中选择。

8. 输入集群名称后，将要求您使用托管的 control plane。选择 **yes**：

9. 当使用 HCP 集群部署 ROSA 时，需要指定计费 AWS 帐户：

```
→  ~ rosa create cluster
I: Enabling interactive mode
? Cluster name: test-cluster-07
? Deploy cluster with Hosted Control Plane: Yes
? Billing Account:  [Use arrows to move, type to filter, ? for more help]
>                   [Contract enabled]
```

- 只有链接到当前使用的红帽机构的 AWS 帐户才会显示。
- 指定 AWS 帐户将使用 ROSA 服务，无论基础架构 AWS 帐户是否在同一 AWS 机构中链接到它。
- 您可以看到是否为给定的 AWS 计费帐户启用 ROSA 合同的指标。
- 要选择没有启用合同的 AWS 帐户，请参阅本教程中的前几个步骤来启用合同，并允许服务计费，这是成功集群部署所需的。

10. 在以下步骤中，您将指定要部署的集群的技术详情：

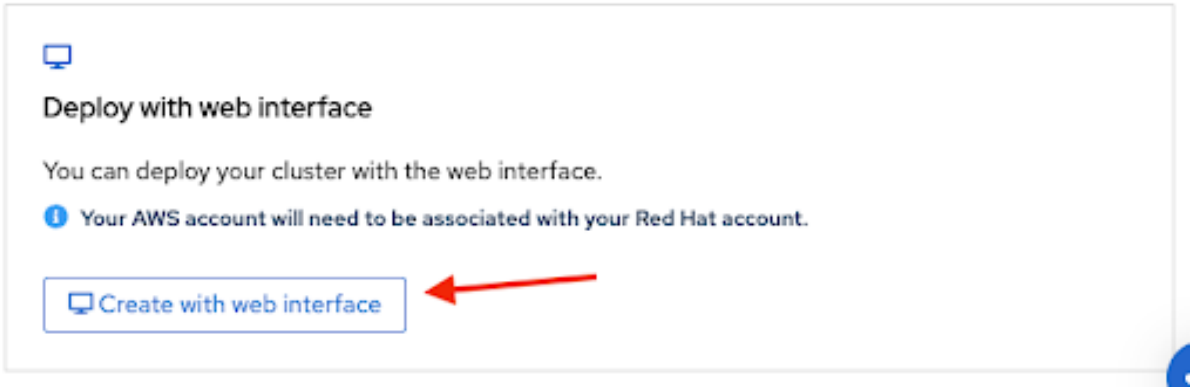
```
→  ~ rosa create cluster
I: Enabling interactive mode
? Cluster name: test-cluster-07
? Deploy cluster with Hosted Control Plane: Yes
? Billing Account:                [Contract enabled]
I: Using '                ' as billing account.
I:
   +--------------------+----------------+
   | Start Date         |Dec 08, 2023    |
   | End Date           |Aug 17, 2024    |
   | Number of vCPUs:   |                |
   | Number of clusters: |               |
   +--------------------+----------------+

? OpenShift version (default = '4.14.8'):  [Use arrows to move, type to filter, ? for more help]
> 4.14.8
  4.14.7
  4.14.6
  4.14.5
```

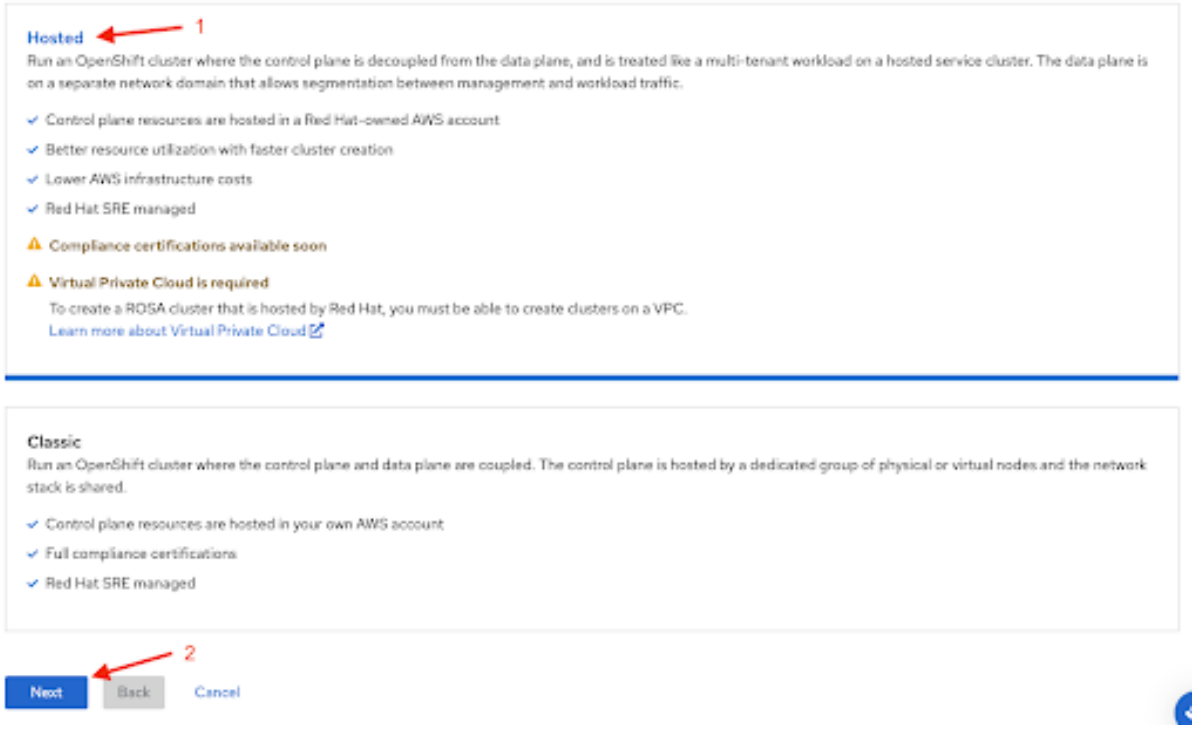
- 这些步骤超出了本教程的范围。如需有关如何使用 CLI 完成 ROSA 使用 HCP 集群 的详细信息，请参阅使用默认选项创建带有 HCP 集群的 ROSA。

2.5. 使用 WEB 控制台使用 HCP 集群部署的 ROSA

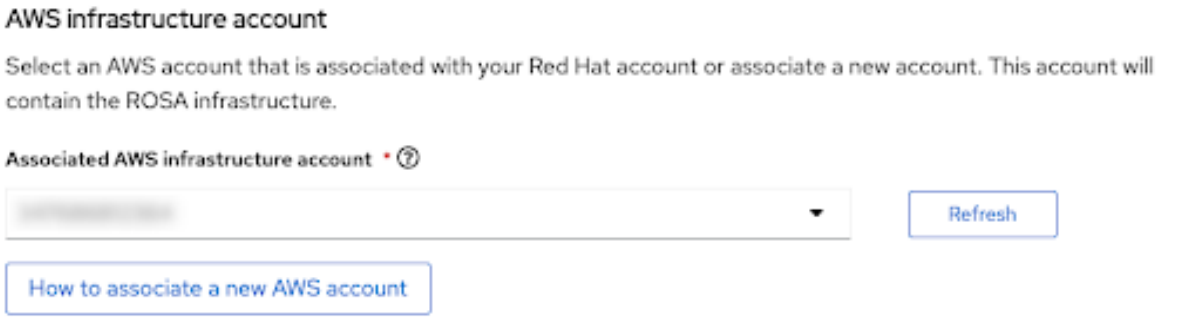
1. 通过选择简介 **设置 ROSA** 页面底部部分中的第二个选项，可以使用 Web 控制台创建集群：



2. 使用 Web 控制台创建 ROSA 集群时的第一步是选择 control plane。在点 **Next** 按钮前，请确定选择了 **Hosted** 选项：

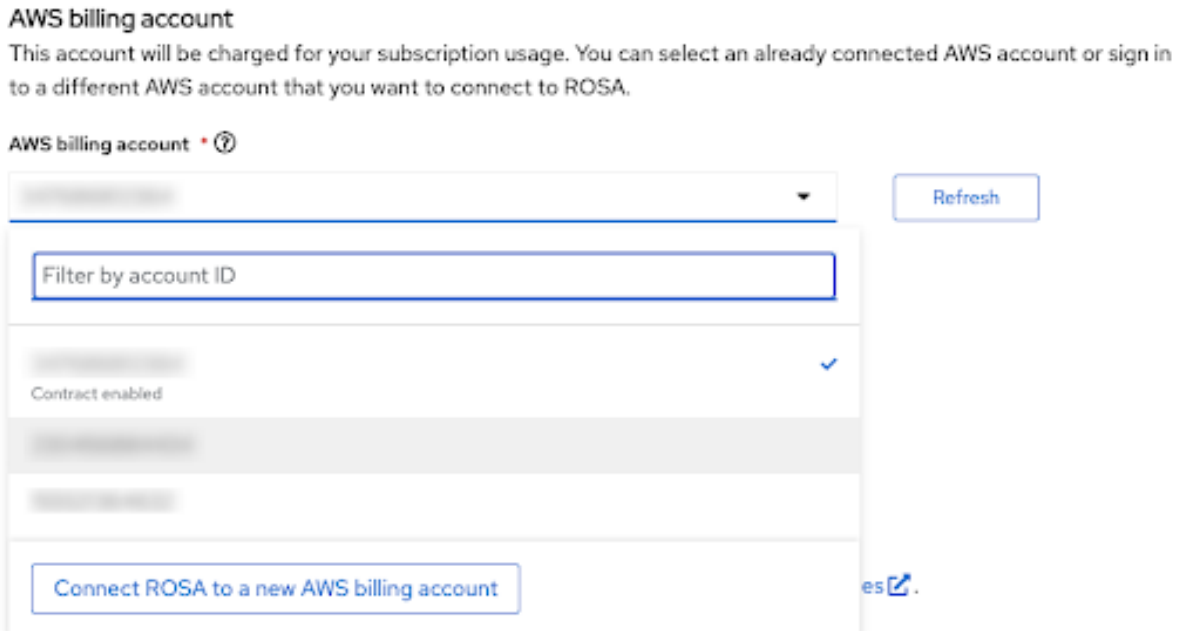


3. 下一步 **Accounts 和 roles** 允许您指定基础架构 AWS 帐户，部署 ROSA 集群以及将消耗和管理资源的位置：



- 如果没有看到您要部署 ROSA 集群的帐户，点 **How to associate a new AWS account**，以了解有关如何为此关联创建或链接帐户角色的详细信息。
- **rosa** CLI 用于此目的。
- 如果您使用多个 AWS 帐户，并为 AWS CLI 配置其配置集，您可以在使用 **rosa** CLI 命令时使用 **--profile** 选择器来指定 AWS 配置集。

4. 账单 AWS 帐户在以下部分中选择：



- 只有链接到当前使用的红帽机构的 AWS 帐户才会显示。
- 指定 AWS 帐户将使用 ROSA 服务，无论基础架构 AWS 帐户是否在同一 AWS 机构中链接到它。
- 您可以看到是否为给定的 AWS 计费帐户启用 ROSA 合同的指示器。
- 如果您要使用尚未启用合同的 AWS 帐户，您可以使用 *Connect ROSA to a new AWS billing 帐户* 来访问 ROSA AWS 控制台页面，您可以在其中按照本教程中描述的步骤使用相应 AWS 帐户登录后激活它。

以下步骤粘贴账单 AWS 帐户选择超出了本教程的范围。

其他资源

- 有关使用 CLI 创建集群的详情，请参考使用 CLI 创建带有 HCP 集群的 ROSA。
- 有关如何使用 Web 控制台完成 ROSA 集群部署的详情，请参阅此学习路径。

第 3 章 教程：验证 ROSA STS 部署的权限

要继续部署 ROSA 集群，帐户必须支持所需的角色和权限。AWS Service Control Policies (SCP)无法阻止安装程序或 Operator 角色发出的 API 调用。

有关启用了 STS 的 ROSA 安装所需的 IAM 资源的详情，请参考：有关使用 STS 的 ROSA 集群的 IAM 资源

本指南针对 ROSA v4.11.X 验证。

3.1. 前提条件

- AWS CLI
- ROSA CLI v1.2.6
- jq CLI
- 具有所需权限的 AWS 角色

3.2. 验证 ROSA 权限

要验证 ROSA 所需的权限，我们可以运行以下部分中包含的脚本，而无需创建任何 AWS 资源。

脚本使用 **rosa**、**aws** 和 **jq** CLI 命令在工作目录中创建文件，这些文件将用于验证连接到当前 AWS 配置的帐户中的权限。

AWS Policy Simulator 用于根据由 **jq** 提取的 API 调用验证每个角色策略的权限；然后，结果会存储在附加 **.results** 的文本文件中。

此脚本旨在验证当前帐户和地区的权限。

3.3. 使用说明

1. 要使用该脚本，请在 **bash** 终端中运行以下命令(-p 选项定义角色的前缀)：

```
$ mkdir scratch
$ cd scratch
$ cat << 'EOF' > verify-permissions.sh
#!/bin/bash
while getopts 'p:' OPTION; do
  case "$OPTION" in
    p)
      PREFIX="$OPTARG"
      ;;
    ?)
      echo "script usage: $(basename \$0) [-p PREFIX]" >&2
      exit 1
      ;;
  esac
done
shift "$(($OPTIND -1))"
rosa create account-roles --mode manual --prefix $PREFIX
INSTALLER_POLICY=$(cat sts_installer_permission_policy.json | jq )
```

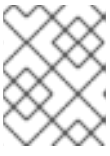
```
CONTROL_PLANE_POLICY=$(cat sts_instance_controlplane_permission_policy.json | jq)
WORKER_POLICY=$(cat sts_instance_worker_permission_policy.json | jq)
SUPPORT_POLICY=$(cat sts_support_permission_policy.json | jq)
simulatePolicy () {
    outputFile="${2}.results"
    echo $2
    aws iam simulate-custom-policy --policy-input-list "$1" --action-names $(jq '.Statement |
map(select(.Effect == "Allow"))[].Action | if type == "string" then . else .[] end' "$2" -r) --output
text > $outputFile
}
simulatePolicy "$INSTALLER_POLICY" "sts_installer_permission_policy.json"
simulatePolicy "$CONTROL_PLANE_POLICY"
"sts_instance_controlplane_permission_policy.json"
simulatePolicy "$WORKER_POLICY" "sts_instance_worker_permission_policy.json"
simulatePolicy "$SUPPORT_POLICY" "sts_support_permission_policy.json"
EOF
$ chmod +x verify-permissions.sh
$ ./verify-permissions.sh -p SimPolTest
```

2. 脚本完成后，查看每个结果文件以确保没有所需的 API 调用被阻止：

```
$ for file in $(ls *.results); do echo $file; cat $file; done
```

输出结果类似如下：

```
sts_installer_permission_policy.json.results
EVALUATIONRESULTS       autoscaling:DescribeAutoScalingGroups   allowed *
MATCHEDSTATEMENTS       PolicyInputList.1       IAM Policy
ENDPOSITION     6       195
STARTPOSITION   17      3
EVALUATIONRESULTS       ec2:AllocateAddress     allowed *
MATCHEDSTATEMENTS       PolicyInputList.1       IAM Policy
ENDPOSITION     6       195
STARTPOSITION   17      3
EVALUATIONRESULTS       ec2:AssociateAddress    allowed *
MATCHEDSTATEMENTS       PolicyInputList.1       IAM Policy
...
```



注意

如果阻止了任何操作，请查看 AWS 提供的错误，并咨询您的管理员来确定 SCP 是否阻断所需的 API 调用。

第 4 章 教程： 使用 AWS WAF 和 AMAZON CLOUDFRONT 来保护 ROSA 工作负载

AWS WAF 是一个 web 应用程序防火墙，可让您监控转发到受保护的 web 应用程序资源的 HTTP 和 HTTPS 请求。

您可以使用 Amazon CloudFront 将 Web 应用程序防火墙(WAF)添加到 Red Hat OpenShift Service on AWS (ROSA)工作负载中。使用外部解决方案可防止 ROSA 资源因为处理 WAF 而导致拒绝服务。

4.1. 前提条件

- ROSA (HCP 或 Classic)集群。
- 您可以访问 OpenShift CLI(**oc**)。
- 您可以访问 AWS CLI (**aws**)。

4.1.1. 环境设置

- 准备环境变量：

```
$ export DOMAIN=apps.example.com 1
$ export AWS_PAGER=""
$ export CLUSTER_NAME=$(oc get infrastructure cluster -o=jsonpath="
{.status.infrastructureName}"  | sed 's/-[a-z0-9]\{5\}$//')
$ export REGION=$(oc get infrastructure cluster -o=jsonpath="
{.status.platformStatus.aws.region}")
$ export AWS_ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
$ export SCRATCH="/tmp/${CLUSTER}/cloudfront-waf"
$ mkdir -p ${SCRATCH}
$ echo "Cluster: ${CLUSTER}, Region: ${REGION}, AWS Account ID:
${AWS_ACCOUNT_ID}"
```

1 使用您要用于 **IngressController** 的自定义域替换。

注意

上一命令中的"Cluster"输出可以是集群名称、集群的内部 ID 或集群的域前缀。如果要使用另一个标识符，您可以通过运行以下命令来手动设置这个值：

```
$ export CLUSTER=my-custom-value
```

4.2. 设置二级入口控制器

需要将二级入口控制器配置为将外部 WAF 保护的流量与标准（和默认）集群入口控制器进行分段。

前提条件

- 自定义域公开可信 SAN 或通配符证书，如 **CN swig.apps.example.com**

重要

Amazon CloudFront 使用 HTTPS 与集群的辅助入口控制器通信。如 Amazon CloudFront 文档中所述，您无法将自签名证书用于 CloudFront 和您的集群之间的 HTTPS 通信。Amazon CloudFront 会验证证书是否由可信证书颁发机构发布。

流程

1. 从私钥和公共证书创建一个新的 TLS secret，其中 **fullchain.pem** 是您的完整的通配符证书链（包括任何中间人），**privkey.pem** 是您的通配符证书的私钥。

   示例

   ```
   $ oc -n openshift-ingress create secret tls waf-tls --cert=fullchain.pem --key=privkey.pem
   ```

2. 创建新的 **IngressController** 资源：

   waf-ingress-controller.yaml示例

   ```
   apiVersion: operator.openshift.io/v1
   kind: IngressController
   metadata:
     name: cloudfront-waf
     namespace: openshift-ingress-operator
   spec:
     domain: apps.example.com (1)
     defaultCertificate:
       name: waf-tls
     endpointPublishingStrategy:
       loadBalancer:
         dnsManagementPolicy: Unmanaged
         providerParameters:
           aws:
             type: NLB
           type: AWS
         scope: External
       type: LoadBalancerService
     routeSelector: (2)
       matchLabels:
        route: waf
   ```

   (1) 使用您要用于 **IngressController** 的自定义域替换。

   (2) 过滤 Ingress Controller 服务的一组路由。在本教程中，我们将使用 **waf** 路由选择器，但如果不提供任何值，则不会发生过滤。

3. 应用 **IngressController** ：

   示例

   ```
   $ oc apply -f waf-ingress-controller.yaml
   ```

4. 验证 IngressController 是否已成功创建外部负载均衡器：

```
$ oc -n openshift-ingress get service/router-cloudfront-waf
```

输出示例

```
NAME                  TYPE           CLUSTER-IP      EXTERNAL-IP
PORT(S)                      AGE
router-cloudfront-waf   LoadBalancer   172.30.16.141
a68a838a7f26440bf8647809b61c4bc8-4225395f488830bd.elb.us-east-1.amazonaws.com
80:30606/TCP,443:31065/TCP   2m19s
```

4.2.1. 配置 AWS WAF

AWS WAF 服务是一个 web 应用程序防火墙，可让您监控、保护和控制转发到受保护的 web 应用程序资源（如 ROSA）的 HTTP 和 HTTPS 请求。

1. 创建 AWS WAF 规则文件以应用到我们的 Web ACL：

```
$ cat << EOF > ${SCRATCH}/waf-rules.json
[
  {
    "Name": "AWS-AWSManagedRulesCommonRuleSet",
    "Priority": 0,
    "Statement": {
      "ManagedRuleGroupStatement": {
        "VendorName": "AWS",
        "Name": "AWSManagedRulesCommonRuleSet"
      }
    },
    "OverrideAction": {
      "None": {}
    },
    "VisibilityConfig": {
      "SampledRequestsEnabled": true,
      "CloudWatchMetricsEnabled": true,
      "MetricName": "AWS-AWSManagedRulesCommonRuleSet"
    }
  },
  {
    "Name": "AWS-AWSManagedRulesSQLiRuleSet",
    "Priority": 1,
    "Statement": {
      "ManagedRuleGroupStatement": {
        "VendorName": "AWS",
        "Name": "AWSManagedRulesSQLiRuleSet"
      }
    },
    "OverrideAction": {
      "None": {}
    },
    "VisibilityConfig": {
      "SampledRequestsEnabled": true,
      "CloudWatchMetricsEnabled": true,
      "MetricName": "AWS-AWSManagedRulesSQLiRuleSet"
    }
```

```
    }
]
EOF
```

这将启用 Core (Common)和 SQL AWS Managed Rule Sets。

2. 使用上面指定的规则创建 AWS WAF Web ACL：

```
$ WAF_WACL=$(aws wafv2 create-web-acl \
  --name cloudfront-waf \
  --region ${REGION} \
  --default-action Allow={} \
  --scope CLOUDFRONT \
  --visibility-config
SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=${CLUSTER}-
waf-metrics \
  --rules file://${SCRATCH}/waf-rules.json \
  --query 'Summary.Name' \
  --output text)
```

4.3. CONFIGURE AMAZON CLOUDFRONT

1. 检索新创建的自定义 ingress 控制器的 NLB 主机名：

```
$ NLB=$(oc -n openshift-ingress get service router-cloudfront-waf \
  -o jsonpath='{.status.loadBalancer.ingress[0].hostname}')
```

2. 将您的证书导入到 Amazon 证书管理器中，其中 **cert.pem** 是您的通配符证书，**fullchain.pem** 是您的通配符证书的链，**privkey.pem** 是您的通配符证书的私钥。

注意

无论您部署了集群是什么，都必须将此证书导入到 **us-east-1** 中，因为 Amazon CloudFront 都是全局 AWS 服务。

示例

```
$ aws acm import-certificate --certificate file://cert.pem \
  --certificate-chain file://fullchain.pem \
  --private-key file://privkey.pem \
  --region us-east-1
```

3. 登录到 AWS 控制台 以创建 CloudFront 发行版。

4. 使用以下信息配置 CloudFront 发行版：

注意

如果没有在下表中指定选项，请保留默认值（可能为空）。

选项	值
原始域	上一命令的输出 [1]
Name	rosa-waf-ingress [2]
查看器协议策略	将 HTTP 重定向到 HTTPS
允许的 HTTP 方法	GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE
缓存策略	CachingDisabled
原始请求策略	AllViewer
Web 应用程序防火墙(WAF)	启用安全保护
使用现有的 WAF 配置	true
选择 Web ACL	**cloudfront-waf**
备用域名(CNAME)	*.apps.example.com [3]
自定义 SSL 证书	从上面的步骤中选择您导入的证书 [4]

1. 运行 **echo ${NLB}** 以获取原始域。
2. 如果您有多个集群，请确保原始名称是唯一的。
3. 这应该与您用来创建自定义入口控制器的通配符域匹配。
4. 这应该与上面输入的替代域名匹配。

5. 检索 Amazon CloudFront Distribution 端点：

```
$ aws cloudfront list-distributions --query "DistributionList.Items[?Origins.Items[?
DomainName=='${NLB}']].DomainName" --output text
```

6. 使用 CNAME 将自定义通配符域的 DNS 更新到上面步骤中的 Amazon CloudFront 分发端点。

示例

```
*.apps.example.com CNAME d1b2c3d4e5f6g7.cloudfront.net
```

4.4. 部署示例应用程序

1. 运行以下命令，为示例应用程序创建一个新项目：

```
$ oc new-project hello-world
```

2. 部署 hello world 应用：

```
$ oc -n hello-world new-app --image=docker.io/openshift/hello-openshift
```

3. 为应用程序创建指定自定义域名的路由：

示例

```
$ oc -n hello-world create route edge --service=hello-openshift hello-openshift-tls \
--hostname hello-openshift.${DOMAIN}
```

4. 标记路由，使其接受到自定义入口控制器：

```
$ oc -n hello-world label route.route.openshift.io/hello-openshift-tls route=waf
```

4.5. 测试 WAF

1. 测试应用程序是否可以在 Amazon CloudFront 后面访问：

示例

```
$ curl "https://hello-openshift.${DOMAIN}"
```

输出示例

```
Hello OpenShift!
```

2. 测试 WAF 是否拒绝错误请求：

示例

```
$ curl -X POST "https://hello-openshift.${DOMAIN}" \
  -F "user='<script><alert>Hello></alert></script>'"
```

输出示例

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<HTML><HEAD><META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-
8859-1">
<TITLE>ERROR: The request could not be satisfied</TITLE>
</HEAD><BODY>
<H1>403 ERROR</H1>
<H2>The request could not be satisfied.</H2>
<HR noshade size="1px">
Request blocked.
We can't connect to the server for this app or website at this time. There might be too much
traffic or a configuration error. Try again later, or contact the app or website owner.
<BR clear="all">
```

```
If you provide content to customers through CloudFront, you can find steps to troubleshoot
and help prevent this error by reviewing the CloudFront documentation.
<BR clear="all">
<HR noshade size="1px">
<PRE>
Generated by cloudfront (CloudFront)
Request ID: nFk9q2yB8jddI6FZOTjdliexzx-FwZtr8xUQUNT75HThPlrALDxbag==
</PRE>
<ADDRESS>
</ADDRESS>
</BODY></HTML>
```

预期的结果是一个 **403 ERROR**，这意味着 AWS WAF 正在保护您的应用程序。

4.6. 其他资源

- 使用 AWS WAF、CloudFront 和 ROSA | Amazon Web Services on iwl 添加 额外的安全性

第 5 章 教程： 使用 AWS WAF 和 AWS ALB 来保护 ROSA 工作负载

AWS WAF 是一个 web 应用程序防火墙，可让您监控转发到受保护的 web 应用程序资源的 HTTP 和 HTTPS 请求。

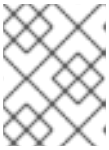
您可以使用 AWS Application Load Balancer (ALB)将 Web 应用程序防火墙(WAF)添加到 Red Hat OpenShift Service on AWS (ROSA)工作负载中。使用外部解决方案可防止 ROSA 资源因为处理 WAF 而导致拒绝服务。

重要

建议您使用更灵活的 CloudFront 方法，除非绝对必须使用基于 ALB 的解决方案。

5.1. 前提条件

- 多个可用区(AZ) ROSA (HCP 或 Classic)集群。



注意

根据 AWS 文档，AWS ALB 在 AZ 之间至少需要两个 *公共子网*。因此，只有多个 AZ ROSA 集群可以与 ALB 一起使用。

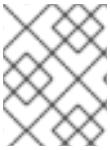
- 您可以访问 OpenShift CLI(**oc**)。
- 您可以访问 AWS CLI (**aws**)。

5.1.1. 环境设置

- 准备环境变量：

```
$ export AWS_PAGER=""
$ export CLUSTER=$(oc get infrastructure cluster -o=jsonpath="
{.status.infrastructureName}")
$ export REGION=$(oc get infrastructure cluster -o=jsonpath="
{.status.platformStatus.aws.region}")
$ export OIDC_ENDPOINT=$(oc get authentication.config.openshift.io cluster -o
jsonpath='{.spec.serviceAccountIssuer}' | sed  's|^https://||')
$ export AWS_ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
$ export SCRATCH="/tmp/${CLUSTER}/alb-waf"
$ mkdir -p ${SCRATCH}
$ echo "Cluster: $(echo ${CLUSTER} | sed 's/-[a-z0-9]\{5\}$//'), Region: ${REGION}, OIDC
Endpoint: ${OIDC_ENDPOINT}, AWS Account ID: ${AWS_ACCOUNT_ID}"
```

5.1.2. AWS VPC 和子网



注意

本节只适用于部署到现有 VPC 的集群。如果您没有将集群部署到现有的 VPC 中，请跳过本节并继续以下安装部分。

1. 将以下变量设置为您的 ROSA 部署的正确值：

```
$ export VPC_ID=<vpc-id> 1
$ export PUBLIC_SUBNET_IDS=(<space-separated-list-of-ids>) 2
$ export PRIVATE_SUBNET_IDS=(<space-separated-list-of-ids>) 3
```

1. 使用集群的 VPC ID 替换，例如： **export VPC_ID=vpc-04c429b7dbc4680ba**。
2. 使用空格分隔集群的专用子网 ID 列表替换，确保保留 （）。例如： **export PUBLIC_SUBNET_IDS=(subnet-056fd6861ad332ba2 subnet-08ce3b4ec753fe74c subnet-071aa28228664972f)**。
3. 使用空格分隔集群的专用子网 ID 列表替换，确保保留 （）。例如： **export PRIVATE_SUBNET_IDS=(subnet-0b933d72a8d72c36a subnet-0817eb72070f1d3c2 subnet-0806e64159b66665a)**。

2. 使用集群标识符向集群的 VPC 添加标签：

```
$ aws ec2 create-tags --resources ${VPC_ID} \
  --tags Key=kubernetes.io/cluster/${CLUSTER},Value=shared --region ${REGION}
```

3. 在您的公共子网中添加标签：

```
$ aws ec2 create-tags \
  --resources ${PUBLIC_SUBNET_IDS} \
  --tags Key=kubernetes.io/role/elb,Value='1' \
        Key=kubernetes.io/cluster/${CLUSTER},Value=shared \
  --region ${REGION}
```

4. 在您的私有子网中添加标签：

```
$ aws ec2 create-tags \
  --resources ${PRIVATE_SUBNET_IDS} \
  --tags Key=kubernetes.io/role/internal-elb,Value='1' \
        Key=kubernetes.io/cluster/${CLUSTER},Value=shared \
  --region ${REGION}
```

5.2. 部署 AWS LOAD BALANCER OPERATOR

AWS Load Balancer Operator 用于在 ROSA 集群中安装、管理和配置 **aws-load-balancer-controller** 实例。要在 ROSA 中部署 ALB，我们需要首先部署 AWS Load Balancer Operator。

1. 运行以下命令，创建一个新项目来部署 AWS Load Balancer Operator：

```
$ oc new-project aws-load-balancer-operator
```

2. 运行以下命令，为 AWS Load Balancer Controller 创建 AWS IAM 策略（如果不存在）：

注意

该策略 来自上游 AWS Load Balancer Controller 策略 。Operator 需要它才能正常工作。

```
$ POLICY_ARN=$(aws iam list-policies --query \
    "Policies[?PolicyName=='aws-load-balancer-operator-policy'].{ARN:Arn}" \
    --output text)
```

```
$ if [[ -z "${POLICY_ARN}" ]]; then
   wget -O "${SCRATCH}/load-balancer-operator-policy.json" \
     https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-
controller/main/docs/install/iam_policy.json
    POLICY_ARN=$(aws --region "$REGION" --query Policy.Arn \
    --output text iam create-policy \
    --policy-name aws-load-balancer-operator-policy \
    --policy-document "file://${SCRATCH}/load-balancer-operator-policy.json")
fi
```

3. 为 AWS Load Balancer Operator 创建 AWS IAM 信任策略：

```
$ cat <<EOF > "${SCRATCH}/trust-policy.json"
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Condition": {
   "StringEquals" : {
    "${OIDC_ENDPOINT}:sub": ["system:serviceaccount:aws-load-balancer-operator:aws-
load-balancer-operator-controller-manager", "system:serviceaccount:aws-load-balancer-
operator:aws-load-balancer-controller-cluster"]
   }
 },
 "Principal": {
   "Federated": "arn:aws:iam::$AWS_ACCOUNT_ID:oidc-provider/${OIDC_ENDPOINT}"
 },
 "Action": "sts:AssumeRoleWithWebIdentity"
 }
 ]
}
EOF
```

4. 为 AWS Load Balancer Operator 创建 AWS IAM 角色：

```
$ ROLE_ARN=$(aws iam create-role --role-name "${CLUSTER}-alb-operator" \
   --assume-role-policy-document "file://${SCRATCH}/trust-policy.json" \
   --query Role.Arn --output text)
```

5. 运行以下命令，将 AWS Load Balancer Operator 策略附加到之前创建的 IAM 角色中：

```
$ aws iam attach-role-policy --role-name "${CLUSTER}-alb-operator" \
    --policy-arn ${POLICY_ARN}
```

6. 为 AWS Load Balancer Operator 创建 secret，以假定我们新创建的 AWS IAM 角色：

```
$ cat << EOF | oc apply -f -
apiVersion: v1
kind: Secret
```

```
metadata:
  name: aws-load-balancer-operator
  namespace: aws-load-balancer-operator
stringData:
  credentials: |
    [default]
    role_arn = ${ROLE_ARN}
    web_identity_token_file = /var/run/secrets/openshift/serviceaccount/token
EOF
```

7. 安装 AWS Load Balancer Operator：

```
$ cat << EOF | oc apply -f -
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: aws-load-balancer-operator
  namespace: aws-load-balancer-operator
spec:
  upgradeStrategy: Default
---
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: aws-load-balancer-operator
  namespace: aws-load-balancer-operator
spec:
  channel: stable-v1.0
  installPlanApproval: Automatic
  name: aws-load-balancer-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
  startingCSV: aws-load-balancer-operator.v1.0.0
EOF
```

8. 使用 Operator 部署 AWS Load Balancer Controller 实例：

注意

如果您在此处收到错误并尝试重试，这意味着 Operator 还没有完成安装。

```
$ cat << EOF | oc apply -f -
apiVersion: networking.olm.openshift.io/v1
kind: AWSLoadBalancerController
metadata:
  name: cluster
spec:
  credentials:
    name: aws-load-balancer-operator
  enabledAddons:
    - AWSWAFv2
EOF
```

9. 检查 Operator 和控制器 pod 是否都在运行：

```
$ oc -n aws-load-balancer-operator get pods
```

如果没有等待片刻并重试，您应该看到以下内容：

```
NAME                                                READY   STATUS    RESTARTS   AGE
aws-load-balancer-controller-cluster-6ddf658785-pdp5d           1/1     Running   0          99s
aws-load-balancer-operator-controller-manager-577d9ffcb9-w6zqn   2/2     Running   0
2m4s
```

5.3. 部署示例应用程序

1. 为示例应用程序创建一个新项目：

```
$ oc new-project hello-world
```

2. 部署 hello world 应用：

```
$ oc new-app -n hello-world --image=docker.io/openshift/hello-openshift
```

3. 将预先创建的服务资源转换为 NodePort 服务类型：

```
$ oc -n hello-world patch service hello-openshift -p '{"spec":{"type":"NodePort"}}'
```

4. 使用 AWS Load Balancer Operator 部署 AWS ALB：

```
$ cat << EOF | oc apply -f -
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: hello-openshift-alb
  namespace: hello-world
  annotations:
    alb.ingress.kubernetes.io/scheme: internet-facing
spec:
  ingressClassName: alb
  rules:
    - http:
        paths:
          - path: /
            pathType: Exact
            backend:
              service:
                name: hello-openshift
                port:
                  number: 8080
EOF
```

5. curl AWS ALB Ingress 端点，以验证 hello world 应用程序是否可访问：

注意

AWS ALB 置备需要几分钟。如果您收到显示 **curl: (6) Could not resolve host** 的错误，请等待再试一次。

```
$ INGRESS=$(oc -n hello-world get ingress hello-openshift-alb -o
jsonpath='{.status.loadBalancer.ingress[0].hostname}')
$ curl "http://${INGRESS}"
```

输出示例

```
Hello OpenShift!
```

5.3.1. 配置 AWS WAF

AWS WAF 服务是一个 web 应用程序防火墙，可让您监控、保护和控制转发到受保护的 web 应用程序资源（如 ROSA）的 HTTP 和 HTTPS 请求。

1. 创建 AWS WAF 规则文件以应用到我们的 Web ACL：

```
$ cat << EOF > ${SCRATCH}/waf-rules.json
[
  {
    "Name": "AWS-AWSManagedRulesCommonRuleSet",
    "Priority": 0,
    "Statement": {
      "ManagedRuleGroupStatement": {
        "VendorName": "AWS",
        "Name": "AWSManagedRulesCommonRuleSet"
      }
    },
    "OverrideAction": {
      "None": {}
    },
    "VisibilityConfig": {
      "SampledRequestsEnabled": true,
      "CloudWatchMetricsEnabled": true,
      "MetricName": "AWS-AWSManagedRulesCommonRuleSet"
    }
  },
  {
    "Name": "AWS-AWSManagedRulesSQLiRuleSet",
    "Priority": 1,
    "Statement": {
      "ManagedRuleGroupStatement": {
        "VendorName": "AWS",
        "Name": "AWSManagedRulesSQLiRuleSet"
      }
    },
    "OverrideAction": {
      "None": {}
    },
    "VisibilityConfig": {
      "SampledRequestsEnabled": true,
```

```
        "CloudWatchMetricsEnabled": true,
        "MetricName": "AWS-AWSManagedRulesSQLiRuleSet"
      }
    }
]
EOF
```

这将启用 Core (Common)和 SQL AWS Managed Rule Sets。

2. 使用上面指定的规则创建 AWS WAF Web ACL：

```
$ WAF_ARN=$(aws wafv2 create-web-acl \
  --name ${CLUSTER}-waf \
  --region ${REGION} \
  --default-action Allow={} \
  --scope REGIONAL \
  --visibility-config
SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=${CLUSTER}-
waf-metrics \
  --rules file://${SCRATCH}/waf-rules.json \
  --query 'Summary.ARN' \
  --output text)
```

3. 使用 AWS WAF Web ACL ARN 注解 Ingress 资源：

```
$ oc annotate -n hello-world ingress.networking.k8s.io/hello-openshift-alb \
  alb.ingress.kubernetes.io/wafv2-acl-arn=${WAF_ARN}
```

4. 等待 10 秒，以便规则传播并测试应用程序是否仍然可以正常工作：

```
$ curl "http://${INGRESS}"
```

输出示例

```
Hello OpenShift!
```

5. 测试 WAF 是否拒绝错误请求：

```
$ curl -X POST "http://${INGRESS}" \
  -F "user='<script><alert>Hello></alert></script>'"
```

输出示例

```
<html>
<head><title>403 Forbidden</title></head>
<body>
<center><h1>403 Forbidden</h1></center>
</body>
</html
```

注意

激活 AWS WAF 集成有时可能需要几分钟。如果您没有收到 **403 Forbidden** 错误，请等待几秒钟，然后重试。

预期的结果是一个 **403 Forbidden** 错误，这意味着 AWS WAF 正在保护您的应用程序。

5.4. 其他资源

- 使用 AWS WAF、CloudFront 和 ROSA | Amazon Web Services on iwl 添加 额外的安全性

第 6 章 教程：在 ROSA 集群上部署 OPENSHIFT API FOR DATA PROTECTION

重要

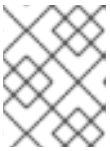
此内容由红帽专家编写，但尚未在所有支持的配置中进行测试。

前提条件

- ROSA 经典集群

环境

- 准备环境变量：



注意

更改集群名称以匹配您的 ROSA 集群，并确保您以管理员身份登录到集群。在继续之前，请确保正确输出所有字段。

```
$ export CLUSTER_NAME=$(oc get infrastructure cluster -o=jsonpath="
{.status.infrastructureName}"  | sed 's/-[a-z0-9]\{5\}$//')
$ export ROSA_CLUSTER_ID=$(rosa describe cluster -c ${CLUSTER_NAME} --output json
| jq -r .id)
$ export REGION=$(rosa describe cluster -c ${CLUSTER_NAME} --output json | jq -r
.region.id)
$ export OIDC_ENDPOINT=$(oc get authentication.config.openshift.io cluster -o
jsonpath='{.spec.serviceAccountIssuer}' | sed  's|^https://||')
$ export AWS_ACCOUNT_ID=`aws sts get-caller-identity --query Account --output text`
$ export CLUSTER_VERSION=`rosa describe cluster -c ${CLUSTER_NAME} -o json | jq -r
.version.raw_id | cut -f -2 -d '.'`
$ export ROLE_NAME="${CLUSTER_NAME}-openshift-oadp-aws-cloud-credentials"
$ export AWS_PAGER=""
$ export SCRATCH="/tmp/${CLUSTER_NAME}/oadp"
$ mkdir -p ${SCRATCH}
$ echo "Cluster ID: ${ROSA_CLUSTER_ID}, Region: ${REGION}, OIDC Endpoint:
${OIDC_ENDPOINT}, AWS Account ID: ${AWS_ACCOUNT_ID}"
```

6.1. 准备 AWS 帐户

1. 创建一个 IAM 策略来允许 S3 访问：

```
$ POLICY_ARN=$(aws iam list-policies --query "Policies[?PolicyName=='RosaOadpVer1'].
{ARN:Arn}" --output text)
if [[ -z "${POLICY_ARN}" ]]; then
$ cat << EOF > ${SCRATCH}/policy.json
{
"Version": "2012-10-17",
"Statement": [
 {
   "Effect": "Allow",
   "Action": [
```

```
      "s3:CreateBucket",
      "s3:DeleteBucket",
      "s3:PutBucketTagging",
      "s3:GetBucketTagging",
      "s3:PutEncryptionConfiguration",
      "s3:GetEncryptionConfiguration",
      "s3:PutLifecycleConfiguration",
      "s3:GetLifecycleConfiguration",
      "s3:GetBucketLocation",
      "s3:ListBucket",
      "s3:GetObject",
      "s3:PutObject",
      "s3:DeleteObject",
      "s3:ListBucketMultipartUploads",
      "s3:AbortMultipartUpload",
      "s3:ListMultipartUploadParts",
      "ec2:DescribeSnapshots",
      "ec2:DescribeVolumes",
      "ec2:DescribeVolumeAttribute",
      "ec2:DescribeVolumesModifications",
      "ec2:DescribeVolumeStatus",
      "ec2:CreateTags",
      "ec2:CreateVolume",
      "ec2:CreateSnapshot",
      "ec2:DeleteSnapshot"
    ],
    "Resource": "*"
  }
]}
EOF
$ POLICY_ARN=$(aws iam create-policy --policy-name "RosaOadpVer1" \
--policy-document file:///${SCRATCH}/policy.json --query Policy.Arn \
--tags Key=rosa_openshift_version,Value=${CLUSTER_VERSION}
Key=rosa_role_prefix,Value=ManagedOpenShift
Key=operator_namespace,Value=openshift-oadp Key=operator_name,Value=openshift-oadp
\
--output text)
fi
$ echo ${POLICY_ARN}
```

2. 为集群创建 IAM 角色信任策略：

```
$ cat <<EOF > ${SCRATCH}/trust-policy.json
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": {
      "Federated": "arn:aws:iam::${AWS_ACCOUNT_ID}:oidc-provider/${OIDC_ENDPOINT}"
    },
    "Action": "sts:AssumeRoleWithWebIdentity",
    "Condition": {
      "StringEquals": {
        "${OIDC_ENDPOINT}:sub": [
          "system:serviceaccount:openshift-adp:openshift-adp-controller-manager",
          "system:serviceaccount:openshift-adp:velero"]
```

```
      }
    }
  }]
}
EOF
$ ROLE_ARN=$(aws iam create-role --role-name \
 "${ROLE_NAME}" \
  --assume-role-policy-document file://${SCRATCH}/trust-policy.json \
  --tags Key=rosa_cluster_id,Value=${ROSA_CLUSTER_ID}
Key=rosa_openshift_version,Value=${CLUSTER_VERSION}
Key=rosa_role_prefix,Value=ManagedOpenShift
Key=operator_namespace,Value=openshift-adp Key=operator_name,Value=openshift-oadp \
  --query Role.Arn --output text)

$ echo ${ROLE_ARN}
```

3. 将 IAM 策略附加到 IAM 角色：

```
$ aws iam attach-role-policy --role-name "${ROLE_NAME}" \
 --policy-arn ${POLICY_ARN}
```

6.2. 在集群中部署 OADP

1. 为 OADP 创建命名空间：

```
$ oc create namespace openshift-adp
```

2. 创建凭证 secret：

```
$ cat <<EOF > ${SCRATCH}/credentials
[default]
role_arn = ${ROLE_ARN}
web_identity_token_file = /var/run/secrets/openshift/serviceaccount/token
EOF
$ oc -n openshift-adp create secret generic cloud-credentials \
 --from-file=${SCRATCH}/credentials
```

3. 部署 OADP Operator：

注意

目前，Operator 版本 1.1 存在一个带有 **PartiallyFailed** 状态的备份版本 1.1 的问题。这不会影响备份和恢复过程，但应注意，因为它存在问题。

```
$ cat << EOF | oc create -f -
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
 generateName: openshift-adp-
 namespace: openshift-adp
 name: oadp
spec:
 targetNamespaces:
```

```
 - openshift-adp
---
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
 name: redhat-oadp-operator
 namespace: openshift-adp
spec:
 channel: stable-1.2
 installPlanApproval: Automatic
 name: redhat-oadp-operator
 source: redhat-operators
 sourceNamespace: openshift-marketplace
EOF
```

4. 等待 Operator 就绪：

```
$ watch oc -n openshift-adp get pods
```

输出示例

```
NAME                                READY   STATUS    RESTARTS   AGE
openshift-adp-controller-manager-546684844f-qqjhn   1/1     Running   0          22s
```

5. 创建云存储：

```
$ cat << EOF | oc create -f -
apiVersion: oadp.openshift.io/v1alpha1
kind: CloudStorage
metadata:
 name: ${CLUSTER_NAME}-oadp
 namespace: openshift-adp
spec:
 creationSecret:
   key: credentials
   name: cloud-credentials
 enableSharedConfig: true
 name: ${CLUSTER_NAME}-oadp
 provider: aws
 region: $REGION
EOF
```

6. 检查应用程序的存储默认存储类：

```
$ oc get pvc -n <namespace> 1
```

1 输入应用程序的命名空间。

输出示例

```
NAME     STATUS   VOLUME                                     CAPACITY   ACCESS MODES
STORAGECLASS   AGE
applog   Bound    pvc-351791ae-b6ab-4e8b-88a4-30f73caf5ef8   1Gi        RWO            gp3-
```

```
csi        4d19h
mysql   Bound    pvc-16b8e009-a20a-4379-accc-bc81fedd0621   1Gi        RWO            gp3-
csi        4d19h
```

```
$ oc get storageclass
```

输出示例

```
NAME                PROVISIONER             RECLAIMPOLICY   VOLUMEBINDINGMODE
ALLOWVOLUMEEXPANSION   AGE
gp2                 kubernetes.io/aws-ebs   Delete          WaitForFirstConsumer   true
4d21h
gp2-csi             ebs.csi.aws.com         Delete          WaitForFirstConsumer   true
4d21h
gp3                 ebs.csi.aws.com         Delete          WaitForFirstConsumer   true
4d21h
gp3-csi (default)   ebs.csi.aws.com         Delete          WaitForFirstConsumer   true
4d21h
```

使用 gp3-csi, gp2-csi, gp3 或 gp2 将可以正常工作。如果要备份的应用程序都使用带有 CSI 的 PV，请在 OADP DPA 配置中包含 CSI 插件。

7. 仅限 CSI：部署数据保护应用程序：

```
$ cat << EOF | oc create -f -
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
 name: ${CLUSTER_NAME}-dpa
 namespace: openshift-adp
spec:
 backupImages: true
 features:
   dataMover:
     enable: false
 backupLocations:
 - bucket:
     cloudStorageRef:
       name: ${CLUSTER_NAME}-oadp
     credential:
       key: credentials
       name: cloud-credentials
     prefix: velero
     default: true
     config:
       region: ${REGION}
 configuration:
   velero:
     defaultPlugins:
     - openshift
     - aws
     - csi
   restic:
     enable: false
EOF
```

注意

如果为 CSI 卷运行这个命令，您可以跳过下一步。

8. 非 CSI 卷：部署数据保护应用程序：

```
$ cat << EOF | oc create -f -
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
 name: ${CLUSTER_NAME}-dpa
 namespace: openshift-adp
spec:
 backupImages: true
 features:
   dataMover:
     enable: false
 backupLocations:
 - bucket:
     cloudStorageRef:
       name: ${CLUSTER_NAME}-oadp
     credential:
       key: credentials
       name: cloud-credentials
     prefix: velero
     default: true
     config:
       region: ${REGION}
 configuration:
   velero:
     defaultPlugins:
     - openshift
     - aws
   restic:
     enable: false
 snapshotLocations:
   - velero:
       config:
         credentialsFile: /tmp/credentials/openshift-adp/cloud-credentials-credentials
         enableSharedConfig: 'true'
         profile: default
         region: ${REGION}
       provider: aws
EOF
```

注意

- 在 OADP 1.1.x ROSA STS 环境中，容器镜像备份和恢复(**spec.backupImages**)值必须设置为 **false**，因为它不被支持。
- Restic 功能(**restic.enable=false**)被禁用，在 ROSA STS 环境中不支持。
- DataMover 功能(**dataMover.enable=false**)被禁用，在 ROSA STS 环境中不支持。

6.3. 执行备份

注意

以下示例 hello-world 应用没有附加的持久性卷。DPA 配置都将正常工作。

1. 创建要备份的工作负载：

```
$ oc create namespace hello-world
$ oc new-app -n hello-world --image=docker.io/openshift/hello-openshift
```

2. 公开路由：

```
$ oc expose service/hello-openshift -n hello-world
```

3. 检查应用程序是否正常工作：

```
$ curl `oc get route/hello-openshift -n hello-world -o jsonpath='{.spec.host}'`
```

输出示例

```
Hello OpenShift!
```

4. 备份工作负载：

```
$ cat << EOF | oc create -f -
apiVersion: velero.io/v1
kind: Backup
metadata:
 name: hello-world
 namespace: openshift-adp
spec:
 includedNamespaces:
 - hello-world
 storageLocation: ${CLUSTER_NAME}-dpa-1
 ttl: 720h0m0s
EOF
```

5. 等待备份完成：

```
$ watch "oc -n openshift-adp get backup hello-world -o json | jq .status"
```

输出示例

```
{
 "completionTimestamp": "2022-09-07T22:20:44Z",
 "expiration": "2022-10-07T22:20:22Z",
 "formatVersion": "1.1.0",
 "phase": "Completed",
 "progress": {
   "itemsBackedUp": 58,
   "totalItems": 58
 },
 "startTimestamp": "2022-09-07T22:20:22Z",
 "version": 1
}
```

6. 删除 demo 工作负载：

```
$ oc delete ns hello-world
```

7. 从备份中恢复：

```
$ cat << EOF | oc create -f -
apiVersion: velero.io/v1
kind: Restore
metadata:
 name: hello-world
 namespace: openshift-adp
spec:
 backupName: hello-world
EOF
```

8. 等待 Restore 完成：

```
$ watch "oc -n openshift-adp get restore hello-world -o json | jq .status"
```

输出示例

```
{
 "completionTimestamp": "2022-09-07T22:25:47Z",
 "phase": "Completed",
 "progress": {
   "itemsRestored": 38,
   "totalItems": 38
 },
 "startTimestamp": "2022-09-07T22:25:28Z",
 "warnings": 9
}
```

9. 检查工作负载是否已恢复：

```
$ oc -n hello-world get pods
```

输出示例

```
NAME                                READY   STATUS    RESTARTS   AGE
hello-openshift-9f885f7c6-kdjpj   1/1     Running   0          90s
```

```
$ curl `oc get route/hello-openshift -n hello-world -o jsonpath='{.spec.host}'`
```

输出示例

```
Hello OpenShift!
```

10. 有关故障排除提示的信息，请参阅 OADP 团队的 故障排除文档
11. 其他示例应用程序可以在 OADP 团队 的示例应用程序目录中找到

6.4. CLEANUP

1. 删除工作负载：

```
$ oc delete ns hello-world
```

2. 如果不再需要，从集群中删除备份和恢复资源：

```
$ oc delete backup hello-world
$ oc delete restore hello-world
```

3. 删除 s3 中的备份/恢复和远程对象：

```
$ velero backup delete hello-world
$ velero restore delete hello-world
```

4. 删除数据保护应用程序：

```
$ oc -n openshift-adp delete dpa ${CLUSTER_NAME}-dpa
```

5. 删除云存储：

```
$ oc -n openshift-adp delete cloudstorage ${CLUSTER_NAME}-oadp
```

警告

如果这个命令挂起，您可能需要删除终结器：

```
$ oc -n openshift-adp patch cloudstorage ${CLUSTER_NAME}-oadp -p
'{"metadata":{"finalizers":null}}' --type=merge
```

6. 如果不再需要，删除 Operator：

```
$ oc -n openshift-adp delete subscription oadp-operator
```

7. 删除 Operator 的命名空间：

```
$ oc delete ns redhat-openshift-adp
```

8. 如果不再有它们，请从集群中删除自定义资源定义：

```
$ for CRD in `oc get crds | grep velero | awk '{print $1}'`; do oc delete crd $CRD; done
$ for CRD in `oc get crds | grep -i oadp | awk '{print $1}'`; do oc delete crd $CRD; done
```

9. 删除 AWS S3 Bucket：

```
$ aws s3 rm s3://${CLUSTER_NAME}-oadp --recursive
$ aws s3api delete-bucket --bucket ${CLUSTER_NAME}-oadp
```

10. 将 Policy 从角色分离：

```
$ aws iam detach-role-policy --role-name "${ROLE_NAME}" \
 --policy-arn "${POLICY_ARN}"
```

11. 删除角色：

```
$ aws iam delete-role --role-name "${ROLE_NAME}"
```

第 7 章 教程：AWS LOAD BALANCER OPERATOR ON ROSA

重要

此内容由红帽专家编写，但尚未在所有支持的配置中进行测试。

提示

AWS Load Balancer Operator 创建的负载均衡器不能用于 OpenShift 路由，并且只应用于不需要 OpenShift Route 完整第 7 层功能的单个服务或入口资源。

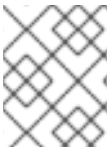
AWS Load Balancer Controller 为 AWS (ROSA)集群管理 Red Hat OpenShift Service 的 AWS Elastic Load Balancers。在使用类型为 LoadBalancer 的 Kubernetes Service 资源时，控制器在创建 Kubernetes Ingress 资源和 AWS Network Load Balancers (NLB) 时置备 AWS Application Load Balancers (ALB)。

与默认的 AWS 树内负载均衡器供应商相比，这个控制器使用 ALB 和 NLBs 的高级注解开发。一些高级用例是：

- 使用带有 ALB 的原生 Kubernetes Ingress 对象
- 将 ALB 与 AWS Web Application Firewall (WAF)服务集成
- 指定自定义 NLB 源 IP 范围
- 指定自定义 NLB 内部 IP 地址

AWS Load Balancer Operator 用于在 ROSA 集群中安装、管理和配置 **aws-load-balancer-controller** 实例。

7.1. 前提条件



注意

AWS ALB 需要 Multi-AZ 集群，以及与集群相同的 VPC 中的三个公共子网分割。这使得 ALB 不适用于许多 PrivateLink 集群。AWS NLBs 没有这个限制。

- 多 AZ ROSA 经典集群
- BYO VPC 集群
- AWS CLI
- OC CLI

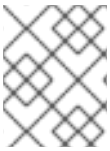
7.1.1. 环境

- 准备环境变量：

```
$ export AWS_PAGER=""
$ export ROSA_CLUSTER_NAME=$(oc get infrastructure cluster -o=jsonpath="
{.status.infrastructureName}"  | sed 's/-[a-z0-9]\{5\}$//')
$ export REGION=$(oc get infrastructure cluster -o=jsonpath="
{.status.platformStatus.aws.region}")
```

```
$ export OIDC_ENDPOINT=$(oc get authentication.config.openshift.io cluster -o
jsonpath='{.spec.serviceAccountIssuer}' | sed  's|^https://||')
$ export AWS_ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
$ export SCRATCH="/tmp/${ROSA_CLUSTER_NAME}/alb-operator"
$ mkdir -p ${SCRATCH}
$ echo "Cluster: ${ROSA_CLUSTER_NAME}, Region: ${REGION}, OIDC Endpoint:
${OIDC_ENDPOINT}, AWS Account ID: ${AWS_ACCOUNT_ID}"
```

7.1.2. AWS VPC 和子网



注意

本节只适用于部署到现有 VPC 的集群。如果您没有将集群部署到现有的 VPC 中，请跳过本节并继续以下安装部分。

1. 将以下变量设置为您的 ROSA 部署的正确值：

```
$ export VPC_ID=<vpc-id>
$ export PUBLIC_SUBNET_IDS=<public-subnets>
$ export PRIVATE_SUBNET_IDS=<private-subnets>
$ export CLUSTER_NAME=$(oc get infrastructure cluster -o=jsonpath="
{.status.infrastructureName}")
```

2. 使用集群名称在集群 VPC 中添加标签：

```
$ aws ec2 create-tags --resources ${VPC_ID} --tags
Key=kubernetes.io/cluster/${CLUSTER_NAME},Value=owned --region ${REGION}
```

3. 在您的公共子网中添加标签：

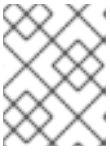
```
$ aws ec2 create-tags \
    --resources ${PUBLIC_SUBNET_IDS} \
    --tags Key=kubernetes.io/role/elb,Value='' \
    --region ${REGION}
```

4. 在您的私有子网中添加标签：

```
$ aws ec2 create-tags \
    --resources "${PRIVATE_SUBNET_IDS}" \
    --tags Key=kubernetes.io/role/internal-elb,Value='' \
    --region ${REGION}
```

7.2. 安装

1. 为 AWS Load Balancer Controller 创建 AWS IAM 策略：



注意

该策略 来自上游 AWS Load Balancer Controller 策略，以及在子网上创建标签的权限。Operator 需要它才能正常工作。

```
$ oc new-project aws-load-balancer-operator
$ POLICY_ARN=$(aws iam list-policies --query \
    "Policies[?PolicyName=='aws-load-balancer-operator-policy'].{ARN:Arn}" \
    --output text)
$ if [[ -z "${POLICY_ARN}" ]]; then
   wget -O "${SCRATCH}/load-balancer-operator-policy.json" \
     https://raw.githubusercontent.com/rh-mobb/documentation/main/content/rosa/aws-load-balancer-operator/load-balancer-operator-policy.json
    POLICY_ARN=$(aws --region "$REGION" --query Policy.Arn \
    --output text iam create-policy \
    --policy-name aws-load-balancer-operator-policy \
    --policy-document "file://${SCRATCH}/load-balancer-operator-policy.json")
fi
$ echo $POLICY_ARN
```

2. 为 AWS Load Balancer Operator 创建 AWS IAM 信任策略：

```
$ cat <<EOF > "${SCRATCH}/trust-policy.json"
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Condition": {
   "StringEquals" : {
    "${OIDC_ENDPOINT}:sub": ["system:serviceaccount:aws-load-balancer-operator:aws-load-balancer-operator-controller-manager", "system:serviceaccount:aws-load-balancer-operator:aws-load-balancer-controller-cluster"]
   }
 },
 "Principal": {
   "Federated": "arn:aws:iam::$AWS_ACCOUNT_ID:oidc-provider/${OIDC_ENDPOINT}"
 },
 "Action": "sts:AssumeRoleWithWebIdentity"
 }
 ]
}
EOF
```

3. 为 AWS Load Balancer Operator 创建 AWS IAM 角色：

```
$ ROLE_ARN=$(aws iam create-role --role-name "${ROSA_CLUSTER_NAME}-alb-operator" \
   --assume-role-policy-document "file://${SCRATCH}/trust-policy.json" \
   --query Role.Arn --output text)
$ echo $ROLE_ARN

$ aws iam attach-role-policy --role-name "${ROSA_CLUSTER_NAME}-alb-operator" \
    --policy-arn $POLICY_ARN
```

4. 为 AWS Load Balancer Operator 创建 secret，以假定我们新创建的 AWS IAM 角色：

```
$ cat << EOF | oc apply -f -
apiVersion: v1
kind: Secret
```

```
metadata:
  name: aws-load-balancer-operator
  namespace: aws-load-balancer-operator
stringData:
  credentials: |
    [default]
    role_arn = $ROLE_ARN
    web_identity_token_file = /var/run/secrets/openshift/serviceaccount/token
EOF
```

5. 安装 AWS Load Balancer Operator：

```
$ cat << EOF | oc apply -f -
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: aws-load-balancer-operator
  namespace: aws-load-balancer-operator
spec:
  upgradeStrategy: Default
---
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: aws-load-balancer-operator
  namespace: aws-load-balancer-operator
spec:
  channel: stable-v1.0
  installPlanApproval: Automatic
  name: aws-load-balancer-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
  startingCSV: aws-load-balancer-operator.v1.0.0
EOF
```

6. 使用 Operator 部署 AWS Load Balancer Controller 实例：

注意

如果您在此处收到错误并尝试重试，这意味着 Operator 还没有完成安装。

```
$ cat << EOF | oc apply -f -
apiVersion: networking.olm.openshift.io/v1
kind: AWSLoadBalancerController
metadata:
  name: cluster
spec:
  credentials:
    name: aws-load-balancer-operator
EOF
```

7. 检查 Operator 和控制器 pod 是否都在运行：

```
$ oc -n aws-load-balancer-operator get pods
```

如果没有等待片刻并重试，您应该看到以下内容：

```
NAME                                                READY   STATUS    RESTARTS   AGE
aws-load-balancer-controller-cluster-6ddf658785-pdp5d            1/1     Running   0          99s
aws-load-balancer-operator-controller-manager-577d9ffcb9-w6zqn   2/2     Running   0
2m4s
```

7.3. 验证部署

1. 创建一个新项目

```
$ oc new-project hello-world
```

2. 部署 hello world 应用：

```
$ oc new-app -n hello-world --image=docker.io/openshift/hello-openshift
```

3. 为 AWS ALB 配置 NodePort 服务以连接：

```
$ cat << EOF | oc apply -f -
apiVersion: v1
kind: Service
metadata:
  name: hello-openshift-nodeport
  namespace: hello-world
spec:
  ports:
    - port: 80
      targetPort: 8080
      protocol: TCP
  type: NodePort
  selector:
    deployment: hello-openshift
EOF
```

4. 使用 AWS Load Balancer Operator 部署 AWS ALB：

```
$ cat << EOF | oc apply -f -
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: hello-openshift-alb
  namespace: hello-world
  annotations:
    alb.ingress.kubernetes.io/scheme: internet-facing
spec:
  ingressClassName: alb
  rules:
    - http:
        paths:
          - path: /
            pathType: Exact
            backend:
```

```
            service:
              name: hello-openshift-nodeport
              port:
                number: 80
EOF
```

5. curl AWS ALB Ingress 端点，以验证 hello world 应用程序是否可访问：

注意

AWS ALB 置备需要几分钟。如果您收到显示 **curl: (6) Could not resolve host** 的错误，请等待再试一次。

```
$ INGRESS=$(oc -n hello-world get ingress hello-openshift-alb \
    -o jsonpath='{.status.loadBalancer.ingress[0].hostname}')
$ curl "http://${INGRESS}"
```

输出示例

```
Hello OpenShift!
```

6. 为您的 hello world 应用程序部署 AWS NLB ：

```
$ cat << EOF | oc apply -f -
apiVersion: v1
kind: Service
metadata:
  name: hello-openshift-nlb
  namespace: hello-world
  annotations:
    service.beta.kubernetes.io/aws-load-balancer-type: external
    service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: instance
    service.beta.kubernetes.io/aws-load-balancer-scheme: internet-facing
spec:
  ports:
    - port: 80
      targetPort: 8080
      protocol: TCP
  type: LoadBalancer
  selector:
    deployment: hello-openshift
EOF
```

7. 测试 AWS NLB 端点：

注意

NLB 置备需要几分钟时间。如果您收到显示 **curl: (6) Could not resolve host** 的错误，请等待再试一次。

```
$ NLB=$(oc -n hello-world get service hello-openshift-nlb \
  -o jsonpath='{.status.loadBalancer.ingress[0].hostname}')
$ curl "http://${NLB}"
```

输出示例

```
Hello OpenShift!
```

7.4. 清理

1. 删除 hello world 应用命名空间（以及命名空间中的所有资源）：

```
$ oc delete project hello-world
```

2. 删除 AWS Load Balancer Operator 和 AWS IAM 角色：

```
$ oc delete subscription aws-load-balancer-operator -n aws-load-balancer-operator
$ aws iam detach-role-policy \
  --role-name "${ROSA_CLUSTER_NAME}-alb-operator" \
  --policy-arn $POLICY_ARN
$ aws iam delete-role \
  --role-name "${ROSA_CLUSTER_NAME}-alb-operator"
```

3. 删除 AWS IAM 策略：

```
$ aws iam delete-policy --policy-arn $POLICY_ARN
```

第 8 章 教程：将 MICROSOFT ENTRA ID （以前称为 AZURE ACTIVE DIRECTORY）配置为身份提供程序

您可以将 Microsoft Entra ID （以前称为 Azure Active Directory）配置为 Red Hat OpenShift Service on AWS (ROSA)中的集群身份提供程序。

本教程指导您完成以下任务：

1. 在 Entra ID 中注册新应用程序进行验证。
2. 在 Entra ID 中配置应用程序注册，以在令牌中包含可选和组声明。
3. 将 Red Hat OpenShift Service on AWS 集群配置为使用 Entra ID 作为身份提供程序。
4. 为各个组授予额外权限。

8.1. 前提条件

- 您已按照 Microsoft 文档 创建了一组安全组并分配用户。

8.2. 在 ENTRA ID 中注册新应用程序进行验证

要在 Entra ID 中注册您的应用程序，首先创建 OAuth 回调 URL，然后注册您的应用程序。

流程

1. 通过更改指定的变量并运行以下命令来创建集群的 OAuth 回调 URL：

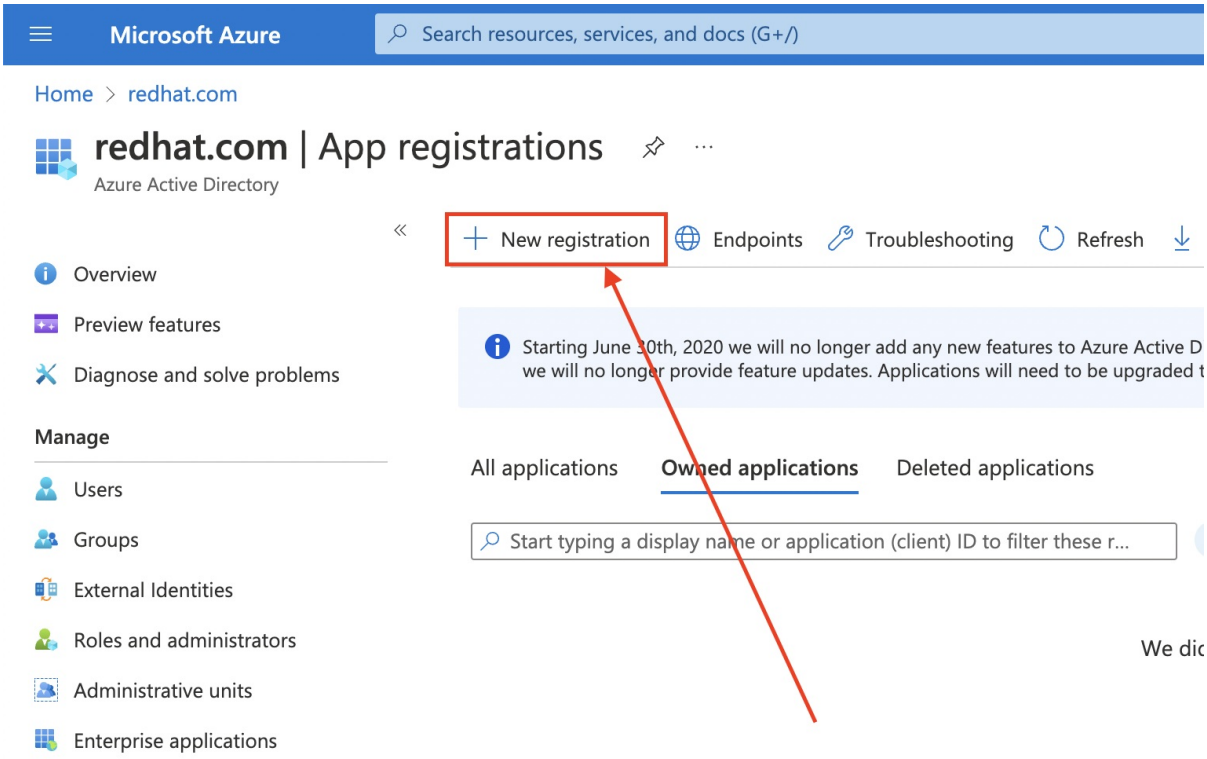
注意

请记住保存此回调 URL；之后需要用到。

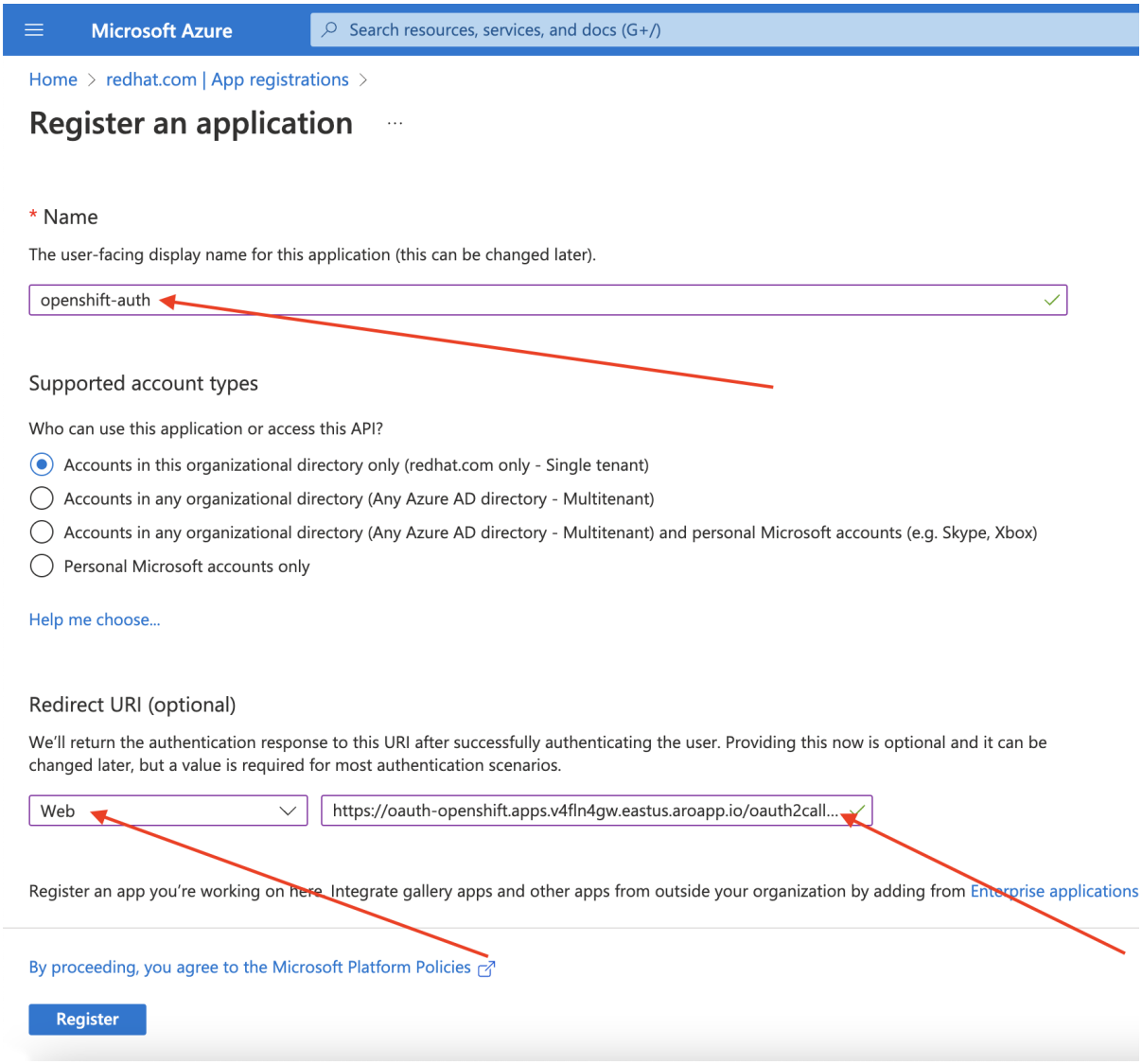
```
$ domain=$(rosa describe cluster -c <cluster_name> | grep "DNS" | grep -oE
'\S+.openshiftapps.com')
$ echo "OAuth callback URL: https://oauth-openshift.apps.$domain/oauth2callback/AAD"
```

OAuth 回调 URL 末尾的"AAD"目录必须与稍后在此过程中设置的 OAuth 身份提供程序名称匹配。

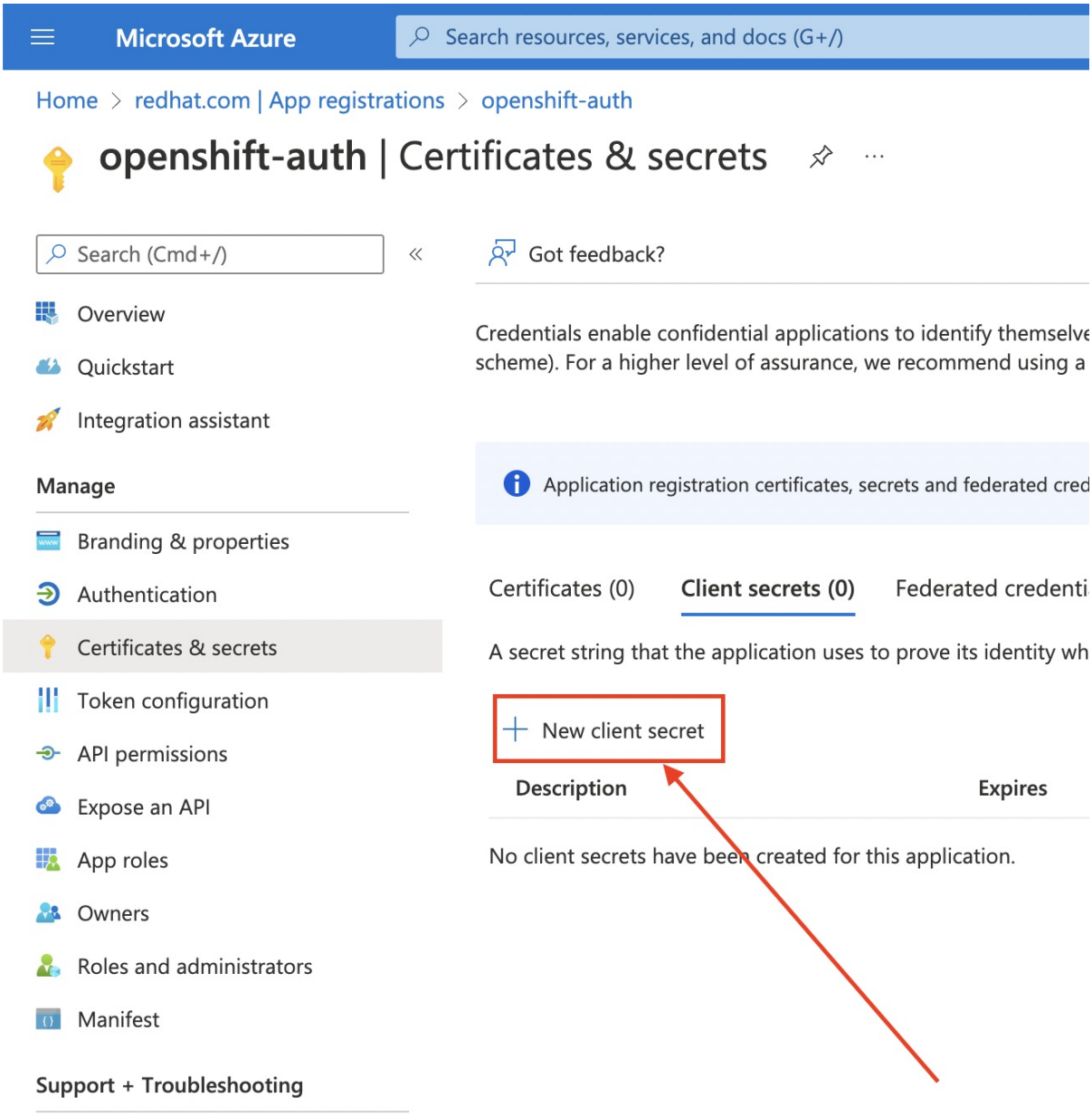
2. 通过登录到 Azure 门户来创建 Entra ID 应用，然后选择 App registrations Blade。然后，选择 **New registration** 来创建新应用程序。



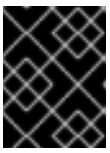
3. 将应用程序命名为，如 **openshift-auth**。
4. 从 *Redirect URI* 下拉菜单中选择 **Web**，并输入您在上一步中检索的 OAuth 回调 URL 的值。
5. 提供所需信息后，点 **Register** 创建应用程序。



6. 选择 **Certificates & secrets** sub-blade 并选择 **New client secret**。

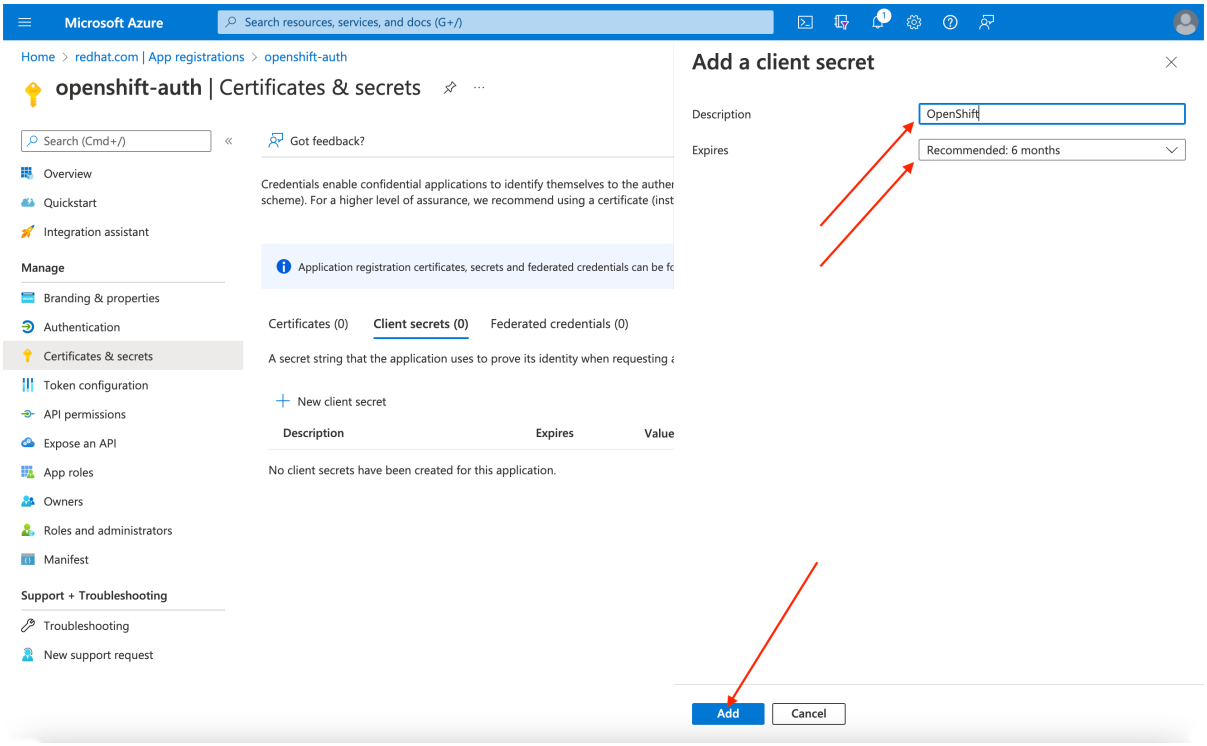


7. 完成请求的详细信息，并存储生成的客户端 secret 值。此过程稍后需要此 secret。

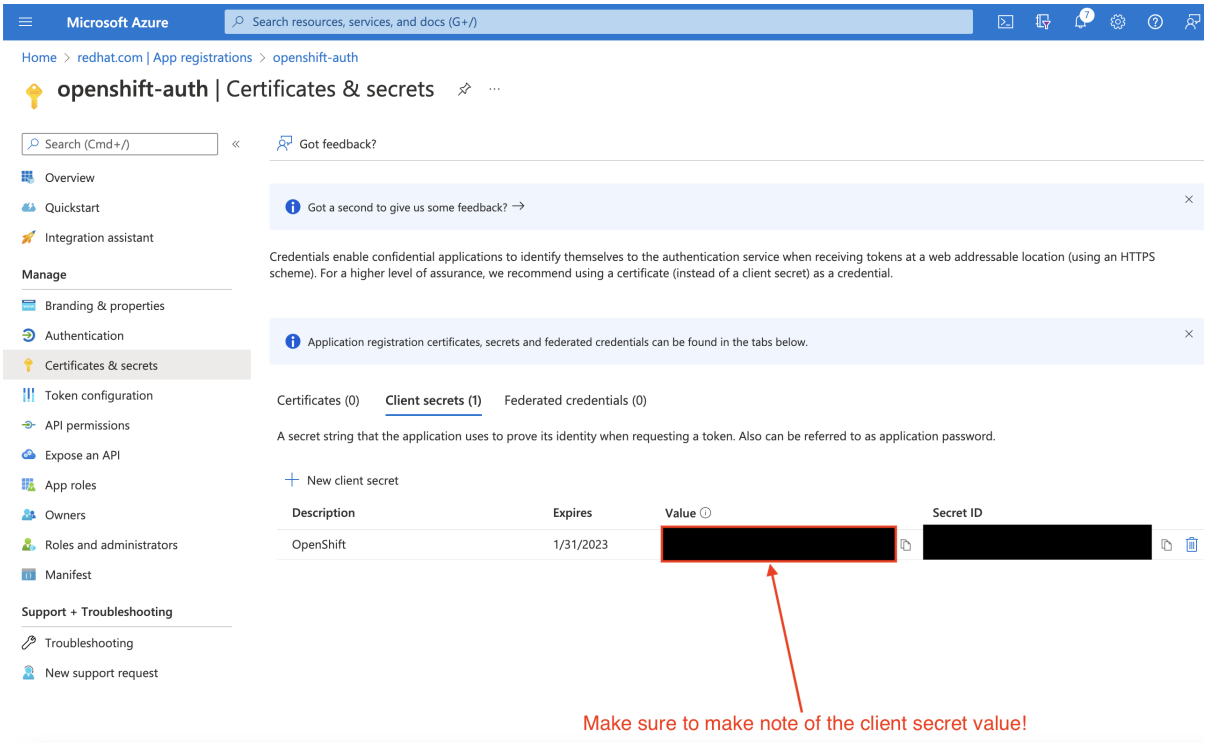


重要

初始设置后，您无法看到客户端 secret。如果没有记录客户端 secret，则必须生成新的 secret。



8. 选择 Overview 子组合并记下 **应用程序（客户端）ID** 和目录 **（租户）ID**。以后的步骤中您将需要这些值。



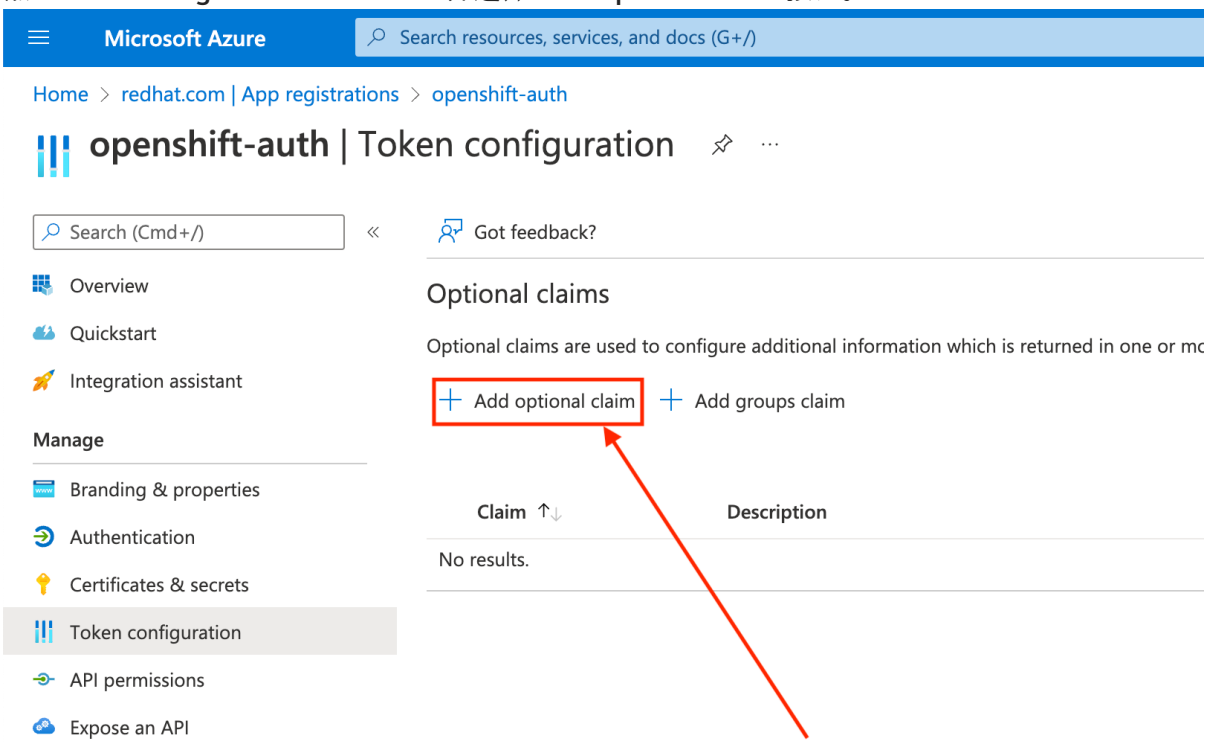
8.3. 在 ENTRA ID 中配置应用程序注册，使其包含可选和组声明

因此，AWS 上的 Red Hat OpenShift Service 有充足的信息来创建用户帐户，您必须配置 Entra ID，以提供两个可选声明： **email** 和 **preferred_username**。有关 Entra ID 中的可选声明的更多信息，请参阅 Microsoft 文档。

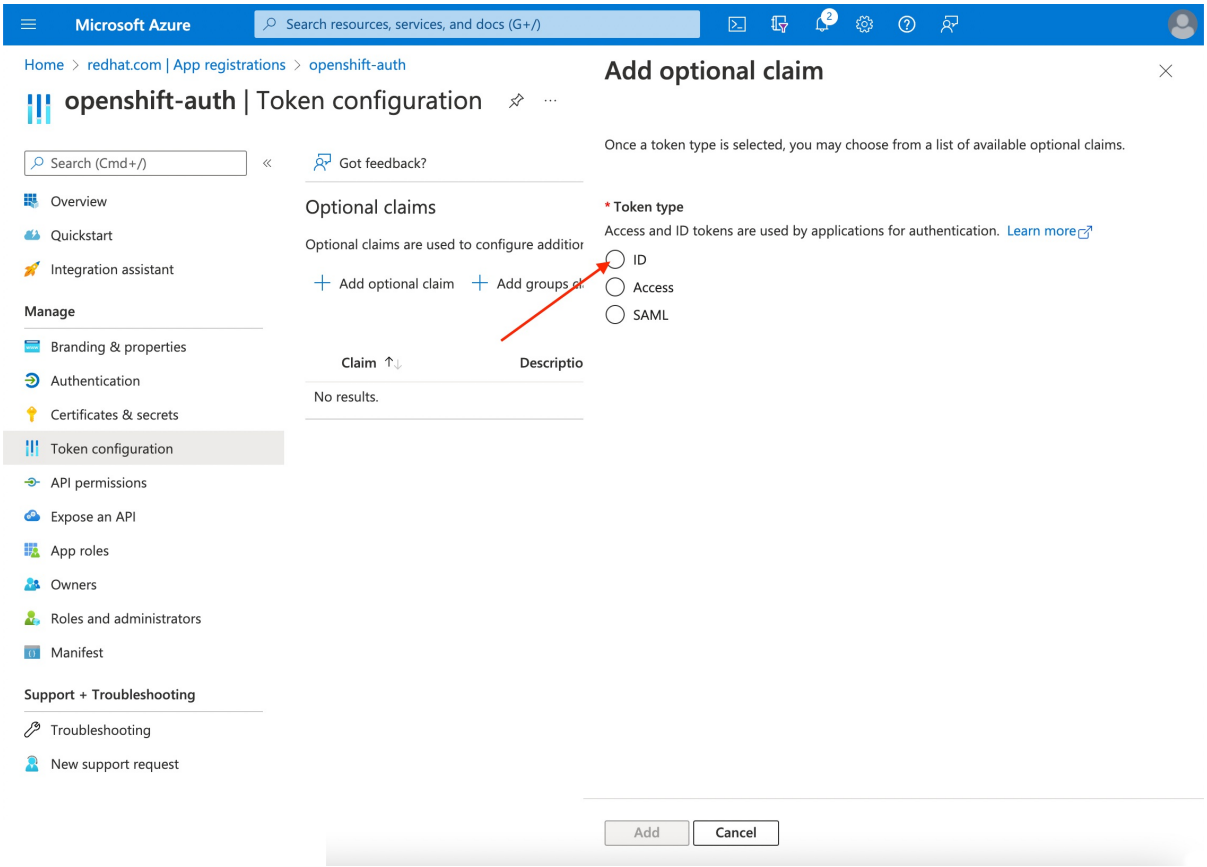
除了单独的用户身份验证外，Red Hat OpenShift Service on AWS 还提供组声明功能。此功能允许 OpenID Connect (OIDC)身份提供程序（如 Entra ID）提供用户的组成员资格，以便在 AWS 上的 Red Hat OpenShift Service 中使用。

配置可选声明
您可以在 Entra ID 中配置可选声明。

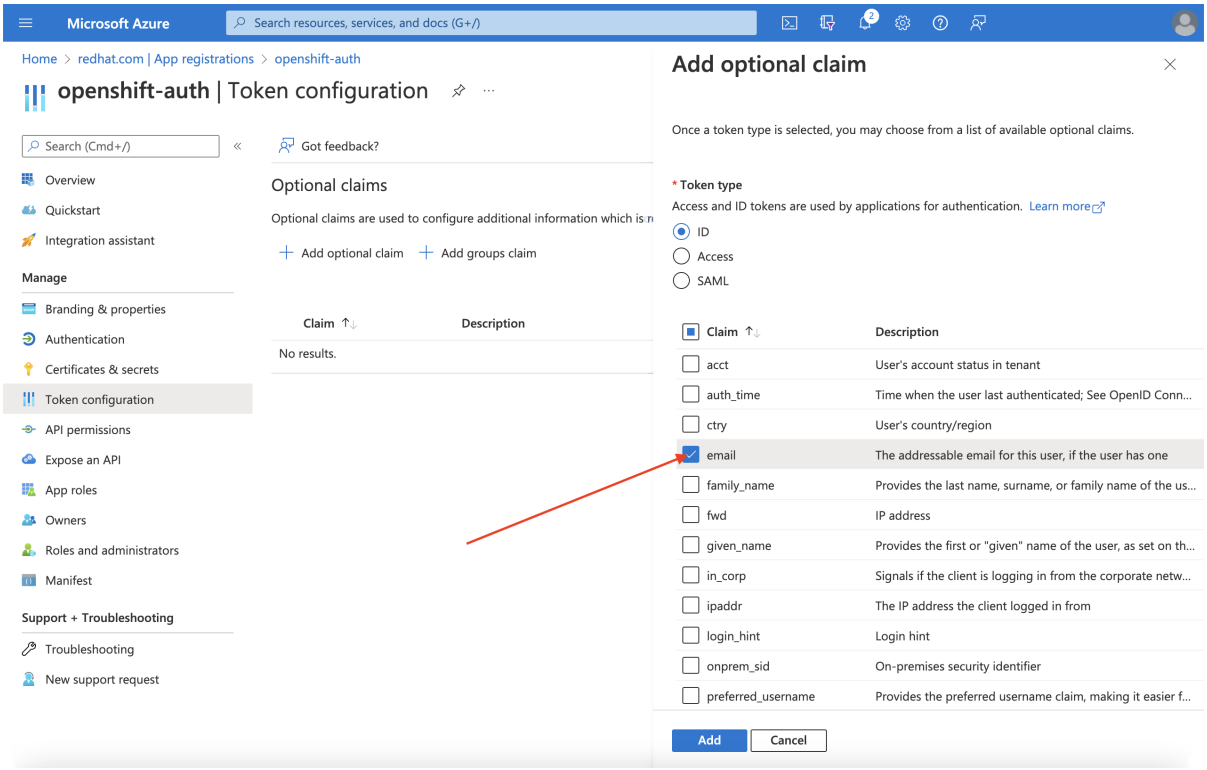
1. 点 **Token configuration** sub-blade 并选择 **Add optional claim** 按钮。



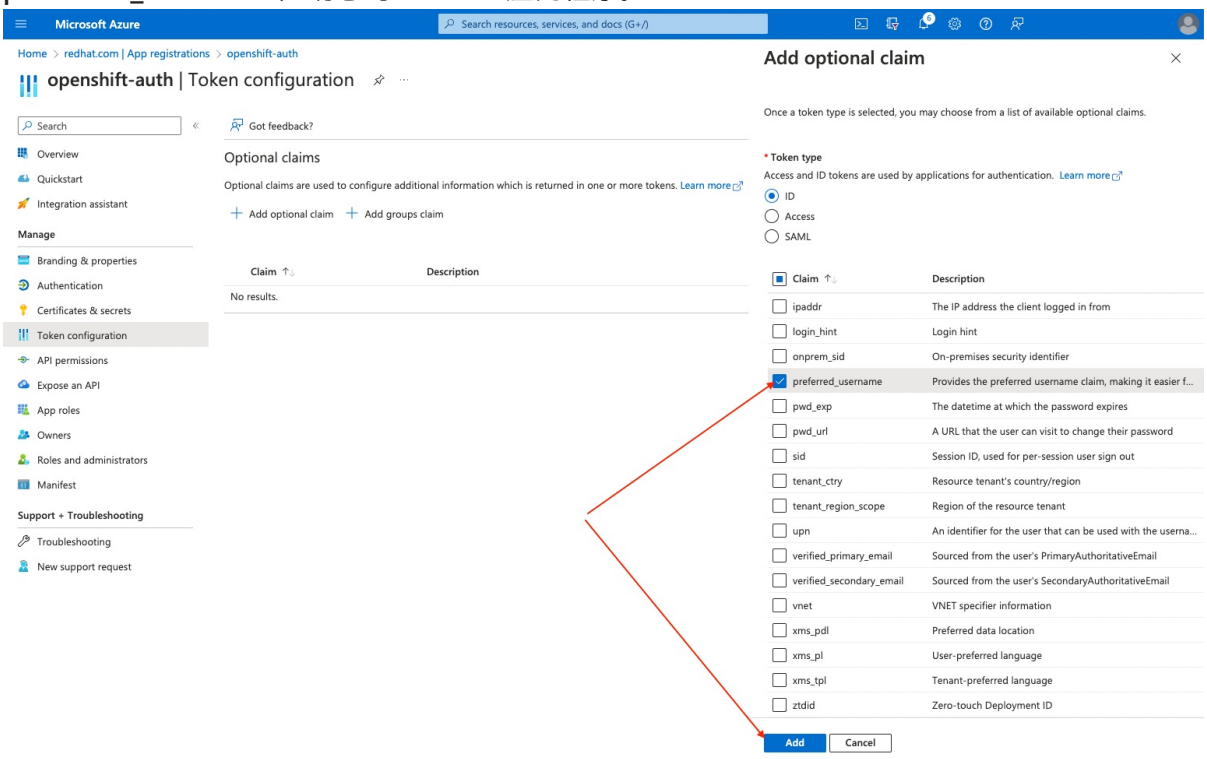
2. 选择 **ID** 单选按钮。



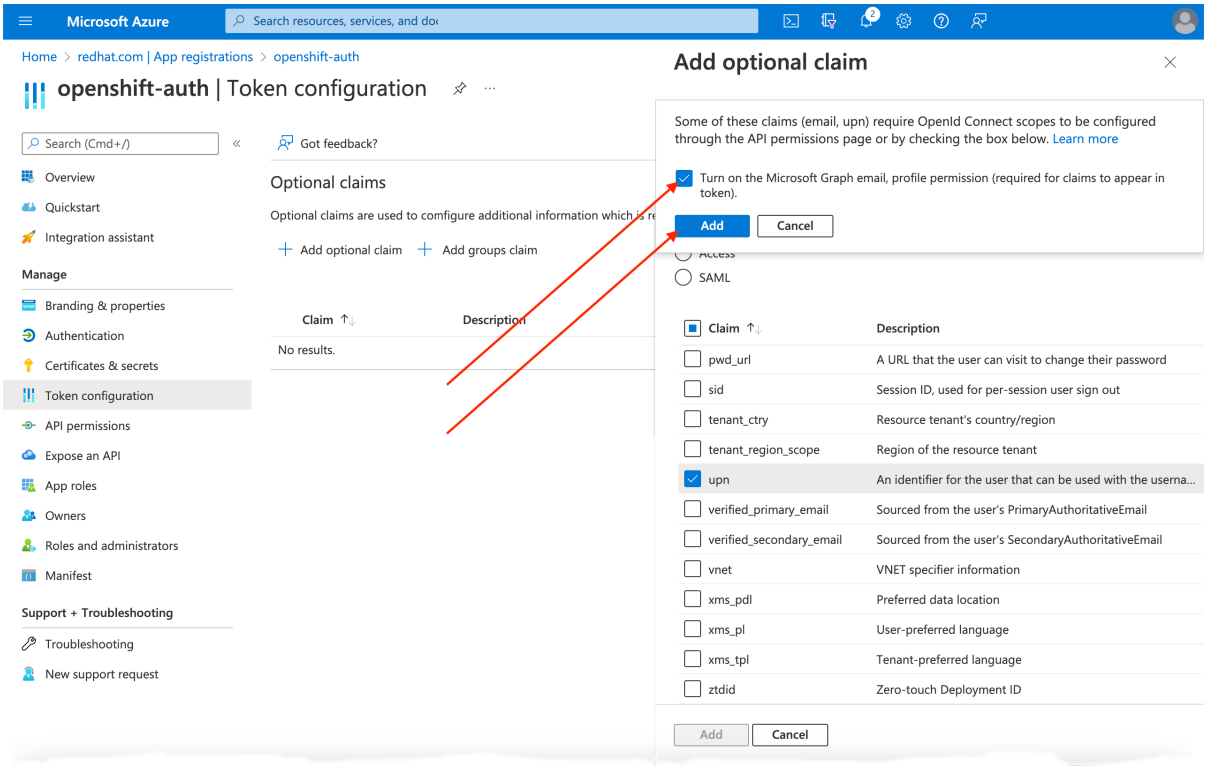
3. 选中 **电子邮件** 声明复选框。



4. 选中 **preferred_username** 声明复选框。然后，点 **Add** 来配置 **电子邮件和 preferred_username** 声明您的 Entra ID 应用程序。



5. 页面的顶部会出现一个对话框。按照提示启用必要的 Microsoft Graph 权限。

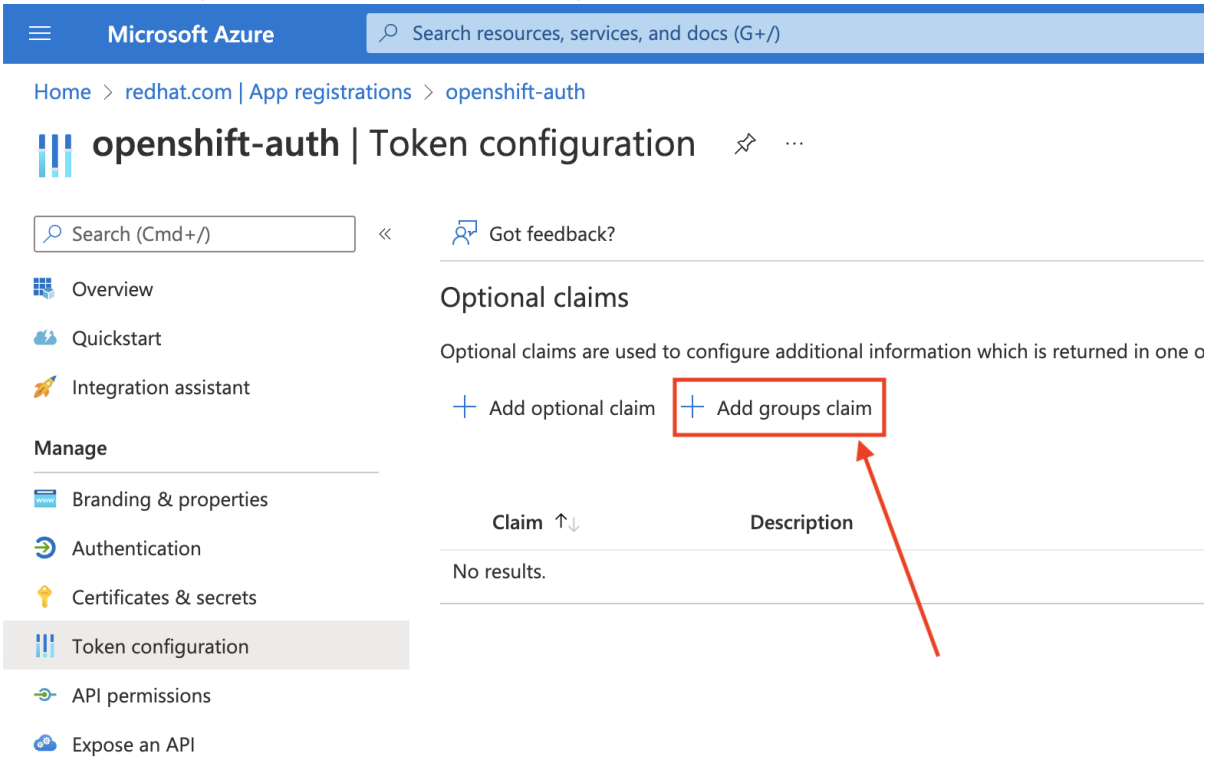


配置组声明（可选）

配置 Entra ID 以提供组声明。

流程

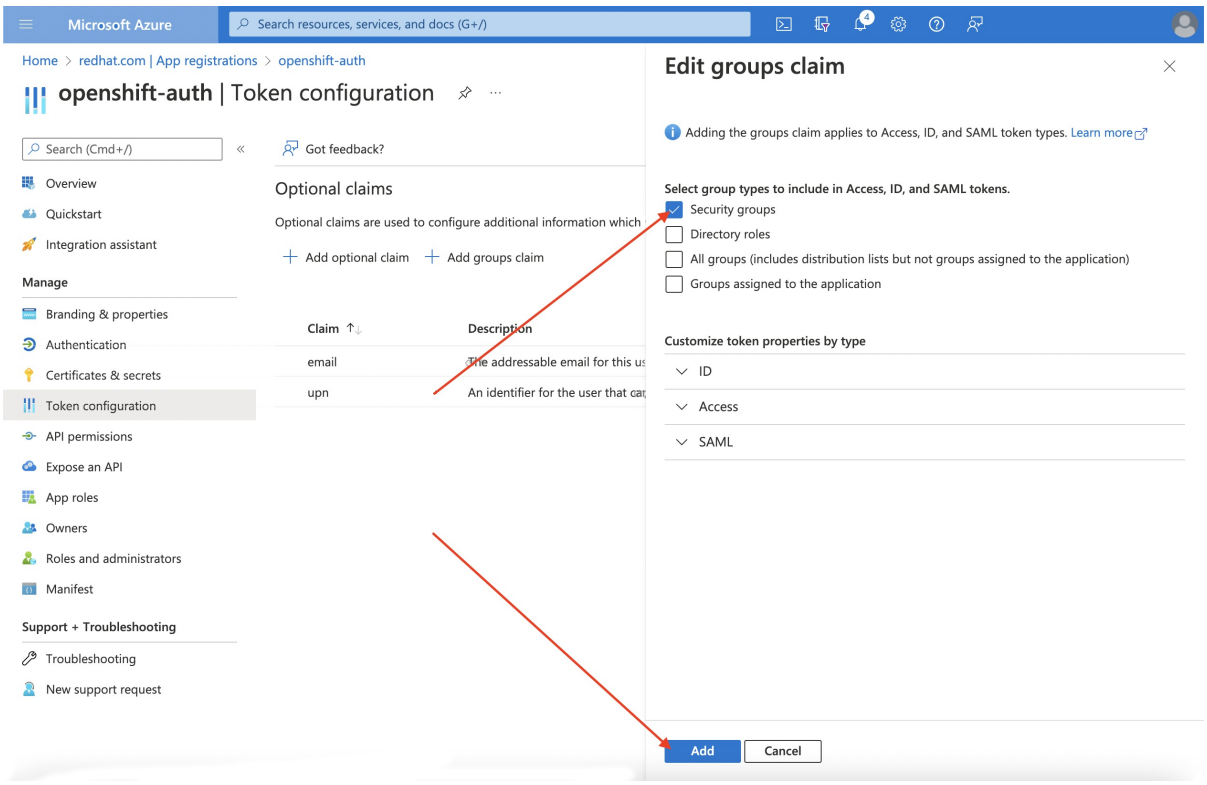
1. 在 **Token configuration** sub-blade 中点 **Add groups claim**。



2. 要为您的 Entra ID 应用程序配置组声明，请选择 **Security groups**，然后点 **Add**。

注意

在本例中，组声明包含用户所属的所有安全组。在真实生产环境中，确保组声明仅包含适用于 Red Hat OpenShift Service on AWS 的组。



8.4. 配置 RED HAT OPENSHIFT SERVICE ON AWS 集群以使用 ENTRA ID 作为身份提供程序

您必须将 Red Hat OpenShift Service on AWS 配置为使用 Entra ID 作为其身份提供程序。

虽然 ROSA 提供了使用 OpenShift Cluster Manager 配置身份提供程序的功能，但 ROSA CLI 使用 ROSA CLI 将集群的 OAuth 供应商配置为使用 Entra ID 作为其身份提供程序。在配置身份提供程序前，为身份提供程序配置设置必要的变量。

流程

1. 运行以下命令来创建变量：

```
$ CLUSTER_NAME=example-cluster 1
$ IDP_NAME=AAD 2
$ APP_ID=yyyyyyyy-yyyy-yyyy-yyyy-yyyyyyyyyyyy 3
$ CLIENT_SECRET=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx 4
$ TENANT_ID=zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzzzzzz 5
```

1. 使用 ROSA 集群的名称替换它。
2. 将此值替换为您之前在此过程中生成的 OAuth 回调 URL 中使用的名称。
3. 使用应用程序（客户端）ID 替换它。
4. 使用 Client Secret 替换它。

5 使用目录（租户）ID 替换它。

2. 运行以下命令来配置集群的 OAuth 提供程序。如果启用了组声明，请确保使用 **--group-claims groups** 参数。

- 如果启用了组声明，请运行以下命令：

```
$ rosa create idp \
--cluster ${CLUSTER_NAME} \
--type openid \
--name ${IDP_NAME} \
--client-id ${APP_ID} \
--client-secret ${CLIENT_SECRET} \
--issuer-url https://login.microsoftonline.com/${TENANT_ID}/v2.0 \
--email-claims email \
--name-claims name \
--username-claims preferred_username \
--extra-scopes email,profile \
--groups-claims groups
```

- 如果没有启用组声明，请运行以下命令：

```
$ rosa create idp \
--cluster ${CLUSTER_NAME} \
--type openid \
--name ${IDP_NAME} \
--client-id ${APP_ID} \
--client-secret ${CLIENT_SECRET} \
--issuer-url https://login.microsoftonline.com/${TENANT_ID}/v2.0 \
--email-claims email \
--name-claims name \
--username-claims preferred_username \
--extra-scopes email,profile
```

几分钟后，集群身份验证 Operator 会协调您的更改，您可以使用 Entra ID 登录到集群。

8.5. 为各个用户和组授予额外权限

第一次登录时，您可能会注意到您具有非常有限的权限。默认情况下，Red Hat OpenShift Service on AWS 仅授予您在集群中创建新项目或命名空间的权限。其他项目在视图中受到限制。

您必须向各个用户和组授予这些额外功能。

为单个用户授予额外权限

Red Hat OpenShift Service on AWS 包括大量预配置的角色，包括允许对集群的完整访问权限和控制的 **cluster-admin** 角色。

流程

- 运行以下命令，授予用户对 **cluster-admin** 角色的访问权限：

```
$ rosa grant user cluster-admin \
  --user=<USERNAME> 1
  --cluster=${CLUSTER_NAME}
```

1 提供您要具有集群管理员权限的 Entra ID 用户名。

为单个组授予额外权限

如果您选择启用组声明，集群 OAuth 供应商将使用组 ID 自动创建或更新用户的组成员资格。集群 OAuth 供应商不会自动为创建的组创建 **RoleBindings** 和 **ClusterRoleBindings**，而是负责使用您自己的进程创建这些绑定。

要授予自动生成的对 **cluster-admin** 角色的组访问权限，您必须为组 ID 创建 **ClusterRoleBinding**。

流程

- 运行以下命令来创建 **ClusterRoleBinding** ：

```
$ oc create clusterrolebinding cluster-admin-group \
--clusterrole=cluster-admin \
--group=<GROUP_ID> 1
```

1 提供您要具有集群管理员权限的 Entra ID 组 ID。

现在，指定组中的任何用户都会自动接收 **cluster-admin** 访问权限。

8.6. 其他资源

有关如何使用 RBAC 在 AWS 中定义和应用权限的更多信息，请参阅 Red Hat OpenShift Service on AWS 文档。

第 9 章 教程： 在带有 STS 的 ROSA 上使用 AWS SECRETS MANAGER CSI

AWS Secret 和配置提供程序(ASCP)提供了一种将 AWS Secret 公开为 Kubernetes 存储卷的方法。使用 ASCP，您可以在 Secrets Manager 中存储和管理您的 secret，然后通过 Red Hat OpenShift Service on AWS (ROSA)上运行的工作负载来检索它们。

9.1. 前提条件

在开始此过程前，请确定您有以下资源和工具：

- 使用 STS 部署的 ROSA 集群
- Helm 3
- **AWS** CLI
- **oc** CLI
- **jq** CLI

额外的环境要求

1. 运行以下命令登录到您的 ROSA 集群：

```
$ oc login --token=<your-token> --server=<your-server-url>
```

   您可以通过从 Red Hat OpenShift Cluster Manager 访问 pull secret 来查找您的登录令牌。

2. 运行以下命令，验证集群是否有 STS：

```
$ oc get authentication.config.openshift.io cluster -o json \
  | jq .spec.serviceAccountIssuer
```

   输出示例

```
"https://xxxxx.cloudfront.net/xxxxx"
```

   如果您的输出不同，请不要继续。在继续此过程前，请参阅创建 STS 集群的红帽文档。

3. 运行以下命令，将 **SecurityContextConstraints** 权限设置为允许 CSI 驱动程序运行：

```
$ oc new-project csi-secrets-store
$ oc adm policy add-scc-to-user privileged \
    system:serviceaccount:csi-secrets-store:secrets-store-csi-driver
$ oc adm policy add-scc-to-user privileged \
    system:serviceaccount:csi-secrets-store:csi-secrets-store-provider-aws
```

4. 运行以下命令，创建环境变量以便稍后使用：

```
$ export REGION=$(oc get infrastructure cluster -o=jsonpath="
{.status.platformStatus.aws.region}")
$ export OIDC_ENDPOINT=$(oc get authentication.config.openshift.io cluster \
```

```
  -o jsonpath='{.spec.serviceAccountIssuer}' | sed  's|^https://||')
$ export AWS_ACCOUNT_ID=`aws sts get-caller-identity --query Account --output text`
$ export AWS_PAGER=""
```

9.2. 部署 AWS SECRET 和配置提供程序

1. 运行以下命令，使用 Helm 注册 secret 存储 CSI 驱动程序：

```
$ helm repo add secrets-store-csi-driver \
  https://kubernetes-sigs.github.io/secrets-store-csi-driver/charts
```

2. 运行以下命令来更新 Helm 仓库：

```
$ helm repo update
```

3. 运行以下命令来安装 secret 存储 CSI 驱动程序：

```
$ helm upgrade --install -n csi-secrets-store \
  csi-secrets-store-driver secrets-store-csi-driver/secrets-store-csi-driver
```

4. 运行以下命令来部署 AWS 供应商：

```
$ oc -n csi-secrets-store apply -f \
  https://raw.githubusercontent.com/rh-mobb/documentation/main/content/misc/secrets-store-csi/aws-provider-installer.yaml
```

5. 运行以下命令，检查两个 Daemonsets 是否正在运行：

```
$ oc -n csi-secrets-store get ds \
  csi-secrets-store-provider-aws \
  csi-secrets-store-driver-secrets-store-csi-driver
```

6. 运行以下命令，标记 Secrets Store CSI Driver 以允许与受限 pod 安全配置集一起使用：

```
$ oc label csidriver.storage.k8s.io/secrets-store.csi.k8s.io security.openshift.io/csi-ephemeral-volume-profile=restricted
```

9.3. 创建 SECRET 和 IAM 访问策略

1. 运行以下命令，在 Secret Manager 中创建 secret：

```
$ SECRET_ARN=$(aws --region "$REGION" secretsmanager create-secret \
  --name MySecret --secret-string \
  '{"username":"shadowman", "password":"hunter2"}' \
  --query ARN --output text)
$ echo $SECRET_ARN
```

2. 运行以下命令来创建 IAM 访问策略文档：

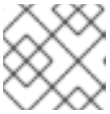
```
$ cat << EOF > policy.json
{
```

```
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "secretsmanager:GetSecretValue",
      "secretsmanager:DescribeSecret"
    ],
    "Resource": ["$SECRET_ARN"]
    }]
}
EOF
```

3. 运行以下命令来创建 IAM 访问策略：

```
$ POLICY_ARN=$(aws --region "$REGION" --query Policy.Arn \
--output text iam create-policy \
--policy-name openshift-access-to-mysecret-policy \
--policy-document file://policy.json)
$ echo $POLICY_ARN
```

4. 运行以下命令来创建 IAM 角色信任策略文档：



注意

信任策略被锁定到您稍后在此过程中创建的命名空间的默认服务帐户。

```
$ cat <<EOF > trust-policy.json
{
  "Version": "2012-10-17",
  "Statement": [
  {
  "Effect": "Allow",
  "Condition": {
    "StringEquals" : {
      "${OIDC_ENDPOINT}:sub": ["system:serviceaccount:my-application:default"]
    }
   },
   "Principal": {
      "Federated": "arn:aws:iam::$AWS_ACCOUNT_ID:oidc-provider/${OIDC_ENDPOINT}"
   },
   "Action": "sts:AssumeRoleWithWebIdentity"
   }
   ]
}
EOF
```

5. 运行以下命令来创建 IAM 角色：

```
$ ROLE_ARN=$(aws iam create-role --role-name openshift-access-to-mysecret \
--assume-role-policy-document file://trust-policy.json \
--query Role.Arn --output text)
$ echo $ROLE_ARN
```

6. 运行以下命令，将角色附加到策略：

```
$ aws iam attach-role-policy --role-name openshift-access-to-mysecret \
   --policy-arn $POLICY_ARN
```

9.4. 创建应用程序以使用此 SECRET

1. 运行以下命令来创建 OpenShift 项目：

```
$ oc new-project my-application
```

2. 运行以下命令，注解 default 服务帐户以使用 STS 角色：

```
$ oc annotate -n my-application serviceaccount default \
   eks.amazonaws.com/role-arn=$ROLE_ARN
```

3. 运行以下命令，创建 secret 供应商类以访问我们的 secret：

```
$ cat << EOF | oc apply -f -
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
  name: my-application-aws-secrets
spec:
  provider: aws
  parameters:
    objects: |
      - objectName: "MySecret"
        objectType: "secretsmanager"
EOF
```

4. 使用以下命令，使用我们的 secret 创建 Deployment：

```
$ cat << EOF | oc apply -f -
apiVersion: v1
kind: Pod
metadata:
  name: my-application
  labels:
    app: my-application
spec:
  volumes:
  - name: secrets-store-inline
    csi:
      driver: secrets-store.csi.k8s.io
      readOnly: true
      volumeAttributes:
        secretProviderClass: "my-application-aws-secrets"
  containers:
  - name: my-application-deployment
    image: k8s.gcr.io/e2e-test-images/busybox:1.29
    command:
      - "/bin/sleep"
      - "10000"
    volumeMounts:
```

```
    - name: secrets-store-inline
      mountPath: "/mnt/secrets-store"
      readOnly: true
EOF
```

5. 运行以下命令，验证 Pod 是否已挂载了 secret：

```
$ oc exec -it my-application -- cat /mnt/secrets-store/MySecret
```

9.5. 清理

1. 运行以下命令来删除应用程序：

```
$ oc delete project my-application
```

2. 运行以下命令来删除 secret 存储 csi 驱动程序：

```
$ helm delete -n csi-secrets-store csi-secrets-store-driver
```

3. 运行以下命令来删除安全性上下文约束：

```
$ oc adm policy remove-scc-from-user privileged \
    system:serviceaccount:csi-secrets-store:secrets-store-csi-driver
$ oc adm policy remove-scc-from-user privileged \
    system:serviceaccount:csi-secrets-store:csi-secrets-store-provider-aws
```

4. 运行以下命令来删除 AWS 供应商：

```
$ oc -n csi-secrets-store delete -f \
https://raw.githubusercontent.com/rh-mobb/documentation/main/content/misc/secrets-store-csi/aws-provider-installer.yaml
```

5. 运行以下命令来删除 AWS 角色和策略：

```
$ aws iam detach-role-policy --role-name openshift-access-to-mysecret \
    --policy-arn $POLICY_ARN
$ aws iam delete-role --role-name openshift-access-to-mysecret
$ aws iam delete-policy --policy-arn $POLICY_ARN
```

6. 运行以下命令来删除 Secrets Manager secret：

```
$ aws secretsmanager --region $REGION delete-secret --secret-id $SECRET_ARN
```

第 10 章 教程：在 ROSA 上使用 AWS CONTROLLER FOR KUBERNETES

AWS Controller for Kubernetes (ACK)可让您直接从 Red Hat OpenShift Service on AWS (ROSA)定义和使用 AWS 服务资源。使用 ACK，您可以为应用程序利用 AWS 管理的服务，而无需定义集群外的资源，或运行提供集群中数据库或消息队列等支持功能的服务。

您可以直接从 OperatorHub 安装各种 ACK Operator。这样便可轻松开始，并将 Operator 用于您的应用程序。此控制器是 Kubernetes 项目的 AWS Controller 的一个组件，当前处于开发人员预览状态。

使用此教程部署 ACK S3 Operator。您还可以针对集群的 OperatorHub 中的任何其他 ACK Operator 进行调整。

10.1. 前提条件

- ROSA 集群
- 具有 **cluster-admin** 特权的用户帐户
- OpenShift CLI (**oc**)
- Amazon Web Services (AWS) CLI (**aws**)

10.2. 设置您的环境

1. 配置以下环境变量，将集群名称更改为适合您的集群：

```
$ export CLUSTER_NAME=$(oc get infrastructure cluster -o=jsonpath="
{.status.infrastructureName}"  | sed 's/-[a-z0-9]\{5\}$//')
$ export REGION=$(rosa describe cluster -c ${ROSA_CLUSTER_NAME} --output json | jq -r
.region.id)
$ export OIDC_ENDPOINT=$(oc get authentication.config.openshift.io cluster -o json | jq -r
.spec.serviceAccountIssuer | sed  's|^https://||')
$ export AWS_ACCOUNT_ID=`aws sts get-caller-identity --query Account --output text`
$ export ACK_SERVICE=s3
$ export ACK_SERVICE_ACCOUNT=ack-${ACK_SERVICE}-controller
$ export POLICY_ARN=arn:aws:iam::aws:policy/AmazonS3FullAccess
$ export AWS_PAGER=""
$ export SCRATCH="/tmp/${ROSA_CLUSTER_NAME}/ack"
$ mkdir -p ${SCRATCH}
```

2. 在移动到下一部分前，请确定所有字段都正确输出：

```
$ echo "Cluster: ${ROSA_CLUSTER_NAME}, Region: ${REGION}, OIDC Endpoint:
${OIDC_ENDPOINT}, AWS Account ID: ${AWS_ACCOUNT_ID}"
```

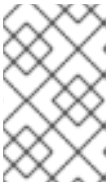
10.3. 准备 AWS 帐户

1. 为 ACK Operator 创建 AWS Identity Access Management (IAM)信任策略：

```
$ cat <<EOF > "${SCRATCH}/trust-policy.json"
{
 "Version": "2012-10-17",
```

```
 "Statement": [
 {
 "Effect": "Allow",
 "Condition": {
   "StringEquals" : {
     "${OIDC_ENDPOINT}:sub": "system:serviceaccount:ack-
system:${ACK_SERVICE_ACCOUNT}"
   }
 },
 "Principal": {
   "Federated": "arn:aws:iam::$AWS_ACCOUNT_ID:oidc-provider/${OIDC_ENDPOINT}"
 },
 "Action": "sts:AssumeRoleWithWebIdentity"
 }
 ]
}
EOF
```

2. 为 ACK Operator 创建 AWS IAM 角色，以假设附加 **AmazonS3FullAccess** 策略：



注意

您可以在每个项目的 GitHub 仓库中找到推荐的策略，例如 https://github.com/aws-controllers-k8s/s3-controller/blob/main/config/iam/recommended-policy-arn。

```
$ ROLE_ARN=$(aws iam create-role --role-name "ack-${ACK_SERVICE}-controller" \
   --assume-role-policy-document "file://${SCRATCH}/trust-policy.json" \
   --query Role.Arn --output text)
$ echo $ROLE_ARN

$ aws iam attach-role-policy --role-name "ack-${ACK_SERVICE}-controller" \
    --policy-arn ${POLICY_ARN}
```

10.4. 安装 ACK S3 CONTROLLER

1. 创建一个项目，将 ACK S3 Operator 安装到：

```
$ oc new-project ack-system
```

2. 使用 ACK S3 Operator 配置创建文件：

注意

ACK_WATCH_NAMESPACE 被视为空白，以便控制器可以正确地监视集群中的所有命名空间。

```
$ cat <<EOF > "${SCRATCH}/config.txt"
ACK_ENABLE_DEVELOPMENT_LOGGING=true
ACK_LOG_LEVEL=debug
ACK_WATCH_NAMESPACE=
AWS_REGION=${REGION}
```

```
AWS_ENDPOINT_URL=
ACK_RESOURCE_TAGS=${CLUSTER_NAME}
ENABLE_LEADER_ELECTION=true
LEADER_ELECTION_NAMESPACE=
EOF
```

3. 使用上一步中的文件来创建 ConfigMap：

```
$ oc -n ack-system create configmap \
  --from-env-file=${SCRATCH}/config.txt ack-${ACK_SERVICE}-user-config
```

4. 从 OperatorHub 安装 ACK S3 Operator：

```
$ cat << EOF | oc apply -f -
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: ack-${ACK_SERVICE}-controller
  namespace: ack-system
spec:
  upgradeStrategy: Default
---
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: ack-${ACK_SERVICE}-controller
  namespace: ack-system
spec:
  channel: alpha
  installPlanApproval: Automatic
  name: ack-${ACK_SERVICE}-controller
  source: community-operators
  sourceNamespace: openshift-marketplace
EOF
```

5. 使用 AWS IAM 角色注解 ACK S3 Operator 服务帐户，以假定和重启部署：

```
$ oc -n ack-system annotate serviceaccount ${ACK_SERVICE_ACCOUNT} \
  eks.amazonaws.com/role-arn=${ROLE_ARN} && \
  oc -n ack-system rollout restart deployment ack-${ACK_SERVICE}-controller
```

6. 验证 ACK S3 Operator 是否正在运行：

```
$ oc -n ack-system get pods
```

输出示例

```
NAME                                READY   STATUS    RESTARTS   AGE
ack-s3-controller-585f6775db-s4lfz  1/1     Running   0          51s
```

10.5. 验证部署

1. 部署 S3 存储桶资源：

```
$ cat << EOF | oc apply -f -
apiVersion: s3.services.k8s.aws/v1alpha1
kind: Bucket
metadata:
  name: ${CLUSTER-NAME}-bucket
  namespace: ack-system
spec:
  name: ${CLUSTER-NAME}-bucket
EOF
```

2. 验证 AWS 中 S3 存储桶是否已创建：

```
$ aws s3 ls | grep ${CLUSTER_NAME}-bucket
```

输出示例

```
2023-10-04 14:51:45 mrmc-test-maz-bucket
```

10.6. 清理

1. 删除 S3 存储桶资源：

```
$ oc -n ack-system delete bucket.s3.services.k8s.aws/${CLUSTER-NAME}-bucket
```

2. 删除 ACK S3 Operator 和 AWS IAM 角色：

```
$ oc -n ack-system delete subscription ack-${ACK_SERVICE}-controller
$ aws iam detach-role-policy \
  --role-name "ack-${ACK_SERVICE}-controller" \
  --policy-arn ${POLICY_ARN}
$ aws iam delete-role \
  --role-name "ack-${ACK_SERVICE}-controller"
```

3. 删除 **ack-system** 项目：

```
$ oc delete project ack-system
```

第 11 章 教程：在 ROSA 上部署外部 DNS OPERATOR

External DNS Operator 部署并管理 **ExternalDNS**，以便为来自外部 DNS 供应商（如 Amazon Route 53）的服务和路由提供名称解析到 Red Hat OpenShift Service on AWS (ROSA)集群。在本教程中，我们将使用二级入口控制器部署和配置外部 DNS Operator，以管理 Amazon Route 53 中的 DNS 记录。

重要

External DNS Operator 不支持使用 IAM 角色用于服务帐户(IRSA)的 STS，并使用长期的 Identity Access Management (IAM)凭证。当 Operator 支持 STS 时，将更新本教程。

11.1. 前提条件

- ROSA 经典集群

注意

目前不支持使用 HCP 的 ROSA。

- 具有 **cluster-admin** 特权的用户帐户
- OpenShift CLI (**oc**)
- Amazon Web Services (AWS) CLI (**aws**)
- 唯一的域，如 **apps.example.com**
- 以上域的 Amazon Route 53 公共托管区

11.2. 设置您的环境

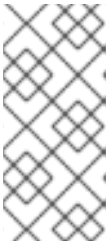
1. 配置以下环境变量：

```
$ export DOMAIN=<apps.example.com> 1
$ export AWS_PAGER=""
$ export CLUSTER=$(oc get infrastructure cluster -o=jsonpath="{.status.infrastructureName}"
| sed 's/-[a-z0-9]\{5\}$//')
$ export REGION=$(oc get infrastructure cluster -o=jsonpath="
{.status.platformStatus.aws.region}")
$ export AWS_ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
$ export SCRATCH="/tmp/${CLUSTER}/external-dns"
$ mkdir -p ${SCRATCH}
```

1 使用您要用于 **IngressController** 的自定义域替换。

2. 在移动到下一部分前，请确定所有字段都正确输出：

```
$ echo "Cluster: ${CLUSTER}, Region: ${REGION}, AWS Account ID:
${AWS_ACCOUNT_ID}"
```



注意

上一命令中的"Cluster"输出可以是集群名称、集群的内部 ID 或集群的域前缀。如果要使用另一个标识符，您可以通过运行以下命令来手动设置这个值：

```
$ export CLUSTER=my-custom-value
```

11.3. 二级入口控制器设置

使用以下步骤使用自定义域部署二级入口控制器。

前提条件

- 唯一的域，如 **apps.example.com**
- 配置了上述自定义域的通配符或 SAN TLS 证书(**CN swig.apps.example.com**)

流程

1. 从私钥和公共证书创建一个新的 TLS secret，其中 **fullchain.pem** 是您的完整的通配符证书链（包括任何中间）和 **privkey.pem** 是您的通配符证书：

```
$ oc -n openshift-ingress create secret tls external-dns-tls --cert=fullchain.pem --
key=privkey.pem
```

2. 创建新的 **IngressController** 资源：

```
$ cat << EOF | oc apply -f -
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  name: external-dns-ingress
  namespace: openshift-ingress-operator
spec:
  domain: ${DOMAIN}
  defaultCertificate:
    name: external-dns-tls
  endpointPublishingStrategy:
    loadBalancer:
      dnsManagementPolicy: Unmanaged
      providerParameters:
        aws:
          type: NLB
        type: AWS
      scope: External
    type: LoadBalancerService
EOF
```

警告

此 **IngressController** 示例将在 AWS 帐户中创建可访问互联网的 Network Load Balancer (NLB)。要置备内部 NLB，请在创建 **IngressController** 资源前将 **.spec.endpointPublishingStrategy.loadBalancer.scope** 参数设置为 **Internal**。

3. 验证自定义域 IngressController 是否已成功创建了外部负载均衡器：

```
$ oc -n openshift-ingress get service/router-external-dns-ingress
```

输出示例

```
NAME                        TYPE           CLUSTER-IP      EXTERNAL-IP
PORT(S)                     AGE
router-external-dns-ingress   LoadBalancer   172.30.71.250
a4838bb991c6748439134ab89f132a43-aeae124077b50c01.elb.us-east-1.amazonaws.com
80:32227/TCP,443:30310/TCP    43s
```

11.4. 准备 AWS 帐户

1. 检索 Amazon Route 53 公共托管区 ID：

```
$ export ZONE_ID=$(aws route53 list-hosted-zones-by-name --output json \
  --dns-name "${DOMAIN}." --query 'HostedZones[0]'.Id --out text | sed 's/\/hostedzone\///')
```

2. 准备包含所需 DNS 更改的文档，以便为 Ingress Controller 的规范域启用 DNS 解析：

```
$ NLB_HOST=$(oc -n openshift-ingress get service/router-external-dns-ingress -ojsonpath="
{.status.loadBalancer.ingress[0].hostname}")
$ cat << EOF > "${SCRATCH}/create-cname.json"
{
  "Comment":"Add CNAME to ingress controller canonical domain",
  "Changes":[{
      "Action":"CREATE",
      "ResourceRecordSet":{
        "Name": "router-external-dns-ingress.${DOMAIN}",
      "Type":"CNAME",
      "TTL":30,
      "ResourceRecords":[{
        "Value": "${NLB_HOST}"
      }]
    }
  }]
}
EOF
```

External DNS Operator 使用这个规范域作为 CNAME 记录的目标。

3. 提交您对 Amazon Route 53 的更改以进行传播：

```
aws route53 change-resource-record-sets \
  --hosted-zone-id ${ZONE_ID} \
  --change-batch file://${SCRATCH}/create-cname.json
```

4. 创建一个 AWS IAM 策略文档，允许 **外部 DNS** Operator 仅更新自定义域公共托管区：

```
$ cat << EOF > "${SCRATCH}/external-dns-policy.json"
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "route53:ChangeResourceRecordSets"
      ],
      "Resource": [
        "arn:aws:route53:::hostedzone/${ZONE_ID}"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "route53:ListHostedZones",
        "route53:ListResourceRecordSets"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
EOF
```

5. 创建 AWS IAM 用户：

```
$ aws iam create-user --user-name "${CLUSTER}-external-dns-operator"
```

6. 附加策略：

```
$ aws iam attach-user-policy --user-name "${CLUSTER}-external-dns-operator" --policy-arn
$POLICY_ARN
```

注意

这将在以后的版本中使用 IRSA 改为 STS。

7. 为 IAM 用户创建 AWS 密钥：

```
$ SECRET_ACCESS_KEY=$(aws iam create-access-key --user-name "${CLUSTER}-
external-dns-operator")
```

8. 创建静态凭证：

```
$ cat << EOF > "${SCRATCH}/credentials"
[default]
aws_access_key_id = $(echo $SECRET_ACCESS_KEY | jq -r '.AccessKey.AccessKeyId')
aws_secret_access_key = $(echo $SECRET_ACCESS_KEY | jq -r
'.AccessKey.SecretAccessKey')
EOF
```

11.5. 安装 EXTERNAL DNS OPERATOR

1. 创建一个新项目

```
$ oc new-project external-dns-operator
```

2. 从 OperatorHub 安装 **外部 DNS** Operator：

```
$ cat << EOF | oc apply -f -
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: external-dns-group
  namespace: external-dns-operator
spec:
  targetNamespaces:
  - external-dns-operator
---
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: external-dns-operator
  namespace: external-dns-operator
spec:
  channel: stable-v1.1
  installPlanApproval: Automatic
  name: external-dns-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
EOF
```

3. 等待 **外部 DNS** Operator 正在运行：

```
$ oc rollout status deploy external-dns-operator --timeout=300s
```

4. 从 AWS IAM 用户凭证创建 secret：

```
$ oc -n external-dns-operator create secret generic external-dns \
  --from-file "${SCRATCH}/credentials"
```

5. 部署 **ExternalDNS** 控制器：

```
$ cat << EOF | oc apply -f -
apiVersion: externaldns.olm.openshift.io/v1beta1
```

```
kind: ExternalDNS
metadata:
  name: ${DOMAIN}
spec:
  domains:
    - filterType: Include
      matchType: Exact
      name: ${DOMAIN}
  provider:
    aws:
      credentials:
        name: external-dns
    type: AWS
  source:
    openshiftRouteOptions:
      routerName: external-dns-ingress
    type: OpenShiftRoute
  zones:
    - ${ZONE_ID}
EOF
```

6. 等待控制器运行：

```
$ oc rollout status deploy external-dns-${DOMAIN} --timeout=300s
```

11.6. 部署示例应用程序

现在 **ExternalDNS** 控制器正在运行，您可以部署一个示例应用程序，以确认在公开新路由时配置了自定义域并信任。

1. 为您的示例应用程序创建一个新项目：

```
$ oc new-project hello-world
```

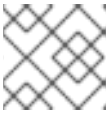
2. 部署 hello world 应用：

```
$ oc new-app -n hello-world --image=docker.io/openshift/hello-openshift
```

3. 为应用程序创建指定自定义域名的路由：

```
$ oc -n hello-world create route edge --service=hello-openshift hello-openshift-tls \
--hostname hello-openshift.${DOMAIN}
```

4. 检查 ExternalDNS 会自动创建 DNS 记录：



注意

记录可能需要几分钟时间才会出现在 Amazon Route 53 中。

```
$ aws route53 list-resource-record-sets --hosted-zone-id ${ZONE_ID} \
  --query "ResourceRecordSets[?Type == 'CNAME']" | grep hello-openshift
```

5. 可选： 您还可以查看 ExternalDNS 创建的 TXT 记录：

```
$ aws route53 list-resource-record-sets --hosted-zone-id ${ZONE_ID} \
  --query "ResourceRecordSets[?Type == 'TXT']" | grep ${DOMAIN}
```

6. curl 新创建的 DNS 记录到示例应用程序，以验证可以访问 hello world 应用程序：

```
$ curl https://hello-openshift.${DOMAIN}
```

输出示例

```
Hello OpenShift!
```

第 12 章 教程：在 ROSA 上使用 CERT-MANAGER OPERATOR 动态发布证书

虽然通配符证书通过保护给定域的所有第一级子域和单个证书提供简单性，但其他用例可能需要为每个域使用单个证书。

了解如何使用 cert-manager Operator for Red Hat OpenShift，Let's Encrypt 为使用自定义域创建的路由动态发布证书。

12.1. 前提条件

- ROSA 集群(HCP 或 Classic)
- 具有 **cluster-admin** 特权的用户帐户
- OpenShift CLI (**oc**)
- Amazon Web Services (AWS) CLI (**aws**)
- 唯一的域，如 3.0. **apps.example.com**
- 以上域的 Amazon Route 53 公共托管区

12.2. 设置您的环境

1. 配置以下环境变量：

```
$ export DOMAIN=apps.example.com 1
$ export EMAIL=email@example.com 2
$ export AWS_PAGER=""
$ export CLUSTER=$(oc get infrastructure cluster -o=jsonpath="{.status.infrastructureName}"
| sed 's/-[a-z0-9]\{5\}$//')
$ export OIDC_ENDPOINT=$(oc get authentication.config.openshift.io cluster -o json | jq -r
.spec.serviceAccountIssuer | sed  's|^https://||')
$ export REGION=$(oc get infrastructure cluster -o=jsonpath="
{.status.platformStatus.aws.region}")
$ export AWS_ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
$ export SCRATCH="/tmp/${CLUSTER}/dynamic-certs"
$ mkdir -p ${SCRATCH}
```

1. 使用您要用于 **IngressController** 的自定义域替换。
2. 使用您希望 Let 的 Encrypt 用于向您的证书发送通知的电子邮件替换。

2. 在移动到下一部分前，请确定所有字段都正确输出：

```
$ echo "Cluster: ${CLUSTER}, Region: ${REGION}, OIDC Endpoint: ${OIDC_ENDPOINT},
AWS Account ID: ${AWS_ACCOUNT_ID}"
```

注意

上一命令中的"Cluster"输出可以是集群名称、集群的内部 ID 或集群的域前缀。如果要使用另一个标识符，您可以通过运行以下命令来手动设置这个值：

```
$ export CLUSTER=my-custom-value
```

12.3. 准备 AWS 帐户

当 cert-manager 从 Let's Encrypt （或其他 ACME 证书签发者）请求证书时，Let 的 Encrypt 服务器会验证您是否使用 *质询* 控制该证书中的域名。在本教程中，您使用 DNS-01 质询，它证明您通过在域名下将特定值放在 TXT 记录中来控制您的域名的 DNS。这一切都由 cert-manager 自动完成。要允许 cert-manager 权限为您的域修改 Amazon Route 53 公共托管区，您需要创建一个具有特定策略权限的 Identity Access Management (IAM)角色，以及允许访问 pod 的信任关系。

本教程中使用的公共托管区与 ROSA 集群位于同一个 AWS 帐户。如果您的公共托管区位于不同的帐户中，则需要执行一些额外的步骤来 跨 帐户访问。

1. 检索 Amazon Route 53 公共托管区 ID：

注意

此命令查找与之前作为 **DOMAIN** 环境变量指定的自定义域匹配的公共托管区。您可以通过运行 **export ZONE_ID=<zone_ID** > 来手动指定 Amazon Route 53 公共托管区，将 &lt **;zone_ID&** gt; 替换为您的特定的 Amazon Route 53 公共托管区 ID。

```
$ export ZONE_ID=$(aws route53 list-hosted-zones-by-name --output json \
  --dns-name "${DOMAIN}." --query 'HostedZones[0]'.Id --out text | sed 's/\/hostedzone\///')
```

2. 为 cert-manager Operator 创建 AWS IAM 策略文档，它提供 *只* 更新指定的公共托管区的功能：

```
$ cat <<EOF > "${SCRATCH}/cert-manager-policy.json"
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "route53:GetChange",
      "Resource": "arn:aws:route53:::change/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "route53:ChangeResourceRecordSets",
        "route53:ListResourceRecordSets"
      ],
      "Resource": "arn:aws:route53:::hostedzone/${ZONE_ID}"
    },
    {
      "Effect": "Allow",
      "Action": "route53:ListHostedZonesByName",
      "Resource": "*"
    }
```

```
  ]
}
EOF
```

3. 使用您在上一步中创建的文件创建 IAM 策略：

```
$ POLICY_ARN=$(aws iam create-policy --policy-name "${CLUSTER}-cert-manager-policy" \
  --policy-document file://${SCRATCH}/cert-manager-policy.json \
  --query 'Policy.Arn' --output text)
```

4. 为 cert-manager Operator 创建 AWS IAM 信任策略：

```
$ cat <<EOF > "${SCRATCH}/trust-policy.json"
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Condition": {
   "StringEquals" : {
     "${OIDC_ENDPOINT}:sub": "system:serviceaccount:cert-manager:cert-manager"
   }
 },
 "Principal": {
   "Federated": "arn:aws:iam::$AWS_ACCOUNT_ID:oidc-provider/${OIDC_ENDPOINT}"
 },
 "Action": "sts:AssumeRoleWithWebIdentity"
 }
 ]
}
EOF
```

5. 使用您在上一步中创建的信任策略，为 cert-manager Operator 创建 IAM 角色：

```
$ ROLE_ARN=$(aws iam create-role --role-name "${CLUSTER}-cert-manager-operator" \
   --assume-role-policy-document "file://${SCRATCH}/trust-policy.json" \
   --query Role.Arn --output text)
```

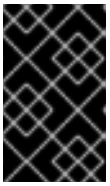
6. 将权限策略附加到角色：

```
$ aws iam attach-role-policy --role-name "${CLUSTER}-cert-manager-operator" \
  --policy-arn ${POLICY_ARN}
```

12.4. 安装 CERT-MANAGER OPERATOR

1. 创建一个项目，将 cert-manager Operator 安装到其中：

```
$ oc new-project cert-manager-operator
```



重要

不要试图在集群中使用多个 cert-manager Operator。如果在集群中安装了社区 cert-manager Operator，则必须在为 Red Hat OpenShift 安装 cert-manager Operator 前卸载它。

2. 为 Red Hat OpenShift 安装 cert-manager Operator：

```
$ cat << EOF | oc apply -f -
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: openshift-cert-manager-operator-group
  namespace: cert-manager-operator
spec:
  targetNamespaces:
  - cert-manager-operator
---
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: openshift-cert-manager-operator
  namespace: cert-manager-operator
spec:
  channel: stable-v1
  installPlanApproval: Automatic
  name: openshift-cert-manager-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
EOF
```

注意

此 Operator 需要几分钟时间来安装并完成它的设置。

3. 验证 cert-manager Operator 是否正在运行：

```
$ oc -n cert-manager-operator get pods
```

输出示例

```
NAME                                                READY   STATUS    RESTARTS   AGE
cert-manager-operator-controller-manager-84b8799db5-gv8mx   2/2     Running   0          12s
```

4. 使用之前创建的 AWS IAM 角色注解 cert-manager pod 使用的服务帐户：

```
$ oc -n cert-manager annotate serviceaccount cert-manager eks.amazonaws.com/role-arn=${ROLE_ARN}
```

5. 运行以下命令重启现有的 cert-manager 控制器 pod：

```
$ oc -n cert-manager delete pods -l app.kubernetes.io/name=cert-manager
```

6. 修补 Operator 配置以使用外部名称服务器以防止 DNS-01 质询问题：

```
$ oc patch certmanager.operator.openshift.io/cluster --type merge \
  -p '{"spec":{"controllerConfig":{"overrideArgs":["--dns01-recursive-nameservers-only","--
dns01-recursive-nameservers=1.1.1.1:53"]}}}'
```

7. 运行以下命令，创建一个 **ClusterIssuer** 资源以使用 Let 的 Encrypt ：

```
$ cat << EOF | oc apply -f -
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
  name: letsencrypt-production
spec:
  acme:
    server: https://acme-v02.api.letsencrypt.org/directory
    email: ${EMAIL}
    # This key doesn't exist, cert-manager creates it
    privateKeySecretRef:
      name: prod-letsencrypt-issuer-account-key
    solvers:
    - dns01:
        route53:
          hostedZoneID: ${ZONE_ID}
          region: ${REGION}
          secretAccessKeySecretRef:
            name: ''
EOF
```

8. 验证 **ClusterIssuer** 资源是否已就绪：

```
$ oc get clusterissuer.cert-manager.io/letsencrypt-production
```

输出示例

```
NAME                  READY   AGE
letsencrypt-production   True    47s
```

12.5. 创建自定义域 INGRESS CONTROLLER

1. 创建并配置证书资源来为自定义域 Ingress Controller 置备证书：

注意

以下示例使用单个域证书。还支持 SAN 和通配符证书。

```
$ cat << EOF | oc apply -f -
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: custom-domain-ingress-cert
  namespace: openshift-ingress
```

```
spec:
  secretName: custom-domain-ingress-cert-tls
  issuerRef:
    name: letsencrypt-production
    kind: ClusterIssuer
  commonName: "${DOMAIN}"
  dnsNames:
  - "${DOMAIN}"
EOF
```

2. 验证证书是否已发布：

注意

Let's Encrypt 发布此证书需要几分钟时间。如果用时超过 5 分钟，请运行 **oc -n openshift-ingress describe certificate.cert-manager.io/custom-domain-ingress-cert** 来查看 cert-manager 报告的问题。

```
$ oc -n openshift-ingress get certificate.cert-manager.io/custom-domain-ingress-cert
```

输出示例

```
NAME                          READY   SECRET                        AGE
custom-domain-ingress-cert    True    custom-domain-ingress-cert-tls   9m53s
```

3. 创建新的 **IngressController** 资源：

```
$ cat << EOF | oc apply -f -
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  name: custom-domain-ingress
  namespace: openshift-ingress-operator
spec:
  domain: ${DOMAIN}
  defaultCertificate:
    name: custom-domain-ingress-cert-tls
  endpointPublishingStrategy:
    loadBalancer:
      dnsManagementPolicy: Unmanaged
      providerParameters:
        aws:
          type: NLB
        type: AWS
      scope: External
    type: LoadBalancerService
EOF
```

警告

此 **IngressController** 示例将在 AWS 帐户中创建可访问互联网的 Network Load Balancer (NLB)。要置备内部 NLB，请在创建 **IngressController** 资源前将 **.spec.endpointPublishingStrategy.loadBalancer.scope** 参数设置为 **Internal**。

4. 验证自定义域 IngressController 是否已成功创建了外部负载均衡器：

```
$ oc -n openshift-ingress get service/router-custom-domain-ingress
```

输出示例

```
NAME                         TYPE           CLUSTER-IP      EXTERNAL-IP
PORT(S)                      AGE
router-custom-domain-ingress   LoadBalancer   172.30.174.34
a309962c3bd6e42c08cadb9202eca683-1f5bbb64a1f1ec65.elb.us-east-1.amazonaws.com
80:31342/TCP,443:31821/TCP   7m28s
```

5. 准备带有所需 DNS 更改的文档，为您的自定义域 Ingress Controller 启用 DNS 解析：

```
$ INGRESS=$(oc -n openshift-ingress get service/router-custom-domain-ingress -
ojsonpath="{.status.loadBalancer.ingress[0].hostname}")
$ cat << EOF > "${SCRATCH}/create-cname.json"
{
  "Comment":"Add CNAME to custom domain endpoint",
  "Changes":[{
      "Action":"CREATE",
      "ResourceRecordSet":{
        "Name": "*.${DOMAIN}",
      "Type":"CNAME",
      "TTL":30,
      "ResourceRecords":[{
        "Value": "${INGRESS}"
      }]
    }
  }]
}
EOF
```

6. 提交您对 Amazon Route 53 的更改以进行传播：

```
$ aws route53 change-resource-record-sets \
  --hosted-zone-id ${ZONE_ID} \
  --change-batch file://${SCRATCH}/create-cname.json
```

注意

虽然通配符 CNAME 记录避免需要为每个使用自定义域 Ingress Controller 部署的新应用程序创建新记录，但这些应用程序使用的证书 **都不是** 通配符证书。

12.6. 为自定义域路由配置动态证书

现在，您可以在指定域的任何第一级子域上公开集群应用程序，但连接不会与应用程序域匹配的 TLS 证书进行保护。为确保这些集群应用程序为每个域名都有有效的证书，请将 cert-manager 配置为动态向域下创建的每个新路由发布证书。

1. 创建必要的 OpenShift 资源 cert-manager 需要管理 OpenShift 路由的证书。
   此步骤会创建一个新的部署（以及 pod），该部署专门监控集群中注解的路由。如果在新路由中找到 **issuer-kind** 和 **issuer-name** 注解，则会为这个路由唯一的新证书请求 Issuer （本例中为 ClusterIssuer），这将遵循创建路由时指定的主机名。

   注意

   如果集群无法访问 GitHub，您可以在本地保存原始内容，并运行 **oc apply -f localfilename.yaml -n cert-manager**。

   ```
   $ oc -n cert-manager apply -f https://github.com/cert-manager/openshift-routes/releases/latest/download/cert-manager-openshift-routes.yaml
   ```

   此步骤中也会创建以下额外 OpenShift 资源：

   - **ClusterRole** - 授予在集群中监控和更新路由的权限
   - **ServiceAccount** - 使用权限运行新创建的 pod
   - **ClusterRoleBinding** - 绑定这两个资源

2. 确保新的 **cert-manager-openshift-routes** pod 成功运行：

   ```
   $ oc -n cert-manager get pods
   ```

   结果示例

   ```
   NAME                                     READY   STATUS    RESTARTS   AGE
   cert-manager-866d8f788c-9kspc            1/1     Running   0          4h21m
   cert-manager-cainjector-6885c585bd-znws8 1/1     Running   0          4h41m
   cert-manager-openshift-routes-75b6bb44cd-f8kd5   1/1     Running   0          6s
   cert-manager-webhook-8498785dd9-bvfdf    1/1     Running   0          4h41m
   ```

12.7. 部署示例应用程序

现在，配置了动态证书，您可以部署一个示例应用程序，以确认在公开新路由时置备并信任证书。

1. 为您的示例应用程序创建一个新项目：

   ```
   $ oc new-project hello-world
   ```

2. 部署 hello world 应用：

```
$ oc -n hello-world new-app --image=docker.io/openshift/hello-openshift
```

3. 创建路由从集群外部公开应用程序：

```
$ oc -n hello-world create route edge --service=hello-openshift hello-openshift-tls --hostname
hello.${DOMAIN}
```

4. 验证路由的证书是否不被信任：

```
$ curl -I https://hello.${DOMAIN}
```

输出示例

```
curl: (60) SSL: no alternative certificate subject name matches target host name
'hello.example.com'
More details here: https://curl.se/docs/sslcerts.html

curl failed to verify the legitimacy of the server and therefore could not
establish a secure connection to it. To learn more about this situation and
how to fix it, please visit the web page mentioned above.
```

5. 注解路由以触发 cert-manager 为自定义域置备证书：

```
$ oc -n hello-world annotate route hello-openshift-tls cert-manager.io/issuer-
kind=ClusterIssuer cert-manager.io/issuer-name=letsencrypt-production
```

注意

创建证书需要 2-3 分钟。证书续订将由 cert-manager Operator 自动管理，因为它接近到期。

6. 验证路由的证书现在是否被信任：

```
$ curl -I https://hello.${DOMAIN}
```

输出示例

```
HTTP/2 200
date: Thu, 05 Oct 2023 23:45:33 GMT
content-length: 17
content-type: text/plain; charset=utf-8
set-cookie: 52e4465485b6fb4f8a1b1bed128d0f3b=68676068bb32d24f0f558f094ed8e4d7;
path=/; HttpOnly; Secure; SameSite=None
cache-control: private
```

12.8. 动态证书置备故障排除

注意

在创建证书时，验证过程通常需要 2-3 分钟才能完成。

如果在证书创建步骤中没有触发证书创建，请针对每个证书、certrequest、order 和 challenge 资源运行 oc describe，以查看有助于识别问题原因的事件或原因。

```
$ oc get certificate,certificaterequest,order,challenge
```

如需故障排除，您可以在 调试证书时参考此帮助指南。

您还可以将 cmctl CLI 工具用于各种证书管理活动，如检查证书的状态和测试续订。

第 13 章 教程：为外部流量分配一致的出口 IP

您可以为离开集群的流量分配一致的 IP 地址，如需要基于 IP 配置的安全组来满足安全标准。

默认情况下，Red Hat OpenShift Service on AWS (ROSA)使用 OVN-Kubernetes 容器网络接口 (CNI)从池中分配随机 IP 地址。这可使配置安全锁定无法预测或打开。

如需更多信息，请参阅配置出口 IP 地址。

目标

- 了解如何为出口流量配置一组可预测的 IP 地址。

前提条件

- 使用 OVN-Kubernetes 部署的 ROSA 集群
- OpenShift CLI (oc)
- ROSA CLI (rosa)
- jq

13.1. 设置环境变量

- 运行以下命令来设置环境变量：

注意

替换 ROSA_MACHINE_POOL_NAME 变量的值，以不同的机器池为目标。

```
$ export ROSA_CLUSTER_NAME=$(oc get infrastructure cluster -o=jsonpath="
{.status.infrastructureName}"  | sed 's/-[a-z0-9]\{5\}$//')
$ export ROSA_MACHINE_POOL_NAME=worker
```

13.2. 确保容量

每个公有云供应商都会限制分配给每个节点的 IP 地址数量。

- 运行以下命令验证足够容量：

```
$ oc get node -o json | \
    jq '.items[] |
        {
          "name": .metadata.name,
          "ips": (.status.addresses | map(select(.type == "InternalIP") | .address)),
          "capacity": (.metadata.annotations."cloud.network.openshift.io/egress-
ipconfig" | fromjson[] | .capacity.ipv4)
        }'
```

输出示例

```
---
{
  "name": "ip-10-10-145-88.ec2.internal",
  "ips": [
    "10.10.145.88"
  ],
  "capacity": 14
}
{
  "name": "ip-10-10-154-175.ec2.internal",
  "ips": [
    "10.10.154.175"
  ],
  "capacity": 14
}
---
```

13.3. 创建出口 IP 规则

1. 在创建出口 IP 规则前，请确定您要使用的出口 IP。

注意

您选择的出口 IP 应该作为置备 worker 节点的子网的一部分存在。

2. 可选：保留您请求的出口 IP，以避免与 AWS Virtual Private Cloud (VPC) Dynamic Host Configuration Protocol (DHCP)服务冲突。

在 AWS 文档中为 CIDR 保留页面请求显式 IP 保留。

13.4. 将出口 IP 分配给命名空间

1. 运行以下命令来创建新项目：

```
$ oc new-project demo-egress-ns
```

2. 运行以下命令，为命名空间中的所有 pod 创建出口规则：

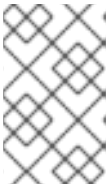
```
$ cat <<EOF | oc apply -f -
apiVersion: k8s.ovn.org/v1
kind: EgressIP
metadata:
  name: demo-egress-ns
spec:
  # NOTE: these egress IPs are within the subnet range(s) in which my worker nodes
  #       are deployed.
  egressIPs:
    - 10.10.100.253
    - 10.10.150.253
    - 10.10.200.253
  namespaceSelector:
    matchLabels:
      kubernetes.io/metadata.name: demo-egress-ns
EOF
```

13.5. 为 POD 分配出口 IP

1. 运行以下命令来创建新项目：

```
$ oc new-project demo-egress-pod
```

2. 运行以下命令，为 pod 创建出口规则：



注意

spec.namespaceSelector 是一个强制字段。

```
$ cat <<EOF | oc apply -f -
apiVersion: k8s.ovn.org/v1
kind: EgressIP
metadata:
  name: demo-egress-pod
spec:
  # NOTE: these egress IPs are within the subnet range(s) in which my worker nodes
  #       are deployed.
  egressIPs:
    - 10.10.100.254
    - 10.10.150.254
    - 10.10.200.254
  namespaceSelector:
    matchLabels:
      kubernetes.io/metadata.name: demo-egress-pod
  podSelector:
    matchLabels:
      run: demo-egress-pod
EOF
```

13.5.1. 标记节点

1. 运行以下命令来获取待处理的出口 IP 分配：

```
$ oc get egressips
```

输出示例

```
NAME              EGRESSIPS       ASSIGNED NODE   ASSIGNED EGRESSIPS
demo-egress-ns    10.10.100.253
demo-egress-pod   10.10.100.254
```

您创建的出口 IP 规则只适用于带有 k8s.ovn.org/egress-assignable 标签的节点。确保该标签只在特定的机器池中。

2. 使用以下命令为您的机器池分配标签：

警告

如果您依赖机器池的节点标签，这个命令会替换这些标签。务必在 --labels 字段中输入所需标签，以确保您的节点标签保留。

```
$ rosa update machinepool ${ROSA_MACHINE_POOL_NAME} \
  --cluster="${ROSA_CLUSTER_NAME}" \
  --labels "k8s.ovn.org/egress-assignable="
```

13.5.2. 查看出口 IP

- 运行以下命令，查看出口 IP 分配：

```
$ oc get egressips
```

输出示例

```
NAME              EGRESSIPS      ASSIGNED NODE                 ASSIGNED EGRESSIPS
demo-egress-ns    10.10.100.253  ip-10-10-156-122.ec2.internal  10.10.150.253
demo-egress-pod   10.10.100.254  ip-10-10-156-122.ec2.internal  10.10.150.254
```

13.6. 验证

13.6.1. 部署示例应用程序

要测试出口 IP 规则，请创建一个仅限于我们指定的出口 IP 地址的服务。这会模拟一个外部服务，该服务预期一小部分 IP 地址。

1. 运行 echoserver 命令以复制请求：

```
$ oc -n default run demo-service --image=gcr.io/google_containers/echoserver:1.4
```

2. 运行以下命令，将 pod 公开为服务，并将入口限制为您指定的出口 IP 地址：

```
$ cat <<EOF | oc apply -f -
apiVersion: v1
kind: Service
metadata:
  name: demo-service
  namespace: default
  annotations:
    service.beta.kubernetes.io/aws-load-balancer-scheme: "internal"
    service.beta.kubernetes.io/aws-load-balancer-internal: "true"
spec:
  selector:
    run: demo-service
  ports:
    - port: 80
      targetPort: 8080
  type: LoadBalancer
  externalTrafficPolicy: Local
  # NOTE: this limits the source IPs that are allowed to connect to our service.  It
  #       is being used as part of this demo, restricting connectivity to our egress
  #       IP addresses only.
  # NOTE: these egress IPs are within the subnet range(s) in which my worker nodes
  #       are deployed.
  loadBalancerSourceRanges:
    - 10.10.100.254/32
    - 10.10.150.254/32
    - 10.10.200.254/32
    - 10.10.100.253/32
    - 10.10.150.253/32
    - 10.10.200.253/32
EOF
```

3. 运行以下命令，检索负载均衡器主机名并将其保存为环境变量：

```
$ export LOAD_BALANCER_HOSTNAME=$(oc get svc -n default demo-service -o json
| jq -r '.status.loadBalancer.ingress[].hostname')
```

13.6.2. 测试命名空间出口

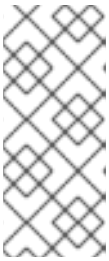
1. 启动交互式 shell 以测试命名空间出站规则：

```
$ oc run \
  demo-egress-ns \
  -it \
  --namespace=demo-egress-ns \
  --env=LOAD_BALANCER_HOSTNAME=$LOAD_BALANCER_HOSTNAME \
  --image=registry.access.redhat.com/ubi9/ubi -- \
  bash
```

2. 向负载均衡器发送请求并确保您可以成功连接：

```
$ curl -s http://$LOAD_BALANCER_HOSTNAME
```

3. 检查输出是否成功连接：



注意

client_address 是负载均衡器的内部 IP 地址，而不是您的出口 IP。您可以通过将服务连接到 .spec.loadBalancerSourceRanges 来验证客户端地址是否已正确配置。

输出示例

```
CLIENT VALUES:
client_address=10.10.207.247
command=GET
real path=/
query=nil
request_version=1.1
request_uri=http://internal-a3e61de18bfca4a53a94a208752b7263-148284314.us-east-
1.elb.amazonaws.com:8080/

SERVER VALUES:
server_version=nginx: 1.10.0 - lua: 10001

HEADERS RECEIVED:
accept=*/*
host=internal-a3e61de18bfca4a53a94a208752b7263-148284314.us-east-
1.elb.amazonaws.com
user-agent=curl/7.76.1
BODY:
-no body in request-
```

4. 运行以下命令来退出 pod：

```
$ exit
```

13.6.3. 测试 pod 出口

1. 启动交互式 shell 以测试 pod 出站规则：

```
$ oc run \
  demo-egress-pod \
  -it \
  --namespace=demo-egress-pod \
  --env=LOAD_BALANCER_HOSTNAME=$LOAD_BALANCER_HOSTNAME \
  --image=registry.access.redhat.com/ubi9/ubi -- \
  bash
```

2. 运行以下命令，向负载均衡器发送请求：

```
$ curl -s http://$LOAD_BALANCER_HOSTNAME
```

3. 检查输出是否成功连接：

注意

client_address 是负载均衡器的内部 IP 地址，而不是您的出口 IP。您可以通过将服务连接到 .spec.loadBalancerSourceRanges 来验证客户端地址是否已正确配置。

输出示例

```
CLIENT VALUES:
client_address=10.10.207.247
command=GET
real path=/
query=nil
request_version=1.1
request_uri=http://internal-a3e61de18bfca4a53a94a208752b7263-148284314.us-east-
1.elb.amazonaws.com:8080/

SERVER VALUES:
```

```
server_version=nginx: 1.10.0 - lua: 10001

HEADERS RECEIVED:
accept=*/*
host=internal-a3e61de18bfca4a53a94a208752b7263-148284314.us-east-
1.elb.amazonaws.com
user-agent=curl/7.76.1
BODY:
-no body in request-
```

4. 运行以下命令来退出 pod：

```
$ exit
```

13.6.4. 可选：测试会阻止的出口

1. 可选： 运行以下命令来测试当出口规则没有应用时流量被成功阻止：

```
$ oc run \
  demo-egress-pod-fail \
  -it \
  --namespace=demo-egress-pod \
  --env=LOAD_BALANCER_HOSTNAME=$LOAD_BALANCER_HOSTNAME \
  --image=registry.access.redhat.com/ubi9/ubi -- \
  bash
```

2. 运行以下命令，向负载均衡器发送请求：

```
$ curl -s http://$LOAD_BALANCER_HOSTNAME
```

3. 如果命令失败，则出口可以成功阻止。

4. 运行以下命令来退出 pod：

```
$ exit
```

13.7. 清理集群

1. 运行以下命令来清理集群：

```
$ oc delete svc demo-service -n default; \
$ oc delete pod demo-service -n default; \
$ oc delete project demo-egress-ns; \
$ oc delete project demo-egress-pod; \
$ oc delete egressip demo-egress-ns; \
$ oc delete egressip demo-egress-pod
```

2. 运行以下命令来清理分配的节点标签：

警告

如果您依赖机器池的节点标签，这个命令会替换这些标签。在 **--labels** 字段中输入所需标签，以确保您的节点标签保留。

```
$ rosa update machinepool ${ROSA_MACHINE_POOL_NAME} \
  --cluster="${ROSA_CLUSTER_NAME}" \
  --labels ""
```

第 14 章 教程：使用自定义域和 TLS 证书更新组件路由

本指南演示了如何修改 Red Hat OpenShift Service on AWS (ROSA)版本 4.14 及更高版本中的 Web 控制台、OAuth 服务器和下载组件路由的主机名和 TLS 证书。[1]

对组件路由所做的更改[2] 在本指南中，详细介绍了自定义 内部 OAuth 服务器 URL、控制台路由和 下载路由 OpenShift Container Platform 文档。

14.1. 前提条件

- ROSA CLI (rosa)版本 1.2.37 或更高版本
- AWS CLI (aws)
- ROSA Classic 集群版本 4.14 或更高版本

注意

目前不支持使用 HCP 的 ROSA。

- OpenShift CLI (oc)
- jq CLI
- 使用具有 cluster-admin 角色的用户访问集群。
- openssl （用于生成演示 SSL/TLS 证书）

14.2. 设置您的环境

1. 使用具有 cluster-admin 权限的账户登录集群。

2. 为集群名称配置环境变量：

```
$ export CLUSTER_NAME=$(oc get infrastructure cluster -o=jsonpath="
{.status.infrastructureName}"  | sed 's/-[a-z0-9]\{5\}$//')
```

3. 在移动到下一部分前，请确定所有字段都正确输出：

```
$ echo "Cluster: ${CLUSTER_NAME}"
```

输出示例

```
Cluster: my-rosa-cluster
```

14.3. 查找当前路由

1. 验证您是否可以访问其默认主机名上的组件路由。

您可以通过查询 openshift-console 和 openshift-authentication 项目中的路由列表来查找主机名。

```
$ oc get routes -n openshift-console
$ oc get routes -n openshift-authentication
```

输出示例

```
NAME       HOST/PORT                                                  PATH      SERVICES
PORT    TERMINATION         WILDCARD
console    console-openshift-console.apps.my-example-cluster-
aws.z9a9.p1.openshiftapps.com    ... 1 more  console    https   reencrypt/Redirect
None
downloads   downloads-openshift-console.apps.my-example-cluster-
aws.z9a9.p1.openshiftapps.com  ... 1 more  downloads  http    edge/Redirect       None
NAME            HOST/PORT                                          PATH       SERVICES
PORT   TERMINATION          WILDCARD
oauth-openshift   oauth-openshift.apps.my-example-cluster-
aws.z9a9.p1.openshiftapps.com ... 1 more  oauth-openshift   6443
passthrough/Redirect   None
```

–

在这个输出中，您可以看到我们的基本主机名为 z9a9.p1.openshiftapps.com。

2. 运行以下命令，获取默认入口的 ID：

```
$ export INGRESS_ID=$(rosa list ingress -c ${CLUSTER_NAME} -o json | jq -r '.[] |
select(.default == true) | .id')
```

3. 在移动到下一部分前，请确定所有字段都正确输出：

```
$ echo "Ingress ID: ${INGRESS_ID}"
```

输出示例

```
Ingress ID: r3l6
```

通过运行这些命令，您可以看到集群的默认组件路由是：

- console-openshift-console.apps.my-example-cluster-aws.z9a9.p1.openshiftapps.com for Console
- downloads-openshift-console.apps.my-example-cluster-aws.z9a9.p1.openshiftapps.com for Downloads
- oauth-openshift.apps.my-example-cluster-aws.z9a9.p1.openshiftapps.com for OAuth

我们可以使用 rosa edit ingress 命令更改每个服务的主机名，并为我们的所有组件路由添加一个 TLS 证书。相关的参数包括在 rosa edit ingress 命令的命令行帮助摘录中：

```
$ rosa edit ingress -h
Edit a cluster ingress for a cluster. Usage:
  rosa edit ingress ID [flags]
  [...]
  --component-routes string              Component routes settings. Available keys [oauth,
console, downloads]. For each key a pair of hostname and tlsSecretRef is expected to be
supplied. Format should be a comma separate list 'oauth: hostname=example-
hostname;tlsSecretRef=example-secret-ref,downloads:...'
```

在本例中，我们将使用以下自定义组件路由：

- Console 的 console.my-new-domain.dev
- download.my-new-domain.dev for Downloads
- oauth.my-new-domain.dev for OAuth

14.4. 为每个组件路由创建有效的 TLS 证书

在本节中，我们创建三个单独的自签名证书密钥对，然后信任它们，以使用真实的 Web 浏览器访问我们的新组件路由。

警告

这仅用于演示目的，不建议将其作为生产工作负载的解决方案。请参考您的证书颁发机构了解如何为生产工作负载创建具有类似属性的证书。

重要

要防止 HTTP/2 连接合并出现问题，您必须为每个端点使用单独的证书。不支持使用通配符或 SAN 证书。

1. 为每个组件路由生成一个证书，注意将证书的主题(-subj)设置为您要使用的组件路由的自定义域：

示例

```
$ openssl req -newkey rsa:2048 -new -nodes -x509 -days 365 -keyout key-console.pem
-out cert-console.pem -subj "/CN=console.my-new-domain.dev"
$ openssl req -newkey rsa:2048 -new -nodes -x509 -days 365 -keyout key-
downloads.pem -out cert-downloads.pem -subj "/CN=downloads.my-new-domain.dev"
$ openssl req -newkey rsa:2048 -new -nodes -x509 -days 365 -keyout key-oauth.pem -
out cert-oauth.pem -subj "/CN=oauth.my-new-domain.dev"
```

这会生成三对 .pem 文件、key-<component>.pem 和 cert-<component>.pem。

14.5. 将证书作为 SECRET 添加到集群中

1. 在 openshift-config 命名空间中创建三个 TLS secret。

   在本指南的稍后更新组件路由时，这些 secret 将成为您的 secret 引用。

```
$ oc create secret tls console-tls --cert=cert-console.pem --key=key-console.pem -n
openshift-config
$ oc create secret tls downloads-tls --cert=cert-downloads.pem --key=key-
downloads.pem -n openshift-config
$ oc create secret tls oauth-tls --cert=cert-oauth.pem --key=key-oauth.pem -n
openshift-config
```

14.6. 在集群中查找负载均衡器的主机名

当您创建集群时，该服务会创建一个负载均衡器，并为该负载均衡器生成主机名。为了为集群创建 DNS 记录，我们需要知道负载均衡器主机名。

您可以通过针对 openshift-ingress 命名空间运行 oc get svc 命令来查找主机名。负载均衡器的主机名是与 openshift-ingress 命名空间中的 router-default 服务关联的 EXTERNAL-IP。

```
$ oc get svc -n openshift-ingress
NAME          TYPE          CLUSTER-IP     EXTERNAL-IP                                PORT(S)
AGE
```

```
router-default  LoadBalancer  172.30.237.88  a234gsr3242rsfsfs-1342r624.us-east-
1.elb.amazonaws.com  80:31175/TCP,443:31554/TCP  76d
```

在我们的示例中，主机名为 234gsr3242rsfsfs-1342r624.us-east-1.elb.amazonaws.com。

稍后保存这个值，因为我们需要为新组件路由主机名配置 DNS 记录。

14.7. 在托管供应商中添加组件路由 DNS 记录

在托管供应商中，添加 DNS 记录，将新组件路由主机名的 CNAME 映射到我们上一步中找到的负载均衡器主机名。

14.8. 使用 ROSA CLI 更新组件路由和 TLS SECRET

更新 DNS 记录后，您可以使用 ROSA CLI 更改组件路由。

1. 使用 rosa edit ingress 命令，使用新的基域和与其关联的 secret 引用更新默认入口路由，需要小心更新每个组件路由的主机名。

```
$ rosa edit ingress -c ${CLUSTER_NAME} ${INGRESS_ID} --component-routes
'console: hostname=console.my-new-domain.dev;tlsSecretRef=console-
tls,downloads: hostname=downloads.my-new-domain.dev;tlsSecretRef=downloads-
tls,oauth: hostname=oauth.my-new-domain.dev;tlsSecretRef=oauth-tls'
```

注意

您还可以通过保留您不想更改设置为空字符串的组件路由来只编辑组件路由的子集。例如，如果您只想更改 Console 和 OAuth 服务器主机名和 TLS 证书，您将运行以下命令：

```
$ rosa edit ingress -c ${CLUSTER_NAME} ${INGRESS_ID} --component-
routes 'console: hostname=console.my-new-
domain.dev;tlsSecretRef=console-tls,downloads:
hostname="";tlsSecretRef="", oauth: hostname=oauth.my-new-
domain.dev;tlsSecretRef=oauth-tls'
```

2. 运行 rosa list ingress 命令来验证您的更改是否成功：

```
$ rosa list ingress -c ${CLUSTER_NAME} -ojson | jq ".[] | select(.id ==
\"${INGRESS_ID}\") | .component_routes"
```

输出示例

```
{
  "console": {
    "kind": "ComponentRoute",
    "hostname": "console.my-new-domain.dev",
    "tls_secret_ref": "console-tls"
  },
  "downloads": {
    "kind": "ComponentRoute",
    "hostname": "downloads.my-new-domain.dev",
    "tls_secret_ref": "downloads-tls"
  },
  "oauth": {
    "kind": "ComponentRoute",
    "hostname": "oauth.my-new-domain.dev",
    "tls_secret_ref": "oauth-tls"
  }
}
```

3. 将您的证书添加到本地系统上的信任存储中，然后确认您可以使用本地 Web 浏览器通过新路由访问组件。

14.9. 使用 ROSA CLI 将组件路由重置为默认值

如果要将组件路由重置为默认配置，请运行以下 rosa edit ingress 命令：

```
$ rosa edit ingress -c ${CLUSTER_NAME} ${INGRESS_ID} --component-routes 'console:
hostname="";tlsSecretRef="",downloads: hostname="";tlsSecretRef="", oauth:
hostname="";tlsSecretRef=""'
```

[1] 在 4.14 之前的 Red Hat OpenShift Service on AWS ROSA 版本上修改这些路由通常不被支持。但是，如果您有一个使用版本 4.13 的集群，可以通过 提交支持问题单，在版本 4.13 集群上启用对此功能的支持。

[2]
我们使用术语"组件路由"来指代首次安装 ROSA 时提供的 OAuth、控制台和下载路由。

第 15 章 ROSA 入门

15.1. 教程：什么是 ROSA

Red Hat OpenShift Service on AWS (ROSA)是一个完全管理的 turnkey 应用平台，它允许您专注于最重要的内容，通过构建和部署应用程序来为您的客户提供价值。红帽和 AWS SRE 专家管理底层平台，因此您不必担心基础架构管理。ROSA 提供与各种 AWS 计算、数据库、分析、机器学习、联网、移动和其他服务进行无缝整合，进一步加快向客户构建和交付不同体验。

ROSA 使用 AWS 安全令牌服务(STS)获取凭证来管理 AWS 帐户中的基础架构。AWS STS 是一个全局 Web 服务，它为 IAM 用户或联邦用户创建临时凭证。ROSA 使用它来分配短期、有限特权的安全凭证。这些凭证与特定于发布 AWS API 调用的每个组件的 IAM 角色关联。此方法与云服务资源管理中最低特权和安全实践的主体一致。ROSA 命令行界面(CLI)工具管理为唯一任务分配的 STS 凭证，并在 OpenShift 功能中对 AWS 资源执行操作。

15.1.1. ROSA 的主要功能

- 原生 AWS 服务： 通过 AWS 管理控制台访问和使用 Red Hat OpenShift on-service join 体验。
- 灵活的、基于消费的定价： 按业务需求扩展，并随着您提供灵活的定价和按需提供的每小时或年度计费模式支付费用。
- Red Hat OpenShift 和 AWS 的单个 bill：客户会从 AWS 收到单个 bill，以供 Red Hat OpenShift 和 AWS 使用。
- 完全集成的支持体验： 红帽站点可靠性工程师(SRE)通过联合红帽和 Amazon 支持以及 99.95% 服务级别协议(SLA)执行安装、管理、维护和升级。
- AWS 服务集成： AWS 具有强大的云服务组合，如计算、存储、网络、数据库、分析和机器学习。所有这些服务均可通过 ROSA 直接访问。这样，通过熟悉的管理界面，可以更轻松地在全局范围内构建、操作和扩展。
- 最大可用性： 在支持的区域的多个可用区中部署集群，以最大化并保持您要求最苛刻的任务关键型应用程序和数据的高可用性。
- 集群节点扩展： 轻松添加或删除计算节点，以匹配资源需求。

- 优化的集群 ： 从内存优化、计算优化或通用 EC2 实例类型中选择，集群大小以满足您的需求。

- 全局可用性： 请参阅 产品区域可用性 页面，以查看 ROSA 在全局范围内可用。

15.1.2. ROSA 和 Kubernetes

在 ROSA 中，您可以捆绑部署和管理容器所需的所有内容，包括容器管理、Operator、网络、负载均衡、服务网格、CI/CD、防火墙、监控、registry、身份验证和授权功能。这些组件共同测试，以统一操作作为完整的平台。自动化集群操作，包括无线平台升级，进一步增强了 Kubernetes 体验。

15.1.3. 基本职责

通常，集群部署和 upkeep 是红帽的职责，而应用程序、用户和数据是客户的职责。有关职责的详细分类，请参阅 责任列表。

15.1.4. 路线图和功能请求

访问 ROSA 路线图 以保持最新状态，以及当前开发中的功能状态。如果您对产品团队有任何建议，请创建一个新问题。

15.1.5. AWS 区域可用性

有关 ROSA 可用的最新视图，请参阅 产品区域可用性 页面。

15.1.6. 合规认证

ROSA 目前符合 SOC-2 类型 2、SOC 3、ISO-27001、ISO 27017、ISO 27018、HIPAA、GDPR 和 PCI-DSS。我们目前还致力于 FedRAMP High。

15.1.7. 节点

15.1.7.1. 跨多个 AWS 区域的 worker 节点

ROSA 集群中的所有节点都必须位于同一 AWS 区域。对于为多个可用区配置的集群，control plane 节点和 worker 节点将在可用区间分布。

15.1.7.2. 最小 worker 节点数量

对于 ROSA 集群，最小是 2 个 worker 节点用于单个可用区，3 个 worker 节点用于多个可用区。

15.1.7.3. 底层节点操作系统

与所有 OpenShift v4.x 产品一样，control plane、infra 和 worker 节点都运行 Red Hat Enterprise Linux CoreOS (RHCOS)。

15.1.7.4. 节点休眠或关闭

目前，ROSA 没有节点的休眠功能或关闭功能。shutdown 和 hibernation 功能是一个 OpenShift 平台的功能，它尚未足够成熟，用于广泛的云服务。

15.1.7.5. worker 节点支持的实例

有关 worker 节点支持的实例的完整列表，请参阅 AWS 实例类型。还支持 Spot 实例。

15.1.7.6. 节点自动扩展

通过自动扩展，您可以根据当前工作负载自动调整集群的大小。如需了解更多详细信息，请参阅关于集群中的自动扩展节点。

15.1.7.7. worker 节点的最大数量

每个 ROSA 集群的最大 worker 节点数量为 180 worker 节点。有关节点数的详情，请参阅 限制和可扩展性。

ROSA 文档中提供了帐户范围和每个集群角色的列表。

15.1.8. 管理员

除了访问所有用户创建项目外，ROSA 客户管理员可以管理用户和配额。

15.1.9. OpenShift 版本和升级

ROSA 是一个基于 OpenShift Container Platform 的受管服务。您可以在 ROSA 文档中 查看当前版本和生命周期日期。

客户可以升级到 OpenShift 的最新版本，并使用来自该 OpenShift 版本的功能。如需更多信息，请参阅 生命周期日期。并非所有 OpenShift 功能都可用于 ROSA。如需更多信息，请参阅 服务定义。

15.1.10. 支持

您可以直接从 OpenShift Cluster Manager 打开 ticket。有关获取支持的详情，请参阅 ROSA 支持文档。

您还可以访问 红帽客户门户网站来搜索 或浏览与红帽产品相关的文章和解决方案，或向红帽支持提交支持问题单。

15.1.10.1. 有限支持

如果在"生命周期结束"前没有升级 ROSA 集群，集群将继续以有限的支持状态运行。该集群的 SLA 将不再适用，但您仍可以获得对该集群的支持。如需了解更多详细信息，请参阅 有限的支持状态 文档。

其他支持资源

- 红帽支持
- AWS Support

  AWS 支持客户必须具有有效的 AWS 支持合同

15.1.11. 服务级别协议(SLA)

详情请查看 ROSA SLA 页面。

15.1.12. 通知和通信

红帽将通过电子邮件和混合云控制台服务日志提供有关新红帽和 AWS 功能、更新和计划的维护通知。

15.1.13. AWS (OBSA)的 Open Service Broker

您可以将 OSBA 与 ROSA 搭配使用。但是，首选方法是 Kubernetes 的最新 AWS Controller。如需有关 OSBA 的更多信息，请参阅 AWS 的 Open Service Broker。

15.1.14. Offboarding

客户可以随时停止使用 ROSA，并将其应用程序迁移到内部、私有云或其他云供应商。标准保留实例(RI)策略适用于未使用的 RI。

15.1.15. 身份验证

ROSA 支持以下身份验证机制：OpenID Connect (OAuth2 的配置集)、Google OAuth、GitHub OAuth、GitLab 和 LDAP。

15.1.16. SRE 集群访问

所有 SRE 集群访问都由 MFA 保护。如需了解更多详细信息，请参阅 SRE 访问。

15.1.17. Encryption

15.1.17.1. 加密密钥

ROSA 使用存储在 KMS 中的密钥来加密 EBS 卷。客户也可以选择在集群创建时提供自己的 KMS 密钥。

15.1.17.2. KMS 密钥

如果您指定了 KMS 密钥，control plane、基础架构和 worker 节点根卷，并使用密钥加密持久性卷。

15.1.17.3. 数据加密

默认情况下，还有加密。AWS Storage 平台会在保留数据前自动加密数据，并在检索前解密数据。如需了解更多详细信息，请参阅 AWS EBS 加密。

您还可以在集群中加密 etcd，将其与 AWS 存储加密合并。这会加倍的加密，这会增加 20% 的性能命中。如需了解更多详细信息，请参阅 etcd 加密 文档。

15.1.17.4. etcd 加密

etcd 加密只能在集群创建时启用。

注意

etcd 加密会导致额外的开销，但存在不计的安全风险缓解方案。

15.1.17.5. etcd 加密配置

etcd 加密的配置与 OpenShift Container Platform 中的相同。使用 aescbc cypher，设置在集群部署期间被修补。如需了解更多详细信息，请参阅 Kubernetes 文档。

15.1.17.6. EBS 加密的多区域 KMS 密钥

目前，ROSA CLI 不接受用于 EBS 加密的多区域 KMS 密钥。此功能是我们进行产品更新的积压。如果在集群创建时定义，ROSA CLI 接受 EBS 加密的单一区域 KMS 密钥。

15.1.18. 基础架构

ROSA 使用几个不同的云服务，如虚拟机、存储和负载均衡器。您可以在 AWS 先决条件 中看到定义的列表。

15.1.19. 凭证方法

有两种凭证方法可以授予红帽在 AWS 帐户中执行所需操作所需的权限：带有 STS 的 AWS 或具有 admin 权限的 IAM 用户。使用 STS 的 AWS 是首选的方法，IAM 用户方法最终会被弃用。AWS 与 STS 更好地与云服务资源管理中最低特权和安全实践的原则保持一致。

15.1.20. 先决条件权限或失败错误

检查 ROSA CLI 的更新版本。ROSA CLI 的每个发行版本都位于两个位置：Github 和红帽签名的二进制版本。https://github.com/openshift/rosa/releases

15.1.21. Storage

请参阅服务定义的 storage 部分。

OpenShift 包含 AWS EFS 的 CSI 驱动程序。如需更多信息，请参阅 在 AWS 上为 Red Hat OpenShift Service 设置 AWS EFS。

15.1.22. 使用 VPC

安装时，您可以选择部署到现有的 VPC 或拥有您自己的 VPC。然后，您可以选择所需的子网，并提供有效的 CIDR 范围，该范围包括安装程序在使用这些子网时的子网。

ROSA 允许多个集群共享相同的 VPC。一个 VPC 上的集群数量受剩余的 AWS 资源配额和 CIDR 范围的限制。如需更多信息，请参阅 CIDR 范围定义。

15.1.23. 网络插件

ROSA 使用 OpenShift OVN-Kubernetes 默认 CNI 网络供应商。

15.1.24. 跨命名空间网络

集群管理员可以使用 NetworkPolicy 对象根据项目自定义和拒绝跨命名空间。如需更多信息，请参阅使用网络策略配置多租户隔离。

15.1.25. 使用 Prometheus 和 Grafana

您可以使用 Prometheus 和 Grafana 监控容器，并使用 OpenShift User Workload Monitoring 管理容量。这是 OpenShift Cluster Manager 中的复选框选项。

15.1.26. 审计日志来自集群 control-plane 的输出

如果 Cluster Logging Operator Add-on 已添加到集群中，则审计日志可通过 CloudWatch 获得。如果没有，则支持请求将允许您请求一些审计日志。对于导出并发送到客户，可以请求较小的目标日志和有时间的日志。可用的审计日志的选择由平台安全和合规性类别中的 SRE 决定。集群整个日志的导出的请求将被拒绝。

15.1.27. AWS 权限绑定

您可以在集群的策略中使用 AWS 权限 Boundary。

15.1.28. AMI

ROSA worker 节点使用与 OSD 和 OpenShift Container Platform 不同的 AMI。Control Plane 和 Infra node AMI 在同一个版本中的产品之间很常见。

15.1.29. 集群备份

ROSA STS 集群没有备份。用户必须为应用程序和数据拥有自己的备份策略。如需更多信息，请参阅我们的 备份策略。

15.1.30. 自定义域

您可以为应用程序定义自定义域。如需更多信息，请参阅为应用程序配置自定义域。

15.1.31. ROSA 域证书

红帽基础架构(Hive)管理默认应用程序入口的证书轮转。

15.1.32. 断开连接的环境

ROSA 不支持 air-gapped 断开连接的环境。ROSA 集群必须具有到互联网的出口，才能访问我们的 registry、S3 并发送指标。该服务需要多个出口端点。Ingress 可以限制为 Red Hat SREs 和 VPN 用于客户访问。

其它资源

- ROSA 产品页面：
  - 红帽产品页面
  - AWS 产品页面
  - 红帽客户门户网站
- ROSA 特定资源

    - AWS ROSA 入门指南
    - ROSA 文档
    - ROSA 服务定义
    - ROSA 责任分配列表
    - 了解进程和安全
    - 关于可用性
    - 更新生命周期
    - 限制和可扩展性
    - ROSA 路线图
- 了解 OpenShift
- OpenShift Cluster Manager
- 红帽支持

15.2. 教程：使用 AWS STS 的 ROSA 解释

本教程概述了允许 Red Hat OpenShift Service on AWS (ROSA)与用户的 Amazon Web Service (AWS)帐户中的资源交互的两个选项。它详细介绍了 ROSA 使用安全令牌服务(STS)获取必要凭证的组件和流程。它还回顾了为什么使用 STS 的 ROSA 是更安全的、首选的方法。

注意

此内容目前涵盖使用 AWS STS 的 ROSA Classic。对于使用 AWS STS 托管 control plane (HCP)的 ROSA，请参阅 AWS STS 和使用 HCP 的 ROSA 解释。

本教程将：

- Enumerate 两个部署选项：
  - 使用 IAM 用户的 ROSA 用户
  - 使用 STS 的 ROSA
- 解释两个选项之间的区别
- 解释为什么使用 STS 的 ROSA 更安全和首选选项
- 解释使用 STS 的 ROSA 的工作原理

15.2.1. 部署 ROSA 的不同凭证方法

作为 ROSA 的一部分，红帽管理 AWS 帐户中的基础架构资源，且必须授予所需的权限。目前有两种方法来授予这些权限：

- 使用带有 AdministratorAccess 策略的静态 IAM 用户凭证

  在本教程中，这被称为 "ROSA with IAM Users"。这不是首选凭证方法。

- 使用带有简短的动态令牌的 AWS STS

  在本教程中，这被称为 "ROSA with STS"。它是首选凭证方法。

15.2.1.1. 使用 IAM 用户的 ROSA 用户

当首次发布 ROSA 时，唯一凭证方法是 IAM 用户的 ROSA。此方法授予具有 AdministratorAccess 策略完全访问权限的 IAM 用户，以便在使用 ROSA 的 AWS 帐户中创建所需资源。然后，集群可以根据需要创建并扩展其凭证。

15.2.1.2. 使用 STS 的 ROSA

使用 STS 的 ROSA 授予用户对 AWS 帐户中资源的有限、短期访问权限。STS 方法使用预定义的角色和策略为 IAM 用户或经过身份验证的用户授予临时的、最低特权权限。在请求后，凭据通常会在一小时后过期。过期后，AWS 不再识别它们，不再从 API 请求访问它们。如需更多信息，请参阅 AWS 文档。虽然当前启用了带有 IAM 用户和带有 STS 的 ROSA，但使用 STS 的 ROSA 是首选的和推荐选项。

15.2.2. 使用 STS 安全性的 ROSA

几个关键组件使带有 STS 的 ROSA 比使用 IAM 用户进行 ROSA 更安全：

- 用户提前创建的明确和有限的角色和策略集合。用户知道每个请求的权限以及所使用的每个角色。
- 该服务不能在这些权限之外执行任何操作。
- 每当服务需要执行某个操作时，它会获取以一小时或更少形式过期的凭证。这意味着不需要轮转或撤销凭证。另外，凭证过期会降低凭证泄漏和重复使用的风险。

15.2.3. AWS STS 解释

ROSA 使用 AWS STS 为特定的和隔离的 IAM 角色，为特定的和隔离的 IAM 角色授予具有短期安全凭证的最低特权权限。凭证与特定于每个组件的 IAM 角色关联，并发出 AWS API 调用的集群。此方法与云服务资源管理中最低特权和安全实践的原则一致。ROSA 命令行界面(CLI)工具管理为唯一任务分配的 STS 角色和策略，并作为 OpenShift 功能的一部分对 AWS 资源执行操作。

必须为每个 ROSA 集群创建 STS 角色和策略。为了更容易实现这一点，安装工具提供了创建角色所需的所有命令和文件，以及一个允许 CLI 自动创建角色和策略的选项。如需有关 different- mode 选项的更多信息，请参阅使用自定义创建带有 STS 的 ROSA 集群。

15.2.4. 特定于使用 STS 的 ROSA 的组件

-

AWS 基础架构 - 这提供了集群所需的基础架构。它包含实际的 EC2 实例、存储和网络组件。请参阅 AWS 计算类型，以查看计算节点支持的实例类型，以及用于 control plane 和基础架构节点配置的 置备 AWS 基础架构。

- AWS STS - 请参阅上面的凭证方法部分。
- OpenID Connect (OIDC) - 这为集群 Operator 提供了与 AWS 进行身份验证的机制，通过信任策略假设集群角色，并从 STS 获取临时凭证来进行所需的 API 调用。
- 角色和策略 - 角色和策略是 ROSA 使用 STS 和带有 IAM 用户的 ROSA 之间的主要区别之一。对于使用 STS 的 ROSA，ROSA 使用的角色和策略被分为集群范围的角色和策略，以及 Operator 角色和策略。

  策略决定了每个角色允许的操作。有关各个角色和策略的更多详情，请参阅使用 STS 的 ROSA 集群的关于 IAM 资源。

  - 集群范围的角色有：
    - ManagedOpenShift-Installer-Role
    - ManagedOpenShift-ControlPlane-Role
    - ManagedOpenShift-Worker-Role
    - ManagedOpenShift-Support-Role
  - 帐户范围内的策略有：
    - ManagedOpenShift-Installer-Role-Policy
    - ManagedOpenShift-ControlPlane-Role-Policy

- ManagedOpenShift-Worker-Role-Policy
- ManagedOpenShift-Support-Role-Policy
- ManagedOpenShift-openshift-ingress-operator-cloud-credentials [1]
- ManagedOpenShift-openshift-cluster-csi-drivers-ebs-cloud-credent [1]
- ManagedOpenShift-openshift-cloud-network-config-controller-cloud [1]
- ManagedOpenShift-openshift-machine-api-aws-cloud-credentials [1]
- ManagedOpenShift-openshift-cloud-credential-operator-cloud-crede [1]
- ManagedOpenShift-openshift-image-registry-installer-cloud-creden [1]

1. 此策略供集群 Operator 角色使用，如下所列。Operator 角色在第二个步骤中创建，因为它们依赖于现有集群名称，且无法与集群范围的角色同时创建。

- Operator 角色是：
  - <cluster-name\>-xxxx-openshift-cluster-csi-drivers-ebs-cloud-credent
  - <cluster-name\>-xxxx-openshift-cloud-network-config-controller-cloud
  - <cluster-name\>-xxxx-openshift-machine-api-aws-cloud-credentials
  - <cluster-name\>-xxxx-openshift-cloud-credential-operator-cloud-crede

- <cluster-name\>-xxxx-openshift-image-registry-installer-cloud-creden
- <cluster-name\>-xxxx-openshift-ingress-operator-cloud-credentials

- 为每个账户和 Operator 角色创建信任策略。

15.2.5. 部署 ROSA STS 集群

您不应该从头开始创建以下步骤中列出的资源。ROSA CLI 为您创建所需的 JSON 文件，并输出您需要的命令。ROSA CLI 也可以进一步执行这一步，并在需要时为您运行命令。

使用 STS 集群部署 ROSA 的步骤

1. 创建集群范围的角色和策略。
2. 将权限策略分配给对应的集群范围的角色。
3. 创建集群。
4. 创建 Operator 角色和策略。
5. 为对应的 Operator 角色分配权限策略。
6. 创建 OIDC 供应商。

角色和策略可由 ROSA CLI 自动创建，也可以使用 ROSA CLI 中的 --mode manual 或 --mode auto 标志手动创建。有关部署的详情，请参阅 使用自定义创建集群 或 部署集群指南。

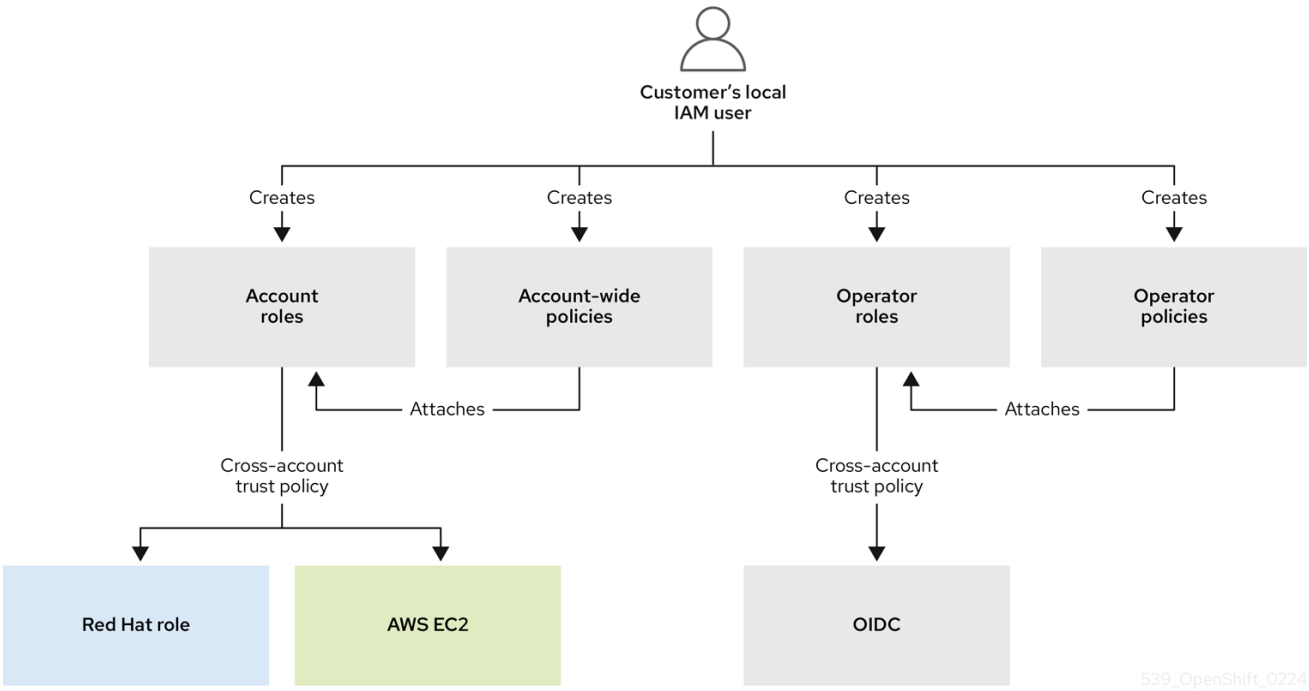
15.2.6. 使用 STS 工作流的 ROSA

用户创建所需的集群范围的角色和集群范围的策略。如需更多信息，请参阅本教程中的组件部分。在创建角色时，会创建一个信任策略，称为跨帐户信任策略，该策略允许红帽拥有的角色假定角色。还会为

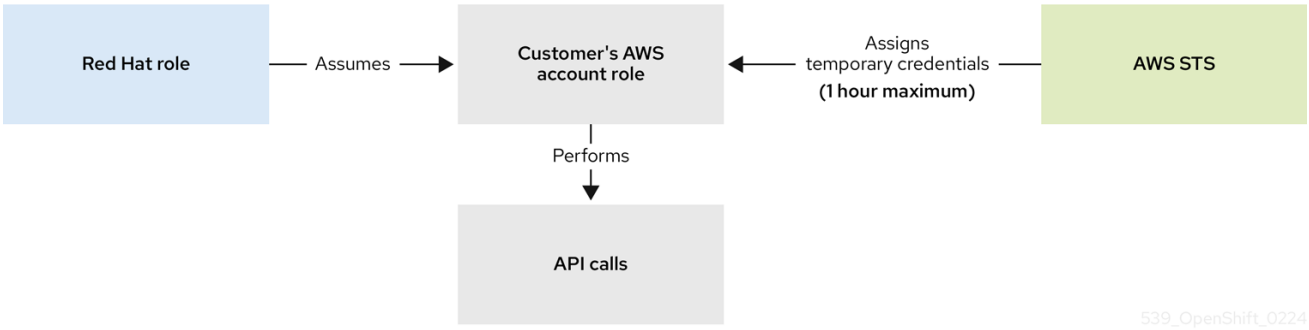
EC2 服务创建信任策略，它允许 EC2 实例上的工作负载假定角色和获取凭据。然后，用户可以为每个角色分配对应的权限策略。

创建集群范围的角色和策略后，用户可以创建集群。启动集群创建后，会创建 Operator 角色，以便集群 Operator 可以发出 AWS API 调用。然后，这些角色被分配给之前创建的相应权限策略，以及带有 OIDC 供应商的信任策略。Operator 角色与集群范围的角色不同，它们最终代表需要访问 AWS 资源的 pod。因为用户无法将 IAM 角色附加到 pod，所以它们必须使用 OIDC 供应商创建信任策略，以便 Operator，因此 pod 可以访问它们所需的角色。

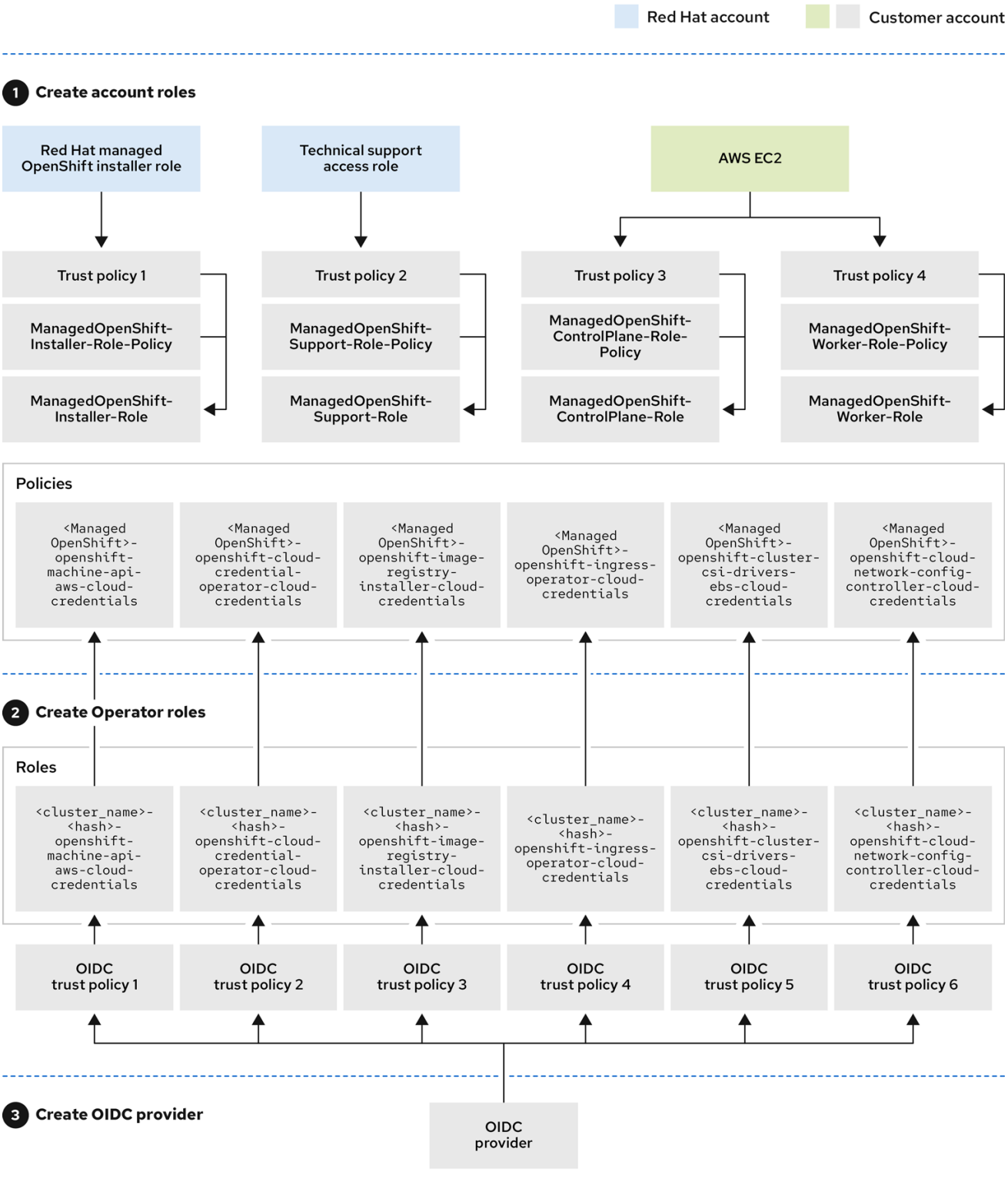
用户将角色分配给对应的策略权限后，最后一步是创建 OIDC 供应商。



当需要新角色时，当前使用红帽角色的工作负载将假定 AWS 帐户中的角色，从 AWS STS 获取临时凭证，并开始使用所假定角色的权限策略允许的用户 AWS 帐户中的 API 调用执行操作。凭据是临时的，最长持续时间为一小时。

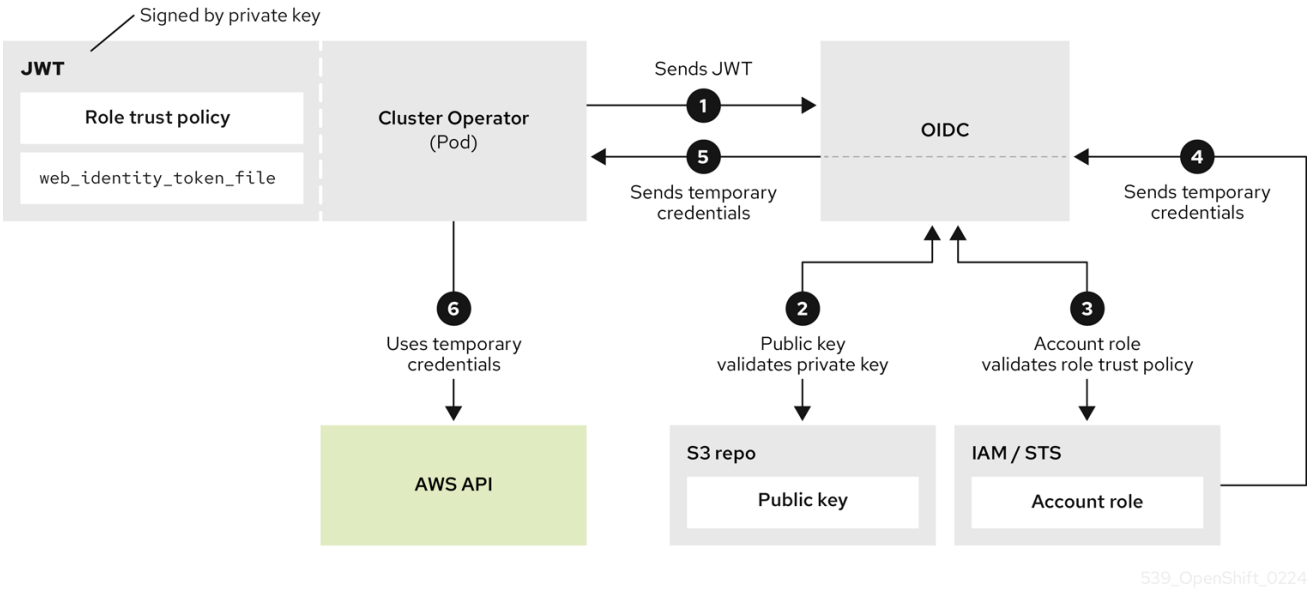


整个工作流在下图中描述：



Operator 使用以下流程获取必要的凭证来执行其任务。每个 Operator 都会被分配一个 Operator 角色、权限策略和带有 OIDC 供应商的信任策略。Operator 将通过将包含角色和令牌文件的 JSON Web 令牌(web_identity_token_file)传递给 OIDC 供应商来假定角色，然后使用公钥验证签名密钥。公钥是在集群创建过程中创建的，并存储在 S3 存储桶中。然后，Operator 会确认签名令牌文件中的主题与角色信任

策略中的角色匹配，以确保 OIDC 供应商只能获取允许的角色。然后，OIDC 供应商会向 Operator 返回临时凭证，以便 Operator 可以发出 AWS API 调用。有关视觉表示，请查看以下信息：



15.2.7. 使用 STS 的 ROSA 用例

在集群安装时创建节点

红帽安装程序使用 RH-Managed-OpenShift-Installer 角色和一个信任策略来假定客户帐户中 Managed-OpenShift-Installer-Role 角色。此过程从 AWS STS 返回临时凭证。安装程序开始使用从 STS 接收的临时凭证进行所需的 API 调用。安装程序在 AWS 中创建所需的基础架构。凭证在一小时内过期，安装程序无法再访问客户的帐户。

相同的流程也适用于支持问题单。在支持问题单中，红帽站点可靠性工程师(SRE)替换了安装程序。

扩展集群

machine-api-operator 使用 AssumeRoleWithWebIdentity 来假定 machine-api-aws-cloud-credentials 角色。这会启动集群 Operator 的顺序来接收凭证。machine-api-operator 角色现在可以发出相关的 API 调用来向集群添加更多 EC2 实例。

15.3. 部署集群

15.3.1. 教程：选择部署方法

本教程概述了部署集群的不同方法。选择最适合您的首选项和需求的部署方法。

15.3.1.1. 部署选项

如果您想要：

- 只有所需的 CLI 命令 - Simple CLI 指南
- 用户界面 - Simple UI 指南
- CLI 命令详情 - 详细的 CLI 指南
- 带有详情的用户界面 - 详细 UI 指南
- 使用 HCP 测试最新的 ROSA 技术 - ROSA

以上所有部署选项都适用于本教程。如果您首次进行此教程，则 简单 CLI 指南 是最简单的和推荐的方法。

15.3.2. 教程：简单 CLI 指南

本页概述了使用命令行界面(CLI)部署 Red Hat OpenShift Service on AWS (ROSA)集群的命令列表。

注意

虽然这个简单部署适用于教程设置，但生产中使用的集群应该使用更详细的方法进行部署。

15.3.2.1. 前提条件

- 您已完成了设置指南中的先决条件。

15.3.2.2. 创建帐户角色

对于每个 AWS 帐户和 y-stream OpenShift 版本，运行以下命令：

```
rosa create account-roles --mode auto --yes
```

15.3.2.3. 部署集群

1. 运行以下命令替换您自己的集群名称，使用默认配置创建集群：

```
rosa create cluster --cluster-name <cluster-name> --sts --mode auto --yes
```

2. 运行以下命令检查集群的状态：

```
rosa list clusters
```

15.3.3. 教程：详细的 CLI 指南

本教程概述了使用 ROSA CLI 部署 ROSA 集群的详细步骤。

15.3.3.1. CLI 部署模式

部署 ROSA 集群有两种模式。一个是自动的，它更快，并为您执行手动工作。另一个是 manual，要求您运行额外的命令，并允许您检查正在创建的角色和策略。本教程记录了这两个选项。

如果要快速创建集群，请使用自动选项。如果您希望了解正在创建的角色和策略，请使用 manual 选项。

在相关命令中使用 --mode 标志来选择部署模式。

--mode 的有效选项有：

- 手动 ： 创建角色和策略，并保存在当前目录中。您必须在下一步中手动运行提供的命令。此选项允许您在创建策略和角色前查看策略和角色。
- auto ： 使用当前 AWS 帐户自动创建和应用角色和策略。

提示

在本教程中，您可以使用任一部署方法。自动 模式速度更快，且步骤更少。

15.3.3.2. 部署工作流

整个部署工作流遵循以下步骤：

1. ROSA create account-roles - 这只为每个帐户 执行一次。创建后，不需要 为同一 y-stream 版本的更多集群再次创建帐户角色。
2. ROSA 创建集群
3. ROSA create operator-roles - 用于手动模式。
4. ROSA create oidc-provider - 只适用于手动模式。

对于同一帐户中的每个额外集群，则只有第 2 步用于自动模式。manual 模式需要第 2 到 4 步。

15.3.3.3. 自动模式

如果您希望 ROSA CLI 自动创建角色和策略来创建集群，请使用此方法。

15.3.3.3.1. 创建帐户角色

如果您 首次在这个帐户 中部署 ROSA，且您 还没有创建 帐户角色，则创建集群范围的角色和策略，包括 Operator 策略。

运行以下命令来创建集群范围的角色：

```
rosa create account-roles --mode auto --yes
```

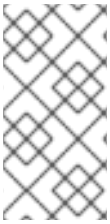
输出示例

```
I: Creating roles using 'arn:aws:iam::000000000000:user/rosa-user'
I: Created role 'ManagedOpenShift-ControlPlane-Role' with ARN
'arn:aws:iam::000000000000:role/ManagedOpenShift-ControlPlane-Role'
I: Created role 'ManagedOpenShift-Worker-Role' with ARN
'arn:aws:iam::000000000000:role/ManagedOpenShift-Worker-Role'
I: Created role 'ManagedOpenShift-Support-Role' with ARN
'arn:aws:iam::000000000000:role/ManagedOpenShift-Support-Role'
I: Created role 'ManagedOpenShift-Installer-Role' with ARN
'arn:aws:iam::000000000000:role/ManagedOpenShift-Installer-Role'
I: Created policy with ARN 'arn:aws:iam::000000000000:policy/ManagedOpenShift-openshift-
machine-api-aws-cloud-credentials'
I: Created policy with ARN 'arn:aws:iam::000000000000:policy/ManagedOpenShift-openshift-
cloud-credential-operator-cloud-crede'
I: Created policy with ARN 'arn:aws:iam::000000000000:policy/ManagedOpenShift-openshift-
image-registry-installer-cloud-creden'
I: Created policy with ARN 'arn:aws:iam::000000000000:policy/ManagedOpenShift-openshift-
ingress-operator-cloud-credentials'
I: Created policy with ARN 'arn:aws:iam::000000000000:policy/ManagedOpenShift-openshift-
cluster-csi-drivers-ebs-cloud-credent'
I: To create a cluster with these roles, run the following command:
   rosa create cluster --sts
```

15.3.3.3.2. 创建集群

运行以下命令创建带有所有默认选项的集群：

```
rosa create cluster --cluster-name <cluster-name> --sts --mode auto --yes
```



注意

这也将创建所需的 Operator 角色和 OIDC 供应商。如果要查看集群的所有可用选项，请使用 --help 标志，或 --interactive 用于互动模式。

输入示例

```
$ rosa create cluster --cluster-name my-rosa-cluster --sts --mode auto --yes
```

输出示例

```
I: Creating cluster 'my-rosa-cluster'
I: To view a list of clusters and their status, run 'rosa list clusters'
I: Cluster 'my-rosa-cluster' has been created.
I: Once the cluster is installed you will need to add an Identity Provider before you can login into the cluster. See 'rosa create idp --help' for more information.
I: To determine when your cluster is Ready, run 'rosa describe cluster -c my-rosa-cluster'.
I: To watch your cluster installation logs, run 'rosa logs install -c my-rosa-cluster --watch'.
Name:                  my-rosa-cluster
ID:                    1mlhulb3bo0l54ojd0ji000000000000
External ID:
OpenShift Version:
Channel Group:         stable
DNS:                   my-rosa-cluster.ibhp.p1.openshiftapps.com
AWS Account:           000000000000
API URL:
Console URL:
Region:                us-west-2
Multi-AZ:              false
Nodes:
- Master:              3
- Infra:               2
- Compute:             2
Network:
- Service CIDR:        172.30.0.0/16
- Machine CIDR:        10.0.0.0/16
- Pod CIDR:            10.128.0.0/14
- Host Prefix:         /23
STS Role ARN:          arn:aws:iam::000000000000:role/ManagedOpenShift-Installer-Role
Support Role ARN:      arn:aws:iam::000000000000:role/ManagedOpenShift-Support-Role
Instance IAM Roles:
- Master:              arn:aws:iam::000000000000:role/ManagedOpenShift-ControlPlane-Role
- Worker:              arn:aws:iam::000000000000:role/ManagedOpenShift-Worker-Role
Operator IAM Roles:
- arn:aws:iam::000000000000:role/my-rosa-cluster-openshift-image-registry-installer-cloud-credentials
- arn:aws:iam::000000000000:role/my-rosa-cluster-openshift-ingress-operator-cloud-credentials
- arn:aws:iam::000000000000:role/my-rosa-cluster-openshift-cluster-csi-drivers-ebs-cloud-credentials
- arn:aws:iam::000000000000:role/my-rosa-cluster-openshift-machine-api-aws-cloud-credentials
- arn:aws:iam::000000000000:role/my-rosa-cluster-openshift-cloud-credential-operator-cloud-credential-oper
State:                 waiting (Waiting for OIDC configuration)
Private:               No
Created:               Oct 28 2021 20:28:09 UTC
Details Page:
https://console.redhat.com/openshift/details/s/1wupmiQy45xr1nN000000000000
OIDC Endpoint URL:     https://rh-oidc.s3.us-east-1.amazonaws.com/1mlhulb3bo0l54ojd0ji000000000000
```

15.3.3.3.2.1. 默认配置

默认设置如下：

- 节点：
  - 3 个 control plane 节点
  - 2 个基础架构节点
  - 2 个 worker 节点
  - 没有自动扩展
  - 如需了解更多详细信息，请参阅 ec2 实例 的文档。
- Region ：为 aws CLI 配置
- 网络 IP 范围：
  - Machine CIDR: 10.0.0.0/16
  - Service CIDR: 172.30.0.0/16
  - Pod CIDR: 10.128.0.0/14
- 新的 VPC

- 用于加密的默认 AWS KMS 密钥
- rosa可用的 OpenShift 的最新版本
- 单个可用区
- 公共集群

15.3.3.3.3. 检查安装状态

1. 运行以下命令之一检查集群的状态：
   - 如需状态的详细视图，请运行：

     ```
     rosa describe cluster --cluster <cluster-name>
     ```

   - 对于状态的 abridged 视图，请运行：

     ```
     rosa list clusters
     ```

2. 集群状态将从 "waiting" 改为 "installing" 改为 "ready"。这大约需要 40 分钟。
3. 当状态变为 "ready" 后，您的集群就会被安装。

15.3.3.4. 手动模式

如果要在将角色和策略应用到集群前查看角色和策略，请使用手动方法。此方法需要运行几个额外的命令来创建角色和策略。

本节使用 --interactive 模式。有关本节中字段的描述信息，请参阅 交互模式 的文档。

15.3.3.4.1. 创建帐户角色

1. 如果您是第一次在这个帐户中部署 ROSA 且还没有创建帐户角色时，请创建集群范围的角色和策略，包括 Operator 策略。该命令会在当前目录中为您的帐户所需的角色和策略创建所需的 JSON 文件。它还输出您需要运行的 aws CLI 命令，以创建这些对象。

   运行以下命令来创建所需的文件并输出附加命令：

```
rosa create account-roles --mode manual
```

输出示例

```
I: All policy files saved to the current directory
I: Run the following commands to create the account roles and policies:
aws iam create-role \
--role-name ManagedOpenShift-Worker-Role \
--assume-role-policy-document file://sts_instance_worker_trust_policy.json \
--tags Key=rosa_openshift_version,Value=4.8
Key=rosa_role_prefix,Value=ManagedOpenShift
Key=rosa_role_type,Value=instance_worker
aws iam put-role-policy \
--role-name ManagedOpenShift-Worker-Role \
--policy-name ManagedOpenShift-Worker-Role-Policy \
--policy-document file://sts_instance_worker_permission_policy.json
```

2. 检查您当前目录的内容以查看新文件。使用 aws CLI 创建每个对象。

输出示例

```
$ ls
openshift_cloud_credential_operator_cloud_credential_operator_iam_ro_creds_policy
.json
sts_instance_controlplane_permission_policy.json
openshift_cluster_csi_drivers_ebs_cloud_credentials_policy.json
sts_instance_controlplane_trust_policy.json
openshift_image_registry_installer_cloud_credentials_policy.json
sts_instance_worker_permission_policy.json
openshift_ingress_operator_cloud_credentials_policy.json
sts_instance_worker_trust_policy.json
openshift_machine_api_aws_cloud_credentials_policy.json
sts_support_permission_policy.json
```

```
sts_installer_permission_policy.json
sts_support_trust_policy.json
sts_installer_trust_policy.json
```

3. 可选：打开文件来查看您要创建的内容。例如，打开 sts_installer_permission_policy.json 显示：

输出示例

```
$ cat sts_installer_permission_policy.json
      {
      "Version": "2012-10-17",
      "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "autoscaling:DescribeAutoScalingGroups",
            "ec2:AllocateAddress",
            "ec2:AssociateAddress",
            "ec2:AssociateDhcpOptions",
            "ec2:AssociateRouteTable",
            "ec2:AttachInternetGateway",
            "ec2:AttachNetworkInterface",
            "ec2:AuthorizeSecurityGroupEgress",
            "ec2:AuthorizeSecurityGroupIngress",
            [...]
```

您还可以 参阅 ROSA 集群文档中的关于 IAM 资源中的内容。

4. 运行第 1 步中列出的 aws 命令。如果您位于与您创建的 JSON 文件位于同一个目录中，您可以复制并粘贴。

15.3.3.4.2. 创建集群

1. 在成功执行 aws 命令后，运行以下命令以互动模式开始 ROSA 集群创建：

```
rosa create cluster --interactive --sts
```

有关 字段的描述，请参阅 ROSA 文档。

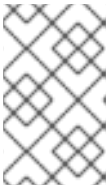
2. *在本教程中，复制并输入以下值：*

```
Cluster name: my-rosa-cluster
OpenShift version: <choose version>
External ID (optional): <leave blank>
Operator roles prefix: <accept default>
Multiple availability zones: No
AWS region: <choose region>
PrivateLink cluster: No
Install into an existing VPC: No
Enable Customer Managed key: No
Compute nodes instance type: m5.xlarge
Enable autoscaling: No
Compute nodes: 2
Machine CIDR: <accept default>
Service CIDR: <accept default>
Pod CIDR: <accept default>
Host prefix: <accept default>
Encrypt etcd data (optional): No
Disable Workload monitoring: No
```

输出示例

```
I: Creating cluster 'my-rosa-cluster'
I: To create this cluster again in the future, you can run:
rosa create cluster --cluster-name my-rosa-cluster --role-arn
arn:aws:iam::000000000000:role/ManagedOpenShift-Installer-Role --support-role-arn
arn:aws:iam::000000000000:role/ManagedOpenShift-Support-Role --master-iam-role
arn:aws:iam::000000000000:role/ManagedOpenShift-ControlPlane-Role --worker-iam-
role arn:aws:iam::000000000000:role/ManagedOpenShift-Worker-Role --operator-roles-
prefix my-rosa-cluster --region us-west-2 --version 4.8.13 --compute-nodes 2 --
machine-cidr 10.0.0.0/16 --service-cidr 172.30.0.0/16 --pod-cidr 10.128.0.0/14 --host-
prefix 23
I: To view a list of clusters and their status, run 'rosa list clusters'
I: Cluster 'my-rosa-cluster' has been created.
I: Once the cluster is installed you will need to add an Identity Provider before you
can login into the cluster. See 'rosa create idp --help' for more information.
Name:                       my-rosa-cluster
ID:                         1t6i760dbum4mqltqh6o000000000000
External ID:
OpenShift Version:
Channel Group:              stable
DNS:                        my-rosa-cluster.abcd.p1.openshiftapps.com
AWS Account:                000000000000
API URL:
Console URL:
Region:                     us-west-2
```

```
Multi-AZ:                 false
Nodes:
- Control plane:          3
- Infra:                  2
- Compute:                2
Network:
- Service CIDR:           172.30.0.0/16
- Machine CIDR:           10.0.0.0/16
- Pod CIDR:               10.128.0.0/14
- Host Prefix:            /23
STS Role ARN:             arn:aws:iam::000000000000:role/ManagedOpenShift-Installer-
Role
Support Role ARN:         arn:aws:iam::000000000000:role/ManagedOpenShift-
Support-Role
Instance IAM Roles:
- Control plane:          arn:aws:iam::000000000000:role/ManagedOpenShift-
ControlPlane-Role
- Worker:                 arn:aws:iam::000000000000:role/ManagedOpenShift-Worker-Role
Operator IAM Roles:
- arn:aws:iam::000000000000:role/my-rosa-cluster-w7i6-openshift-ingress-operator-
cloud-credentials
- arn:aws:iam::000000000000:role/my-rosa-cluster-w7i6-openshift-cluster-csi-drivers-
ebs-cloud-credentials
- arn:aws:iam::000000000000:role/my-rosa-cluster-w7i6-openshift-cloud-network-
config-controller-cloud-cre
- arn:aws:iam::000000000000:role/my-rosa-cluster-openshift-machine-api-aws-cloud-
credentials
- arn:aws:iam::000000000000:role/my-rosa-cluster-openshift-cloud-credential-
operator-cloud-credentia
- arn:aws:iam::000000000000:role/my-rosa-cluster-openshift-image-registry-installer-
cloud-credential
State:                    waiting (Waiting for OIDC configuration)
Private:                  No
Created:                  Jul  1 2022 22:13:50 UTC
Details Page:
https://console.redhat.com/openshift/details/s/2BMQm8xz8Hq5yEN000000000000
OIDC Endpoint URL:        https://rh-oidc.s3.us-east-
1.amazonaws.com/1t6i760dbum4mqltqh6o000000000000
I: Run the following commands to continue the cluster creation:
rosa create operator-roles --cluster my-rosa-cluster
rosa create oidc-provider --cluster my-rosa-cluster
I: To determine when your cluster is Ready, run 'rosa describe cluster -c my-rosa-
cluster'.
I: To watch your cluster installation logs, run 'rosa logs install -c my-rosa-cluster --
watch'.
```



注意

集群状态将保持为"waiting"，直到接下来的两个步骤完成为止。

15.3.3.4.3. 创建 Operator 角色

1. 上面的步骤输出要运行的下一个命令。需要为每个集群创建 这些角色。要创建角色，请运行以下命令：

```
rosa create operator-roles --mode manual --cluster <cluster-name>
```

输出示例

```
I: Run the following commands to create the operator roles:
   aws iam create-role \
      --role-name my-rosa-cluster-openshift-image-registry-installer-cloud-credentials \
      --assume-role-policy-document
file://operator_image_registry_installer_cloud_credentials_policy.json \
      --tags Key=rosa_cluster_id,Value=1mkesci269png3tck000000000000000
Key=rosa_openshift_version,Value=4.8 Key=rosa_role_prefix,Value=
Key=operator_namespace,Value=openshift-image-registry
Key=operator_name,Value=installer-cloud-credentials

   aws iam attach-role-policy \
      --role-name my-rosa-cluster-openshift-image-registry-installer-cloud-credentials \
      --policy-arn arn:aws:iam::000000000000:policy/ManagedOpenShift-openshift-
image-registry-installer-cloud-creden
   [...]
```

2. 运行每个 aws 命令。

15.3.3.4.4. 创建 OIDC 供应商

1. 运行以下命令来创建 OIDC 供应商：

```
rosa create oidc-provider --mode manual --cluster <cluster-name>
```

2. 这将显示您需要运行的 aws 命令。

输出示例

```
I: Run the following commands to create the OIDC provider:
$ aws iam create-open-id-connect-provider \
--url https://rh-oidc.s3.us-east-1.amazonaws.com/1mkesci269png3tckknhh0rfs2da5fj9 \
--client-id-list openshift sts.amazonaws.com \
--thumbprint-list a9d53002e97e00e043244f3d170d000000000000

$ aws iam create-open-id-connect-provider \
--url https://rh-oidc.s3.us-east-1.amazonaws.com/1mkesci269png3tckknhh0rfs2da5fj9 \
--client-id-list openshift sts.amazonaws.com \
--thumbprint-list a9d53002e97e00e043244f3d170d000000000000
```

3. 您的集群现在将继续安装过程。

15.3.3.4.5. 检查安装状态

1. 运行以下命令之一检查集群的状态：

   - 如需状态的详细视图，请运行：

     ```
     rosa describe cluster --cluster <cluster-name>
     ```

   - 对于状态的 abridged 视图，请运行：

     ```
     rosa list clusters
     ```

2. 集群状态将从 "waiting" 改为 "installing" 改为 "ready"。这大约需要 40 分钟。

3. 当状态变为 "ready" 后，您的集群就会被安装。

15.3.3.5. 获取 Red Hat Hybrid Cloud Console URL

- 要获取 Hybrid Cloud Console URL，请运行以下命令：

  ```
  rosa describe cluster -c <cluster-name> | grep Console
  ```

集群现在已被成功部署。下一教程介绍了如何创建 admin 用户来立即使用集群。

15.3.4. 教程：托管 Control Planes 指南

本教程概述了使用托管 control plane (HCP)集群部署 Red Hat OpenShift Service on AWS (ROSA)。

使用 HCP 的 ROSA，您可以将 control plane 与数据平面分离。这是 ROSA 的新部署模型，其中 control plane 托管在红帽拥有的 AWS 帐户中。control plane 不再托管在 AWS 帐户中，从而减少 AWS 基础架构费用。control plane 专用于单个集群，高度可用。如需更多信息，请参阅使用 HCP 的 ROSA 文档。

15.3.4.1. 前提条件

在使用 HCP 集群部署 ROSA 之前，您必须有以下资源：

- VPC - 这是一个自带 VPC 模型，也称为 BYO-VPC。
- OIDC - OIDC 配置和带有该特定配置的 OIDC 供应商。
- ROSA 版本 1.2.31 或更高版本

在本教程中，我们将首先创建这些资源。我们还将设置一些环境变量，以便更轻松地运行命令以使用 HCP 集群创建 ROSA。

15.3.4.1.1. 创建 VPC

1. 首先，请确保您的 AWS CLI (aws)配置为使用带有 HCP 的 ROSA 可用区域。要找出支持哪些区域，请运行以下命令：

```
rosa list regions --hosted-cp
```

2. 创建 VPC。在本教程中，以下 脚本 会为您创建 VPC 及其所需组件。它使用为 aws CLI 配置的区域。

```
#!/bin/bash

set -e
##########
# This script will create the network requirements for a ROSA cluster. This will be
# a public cluster. This creates:
# - VPC
# - Public and private subnets
# - Internet Gateway
# - Relevant route tables
# - NAT Gateway
#
# This will automatically use the region configured for the aws cli
#
##########

VPC_CIDR=10.0.0.0/16
PUBLIC_CIDR_SUBNET=10.0.1.0/24
PRIVATE_CIDR_SUBNET=10.0.0.0/24

# Create VPC
echo -n "Creating VPC..."
VPC_ID=$(aws ec2 create-vpc --cidr-block $VPC_CIDR --query Vpc.VpcId --output text)

# Create tag name
aws ec2 create-tags --resources $VPC_ID --tags Key=Name,Value=$CLUSTER_NAME

# Enable dns hostname
aws ec2 modify-vpc-attribute --vpc-id $VPC_ID --enable-dns-hostnames
echo "done."

# Create Public Subnet
echo -n "Creating public subnet..."
PUBLIC_SUBNET_ID=$(aws ec2 create-subnet --vpc-id $VPC_ID --cidr-block
$PUBLIC_CIDR_SUBNET --query Subnet.SubnetId --output text)

aws ec2 create-tags --resources $PUBLIC_SUBNET_ID --tags
Key=Name,Value=$CLUSTER_NAME-public
echo "done."

# Create private subnet
echo -n "Creating private subnet..."
PRIVATE_SUBNET_ID=$(aws ec2 create-subnet --vpc-id $VPC_ID --cidr-block
$PRIVATE_CIDR_SUBNET --query Subnet.SubnetId --output text)

aws ec2 create-tags --resources $PRIVATE_SUBNET_ID --tags
Key=Name,Value=$CLUSTER_NAME-private
echo "done."

# Create an internet gateway for outbound traffic and attach it to the VPC.
echo -n "Creating internet gateway..."
IGW_ID=$(aws ec2 create-internet-gateway --query InternetGateway.InternetGatewayId
--output text)
echo "done."

aws ec2 create-tags --resources $IGW_ID --tags Key=Name,Value=$CLUSTER_NAME
```

```
aws ec2 attach-internet-gateway --vpc-id $VPC_ID --internet-gateway-id $IGW_ID >
/dev/null 2>&1
echo "Attached IGW to VPC."

# Create a route table for outbound traffic and associate it to the public subnet.
echo -n "Creating route table for public subnet..."
PUBLIC_ROUTE_TABLE_ID=$(aws ec2 create-route-table --vpc-id $VPC_ID --query
RouteTable.RouteTableId --output text)

aws ec2 create-tags --resources $PUBLIC_ROUTE_TABLE_ID --tags
Key=Name,Value=$CLUSTER_NAME
echo "done."

aws ec2 create-route --route-table-id $PUBLIC_ROUTE_TABLE_ID --destination-cidr-
block 0.0.0.0/0 --gateway-id $IGW_ID > /dev/null 2>&1
echo "Created default public route."

aws ec2 associate-route-table --subnet-id $PUBLIC_SUBNET_ID --route-table-id
$PUBLIC_ROUTE_TABLE_ID > /dev/null 2>&1
echo "Public route table associated"

# Create a NAT gateway in the public subnet for outgoing traffic from the private
network.
echo -n "Creating NAT Gateway..."
NAT_IP_ADDRESS=$(aws ec2 allocate-address --domain vpc --query AllocationId --
output text)

NAT_GATEWAY_ID=$(aws ec2 create-nat-gateway --subnet-id $PUBLIC_SUBNET_ID --
allocation-id $NAT_IP_ADDRESS --query NatGateway.NatGatewayId --output text)

aws ec2 create-tags --resources $NAT_IP_ADDRESS --resources $NAT_GATEWAY_ID
--tags Key=Name,Value=$CLUSTER_NAME
sleep 10
echo "done."

# Create a route table for the private subnet to the NAT gateway.
echo -n "Creating a route table for the private subnet to the NAT gateway..."
PRIVATE_ROUTE_TABLE_ID=$(aws ec2 create-route-table --vpc-id $VPC_ID --query
RouteTable.RouteTableId --output text)

aws ec2 create-tags --resources $PRIVATE_ROUTE_TABLE_ID $NAT_IP_ADDRESS --
tags Key=Name,Value=$CLUSTER_NAME-private

aws ec2 create-route --route-table-id $PRIVATE_ROUTE_TABLE_ID --destination-cidr-
block 0.0.0.0/0 --gateway-id $NAT_GATEWAY_ID > /dev/null 2>&1

aws ec2 associate-route-table --subnet-id $PRIVATE_SUBNET_ID --route-table-id
$PRIVATE_ROUTE_TABLE_ID > /dev/null 2>&1

echo "done."

# echo "**********VARIABLE VALUES*********"
# echo "VPC_ID="$VPC_ID
# echo "PUBLIC_SUBNET_ID="$PUBLIC_SUBNET_ID
# echo "PRIVATE_SUBNET_ID="$PRIVATE_SUBNET_ID
```

```
# echo "PUBLIC_ROUTE_TABLE_ID="$PUBLIC_ROUTE_TABLE_ID
# echo "PRIVATE_ROUTE_TABLE_ID="$PRIVATE_ROUTE_TABLE_ID
# echo "NAT_GATEWAY_ID="$NAT_GATEWAY_ID
# echo "IGW_ID="$IGW_ID
# echo "NAT_IP_ADDRESS="$NAT_IP_ADDRESS

echo "Setup complete."
echo ""
echo "To make the cluster create commands easier, please run the following
commands to set the environment variables:"
echo "export PUBLIC_SUBNET_ID=$PUBLIC_SUBNET_ID"
echo "export PRIVATE_SUBNET_ID=$PRIVATE_SUBNET_ID"
```

有关 VPC 要求的更多信息，请参阅 VPC 文档。

3. 以上脚本会输出两个命令。将命令设置为环境变量，以便更轻松地运行 create cluster 命令。从输出中复制它们并运行它们，如下所示：

```
export PUBLIC_SUBNET_ID=<public subnet id here>
export PRIVATE_SUBNET_ID=<private subnet id here>
```

4. 运行以下命令确认已设置了环境变量：

```
echo "Public Subnet: $PUBLIC_SUBNET_ID"; echo "Private Subnet:
$PRIVATE_SUBNET_ID"
```

输出示例

```
Public Subnet: subnet-0faeeeb0000000000
Private Subnet: subnet-011fe340000000000
```

15.3.4.1.2. 创建 OIDC 配置

在本教程中，在创建 OIDC 配置时，我们将使用自动模式。我们还将 OIDC ID 存储为环境变量，以备以后使用。该命令使用 ROSA CLI 创建集群的唯一 OIDC 配置。

- 要为本教程创建 OIDC 配置，请运行以下命令：

```
export OIDC_ID=$(rosa create oidc-config --mode auto --managed --yes -o json | jq -r
'.id')
```

15.3.4.1.3. 创建额外的环境变量

- 运行以下命令来设置一些环境变量，以便更轻松地运行命令来使用 HCP 集群创建 ROSA：

```
export CLUSTER_NAME=<enter cluster name>
export REGION=<region VPC was created in>
```

提示

运行 rosa whoami 来查找 VPC 区域。

15.3.4.2. 创建集群

如果您是 第一次在这个 帐户中部署 ROSA 且还没有创建帐户角色时，请创建集群范围的角色和策略，包括 Operator 策略。由于 ROSA 使用 AWS 安全令牌服务(STS)，因此此步骤会创建 ROSA 与您的帐户交互所需的 AWS IAM 角色和策略。

1. 运行以下命令来创建集群范围的角色：

```
rosa create account-roles --mode auto --yes
```

2. 运行以下命令来创建集群：

```
rosa create cluster --cluster-name $CLUSTER_NAME \
  --subnet-ids ${PUBLIC_SUBNET_ID},${PRIVATE_SUBNET_ID} \
  --hosted-cp \
  --region $REGION \
  --oidc-config-id $OIDC_ID \
  --sts --mode auto --yes
```

集群就绪并在大约 10 分钟后完全可用。集群将在您选择的区域的三个 AWS 可用区中有一个 control plane，并在 AWS 帐户中创建两个 worker 节点。

15.3.4.3. 检查安装状态

1. 运行以下命令之一检查集群的状态：

- 如需集群状态的详细视图，请运行：

```
rosa describe cluster --cluster $CLUSTER_NAME
```

- 对于集群状态的 abridged 视图，请运行：

```
rosa list clusters
```

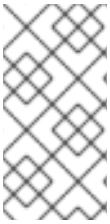
- 要在进行时监控日志，请运行：

```
rosa logs install --cluster $CLUSTER_NAME --watch
```

2. 当状态变为 "ready" 后，您的集群就会被安装。可能需要过几分钟后，worker 节点才会上线。

15.3.5. 教程：简单 UI 指南

本页概述了使用用户界面(UI)部署 ROSA 集群的命令的最小列表。



注意

虽然这个简单部署适用于教程设置，但生产中使用的集群应该使用更详细的方法进行部署。

15.3.5.1. 前提条件

- 您已完成了设置指南中的先决条件。

15.3.5.2. 创建帐户角色

对于每个 AWS 帐户和 y-stream OpenShift 版本，运行以下命令：

```
rosa create account-roles --mode auto --yes
```

15.3.5.3. 创建 Red Hat OpenShift Cluster Manager 角色

1. *运行以下命令，为每个 AWS 帐户创建一个 OpenShift Cluster Manager 角色：*

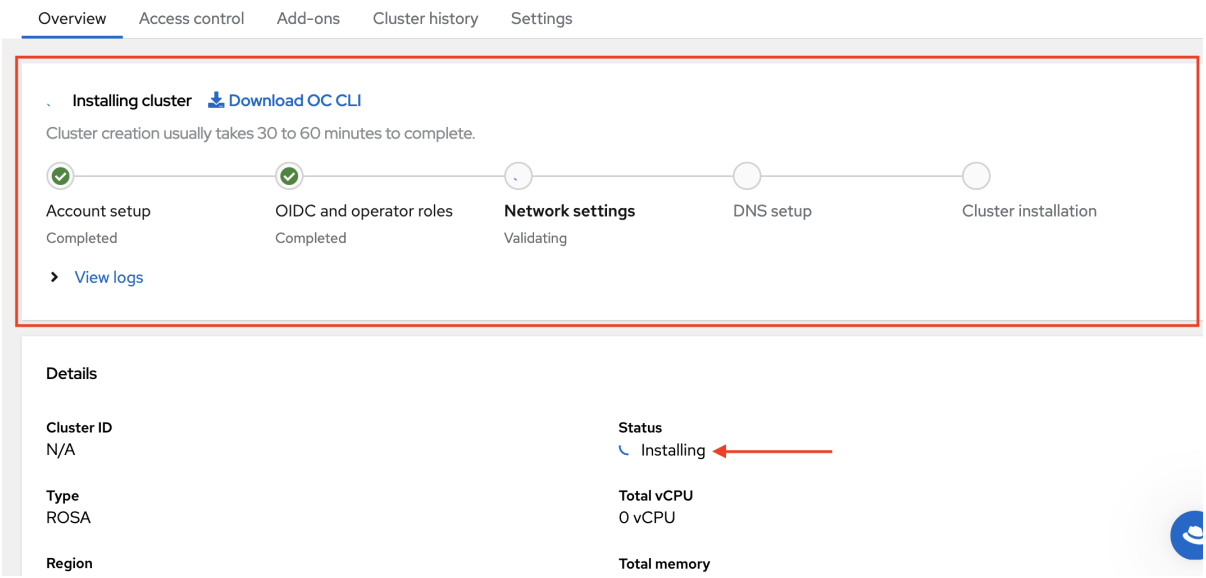
```
rosa create ocm-role --mode auto --admin --yes
```

2. *运行以下命令，为每个 AWS 帐户创建一个 OpenShift Cluster Manager 用户角色：*

```
rosa create user-role --mode auto --yes
```

3. *使用 OpenShift Cluster Manager 选择 AWS 帐户、集群选项并开始部署。*

4. *OpenShift Cluster Manager UI 显示集群状态。*



15.3.6. 教程：详细的 UI 指南

本教程概述了使用 Red Hat OpenShift Cluster Manager 用户界面(UI)部署 Red Hat OpenShift Service on AWS (ROSA)集群的详细步骤。

15.3.6.1. 部署工作流

整个部署工作流遵循以下步骤：

1.

1. 创建集群范围的角色和策略。

2. 将您的 AWS 帐户与您的红帽帐户相关联。

   a. 创建并链接 Red Hat OpenShift Cluster Manager 角色。

   b. 创建并链接用户角色。

3. 创建集群。

只需 首次 部署到 AWS 帐户时，只需执行第 1 步。第 2 步只需要在 第一次使用 UI 时执行。对于同一 y-stream 版本的连续集群，您只需要创建集群。

15.3.6.2. 创建集群范围的角色

注意

如果您已有之前部署中的帐户角色，请跳过这一步。选择关联的 AWS 帐户后，UI 会检测您的现有角色。

如果您是 第一次在这个 帐户中部署 ROSA 且还没有创建帐户角色时，请创建集群范围的角色和策略，包括 Operator 策略。

- 在终端中，运行以下命令来创建集群范围的角色：

```
$ rosa create account-roles --mode auto --yes
```

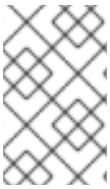
输出示例

```
I: Creating roles using 'arn:aws:iam::000000000000:user/rosa-user'
I: Created role 'ManagedOpenShift-ControlPlane-Role' with ARN
'arn:aws:iam::000000000000:role/ManagedOpenShift-ControlPlane-Role'
I: Created role 'ManagedOpenShift-Worker-Role' with ARN
'arn:aws:iam::000000000000:role/ManagedOpenShift-Worker-Role'
```

```
I: Created role 'ManagedOpenShift-Support-Role' with ARN
'arn:aws:iam::000000000000:role/ManagedOpenShift-Support-Role'
I: Created role 'ManagedOpenShift-Installer-Role' with ARN
'arn:aws:iam::000000000000:role/ManagedOpenShift-Installer-Role'
I: Created policy with ARN 'arn:aws:iam::000000000000:policy/ManagedOpenShift-
openshift-machine-api-aws-cloud-credentials'
I: Created policy with ARN 'arn:aws:iam::000000000000:policy/ManagedOpenShift-
openshift-cloud-credential-operator-cloud-crede'
I: Created policy with ARN 'arn:aws:iam::000000000000:policy/ManagedOpenShift-
openshift-image-registry-installer-cloud-creden'
I: Created policy with ARN 'arn:aws:iam::000000000000:policy/ManagedOpenShift-
openshift-ingress-operator-cloud-credentials'
I: Created policy with ARN 'arn:aws:iam::000000000000:policy/ManagedOpenShift-
openshift-cluster-csi-drivers-ebs-cloud-credent'
I: To create a cluster with these roles, run the following command:
rosa create cluster --sts
```

15.3.6.3. 将您的 AWS 帐户与您的红帽帐户相关联

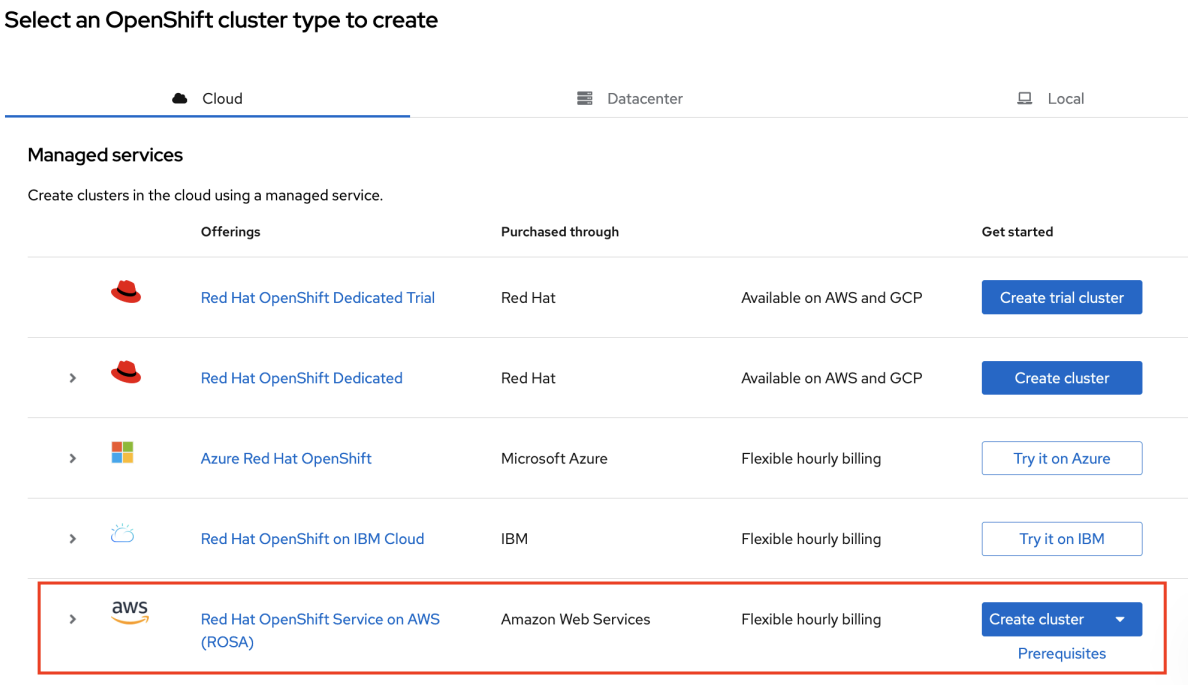
此步骤告知 OpenShift Cluster Manager 部署 ROSA 时要使用的 AWS 帐户。



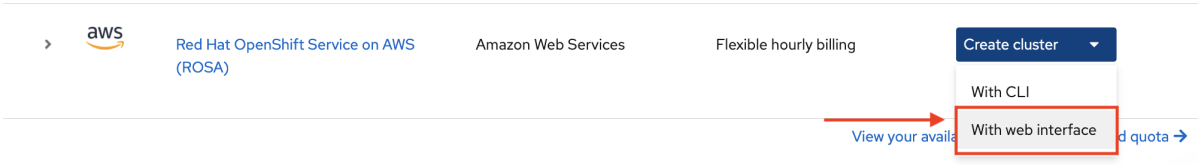
注意

如果您已经关联了 AWS 帐户，请跳过这一步。

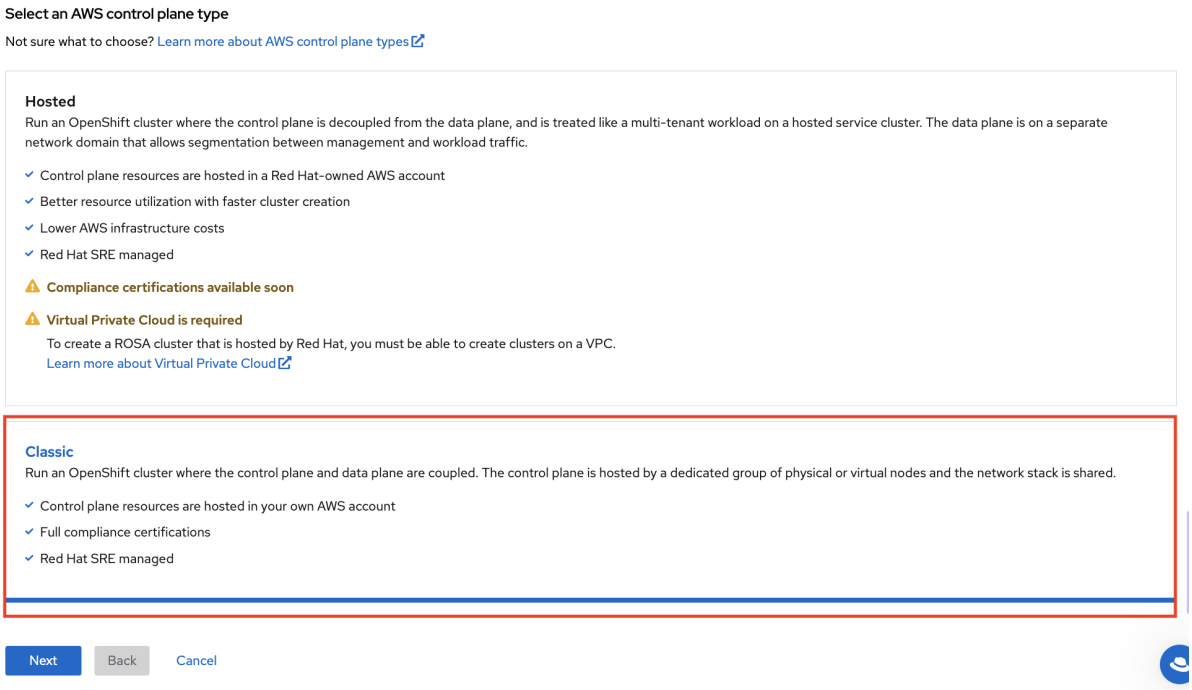
1. 访问 OpenShift Cluster Manager 并登录您的红帽帐户，打开 Red Hat Hybrid Cloud 控制台。
2. 点 Create Cluster。
3. 向下滚动到 Red Hat OpenShift Service on AWS (ROSA)行，点 Create Cluster。



4. *此时会出现下拉菜单。单击 **With Web 界面**。*



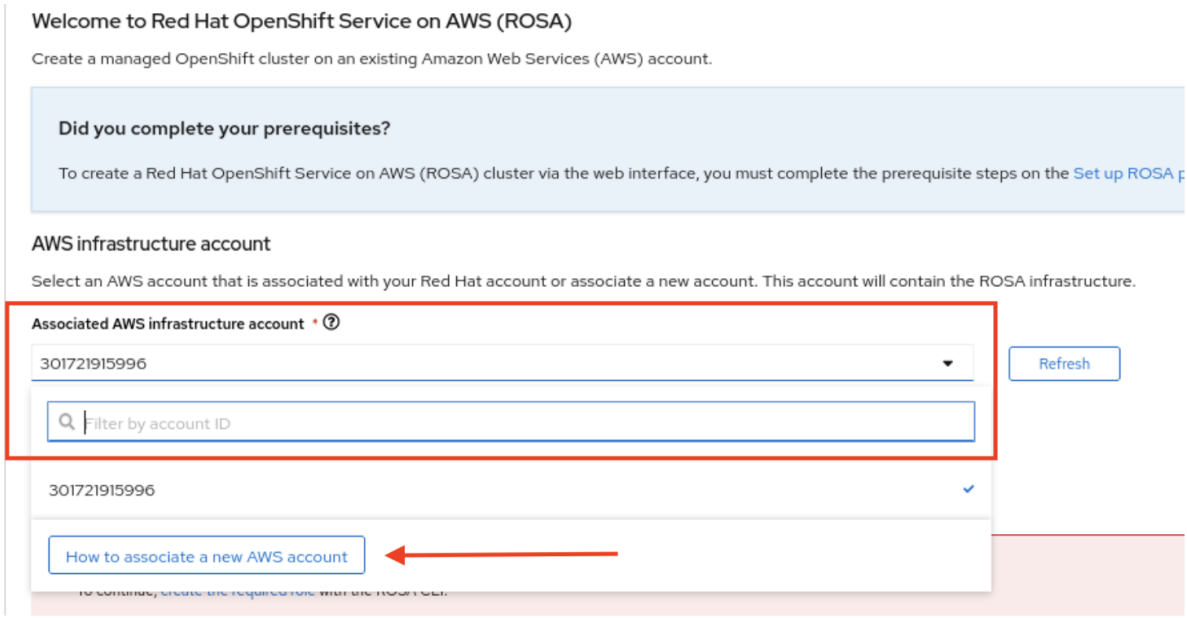
5. *在"选择 **AWS control plane** 类型"下，选择 **Classic**。然后单击"下一步"。*



6. *单击 **关联 AWS 基础架构帐户** 下的 **dropbox**。如果您还没有关联任何 **AWS** 帐户，则 **dropbox** 可能为空。*

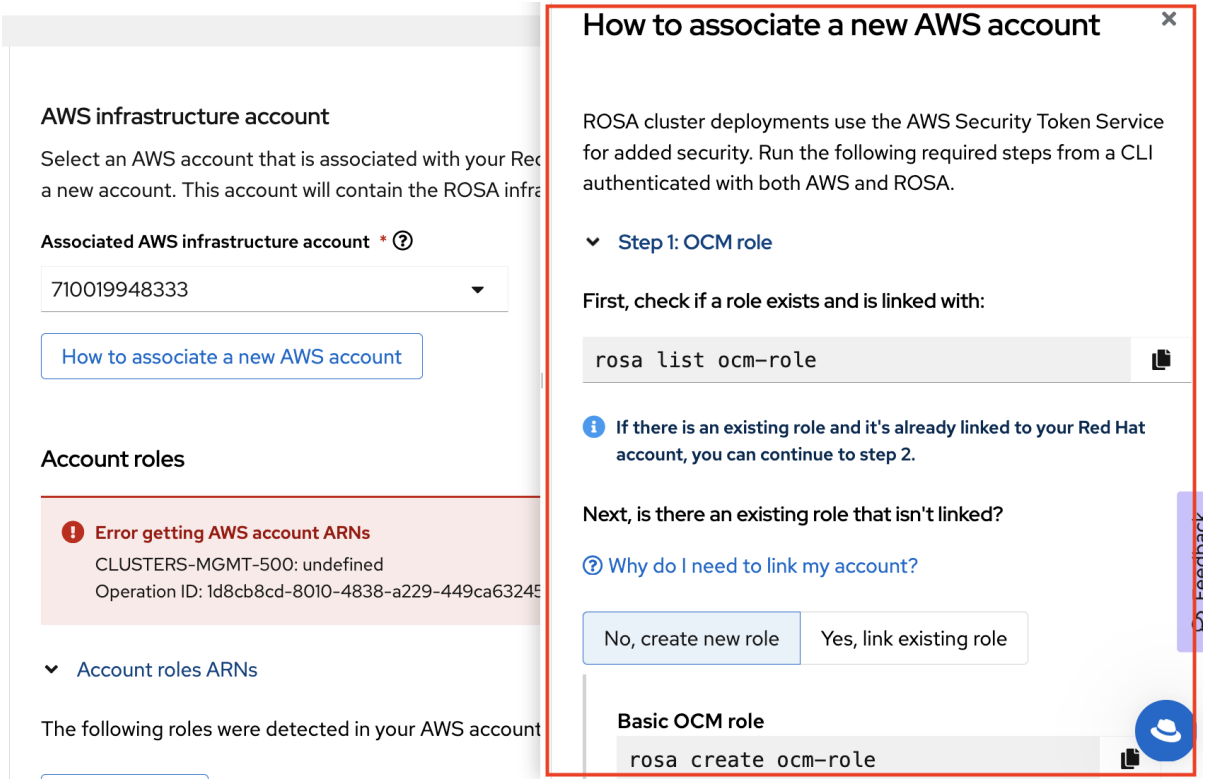
7.

点 How to associate a new AWS account。



8.

此时会出现一个侧边栏，其中包含关联新 AWS 帐户的说明。



15.3.6.4. 创建并关联 OpenShift Cluster Manager 角色

1.

运行以下命令，以查看 OpenShift Cluster Manager 角色是否存在：

```
$ rosa list ocm-role
```

2. UI 显示创建具有不同权限级别的 OpenShift Cluster Manager 角色的命令：

- 基本 OpenShift Cluster Manager 角色：允许 OpenShift Cluster Manager 具有帐户的只读访问权限，以便在创建集群时是否存在 ROSA 所需的角色和策略。您需要使用 CLI 手动创建所需的角色、策略和 OIDC 供应商。

- admin OpenShift Cluster Manager 角色：授予 OpenShift Cluster Manager 额外权限，以便为 ROSA 创建所需的角色、策略和 OIDC 供应商。使用这个方法可以更快地部署 ROSA 集群，因为 OpenShift Cluster Manager 能够为您创建所需资源。

  要了解更多有关这些角色的信息，请参阅文档中的 OpenShift Cluster Manager 角色和权限 部分。

  在本教程中，使用 Admin OpenShift Cluster Manager 角色 进行最简单的和快速的方法。

3. 复制命令，从侧边栏中创建 Admin OpenShift Cluster Manager 角色，或切换到您的终端并输入以下命令：

```
$ rosa create ocm-role --mode auto --admin --yes
```

此命令会创建 OpenShift Cluster Manager 角色，并将其与您的红帽帐户相关联。

输出示例

```
I: Creating ocm role
I: Creating role using 'arn:aws:iam::000000000000:user/rosa-user'
I: Created role 'ManagedOpenShift-OCM-Role-12561000' with ARN
'arn:aws:iam::000000000000:role/ManagedOpenShift-OCM-Role-12561000'
I: Linking OCM role
I: Successfully linked role-arn 'arn:aws:iam::000000000000:role/ManagedOpenShift-
OCM-Role-12561000' with organization account '1MpZfntsZeUdjWHg7XRgP000000'
```

4. 单击 **Step 2: User role**。

15.3.6.4.1. 其他 OpenShift Cluster Manager 角色创建选项

- 手动模式：如果您希望自己运行 AWS CLI 命令，您可以将模式定义为 **manual** 而不是 **auto**。CLI 将输出 AWS 命令以及相关的 JSON 文件在当前目录中创建。

  使用以下命令，以手动模式创建 OpenShift Cluster Manager 角色：

  ```
  $ rosa create ocm-role --mode manual --admin --yes
  ```

- 基本 OpenShift Cluster Manager 角色：如果您希望 OpenShift Cluster Manager 对帐户具有只读访问权限，请创建一个基本的 OpenShift Cluster Manager 角色。然后，您需要使用 CLI 手动创建所需的角色、策略和 OIDC 供应商。

  使用以下命令来创建基本 OpenShift Cluster Manager 角色：

  ```
  $ rosa create ocm-role --mode auto --yes
  ```

15.3.6.5. 创建 OpenShift Cluster Manager 用户角色

如 用户角色文档中所述，需要创建用户角色，以便 ROSA 可以验证 AWS 身份。此角色没有权限，它仅用于在安装程序帐户和 OpenShift Cluster Manager 角色资源之间建立信任关系。

1. 运行以下命令，检查用户角色是否已存在：

   ```
   $ rosa list user-role
   ```

2. 运行以下命令来创建用户角色并将其链接到您的红帽帐户：

   ```
   $ rosa create user-role --mode auto --yes
   ```

   输出示例

   ```
   I: Creating User role
   I: Creating ocm user role using 'arn:aws:iam::000000000000:user/rosa-user'
   I: Created role 'ManagedOpenShift-User-rosa-user-Role' with ARN
   ```

```
'arn:aws:iam::000000000000:role/ManagedOpenShift-User-rosa-user-Role'
I: Linking User role
I: Successfully linked role ARN 'arn:aws:iam::000000000000:role/ManagedOpenShift-User-rosa-user-Role' with account '1rbOQez0z5j1YolInhcXY000000'
```

注意

如上所述，如果您更喜欢自己运行 AWS CLI 命令，您可以定义 **--mode manual**。CLI 输出 AWS 命令和相关 JSON 文件在当前目录中创建。确保链接角色。

3. 点 **Step 3: Account roles**。

15.3.6.6. 创建帐户角色

1. 运行以下命令来创建帐户角色：

```
$ rosa create account-roles --mode auto
```

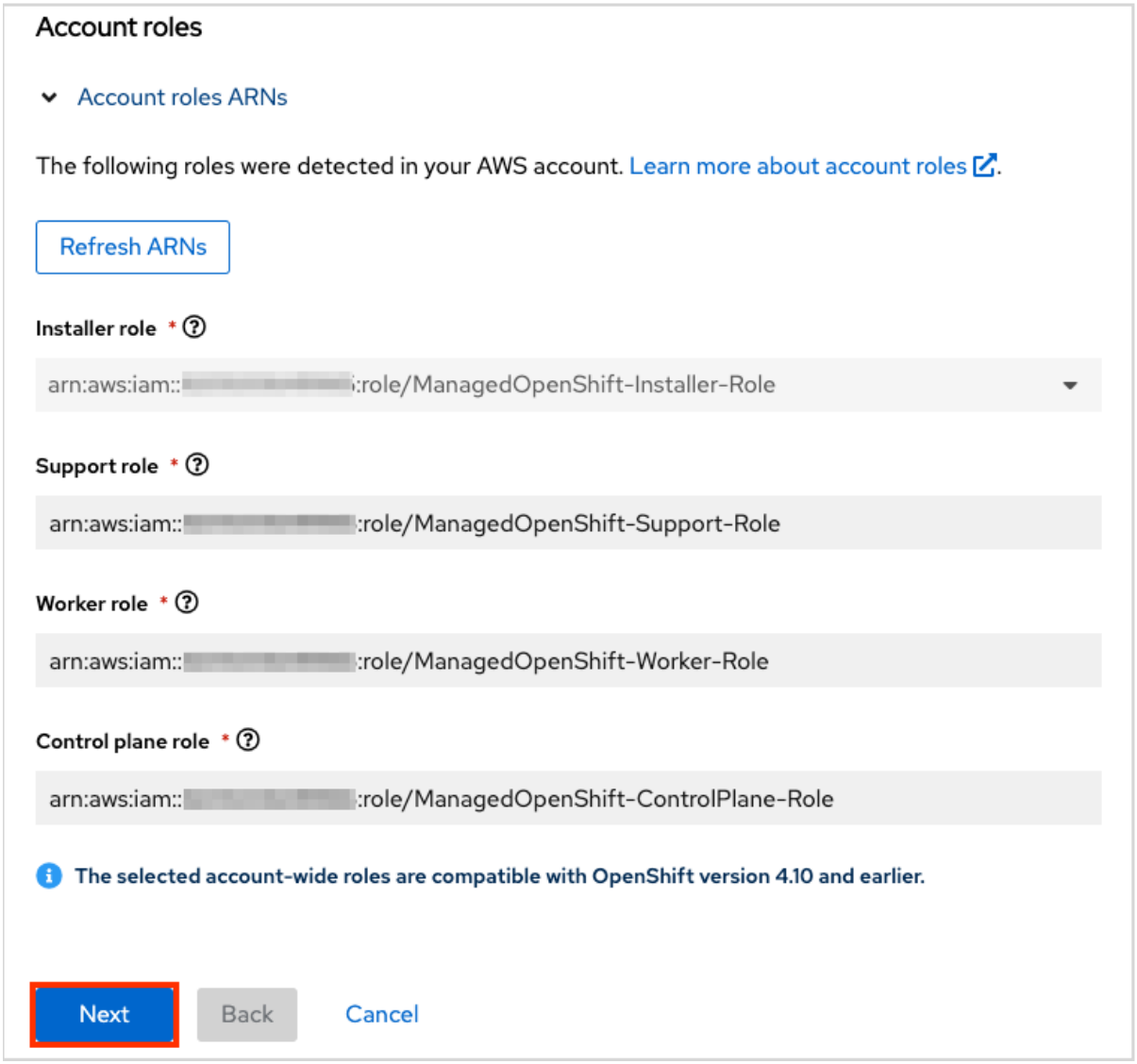
2. 单击 **OK** 以关闭边栏。

15.3.6.7. 确认帐户关联的成功

1. 现在，您应该在 关联 AWS 基础架构帐户下拉菜单中选择 AWS 帐户。如果您看到您的帐户，帐户关联成功。

2. 选择帐户。

3. 您将看到以下填充的帐户角色 ARN。



4. *点击 **Next**。*

15.3.6.8. 创建集群

1. *在本教程中，以下选择如下：*

集群设置

- *Cluster name: & lt;pick a name\>*
- *Version: & lt;select latest version\>*
- *region: & lt;select region\>*

- *可用性： 单一区*
- *启用用户工作负载监控： 保留检查*
- *启用额外的 etcd 加密： 保留未选中*
- *使用客户密钥加密持久性卷： 保留未选中*

2. *点击 Next。*
3. *为机器池保留默认设置：*

默认机器池设置

- *Compute 节点实例类型： m5.xlarge - 4 vCPU 16 GiB RAM*
- *enable autoscaling: unchecked*
- *Compute 节点数： 2*
- *将节点标签留空*

4. *点击 Next。*

15.3.6.8.1. 网络

1. *保留配置的所有默认值。*
2. *点击 Next。*

3. 保留 CIDR 范围的所有默认值。

4. 点击 Next。

15.3.6.8.2. 集群角色和策略

在本教程中，请保留 Auto selected。它将使集群部署过程变得更加简单、更快。

注意

如果您之前选择了 基本 OpenShift Cluster Manager 角色，则只能使用手动模式。您必须手动创建 Operator 角色和 OIDC 供应商。完成"集群更新"部分并启动集群创建后，请参见下面的"基本 OpenShift Cluster Manager 角色"部分。

15.3.6.8.3. 集群更新

- 本节中将所有选项保留为默认值。

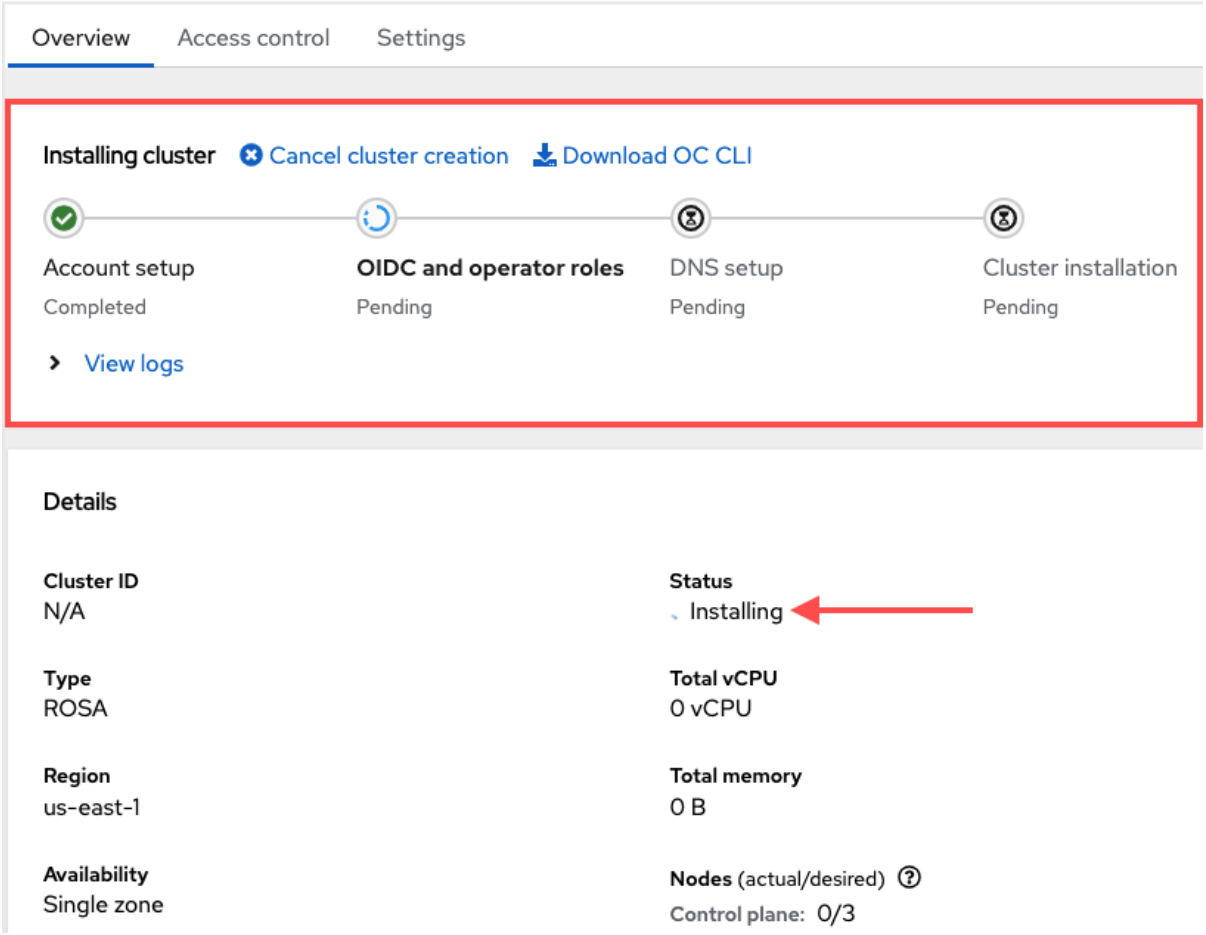
15.3.6.8.4. 检查并创建集群

1. 查看集群配置的内容。

2. 点 Create cluster。

15.3.6.8.5. 监控安装进度

- 保留在当前页面以监控安装进度。它应该需要大约 40 分钟。



15.3.6.9. 基本 OpenShift Cluster Manager 角色

注意

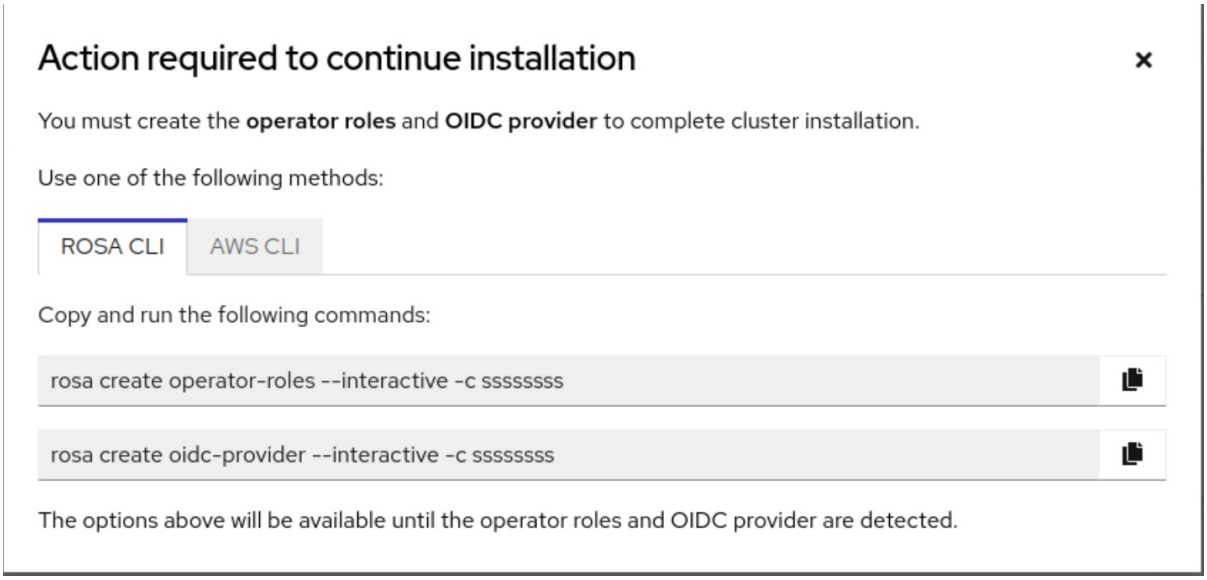
如果您根据上述指示创建了 Admin OpenShift Cluster Manager 角色，请忽略 此整个部分。OpenShift Cluster Manager 将为您创建资源。

如果您之前创建了 基本 OpenShift Cluster Manager 角色，则需要在集群安装可以继续前手动创建两个元素：

- *Operator 角色*
- *OIDC 供应商*

15.3.6.9.1. 创建 Operator 角色

1. *弹出窗口将显示要运行的命令。*



2. *在终端中的窗口中运行命令以启动交互模式。或者，为了简单起见，请运行以下命令来创建 Operator 角色：*

```
$ rosa create operator-roles --mode auto --cluster <cluster-name> --yes
```

输出示例

```
I: Creating roles using 'arn:aws:iam::000000000000:user/rosauser'
I: Created role 'rosacluster-b736-openshift-ingress-operator-cloud-credentials' with ARN 'arn:aws:iam::000000000000:role/rosacluster-b736-openshift-ingress-operator-cloud-credentials'
I: Created role 'rosacluster-b736-openshift-cluster-csi-drivers-ebs-cloud-credent' with ARN 'arn:aws:iam::000000000000:role/rosacluster-b736-openshift-cluster-csi-drivers-ebs-cloud-credent'
I: Created role 'rosacluster-b736-openshift-cloud-network-config-controller-cloud' with ARN 'arn:aws:iam::000000000000:role/rosacluster-b736-openshift-cloud-network-config-controller-cloud'
I: Created role 'rosacluster-b736-openshift-machine-api-aws-cloud-credentials' with ARN 'arn:aws:iam::000000000000:role/rosacluster-b736-openshift-machine-api-aws-cloud-credentials'
I: Created role 'rosacluster-b736-openshift-cloud-credential-operator-cloud-crede' with ARN 'arn:aws:iam::000000000000:role/rosacluster-b736-openshift-cloud-credential-operator-cloud-crede'
I: Created role 'rosacluster-b736-openshift-image-registry-installer-cloud-creden' with ARN 'arn:aws:iam::000000000000:role/rosacluster-b736-openshift-image-registry-installer-cloud-creden'
```

15.3.6.9.2. 创建 OIDC 供应商

- 在终端中，运行以下命令来创建 OIDC 供应商：

```
$ rosa create oidc-provider --mode auto --cluster <cluster-name> --yes
```

输出示例

```
I: Creating OIDC provider using 'arn:aws:iam::000000000000:user/rosauser'
I: Created OIDC provider with ARN 'arn:aws:iam::000000000000:oidc-provider/rh-oidc.s3.us-east-1.amazonaws.com/1tt4kvrr2kha2rgs8gjfvf0000000000'
```

15.4. 教程：创建管理员用户

通过创建管理(admin)用户，您可以快速访问集群。按照以下步骤创建管理员用户。

注意

管理员用户在此教程设置中可以正常工作。对于实际部署，请使用 正式的身份提供程序 访问集群并授予用户 admin 特权。

1. 运行以下命令来创建 admin 用户：

```
rosa create admin --cluster=<cluster-name>
```

输出示例

```
W: It is recommended to add an identity provider to login to this cluster. See 'rosa create idp --help' for more information.
I: Admin account has been added to cluster 'my-rosa-cluster'. It may take up to a minute for the account to become active.
I: To login, run the following command:
oc login https://api.my-rosa-cluster.abcd.p1.openshiftapps.com:6443 \
--username cluster-admin \
--password FWGYL-2mkJI-00000-00000
```

2. *复制上一步中返回到的登录命令，并将它粘贴到终端中。这将使用 CLI 登录集群，以便您可以开始使用集群。*

```
$ oc login https://api.my-rosa-cluster.abcd.p1.openshiftapps.com:6443 \
>    --username cluster-admin \
>    --password FWGYL-2mkJI-00000-00000
```

输出示例

```
Login successful.

You have access to 79 projects, the list has been suppressed. You can list all projects with ' projects'

Using project "default".
```

3. *要检查您是否已以 admin 用户身份登录，请运行以下命令之一：*

- *选项 1：*

```
$ oc whoami
```

输出示例

```
cluster-admin
```

- *选项 2：*

```
oc get all -n openshift-apiserver
```

只有 admin 用户可以在不出错的情况下运行此命令。

4. 现在，您可以使用集群作为 admin 用户，这将满足本教程的需要。对于实际部署，强烈建议设置身份提供程序，具体参见 下一教程。

15.5. 教程：设置身份提供程序

要登录到集群，请设置身份提供程序(IDP)。本教程使用 GitHub 作为示例 IDP。请参阅 ROSA 支持的 IDP 的完整列表。

- 要查看所有 IDP 选项，请运行以下命令：

```
rosa create idp --help
```

15.5.1. 使用 GitHub 设置 IDP

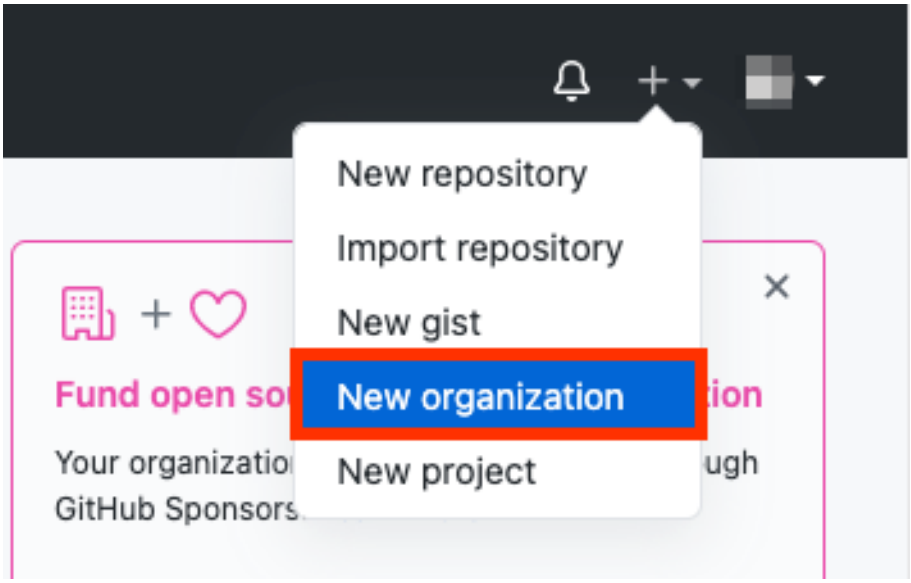
1. 登录您的 GitHub 帐户。

2. 创建一个您是管理员的新 GitHub 组织。

提示

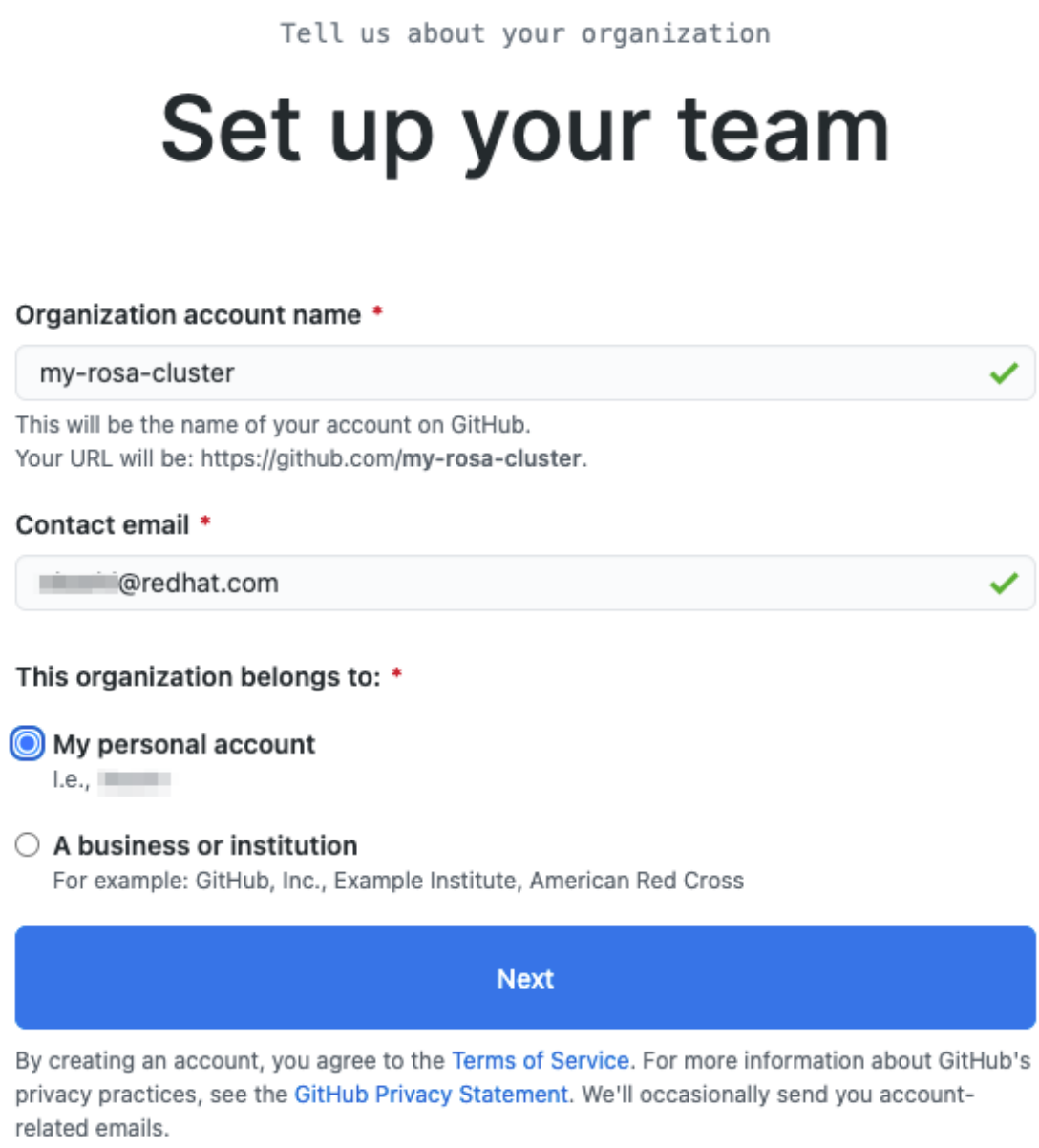
如果您已是现有机构的管理员，并希望使用该机构，请跳至第 9 步。

单击 + 图标，然后单击 New Organization。



3. *为您的情况选择最适用的计划，或者点击* ***Join for free。***

4. *输入机构帐户名称、电子邮件，以及它是个人或商业账户。然后，单击* ***Next。***



5. *可选：添加其他用户的 GitHub ID 来授予对 ROSA 集群的额外访问权限。您还可以稍后添加它们。*

6. *点 Complete Setup。*

7. *可选：在以下页面中输入请求的信息。*

8. *点 Submit。*

9. *返回到终端，输入以下命令设置 GitHub IDP：*

```
rosa create idp --cluster=<cluster name> --interactive
```

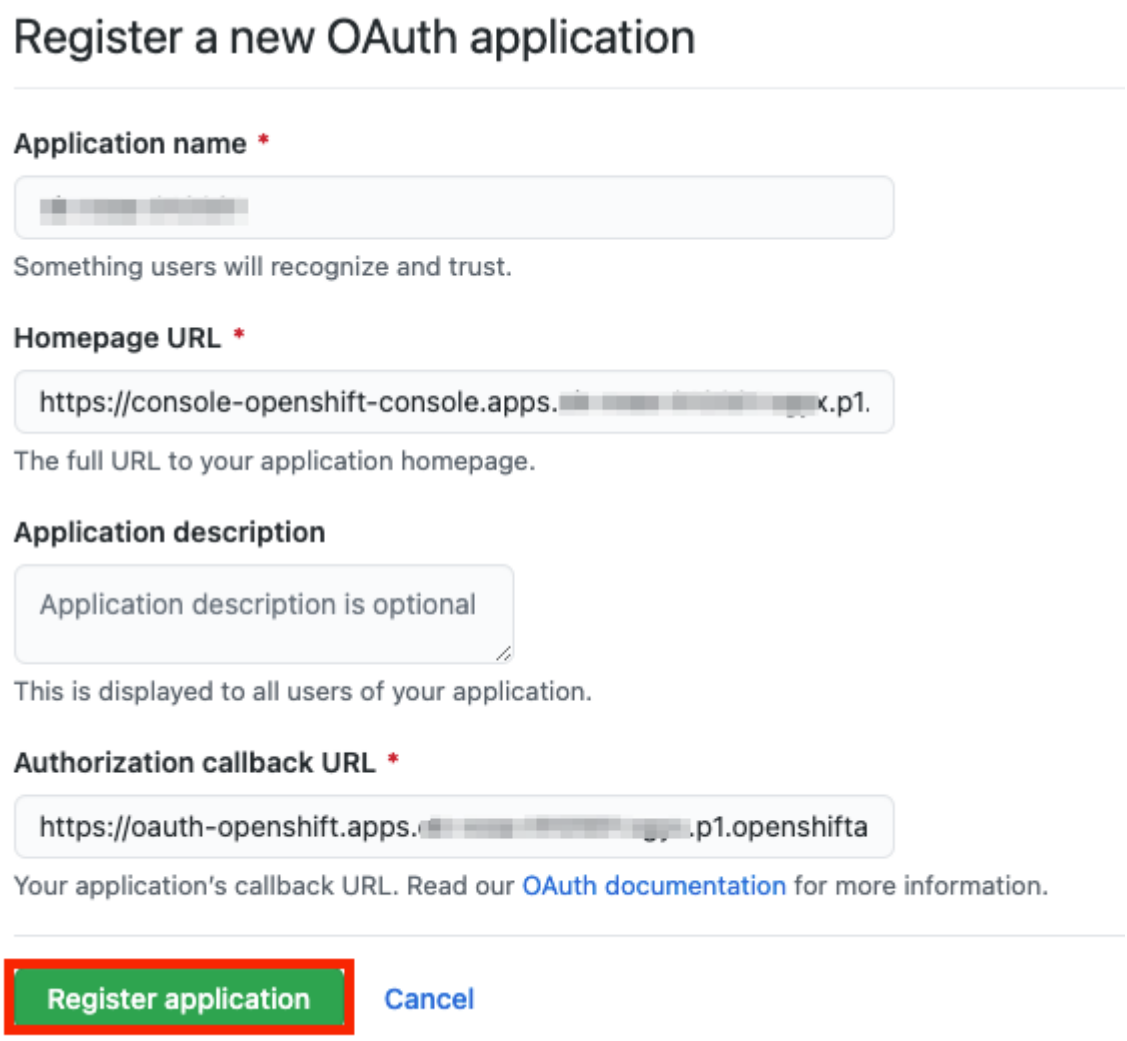
10. 输入以下值：

```
Type of identity provider: github
Identity Provider Name: <IDP-name>
Restrict to members of: organizations
GitHub organizations: <organization-account-name>
```

11. CLI 将为您提供链接。将链接复制并粘贴到浏览器中，然后按 Enter。这将填写为 OAuth 注册此应用所需的信息。您不需要修改任何信息。

```
I: Interactive mode enabled.
Any optional fields can be left empty and a default will be selected.
? Type of identity provider: github
? Identity provider name: rosa-github
? Restrict to members of: organizations
? GitHub organizations: my-rosa-cluster
? To use GitHub as an identity provider, you must first register the application:
  - Open the following URL:
    https://github.com/organizations/my-rosa-cluster/settings/applications/new?oauth_application%5Bcallback_url%5D=h
uth2callback%2Frosa-github&oauth_application%5Bname%5D=                    &oauth_application%5Burl%5D=https%3A%2F%2Fconso
  - Click on 'Register application'
? Client ID: [? for help]
```

12. 点击 Register application。

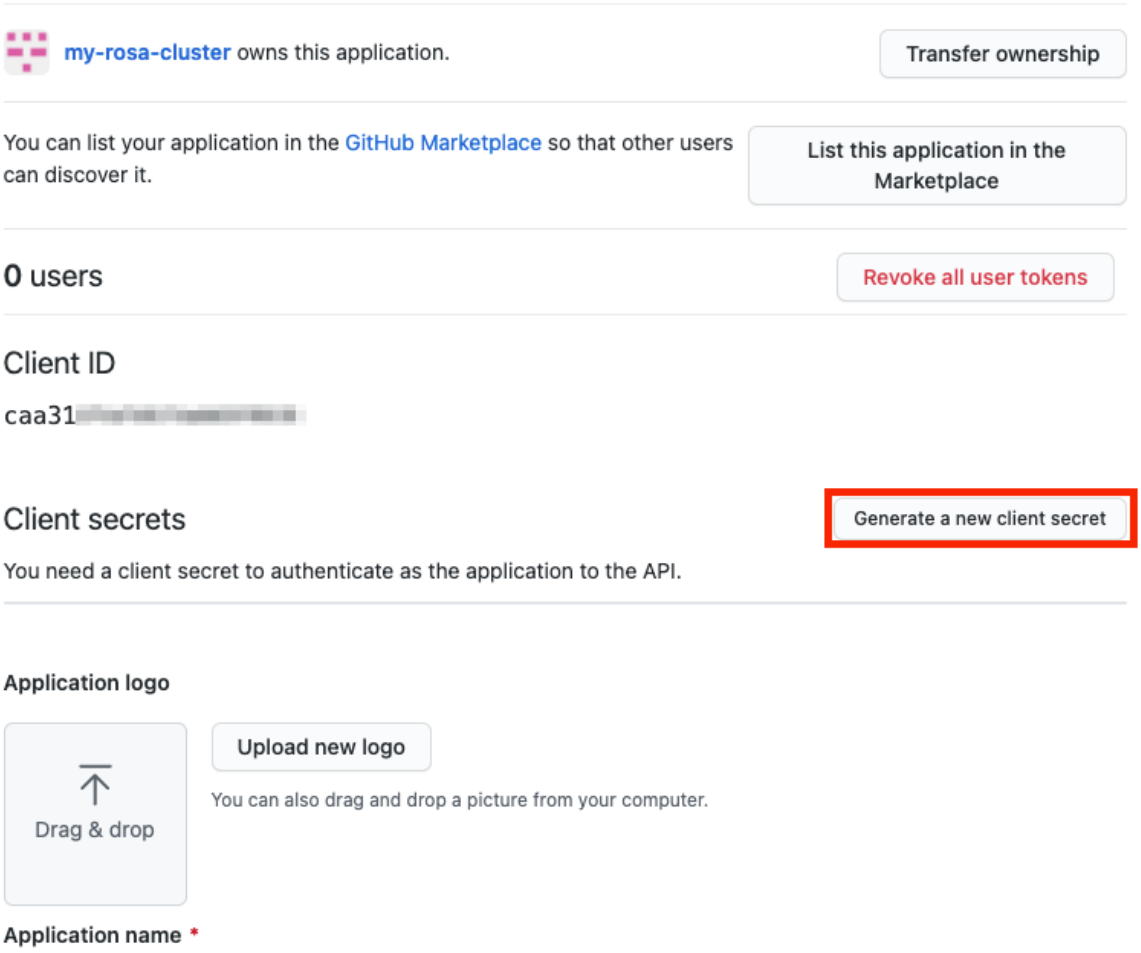


13. *下一页显示 客户端 ID。复制 ID 并将其粘贴到请求 客户端 ID 的终端。*

注意

不要关闭选项卡。

14. ***CLI 将要求提供 客户端 Secret。返回到浏览器，再单击 Generate a new client secret。***



15. *为您生成了一个 secret。复制您的 secret，因为它永远不会再次可见。*

16. *将您的 secret 粘贴到终端中，然后按 Enter。*

17. *将 GitHub Enterprise 主机名 留空。*

18. *选择 声明。*

19. *等待大约 1 分钟，让 IDP 创建，并将配置放在集群中。*

```
I: Interactive mode enabled.
Any optional fields can be left empty and a default will be selected.
? Type of identity provider: github
? Identity provider name: rosa-github
? Restrict to members of: organizations
? GitHub organizations: my-rosa-cluster
? To use GitHub as an identity provider, you must first register the application:
  - Open the following URL:
    https://github.com/organizations/my-rosa-cluster/settings/applications/new?oauth_application%5Bcallback_url%5D=
https%3A%2F%2Foauth-openshift.apps.[redacted].p1.openshiftapps.com%2Foauth2callback%2Frosa-github&oauth_ap
plication%5Bname%5D=[redacted]L&oauth_application%5Burl%5D=https%3A%2F%2Fconsole-openshift-console.apps.[redacted]
[redacted].p1.openshiftapps.com
  - Click on 'Register application'
? Client ID: caa31[redacted]
? Client Secret: [? for help] ****************************************
? GitHub Enterprise Hostname (optional):
? Mapping method: claim
I: Configuring IDP for cluster '[redacted]'
I: Identity Provider 'rosa-github' has been created.
   It will take up to 1 minute for this configuration to be enabled.
   To add cluster administrators, see 'rosa create user --help'.
   To login into the console, open https://console-openshift-console.apps.[redacted].p1.openshiftapps.com
and click on rosa-github.
```

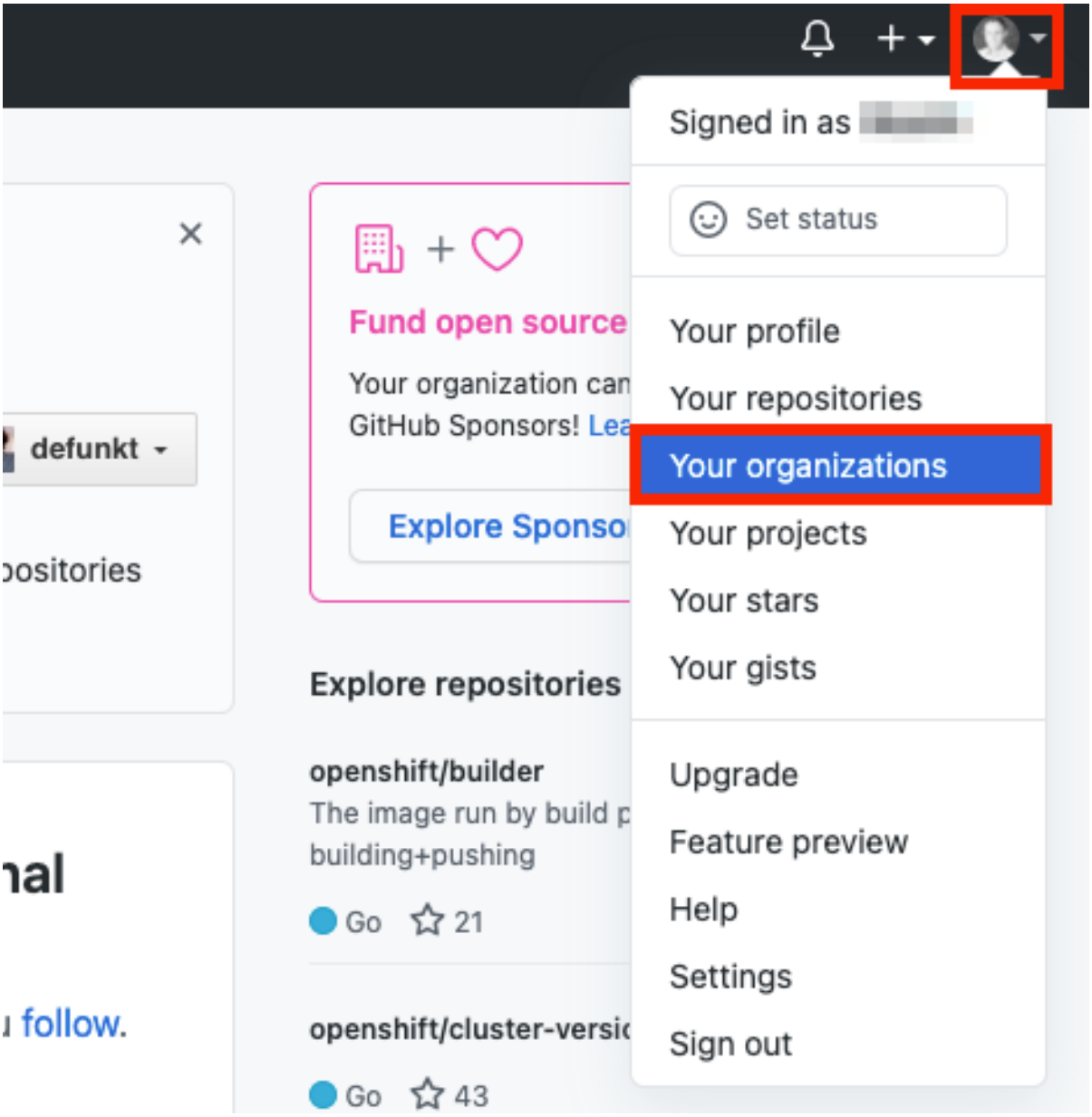
20. *复制返回的链接并将其粘贴到浏览器中。新的 IDP 应位于您选择的名称下。点您的 IDP，并使用您的 GitHub 凭证访问集群。*

15.5.2. 授予其他用户对集群的访问权限

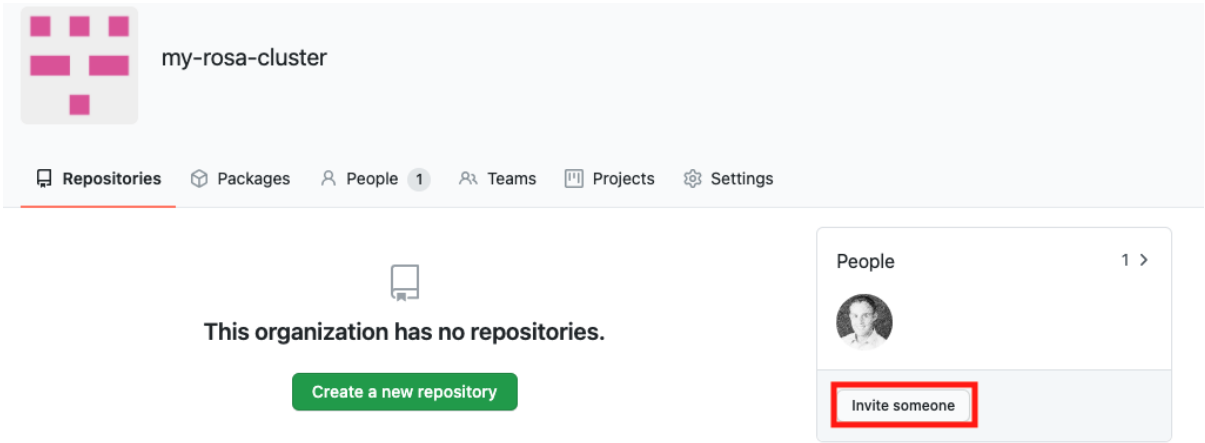
要授予对其他集群用户的访问权限，您需要将 GitHub 用户 ID 添加到用于此集群的 GitHub 机构中。

1. *在 GitHub 中，进入 您的机构 页面。*

2. *点 您的个人资料图标，然后单击 您的机构。然后单击 <your-organization-name>。在我们的示例中，是 my-rosa-cluster。*



3. ***点 Invite someone。***



4. ***输入新用户的 GitHub ID，选择正确的用户，然后单击 Invite。***

5. 新用户接受邀请后，他们将能够使用 Hybrid Cloud Console 链接及其 GitHub 凭据登录 ROSA 集群。

15.6. 教程：授予 ADMIN 权限

管理(admin)权限不会自动授予您添加到集群中的用户。如果要向某些用户授予管理员特权，则需要手动将他们授予每个用户。您可以从 ROSA 命令行界面(CLI)或 Red Hat OpenShift Cluster Manager Web 用户界面(UI)授予 admin 权限。

红帽提供了两种类型的管理特权：

- cluster-admin:cluster-admin 权限授予 admin 用户集群中的完整特权。
- dedicated-admin:dedicated-admin 特权允许 admin 用户完成大多数管理任务，但有一些限制以防止集群损坏。在需要提升权限时，最好使用 dedicated-admin。

如需有关 admin 特权的更多信息，请参阅管理集群 文档。

15.6.1. 使用 ROSA CLI

1. 假设您是创建集群的用户，请运行以下命令之一来授予 admin 权限：
   - 对于 cluster-admin ：

     ```
     $ rosa grant user cluster-admin --user <idp_user_name> --cluster=<cluster-name>
     ```

   - 对于 dedicated-admin ：

     ```
     $ rosa grant user dedicated-admin --user <idp_user_name> --cluster=<cluster-name>
     ```

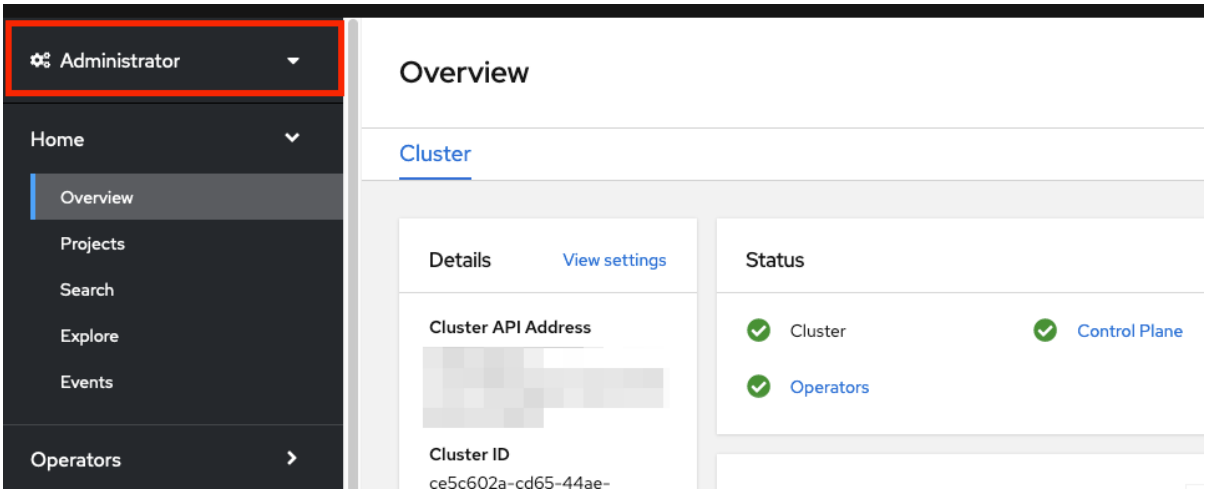
2. 运行以下命令验证 admin 权限是否已添加：

   ```
   $ rosa list users --cluster=<cluster-name>
   ```

输出示例

```
$ rosa list users --cluster=my-rosa-cluster
ID                GROUPS
<idp_user_name>   cluster-admins
```

3. *如果您当前已登录到 Red Hat Hybrid Cloud 控制台，请登出控制台并重新登录集群，以查看具有"Administrator Panel"的新视角。您可能需要一个 incognito 或 private 窗口。*



4. *您还可以运行以下命令来测试是否已将 admin 特权添加到您的帐户中。只有 cluster-admin 用户可以在不出错的情况下运行此命令。*

```
$ oc get all -n openshift-apiserver
```

15.6.2. 使用 Red Hat OpenShift Cluster Manager UI

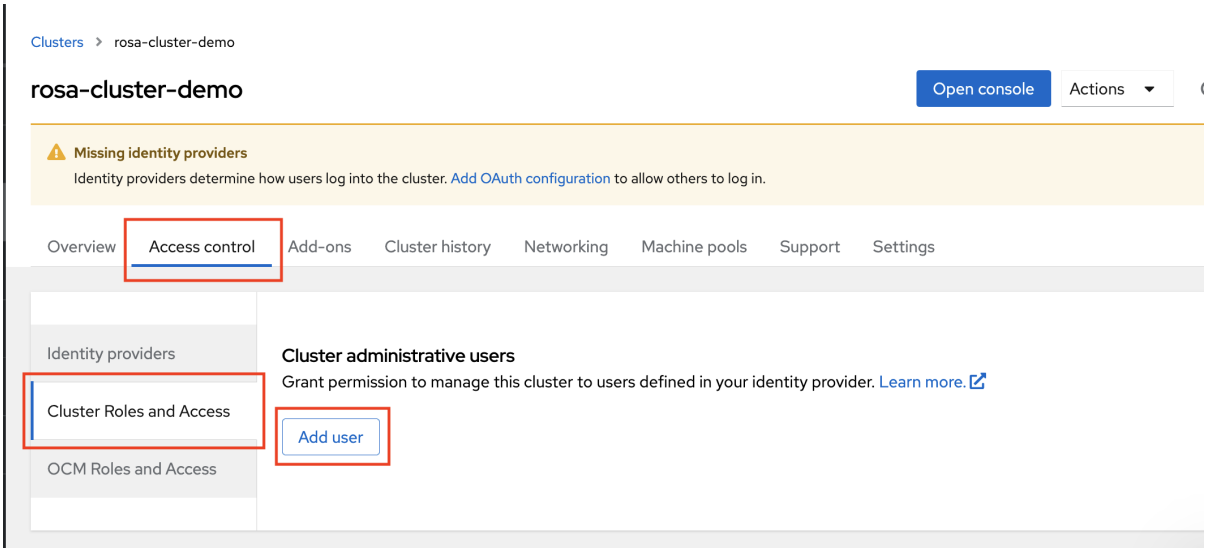
1. *登录 OpenShift Cluster Manager。*

2. *选择集群。*

3. *点 Access Control 选项卡。*

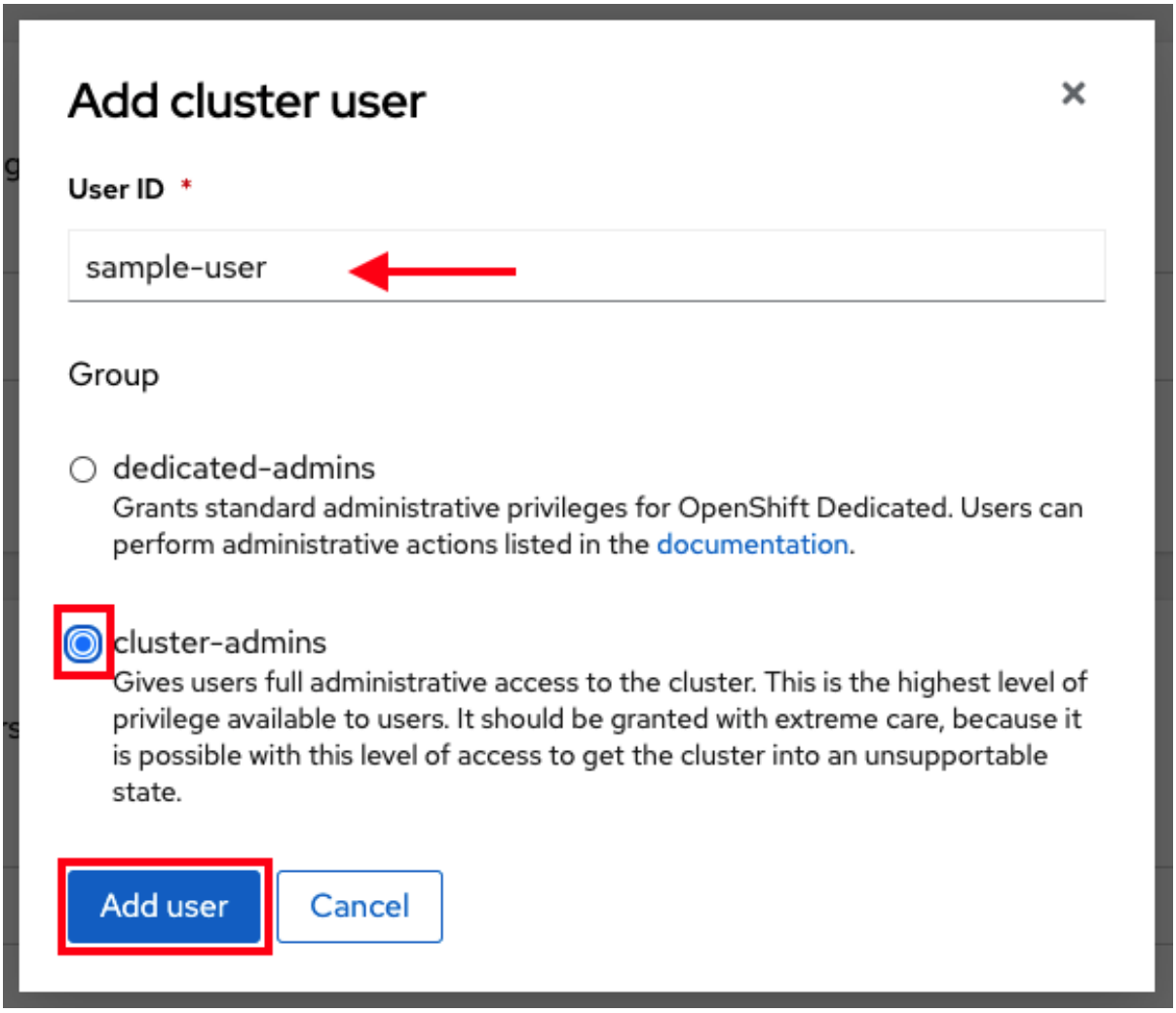
4. *点边栏中的 Cluster roles and Access 选项卡。*

5. *单击 Add user。*



6. *在弹出屏幕中，输入用户 ID。*

7. *选择您要授予用户 cluster-admins 或 dedicated-admins 权限。*



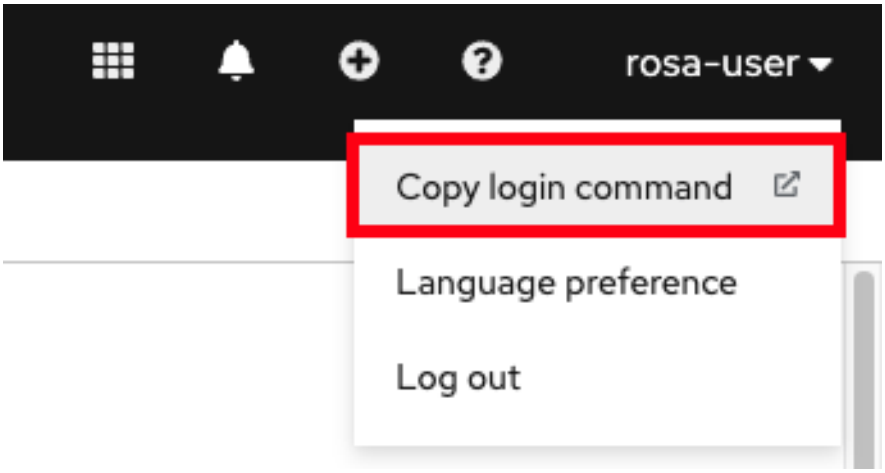
15.7. 教程：访问集群

您可以使用命令行界面(CLI)或 Red Hat Hybrid Cloud Console 用户界面(UI)连接到集群。

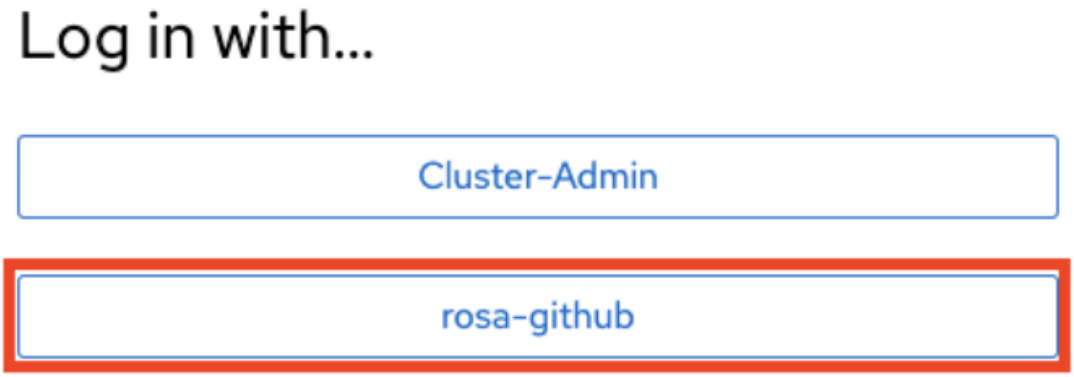
15.7.1. 使用 CLI 访问集群

要使用 CLI 访问集群，必须安装 oc CLI。如果您遵循教程，则代表已安装了 oc CLI。

1. *登录 OpenShift Cluster Manager。*
2. *点右上角的用户名。*
3. *单击 Copy Login Command。*



4. *这会打开一个新选项卡，其中可选择身份提供程序(IDP)。点您要使用的 IDP。例如："rosa-github"。*



5. *此时会打开一个新标签页。单击* **Display token。**

6. *在终端中运行以下命令：*

```
$ oc login --token=sha256~GBAfS4JQ0t1UTKYHbWAK6OUWGUkdMGz000000000000 --server=https://api.my-rosa-cluster.abcd.p1.openshiftapps.com:6443
```

输出示例

```
Logged into "https://api.my-rosa-cluster.abcd.p1.openshiftapps.com:6443" as "rosa-user" using the token provided.

You have access to 79 projects, the list has been suppressed. You can list all projects
```

```
with ' projects'

Using project "default".
```

7. 运行以下命令确认您已登录：

```
$ oc whoami
```

输出示例

```
rosa-user
```

8. 现在，您可以访问集群。

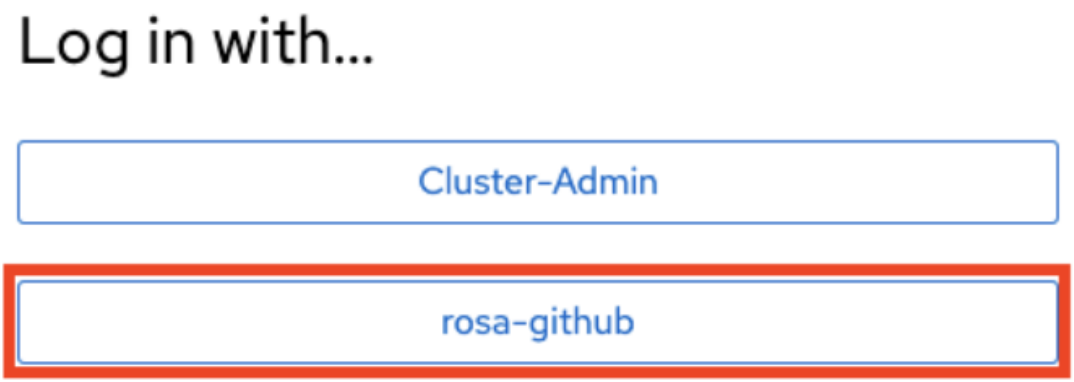
15.7.2. 通过混合云控制台访问集群

1. 登录 OpenShift Cluster Manager。

   a. 要检索混合云控制台 URL 运行：

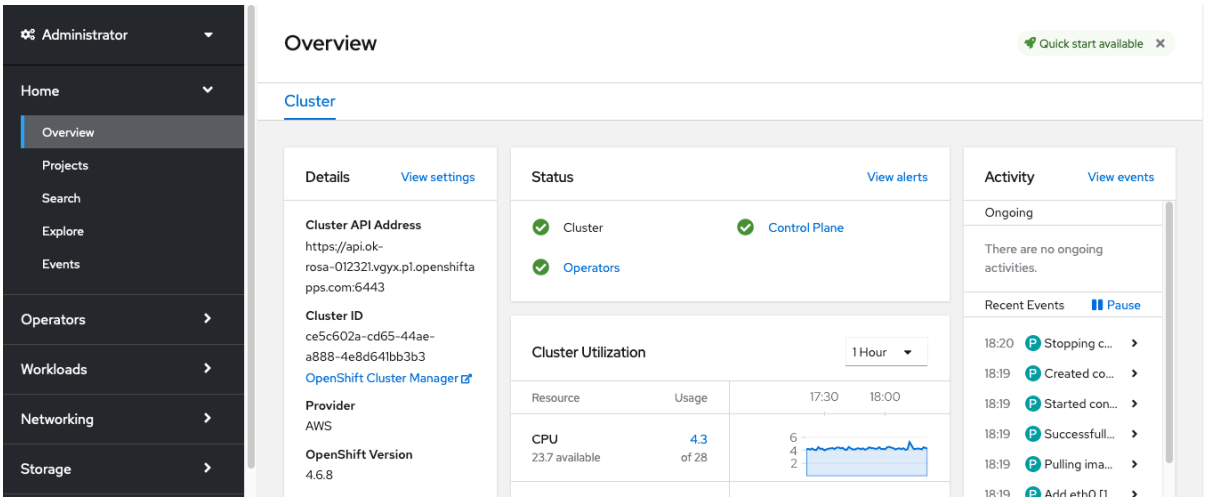
   ```
   rosa describe cluster -c <cluster-name> | grep Console
   ```

2. 点您的 IDP。例如："rosa-github"。



3. *输入您的用户凭证。*

4. *您应该已登录。如果您遵循教程，您将是一个 cluster-admin，并应该看到 Hybrid Cloud Console 网页，其中显示了 Administrator 面板。*



15.8. 教程：管理 WORKER 节点

在 Red Hat OpenShift Service on AWS (ROSA)中，使用机器池更改 worker 节点的各个方面。机器池允许用户将多个机器作为单一实体进行管理。每个 ROSA 集群都有一个创建集群时创建的默认机器池。如需更多信息，请参阅 机器池 文档。

15.8.1. 创建机器池

您可以使用命令行界面(CLI)或用户界面(UI)创建机器池。

15.8.1.1. 使用 CLI 创建机器池

1. *运行以下命令：*

```
rosa create machinepool --cluster=<cluster-name> --name=<machinepool-name> --
replicas=<number-nodes>
```

输入示例

```
$ rosa create machinepool --cluster=my-rosa-cluster --name=new-mp
--replicas=2
```

输出示例

```
I: Machine pool 'new-mp' created successfully on cluster 'my-rosa-cluster'
I: To view all machine pools, run 'rosa list machinepools -c my-rosa-cluster'
```

2. *可选：通过运行以下命令，将节点标签或污点添加到新机器池中的特定节点：*

```
rosa create machinepool --cluster=<cluster-name> --name=<machinepool-name> --
replicas=<number-nodes> --labels=`<key=pair>`
```

输入示例

```
$ rosa create machinepool --cluster=my-rosa-cluster --name=db-nodes-mp --
replicas=2 --labels='app=db','tier=backend'
```

输出示例

```
I: Machine pool 'db-nodes-mp' created successfully on cluster 'my-rosa-cluster'
```

这会创建额外的 2 个节点，它可以作为一个单元进行管理，并为它们分配显示的标签。

3. 运行以下命令确认机器池创建和分配的标签：

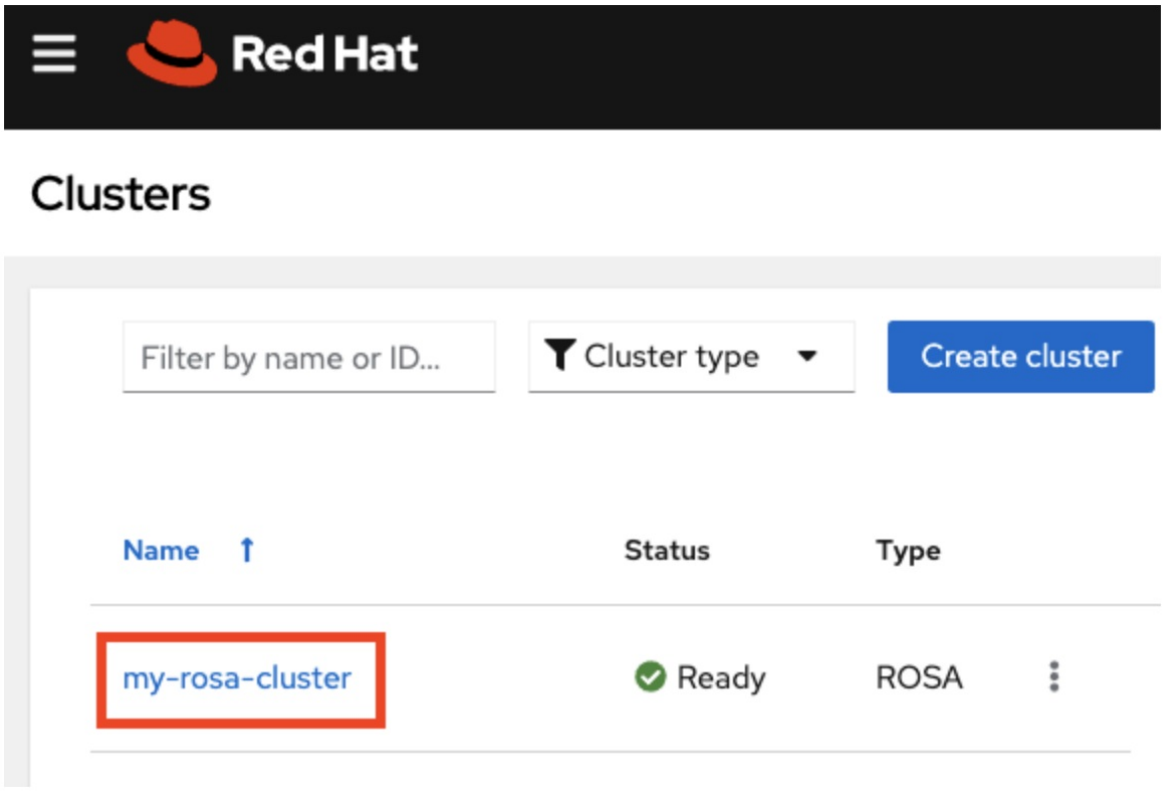
```
rosa list machinepools --cluster=<cluster-name>
```

输出示例

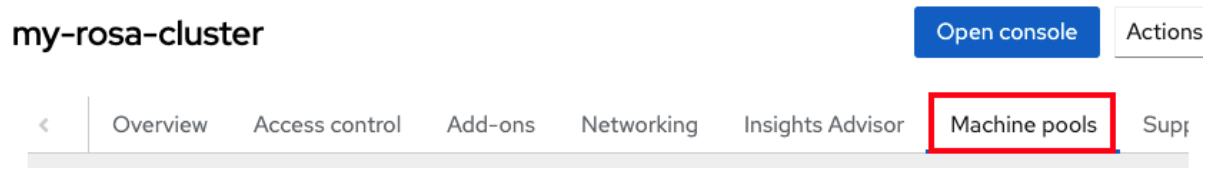
```
ID        AUTOSCALING  REPLICAS  INSTANCE TYPE  LABELS          TAINTS  AVAILABILITY ZONES
Default   No           2         m5.xlarge                              us-east-1a
```

15.8.1.2. 使用 UI 创建机器池

1. 登录 OpenShift Cluster Manager 并点您的集群。



2. *单击 **Machine pool**。*

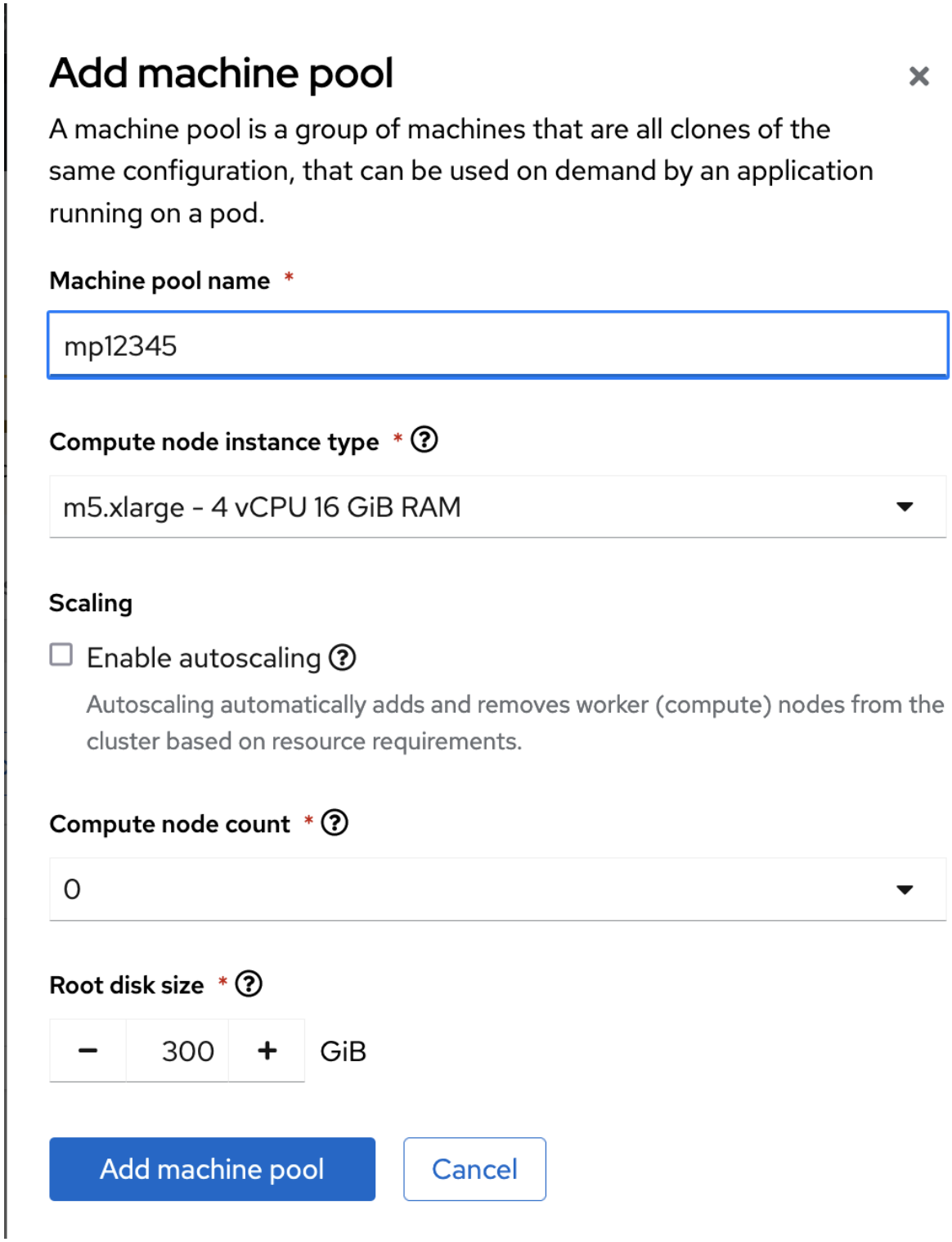


3. ***点 Add machine pool。***

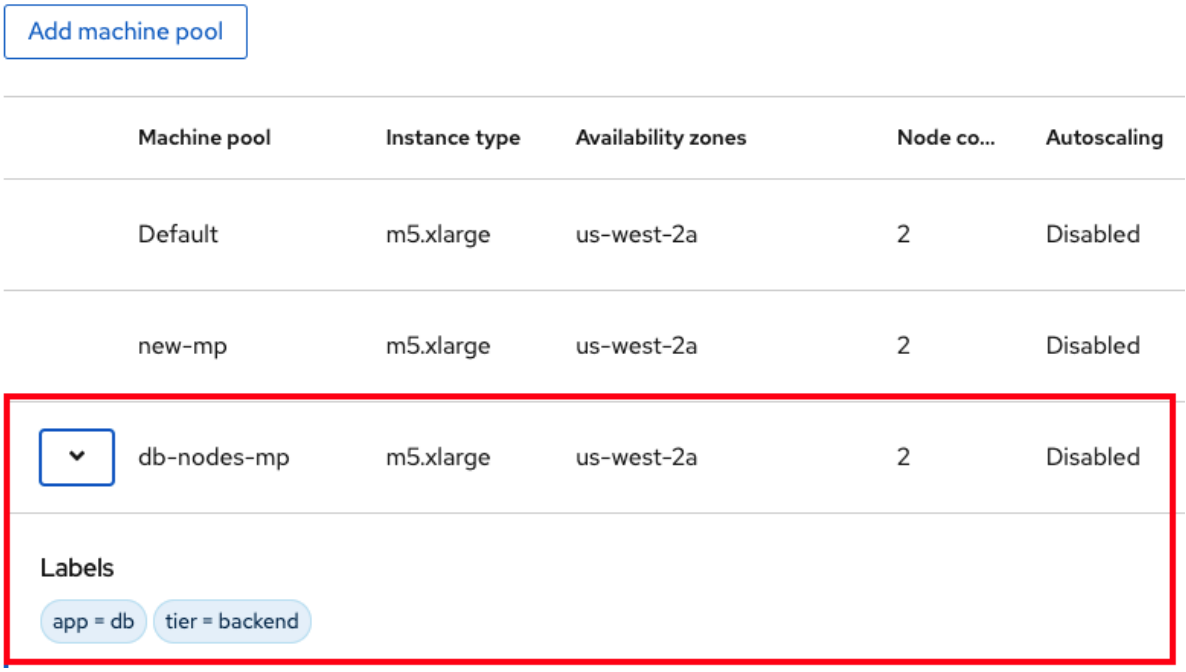
4. ***输入所需的配置。***

提示

您还可以扩展 Edit 节点标签和污点 部分，将节点标签和污点添加到机器池中的节点。



5. *您将看到您创建的新机器池。*



15.8.2. 扩展 worker 节点

编辑机器池，以扩展该特定机器池中的 worker 节点数量。您可以使用 CLI 或 UI 来缩放 worker 节点。

15.8.2.1. 使用 CLI 扩展 worker 节点

1. *运行以下命令，以查看每个集群创建的默认机器池：*

```
rosa list machinepools --cluster=<cluster-name>
```

输出示例

```
ID        AUTOSCALING  REPLICAS  INSTANCE TYPE  LABELS          TAINTS
AVAILABILITY ZONES
Default   No           2         m5.xlarge                      us-east-1a
```

2. *要将默认机器池扩展到不同数量的节点，请运行以下命令：*

```
rosa edit machinepool --cluster=<cluster-name> --replicas=<number-nodes>
<machinepool-name>
```

输入示例

```
rosa edit machinepool --cluster=my-rosa-cluster --replicas 3 Default
```

3. 运行以下命令确认机器池已扩展：

```
rosa describe cluster --cluster=<cluster-name> | grep Compute
```

输入示例

```
$ rosa describe cluster --cluster=my-rosa-cluster | grep Compute
```

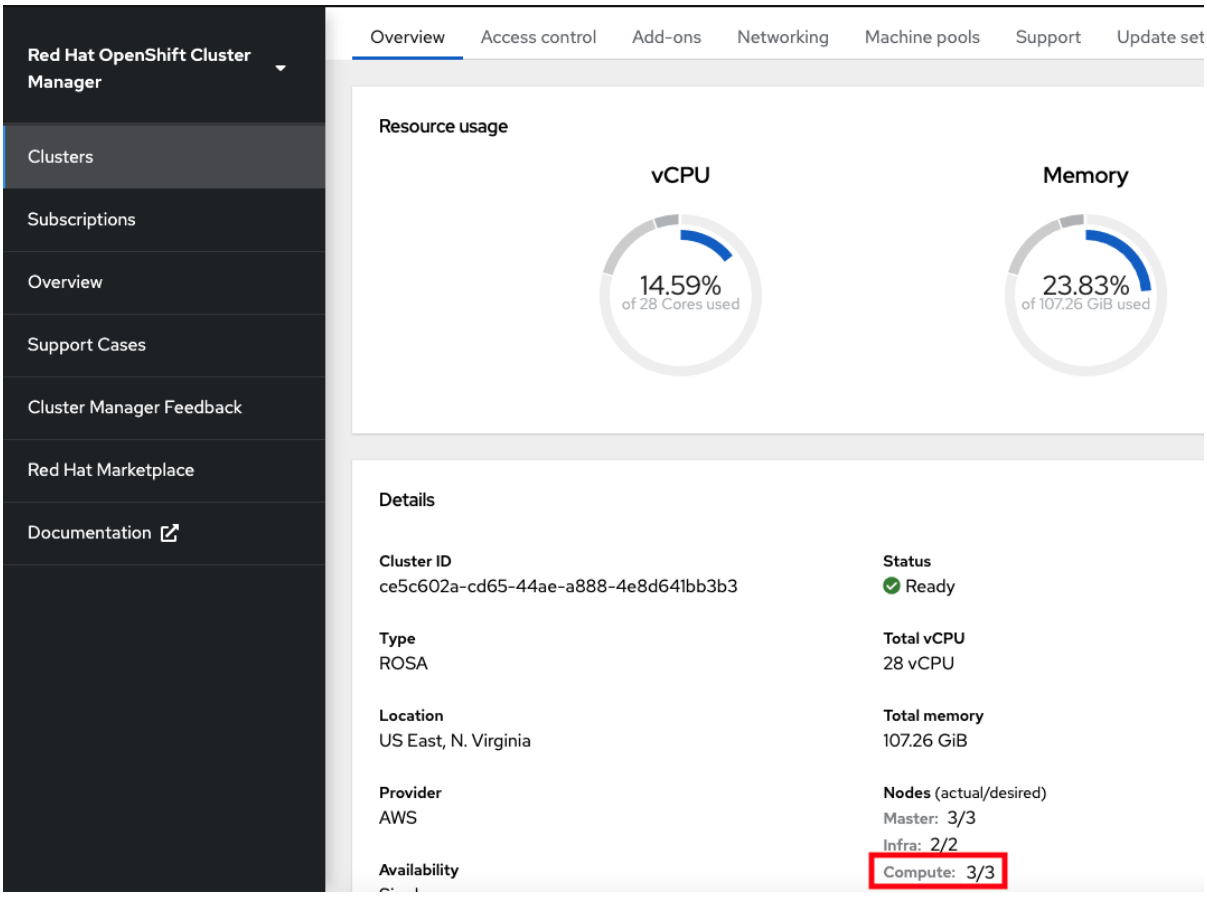
输出示例

```
- Compute:                3 (m5.xlarge)
```

15.8.2.2. 使用 UI 扩展 worker 节点

1. 点击您要编辑的机器池右侧的三个点。
2. 点 Edit。
3. 输入所需的节点数量，然后单击 Save。
4. 选择集群，点 Overview 选项卡并滚动到 Compute 列表 来确认集群已扩展。计算列表应当

等于扩展的节点。例如： 3/3。



15.8.2.3. 添加节点标签

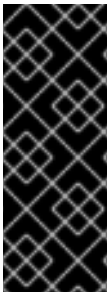
1. *使用以下命令添加节点标签：*

```
rosa edit machinepool --cluster=<cluster-name> --replicas=<number-nodes> --
labels='key=value' <machinepool-name>
```

输入示例

```
rosa edit machinepool --cluster=my-rosa-cluster --replicas=2 --labels
'foo=bar','baz=one' new-mp
```

这会在新机器池中添加 2 个标签。



重要

这个命令将所有机器池配置替换为新定义的配置。如果要添加另一个标签并保留旧标签，则必须同时说明新的和已存在的标签。否则，命令会将所有已存在的标签替换为您要添加的标签。同样，如果要删除标签，请运行命令并声明您要删除的标签，不包括您要删除的标签。

15.8.3. 混合节点类型

您还可以使用新机器池在同一集群中混合不同的 worker 节点机器类型。创建机器池后，您无法更改节点类型，但您可以通过添加 --instance-type 标志来创建具有不同节点的新机器池。

1. 例如，要将数据库节点改为不同的节点类型，请运行以下命令：

```
rosa create machinepool --cluster=<cluster-name> --name=<mp-name> --replicas=
<number-nodes> --labels='<key=pair>' --instance-type=<type>
```

输入示例

```
rosa create machinepool --cluster=my-rosa-cluster --name=db-nodes-large-mp --
replicas=2 --labels='app=db','tier=backend' --instance-type=m5.2xlarge
```

2. 要查看 所有可用的实例类型，请运行以下命令：

```
rosa list instance-types
```

3. 要逐步更改，请使用 --interactive 标志：

```
rosa create machinepool -c <cluster-name> --interactive
```

```
[? Machine pool name: large-nodes-pool
[? Enable autoscaling (optional): No
[? Replicas: 3
 ? Instance type:  [Use arrows to move, type to filter, ? for more help]
 > m5.xlarge
   r5.xlarge
   r5.2xlarge
   m5.2xlarge
   c5.2xlarge
   r5.4xlarge
   m5.4xlarge
```

4. *运行以下命令列出机器池并查看新的、更大的实例类型：*

```
rosa list machinepools -c <cluster-name>
```

```
ID                  AUTOSCALING  REPLICAS  INSTANCE TYPE  LABELS                  TAINTS    AVAILABILITY ZONES
default             No           3         m5.xlarge                                        us-east-1a
db-nodes-large--mp  No           2         m5.2xlarge     app=db, tier=backend              us-east-1a
db-nodes-mp         No_          2         m5.xlarge      app=db, tier=backend              us-east-1a
```

15.9. 教程： 自动扩展

集群自动扩展 根据 pod 资源从集群中删除 worker 节点。

集群自动扩展在以下情况下增加集群的大小：

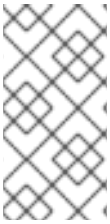
- *由于资源不足，Pod 无法调度到当前节点上。*
- *另一个节点是满足部署需求所必需的。*

集群自动扩展不会将集群资源增加到超过您指定的限制。

集群自动扩展在以下情况下减小集群的大小：

- *显著期间不需要某些节点。例如，当某个节点的资源使用率较低且其所有重要 pod 都可以安置在其他节点上时。*

15.9.1. 使用 CLI 为现有机器池启用自动扩展



注意

集群自动扩展可以在集群创建时启用，并使用 --enable-autoscaling 选项创建新机器池。

1. 根据机器池可用性设置自动扩展。要查找哪些机器池可用于自动扩展，请运行以下命令：

```
$ rosa list machinepools -c <cluster-name>
```

输出示例

```
ID       AUTOSCALING  REPLICAS  INSTANCE TYPE  LABELS    TAINTS
AVAILABILITY ZONES
Default  No           2         m5.xlarge                  us-east-1a
```

2. 运行以下命令，将自动扩展添加到可用机器池中：

```
$ rosa edit machinepool -c <cluster-name> --enable-autoscaling <machinepool-name>
--min-replicas=<num> --max-replicas=<num>
```

输入示例

```
$ rosa edit machinepool -c my-rosa-cluster --enable-autoscaling Default --min-
replicas=2 --max-replicas=4
```

以上命令为 worker 节点创建一个自动扩展程序，根据资源在 2 到 4 个节点之间扩展。

15.9.2. 使用 UI 为现有机器池启用自动扩展

注意

在创建机器池时，可以通过选中 Enable autoscaling 复选框，在集群创建时启用集群自动扩展。

1. 进入 Machine pool 选项卡，然后点击右侧的三个点。

2. 单击 Scale，然后单击 Enable autoscaling。

3. 运行以下命令确认添加了自动扩展：

```
$ rosa list machinepools -c <cluster-name>
```

输出示例

```
ID       AUTOSCALING  REPLICAS  INSTANCE TYPE  LABELS    TAINTS
AVAILABILITY ZONES
Default  Yes          2-4       m5.xlarge                  us-east-1a
```

15.10. 教程：升级集群

Red Hat OpenShift Service on AWS (ROSA)作为受管服务的一部分执行所有集群升级。您不需要运行任何命令或对集群进行更改。您可以在方便的时间调度升级。

调度集群升级的方法包括：

- 手动使用命令行界面(CLI)：启动一次性立即升级，或调度未来日期和时间的一次性升级。
- 手动使用 Red Hat OpenShift Cluster Manager 用户界面(UI)：启动一次性升级，或为将来的日期和时间计划一次性升级。
-

自动化升级 ：当有新版本可用时，为重复的 y-stream 升级设置一个升级窗口，而无需手动调度。必须手动调度次版本。

有关集群升级的详情，请运行以下命令：

```
$ rosa upgrade cluster --help
```

15.10.1. 使用 CLI 手动升级集群

1. 运行以下命令检查是否有可用的升级：

```
$ rosa list upgrade -c <cluster-name>
```

输出示例

```
$ rosa list upgrade -c <cluster-name>
VERSION  NOTES
4.14.7   recommended
4.14.6
...
```

在上例中，版本 4.14.7 和 4.14.6 都可用。

2. 运行以下命令，将集群计划在一小时内升级：

```
$ rosa upgrade cluster -c <cluster-name> --version <desired-version>
```

3. 可选：运行以下命令，将集群计划为在更新的日期和时间升级：

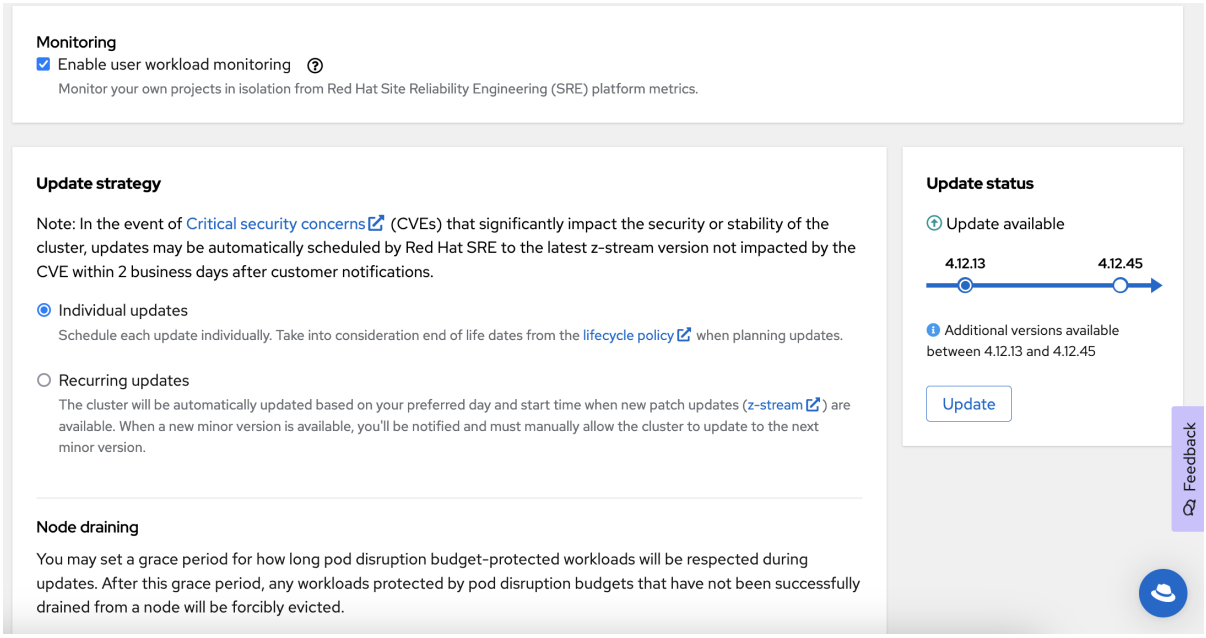
```
$ rosa upgrade cluster -c <cluster-name> --version <desired-version> --schedule-date
<future-date-for-update> --schedule-time <future-time-for-update>
```

15.10.2. 使用 UI 手动升级集群

1. *登录到 OpenShift Cluster Manager，再选择您要升级的集群。*

2. *单击 Settings。*

3. *如果有可用的升级，点 Update。*



4. *选择您要在新窗口中升级的版本。*

5. *为升级调度一个时间，或立即开始。*

15.10.3. 设置自动重复升级

1. *登录到 OpenShift Cluster Manager，再选择您要升级的集群。*

2. *单击 Settings。*

    1. *在 Update Strategy 下，单击 Recurring updates。*

3. *设置升级的日期和时间。*

4. 在 节点排空 下，选择一个宽限期，以允许节点在 pod 驱除前排空。

5. 点击 Save。

15.11. 教程：删除集群

您可以使用命令行界面(CLI)或用户界面(UI)删除 Red Hat OpenShift Service on AWS (ROSA)集群。

15.11.1. 使用 CLI 删除 ROSA 集群

1. 可选：运行以下命令来列出集群以确保您删除正确的集群：

```
$ rosa list clusters
```

2. 运行以下命令来删除集群：

```
$ rosa delete cluster --cluster <cluster-name>
```

警告

此命令不可恢复。

3. CLI 会提示您确认要删除集群。按 y，然后输入。集群及其所有相关基础架构都将被删除。

注意

所有 AWS STS 和 IAM 角色和策略将保留，必须按照以下步骤在集群删除完成后手动删除。

4. CLI 输出删除创建的 OpenID Connect (OIDC)供应商和 Operator IAM 角色资源的命令。等待集群完成删除后再删除这些资源。运行以下命令执行快速状态检查：

```
$ rosa list clusters
```

5. 删除集群后，运行以下命令来删除 OIDC 供应商：

```
$ rosa delete oidc-provider -c <clusterID> --mode auto --yes
```

6. 运行以下命令来删除 Operator IAM 角色：

```
$ rosa delete operator-roles -c <clusterID> --mode auto --yes
```

注意

此命令需要集群 ID，而不是集群名称。

7. 只有剩余帐户角色被同一帐户中其他集群不再需要时，才删除剩余的帐户角色。如果要在此帐户中创建其他 ROSA 集群，请不要执行此步骤。

要删除帐户角色，您需要知道创建它们时使用的前缀。除非另有指定，否则默认为 "ManagedOpenShift"。

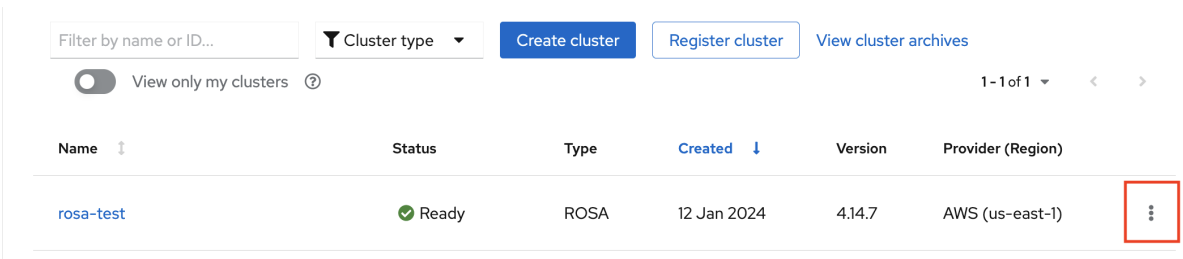
运行以下命令来删除帐户角色：

```
$ rosa delete account-roles --prefix <prefix> --mode auto --yes
```

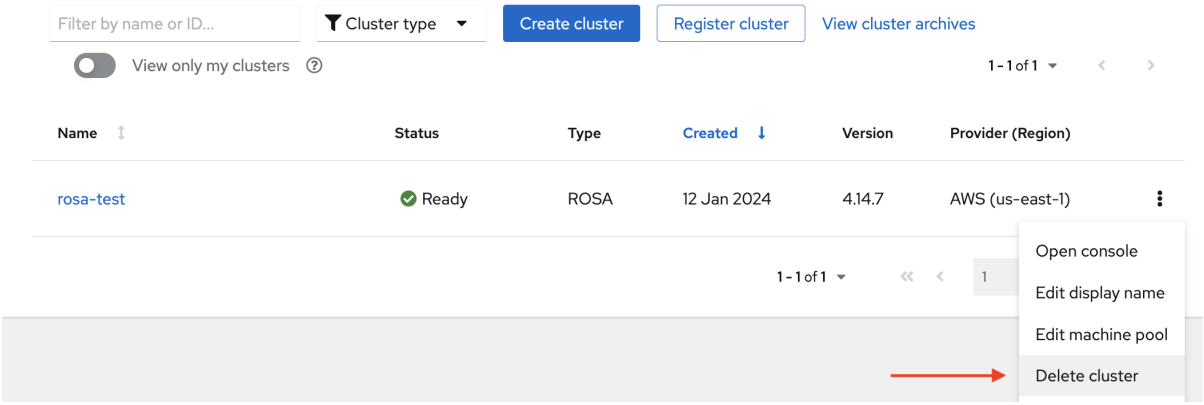
15.11.2. 使用 UI 删除 ROSA 集群

1. 登录 OpenShift Cluster Manager，找到您要删除的集群。

2. 点击集群右侧的三个点。



3. 在下拉菜单中，单击 删除集群。



4. 输入集群名称以确认删除，然后点 Delete。

15.12. 教程：获取支持

在需要时查找正确的帮助非常重要。当您需要帮助时，这些是一些资源。

15.12.1. 添加支持联系人

您可以为集群通信添加额外的电子邮件地址。

1. 在 Red Hat OpenShift Cluster Manager 用户界面(UI)上，单击 选择集群。
2. 点 Support 选项卡。
3. 单击 Add notification contact，并输入其他电子邮件地址。

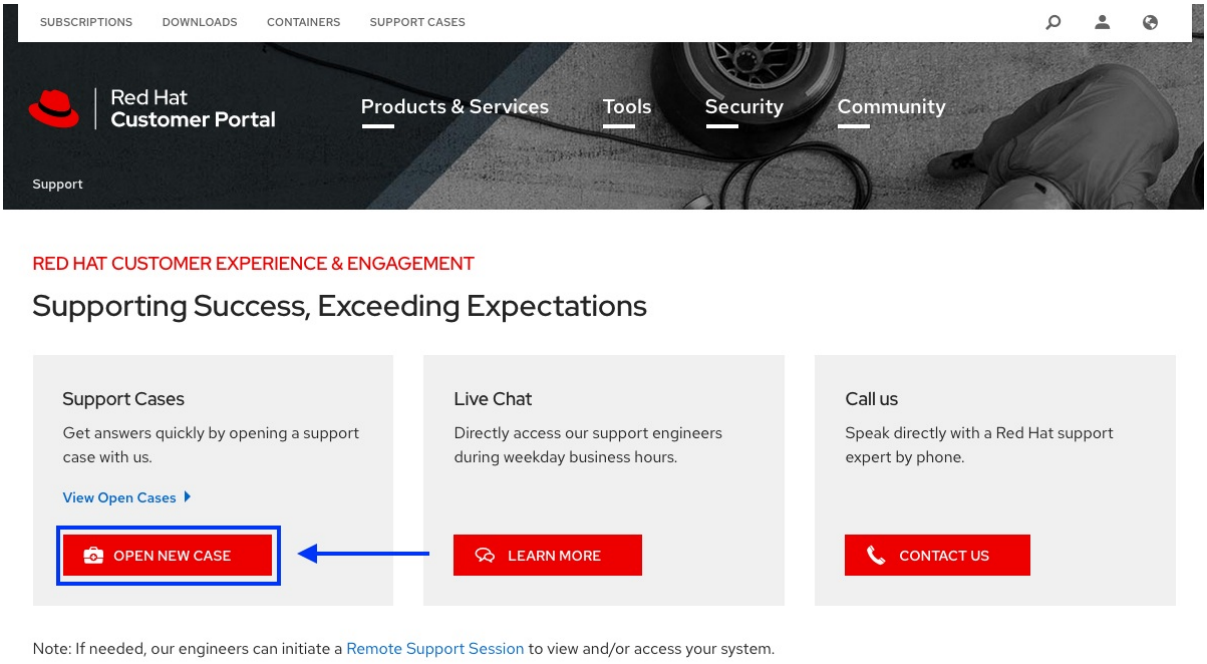
15.12.2. 使用 UI 联系红帽以获取支持

1. 在 OpenShift Cluster Manager UI 中，点 Support 选项卡。
2. 点 Open 支持问题单。

15.12.3. 使用支持页面联系红帽以获取支持

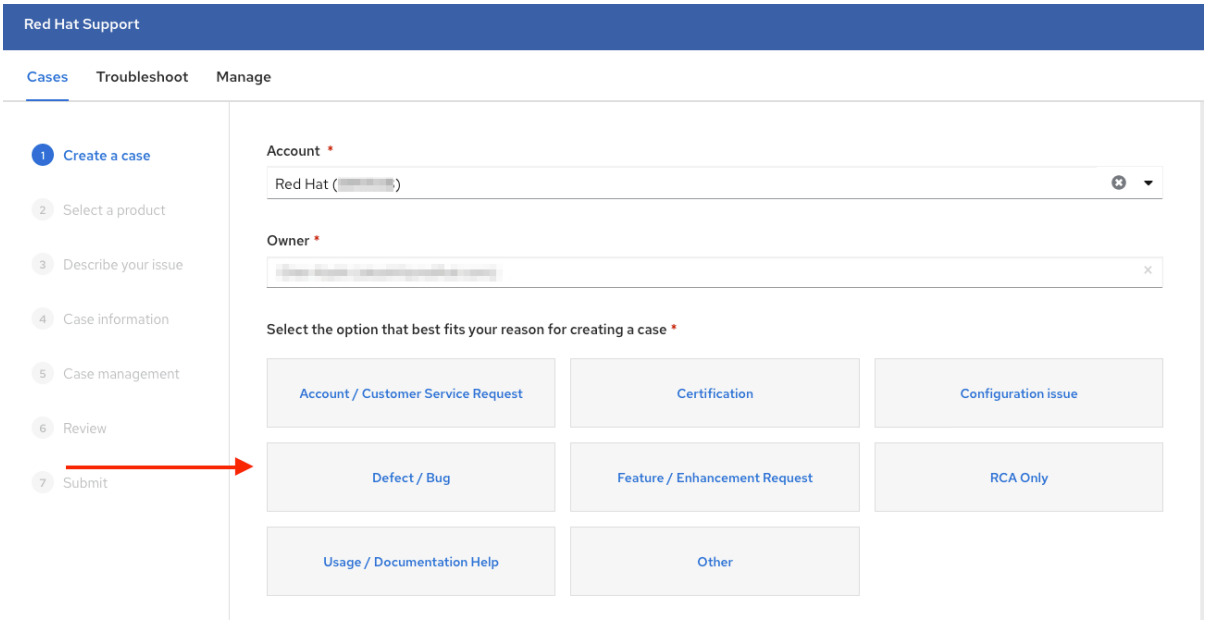
1. *请访问 红帽支持页面。*

2. *点 Open a new Case。*

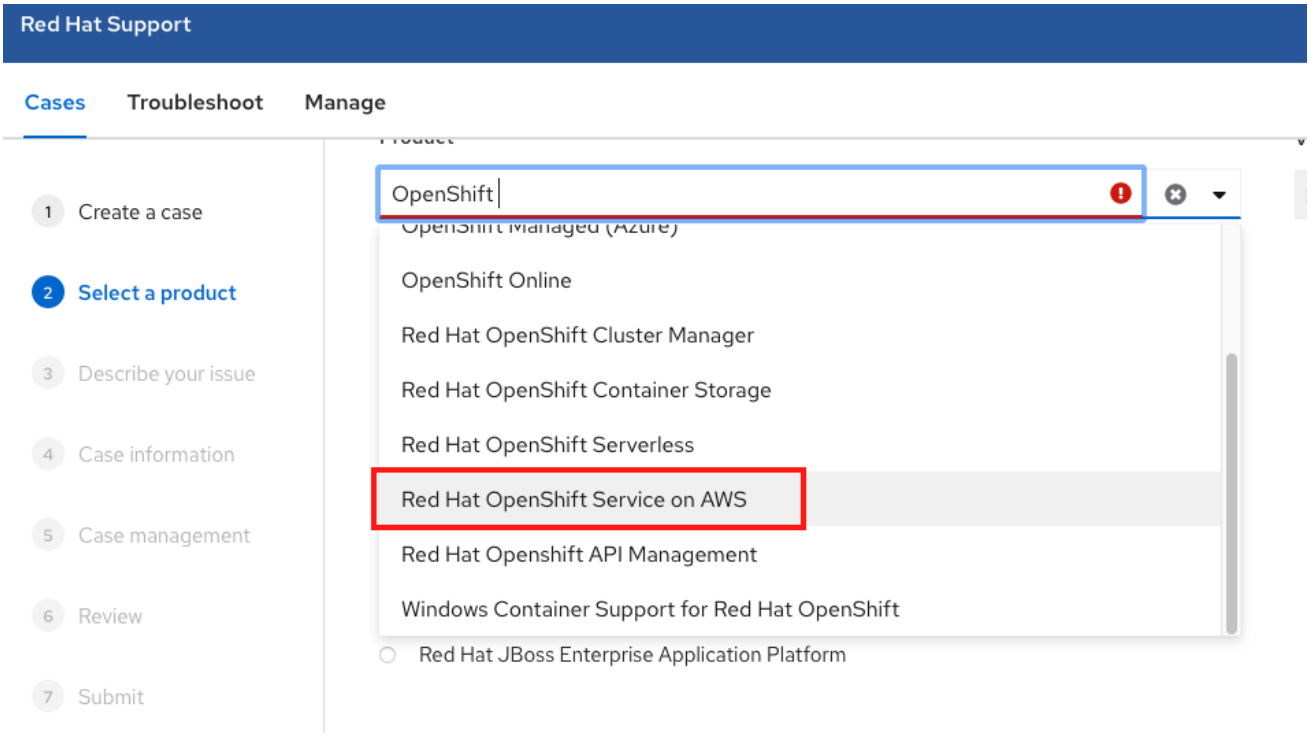


3. *登录到您的红帽帐户。*

4. *选择联系支持的原因。*



5. *选择 Red Hat OpenShift Service on AWS。*

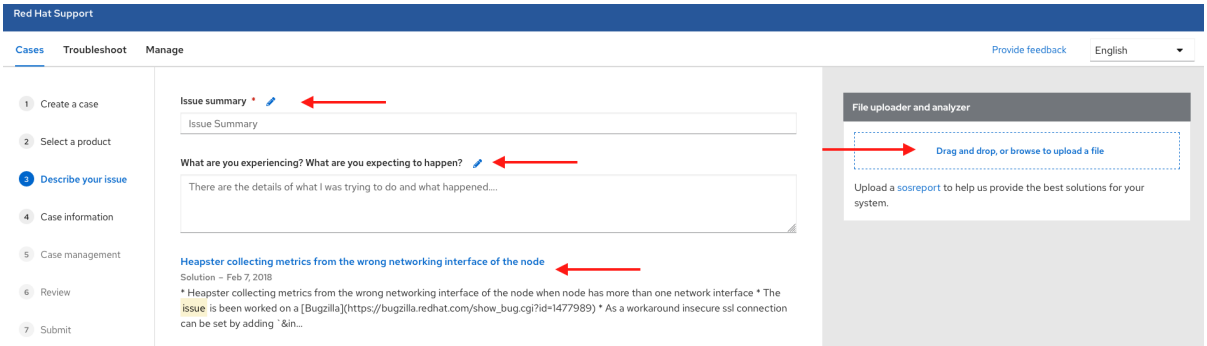


1. *单击 继续。*

2. *输入问题概述以及您的请求详情。上传任何文件、日志和屏幕截图。您提供的更多详细信息，更好的红帽支持可以帮助您。*

注意

本页底部可能会帮助您解决相关问题的相关建议。



3. *点 Continue。*

4. *回答新字段中的问题。*

5. 点 Continue。

6. 输入有关您的问题单的以下信息：

   a. 支持级别： 高级

   b. 严重性 ： 查看红帽支持严重性等级定义以选择正确的定义。

   c. Group ： 如果与其它几个情况相关，您可以选择对应的组。

   d. 语言

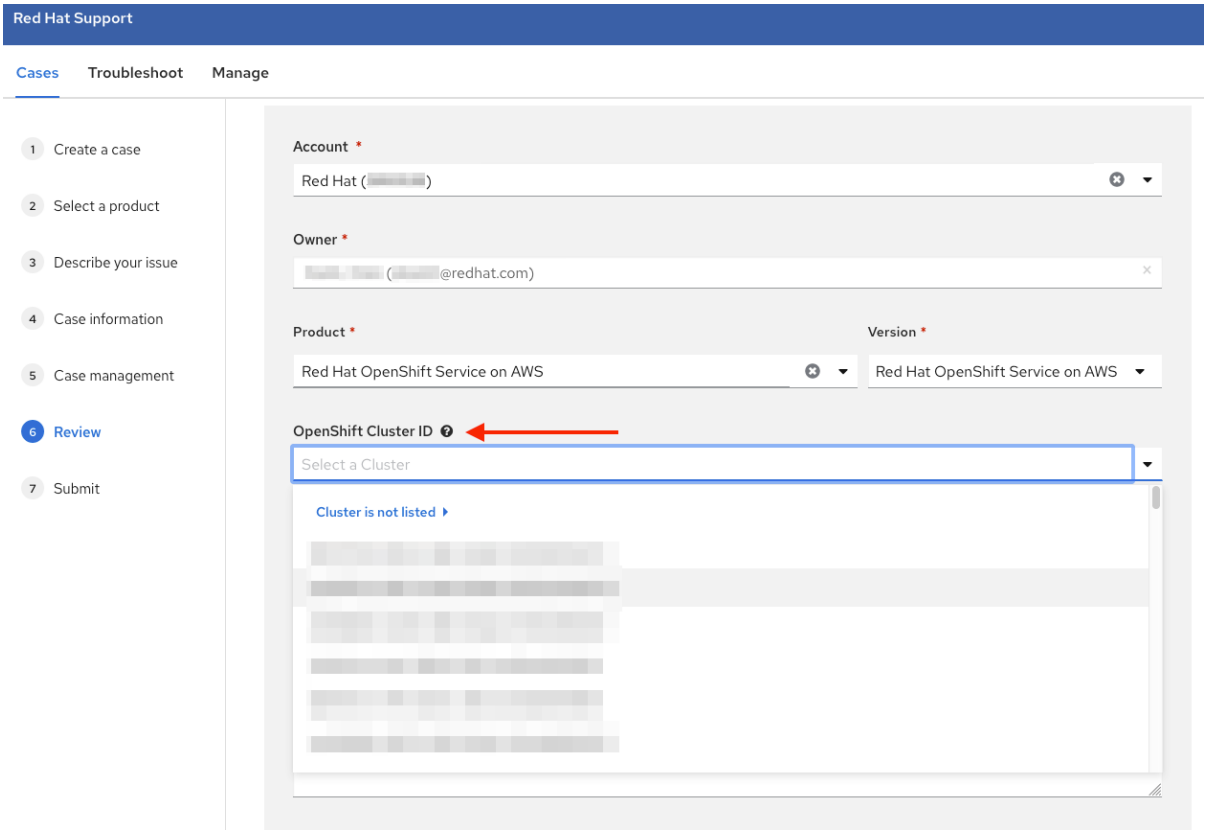
   e. 发送通知： 添加任何其他电子邮件地址以通知活动。

   f. 红帽人员： 如果您正在与红帽的任何人合作，并希望将其保存在循环中，您可以在此处输入其电子邮件地址。

   g. 备用问题单 ID： 如果要将自己的 ID 附加到其中，您可以在此处输入它。

7. 点 Continue。

8. 在查看屏幕上，确保您选择要联系支持的正确集群 ID。



9. ***点 Submit。***

10. ***您将根据为 指定严重性级别 提交的响应时间联系。***

第 16 章 部署应用程序

16.1. 教程：部署应用程序

16.1.1. 简介

成功置备集群后，您可以在其上部署应用程序。此应用程序允许您更熟悉 Red Hat OpenShift Service on AWS (ROSA)和 Kubernetes 的一些功能。

16.1.1.1. 实验概述

在此实验室中，您将完成以下一组任务，旨在帮助您了解部署和操作基于容器的应用程序的概念：

- 使用 S2I 和 Kubernetes Deployment 对象部署基于 Node.js 的应用。
- 设置持续交付(CD)管道，以自动推送源代码更改。
- 探索日志记录。
- 体验自我修复应用程序。
- 通过 configmaps、secret 和环境变量探索配置管理。
- 使用持久性存储在 Pod 重启后共享数据。
- 探索 Kubernetes 和应用程序内的网络。
- 熟悉 ROSA 和 Kubernetes 功能。
- 根据 Horizontal Pod Autoscaler 的负载自动缩放 pod。

- 使用 AWS Controller for Kubernetes (ACK)来部署和使用 S3 存储桶。

此实验室使用 ROSA CLI 或 ROSA Web 用户界面(UI)。

16.2. 教程：部署应用程序

16.2.1. 前提条件

1. Provisioned ROSA 集群

   此实验假设您有权访问成功调配的 ROSA 集群。如果您还没有创建 ROSA 集群，请参阅 Red Hat OpenShift Service on AWS 快速开始指南。

2. OpenShift 命令行界面(CLI)

   如需更多信息，请参阅 OpenShift CLI 入门。

3. GitHub 帐户

   使用您现有的 GitHub 帐户或在 https://github.com/signup 注册。

16.3. 教程：部署应用程序

16.3.1. 实验概述

16.3.1.1. 实验资源

- OSToy 应用程序的源代码
- OSToy 前端容器镜像
- OSToy 微服务容器镜像

- *部署定义 YAML 文件：*

ostoy-frontend-deployment.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: ostoy-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ostoy-frontend
  labels:
    app: ostoy
spec:
  selector:
    matchLabels:
      app: ostoy-frontend
  strategy:
    type: Recreate
  replicas: 1
  template:
    metadata:
      labels:
        app: ostoy-frontend
    spec:
      # Uncomment to use with ACK portion of the workshop
      # If you chose a different service account name please replace it.
      # serviceAccount: ostoy-sa
      containers:
      - name: ostoy-frontend
        securityContext:
          allowPrivilegeEscalation: false
          runAsNonRoot: true
          seccompProfile:
            type: RuntimeDefault
          capabilities:
            drop:
            - ALL
        image: quay.io/ostoylab/ostoy-frontend:1.6.0
        imagePullPolicy: IfNotPresent
        ports:
        - name: ostoy-port
          containerPort: 8080
```

```
        resources:
          requests:
            memory: "256Mi"
            cpu: "100m"
          limits:
            memory: "512Mi"
            cpu: "200m"
        volumeMounts:
        - name: configvol
          mountPath: /var/config
        - name: secretvol
          mountPath: /var/secret
        - name: datavol
          mountPath: /var/demo_files
        livenessProbe:
          httpGet:
            path: /health
            port: 8080
          initialDelaySeconds: 10
          periodSeconds: 5
        env:
        - name: ENV_TOY_SECRET
          valueFrom:
            secretKeyRef:
              name: ostoy-secret-env
              key: ENV_TOY_SECRET
        - name: MICROSERVICE_NAME
          value: OSTOY_MICROSERVICE_SVC
        - name: NAMESPACE
          valueFrom:
            fieldRef:
              fieldPath: metadata.namespace
      volumes:
        - name: configvol
          configMap:
            name: ostoy-configmap-files
        - name: secretvol
          secret:
            defaultMode: 420
            secretName: ostoy-secret
        - name: datavol
          persistentVolumeClaim:
            claimName: ostoy-pvc
---
apiVersion: v1
kind: Service
metadata:
  name: ostoy-frontend-svc
  labels:
    app: ostoy-frontend
spec:
  type: ClusterIP
  ports:
    - port: 8080
      targetPort: ostoy-port
      protocol: TCP
```

```
    name: ostoy
  selector:
    app: ostoy-frontend
---
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  name: ostoy-route
spec:
  to:
    kind: Service
    name: ostoy-frontend-svc
---
apiVersion: v1
kind: Secret
metadata:
  name: ostoy-secret-env
type: Opaque
data:
  ENV_TOY_SECRET: VGhpcyBpcyBhIHRlc3Q=
---
kind: ConfigMap
apiVersion: v1
metadata:
  name: ostoy-configmap-files
data:
  config.json:  '{ "default": "123" }'
---
apiVersion: v1
kind: Secret
metadata:
  name: ostoy-secret
data:
  secret.txt:
VVNFUk5BTUU9bXlfdXNlcgpQQVNTV09SRD1AT3RCbCVYQXAhIzYzMlk1RndDQE1UU
WsKU01UUD1sb2NhbGhvc3QKU01UUF9QT1JUPTI1
type: Opaque
```

ostoy-microservice-deployment.yaml

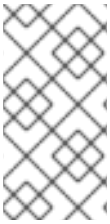
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ostoy-microservice
  labels:
    app: ostoy
spec:
  selector:
    matchLabels:
```

```
      app: ostoy-microservice
  replicas: 1
  template:
    metadata:
      labels:
        app: ostoy-microservice
    spec:
      containers:
      - name: ostoy-microservice
        securityContext:
          allowPrivilegeEscalation: false
          runAsNonRoot: true
          seccompProfile:
            type: RuntimeDefault
          capabilities:
            drop:
            - ALL
        image: quay.io/ostoylab/ostoy-microservice:1.5.0
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 8080
          protocol: TCP
        resources:
          requests:
            memory: "128Mi"
            cpu: "50m"
          limits:
            memory: "256Mi"
            cpu: "100m"
---
apiVersion: v1
kind: Service
metadata:
  name: ostoy-microservice-svc
  labels:
    app: ostoy-microservice
spec:
  type: ClusterIP
  ports:
    - port: 8080
      targetPort: 8080
      protocol: TCP
  selector:
    app: ostoy-microservice
```

- *ACK S3 的 S3 存储桶清单*

s3-bucket.yaml

```
apiVersion: s3.services.k8s.aws/v1alpha1
kind: Bucket
metadata:
  name: ostoy-bucket
  namespace: ostoy
spec:
  name: ostoy-bucket
```



注意

为简化 OSToy 应用程序的部署，上述部署清单中所需的所有对象都被分组在一起。对于典型的企业部署，建议为每个 Kubernetes 对象有一个单独的清单文件。

16.3.1.2. 关于 OSToy 应用程序

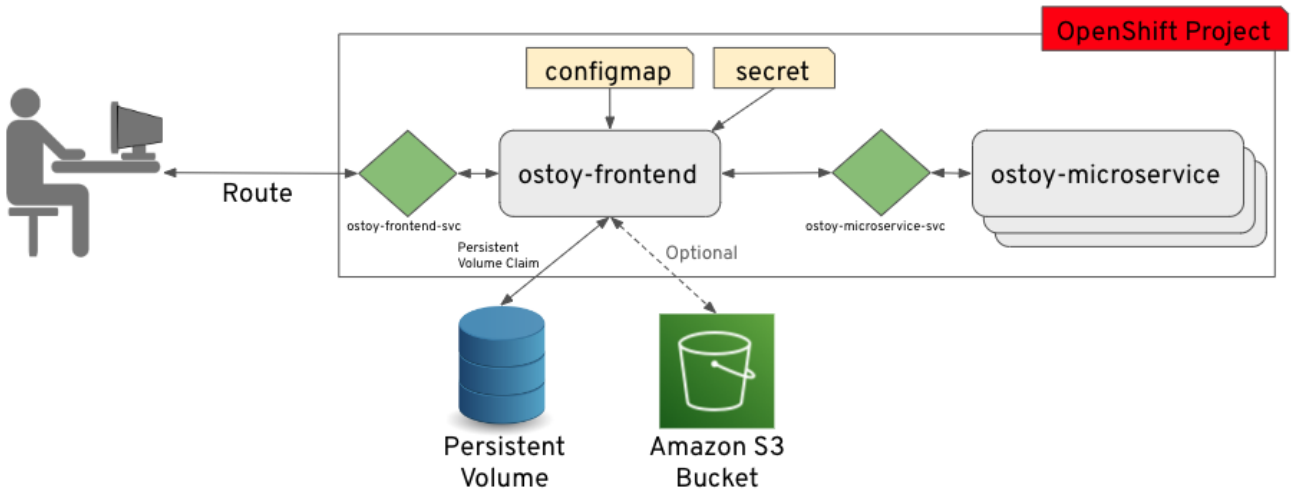
OSToy 是一个简单的 Node.js 应用程序，您将部署到 ROSA 集群，以帮助探索 Kubernetes 的功能。此应用程序有一个用户界面，您可以在其中：

- 将消息写入日志(stdout / stderr)。
- 有意使应用崩溃，以查看自我修复。
- 切换存活度探测并监控 OpenShift 行为。
- 读取配置映射、secret 和 env 变量。
- 如果连接到共享存储，读取和写入文件。
- 检查包含微服务的网络连接、集群内 DNS 和 intra-communication。

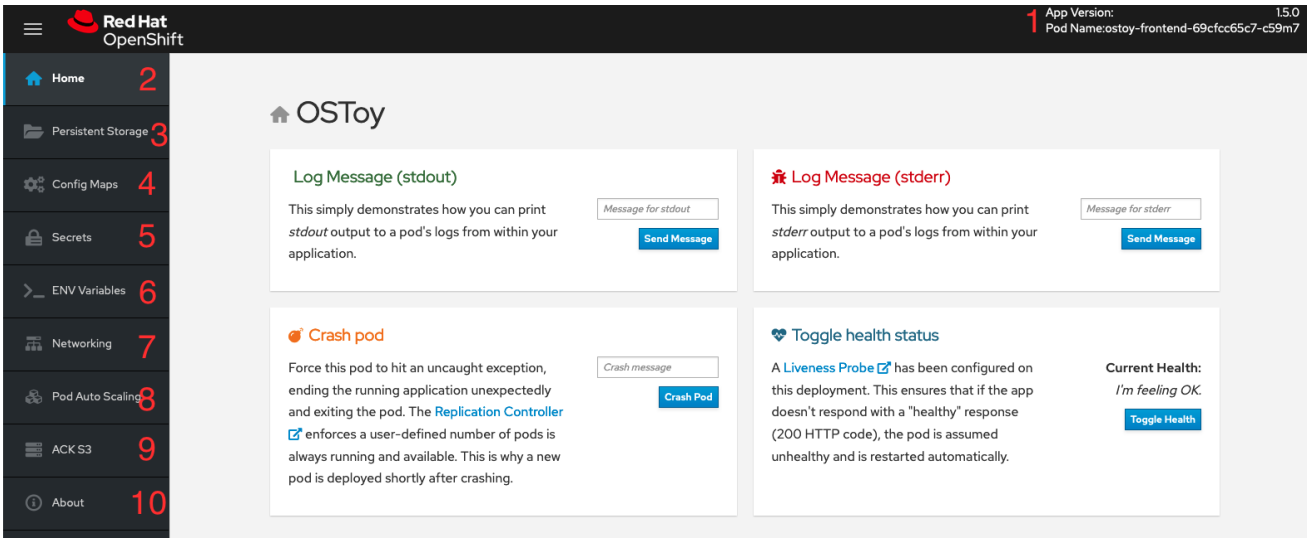
- *增加负载，以查看 pod 的自动扩展，以使用 Horizontal Pod Autoscaler 处理负载。*

- *可选：连接到 AWS S3 存储桶以读取和写入对象。*

16.3.1.3. OSToy Application 图表



16.3.1.4. 了解 OSToy UI



1. *显示为浏览器提供页面的 pod 名称。*

2. ***Home：** 执行我们将了解的一些功能的应用程序主页。*

3. ***持久性存储：** 允许您将数据写入绑定到这个应用程序的持久性卷。*

4. *Config Map：显示应用和 key:value 对可用的 configmaps 的内容。*

5. *secrets：显示应用可用的 secret 的内容，以及 key:value 对。*

6. *ENV 变量 ：显示应用程序可用的环境变量。*

7. *网络：用于说明应用程序内网络的工具。*

8. *Pod 自动扩展：用于增加 pod 的负载并测试 HPA 的工具。*

9. *ACK S3: 可选：与 AWS S3 集成，以读取和写入对象到存储桶。*

注意

要查看 OSToy 的"ACK S3"部分，您必须完成此研讨会的 ACK 部分。如果您决定不完成该部分，则 OSToy 应用程序仍将正常工作。

10. *关于 ：显示应用程序的更多信息。*

16.4. 教程：部署应用程序

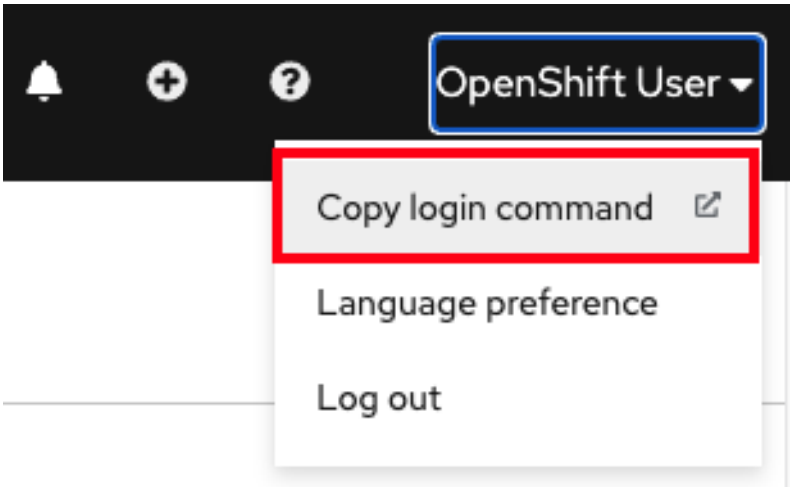
16.4.1. 使用 Kubernetes 部署 OSToy 应用程序

您可以通过在镜像存储库中创建并存储前端和后端微服务容器的镜像，以部署 OSToy 应用。然后，您可以创建 Kubernetes 部署来部署应用程序。

16.4.1.1. 检索 login 命令

1. *如果没有登录到 CLI，使用 Web 控制台访问集群。*

2. *单击右上角的登录名称旁边的下箭头，然后选择 Copy Login Command。*



此时会打开一个新标签页。

3. 选择您的验证方法。

4. 单击 **Display Token**。

5. 使用此令牌将命令复制到 **Log in** 下。

6. 在终端中粘贴并运行复制的命令。如果登录成功，您会看到以下确认信息：

```
$ oc login --token=<your_token> --server=https://api.osd4-
demo.abc1.p1.openshiftapps.com:6443
Logged into "https://api.myrosacluster.abcd.p1.openshiftapps.com:6443" as "rosa-
user" using the token provided.

You don't have any projects. You can try to create a new project, by running

oc new-project <project name>
```

16.4.1.2. 创建新项目

16.4.1.2.1. 使用 CLI

1. 运行以下命令，在集群中创建一个名为 **ostoy** 的新项目：

```
$ oc new-project ostoy
```

输出示例

> *Now using project "ostoy" on server "https://api.myrosacluster.abcd.p1.openshiftapps.com:6443".*

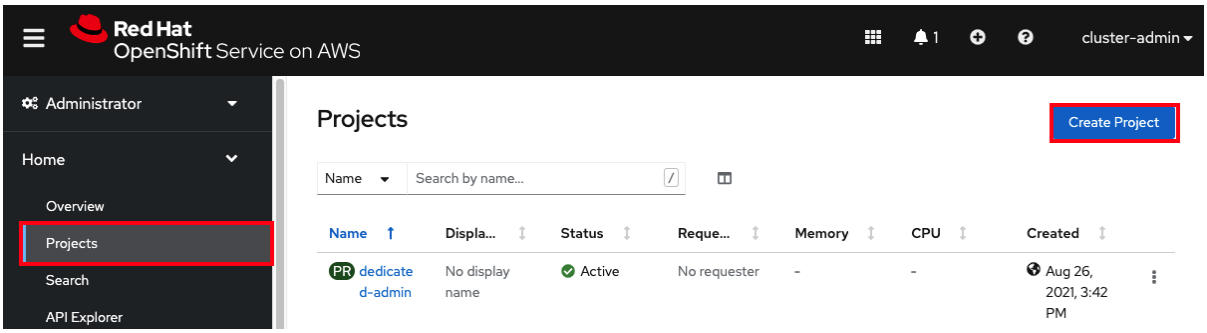
2. *可选 ：或者，运行以下命令来创建唯一的项目名称：*

> *$ oc new-project ostoy-$(uuidgen | cut -d - -f 2 | tr '[:upper:]' '[:lower:]')*

16.4.1.2.2. 使用 Web 控制台

1. *在 Web 控制台中，点击 Home → Projects。*

2. *在 Projects 页面上，单击 Create Project。*



16.4.1.3. 部署后端微服务

微服务为内部 Web 请求提供服务，并返回包含当前主机名和随机生成的颜色字符串的 JSON 对象。

- *通过在终端中运行以下命令来部署微服务：*

> *$ oc apply -f https://raw.githubusercontent.com/openshift-cs/rosaworkshop/master/rosa-workshop/ostoy/yaml/ostoy-microservice-deployment.yaml*

输出示例

```
$ oc apply -f https://raw.githubusercontent.com/openshift-
cs/rosaworkshop/master/rosa-workshop/ostoy/yaml/ostoy-microservice-
deployment.yaml
deployment.apps/ostoy-microservice created
service/ostoy-microservice-svc created
```

16.4.1.4. 部署前端服务

前端部署将 Node.js 前端用于应用和其他 Kubernetes 对象。

ostoy-frontend-deployment.yaml 文件显示前端部署定义了以下功能：

- 持久性卷声明
- Deployment 对象
- Service
- Route
- configmaps
- Secrets
  - 输入以下命令部署应用程序前端并创建所有对象：

```
$ oc apply -f https://raw.githubusercontent.com/openshift-
cs/rosaworkshop/master/rosa-workshop/ostoy/yaml/ostoy-frontend-
deployment.yaml
```

输出示例

```
persistentvolumeclaim/ostoy-pvc created
deployment.apps/ostoy-frontend created
service/ostoy-frontend-svc created
route.route.openshift.io/ostoy-route created
configmap/ostoy-configmap-env created
secret/ostoy-secret-env created
configmap/ostoy-configmap-files created
secret/ostoy-secret created
```

您应该会看到所有创建成功的对象。

16.4.1.5. 获取路由

您必须获取访问应用程序的路由。

- *运行以下命令，获取到应用程序的路由：*

```
$ oc get route
```

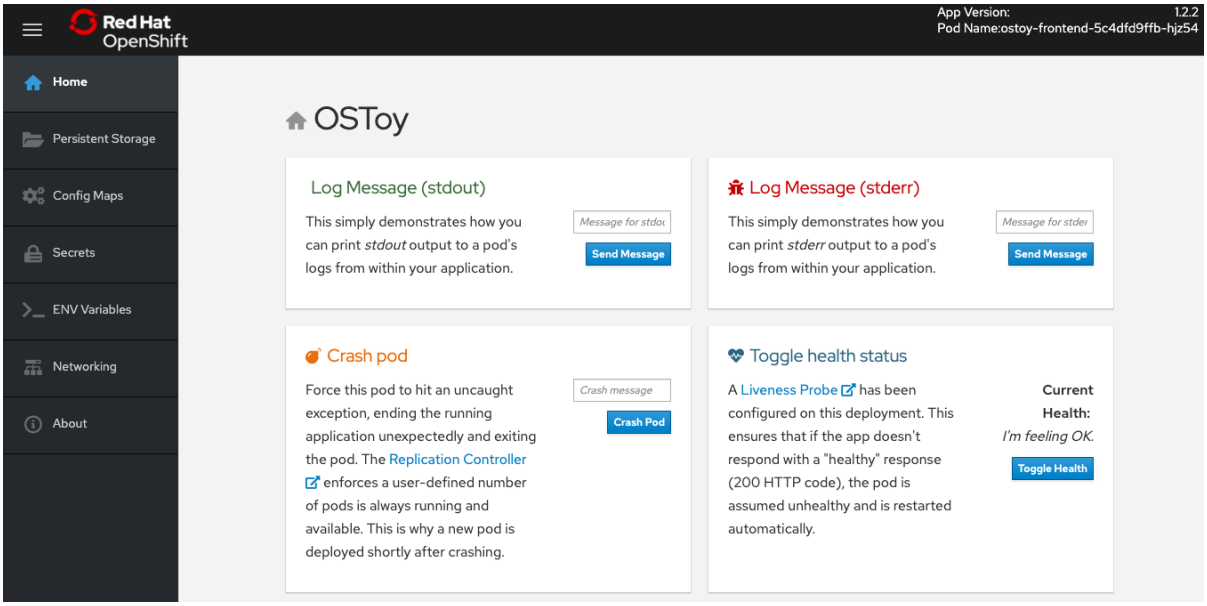
输出示例

```
NAME          HOST/PORT                                                    PATH   SERVICES          PORT
TERMINATION   WILDCARD
ostoy-route   ostoy-route-ostoy.apps.<your-rosa-cluster>.abcd.p1.openshiftapps.com
ostoy-frontend-svc   <all>                 None
```

16.4.1.6. 查看应用程序

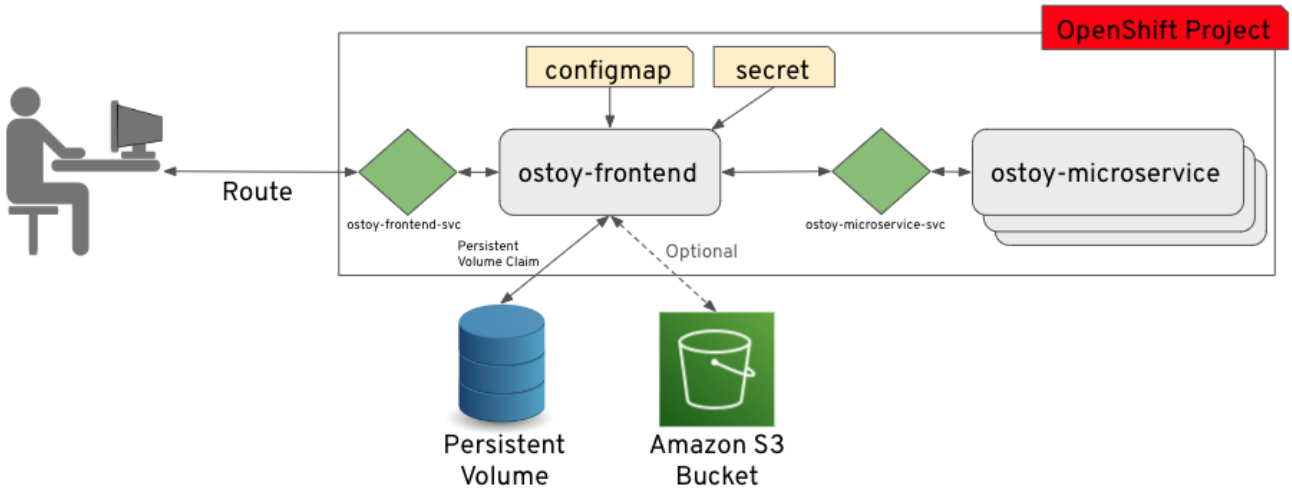
1. *复制上一步中的 ostoy-route-ostoy.apps.<your-rosa-cluster>.abcd.p1.openshiftapps.com URL 输出。*

2. *将复制的 URL 粘贴到网页浏览器中，然后按 Enter 键。您应该会看到应用程序的主页。如果页面没有加载，请确保使用 http 而不是 https。*



16.5. 教程：网络

本教程演示了 OSToy 应用程序如何使用集群内联网来通过微服务和视觉化 pod 扩展来分隔功能。



图中显示至少有两个独立的 pod，每个 pod 都有自己的服务。

一个 pod 充当前端 Web 应用，服务和一个公开访问的路由。其他容器集充当 backend 微服务与服务对象，以便前端容器集能够与微服务通信。如果多个 pod，此通信发生在 pod 之间。由于这些通信限制，此微服务无法从此集群外部或其他命名空间或项目访问（如果已配置）。此微服务的唯一用途是提供内部 Web 请求，并返回包含当前主机名（即 pod 的名称）的 JSON 对象，以及随机生成的颜色字符串。此颜色字符串用于显示方框，其中颜色显示在标题为"Intra-cluster Communication"的标题中。

有关网络限制的更多信息，请参阅关于网络策略。

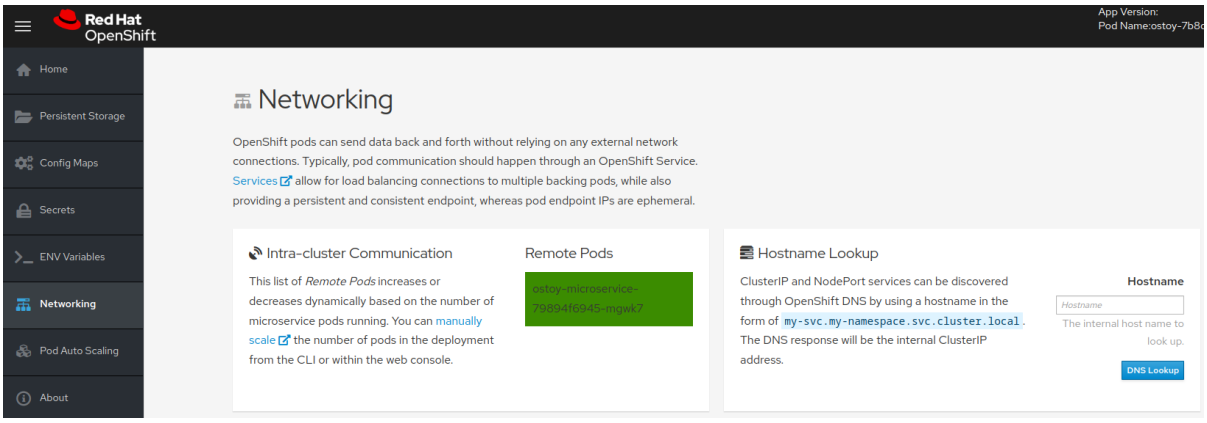
16.5.1. 集群内网络

您可以在 OSToy 应用程序中查看您的网络配置。

流程

1. 在 OSToy 应用程序中，点击左侧菜单中的 Networking。

2. 检查网络配置。名为"Hostname Lookup"的右侧标题演示了如何使用为 pod 创建的服务名称转换为内部 ClusterIP 地址。



3. 输入在正确的标题("Hostname Lookup")中创建的微服务名称，其格式为 <service_name>.<namespace>.svc.cluster.local。您可以运行以下命令来在 ostoy-microservice.yaml 的服务定义中找到此服务名称：

```
$ oc get service <name_of_service> -o yaml
```

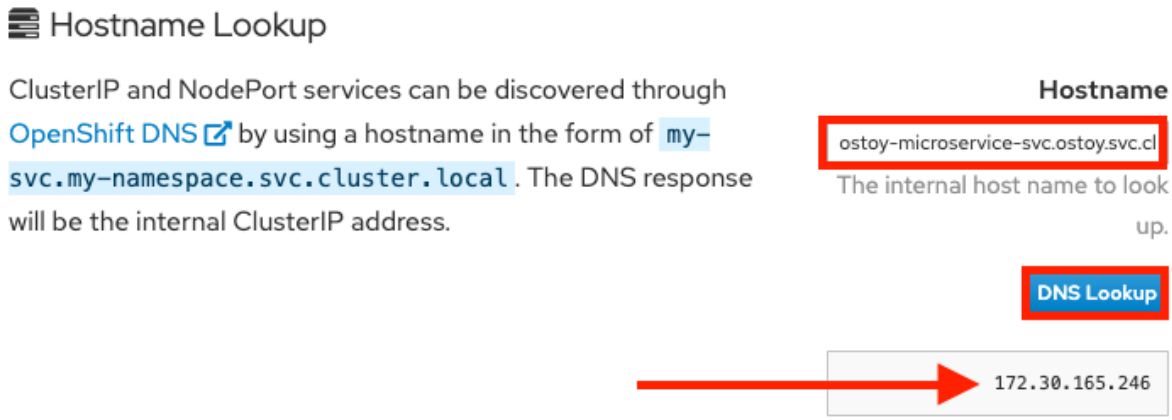
输出示例

```
apiVersion: v1
kind: Service
metadata:
  name: ostoy-microservice-svc
  labels:
    app: ostoy-microservice
spec:
  type: ClusterIP
  ports:
    - port: 8080
      targetPort: 8080
      protocol: TCP
  selector:
    app: ostoy-microservice
```

-

在本例中，完整主机名为 ostoy-microservice-svc.ostoy.svc.cluster.local。

4. *您会看到一个 IP 地址返回。在本例中，它是 172.30.165.246。这是集群内 IP 地址，只能从集群内部访问。*



16.6. 教程：集群存储的持久性卷

Red Hat OpenShift Service on AWS (ROSA)（经典架构）和 Red Hat OpenShift Service on AWS (ROSA)支持使用 Amazon Web Services (AWS) Elastic Block Store (EBS) 或 AWS Elastic File System (EFS) 存储持久性卷。

16.6.1. 使用持久性卷

使用以下步骤创建文件，将其存储在集群中的持久性卷中，并确认它在 pod 故障和重新创建后仍然存在。

16.6.1.1. 查看持久性卷声明

1. *导航到集群的 OpenShift Web 控制台。*

2. *点左侧菜单中的 Storage，然后点 PersistentVolumeClaims 以查看所有持久性卷声明的列表。*

3. 点持久性卷声明来查看大小、访问模式、存储类和其他声明详情。

注意

访问模式为 ReadWriteOnce (RWO)。这意味着卷只能挂载到一个节点，pod 或 pod 可以读取和写入卷。

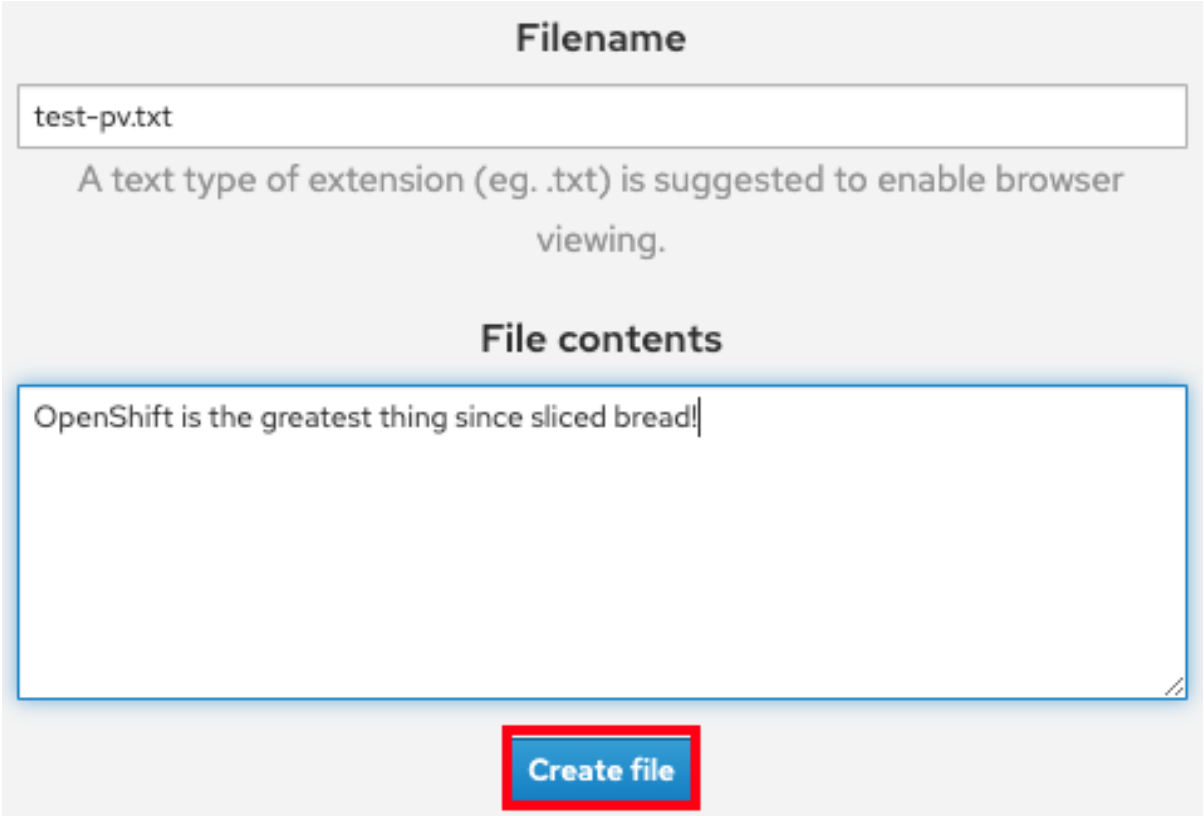
16.6.1.2. 存储您的文件

1. 在 OSToy app 控制台中，点左侧菜单中的 Persistent Storage。

2. 在 Filename 框中，输入带有 .txt 扩展名的文件名，如 test-pv.txt。

3. 在 File content 框中，输入文本句子，例如 OpenShift 是自分片 bread! 起的最大事情。

4. 点 Create file。



5. 滚动到 OSToy 应用程序控制台上的现有文件。

6. 点您创建的文件，以查看文件名和内容。

16.6.1.3. 崩溃 pod

1. 在 OSToy app 控制台中，点击左侧菜单中的 Home。

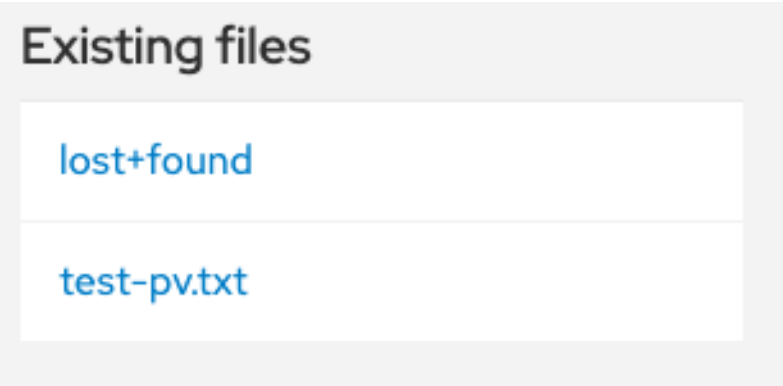
2. 单击 Crash pod。

16.6.1.4. 确认持久性存储

1. 等待 pod 重新创建。

2. 在 OSToy app 控制台中，点左侧菜单中的 Persistent Storage。

3. 查找您创建的文件，并打开该文件来查看并确认内容。



验证

部署 YAML 文件显示我们将 目录 /var/demo_files 挂载到我们的持久性卷声明中。

1. 运行以下命令，检索前端 pod 的名称：

■

```
$ oc get pods
```

2. 运行以下命令，在容器内启动安全 shell (SSH)会话：

```
$ oc rsh <pod_name>
```

3. 运行以下命令来进入目录：

```
$ cd /var/demo_files
```

4. 可选：运行以下命令来查看您创建的所有文件：

```
$ ls
```

5. 运行以下命令，打开该文件以查看内容：

```
$ cat test-pv.txt
```

6. 验证输出是否为您在 OSToy 应用程序控制台中输入的文本。

终端示例

```
$ oc get pods
NAME                              READY    STATUS    RESTARTS   AGE
ostoy-frontend-5fc8d486dc-wsw24       1/1      Running   0          18m
ostoy-microservice-6cf764974f-hx4qm   1/1      Running   0          18m

$ oc rsh ostoy-frontend-5fc8d486dc-wsw24

$ cd /var/demo_files/

$ ls
lost+found   test-pv.txt

$ cat test-pv.txt
OpenShift is the greatest thing since sliced bread!
```

16.6.1.5. 结束会话

- 在终端中键入 exit 退出会话并返回 CLI。

16.6.2. 其他资源

- 如需有关持久性卷存储的更多信息，请参阅了解持久性存储。
- 有关 ROSA 存储选项的更多信息，请参阅 存储概述。