Ansible on Clouds 2.3

Red Hat Ansible Automation Platform from GCP Marketplace Guide

Install and configure Red Hat Ansible Automation Platform from GCP Marketplace

Last Updated: 2023-11-14

Ansible on Clouds 2.3 Red Hat Ansible Automation Platform from GCP Marketplace Guide

Install and configure Red Hat Ansible Automation Platform from GCP Marketplace

Abstract

Thank you for your interest in Red Hat Ansible Automation Platform from GCP Marketplace. Ansible Automation Platform is a commercial offering that helps teams manage complex multi-tier deployments by adding control, knowledge, and delegation to Ansible-powered environments. This guide helps you to understand the installation and use of Ansible Automation Platform from GCP Marketplace.

Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright’s message .

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

If you have a suggestion to improve this documentation, or find an error, please contact technical support at https://access.redhat.com to create an issue on the Ansible Automation Platform Jira project using the **docs-product** component.

IMPORTANT

Disclaimer: Links contained in this document to external website(s) are provided for convenience only. Red Hat has not reviewed the links and is not responsible for the content or its availability. The inclusion of any link to an external website does not imply endorsement by Red Hat of the website or their entities, products or services. You agree that Red Hat is not responsible or liable for any loss or expenses that may result due to your use of (or reliance on) the external site or content.

CHAPTER 1. INTRODUCTION TO ANSIBLE AUTOMATION PLATFORM FROM GCP MARKETPLACE

Ansible Automation Platform from GCP Marketplace is an offering that you can deploy from the GCP Marketplace portal. Ansible Automation Platform from GCP Marketplace provides access to a library of Ansible content collections, and is integrated with key GCP services, so you can start automating the deployment, configuration, and management of infrastructure and applications quickly.

The following Red Hat Ansible Automation Platform components are available on Ansible Automation Platform from GCP Marketplace:

- Automation controller
- Ansible automation hub
- Private automation hub
- Ansible Content Collections
- Automation execution environments
- Ansible content tools, including access to Red Hat Insights for Red Hat Ansible Automation Platform

NOTE

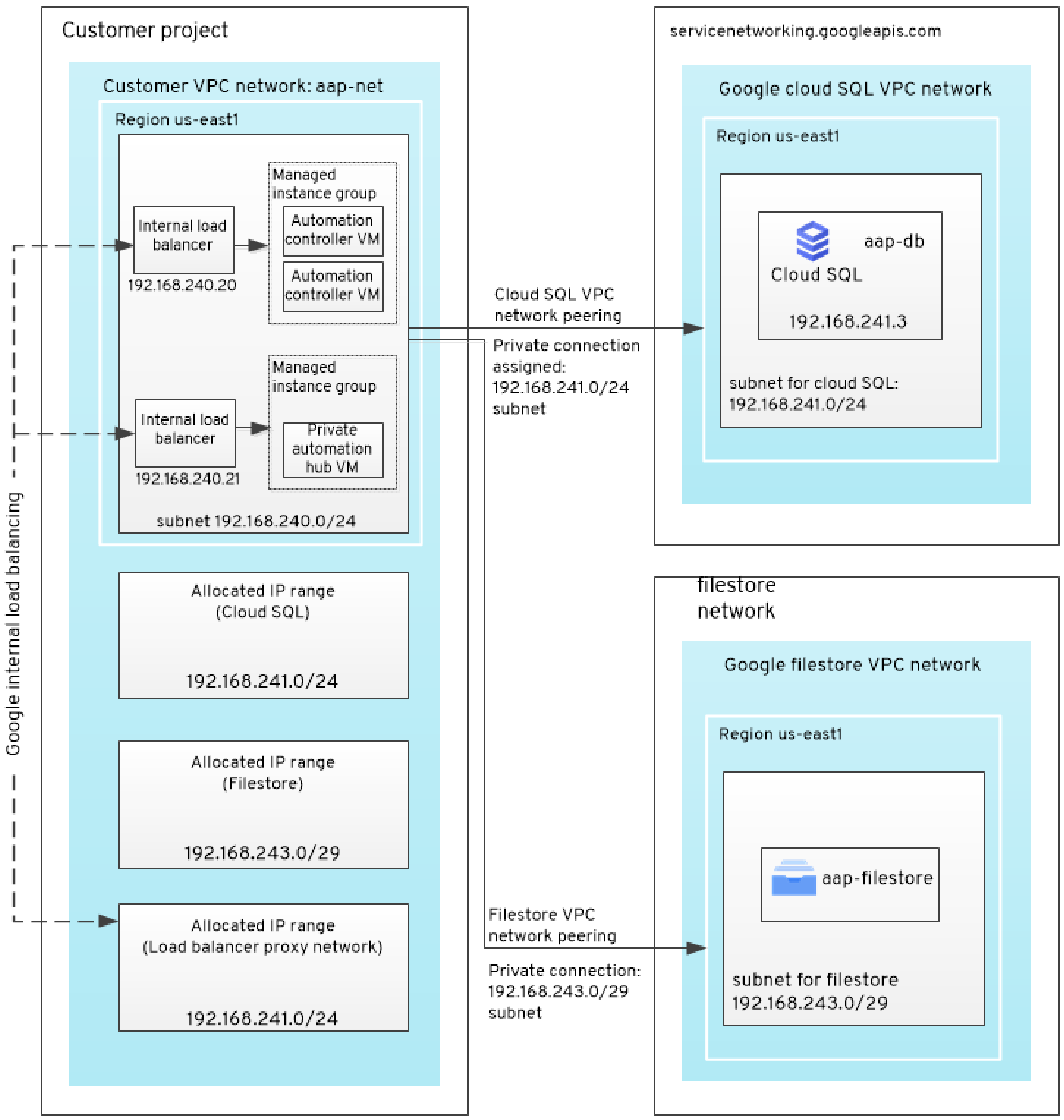
Automation mesh is not available on Ansible Automation Platform from GCP Marketplace at this time.

1.1. APPLICATION ARCHITECTURE

Red Hat Ansible Automation Platform from GCP Marketplace is installed into infrastructure resources running within your GCP account.

There are four virtual machines included in this product listing. Three n2-standard-2 virtual machines make up the permanent compute components of the solution. Additionally, a single ephemeral e2-medium instance is used to run Red Hat Ansible Automation Platform and Google Cloud deployment workloads. This virtual machine is only used during deployment and then is permanently removed from the solution. The infrastructure cost for the ephemeral VM is charged at the hourly rate during the period that the VM exists, usually less than an hour.

1.1.1. GCP infrastructure


Customer project
Customer VPC network: aap-net
Region us-east1
Internal load balancer
192.168.240.20
Managed instance group
Automation controller VM
Automation controller VM
Managed instance group
Internal load balancer
192.168.240.21
Private automation hub VM
subnet 192.168.240.0/24
Allocated IP range (Cloud SQL)
192.168.241.0/24
Allocated IP range (Filestore)
192.168.243.0/29
Allocated IP range (Load balancer proxy network)
192.168.241.0/24
Google internal load balancing
Cloud SQL VPC network peering
Private connection assigned: 192.168.241.0/24 subnet
servicenetworking.googleapis.com
Google cloud SQL VPC network
Region us-east1
aap-db
Cloud SQL
192.168.241.3
subnet for cloud SQL: 192.168.241.0/24
Filestore VPC network peering
Private connection: 192.168.243.0/29 subnet
filestore network
Google filestore VPC network
Region us-east1
aap-filestore
subnet for filestore 192.168.243.0/29

1.1.2. Architecture Description

Ansible Automation Platform from GCP Marketplace was designed to be private, with no public ingress enabled by default. This requires customers to connect existing GCP infrastructure to Ansible Automation Platform infrastructure manually. Ansible Automation Platform components that are deployed are isolated into their own VPC. Applications then run on instances managed by an instance group, which run on RHEL-based Google Compute machine images. These machine images are preloaded with required container images to run Ansible Automation Platform including automation hub, automation controller, and receptor rpm/container components. A shared Google Filestore volume is mounted into each instance deployed by the instance group manager.

All GCP Compute instances have private IP addresses by default. The Ansible Automation Platform applications are run in containers managed through podman on the instances. This podman configuration is itself managed by systemd. SELinux is enabled on GCP Compute instances and is supported down to the container level. Each automation controller instance is configured as an Ansible Automation Platform hybrid node by default to simplify the scale up and out procedures.

You cannot restart virtual machines, instead virtual machines must be replaced. You cannot configure on virtual machines, because any configuration is lost when virtual machines are replaced. This always happens on upgrade.

1.1.3. Service descriptions

Service Name	Description
Compute Engine	GCP VM compute platform
Cloud SQL	GCP database service
Filestore	GCP file storage service
Virtual Private Cloud (VPC)	GCP networking service
Cloud Monitoring	GCP metrics collector
Cloud Logging	GCP log management Service

CHAPTER 2. INSTALLING RED HAT ANSIBLE AUTOMATION PLATFORM FROM GCP MARKETPLACE

2.1. PREREQUISITES

Before you can install and register Ansible Automation Platform, you must be familiar with GCP including how services operate, how data is stored, and any privacy implications that may exist by using these services. You must also set up an account with *Google Cloud Platform* (GCP).

You must have a working knowledge of the following aspects of Google Cloud Platform:

- Deploying solutions from the GCP Marketplace
- Compute engine *Virtual machine* (VM) instances
- Cloud SQL for PostgreSQL
- Filestore
- GPC *Virtual Private Clouds* (VPCs)
    - Subnets
    - Route Tables
    - Load Balancers
- Network Design
    - Hub-and-spoke networking designs
    - VPC Peering
    - *Class Inter-Domain Routing* (CIDR) blocks
    - Transit routing
- GCP Cloud monitoring
    - GCP Ops agent
- SSH

For more information about Google Cloud Platform and terminology, see the GCP product documentation.

2.2. PERMISSIONS

Your GCP account must have the following permissions to successfully create an Ansible Automation Platform deployment on *Google Cloud Platform*.

- Cloud SQL Permissions
    - cloudsql.databases.create
    - cloudsql.databases.delete

  - cloudsql.databases.get
  - cloudsql.databases.list
  - cloudsql.instances.connect
  - cloudsql.instances.create
  - cloudsql.instances.delete
  - cloudsql.instances.get
  - cloudsql.instances.list
  - cloudsql.instances.login
- Compute Permissions
  - compute.addresses.create
  - compute.addresses.get
  - compute.addresses.list
  - compute.addresses.use
  - compute.forwardingRules.create
  - compute.forwardingRules.get
  - compute.globalAddresses.get
  - compute.globalOperations.get
  - compute.instances.delete
  - compute.instances.get
  - compute.instances.getGuestAttributes
  - compute.instances.list
  - compute.projects.get
  - compute.projects.setCommonInstanceMetadata
  - compute.regionBackendServices.create
  - compute.regionBackendServices.get
  - compute.regionBackendServices.update
  - compute.regionBackendServices.use
  - compute.regionHealthChecks.create
  - compute.regionHealthChecks.get

    - compute.regionHealthChecks.useReadOnly
    - compute.regionOperations.get
    - compute.regionTargetHttpProxies.create
    - compute.regionTargetHttpProxies.get
    - compute.regionTargetHttpProxies.use
    - compute.regionUrlMaps.create
    - compute.regionUrlMaps.get
    - compute.regionUrlMaps.use
    - compute.regions.list
    - compute.zoneOperations.get
- Deployment Manager Permissions
    - deploymentmanager.deployments.create
    - deploymentmanager.deployments.delete
    - deploymentmanager.deployments.get
    - deploymentmanager.deployments.list
    - deploymentmanager.manifests.get
    - deploymentmanager.operations.get
    - deploymentmanager.resources.list
- IAM Permissions
    - iam.serviceAccounts.actAs
- Logging Permissions:
    - logging.logEntries.create
- Monitoring Permissions:
    - monitoring.metricDescriptors.get
    - monitoring.timeSeries.create
- Resource Manager Permissions
    - resourcemanager.projects.get
- Runtime Configurator Permissions
    - runtimeconfig.configs.list
    - runtimeconfig.variables.create

- runtimeconfig.variables.get
- runtimeconfig.variables.list
- runtimeconfig.variables.update

- Secret Manager Permissions
  - secretmanager.secrets.create
  - secretmanager.secrets.delete
  - secretmanager.secrets.get
  - secretmanager.secrets.list
  - secretmanager.versions.access
  - secretmanager.versions.add
  - secretmanager.versions.destroy
  - secretmanager.versions.disable
  - secretmanager.versions.get
  - secretmanager.versions.list
- Service Networking Permissions
  - servicenetworking.operations.get
  - servicenetworking.services.addPeering
- Service Usage Permissions
  - serviceusage.services.list

2.3. CREATE A PROJECT

To install Ansible Automation Platform, you must create a project in your Google Cloud Platform account to host the application if you do not already have one. See Creating and managing projects in the GCP documentation.

2.3.1. Required APIs

Your Google Cloud Platform (GCP) project requires access to several API services to complete Ansible Automation Platform installation. On marketplace deployment, the process automatically attempts to enable the following APIs.

If you prefer, you can enable the APIs ahead of time to ensure they are permitted by your organization.

API Service	Console service name
Compute Engine API	**compute.googleapis.com**

API Service	Console service name
Google Cloud APIs	**cloudapis.googleapis.com**
Identity and Access Management (IAM) API	**iam.googleapis.com**
Cloud SQL Admin API	**sql-component.googleapis.com**
Cloud Logging API	**logging.googleapis.com** See Monitoring and logging
Cloud Monitoring API	**monitoring.googleapis.com** See Monitoring and logging
Cloud Resource Manager API	**cloudresourcemanager.googleapis.com**
Cloud Identity-Aware Proxy API	**iap.googleapis.com**
Secret Manager API	**secretmanager.googleapis.com**
Service Networking API	**servicenetworking.googleapis.com**
Service Usage API	**serviceusage.googleapis.com**
OS Config API	**osconfig.googleapis.com**
Cloud Runtime Configuration API	**runtimeconfig.googleapis.com**
Cloud Filestore API	**file.googleapis.com**

2.3.2. Your service account

You must have a service account to set up Ansible Automation Platform from GCP Marketplace. This account is used to install and configure the application and remains associated with the Ansible Automation Platform virtual machines. You can use an existing service account with the required roles, or the Ansible Automation Platform deployment can create one with the required roles on your behalf. If you use an existing account, or create a service account in advance, it must have the following roles:

- Editor
- Logs Writer
- Cloud SQL Client
- Cloud SQL Instance User
- Secret Manager Secret Accessor

- Compute Network Admin

See Grant a single role for steps to add the required roles to your existing service account.

The Compute Network Administrator role is only required at the time of deployment to configure the application properly. When installation and configuration are complete, this role can be removed from the service account, which remains configured on the Ansible Automation Platform Virtual Machines.

NOTE

The secrets created during the deployment must be able to use the automatic replication policy. The user specified replication policy is not supported and must be disabled during deployment, backup or upgrade.

2.4. APPLICATION DEPLOYMENT

To launch your offer on the Google Cloud Console, navigate to the marketplace and search for **Red Hat Ansible Automation Platform 2 - Up to 100 Managed Nodes**. After selecting this offer click **Launch**.

There are four virtual machines included in this product listing. Three n2-standard-2 virtual machines make up the permanent compute components of the solution. Additionally, a single ephemeral e2-medium instance is used to run Red Hat Ansible Automation Platform and Google Cloud deployment workloads. This virtual machine is only used during deployment and then is permanently removed from the solution. The infrastructure cost for the ephemeral VM is charged at the hourly rate during the period that the VM exists, usually less than an hour.

IMPORTANT

A temporary yet necessary constraint has been placed on the length of deployment names. This is due to GCP’s naming scheme for internal components that make up an Ansible Automation Platform deployment. Component names based on this naming scheme can get too long and often break naming constraints imposed by other services, creating deployment errors.

The length of the name of your deployment, plus the length of your GCP project name must be less than 35 characters and the length of the deployment name must be less than 30 characters.

The calculation below will help you find the maximum length of the name of an Ansible Automation Platform deployment in your project.

length of deployment name < = (minimum between 30 and 35) - length(gcp project name)

There are two methods for deploying the application:

2.4.1. Deploying an application with a new VPC

NOTE

The process of deploying the application using a new VPC has been deprecated, and the functionality will be removed from Ansible Automation Platform from GCP Marketplace in a future release.

This procedure creates a new VPC network and deploys the application in the created VPC.

Procedure

1. In the Deployment page, select the **Service Usage API** link below the **Confirm Service Usage API is enabled** checkbox.
2. In the **API/Service Details** tab, ensure that the API is enabled, then return to the Deployment page.
3. Check the **Confirm Service Usage API** is enabled checkbox.
4. Select or create a **Service Account**. For further information see Your service account.
5. In the **Region** field, select the region where you want the application deployed.
6. In the **Zone** field, select the zone where you want the Filestore deployed. The zone must be in the Region you selected.
7. In the **Observability** section, you can enable logging and metrics to be sent to Cloud Logging and Cloud Monitoring. See Operations Suite Pricing for the financial cost of enabling these services. See See Monitoring and logging for more information on configuring this feature.
8. In the **Network Selection** section, select **New network**. The Networking section provides defaults for all of the network ranges used in the deployment. If you want to modify these values, see Networking Options.
9. Click **DEPLOY**.
10. The Deployment Manager displays the running deployment.
11. The application begins provisioning. It can take some time for the infrastructure and application to fully provision.

NOTE

You will see a warning on the deployment

> This deployment has resources from the Runtime Configurator service, which is in Beta

This warning is expected and is not a cause for concern.

2.4.2. Deploying an application with an existing VPC

The following procedure uses an existing VPC network to deploy an application.

NOTE

The existing proxy subnet must be of type **Regional Managed Proxy**, which is the reserved proxy-only subnet for load balancing.

Procedure

1. In the **Deployment** page, select the **Service Usage API** link below the **Confirm Service Usage API is enabled** checkbox.
2. In the **API/Service Details** tab, ensure that the API is enabled, then return to the **Deployment** page.
3. Check the **Confirm Service Usage API** is enabled checkbox.
4. Select or create a **Service Account**. For further information, see Your service account.
5. In the **Region** field, select the region where you want the application deployed.
6. In the **Zone** field, select the zone where you want the Filestore deployed. The zone must be in the Region you selected.
7. In the **Observability** section, you can enable logging and metrics to be sent to Cloud Logging and Cloud Monitoring. See Operations Suite Pricing for the financial cost of enabling these services. See Monitoring and logging for more information on configuring this feature.
8. In the **Network Selection** section, select **Existing network**.
9. In the **Existing Network** section, provide your existing VPC network name, existing subnet name and existing proxy subnet name. Select **cloud NAT router** to create a **NAT router** in your VPC network.
10. The **Networking section** provides defaults for all of the network ranges used in the deployment. Provide these values based on your existing network configuration. If you want to modify these values, see Networking Options
11. Click **DEPLOY**.
12. The **Deployment Manager** displays the running deployment.
13. The application begins provisioning. It can take some time for the infrastructure and application to fully provision.

NOTE

You will see a warning on the deployment.

> This deployment has resources from the Runtime Configurator service, which is in Beta.

This warning is expected and is not a cause for concern.

2.4.3. Retrieving the administration password

Use the following procedure to retrieve the administration password.

Procedure

1. In the GCP UI, select the main menu from the top left.
2. Select **Security**. If you do not see **Security**, select **View All Products**.
3. Select **Secret Manager**.

4. Filter with the name of the deployment. The secret name format is **<DeploymentName>-aap-admin**.

5. Click on the secret name of the deployment

6. Click the **More Actions** icon * ⋮ on the line of the deployment.

7. Select **View secret value**. The administration password is displayed

2.4.4. Retrieving the controller and hub load balancer IP address

Use the following procedure to retrieve the controller and hub IP address.

Procedure

1. In the GCP UI, select the main menu from the top left.

2. Select **Deployment Manager**.

3. Select **Deployment**.

4. Select the deployment name.

5. Select **View Details**.

6. In the right pane, under **Deployment properties**, find the **Layout** line

7. Select **View**. The **outputs:** section shows the **finalValue** for the **name** controllerIp and hubIp.

2.5. DEPLOYING AN EXTENSION NODE

You configure extension nodes after you have purchased and launched an extension from the public or private offer.

Procedure

1. In the **Deployment name** field, enter a sufficiently short name as described in Application deployment.

2. For the Service Account, select the service account used with your Red Hat Ansible Automation Platform with up to 100 Managed Nodes deployment.

3. In the **Region** field, select the region where you want the application deployed.

4. In the **Zone** field, select a zone in the **Region** you selected when deploying your foundational offer. This field is only used to filter the Network list.

5. In the **Main Deployment Name** field, enter the foundation deployment name for which you are deploying an extension.

6. In the **Networking** section, expand the default option.

7. In the **Network** field, select the existing foundation network ending with **-aap-net**.

8. In the **Subnetwork** field, select the existing foundation subnetwork ending with **-aap-subnet**.

IMPORTANT

Do not select the subnet ending with **-aap-prxy-subnet**.

9. Ensure **Extend Ansible Automation Platform deployment in selected VPC**is checked.
10. Click **DEPLOY**.
11. The Deployment Manager displays the running deployment.

The extension begins provisioning.

It can take some time for the infrastructure and extension to fully provision.

2.6. NETWORKING OPTIONS

The network topology of Ansible Automation Platform from GCP Marketplace includes several configurable network segments that can be changed to fit into your organization's network requirements.

The deployment creates a new VPC and subnet that are not accessible from the public internet, or by using an existing VPC network. You must provide access to the application as described in Networking and Application Access.

Consider your existing network configurations when specifying the following network ranges below. Ensure that each range does not overlap with any other specified here, and does not overlap with existing ranges in your network. Each range should exist within a private network class.

Application Subnet Range

The network CIDR defining the subnet range used by the custom VPC deployed by the offering. Must be a minimum /**24** segment and must be in the private network range (192.168 or 10.0).

Default: 192.168.240.0/24

Cloud SQL Peering Network Range

The network CIDR defines the network segment used to peer the GCP CloudSQL network with the application subnet deployed by the offering. Must be a /**24** segment. The /**24** segment range is a requirement of GCP CloudSQL network peering configuration.

Default: 192.168.241.0/24

Filestore Peering Network Range

Ansible Automation Platform from GCP Marketplace uses GCP Filestore service to share configuration files between multiple automation controller and automation hub VMs provisioned as part of the deployment. This network CIDR range defines the peer network that is used by the offering to peer between the GCP Filestore network and the custom VPC application subnet of the offering. Must be a minimum /**29** segment.

Default: 192.168.243.0/29

Load Balancer Proxy Subnet Range

Ansible Automation Platform from GCP Marketplace is deployed using GCP’s native cloud capabilities to provide a scalable and reliable installation. As part of the Ansible Automation Platform from GCP Marketplace deployment topology two load balancers are deployed in front of the automation hub and automation controller VMs. All traffic is directed at these load balancers and is proxied to the available backend VMs. The deployment takes advantage of GCP’s native load balancing support enabling the customer to add additional ports (https) to the load balancers to capture and forward requests onto the backend VMs. This also provides request balancing and session tracking for better reliability. As part of the load balancer deployment, GCP requires creation of a special proxy network where GCP natively handles the redirection of requests to the backend VMs. This special proxy network is not used within Ansible Automation Platform from GCP Marketplace for any other purpose than GCP’s load balancer’s proxy network requirement. A /**24** segment is required.

Default: 192.168.242.0/24

Controller Internal Load Balancer IP Address

This is the static IP Address assigned to the automation controller Load Balancer. This address must be within the Application Subnet Range segment.

Default: 192.168.240.20

Hub Internal Load Balancer IP Address

This is the static IP Address assigned to the automation hub Load Balancer. This address must be within the Application Subnet Range segment.

Default: 192.168.240.21

CHAPTER 3. NETWORKING AND APPLICATION ACCESS

When Ansible Automation Platform from GCP Marketplace is deployed, it is installed into an isolated VPC and cannot be accessed. The following instructions describe how to connect the VPC used by Ansible Automation Platform from GCP Marketplace to your existing GCP network. When connected, you must determine how your users connect to Ansible Automation Platform.

There are many ways to enable this connectivity such as VPNs, Google Cloud Interconnect, or bastion servers for private network access. You can also expose the platform with public internet access using GCP services such as a Load Balancer.

How your organization configures application access on GCP is outside the scope of Red Hat's guidelines and support for Ansible Automation Platform from GCP Marketplace. For further information, see Securely connecting to VMs for guidelines on these topics.

3.1. NETWORK PEERING OPTIONS

Many networking configurations to access the platform are possible, but the following configurations have been validated to work with Ansible Automation Platform from GCP Marketplace.

NOTE

While every effort has been made to align with Google Cloud Platform's documentation for this content, there may be a drift in accuracy over time. Use GCP documentation as the source of information regarding networking topics for GCP.

3.2. VPC PEERING

VPC Peering is required for Ansible Automation Platform to access resources that reside on private VPCs or where transit routing between Google Cloud Platform and your on-premises network(s) exists. It offers the ability to directly connect different networks within your GCP infrastructure. VPCs are individually connected to one another with no other routing hops between them. VPC deployments omit access from the public internet by default.

This is also the deployment model for the GCP architecture used by Ansible Automation Platform from GCP Marketplace.

This is a simple peering model and is useful when connecting several networks.

Complex peering can be configured, but routing can become more complex over time.

When VPC peering and routing are configured, you can access Ansible Automation Platform through a VM on a connected VPC subnet, or directly if your organization has a transit routing setup between GCP and your local networks.

When two or more networks are peered, Google performs automated actions to assist with routing, but can perform routable updates to enable taffic flow within your GCP infrastructure.

Prerequisites

Before using VPC peering to connect any VPC, you must ensure that there is no network address space overlap between the networks that you intend to route traffic between your VPCs and Ansible Automation Platform from GCP Marketplace's VPC address space. The GCP Portal should prevent peering if this is attempted.

Configure VPC peering with Ansible Automation Platform with the following procedure.

Procedure

1. In the GCP Portal, navigate to **VPC Network**.
2. In the **VPC menu**, select **VPC Network Peering**.
3. Click **Create peering connection**.
4. In the **Name** field, enter the name of the peering connection that you need.
5. In the **Your VPC Network** field, select the first VPC that you plan to peer.
6. In the **Peered VPC Network** field, select the second VPC to peer. This must be the Ansible Automation Platform from GCP Marketplace VPC.
   - Subnet routes between these two networks are created by default. Therefore, access to Ansible Automation Platform can occur from VMs that reside on the peered VPC. However, if you have more complex routing, such as a hub-and-spoke network model, then other routes must be created.
   - Select **Exchange custom routes** and **Exchange subnet routes with public IP** carefully. Google provides explanations what each fields does. Your selection affects how traffic flows through your VPCs. Normally, these checkboxes configure routing between the newly peered networks and other networks through the route table exchange and enable network traffic to traverse multiple networks (transit routing).
7. Click **Create**.

For additional information on routing tables, firewalls and other networking components regarding VPC Peering, see the GCP documentation.

3.3. USING AN EXTERNAL LOAD BALANCER

It is possible to expose Ansible automation controller and Ansible private automation hub to the public internet if you want users to have access to the platform from any internet connected machine.

This is not recommended as a best practice for security reasons.

However this implementation can be secured with different GCP tools. This section describes how to connect a load balancer that faces the public internet, but it does not include steps to harden access through Google's security products. Only use this approach if you intend to protect the public endpoints with Google Cloud Armor or similar product(s).

NOTE

Ansible Automation Platform from GCP Marketplace is deployed with two *internal* application load balancers. These load balancers direct traffic within local VPCs and must remain part of the deployment even if you configure public load balancers.

Procedure

1. Select the main menu from the top left.
2. Select **Network Services**. If you do not see **Network Services**, then select **View All Products**.

3. Select **Load Balancing**.
4. In the top menu, select **CREATE LOAD BALANCER**.
5. Select the **HTTP(S) Load Balancing** tile and click **START CONFIGURATION**.
6. Select **From Internet to my VMs or serverless services** and **Global HTTP(s) Load Balancer**.
7. Click **CONTINUE**.
8. Give your load balancer a name, for example, **<DeploymentName>-aap-<cntrlr/hub>-ext-lb**
9. Ensure that **Frontend** configuration is selected at the left of the screen.
10. On the **Frontend** configuration page, complete the following fields:
    - **Protocol**: HTTPS.
    - **Port**: 443.
    - **IP Address**: Select an existing IP address or create a new one.
    - **Certificate**: You can use your own SSL Certificate or use a Google managed certificate.
    - **SSL Policy**: GCP default, or another configured policy.
11. Click **DONE**.
12. Select **Backend configuration** from the menu on the left.
13. Click the **Backend services & backend buckets** drop-down.
14. Click **CREATE A BACKEND SERVICE**.
    a. Give your backend service a name, for example, **<DeploymentName>-aap-<cntrlr/hub>-ext-lb-bknd-svc**
    b. Set **Backend type** to **Instance group**.
    c. Set **Protocol** to **HTTP** and **Named Port** to **http**.
    d. Change **Timeout** to **300**.
    e. Scroll down to the **Backends** section and in the **New backend** field, select the **Instance group** drop-down. Select the correct instance group.
       For the automation controller load balancer, the correct instance group has the suffix **-aap-cntrlr-igm**.

       For the automation hub load balancer, the correct instance group has the suffix **-aap-hub-igm**.
    f. In the resulting dialog box, select **USE EXISTING PORT NAME**.
    g. Set **Balancing mode** to **Rate**.
    h. Set **Maximum RPS** to 300.
    i. Scroll down until you see a text box showing **Health check**.

j. Select the drop-down menu and click **CREATE A HEALTH CHECK**.

k. Enter a name for your health check, for example, **<DeploymentName>-aap-<cntrlr/hub>-ext-lb-hc**

l. For automation controller, use the following health check settings

   - In the **Health Check** dialog box, set **Protocol** to **TCP**, and **Port** to **8052**.
   - Set **Request** to **/api/v2/ping/**.

m. For automation hub, use the following health check settings

   - In the **Health Check** dialog box, set **Protocol** to **TCP**, and **Port** to **8080**.
   - Set **Request** to **/api/galaxy/pulp/api/v3/status/**.
   - Keep all other default settings.

n. Click **SAVE**.
   This returns you to the **Backend services & backend buckets** window.

o. Click **CREATE**.
   This returns you to **Backend configuration** section.

p. Click **OK**.

q. Click **CREATE** to create the load balancer for the automation controller or automation hub UI. This will take a few minutes to complete.

15. A load balancer has now been created.

16. Configure the DNS record for the domain that you used in your SSL certificate to point to the IP address of the load balancer.

At this point you should have accesss to the Ansible Automation Platform automation controller UI. You can log in after you retrieve the administration password.

Repeat the same process for private automation hub. The process is identical except for selecting the instance group **-aap-hub-igm** during backend configuration.

CHAPTER 4. ADDITIONAL CONFIGURATIONS

The following chapter describes Ansible Automation Platform configuration steps that you can perform once your deployment is complete on GCP.

4.1. CHANGING THE DEFAULT ANSIBLE AUTOMATION PLATFORM ADMINISTRATOR PASSWORD

The default administrator password for Ansible Automation Platform is generated randomly when Ansible Automation Platform from GCP Marketplace is deployed. Follow these steps to change the administrator password for both automation controller and automation hub.

Procedure

1. Locate the correct secret for the Ansible Automation Platform deployment by opening the GCP Secrets Manager console and selecting the secret with the name *<deployment_name>-aap-admin*.
2. Select **NEW VERSION** to add a new version.
3. Enter a password secret value.
4. Check **Disable all past versions**.
5. Click **ADD NEW VERSION**.
6. Replace the running Ansible Automation Platform VM instances to use the new administrator password.
   a. Navigate to the **GCP VM Instances** console.
   b. Identify and delete one automation controller VM instance and one automation hub VM instance for the Ansible Automation Platform deployment.
   c. Wait for the automation controller Auto Scale group and the automation hub Auto Scaling group to create new VM instances.
7. The new administrator password can be used when the new automation controller and automation hub VM instances reach a *Running* Instance State.

4.2. SECURING INTERNAL COMMUNICATION WITH SSL

Ansible Automation Platform from GCP Marketplace is deployed with two *internal* application load balancers, one each in front of the hub and controller instances. These internal load balancers must be configured with SSL certificates after deployment completes.

NOTE

Securing traffic through these internal load balancers is different than securing traffic through external load balancers in previous steps. This process ensures HTTP traffic is encrypted even when the traffic is localized to private GCP VPCs. The same procedure can be followed for both the automation controller and automation hub load balancers.

To modify both the automation controller and automation hub load balancers, which have the name format **<DEPLOYMENT_NAME>-aap-<cntrlr/hub>-int-lb**.

Procedure

1. In the GCP console, navigate to the Load Balancing page.
2. In the search bar, enter the name of your deployed image name to filter down to your load balancers.
3. Click **Edit**.
4. Click **Frontend configuration**.
5. Click **ADD FRONTEND IP AND PORT**. Use the following values:
    a. **Protocol**: HTTPS (includes HTTP/2).
    b. **Subnetwork**: Select the available aap-subnet.
    c. **Port**: 443
    d. **IP Address**: **<DEPLOYMENT_NAME>-aap<cntrlr/hub>-intl-lb-ip**
6. If you have already added your certificate, select it.
    a. If you have not added your certificate, click **CREATE A NEW CERTIFICATE**.
    b. Provide a name for your certificate.
    c. Using your previously generated certificate, copy its contents and paste it under **Certificate**.
    d. Using your previously generated certificate key, copy its contents and paste it under **Private Key**.
    e. Click **Create**.
7. Click **Done**.
8. Optional: To delete the HTTP Frontend configuration.
    a. Open the Load balancer instance.
    b. Click **Frontend Configuration** The configurations appear on the left side of the UI.
    c. Scroll to the configuration you want to delete.
    d. Click the trashcan icon to delete the configuration.
9. Click **Update**, and confirm the update.

4.3. SECURITY CONSIDERATIONS

To configure Red Hat Single Sign-On with an identity provider that can enable *Multi factor Authentication* (MFA), follow the steps here for connecting enterprise authentication to Ansible Automation Platform.

Securing infrastructure services is an important step in any cloud deployment. Follow the implementation and security suggestions from GCP documentation for services used as part of an Ansible Automation Platform from GCP Marketplace deployment.

CHAPTER 5. USING THE COMMAND GENERATOR

The Command Generator is used to generate commands for launching operational playbooks provided by the Ansible-on-clouds operational playbook collection.

The process involves five steps:

1. Pull the **ansible-on-clouds-ops** container image.
2. List the available playbooks.
3. Use a command generator to generate a data file and the next command to run. **command_generator_vars** and the command_generator are implemented using a docker container and are run using the docker command line interface.
4. Populate the data file and run the previous generated command. This generates the final command with all parameters.

   NOTE

   When this step is complete, you can save the generated command and run the playbook when it is required.

5. Run the final command.

Prerequisites

- Docker
- A GCP credentials file
- An internet connection to Google Cloud

5.1. PULLING THE ANSIBLE-ON-CLOUDS-OPS CONTAINER IMAGE

Pull the Docker image for the Ansible on Clouds operational container with the same tag version as your deployment.

NOTE

Before Pulling the docker image, make sure you are logged in to registry.redhat.com using docker. Use the following command to login to registry.redhat.com.

```
$ docker login registry.redhat.io
```

For more information about registry login, see Registry Authentication

For example, if your foundation deployment version is 2.3.20230221-00, you must pull the operational image with tag 2.3.20230221.

Use the following commands:

```
$ export IMAGE=registry.redhat.io/ansible-on-clouds/ansible-on-clouds-ops-rhel8:2.3.20230221
$ docker pull $IMAGE --platform=linux/amd64
```

For EMEA regions (Europe, Middle East, Africa) run the following command instead:

```
$ export IMAGE=registry.redhat.io/ansible-on-clouds/ansible-on-clouds-ops-emea-
rhel8:2.3.20230221
$ docker pull $IMAGE --platform=linux/amd64
```

5.2. LISTING THE AVAILABLE PLAYBOOKS

Procedure

1. For the list of available playbooks without details, use the following command.

   ```
   $ docker run --rm $IMAGE command_generator_vars | grep Playbook

   The current version of the operational playbooks collection contains the following playbooks:

   Playbook: aws_add_extension_nodes
   Playbook: aws_backup_stack
   Playbook: aws_deployment_inventory
   Playbook: aws_remove_extension_nodes
   Playbook: aws_restore_stack
   Playbook: aws_upgrade
   Playbook: gcp_aap_health_check
   Playbook: gcp_backup_deployment
   Playbook: gcp_create_external_load_balancer
   Playbook: gcp_delete_external_load_balancer
   Playbook: gcp_deployment_inventory
   Playbook: gcp_health_check
   Playbook: gcp_list_deployments
   Playbook: gcp_nodes_health_check
   Playbook: gcp_restore_deployment
   Playbook: gcp_setup_logging_monitoring
   Playbook: gcp_upgrade
   ```

2. To provide a list of all available playbooks, and to use the command generator, use the following command.

   ```
   $ docker run --rm $IMAGE command_generator_vars
   ```

   This provides a list of playbooks and a command similar to the following:

   ```
   ===============================================
   Playbook: gcp_upgrade
   Description: Performs the upgrade of the Ansible Automation Platform from GCP
   Marketplace components to the latest version.
   -----------------------------------------------
   Performs the upgrade of the Ansible Automation Platform from GCP Marketplace
   components to the latest version.

   -----------------------------------------------
   ```

```
Command generator template:

docker run --rm $IMAGE command_generator gcp_upgrade [--ansible-config
ansible_config_path>] \
-d <deployment_name> -c <cloud_credentials_path> --extra-vars 'gcp_compute_region=
<gcp_compute_region> gcp_compute_zone=<gcp_compute_zone> gcp_backup_taken=
<true|false>'
===============================================
```

5.3. GENERATING THE DATA FILE

Procedure

1. Run the command generator.

```
$ docker run --rm -v <local_directory_data_file>:/data $IMAGE command_generator_vars
<playbook_name> --output-data-file /data/<data-file>.yml
```

The outputs of this command are the command to run and the data file template. The data file is also saved in your **<local_data_file_directory>**. This is the template which you populate with your data.

The following example uses the **gcp_backup_deployment** playbook.

```
$ docker run --rm -v <local_data_file_directory>:/data $IMAGE command_generator_vars
gcp_backup_deployment \
--output-data-file /data/backup.yml
```

2. Producing the following output.

```
===============================================
Playbook: gcp_backup_deployment
Description: This playbook is used to backup the AoC Self-managed GCP environment.
-----------------------------------------------
This playbook is used to backup the AoC Self-managed GCP environment.
For more information regarding backup and restore, visit our official documentation -

-----------------------------------------------
Command generator template:

docker run --rm -v /tmp:/data $IMAGE command_generator gcp_backup_deployment --data-
file /data/backup.yml

Data template:

gcp_backup_deployment:
  cloud_credentials_path:
  deployment_name:
  extra_vars:
    gcp_bucket_backup_name:
    gcp_compute_region:
    gcp_compute_zone:

===============================================
```

5.4. POPULATING THE DATA FILE

Procedure

- Edit the data-file generated in Generating the data file.
  Any attribute that represents a path must be an absolute path. The command_generator automatically mounts a volume for it in the final command.

  For example, in the case of the **gcp_backup_deployment** playbook, the file becomes:

```
gcp_backup_deployment
  cloud_credentials_path: /path/to/credentials
  deployment_name: my-deployment
  extra_vars:
    cp_bucket_backup_name: my-bucket
    gcp_compute_region: us-east1
    gcp_compute_zone: us-east1-b
```

5.5. RUNNING THE GENERATED COMMAND

Procedure

1. Ensure that the mounted volume points to the directory where the data file is.
   For the **gcp_backup_deployment** playbook example, this is:

```
$ docker run --rm -v /tmp:/data $IMAGE command_generator gcp_backup_deployment --
data-file /data/backup.yml
```

   Which generates the following output:

```
Command to run playbook:

$ docker run --rm --env PLATFORM=GCP -v
/path/to/credentials:/home/runner/.gcp/credentials:ro \
--env ANSIBLE_CONFIG=../gcp-ansible.cfg  $IMAGE\
redhat.ansible_on_clouds.gcp_backup_deployment \
-e 'gcp_service_account_credentials_json_path=/home/runner/.gcp/credentials  \
gcp_deployment_name=my-deployment gcp_compute_region=us-east1
gcp_compute_zone=us-east1-b'
```

   This new command has the required parameters, environment variables and mounted volumes to run the playbook.

2. Run the generated command. You can save this command to rerun it later if required.

CHAPTER 6. AUTOMATION WORKLOADS

The default Ansible Automation Platform from GCP Marketplace deployment is designed and licensed to automate 100 managed nodes.

6.1. AUTOMATION PERFORMANCE

Automating against your licensed managed node allocation in the offer is provided with the following operational expectations. Automating outside of the boundaries of these criteria is not supported, but can still function depending on your automation.

Metric	Threshold
Concurrent Jobs	10
Forks per job	10

NOTE

Ansible Automation Platform from GCP Marketplace is driven using three n2-standard-2 instances, two of which run automation controller and one which runs automation hub. The automation controller instances collectively support a standard workload for 100 managed active nodes. Red Hat has tested this and proved that there is support for up to 10 forks each. This operating criteria has been set and tested using output-intensive, “chatty” workloads that produce 2 messages 7 seconds apart on both automation controller nodes. Workloads that are more I/O intensive might not work inside the boundaries of these conditions and can require the use of extension nodes to scale the deployment to support such automation. For further information, see Purchasing an extension node.

6.2. DEPLOYMENT SCALING

If you want to scale your deployment beyond the initial number of supported managed nodes, Ansible Automation Platform from GCP Marketplace can be manually scaled using separately sold extension nodes.

Extension nodes are additional compute instances that can be deployed to scale-up or scale-out depending on the immediate scaling requirements. If your requirements are for higher parallel automation operations you can select compute shapes that scale-up, while if you have a requirement to automate more nodes over time you can select compute shapes that scale out.

Extension nodes are the supported way of extending the capabilities of Ansible Automation Platform from GCP Marketplace.

NOTE

Environments that are extended through customer design and implementation are not supported by Red Hat.

6.3. PURCHASING AN EXTENSION NODE

The extension node offering enables you to add additional Ansible managed node licenses and additional compute VMs into your Ansible Automation Platform from GCP Marketplace deployment.

Extension nodes for Red Hat Ansible Automation Platform from GCP Marketplace can be obtained directly from the Google Marketplace. Follow these steps to purchase and deploy the public offer.

Procedure

1. In the Google Cloud Console, navigate to the Marketplace.
2. In the **Search Marketplace** text box, search for **Ansible Automation Platform**.
3. Click one of the listings for **Extension Node - Ansible Automation Platform** Depending on your needs, select 100, 200 or 400 Managed Nodes.
4. Ensure that you have selected the offer where Red Hat is the publisher.
5. Review the details, and click **LAUNCH**.

NOTE

You can purchase private offers for extension nodes through your Red Hat account executive.

CHAPTER 7. MONITORING AND LOGGING

You can send metrics to the Google Cloud Platform monitoring system to be visualized in the Google Cloud Platform UI. Ansible Automation Platform from GCP Marketplace metrics and logging are disabled by default as there is a cost to send these metrics to GCP. Refer to Cloud Monitoring and Cloud Logging respectively for more information.

You can set up GCP monitoring and logging either:

- At deployment time, or
- After the deployment

7.1. SETTING UP MONITORING AND LOGGING AT DEPLOYMENT TIME

Procedure

1. In the GCP UI, navigate to **Observability**.
2. Check the **Connect Logging** and **Connect Metrics** checkboxes.

NOTE

These checkboxes are only available in the foundation deployment.

7.2. SETTING UP MONITORING AND LOGGING AFTER DEPLOYMENT

You can start or stop the logging and monitoring after the deployment by using the **gcp_setup_logging_monitoring** playbook available from registry.redhat.com.

7.2.1. Pulling the ansible-on-clouds-ops container image

Pull the Docker image for the Ansible on Clouds operational container which aligns with the version of your foundation deployment.

For example, if your foundation deployment version is 2.3.20230221-00, you must pull the operational image with tag 2.3.20230221.

Use the following commands:

```
$ export IMAGE=registry.redhat.io/ansible-on-clouds/ansible-on-clouds-ops-rhel8:2.3.20230221
$ docker pull $IMAGE --platform=linux/amd64
```

For EMEA regions (Europe, Middle East, Africa) run the following command instead:

```
$ export IMAGE=registry.redhat.io/ansible-on-clouds/ansible-on-clouds-ops-emea-
rhel8:2.3.20230221
$ docker pull $IMAGE --platform=linux/amd64
```

7.2.2. Generating data files by running the ansible-on-clouds-ops container

The following commands generate the required data file. These commands create a directory, and an empty data template that, when populated, is used to generate the playbook.

Procedure

1. Create a folder to hold the configuration files.

```
$ mkdir command_generator_data
```

2. Populate the **command_generator_data** folder with the configuration file template.

```
$ docker run --rm -v $(pwd)/command_generator_data:/data $IMAGE \
command_generator_vars gcp_setup_logging_monitoring \
--output-data-file /data/logging-monitoring.yml
```

Creates the following command file:

```
===============================================
Playbook: gcp_setup_logging_monitoring
Description: This playbook setup the logging and monitoring.
-----------------------------------------------
Install monitoring tools and configure them to send data to the Google Cloud Monitoring
service.
-----------------------------------------------
Command generator template:

docker run --rm -v <local_data_file_directory>:/data $IMAGE command_generator \
gcp_setup_logging_monitoring --data-file /data/logging-monitoring.yml
```

3. When you have run these commands, a **command_generator_data/logging-monitoring.yml** template file is created.

NOTE

In the following example file, **ansible_config_path** is optional.

This template file resembles the following:-

```
gcp_setup_logging_monitoring:
  ansible_config_path:
  cloud_credentials_path:
  deployment_name:
  extra_vars:
    components:
    default_collector_interval:
    logging_enabled:
    monitoring_enabled:
```

7.2.3. Updating the data file

If you do not require a parameter, remove that parameter from the configuration file.

Procedure

- Edit the **command_generator_data/logging-monitoring.yml** file and set the following parameters:
- **ansible_config_path** is used by default as the standard configuration for the **ansible-on-cloud offering** but if you have extra requirements in your environment you can specify your own.
- **cloud_credentials_path** is the absolute path toward your credentials. This must be an absolute path.
- **deployment_name** is the name of the deployment.
- **components** is an array containing the type of component where you want to do the setup. The default is [“controller”, “hub”] which means that the logging monitoring will be enabled on both automation controller and automation hub.
- **monitoring_enabled** is set to **true** to enable the monitoring, **false** otherwise. Default = **false**.
- **logging_enabled** is set to **true** to enable the logging, **false** otherwise. Default = **false**.
- **default_collector_interval** is the periodicity at which the monitoring data must be send to the Google Cloud. Default = 59s.

NOTE

The Google cost of this service depends on that periodicity and so the higher the value of the collector interval, the less it will cost.

Do not set values less than 59 seconds.

NOTE

If monitoring and logging is disabled, the value of 'default_collector_interval' is automatically set to **0**.

7.2.4. Generating the playbook

To generate the playbook, run the command generator to generate the CLI command.

```
docker run --rm -v $(pwd)/command_generator_data:/data $IMAGE command_generator
gcp_setup_logging_monitoring \
--data-file /data/logging-monitoring.yml
```

Provides the following command:

```
docker run --rm --env PLATFORM=GCP -v </path/to/gcp/service-
account.json>:/home/runner/.gcp/credentials:ro \
--env ANSIBLE_CONFIG=../gcp-ansible.cfg --env DEPLOYMENT_NAME=<deployment_name> --
env GENERATE_INVENTORY=true  \
$IMAGE redhat.ansible_on_clouds.gcp_setup_logging_monitoring -e 'gcp_deployment_name=
<deployment_name> \
gcp_service_account_credentials_json_path=/home/runner/.gcp/credentials  monitoring_enabled=
<monitoring_enabled> \
logging_enabled=<logging_enabled> default_collector_interval=<interval>'
```

Run the supplied command to run the playbook.

```
$ docker run --rm --env PLATFORM=GCP -v /path/to/credentials:/home/runner/.gcp/credentials:ro \
--env ANSIBLE_CONFIG=../gcp-ansible.cfg --env DEPLOYMENT_NAME=mu-deployment \
--env GENERATE_INVENTORY=true  $IMAGE
redhat.ansible_on_clouds.gcp_setup_logging_monitoring \
-e 'gcp_deployment_name=mu-deployment \
gcp_service_account_credentials_json_path=/home/runner/.gcp/credentials  components=
["hubs","controllers"]\
monitoring_enabled=True logging_enabled=True default_collector_interval=60s'
```

The process may take some time, and provides output similar to the following:

```
TASK [redhat.ansible_on_clouds.setup_logging_monitoring : Update runtime variable
logging_enabled] ***
changed: [<user_name> -> localhost]

TASK [redhat.ansible_on_clouds.setup_logging_monitoring : Update runtime variable
monitoring_enabled] ***
changed: [<user_name> -> localhost]

PLAY RECAP *************************************************************************
<user_name>    : ok=20   changed=6    unreachable=0    failed=0    skipped=2    rescued=0
ignored=0
```

7.3. CUSTOMIZING MONITORING AND LOGGING

Metrics are provided by Ansible, Podman and Google Ops Agent. Ansible metrics is only installed on automation hub, the Google Ops Agent and Podman are also installed on automation controllers.

A configurable process (collector) runs on each automation controller and automation hub to export the collected Ansible and Podman metrics to Google Cloud Platform Monitoring. As the Google Ops Agent is part of the Google Cloud solution, it has its own configuration file.

The Google Ops Agent is also responsible for the logging configuration.

The service APIs **monitoring.googleapis.com** and **logging.googleapis.com** must be respectively enabled for the monitoring and logging capabilities.

Configuration

Configuration files are located on a disk shared by each automation controller and automation hub. Modify the file **/aap/bootstrap/config_file_templates/<controller|hub>/monitoring.yml** to configure all exporters and agents.

7.3.1. Ansible and podman configuration

The file **/aap/bootstrap/config_file_templates/<controller|hub>/monitoring.yaml** on automation controller or automation hub contains the configuration for collecting and sending Ansible and podman metrics to GCP.

The default configuration for automation controller looks like this:

```
# This value will be set at deployment time.
# Set to zero if monitoringEnabled is false otherwise 59s
# The collection interval for each collector will be the minimum
# between the defaultCollectorInterval and all send Interval
# of a given collector
# NB: The awx exported should not run on controllers as
# it duplicates the number of records sent to GCP Monitoring

defaultCollectorInterval: $DEFAULT_COLLECTOR_INTERVAL
collectors:
- name: podman
 endpoint: http://localhost:9882/podman/metrics
 enabled: true

# list of metrics to exclude
# excludedMetrics:
# - podman_container_created_seconds

 metrics:
 - name: podman_container_exit_code
   # interval on which the metric must be pushed to gcp
   sendInterval: 59s
```

The default configuration for automation hub looks like:

```
# This value will be set at deployment time.
# Set to zero if monitoringEnabled is false otherwise 59s
# The collection interval for each collector will be the minimum
# between the defaultCollectorInterval and all sendInterval
# of a given collector
# NB: The awx exporter should not run on controllers as
# it duplicates the number of records sent to GCP Monitoring

defaultCollectorInterval: 59s
collectors:
- name: awx
 userName: admin
 endpoint: http://<Controller_LB_IP>/api/v2/metrics/
 enabled: true
 metrics:
 - name: awx_inventories_total
   # interval on which the metric must be pushed to gcp
   sendInterval: 59s
- name: podman
 endpoint: http://localhost:9882/podman/metrics
 enabled: true

# list of metrics to exclude
#  excludedMetrics:
#  - podman_container_created_seconds

 metrics:
 - name: podman_container_exit_code
   # interval on which the metric must be pushed to gcp
   sendInterval: 59s
```

where

collectors is a configuration array with one item per collector, that is, awx and podman.

The awx collector requires authentication and so **userName** must be set to **admin**. The password is retrieved from the secret-manager.

The endpoint should not be changed.

defaultCollectorInterval specifies the default interval at which the exporter collects the information from the metric end-point and sends it to Google Cloud Platform Monitoring.

Setting this value to **0** or omitting this attribute disables all collectors.

Each collector can be enabled or disabled separately by setting **enabled** to **true** or **false**.

A collector returns all available metrics grouped by families, but you can exclude the families that should not be sent to Google Cloud Platform Monitoring by adding their name in the array **excludedMetrics**.

For all other family metrics, you can specify the interval at which you want to collect and send them to the Google Cloud Platform Monitoring. The collector interval is the minimum between all family metrics interval and the **defaultCollectorInterval**. This to ensure that a collection is made for each set of metrics sent to Google Cloud Platform Monitoring.

7.3.2. Google cloud ops agent configuration

The configuration file details can be found here.

The configuration file is located in **/etc/google-cloud-ops-agent/config.yml**.

This is a symbolic link to the shared disk **/aap/bootstrap/config_file_templates/controller/gcp-ops-agent-config.yml** or **/aap/bootstrap/config_file_templates/hub/gcp-ops-agent-config.yml** depending on the component type.

The configuration file contains a number of receivers specifying what should be collected by the ops agent.

Your selection of **Connect Logging** and **Connect Metrics** during deployment determines which pipelines are included in the file and therefore which logs and metrics are collected and sent to GCP.

If you need to add more pipelines post-deployment, you can insert them in **/aap/bootstrap/config_file_templates/hub|controller/gcp-ops-agent-config.yml**.

A crontab job restarts the agent if **gcp-ops-agent-config.yml** changed in the last 10 minutes. The agent rereads its configuration after a restart.

CHAPTER 8. UNINSTALLING RED HAT ANSIBLE AUTOMATION PLATFORM FROM GCP MARKETPLACE

This chapter explains how to uninstall Ansible Automation Platform from GCP Marketplace.

8.1. UNINSTALLING THE APPLICATION

IMPORTANT

This action cannot be reversed.

If a Load Balancer was added for the deployment, it must be deleted before uninstalling the deployment.

If an extension node was added to the deployment, it must also be deleted before uninstalling the deployment. Follow the instructions in Removing extension nodes

Procedure

1. Select the main menu from the top left.
2. Select **Deployment Manager**. If you do not see **Deployment Manager**, select **View All Products**.
3. Tick the checkbox in front of the deployment to be deleted.
4. Click **Delete** in the top bar. This action cannot be reversed.

The deployment is deleted.

8.2. DELETING EXTENSION NODES

IMPORTANT

This action cannot be reversed.

Extension nodes are created as deployments in GCP Deployment Mananger. As such, the process to delete an Ansible Automation Platform extension node is the same as the process to delete the main Ansible Automation Platform deployment.

Procedure

1. Select the main menu.
2. Select **Deployment Manager**. If you do not see **Deployment Manager**, select **View All Products**.
3. Tick the checkbox in front of the extension node you want to delete.

NOTE

Extension nodes and the main deployment are both found in **Deployment Manager**. Ensure that you select the extension node and not the main deployment. The name of the extension node was chosen by the person that created it.

4. Click **Delete** in the top bar. This action cannot be reversed.

The extension node is deleted.

8.3. DELETING THE ADMINISTRATION SECRET

Procedure

1. Select the main menu from the top left.
2. Select **Security**. If you do not see **Security**, select **View All Products**.
3. Select **Secret Manager**.
4. The format of the secret name is **<DeploymentName>-aap-admin**. Filter using this name.
5. Click the **More Actions** icon * ⋮ on the line of the deployment.
6. Select **Delete**.
7. The administration password is displayed.
8. Enter the name of the secret.
9. Click **Delete Secret**.

8.4. DELETING THE SERVICE ACCOUNT

IMPORTANT

If a new service account was created for the deployment and if it is not used anywhere else, it can be deleted

Procedure

1. Select the main menu from the top left.
2. Select **IAM & Admin** If you do not see **IAM & Admin**, select **View All Products**.
3. Select **Service Accounts**.
4. Filter with the name of the service account.
5. Click the **More Actions** icon * ⋮ on the line of the service account.
6. Select **Delete**.
7. The administration password is displayed.

8. Click **Delete**.

8.5. DELETING SECRETS LEFT OVER AFTER UPGRADE, BACKUP AND RESTORE

NOTE

This action cannot be reversed.

Only perform this action if your deployment is deleted.

There are several possible secret names you can find, depending on the procedures you have completed:

- *<deployment_name>*-aap-secret-key
- *<deployment_name>*-aap-admin
- *<deployment_name>*-container_auth_private_key_pem
- *<deployment_name>*-container_auth_public_key_pem
- *<deployment_name>*-database_fields_symmetric_key
- *<deployment_name>*-pulp_cert
- *<deployment_name>*-pulp_key

Procedure

1. Select the main menu from the top left.
2. Select **Security**. If you do not see **Security**, select **View All Products**.
3. Select **Secret Manager**.
4. Search your deployment name, looking for the names in the list above.
5. Click the **More Actions** icon * ⋮ on the line of the deployment.
6. Select **Delete**. The administration password is displayed.
7. Enter the name of the secret.
8. Click **Delete Secret**.

8.6. DELETING THE FILESTORE AFTER BACKUP

NOTE

This action cannot be reversed.

1. Procedure

2. Select the main menu from the top left.
3. Select **Filestore**. If you do not see **Filestore**, select **View All Products**.
4. Select **Backups**.
5. The format of the backup name is **<DeploymentName>-filestore-<RandomSufix>**. Filter using this name.
6. Click the **More Actions** icon * ⋮ on the line of the deployment.
7. Select **Delete**.
8. The administration password is displayed.
9. Enter the name of the secret.
10. Click **Delete Backup**.

8.7. DELETING THE OBJECT STORAGE BUCKET AFTER BACKUP

Procedure

1. Select the main menu from the top left.
2. Select **Cloud Storage**. If you do not see **Cloud Storage**, select **View All Products**.
3. Select **Buckets**.
4. Search for your deployment name.
5. Click the **More Actions** icon * ⋮ on the line of the deployment.
6. Select **Delete**.
7. The administration password is displayed.
8. Enter the name of the secret.
9. Click **Delete Secret**.

CHAPTER 9. BACKUP AND RESTORE FOR ANSIBLE AUTOMATION PLATFORM FROM GCP MARKETPLACE

IMPORTANT

- You must restore with the same operational image version as the backup.
- You can only restore using a new VPC network. Backup and restore into an existing VPC network is not currently supported.
- You cannot delete the backed up environment before a restore because the database backups are stored in the deployment. Deleting your deployment therefore deletes the database backups.
- Versioned or point-in-time backups are not supported by the **ansible-on-clouds-op** container. Only the most recent backup of a deployment is used to restore the deployment.

To backup and restore your Ansible Automation Platform deployment, it is vital to have your existing Ansible Automation Platform administration secret name and value recorded somewhere safe.

It is also important to take regular manual backups of the Cloud SQL database instance and filestore backups, to ensure a deployment can be restored as close as possible to its previous working state.

The playbooks backup and restore provides backup and restore support for the Ansible Automation Platform from GCP Marketplace foundation deployment.

NOTE

The restore process deploys a new Ansible Automation Platform with the filestore and SQL database instance being restored to the specified backup.

9.1. THE BACKUP PROCESS

A backup enables you to backup your environment by saving the database and the shared file system. A new environment is created during the restore using the saved shared file system. When the new environment is in place, the process restores the database.

The backup and restore process must use the same version. If your backup was done with an earlier version, you must use the restore process of that version. Then, if required, you can run an upgrade.

You must also make a backup before an upgrade. For further information, see Upgrading your deployment

The backup process involves taking a backup of the Cloud SQL database and filestore instances at a given point in time. The backup playbook requires an active Ansible Automation Platform from GCP Marketplace foundation deployment to be running.

A bucket needs to be created in your project as restore information will be stored in that bucket.

Only one backup per version is kept in the bucket, if multiple versions of the backup need to be retain then a new bucket must be created for the new version of the backup.

The following procedures describe how to backup the Ansible Automation Platform from GCP Marketplace deployment.

9.1.1. Pulling the ansible-on-clouds-ops 2.3 container image

Procedure

- Pull the docker image for the **ansible-on-clouds-ops** 2.3 container with the same tag as the foundation deployment.

  NOTE

  Before pulling the docker image, make sure you are logged in to registry.redhat.com using docker. Use the following command to login to registry.redhat.com.

  ```
  $ docker login registry.redhat.io
  ```

  For more information about registry login, see Registry Authentication

  ```
  $ export IMAGE=registry.redhat.io/ansible-on-clouds/ansible-on-clouds-ops-
  rhel8:2.3.20230221
  $ docker pull $IMAGE --platform=linux/amd64
  ```

For EMEA regions (Europe, Middle East, Africa) run the following command instead:

```
$ export IMAGE=registry.redhat.io/ansible-on-clouds/ansible-on-clouds-ops-emea-
rhel8:2.3.20230221
$ docker pull $IMAGE --platform=linux/amd64
```

9.1.2. Setting up the environment

Procedure

- Create a folder to hold the configuration files.

  ```
  $ mkdir command_generator_data
  ```

9.1.3. Creating the backup data file

Procedure

1. Populate the **command_generator_data** directory with the configuration file template.

   ```
   docker run --rm -v $(pwd)/command_generator_data/:/data $IMAGE
   command_generator_vars gcp_backup_deployment --output-data-file /data/backup.yml
   ```

   Produces the following output:

   ```
   docker run --rm -v $(pwd)/command_generator_data/:/data $IMAGE \
   ```

```
command_generator_vars gcp_backup_deployment --output-data-file /data/backup.yml

===============================================
Playbook: gcp_backup_deployment
Description: This playbook is used to backup the Ansible Automation Platform from GCP
Marketplace environment.
-----------------------------------------------
This playbook is used to backup the Ansible Automation Platform from GCP Marketplace
environment.
For more information regarding backup and restore, visit our official documentation -
https://access.redhat.com/documentation/en-
us/ansible_on_clouds/2.x/html/red_hat_ansible_automation_platform_from_gcp_marketplace_g
uide/assembly-gcp-backup-and-restore
-----------------------------------------------
```

2. Run the supplied command:

```
$ docker run --rm -v $(pwd)/command_generator_data/:/data $IMAGE \
command_generator_vars gcp_backup_deployment --output-data-file /data/backup.yml
```

3. After running the command, a **/tmp/backup.yml** template file is created. This template file resembles the following:

```
gcp_backup_deployment:
  cloud_credentials_path:
  deployment_name:
  extra_vars:
    gcp_bucket_backup_name:
    gcp_compute_region:
    gcp_compute_zone:
```

9.1.4. Parameters in the backup.yml file

You must populate the data file before triggering the backup. The following variables are parameters listed in the data file.

- **cloud_credentials_path** is the path for your Google Cloud service account credentials file. This must be an absolute path.
- **gcp_deployment_name** is the name of the AAP deployment manager deployment you want to back up.
- **gcp_bucket_backup_name** is the bucket that was previously created to use for the backup. Only the most recent backup is stored in the bucket. Every subsequent backup to the same bucket overwrites the backup files with the latest backup.
- **gcp_compute_region** is GCP region where the foundation deployment is deployed. This can be retrieved by checking the Deployments config in Deployment Manager.
- **gcp_compute_zone** is the GCP zone where the foundation deployment is deployed. This can be retrieved by checking the Deployments config in Deployment Manager.

9.1.5. Running the backup playbook

Procedure

Procedure

1. To run the backup, run the command generator to generate the backup command.

```
docker run --rm -v $(pwd)/command_generator_data:/data $IMAGE command_generator
gcp_backup_deployment --data-file /data/backup.yml
```

   Resulting in the following ouput:

```
-----------------------------------------------
Command to run playbook:

docker run --rm --env PLATFORM=GCP -v </path/to/gcp/service-
account.json>:/home/runner/.gcp/credentials:ro \
--env ANSIBLE_CONFIG=../gcp-ansible.cfg  $IMAGE
redhat.ansible_on_clouds.gcp_backup_deployment \
-e 'gcp_service_account_credentials_json_path=/home/runner/.gcp/credentials  \
gcp_deployment_name=<deployment_name> gcp_compute_region=<region> \
gcp_compute_zone=<zone> gcp_bucket_backup_name=<bucket>'
```

2. Run the supplied backup command to trigger the backup.

```
$ docker run --rm --env PLATFORM=GCP -v </path/to/gcp/service-
account.json>:/home/runner/.gcp/credentials:ro \
--env ANSIBLE_CONFIG=../gcp-ansible.cfg  $IMAGE
redhat.ansible_on_clouds.gcp_backup_deployment \
-e 'gcp_service_account_credentials_json_path=/home/runner/.gcp/credentials  \
gcp_deployment_name=<deployment_name> gcp_compute_region=<region> \
gcp_compute_zone=<zone> gcp_bucket_backup_name=<bucket>'
```

3. When the playbook has finished running, the output resembles the following:

```
TASK [redhat.ansible_on_clouds.standalone_gcp_backup : [backup_deployment] Print vars
required for restore process] ***
ok: [localhost] =>
  msg:
  - AAP on GCP Backup successful. Please note below the bucket name which is required for
restore process.
  - <bucket_name>

PLAY RECAP
*********************************************************************************************
localhost          : ok=33   changed=6    unreachable=0    failed=0    skipped=1    rescued=0
ignored=0
```

9.2. RESTORE PROCESS

The restore process deploys a new deployment, and restores the filestore and SQL database instance to the specified backup.

IMPORTANT

- You must restore with the same operational image version which was used for the backup.
- Backup and restore into an existing VPC network is not currently supported.
- You cannot delete the backed up environment before a restore because the database backups are stored in the deployment. Deleting your deployment therefore deletes the database backups.
- Versioned or point-in-time backups are not supported by the **ansible-on-clouds-op** container. Only the most recent backup of a deployment is used to restore the deployment.

The following procedures describe how to restore the Ansible Automation Platform from GCP Marketplace deployment.

9.2.1. Pulling the ansible-on-clouds-ops 2.3 container image

Procedure

- Pull the docker image for the **ansible-on-clouds-ops** 2.3 container with the same tag as the foundation deployment.

  NOTE

  Before pulling the docker image, make sure you are logged in to registry.redhat.com using docker. Use the following command to login to registry.redhat.com.

  ```
  $ docker login registry.redhat.io
  ```

  For more information about registry login, see Registry Authentication

  ```
  $ export IMAGE=registry.redhat.io/ansible-on-clouds/ansible-on-clouds-ops-rhel8:2.3.20230221
  $ docker pull $IMAGE --platform=linux/amd64
  ```

For EMEA regions (Europe, Middle East, Africa) run the following command instead:

```
$ export IMAGE=registry.redhat.io/ansible-on-clouds/ansible-on-clouds-ops-emea-rhel8:2.3.20230221
$ docker pull $IMAGE --platform=linux/amd64
```

9.2.2. Setting up the environment

Procedure

- Create a folder to hold the configuration files.

  ```
  $ mkdir command_generator_data
  ```

9.2.3. Generating the restore.yml file

Procedure

1. Run the command generator **command_generator_vars**` to generate **restore.yml**.

```
docker run --rm -v $(pwd)/command_generator_data:/data $IMAGE
command_generator_vars gcp_restore_deployment --output-data-file /data/restore.yml
```

Providing the following output:

```
===============================================
Playbook: gcp_restore_deployment
Description: This playbook is used to restore the Ansible Automation Platform from GCP
Marketplace environment from a backup.
-----------------------------------------------
This playbook is used to restore the Ansible Automation Platform from GCP Marketplace
environment from a backup.
For more information regarding backup and restore, visit our official documentation -
https://access.redhat.com/documentation/en-
us/ansible_on_clouds/2.x/html/red_hat_ansible_automation_platform_from_gcp_marketplace_g
uide/assembly-gcp-backup-and-restore
-----------------------------------------------
Command generator template:

docker run --rm -v <local_data_file_directory>:/data $IMAGE command_generator
gcp_restore_deployment --data-file /data/restore.yml
```

The template resembles the following:

```
gcp_restore_deployment:
  cloud_credentials_path:
  deployment_name:
  extra_vars:
    gcp_bucket_backup_name:
    gcp_cloud_sql_peering_network:
    gcp_compute_region:
    gcp_compute_zone:
    gcp_controller_internal_ip_address:
    gcp_existing_vpc:
    gcp_filestore_ip_range:
    gcp_hub_internal_ip_address:
    gcp_restored_deployment_name:
```

9.2.4. Parameters of the restore.yml file

You can only restore into a new VPC network.

For a new VPC

If you want to restore using a new VPC set the following parameters:

- **gcp_existing_vpc** must be set to **false**.

The following parameters must be removed:

- **gcp_filestore_ip_range**
- **gcp_cloud_sql_peering_network**
- **gcp_controller_internal_ip_address**
- **gcp_hub_internal_ip_address**

Provide values for the following parameters:

- **gcp_existing_vpc** must be set to **false**.
- **cloud_credentials_path** is the path for your Google Cloud service account credentials file.
- **gcp_deployment_name** is the name of the AAP deployment manager deployment you want to back up.
- **gcp_restored_deployment_name** is the name under which the deployment must be restored. A new deployment will be created with this name. A deployment must not already exist with this name.
- **gcp_bucket_backup_name** is the bucket name you used for the backup.
- **gcp_compute_region** is the region where the backup was taken. This can be retrieved by checking the Deployments config in Deployment Manager.
- **gcp_compute_zone** is the zone where the backup was taken. This can be retrieved by checking the Deployments config in Deployment Manager.

9.2.5. Running the restore command

When **restore.yml** is populated, you can use the command generator to create the restore command.

Procedure

1. Run the command generator.

   NOTE

   As /**tmp** is used, the **<local_data_file_directory>** must be set to /**tmp** unless you chose a different location for **restore.yml**.

   ```
   $ docker run --rm -v $(pwd)/command_generator_data:/data $IMAGE command_generator
   gcp_restore_deployment --data-file /data/restore.yml
   ```

   This generates a new command containing all needed volumes, environment variables and parameters.

   The generated command resembles the following:

   ```
   docker run --rm --env PLATFORM=GCP -v
   <local_credential_file>:/home/runner/.gcp/credentials:ro \
   --env ANSIBLE_CONFIG=../gcp-ansible.cfg  $IMAGE
   ```

```
redhat.ansible_on_clouds.gcp_restore_deployment \
-e 'gcp_service_account_credentials_json_path=/home/runner/.gcp/credentials  \
gcp_deployment_name=<former_deployment_name> gcp_restored_deployment_name=
<new_deployment_name> \
gcp_compute_region=<region> gcp_compute_zone=<zone> gcp_bucket_backup_name=
<bucket> gcp_existing_vpc=False'
```

2. Run the generated command.

```
$ docker run --rm --env PLATFORM=GCP -v
<local_credential_file>:/home/runner/.gcp/credentials:ro \
--env ANSIBLE_CONFIG=../gcp-ansible.cfg  $IMAGE
redhat.ansible_on_clouds.gcp_restore_deployment \
-e 'gcp_service_account_credentials_json_path=/home/runner/.gcp/credentials  \
gcp_deployment_name=<former_deployment_name> gcp_restored_deployment_name=
<new_deployment_name> \
gcp_compute_region=<region> gcp_compute_zone=<zone> gcp_bucket_backup_name=
<bucket> gcp_existing_vpc=False'
```

3. When the playbook has completed, the output resembles the following:

```
TASK [redhat.ansible_on_clouds.standalone_gcp_restore : Display internal IP addresses] ***
ok: [localhost] =>
  msg:
  - 'Hub        internal IP: 192.168.240.21'
  - 'Controller internal IP: 192.168.240.20'

PLAY RECAP *********************************************************************
localhost                  : ok=33   changed=8    unreachable=0    failed=0    skipped=6
rescued=0    ignored=2
```

CHAPTER 10. UPGRADING YOUR DEPLOYMENT

You can upgrade your existing Ansible Automation Platform from GCP Marketplace deployment to the newer version. The upgrade process covers upgrading of automation hub and automation controller. The upgrade process takes roughly the same amount of time as a Ansible Automation Platform from GCP Marketplace deployment install.

NOTE

Ansible Automation Platform from GCP Marketplace supports sequential upgrades. All upgrades should be no more than one major version behind the version you are currently upgrading to. For example, to upgrade to Ansible Automation Platform from GCP Marketplace to 2.3.20230221-00, you must be on version 2.2.20230215-00.

Prerequisites

- Docker must be installed to run the upgrade playbook.
- The upgrade process requires several volumes to be mounted. Prepare a fresh directory to be used for this process.

The following procedures form the upgrade process:

1. Backup your Ansible Automation Platform 2.2 stack.
   a. Pull the **ansible-on-clouds-ops** 2.2 container image
   b. Prepare the environment
   c. Run the **ansible-on-clouds-ops** 2.2 container to backup the stack
2. Upgrade Ansible Automation Platform.
   a. Pull the **ansible-on-clouds-ops** 2.3 container image
   b. Generate data files by running the **ansible-on-clouds-ops** container
   c. Update the data file
   d. Run the **ansible-on-clouds-ops** 2.3 container to upgrade your Ansible Automation Platform from GCP Marketplace deployment
3. (Optional) Restore the stack from backup.
   a. Pull the **ansible-on-clouds-ops** 2.2 container image
   b. Prepare the environment
   c. Run the **ansible-on-clouds-ops** 2.2 container to restore the stack

10.1. BACKING UP BEFORE THE UPGRADE

The following procedures backup the deployment before the upgrade process begins.

10.1.1. Pulling the ansible-on-clouds-ops 2.2 container image

Procedure

- Pull the **ansible-on-clouds-ops 2.2** container image with the same tag version as the foundation deployment.

  NOTE

  Before pulling the docker image, make sure you are logged in to registry.redhat.com using docker. Use the following command to login to registry.redhat.com.

  ```
  $ docker login registry.redhat.io
  ```

  For more information about registry login, see Registry Authentication

  ```
  $ export IMAGE=registry.redhat.io/ansible-on-clouds/ansible-on-clouds-ops-rhel8:2.2.20230215
  $ docker pull $IMAGE --platform=linux/amd64
  ```

For EMEA regions (Europe, Middle East, Africa) run the following command instead:

```
$ export IMAGE=registry.redhat.io/ansible-on-clouds/ansible-on-clouds-ops-emea-rhel8:2.2.20230215
$ docker pull $IMAGE --platform=linux/amd64
```

10.1.2. Preparing the backup environment

Procedure

- Create the directories used to configure the upgrade process using the following commands:

  ```
  # Secrets directory. Store the GCP credentials file.
  $ mkdir secrets
  # Extra vars directory. All the configuration variables for the upgrade process.
  $ mkdir extra_vars
  ```

10.1.3. Backing up your Ansible Automation Platform stack

Procedure

1. Store your service account credential JSON file in the /**secrets** directory.

   NOTE

   To create the GCP credentials file, see, Create credentials and Create and delete service account keys

2. The **extra_vars.yml** file contains the following input variables within the **extra_vars** directory:

   - **gcp_service_account_credentials_json_path** is the service account credential JSON file path for the Google cloud account. This is the path within an **ansible-on-clouds-ops**

container where the credentials JSON file is mounted to the /**secrets** directory. The path should be /**secrets**/**<service-account-creds-json-filename>**.
Replace **<service-account-creds-json-filename>** with your Google service account credential filename.

- **gcp_deployment_name** is the deployment name that you want to upgrade.
- **gcp_compute_region** is the GCP region that you provided when deploying foundation deployment. If you have not provided a region when deploying the foundation, use the default region **us-east1** here.
- **gcp_compute_zone** is the GCP zone that you provided when deploying foundation deployment. If you have not provided a zone when deploying the foundation, use the default **us-east1-b** here.

NOTE

You can find the region and zone from the filestore instance deployed with the foundation deployment. Look for the filestore with the name **<deployment-name>-filestore**.

- Make note of the location field that is your **gcp_compute_zone**. For example, the filestore instance location might be **us-east1-d**
- Remove the last two characters from the location that is your **gcp_compute_region**. For example, if your **gcp_compute_zone** location is **us-east1-d**, your region is **us-east1**.

```
gcp_service_account_credentials_json_path: "/secrets/<service-account-creds-json-filename>"
gcp_deployment_name: ""
gcp_compute_region: ""
gcp_compute_zone: ""
```

3. The final result should look similar to the following:

```
$ tree
tree
.
├── extra_vars
│   └── extra_vars.yaml
└── secrets
    └── gc-ansible-cloud-123434.json
```

10.1.4. Running the backup playbook as a container

Procedure

1. Replace **<absolute path to Google application credentials dir>** with the path to the service account credential JSON file for the Google cloud.
2. Replace **<absolute path of extra vars dir>** with the path to the directory where the **extra_vars.yml** file you created in step 1 of the backup process.

3. Use the following command to run the playbook:

```
$ docker run --rm \
-v <absolute path to google application credentials dir>:/secrets:ro \
-v <absolute path of extra vars dir>:/extra_vars:ro \
$IMAGE \
redhat.ansible_on_clouds.gcp_backup_deployment \
-e @/extra_vars/extra_vars.yaml
```

4. Note the output of the playbook.
The playbook output resembles the following:

```
TASK [redhat.ansible_on_clouds.standalone_gcp_backup : [backup_deployment] Print vars
required for restore process] ***
ok: [localhost] => {
    "msg": [
        "AAP on GCP Backup successful. Please note below vars which are required for restore
process.",
        "{ gcp_sql_backup_id: 1677552759614 , gcp_sql_backup_db_name: test-bnr-aap-db
,gcp_filestore_backup_name: test-bnr-filestore-iygs }"
    ]
}
```

IMPORTANT

Make a note of the SQL database backup ID, SQL database backup name and the filestore backup name. You will require these for the restore playbook.

10.2. UPGRADING ANSIBLE AUTOMATION PLATFORM

The following procedures form the upgrading process for Ansible Automation Platform from GCP Marketplace.

10.2.1. Pulling the ansible-on-clouds-ops 2.3 container image

Procedure

- Pull the docker image for the **ansible-on-clouds-ops** 2.3 container with the same tag as the version you want to upgrade to.

NOTE

Before pulling the docker image, make sure you are logged in to registry.redhat.com using docker. Use the following command to login to registry.redhat.com.

```
$ docker login registry.redhat.io
```

For more information about registry login, see Registry Authentication

NOTE

The **ansible-on-clouds-ops** container image tag must match the version that you want to upgrade to. For example, if your foundation deployment version is 2.2.20230215-00, pull the **ansible-on-clouds-ops** image with tag 2.3.20230221 to upgrade to version 2.3.20230221.

```
$ export IMAGE=registry.redhat.io/ansible-on-clouds/ansible-on-clouds-ops-
rhel8:2.3.20230221
$ docker pull $IMAGE --platform=linux/amd64
```

For EMEA regions (Europe, Middle East, Africa) run the following command instead:

```
$ export IMAGE=registry.redhat.io/ansible-on-clouds/ansible-on-clouds-ops-emea-
rhel8:2.3.20230221
$ docker pull $IMAGE --platform=linux/amd64
```

10.2.2. Generating the data file

Procedure

1. Run the following commands to generate the required data file. These commands create a directory, and populate it with an empty data template that, when populated, is used during the upgrade.

```
# Create a folder to hold the configuration files
$ mkdir command_generator_data
# Populate command_generator_data folder with configuration file template
$ docker run --rm -v $(pwd)/command_generator_data:/data $IMAGE \
command_generator_vars gcp_upgrade \
--output-data-file /data/extra_vars.yml
```

2. After running the previous commands, a **command_generator_data/extra_vars.yml** template is created. This template file should resemble:

```
gcp_upgrade:
ansible_config_path:
cloud_credentials_path:
deployment_name:
extra_vars:
  gcp_backup_taken:
  gcp_compute_region:
  gcp_compute_zone:
```

10.2.3. Updating the data file

You must populate the data file before triggering the upgrade. The following variables are parameters listed in the data file.

- **ansible_config_path** (Optional) Only use if overriding with a custom ansible_config.
- **cloud_credentials_path** is the path to your GCP credentials file.

- **deployment_name** is the name of the foundation deployment. This is the same name you used when you deployed the foundation.
- **gcp_backup_taken** is the verification that a manual backup of the current deployment was recently created prior to running this upgrade. Use **true** here to verify a recent backup was created.
- **gcp_compute_region** is the GCP region that you provided when deploying foundation deployment. If you have not provided a region when deploying the foundation, use the default region **us-east1** here.
- **gcp_compute_zone** is the GCP zone that you provided when deploying foundation deployment. If you have not provided a zone when deploying the foundation, use the default **us-east1-b** here.

After populating the data file, it should resemble the following.

The following values are provided as examples:

```
gcp_upgrade:
  ansible_config_path:
  cloud_credentials_path: ~/secrets/GCP-secrets.json
  deployment_name: AnsibleAutomationPlatform
  extra_vars:
    gcp_backup_taken: true
    gcp_compute_region: us-east1
    gcp_compute_zone: us-east1-b
```

10.2.4. Running the upgrade playbook

1. To run the upgrade, run the command generator to generate the upgrade CLI command:

```
$ docker run --rm -v $(pwd)/command_generator_data:/data $IMAGE command_generator --
data-file /data/extra_vars.yml
```

   This generates the following command:

```
-----------------------------------------------
docker run --rm --env PLATFORM=GCP -v ~/secrets/GCP-
secrets.json:/home/runner/.gcp/credentials:ro
--env ANSIBLE_CONFIG=../gcp-ansible.cfg
--env DEPLOYMENT_NAME=AnsibleAutomationPlatform
--env GENERATE_INVENTORY=true  $IMAGE redhat.ansible_on_clouds.gcp_upgrade \
-e 'gcp_deployment_name=AnsibleAutomationPlatform \
gcp_service_account_credentials_json_path=/home/runner/.gcp/credentials \
gcp_compute_region=us-east1 gcp_compute_zone=us-east1-b gcp_backup_taken=True'
===============================================
```

2. Run the given upgrade command to trigger the upgrade.

```
$ docker run --rm --env PLATFORM=GCP -v ~/secrets/GCP-
secrets.json:/home/runner/.gcp/credentials:ro \
--env ANSIBLE_CONFIG=../gcp-ansible.cfg \
--env DEPLOYMENT_NAME=AnsibleAutomationPlatform \
--env GENERATE_INVENTORY=true  $IMAGE redhat.ansible_on_clouds.gcp_upgrade \
-e 'gcp_deployment_name=AnsibleAutomationPlatform \
gcp_service_account_credentials_json_path=/home/runner/.gcp/credentials \
gcp_compute_region=us-east1 gcp_compute_zone=us-east1-b gcp_backup_taken=True'
```

3. The upgrade can take some time to complete, but can take longer depending on the number of extension nodes on the system. A successful upgrade is marked by the log below.

```
TASK [redhat.ansible_on_clouds.standalone_gcp_upgrade : [upgrade] Show GCP current
version] ***
ok: [localhost] => {
    "msg": "gcp_current_version: 2.3.20230221-00"
}
```

4. Your Ansible Automation Platform from GCP Marketplace deployment is now upgraded to a newer version and you can log in to Red Hat Ansible Automation Platform automation controller and automation hub using your deployment credentials.

10.3. RESTORING THE STACK FROM BACKUPS

If, for some reason, the upgrade process fails, you can use the backup files from Backing up your Ansible Automation Platform stack to restore your stack.

NOTE

The following procedures are optional.

If the upgrade process fails, you can use the backup files from Backing up before the upgrade to restore your stack. The restore process installs a new deployment, and restores the filestore and SQL database instance to the specified backup.

NOTE

The restored filestore instance uses CIDR network range 192.168.244.0/29, which is different from the default network filestore network range of the backup deployment.

The following instructions describe how to run the restore playbook.

10.3.1. Pulling the ansible-on-clouds-ops 2.2 container image to restore the stack

Procedure

- Pull the **ansible-on-clouds-ops 2.2** container image with the same tag version as the foundation deployment.

NOTE

Before pulling the docker image, make sure you are logged in to registry.redhat.com using docker. Use the following command to login to registry.redhat.com.

```
$ docker login registry.redhat.io
```

For more information about registry login, see Registry Authentication

```
$ export IMAGE=registry.redhat.io/ansible-on-clouds/ansible-on-clouds-ops-rhel8:2.2.20230215
$ docker pull $IMAGE --platform=linux/amd64
```

For EMEA regions (Europe, Middle East, Africa) run the following command instead:

```
$ export IMAGE=registry.redhat.io/ansible-on-clouds/ansible-on-clouds-ops-emea-rhel8:2.2.20230215
$ docker pull $IMAGE --platform=linux/amd64
```

10.3.2. Preparing the backup environment

Procedure

- Create the directories used to configure the upgrade process using the following commands:

```
# Secrets directory. Store the GCP credentials file.
$ mkdir secrets
# Extra vars directory. All the configuration variables for the upgrade process.
$ mkdir extra_vars
```

10.3.3. Running the restore playbook as a container

Procedure

1. Replace **<absolute path to Google application credentials dir>** with the path to the service account credential JSON file for the Google cloud.
2. Replace **<absolute path of extra vars dir>** with the path to the **extra_vars.yml** file you created in Preparing the backup environment.
3. Use the following command to run the playbook.

```
$ docker run --rm \
-v <absolute path to Google application credentials dir>:/secrets:ro \
-v <absolute path of extra vars dir>:/extra_vars:ro \
$IMAGE \
redhat.ansible_on_clouds.gcp_restore_deployment \
-e @/extra_vars/extra_vars.yaml
```

4. When you have run the playbook, a new restored deployment is visible in the GCP deployment. It can take a few minutes for the deployment to finish and for all the containers to run.

NOTE

Access to the restored deployment must be configured through either an external load balancer or VPN. When a connection method is configured you can log in to Red Hat Ansible Automation Platform automation controller and automation hub using your old deployment credentials. In addition, all job history, uploaded collections and other records must be in the same state as the restored deployment.

CHAPTER 11. TECHNICAL NOTES

Ansible Automation Platform from GCP Marketplace is a self-managed deployment. The following are technical notes regarding the Ansible Automation Platform from GCP Marketplace deployment.

11.1. VIRTUAL MACHINE LIMITATIONS

You cannot restart virtual machines, instead virtual machines must be replaced. You cannot configure on virtual machines, because any configuration is lost when virtual machines are replaced. This always happens on upgrade.

CHAPTER 12. SUPPORT

Ansible Automation Platform from GCP Marketplace is a self-managed deployment. When the application is deployed into your GCP infrastructure, customers are responsible for the maintenance of the GCP infrastructure and Ansible Automation Platform patching. However, Red Hat does not support any changes to the infrastructure resources deployed as part of the solution unless there is documentation by Red Hat to do so, such as in route table configuration for networking or documented upgrade processes. Adding additional compute resources outside of extension nodes voids Red Hat support for the Ansible Automation Platform from GCP Marketplace deployment.

12.1. VM IMAGE PACKAGES

Ansible Automation Platform from GCP Marketplace is delivered through virtual machines that are based on a VM image produced by the Red Hat Ansible team. This image contains packages from Red Hat and Google in order to properly function on Google Cloud Platform and run Red Hat Ansible Automation Platform. The image contains the following Google packages and containers:

Package Name	Description
google-osconfig-agent	The Google OS Configuration Agent uses the management service to deploy, query, and maintain consistent configurations, that is, the desired state and software for your VM instance.
google-cloud-ops-agent	The Ops Agent is the primary agent for collecting telemetry from your Compute Engine instances. Combining logging and metrics into a single agent, the Ops Agent uses Fluent Bit for logs, which supports high-throughput logging, and the OpenTelemetry Collector for metrics.
gce-proxy	The Cloud SQL Authorization proxy is used to connect Ansible Automation Platform components with the Cloud SQL database using the virtual machine service account.
gcloud CLI	The gcloud command line interface is used to configure the Ansible Automation Platform installation.

Upgrades, including package updates and CVE patches, will be provided as new VM images through the GCP Marketplace. Red Hat operates within its premium support policies for providing updated machine images to Red Hat Ansible Automation Platform from GCP Marketplace, and will provide upgrade instructions to customers when upgradable machine images are available.

12.2. SUPPORTED INFRASTRUCTURE CONFIGURATION CHANGES

Red Hat supports changes to the following options:

- VPC Route Configuration
- VPC Firewall Configuration
- VPC Load Balancer Configuration

- Block Storage Expansion
- DNS and **resolv.conf** files

Red Hat Responsibilities

Red Hat has the following responsibilities.

- Premium support for Ansible Automation Platform.
- Directions and how-to processes for upgrading Ansible Automation Platform from GCP Marketplace.
  - Provided through knowledge base articles when upgrades are published.

Customer Responsibilities

You have the following responsibilities:

- GCP Infrastructure Uptime
- GCP Infrastructure Changes (ex: increasing block storage sizes)
- GCP Network Peering and Configuration
- Backing up Ansible data from GCP databases and file storageResources
- Application of Ansible Automation Platform upgrades
  - Upgrade instructions will be provided at upgrade time through Red Hat Knowledge Base articles.