Ansible on Clouds 2.x

Red Hat Ansible Automation Platform from GCP Marketplace Guide

Install and configure Red Hat Ansible Automation Platform from GCP Marketplace

Last Updated: 2024-05-03

Ansible on Clouds 2.x Red Hat Ansible Automation Platform from GCP Marketplace Guide

Install and configure Red Hat Ansible Automation Platform from GCP Marketplace

Legal Notice

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.

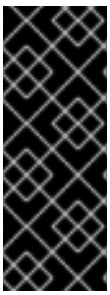
Table of Contents

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

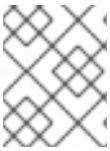
Disclaimer: Links contained in this document to external website(s) are provided for convenience only. Red Hat has not reviewed the links and is not responsible for the content or its availability. The inclusion of any link to an external website does not imply endorsement by Red Hat of the website or 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

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



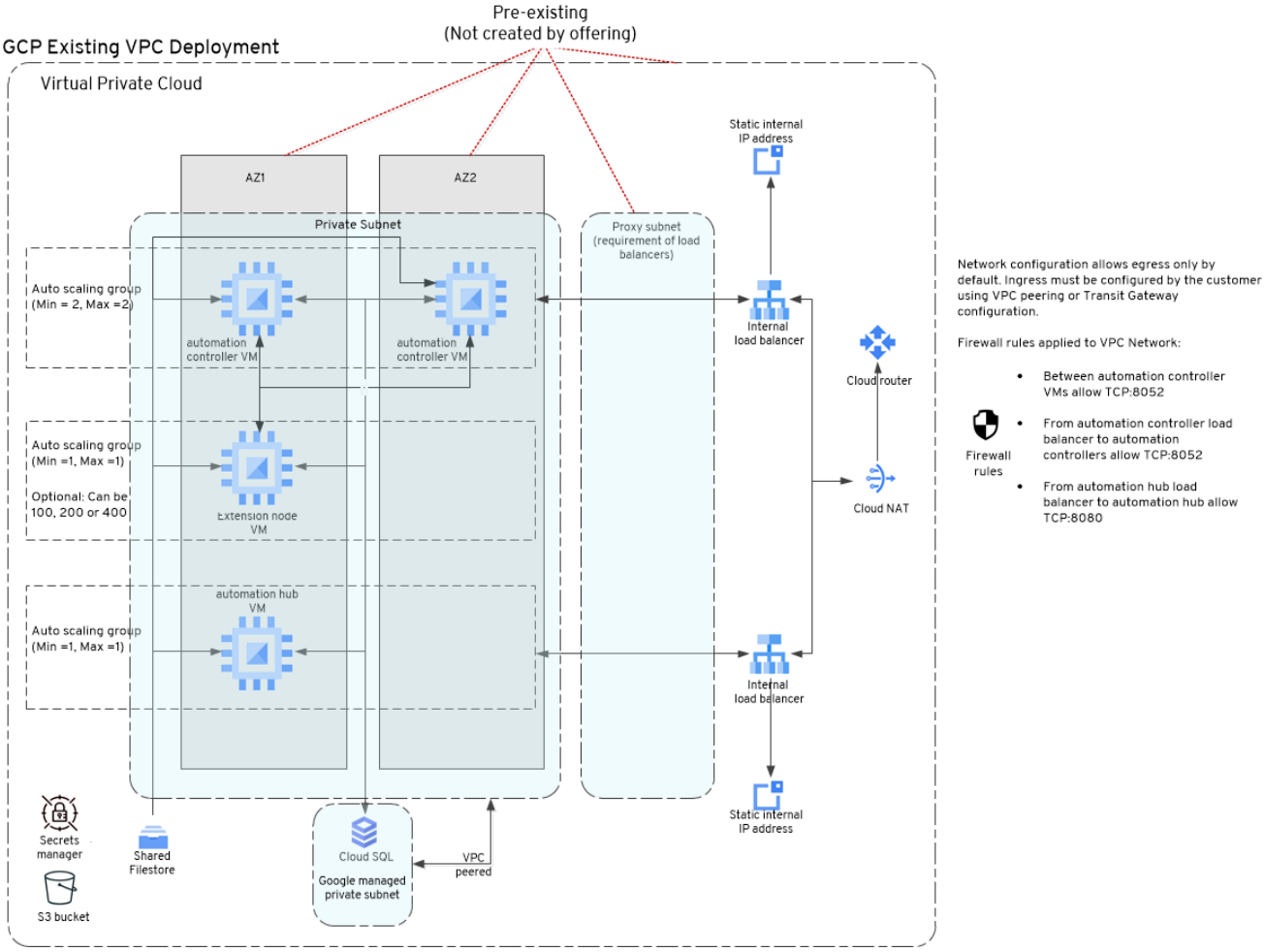
NOTE

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 periiod that the VM exists, usually less than an hour.



Ansible Automation Platform from GCP Marketplace is designed to be private, with no public access allowed by default.

This requires customers to expose the deployed *Internal Load Balancers* (ILBs) themselves pursuant to their own network requirements and security practices. Some potential ways to expose the ILBs include VPC Peering, Transit Gateway, VPN, External Load Balancers, amongst others.

All cloud infrastructure components are deployed in a *Virtual Private Cloud* (VPC).

Customers can choose between deploying into an existing VPC, or to have the product deploy a new VPC for them. All VM instances and Cloud infrastructure have private IP addresses (allocation determined by the VPC and subnetworks specified at deployment time) by default.

All internal traffic is encrypted using self-signed certificates generated at deployment time (external traffic can also be encrypted by deploying your own certificate on the Internal Load Balancers deployed by the product).

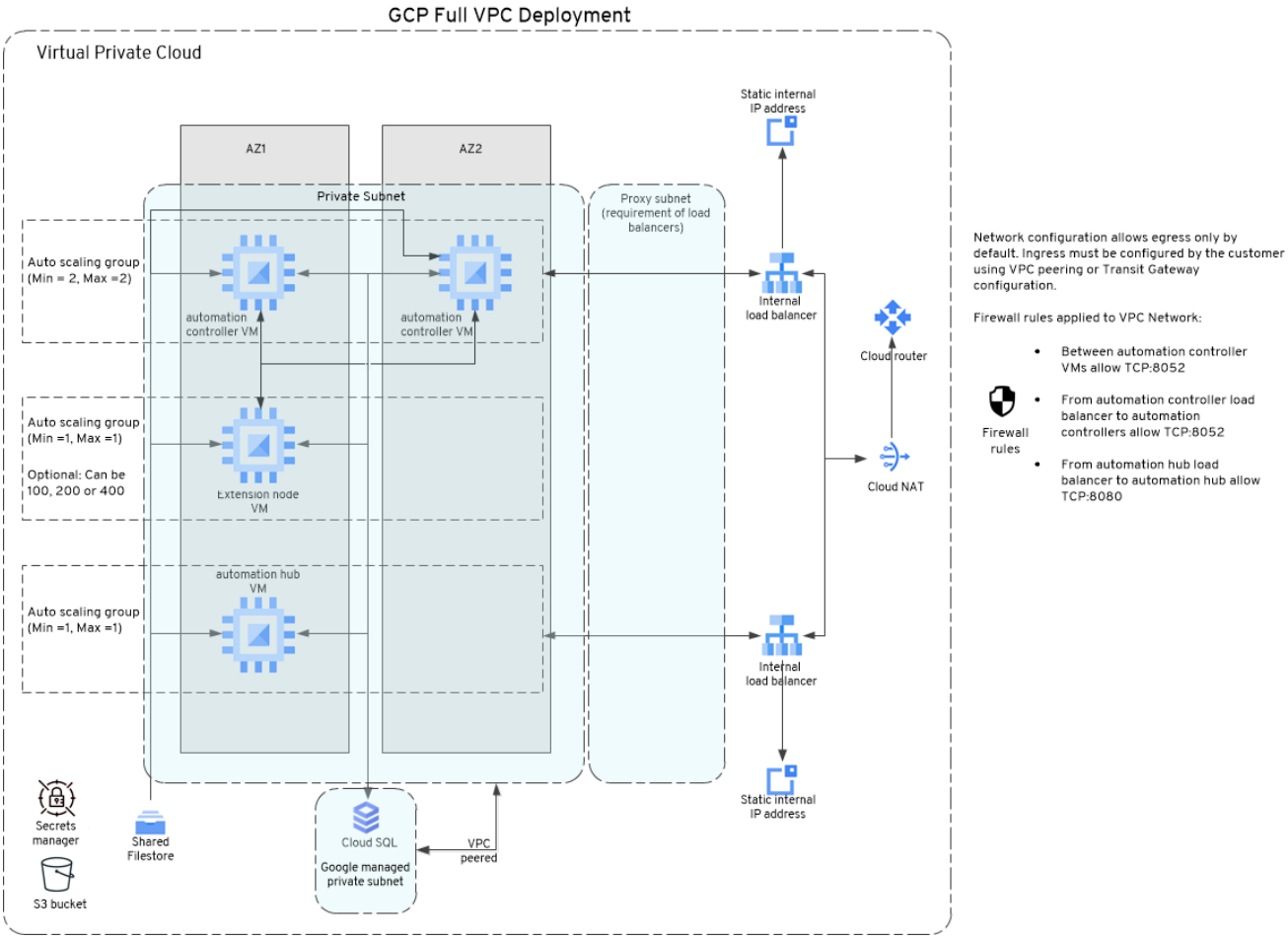
The Ansible Automation Platform software runs as containers on the deployed VM instances.

Managed instance groups (MIGs) manage VM instance lifecycles and monitor the health of each service running on the VM instances, automatically cycling the VM instances down and replacing them with new VM instances if the health check fails to respond, ensuring that the Ansible Automation Platform services stays up and available to process requests.

The VM instances run a customized *RedHat Enterprise Linux* (RHEL) Google Cloud Machine Image as their base image. This Google Cloud Machine Image is preloaded with all the required container images and packages to run the Ansible Automation Platform (automation hub, automation controller, and Execution Node components).

A shared Google File Store (GFS) volume is mounted into each VM instance provisioned by the product and is used for shared access to common files and resources.

A Google Cloud SQL Service is provisioned by the product at deployment time and contains databases for both the automation controller and automation hub.



The Foundation product includes two Execution Nodes running on the same VM instances as the automation controller components (this is called a Hybrid Node configuration in Ansible Automation Platform). Additional Execution Node offerings can be purchased to increase the scale (total number of managed nodes) the Ansible Automation Platform deployment is licensed to automate. When deploying the Execution Node offerings into an existing Ansible Automation Platform Foundation deployment, additional Execution Node VM instances can be deployed and automatically connected to the automation controller of the Foundation deployment where they immediately begin processing automation tasks.

Ansible Automation Platform components are run as containers using the Podman container runtime on the VM instances. The Podman runtime configuration is managed as a system service using **systemd** to ensure uptime and availability, and restarting any failed containers automatically.

SELinux is enabled on the VM instances and is supported down to the container level.

Additional operational automations are provided by the offering, available as a separate docker container for download from registry.redhat.io. These additional operational automations include backup, restore, and upgrade.

Any *Common Vulnerabilities and Exposures* (CVEs) found in the RHEL OS base image, the Ansible Automation Platform containers, or any included packages are addressed during upgrade of the Ansible Automation Platform offering by swapping out the base RHEL Google Cloud Machine Image with a newer version including all required software, packages, and containers.

This is done automatically for you through the use of the included upgrade automation.

Customers can take advantage of these operational automations to simplify the operational readiness of Ansible Automation Platform within their own corporate standards freeing themselves up to focus on developing Ansible Automation to manage their own infrastructure and applications rather than spending time developing automations to manage Ansible Automation Platform.

1.2. 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. INSTALLATION

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. 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.

2.2.1. 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**
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**

2.2.2. Create a 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.

2.2.3. Policies and permissions

Your GCP account must have the following *Identity and Access Management* (IAM) permissions to successfully create and manage Ansible Automation Platform deployments as well as the resources described in Application architecture.

Your GCP account must also be licensed to deploy Ansible Automation Platform from GCP Marketplace.

The application can fail to deploy if your IAM policies restrict deployment and management of these resources.

The application has two deployment options:

- Create a deployment with a new VPC.
- Create a deployment using an existing VPC.

Both options require the same minimum permissions

```
Cloud SQL Client
Cloud SQL Instance User
Compute Network Admin
Editor
Logs Writer
Secret Manager Secret Accessor
```

2.3. 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 periiod 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.

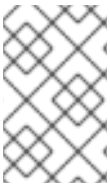
The calculation below will help you find the maximum length of the name of an Ansible Automation Platform deployment in your project.

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

There are two methods for deploying the application:

2.3.1. Deploying an application with a new VPC

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

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 **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. Optional: In the **Additional Labels** section, provide any additional label key and value pairs to be

added to the GCP resources that are part of the deployment. Valid keys and values must meet GCP label requirements. For a **key**, only hyphens, underscores, lowercase characters and numbers are permitted. A **key** must start with a lowercase character. For a **value**, only hyphens, underscores, lowercase characters and numbers are permitted.

10. Click **DEPLOY**.
11. The Deployment Manager displays the running deployment.
12. 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.

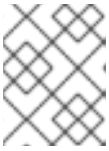
If you want to modify your network ranges post deployment, your current deployment must be deleted, then follow the instructions in Deploying an application with an existing VPC .

2.3.2. Deploying an application with an existing VPC

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

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.



NOTE

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

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. Optional: In the **Additional Labels** section, provide any additional label key and value pairs to be added to the GCP resources that are part of the deployment. Valid keys and values must meet GCP label requirements. For a **key**, only hyphens, underscores, lowercase characters and numbers are permitted. A **key** must start with a lowercase character. For a **value**, only hyphens, underscores, lowercase characters and numbers are permitted.

12. Click **DEPLOY**.

13. The **Deployment Manager** displays the running deployment.

14. 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. DEPLOYMENT INFORMATION

After deploying Ansible Automation Platform, use the following procedures to retrieve information about your deployment.

2.4.1. Retrieving the administration password

Use the following procedure to retrieve the administration password.

Procedure

1. In the GCP UI, select the main menu.
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.

2.4.2. Retrieving the load balancer addresses

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

Procedure

1. In the GCP UI, select the main menu.
2. Select **Deployment Manager**.
3. Select **Deployments**.
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. 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.

2.6. DEPLOYING AN EXTENSION NODE

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

Procedure

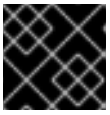
1. In the **Deployment name** field, enter a sufficiently short name as described in Application deployment.
2. For the **Service Account**, select the service account used with your Red Hat Ansible Automation Platform with up to 100 Managed Nodes deployment.
3. In the **Region** field, select the region 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.

NOTE

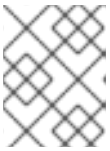
Main Deployment Name is a required field.

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-proxy-subnet**.



NOTE

In case of an error stating **Make sure all fields are correct to continue** and **You must select a Network**. Reselect the values from the menu and click **Done**.

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.

CHAPTER 3. DEPLOYING EXTENSION NODES

The foundation deployment is deployed with two automation controller nodes and one automation hub node by default. You can add execution nodes to scale out Ansible Automation Platform from GCP Marketplace. Extension node offers can be mixed and matched together to scale up and out the Ansible Automation Platform deployment.

You must create a backup of the Ansible Automation Platform deployment before deploying new extension nodes.

The procedure contains the following steps:

1. Decide the offer type for the extension nodes.
2. Ensure the minimum permissions are met to manage extension nodes.
3. Pull the **ansible-on-clouds-ops** container image.
4. Generate data files by running the **ansible-on-clouds-ops** container.
5. Populate the data file.
6. Run the **ansible-on-clouds-ops** container to deploy extension nodes.

Prerequisite

- A Linux or macOS system (where the ansible-on-clouds-ops container image will run).
- Docker

3.1. DECIDING THE OFFER TYPE

The following table lists the offer types and the corresponding GCP instance types. Depending on the workload needs, you can choose a more suitable offer type for the extension nodes.

Offer Type (nodes)	GCP Instance Type
100	n2-standard-2
200	n2-standard-4
400	n2-standard-8

Extension node offers can be mixed and matched together to scale the Ansible Automation Platform deployment up.

3.2. IAM MINIMUM PERMISSIONS

Your GCP account must have the following *Identity and Access Management* (IAM) permissions to successfully create and manage extension nodes on Ansible Automation Platform.

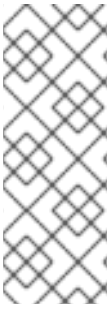
Your GCP account must also be licensed to deploy extension node offers for Ansible Automation Platform from GCP Marketplace.

```
Minimum Permissions -

Cloud SQL Client
Cloud SQL Instance User
Editor
Logs Writer
Secret Manager Secret Accessor
IAP-secured Tunnel User
```

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

Pull the Docker image for the Ansible on Clouds operational container with the same tag as the version you are deploying to. If you are unsure of the version you have deployed, see Command Generator and playbook **gcp_get_aoc_version** for more information on finding the current version of Ansible on Clouds deployment.



NOTE

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

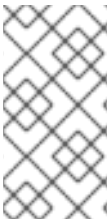
```
$ docker login registry.redhat.io
```

For more information about registry login, read Registry Authentication

For example, if your foundation deployment version is 2.4.20240215-00, you must pull the operational image with tag 2.4.20240215 to deploy extension nodes to the foundation deployment.

Use the following commands:

```
$ export IMAGE=registry.redhat.io/ansible-on-clouds/ansible-on-clouds-ops-rhel9:2.4.20240215
$ docker pull $IMAGE --platform=linux/amd64
```



NOTE

If your foundation deployment version is not 2.4.20240215-00, refer to the tables on the Released versions page for a matching deployment version, in the **Ansible on Clouds version** column. Find the corresponding operational image to use, in the **Ansible-on-clouds-ops container image** column, for the IMAGE environment variable.

3.4. 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 during the deployment of the extension nodes.

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_add_extension_nodes \
--output-data-file /data/extra_vars.yml
```

3. These commands create a **command_generator_data/extra_vars.yml** template file. This template file resembles the following:

```
gcp_add_extension_nodes:
  cloud_credentials_path:
  deployment_name:
  extra_vars:
    gcp_compute_region:
    gcp_extension_node_subscription:
    gcp_instance_group_name:
    gcp_instance_template_name:
    gcp_offer_type:
```

3.5. POPULATE THE DATA FILE

You must populate the data file before triggering the operation. The variables listed in the data file are defined below.

- **cloud_credentials_path** is the path for your Google Cloud service account credentials file. This must be an absolute path.
- **deployment_name** is the name of the AAP deployment manager deployment for which you want to create an extension node.
- **gcp_instance_group_name** is the name of the GCP instance group to create for the extension nodes.
- **gcp_instance_template_name** is the name of the GCP instance template to create.
- **gcp_offer_type** is the offer type of the extension node. This must be 100, 200 or 400.
- **gcp_compute_region** is the GCP region where the foundation deployment is deployed. This can be retrieved by checking the Deployments configuration in **Deployment Manager**.
- **gcp_extension_node_subscription** is the flag to confirm whether extension node subscription is purchased. Must be **true** or **false**.

3.6. DEPLOYING THE EXTENSION NODE

Procedure

1. To deploy the extension nodes, run the command generator to generate the CLI command.

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

Provides the following command:

```
---------------------------------------------
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 --env DEPLOYMENT_NAME=
<deployment_name> /
--env GENERATE_INVENTORY=true  $IMAGE
redhat.ansible_on_clouds.gcp_add_extension_nodes /
-e 'gcp_deployment_name=<deployment_name>
gcp_service_account_credentials_json_path=/home/runner/.gcp/credentials
gcp_compute_region=<region> gcp_instance_template_name=<instance_template_name> /
gcp_instance_group_name=<instance_group_name> gcp_offer_type=100
gcp_extension_node_subscription=True'
=============================================
```

2. Run the supplied command to add the extension nodes.

```
$ 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=leena1 /
--env GENERATE_INVENTORY=true  $IMAGE
redhat.ansible_on_clouds.gcp_add_extension_nodes /
-e 'gcp_deployment_name=<deployment_name>
gcp_service_account_credentials_json_path=/home/runner/.gcp/credentials
gcp_compute_region=<region> gcp_instance_template_name=<instance_template_name> /
gcp_instance_group_name=<instance_group_name> gcp_offer_type=100
gcp_extension_node_subscription=True'
```

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

```
TASK [redhat.ansible_on_clouds.standalone_gcp_add_extension_nodes :
[deploy_extension_nodes] Extension node created] ***
ok: [localhost] => {
    "msg": "Extension node is created for deployment test-ext1."
}

PLAY RECAP *********************************************************************
localhost                  : ok=39   changed=5    unreachable=0    failed=0    skipped=6
rescued=0    ignored=0
```

CHAPTER 4. REMOVING EXTENSION NODES

You must create a backup of the Ansible Automation Platform deployment before removing extension nodes.

Follow these steps to remove execution nodes from your Ansible Automation Platform from GCP Marketplace environment.

Prerequisites

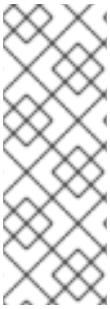
- Linux or macOS system (where the **ansible-on-clouds-ops** container image will run)
- Docker

Steps

1. Pull the **ansible-on-clouds-ops** container image.
2. Generate data files by running the ansible-on-clouds-ops container.
3. Update the data file.
4. Run the **ansible-on-clouds-ops** container to remove the extension nodes.

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

Pull the Docker image for the Ansible on Clouds operational container with the same tag as your foundation deployment. If you are unsure of the version you have deployed, see Command Generator and playbook **gcp_get_aoc_version** for more information on finding the current version of Ansible on Clouds deployment.



NOTE

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

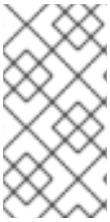
```
$ docker login registry.redhat.io
```

For more information about registry login, read Registry Authentication

For example, if your foundation deployment version is 2.4.20240215-00, you must pull the operational image with tag 2.4.20240215 to deploy extension nodes to the foundation deployment.

Use the following commands:

```
$ export IMAGE=registry.redhat.io/ansible-on-clouds/ansible-on-clouds-ops-rhel9:2.4.20240215
$ docker pull $IMAGE --platform=linux/amd64
```



NOTE

If your foundation deployment version is not 2.4.20240215-00, refer to the tables on the Released versions page for a matching deployment version, in the **Ansible on Clouds version** column. Find the corresponding operational image to use, in the **Ansible-on-clouds-ops container image** column, for the IMAGE environment variable.

4.2. 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 during the upgrade.

Procedure

1. Create a folder to hold the configuration files.

```
$ mkdir command_generator_data
```

2. Populate the **$(pwd)/command_generator_data** folder with the configuration file template.

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

When you have run these commands, a **command_generator_data/extra_vars.yml** template file is created. This template file resembles the following:

```
gcp_add_extension_nodes:
  cloud_credentials_path:
  deployment_name:
  extra_vars:
    gcp_compute_region:
    gcp_instance_group_name:
    gcp_instance_template_name:
```

4.3. POPULATE THE DATA FILE

You must populate the data file before triggering the operation. The variables listed in the data file are defined below.

- **cloud_credentials_path** is the path for your Google Cloud service account credentials file. This must be an absolute path.
- **deployment_name** is the name of the deployment for which you want to create an extension node.
- **gcp_instance_group_name** is the name of the GCP instance group to create for the extension nodes.
- **gcp_instance_template_name** is the name of the GCP instance template to create.

- **gcp_compute_region** is the GCP region where the foundation deployment is deployed. This can be retrieved by checking the Deployments config in Deployment Manager.

4.4. RUNNING THE ANSIBLE-ON-CLOUDS-OPS CONTAINER TO REMOVE THE EXTENSION NODES

Procedure

1. To remove a set of extension nodes, run the command generator to generate the CLI command.

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

   The command generator output provides the following command:

```
d----------------------------------------------
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 --env DEPLOYMENT_NAME=
<deployment_name> \
--env GENERATE_INVENTORY=true $IMAGE
redhat.ansible_on_clouds.gcp_remove_extension_nodes \
-e 'gcp_deployment_name=<deployment_name>
gcp_service_account_credentials_json_path=/home/runner/.gcp/credentials
gcp_compute_region=<region> gcp_instance_template_name=<instance_template_name> \
gcp_instance_group_name=<instance_group_name>'
==============================================
```

2. Run the supplied command to remove a set of extension nodes.

```
$ 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_remove_extension_nodes \
-e 'gcp_deployment_name=<deployment_name>
gcp_service_account_credentials_json_path=/home/runner/.gcp/credentials
gcp_compute_region=<region> gcp_instance_template_name=<instance_template_name> \
gcp_instance_group_name=<instance_group_name>'
```

If there are no errors, the extension node is removed successfully.

CHAPTER 5. 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.

5.1. 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

5.2. 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.

5.3. 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 traffic 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. Click **CONTINUE**.
5. In the **Name** field, enter the name of the peering connection that you need.
6. In the **Your VPC Network** field, select the first VPC that you plan to peer.
7. 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).
8. Click **Create**.

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

5.4. 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 **Application Load Balancer (HTTP/S)** tile and click **START CONFIGURATION**.
6. Select **From Internet to my VMs or serverless services** and **Global external Application 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.

    NOTE

    For more information, see SSL Certificates
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 **HTTPS** and **Named Port** to **https**.

d. Change **Timeout** to **86400**.

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 **Cloud CDN**. Unselect the checkbox for **Cloud CDN**.

j. Scroll down until you see a text box showing **Health check**.

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

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

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

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

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

- In the **Health Check** dialog box, set **Protocol** to **HTTPS**, and **Port** to **8080**.
- Set **Request** to **/api/galaxy/pulp/api/v3/status/**.

o. Scroll down to **Health criteria** section.

p. In the **Check interval** field, enter the value 15.

q. In the **Timeout** field, enter the value 10.

r. In the **Healthy threshold** field, enter the value 2.

s. In the **Unhealthy threshold** field, enter the value 10.

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

u. Click **CREATE**.

This returns you to **Backend configuration** section.

v. Click **OK**.

w. 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 6. ADDITIONAL CONFIGURATIONS

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

6.1. CHANGING THE DEFAULT 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. Navigate to the GCP Secrets Manager Console.
    a. Locate and open the secret for the Ansible Automation Platform deployment with the name **<deployment_name>-aap-admin**.
    b. Select **NEW VERSION** to add a new version.
    c. Enter a password secret value.
    d. Check **Disable all past versions** checkbox.
    e. Click **ADD NEW VERSION**.
2. Change 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 and automation hub Instance groups to create new VM instances.
3. The new administrator password can be used when the new automation controller and automation hub VM instances reach a *Running* Instance State.

6.2. REPLACING AUTOMATION CONTROLLER AND AUTOMATION HUB VM INSTANCES SSL/TLS CERTIFICATE AND KEY

By default, VM instances are secured with a self-signed SSL/TLS certificate with a validity period of ten years. When the certificate expires or you want VM instances to use your own certificate, you are required to replace the SSL/TLS certificate and key.

Procedure

1. Navigate to the GCP Secrets Manager Console.
    a. Locate and open the secret for the Ansible Automation Platform deployment with the name **<deployment_name>-pulp_cert**.
    b. Select **NEW VERSION** to add a new version.

c. Enter new SSL/TLS certificate value.

d. Check **Disable all past versions** checkbox.

e. Click **ADD NEW VERSION**.

2. Navigate to the GCP Secrets Manager Console.

a. Locate and open the secret for the Ansible Automation Platform deployment with the name **<deployment_name>-pulp_key**.

b. Select **NEW VERSION** to add a new version.

c. Enter new SSL/TLS key value.

d. Check **Disable all past versions** checkbox.

e. Click **ADD NEW VERSION**.

3. Change the running Ansible Automation Platform VM instances to use the new SSL/TLS certificate and key.

a. Navigate to the **GCP VM Instances** console.

b. Identify and delete all automation controller and automation hub VM instances for the Ansible Automation Platform deployment.

c. Wait for the automation controller and automation hub Instance groups to create new VM instances.

4. The new certificate is in use when the new automation controller and automation hub VM instances reach a *Running* Instance State.

6.3. 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. Generate the automation controller or automation hub certificate with the following command:

```
$ openssl req -x509 -nodes -newkey rsa:2048 -keyout key.pem -out cert.pem -sha256 -days
365
```

2. In the GCP console, navigate to the Load Balancing page.
3. In the search bar, enter the name of your deployment to filter down to your load balancers.
4. Click **<DEPLOYMENT_NAME>-aap-<cntrlr/hub>-int-lb**.
5. Click **Edit**.
6. Click **Frontend configuration**.
7. 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**
8. 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 **cert.pem** contents and paste it under **Certificate**.
    d. Using your previously generated certificate key, copy **key.pem** contents and paste it under **Private Key**.
    e. Click **Create**.
9. Click **Done**.
10. 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.
11. Click **Update**, and confirm the update.

6.4. 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.

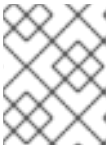
Securing infrastructure services is an important step in any cloud deployment. Follow the implementation and security suggestions from GCP documentation for services used as part of an Ansible Automation Platform from GCP Marketplace deployment.

CHAPTER 7. 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

7.1. 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. If you are unsure of the version you have deployed, see Command Generator and playbook **gcp_get_aoc_version** for more information on finding the currention of Ansible on Clouds deployment.

NOTE

Before Pulling the docker image, ensure you are logged in to registry.redhat.io using docker. Use the following command to login to registry.redhat.io.

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

For more information about registry login, see Registry Authentication

For example, if your foundation deployment version is 2.4.20240215-00, you must pull the operational image with tag 2.4.20240215.

Use the following commands:

```
$ export IMAGE=registry.redhat.io/ansible-on-clouds/ansible-on-clouds-ops-rhel9:2.4.20240215
$ docker pull $IMAGE --platform=linux/amd64
```

NOTE

If your foundation deployment version is not 2.4.20240215-00, refer to the tables on the Released versions page for a matching deployment version, in the column **Ansible on Clouds version**, and find the corresponding operational image to use, in the column **Ansible-on-clouds-ops container image**, for the IMAGE environment variable.

7.2. 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: gcp_aap_health_check
Playbook: gcp_add_extension_nodes
Playbook: gcp_add_labels
Playbook: gcp_backup_delete
Playbook: gcp_backup_deployment
Playbook: gcp_backup_list
Playbook: gcp_backups_delete
Playbook: gcp_check_aoc_version
Playbook: gcp_deployment_inventory
Playbook: gcp_get_aoc_version
Playbook: gcp_health_check
Playbook: gcp_list_deployments
Playbook: gcp_nodes_health_check
Playbook: gcp_remove_extension_nodes
Playbook: gcp_remove_labels
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>'
=============================================
```

7.3. 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:

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

7.4. 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
```

7.5. 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.

7.6. USING THE PLAYBOOKS

Some, but not all of the playbooks are described in this document. Here, those which are used either to retrieve information from the Ansible on Clouds deployment or to check it are described. These playbooks do not modify the deployment.

gcp_aap_health_check

This playbook checks if the Ansible application is healthy.

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

Which generates the following output:

```
================================================
Playbook: gcp_aap_health_check
Description: This playbook checks if the deployment is healthy using the Ansible health service.
------------------------------------------------
The health check consists of checking the Ansible Automation Platform from GCP Marketplace
environemnt to verify it is healthy.

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

docker run --rm $IMAGE command_generator gcp_aap_health_check [--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>'
================================================
```

Launching this command by replacing the parameters generates a new command to launch and outputs:

```
...
PLAY RECAP *********************************************************************
localhost                  : ok=29   changed=1    unreachable=0    failed=0    skipped=0    rescued=0
ignored=0
```

A *failed* not equal zero indicates an issue with Ansible on Cloud deployment.

gcp_add_labels

This playbook adds labels to the deployment.

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

Which generates the following output:

```
================================================
Playbook: gcp_add_labels
Description: This playbook adds labels to the deployment
------------------------------------------------
Add labels to the Ansible Automation Platform from GCP Marketplace deployment

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

docker run --rm $IMAGE command_generator gcp_add_labels -d <deployment_name> -c
```

```
<cloud_credentials_path> --extra-vars 'gcp_compute_region=<gcp_compute_region>
gcp_compute_zone=<gcp_compute_zone> gcp_labels=<gcp_labels>'
================================================
```

The parameter **gcp_labels** is a comma separated list of **key=value** pairs to add or update. For example: key1=value1,key2=value2

Launching this command by replacing the parameters generates a new command to launch and outputs:

```
...
PLAY RECAP *********************************************************************
localhost                  : ok=22   changed=2    unreachable=0    failed=0    skipped=1    rescued=0
ignored=0
```

gcp_remove_labels

This playbook removes labels from the deployment.

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

Which generates the following output:

```
================================================
Playbook: gcp_remove_labels
Description: This playbook removes labels from the deployment.
------------------------------------------------
Remove labels from the Ansible Automation Platform from GCP Marketplace deployment.

------------------------------------------------
Command generator template:
docker run --rm $IMAGE command_generator gcp_remove_labels -d <deployment_name> -c
<cloud_credentials_path> --extra-vars 'gcp_compute_region=<gcp_compute_region>
gcp_compute_zone=<gcp_compute_zone> gcp_labels=<gcp_labels>'
================================================
```

The parameter **gcp_labels** is a comma separated list of keys to remove. For example: key1,key2

Launching this command by replacing the parameters generates a new command to launch and outputs:

```
...
PLAY RECAP *********************************************************************
localhost                  : ok=22   changed=2    unreachable=0    failed=0    skipped=1    rescued=0
ignored=0
```

gcp_check_aoc_version

This playbook checks if the Ansible on Cloud version is the same as the command generator container. The check is done each time a playbook is called.

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

Which generates the following output:

```
================================================
Playbook: gcp_check_aoc_version
Description: Check the operational container version matches the Ansible on Clouds version.
```

```
---------------------------------------------
Check the operational container version matches the Ansible on Clouds version.

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

docker run --rm $IMAGE command_generator gcp_check_aoc_version [--ansible-config
ansible_config_path>] -c <cloud_credentials_path> -d <deployment_name>
=============================================
```

Launching this command by replacing the parameters generates a new command to launch and outputs:

```
...
TASK [redhat.ansible_on_clouds.standalone_check_aoc_version : Verify operational playbook and
Ansible on Clouds deployment versions] ***
ok: [localhost] => {
    "changed": false,
    "msg": "This operation playbook version and the Ansible on Clouds deployment version are
identical: 2.4.20230606-00"
}

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

A *failed* not equal zero means the Ansible on Clouds deployment version does not match the **command_generator** container and a different version is required for the command generator to manage that deployment.

gcp_get_aoc_version

This playbook retrieves the version of the Ansible on Clouds deployment.

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

Which generates the following output:

```
=============================================
Playbook: gcp_get_aoc_version
Description: Get the current Ansible on Clouds version.
---------------------------------------------
Get the current Ansible on Clouds version.

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

docker run --rm $IMAGE command_generator gcp_get_aoc_version [--ansible-config
ansible_config_path>] -c <cloud_credentials_path> -d <deployment_name>
=============================================
```

Launching this command by replacing the parameters generates a new command to launch and outputs:

```
...
TASK [Print version] ***********************************************************
ok: [localhost] => {
```

```
    "msg": "The AOC version is 2.4.20230606-00"
}

PLAY RECAP *********************************************************************
localhost                  : ok=5    changed=0    unreachable=0    failed=0    skipped=0    rescued=0
ignored=0
```

gcp_health_check

This playbook checks if the nodes and Ansible application are healthy.

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

Which generates the following output:

```
================================================
Playbook: gcp_health_check
Description: This playbook checks if the Ansible Automation Platform from GCP Marketplace
deployment is healthy.
------------------------------------------------
The health check consists of checking the Ansible Automation Platform from GCP Marketplace heatlh
checks
and the health of the monitoring exporter.

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

docker run --rm $IMAGE command_generator gcp_health_check [--ansible-config
ansible_config_path>] -c <cloud_credentials_path> -d <deployment_name> --extra-vars
'gcp_compute_region=<gcp_compute_region> gcp_compute_zone=<gcp_compute_zone>'
================================================
```

Launching this command by replacing the parameters will generate a new command to launch and will output:

```
...
PLAY RECAP *********************************************************************
localhost                  : ok=47   changed=1    unreachable=0    failed=0    skipped=0    rescued=0
ignored=0
```

A *failed* not equal zero indicates an issue with nodes or the Ansible on Cloud deployment.

gcp_list_deployments

This playbook list the deployments, the region and zone are optional.

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

Which generates the following output:

```
================================================
Playbook: gcp_list_deployments
Description: This playbook generates a list of available Ansible Automation Platform from GCP
Marketplace deployments.
------------------------------------------------
This playbook is used to generate a list of available Ansible Automation Platform from GCP
```

```
Marketplace deployments.

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

docker run --rm $IMAGE command_generator gcp_list_deployments -c <cloud_credentials_path> --
extra-vars '[gcp_compute_region=<gcp_compute_region>] [gcp_compute_zone=
<gcp_compute_zone>]'
=============================================
```

Launching this command by replacing the parameters generates a new command to launch and outputs:

```
...
TASK [Show deployment list] ***************************************************
ok: [localhost] => {
    "msg": [
        "Deployment list: ['dep1', 'dep2', 'dep3']"
    ]
}

PLAY RECAP ********************************************************************
localhost                  : ok=7    changed=0    unreachable=0    failed=0    skipped=0    rescued=0
ignored=0
```

gcp_nodes_health_check

This playbook checks if the nodes are healthy.

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

Which generates the following output:

```
=============================================
Playbook: gcp_nodes_health_check
Description: This role runs a health check on a group of nodes in the Ansible Automation Platform
from GCP Marketplace deployment
---------------------------------------------
The playbook checks if the Ansible Automation Platform from GCP Marketplace monitoring exporter
is up and running.

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

docker run --rm $IMAGE command_generator gcp_nodes_health_check [--ansible-config
ansible_config_path>] -d <deployment_name> -c <cloud_credentials_path> --extra-vars
'check_monitoring=True'
=============================================
```

Launching this command by replacing the parameters generates a new command to launch and outputs:

```
...
PLAY RECAP ********************************************************************
localhost                  : ok=47   changed=1    unreachable=0    failed=0    skipped=0    rescued=0
ignored=0
```

A *failed* not equal zero indicates an issue with nodes in the deployment.

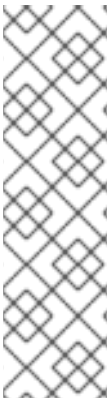
CHAPTER 8. AUTOMATION WORKLOADS

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

8.1. 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

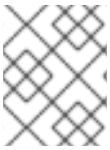
Ansible Automation Platform from GCP Marketplace is driven using three n2-standard-2 instances, two of which run automation controller and one which runs automation hub. The automation controller instances collectively support a standard workload for 100 managed active nodes. Red Hat has tested this 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.

8.2. 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

Red Hat does not support environments that are extended through customer design and implementation.

CHAPTER 9. 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, read Setting up monitoring and logging at deployment time , or
- After the deployment

9.1. 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.

9.1.1. Required permissions

You must have the following GCP IAM permissions to set up logging and monitoring:

```
required-roles:

Service Account User
Compute Instance Admin (v1)

required-permissions:

cloudsql.instances.connect
cloudsql.instances.get
cloudsql.instances.login
cloudsql.users.update
compute.addresses.get
compute.addresses.list
compute.instances.delete
compute.instances.get
compute.instances.list
compute.instances.setLabels
compute.zoneOperations.get
deploymentmanager.deployments.list
deploymentmanager.manifests.get
deploymentmanager.manifests.list
file.instances.get
file.instances.list
file.instances.update
file.operations.get
iap.tunnelInstances.accessViaIAP
logging.logEntries.create
monitoring.timeSeries.create
resourcemanager.projects.get
runtimeconfig.variables.create
runtimeconfig.variables.get
runtimeconfig.variables.list
runtimeconfig.variables.update
```

```
secretmanager.secrets.create
secretmanager.secrets.delete
secretmanager.secrets.get
secretmanager.versions.add
secretmanager.versions.get
secretmanager.versions.list
servicenetworking.operations.get
servicenetworking.services.addPeering
serviceusage.services.list
```

9.1.2. 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. If you are unsure of the version you have deployed, see Command Generator and playbook **gcp_get_aoc_version** for more information on finding the current version of Ansible on Clouds deployment.

NOTE

Before pulling the docker image, ensure you are logged in to registry.redhat.io using docker. Use the following command to login to registry.redhat.io.

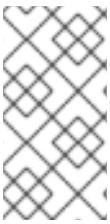
```
$ docker login registry.redhat.io
```

For more information about registry login, see Registry Authentication

For example, if your foundation deployment version is 2.4.20240215-00, you must pull the operational image with tag 2.4.20240215.

Use the following commands:

```
$ export IMAGE=registry.redhat.io/ansible-on-clouds/ansible-on-clouds-ops-rhel9:2.4.20240215
$ docker pull $IMAGE --platform=linux/amd64
```



NOTE

If your foundation deployment version is not 2.4.20240215-00, refer to the tables on the Released versions page for a matching deployment version, in the **Ansible on Clouds version** column. Find the corresponding operational image to use, in the **Ansible-on-clouds-ops container image** column, for the IMAGE environment variable.

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

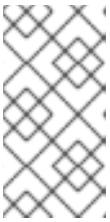
The following commands generate the required data file. These commands create a directory, and an empty data template that, when populated, is used to generate the playbook.

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.



NOTE

On Linux, any file or directory created by the command generator is owned by **root:root** by default. To change the ownership of the files and directories, you can run the **sudo chmod** command after the files are created. For more information, read Command generator - Linux files owned by root .

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

9.1.4. 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** (Optional) the type of component on which you want to carry out 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** (Optional) is set to **true** to enable the monitoring, **false** otherwise. Default = **false**.
- **logging_enabled** (Optional) is set to **true** to enable the logging, **false** otherwise. Default = **false**.
- **default_collector_interval** (Optional) is the frequency at which the monitoring data must be sent to 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**.

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

The following values are provided as examples:

NOTE

The optional parameters described in the section are ommitted in the data file example below. The playbook uses the default value for any optional parameter that is ommitted from the data file. If you want to override a default value for an optional parameter, then it must be included in the data file and assigned a value.

```
gcp_setup_logging_monitoring:
  cloud_credentials_path: ~/secrets/GCP-secrets.json
  deployment_name: AnsibleAutomationPlatform
  extra_vars:
```

9.1.5. 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
```

9.2. CUSTOMIZING MONITORING AND LOGGING

Metrics are provided by Ansible, Podman and Google Ops Agent. The Google Ops Agent and Podman are installed on automation controller and automation hub VM instances, however Ansible metrics are only installed on automation hub instances.

A configurable process (collector) runs on each automation controller VM instance and automation hub VM instance 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.

9.2.1. 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.

9.2.2. 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**.

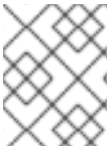
A crontab job restarts the agent if **gcp-ops-agent-config.yml** changed in the last 10 minutes. The agent rereads its configuration after a restart.

CHAPTER 10. BACKUP AND RESTORE

IMPORTANT

- You must restore with the same operational image version as the backup.
- 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.

10.1. 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 you performed the backup 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, read 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.

The bucket can contain multiple backups from the same deployment or different ones. A backup will generate a directory named <prefix>-<deployment_name>-<timestamp> and a file named <prefix>-<deployment_name>-<timestamp>.json. The directory contains the awx and pulp databases backup, the deployment configuration and the secrets. The json file contains information for the restore procedure.

Playbooks are provided through the CLI to list and delete backups.

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

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

Procedure

- Pull the docker image for the **ansible-on-clouds-ops** container with the same tag as the foundation deployment. If you are unsure of the version you have deployed, see Command Generator and playbook **gcp_get_aoc_version** for more information on finding the current version of Ansible on Clouds deployment.

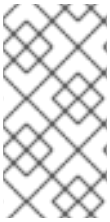
NOTE

Before pulling the docker image, ensure you are logged in to registry.redhat.io using docker. Use the following command to login to registry.redhat.io.

```
$ 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-rhel9:2.4.20240215
$ docker pull $IMAGE --platform=linux/amd64
```



NOTE

If your foundation deployment version is not 2.4.20240215-00, refer to the tables on the Released versions page for a matching deployment version, in the **Ansible on Clouds version** column. Find the corresponding operational image to use, in the **Ansible-on-clouds-ops container image** column, for the IMAGE environment variable.

10.1.2. Required permissions

You must have the following GCP IAM permissions to backup the stack:

```
required-roles:

Service Account User
Compute Instance Admin (v1)
required-permissions:

compute.instances.list
deploymentmanager.deployments.get
deploymentmanager.manifests.get
deploymentmanager.manifests.list
deploymentmanager.resources.list
file.backups.create
file.operations.get
iap.tunnelInstances.accessViaIAP
storage.objects.create
storage.objects.list
```

10.1.3. Setting up the environment

Procedure

- Create a folder to hold the configuration files.

```
$ mkdir command_generator_data
```

10.1.4. Backup requirements

You must create a bucket to store the Ansible on Clouds deployment backup. The bucket contains the backup information, the secrets and the databases backups.

Procedure

1. In the Google Cloud console, navigate to **Cloud Storage → Buckets**
2. Choose the project.
3. Click **Create**.
4. Enter a name.
5. Enter the data location that fits the most your requirements. Multi and Dual regions enable you to run a restore to another region.
6. Enter the storage class that fits the most your requirements.
7. Enter the control access that fits the most your requirements.
8. Enter the data protection that fits the most your requirements.
9. Click **Create**

Troubleshooting

An error is raised if the bucket name is already used in another project.

10.1.5. Creating the backup data file

Procedure

1. Run the command generator **command_generator_vars** to generate the **backup.yml**.

NOTE

On Linux, any file or directory created by the command generator is owned by **root:root** by default. To change the ownership of the files and directories, you can run the **sudo chmod** command after the files are created. For more information, read Command generator - Linux files owned by root .

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

2. After running the command, a **$(pwd)/command_generator_data/backup.yml** template file is created. This template file resembles the following:

```
gcp_backup_deployment:
  cloud_credentials_path:
```

```
deployment_name:
extra_vars:
  backup_prefix: aoc-backup
  gcp_bucket_backup_name:
  gcp_compute_region:
  gcp_compute_zone:
```

10.1.6. 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.
- **deployment_name** is the name of the AAP deployment manager deployment you want to back up.
- **backup_prefix** is a prefix you would like to add to the backup name (default: aoc-backup)
- **gcp_bucket_backup_name** is the bucket that was previously created to use for the 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.

10.1.7. Running the backup playbook

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 --env DEPLOYMENT_NAME=
<deployment_name --env GENERATE_INVENTORY=true  \
$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> backup_prefix=aoc-backup'
```

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 --env DEPLOYMENT_NAME=
<deployment_name --env GENERATE_INVENTORY=true  \
$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> backup_prefix=aoc-backup'
```

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

```
TASK [redhat.ansible_on_clouds.standalone_gcp_backup : [backup_deployment] Print the
variable required to restore deployment my-deployment] ***
ok: [localhost] => {
    "msg": [
        "AAP on GCP Backup successful. Please note below the bucket name and backup
name which are required for restore process.",
        "gcp_bucket_backup_name: my-bucket",
        "backup_name: aoc-backup-my-deployment-20230616T134002"
    ]
}

PLAY RECAP *********************************************************************
localhost                  : ok=38   changed=6    unreachable=0    failed=0    skipped=1
rescued=0    ignored=0
```

10.1.8. List backups

This playbook enables you to list the existing backups in a specific bucket.

Procedure

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

NOTE

On Linux, any file or directory created by the command generator is owned by **root:root** by default. To change the ownership of the files and directories, you can run the **sudo chmod** command after the files are created. For more information, read technical notes.

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

2. After running the command, a **$(pwd)/command_generator_data/backups_list.yml** template file is created. This template file resembles the following:

```
gcp_backup_list:
  cloud_credentials_path:
  extra_vars:
    gcp_bucket_backup_name:
```

3. 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_list --data-file /data/backups_list.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_list \
-e 'gcp_service_account_credentials_json_path=/home/runner/.gcp/credentials
gcp_bucket_backup_name=<bucket>'
```

4. Run the supplied backup command to trigger the list of backups.

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

```
TASK [redhat.ansible_on_clouds.standalone_gcp_backup_list : [list_backup] Display list of
backups] ***
ok: [localhost] => {
    "msg": [
        "aoc-backup-deployment1-20230614T203926",
        "aoc-backup-deployment1-20230616T114134",
        "aoc-backup-deployment1-20230616T134002",
        "aoc-backup-deployment2-20230613T124127"
    ]
}

PLAY RECAP *********************************************************************
localhost                  : ok=11   changed=0    unreachable=0    failed=0    skipped=0
rescued=0    ignored=0
```

10.1.9. Delete backups

There are two playbooks to delete backups:

- Use the **gcp_backup_delete** playbook which deletes a single backup.
- Use the **gcp_backups_delete** playbook which deletes multiple backups at once.

gcp_backups_delete takes a array of strings ["backup1","backup2",...], while **gcp_backup_delete** takes only one string, that being the name of a specific backup, "backup1".

The use of **gcp_backups_delete** is described in this section.

Procedure

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

NOTE

On Linux, any file or directory created by the command generator is owned by **root:root** by default. To change the ownership of the files and directories, you can run the **sudo chmod** command after the files are created. For more information, read Command generator - Linux files owned by root .

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

2. After running the command, a **$(pwd)/command_generator_data/backups_delete.yml** template file is created. This template file resembles the following:

```
gcp_backups_delete:
  cloud_credentials_path:
  extra_vars:
    backup_names:
    delete:
    gcp_bucket_backup_name:
```

The **backup_names** parameter must specify an array of strings, for example, **["backup1","backup2"]**. The **delete** parameter must be set to **true** to successfully delete.

1. To delete the backups, run the command generator to generate the **gcp_backups_delete`** command.

```
docker run --rm -v $(pwd)/command_generator_data:/data $IMAGE command_generator
gcp_backups_delete --data-file /data/backups_delete.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_backups_delete \
-e 'gcp_service_account_credentials_json_path=/home/runner/.gcp/credentials
gcp_bucket_backup_name=<bucket> \
backup_names=<backup_names> delete=True'
```

2. Run the supplied backup command to delete the backups.

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

```
TASK [redhat.ansible_on_clouds.standalone_gcp_backup_delete : [delete_backup] Dry-run
message] ***
skipping: [localhost]

PLAY RECAP *********************************************************************
localhost                  : ok=23   changed=2    unreachable=0    failed=0    skipped=2
rescued=0    ignored=0
```

10.1.10. Fixing a failed backup deletion

If the deletion of a backup failed, take the following actions:

Procedure

1. Navigate to the bucket containing the backup.
2. Locate the directory that has the name of the backup.
3. Open the backup directory.
4. Delete the directory with the name of the backup.
5. Delete the file with the backup name with **.json** as extension.
6. Navigate to **Filestore → Backup**.
7. Delete the Filestore backup with the same name as the backup.

10.2. RESTORE PROCESS

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

IMPORTANT

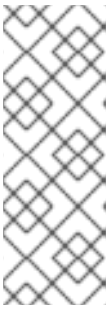
- You must restore with the same operational image version which was used for the backup.

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

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

Procedure

- Pull the docker image for the **ansible-on-clouds-ops** container with the same tag as the backup was created with. If you are unsure of the version backed up, you can look in the backup json file for the key **restore_properties.aoc_version**.



NOTE

Before pulling the docker image, ensure you are logged in to registry.redhat.io using docker. Use the following command to login to registry.redhat.io.

```
$ 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-rhel9:2.4.20240215
$ docker pull $IMAGE --platform=linux/amd64
```

-

NOTE

If your backup version is not 2.4.20240215-00, refer to the tables on the Released versions page to for a matching version, in the **Ansible on Clouds version** column. Find the corresponding operational image to use, in the **Ansible-on-clouds-ops container image** column, for the IMAGE environment variable.

10.2.2. Setting up the environment

Procedure

- Create a folder to hold the configuration files.

```
$ mkdir command_generator_data
```

10.2.3. Required permissions

You must have the following GCP IAM permissions to restore the stack:

```
required-roles:

Cloud SQL Client
Cloud SQL Instance User
Compute Instance Admin (v1)
Compute Network Admin
Editor
Logs Writer
Secret Manager Secret Accessor
Service Account User
required-permissions:

compute.instances.list
compute.networks.create
deploymentmanager.deployments.create
deploymentmanager.deployments.get
deploymentmanager.operations.get
file.instances.create
file.operations.get
iap.tunnelInstances.accessViaIAP
secretmanager.secrets.create
secretmanager.secrets.delete
secretmanager.secrets.get
secretmanager.secrets.update
secretmanager.versions.add
secretmanager.versions.list
storage.objects.get
storage.objects.list
```

10.2.4. Generating the restore.yml file

Procedure

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

NOTE

On Linux, any file or directory created by the command generator is owned by **root:root** by default. To change the ownership of the files and directories, you can run the **sudo chmod** command after the files are created. For more information, read technical notes.

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

2. After running the command, a **$(pwd)/command_generator_data/restore.yml** template file is created. This template file resembles the following:

```
================================================
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/index

------------------------------------------------
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:
    backup_name:
    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:
```

10.2.5. 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.
- **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.
- **backup_name** is the name of the backup in the bucket. This name is displayed while using the **gcp_backup_deployment** or **gcp_backup_list** commands.
- **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.

For an existing VPC

If you want to restore using an existing VPC, you must set the parameters shown above.

You must also set the following additional parameters:

- **gcp_existing_vpc** is set to **true**.
- **gcp_filestore_ip_range** must be set to a free ip/29 range of your VPC. For example: 192.168.245.0/29. You must not use 192.168.243.0/29 as this the default used when deploying Ansible Automation Platform from GCP Marketplace.
- **gcp_cloud_sql_peering_network** must be set to a free **/24** subnet. You must not use 192.168.241.0/24 as this is used during the original deployment.
- **gcp_controller_internal_ip_address** must be set to a free IP in your VPC network.
- **gcp_hub_internal_ip_address** must set to a free IP in your VPC network.

10.2.6. Running the restore command

When **$(pwd)/command_generator_data/restore.yml** is populated, you can use the command generator to create the restore command.

Procedure

1. Run the command generator.

```
$ 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 --env DEPLOYMENT_NAME=
<deployment_name> --env GENERATE_INVENTORY=true -\
-env CHECK_GENERATED_INVENTORY=false  $IMAGE
redhat.ansible_on_clouds.gcp_restore_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> backup_name=<backup_name> gcp_existing_vpc=
<existing_vpc>'
```

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
```

10.2.7. Failure to restore

If a message similar to the following appears during the restore, you must contact support as the restore must be done manually.

```
TASK [redhat.ansible_on_clouds.standalone_gcp_restore : [restore_deployment] Restore awx db] *
fatal: [localhost -> dvernier-restore1-aap-cntrlr-x2c6]: FAILED!
```

CHAPTER 11. UPGRADING

You can upgrade your existing Ansible Automation Platform deployment to the newer version. The upgrade process covers upgrading the automation hub, automation controller, and extension nodes. The upgrade process takes roughly the same amount of time as a Ansible Automation Platform deployment install. You are required to create a backup before running the upgrade.

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 Ansible Automation Platform to 2.4.20240215, you must be on version 2.4.20231024.

Logging and monitoring must be disabled before running the upgrade. Follow these instructions to toggle off monitoring and logging for your current version before beginning the upgrade.

When the upgrade has completed, logging and monitoring can be reenabled by following these instructions.

Prerequisites

- Docker must be installed to run the upgrade playbook.
- A Linux or macOS system, where the **ansible-on-clouds-ops** container image will run.
- 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 stack.
   - Follow the Ansible on Clouds backup instructions.
2. Upgrade Ansible Automation Platform
   a. Pull the next sequential ansible-on-clouds-ops version container image
   b. Ensure minimum required permissions are met
   c. Generate the data file
   d. Update the data file
   e. Begin upgrading Ansible Automation Platform by running the operational container
3. (Optional) Restore the stack from a backup.
   - Follow the Ansible on Clouds restore instructions.

11.1. BACKUP BEFORE UPGRADE

IMPORTANT

Before starting the upgrade of your Ansible Automation Platform environment, you must first take a backup of your environment at its current version.

To backup your Ansible Automation Platform environment at the earlier version, follow the Ansible on Clouds backup instructions.

11.2. UPGRADE ANSIBLE AUTOMATION PLATFORM

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

Procedure

- Pull the Docker image for Ansible on Clouds operational container with the tag based on the version you want to upgrade to. If you are unsure of the version you have deployed and the one you need to upgrade to, see Command Generator and playbook **gcp_get_aoc_version** for more information on finding the current version of Ansible on Clouds deployment and the version to upgrade to.

NOTE

Before pulling the docker image, ensure you are logged in to registry.redhat.io using docker. Use the following command to login to registry.redhat.io.

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

For more information about registry login, see Registry Authentication

NOTE

The Ansible on Clouds operational image tag must match the version that you want to upgrade to. For example, if your foundation deployment version is 2.4.20231024, pull the operational image with tag 2.4.20240215 to upgrade to version 2.4.20240215.

```
$ export IMAGE=registry.redhat.io/ansible-on-clouds/ansible-on-clouds-ops-rhel9:2.4.20240215
$ docker pull $IMAGE --platform=linux/amd64
```

NOTE

If your foundation deployment version is not 2.4.20240215-00, refer to the tables on the Released versions page for a matching deployment version, in the **Upgrade from version** column. Find the corresponding operational image to use, in the **Ansible-on-clouds-ops container image** column, for the IMAGE environment variable.

11.2.2. Required permissions

You must have the following GCP IAM permissions to upgrade the stack:

```
required-roles:
  - Service Account User
  - Compute Instance Admin (v1)
required-permissions:
  - compute.healthChecks.update
  - compute.healthChecks.use
  - compute.healthChecks.useReadOnly
  - compute.regionBackendServices.update
  - iap.tunnelInstances.accessViaIAP
  - runtimeconfig.variables.get
  - secretmanager.locations.get
  - secretmanager.locations.list
  - secretmanager.secrets.create
  - secretmanager.secrets.delete
  - secretmanager.secrets.get
  - secretmanager.secrets.list
  - secretmanager.secrets.update
  - secretmanager.versions.access
  - secretmanager.versions.add
  - secretmanager.versions.disable
  - secretmanager.versions.enable
  - secretmanager.versions.get
  - secretmanager.versions.list
```

11.2.3. Generating the data file

The commands in this section create a directory and populate it with an empty data template that, when populated, is used during the upgrade.

Procedure

1. Run the following commands to generate the required data file.

```
# Create a folder to hold the configuration files
$ mkdir command_generator_data
```

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

NOTE

On Linux, any file or directory created by the command generator is owned by **root:root** by default. To change the ownership of the files and directories, you can run the **sudo chmod** command after the files are created. For more information, read Command generator - Linux files owned by root .

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

3. After running these commands, a **command_generator_data/extra_vars.yml** template file is created. This template file resembles the following:

```
gcp_upgrade:
```

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

11.2.4. Updating the data file

You must populate the data file before triggering the upgrade. Each parameter is required unless it is noted as optional. If the parameter is noted as optional, both its key and value can be removed from 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
```

11.2.5. Running the upgrade playbook

NOTE

Ansible Automation Platform upgrade to 2.4.20240215 updates the protocol of its internal load balancers. If additional networking configurations were added after installation, they can also require updating to ensure connectivity. For more information, read the Upgrade note.

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: 2.4.20231024-00"
}
```

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.

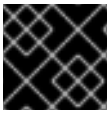
11.3. RESTORE FROM BACKUP

If you would like to restore the earlier version of your Ansible Automation Platform environment to the state it was in before the upgrade, follow the Ansible on Clouds 2.3 restore instructions.

CHAPTER 12. UNINSTALLING

This chapter explains how to uninstall Ansible Automation Platform.

12.1. REMOVING 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. Check the checkbox for the extension node you want to delete.

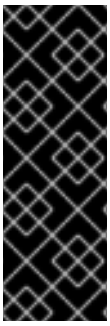
NOTE

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.

12.2. UNINSTALLING ANSIBLE AUTOMATION PLATFORM



IMPORTANT

This action cannot be reversed.

If an external Load Balancer was added for the deployment, it must be deleted before uninstalling the deployment. For more information, see Load Balancers.

If extension nodes were added to the deployment, they 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.

It can take some time to complete the deletion.

12.3. REMOVING UPGRADE RESOURCES

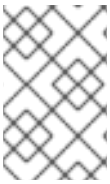
The following procedures describes how to delete any resources left behind after an upgrade.

NOTE

These actions cannot be reversed.

You must only perform these actions if your deployment has been deleted.

12.3.1. Removing the secrets



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

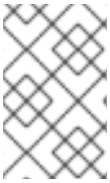
Procedure

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**.

12.4. REMOVING BACKUP RESOURCES

The following procedures describe how to delete any resources left behind when uninstalling your Ansible Automation Platform deployment, after having run a backup on it.



NOTE

These actions cannot be reversed.

You must only perform these actions if your deployment has been deleted.

12.4.1. Removing the filestore

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**.

12.4.2. Removing the storage bucket

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**.

12.5. REMOVING REMAINING RESOURCES

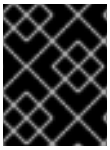
The following procedures describes how to delete any resources left behind when uninstalling your Ansible Automation Platform deployment.

NOTE

These actions cannot be reversed.

You must only perform these actions if your deployment has been deleted.

12.5.1. Removing 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**.

CHAPTER 13. 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.

13.1. UPGRADE - LOGGING AND MONITORING

To ensure a successful upgrade, logging and monitoring must be disabled before running the upgrade. Follow these instructions to toggle off monitoring and logging for your current version before beginning the upgrade. when the upgrade has completed, logging and monitoring can be reenabled by following these instructions.

13.2. COMMAND GENERATOR - LINUX FILES OWNED BY ROOT

On Linux, any file or directory created by the command generator is owned by **root:root** by default. To change the ownership of the files and directories, you can run the **sudo chmod** command after the files are created:

```
# Change the owner of the command_generator_data directory recursively
$ sudo chown -R $USER:$USER command_generator_data/

# Check the permissions
$ ls -la command_generator_data/
```

The command generator currently expects to use the Docker CLI in the daemon mode. The default Docker installation does not have User namespace mapping for Discretionary access control (DAC). So, for any file created by **root** from the container will also be owned by **root** on the host if the file is located in a shared volume.

You can learn more about the Linux namespaces, including the User namespace at the article The 7 most used Linux namespaces.

13.3. UPGRADE NOTE

Upgrading Ansible Automation Platform to 2.4.20240215 updates the protocol of its internal load balancers from HTTP to HTTPs. If additional networking configurations were added, they can also be updated to ensure connectivity. When the upgrade has succeeded, you must revalidate any additional added networking configurations.

13.4. ANSIBLE AUTOMATION PLATFORM CONTROLLER API

API endpoints for Controller must contain a trailing slash in order for requests to go through. Automatic trailing slash redirects are not supported in this current offering of Ansible Automation Platform from GCP Marketplace. For example a request such as **<controller_base_url>/api/v2/metrics** times out while **<controller_base_url>/api/v2/metrics/** goes through.

13.5. REMOVE EXTENSION NODE NOTE

When removing extension node using the Ansible-on-Clouds ops playbook, ensure you have provided correct instance group name and instance template name. Incorrect instance group name and instance template name results in some orphan resources.

13.6. SECRETS UPDATE

When you update any secrets in the GCP secret manager, ensure that the latest secret version is enabled. For example, if you have two secret versions for a **<deployment-name>-aap-admin** secret, secret version 2 must be enabled, where **<deployment-name>** is the name of your foundation deployment.

CHAPTER 14. 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, OS patching, and Ansible Automation Platform patching.

Red Hat does not support changes to the infrastructure resources deployed as part of the solution unless there is documentation by Red Hat describing the process, 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 Ansible Automation Platform from GCP Marketplace deployment.

Upgrades to Ansible Automation Platform from GCP Marketplace are performed differently from self-installed Ansible Automation Platform. Upgrading individual packages on virtual machines using **dnf** or other means is also not supported. Instructions will be provided in the upgrade section of this documentation with relevant content for each version, as they become available.

14.1. 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.
- Red Hat supports the current version of Ansible Automation Platform. Bug fixes and CVE patches require an upgrade to the latest version

Customer Responsibilities

You have the following responsibilities:

- GCP infrastructure uptime
- GCP infrastructure changes, for example, increasing block storage sizes
- GCP network peering and configuration
- Application of Ansible Automation Platform upgrades
    - Operating system upgrades are included

- Backing up GCP resources

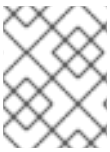
14.2. VM IMAGE PACKAGES

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.

APPENDIX A. RELEASE NOTES FOR ANSIBLE ON CLOUDS 2.4

This release includes a number of enhancements, additions, and fixes that have been implemented for Ansible Automation Platform from GCP Marketplace.



NOTE

Automation Mesh and Event Driven Automation are not yet supported by the self-managed Ansible on Cloud offerings at this time.

Enhancements

The release version 2.4.20240215 of Ansible Automation Platform from GCP Marketplace includes the following enhancements:

- Various CVE fixes

Deprecated and removed features

Some features available in previous releases have been deprecated or removed. Deprecated functionality is still included in Ansible Automation Platform and continues to be supported; however, it will be removed in a future release of this product and is not recommended for new deployments. The following is a list of major functionality deprecated and removed within Ansible Automation Platform 2.4:

- On-premise component Automation Services Catalog is now removed from Ansible Automation Platform 2.4 onwards.
- With the Ansible Automation Platform 2.4 release, the Execution Environment container image for Ansible 2.9 (ee-29-rhel-8) is no longer loaded into the Automation Controller configuration by default.
- 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.

Additional Release Notes related to Ansible Automation Platform

- Check out latest features in Red Hat Enterprise Linux 9
- Read more about latest Ansible Automation Platform features