Red Hat Ansible Automation Platform 2.5

Containerized installation

Install the containerized version of Ansible Automation Platform

Last Updated: 2024-10-18

Red Hat Ansible Automation Platform 2.5 Containerized installation

Install the containerized version of Ansible Automation Platform

Legal Notice

Abstract

This guide helps you to understand the installation requirements and processes behind our containerized version of Ansible Automation Platform.

Table of Contents

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION 3

CHAPTER 1. ANSIBLE AUTOMATION PLATFORM CONTAINERIZED INSTALLATION 4
1.1. TESTED DEPLOYMENT TOPOLOGIES 4
1.2. SYSTEM REQUIREMENTS 4
1.3. PREPARING THE RED HAT ENTERPRISE LINUX HOST FOR CONTAINERIZED INSTALLATION 5
1.4. DOWNLOADING ANSIBLE AUTOMATION PLATFORM 6
1.5. INSTALLING CONTAINERIZED ANSIBLE AUTOMATION PLATFORM 6
1.5.1. Inventory file for online installation for containerized growth topology (all-in-one) 7
1.5.2. Inventory file for online installation for containerized enterprise topology 9
1.5.3. Additional information for configuring your inventory file 11
1.5.4. Running the installation command 11
1.6. ACCESSING ANSIBLE AUTOMATION PLATFORM 11
1.7. USING CUSTOM TLS CERTIFICATES 12
1.8. USING CUSTOM RECEPTOR SIGNING KEYS 13
1.9. ENABLING AUTOMATION HUB COLLECTION AND CONTAINER SIGNING 13
1.10. ADDING EXECUTION NODES 13
1.11. ADDING A SAFE PLUGIN VARIABLE TO EVENT-DRIVEN ANSIBLE CONTROLLER 14
1.12. UNINSTALLING CONTAINERIZED ANSIBLE AUTOMATION PLATFORM 14

CHAPTER 2. HORIZONTAL SCALING IN RED HAT ANSIBLE AUTOMATION PLATFORM 16
2.1. HORIZONTAL SCALING IN EVENT-DRIVEN ANSIBLE CONTROLLER 16
2.1.1. Sizing and scaling guidelines 16
2.1.2. Setting up horizontal scaling for Event-Driven Ansible controller 17

APPENDIX A. TROUBLESHOOTING CONTAINERIZED ANSIBLE AUTOMATION PLATFORM 18
A.1. TROUBLESHOOTING CONTAINERIZED ANSIBLE AUTOMATION PLATFORM INSTALLATION 18
A.2. TROUBLESHOOTING CONTAINERIZED ANSIBLE AUTOMATION PLATFORM CONFIGURATION 21
A.3. CONTAINERIZED ANSIBLE AUTOMATION PLATFORM REFERENCE 22

APPENDIX B. INVENTORY FILE VARIABLES 28
B.1. GENERAL VARIABLES 28
B.2. AUTOMATION HUB VARIABLES 31
B.3. AUTOMATION CONTROLLER VARIABLES 41
B.4. EVENT-DRIVEN ANSIBLE CONTROLLER VARIABLES 47
B.5. PLATFORM GATEWAY VARIABLES 52
B.6. DATABASE VARIABLES 56
B.7. IMAGE VARIABLES 58
B.8. RECEPTOR VARIABLES 59
B.9. ANSIBLE VARIABLES 60

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

If you have a suggestion to improve this documentation, or find an error, you can contact technical support at https://access.redhat.com to open a request.

CHAPTER 1. ANSIBLE AUTOMATION PLATFORM CONTAINERIZED INSTALLATION

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 requirements and processes behind the containerized version of Ansible Automation Platform.

NOTE

Upgrades from 2.4 Containerized Ansible Automation Platform Tech Preview to 2.5 Containerized Ansible Automation Platform are unsupported at this time.

Prerequisites

- A Red Hat Enterprise Linux (RHEL) 9.2 based host. Use a minimal operating system base install.
- A non-root user for the Red Hat Enterprise Linux host, with sudo or other Ansible supported privilege escalation (sudo recommended). This user is responsible for the installation of containerized Ansible Automation Platform.
- SSH public key authentication for the non-root user. For guidelines on setting up SSH public key authentication for the non-root user, see How to configure SSH public key authentication for passwordless login.
    - SSH keys are only required when installing on remote hosts. If doing a self contained local VM based installation, you can use **ansible_connection=local**.
- Internet access from the Red Hat Enterprise Linux host if you are using the default online installation method.
- The appropriate network ports are open if a firewall is in place. For more information about the ports to open, see Container topologies in *Tested deployment models*.

1.1. TESTED DEPLOYMENT TOPOLOGIES

Red Hat tests Ansible Automation Platform 2.5 with a defined set of topologies to give you opinionated deployment options. The supported topologies include infrastructure topology diagrams, tested system configurations, example inventory files, and network ports information.

For containerized Ansible Automation Platform, there are two infrastructure topology shapes:

1. Growth - (All-in-one) Intended for organizations that are getting started with Ansible Automation Platform. This topology allows for smaller footprint deployments.
2. Enterprise - Intended for organizations that require Ansible Automation Platform deployments to have redundancy or higher compute for large volumes of automation. This is a more future-proofed scaled out architecture.

For more information about the tested deployment topologies for containerized Ansible Automation Platform, see Container topologies in *Tested deployment models*.

1.2. SYSTEM REQUIREMENTS

Each virtual machine (VM) has the following system requirements:

Requirement	Minimum requirement
RAM	16 GB
CPUs	4
Local disk	60 GB
Disk IOPS	3000

1.3. PREPARING THE RED HAT ENTERPRISE LINUX HOST FOR CONTAINERIZED INSTALLATION

Containerized Ansible Automation Platform runs the component services as Podman based containers on top of a Red Hat Enterprise Linux host. Prepare the Red Hat Enterprise Linux host to ensure a successful installation.

Procedure

1. Log in to the Red Hat Enterprise Linux host as your non-root user.
2. Set a hostname that is a fully qualified domain name (FQDN):

```
sudo hostnamectl set-hostname <your_hostname>
```

3. Register your Red Hat Enterprise Linux host with **subscription-manager**:

```
sudo subscription-manager register
```

4. Run **sudo dnf repolist** to validate that only the BaseOS and AppStream repositories are setup and enabled on the host:

```
$ sudo dnf repolist
Updating Subscription Management repositories.
repo id                                  repo name
rhel-9-for-x86_64-appstream-rpms         Red Hat Enterprise Linux 9 for x86_64 -
AppStream (RPMs)
rhel-9-for-x86_64-baseos-rpms            Red Hat Enterprise Linux 9 for x86_64 -
BaseOS (RPMs)
```

5. Ensure that only these repositories are available to the Red Hat Enterprise Linux host. For more information about managing custom repositories, see: Managing custom software repositories.
6. Ensure that the host has DNS configured and can resolve host names and IP addresses by using a fully qualified domain name (FQDN). This is essential to ensure services can talk to one another.
7. Install **ansible-core**:

```
sudo dnf install -y ansible-core
```

8. Optional: You can install additional utilities that can be useful for troubleshooting purposes, for example **wget**, **git-core**, **rsync**, and **vim**:

```
sudo dnf install -y wget git-core rsync vim
```

9. Optional: To have the installer automatically pick up and apply your Ansible Automation Platform subscription manifest license, follow the steps in Obtaining a manifest file .

Additional resources

- For more information about registering your RHEL system, see Getting Started with RHEL System Registration.
- For information about configuring unbound DNS, see Setting up an unbound DNS server.
- For information about configuring DNS using BIND, see Setting up and configuring a BIND DNS server.

1.4. DOWNLOADING ANSIBLE AUTOMATION PLATFORM

Choose the installer you need based on your Red Hat Enterprise Linux environment internet connectivity and download the installer to your Red Hat Enterprise Linux host.

Procedure

1. Download the latest installer .tar file from the Ansible Automation Platform download page .
   a. For online installations: **Ansible Automation Platform 2.5 Containerized Setup**
   b. For offline or bundled installations: **Ansible Automation Platform 2.5 Containerized Setup Bundle**
2. Copy the installer .tar file and the optional manifest .zip file onto your Red Hat Enterprise Linux host.
3. Decide where you want the installer to reside on the file system. Installation related files are created under this location and require at least 10 GB for the initial installation.
4. Unpack the installer .tar file into your installation directory, and go to the unpacked directory.
   a. To unpack the online installer:

```
$ tar xfvz ansible-automation-platform-containerized-setup-<version>.tar.gz
```

   b. To unpack the offline or bundled installer:

```
$ tar xfvz ansible-automation-platform-containerized-setup-bundle-<version>-
<arch_name>.tar.gz
```

1.5. INSTALLING CONTAINERIZED ANSIBLE AUTOMATION PLATFORM

You can control the installation of Ansible Automation Platform with inventory files. Inventory files define the hosts and containers used and created, variables for components, and other information needed to customize the installation.

Example inventory files are provided in this document that you can copy and change to quickly get started.

Inventory files for the growth and enterprise topology are also found in the downloaded installer package:

- The default one named **inventory** is for the enterprise topology pattern.
- If you want to deploy the growth or all-in-one pattern you need to copy over or use the **inventory-growth** file instead.

Additionally, you can find example inventory files in Container topologies in *Tested deployment models*.

To use the example inventory files, replace the **< >** placeholders with your specific variables, and update the host names. Refer to the **README.md** file in the installation directory for more information about optional and required variables.

1.5.1. Inventory file for online installation for containerized growth topology (all-in-one)

Use the following example inventory file to perform an online installation for the containerized growth topology (all-in-one):

```
# This is the Ansible Automation Platform growth installer inventory file
# Please consult the docs if you are unsure what to add
# For all optional variables please consult the included README.md
# or the Red Hat documentation:
#
https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.5/html/containerized_installation

# This section is for your platform gateway hosts
# -----------------------------------------------------
[automationgateway]
aap.example.org

# This section is for your automation controller hosts
# -----------------------------------------------------
[automationcontroller]
aap.example.org

# This section is for your automation hub hosts
# -----------------------------------------------------
[automationhub]
aap.example.org

# This section is for your Event-Driven Ansible controller hosts
# -----------------------------------------------------
[automationeda]
aap.example.org

# This section is for the Ansible Automation Platform database
```

```
# ----------------------------------------------------
[database]
aap.example.org

[all:vars]

# Common variables
#
https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.5/html/containerized_installation/appendix-inventory-files-vars#ref-general-inventory-variables
# ----------------------------------------------------
postgresql_admin_username=postgres
postgresql_admin_password=<set your own>

registry_username=<your RHN username>
registry_password=<your RHN password>

redis_mode=standalone

# Platform gateway
#
https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.5/html/containerized_installation/appendix-inventory-files-vars#ref-gateway-variables
# ----------------------------------------------------
gateway_admin_password=<set your own>
gateway_pg_host=aap.example.org
gateway_pg_password=<set your own>

# Automation controller
#
https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.5/html/containerized_installation/appendix-inventory-files-vars#ref-controller-variables
# ----------------------------------------------------
controller_admin_password=<set your own>
controller_pg_host=aap.example.org
controller_pg_password=<set your own>

# Automation hub
#
https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.5/html/containerized_installation/appendix-inventory-files-vars#ref-hub-variables
# ----------------------------------------------------
hub_admin_password=<set your own>
hub_pg_host=aap.example.org
hub_pg_password=<set your own>

# Event-Driven Ansible controller
#
https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.5/html/containerized_installation/appendix-inventory-files-vars#event-driven-ansible-controller
# ----------------------------------------------------
eda_admin_password=<set your own>
eda_pg_host=aap.example.org
eda_pg_password=<set your own>
```

1.5.2. Inventory file for online installation for containerized enterprise topology

Use the following example inventory file to perform an online installation for the containerized enterprise topology:

```
# This is the Ansible Automation Platform enterprise installer inventory file
# Please consult the docs if you are unsure what to add
# For all optional variables please consult the included README.md
# or the Red Hat documentation:
#
https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.5/html/containerized_installation

# This section is for your platform gateway hosts
# -----------------------------------------------------
[automationgateway]
gateway1.example.org
gateway2.example.org

# This section is for your automation controller hosts
# -----------------------------------------------------
[automationcontroller]
controller1.example.org
controller2.example.org

# This section is for your Ansible Automation Platform execution hosts
# -----------------------------------------------------
[execution_nodes]
hop1.example.org receptor_type='hop'
exec1.example.org
exec2.example.org

# This section is for your automation hub hosts
# -----------------------------------------------------
[automationhub]
hub1.example.org
hub2.example.org

# This section is for your Event-Driven Ansible controller hosts
# -----------------------------------------------------
[automationeda]
eda1.example.org
eda2.example.org

[redis]
gateway1.example.org
gateway2.example.org
hub1.example.org
hub2.example.org
eda1.example.org
eda2.example.org

[all:vars]

# Common variables
#
```

```
https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.5/html/containerized_installation/appendix-inventory-files-vars#ref-general-inventory-variables
# ------------------------------------------------------
registry_username=<your RHN username>
registry_password=<your RHN password>

# Platform gateway
#
https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.5/html/containerized_installation/appendix-inventory-files-vars#ref-gateway-variables
# ------------------------------------------------------
gateway_admin_password=<set your own>
gateway_pg_host=externaldb.example.org
gateway_pg_database=<set your own>
gateway_pg_username=<set your own>
gateway_pg_password=<set your own>

# Automation controller
#
https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.5/html/containerized_installation/appendix-inventory-files-vars#ref-controller-variables
# ------------------------------------------------------
controller_admin_password=<set your own>
controller_pg_host=externaldb.example.org
controller_pg_database=<set your own>
controller_pg_username=<set your own>
controller_pg_password=<set your own>

# Automation hub
#
https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.5/html/containerized_installation/appendix-inventory-files-vars#ref-hub-variables
# ------------------------------------------------------
hub_admin_password=<set your own>
hub_pg_host=externaldb.example.org
hub_pg_database=<set your own>
hub_pg_username=<set your own>
hub_pg_password=<set your own>

# Event-Driven Ansible controller
#
https://docs.redhat.com/en/documentation/red_hat_ansible_automation_platform/2.5/html/containerized_installation/appendix-inventory-files-vars#event-driven-ansible-controller
# ------------------------------------------------------
eda_admin_password=<set your own>
eda_pg_host=externaldb.example.org
eda_pg_database=<set your own>
eda_pg_username=<set your own>
eda_pg_password=<set your own>
```

Redis configuration for an enterprise topology

- Redis can be colocated with any other node in a clustered installation.
- By default the **redis_mode** is set to **cluster**.

- **redis_mode=cluster**

- For more information about Redis, see Caching and queueing system in *Planning your installation*.

1.5.3. Additional information for configuring your inventory file

Offline or bundled installation

- To perform an offline installation, add the following under the **[all:vars]** group:

```
bundle_install=true
# The bundle directory must include /bundle in the path
bundle_dir=<full path to the bundle directory>
```

Configuring a HAProxy load balancer

- To configure a HAProxy load balancer in front of platform gateway with a custom CA cert, set the following inventory file variables under the **[all:vars]** group:

```
custom_ca_cert=<path_to_cert_crt>
gateway_main_url=<https://load_balancer_url>
```

NOTE

HAProxy SSL passthrough mode is not supported with platform gateway.

Loading an automation controller license file

- To define the location of your automation controller license file, set the following variable in the inventory file:

```
controller_license_file=<full_path_to_your_manifest_zip_file>
```

1.5.4. Running the installation command

Use the following command to install containerized Ansible Automation Platform:

```
ansible-playbook -i inventory ansible.containerized_installer.install
```

- If your privilege escalation requires you to enter a password, append **-K** to the command line. You are then prompted for the **BECOME** password.

- You can use increasing verbosity, up to 4 v's (**-vvvv**) to see the details of the installation process. However, it is important to note that this can significantly increase installation time, so it is recommended that you use it only as needed or requested by Red Hat support.

1.6. ACCESSING ANSIBLE AUTOMATION PLATFORM

After the installation completes, the default protocol and ports used for Ansible Automation Platform are 80 (HTTP) and 443 (HTTPS).

You can customize the ports with the following variables:

```
envoy_http_port=80
envoy_https_port=443
```

If you want to disable HTTPS, set **envoy_disable_https** to **true**:

```
envoy_disable_https=true
```

Accessing the platform UI

The platform UI is available by default at:

```
https://<gateway-node>:443
```

Log in as the admin user with the password you created for **gateway_admin_password**.

1.7. USING CUSTOM TLS CERTIFICATES

By default, the installer generates TLS certificates and keys for all services that are signed by a custom Certificate Authority (CA). You can provide a custom TLS certificate and key for each service. If that certificate is signed by a custom CA, you must provide the CA TLS certificate and key.

- Certificate Authority

```
ca_tls_cert=/full/path/to/tls/certificate
ca_tls_key=/full/path/to/tls/key
```

- Platform gateway

```
gateway_tls_cert=/full/path/to/tls/certificate
gateway_tls_key=/full/path/to/tls/key
```

- Automation controller

```
controller_tls_cert=/full/path/to/tls/certificate
controller_tls_key=/full/path/to/tls/key
```

- Automation hub

```
hub_tls_cert=/full/path/to/tls/certificate
hub_tls_key=/full/path/to/tls/key
```

- Event-Driven Ansible

```
eda_tls_cert=/full/path/to/tls/certificate
eda_tls_key=/full/path/to/tls/key
```

- PostgreSQL

```
postgresql_tls_cert=/full/path/to/tls/certificate
postgresql_tls_key=/full/path/to/tls/key
```

- Receptor

```
receptor_tls_cert=/full/path/to/tls/certificate
receptor_tls_key=/full/path/to/tls/key
```

1.8. USING CUSTOM RECEPTOR SIGNING KEYS

Receptor signing is enabled by default unless **receptor_disable_signing=true** is set, and an RSA key pair (public and private) is generated by the installer. However, you can give custom RSA public and private keys by setting the path variables:

```
receptor_signing_private_key=<full_path_to_private_key>
receptor_signing_public_key=<full_path_to_public_key>
```

1.9. ENABLING AUTOMATION HUB COLLECTION AND CONTAINER SIGNING

With automation hub you can sign Ansible collections and container images. This feature is not enabled by default, and you must provide the GPG key.

```
hub_collection_signing=true
hub_collection_signing_key=<full_path_to_collections_gpg_key>
hub_container_signing=true
hub_container_signing_key=<full_path_to_containers_gpg_key>
```

When the GPG key is protected by a passphrase, you must provide the passphrase.

```
hub_collection_signing_pass=<collections_gpg_key_passphrase>
hub_container_signing_pass=<containers_gpg_key_passphrase>
```

1.10. ADDING EXECUTION NODES

The containerized installer can deploy remote execution nodes. The **execution_nodes** group in the inventory file handles this.

```
[execution_nodes]
<fqdn_of_your_execution_host>
```

An execution node is by default configured as an execution type running on port 27199 (TCP). This can be changed by the following variables:

```
receptor_port=27199
receptor_protocol=tcp
receptor_type=hop
```

The **receptor_type** value can be either **execution** or **hop**, while the **receptor_protocol** is either **tcp** or **udp**. By default, the nodes in the **execution_nodes** group are added as peers for the controller node. However, you can change the peers configuration by using the **receptor_peers** variable.

```
[execution_nodes]
fqdn_of_your_execution_host
fqdn_of_your_hop_host receptor_type=hop receptor_peers='["<fqdn_of_your_execution_host>"]'
```

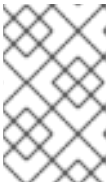
1.11. ADDING A SAFE PLUGIN VARIABLE TO EVENT-DRIVEN ANSIBLE CONTROLLER

When using redhat.insights_eda or similar plugins to run rulebook activations in Event-Driven Ansible controller, you must add a safe plugin variable to a directory in Ansible Automation Platform. This ensures connection between Event-Driven Ansible controller and the source plugin, and displays port mappings correctly.

Procedure

1. Create a directory for the safe plugin variable: **mkdir -p ./group_vars/automationedacontroller**
2. Create a file within that directory for your new setting (for example, **touch ./group_vars/automationedacontroller/custom.yml**)
3. Add the variable **automationedacontroller_additional_settings** to extend the default **settings.yaml** template for Event-Driven Ansible controller and add the **SAFE_PLUGINS** field with a list of plugins to enable. For example:

```
automationedacontroller_additional_settings:
  SAFE_PLUGINS:
    - ansible.eda.webhook
    - ansible.eda.alertmanager
```



NOTE

You can also extend the **automationedacontroller_additional_settings** variable beyond SAFE_PLUGINS in the Django configuration file, /etc/ansible-automation-platform/eda/settings.yaml

1.12. UNINSTALLING CONTAINERIZED ANSIBLE AUTOMATION PLATFORM

To uninstall a containerized deployment, run the **uninstall.yml** playbook:

```
$ ansible-playbook -i inventory ansible.containerized_installer.uninstall
```

This stops all systemd units and containers and then deletes all resources used by the containerized installer such as:

- config and data directories and files
- systemd unit files

- Podman containers and images
- RPM packages

To keep container images, you can set the **container_keep_images** variable to **true**.

```
$ ansible-playbook -i inventory ansible.containerized_installer.uninstall -e
container_keep_images=true
```

To keep PostgreSQL databases, you can set the **postgresql_keep_databases** variable to **true**.

```
$ ansible-playbook -i inventory ansible.containerized_installer.uninstall -e
postgresql_keep_databases=true
```

NOTE

Use the Ansible Automation Platform secret key values rather than the autogenerated ones.

CHAPTER 2. HORIZONTAL SCALING IN RED HAT ANSIBLE AUTOMATION PLATFORM

You can set up multi-node deployments for components across Ansible Automation Platform. Whether you require horizontal scaling for Automation Execution, Automation Decisions, or automation mesh, you can scale your deployments based on your organization's needs.

2.1. HORIZONTAL SCALING IN EVENT-DRIVEN ANSIBLE CONTROLLER

With Event-Driven Ansible controller, you can set up horizontal scaling for your events automation. This multi-node deployment enables you to define as many nodes as you prefer during the installation process. You can also increase or decrease the number of nodes at any time according to your organizational needs.

The following node types are used in this deployment:

API node type

Responds to the HTTP REST API of Event-Driven Ansible controller.

Worker node type

Runs an Event-Driven Ansible worker, which is the component of Event-Driven Ansible that not only manages projects and activations, but also executes the activations themselves.

Hybrid node type

Is a combination of the API node and the worker node.

The following example shows how you would set up an inventory file for horizontal scaling of Event-Driven Ansible controller on Red Hat Enterprise Linux VMs using the host group name **[automationedacontroller]** and the node type variable **eda_node_type**:

```
[automationedacontroller]

3.88.116.111
routable_hostname=automationedacontroller-api.example.com eda_node_type=api

# worker node
3.88.116.112 routable_hostname=automationedacontroller-api.example.com eda_node_type=worker
```

2.1.1. Sizing and scaling guidelines

API nodes process user requests (interactions with the UI or API) while worker nodes process the activations and other background tasks required for Event-Driven Ansible to function properly. The number of API nodes you require correlates to the desired number of users of the application and the number of worker nodes correlates to the desired number of activations you want to run.

Since activations are variable and controlled by worker nodes, the supported approach for scaling is to use separate API and worker nodes instead of hybrid nodes due to the efficient allocation of hardware resources by worker nodes. By separating the nodes, you can scale each type independently based on specific needs, leading to better resource utilization and cost efficiency.

An example of an instance in which you might consider scaling up your node deployment is when you want to deploy Event-Driven Ansible for a small group of users who will run a large number of activations. In this case, one API node is adequate, but if you require more, you can scale up to three additional worker nodes.

To set up a multi-node deployment, follow the procedure in Setting up horizontal scaling for Event-Driven Ansible controller.

2.1.2. Setting up horizontal scaling for Event-Driven Ansible controller

To scale up (add more nodes) or scale down (remove nodes), you must update the content of the inventory to add or remove nodes and rerun the installer.

Procedure

1. Update the inventory to add two more worker nodes:

```
[automationedacontroller]

3.88.116.111 routable_hostname=automationedacontroller-api.example.com
eda_node_type=api

3.88.116.112 routable_hostname=automationedacontroller-api.example.com
eda_node_type=worker

# two more worker nodes
3.88.116.113 routable_hostname=automationedacontroller-api.example.com
eda_node_type=worker

3.88.116.114 routable_hostname=automationedacontroller-api.example.com
eda_node_type=worker
```

2. Re-run the installer.

APPENDIX A. TROUBLESHOOTING CONTAINERIZED ANSIBLE AUTOMATION PLATFORM

Use this information to troubleshoot your containerized Ansible Automation Platform installation.

A.1. TROUBLESHOOTING CONTAINERIZED ANSIBLE AUTOMATION PLATFORM INSTALLATION

The installation takes a long time, or has errors, what should I check?

1. Ensure your system meets the minimum requirements as outlined in the installation guide. Items such as improper storage choices and high latency when distributing across many hosts will all have a significant impact.

2. Check the installation log file located by default at **./aap_install.log** unless otherwise changed within the local installer **ansible.cfg**.

3. Enable task profiling callbacks on an ad hoc basis to give an overview of where the installation program spends the most time. To do this, use the local **ansible.cfg** file. Add a callback line such as this under the **[defaults]** section:

```
$ cat ansible.cfg
[defaults]
callbacks_enabled = ansible.posix.profile_tasks
```

Automation controller returns an error of 413

This error is due to **manifest.zip** license files that are larger than the **nginx_client_max_body_size** setting. If this error occurs, you will need to change the installation inventory file to include the following variables:

```
nginx_disable_hsts=false
nginx_http_port=8081
nginx_https_port=8444
nginx_client_max_body_size=20m
nginx_user_headers=[]
```

The current default setting of **20m** should be enough to avoid this issue.

The installation failed with a "502 Bad Gateway" when going to the controller UI.

This error can occur and manifest itself in the installation application output as:

```
TASK [ansible.containerized_installer.automationcontroller : Wait for the Controller API to te ready]
**************************************************
fatal: [daap1.lan]: FAILED! => {"changed": false, "connection": "close", "content_length": "150",
"content_type": "text/html", "date": "Fri, 29 Sep 2023 09:42:32 GMT", "elapsed": 0, "msg": "Status
code was 502 and not [200]: HTTP Error 502: Bad Gateway", "redirected": false, "server": "nginx",
"status": 502, "url": "https://daap1.lan:443/api/v2/ping/"}
```

- Check if you have an **automation-controller-web** container running and a systemd service.

NOTE

This is used at the regular unprivileged user not system wide level. If you have used **su** to switch to the user running the containers, you must set your **XDG_RUNTIME_DIR** environment variable to the correct value to be able to interact with the user **systemctl** units.

```
export XDG_RUNTIME_DIR="/run/user/$UID"
```

```
podman ps | grep web
systemctl --user | grep web
```

No output indicates a problem.

1. Try restarting the **automation-controller-web** service:

```
systemctl start automation-controller-web.service --user
systemctl --user | grep web
systemctl status automation-controller-web.service --user
```

```
Sep 29 10:55:16 daap1.lan automation-controller-web[29875]: nginx: [emerg] bind() to
0.0.0.0:443 failed (98: Address already in use)
Sep 29 10:55:16 daap1.lan automation-controller-web[29875]: nginx: [emerg] bind() to
0.0.0.0:80 failed (98: Address already in use)
```

   The output indicates that the port is already, or still, in use by another service. In this case **nginx**.

2. Run:

```
sudo pkill nginx
```

3. Restart and status check the web service again.

Normal service output should look similar to the following, and should still be running:

```
Sep 29 10:59:26 daap1.lan automation-controller-web[30274]: WSGI app 0 (mountpoint='/') ready in
3 seconds on interpreter 0x1a458c10 pid: 17 (default app)
Sep 29 10:59:26 daap1.lan automation-controller-web[30274]: WSGI app 0 (mountpoint='/') ready in
3 seconds on interpreter 0x1a458c10 pid: 20 (default app)
Sep 29 10:59:27 daap1.lan automation-controller-web[30274]: 2023-09-29 09:59:27,043 INFO     [-]
daphne.cli Starting server at tcp:port=8051:interface=127.0.>
Sep 29 10:59:27 daap1.lan automation-controller-web[30274]: 2023-09-29 09:59:27,043 INFO
Starting server at tcp:port=8051:interface=127.0.0.1
Sep 29 10:59:27 daap1.lan automation-controller-web[30274]: 2023-09-29 09:59:27,048 INFO     [-]
daphne.server HTTP/2 support not enabled (install the http2 >
Sep 29 10:59:27 daap1.lan automation-controller-web[30274]: 2023-09-29 09:59:27,048 INFO
HTTP/2 support not enabled (install the http2 and tls Twisted ex>
Sep 29 10:59:27 daap1.lan automation-controller-web[30274]: 2023-09-29 09:59:27,049 INFO     [-]
daphne.server Configuring endpoint tcp:port=8051:interface=1>
Sep 29 10:59:27 daap1.lan automation-controller-web[30274]: 2023-09-29 09:59:27,049 INFO
Configuring endpoint tcp:port=8051:interface=127.0.0.1
Sep 29 10:59:27 daap1.lan automation-controller-web[30274]: 2023-09-29 09:59:27,051 INFO     [-]
daphne.server Listening on TCP address 127.0.0.1:8051
Sep 29 10:59:27 daap1.lan automation-controller-web[30274]: 2023-09-29 09:59:27,051 INFO
```

```
Listening on TCP address 127.0.0.1:8051
Sep 29 10:59:54 daap1.lan automation-controller-web[30274]: 2023-09-29 09:59:54,139 INFO
success: nginx entered RUNNING state, process has stayed up for > th>
Sep 29 10:59:54 daap1.lan automation-controller-web[30274]: 2023-09-29 09:59:54,139 INFO
success: nginx entered RUNNING state, process has stayed up for > th>
Sep 29 10:59:54 daap1.lan automation-controller-web[30274]: 2023-09-29 09:59:54,139 INFO
success: uwsgi entered RUNNING state, process has stayed up for > th>
Sep 29 10:59:54 daap1.lan automation-controller-web[30274]: 2023-09-29 09:59:54,139 INFO
success: uwsgi entered RUNNING state, process has stayed up for > th>
Sep 29 10:59:54 daap1.lan automation-controller-web[30274]: 2023-09-29 09:59:54,139 INFO
success: daphne entered RUNNING state, process has stayed up for > t>
Sep 29 10:59:54 daap1.lan automation-controller-web[30274]: 2023-09-29 09:59:54,139 INFO
success: daphne entered RUNNING state, process has stayed up for > t>
Sep 29 10:59:54 daap1.lan automation-controller-web[30274]: 2023-09-29 09:59:54,139 INFO
success: ws-heartbeat entered RUNNING state, process has stayed up f>
Sep 29 10:59:54 daap1.lan automation-controller-web[30274]: 2023-09-29 09:59:54,139 INFO
success: ws-heartbeat entered RUNNING state, process has stayed up f>
Sep 29 10:59:54 daap1.lan automation-controller-web[30274]: 2023-09-29 09:59:54,139 INFO
success: cache-clear entered RUNNING state, process has stayed up fo>
Sep 29 10:59:54 daap1.lan automation-controller-web[30274]: 2023-09-29 09:59:54,139 INFO
success: cache-clear entered RUNNING state, process has stayed up
```

You can run the installation program again to ensure everything installs as expected.

When attempting to install containerized Ansible Automation Platform in Amazon Web Services you receive output that there is no space left on device

```
TASK [ansible.containerized_installer.automationcontroller : Create the receptor container]
**************************************************
fatal: [ec2-13-48-25-168.eu-north-1.compute.amazonaws.com]: FAILED! => {"changed": false, "msg":
"Can't create container receptor", "stderr": "Error: creating container storage: creating an ID-mapped
copy of layer \"98955f43cc908bd50ff43585fec2c7dd9445eaf05eecd1e3144f93ffc00ed4ba\": error
during chown: storage-chown-by-maps: lchown usr/local/lib/python3.9/site-
packages/azure/mgmt/network/v2019_11_01/operations/__pycache__/_available_service_aliases_oper
ations.cpython-39.pyc: no space left on device: exit status 1\n", "stderr_lines": ["Error: creating
container storage: creating an ID-mapped copy of layer
\"98955f43cc908bd50ff43585fec2c7dd9445eaf05eecd1e3144f93ffc00ed4ba\": error during chown:
storage-chown-by-maps: lchown usr/local/lib/python3.9/site-
packages/azure/mgmt/network/v2019_11_01/operations/__pycache__/_available_service_aliases_oper
ations.cpython-39.pyc: no space left on device: exit status 1"], "stdout": "", "stdout_lines": []}
```

If you are installing a **/home** filesystem into a default Amazon Web Services marketplace RHEL instance, it might be too small since **/home** is part of the root / filesystem. You will need to make more space available. The documentation specifies a minimum of 40GB for a single-node deployment of containerized Ansible Automation Platform.

"Install container tools" task fails due to unavailable packages

This error occurs in the installation application output as:

```
TASK [ansible.containerized_installer.common : Install container tools]
**********************************************************************************************************
fatal: [192.0.2.1]: FAILED! => {"changed": false, "failures": ["No package crun available.", "No
package podman available.", "No package slirp4netns available.", "No package fuse-overlayfs
available."], "msg": "Failed to install some of the specified packages", "rc": 1, "results": []}
```

```
fatal: [192.0.2.2]: FAILED! => {"changed": false, "failures": ["No package crun available.", "No
package podman available.", "No package slirp4netns available.", "No package fuse-overlayfs
available."], "msg": "Failed to install some of the specified packages", "rc": 1, "results": []}
fatal: [192.0.2.3]: FAILED! => {"changed": false, "failures": ["No package crun available.", "No
package podman available.", "No package slirp4netns available.", "No package fuse-overlayfs
available."], "msg": "Failed to install some of the specified packages", "rc": 1, "results": []}
fatal: [192.0.2.4]: FAILED! => {"changed": false, "failures": ["No package crun available.", "No
package podman available.", "No package slirp4netns available.", "No package fuse-overlayfs
available."], "msg": "Failed to install some of the specified packages", "rc": 1, "results": []}
fatal: [192.0.2.5]: FAILED! => {"changed": false, "failures": ["No package crun available.", "No
package podman available.", "No package slirp4netns available.", "No package fuse-overlayfs
available."], "msg": "Failed to install some of the specified packages", "rc": 1, "results": []}
```

To fix this error, run the following command on the target hosts:

```
sudo subscription-manager register
```

A.2. TROUBLESHOOTING CONTAINERIZED ANSIBLE AUTOMATION PLATFORM CONFIGURATION

Sometimes the post install for seeding my Ansible Automation Platform content errors out. This could manifest itself as output similar to this:

```
TASK [infra.controller_configuration.projects : Configure Controller Projects | Wait for finish the
projects creation] ************************************
Friday 29 September 2023  11:02:32 +0100 (0:00:00.443)       0:00:53.521 ******
FAILED - RETRYING: [daap1.lan]: Configure Controller Projects | Wait for finish the projects creation
(1 retries left).
failed: [daap1.lan] (item={'failed': 0, 'started': 1, 'finished': 0, 'ansible_job_id': '536962174348.33944',
'results_file': '/home/aap/.ansible_async/536962174348.33944', 'changed': False,
'__controller_project_item': {'name': 'AAP Config-As-Code Examples', 'organization': 'Default',
'scm_branch': 'main', 'scm_clean': 'no', 'scm_delete_on_update': 'no', 'scm_type': 'git',
'scm_update_on_launch': 'no', 'scm_url': 'https://github.com/user/repo.git'}, 'ansible_loop_var':
'__controller_project_item'}) => {"__projects_job_async_results_item": {"__controller_project_item":
{"name": "AAP Config-As-Code Examples", "organization": "Default", "scm_branch": "main",
"scm_clean": "no", "scm_delete_on_update": "no", "scm_type": "git", "scm_update_on_launch": "no",
"scm_url": "https://github.com/user/repo.git"}, "ansible_job_id": "536962174348.33944",
"ansible_loop_var": "__controller_project_item", "changed": false, "failed": 0, "finished": 0,
"results_file": "/home/aap/.ansible_async/536962174348.33944", "started": 1}, "ansible_job_id":
"536962174348.33944", "ansible_loop_var": "__projects_job_async_results_item", "attempts": 30,
"changed": false, "finished": 0, "results_file": "/home/aap/.ansible_async/536962174348.33944",
"started": 1, "stderr": "", "stderr_lines": [], "stdout": "", "stdout_lines": []}
```

The **infra.controller_configuration.dispatch** role uses an asynchronous loop with 30 retries to apply each configuration type, and the default delay between retries is 1 second. If the configuration is large, this might not be enough time to apply everything before the last retry occurs.

Increase the retry delay by setting the **controller_configuration_async_delay** variable to something other than 1 second. For example, setting it to 2 seconds doubles the retry time. The place to do this would be in the repository where the controller configuration is defined. It could also be added to the **[all:vars]** section of the installation program inventory file.

A few instances have shown that no additional modification is required, and re-running the installation program again worked.

A.3. CONTAINERIZED ANSIBLE AUTOMATION PLATFORM REFERENCE

Can you provide details of the architecture for the Ansible Automation Platform containerized design?

We use as much of the underlying native RHEL technology as possible. For the container runtime and management of services we use Podman. Many Podman services and commands are used to show and investigate the solution.

For instance, use **podman ps**, and **podman images** to see some of the foundational and running pieces:

```
[aap@daap1 aap]$ podman ps
CONTAINER ID  IMAGE                                                                COMMAND               CREATED
STATUS         PORTS       NAMES
88ed40495117  registry.redhat.io/rhel8/postgresql-13:latest                        run-postgresql        48
minutes ago  Up 47 minutes              postgresql
8f55ba612f04  registry.redhat.io/rhel8/redis-6:latest                              run-redis             47
minutes ago  Up 47 minutes              redis
56c40445c590  registry.redhat.io/ansible-automation-platform-24/ee-supported-rhel8:latest
/usr/bin/receptor...  47 minutes ago  Up 47 minutes              receptor
f346f05d56ee  registry.redhat.io/ansible-automation-platform-24/controller-rhel8:latest
/usr/bin/launch_a...  47 minutes ago  Up 45 minutes              automation-controller-rsyslog
26e3221963e3  registry.redhat.io/ansible-automation-platform-24/controller-rhel8:latest
/usr/bin/launch_a...  46 minutes ago  Up 45 minutes              automation-controller-task
c7ac92a1e8a1  registry.redhat.io/ansible-automation-platform-24/controller-rhel8:latest
/usr/bin/launch_a...  46 minutes ago  Up 28 minutes              automation-controller-web

[aap@daap1 aap]$ podman images
REPOSITORY                                                TAG         IMAGE ID      CREATED      SIZE
registry.redhat.io/ansible-automation-platform-24/ee-supported-rhel8  latest      b497bdbee59e  10
days ago  3.16 GB
registry.redhat.io/ansible-automation-platform-24/controller-rhel8    latest      ed8ebb1c1baa  10 days
ago  1.48 GB
registry.redhat.io/rhel8/redis-6                          latest      78905519bb05  2 weeks ago  357 MB
registry.redhat.io/rhel8/postgresql-13                    latest      9b65bc3d0413  2 weeks ago  765
MB
[aap@daap1 aap]$
```

Containerized Ansible Automation Platform runs as rootless containers for maximum out-of-the-box security. This means you can install containerized Ansible Automation Platform by using any local unprivileged user account. Privilege escalation is only needed for certain root level tasks, and by default is not needed to use root directly.

Once installed, you will notice certain items have populate on the filesystem where the installation program is run (the underlying RHEL host).

```
[aap@daap1 aap]$ tree -L 1
.
├── aap_install.log
```

```
├── ansible.cfg
├── collections
├── galaxy.yml
├── inventory
├── LICENSE
├── meta
├── playbooks
├── plugins
├── README.md
├── requirements.yml
├── roles
```

Other containerized services that make use of things such as Podman volumes, reside under the installation root directory used. Here are some examples for further reference:

The containers directory contains some of the Podman specifics used and installed for the execution plane:

```
containers/
├── podman
├── storage
│   ├── defaultNetworkBackend
│   ├── libpod
│   ├── networks
│   ├── overlay
│   ├── overlay-containers
│   ├── overlay-images
│   ├── overlay-layers
│   ├── storage.lock
│   └── userns.lock
└── storage.conf
```

The controller directory has some of the installed configuration and runtime data points:

```
controller/
├── data
│   ├── job_execution
│   ├── projects
│   └── rsyslog
├── etc
│   ├── conf.d
│   ├── launch_awx_task.sh
│   ├── settings.py
│   ├── tower.cert
│   └── tower.key
├── nginx
│   └── etc
├── rsyslog
│   └── run
└── supervisor
    └── run
```

The receptor directory has the automation mesh configuration:

```
receptor/
```

```
├── etc
│   └── receptor.conf
└── run
    ├── receptor.sock
    └── receptor.sock.lock
```

After installation, you will also find other pieces in the local users home directory such as the **.cache** directory:

```
.cache/
├── containers
│   └── short-name-aliases.conf.lock
└── rhsm
    └── rhsm.log
```

As we run by default in the most secure manner, such as rootless Podman, we can also use other services such as running **systemd** as non-privileged users. Under **systemd** you can see some of the component service controls available:

The **.config** directory:

```
.config/
├── cni
│   └── net.d
│       └── cni.lock
├── containers
│   ├── auth.json
│   └── containers.conf
└── systemd
    └── user
        ├── automation-controller-rsyslog.service
        ├── automation-controller-task.service
        ├── automation-controller-web.service
        ├── default.target.wants
        ├── podman.service.d
        ├── postgresql.service
        ├── receptor.service
        ├── redis.service
        └── sockets.target.wants
```

This is specific to Podman and conforms to the Open Container Initiative (OCI) specifications. Whereas Podman run as the root user would use **/var/lib/containers** by default, for standard users the hierarchy under **$HOME/.local** is used.

The **.local** directory:

```
.local/
└── share
    └── containers
        ├── cache
        ├── podman
        └── storage
```

As an example `.local/storage/volumes` contains what the output from `podman volume ls` provides:

```
[aap@daap1 containers]$ podman volume ls
DRIVER      VOLUME NAME
local       d73d3fe63a957bee04b4853fd38c39bf37c321d14fdab9ee3c9df03645135788
local       postgresql
local       redis_data
local       redis_etc
local       redis_run
```

We isolate the execution plane from the control plane main services (PostgreSQL, Redis, automation controller, receptor, automation hub and Event-Driven Ansible).

Control plane services run with the standard Podman configuration (**~/.local/share/containers/storage**).

Execution plane services use a dedicated configuration or storage (**~/aap/containers/storage**) to avoid execution plane containers to be able to interact with the control plane.

How can I see host resource utilization statistics?

- Run:

```
$ podman container stats -a
```

```
podman container stats -a
ID            NAME                        CPU %       MEM USAGE / LIMIT  MEM %       NET IO      BLOCK
IO    PIDS        CPU TIME    AVG CPU %
0d5d8eb93c18  automation-controller-web     0.23%       959.1MB / 3.761GB  25.50%      0B / 0B
0B / 0B     16          20.885142s  1.19%
3429d559836d  automation-controller-rsyslog  0.07%       144.5MB / 3.761GB  3.84%       0B / 0B
0B / 0B     6           4.099565s   0.23%
448d0bae0942  automation-controller-task    1.51%       633.1MB / 3.761GB  16.83%      0B / 0B     0B
/ 0B     33          34.285272s  1.93%
7f140e65b57e  receptor                     0.01%       5.923MB / 3.761GB  0.16%       0B / 0B     0B / 0B
7           1.010613s   0.06%
c1458367ca9c  redis                        0.48%       10.52MB / 3.761GB  0.28%       0B / 0B     0B / 0B     5
9.074042s   0.47%
ef712cc2dc89  postgresql                   0.09%       21.88MB / 3.761GB  0.58%       0B / 0B     0B / 0B
21          15.571059s  0.80%
```

The previous is an example of a Dell sold and offered containerized Ansible Automation Platform solution (DAAP) install and utilizes ~1.8Gb RAM.

How much storage is used and where?

As we run rootless Podman the container volume storage is under the local user at **$HOME/.local/share/containers/storage/volumes**.

1. To view the details of each volume run:

   ```
   $ podman volume ls
   ```

2. Then run:

   ```
   $ podman volume inspect <volume_name>
   ```

Here is an example:

```
$ podman volume inspect postgresql
[
    {
        "Name": "postgresql",
        "Driver": "local",
        "Mountpoint": "/home/aap/.local/share/containers/storage/volumes/postgresql/_data",
        "CreatedAt": "2024-01-08T23:39:24.983964686Z",
        "Labels": {},
        "Scope": "local",
        "Options": {},
        "MountCount": 0,
        "NeedsCopyUp": true
    }
]
```

Several files created by the installation program are located in **$HOME/aap/** and bind-mounted into various running containers.

1. To view the mounts associated with a container run:

```
$ podman ps --format "{{.ID}}\t{{.Command}}\t{{.Names}}"
```

```
Example:
$ podman ps --format "{{.ID}}\t{{.Command}}\t{{.Names}}"
89e779b81b83 run-postgresql postgresql
4c33cc77ef7d run-redis redis
3d8a028d892d /usr/bin/receptor... receptor
09821701645c /usr/bin/launch_a... automation-controller-rsyslog
a2ddb5cac71b /usr/bin/launch_a... automation-controller-task
fa0029a3b003 /usr/bin/launch_a... automation-controller-web
20f192534691 gunicorn --bind 1... automation-eda-api
f49804c7e6cb daphne -b 127.0.0... automation-eda-daphne
d340b9c1cb74 /bin/sh -c nginx ... automation-eda-web
111f47de5205 aap-eda-manage rq... automation-eda-worker-1
171fcb1785af aap-eda-manage rq... automation-eda-worker-2
049d10555b51 aap-eda-manage rq... automation-eda-activation-worker-1
7a78a41a8425 aap-eda-manage rq... automation-eda-activation-worker-2
da9afa8ef5e2 aap-eda-manage sc... automation-eda-scheduler
8a2958be9baf gunicorn --name p... automation-hub-api
0a8b57581749 gunicorn --name p... automation-hub-content
68005b987498 nginx -g daemon o... automation-hub-web
cb07af77f89f pulpcore-worker automation-hub-worker-1
a3ba05136446 pulpcore-worker automation-hub-worker-2
```

2. Then run:

```
$ podman inspect <container_name> | jq -r .[].Mounts[].Source
```

```
Example:
/home/aap/.local/share/containers/storage/volumes/receptor_run/_data
/home/aap/.local/share/containers/storage/volumes/redis_run/_data
/home/aap/aap/controller/data/rsyslog
```

```
/home/aap/aap/controller/etc/tower.key
/home/aap/aap/controller/etc/conf.d/callback_receiver_workers.py
/home/aap/aap/controller/data/job_execution
/home/aap/aap/controller/nginx/etc/controller.conf
/home/aap/aap/controller/etc/conf.d/subscription_usage_model.py
/home/aap/aap/controller/etc/conf.d/cluster_host_id.py
/home/aap/aap/controller/etc/conf.d/insights.py
/home/aap/aap/controller/rsyslog/run
/home/aap/aap/controller/data/projects
/home/aap/aap/controller/etc/settings.py
/home/aap/aap/receptor/etc/receptor.conf
/home/aap/aap/controller/etc/conf.d/execution_environments.py
/home/aap/aap/tls/extracted
/home/aap/aap/controller/supervisor/run
/home/aap/aap/controller/etc/uwsgi.ini
/home/aap/aap/controller/etc/conf.d/container_groups.py
/home/aap/aap/controller/etc/launch_awx_task.sh
/home/aap/aap/controller/etc/tower.cert
```

3. If the **jq** RPM is not installed, install with:

```
$ sudo dnf -y install jq
```

APPENDIX B. INVENTORY FILE VARIABLES

The following tables contain information about the variables used in Ansible Automation Platform's installation **inventory** files. The tables include the variables that you can use for RPM-based installation and container-based installation.

B.1. GENERAL VARIABLES

RPM variable name	Container variable name	Description
	bundle_dir	The path to the bundle directory. Default = **false**
	bundle_install	Use offline installation. Set to **true** to enable offline installation. Default = **false**
	ca_tls_cert	Define a Certification Authority certificate here along with a matching key in **ca_tls_key** when you want the installer to create leaf certificates for each product for you.
	ca_tls_key	Define the key for a Certication Authority certificate here for the matching certificate in **ca_tls_cert** when you want the installer to create leaf certs for each product for you.
	ca_tls_remote	TLS CA remote files. Default = **false**
	container_compress	Container compression software. Default = **gzip**
	container_keep_images	Keep container images. Default = **false**
	container_pull_images	Pull newer container images. Default = **true**

RPM variable name	Container variable name	Description
	custom_ca_cert	Define a custom Certification Authority certificate here when you have the leaf certificates created for each product and need the certificate trust to be established.
enable_insights_collection		The default install registers the node to the Red Hat Insights for Red Hat Ansible Automation Platform for the Red Hat Ansible Automation Platform Service if the node is registered with Subscription Manager. Set to **False** to disable. Default = **true**
nginx_tls_protocols		Defines support for **ssl_protocols** in NGINX. Values available **TLSv1**, **TLSv1.1**, **TLSv1.2**, **TLSv1.3**. The TLSv1.1 and TLSv1.2 parameters only work when OpenSSL 1.0.1 or higher is used. The TLSv1.3 parameter only works when OpenSSL 1.1.1 or higher is used. If **nginx_tls-protocols = ['TLSv1.3']** only TLSv1.3 is enabled. To set more than one protocol use **nginx_tls_protocols = ['TLSv1.2', 'TLSv.1.3']**. Default = **TLSv1.2**
nginx_user_http_config		List of NGINX configurations for **/etc/nginx/nginx.conf** under the http section. Each element in the list is provided into **http nginx config** as a separate line. Default = empty list

RPM variable name	Container variable name	Description
	registry_auth	Use registry authentication. Default = **true**
	registry_ns_aap	Ansible Automation Platform registry namespace. Default = **ansible-automation-platform-25**
	registry_ns_rhel	RHEL registry namespace. Default = **rhel8**
registry_password	**registry_password**	This variable is only required if a non-bundle installer is used. Password credential for access to **registry_url**. Enter your Red Hat Registry Service Account credentials in **registry_username** and **registry_password** to link to the Red Hat container registry. When **registry_url** is **registry.redhat.io**, username and password are required if not using a bundle installer. For more information, see Setting registry_username and registry_password.
	registry_tls_verify	Verify registry TLS. Default = **true**
registry_url	**registry_url**	URL for the registry source. Default = **registry.redhat.io**

RPM variable name	Container variable name	Description
registry_username	**registry_username**	This variable is only required if a non-bundle installer is used. User credential for access to **registry_url**. Enter your Red Hat Registry Service Account credentials in **registry_username** and **registry_password** to link to the Red Hat container registry. For more information, see Setting registry_username and registry_password.
routable_hostname	**routable_hostname**	This variable is used if the machine running the installer can only route to the target host through a specific URL. For example, if you use short names in your inventory, but the node running the installer can only resolve that host by using a FQDN. If **routable_hostname** is not set, it should default to **ansible_host**. If you do not set **ansible_host**, **inventory_hostname** is used as a last resort. This variable is used as a host variable for particular hosts and not under the **[all:vars]** section. For further information, see Assigning a variable to one machine: host variables.

B.2. AUTOMATION HUB VARIABLES

RPM variable name	Container variable name	Description

RPM variable name	Container variable name	Description
automationhub_admin_password	**hub_admin_password**	*Required* Required passwords must be enclosed in quotes when they are provided in plain text in the inventory file. Use of special characters for this variable is limited. The password can include any printable ASCII character except /, ", or @.
automationhub_api_token		This variable can be used to provide the installer with an existing token. For example, a regenerated token in Hub UI will invalidate an existing token. Use **automationhub_api_token** to use that token in the installer the next time you run the installer.
automationhub_auto_sign_collections	**hub_collection_auto_sign**	If a collection signing service is enabled, collections are not signed automatically by default. Setting this parameter to **true** signs them by default. Default = **false**
automationhub_backup_collections		*Optional* Ansible automation hub provides artifacts in **/var/lib/pulp**. Automation controller automatically backs up the artifacts by default. You can also set **automationhub_backup_collections** to **false** and the backup and restore process will not backup or restore **/var/lib/pulp**. Default = **true**

RPM variable name	Container variable name	Description
automationhub_collection_download_count		*Optional* Determines whether download count is displayed on the UI. Default = **false**
automationhub_collection_seed_repository		When you run the bundle installer, validated content is uploaded to the **validated** repository, and certified content is uploaded to the **rh-certified** repository. By default, both certified and validated content are uploaded. Possible values of this variable are **certified** or **validated**. If you do not want to install content, set **automationhub_seed_collections** to **false** to disable the seeding. If you only want one type of content, set **automationhub_seed_collections** to **true** and **automationhub_collection_seed_repository** to the type of content you do want to include.
automationhub_collection_signing_service_key	**hub_collection_signing_key**	If a collection signing service is enabled, you must provide this variable to ensure that collections can be properly signed. **/absolute/path/to/key/to/sign**
automationhub_collection_signing_service_script		If a collection signing service is enabled, you must provide this variable to ensure that collections can be properly signed. **/absolute/path/to/script/that/signs**

RPM variable name	Container variable name	Description
automationhub_container_signing_service_key	**hub_container_signing_key**	If a container signing service is enabled, you must provide this variable to ensure that containers can be properly signed. **/absolute/path/to/key/to/sign**
automationhub_container_signing_service_script		If a container signing service is enabled, you must provide this variable to ensure that containers can be properly signed. **/absolute/path/to/script/that/signs**
automationhub_create_default_collection_signing_service	**hub_collection_signing_service**	Set this variable to **true** to create a collection signing service. Default = **false**
automationhub_create_default_container_signing_service	**hub_container_signing_service**	Set this variable to **true** to create a container signing service. Default = **false**
automationhub_disable_hsts	**hub_nginx_disable_hsts**	The default installation deploys a TLS enabled automation hub. Use this variable if you deploy automation hub with *HTTP Strict Transport Security* (HSTS) web-security policy enabled. This variable disables the HSTS web-security policy mechanism. Default = **false**
automationhub_disable_https	**hub_nginx_disable_https**	*Optional* If automation hub is deployed with HTTPS enabled. Default = **false**

RPM variable name	Container variable name	Description
automationhub_enable_analytics		A Boolean indicating whether to enable pulp analytics for the version of **pulpcore** used in automation hub in Ansible Automation Platform 2.5. To enable pulp analytics, set **automationhub_enable_analytics** to **true**. Default = **false**
automationhub_enable_api_access_log		When set to **true**, this variable creates a log file at **/var/log/galaxy_api_access.log** that logs all user actions made to the platform, including their username and IP address. Default = **false**
automationhub_enable_unauthenticated_collection_access		Set this variable to **true** to enable unauthorized users to view collections. Default = **false**
automationhub_enable_unauthenticated_collection_download		Set this variable to **true** to enable unauthorized users to download collections. Default = **false**
automationhub_importer_settings	**hub_galaxy_importer**	*Optional* Dictionary of setting to pass to galaxy-importer. At import time, collections can go through a series of checks. Behavior is driven by **galaxy-importer.cfg** configuration. Examples are **ansible-doc**, **ansible-lint**, and **flake8**. This parameter enables you to drive this configuration.

RPM variable name	Container variable name	Description
automationhub_pg_database	**hub_pg_database**	The PostgreSQL database name. RPM default = **automationhub** Container default = **pulp**
automationhub_pg_host	**hub_pg_host**	Required if not using an internal database. The hostname of the remote PostgreSQL database used by automation hub. Default = **127.0.0.1**
automationhub_pg_password	**hub_pg_password**	Required if not using an internal database. The password for the automation hub PostgreSQL database. Use of special characters for this variable is limited. The **!**, **#**, **0** and **@** characters are supported. Use of other special characters can cause the setup to fail.
automationhub_pg_port	**hub_pg_port**	Required if not using an internal database. Default = **5432**
automationhub_pg_sslmode		*Required* Default = **prefer**
automationhub_pg_username	**hub_pg_username**	The username for your automation hub PostgreSQL database RPM default = **automationhub** Container default = **pulp**

RPM variable name	Container variable name	Description
automationhub_require_content_approval		*Optional* Value is **true** if automation hub enforces the approval mechanism before collections are made available. By default when you upload collections to automation hub, an administrator must approve it before they are made available to the users. If you want to disable the content approval flow, set the variable to **false**. Default = **true**
automationhub_seed_collections		A Boolean that defines whether or not pre-loading is enabled. When you run the bundle installer, validated content is uploaded to the **validated** repository, and certified content is uploaded to the **rh-certified** repository. By default, both certified and validated content are uploaded. If you do not want to install content, set **automationhub_seed_collections** to **false** to disable the seeding. If you only want one type of content, set **automationhub_seed_collections** to **true** and **automationhub_collection_seed_repository** to the type of content you do want to include. Default = **true**

RPM variable name	Container variable name	Description
automationhub_ssl_cert	**hub_tls_cert**	*Optional* **/path/to/automationhub.cert** Same as **web_server_ssl_cert** but for automation hub UI and API.
automationhub_ssl_key	**hub_tls_key**	*Optional* **/path/to/automationhub.key**. Same as **web_server_ssl_key** but for automation hub UI and API.
automationhub_user_headers		List of nginx headers for Ansible automation hub's web server. Each element in the list is provided to the web server's nginx configuration as a separate line. Default = empty list
ee_from_hub_only		When deployed with automation hub, the installer pushes execution environment images to automation hub and configures automation controller to pull images from the automation hub registry. To make automation hub the only registry to pull execution environment images from, set this variable to **true**. If set to **false**, execution environment images are also taken directly from Red Hat. Default = **true** when the bundle installer is used.

RPM variable name	Container variable name	Description
generate_automationhub_token		When performing a fresh installation, a new token will automatically be generated by default. If you want the installer to regenerate a new token, set **generate_automationhub_token=true** and the installer will use it in the installation process.
nginx_hsts_max_age	**hub_nginx_hsts_max_age**	This variable specifies how long, in seconds, the system should be considered as an *HTTP Strict Transport Security* (HSTS) host. That is, how long HTTPS is used exclusively for communication. Default = **63072000** seconds, or two years.
pulp_db_fields_key		Relative or absolute path to the Fernet symmetric encryption key that you want to import. The path is on the Ansible management node. It is used to encrypt certain fields in the database, such as credentials. If not specified, a new key will be generated.
	hub_tls_remote	Automation hub TLS remote files. Default = **false**
	hub_main_url	Automation hub main URL.
	hub_nginx_client_max_body_size	NGINX maximum body size. Default = **20m**
	hub_nginx_http_port	NGINX HTTP port. Default = **8081**
	hub_nginx_https_port	NGINX HTTPS port. Default = **8444**
	hub_nginx_https_protocols	NGINX HTTPS protocols. Default = **[TLSv1.2, TLSv1.3]**

RPM variable name	Container variable name	Description
	hub_pg_socket	PostgreSQL Automation hub UNIX socket.
	hub_secret_key	Automation hub secret key.
	hub_storage_backend	Automation hub storage backend.
	hub_workers	Automation hub workers count.
	hub_collection_signing	Enable Automation hub collection signing. Default = **false**
	hub_container_signing	Enable Automation hub container signing. Default = **false**
	hub_container_signing_pass	Automation hub container signing passphrase.
	hub_collection_signing_pass	Automation hub collection signing passphrase.
	hub_postinstall	Enable Automation hub postinstall. Default = **false**
	hub_postinstall_async_delay	Postinstall delay between retries. Default = **1**
	hub_postinstall_async_retries	Postinstall number of retries to perform. Default = **30**
	hub_postinstall_dir	Automation hub postinstall directory.
	hub_postinstall_ignore_files	Automation hub ignore files.
	hub_postinstall_repo_ref	Automation hub repository branch or tag. Default = **main**

RPM variable name	Container variable name	Description
	hub_postinstall_repo_url	Automation hub repository URL.

B.3. AUTOMATION CONTROLLER VARIABLES

RPM variable name	Container variable name	Description
admin_email		The email address used for the admin user for automation controller.
admin_password	**controller_admin_password**	*Required* Automation controller admin password. Passwords must be enclosed in quotes when they are provided in plain text in the inventory file. Use of special characters for this variable is limited. The password can include any printable ASCII character except /, ", or @.
admin_username	**controller_admin_user**	Automation controller admin user. Default = **admin**
automation_controller_main_url		Automation controller main URL.
controller_tls_files_remote	**controller_tls_remote**	Automation controller TLS remote files. Default = **false**
nginx_disable_hsts	**controller_nginx_disable_hsts**	Disable NGINX HTTP Strict Transport Security (HSTS). Default = **false**
nginx_disable_https	**controller_nginx_disable_https**	Disable NGINX HTTPS. Default = **false**

RPM variable name	Container variable name	Description
nginx_hsts_max_age	**controller_nginx_hsts_max_age**	This variable specifies how long, in seconds, the system must be considered as an *HTTP Strict Transport Security* (HSTS) host. That is, how long HTTPS is used only for communication. Default = **63072000** seconds, or two years.
nginx_http_port	**controller_nginx_http_port**	The NGINX HTTP server listens for inbound connections. RPM default = **80** Container default = **8080**
nginx_https_port	**controller_nginx_https_port**	The NGINX HTTPS server listens for secure connections. RPM Default = **443** Container default = **8443**
nginx_user_headers	**controller_nginx_user_headers**	List of NGINX headers for the automation controller web server. Each element in the list is provided to the web server's NGINX configuration as a separate line. Default = empty list
node_state		*Optional* The status of a node or group of nodes. Valid options are **active**, **deprovision** to remove a node from a cluster, or **iso_migrate** to migrate a legacy isolated node to an execution node. Default = **active**

RPM variable name	Container variable name	Description
node_type		For **[automationcontroller]** group. Two valid **node_types** can be assigned for this group. A **node_type=control** means that the node only runs project and inventory updates, but not regular jobs. A **node_type=hybrid** can run everything. Default for this group = **hybrid**. For **[execution_nodes]** group: Two valid **node_types** can be assigned for this group. A **node_type=hop** implies that the node forwards jobs to an execution node. A **node_type=execution** implies that the node can run jobs. Default for this group = **execution**.
peers		*Optional* The **peers** variable is used to indicate which nodes a specific host or group connects to. Wherever this variable is defined, an outbound connection to the specific host or group is established. This variable is used to add **tcp-peer** entries in the **receptor.conf** file used for establishing network connections with other nodes. The peers variable can be a comma-separated list of hosts and groups from the inventory. This is resolved into a set of hosts that is used to construct the **receptor.conf** file.

RPM variable name	Container variable name	Description
pg_database	**controller_pg_database**	The name of the PostgreSQL database. Default = **awx**
pg_host	**controller_pg_host**	*Required* The PostgreSQL host, which can be an externally managed database.
pg_password	**controller_pg_password**	*Required* The password for the PostgreSQL database. Use of special characters for this variable is limited. The **!**, **#**, **0** and **@** characters are supported. Use of other special characters can cause the setup to fail. NOTE You no longer have to provide a **pg_hashed_password** in your inventory file at the time of installation, because PostgreSQL 13 can now store user passwords more securely. When you supply **pg_password** in the inventory file for the installer, PostgreSQL uses the SCRAM-SHA-256 hash to secure that password as part of the installation process.
pg_port	**controller_pg_port**	The PostgreSQL port to use. Default = **5432**
pg_username	**controller_pg_username**	Your PostgreSQL database username. Default = **awx**.

RPM variable name	Container variable name	Description
supervisor_start_retry_count		When specified, it adds **startretries = <value specified>** to the supervisor config file (/**etc/supervisord.d/tower.ini**). See program:x Section Values for more information about **startretries**. No default value exists.
web_server_ssl_cert	**controller_tls_cert**	*Optional* **/path/to/webserver.cert** Same as **automationhub_ssl_cert** but for web server UI and API.
web_server_ssl_key	**controller_tls_key**	*Optional* **/path/to/webserver.key** Same as **automationhub_server_ssl_key** but for web server UI and API.
	controller_event_workers	Automation controller event workers. Default = **4**
	controller_license_file	The location of your automation controller license file. For example: **controller_license_file=/path/to/license.zip** If you are defining this variable as part of the postinstall process (**controller_postinstall = true**), then you need to also set the **controller_postinstall_dir** variable.
	controller_nginx_client_max_body_size	NGINX maximum body size. Default = **5m**

RPM variable name	Container variable name	Description
	controller_nginx_https_protocols	NGINX HTTPS protocols. Default = **[TLSv1.2, TLSv1.3]**
	controller_pg_socket	PostgreSQL Controller UNIX socket.
	controller_secret_key	Automation controller secret key.
	controller_uwsgi_listen_queue_size	Automation controller uWSGI listen queue size. Default = **2048**
	controller_postinstall	Enable or disable the postinstall feature of the containerized installer. If set to **true**, then you also need to set **controller_license_file** and **controller_postinstall_dir**. Default = **false**
	controller_postinstall_dir	The location of your automation controller postinstall directory.
	controller_postinstall_async_delay	Postinstall delay between retries. Default = **1**
	controller_postinstall_async_retries	Postinstall number of tries to attempt. Default = **30**
	controller_postinstall_ignore_files	Automation controller ignore files.
	controller_postinstall_repo_ref	Automation controller repository branch or tag. Default = **main**
	controller_postinstall_repo_url	Automation controller repository URL.

B.4. EVENT-DRIVEN ANSIBLE CONTROLLER VARIABLES

RPM variable name	Container variable name	Description
automationedacontroller_activation_workers	**eda_activation_workers**	*Optional* Number of workers for ansible-rulebook activation pods in Event-Driven Ansible. Default = (# of cores or threads) * 2 + 1
automationedacontroller_admin_email	**eda_admin_email**	*Optional* Email address used by Django for the admin user for Event-Driven Ansible controller. Default = **admin@example.com**
automationedacontroller_admin_password	**eda_admin_password**	*Required* The admin password used by the Event-Driven Ansible controller instance. Passwords must be enclosed in quotes when they are provided in plain text in the **inventory** file. Use of special characters for this variable is limited. The password can include any printable ASCII character except /, **"**, or **@**.
automationedacontroller_admin_username	**eda_admin_user**	Username used by Django to identify and create the admin superuser in Event-Driven Ansible controller. Default = **admin**
automationedacontroller_allowed_hostnames		List of additional addresses to enable for user access to Event-Driven Ansible controller. Default = empty list

RPM variable name	Container variable name	Description
automationedacontroller_controller_verify_ssl		Boolean flag used to verify automation controller's web certificates when making calls from Event-Driven Ansible controller. Verified is **true** and not verified is **false**. Default = **false**
automationedacontroller_disable_hsts	**eda_nginx_disable_hsts**	*Optional* Boolean flag to disable HSTS for Event-Driven Ansible controller. Default = **false**
automationedacontroller_disable_https	**eda_nginx_disable_https**	*Optional* Boolean flag to disable HTTPS for Event-Driven Ansible controller. Default = **false**
automationedacontroller_event_stream_path	**eda_event_stream_prefix_path**	API prefix path used for Event-Driven Ansible event-stream through platform gateway. Default = **/eda-event-streams**
automationedacontroller_gunicorn_workers	**eda_gunicorn_workers**	Number of workers for the API served through Gunicorn. Default = (# of cores or threads) * 2 + 1
automationedacontroller_max_running_activations	**eda_max_running_activations**	*Optional* The number of maximum activations running concurrently per node. This is an integer that must be greater than 0. Default = **12**

RPM variable name	Container variable name	Description
automationedacontroller_nginx_tls_files_remote	**eda_tls_remote**	Boolean flag to specify whether cert sources are on the remote host (true) or local (false). Default = **false**
automationedacontroller_pg_database	**eda_pg_database**	*Optional* The PostgreSQL database used by Event-Driven Ansible controller. RPM default = **automationedacontroller** Container default = **eda**
automationedacontroller_pg_password	**eda_pg_password**	*Required* The password for the PostgreSQL database used by Event-Driven Ansible controller. Use of special characters for this variable is limited. The **!**, **#**, **0** and **@** characters are supported. Use of other special characters can cause the setup to fail.
automationedacontroller_pg_port	**eda_pg_port**	*Optional* The port number of the PostgreSQL database used by Event-Driven Ansible controller. Default = **5432**
automationedacontroller_pg_username	**eda_pg_username**	*Optional* The username for your Event-Driven Ansible controller PostgreSQL database. RPM default = **automationedacontroller** Container default = **eda**
automationedacontroller_redis_host	**eda_redis_host**	The Redis hostname used by Event-Driven Ansible controller.

RPM variable name	Container variable name	Description
automationedacontroller_redis_port	**eda_redis_port**	The port used for the Redis host defined by **automationedacontroller_redis_host** for Event-Driven Ansible controller.
automationedacontroller_rq_workers		Number of Redis Queue (RQ) workers used by Event-Driven Ansible controller. RQ workers are Python processes that run in the background. Default = (# of cores or threads) * 2 + 1
automationedacontroller_ssl_cert	**eda_tls_cert**	*Optional* **/root/ssl_certs/eda.<example>.com.crt** Same as **automationhub_ssl_cert** but for Event-Driven Ansible controller UI and API.
automationedacontroller_ssl_key	**eda_tls_key**	*Optional* **/root/ssl_certs/eda.<example>.com.key** Same as **automationhub_server_ssl_key** but for Event-Driven Ansible controller UI and API.
automationedacontroller_user_headers	**eda_nginx_user_headers**	List of additional NGINX headers to add to Event-Driven Ansible controller's NGINX configuration. Default = empty list
automationnedacontroller_pg_host	**eda_pg_host**	*Required* The hostname of the PostgreSQL database used by Event-Driven Ansible controller, which can be an externally managed database.

RPM variable name	Container variable name	Description
eda_node_type	**eda_type**	*Optional* Event-Driven Ansible controller node type. Default = **hybrid**
	eda_debug	Event-Driven Ansible controller debug. Default = **false**
	eda_event_stream_url	Event-Driven Ansible controller event stream URL.
	eda_main_url	Event-Driven Ansible controller main URL.
	eda_nginx_client_max_body_size	NGINX maximum body size. Default = **1m**
	eda_nginx_hsts_max_age	NGINX HSTS maximum age. Default = **63072000**
	eda_nginx_http_port	NGINX HTTP port. Default = **8082**
	eda_nginx_https_port	NGINX HTTPS port. Default = **8445**
	eda_nginx_https_protocols	NGINX HTTPS protocols. Default = **[TLSv1.2, TLSv1.3]**
	eda_pg_socket	PostgreSQL Event-Driven Ansible UNIX socket.
	eda_redis_disable_tls	Disable TLS Redis (for many nodes). Default = **false**
	eda_redis_password	Redis Event-Driven Ansible controller password (for many nodes).

RPM variable name	Container variable name	Description
	eda_redis_tls_cert	Event-Driven Ansible controller Redis TLS certificate.
	eda_redis_tls_key	Event-Driven Ansible controller Redis TLS key.
	eda_redis_username	Redis Event-Driven Ansible controller username (for many nodes).
	eda_safe_plugins	Event-Driven Ansible controller safe plugins.
	eda_secret_key	Event-Driven Ansible controller secret key.
	eda_workers	Event-Driven Ansible controller workers count. Default = **2**

B.5. PLATFORM GATEWAY VARIABLES

RPM variable name	Container variable name	Description
automationgateway_admin_email	**gateway_admin_email**	The email address used for the admin user for platform gateway.
automationgateway_admin_password	**gateway_admin_password**	*Required* The admin password used to connect to the platform gateway instance. Passwords must be enclosed in quotes when they are provided in plain text in the **inventory** file. Use of special characters for this variable is limited. The password can include any printable ASCII character except /, ", or @.

RPM variable name	Container variable name	Description
automationgateway_admin_username	**gateway_admin_user**	*Optional* The username used to identify and create the admin superuser in platform gateway. Default = **admin**
automationgateway_disable_hsts	**gateway_nginx_disable_hsts**	*Optional* Disable NGINX HSTS. Default = **false**
automationgateway_disable_https	**gateway_nginx_disable_https**	*Optional* Disable NGINX HTTPS. Default = **false**
automationgateway_grpc_auth_service_timeout	**gateway_grpc_auth_service_timeout**	Platform gateway auth server timeout. Default = **30s**
automationgateway_grpc_server_max_threads_per_process	**gateway_grpc_server_max_threads_per_process**	Platform gateway auth server threads per process. Default = **10**
automationgateway_grpc_server_processes	**gateway_grpc_server_processes**	Platform gateway auth server processes Default = **5**
automationgateway_main_url	**gateway_main_url**	*Optional* The main platform gateway URL that clients will connect to (e.g. **https://<gateway_node>**). If not specified, the first the first node in the **[automationgateway]** group will be used when needed.

RPM variable name	Container variable name	Description
automationgateway_pg_database	**gateway_pg_database**	*Optional* The PostgreSQL database used by platform gateway. RPM default = **automationgateway** Container default = **gateway**
automationgateway_pg_host	**gateway_pg_host**	*Required* The hostname of the PostgreSQL database used by platform gateway, which can be an externally managed database.
automationgateway_pg_password	**gateway_pg_password**	*Required* The password for the PostgreSQL database used by platform gateway. Use of special characters for **automationgateway_pg_password** is limited. The **!**, **#**, **0** and **@** characters are supported. Use of other special characters can cause the setup to fail.
automationgateway_pg_port	**gateway_pg_port**	*Optional* The port number of the PostgreSQL database used by platform gateway. Default = **5432**
automationgateway_pg_sslmode	**gateway_pg_sslmode**	Choose one of the two available modes: **prefer** and **verify-full**. Set to **verify-full** for client-side enforced SSL. Default = **prefer**

RPM variable name	Container variable name	Description
automationgateway_pg_username	**gateway_pg_username**	*Optional* The username for your platform gateway PostgreSQL database. RPM default = **automationgateway** Container default = **gateway**
automationgateway_redis_host	**gateway_redis_host**	The Redis hostname used by platform gateway.
automationgateway_redis_port	**gateway_redis_port**	The Redis platform gateway port. Default = **6379**
automationgateway_ssl_cert	**gateway_tls_cert**	*Optional* **/path/to/automationgateway.cert** Same as **automationhub_ssl_cert** but for platform gateway UI and API.
automationgateway_ssl_key	**gateway_tls_key**	*Optional* **/path/to/automationgateway.key** Same as **automationhub_server_ssl_key** but for platform gateway UI and API.
	gateway_nginx_client_max_body_size	NGINX maximum body size. Default = **5m**
	gateway_nginx_hsts_max_age	NGINX HSTS maximum age. Default = **63072000**
	gateway_nginx_http_port	NGINX HTTP port.
	gateway_nginx_https_port	NGINX HTTPS port.
	gateway_nginx_https_protocols	NGINX HTTPS protocols. Default = **[TLSv1.2, TLSv1.3]**

RPM variable name	Container variable name	Description
	gateway_nginx_user_headers	Custom NGINX headers.
	gateway_redis_disable_tls	Disable TLS Redis. Default = **false**
	gateway_redis_password	Redis platform gateway password.
	gateway_redis_tls_cert	Platform gateway Redis TLS certificate.
	gateway_redis_tls_key	Platform gateway Redis TLS key.
	gateway_redis_username	Redis platform gateway username. Default = **gateway**
	gateway_secret_key	Platform gateway secret key.
	gateway_tls_remote	Platform gateway TLS remote files. Default = **false**
	gateway_uwsgi_listen_queue_size	Platform gateway uWSGI listen queue size. Default = **4096**

B.6. DATABASE VARIABLES

RPM variable name	Container variable name	Description
pg_ssl_mode		Choose one of the two available modes: **prefer** and **verify-full**. Set to **verify-full** for client-side enforced SSL. Default = **prefer**
postgres_ssl_cert	**postgresql_tls_cert**	Location of the PostgreSQL SSL/TLS certificate. **/path/to/pgsql_ssl.cert**

RPM variable name	Container variable name	Description
postgres_ssl_key	**postgresql_tls_key**	Location of the PostgreSQL SSL/TLS key. **/path/to/pgsql_ssl.key**
postgres_use_cert		Location of the PostgreSQL user certificate. **/path/to/pgsql.crt**
postgres_use_ssl	**postgresql_disable_tls**	Determines if the connection between Ansible Automation Platform and the PostgreSQL database should use SSL/TLS. The default for this variable is **false** which means SSL/TLS is not used for PostgreSQL connections. When set to **true**, the platform connects to PostgreSQL by using SSL/TLS.
postgres_max_connections	**postgresql_max_connections**	Maximum database connections setting to apply if you are using installer-managed PostgreSQL. See PostgreSQL database configuration and maintenance for automation controller for help selecting a value. Default = **1024**
	postgresql_admin_database	PostgreSQL admin database. Default = **postgres**
	postgresql_admin_username	PostgreSQL admin user. Default = **postgres**
	postgresql_admin_password	*Required* PostgreSQL admin password.
	postgresql_effective_cache_size	PostgreSQL effective cache size.
	postgresql_keep_databases	Keep databases during uninstall. Default = **false**

RPM variable name	Container variable name	Description
	postgresql_log_destination	PostgreSQL log file location. Default = **/dev/stderr**
	postgresql_password_encryption	PostgreSQL password encryption. Default = **scram-sha-256**
	postgresql_shared_buffers	PostgreSQL shared buffers.
	postgresql_tls_remote	PostgreSQL TLS remote files. Default = **false**
	postgresql_port	PostgreSQL port number. Default = **5432**

B.7. IMAGE VARIABLES

RPM variable name	Container variable name	Description
	controller_image	Automation controller image. Default = **controller-rhel8:latest**
	de_extra_images	Decision environment extra images.
	de_supported_image	Decision environment supported image. Default = **de-supported-rhel8:latest**
	eda_image	Event-Driven Ansible image. Default = **eda-controller-rhel8:latest**
	eda_web_image	Event-Driven Ansible web image. Default = **eda-controller-ui-rhel8:latest**

RPM variable name	Container variable name	Description
	ee_29_enabled	Enable execution environment 29. Default = **false**
	ee_29_image	Execution environment 29 image. Default = **ee-29-rhel8:latest**
	ee_extra_images	Execution environment extra images.
	ee_minimal_image	Execution environment minimal image. Default = **ee-minimal-rhel8:latest**
	ee_supported_image	Execution environment supported image. Default = **ee-supported-rhel8:latest**
	hub_image	Automation hub image. Default = **hub-rhel8:latest**
	hub_web_image	Automation hub web image. Default = **hub-web-rhel8:latest**
	postgresql_image	PostgreSQL image. Default = **postgresql-15:latest**
	receptor_image	Receptor image. Default = **receptor-rhel8:latest**
	redis_image	Redis image. Default = **redis-6:latest**
	pcp_image	Performance Co-Pilot image. Default = **rhel8-pcp:latest**

B.8. RECEPTOR VARIABLES

RPM variable name	Container variable name	Description
	receptor_disable_signing	Disable receptor signing. Default = **false**
	receptor_disable_tls	Disable receptor TLS. Default = **false**
	receptor_log_level	Receptor logging level. Default = **info**
	receptor_mintls13	Receptor TLS 1.3 minimal. Default = **false**
	receptor_peers	Receptor peers list.
	receptor_port	Receptor port. Default = **27199**
	receptor_protocol	Receptor protocol. Default = **tcp**
	receptor_signing_private_key	Receptor signing private key.
	receptor_signing_public_key	Receptor signing public key.
	receptor_signing_remote	Receptor signing remote files. Default = **false**
	receptor_tls_cert	Receptor TLS certificate.
	receptor_tls_key	Receptor TLS key.
	receptor_tls_remote	Receptor TLS remote files. Default = **false**
	receptor_type	Receptor node type. Default = **execution**

B.9. ANSIBLE VARIABLES

The following variables control how Ansible Automation Platform interacts with remote hosts.

For more information about variables specific to certain plugins, see the documentation for Ansible.Builtin.

For a list of global configuration options, see Ansible Configuration Settings.

Variable	Description
ansible_connection	The connection plugin used for the task on the target host. This can be the name of any of Ansible connection plugins. SSH protocol types are **smart**, **ssh** or **paramiko**. Default = **smart**
ansible_host	The IP or name of the target host to use instead of **inventory_hostname**.
ansible_port	The connection port number. Default: 22 for ssh
ansible_user	The user name to use when connecting to the host.
ansible_password	The password to authenticate to the host. Never store this variable in plain text. Always use a vault.
ansible_ssh_private_key_file	Private key file used by SSH. Useful if using multiple keys and you do not want to use an SSH agent. Useful if using multiple keys and you do not want to use an SSH agent.
ansible_ssh_common_args	This setting is always appended to the default command line for **sftp**, **scp**, and **ssh**. Useful to configure a ProxyCommand for a certain host or group.
ansible_sftp_extra_args	This setting is always appended to the default **sftp** command line.
ansible_scp_extra_args	This setting is always appended to the default **scp** command line.
ansible_ssh_extra_args	This setting is always appended to the default **ssh** command line.

Variable	Description
ansible_ssh_pipelining	Determines if SSH pipelining is used. This can override the pipelining setting in **ansible.cfg**. If using SSH key-based authentication, the key must be managed by an SSH agent.
ansible_ssh_executable	Added in version 2.2. This setting overrides the default behavior to use the system SSH. This can override the ssh_executable setting in **ansible.cfg**.
ansible_ssh_executable	Added in version 2.2. This setting overrides the default behavior to use the system SSH. This can override the ssh_executable setting in 'ansible.cfg'.
ansible_shell_type	The shell type of the target system. Do not use this setting unless you have set the **ansible_shell_executable** to a non-Bourne (sh) compatible shell. By default commands are formatted using sh-style syntax. Setting this to **csh** or **fish** causes commands executed on target systems to follow the syntax of those shells instead.
ansible_shell_executable	This sets the shell that the Ansible controller uses on the target machine and overrides the executable in **ansible.cfg** which defaults to/**bin**/**sh**. Do not change this variable unless /**bin**/**sh** is not installed on the target machine or cannot be run from sudo.
inventory_hostname	This variable takes the hostname of the machine from the inventory script or the Ansible configuration file. You cannot set the value of this variable. Because the value is taken from the configuration file, the actual runtime hostname value can vary from what is returned by this variable.