Red Hat Ansible Automation Platform 2.5

Operating Ansible Automation Platform

Post installation configurations to ensure a smooth deployment of Ansible Automation Platform installation

Last Updated: 2024-10-30

Red Hat Ansible Automation Platform 2.5 Operating Ansible Automation Platform

Post installation configurations to ensure a smooth deployment of Ansible Automation Platform installation

Legal Notice

Abstract

This guide provides instructions and guidance on post installation activities for Red Hat Ansible Automation Platform.

Table of Contents

PREFACE

After installing Red Hat Ansible Automation Platform, your system might need extra configuration to ensure your deployment runs smoothly. This guide provides procedures for configuration tasks that you can perform after installing Red Hat Ansible Automation Platform.

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. POST-INSTALLATION STEPS

Whether you are a new Ansible Automation Platform user looking to start automating, or an existing administrator looking to migrate old Ansible content to your latest installed version of Red Hat Ansible Automation Platform, explore the next steps to begin using the new features of Ansible Automation Platform 2.5.

1.1. UPDATING EXECUTION ENVIRONMENT IMAGE LOCATIONS

If you installed private automation hub separately from Ansible Automation Platform, you can update your execution environment image locations to point to your private automation hub.

Procedure

1. Go to the directory that contains **setup.sh**
2. Create **./group_vars/automationcontroller** by running the following command:

```
touch ./group_vars/automationcontroller
```

3. Paste the following content into **./group_vars/automationcontroller**. Adjust the settings to fit your environment:

```
# Automation Hub Registry
registry_username: 'your-automation-hub-user'
registry_password: 'your-automation-hub-password'
registry_url: 'automationhub.example.org'
registry_verify_ssl: False

## Execution Environments
control_plane_execution_environment: 'automationhub.example.org/ee-supported-rhel8:latest'

global_job_execution_environments:
  - name: "Default execution environment"
    image: "automationhub.example.org/ee-supported-rhel8:latest"
  - name: "Minimal execution environment"
    image: "automationhub.example.org/ee-minimal-rhel8:latest"
```

4. Run the **./setup.sh** script

```
$ ./setup.sh
```

Verification

1. Log in to Ansible Automation Platform as a user with system administrator access.
2. Go to **Automation Execution → Infrastructure → Execution Environments**.
3. In the **Image** column, confirm that the execution environment image location has changed from the default value of **<registry url>/ansible-automation-platform-<version>/<image name>:<tag>** to **<automation hub url>/<image name>:<tag>**.

1.2. BENEFITS OF AUTOMATION MESH

The automation mesh component of the Red Hat Ansible Automation Platform simplifies the process of distributing automation across multi-site deployments. For enterprises with multiple isolated IT environments, automation mesh provides a consistent and reliable way to deploy and scale up automation across your execution nodes using a peer-to-peer mesh communication network.

Additional resources

- For information about automation mesh and the various ways to design your automation mesh for your environment:
  - For a VM-based installation, see the Automation mesh for VM environments.
  - For an operator-based installation, see the Automation mesh for managed cloud or operator environments

CHAPTER 2. CONFIGURING PROXY SUPPORT FOR RED HAT ANSIBLE AUTOMATION PLATFORM

You can configure Red Hat Ansible Automation Platform to communicate with traffic using a proxy. Proxy servers act as an intermediary for requests from clients seeking resources from other servers. A client connects to the proxy server, requesting some service or available resource from a different server, and the proxy server evaluates the request as a way to simplify and control its complexity. The following sections describe the supported proxy configurations and how to set them up.

2.1. ENABLE PROXY SUPPORT

To provide proxy server support, automation controller handles proxied requests (such as ALB, NLB , HAProxy, Squid, Nginx and tinyproxy in front of automation controller) via the **REMOTE_HOST_HEADERS** list variable in the automation controller settings. By default, **REMOTE_HOST_HEADERS** is set to **["REMOTE_ADDR", "REMOTE_HOST"]**.

To enable proxy server support, edit the **REMOTE_HOST_HEADERS** field in the settings page for your automation controller:

Procedure

1. From the navigation panel, select **Settings → System**.
2. In the **Remote Host Headers** field, enter the following values:

```
[
  "HTTP_X_FORWARDED_FOR",
  "REMOTE_ADDR",
  "REMOTE_HOST"
]
```

Automation controller determines the remote host's IP address by searching through the list of headers in **Remote Host Headers** until the first IP address is located.

2.2. KNOWN PROXIES

When automation controller is configured with **REMOTE_HOST_HEADERS = ['HTTP_X_FORWARDED_FOR', 'REMOTE_ADDR', 'REMOTE_HOST']**, it assumes that the value of **X-Forwarded-For** has originated from the proxy/load balancer sitting in front of automation controller. If automation controller is reachable without use of the proxy/load balancer, or if the proxy does not validate the header, the value of **X-Forwarded-For** can be falsified to fake the originating IP addresses. Using **HTTP_X_FORWARDED_FOR** in the **REMOTE_HOST_HEADERS** setting poses a vulnerability.

To avoid this, you can configure a list of known proxies that are allowed.

Procedure

1. From the navigation panel, select **Settings → System**.
2. Enter a list of proxy IP addresses from which the service should trust custom remote header values in the **Proxy IP Allowed List** field.

NOTE

Load balancers and hosts that are not on the known proxies list will result in a rejected request.

2.2.1. Configuring known proxies

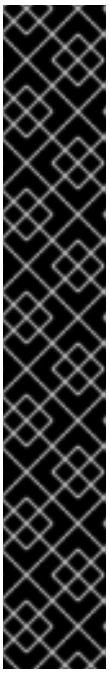
To configure a list of known proxies for your automation controller, add the proxy IP addresses to the **Proxy IP Allowed List** field in the System Settings page.

Procedure

1. From the navigation panel, select **Settings → System**.
2. In the **Proxy IP Allowed List** field, enter IP addresses that are allowed to connect to your automation controller, following the syntax in the example below:

Example Proxy IP Allowed List entry

```
[
  "example1.proxy.com:8080",
  "example2.proxy.com:8080"
]
```



IMPORTANT

- **Proxy IP Allowed List** requires proxies in the list are properly sanitizing header input and correctly setting an **X-Forwarded-For** value equal to the real source IP of the client. Automation controller can rely on the IP addresses and hostnames in **Proxy IP Allowed List** to provide non-spoofed values for **X-Forwarded-For**.`
- Do not configure **HTTP_X_FORWARDED_FOR** as an item in **Remote Host Headers** unless **all** of the following conditions are satisfied:
  - You are using a proxied environment with ssl termination;
  - The proxy provides sanitization or validation of the **X-Forwarded-For** header to prevent client spoofing;
  - **/etc/tower/conf.d/remote_host_headers.py** defines **PROXY_IP_ALLOWED_LIST** that contains only the originating IP addresses of trusted proxies or load balancers.

3. Click **Save** to save the settings.

2.3. CONFIGURING A REVERSE PROXY

You can support a reverse proxy server configuration by adding **HTTP_X_FORWARDED_FOR** to the **Remote Host Headers** field in the Systems Settings. The **X-Forwarded-For** (XFF) HTTP header field identifies the originating IP address of a client connecting to a web server through an HTTP proxy or load balancer.

Procedure

1. From the navigation panel, select **Settings → System**.
2. In the **Remote Host Headers** field, enter the following values:

```
[
  "HTTP_X_FORWARDED_FOR",
  "REMOTE_ADDR",
  "REMOTE_HOST"
]
```

3. Add the lines below to **/etc/tower/conf.d/custom.py** to ensure the application uses the correct headers:

```
USE_X_FORWARDED_PORT = True
USE_X_FORWARDED_HOST = True
```

4. Click **Save** to save the settings.

2.4. ENABLE STICKY SESSIONS

By default, an application load balancer routes each request independently to a registered target based on the chosen load-balancing algorithm. To avoid authentication errors when running multiple instances of automation hub behind a load balancer, you must enable sticky sessions. Enabling sticky sessions sets a custom application cookie that matches the cookie configured on the load balancer to enable stickiness. This custom cookie can include any of the cookie attributes required by the application.

Additional resources

- Refer to Sticky sessions for your Application Load Balancer for more information about enabling sticky sessions.

CHAPTER 3. CONFIGURING AUTOMATION CONTROLLER WEBSOCKET CONNECTIONS

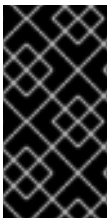
You can configure automation controller in order to align the websocket configuration with your nginx or load balancer configuration.

3.1. WEBSOCKET CONFIGURATION FOR AUTOMATION CONTROLLER

Automation controller nodes are interconnected through websockets to distribute all websocket-emitted messages throughout your system. This configuration setup enables any browser client websocket to subscribe to any job that might be running on any automation controller node. Websocket clients are not routed to specific automation controller nodes. Instead, any automation controller node can handle any websocket request and each automation controller node must know about all websocket messages destined for all clients.

You can configure websockets at **/etc/tower/conf.d/websocket_config.py** in all of your automation controller nodes and the changes will be effective after the service restarts.

Automation controller automatically handles discovery of other automation controller nodes through the Instance record in the database.



IMPORTANT

Your automation controller nodes are designed to broadcast websocket traffic across a private, trusted subnet (and not the open Internet). Therefore, if you turn off HTTPS for websocket broadcasting, the websocket traffic, composed mostly of Ansible playbook stdout, is sent unencrypted between automation controller nodes.

3.1.1. Configuring automatic discovery of other automation controller nodes

You can configure websocket connections to enable automation controller to automatically handle discovery of other automation controller nodes through the Instance record in the database.

1. Edit automation controller websocket information for port and protocol, and confirm whether to verify certificates with **True** or **False** when establishing the websocket connections:

```
BROADCAST_WEBSOCKET_PROTOCOL = 'http'
BROADCAST_WEBSOCKET_PORT = 80
BROADCAST_WEBSOCKET_VERIFY_CERT = False
```

2. Restart automation controller with the following command:

```
$ automation-controller-service restart
```

CHAPTER 4. MANAGING USABILITY ANALYTICS AND DATA COLLECTION FROM AUTOMATION CONTROLLER

You can change how you participate in usability analytics and data collection from automation controller by opting out or changing your settings in the automation controller user interface.

4.1. USABILITY ANALYTICS AND DATA COLLECTION

Usability data collection is included with automation controller to collect data to better understand how automation controller users specifically interact with automation controller, to help enhance future releases, and to continue streamlining your user experience.

Only users installing a trial of automation controller or a fresh installation of automation controller are opted-in for this data collection.

Additional resources

- For more information, see the Red Hat privacy policy .

4.1.1. Controlling data collection from automation controller

You can control how automation controller collects data from the **Settings → System** menu.

Procedure

1. Log in to your automation controller.
2. From the navigation panel, select **Settings → System**.
3. Select **Gather data for Automation Analytics**to enable automation controller to gather data on automation and send it to Automation Analytics.

CHAPTER 5. ENCRYPTING PLAINTEXT PASSWORDS IN AUTOMATION CONTROLLER CONFIGURATION FILES

Passwords stored in automation controller configuration files are stored in plain text. A user with access to the **/etc/tower/conf.d/** directory can view the passwords used to access the database. Access to the directories is controlled with permissions, so they are protected, but some security findings deem this protection to be inadequate. The solution is to encrypt the passwords individually.

5.1. CREATING POSTGRESQL PASSWORD HASHES

Procedure

1. On your automation controller node, run the following:

```
# awx-manage shell_plus
```

2. Then run the following from the python prompt:

```
>>> from awx.main.utils import encrypt_value, get_encryption_key \
>>> postgres_secret = encrypt_value('$POSTGRES_PASS') \
>>> print(postgres_secret)
```

NOTE

Replace the **$POSTGRES_PASS** variable with the actual plain text password you want to encrypt.

The output should resemble the following:

```
$encrypted$UTF8$AESCBC$Z0FBQUFBQmtLdGNRWXFjZGtkV1ZBR3hkNGVVbFFIU3hhY
21UT081eXFkR09aUWZLcG9TSmpndmZYQXFyRHVFQ3ZYSE15OUFuM1RHZHBqTFU3S
0MyNEo2Y2JWUURSYktsdmc9PQ==
```

3. Copy the full values of these hashes and save them.

- The hash value begins with **$encrypted$**, and is not just the string of characters, as shown in the following example:

```
$encrypted$AESCBC$Z0FBQUFBQmNONU9BbGQ1VjJyNDJRVTRKaFRIR09Ib2U5TGd
aYVRfcXFXRjlmdmpZNjdoZVpEZ21QRWViMmNDOGJaM0dPeHN2b194NUxvQ1M5X3d
Sc1gxQ29TdDBKRkljWHc9PQ==
```

Note that the **$*_PASS** values are already in plain text in your inventory file.

These steps supply the hash values that replace the plain text passwords within the automation controller configuration files.

5.2. ENCRYPTING THE POSTGRES PASSWORD

The following procedure replaces the plain text passwords with encrypted values. Perform the following steps on each node in the cluster:

Procedure

1. Edit **/etc/tower/conf.d/postgres.py** using:

```
$ vim /etc/tower/conf.d/postgres.py
```

2. Add the following line to the top of the file.

```
from awx.main.utils import decrypt_value, get_encryption_key
```

3. Remove the password value listed after 'PASSWORD': and replace it with the following line, replacing the supplied value of **$encrytpted..** with your own hash value:

```
decrypt_value(get_encryption_key('value'),'$encrypted$AESCBC$Z0FBQUFBQmNONU9BbG
Q1VjJyNDJRVTRKaFRIR09Ib2U5TGdaYVRfcXFXRjlmdmpZNjdoZVpEZ21QRWViMmNDOG
JaM0dPeHN2b194NUxvQ1M5X3dSc1gxQ29TdDBKRkljWHc9PQ==='),
```

NOTE

The hash value in this step is the output value of **postgres_secret**.

4. The full **postgres.py** resembles the following:

```
# Ansible Automation platform controller database settings. from awx.main.utils import
decrypt_value, get_encryption_key DATABASES = { 'default': { 'ATOMIC_REQUESTS': True,
'ENGINE': 'django.db.backends.postgresql', 'NAME': 'awx', 'USER': 'awx', 'PASSWORD':
decrypt_value(get_encryption_key('value'),'$encrypted$AESCBC$Z0FBQUFBQmNONU9BbG
Q1VjJyNDJRVTRKaFRIR09Ib2U5TGdaYVRfcXFXRjlmdmpZNjdoZVpEZ21QRWViMmNDOG
JaM0dPeHN2b194NUxvQ1M5X3dSc1gxQ29TdDBKRkljWHc9PQ==='), 'HOST': '127.0.0.1',
'PORT': 5432, } }
```

5.3. RESTARTING AUTOMATION CONTROLLER SERVICES

Procedure

1. When encryption is completed on all nodes, perform a restart of services across the cluster using:

```
# automation-controller-service restart
```

2. Navigate to the UI, and verify you are able to run jobs across all nodes.

CHAPTER 6. RENEWING AND CHANGING THE SSL CERTIFICATE

If your current SSL certificate has expired or will expire soon, you can either renew or replace the SSL certificate used by Ansible Automation Platform.

You must renew the SSL certificate if you need to regenerate the SSL certificate with new information such as new hosts.

You must replace the SSL certificate if you want to use an SSL certificate signed by an internal certificate authority.

6.1. RENEWING THE SELF-SIGNED SSL CERTIFICATE

The following steps regenerate a new SSL certificate for both automation controller and automation hub.

Procedure

1. Add **aap_service_regen_cert=true** to the inventory file in the **[all:vars]** section:

```
[all:vars]
aap_service_regen_cert=true
```

2. Run the installer.

Verification

- Validate the CA file and server.crt file on automation controller:

```
openssl verify -CAfile ansible-automation-platform-managed-ca-cert.crt /etc/tower/tower.cert
openssl s_client -connect <AUTOMATION_HUB_URL>:443
```

- Validate the CA file and server.crt file on automation hub:

```
openssl verify -CAfile ansible-automation-platform-managed-ca-cert.crt
/etc/pulp/certs/pulp_webserver.crt
openssl s_client -connect <AUTOMATION_CONTROLLER_URL>:443
```

6.2. CHANGING SSL CERTIFICATES

To change the SSL certificate, you can edit the inventory file and run the installer. The installer verifies that all Ansible Automation Platform components are working. The installer can take a long time to run.

Alternatively, you can change the SSL certificates manually. This is quicker, but there is no automatic verification.

Red Hat recommends that you use the installer to make changes to your Ansible Automation Platform instance.

6.2.1. Prerequisites

- If there is an intermediate certificate authority, you must append it to the server certificate.
- Both automation controller and automation hub use NGINX so the server certificate must be in PEM format.
- Use the correct order for the certificates: The server certificate comes first, followed by the intermediate certificate authority.

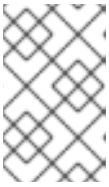
For further information, see the ssl certificate section of the NGINX documentation.

6.2.2. Changing the SSL certificate and key using the installer

The following procedure describes how to change the SSL certificate and key in the inventory file.

Procedure

1. Copy the new SSL certificates and keys to a path relative to the Ansible Automation Platform installer.
2. Add the absolute paths of the SSL certificates and keys to the inventory file. Refer to the Automation controller variables, Automation hub variables, and Event-Driven Ansible controller variables sections of RPM installation for guidance on setting these variables.
   - Automation controller: **web_server_ssl_cert**, **web_server_ssl_key**, **custom_ca_cert**
   - Automation hub: **automationhub_ssl_cert**, **automationhub_ssl_key**, **custom_ca_cert**
   - Event-Driven Ansible controller: **automationedacontroller_ssl_cert**, **automationedacontroller_ssl_key**, **custom_ca_cert**

   NOTE

   The **custom_ca_cert** must be the root certificate authority that signed the intermediate certificate authority. This file is installed in **/etc/pki/ca-trust/source/anchors**.
3. Run the installer.

6.2.3. Changing the SSL certificate manually

6.2.3.1. Changing the SSL certificate and key manually on automation controller

The following procedure describes how to change the SSL certificate and key manually on Automation Controller.

Procedure

1. Backup the current SSL certificate:

```
cp /etc/tower/tower.cert /etc/tower/tower.cert-$(date +%F)
```

2. Backup the current key files:

```
cp /etc/tower/tower.key /etc/tower/tower.key-$(date +%F)+
```

3. Copy the new SSL certificate to **/etc/tower/tower.cert**.
4. Copy the new key to **/etc/tower/tower.key**.
5. Restore the SELinux context:

```
restorecon -v /etc/tower/tower.cert /etc/tower/tower.key
```

6. Set appropriate permissions for the certificate and key files:

```
chown root:awx /etc/tower/tower.cert /etc/tower/tower.key
chmod 0600 /etc/tower/tower.cert /etc/tower/tower.key
```

7. Test the NGINX configuration:

```
nginx -t
```

8. Reload NGINX:

```
systemctl reload nginx.service
```

9. Verify that new SSL certificate and key have been installed:

```
true | openssl s_client -showcerts -connect ${CONTROLLER_FQDN}:443
```

6.2.3.2. Changing the SSL certificate and key on automation controller on OpenShift Container Platform

The following procedure describes how to change the SSL certificate and key for automation controller running on OpenShift Container Platform.

Procedure

1. Copy the signed SSL certificate and key to a secure location.
2. Create a TLS secret within OpenShift:

```
oc create secret tls ${CONTROLLER_INSTANCE}-certs-$(date +%F) --cert=/path/to/ssl.crt --
key=/path/to/ssl.key
```

3. Modify the automation controller custom resource to add **route_tls_secret** and the name of the new secret to the spec section.

```
oc edit automationcontroller/${CONTROLLER_INSTANCE}
```

```
...
spec:
  route_tls_secret: automation-controller-certs-2023-04-06
...
```

The name of the TLS secret is arbitrary. In this example, it is timestamped with the date that the secret is created, to differentiate it from other TLS secrets applied to the automation controller instance.

1. Wait a few minutes for the changes to be applied.
2. Verify that new SSL certificate and key have been installed:

```
true | openssl s_client -showcerts -connect ${CONTROLLER_FQDN}:443
```

6.2.3.3. Changing the SSL certificate and key on Event-Driven Ansible controller

The following procedure describes how to change the SSL certificate and key manually on Event-Driven Ansible controller.

Procedure

1. Backup the current SSL certificate:

```
cp /etc/ansible-automation-platform/eda/server.cert /etc/ansible-automation-platform/eda/server.cert-$(date +%F)
```

2. Backup the current key files:

```
cp /etc/ansible-automation-platform/eda/server.key /etc/ansible-automation-platform/eda/server.key-$(date +%F)
```

3. Copy the new SSL certificate to **/etc/ansible-automation-platform/eda/server.cert**.
4. Copy the new key to **/etc/ansible-automation-platform/eda/server.key**.
5. Restore the SELinux context:

```
restorecon -v /etc/ansible-automation-platform/eda/server.cert /etc/ansible-automation-platform/eda/server.key
```

6. Set appropriate permissions for the certificate and key files:

```
chown root:eda /etc/ansible-automation-platform/eda/server.cert /etc/ansible-automation-platform/eda/server.key
```

```
chmod 0600 /etc/ansible-automation-platform/eda/server.cert /etc/ansible-automation-platform/eda/server.key
```

7. Test the NGINX configuration:

```
nginx -t
```

8. Reload NGINX:

```
systemctl reload nginx.service
```

9. Verify that new SSL certificate and key have been installed:

```
true | openssl s_client -showcerts -connect ${CONTROLLER_FQDN}:443
```

6.2.3.4. Changing the SSL certificate and key manually on automation hub

The following procedure describes how to change the SSL certificate and key manually on automation hub.

Procedure

1. Backup the current SSL certificate:

```
cp /etc/pulp/certs/pulp_webserver.crt /etc/pulp/certs/pulp_webserver.crt-$(date +%F)
```

2. Backup the current key files:

```
cp /etc/pulp/certs/pulp_webserver.key /etc/pulp/certs/pulp_webserver.key-$(date +%F)
```

3. Copy the new SSL certificate to **/etc/pulp/certs/pulp_webserver.crt**.
4. Copy the new key to **/etc/pulp/certs/pulp_webserver.key**.
5. Restore the SELinux context:

```
restorecon -v /etc/pulp/certs/pulp_webserver.crt /etc/pulp/certs/pulp_webserver.key
```

6. Set appropriate permissions for the certificate and key files:

```
chown root:pulp /etc/pulp/certs/pulp_webserver.crt /etc/pulp/certs/pulp_webserver.key
```

```
chmod 0600 /etc/pulp/certs/pulp_webserver.crt /etc/pulp/certs/pulp_webserver.key
```

7. Test the NGINX configuration:

```
nginx -t
```

8. Reload NGINX:

```
systemctl reload nginx.service
```

9. Verify that new SSL certificate and key have been installed:

```
true | openssl s_client -showcerts -connect ${CONTROLLER_FQDN}:443
```