Red Hat Ansible Automation Platform 2.4

Red Hat Ansible Automation Platform operations guide

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

Last Updated: 2024-06-04

Red Hat Ansible Automation Platform 2.4 Red Hat Ansible Automation Platform operations guide

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

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.

MAKING OPEN SOURCE MORE INCLUSIVE

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

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

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

CHAPTER 1. ACTIVATING RED HAT ANSIBLE AUTOMATION PLATFORM

Red Hat Ansible Automation Platform uses available subscriptions or a subscription manifest to authorize the use of Ansible Automation Platform. To obtain a subscription, you can do either of the following:

1. Use your Red Hat customer or Satellite credentials when you launch Ansible Automation Platform.

2. Upload a subscriptions manifest file either using the Red Hat Ansible Automation Platform interface or manually in an Ansible playbook.

1.1. ACTIVATE WITH CREDENTIALS

When Ansible Automation Platform launches for the first time, the Ansible Automation Platform Subscription screen automatically displays. You can use your Red Hat credentials to retrieve and import your subscription directly into Ansible Automation Platform.

Procedures

1. Enter your Red Hat username and password.

2. Click **Get Subscriptions**.

   NOTE

   You can also use your Satellite username and password if your cluster nodes are registered to Satellite through Subscription Manager.

3. Review the End User License Agreement and select **I agree to the End User License Agreement**.

4. The Tracking and Analytics options are checked by default. These selections help Red Hat improve the product by delivering you a much better user experience. You can opt out by deselecting the options.

5. Click **Submit**.

6. Once your subscription has been accepted, the license screen displays and navigates you to the Dashboard of the Ansible Automation Platform interface. You can return to the license screen by clicking the **Settings** icon ⚙ and selecting the **License** tab from the Settings screen.

1.2. ACTIVATE WITH A MANIFEST FILE

If you have a subscriptions manifest, you can upload the manifest file either using the Red Hat Ansible Automation Platform interface or manually in an Ansible playbook.

Prerequisites

You must have a Red Hat Subscription Manifest file exported from the Red Hat Customer Portal. For more information, see Obtaining a manifest file .

[illegible]

Uploading with the interface

1. Complete steps to generate and download the manifest file
2. Log in to Red Hat Ansible Automation Platform.
3. If you are not immediately prompted for a manifest file, go to **Settings → License**.
4. Make sure the **Username** and **Password** fields are empty.
5. Click **Browse** and select the manifest file.
6. Click **Next**.

NOTE

If the **BROWSE** button is disabled on the License page, clear the **USERNAME** and **PASSWORD** fields.

Uploading manually

If you are unable to apply or update the subscription info using the Red Hat Ansible Automation Platform interface, you can upload the subscriptions manifest manually in an Ansible playbook using the **license** module in the **ansible.controller** collection.

```
- name: Set the license using a file
  license:
  manifest: "/tmp/my_manifest.zip"
```

CHAPTER 2. OBTAINING A MANIFEST FILE

You can obtain a subscription manifest in the Subscription Allocations section of Red Hat Subscription Management. After you obtain a subscription allocation, you can download its manifest file and upload it to activate Ansible Automation Platform.

To begin, login to the Red Hat Customer Portal using your administrator user account and follow the procedures in this section.

2.1. CREATE A SUBSCRIPTION ALLOCATION

Creating a new subscription allocation allows you to set aside subscriptions and entitlements for a system that is currently offline or air-gapped. This is necessary before you can download its manifest and upload it to Ansible Automation Platform.

Procedure

1. From the Subscription Allocations page, click **New Subscription Allocation**.
2. Enter a name for the allocation so that you can find it later.
3. Select **Type: Satellite 6.8** as the management application.
4. Click **Create**.

2.2. ADDING SUBSCRIPTIONS TO A SUBSCRIPTION ALLOCATION

Once an allocation is created, you can add the subscriptions you need for Ansible Automation Platform to run properly. This step is necessary before you can download the manifest and add it to Ansible Automation Platform.

Procedure

1. From the Subscription Allocations page, click on the name of the **Subscription Allocation** to which you would like to add a subscription.
2. Click the **Subscriptions** tab.
3. Click **Add Subscriptions**.
4. Enter the number of Ansible Automation Platform Entitlement(s) you plan to add.
5. Click **Submit**.

Verification

After your subscription has been accepted, subscription details are displayed. A status of *Compliant* indicates your subscription is in compliance with the number of hosts you have automated within your subscription count. Otherwise, your status will show as *Out of Compliance*, indicating you have exceeded the number of hosts in your subscription.

Other important information displayed include the following:

Hosts automated

Host count automated by the job, which consumes the license count

Hosts imported

Host count considering all inventory sources (does not impact hosts remaining)

Hosts remaining

Total host count minus hosts automated

2.3. DOWNLOADING A MANIFEST FILE

After an allocation is created and has the appropriate subscriptions on it, you can download the manifest from Red Hat Subscription Management.

Procedure

1. From the Subscription Allocations page, click on the name of the **Subscription Allocation** to which you would like to generate a manifest.
2. Click the **Subscriptions** tab.
3. Click **Export Manifest** to download the manifest file.

NOTE

The file is saved to your default downloads folder and can now be uploaded to activate Red Hat Ansible Automation Platform.

CHAPTER 3. 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 2.4.

3.1. STEPS TO MIGRATE DATA TO ANSIBLE AUTOMATION PLATFORM 2.4

For platform administrators looking to complete an upgrade to the Ansible Automation Platform 2.4, there may be additional steps needed to migrate data to a new instance:

To complete an upgrade to Ansible Automation Platform 2.4, you must migrate your data. Migrating data to a new instance requires additional steps.

3.1.1. Migrating from legacy virtual environments (venvs) to automation execution environments

Ansible Automation Platform 2.4 moves you away from custom Python virtual environments (venvs) in favor of automation execution environments - containerized images that package the necessary components needed to run and scale your Ansible automation. These components include ansible-core, Ansible Content Collections, Python dependencies, Red Hat Enterprise Linux UBI 8, and any additional package dependencies.

To migrate your venvs to execution environments, you must use the **awx-manage** command to list and export a list of venvs from your original instance, and then use **ansible-builder** to create execution environments.

Additional resources

- Upgrading to Automation Execution Environments guide
- Creating and Consuming Execution Environments .

3.1.2. Migrating Ansible Engine images using Ansible Builder

To migrate previous Ansible Engine images for use with Ansible Automation Platform 2.4, use the **ansible-builder** tool to automate the process of rebuilding images (including its custom plugins and dependencies) for use with automation execution environments.

Additional resources

- See Creating and Consuming Execution Environments for more information about using Ansible Builder to build execution environments.

3.1.3. Migrating to Ansible Core 2.13

When upgrading to ansible-core 2.13, you must update your playbooks and plugins, or other parts of your Ansible infrastructure to be supported by the latest version of ansible-core.

Additional resources

For instructions on updating your Ansible content for ansible-core 2.13 compatibility, see the Ansible-core 2.13 Porting Guide.

3.2. 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 **Administration → 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>**.

3.3. 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.

When upgrading from version 1.x to the latest version of Ansible Automation Platform, you must migrate the data from your legacy isolated nodes into execution nodes necessary for automation mesh. You can implement automation mesh by planning out a network of hybrid and control nodes, then editing the inventory file found in the Ansible Automation Platform installer to assign mesh-related values to each of your execution nodes.

Additional resources

- For instructions on how to migrate from isolated nodes to execution nodes, see the Red Hat Ansible Automation Platform Upgrade and Migration Guide.
- For information about automation mesh and the various ways to design your automation mesh for your environment:
  - For a VM-based installation, see the Red Hat Ansible Automation Platform automation mesh guide for VM-based installations.
  - For an operator-based installation, see the Red Hat Ansible Automation Platform automation mesh for operator-based installations.

CHAPTER 4. 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.

4.1. 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. On your automation controller, navigate to **Settings**.
2. Select **Miscellaneous System settings** from the list of **System** options.
3. 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.

4.2. 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 using the **PROXY_IP_ALLOWED_LIST** field in the settings menu on your automation controller. Load balancers and hosts that are not on the known proxies list will result in a rejected request.

4.2.1. 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 settings page for your automation controller.

Procedure

1. On your automation controller, navigate to **Settings** and select **Miscellaneous System settings** from the list of **System** options.

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 the **X-Forwarded-For** field.

- 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.

4.3. 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 your automation controller 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. On your automation controller, navigate to **Settings** and select **Miscellaneous System settings** from the list of **System** options.

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
```

4.4. 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.

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

CHAPTER 5. CONFIGURING AUTOMATION CONTROLLER WEBSOCKET CONNECTIONS

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

5.1. 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.

5.1.1. 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 6. 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.

6.1. 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 .

6.1.1. Controlling data collection from automation controller

You can control how automation controller collects data by setting your participation level in the **User Interface** tab in the **Settings** menu.

Procedure

1. Log in to your automation controller.
2. Navigate to **Settings** and select **User Interface settings** from the **User Interface** option.
3. Select the desired level of data collection from the **User Analytics Tracking State** drop-down list:
    - **Off**: Prevents any data collection.
    - **Anonymous**: Enables data collection without your specific user data.
    - **Detailed**: Enables data collection including your specific user data.
4. Click **Save** to apply the settings or **Cancel** to discard the changes.

CHAPTER 7. 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.

7.1. 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.

7.2. 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, } }
```

7.3. 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 8. 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.

8.1. 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
```

8.2. 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.

8.2.1. 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.

8.2.2. 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 the Red Hat Ansible Automation Platform Installation Guide 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.

8.2.3. Changing the SSL certificate manually

8.2.3.1. 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
```

8.2.3.2. 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
   ```

8.2.3.3. 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
```

8.2.3.4. 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
   ```