Red Hat Ansible Automation Platform 2.1

Red Hat Ansible Automation Platform Installation Guide

This guide provides procedures and reference information for the supported installation scenarios for Red Hat Ansible Automation Platform

Last Updated: 2023-09-13

Red Hat Ansible Automation Platform 2.1 Red Hat Ansible Automation Platform Installation Guide

This guide provides procedures and reference information for the supported installation scenarios for Red Hat Ansible Automation Platform

Legal Notice

Abstract

Providing Feedback: If you have a suggestion to improve this documentation, or find an error, please contact technical support at to create an issue on the Ansible Automation Platform Jira project using the Docs component.

Table of Contents

PREFACE

Thank you for your interest in Red Hat Ansible Automation Platform. 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 installing Ansible Automation Platform. This document has been updated to include information for the latest release of 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.

CHAPTER 1. PLANNING YOUR RED HAT ANSIBLE AUTOMATION PLATFORM INSTALLATION

You can use this section to help plan your Red Hat Ansible Automation Platform installation. Before installation, review information on the setup installer, system requirements, and supported installation scenarios.

1.1. RED HAT ANSIBLE AUTOMATION PLATFORM SYSTEM REQUIREMENTS

Use this information when planning your Red Hat Ansible Automation Platform installations and designing automation mesh topologies that fit your use case.

Your system must meet the following minimum system requirements to install and run Red Hat Ansible Automation Platform.

Table 1.1. Base system

	Required	Notes
Subscription	Valid Red Hat Ansible Automation Platform	
OS	Red Hat Enterprise Linux 8.4 or later 64-bit (x86)	
Ansible	version 2.11 required	If Ansible is not already present on the system, the setup playbook will install **ansible-core** 2.12.
Python	3.8 or later	

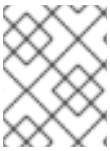
The following are necessary for you to work with project updates and collections:

- Ensure that the following domain names are part of either the firewall or the proxy's allowlist for successful connection and download of collections from automation hub or Galaxy server:
  - **galaxy.ansible.com**
  - **cloud.redhat.com**
  - **console.redhat.com**
  - **sso.redhat.com**
- SSL inspection must be disabled either when using self signed certificates or for the Red Hat domains.

1.1.1. Automation controller

Automation controller is a distributed system, where different software components can be co-located or deployed across multiple compute nodes. In the installer, node types of control, hybrid, execution,

and hop are provided as abstractions to help the user design the topology appropriate for their use case. The following table provides recommendations for node sizing:



NOTE

On all nodes except hop nodes, allocate a minimum of 20 GB to **/home/awx** for execution environment storage.

Execution nodes	Required	Notes
RAM	16 GB	
CPUs	4	• Runs automation. Increase memory and CPU to increase capacity for running more forks
Control nodes	**Required**	**Notes**
RAM	16 GB	
CPUs	4	• Processes events and runs cluster jobs including project updates and cleanup jobs. Increasing CPU and memory can help with job event processing.
Hybrid nodes	**Required**	**Notes**
RAM	16 GB	• Notes on RAM for execution and control nodes also apply to this node type.
CPUs	4	• Runs both automation and cluster jobs. Notes on CPUs for execution and control nodes also apply to this node type.
Hop nodes	**Required**	**Notes**
RAM	16 GB	

CPUs	4	• Serves to route traffic from one part of the Automation Mesh to another (for example, could be a bastion host into another network). RAM could affect throughput, CPU activity is low. Network bandwidth and latency generally a more important factor than either RAM/CPU.
Disk: service node	40 GB dedicated hard disk space	• **automation controller**: dedicate a minimum of 20 GB to **/var/** for file and working directory storage • Storage volume should be rated for a minimum baseline of 1500 IOPS. • Projects are stored on control and hybrid, and for the duration of jobs, also on execution nodes. If the cluster has many large projects, consider having twice the GB in /var/lib/awx/projects, to avoid disk space errors.
Database node	**Required**	**Notes**
RAM	16 GB	
CPUs	4	
Disk	20 GB dedicated hard disk space	• Minimum dedicated hard disk space is 20 GB • 150 GB+ recommended • Storage volume should be rated for a high baseline IOPS (1500 or more).

Browser	A currently supported version of Mozilla FireFox or Google Chrome	
Database	PostgreSQL version 12	

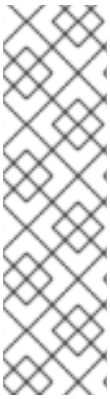
Additional resources

- To authorize the use of automation controller, see Import a subscription.

Table 1.2. Automation hub

	Required	Notes
RAM	8 GB minimum	• 8 GB RAM (minimum and recommended for Vagrant trial installations) • 8 GB RAM (minimum for external standalone PostgreSQL databases) • For capacity based on forks in your configuration, see additional resources
CPUs	2 minimum	• For capacity based on forks in your configuration, see additional resources
Disk: service node	60 GB dedicated hard disk space	• Storage volume should be rated for a minimum baseline of 1500 IOPS.
Database node	**Required**	**Notes**
RAM	16 GB	
CPUs	4	

	Required	Notes
Disk	20 GB dedicated hard disk space	• Minimum dedicated hard disk space is 20 GB • 150 GB+ recommended • Storage volume should be rated for a high baseline IOPS (1500 or more).
Browser	A currently supported version of Mozilla FireFox or Google Chrome	
Database	PostgreSQL version 12	



NOTE

- All automation controller data is stored in the database. Database storage increases with the number of hosts managed, number of jobs run, number of facts stored in the fact cache, and number of tasks in any individual job. For example, a playbook run every hour (24 times a day) across 250, hosts, with 20 tasks will store over 800000 events in the database every week.
- If not enough space is reserved in the database, old job runs and facts will need cleaned on a regular basis. Refer to Management Jobs in the *Automation Controller Administration Guide* for more information

Amazon EC2

- Instance size of m5.large or larger
- An instance size of m4.xlarge or larger if there are more than 100 hosts

Additional notes for Red Hat Ansible Automation Platform requirements

- Actual RAM requirements vary based on how many hosts automation controller will manage simultaneously (which is controlled by the **forks** parameter in the job template or the system **ansible.cfg** file). To avoid possible resource conflicts, Ansible recommends 1 GB of memory per 10 forks + 2 GB reservation for automation controller, see Automation controller Capacity Determination and Job Impact for further details. If **forks** is set to 400, 42 GB of memory is recommended.
- A larger number of hosts can of course be addressed, though if the fork number is less than the total host count, more passes across the hosts are required. These RAM limitations are avoided when using rolling updates or when using the provisioning callback system built into automation controller, where each system requesting configuration enters a queue and is processed as quickly as possible; or in cases where automation controller is producing or deploying images

such as AMIs. All of these are great approaches to managing larger environments. For further questions, please contact Ansible support via the Red Hat Customer portal at https://access.redhat.com/.

- The requirements for systems managed by Ansible Automation Platform are the same as for Ansible. See Getting Started in the Ansible *User Guide*.

Notable PostgreSQL changes

Red Hat Ansible Automation Platform uses PostgreSQL 12.

- PostgreSQL user passwords will now be hashed with SCRAM-SHA-256 secure hashing algorithm before storing in the database.
- You will no longer need to provide a **pg_hashed_password** in your inventory file at the time of installation because PostgreSQL 12 can now store the user's password more securely. If users supply a password in the inventory file for the installer (**pg_password**), that password will be SCRAM-SHA-256 hashed by PostgreSQL as part of the installation process. **DO NOT** use special characters in **pg_password** as it may cause the setup to fail.
- Since automation controller and automation hub are using a Software Collections version of PostgreSQL in 3.8, the **rh-postgresql10** scl must be enabled in order to access the database. Administrators can use the **awx-manage dbshell** command, which will automatically enable the PostgreSQL SCL.
- If you just need to determine if your automation controller instance has access to the database, you can do so with the command, **awx-manage check_db**.

PostgreSQL Configurations

Optionally, you can configure the PostgreSQL database as separate nodes that are not managed by the Red Hat Ansible Automation Platform installer. When the Ansible Automation Platform installer manages the database server, it configures the server with defaults that are generally recommended for most workloads. However, you can adjust these PostgreSQL settings for standalone database server node where **ansible_memtotal_mb** is the total memory size of the database server:

```
max_connections == 1024
shared_buffers == ansible_memtotal_mb*0.3
work_mem == ansible_memtotal_mb*0.03
maintenance_work_mem == ansible_memtotal_mb*0.04
```

Refer to the PostgreSQL documentation for more detail on tuning your PostgreSQL server.

Ansible software requirements

While Red Hat Ansible Automation Platform depends on Ansible Playbooks and requires the installation of the latest stable version of Ansible before installing automation controller, manual installations of Ansible are no longer required.

Upon new installations, automation controller installs the latest release package of Ansible 2.11.

If performing a bundled Ansible Automation Platform installation, the installation program attempts to install Ansible (and its dependencies) from the bundle for you.

If you choose to install Ansible on your own, the Ansible Automation Platform installation program will detect that Ansible has been installed and will not attempt to reinstall it. Note that you must install

Ansible using a package manager like **yum** and that the latest stable version must be installed for Red Hat Ansible Automation Platform to work properly. Ansible version 2.9 is required for |at| versions 3.8 and later.

1.2. CHOOSING AND OBTAINING A RED HAT ANSIBLE AUTOMATION PLATFORM INSTALLER

Choose the Red Hat Ansible Automation Platform installer you need based on your Red Hat Enterprise Linux environment internet connectivity. Review the scenarios below and determine which Red Hat Ansible Automation Platform installer meets your needs.

NOTE

A valid Red Hat customer account is required to access Red Hat Ansible Automation Platform installer downloads on the Red Hat Customer Portal.

Installing with internet access

Choose the Red Hat Ansible Automation Platform installer if your Red Hat Enterprise Linux environment is connected to the internet. Installing with internet access will retrieve the latest required repositories, packages, and dependencies.

1. Navigate to https://access.redhat.com/downloads/content/480
2. Click **Download Now** for the **Ansible Automation Platform <latest-version> Setup**
3. Extract the files:

```
$ tar xvzf ansible-automation-platform-setup-<latest-version>.tar.gz
```

Installing without internet access

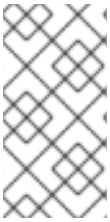
Use the Red Hat Ansible Automation Platform **Bundle** installer if you are unable to access the internet, or would prefer not to install separate components and dependencies from online repositories. Access to Red Hat Enterprise Linux repositories is still needed. All other dependencies are included in the tar archive.

1. Navigate to https://access.redhat.com/downloads/content/480
2. Click **Download Now** for the **Ansible Automation Platform <latest-version> Setup Bundle**
3. Extract the files:

```
$ tar xvzf ansible-automation-platform-setup-bundle-<latest-version>.tar.gz
```

1.3. ATTACHING YOUR RED HAT ANSIBLE AUTOMATION PLATFORM SUBSCRIPTION

You **must** have valid subscriptions attached on all nodes before installing Red Hat Ansible Automation Platform. Attaching your Ansible Automation Platform subscription allows you to access subcription-only resources necessary to proceed with the installation.



NOTE

Attaching a subscription is unnecessary if you have enabled Simple Content Access Mode on your Red Hat account. Once enabled, you will need to register your systems to either Red Hat Subscription Management (RHSM) or Satellite before installing the Ansible Automation Platform. See Simple Content Access Mode for more information.

Procedure

1. Obtain the **pool_id** for your Red Hat Ansible Automation Platform subscription:

```
# subscription-manager list --available --all | grep "Ansible Automation Platform" -B 3 -A 6
```

Example

An example output of the **subsciption-manager list** command. Obtain the **pool_id** as seen in the **Pool ID:** section:

```
Subscription Name: Red Hat Ansible Automation, Premium (5000 Managed Nodes)
  Provides: Red Hat Ansible Engine
  Red Hat Ansible Automation Platform
  SKU: MCT3695
  Contract: ````
  Pool ID: <pool_id>
  Provides Management: No
  Available: 4999
  Suggested: 1
```

2. Attach the subscription:

```
# subscription-manager attach --pool=<pool_id>
```

You have now attached your Red Hat Ansible Automation Platform subscriptions to all nodes.

Verification

- Verify the subscription was successfully attached:

```
# subscription-manager list --consumed
```

Troubleshooting

- If you are unable to locate certain packages that came bundled with the Ansible Automation Platform installer, or if you are seeing a ***Repositories disabled by configuration*** message, try enabling the repository using the command:

```
subscription-manager repos --enable ansible-automation-platform-2.1-for-rhel-8-x86_64-rpms
```

1.4. SUPPORTED INSTALLATION SCENARIOS

Red Hat supports the following installations scenarios for Red Hat Ansible Automation Platform

1.4.1. Standalone automation controller with a database on the same node, or a non-installer managed database

This scenario includes installation of automation controller, including the web frontend, REST API backend, and database on a single machine. It installs PostgreSQL, and configures the automation controller to use that as its database. This is considered the standard automation controller installation scenario.

See Installing automation controller with a database on the same node in *Installing Red Hat Ansible Automation Platform components on a single machine* to get started.

1.4.2. Standalone automation controller with an external managed database

This scenario includes installation of the automation controller server on a single machine and configures communication with a remote PostgreSQL instance as its database. This remote PostgreSQL can be a server you manage, or can be provided by a cloud service such as Amazon RDS.

See Installing automation controller with an external managed database in *Installing Red Hat Ansible Automation Platform components on a single machine* to get started.

1.4.3. Standalone automation hub with a database on the same node, or a non-installer managed database

This scenario includes installation of automation hub, including the web frontend, REST API backend, and database on a single machine. It installs PostgreSQL, and configures the automation hub to use that as its database.

See Installing automation hub with a database on the same node in *Installing Red Hat Ansible Automation Platform components on a single machine* to get started.

1.4.4. Standalone automation hub with an external managed database

This scenario includes installation of the automation hub server on a single machine, and installs a remote PostgreSQL database, managed by the Red Hat Ansible Automation Platform installer.

See Installing automation hub with an external database in *Installing Red Hat Ansible Automation Platform components on a single machine* to get started.

1.4.5. Platform installation with a database on the automation controller node, or non-installer managed database

This scenario includes installation of automation controller and automation hub with a database on the automation controller node, or a non-installer managed database.

See Installing Red Hat Ansible Automation Platform with a database on the automation controller node or non-installer managed database in *Installing Red Hat Ansible Automation Platform* to get started.

1.4.6. Platform installation with an external managed database

This scenario includes installation of automation controller and automation hub and configures communication with a remote PostgreSQL instance as its database. This remote PostgreSQL can be a server you manage, or can be provided by a cloud service such as Amazon RDS.

See Installing Red Hat Ansible Automation Platform with an external managed database in *Installing Red Hat Ansible Automation Platform* to get started.

1.4.7. Multi-machine cluster installation with an external managed database

This scenario includes installation of multiple automation controller nodes and an automation hub instance and configures communication with a remote PostgreSQL instance as its database. This remote PostgreSQL can be a server you manage, or can be provided by a cloud service such as Amazon RDS. In this scenario, all automation controller are active and can execute jobs, and any node can receive HTTP requests.

NOTE

- Running in a cluster setup requires any database that automation controller uses to be external—PostgreSQL must be installed on a machine that is not one of the primary or secondary tower nodes. When in a redundant setup, the remote PostgreSQL version requirements is **PostgreSQL 12**.
  - See Clustering for more information on configuring a clustered setup.
- Provide a reachable IP address for the **[automationhub]** host to ensure users can sync content from Private Automation Hub from a different node.

See Installing a multi-node Red Hat Ansible Automation Platform with an external managed database in *Multi-machine cluster installation* to get started.

CHAPTER 2. INSTALLING RED HAT ANSIBLE AUTOMATION PLATFORM COMPONENTS ON A SINGLE MACHINE

You can install Red Hat Ansible Automation Platform components on a single machine in one of the following supported scenarios.

2.1. INSTALLING AUTOMATION CONTROLLER WITH A DATABASE ON THE SAME NODE

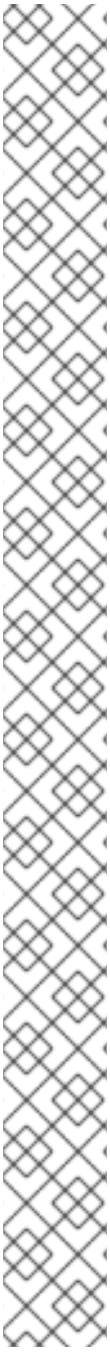
You can use these instructions to install a standalone instance of automation controller with a database on the same node, or a non-installer managed database. This scenario includes installation of automation controller, including the web frontend, REST API backend, and database on a single machine. It installs PostgreSQL, and configures the automation controller to use that as its database. This is considered the standard automation controller installation scenario.

2.1.1. Prerequisites

- You chose and obtained a platform installer from the Red Hat Ansible Automation Platform Product Software.
- You are installing on a machine that meets base system requirements.
- You have created a Red Hat Registry Service Account, following the instructions in the Creating Registry Service Accounts guide.

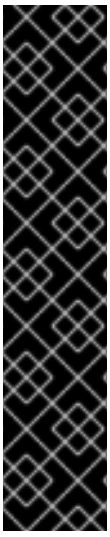
2.1.2. Editing the Red Hat Ansible Automation Platform installer inventory file

You can use the Red Hat Ansible Automation Platform installer inventory file to specify your installation scenario.



NOTE

- Using **external databases**: ensure the database sections of your inventory file are properly setup.
- Add automation hub information in the **[automationhub]** group
- automation hub and automation controller cannot be installed on the same node.
- automation controller will not configure replication or failover for the database that it uses. automation controller should work with any replication you have.
- The database server should be on the same network or in the same data center as the automation controller server for performance reasons.
- A container registry service is required to install the Ansible Automation Platform. Access to a container registry enables you to load automation execution environments onto the Ansible Automation Platform, giving you a consistent and containerized environment for executing Ansible playbooks and roles. By default, the Ansible Automation Platform uses **registry.redhat.io**, which requires a Red Hat registry service account. See the Creating Registry Service Accounts guide to create a registry service account.
- For **upgrading an existing cluster**: When upgrading a cluster, you may decide that you want to also reconfigure your cluster to omit existing instances or instance groups. Omitting the instance or the instance group from the inventory file will not be enough to remove them from the cluster. In addition to omitting instances or instance groups from the inventory file, you must also deprovision instances or instance groups before starting the upgrade. Otherwise, omitted instances or instance groups will continue to communicate with the cluster, which can cause issues with tower services during the upgrade.
- For **clustered installations**: If you are creating a clustered setup, you must replace **localhost** with the hostname or IP address of all instances. All nodes/instances must be able to reach any others using this hostname or address. In other words, you cannot use the **localhost ansible_connection=local** on one of the nodes AND all of the nodes should use the same format for the host names.



IMPORTANT

- Root access to remote machines is required. With Ansible, this can be achieved in different ways:
- **ansible_user=root ansible_ssh_pass="your_password_here"** inventory host or group variables
- **ansible_user=root ansible_ssh_private_key_file="path_to_your_keyfile.pem"** inventory host or group variables
- **ANSIBLE_BECOME_METHOD='sudo' ANSIBLE_BECOME=True ./setup.sh**

For more information on **become** plugins, see Understanding privilege escalation .

Procedure

1. Navigate to the installer
   a. [bundled installer]

```
$ cd ansible-automation-platform-setup-bundle-<latest-version>
```

   b. [online installer]

```
$ cd ansible-automation-platform-setup-<latest-version>
```

2. Open the **inventory** file with a text editor.
3. Edit **inventory** file parameters to specify your installation scenario. Follow the example below.

2.1.3. Example Red Hat Ansible Automation Platform single node inventory file

This example describes how you can populate the inventory file for a single node installation of automation controller.

IMPORTANT

- Do not use special characters for **pg_password**. It may cause the setup to fail.
- Enter your Red Hat Registry Service Account credentials in **registry_username** and **registry_password** to link to the Red Hat container registry.

```
[automationcontroller]
127.0.0.1 ansible_connection=local (1)

[database]

[all:vars]
admin_password='<password>'

pg_host=''
pg_port=''

pg_database='awx'
pg_username='awx'
pg_password='<password>'

registry_url='registry.redhat.io'
registry_username='<registry username>'
registry_password='<registry password>'
```

(1) This should be set as a FQDN/IP.

2.1.4. Additional inventory file variables

You can further configure your Red Hat Ansible Automation Platform installation by including additional variables to the inventory file. These configurations add various optional features for managing your Red Hat Ansible Automation Platform. Add these variables by editing the **inventory** file using a text editor.

Table 2.1. Additional inventory file variables

Variable	Description	Default
GALAXY_ENABLE_API_ACCESS_LOG	When set to **True**, 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	**False**

Example

- To enable the api access log, add the variable to the inventory and flag as **true**:

```
[all:vars:]
admin_password = 'password'

pg_host=''
pg_port=''

GALAXY_ENABLE_API_ACCESS_LOG=true
```

2.1.5. Setup script flags and extra variables

You can also pass flags and extra variables when running the setup script to install automation controller:

Table 2.2. Flags

Argument	Description
-h	Show this help message and exit
-i INVENTORY_FILE	Path to Ansible inventory file (default: **inventory**)
-e EXTRA_VARS	Set additional Ansible variables as key=value or YAML/JSON
-b	Perform a database backup in lieu of installing
-r	Perform a database restore in lieu of installing
-k	Generate **and** dsitribute a SECRET_KEY

Use the **--** separator to add any Ansible arguments you wish to apply. For example: **./setup.sh -i my_awesome_inventory.yml -e matburt_is_country_gold=True — -K**.

NOTE

- When passing **-r** to perform a database restore default restore path is used unless EXTRA_VARS are provided with a non-default path. See the example below that passed an EXTRA_VAR specifying the restore path:

```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

- You can force an online installation by passing **-e bundle_install=false**:

```
$ ./setup.sh -e bundle_install=false
```

Table 2.3. Extra variables

Variable	Description	Default
upgrade_ansible_with_tower	When installing automation controller make sure Ansible is also up to date	**False**
create_preload_data	When installing Tower also create the Demo Org, project, credential, Job Template, etc.	**True**
bundle_install_folder	When installing from a bundle where to put the bundled repos	**var/lib/tower-bundle**
nginx_disable_https	Disable HTTPS traffic through nginx, this is useful if offloading HTTPS to a load balancer	**False**
nginx_disable_hsts	Disable HSTS web-security policy mechanism	**False**
nginx_http_port	Port to configure nginx to listen to for HTTP	**80**
nginx_https_port	Port to configure nginx to listen to for HTTPS	**443**
backup_dir	A temp location to use when backing up	**/var/backups/tower/**
restore_backup_file	Specify an alternative backup file to restore from	None
required_ram	The minimum RAM required to install Tower (should only be changed for test installation)	**3750**

Variable	Description	Default
min_open_fds	The minimum open file descriptions (should only be changed for test installations)	None
ignore_preflight_errors	Ignore preflight checks, useful when installing into a template or other non-system image (overrides **required_ram** and **min_open_fds**)	**False**

Examples

- To upgrade core:

```
./setup.sh -e upgrade_ansible_with_tower=1
```

- To disable https handling at nginx:

```
./setup.sh -e nginx_disable_https=true
```

- To specify a non-default path when restoring from a backup file:

```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

2.1.6. Running the Red Hat Ansible Automation Platform installer setup script

You can run the setup script once you finish updating the **inventory** file with required parameters for installing your Private Automation Hub.

Procedure

1. Run the **setup.sh** script

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

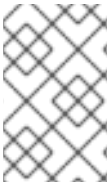
The installation will begin.

2.1.7. Verifying automation controller installation

Once the installation completes, you can verify your automation controller has been installed successfully by logging in with the admin credentials you inserted into the **inventory** file.

Procedure

1. Navigate to the IP address specified for the automation controller node in the **inventory** file.
2. Log in with the Admin credentials you set in the **inventory** file.



NOTE

The automation controller server is accessible from port 80 (**https://<TOWER_SERVER_NAME>/**) but will redirect to port 443 so 443 needs to be available also.

IMPORTANT

If the installation fails and you are a customer who has purchased a valid license for Red Hat Ansible Automation Platform, please contact Ansible via the Red Hat Customer portal at https://access.redhat.com/.

Upon a successful login to automation controller, your installation of Red Hat Ansible Automation Platform 2.1 is now complete.

2.1.7.1. Additional automation controller configuration and resources

See the following resources to explore additional automation controller configurations.

Table 2.4. Resources to configure automation controller

Link	Description
Automation Controller Quick Setup Guide	Set up automation controller and run your first playbook
Automation Controller Administration Guide	Configure automation controller administration through customer scripts, management jobs, etc.
Configuring proxy support for Red Hat Ansible Automation Platform	Set up automation controller with a proxy server
Managing usability analytics and data collection from automation controller	Manage what automation controller information you share with Red Hat
Automation Controller User Guide	Review automation controller functionality in more detail

2.1.8. What's next with Ansible Automation Platform 2.1

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 leveraging the new features of Ansible Automation Platform 2.1:

2.1.8.1. Migrating data to Ansible Automation Platform 2.1

For platform administrators looking to complete an upgrade to the Ansible Automation Platform 2.1, there may be additional steps needed to migrate data to a new instance:

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

Ansible Automation Platform 2.1 moves you away from custom Python virtual environments (venvs) in favor of automation execution environments - containerized images that package the necessary components needed to execute and scale your Ansible automation. This includes Ansible Core, Ansible Content Collections, Python dependencies, Red Hat Enterprise Linux UBI 8, and any additional package dependencies.

If you are looking to migrate your venvs to execution environments, you will (1) need to use the **awx-manage** command to list and export a list of venvs from your original instance, then (2) use **ansible-builder** to create execution environments. For more information, see the Upgrading to Automation Execution Environments guide and the Ansible Builder Guide .

2.1.8.1.2. Migrating to Ansible Engine 2.9 images using Ansible Builder

To migrate Ansible Engine 2.9 images for use with Ansible Automation Platform 2.1, the **ansible-builder** tool automates the process of rebuilding images (including its custom plugins and dependencies) for use with automation execution environments. For more information on using Ansible Builder to build execution environments, see the Ansible Builder Guide .

2.1.8.1.3. Migrating to Ansible Core 2.12

When upgrading to Ansible Core 2.12, you will need to update your playbooks, plugins, or other parts of your Ansible infrastructure in order to be supported by the latest version of Ansible Core. For instructions on updating your Ansible content to be Ansible Core 2.12 compatible, see the Ansible-core 2.12 Porting Guide.

2.1.8.2. Scale up your automation using 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 the Ansible Automation Platform, you will need to 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.

For instructions on how to migrate from isolated nodes to execution nodes, see the **upgrade & migration guide**.

For information about automation mesh and the various ways to design your automation mesh for your environment, see the Red Hat Ansible Automation Platform automation mesh guide .

2.2. INSTALLING AUTOMATION CONTROLLER WITH AN EXTERNAL MANAGED DATABASE

You can use these instructions to install a standalone automation controller server on a single machine configured to communicate with a remote PostgreSQL instance as its database. This remote PostgreSQL can be a server you manage, or can be provided by a cloud service such as Amazon RDS.

2.2.1. Prerequisites

- You chose and obtained a platform installer from the Red Hat Ansible Automation Platform Product Software.
- You are installing on a machine that meets base system requirements.
- You have created a Red Hat Registry Service Account, following the instructions in the Creating Registry Service Accounts guide.

2.2.2. Editing the Red Hat Ansible Automation Platform installer inventory file

You can use the Red Hat Ansible Automation Platform installer inventory file to specify your installation scenario.

NOTE

- Using **external databases**: ensure the database sections of your inventory file are properly setup.
- Add automation hub information in the **[automationhub]** group
- automation hub and automation controller cannot be installed on the same node.
- automation controller will not configure replication or failover for the database that it uses. automation controller should work with any replication you have.
- The database server should be on the same network or in the same data center as the automation controller server for performance reasons.
- A container registry service is required to install the Ansible Automation Platform. Access to a container registry enables you to load automation execution environments onto the Ansible Automation Platform, giving you a consistent and containerized environment for executing Ansible playbooks and roles. By default, the Ansible Automation Platform uses **registry.redhat.io**, which requires a Red Hat registry service account. See the Creating Registry Service Accounts guide to create a registry service account.
- For **upgrading an existing cluster**: When upgrading a cluster, you may decide that you want to also reconfigure your cluster to omit existing instances or instance groups. Omitting the instance or the instance group from the inventory file will not be enough to remove them from the cluster. In addition to omitting instances or instance groups from the inventory file, you must also deprovision instances or instance groups before starting the upgrade. Otherwise, omitted instances or instance groups will continue to communicate with the cluster, which can cause issues with tower services during the upgrade.
- For **clustered installations**: If you are creating a clustered setup, you must replace **localhost** with the hostname or IP address of all instances. All nodes/instances must be able to reach any others using this hostname or address. In other words, you cannot use the **localhost ansible_connection=local** on one of the nodes AND all of the nodes should use the same format for the host names.

IMPORTANT

- Root access to remote machines is required. With Ansible, this can be achieved in different ways:
- **ansible_user=root ansible_ssh_pass="your_password_here"** inventory host or group variables
- **ansible_user=root ansible_ssh_private_key_file="path_to_your_keyfile.pem"** inventory host or group variables
- **ANSIBLE_BECOME_METHOD='sudo' ANSIBLE_BECOME=True ./setup.sh**

For more information on **become** plugins, see Understanding privilege escalation .

Procedure

1. Navigate to the installer
   a. [bundled installer]

```
$ cd ansible-automation-platform-setup-bundle-<latest-version>
```

   b. [online installer]

```
$ cd ansible-automation-platform-setup-<latest-version>
```

2. Open the **inventory** file with a text editor.
3. Edit **inventory** file parameters to specify your installation scenario. Follow the example below.

2.2.3. Example inventory file for a standalone automation controller with an external managed database

This example describes how you can populate the inventory file to deploy an installation of automation controller with an external database.

IMPORTANT

- Do not use special characters for **pg_password**. It may cause the setup to fail.
- Enter your Red Hat Registry Service Account credentials in **registry_username** and **registry_password** to link to the Red Hat container registry.

```
[automationcontroller]
127.0.0.1 ansible_connection=local


[database]
database.example.com

[all:vars]
admin_password='<password>'
```

```
pg_password='<password>'

pg_host='database.example.com'
pg_port='5432'

pg_database='awx'
pg_username='awx'

registry_url='registry.redhat.io'
registry_username='<registry username>'
registry_password='<registry password>'
```

2.2.4. Additional inventory file variables

You can further configure your Red Hat Ansible Automation Platform installation by including additional variables to the inventory file. These configurations add various optional features for managing your Red Hat Ansible Automation Platform. Add these variables by editing the **inventory** file using a text editor.

Table 2.5. Additional inventory file variables

Variable	Description	Default
GALAXY_ENABLE_API_ACCESS_LOG	When set to **True**, 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	**False**

Example

- To enable the api access log, add the variable to the inventory and flag as **true**:

```
[all:vars:]
admin_password = 'password'

pg_host=''
pg_port=''

GALAXY_ENABLE_API_ACCESS_LOG=true
```

2.2.5. Setup script flags and extra variables

You can also pass flags and extra variables when running the setup script to install automation controller:

Table 2.6. Flags

Argument	Description
-h	Show this help message and exit

Argument	Description
-i INVENTORY_FILE	Path to Ansible inventory file (default: **inventory**)
-e EXTRA_VARS	Set additional Ansible variables as key=value or YAML/JSON
-b	Perform a database backup in lieu of installing
-r	Perform a database restore in lieu of installing
-k	Generate **and** dsitribute a SECRET_KEY

Use the **--** separator to add any Ansible arguments you wish to apply. For example: **./setup.sh -i my_awesome_inventory.yml -e matburt_is_country_gold=True — -K**.

NOTE

- When passing **-r** to perform a database restore default restore path is used unless EXTRA_VARS are provided with a non-default path. See the example below that passed an EXTRA_VAR specifying the restore path:

```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

- You can force an online installation by passing **-e bundle_install=false**:

```
$ ./setup.sh -e bundle_install=false
```

Table 2.7. Extra variables

Variable	Description	Default
upgrade_ansible_with_tower	When installing automation controller make sure Ansible is also up to date	**False**
create_preload_data	When installing Tower also create the Demo Org, project, credential, Job Template, etc.	**True**
bundle_install_folder	When installing from a bundle where to put the bundled repos	**var/lib/tower-bundle**
nginx_disable_https	Disable HTTPS traffic through nginx, this is useful if offloading HTTPS to a load balancer	**False**

Variable	Description	Default
nginx_disable_hsts	Disable HSTS web-security policy mechanism	**False**
nginx_http_port	Port to configure nginx to listen to for HTTP	**80**
nginx_https_port	Port to configure nginx to listen to for HTTPS	**443**
backup_dir	A temp location to use when backing up	**/var/backups/tower/**
restore_backup_file	Specify an alternative backup file to restore from	None
required_ram	The minimum RAM required to install Tower (should only be changed for test installation)	**3750**
min_open_fds	The minimum open file descriptions (should only be changed for test installations)	None
ignore_preflight_errors	Ignore preflight checks, useful when installing into a template or other non-system image (overrides **required_ram** and **min_open_fds**)	**False**

Examples

- To upgrade core:

```
./setup.sh -e upgrade_ansible_with_tower=1
```

- To disable https handling at nginx:

```
./setup.sh -e nginx_disable_https=true
```

- To specify a non-default path when restoring from a backup file:

```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

2.2.6. Running the Red Hat Ansible Automation Platform installer setup script

You can run the setup script once you finish updating the **inventory** file with required parameters for installing your Private Automation Hub.

Procedure

1. Run the **setup.sh** script

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

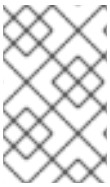
The installation will begin.

2.2.7. Verifying automation controller installation

Once the installation completes, you can verify your automation controller has been installed successfully by logging in with the admin credentials you inserted into the **inventory** file.

Procedure

1. Navigate to the IP address specified for the automation controller node in the **inventory** file.
2. Log in with the Admin credentials you set in the **inventory** file.



NOTE

The automation controller server is accessible from port 80 (**https://<TOWER_SERVER_NAME>/**) but will redirect to port 443 so 443 needs to be available also.

IMPORTANT

If the installation fails and you are a customer who has purchased a valid license for Red Hat Ansible Automation Platform, please contact Ansible via the Red Hat Customer portal at https://access.redhat.com/.

Upon a successful login to automation controller, your installation of Red Hat Ansible Automation Platform 2.1 is now complete.

2.2.7.1. Additional automation controller configuration and resources

See the following resources to explore additional automation controller configurations.

Table 2.8. Resources to configure automation controller

Link	Description
Automation Controller Quick Setup Guide	Set up automation controller and run your first playbook
Automation Controller Administration Guide	Configure automation controller administration through customer scripts, management jobs, etc.
Configuring proxy support for Red Hat Ansible Automation Platform	Set up automation controller with a proxy server

Link	Description
Managing usability analytics and data collection from automation controller	Manage what automation controller information you share with Red Hat
Automation Controller User Guide	Review automation controller functionality in more detail

2.2.8. What's next with Ansible Automation Platform 2.1

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 leveraging the new features of Ansible Automation Platform 2.1:

2.2.8.1. Migrating data to Ansible Automation Platform 2.1

For platform administrators looking to complete an upgrade to the Ansible Automation Platform 2.1, there may be additional steps needed to migrate data to a new instance:

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

Ansible Automation Platform 2.1 moves you away from custom Python virtual environments (venvs) in favor of automation execution environments - containerized images that package the necessary components needed to execute and scale your Ansible automation. This includes Ansible Core, Ansible Content Collections, Python dependencies, Red Hat Enterprise Linux UBI 8, and any additional package dependencies.

If you are looking to migrate your venvs to execution environments, you will (1) need to use the **awx-manage** command to list and export a list of venvs from your original instance, then (2) use **ansible-builder** to create execution environments. For more information, see the Upgrading to Automation Execution Environments guide and the Ansible Builder Guide.

2.2.8.1.2. Migrating to Ansible Engine 2.9 images using Ansible Builder

To migrate Ansible Engine 2.9 images for use with Ansible Automation Platform 2.1, the **ansible-builder** tool automates the process of rebuilding images (including its custom plugins and dependencies) for use with automation execution environments. For more information on using Ansible Builder to build execution environments, see the Ansible Builder Guide.

2.2.8.1.3. Migrating to Ansible Core 2.12

When upgrading to Ansible Core 2.12, you will need to update your playbooks, plugins, or other parts of your Ansible infrastructure in order to be supported by the latest version of Ansible Core. For instructions on updating your Ansible content to be Ansible Core 2.12 compatible, see the Ansible-core 2.12 Porting Guide.

2.2.8.2. Scale up your automation using 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 the Ansible Automation Platform, you will need to 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.

For instructions on how to migrate from isolated nodes to execution nodes, see the **upgrade & migration guide**.

For information about automation mesh and the various ways to design your automation mesh for your environment, see the Red Hat Ansible Automation Platform automation mesh guide .

2.3. INSTALLING AUTOMATION HUB WITH A DATABASE ON THE SAME NODE

You can use these instructions to install a standalone instance of automation hub with a database on the same node, or a non-installer managed database.

2.3.1. Prerequisites

- You chose and obtained a platform installer from the Red Hat Ansible Automation Platform Product Software.
- You are installing on a machine that meets base system requirements.
- You have created a Red Hat Registry Service Account, following the instructions in the Creating Registry Service Accounts guide.

2.3.2. Red Hat Ansible Automation Platform installation settings

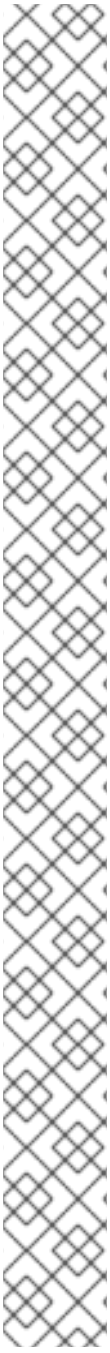
You can use the following settings when installing automation hub:

- **automationhub_importer_settings**: Dictionary of settings/configuration to pass to **galaxy-importer**. It will end up in **/etc/galaxy-importer/galaxy-importer.cfg**
- **automationhub_require_content_approval**: Whether or not automation hub enforces the approval mechanism before collections are made available
- **automationhub_disable_https**: Whether or not automation hub should be deployed with TLS enabled
- **automationhub_disable_hsts**: Whether or not automation hub should be deployed with the HTTP Strict Transport Security (HSTS) web-security policy mechanism enabled
- **automationhub_ssl_validate_certs**: Whether or not automation hub should validate certificate when requesting itself (default = False) because by default, Platform deploys with self-signed certificates
- **automationhub_ssl_cert**: Same as **web_server_ssl_cert** but for automation hub UI and API
- **automationhub_ssl_key**: Same as **web_server_ssl_key** but for automation hub UI and API
- **automationhub_backup_collections**: automation hub provides artifacts in **/var/lib/pulp**. By default, this is set to **true** so automation controller automatically backs up the artifacts by default. If a partition (e.g., LVM, NFS, CephFS, etc.) was mounted there, an enterprise

organization would ensure it is always backed up. If this is the case, you can set **automationhub_backup_collections = false** and the backup/restore process will not have to backup/restore **/var/lib/pulp**.

2.3.3. Editing the Red Hat Ansible Automation Platform installer inventory file

You can use the Red Hat Ansible Automation Platform installer inventory file to specify your installation scenario.



NOTE

- Using **external databases**: ensure the database sections of your inventory file are properly setup.
- Add automation hub information in the **[automationhub]** group
- automation hub and automation controller cannot be installed on the same node.
- automation controller will not configure replication or failover for the database that it uses. automation controller should work with any replication you have.
- The database server should be on the same network or in the same data center as the automation controller server for performance reasons.
- A container registry service is required to install the Ansible Automation Platform. Access to a container registry enables you to load automation execution environments onto the Ansible Automation Platform, giving you a consistent and containerized environment for executing Ansible playbooks and roles. By default, the Ansible Automation Platform uses **registry.redhat.io**, which requires a Red Hat registry service account. See the Creating Registry Service Accounts guide to create a registry service account.
- For **upgrading an existing cluster**: When upgrading a cluster, you may decide that you want to also reconfigure your cluster to omit existing instances or instance groups. Omitting the instance or the instance group from the inventory file will not be enough to remove them from the cluster. In addition to omitting instances or instance groups from the inventory file, you must also deprovision instances or instance groups before starting the upgrade. Otherwise, omitted instances or instance groups will continue to communicate with the cluster, which can cause issues with tower services during the upgrade.
- For **clustered installations**: If you are creating a clustered setup, you must replace **localhost** with the hostname or IP address of all instances. All nodes/instances must be able to reach any others using this hostname or address. In other words, you cannot use the **localhost ansible_connection=local** on one of the nodes AND all of the nodes should use the same format for the host names.

IMPORTANT

- Root access to remote machines is required. With Ansible, this can be achieved in different ways:
- **ansible_user=root ansible_ssh_pass="your_password_here"** inventory host or group variables
- **ansible_user=root ansible_ssh_private_key_file="path_to_your_keyfile.pem"** inventory host or group variables
- **ANSIBLE_BECOME_METHOD='sudo' ANSIBLE_BECOME=True ./setup.sh**

For more information on **become** plugins, see Understanding privilege escalation .

Procedure

1. Navigate to the installer
   a. [bundled installer]

```
$ cd ansible-automation-platform-setup-bundle-<latest-version>
```

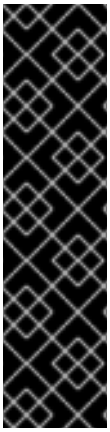
   b. [online installer]

```
$ cd ansible-automation-platform-setup-<latest-version>
```

2. Open the **inventory** file with a text editor.
3. Edit **inventory** file parameters to specify your installation scenario. Follow the example below.

2.3.4. Example standalone automation hub inventory file

This example describes how you can populate the inventory file to deploy a standalone instance of automation hub.



IMPORTANT

- For Red Hat Ansible Automation Platform or automation hub: Add an automation hub host in the **[automationhub]** group. You cannot install automation controller and automation hub on the same node.
- Provide a reachable IP address or fully qualified domain name (FDQN) for the **[automationhub]** host to ensure users can sync and install content from automation hub from a different node. Do not use 'localhost'.
- Enter your Red Hat Registry Service Account credentials in **registry_username** and **registry_password** to link to the Red Hat container registry.

```
[automationcontroller]


[automationhub]
```

```
<FQDN> ansible_connection=local

[all:vars]
registry_url='registry.redhat.io'
registry_username='<registry username>'
registry_password='<registry password>'

automationhub_admin_password= <PASSWORD>

automationhub_pg_host=''
automationhub_pg_port=''

automationhub_pg_database='automationhub'
automationhub_pg_username='automationhub'
automationhub_pg_password=<PASSWORD>
automationhub_pg_sslmode='prefer'

# The default install will deploy a TLS enabled Automation Hub.
# If for some reason this is not the behavior wanted one can
# disable TLS enabled deployment.
#
# automationhub_disable_https = False
# The default install will generate self-signed certificates for the Automation
# Hub service. If you are providing valid certificate via automationhub_ssl_cert
# and automationhub_ssl_key, one should toggle that value to True.
#
# automationhub_ssl_validate_certs = False
# SSL-related variables
# If set, this will install a custom CA certificate to the system trust store.
# custom_ca_cert=/path/to/ca.crt
# Certificate and key to install in Automation Hub node
# automationhub_ssl_cert=/path/to/automationhub.cert
# automationhub_ssl_key=/path/to/automationhub.key
```

2.3.5. Additional inventory file variables

You can further configure your Red Hat Ansible Automation Platform installation by including additional variables to the inventory file. These configurations add various optional features for managing your Red Hat Ansible Automation Platform. Add these variables by editing the **inventory** file using a text editor.

Table 2.9. Additional inventory file variables

Variable	Description	Default
GALAXY_ENABLE_API_ACCESS_LOG	When set to **True**, 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	**False**

Example

- To enable the api access log, add the variable to the inventory and flag as **true**:

```
[all:vars:]
admin_password = 'password'

pg_host=''
pg_port=''

GALAXY_ENABLE_API_ACCESS_LOG=true
```

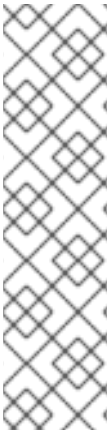
2.3.6. Setup script flags and extra variables

You can also pass flags and extra variables when running the setup script to install automation controller:

Table 2.10. Flags

Argument	Description
-h	Show this help message and exit
-i INVENTORY_FILE	Path to Ansible inventory file (default: **inventory**)
-e EXTRA_VARS	Set additional Ansible variables as key=value or YAML/JSON
-b	Perform a database backup in lieu of installing
-r	Perform a database restore in lieu of installing
-k	Generate **and** dsitribute a SECRET_KEY

Use the **--** separator to add any Ansible arguments you wish to apply. For example: **./setup.sh -i my_awesome_inventory.yml -e matburt_is_country_gold=True — -K**.



NOTE

- When passing **-r** to perform a database restore default restore path is used unless EXTRA_VARS are provided with a non-default path. See the example below that passed an EXTRA_VAR specifying the restore path:

```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

- You can force an online installation by passing **-e bundle_install=false**:

```
$ ./setup.sh -e bundle_install=false
```

Table 2.11. Extra variables

Variable	Description	Default
upgrade_ansible_with_tower	When installing automation controller make sure Ansible is also up to date	**False**
create_preload_data	When installing Tower also create the Demo Org, project, credential, Job Template, etc.	**True**
bundle_install_folder	When installing from a bundle where to put the bundled repos	**var/lib/tower-bundle**
nginx_disable_https	Disable HTTPS traffic through nginx, this is useful if offloading HTTPS to a load balancer	**False**
nginx_disable_hsts	Disable HSTS web-security policy mechanism	**False**
nginx_http_port	Port to configure nginx to listen to for HTTP	**80**
nginx_https_port	Port to configure nginx to listen to for HTTPS	**443**
backup_dir	A temp location to use when backing up	**/var/backups/tower/**
restore_backup_file	Specify an alternative backup file to restore from	None
required_ram	The minimum RAM required to install Tower (should only be changed for test installation)	**3750**
min_open_fds	The minimum open file descriptions (should only be changed for test installations)	None
ignore_preflight_errors	Ignore preflight checks, useful when installing into a template or other non-system image (overrides **required_ram** and **min_open_fds**)	**False**

Examples

- To upgrade core:

```
./setup.sh -e upgrade_ansible_with_tower=1
```

- To disable https handling at nginx:

```
./setup.sh -e nginx_disable_https=true
```

- To specify a non-default path when restoring from a backup file:

```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

2.3.7. Running the Red Hat Ansible Automation Platform installer setup script

You can run the setup script once you finish updating the **inventory** file with required parameters for installing your Private Automation Hub.

Procedure

1. Run the **setup.sh** script

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

The installation will begin.

2.3.8. Verifying automation hub installation

Once the installation completes, you can verify your automation hub has been installed successfully by logging in with the admin credentials you inserted into the **inventory** file.

Procedure

1. Navigate to the IP address specified for the automation hub node in the **inventory** file.
2. Log in with the Admin credentials you set in the **inventory** file.

IMPORTANT

If the installation fails and you are a customer who has purchased a valid license for Red Hat Ansible Automation Platform, please contact Ansible via the Red Hat Customer portal at https://access.redhat.com/.

Upon a successful login to automation hub, your installation of Red Hat Ansible Automation Platform 2.1 is now complete.

2.3.8.1. Additional automation hub configuration and resources

See the following resources to explore additional automation hub configurations.

Table 2.12. Resources to configure automation controller

Link	Description
Managing user access in private automation hub	Configure user access for automation hub
Managing Red Hat Certified and Ansible Galaxy collections in automation hub	Add content to your automation hub
Publishing proprietary content collections in automation hub	Publish internally developed collections on your automation hub

2.3.9. What's next with Ansible Automation Platform 2.1

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 leveraging the new features of Ansible Automation Platform 2.1:

2.3.9.1. Migrating data to Ansible Automation Platform 2.1

For platform administrators looking to complete an upgrade to the Ansible Automation Platform 2.1, there may be additional steps needed to migrate data to a new instance:

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

Ansible Automation Platform 2.1 moves you away from custom Python virtual environments (venvs) in favor of automation execution environments - containerized images that package the necessary components needed to execute and scale your Ansible automation. This includes Ansible Core, Ansible Content Collections, Python dependencies, Red Hat Enterprise Linux UBI 8, and any additional package dependencies.

If you are looking to migrate your venvs to execution environments, you will (1) need to use the **awx-manage** command to list and export a list of venvs from your original instance, then (2) use **ansible-builder** to create execution environments. For more information, see the Upgrading to Automation Execution Environments guide and the Ansible Builder Guide.

2.3.9.1.2. Migrating to Ansible Engine 2.9 images using Ansible Builder

To migrate Ansible Engine 2.9 images for use with Ansible Automation Platform 2.1, the **ansible-builder** tool automates the process of rebuilding images (including its custom plugins and dependencies) for use with automation execution environments. For more information on using Ansible Builder to build execution environments, see the Ansible Builder Guide.

2.3.9.1.3. Migrating to Ansible Core 2.12

When upgrading to Ansible Core 2.12, you will need to update your playbooks, plugins, or other parts of your Ansible infrastructure in order to be supported by the latest version of Ansible Core. For instructions on updating your Ansible content to be Ansible Core 2.12 compatible, see the Ansible-core 2.12 Porting Guide.

2.3.9.2. Scale up your automation using 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 the Ansible Automation Platform, you will need to 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.

For instructions on how to migrate from isolated nodes to execution nodes, see the **upgrade & migration guide**.

For information about automation mesh and the various ways to design your automation mesh for your environment, see the Red Hat Ansible Automation Platform automation mesh guide .

2.4. INSTALLING AUTOMATION HUB WITH AN EXTERNAL DATABASE

You can use these instructions to install a standalone instance of automation hub with an external managed database. This installs the automation hub server on a single machine and installs a remote PostgreSQL database using the Ansible Automation Platform installer.

2.4.1. Prerequisites

- You chose and obtained a platform installer from the Red Hat Ansible Automation Platform Product Software.
- You are installing on a machine that meets base system requirements.
- You have created a Red Hat Registry Service Account, following the instructions in the Creating Registry Service Accounts guide.

2.4.2. Red Hat Ansible Automation Platform installation settings

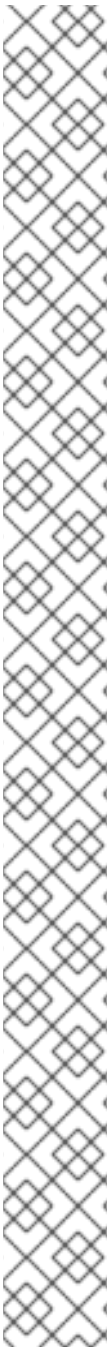
You can use the following settings when installing automation hub:

- **automationhub_importer_settings**: Dictionary of settings/configuration to pass to **galaxy-importer**. It will end up in **/etc/galaxy-importer/galaxy-importer.cfg**
- **automationhub_require_content_approval**: Whether or not automation hub enforces the approval mechanism before collections are made available
- **automationhub_disable_https**: Whether or not automation hub should be deployed with TLS enabled
- **automationhub_disable_hsts**: Whether or not automation hub should be deployed with the HTTP Strict Transport Security (HSTS) web-security policy mechanism enabled
- **automationhub_ssl_validate_certs**: Whether or not automation hub should validate certificate when requesting itself (default = False) because by default, Platform deploys with self-signed certificates
- **automationhub_ssl_cert**: Same as **web_server_ssl_cert** but for automation hub UI and API
- **automationhub_ssl_key**: Same as **web_server_ssl_key** but for automation hub UI and API

- **automationhub_backup_collections**: automation hub provides artifacts in **/var/lib/pulp**. By default, this is set to **true** so automation controller automatically backs up the artifacts by default. If a partition (e.g., LVM, NFS, CephFS, etc.) was mounted there, an enterprise organization would ensure it is always backed up. If this is the case, you can set **automationhub_backup_collections = false** and the backup/restore process will not have to backup/restore **/var/lib/pulp**.

2.4.3. Editing the Red Hat Ansible Automation Platform installer inventory file

You can use the Red Hat Ansible Automation Platform installer inventory file to specify your installation scenario.



NOTE

- Using **external databases**: ensure the database sections of your inventory file are properly setup.
- Add automation hub information in the **[automationhub]** group
- automation hub and automation controller cannot be installed on the same node.
- automation controller will not configure replication or failover for the database that it uses. automation controller should work with any replication you have.
- The database server should be on the same network or in the same data center as the automation controller server for performance reasons.
- A container registry service is required to install the Ansible Automation Platform. Access to a container registry enables you to load automation execution environments onto the Ansible Automation Platform, giving you a consistent and containerized environment for executing Ansible playbooks and roles. By default, the Ansible Automation Platform uses **registry.redhat.io**, which requires a Red Hat registry service account. See the Creating Registry Service Accounts guide to create a registry service account.
- For **upgrading an existing cluster**: When upgrading a cluster, you may decide that you want to also reconfigure your cluster to omit existing instances or instance groups. Omitting the instance or the instance group from the inventory file will not be enough to remove them from the cluster. In addition to omitting instances or instance groups from the inventory file, you must also deprovision instances or instance groups before starting the upgrade. Otherwise, omitted instances or instance groups will continue to communicate with the cluster, which can cause issues with tower services during the upgrade.
- For **clustered installations**: If you are creating a clustered setup, you must replace **localhost** with the hostname or IP address of all instances. All nodes/instances must be able to reach any others using this hostname or address. In other words, you cannot use the **localhost ansible_connection=local** on one of the nodes AND all of the nodes should use the same format for the host names.

IMPORTANT

- Root access to remote machines is required. With Ansible, this can be achieved in different ways:
- **ansible_user=root ansible_ssh_pass="your_password_here"** inventory host or group variables
- **ansible_user=root ansible_ssh_private_key_file="path_to_your_keyfile.pem"** inventory host or group variables
- **ANSIBLE_BECOME_METHOD='sudo' ANSIBLE_BECOME=True ./setup.sh**

For more information on **become** plugins, see Understanding privilege escalation .

Procedure

1. Navigate to the installer
   a. [bundled installer]

```
$ cd ansible-automation-platform-setup-bundle-<latest-version>
```

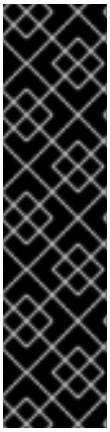
   b. [online installer]

```
$ cd ansible-automation-platform-setup-<latest-version>
```

2. Open the **inventory** file with a text editor.
3. Edit **inventory** file parameters to specify your installation scenario. Follow the example below.

2.4.4. Example standalone automation hub inventory file

This example describes how you can populate the inventory file to deploy a standalone instance of automation hub.



IMPORTANT

- For Red Hat Ansible Automation Platform or automation hub: Add an automation hub host in the `[automationhub] group. You cannot install automation controller and automation hub on the same node.
- Provide a reachable IP address or fully qualified domain name (FDQN) for the **[automationhub]** host to ensure users can sync and install content from automation hub from a different node. Do not use 'localhost'.
- Enter your Red Hat Registry Service Account credentials in **registry_username** and **registry_password** to link to the Red Hat container registry.

```
[automationcontroller]


[automationhub]
```

```
<FQDN> ansible_connection=local

[database]
host2

[all:vars]
registry_url='registry.redhat.io'
registry_username='<registry username>'
registry_password='<registry password>'

automationhub_admin_password= <PASSWORD>

automationhub_pg_host=''
automationhub_pg_port=''

automationhub_pg_database='automationhub'
automationhub_pg_username='automationhub'
automationhub_pg_password=<PASSWORD>
automationhub_pg_sslmode='prefer'

# The default install will deploy a TLS enabled Automation Hub.
# If for some reason this is not the behavior wanted one can
# disable TLS enabled deployment.
#
# automationhub_disable_https = False
# The default install will generate self-signed certificates for the Automation
# Hub service. If you are providing valid certificate via automationhub_ssl_cert
# and automationhub_ssl_key, one should toggle that value to True.
#
# automationhub_ssl_validate_certs = False
# SSL-related variables
# If set, this will install a custom CA certificate to the system trust store.
# custom_ca_cert=/path/to/ca.crt
# Certificate and key to install in Automation Hub node
# automationhub_ssl_cert=/path/to/automationhub.cert
# automationhub_ssl_key=/path/to/automationhub.key
```

2.4.5. Additional inventory file variables

You can further configure your Red Hat Ansible Automation Platform installation by including additional variables to the inventory file. These configurations add various optional features for managing your Red Hat Ansible Automation Platform. Add these variables by editing the **inventory** file using a text editor.

Table 2.13. Additional inventory file variables

Variable	Description	Default
GALAXY_ENABLE_API_ACCESS_LOG	When set to **True**, 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	**False**

Example

- To enable the api access log, add the variable to the inventory and flag as **true**:

```
[all:vars:]
admin_password = 'password'

pg_host=''
pg_port=''

GALAXY_ENABLE_API_ACCESS_LOG=true
```

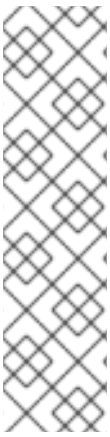
2.4.6. Setup script flags and extra variables

You can also pass flags and extra variables when running the setup script to install automation controller:

Table 2.14. Flags

Argument	Description
-h	Show this help message and exit
-i INVENTORY_FILE	Path to Ansible inventory file (default: **inventory**)
-e EXTRA_VARS	Set additional Ansible variables as key=value or YAML/JSON
-b	Perform a database backup in lieu of installing
-r	Perform a database restore in lieu of installing
-k	Generate **and** dsitribute a SECRET_KEY

Use the **--** separator to add any Ansible arguments you wish to apply. For example: **./setup.sh -i my_awesome_inventory.yml -e matburt_is_country_gold=True — -K**.



NOTE

- When passing **-r** to perform a database restore default restore path is used unless EXTRA_VARS are provided with a non-default path. See the example below that passed an EXTRA_VAR specifying the restore path:

```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

- You can force an online installation by passing **-e bundle_install=false**:

```
$ ./setup.sh -e bundle_install=false
```

Table 2.15. Extra variables

Variable	Description	Default
upgrade_ansible_with_tower	When installing automation controller make sure Ansible is also up to date	**False**
create_preload_data	When installing Tower also create the Demo Org, project, credential, Job Template, etc.	**True**
bundle_install_folder	When installing from a bundle where to put the bundled repos	**var/lib/tower-bundle**
nginx_disable_https	Disable HTTPS traffic through nginx, this is useful if offloading HTTPS to a load balancer	**False**
nginx_disable_hsts	Disable HSTS web-security policy mechanism	**False**
nginx_http_port	Port to configure nginx to listen to for HTTP	**80**
nginx_https_port	Port to configure nginx to listen to for HTTPS	**443**
backup_dir	A temp location to use when backing up	**/var/backups/tower/**
restore_backup_file	Specify an alternative backup file to restore from	None
required_ram	The minimum RAM required to install Tower (should only be changed for test installation)	**3750**
min_open_fds	The minimum open file descriptions (should only be changed for test installations)	None
ignore_preflight_errors	Ignore preflight checks, useful when installing into a template or other non-system image (overrides **required_ram** and **min_open_fds**)	**False**

Examples

- To upgrade core:

```
./setup.sh -e upgrade_ansible_with_tower=1
```

- To disable https handling at nginx:

```
./setup.sh -e nginx_disable_https=true
```

- To specify a non-default path when restoring from a backup file:

```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

2.4.7. Running the Red Hat Ansible Automation Platform installer setup script

You can run the setup script once you finish updating the **inventory** file with required parameters for installing your Private Automation Hub.

Procedure

1. Run the **setup.sh** script

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

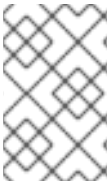
The installation will begin.

2.4.8. Verifying automation controller installation

Once the installation completes, you can verify your automation controller has been installed successfully by logging in with the admin credentials you inserted into the **inventory** file.

Procedure

1. Navigate to the IP address specified for the automation controller node in the **inventory** file.
2. Log in with the Admin credentials you set in the **inventory** file.



NOTE

The automation controller server is accessible from port 80 (**https://<TOWER_SERVER_NAME>/**) but will redirect to port 443 so 443 needs to be available also.

IMPORTANT

If the installation fails and you are a customer who has purchased a valid license for Red Hat Ansible Automation Platform, please contact Ansible via the Red Hat Customer portal at https://access.redhat.com/.

Upon a successful login to automation controller, your installation of Red Hat Ansible Automation Platform 2.1 is now complete.

2.4.8.1. Additional automation hub configuration and resources

See the following resources to explore additional automation hub configurations.

Table 2.16. Resources to configure automation controller

Link	Description
Managing user access in private automation hub	Configure user access for automation hub
Managing Red Hat Certified and Ansible Galaxy collections in automation hub	Add content to your automation hub
Publishing proprietary content collections in automation hub	Publish internally developed collections on your automation hub

2.4.9. What's next with Ansible Automation Platform 2.1

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 leveraging the new features of Ansible Automation Platform 2.1:

2.4.9.1. Migrating data to Ansible Automation Platform 2.1

For platform administrators looking to complete an upgrade to the Ansible Automation Platform 2.1, there may be additional steps needed to migrate data to a new instance:

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

Ansible Automation Platform 2.1 moves you away from custom Python virtual environments (venvs) in favor of automation execution environments - containerized images that package the necessary components needed to execute and scale your Ansible automation. This includes Ansible Core, Ansible Content Collections, Python dependencies, Red Hat Enterprise Linux UBI 8, and any additional package dependencies.

If you are looking to migrate your venvs to execution environments, you will (1) need to use the **awx-manage** command to list and export a list of venvs from your original instance, then (2) use **ansible-builder** to create execution environments. For more information, see the Upgrading to Automation Execution Environments guide and the Ansible Builder Guide.

2.4.9.1.2. Migrating to Ansible Engine 2.9 images using Ansible Builder

To migrate Ansible Engine 2.9 images for use with Ansible Automation Platform 2.1, the **ansible-builder** tool automates the process of rebuilding images (including its custom plugins and dependencies) for use with automation execution environments. For more information on using Ansible Builder to build execution environments, see the Ansible Builder Guide.

2.4.9.1.3. Migrating to Ansible Core 2.12

When upgrading to Ansible Core 2.12, you will need to update your playbooks, plugins, or other parts of your Ansible infrastructure in order to be supported by the latest version of Ansible Core. For instructions on updating your Ansible content to be Ansible Core 2.12 compatible, see the Ansible-core 2.12 Porting Guide.

2.4.9.2. Scale up your automation using 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.

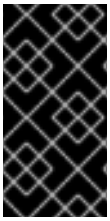
When upgrading from version 1.x to the latest version of the Ansible Automation Platform, you will need to migrate the data from your legacy isolated nodes into execution nodes necessary for automation mesh. You can implement automation mesh by planning out a network of hybrid and control nodes, then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

For instructions on how to migrate from isolated nodes to execution nodes, see the **upgrade & migration guide**.

For information about automation mesh and the various ways to design your automation mesh for your environment, see the Red Hat Ansible Automation Platform automation mesh guide .

CHAPTER 3. INSTALLING RED HAT ANSIBLE AUTOMATION PLATFORM

Red Hat Ansible Automation Platform installation involves deploying automation controller and automation hub.



IMPORTANT

The Ansible Automation Platform installer allows you to deploy **only one** automation hub per inventory. You can use the Ansible Automation Platform installer for a standalone instance of automation hub and run the installer any number of times with any number of different inventories to deploy multiple automation hub nodes.

This installation option includes two supported scenarios:

3.1. INSTALLING RED HAT ANSIBLE AUTOMATION PLATFORM WITH A DATABASE ON THE AUTOMATION CONTROLLER NODE OR NON-INSTALLER MANAGED DATABASE

You can use these instructions to install Red Hat Ansible Automation Platform (both automation controller and automation hub) with a database on the automation controller node, or a non-installer managed database.

3.1.1. Prerequisites

- You chose and obtained a platform installer from the Red Hat Ansible Automation Platform Product Software.
- You are installing on a machine that meets base system requirements.
- You have created a Red Hat Registry Service Account, following the instructions in the Creating Registry Service Accounts guide.

3.1.2. Red Hat Ansible Automation Platform installation settings

You can use the following settings when installing automation hub:

- **automationhub_importer_settings**: Dictionary of settings/configuration to pass to **galaxy-importer**. It will end up in **/etc/galaxy-importer/galaxy-importer.cfg**
- **automationhub_require_content_approval**: Whether or not automation hub enforces the approval mechanism before collections are made available
- **automationhub_disable_https**: Whether or not automation hub should be deployed with TLS enabled
- **automationhub_disable_hsts**: Whether or not automation hub should be deployed with the HTTP Strict Transport Security (HSTS) web-security policy mechanism enabled
- **automationhub_ssl_validate_certs**: Whether or not automation hub should validate certificate when requesting itself (default = False) because by default, Platform deploys with self-signed certificates

- **automationhub_ssl_cert**: Same as **web_server_ssl_cert** but for automation hub UI and API
- **automationhub_ssl_key**: Same as **web_server_ssl_key** but for automation hub UI and API
- **automationhub_backup_collections**: automation hub provides artifacts in **/var/lib/pulp**. By default, this is set to **true** so automation controller automatically backs up the artifacts by default. If a partition (e.g., LVM, NFS, CephFS, etc.) was mounted there, an enterprise organization would ensure it is always backed up. If this is the case, you can set **automationhub_backup_collections = false** and the backup/restore process will not have to backup/restore **/var/lib/pulp**.

3.1.3. Editing the Red Hat Ansible Automation Platform installer inventory file

You can use the Red Hat Ansible Automation Platform installer inventory file to specify your installation scenario.

NOTE

- Using **external databases**: ensure the database sections of your inventory file are properly setup.
- Add automation hub information in the **[automationhub]** group
- automation hub and automation controller cannot be installed on the same node.
- automation controller will not configure replication or failover for the database that it uses. automation controller should work with any replication you have.
- The database server should be on the same network or in the same data center as the automation controller server for performance reasons.
- A container registry service is required to install the Ansible Automation Platform. Access to a container registry enables you to load automation execution environments onto the Ansible Automation Platform, giving you a consistent and containerized environment for executing Ansible playbooks and roles. By default, the Ansible Automation Platform uses **registry.redhat.io**, which requires a Red Hat registry service account. See the Creating Registry Service Accounts guide to create a registry service account.
- For **upgrading an existing cluster**: When upgrading a cluster, you may decide that you want to also reconfigure your cluster to omit existing instances or instance groups. Omitting the instance or the instance group from the inventory file will not be enough to remove them from the cluster. In addition to omitting instances or instance groups from the inventory file, you must also deprovision instances or instance groups before starting the upgrade. Otherwise, omitted instances or instance groups will continue to communicate with the cluster, which can cause issues with tower services during the upgrade.
- For **clustered installations**: If you are creating a clustered setup, you must replace **localhost** with the hostname or IP address of all instances. All nodes/instances must be able to reach any others using this hostname or address. In other words, you cannot use the **localhost ansible_connection=local** on one of the nodes AND all of the nodes should use the same format for the host names.

IMPORTANT

- Root access to remote machines is required. With Ansible, this can be achieved in different ways:
- **ansible_user=root ansible_ssh_pass="your_password_here"** inventory host or group variables
- **ansible_user=root ansible_ssh_private_key_file="path_to_your_keyfile.pem"** inventory host or group variables
- **ANSIBLE_BECOME_METHOD='sudo' ANSIBLE_BECOME=True ./setup.sh**

For more information on **become** plugins, see Understanding privilege escalation .

Procedure

1. Navigate to the installer
   a. [bundled installer]

```
$ cd ansible-automation-platform-setup-bundle-<latest-version>
```

   b. [online installer]

```
$ cd ansible-automation-platform-setup-<latest-version>
```

2. Open the **inventory** file with a text editor.
3. Edit **inventory** file parameters to specify your installation scenario. Follow the example below.

3.1.4. Example inventory file for a database on the automation controller node or a non-installer managed database

This example describes how you can populate the inventory file to install Red Hat Ansible Automation Platform. This installation inventory file includes both automation controller and automation hub with a database on the automation controller node or non-installer managed database.

IMPORTANT

- You cannot install automation controller and automation hub on the same node.
- Provide a reachable IP address for the **[automationhub]** host to ensure users can sync content from Private Automation Hub from a different node.
- Enter your Red Hat Registry Service Account credentials in **registry_username** and **registry_password** to link to the Red Hat container registry.

```
[automationcontroller]
controller.acme.org

[automationhub]
automationhub.acme.org
```

```
[all:vars]
admin_password='<password>'
pg_host=''
pg_port=''
pg_database='awx'
pg_username='awx'
pg_password='<password>'
pg_sslmode='prefer'  # set to 'verify-full' for client-side enforced SSL

registry_url='registry.redhat.io'
registry_username='<registry username>'
registry_password='<registry password>'

# Automation Hub Configuration
#
automationhub_admin_password='<password>'
automationhub_pg_host='controller.acme.org'
automationhub_pg_port='5432'
automationhub_pg_database='automationhub'
automationhub_pg_username='automationhub'
automationhub_pg_password='<password>'
automationhub_pg_sslmode='prefer'

# The default install will deploy a TLS enabled Automation Hub.
# If for some reason this is not the behavior wanted one can
# disable TLS enabled deployment.
#
# automationhub_disable_https = False
# The default install will generate self-signed certificates for the Automation
# Hub service. If you are providing valid certificate via automationhub_ssl_cert
# and automationhub_ssl_key, one should toggle that value to True.
#
# automationhub_ssl_validate_certs = False
# SSL-related variables
# If set, this will install a custom CA certificate to the system trust store.
# custom_ca_cert=/path/to/ca.crt
# Certificate and key to install in nginx for the web UI and API
# web_server_ssl_cert=/path/to/tower.cert
# web_server_ssl_key=/path/to/tower.key
# Certificate and key to install in Automation Hub node
# automationhub_ssl_cert=/path/to/automationhub.cert
# automationhub_ssl_key=/path/to/automationhub.key
# Server-side SSL settings for PostgreSQL (when we are installing it).
# postgres_use_ssl=False
# postgres_ssl_cert=/path/to/pgsql.crt
# postgres_ssl_key=/path/to/pgsql.key
```

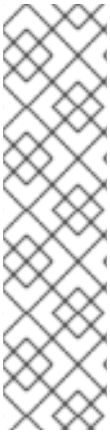
3.1.5. Setup script flags and extra variables

You can also pass flags and extra variables when running the setup script to install automation controller:

Table 3.1. Flags

Argument	Description
-h	Show this help message and exit
-i INVENTORY_FILE	Path to Ansible inventory file (default: **inventory**)
-e EXTRA_VARS	Set additional Ansible variables as key=value or YAML/JSON
-b	Perform a database backup in lieu of installing
-r	Perform a database restore in lieu of installing
-k	Generate **and** dsitribute a SECRET_KEY

Use the **--** separator to add any Ansible arguments you wish to apply. For example: **./setup.sh -i my_awesome_inventory.yml -e matburt_is_country_gold=True — -K**.



NOTE

- When passing **-r** to perform a database restore default restore path is used unless EXTRA_VARS are provided with a non-default path. See the example below that passed an EXTRA_VAR specifying the restore path:

```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

- You can force an online installation by passing **-e bundle_install=false**:

```
$ ./setup.sh -e bundle_install=false
```

Table 3.2. Extra variables

Variable	Description	Default
upgrade_ansible_with_tower	When installing automation controller make sure Ansible is also up to date	**False**
create_preload_data	When installing Tower also create the Demo Org, project, credential, Job Template, etc.	**True**
bundle_install_folder	When installing from a bundle where to put the bundled repos	**var/lib/tower-bundle**
nginx_disable_https	Disable HTTPS traffic through nginx, this is useful if offloading HTTPS to a load balancer	**False**

Variable	Description	Default
nginx_disable_hsts	Disable HSTS web-security policy mechanism	**False**
nginx_http_port	Port to configure nginx to listen to for HTTP	**80**
nginx_https_port	Port to configure nginx to listen to for HTTPS	**443**
backup_dir	A temp location to use when backing up	**/var/backups/tower/**
restore_backup_file	Specify an alternative backup file to restore from	None
required_ram	The minimum RAM required to install Tower (should only be changed for test installation)	**3750**
min_open_fds	The minimum open file descriptions (should only be changed for test installations)	None
ignore_preflight_errors	Ignore preflight checks, useful when installing into a template or other non-system image (overrides **required_ram** and **min_open_fds**)	**False**

Examples

- To upgrade core:

```
./setup.sh -e upgrade_ansible_with_tower=1
```

- To disable https handling at nginx:

```
./setup.sh -e nginx_disable_https=true
```

- To specify a non-default path when restoring from a backup file:

```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

3.1.6. Running the Red Hat Ansible Automation Platform installer setup script

You can run the setup script once you finish updating the **inventory** file with required parameters for installing your Private Automation Hub.

Procedure

1. Run the **setup.sh** script

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

The installation will begin.

3.1.7. Verifying automation controller installation

Once the installation completes, you can verify your automation controller has been installed successfully by logging in with the admin credentials you inserted into the **inventory** file.

Procedure

1. Navigate to the IP address specified for the automation controller node in the **inventory** file.
2. Log in with the Admin credentials you set in the **inventory** file.

NOTE

The automation controller server is accessible from port 80 (**https://<TOWER_SERVER_NAME>/**) but will redirect to port 443 so 443 needs to be available also.

IMPORTANT

If the installation fails and you are a customer who has purchased a valid license for Red Hat Ansible Automation Platform, please contact Ansible via the Red Hat Customer portal at https://access.redhat.com/.

Upon a successful login to automation controller, your installation of Red Hat Ansible Automation Platform 2.1 is now complete.

3.1.7.1. Additional automation controller configuration and resources

See the following resources to explore additional automation controller configurations.

Table 3.3. Resources to configure automation controller

Link	Description
Automation Controller Quick Setup Guide	Set up automation controller and run your first playbook
Automation Controller Administration Guide	Configure automation controller administration through customer scripts, management jobs, etc.
Configuring proxy support for Red Hat Ansible Automation Platform	Set up automation controller with a proxy server

Link	Description
Managing usability analytics and data collection from automation controller	Manage what automation controller information you share with Red Hat
Automation Controller User Guide	Review automation controller functionality in more detail

3.1.8. Verifying automation hub installation

Once the installation completes, you can verify your automation hub has been installed successfully by logging in with the admin credentials you inserted into the **inventory** file.

Procedure

1. Navigate to the IP address specified for the automation hub node in the **inventory** file.
2. Log in with the Admin credentials you set in the **inventory** file.

IMPORTANT

If the installation fails and you are a customer who has purchased a valid license for Red Hat Ansible Automation Platform, please contact Ansible via the Red Hat Customer portal at https://access.redhat.com/.

Upon a successful login to automation hub, your installation of Red Hat Ansible Automation Platform 2.1 is now complete.

3.1.8.1. Additional automation hub configuration and resources

See the following resources to explore additional automation hub configurations.

Table 3.4. Resources to configure automation controller

Link	Description
Managing user access in private automation hub	Configure user access for automation hub
Managing Red Hat Certified and Ansible Galaxy collections in automation hub	Add content to your automation hub
Publishing proprietary content collections in automation hub	Publish internally developed collections on your automation hub

3.1.9. What's next with Ansible Automation Platform 2.1

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 leveraging the new features of Ansible Automation Platform 2.1:

3.1.9.1. Migrating data to Ansible Automation Platform 2.1

For platform administrators looking to complete an upgrade to the Ansible Automation Platform 2.1, there may be additional steps needed to migrate data to a new instance:

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

Ansible Automation Platform 2.1 moves you away from custom Python virtual environments (venvs) in favor of automation execution environments - containerized images that package the necessary components needed to execute and scale your Ansible automation. This includes Ansible Core, Ansible Content Collections, Python dependencies, Red Hat Enterprise Linux UBI 8, and any additional package dependencies.

If you are looking to migrate your venvs to execution environments, you will (1) need to use the **awx-manage** command to list and export a list of venvs from your original instance, then (2) use **ansible-builder** to create execution environments. For more information, see the Upgrading to Automation Execution Environments guide and the Ansible Builder Guide .

3.1.9.1.2. Migrating to Ansible Engine 2.9 images using Ansible Builder

To migrate Ansible Engine 2.9 images for use with Ansible Automation Platform 2.1, the **ansible-builder** tool automates the process of rebuilding images (including its custom plugins and dependencies) for use with automation execution environments. For more information on using Ansible Builder to build execution environments, see the Ansible Builder Guide .

3.1.9.1.3. Migrating to Ansible Core 2.12

When upgrading to Ansible Core 2.12, you will need to update your playbooks, plugins, or other parts of your Ansible infrastructure in order to be supported by the latest version of Ansible Core. For instructions on updating your Ansible content to be Ansible Core 2.12 compatible, see the Ansible-core 2.12 Porting Guide.

3.1.9.2. Scale up your automation using 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 the Ansible Automation Platform, you will need to 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.

For instructions on how to migrate from isolated nodes to execution nodes, see the **upgrade & migration guide**.

For information about automation mesh and the various ways to design your automation mesh for your environment, see the Red Hat Ansible Automation Platform automation mesh guide .

3.2. INSTALLING RED HAT ANSIBLE AUTOMATION PLATFORM WITH AN EXTERNAL MANAGED DATABASE

You can use these instructions to install Red Hat Ansible Automation Platform (both automation controller and automation hub) with an external managed database.

3.2.1. Prerequisites

- You chose and obtained a platform installer from the Red Hat Ansible Automation Platform Product Software.
- You are installing on a machine that meets base system requirements.
- You have created a Red Hat Registry Service Account, following the instructions in the Creating Registry Service Accounts guide.

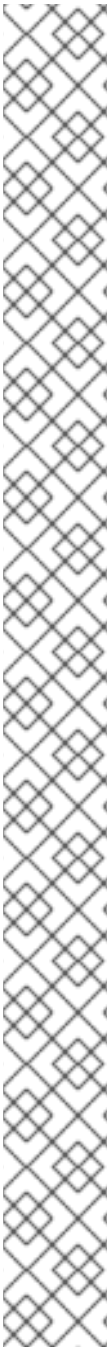
3.2.2. Red Hat Ansible Automation Platform installation settings

You can use the following settings when installing automation hub:

- **automationhub_importer_settings**: Dictionary of settings/configuration to pass to **galaxy-importer**. It will end up in **/etc/galaxy-importer/galaxy-importer.cfg**
- **automationhub_require_content_approval**: Whether or not automation hub enforces the approval mechanism before collections are made available
- **automationhub_disable_https**: Whether or not automation hub should be deployed with TLS enabled
- **automationhub_disable_hsts**: Whether or not automation hub should be deployed with the HTTP Strict Transport Security (HSTS) web-security policy mechanism enabled
- **automationhub_ssl_validate_certs**: Whether or not automation hub should validate certificate when requesting itself (default = False) because by default, Platform deploys with self-signed certificates
- **automationhub_ssl_cert**: Same as **web_server_ssl_cert** but for automation hub UI and API
- **automationhub_ssl_key**: Same as **web_server_ssl_key** but for automation hub UI and API
- **automationhub_backup_collections**: automation hub provides artifacts in **/var/lib/pulp**. By default, this is set to **true** so automation controller automatically backs up the artifacts by default. If a partition (e.g., LVM, NFS, CephFS, etc.) was mounted there, an enterprise organization would ensure it is always backed up. If this is the case, you can set **automationhub_backup_collections = false** and the backup/restore process will not have to backup/restore **/var/lib/pulp**.

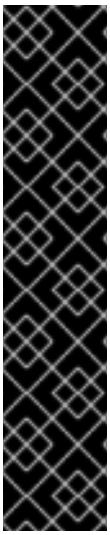
3.2.3. Editing the Red Hat Ansible Automation Platform installer inventory file

You can use the Red Hat Ansible Automation Platform installer inventory file to specify your installation scenario.



NOTE

- Using **external databases**: ensure the database sections of your inventory file are properly setup.
- Add automation hub information in the **[automationhub]** group
- automation hub and automation controller cannot be installed on the same node.
- automation controller will not configure replication or failover for the database that it uses. automation controller should work with any replication you have.
- The database server should be on the same network or in the same data center as the automation controller server for performance reasons.
- A container registry service is required to install the Ansible Automation Platform. Access to a container registry enables you to load automation execution environments onto the Ansible Automation Platform, giving you a consistent and containerized environment for executing Ansible playbooks and roles. By default, the Ansible Automation Platform uses **registry.redhat.io**, which requires a Red Hat registry service account. See the Creating Registry Service Accounts guide to create a registry service account.
- For **upgrading an existing cluster**: When upgrading a cluster, you may decide that you want to also reconfigure your cluster to omit existing instances or instance groups. Omitting the instance or the instance group from the inventory file will not be enough to remove them from the cluster. In addition to omitting instances or instance groups from the inventory file, you must also deprovision instances or instance groups before starting the upgrade. Otherwise, omitted instances or instance groups will continue to communicate with the cluster, which can cause issues with tower services during the upgrade.
- For **clustered installations**: If you are creating a clustered setup, you must replace **localhost** with the hostname or IP address of all instances. All nodes/instances must be able to reach any others using this hostname or address. In other words, you cannot use the **localhost ansible_connection=local** on one of the nodes AND all of the nodes should use the same format for the host names.



IMPORTANT

- Root access to remote machines is required. With Ansible, this can be achieved in different ways:
- **ansible_user=root ansible_ssh_pass="your_password_here"** inventory host or group variables
- **ansible_user=root ansible_ssh_private_key_file="path_to_your_keyfile.pem"** inventory host or group variables
- **ANSIBLE_BECOME_METHOD='sudo' ANSIBLE_BECOME=True ./setup.sh**

For more information on **become** plugins, see Understanding privilege escalation .

Procedure

1. Navigate to the installer
   a. [bundled installer]

```
$ cd ansible-automation-platform-setup-bundle-<latest-version>
```

   b. [online installer]

```
$ cd ansible-automation-platform-setup-<latest-version>
```

2. Open the **inventory** file with a text editor.
3. Edit **inventory** file parameters to specify your installation scenario. Follow the example below.

3.2.4. Example Red Hat Ansible Automation Platform inventory file with an external managed database

This example describes how you can populate the inventory file to install Red Hat Ansible Automation Platform. This installation inventory file includes both automation controller and automation hub with an external managed database.

IMPORTANT

- You cannot install automation controller and automation hub on the same node.
- Provide a reachable IP address for the **[automationhub]** host to ensure users can sync content from Private Automation Hub from a different node.
- Enter your Red Hat Registry Service Account credentials in **registry_username** and **registry_password** to link to the Red Hat container registry.

```
[automationcontroller]
controller.acme.org

[automationhub]
automationhub.acme.org

[database]
database-01.acme.org

[all:vars]
admin_password='<password>'
pg_host='database-01.acme.org'
pg_port='5432'
pg_database='awx'
pg_username='awx'
pg_password='<password>'
pg_sslmode='prefer'  # set to 'verify-full' for client-side enforced SSL

registry_url='registry.redhat.io'
registry_username='<registry username>'
registry_password='<registry password>'

# Automation Hub Configuration
```

```
#
automationhub_admin_password='<password>'
automationhub_pg_host='database-01.acme.org'
automationhub_pg_port='5432'
automationhub_pg_database='automationhub'
automationhub_pg_username='automationhub'
automationhub_pg_password='<password>'
automationhub_pg_sslmode='prefer'

# The default install will deploy a TLS enabled Automation Hub.
# If for some reason this is not the behavior wanted one can
# disable TLS enabled deployment.
#
# automationhub_disable_https = False
# The default install will generate self-signed certificates for the Automation
# Hub service. If you are providing valid certificate via automationhub_ssl_cert
# and automationhub_ssl_key, one should toggle that value to True.
#
# automationhub_ssl_validate_certs = False
# SSL-related variables
# If set, this will install a custom CA certificate to the system trust store.
# custom_ca_cert=/path/to/ca.crt
# Certificate and key to install in nginx for the web UI and API
# web_server_ssl_cert=/path/to/tower.cert
# web_server_ssl_key=/path/to/tower.key
# Certificate and key to install in Automation Hub node
# automationhub_ssl_cert=/path/to/automationhub.cert
# automationhub_ssl_key=/path/to/automationhub.key
# Server-side SSL settings for PostgreSQL (when we are installing it).
# postgres_use_ssl=False
# postgres_ssl_cert=/path/to/pgsql.crt
# postgres_ssl_key=/path/to/pgsql.key
```

3.2.5. Setup script flags and extra variables

You can also pass flags and extra variables when running the setup script to install automation controller:

Table 3.5. Flags

Argument	Description
-h	Show this help message and exit
-i INVENTORY_FILE	Path to Ansible inventory file (default: **inventory**)
-e EXTRA_VARS	Set additional Ansible variables as key=value or YAML/JSON
-b	Perform a database backup in lieu of installing
-r	Perform a database restore in lieu of installing

Argument	Description
-k	Generate **and** dsitribute a SECRET_KEY

Use the **--** separator to add any Ansible arguments you wish to apply. For example: **./setup.sh -i my_awesome_inventory.yml -e matburt_is_country_gold=True — -K**.

NOTE

- When passing **-r** to perform a database restore default restore path is used unless EXTRA_VARS are provided with a non-default path. See the example below that passed an EXTRA_VAR specifying the restore path:

```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

- You can force an online installation by passing **-e bundle_install=false**:

```
$ ./setup.sh -e bundle_install=false
```

Table 3.6. Extra variables

Variable	Description	Default
upgrade_ansible_with_tower	When installing automation controller make sure Ansible is also up to date	**False**
create_preload_data	When installing Tower also create the Demo Org, project, credential, Job Template, etc.	**True**
bundle_install_folder	When installing from a bundle where to put the bundled repos	**var/lib/tower-bundle**
nginx_disable_https	Disable HTTPS traffic through nginx, this is useful if offloading HTTPS to a load balancer	**False**
nginx_disable_hsts	Disable HSTS web-security policy mechanism	**False**
nginx_http_port	Port to configure nginx to listen to for HTTP	**80**
nginx_https_port	Port to configure nginx to listen to for HTTPS	**443**
backup_dir	A temp location to use when backing up	**/var/backups/tower/**

Variable	Description	Default
restore_backup_file	Specify an alternative backup file to restore from	None
required_ram	The minimum RAM required to install Tower (should only be changed for test installation)	**3750**
min_open_fds	The minimum open file descriptions (should only be changed for test installations)	None
ignore_preflight_errors	Ignore preflight checks, useful when installing into a template or other non-system image (overrides **required_ram** and **min_open_fds**)	**False**

Examples

- To upgrade core:

```
./setup.sh -e upgrade_ansible_with_tower=1
```

- To disable https handling at nginx:

```
./setup.sh -e nginx_disable_https=true
```

- To specify a non-default path when restoring from a backup file:

```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

3.2.6. Running the Red Hat Ansible Automation Platform installer setup script

You can run the setup script once you finish updating the **inventory** file with required parameters for installing your Private Automation Hub.

Procedure

1. Run the **setup.sh** script

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

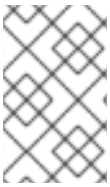
The installation will begin.

3.2.7. Verifying automation controller installation

Once the installation completes, you can verify your automation controller has been installed successfully by logging in with the admin credentials you inserted into the **inventory** file.

Procedure

1. Navigate to the IP address specified for the automation controller node in the **inventory** file.
2. Log in with the Admin credentials you set in the **inventory** file.



NOTE

The automation controller server is accessible from port 80 (**https://<TOWER_SERVER_NAME>/**) but will redirect to port 443 so 443 needs to be available also.

IMPORTANT

If the installation fails and you are a customer who has purchased a valid license for Red Hat Ansible Automation Platform, please contact Ansible via the Red Hat Customer portal at https://access.redhat.com/.

Upon a successful login to automation controller, your installation of Red Hat Ansible Automation Platform 2.1 is now complete.

3.2.7.1. Additional automation controller configuration and resources

See the following resources to explore additional automation controller configurations.

Table 3.7. Resources to configure automation controller

Link	Description
Automation Controller Quick Setup Guide	Set up automation controller and run your first playbook
Automation Controller Administration Guide	Configure automation controller administration through customer scripts, management jobs, etc.
Configuring proxy support for Red Hat Ansible Automation Platform	Set up automation controller with a proxy server
Managing usability analytics and data collection from automation controller	Manage what automation controller information you share with Red Hat
Automation Controller User Guide	Review automation controller functionality in more detail

3.2.8. Verifying automation hub installation

Once the installation completes, you can verify your automation hub has been installed successfully by logging in with the admin credentials you inserted into the **inventory** file.

Procedure

1. Navigate to the IP address specified for the automation hub node in the **inventory** file.

2. Log in with the Admin credentials you set in the **inventory** file.

IMPORTANT

If the installation fails and you are a customer who has purchased a valid license for Red Hat Ansible Automation Platform, please contact Ansible via the Red Hat Customer portal at https://access.redhat.com/.

Upon a successful login to automation hub, your installation of Red Hat Ansible Automation Platform 2.1 is now complete.

3.2.8.1. Additional automation hub configuration and resources

See the following resources to explore additional automation hub configurations.

Table 3.8. Resources to configure automation controller

Link	Description
Managing user access in private automation hub	Configure user access for automation hub
Managing Red Hat Certified and Ansible Galaxy collections in automation hub	Add content to your automation hub
Publishing proprietary content collections in automation hub	Publish internally developed collections on your automation hub

3.2.9. What's next with Ansible Automation Platform 2.1

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 leveraging the new features of Ansible Automation Platform 2.1:

3.2.9.1. Migrating data to Ansible Automation Platform 2.1

For platform administrators looking to complete an upgrade to the Ansible Automation Platform 2.1, there may be additional steps needed to migrate data to a new instance:

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

Ansible Automation Platform 2.1 moves you away from custom Python virtual environments (venvs) in favor of automation execution environments - containerized images that package the necessary components needed to execute and scale your Ansible automation. This includes Ansible Core, Ansible Content Collections, Python dependencies, Red Hat Enterprise Linux UBI 8, and any additional package dependencies.

If you are looking to migrate your venvs to execution environments, you will (1) need to use the **awx-manage** command to list and export a list of venvs from your original instance, then (2) use **ansible-builder** to create execution environments. For more information, see the Upgrading to Automation Execution Environments guide and the Ansible Builder Guide.

3.2.9.1.2. Migrating to Ansible Engine 2.9 images using Ansible Builder

To migrate Ansible Engine 2.9 images for use with Ansible Automation Platform 2.1, the **ansible-builder** tool automates the process of rebuilding images (including its custom plugins and dependencies) for use with automation execution environments. For more information on using Ansible Builder to build execution environments, see the Ansible Builder Guide .

3.2.9.1.3. Migrating to Ansible Core 2.12

When upgrading to Ansible Core 2.12, you will need to update your playbooks, plugins, or other parts of your Ansible infrastructure in order to be supported by the latest version of Ansible Core. For instructions on updating your Ansible content to be Ansible Core 2.12 compatible, see the Ansible-core 2.12 Porting Guide.

3.2.9.2. Scale up your automation using 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 the Ansible Automation Platform, you will need to 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.

For instructions on how to migrate from isolated nodes to execution nodes, see the **upgrade & migration guide**.

For information about automation mesh and the various ways to design your automation mesh for your environment, see the Red Hat Ansible Automation Platform automation mesh guide .

CHAPTER 4. MULTI-MACHINE CLUSTER INSTALLATION

You can install Ansible Automation Platform as clustered automation controller with automation hub with an external managed database. In this mode, multiple automation controller nodes are installed and active. Any node can receive HTTP requests and all nodes can execute jobs. This installs the Ansible Automation Platform server in a cluster and configures it to talk to a remote instance of PostgreSQL as its database. This remote PostgreSQL can be a server you manage, or can be provided by a cloud service such as Amazon RDS.

IMPORTANT

The Ansible Automation Platform installer allows you to deploy **only one** automation hub per inventory. You can use the Ansible Automation Platform installer for a standalone instance of automation hub and run the installer any number of times with any number of different inventories to deploy multiple automation hub nodes.

4.1. INSTALLING A MULTI-NODE RED HAT ANSIBLE AUTOMATION PLATFORM WITH AN EXTERNAL MANAGED DATABASE

You can use these instructions to install Red Hat Ansible Automation Platform as multiple automation controller nodes and automation hub with an external managed database.

4.1.1. Prerequisites

- You chose and obtained a platform installer from the Red Hat Ansible Automation Platform Product Software.
- You are installing on a machine that meets base system requirements.
- You have created a Red Hat Registry Service Account, following the instructions in the Creating Registry Service Accounts guide.

4.1.2. Red Hat Ansible Automation Platform installation settings

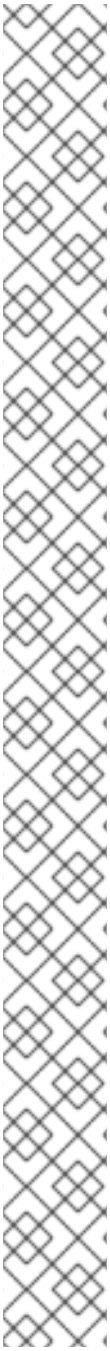
You can use the following settings when installing automation hub:

- **automationhub_importer_settings**: Dictionary of settings/configuration to pass to **galaxy-importer**. It will end up in **/etc/galaxy-importer/galaxy-importer.cfg**
- **automationhub_require_content_approval**: Whether or not automation hub enforces the approval mechanism before collections are made available
- **automationhub_disable_https**: Whether or not automation hub should be deployed with TLS enabled
- **automationhub_disable_hsts**: Whether or not automation hub should be deployed with the HTTP Strict Transport Security (HSTS) web-security policy mechanism enabled
- **automationhub_ssl_validate_certs**: Whether or not automation hub should validate certificate when requesting itself (default = False) because by default, Platform deploys with self-signed certificates
- **automationhub_ssl_cert**: Same as **web_server_ssl_cert** but for automation hub UI and API

- **automationhub_ssl_key**: Same as **web_server_ssl_key** but for automation hub UI and API
- **automationhub_backup_collections**: automation hub provides artifacts in **/var/lib/pulp**. By default, this is set to **true** so automation controller automatically backs up the artifacts by default. If a partition (e.g., LVM, NFS, CephFS, etc.) was mounted there, an enterprise organization would ensure it is always backed up. If this is the case, you can set **automationhub_backup_collections = false** and the backup/restore process will not have to backup/restore **/var/lib/pulp**.

4.1.3. Editing the Red Hat Ansible Automation Platform installer inventory file

You can use the Red Hat Ansible Automation Platform installer inventory file to specify your installation scenario.



NOTE

- Using **external databases**: ensure the database sections of your inventory file are properly setup.
- Add automation hub information in the **[automationhub]** group
- automation hub and automation controller cannot be installed on the same node.
- automation controller will not configure replication or failover for the database that it uses. automation controller should work with any replication you have.
- The database server should be on the same network or in the same data center as the automation controller server for performance reasons.
- A container registry service is required to install the Ansible Automation Platform. Access to a container registry enables you to load automation execution environments onto the Ansible Automation Platform, giving you a consistent and containerized environment for executing Ansible playbooks and roles. By default, the Ansible Automation Platform uses **registry.redhat.io**, which requires a Red Hat registry service account. See the Creating Registry Service Accounts guide to create a registry service account.
- For **upgrading an existing cluster**: When upgrading a cluster, you may decide that you want to also reconfigure your cluster to omit existing instances or instance groups. Omitting the instance or the instance group from the inventory file will not be enough to remove them from the cluster. In addition to omitting instances or instance groups from the inventory file, you must also deprovision instances or instance groups before starting the upgrade. Otherwise, omitted instances or instance groups will continue to communicate with the cluster, which can cause issues with tower services during the upgrade.
- For **clustered installations**: If you are creating a clustered setup, you must replace **localhost** with the hostname or IP address of all instances. All nodes/instances must be able to reach any others using this hostname or address. In other words, you cannot use the **localhost ansible_connection=local** on one of the nodes AND all of the nodes should use the same format for the host names.

IMPORTANT

- Root access to remote machines is required. With Ansible, this can be achieved in different ways:
- **ansible_user=root ansible_ssh_pass="your_password_here"** inventory host or group variables
- **ansible_user=root ansible_ssh_private_key_file="path_to_your_keyfile.pem"** inventory host or group variables
- **ANSIBLE_BECOME_METHOD='sudo' ANSIBLE_BECOME=True ./setup.sh**

For more information on **become** plugins, see Understanding privilege escalation .

Procedure

1. Navigate to the installer
   a. [bundled installer]

```
$ cd ansible-automation-platform-setup-bundle-<latest-version>
```

   b. [online installer]

```
$ cd ansible-automation-platform-setup-<latest-version>
```

2. Open the **inventory** file with a text editor.
3. Edit **inventory** file parameters to specify your installation scenario. Follow the example below.

4.1.4. Example Red Hat Ansible Automation Platform multi-node inventory file

This example describes how you can populate the inventory file for a multi-node cluster installation of automation controller.

IMPORTANT

- You cannot install automation controller and automation hub on the same node.
- Provide a reachable IP address for the **[automationhub]** host to ensure users can sync content from Private Automation Hub from a different node.
- Do not use special characters for **pg_password**. It may cause the setup to fail.
- Enter your Red Hat Registry Service Account credentials in **registry_username** and **registry_password** to link to the Red Hat container registry.

```
[automationcontroller]
host1
host11
host12
```

```
[automationhub]
host2

[database]
1

[all:vars]
ansible_become=true

admin_password='<password>'

pg_host='dbnode.example.com'
pg_port='5432'

pg_database='tower'
pg_username='tower'
pg_password='<password>'

registry_url='registry.redhat.io'
registry_username='<registry username>'
registry_password='<registry password>'
```

1 Field should be empty.

4.1.5. Setup script flags and extra variables

You can also pass flags and extra variables when running the setup script to install automation controller:

Table 4.1. Flags

Argument	Description
-h	Show this help message and exit
-i INVENTORY_FILE	Path to Ansible inventory file (default: **inventory**)
-e EXTRA_VARS	Set additional Ansible variables as key=value or YAML/JSON
-b	Perform a database backup in lieu of installing
-r	Perform a database restore in lieu of installing
-k	Generate **and** dsitribute a SECRET_KEY

Use the **--** separator to add any Ansible arguments you wish to apply. For example: **./setup.sh -i my_awesome_inventory.yml -e matburt_is_country_gold=True — -K**.

NOTE

- When passing **-r** to perform a database restore default restore path is used unless EXTRA_VARS are provided with a non-default path. See the example below that passed an EXTRA_VAR specifying the restore path:

```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

- You can force an online installation by passing **-e bundle_install=false**:

```
$ ./setup.sh -e bundle_install=false
```

Table 4.2. Extra variables

Variable	Description	Default
upgrade_ansible_with_tower	When installing automation controller make sure Ansible is also up to date	**False**
create_preload_data	When installing Tower also create the Demo Org, project, credential, Job Template, etc.	**True**
bundle_install_folder	When installing from a bundle where to put the bundled repos	**var/lib/tower-bundle**
nginx_disable_https	Disable HTTPS traffic through nginx, this is useful if offloading HTTPS to a load balancer	**False**
nginx_disable_hsts	Disable HSTS web-security policy mechanism	**False**
nginx_http_port	Port to configure nginx to listen to for HTTP	**80**
nginx_https_port	Port to configure nginx to listen to for HTTPS	**443**
backup_dir	A temp location to use when backing up	**/var/backups/tower/**
restore_backup_file	Specify an alternative backup file to restore from	None
required_ram	The minimum RAM required to install Tower (should only be changed for test installation)	**3750**

Variable	Description	Default
min_open_fds	The minimum open file descriptions (should only be changed for test installations)	None
ignore_preflight_errors	Ignore preflight checks, useful when installing into a template or other non-system image (overrides **required_ram** and **min_open_fds**)	**False**

Examples

- To upgrade core:

```
./setup.sh -e upgrade_ansible_with_tower=1
```

- To disable https handling at nginx:

```
./setup.sh -e nginx_disable_https=true
```

- To specify a non-default path when restoring from a backup file:

```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

4.1.6. Running the Red Hat Ansible Automation Platform installer setup script

You can run the setup script once you finish updating the **inventory** file with required parameters for installing your Private Automation Hub.

Procedure

1. Run the **setup.sh** script

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

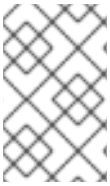
The installation will begin.

4.1.7. Verifying automation controller installation

Once the installation completes, you can verify your automation controller has been installed successfully by logging in with the admin credentials you inserted into the **inventory** file.

Procedure

1. Navigate to the IP address specified for the automation controller node in the **inventory** file.
2. Log in with the Admin credentials you set in the **inventory** file.



NOTE

The automation controller server is accessible from port 80 (**https://<TOWER_SERVER_NAME>/**) but will redirect to port 443 so 443 needs to be available also.

IMPORTANT

If the installation fails and you are a customer who has purchased a valid license for Red Hat Ansible Automation Platform, please contact Ansible via the Red Hat Customer portal at https://access.redhat.com/.

Upon a successful login to automation controller, your installation of Red Hat Ansible Automation Platform 2.1 is now complete.

4.1.7.1. Additional automation controller configuration and resources

See the following resources to explore additional automation controller configurations.

Table 4.3. Resources to configure automation controller

Link	Description
Automation Controller Quick Setup Guide	Set up automation controller and run your first playbook
Automation Controller Administration Guide	Configure automation controller administration through customer scripts, management jobs, etc.
Configuring proxy support for Red Hat Ansible Automation Platform	Set up automation controller with a proxy server
Managing usability analytics and data collection from automation controller	Manage what automation controller information you share with Red Hat
Automation Controller User Guide	Review automation controller functionality in more detail

4.1.8. Verifying automation hub installation

Once the installation completes, you can verify your automation hub has been installed successfully by logging in with the admin credentials you inserted into the **inventory** file.

Procedure

1. Navigate to the IP address specified for the automation hub node in the **inventory** file.
2. Log in with the Admin credentials you set in the **inventory** file.

IMPORTANT

If the installation fails and you are a customer who has purchased a valid license for Red Hat Ansible Automation Platform, please contact Ansible via the Red Hat Customer portal at https://access.redhat.com/.

Upon a successful login to automation hub, your installation of Red Hat Ansible Automation Platform 2.1 is now complete.

4.1.8.1. Additional automation hub configuration and resources

See the following resources to explore additional automation hub configurations.

Table 4.4. Resources to configure automation controller

Link	Description
Managing user access in private automation hub	Configure user access for automation hub
Managing Red Hat Certified and Ansible Galaxy collections in automation hub	Add content to your automation hub
Publishing proprietary content collections in automation hub	Publish internally developed collections on your automation hub

4.1.9. What's next with Ansible Automation Platform 2.1

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 leveraging the new features of Ansible Automation Platform 2.1:

4.1.9.1. Migrating data to Ansible Automation Platform 2.1

For platform administrators looking to complete an upgrade to the Ansible Automation Platform 2.1, there may be additional steps needed to migrate data to a new instance:

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

Ansible Automation Platform 2.1 moves you away from custom Python virtual environments (venvs) in favor of automation execution environments - containerized images that package the necessary components needed to execute and scale your Ansible automation. This includes Ansible Core, Ansible Content Collections, Python dependencies, Red Hat Enterprise Linux UBI 8, and any additional package dependencies.

If you are looking to migrate your venvs to execution environments, you will (1) need to use the **awx-manage** command to list and export a list of venvs from your original instance, then (2) use **ansible-builder** to create execution environments. For more information, see the Upgrading to Automation Execution Environments guide and the Ansible Builder Guide.

4.1.9.1.2. Migrating to Ansible Engine 2.9 images using Ansible Builder

To migrate Ansible Engine 2.9 images for use with Ansible Automation Platform 2.1, the **ansible-builder**

tool automates the process of rebuilding images (including its custom plugins and dependencies) for use with automation execution environments. For more information on using Ansible Builder to build execution environments, see the Ansible Builder Guide .

4.1.9.1.3. Migrating to Ansible Core 2.12

When upgrading to Ansible Core 2.12, you will need to update your playbooks, plugins, or other parts of your Ansible infrastructure in order to be supported by the latest version of Ansible Core. For instructions on updating your Ansible content to be Ansible Core 2.12 compatible, see the Ansible-core 2.12 Porting Guide.

4.1.9.2. Scale up your automation using 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 the Ansible Automation Platform, you will need to 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.

For instructions on how to migrate from isolated nodes to execution nodes, see the **upgrade & migration guide**.

For information about automation mesh and the various ways to design your automation mesh for your environment, see the Red Hat Ansible Automation Platform automation mesh guide .

CHAPTER 5. 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.

5.1. 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 → Miscellaneous 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.

5.2. 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.

5.2.1. Configuring known proxies

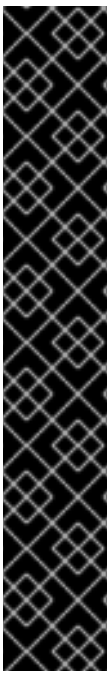
To configure a list of known proxies for your automation controller, add the proxy IP addresses to the **PROXY_IP_ALLOWED_LIST** field in the settings page for your automation controller.

Procedure

1. On your automation controller, navigate to **Settings → Miscellaneous 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 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.

5.3. 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 → Miscellaneous System**.
2. In the **REMOTE_HOST_HEADERS** field, enter the following values:

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

CHAPTER 6. CONFIGURING AUTOMATION CONTROLLER WEBSOCKET CONNECTIONS

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

6.1. WEBSOCKET CONFIGURATION FOR AUTOMATION CONTROLLER

Automation controller nodes connect to all other automation controller nodes via websockets. This interconnect is used to distribute all websocket emitted messages to all other automation controller nodes. This is required because any browser client websocket can subscribe to any job that may be running on any automation controller node. Websocket clients are not routed to specific automation controller nodes. Any automation controller node can handle any websocket request and each automation controller node must know about all websocket messages destined for all clients.

Automation controller will automatically handle discovery of other automation controller nodes via the Instance record in the database.

IMPORTANT

- It is intended that your nodes are broadcasting websocket traffic across a private, trusted subnet (and not the open Internet). Therefore, if you turn off HTTPS for websocket broadcasting, the websocket traffic, comprised mostly of Ansible playbook stdout, is sent between automation controller nodes unencrypted.

6.1.1. 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.

- Edit automation controller websocket information for port, protocol, and whether or not to verify certificates when establishing the websocket connections.

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

CHAPTER 7. 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.

7.1. 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 .

7.1.1. 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. Navgate to **Settings → User Interface**
3. Select the desired level of data collection from the User Analytics Tracking State drop-down list:
    a. **Off**: Prevents any data collection.
    b. **Anonymous**: Enables data collection without your specific user data.
    c. **Detailed**: Enables data collection including your specific user data.
4. Click **Save** to apply the settings or **Cancel** to abandon the changes.

CHAPTER 8. SUPPORTED INVENTORY PLUGINS TEMPLATES

On upgrade, existing configurations will be migrated to the new format that will produce a backwards compatible inventory output. Use the templates below to help aid in migrating your inventories to the new style inventory plugin output.

8.1. AMAZON WEB SERVICES EC2

```
compose:
  ansible_host: public_ip_address
  ec2_account_id: owner_id
  ec2_ami_launch_index: ami_launch_index | string
  ec2_architecture: architecture
  ec2_block_devices: dict(block_device_mappings | map(attribute='device_name') | list |
zip(block_device_mappings | map(attribute='ebs.volume_id') | list))
  ec2_client_token: client_token
  ec2_dns_name: public_dns_name
  ec2_ebs_optimized: ebs_optimized
  ec2_eventsSet: events | default("")
  ec2_group_name: placement.group_name
  ec2_hypervisor: hypervisor
  ec2_id: instance_id
  ec2_image_id: image_id
  ec2_instance_profile: iam_instance_profile | default("")
  ec2_instance_type: instance_type
  ec2_ip_address: public_ip_address
  ec2_kernel: kernel_id | default("")
  ec2_key_name: key_name
  ec2_launch_time: launch_time | regex_replace(" ", "T") | regex_replace("(\+)(\d\d):(\d)(\d)$",
".\g<2>\g<3>Z")
  ec2_monitored: monitoring.state in ['enabled', 'pending']
  ec2_monitoring_state: monitoring.state
  ec2_persistent: persistent | default(false)
  ec2_placement: placement.availability_zone
  ec2_platform: platform | default("")
  ec2_private_dns_name: private_dns_name
  ec2_private_ip_address: private_ip_address
  ec2_public_dns_name: public_dns_name
  ec2_ramdisk: ramdisk_id | default("")
  ec2_reason: state_transition_reason
  ec2_region: placement.region
  ec2_requester_id: requester_id | default("")
  ec2_root_device_name: root_device_name
  ec2_root_device_type: root_device_type
  ec2_security_group_ids: security_groups | map(attribute='group_id') | list |  join(',')
  ec2_security_group_names: security_groups | map(attribute='group_name') | list |  join(',')
  ec2_sourceDestCheck: source_dest_check | default(false) | lower | string
  ec2_spot_instance_request_id: spot_instance_request_id | default("")
  ec2_state: state.name
  ec2_state_code: state.code
  ec2_state_reason: state_reason.message if state_reason is defined else ""
  ec2_subnet_id: subnet_id | default("")
  ec2_tag_Name: tags.Name
  ec2_virtualization_type: virtualization_type
  ec2_vpc_id: vpc_id | default("")
```

```
filters:
  instance-state-name:
  - running
groups:
  ec2: true
hostnames:
  - network-interface.addresses.association.public-ip
  - dns-name
  - private-dns-name
keyed_groups:
  - key: image_id | regex_replace("[^A-Za-z0-9\_]", "_")
    parent_group: images
    prefix: ''
    separator: ''
  - key: placement.availability_zone
    parent_group: zones
    prefix: ''
    separator: ''
  - key: ec2_account_id | regex_replace("[^A-Za-z0-9\_]", "_")
    parent_group: accounts
    prefix: ''
    separator: ''
  - key: ec2_state | regex_replace("[^A-Za-z0-9\_]", "_")
    parent_group: instance_states
    prefix: instance_state
  - key: platform | default("undefined") | regex_replace("[^A-Za-z0-9\_]", "_")
    parent_group: platforms
    prefix: platform
  - key: instance_type | regex_replace("[^A-Za-z0-9\_]", "_")
    parent_group: types
    prefix: type
  - key: key_name | regex_replace("[^A-Za-z0-9\_]", "_")
    parent_group: keys
    prefix: key
  - key: placement.region
    parent_group: regions
    prefix: ''
    separator: ''
  - key: security_groups | map(attribute="group_name") | map("regex_replace", "[^A-Za-z0-9\_]", "_") |
list
    parent_group: security_groups
    prefix: security_group
  - key: dict(tags.keys() | map("regex_replace", "[^A-Za-z0-9\_]", "_") | list | zip(tags.values()
      | map("regex_replace", "[^A-Za-z0-9\_]", "_") | list))
    parent_group: tags
    prefix: tag
  - key: tags.keys() | map("regex_replace", "[^A-Za-z0-9\_]", "_") | list
    parent_group: tags
    prefix: tag
  - key: vpc_id | regex_replace("[^A-Za-z0-9\_]", "_")
    parent_group: vpcs
    prefix: vpc_id
  - key: placement.availability_zone
    parent_group: '{{ placement.region }}'
    prefix: ''
```

```
    separator: ''
plugin: amazon.aws.aws_ec2
use_contrib_script_compatible_sanitization: true
```

8.2. GOOGLE COMPUTE ENGINE

```
auth_kind: serviceaccount
compose:
  ansible_ssh_host: networkInterfaces[0].accessConfigs[0].natIP |
default(networkInterfaces[0].networkIP)
  gce_description: description if description else None
  gce_id: id
  gce_image: image
  gce_machine_type: machineType
  gce_metadata: metadata.get("items", []) | items2dict(key_name="key", value_name="value")
  gce_name: name
  gce_network: networkInterfaces[0].network.name
  gce_private_ip: networkInterfaces[0].networkIP
  gce_public_ip: networkInterfaces[0].accessConfigs[0].natIP | default(None)
  gce_status: status
  gce_subnetwork: networkInterfaces[0].subnetwork.name
  gce_tags: tags.get("items", [])
  gce_zone: zone
hostnames:
- name
- public_ip
- private_ip
keyed_groups:
- key: gce_subnetwork
  prefix: network
- key: gce_private_ip
  prefix: ''
  separator: ''
- key: gce_public_ip
  prefix: ''
  separator: ''
- key: machineType
  prefix: ''
  separator: ''
- key: zone
  prefix: ''
  separator: ''
- key: gce_tags
  prefix: tag
- key: status | lower
  prefix: status
- key: image
  prefix: ''
  separator: ''
plugin: google.cloud.gcp_compute
retrieve_image_info: true
use_contrib_script_compatible_sanitization: true
```

8.3. MICROSOFT AZURE RESOURCE MANAGER

```
conditional_groups:
  azure: true
default_host_filters: []
fail_on_template_errors: false
hostvar_expressions:
  computer_name: name
  private_ip: private_ipv4_addresses[0] if private_ipv4_addresses else None
  provisioning_state: provisioning_state | title
  public_ip: public_ipv4_addresses[0] if public_ipv4_addresses else None
  public_ip_id: public_ip_id if public_ip_id is defined else None
  public_ip_name: public_ip_name if public_ip_name is defined else None
  tags: tags if tags else None
  type: resource_type
keyed_groups:
- key: location
  prefix: ''
  separator: ''
- key: tags.keys() | list if tags else []
  prefix: ''
  separator: ''
- key: security_group
  prefix: ''
  separator: ''
- key: resource_group
  prefix: ''
  separator: ''
- key: os_disk.operating_system_type
  prefix: ''
  separator: ''
- key: dict(tags.keys() | map("regex_replace", "^(.*)$", "\1_") | list | zip(tags.values() | list)) if tags else
[]
  prefix: ''
  separator: ''
plain_host_names: true
plugin: azure.azcollection.azure_rm
use_contrib_script_compatible_sanitization: true
```

8.4. VMWARE VCENTER

```
compose:
  ansible_host: guest.ipAddress
  ansible_ssh_host: guest.ipAddress
  ansible_uuid: 99999999 | random | to_uuid
  availablefield: availableField
  configissue: configIssue
  configstatus: configStatus
  customvalue: customValue
  effectiverole: effectiveRole
  guestheartbeatstatus: guestHeartbeatStatus
  layoutex: layoutEx
  overallstatus: overallStatus
  parentvapp: parentVApp
  recenttask: recentTask
  resourcepool: resourcePool
  rootsnapshot: rootSnapshot
```

```
  triggeredalarmstate: triggeredAlarmState
filters:
- runtime.powerState == "poweredOn"
keyed_groups:
- key: config.guestId
  prefix: ''
  separator: ''
- key: '"templates" if config.template else "guests"'
  prefix: ''
  separator: ''
plugin: community.vmware.vmware_vm_inventory
properties:
- availableField
- configIssue
- configStatus
- customValue
- datastore
- effectiveRole
- guestHeartbeatStatus
- layout
- layoutEx
- name
- network
- overallStatus
- parentVApp
- permission
- recentTask
- resourcePool
- rootSnapshot
- snapshot
- triggeredAlarmState
- value
- capability
- config
- guest
- runtime
- storage
- summary
strict: false
with_nested_properties: true
```

8.5. RED HAT SATELLITE 6

```
group_prefix: foreman_
keyed_groups:
- key: foreman['environment_name'] | lower | regex_replace(' ', '') | regex_replace('[^A-Za-z0-9_]', '_') |
regex_replace('none', '')
  prefix: foreman_environment_
  separator: ''
- key: foreman['location_name'] | lower | regex_replace(' ', '') | regex_replace('[^A-Za-z0-9_]', '_')
  prefix: foreman_location_
  separator: ''
- key: foreman['organization_name'] | lower | regex_replace(' ', '') | regex_replace('[^A-Za-z0-9_]', '_')
  prefix: foreman_organization_
  separator: ''
```

```
- key: foreman['content_facet_attributes']['lifecycle_environment_name'] | lower | regex_replace(' ', '') |
regex_replace('[^A-Za-z0-9_]', '_')
  prefix: foreman_lifecycle_environment_
  separator: ''
- key: foreman['content_facet_attributes']['content_view_name'] | lower | regex_replace(' ', '') |
regex_replace('[^A-Za-z0-9_]', '_')
  prefix: foreman_content_view_
  separator: ''
legacy_hostvars: true
plugin: theforeman.foreman.foreman
validate_certs: false
want_facts: true
want_hostcollections: false
want_params: true
```

8.6. OPENSTACK

```
expand_hostvars: true
fail_on_errors: true
inventory_hostname: uuid
plugin: openstack.cloud.openstack
```

8.7. RED HAT VIRTUALIZATION

```
compose:
  ansible_host: (devices.values() | list)[0][0] if devices else None
keyed_groups:
- key: cluster
  prefix: cluster
  separator: _
- key: status
  prefix: status
  separator: _
- key: tags
  prefix: tag
  separator: _
ovirt_hostname_preference:
- name
- fqdn
ovirt_insecure: false
plugin: ovirt.ovirt.ovirt
```

8.8. AUTOMATION CONTROLLER

```
include_metadata: true
inventory_id: <inventory_id or url_quoted_named_url>
plugin: awx.awx.tower
validate_certs: <true or false>
```

CHAPTER 9. SUPPORTED ATTRIBUTES FOR CUSTOM NOTIFICATIONS

This section describes the list of supported job attributes and the proper syntax for constructing the message text for notifications. The supported job attributes are:

- **allow_simultaneous** - (boolean) indicates if multiple jobs can run simultaneously from the JT associated with this job
- **controller_node** - (string) the instance that managed the isolated execution environment
- **created** - (datetime) timestamp when this job was created
- **custom_virtualenv** - (string) custom virtual environment used to execute job
- **description** - (string) optional description of the job
- **diff_mode** - (boolean) if enabled, textual changes made to any templated files on the host are shown in the standard output
- **elapsed** - (decimal) elapsed time in seconds that the job ran
- **execution_node** - (string) node the job executed on
- **failed** - (boolean) true if job failed
- **finished** - (datetime) date and time the job finished execution
- **force_handlers** - (boolean) when handlers are forced, they will run when notified even if a task fails on that host (note that some conditions - e.g. unreachable hosts - can still prevent handlers from running)
- **forks** - (int) number of forks requested for job
- **id** - (int) database id for this job
- **job_explanation** - (string) status field to indicate the state of the job if it wasn't able to run and capture stdout
- **job_slice_count** - (integer) if run as part of a sliced job, the total number of slices (if 1, job is not part of a sliced job)
- **job_slice_number** - (integer) if run as part of a sliced job, the ID of the inventory slice operated on (if not part of a sliced job, attribute is not used)
- **job_tags** - (string) only tasks with specified tags will execute
- **job_type** - (choice) run, check, or scan
- **launch_type** - (choice) manual, relaunch, callback, scheduled, dependency, workflow, sync, or scm
- **limit** - (string) playbook execution limited to this set of hosts, if specified
- **modified** - (datetime) timestamp when this job was last modified
- **name** - (string) name of this job

- **playbook** - (string) playbook executed
- **scm_revision** - (string) scm revision from the project used for this job, if available
- **skip_tags** - (string) playbook execution skips over this set of tag(s), if specified
- **start_at_task** - (string) playbook execution begins at the task matching this name, if specified
- **started** - (datetime) date and time the job was queued for starting
- **status** - (choice) new, pending, waiting, running, successful, failed, error, canceled
- **timeout** - (int) amount of time (in seconds) to run before the task is canceled
- **type** - (choice) data type for this job
- **url** - (string) URL for this job
- **use_fact_cache** - (boolean) if enabled for job, Tower acts as an Ansible Fact Cache Plugin, persisting facts at the end of a playbook run to the database and caching facts for use by Ansible
- **verbosity** - (choice) 0 through 5 (corresponding to Normal through WinRM Debug)
- **host_status_counts** (count of hosts uniquely assigned to each status)
  - **skipped** (integer)
  - **ok** (integer)
  - **changed** (integer)
  - **failures** (integer)
  - **dark** (integer)
  - **processed** (integer)
  - **rescued** (integer)
  - **ignored** (integer)
  - **failed** (boolean)
- **summary_fields:**
  - **inventory**
    - **id** - (integer) database ID for inventory
    - **name** - (string) name of the inventory
    - **description** - (string) optional description of the inventory
    - **has_active_failures** - (boolean) (deprecated) flag indicating whether any hosts in this inventory have failed
    - **total_hosts** - (deprecated) (int) total number of hosts in this inventory.

        - **hosts_with_active_failures** - (deprecated) (int) number of hosts in this inventory with active failures
        - **total_groups** - (deprecated) (int) total number of groups in this inventory
        - **groups_with_active_failures** - (deprecated) (int) number of hosts in this inventory with active failures
        - **has_inventory_sources** - (deprecated) (boolean) flag indicating whether this inventory has external inventory sources
        - **total_inventory_sources** - (int) total number of external inventory sources configured within this inventory
        - **inventory_sources_with_failures** - (int) number of external inventory sources in this inventory with failures
        - **organization_id** - (id) organization containing this inventory
        - **kind** - (choice) (empty string) (indicating hosts have direct link with inventory) or 'smart'
    - **project**
        - **id** - (int) database ID for project
        - **name** - (string) name of the project
        - **description** - (string) optional description of the project
        - **status** - (choices) one of new, pending, waiting, running, successful, failed, error, canceled, never updated, ok, or missing
        - **scm_type** (choice) - one of (empty string), git, hg, svn, insights
    - **job_template**
        - **id** - (int) database ID for job template
        - **name** - (string) name of job template
        - **description** - (string) optional description for the job template
    - **unified_job_template**
        - **id** - (int) database ID for unified job template
        - **name** - (string) name of unified job template
        - **description** - (string) optional description for the unified job template
        - **unified_job_type** - (choice) unified job type (job, workflow_job, project_update, etc.)
    - **instance_group**
        - **id** - (int) database ID for instance group
        - **name** - (string) name of instance group

- **created_by**
  - **id** - (int) database ID of user that launched the operation
  - **username** - (string) username that launched the operation
  - **first_name** - (string) first name
  - **last_name** - (string) last name
- **labels**
  - **count** - (int) number of labels
  - **results** - list of dictionaries representing labels (e.g. {"id": 5, "name": "database jobs"})

Information about a job can be referenced in a custom notification message using grouped curly braces {{ }}. Specific job attributes are accessed using dotted notation, for example {{ job.summary_fields.inventory.name }}. Any characters used in front or around the braces, or plain text, can be added for clarification, such as '#' for job ID and single-quotes to denote some descriptor. Custom messages can include a number of variables throughout the message:

```
{{ job_friendly_name }} {{ job.id }} ran on {{ job.execution_node }} in {{ job.elapsed }} seconds.
```

In addition to the job attributes, there are some other variables that can be added to the template:

approval_node_name - (string) the approval node name

approval_status - (choice) one of approved, denied, and timed_out

url - (string) URL of the job for which the notification is emitted (this applies to start, success, fail, and approval notifications)

workflow_url - (string) URL to the relevant approval node. This allows the notification recipient to go to the relevant workflow job page to see what's going on (i.e., This node can be viewed at: {{ workflow_url }}). In cases of approval-related notifications, both url and workflow_url are the same.

job_friendly_name - (string) the friendly name of the job

job_metadata - (string) job metadata as a JSON string, for example:

```
{'url': 'https://towerhost/$/jobs/playbook/13',
 'traceback': '',
 'status': 'running',
 'started': '2019-08-07T21:46:38.362630+00:00',
 'project': 'Stub project',
 'playbook': 'ping.yml',
 'name': 'Stub Job Template',
 'limit': '',
 'inventory': 'Stub Inventory',
 'id': 42,
 'hosts': {},
 'friendly_name': 'Job',
 'finished': False,
 'credential': 'Stub credential',
 'created_by': 'admin'}
```