





Join a community dedicated to learning open source

The Red Hat® Learning Community is a collaborative platform for users to accelerate open source skill adoption while working with Red Hat products and experts.



Network with tens of thousands of community members



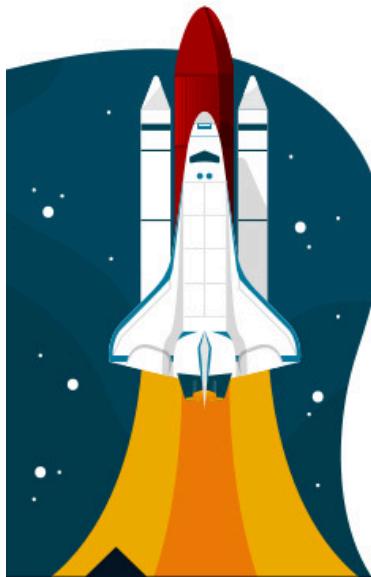
Engage in thousands of active conversations and posts



Join and interact with hundreds of certified training instructors



Unlock badges as you participate and accomplish new goals



This knowledge-sharing platform creates a space where learners can connect, ask questions, and collaborate with other open source practitioners.

Access free Red Hat training videos

Discover the latest Red Hat Training and Certification news

Connect with your instructor - and your classmates - before, after, and during your training course.

Join peers as you explore Red Hat products

Join the conversation learn.redhat.com



Copyright © 2020 Red Hat, Inc. Red Hat, Red Hat Enterprise Linux, the Red Hat logo, and Ansible are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.

Managing Enterprise Automation with Red Hat Ansible Automation Platform



Red Hat Ansible Automation Platform 2.2 DO467
Managing Enterprise Automation with Red Hat Ansible
Automation Platform
Edition 3 20240829
Publication date 20240829

Authors: Michael Phillips, Karan Rai, Alejandra Ramírez Palacios,
Dallas Spohn, Antonio Marí Romero
Course Architect: Steven Bonneville
DevOps Engineer: Dan Kolepp
Editor: David O'Brien

© 2024 Red Hat, Inc.

The contents of this course and all its modules and related materials, including handouts to audience members, are © 2024 Red Hat, Inc.

No part of this publication may be stored in a retrieval system, transmitted or reproduced in any way, including, but not limited to, photocopy, photograph, magnetic, electronic or other record, without the prior written permission of Red Hat, Inc.

This instructional program, including all material provided herein, is supplied without any guarantees from Red Hat, Inc. Red Hat, Inc. assumes no liability for damages or legal action arising from the use or misuse of contents or details contained herein.

If you believe Red Hat training materials are being used, copied, or otherwise improperly distributed, please send email to training@redhat.com [mailto:training@redhat.com] or phone toll-free (USA) +1 (866) 626-2994 or +1 (919) 754-3700.

Red Hat, Red Hat Enterprise Linux, the Red Hat logo, JBoss, OpenShift, Fedora, Hibernate, Ansible, RHCA, RHCE, RHCSA, Ceph, and Gluster are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.

Linux® is the registered trademark of Linus Torvalds in the United States and other countries.

Java® is a registered trademark of Oracle American, Inc. and/or its affiliates.

XFS® is a registered trademark of Hewlett Packard Enterprise Development LP or its subsidiaries in the United States and/or other countries.

MySQL® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js® is a trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack word mark and the Square O Design, together or apart, are trademarks or registered trademarks of OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. Red Hat, Inc. is not affiliated with, endorsed by, or sponsored by the OpenStack Foundation or the OpenStack community.

All other trademarks are the property of their respective owners.

Contributors:David Sacco, Sajith Eyamkuzhy Sugathan, Richard Allred, Travis Michette, Carlos Arias

Document Conventions	xi
Admonitions	xi
Inclusive Language	xii
Introduction	xiii
Managing Enterprise Automation with Red Hat Ansible Automation Platform	xiii
Orientation to the Classroom Environment	xiv
Performing Lab Exercises	xviii
Obtaining a Trial Subscription to Red Hat Ansible Automation Platform	xix
1. Installing Red Hat Ansible Automation Platform	1
Explaining the Red Hat Ansible Automation Platform Architecture	2
Quiz: Explaining the Red Hat Ansible Automation Platform Architecture	8
Installing Automation Controller and Private Automation Hub	12
Guided Exercise: Installing Automation Controller and Private Automation Hub	24
Initial Configuration of Automation Controller and Private Automation Hub	32
Guided Exercise: Initial Configuration of Automation Controller and Private Automation Hub	43
Quiz: Installing Red Hat Ansible Automation Platform	48
Summary	52
2. Managing User Access	53
Creating and Managing Automation Controller Users	54
Guided Exercise: Creating and Managing Automation Controller Users	60
Managing Automation Controller Access with Teams	63
Guided Exercise: Managing Automation Controller Access with Teams	68
Creating and Managing Users and Groups for Private Automation Hub	73
Guided Exercise: Creating and Managing Users and Groups for Private Automation Hub	79
Lab: Managing User Access	83
Summary	92
3. Managing Inventories and Machine Credentials	93
Creating a Static Inventory	94
Guided Exercise: Creating a Static Inventory	102
Creating Machine Credentials for Access to Inventory Hosts	106
Guided Exercise: Creating Machine Credentials for Access to Inventory Hosts	113
Lab: Managing Inventories and Machine Credentials	116
Summary	122
4. Managing Projects and Launching Ansible Jobs	123
Creating a Project for Ansible Playbooks	124
Guided Exercise: Creating a Project for Ansible Playbooks	131
Creating Job Templates and Launching Jobs	134
Guided Exercise: Creating Job Templates and Launching Jobs	141
Lab: Managing Projects and Launching Ansible Jobs	146
Summary	154
5. Advanced Job Configuration	155
Improving Performance with Fact Caching	156
Guided Exercise: Improving Performance with Fact Caching	159
Creating Job Template Surveys to Set Variables for Jobs	162
Guided Exercise: Creating Job Template Surveys to Set Variables for Jobs	167
Scheduling Jobs and Configuring Notifications	171
Guided Exercise: Scheduling Jobs and Configuring Notifications	178
Lab: Advanced Job Configuration	183
Summary	189

6. Constructing Job Workflows	191
Creating Workflow Job Templates and Launching Workflow Jobs	192
Guided Exercise: Creating Workflow Job Templates and Launching Workflow Jobs	201
Requiring Approvals in Workflow Jobs	207
Guided Exercise: Requiring Approvals in Workflow Jobs	212
Lab: Constructing Job Workflows	219
Summary	235
7. Managing Advanced Inventories	237
Importing External Static Inventories	238
Guided Exercise: Importing External Static Inventories	241
Configuring Dynamic Inventory Plug-ins	245
Quiz: Configuring Dynamic Inventory Plug-ins	251
Filtering Hosts with Smart Inventories	255
Guided Exercise: Filtering Hosts with Smart Inventories	259
Lab: Managing Advanced Inventories	262
Summary	270
8. Automating Configuration of Ansible Automation Platform	271
Configuring Red Hat Ansible Automation Platform with Collections	272
Guided Exercise: Configuring Red Hat Ansible Automation Platform with Collections	280
Automating Configuration Updates with Git Webhooks	289
Guided Exercise: Automating Configuration Updates with Git Webhooks	296
Launching Jobs with the Automation Controller API	300
Guided Exercise: Launching Jobs with the Automation Controller API	317
Lab: Automating Configuration of Ansible Automation Platform	323
Summary	333
9. Maintaining Red Hat Ansible Automation Platform	335
Performing Basic Troubleshooting of Automation Controller	336
Guided Exercise: Performing Basic Troubleshooting of Automation Controller	343
Backing up and Restoring Red Hat Ansible Automation Platform	347
Guided Exercise: Backing up and Restoring Red Hat Ansible Automation Platform	351
Quiz: Maintaining Red Hat Ansible Automation Platform	355
Summary	359
10. Getting Insights into Automation Performance	361
Gathering Data for Cloud-based Analysis	362
Quiz: Gathering Data for Cloud-based Analysis	366
Getting Insights into Automation Performance	368
Quiz: Getting Insights into Automation Performance	379
Evaluating Performance with Automation Analytics	381
Quiz: Evaluating Performance with Automation Analytics	386
Producing Reports from Automation Analytics	388
Quiz: Producing Reports from Automation Analytics	398
Summary	400
11. Building a Large-scale Red Hat Ansible Automation Platform Deployment	401
Designing a Clustered Ansible Automation Platform Implementation	402
Quiz: Designing a Clustered Ansible Automation Platform Implementation	411
Deploying Distributed Execution with Automation Mesh	413
Guided Exercise: Deploying Distributed Execution with Automation Mesh	420
Managing Distributed Execution with Automation Mesh	425
Guided Exercise: Managing Distributed Execution with Automation Mesh	434
Quiz: Building a Large-scale Red Hat Ansible Automation Platform Deployment	438
Summary	442
12. Comprehensive Review	443

Comprehensive Review	444
Lab: Deploying and Operating an Automation Mesh	446
Lab: Adding Users and Teams	456
Lab: Uploading Automation Execution Environments to Private Automation Hub	465
Lab: Creating an Inventory Managed as a Project	472
Lab: Configuring Job Templates	484
Lab: Configuring Workflow Job Templates, Surveys, and Notifications	498
Lab: Operating Automation Controller using the API	513
Lab: Backup and Restore Red Hat Ansible Automation Platform	520

Document Conventions

This section describes various conventions and practices that are used throughout all Red Hat Training courses.

Admonitions

Red Hat Training courses use the following admonitions:



References

These describe where to find external documentation that is relevant to a subject.



Note

Notes are tips, shortcuts, or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on something that makes your life easier.



Important

Important sections provide details of information that is easily missed: configuration changes that apply only to the current session, or services that need restarting before an update applies. Ignoring these admonitions will not cause data loss, but might cause irritation and frustration.



Warning

Do not ignore warnings. Ignoring these admonitions will most likely cause data loss.

Inclusive Language

Red Hat Training is currently reviewing its use of language in various areas to help remove any potentially offensive terms. This is an ongoing process and requires alignment with the products and services that are covered in Red Hat Training courses. Red Hat appreciates your patience during this process.

Introduction

Managing Enterprise Automation with Red Hat Ansible Automation Platform

Managing Enterprise Automation with Red Hat Ansible Automation Platform (DO467) is for experienced DevOps engineers or Linux system administrators who want to centralize and control their automation execution at scale and privately share Ansible content collections across their organizations.

Course Objectives

- Installing Red Hat Ansible Automation Platform at scale.
- Managing complex automation workflows with automation controller.
- Installing and managing a private automation hub to provide a single source of truth to collaborate and publish automation content.
- Using automation mesh to provide communication between controller nodes and execution nodes in the data center, in the cloud and at the edge.

Audience

- This course is designed for users who need to create recommended design patterns and operate automation practices at scale, including these roles:
 - DevOps engineers
 - Linux system administrators
 - Other IT professionals with basic expertise using Red Hat Ansible Automation Platform to automate, provision, configure, and deploy applications and services in a Linux environment

Prerequisites

- *Developing Advanced Automation with Red Hat Ansible Automation Platform* (DO374) or equivalent Ansible experience.

Orientation to the Classroom Environment

In this course, the main computer system used for hands-on learning activities is **workstation**. The system called **bastion** must always be running.

All student computer systems have a standard user account, **student**, which has the password **student**. The root password on all student systems is **redhat**.

Classroom Machines

Machine name	IP addresses	ROLE
workstation.lab.example.com	172.25.250.9	Graphical workstation used for system administration
servera.lab.example.com	172.25.250.10	Managed server "A"
serverb.lab.example.com	172.25.250.11	Managed server "B"
serverc.lab.example.com	172.25.250.12	Managed server "C"
serverd.lab.example.com	172.25.250.13	Managed server "D"
servere.lab.example.com	172.25.250.14	Managed server "E"
serverf.lab.example.com	172.25.250.15	Managed server "F"
git.lab.example.com	172.25.250.5	GitLab server
hub.lab.example.com	172.25.250.6	Private automation hub server
controller.lab.example.com	172.25.250.7	Automation controller server
control2.lab.example.com	172.25.250.16	Second automation controller
utility.lab.example.com	172.25.250.8	System with utility services required for classroom
db.lab.example.com	172.25.250.20	Database server
exec1.lab.example.com	172.25.250.21	Execution node
exec2.lab.example.com	172.25.250.22	Execution node
exec3.lab.example.com	172.25.250.17	Execution node
hop1.lab.example.com	172.25.250.24	Hop node
bastion.lab.example.com	172.25.250.254	Bridges classroom and student networks

Introduction

The six managed servers are machines that you use to automate tasks in the hands-on activities. The server `controller.lab.example.com` is the automation controller that you use throughout most of this course, and `hub.lab.example.com` is the private automation hub server. Both of them share a PostgreSQL database server, `db.lab.example.com`. Ansible content is stored under version control in Git repositories on the GitLab CE server, `git.lab.example.com`.

A number of additional machines (a second automation controller, three execution nodes, and the hop node) are used in exercises in which you scale up your Red Hat Ansible Automation Platform installation by using automation mesh.

Several systems in the classroom provide supporting services. Two servers, `content.example.com` and `materials.example.com`, are sources for software and lab materials used in hands-on activities. Information on how to use these servers is provided in the instructions for those activities. These are provided by the `classroom.example.com` virtual machine. Both `classroom` and `bastion` should always be running for proper use of the lab environment.

Controlling Your Systems

You are assigned remote computers in a Red Hat Online Learning (ROLE) classroom. Self-paced courses are accessed through a web application that is hosted at `rol.redhat.com` [<http://rol.redhat.com>]. Log in to this site with your Red Hat Customer Portal user credentials.

Controlling the Virtual Machines

The virtual machines in your classroom environment are controlled through web page interface controls. The state of each classroom virtual machine is displayed on the **Lab Environment** tab.

The screenshot shows a web-based interface for managing a lab environment. At the top, there are tabs for 'Table of Contents', 'Course' (which is selected), and 'Lab Environment'. Below the tabs, there's a section titled '▶ Lab Controls' with instructions about creating and deleting the lab environment. A large button labeled 'CREATE' is present. Below this, a table lists five virtual machines:

VM Name	State	Action	Open Console
bastion	active	ACTION -	OPEN CONSOLE
classroom	active	ACTION -	OPEN CONSOLE
servera	building	ACTION -	OPEN CONSOLE
serverb	building	ACTION -	OPEN CONSOLE
workstation	active	ACTION -	OPEN CONSOLE

Figure 0.1: An example course Lab Environment management page

Machine States

Virtual Machine State	Description
building	The virtual machine is being created.

Introduction

Virtual Machine State	Description
active	The virtual machine is running and available. If it just started, it still might be starting services.
stopped	The virtual machine is completely shut down. On starting, the virtual machine boots into the same state as it had before it was shut down. The disk state is preserved.

Classroom Actions

Button or Action	Description
CREATE	Create the ROLE classroom. Creates and starts all of the virtual machines needed for this classroom. Creation can take several minutes to complete.
CREATING	The ROLE classroom virtual machines are being created. Creates and starts all of the virtual machines that are needed for this classroom. Creation can take several minutes to complete.
DELETE	Delete the ROLE classroom. Destroys all virtual machines in the classroom. All saved work on those systems' disks is lost.
START	Start all virtual machines in the classroom.
STARTING	All virtual machines in the classroom are starting.
STOP	Stop all virtual machines in the classroom.

Machine Actions

Button or Action	Description
OPEN CONSOLE	Connect to the system console of the virtual machine in a new browser tab. You can log in directly to the virtual machine and run commands, when required. Normally, log in to the workstation virtual machine only, and from there, use ssh to connect to the other virtual machines.
ACTION > Start	Start (power on) the virtual machine.
ACTION > Shutdown	Gracefully shut down the virtual machine, preserving disk contents.
ACTION > Power Off	Forcefully shut down the virtual machine, while still preserving disk contents. This is equivalent to removing the power from a physical machine.
ACTION > Reset	Forcefully shut down the virtual machine and reset associated storage to its initial state. All saved work on that system's disks is lost.

At the start of an exercise, if instructed to reset a single virtual machine node, click ACTION > Reset for only that specific virtual machine.

Introduction

At the start of an exercise, if instructed to reset all virtual machines, click **ACTION > Reset** on every virtual machine in the list.

If you want to return the classroom environment to its original state at the start of the course, you can click **DELETE** to remove the entire classroom environment. After the lab has been deleted, you can click **CREATE** to provision a new set of classroom systems.



Warning

The **DELETE** operation cannot be undone. All completed work in the classroom environment is lost.

The Auto-stop and Auto-destroy Timers

The Red Hat Online Learning enrollment entitles you to a set allotment of computer time. To help conserve your allotted time, the ROLE classroom uses timers, which shut down or delete the classroom environment when the appropriate timer expires.

To adjust the timers, locate the two + buttons at the bottom of the course management page. Click the auto-stop + button to add another hour to the auto-stop timer. Click the auto-destroy + button to add another day to the auto-destroy timer. Auto-stop has a maximum of 11 hours, and auto-destroy has a maximum of 14 days. Be careful to keep the timers set while you are working, so that your environment is not unexpectedly shut down. Be careful not to set the timers unnecessarily high, which could waste your subscription time allotment.

Performing Lab Exercises

You might see the following lab activity types in this course:

- A *guided exercise* is a hands-on practice exercise that follows a presentation section. It walks you through a procedure to perform, step by step.
- A *quiz* is typically used when checking knowledge-based learning, or when a hands-on activity is impractical for some other reason.
- An *end-of-chapter lab* is a gradable hands-on activity to help you to check your learning. You work through a set of high-level steps, based on the guided exercises in that chapter, but the steps do not walk you through every command. A solution is provided with a step-by-step walk-through.
- A *comprehensive review lab* is used at the end of the course. It is also a gradable hands-on activity, and might cover content from the entire course. You work through a specification of what to accomplish in the activity, without receiving the specific steps to do so. Again, a solution is provided with a step-by-step walk-through that meets the specification.

To prepare your lab environment at the start of each hands-on activity, run the `lab start` command with a specified activity name from the activity's instructions. Likewise, at the end of each hands-on activity, run the `lab finish` command with that same activity name to clean up after the activity. Each hands-on activity has a unique name within a course.

The syntax for running an exercise script is as follows:

```
[student@workstation ~]$ lab action exercise
```

The `action` is a choice of `start`, `grade`, or `finish`. All exercises support `start` and `finish`. Only end-of-chapter labs and comprehensive review labs support `grade`.

start

The `start` action verifies the required resources to begin an exercise. It might include configuring settings, creating resources, checking prerequisite services, and verifying necessary outcomes from previous exercises. You can perform an exercise at any time, even without performing preceding exercises.

grade

For gradable activities, the `grade` action directs the `lab` command to evaluate your work, and shows a list of grading criteria with a `PASS` or `FAIL` status for each. To achieve a `PASS` status for all criteria, fix the failures and rerun the `grade` action.

finish

The `finish` action cleans up resources that were configured during the exercise. You can perform an exercise as many times as you want.

The `lab` command supports tab completion. For example, to list all exercises that you can start, enter `lab start` and then press the Tab key twice.

Obtaining a Trial Subscription to Red Hat Ansible Automation Platform

Objectives

- Get a trial subscription to Red Hat Ansible Automation Platform and access cloud-based services.

Evaluating Red Hat Ansible Automation Platform

This section provides information on one way to get Red Hat Ansible Automation Platform software for evaluation outside the context of this course.



Important

You do not need to do anything in this section to complete this course.

This section provides information on one way to get access to Red Hat Ansible Automation Platform for your own evaluation and study outside the lab environment.

The necessary software is already available to you in this course's lab environment.

Accessing the Red Hat Hybrid Cloud Console

The Red Hat Hybrid Cloud Console (<https://console.redhat.com>) is a Software-as-a-Service (SaaS) offering that hosts services and applications available to customers.

The platform provides services for several Red Hat products. For example, you can use the OpenShift Cluster application to monitor your clusters and access reporting tools. Insights for Red Hat Enterprise Linux can alert you to security or stability issues with your systems.

For Ansible Automation Platform, the Red Hat Hybrid Cloud Console offers several services:

- The *automation hub* service hosts supported Ansible Content Collections from Red Hat and its partners.
- The *Red Hat Insights for Red Hat Ansible Automation Platform* service, named **Insights** in the web interface, collects data from your automation controller systems and generates graphical reports that can help you better understand automation utilization in your organization.

Authenticating to the Red Hat Hybrid Cloud Console

Use your customer portal username and password to authenticate to the Red Hat Hybrid Cloud Console. To access the Ansible Automation Platform services, you need a valid Ansible Automation Platform subscription.

Creating a Personal Account and Acquiring a Trial Subscription

If you use a corporate account to access the Red Hat Hybrid Cloud Console, then all the users within your organization share your configuration. For example, if you register an automation controller system with Red Hat Insights, then all the users within the organization see that system.

**Warning**

Never use your corporate account for testing purposes. The test configuration you perform might break your existing organization configuration.

If you do not have a customer portal account or do not want to use your corporate account, then create a personal account. Navigate to <https://access.redhat.com/>, click the user icon at the upper left, and then click **Register**.

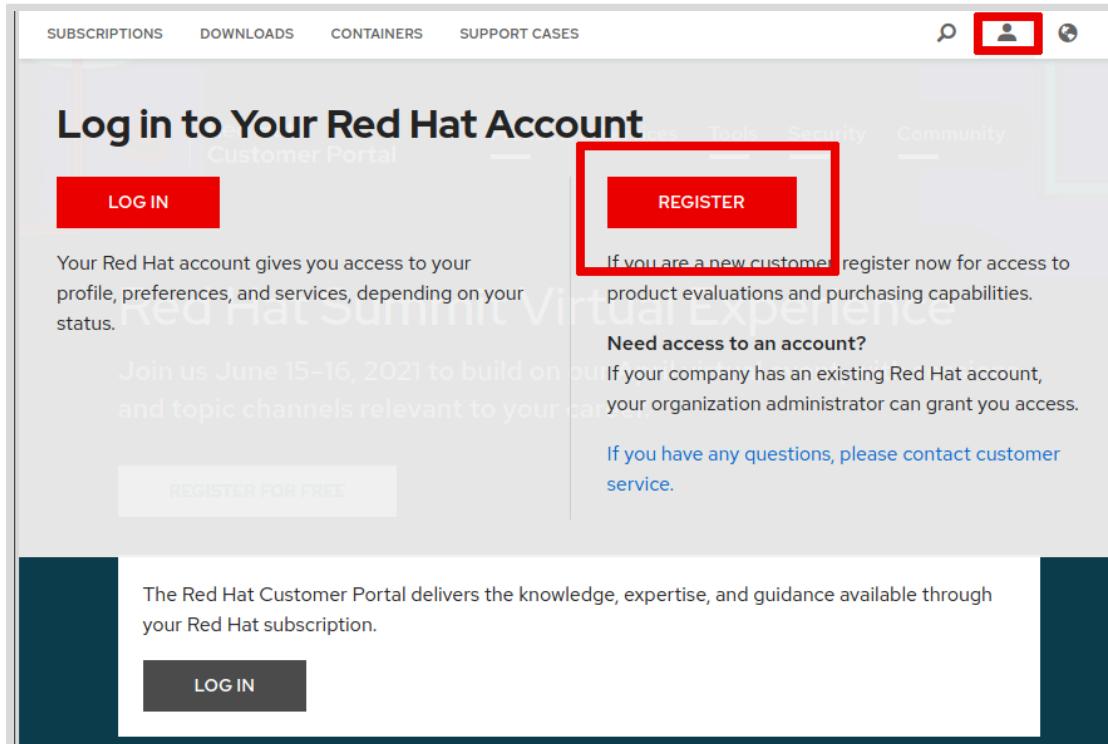


Figure 0.2: Creating a personal account

Select **Personal** for the **Account Type** and then enter your personal information.

Alternatively, you can create your personal account from <https://developers.redhat.com/>. Click **Log In** and then click **Don't have an account? Create one now**. Enter your personal information and then click **Create my Account**.

To get an Ansible Automation Platform subscription, enroll in the Ansible Automation Platform trial. That subscription allows you to access the Ansible Automation Platform services on the Red Hat Hybrid Cloud Console. Navigate to <https://console.redhat.com/ansible> and request an evaluation.

Ansible Automation Platform services requires a valid subscription x



Get analytics and knowledge of your automation, access to certified content, and more with a Red hat Ansible Automation Platform subscription.

Try it Learn More Not now

Figure 0.3: Enrolling in the Ansible Automation Platform trial



Note

It might take up to two hours for your trial subscription to be available. During that period, the preceding trial window displays every time you access the Ansible Automation Platform services.



References

Try Red Hat Ansible Automation Platform

<https://www.redhat.com/en/technologies/management/ansible/try-it>

Chapter 1

Installing Red Hat Ansible Automation Platform

Goal

Explain what Red Hat Ansible Automation Platform is and perform a basic installation of automation controller and private automation hub.

Sections

- Explaining the Red Hat Ansible Automation Platform Architecture (and Quiz)
- Installing Automation Controller and Private Automation Hub (and Guided Exercise)
- Initial Configuration of Automation Controller and Private Automation Hub (and Guided Exercise) (and Quiz)

Explaining the Red Hat Ansible Automation Platform Architecture

Objectives

- Describe the architecture and use cases of Red Hat Ansible Automation Platform.

Red Hat Ansible Automation Platform

Red Hat Ansible Automation Platform is the next evolution in automation from Red Hat. Featuring new tools, services, and capabilities that offer a whole new level of customization and control, it delivers an elevated automation experience that expands the boundaries of what is possible for your enterprise.

Ansible Automation Platform can help you to:

- Accelerate business outcomes.
- Orchestrate across teams.
- Innovate at scale.

Red Hat Ansible Automation Platform Components

Ansible Automation Platform includes various distinct components that together provide a complete and integrated set of automation tools and resources.

Ansible Core

Ansible Core provides the functions used to run Ansible Playbooks.

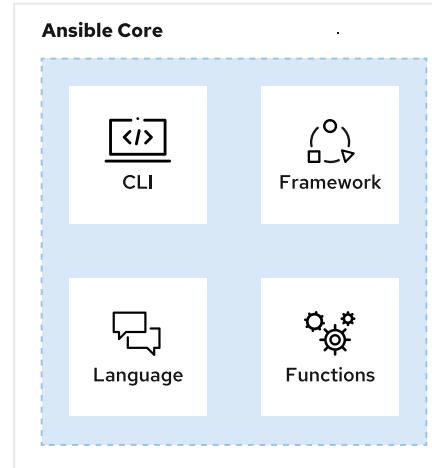


Figure 1.1: Components of Ansible core

CLI

This includes `ansible`, `ansible-playbook`, `ansible-doc` and numerous other command-line utilities for driving and interacting with automation.

Language

Ansible uses YAML to create a compact but powerful set of rules for developing Ansible Playbooks.

Framework

Core functionality can be extended by installing Ansible content collections from either automation hub or Ansible Galaxy.

Functions

This includes conditionals, blocks, includes, loops and other Ansible imperatives.

Ansible Content Collections

Historically, Ansible provided a large number of modules as part of the core package; an approach referred to in the Ansible community as "batteries included". However, with the success and rapid growth of Ansible, the number of modules included with Ansible grew exponentially. This led to certain challenges with support, especially because users sometimes wanted to use earlier or later versions of modules than were packaged with a particular version of Ansible.

The upstream developers decided to reorganize most modules into separate *Ansible Content Collections* made up of related modules, roles, and plug-ins that are supported by the same group of developers. Ansible Core contains a small set of modules provided by the `ansible.builtin` Ansible Content Collection, which is always part of Ansible Core.

This provides users with the flexibility to select different versions of collections, or different sets of collections, based on their needs. It also provides developers with the ability to update their modules on a separate cadence from Ansible itself.

Red Hat Ansible Certified Content Collections are content collections that are officially supported by Red Hat and its partners through Ansible Automation Platform.

Automation Content Navigator

Ansible Automation Platform provides a new top-level tool to develop and test Ansible Playbooks, the *automation content navigator* (`ansible-navigator`). This tool provides the functions of several earlier command-line utilities, including `ansible-playbook`, `ansible-inventory`, and `ansible-config`.

In addition, it separates the control node on which you run Ansible from the automation execution environment that runs it, by running your playbooks in a container. This makes it easier for you to provide a complete working environment for your automation code for deployment to production.

Automation Execution Environments

An *automation execution environment* is a container image that contains Ansible Core, Ansible Content Collections, and any Python libraries, executables, or other dependencies needed to run your playbook.

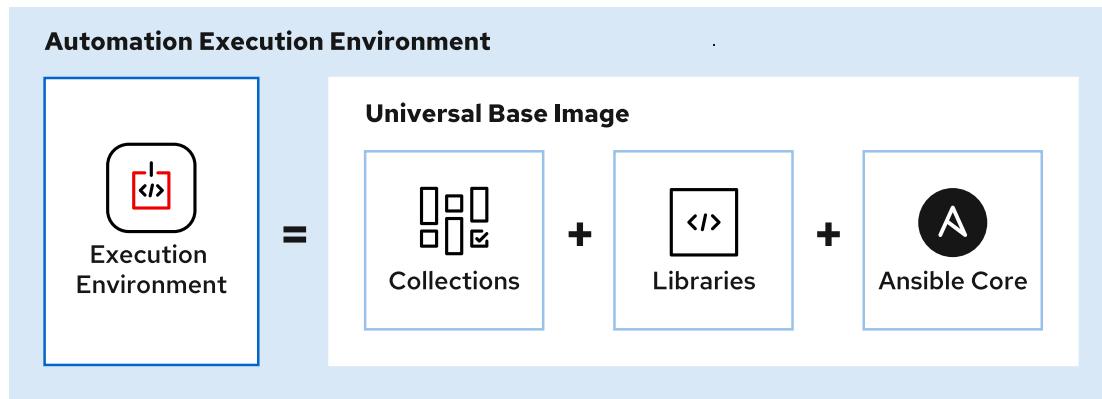


Figure 1.2: Parts of an automation execution environment

When you run a playbook with `ansible-navigator`, you can select an automation execution environment for it to use to run that playbook. When your code is working, you can provide the playbook and the automation execution environment to automation controller (formerly called Red Hat Ansible Tower) and know that automation controller has everything it needs to correctly run your playbook.

The default environment used in Ansible Automation Platform provides Ansible Core and many Red Hat Ansible Certified Content Collections to give you a user experience similar to Ansible 2.9.

Another advantage of automation execution environments is that you can use them to run earlier versions of Ansible as well. Red Hat also supports an automation execution environment that provides Ansible 2.9 for compatibility with earlier versions.

Alternatively, you can use a new tool provided with Ansible Automation Platform called `ansible-builder` to create your own custom execution environments.

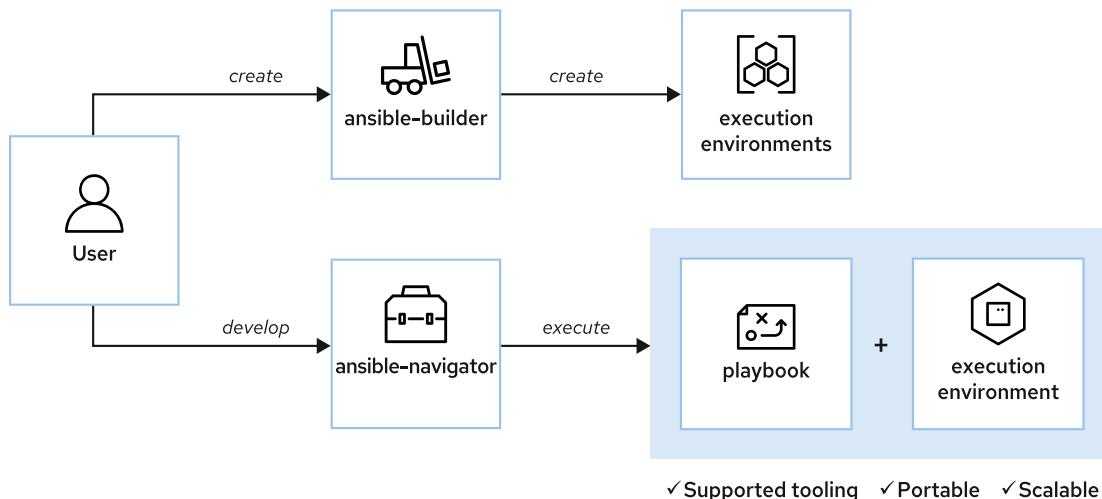


Figure 1.3: Adapting automation execution environments to your needs

Automation Controller

Ansible Automation Platform includes the automation controller as a core component, allowing users to define, operate, scale, and delegate automation across their enterprise.

Automation controller is the control plane for automation and includes a user interface, browsable API, role-based access control, job scheduling, integrated notifications, graphical inventory management, CI/CD integration, and workflow visualizer functions.

Chapter 1 | Installing Red Hat Ansible Automation Platform

Automation controller enables you to manage inventories, launch and schedule workflows, track changes, and integrate into reporting, all from a centralized user interface and REST API.

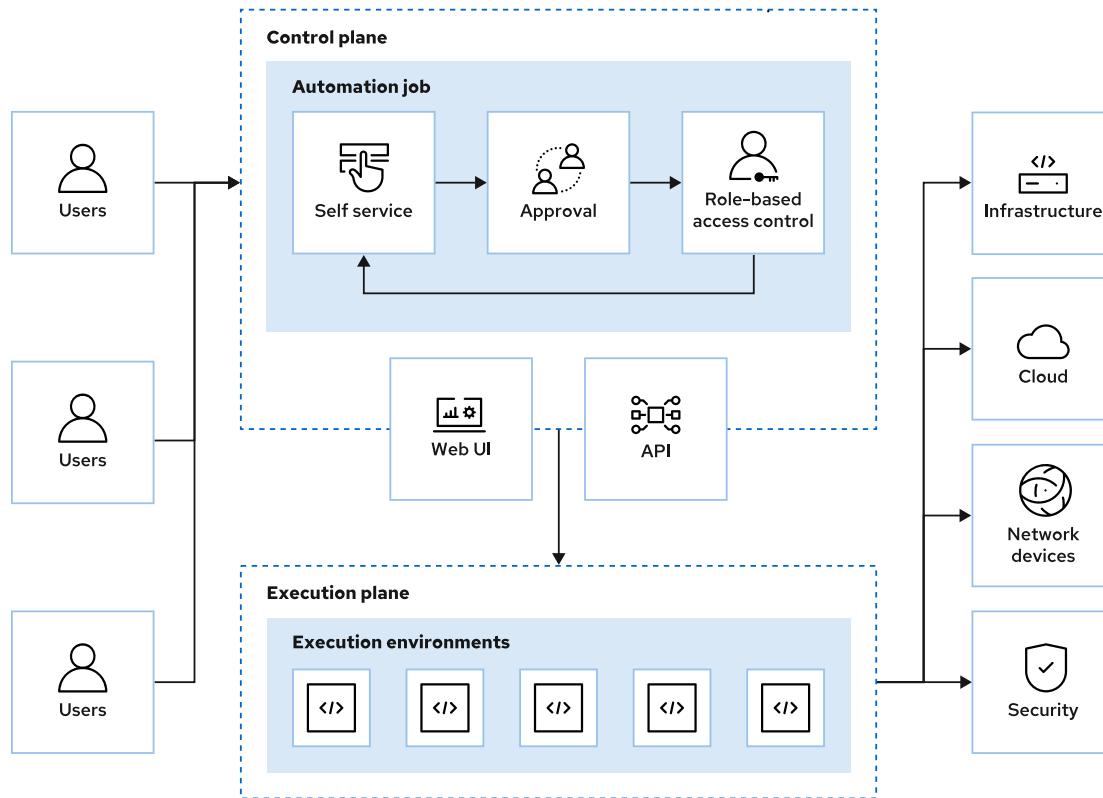


Figure 1.4: Components of automation controller

Automation Hub and Private Automation Hub

You can use <https://console.redhat.com/ansible/automation-hub> to manage and distribute automation content. Automation hub provides access to Red Hat Ansible Certified Content Collections. Although you can download and use Ansible content collections on your local machine, the ee-supported-rhel8 automation execution environment already includes many Ansible content collections.

You can also set up a *private automation hub*. This enables you to create your own curated set of Ansible Content Collections. It also provides a container registry that you can use for distributing your automation execution environments, if you do not already have one.

Private automation hub provides a centralized place for automation controller and developers in your organization to get and use automation content.

Red Hat Insights for Red Hat Ansible Automation Platform

In addition to the hosted Ansible automation hub at <https://console.redhat.com>, other hosted services are available.

Red Hat Insights for Red Hat Ansible Automation Platform and *automation analytics* can help you understand what automation code you are running and whether it is successful. Red Hat Insights can help you identify issues with your managed hosts and generate Ansible Playbooks to remediate those issues. Automation analytics can help you evaluate the positive impact of automation on your organization, by reporting metrics on your use of automation and helping you estimate cost and time savings due to your automation implementation.

Why Use Ansible Automation Platform?

When implementing automation across the enterprise, IT organizations need:

- A standardized way of defining automation workflows into other tools and processes.
- Reliable and scalable automation execution.
- A centralized system that enables auditing.

IT organizations go through three stages of the automation adoption journey: create, manage, and scale. They start by creating the content and progress to centrally managing their content. Ultimately they increase the scale of their automation.

Ansible Automation Platform fulfills these needs and helps IT organizations at every stage of this journey.

Create

Ansible Automation Platform provides certified content, collaboration-ready tools, and multidomain support to help enterprises create and design their automation. The benefits include:

- Providing more than 100 Red Hat Certified Ansible Content Collections maintained by Red Hat and Red Hat Technology Partners.
- Supporting automation execution environments that teams can use to write specific automation.
- Automating use cases for infrastructure, networks, security, DevOps, cloud, and Edge computing.

Manage

Ansible Automation Platform provides tools and features such as single subscription, automation controller, and private automation hub to deploy automation quickly and reliably.

- A single subscription includes everything needed to manage and scale automation, based on stable enterprise code and Red Hat security hardening.
- Automation controller provides various features, such as centralized management, a dashboard, reporting, and logging to facilitate easier automation management.
- Automation hub can host the automation code using collections and execution environments, and provide it to the automation controller.
- An Ansible Automation Platform includes support for platform components and ecosystem integrations with access to Red Hat SMEs, training and professional services as a single source of accountability for broad Red Hat customers.

Scale

You can expand your Ansible Automation Platform by scaling up some of the components, such as automation controller, private automation hub, and automation mesh.

- Flexibility to automate across technology silos, domain-specific tools, geographically distributed deployments, and end-to-end processes.
- Provide automation content and integrations to support deployments across hybrid cloud deployments, including OpenShift, Red Hat Enterprise Linux, and more.



References

Red Hat Ansible Automation Platform

<https://www.redhat.com/en/technologies/management/ansible>

Introducing Ansible Automation Platform 2

<https://www.ansible.com/blog/introducing-ansible-automation-platform-2>

► Quiz

Explaining the Red Hat Ansible Automation Platform Architecture

Choose the correct answers to the following questions:

- ▶ **1. Which four of the following choices are components of Red Hat Ansible Automation Platform? (Choose four.)**
 - a. Ansible Core
 - b. Ansible Content Collections
 - c. Automation content navigator
 - d. Red Hat Ansible for Red Hat OpenShift
 - e. Automation execution environments

- ▶ **2. Which three of the following items belong to an Ansible Content Collection? (Choose three.)**
 - a. Roles
 - b. Plug-ins
 - c. Python virtual environments
 - d. Modules

- ▶ **3. Which collection is part of Ansible Core?**
 - a. ansible.builtin
 - b. ansible.core
 - c. community.mysql
 - d. ansible.posix.firewalld

- ▶ **4. Which three of the following choices are true about ansible-navigator? (Choose three.)**
 - a. It separates the control node from the execution environment.
 - b. It provides an interface for writing Ansible Playbooks.
 - c. It defaults to running playbooks in an execution environment.
 - d. It provides the functionality of the ansible-playbook, ansible-inventory, and ansible-config commands.

- ▶ **5. Which three of the following components are contained in the ee-supported-rhel8 automation execution environment? (Choose three.)**
 - a. An API for automation controller
 - b. Ansible Core
 - c. Ansible Content Collections
 - d. Python libraries
 - e. RPM packages for podman
- ▶ **6. Which four of the following choices are features of the automation controller? (Choose four.)**
 - a. CI/CD integration
 - b. Browsable API
 - c. Distributed Container Registry
 - d. User Interface
 - e. Graphical inventory management
- ▶ **7. Which two of the following choices are hosted services supported with the Red Hat Ansible Automation Platform subscription? (Choose two.)**
 - a. Red Hat Hybrid Cloud Console (console.redhat.com)
 - b. Red Hat OpenShift for Automation
 - c. Red Hat Insights for Red Hat Ansible Automation Platform
 - d. Ansible Galaxy
- ▶ **8. Which artifacts can be stored and distributed in automation hub? (Choose two.)**
 - a. RPM packages
 - b. Ansible Content Collections
 - c. TLS certificates for managed hosts
 - d. Automation execution environments

► Solution

Explaining the Red Hat Ansible Automation Platform Architecture

Choose the correct answers to the following questions:

- ▶ **1. Which four of the following choices are components of Red Hat Ansible Automation Platform? (Choose four.)**
 - a. Ansible Core
 - b. Ansible Content Collections
 - c. Automation content navigator
 - d. Red Hat Ansible for Red Hat OpenShift
 - e. Automation execution environments

- ▶ **2. Which three of the following items belong to an Ansible Content Collection? (Choose three.)**
 - a. Roles
 - b. Plug-ins
 - c. Python virtual environments
 - d. Modules

- ▶ **3. Which collection is part of Ansible Core?**
 - a. ansible.builtin
 - b. ansible.core
 - c. community.mysql
 - d. ansible.posix.firewalld

- ▶ **4. Which three of the following choices are true about ansible-navigator? (Choose three.)**
 - a. It separates the control node from the execution environment.
 - b. It provides an interface for writing Ansible Playbooks.
 - c. It defaults to running playbooks in an execution environment.
 - d. It provides the functionality of the ansible-playbook, ansible-inventory, and ansible-config commands.

- **5. Which three of the following components are contained in the ee-supported-rhel8 automation execution environment? (Choose three.)**
- a. An API for automation controller
 - b. Ansible Core
 - c. Ansible Content Collections
 - d. Python libraries
 - e. RPM packages for podman
- **6. Which four of the following choices are features of the automation controller? (Choose four.)**
- a. CI/CD integration
 - b. Browsable API
 - c. Distributed Container Registry
 - d. User Interface
 - e. Graphical inventory management
- **7. Which two of the following choices are hosted services supported with the Red Hat Ansible Automation Platform subscription? (Choose two.)**
- a. Red Hat Hybrid Cloud Console (console.redhat.com)
 - b. Red Hat OpenShift for Automation
 - c. Red Hat Insights for Red Hat Ansible Automation Platform
 - d. Ansible Galaxy
- **8. Which artifacts can be stored and distributed in automation hub? (Choose two.)**
- a. RPM packages
 - b. Ansible Content Collections
 - c. TLS certificates for managed hosts
 - d. Automation execution environments

Installing Automation Controller and Private Automation Hub

Objectives

- Install automation controller and private automation hub on individual servers.

Planning the Installation

The components of Red Hat Ansible Automation Platform can be deployed in different ways depending on the resources that you have available and the number of hosts that you want to support.

Stand-alone Automation Controller with a Database on the Same Node

The simplest deployment installs a single instance of automation controller with its supporting PostgreSQL database on the same node. This includes a web UI and REST API to operate automation controller. The same instance also executes your Ansible jobs.

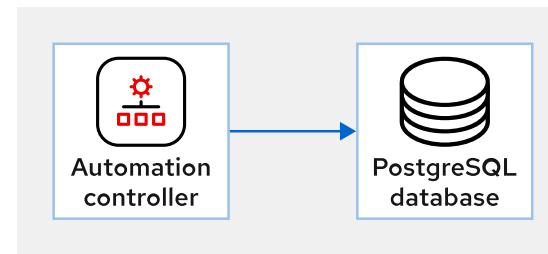


Figure 1.5: Stand-alone automation controller with database on the same node

This installation mode does have scaling limitations, but it is useful for a small environment that does not have many managed systems or high availability requirements.

Stand-alone Private Automation Hub with a Database on the Same Node

You can deploy private automation hub with its database on the same node in a similar way. This is a simple deployment mode, with the same advantages and limitations as the similar automation controller deployment.



Important

You cannot install automation controller and private automation hub on the same node.

Automation Controller and Private Automation Hub with External Database Servers

To improve performance, you can install the PostgreSQL databases used by automation controller and private automation hub on an external database server. By separating the functions to separate servers, you can spread the load and tune the servers for their particular functions.

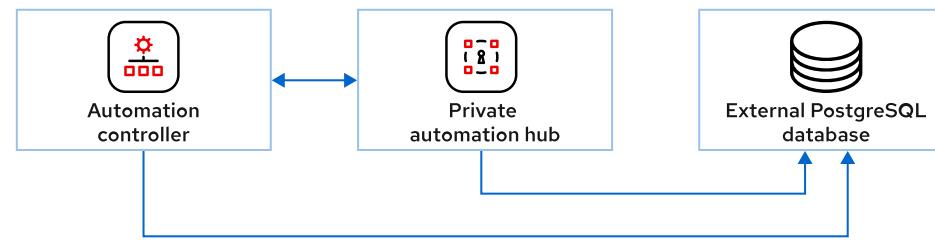


Figure 1.6: Automation controller and private automation hub with external databases

Advanced Deployment Scenarios

More advanced deployment scenarios are available to enable further scaling and availability, but those are not covered in this section. For more information, see the Red Hat Ansible Platform Installation Guide [https://access.redhat.com/documentation/en-us/red_hat_ansible_automation_platform/2.1/html/red_hat_ansible_automation_platform_installation_guide] or other sections of this course.

Installation Requirements

You can install automation controller and private automation hub on systems that run the 64-bit x86_64 version of Red Hat Enterprise Linux, or by using the platform operator in an OpenShift environment. Ensure you satisfy the following requirements before installing automation controller and private automation hub on Red Hat Enterprise Linux systems.

Requirement	Description
Operating System	For Red Hat Enterprise Linux installations, the automation controller and the private automation hub are supported on systems running Red Hat Enterprise Linux 8.4 or later and the 64-bit x86_64 processor architecture.
Web Browser	To connect to the web UI for the automation controller or the private automation hub, use the current supported version of Mozilla Firefox or Google Chrome web browsers.
Memory	The automation controller requires systems with a minimum of 16 GB of RAM. The private automation hub requires systems with a minimum of 8 GB of RAM. The actual memory requirement for the automation controller depends on the maximum number of hosts that it is expected to configure in parallel. This is managed by the <code>forks</code> configuration parameter in the job template or system configuration. You have to increase memory to grow the capacity for running more forks. Red Hat recommends 1 GB of memory per 10 forks, and 2 GB for the automation controller services. If you set the <code>forks</code> parameter to 400, then the automation controller requires 42 GB of RAM.
CPU	The automation controller requires a minimum of 4 CPUs. The private automation hub requires 2 CPUs. You have to increase the number of CPUs to grow the capacity based on forks.

Requirement	Description
Disk Storage	Servers that run either the automation controller or the private automation hub need at least 40 GB of dedicated hard disk space. In the case of the automation controller, 20 GB of this space must be available to the /var directory. If you are installing automation controller or private automation hub with the database on the same server, then consider an additional 20 GB of hard disk space.

Database Storage

You might need more database storage based on the following factors:

- The number of hosts managed by the automation controller.
- The number of Ansible Content Collections and automation execution environments stored by the private automation hub.

Red Hat recommends at least 150 GB for database storage.

The storage volume must have a high baseline input/output operations per second (IOPS) rating (1500 or more).

If you are installing in an Amazon EC2 instance, then use at least the m5. large instance type. Use m4.xlarge if you are managing more than 100 hosts.

Subscription and Support

You need a Red Hat Ansible Automation Platform subscription and you must enable the Red Hat Ansible Automation Platform 2 repository.

Use the following procedure to register your systems and enable the repositories:

- As the root user, use Red Hat Subscription Manager to register each of your systems.

```
[root@host ~]# subscription-manager register
```

- Enable the Red Hat Ansible Automation Platform 2 repository on all systems that need to use packages from that channel. This includes all automation controller and private automation hub nodes. The following example is for a Red Hat Enterprise Linux 8 system.

```
[root@host ~]# subscription-manager repos \
> --enable ansible-automation-platform-2.2-for-rhel-8-x86_64-rpms
```

Installing Red Hat Ansible Automation Platform

The current installation process executes a shell script that runs an Ansible Playbook. This process automatically attempts to install the latest `ansible-core` release package.

One way is to download an archive file from Red Hat Customer Portal (<https://access.redhat.com/>) and unpack it into a directory on your workstation.

The other way is to enable access to the Ansible Automation Platform repository (such as `ansible-automation-platform-2.2-for-rhel-8-x86_64-rpms` for RHEL 8) on one

Chapter1 | Installing Red Hat Ansible Automation Platform

of your systems, and then install the `ansible-automation-platform-installer` RPM package on that system. That RPM unpacks the installer into the `/opt/ansible-automation-platform/installer` directory on that system.

If you download the archive file instead, two different installation files are available:

- Red Hat Ansible Automation Platform installer
- Red Hat Ansible Automation Platform Bundle installer

The Red Hat Ansible Automation Platform installer file is smaller, but requires internet connectivity to download the required repositories, packages, and dependencies.

The Red Hat Ansible Automation Platform Bundle installer file includes an initial set of RPM packages, so that you can install it on systems disconnected from the internet. This file also includes `.tar` files for the supported, minimal, and compatibility automation execution environments. These automation execution environments can be uploaded to the private automation hub or added to the automation controller rather than pulling them down from the Red Hat Ecosystem Catalog available at <https://catalog.redhat.com>.

Use the following procedure to install automation controller and/or automation hub:

- Download and extract the installation `.tar` file, or install the `ansible-automation-platform-installer` RPM package.
- Change to the directory containing the installer files.
- Edit the `inventory` file to configure the installation process.
- Run the `setup.sh` script, which uses Ansible to install and configure the servers.
- Log in to the web UI of the new servers and complete their configuration.

Installing Automation Controller

The following procedure describes how to install a single automation controller and supporting PostgreSQL database server. That database server can be on the same host as the automation controller, but for better scaling and performance might be on a system other than the automation controller.

1. Download the installer from <https://access.redhat.com/downloads/content/480>.
2. Extract the files from the installer and change to the directory containing the extracted contents.

For the Red Hat Ansible Automation Platform installer:

```
[user@host ~]$ tar xzf ansible-automation-platform-setup-2.2.0-6.1.tar.gz  
[user@host ~]$ cd ansible-automation-platform-setup-2.2.0-6.1/
```

For the Red Hat Ansible Automation Platform Bundle installer:

```
[user@host ~]$ tar xzf ansible-automation-platform-setup-bundle-2.2.0-6.1.tar.gz  
[user@host ~]$ cd ansible-automation-platform-setup-bundle-2.2.0-6.1/
```

3. Edit the `inventory` file. At a minimum, specify the fully qualified domain name of the automation controller (`automationcontroller`).

Chapter 1 | Installing Red Hat Ansible Automation Platform

```
[automationcontroller]  
fqdn-for-the-controller-server
```

If you install the PostgreSQL database in an external server, then specify the fully qualified domain name of the database server.

```
[database]  
fqdn-for-the-database-server
```

You can set additional required variables by either modifying the `inventory` file or by creating a separate variables file that you can pass to the `setup.sh` installation script.

**Note**

Creating a separate variables file makes sense if you plan to override additional installation variables that are not found in the `inventory` file, such as the `control_plane_execution_environment` variable or the `global_job_execution_environments` variable.

4. Set the passwords for the automation controller `admin` account (`admin_password`) and for the PostgreSQL database user account (`pg_password`).

For installing the automation controller with the PostgreSQL database on the same server, leave the values for the `pg_host` and `pg_port` variables empty.

```
admin_password='password'  
  
pg_host=''  
pg_port=''  
  
pg_database='awx'  
pg_username='awx'  
pg_password='password'
```

For installing the automation controller's PostgreSQL database on a separate server, specify the fully qualified domain name of the database server (`pg_host`) and the port to communicate with the database server (`pg_port`). Unless you configure it differently, the PostgreSQL database server uses port 5432.

```
admin_password='password'  
  
pg_host='fqdn-for-the-database-server'  
pg_port=5432  
  
pg_database='awx'  
pg_username='awx'  
pg_password='password'
```

**Important**

You should set the passwords to something secure. To prevent the installation from failing, do not use special characters for the database password.

5. Set the `registry_url`, `registry_username`, and `registry_password` variables. The installation script uses these variables to create the `Default Execution Environment Registry Credential` resource in the automation controller.

If you use the default `registry_url` variable value of `registry.redhat.io`, then specify the registry credentials (`registry_username` and `registry_password`) that you use to pull container images from that URL.

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

**Note**

If you plan to host automation execution environment images on the private automation hub, then set the `registry_url` variable to the fully qualified domain name of the private automation hub, such as `hub.lab.example.com`, and set the registry credentials (`registry_username` and `registry_password`) to a user who can pull container images from the registry server.

6. If desired, then specify values for additional variables defined in the `inventory` file, such as variables related to certificates.
7. Run the `setup.sh` installation script.

```
[user@host ansible-automation-platform-setup-2.2.0-6.1]$ ./setup.sh  
...output omitted...  
The setup process completed successfully.  
[warn] /var/log/tower does not exist.  
Setup log saved to setup.log.
```

**Important**

You need root access to the automation controller to run the install playbook. You can achieve this in different ways:

- Creating an `ansible.cfg` configuration file in the same directory as the `setup.sh` installation script with the configuration directives for privilege escalation.
- Defining inventory host variables, inventory group variables, or environment variables before running the `setup.sh` installation script.

In the references at the end of this section you can find more information about privilege escalation.

8. After the installer finishes successfully, connect to the web UI for the automation controller with a web browser. If you did not specify variables for certificates, then the web browser

Chapter 1 | Installing Red Hat Ansible Automation Platform

generates a warning message regarding a self-signed security certificate presented by the automation controller website. Accept the risk and continue.

9. Log in to the automation controller web UI with the **admin** account and the password you set in the **inventory** file.
10. After you log in for the first time, the web UI prompts you to activate your subscription. Activating your subscription is a three step process and the web UI displays your progress through the steps.

In the first step, you can either request a subscription or select your subscription. You have two choices for selecting your subscription:

- Upload a subscription manifest. You can download a subscription manifest from your **Subscription Allocations** page on the customer portal. More details can be found in the references section.
- Enter your Red Hat customer credentials or Red Hat Satellite username and password.

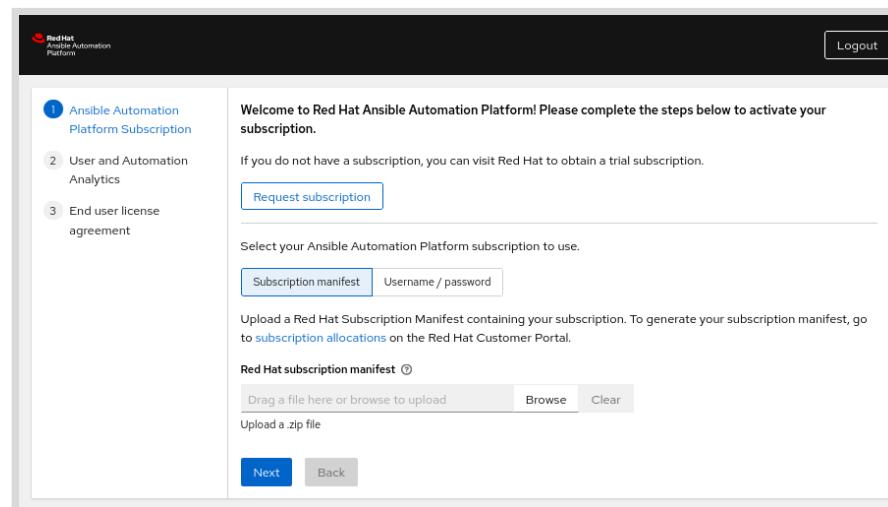


Figure 1.7: Activating your Red Hat subscription

In the second step, you can enable or disable integration with user analytics and automation analytics.

Chapter 1 | Installing Red Hat Ansible Automation Platform

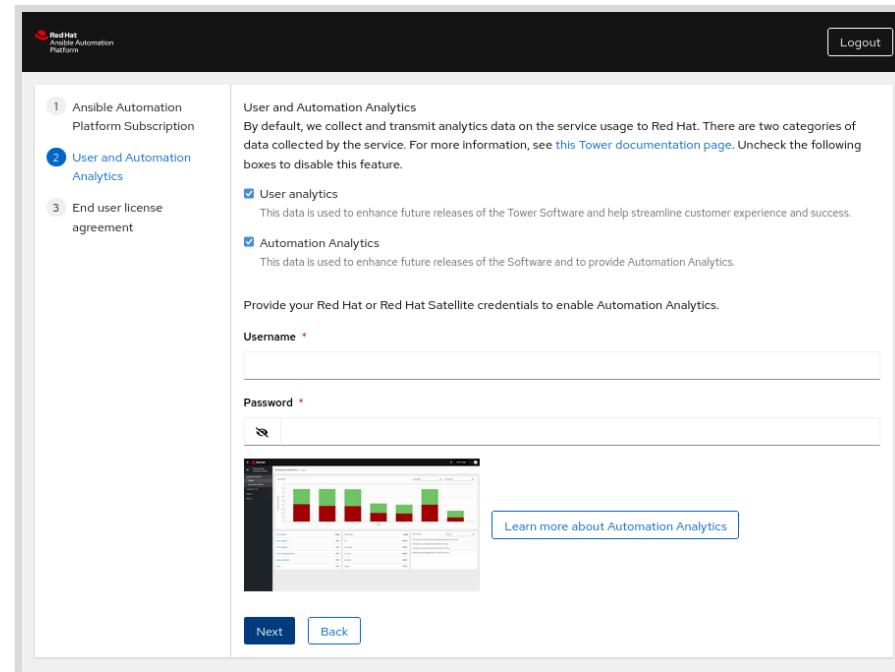


Figure 1.8: Options for user analytics and automation analytics

In the third step, you must review and accept the end user license agreement. After accepting, the web UI displays the automation controller dashboard.

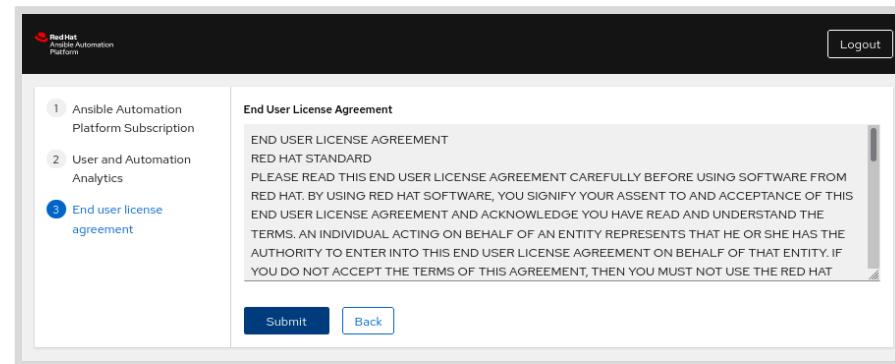


Figure 1.9: End user license agreement

Upcoming sections provide a more detailed orientation to the automation controller interface.

Installing Private Automation Hub

The following procedure describes how to install a single private automation hub and supporting PostgreSQL database server. That database server can be on the same host as the private automation hub, but for better scaling and performance might be on a system other than the private automation hub.

1. The installer for the private automation hub is the same that you downloaded for the automation controller. The first two steps are the same as installing the automation controller. Download the installer and extract the contents.
2. Edit the `inventory` file. At a minimum, specify the fully qualified domain name of the private automation hub (`automationhub`).

Chapter 1 | Installing Red Hat Ansible Automation Platform

```
[automationhub]  
fqdn-for-the-hub-server
```

As with the automation controller, if you install the PostgreSQL database on an external server, then specify the fully qualified domain name of the database server.

```
[database]  
fqdn-for-the-database-server
```

You can set additional required variables by either modifying the `inventory` file or by creating a separate variables file that you can pass to the `setup.sh` installation script.

3. Set the passwords for the private automation hub `admin` account (`automationhub_admin_password`) and for the PostgreSQL database user account (`automationhub_pg_password`).

For installing the private automation hub with the PostgreSQL database on the same node, leave the values for the `automationhub_pg_host` and `automationhub_pg_port` variables empty.

```
automationhub_admin_password='password'  
  
automationhub_pg_host=''  
automationhub_pg_port=''  
  
automationhub_pg_database='automationhub'  
automationhub_pg_username='automationhub'  
automationhub_pg_password='password'
```

For an installation with the PostgreSQL database on a separate server, specify the fully qualified domain name of the database server (`automationhub_pg_host`) and the port used to communicate with the database server (`automationhub_pg_port`). Unless you configure it differently, the PostgreSQL database server uses port 5432.

```
automationhub_admin_password='password'  
  
automationhub_pg_host='fqdn-for-the-database-server'  
automationhub_pg_port=5432  
  
automationhub_pg_database='automationhub'  
automationhub_pg_username='automationhub'  
automationhub_pg_password='password'
```

4. Set the `registry_username` and `registry_password` variables to pull container images from `registry.redhat.io`. If you use a different registry resource, set the `registry_url` variable and the registry credentials accordingly.

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

Chapter 1 | Installing Red Hat Ansible Automation Platform

5. If desired, then specify values for additional variables defined in the inventory file, such as variables related to certificates.
6. Run the `setup.sh` installation script.

```
[user@host ansible-automation-platform-setup-2.2.0-6.1]$ ./setup.sh  
...output omitted...  
The setup process completed successfully.  
[warn] /var/log/tower does not exist.  
Setup log saved to setup.log.
```

**Important**

You need root access to the server to run the playbook to install the private automation hub. Choose the privilege escalation setting you prefer to achieve this access.

7. After the installer finishes successfully, connect to the web UI for the private automation hub with a web browser. If you did not specify variables for certificates, then the web browser generates a warning message regarding a self-signed security certificate presented by the private automation hub website. Accept the risk and continue.
8. Log in to the private automation hub web UI with the `admin` account and the password you set in the `inventory` file.

**Important**

You can install the automation controller and the private automation hub at the same time by editing the `inventory` file with the data for both of them.

Installing this way helps create assets in automation controller and private automation hub automatically as part of the installation. For example:

- Credentials and links between automation controller and private automation hub.
- Download of three execution environments to private automation hub.

You cannot install the automation controller and the private automation hub components on the same server.

If you set the private automation hub as a registry for the automation controller and you install using the Red Hat Ansible Automation Platform Bundle installer, then the three automation execution environments in the bundle upload to the private automation hub during the installation process.

Replacing the CA Certificate

You can configure automation controller, private automation hub, and the database server to use valid certificates. These certificates could be signed by a publicly recognizable certificate authority or by a corporate or enterprise certificate authority that is trusted by your company.

Gathering Certificates and Private Keys

Before replacing certificates, ensure that you have the following files:

Chapter 1 | Installing Red Hat Ansible Automation Platform

- The certificate authority (CA) certificate if using a corporate or enterprise CA. This is not needed for a public certificate authority that is already configured as trusted by Red Hat Enterprise Linux.
- The signed certificate for the automation controller, private automation hub, or database server.
- The associated private key for each signed certificate.

Preparing the Systems

To configure the certificates during the installation process, edit the `inventory` file before running the `setup.sh` installation script. If the certificate was signed by a corporate or enterprise certificate authority, then specify the location of the CA certificate (`custom_ca_cert`) in the `inventory` file.

```
custom_ca_cert=/etc/pki/tls/certs/third-party-ca.pem
```

Depending on the server, uncomment the variables related to certificates and specify the correct values for them:

- For the automation controller, specify the signed certificate (`web_server_ssl_cert`) and the private key (`web_server_ssl_key`).

```
web_server_ssl_cert=/etc/pki/tls/certs/controller.lab.example.com.crt  
web_server_ssl_key=/etc/pki/tls/private/controller.lab.example.com.key
```

- For the private automation hub, specify the signed certificate (`automationhub_ssl_cert`) and the private key (`automationhub_ssl_key`).

```
automationhub_ssl_cert=/etc/pki/tls/certs/hub.lab.example.com.crt  
automationhub_ssl_key=/etc/pki/tls/private/hub.lab.example.com.key
```

- For the PostgreSQL database server, set the value for the `postgres_use_ssl` variable to `True`, and then specify the signed certificate (`postgres_ssl_cert`) and the private key (`postgres_ssl_key`).

```
postgres_use_ssl=True  
  
postgres_ssl_cert=/etc/pki/tls/certs/db.lab.example.com.crt  
postgres_ssl_key=/etc/pki/tls/private/db.lab.example.com.key
```

Trusting Custom CA Certificates

If you specify the `custom_ca_cert` variable, then the installation script configures the servers targeted by the playbook to trust any certificate signed by that certificate authority.

You can configure additional servers in your environment to trust certificates signed by that certificate authority. As the `root` user, copy the CA certificate to the `/etc/pki/ca-trust/source/anchors/` directory and then run the `update-ca-trust` command to add the trusted CA certificate.

```
[root@host ~]$ update-ca-trust
```

**Note**

You can configure Ansible Automation Platform to use custom certificates either before or after the installation.

Updating RPM Packages on Ansible Automation Platform Servers

Do not use `dnf` to update RPM packages on servers installed with Ansible Automation Platform by using the `setup.sh` installation script.

Instead, to upgrade packages on automation controller or private automation hub systems, you must run the `setup.sh` installation script again. Both services must be updated with the installation script to perform database migrations and other operations correctly, and the installation script applies other RPM package updates as well.

**Warning**

Using `dnf` to update RPM packages on automation controller or private automation hub can cause issues with your installation. For more information, see the Knowledgebase article at <https://access.redhat.com/solutions/4566711>.

**References****What is included in Red Hat Ansible Automation Platform subscription?**

<https://access.redhat.com/articles/6057451>

Red Hat Ansible Automation Platform Installation Guide

https://access.redhat.com/documentation/en-us/red_hat_ansible_automation_platform/2.2/html/red_hat_ansible_automation_platform_installation_guide

Deploying Ansible Automation Platform 2.1

https://access.redhat.com/documentation/en-us/reference_architectures/2021/html/deploying_ansible_automation_platform_2.1/index

Ansible Documentation: Understanding Privilege Escalation

https://docs.ansible.com/ansible/latest/user_guide/become.html

Ansible Documentation: Obtaining a subscriptions manifest

https://docs.ansible.com/automation-controller/4.1.1/html/userguide/import_license.html#obtaining-a-subscriptions-manifest

How Do I Perform Security Patching / OS Package Upgrades On Ansible Tower/Automation Controller Nodes Without Breaking Any Ansible Tower/Automation Controller Functionality?

<https://access.redhat.com/solutions/4566711>

► Guided Exercise

Installing Automation Controller and Private Automation Hub

Install a single automation controller in hybrid mode and a single private automation hub on separate servers.

Outcomes

- Install an automation controller, a private automation hub, and an external database for each on a different server.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start install-installation
```

Instructions

- 1. The `lab` command downloads the `certs` and the `controller` directories to the workstation machine. Verify the contents in both directories.
 - 1.1. The `certs` directory contains the signed certificates for the automation controller, the private automation hub and the database servers, their associated private keys, and a GLS Training Classroom CA certificate. Use the `tree` command to list the files in the `certs` directory.

```
[student@workstation ~]$ tree certs
certs
├── classroom-ca.pem
├── control2.lab.example.com.crt
├── control2.lab.example.com.key
├── controller.lab.example.com.crt
├── controller.lab.example.com.key
├── db.lab.example.com.crt
├── db.lab.example.com.key
├── exec1.lab.example.com.crt
├── exec1.lab.example.com.key
├── exec2.lab.example.com.crt
├── exec2.lab.example.com.key
├── exec3.lab.example.com.crt
├── exec3.lab.example.com.key
├── hop1.lab.example.com.crt
├── hop1.lab.example.com.key
└── hub.lab.example.com.crt
```

Chapter1 | Installing Red Hat Ansible Automation Platform

```
└── hub.lab.example.com.key  
0 directories, 17 files
```

You do not use all of these files in this exercise.

- 1.2. The **controller** directory contains the manifest file to use for subscribing to the automation controller. Use the **tree** command to verify this.

```
[student@workstation ~]$ tree controller  
controller  
└── manifest.zip  
0 directories, 1 file
```

- 2. The **lab** command also downloads the Red Hat Ansible Automation Platform Bundle installer. Extract it. For simplicity, rename the extracted directory to **aap2.2-bundle**. Change to the directory that contains the extracted contents.

```
[student@workstation ~]$ tar xzf \  
> ansible-automation-platform-setup-bundle-2.2.0-6.1.tar.gz  
[student@workstation ~]$ mv ansible-automation-platform-setup-bundle-2.2.0-6.1 \  
> aap2.2-bundle  
[student@workstation ~]$ cd aap2.2-bundle
```

- 3. Modify the inventory file to specify the following details:

- The three servers to install
- The passwords for the administrator account on each server
- The container registry for the automation controller
- The variables related to the database
- The variables related to the certificates

- 3.1. Edit the **inventory** file to specify the fully qualified domain name (FQDN) of the automation controller, the private automation hub, and the database servers as follows:

Section of the inventory	FQDN
[automationcontroller]	controller.lab.example.com
[automationhub]	hub.lab.example.com
[database]	db.lab.example.com

- 3.2. Set the values for the following variables related to the automation controller, its container registry, and its database.

Variable	Value
admin_password	redhat
pg_host	db.lab.example.com
pg_password	redhat
registry_url	hub.lab.example.com
registry_username	admin
registry_password	redhat

**Note**

You use the private automation hub as the container registry for the automation controller.

- 3.3. Set the values for the following variables related to the private automation hub and its database:

Variable	Value
automationhub_admin_password	redhat
automationhub_pg_host	db.lab.example.com
automationhub_pg_password	redhat

- 3.4. Set the variables for the signed certificates for the controller, hub, and db servers, the associated private keys, and the GLS Training Classroom CA certificate as follows:

Variable	Value
custom_ca_cert	/home/student/certs/classroom-ca.pem
web_server_ssl_cert	/home/student/certs/controller.lab.example.com.crt
web_server_ssl_key	/home/student/certs/controller.lab.example.com.key
automationhub_ssl_cert	/home/student/certs/hub.lab.example.com.crt
automationhub_ssl_key	/home/student/certs/hub.lab.example.com.key
postgres_use_ssl	True
postgres_ssl_cert	/home/student/certs/db.lab.example.com.crt
postgres_ssl_key	/home/student/certs/db.lab.example.com.key

3.5. When modified, the uncommented content of the `inventory` file displays as follows:

```
[automationcontroller]
controller.lab.example.com
[automationcontroller:vars]
peers=execution_nodes
[execution_nodes]
[automationhub]
hub.lab.example.com
[automationcatalog]
[database]
db.lab.example.com
[sso]
[all:vars]
admin_password='redhat'
pg_host='db.lab.example.com'
pg_port=5432
pg_database='awx'
pg_username='awx'
pg_password='redhat'
pg_sslmode='prefer' # set to 'verify-full' for client-side enforced SSL
registry_url='hub.lab.example.com'
registry_username='admin'
registry_password='redhat'
receptor_listener_port=27199
automationhub_admin_password='redhat'
automationhub_pg_host='db.lab.example.com'
automationhub_pg_port=5432
automationhub_pg_database='automationhub'
automationhub_pg_username='automationhub'
```

Chapter 1 | Installing Red Hat Ansible Automation Platform

```
automationhub_pg_password='redhat'
automationhub_pg_sslmode='prefer'
automationcatalog_pg_host=''
automationcatalog_pg_port=5432
automationcatalog_pg_database='automationservicescatalog'
automationcatalog_pg_username='automationservicescatalog'
automationcatalog_pg_password=''
custom_ca_cert=/home/student/certs/classroom-ca.pem
web_server_ssl_cert=/home/student/certs/controller.lab.example.com.crt
web_server_ssl_key=/home/student/certs/controller.lab.example.com.key
automationhub_ssl_cert=/home/student/certs/hub.lab.example.com.crt
automationhub_ssl_key=/home/student/certs/hub.lab.example.com.key
postgres_use_ssl=True
postgres_ssl_cert=/home/student/certs/db.lab.example.com.crt
postgres_ssl_key=/home/student/certs/db.lab.example.com.key
sso_keystore_password=''
sso_console_admin_password=''
```

Save and close the file.

**Note**

You can use the `~/install-installation/inventory` file for comparison.

▶ **4.** As the root user, run the installation script.

- 4.1. Use the `sudo` command to change to the root user, using `student` as the password. Change to the `/home/student/aap2.2-bundle` directory and execute the `setup.sh` installation script. The script can take up to 20 minutes to complete.

```
[student@workstation aap2.2-bundle]$ sudo -i
[sudo] password for student: student
[root@workstation ~]# cd /home/student/aap2.2-bundle
[root@workstation aap2.2-bundle]# ./setup.sh -e ignore_preflight_errors=true
Using /etc/ansible/ansible.cfg as config file
...output omitted...

PLAY RECAP ****
controller.lab.example.com : ok=289  changed=136  ...  failed=0  ...  ignored=6
db.lab.example.com        : ok=77   changed=31   ...  failed=0  ...  ignored=1
hub.lab.example.com       : ok=213  changed=89  ...  failed=0  ...  ignored=1
localhost                 : ok=3    changed=1    ...  failed=0  ...  ignored=0

The setup process completed successfully.
[warn] /var/log/tower does not exist. Setup log saved to setup.log.
```

**Important**

You run the installation script as the `root` user because the `root` user on the `workstation` machine has root access to the `controller`, `hub`, and `db` machines in this lab environment.

Failure to run the installation script as the `root` user in the classroom environment results in the following error messages:

```
fatal: [controller.lab.example.com]: FAILED! => {"changed": false, "msg": "UID on remote machine is 1000 (0 required). Check Ansible connection and become settings."}
fatal: [hub.lab.example.com]: FAILED! => {"changed": false, "msg": "UID on remote machine is 1000 (0 required). Check Ansible connection and become settings."}
fatal: [db.lab.example.com]: FAILED! => {"changed": false, "msg": "UID on remote machine is 1000 (0 required). Check Ansible connection and become settings."}
```

Because the lab environment does not satisfy the minimum memory requirements for the automation controller and the private automation hub, set the `ignore_preflight_errors` Ansible variable to `true` to ignore prerequisite checks made before installation starts. This should not be set in a production environment.

- 4.2. After the installer finishes successfully, exit from the `root` session.

```
[root@workstation aap2.2-bundle]# exit
```

5. On the `workstation` machine, navigate to the automation controller web UI to finish the initial configuration. Verify that no warning messages are displayed about the authenticity of the certificate for the automation controller.
 - 5.1. Open a browser and navigate to `https://controller.lab.example.com`. Because you configured the certificate for the automation controller in the `inventory` file, and because the `workstation` machine trusts the CA that signed the certificate, no warning message is displayed about the authenticity of the certificate.

**Important**

The machines in the classroom environment trust any certificate signed by the GLS Training Classroom CA. If your organization uses a corporate or an enterprise certificate authority, then you can configure your machines to trust certificates signed by that certificate authority. Copy the file identified by the `custom_ca_cert` variable to the `/etc/pki/ca-trust/source/anchors` directory and run the `update-ca-trust` command.

These steps are not necessary if you use certificates signed by a publicly recognizable certificate authority.

Chapter 1 | Installing Red Hat Ansible Automation Platform

- 5.2. Click the lock icon next to the URL, and then on the pop-up menu, verify this is a secure connection.
Optional. Click Connection secure, More Information, and then click View Certificate to view the certificate.
- 5.3. Log in to the web UI as the admin user with redhat as the password.
- 5.4. The screen automatically displays the option to upload the subscription manifest file. Click Browse and select the manifest.zip file located in the /home/student/controller directory. Click Next.
- 5.5. Do not make any changes to the User and Insights analytics step. Click Next.

**Note**

Although the red asterisk in the Username and Password fields might suggest that it is mandatory to provide your Red Hat credentials, it is not. When you provide the credentials the automation controller can send the collected data to https://cloud.redhat.com. This is not necessary for this exercise.

- 5.6. Click Submit to accept the End User License Agreement.
 - 5.7. The automation controller web UI displays the Dashboard.
- 6. On the workstation machine, navigate to the private automation hub web UI to ensure that the installation was successful. Verify that there is no warning message about the authenticity of the certificate for the private automation hub.
- 6.1. Open a new tab in the browser and navigate to https://hub.lab.example.com. As with the controller server, there is no warning message about the authenticity of the certificate. Click the lock next to the URL, and then verify the secure connection on the pop-up menu.
 - 6.2. Log in to the web UI as the admin user with redhat as the password.
 - 6.3. The private automation hub web UI displays the dashboard.
- 7. As the postgres user on the database server, verify the existence of the awx and the automationhub databases.
- 7.1. Log in to the db.lab.example.com database server as the student user. Use the sudo command to become the postgres user, and then list the databases using the psql command. Verify that both databases appear in the list.

```
[student@workstation aap2.2-bundle]$ ssh student@db.lab.example.com
Warning: Permanently added 'db.lab.example.com' (ECDSA) to the list of known
hosts.
...output omitted...
[student@db ~]$ sudo su - postgres
[sudo] password for student: student
[postgres@db ~]$ psql -U postgres -l
                                         List of databases
   Name    |  Owner   | Encoding | Collate   |   Ctype    | ...
-----+-----+-----+-----+-----+-----+
automationhub | automationhub | UTF8      | en_US.UTF-8 | en_US.UTF-8 | ...
```

Chapter1 | Installing Red Hat Ansible Automation Platform

awx	awx	UTF8	en_US.UTF-8	en_US.UTF-8	...
postgres	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	...
template0	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	...
template1	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	...
(5 rows)					

7.2. Exit from the `postgres` session and from the `db` machine.

```
[postgres@db ~]$ exit
logout
[student@db ~]$ exit
logout
Connection to db.lab.example.com closed.
```

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish install-installation
```

Initial Configuration of Automation Controller and Private Automation Hub

Objectives

- Perform initial configuration of automation controller and private automation hub and explore the services' web UIs.

Configuration Overview

Red Hat Ansible Platform includes private automation hub, which allows an organization to provide curated Ansible content. You might use this content in your playbooks or to create customized automation execution environments. Because automation controller can also use this same content, a playbook developed and tested with `ansible-navigator` can run without modification on automation controller.

One of the first steps after installing private automation hub is to configure it by uploading and publishing content. This is especially important if automation controller expects to be able to download automation execution environments from private automation hub.

Aside from activating its subscription, automation controller does not require any initial configuration. If you configured automation controller to use automation execution environments provided by private automation hub, then you might perform some basic functionality tests to verify that Red Hat Ansible Platform works as expected.

Making Automation Execution Environments Available from Private Automation Hub

Private automation hub includes a simple container registry. Administrators of private automation hub can synchronize container images from remote repositories, such as from the Red Hat Ecosystem Catalog, and can also manually upload container images.



Note

Although the web interface navigation for the included container registry is titled **Execution Environments**, it is not restricted to hosting automation execution environment images.

Synchronizing Automation Execution Environments

If you plan to host automation execution environments on private automation hub, then you might add one or more remote registries to private automation hub. You can choose how frequently you synchronize images from the remote registries, and you might even remove images from private automation hub. By hosting container images on private automation hub, you have a standard location for them to help control which images are used in your organization.

Use the following procedure to add the Red Hat Ecosystem Catalog remote registry to a private automation hub.

1. Log in to the private automation hub web UI as a user that has permissions to manage remote registries.

Chapter 1 | Installing Red Hat Ansible Automation Platform

2. Navigate to **Execution Environments > Remote Registries** and then click **Add remote registry**.
3. Enter a name, such as **Red Hat Ecosystem Catalog**.
4. Enter the remote registry URL, such as <https://registry.redhat.io>.
5. Enter a valid username and password combination for a user that has permission to download container images from the URL.
6. Click **Save** to create the remote registry.

You can selectively synchronize specific container repositories.

- Navigate to **Execution Environments > Remote Registries** to display the configured remote registries.
- Find the row for the remote registry, such as **Red Hat Ecosystem Catalog**, and then select **Index execution environments** from the vertical ellipsis icon.
- Navigate to **Execution Environments > Execution Environments** to display the container repositories available for synchronization.
- To synchronize a specific container repository, such as the **ansible-automation-platform-22/ee-supported-rhel8** container repository, select **Sync from registry** from the vertical ellipsis icon for the desired container repository row.

After synchronization completes, you can view the available images. Navigate to **Execution Environments > Execution Environments**, click the name of a container repository, and then select the **Images** tag. Some images have more than one tag. For Ansible Automation Platform 2.1, the most recent image has several tags, such as **1.0** and **latest**.

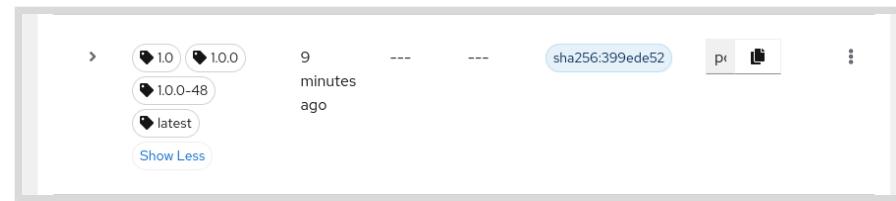


Figure 1.10: Multiple image tags

Rather than synchronizing individual container repositories, you can synchronize *all* Ansible Automation Platform container repositories (this includes versions of the Ansible Automation Platform 2.0, 2.1, and 2.2 automation execution environment container images).

- From the **Remote Registries** page, find the row for the **Red Hat Ecosystem Catalog** remote registry and select **Index execution environments** from the vertical ellipsis icon.
- Click **Sync from registry** to start the synchronization.

Although the task completes quickly, it also creates additional tasks to synchronize each container registry. These additional tasks are visible by navigating to **Task Management**.

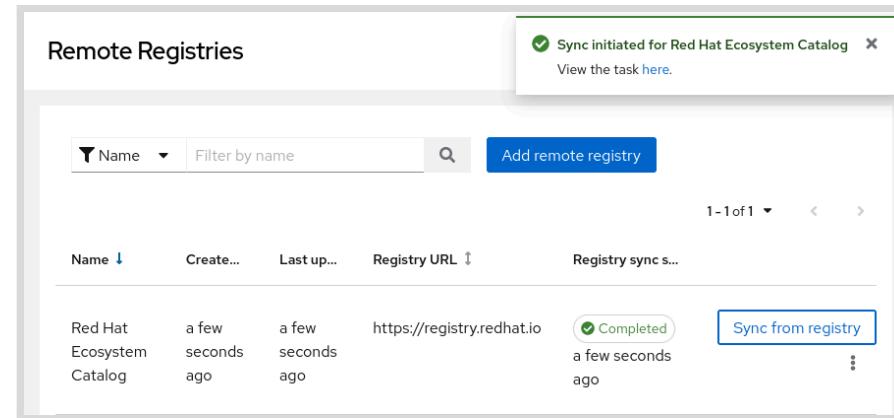


Figure 1.11: Synchronizing a remote registry

Manually Adding Container Images

Organizations might choose to manually upload container images to private automation hub, such as custom automation execution environments or other approved container images.

The `skopeo` command (provided by the `skopeo` RPM) can perform several tasks on container images, such as inspecting both local and remote images, uploading a local archive to a remote repository, and copying container images from one repository to another without needing to download it to the local machine.

If a remote repository requires authentication, then use the `skopeo login` command to log in to the container registry. This is typically required in order to push or copy a container image to a remote repository. Authentication might also be required to inspect or copy a remote repository image.

```
[user@host ~]$ skopeo login registry.redhat.io
Username: your-username
Password: your-password
Login Succeeded!
```

Use the `skopeo inspect` command to inspect details of a container image. For example, you might use the command to inspect the `ee-minimal-rhel8:latest` container image, available from the Red Hat Ecosystem Catalog.

```
[user@host ~]$ skopeo inspect \
> docker://registry.redhat.io/ansible-automation-platform-22/ee-minimal-rhel8
...output omitted...
```

Use the `skopeo copy` command to copy a container image (either local or remote) to a remote registry. When copying from one remote registry to another, you might need to log in to both registries. If you prefer not to log in, then the `skopeo copy` command has options for specifying credentials from the command line. The following example copies a local archive to a remote registry and adds the `1.0` tag to the container image.

```
[user@host ~]$ skopeo copy docker-archive:///tmp/ee-29-rhel8.tar \
> docker://hub.lab.example.com/ansible-automation-platform-22/ee-29-rhel8:1.0
...output omitted...
```

Managing Container Repositories, Images, and Tags

Synchronizing content from a remote registry can download more content than you need. If you do not need a container repository, then you can delete it. For example, you might not need the Ansible Automation Platform 2.0 container repositories that appear when you synchronize all of container repositories from the <https://registry.redhat.io> remote registry.

To delete a container registry:

1. Navigate to **Execution Environments > Execution Environments** to display all container repositories.
2. Select **Delete** from the vertical ellipsis icon on the container repository row that you wish to delete, such as `ansible-automation-platform-20-early-access/ee-29-rhel8`.
3. Confirm your decision to delete the container repository.

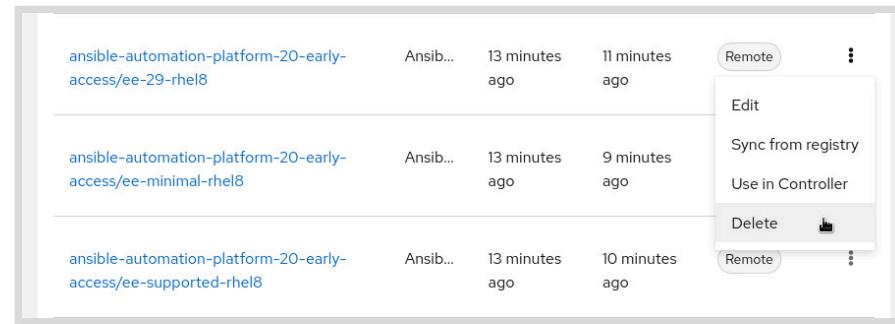


Figure 1.12: Deleting a container repository

An individual container repository can contain many container images, especially if it is synchronized from a remote registry. To delete an individual container image:

1. Navigate to **Execution Environments > Execution Environments** to display all container repositories.
2. Click the name of a container repository, such as `ansible-automation-platform-21/ee-supported-rhel8`.
3. Identify a container image to delete, such as `1.0.1-15-source`, and select **Delete** from the vertical ellipsis icon.
4. Confirm your decision to delete the container image.

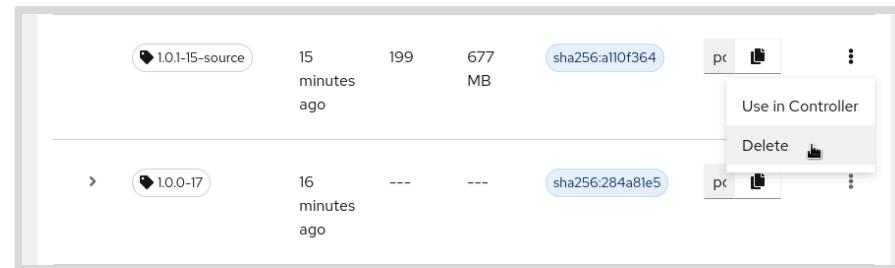


Figure 1.13: Deleting a container image

If you manually upload container images, then you might need to add new tags or remove tags from container images. Container images can have more than one tag, but the tags used within a container repository must be unique. For example, only one container image in a container repository can have the `latest` tag or the `1.0` tag.

Chapter 1 | Installing Red Hat Ansible Automation Platform

Use the `skopeo inspect` command to display details about a container image. You can filter the output with the `--format` option to only display specific information, such as the version and release of the image. When not specified, the container image tag defaults to the `latest` tag. In the following example, the version and release of the latest container image available from the Red Hat Ecosystem Catalog is `1.0.1-27`.

```
[user@host ~]$ skopeo inspect \
> --format "{{ .Labels.version }}-{{ .Labels.release }}" \
> docker://registry.redhat.io/ansible-automation-platform-22/ee-supported-rhel8
1.0.0-113
```

The `1.0.0-113` matches a tag visible from the Red Hat Ecosystem Catalog for the `ansible-automation-platform-22/ee-supported-rhel8` container repository.

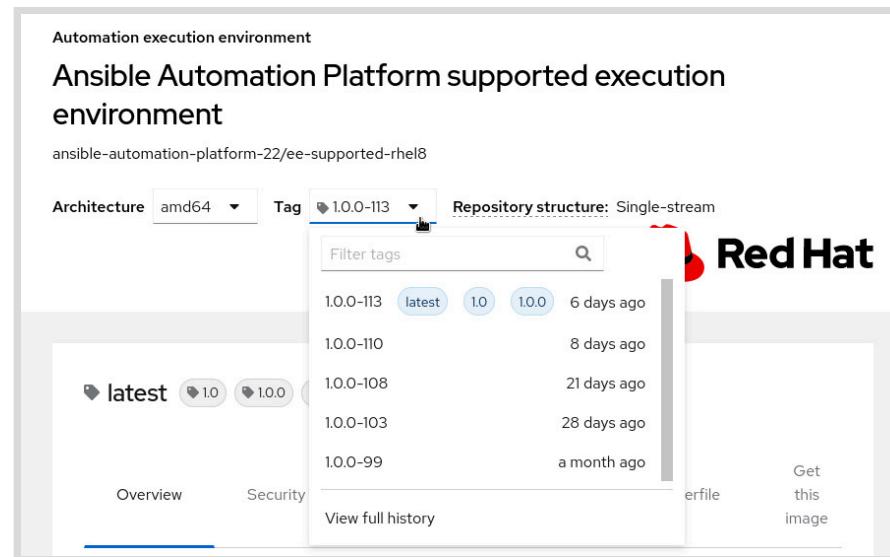


Figure 1.14: Tags visible from the Red Hat Ecosystem Catalog

You can manage container image tags from the private automation hub web UI.

1. Navigate to **Execution Environments** > **Execution Environments** to display all container repositories.
2. Click the name of a container repository, such as `ansible-automation-platform-22/ee-supported-rhel8`.
3. Find a container name to manage and then select **Manage tags** from the vertical ellipsis icon.
4. Add or remove tags and then click **Save**.

Synchronizing Ansible Content Collections

To provide Ansible Content Collections from your private automation hub, you must synchronize or upload them from another source. These content collections can include:

- Supported Red Hat Certified Ansible Content Collections from the cloud-based automation hub at <https://console.redhat.com/ansible/automation-hub>.
- Ansible Content Collections published by the upstream community at Ansible Galaxy, <https://galaxy.ansible.com>.
- Ansible Content Collections that you manually upload.

Synchronizing Red Hat Certified Ansible Content Collections

Synchronizing Ansible Content Collections from automation hub (available at <https://console.redhat.com/ansible/automation-hub>) is the most efficient way to make Red Hat Certified Content Collections available to your private automation hub. Red Hat reviews, maintains, updates, and fully supports those collections. By using Red Hat Ansible Certified Content Collections instead of the collections from Ansible Galaxy, you benefit from the support from Red Hat.

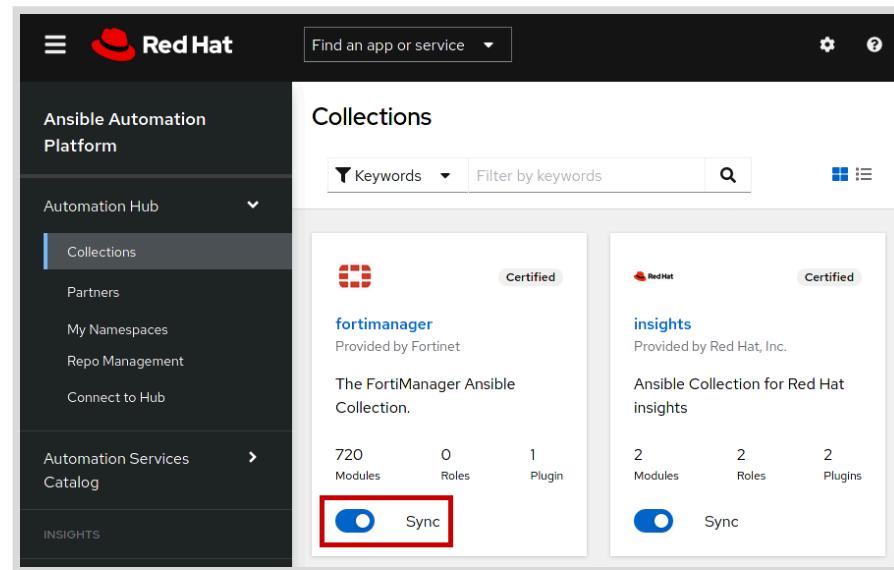


Figure 1.15: Allowing collection synchronization

Within automation hub, navigate to **Collections**. If you are an automation hub administrator for your organization, then each collection has a **Sync** button. When enabled, this button allows the collection to be synchronized with your private automation hub. By synchronizing, you can provide a curated set of collections to your users from your private automation hub.



Warning

Never use your corporate account for testing purposes. The test configuration you perform might corrupt your existing organization configuration.

If you do not have a customer portal account or do not want to use your corporate account, then create a personal account. Navigate to <https://access.redhat.com/>, click the user icon at the upper left, and then click **Register**.

The following process synchronizes Ansible Content Collections (that have been marked for synchronization) from automation hub to private automation hub.

1. Log in to the private automation hub web UI as a user that has permissions to manage remote collections.
2. Navigate to **Collections > Repository Management** and then click the **Remote** tab.
3. On the **rh-certified** row, select **Edit** from the vertical ellipsis icon.
4. Enter your token for downloading content from automation hub. You can copy your automation hub token from <https://console.redhat.com/ansible/automation-hub/token>.
5. Click **Save** to save your changes.

Chapter 1 | Installing Red Hat Ansible Automation Platform

- From the **Repo Management** page, click **Sync** to synchronize Ansible Content Collections from automation hub. If successful, then the **Sync status** column eventually displays a status of **Completed**.

Remote repository synchronization creates the relevant namespaces for the Ansible Content Collections.

Synchronizing Ansible Content Collections from Ansible Galaxy

You can also synchronize Ansible Content Collections from Ansible Galaxy.



Important

Content collections that are published on Ansible Galaxy are managed by the upstream community and are not Red Hat Certified Ansible Content Collections officially supported by Red Hat. Content provided through the cloud-based automation hub at <https://console.redhat.com/ansible/automation-hub> is officially supported.

- Log in to the private automation hub web UI as a user who can manage remote collections.
- Navigate to **Collections > Repository Management** and then click the **Remote** tab.
- On the **community** row, click **Configure**.
- Click **Browse** and locate a **requirements .yml** file containing a list of Ansible Content Collections to synchronize.
- Ansible Galaxy does not require the **Username** and **Password** fields. You can remove any populated values.
- Click **Save** to save your changes.
- From the **Repo Management** page, click **Sync** to synchronize Ansible Content Collections from Ansible Galaxy. If successful, then the **Sync status** column eventually displays a status of **Completed**.

Repo Management				
Local	Remote	Last updated	Last synced	Sync status
	community	community	4 minutes ago	a few seconds ago ✓ Completed
	rh-certified	rh-certified	7 minutes ago	a few seconds ago ✓ Completed

Figure 1.16: Successful remote repository synchronization

Manually Adding Ansible Content Collections

You can manually add Ansible Content Collections to private automation hub, such as custom Ansible Content Collections created by your organization. Organizations in a disconnected

Chapter 1 | Installing Red Hat Ansible Automation Platform

environment can also use this method to add previously downloaded Ansible Content Collections to their private automation hub.

Both automation hub (available at <https://console.redhat.com/ansible/automation-hub>) and Ansible Galaxy (available at <https://galaxy.ansible.com>) allow downloading Ansible Content Collections as TAR files. After locating an Ansible Content Collection, select a specific version and then click the **Download tarball** link. If you do not select a version, then you download the latest version of the Ansible Content Collection.

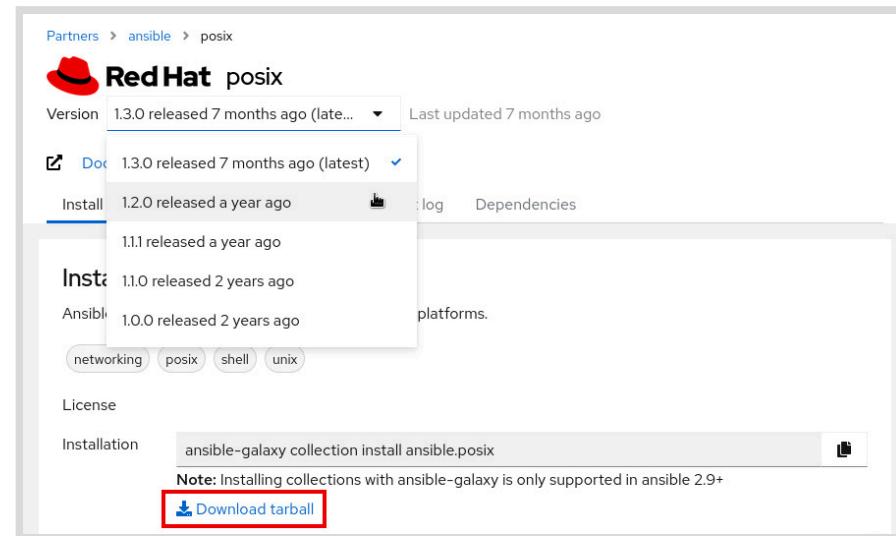


Figure 1.17: Downloading an Ansible Content Collection

The following procedure creates an Ansible Content Collection namespace.

1. Log in to the private automation hub web UI as a user who can manage collections and collection namespaces.
2. Navigate to **Collections > Namespaces** and then click **Create**.
3. Specify a name for the namespace. This name must match the first part of the name of the archive file that you intend to upload. For example: use **ansible** as the namespace name for the **ansible-posix-1.3.0.tar.gz** archive; use **community** as the namespace name for the **community-crypto-2.2.2.tar.gz** archive; and use **containers** as the namespace name for the **containers-podman-1.9.1.tar.gz** archive.
4. (*Optional*) Select one or more existing groups as namespace owners. Namespace owners can change the namespace and can upload to the namespace.
5. Click **Create** to create the new namespace.

The following procedure uploads an Ansible Content Collection to an existing namespace.

1. Log in to the private automation hub web UI as a user who can upload collections.
2. Navigate to **Collections > Namespaces** and then click the **View collections** link for the desired namespace.
3. Click **Upload collection**.
4. Click **Select file** and then browse to the archive file that you want to upload.
5. Click **Upload** to upload the Ansible Content Collection.

Chapter 1 | Installing Red Hat Ansible Automation Platform

By default, private automation hub requires a user to approve uploaded Ansible Content Collections. Use the following procedure to approve or reject an uploaded Ansible Content Collection.

1. Navigate to **Collections > Approval** and then locate the collection.
2. *(Optional)* Select **View Import Logs** from the vertical ellipsis icon.
3. Click the vertical ellipsis icon and choose either **Approve** or **Reject**.

You can configure private automation hub to automatically approve uploaded Ansible Content Collections. Set the `automationhub_require_content_approval` variable to `False` in the installation `inventory` file. If you change this value after installation, then run the `setup.sh` installation script again to apply the change.

Testing Basic Automation Controller Functionality

When you install automation controller, it is initially configured with a **Demo** project, which includes resources that you can use to perform basic functionality tests. Performing these tests can help verify that automation controller is working as expected.

The Demo Project

The **Demo Project** project resource pulls content from the Git repository located at `https://github.com/ansible/ansible-tower-samples`. Navigate to **Resources > Projects** and click the **Sync Project** icon.

Initiating a project synchronization verifies two things:

1. It verifies that automation controller can download the **Control Plane Execution Environment** defined at **Administration > Execution Environments**. If not already present on automation controller, then automation controller uses the **Default Execution Environment Registry Credential** credential (defined at **Resources > Credentials**) to pull the container image defined by the **Control Plane Execution Environment** resource.

In the classroom environment, the initial project synchronization failed because the **Control Plane Execution Environment** container image did not initially exist on private automation hub.

2. The project synchronization verifies that the **Control Plane Execution Environment** container image can clone the project source control repository to automation controller.

Default Execution Environment Registry Credential

The installer uses the `registry_url`, `registry_username`, and `registry_password` variables in the installer `inventory` file to create the **Default Execution Environment Registry Credential** container registry credential. Navigate to **Resources > Credentials** on your automation controller to display existing credentials. The automation controller web UI does not allow you to make changes to the **Default Execution Environment Registry Credential** container registry credential. If you need to make a change, then modify the relevant inventory variables and run the `setup.sh` installation script again.

The Demo Credential

The installer creates the **Demo Credential** machine credential. The **Demo Credential** machine credential is not immediately useful by default. Recall that a machine credential allows automation controller to connect to machines.

You might modify the **Demo Credential** machine credential to use a valid username, and either a password or SSH private key for authentication, so that it can connect to machines in your environment.

The Demo Inventory

The installer creates the **Demo Inventory** inventory resource, which includes the **localhost** host resource. When you run a playbook in an automation execution environment with automation controller, **localhost** refers to the automation execution environment container, not the automation controller that launched the job.

Navigate to **Resources > Inventories** to display existing inventories. Navigate to **Resources > Hosts** to display existing hosts.

To test functionality, you might consider adding a host from your environment to the **Demo Inventory** inventory. If so, then you should also update the **Demo Credential** machine credential. You might also delete the **localhost** host from the **Demo Inventory** inventory.

The Demo Job Template

The **Demo Job Template** job template uses the `hello_world.yml` playbook from the **Demo Project** project. The job template targets all hosts defined in the **Demo Inventory** and uses the **Demo Credential** machine credential. Because **localhost** in the default inventory is configured to use a local connection, it is not necessary to use the **Demo Credential** machine credential when targeting just this host.

Navigate to **Resources > Templates** to display existing templates. Launching the **Demo Job Template** template initiates a job. The `hello_world.yml` playbook uses the `debug` module to print a `Hello, World!` message.



References

For more information about private automation hub namespaces, refer to *Curating Collections Using Namespaces in Automation Hub* at
https://access.redhat.com/documentation/en-us/red_hat_ansible_automation_platform/2.1/html-single/curating_collections_using_namespaces_in_automation_hub/index

For more information about publishing content to private automation hub, refer to *Publishing Proprietary Content Collections in Automation Hub* at
https://access.redhat.com/documentation/en-us/red_hat_ansible_automation_platform/2.1/html-single/publishing_proprietary_content_collections_in_automation_hub/index

For more information about managing containers in private automation hub, refer to *Managing Containers in Private Automation Hub* at
https://access.redhat.com/documentation/en-us/red_hat_ansible_automation_platform/2.1/html-single/managing_containers_in_private_automation_hub/index

For more information about using Ansible Content Collections in private automation hub, refer to *Managing Red Hat Certified and Ansible Galaxy Collections in Automation Hub* at
https://access.redhat.com/documentation/en-us/red_hat_ansible_automation_platform/2.1/html-single/managing_red_hat_certified_and_ansible_galaxy_collections_in_automation_hub/index

For more information, refer to the *Automation Controller User Guide* at
<https://docs.ansible.com/automation-controller/latest/html/userguide/>

skopeo -login(1), skopeo -inspect(1), and skopeo -copy(1) man pages

► Guided Exercise

Initial Configuration of Automation Controller and Private Automation Hub

Navigate the web UI of private automation hub and automation controller, and launch a test job.

Outcomes

- Upload automation execution environments to private automation hub.
- Create namespaces for automation content collections on private automation hub.
- Upload automation content collections to private automation hub.
- Modify existing automation controller resources.
- Launch an automation controller job template.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command ensures that private automation hub and automation controller are installed.

```
[student@workstation ~]$ lab start install-configuration
```

Instructions

- 1. Upload the three provided automation execution environments to private automation hub.

- 1.1. From a terminal window, change into the `/home/student/certified-EEs` directory.

```
[student@workstation ~]$ cd ~/certified-EEs/
```

- 1.2. Use the `skopeo login` command to log in to private automation hub, using `admin` as the username and `redhat` as the password.

```
[student@workstation certified-EEs]$ skopeo login hub.lab.example.com
Username: admin
Password: redhat
Login Succeeded!
```

- 1.3. Use the `skopeo copy` command to upload the `compatibility`, `minimal`, and `supported` automation execution environment image archives. Red Hat uses `compatibility` to refer to the `ee-29-rhel8` automation execution

Chapter 1 | Installing Red Hat Ansible Automation Platform

environment because that execution environment contains Ansible 2.9 (a version that is compatible with many earlier playbooks). Upload the image archives to the hub.lab.example.com server using the ansible-automation-platform-22 namespace and the latest tag.

Set the HUB and AAP shell variables to reduce typing in the following skopeo commands.

```
[student@workstation certified-EEs]$ HUB="hub.lab.example.com"
[student@workstation certified-EEs]$ AAP="ansible-automation-platform-22"

[student@workstation certified-EEs]$ skopeo copy \
> docker-archive:ee-29-rhel8.tgz docker://${HUB}/${AAP}/ee-29-rhel8:latest
...output omitted...

[student@workstation certified-EEs]$ skopeo copy \
> docker-archive:ee-minimal-rhel8.tgz \
> docker://${HUB}/${AAP}/ee-minimal-rhel8:latest
...output omitted...

[student@workstation certified-EEs]$ skopeo copy \
> docker-archive:ee-supported-rhel8.tgz \
> docker://${HUB}/${AAP}/ee-supported-rhel8:latest
...output omitted...
```



Note

This example uses the skopeo command, but you can use the podman command if you prefer. Assuming that you logged in to private automation hub with the podman login command, you can upload an archive with the following commands. When not specified, the podman tag and podman push commands default to using the latest tag.

```
[student@workstation certified-EEs]$ podman load -i ee-supported-rhel8.tgz
...output omitted...
[student@workstation certified-EEs]$ podman tag \
> registry.redhat.io/ansible-automation-platform-22/ee-supported-rhel8 \
> hub.lab.example.com/ansible-automation-platform-22/ee-supported-rhel8
...output omitted...
[student@workstation certified-EEs]$ podman push \
> hub.lab.example.com/ansible-automation-platform-22/ee-supported-rhel8
...output omitted...
```

- ▶ 2. Verify that the automation execution environments are available from private automation hub.

- 2.1. Navigate to <https://hub.lab.example.com> and log in as the admin user with redhat as the password.
- 2.2. Navigate to **Execution Environments > Execution Environments** to display the available automation execution environments.

The installer created the ee-29-rhel8, ee-minimal-rhel8, and ee-supported-rhel8 container repositories using the container image archives included in the Ansible

Chapter1 | Installing Red Hat Ansible Automation Platform

Automation Platform setup bundle. You created the three container repositories that use the `ansible-automation-platform-22` namespace.

Container repository name	Desc...	Created	Last modified	Conta...
ansible-automation-platform-22/ee-29-rhel8		2 minutes ago	2 minutes ago	Local
ansible-automation-platform-22/ee-minimal-rhel8		2 minutes ago	2 minutes ago	Local
ansible-automation-platform-22/ee-supported-rhel8		2 minutes ago	2 minutes ago	Local
ee-29-rhel8		33 minutes ago	33 minutes ago	Local
ee-minimal-rhel8		33 minutes ago	33 minutes ago	Local
ee-supported-rhel8		32 minutes ago	32 minutes ago	Local

Figure 1.18: Private automation hub automation execution environments

- 2.3. Click the link for the `ansible-automation-platform-22/ee-supported-rhel8` container repository and then click the **Images** tab. Private automation hub provides the container image using the **latest** tag.
- ▶ 3. Create the `containers` namespace and then upload the `containers.podman` content collection.
 - 3.1. Navigate to **Collections > Namespaces** and then click **Create**.
 - 3.2. Enter `containers` in the **Name** field and then click **Create** to create the namespace.
 - 3.3. Click **Upload collection**.
 - 3.4. Click **Select file**, select `/home/student/content-collections/community/containers-podman-1.9.1.tar.gz`, and then click **Upload**.
- ▶ 4. Create the `ansible` namespace and then upload the `ansible.posix` content collection.
 - 4.1. Navigate to **Collections > Namespaces** and then click **Create**.
 - 4.2. Enter `ansible` in the **Name** field and then click **Create** to create the namespace.
 - 4.3. Click **Upload collection**.
 - 4.4. Click **Select file**, select the `/home/student/content-collections/certified/ansible-posix-1.3.0.tar.gz` archive, and then click **Upload**.
- ▶ 5. Approve the uploaded content collections.
 - 5.1. Navigate to **Collections > Approval**.

Chapter 1 | Installing Red Hat Ansible Automation Platform

- 5.2. Click **Approve** to approve the `ansible.posix` content collection.
 - 5.3. Click **Approve** to approve the `containers.podman` content collection.
 - 5.4. Navigate to **Collections > Collections** and verify that private automation hub displays the `podman` and `posix` automation content collections.
- **6.** Synchronize the automation controller `Demo Project` project resource.
- 6.1. Navigate to `https://controller.lab.example.com` and log in as the `admin` user with `redhat` as the password.
 - 6.2. Navigate to **Resources > Projects** to display existing project resources.
 - 6.3. Click the **Sync Project** icon to synchronize the `Demo Project` resource.
- **7.** Update the `Demo Inventory` resource to remove the `localhost` resource. Add the `workstation.lab.example.com` host to the `Demo Inventory` resource.
- Making this change allows the `Demo Inventory` resource to target a machine in your environment (`workstation.lab.example.com`) rather than the automation execution environment (`localhost`).
- 7.1. Navigate to **Resources > Inventories** to display existing inventory resources.
 - 7.2. Click the link for the `Demo Inventory` resource and then click the **Hosts** tab.
 - 7.3. Select the `localhost` resource and then click **Delete**. Confirm your decision to delete the `localhost` resource.
 - 7.4. Click **Add**. Enter `workstation.lab.example.com` in the **Name** field and then click **Save**.
- **8.** Update the `Demo Credential` machine credential resource so that it uses valid credentials for accessing the `workstation.lab.example.com` host.
- 8.1. Navigate to **Resources > Credentials** to display existing credentials.
 - 8.2. Locate the `Demo Credential` resource and click the **Edit Credential** icon for that row.
 - 8.3. Enter `student` in the **Username** field, enter `student` in the **Password** field, and then click **Save**.
- If you previously completed this exercise, then click the **Replace** icon to enter a new password.
- **9.** Launch the `Demo Job Template` job.
- 9.1. Navigate to **Resources > Templates** to display existing templates.
 - 9.2. Click the **Launch Template** icon to initiate the job. The `Demo Job Template` resource uses the `Demo Credential` machine credential resource to connect to the `workstation.lab.example.com` host in the `Demo Inventory` resource.

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish install-configuration
```

► Quiz

Installing Red Hat Ansible Automation Platform

Choose the correct answers to the following questions:

- ▶ **1. What is the minimum amount of RAM and CPU supported for installation of an automation controller?**
 - a. 2 GB of RAM and 1 CPU
 - b. 4 GB of RAM and 2 CPUs
 - c. 8 GB of RAM and 3 CPUs
 - d. 16 GB of RAM and 4 CPUs

- ▶ **2. Which installer should you choose to install Red Hat Ansible Automation Platform in a disconnected environment?**
 - a. Red Hat Ansible Automation Platform installer
 - b. Red Hat Ansible Automation Platform Bundle installer

- ▶ **3. You can get a subscription manifest from:**
 - a. Red Hat Ansible Automation Platform Bundle
 - b. Red Hat Services Catalog
 - c. Red Hat Customer Portal
 - d. Red Hat Automation Hub

- ▶ **4. Which two of the following choices include support for Red Hat Ansible Automation Platform subscription? (Choose two.)**
 - a. Automation controller
 - b. Private automation hub
 - c. Ansible Galaxy
 - d. Ansible Content Collections from any source

- ▶ **5. Which two of the following content types can you store in a private automation hub? (Choose two.)**
 - a. ansible-core packages
 - b. Ansible Content Collections
 - c. Automation execution environments
 - d. Subscription manifest

► **6. Which tool is used to create custom automation execution environments?**

- a. ansible-navigator
- b. ansible-galaxy
- c. ansible-builder
- d. ansible-execution

► **7. Which command is used to upload an Ansible Content Collection to a private automation hub?**

- a. ansible-galaxy collection publish
- b. ansible-galaxy collection upload
- c. ansible-navigator collection publish
- d. ansible-navigator collection upload

► **8. Which command is used to upload an automation execution environment to the automation hub?**

- a. ansible-navigator --ee upload
- b. ansible-galaxy --ee publish
- c. podman push
- d. ansible-builder publish

► **9. Which three of the following choices are resources that you can create with the automation controller web UI? (Choose three.)**

- a. Projects
- b. Namespaces
- c. Credentials
- d. Inventories

► Solution

Installing Red Hat Ansible Automation Platform

Choose the correct answers to the following questions:

- ▶ **1. What is the minimum amount of RAM and CPU supported for installation of an automation controller?**
 - a. 2 GB of RAM and 1 CPU
 - b. 4 GB of RAM and 2 CPUs
 - c. 8 GB of RAM and 3 CPUs
 - d. 16 GB of RAM and 4 CPUs

- ▶ **2. Which installer should you choose to install Red Hat Ansible Automation Platform in a disconnected environment?**
 - a. Red Hat Ansible Automation Platform installer
 - b. Red Hat Ansible Automation Platform Bundle installer

- ▶ **3. You can get a subscription manifest from:**
 - a. Red Hat Ansible Automation Platform Bundle
 - b. Red Hat Services Catalog
 - c. Red Hat Customer Portal
 - d. Red Hat Automation Hub

- ▶ **4. Which two of the following choices include support for Red Hat Ansible Automation Platform subscription? (Choose two.)**
 - a. Automation controller
 - b. Private automation hub
 - c. Ansible Galaxy
 - d. Ansible Content Collections from any source

- ▶ **5. Which two of the following content types can you store in a private automation hub? (Choose two.)**
 - a. ansible-core packages
 - b. Ansible Content Collections
 - c. Automation execution environments
 - d. Subscription manifest

► **6. Which tool is used to create custom automation execution environments?**

- a. ansible-navigator
- b. ansible-galaxy
- c. ansible-builder
- d. ansible-execution

► **7. Which command is used to upload an Ansible Content Collection to a private automation hub?**

- a. ansible-galaxy collection publish
- b. ansible-galaxy collection upload
- c. ansible-navigator collection publish
- d. ansible-navigator collection upload

► **8. Which command is used to upload an automation execution environment to the automation hub?**

- a. ansible-navigator --ee upload
- b. ansible-galaxy --ee publish
- c. podman push
- d. ansible-builder publish

► **9. Which three of the following choices are resources that you can create with the automation controller web UI? (Choose three.)**

- a. Projects
- b. Namespaces
- c. Credentials
- d. Inventories

Summary

- Red Hat Ansible Automation Platform includes Ansible Core, Ansible Content Collections, automation content navigator, automation execution environments, automation controller, automation hub, private automation hub, and hosted services.
- Use the bundle installer to install Red Hat Ansible Automation Platform in a disconnected environment.
- Edit the included installation `inventory` file and run its `setup.sh` script to install or modify individual or multiple Ansible Automation Platform components.
- Connected environments can synchronize content, such as Ansible Content Collections and automation execution environments, with private automation hub.
- Manually upload Ansible Content Collections and automation execution environments to private automation hub in disconnected environments.
- Automation controller can pull content from several locations, such as automation hub, private automation hub, and Red Hat Ecosystem Catalog.

Chapter 2

Managing User Access

Goal

Create user accounts and organize them into teams and groups in automation controller and private automation hub, respectively, and assign them permissions to administer and access resources in each service.

Sections

- Creating and Managing Automation Controller Users (and Guided Exercise)
- Managing Automation Controller Access with Teams (and Guided Exercise)
- Creating and Managing Users and Groups for Private Automation Hub (and Guided Exercise)

Lab

- Managing User Access

Creating and Managing Automation Controller Users

Objectives

- Create new users in the web UI, and explain the different types of user in automation controller.

Role-based Access Controls

Different people using an automation controller installation require different levels of access. Some might need to run existing job templates against a preconfigured inventory of machines. Others need to be able to modify particular inventories, job templates, and playbooks, or need access to change anything in the automation controller installation.

The automation controller interface has a built-in administrative user, `admin`, which has superuser access to the entire automation controller configuration. Setting up user accounts for each person makes it easier to manage individual access to inventories, credentials, projects, and job templates.

Users are assigned *roles*, which in turn grant permissions. These permissions specify who can view, change, or delete an object in automation controller. Role-based Access Controls (RBAC) manage roles. To manage roles collectively, grant them to a *team*. A team is a collection of users. All users in a team inherit the team's roles.

Roles determine whether users and teams can view, use, change, or delete objects, such as inventories and projects.

Automation Controller Organizations

An automation controller organization is a logical collection of teams, projects, and inventories. All users must belong to an organization.

One of the benefits of implementing an automation controller server is sharing Ansible content, including Ansible Content Collections, roles, and automation execution environments, across departmental or functional boundaries within an enterprise. For example, an operations group of an organization might already have a custom automation execution environment for provisioning production versions of web, database, and application servers. The developers group can use the same environment to provision servers for their development environment. Automation controller makes it easier for different users and groups to use existing content collections, roles, and automation execution environments.

For large deployments, however, it can be useful to categorize large numbers of users, teams, projects, and inventories under one umbrella organization. Certain departments might not deploy to specific inventories of hosts, or run certain playbooks. By using organizations, you can configure a collection of users and teams to work with only those automation controller resources that they are expected to use.

The automation controller installation process creates a **Default** organization. You can create additional organizations in the web UI using the following procedure:

- Log in to the automation controller web UI as the `admin` user.
- Navigate to **Access > Organizations** and then click **Add**.

Chapter 2 | Managing User Access

- Enter a name for the new organization and, if desired, complete the optional fields.
 - Use the **Max Hosts** field to restrict the number of hosts that the organization can manage.
 - Use the **Execution Environment** field to specify a fallback automation execution environment if a project or job template does not explicitly specify an automation execution environment.
 - Use the **Galaxy Credentials** field to specify credentials that can pull roles and content collections from Ansible Galaxy, automation hub, or a private automation hub.
- Click **Save**.

The screenshot shows the 'Create New Organization' page. At the top left is the 'Organizations' navigation link. Below it is the title 'Create New Organization'. The form contains several input fields:

- Name ***: An input field with a red asterisk indicating it is required.
- Description**: An input field for a descriptive text.
- Max Hosts**: A dropdown menu showing the value '0'.
- Instance Groups**: A search input field with a magnifying glass icon.
- Execution Environment**: A search input field with a magnifying glass icon.
- Galaxy Credentials**: A search input field with a magnifying glass icon, containing the text 'Ansible Galaxy' with a delete 'X' button.

At the bottom are two buttons: a blue **Save** button and a blue **Cancel** button.

Figure 2.1: Creating a new organization

Types of Users

By default, the automation controller installer creates an **admin** user account with full control of the automation controller installation. You can use this account to log in to the web UI and create additional users.

The three user types in automation controller are **System Administrator**, **System Auditor**, and **Normal User**.

System Administrator

The **System Administrator** user type (also known as superuser) provides unrestricted access to perform any action within the entire automation controller installation. **System Administrator** is a special role, which has read/write permission on all objects in all organizations on the automation controller.

The **admin** user created by the installer also has the **System Administrator** single role and should therefore only be used by the automation controller administrator.

System Auditor

The **System Auditor** user type also has a special single role, which has read-only access to the entire automation controller installation.

Normal User

The **Normal User** is the standard user type. It initially has no special roles assigned and starts with minimal access. It is not assigned any single roles and is only assigned roles associated with the organization of which the user is a member.

Creating Users

Use the following procedure to create new users:

- Navigate to <https://controller.lab.example.com> and log in as the **admin** user with **redhat** as the password.
- Navigate to **Access > Users** and then click **Add**.
- Complete the **First Name**, **Last Name**, and **Email** fields.
- Specify a unique username in the **Username** field.
- Enter a password in the **Password** and **Confirm Password** fields.
- Select a **User Type** and **Organization**. By default, new users are assigned the **Normal User** user type and belong to the **Default** organization.
- Click **Save**.

The screenshot shows a 'Create New User' form. At the top left is a 'Users' link and a 'Create New User' button. The main area contains the following fields:

- First Name**: An input field.
- Last Name**: An input field.
- Email**: An input field.
- Username ***: An input field.
- Password ***: An input field with a visibility icon.
- Confirm Password ***: An input field with a visibility icon.
- User Type ***: A dropdown menu set to 'Normal User'.
- Organization ***: A dropdown menu set to 'Default'.

At the bottom are 'Save' and 'Cancel' buttons.

Figure 2.2: Creating a new user

Editing Users

To modify user details, use the same fields you used to create the user.

- Navigate to **Access > Users** and then click the **Edit User** icon for the user that you wish to edit.
- Make the changes to any of the desired fields.
- Click **Save** to finish.

Organization Roles

Users inherit specific roles from their organization based on their user type. You can assign additional roles to grant permissions to view, use, or change other automation controller objects. An organization is itself one of these objects.

Many roles provide access to an organization, and you can assign multiple roles to users and teams. The lecture on managing users efficiently with teams describes additional organization roles.

The following sections describe organization roles that are inherited based on user type.

The Admin Role

Users with the **System Administrator** single role inherit the **Admin** role on every organization within automation controller.

When assigned the **Admin** role on an organization, users gain the ability to manage all aspects of that organization, including reading and changing the organization, and adding and removing users and teams from the organization. A number of related administrative roles exist, with **Admin** in their names, that grant more limited access than the **Admin** role.

Teams cannot be assigned the **Admin** role in an organization.

The Auditor Role

Users with the **System Auditor** single role inherit the **Auditor** role on every organization within automation controller.

When assigned the **Auditor** role in an organization, users and teams gain read-only access to the organization.

The Member Role

When users are created with the **Normal User** type, automation controllers assign them a **Member** role in the organization. Other roles can be added later, including additional **Member** roles on other organizations.

When assigned the **Member** role in an organization, a user gains read permission to the organization. The organization **Member** role only provides a user the ability to view the list of users who are members of the organization and their assigned organization roles.

Unlike the organization **Admin** and **Auditor** roles, the **Member** role does not provide users permissions to any of the resources that the organization contains, such as teams, credentials, projects, inventories, job templates, workflow job templates, and notifications.

Teams cannot be assigned the **Member** role in an organization.

Managing User Organization Roles

Only the organization can manage the roles that a user has within that organization. You cannot manage roles by editing the user. To manage access to an organization, use the following procedure:

- Log in to the automation controller web UI as **admin** or any user with the **Admin** role on the organization being modified.
- Navigate to **Access > Organizations** and then click the name of the organization.

Chapter 2 | Managing User Access

- Click **Access**.

The **Access** page displays a list of users who have been granted roles for the organization.

To remove an existing role for a user, find the user's row and then click the X in the role name box.

The screenshot shows the 'Access' tab selected in the top navigation bar of a web interface. Below the navigation are search and 'Add' buttons. The main area displays a table of users with columns for Username, First name, Last name, and User Roles. The 'User Roles' column contains buttons for each assigned role, which include an 'X' icon to remove them. The table lists four users: admin, daniel, david, and donnie, each with their respective details and assigned roles.

Username	First name	Last name	User Roles
admin			System Administrator
daniel	Daniel	George	Project Admin X Member X
david	David	Jackobs	Member X
donnie	Donnie	Jameson	Member X

Figure 2.3: Organization roles assigned to users

To add a user to an organization, or to add additional roles to an existing user within the organization, use the following procedure:

- Click **Add**.
- Choose **Users** and then click **Next**.
- Select each user that you wish to manage together. You can add the same set of roles to one or more users. If you want to assign distinct roles for each user, then only select one user at a time.
- Select each role that you want to assign.
- Click **Save** to assign roles to the user for the organization.



References

For more information about users, refer to the *Automation Controller User Guide* at <https://docs.ansible.com/automation-controller/latest/html/userguide/users.html>

For more information about organizations, refer to the *Automation Controller User Guide* at

<https://docs.ansible.com/automation-controller/latest/html/userguide/organizations.html>

► Guided Exercise

Creating and Managing Automation Controller Users

Create user accounts of different types and explore the different levels of access of those account types.

Outcomes

- Create user accounts of different account types.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command ensures that automation controller is installed.

```
[student@workstation ~]$ lab start org-user
```

Instructions

► 1. Navigate to <https://controller.lab.example.com> and log in as the `admin` user with `redhat` as the password.

► 2. Create a user, `sam`, as a Normal User.

2.1. Navigate to **Access > Users** and then click **Add**.

2.2. On the **Create New User** page, fill in the details as follows:

Field	Value
First Name	Sam
Last Name	Simons
Email	<code>sam@lab.example.com</code>
Username	<code>sam</code>
Password	<code>redhat123</code>
Confirm Password	<code>redhat123</code>
User Type	Normal User
Organization	Default

2.3. Click **Save** to create the new user.

Chapter 2 | Managing User Access

- 3. Verify the roles for the newly created **sam** user.
- 3.1. Click the **Roles** tab to see the user's roles.
 - 3.2. User **sam** has the **Member** role for the **Default** organization.
 - 3.3. Click **admin** > **Logout** in the upper-right corner to log out and then log back in as the **sam** user with **redhat123** as the password.
 - 3.4. User **sam** has limited access. In the left navigation, some items are not displayed, such as **Administration** > **Notifications**, **Administration** > **Management Jobs**, **Administration** > **Instance Groups**, and **Settings**.
 - 3.5. Navigate to **Access** > **Users**. User **sam** cannot add, modify, or delete users. This user can update basic settings, such as the first name, last name, email, username, and password fields.
 - 3.6. Click **sam** > **Logout** in the upper-right corner to log out.
- 4. Create a user, **sylvia**, as a System Auditor.
- 4.1. Log back in as the **admin** user and use **redhat** as the password.
 - 4.2. Navigate to **Access** > **Users** and then click **Add**.
 - 4.3. On the **Create New User** page, fill in the details as follows:

Field	Value
First Name	Sylvia
Last Name	Simons
Email	sylvia@lab.example.com
Username	sylvia
Password	redhat123
Confirm Password	redhat123
User Type	System Auditor
Organization	Default

- 4.4. Click **Save** to create the new user.
- 5. Verify the roles for **sylvia**.
- 5.1. Click the **Roles** tab to see the user's roles. Notice that user **sylvia** has the **System Auditor** role.
 - 5.2. Log out and log back in as **sylvia** user with **redhat123** as the password.
 - 5.3. The user **sylvia** has access to all elements, such **Users**, **Teams**, **Projects**, **Credentials**, and so on. Although the user can see all of the resources, the user cannot add, modify, delete, or use any of the resources.

Chapter 2 | Managing User Access

- 5.4. Navigate to **Access > Users**. The user **sylvia** cannot add, modify, or delete users. This user can update basic settings, such as the first name, last name, email, username, and password fields.
 - 5.5. Click **sylvia > Logout** in the upper-right corner to log out.
- 6. Create a user, **simon**, as a System Administrator.
- 6.1. Log back in as the **admin** user and use **redhat** as the password.
 - 6.2. Navigate to **Access > Users** and then click **Add**.
 - 6.3. On the **Create New User** page, fill in the details as follows:

Field	Value
First Name	Simon
Last Name	Stephens
Email	simon@lab.example.com
Username	simon
Password	redhat123
Confirm Password	redhat123
User Type	System Administrator
Organization	Default

- 6.4. Click **Save** to create the new user.
- 7. Verify the roles for **simon**.
- 7.1. Click the **Roles** tab to see the user's roles. This time the **Roles** page explicitly displays the message: *System administrators have unrestricted access to all resources*.
 - 7.2. Log out and log back in as **simon** user with **redhat123** as the password.
 - 7.3. The user **simon** has access to all elements, such as **Users**, **Teams**, **Projects**, **Credentials**, and so on.
 - 7.4. Navigate to **Access > Users**. The user **simon** can add, modify, and delete users.
 - 7.5. Click **simon > Logout** to log out of the automation controller web UI.

Finish

On the **workstation** machine, change to the **student** user home directory and use the **lab** command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish org-user
```

Managing Automation Controller Access with Teams

Objectives

- Create new teams in the automation controller web user interface, assign users to them, and explain the different roles that can be assigned to users.

Teams in Automation Controller

Teams are groups of users. Teams make managing roles on automation controller objects, such as inventories, projects, and job templates, more efficient than managing roles for each user separately. Automation controller administrators can assign roles to teams. All team members inherit the roles assigned to that team.

This means that you do not have to assign the same roles to multiple, individual users.

In automation controller, users exist as objects at an automation controller-wide level. Therefore, a user can have roles in multiple organizations.

By contrast, a team belongs to exactly one organization. However, an admin user can assign the team roles on resources that belong to other organizations.

You can assign organization roles to teams.

Creating Teams

Use the following procedure to create teams within each organization:

- Log in to the automation controller web UI as the **admin** user or as a user assigned the **Admin** role for the organization in which you intend to create the new team.
- Navigate to **Access > Teams** and then click **Add**.
- Enter a name for the new team in the **Name** field.
- If desired, enter a description in the **Description** field.
- In the **Organization** field choose the **Default** organization, or click the search icon to select a different organization.
- Click **Save**.

Team Roles

You can assign roles to users for a particular team. These roles control whether the user can manage the team, or can only view team membership.

You can assign multiple team roles to users. These roles are described in the following sections.

The Member Role

Users with the team **Member** role inherits roles on automation controller resources granted to the team. It also grants users the ability to view the team's users and associated team roles.

The Admin Role

The **Admin** team role grants users full control of the team. Users with this team role can manage the team's users and their associated team roles. Users with **Admin** team roles can also manage the team's roles on resources for which the team has been assigned the **Admin** role.

Users with **Admin** team roles can only manage the team's roles on a resource when the resource also grants the **Admin** team role on itself. Just because a team **Admin** can manage team membership, it does not imply that the team **Admin** has any rights to manage roles on objects to which the team has access.

For example, for a user to grant a team the **Use** role for a project, the user must have the **Admin** role for both the team and the project.

The Read Role

The **Read** team role gives users the ability to view the team's users and their associated team roles. A user assigned a **Read** team role does not inherit roles that the team has been granted on automation controller resources.



Note

In practice, most organizations do not use team roles other than **Member**. Instead, team membership is managed through an external authentication source, or the **Organization Administrator** and **System Administrator** roles are used for administrative purposes and **System Auditor** for auditing requirements instead of **Read** on individual teams.

Adding Users to a Team and Assigning Team Roles

After you have created a team, you can add users to that team. Add users to a team by assigning one or more team roles using the following procedure:

- Log in to the automation controller web UI as the **admin** user or as a user assigned the **Admin** role for the organization to which the team belongs.
- Navigate to **Access > Teams** and then click the link for the name of a team.
- Click the **Access** tab and then click **Add**.
- Select **Users** and then click **Next**.
- Select each user that you want to manage together. You can add the same set of team roles to one or more users. If you want to assign distinct team roles for each user, then only select one user at a time.
- Select each team role that you want to assign.
- Click **Save**.

Associating a User with a Team

You can also assign the **Member** role to a user by associating the user with a team. This process is the equivalent of adding a user to a team and assigning the **Member** role to the user.

- Log in to the automation controller web UI as the **admin** user or as a user assigned the **Admin** role for the organization to which the team belongs.

Chapter 2 | Managing User Access

- Navigate to Access > Users and then click the link for the name of a user.
- Click the **Teams** tab. Associating a user with a team adds the team **Member** role. Disassociating a user from a team removes the team **Member** role.

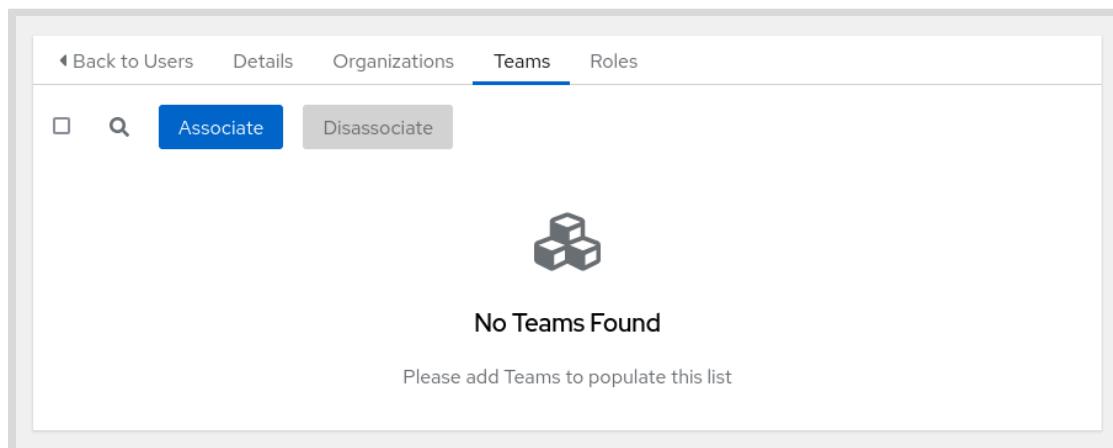


Figure 2.4: Team membership for a user

- Click **Associate**.
- Assign team membership by selecting a team name and then click **Save**.

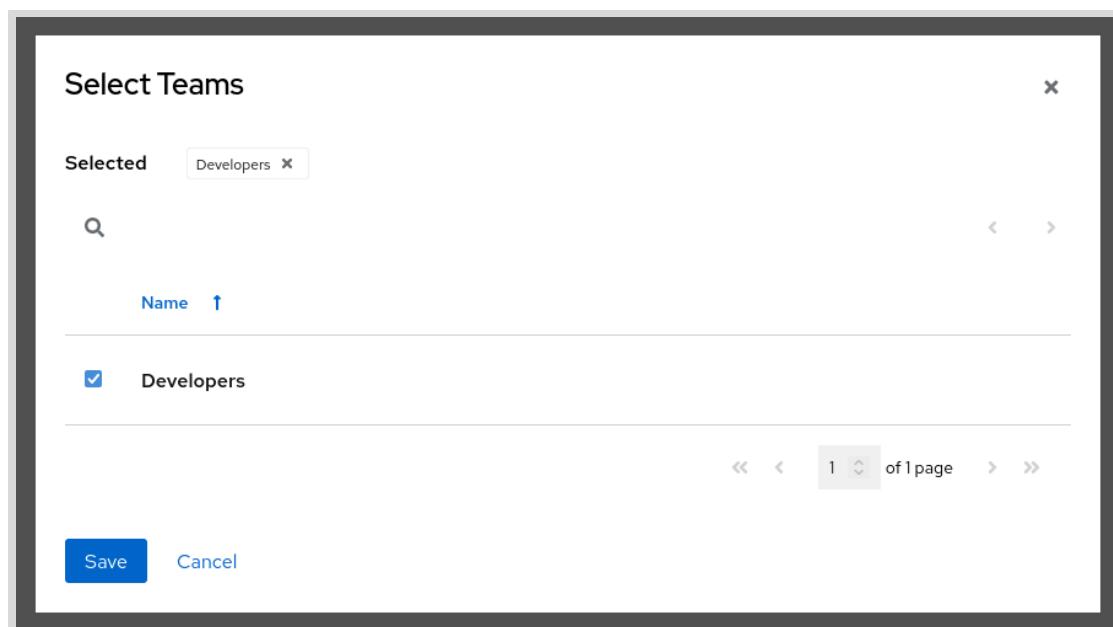


Figure 2.5: Assign team membership

Organization Roles

As previously stated, many roles provide access to an organization and multiple roles can be assigned to users and teams.

The following organization roles can be assigned to both users and teams.

Execute

A user with the **Execute** role has permission to execute job templates and workflow job templates belonging to the organization.

Chapter 2 | Managing User Access

Admin

A user with the **Admin** role has full administrative control over the organization and its objects.

Project Admin

A user with the **Project Admin** role can create, read, update and delete any project in the organization. In conjunction with the **Inventory Admin** permission, this allows users to create job templates.

Inventory Admin

A user with the **Inventory Admin** role can create, read, update and delete any inventory in the organization. In conjunction with the **Job Template Admin** and **Project Admin** roles, this allows the user full control over job templates within the organization.

Credential Admin

A user with the **Credential Admin** role can manage all credentials of the organization.

Workflow Admin

A user with the **Workflow Admin** role can manage all workflows of the organization.

Notification Admin

A user with the **Notification Admin** role can manage all notifications belonging to the organization.

Job Template Admin

A user with the **Job Template Admin** role can make changes to nonsensitive fields within job templates. To make changes to fields that impact job runs, the user also needs the **Admin** role on the job template, the **Use** role on the related project, and the **Use** role in the related inventory.

Execution Environment Admin

A user with the **Execution Environment Admin** role can manage all execution environments of the organization.

Auditor

When assigned the **Auditor** role on an organization, a user gains read-only access to the organization.

Read

When assigned the **Read** role on an organization, a user gains read permission to the organization only. The organization **Read** role only provides a user with the ability to view the list of users who are members of the organization and their assigned organization roles. Unlike the organization **Admin** and **Auditor** roles, the **Read** role does not inherit roles on any of the resources that the organization contains, such as teams, credentials, projects, inventories, job templates, workflow job templates, and notifications. The organization object cannot be assigned roles on automation controller resources. Therefore, a user that has the **Member** role on an organization only has access to the organization object and inherits no other permissions as a result of this role. Consequently, a user that has the **Member** role on an organization is the equivalent of a user that has the **Read** role on an organization.

Approve

A user with the **Approve** role can approve or deny a workflow approval node.

Managing Organization Roles

Use the following procedure to manage access to an organization:

- Log in to the automation controller web UI as **admin** or any user with the **Admin** role on the organization being modified.

Chapter 2 | Managing User Access

- Navigate to Access > **Organizations** and then click the name of the organization.
- Click the **Access** tab.

The **Access** page displays a list of users who have been granted roles for the organization. Roles are categorized as **User Roles** if they were assigned to an individual role or as **Team Roles** if they were assigned to a team.

To remove an existing role for a user, find the user's row and then click the X in the role name box. Removing a team role removes the role from all team members, not just for an individual user.

Use the following procedure to assign organization roles to users and teams:

- Click **Add**.
- Choose either **Users** or **Teams** and then click **Next**. You cannot assign roles to both users and teams at the same time.
- Select each user or team that you want to manage together. You can add the same set of roles to one or more users or teams. If you want to assign distinct roles for each user or team, then only select one user or team at a time.
- Select each role that you want to assign.
- Click **Save**.



References

For more information about teams, refer to the *Automation Controller User Guide* at
<https://docs.ansible.com/automation-controller/latest/html/userguide/teams.html>

For more information about organizations, refer to the *Automation Controller User Guide* at
<https://docs.ansible.com/automation-controller/latest/html/userguide/organizations.html>

► Guided Exercise

Managing Automation Controller Access with Teams

Organize users into teams and explore the access provided by different team roles.

Outcomes

- Organize users into teams.
- Assign different roles to team members.

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that automation controller is installed.

```
[student@workstation ~]$ lab start org-team
```

Instructions

► 1. Create a new team called **Developers**.

- 1.1. Navigate to `https://controller.lab.example.com` and log in as the `admin` user with `redhat` as the password.
- 1.2. Navigate to **Access > Teams** and then click **Add**.
- 1.3. Create the team using the following information:

Field	Value
Name	Developers
Description	Dev Team
Organization	Default

- 1.4. Click **Save** to create the new team.

► 2. Create the `daniel` user.

- 2.1. Navigate to **Access > Users** and then click **Add**.
- 2.2. Create the new user with the following information:

Field	Value
First Name	Daniel
Last Name	George
Email	daniel@lab.example.com
Username	daniel
Password	redhat123
Confirm Password	redhat123
User Type	Normal User
Organization	Default

**Note**

The user type is set to **Normal User** rather than **System Administrator**. The **daniel** user will be an administrator of the **Developers** team, but the user will not be assigned administrative capabilities on the entire automation controller instance.

- 2.3. Click **Save** to create the new user.
- ▶ 3. Assign the **daniel** user the **Admin** role on the **Developers** team.
 - 3.1. Navigate to **Access > Teams** and then click the link for the **Developers** team.
 - 3.2. Click the **Access** tab and then click **Add**.
 - 3.3. Click **Users** and then click **Next**.
 - 3.4. Select **daniel** and then click **Next**.
 - 3.5. Select **Admin** and then click **Save** to assign the role.
- ▶ 4. Create the **donnie** user.
 - 4.1. Navigate to **Access > Users** and then click **Add**.
 - 4.2. Create the new user with the following information:

Field	Value
First Name	Donnie
Last Name	Jameson
Email	donnie@lab.example.com
Username	donnie
Password	redhat123
Confirm Password	redhat123
User Type	Normal User
Organization	Default

4.3. Click **Save** to create the new user.

► **5.** Assign the **donnie** user the **Read** role on the **Developers** team.

- 5.1. Navigate to **Access > Teams** and then click the link for the **Developers** team.
- 5.2. Click the **Access** tab and then click **Add**.
- 5.3. Click **Users** and then click **Next**.
- 5.4. Select **donnie** and then click **Next**.
- 5.5. Select **Read** and then click **Save** to assign the role.

► **6.** Create the **david** user and associate the user with the **Developers** team. Associating the user adds the **Member** role for the team.

- 6.1. Navigate to **Access > Users** and then click **Add**.
- 6.2. Create the new user with the following information:

Field	Value
First Name	David
Last Name	Jackobs
Email	david@lab.example.com
Username	david
Password	redhat123
Confirm Password	redhat123
User Type	Normal User
Organization	Default

Chapter 2 | Managing User Access

- 6.3. Click **Save** to create the new user.
- 6.4. On the **Details** page for the **david** user, click the **Teams** tab and then click **Associate**.
- 6.5. Select **Developers** and then click **Save** to assign the **Member** role.
- 6.6. Click **admin > Logout** to exit the automation controller web UI.

► 7. Verify the permissions for the **daniel user.**

- 7.1. Log in to the automation controller web UI as **daniel** with **redhat123** as the password.
- 7.2. Navigate to **Access > Teams** and then click the link for the **Developers** team.
Notice that the **Edit** button is available, indicating that the **daniel** user can make changes to the **Developers** team.
- 7.3. Click the **Access** tab.
Notice that the **Add** button is available and that each existing role assignment for members of the **Developers** team contains an **X**, indicating that the role assignment can be removed. The **daniel** user can manage permissions for the **Developers** team. Having the **Admin** role additionally allows the **daniel** user to inherit permissions assigned to the **Developers** team.
- 7.4. Click **daniel > Logout** to exit the automation controller web UI.

► 8. Verify the permissions for the **donnie user.**

- 8.1. Log in to the automation controller web UI as **donnie** with **redhat123** as the password.
- 8.2. Navigate to **Access > Teams** and then click the link for the **Developers** team.
- 8.3. Click the **Access** tab.
Notice that the **Add** button is not available and that each existing role assignment does not contain an **X**, indicating that the role assignment cannot be removed. The **donnie** user can view information about the **Developers** team, but the user cannot make changes to it. Additionally, by only having the **Read** role, the **donnie** user does not inherit permissions assigned to the **Developers** team.
- 8.4. Click **donnie > Logout** to exit the automation controller web UI.

► 9. Verify the permissions for the **david user.**

- 9.1. Log in to the automation controller web UI as **david** with **redhat123** as the password.
- 9.2. Navigate to **Access > Teams** and then click the link for the **Developers** team.
- 9.3. Click the **Access** tab.
Notice that the **Add** button is not available and that each existing role assignment does not contain an **X**, indicating that the role assignment cannot be removed. Like the **donnie** user, the **david** user can view information about the **Developers** team, but the user cannot make changes to it. Unlike the **donnie** user, having the **Member** role allows the **david** user to inherit permissions assigned to the **Developers** team.

- 9.4. Click **david** > **Logout** to exit the automation controller web UI.

Finish

On the **workstation** machine, change to the **student** user home directory and use the **lab** command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish org-team
```

Creating and Managing Users and Groups for Private Automation Hub

Objectives

- Create and manage users and groups for private automation hub through its web UI and configure access permissions for users by using groups.

User Access

Enterprises can use private automation hub to manage and control the lifecycle of their Ansible content. They can host Ansible Content Collections and automation execution environments on their private automation hub and provide controlled access to their users.

Different user groups can be content creators, operators, or domain experts, who each need a different level of access to the content. For example, the `content_creators` group needs permission to write and modify the automation code, whereas the `operator` group needs read-only access to run an automation job.

Private automation hub provides a simple but efficient way of managing user access to the content. User access is based on managing permissions to system objects. The system objects are users, groups, namespaces, and repositories.

To manage content and access to content in private automation hub, you can create groups and assign object permissions to those groups. Then you can assign users to these groups, so that each user in a group has the permissions assigned to that group. Managing permissions for groups might be easier than managing permissions for individual users.

Creating Groups

You can create and assign permissions to a group in private automation hub that provide access to specified features in the system for members of that group..

Use the following procedure to add a new group to private automation hub:

1. Log in to your private automation hub using credentials for the `admin` user configured during installation.
2. Navigate to `User Access > Groups` and then click `Create`.
3. Enter a valid name and click `Create` to create the group.
4. Click `Edit`.
5. Click in the field for each permission type and select permissions that appear in the list.
6. Click `Save` when finished assigning permissions.

You can add permissions when you create groups or edit an existing group to add or remove permissions.

The following table lists the types of private automation hub permissions.

Object	Permissions	Permission description
Collection Namespaces	Add namespace, Change namespace, Delete namespace, and Upload to namespace	Create, modify, or delete namespaces, and upload Ansible Content Collections to them.
Collections	Delete collections	Delete Ansible Content Collections.
	Modify Ansible repo content	Move Ansible Content Collections between repositories, using the Approval feature to certify an Ansible Content Collection and move it from the staging repository to the published repository, or to reject it and move it from the staging repository to the rejected repository.
Users	Add user, Change user, Delete user, and View user	Manage user configuration and access in private automation hub.
Groups	Add group, Change group, Delete group, and View group	Manage group configuration and access in private automation hub.
Collection Remotes	Change collection remote and View collection remote	Configure or view configured remote repositories of Ansible Content Collections that can be synchronized to the private automation hub, under Collections > Repository Management.
Containers	Change container namespace permissions	Change permissions on the container repository.
	Change containers	Change information on containers.
	Change image tags	Modify image tags on containers.
	Create new containers	Upload new containers.
	Delete container repository	Delete a container repository.
	Push to existing containers	Push an image to an existing container.
Remote Registries	Add remote registry, Change remote registry, and Delete remote registry	Add, change, or delete remote registries in private automation hub.
Task Management	Change task, Delete task, and View all tasks	Manage tasks under Task Management in private automation hub.

Creating Users

The private automation hub installation process creates the default `admin` user. This user is assigned all permissions in the system.

Chapter 2 | Managing User Access

You can create users in private automation hub and add them to groups. Use the following procedure to add a new user to private automation hub:

1. Log in to your private automation hub using credentials for the **admin** user or as a user who has permission to manage users.
2. Navigate to **User Access > Users** and then click **Create**.
3. Enter a valid **Username**, **First name**, **Last name**, **Email**, and **Password**.
4. Assign the user to a group by clicking the **Groups** field and selecting from the list of groups.
5. Keep the **User type** as **Not a super user**.
6. Click **Save**.

The screenshot shows the 'Create User' form in the Red Hat private automation hub. The fields are as follows:

- Email:** hob@lab.example.com
- Password:** *****
- Password confirmation:** *****
- Groups:** apgroup
- User type:** Not a super user

At the bottom are 'Save' and 'Cancel' buttons.

Figure 2.6: Creating a new user



Important

Super users are assigned all system permissions regardless of what groups they are in.

Creating Groups to Manage Content

You can create different groups in private automation hub and assign different permissions based on their role. For example, you can create one group for system administrators responsible for governing internal Ansible Content Collections, configuring user access, and repository management. You can create another group for content curators responsible for organizing and uploading internally developed content to private automation hub.

Suppose in your organization you need to create a new group in private automation hub to manage automation content. The group manages the internally developed Automation Content Collections and the automation execution environments in private automation hub.

Chapter 2 | Managing User Access

Use the following procedure to add a new group to private automation hub and assign necessary permissions:

1. Log in to private automation hub using credentials for the `admin` user.
2. Navigate to **User Access > Groups** and then click **Create**.
3. Enter `app_team` in the **Name** field and click **Create** to create the group.
4. Click **Edit**.
5. For **Collection Namespaces** permissions, select **Add namespace**, **Change namespace**, **Delete namespace**, and **Upload to namespace**.
6. For **Collections** permission, select **Modify Ansible repo content**.
7. For **Containers** permission, select **Change containers**, **Change image tags**, **Create new containers**, **Delete container repository**, **Push to existing containers**.
8. Click **Save**.

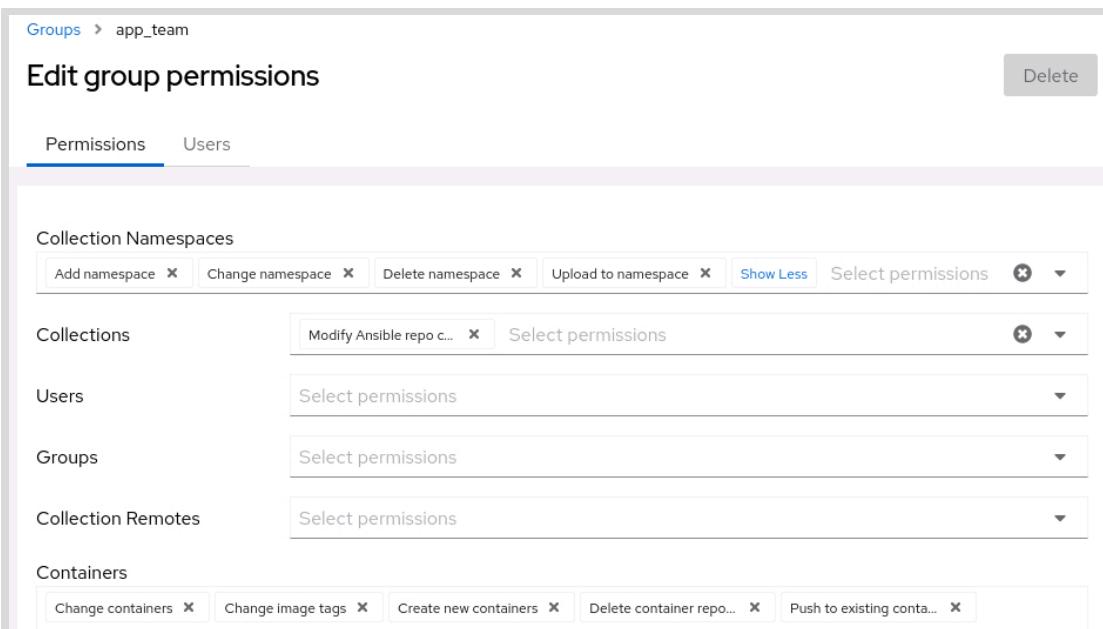


Figure 2.7: Specific group permissions

Next, create a new user `andrew` as a member of `app_team` group.

1. Log in to your private automation hub using credentials for the `admin` user or as a user who has permission to manage users.
2. Navigate to **User Access > Users** and then click **Create**.
3. Enter `andrew` as the **Username** and `redhat123` as the **Password**.
4. Assign the user to group `app_team` by clicking in the **Groups** field and selecting from the list of groups.
5. Keep the **User type** as **Not a super user**.
6. Click **Save**.

Chapter 2 | Managing User Access**Verify that user andrew can manage Ansible Content Collections.**

1. Log in to private automation hub as the user andrew.
2. Navigate to Collections > Namespaces.
3. Click Create.
4. Enter a name for the namespace and select app_team as Namespace owners.
5. Confirm that the user can upload Ansible Content Collections in the namespace.

The screenshot shows the Red Hat Automation Hub interface. At the top, there's a navigation bar with the Red Hat logo and a dropdown for 'Andrew Ryan'. Below the header, there are filters for 'Filter by repository' (set to 'Published') and a breadcrumb navigation path: 'Namespaces > app_dev'. On the right side of the header, there are buttons for 'Upload collection' and three vertical dots for more options. The main content area is titled 'app_dev' and has tabs for 'Collections' (which is selected) and 'CLI Configuration'. Under the 'Collections' tab, there's a search bar with 'Keywords' and a dropdown, a 'Filter by keywords' input, and a search icon. Below the search bar, it says '1-1 of 1' with back and forward arrows. The collection details show a single item named 'application1' with a description 'your collection description'. It lists '0 Modules', '0 Roles', '0 Plugins', and '0 Dependencies'. To the right of the collection details, there are buttons for 'Upload new version' and three vertical dots, followed by the text 'Updated 6 minutes ago' and 'v1.0.0'.

Figure 2.8: Permission to upload collections

Verify that user andrew can manage automation execution environments.

1. Log in to private automation hub as the user andrew.
2. Navigate to Execution Environments > Execution Environments.

The screenshot shows the Red Hat Private Automation Hub interface. At the top, there's a navigation bar with the Red Hat logo and a user profile for Andrew Ryan. Below the header, the title "Execution Environments" is displayed. A search bar and a button to "Add execution environment" are visible. The main area shows a table of execution environments with two entries:

Container repository name	Description	Created	Last modified	Contains	Action
ansible-automation-platform-20-early-access/ee-29-rhel8	Ansible...	an hour ago	an hour ago	Remote	...
ansible-automation-platform-20-early-access/ee-minimal-rhel8	Ansible...	an hour ago	an hour ago	Remote	...

Figure 2.9: Permission to manage execution environment

Private automation hub provides content creators a single source of truth to collaborate and publish their automation content within their organizations. To efficiently manage access to your automation content, you can create groups with the right permissions and add users into those groups. This approach is simple compared to assigning permissions to individual users.



References

Managing user access in Private Automation Hub

https://access.redhat.com/documentation/en-us/red_hat_ansible_automation_platform/2.1/html/managing_user_access_in_private_automation_hub/index

Managing containers in private automation hub

https://access.redhat.com/documentation/en-us/red_hat_ansible_automation_platform/2.1/html/managing_containers_in_private_automation_hub/index

► Guided Exercise

Creating and Managing Users and Groups for Private Automation Hub

Create users and groups in private automation hub and assign appropriate access permissions to the groups.

Outcomes

- Create users and groups in private automation hub.
- Assign appropriate access permissions to the groups.

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that private automation hub is installed.

```
[student@workstation ~]$ lab start org-hub
```

Instructions

- ▶ 1. Create a new group called `Developers` and assign permissions to manage Ansible Content Collections and containers in private automation hub.
 - 1.1. Navigate to <https://hub.lab.example.com> and log in as the `admin` user with `redhat` as the password.
 - 1.2. Navigate to `User Access > Groups` and then click `Create`.
 - 1.3. Enter `Developers` in the `Name` field and then click `Create`.
 - 1.4. Click `Edit`. In the `Collection Namespaces` object list, select the following permissions:
 - Add namespace
 - Change namespace
 - Delete namespace
 - Upload to namespace
 - 1.5. In the `Collections` object list, select the following permissions:
 - Delete collection
 - Modify Ansible repo content.
 - 1.6. In the `Containers` object list, select the following permissions:
 - Change container namespace permissions
 - Change containers
 - Change image tags

Chapter 2 | Managing User Access

- Create new containers
 - Delete container repository
 - Push to existing containers
- 1.7. Click **Save** to create the new group.
- 2. Create a new group called **Image Managers** and assign permissions to manage images in private automation hub.
- 2.1. Navigate to **User Access > Groups** and then click **Create**.
 - 2.2. Enter **Image Managers** in the **Name** field and then click **Create**.
 - 2.3. Click **Edit**. In the **Containers** object list, select the following permissions:
 - Change containers
 - Change image tags
 - Create new containers
 - Delete container repository
 - Push to existing containers
 - 2.4. Click **Save** to create the new group.
- 3. Create a new group called **Operations** and assign all the permissions to all the objects in private automation hub.
- 3.1. Navigate to **User Access > Groups** and then click **Create**.
 - 3.2. Enter **Operations** in the **Name** field and then click **Create**.
 - 3.3. Click **Edit** and for each object select all the permissions.
 - 3.4. Click **Save** to create the new group.
- 4. Create a new user called **daniel** and add the user to the **Developers** group.
- 4.1. Navigate to **User Access > Users** and then click **Create**.
 - 4.2. On the **Create new user** page, fill in the details as follows and click **Save** to create the new user.

Field	Value
Username	daniel
First name	Daniel
Last name	George
Email	daniel@lab.example.com
Password	redhat123
Password confirmation	redhat123
Groups	Developers
User type	Not a super user

Chapter 2 | Managing User Access

- 5. Create a new user called `oliver` and add the user to the `Operations` group.
- 5.1. Navigate to `User Access > Users` and then click `Create`.
 - 5.2. On the `Create new user` page, fill in the details as follows and click `Save` to create the new user.

Field	Value
Username	<code>oliver</code>
First name	<code>Oliver</code>
Last name	<code>Stone</code>
Email	<code>oliver@lab.example.com</code>
Password	<code>redhat123</code>
Password confirmation	<code>redhat123</code>
Groups	<code>Operations</code>
User type	<code>Not a super user</code>

- 6. Create a new user called `simon` as a super user.
- 6.1. Navigate to `User Access > Users` and then click `Create`.
 - 6.2. On the `Create new user` page, fill in the details as follows and click `Save` to create the new user.

Field	Value
Username	<code>simon</code>
First name	<code>Simon</code>
Last name	<code>Stephens</code>
Email	<code>simon@lab.example.com</code>
Password	<code>redhat123</code>
Password confirmation	<code>redhat123</code>
Groups	<code>(no group)</code>
User type	<code>Super user</code>

- 7. Verify the permissions for the `Developers` group by creating a namespace and then uploading a content collection.
- 7.1. Log out from the private automation hub web UI and log in as `daniel` with `redhat123` as the password.
 - 7.2. Navigate to `Collections > Namespaces` and then click `Create`.

Chapter 2 | Managing User Access

- 7.3. On the **Create new namespace** page, fill in the details as follows and click **Create** to create the new namespace.

Field	Value
Name	community
Namespace owners	Developers

**Important**

The group must be a namespace owner in order to upload to the namespace. Adding a group as namespace owner provides **Change namespace** and **Upload to namespace** permissions to the group.

- 7.4. Click **Upload collection**.
- 7.5. Click **Select file**, select the archive located at /home/student/content-collections/community/community-mysql-3.1.1.tar.gz, and then click **Upload**.
- 7.6. After the upload completes successfully, click **Collections > Approval**.
- 7.7. Click **Approve** to approve the community.mysql content collection.
- 7.8. Navigate to **Collections > Collections** and verify that the private automation hub server displays the mysql automation content collection.
- ▶ **8.** Confirm that user simon has all permissions, as the super user.
- 8.1. Log out from the private automation hub web UI and log in as simon with redhat123 as the password.
- 8.2. Navigate to **User access > Users** and confirm that **Create** button is enabled. It means user simon has permission to create new users.
- 8.3. Navigate to **Collections > Namespaces** and confirm that **Create** button is enabled. It means user simon has permission to create new namespaces.
- 8.4. Navigate to **Execution Environments > Execution Environments** and confirm that **Add execution environment** is enabled. It means user simon has permission to add new execution environment.
- 8.5. Log out from the private automation hub web UI.

Finish

On the **workstation** machine, change to the **student** user home directory and use the **lab** command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish org-hub
```

► Lab

Managing User Access

Create users on automation controller and private automation hub and assign the users access permissions as specified by the lab instructions.

Outcomes

- Create users in automation controller and in private automation hub.
- Create teams in automation controller and groups in private automation hub.
- Add users to automation controller teams and private automation hub groups.
- Assign team roles to automation controller users and permissions to private automation hub groups.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command ensures that automation controller and private automation hub are installed.

```
[student@workstation ~]$ lab start org-review
```

Instructions

1. Go to `https://controller.lab.example.com` and log in as the admin user with `redhat` as the password.
2. Add two automation controller users with the following information:

Field	Value for the first user	Value for the second user
First Name	Oliver	Ophelia
Last Name	Stone	Dunham
Email	<code>oliver@lab.example.com</code>	<code>ophelia@lab.example.com</code>
Username	<code>oliver</code>	<code>ophelia</code>
Password	<code>redhat123</code>	<code>redhat123</code>
Confirm Password	<code>redhat123</code>	<code>redhat123</code>
User Type	Normal User	Normal User
Organization	Default	Default

Chapter 2 | Managing User Access

3. Create an automation controller team called **Operations** using **Ops Team** as the description. The new team must belong to the **Default** organization.
4. Add **Oliver** as a **Member** to the **Operations** team.
5. Add **Ophelia** as an **Admin** to the **Operations** team.
6. Add a private automation hub group called **Container Developers**. Assign all the permissions in the **Containers** category to the new **Container Developers** group.
7. Add a private automation hub user using the information in the following table. When finished, log out of the private automation hub web UI.

Field	Value
Username	carlos
First name	Carlos
Last name	Cortez
Email	carlos@lab.example.com
Password	redhat123
Password confirmation	redhat123
Groups	Container Developers
User type	Not a super user

8. From a terminal window, use the `skopeo` command to identify the version and release of the private automation hub `ansible-automation-platform-22/ee-supported-rhel8:latest` container image.

To do this, access the container repository on `hub.lab.example.com` as the newly created `carlos` user.

9. Using the private automation hub web UI, add a new tag to the `ansible-automation-platform-22/ee-supported-rhel8:latest` container image. Use information about the version and release of the container image obtained in the previous step to create the tag using the format of "VERSION-RELEASE". For example, the `ansible-automation-platform-22/ee-29-rhel8:latest` container image would use the `1.0.0-119` tag, where `1.0.0` is the version, and `119` is the release.
10. Use the `skopeo list-tags` command to verify that the `ansible-automation-platform-22/ee-supported-rhel8` container repository uses the `1.0.0-99` tag.

Evaluation

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade org-review
```

Finish

On the **workstation** machine, change to the **student** user home directory and use the **lab** command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish org-review
```

► Solution

Managing User Access

Create users on automation controller and private automation hub and assign the users access permissions as specified by the lab instructions.

Outcomes

- Create users in automation controller and in private automation hub.
- Create teams in automation controller and groups in private automation hub.
- Add users to automation controller teams and private automation hub groups.
- Assign team roles to automation controller users and permissions to private automation hub groups.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command ensures that automation controller and private automation hub are installed.

```
[student@workstation ~]$ lab start org-review
```

Instructions

1. Go to <https://controller.lab.example.com> and log in as the `admin` user with `redhat` as the password.
2. Add two automation controller users with the following information:

Field	Value for the first user	Value for the second user
First Name	Oliver	Ophelia
Last Name	Stone	Dunham
Email	oliver@lab.example.com	ophelia@lab.example.com
Username	oliver	ophelia
Password	redhat123	redhat123
Confirm Password	redhat123	redhat123
User Type	Normal User	Normal User
Organization	Default	Default

Chapter 2 | Managing User Access

- 2.1. Go to **Access > Users** and then click **Add**. Create the **oliver** user with the following information. When finished, click **Save** to create the user.

Field	Value
First Name	Oliver
Last Name	Stone
Email	oliver@lab.example.com
Username	oliver
Password	redhat123
Confirm Password	redhat123
User Type	Normal User
Organization	Default

- 2.2. Go to **Access > Users** and then click **Add**. Create the **ophelia** user with the following information. When finished, click **Save** to create the user.

Field	Value
First Name	Ophelia
Last Name	Dunham
Email	ophelia@lab.example.com
Username	ophelia
Password	redhat123
Confirm Password	redhat123
User Type	Normal User
Organization	Default

3. Create an automation controller team called **Operations** using **Ops Team** as the description. The new team must belong to the **Default** organization.

- 3.1. Go to **Access > Teams** and then click **Add**.
- 3.2. Create the team using the following information. When finished, click **Save** to create the team.

Field	Value
Name	Operations
Description	Ops Team
Organization	Default

- 4.** Add Oliver as a Member to the Operations team.
 - 4.1. Go to **Access > Teams** and then click the link for the **Operations** team.
 - 4.2. Click the **Access** tab and then click **Add**.
 - 4.3. Select **Users** and then click **Next**.
 - 4.4. Select **oliver** and then click **Next**.
 - 4.5. Select **Member** and then click **Save** to assign the role.
 - 4.6. The **Access** page for the Operations team indicates that Oliver has the **Member** user role and the **Read** team role.
- 5.** Add Ophelia as an Admin to the Operations team.
 - 5.1. From the **Access** page for the Operations team, click **Add**.
 - 5.2. Select **Users** and then click **Next**.
 - 5.3. Select **ophelia** and then click **Next**.
 - 5.4. Select **Admin** and then click **Save** to assign the role.
 - 5.5. The **Access** page for the Operations team indicates that Ophelia has the **Admin** user role and the **Read** team role.
- 6.** Add a private automation hub group called **Container Developers**. Assign all the permissions in the **Containers** category to the new **Container Developers** group.
 - 6.1. Go to <https://hub.lab.example.com> and log in as the **admin** user with **redhat** as the password.
 - 6.2. Go to **User Access > Groups** and then click **Create**. Enter **Container Developers** in the **Name** field and then click **Create** to create the group.
 - 6.3. Click **Edit**. Add all the permissions in the **Containers** category and then click **Save**.
- 7.** Add a private automation hub user using the information in the following table. When finished, log out of the private automation hub web UI.

Field	Value
Username	carlos
First name	Carlos
Last name	Cortez
Email	carlos@lab.example.com
Password	redhat123
Password confirmation	redhat123
Groups	Container Developers
User type	Not a super user

- 7.1. Go to **User Access > Users** and then click **Create**. Create the **carlos** user with information in the previous table. When finished, click **Save** to create the private automation hub user.
8. From a terminal window, use the **skopeo** command to identify the version and release of the private automation hub **ansible-automation-platform-22/ee-supported-rhel8:latest** container image.
To do this, access the container repository on **hub.lab.example.com** as the newly created **carlos** user.
 - 8.1. Use the **skopeo login** command to log in to private automation hub as **carlos** using **redhat123** as the password.

```
[student@workstation ~]$ skopeo login hub.lab.example.com
Username: carlos
Password: redhat123
Login Succeeded!
```



Note

If running the **skopeo login** command produces a message indicating that you are already logged in, then run the **skopeo logout hub.lab.example.com** command and then attempt to log in again as the **carlos** user.

- 8.2. Reduce the amount of typing in the following **skopeo** commands by setting a variable:

```
[student@workstation ~]$ REPO=ansible-automation-platform-22/ee-supported-rhel8
```

- 8.3. Use the **skopeo inspect** command to display information about the **ansible-automation-platform-22/ee-supported-rhel8:latest** container image on private automation hub. Use the **--format "{{ .Labels.version }}-{{ .Labels.release }}**" option to only display the version and release of the container image.

```
[student@workstation ~]$ skopeo inspect docker://hub.lab.example.com/${REPO} \
> --format "{{ .Labels.version }}-{{ .Labels.release }}"
1.0.0-99
```

**Important**

When not specified, container images default to using the **latest** tag.

9. Using the private automation hub web UI, add a new tag to the `ansible-automation-platform-22/ee-supported-rhel8:latest` container image. Use information about the version and release of the container image obtained in the previous step to create the tag using the format of "VERSION-RELEASE". For example, the `ansible-automation-platform-22/ee-29-rhel8:latest` container image would use the `1.0.0-119` tag, where `1.0.0` is the version, and `119` is the release.
 - 9.1. Go to Execution Environments > Execution Environments and then click the link for the `ansible-automation-platform-22/ee-supported-rhel8` container repository.
 - 9.2. From the **Images** tab, click the vertical ellipsis icon and select **Manage tags**. Add the `1.0.0-99` tag to `ansible-automation-platform-22/ee-supported-rhel8`, save your changes, and close the **Manage tags** window.
 - 9.3. Go to admin > Logout to exit the private automation hub web UI.

**Note**

Instead of using the private automation hub web UI, you could add the `1.0.0-99` container image tag by using the `skopeo copy` command:

```
[student@workstation ~]$ skopeo copy docker://hub.lab.example.com/${REPO} \
> docker://hub.lab.example.com/${REPO}:1.0.0-99
...output omitted...
```

10. Use the `skopeo list-tags` command to verify that the `ansible-automation-platform-22/ee-supported-rhel8` container repository uses the `1.0.0-99` tag.

```
[student@workstation ~]$ skopeo list-tags docker://hub.lab.example.com/${REPO}
{
    "Repository": "hub.lab.example.com/ansible-automation-platform-22/ee-
supported-rhel8",
    "Tags": [
        "1.0",
        "1.0.0",
        "1.0.0-99",
        "latest"
    ]
}
```

Evaluation

On the **workstation** machine, change to the **student** user home directory and use the **lab** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade org-review
```

Finish

On the **workstation** machine, change to the **student** user home directory and use the **lab** command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish org-review
```

Summary

- Automation controller organizations represent logical collections of users, teams, projects, and inventories.
- Automation controller users must be assigned one of three user types: **System Administrator**, **System Auditor**, or **Normal User**.
- The **System Administrator** and **System Auditor** user types grant read/write and read-only access to all automation controller objects, respectively.
- Automation controller teams make it easier to assign roles on automation controller resources to a set of users.
- Roles assigned to automation controller users and teams control what access users have on objects.
- Private automation hub controls access to objects through users and groups.
- After creating a new private automation hub group, assign permissions to the group and then add users to the group.

Chapter 3

Managing Inventories and Machine Credentials

Goal

Create inventories of machines to manage, and configure credentials necessary for automation controller's execution nodes to log in and run Ansible jobs on those systems.

Sections

- Creating a Static Inventory (and Guided Exercise)
- Creating Machine Credentials for Access to Inventory Hosts (and Guided Exercise)

Lab

- Managing Inventories and Machine Credentials

Creating a Static Inventory

Objectives

- Create a static inventory of managed hosts, using the web UI.

Red Hat Ansible Inventory

An Ansible inventory is a list of managed hosts in your infrastructure on which you run your automation tasks. Those managed hosts can be organized into groups that can be used to specify subsets of managed hosts on which you run particular plays. The inventory can also be used to define host and group variables that apply to those hosts and are used by your plays.

With `ansible-navigator`, you might have defined your inventory in a text file, in either INI or YAML format, and then specified which inventory file to use on the command line or with an `ansible.cfg` file. The following is an example of an inventory file in INI format.

```
[production_servers]
prod1.example.com
prod2.example.com

[database_servers]
db_server01.example.com

[production_servers:vars]
apache_listen_port=8080
apache_root_path=/var/www/mywebdocs/

[all:vars]
ansible_user=student
ansible_ssh_private_key_file=/home/student/.ssh/id_rsa
```

To run automation jobs with automation controller, you must specify an inventory for the job. There are several ways to do this, including:

- By configuring the inventory through the web UI.
- By managing the inventory as a static file in a Git repository.
- By dynamically generating the inventory from an external source.

This section covers how to configure, inspect, and edit a static inventory using the automation controller web UI.

Creating an Inventory Using the Automation Controller Web UI

Automation controller manages inventories as objects. Each organization might have many inventories available. Users can create job templates and configure those job templates to use a specific inventory belonging to the organization. Access to an inventory object on the automation controller depends on the roles a user has been assigned for the inventory.

**Important**

Your automation controller license determines the maximum number of hosts that you can manage with automation controller.

If you have multiple hosts in your inventory that have the same name, such as `webserver1`, they count for licensing purposes as a single node. Note that this differs from the **Hosts** count on the Dashboard, which counts hosts in separate inventories separately.

In the automation controller web UI, click **Settings** in the left pane and select **Subscription settings** from the **Settings** page to verify how many hosts your license supports and how many are remaining.

Creating a New Inventory

Use the following procedure to create inventories within each organization:

1. Log in to the automation controller web UI as the **admin** user, or as a user assigned the **Admin** or **Inventory Admin** role for the organization under which you intend to create the inventory.
2. Click **Resources > Inventories** and then click **Add**.
3. Select **Add inventory**.
4. Enter a name and a description for the new inventory in the **Name** and **Description** fields, respectively.
5. Use the **Default** organization, or click the search icon to select a different organization for the inventory.
6. Similarly, click the search icon under **Instance Groups** to select instance groups.
7. Click **Save**.

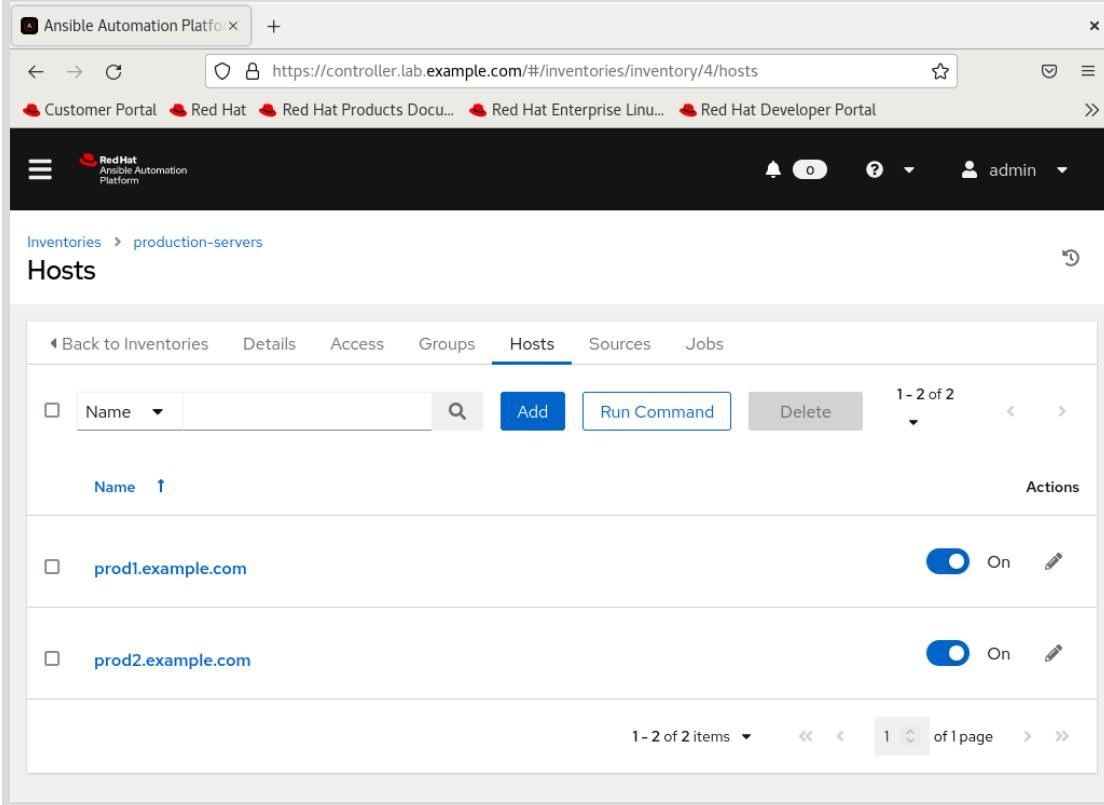
Creating a Host Group in an Inventory

1. On the inventory **Details** page, click **Groups** and then click **Add**.
2. Enter a name and description for the new group in the **Name** and **Description** fields, respectively.
3. Click **Save**.

Creating Hosts in an Inventory

1. On the **Group Details** page, click **Hosts** and then click **Add**.
2. Select **Add new host**.
3. Enter a name and description for the new host in the **Name** and **Description** fields, respectively.
4. Click **Save**.

Chapter 3 | Managing Inventories and Machine Credentials



The screenshot shows the Ansible Automation Platform web interface. The URL in the browser is <https://controller.lab.example.com/#/inventories/inventory/4/hosts>. The page title is "Ansible Automation Platform". The top navigation bar includes links for Customer Portal, Red Hat, Red Hat Products Documentation, Red Hat Enterprise Linux, and Red Hat Developer Portal. The user is logged in as "admin". The main content area is titled "Hosts" under "Inventories > production-servers". The "Hosts" tab is selected, showing two hosts: "prod1.example.com" and "prod2.example.com". Both hosts have their status set to "On". The interface includes a search bar, an "Add" button, a "Run Command" button, and a "Delete" button. Pagination at the bottom indicates there are 1-2 of 2 items, with 1 page shown.

Figure 3.1: Inventory details

Inventory Roles

You can assign appropriate RBAC roles to users and teams for an inventory, in order to grant users the ability to read, use, or manage it.

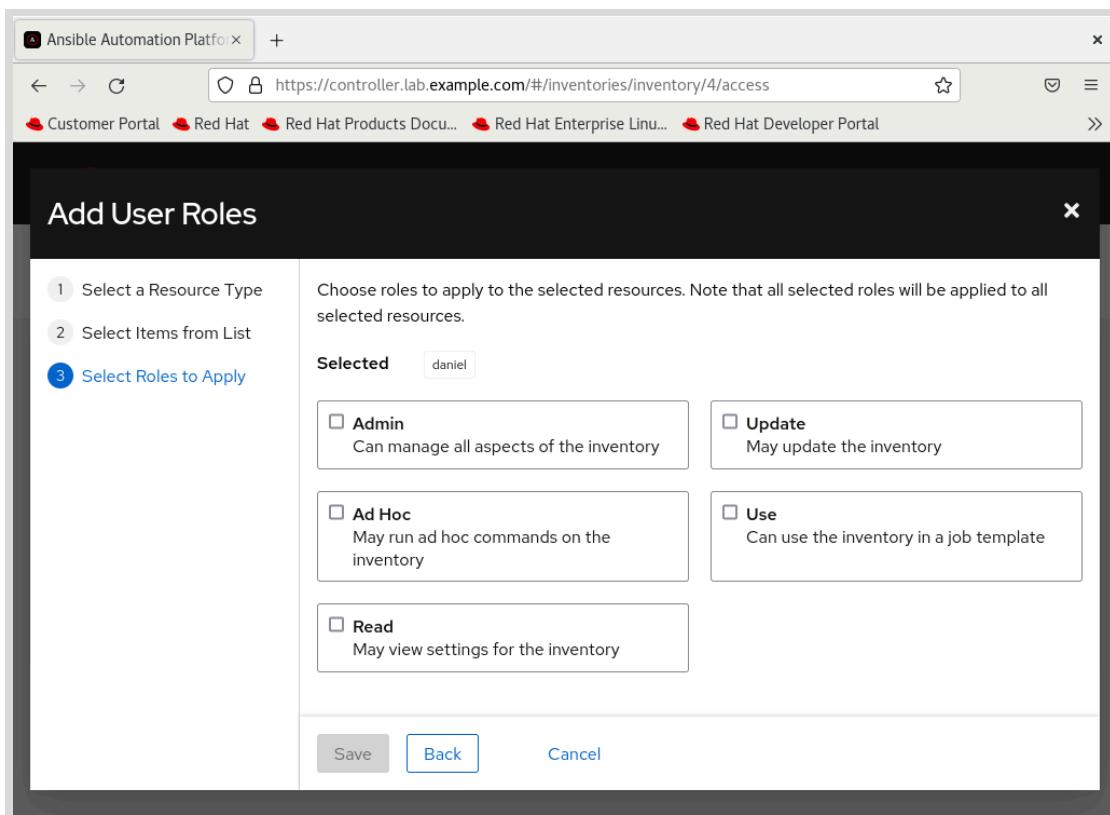


Figure 3.2: User Roles

The following is a list of available roles for an inventory:

Roles	Description
Admin	Grants users full permissions over an inventory. These permissions include deleting and modifying an inventory. In addition, this role also grants permissions associated with the inventory roles Use, Ad Hoc, and Update.
Update	Grants users the ability to update a dynamic inventory from its external data source.
Ad Hoc	Grants users the ability to use the inventory to execute ad hoc commands. Using Ansible automation controller for ad hoc command execution is discussed in detail in the Ansible automation controller User Guide.
Use	Grants users the ability to use an inventory in a job template resource. This controls which inventory is used to start jobs using the job template's playbook.
Read	Grants users the ability to view the contents of an inventory.

Chapter 3 | Managing Inventories and Machine Credentials

When you create an inventory, it is only accessible by users who have the **Admin**, **Inventory Admin**, or **Auditor** roles for the organization to which the inventory belongs. You have to configure access to all other users.

First, you need to create the inventory and save it, and after that, you can assign users and teams appropriate roles as discussed previously.

Assigning Roles

Use the following procedure to assign users and teams roles on an inventory:

1. Log in to the automation controller web UI as the **admin** user or as a user assigned the **Admin** or **Inventory Admin** role for the organization in which you intend to modify the permissions of an inventory.
2. Click **Resources > Inventories**.
3. Click the inventory.
4. Click **Access** and then click **Add**.
5. Click either **Users** or **Teams** and then click **Next**.
6. Select the users or teams from the list and then click **Next**.
7. Select one or more roles by clicking on them.
8. Click **Save** to complete the new roles.

Inventory Variables

Ansible supports inventory variables that apply values to variables on particular hosts or host groups.

When you manage a static inventory in the Ansible automation controller web UI, you can define inventory variables directly in the inventory object.

On the **Inventories** page, set the variables by clicking the **Edit inventory**(pencil) icon next to the inventory name. On the **Details** page for the inventory, you can set variables that affect all hosts in the inventory:

Chapter 3 | Managing Inventories and Machine Credentials

The screenshot shows the Ansible Automation Platform interface. At the top, there's a navigation bar with links to 'Customer Portal', 'Red Hat', 'Red Hat Products Documentation', 'Red Hat Enterprise Linux', and 'Red Hat Developer Portal'. Below the navigation is a header with the Red Hat logo and the text 'Ansible Automation Platform'. The main content area is titled 'Edit details' under 'Inventories > production-servers'. It includes fields for 'Name *' (set to 'production-servers'), 'Description' (set to 'Production servers inventory'), 'Organization *' (set to 'Default'), and 'Instance Groups' (empty). A 'Variables' section is expanded, showing a code editor with the following YAML:

```
1 ---  
2 apache_listen_port: 8080  
3 apache_root_path: /var/www/mywebdocs/  
4 |
```

Below the code editor, a note says 'Press Enter to edit. Press ESC to stop editing.'

Figure 3.3: Variables for all hosts

When creating a host group within an inventory, you can define the group variables using YAML or JSON in the **Variables** field on the **Create new group** page.

You can also set the group variables by clicking on the **Edit group** (pencil) icon next to the host group's name in the inventory. These variables apply to all hosts that are part of the group:

Chapter 3 | Managing Inventories and Machine Credentials

The screenshot shows the 'Edit details' page for a host group named 'prod-servers'. The 'Name' field is set to 'prod-servers' and the 'Description' field is 'group for prod'. The 'Variables' section is active, showing the following YAML code:

```
1 ---  
2 ntp_server: ntp.atlanta.example.com  
3 proxy: proxy.atlanta.example.com  
4 |
```

A note at the bottom of the 'Variables' input area says 'Press Enter to edit. Press ESC to stop editing.' At the bottom of the dialog are 'Save' and 'Cancel' buttons.

Figure 3.4: Variables for a host group

Likewise, you can define the host variables using either YAML or JSON in the **Variables** field on the **Create new host** page when an individual host is created within an inventory. Alternatively, click the **Edit host** (pencil) icon next to the name of the host in the inventory to define the host variables. Variables defined in this manner only apply to the specific host.

Name * prod1.example.com

Description prod1 server

Variables [YAML](#) [JSON](#)

```
1 ---
2 ansible_user: student
3 ansible_ssh_private_key_file: /home/student/.ssh/id_rsa
4
5
```

Press Enter to edit. Press ESC to stop editing.

Save Cancel

Figure 3.5: Variables for an individual host



Important

Other variables with higher precedence override inventory variables. Extra variables defined in a job template and playbook variables have higher precedence than inventory variables.



References

How to build your inventory

https://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html#intro-inventory

A discussion of how node counting works for licenses is available at

<https://docs.ansible.com/ansible-tower/latest/html/administration/licensing-support.html#node-counting-in-licenses>

► Guided Exercise

Creating a Static Inventory

Create a static inventory in automation controller and grant teams permission to administer and use that inventory.

Outcomes

- Create a static inventory containing hosts and a host group.
- Assign appropriate inventory permissions to a team.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command ensures that automation controller is installed and configured with any necessary resources created in previous exercises. The command also creates an additional inventory, with hosts and a host group, needed for this exercise.

```
[student@workstation ~]$ lab start host-inventory
```

Instructions

- ▶ 1. Navigate to `https://controller.lab.example.com` and log in as the `admin` user with `redhat` as the password.
- ▶ 2. Create a static inventory named `Prod`.
 - 2.1. Navigate to `Resources > Inventories` and then click `Add > Add inventory`.
 - 2.2. On the `Create new inventory` page, enter the following details:

Field	Value
Name	Prod
Description	Production Inventory
Organization	Default

- 2.3. Click `Save` to create the new inventory. You are redirected to the inventory `Details` page for the `Prod` inventory.
- ▶ 3. Create the `prod_servers` group in the `Prod` inventory.
 - 3.1. Click the `Groups` tab and then click `Add` to add a new group.
 - 3.2. On the `Create new group` page, enter the following details:

Field	Value
Name	prod_servers
Description	Production servers

- 3.3. Click **Save** to create the new group.
- ▶ 4. Add the `servere.lab.example.com` host to the `prod_servers` group in the `Prod` inventory.
 - 4.1. Click the **Hosts** tab to add hosts to the group you just created.
 - 4.2. Click **Add > Add new host** to add a new host to the group.
 - 4.3. On the **Create new host** page, enter the following details:

Field	Value
Name	<code>servere.lab.example.com</code>
Description	Server E
 - 4.4. Click **Save** to add the new host.
- ▶ 5. Assign the `Operations` team the `Admin` role on the `Prod` inventory.
 - 5.1. Navigate to **Resources > Inventories** and then click the link for the `Prod` inventory entry.
 - 5.2. On the **Access** tab, click **Add** to add permissions.
 - 5.3. Click **Teams** and then click **Next** to display the list of available teams.
 - 5.4. Select the `Operations` team, and then click **Next** to display the possible roles to apply.
 - 5.5. Select the `Admin` role.
 - 5.6. Click **Save** to finalize the role assignment. This redirects you to the list of permissions for the `Prod` inventory, which now shows that all members of the `Operations` team are assigned the `Admin` role on the `Prod` inventory.
- ▶ 6. Verify access to the `Prod` inventory with the `oliver` user, who belongs to the `Operations` team.
 - 6.1. Log out and log back in as the `oliver` user using `redhat123` as the password. This user is assigned the `Member` role for the `Operations` team.
 - 6.2. Navigate to **Resources > Inventories** and then click the link for the `Prod` inventory created earlier.
 - 6.3. Review the contents of the `Prod` inventory to see the hosts and group it contains.

Chapter 3 | Managing Inventories and Machine Credentials

- 7. Add the `serverf.lab.example.com` host to the `Prod` inventory while logged in as the `oliver` user.
- 7.1. Click **Resources > Inventories** and then click the link for the `Prod` inventory.
 - 7.2. On the **Groups** tab, click the link for the `prod_servers` group to enter the group.
 - 7.3. On the **Hosts** tab, click **Add > Add new host** to add a new host to the group.
 - 7.4. On the **Create new host** page, enter the following details, and then click **Save**:

Field	Value
Name	<code>serverf.lab.example.com</code>
Description	Server F

- 8. Assign the `Use` role on the `Test` inventory to the `Developers` team.
- 8.1. Log out and log back in as the `admin` user with `redhat` as the password.
 - 8.2. Navigate to **Resources > Inventories** and click the link for the `Test` inventory.
 - 8.3. On the **Access** tab, click **Add** to add permissions.
 - 8.4. Click **Teams** and then click **Next** to display the list of available teams.
 - 8.5. Select the `Developers` team, and then click **Next** to display the possible roles to apply.
 - 8.6. Select the `Use` role and then click **Save**. This redirects you to the list of permissions for the `Test` inventory, which now shows that normal members of the `Developers` team are assigned the `Use` role on the `Test` inventory. The `dannie` user does not inherit this permission because `dannie` only has `read` access on the `Developers` team.
- 9. Verify access to the `Test` inventory with the `daniel` user, who belongs to the `Developers` team.
- 9.1. Log out and then log back in as the `daniel` user with the password `redhat123`. This user is assigned the `Admin` role for the `Developers` team.
 - 9.2. Navigate to **Resources > Inventories** and click the link for the `Test` inventory.
 - 9.3. Review the contents of the `Test` inventory to see the hosts and group it contains. Note that even though `daniel` is an `Admin` of the `Developers` team, he cannot manage the `Test` inventory because you only granted the `Use` role for the inventory to the `Developers` team.

**Note**

This is representative of a real world scenario where developers might have access to the list of systems in the testing environment but are not able to modify the list.

- 9.4. Log out of the automation controller web UI.

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish host-inventory
```

Creating Machine Credentials for Access to Inventory Hosts

Objectives

- Create a machine credential for inventory hosts to allow automation controller to run jobs on those inventory hosts.

Storing Secrets in Credentials

Credentials are the automation controller objects used to authenticate to remote systems. They provide secrets, such as passwords and SSH keys, or other supporting information needed to successfully access or use a remote resource.

The automation controller maintains secure storage for the secrets in these credential objects. Credential passwords and keys are encrypted before they are saved to the automation controller database, and cannot be retrieved in clear text from the automation controller user interface. Although users and teams can be assigned privileges to use credentials, the secrets are not exposed to them. This means that when a user changes teams or leaves the organization, the credentials and systems do not need to be rekeyed. When a credential is needed by automation controller, it decrypts the data internally and passes it to SSH or other program directly.



Important

After you have entered sensitive authentication data into a credential and encrypted, it can no longer be retrieved in decrypted form through the automation controller web UI.

Credential Types

The automation controller can manage several different types of credentials, including:

Ansible Galaxy/Automation Hub API Token

Create this type of resource so that automation controller can download Ansible Content Collections and roles from Ansible Galaxy, automation hub, and private automation hub.

After creating the credential, you must enable the credential for an organization. During this process, you can specify a precedence by ordering the credentials.

Container Registry

Create this type of credential if you need to authenticate before you can pull down a container image from a container registry, such as `registry.redhat.io` or a private automation hub.

GitHub Personal Access Token

GitHub no longer supports password-based authentication to synchronize projects with the HTTPS protocol. To continue using HTTPS (rather than SSH), create and use a GitHub personal access token (PAT), and then create an automation controller credential that uses the personal access token.

Machine

Automation controller can use this credential type to access and make changes to the managed hosts. Later sections discuss the machine credential in more detail.

Network

Create this type of credential to use Ansible network modules to manage networking equipment. This credential specifies a username and either a password or the content of the user's SSH private key.

Source Control

Create this type of credential to synchronize project resources from a remote repository. Specify a username and either a password or the user's SSH private key.

Vault

Create this type of credential to decrypt files that have been encrypted with Ansible Vault.

Several types of inventory credentials are available to update dynamic inventory information from one of the automation controller's built-in dynamic inventory sources. Separate credential types exist for each inventory source in question: Amazon Web Services, VMware vCenter, Red Hat Satellite 6, Google Compute Engine, Microsoft Azure Resource Manager, OpenStack, and so on.

**Note**

Automation controller administrators can create custom credential types that can be used like the built-in credential types, specified using a YAML definition. Find more information about how to create a new credential type in the References at the end of this section.

Creating Machine Credentials

This section focuses on how to set up machine credentials that provide appropriate login and privilege escalation information for use with hosts in an inventory. Some other types of credentials are discussed in more detail elsewhere in this course.

You can manage credentials through the **Resources > Credentials** page in the automation controller web UI.

Any user can create a credential, and is considered the owner of credentials that they create. If a credential is not assigned to an organization, it is a *private* credential. This means that only the user that owns the credential, and users with the automation controller **System Administrator** user type can use it, and only those users and users with the automation controller **System Auditor** user type can see it.

On the other hand, if the credential is assigned to an organization, then it is an *organization* credential. Only the automation controller **System Administrator** users and users with the **Admin** role on an organization can create credentials assigned to an organization. Users and teams in the organization can be granted roles on a credential assigned to the organization to use or manage the credential.

In summary, the main distinctions between private credentials and organization credentials are:

- Any user can create a private credential, but only automation controller **System Administrator** users and users with the **Admin** role on an organization can create an organization credential.
- If a credential belongs to an organization, users and teams can be granted roles on it, and it can be shared. Private credentials not assigned to an organization can only be used by the owner and the automation controller **System Administrator** users. Other users and teams cannot be granted roles.

Important

The automation controller Admin user can assign an organization to an existing private credential, converting a private credential into an organization credential.

The following procedure details how to create machine credentials.

1. Log in to the automation controller web UI as a user with the appropriate role assignment. If you are creating a private credential, there are no specific role requirements. If you are creating an organization credential, log in as a user with the Admin role for the organization.
2. Click **Resources > Credentials**.
3. Click **Add** to create a new credential.
4. Enter a name and description for the new credential in the **Name** and **Description** fields, respectively.
5. If you have organization Admin privileges, you can click the search icon to select the organization to which you assign this credential in order to make it an organization credential. If you do not have Admin privileges in an organization and you attempt to select it, automation controller displays the message "You do not have permission to perform this action", and you cannot save the credential.
If you do not have Admin privileges, do not select any organization. This creates a private credential.
6. For machine credentials, additional fields appear in the **Type Details** section:

The screenshot shows the 'Create New Credential' page. At the top, there are fields for 'Name *' and 'Description'. Below that, there are fields for 'Organization' (with a search icon) and 'Credential Type *' (set to 'Machine'). A 'Type Details' section follows, containing fields for 'Username' and 'Password' (with a 'Prompt on launch' checkbox), and an 'SSH Private Key' section with a file upload input and browse/clear buttons. The entire form is set against a light gray background.

Figure 3.6: Creating a new machine credential

Chapter 3 | Managing Inventories and Machine Credentials

These fields can contain the information needed to authenticate to and escalate privileges on the machines that use this credential. Many of them are mapped to settings, which might be in an `ansible.cfg` file:

1. **Username** is the username used to log in to the managed hosts (`remote_user` in `ansible.cfg`).
2. **Password** is the SSH password to use for this user. Leave this blank if you use SSH private key authentication.
3. The **SSH Private Key** field contains an SSH private key that you use to log in as **Username** on the managed hosts. It is easier to cut and paste the text from the file rather than to manually type it in. If you are administering automation controller from a Firefox web browser running under GNOME 3 or higher, you can drag and drop the private key file from the **Files** application window in the desktop environment into this field in your web browser window.

When you save the credential, the automation controller encrypts it. The **Password** or **SSH Private Key** field reads `$encrypted$` instead of displaying the credential.

4. You can use the **Private Key Passphrase** field when the key in **SSH Private Key** is encrypted by SSH for additional protection. The field accepts a passphrase that can decrypt the key. Otherwise, this field can be blank.
5. **Privilege Escalation Method** is a drop-down menu that specifies what type of privilege escalation (`become_method`), if any, that is needed for playbooks run on the machines accessed by this credential.

For sudo privilege escalation, **Privilege Escalation Username** is the privileged user that Ansible should use on the managed host (`become_user`). **Privilege Escalation Password** is the sudo password to use. This can be blank if no password is needed.

6. Click **Save** to create the new machine credential.

Editing Machine Credentials

When you create a machine credential, you can edit it, if needed. The following procedure details how to modify a machine credential.

1. Log in as a user with the appropriate role assignment. If editing a private credential, then log in as the user who created the credential. If editing an organization credential, then log in as a user with **Admin** role on the organization credential.
2. Click **Resources > Credentials**.
3. Click the name of the credential to edit.
4. On the credential editor screen, make the necessary changes to the desired credential properties.
5. Click **Save**.

Credential Roles

As discussed earlier, private credentials (credentials that are not assigned to an organization) are only accessible to their creators or to users that have the **System Administrator** or **System Auditor** user type. Other users cannot be assigned roles on private credentials.

To assign roles to credentials, the credential must have an organization. Users and teams in that organization can then share that credential through role assignments.

Chapter 3 | Managing Inventories and Machine Credentials

The following is the list of available credential roles.

Credential role	Description
Admin	Grants users full permissions on a credential. These permissions include deleting and modifying the credential, as well as the ability to use the credential in a job template.
Use	Grants users the ability to use a credential in a job template. How to use a credential in a job template is discussed later in this course.
Read	Grants users the ability to view the details of a credential. Does not allow these users to decrypt the secrets which belong to that credential through the web UI.

Managing Credential Access

When you create an organization credential, it is only accessible by the owner and users with either the Admin or Auditor role in the organization in which you created it. You must configure additional access, if desired.

You cannot assign additional roles to the credential until you save it, after which you can set them by editing the credential.

Use the following procedure to grant permissions to an existing organization credential:

1. Log in as a user with Admin role on the organization in which the credential was created.
2. Click **Resources > Credentials** and then click the name of the credential to edit.
3. On the credential editor page, on the **Access** tab, click **Add** to add permissions.
4. Click either **Users** or **Teams**, and then click **Next** to select the users or teams to be assigned roles.
5. Click **Next** to display the list of available roles to apply.
6. Click **Save**.



Important

Permissions for credentials can also be added through either the user or team management pages.

Common Credential Scenarios

To help you understand ways in which credentials are used, the following sections describe some common credential scenarios.

Credentials Protected by Automation Controller, Not Known to Users

One common scenario for the use of automation controller credentials is the delegation of task execution from administrators to Tier 1 support staff.

For example, suppose that the support staff needs to be delegated the ability to run a playbook ensuring a web application has been restarted to restore service when outages occur outside of

Chapter 3 | Managing Inventories and Machine Credentials

business hours. The support staff's credential uses a shared account, `support`, and a passphrase-protected private key for SSH authentication to managed hosts. The `support` account needs to escalate privileges using `sudo`, with a `sudo` password, in order to run the playbook.

Because the credential is shared by the support staff's team, an organization credential resource should be created to store the username, SSH private key, and SSH key passphrase needed to authenticate SSH sessions to the managed hosts. The credential also stores the privilege escalation method, username, and `sudo` password information. Once created, the credential can be used by the support staff to launch jobs on the managed hosts without needing to know the SSH key passphrase or `sudo` password.

The screenshot shows the 'Create New Credential' form in Red Hat Ansible Tower. The 'Name' field is set to 'Web Support' and is marked as required (*). The 'Description' field contains the text 'Credential for the support staff'. The 'Organization' dropdown is set to 'Default'. The 'Credential Type' dropdown is set to 'Machine'. In the 'Type Details' section, the 'Username' field is set to 'support'. The 'Password' field is a masked input. A checkbox labeled 'Prompt on launch' is checked. In the 'SSH Private Key' section, there is a file input field containing a private key file named '-----BEGIN OPENSSH PRIVATE KEY-----'. The file content is partially visible, showing a long string of characters starting with 'b3BlbnNzaC1rZXktdjEAAAABG5vb...'. There are also 'Browse...', 'Clear', and 'Key' icons associated with the file input.

Figure 3.7: Organization credential for shared account

Credential Prompts for Sensitive Password, Not Stored in Automation Controller

Another scenario is to use credentials to store username authentication information while still prompting interactively for a sensitive password when the credential is used.

Suppose that a database administrator wants to run a playbook managed in automation controller to execute tasks on a database server that houses sensitive data for the company finances. Due to the highly sensitive nature of the data, the company's financial compliance regulations forbid the storage of the account's password.

You can still use a machine credential to configure the database administrator's authentication to the database server. Because the credential is not to be shared, you can use a private credential to store the **SSH Username** and configure it by selecting the **Prompt on launch** checkbox for **Password**. This prompts the user for the account's password when a job uses the credential.

The screenshot shows the 'Create New Credential' page. At the top, there's a 'Name *' field containing 'Finance DBA' and a 'Description' field with the placeholder 'Credential with password prompting'. Below that, 'Organization' is set to 'Default' and 'Credential Type *' is 'Machine'. In the 'Type Details' section, 'Username' is 'findba' and 'Password' has a checked 'Prompt on launch' option. There's also an 'SSH Private Key' section with a placeholder 'Drag a file here or browse to upload' and buttons for 'Browse...' and 'Clear'.

Figure 3.8: Private credential with password prompting



Important

Automation controller has a feature that allows playbooks to be run automatically on a schedule, much like a cron job in Linux. Credentials configured to prompt interactively for password information at runtime cannot be used with scheduled jobs, because automation controller cannot provide that information without user interaction.



References

Ansible Documentation: Credentials

<https://docs.ansible.com/automation-controller/latest/html/userguide/credentials.html>

Ansible Documentation: Custom Credential Types

https://docs.ansible.com/automation-controller/latest/html/userguide/credential_types.html

► Guided Exercise

Creating Machine Credentials for Access to Inventory Hosts

Create a machine credential and assign roles to teams that permit members of those teams to use that credential.

Outcomes

- Create a machine credential.
- Assign roles to the machine credential.

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that the automation controller is installed and configured with any necessary resources created in previous exercises.

```
[student@workstation ~]$ lab start host-credential
```

Instructions

- ▶ 1. Navigate to <https://controller.lab.example.com> and log in as the `admin` user with `redhat` as the password.
- ▶ 2. Create a new credential called `Operations`.
 - 2.1. Navigate to Resources > Credentials.
 - 2.2. Click **Add** to add a new credential.
 - 2.3. On the next page, fill in the details as follows:

Field	Value
Name	Operations
Description	Operations Credential
Organization	Default
Credential Type	Machine
Username	devops
Password	redhat
Privilege Escalation Method	sudo
Privilege Escalation Username	root

**Note**

Because the devops user does not need to enter a password to run sudo commands, you do not need to enter a password in the **Privilege Escalation Password** field.

- 2.4. Leave the other fields untouched and click **Save** to create the new credential.
- ▶ 3. Assign the Operations team the Admin role on the Operations credential.
 - 3.1. Navigate to **Resources > Credentials**.
 - 3.2. Click the **Operations** credential and then click the **Access** tab.
 - 3.3. Click **Add** to add access permissions.
 - 3.4. Click **Teams**, and then click **Next**.
 - 3.5. Select the **Operations** team and then click **Next**.
 - 3.6. Select the **Admin** role.
 - 3.7. Click **Save**. This redirects you to the list of access permissions for the **Operations** credential, which now shows that all members of the **Operations** team, **oliver** and **ophelia**, are assigned the **Admin** role on the **Operations** credential.
- ▶ 4. Verify access of the Admin role to the Operations credential with the **oliver** user, who belongs to the Operations team.
 - 4.1. Log out and log back in as **oliver** using **redhat123** as the password. This user is assigned the **Member** role for the **Operations** team.
 - 4.2. Navigate to **Resources > Credentials** and then click the link for the **Operations** credential that you created earlier.
 - 4.3. Notice that the **oliver** user can modify the credential.
- ▶ 5. Assign the Developers team the Use role on the Operations credential.

Chapter 3 | Managing Inventories and Machine Credentials

- 5.1. Log out and log back in as the `admin` user with `redhat` as the password.
 - 5.2. Navigate to **Resources > Credentials**.
 - 5.3. Click the **Operations** credential and then click the **Access** tab.
 - 5.4. Click **Add** to add access permissions.
 - 5.5. Click **Teams** and then click **Next**.
 - 5.6. Select the **Developers** team and then click **Next**.
 - 5.7. Select the **Use** role.
 - 5.8. Click **Save**. This redirects you to the list of access permissions for the **Operations** credential, which now shows that all members of the **Developers** team, `daniel` and `david`, are assigned the **Use** role on the **Operations** credential.
- ▶ **6.** Verify the **Use** role for the **Operations** credential with the `daniel` user, who belongs to the **Developers** team.
- 6.1. Log out and log back in as `daniel` using `redhat123` as the password. This user has an **Admin** role for the **Developers** team.
 - 6.2. Navigate to **Resources > Credentials**.
 - 6.3. Click the **Operations** credential and then click the **Access** tab. Notice that the `daniel` user cannot modify the credential even though he has an **Admin** role for the team.
 - 6.4. Log out of the automation controller web UI.

Finish

On the **workstation** machine, change to the **student** user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish host-credential
```

► Lab

Managing Inventories and Machine Credentials

Create inventories and credentials and assign roles to teams that permit members of those teams to manage the new inventories.

Outcomes

- Manage inventories.
- Manage credentials.
- Allow a team to run a playbook against an inventory.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command ensures that private automation hub and automation controller are installed and configured with any necessary resources created in previous exercises.

```
[student@workstation ~]$ lab start host-review
```

Instructions

1. Create a new automation controller inventory called Dev within the Default organization. Use Development Inventory as the description. Navigate to <https://controller.lab.example.com> and log in as the admin user with redhat as the password.
2. Create a host group called dev_servers in the Dev inventory. Use Development servers as the description.
3. Add two hosts with the hostnames servera.lab.example.com and serverb.lab.example.com to the dev_servers host group. Use Server A as the description for the servera.lab.example.com host and use Server B as the description for the serverb.lab.example.com host.
4. Grant the Admin role on the Dev inventory to the Developers team.
5. Create a new machine credential with the following information:

Developers Credential

Field	Value
Name	Developers
Description	Developers Credential
Organization	Default
Credential Type	Machine
Username	devops
Password	redhat
Privilege Escalation Method	sudo
Privilege Escalation Username	root

6. Grant the Admin role on the Developers credential to the Developers team.

Evaluation

On the workstation machine, change to the student user home directory and use the lab command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade host-review
```

Finish

On the workstation machine, change to the student user home directory and use the lab command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish host-review
```

► Solution

Managing Inventories and Machine Credentials

Create inventories and credentials and assign roles to teams that permit members of those teams to manage the new inventories.

Outcomes

- Manage inventories.
- Manage credentials.
- Allow a team to run a playbook against an inventory.

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that private automation hub and automation controller are installed and configured with any necessary resources created in previous exercises.

```
[student@workstation ~]$ lab start host-review
```

Instructions

1. Create a new automation controller inventory called `Dev` within the `Default` organization. Use `Development Inventory` as the description. Navigate to `https://controller.lab.example.com` and log in as the `admin` user with `redhat` as the password.
 - 1.1. Navigate to `https://controller.lab.example.com` and log in as the `admin` user with `redhat` as the password.
 - 1.2. Navigate to `Resources > Inventories` and then click `Add`. Select `Add inventory`.
 - 1.3. Create the new inventory using the following information, and then click `Save`.

Field	Value
Name	Dev
Description	Development Inventory
Organization	Default

2. Create a host group called `dev_servers` in the `Dev` inventory. Use `Development servers` as the description.
 - 2.1. From the `Dev` inventory details page, click the `Groups` tab and then click `Add`.

Chapter 3 | Managing Inventories and Machine Credentials

- 2.2. Create a new host group called `dev_servers` and use `Development servers` as the description.
- 2.3. Click **Save**.
3. Add two hosts with the hostnames `servera.lab.example.com` and `serverb.lab.example.com` to the `dev_servers` host group. Use `Server A` as the description for the `servera.lab.example.com` host and use `Server B` as the description for the `serverb.lab.example.com` host.
 - 3.1. From the `dev_servers` group details page, click the **Hosts** tab and then click **Add**. Select **Add new host**.
 - 3.2. Create the `servera.lab.example.com` host and use `Server A` as the description.
 - 3.3. Click **Save**.
 - 3.4. Return to the `dev_servers` group details page. Navigate to **Resources > Inventories** and then click the link for the `Dev` inventory. Click the **Groups** tab and then click the link for the `dev_servers` group.
 - 3.5. From the `dev_servers` group details page, click the **Hosts** tab and then click **Add**. Select **Add new host**.
 - 3.6. Create the `serverb.lab.example.com` host and use `Server B` as the description.
 - 3.7. Click **Save**.
4. Grant the `Admin` role on the `Dev` inventory to the `Developers` team.
 - 4.1. Navigate to **Resources > Inventories** and then click the link for the `Dev` inventory.
 - 4.2. Click the **Access** tab and then click **Add**.
 - 4.3. Select **Teams** and then click **Next**.
 - 4.4. Select **Developers** and then click **Next**.
 - 4.5. Select **Admin** and then click **Save**. You are redirected to the list of permissions for the `Dev` inventory, which now shows that all members of the `Developers` team are assigned the `Admin` role on the `Dev` inventory.
5. Create a new machine credential with the following information:

Developers Credential

Field	Value
Name	Developers
Description	Developers Credential
Organization	Default
Credential Type	Machine
Username	devops
Password	redhat
Privilege Escalation Method	sudo
Privilege Escalation Username	root

- 5.1. Navigate to **Resources > Credentials** and then click **Add**.
- 5.2. Create a new machine credential using information from the **Developers Credential** table. When finished, click **Save**.



Note

Because the devops user does not need to enter a password to run sudo commands, you do not need to enter a password in the **Privilege Escalation Password** field.

6. Grant the **Admin** role on the **Developers** credential to the **Developers** team.
 - 6.1. Navigate to **Resources > Credentials** and then click the link for the **Developers** credential.
 - 6.2. Click the **Access** tab and then click **Add**.
 - 6.3. Select **Teams** and then click **Next**.
 - 6.4. Select **Developers** and then click **Next**.
 - 6.5. Select **Admin** and then click **Save**. You are redirected to the list of permissions for the **Developers** credential, which now shows that all members of the **Developers** team are assigned the **Admin** role on the **Developers** credential.

Evaluation

On the **workstation** machine, change to the **student** user home directory and use the **lab** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade host-review
```

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish host-review
```

Summary

- You can configure static inventories on automation controller through its web UI.
- Inventory resources on automation controller manage Ansible inventories of hosts and groups with inventory variables.
- You can assign roles to users and teams for inventories to grant users the ability to read, use, or manage the inventories.
- Credentials store authentication information, such as for machines, network devices, source control, and dynamic inventories.
- Machine credentials allow automation controller to authenticate, access, and enable privilege escalation.
- You can share an organization credential by granting roles to users and teams.
- Only the owner and the automation controller **System Administrator** users can use a private credential.

Chapter 4

Managing Projects and Launching Ansible Jobs

Goal

Create projects and job templates in the web UI, and use them to launch Ansible Playbooks that are stored in Git repositories in order to automate tasks on managed hosts.

Sections

- Creating a Project for Ansible Playbooks (and Guided Exercise)
- Creating Job Templates and Launching Jobs (and Guided Exercise)
- Managing Projects and Launching Ansible Jobs

Lab

Creating a Project for Ansible Playbooks

Objectives

- Create a project in automation controller that uses playbooks and other project materials from an existing Git repository.

Automation Controller Projects

A Red Hat Ansible project represents at least one playbook that is usually stored in a remote version control system, typically using Git. The project might also include additional playbooks, inventories, and inventory variable files. If the project requires Ansible Content Collections or roles, then the project might also include them, or it might include `requirements.yml` files that specify them and from where automation controller can download them.

Automation controller can automatically download and get updates for project materials and the resources specified by `requirements.yml` files when a job template that uses the project is launched.

Creating a Project

You can use the following procedure to create a project to share a collection of Ansible Playbooks and roles managed in an existing Git repository.

1. Log in to the automation controller web UI as a user with either the **Admin** or **Project Admin** role in an organization.
2. Navigate to **Resources > Projects** and then click **Add** to open the **Create New Project** page.
3. Enter a unique name for the project in the **Name** field. If desired, enter a description in the **Description** field.
4. Click the search icon next to the **Organization** field to display a list of organizations within automation controller. Select an organization from the list and then click **Select**. By default, new projects are assigned to the **Default** organization.
5. Specify the default execution environment used for jobs in this project (optional).
6. In the **Source Control Type** drop-down menu, choose **Git**.
7. In the **Type Details** section, enter the location of the Git repository in the **Source Control URL** field. In the **Source Control Branch/Tag/Commit** field, specify the branch, tag, or commit of the repository to use for the project (optional).
8. If authentication is required to access the Git repository, click the search icon next to the **Source Control Credential** field to display a list of available source code management (SCM) credentials. Select an SCM credential from the list and click **Select**. The creation of SCM credentials is discussed later in this section.
9. Select the desired action to be taken to update the project against its SCM source. Available options are **Clean**, **Delete**, **Update Revision on Launch**, and **Allow Branch Override**. These four options are discussed in further detail at the end of this section.
10. Click **Save**.

Project Roles

Users are granted permissions to project resources through assigned roles on the project resource. Roles can be assigned directly to a user or indirectly through a team. For example, for a user to gain permissions on a specific project, they must be assigned or inherit a role for that project.

The following project roles are available:

Admin

The Admin role grants users full access to a project. Users with this role can modify or delete the project, including its permissions. This role also grants users the Use, Update, and Read roles, which are discussed later in this section.

Use

The Use role grants users the ability to use a project in a template resource. The use of a project in a template resource is discussed in detail in a later section. This role also grants users the permissions associated with the project Read role.

Update

The Update role enables users to schedule or manually update project materials from its SCM source. This role also grants users the permissions associated with the project Read role.

Read

The Read role enables users to view the details, permissions, and notifications associated with a project.

Managing Project Access

Initially, only users with the System Administrator or System Auditor user type can access projects. You must specifically configure access for users with the Normal User user type. You cannot assign roles when you create the project; create the project first and then add any roles.

You can assign roles in the Access section of the project editor. Use the following procedure to set roles for a project:

1. Log in as a user with the Admin role for the organization in which the project was created.
2. Navigate to Resources > Projects and then click the name of the project.
3. Click the Access tab and then click Add to start adding permissions.
4. Select either Users or Teams and then click Next.
5. Select the users or teams to be granted permissions and then click Next to display the list of project roles and their definitions.
6. Select the desired project roles for each user or team and then click Save.



Note

You can also add roles for projects through either the user or the team management pages.

Creating SCM Credentials

Source control management credentials, also called SCM credentials, store authentication information that automation controller can use to access project materials stored in a version

Chapter 4 | Managing Projects and Launching Ansible Jobs

control system such as Git. SCM credentials store the username and the password or private key (and private key passphrase, if any) needed to authenticate access to the source control repository.

Use the following procedure to create an SCM credential so that automation controller can retrieve playbooks, roles, or other materials from a Git repository for a project.

1. Log in as a user with the appropriate role assignment.
 - To create a private SCM credential, there are no specific role requirements.
 - To create an SCM credential that belongs to an organization, log in as a user with the **Admin** role for the organization.
2. Navigate to **Resources > Credentials** and then click **Add** to display the **Create New Credential** page.
3. Enter the required information for the new credential.
 - a. Enter a unique name for the credential in the **Name** field.
 - b. If you are creating an organization credential, click the search icon next to the **Organization** field, select the organization to create the credential in, and then click **Select**. Skip this step if creating a private credential.
 - c. In the **Credential Type** field, select the **Source Control** credential type. This exposes the **Type Details** section.
 - d. Enter authentication data into the appropriate fields. For example, you might need to specify a username and password for your account on the version control system. If you are using an SSH private key to authenticate to the version control system, either copy and paste or drag and drop your private key into the **SCM Private Key** field. That key can be a passphrase-encrypted SSH private key, in which case you must provide the passphrase to automation controller in the **Private Key Passphrase** field.
4. Click **Save** to create the new SCM credential.

The screenshot shows a web-based form titled 'Create New Credential'. At the top left is a 'Credentials' link. The main title is 'Create New Credential'. There are three input fields: 'Name *' (with a required asterisk), 'Description', and 'Organization'. Below these is a dropdown menu labeled 'Credential Type *' with 'Source Control' selected. A section titled 'Type Details' contains 'Username' and 'Password' fields, each with a keyhole icon indicating they are password fields. Below this is a 'SCM Private Key' section with a file upload input field, a 'Browse...' button, a 'Clear' button, and a keyhole icon.

Figure 4.1: Creating a new SCM credential

SCM Credential Roles

Like machine credentials, private SCM credentials are only usable by their creators and System Administrator and System Auditor users. However, you can share SCM credentials with other users by assigning the credentials to an organization and assigning appropriate roles on those credentials to teams or users in that organization.

The following is the list of roles for SCM credentials:

Admin

The Admin role grants users full permissions over an SCM credential. These permissions include deleting and modifying the SCM credential. This role also grants users permissions associated with the credential Use and Read roles.

Use

The Use role grants users the ability to associate an SCM credential with a project resource. This role also grants users the permissions associated with the credential Read role.

The Use role does not control whether a user can themselves use the SCM credential to update a project, only whether they can assign that SCM credential so that it can then be used by someone who has the Update role on the project.

For example, if an SCM credential is associated with a project, any user assigned the Update role on the project can use the associated SCM credential without being granted the Use role on the credential.

Read

The Read role grants users the ability to view the details of an SCM credential.

Managing Access to SCM Credentials

Initially, only users with the **Admin** or **Auditor** organization role can access SCM credentials within their organization. You need to specifically configure any additional user access.

The assignment of SCM credential roles to users or teams dictates who has permissions to an SCM credential belonging to an organization. You cannot assign these permissions when you create the SCM credential. You can adjust them after their creation by editing the credential.

You can assign roles through the **Access** section of the credential editor page. Use the following procedure to grant permissions to an existing SCM credential that is assigned to an organization.

1. Log in as a user with the **Admin** role in the organization to which the SCM credential belongs.
2. Navigate to **Resources > Credentials** and then click the name of the SCM credential.
3. Click the **Access** tab and then click **Add** to start adding permissions.
4. Select either **Users** or **Teams** and then click **Next**.
5. Select the users or teams to which you are granting permissions, and then click **Next** to display the list of credential roles and their definitions.
6. Select the desired credential roles for each user or team and then click **Save**.



Note

You can also add permissions for SCM credentials through either the user or team management pages.

Updating Projects

An SCM project resource in automation controller represents a copy of playbooks, collections, and roles obtained from an SCM source. Because modifications to the contents of these playbooks, collections, and roles are managed in an external SCM system, their respective counterparts in an automation controller project must be routinely updated from the SCM source to reflect changes.

The following sections describe different ways to update SCM project resources in automation controller.

Clean

This option removes local modifications before getting the latest revision from the source control repository.

Delete

This option completely removes the local project repository on automation controller before getting the latest revision from the source control repository. This takes longer than the **Clean** option for large repositories.

Allow Branch Override

This option allows overriding the branch, tag, or reference when executing job templates that use a given project.

Update Revision on Launch

This option updates the project from the source control repository each time the project is used to launch a job. Automation controller creates a separate job to update the project.

Manual Updates

If you do not want to use these automatic settings, you can manually update a project to the latest version in the source control repository.

Use the following procedure to manually update a project from its SCM source:

1. Log in as a user with the **Update** role in the project.
2. Navigate to **Resources > Projects** and then click the **Sync Project** icon for the project that you want to update.

If the update succeeds, then the status column displays the **Successful** message and the revision column displays the latest commit hash.

Name	Status	Type	Revision	Actions
Demo Project	Successful	Git	347e44f	

Figure 4.2: Updating an SCM project



Important

The **Update** role only dictates whether a user can manually trigger an update of a project from its SCM source. It does not impact the update behavior configured by the project's SCM update option.

For example, a project configured with the **Update on Launch** SCM update option still performs updates even when the project is used by a user who has not been granted the **Update** role on the project.

Support for Ansible Content Collections and Roles

A project can include a copy of the Ansible Content Collections or roles that it needs. It can be a better practice, however, to store those Ansible Content Collections or roles in a separate repository, especially if they are managed by other people, and have automation controller automatically retrieve that content as needed. These sources could be automation hub, the community Ansible Galaxy site, or your own Git repository server, for example.

Chapter 4 | Managing Projects and Launching Ansible Jobs

You manage this by creating a `requirements.yml` file, just like the one used by the `ansible-galaxy` command. At the end of a project update, if a project repository includes a valid `roles/requirements.yml` file, then automation controller automatically installs the defined roles. The output of the project update job contains a task similar to the following:

```
...output omitted...
TASK [fetch galaxy roles from requirements.(yml/yaml)] *****
changed: [localhost] => (item=/var/lib/awx/projects/_13_example_project/roles/
requirements.yml)
...output omitted...
```

Similarly, if a project repository includes a valid `collections/requirements.yml` file, then automation controller automatically installs the defined collections. The output of the project update job contains a task similar to the following:

```
...output omitted...
TASK [fetch galaxy collections from collections/requirements.(yml/yaml)] *****
changed: [localhost] => (item=/var/lib/awx/projects/_13_example_project/
collections/requirements.yml)
...output omitted...
```



References

Automation Controller User Guide

<https://docs.ansible.com/automation-controller/latest/html/userguide/index.html>

Install Multiple Collections with a Requirements File

https://docs.ansible.com/ansible/latest/galaxy/user_guide.html#install-multiple-collections-with-a-requirements-file

Install Multiple Roles from a File

https://docs.ansible.com/ansible/latest/galaxy/user_guide.html#installing-multiple-roles-from-a-file

► Guided Exercise

Creating a Project for Ansible Playbooks

Create a new source control credential and a new project, and assign a role to one of your teams that allows the team to use that project.

Outcomes

- Create a source control credential.
- Create a new project.
- Assign roles to the project.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command ensures that automation controller is installed and configured with any necessary resources created in previous exercises.

```
[student@workstation ~]$ lab start provision-project
```

Instructions

- ▶ 1. Navigate to <https://controller.lab.example.com> and log in as the `admin` user with `redhat` as the password.
- ▶ 2. Create a new source control credential to use in the new project.
 - 2.1. Navigate to **Resources > Credentials** and then click **Add** to display the **Create New Credential** page.
 - 2.2. Enter the following details, and then click **Save**:

Field	Value
Name	Student Git Credential
Organization	Default
Credential Type	Source Control
Username	git
SCM Private Key	(Content of the /home/student/.ssh/gitlab_rsa file.)

**Note**

If you choose to browse for the /home/student/.ssh/gitlab_rsa file, then right-click anywhere in the directory navigation and select **Show Hidden Files**. With this option enabled you can see the .ssh directory within the /home/student directory.

- 3. Create a new project called `My Webservers DEV`.

3.1. Navigate to **Resources > Projects** and then click **Add** to display the **Create New Project** page.

3.2. Enter the following details, and then click **Save**:

Field	Value
Name	<code>My Webservers DEV</code>
Description	Development Webservers Project
Organization	Default
Source Control Type	Git
Source Control URL	<code>git@git.lab.example.com:git/my_webservers_DEV.git</code>
Source Control Credential	Student Git Credential

After you click **Save**, automation controller immediately attempts to synchronize the project.

3.3. On the **Details** tab, verify that the value of **Last Job Status** shows **Successful**.

**Important**

If the project synchronization fails, then ensure that the project specifies the correct source control URL and the correct source control credential. Ensure that the source control credential uses the correct values and then attempt the project synchronization again.

- 4. Assign the **Developers** team the **Admin** role on the `My Webservers DEV` project.

- 4.1. Navigate to **Resources > Projects** and then click the link for the `My Webservers DEV` project entry.
- 4.2. Click the **Access** tab to manage the permissions for the `My Webservers DEV` project.
- 4.3. Click **Add** to start adding permissions.
- 4.4. Click **Teams** and then click **Next** to display the list of available teams.
- 4.5. Select the **Developers** team and then click **Next** to display the possible roles to apply.

Chapter 4 | Managing Projects and Launching Ansible Jobs

- 4.6. Select the Admin role and then click **Save**. This redirects you to the list of permissions for the My Webservers DEV project, which now shows that all members of the Developers team are assigned the Admin role on the My Webservers DEV project.

Finish

On the **workstation** machine, change to the **student** user home directory and use the **lab** command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish provision-project
```

Creating Job Templates and Launching Jobs

Objectives

- Create and manage a job template that specifies a project and playbook, an execution environment from a private automation hub, an inventory, and credentials that you can use to launch Red Hat Ansible Platform jobs on managed hosts.

Job Templates

In automation controller, a job template is a predefined collection of settings that you can use to launch jobs that run playbooks. Running a playbook from the command line requires a playbook, inventory, and authentication. Similarly, a job template associates a playbook from a project with an inventory of hosts, credentials for authentication, an automation execution environment, and other parameters used when you launch an Ansible job to run that playbook.

Whether a user can launch jobs or create job templates with particular projects and inventories depends on the roles that you have assigned them. When granted the **Use** role, users can use a job template to associate projects with inventories.

A job template defines the parameters for the execution of an Ansible job. An Ansible job executes a playbook against a set of managed hosts. Therefore, a job template must define which project provides the playbook and which inventory contains the list of managed hosts.

The job template can also specify an automation execution environment to use when running the playbook. For example, if you are using a playbook that worked with Ansible 2.9, you can specify a compatible automation execution environment to run the playbook.

Additionally, the job template must also define the machine credential to use to authenticate to the managed hosts. Like projects and inventories, users must have the **Use** role assigned to a machine credential before they can associate it with a job template.

After it has been defined, a job template allows for the repeated execution of a job and is therefore ideal for routine execution of tasks. Because the project, inventory, automation execution environment, and machine credential parameters are part of the job template definition, the job runs the same way each time.

Creating Job Templates

Unlike other automation controller resources, job templates do not directly belong to an organization. Instead, they are used by a project that belongs to an organization. The relationship of a job template to an organization is determined by the project that it uses. Therefore, you do not need the **Admin** role in an organization to create a job template. Instead, you only need the **Use** role for the project that you assign to the job template.

Because a job template has to be defined with an inventory, project, automation execution environment, and machine credential, a user can only create a job template if they have **Use** roles assigned to one or more of each of these three automation controller resources.

The following procedure to create a job template focuses on mandatory parameters; optional parameters are discussed later:

Chapter 4 | Managing Projects and Launching Ansible Jobs

1. Log in to the automation controller web UI as a user who has been assigned the Use role for the inventory, project, and machine credential resources for the project that you are creating.
2. Navigate to Resources > Templates and then click Add > Add job template to open the Create New Job Template page.
3. Enter a name for the job template in the Name field.
4. In the Job type list, choose Run.
5. Click the search icon for the Inventory field, select the required inventory, and then click Select.
6. Click the search icon for the Project field, select the desired project, and then click Select.
7. Click the drop-down menu for the Playbook field and select the desired playbook.
8. Click the search icon for the Credentials field, select the desired credential, and then click Select.
9. Scroll to the bottom of the page and click Save.

The screenshot shows the 'Create New Job Template' page. At the top, there are tabs for 'Templates' and 'Create New Job Template'. Below the tabs, there are several input fields and dropdown menus:

- Name ***: A text input field.
- Description**: A text input field.
- Job Type ***: A dropdown menu set to 'Run'.
- Inventory ***: A search input field with a magnifying glass icon.
- Project ***: A dropdown menu set to 'Demo Project'.
- Execution Environment**: A search input field with a magnifying glass icon.
- Playbook ***: A dropdown menu set to 'Select a playbook'.
- Credentials**: A search input field with a magnifying glass icon.
- Labels**: A search input field with a magnifying glass icon.

There are also several checkboxes labeled 'Prompt on launch' located next to the input fields.

Figure 4.3: Creating a new job template

Modifying Job Execution

A job template has other settings you can use to adjust how automation controller runs the playbook when you launch the template:

Description

Use this field to specify an optional description of the job template.

Execution Environment

Use this field to specify an automation execution environment for the job template. Click the search icon and select the automation execution environment from the list. This field is available after you have selected a project for the job template. When you specify an

Chapter 4 | Managing Projects and Launching Ansible Jobs

automation execution environment, it overrides the one that is used when launching the job template. You can verify the automation execution environment used by the job template in its details page.

Labels

Labels are names that you can attach to job templates to help you group or filter job templates.

Variables

You can use this field to pass extra command-line variables to the playbook executed by a job. You define these extra variables as key-value pairs using either YAML or JSON.

Forks

Use this field to specify the `forks` setting that controls the number of parallel processes to allow during playbook execution. Set this to 0 to use the default setting from the Ansible configuration file.

Limit

Use this field to restrict the list of managed hosts provided by the inventory that you defined for the job template.

Verbosity

This determines the level of detail generated in the output of the job run.

Job Slicing

Use this field to specify the number of slices into which you want to divide the work done by this job template.

Timeout

Use this field to specify the time in seconds to run the job before it is canceled. A value of 0 causes no job timeout.

Show Changes

Enable this option to show the changes made by Ansible tasks.

Instance Groups

Use this field to specify the instance groups on which you want to run the job template.

Job Tags

This field accepts a comma-separated list of tags that exist in a playbook. Use tags to identify distinct portions of a playbook. By specifying a list of tags in this field, you can selectively execute only certain portions of a playbook.

Skip Tags

This field accepts a comma-separated list of tags that exist in a playbook. By specifying a list of tags in this field, you can selectively skip certain portions of a playbook during its execution.

Enable Privilege Escalation

Select this option to run the playbook with escalated privileges.

Provisioning Callbacks

Select this option to create a provisioning callback URL on automation controller, which hosts can use to request a configuration update using the job template.

Enable Webhook

Select this option to enable the ability to interact with GitHub or GitLab.

Concurrent Jobs

Select this option to allow for multiple, simultaneous executions of this job template.

Chapter 4 | Managing Projects and Launching Ansible Jobs

Enable Fact Storage

Select this option to save the gathered facts to be viewed at the host level.

Prompting for Job Parameters

When executing playbooks from the command line, administrators can modify playbook execution with command-line options. Automation controller provides some of this flexibility by allowing certain parameters in job templates to prompt for user input at the time of job execution. The `Prompt on launch` option is available for the following parameters:

- Job Type
- Inventory
- Credentials
- Variables
- Limit
- Verbosity
- Show Changes
- Job Tags
- Skip Tags

The flexibility to change job parameters at the time of job execution encourages playbook reuse. For example, rather than creating multiple job templates to run the same playbook on different sets of managed hosts, a single job template that has the `Prompt on launch` option enabled for the inventory field is sufficient. When the job is launched, the user executing the job is given the option to specify an inventory to execute the playbook on. When prompted, users can only select from inventories that they have been assigned the `Use` role on.

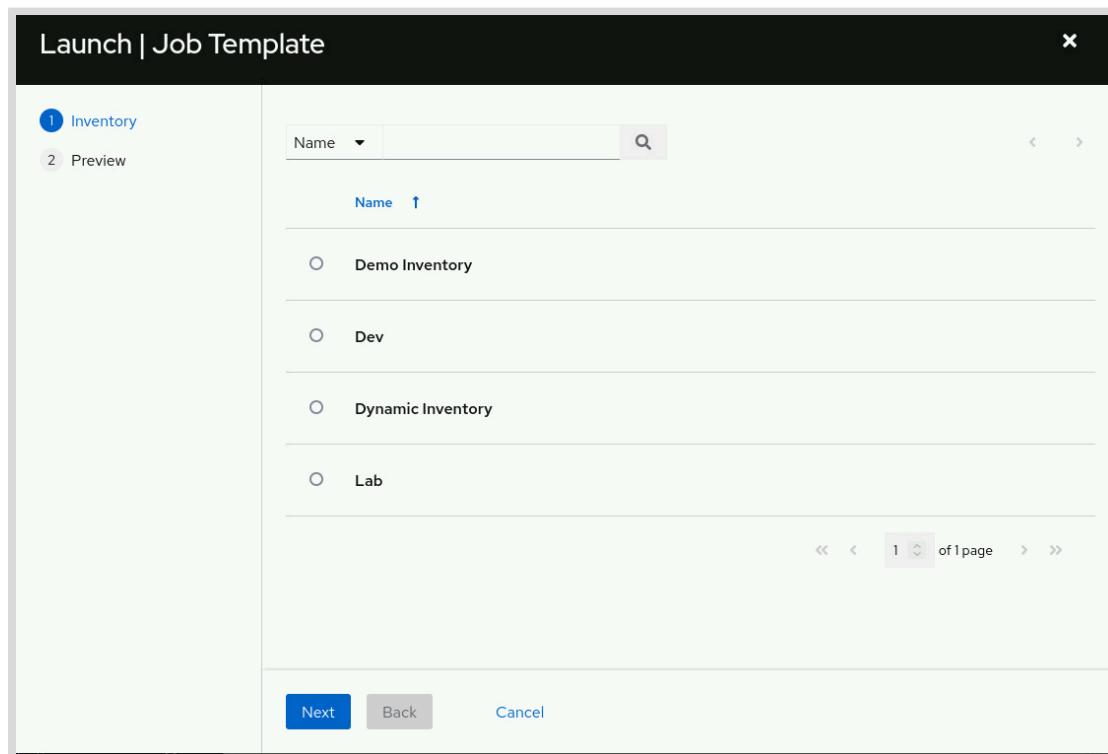


Figure 4.4: Prompting for inventory on launch

Job Template Roles

Three roles are available to control user access to job templates.

Role	Description
Admin	Users with the Admin role can delete a job template or edit its properties, including its associated permissions. This role also grants permissions associated with the job template Execute and Read roles.
Execute	Users with the Execute role can execute a job using the job template. The users can also schedule a job using the job template. This role also grants permissions associated with the job template Read role.
Read	Users with the Read role have read-only access to the properties of a job template. They can also view other information related to the job template, such as the list of jobs executed using the job template, as well as its associated permissions and notifications.

**Important**

A job template makes use of other automation controller resources such as projects, inventories, and credentials. For a user to execute a job using a job template, they only need the Execute role on the job template; they do not need the Use roles on any of these associated automation controller resources.

Managing Job Template Access

When you create a job template, it is only accessible to you or users with either an Admin or Auditor role in the organization where the project was created. Additional access must be specifically configured if desired.

The assignment of the previously discussed job template roles to users or teams dictates who has permissions to a job template.

You cannot assign these permissions when you create the job template; create the job template first and then assign any required permissions.

You can assign roles through the **Access** section of the job template page. Use the following procedure to grant permissions to a job template:

1. Log in as a user with the Admin role in the organization that the job template is associated with or as the user who created the job template.
2. Navigate to **Resources > Templates** and click the name of the job template.
3. On the **Access** page, click **Add** to start adding permissions.
4. Click either **Users** or **Teams** and then click **Next**.
5. Select the users or teams to be granted permissions, and then click **Next** to display the list of job template roles and their definitions.
6. Select the desired job template role for the user or team and then click **Save** to finalize the changes to permissions.

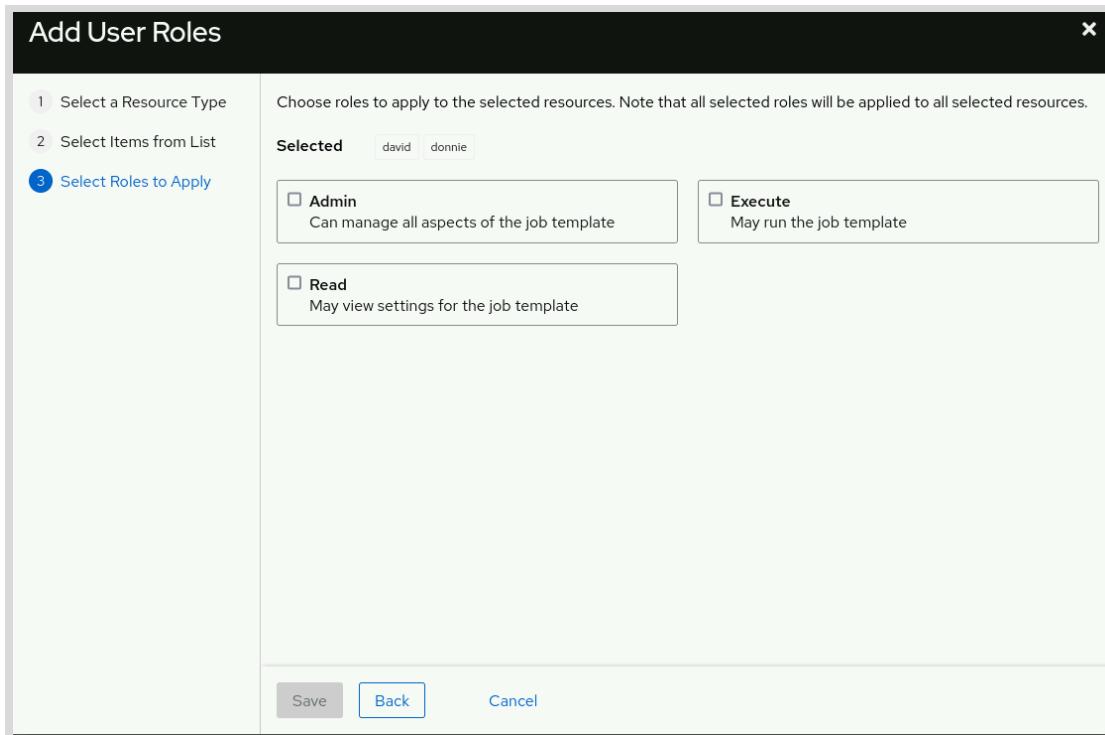
Chapter 4 | Managing Projects and Launching Ansible Jobs

Figure 4.5: Assigning job template roles to users

Launching Jobs

Use the following procedure to launch a job after you have created a job template:

1. Log in as a user with the **Execute** role on the desired job template.
2. Navigate to **Resources > Templates** to display the list of templates.
3. Click the Launch Template icon under the **Actions** column of the desired job template to launch the job.
4. If any of the job template parameters have the **Prompt on launch** option enabled, then you are prompted for input prior to job execution. Enter the desired input for each parameter and then click **Launch**.

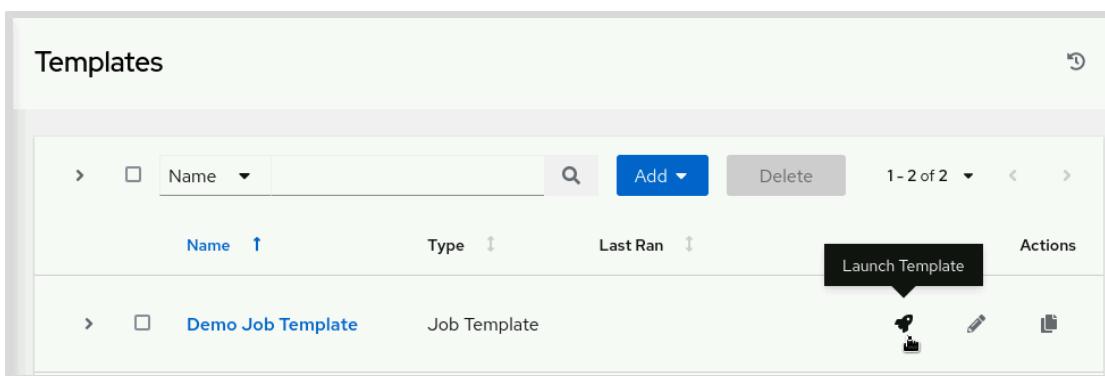


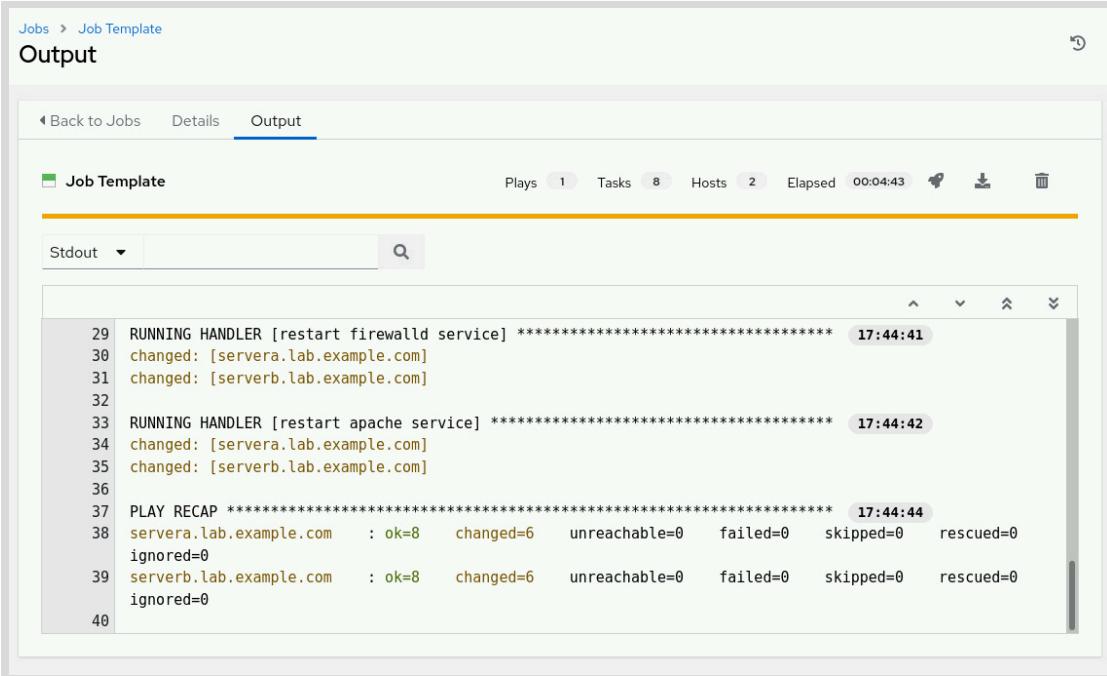
Figure 4.6: Launching a job

Evaluating the Results of a Job

After you have launched a job from a job template in the automation controller web UI, you are automatically redirected to the job page. Alternatively, navigate to **Views > Jobs** to see the list of executed jobs and click the link for the job of interest.

The job page has two tabs. The **Details** tab displays the details of the job parameters. The **Output** tab displays the output of the playbook executed by the job.

The job output tab displays information similar to that seen when you run the playbook from the command line, but includes additional information as well. This page includes the number of plays and tasks that were executed, the number of hosts that the job was executed against, and the time it took for the job to execute. It also includes three icons: to launch the job, to download the output, and to delete the job.



```
29 RUNNING HANDLER [restart firewalld service] ****
30 changed: [servera.lab.example.com]
31 changed: [serverb.lab.example.com]
32
33 RUNNING HANDLER [restart apache service] ****
34 changed: [servera.lab.example.com]
35 changed: [serverb.lab.example.com]
36
37 PLAY RECAP ****
38 servera.lab.example.com : ok=8    changed=6    unreachable=0    failed=0    skipped=0    rescued=0
39 ignored=0
40 serverb.lab.example.com : ok=8    changed=6    unreachable=0    failed=0    skipped=0    rescued=0
41 ignored=0
```

Figure 4.7: Job run results



References

Ansible Documentation: Job Templates

https://docs.ansible.com/automation-controller/latest/html/userguide/job_templates.html

► Guided Exercise

Creating Job Templates and Launching Jobs

Create a job template, assign a role to a team so that team members can use that job template, and use that job template to launch a job.

Outcomes

- Create a job template.
- Assign roles to the job template.
- Launch a job from the automation controller web UI.

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that automation controller is installed and configured with any necessary resources created in previous exercises.

```
[student@workstation ~]$ lab start provision-job
```

Instructions

- ▶ 1. Review the automation controller resources created by the `lab` command for this exercise.
 - 1.1. Navigate to `https://controller.lab.example.com` and log in as the `admin` user with `redhat` as the password.
 - 1.2. Navigate to **Resources > Projects** and then click the **DB Project** link. This is the Git project that contains the playbooks to set up the database servers.
 - 1.3. Review the **Source Control URL** and **Source Control Credential** fields.
 - 1.4. Navigate to **Resources > Credentials** and then click the link for the **Developers** credential. This is the machine credential used for running tasks on the hosts.
 - 1.5. Navigate to **Resources > Inventories** and then click the **DB Inventory** link. This is the inventory for the database servers.
 - 1.6. Review the **Groups** and **Hosts** tabs. Notice that this inventory has one host group, `db_servers`, and one host, `serverd.lab.example.com`.
- ▶ 2. Create a new job template to set up the database servers.
 - 2.1. Navigate to **Resources > Templates** and then click **Add > Add job template** to open the **Create New Job Template** page.
 - 2.2. Enter the following details and then click **Save**:

Field	Value
Name	Database setup
Description	Setup database servers
Job Type	Run
Inventory	DB Inventory
Project	DB Project
Playbook	create_samples_db.yml
Credentials	Developers

- 3. Assign the Admin role to the Developers team on the Database setup job template.
- 3.1. Navigate to **Resources > Templates** and then click the Database setup job template link.
 - 3.2. Click the **Access** tab to manage the permissions for the Database setup job template.
 - 3.3. Click **Add** to open the **Add Roles** dialog box.
 - 3.4. Click **Teams** and then click **Next** to display the list of available teams.
 - 3.5. Select the Developers team, and then click **Next** to display the possible roles to apply.
 - 3.6. Select the Admin role and then click **Save**. This redirects you to the list of permissions for the Database setup job template, which now shows that all members of the Developers team are assigned the Admin role on the Database setup job template.
- 4. Run the Database setup job template as the user `daniel`, who is a member of the Developers team. This attempt to run the job template fails. You examine the cause of failure in the next step.
- 4.1. Log out as the `admin` user and log in as the `daniel` user with `redhat123` as the password.
 - 4.2. Navigate to **Resources > Templates** and then click the **Launch Template** icon for the Database setup job template.
 - 4.3. The job fails. Review the error message in the output.
- ```
ERROR! couldn't resolve module/action 'mysql_db'. This often indicates a misspelling, missing collection, or incorrect module path.
...output omitted...
```
- 4.4. Log out of the automation controller web UI.
- 5. Review the `create_samples_db.yml` playbook and examine the reason the job failed.

**Chapter 4 |** Managing Projects and Launching Ansible Jobs

- 5.1. From a terminal, create the /home/student/git-repos directory if it does not already exist, and then change into it.

```
[student@workstation ~]$ mkdir -p ~/git-repos/
[student@workstation ~]$ cd ~/git-repos/
```

- 5.2. Clone the [https://git.lab.example.com/git/my\\_dbservers.git](https://git.lab.example.com/git/my_dbservers.git) Git repository into the /home/student/git-repos directory.

```
[student@workstation git-repos]$ git clone \
> https://git.lab.example.com/git/my_dbservers.git
Cloning into 'my_dbservers'...
...output omitted...
[student@workstation git-repos]$ cd my_dbservers
```

- 5.3. Review the content of the `create_samples_db.yml` playbook.

```

- name: Install and create a sample database
 hosts: all
 become: true
 gather_facts: false

 tasks:
 - name: Ensure MariaDB is installed
 yum:
 name:
 - mariadb
 - mariadb-server
 - python3-PyMySQL
 state: present

 - name: Ensure the mariadb service is started and enabled
 systemd:
 name: mariadb
 state: started
 enabled: true

 - name: Ensure the samples database exists
 mysql_db:
 name: samples
 state: present
```

- 5.4. In the step 4.3, the error indicated that the job failure was related to the `mysql_db` module. The `mysql_db` module is part of the `community.mysql` collection, which is not part of the `ee-supported-rhel8` execution environment. It was included in Ansible 2.9, so it is part of the `ee-29-rhel8` execution environment. This playbook was probably originally written for Ansible 2.9.

- 6. Determine which execution environment the `Database setup` job template is using.

- 6.1. Navigate to <https://controller.lab.example.com> and log in as the `admin` user with `redhat` as the password.

**Chapter 4 |** Managing Projects and Launching Ansible Jobs

- 6.2. Navigate to **Resources > Templates** and then click the **Database setup** job template link.
  - 6.3. Click the **Automation Hub Default execution environment** link.
  - 6.4. The **Details** page indicates that the execution environment is using the **hub.lab.example.com/ee-supported-rhel8:latest** image. The **community.mysql** collection is not included in this execution environment.
- 7. To resolve this issue, modify the **Database setup** job template to use the **Ansible Engine 2.9 execution environment** execution environment. This execution environment provides the modules available in Ansible 2.9, including the **mysql\_db** module.
- 7.1. Navigate to **Resources > Templates** and then click the **Edit Template** icon for the **Database setup** job template.
  - 7.2. Click the search icon for **Execution Environment**.
  - 7.3. Select **Ansible Engine 2.9 execution environment** and then click **Select**.
  - 7.4. Click **Save** to save the changes to the job template.

**Note**

An alternative solution that allows you to continue using the later execution environment is to modify the project to include or retrieve the missing **community.mysql** Ansible Content Collection, so that it is available to the execution environment.

You could embed the collection in the project's Git repository, or you could use a **requirements.yml** file to have the automation controller automatically pull it from Ansible Galaxy. For this to work, you need to make sure the execution environment has all the prerequisites needed by that collection already installed.

You also need to modify the playbook so that instead of calling the **mysql\_db** module as **mysql\_db**, you call it by its fully qualified collection name, **community.mysql.mysql\_db**.

- 8. Rerun the **Database setup** job template as the **daniel** user.
- 8.1. Navigate to **https://controller.lab.example.com**, log out as the **admin** user, and log in as the **daniel** user with **redhat123** as the password.
  - 8.2. Navigate to **Resources > Templates** and then click the **Launch Template** icon for the **Database setup** job template. After some time, the job launched by the job template succeeds.
- 9. Confirm that the job template correctly configured a MariaDB database on the **serverd.lab.example.com** server.
- 9.1. Open a terminal on the **workstation** machine, connect to the **serverd** machine, and become the **root** user.

**Chapter 4 |** Managing Projects and Launching Ansible Jobs

```
[student@workstation my_dbservers]$ ssh serverd
...output omitted...
[student@serverd ~]$ sudo -i
[sudo] password for student: student
[root@serverd ~]#
```

9.2. Confirm that the **samples** database exists.

```
[root@serverd ~]# mysql -e "SHOW DATABASES"
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| samples |
+-----+
```

9.3. Log out of the **serverd** machine.

```
[root@serverd ~]# exit
logout
[student@serverd ~]$ exit
logout
Connection to serverd closed.
[student@workstation my_dbservers]$
```

## Finish

On the **workstation** machine, change to the **student** user home directory and use the **lab** command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish provision-job
```

## ► Lab

# Managing Projects and Launching Ansible Jobs

Create a new project and job template, and assign an appropriate role for a team to be able to launch a job.

### Outcomes

- Create a project.
- Create a job template.
- Allow a team to launch a job using a job template.

### Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that automation controller is installed and configured with any necessary resources created in previous exercises.

```
[student@workstation ~]$ lab start provision-review
```

### Instructions

1. Go to <https://controller.lab.example.com> and log in as the `admin` user with `redhat` as the password.
2. Create a project called `My Webservers` with the following information:

#### New Project Details

| Field                     | Value                                                      |
|---------------------------|------------------------------------------------------------|
| Name                      | My Webservers                                              |
| Description               | Webservers Project                                         |
| Organization              | Default                                                    |
| Source Control Type       | Git                                                        |
| Source Control URL        | <code>git@git.lab.example.com:git/my_webservers.git</code> |
| Source Control Credential | Student Git Credential                                     |

3. Give the Developers team both the Use and the Update roles on the `My Webservers` project.

4. Create a job template called TEST webservers setup with the following information:

#### New Job Template Details

| Field       | Value                           |
|-------------|---------------------------------|
| Name        | TEST webservers setup           |
| Description | Setup apache on TEST webservers |
| Job Type    | Run                             |
| Inventory   | Test                            |
| Project     | My Webservers                   |
| Playbook    | apache-setup.yml                |
| Credentials | Operations                      |

5. Give the Developers team the Execute role on the TEST webservers setup job template.
6. Clone the [https://git.lab.example.com/git/my\\_webservers.git](https://git.lab.example.com/git/my_webservers.git) Git repository into the /home/student/git-repos directory.
7. Edit the ~/git-repos/my\_webservers/templates/index.html.j2 Jinja2 template file. Add the highlighted Host IP Address line so that the file has the following content. Add and commit your changes to your Git repository, and then push your changes to the remote repository.

```
Current Host: {{ ansible_facts['fqdn'] }}

Host IP Address: {{ ansible_facts['default_ipv4']['address'] }}

Server list:

{% for host in groups['all'] %}
{{ host }}

{% endfor %}
```

8. Test the project and job template permissions that you previously assigned to the Developers team. Log in as david using redhat123 as the password, update the My Webservers project, and then launch a job using the TEST webservers setup job template. When finished, log out of the automation controller web UI.
9. Verify the web content displayed on <http://serverc.lab.example.com> and <http://serverd.lab.example.com>. The web pages should display the fully qualified domain name and IP address of the current host.

## Evaluation

On the workstation machine, change to the student user home directory and use the lab command to grade your work. Correct any reported failures and rerun the command until successful. You must add, commit, and push any additional changes to the remote Git repository before grading.

```
[student@workstation ~]$ lab grade provision-review
```

## Finish

On the **workstation** machine, change to the **student** user home directory and use the **lab** command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish provision-review
```

## ► Solution

# Managing Projects and Launching Ansible Jobs

Create a new project and job template, and assign an appropriate role for a team to be able to launch a job.

### Outcomes

- Create a project.
- Create a job template.
- Allow a team to launch a job using a job template.

### Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command ensures that automation controller is installed and configured with any necessary resources created in previous exercises.

```
[student@workstation ~]$ lab start provision-review
```

### Instructions

1. Go to <https://controller.lab.example.com> and log in as the `admin` user with `redhat` as the password.
2. Create a project called `My Webservers` with the following information:

#### New Project Details

| Field                     | Value                                                      |
|---------------------------|------------------------------------------------------------|
| Name                      | My Webservers                                              |
| Description               | Webservers Project                                         |
| Organization              | Default                                                    |
| Source Control Type       | Git                                                        |
| Source Control URL        | <code>git@git.lab.example.com:git/my_webservers.git</code> |
| Source Control Credential | Student Git Credential                                     |

- 2.1. Go to **Resources > Projects** and then click **Add**.

**Chapter 4 |** Managing Projects and Launching Ansible Jobs

- 2.2. Create the project using the information in the **New Project Details** table. When finished, click **Save**.
- 2.3. Automation controller automatically attempts to synchronize the new project. On the **Details** page for the **My Webservers** project, verify that the **Last Job Status** field displays the **Successful** message.

**Important**

If the project synchronization fails, then ensure that the project specifies the correct source control URL and the correct source control credential.

3. Give the **Developers** team both the **Use** and the **Update** roles on the **My Webservers** project.
  - 3.1. Go to **Resources > Projects** and click the **My Webservers** link.
  - 3.2. Click the **Access** tab and then click **Add**.
  - 3.3. Select **Teams** and then click **Next**.
  - 3.4. Select **Developers** and then click **Next**.
  - 3.5. Select both **Use** and **Update** and then click **Save** to assign the roles. This redirects you to the list of permissions for the **My Webservers** project. The **daniel** and **david** users have the **Use** and **Update** roles because they belong to the **Developers** team.
4. Create a job template called **TEST webservers setup** with the following information:

**New Job Template Details**

| Field       | Value                           |
|-------------|---------------------------------|
| Name        | TEST webservers setup           |
| Description | Setup apache on TEST webservers |
| Job Type    | Run                             |
| Inventory   | Test                            |
| Project     | My Webservers                   |
| Playbook    | apache-setup.yml                |
| Credentials | Operations                      |

- 4.1. Go to **Resources > Templates** and click **Add > Add job template**.
- 4.2. Create the job template using the information in the **New Job Template Details** table. When finished, click **Save**.
5. Give the **Developers** team the **Execute** role on the **TEST webservers setup** job template.
  - 5.1. Go to **Resources > Templates** and click the **TEST webservers setup** link.
  - 5.2. Click the **Access** tab and then click **Add**.

**Chapter 4 |** Managing Projects and Launching Ansible Jobs

- 5.3. Select **Teams** and then click **Next**.
- 5.4. Select **Developers** and then click **Next**.
- 5.5. Select **Execute** and then click **Save** to assign the role. This redirects you to the list of permissions for the **TEST webservers setup** job template. The **daniel** and **david** users have the **Execute** role because they belong to the **Developers** team.
6. Clone the `https://git.lab.example.com/git/my_webservers.git` Git repository into the `/home/student/git-repos` directory.
  - 6.1. From a terminal, create the `/home/student/git-repos` directory if it does not already exist, and then change into it.

```
[student@workstation ~]$ mkdir -p ~/git-repos/
[student@workstation ~]$ cd ~/git-repos/
```

- 6.2. Clone the `https://git.lab.example.com/git/my_webservers.git` repository and then change to the cloned repository:
- ```
[student@workstation git-repos]$ git clone \  
> https://git.lab.example.com/git/my_webservers.git  
  Cloning into 'my_webservers'...  
  ...output omitted...  
[student@workstation git-repos]$ cd my_webservers
```
7. Edit the `~/git-repos/my_webservers/templates/index.html.j2` Jinja2 template file. Add the highlighted Host IP Address line so that the file has the following content. Add and commit your changes to your Git repository, and then push your changes to the remote repository.

```
Current Host: {{ ansible_facts['fqdn'] }} <br>Host IP Address: {{ ansible_facts['default_ipv4']['address'] }} <br>  
Server list: <br>  
{% for host in groups['all'] %}  
{{ host }} <br>  
{% endfor %}
```

- 7.1. Edit the `~/git-repos/my_webservers/templates/index.html.j2` file so that it contains the previously displayed Host IP Address line.
- 7.2. Use the `git add` command to add the modified file to the next Git commit.

```
[student@workstation my_webservers]$ git add templates/index.html.j2
```

- 7.3. Commit your changes to your local Git repository:
- ```
[student@workstation my_webservers]$ git commit -m "Added Host IP Address"
[main 373e307] Added Host IP Address
 1 file changed, 1 insertion(+)
```
- 7.4. Use the `git push` command to push the changes to the remote Git repository.

```
[student@workstation my_webservers]$ git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 524 bytes | 524.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To https://git.lab.example.com/git/my_webservers.git
 abd5989..373e307 main -> main
```

8. Test the project and job template permissions that you previously assigned to the Developers team. Log in as david using redhat123 as the password, update the My Webservers project, and then launch a job using the TEST web servers setup job template. When finished, log out of the automation controller web UI.
  - 8.1. From the automation controller web UI, click **admin > Logout** and then log in as david using redhat123 as the password.
  - 8.2. Go to **Resources > Projects** and then click the **Sync Project** icon for the My Webservers project.
  - 8.3. Go to **Resources > Templates** and then click the **Launch Template** icon for the TEST web servers setup job template.
  - 8.4. Observe the live output of the running job.
  - 8.5. Click the **Details** tab and verify that the **Status** field displays the **Successful** message. This step could take several seconds or minutes to complete depending of the size of the Git repository,
  - 8.6. From the automation controller web UI, click **david > Logout**.
9. Verify the web content displayed on <http://serverc.lab.example.com> and <http://serverd.lab.example.com>. The web pages should display the fully qualified domain name and IP address of the current host.
  - 9.1. From a terminal window, use the curl command to review the web content on <http://serverc.lab.example.com> and <http://serverd.lab.example.com>.

```
[student@workstation my_webservers]$ for X in server{c,d}.lab.example.com
> do curl http://${X}
> echo
> done
Current Host: serverc.lab.example.com

Host IP Address: 172.25.250.12

Server list:

serverc.lab.example.com

serverd.lab.example.com

Current Host: serverd.lab.example.com

Host IP Address: 172.25.250.13

Server list:

serverc.lab.example.com

serverd.lab.example.com

```

**Note**

Alternatively, review the content of both `http://serverc.lab.example.com` and `http://serverd.lab.example.com` in a graphical web browser such as Firefox.

## Evaluation

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful. You *must* add, commit, and push any additional changes to the remote Git repository before grading.

```
[student@workstation ~]$ lab grade provision-review
```

## Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish provision-review
```

# Summary

---

- An automation controller project contains one or more playbooks used to launch jobs.
- A project can get its materials from a source control repository, such as Git. The project might need to use a source control credential configured in automation controller in order to authenticate to the source control repository.
- You use job templates to launch jobs that run Ansible Playbooks.
- A job template associates a number of Red Hat Ansible resources: a playbook from a project; an inventory of hosts; an automation execution environment; and any credentials needed for authentication to the managed hosts in the inventory or to decrypt files protected with Ansible Vault.

## Chapter 5

# Advanced Job Configuration

### Goal

Configure advanced features of automation controller in order to more effectively and efficiently implement jobs.

### Sections

- Improving Performance with Fact Caching (and Guided Exercise)
- Creating Job Template Surveys to Set Variables for Jobs (and Guided Exercise)
- Scheduling Jobs and Configuring Notifications (and Guided Exercise)
- Advanced Job Configuration

### Lab

# Improving Performance with Fact Caching

## Objectives

- Speed up job execution by using and managing fact caching.

## Fact Caching

Ansible facts are variables automatically discovered by Ansible on a managed host. Facts contain host-specific information that can be used just like regular variables in plays, conditionals, loops, or any other statement that depends on a value.

By default, each play in a playbook automatically runs the `setup` module before its first task to gather facts from each managed host that matches the play's host pattern.

Running the `setup` module to collect facts for every play has performance consequences, especially on a large set of managed hosts. If you are not using any facts in the play, you can speed up execution by turning off automatic fact gathering by setting `gather_facts: no` in the play.

However, you might need to use facts in your play. You can configure fact caching in automation controller to store facts collected by one job so that they can be used by a later job. One playbook can collect facts for all hosts in the inventory and cache those facts so that later playbooks can use them without fact gathering or manually running the `setup` module.

Fact caching is also useful if you use the "magic" variable `hostvars` to write a play in which a task running on one host references facts that belong to another host. For example, a task running on the managed host `servera` can access the value of the `ansible_facts['default_ipv4']['address']` fact for `serverb` by referencing the `hostvars['serverb']['ansible_facts']['default_ipv4']['address']` variable. If you are not using fact caching, then referencing the `hostvars['serverb']['ansible_facts']['default_ipv4']['address']` variable only works if facts have already been gathered from `serverb` by this play or by an earlier play in the same playbook.

Retaining cached facts ensures that the play has current data but can also have some negative consequences. One possible consequence is that the fact values become outdated. For example, a fact might contain a list of the software packages installed on the system. If the fact cache is not updated after a software update, that fact might contain inaccurate data. You must therefore periodically refresh your cached facts.

## Enabling Fact Caching in Automation Controller

Automation controller includes integrated support for fact caching and a database for storing the fact cache. You need to manage time-outs for the fact cache at a global level. Fact caching control is determined by the job template.

Automation controller uses a global setting to control when facts expire. When configured, facts for each host are only considered valid for the defined number of seconds since the last time the facts were refreshed. Navigate to **Settings** and then click the **Job Settings** link. The value of **Per-Host Ansible Fact Cache Timeout** specifies how long in seconds cached Ansible facts are considered valid after they have been collected. When the facts in the cache become invalid, you need to collect them again before you can reference them. To edit the field, click **Edit** at the

## Chapter 5 | Advanced Job Configuration

bottom of the page, enter a new value for the **Per-Host Ansible Fact Cache Timeout** field, and click **Save**.

The screenshot shows the 'Details' tab of a job configuration in the Red Hat Ansible Automation Platform. The 'Per-Host Ansible Fact Cache Timeout' field is highlighted with a red box. Other visible fields include 'When can extra variables contain Jinja templates?' (template), 'Job execution path' (/tmp), 'Job Event Maximum Websocket Messages Per Second' (30), 'Default Job Timeout' (0 seconds), 'Default Inventory Update Timeout' (0 seconds), 'Default Project Update Timeout' (0 seconds), 'Run Project Updates With Higher Verbosity' (Off), 'Enable Role Download' (On), 'Maximum number of forks per job' (200), 'Follow symlinks' (Off), 'Ignore Ansible Galaxy SSL Certificate Verification' (Off), and 'Ansible Modules Allowed for Ad Hoc Jobs' (listing 'command', 'shell', and 'runcmd').

**Figure 5.1: Setting the per-host Ansible fact cache time-out**

To optimize fact caching, set `gather_facts: no` in the playbook file to disable automatic fact gathering in the play.

To use cached facts:

1. A play must gather facts and the job template that runs the associated playbook must enable fact storage.
2. If a different playbook requires cached facts, then the job template that runs the associated playbook must also enable fact storage.

One way to do this in automation controller is to set up a playbook that gathers facts as a scheduled job. That job could run a normal playbook that gathers facts, or you could set up a minimal playbook to gather facts, such as the following example:

```
- name: Refresh fact cache
 hosts: all
 gather_facts: true
```

Because the play does not include a `tasks` or a `roles` section, this play only gathers facts.



### Note

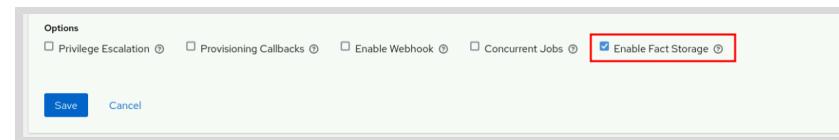
You do not need to gather facts for all your hosts at the same time. It might make sense to gather facts for smaller sets of hosts to spread the load. What matters is that you gather facts for all of your hosts before they expire or become stale.

The following procedure shows how to enable fact caching in the automation controller interface:

1. Navigate to **Resources > Templates**.
2. For the appropriate job template, click the **Edit Template** icon and then scroll to the bottom of the page and click **Edit** to edit the settings.

**Chapter 5 | Advanced Job Configuration**

3. In the **Edit Details** section, scroll to the bottom and select **Enable Fact Storage**.



**Figure 5.2: Enabling fact storage**

4. Click **SAVE** to save the modified job template configuration.

Whenever you run a new job based on a template that has the **Enable Fact Storage** option set, that job uses the fact cache. If the Ansible Playbook also has the `gather_facts` variable set to `yes` or calls the `setup` module as a task, the job also gathers facts and stores them in the fact cache.

**Important**

If a job gathers some facts but not others, the facts that have been newly gathered are updated in the fact cache.

When you launch a job, automation controller injects all Ansible facts for each of the managed hosts from the running job into a memory cache. After finishing the job, automation controller retrieves all the records for a particular host from this cache, and then saves each fact that has an update time later than the cached copy in the fact cache database.

**References****Ansible User Guide: Caching Facts**

[https://docs.ansible.com/ansible/latest/user\\_guide/playbooks\\_variables.html#caching-facts](https://docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html#caching-facts)

**Automation Controller User Guide: Fact Caching**

[https://docs.ansible.com/automation-controller/latest/html/userguide/job\\_templates.html#fact-caching](https://docs.ansible.com/automation-controller/latest/html/userguide/job_templates.html#fact-caching)

## ► Guided Exercise

# Improving Performance with Fact Caching

Use fact caching to speed up job execution and explore how to manage the fact cache.

### Outcomes

- Use fact caching in job templates.

### Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command ensures that automation controller is installed and configured with any necessary resources created in previous exercises.

```
[student@workstation ~]$ lab start job-facts
```

### Instructions

- ▶ 1. Navigate to `https://controller.lab.example.com` and log in as the `admin` user with `redhat` as the password.
- ▶ 2. Update the global job settings for automation controller to specify that cached facts are valid for one day (86,400 seconds).
  - 2.1. Navigate to **Settings** and click the **Job Settings** link.
  - 2.2. Click **Edit**.
  - 2.3. Enter `86400` in the **Per-Host Ansible Fact Cache Timeout** field and then click **Save**.
- ▶ 3. Display the cached facts for hosts in the Dev inventory. Navigate to **Resources > Hosts** and click the link for the `servera.lab.example.com` host. Click the **Facts** tab and notice that facts do not exist for the host.  
Repeat for the `serverb.lab.example.com` host.



#### Note

If you previously completed this exercise or if you perform the exercises in a different order, then facts might already be cached for the `servera.lab.example.com` and `serverb.lab.example.com` hosts.

- ▶ 4. Collect (or update) facts for hosts in the Dev inventory.
  - 4.1. Navigate to **Resources > Templates** and click the **Edit Template** icon for the **Refresh Fact Cache** job template.

- 4.2. Select **Enable Fact Storage** and then click **Save**.
- 4.3. From the **Details** page, click **Launch**. Select the Dev inventory, click **Next**, and then click **Launch**. This job targets hosts in the Dev inventory using the `refresh_fact_cache.yml` playbook in the **Fact Caching** project. The job succeeds.
- 4.4. *(Optional)* Display the cached facts for hosts in the Dev inventory. Navigate to **Resources > Hosts** and click the link for the `servera.lab.example.com` host. Click the **Facts** tab and notice that facts exist for the host.  
Repeat for the `serverb.lab.example.com` host.



### Important

If you do not see host facts, then verify the **Per-Host Ansible Fact Cache Timeout** setting under **Settings > Job Settings** on the automation controller web UI. By default, the fact cache timeout is set to zero.

- ▶ 5. The `lab` command creates the **Internal Web** project and the **Internal Web Access** job template. The playbook used by the job template disables fact gathering, but it requires networking facts to create a custom firewall rule. The job template prompts for an inventory and then runs the playbook on hosts in the selected inventory. Update the job template so that it can use cached facts and then launch the job template.
  - 5.1. Navigate to **Resources > Templates** and click the **Edit Template** icon for the **Internal Web Access** job template.
  - 5.2. Select **Enable Fact Storage** and then click **Save**.
  - 5.3. From the **Details** page, click **Launch**. Select the Dev inventory, click **Next**, and then click **Launch**. Verify that it succeeds.
- ▶ 6. In addition to configuring a basic web server, the `internal_apache_setup.yml` playbook used by the job template only allows access to hosts on the `172.25.250.0/24` network.
  - 6.1. The `workstation.lab.example.com` server contains two NICs. The `eth0` NIC is on the same network as the `servera.lab.example.com` and `serverb.lab.example.com` servers. The `virbr0` NIC is a virtual device on a separate network. Display the IP addresses assigned to the `workstation.lab.example.com` server.

```
[student@workstation ~]$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
 qlen 1000
 link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
 inet 127.0.0.1/8 scope host lo
 valid_lft forever preferred_lft forever
 inet6 ::1/128 scope host
 valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default
 qlen 1000
 link/ether 52:54:00:00:fa:09 brd ff:ff:ff:ff:ff:ff
 inet 172.25.250.9/24 brd 172.25.250.255 scope global noprefixroute eth0
 valid_lft forever preferred_lft forever
```

**Chapter 5 | Advanced Job Configuration**

```
inet6 fe80::b250:fdb:4e16:ff7c/64 scope link noprefixroute
 valid_lft forever preferred_lft forever
3: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN
 group default qlen 1000
 link/ether 52:54:00:df:3a:fc brd ff:ff:ff:ff:ff:ff
 inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0
 valid_lft forever preferred_lft forever
...output omitted...
```

- 6.2. Run the `curl` command to access `http://servera.lab.example.com` from the `eth0` interface. Because the `--interface` option for the `curl` command must be run as the `root` user, use the `sudo` command with `student` as the password.

```
[student@workstation ~]$ sudo curl --interface eth0 \
> http://servera.lab.example.com
[sudo] password for student: student
Host: servera.lab.example.com

Accessible by the 172.25.250.0/24 network.
```

- 6.3. Run the `curl` command to access `http://servera.lab.example.com` from the `virbr0` interface. Use the `-m3` option so that you do not have to wait long for the request to time out. The command fails because the firewall running on the `servera.lab.example.com` server rejects web requests coming from the `192.168.122.1/24` network.

```
[student@workstation ~]$ sudo curl --interface virbr0 -m3 \
> http://servera.lab.example.com
curl: (28) Connection timed out after 3000 milliseconds
```

## Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish job-facts
```

# Creating Job Template Surveys to Set Variables for Jobs

## Objectives

- Create a job template survey to help users more easily launch a job with custom variable settings.

## Managing Variables

You should write playbooks that you can reuse in similar situations. For example, you might write a playbook that deploys an application to a managed host. If you need to deploy that same application to a different managed host, but need to make minor changes to its configuration, perhaps by setting IP addresses unique to that host in the application's setting, it would save time if you could just run that playbook on the new host with settings specific to that run. One way to accomplish this is by using variables to specify those configuration settings.

You can set variables with specific values in many different ways, and you can override them depending on how you set the variables. For example, a role or collection can provide a default value for a variable that you might override with the value set in the inventory that you are using, or in the playbook itself.

In automation controller, you can interactively set variables in two ways:

- Automation controller can prompt you to enter extra variables when you launch a job from a job template.
- You can create a survey that presents you with a form to help you set variables when you launch a job from a job template or a workflow template.



### Important

Automation controller does not support playbooks that use `vars_prompt` to interactively set variables.

## Defining Extra Variables

In automation controller, you can use a job template to directly set extra variables, using either of the following two methods:

1. Define the variables in YAML or JSON format in the **Variables** field of the job template. This sets the specific values for extra variables used by the job template.
2. Select **Prompt on launch** for the **Variables** field of the job template. This causes automation controller to prompt you to interactively specify and modify extra variables and their values when you use the job template to launch a job.

These extra variables are exactly like the variables specified by the `-e` or `--extra-vars` options for `ansible-navigator` or `ansible-playbook`, and their values override any values set for those variables. In other words, if a variable is set with a value in some other way, and it has the same name as an extra variable, the value set for it as an extra variable is the one that is used by automation controller.

## Chapter 5 | Advanced Job Configuration

If you select **Prompt on launch** for the **Variables** field, a dialog box opens where you can edit the extra variables for the job.



Figure 5.3: Adjusting extra variables on job launch



### Important

If you relaunch a job that ran earlier (for example, by clicking the **Relaunch Job** icon for a job from the **Jobs** page), then you use the same values for its extra variables that the preceding job used. The extra variables for a job cannot be changed when you relaunch it.

Launch the job from its job template, not by relaunching a previous job, if you want to use different values for its extra variables.

## Job Template Surveys

Modifying playbooks with extra variables can be challenging because you need to understand what variables are available and how the job template uses the variables. For example, misspelling the name of an extra variable does not generate an error, but it does not override the intended variable. Even if you wrote a playbook recently, you might not remember the names of all of the variables used by the playbook by the time you launch the job template.

Rather than relying on remembering variable names, you can configure a job template or workflow template with a survey. When you launch a job that contains a survey, the web UI displays the survey to prompt you for information that automation controller uses to set values for extra variables. Surveys provide clearer guidance and control for setting extra variables.

Job template surveys offer several advantages over other ways to set extra variables. Users who launch the job do not need to have detailed knowledge about how extra variables work and do not need to know the names of the extra variables used by the playbook.



### Important

A survey sets extra variables. The values set by surveys for variables take precedence over all other ways that value might be set.

This includes values set in the job template's **Variables** field or by using its **Prompt on launch** setting.

## Managing Answers to Survey Questions

Survey questions and their descriptions can provide meaningful context to the user who launches the job template. Surveys allow you to configure how a user responds to a question. If a question can only have a defined set of answers, then you might choose one of the multiple choice answer types. For example, a survey could prompt for the name of a service to restart and then the multiple choice answers could define some system services, such as `sshd`, `httpd`, and `postfix`.

Other survey questions might expect arbitrary text or numbers and you can restrict the entered values by defining the minimum and the maximum number of characters that can be used. If desired, then you can also provide default answers and decide which questions require an answer.

The responses to a survey question must use one of the following answer types:

| Type                              | Description                                                                   |
|-----------------------------------|-------------------------------------------------------------------------------|
| Text                              | Single-line text.                                                             |
| Textarea                          | Multiline text.                                                               |
| Password                          | Data is treated as sensitive information.                                     |
| Multiple Choice (single select)   | A list of options from which only one option can be chosen as a response.     |
| Multiple Choice (multiple select) | A list of options from which one or more options can be chosen as a response. |
| Integer                           | Integer number.                                                               |
| Float                             | Floating-point decimal number.                                                |

## Creating a Job Template Survey

When you create a job template, you cannot add a survey at the same time. You can only add a survey after you create the job template.

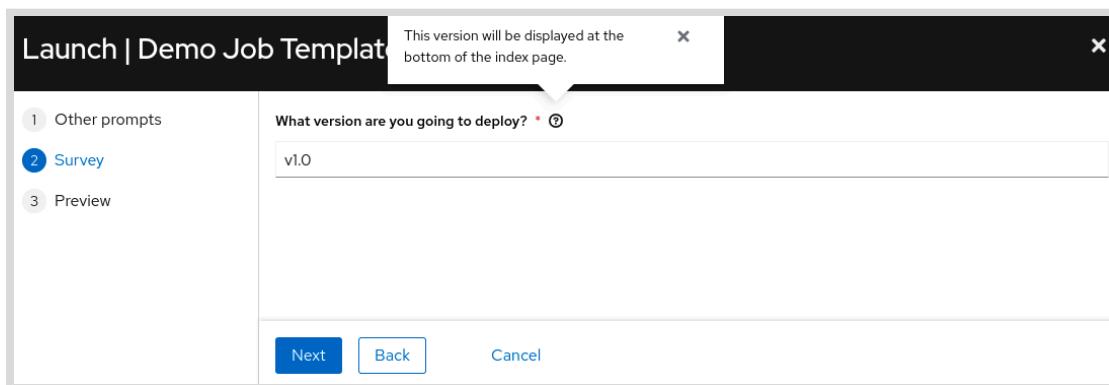
Use the following procedure to add a survey to an existing job template:

1. Navigate to **Resources > Templates** and click the desired job template.
2. Click the **Survey** tab and then click **Add** to open the **Add Question** page.
3. To add a question to the survey:
  - a. In the **Question** field, enter the question to display. You can add an optional description to provide more information about the question.
  - b. In the **Answer variable name** field, enter the name of the extra variable that stores the response. You might need to consult the playbook to identify the exact variable name.
  - c. From the **Answer type** list, select the answer type that you want to use for the response.
  - d. If you use a multiple choice answer type, then define the list by entering the first choice in the **Multiple Choice Options** field, and then press **Enter** to display a new blank row. Continue until you have entered all required choices.

**Chapter 5 | Advanced Job Configuration**

- e. If you are not using a multiple choice answer type, then you can specify the minimum and maximum character length for the response in the **Minimum length** and **Maximum length** fields, respectively.
  - f. You can also define a default value for the extra variable in the **Default answer** field. The survey starts with this value, but the user responding to the survey can choose a different value.
  - g. Select **Required** to specify that a response is mandatory. Clear this field if a response is optional.
  - h. Click **Save** to add the question to the survey. The survey now lists the new question.
4. Repeat the previous steps to add additional questions to the survey. After adding all of your questions, scroll to the top of the screen.
  5. New surveys are disabled by default. To enable the survey, click **Survey Disabled** so that its label changes to **Survey Enabled**.
  6. If you need to change the order of the survey questions, click **Edit Order** and drag and drop the questions as required.

When you launch a job template that includes a survey, automation controller presents the survey before running the job. A question mark icon appears next to a question if you added a description for the survey question. Clicking the question mark icon displays the description.



**Figure 5.4: Survey prompt**

After completing the survey, click **Next**. The **Preview** page displays variables and the values that the survey configures. If you are satisfied with your responses, then click **Launch** to start the job. Otherwise, click **Back** to change a response to a survey question.

The screenshot shows the 'Launch | Demo Job Template' interface. On the left, there's a sidebar with three items: 'Other prompts', 'Survey', and 'Preview' (which is selected). The main area displays job configuration details: Inventory (Demo Inventory), Project (Demo Project), Playbook (hello\_world.yml), Forks (0), Verbosity (0 (Normal)), Show Changes (Off), Job Slicing (1), Credential (SSH: Demo Credential), Created (1/11/2022, 5:57:22 PM by admin), and Last Modified (5/2/2022, 11:20:52 AM by admin). Below this is a 'Prompted Values' section with a 'Variables' tab (selected) showing YAML content: deployment\_version: v1.0. At the bottom are 'Launch', 'Back', and 'Cancel' buttons.

Figure 5.5: Preview page



## References

### Automation Controller User Guide - Surveys

[https://docs.ansible.com/automation-controller/latest/html/userguide/job\\_templates.html#surveys](https://docs.ansible.com/automation-controller/latest/html/userguide/job_templates.html#surveys)

## ► Guided Exercise

# Creating Job Template Surveys to Set Variables for Jobs

Add a survey to an existing job template and launch a job using that survey.

### Outcomes

- Add a survey to an existing job template.
- Launch a job using a survey from the automation controller web UI.

### Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that automation controller is installed and configured with any necessary resources created in previous exercises.

```
[student@workstation ~]$ lab start job-survey
```

### Instructions

- ▶ 1. Go to `https://controller.lab.example.com` and log in as the `admin` user with `redhat` as the password.
- ▶ 2. Add a survey to the `DEV webservers setup` job template.
  - 2.1. Go to **Resources > Templates** and then click the **DEV webservers setup** link.
  - 2.2. Click the **Survey** tab and then click **Add**.
  - 2.3. Add a single question to the survey, using the following information. When finished, click **Save**.

| Field                | Value                                                             |
|----------------------|-------------------------------------------------------------------|
| Question             | What version are you deploying?                                   |
| Description          | This version number is displayed at the bottom of the index page. |
| Answer variable name | deployment_version                                                |
| Answer type          | Text                                                              |
| Required             | (selected)                                                        |
| Minimum length       | 1                                                                 |
| Maximum length       | 40                                                                |
| Default Answer       | v1.0                                                              |

- 2.4. Click **Survey Disabled** and verify that the label changes to **Survey Enabled**. The survey is now enabled.



### Important

The survey is disabled by default. Make sure that you enable it.

- 3. Clone the `https://git.lab.example.com/git/my_webservers_DEV.git` Git repository into the `/home/student/git-repos` directory.

- 3.1. From a terminal, create the `/home/student/git-repos` directory if it does not already exist, and then change into it:

```
[student@workstation ~]$ mkdir -p ~/git-repos/
[student@workstation ~]$ cd ~/git-repos/
```

- 3.2. Clone the `https://git.lab.example.com/git/my_webservers_DEV.git` Git repository and then change to the cloned repository:

```
[student@workstation git-repos]$ git clone \
> https://git.lab.example.com/git/my_webservers_DEV.git
Cloning into 'my_webservers_DEV'...
...output omitted...
[student@workstation git-repos]$ cd my_webservers_DEV
```

- 4. Modify the `~/git-repos/my_webservers_DEV/templates/index.html.j2` Jinja2 template to use the variable from the survey.

- 4.1. Edit the `index.html.j2` template to add the following line to the bottom of the file:

```
Deployment Version: {{ deployment_version }}

```

After you edit and save `index.html.j2`, the file contains the following content:

**Chapter 5 | Advanced Job Configuration**

```
Current Host: {{ ansible_facts['fqdn'] }}

Server list:

{% for host in groups['all'] %}
{{ host }}

{% endfor %}
Deployment Version: {{ deployment_version }}

```

- 4.2. Add, commit, and push the file to the remote Git repository:

```
[student@workstation my_webservers_DEV]$ git add templates/index.html.j2
[student@workstation my_webservers_DEV]$ git commit \
> -m "Display Deployment Version on index page"
[main 2c435c7] Display Deployment Version on index page
 1 file changed, 1 insertion(+)
[student@workstation my_webservers_DEV]$ git push -u origin
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 468 bytes | 468.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0
To https://git.lab.example.com/git/my_webservers_DEV.git
 3b3341b..2c435c7 main -> main
```

- 5. Update the local copy of the Git repository on automation controller for the My Webservers DEV project.
- 5.1. Go to **Resources > Projects**.
  - 5.2. On the same line as the My Webservers DEV project, click the **Sync Project** icon and wait for the **Status** column to display **Successful**.
  - 5.3. Click the **Refresh project revision** icon in the **Revision** column to update the revision version. The revision matches the last commit. You can verify this information with the following command:

```
[student@workstation my_webservers_DEV]$ git log --pretty=format:'%h' -1
2c435c7
```

- 6. As a member of the Developers team, launch a job using the updated DEV webservers setup job template.
- 6.1. Go to **admin > Logout** in the upper right corner to log out, and then log back in as **daniel** using **redhat123** as the password.
  - 6.2. Go to **Resources > Templates** and then click the **Launch Template** icon for the **DEV webservers setup** job template. This opens the survey you just created and prompts for your input.
  - 6.3. Leave **v1.0** in the text field and click **Next** followed by **Launch** to launch the job. This redirects you to a detailed status page of the running job.
  - 6.4. Briefly observe the live output of the running job.

**Chapter 5 | Advanced Job Configuration**

- 6.5. When the job is complete, confirm that the **Status** of the job on the **Details** tab is **Successful**.
- 7. Verify that the web servers have been updated on `servera.lab.example.com` and `serverb.lab.example.com`.
- 7.1. Open a web browser and go to `http://servera.lab.example.com` and `http://serverb.lab.example.com` in separate tabs. You should see this line at the bottom of both pages:

```
...output omitted...
Deployment Version: v1.0
```

**Important**

If you complete the exercises in this course out of order, or if you repeat this exercise after completing later exercises, then your web browser might redirect requests to the HTTPS versions of `http://servera.lab.example.com` and `http://serverb.lab.example.com`. The redirection results in **Unable to connect** messages because the `DEV webservers setup` job template does not configure HTTPS.

If this happens, then you can either clear your web browser cache or you can use the `curl` command to verify the content of the web servers.

- 8. Go to **daniel > Logout** in the upper right corner to log out of the automation controller web UI.

## Finish

On the **workstation** machine, change to the **student** user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish job-survey
```

# Scheduling Jobs and Configuring Notifications

## Objectives

- Schedule automatic job execution and configure job notifications.

## Scheduling Job Execution

Sometimes you might need to launch a job template automatically at a particular time, or on a particular schedule. You can use automation controller to configure schedules that launch job templates on a customizable schedule.

If you have the **Execute** role on a job template, then you can schedule the job template to run at a specific date, time, and frequency. To create a new job template schedule, navigate to **Resources** > **Templates** and click the job template that you want to schedule. Click the **Schedules** tab and then click **Add** to create a new schedule for that job template.

Enter the required details:

### Name

The name of the schedule.

### Start date/time

The date and time that the job schedule starts.

### Local time zone

You can use the `timedatectl` command to determine your local time zone in this format.

### Run frequency

How often to repeat the associated job. You can choose **None (run once)**, **Minute**, **Hour**, **Day**, **Week**, **Month**, or **Year**.

Depending on the chosen frequency, you might need to provide additional information (for example, to launch a job every two days, or on the first Sunday of every month).

When finished, click **Save** to save the schedule. You can activate or deactivate a schedule by setting the **On/Off** switch at the top of the **Details** page for the schedule.

## Chapter 5 | Advanced Job Configuration

The screenshot shows the 'Create New Schedule' dialog box. At the top, there are tabs for 'Templates > Demo Job Template > Schedules'. The main area has fields for 'Name \*' (Demo Job Schedule), 'Description' (empty), 'Start date/time \*' (2022-04-21, 11:30 AM), 'Local time zone \*' (UTC), and 'Run frequency \*' (Day). Below these, a 'Frequency Details' section allows setting 'Run every \*' (2 days) and 'End \*' (Never, After number of occurrences, or On date). At the bottom are 'Save' and 'Cancel' buttons.

Figure 5.6: Scheduling job execution

## Temporarily Disabling a Schedule

Navigate to Views > Schedules in the automation controller web UI to open the **Schedules** page. This page lists all job schedules. Set the **On/Off** switch under the **Actions** column of the desired schedule name to activate or deactivate the schedule.

You can also edit or delete any schedule, assuming you have sufficient privileges to do so, from this page.

| Schedules                                     |                |                       |                                         |
|-----------------------------------------------|----------------|-----------------------|-----------------------------------------|
| <input type="checkbox"/> Name                 | Type           | Next Run              | Actions                                 |
| <input type="checkbox"/> Demo Job Schedule    | Playbook Run   |                       | <input checked="" type="checkbox"/> Off |
| <input type="checkbox"/> Cleanup Job Schedule | Management Job | 4/24/2022, 7:40:26 PM | <input type="checkbox"/> On             |

Figure 5.7: Deactivated and activated schedules

## Scheduled Management Jobs

Automation controller ships with four scheduled management jobs by default. These schedules run built-in management jobs to perform periodic maintenance on the automation controller.

| Management job                 | Description                                                                                              |
|--------------------------------|----------------------------------------------------------------------------------------------------------|
| Cleanup Job Schedule           | Removes information about jobs that are more than 120 days old. It runs weekly on Sundays.               |
| Cleanup Activity Schedule      | Removes information from the activity stream that is more than 355 days old. It runs weekly on Tuesdays. |
| Cleanup Expired OAuth 2 Tokens | Deletes the expired OAuth 2 access and refresh tokens. It runs weekly.                                   |
| Cleanup Expired Sessions       | Removes expired browser sessions. It runs weekly.                                                        |

You can change the frequency of these management jobs by editing their schedules.

## Reporting Job Execution Results

Red Hat Ansible Automation Platform provides centralized logging and auditing. When you execute a job, details about the job execution are logged in the automation controller database. You can use this database to review job execution history.

Historical job execution details are helpful for confirming the success or failure of scheduled and delegated job executions. The ability to retrieve historical job execution details is helpful, but for jobs related to critical functions, you might prefer immediate notification of when a job starts, as well as the success or failure of its execution.

Automation controller can send notifications when a job starts, as well as the job execution results. You need to create notification templates that specify how to send notifications to take advantage of this feature. Automation controller supports many mechanisms for sending notifications. Some are based on open protocols, such as email and IRC, and others are based on proprietary solutions, such as Slack and Twilio.

## Notification Templates

Notification templates help you to send notifications on job start, job success, and job failure, even in events related to approvals, or any combination thereof.

Notification templates specify how to send notifications. The following notification types are supported:

- Email
- Grafana
- IRC
- Mattermost
- PagerDuty
- Rocket.Chat
- Slack
- Twilio
- Webhook

## Creating Notification Templates

Use the following procedure to create a notification template:

**Chapter 5 |** Advanced Job Configuration

1. Navigate to **Administration > Notifications** in the automation controller web UI.
2. Click **Add** to create a notification template, and enter the required details:

**Name**

The name of the notification template.

**Description**

Enter a description for the notification template. (Optional)

**Organization**

The organization within which to create the notification template.

**Type**

The mechanism to be used by the notification template for generating notifications.

3. Enter the required details for the notification template type in the **Type Details** section.

For example, for the **E-mail** notification template type, the following fields are displayed in the **Type details** section:

**Username**

Specifies the username when using SMTP authentication. (Optional)

**Password**

Specifies the password for the username when using SMTP authentication. (Optional)

**Host**

Specifies which SMTP server to use.

**Recipient list**

Specifies the email addresses of the recipients for the notification email, one per line.

**Sender e-mail**

Specifies the email address of the sender when composing the notification.

**Port**

Specifies the port to connect to on the SMTP server.

**Timeout**

Specifies the maximum time in seconds (from 1 to 120) that the automation controller attempts connecting to the SMTP server.

**E-mail options**

You can enable SSL or TLS to secure email communications by selecting the respective checkbox. (Optional)

4. Set the **Customize messages** switch to on if you want to customize the content of the notification messages.

For each of the notification types, preconfigured messages are available that use Jinja2 expressions to provide information about the job. The preconfigured messages use variables related to the `job` variable (such as `job_friendly_name` for the friendly name of the job, `job_metadata` for the job metadata, and so on), and some of its attributes (such as `name` for the name of the job, `status` for the status of the job, and so on).

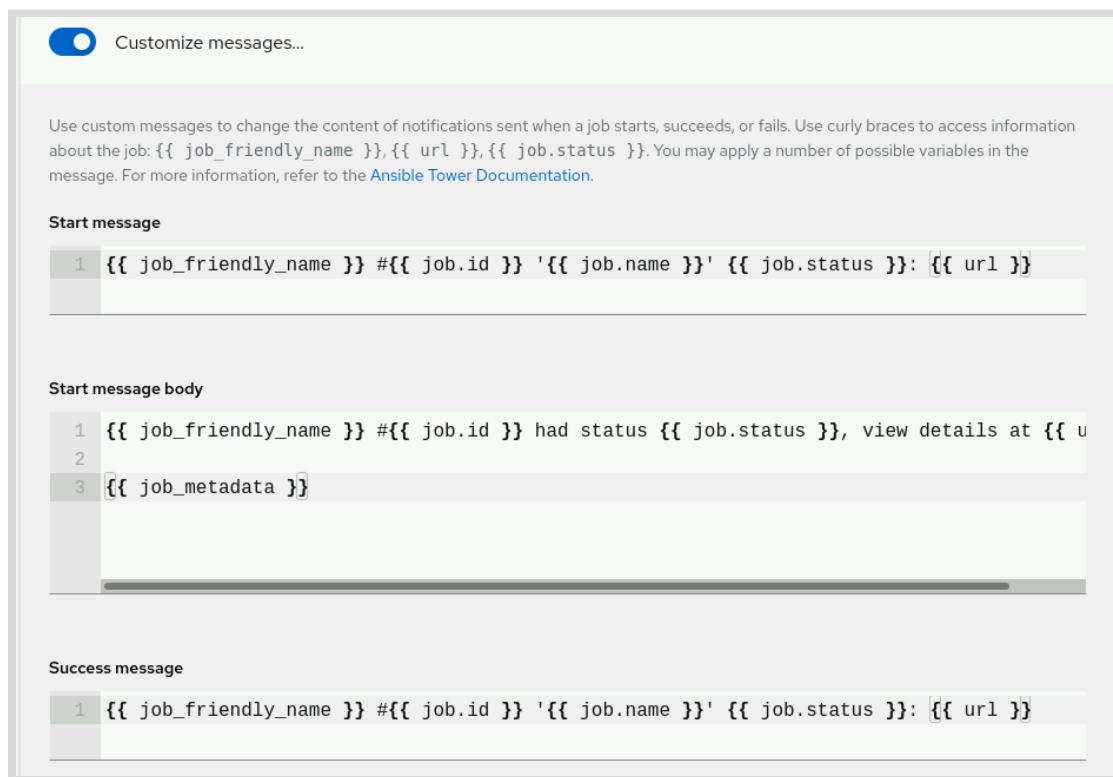
For example, for the **E-mail** notification template type, the preconfigured start message body uses the friendly name of the job, the `id` and `status` job attributes, as well as the URL and all metadata information for the job:

**Chapter 5 | Advanced Job Configuration**

```
 {{ job_friendly_name }} #{{ job.id }} had status {{ job.status }}, view details at
 {{ url }}

 {{ job_metadata }}
```

You can create custom notifications by modifying the preconfigured messages. A link to a complete list of supported attributes and variables is available in the references at the end of this section.



**Figure 5.8: Custom messages for Email type**

5. Click **Save** to finalize the creation of the notification template.
6. You can send a test notification for each notification template after creating or editing them. Click the **Test notification** icon in the **Actions** column of the desired notification template. The **Status** column should display **Successful**.

The screenshot shows a table with columns: Name, Status, Type, and Actions. A tooltip 'Test notification' points to the Actions column for the first row. The table has one item: 'Notification Template for Testing' with a status of 'Successful' and an 'Email' type.

Figure 5.9: Testing notification templates

## Enabling Job Result Notification

When you create a notification template, it becomes available for use by the following automation controller resources:

- Job templates
- Projects
- Inventory sources
- Workflow templates
- Organizations

For job templates, projects, and inventory source, the available notifications are **Start**, **Success** and **Failure**. For workflow templates and organizations, you can also get **Approval** notifications for events with approvals. This is discussed later in the course.

The screenshot shows the 'Notifications' tab for a job template. It lists a single notification named 'Notification Template for Testing' with an 'Email' type. The 'Success' notification is enabled, indicated by a blue toggle switch. The 'Failure' notification is also listed but appears to be disabled.

Figure 5.10: Enabling notifications on a job template

Use the following procedure to enable notifications for a job template:

1. Navigate to **Resources > Templates** in the automation controller web UI.
2. Click the name of the required job template and then click the **Notifications** tab to display the list of notification templates.

3. Each listed notification template has switches for controlling **Start**, **Success**, and **Failure** notifications. Set the corresponding switches to configure the desired notifications for the job template.



### Warning

Under some circumstances, you might encounter an issue in which automation controller fails to send a notification. This is a known issue at the time of writing, caused by a race condition related to <https://github.com/ansible/awx/issues/11422>.

One symptom, besides missing notification messages, is content similar to the following in the `/var/log/tower/task_system.log` file:

```
2022-02-17 06:54:54,645 WARNING [b8c34f2e75d2430c8874f3f8feb557cc]
awx.main.tasks Failed to even try to send notifications for job '2022-02-17
06:37:20.281267+00:00-56302-running' due to job not being in finished state.
```

Automation controller in Red Hat Ansible Automation Platform 2.1.2 and later includes a setting that you can adjust to work around the issue.

By default, automation controller uses the setting `AWX_NOTIFICATION_JOB_FINISH_MAX_RETRY = 5`. If automation controller fails to send a notification, then you can change this variable to have a higher value, such as 20. Modify the `/var/lib/awx/venv/awx/lib/python3.9/site-packages/awx/settings/defaults.py` file on the automation controller. After making this change, run the `automation-controller-service restart` command.



### References

#### **Ansible Documentation: Notifications**

<https://docs.ansible.com/automation-controller/latest/html/userguide/notifications.html#>

#### **Ansible Documentation: Supported Attributes for Custom Notifications**

[https://docs.ansible.com/automation-controller/latest/html/userguide/notification\\_parameters\\_supported.html#ir-notifications-reference](https://docs.ansible.com/automation-controller/latest/html/userguide/notification_parameters_supported.html#ir-notifications-reference)

## ► Guided Exercise

# Scheduling Jobs and Configuring Notifications

Create an email notification template, configure a job template to use the notification template, and launch a job to confirm that the notification works.

### Outcomes

- Create a notification template.
- Use the notification template with a job template to generate email notifications of job results.

### Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that automation controller is installed and configured with any necessary resources created in previous exercises.

```
[student@workstation ~]$ lab start job-notification
```

### Instructions

- ▶ 1. Navigate to `https://controller.lab.example.com` and log in as the `admin` user with `redhat` as the password.
- ▶ 2. Create a new notification template.
  - 2.1. Navigate to **Administration > Notifications** and then click **Add**.
  - 2.2. Create the new notification template using the following information. When finished, click **Save**.

| Field          | Value                                          |
|----------------|------------------------------------------------|
| Name           | Email AAP Admins                               |
| Description    | Sends an email to notify the status of the job |
| Organization   | Default                                        |
| Type           | E-mail                                         |
| Host           | localhost                                      |
| Recipient list | aap-admins@lab.example.com                     |
| Sender e-mail  | system@controller.lab.example.com              |
| Port           | 25                                             |
| Timeout        | 30                                             |

- 3. Validate the new notification template.
- 3.1. Navigate to **Administration > Notifications**.
  - 3.2. Click the **Test notification** icon for the **Email AAP Admins** job template to test the notification process.
  - 3.3. After several seconds, the **Status** column displays the **Successful** message.
- 4. Configure the **DEV webservers setup** job template to use the new **Email AAP Admins** notification template. Trigger the notification when jobs using the **DEV webservers setup** job template either succeed or fail.
- 4.1. Navigate to **Resources > Templates** and then click the **DEV webservers setup** link.
  - 4.2. Click the **Notifications** tab and then set the **Success** and **Failure** switches to on for the **Email AAP Admins** notification.
- 5. Verify that the **DEV webservers setup** job template triggers a notification email after job completion.
- 5.1. Open a terminal and connect to the **utility** server.

```
[student@workstation ~]$ ssh student@utility
```

- 5.2. Use the **tail** command to read incoming messages to the local mailbox file of the **student** user. You should see the email that was sent by the test notification launched from a previous step. Your message looks similar to the following:

```
[student@utility ~]$ tail -f /var/mail/student
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit
Subject: Notification Test 1 https://controller.lab.example.com
From: system@controller.lab.example.com
```

**Chapter 5 | Advanced Job Configuration**

```
To: aap-admins@lab.example.com
Date: Fri, 22 Apr 2022 19:39:31 -0000
Message-ID: <165065637197.1384.7900890983573152025@controller.lab.example.com>

Test Notification 1 https://controller.lab.example.com
```

**Note**

If you prefer, then use the `mailx` or `mutt` commands to display mail messages on the utility server.

- 5.3. From the automation controller web UI, navigate to **Resources > Templates** and then click the **Launch Template** icon for the **DEV webservers setup** job template.
- 5.4. In the survey window, enter **v1.2** for the deployment version, click **Next**, and then click **Launch**.
- 5.5. Go to the terminal window running the `tail` command and wait. After about one minute, you should see a notification email arrive that looks similar to the following example. When finished, press **Ctrl+C** to exit the `tail` command.

```
From system@controller.lab.example.com Fri Apr 22 15:44:03 2022
Return-Path: <system@controller.lab.example.com>
X-Original-To: aap-admins@lab.example.com
Delivered-To: aap-admins@lab.example.com
Received: from controller.lab.example.com (controller.lab.example.com
[172.25.250.7])
 by utility.lab.example.com (Postfix) with ESMTPS id EDB3418BB563
 for <aap-admins@lab.example.com>; Fri, 22 Apr 2022 15:44:03 -0400 (EDT)
Received: from controller.lab.example.com (localhost [IPv6:::1])
 by controller.lab.example.com (Postfix) with ESMTP id CF64D195822E
 for <aap-admins@lab.example.com>; Fri, 22 Apr 2022 15:44:03 -0400 (EDT)
Content-Type: text/plain; charset="utf-8"
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit
Subject: Job #10 'DEV webservers setup' successful:
https://controller.lab.example.com/#/jobs/playbook/10
From: system@controller.lab.example.com
To: aap-admins@lab.example.com
Date: Fri, 22 Apr 2022 19:44:03 -0000
Message-ID:
<165065664382.1385.10783097240459901012@controller.lab.example.com>

Job #10 had status successful, view details at https://controller.lab.example.com/
#/jobs/playbook/10

{>
 "id": 10,
 "name": "DEV webservers setup",
 "url": "https://controller.lab.example.com/#/jobs/playbook/10",
 "created_by": "admin",
 "started": "2022-04-22T19:43:40.383371+00:00",
 "finished": "2022-04-22T19:44:02.979429+00:00",
 "status": "successful",
```

**Chapter 5 | Advanced Job Configuration**

```
"traceback": "",
"inventory": "Dev",
"project": "My Webservers DEV",
"playbook": "apache-setup.yml",
"credential": "Developers",
"limit": "",
"extra_vars": "{\"deployment_version\": \"v1.2\"}",
"hosts": {
 "serverb.lab.example.com": {
 "failed": false,
 "changed": 6,
 "dark": 0,
 "failures": 0,
 "ok": 7,
 "processed": 1,
 "skipped": 0,
 "rescued": 0,
 "ignored": 0
 },
 "servera.lab.example.com": {
 "failed": false,
 "changed": 6,
 "dark": 0,
 "failures": 0,
 "ok": 7,
 "processed": 1,
 "skipped": 0,
 "rescued": 0,
 "ignored": 0
 }
}
}^C
```

- 5.6. Exit the terminal session on the **utility** system.

```
[student@utility ~]$ exit
```

- ▶ 6. Verify that the web servers have been updated successfully on **servera.lab.example.com** and **serverb.lab.example.com**.
- 6.1. Open a web browser and navigate to **http://servera.lab.example.com** and **http://serverb.lab.example.com** in separate tabs. You should see this line at the bottom of each page:
- ```
Deployment Version: v1.2
```
- ▶ 7. Based on the **DEV webservers** setup job template, schedule a new job at three minutes from the current time.
- 7.1. From a terminal window, display the current date, time, and time zone used by the **controller** server. Your environment might display a different date and time than this example. Use the identified time zone when you create your new schedule.

Chapter 5 | Advanced Job Configuration

```
[student@workstation ~]$ ssh controller timedatectl
    Local time: Fri 2022-04-22 16:14:45 EDT
    Universal time: Fri 2022-04-22 20:14:45 UTC
        RTC time: Fri 2022-04-22 20:14:44
      Time zone: America/New_York (EDT, -0400)
System clock synchronized: yes
          NTP service: active
    RTC in local TZ: no
```

- 7.2. Navigate to **Resources > Templates** and then click the **DEV webservers** setup link.
- 7.3. Click the **Schedules** tab and then click **Add**.
- 7.4. Schedule the job using the following information. The date defaults to the current date. If necessary, adjust the time zone to match the one used by the **controller** server. When finished, click **Save** to create the new schedule.

Field	Value
Name	Automatic job run
Start date/time	(Set the execution of the job to +3 minutes from the current time.)
Local time zone	(The time zone that you just determined for the automation controller, America/New_York in the preceding example.)
Run frequency	None (run once)

- 7.5. Navigate to **Views > Jobs** and wait for the scheduled job to be executed. The scheduled job succeeds.

Finish

On the **workstation** machine, change to the **student** user home directory and use the **lab** command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish job-notification
```

▶ Lab

Advanced Job Configuration

Enable fact caching and add a job template survey to an existing job template.

Outcomes

- Create a job template that uses cached facts.
- Create an associated survey.
- Launch the job template from the automation controller web UI.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command ensures that automation controller is installed and configured with any necessary resources created in previous exercises. The command also reverts the global settings for automation controller to the default value for cached facts.

```
[student@workstation ~]$ lab start job-review
```

Instructions

A developer on your team committed playbook changes to a project repository. The playbook changes open firewall access to TCP port 80 to any network assigned to the `additional_networks` variable. You have been asked to add a survey to an existing job template that sets this variable and allows access to some of your internal networks.

After testing that the playbook changes and survey work for hosts in the `Dev` inventory, you must apply the changes to hosts in the `Test` inventory.



Note

The `lab` command changes the `internal_apache_setup.yml` playbook in the `https://git.lab.example.com/git/internal_web.git` Git repository. If you are interested in reviewing the changes, then you can clone the repository and run the `git show -1` command to display details about the most recent commit.

1. Go to `https://controller.lab.example.com` and log in as the `admin` user with `redhat` as the password.
2. Update the global job settings for automation controller to specify that cached facts are valid for one day (86,400 seconds).
3. Add a new survey to the `Internal Web Access` job template. Add the survey using the following information and enable the survey.

New Survey

Field	Value
Question	Open additional networks?
Description	Open web access to selected networks.
Answer variable name	additional_networks
Answer Type	Multiple Choice (multiple select)
Required	(not selected)
Multiple Choice Options	192.168.0.0/24 (not selected as default) 192.168.100.0/24 (not selected as default) 172.25.252.0/24 (not selected as default)

4. Launch the Refresh Fact Cache job template to gather and cache facts about hosts in the Dev inventory.
5. Launch the Internal Web Access job template to verify that the survey and the most recent Git repository changes work successfully. Target hosts in the Dev inventory and when prompted by the survey, select the 172.25.252.0/24 network.
6. Launch the Refresh Fact Cache job template to gather and cache facts about hosts in the Test inventory.
7. Update the survey for the Internal Web Access job template to open TCP port 80 to the 172.25.252.0/24 network by default.
8. Launch the Internal Web Access job template, target hosts in the Test inventory, and accept the default answer of using the 172.25.252.0/24 network.

Evaluation

On the **workstation** machine, change to the **student** user home directory and use the **lab** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade job-review
```

Finish

On the **workstation** machine, change to the **student** user home directory and use the **lab** command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish job-review
```

► Solution

Advanced Job Configuration

Enable fact caching and add a job template survey to an existing job template.

Outcomes

- Create a job template that uses cached facts.
- Create an associated survey.
- Launch the job template from the automation controller web UI.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command ensures that automation controller is installed and configured with any necessary resources created in previous exercises. The command also reverts the global settings for automation controller to the default value for cached facts.

```
[student@workstation ~]$ lab start job-review
```

Instructions

A developer on your team committed playbook changes to a project repository. The playbook changes open firewall access to TCP port 80 to any network assigned to the `additional_networks` variable. You have been asked to add a survey to an existing job template that sets this variable and allows access to some of your internal networks.

After testing that the playbook changes and survey work for hosts in the Dev inventory, you must apply the changes to hosts in the Test inventory.



Note

The `lab` command changes the `internal_apache_setup.yml` playbook in the `https://git.lab.example.com/git/internal_web.git` Git repository. If you are interested in reviewing the changes, then you can clone the repository and run the `git show -1` command to display details about the most recent commit.

1. Go to `https://controller.lab.example.com` and log in as the `admin` user with `redhat` as the password.
2. Update the global job settings for automation controller to specify that cached facts are valid for one day (86,400 seconds).
 - 2.1. In the navigation pane, go to **Settings > Job Settings** link.
 - 2.2. Click **Edit**.

Chapter 5 | Advanced Job Configuration

- 2.3. Enter 86400 in the **Per-Host Ansible Fact Cache Timeout** field and then click **Save**.
- 3.** Add a new survey to the **Internal Web Access** job template. Add the survey using the following information and enable the survey.

New Survey

Field	Value
Question	Open additional networks?
Description	Open web access to selected networks.
Answer variable name	additional_networks
Answer Type	Multiple Choice (multiple select)
Required	(not selected)
Multiple Choice Options	192.168.0.0/24 (not selected as default) 192.168.100.0/24 (not selected as default) 172.25.252.0/24 (not selected as default)

- 3.1. Go to **Resources > Templates** and click the link for the **Internal Web Access** job template.
- 3.2. Click the **Survey** tab and then click **Add**.
- 3.3. Create the survey using the information in the **New Survey** table. When finished, click **Save**.
- 3.4. Click **Survey Disabled** and ensure that the label changes to **Survey Enabled**.
- 4.** Launch the **Refresh Fact Cache** job template to gather and cache facts about hosts in the **Dev** inventory.
- 4.1. Go to **Resources > Templates** and click the **Launch Template** icon for the **Refresh Fact Cache** job template. Select the **Dev** inventory, click **Next**, and then click **Launch**.
- 4.2. Wait until the job completes and then click the **Details** tab for the job. Verify that the job displays the **Successful** status.
- 4.3. (Optional). Go to **Resources > Hosts** and click the link for the **servera.lab.example.com** host. Click the **Facts** tab to view cached facts for the host.
Repeat this step for the **serverb.lab.example.com** host.

**Important**

If you do not see host facts, then verify the **Per-Host Ansible Fact Cache Timeout** setting under **Settings > Job Settings** on the automation controller web UI. By default, the fact cache timeout is set to zero.

- 5.** Launch the **Internal Web Access** job template to verify that the survey and the most recent Git repository changes work successfully. Target hosts in the **Dev** inventory and when prompted by the survey, select the **172.25.252.0/24** network.

Chapter 5 | Advanced Job Configuration

- 5.1. Go to **Resources > Templates** and click the **Launch Template** icon for the **Internal Web Access** job template.
- 5.2. Select the **Dev** inventory and then click **Next**.
- 5.3. In the survey window, select the **172.25.252.0/24** network from the menu and then click **Next**.
- 5.4. Click **Launch**. The job succeeds and the **Open ports for local network** task displays the following output:

```
TASK [Open ports for local network] *****
changed: [servera.lab.example.com] => (item=[{'port': 80, 'protocol': 'tcp'}, '172.25.250.0/24'])
changed: [serverb.lab.example.com] => (item=[{'port': 80, 'protocol': 'tcp'}, '172.25.250.0/24'])
changed: [servera.lab.example.com] => (item=[{'port': 80, 'protocol': 'tcp'}, '172.25.252.0/24'])
changed: [serverb.lab.example.com] => (item=[{'port': 80, 'protocol': 'tcp'}, '172.25.252.0/24'])
```

In addition to the **172.25.250.0/24** network (the network used by the **eth0** interface on both hosts), the playbook opened TCP port 80 to the network specified by the survey.

6. Launch the **Refresh Fact Cache** job template to gather and cache facts about hosts in the **Test** inventory.
 - 6.1. Go to **Resources > Templates** and click the **Launch Template** icon for the **Refresh Fact Cache** job template. Select the **Test** inventory, click **Next**, and then click **Launch**.
 - 6.2. Wait until the job completes and then click the **Details** tab for the job. Verify that the job displays the **Successful** status.
 - 6.3. **(Optional)**. Go to **Resources > Hosts** and click the link for the **serverc.lab.example.com** host. Click the **Facts** tab to view cached facts for the host. Repeat this step for the **serverd.lab.example.com** host.
7. Update the survey for the **Internal Web Access** job template to open TCP port 80 to the **172.25.252.0/24** network by default.
 - 7.1. Go to **Resources > Templates** and click the link for the **Internal Web Access** job template.
 - 7.2. Click the **Survey** tab and then click the **Edit Survey** icon for the **Open additional networks?** survey.
 - 7.3. Click the checkmark icon for the **172.25.252.0/24** network to make it a default answer for the survey. Click **Save**.
8. Launch the **Internal Web Access** job template, target hosts in the **Test** inventory, and accept the default answer of using the **172.25.252.0/24** network.
 - 8.1. Go to **Resources > Templates** and click the **Launch Template** icon for the **Internal Web Access** job template.

Chapter 5 | Advanced Job Configuration

- 8.2. Select the Test inventory and then click **Next**.
- 8.3. On the survey page, click **Next** to accept 172.25.252.0/24 as the additional network to open. Click **Launch**.
- 8.4. Observe the live output of the running job. The job succeeds and the **Open ports for local network** task displays the following output:

```
TASK [Open ports for local network] *****
changed: [serverd.lab.example.com] => (item=[{'port': 80, 'protocol': 'tcp'}, '172.25.250.0/24'])
changed: [serverc.lab.example.com] => (item=[{'port': 80, 'protocol': 'tcp'}, '172.25.250.0/24'])
changed: [serverd.lab.example.com] => (item=[{'port': 80, 'protocol': 'tcp'}, '172.25.252.0/24'])
changed: [serverc.lab.example.com] => (item=[{'port': 80, 'protocol': 'tcp'}, '172.25.252.0/24'])
```

In addition to the 172.25.250.0/24 network (the network used by the `eth0` interface on both hosts), the playbook opened TCP port 80 access to the network specified by the survey.

The web servers in the **Test** environment now match the web servers in the **Dev** environment.

Evaluation

On the **workstation** machine, change to the **student** user home directory and use the **lab** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade job-review
```

Finish

On the **workstation** machine, change to the **student** user home directory and use the **lab** command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish job-review
```

Summary

- Enabling fact caching can speed up job execution but requires management of fact gathering.
- You can create job template surveys to help users launch jobs with custom variable settings.
- You can configure job templates to launch jobs on a one-time or recurring schedule.
- You can configure job templates to send notifications when jobs start, succeed, or fail.

Chapter 6

Constructing Job Workflows

Goal

Use advanced features of job templates to improve performance, simplify the customization of jobs, launch multiple jobs, and provide notification of job results.

Sections

- Creating Workflow Job Templates and Launching Workflow Jobs (and Guided Exercise)
- Requiring Approvals in Workflow Jobs (and Guided Exercise)

Lab

- Constructing Job Workflows

Creating Workflow Job Templates and Launching Workflow Jobs

Objectives

- Create a workflow job template to launch a branching chain of jobs based on the success or failure of each job in the workflow sequence.

Workflow Job Templates

Job templates run single Ansible Playbooks as jobs. As the use of Red Hat Ansible grows in an organization, so does the number of Ansible Playbooks it has. Each playbook typically performs a set of tasks associated with a certain function.

Instead of writing one large playbook to automate a complex operation, you might want to run several playbooks in sequence. For example, you might have the goal of provisioning a virtual machine or cloud instance. This provisioning process might first require using a playbook from the networking team to allocate an IP address for the server and set up a DNS record. You might then use a separate playbook from the operations team to install and configure the server operating system. Finally, you might use a playbook from the development team to deploy an application. In other words, there might be a particular workflow that you need to follow for the process to succeed.

You could manage this in automation controller by manually launching multiple jobs in sequence. But, you have to execute the jobs in the correct order, as defined by your workflow, for everything to work correctly. For example, you might have the following workflow:

1. The networking jobs must run first.
2. The operations jobs only run if the networking jobs completed successfully.
3. The application development jobs only run if the networking and operations jobs completed successfully.

Finally, if one of these playbooks fails, you might need to run other playbooks to recover from the failure or to release resources that were used. For example, you might need to unassign an IP address and remove a DNS record that the workflow set up earlier in the process.

Automation controller supports *workflow job templates* to make it easier to automate complex operations. A workflow job template connects multiple job templates into a single workflow. When launched, a workflow job template launches a job using the first job template. Depending on whether that job succeeds or fails, the workflow job template determines which job template to launch next. This allows you to launch a sequence of jobs, and to automatically take recovery steps if a job fails.

You can start workflow job templates in many ways:

- Manually, from the automation controller web UI
- As a scheduled job
- By an external program using the automation controller API

Workflow job templates do not just run job templates serially. Using the graphical workflow editor, workflow job templates chain together multiple job templates and run different job templates depending on whether the previous one succeeded or failed.

Creating Workflow Job Templates

You need to create a workflow job template before defining it and associating a workflow with it. You can create a workflow template with or without an organization. To create a workflow job template within the context of an organization, you must have either the Admin or the Workflow Admin role for that organization. Recall that the System Administrator user type is a super user with full permissions to perform any action. Users of this type can create a workflow job template that is not part of an organization.

You create a workflow job template much like you create a job template:

1. Navigate to Resources > Templates and then click Add > Add workflow template.
2. At a minimum, enter a name for the workflow template in the Name field.
3. Click Save.

The screenshot shows the 'Create New Workflow Template' dialog box. At the top left is a 'Templates' link and a back arrow icon. The title 'Create New Workflow Template' is centered at the top. Below the title are several input fields and options:

- Name ***: A text input field containing 'My workflow'.
- Description**: An empty text input field.
- Organization**: A dropdown menu with a magnifying glass icon.
- Inventory**: A dropdown menu with a magnifying glass icon.
- Prompt on launch**: A checkbox next to the Inventory dropdown.
- Limit**: A dropdown menu with a magnifying glass icon.
- Prompt on launch**: A checkbox next to the Limit dropdown.
- Source control branch**: A dropdown menu with a magnifying glass icon.
- Prompt on launch**: A checkbox next to the Source control branch dropdown.
- Labels**: A dropdown menu with a magnifying glass icon.
- Variables**: A section with tabs for 'YAML' (selected) and 'JSON'. It includes a 'Prompt on launch' checkbox and a delete icon.
- Options**: A section with checkboxes for 'Enable Webhook' and 'Enable Concurrent Jobs'.

At the bottom are 'Save' and 'Cancel' buttons.

Figure 6.1: Creating a workflow job template

Chapter 6 | Constructing Job Workflows

After creating a workflow job template, you can define an associated workflow by using the workflow visualizer.

Using the Workflow Visualizer

After you have created a workflow job template, the workflow visualizer becomes active in the **Visualizer** tab of the workflow job template. The workflow visualizer is a graphical interface for defining the node types to incorporate in a workflow, as well as the decision tree structure that chains the job templates together.

The workflow visualizer contains a single **Start** button that represents the starting point for the workflow. Click **Start** to start editing the workflow. The workflow visualizer displays a list of node types that you can use for the first step of the workflow. These node types are described in the following table. When added to the workflow, each node is identified by a symbol in the lower-left of the node.

Node type	Purpose	Symbol
Approval	Request manual approval before workflow continues.	
Inventory Source Sync	Synchronize inventory from a specified source.	I
Job Template	Run a job template.	JT
Project Sync	Synchronize a specified project.	P
Management Job	Run one of the management jobs.	M
Workflow Job Template	Run another workflow job template.	W

Being able to add not only job templates but other resources ensures that, for example, project and inventory resources are synchronized before the workflow job uses job templates that depend on those resources.

You can select the desired node type, the specific resource, and then click **Save** to add it as the first node in the workflow.

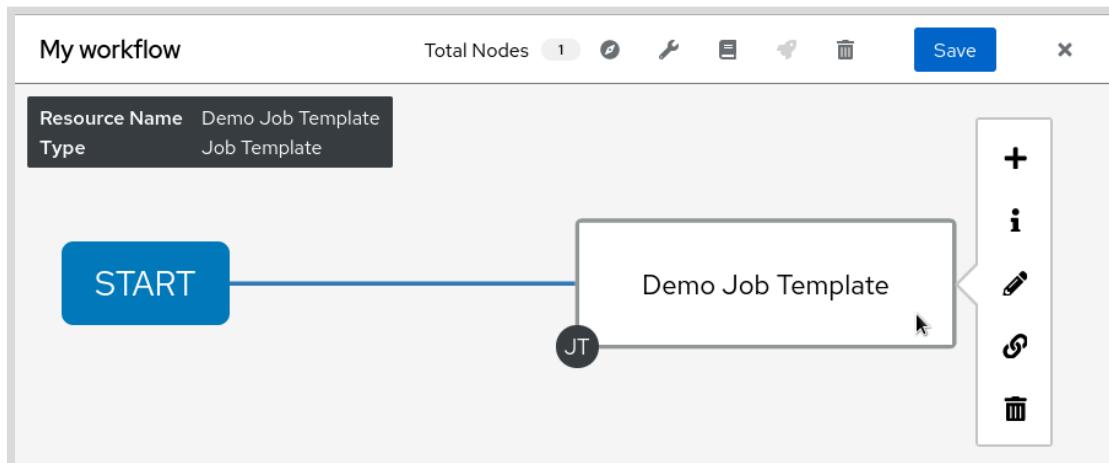


Figure 6.2: Adding a job template node and hovering over it

After you add a workflow node, you can hover over it to open a column of icons. You can use these icons to add another node, view node details, edit the node, link to an available node, or delete the node.

Chapter 6 | Constructing Job Workflows

When you add subsequent nodes, the **Run** page opens, prompting for the conditions under which the node should execute. You can choose one of three relationships between the new node and the preceding node:

Run	Node relationship
On Success	The node resource is executed upon successful completion of the actions associated with the previous node.
On Failure	The node resource is executed upon failure of the actions associated with the previous node.
Always	The node resource is executed regardless of the outcome of the actions associated with the previous node.

Select the desired condition and then click **Next** to select the new node type and the specific resource. Click **Save** to add the new node.

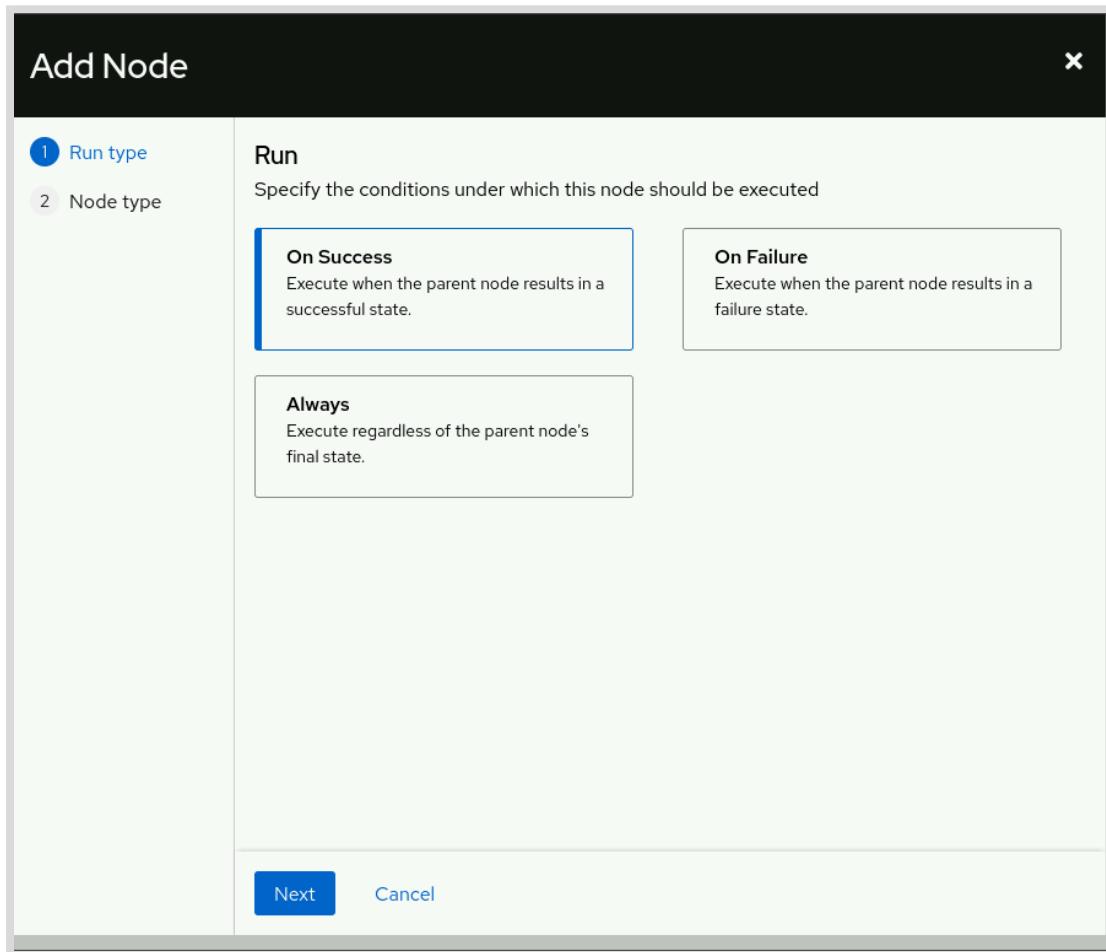


Figure 6.3: Adding an "On Success" node to the workflow

A node can have more than one child node. For example, you can add a child node to a parent node using an **On Success** association type. You can add a second child node to the same parent using an **On Failure** association type. This creates a branch in the workflow structure such that one course of action is taken upon the success of an action, and a different course is taken upon failure.

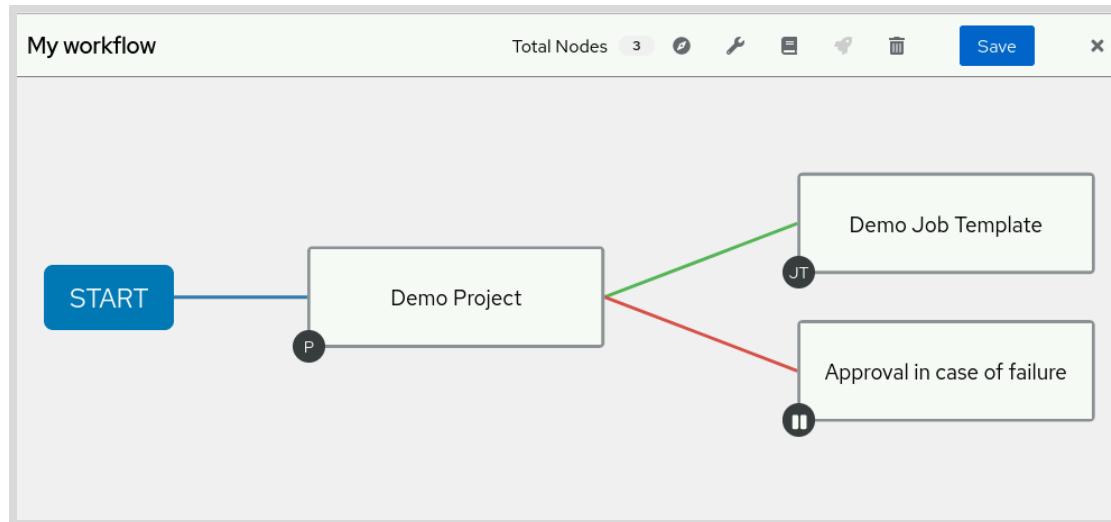


Figure 6.4: Adding multiple child nodes to the workflow

As you add nodes to a workflow, differently colored lines connecting the nodes in the workflow editor indicate the relationships between parent and child nodes. A green line indicates an **On Success** relationship between a parent and child node, and a red line indicates an **On Failure** relationship. A blue line indicates an **Always** relationship. In addition, if you hover your mouse pointer over a line, then the web UI displays the relationship type as text in the upper left corner of the visualizer.



Important

You can change the type of relationship between two nodes in the visualizer by hovering over the line connecting the two nodes, and then clicking the pencil icon to edit the relationship. You can also hover over the line connecting two nodes and click the plus icon to add a new node between the two nodes.

After you create the entire decision tree structure of the workflow in the workflow editor, click **Save** to save the workflow.

The new workflow job template displays on the same page as the job templates but with a different type (**Workflow Job Template**) and with an icon that directs you to the workflow visualizer.

The screenshot shows a table with columns: Name, Type, Last Ran, and Actions. There are two rows:

- Demo Job Template (Job Template)
- My workflow (Workflow Job Template)

The 'Actions' column for 'My workflow' contains several icons, with the 'Visualizer' icon being highlighted by a red box.

Figure 6.5: Visualizer icon for a workflow job template

Adding Multiple Nodes with the Same Relationship

You can have a node that has the same relationship with several subsequent nodes. For example, one node can be configured with **Always** relationships to several nodes.

In the following example, automation controller always runs the three job template nodes after running the project synchronization job for the **Demo Project** node. Because the **Check Virtual Machines**, **Check Cloud Servers**, and **Check Physical Servers** nodes are separate job templates, automation controller runs the jobs in parallel.

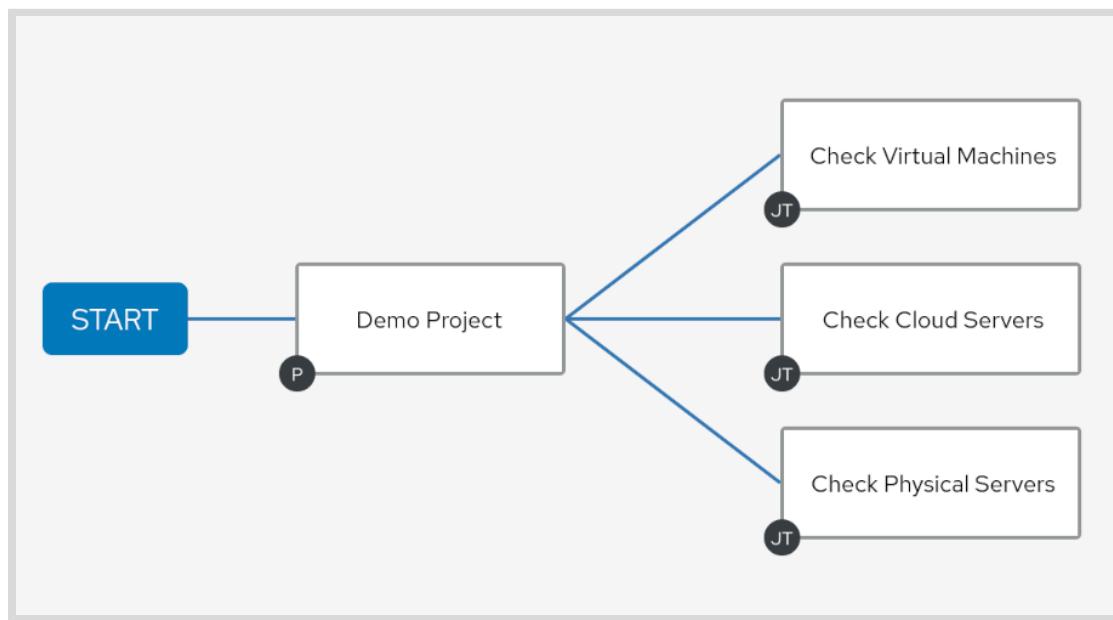


Figure 6.6: Multiple nodes with the same relationship to the preceding node

Creating Convergent Nodes

You can have a node that has multiple parent nodes. For example, you can have a node that runs if **any** of the parent nodes succeed, or only if **all** of its parent nodes succeed.

Chapter 6 | Constructing Job Workflows

If you have a configuration that uses a multiple node relationship (as described in the previous section), then you can configure convergent nodes. Hover over any of the parent nodes and add a new node. Hover over one of the other parent nodes and click the chain icon. A dashed gray arrow displays that follows your mouse pointer. Connect that arrow to the node you just added and specify when the node should run (always, on success, or on failure). That node now has two parent nodes, and is a *convergent node*.

If you edit the convergent node, on its **Node type** page there is a field labeled **Convergence**. This controls the conditions that cause the node to run.

- **Any** is the default setting. After all parent nodes complete, if any parent node triggers the convergent node, then automation controller runs the job for the convergent node.
- **All** requires that all parent nodes trigger the convergent node, based on their relationship, for the convergent node to run.

In the following example from the visualizer, the convergent node **Update application** only runs if all three of its parent nodes report success.

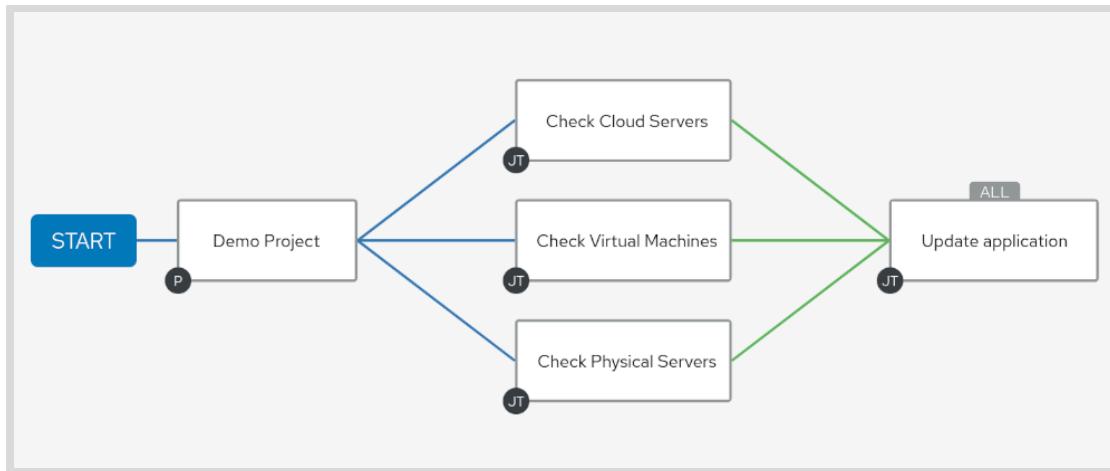


Figure 6.7: Convergent node with multiple parent nodes

Workflow Job Template Surveys

Workflow job templates have access to many of the features that have been discussed for job templates. Like job templates, you can add surveys to workflow job templates so that users can interactively set extra variables.



Note

When you add a survey to a workflow job template, the resulting extra variables are accessible by every job executed by the workflow.

Launching Workflow Jobs

Like job templates, users need the **execute** role on the workflow job template to execute it. Users with the **execute** role can launch a job through a workflow job template, even if they do not have permissions to independently launch the job templates it uses.

The procedure to launch a workflow job template is similar to how you launch a job template:

1. Log in as a user that has the **execute** role on the desired workflow job template.

Chapter 6 | Constructing Job Workflows

2. Navigate to **Resources > Templates** to list the available templates.
3. Click the **Launch Template** icon for the desired workflow job template.

Evaluating Workflow Job Execution

After you launch a workflow job template, the automation controller web UI redirects you to the **Output** tab for this workflow job template, and displays the progress and result in the workflow visualizer.

The workflow job page consists of two tabs. The **Details** tab displays details of the parameters for the workflow job template. The **Output** tab displays the progress of the job through the steps in the workflow.

As each step in the workflow job completes, the web UI indicates whether the step succeeded or failed. If the step succeeds, then the web UI outlines the node in green and adds a box to the node in which the upper half of the box is colored green. If the step fails, then the web UI outlines the node in red and adds a box to the node in which the lower half of the box is colored red. Within each successful or failed node, the web UI displays the amount of time it took to complete the step. Nodes that are not run are outlined in gray.

Progress from one node to another is represented in the same way as it is in the visualizer; by colored lines indicating the decision responsible for the progression. Green indicates that the next node runs if the previous node succeeds. Red indicates that the next node runs if the previous node fails. Blue indicates that the next node always runs.

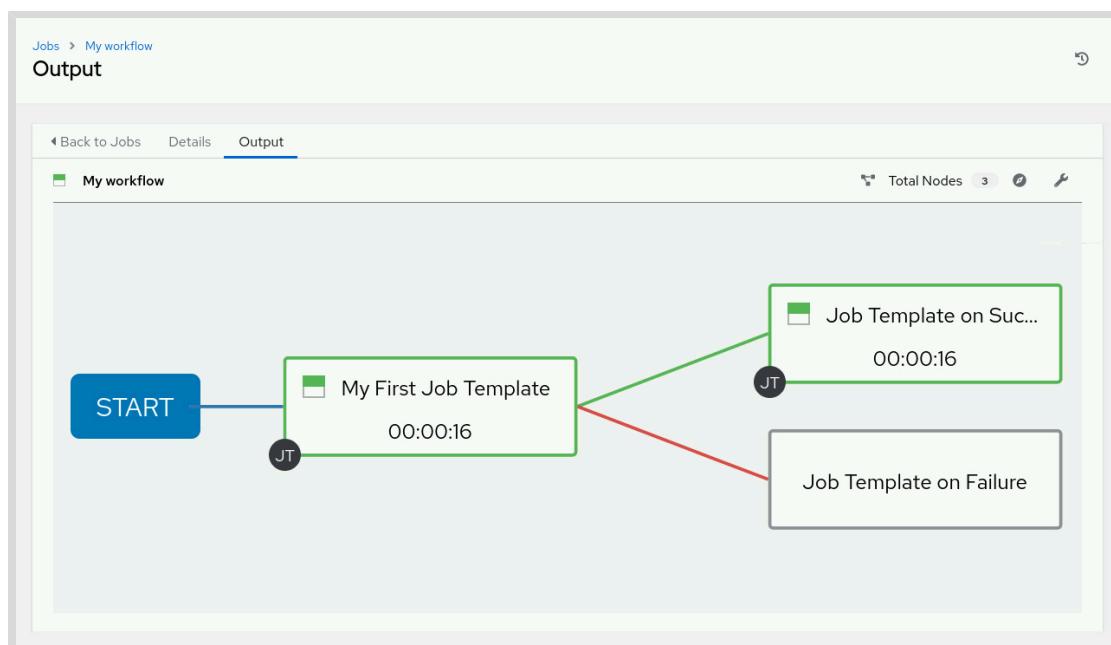


Figure 6.8: Workflow job progress

You can examine the live output of the running workflow job or review the output after the job completes. In a workflow diagram that represents a currently running job or completed job, you can click each node to display the results and standard output for the job run.



References

Ansible Documentation: Workflows

<https://docs.ansible.com/automation-controller/latest/html/userguide/workflows.html>

Ansible Documentation: Workflow Job Templates

https://docs.ansible.com/automation-controller/latest/html/userguide/workflow_templates.html

► Guided Exercise

Creating Workflow Job Templates and Launching Workflow Jobs

Create a workflow job template consisting of a branching chain of jobs and launch it using the web UI.

Outcomes

- Create a workflow job template.
- Launch a workflow from the automation controller web UI.

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that automation controller is installed and configured with any necessary resources created in previous exercises.

```
[student@workstation ~]$ lab start workflow-template
```

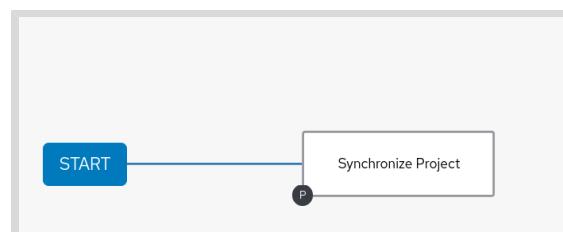
Instructions

- 1. Review the following automation controller resources created by the `lab` command for this exercise.
- The Git project that contains the playbooks used by several job templates that set up secure web servers and populate them with content
 - The job templates for those playbooks, as well as another playbook that restores the web servers to a basic configuration if the workflow fails
 - The Dev inventory for the job templates, and an `Email AAP Admins` notification template that can be used to report on the status of job completion

In this exercise, you take these components and use them to create a workflow job template.

1. Navigate to `https://controller.lab.example.com` and log in as the `admin` user with `redhat` as the password.
2. Navigate to **Resources > Projects** and then click the link for the **DEV - Secure Webservers** project. This is the Git project that contains the playbooks to set up the secure web servers.
3. Navigate to **Resources > Templates** and then click the link for the **DEV - Deploy HTTPS** job template. This job template deploys secure web servers on the hosts.

Chapter 6 | Constructing Job Workflows

- 1.4. Navigate to **Resources > Templates** and then click the link for the **DEV - Populate Webservers** job template. This job template populates the web servers with content pulled from a Git repository.
 - 1.5. Navigate to **Resources > Templates** and then click the link for the **DEV - Revert HTTPS** job template. In case of failure, this job template restores the web servers to a basic web server configuration.
 - 1.6. Navigate to **Resources > Inventories** and then click the link for the **Dev** inventory. This is the inventory for the secure web servers.
 - 1.7. Navigate to **Administration > Notifications** and then click the link for the **Email AAP Admins** notification template. This notification template sends emails to report the status of jobs.
- ▶ 2. Create a new workflow job template named **DEV - Deploy HTTPS and Populate Webservers** that uses the **Dev** inventory to deploy the secure web servers to managed hosts.
- 2.1. Navigate to **Resources > Templates** and then click **Add > Add workflow template**.
 - 2.2. On the next page, fill in the details as follows:
- | Field | Value |
|--------------|---|
| Name | DEV - Deploy HTTPS and Populate Webservers |
| Organization | Default |
| Inventory | Dev |
- 2.3. Click **Save**.
- ▶ 3. Add a **Project Sync** node named **Synchronize Project** to synchronize the **DEV - Secure Webservers** project.
- 3.1. On the workflow visualizer page, click **Start**.
 - 3.2. On the **Add Node** page, select **Project Sync** in the **Node Type** list.
 - 3.3. Select the **DEV - Secure Webservers** project, scroll down to view the **Node Alias** field, and enter **Synchronize Project** in that field.
 - 3.4. Click **Save**. On the workflow visualizer page under **Synchronize Project**, notice the **P** symbol that indicates it is a project sync node.
- 
- Figure 6.9: Project synchronization node**
- ▶ 4. Add a **Job Template** node that launches the **DEV - Deploy HTTPS** job template to deploy the secure web servers if the **Synchronize Project** node succeeds.

Chapter 6 | Constructing Job Workflows

- 4.1. In the workflow visualizer, click the **Synchronize Project** node and click the + icon to add a new node.
- 4.2. On the **Run type** page, select the **On Success** condition and then click **Next**.
- 4.3. Select the **DEV - Deploy HTTPS** job template, scroll down to view the **Node Alias** field, and enter **Deploy HTTPS** in that field.
- 4.4. Click **Save**. Notice the **JT** symbol, which indicates it is a job template node.

**Note**

You might need to increase the display resolution on the **workstation** machine, decrease the web browser zoom, or both, in order to see the entire visualizer display screen.

Additionally, if your mouse has a scroll wheel, then you can use the scroll wheel to zoom in and out of the visualizer.

- ▶ 5. Add a job template node that launches the **DEV - Populate Webservers** job template to populate the secure web servers with content if the **Deploy HTTPS** node succeeds.
 - 5.1. In the workflow visualizer, click the **Deploy HTTPS** node and then click the + icon to add a new node.
 - 5.2. On the **Run type** page, select the **On Success** condition and then click **Next**.
 - 5.3. Select the **DEV - Populate Webservers** job template, scroll down to view the **Node Alias** field, and enter **Populate Webservers** in that field.
 - 5.4. Click **Save**.
- ▶ 6. Add a job template node that launches the **DEV - Revert HTTPS** job template to revert the configuration of the secure web servers if the **Deploy HTTPS** node fails.
 - 6.1. In the workflow visualizer, click the **Deploy HTTPS** node and then click the + icon to add a new node.
 - 6.2. On the **Run type** page, select the **On Failure** condition and then click **Next**.
 - 6.3. Select the **DEV - Revert HTTPS** job template, scroll down to view the **Node Alias** field, and enter **Revert HTTPS** in that field.

**Note**

You might need to go to page 2 to look for the **DEV - Revert HTTPS** job template. Alternatively, you can search for "Revert" to see the job template.

- 6.4. Click **Save**.
- 6.5. Click **Save** again in the upper-right corner to save this workflow template.

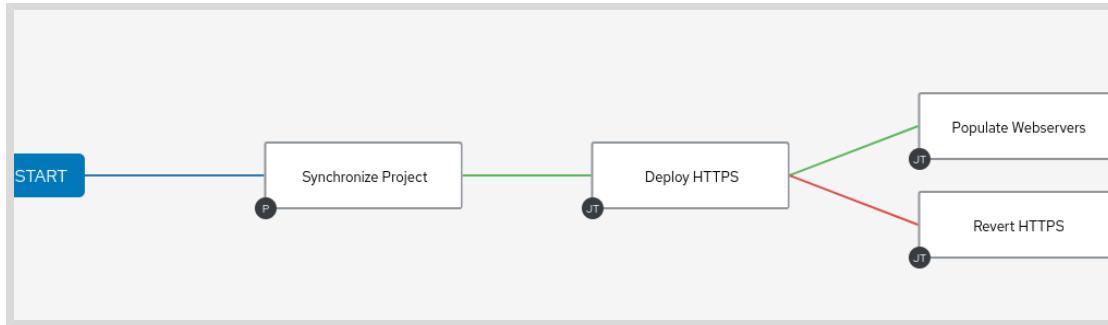


Figure 6.10: Workflow job template

- ▶ 7. Launch the workflow job template and observe the result. The Deploy HTTPS job template fails and triggers the Revert HTTPS job template.
- 7.1. Navigate to Resources > Templates.
 - 7.2. Click the Launch Template icon for the DEV - Deploy HTTPS and Populate Webservers.
 - 7.3. Observe that the Deploy HTTPS job template fails and triggers the Revert HTTPS job template.
 - 7.4. Click the Deploy HTTPS node and then click the Output tab to determine the cause of the failure.

The output indicates that the playbook successfully started and enabled the httpd service on the servera.lab.example.com host, but that the playbook failed to start the httpd service on the serverb.lab.example.com host.

- ▶ 8. Review the httpd logs on serverb.lab.example.com to identify the problem.

```
[student@workstation ~]$ ssh serverb journalctl -u httpd | grep error -A4
Jun 01 09:29:57 serverb.lab.example.com httpd[3983]: AH00526: Syntax error on line
86 of /etc/httpd/conf.d/ssl.conf:
Jun 01 09:29:57 serverb.lab.example.com httpd[3983]: SSLCertificateFile: file '/
etc/pki/tls/certs/servera.lab.example.com.crt' does not exist or is empty
Jun 01 09:29:57 serverb.lab.example.com systemd[1]: httpd.service: Main process
exited, code=exited, status=1/FAILURE
Jun 01 09:29:57 serverb.lab.example.com systemd[1]: httpd.service: Failed with
result 'exit-code'.
Jun 01 09:29:57 serverb.lab.example.com systemd[1]: Failed to start The Apache
HTTP Server.
```

- The problem is that the ssl.conf configuration file incorrectly specifies servera.lab.example.com in the file paths for the HTTPS certificate file and the associated key file.
- For serverb.lab.example.com, the certificate file should be /etc/pki/tls/certs/serverb.lab.example.com.crt.
- The ssl.conf configuration file should be deployed as a Jinja2 template that uses the ansible_facts['fqdn'] fact to generate the correct path for the certificate and key files for each server.

- ▶ 9. Repair the issue with the playbook.

Chapter 6 | Constructing Job Workflows

Clone the Git repository for the project, convert the affected file into a Jinja2 template, make the change, and update the `deploy_https.yml` playbook so that it correctly deploys the template to the managed hosts being provisioned. Commit your changes to your local Git repository and push them to the remote GitLab server.

- 9.1. From a terminal, create the `/home/student/git-repos` directory if it does not already exist, and then change into it.

```
[student@workstation ~]$ mkdir -p ~/git-repos/  
[student@workstation ~]$ cd ~/git-repos/
```

- 9.2. Clone the `https://git.lab.example.com/git/secure_webservers_DEV.git` Git repository into the `/home/student/git-repos` directory.

```
[student@workstation git-repos]$ git clone \  
> https://git.lab.example.com/git/secure_webservers_DEV.git  
Cloning into 'secure_webservers_DEV'...  
...output omitted...  
[student@workstation git-repos]$ cd secure_webservers_DEV
```

- 9.3. Look for lines starting with the `SSLCertificate` pattern in the `files/ssl.conf` file. Notice how the configuration file uses the certificate and key for the `servera.lab.example.com` host.

```
[student@workstation secure_web_servers_DEV]$ grep ^SSLCertificate files/ssl.conf  
SSLCertificateFile /etc/pki/tls/certs/servera.lab.example.com.crt  
SSLCertificateKeyFile /etc/pki/tls/private/servera.lab.example.com.key
```

- 9.4. Modify the `files/ssl.conf` file to replace both instances of `servera.lab.example.com` with the `ansible_facts['fqdn']` variable. The modified lines should have the following content:

```
SSLCertificateFile /etc/pki/tls/certs/{{ ansible_facts['fqdn'] }}.crt  
SSLCertificateKeyFile /etc/pki/tls/private/{{ ansible_facts['fqdn'] }}.key
```

- 9.5. Rename the `files` directory to `templates` and the `ssl.conf` file to `ssl.conf.j2`

```
[student@workstation secure_webservers_DEV]$ mv files/ templates  
[student@workstation secure_webservers_DEV]$ mv templates/ssl.conf \  
> templates/ssl.conf.j2
```

- 9.6. Update the `deploy_https.yml` file to use the `template` module instead of the `copy` module and change the `src` option to reference the new `templates/ssl.conf.j2` file. The `Configure Apache for HTTPS` task should have the following content:

Chapter 6 | Constructing Job Workflows

```
- name: Configure Apache for HTTPS
  template:
    src: templates/ssl.conf.j2
    dest: /etc/httpd/conf.d/ssl.conf
    notify: restart_apache
```

9.7. Commit and push the changes to the Git repository.

```
[student@workstation secure_webservers_DEV]$ git add deploy_https.yml \
> files/templates/

[student@workstation secure_web servers_DEV]$ git commit -m \
> "Fix Deploy HTTPS playbook"
[main 94dd74c] Fix Deploy HTTPS playbook
 2 files changed, 4 insertions(+), 4 deletions(-)
 rename files/ssl.conf => templates/ssl.conf.j2 (98%)

[student@workstation secure_web servers_DEV]$ git push -u origin
Enumerating objects: 7, done.
...output omitted...
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

- **10.** Launch the template through the automation controller web UI. The Deploy HTTPS job template completes successfully and launches the Populate Webservers job template.

- 10.1. Navigate to Resources > Templates.
- 10.2. Click the Launch Template icon for the DEV - Deploy HTTPS and Populate Webservers workflow job template.
- 10.3. The first step updates the project with the changes you have committed.
- 10.4. Observe that this time the Deploy HTTPS job template completes successfully and as a result the Populate Webservers job template is launched.

- **11.** Verify that the secure web servers are configured and the slides are populated.

- 11.1. Navigate to `https://servera.lab.example.com` and confirm that you see slides from DO467.
- 11.2. Navigate to `https://serverb.lab.example.com` and confirm that you see slides from DO467.

Finish

On the workstation machine, change to the student user home directory and use the lab command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish workflow-template
```

Requiring Approvals in Workflow Jobs

Objectives

- Create an approval node in a workflow job template that uses notifications to request permission from an administrator for the job to continue, and approve or reject those requests.

Approval Nodes

For various reasons, you might want to require that a particular person or group explicitly approve or reject execution of a workflow job template.

You can add approval nodes to a workflow job template that pause the workflow until a user with an appropriate role manually approves or rejects further execution of the workflow. You can also set a time limit on the approval node, so that further execution is automatically rejected if execution is not approved before the limit expires. It is possible to add several approval nodes inside a workflow, on various branches of a workflow, and at the start of a workflow.

Some reasons to use approval nodes within your workflow job templates include:

- Enabling users to launch workflow jobs, but requiring that someone else approve it before it runs.
- Pausing the workflow if a job template fails, so that you can investigate the problem, and before the workflow attempts to recover or clean up the affected systems.
- Pausing a workflow that is succeeding, so that you can complete testing external to the workflow before approving its final steps.

Adding Approval Nodes to Workflows

You can create an approval node through the workflow visualizer in the automation controller UI. To create an approval node, you need to be logged in as a user with sufficient permissions:

- A user with the `Admin` role on the workflow job template
- A user with the `Workflow Admin` role for the organization
- A user with the `System Administrator` user type

There are several different ways to add approval nodes to a workflow:

- You can click the link between two nodes and then click the plus (+) icon to create an approval node between those two nodes.
- You can click a node and then click the plus (+) icon to create an approval node that follows that node.
- You can edit an existing node to convert it into an approval node.

Whichever method you use, select **Approval** in the **Node Type** field when you create the new node or edit the existing node.

For example, if you need approval before running a job template then click directly on the line. A pop-up window opens that displays the message **Add a new node between these two nodes**.

Chapter 6 | Constructing Job Workflows

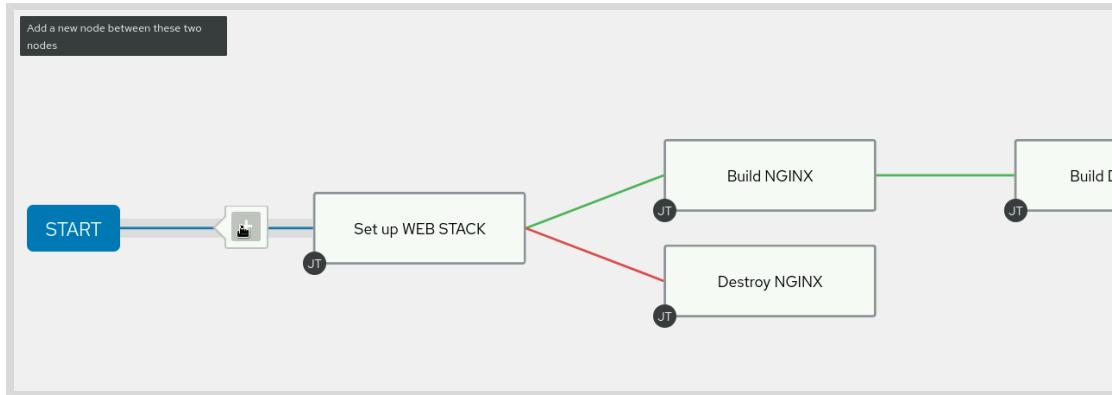


Figure 6.11: Creating an approval node before the first job template node

If you want to replace an existing node with an approval node, click the **Edit this node** pencil icon on the job template.

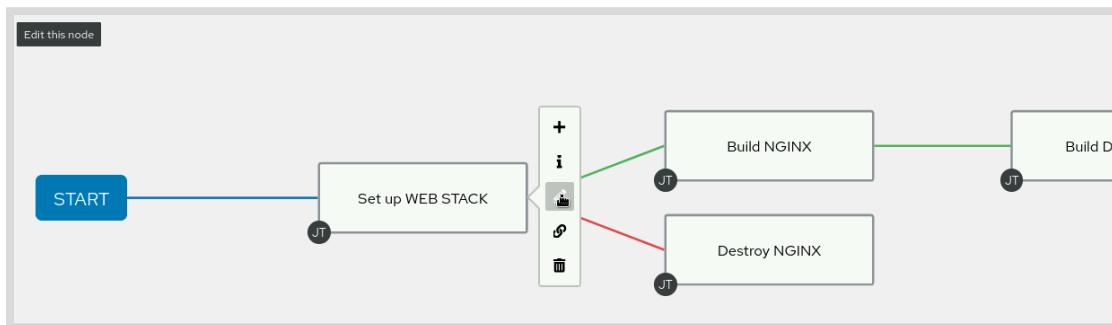


Figure 6.12: Editing the "Set up WEB STACK" job template node to make it an approval node instead

Approving and Denying Workflow Approval Requests

To view workflow approval requests, you need to have the Admin, Execute, Read, or Approve role on the workflow job template, or the System Administrator or System Auditor user type.

The **Pending Workflow Approvals** icon at the top of the page displays the number of workflows that require your approval. This number indicates how many workflow approval requests are pending.



Figure 6.13: Notification waiting

Click the **Pending Workflow Approvals** icon to see pending workflow approval requests, or navigate to **Views > Workflow Approvals** to see all workflow approval requests. Click the name of the pending workflow approval request that you want to approve or deny.

To actually approve or deny a workflow approval request, you must have either the Admin or the Approve role on the workflow job template. As always, a user with the System Administrator user type can perform any action. If you do not have sufficient access, then the **Approve** and **Deny** buttons are unavailable.

Chapter 6 | Constructing Job Workflows

The screenshot shows a 'Workflow Job' named 'Needs Approval' with a status of 'Workflow Job' and 'Workflow Job Template' both set to 'DEV - Deploy WEB STACK'. It was created on 5/12/2022 at 5:35:18 PM by 'admin' and last modified on the same date at 5:35:18 PM. The 'Elapsed' time is 00:00:09. At the bottom, there are two buttons: 'Approve' (blue) and 'Deny' (red).

Figure 6.14: Pending workflow approval request

Approval Time-outs

Often, you want workflow approval requests to be approved or denied within a certain period of time. You can set a time-out that automatically denies a request if it has not been approved within a certain time limit.

You can set a time-out from the **Add Node** page. Click the down arrow of the **Node Type** drop-down list and select **Approval**. There are two fields in the **Timeout** section, for minutes and seconds. This specifies the length of time before the approval request expires and is automatically denied.

If you leave both fields blank, then no time-out is set. The workflow approval request never expires, and the workflow job waits indefinitely for manual approval or denial.

The screenshot shows the 'Add Node' page with the 'Node type' set to 'Approval'. In the 'Timeout' section, the 'min' field is set to '30' and the 'sec' field is set to '0', both highlighted with a red box. Other fields shown include 'Name' (Workflow approval timeout), 'Description', 'Convergence' (set to 'Any'), and 'Node Alias'.

Figure 6.15: Setting the approval time-out

When you review pending workflow approval requests in automation controller, you can see their expiration times in the **Status** column.

Chapter 6 | Constructing Job Workflows

The screenshot shows a table of workflow approvals. The first row, titled 'Workflow approval timeout', has its status cell highlighted with a red box. The status is listed as 'Expires on 5/4/2022, 9:20:35 PM'.

Name	Job	Started	Status
Workflow approval timeout	88 - Workflow Approval Timeout	5/4/2022, 8:50:35 PM	Expires on 5/4/2022, 9:20:35 PM

Figure 6.16: Workflow approval with an expiration time

If someone does not approve the pending workflow approval request within the specified time limit, it is automatically denied and the next nodes that have an Always or On Failure configuration run.

If the pending workflow approval request is approved, then the next nodes that have an Always or On Success configuration run.

Approval Notifications

It is a good idea to use notifications in your workflow job template to alert the people who address workflow approval requests that a new request needs review. Notifications also alert the users about whether a workflow approval has been approved, denied, or timed out.

To configure your workflow job template to send workflow approval notifications, first make sure that you have configured an appropriate notification. IRC, Slack, email, or webhooks that you integrate with other chat or notification applications might be appropriate. You can customize the message that the notification sends if necessary.

Navigate to **Templates** and select the name of your workflow job template. Click **Notifications** and find the notification you want to use. You can use the search field to help locate the notification. For that notification, look in the **Options** column and enable **Approval**.

The screenshot shows the 'Notifications' tab for the 'DEV - Deploy WEB STACK' template. A single notification is listed: 'Notify web services workflow approvers' of type 'Webhook'. In the 'Options' column, the 'Approval' checkbox is selected, while 'Start', 'Success', and 'Failure' are unselected.

Figure 6.17: Enabling a notification for approval activity on a workflow job template

Remember to test your workflow job template to confirm that workflow approval notifications are being sent.



References

For more information, see the *Approval nodes* section in the *Automation Controller User Guide* at
https://docs.ansible.com/automation-controller/latest/html/userguide/workflow_templates.html#ug-wf-approval-nodes

► Guided Exercise

Requiring Approvals in Workflow Jobs

Create a workflow job template that requires approval to run.

Outcomes

- Modify a workflow job template by adding an approval node to the workflow.
- Configure an email notification when the workflow requires an approval.
- Launch and approve the workflow job.

Before You Begin

As the student user on the **workstation** machine, use the **lab** command to prepare your system for this exercise.

This command ensures that automation controller is installed and configured with any necessary resources created in previous exercises.

```
[student@workstation ~]$ lab start workflow-approval
```

Instructions

- 1. Review the automation controller resources created by the **lab** command for this exercise.
- The **lab** command configured a workflow job template named **DEV - Deploy HTTPS and Populate Webservers**. You can add an approval node to that workflow job template that uses an existing notification template to alert your operations team by email. If you add that approval node at the beginning of the workflow job template, then your operations team must approve jobs launched using the workflow job template before the next `on success` node in the job runs.
- 1.1. Navigate to <https://controller.lab.example.com> and log in as the **admin** user with **redhat** as the password.
 - 1.2. Navigate to **Resources > Templates**. The **lab** command created the workflow job template and the job templates used in this exercise. The **DEV - Deploy HTTPS and Populate Webservers** workflow job template runs the **DEV - Deploy HTTPS** job template and then runs either the **DEV - Populate Webservers** job template if the previous job succeeds, or the **DEV - Revert HTTPS** job template if the previous job fails.
 - 1.3. Navigate to **Administration > Notifications**. You can see the **Email AAP Admins** notification listed as a notification template of type **Email**. Click the **Test notification** icon for the notification template and wait for the **Status** column to display the **Successful** message. This ensures that the notification is working as expected.
 - 1.4. Navigate to **Access > Teams**. You can see the **Developers** and **Operations** teams listed as part of the **Default** organization.

Chapter 6 | Constructing Job Workflows

- ▶ 2. Insert a workflow approval node as the first step for the workflow job template and modify the workflow so that it only executes when approved.
- 2.1. Navigate to Resources > Templates and click the Visualizer icon for the DEV - Deploy HTTPS and Populate Webservers workflow job template.
 - 2.2. In the workflow visualizer, hover over the line connecting the START button and the Synchronize Project node, and then click the + icon (notice the Add a new node between these two nodes description at the upper-left corner of the workflow editor).
 - 2.3. Create the node using the following information. Do not modify any of the other fields. Click Save when finished.

Field	Value
Node Type	Approval
Name	Dev approval
Description	Require approval before even starting the workflow
Timeout (min)	60
Alias	Approval

- 2.4. Hover over the line connecting the new Approval approval node and the Synchronize Project node and click the Edit this link icon.

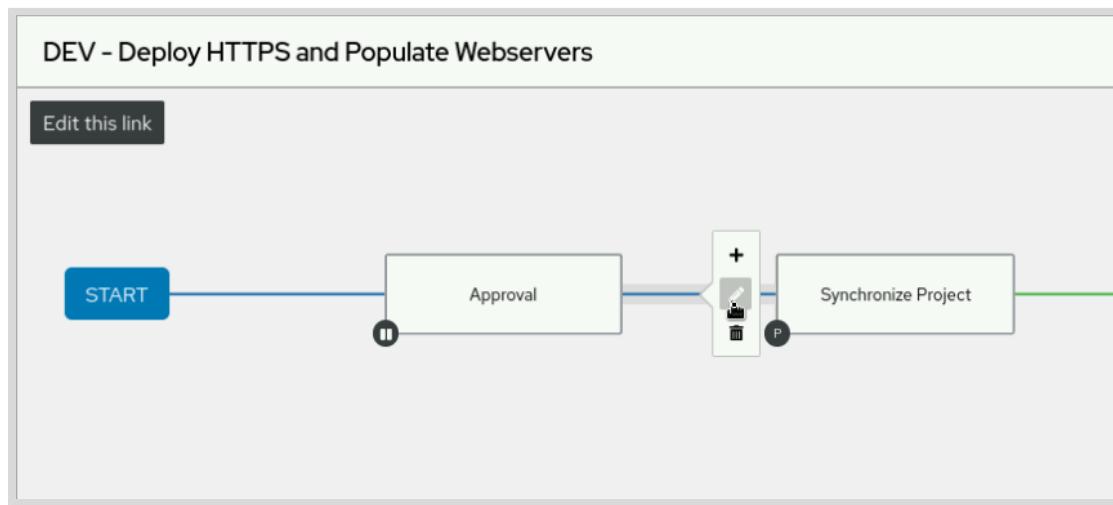


Figure 6.18: Editing an existing link in the workflow

- 2.5. In the Run list, choose On Success and click Save. Notice how the line that connects the two nodes changes from blue (Always) to green (On Success).
- 2.6. Click Save in the upper-right corner to save the changes you made to the workflow job template.

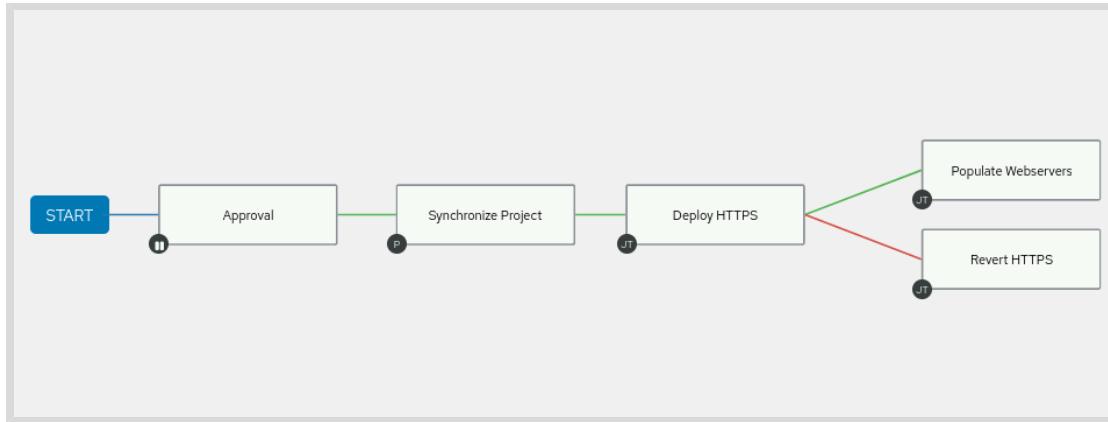


Figure 6.19: Inserted approval node as the first step for the workflow

- ▶ 3. On the DEV - Deploy HTTPS and Populate Webservers workflow job template, assign the **Approve** role to the Developers team. Give the **Operations** team the **Execute** role.
 - 3.1. Navigate to **Resources > Templates** and click the **DEV - Deploy HTTPS and Populate Webservers** link.
 - 3.2. Click the **Access** tab and then click **Add**.
 - 3.3. Click **Teams** and then click **Next**.
 - 3.4. Select the **Developers** team and then click **Next**.
 - 3.5. Select the **Approve** role and then click **Save** to assign the role. Because they are part of the **Developers** team, the **daniel** and **david** users now have the **Approve** role.
 - 3.6. Click **Add** again, select **Teams**, and then click **Next**.
 - 3.7. Select the **Operations** team and then click **Next**.
 - 3.8. Select the **Execute** role and then click **Save** to assign the role. Because they are part of the **Operations** team, the **oliver** and **ophelia** users now have the **Execute** role.
- ▶ 4. Configure the workflow job template to use the **Email AAP Admins** notification template. When a workflow job launched by the template requests an approval, it should trigger the notification.
 - 4.1. Navigate to **Resources > Templates** and click the **DEV - Deploy HTTPS and Populate Webservers** link.
 - 4.2. Click the **Notifications** tab and set **Approval** to on for the **Email AAP Admins** notification.

The screenshot shows the 'Notifications' tab in the Red Hat OpenShift web UI. At the top, there are tabs for 'Back to Templates', 'Details', 'Access', 'Notifications' (which is selected), 'Schedules', 'Visualizer', 'Jobs', and 'Survey'. Below the tabs is a search bar with a magnifying glass icon and a dropdown menu for 'Name'. The main area contains a table with the following data:

Name	Type	Options
Email AAP Admins	Email	<input checked="" type="checkbox"/> Approval <input type="checkbox"/> Start <input type="checkbox"/> Success <input type="checkbox"/> Failure

At the bottom of the table are navigation links for '1-1 of 1 items' and '1 of 1 page'.

Figure 6.20: Notification for a required approval

- ▶ 5. Run the workflow job template as user **ophelia**, who is a member of the **Operations** team.
 - 5.1. Navigate to **admin > Logout** to log out of the **admin** account. Log in again, using the **ophelia** user, with **redhat123** as the password.



Important

Because you were on the **Notifications** tab when you logged out as the **admin** user, when you log in with the **ophelia** user, the web UI attempts to return to the same tab.

The **ophelia** user only has the **Execute** role on the workflow job template. Consequently, the **ophelia** user does not have permission to see the notifications associated with the workflow job template, and the web UI displays a **Not Found** message.

- 5.2. Navigate to **Resources > Templates**. Because the **Operations** team has the **Execute** role, and the **ophelia** user is a member of that team, the **ophelia** user can launch the **DEV - Deploy HTTPS and Populate Webservers** workflow job template.
 - 5.3. Click the **Launch Template** icon for the **DEV - Deploy HTTPS and Populate Webservers** workflow job template. A workflow job launches, but its workflow stops at the approval node.
 - 5.4. Notice that the **Pending Workflow Approvals** icon at the top of the page indicates one pending workflow approval. Click the notification icon in the top of the page, and then select the checkbox for the **Dev approval** workflow approval. Even though the **ophelia** user can see the workflow approval, that user does not have sufficient permissions to approve it.
- ▶ 6. Verify that the workflow job triggered a notification email. In a real situation, this mail would reach the people with permissions to approve the workflow. In this exercise, those people are the members of the **Developers** team.

- 6.1. Open a terminal and connect to the utility server.

```
[student@workstation ~]$ ssh student@utility
```

- 6.2. Use the `tail` command to read incoming email messages delivered to the local mailbox file of the `student` user. You should see the email that was sent for the approval. Your message looks similar to the following example:

```
[student@utility ~]$ tail -f /var/mail/student
{
  "id": 4,
  "name": "Dev approval",
  "url": "https://controller.lab.example.com/#/jobs/workflow/3",
  "created_by": "ophelia",
  "started": "2022-05-05T18:00:11.884846+00:00",
  "finished": null,
  "status": "pending",
  "traceback": ""
}
```

- 7. As user `david`, who is a member of the `Developers` team, approve the workflow job.

- 7.1. Navigate to `ophelia > Logout` to log out as user `ophelia`. Log in again as the `david` user, with `redhat123` as the password.
- 7.2. Click the notification icon, and then select the checkbox for the `Dev approval` workflow approval. The `david` user has the permissions needed to approve or deny the approval request.
- 7.3. Click **Approve** to approve the request.
- 7.4. Go to the terminal window that is running the `tail` command and wait. After a few seconds, you should see a notification email arrive that looks similar to the following example:

```
From system@controller.lab.example.com Thu May 5 14:18:35 2022
Return-Path: <system@controller.lab.example.com>
X-Original-To: aap-admins@lab.example.com
Delivered-To: aap-admins@lab.example.com
Received: from controller.lab.example.com (controller.lab.example.com
[172.25.250.7])
      by utility.lab.example.com (Postfix) with ESMTPS id B5ABD938DAC
      for <aap-admins@lab.example.com>; Thu, 5 May 2022 14:18:35 -0400 (EDT)
Received: from controller.lab.example.com (localhost [IPv6:::1])
      by controller.lab.example.com (Postfix) with ESMTP id 9DF3A30404AA
      for <aap-admins@lab.example.com>; Thu, 5 May 2022 14:18:35 -0400 (EDT)
Content-Type: text/plain; charset="utf-8"
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit
Subject: The approval node "Dev approval" was approved.
https://controller.lab.example.com/#/jobs/workflow/3
From: system@controller.lab.example.com
To: aap-admins@lab.example.com
Date: Thu, 05 May 2022 18:18:35 -0000
Message-ID:
```

Chapter 6 | Constructing Job Workflows

```
<165177471562.4668.18225658775475317495@controller.lab.example.com>

The approval node "Dev approval" was approved. https://controller.lab.example.com/
#/jobs/workflow/3

{
  "id": 4,
  "name": "Dev approval",
  "url": "https://controller.lab.example.com/#/jobs/workflow/3",
  "created_by": "ophelia",
  "started": "2022-05-05T18:00:11.884846+00:00",
  "finished": "2022-05-05T18:18:35.040965+00:00",
  "status": "successful",
  "traceback": ""
}
```

When finished, press **Ctrl+C** to exit the **tail** command.

7.5. Exit the terminal session on the **utility** system.

```
[student@utility ~]$ exit
```

► 8. Verify that the workflow job completed successfully.

- 8.1. Navigate to **Views > Jobs** and wait for the workflow job to finish.
- 8.2. After the job succeeds, click the link for the job name, such as the **3-DEV - Deploy HTTPS and Populate Webservers** link, to see the workflow results for the completed job. The number in the job name might be different from the one listed here.

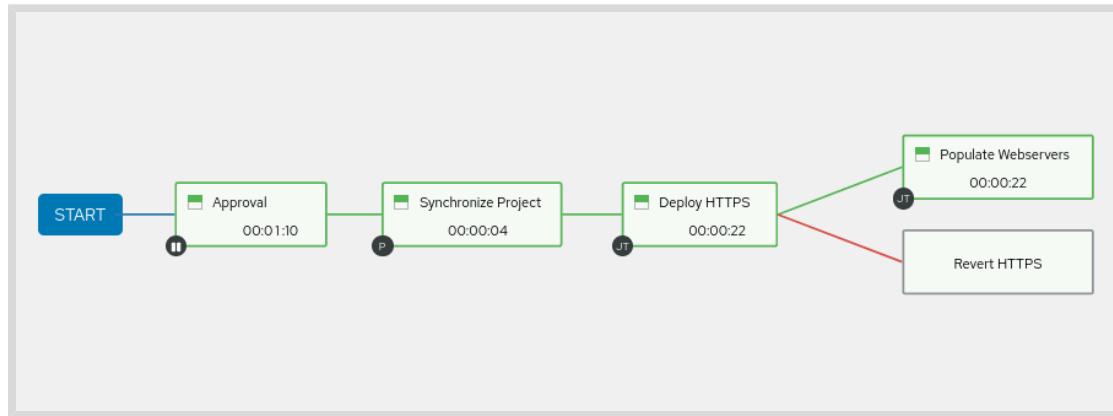


Figure 6.21: The results of the completed workflow

- 9. (Optional) Repeat the exercise starting at step 5. This time, have the **david** user deny the approval request for the workflow job.
- 10. Log out of the automation controller web UI.

Finish

On the **workstation** machine, change to the **student** user home directory and use the **lab** command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish workflow-approval
```

▶ Lab

Constructing Job Workflows

Create a new workflow job template that uses a survey to set variables, and fact caching to speed up the workflow.

Outcomes

- Create a workflow job template.
- Create an associated survey and notification template.
- Launch the workflow job template from the automation controller web UI.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command ensures that automation controller is installed and configured with any necessary resources created in previous exercises.

```
[student@workstation ~]$ lab start workflow-review
```

Instructions

Create a workflow job template that deploys your web servers to the test environment, and after approval, to production. As part of this process, rename the existing `TEST webservers setup` job template to `Webservers setup` and configure the job template to prompt for an inventory. This allows you to use a single job template for both the test and production environments. The `Webservers setup` job template needs to use facts cached from earlier in the workflow. The workflow job template also needs to prompt the user with a survey to set some variable values when the workflow job template is launched.

1. Go to `https://controller.lab.example.com` and log in as the `admin` user with `redhat` as the password.
2. Make sure that the per-host Ansible fact cache time-out for automation controller is set for exactly one day (86400 seconds). This ensures that facts are cached for hosts but also expire if those facts have not been refreshed recently.
3. Update the `TEST webservers setup` job template with the following information, leaving the other fields untouched. Make sure to clear the existing inventory entry for the job template.

Chapter 6 | Constructing Job Workflows

Field	Value
Name	Webservers setup
Description	Setup apache webservers
Inventory	Prompt on launch
Options	Enable Fact Storage

4. Create a workflow job template called **From Test to Prod** with the following information. In the following steps, you create nodes for the workflow job template.

Workflow Template

Field	Value
Name	From Test to Prod
Description	Deploy to Test and on success deploy to Prod
Organization	Default

5. Start the workflow for the **From Test to Prod** workflow job template by synchronizing the **My Webservers** project. Create the first node using the following information:

Workflow Node #1

Field	Value
Node Type	Project Sync
Name	My Webservers

6. If the **My Webservers** node succeeds, then launch the **Refresh Fact Cache** job template. Create the second node using the following information:

Workflow Node #2

Field	Value
Node Type	Job Template
Name	Refresh Fact Cache
Node Alias	Refresh Fact Cache (Test)
Inventory	Test

Chapter 6 | Constructing Job Workflows

7. If the **Refresh Fact Cache (Test)** node succeeds, then launch the **Webservers setup** job template. Create the third node using the following information:

Workflow Node #3

Field	Value
Node Type	Job Template
Name	Webservers setup
Node Alias	Webservers setup (Test)
Inventory	Test

8. If the **Webservers setup (Test)** node succeeds, then create an approval node using the following information:

Workflow Node #4

Field	Value
Node Type	Approval
Name	Prod Approval
Description	Pushing to Prod requires approval
Timeout (min)	60

9. If the **Prod Approval** node succeeds, then launch the **Refresh Fact Cache** job template. Create the fifth node using the following information:

Workflow Node #5

Field	Value
Node Type	Job Template
Name	Refresh Fact Cache
Node Alias	Refresh Fact Cache (Prod)
Inventory	Prod

Chapter 6 | Constructing Job Workflows

- 10.** If the Refresh Fact Cache (Prod) node succeeds, then launch the Webservers setup job template. Create the sixth node using the following information. This is the final node in the From Test to Prod workflow job template.

Workflow Node #6

Field	Value
Node Type	Job Template
Name	Webservers setup
Node Alias	Webservers setup (Prod)
Inventory	Prod

- 11.** Add a survey to the From Test to Prod workflow job template containing the following information. Make sure you enable the survey.

Workflow Job Template Survey

Field	Value
Question	What version are you deploying?
Description	This version number is displayed at the bottom of the index page.
Answer variable name	deployment_version
Answer type	Text
Required	(selected)
Minimum length	1
Maximum length	40
Default answer	v1.0

- 12.** Activate the Approval, Success, and Failure notifications for the From Test to Prod workflow job template using the existing Email AAP Admins notification template.
- 13.** Launch the From Test to Prod workflow job template. When prompted by the survey, enter v1.3 for the deployment version.
- 14.** Approve the execution of the remaining workflow job template nodes and verify that the workflow triggers an email notification for the approval, when approved, and after completion.
- The Email AAP Admins notification template sends emails to the aap-admins@lab.example.com email alias. The student user can access these emails on the utility server.
- 15.** Verify that the web servers in the Test and Prod inventories (serverc.lab.example.com, serverd.lab.example.com, servere.lab.example.com, and serverf.lab.example.com) display the Deployment Version: v1.3 line.

Evaluation

On the **workstation** machine, change to the **student** user home directory and use the **lab** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade workflow-review
```

Finish

On the **workstation** machine, change to the **student** user home directory and use the **lab** command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish workflow-review
```

► Solution

Constructing Job Workflows

Create a new workflow job template that uses a survey to set variables, and fact caching to speed up the workflow.

Outcomes

- Create a workflow job template.
- Create an associated survey and notification template.
- Launch the workflow job template from the automation controller web UI.

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that automation controller is installed and configured with any necessary resources created in previous exercises.

```
[student@workstation ~]$ lab start workflow-review
```

Instructions

Create a workflow job template that deploys your web servers to the test environment, and after approval, to production. As part of this process, rename the existing `TEST webservers setup` job template to `Webservers setup` and configure the job template to prompt for an inventory. This allows you to use a single job template for both the test and production environments. The `Webservers setup` job template needs to use facts cached from earlier in the workflow. The workflow job template also needs to prompt the user with a survey to set some variable values when the workflow job template is launched.

1. Go to `https://controller.lab.example.com` and log in as the `admin` user with `redhat` as the password.
2. Make sure that the per-host Ansible fact cache time-out for automation controller is set for exactly one day (86400 seconds). This ensures that facts are cached for hosts but also expire if those facts have not been refreshed recently.
 - 2.1. Click **Settings** in the left navigation bar.
 - 2.2. In the **Settings** window, on the **Jobs** tile, click **Jobs settings**.
 - 2.3. Confirm that **Per-Host Ansible Fact Cache Timeout** is set to 86400 seconds. If the setting is different, click **Edit**, edit that field to contain the value 86400, and click **Save**.
3. Update the `TEST webservers setup` job template with the following information, leaving the other fields untouched. Make sure to clear the existing inventory entry for the job template.

Chapter 6 | Constructing Job Workflows

Field	Value
Name	Webservers setup
Description	Setup apache webservers
Inventory	Prompt on launch
Options	Enable Fact Storage

- 3.1. Go to Resources > Templates and click the Edit Template icon for the TEST webservers setup job template.
- 3.2. Rename to Webservers setup and change the description to Setup apache webservers.
- 3.3. Clear the inventory selection and select Prompt on launch for the inventory.
- 3.4. Scroll to the bottom of the page, select Enable Fact Storage in the Options section, and then click Save. Do not change any of the other fields.
4. Create a workflow job template called From Test to Prod with the following information. In the following steps, you create nodes for the workflow job template.

Workflow Template

Field	Value
Name	From Test to Prod
Description	Deploy to Test and on success deploy to Prod
Organization	Default

- 4.1. Go to Resources > Templates and click Add > Add workflow template.
- 4.2. Create a workflow template using information from the Workflow Template table. When finished, click Save.
5. Start the workflow for the From Test to Prod workflow job template by synchronizing the My Webservers project. Create the first node using the following information:

Workflow Node #1

Field	Value
Node Type	Project Sync
Name	My Webservers

- 5.1. Using the workflow visualizer for the From Test to Prod workflow job template, click Start. Create the first node using information from the Workflow Node #1 table and then click Save.

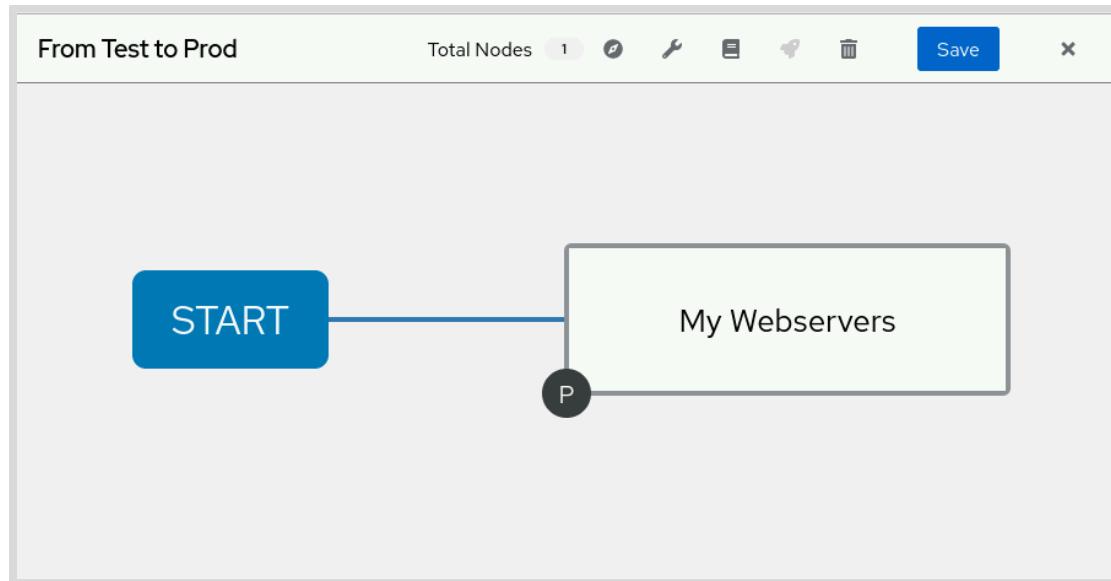


Figure 6.22: Synchronizing a project as the first step in the workflow



Note

The P symbol under the **My Webservers** node in the workflow visualizer indicates it is a project sync node.

The blue line connecting **Start** and the **My Webservers** node indicates that this step is always performed.

6. If the **My Webservers** node succeeds, then launch the **Refresh Fact Cache** job template. Create the second node using the following information:

Workflow Node #2

Field	Value
Node Type	Job Template
Name	Refresh Fact Cache
Node Alias	Refresh Fact Cache (Test)
Inventory	Test

- 6.1. Using the workflow visualizer for the **From Test to Prod** workflow job template, hover over the **My Webservers** node and then click **Add a new node**.
- 6.2. Select **On Success** and click **Next**.
- 6.3. Create the second node using information from the **Workflow Node #2** table and then click **Next**.
- 6.4. Select the **Test** inventory and click **Next**. Click **Save**.

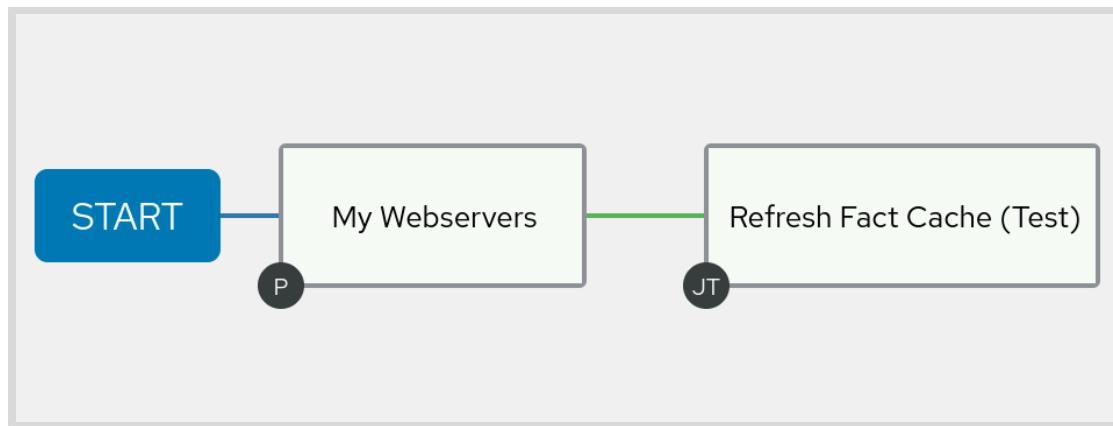


Figure 6.23: Adding a job template node to the workflow



Note

The JT symbol under the Refresh Fact Cache (Test) node in the workflow visualizer indicates it is a job template node.

The green line connecting the My Webservers node and the Refresh Fact Cache (Test) node indicates that this step is performed only on success.

7. If the Refresh Fact Cache (Test) node succeeds, then launch the Webservers setup job template. Create the third node using the following information:

Workflow Node #3

Field	Value
Node Type	Job Template
Name	Webservers setup
Node Alias	Webservers setup (Test)
Inventory	Test

- 7.1. Using the workflow visualizer for the From Test to Prod workflow job template, hover over the Refresh Fact Cache (Test) node and then click Add a new node.
- 7.2. Select On Success and click Next.
- 7.3. Create the third node using information from the Workflow Node #3 table and then click Next.
- 7.4. Select the Test inventory and click Next. Click Save.
8. If the Webservers setup (Test) node succeeds, then create an approval node using the following information:

Workflow Node #4

Field	Value
Node Type	Approval
Name	Prod Approval
Description	Pushing to Prod requires approval
Timeout (min)	60

- 8.1. Using the workflow visualizer for the **From Test to Prod** workflow job template, hover over the **Webservers setup (Test)** node and then click **Add a new node**.
- 8.2. Select **On Success** and click **Next**.
- 8.3. Create the fourth node using information from the **Workflow Node #4** table and then click **Save**.

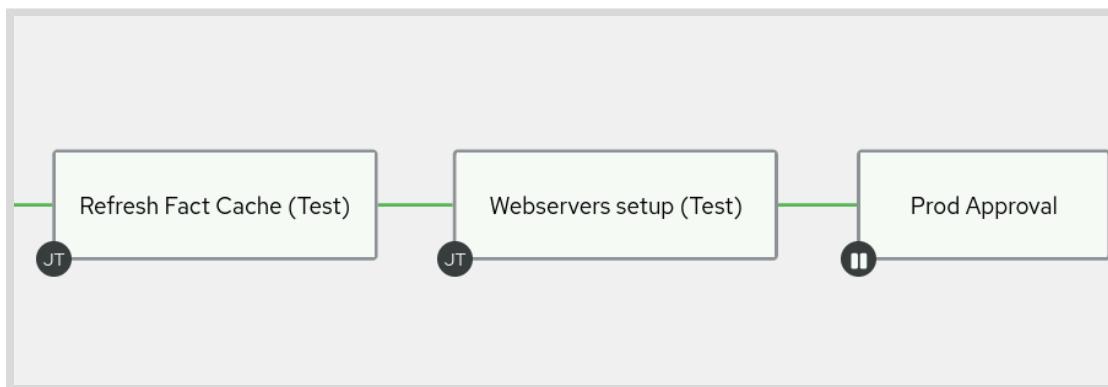


Figure 6.24: Requesting an approval in the workflow

**Note**

The pause symbol (||) under the **Prod Approval** node in the workflow visualizer indicates it is an approval node.

After a successful launch of job templates in the test environment, the workflow indicates requesting an approval to launch them in production.

9. If the **Prod Approval** node succeeds, then launch the **Refresh Fact Cache** job template. Create the fifth node using the following information:

Workflow Node #5

Field	Value
Node Type	Job Template
Name	Refresh Fact Cache
Node Alias	Refresh Fact Cache (Prod)
Inventory	Prod

- 9.1. Using the workflow visualizer for the **From Test to Prod** workflow job template, hover over the **Prod Approval** node and then click **Add a new node**.
- 9.2. Select **On Success** and click **Next**.
- 9.3. Create the fifth node using information from the **Workflow Node #5** table and then click **Next**.
- 9.4. Select the **Prod** inventory and click **Next**. Click **Save**.
10. If the **Refresh Fact Cache (Prod)** node succeeds, then launch the **Webservers setup** job template. Create the sixth node using the following information. This is the final node in the **From Test to Prod** workflow job template.

Workflow Node #6

Field	Value
Node Type	Job Template
Name	Webservers setup
Node Alias	Webservers setup (Prod)
Inventory	Prod

- 10.1. Using the workflow visualizer for the **From Test to Prod** workflow job template, hover over the **Refresh Fact Cache (Prod)** node and then click **Add a new node**.
- 10.2. Select **On Success** and click **Next**.
- 10.3. Create the sixth node using information from the **Workflow Node #6** table and then click **Next**.
- 10.4. Select the **Prod** inventory and click **Next**. Click **Save**.
- 10.5. Click **Save** to save the workflow and exit the workflow visualizer.
11. Add a survey to the **From Test to Prod** workflow job template containing the following information. Make sure you enable the survey.

Workflow Job Template Survey

Field	Value
Question	What version are you deploying?
Description	This version number is displayed at the bottom of the index page.
Answer variable name	deployment_version
Answer type	Text
Required	(selected)
Minimum length	1
Maximum length	40
Default answer	v1.0

- 11.1. Go to **Resources > Templates** and click the **From Test to Prod** link.
- 11.2. Click the **Survey** tab and then click **Add**.
- 11.3. Add a single question to the survey, using the information in the **Workflow Job Template Survey** table. When finished, click **Save**.
- 11.4. Enable the survey by setting the **Survey Disabled|Survey Enabled** switch to on.

**Important**

The survey is disabled by default. Make sure that you enable it.

12. Activate the **Approval**, **Success**, and **Failure** notifications for the **From Test to Prod** workflow job template using the existing **Email AAP Admins** notification template.
 - 12.1. Go to **Resources > Templates** and click the **From Test to Prod** link.
 - 12.2. Click the **Notifications** tab and then select the **Approval**, **Success**, and **Failure** options for the **Email AAP Admins** notification template.
13. Launch the **From Test to Prod** workflow job template. When prompted by the survey, enter **v1.3** for the deployment version.
 - 13.1. Go to **Resources > Templates** and then click the **Launch Template** icon for the **From Test to Prod** workflow job template.
 - 13.2. Enter **v1.3** in the text field for the survey that you just created, and then click **Next**.
 - 13.3. Click **Launch** to launch the workflow. Observe the running jobs of the workflow.
 - 13.4. The workflow stops on the approval node. Notice the notification icon at the top of the page, indicating one pending workflow approval.

Chapter 6 | Constructing Job Workflows

14. Approve the execution of the remaining workflow job template nodes and verify that the workflow triggers an email notification for the approval, when approved, and after completion.

The `Email AAP Admins` notification template sends emails to the `aap-admins@lab.example.com` email alias. The `student` user can access these emails on the `utility` server.

- 14.1. Open a terminal and connect to the `utility` server.

```
[student@workstation ~]$ ssh student@utility
```

- 14.2. Use the `tail` command to view incoming messages to the local mailbox file of the `student` user. You should see the email that was sent for the approval. Your message looks similar to the following:

```
[student@utility ~]$ tail -f /var/mail/student
"id": 10,
"name": "Prod Approval",
"url": "https://controller.lab.example.com/#/jobs/workflow/6",
"created_by": "admin",
"started": "2022-05-13T21:53:29.886316+00:00",
"finished": null,
"status": "pending",
"traceback": ""
}
```

- 14.3. Click the notification icon in the automation controller web UI, select the checkbox for the `Prod Approval` workflow approval, and then click **Approve**.

- 14.4. Go to the terminal window running the `tail` command and wait a few seconds. You should see an approval notification email arrive that looks similar to the following example:

```
From system@controller.lab.example.com Fri May 13 17:57:15 2022
Return-Path: <system@controller.lab.example.com>
X-Original-To: aap-admins@lab.example.com
Delivered-To: aap-admins@lab.example.com
Received: from controller.lab.example.com (controller.lab.example.com
[172.25.250.7])
by utility.lab.example.com (Postfix) with ESMTPS id E81EA938CEC
for <aap-admins@lab.example.com>; Fri, 13 May 2022 17:57:13 -0400 (EDT)
Received: from controller.lab.example.com (localhost [IPv6:::1])
by controller.lab.example.com (Postfix) with ESMTP id CF7DC25D127
for <aap-admins@lab.example.com>; Fri, 13 May 2022 17:57:13 -0400 (EDT)
Content-Type: text/plain; charset="utf-8"
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit
Subject: The approval node "Prod Approval" was approved.
https://controller.lab.example.com/#/jobs/workflow/6
From: system@controller.lab.example.com
To: aap-admins@lab.example.com
Date: Fri, 13 May 2022 21:57:13 -0000
Message-ID:
<165247903382.5344.15434941766628918585@controller.lab.example.com>
```

Chapter 6 | Constructing Job Workflows

The approval node "Prod Approval" was approved. <https://controller.lab.example.com/#/jobs/workflow/6>

```
{  
  "id": 10,  
  "name": "Prod Approval",  
  "url": "https://controller.lab.example.com/#/jobs/workflow/6",  
  "created_by": "admin",  
  "started": "2022-05-13T21:53:29.886316+00:00",  
  "finished": "2022-05-13T21:57:13.356070+00:00",  
  "status": "successful",  
  "traceback": ""  
}
```

- 14.5. Go to **Views > Jobs** in the automation controller web UI and wait for the **6 - From Test to Prod** workflow job to finish.

The screenshot shows the 'Jobs' page in the Automation Controller web interface. At the top, there is a search bar and a 'Delete' button. Below the search bar, there are filters for 'Name', 'Status', 'Type', 'Start Time', 'Finish Time', and 'Actions'. A table below the filters lists a single job entry: '6 – From Test to Prod' with a status of 'Successful'. The job is a 'Workflow Job' that started on 5/13/2022, 5:52:53 PM and finished on 5/13/2022, 5:57:46 PM. There is also a small icon next to the job entry.

Figure 6.25: The completed workflow job template

- 14.6. Go to the terminal window running the `tail` command. You should see a new notification email indicating the successful execution of the workflow. The notification looks similar to the following example:

```
From system@controller.lab.example.com Fri May 13 17:57:47 2022  
Return-Path: <system@controller.lab.example.com>  
X-Original-To: aap-admins@lab.example.com  
Delivered-To: aap-admins@lab.example.com  
Received: from controller.lab.example.com (controller.lab.example.com  
[172.25.250.7])  
by utility.lab.example.com (Postfix) with ESMTPS id 53C0C938CEC  
for <aap-admins@lab.example.com>; Fri, 13 May 2022 17:57:47 -0400 (EDT)  
Received: from controller.lab.example.com (localhost [IPv6:::1])  
by controller.lab.example.com (Postfix) with ESMTP id 4D82325D127  
for <aap-admins@lab.example.com>; Fri, 13 May 2022 17:57:47 -0400 (EDT)  
Content-Type: text/plain; charset="utf-8"  
MIME-Version: 1.0  
Content-Transfer-Encoding: 7bit  
Subject: Workflow Job #6 'From Test to Prod' successful:  
https://controller.lab.example.com/#/jobs/workflow/6  
From: system@controller.lab.example.com  
To: aap-admins@lab.example.com  
Date: Fri, 13 May 2022 21:57:47 -0000
```

Chapter 6 | Constructing Job Workflows

```
Message-ID:  
<165247906731.5344.18226747658229373298@controller.lab.example.com>  
  
Workflow Job #6 had status successful, view details at https://  
controller.lab.example.com/#/jobs/workflow/6  
  
{  
    "id": 6,  
    "name": "From Test to Prod",  
    "url": "https://controller.lab.example.com/#/jobs/workflow/6",  
    "created_by": "admin",  
    "started": "2022-05-13T21:52:53.307159+00:00",  
    "finished": "2022-05-13T21:57:46.714992+00:00",  
    "status": "successful",  
    "traceback": "",  
    "body": "Workflow job summary:\n\n- node #1 spawns job #7, \"My Webservers\", which finished with status successful.\n- node #2 spawns job #8, \"Refresh Fact Cache\", which finished with status successful.\n- node #3 spawns job #9, \"Webservers setup\", which finished with status successful.\n- node #6 spawns job #10, \"Prod Approval\", which finished with status successful.\n- node #5 spawns job #11, \"Refresh Fact Cache\", which finished with status successful.\n- node #4 spawns job #12, \"Webservers setup\", which finished with status successful."  
}
```

When finished, press **Ctrl+C** to exit the `tail` command.

**Note**

If you prefer, you can use the `mailx` or `mutt` commands to display mail messages on the utility server. The subject of the emails received includes the following text:

- The approval node "Prod Approval" needs review.
- The approval node "Prod Approval" was approved.
- Workflow Job #6 'From Test to Prod' successful.

14.7. Exit the terminal session on the utility system.

```
[student@utility ~]$ exit
```

- 15.** Verify that the web servers in the Test and Prod inventories (`serverc.lab.example.com`, `serverd.lab.example.com`, `servere.lab.example.com`, and `serverf.lab.example.com`) display the Deployment Version: v1.3 line.

- 15.1. Open a web browser and go to `http://serverc.lab.example.com`, `http://serverd.lab.example.com`, `http://servere.lab.example.com`, and `http://serverf.lab.example.com` in separate tabs. You should see this line at the bottom of each page:

```
Deployment Version: v1.3
```

- 15.2. When ready, log out from the automation controller web UI.

Evaluation

On the **workstation** machine, change to the **student** user home directory and use the **lab** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade workflow-review
```

Finish

On the **workstation** machine, change to the **student** user home directory and use the **lab** command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish workflow-review
```

Summary

- Create workflow job templates in automation controller to automate complex operations including synchronizing projects, synchronizing inventories, and launching multiple job templates.
- Use the workflow visualizer to design the workflow of a workflow job template including deciding if a node should always run, if a node should run if the previous node succeeded, or if a node should run if the previous node failed.
- Add approval nodes to workflow job templates when you require approval or manual intervention before proceeding with a workflow.

Chapter 7

Managing Advanced Inventories

Goal

Manage inventories that are generated dynamically from scripts or the automation controller smart inventory feature.

Sections

- Importing External Static Inventories (and Guided Exercise)
- Configuring Dynamic Inventory Plug-ins (and Quiz)
- Filtering Hosts with Smart Inventories (and Guided Exercise)
- Managing Advanced Inventories

Lab

Importing External Static Inventories

Objectives

- Import existing static inventory files managed in a Git repository into automation controller.

Importing Existing Static Inventories

If you are adding an existing Ansible project that you have been managing from a traditional control node into automation controller, you might already have a large static inventory file for that project. It can be inconvenient to add that static inventory manually through the web UI. You might also want to continue to manage that static inventory outside of automation controller rather than through the web UI. A number of ways exist to handle these situations.

One of the most effective solutions is to configure automation controller to retrieve a static inventory file from an existing automation controller project. This approach enables you to manage changes to the inventory (and inventory variables) just like changes to other files under version control. Although possible, you should avoid using the web UI to make changes to this type of inventory because automation does not push changes back to your version control system. When you make a change to your source control repository that affects the inventory, you need to synchronize both the associated project and inventory before automation controller recognizes the changes.

Storing an Inventory in a Project

Automation controller can use inventory files that you manage in a source code management (SCM) repository. You can continue to store your inventory in a Git repository, and use commits, pull requests, and other features of Git or your repository server to manage updates. You can also use any other type of SCM that is supported by projects in automation controller.

To configure this functionality, start by setting up a project that points to your Git repository. This is done in exactly the same way as when you are setting up a project to be part of a job template. Specify the **Source Control Type** (such as Git) and **Source Control URL** of the repository containing the inventory file or files. You also need to specify the **Source Control Credential** that contains the authentication information for that repository. You can specify a particular branch, tag, or commit to use, and whether the contents of the project should update the revision on launch like any other project.

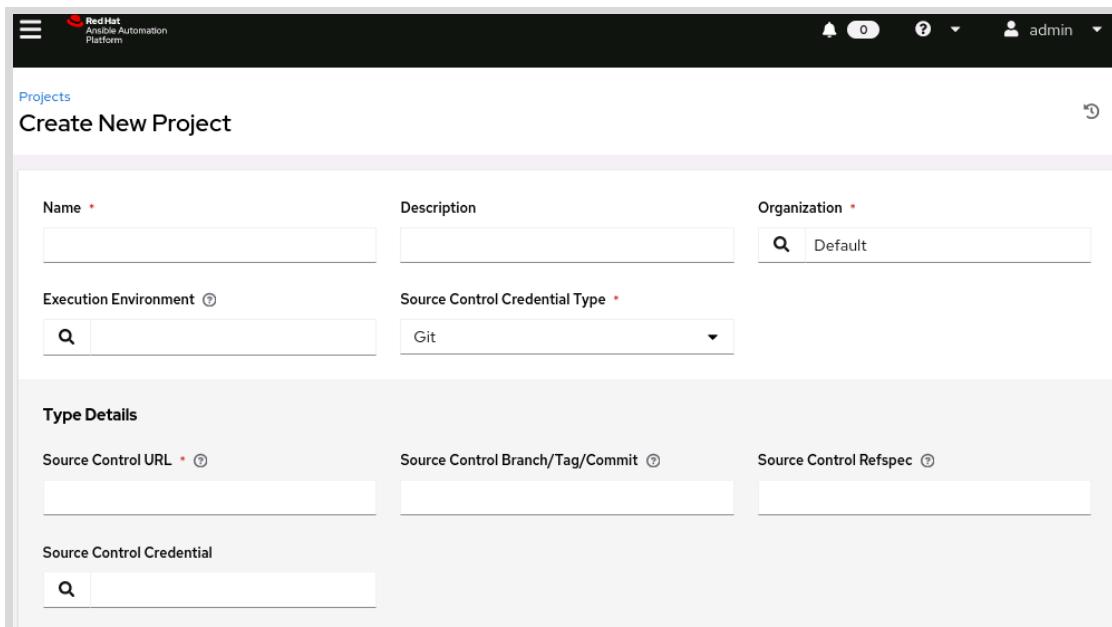


Figure 7.1: Configuring Git-based projects

After you have set up the project, you can configure the inventory in automation controller. Create the inventory normally and open it for editing. Click **Sources**, and then click **Add** to add a new source.

On the **Create new source** page, give the source a name and then select a source. Selecting the **Source > Sourced from a Project** source exposes extra fields on the page. Specify the **Project** that contains your inventory. Select the inventory file from the **Inventory file** list, or enter the file name if it does not appear.

You can enter an SCM credential in the **Credential** field if one is needed to access the project's Git repository. You can also select the **Update on project update** checkbox to refresh the inventory after every project update that involves a Git revision update. Finally, you can save the changes and synchronize the inventory with the project.

The screenshot shows the 'Create new source' page in the Red Hat Ansible Automation Platform. The top navigation bar includes the Red Hat logo and the platform name. The main header is 'Inventories > git-inventory > Sources'. Below the header, the title 'Create new source' is displayed. The form fields include:

- Name ***: A text input field.
- Description**: A text input field.
- Execution Environment**: A search input field with a magnifying glass icon.
- Source ***: A dropdown menu set to 'Sourced from a Project'.

Below the source selection, there is a section titled 'Source details' containing the following fields:

- Credential**: A search input field with a magnifying glass icon.
- Project ***: A search input field with a magnifying glass icon, currently set to 'Demo Project'.
- Inventory file * ⓘ**: A dropdown menu.
- Verbosity ⓘ**: A dropdown menu set to '1 (Info)'.
- Host Filter ⓘ**: An empty input field.
- Enabled Variable ⓘ**: An empty input field.

Figure 7.2: Inventory sourced from a project



References

For more information about adding inventories to automation controller, refer to the *Automation Controller User Guide* at
<https://docs.ansible.com/automation-controller/latest/html/userguide/inventories.html#add-a-new-inventory>

► Guided Exercise

Importing External Static Inventories

Use an existing static inventory stored in a Git repository in automation controller.

Outcomes

- Use an existing static inventory file stored in a Git repository.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command ensures that automation controller is installed and configured with any necessary resources created in previous exercises.

```
[student@workstation ~]$ lab start advinventory-static
```

Instructions

- 1. Use an existing static inventory file stored in a Git repository.
- 1.1. Navigate to <https://controller.lab.example.com> and log in as the **admin** user with **redhat** as the password.
 - 1.2. Create a new inventory. Navigate to **Resources > Inventories** and then click **Add > Add inventory**.
 - 1.3. Use **Git Inventory** as the name, use **Inventory with Git source** as the description, and then click **Save**.
 - 1.4. On the **Details** page, click the **Sources** tab and then click **Add** to add a source.
 - 1.5. Fill in the details as follows and then click **Save**.

Field	Value
Name	Git Source
Description	Git source for inventory
Source	Sourced from a Project
Project	Advanced Inventory
Inventory file	inventory
Update options	Update on launch

Important

You might see the following message below the **Project** field even if you selected the project name from the list of projects:

That value was not found. Please enter or select a valid value.

The message disappears as soon as you select a different field.

- 1.6. Click **Sync** and wait until **Last Job Status** displays **Successful**.
- 1.7. Navigate to **Resources > Inventories** and click **Git Inventory**.
- 1.8. Review the **Hosts** tab and the **Groups** tab and notice the hosts and groups that were added to the inventory.
- 1.9. View the details of the **web_servers** host group and notice that the variables are also imported.

The screenshot shows the 'Group details' page for the 'web_servers' host group. The URL in the browser is 'Inventories > Git Inventory > Groups > web_servers'. The page has tabs for 'Back to Groups', 'Details', 'Related Groups', and 'Hosts'. The 'Details' tab is selected. The 'Name' is 'web_servers' and 'Description' is 'imported'. Under 'Variables', there are two tabs: 'YAML' (selected) and 'JSON'. The JSON code shown is:
1. {
2. "packages": [
3. "httpd",
4. "mod_ssl"
]
The 'Created' date is '8/12/2024, 9:50:32 AM' and 'Last Modified' date is '8/12/2024, 9:50:32 AM'. At the bottom are 'Edit' and 'Delete' buttons.

**Note**

Although the variables are displayed in JSON format here, the web UI can also display variables in YAML format.

- 2. Modify an existing job template to use the new inventory.
- 2.1. Navigate to **Resources > Templates**.
 - 2.2. Click the **Edit Template** icon for the **Hello World** job template.
 - 2.3. On the **Edit Details** page, enter **Git Inventory** in the **Inventory** field.

Chapter 7 | Managing Advanced Inventories

Click **Save** to save the changes.

- 2.4. On the **Details** page, click **Launch**.
- 2.5. Review the **Output** page. Notice that the job runs on three hosts: `servera.lab.example.com`, `serverb.lab.example.com`, and `serverc.lab.example.com`.

- 3. Clone the `https://git.lab.example.com/git/advanced-inventory.git` Git repository into the `/home/student/git-repos` directory.

- 3.1. From a terminal, create the `/home/student/git-repos` directory if it does not already exist, and then change into it.

```
[student@workstation ~]$ mkdir -p ~/git-repos/  
[student@workstation ~]$ cd ~/git-repos/
```

- 3.2. Clone the `advanced-inventory` git repository and then change into the cloned repository:

```
[student@workstation git-repos]$ git clone \  
> https://git.lab.example.com/git/advanced-inventory.git  
Cloning into 'advanced-inventory'...  
...output omitted...  
[student@workstation git-repos]$ cd advanced-inventory
```

- 3.3. Review the files in the project directory using the `tree` command:

```
[student@workstation advanced-inventory]$ tree  
.---- ansible.cfg  
---- group_vars  
|---- web_servers  
|---- packages.yml  
---- hello_world.yml  
---- inventory  
  
2 directories, 4 files
```

- 3.4. Add `serverd.lab.example.com` to the `[lb_servers]` section of the `inventory` file. The first three lines of the file should consist of the following content:

```
[student@workstation advanced-inventory]$ head -n 3 inventory  
[lb_servers]  
servera.lab.example.com  
serverd.lab.example.com
```

- 3.5. Add the changes to the index, commit the changes using `Updated inventory` as the commit message, and then push the changes to the remote repository.

```
[student@workstation advanced-inventory]$ git add inventory
[student@workstation advanced-inventory]$ git commit -m "Updated inventory"
[main e76b744] Updated inventory
 1 file changed, 1 insertion(+)
[student@workstation advanced-inventory]$ git push -u origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
...output omitted...
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

► **4.** Verify that the changes are reflected in the inventory.

- 4.1. From the automation controller web UI, navigate to **Resources > Projects**.
- 4.2. Click the **Sync Project** icon for the **Advanced Inventory** project. Wait until the synchronization is complete.
- 4.3. Navigate to **Resources > Templates** and click the **Launch Template** icon for the **Hello World** job template.
- 4.4. After the job is finished, review the **Output** page. Notice that the job runs on four hosts: `servera.lab.example.com`, `serverb.lab.example.com`, `serverc.lab.example.com`, and `serverd.lab.example.com`.
- 4.5. Navigate to **Resources > Inventories** and click **Git Inventory**. Review the **Hosts** tab and notice that `serverd.lab.example.com` is now present in the inventory.

Finish

On the **workstation** machine, change to the **student** user home directory and use the **lab** command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish advinventory-static
```

Configuring Dynamic Inventory Plug-ins

Objectives

- Create a dynamic inventory that uses an inventory plug-in to set hosts and host groups.

Dynamic Inventories

When Ansible runs a playbook, it uses an inventory to help determine the managed hosts on which the plays should be run. Administrators explicitly specify hosts in static inventories that both Ansible and automation controller can use.

However, static inventories require manual administration to keep the inventories up-to-date. This can be inconvenient or challenging, especially when an organization wants to run playbooks against hosts that are dynamically created in a virtualization or cloud computing environment.

Dynamic inventories determine which hosts and host groups should be in the inventory based on information from some external source. Large IT environments that frequently deploy, test, and remove systems can benefit from using dynamic inventories.

By default, automation controller includes built-in dynamic inventory support for various external inventory sources (or cloud inventory sources), including:

- Amazon EC2
- Google Compute Engine
- Microsoft Azure Resource Manager
- VMware vCenter
- Red Hat Satellite 6
- OpenStack
- Red Hat Virtualization
- Red Hat Ansible Automation Platform
- Red Hat Insights

In addition, you can use inventory plug-ins that are provided by installed Ansible Content Collections. The following command displays the inventory plug-ins that are available from the `ee-supported-rhel8` automation execution environment.

```
[user@host ~]$ ansible-navigator doc --list -t inventory \
> --eei ee-supported-rhel8 --pp never --mode stdout
[WARNING]: Error parsing collection metadata requires_ansible value from coll...
[WARNING]: Error parsing collection metadata requires_ansible value from coll...
[WARNING]: Collection redhat.insights does not support Ansible version 2.13.0
advanced_host_list          Parses a 'host list' with ranges
amazon.aws.aws_ec2          EC2 inventory source
amazon.aws.aws_rds           rds instance source
ansible.controller.controller Ansible dynamic inventory plug-in for the Automa...
auto                         Loads and executes an inventory plug-in specific...
constructed                  Uses Jinja2 to construct vars and groups based ...
generator                   Uses Jinja2 to construct hosts and groups from ...
host_list                   Parses a 'host list' string
ini                          Uses an Ansible INI file as inventory source
```

Chapter 7 | Managing Advanced Inventories

kubernetes.core.k8s	Kubernetes (K8s) inventory source
redhat.insights.insights	insights inventory source
redhat.openshift.openshift	OpenShift inventory source
redhat.rhv.ovirt	oVirt inventory source
redhat.satellite.foreman	Foreman inventory source
script	Executes an inventory script that returns JSON
servicenow.itsm.now	Inventory source for Service Now table records
toml	Uses a specific TOML file as an inventory source
yaml	Uses a specific YAML file as an inventory source

Some of these, such as `yaml` and `ini`, are the plug-ins that provide static inventory support. Others, such as `redhat.rhv.ovirt` and `amazon.aws.aws_ec2`, construct the inventory dynamically from a separate source of information.

The remainder of this section examines two dynamic inventory plug-ins and how to configure them in automation controller. The first example explores the built-in support for getting information from OpenStack.

The second example briefly examines the built-in support for getting inventory information from a Red Hat Satellite 6 server.

OpenStack Dynamic Inventories

Cloud technologies such as OpenStack bring many changes to the server lifecycle. Managed hosts come and go much more frequently and they can be created and started by external applications. Maintaining an accurate static inventory file can be challenging over any length of time, and is even more difficult when managed hosts must be added and removed frequently with little or no warning. Having an inventory that updates itself dynamically, based on information provided directly from OpenStack, solves this problem.

The process for configuring dynamic inventories, using any of the built-in cloud sources, follows the same basic steps:

1. Create a credential to authenticate to the cloud data source you intend to use, using a credential type matching the source.
2. Create a new inventory to provide dynamic inventory information.
3. In that new inventory, create a source with one of the built-in dynamic inventory sources. Configure the new inventory source to use the associated credential for the cloud service. You might set other options, such as enabling an automatic update of the inventory from its source whenever the inventory is used (using the `Update on launch` feature).
4. Perform an initial synchronization of the inventory from the source to populate the list of inventory hosts and host groups in automation controller.

The screenshot shows a 'Create New Credential' form. At the top left is a 'Credentials' link. Below it, the title 'Create New Credential' is displayed. The main area has two columns: 'Name *' and 'Description' (empty), and 'Organization' (with a search icon) and 'Credential Type *' (set to 'OpenStack'). Below this is a section titled 'Type Details' containing six fields arranged in a 3x2 grid: 'Username *' and 'Password (API Key) *' (both with key icons); 'Host (Authentication URL) *' and 'Project (Tenant Name) *' (both with key icons); and 'Project (Domain Name)' and 'Domain Name' (both with key icons). A small circular icon with a question mark is located next to the 'Host' field.

Figure 7.4: Creating a new OpenStack credential

The fields displayed in the **Type Details** section change based on the selected credential type. Mandatory fields are marked with a red asterisk.

Field	Description
Username	The user who can access the required resources.
Password (API Key)	The password or API key for the specified user.
Host (Authentication URL)	The authentication URL of the OpenStack host to authenticate with, for example <code>https://demo.lab.example.com/v2.0/</code> .
Project (Tenant Name)	The name of the project (tenant) that you want to use.

After creating an OpenStack credential, you can use the credential with a dynamic OpenStack source. You can either create a new inventory and add the OpenStack source to it, or you can add the OpenStack source to an existing inventory.

The screenshot shows the 'Create new source' page in the Red Hat Ansible Automation Platform. At the top, there's a breadcrumb navigation: Inventories > OpenStack > Sources. Below the title 'Create new source' is a 'Name *' field with a placeholder 'Source Name'. To its right is a 'Description' field. Under 'Execution Environment', there's a search bar with a magnifying glass icon and a dropdown menu set to 'OpenStack'. The main area is titled 'Source details' and contains several groups of fields: 'Credential *' with a search bar and dropdown, 'Verbosity' set to '1 (Info)', 'Host Filter' with a search bar, 'Enabled Variable' with a search bar, and 'Enabled Value' with a search bar.

Figure 7.5: Creating an OpenStack inventory source

Dynamic sources provide the following update options:

Overwrite

The inventory update process deletes all child groups and hosts from the local inventory that are not found on the external source. By default, this option is not selected, which means that all child hosts and groups that are not found on the external source remain untouched.

Overwrite Variables

All inventory variables that are not found on the external source are removed. By default, this option is not selected, which means that an update of this dynamic inventory combines local variables with those found on the external source.

Update on Launch

Automation controller launches a separate job to update the inventory from the external source each time a job runs that uses the inventory. The inventory update job runs before automation controller runs the automation job initiated by the job template.

The source list for an inventory displays all sources for the inventory. The status column displays the results of the last synchronization attempt for each inventory. Hover over an individual status icon to see brief information about the synchronization status, or click the status icon to see the entire output of the inventory synchronization job.

Click the **Start sync process** icon to manually synchronize an inventory source.

After a successful synchronization, review the child groups and hosts that have been created in automation controller using the information from the external source. A dynamic inventory might update every time you synchronize it with the external source.

Chapter 7 | Managing Advanced Inventories

- To see all hosts in an inventory, navigate to **Resources > Inventories**, click the link for the inventory, and then click the **Hosts** tab.
- To see all groups in an inventory, navigate to **Resources > Inventories**, click the link for the inventory, and then click the **Groups** tab. To see the hosts that belong to a group, click the link for the group and then click the **Hosts** tab.

You can synchronize an inventory in the following ways:

- Click the **Start sync process** icon to manually synchronize an inventory source.
- Schedule a job to synchronize an inventory source. The scheduled job can synchronize at a recurring interval.
- Configure the inventory source to automatically update when a job template uses the inventory by selecting the **Update on Launch** checkbox.
- Add an **Inventory Source Sync** node to a workflow job template.

Red Hat Satellite 6 Dynamic Inventories

You can use Red Hat Satellite 6 with automation controller to deploy and configure new bare-metal servers. By using a dynamic inventory that gets its information from a Red Hat Satellite 6 server, you can run playbooks on hosts that have registered with the Satellite server.

The following workflow could be used:

1. The new managed host uses some combination of PXE, DHCP, and TFTP to boot from the network, or a boot image to boot locally, to prepare for either an unattended installation from the Satellite server or an installation through the Satellite Discovery service.
2. The new managed host performs a Kickstart installation from materials provided by the Satellite server. As part of the process, the managed host registers itself to the Satellite server.
3. After it registers, the new managed host appears in the dynamic inventory generated from information stored in the Red Hat Satellite 6 server.
4. Automation controller can launch jobs that use that dynamic inventory to ensure that the new managed host is provisioned correctly.



Note

Automation controller also has a *provisioning callbacks* feature that can trigger initial provisioning jobs for new managed hosts. More information about this feature is available at https://docs.ansible.com/automation-controller/latest/html/userguide/job_templates.html#provisioning-callbacks.

The procedure to create a dynamic inventory that uses Red Hat Satellite 6 is similar to the procedure used to create one that uses OpenStack.

1. Create a new credential using the **Red Hat Satellite 6** credential type. In addition to the credential name, a Red Hat Satellite 6 credential requires you to complete the **Satellite 6 URL**, **Username**, and **Password** fields.

Field	Description
Satellite 6 URL	The URL for the Satellite server, such as <code>https://satellite.example.com</code> .
Username	The name of a Satellite user.
Password	The password for the Satellite user.

2. Create a new inventory to provide dynamic inventory information.
3. In that inventory, add a new source. Select **Red Hat Satellite 6** as the source and select the Red Hat Satellite 6 credential that you created previously. If desired, select one or more options that control how to update the inventory source. Select the **Update on Launch** checkbox if you plan to use provisioning callbacks.
4. Synchronize the Red Hat Satellite 6 inventory source. If the synchronization completes successfully, then automation controller updates the inventory with the hosts and groups provided by the external source.



References

Automation Controller User Guide, "Inventories"

<https://docs.ansible.com/automation-controller/latest/html/userguide/inventories.html>

Automation Controller User Guide, "Provisioning Callbacks"

https://docs.ansible.com/automation-controller/latest/html/userguide/job_templates.html#provisioning-callbacks

► Quiz

Configuring Dynamic Inventory Plug-ins

Choose the correct answers to the following questions:

► 1. **Which statement is true about dynamic inventories?**

- a. A dynamic inventory is a list of hosts defined in a text file in your Ansible project.
- b. A dynamic inventory is a file in a Git repository that contains the list of hosts.
- c. A dynamic inventory is an inventory that is downloaded to your automation hub by Red Hat Insights.
- d. A dynamic inventory is a list of hosts generated by a plug-in module that usually gets its information from an external source.

► 2. **Which three are benefits of using dynamic inventories? (Choose three.)**

- a. Dynamic inventories help you manage IT environments that have many short-lived systems that they frequently deploy, test, and remove.
- b. Automation controller can use multiple dynamic inventories that get information from different external inventory sources.
- c. Dynamic inventories provide anonymous access to the list of managed hosts, which helps developers perform faster deployments.
- d. A dynamic inventory can get information about managed hosts from a single central "source of truth" for your organization.

► 3. **Which three inventory sources does automation controller support? (Choose three.)**

- a. Amazon EC2
- b. Google Compute Engine
- c. Red Hat Satellite 6
- d. Virtual Machine Manager

► 4. **Which three steps are part of the procedure to configure a dynamic inventory in automation controller? (Choose three.)**

- a. Create a credential to authenticate to the data source for the inventory, if needed.
- b. Create a Source for your inventory and associate the right credential with the inventory.
- c. Add an existing list of hosts to the new dynamic inventory manually.
- d. Synchronize the inventory from the data source.

► **5. Which three methods synchronize a dynamic inventory? (Choose three.)**

- a. You can configure the inventory source to push updates to the dynamic inventory.
- b. You can manually start the inventory synchronization from automation controller.
- c. You can schedule a job that synchronizes the inventory with its source when it runs.
- d. You can configure the inventory to use the `Update on Launch` option and launch a job that uses the inventory.

► Solution

Configuring Dynamic Inventory Plug-ins

Choose the correct answers to the following questions:

► 1. **Which statement is true about dynamic inventories?**

- a. A dynamic inventory is a list of hosts defined in a text file in your Ansible project.
- b. A dynamic inventory is a file in a Git repository that contains the list of hosts.
- c. A dynamic inventory is an inventory that is downloaded to your automation hub by Red Hat Insights.
- d. A dynamic inventory is a list of hosts generated by a plug-in module that usually gets its information from an external source.

► 2. **Which three are benefits of using dynamic inventories? (Choose three.)**

- a. Dynamic inventories help you manage IT environments that have many short-lived systems that they frequently deploy, test, and remove.
- b. Automation controller can use multiple dynamic inventories that get information from different external inventory sources.
- c. Dynamic inventories provide anonymous access to the list of managed hosts, which helps developers perform faster deployments.
- d. A dynamic inventory can get information about managed hosts from a single central "source of truth" for your organization.

► 3. **Which three inventory sources does automation controller support? (Choose three.)**

- a. Amazon EC2
- b. Google Compute Engine
- c. Red Hat Satellite 6
- d. Virtual Machine Manager

► 4. **Which three steps are part of the procedure to configure a dynamic inventory in automation controller? (Choose three.)**

- a. Create a credential to authenticate to the data source for the inventory, if needed.
- b. Create a Source for your inventory and associate the right credential with the inventory.
- c. Add an existing list of hosts to the new dynamic inventory manually.
- d. Synchronize the inventory from the data source.

► **5. Which three methods synchronize a dynamic inventory? (Choose three.)**

- a. You can configure the inventory source to push updates to the dynamic inventory.
- b. You can manually start the inventory synchronization from automation controller.
- c. You can schedule a job that synchronizes the inventory with its source when it runs.
- d. You can configure the inventory to use the `Update on Launch` option and launch a job that uses the inventory.

Filtering Hosts with Smart Inventories

Objectives

- Create a smart inventory that is dynamically constructed from the other inventories on your automation controller using a filter.

Defining Smart Inventories

You have already learned how to manage static and dynamic inventories in automation controller.

- You can manually create a static inventory in the web UI.
- You can configure automation controller to use a project to get the inventory from files stored in version control and manage that inventory in the version control system.
- You can configure a dynamic inventory to get host information from an external source.

Automation controller can dynamically construct a new inventory from inventories that already exist in automation controller by using a *smart inventory*. A smart inventory generates information by applying a host filter to the union of all static and dynamic inventories configured on the automation controller.

In the most common use case, you configure the host filter to determine if an Ansible fact has a particular value for each host. Hosts that match the host filter are included in the smart inventory. By using smart inventories, you have greater flexibility in how you manage subsets of the hosts defined by the other inventories on the automation controller.

A filter for a smart inventory can select hosts by hostname, group name, or Ansible facts. To modify its filter, your user account needs to have the **Organization Administrator** role for the smart inventory's organization. If it is using Ansible facts, then you need to periodically populate the fact cache with a job template that gathers facts and that uses the **Enable Fact Storage** option to save facts.

You can do this by running a normal play that has `gather_facts: yes` enabled (this setting is normally set by default), or that runs the `setup` module as a task. An example minimal play to do this follows:

```
- name: Refresh fact cache
  hosts: all
  gather_facts: yes
```



Important

If you observe that automation controller is not caching host facts, then you might need to adjust the setting that determines for how long it caches host facts.

Navigate to **Settings** and click the **Jobs settings** link. Review the value of the **Per-Host Ansible Fact Cache Timeout** setting and adjust it if necessary.

Using Ansible Facts in Smart Inventory Filters

You can create smart inventory filters that select hosts for the inventory based on many different criteria. One of the most compelling might be to select hosts based on Ansible facts.

For example, you could configure an Ansible fact-based smart inventory to automatically consist of hosts in your other inventories that satisfy the following or other criteria:

- Have a specific CPU model.
- Have a specific processor architecture.
- Have their `eth0` interface configured for a specific subnet.
- Are running a specific Linux distribution and version.
- Are running a particular version of the Linux kernel.
- Mount particular storage devices.

This flexibility and the automatic management of the members of the inventory by automation controller can make a number of system administration tasks easier to automate.

Creating a Smart Inventory Based on Ansible Facts

To create a smart inventory, navigate to **Resources > Inventories** and click **Add > Add smart inventory**. Specify a name for the smart inventory and assign it to an organization. Click the search icon to search for the smart host filter for the smart inventory. This opens the **Perform a search to define a host filter** page.

If you want to filter hosts based on collected Ansible facts, then change the search criteria from **Name** to **Advanced** and then select **ansible_facts** from the **Key** field.

ID	Inventory
Group	Dev
Inventory ID .com	Dev
Enabled	Test
Instance ID .com	Dev
Last job .com	Test
Insights system ID .com	Test
Advanced .com	Test

Figure 7.6: Defining a smart inventory filter with an advanced search

From the **Key** list, select **ansible_facts**.

The screenshot shows a search interface for defining a host filter. At the top, a message says "Perform a search to define a host filter". Below it, a note states: "Searching by ansible_facts requires special syntax. Refer to the documentation for more info." The search bar contains the query "ansible_facts" followed by a dropdown menu and the specific filter "ansible_eth0__mtu=1500". There are also a magnifying glass icon and a question mark icon.

Figure 7.7: Creating a filter using Ansible facts

In the search field, you must enter the Ansible fact that you want to use for the filter, and its value. Use the "inject facts as variables" form of the fact name. For example, instead of using the fact named `ansible_facts['eth0']`, you should use the name `ansible_eth0`.

If you need to refer to a fact such as `ansible_eth0['mtu']` (which can also be written `ansible_eth0.mtu`), for the smart inventory filter, then you should use a double underscore in place of the period: `ansible_eth0__mtu`.

The fact name must be followed by an equals sign (=) and then the exact value for that fact on the hosts that you want to be in the inventory.

The screenshot shows a list of cached fact data in JSON format. The data includes various system information such as IP addresses, netmask, and network details. The JSON structure is as follows:

```
121     },
122     "ansible_lsb": {},
123     "ansible_eth0": {
124       "mtu": 1500,
125       "ipv4": {
126         "prefix": "24",
127         "address": "172.25.250.10",
128         "netmask": "255.255.255.0",
129         "network": "172.25.250.0",
```

Figure 7.8: Some cached fact data

Other Smart Inventory Filters

Many other ways to create a smart host filter are available. For example, you can create simple filters using the name or partial name of a host or group. You can also create advanced filters. The web UI offers the following choices:

- Name
- Group
- ID
- Inventory ID
- Enabled
- Instance ID
- Last job
- Insights system ID
- Advanced

For example, if you use the Name key with the value `example.com`, then any host that has `example.com` in its name is selected for this smart inventory.

Chapter 7 | Managing Advanced Inventories

A number of these choices use the internal ID number of the object in automation controller, which can be obtained from its API.

If you use the Advanced host filter type, then additional keys and values are available.

A range of operators and lookup types are available to match hosts.

Operator	Description
exact	Exact match (default lookup if not specified).
startswith	Field starts with value.
istartswith	Case-insensitive version of <code>startswith</code> .
endswith	Field ends with value.
iendswith	Case-insensitive version of <code>endswith</code> .
regex	Field matches the given regular expression.
iregex	Case-insensitive version of <code>regex</code> .
gt	Greater-than comparison.
gte	Greater-than or equal to comparison.
lt	Less-than comparison.
lte	Less-than or equal to comparison.
isnull	Specified field is null.
in	Field's value is present in the list provided. Items can be separated by a comma.

After creating a filter, press **Enter** to apply the filter. This gives you the opportunity to test your filter. If you create a filter and it does not return any results, then either you made a mistake or no hosts match the filter. Another common issue is that automation controller might not have any facts cached. To fix the latter problem, create and run a job template that gathers facts and stores them in the fact cache.

When you are satisfied with the filter, you can save the smart inventory. Like other inventories, you can see the hosts in a smart inventory by clicking its **Hosts** tab in the web UI.



References

Automation Controller User Guide

<https://docs.ansible.com/automation-controller/latest/html/userguide/inventories.html#smart-inventories>

► Guided Exercise

Filtering Hosts with Smart Inventories

Create a smart inventory and explore how smart inventories work.

Outcomes

- Create a smart inventory that dynamically includes hosts selected from all other inventories on this automation controller based on whether a particular Ansible fact gathered from each host has a particular value.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command ensures that automation controller is installed and configured with any necessary resources created in earlier exercises.

```
[student@workstation ~]$ lab start advinventory-smart
```

Instructions

- ▶ 1. Navigate to <https://controller.lab.example.com> and log in as the **admin** user with **redhat** as the password.
- ▶ 2. Make sure that the per-host Ansible fact cache time-out for automation controller is set for exactly one day (86400 seconds). This ensures that facts are cached for hosts but also expire if those facts have not been refreshed recently.
 - 2.1. Click **Settings** in the left navigation bar.
 - 2.2. In the **Settings** window, on the **Jobs** tile, click **Jobs settings**.
 - 2.3. Confirm that **Per-Host Ansible Fact Cache Timeout** is set to 86400 seconds. If the setting is different, click **Edit**, edit that field to contain the value 86400, and click **Save**.
- ▶ 3. Edit the **Refresh Fact Cache** job template to enable it to use fact storage, and then launch a test job using the **Dev** inventory.
 - 3.1. Click **Resources > Templates**. For the **Refresh Fact Cache** job template, click the **Edit Template** icon to edit that job template.
 - 3.2. Scroll to the bottom of the page, select the **Enable Fact Storage** checkbox, and then click **Save**.
 - 3.3. On the **Details** page, click **Launch**.
 - 3.4. Select the **Dev** inventory and then click **Next**.

- 3.5. Click **Launch** to launch the job.
- 4. Verify that the facts for `servera` and `serverb` in the Dev inventory are available in the automation controller cache.
- 4.1. Navigate to **Resources > Inventories**.
 - 4.2. Click the Dev inventory, and then click the **Hosts** tab.
 - 4.3. Click `servera.lab.example.com`, and then click the **Facts** tab.
 - 4.4. Verify that the facts for `servera.lab.example.com` are available in the cache.
 - 4.5. Repeat the preceding substeps with `serverb.lab.example.com` instead of `servera.lab.example.com` to verify that the facts for `serverb` are also available in the cache.
- 5. Create a smart inventory named `Smart` that includes the Red Hat Enterprise Linux-based systems that currently have facts cached by automation controller. This smart inventory must select hosts for which the Ansible fact `ansible_distribution` has the value `RedHat`.
- 5.1. Navigate to **Resources > Inventories**.
 - 5.2. Click **Add > Add smart inventory** to create a new smart inventory.
 - 5.3. On the **Create new smart inventory** page, complete the details as follows:

Field	Value
Name	<code>Smart</code>
Description	<code>Smart Inventory</code>
Organization	<code>Default</code>

- 5.4. Click the search icon for the **Smart host filter** field. Create an advanced filter using the following information and then click **Select**.

Field	Value
Filter type	<code>Advanced</code>
Key	<code>ansible_facts</code>
Search field	<code>ansible_distribution="RedHat"</code>

- 5.5. Click **Save** to create the smart inventory.
- 5.6. Click the **Hosts** tab and verify that both `servera.lab.example.com` and `serverb.lab.example.com` are available.

**Note**

If you previously collected facts for hosts in other inventories, then you might see additional hosts that match the smart host filter. For example, if you collected facts for the `Test` inventory, then the smart host filter also matches the `serverc.lab.example.com` and `serverd.lab.example.com` hosts.

If you collect facts for additional inventories, then hosts that match the smart host filter are added to the existing hosts in the smart inventory.

- ▶ 6. Log out of automation controller.

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from earlier exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish advinventory-smart
```

▶ Lab

Managing Advanced Inventories

Configure an inventory sourced from a project and create a smart inventory that selects its hosts from existing inventories based on the value of a host variable.

Outcomes

- Use an existing static inventory file stored in a Git repository.
- Create a smart inventory.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command ensures that private automation hub and automation controller are installed and configured with any necessary resources created in previous exercises.

```
[student@workstation ~]$ lab start advinventory-review
```

Instructions

1. Go to <https://controller.lab.example.com> and log in as the `admin` user with `redhat` as the password.
2. Create a project based on the following information:

Field	Value
Name	LabProjectGit
Organization	Default
Source Control Type	Git
Source Control URL	git@git.lab.example.com:git/advinventory-review.git
Source Control Credential	Student Git Credential
Options	Update Revision on Launch (selected)

3. Create an inventory that includes a source based on the following information:

Inventory

Field	Value
Name	LabGit
Organization	Default

Source

Field	Value
Name	git-inventory-lab
Source	Sourced from a Project
Project	LabProjectGit
Inventory file	inventory
Update options	Update on launch (selected)

When configured correctly, a successful synchronization of the `git-inventory-lab` inventory source creates the `storage` group with the `servere.lab.example.com` and `serverf.lab.example.com` hosts within the group.

4. Update the automation controller job settings so that the `Per-Host Ansible Fact Cache Timeout` setting has a value of 86400 seconds.
5. Modify the `Demo Job Template` resource to use the `LabGit` inventory and the `Operations` machine credential. Select the `Enable Fact Storage` option.
6. Launch a job using the `Demo Job Template` resource and verify that the job caches facts for the `servere.lab.example.com` and `serverf.lab.example.com` hosts in the `LabGit` inventory.
7. Create a smart inventory named `LabSmart` in the `Default` organization. Configure the smart inventory to use the `ansible_distribution="RedHat"` filter against cached Ansible facts. When configured correctly, the `LabSmart` inventory includes at least the `servere.lab.example.com` and `serverf.lab.example.com` hosts.

Evaluation

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade advinventory-review
```

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish advinventory-review
```

► Solution

Managing Advanced Inventories

Configure an inventory sourced from a project and create a smart inventory that selects its hosts from existing inventories based on the value of a host variable.

Outcomes

- Use an existing static inventory file stored in a Git repository.
- Create a smart inventory.

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that private automation hub and automation controller are installed and configured with any necessary resources created in previous exercises.

```
[student@workstation ~]$ lab start advinventory-review
```

Instructions

1. Go to <https://controller.lab.example.com> and log in as the `admin` user with `redhat` as the password.
2. Create a project based on the following information:

Field	Value
Name	LabProjectGit
Organization	Default
Source Control Type	Git
Source Control URL	<code>git@git.lab.example.com:git/advinventory-review.git</code>
Source Control Credential	Student Git Credential
Options	<code>Update Revision on Launch (selected)</code>

- 2.1. Go to **Resources > Projects** and then click **Add**.
- 2.2. Create the `LabProjectGit` project using the details from the preceding table, and then click **Save**. After you click **Save**, automation controller immediately attempts to synchronize the project.
- 2.3. On the **Details** tab, verify that the value of **Last Job Status** shows **Successful**.

 **Important**

If the project synchronization fails, then ensure that the project specifies the correct source control URL and credentials.

3. Create an inventory that includes a source based on the following information:

Inventory

Field	Value
Name	LabGit
Organization	Default

Source

Field	Value
Name	git-inventory-lab
Source	Sourced from a Project
Project	LabProjectGit
Inventory file	inventory
Update options	Update on launch (selected)

When configured correctly, a successful synchronization of the `git-inventory-lab` inventory source creates the `storage` group with the `servere.lab.example.com` and `serverf.lab.example.com` hosts within the group.

- 3.1. Go to **Resources > Inventories** and then click **Add > Add inventory**.
- 3.2. Enter **LabGit** as the name, use **Default** as the organization, and then click **Save**.
- 3.3. On the **Details** page, click the **Sources** tab and then click **Add**.
- 3.4. Create the `git-inventory-lab` source using the details from the preceding table, and then click **Save**.

 **Important**

You might see the following message below the **Project** field even if you selected the project name from the list of projects.

That value was not found. Please enter or select a valid value.

The message disappears as soon as you select a different field.

- 3.5. Click **Sync** and wait until the value of **Last Job Status** shows **Successful**.
- 3.6. Go to **Resources > Inventories** and click the **LabGit** link.

Chapter 7 | Managing Advanced Inventories

- 3.7. Click the **Groups** tab and verify that the **storage** group exists.
- 3.8. Click the **storage** link and then click the **Hosts** tab. Verify that the **storage** group contains the **servere.lab.example.com** and **serverf.lab.example.com** hosts.
4. Update the automation controller job settings so that the **Per-Host Ansible Fact Cache Timeout** setting has a value of 86400 seconds.
 - 4.1. Go to **Settings** and then click the **Job settings** link.
 - 4.2. Click **Edit**.
 - 4.3. Update the **Per-Host Ansible Fact Cache Timeout** field to use a value of 86400 seconds and then click **Save**.
5. Modify the **Demo Job Template** resource to use the LabGit inventory and the **Operations** machine credential. Select the **Enable Fact Storage** option.
 - 5.1. Go to **Resources > Templates** and then click the **Edit Template** icon for the **Demo Job Template** resource.
 - 5.2. In the **Inventory** field, select the **LabGit** inventory.
 - 5.3. In the **Credentials** field, select the **Operations** credential.
 - 5.4. Select the **Enable Fact Storage** checkbox and then click **Save**.
6. Launch a job using the **Demo Job Template** resource and verify that the job caches facts for the **servere.lab.example.com** and **serverf.lab.example.com** hosts in the **LabGit** inventory.
 - 6.1. Go to **Resources > Templates** and click the **Launch Template** icon for the **Demo Job Template** resource.
 - 6.2. The job succeeds and the job output displays the **Gathering Facts** task for the **servere.lab.example.com** and **serverf.lab.example.com** hosts.

```
...output omitted...
TASK [Gathering Facts] ****
ok: [servere.lab.example.com]
ok: [serverf.lab.example.com]
...output omitted...
```

- 6.3. Go to **Resources > Hosts** and click the link for the **servere.lab.example.com** host in the **LabGit** inventory. Click the **Facts** tab and verify that facts exist for the host.
- 6.4. Go to **Resources > Hosts** and click the link for the **serverf.lab.example.com** host in the **LabGit** inventory. Click the **Facts** tab and verify that facts exist for the host.
7. Create a smart inventory named **LabSmart** in the **Default** organization. Configure the smart inventory to use the **ansible_distribution="RedHat"** filter against cached Ansible facts. When configured correctly, the **LabSmart** inventory includes at least the **servere.lab.example.com** and **serverf.lab.example.com** hosts.
 - 7.1. Go to **Resources > Inventories** and then click **Add > Add smart inventory**.

Chapter 7 | Managing Advanced Inventories

7.2. Enter the initial details for the smart inventory:

Field	Value
Name	LabSmart
Organization	Default

7.3. Click the search icon for the **Smart host filter** field. Create an advanced filter using the following information and then click **Select**.

Field	Value
Filter type	Advanced
Key	ansible_facts
Search field	ansible_distribution="RedHat"

**Important**

Depending on the screen resolution on workstation and the size or zoom setting of your web browser, you might not be able to select **Advanced**. The responsive web UI on automation controller might hide some controls in narrow browser windows.

If this applies to you, then click the search icon again, increase your screen resolution, or reduce the zoom in your web browser.

Perform a search to define a host filter

Searching by ansible_facts requires special syntax. Refer to the documentation for more info.

Advanced ▾ ansible_facts

Name ↑	Inventory
servere.lab.example.com	LabGit
serverf.lab.example.com	LabGit
workstation.lab.example.com	Demo Inventory

1- 3 of 3 items ▾ << < 1 of 1 page > >>

Select Cancel

Figure 7.9: Create a smart host filter

7.4. Click **Save** to create the smart inventory.

- 7.5. Click the Hosts tab and verify that the `servere.lab.example.com` and `serverf.lab.example.com` hosts are available. If you do not see the specified hosts, then look for a typing mistake in the smart inventory host filter.

**Note**

You might see other hosts in addition to `servere.lab.example.com` and `serverf.lab.example.com`.

The LabSmart inventory host list contains all hosts that have cached facts and all hosts in the classroom environment are Red Hat Enterprise Linux hosts.

Evaluation

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade advinventory-review
```

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish advinventory-review
```

Summary

- You can import existing static inventory files managed in a Git repository into automation controller.
- To use an inventory stored and managed in a Git repository, configure the repository as a project and set up the inventory source as `Sourced from Project`.
- Automation controller ships with built-in dynamic inventory support for external inventory sources such as Amazon EC2, Google Compute Engine, Microsoft Azure Resource Manager, and Red Hat Satellite 6, among others.
- Automation controller provides *smart inventories*, which allow you to build an inventory by selecting hosts based on Ansible facts or other criteria from all the other inventories on automation controller.
- Smart inventories use the fact cache on automation controller to apply Ansible fact-based filters.

Chapter 8

Automating Configuration of Ansible Automation Platform

Goal

Automate the configuration and deployment of Red Hat Ansible Automation Platform services by using Ansible Content Collections, the automation controller API, and Git webhooks.

Sections

- Configuring Red Hat Ansible Automation Platform with Collections (and Guided Exercise)
- Automating Configuration Updates with Git Webhooks (and Guided Exercise)
- Launching Jobs with the Automation Controller API (and Guided Exercise)

Lab

- Automating Configuration of Ansible Automation Platform

Configuring Red Hat Ansible Automation Platform with Collections

Objectives

- Use Red Hat Certified Ansible Content Collections to configure and maintain Ansible Automation Platform services.

Automating Red Hat Ansible Automation Platform Configuration

Instead of manually configuring automation controller, private automation hub, and other components of your Ansible Automation Platform infrastructure, you can use Ansible itself to automate those tasks.

This can have clear benefits for your management of those servers and components.

- If you want to rebuild your infrastructure from scratch, you can run playbooks to reconfigure it with appropriate users, teams, credentials, projects, job templates, and more. This can help with migrations from one version to another, or for disaster recovery planning.
- You can use the version control capabilities of the Git repositories containing your automation to store the history of changes that you have made to the configuration of your automation infrastructure.
- You can use configuration as code techniques to quickly validate infrastructure changes on test servers before implementing the changes in production. You can then use the same automation to deploy those changes to production.

Red Hat supports `ansible.controller`, a Red Hat Certified Ansible Content Collection, which provides modules and other tools to help you automate the configuration and management of your automation controllers. Other community-supported Ansible Content Collections are also available but are not officially supported by Red Hat.

Getting the Supported Ansible Content Collection

You can obtain the `ansible.controller` Ansible Content Collection in several ways:

- It is included in the `ee-supported-rhel8` automation execution environment.
- Use the `ansible-galaxy collection install ansible.controller` command to install it from your automation hub, depending on your local system configuration.
- Download it from <https://console.redhat.com/ansible/automation-hub/>.

The screenshot shows the Red Hat Automation Hub download page for the `ansible.controller` collection. At the top, there's a navigation bar with links for Partners, ansible, controller, and a Red Hat logo. Below that, the collection name `Red Hat controller` is displayed, along with its version `4.2.0` (released a month ago) and last update time. A horizontal menu bar includes Install, Documentation, Contents, Import log, Dependencies, Docs site, Website, and Issue tracker. The main content area is titled "Install" and describes Ansible content that interacts with the AWX or Automation Platform Controller API. It lists categories like cloud, infrastructure, awx, ansible, and automation. It also specifies the license as GPL-3.0-only and provides instructions for installation via `ansible-galaxy collection install ansible.controller`. A note states that installing collections with `ansible-galaxy` is only supported in Ansible 2.9+. A "Download tarball" button is present. Requirements are noted as Ansible >=2.9.10.

Figure 8.1: Automation hub download page for `ansible.controller`

Exploring the Supported Ansible Content Collection

The `ansible.controller` Ansible Content Collection contains modules and plug-ins for managing automation controller. It contains modules to create, update, and delete resources such as `project`, `inventory`, `credential`, `job_template`, and `workflow_job_template`. To determine what modules are available and to view their documentation, you can:

1. Use the `ansible-navigator collections` command to view the documentation for `ansible.controller` interactively.
2. Use your web browser to find `ansible.controller` on automation hub and go to its `Documentation` tab to read the documentation.

Reading Documentation with Ansible Content Navigator

You can use the `ansible-navigator collections` command to view the documentation for Ansible Content Collections that are included with your current automation execution environment.

Because the `ee-supported-rhel8` execution environment includes `ansible.controller`, you can use the following command to access its documentation:

```
[user@demo ~]$ ansible-navigator collections --eei ee-supported-rhel8
```

This displays a list of all the collections available in the execution environment. Select `ansible.controller` from that list by typing a colon (`:`) followed by its line number on the list.

A list of the modules and other plug-ins provided by the `ansible.controller` collection is displayed. The following example shows this list, scrolled partway through the output as indicated by the scroll bar on the right.

Chapter 8 | Automating Configuration of Ansible Automation Platform

	Ansible.controller	Type	Added	Deprecated	Description
12	group	module	None	False	create, update, or destr
13	host	module	None	False	create, update, or destr
14	import	module	3.7.0	False	import resources into Au
15	instance_group	module	4.0.0	False	create, update, or destr
16	inventory	module	None	False	create, update, or destr
17	inventory_source	module	None	False	create, update, or destr
18	inventory_source_update	module	None	False	Update inventory source(
19	job_cancel	module	None	False	Cancel an Automation Pla
20	job_launch	module	None	False	Launch an Ansible Job.
21	job_list	module	None	False	List Automation Platform
22	job_template	module	None	False	create, update, or destr
23	job_wait	module	None	False	Wait for Automation Plat
24	label	module	None	False	create, update, or destr
25	license	module	None	False	Set the license for Auto
26	notification_template	module	None	False	create, update, or destr
27	organization	module	None	False	create, update, or destr
28	project	module	None	False	create, update, or destr
29	project_update	module	None	False	Update a Project in Auto
30	role	module	None	False	grant or revoke an Autom
^b/PgUp page up ^f/PgDn page down ↑↓ scroll esc back [0-9] goto :help help					

You can select the line number of any of the items on that list to get more information about how to use it.

For example, given the preceding output, if you enter :20, the documentation for the `job_launch` module, which launches a job, is displayed.

The documentation contains two particularly important sections. The `doc:` line marks the start of usage documentation.

27|`doc:`

If you scroll down further to the indented `options:` line in that output, the subsequent indented lines document options used by that module. In the following example, the first option is `controller_config_file`.

38|`options:`
39|`controller_config_file:`

If you scroll down further you can see the name option:

126|`name:`
127|`aliases:`
128|`- job_template`
129|`description:`
130|`- Name of the job template to use.`
131|`required: true`
132|`type: str`

Chapter 8 | Automating Configuration of Ansible Automation Platform

This specifies that the `job_launch` module has an option called `name` of type `string`. You can use `job_template` as a synonym for `name`. It contains the name of the job template to use to launch the job. You must specify this option when you use this module.

The other section of this output that is useful is the `examples:` section, which contains a number of examples of how to use the module.

The documentation for `job_launch` contains the following example, among others. The example on lines 187 to 190 is a task that launches a job from the `My Job Template` job template and registers a variable, `job`, that contains the job ID of the job that the task launched, among other data.

```
186|examples: |-  
187|  - name: Launch a job  
188|    job_launch:  
189|      job_template: "My Job Template"  
190|      register: job
```

Reading Documentation on Automation Hub

You can also use your web browser to read documentation for the `ansible.controller` content collection on automation hub. After you search for the `ansible.controller` content collection, go to its **Documentation** tab to learn more about its modules and other plug-ins.

Partners > ansible > controller > Documentation

Red Hat controller

Version 4.2.0 released a month ago (unstable) Last updated a month ago

Install Documentation Contents Import log Dependencies Docs site Website Issue tracker

module > user

create, update, or destroy Automation Platform Controller users.

- Synopsis
- Parameters
- Notes
- Examples

Synopsis

- Create, update, or destroy Automation Platform Controller users. See <https://www.ansible.com/tower> for an overview.

Figure 8.2: `ansible.controller.user`

Examples of Automation with `ansible.controller`

The `ansible.controller` Ansible Content Collection includes over forty modules and other plug-ins. Instead of trying to cover them all here, this presentation focuses on three of the modules as examples of how they generally work. You can review the `ansible.controller` documentation for the other modules to see what other capabilities this Ansible Content Collection provides.

Creating Automation Controller Users

The `ansible.controller.user` module adds, modifies, and deletes automation controller users.

If you do not specify that the user should have the `System Administrator` user type (using the `is_superuser` option), or `System Auditor` (using the `is_system_auditor` option), then automation controller sets up the user with the `Normal User` user type.

You can directly open the documentation for the `ansible.controller.user` module with the following command:

```
[user@demo ~]$ ansible-navigator doc ansible.controller.user \
> --eei ee-supported-rhel8
```

Documentation for the `ansible.controller.user` module shows the following example:

```
- name: Add user
  user: ①
    username: jdoe
    password: foobarbaz
    email: jdoe@example.org
    first_name: John
    last_name: Doe
    state: present
```

- ① Although the example uses `user` for the module name, you should actually use the fully qualified collection name for the module: `ansible.controller.user`.

Most modules in the `ansible.controller` Ansible Content Collection require you to specify your automation controller host and provide authentication information for that automation controller. Each module documents the following options:

Option	Description
<code>controller_config_file</code>	You might use this option if you run your playbooks from a local machine rather than from automation controller itself.
<code>controller_host</code>	The name of your controller host, such as <code>controller.example.com</code> .
<code>controller_oauthtoken</code>	Authenticate using an OAuth token.
<code>controller_password</code>	Specify the password for the automation controller user. You might define this option within a variable file and then protect the file with Ansible Vault.
<code>controller_username</code>	Specify an automation controller user that has permission to perform the desired actions. You might define this option within a variable file and then protect the file with Ansible Vault.

Option	Description
validate_certs	Set this option to either no or false if your automation controller uses custom certificates and the machine running the playbook does not trust the certificate authority that signed the custom certificates. (Ideally, for security reasons this should be set to yes or true.)

You can run your playbook locally using the `ansible-navigator` command, or you can configure automation controller to run the playbook. Running the playbook from automation controller might require creating several resources:

- An inventory resource that specifies your automation controller host.
- A project resource pointing to the source control repository that contains your playbooks and associated files. This might also require an additional SCM credential resource for the repository.
- One or more job template resources for your playbooks.
- A Vault credential if you protected your automation controller credentials (or other files) using Ansible Vault.

Creating Automation Controller Teams

The `ansible.controller.team` module adds, modifies, and deletes automation controller teams.

Documentation for the `ansible.controller.team` module shows the following example:

```
- name: Create team
  team: ①
    name: Team Name
    description: Team Description
    organization: test-org
    state: present
    controller_config_file: "~/tower_cli.cfg" ②
```

- ① Although the example uses `team` for the module name, you should actually use the fully qualified collection name for the module: `ansible.controller.team`.
- ② Only use the `controller_config_file` option if you run the playbook from your local machine.

Just like the previous example, you can run the playbook locally using the `ansible-navigator` command, or you can configure automation controller to run the playbook.

Adding Users to Organizations and Teams

The `ansible.controller.role` module grants or revokes automation controller roles for users and teams.

Individual resource types only allow specific roles. For example, you can only use the `admin`, `member`, and `read` roles when you assign a team role. At a minimum, when you create a new user, the user should have the `member` role in the automation controller organization.

Chapter 8 | Automating Configuration of Ansible Automation Platform

The following example grants the `jdoe` user the `member` role on the `Default` organization.

```
- name: Add jdoe as a member of the Default organization
  ansible.controller.role:
    user: jdoe
    organization: Default
    role: member
    state: present
```

Add a user to a team by adding a team role for the user. The following example grants the `jdoe` user the `member` role on the `My Team` team.

```
- name: Add jdoe to the member role of My Team
  ansible.controller.role:
    user: jdoe
    target_team: "My Team"
    role: member
    state: present
```



Important

The `ansible.controller.role` module has both the `team` and `target_team` options. Use the `team` option when you add or remove other resource roles (such as roles for credentials, inventories, and job templates) for a *team*. Use the `target_team` option when you add or remove team roles for a *user*.

Community-supported Ansible Content Collections

Ansible Galaxy provides additional content collections that you can use to manage Ansible Automation Platform. The following content collections from Ansible Galaxy are of interest:

- `redhat_cop.controller_configuration`: This content collection contains Ansible roles that interact with the `ansible.controller` content collection to configure automation controller.
- `redhat_cop.ah_configuration`: This content collection contains roles and modules for configuring private automation hub.

These community-supported Ansible Content Collections are currently not supported by Red Hat, but they do provide additional capabilities for the automation of Red Hat Ansible Automation Platform.



Important

Red Hat does not provide support for community Ansible Content Collections.



References

For more information, refer to the *Configuration consistency across multi Ansible Automation Platform deployments* chapter in the *Deploying Ansible Automation Platform* reference architecture at

https://access.redhat.com/documentation/en-us/reference_architectures/2021/html-single/deploying_ansible_automation_platform_2.1/index#config_as_code_using_webhooks

► Guided Exercise

Configuring Red Hat Ansible Automation Platform with Collections

Use a Red Hat certified collection to configure resources on automation controller.

Outcomes

- Use the `ansible.controller` collection in playbooks to create users and teams in automation controller.
- Use a job template to create users and teams in automation controller.

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that automation controller is installed, and creates a Git repository that provides the required files for the exercise. It also creates the following automation controller resources: a project, an inventory (with an inventory source), and two job templates.

```
[student@workstation ~]$ lab start code-collection
```

Instructions

- 1. Pull the `ee-supported-rhel8` automation execution environment from `hub.lab.example.com`. You use this automation execution environment to execute playbooks with the `ansible-navigator` command.
- 1.1. Use the `podman login` command to log in to the private automation hub at `hub.lab.example.com`, using the `admin` user with `redhat` as the password.

```
[student@workstation ~]$ podman login hub.lab.example.com
Username: admin
Password: redhat
Login Succeeded!
```

- 1.2. Pull the `ee-supported-rhel8` automation execution environment image.

```
[student@workstation ~]$ podman pull hub.lab.example.com/ee-supported-rhel8
Trying to pull hub.lab.example.com/ee-supported-rhel8:latest...
Getting image source signatures
Copying blob 858b1db62e17 done
Copying blob 3cbba8687c73 done
...output omitted...
```

- ▶ 2. Clone the `https://git.lab.example.com/git/code-collection.git` Git repository into the `/home/student/git-repos` directory. This repository contains the files for this exercise.
- 2.1. From a terminal, create the `/home/student/git-repos` directory if it does not already exist, and then change into it.

```
[student@workstation ~]$ mkdir -p ~/git-repos/  
[student@workstation ~]$ cd ~/git-repos/
```

- 2.2. Clone the `code-collection.git` repository and then change into the cloned repository:

```
[student@workstation git-repos]$ git clone \  
> https://git.lab.example.com/git/code-collection.git  
Cloning into 'code-collection'...  
...output omitted...  
[student@workstation git-repos]$ cd code-collection
```

- 2.3. Review the files inside the `code-collection` directory.

```
[student@workstation code-collection]$ tree  
. . .  
├── add_teams.yml ①  
├── add_users.yml ②  
├── ansible.cfg ③  
├── ansible-navigator.yml ④  
├── inventory ⑤  
├── tasks ⑥  
│   ├── add_user_roles.yml  
│   └── add_users.yml  
└── vars ⑦  
    ├── auth.yml  
    └── users_teams.yml  
2 directories, 9 files
```

- ① The `add_teams.yml` playbook uses the `ansible.controller.team` module to add teams to automation controller. If the team has users, the playbook calls the `tasks/add_users.yml` file to add those users to automation controller.
- ② The `add_users.yml` playbook calls the `tasks/add_users.yml` file to add users to automation controller. This playbook is appropriate for adding users who do not belong to a team, such as system administrators and system auditors.
- ③ The `ansible.cfg` file is the configuration file for this project.
- ④ The `ansible-navigator.yml` file contains configuration for the `ansible-navigator` command. This file is only used by the `ansible-navigator` command and is not used by automation controller.
- ⑤ The `inventory` file contains the inventory for this project. Because the playbooks in this project only connect to the automation controller API, the playbooks use the `localhost` host. Recall that a playbook that targets the `localhost` host runs on the automation execution environment itself.

- ⑥ The `add_user_roles.yml` and `add_users.yml` task files add users and assign roles to the automation controller resources.
 - ⑦ The `auth.yml` and `users_teams.yml` files contain variables used by the playbooks in this project. Because these files contain passwords, the files are encrypted with a Vault password of `redhat123`.
- 2.4. View the `vars/users_teams.yml` file with the `ansible-vault view` command using `redhat123` as the password. This file contains the list of users and teams. The `add_users.yml` and `add_teams.yml` playbooks use the variables defined in this file to add users and teams to automation controller. After viewing the file, press `q` to exit from the `ansible-vault view` command.

```
[student@workstation code-collection]$ ansible-vault view vars/users_teams.yml
Vault password: redhat123
---
system_auditors:
  users:
    - username: sylvia
      first_name: Syliva
      last_name: Simons
...output omitted...
```

▶ 3. Add users to automation controller by using the `ansible.controller` collection.

- 3.1. Navigate to `https://controller.lab.example.com` and log in as the `admin` user with `redhat` as the password.
- 3.2. Navigate to **Access > Users** and make a note of the existing users. The `lab` command removed previously created users and teams.
- 3.3. Go back to the terminal and review the content of the `add_users.yml` playbook.

```
---
- name: Add users to automation controller
  hosts: localhost
  gather_facts: False
  vars_files: ①
    - vars/auth.yml
    - vars/users_teams.yml
  vars:
    user_array: ②
      - "{{ system_auditors['users'] }}"
      - "{{ system_administrators['users'] }}"
      - "{{ users_no_team['users'] }}"
  tasks:
    - name: Add users with task file
      vars:
        the_users: "{{ user_array | flatten }}"
      include_tasks: tasks/add_users.yml ③
```

① Variable files encrypted with Ansible Vault.

- ❷ The `user_array` variable contains a list of users to add to automation controller. In this example, the `user_array` variable contains users from the `system_auditors`, `system_administrators`, and `users_no_teams` variables, which are defined in the `vars/users_teams.yml` file.
- ❸ The `tasks/add_users.yml` file contains tasks to add users.

3.4. Review the content of the `tasks/add_users.yml` playbook.

```
---
```

```
- name: Add users
  ansible.controller.user: ❶
    controller_host: '{{ controller_auth["host"] }}'
    controller_username: '{{ controller_auth["username"] }}'
    controller_password: '{{ controller_auth["password"] }}'
    validate_certs: False
    username: '{{ item["username"] }}'
    first_name: '{{ item["first_name"] }}'
    last_name: '{{ item["last_name"] }}'
    email: '{{ item["email"] }}'
    password: '{{ item["password"] }}'
    update_secrets: '{{ item["update_secrets"] | default(False) }}'
    is_superuser: '{{ item["is_superuser"] | default(False) }}'
    is_system_auditor: '{{ item["is_system_auditor"] | default(False) }}'
    state: present
  loop: '{{ the_users }}' ❷
  loop_control:
    label: Adding the '{{ item["username"] }}' user.

- name: Assign user roles
  vars:
    the_roles: '{{ role_item["user_roles"] }}'
  include_tasks: add_user_roles.yml ❸
  loop: '{{ the_users }}'
  loop_control:
    loop_var: role_item
    label: Assigning roles to the '{{ role_item["username"] }}' user.
```

- ❶ The task uses the `user` module from the `ansible.controller` collection to add automation controller users.
- ❷ The task loops through a variable to provide values for the `ansible.controller.user` module options. The task also provides default values for the `update_secrets`, `is_superuser`, and `is_system_auditor` options if they are not defined. Consult the documentation for the `ansible.controller.user` module for information about module options.
- ❸ The second task includes the `add_user_roles.yml` task file for each defined user. This task file uses the `ansible.controller.role` module to assign roles, such as assigning organization and team roles to users. Consult the documentation for the `ansible.controller.role` module for information about module options.

- 3.5. Run the playbook using the `ansible-navigator run add_users.yml` command with the `--ask-vault-pass` option. When prompted, enter `redhat123` as the password.

```
[student@workstation code-collection]$ ansible-navigator run add_users.yml \
> --ask-vault-pass
Vault password: redhat123
...output omitted...
TASK [Add users] ****
changed: [localhost] => (item=Adding the 'sylvia' user.)
changed: [localhost] => (item=Adding the 'simon' user.)
changed: [localhost] => (item=Adding the 'sam' user.)
changed: [localhost] => (item=Adding the 'user1' user.)
...output omitted...
```

- 3.6. In the automation controller web UI, navigate to **Access > Users** and verify that the `sam`, `simon`, `sylvia`, and `user1` users exist. The `sam` and `user1` users are normal users. The `simon` user is a system administrator and the `sylvia` user is a system auditor.

► **4.** Add teams to automation controller by using the `ansible.controller` collection.

- 4.1. Review the content of the `add_teams.yml` playbook.

```
---
- name: Add team
  hosts: localhost
  gather_facts: False
  vars_files:
    - vars/auth.yml
    - vars/users_teams.yml
  vars:
    team_array: ①
      - '{{ team1 }}'
      - '{{ team2 }}'
      - '{{ team3 }}'
      - '{{ team_developers }}'
  tasks:
    - name: Add team in controller ②
      ansible.controller.team:
        controller_host: '{{ controller_auth["host"] }}'
        controller_username: '{{ controller_auth["username"] }}'
        controller_password: '{{ controller_auth["password"] }}'
        validate_certs: False
        name: '{{ the_team["name"] }}'
        description: '{{ the_team["description"] | default(omit) }}'
        organization: '{{ the_team["organization"] | default("Default") }}'
        loop: '{{ team_array }}'
        loop_control:
          loop_var: the_team
          label: Adding the '{{ the_team["name"] }}' team.

    - name: Add users and assign roles using a task file ③
      vars:
        the_users: '{{ the_team["users"] }}'
```

```
include_tasks: tasks/add_users.yml
loop: '{{ team_array }}'
loop_control:
  loop_var: the_team
  label: Adding users to the '{{ the_team["name"] }}' team.
when: the_team["users"] is defined
```

- ➊ The `team_array` variable contains a list of teams to add. The `vars/users_teams.yml` file defines the `team1`, `team2`, `team3`, and `team_developers` variables.
 - ➋ This task adds teams to automation controller using the `ansible.controller.team` module. Consult the documentation for the `ansible.controller.team` module for information about module options.
 - ➌ Each team might define a list of users. If a team defines users, then this task passes the team users to the `tasks/add_users.yml` task file. As seen previously, that task file creates users and assigns roles to users. Because the `team1` variable does not define users, the playbook skips this task for the `team1` team.
- 4.2. Run the playbook using the `ansible-navigator` command with the `--ask-vault-pass` option. When prompted, enter `redhat123` as the password.

```
[student@workstation code-collection]$ ansible-navigator run add_teams.yml \
> --ask-vault-pass
Vault password: redhat123

PLAY [Add team] ****
TASK [Add team in controller] ****
changed: [localhost] => (item=Adding the 'team1' team.)
changed: [localhost] => (item=Adding the 'team2' team.)
changed: [localhost] => (item=Adding the 'team3' team.)
changed: [localhost] => (item=Adding the 'Developers' team.)
...output omitted...
```

- 4.3. In the automation controller web UI, navigate to **Access > Teams** and verify that the `Developers`, `team1`, `team2`, and `team3` teams exist.
 - 4.4. (Optional) For each team, click the name of the team and then click the **Access** tab to see which users belong to the team. The playbook created the team users and assigned a team role for each user. The `team1` team does not have any users aside from system users.
- ▶ 5. Modify the `add_teams.yml` playbook to add another team called `Operations`. When you are finished, commit and push your changes to the remote Git repository.
- 5.1. View the `vars/users_teams.yml` variable file and search for the `team_operations` variable. When prompted, enter `redhat123` as the password. Look at the entries in the `team_operations` variable for the `Operations` team, and the `oliver` and `ophelia` users. After viewing the file, press `q` to exit from the `ansible-vault view` command.

Chapter 8 | Automating Configuration of Ansible Automation Platform

```
[student@workstation code-collection]$ ansible-vault view vars/users_teams.yml
Vault password: redhat123
...output omitted...
team_operations:
  name: Operations
  description: Ops Team
  organization: Default
  users:
    - username: oliver
      first_name: Oliver
      last_name: Stone
      email: oliver@lab.example.com
      password: redhat123
      update_secrets: false
      is_superuser: false
      is_system_auditor: false
      user_roles:
        - role: member
          user: oliver
          organization: Default
        - role: member
          user: oliver
          target_team: Operations
...output omitted...
```

- 5.2. Modify the `add_teams.yml` playbook to include the `team_operations` variable, which is already defined in the `vars/users_teams.yml` file. By making this change, the playbook adds the `Operations` team and the `oliver` and `ophelia` users. Edit the `add_teams.yml` file and add the `'{{ team_operations }}'` variable to the `team_array` variable. The content of the `add_teams.yml` file should contain the following:

```
---
- name: Add team
  hosts: localhost
  gather_facts: False
  vars_files:
    - vars/auth.yml
    - vars/users_teams.yml
  vars:
    team_array:
      - '{{ team1 }}'
      - '{{ team2 }}'
      - '{{ team3 }}'
      - '{{ team_developers }}'
      - '{{ team_operations }}'
```

- 5.3. Commit and push the changes to the remote Git repository.

```
[student@workstation code-collection]$ git add add_teams.yml
[student@workstation code-collection]$ git commit -m "Added Operations team"
[main d164229] Added Operations team
```

```
1 file changed, 1 deletion(-)
[student@workstation code-collection]$ git push -u origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 298 bytes | 298.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
To https://git.lab.example.com/git/code-collection.git
  c967656..d164229  main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

- ▶ 6. The `lab` command created several automation controller resources for you, but the `lab` command did not create a Vault credential. Create a Vault credential that you can use to automatically decrypt the `vars/auth.yml` and `vars/users_teams.yml` files.
- 6.1. In the automation controller web UI, navigate to **Resources > Credentials** and then click **Add**.
 - 6.2. On the **Create New Credential** page fill in the details as follows:

Field	Value
Name	<code>vault-cred</code>
Description	To unlock files in the Code Collection project
Organization	Default
Credential Type	Vault
Vault Password	<code>redhat123</code>

- 6.3. Click **Save**.

▶ 7. Associate the `vault-cred` credential with the `Add Teams` and `Add Users` job templates.

 - 7.1. Navigate to **Resources > Templates**.
 - 7.2. Click the **Edit Template** icon for the `Add Teams` job template.
 - 7.3. Click the search icon under **Credentials**.
 - 7.4. In the **Select Credentials** dialog box, click the **Selected Category** list to access the list of available credential types. Choose the **Vault** credential from the list.
 - 7.5. In the lower pane, select the credential named `vault-cred` and click **Select**.
 - 7.6. Click **Save**.
 - 7.7. Repeat these steps for the `Add Users` job template.

**Important**

Make sure that you add the Vault credential to both the Add Teams and the Add Users job templates.

► 8. Launch the Add Teams job template.

- 8.1. Navigate to **Resources > Templates**.
- 8.2. Click the **Launch Template** icon for the **Add Teams** job template and wait for the job to complete.

► 9. Verify that the job template adds the Operations team to the automation controller.

- 9.1. Navigate to **Access > Teams**.
- 9.2. Click the link for the **Operations** team.
- 9.3. Click the **Access** tab and confirm that the **oliver** and **ophelia** users have roles in the **Operations** team.

**Note**

This exercise automated the creation of users and teams for your automation controller, as one example of what the `ansible.controller` Ansible Content Collection can do.

You can also use the modules and other plug-ins provided by the collection to automate many other tasks, whether you want to create or remove other resources such as projects, credentials, and job templates, or to actually run job templates and workflows on your automation controller.

Finish

On the **workstation** machine, change to the **student** user home directory and use the **lab** command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish code-collection
```

Automating Configuration Updates with Git Webhooks

Objectives

- Apply configuration changes to Red Hat Ansible Automation Platform based on files stored in a Git repository by triggering job execution with webhooks.

Using Webhooks in Red Hat Ansible Automation Platform

A webhook is a user-defined HTTP callback, performed as a HTTP POST request, that allows applications to communicate and share information with each other. Most frequently, webhooks are used to notify applications about events. The message sent by the POST request is usually formatted as JSON or XML content or form data that the application can interpret. Webhook calls are usually authenticated to ensure that unauthorized requests are rejected.

You can set up your automation controller to use webhooks so that it is notified when events occur for a Git repository, such as new commits being added to a particular branch. Automation controller can then perform specific tasks when it receives certain events. For example, a new commit to a Git repository could cause automation controller to automatically update a related project and launch a job template that uses it.

What Are the Benefits of Webhooks?

Webhooks help you automate workflows that involve multiple applications and components.

As an example of one way that webhooks can help automate workflows, the following diagram illustrates a common Continuous Integration (CI) workflow that might be used to perform tasks for the devices managed by Ansible.

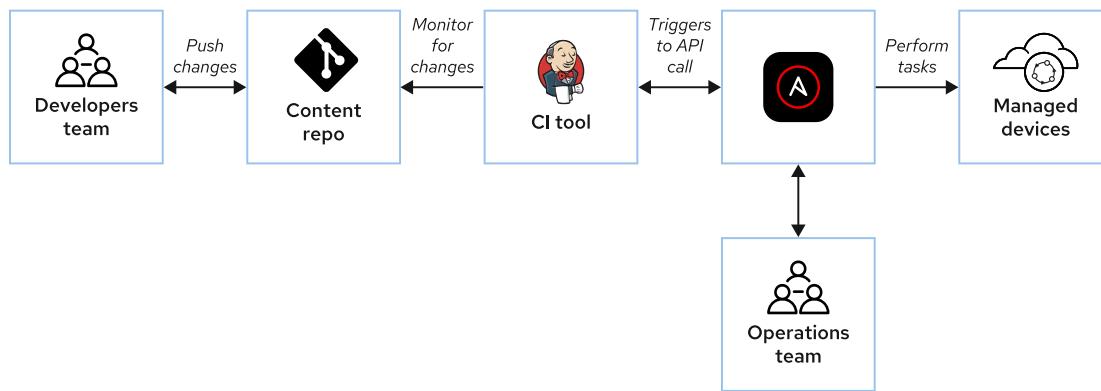


Figure 8.3: Common CI workflow

In this example, the operations team is in charge of providing automation content that performs tasks on the Ansible-managed hosts. When the developers' team pushes a change to the Git content repository, the Jenkins CI tool captures this event. The event signals Jenkins to use the API on automation controller to automatically launch a job template that updates its copy of the project and runs on the managed hosts.

The following diagram shows another approach that uses webhooks:

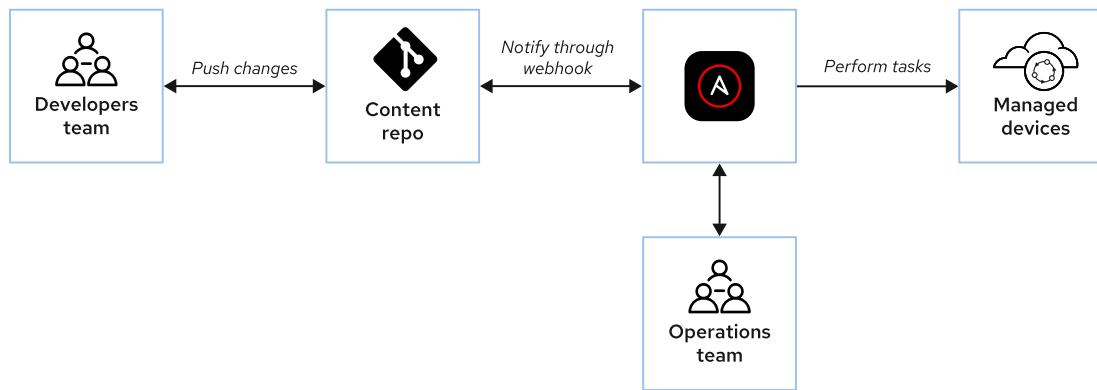


Figure 8.4: Continuous integration workflow using webhooks

By using webhooks, you simplify the workflow. In this case, when a developer pushes something to the Git content repository, automation controller gets notified directly by a webhook. Automation controller then launches a job template that updates its copy of the project and runs on the managed hosts.

Therefore, one major benefit is that you can use webhooks to trigger actions on automation controller without the need to maintain and manage an additional Continuous Integration (CI) tool such as Jenkins.

Configuring Webhooks

To set up webhooks, you need to prepare both your Git repository server so that it triggers webhooks, and your automation controller to listen for them. You also need to configure both sides so that the Git repository can authenticate its messages to automation controller.



Important

In the following example, GitLab is used as the Git repository server because it matches the classroom environment used by this course for hands-on exercises.

In practice, you can use other Git repository servers and services such as GitHub instead. They can be configured using a procedure similar to the one illustrated for GitLab by this course.

Configuring Automation Controller to Listen for Webhooks

To configure a project to update a job template to launch automatically when automation controller is notified by a webhook, you must configure the project and job template to meet certain requirements:

- For the project, you must enable the **Update Revision on Launch** option. This ensures that automation controller pulls the latest revision from the Git repository when it launches the job template because the Git repository has changed.
- Because you cannot interact with the job template, you must disable *all* the **Prompt on launch** settings in any job template that webhook notifications might launch.
- After you have met the previous requirements, then you have to configure automation controller to listen for webhook notifications.

The following example outlines one way to do this for notifications from a GitLab repository. It enables a webhook listener for an existing job template that is already configured to run without prompting and that uses a project that is configured with **Update Revision on Launch** selected.

Chapter 8 | Automating Configuration of Ansible Automation Platform

1. Log in to the automation controller web UI as the **admin** user.
2. Navigate to **Resources > Templates**, and click the **Edit Template** icon for the job template that you want to edit.
3. Select the **Enable Webhook** checkbox, and choose **GitLab** from the **Webhook Service** field.
4. Click **Save** and then go to **Details** tab to see the webhook URL and webhook key needed to configure the webhook in GitLab. GitLab needs to send the webhook notification to the webhook URL and use the webhook key to authenticate its message to automation controller.



Figure 8.5: Configuring a job template webhook on automation controller

Creating the Webhook for the Repository in GitLab

Before any user can configure webhooks on your GitLab server, an administrator of the server must allow them to be used.

If this has not been done, you need to log in to your GitLab server as a user that has an **admin** role. Navigate to **Admin Area > Settings > Network > Outbound requests**. Make sure that the automation controller to which you want to send webhook notifications is listed under **Local IP addresses and domain names that hooks and services may access**.

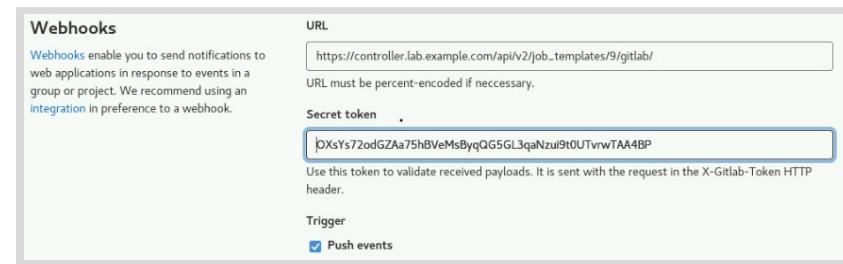
The result should appear similar to the following:



Figure 8.6: Enabling GitLab to permit webhook notifications to an automation controller

If this requirement is correctly configured, a regular user can create webhooks for their Git projects.

1. Log in to the GitLab web UI as a regular user.
2. Navigate to your Git project (one of the repositories under **Projects > Your projects**), and then go to **Settings > Webhooks**.
3. On the **Webhooks** configuration page, create the webhook by adding the information you obtained from automation controller, putting the webhook URL into the **URL** field and the webhook key into the **Secret token** field.
4. In the **Trigger** section, select which events trigger the webhook notification, and click **Add webhook**.

**Figure 8.7: Webhook configuration in GitLab**

5. You can test the webhook by clicking **Test > Push events**. You get the message **Hook executed successfully** if you configured it correctly.

Use Cases for Webhooks

You can use webhooks with automation controller in a variety of ways. The subsections that follow explore two of them.

Triggering Different Job Templates Using Branches

For this use case, you use a webhook for a job template specifying a branch from the **Source Control Branch** field. When you push new content for the given branch, the job template starts performing Ansible tasks on the inventory that you set up for that job template.



Important

The **Source Control Branch** field is only available if you set up a Git project with **Allow Branch Override** selected.

For example, suppose you want to test a Git repository for a development environment before going to production. In that case, you can create a development branch for the repository, configure the job template to use this branch, and associate the inventory with the development hosts. When you add a commit to the development branch, the webhook launches the job template for the development hosts.

After you verify that the development branch of your automation is working correctly, you can merge your changes to the production branch, and even use the same webhook mechanism to automatically launch a job template that runs for managed hosts in your production inventory.

Configuration as Code

Configuration as code is a technique that uses a version control repository to describe the desired state of your configuration.

You can use configuration as code methods in conjunction with Ansible Content Collections and Ansible Playbooks to manage the configuration of your Red Hat Ansible Automation Platform systems in a version control system. This is similar in principle to the concept of Infrastructure as Code, and has similar benefits.

For example, you could implement configuration as code to automatically reconfigure and help you maintain your Ansible Automation Platform servers. You set up files and Ansible Playbooks in a Git repository that define and apply the desired configuration for your automation infrastructure.

You can accomplish this by combining webhooks with playbooks that use the `ansible.controller` Ansible Content Collection. Using this Ansible Content Collection, you

can configure most of the objects in automation controller (such as job templates, projects, inventories, and so on), using Git as a single source of truth.

To illustrate how this might work, imagine that you have a Git repository that contains an Ansible Playbook named `add-users.yml` that adds users to your automation controller.

That playbook contains the following play:

```
---
- name: Add user
  hosts: workstation
  gather_facts: False
  tasks:
    - name: Add user in controller
      ansible.controller.user:
        controller_host: "{{ controller_auth['host'] }}"
        controller_username: "{{ controller_auth['username'] }}"
        controller_password: "{{ controller_auth['password'] }}"
        validate_certs: False
        username: "{{ item['username'] }}"
        first_name: "{{ item['first_name'] }}"
        last_name: "{{ item['last_name'] }}"
        email: "{{ item['email'] }}"
        password: "{{ item['password'] }}"
        update_secrets: "{{ item['update_secrets'] | default(False) }}" 1
        is_superuser: "{{ item['is_superuser'] | default('False') }}" 2
        is_system_auditor: "{{ item['is_system_auditor'] | default('False') }}" 3
        state: present
      loop: "{{ system_users['users'] }}"
      loop_control:
        label: "Adding user: {{ item['username'] }}"
```

- 1** If the `update_secrets` variable is not defined for the user, then instead of producing an error, the variable is assigned a value of `False`.
- 2** The `is_superuser` variable provides a default value of `False` if the variable is not set.
- 3** The `is_system_auditor` variable provides a default value of `False` if the variable is not set.



Important

A user can be a system administrator (a super user), a system auditor, or a normal user. If the `is_superuser` variable has a value of `True`, then the `is_system_auditor` variable must have a value of `False` (and vice versa). If both variables have a value of `False`, then the user is a normal user.

The preceding play ensures that users exist on automation controller by looping over a list of users, and information about those users, stored in the `system_users['users']` variable.

For example, to update that information without changing the Ansible Playbook, that variable is set in the project's `host_vars/workstation/users_teams.yml` file. This could be written as follows:

```

---
system_users:
  users:
    - username: patrick
      first_name: Patrick
      last_name: Star
      email: patrick@lab.example.com
      password: redhat123
      update_secrets: False
      is_superuser: False
      is_system_auditor: False
      user_roles:
        - role: member
          user: patrick
          organization: Default
...output omitted...

```

You modify the `users_teams.yml` host variable file to update the list of users that should exist on automation controller, including appropriate information about what user type and team memberships each user should have.

Then, the following steps occur when you commit and push to the Git repository:

- It triggers the webhook and notifies automation controller.
- Automation controller detects the change through the webhook and then launches a job template to run the changed content based on information in the notification message.

The webhook message contains information about who pushed the change, to what branch, what files were modified, and so on. The content of the message is available to your plays through the `tower_webhook_payload` variable. The following figure shows an excerpt of the payload that automation controller receives from the GitLab webhook:

```

Variables
Variables YAML JSON
95+ tower_webhook_payload:
96  object_kind: push
97  event_name: push
98  before: 181bfcd3844dff1a66310d02562398fdaf52e39cc
99  after: 89644586638fb7a37d788cc78a2693fff4be9d54
100 ref: refs/heads/main
101 ref_type: push
102 checkout_sha: 89644586638fb7a37d788cc78a2693fff4be9d54
103 message: null
104 user_id: 2
105 user_name: Student User
106 user_email: ''
107 user_avatar: >
108 https://secure.gravatar.com/avatar/96d4a357e7137eb31491bab13f2fd01f?s=80&d=identicon
109 project_id: 2
110 project_name: MOTD
111 id: 2
112 name: MOTD

```

Figure 8.8: Selected content from a GitLab webhook notification

You can also combine the ability to trigger a job template based on a given Git branch with this configuration as code technique to link different configuration states to different environments, such as development, test, and production.

To summarize some benefits of storing and managing your Ansible Automation Platform configuration as code:

- You can standardize your configuration in several operating environments.

- You can easily track the changes in your configuration by using the version control features and see who made what changes and when, which can help you troubleshoot and resolve issues.
- You can easily replicate and deploy additional Ansible Automation Platform deployments with the same configuration.



References

For more information on webhooks, refer to the "Working with Webhooks" chapter in the *Automation Controller User Guide* at
<https://docs.ansible.com/automation-controller/latest/html/userguide/webhooks.html>

For more information on configuration as code, refer to the "Configuration consistency across multi Ansible Automation Platform deployments" chapter in the *Deploying Ansible Automation Platform 2.1* reference architecture at
https://access.redhat.com/documentation/en-us/reference_architectures/2021/html-single/deploying_ansible_automation_platform_2.1/index#config_as_code_using_webhooks

► Guided Exercise

Automating Configuration Updates with Git Webhooks

Use webhooks to reconfigure automation controller when you commit changes to content that is stored in a Git repository.

Outcomes

- Configure a job template that runs when a Git server triggers a webhook to notify your automation controller that a new commit has been applied to the repository for the job template's project.

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that automation controller is installed. The command creates a Git repository that provides the required files for the exercise, and also set up the configuration needed in GitLab to allow webhooks from `controller`. Finally, the command creates the following automation controller resources: a project, an inventory (with an inventory source), a Vault credential, and two job templates that use the Vault credential.

```
[student@workstation ~]$ lab start code-webhooks
```

Instructions

- 1. On your automation controller, configure an existing job template named `Add Teams` to use a webhook.
- 1.1. Navigate to `https://controller.lab.example.com` and log in as the `admin` user with `redhat` as the password.
 - 1.2. Navigate to `Resources > Templates` and click the `Edit Template` icon for the `Add Teams` job template.
 - 1.3. Scroll down to the `Options` section and select `Enable Webhook`.
 - 1.4. In the `Webhook details` section, choose `GitLab` from the `Webhook Service` list.
 - 1.5. Click `Save`.
 - 1.6. On the `Details` tab, take note of the `Webhook URL` and `Webhook Key` values needed to configure the webhook in GitLab in the next step.

Webhook URL	<code>https://controller.lab.example.com/api/v2/job_templates/11/gitlab/</code>	Webhook Key	<code>vJygx4F7PaCGD6g4D57y3hruzYmxqJWmvSMCMv7COrIKQD7Zp</code>
-------------	---	-------------	--

Figure 8.9: Webhook configuration in job template

- 2. On your GitLab server, create a webhook for the **Code Collection** Git repository that notifies your automation controller about push events.

**Important**

The `lab start code-webhooks` script has run as an administrator on your GitLab server to allow its users to create webhooks, so you do not need to do that part of the configuration.

- 2.1. Navigate to <https://git.lab.example.com> and log in as the **student** user using **Student@123** as the password.
- 2.2. Navigate to **Projects > Your projects > Code Collection**, and then navigate to **Settings > Webhooks** in the left panel.
- 2.3. On the webhook configuration page, create the webhook by adding the **Webhook URL** and **Webhook Key** values, obtained from automation controller in the previous step, to the **URL** and **Secret token** fields, respectively.

The screenshot shows the 'Webhooks' configuration page in GitLab. It includes fields for 'URL' (containing a placeholder URL), 'Secret token' (containing a long secret key), and a 'Trigger' section with a checked 'Push events' option.

URL	<code>https://controller.lab.example.com/api/v2/job_templates/11/gitlab/</code>
URL must be percent-encoded if necessary.	
Secret token	<code>vJygx4F7PaCGD6g4D57y3hruzYmxqJWmvSMCMvS7COrIKQD7Zp</code>
Use this token to validate received payloads. It is sent with the request in the X-Gitlab-Token HTTP header.	
Trigger	<input checked="" type="checkbox"/> Push events

Figure 8.10: Webhook configuration in GitLab

**Important**

The preceding screen capture is only an illustration. Use the **URL** and **Secret token** values for your own environment. Use the **Webhook URL** and **Webhook Key** values that you generated in your automation controller UI for your **URL** and **Secret token** fields.

- 2.4. Select **Push events** and clear **Enable SSL verification**.

**Note**

Normally, you would not disable SSL verification. You would instead ensure that both automation controller and GitLab could validate each other's TLS/SSL certificates.

- 2.5. Click **Add webhook** to save the webhook.
- 2.6. Scroll down to the bottom of the page. In the **Project Hooks** section, notice that you now have one webhook configured. Click **Test** and choose **Push events** to test the webhook. You should see the **Hook executed successfully** message.

► 3. Test the webhook by making a change to the code-collection Git repository.

- 3.1. As the student user on the workstation machine, create the /home/student/git-repos directory if it does not already exist, and then change into it.

```
[student@workstation ~]$ mkdir -p ~/git-repos/  
[student@workstation ~]$ cd ~/git-repos/
```

- 3.2. Clone the code-collection.git repository and then change into the cloned repository:

```
[student@workstation git-repos]$ git clone \  
> https://git.lab.example.com/git/code-collection.git  
Cloning into 'code-collection'...  
...output omitted...  
[student@workstation git-repos]$ cd code-collection
```

- 3.3. Edit the add_teams.yml file and add the '{{ team_operations }}' variable to the team_array variable. The content of the add_teams.yml file should contain the following.

```
---  
- name: Add team  
  hosts: localhost  
  gather_facts: False  
  vars_files:  
    - vars/auth.yml  
    - vars/users_teams.yml  
  vars:  
    team_array:  
      - '{{ team1 }}'  
      - '{{ team2 }}'  
      - '{{ team3 }}'  
      - '{{ team_developers }}'  
      - '{{ team_operations }}'
```

- 3.4. Commit and push the changes to the remote Git repository.

```
[student@workstation code-collection]$ git add add_teams.yml  
[student@workstation code-collection]$ git commit -m "Added Operations team"  
[main d164229] Added Operations team  
 1 file changed, 1 deletion(-)  
[student@workstation code-collection]$ git push -u origin main  
Enumerating objects: 5, done.  
Counting objects: 100% (5/5), done.  
Delta compression using up to 4 threads  
Compressing objects: 100% (3/3), done.  
Writing objects: 100% (3/3), 298 bytes | 298.00 KiB/s, done.  
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0  
To https://git.lab.example.com/git/code-collection.git  
  c967656..d164229  main -> main  
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

- 3.5. In the automation controller web UI, navigate to **View > Jobs**.
- 3.6. Your `git push` command launched the **Add Teams** job template and started the **jobs Source Control Update, Inventory Sync, and Playbook Run**.
- 3.7. After the **Playbook Run** job is successful, navigate to **Access > Teams** and verify that all the teams now exist in automation controller.
- 3.8. *(Optional)* Display the content of the `vars/users_teams.yml` file to verify the `team_operations` definition. Use `redhat123` as a Vault password.

```
[student@workstation code-collection]$ ansible-vault view vars/users_teams.yml
Vault password: redhat123
...output omitted...
team_operations:
  name: Operations
  description: Ops Team
  organization: Default
  users:
    - username: oliver
      first_name: Oliver
      last_name: Stone
      email: oliver@lab.example.com
      password: redhat123
      update_secrets: false
      is_superuser: false
      is_system_auditor: false
...output omitted...
```

Finish

On the **workstation** machine, change to the **student** user home directory and use the **lab** command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish code-webhooks
```

Launching Jobs with the Automation Controller API

Objectives

- Control automation controller by accessing its API.

The Automation Controller REST API

Automation controller provides a Representational State Transfer (REST) API that allows you to control automation controller without using the web UI. Custom scripts or external applications use standard HTTP messages to access the REST API. The REST API is useful if you want to integrate automation controller with other programs or make automated changes to automation controller.

One of the benefits of the REST API is that any programming language, framework, or system that supports HTTP can use the API. This provides a way to automate repetitive tasks and integrate other enterprise IT systems with automation controller.



Note

The API is in active development, and some features of the web UI might not be accessible through the API. Version 2 is the only version of the API currently available.

Using the REST API

In case you are not familiar with REST APIs, the way they work is relatively straightforward.

A web client sends a request to a server element located at a Uniform Resource Identifier (URI) and performs operations with standard HTTP methods, such as GET, POST, PUT, PATCH, and DELETE. The REST architecture provides a stateless communication channel between the client and the server. Each client request acts independently of any other request, and contains all the necessary information to complete the request.

The following example request uses an HTTP GET method to retrieve a representation of the main entry point of the API. You can use a graphical web browser or Linux command-line tools to issue the request manually. This example uses the curl command to make the request from the command line:

```
[user@demo ~]$ curl -X GET https://automation.example.com/api/ -k
{"description":"AWX REST API","current_version":"/api/v2/","available_versions":
{"v2":"/api/v2/"},"oauth2":"/api/o/","custom_logo":"","custom_login_info":"",
"login_redirect_override":""}
```

The output of the API request is in JSON format, which is readily parseable by computer programs, but might be a little challenging for a human to read.

The automation controller API is browsable. For example, if your automation controller is the automation.example.com host, you can access the browsable API at https://

automation.example.com/api/. You can click the /api/v2/ link on that page to browse information specific to version 2 of the API.

The following example shows how to do this using the curl command. The jq command is provided by the jq RPM package and "pretty-prints" the JSON output of the API for easier reading.

```
[user@demo ~]$ curl -X GET https://automation.example.com/api/v2/ -k -s | jq .  
{  
    "ping": "/api/v2/ping/",  
    "instances": "/api/v2/instances/",  
    "instance_groups": "/api/v2/instance_groups/",  
    "config": "/api/v2/config/",  
    "settings": "/api/v2/settings/",  
    "me": "/api/v2/me/",  
    "dashboard": "/api/v2/dashboard/",  
    "organizations": "/api/v2/organizations/",  
    "users": "/api/v2/users/",  
    "execution_environments": "/api/v2/execution_environments/",  
    "projects": "/api/v2/projects/",  
    "project_updates": "/api/v2/project_updates/",  
    "teams": "/api/v2/teams/",  
    "credentials": "/api/v2/credentials/",  
    "credential_types": "/api/v2/credential_types/",  
    "credential_input_sources": "/api/v2/credential_input_sources/",  
    "applications": "/api/v2/applications/",  
    "tokens": "/api/v2/tokens/",  
    "metrics": "/api/v2/metrics/",  
    "inventory": "/api/v2/inventories/",  
    "inventory_sources": "/api/v2/inventory_sources/",  
    "inventory_updates": "/api/v2/inventory_updates/",  
    "groups": "/api/v2/groups/",  
    "hosts": "/api/v2/hosts/",  
    "job_templates": "/api/v2/job_templates/",  
    "jobs": "/api/v2/jobs/",  
    "ad_hoc_commands": "/api/v2/ad_hoc_commands/",  
    "system_job_templates": "/api/v2/system_job_templates/",  
    "system_jobs": "/api/v2/system_jobs/",  
    "schedules": "/api/v2/schedules/",  
    "roles": "/api/v2/roles/",  
    "notification_templates": "/api/v2/notification_templates/",  
    "notifications": "/api/v2/notifications/",  
    "labels": "/api/v2/labels/",  
    "unified_job_templates": "/api/v2/unified_job_templates/",  
    "unified_jobs": "/api/v2/unified_jobs/",  
    "activity_stream": "/api/v2/activity_stream/",  
    "workflow_job_templates": "/api/v2/workflow_job_templates/",  
    "workflow_jobs": "/api/v2/workflow_jobs/",  
    "workflow_approvals": "/api/v2/workflow_approvals/",  
    "workflow_job_template_nodes": "/api/v2/workflow_job_template_nodes/",  
    "workflow_job_nodes": "/api/v2/workflow_job_nodes/",  
    "mesh_visualizer": "/api/v2/mesh_visualizer/"  
}
```

This entry point provides a collection of links in the API environment. As you can see in the example, many links are available.

**Note**

Other packages besides jq are available that can parse the JSON output in a human-readable way. For example, you can use the json_pp command provided by the perl-JSON-PP RPM package.

The following example illustrates some information accessible through the API. To examine what actions have been performed on automation controller, use the /api/v2/activity_stream/ URI. Make a GET request to that resource to retrieve the list of activity streams:

```
[user@demo ~]$ curl -X GET \
> https://automation.example.com/api/v2/activity_stream/ -k
{"detail":"Authentication credentials were not provided.
To establish a login session, visit /api/login/.\"}
```

Notice that not all information generated by the API is publicly available. You need to log in to access this resource.

The next example shows the output of the activity_stream resource when you provide correct authentication information (in this case, providing the password redhat for the admin account with the --user option):

```
[user@demo ~]$ curl -X GET --user admin:redhat \
> https://automation.example.com/api/v2/activity_stream/ -k -s | jq .
{
  "count": 45,
  "next": "/api/v2/activity_stream/?page=2",
  "previous": null,
  "results": [
    {
      ...
    }
    ...output omitted...
    {
      "id": 23,
      "type": "activity_stream",
      "url": "/api/v2/activity_stream/23/",
      "related": {
        "execution_environment": [
          "/api/v2/execution_environments/6/"
        ]
      },
      "summary_fields": {
        "execution_environment": [
          {
            "id": 6,
            "name": "Automation Hub Default execution environment",
            "description": "",
            "image": "private.example.com/ee-supported-rhel8:latest"
          }
        ]
      }
    },
    ...
  ]
},
```

```
"timestamp": "2022-06-03T16:43:38.100345Z",
"operation": "create",
"changes": {
    "name": "Automation Hub Default execution environment",
    "description": "",
    "organization": null,
    "image": "private.example.com/ee-supported-rhel8:latest",
    "credential": null,
    "pull": "",
    "id": 6
},
"object1": "execution_environment",
"object2": "",
"object_association": "",
"action_node": "automation.example.com",
"object_type": ""
},
...output omitted...
```

JSON Pagination

The JSON output of the API can be *paginated*. This divides the output into separate sequential pages that have similar content. Automation controller only returns a limited number of records from a particular request for performance reasons.

Review the first few lines of the preceding example, and notice the following information:

```
"next": "/api/v2/activity_stream/?page=2"
"previous": null
```

The `next` value gives you the URI of the next page of results. The `previous` value gives you the URI of the previous page of results. If the value for the previous page is `null`, then you are on the first page. Likewise, if the value for the next page is `null`, then you are on the last page.

In the same example as before, you can use the `https://automation.example.com/api/v2/activity_stream/?page=2` URI to display the second page of the `activity_stream` resource. This updates the `next` and `previous` values:

```
"next": null,
"previous": "/api/v2/activity_stream/?page=1"
```

In this example the JSON output for the API of the `activity_stream` resource has two pages.

Accessing the REST API from a Graphical Web Browser

Because the output in JSON format for the API can be difficult to read without a parser such as `jq`, you can also access the browsable REST API with a graphical web browser. When you use a graphical web browser, you have the same information in a more readable format.

The following example shows the output of the `activity_stream` resource using the Firefox web browser. As before, if the resource provided by the API is not public, you need to log in to access that resource.

The screenshot shows a REST API interface for version 2. The top navigation bar includes 'REST API / Version 2 / Job List'. Below it, a 'Job List' section has a 'GET' method selected. A 'GET /api/v2/jobs/' request is shown. The response is an 'HTTP 401 Unauthorized' error with the following headers and body:

```
HTTP 401 Unauthorized
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
WWW-Authenticate: Bearer realm=api authorization_url=/api/o/authorize/
X-API-Node: automation.example.com
X-API-Product-Name: Red Hat Ansible Automation Platform
X-API-Product-Version: 4.2.0
X-API-Time: 0.008s

{
    "detail": "Authentication credentials were not provided. To establish a login session, visit /a
}
```

Figure 8.11: API output without authentication credentials

After you authenticate, you can perform the query for the resource that is not publicly available. The next and previous values and the exact number of pages shown in the output match the ones you had before, this time in an easier-to-read way.

The screenshot shows a REST API interface for version 2. The top navigation bar includes 'REST API / Version 2 / Activity Stream List'. Below it, an 'Activity Stream List' section has an 'OPTIONS' method selected. A 'GET /api/v2/activity_stream/' request is shown. The response is an 'HTTP 200 OK' success with the following headers and body:

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
X-API-Node: automation.example.com
X-API-Product-Name: Red Hat Ansible Automation Platform
X-API-Product-Version: 4.2.0
X-API-Time: 0.765s

{
    "count": 45,
    "next": "/api/v2/activity_stream/?page=2",
    "previous": null,
    "results": [
        {
            "id": 1,
            "type": "activity_stream",
            "url": "/api/v2/activity_stream/1/",
            "related": {
                "user": [
                    "/api/v2/users/1/"
                ]
            }
        }
    ]
}
```

Figure 8.12: Activity stream API output

Chapter 8 | Automating Configuration of Ansible Automation Platform

You can click various links in the API to explore related resources.

REST API / Version 2

Version 2 ⓘ

OPTIONS GET

GET /api/v2/

HTTP 200 OK

Allow: GET, HEAD, OPTIONS

Content-Type: application/json

Vary: Accept

X-API-Node: automation.example.com

X-API-Product-Name: Red Hat Ansible Automation Platform

X-API-Product-Version: 4.2.0

X-API-Time: 0.009s

{

 "ping": "/api/v2/ping/",
 "instances": "/api/v2/instances/",
 "instance_groups": "/api/v2/instance_groups/",
 "config": "/api/v2/config/",
 "settings": "/api/v2/settings/",
 "me": "/api/v2/me/",
 "dashboard": "/api/v2/dashboard/",
 "organizations": "/api/v2/organizations/",
 "users": "/api/v2/users/",
 "execution_environments": "/api/v2/execution_environments/",
 "projects": "/api/v2/projects/",
 "project_updates": "/api/v2/project_updates/",
 "teams": "/api/v2/teams/",
 "credentials": "/api/v2/credentials/",
 "credential_types": "/api/v2/credential_types/",
 "credential_input_sources": "/api/v2/credential_input_sources/",
 "applications": "/api/v2/applications/",
 "tokens": "/api/v2/tokens/",
 "metrics": "/api/v2/metrics/",
 "inventory": "/api/v2/inventories/",
 "inventory_sources": "/api/v2/inventory_sources/",
 "inventory_updates": "/api/v2/inventory_updates/",
 "groups": "/api/v2/groups/",
 "hosts": "/api/v2/hosts/",
 "job_templates": "/api/v2/job_templates/",
 "jobs": "/api/v2/jobs/",
 "ad_hoc_commands": "/api/v2/ad_hoc_commands/",
 "system_job_templates": "/api/v2/system_job_templates/",
 "system_jobs": "/api/v2/system_jobs/",
 "schedules": "/api/v2/schedules/",
 "roles": "/api/v2/roles/",
 "notification_templates": "/api/v2/notification_templates/",
 "notifications": "/api/v2/notifications/",
 "labels": "/api/v2/labels/",
 "unified_job_templates": "/api/v2/unified_job_templates/",
 "unified_jobs": "/api/v2/unified_jobs/",
 "activity_stream": "/api/v2/activity_stream/",
 "workflow_job_templates": "/api/v2/workflow_job_templates/",
 "workflow_jobs": "/api/v2/workflow_jobs/",
 "workflow_approvals": "/api/v2/workflow_approvals/",
 "workflow_job_template_nodes": "/api/v2/workflow_job_template_nodes/",
 "workflow_job_nodes": "/api/v2/workflow_job_nodes/",
 "mesh_visualizer": "/api/v2/mesh_visualizer/"

}

Copyright © 2021 Red Hat, Inc. All Rights Reserved.

Figure 8.13: Browsable API

Chapter 8 | Automating Configuration of Ansible Automation Platform

Click the ? icon next to an API endpoint name to get documentation on the access methods for that endpoint. The documentation also provides information on what data is returned when using those methods.

The screenshot shows a REST API documentation page for the 'Job List' endpoint. At the top, there are navigation links for 'REST API / Version 2 / Job List' and a 'Show/Hide Description' button. To the right are buttons for 'OPTIONS', 'GET', and a dropdown menu. Below the header, the title 'Job List' is displayed with a question mark icon. On the right side of the title are 'OPTIONS', 'GET', and a dropdown menu. A star icon is located in the top right corner of the main content area. The main content starts with the heading 'List Jobs:' followed by the instruction 'Make a GET request to this resource to retrieve the list of jobs.' Below this, it says 'The resulting data structure contains:' and shows a JSON snippet:

```
{  
  "count": 99,  
  "next": null,  
  "previous": null,  
  "results": [  
    ...  
  ]  
}
```

The text explains that the `count` field indicates the total number of jobs found, `next` and `previous` fields provide links to additional results if there are more than will fit on a single page, and the `results` list contains zero or more job records.

Results

Each job data structure includes the following fields:

- `id` : Database ID for this job. (integer)
- `type` : Data type for this job. (choice)
- `url` : URL for this job. (string)
- `related` : Data structure with URLs of related resources. (object)

Figure 8.14: Documentation for API endpoints



Note

The documentation available to graphical browsers through the ? icon is useful when you are trying to understand how to use the automation controller API.

You can also use PUT, POST, or PATCH methods on the specific API pages by providing JSON-formatted text or files in the graphical interface.

For example, you can put or patch content in the text field for the `api/v2/settings/jobs` API resource using your Firefox browser. After editing the content, click `PUT` to put the content or `PATCH` to patch it:



Figure 8.15: Example of PUT and PATCH methods text field

Launching a Job Template Using the API

One common use of the API is to launch an existing job template. This example outlines how to use the API to find and launch a job template that has already been configured in automation controller. It uses the `curl` command to interact with the API.

You can refer to a job template by name in the API. For example, you can use the GET method to get information about a job template. The following example illustrates how to use this method on the `Demo Job Template` job template. Because the name of the job template contains spaces, you must escape them using double quotes or URL percent encoding (%20 for each space character).

```
[user@demo ~]$ curl -X GET --user admin:redhat \  
> https://automation.example.com/api/v2/job_templates/"Demo Job Template"/ \  
> -k -s | jq .
```

Launching a job template from the API is done in two steps.

1. Access the API with the GET method to get information about any parameters or data that you need to launch the job.
2. Access the API with the POST method to launch the job.

The following example uses the GET method to get the parameters that you need to launch the `Demo Job Template` job template through the API. The output is piped into `jq` for better readability.

```
[user@demo ~]$ curl -X GET --user admin:redhat \  
> https://automation.example.com/api/v2/job_templates/"Demo Job Template"/launch/ \  
> -k -s | jq .  
{  
    ...output omitted...  
    "defaults": {
```

```
"extra_vars": "",  
"diff_mode": false,  
"limit": "",  
"job_tags": "",  
"skip_tags": "",  
"job_type": "run",  
"verbosity": 0,  
"inventory": {  
    "name": "Demo Inventory",  
    "id": 1  
},  
"credentials": [  
    {  
        "id": 1,  
        "name": "Demo Credential",  
        "credential_type": 1,  
        "passwords_needed": []  
    }  
],  
"scm_branch": ""  
}
```



Note

For more detailed information, refer to the chapter on launching job templates in the "Automation Controller API Guide" in the references at the end of this section.

To launch the job, the job template or API call must provide the inventory ID, the inventory name, the credential ID and the credential name. The output of the previous example, which used the GET method, included all this information, so you would not need to provide it in your API call to launch the job template. You can launch the job by accessing the same URI with the POST method. In this example, the job with ID 1 reports "status": "pending" in the returned information about the job because it has been launched but has not completed yet.

```
[user@demo ~]$ curl -X POST --user admin:redhat \  
> https://automation.example.com/api/v2/job_templates/"Demo Job Template"/launch/\<br>  
> -k -s | jq .  
{  
    "job": 1,  
    "ignored_fields": {},  
    "id": 1,  
    "type": "job",  
    "url": "/api/v2/jobs/1/",  
    ...output omitted...  
    "created": "2022-06-16T22:16:56.384290Z",  
    "modified": "2022-06-16T22:16:56.464627Z",  
    "name": "Demo Job Template",  
    "description": "",  
    "job_type": "run",  
    "inventory": 1,  
    "project": 6,  
    "playbook": "hello_world.yml",
```

```
"scm_branch": "",  
"forks": 0,  
"limit": "",  
"verbosity": 0,  
"extra_vars": "{}",  
"job_tags": "",  
"force_handlers": false,  
"skip_tags": "",  
"start_at_task": "",  
"timeout": 0,  
"use_fact_cache": false,  
"organization": 1,  
"unified_job_template": 7,  
"launch_type": "manual",  
"status": "pending",  
"execution_environment": null,  
...output omitted...
```

The JSON-formatted output of this example shows the `id` of this job is 1. You can use the job ID to retrieve updated status information, such as whether the job has completed. For job 1, use the URI `/api/v2/jobs/1/`, as indicated by the `url` field in the preceding output.

```
[user@demo ~]$ curl -X GET --user admin:redhat \  
> https://automation.example.com/api/v2/jobs/1/ -k -s | jq .
```

This reports job status (success or failure), what time the job finished, a link to the output of the Ansible Playbook, which playbook was used, as well as other information about the inventory, credentials, and project from the job template, and more.



Note

You can also launch a job template using its internal ID number instead of its name.

First, find the `id` number of your job template. If you know the name of the job template, you can use the API to search for it. For example, if the desired job template is named `A Job Template`, you can search for it using the `curl` command:

```
[user@demo ~]$ curl -X GET --user admin:redhat \  
> https://automation.example.com/api/v2/job_templates/?name=\  
> "A Job Template" -k -s | jq .  
...output omitted...  
    "id": 6,  
    "type": "job_template",  
    "url": "/api/v2/job_templates/6/",  
...output omitted...
```

You can now refer to the job template by its ID number and not its name, as follows:

```
[user@demo ~]$ curl -X POST --user admin:redhat \  
> https://automation.lab.example.com/api/v2/job_templates/6/launch/ -k -s
```

Launching a Job Using the API from an Ansible Playbook

You can use an Ansible Playbook to launch a job template by using the `uri` module to access the automation controller API. You can even run that playbook from a job template in automation controller and use it to launch another job template as one of its tasks.

In the playbook, you have to specify the correct URL to your job template, using its ID or its named URL. You must also provide sufficient credentials to automation controller to authenticate as a user who has permission to launch the job.

The following Ansible Playbook can start a new job from one of the existing job templates in automation controller, using the automation controller API:

```
---
- name: Automation Controller API
  hosts: localhost
  become: false

  vars:
    automation_controller_user: admin ①
    automation_controller_pass: redhat
    automation_controller_host: automation.example.com
    automation_controller_job: Demo%20Job%20Template ②

  tasks:
    - name: Launch a new Job
      uri: ③
        url: https://{{ automation_controller_host }}/api/v2/job_templates/{{ automation_controller_job }}/launch/
        method: POST
        validate_certs: no
        return_content: yes
        user: "{{ automation_controller_user }}"
        password: "{{ automation_controller_pass }}"
        force_basic_auth: yes
        status_code: 201
```

- ①** To access the API, Ansible requires the login credentials for a user with sufficient permissions to launch a job from a job template. In the example, two variables are used to store the relevant information: `automation_controller_user` and `automation_controller_pass`.
- ②** The playbook also requires the URL of the actual API to which Ansible must connect. This example uses the named URL to the `Demo Job Template`. Notice how the spaces in the job template name have been specified using the `%20` code. You can also use the `urlencode` filter: `automation_controller_job: "{{ 'Demo Job Template' | urlencode }}"`.
- ③** Ansible can access the automation controller API from the automation controller server by using the `uri` module. As the `uri` module is part of `ansible-core`, you can refer to it using its module name, `uri`, without specifying the collection.

The problem with this example is that it embeds the username and password for authentication to automation controller in the playbook. To protect that data, you should either encrypt the

Chapter 8 | Automating Configuration of Ansible Automation Platform

playbook with Ansible Vault, or move the secrets into a variable file and encrypt that file with Ansible Vault. You should do this before you commit files containing those secrets to your source control repository.

```
[user@demo ~]$ ansible-vault encrypt api_demo.yml
New Vault password: your_password
Confirm New Vault password: your_password
```

Vault Credentials

For automation controller to use encrypted files (such as a playbook or an included variable file containing secrets), you must set up a Vault credential that can decrypt those files in automation controller. You must also configure any job templates that use the project with a Vault credential, in addition to any machine or other credentials that the project needs.

First, create a Vault credential that stores the Vault password for those files. This credential is encrypted and stored in the database of the automation controller server, just like the machine credentials.

The following procedure describes how to create this type of credential:

- Log in to the automation controller web UI as a user with the appropriate role assignment. If you are creating a private credential, no specific role requirements apply. If you are creating an organization credential, log in as a user with the Admin role for the organization.
- Click **Resources > Credentials**.
- Click **Add** to open the **Create New Credential** page.

Credentials

Create New Credential

Name *	Description
vault-credential-demo	Vault credential
Organization	Credential Type *
Default	Vault
Type Details	
Vault Password *	<input type="checkbox"/> Prompt on launch
	••••••••
Save	Cancel

Figure 8.16: New Vault credential

Chapter 8 | Automating Configuration of Ansible Automation Platform

- Enter a name for the new credential in the **Name** field.
- If desired, enter a description for the Vault credential in the **Description** field.
- If you have organization Admin privileges, click the search icon and select the organization name from the **Select Organization** dialog box to assign this credential to to make it an organization credential. If you do not have Admin privileges, do not select any organization.
- Select **Vault** from the **Credential Type** list. Notice that additional fields are displayed in the **Type Details** section.
- The **Vault Password** is the password with which you encrypted the playbook. This is a required field for Vault credentials.
- The **Vault Identifier** is the optional Vault ID; only needed if the playbook has been encrypted using multiple passwords.
- Click **Save** to save the new Vault credential.

After you have the Vault credential in place, you can create a new job template that uses it to decrypt your encrypted project file or files. This job template must include the Vault credential required to decrypt the project files.

Use the following procedure to include the Vault credential in the new job template:

- Click the search icon in the **Credentials** field of the job template.

The screenshot shows the 'Create New Job Template' dialog box. The 'Name' field contains 'Job template using a vault credential'. Under 'Job Type', 'Run' is selected. In the 'Project' field, 'Project' is typed into the search bar. The 'Playbook' field contains 'vault_cred.yml'. The 'Credentials' field is highlighted with a red box and contains 'SSH: Demo Credential'. A 'Prompt on launch' checkbox is located next to the 'Credentials' field.

Figure 8.17: Selecting a credential for the job template

- A pop-up window opens to select the credentials. Choose **Vault** in the **Selected Category** list.
- Select the credential that corresponds to the job template.

- Click **Select** to select the credential.

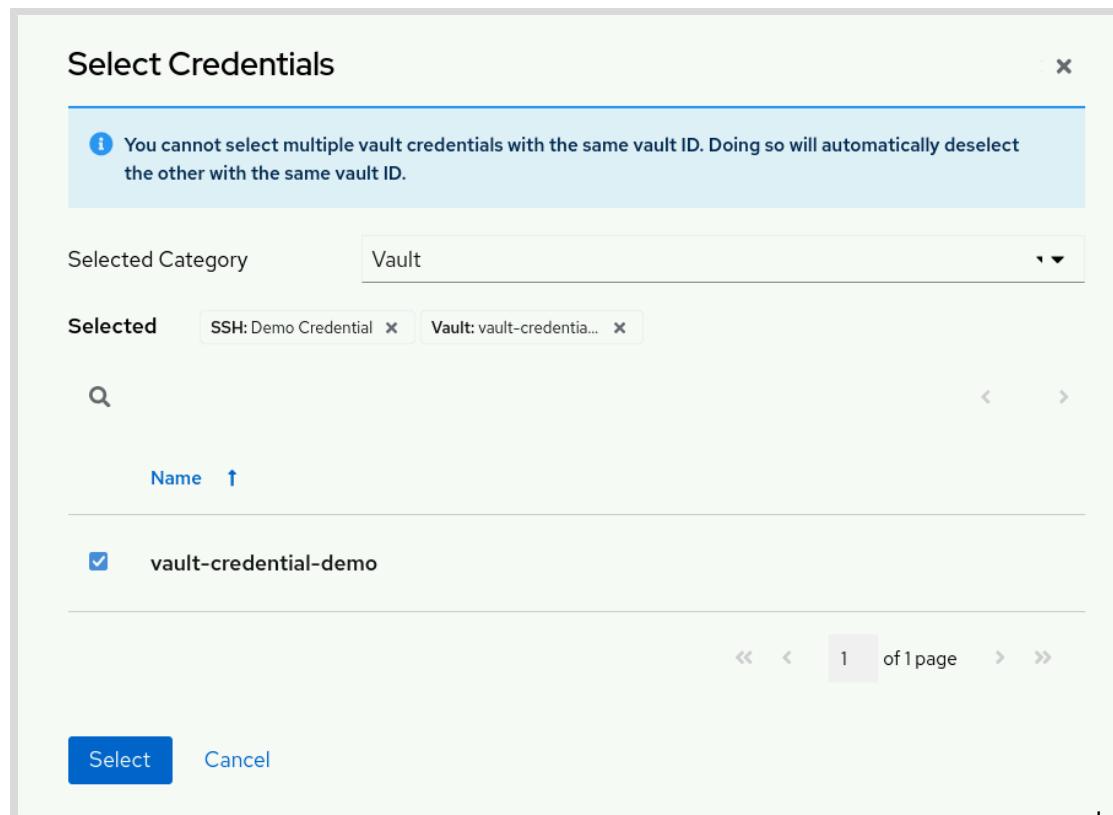


Figure 8.18: Vault credential for a job template

You can launch a job using the new job template. Automation controller decrypts the encrypted playbook using the Vault credential. When it runs the playbook, automation controller executes the task that accesses the API to launch another job template. This launches a new job on the automation controller server, using the job template referenced by the task in the playbook.



Note

In recent versions of Ansible, you can encrypt different files with different Ansible Vault passwords. Automation controller can use multiple Vault credentials in the same job template to ensure that it can decrypt all files in the project that were encrypted with Ansible Vault.

Token-based Authentication

The API for automation controller uses OAuth 2 to provide token-based authentication. Any call to the API with a valid token in the headers of the request can be authenticated.

Two kinds of tokens exist:

Application Tokens

Requested for an application that was previously created in automation controller, and represents a client application that accesses the API frequently with multiple users.

Personal Access Tokens (PATs)

PATs are a much simpler mechanism, providing access to the API for a single user.

Chapter 8 | Automating Configuration of Ansible Automation Platform

This example shows a PAT request, as well as how to use the issued token to launch a job template.

```
---
- name: Automation Controller API
  hosts: localhost
  gather_facts: false

  vars:
    automation_controller_user: admin
    automation_controller_pass: redhat
    automation_controller_host: automation.example.com
    template_name: DEV ftpservers setup

  tasks:
    - name: Get the token
      uri:
        url: "https://{{ automation_controller_host }}/api/v2/users/1/personal_tokens/"
        method: POST
        validate_certs: false
        return_content: true
        user: "{{ automation_controller_user }}"
        password: "{{ automation_controller_pass }}"
        force_basic_auth: true 1
        status_code: 201
      register: response 2

    - name: Use the token
      uri:
        url: "https://{{ automation_controller_host }}/api/v2/job_templates/{{ template_name | urlencode }}/launch/"
        method: POST
        validate_certs: false
        return_content: true
        status_code: 201
        headers:
          Authorization: "Bearer {{ response['json']['token'] }}" 3
          Content-Type: "application/json"
      register: launch
```

- 1** Use a regular authentication mechanism.
- 2** Save the result for later use.
- 3** Use the token in the response variable to provide the authentication.

Some token-related configurations are possible that you can set up from the automation controller web UI.

For example, you can configure the expiration date of the access token, or add the personal tokens for a user. Use the following procedure to add a new personal token for a user:

- Log in to the automation controller web UI as your user.
- Navigate to **Access > Users** and then click **Username**.

Chapter 8 | Automating Configuration of Ansible Automation Platform

- Click the **Tokens** tab.
- Click **Add** to add the new token.
- In the **Scope** list, choose either **Read or Write**.
- If desired, enter a description for the token.
- If you are creating a PAT, leave the **Application** field empty. Otherwise, select the application for the token.

The following procedure describes how to modify the access and refresh expiration times for the token:

- Log in to the automation controller web UI as the **admin** user.
- Navigate to **Settings**.
- On the **System** tile, click **Miscellaneous Authentication settings**.
- Scroll to the bottom of the page and click **Edit**.

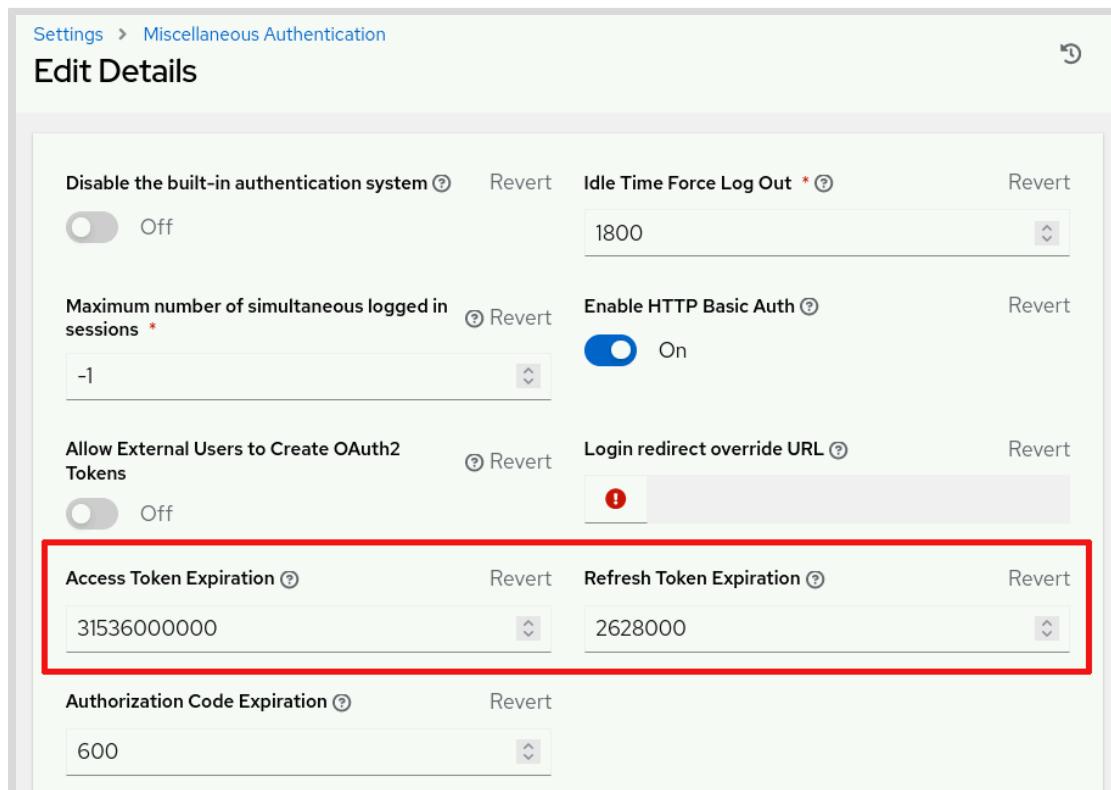


Figure 8.19: Configuring the expiration of an access token and its associated refresh token

- Modify the **Access Token Expiration** or **Refresh Token Expiration** as required, and then click **Save**.



References

Automation Controller API Guide

<https://docs.ansible.com/automation-controller/latest/html/controllerapi/index.html>

Controller API Reference Guide

https://docs.ansible.com/automation-controller/latest/html/controllerapi/api_ref.html

Token-Based Authentication

https://docs.ansible.com/automation-controller/latest/html/administration/oauth2_token_auth.html

Multi-Vault Credentials

<https://docs.ansible.com/automation-controller/4.2.0/html/administration/multi-creds-assignment.html#ag-multi-vault>

Add Tokens

https://docs.ansible.com/automation-controller/latest/html/userguide/applications_auth.html#ug-tokens-auth-create

► Guided Exercise

Launching Jobs with the Automation Controller API

Launch a job from an existing job template by running an Ansible Playbook.

Outcomes

- Launch a job from an existing job template by running an Ansible Playbook.

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that automation controller is installed and configured with any necessary resources created in previous exercises. It also creates additional Vault credentials and uploads a new playbook into the Git repository.

```
[student@workstation ~]$ lab start code-api
```

Instructions

- ▶ 1. In a web browser, log in to automation controller as `admin` and set the fact cache timeout to one day.
 - 1.1. Go to `https://controller.lab.example.com` and log in as the `admin` user with `redhat` as the password.
 - 1.2. Click **Settings** in the navigation bar, and then click the **Jobs settings** link.
 - 1.3. Scroll down and click **Edit**.
 - 1.4. Set **Per-Host Ansible Fact Cache Timeout** to 86400 seconds (one day), and then click **Save**.
- ▶ 2. Go to `https://controller.lab.example.com/api/` to examine the browsable API on that automation controller, using a new tab in your web browser.
 - 2.1. Open a new tab in your web browser, and go to `https://controller.lab.example.com/api/`. You should see the browsable API for automation controller. Because you logged in as `admin` on this web browser and have not logged out, you are authenticated to the API as the `admin` user.
 - 2.2. Click one of the `/api/v2/` link to access the browsable list of all the available resources accessible throughout the API.
 - 2.3. From the list, click the `/api/v2/ping/` link to access that URI. On the `instances` object for the `controller.lab.example.com` node you can see the heartbeat time stamp value.

An external program can use this URI to verify that the automation controller server is operating.

```
...output omitted...
"instances": [
{
  "node": "controller.lab.example.com",
  "node_type": "hybrid",
  "uuid": "0230c1cb-5f20-4db4-b186-394bc8e9ccea",
  "heartbeat": "2022-06-14T17:07:55.992650Z",
  "capacity": 36,
  "version": "4.2.0"
}
...output omitted...
```

- 3. In your web browser, use the API to launch a job from the existing Refresh Fact Cache job template.
- 3.1. Click the **Version 2** link at the top of the page to return to the `/api/v2/` URI.
 - 3.2. Click the `/api/v2/job_templates/` link to access the list of available job templates.
 - 3.3. From the list of results, locate the job template containing "name": "Refresh Fact Cache". Scroll up to find its `url` resource and click the link to access the template. The link should be `/api/v2/job_templates/11/` or similar. (The number at the end of the link might be a different number.)
 - 3.4. Look for the job template's `related` resource named `launch`. Click the link associated with that resource. The link should be `/api/v2/job_templates/11/launch/` or similar.

```
...output omitted...
  "schedules" : "/api/v2/job_templates/11/schedules/",
  "launch" : "/api/v2/job_templates/11/launch/",
  "notification_templates_success" : "/api/v2/job_templates/11/
notification_templates_success/"
...output omitted...
```

This link sends a GET request to the `launch` resource for that job template, providing details about what information is needed to launch a job from the template.

- 3.5. To watch the execution of the job you are about to launch, return to the automation controller web UI tab. Click **Views > Jobs** in the navigation bar.
- 3.6. To watch the execution of the job you are about to launch, return to the automation controller web UI tab. Click **Views > Jobs** in the navigation bar.
- 3.7. Go back to the `https://controller.lab.example.com/api/v2/job_templates/11/launch/` tab.
Go to the bottom of the page and click **POST** to launch the job.
The page immediately refreshes with JSON information about the launched job, including the job `id` number.
- 3.8. Quickly switch to the tab displaying the **Jobs** page in the automation controller web UI.

Chapter 8 | Automating Configuration of Ansible Automation Platform

The launched job should be at the top of the list of jobs that have been run. You can confirm it is the job you launched by matching its ID in the web UI on this tab with the id in the JSON output for the job on the other tab.

► 4. Use curl to launch a job from an existing job template through the API.

- 4.1. Search for the Refresh Fact Cache job template by using the API with a name filter. Determine the ID of the job template. This ID should be the same as the one that you saw using your web browser earlier in this exercise.

```
[student@workstation ~]$ curl -X GET --user admin:redhat \
> https://controller.lab.example.com/api/v2/job_templates/?name="Refresh Fact
Cache" \
> -k -s | jq
```

In the preceding command, the -k option instructs curl to operate even if it cannot verify the SSL certificate. The -s option instructs curl to only display the returned data; it does not display progress or error messages.

```
"survey_spec" : "/api/v2/job_templates/11/survey_spec/",
"launch" : "/api/v2/job_templates/11/launch/",
"activity_stream" : "/api/v2/job_templates/11/activity_stream/",
"jobs" : "/api/v2/job_templates/11/jobs/",
```

- 4.2. Now that you have confirmed the ID for the Refresh Fact Cache job template, use that number and the job template's launch resource to get information about how to launch the job.

```
[student@workstation ~]$ curl -X GET --user admin:redhat \
> https://controller.lab.example.com/api/v2/job_templates/11/launch/ -k -s \
> | jq
{
  "passwords_needed_to_start" : ,
  "ask_limit_on_launch" : false,
  "ask_inventory_on_launch" : false,
  "can_start_without_user_input" : true,
  "defaults" : {
    ...output omitted...
```

- 4.3. Because you can launch this job without user input, you can issue a POST request to the same URL to launch the job, without providing any additional data.

```
[student@workstation ~]$ curl -X POST --user admin:redhat \
> https://controller.lab.example.com/api/v2/job_templates/11/launch/ -k -s \
> | jq
{
  ...output omitted...
  "unified_job_template" : 11,
  "skip_tags" : "",
  "execution_environment" : 3,
  "job_type" : "run",
  "forks" : 0,
  "job_slice_number" : 0,
```

```
"job_template" : 11,
"extra_vars" : "{}",
"created" : "2022-06-17T20:35:38.191129Z",
...output omitted...
"use_fact_cache" : true,
"result_traceback" : "",
"artifacts" : {},
"job_env" : {},
"project" : 10,
"id" : 7,
"ignored_fields" : {},
"job" : 7,
"job_cwd" : "",
...output omitted...
}
```

Pay attention to the **id** number for the launched job in the JSON output.

- 4.4. Return to the automation controller web UI tab that displays the **Jobs** page.

The job launched by the `curl` command should now be at the top of the list of jobs that have run. Again, you can confirm that it is the job you just launched by comparing the ID in the web UI with the **id** you just saw in the JSON output from the `curl` command.

- 5. You have been provided with a playbook named `api_controller.yml` that uses the automation controller API to launch a job from an existing job template. Look at the playbook to see how it works.

The username and password to access the API are stored in this playbook, and so it is encrypted using the `ansible-vault` command, with a password of `redhat`.

- 5.1. From a terminal, create the `/home/student/git-repos` directory if it does not already exist, and then change into it.

```
[student@workstation ~]$ mkdir -p ~/git-repos/
[student@workstation ~]$ cd ~/git-repos/
```

- 5.2. Clone the `https://git.lab.example.com/git/internal_web.git` repository, which contains the `api_controller.yml` playbook, to the `/home/student/git-repos` directory.

```
[student@workstation git-repos]$ git clone \
> https://git.lab.example.com/git/internal_web.git
Cloning into 'internal_web'...
...output omitted...
[student@workstation git-repos]$ cd internal_web
```

- 5.3. Inspect the contents of the `api_controller.yml` playbook. The contents of the file are encrypted:

```
[student@workstation internal_web]$ cat api_controller.yml
$ANSIBLE_VAULT;1.1;AES256
62633766376433336366313731643761663863656339623938646234386334633764646436626363
3065366133316630616164343461366465323833626631660a373165303136383236653033613663
31646132653439646464363735666236653636393761336632326138373137343438626430663039
3165356237326236310a37303432343433636438303762353662396135626333863396334303163
...output omitted...
```

- 5.4. You can use the `ansible-vault view` command to temporarily decrypt and display the contents of the playbook:

```
[student@workstation internal_web]$ ansible-vault view api_controller.yml
Vault password: redhat
...output omitted...
vars:
  controller_user: admin
  controller_pass: redhat
  controller_host: controller.lab.example.com
tasks:
  - name: Launch Job
    uri:
      url: https://{{ controller_host }}/api/v2/job_templates/Internal%20Web
%20Access/launch/
      method: POST
      validate_certs: no
      return_content: yes
      user: "{{ controller_user }}"
      password: "{{ controller_pass }}"
      force_basic_auth: yes
      status_code: 201
...output omitted...
```

- 6. In automation controller, create a new Vault credential named `Vault Usage` that contains the password to decrypt the `api_controller.yml` playbook.

- 6.1. Click **Resources > Credentials** and then click **Add**.

- 6.2. On the **Create New Credential** page, fill in the details as follows:

Field	Value
Name	<code>Vault Usage</code>
Description	To unlock <code>api_controller.yml</code>
Credential Type	<code>Vault</code>
Password	<code>redhat</code>

- 6.3. Click **Save**.

Chapter 8 | Automating Configuration of Ansible Automation Platform

- 7. A job template named API Usage that launches the `api_controller.yml` playbook has already been partially configured for you. Edit the job template to associate the Vault Usage credential with it so that automation controller can decrypt the playbook.
- 7.1. Go to Resources > Templates.
 - 7.2. Click the **Edit Template** icon for the API Usage job template.
 - 7.3. Click the search icon under **Credentials**. In the **Select Credentials** window, click the **Selected Category** list to access the list of available credential types. Choose **Vault** from the list.
 - 7.4. In the lower pane, select the Vault Usage credential name, and then click **Select** to add this new credential to the API Usage job template.
 - 7.5. Click **Save** at the bottom of the **Edit Details** page.
- 8. Launch a new job based on the modified API Usage job template.
- 8.1. Go to Resources > Templates.
 - 8.2. Launch the job by clicking the **Launch Template** icon for the API Usage job template. Observe the output of the job. The **Print Output** task shows how the automation controller API was used by the job playbook to launch a second job.
 - 8.3. Click **Jobs** in the navigation bar.
A new job named Internal Web Access has been launched. This new job was launched by the job you launched using the API Usage job template.

Finish

On the **workstation** machine, change to the **student** user home directory and use the **lab** command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish code-api
```

▶ Lab

Automating Configuration of Ansible Automation Platform

Automate the configuration of automation controller settings using the `ansible.controller` Ansible content collection and a job template triggered by a webhook.

Outcomes

- Configure an existing job template to use a GitLab webhook.
- Browse the automation controller API to identify variable names for settings.
- Push changes to a Git repository and verify that automation controller launches the job template that applies changes.

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that automation controller is installed and configured with any necessary resources created in previous exercises. It also initializes the remote `https://git.lab.example.com/git/code_review.git` Git repository, which you need for this exercise.

```
[student@workstation ~]$ lab start code-review
```

Instructions

- The `lab start code-review` command configured a job template named `Apply Controller Settings`, which modifies some automation controller settings.
Go to `https://controller.lab.example.com` and log in as the `admin` user with `redhat` as the password. Verify that the job template exists and configure it to use a GitLab webhook.
- The `https://git.lab.example.com/git/code_review.git` Git repository contains the `Controller Settings` GitLab project that includes a playbook to modify settings for your automation controller. You can log in to the GitLab server as `student` using `Student@123` as the password.
Configure the `Controller Settings` GitLab project to use the webhook URL and webhook key defined for the `Apply Controller Settings` job template. The GitLab project should trigger the webhook on push events, and you need to disable SSL verification for it.
- At `https://controller.lab.example.com/api/v2/settings/all/`, browse the API settings for your automation controller and identify the name of the variable used to set the maximum number of forks per job. By default, automation controller sets the value of that variable to 200.

4. Change the setting on your automation controller for the maximum number of forks per job to a value of 50. Test the webhook by making that change in your Git repository.

As the student user on the `workstation` machine, open a terminal window and clone the Git repository at `https://git.lab.example.com/git/code_review.git` to the `/home/student/git-repos` directory. Modify the `~/git-repos/code_review/vars/settings.yml` variable file to set the maximum number of forks per job to 50. Save, commit, and push your changes to the remote repository. Use `Set maximum forks` as your commit message.

5. Confirm that the webhook launched the `Apply Controller Settings` job template when you pushed your changes to the Git repository. From the automation controller web UI, verify that the job was successful and the maximum number of forks per job is now 50.

Evaluation

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful. You *must* add, commit, and push any additional changes to the remote Git repository before grading.

```
[student@workstation ~]$ lab grade code-review
```

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish code-review
```

► Solution

Automating Configuration of Ansible Automation Platform

Automate the configuration of automation controller settings using the `ansible.controller` Ansible content collection and a job template triggered by a webhook.

Outcomes

- Configure an existing job template to use a GitLab webhook.
- Browse the automation controller API to identify variable names for settings.
- Push changes to a Git repository and verify that automation controller launches the job template that applies changes.

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that automation controller is installed and configured with any necessary resources created in previous exercises. It also initializes the remote `https://git.lab.example.com/git/code_review.git` Git repository, which you need for this exercise.

```
[student@workstation ~]$ lab start code-review
```

Instructions

- The `lab start code-review` command configured a job template named `Apply Controller Settings`, which modifies some automation controller settings.
Go to `https://controller.lab.example.com` and log in as the `admin` user with `redhat` as the password. Verify that the job template exists and configure it to use a GitLab webhook.
 - Go to `https://controller.lab.example.com` and log in as the `admin` user with `redhat` as the password.
 - Go to **Resources > Templates** and verify that **Apply Controller Settings** is in the list of job templates.
 - Click the **Edit Template** icon for the **Apply Controller Settings** job template.
 - Select the **Enable Webhook** checkbox.
 - In the **Webhook Service** list, choose **GitLab**, and then click **Save**.

Chapter 8 | Automating Configuration of Ansible Automation Platform

The screenshot shows a configuration dialog for enabling a webhook. At the top, under 'Options', the 'Enable Webhook' checkbox is checked. Below that, the 'Webhook details' section is shown. It includes fields for 'Webhook Service' (set to 'GitLab'), 'Webhook URL' (set to 'https://controller.lab.example.com/api/v2/job_templates/9/gitlab/'), 'Webhook Key' (a placeholder text 'A NEW WEBHOOK KEY WILL BE GENERATED...'), and 'Webhook Credential' (a search bar). At the bottom are 'Save' and 'Cancel' buttons.

Figure 8.20: Enabling a webhook in the "Apply Controller Settings" job template

- 1.6. Pay attention to the Webhook URL and Webhook Key values generated with this configuration because you use them in the next step.

The screenshot shows the 'Details' tab for the 'Apply Controller Settings' job template. The table lists various configuration parameters. The 'Webhook URL' and 'Webhook Key' rows are highlighted with a red border. The 'Webhook URL' value is 'https://controller.lab.example.com/api/v2/job_templates/9/gitlab/' and the 'Webhook Key' value is 'Pnj4SsWVSbClgHbcJLrQAvTPP2ZIRIJazhYojDZLYOCV2yhjO'.

Details	
Name	Apply Controller Settings
Description	Modify some automation controller settings.
Job Type	run
Organization	Default
Inventory	Dev
Project	Controller Settings
Execution Environment	Default execution environment
Playbook	modify_settings.yml
Forks	0
Verbosity	0 (Normal)
Timeout	0
Show Changes	Off
Job Slicing	1
Webhook Service	GitLab
Webhook URL	https://controller.lab.example.com/api/v2/job_templates/9/gitlab/
Webhook Key	Pnj4SsWVSbClgHbcJLrQAvTPP2ZIRIJazhYojDZLYOCV2yhjO

Figure 8.21: Webhook data for the "Apply Controller Settings" job template

**Important**

The values in the **Webhook URL** and **Webhook Key** fields on your automation controller differ from the ones shown here. Ensure you use the ones provided by your automation controller.

2. The `https://git.lab.example.com/git/code_review.git` Git repository contains the **Controller Settings** GitLab project that includes a playbook to modify settings for your automation controller. You can log in to the GitLab server as **student** using **Student@123** as the password.

Configure the **Controller Settings** GitLab project to use the webhook URL and webhook key defined for the **Apply Controller Settings** job template. The GitLab project should trigger the webhook on push events, and you need to disable SSL verification for it.

- 2.1. In a separate browser window or tab, go to `https://git.lab.example.com` and log in as **student** using **Student@123** as the password.
- 2.2. Click the **git/Controller Settings** project, and then go to **Settings > Webhooks**.
- 2.3. In the **URL** field, enter the **Webhook URL** value generated by automation controller for its **Apply Controller Settings** job template.
- 2.4. In the **Secret token** field, enter the **Webhook Key** value generated by automation controller for its **Apply Controller Settings** job template.

The screenshot shows the 'Webhook Settings' page for the 'Controller Settings' project in the 'git' namespace. The URL field contains 'https://controller.lab.example.com/api/v2/job_templates/9/gitlab/'. The Secret token field contains 'Pnj4SsWVSbClgHbcJLrQAivTPP2ZIRUazhYojDZLY0CV2yhjO'. The Trigger section has 'Push events' checked. A search bar at the top is empty.

git > Controller Settings > **Webhook Settings**

Search settings

Webhooks

Webhooks enable you to send notifications to web applications in response to events in a group or project. We recommend using an integration in preference to a webhook.

URL

https://controller.lab.example.com/api/v2/job_templates/9/gitlab/

URL must be percent-encoded if necessary.

Secret token

Pnj4SsWVSbClgHbcJLrQAivTPP2ZIRUazhYojDZLY0CV2yhjO

Use this token to validate received payloads. It is sent with the request in the X-Gitlab-Token HTTP header.

Trigger

Push events

Figure 8.22: URL and secret token for the webhook configuration

**Important**

The values in the URL and Secret key fields on your GitLab server must match the values provided by your automation controller, not those shown in the example.

- 2.5. Ensure that you select Push events.
- 2.6. Clear Enable SSL verification and click Add webhook when done.

**Note**

Because the servers in your lab environment are configured with TLS/SSL certificates that are not signed by a certificate authority trusted by the server, SSL verification fails.

If you do not clear the **Enable SSL verification** checkbox, it results in the following error message when you try to use the webhook:

```
Hook execution failed: SSL_connect returned=1 errno=0 state=error: certificate verify failed (unable to get local issuer certificate)
```

In production, configure the certificates and servers so that SSL verification can be enabled.

- 2.7. Scroll to the bottom of the page. Your webhook is displayed in the Project Hooks list.

Project Hooks (1)

https://controller.lab.example.com/api/v2/job_templates/9/gitlab/

Push Events SSL Verification: disabled

[Test](#) ▾ [Edit](#) [Delete](#)

Figure 8.23: Added webhook

- 2.8. (Optional). In the Test list for the project hook, select Push events to test the webhook configuration. When done, the following message is displayed at the top of the page:

```
Hook executed successfully: HTTP 202
```

3. At <https://controller.lab.example.com/api/v2/settings/all/>, browse the API settings for your automation controller and identify the name of the variable used to set the maximum number of forks per job. By default, automation controller sets the value of that variable to 200.
 - 3.1. Go to <https://controller.lab.example.com/api/v2/settings/all/>. If you have been logged out due to inactivity, log in again as the admin user with redhat as the password.

- 3.2. Scroll down until you find the variable used to set the maximum number of forks per job.



```
"MAX_WEBSOCKET_EVENT_RATE": 30,  
"SCHEDULE_MAX_JOBS": 10,  
"AWX_ANSIBLE_CALLBACK_PLUGINS": [],  
"DEFAULT_JOB_TIMEOUT": 0,  
"DEFAULT_JOB_IDLE_TIMEOUT": 0,  
"DEFAULT_INVENTORY_UPDATE_TIMEOUT": 0,  
"DEFAULT_PROJECT_UPDATE_TIMEOUT": 0,  
"ANSIBLE_FACT_CACHE_TIMEOUT": 0,  
"MAX_FORKS": 200,  
"LOG_AGGREGATOR_HOST": null,  
"LOG_AGGREGATOR_PORT": null,  
"LOG_AGGREGATOR_TYPE": null,  
"LOG_AGGREGATOR_USERNAME": "",  
"LOG_AGGREGATOR_PASSWORD": "",  
"LOG_AGGREGATOR_LOGGERS": [  
    "awx",  
    "activity_stream",  
    "job_events",  
    "system_tracking"  
,
```

Figure 8.24: Variable used to set the maximum number of forks per job

- 3.3. The variable used to set the maximum number of forks per job is MAX_FORKS. Use this variable in the following step.
4. Change the setting on your automation controller for the maximum number of forks per job to a value of 50. Test the webhook by making that change in your Git repository.

As the student user on the workstation machine, open a terminal window and clone the Git repository at https://git.lab.example.com/git/code_review.git to the /home/student/git-repos directory. Modify the ~/git-repos/code_review/vars/settings.yml variable file to set the maximum number of forks per job to 50. Save, commit, and push your changes to the remote repository. Use Set maximum forks as your commit message.

- 4.1. From a terminal, create the /home/student/git-repos directory if it does not already exist, and then change into it.

```
[student@workstation ~]$ mkdir -p ~/git-repos/  
[student@workstation ~]$ cd ~/git-repos/
```

- 4.2. Clone the https://git.lab.example.com/git/code_review.git repository and then change into it:

```
[student@workstation git-repos]$ git clone \  
> https://git.lab.example.com/git/code_review.git  
Cloning into 'code_review'...  
...output omitted...  
[student@workstation git-repos]$ cd code_review
```

- 4.3. In the ~/git-repos/code_review/vars/settings.yml variable file, update the controller_settings variable to add a new key-value pair using the previously identified variable name and the new value.

When you have edited it, the `~/git-repos/code_review/vars/settings.yml` variable file should contain the following content for the `controller_settings` variable:

```
---
```

```
controller_settings:
  - key: ANSIBLE_FACT_CACHE_TIMEOUT
    value: 86400
  - key: SESSION_COOKIE_AGE
    value: 3600
  - key: SESSIONS_PER_USER
    value: 2
  - key: MAX_FORKS
    value: 50
```

- 4.4. Save, commit, and push your changes to the remote repository. Use `Set maximum forks` for your commit message.

```
[student@workstation code_review]$ git add vars/settings.yml
[student@workstation code_review]$ git commit -m "Set maximum forks"
...output omitted...
[student@workstation code_review]$ git push
...output omitted...
```

5. Confirm that the webhook launched the `Apply Controller Settings` job template when you pushed your changes to the Git repository. From the automation controller web UI, verify that the job was successful and the maximum number of forks per job is now 50.
 - 5.1. In the browser window for automation controller, go to `Views > Jobs`.
 - 5.2. Click the `Apply Controller Settings` job. The output shows the change in the configuration for the `MAX_FORKS` variable.



Important

If you did the optional step 2.8, you get slightly different playbook output than is shown in the following example. This is because the push from the test already changed some variables configured in the `settings.yml` file.

Jobs > 2 - Apply Controller Settings

Output

Back to Jobs Details Output

Apply Controller Settings (Successful)

Plays 1 Tasks 1 Hosts 1 Elapsed 00:00:10

Stdout ▾

Vault password:

PLAY [Configure Automation Controller Settings] **** 18:28:42

TASK [Loop through settings] **** 18:28:42

changed: [localhost] => (item=Configuring the 'ANSIBLE_FACT_CACHE_TIMEOUT' setting.)

changed: [localhost] => (item=Configuring the 'SESSION_COOKIE_AGE' setting.)

changed: [localhost] => (item=Configuring the 'SESSIONS_PER_USER' setting.)

changed: [localhost] => (item=Configuring the 'MAX_FORKS' setting.)

PLAY RECAP **** 18:28:48

localhost : ok=1 changed=1 unreachable=0 failed=0 skipped=0 re

Figure 8.25: Webhooks launched the job template after the push

- 5.3. Go to **Settings** in automation controller.
- 5.4. Under the **Jobs** box select **Jobs settings**.
- 5.5. On the **Details** tab confirm that the value for the maximum number of forks per job is 50.

The screenshot shows the 'Details' tab of the 'Jobs' settings in the Ansible Automation Platform. It lists various configuration options with their current values:

When can extra variables contain Jinja templates?	template
Job execution path	/tmp
Job Event Maximum Websocket Messages Per Second	30
Maximum Scheduled Jobs	10
Default Job Timeout	0 seconds
Default Job Idle Timeout	0 seconds
Default Inventory Update Timeout	0 seconds
Default Project Update Timeout	0 seconds
Per-Host Ansible Fact Cache Timeout	86400 seconds
Maximum number of forks per job	50
Run Project Updates With Higher Verbosity	Off

Figure 8.26: Modified value for the maximum number of forks per job

Evaluation

On the workstation machine, change to the `student` user home directory and use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful. You *must* add, commit, and push any additional changes to the remote Git repository before grading.

```
[student@workstation ~]$ lab grade code-review
```

Finish

On the workstation machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish code-review
```

Summary

- You can automate Red Hat Ansible Automation Platform configuration by using Red Hat Certified Ansible Content Collections such as `ansible.controller`.
- You can use Git webhooks to automatically trigger job execution when you make changes to a project's Git repository.
- You can use Git webhooks with the `ansible.controller` Ansible Content Collection to automate configuration as code.
- Automation controller provides a REST API that enables you to control it without using the web UI.
- You can use Ansible Playbooks to launch job templates or perform other tasks or configuration of automation controller by using the `uri` module to access the automation controller API.
- Two steps are involved in launching a job template from the API: use the GET method to get the information you need, and then use the POST method to launch the job.

Chapter 9

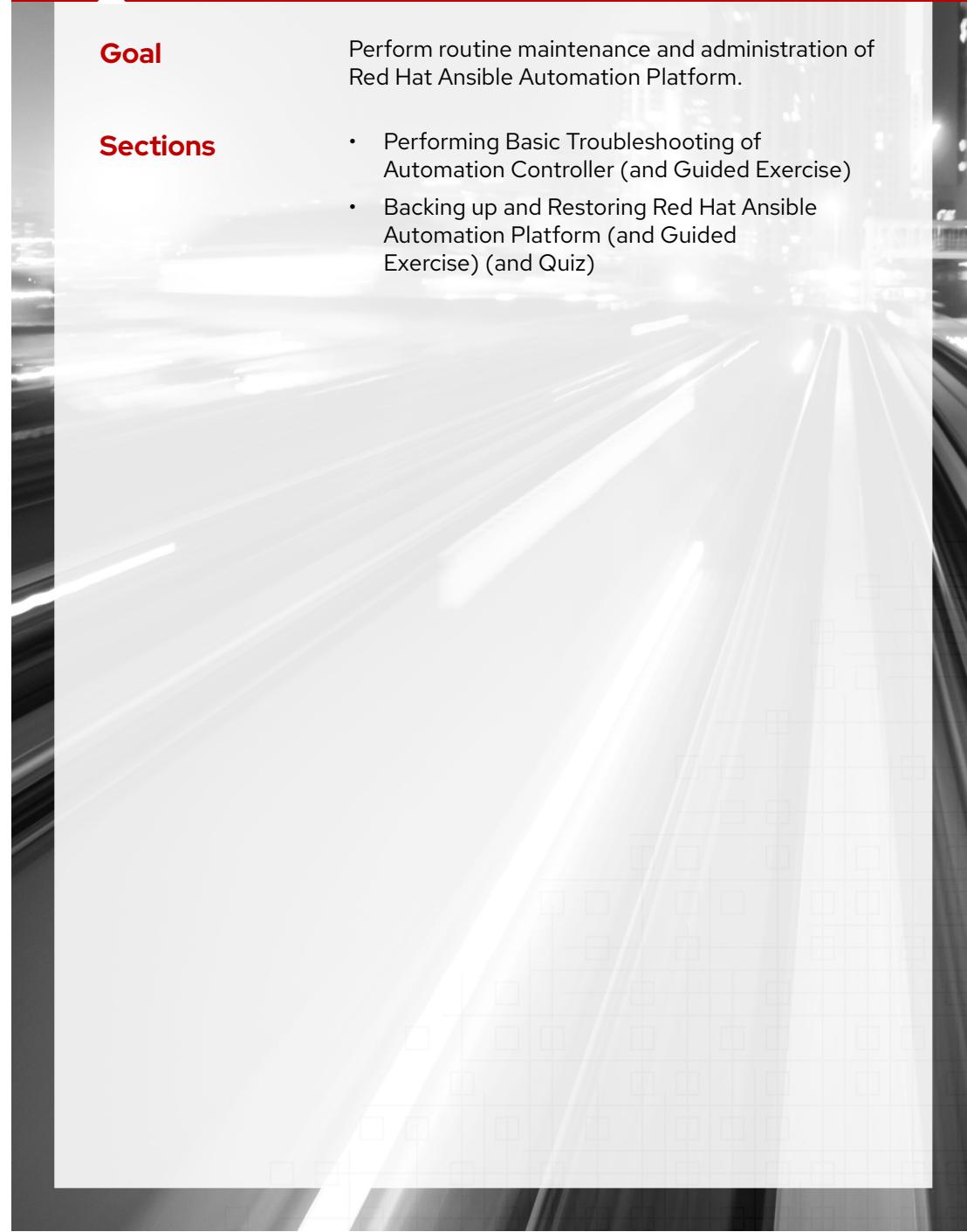
Maintaining Red Hat Ansible Automation Platform

Goal

Perform routine maintenance and administration of Red Hat Ansible Automation Platform.

Sections

- Performing Basic Troubleshooting of Automation Controller (and Guided Exercise)
- Backing up and Restoring Red Hat Ansible Automation Platform (and Guided Exercise) (and Quiz)



Performing Basic Troubleshooting of Automation Controller

Objectives

- Describe the low-level components of automation controller, locate and examine relevant log files, control its services, and perform basic troubleshooting.

Automation Controller Components

Automation controller is a web application made up of a number of cooperating processes and services. Four main network services are enabled, which start the rest of the components of automation controller:

- Nginx provides the web server that hosts the automation controller application and supports the web UI and the API.
- PostgreSQL is the database that stores most automation controller data, configuration, and history.
- Supervisord is a process control system that itself manages the various components of the automation controller application to perform operations such as schedule and run jobs, listen for callbacks from running jobs, and so on.
- Receptor provides an overlay network intended to ease the distribution of work across a large and dispersed collection of workers.

A fifth component also used by automation controller is the memcached memory object caching daemon, which is used as a local caching service.

These network services communicate with each other using normal network protocols. For a self-contained automation controller server, the main ports that need to be exposed outside the system are 80/tcp and 443/tcp, to allow clients to access the web UI and API.

However, the other services might also expose ports to external clients unless specifically protected. For example, the PostgreSQL service listens for connections from anywhere on 5432/tcp, and receptor needs access to certain ports for automation mesh communications. You can control access to these ports with a firewall, but it is important that you allow network communication from the hosts that need access to those ports, but deny access to hosts that should not be using those services.



Warning

This is one reason why setting good passwords for the PostgreSQL service in the inventory file used to install automation controller is important. These services can be contacted by internet clients directly by default, and weak passwords can leave them vulnerable to remote attack.

Starting, Stopping, and Restarting Automation Controller

Automation controller ships with `automation-controller-service`, an administrative utility script that can start, stop, and restart all the controller services running on the current controller node.

Chapter 9 | Maintaining Red Hat Ansible Automation Platform

This includes the message queue components, and the database if it is an integrated installation on that host. External databases must be explicitly managed by the administrator.

The `automation-controller-service` script is installed at `/usr/bin/automation-controller-service` and can be run as follows:

```
[root@control ~]# automation-controller-service status
● automation-controller.service - Automation Controller service
  Loaded: loaded (/etc/systemd/system/automation-controller.service; enabled;
  vendor preset: disabled)
    Active: active (exited) since Wed 2022-06-29 00:34:54 EDT; 1h 41min ago
      Process: 10217 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
      ...output omitted...

● redis.service - Redis persistent key-value database
  Loaded: loaded (/usr/lib/systemd/system/redis.service; enabled; vendor preset:
  disabled)
  Drop-In: /etc/systemd/system/redis.service.d
            └─limit.conf, override.conf
    Active: active (running) since Wed 2022-06-29 00:34:54 EDT; 1h 41min ago
      Process: 6128 ExecStop=/usr/libexec/redis-shutdown (code=exited, status=0/
  SUCCESS)
      ...output omitted...

● nginx.service - The nginx HTTP and reverse proxy server
  Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; vendor preset:
  disabled)
  Drop-In: /etc/systemd/system/nginx.service.d
            └─override.conf
    Active: active (running) since Wed 2022-06-29 00:34:54 EDT; 1h 41min ago
      Process: 10210 ExecStart=/usr/sbin/nginx (code=exited, status=0/SUCCESS)
      ...output omitted...

● supervisord.service - Process Monitoring and Control Daemon
  Loaded: loaded (/usr/lib/systemd/system/supervisord.service; enabled; vendor
  preset: disabled)
  Drop-In: /etc/systemd/system/supervisord.service.d
            └─override.conf
    Active: active (running) since Wed 2022-06-29 00:34:54 EDT; 1h 41min ago
      Process: 10200 ExecStart=/usr/bin/supervisord -c /etc/supervisord.conf
  (code=exited, status=0/SUCCESS)
      ...output omitted...

● receptor.service - Receptor
  Loaded: loaded (/usr/lib/systemd/system/receptor.service; enabled; vendor
  preset: disabled)
  Drop-In: /etc/systemd/system/receptor.service.d
            └─override.conf
    Active: active (running) since Tue 2022-06-28 23:49:41 EDT; 2h 26min ago
      ...output omitted...
```

To access the list of available options, run the `automation-controller-service` command without any options:

```
[root@control ~]# automation-controller-service
Usage: automation-controller-service start|stop|restart|status
```

The following example illustrates the effect of stopping the automation controller with `automation-controller-service`:

```
[root@control ~]# automation-controller-service stop
[root@controller ~]# automation-controller-service status
● automation-controller.service - Automation Controller service
  Loaded: loaded (/etc/systemd/system/automation-controller.service; enabled;
  vendor preset: disabled)
    Active: inactive (dead) since Thu 2022-06-30 06:09:14 EDT; 3s ago
      Process: 2614 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
    Main PID: 2614 (code=exited, status=0/SUCCESS)

...output omitted...

● receptor.service - Receptor
  Loaded: loaded (/usr/lib/systemd/system/receptor.service; enabled; vendor
  preset: disabled)
  Drop-In: /etc/systemd/system/receptor.service.d
            └─override.conf
    Active: active (running) since Thu 2022-06-30 01:13:06 EDT; 4h 56min ago
```

Compare that to the next example that illustrates the effect of starting the automation controller with `automation-controller-service`:

```
[root@control ~]# automation-controller-service start
[root@control ~]# automation-controller-service status
● automation-controller.service - Automation Controller service
  Loaded: loaded (/etc/systemd/system/automation-controller.service; enabled;
  vendor preset: disabled)
    Active: active (exited) since Thu 2022-06-30 06:11:41 EDT; 2s ago
      Process: 3054 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
    Main PID: 3054 (code=exited, status=0/SUCCESS)

...output omitted...

● receptor.service - Receptor
  Loaded: loaded (/usr/lib/systemd/system/receptor.service; enabled; vendor
  preset: disabled)
  Drop-In: /etc/systemd/system/receptor.service.d
            └─override.conf
    Active: active (running) since Thu 2022-06-30 01:13:06 EDT; 4h 58min ago

...output omitted...
```

**Important**

`automation-controller-service` does not start or stop `receptor.service`. This is to allow you to reload or restart `receptor` on your nodes without restarting the control plane, or vice versa.

To start, stop, or restart `receptor.service`, use the `systemctl` command.

Supervisord Components

`Supervisord` is a process control system often used to control Django-based applications such as automation controller. It is used to manage and monitor long-running processes or daemons, and to automatically restart them as needed. In automation controller, `supervisord` manages important components of the automation controller application itself.

You can use the `supervisorctl status` command to see the list of automation controller processes controlled by the `supervisord` service:

```
[root@control ~]# supervisorctl status
master-event-listener                  RUNNING    pid 10997, uptime 0:02:03
tower-processes:awx-callback-receiver   RUNNING    pid 10999, uptime 0:02:03
tower-processes:awx-daphne              RUNNING    pid 11001, uptime 0:02:03
tower-processes:awx-dispatcher          RUNNING    pid 10998, uptime 0:02:03
tower-processes:awx-rsyslogd            RUNNING    pid 11032, uptime 0:02:00
tower-processes:awx-uwsgi               RUNNING    pid 11000, uptime 0:02:03
tower-processes:awx-wsbroadcast         RUNNING    pid 11002, uptime 0:02:03
```

As you can see in the preceding output, `supervisord` controls a number of processes owned by the `awx` user.

Automation Controller Configuration and Log Files

Configuration Files

The main configuration files for automation controller are kept in the `/etc/tower` directory. These include settings files for the automation controller application, the TLS certificate for `nginx`, and other key files.

Perhaps the most important of these files for the automation controller application is the `/etc/tower/settings.py` file, which specifies the locations for job output, project storage, and other directories.

The other individual services might have service-specific configuration files elsewhere on the system, such as the `/etc/nginx` files used by the web server.

Log Files

The automation controller application log files are stored in one of two centralized locations:

- `/var/log/tower/`
- `/var/log/supervisor/`

Automation controller server errors are logged in the `/var/log/tower/` directory. Some key files in the `/var/log/tower/` directory include:

Chapter 9 | Maintaining Red Hat Ansible Automation Platform

- `/var/log/tower/tower.log`: The main log file for the automation controller application.
- `/var/log/tower/task_system.log`: The log file that captures the logs of tasks that the controller is running in the background, such as adding cluster instances and logs related to information gathering as well as processing for analytics, and so on.

The `/var/log/supervisor/` directory stores log files for services, daemons, and applications managed by `supervisord`. The `supervisord.log` file in this directory is the main log file for the service that controls all these daemons. The other files contain log information about the activity of those daemons.

Automation controller can also send detailed logs to external log aggregation services. Log aggregation can offer insight into automation controller technical trends or usage. The data can be used to monitor for anomalies, analyze events, and correlate events. Splunk, Elastic stack (formerly ELK stack), Loggly, and Sumologic are all log aggregation and data analysis systems that can be used with automation controller.

For more information on how to configure such services, see the References section.



Important

This discussion has focused on looking at the log files to troubleshoot problems with the automation controller server itself.

If you encounter errors running playbooks that do not appear to be related to actual errors in the automation controller configuration, remember to look at the output of your launched jobs in the automation controller web UI or the API.

Other Automation Controller Files

A number of other key files for automation controller are kept in the `/var/lib/awx` directory. This directory includes:

- `/var/lib/awx/projects`: This is the main directory for projects. For projects that use source control, automation controller clones the project to this directory. The name of each directory includes the identification number for the project and the project name.
- `/var/lib/awx/job_status`: Job status output from playbooks is stored in this directory.

Common Troubleshooting Scenarios

Problems Running Playbooks

If you cannot run playbooks due to playbook errors, try the following suggestions:

- Are you authenticating as the user currently running the commands? If not, review how the `username` has been set up or pass the `--user=username` or `-u username` options to specify a user.
- Is your YAML file correctly indented? The indentation level is significant in YAML. Ensure you align your white space correctly.

You can use `yamllint` to test the syntax of your playbook.

You can also use `--syntax-check` with `ansible-navigator` to identify syntax errors and fix them.

```
[user@demo ~]$ ansible-navigator run -m stdout test_playbook.yml --syntax-check
```

- Red Hat Ansible Automation Platform 2.2 also provides `ansible-lint` as a tech preview tool to review your playbooks for possible issues.
- Items beginning with a dash (-) are considered list items or plays. Items with the format of key: value operate as hashes or dictionaries. Ensure that you do not have extra or missing - plays in your file.
- Review your license status and the number of unique hosts that the automation controller server manages.

If the license has expired, or too many hosts are registered, launching jobs might not be possible.

Problems Connecting to Your Host

If you encounter connectivity issues when running playbooks, try the following suggestions:

- Verify that you can establish an SSH or WinRM connection with the managed host. Ansible depends upon SSH (or WinRM for Microsoft Windows systems) to access the servers you are managing.
- Review your inventory file. Review the hostnames and IP addresses.

Playbooks Do Not Appear in the List of Job Templates

If your playbooks are not showing up in the job template list, then review the playbook's YAML syntax and make sure that it can be parsed by Ansible.

Playbook Stays in Pending State

When you are trying to run a job and it stays in the Pending state, try the following suggestions:

- Ensure that the automation controller server has enough memory available and that the services governed by `supervisord` are running. Run the `supervisorctl status` command.
- Ensure that the partition where the `/var/` directory is located has more than 1 GB of space available. Jobs cannot complete when there is insufficient free space in the `/var/` directory.
- Restart the automation controller infrastructure using the `automation-controller-service restart` command.

Error: Provided Hosts List Is Empty

If you encounter the error message `Skipping: No Hosts Matched` when you are trying to run a playbook through automation controller, review these possibilities:

- Review and make sure that the host patterns used by the `hosts` declaration in your play matches the group or hostnames in the inventory. The host patterns are case-sensitive.
- Make sure that your group names have no spaces. Modify them to use underscores or no spaces to ensure that the groups are correctly recognized.
- If you have specified a limit in the job template, make sure that it is a valid limit and that it matches something in your inventory.

Performing Command-line Management

Automation controller ships with the `awx-manage` command-line utility, which can be used to access detailed internal automation controller information. The `awx-manage` command must be run as `root` or as the `awx` (automation controller) user. This utility is most commonly used to reset the automation controller's `admin` password.

Changing the Automation Controller Admin Password

The password for the built-in automation controller System Administrator account, `admin`, is initially set when the automation controller server is installed. The `awx-manage` command provides a way to change the administrator password from the command line. To do this, as the `root` or `awx` user on the automation controller server, use the `changepassword` option:

```
[root@control ~]# awx-manage changepassword admin
Changing password for user 'admin'
Password: new_password
Password (again): new_password
Password changed successfully for user 'admin'
```



Note

You can also change the password for the `admin` user through the automation controller web UI.

You can also create a new automation controller superuser, with administrative privileges if needed. To create a new superuser, you can use `awx-manage` with the `createsuperuser` option.

```
[root@control ~]# awx-manage createsuperuser
Username (leave blank to use 'root'): admin3
Email address: admin@demo.example.com
Password: new_password
Password (again): new_password
Superuser created successfully.
```



References

Troubleshooting the controller

<https://docs.ansible.com/automation-controller/latest/html/administration/troubleshooting.html>

Logging Aggregator Services

<https://docs.ansible.com/automation-controller/latest/html/administration/logging.html#logging-aggregator-services>

► Guided Exercise

Performing Basic Troubleshooting of Automation Controller

Troubleshoot and identify a problem with the configuration of a playbook for a new job template.

Outcomes

- Fix a problem in which automation controller does not show a playbook that you want to add to a new job template.

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that automation controller is installed and configured with any necessary resources created in previous exercises.

```
[student@workstation ~]$ lab start admin-troubleshoot
```

Instructions

- ▶ **1.** In this exercise, you need to create a new job template named `Job_Facts` that uses the `refresh_fact_cache.yml` playbook from the existing `Job_facts` project to refresh the fact cache for managed hosts in the `Dev` inventory.
Navigate to <https://controller.lab.example.com> and log in as the `admin` user with `redhat` as the password.
- ▶ **2.** Make sure that the per-host Ansible fact cache timeout for automation controller is set for exactly one day (86400 seconds). This ensures that the facts are cached for hosts but also expire if the facts have not been refreshed recently.
 - 2.1. Navigate to **Settings** in the automation controller.
 - 2.2. In the **Settings** window, under the **Jobs** category, click **Jobs settings**.
 - 2.3. Set **Per-Host Ansible Fact Cache Timeout** to 86400 seconds (one day), and then click **Save**.
- ▶ **3.** Create a new job template named `Job_Facts`.
 - 3.1. Navigate to **Resources > Templates** and then click **Add > Add job template** to open the **Create New Job Template** page.
 - 3.2. Attempt to configure the new job template with the following details:

Field	Value
Name	Job Facts
Description	Gather ansible facts from cluster
Job Type	Run
Inventory	Dev
Project	Job facts
Playbook	refresh_fact_cache.yml
Credentials	Developers

When you try to do this, the `refresh_fact_cache.yml` playbook is not available. You need to troubleshoot this problem.

- ▶ 4. The most likely cause for this issue is that the playbook is not present in the Git repository. Make sure that the `refresh_fact_cache.yml` playbook is present in the `job-facts` repository.
 - 4.1. From a terminal, create the `/home/student/git-repos` directory if it does not already exist, and then change into it.

```
[student@workstation ~]$ mkdir -p ~/git-repos/  
[student@workstation ~]$ cd ~/git-repos/
```

- 4.2. Clone the `https://git.lab.example.com/git/job-facts.git` repository and then review the contents.

Verify that the `refresh_fact_cache.yml` playbook is present.

```
[student@workstation git-repos]$ git clone \  
> https://git.lab.example.com/git/job-facts.git  
Cloning into 'job-facts'...  
...output omitted...  
[student@workstation git-repos]$ cd job-facts  
[student@workstation job-facts]$ ll  
total 12  
-rw-rw-r--. 1 student student 65 Jul 7 20:41 refresh_fact_cache.yml  
[student@workstation job-facts]$
```

- ▶ 5. Because the playbook exists, that is not the problem. Playbooks that exist but do not appear in the `Job Template` list are most likely formatted incorrectly. Test the `refresh_job_facts.yml` playbook with the `ansible-lint` command, and repair any reported issues.
 - 5.1. Run the `ansible-lint` command on the file to search for formatting errors.

```
[student@workstation job-facts]$ ansible-lint -p refresh_fact_cache.yml
...output omitted...
refresh_fact_cache.yml:2:3: syntax-check 'host' is not a valid attribute for a
Play
...output omitted...
```

- 5.2. Edit the `refresh_fact_cache.yml` playbook and fix the mistake. The play in the playbook has a typing mistake. After you edit it, the playbook should have the following contents:

```
---
- name: Refresh fact cache
  hosts: all
  gather_facts: true
```

- 5.3. Use the `git add` command to stage the changes, and then commit the changes using `Fixed a mistake in hosts` as the commit message. Push the locally committed changes to the remote server.

```
[student@workstation job-facts]$ git add refresh_fact_cache.yml
[student@workstation job-facts]$ git commit -m "Fixed a mistake in hosts"
[main e76b744] Fixed a mistake in hosts.
 1 file changed, 1 insertion(+), 1 deletion(-)
[student@workstation job-facts]$ git push -u origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
...output omitted...
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

▶ **6.** Synchronize the project to the automation controller.

- 6.1. From the automation controller web UI, navigate to **Resources > Projects**.
- 6.2. Click the **Sync Project** icon for the **Job facts** project. Wait until the synchronization completes successfully.

▶ **7.** Attempt to add the `refresh_fact_cache.yml` playbook to the job template again.

- 7.1. Navigate to **Resources > Templates** and click **Add > Add job template** to open the **Create New Job Template** page.
- 7.2. Enter the following details. This time it should work. Click **Save** when you are finished.

Field	Value
Name	Job Facts
Description	Gather ansible facts from cluster
Job Type	Run
Inventory	Dev
Project	Job facts
Playbook	refresh_fact_cache.yml
Credentials	Developers

7.3. Select the **Enable Fact Storage** option and then click **Save**.

7.4. Click **Launch**. The job should run successfully.

Finish

On the **workstation** machine, change to the **student** user home directory and use the **lab** command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish admin-troubleshoot
```

Backing up and Restoring Red Hat Ansible Automation Platform

Objectives

- Back up and restore the automation controller and automation hub databases and configuration files.

Backing up Red Hat Ansible Automation Platform

The ability to manually back up and restore a Red Hat Ansible Automation Platform installation is integrated into the Ansible Automation Platform installation software. After you make a backup, you can use other tools to make sure that the backup files are stored in a safe and secure location.

The backup procedure uses the same `setup.sh` script and the same `inventory` file that you used to install your Ansible Automation Platform environment.



Important

If you deleted the original installation directory, you can download and install or unpack the installation files for the same version of Red Hat Ansible Automation Platform that you have installed.

If you do, then you must edit the `inventory` file in the installation directory to match the one that you used for installation. (That is, it should match your current installation settings.)

To start the backup, run the `./setup.sh -b` command in the installation directory as the `root` user.

The script creates a backup archive file named `automation-platform-DATE.tar.gz`, where `DATE` is the current date and time in the `date +%F-%T` format. The backup script creates the backup archive in the same directory as the `setup.sh` script and also creates a symbolic link named `automation-platform-backup-latest.tar.gz` that points to the most recent backup archive.

The backup archive consists of additional archive files that vary based on your installation.

- The backup script creates an archive for each controller, such as `control1.example.com.tar.gz` and `control2.example.com.tar.gz`. These archives contain `conf` and `projects` directories. The `conf` directory contains configuration files from the `/etc/tower` directory. The `projects` directory contains project lock files for projects that have been synchronized by the automation controller host. This `projects` directory does not contain any files from the projects themselves.
- The backup script also creates the `common.tar.gz` archive. This archive contains the `SECRET_KEY`, `tower.db`, and `version` files from the database. The `tower.db` file is a PostgreSQL database dump file. The `version` file displays the version of Ansible Automation Platform at the time of the backup.

- If your installation included private automation hub, then the backup script creates the `automationhub.tar.gz` archive. This archive contains configuration files for private automation hub.

Ensure that you have enough free space available before running the backup.

This procedure backs up the configuration of your Ansible Automation Platform servers, but it does not back up log files or installed programs.



Warning

The manual backup procedure using `setup.sh -b` only creates the backup archive file. You are responsible for setting up a procedure to periodically create a backup and store the backup archive in a safe place.

Backup Procedure

The following procedure creates a new backup of your Ansible Automation Platform environment.

1. As the root user, change to the installation directory.

In the following example, the installer has been extracted to the `/opt/ansible-automation-platform/installer` directory.

```
[root@host ~]# cd /opt/ansible-automation-platform/installer
```

2. As the root user, run the `setup.sh` script with the `-b` option to start the backup process.

```
[root@host installer]# ./setup.sh -b
...output omitted...

PLAY RECAP ****
control1.example.com : ok=59  changed=33  ...  failed=0  skipped=85  ...
hub.example.com      : ok=37  changed=14  ...  failed=0  skipped=67  ...
localhost            : ok=2   changed=0   ...  failed=0  skipped=0  ...

The setup process completed successfully.
[warn] /var/log/tower does not exist. Setup log saved to backup.log.
```

3. List the `automation-platform-*` files in the current directory to ensure that the backup archive exists and is accessible.

```
[root@host installer]# ls -l automation-platform-*
-rw-----. ... automation-platform-backup-2022-06-30-10:02:40.tar.gz
lrwxrwxrwx. ... automation-platform-backup-latest.tar.gz -> /opt/ansible-automation-platform./automation-platform-backup-2022-06-30-10:02:40.tar.gz
```



Note

The `automation-platform-backup-latest.tar.gz` symbolic link points to the most recent backup archive. When performing a recovery, the recovery script uses this symbolic link by default.

Restoring Ansible Automation Platform from Backup

Run the `setup.sh` script with the `-r` option to restore Ansible Automation Platform from a backup archive. You need the backup archive, the `setup.sh` script for the same version of Ansible Automation Platform that was used to create the backup, and the `inventory` file for the installer.

If you have multiple backup archives available, make sure that the `automation-platform-backup-latest.tar.gz` symbolic link points to the exact backup file from which you want to restore. If you need to use an earlier backup file, delete the existing `automation-platform-backup-latest.tar.gz` symbolic link and create a new link pointing to the correct backup archive.



Warning

When restoring a backup, make sure you use the same version of Ansible Automation Platform that you used to create the backup.

If you do not remember the exact version you used to create the backup, the `common.tar.gz` archive in the backup archive file contains a file named `version` that specifies that software version.

Restoration Procedure

The following procedure demonstrates how to restore the Ansible Automation Platform infrastructure from an existing backup archive.

- As the `root` user, change to the extracted archive directory. In the following example, the installation archive has been extracted and moved to the `/opt/ansible-automation-platform/installer` directory. The `/opt/ansible-automation-platform/installer` directory contains the backup archive file.

```
[root@host ~]# cd /opt/ansible-automation-platform/installer
```

- Ensure that the backup archive is in that directory.

```
[root@host installer]# ls -l automation-platform-*
-rw----- . . . automation-platform-backup-2022-06-30-10:02:40.tar.gz
lrwxrwxrwx . . . automation-platform-backup-latest.tar.gz -> /opt/ansible-automation-platform/./automation-platform-backup-2022-06-30-10:02:40.tar.gz
```



Important

The symbolic link `automation-platform-backup-latest.tar.gz` points to the *latest* backup archive. If you need to restore from an earlier backup, you have to recreate that link so that it points to the correct archive.

- As the `root` user, run `setup.sh -r` to start the recovery process.



Important

Remember to use the `-r` option with the `setup.sh` script. If you forget, then the `setup.sh` script performs a clean installation but does not restore your backup.

If that happens, wait until the installation process finishes and then restore the backup.

```
[root@host installer]# ./setup.sh -r
...output omitted...
PLAY RECAP ****
control2.lab.com : ok=53    changed=16    ...    failed=0    skipped=56    ...    ignored=2
control1.lab.com : ok=60    changed=21    ...    failed=0    skipped=85    ...    ignored=2
db.example.com   : ok=6     changed=6     ...    failed=0    skipped=0     ...    ignored=0
hub.example.com  : ok=35    changed=13    ...    failed=0    skipped=56    ...    ignored=1

The setup process completed successfully.
[warn] /var/log/tower does not exist. Setup log saved to restore.log.
```

4. Log in to the automation controller web UI and verify that automation controller has been restored correctly from backup.
5. Log in to the private automation hub web UI and verify that automation hub has been restored correctly from backup.



References

Ansible Automation Controller Administration Guide

https://docs.ansible.com/automation-controller/latest/html/administration/backup_restore.html

► Guided Exercise

Backing up and Restoring Red Hat Ansible Automation Platform

Perform a backup of the Red Hat Ansible Automation Platform databases and configuration files.

Outcomes

- Back up the existing Ansible Automation Platform installation.
- Restore the Ansible Automation Platform configuration and database from an existing backup.

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command downloads and extracts the installation archive to the `/home/student/aap2.2-bundle` directory. It also replaces the `inventory` file in the extracted archive with the `inventory` file used during the initial installation.

```
[student@workstation ~]$ lab start admin-recovery
```

Instructions

- 1. Back up your existing Red Hat Ansible Automation Platform installation.
- 1.1. On `workstation`, open a terminal and use the `sudo` command to change to the `root` user, using `student` as the password.

```
[student@workstation ~]$ sudo -i  
[sudo] password for student: student  
[root@workstation ~]#
```

- 1.2. Change to the `/home/student/aap2.2-bundle` directory.

```
[root@workstation ~]# cd /home/student/aap2.2-bundle/
```

- 1.3. Run the `./setup.sh -b` command to start the back up process. The back up only takes a few minutes to complete.

```
[root@workstation aap2.2-bundle]# ./setup.sh -b  
..output omitted...  
PLAY RECAP ****  
controller.lab.example.com : ok=59    changed=33    ...    failed=0    skipped=85    ...  
hub.lab.example.com      : ok=37    changed=14    ...    failed=0    skipped=67    ...
```

Chapter 9 | Maintaining Red Hat Ansible Automation Platform

```
localhost : ok=2    changed=0 ... failed=0  skipped=0 ...
```

The setup process completed successfully.
[warn] /var/log/tower does not exist. Setup log saved to backup.log.

- 1.4. List the contents of the current directory to verify that the backup archive exists. In addition to the archive, the `automation-platform-backup-latest.tar.gz` symbolic link points to the latest archive file.

```
[root@workstation aap2.2-bundle]# ls -l
automation-platform-backup-2022-06-28-17:29:16.tar.gz
automation-platform-backup-latest.tar.gz
backup.log
bundle
collections
group_vars
images
inventory
licenses
README.md
setup.sh
```

- ▶ 2. Change the password for the automation controller `admin` user to `redhat2` using the `awx-manage` command.

**Important**

This step is for demonstration purposes only and is not part of a normal backup and restoration procedure.

This step demonstrates that any changes made after performing a back up are not restored by the backup archive.

- 2.1. Log in to the `controller.lab.example.com` server as the `awx` user. Because of existing SSH keys, you do not need to enter a password.

```
[root@workstation aap2.2-bundle]# ssh awx@controller.lab.example.com
...output omitted...
[awx@controller ~]$
```

- 2.2. Run the `awx-manage` command to change the password for the `admin` user. Use `redhat2` as the new password.

```
[awx@controller ~]$ awx-manage changepassword admin
Changing password for user 'admin'
Password: redhat2
Password (again): redhat2
Password changed successfully for user 'admin'
```

- 2.3. Exit from the `controller.lab.example.com` server.

```
[awx@controller ~]$ exit  
logout  
Connection to controller.lab.example.com closed.
```

- 2.4. Confirm that you can log in to the automation controller web UI using the new password. Navigate to `https://controller.lab.example.com` and log in as the `admin` user with `redhat2` as the password.
- 2.5. After successfully logging in, click `admin > Logout`.

- ▶ 3. Restore the Ansible Automation Platform configuration and database.
- 3.1. As the `root` user on `workstation`, change to the `/home/student/aap2.2-bundle` directory.

```
[root@workstation ~]# cd /home/student/aap2.2-bundle/
```

- 3.2. Run the `./setup.sh -r` command to restore the backup. The restoration operation only takes a few minutes to complete.

```
[root@workstation aap2.2-bundle]# ./setup.sh -r  
...output omitted...  
PLAY RECAP ****  
controller.lab.example.com : ok=60    changed=21    ... failed=0    skipped=85    ...  
db.lab.example.com       : ok=6      changed=6     ... failed=0    skipped=0    ...  
hub.lab.example.com     : ok=35    changed=13    ... failed=0    skipped=56    ...  
  
The setup process completed successfully.  
[warn] /var/log/tower does not exist. Setup log saved to restore.log.
```



Important

Remember to use the `-r` option with the `setup.sh` script. If you forget, then the `setup.sh` script performs a clean installation but does not restore your backup.

If that happens, wait until the installation process finishes and then repeat this step with the `-r` option to restore the backup.

- 3.3. Exit from the `root` session.

```
[root@workstation aap2.2-bundle]# exit
```

- ▶ 4. Confirm that the restoration process reverted the password for the automation controller `admin` user. It should be set to whatever is specified by the `inventory` file.
- 4.1. Navigate to `https://controller.lab.example.com` and attempt to log in as the `admin` user with `redhat2` as the password. The log in attempt fails.
 - 4.2. Attempt to log in again as the `admin` user with `redhat` as the password. The log in attempt succeeds.

Finish

On the **workstation** machine, change to the **student** user home directory and use the **lab** command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish admin-recovery
```

► Quiz

Maintaining Red Hat Ansible Automation Platform

Choose the correct answers to the following questions:

- ▶ 1. **Which network services should you review in your automation controller when performing troubleshooting?**
 - a. Apache HTTP Server, PostgreSQL, receptor, and supervisord.
 - b. Nginx, MySQL, receptor, and supervisord.
 - c. Nginx, PostgreSQL, receptor, and supervisord.
 - d. Apache HTTP Server, MySQL, receptor, and supervisord.
- ▶ 2. **In which two operating system directories can you find log files for automation controller? (Choose two.)**
 - a. /var/log/tower/
 - b. /var/log/controller/
 - c. /var/log/awx/
 - d. /var/log/supervisor/
- ▶ 3. **Which statement is NOT true about the location of automation controller files?**
 - a. The main configuration files for automation controller are kept in the /etc/automation-controller directory.
 - b. The main configuration files for automation controller are kept in the /etc/tower directory.
 - c. The main directory for the automation controller projects is the /var/lib/awx/projects directory.
 - d. The /etc/tower/settings.py file specifies the locations for job output, project storage, and other directories.
- ▶ 4. **Which command can you use to see the list of automation controller processes controlled by the supervisord service?**
 - a. automation-controller-service status
 - b. awx-controller-service status
 - c. supervisorctl status
 - d. awx-supervisor status

► **5. In which two situations can you use the awx-manage command to perform troubleshooting tasks? (Choose two.)**

- a. You can use the `awx-manage changepassword admin` command to change the `admin` user password.
- b. You can use the `awx-manage start` command to restart the services on automation controller.
- c. You can use the `awx-manage status` command to review the status of the services on automation controller.
- d. You can use the `awx-manage createsuperuser` command to create a new automation controller superuser.

► **6. Which statement is not true about backing up Red Hat Ansible Automation Platform?**

- a. The procedure uses the same `setup.sh` script and the same inventory file that you used to do the installation.
- b. You have to run the backup command in the installer directory as the `root` user.
- c. You cannot perform a backup procedure if you deleted the original installation directory.
- d. The backup script creates a backup archive for each automation controller.

► **7. Which three statements describe what is required to restore an Ansible Automation Platform deployment from backup? (Choose three.)**

- a. The backup archive created for your Ansible Automation Platform installation.
- b. At least one private automation hub server in your architecture.
- c. To use the `-b` option with the `setup.sh` script when running the restore.
- d. The `setup.sh` script for the same version of Ansible Automation Platform that was used to create the backup.
- e. The inventory file for the installer.

► Solution

Maintaining Red Hat Ansible Automation Platform

Choose the correct answers to the following questions:

- ▶ **1. Which network services should you review in your automation controller when performing troubleshooting?**
 - a. Apache HTTP Server, PostgreSQL, receptor, and supervisord.
 - b. Nginx, MySQL, receptor, and supervisord.
 - c. Nginx, PostgreSQL, receptor, and supervisord.
 - d. Apache HTTP Server, MySQL, receptor, and supervisord.
- ▶ **2. In which two operating system directories can you find log files for automation controller? (Choose two.)**
 - a. /var/log/tower/
 - b. /var/log/controller/
 - c. /var/log/awx/
 - d. /var/log/supervisor/
- ▶ **3. Which statement is NOT true about the location of automation controller files?**
 - a. The main configuration files for automation controller are kept in the /etc/automation-controller directory.
 - b. The main configuration files for automation controller are kept in the /etc/tower directory.
 - c. The main directory for the automation controller projects is the /var/lib/awx/projects directory.
 - d. The /etc/tower/settings.py file specifies the locations for job output, project storage, and other directories.
- ▶ **4. Which command can you use to see the list of automation controller processes controlled by the supervisord service?**
 - a. automation-controller-service status
 - b. awx-controller-service status
 - c. supervisorctl status
 - d. awx-supervisor status

► **5. In which two situations can you use the awx-manage command to perform troubleshooting tasks? (Choose two.)**

- a. You can use the `awx-manage changepassword admin` command to change the `admin` user password.
- b. You can use the `awx-manage start` command to restart the services on automation controller.
- c. You can use the `awx-manage status` command to review the status of the services on automation controller.
- d. You can use the `awx-manage createsuperuser` command to create a new automation controller superuser.

► **6. Which statement is not true about backing up Red Hat Ansible Automation Platform?**

- a. The procedure uses the same `setup.sh` script and the same inventory file that you used to do the installation.
- b. You have to run the `backup` command in the installer directory as the `root` user.
- c. You cannot perform a backup procedure if you deleted the original installation directory.
- d. The `backup` script creates a backup archive for each automation controller.

► **7. Which three statements describe what is required to restore an Ansible Automation Platform deployment from backup? (Choose three.)**

- a. The backup archive created for your Ansible Automation Platform installation.
- b. At least one private automation hub server in your architecture.
- c. To use the `-b` option with the `setup.sh` script when running the `restore`.
- d. The `setup.sh` script for the same version of Ansible Automation Platform that was used to create the backup.
- e. The inventory file for the installer.

Summary

- Automation controller integrates four primary network services as its components: Nginx as the web server for the application, PostgreSQL as its database, Supervisord as the process control system, and Receptor as an overlay network.
- Use the `automation-controller-service` command to start, stop, or restart automation controller services, and `systemctl` to start, stop, or restart `receptor` services.
- Automation controller stores log files in the `/var/log/tower` and `/var/log/supervisor` directories.
- Automation controller can also send detailed logs to external log aggregation services.
- The Red Hat Ansible Automation Platform installation script (`setup.sh`) provides the `-b` option to perform a backup and the `-r` option to perform a recovery.

Chapter 10

Getting Insights into Automation Performance

Goal

Get information from Red Hat Insights for Red Hat Ansible Automation Platform to evaluate the performance of your Ansible automation and identify possible ways to improve it.

Sections

- Gathering Data for Cloud-based Analysis (and Quiz)
- Getting Insights into Automation Performance (and Quiz)
- Evaluating Performance with Automation Analytics (and Quiz)
- Producing Reports from Automation Analytics (and Quiz)

Gathering Data for Cloud-based Analysis

Objectives

- Send data to the Red Hat Insights for Red Hat Ansible Automation Platform and automation analytics cloud services in order to analyze and improve your use of automation.

Introducing Red Hat Hybrid Cloud Console Services

Red Hat provides a number of hosted services on <https://console.redhat.com> that can help you better manage your Red Hat Ansible Automation Platform infrastructure. The following two Red Hat Hybrid Cloud Console services are relevant to gather data for cloud-based analysis:

- Insights for Ansible Automation Platform
- Automation analytics

Insights for Ansible Automation Platform enables you to keep track of warnings about security vulnerabilities and the security risk levels for the members of your automation infrastructure. You can also use Insights for Ansible Automation Platform to determine the changes that each system has undergone over time, or how far it has moved away from your baselines.

Automation analytics helps to provide better insight into the performance of your automation infrastructure. You can use it to analyze how you use automation and what modules, playbooks, and workflows you most frequently use. Automation analytics is also useful to estimate the return on investment of current or future automation projects. You can even develop reports in automation analytics for others to review.



Important

You need to satisfy the following requirements to use automation analytics and Insights for Ansible Automation Platform:

- A Red Hat Hybrid Cloud Console account.
- A Red Hat Ansible Automation Platform subscription.
- Your automation controllers must be able to reach <https://console.redhat.com/>.

Collecting Data for Cloud Services

Before Red Hat Hybrid Cloud Console can report data about your automation controller clusters, you need to configure those systems to send data to it. Information about what data is sent and how it is managed and secured is available from the references at the end of this section.

You can enable data gathering from your automation controller by using the following procedure:

- Log in to the web UI of your automation controller as the **admin** user and navigate to **Settings**.
- In the **System** panel, click **Miscellaneous System settings** and then click **Edit**.
- Set **Gather data for Insights for Ansible Automation Platform** to **On** to activate data gathering.

- Enter your customer portal credentials in the **Red Hat customer username** and **Red Hat customer password** fields. If you use a corporate account, then everybody in your organization can access reports for that automation controller cluster from the Red Hat Hybrid Cloud Console.

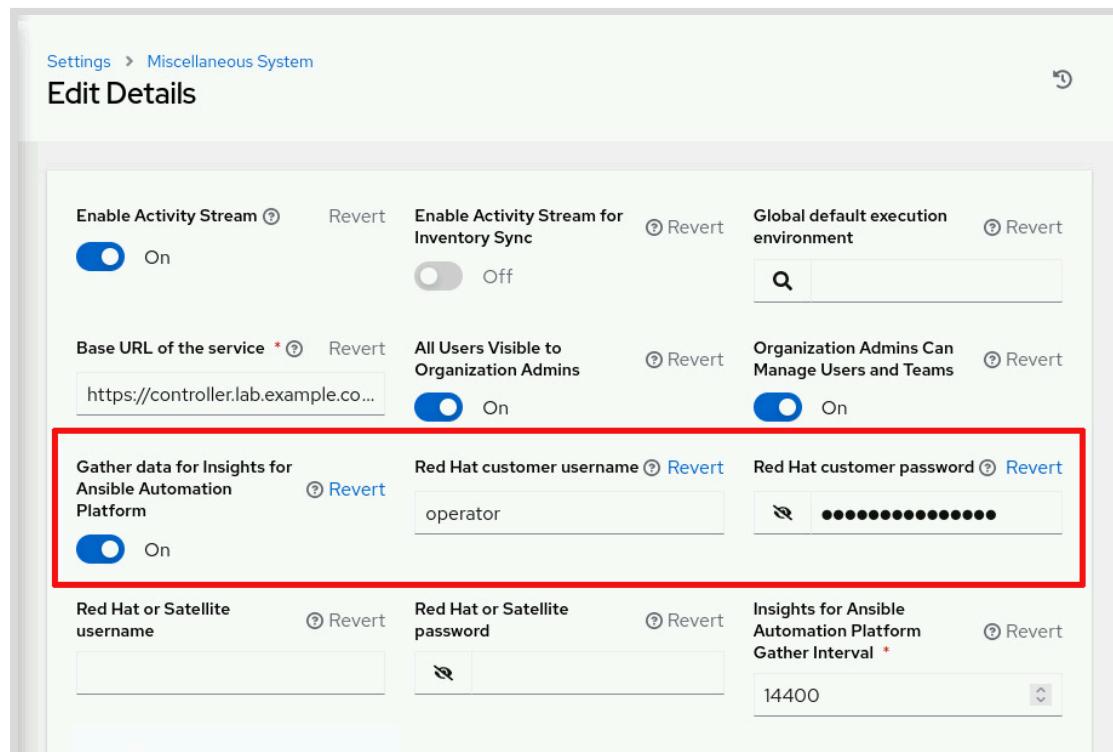


Figure 10.1: Configuring data gathering

- By default, automation controller sends data to Red Hat Insights every four hours (14400 seconds). Use the **Insights for Ansible Automation Platform Gather Interval** field to change that parameter. The value is in seconds and cannot be less than 1800 (30 minutes).
- Click Save when done.

Registering Managed Hosts with Insights for Ansible Automation Platform

The Advisor tool provided by Red Hat Insights can identify issues with your managed hosts and automatically generate Ansible Playbooks to remediate those issues. To help your Red Hat Enterprise Linux managed hosts gather the information to do this effectively, you can also register them with Red Hat Insights.

To register your managed hosts with Red Hat Insights, download and install the `insights-client` package on your managed host. If it is already registered to get updates through a Customer Portal software entitlement, then you can configure the managed host to send data to Red Hat Insights with one command on that system.

```
[root@host ~]# insights-client --register
```

The client periodically updates the metadata that is provided to Insights for Ansible Automation Platform. However, you can manually run the `insights-client` command on a registered managed host to immediately refresh the client's metadata.

```
[root@host ~]# insights-client
Starting to collect Insights data for host.example.com
Uploading Insights data.
Successfully uploaded report from host.example.com to account 6076664.
View details about this system on console.redhat.com:
https://console.redhat.com/insights/inventory/dc480efd-4782-417e-a496-cb33e23642f0
```

Accessing Red Hat Hybrid Cloud Console

To access Red Hat Hybrid Cloud Console, log in to <https://console.redhat.com> with your Red Hat Customer Portal account and password.

Navigate to **Manage Infrastructure > Ansible Automation Platform** to open the Overview dashboard.

This dashboard provides basic information about your automation infrastructure as well as an updated navigation bar so that you can access other features of Insights for Ansible Automation Platform and automation analytics.

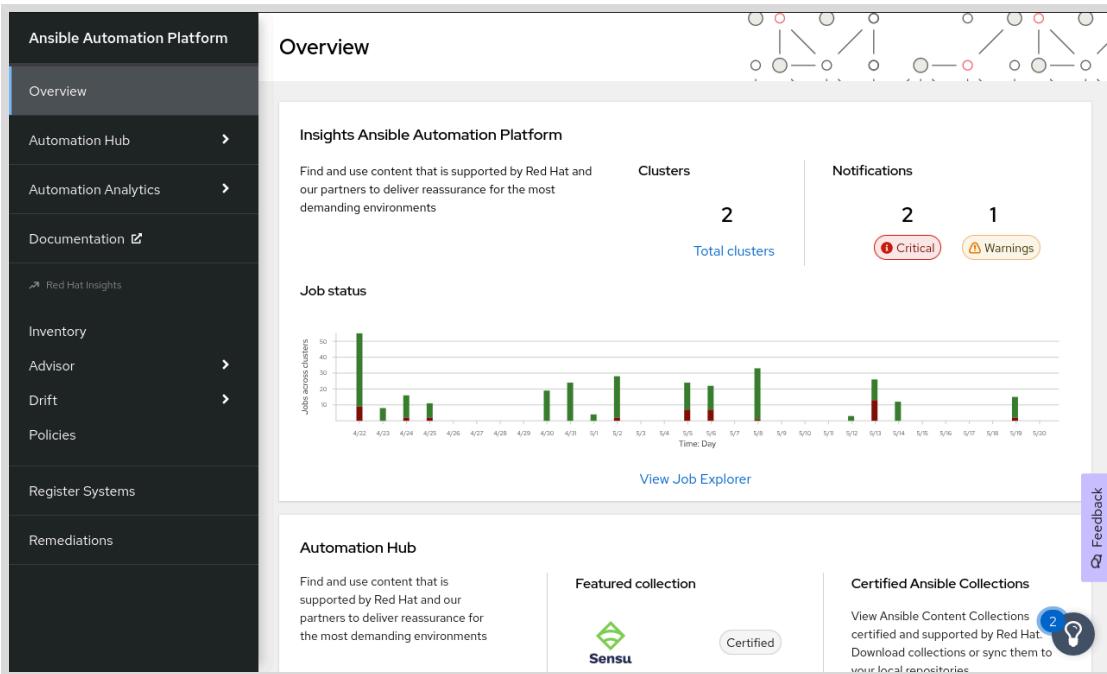


Figure 10.2: Overview dashboard for Insights for Ansible Automation Platform



References

Plan and measure your automation ROI

<https://www.youtube.com/watch?v=Xe8fBnJhAKI>

Automation Analytics Security and Data Handling

<https://access.redhat.com/articles/4501671>

Red Hat Insights for Ansible Automation Platform security FAQ

<https://www.ansible.com/products/insights-for-ansible/faq>

How to Activate Automation Analytics and Red Hat Insights

<https://www.ansible.com/blog/activate-insights-for-ansible-automation-platform>

Ansible Documentation: Automation Analytics

https://docs.ansible.com/automation-controller/latest/html/administration/usability_data_collection.html#automation-analytics

► Quiz

Gathering Data for Cloud-based Analysis

Choose the correct answers to the following questions:

- ▶ 1. **Which three options are required to get automation analytics and Insights for Ansible Automation Platform working? (Choose three.)**
 - a. A private automation hub cluster.
 - b. A Red Hat Hybrid Cloud Console account.
 - c. A Red Hat Ansible Automation Platform subscription.
 - d. An Ansible Galaxy account.
 - e. Automation controllers that can reach <https://console.redhat.com/>.
- ▶ 2. **Which statement is true about how data gathering for cloud-based services is configured in automation controller?**
 - a. You can only configure data gathering in the `inventory` file used by `./setup.sh` when you install automation controllers.
 - b. Anyone who can log in to your automation controller can enable data gathering at any time.
 - c. You cannot control how often automation controller sends data to the Red Hat Hybrid Cloud Console services.
 - d. The `admin` user can enable data gathering immediately after installing the automation controller, or through the web UI at any later time.
- ▶ 3. **Which two options are cloud-based services included with a subscription to Red Hat Ansible Automation Platform? (Choose two.)**
 - a. Insights for Ansible Automation Platform
 - b. Ansible Explorer
 - c. Automation analytics
 - d. Red Hat Ansible Decision Planner

► Solution

Gathering Data for Cloud-based Analysis

Choose the correct answers to the following questions:

- ▶ 1. **Which three options are required to get automation analytics and Insights for Ansible Automation Platform working? (Choose three.)**
 - a. A private automation hub cluster.
 - b. A Red Hat Hybrid Cloud Console account.
 - c. A Red Hat Ansible Automation Platform subscription.
 - d. An Ansible Galaxy account.
 - e. Automation controllers that can reach <https://console.redhat.com/>.
- ▶ 2. **Which statement is true about how data gathering for cloud-based services is configured in automation controller?**
 - a. You can only configure data gathering in the inventory file used by `./setup.sh` when you install automation controllers.
 - b. Anyone who can log in to your automation controller can enable data gathering at any time.
 - c. You cannot control how often automation controller sends data to the Red Hat Hybrid Cloud Console services.
 - d. The `admin` user can enable data gathering immediately after installing the automation controller, or through the web UI at any later time.
- ▶ 3. **Which two options are cloud-based services included with a subscription to Red Hat Ansible Automation Platform? (Choose two.)**
 - a. Insights for Ansible Automation Platform
 - b. Ansible Explorer
 - c. Automation analytics
 - d. Red Hat Ansible Decision Planner

Getting Insights into Automation Performance

Objectives

- Review data in the Red Hat Insights for Red Hat Ansible Automation Platform cloud service in order to automate issue remediation, and to detect and alert you to configuration drift.

Insights for Ansible Automation Platform

Insights for Ansible Automation Platform can guide you through the process of resolving existing issues on your platform. It can also alert you to proactively eliminate problems before they occur. Insights for Ansible Automation Platform provides the following tools:

Advisor

Reports on issues with systems and produces remediation playbooks.

Drift

Reports on differences between systems.

Policies

Alerts you to specific changes to system facts.

Notifications

Provides information on automation infrastructure status.

Generating Remediation Playbooks with Advisor

Advisor reports on issues with your systems that Insights for Ansible Automation Platform has identified, and automatically produces Ansible Playbooks to remediate many of those issues. You can download these playbooks and run them with your automation controller or with ansible-navigator.

To use Advisor, from the Red Hat Hybrid Cloud Console, access the Overview dashboard, and navigate to **Advisor > Recommendations**.

Name	Modified	Category	Total risk	Risk of ...	Systems impacted	Remediation
Decreased security: TLS key permissions	3 months ago	Security	Important	High	8	Playbook
The postgresql database performance decreases when the tuned best practices are not applied	2 months ago	Performance	Moderate	Low	3	Playbook

Figure 10.3: Advisor recommendations page

Chapter 10 | Getting Insights into Automation Performance

The **Advisor recommendations** page displays a list of issues that apply to your systems. By default, the list is ordered based on the **Total risk** value for each issue. This risk is assigned one of four values (**Critical**, **Important**, **Moderate**, or **Low**) based on the probability that the issue has a negative impact on your infrastructure and the severity of that impact if it were to occur. You can sort the list by any column in the table.

The **Category** column organizes each issue by the type of impact it has (on availability, performance, security, or stability). The **Risk of change** column estimates the risk that applying this change might disrupt your systems (**High**, **Moderate**, **Low**, and **Very Low**). You can find more information on the issue and its impact in the detail page of an issue. This page usually includes a link to a Knowledgebase article with more information. To open the detail page of an issue, click the name of the issue.

You can download an executive report of all recommendations in PDF format by clicking the [Download executive report](#) link.

Automating Remediation of an Issue for Multiple Systems

In the **Remediation** column of the issue, an Ansible icon and the word **Playbook** is displayed if Advisor can generate an Ansible Playbook to remediate that issue.



Important

Advisor cannot generate remediation Ansible Playbooks for some issues. In those cases, it provides other information and manual instructions on how to remediate those issues.

To generate the required playbook for affected systems:

- Click the name of the issue to open its detail page.
- Select the checkbox next to each affected system that you want to remediate.

The screenshot shows a remediation task titled "The postgresql database performance decreases when the tuned best practices are not applied". The task was modified on 03 April 2022 and is categorized under "Performance". The total risk is "Moderate". A chart indicates "High likelihood" and "Medium impact". The risk of change is "Low", and a note states "System reboot is not required". Below this, the "Affected systems" section lists three selected hosts: "ansible-homelab.example.com", "aap-rhel84.example.com", and "aap.local.example.com", all of which are RHEL 9.0 and last seen 2 days ago.

Figure 10.4: Selecting affected systems to remediate

- Click **Remediate** and enter a name for the new playbook in the **Create new playbook** field, and then click **Next**.
- Review the selected systems and make any necessary changes, and then click **Next**.
- Review the information about the issues included in the remediation playbook. Click **Turn on autoreboot** if you want the remediated systems to reboot after the changes have been applied. The detail page in Advisor indicates if a reboot is required to complete the remediation of the issue.

The screenshot shows a user interface for creating an Ansible playbook. On the left, a vertical navigation bar lists three steps: 1. Select playbook, 2. Review systems, and 3. Remediation review (which is currently selected). The main area is titled 'Remediation review' and contains the following text:
Issues listed below will be added to the playbook **remediation**.
The playbook **remediation** **does not** auto reboot systems.

A red box highlights the 'Turn on autoreboot' button. Below it is a table with the following data:

Action	Resolution	Reboot required	Systems
The postgresql database performance decreases when the tuned best practices are not applied	Install required packages and set proper tuned profile	No	3

At the bottom are three buttons: 'Submit' (dark blue), 'Back' (light blue), and 'Cancel' (light blue).

Figure 10.5: Creating a remediating playbook

- Click **Submit** to create the remediation playbook.
- Navigate to **Remediations** and locate the playbook that you just created.
- Click the name of the created remediation playbook, and then click **Download Playbook** to download the playbook to your workstation.

Automating Remediation of Multiple Issues for One System

You can use the following procedure to generate a playbook to remediate some or all issues that apply to a particular system:

- Navigate to **Advisor > Systems**. Click the name of the system that you want to remediate to open its detail page.
- Select the checkbox next to each issue affecting the system that you want to remediate.

The screenshot shows the 'labutility.example.com' system details page in the Advisor interface. At the top, it displays the system's UUID (66e7ec8d-ecdd-4d3a-bbd5-0bd2739c85db) and last seen time (10 Jun 2022 11:39 UTC). A 'Actions' dropdown menu is visible. Below this, a summary section indicates '2 selected' issues and '2 Recommendations'. The main table lists two issues:

Description	Modified	Total risk	Remediation
Decreased security: TLS key permissions	3 years ago	Important	Playbook
The postgresql database performance decreases when the tuned best practices are not applied	2 months ago	Moderate	Playbook

Figure 10.6: Multiple issues to remediate in a single system

- Click **Remediate**, enter a name for the new playbook in the **Create new playbook** field, and then click **Next**.
- Review the selected issues and make any necessary changes, and then click **Next**.
- Review the information about the issues included in the remediation playbook. Click **Turn on autoreboot** if you want the remediated systems to reboot after the changes have been applied. The detail page in Advisor indicates if a reboot is required to complete the remediation of the issue.

The screenshot shows a web-based interface for creating a remediation playbook. At the top, a header bar reads "Remediate with Ansible" and has a close button "X". Below the header, a section titled "3 Remediation review" is shown. A sub-section titled "Remediation review" contains a note: "Issues listed below will be added to the playbook **remediation for labutility**. The playbook **remediation for labutility** **does not** auto reboot systems." Under this note, there is a link "Turn on autoreboot". Below these sections is a table listing two issues:

Action	Resolution	Reboot required	Systems
Decreased security: TLS key permissions	Fix bad permissions/ownership of TLS key files	No	1
The postgresql database performance decreases when the tuned best practices are not applied	Install required packages and set proper tuned profile	No	1

At the bottom of the interface are three buttons: "Submit" (dark blue), "Back" (light blue), and "Cancel" (light blue).

Figure 10.7: Creating a remediation playbook for multiple issues in a single system

- Click **Submit** to create the remediation playbook.
- Navigate to **Remediations** and locate the playbook that you just created.
- Click the name of the created remediation playbook, and then click **Download Playbook** to download the playbook to your workstation.

Comparing Systems with Drift

Drift is used to detect unexpected differences between the current configuration of a machine and the configuration that you expect. It reports on differences between two systems, between a system and a standard baseline configuration, or one system at two different points in time.

You can use Drift in the following scenarios:

- Find different versions of operating systems among servers that should have a uniform operating system.
- Determine whether or not two systems have the same version of a software package installed.
- Identify differences between a working and a nonworking system to help perform root-cause analysis of issues.
- Detect differences in hardware configuration that might be the cause of differences in performance.

Finding Differences Between Systems

The following procedure details how to use Drift to compare systems.

- From the Red Hat Hybrid Cloud Console, access the Overview dashboard, and then navigate to **Drift > Comparison**.
- Click **Add systems or baselines**.
- Select the checkbox next to each system that you want to compare, and then click **Submit**.

The screenshot shows a modal dialog titled "Add to comparison". At the top, there is a message: "Your systems are pre-filtered by the global context selector. Workloads: Ansible Automation Platform." Below this, there are two tabs: "Systems" (which is selected) and "Baselines". A search bar at the top right shows "Selected (2)". Under the search bar, there are filters: "Status Fresh" and "Stale". A "Reset filters" button is also present. The main area is a table with columns: Name, Tags, OS, Last seen, and Historical profil... (partially visible). There are four rows of data:

Name	Tags	OS	Last seen	Historical profil...
<input checked="" type="checkbox"/> aap0lab.example.com	7	RHEL 8.6	45 minutes ago	⌚
<input type="checkbox"/> aap2lab.example.com	7	RHEL 8.6	45 minutes ago	⌚
<input type="checkbox"/> aap6lab.example.com	7	RHEL 8.6	4 hours ago	⌚
<input checked="" type="checkbox"/> aap8lab.example.com	7	RHEL 9.0	11 hours ago	⌚

At the bottom of the table, there are "Submit" and "Cancel" buttons.

Figure 10.8: Selecting the systems to compare

- Review the **Comparison** page. The page shows a table where the left-most column is a list of facts, and the other columns represent the values on each compared system. By default, the columns are sorted by the state of the differences, starting with facts that are different between the compared systems, followed by the facts that are the same.

If a fact has subvalues (if its value is a dictionary, for example), you can click the arrow next to the fact to display the subvalues.

Comparing the State of One System at Different Times

You can use historical profiles to identify changes that a single system has undergone over time.

Use the following procedure to compare a system to one of its historical profiles:

- On the comparison page for the selected system, click the blue clock symbol under the name of your system.

Chapter 10 | Getting Insights into Automation Performance

- Select the checkbox next to the historical profile you want to compare.

The screenshot shows the 'Comparison' interface. At the top, there's a search bar with 'Fact name' and a 'Filter by fact' dropdown. Below it are buttons for 'State' (Same, Different, Incomplete data) and a 'Compare' button. On the right, a sidebar lists 'Historical profiles for this system' with checkboxes for various dates. One checkbox for '05 Jun 2022, 00:56 UTC' is checked and highlighted with a red box. Below the sidebar is a section for 'aapOlab.example.com' with a star icon and the date '10 Jun 2022, 05:19 UTC'.

Figure 10.9: Comparing historical profiles

- Click **Compare** to display the comparison page.

The screenshot shows the comparison results for 'controlab.example.com'. It lists facts with their states in two columns. A vertical red line separates the two columns. The first column has a blue header 'Fact ↑' and the second has a blue header 'State ↓'. The facts listed are: kernel_modules (warning), network_interfaces (warning), eth1.ipv4_addresses (warning), eth1.ipv6_addresses (warning), eth1.mac_address (warning), and eth0.ipv4_addresses (ok). The second column shows the corresponding values and states for each fact.

Fact ↑	State ↓	controlab.example.com ☆ 10 Jun 2022, 18:46 UTC ⓘ	controlab.example.com ☆ 07 Jun 2022, 23:34 UTC ⓘ
kernel_modules	!		
network_interfaces	!		
eth1.ipv4_addresses	!	10.30.0.147	10.30.0.247
eth1.ipv6_addresses	!	fe80::6ba0:4926:d11d:c86f	fe80::521e:8f37:901b:2c43
eth1.mac_address	!	fa:16:3e:ec:9d:2d	fa:16:3e:2a:c5:c4
eth0.ipv4_addresses	✓	172.25.25.17	172.25.25.17

Figure 10.10: Finding changes over time on the same system

Comparing Systems to a Standard Baseline

You can create one or more standard baseline configurations and then compare your systems against them.

- From the Red Hat Hybrid Cloud Console, access the Overview dashboard.
- Navigate to **Drift > Baselines**.
- Click **Create baseline**.
- In the window that opens, enter a name for the baseline.
 - You can create a baseline from scratch, in which case you must enter each fact by hand.
 - You can copy an existing baseline, if you have one.
 - Finally, if you have a system already configured to reflect the baseline, you can copy the facts from an existing system or historical profile.

Chapter 10 | Getting Insights into Automation Performance

If you create a baseline from an existing system or historical profile, use the following procedure:

- Select a system or historical profile from the list that is added to the **Create baseline** window.
- Click **Create baseline**.

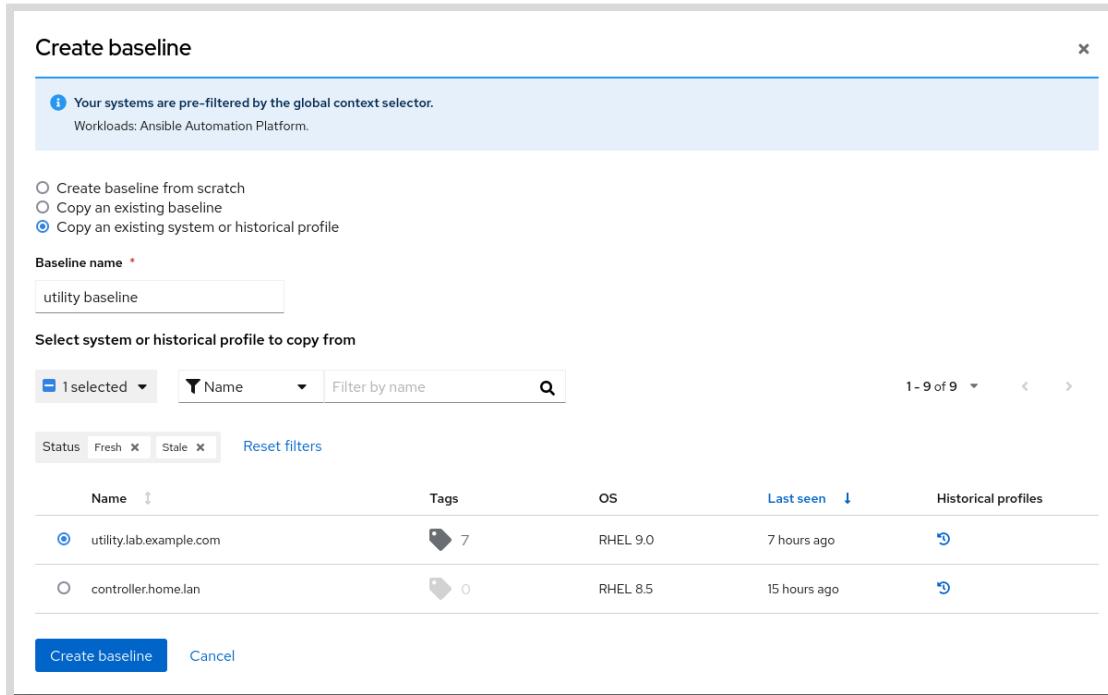


Figure 10.11: Creating a baseline from an existing system

To compare the baseline to a system, use the following procedure:

- Navigate to **Drift > Comparison**.
- When finding differences between systems, instead of comparing two systems, select the baseline and a system from the web UI.

Using Policies to Send Alerts Based on Ansible Facts

You can use the *Policies* component to create alerts to notify you when certain things change on your managed hosts. You can define the condition you want to monitor, based on facts collected by the platform, and trigger a notification when the condition is met. Typically, the notification is either sent by email or by triggering a webhook that you have already configured.

For example, your security team might need to know if a certain RPM package has been installed on one of your systems. You can set up a policy that monitors the `installed_packages` system fact, which lists the installed RPM packages on a system. If new facts are gathered for a managed host and that sensitive RPM package is present on that system, then the policy triggers the notification.



Note

See the References section for more information about the system facts and operators that you can use as conditions for these policies.

Use the following procedure to create a new policy:

Chapter 10 | Getting Insights into Automation Performance

- From the Red Hat Hybrid Cloud Console, access the Overview dashboard, and then navigate to **Policies** to display the **Policies** page.
- Click **Create policy**.
- Click **From scratch**, and then click **Next**.
- Enter a name for the policy in the **Name** field. Optionally, enter a description for the policy. Click **Next** when done.
- Define the conditions for your policy in the **Conditions text** field. Click **Validate condition** to validate that your condition is well structured, and then click **Next**.

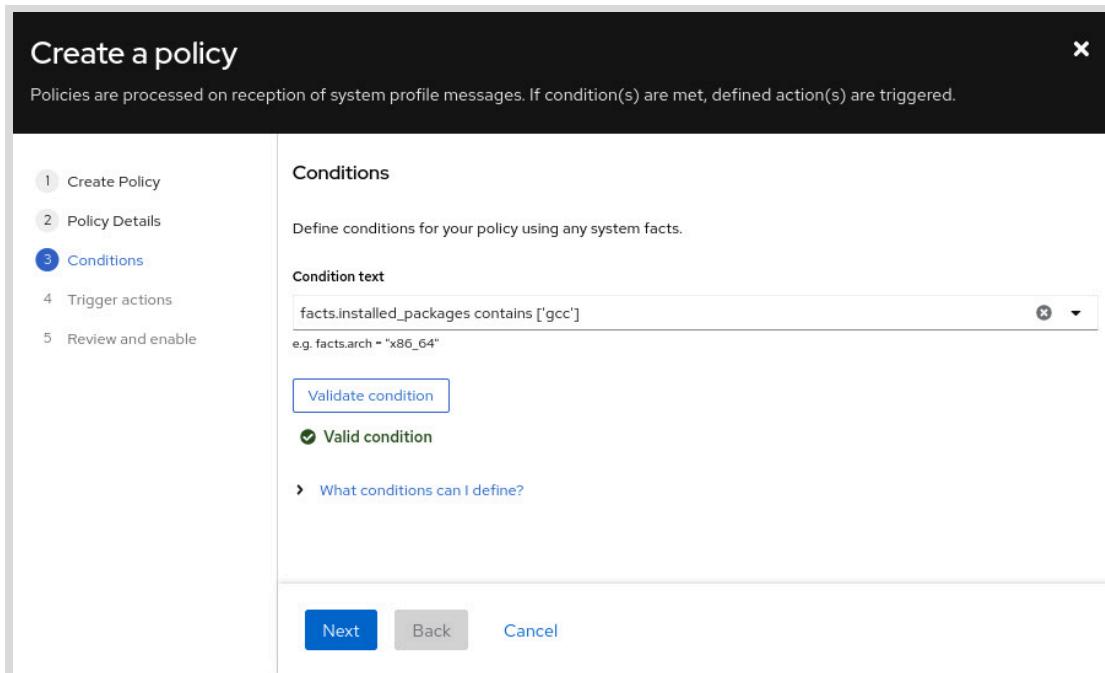


Figure 10.12: Validating the policy condition

- Click **Add trigger actions**.
- Click **Notifications** and then click **Next**.
- Click the **Enable this policy** switch to enable it.



References

Remediating Configuration Issues Using Advisor and Ansible Playbooks

https://access.redhat.com/documentation/en-us/red_hat_insights/2022/html/remediating_configuration_issues_using_advisor_and_ansible_playbooks/

Comparing System Configurations and Baselines in Red Hat Insights Inventory

https://access.redhat.com/documentation/en-us/red_hat_insights/2022/html/comparing_system_configurations_and_baselines_in_red_hat_insights_inventory/

Monitoring and Reacting to Configuration Changes Using Policies

https://access.redhat.com/documentation/en-us/red_hat_insights/2022/html-single/monitoring_and_reacting_to_configuration_changes_using_policies/

For more information about using Ansible facts in Policies, refer to the Appendix

from *Monitoring and Reacting to Configuration Changes Using Policies* at

https://access.redhat.com/documentation/en-us/red_hat_insights/2022/html-single/monitoring_and_reacting_to_configuration_changes_using_policies/index#assembly-policies-monitoring-appendix-ref-materials

► Quiz

Getting Insights into Automation Performance

Choose the correct answers to the following questions:

- ▶ 1. Which component of Insights for Ansible Automation Platform can you use to detect differences between two systems, between a system and a standard baseline configuration, or one system at two different points in time?
 - a. Advisor
 - b. Drift
 - c. Policies
 - d. Notifications

- ▶ 2. Which component of Insights for Ansible Automation Platform can you use to generate playbooks to remediate issues with your systems that have been detected by Red Hat Insights?
 - a. Advisor
 - b. Drift
 - c. Policies
 - d. Notifications

- ▶ 3. Which component of Insights for Ansible Automation Platform can you use to send alerts based on a change to some condition about your system that is reported by an Ansible fact?
 - a. Advisor
 - b. Drift
 - c. Policies
 - d. Notifications

► Solution

Getting Insights into Automation Performance

Choose the correct answers to the following questions:

- ▶ 1. Which component of Insights for Ansible Automation Platform can you use to detect differences between two systems, between a system and a standard baseline configuration, or one system at two different points in time?
 - a. Advisor
 - b. Drift
 - c. Policies
 - d. Notifications

- ▶ 2. Which component of Insights for Ansible Automation Platform can you use to generate playbooks to remediate issues with your systems that have been detected by Red Hat Insights?
 - a. Advisor
 - b. Drift
 - c. Policies
 - d. Notifications

- ▶ 3. Which component of Insights for Ansible Automation Platform can you use to send alerts based on a change to some condition about your system that is reported by an Ansible fact?
 - a. Advisor
 - b. Drift
 - c. Policies
 - d. Notifications

Evaluating Performance with Automation Analytics

Objectives

- Review data analysis provided by the automation analytics cloud service in order to identify issues with your automation infrastructure and automation jobs.

Automation Analytics

Automation analytics can help provide complete visibility of the performance for the automation infrastructure in your company, obtain the return on investment (ROI) of the automated projects, create savings plans, monitor them, measure automation costs, and so on. Using automation analytics, you can track medium and long-term strategic plans quickly and efficiently.



Note

Some of this information is also presented from the main controller dashboard.

The tools for automation analytics include:

Clusters

Information about success and failure of jobs in your clusters.

Job Explorer

Information about playbooks and workflows run by your clusters.

Reports

Produce reports on performance of your clusters and automation.

Automation Calculator

Report that estimates cost savings of your automation.

Savings Planner

Predicts cost and time savings of automation.



Note

Reports, Automation Calculator, and Savings Planner are covered in the next section.

Reporting Playbook Execution Status

The *Clusters* feature provides reports on the success and failure of playbook and workflow jobs.

From the Red Hat Hybrid Cloud Console, access the Overview dashboard, and then navigate to **Automation Analytics > Clusters**. Select a date range for the report. The default is the past 30 days. Automation analytics displays the job status for the jobs in your automation controller clusters during the specified date range.

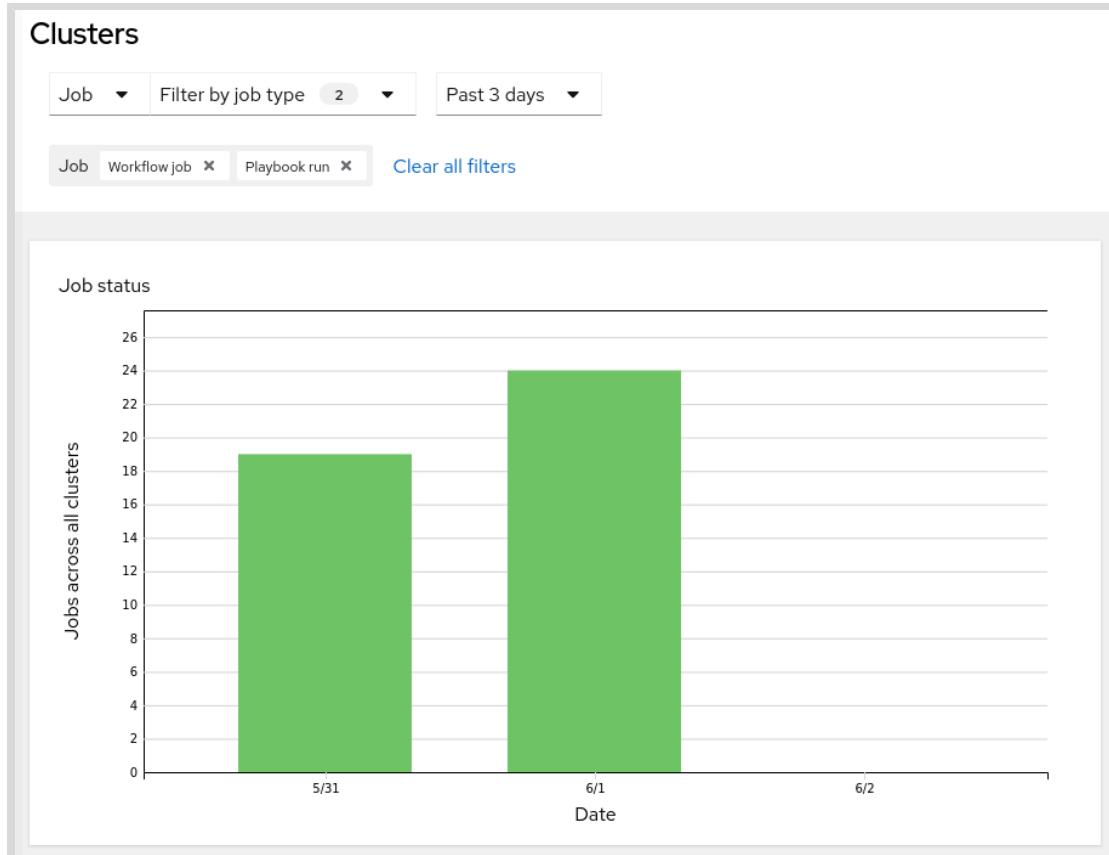


Figure 10.13: Reporting per automation controller

If you hover over the jobs graph, it displays summary statistics of the number of successful and failed jobs. You can get further detail by clicking on the summary statistics, or by directly navigating to the **Job Explorer** page.

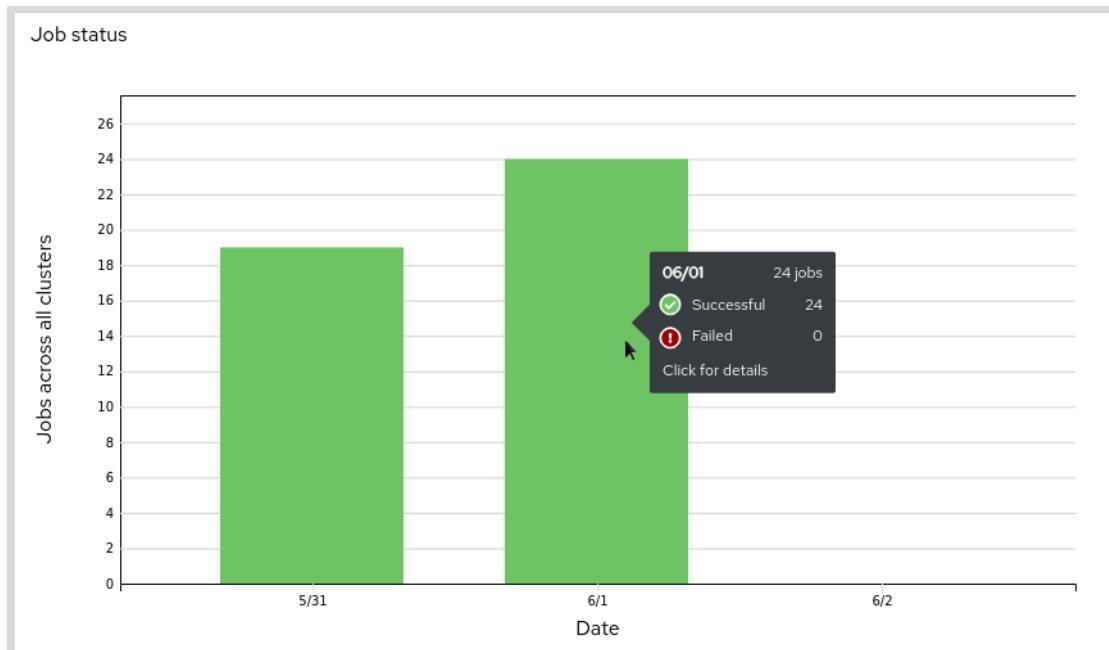


Figure 10.14: Successful and failed jobs on a certain date

Chapter 10 | Getting Insights into Automation Performance

You can apply filters to this data based on jobs, automation controller clusters, inventories, organizations, or templates. For example, you can limit the job status report to a single cluster.

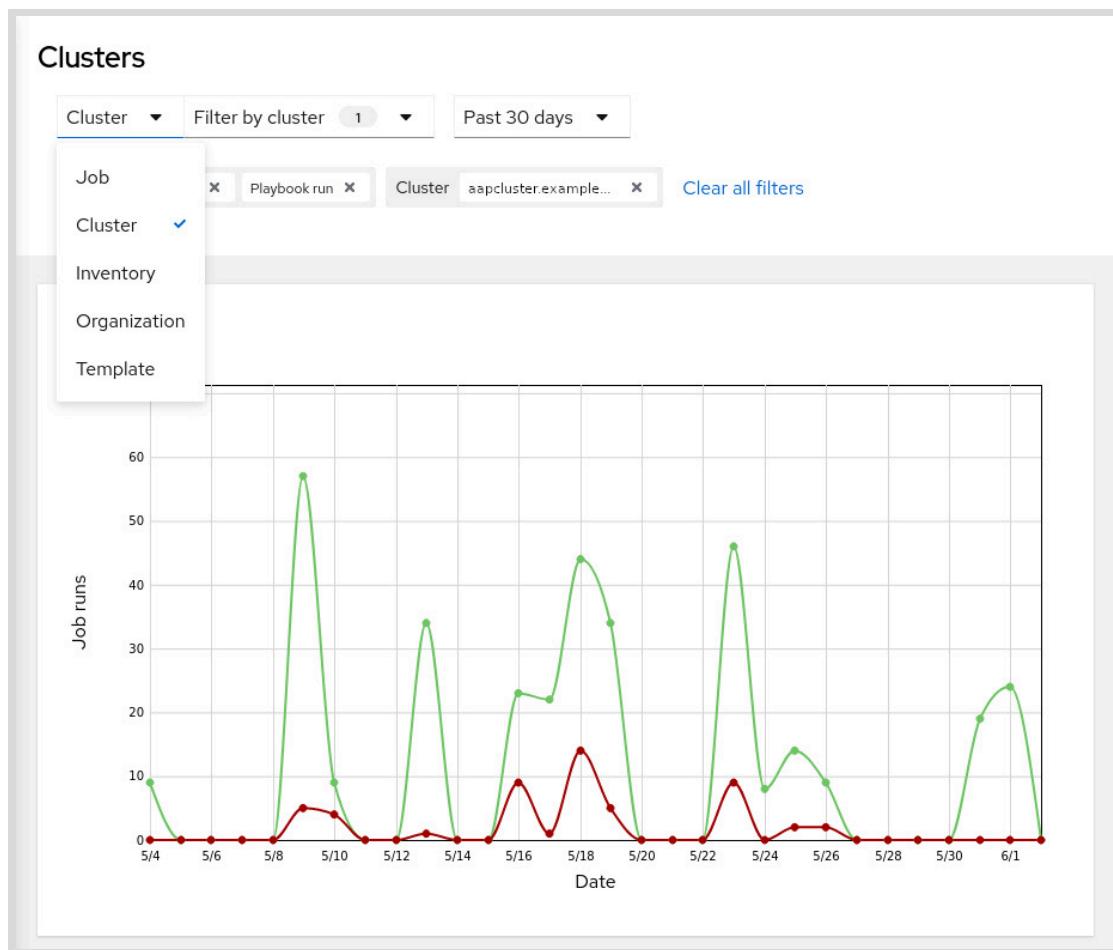


Figure 10.15: Reporting according to different types of filters

The interface also displays lists of the most frequently run workflows, templates, and modules.

Examining Job History

The **Job Explorer** feature lists all the playbooks and workflows that your automation controller clusters have run. From the Red Hat Hybrid Cloud Console, access the Overview dashboard, and then navigate to **Automation Analytics > Job Explorer**.

By default, the **Job Explorer** page displays both successful and failed jobs that were run over the past 30 days on all of your clusters and organizations. Click the arrow next to each job name to see when the job was created, when it was started, and when it finished. You can adjust filters to refine the list of results. For example, you might adjust the filters to only display successful jobs.

The screenshot shows the Job Explorer interface with the following details:

- Filtering:** The "Status" dropdown is set to "Successful". Other options include New, Pending, Waiting, Running, Failed, Error, and Canceled.
- Search:** A search bar is present at the top.
- Time Range:** The "Created" dropdown is set to "Past 30 days".
- Table Headers:** Cluster, Organization, Type.
- Data:** The table lists 5 jobs out of 205, all of which are successful. All jobs belong to the cluster "aapcluster.example.com", organization "Default", and type "Playbook run".

Figure 10.16: Filtering by successful jobs

To get more details on a job, click its name. Your browser opens the corresponding automation controller web UI, assuming that it is reachable by your web browser.

Monitoring Notifications

The **Notifications** feature provides information on the status of your automation infrastructure. It is organized under automation analytics in the Red Hat Hybrid Cloud Console web UI, but it is considered to be part of Red Hat Insights for Red Hat Ansible Automation Platform.

From the Red Hat Hybrid Cloud Console, access the Overview dashboard, and then navigate to **Automation Analytics > Notifications**.

By default, Notifications shows you all notifications for all your clusters. You can adjust the filters to select only the clusters you are interested in, or to select a notification severity from the following list:

- View Critical
- View Warning
- View Notice
- View All

By clicking the **Notice**, **Warning**, or **Critical** link, the platform redirects you to the corresponding automation controller web UI if your browser can reach it.

The screenshot shows the 'Notifications' page with the following details:

- Filter dropdowns: 'All Clusters' and 'View All'.
- Pagination: '1 - 4 of 4'.
- Notification 1: **Warning** (yellow icon). Cluster tower.lab.example.com license remaining host capacity is only 10 hosts. 1 day(s) ago.
- Notification 2: **Notice** (blue icon). Notifications last updated 2022-06-03T18:44:07+00:00.
- Notification 3: **Error** (red icon). Cluster Error: No data from tower.prod.com in 1 day. 1 day(s) ago.
- Notification 4: **Error** (red icon). Cluster Error: No data from tower.lab.example.com in 1 day. 1 day(s) ago.

Pagination at the bottom: '1 - 4 of 4' and '1 of 1'.

Figure 10.17: Reviewing notifications



References

Ansible Documentation: Automation Analytics

https://docs.ansible.com/automation-controller/latest/html/administration/usability_data_collection.html#automation-analytics

Evaluating your Automation controller job runs using the Job Explorer

https://access.redhat.com/documentation/en-us/red_hat_automation_platform/2.1/html/evaluating_your_automation_controller_job_runs_using_the_job_explorer/index

► Quiz

Evaluating Performance with Automation Analytics

Choose the correct answers to the following questions:

► 1. Which statement is NOT true about automation analytics?

- a. Automation analytics is a hosted cloud service at the Red Hat Hybrid Cloud Console website.
- b. Automation analytics provides consolidated feedback and reports on your registered automation controllers.
- c. Your secret credentials and the output of tasks are included in the data that the automation controller sends to automation analytics.
- d. Automation analytics can help you to calculate the return on investment (ROI) of your automation projects.

► 2. Which automation analytics feature can you use to list the names of all the playbooks and workflows in your automation controller?

- a. Organization Statistics
- b. Savings Planner
- c. Notifications
- d. Job Explorer

► 3. Which two items does the Clusters feature of automation analytics report on? (Choose two.)

- a. The number of jobs that were successful or that failed in your automation controller cluster on a specific date.
- b. The number of simultaneous users currently connected to your automation controllers.
- c. A list of the most frequently used workflows, templates, and modules.
- d. The uptime of all of your automation controllers.

► Solution

Evaluating Performance with Automation Analytics

Choose the correct answers to the following questions:

► 1. **Which statement is NOT true about automation analytics?**

- a. Automation analytics is a hosted cloud service at the Red Hat Hybrid Cloud Console website.
- b. Automation analytics provides consolidated feedback and reports on your registered automation controllers.
- c. Your secret credentials and the output of tasks are included in the data that the automation controller sends to automation analytics.
- d. Automation analytics can help you to calculate the return on investment (ROI) of your automation projects.

► 2. **Which automation analytics feature can you use to list the names of all the playbooks and workflows in your automation controller?**

- a. Organization Statistics
- b. Savings Planner
- c. Notifications
- d. Job Explorer

► 3. **Which two items does the Clusters feature of automation analytics report on? (Choose two.)**

- a. The number of jobs that were successful or that failed in your automation controller cluster on a specific date.
- b. The number of simultaneous users currently connected to your automation controllers.
- c. A list of the most frequently used workflows, templates, and modules.
- d. The uptime of all of your automation controllers.

Producing Reports from Automation Analytics

Objectives

- Generate reports based on data analysis provided by the automation analytics cloud service in order to improve your use of automation and to plan for future automation projects.

Producing Reports from Automation Analytics

Automation analytics provides a reporting function that you can use to evaluate your automation implementation. To work with these reports, from the Red Hat Hybrid Cloud Console, access the Ansible Automation Platform Overview dashboard, and then navigate to **Automation Analytics > Reports**.

Choosing an Appropriate Report

Automation analytics currently supports eleven different reports. The following table describes the reports produced by automation analytics.

List of Automation Analytics Reports

Report	Description	Use cases
Automation calculator	Estimate how much money you saved by automating tasks, compared to the cost of completing those tasks manually.	Identify which job templates contribute the most to your savings. Determine the return on investment (ROI) of your use of automation.
Changes made by job template	The total number of changes made by each job template during a certain period of time.	Evaluate whether the expected number of changes are being made by each job template. Identify the job templates that make the most changes.
Hosts by organization	The number of unique hosts, grouped by organizations, in automation controller.	Identify the organizations managing the most hosts.
Hosts changed by job template	The number of hosts changed by each job template during a certain period of time.	Identify if fewer or more hosts are changed than you expect at a particular time.

Report	Description	Use cases
Job template run rate	The number of times a job template has run in a specific time interval.	Determine which job templates run the most often.
Jobs/Tasks by organization	The number of jobs and tasks that were run at a particular time, grouped by organizations, in automation controller.	Identify which organizations are running the most Ansible jobs.
Module usage by job template	The number of jobs and tasks that were run using a specified set of Ansible modules, grouped by job template.	Determine which job templates use specific modules.
Module usage by organization	The number of jobs and tasks that were run using a specified set of Ansible modules, grouped by organization.	Identify which organizations are using particular modules.
Module usage by task	The number of jobs and tasks that were run using a specified set of Ansible modules, grouped by task.	Determine which tasks are using particular modules.
Most used modules	The number of jobs and tasks that were run using specific modules.	Identify the most commonly used modules in your automation code.
Templates explorer	An overview of the job templates that have been run by your Ansible Automation Platform cluster.	Review the status of a particular job template across its job runs to see how many times it fails or succeeds. Review the host and task status for the tasks that fail the most in order to identify potential problems.

When you navigate to the **Automation Analytics > Reports** page, the graph at the top of the page corresponds to the currently selected report at the bottom of the page. If you select a different report by clicking in its tile in the web UI, the graph changes to reflect that information.



Important

If you click inside the tile for the report, the graph at the top of the **Reports** page is updated to use that data, but if you click the *name* of the report in that tile, that is a link that takes you to a new page that consists of the full report.

Chapter 10 | Getting Insights into Automation Performance

For each report, you can filter the data by status, jobs, clusters, inventories, organizations, or templates. You can also specify the time period of the data you want to view. For example, you might select the **Changes made by job template** report and adjust its filters to show the subset of that data that was recorded over the last 6 months.

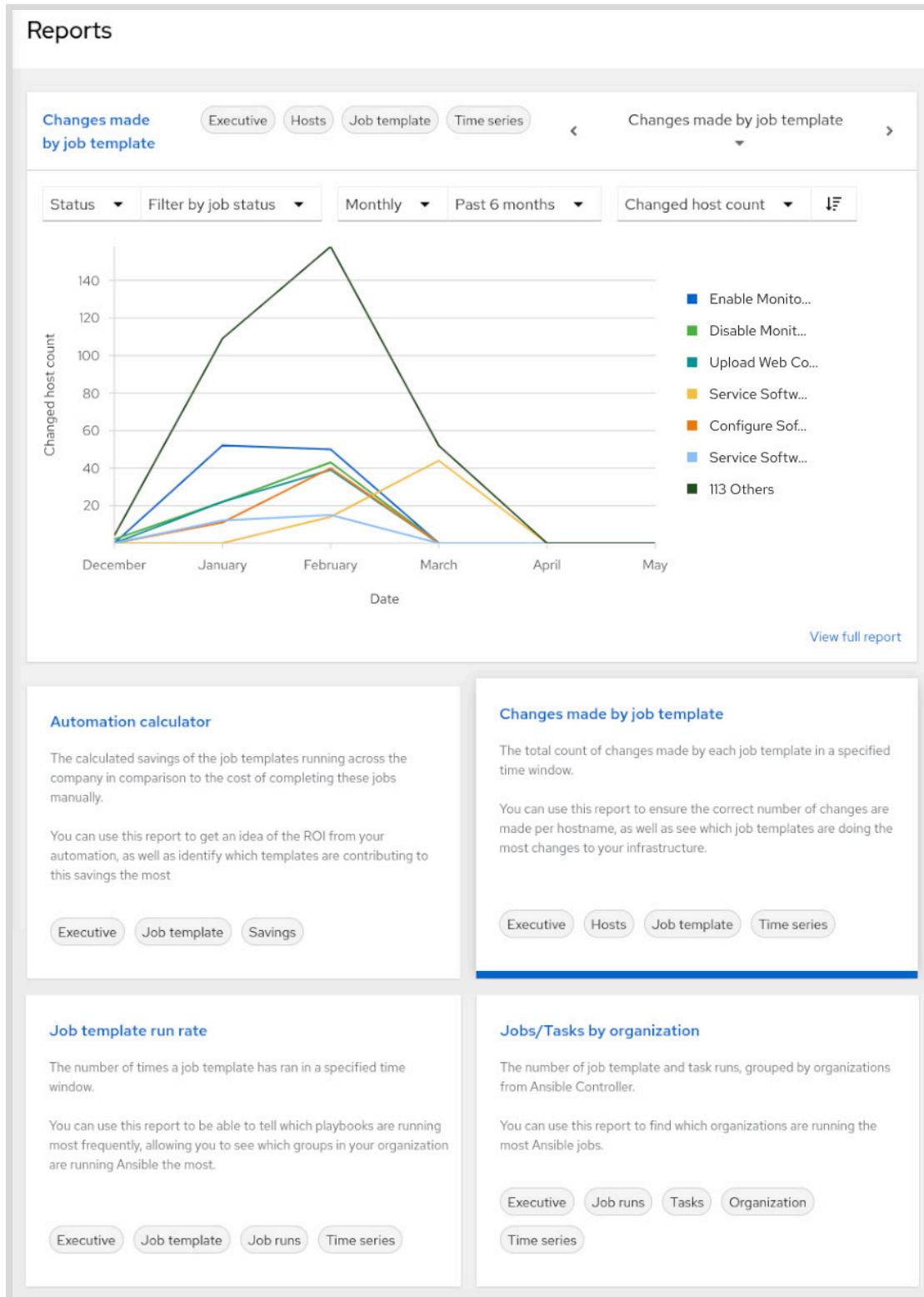


Figure 10.18: Selecting the "Changes made by job template" report tile

Chapter 10 | Getting Insights into Automation Performance

The **View full report** link in the graph area on the **Reports** page takes you to a new page for the full report. On that page you can make further adjustments to your filters, change between the **Line Chart** and **Bar Chart** graphing options, and download the report or send it by email.

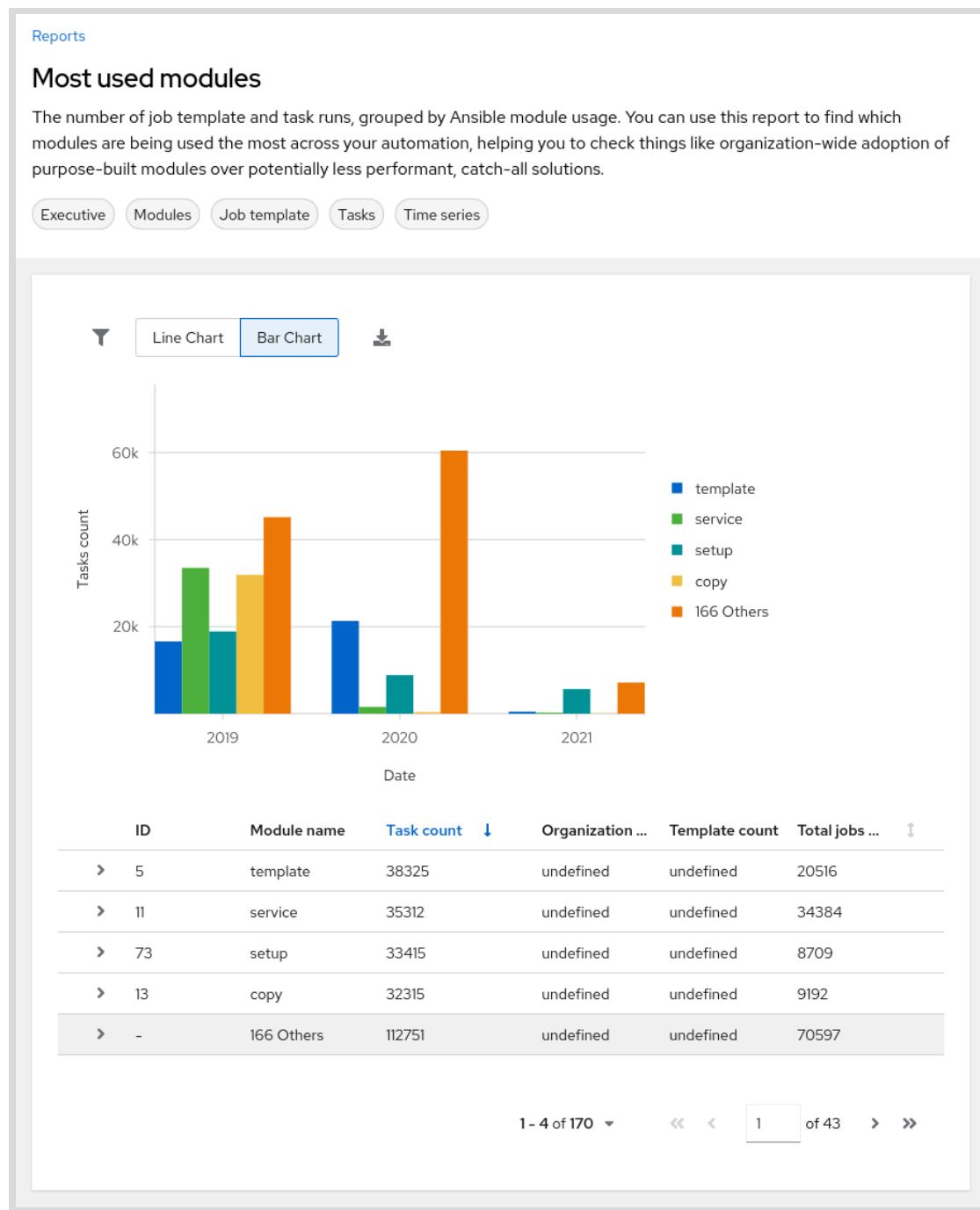


Figure 10.19: Viewing the full report for the "Most used modules" report

Using Automation Calculator to Compute Savings

The automation calculator report helps you identify how much money Ansible automation is potentially saving you, compared to the cost of performing those tasks manually. To access the automation calculator, navigate to **Automation Analytics > Automation Calculator**. You need to configure this report to realize its full potential.

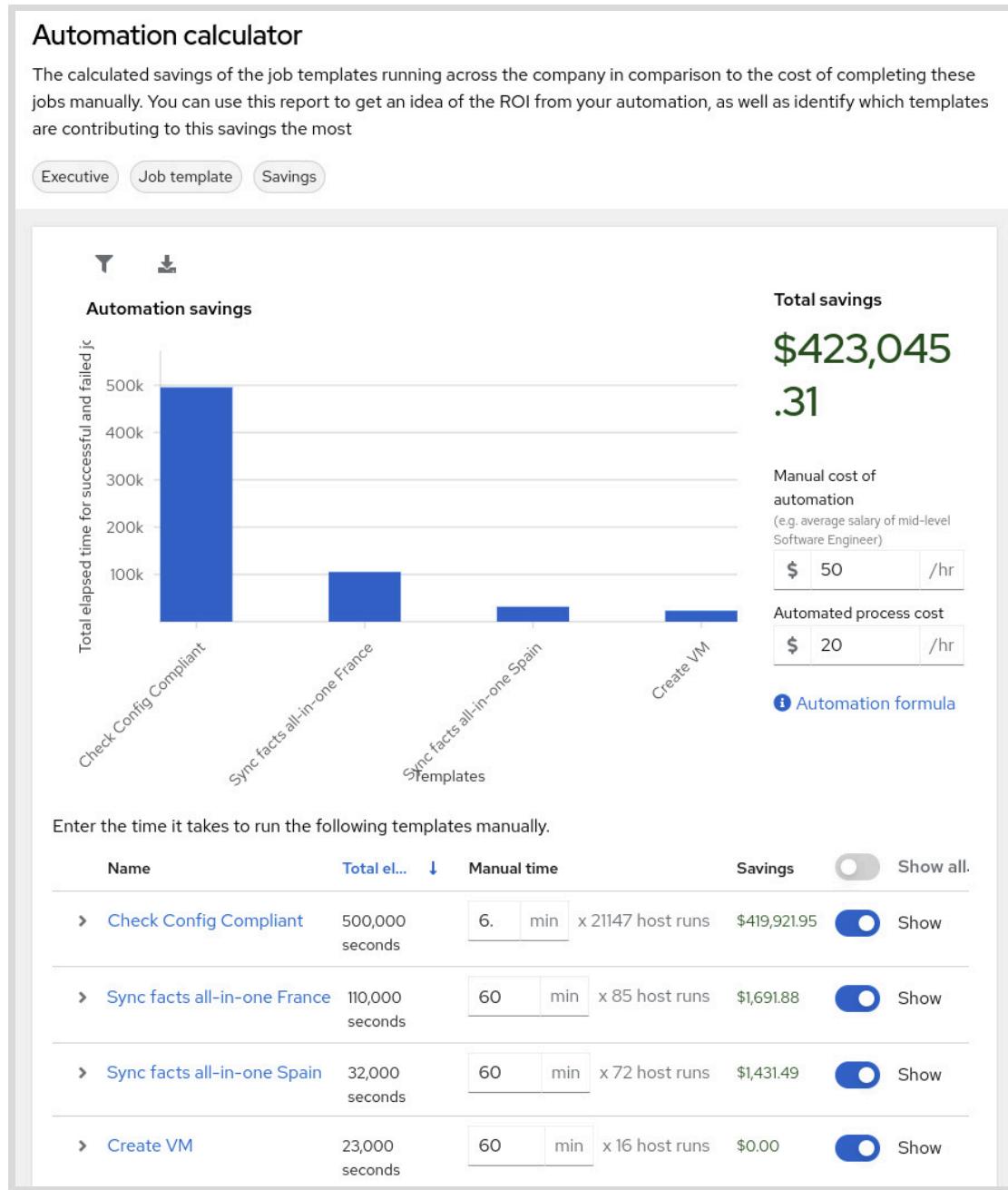


Figure 10.20: Estimating savings

The automation calculator report displays a list of your job templates, which you can filter.

For each job template, you enter the number of minutes it would take to perform its tasks manually in the **Manual time** field. The report multiplies the **Manual time** value by the number of times the associated job template runs to determine the total manual time.

To calculate the total monetary cost of manually performing those tasks, enter the hourly rate of the IT professional that would perform those tasks in the **Manual cost of automation** field. The report multiplies the manual cost of automation by the total manual time to determine the total manual cost for the tasks in that job template.

Chapter 10 | Getting Insights into Automation Performance

The *total automation time* for the tasks in that job template is calculated by adding up all the time used by successful runs of the job template. To calculate the total monetary cost of running these tasks automatically, enter the hourly rate of your automation processes in the **Automation process cost** field. The report multiplies the automation process cost by the total automation time to determine the total automation cost for that job template.

The report also calculates the cost of *failed* automation jobs. It is the amount of time used by failed jobs from that job template multiplied by the automation process cost.

- The cost savings in time for a job template is calculated by subtracting its total automation time from its total manual time.
- The monetary cost savings for a job template is the automation cost subtracted from the manual cost.
- The failed hosts cost per template is the amount of time spent on jobs that failed multiplied by the cost of the configured automated process.
- The actual monetary gain you have from a job template subtracts the failed host cost per template from your cost savings in money.

Exporting a Report

You can export any report as a PDF file or an email message so that you can share the report:

- Navigate to the report that you want to export. Adjust the filters on the report based on your needs.
- Click the **Export report** icon.
- Select the format for the report and then click **Next**.
- If you selected the **PDF** format, click **Export**. After a few seconds, your report is ready for downloading.

If you selected the **E-mail** format, configure the details as requested and then click **Send e-mail**. After a few seconds, automation analytics sends the email.

Predicting the Cost Savings of Automation

In addition to the automation calculator report, Red Hat Hybrid Cloud Console provides a reporting tool, the *automation savings planner*, that you can use to predict the time and money you might save by automating a specific set of tasks.

Creating a Savings Plan

To create a savings plan, you need to perform the following tasks:

1. Define the details of the savings plan.
2. Create a list of the tasks to be automated, in order.
3. Link a previously created job template to the tasks of the plan.

On Red Hat Hybrid Cloud Console, create a savings plan by navigating to the Overview dashboard and then navigating to **Automation Analytics > Savings Planner**. Click **Add plan** to start creating a new savings plan.

The automation savings planner opens the **Details** page, where you define the basic details of the plan.

The screenshot shows a 'Create new plan' form. At the top left is a 'Savings Planner' link. Below it, the title 'Create new plan' is displayed. A blue circular icon with the number '1' is followed by the word 'Details'. The form contains several input fields and dropdown menus:

- What do you want to automate? ***: A text input field containing 'Private cloud environment'.
- What type of task is it?**: A dropdown menu set to 'Development'.
- Enter a description of your automation plan**: A text input field containing 'Creation of a private cloud environment'.
- How long does it take to do this manually?**: A dropdown menu set to '8 hours (or more)'.
- How many hosts do you plan to run this on?**: An input field with a value of '8' and minus (-) and plus (+) buttons on either side.
- How often do you do this?**: A dropdown menu set to 'Monthly'.

At the bottom left are 'Next' and 'Cancel' buttons. On the right, there is a purple 'Feedback' button with a lightbulb icon.

Figure 10.21: Adding details to a new savings plan

Enter the following information:

What do you want to automate?

A name to identify the automation plan.

Enter a description of your automation plan

A description for the automation plan. (Optional.)

How many hosts do you plan to run this on?

You can increase the number of hosts by entering the number in the field, or by using the plus (+) or minus (-) symbols to increase or decrease the number of hosts.

What type of task is it?

Choose from the available list of options. The default value is **Operating System**.

How long does it take to do this manually?

Choose between **1 hour (or less)**, **2 hours**, **4 hours (default)**, **6 hours**, or **8 hours (or more)**. The default value is **4 hours (default)**.

How often do you do this?

Choose between **Daily**, **Weekly**, **Monthly**, or **Yearly**. The default value is **Weekly**.

Click **Next** to proceed to the **Tasks** page.

On the **Tasks** page, add a list of tasks that need to be automated.

In the **What tasks do you need to accomplish this plan?** field, enter a description for the new task, and then click the plus (+) symbol to add the task to the list. Add tasks one by one until you have entered them all.

The screenshot shows the 'Create new plan' interface in a web application. The title 'Create new plan' is at the top. On the left, a vertical navigation bar has three items: '1 Details' (selected), '2 Tasks' (current step), and '3 Link template'. The main area is titled 'What tasks do you need to accomplish this plan?'. A text input field says 'Enter a description of each task' with a '+' button to add more. Below is a table of tasks:

Task Description	Action
1. Request and approvals through Service Now	X
2. Vms creation	X
3. Load balancer setup	X
4. Connectivity settings	X
5. E-mail to communicate the environment creation	X

At the bottom are 'Next', 'Back', and 'Cancel' buttons, and a help icon.

Figure 10.22: Adding tasks to the new savings plan

You can remove tasks from the list by clicking X. You can drag and drop tasks to change their order.

When you have added all the tasks to the plan, click **Next** to proceed to the **Link template** page.

The screenshot shows the 'Create new plan' interface in a web application. The title 'Create new plan' is at the top. On the left, a vertical navigation bar has three items: '1 Details' (selected), '2 Tasks' (selected), and '3 Link template' (current step). The main area is titled 'Link a template to this plan:'. It shows a table with one row:

Status	Filter by job status	1-1 of 1
Inventory	Demo Inventory X	Clear all filters

Below the table is a search bar with 'Name ↑' and a radio button for 'Stop/Delete/Save all Instances'. At the bottom are 'Save', 'Back', and 'Cancel' buttons, and a help icon.

Figure 10.23: Adding a job template to the new savings plan

Chapter 10 | Getting Insights into Automation Performance

Select an existing job template from one of your automation controllers that is sending data to automation analytics, and link it to the savings plan. If you have many job templates, you can filter by status, job, automation controller cluster, inventory, organization, or job template.

Click **Save** to save the savings plan. The automation savings planner displays the **Details** tab of your completed savings plan.

The screenshot shows the 'Details' tab of a completed savings plan. At the top, there are three tabs: 'Back to Plans', 'Details' (which is selected and underlined), and 'Statistics'. Below the tabs, there is a table with the following data:

Name	Private cloud environment	Automation type	Development	Description	Creation of a private cloud environment
Manual time	8 hours (or more)	Run on hosts	8	Frequency	Monthly
Template	provision	Last job status	Successful	Last updated	6/6/2022 6:03:12pm

Below the table, there is a section titled 'Tasks' with a numbered list:

1. Request and approvals through Service Now
2. Vms creation
3. Load balancer setup
4. Connectivity settings
5. E-mail to communicate the environment creation

At the bottom of the screen, there are two buttons: 'Edit' and 'Delete'.

Figure 10.24: Completed savings plan

Reviewing the Cost Savings Calculations

When you have created at least one savings plan, you can calculate how much time and money your company might save by using it. Automation analytics shows you a three-year projection of costs and savings for your automation plan.

To review savings calculations:

- Navigate to **Automation Analytics > Savings Planner**.
- Click the name of the savings plan to review.
- Click the **Statistics** tab. By default it shows you the cost savings in dollars. Click between **Money** and **Time** to visualize either the monetary savings or the time savings of the plan.

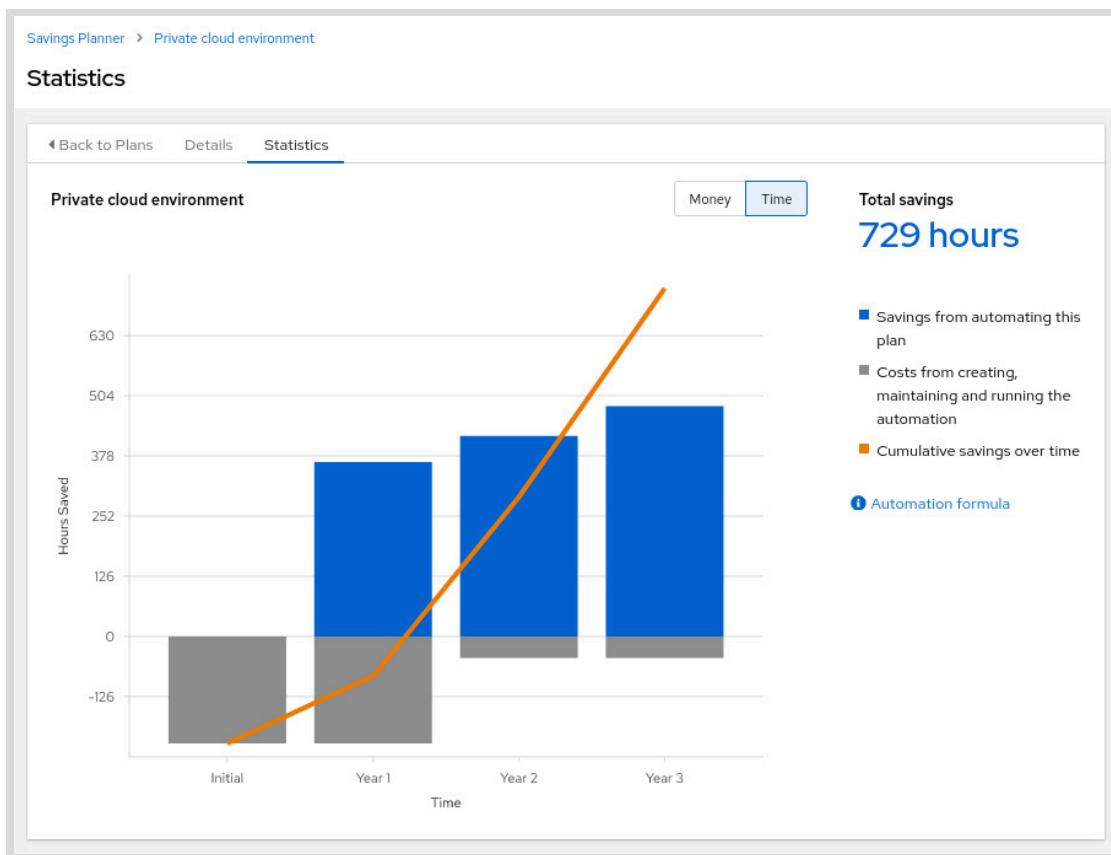


Figure 10.25: Time savings in a savings plan



References

Plan and measure your automation ROI

<https://www.youtube.com/watch?v=Xe8fBnJhAKI>

Planning your automation jobs using the automation savings planner

https://access.redhat.com/documentation/en-us/red_hat_automation_platform/2.1/html/planning_your_automation_jobs_using_the_automation_savings_planner/index

Using the Automation Calculator

https://access.redhat.com/documentation/en-us/red_hat_automation_platform/2.1/html/using_the_automation_calculator/index

► Quiz

Producing Reports from Automation Analytics

Choose the correct answers to the following questions:

► 1. **Which three statements are true about reports in automation analytics? (Choose three.)**

- a. You can use reports to estimate how much time and money your company can save with automation.
- b. The reports are only useful for detecting anomalies in Ansible automation performance.
- c. The reports can be exported in PDF format or can be sent by email.
- d. Each report can be customized with the type of graph to display, relevant date ranges, and different types of filters, such as jobs, clusters, inventories, organizations, templates, and so on.

► 2. **Which three pieces of information do you need to manually enter into automation calculator so that it can provide estimates of your current cost savings through Ansible automation? (Choose three.)**

- a. The number of minutes you would need to manually perform the tasks automated by each job template.
- b. The number of machines on which the automated tasks are performed.
- c. An estimate of the manual cost of automation.
- d. An estimate of the cost of providing your automation infrastructure and process support.

► Solution

Producing Reports from Automation Analytics

Choose the correct answers to the following questions:

- 1. **Which three statements are true about reports in automation analytics? (Choose three.)**
- a. You can use reports to estimate how much time and money your company can save with automation.
 - b. The reports are only useful for detecting anomalies in Ansible automation performance.
 - c. The reports can be exported in PDF format or can be sent by email.
 - d. Each report can be customized with the type of graph to display, relevant date ranges, and different types of filters, such as jobs, clusters, inventories, organizations, templates, and so on.
- 2. **Which three pieces of information do you need to manually enter into automation calculator so that it can provide estimates of your current cost savings through Ansible automation? (Choose three.)**
- a. The number of minutes you would need to manually perform the tasks automated by each job template.
 - b. The number of machines on which the automated tasks are performed.
 - c. An estimate of the manual cost of automation.
 - d. An estimate of the cost of providing your automation infrastructure and process support.

Summary

- Red Hat Insights for Red Hat Ansible Automation Platform is a cloud-based service that helps you identify, troubleshoot, and resolve operational, business, and security issues across your entire automation ecosystem.
- Advisor provides information about issues detected on your systems based on the data you provide and can generate Ansible Playbooks to remediate many issues automatically.
- Drift enables you to identify the differences between two systems, a system and a standard baseline, or one system at two different points in time.
- Policies can alert you to undesirable or unexpected changes to key facts about your systems.
- Automation analytics is a cloud-based service that provides a detailed view of automation activity across your organization so that you can observe and track how your organization is using automation.
- Automation Calculator and Savings Planner help estimate how much time and money your company can save when you use automation to replace manual, repetitive tasks.

Chapter 11

Building a Large-scale Red Hat Ansible Automation Platform Deployment

Goal

Use high availability techniques and automation mesh to scale up your Red Hat Ansible Automation Platform deployment.

Sections

- Designing a Clustered Ansible Automation Platform Implementation (and Quiz)
- Deploying Distributed Execution with Automation Mesh (and Guided Exercise)
- Managing Distributed Execution with Automation Mesh (and Guided Exercise) (and Quiz)

Designing a Clustered Ansible Automation Platform Implementation

Objectives

- Design a set of distributed Ansible Automation Platform servers to operate at a large scale and with improved reliability and redundancy.

Running Red Hat Ansible Automation Platform at Scale

The new architectural design and features in Red Hat Ansible Automation Platform 2 enables you to run Ansible automation on a large scale and with improved reliability, redundancy, and efficiency.

One of the key changes to Ansible Automation Platform is the new decoupled architecture, which splits the control plane from the execution plane, and the new *automation mesh*, which is used by nodes in the control plane to delegate automation jobs to nodes in the execution plane.

In earlier versions of Red Hat Ansible Automation Platform, Red Hat Ansible Tower (the predecessor of automation controller) ran in a hybrid mode, in which single servers provided both a web UI and API to *control* automation jobs, and the execution environment used to *run* automation jobs.

You can configure automation controller to separate these two modes. This separation means that you can have some nodes that only provide a web UI, an API, and control automation jobs, and many other nodes that can be positioned closer to your managed hosts, and which only execute automation jobs.

You can have multiple control nodes that schedule jobs cooperatively on execution nodes, spreading out the load across multiple systems. You can also add additional execution nodes to scale up your capacity to run jobs.

The control nodes communicate with the execution nodes by using automation mesh.



Note

Automation mesh and the new automation architecture replaces the less powerful isolated nodes feature used by Red Hat Ansible Automation Platform 1.

Automation Mesh

Automation mesh is an overlay network used to distribute work from machines running the automation controller web UI to dedicated, dispersed execution nodes that run jobs. Nodes in the automation mesh establish peer-to-peer connections with each other over the existing network.

Automation mesh provides the following benefits:

- Automation mesh offers a simple, flexible, and reliable way to independently scale up your control and execution capacity, expanding your automation close to the endpoints with minimal or no downtime.

- Automation mesh uses an end-to-end encrypted, authenticated network protocol over TCP. By separating automation mesh communication from SSH communication, you can more easily control automation mesh traffic through firewalls.
- Automation mesh is a multidirectional, multihopped overlay network that enables communication across constrained networks, to access endpoints not directly connected to automation controller. Traffic for a set of execution nodes that cannot be directly reached by the control node can be relayed through one or more hop nodes that can reach both the controller and the execution nodes.
- Automation mesh provides the ability to reconfigure how traffic is routed across the mesh if one or more nodes becomes unresponsive or unreachable. This helps make Ansible Automation Platform resilient to network disruptions and latency issues.
- Automation mesh enables operating a single distributed platform. This single distributed platform help reduce the overhead of managing multiple, isolated Ansible Automation Platform clusters.

Types of Nodes on Automation Mesh

Red Hat Ansible Automation Platform 2 separates the *control plane*, which starts automation jobs, from the *execution plane*, which runs automation jobs. In each category, it provides the following different types of nodes:

Control Plane

The control plane runs persistent automation controller services, such as the web UI, the task dispatcher, project updates, and management jobs. Control plane nodes can be either hybrid nodes or control nodes.

- **Hybrid nodes:** Hybrid nodes perform tasks for both the control plane and the execution plane. They control jobs and run jobs. This is the default node type for control plane nodes.
- **Control nodes:** Control plane nodes that use the `control` node type only perform control plane tasks and do not perform any execution plane tasks.

Execution Plane

The execution plane executes automation functions on behalf of the control plane.

- **Execution nodes:** Execution nodes run jobs by using container-based execution environments.
- **Hop nodes:** Hop nodes route traffic to other execution nodes. Hop nodes cannot execute automation functions.

The connections between nodes in automation mesh are expressed by peer relationships. If two nodes have a peer relationship, they can directly contact each other over automation mesh.

The default inventory file for the installation script peers all nodes in the control plane with all nodes in the execution plane. You can reconfigure these peering relationships to suit your network infrastructure and requirements.

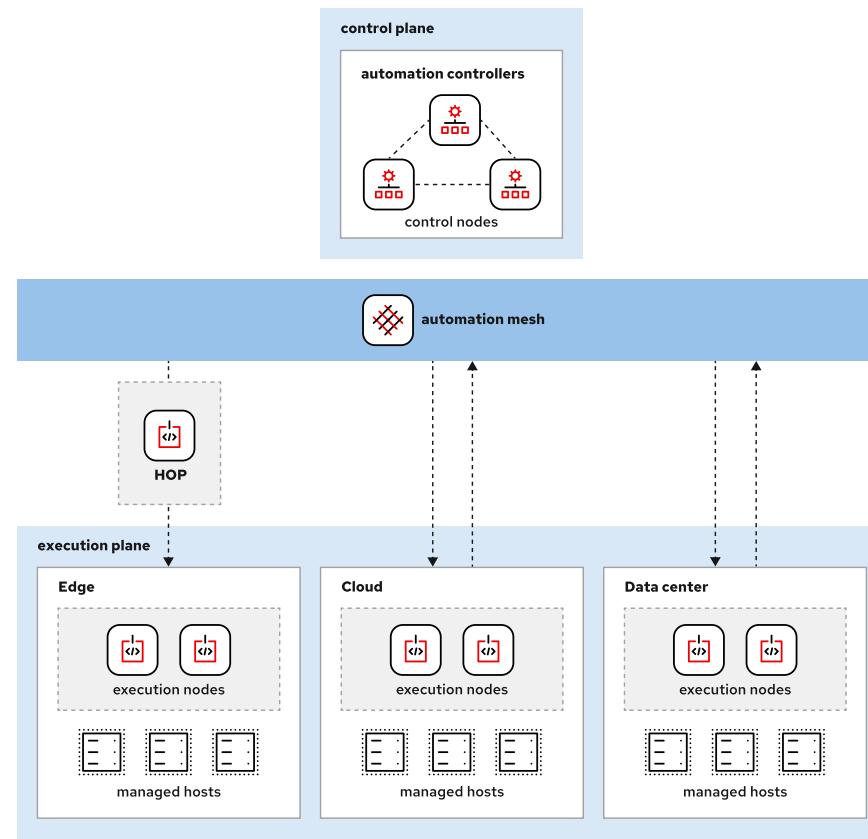


Figure 11.1: Execution and control plane

Using Instance Groups

An *instance group* is a group of execution or control nodes. You can use instance groups to run your automation jobs on specific sets of execution nodes.

You can configure a specific inventory of managed hosts to use a particular instance group of execution nodes to run automation jobs by default. You can also configure particular job templates to use a specific instance group by default. If you configure using a specific instance group, then you can improve performance by running jobs on execution nodes that are in the same data center as the managed hosts in the inventory.

You might also have managed hosts that can only be reached by specific execution nodes. By putting those execution nodes in an instance group, you can make sure that jobs for those hosts only run on nodes in that instance group.

Automation mesh creates the `default` instance group for all execution nodes and hybrid nodes, and the `controlplane` instance group for all control nodes and hybrid nodes. You cannot remove the `default` instance group.

You can create additional instance groups, either when you install Red Hat Ansible Automation Platform or through the web UI of automation controller. For example, you might create instance groups to group execution nodes that share a similar geographic location or that you want to serve a particular data center or cluster.

The instance group specified in the job template takes precedence over the one specified in the inventory. This in turn takes precedence over any default instance group specified for the organization that owns the project that was used for the job template.

The control node send jobs to the execution node in the instance group with the most available capacity. If multiple execution nodes have the same capacity, jobs are sent to the first node listed in the instance group. If no instance group is specified for a job, then the default instance group is used.

Planning Network Communication and Firewalls

Automation mesh uses its own network protocol for communications between the control nodes, hub nodes, and execution nodes. This is a TLS-encrypted protocol that connects to port 27199/TCP by default, although you can change this port and the TLS certificates used during installation. The receptor service listens on this port for automation mesh communications.

Execution nodes still communicate directly with managed hosts using the SSH protocol. You can use a hop node to relay communications from a control node to an execution node using port 27199/TCP. By using a hop node, you can block control nodes from directly accessing execution nodes using the SSH protocol.



Important

The initial installation of all nodes requires SSH access from the machine where you run `setup.sh`.

Requirements for Control Nodes and Hybrid Nodes

Allow network communication using the following network ports:

Network ports	Service	Purpose
27199/TCP	Receptor	Used for communication between the control plane and hop nodes.
80/TCP and 443/TCP	HTTP/HTTPS	Used for accessing the automation controller web UI and API.
22/TCP	SSH	Used for secure access and administration, including configuration performed by Ansible using the <code>setup.sh</code> script.

If you restrict outbound access, then you must allow outbound access to 80/TCP and 443/TCP in order to download automation execution environments from a container registry.

Requirements for Hop Nodes

Allow network communication using the following network ports:

Network ports	Service	Purpose
27199/TCP	Receptor	Used for communication between the control plane and hop nodes.

Network ports	Service	Purpose
22/TCP	SSH	Used for secure access and administration, including configuration performed by Ansible using the <code>setup.sh</code> script.

Requirements for Execution Nodes

Allow network communication using the following network ports:

Network ports	Service	Purpose
22/TCP	SSH	Used for secure access and administration, including configuration performed by Ansible using the <code>setup.sh</code> script.
27199/TCP	Receptor	Used for communication between control nodes, hop nodes and execution nodes.

If you restrict outbound access, then you must allow outbound access to 80/TCP and 443/TCP in order to download automation execution environments from a container registry.

Execution nodes that are in a restricted environment might block SSH access to all machines except for hop nodes.

Planning for Automation Mesh

Your use of automation mesh might evolve in stages.

- Initially, you might start with one hybrid node that acts as both a control node and an execution node.

If you plan to add more nodes through automation mesh in the future, set up the initial hybrid node with an external, shared database or database cluster. (Otherwise the initial automation controller would be a single point of failure and would have load for both automation controller operations and database operations.)

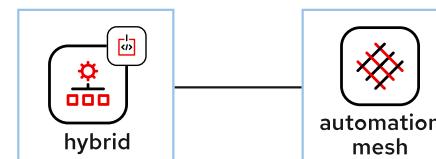


Figure 11.2: Single automation controller in hybrid mode

- You might add more hybrid nodes for resilience. The installer automatically configures peering among the control nodes.

You might want to add a load balancer in front of the hybrid nodes to distribute user requests among them, or you might allow users to access whichever control node is most convenient.

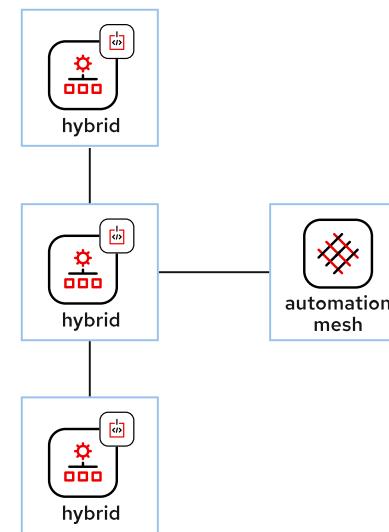


Figure 11.3: Control plane resilience with multiple hybrid mode automation controllers

3. You might segregate the control plane and the execution plane by adding an execution node and converting the hybrid nodes to control nodes that provide the control interface but do not run jobs.

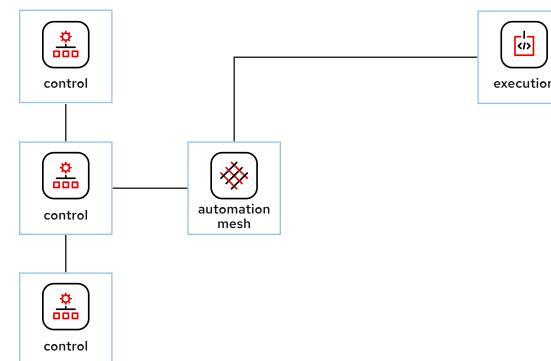


Figure 11.4: Separated control plane and execution plane

4. You might add more execution nodes for more execution capacity. Having more execution nodes also provides more resilience for the execution plane, in case an execution node fails.

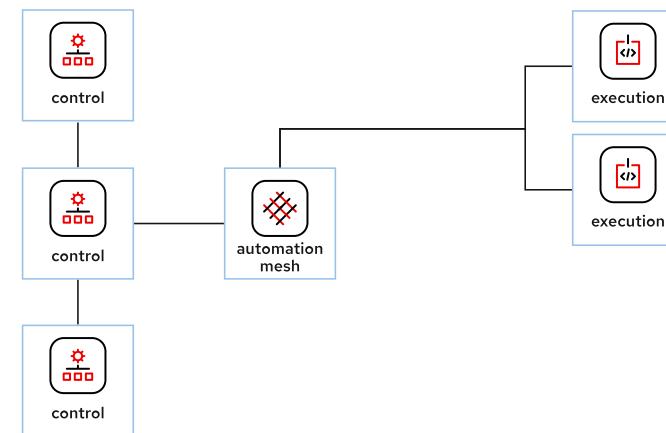


Figure 11.5: Execution plane resilience with multiple execution nodes

5. Your network configuration might use an internal firewall to restrict access to certain hosts. You might add a hop node that has controlled access to both networks to reach restricted execution nodes that can manage the restricted hosts.

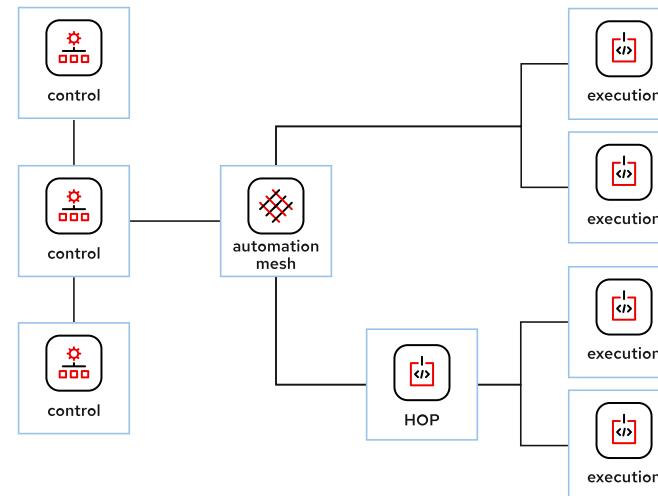


Figure 11.6: Added hop node to relay communications



Important

You might need to work with your networking and security teams to ensure that the hop node can access the restricted network.

6. You might add more hop nodes for resilience.

In this example, the two hop nodes normally communicate with specific execution nodes, but both hop nodes can communicate with each of the execution nodes. If one hop node fails or loses connectivity, then you can update automation mesh peering relationships to direct traffic to the affected execution nodes through the other hop node.

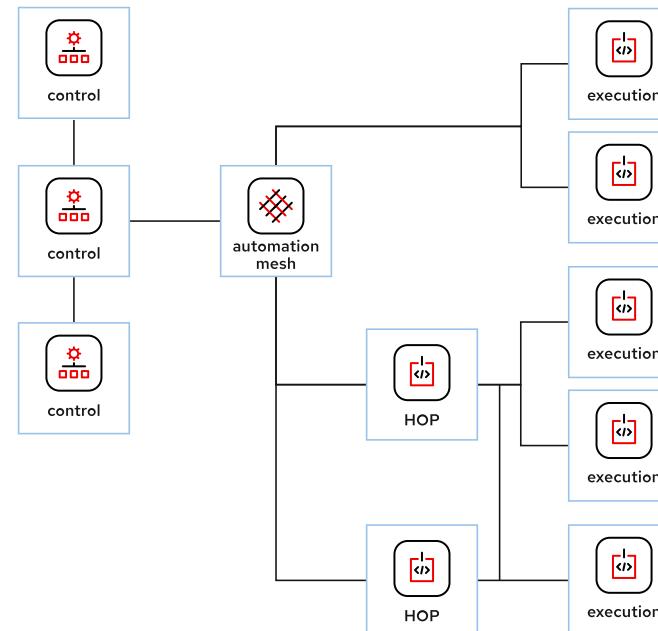


Figure 11.7: Multiple hop nodes that can access certain execution nodes

7. You might organize execution nodes into instance groups. You can configure your job templates and inventories to run jobs for particular managed hosts on execution nodes in particular instance groups that are close to (or have access to) those managed hosts.

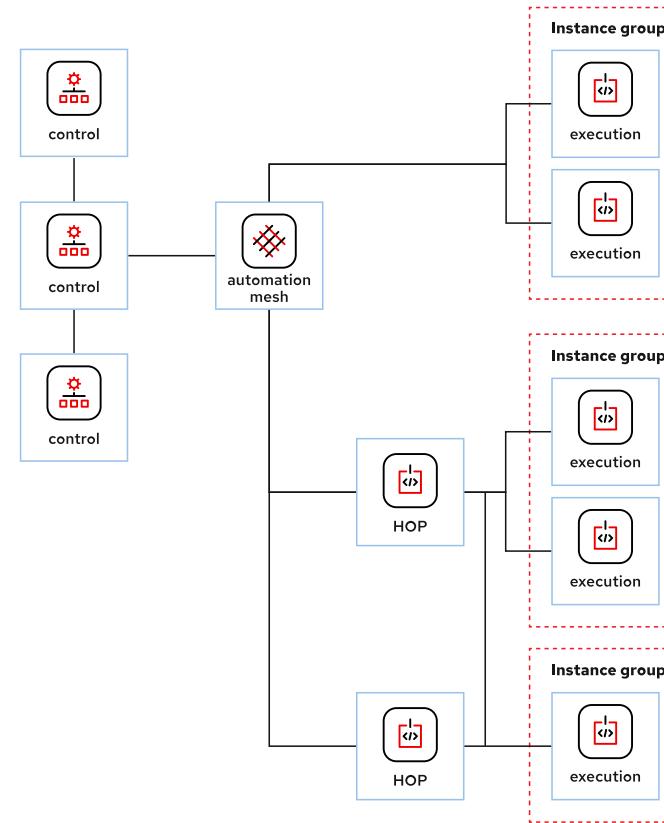


Figure 11.8: Using instance groups to localize job execution

Providing Resilient Services

In addition to having multiple control nodes, you should consider deploying multiple private automation hub servers. If your job templates routinely use Ansible Content Collections from your private automation hub, or if you store custom automation execution environment images in the container registry provided by private automation hub, then that service becomes a single point of failure. Having multiple private automation hub servers can help reduce this risk.

You can also use a load balancer for multiple control nodes or multiple private automation hub servers. Using a load balancer can avoid down time if one or more nodes or servers fails or becomes unreachable.

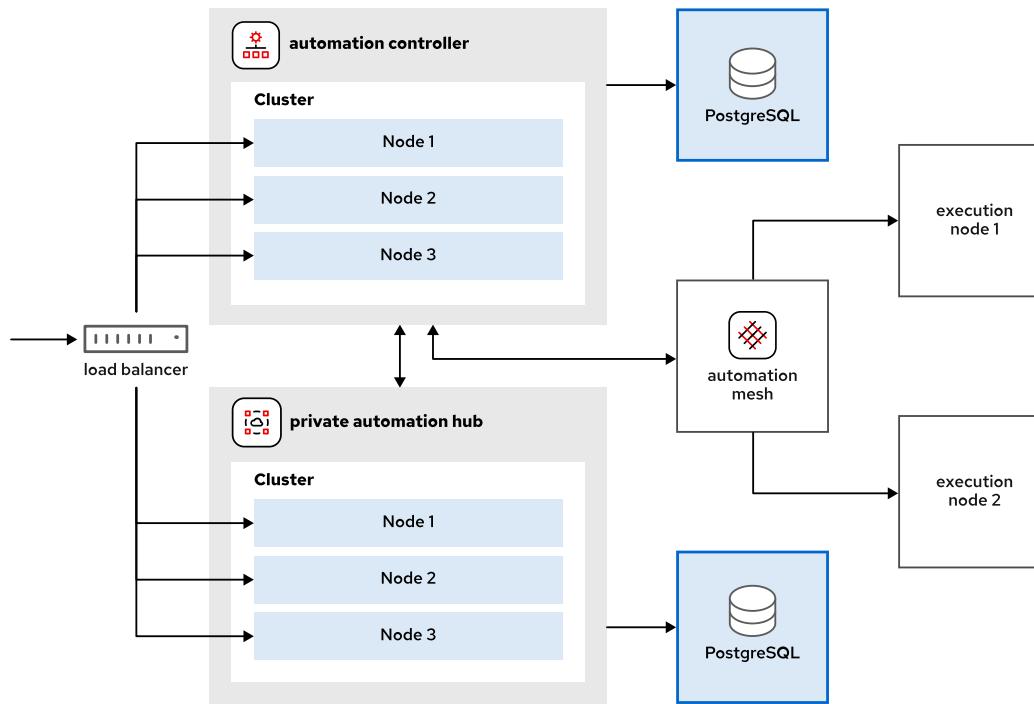


Figure 11.9: Reference architecture for large scale with redundancy



Note

Building more complex architectures that provide disaster recovery capabilities and use high availability database services are out of scope for this course.

If you are interested in these topics, then refer to the *Deploying Red Hat Ansible Automation Platform* reference architecture in the references at the end of this section.



References

Red Hat Ansible Automation Platform Automation Mesh Guide

https://access.redhat.com/documentation/en-us/red_hat_ansible_automation_platform/2.1/html-single/red_hat_ansible_automation_platform_automation_mesh_guide/index

Ansible Automation Platform 2 Release FAQ: Automation Mesh

<https://access.redhat.com/articles/6192881#automation-mesh>

For more information, refer to the *Deploying Red Hat Ansible Automation Platform* reference architecture at

https://access.redhat.com/documentation/en-us/reference_architectures/2021/html-single/deploying_ansible_automation_platform_2.1/index

► Quiz

Designing a Clustered Ansible Automation Platform Implementation

Choose the correct answers to the following questions:

► 1. **Which two of the following node types operate on the control plane? (Choose two.)**

- a. Control nodes
- b. Hop nodes
- c. Hybrid nodes
- d. Execution nodes

► 2. **What is the role of a hybrid node?**

- a. It only runs jobs by using container-based execution environments.
- b. It only routes traffic between the control plane and execution nodes.
- c. It performs tasks for both the control plane and the execution plane, controlling jobs and running jobs.
- d. It only performs control plane tasks and does not perform any execution plane tasks.

► 3. **Which three of the following choices are benefits that automation mesh introduces? (Choose three.)**

- a. Automation mesh transmits messages using the UDP protocol for lower overhead.
- b. Automation mesh can run on systems close to managed hosts, in order to reduce the impact of network latency.
- c. Automation mesh provides the ability to independently scale the control and execution planes.
- d. Automation mesh uses one-way communication from the controller using only SSH.
- e. Automation mesh uses an overlay network to route its traffic over a mesh that can be resilient to connectivity failures.

► 4. **What is an instance group?**

- a. Any group of automation controllers that are all on the same subnet.
- b. A complete Red Hat Ansible Automation Platform cluster of automation controllers, databases, execution nodes, and managed hosts.
- c. The set of managed hosts that you automate with Red Hat Ansible Automation Platform.
- d. A group of execution or control nodes on which you can run automation jobs, often used to ensure that specific jobs run on execution nodes close to the managed hosts on which they are automating tasks.

► Solution

Designing a Clustered Ansible Automation Platform Implementation

Choose the correct answers to the following questions:

► 1. Which two of the following node types operate on the control plane? (Choose two.)

- a. Control nodes
- b. Hop nodes
- c. Hybrid nodes
- d. Execution nodes

► 2. What is the role of a hybrid node?

- a. It only runs jobs by using container-based execution environments.
- b. It only routes traffic between the control plane and execution nodes.
- c. It performs tasks for both the control plane and the execution plane, controlling jobs and running jobs.
- d. It only performs control plane tasks and does not perform any execution plane tasks.

► 3. Which three of the following choices are benefits that automation mesh introduces? (Choose three.)

- a. Automation mesh transmits messages using the UDP protocol for lower overhead.
- b. Automation mesh can run on systems close to managed hosts, in order to reduce the impact of network latency.
- c. Automation mesh provides the ability to independently scale the control and execution planes.
- d. Automation mesh uses one-way communication from the controller using only SSH.
- e. Automation mesh uses an overlay network to route its traffic over a mesh that can be resilient to connectivity failures.

► 4. What is an instance group?

- a. Any group of automation controllers that are all on the same subnet.
- b. A complete Red Hat Ansible Automation Platform cluster of automation controllers, databases, execution nodes, and managed hosts.
- c. The set of managed hosts that you automate with Red Hat Ansible Automation Platform.
- d. A group of execution or control nodes on which you can run automation jobs, often used to ensure that specific jobs run on execution nodes close to the managed hosts on which they are automating tasks.

Deploying Distributed Execution with Automation Mesh

Objectives

- Distribute execution of Ansible Playbooks from automation controller control or hybrid nodes to remote execution nodes, communicating with them using automation mesh.

Configuring Automation Mesh

You can configure automation mesh to separate execution nodes from your control nodes, so that you can provide resilience and scalability to your automation, and so that you can move execution of your automation closer to your managed hosts.

To configure Red Hat Ansible Automation Platform nodes that are connected by automation mesh, edit the inventory file that is used by the installation script.

The following example deploys one control node (`controller.lab.example.com`) and one execution node (`exec1.lab.example.com`).

```
[automationcontroller] ①
controller.lab.example.com

[execution_nodes] ②
exec1.lab.example.com

[automationcontroller:vars] ③
peers=execution_nodes ④
node_type=control ⑤
```

- ① To add hybrid or control nodes, add an entry for each node in the `automationcontroller` group. By default, hosts in this group are hybrid nodes (`node_type=hybrid`).
- ② To add execution nodes or hop nodes, add an entry for each node in the `execution_nodes` group. By default, hosts in this group are execution nodes (`node_type=execution`).
- ③ Use group variables to set up definitions, such as `peers` and `node_type`, for all nodes in the group. You can also set the definitions individually on each host entry, but if you are repeating the configuration, then it is easier to set up a group and define variables.
- ④ The `peers=execution_nodes` directive specifies that all nodes in the `automationcontroller` group have a direct peering relationship in automation mesh with all nodes in `execution_nodes` (only `exec1.lab.example.com` in this example).
- ⑤ The `node_type=control` directive specifies that all nodes in the `automationcontroller` group be control nodes instead of hybrid nodes. You can override settings applied through group variables by setting variables for each individual host.

Creating Instance Groups

By default, automation mesh creates the `controlplane` instance group for nodes in the `[automationcontroller]` section of the inventory file, and the `default` instance group for execution nodes in the `[execution_nodes]` section of the inventory file.

You can configure the inventory to create additional instance groups. For example, you might want to create an instance group of execution nodes that are local to a particular data center. If you create instance groups based on data centers, then you can configure an inventory or a job template to run jobs for managed hosts that are in the same data center as the execution nodes.

The installer automatically creates instance groups for any group names in the installer's inventory with the `instance_group_` prefix. All of the hosts in the instance group must be execution nodes and cannot be hop nodes.

The following fragment of an inventory file creates two instance groups when you run `setup.sh`: `local`, which consists of the execution nodes `exec1` and `exec2`, and `remote`, which consists of the execution node `exec3`.

```
[instance_group_local]
exec1.lab.example.com
exec2.lab.example.com

[instance_group_remote]
exec3.lab.example.com
```

You can also manage instance groups and assign execution nodes to them in the automation controller web UI.

Adding Nodes to Automation Mesh

To add an additional node, add the new node to the inventory file and run the installation script again.

Removing Nodes from Automation Mesh

To remove a node, append `node_state=deprovision` to the node entry in the inventory file and run the installation script again.

```
[automationcontroller]
controller1.lab.example.com node_type=control
controller2.lab.example.com
controller3.lab.example.com node_state=deprovision

[execution_nodes]
exec1.lab.example.com
exec2.lab.example.com node_state=deprovision
```

**Important**

You cannot use the `node_state=deprovision` variable with the first entry in the `[automationcontroller]` section because we need at least one controller for an operational Ansible Automation Platform. The installer uses the first entry to launch the remaining installation and configuration. If you want to use the `node_state=deprovision` variable with the host listed in the first entry, then move that line to a different position in the `[automationcontroller]` section.

Removing Groups from Automation Mesh

You can also remove entire groups from your automation mesh. The installer removes all configuration files and logs attached to the nodes in the group.

```
[execution_nodes]
exec1.lab.example.com peers=exec2.lab.example.com
exec2.lab.example.com peers=exec3.lab.example.com
exec3.lab.example.com

[execution_nodes:vars]
node_state=deprovision
```

Visualizing Automation Mesh Topology

The Red Hat Ansible Automation Platform 2 installer provides a way to visualize the automation mesh topology defined in your installation inventory file. Using your inventory file, the installer can generate a text file that shows the relationships between the different nodes in your automation mesh. You can then use the `dot` command, provided by the `graphviz` package, to render the automation mesh topology text file as a graphic file.

**Note**

Red Hat Ansible Automation Platform 2.2 adds a topology viewer in the automation controller web UI under **Administration > Topology View**.

The following steps create and visualize the mesh topology:

1. Use an existing inventory file or create a new one. This example uses an inventory with two hybrid control nodes and one execution node.

```
[automationcontroller]
controller1.lab.example.com
control2.lab.example.com

[automationcontroller:vars]
peers=execution_nodes

[execution_nodes]
exec1.lab.example.com
```

2. Execute the script to generate the `mesh-topology.dot` file.

```
[user@demo aap-bundle]$ ./setup.sh --tag generate_dot_file
```



Note
You can run the `./setup.sh --help` command to display command usage, including how to pass Ansible options.

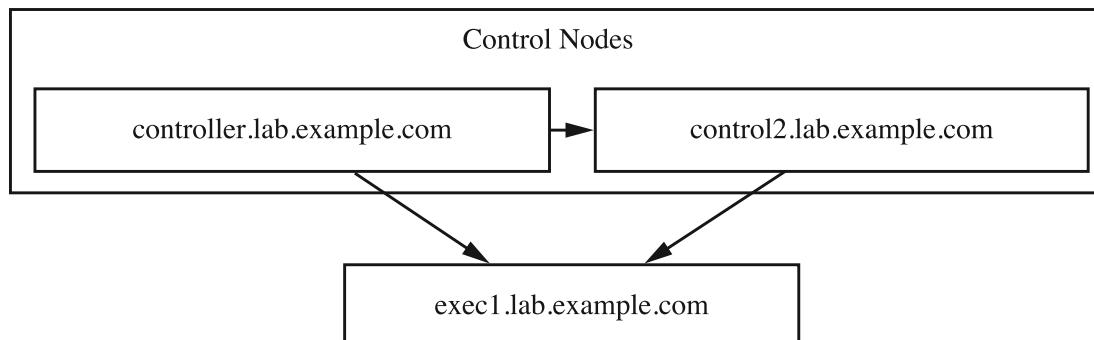
3. Make sure that the `graphviz` RPM package is installed.
4. Render the generated topology file, `mesh-topology.dot`, as a graphic.

```
[user@demo aap-bundle]$ dot -Tjpg mesh-topology.dot \
> -o graph-topology.jpg
...output omitted...
```



Note
You can render the file in other formats by using the `-T` option to specify the output format, such as GIF, PNG, or SVG (`-Tgif`, `-Tpng`, or `-Tsvg`).

5. Open the generated `graph-topology.jpg` file with a web browser or other image viewer.



Important

Even though the preceding diagram shows arrows for the connections between the various nodes on the automation mesh, these are peer connections and the arrow heads can be misleading.

Automation Mesh Design Patterns

Automation mesh is flexible and you can adapt it to meet your needs. The following two examples provide starting points. You might expand from these examples as you add locations in different regions, data centers, or behind firewalls. The key to setting up an effective automation mesh is to know the function of your nodes and which ones can directly peer with each other over your network.

Example 1: A Minimal Resilient Configuration

The minimal resilient automation mesh configuration provides resilience in the control and execution planes. The control nodes connect to each other and to each execution node. There is no single point of failure.

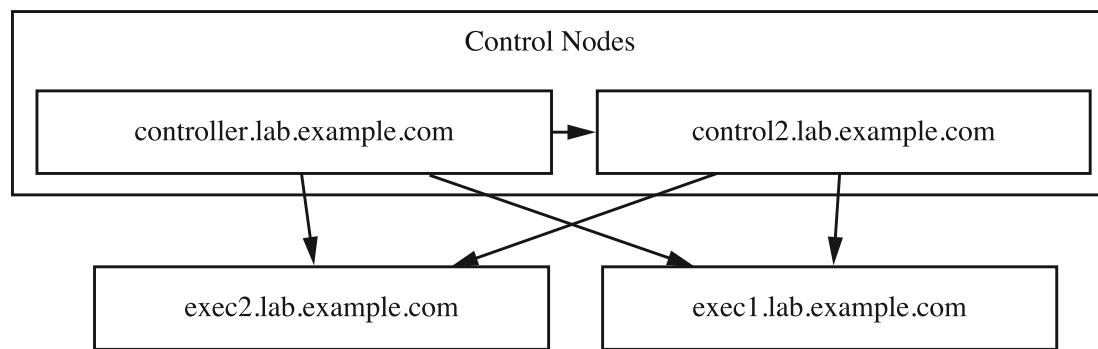


Figure 11.11: Minimal resilient configuration

The inventory file contains the following content:

```
[automationcontroller] ①
controller.lab.example.com
control2.lab.example.com

[automationcontroller:vars]
node_type=control ②
peers=execution_nodes ③

[execution_nodes] ④
exec1.lab.example.com
exec2.lab.example.com
```

- ① Two controllers manage the control plane.
- ② The controllers are `control` nodes instead of the default `hybrid` nodes.
- ③ The controllers peer with each node in the `execution_nodes` group.
- ④ Two execution nodes manage the execution plane.

Example 2: A Resilient Configuration with Local and Remote Instance Groups

This example modifies the minimal resilient configuration to add an additional execution node that is reached through a hop node. It also sets up two instance groups, `local` (consisting of `exec1` and `exec2`) and `remote` (consisting of `exec3`, which is behind the hop node). The instance groups are not shown on the following diagram.

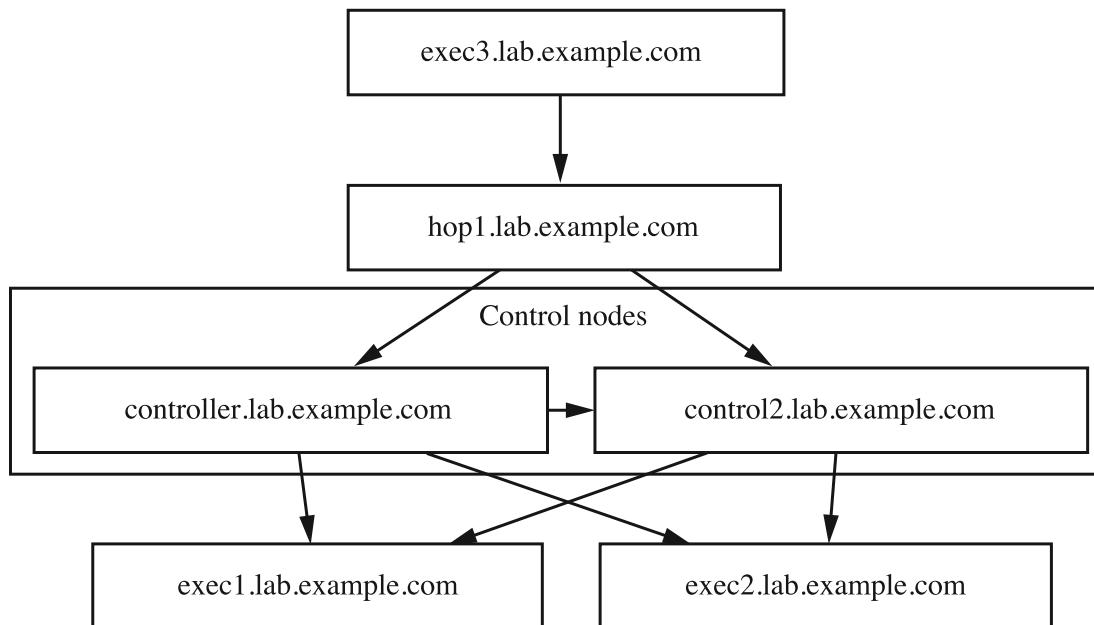


Figure 11.12: Local and remote execution

This excerpt is from the inventory file that sets up this configuration:

```
[automationcontroller]
controller.lab.example.com
control2.lab.example.com

[automationcontroller:vars] ①
node_type=control
peers=instance_group_local

[execution_nodes] ②
exec1.lab.example.com
exec2.lab.example.com
exec3.lab.example.com
hop1.lab.example.com

[instance_group_local] ③
exec1.lab.example.com
exec2.lab.example.com

[instance_group_remote] ④
exec3.lab.example.com

[instance_group_remote:vars]
peers=hop ⑤

[hop] ⑥
hop1.lab.example.com

[hop:vars] ⑦
node_type=hop
peers=automationcontroller
```

- ➊ All of the controllers are `control` nodes. Because at least one execution node is a hop node, the `automationcontrollers` group cannot peer with the entire `execution_nodes` group.
- ➋ List all execution nodes regardless of the node type. You could define the node type for each node, but it might be easier to create groups and group variables as shown in this example.
- ➌ The four hosts defined in the `execution_nodes` group, only `exec1.lab.example.com` and `exec2.lab.example.com` can establish peer connections with hosts in the `automationcontrollers` group. Because of the `instance_group_` prefix, the installation script creates this instance group resource.
- ➍ Execution nodes in the `instance_group_remote` group cannot directly connect to the control nodes.
- ➎ The execution nodes in the `instance_group_remote` group peer with the hosts in the hop group.
- ➏ Creating a separate group for hop nodes makes it easier to assign common variables and add or remove hop nodes from the group.
- ➐ All of the hosts in the hop group are hop nodes and they peer with all hosts in the `automationcontroller` group.

Validation Checks

Prior to installation, the installer script runs validation checks on your defined automation mesh configuration.

- A host cannot belong to both the `[automationcontroller]` and `[execution_nodes]` groups.
- A host cannot peer to a node that does not exist.
- A host cannot peer to itself.
- A host cannot have an inbound and an outbound connection to the same nodes.
- Execution nodes must have a path back to the control plane.



References

Red Hat Ansible Automation Platform Automation Mesh Guide

https://access.redhat.com/documentation/en-us/red_hat_ansible_automation_platform/2.1/html-single/red_hat_ansible_automation_platform_automation_mesh_guide/index

► Guided Exercise

Deploying Distributed Execution with Automation Mesh

Configure your automation controller as a control node that is connected using automation mesh to three execution nodes, one of which is behind a hop node.

Outcomes

- Configure an inventory file to support automation mesh.
- Install automation mesh.

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command downloads and extracts the Red Hat Ansible Automation Platform 2.2 bundled archive into the `/home/student/aap2.2-bundle` directory. It also downloads machine certificates and private keys into the `/home/student/certs` directory. Finally, it replaces the `inventory` file in the extracted bundle with the `inventory` file used in the chapter 1 installation guided exercise.

```
[student@workstation ~]$ lab start mesh-deploy
```

Instructions

- 1. Update the `/home/student/aap2.2-bundle/inventory` file to install a second automation controller, two execution nodes and a hop node connected directly to the automation controllers using automation mesh, and a third execution node that is connected to the hop node.
- 1.1. Change to the `/home/student/aap2.2-bundle` directory.

```
[student@workstation ~]$ cd ~/aap2.2-bundle/
```

- 1.2. Update the `inventory` file to add `control2.lab.example.com` to the `[automationcontroller]` section:

```
[automationcontroller]
controller.lab.example.com
control2.lab.example.com
```

- 1.3. Configure the `[automationcontroller:vars]` section to add the `node_type`, `web_server_ssl_cert`, and `web_server_ssl_key` variables. Remove the existing `peers=execution_nodes` line. The updated `inventory` file contains the following lines for the `[automationcontroller:vars]` section:

```
[automationcontroller:vars]
node_type=control
web_server_ssl_cert=/home/student/certs/{{ inventory_hostname }}.crt
web_server_ssl_key=/home/student/certs/{{ inventory_hostname }}.key
```

- 1.4. Because the [automationcontroller:vars] section now configures unique web server SSL certificates and keys for each automation controller host, comment out the existing web_server_ssl_cert and web_server_ssl_key variables in the [all:vars] section. The existing lines change to the following:

```
#web_server_ssl_cert=/home/student/certs/controller.lab.example.com.crt
#web_server_ssl_key=/home/student/certs/controller.lab.example.com.key
```

- 1.5. Update the inventory file to add hosts to the [execution_nodes] section.

- Add the exec1.lab.example.com and exec2.lab.example.com hosts and specify that they peer with the automationcontroller group.
- Add the exec3.lab.example.com host and specify that it peers with the hop1.lab.example.com host.
- Add the hop1.lab.example.com host and specify that it peers with the automationcontroller group and that it is a hop node.

The updated inventory file contains the following lines for the [execution_nodes] section:

```
[execution_nodes]
exec1.lab.example.com peers=automationcontroller
exec2.lab.example.com peers=automationcontroller
exec3.lab.example.com peers=hop1.lab.example.com
hop1.lab.example.com peers=automationcontroller node_type=hop
```

- 1.6. Use the diff command to compare your modified inventory file with the ~/mesh-deploy/inventory file. The diff command does not display any output if the files have the same content. The -B option ignores blank lines. Correct any mistakes before proceeding.

```
[student@workstation aap2.2-bundle]$ diff -B inventory ../mesh-deploy/inventory
```

► 2. Generate and view the automation mesh topology file.

- 2.1. Run the setup.sh script using the generate_dot_file tag.

```
[student@workstation aap2.2-bundle]$ ./setup.sh --tags generate_dot_file
...output omitted...
TASK [debug] ****
ok: [controller.lab.example.com] => {
    "msg": "Ansible Mesh topology graph created at 'mesh-topology.dot'. To render
    your dot graph, you could run: dot -Tjpg mesh-topology.dot -o graph-topology.jpg
    \n"
}
...output omitted...
```

2.2. Display the generated `mesh-topology.dot` topology file.

```
[student@workstation aap2.2-bundle]$ cat mesh-topology.dot
strict digraph "" {
    rankdir = TB
    node [shape=box];
    subgraph cluster_0 {
        graph [label="Control Nodes", type=solid];
        {
            rank = same;
            "controller.lab.example.com";
            "control2.lab.example.com";
            "controller.lab.example.com" -> "control2.lab.example.com";
        }
    }

    "exec1.lab.example.com";
    "exec2.lab.example.com";
    "exec3.lab.example.com";
    "hop1.lab.example.com";
    "exec1.lab.example.com" -> "control2.lab.example.com";
    "exec1.lab.example.com" -> "controller.lab.example.com";
    "exec2.lab.example.com" -> "control2.lab.example.com";
    "exec2.lab.example.com" -> "controller.lab.example.com";
    "exec3.lab.example.com" -> "hop1.lab.example.com";
    "hop1.lab.example.com" -> "control2.lab.example.com";
    "hop1.lab.example.com" -> "controller.lab.example.com";
}
```

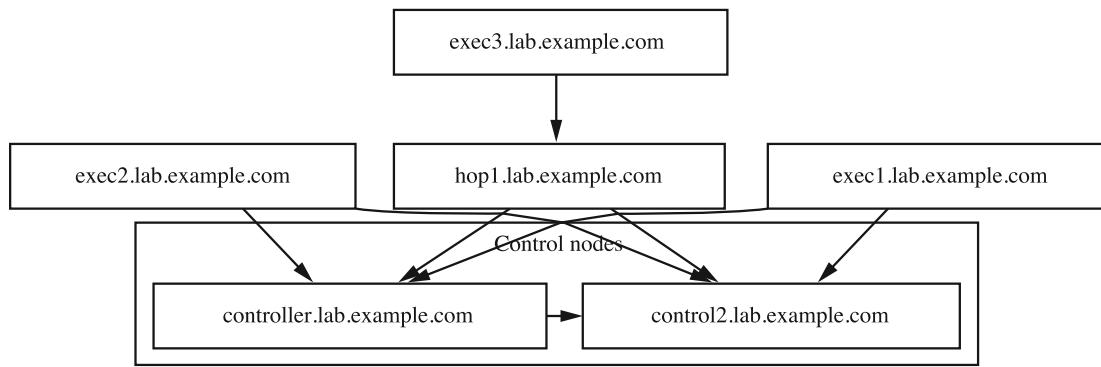
2.3. Install the `graphviz` package.

```
[student@workstation aap2.2-bundle]$ sudo dnf install graphviz
[sudo] password for student: student
...output omitted...
```

2.4. Render the generated topology file as a graphic.

```
[student@workstation aap2.2-bundle]$ dot -Tjpg mesh-topology.dot \
> -o graph-topology.jpg
...output omitted...
```

2.5. Open the generated graphic file, `graph-topology.jpg`, in a web browser.



► 3. Install automation mesh by applying the changes made to the `inventory` file.

3.1. Become the `root` user.

```
[student@workstation aap2.2-bundle]$ sudo -i  
[sudo] password for student: student  
[root@workstation ~]#
```

3.2. Change to the `/home/student/aap2.2-bundle` directory.

```
[root@workstation ~]# cd /home/student/aap2.2-bundle/
```

3.3. Run the `setup.sh` script with `-e ignore_preflight_errors=true` set to ignore the results of checks it makes before the installation starts. (The classroom systems have less RAM than is optimal for a production installation.) The installation takes approximately 15 minutes to complete.

```
[root@workstation aap2.2-bundle]# ./setup.sh -e ignore_preflight_errors=true  
...output omitted...  
PLAY RECAP *****  
control2.lab.example.com : ok=246 changed=129 ... failed=0 ... ignored=5  
controller.lab.example.com : ok=263 changed=54 ... failed=0 ... ignored=1  
db.lab.example.com : ok=75 changed=16 ... failed=0 ... ignored=1  
exec1.lab.example.com : ok=104 changed=51 ... failed=0 ... ignored=3  
exec2.lab.example.com : ok=104 changed=51 ... failed=0 ... ignored=3  
exec3.lab.example.com : ok=104 changed=51 ... failed=0 ... ignored=3  
hop1.lab.example.com : ok=83 changed=36 ... failed=0 ... ignored=2  
hub.lab.example.com : ok=195 changed=23 ... failed=0 ... ignored=1  
localhost : ok=3 changed=1 ... failed=0 ... ignored=0  
  
The setup process completed successfully.  
[warn] /var/log/tower does not exist. Setup log saved to setup.log.
```

3.4. After the installer finishes successfully, exit from the `root` session.

```
[root@workstation aap2.2-bundle]# exit
```

► 4. Access the `https://controller.lab.example.com` and `https://control2.lab.example.com` automation controllers.

Chapter 11 | Building a Large-scale Red Hat Ansible Automation Platform Deployment

- 4.1. Navigate to `https://controller.lab.example.com` and log in as the `admin` user with `redhat` as the password.
 - 4.2. In a separate browser window or tab, navigate to `https://control2.lab.example.com` and log in as the `admin` user with `redhat` as the password.
- 5. View the hosts in the `controlplane` and `default` instance groups.

- 5.1. Navigate to **Administration > Instance Groups**.

<input type="checkbox"/> Name	Type	Running Jobs	Total Jobs	Instances	Capacity	Actions
<input type="checkbox"/> controlplane	Instance group	0	2	2	Used capacity	
<input type="checkbox"/> default	Instance group	0	1	3	Used capacity	

- 5.2. Click the link for the `controlplane` instance group and then click the **Instances** tab. Both the `control2.lab.example.com` and the `controller.lab.example.com` hosts display the **Healthy** status. If you click the link for each hostname, then the **Instance details** page displays that each host is the `control` node type.
- 5.3. Navigate to **Administration > Instance Groups** and click the link for the `default` instance group.
- 5.4. Click the **Instances** tab. The `exec1.lab.example.com`, `exec2.lab.example.com`, and `exec3.lab.example.com` hosts display the **Healthy** status. If you click the link for each hostname, then the **Instance details** page displays that each host is the `execution` node type.

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish mesh-deploy
```

Managing Distributed Execution with Automation Mesh

Objectives

- Monitor execution of jobs on different execution nodes and maintain and adjust the automation mesh.

Managing Instance Groups in Automation Controller

When you install automation controller, it automatically creates the `controlplane` and `default` instance groups. The `controlplane` instance group is used by internal jobs that synchronize project contents and perform maintenance tasks. The `default` instance group runs user jobs that do not specify any other instance group.

You can create additional instance groups when you install automation controller by adding groups to the inventory for the `setup.sh` command with names that begin with `instance_group_`, followed by the name of the instance group you want to create.

You can also use the automation controller web UI to quickly create, modify, and delete instance groups without running the installation script.



Important

You cannot delete the `controlplane` or the `default` instance groups.

Creating Instance Groups

- Navigate to the web UI for one of the control nodes and log in as the `admin` user.
- Navigate to **Administration > Instance Groups** and then click **Add > Add instance group**.
- Enter a name for the instance group and then click **Save**.

Assigning Execution Nodes to an Instance Group

You can assign existing execution nodes to an instance group. If you created instance groups for geographic locations, then you might add execution nodes within those geographic locations to the appropriate instance groups. Using execution nodes that are in close geographic proximity to managed hosts reduces latency.

- Navigate to **Administration > Instance Groups** and then click the link for an instance group.
- Click the **Instances** tab and then click **Associate**.
- Select the nodes that you want to add to the instance group, and then click **Save**.

The following screen capture shows hosts that belong to the `Test Servers` instance group.

Name	Status	Running Jobs	Total Jobs	CPU	forks	RAM	Used Capacity	Actions
exec1.lab.example.com	Healthy	0	0	4	16	16	0%	Enabled
exec2.lab.example.com	Healthy	0	0	4	16	16	0%	Enabled

Running a Health Check on the Nodes

Automation controller monitors the health of instances. On the instance group's **Instances** tab in the web UI, hover over the health status for each instance to see the date and time stamp for the last health check. If desired, you can manually run a health check for one or more instances.

1. To view the date and time of the last health check, hover over the health status icon for an instance, expand the brief details for an instance, or click the link for the instance name to view the full instance details.

Name	Status	Node Type	Capacity Adjustment	Used Capacity
exec1.lab.example.com	Healthy	execution	16 CPU forks RAM 4 16 16	0% Used capacity 0%

Figure 11.16: Display last health check date and time

2. To manually run a health check, select one or more instances and then click **Health Check**.

Disassociating a Node from an Instance Group

Disassociating a node from an instance group removes the node from the instance group. You might do this so that you can associate the node with a different instance group or because you need to remove the node from your cluster.

1. Navigate to **Administration > Instance Groups** and then click the name of the desired instance group.
2. On the **Instances** tab, select the node that you want to remove from the instance group, and then click **Disassociate**. In the pop-up window, click **Disassociate** to confirm.

**Important**

Disassociating a node from an instance group does not remove the node from your cluster.

If you want to remove a node from your cluster, then you must add `node_state=deprovision` to the appropriate node or group in your installation script's inventory file and then run the installation script again.

Assigning Default Instance Groups to Inventories and Job Templates

You can explicitly configure inventories and job templates to use a particular instance group by default. If you do not specify an instance group for a job in either the job template or the inventory, then automation controller launches the job using the **default** instance group. When you install Ansible Automation Platform, the **default** instance group includes all hybrid and execution nodes (all nodes that can run jobs).

However, automation controller might select an execution node in the **default** instance group that is geographically distant on the network from the managed host, resulting in less than optimal performance.

Configuring an Inventory to Use Instance Groups

If you configure an inventory to select an instance group, then when a job template uses that inventory, automation controller assigns jobs to the execution nodes in that instance group. Configure an inventory to select an instance group using the following procedure:

1. Navigate to **Resources > Inventories** and then click the **Edit Inventory** icon for the inventory that should use an instance group.
2. Click the search icon for **Instance Groups**.
3. Select the desired instance group from the list, click **Select**, and then click **Save**.

Configuring a Job Template to Use Instance Groups

Like inventories, you can configure job templates to use instance groups. If an instance group is defined in both an inventory and in a job template that uses the inventory, then the instance group defined in the job template takes precedence.

1. Navigate to **Resources > Templates** and then click the **Edit Template** icon for the job template that you want to modify.
2. Click the search icon for **Instance Groups**.
3. Select the desired instance group from the list and click **Select**.
4. Click **Save**.

Running a Job Template with Instance Groups

A job always runs in an instance group. The job might use an instance group defined in the job template, in the inventory, or it might use the `default` instance group.

The **Details** tab for a job displays the name of the instance group and the name of the execution node used to run that job.

1. Navigate to **Resources > Templates** and then click the **Launch Template** icon for the desired job template.
2. After the job completes, click the **Details** tab.
3. Job details display the name of the instance group and the name of the execution node used by the job.

Testing the Resilience of Automation Mesh

You can manually test the resilience of the control and execution planes by making the nodes unavailable and then running jobs to verify that they still succeed.

Testing Control Plane Resilience

1. Navigate to **Resources > Projects** and click the **Sync Project** icon for the **Demo Project** resource.
2. Navigate to **Views > Jobs** and then click the link for the **Demo Project** job. The job has the **Source Control Update** type.
3. Click the **Details** tab and notice that the job used the **controlplane** instance group. Make note of the execution node used by the job.
4. Navigate to **Administration > Instances** and then disable the execution node identified in the previous step. Set **Enabled** to off. The status changes to **Disabled**.
5. Navigate to **Views > Jobs** and then click the **Relaunch Job** icon for the **Demo Project** job.
6. After the job completes, click the **Details** tab and notice that the job used a different execution node.
7. Navigate to **Administration > Instances** and then enable the previously disabled execution node. Set **Disabled** to off. The status changes to **Enabled**.

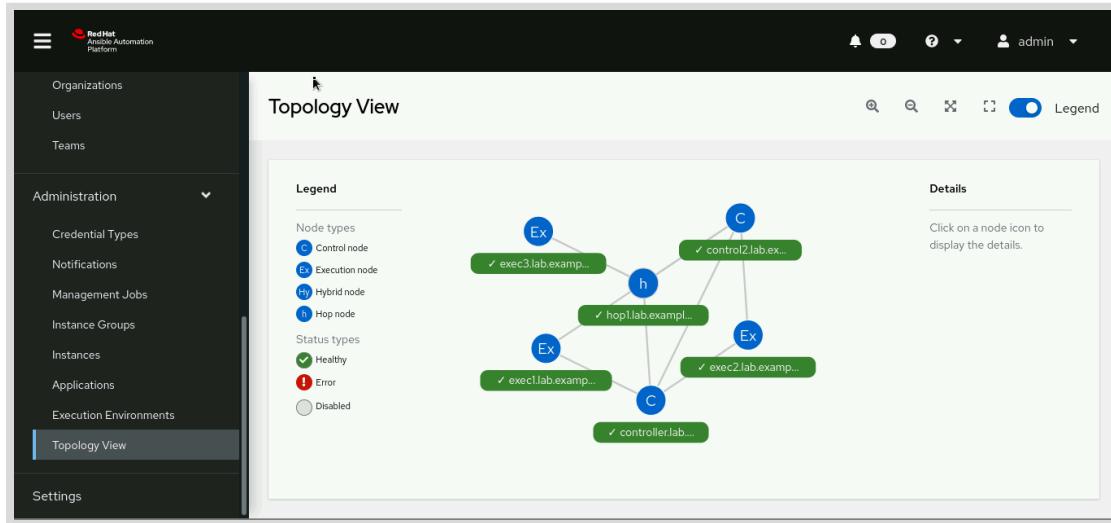
Testing Execution Plane Resilience

1. Navigate to **Resources > Templates** and click the **Launch Template** icon for the **Demo Job Template** resource.
2. After the job completes, click the **Details** tab. Make note of the instance group and execution node used by the job.
3. Navigate to **Administration > Instance Groups** and then click the link for the instance group identified in the previous step.
4. On the **Instances** tab, disable the previously identified execution node by setting **Enabled** to off. The status changes to **Disabled**.
5. Navigate to **Views > Jobs** and then click the **Relaunch Job** icon for the **Demo Job Template** job.

6. After the job completes, click the **Details** tab and notice that the job used a different execution node.
7. Navigate to **Administration > Instances** and then enable the previously disabled execution node. Set **Disabled** to off. The status changes to **Enabled**.

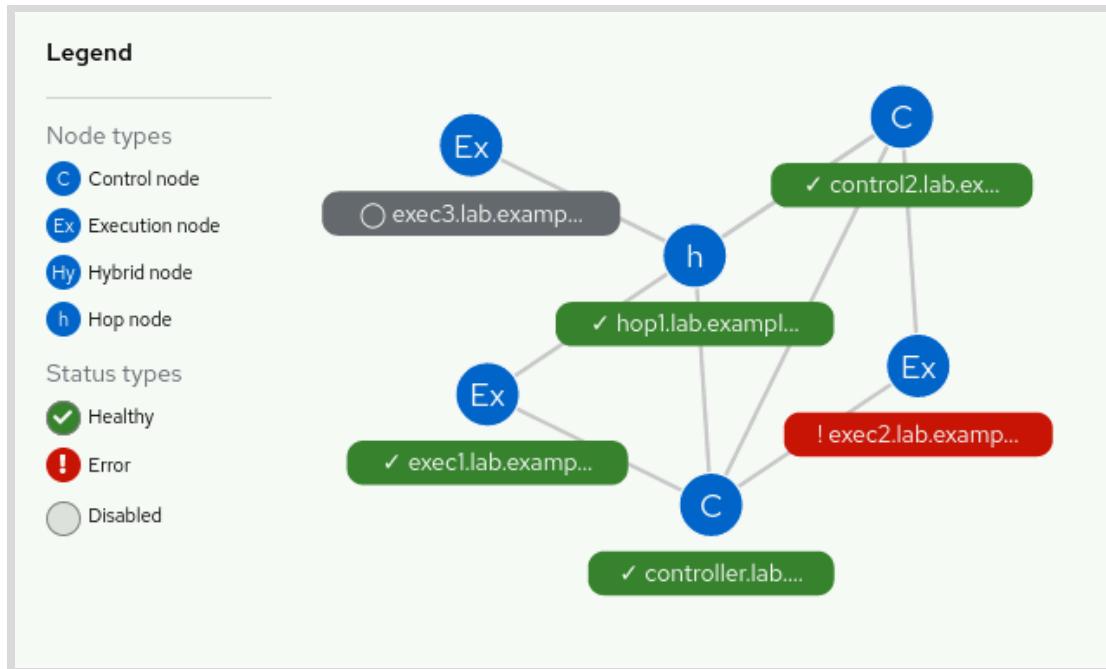
Monitoring Automation Mesh from the Web UI

Navigate to **Administration > Topology View** in the automation controller web UI to see an overview of the current status of automation mesh and its nodes.



- Healthy nodes are marked with a checkmark and are colored green.
- Unavailable nodes are marked with an exclamation point and are colored red.
- Disabled nodes are marked with a circle and are colored gray.

In the following example, the **control2**, **controller**, **exec1**, and **hop1** nodes are healthy. The **exec2** node is in an error state and the **exec3** node is disabled.



You can use the icons at the upper-right of the **Topology View** page to zoom in and out, resize the diagram to fit the screen, reset the zoom to its default level, and to turn on and off the descriptive legend. You can click and drag to change the position of the diagram, and use your mouse wheel to zoom in and out.

If you hover over any of the nodes, the web UI highlights the lines representing the peer relationships between that node and the other nodes in automation mesh. If you click any of the nodes, the web UI displays additional information about that node under **Details** to the right of the diagram.

Monitoring Automation Mesh from the Command Line

This section demonstrates useful commands for monitoring and troubleshooting automation mesh. Log in as the awx user on one of the automation controller machines and run the following commands.

Listing Nodes and Instance Groups

You can use the `awx-manage list_instances` command to list all the instances in the mesh. The command shows the status of each node.

- Active and available nodes appear in green. These nodes display a **Healthy** status in the automation controller web UI.
- Unavailable nodes appear in red and the version of `ansible-runner` displays as question marks. These nodes display an **Error** status in the automation controller web UI.
- Disabled nodes contain the `[DISABLED]` text and appear in gray. These nodes display **Disabled** in the automation controller web UI.

In the following example, the `control2`, `controller`, `exec1`, and `hop1` nodes are active and available. The `exec2` node is unavailable and the `exec3` node is disabled.

```
[awx@controller ~]$ awx-manage list_instances
[controlplane capacity=53 policy=100%]
control2.lab.example.com capacity=16 node_type=control ...
controller.lab.example.com capacity=37 node_type=control ...

[default capacity=16 policy=100%]
exec1.lab.example.com capacity=16 node_type=execution version=ansible-runner...
exec2.lab.example.com capacity=0 node_type=execution version=ansible-runner-??
[DISABLED] exec3.lab.example.com capacity=0 node_type=execution version=ansi...

[ungrouped capacity=0]
hop1.lab.example.com node_type=hop heartbeat="2022-06-01 17:46:51"
```

Monitoring Automation Mesh from the Command Line

You can use the `receptorctl` command to test communication on the automation mesh. The `receptorctl` command provides several subcommands, including:

- `receptorctl status` to get the status of the entire automation mesh.
- `receptorctl ping` to test connectivity between the current node and another node in the automation mesh.
- `receptorctl traceroute` to determine the route and latency of communication on the automation mesh between the current node and another node.

The command requires that you specify the `systemd` socket unit for the automation mesh `receptor` service. The following examples use the `/var/run/awx-receptor/receptor.sock` unit.

Use the `status` subcommand to view the entire mesh, including all of the nodes and how the nodes are connected.

In this example, the `Route` section indicates that communication to the `exec3.lab.example.com` execution node is routed through the `hop1.lab.example.com` hop node.

```
[awx@controller ~]# receptorctl --socket /var/run/awx-receptor/receptor.sock \
> status
Node ID: controller.lab.example.com
Version: 1.2.3
System CPU Count: 4
System Memory MiB: 5752

Connection          Cost
exec1.lab.example.com   1
exec2.lab.example.com   1
control2.lab.example.com 1
hop1.lab.example.com    1

Known Node           Known Connections
control2.lab.example.com controller.lab.example.com: 1 exec1.lab.example.co...
controller.lab.example.com control2.lab.example.com: 1 exec1.lab.example.com:...
exec1.lab.example.com   control2.lab.example.com: 1 controller.lab.example...
exec2.lab.example.com   control2.lab.example.com: 1 controller.lab.example...
```

exec3.lab.example.com	hop1.lab.example.com: 1
hop1.lab.example.com	control2.lab.example.com: 1 controller.lab.example...
Route	Via
control2.lab.example.com	control2.lab.example.com
exec1.lab.example.com	exec1.lab.example.com
exec2.lab.example.com	exec2.lab.example.com
exec3.lab.example.com	hop1.lab.example.com
hop1.lab.example.com	hop1.lab.example.com
Node	Service Type ... Tags
exec1.lab.example.com	control StreamTLS ... {'type': 'Control Service'}
exec2.lab.example.com	control StreamTLS ... {'type': 'Control Service'}
control2.lab.example.com	control StreamTLS ... {'type': 'Control Service'}
controller.lab.example.com	control StreamTLS ... {'type': 'Control Service'}
exec3.lab.example.com	control StreamTLS ... {'type': 'Control Service'}
hop1.lab.example.com	control StreamTLS ... {'type': 'Control Service'}
Node	Secure Work Types
exec1.lab.example.com	ansible-runner
exec2.lab.example.com	ansible-runner
control2.lab.example.com	local, kubernetes-runtime-auth, kubernetes-inclust...
controller.lab.example.com	local, kubernetes-runtime-auth, kubernetes-inclust...
exec3.lab.example.com	ansible-runner

Use the `ping` subcommand to test connectivity between the current host and another host.

The following example tests connectivity between the `controller.lab.example.com` and `exec2.lab.example.com` hosts.

```
[awx@controller ~]# receptorctl --socket /var/run/awx-receptor/receptor.sock \
> ping exec2.lab.example.com
Reply from exec2.lab.example.com in 1.461675ms
Reply from exec2.lab.example.com in 504.934µs
Reply from exec2.lab.example.com in 528.547µs
Reply from exec2.lab.example.com in 722.001µs
```

Use the `traceroute` subcommand to view the route between nodes. In the following example, the `controller.lab.example.com` node connects to the `exec3.lab.example.com` node through the `hop1.lab.example.com` node.

```
[awx@controller ~]# receptorctl --socket /var/run/awx-receptor/receptor.sock \
> traceroute exec3.lab.example.com
0: controller.lab.example.com in 507.316µs
1: hop1.lab.example.com in 1.032767ms
2: exec3.lab.example.com in 820.719µs
```



References

Red Hat Ansible Automation Platform Automation Mesh Guide

https://access.redhat.com/documentation/en-us/red_hat_ansible_automation_platform/2.1/html-single/red_hat_ansible_automation_platform_automation_mesh_guide/index

► Guided Exercise

Managing Distributed Execution with Automation Mesh

Launch jobs from your controllers, observe them running on your execution nodes, and make adjustments to your automation mesh.

Outcomes

- Create instance groups and associate execution nodes with instance groups.
- Configure inventories to use an instance group.

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that automation mesh is installed. If automation mesh is not installed, then you must complete the previous exercise or run the `lab start mesh-deploy-solve` command prior to running the `lab` command for this exercise.

```
[student@workstation ~]$ lab start mesh-manage
```

Instructions

- 1. Create the `Development Datacenter` instance group and add the `exec1.lab.example.com` and `exec2.lab.example.com` hosts to it.
- 1.1. Navigate to `https://controller.lab.example.com` and log in as the `admin` user with `redhat` as the password.



Note

As an alternative, navigate to `https://control2.lab.example.com` and log in as the `admin` user with `redhat` as the password. Because both `controller.lab.example.com` and `control2.lab.example.com` are control nodes using a shared database, you can complete this exercise from either host.

- 1.2. Navigate to `Administration > Instance Groups` and then click `Add > Add instance group`.
- 1.3. Create the instance group using `Development Datacenter` as the name and then click `Save`.
- 1.4. Click the `Instances` tab and then click `Associate`.
- 1.5. Select the `exec1.lab.example.com` and `exec2.lab.example.com` hosts and then click `Save`.

- ▶ **2.** Create the **Production Datacenter** instance group and add the `exec3.lab.example.com` host to it.
 - 2.1. Navigate to **Administration > Instance Groups** and then click **Add > Add instance group**.
 - 2.2. Create the instance group using **Production Datacenter** as the name and then click **Save**.
 - 2.3. Click the **Instances** tab and then click **Associate**.
 - 2.4. Select the `exec3.lab.example.com` host and then click **Save**.
- ▶ **3.** Update the **Dev** inventory to use the **Development Datacenter** instance group.
 - 3.1. Navigate to **Resources > Inventories** and then click the **Edit Inventory** icon for the **Dev** inventory.
 - 3.2. Click the search icon for the **Instance Groups** field, select the **Development Datacenter** instance group, and then click **Select**.
 - 3.3. Click **Save**.
- ▶ **4.** Update the **Prod** inventory to use the **Production Datacenter** instance group.
 - 4.1. Navigate to **Resources > Inventories** and then click the **Edit Inventory** icon for the **Prod** inventory.
 - 4.2. Click the search icon for the **Instance Groups** field, select the **Production Datacenter** instance group, and then click **Select**.
 - 4.3. Click **Save**.
- ▶ **5.** Launch the **Deploy MOTD** job template and select the **Dev** inventory. Identify the instance group and execution node used by the job.
 - 5.1. Navigate to **Resources > Templates** and click the **Launch Template** icon for the **Deploy MOTD** job template.
 - 5.2. Select the **Dev** inventory and then click **Next**.
 - 5.3. Click **Next** to use the default answer to the survey question, and then click **Launch**.
 - 5.4. After the job completes, click the **Details** tab. The job used the **Development Datacenter** instance group and used either the `exec1.lab.example.com` or `exec2.lab.example.com` execution node.

**Note**

Be patient because it might take more than 30 seconds for the job to complete.

- ▶ **6.** Launch the **Deploy MOTD** job template and select the **Prod** inventory. Observe how jobs that use the **Prod** inventory use the **Production Datacenter** instance group and the `exec3.lab.example.com` execution node.

- 6.1. Navigate to **Resources > Templates** and click the **Launch Template** icon for the **Deploy MOTD** job template.
- 6.2. Select the **Prod** inventory and click **Next**.
- 6.3. Use **Production push** as the answer to the survey question and then click **Next**.
- 6.4. Click **Launch**.
- 6.5. After the job completes, click the **Details** tab. The job used the **Production Datacenter** instance group and the **exec3.lab.example.com** execution node.

**Note**

Be patient because it might take more than 30 seconds for the job to complete.

- 7. View the message of the day for hosts in the **Dev** and **Prod** inventories. The date stamps in the output of your commands differs from the example output shown in the following instructions.
- 7.1. Display the message of the day for hosts in the **Dev** inventory. Connect to **servera.lab.example.com**, notice the message of the day, and then disconnect from the server.

```
[student@workstation ~]$ ssh student@servera.lab.example.com
2022-05-25: Nothing new to report.
...output omitted...
[student@servera ~]$ exit
logout
Connection to servera.lab.example.com closed.
```

Connect to **serverb.lab.example.com**, notice the message of the day, and then disconnect from the server.

```
[student@workstation ~]$ ssh student@serverb.lab.example.com
2022-05-25: Nothing new to report.
...output omitted...
[student@serverb ~]$ exit
logout
Connection to serverb.lab.example.com closed.
```

- 7.2. Display the message of the day for hosts in the **Prod** inventory. Connect to **servere.lab.example.com**, notice the message of the day, and then disconnect from the server.

```
[student@workstation ~]$ ssh student@servere.lab.example.com
2022-05-25: Production push
...output omitted...
[student@servere ~]$ exit
logout
Connection to servere.lab.example.com closed.
```

Connect to **serverf.lab.example.com**, notice the message of the day, and then disconnect from the server.

```
[student@workstation ~]$ ssh student@serverf.lab.example.com
2022-05-25: Production push
...output omitted...
[student@serverf ~]$ exit
logout
Connection to serverf.lab.example.com closed.
```

Finish

On the **workstation** machine, change to the **student** user home directory and use the **lab** command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish mesh-manage
```

► Quiz

Building a Large-scale Red Hat Ansible Automation Platform Deployment

Choose the correct answers to the following questions:

► 1. **What is automation mesh?**

- a. A centralized service operated by Red Hat that provides Ansible Content Collections to your automation infrastructure.
- b. The content distribution network used to deliver container images from `registry.redhat.io` to your private automation hub.
- c. An overlay network intended to scale and improve the distribution of automation tasks across a large and dispersed enterprise.
- d. The set of all managed hosts automated with your Red Hat Ansible Automation Platform installation.

► 2. **Which two node types are found on the control plane? (Choose two.)**

- a. Hybrid nodes
- b. Execution nodes
- c. Hop nodes
- d. Control nodes

► 3. **Which two instance groups are created by default? (Choose two.)**

- a. `controlplane`
- b. `local`
- c. `remote`
- d. `default`

► 4. **Which two answers are ways to create instance groups? (Choose two.)**

- a. Run the `setup.sh` installation script, using the `--create-instance-group=instance_group_name` option.
- b. Run the `Create_instance` job template included with Red Hat Ansible Automation Platform.
- c. Define instance groups in the inventory file and run the `setup.sh` installation script.
- d. Create instance groups using the automation controller web UI.

- 5. If you define an instance group in both an inventory and a job template that uses that inventory, then which instance group takes precedence?
- The instance group defined in the inventory.
 - The instance group defined in the job template.
 - Neither, default is used if the two conflict.
- 6. Which two validation checks are run by the installer script? (Choose two.)
- Confirm that all hosts in the automationcontroller group are in the execution_nodes group.
 - Confirm that hosts do not peer with a host that does not exist.
 - Confirm that all hop nodes are peered with all control nodes.
 - Confirm that hosts do not have both inbound and outbound connections to the same nodes (so that they do not form a loop).

► Solution

Building a Large-scale Red Hat Ansible Automation Platform Deployment

Choose the correct answers to the following questions:

► 1. What is automation mesh?

- a. A centralized service operated by Red Hat that provides Ansible Content Collections to your automation infrastructure.
- b. The content distribution network used to deliver container images from `registry.redhat.io` to your private automation hub.
- c. An overlay network intended to scale and improve the distribution of automation tasks across a large and dispersed enterprise.
- d. The set of all managed hosts automated with your Red Hat Ansible Automation Platform installation.

► 2. Which two node types are found on the control plane? (Choose two.)

- a. Hybrid nodes
- b. Execution nodes
- c. Hop nodes
- d. Control nodes

► 3. Which two instance groups are created by default? (Choose two.)

- a. `controlplane`
- b. `local`
- c. `remote`
- d. `default`

► 4. Which two answers are ways to create instance groups? (Choose two.)

- a. Run the `setup.sh` installation script, using the `--create-instance-group=instance_group_name` option.
- b. Run the `Create_instance` job template included with Red Hat Ansible Automation Platform.
- c. Define instance groups in the inventory file and run the `setup.sh` installation script.
- d. Create instance groups using the automation controller web UI.

- **5. If you define an instance group in both an inventory and a job template that uses that inventory, then which instance group takes precedence?**
- a. The instance group defined in the inventory.
 - b. The instance group defined in the job template.
 - c. Neither, default is used if the two conflict.
- **6. Which two validation checks are run by the installer script? (Choose two.)**
- a. Confirm that all hosts in the automationcontroller group are in the execution_nodes group.
 - b. Confirm that hosts do not peer with a host that does not exist.
 - c. Confirm that all hop nodes are peered with all control nodes.
 - d. Confirm that hosts do not have both inbound and outbound connections to the same nodes (so that they do not form a loop).

Summary

- Automation mesh is an overlay network intended to scale and improve the distribution of work across a large and dispersed enterprise.
- Automation mesh provides resilience and scalability to Red Hat Ansible Automation Platform using a combination of control nodes, hybrid nodes, hop nodes, and execution nodes.
- The control plane performs functions such as synchronizing project updates and running management jobs. The execution plane executes automation on behalf of the control plane.
- To add and remove nodes from automation mesh, edit the inventory file used by the installation script, and then run the installation script.
- An instance group is a set of execution nodes that can run jobs, and are often used to schedule jobs on execution nodes close to the managed hosts on which those jobs automate tasks.
- Instance groups can be configured by the installation script or through automation controller's web UI.

Chapter 12

Comprehensive Review

Goal

Review tasks from *Managing Enterprise Automation with Red Hat Ansible Automation Platform*.

Sections

- Comprehensive Review

Lab

- Deploying and Operating an Automation Mesh
- Adding Users and Teams
- Uploading Automation Execution Environments to Private Automation Hub
- Creating an Inventory Managed as a Project
- Configuring Job Templates
- Configuring Workflow Job Templates, Surveys, and Notifications
- Operating Automation Controller using the API
- Backup and Restore Red Hat Ansible Automation Platform

Comprehensive Review

Objectives

After completing this section, you should have reviewed and refreshed the knowledge and skills that you learned in *Managing Enterprise Automation with Red Hat Ansible Automation Platform*.

Reviewing Managing Enterprise Automation with Red Hat Ansible Automation Platform

Before beginning the comprehensive review for this course, you should be comfortable with the topics covered in each chapter. Do not hesitate to ask the instructor for extra guidance or clarification on these topics.

Chapter 1, Installing Red Hat Ansible Automation Platform

Explain what Red Hat Ansible Automation Platform is and perform a basic installation of automation controller and private automation hub.

Chapter 2, Managing User Access

Create user accounts and organize them into teams and groups in automation controller and private automation hub, respectively, and assign them permissions to administer and access resources in each service.

Chapter 3, Managing Inventories and Machine Credentials

Create inventories of machines to manage, and configure credentials necessary for automation controller's execution nodes to log in and run Ansible jobs on those systems.

Chapter 4, Managing Projects and Launching Ansible Jobs

Create projects and job templates in the web UI, and use them to launch Ansible Playbooks that are stored in Git repositories in order to automate tasks on managed hosts.

Chapter 5, Advanced Job Configuration

Configure advanced features of automation controller in order to more effectively and efficiently implement jobs.

Chapter 6, Constructing Job Workflows

Use advanced features of job templates to improve performance, simplify the customization of jobs, launch multiple jobs, and provide notification of job results.

Chapter 7, Managing Advanced Inventories

Manage inventories that are generated dynamically from scripts or the automation controller smart inventory feature.

Chapter 8, Automating Configuration of Ansible Automation Platform

Automate the configuration and deployment of Red Hat Ansible Automation Platform services by using Ansible Content Collections, the automation controller API, and Git webhooks.

Chapter 9, Maintaining Red Hat Ansible Automation Platform

Perform routine maintenance and administration of Red Hat Ansible Automation Platform.

Chapter 10, Getting Insights into Automation Performance

Get information from Red Hat Insights for Red Hat Ansible Automation Platform to evaluate the performance of your Ansible automation and identify possible ways to improve it.

Chapter 11, Building a Large-scale Red Hat Ansible Automation Platform Deployment

Use high availability techniques and automation mesh to scale up your Red Hat Ansible Automation Platform deployment.

▶ Lab

Deploying and Operating an Automation Mesh

Install Red Hat Ansible Automation Platform configured to use an automation controller, a private automation hub, and automation mesh.

Outcomes

- Configure an inventory file to install Red Hat Ansible Automation Platform, which includes an automation controller, a private automation hub, and automation mesh.
- Create instance groups and assign execution nodes to the instance groups.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command removes resources created in previous chapters and downloads and extracts the Red Hat Ansible Automation Platform installation bundle archive to the `/home/student/aap2.2-bundle` directory.



Important

If you have already installed automation mesh, or if you want to perform this exercise again, then the most efficient method is to reset your classroom environment. Refer to the *Orientation to the Classroom Environment* section for instructions on how to reset the classroom environment to its original state.

Resetting your classroom environment removes all previous work.

```
[student@workstation ~]$ lab start compreview-mesh
```

Specifications

Use the installation bundle in the `/home/student/aap2.2-bundle` directory to install Red Hat Ansible Automation Platform and automation mesh, based on the following specification.

- Configure the `[automationcontroller]` and `[automationcontroller:vars]` sections of the `/home/student/aap2.2-bundle/inventory` file using the following information:
 - Add the `controller.lab.example.com` and `control2.lab.example.com` hosts as control nodes.
 - Do not explicitly configure any peering connections.

Chapter 12 | Comprehensive Review

- Using the `web_server_ssl_cert` and `web_server_ssl_key` variables, configure each host to use its own web certificate and private key. Use the certificates and private keys in the `/home/student/certs` directory.
- Configure the `[execution_nodes]` section of the `inventory` file using the following information:
 - Add the `exec1.lab.example.com` host as an execution node and have it peer with hosts that are defined in the `[automationcontroller]` section.
 - Add the `exec2.lab.example.com` and `exec3.lab.example.com` hosts as execution nodes and have both hosts peer with the `hop1.lab.example.com` host.
 - Add the `hop1.lab.example.com` host as a hop node and have it peer with hosts defined in the `[automationcontroller]` section.
- Configure the `[automationhub]` section of the `inventory` file using the following information:
 - Install the private automation hub on the `hub.lab.example.com` host.
- Configure the `[database]` section of the `inventory` file using the following information:
 - Add the `db.lab.example.com` host.
- Configure the `[all:vars]` section of the `inventory` file with the following lines. Do not modify the other lines in that section.

```
admin_password='redhat'
pg_host='db.lab.example.com'
pg_password='redhat'
registry_url='hub.lab.example.com'
registry_username='admin'
registry_password='redhat'
automationhub_admin_password='redhat'
automationhub_pg_host='db.lab.example.com'
automationhub_pg_password='redhat'
custom_ca_cert=/home/student/certs/classroom-ca.pem
automationhub_ssl_cert=/home/student/certs/hub.lab.example.com.crt
automationhub_ssl_key=/home/student/certs/hub.lab.example.com.key
postgres_use_ssl=True
postgres_ssl_cert=/home/student/certs/db.lab.example.com.crt
postgres_ssl_key=/home/student/certs/db.lab.example.com.key
```

- Validate the automation mesh configuration that is defined in the `/home/student/aap2.2-bundle/inventory` file.
 - Run the following command in the `/home/student/aap2.2-bundle` directory to generate a topology file:

```
[student@workstation aap2.2-bundle]$ ./setup.sh --tag generate_dot_file
...output omitted...
```

- Render the generated topology file using the `dot` command, provided by the `graphviz` package. The graphic should look similar to the following example.

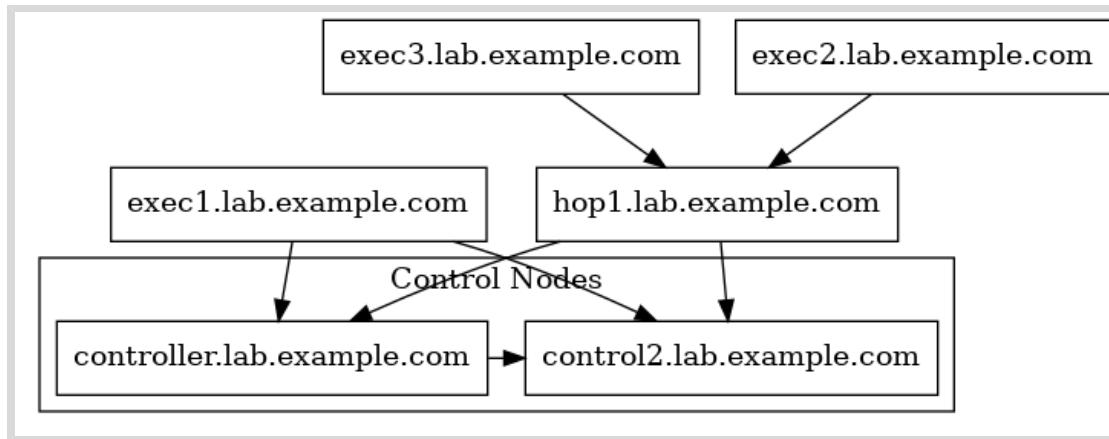


Figure 12.1: Expected automation mesh topology

- As the `root` user, run the `/home/student/aap2.2-bundle/setup.sh` script to install Red Hat Ansible Automation Platform. Because the systems in the classroom environment do not meet the minimum hardware requirements for installation, you must use the `-e ignore_preflight_errors=true` option when you run the script.
- After the installation completes, go to `https://controller.lab.example.com` and log in as the `admin` user with `redhat` as the password. Create two instance groups and assign execution nodes to those groups using the following table:

	Instance group name	Instances
Instance Group 1	<code>public</code>	<code>exec1.lab.example.com</code>
Instance Group 2	<code>internal</code>	<code>exec2.lab.example.com</code> <code>exec3.lab.example.com</code>

- Use the automation controller web UI to verify the automation mesh installation and the instance groups specified in this exercise.

Evaluation

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade comprevew-mesh
```

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish comprevew-mesh
```

► Solution

Deploying and Operating an Automation Mesh

Install Red Hat Ansible Automation Platform configured to use an automation controller, a private automation hub, and automation mesh.

Outcomes

- Configure an inventory file to install Red Hat Ansible Automation Platform, which includes an automation controller, a private automation hub, and automation mesh.
- Create instance groups and assign execution nodes to the instance groups.

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command removes resources created in previous chapters and downloads and extracts the Red Hat Ansible Automation Platform installation bundle archive to the `/home/student/aap2.2-bundle` directory.



Important

If you have already installed automation mesh, or if you want to perform this exercise again, then the most efficient method is to reset your classroom environment. Refer to the *Orientation to the Classroom Environment* section for instructions on how to reset the classroom environment to its original state.

Resetting your classroom environment removes all previous work.

```
[student@workstation ~]$ lab start compreview-mesh
```

- Configure the `/home/student/aap2.2-bundle/inventory` file for automation mesh.

- Change to the `/home/student/aap2.2-bundle` directory.

```
[student@workstation ~]$ cd ~/aap2.2-bundle/
```

- Add the `controller.lab.example.com` and `control2.lab.example.com` hosts to the `[automationcontroller]` section of the inventory file. When completed, the inventory file contains the following content for the `[automationcontroller]` section:

Chapter 12 | Comprehensive Review

```
[automationcontroller]
controller.lab.example.com
control2.lab.example.com
```

- 1.3. Configure variables in the [automationcontroller:vars] section.

- Remove the existing peers=execution_nodes line so that you can define peer connections in the [execution_nodes] section.
- Add the node_type=control line to indicate that the control nodes should be configured as the control node type.
- Configure the web_server_ssl_cert and web_server_ssl_key variables for each control node using Jinja2 variables. Alternatively, configure these variables in the [automationcontroller] section using the absolute path to each file.

When completed, the inventory file contains the following content for the [automationcontroller:vars] section:

```
[automationcontroller:vars]
node_type=control
web_server_ssl_cert=/home/student/certs/{{ inventory_hostname }}.crt
web_server_ssl_key=/home/student/certs/{{ inventory_hostname }}.key
```

- 1.4. Configure variables in the [execution_nodes] section.

- Add the exec1.lab.example.com host as an execution node and have it peer with hosts defined in the [automationcontroller] section.
- Add the exec2.lab.example.com and exec3.lab.example.com hosts as execution nodes and have both hosts peer with the hop1.lab.example.com host.
- Add the hop1.lab.example.com host as a hop node and have it peer with hosts defined in the [automationcontroller] section.

When completed, the inventory file contains the following content for the [execution_nodes] section:

```
[execution_nodes]
exec1.lab.example.com peers=automationcontroller
exec2.lab.example.com peers=hop1.lab.example.com
exec3.lab.example.com peers=hop1.lab.example.com
hop1.lab.example.com node_type=hop peers=automationcontroller
```

- 1.5. Add the hub.lab.example.com host to the [automationhub] section.

When completed, the inventory file contains the following content for the [automationhub] section:

```
[automationhub]
hub.lab.example.com
```

- 1.6. Add the db.lab.example.com host to the [database] section. When completed, the inventory file contains the following content for the [database] section:

```
[database]
db.lab.example.com
```

- 1.7. Configure common variables in the [all:vars] section. All other variables should retain their original value from the inventory file.

```
admin_password='redhat'
pg_host='db.lab.example.com'
pg_password='redhat'
registry_url='hub.lab.example.com'
registry_username='admin'
registry_password='redhat'
automationhub_admin_password='redhat'
automationhub_pg_host='db.lab.example.com'
automationhub_pg_password='redhat'
custom_ca_cert=/home/student/certs/classroom-ca.pem
automationhub_ssl_cert=/home/student/certs/hub.lab.example.com.crt
automationhub_ssl_key=/home/student/certs/hub.lab.example.com.key
postgres_use_ssl=True
postgres_ssl_cert=/home/student/certs/db.lab.example.com.crt
postgres_ssl_key=/home/student/certs/db.lab.example.com.key
```



Note

The /home/student/comprevew-mesh directory contains inventory files that you can use for comparison.

2. Validate the automation mesh configuration defined in the /home/student/aap2.2-bundle/inventory file.
- 2.1. Use the ./setup.sh --tag generate_dot_file command to generate a topology file.

```
[student@workstation aap2.2-bundle]$ ./setup.sh --tag generate_dot_file
...output omitted...
TASK [debug] *****
ok: [controller.lab.example.com] => {
    "msg": "Ansible Mesh topology graph created at 'mesh-topology.dot'. To render
your dot graph, you could run: dot -Tjpg mesh-topology.dot -o graph-topology.jpg
\n"
}
...output omitted...
```

- 2.2. Install the graphviz package. When prompted, enter student for the password and then install the package.

```
[student@workstation aap2.2-bundle]$ sudo yum install graphviz
[sudo] password for student: student
...output omitted...
```

- 2.3. Render the generated topology file using the dot command.

Chapter 12 | Comprehensive Review

```
[student@workstation aap2.2-bundle]$ dot -Tjpg mesh-topology.dot \
> -o graph-topology.jpg
```

- 2.4. Open the /home/student/aap2.2-bundle/graph-topology.jpg file in a web browser. The graphic should look similar to the following example.

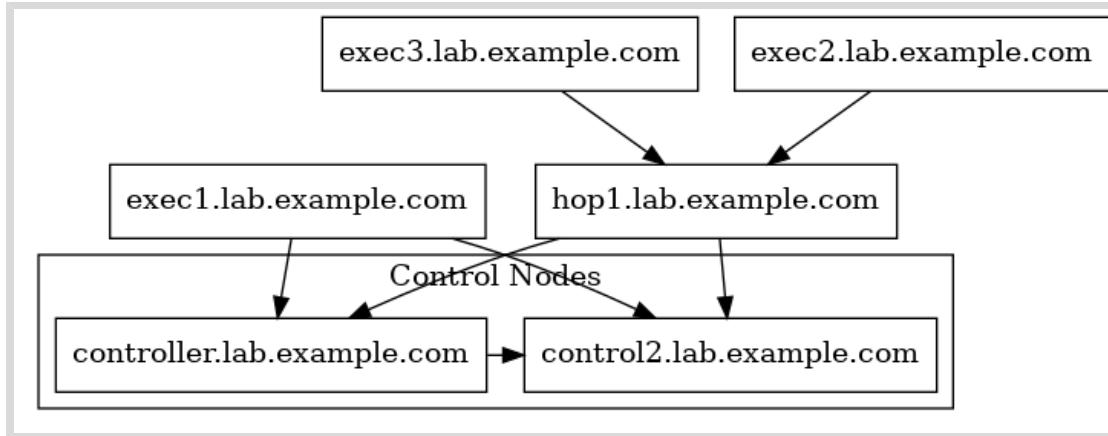


Figure 12.17: Expected automation mesh topology

3. Initiate the Ansible Automation Platform installation as the `root` user using the `/home/student/aap2.2-bundle/setup.sh` script. Because the systems in the classroom environment do not meet the minimum installation requirements, you must add the `-e ignore_preflight_errors=true` option.

- 3.1. Use the `sudo` command to change to the `root` user, using `student` as the password.

```
[student@workstation aap2.2-bundle]$ sudo -i
[sudo] password for student: student
[root@workstation ~]#
```

- 3.2. Change to the `/home/student/aap2.2-bundle` directory.

```
[root@workstation ~]# cd /home/student/aap2.2-bundle/
```

- 3.3. Run the `setup.sh` script with `-e ignore_preflight_errors=true` set to ignore the results of checks it makes before the installation starts. (The classroom systems have less RAM than is optimal for a production installation.) The installation takes approximately 15 minutes to complete. The output of your installation might be slightly different from the following output.

```
[root@workstation aap2.2-bundle]# ./setup.sh -e ignore_preflight_errors=true
...output omitted...
PLAY RECAP ****
control2.lab.example.com : ok=246 changed=129 ... failed=0 ... ignored=5
controller.lab.example.com : ok=263 changed=52 ... failed=0 ... ignored=1
db.lab.example.com : ok=73 changed=12 ... failed=0 ... ignored=1
exec1.lab.example.com : ok=104 changed=51 ... failed=0 ... ignored=3
exec2.lab.example.com : ok=104 changed=51 ... failed=0 ... ignored=3
exec3.lab.example.com : ok=104 changed=51 ... failed=0 ... ignored=3
```

```
hop1.lab.example.com      : ok=83  changed=36  ...  failed=0  ...  ignored=2
hub.lab.example.com       : ok=195  changed=22  ...  failed=0  ...  ignored=1
localhost                 : ok=3    changed=1   ...  failed=0  ...  ignored=0
```

```
The setup process completed successfully.
[warn] /var/log/tower does not exist. Setup log saved to setup.log.
```

- 3.4. Exit from the root user session.

```
[root@workstation aap2.2-bundle]# exit
```

4. Create two automation controller instance groups and assign execution nodes to the instance groups using the following table:

	Instance group name	Instances
Instance Group 1	public	exec1.lab.example.com
Instance Group 2	internal	exec2.lab.example.com exec3.lab.example.com

- 4.1. Go to <https://controller.lab.example.com> and log in as the admin user with redhat as the password.
- 4.2. Go to Administration > Instance Groups and then click Add > Add instance group.
- 4.3. Create the first instance group using public as the name and then click Save.
- 4.4. Click the Instances tab and then click Associate.
- 4.5. Select the exec1.lab.example.com host and then click Save.
- 4.6. Go to Administration > Instance Groups and then click Add > Add instance group.
- 4.7. Create the second instance group using internal as the name and then click Save.
- 4.8. Click the Instances tab and then click Associate.
- 4.9. Select the exec2.lab.example.com and exec3.lab.example.com hosts and then click Save.
5. Use the automation controller web UI to verify the automation mesh installation and the instance groups specified in this exercise.
 - 5.1. Go to Administration > Topology View. Your automation mesh environment should display a graphic similar to the following:

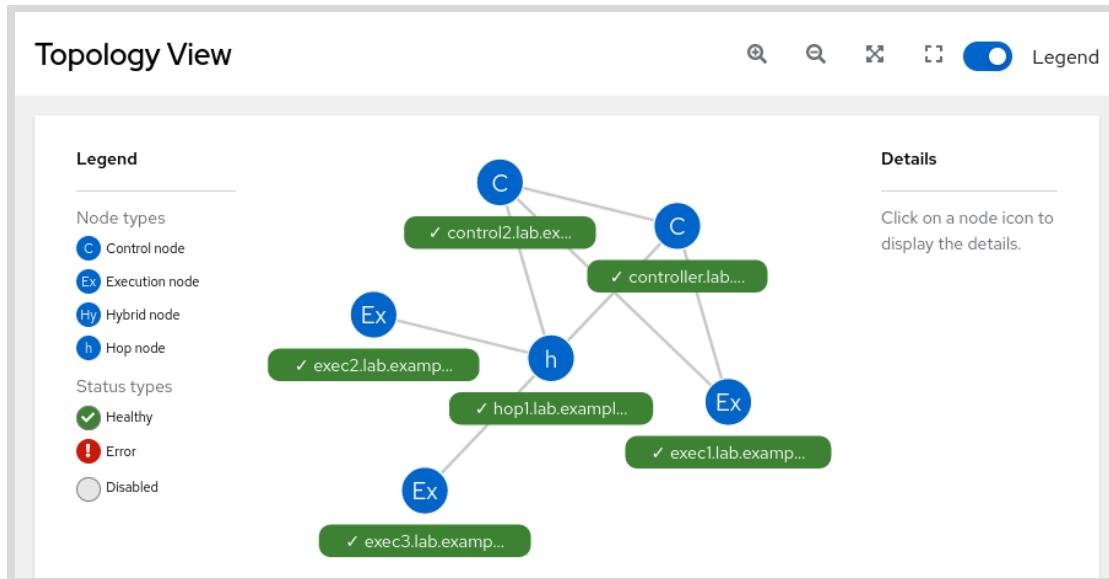


Figure 12.18: Automation mesh topology view

- All nodes are healthy.
- The node type of each node matches the exercise specifications.
- The execution nodes can communicate with the control nodes.

5.2. Go to Administration > Instance Groups. Your instance groups should match the following:

Instance Groups							
	Name	Type	Running Jobs	Total Jobs	Instances	Capacity	Actions
<input type="checkbox"/>	controlplane	Instance group	0	0	2	Used capacity <div style="width: 100%;"><div style="width: 100%;"> </div></div> 0%	
<input type="checkbox"/>	default	Instance group	0	0	3	Used capacity <div style="width: 100%;"><div style="width: 100%;"> </div></div> 0%	
<input type="checkbox"/>	internal	Instance group	0	0	2	Used capacity <div style="width: 100%;"><div style="width: 100%;"> </div></div> 0%	
<input type="checkbox"/>	public	Instance group	0	0	1	Used capacity <div style="width: 100%;"><div style="width: 100%;"> </div></div> 0%	

Figure 12.19: Automation controller instance groups

- The installer created the `controlplane` instance group and associated the `controller.lab.example.com` and `control2.lab.example.com` instances with the instance group.
- The installer created the `default` instance group and associated the `exec1.lab.example.com`, `exec2.lab.example.com`, and `exec3.lab.example.com` instances with the instance group.
- You created the `internal` instance group and associated the `exec2.lab.example.com` and `exec3.lab.example.com` instances with the instance group.
- You created the `public` instance group and associated the `exec1.lab.example.com` instance with the instance group.

Evaluation

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade compreview-mesh
```

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish compreview-mesh
```

▶ Lab

Adding Users and Teams

Create new users and a new team in automation controller and create users and groups in a private automation hub.

Outcomes

- Create organizations, users, and teams in automation controller.
- Create users and groups in a private automation hub.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command ensures that automation controller and private automation hub are installed. The command also removes resources created in previous chapters and downloads content collections to the `/home/student/content-collections` directory.

```
[student@workstation ~]$ lab start compreview-users
```

Specifications

Configure organizations, users, and teams on your automation controller at `https://controller.lab.example.com` based on the following specification. The `admin` user on your automation controller has `redhat` as its password.

In addition, configure users and groups on your private automation hub at `https://hub.lab.example.com`, based on the following specification. An Ansible Content Collection also must be uploaded to your private automation hub and approved for use. The `admin` user on your private automation hub also has `redhat` as its password.

- On your automation controller, create an organization based on the following information:

Field	Value
Name	BLA
Description	Business Line Applications

- On your automation controller, create a team based on the following information:

Field	Value
Name	SRE
Description	Site Reliability Engineering
Organization	BLA

Chapter 12 | Comprehensive Review

- On your automation controller, create three users based on the following information:

Field	Value
Username	sre1
Password	redhat123
Confirm Password	redhat123
User Type	Normal User
Organization	BLA

Field	Value
Username	sre2
Password	redhat123
Confirm Password	redhat123
User Type	Normal User
Organization	BLA

Field	Value
Username	sysadmin
Password	redhat123
Confirm Password	redhat123
User Type	System Administrator
Organization	BLA

- On your automation controller, assign team roles to users on the SRE team based on the following information:

User	Role
sre1	Admin
sre2	Member

- On your private automation hub, add a group named **Infrastructure**, and assign that group permissions that allow it to manage Ansible Content Collections and containers.
- On your private automation hub, add two users using the information in the following table.

Field	Value for the first user	Value for the second user
Username	infra1	super

Field	Value for the first user	Value for the second user
Password	redhat123	redhat123
Password confirmation	redhat123	redhat123
Groups	Infrastructure	(no group)
User Type	Not a super user	Super user

- On your private automation hub, use the `infra1` user to create a namespace called `community` and select the `Infrastructure` group as the namespace owner.
- On your private automation hub, as the `infra1` user, upload the Ansible Content Collection archive, at `/home/student/content-collections/community/community-mysql-3.1.1.tar.gz` on the `workstation` machine to that namespace, and then approve the `community.mysql` Ansible Content Collection.

Evaluation

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade compreview-users
```

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish compreview-users
```

► Solution

Adding Users and Teams

Create new users and a new team in automation controller and create users and groups in a private automation hub.

Outcomes

- Create organizations, users, and teams in automation controller.
- Create users and groups in a private automation hub.

Before You Begin

As the **student** user on the **workstation** machine, use the **lab** command to prepare your system for this exercise.

This command ensures that automation controller and private automation hub are installed. The command also removes resources created in previous chapters and downloads content collections to the `/home/student/content-collections` directory.

```
[student@workstation ~]$ lab start compreview-users
```

1. Create an organization called **BLA** in automation controller.
 - 1.1. Go to <https://controller.lab.example.com> and log in as the **admin** user with **redhat** as the password.
 - 1.2. Go to **Access > Organizations** and then click **Add**.
 - 1.3. On the **Create New Organization** page, complete the following details:

Field	Value
Name	BLA
Description	Business Line Applications

- 1.4. Click **Save** to create the organization.
2. Create a team called **SRE** in automation controller.
 - 2.1. Go to **Access > Teams** and then click **Add**.
 - 2.2. On the **Create New Team** page, complete the following details:

Field	Value
Name	SRE
Description	Site Reliability Engineering
Organization	BLA

- 2.3. Click **Save** to create the team.
- 3.** Create three users in automation controller.
- 3.1. Go to Access > Users and then click Add.
 - 3.2. On the Create New User page, complete the following details:

Field	Value
Username	sre1
Password	redhat123
Confirm Password	redhat123
User Type	Normal User
Organization	BLA

- 3.3. Click **Save** to create the user.
- 3.4. Repeat the steps to create the sre2 user and complete the following details:

Field	Value
Username	sre2
Password	redhat123
Confirm Password	redhat123
User Type	Normal User
Organization	BLA

- 3.5. Repeat the steps to create the sysadmin user and complete the following details:

Field	Value
Username	sysadmin
Password	redhat123
Confirm Password	redhat123
User Type	System Administrator
Organization	BLA

4. Assign the **sre1** user the **Admin** role on the SRE team.
 - 4.1. Go to **Access > Teams** and then click the link for the SRE team.
 - 4.2. Click the **Access** tab and then click **Add**.
 - 4.3. Click **Users** and then click **Next**.
 - 4.4. Select **sre1** and then click **Next**.
 - 4.5. Select **Admin** and then click **Save** to assign the role.
5. Assign the **sre2** user the **Member** role on the SRE team.
 - 5.1. Go to **Access > Teams** and then click the link for the SRE team.
 - 5.2. Click the **Access** tab and then click **Add**.
 - 5.3. Click **Users** and then click **Next**.
 - 5.4. Select **sre2** and then click **Next**.
 - 5.5. Select **Member** and then click **Save** to assign the role.
6. Verify the permissions for the SRE team.
 - 6.1. Go to **Access > Teams** and then click the link for the SRE team.
 - 6.2. Click the **Access** tab.

Notice that the **sre1** user has the **Admin** role and the **sre2** user has the **Member** role on the SRE team. The **sysadmin** user automatically inherited the **System Administrator** role on all resources.
7. Add a new group in your private automation hub called **Infrastructure** and assign permissions to manage Ansible Content Collections and containers.
 - 7.1. Go to <https://hub.lab.example.com> and log in as the **admin** user with **redhat** as the password.
 - 7.2. Go to **User Access > Groups** and then click **Create**.
 - 7.3. Enter **Infrastructure** in the **Name** field and then click **Create**.

Chapter 12 | Comprehensive Review

7.4. Click **Edit**. In the **Collection Namespaces** object list, select the following permissions:

- Add namespace
- Change namespace
- Delete namespace
- Upload to namespace

7.5. In the **Collections** object list, select the following permissions:

- Delete collection
- Modify Ansible repo content

7.6. In the **Collection Remotes** object list, select the following permissions:

- Change collection remote
- View collection remote

7.7. In the **Containers** object list, select the following permissions:

- Change container namespace permissions
- Change containers
- Change image tags
- Create new containers
- Delete container repository
- Push to existing containers

7.8. In the **Remote Registries** object list, select the following permissions:

- Add remote registry
- Change remote registry
- Delete remote registry

7.9. Click **Save** to create the group.

8. Add two private automation hub users.

8.1. Go to **User Access > Users** and then click **Create**.

8.2. On the **Create new user** page, complete the details for **infra1** as follows and click **Save** to create the user.

Field	Value
Username	infra1
Password	redhat123
Password confirmation	redhat123
Groups	Infrastructure
User Type	Not a super user

8.3. Repeat the step to create the super user with the following details:

Field	Value
Username	super
Password	redhat123
Password confirmation	redhat123
Groups	(no group)
User Type	Super user

Notice that the super user displays the Super user icon next to the name.

9. Verify the permissions for the Infrastructure group. Using the infra1 user, create a namespace and then upload and approve a content collection.
 - 9.1. Log out of the private automation hub web UI and log in as infra1 with redhat123 as the password.
 - 9.2. Go to Collections > Namespaces and then click Create.
 - 9.3. On the Create new namespace page, complete the following details and click Create to create the namespace.

Field	Value
Name	community
Namespace owners	Infrastructure



Important

Assigning a namespace owner allows that group to upload to the namespace. Adding a group as a namespace owner provides the Change namespace and Upload to namespace permissions to the group.

- 9.4. Click Upload collection.
- 9.5. Click Select file, select the archive at /home/student/content-collections/community/community-mysql-3.1.1.tar.gz, and then click Upload.
- 9.6. After the upload completes successfully, go to Collections > Approval.
- 9.7. Click Approve to approve the community.mysql content collection.
- 9.8. Go to Collections > Collections and verify that your private automation hub server displays the mysql automation content collection.

Evaluation

On the workstation machine, change to the student user home directory and use the lab command to grade your work. Correct any reported failures and rerun the command until successful.

Chapter 12 | Comprehensive Review

```
[student@workstation ~]$ lab grade compreview-users
```

Finish

On the workstation machine, change to the student user home directory and use the lab command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish compreview-users
```

▶ Lab

Uploading Automation Execution Environments to Private Automation Hub

Upload Automation execution environments to a private automation hub.

Outcomes

- Upload Automation execution environments to a private automation hub.
- Identify the version and release of container images.
- Add tags to the private automation hub container repositories.

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that automation controller and private automation hub are installed. The command also removes three container repositories from private automation hub and downloads container image archive files to the `/home/student/certified-EEs` directory.

```
[student@workstation ~]$ lab start compreview-hub
```

Specifications

Based on the following specification, upload the three automation execution environment image archives located in the `/home/student/certified-EEs` directory on `workstation` to your private automation hub, `hub.lab.example.com`. The password for the `admin` user on that private automation hub is `redhat`.

- Upload the following image archives as container images in the `ansible-automation-platform-22` namespace and use the `latest` tag:

Image archive	Container image
<code>ee-29-rhel8.tgz</code>	<code>hub.lab.example.com/ansible-automation-platform-22/ee-29-rhel8:latest</code>
<code>ee-minimal-rhel8.tgz</code>	<code>hub.lab.example.com/ansible-automation-platform-22/ee-minimal-rhel8:latest</code>
<code>ee-supported-rhel8.tgz</code>	<code>hub.lab.example.com/ansible-automation-platform-22/ee-supported-rhel8:latest</code>

- Determine the version and release of each container image file by using the `skopeo inspect` command on `workstation`, such as `1.0.0-119` where `1.0.0` is the version, and `119` is the release. Based on this information, add a tag using the matching string for each container file.

Chapter 12 | Comprehensive Review

In other words, the version and release reported by `skopeo inspect` for a particular image should match a tag on the container image in your private automation hub.

- Verify that all three automation execution environments are available and that you can use the `podman pull` command to download the container images to `workstation`. Make sure that you can download each container image when using the version-and-release tag that you added for that image.

Evaluation

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade compreview-hub
```

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish compreview-hub
```

► Solution

Uploading Automation Execution Environments to Private Automation Hub

Upload Automation execution environments to a private automation hub.

Outcomes

- Upload Automation execution environments to a private automation hub.
- Identify the version and release of container images.
- Add tags to the private automation hub container repositories.

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that automation controller and private automation hub are installed. The command also removes three container repositories from private automation hub and downloads container image archive files to the `/home/student/certified-EEs` directory.

```
[student@workstation ~]$ lab start compreview-hub
```

1. Upload the container image archives in the `/home/student/certified-EEs` directory to your private automation hub.
 - 1.1. From a terminal window, change to the `/home/student/certified-EEs` directory.

```
[student@workstation ~]$ cd ~/certified-EEs/
```

- 1.2. List the container image archives in the directory.

```
[student@workstation certified-EEs]$ ls  
ee-29-rhel8.tgz ee-minimal-rhel8.tgz ee-supported-rhel8.tgz
```

- 1.3. Log in to private automation hub at `hub.lab.example.com`, using `admin` as the username and `redhat` as the password.

```
[student@workstation certified-EEs]$ skopeo login hub.lab.example.com  
Username: admin  
Password: redhat  
Login Succeeded!
```

Chapter 12 | Comprehensive Review

- 1.4. Use the `skopeo copy` command to upload the image archives. Upload the image archives to the `hub.lab.example.com` server using the `ansible-automation-platform-22` namespace and the `latest` tag.

```
[student@workstation certified-EEs]$ skopeo copy \
> docker-archive:ee-29-rhel8.tgz \
> docker://hub.lab.example.com/ansible-automation-platform-22/ee-29-rhel8:latest
Getting image source signatures
...output omitted...
Writing manifest to image destination
Storing signatures

[student@workstation certified-EEs]$ skopeo copy \
> docker-archive:ee-minimal-rhel8.tgz docker://hub.lab.example.com/\
> ansible-automation-platform-22/ee-minimal-rhel8:latest
Getting image source signatures
...output omitted...
Writing manifest to image destination
Storing signatures

[student@workstation certified-EEs]$ skopeo copy \
> docker-archive:ee-supported-rhel8.tgz docker://hub.lab.example.com/\
> ansible-automation-platform-22/ee-supported-rhel8:latest
Getting image source signatures
...output omitted...
Writing manifest to image destination
Storing signatures
```

**Important**

The previous commands use the line continuation character, `\`, to split the commands across multiple lines. Do not insert a space before the line continuation character used in the `docker://hub.lab.example.com/\` portion of the commands.

2. Using the web UI, verify that private automation hub provides at least the following three container repositories:
- `ansible-automation-platform-22/ee-29-rhel8`
 - `ansible-automation-platform-22/ee-minimal-rhel8`
 - `ansible-automation-platform-22/ee-supported-rhel8`
- 2.1. Go to `https://hub.lab.example.com` and log in as the `admin` user with `redhat` as the password.
- 2.2. Go to **Execution Environments > Execution Environments** to display the available Automation execution environments.
- You created the three container repositories that use `ansible-automation-platform-22` in their names.
 - The installer created the `ee-29-rhel8`, `ee-minimal-rhel8`, and `ee-supported-rhel8` container repositories using the container image archives included in the Ansible Automation Platform setup bundle.

Container repository name	Description	Created	Last modified	Status
ansible-automation-platform-22/ee-29-rhel8		4 minutes ago	3 minutes ago	Local
ansible-automation-platform-22/ee-minimal-rhel8		3 minutes ago	3 minutes ago	Local
ansible-automation-platform-22/ee-supported-rhel8		3 minutes ago	3 minutes ago	Local

Figure 12.19: Execution environments

3. Identify the version and release of each container image archive in the /home/student/certified-EEs directory and then use that information to add new tags to the previously created private automation hub container repositories.
 - 3.1. From the terminal window, use the `skopeo inspect` command to identify the version and release of the container image archives.

```
[student@workstation certified-EEs]$ skopeo inspect \
> docker-archive:ee-29-rhel8.tgz \
> --format "{{ .Labels.version }}-{{ .Labels.release }}"
1.0.0-119

[student@workstation certified-EEs]$ skopeo inspect \
> docker-archive:ee-minimal-rhel8.tgz \
> --format "{{ .Labels.version }}-{{ .Labels.release }}"
1.0.0-128

[student@workstation certified-EEs]$ skopeo inspect \
> docker-archive:ee-supported-rhel8.tgz \
> --format "{{ .Labels.version }}-{{ .Labels.release }}"
1.0.0-99
```

- 3.2. From the private automation hub web UI, go to **Execution Environments > Execution Environments** and click the link for the `ansible-automation-platform-22/ee-29-rhel8` container repository.
- 3.3. Click the **Images** tab. Click the vertical ellipsis icon \vdots and then click **Manage tags**.
- 3.4. Enter `1.0.0-119` in the **Add new tag** field and click **Add**.
- 3.5. Click **Save** and then close the window. After a few seconds, you should see the `1.0.0-119` tag in the tag list.
- 3.6. Use the same steps to add the `1.0.0-128` tag to the `ansible-automation-platform-22/ee-minimal-rhel8` container repository.

Chapter 12 | Comprehensive Review

- 3.7. Use the same steps to add the 1.0.0-99 tag to the ansible-automation-platform-22/ee-supported-rhel8 container repository.
4. Use the podman pull command to verify that you can download the following three container images from your private automation hub:
- ansible-automation-platform-22/ee-29-rhel8:1.0.0-119
 - ansible-automation-platform-22/ee-minimal-rhel8:1.0.0-128
 - ansible-automation-platform-22/ee-supported-rhel8:1.0.0-99
- 4.1. Download the ansible-automation-platform-22/ee-29-rhel8:1.0.0-119 container image:

```
[student@workstation certified-EEs]$ podman pull \
> hub.lab.example.com/ansible-automation-platform-22/ee-29-rhel8:1.0.0-119
...output omitted...
```

- 4.2. Download the ansible-automation-platform-22/ee-minimal-rhel8:1.0.0-128 container image:

```
[student@workstation certified-EEs]$ podman pull \
> hub.lab.example.com/ansible-automation-platform-22/ee-minimal-rhel8:1.0.0-128
...output omitted...
```

- 4.3. Download the ansible-automation-platform-22/ee-supported-rhel8:1.0.0-99 container image:

```
[student@workstation certified-EEs]$ podman pull \
> hub.lab.example.com/ansible-automation-platform-22/ee-supported-rhel8:1.0.0-99
...output omitted...
```

- 4.4. List the downloaded container images:

```
[student@workstation certified-EEs]$ podman images
REPOSITORY                                     TAG
hub.lab.example.com/ansible-automation-platform-22/ee-supported-rhel8  1.0.0-99
hub.lab.example.com/ansible-automation-platform-22/ee-minimal-rhel8    1.0.0-128
hub.lab.example.com/ansible-automation-platform-22/ee-29-rhel8          1.0.0-119
```

Evaluation

On the workstation machine, change to the student user home directory and use the lab command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade compreview-hub
```

Finish

On the workstation machine, change to the student user home directory and use the lab command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish compreview-hub
```

▶ Lab

Creating an Inventory Managed as a Project

Create an inventory in automation controller that is stored and managed in a Git repository and then create a smart inventory that filters your existing inventories.

Outcomes

- Create an inventory.
- Use an existing static inventory file stored in a Git repository as the source for the new inventory.
- Create a smart inventory based on a custom fact.

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that automation controller is installed and creates a `sysctl.fact` custom fact on your managed hosts.

The command also updates automation controller, removing resources created in previous chapters and creating the following resources:

- A custom fact on your managed hosts named `sysctl.fact`.
- A project named `Compreview Inventory` that synchronizes content from a Git repository containing a static inventory and group variable files.
- A job template named `Deploy compreview MOTD` that updates the message of the day on your managed hosts based on group variables that it gets from the inventory.
- A job template named `Refresh Fact Cache Compreview` that updates the fact cache for your managed hosts on automation controller.

```
[student@workstation ~]$ lab start compreview-inventory
```

Specifications

In this lab, you create an inventory in automation controller that gets its information from an existing static inventory file that is stored in a Git repository. You also create a smart inventory based on your existing inventories, selecting managed hosts from those inventories based on the value of a custom fact on your managed hosts.

Your automation controller is at `https://controller.lab.example.com`. You can log in to it using `admin` as the user and `redhat` as the password.

Configure your automation controller based on the following specification.

Chapter 12 | Comprehensive Review

- Create an inventory on your automation controller using the following information.

Field	Value
Name	Git Site1
Organization	Default

Add a source for this inventory using the information in the following table. Make sure that you can successfully synchronize the source.

Field	Value
Name	Git Inventory Lab
Source	Sourced from a Project
Project	Comprevew Inventory
Inventory file	inventory
Update options	Update on launch (selected)

**Important**

You might see the following message below the **Project** field, even if you selected the project name from the list of projects.

That value was not found. Please enter or select a valid value.

The message disappears as soon as you select a different field.

After you configure the **Git Site1** inventory, inspect it to confirm that groups and hosts have been imported and that variables for the **dev_webservers** and **dev_dbservers** groups are set.

- Modify the **Deploy comprevew MOTD** job template so that it uses the **Git Site1** inventory. Limit the job template so that it only runs for the **dev_webservers** and **dev_dbservers** groups.

When you run the job template, confirm that the contents of the **/etc/motd** file on managed hosts in the inventory reflect the group variables that apply to those hosts.

- After you complete the preceding item, clone the <https://git.lab.example.com/git/comprevew-inventory.git> Git repository into the **/home/student/git-repos** directory on the **workstation** machine. Add **serverc.lab.example.com** to the **dev_webservers** group and then commit and push the changes to the Git repository.

Test the change to the Git repository by synchronizing the **Comprevew Inventory** project and launching the **Deploy comprevew MOTD** job template. Confirm that the contents of the **/etc/motd** file on **serverc.lab.example.com** reflects the values of the group variables for the **dev_webservers** group in the inventory.

Chapter 12 | Comprehensive Review

- Make sure that your automation controller caches facts for one day (86400 seconds). Update the Refresh Fact Cache Compreview job template so that it uses the Git Site1 inventory and it stores facts in your automation controller's cache. Launch the Refresh Fact Cache Compreview job template to populate the cache.
- Create a smart inventory named Site1 Servers for Patching in the Default organization. Configure the smart inventory to consist of managed hosts that have the custom fact `ansible_local['sysctl']['netdev_max_backlog']` set to "1000".

Inspect the completed smart inventory to confirm that it has managed hosts.

Evaluation

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful. You *must* add, commit, and push any additional changes to the remote Git repository before grading.

```
[student@workstation ~]$ lab grade compreview-inventory
```

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish compreview-inventory
```

► Solution

Creating an Inventory Managed as a Project

Create an inventory in automation controller that is stored and managed in a Git repository and then create a smart inventory that filters your existing inventories.

Outcomes

- Create an inventory.
- Use an existing static inventory file stored in a Git repository as the source for the new inventory.
- Create a smart inventory based on a custom fact.

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that automation controller is installed and creates a `sysctl.fact` custom fact on your managed hosts.

The command also updates automation controller, removing resources created in previous chapters and creating the following resources:

- A custom fact on your managed hosts named `sysctl.fact`.
- A project named `Compreview Inventory` that synchronizes content from a Git repository containing a static inventory and group variable files.
- A job template named `Deploy compreview MOTD` that updates the message of the day on your managed hosts based on group variables that it gets from the inventory.
- A job template named `Refresh Fact Cache Compreview` that updates the fact cache for your managed hosts on automation controller.

```
[student@workstation ~]$ lab start compreview-inventory
```

1. Go to `https://controller.lab.example.com` and log in as the `admin` user with `redhat` as the password.
2. Create an inventory in your automation controller called `Git Site1` in the `Default` organization.

Field	Value
Name	Git Site1
Organization	Default

Chapter 12 | Comprehensive Review

Then, add a source for this inventory using the information in the following table. After you save the **Git Inventory Lab** source, start the synchronization process. Verify that the synchronization process succeeds.

Field	Value
Name	Git Inventory Lab
Source	Sourced from a Project
Project	Compreview Inventory
Inventory file	inventory
Update options	Update on launch (<i>selected</i>)

- 2.1. Go to Resources > Inventories and then click Add > Add inventory.
- 2.2. Enter **Git Site1** in the **Name** field and select the **Default** organization. When done, click **Save**.
- 2.3. Click the **Sources** tab, and then click **Add** to add a source for the inventory.
- 2.4. Enter **Git Inventory Lab** in the **Name** field.
- 2.5. In the **Source** list, choose **Sourced from a Project**.
- 2.6. Select the **Compreview Inventory** project. You might see the following message below the **Project** field:

That value was not found. Please enter or select a valid value.

The message disappears as soon as you select a different field.

- 2.7. Select **inventory** as the **Inventory file**.
- 2.8. Select the **Update on launch** option and then click **Save**.
- 2.9. Click **Sync** and wait until the value of the **Last Job Status** field displays the **Successful** message.
3. Review the **Git Site1** inventory to identify the hosts and the groups.
 - 3.1. Go to Resources > Inventories and click **Git Site1**.
 - 3.2. Review the **Hosts** and the **Groups** tab and notice the hosts and groups that were added to the inventory.
 - 3.3. On the **Groups** tab, click the link for **dev_webservers** group and notice the variables are also imported.

The screenshot shows the Red Hat Ansible Automation Platform web interface. On the left, there's a sidebar with 'Views' (Dashboard, Jobs, Schedules, Activity Stream, Workflow Approvals) and 'Resources' (Templates, Credentials, Projects, Inventories, Hosts). The 'Inventories' option under Resources is highlighted. The main content area shows the 'Group details' for the 'dev_webservers' group. The navigation bar at the top includes 'Inventories > Git Site1 Inventory > Groups > dev_webservers'. Below the navigation is a 'Group details' section with tabs for 'Back to Groups', 'Details' (which is selected), 'Related Groups', and 'Hosts'. The 'Name' is listed as 'dev_webservers' and 'Description' as 'imported'. Under 'Variables', there are two tabs: 'YAML' (selected) and 'JSON'. The JSON code shown is:

```
1 {  
2   "motd_environment": "DEV",  
3   "motd_server_type": "webservers"  
4 }
```

Below the variables, the 'Created' date is 8/11/2022, 8:21:36 AM and the 'Last Modified' date is 8/11/2022, 8:21:36 AM. There are 'Edit' and 'Delete' buttons at the bottom.

Figure 12.16: Group variables



Note

Although the variables here are displayed in JSON format, the web UI can also display variables in YAML format.

4. Modify the Deploy_compreview MOTD job template to use the Git_Site1 inventory and limit it to run only for the dev_webservers and dev_dbservers groups.
 - 4.1. Go to Resources > Templates.
 - 4.2. Click the Edit Template icon for the Deploy compreview MOTD template.
 - 4.3. Click the search icon for Inventory and select Git_Site1.
 - 4.4. In the Limit field, enter dev_webservers, dev_dbservers, and then click Save.
 - 4.5. Click Launch to launch the job template.
 - 4.6. Observe the live output of the running job.

Chapter 12 | Comprehensive Review

The screenshot shows the Ansible Tower 'Output' tab for a job named 'Deploy comprevue MOTD'. The status is 'Successful'. The output pane displays the command-line logs from the job execution. The logs show the deployment of a message of the day (MOTD) file, with tasks for gathering facts and populating the MOTD. The log entries include timestamps (12:16:02, 12:16:04, 12:16:05), host names (servera.lab.example.com, serverb.lab.example.com), and various metrics like ok, changed, unreachable, failed, skipped, rescued, and ignored counts.

Figure 12.17: Successful output for the "Deploy comprevue MOTD" job template

- 4.7. Click the **Details** tab and verify that the **Status** field displays the **Successful** message.
5. Verify the servers from the `dev_webservers` and `dev_dbservers` groups have the correct variables in the message of the day.
 - 5.1. Open a terminal on the `workstation` machine and use SSH to log in to `servera.lab.example.com`.

```
[student@workstation ~]$ ssh servera.lab.example.com
Warning: Permanently added 'servera.lab.example.com,172.25.250.10' (ECDSA) to the
list of known hosts.
=====
Ansible manages this server using the webservers group and the DEV environment.
=====
...output omitted...
[student@servera ~]$
```

- 5.2. Notice the message of the day when accessing the server, and then log out of `servera.lab.example.com`.

```
[student@servera ~]$ exit
logout
Connection to servera.lab.example.com closed.
[student@workstation ~]$
```

6. Clone the `https://git.lab.example.com/git/comprevue-inventory.git` Git repository into the `/home/student/git-repos` directory on the `workstation` machine. Add the `serverc.lab.example.com` to the `dev_webservers` group and then commit and push the changes to the Git repository.
 - 6.1. From a terminal, create the `/home/student/git-repos` directory if it does not already exist, and then change into it.

Chapter 12 | Comprehensive Review

```
[student@workstation ~]$ mkdir -p ~/git-repos/  
[student@workstation ~]$ cd ~/git-repos/
```

- 6.2. Clone the <https://git.lab.example.com/git/comprevew-inventory.git> Git repository into the /home/student/git-repos directory.

```
[student@workstation git-repos]$ git clone \  
> https://git.lab.example.com/git/comprevew-inventory.git  
Cloning into 'comprevew-inventory'...  
...output omitted...  
[student@workstation git-repos]$ cd comprevew-inventory
```

- 6.3. Review the inventory file.

```
[dev_webservers]  
servera.lab.example.com  
  
[dev_dbservers]  
serverb.lab.example.com  
  
[dev:children]  
dev_webservers  
dev_dbservers  
  
[prod_webservers]  
serverd.lab.example.com  
  
[prod_dbservers]  
servere.lab.example.com  
  
[prod:children]  
prod_webservers  
prod_dbservers
```

- 6.4. Add serverc.lab.example.com in the [dev_webservers] section in the inventory file.

```
[dev_webservers]  
servera.lab.example.com  
serverc.lab.example.com  
...output omitted...
```

- 6.5. Commit and push the changes.

Chapter 12 | Comprehensive Review

```
[student@workstation compreview-inventory]$ git add inventory
[student@workstation compreview-inventory]$ git commit -m "Added serverc"
[main ec9bd43] Added serverc
 1 file changed, 1 insertion(+)
[student@workstation compreview-inventory]$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
...output omitted...
```

7. Synchronize the Comprevew Inventory project.
 - 7.1. From the automation controller web UI, go to **Resources > Projects**.
 - 7.2. Click the **Sync Project** icon for the **Comprevew Inventory** project and wait until the synchronization is complete.
8. Launch the `Deploy compreview MOTD` job template again and verify that the job updates the `/etc/motd` file on the `serverc.lab.example.com` host.
 - 8.1. Go to **Resources > Templates**.
 - 8.2. Click the **Launch Template** icon for the `Deploy compreview MOTD` template.
 - 8.3. Observe the live output of the running job.
 - 8.4. After the job is successful, notice that the tasks also executed on `serverc.lab.example.com`.

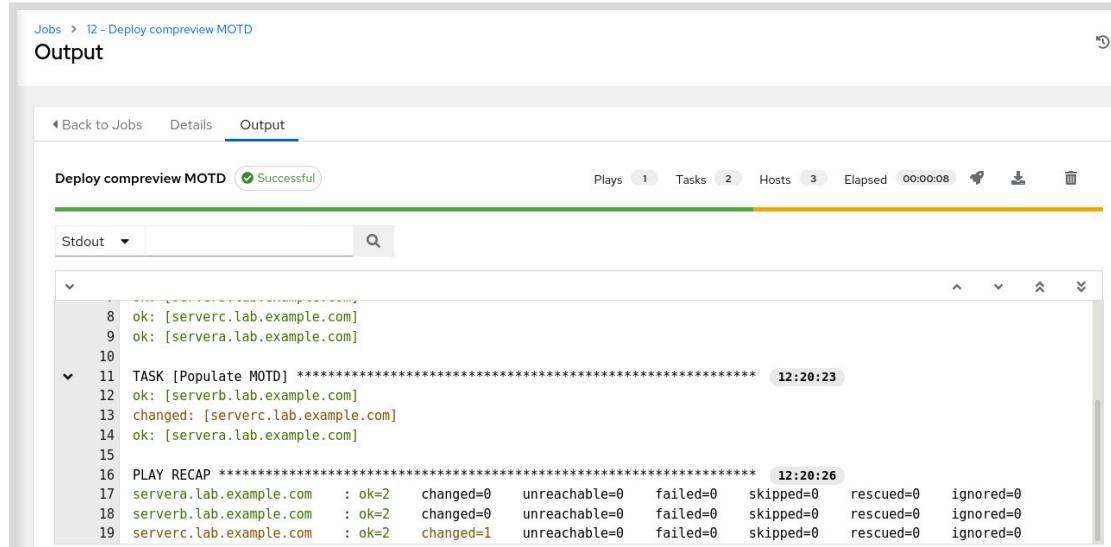


Figure 12.18: Successful output for the "Deploy compreview MOTD" job template

- 8.5. Open a terminal on the `workstation` machine and use SSH to log in to `serverc.lab.example.com`.

```
[student@workstation compreview-inventory]$ ssh serverc.lab.example.com
Warning: Permanently added 'serverc.lab.example.com,172.25.250.12' (ECDSA) to the
list of known hosts.
=====
Ansible manages this server using the webservers group and the DEV environment.
=====
...output omitted...
[student@serverc ~]$
```

- 8.6. Notice the message of the day when accessing the server, and then log out of `serverc.lab.example.com`

```
[student@serverc ~]$ exit
logout
Connection to serverc.lab.example.com closed.
[student@workstation compreview-inventory]$
```

9. Ensure that the automation controller job settings are configured so that the **Per-Host Ansible Fact Cache Timeout** setting has a value of 86400 seconds.
 - 9.1. Go to **Settings** in the left navigation bar.
 - 9.2. In the **Settings** window, under the **Jobs** section, click **Jobs settings**.
 - 9.3. Confirm that **Per-Host Ansible Fact Cache Timeout** is set to 86400 seconds. If the setting is different, click **Edit**, edit that field to contain the value 86400, and click **Save**.
10. Modify the **Refresh Fact Cache Compreview** job template to use the **Git Site1** inventory and to enable it to use fact storage. Launch a job to populate the automation controller's cache.
 - 10.1. Navigate **Resources > Templates**. For the **Refresh Fact Cache Compreview** job template, click the **Edit Template** icon to edit that job template.
 - 10.2. In the **Inventory** field, select the **Git Site1**.
 - 10.3. Select the **Enable Fact Storage** checkbox and then click **Save**.
 - 10.4. On the **Details** page, click **Launch** and wait until the **Status** field displays the **Successful** message.
11. Verify that the `sysctl` custom fact for `servera.lab.example.com` is available in the automation controller cache (the custom fact should be also available for all the servers).
 - 11.1. Go to **Resources > Inventories**.
 - 11.2. Click the **Git Site1** inventory, and then click the **Hosts** tab.
 - 11.3. Click `servera.lab.example.com`, and then click the **Facts** tab.
 - 11.4. Verify that the `ansible_local['sysctl']['netdev_max_backlog']` custom fact for `servera.lab.example.com` is available in the cache.

Facts [YAML](#) [JSON](#)

```

211 },
212 "ansible_fips": false,
213 "ansible_fqdn": "servera.lab.example.com",
214 "module_setup": true,
215 "ansible_local": {
216   "sysctl": {
217     "netdev_max_backlog": "1000"
218   }

```

Figure 12.19: Custom fact in the automation controller cache

12. Create a smart inventory named Site1 Servers for Patching in the Default organization. Configure the smart inventory to use the ansible_local__sysctl__netdev_max_backlog="1000" filter against cached Ansible facts.

12.1. Go to Resources > Inventories and then click Add > Add smart inventory.

12.2. Enter the initial details for the smart inventory, but do not save the smart inventory yet.

Field	Value
Name	Site1 Servers for Patching
Organization	Default

12.3. Click the search icon for the Smart host filter field. Create an advanced filter using the following information and then click Select.

Field	Value
Filter type	Advanced
Key	ansible_facts
Search field	ansible_local__sysctl__netdev_max_backlog="1000"



Important

Make sure to use double underscores as delimiters before and after the sysctl key.

12.4. Press Enter after you add the search field, and then click Select.

12.5. Click Save to create the smart inventory.

12.6. Click the Hosts tab and verify that all the hosts are available, including servera.lab.example.com, serverb.lab.example.com, serverc.lab.example.com, serverd.lab.example.com, and servere.lab.example.com.

**Note**

If you do not see the specified hosts, then look for a typing mistake in the smart inventory host filter.

Evaluation

On the workstation machine, change to the student user home directory and use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful. You *must* add, commit, and push any additional changes to the remote Git repository before grading.

```
[student@workstation ~]$ lab grade compreview-inventory
```

Finish

On the workstation machine, change to the student user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish compreview-inventory
```

▶ Lab

Configuring Job Templates

Configure several job templates, as well as a supporting project and source control credentials.

Outcomes

- Create a machine credential and a source control credential.
- Create a project.
- Create an inventory and use a static inventory file stored in a Git repository as its source.
- Create some job templates.
- Launch the job templates from the automation controller web UI.

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start compreview-templates
```

Specifications

In this lab, you create a job template for a log server configuration and a job template for the configuration of the clients. You also create the necessary credentials and use an existing inventory stored in a Git repository as an inventory source.

Your automation controller is at <https://controller.lab.example.com>. You can log in to it using `admin` as the user and `redhat` as the password.

- Create a machine credential in your automation controller using the information in the following table.

Field	Value
Name	Virtual Machines
Description	Credential for managing virtual machines
Organization	Default
Credential Type	Machine
Username	devops

Field	Value
Password	redhat
Privilege Escalation Method	sudo
Privilege Escalation Username	root

**Note**

Because the devops user does not need to enter a password to run sudo commands, you do not need to enter a password in the **Privilege Escalation Password** field.

- Create a source control credential in your automation controller using the information in the following table.

Field	Value
Name	Git Credential
Organization	Default
Credential Type	Source Control
Username	git
SCM Private Key	Contents of the /home/student/.ssh/gitlab_rsa file

**Note**

If you choose to browse for the /home/student/.ssh/gitlab_rsa file, then right-click anywhere in the directory navigation and select **Show Hidden Files**. With this option enabled, you can see the .ssh directory in the /home/student directory.

- Create a project in your automation controller using the information in the following table. Notice that you use the source control credential previously created. After you save the project, verify that automation controller reports the success of the project synchronization.

Field	Value
Name	Log Centralization
Organization	Default
Source Control Type	Git
Source Control URL	git@git.lab.example.com:git/comprevue-templates.git
Source Control Credential	Git Credential

Important

If the project synchronization fails, then ensure that the project specifies the correct source control URL and the correct source control credential. Ensure that the source control credential uses the correct values and then attempt the project synchronization again.

- Create an inventory in your automation controller called `Lab Git Sourced` in the `Default` organization.

Field	Value
Name	<code>Lab Git Sourced</code>
Organization	<code>Default</code>

Then, add a source for this inventory using the information in the following table. After you save the `Git Inventory Lab` source, start the synchronization process. Verify that the synchronization process succeeds.

Field	Value
Name	<code>Git Inventory Lab</code>
Source	<code>Sourced from a Project</code>
Project	<code>Log Centralization</code>
Inventory file	<code>inventory</code>
Update options	<code>Update on launch (selected)</code>

Important

You might see the following message below the `Project` field, even if you selected the project name from the list of projects.

That value was not found. Please enter or select a valid value.

The message disappears as soon as you select a different field.

- Review the `Groups` tab and the `Hosts` tab for the `Lab Git Sourced` inventory. The hosts are grouped as shown in the following table. Notice that the variables for the `dev_clients` and `prod_clients` groups are also imported.

Parent group	Children group	Host
<code>log_servers</code>	<code>dev_servers</code>	<code>servera.lab.example.com</code>
	<code>prod_servers</code>	<code>serverd.lab.example.com</code>
<code>log_clients</code>	<code>dev_clients</code>	<code>serverb.lab.example.com</code>

Parent group	Children group	Host
	prod_clients	serverc.lab.example.com
		servere.lab.example.com
		serverf.lab.example.com

- Create a job template in your automation controller using the information in the following table. This job template configures a centralized log server on each of the hosts in the `log_servers` group for the `Lab Git Sourced` inventory.

Field	Value
Name	Log Servers Deploy
Job Type	Run
Inventory	Lab Git Sourced
Project	Log Centralization
Playbook	rsyslog_server_deploy.yml
Credentials	Virtual Machines

- Create a job template in your automation controller using the information in the following table. This job template configures each of the hosts in the `log_clients` group for the `Lab Git Sourced` inventory as a client for its respective log server.

Field	Value
Name	Log Clients Deploy
Job Type	Run
Inventory	Lab Git Sourced
Project	Log Centralization
Playbook	rsyslog_client_deploy.yml
Credentials	Virtual Machines

- Launch the `Log Servers Deploy` job template. Observe the live output of the running job. Verify that the job succeeds.
- Launch the `Log Clients Deploy` job template. Observe the live output of the running job. Verify that the job succeeds.
- Verify that the hosts from the `dev_clients` group in the `Lab Git Sourced` inventory send logs to `servera.lab.example.com`. Use the logger "This is a test" command to create a log entry in the `dev_clients` hosts.
- Verify that the hosts from the `prod_clients` group in the `Lab Git Sourced` inventory send logs to `serverd.lab.example.com`. Use the logger "This is a test" command to create a log entry in the `prod_clients` hosts.

Evaluation

On the **workstation** machine, change to the **student** user home directory and use the **lab** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade compreview-templates
```

Finish

On the **workstation** machine, change to the **student** user home directory and use the **lab** command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish compreview-templates
```

► Solution

Configuring Job Templates

Configure several job templates, as well as a supporting project and source control credentials.

Outcomes

- Create a machine credential and a source control credential.
- Create a project.
- Create an inventory and use a static inventory file stored in a Git repository as its source.
- Create some job templates.
- Launch the job templates from the automation controller web UI.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start compreview-templates
```

1. Go to <https://controller.lab.example.com> and log in as the `admin` user with `redhat` as the password.
2. Create a machine credential using the information in the following table.

Field	Value
Name	Virtual Machines
Description	Credential for managing virtual machines
Organization	Default
Credential Type	Machine
Username	devops
Password	redhat
Privilege Escalation Method	sudo
Privilege Escalation Username	root

- 2.1. Go to Resources > Credentials and click Add to add a new credential.

Chapter 12 | Comprehensive Review

- 2.2. Enter Virtual Machines in the **Name** field and Credential for managing virtual machines in the **Description** field.
 - 2.3. Select the **Default** organization.
 - 2.4. In the **Credential Type** list, choose **Machine**.
 - 2.5. Enter devops in the **Username** field and redhat in the **Password** field.
 - 2.6. In the **Privilege Escalation Method** list, choose **sudo**.
 - 2.7. Enter root in the **Privilege Escalation Username** field.
 - 2.8. Click **Save** to save the machine credential.
- 3.** Create a source control credential using the information in the following table.

Field	Value
Name	Git Credential
Organization	Default
Credential Type	Source Control
Username	git
SCM Private Key	Content of the /home/student/.ssh/gitlab_rsa file.

- 3.1. Go to Resources > Credentials and click **Add** to add a new credential.
- 3.2. Enter **Git Credential** in the **Name** field.
- 3.3. Select the **Default** organization.
- 3.4. In the **Credential Type** list, choose **Source Control**.
- 3.5. Enter **git** in the **Username** field and click **Browse** in the **SCM Private Key** field.
- 3.6. By default, the **File Upload** directory navigation window opens in the **Desktop** directory. Click **student** to move to the home directory for the **student** user.

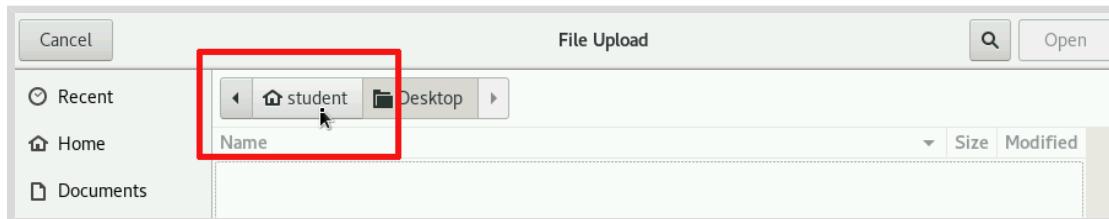


Figure 12.15: Browsing for the SCM Private Key

- 3.7. Right-click anywhere in the directory navigation window and select **Show Hidden Files**.
- 3.8. Change to the **.ssh** hidden directory, then select the **gitlab_rsa** file, and click **Open**.
- 3.9. Click **Save** to save the source control credential.

Chapter 12 | Comprehensive Review

4. Create a project using the information in the following table. After you save the project, verify that automation controller reports the success of the project synchronization.

Field	Value
Name	Log Centralization
Organization	Default
Source Control Type	Git
Source Control URL	git@git.lab.example.com:git/compreviwt-templates.git
Source Control Credential	Git Credential

- 4.1. Go to **Resources > Projects** and then click **Add** to create a project.
- 4.2. Enter **Log Centralization** in the **Name** field.
- 4.3. Select the **Default** organization if it is not selected by default.
- 4.4. In the **Source Control Type** list, choose **Git**.
- 4.5. Enter **git@git.lab.example.com:git/compreviwt-templates.git** in the **Source Control URL** field.
- 4.6. Select the **Git Credential** source control credential.
- 4.7. Click **Save** to save the project.
- 4.8. Automation controller automatically attempts to synchronize the new project. On the **Details** page for the **Log Centralization** project, verify that the **Last Job Status** field displays the **Successful** message.

**Important**

If the project synchronization fails, then ensure that the project specifies the correct source control URL and the correct source control credential.

5. Create an inventory called **Lab Git Sourced** in the **Default** organization.

Field	Value
Name	Lab Git Sourced
Organization	Default

Then, add a source for this inventory using the information in the following table. After you save the **Git Inventory Lab** source, start the synchronization process. Verify that the synchronization process succeeds.

Field	Value
Name	Git Inventory Lab
Source	Sourced from a Project
Project	Log Centralization
Inventory file	inventory
Update options	Update on launch (selected)

- 5.1. Go to Resources > Inventories.
- 5.2. Click Add > Add inventory.
- 5.3. Enter Lab Git Sourced in the Name field and select the Default organization. When done, click Save.
- 5.4. Click the Sources tab, and then click Add to add a source for the inventory.
- 5.5. Enter Git Inventory Lab in the Name field.
- 5.6. In the Source list, choose Sourced from a Project.
- 5.7. Select the Log Centralization project. You might see the following message below the Project field:

That value was not found. Please enter or select a valid value.

The message disappears as soon as you select a different field.

- 5.8. Select inventory as the Inventory file.
- 5.9. Select the Update on launch option and click Save when done.
- 5.10. Click Sync and wait until the value of the Last Job Status field displays the Successful message.
6. Review the Groups tab and the Hosts tab for each group in the Lab Git Sourced inventory. The hosts are grouped as shown in the following table. Notice that the variables for the hosts in the log_clients group are also imported.

Parent group	Child group	Host
log_servers	dev_servers	servera.lab.example.com
	prod_servers	serverd.lab.example.com
log_clients	dev_clients	serverb.lab.example.com
		serverc.lab.example.com
	prod_clients	servere.lab.example.com
		serverf.lab.example.com

Chapter 12 | Comprehensive Review

- 6.1. Go to **Resources > Inventories**.
- 6.2. Click the **Lab Git Sourced** link.
- 6.3. Review the **Groups** tab and the **Hosts** tab and notice the groups and hosts that were added to the inventory. You can use the preceding table for reference.
- 6.4. Click the **Details** tab for the **dev_clients** group and notice the variables are also imported.

The screenshot shows the 'Details' tab for the 'dev_clients' group. At the top, there are tabs for 'Back to Groups', 'Details' (which is selected), 'Related Groups', and 'Hosts'. Below the tabs, the group details are listed: Name (dev_clients), Description (imported), and Variables (YAML or JSON). The 'Variables' section contains the following YAML code:

```
1: {  
2:   "rsyslog_port": 514,  
3:   "rsyslog_server": "servera.lab.example.com"  
4: }
```

At the bottom of the screen, there are 'Edit' and 'Delete' buttons.

Figure 12.16: Imported variables for the hosts in the dev_clients group

- 6.5. Click the **Details** tab for the **prod_clients** group and notice the variables are also imported.

The screenshot shows the 'Details' tab for the 'prod_clients' group. The layout is identical to Figure 12.16, with tabs for 'Back to Groups', 'Details' (selected), 'Related Groups', and 'Hosts'. Group details: Name (prod_clients), Description (imported), and Variables (YAML or JSON). The 'Variables' section contains the following YAML code:

```
1: {  
2:   "rsyslog_port": 514,  
3:   "rsyslog_server": "serverd.lab.example.com"  
4: }
```

At the bottom, there are 'Edit' and 'Delete' buttons.

Figure 12.17: Imported variables for the hosts in the prod_clients group

7. Create a job template using the information in the following table. This job template configures a centralized log server on each of the hosts in the **log_servers** group for the **Lab Git Sourced** inventory.

Field	Value
Name	Log Servers Deploy
Job Type	Run
Inventory	Lab Git Sourced
Project	Log Centralization
Playbook	rsyslog_server_deploy.yml
Credentials	Virtual Machines

- 7.1. Go to Resources > Templates.
 - 7.2. Click Add > Add job template.
 - 7.3. Enter Log Servers Deploy in the Name field.
 - 7.4. Select Run as the Job Type if it is not selected by default.
 - 7.5. Select the Lab Git Sourced inventory and the Log Centralization project.
 - 7.6. In the Playbook list, choose rsyslog_server_deploy.yml.
 - 7.7. Select the Virtual Machines credential.
 - 7.8. Click Save to save the job template.
8. Create a job template using the information in the following table. This job template configures each of the hosts in the log_clients group for the Lab Git Sourced inventory as a client for its respective log server.

Field	Value
Name	Log Clients Deploy
Job Type	Run
Inventory	Lab Git Sourced
Project	Log Centralization
Playbook	rsyslog_client_deploy.yml
Credentials	Virtual Machines

- 8.1. Go to Resources > Templates.
- 8.2. Click Add > Add job template.
- 8.3. Enter Log Clients Deploy in the Name field.
- 8.4. Select Run as the Job Type if it is not selected by default.

Chapter 12 | Comprehensive Review

- 8.5. Select the **Lab Git Sourced** inventory and the **Log Centralization** project.
 - 8.6. In the **Playbook** list, choose **rsyslog_client_deploy.yml**.
 - 8.7. Select the **Virtual Machines** credential.
 - 8.8. Click **Save** to save the job template.
- 9.** Launch the **Log Servers Deploy** job template. Observe the live output of the running job. Verify that the job succeeds.
- 9.1. Go to **Resources > Templates**.
 - 9.2. Click the **Launch Template** icon for the **Log Servers Deploy** job template.
 - 9.3. Observe the live output of the running job.

```
11
12 RUNNING HANDLER [Reload rsyslog service] ****
13 changed: [servera.lab.example.com]
14 changed: [serverd.lab.example.com]
15
16 PLAY RECAP ****
17 servera.lab.example.com : ok=3    changed=2    unreachable=0    failed=0    skipped=0    rescued=0
18 ignored=0
18 serverd.lab.example.com : ok=3    changed=2    unreachable=0    failed=0    skipped=0    rescued=0
18 ignored=0
```

Figure 12.18: Successful output for the "Log Servers Deploy" job

- 9.4. Click the **Details** tab and verify that the **Status** field displays the **Successful** message.
- 10.** Launch the **Log Clients Deploy** job template. Observe the live output of the running job. Verify that the job succeeds.
- 10.1. Go to **Resources > Templates**.
 - 10.2. Click the **Launch Template** icon for the **Log Clients Deploy** job template.
 - 10.3. Observe the live output of the running job.

Chapter 12 | Comprehensive Review

```
Stdout
9
10 RUNNING HANDLER [Reload rsyslog service] ****
11 changed: [serverf.lab.example.com]
12 changed: [serverb.lab.example.com]
13 changed: [serverc.lab.example.com]
14 changed: [servere.lab.example.com]
15
16 PLAY RECAP ****
17 serverb.lab.example.com : ok=2    changed=2    unreachable=0   failed=0    skipped=0    rescued=0
18 ignored=0
19 serverc.lab.example.com : ok=2    changed=2    unreachable=0   failed=0    skipped=0    rescued=0
20 ignored=0
20 servere.lab.example.com : ok=2    changed=2    unreachable=0   failed=0    skipped=0    rescued=0
20 ignored=0
```

Figure 12.19: Successful output for the "Log Clients Deploy" job template related job

- 10.4. Click the **Details** tab and verify that the **Status** field displays the **Successful** message.
11. Use the logger "This is a test" command to create a log entry in each of the dev_clients hosts. Verify that servera.lab.example.com, which is the only host in the dev_servers group, receives the log entries.
- 11.1. Open a terminal on the **workstation** machine and use the **logger** command to create a log entry in the **serverb.lab.example.com** and **serverc.lab.example.com** hosts.
- ```
[student@workstation ~]$ ssh serverb.lab.example.com "logger 'This is a test'"
[student@workstation ~]$ ssh serverc.lab.example.com "logger 'This is a test'"
[student@workstation ~]$
```
- 11.2. Verify that the logging server receives the messages. Use the **devops** user in the **servera.lab.example.com** host for consulting the message in the **/var/log/messages** file. This user can run **sudo** commands without entering a password.
- ```
[student@workstation ~]$ ssh devops@servera.lab.example.com \
> "sudo grep 'This is a test' /var/log/messages"
Aug 10 13:06:01 serverb student[1266]: This is a test
Aug 10 13:06:07 serverc student[1263]: This is a test
[student@workstation ~]$
```
12. Use the logger "This is a test" command to create a log entry in the prod_clients hosts. Verify that **serverd.lab.example.com**, which is the only host in the **prod_servers** group, receive the log entry.
- 12.1. Open a terminal on **workstation** and use the **logger** command to create a log entry in the **servere.lab.example.com** and **serverf.lab.example.com** hosts.

```
[student@workstation ~]$ ssh servere.lab.example.com "logger 'This is a test'"  
[student@workstation ~]$ ssh serverf.lab.example.com "logger 'This is a test'"  
[student@workstation ~]$
```

- 12.2. Verify that the logging server receives the messages. Use the devops user in the serverd.lab.example.com host for consulting the message in the /var/log/messages file. This user can run sudo commands without entering a password.

```
[student@workstation ~]$ ssh devops@serverd.lab.example.com \  
> "sudo grep 'This is a test' /var/log/messages"  
Aug 10 13:07:37 servere student[1310]: This is a test  
Aug 10 13:07:41 serverf student[1308]: This is a test  
[student@workstation ~]$
```

Evaluation

On the workstation machine, change to the student user home directory and use the lab command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade compreview-templates
```

Finish

On the workstation machine, change to the student user home directory and use the lab command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish compreview-templates
```

▶ Lab

Configuring Workflow Job Templates, Surveys, and Notifications

Create a workflow job template that uses a survey and sends email notifications.

Outcomes

- Add a survey to an existing job template.
- Create a notification template using the `Email` notification type.
- Create a workflow job template and associate it to the notification template.
- Launch and approve the workflow job template.

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start compreview-workflow
```

Specifications

In this activity, you create a workflow job template on your automation controller. You can log in to your automation controller at <https://controller.lab.example.com> as the `admin` user with `redhat` as the password.

The `lab start compreview-workflow` command creates two job templates, `MOTD Update` and `Network Performance Improvements`. The `Network Performance Improvements` job template sets some `sysctl` kernel tunables (`net.core.netdev_max_backlog` and `net.core.netdev_budget`) for the hosts on which it runs. The `MOTD Update` job template updates the message of the day file (`/etc/motd`).

The `lab start compreview-workflow` command also creates two inventories: `Dev` for development machines (`servera.lab.example.com` and `serverb.lab.example.com`) and `Prod` for production machines (`servere.lab.example.com` and `serverf.lab.example.com`).

Configure a new workflow job template, named `Improving Network Performance from Dev to Prod`, based on the following specification:

- Add a survey to the `MOTD Update` job template using the following information. When done, enable the survey.

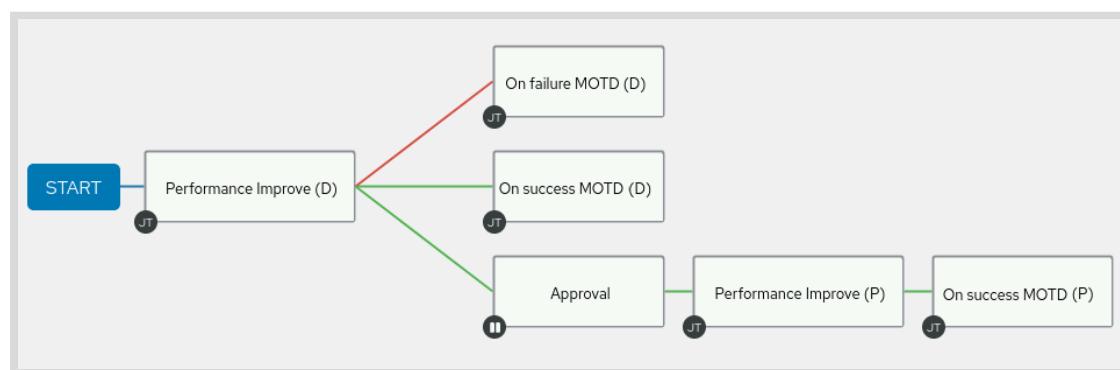
Chapter 12 | Comprehensive Review

Field	Value
Question	What is the performance status for this system?
Answer variable name	performance_message
Answer type	Multiple Choice (single select)
Required	(selected)
Multiple Choice Options	<p>System has NOT been customized to improve network performance. (selected)</p> <p>System has been customized to improve network performance.</p>

- Create a notification template using the following information.

Field	Value
Name	Email AAP Admins
Description	Sends an email to notify the status of the job
Organization	Default
Type	E-mail
Host	localhost
Recipient list	aap-admins@lab.example.com
Sender e-mail	system@controller.lab.example.com
Port	25
Timeout	30

- Create a workflow job template named **Improving Network Performance** from Dev to Prod in the Default organization. Structure it as shown in the following diagram.

**Figure 12.15: The workflow job template**

Chapter 12 | Comprehensive Review

- Set up the first node in the workflow job template to deploy performance improvements to development nodes, based on the following information.

Node: Performance Improve (D)

Field	Value
Node Type	Job Template
Name	Network Performance Improvements
Node Alias	Performance Improve (D)
Inventory	Dev

- The On failure MOTD (D) node runs if the first node (Performance Improve (D)) fails. Configure it based on the information in the following table.

Node: On failure MOTD (D)

Field	Value
Node Type	Job Template
Name	MOTD Update
Node Alias	On failure MOTD (D)
Inventory	Dev
Survey Answer	System has NOT been customized to improve network performance.

- If the first node (Performance Improve (D)) succeeds, then the workflow launches two nodes that are configured based on the following information.

Node: On success MOTD (D)

Field	Value
Node Type	Job Template
Name	MOTD Update
Node Alias	On success MOTD (D)
Inventory	Dev
Survey Answer	System has been customized to improve network performance.

Node: Approval

Field	Value
Node Type	Approval
Name	Approval
Description	Pushing to Prod requires approval
Timeout (min)	60

- If the approval node succeeds, then deploy performance improvements to production nodes based on the following information:

Node: Performance Improve (P)

Field	Value
Node Type	Job Template
Name	Network Performance Improvements
Node Alias	Performance Improve (P)
Inventory	Prod

- If the Performance Improve (P) node succeeds, then launch the MOTD Update job template for production nodes based on the following information:

Node: On success MOTD (P)

Field	Value
Node Type	Job Template
Name	MOTD Update
Node Alias	On success MOTD (P)
Inventory	Prod
Survey Answer	System has been customized to improve network performance.

- Configure your Improving Network Performance from Dev to Prod workflow job template with success and failure notifications using the Email AAP Admins notification template previously discussed in this specification. Do not enable notifications for the Approval or Start options.
- To confirm that your workflow job template is correctly configured:

Launch the Improving Network Performance from Dev to Prod workflow job template and approve it when required.

Chapter 12 | Comprehensive Review

Verify that the completion of the workflow job template triggers an email notification. The student user can access these emails by using ssh to log in to utility.lab.example.com and inspecting the file /var/mail/student, or by installing and using the mutt command to read the mail messages. The password for the student user is student.

Verify that the managed hosts listed in the Dev and Prod inventories display an updated message of the day on login, and that the net.core.netdev_max_backlog and net.core.netdev_budget kernel tunables were modified.

If everything worked, the message of the day should be updated to System has been customized to improve network performance., the net.core.netdev_max_backlog kernel parameter should have a value of 10000, and the net.core.netdev_budget kernel parameter should have a value of 600.

Evaluation

On the workstation machine, change to the student user home directory and use the lab command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade comprevew-workflow
```

Finish

On the workstation machine, change to the student user home directory and use the lab command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish comprevew-workflow
```

► Solution

Configuring Workflow Job Templates, Surveys, and Notifications

Create a workflow job template that uses a survey and sends email notifications.

Outcomes

- Add a survey to an existing job template.
- Create a notification template using the `Email` notification type.
- Create a workflow job template and associate it to the notification template.
- Launch and approve the workflow job template.

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start compreview-workflow
```

1. Go to `https://controller.lab.example.com` and log in as the `admin` user with `redhat` as the password. Verify that the `lab start compreview-workflow` command created the `MOTD Update` and `Network Performance Improvements` job templates, as well as the `Dev` and `Prod` inventories.
 - 1.1. Go to `https://controller.lab.example.com` and log in as the `admin` user with `redhat` as the password.
 - 1.2. Go to **Resources > Templates**, then verify that `MOTD Update` and `Network Performance Improvements` are in the list of job templates.
 - 1.3. Go to **Resources > Inventories**, then verify that `Dev` and `Prod` are in the list of inventories.
2. Add a survey to the `MOTD Update` job template using the following information. When done, enable the survey.

Chapter 12 | Comprehensive Review

Field	Value
Question	What is the performance status for this system?
Answer variable name	performance_message
Answer type	Multiple Choice (single select)
Required	(selected)
Multiple Choice Options	System has NOT been customized to improve network performance. (selected) System has been customized to improve network performance.

- 2.1. Go to Resources > Templates in automation controller.
 - 2.2. Click the MOTD Update link.
 - 2.3. Click the Survey tab.
 - 2.4. Click Add to add a survey.
 - 2.5. Add a question to the survey using the information from the table. When finished, click Save.
 - 2.6. Click Survey Disabled and verify that the label changes to Survey Enabled.
- 3.** Create a notification template using the following information. When finished, click Save. Then, send a test notification to validate that it is working.

Field	Value
Name	Email AAP Admins
Description	Sends an email to notify the status of the job
Organization	Default
Type	E-mail
Host	localhost
Recipient list	aap-admins@lab.example.com
Sender e-mail	system@controller.lab.example.com
Port	25
Timeout	30

- 3.1. Go to Administration > Notifications.
- 3.2. Click Add to create a notification template.

Chapter 12 | Comprehensive Review

- 3.3. Use the information from the table to create the notification template. When finished, click **Save**.
- 3.4. Click **Test** to test the notification process. Then, click the **Back to Notifications** tab. After some seconds, the **Status** column displays the **Successful** message.
4. Create a workflow job template named **Improving Network Performance from Dev to Prod** in the **Default** organization. Then, create the first node in the workflow job template using the following information.

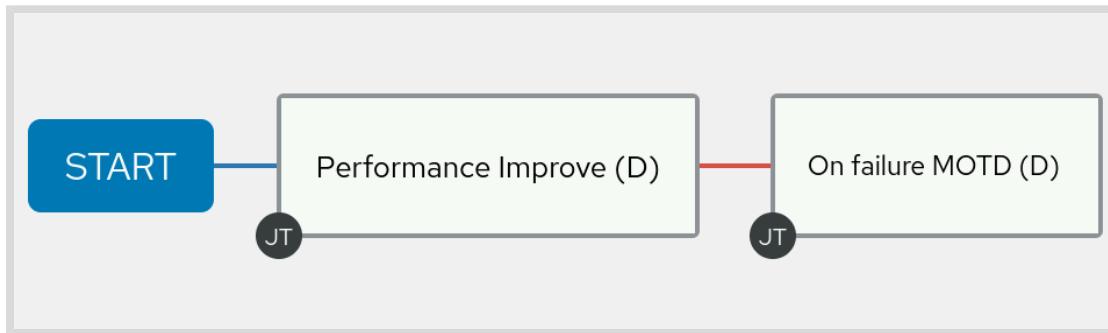
Field	Value
Node Type	Job Template
Name	Network Performance Improvements
Node Alias	Performance Improve (D)
Inventory	Dev

- 4.1. Go to **Resources > Templates**.
- 4.2. Click **Add > Add workflow template**.
- 4.3. Complete the details using the following information, and then click **Save**.
- 4.4. Click **Start** to add the first workflow job template node using the workflow visualizer.
- 4.5. In the **Node Type** list, choose **Job Template** and select the **Network Performance Improvements** job template.
- 4.6. Scroll to the bottom of the page and enter **Performance Improve (D)** in the **Node Alias** field. Click **Next** when done.
- 4.7. Select the **Dev** inventory. Click **Next** when done.
- 4.8. Click **Save** to save the node. Save information about the node, but do not save the workflow job template yet.
5. If the **Performance Improve (D)** node fails, then the workflow job template must launch a job template that updates the message of the day in the development environment hosts. Use the information in the table to create the node. Leave the default answer for the survey, indicating that the system has not been customized to improve the network performance. Save information about the node, but do not save the workflow job template yet.

Chapter 12 | Comprehensive Review

Field	Value
Node Type	Job Template
Name	MOTD Update
Node Alias	On failure MOTD (D)
Inventory	Dev
Survey Answer	System has NOT been customized to improve network performance.

- 5.1. Without leaving the workflow visualizer, hover over the **Performance Improve (D)** node and click **Add a new node**.
- 5.2. Select **On Failure** and click **Next**.
- 5.3. In the **Node Type** list, choose **Job Template** and select the **MOTD Update** job template.
- 5.4. Scroll to the bottom of the page and enter **On failure MOTD (D)** in the **Node Alias** field. Click **Next** when done.
- 5.5. Select the **Dev** inventory and click **Next**.
- 5.6. Leave the default answer for the survey untouched and click **Next**.
- 5.7. Click **Save** to save the node. Save information about the node, but do not save the workflow job template yet.

**Figure 12.17: Workflow job template after adding the 'On failure' node.**

6. If the **Performance Improve (D)** node succeeds, then the workflow must launch two nodes. Use the information in the following table to create the first of the two nodes. Make sure that you add this node after the **Performance Improve (D)** node. Save information about the node, but do not save the workflow job template yet.

Field	Value
Node Type	Job Template
Name	MOTD Update
Node Alias	On success MOTD (D)
Inventory	Dev
Survey Answer	System has been customized to improve network performance.

- 6.1. Without leaving the workflow visualizer, hover over the **Performance Improve (D)** node and click **Add a new node**.
 - 6.2. This time, select **On Success** and click **Next**.
 - 6.3. In the **Node Type** list, choose **Job Template** and select the MOTD Update job template.
 - 6.4. Scroll to the bottom of the page and enter **On success MOTD (D)** in the **Node Alias** field. Click **Next** when done.
 - 6.5. Select the **Dev** inventory and click **Next**.
 - 6.6. On the survey page, select the **System has been customized to improve network performance**. answer, and then click **Next**.
 - 6.7. Click **Save** to save the node, but do not save the workflow job template yet.
7. Use the information in the following table to add a new node to the workflow. Make sure that you add it after the **Performance Improve (D)** node. Save information about the node, but do not save the workflow job template yet.

Field	Value
Node Type	Approval
Name	Approval
Description	Pushing to Prod requires approval
Timeout (min)	60

- 7.1. Without leaving the workflow visualizer, hover over the **Performance Improve (D)** node and click **Add a new node**.
- 7.2. Select **On Success** and click **Next**.
- 7.3. Choose **Approval** in the **Node Type** list.
- 7.4. Enter **Approval** in the **Name** field.
- 7.5. Enter **Pushing to Prod requires approval** in the description of the node.
- 7.6. Modify the **Timeout** field to 60 minutes.

Chapter 12 | Comprehensive Review

- 7.7. Click **Save** to save the node, but do not save the workflow job template yet.

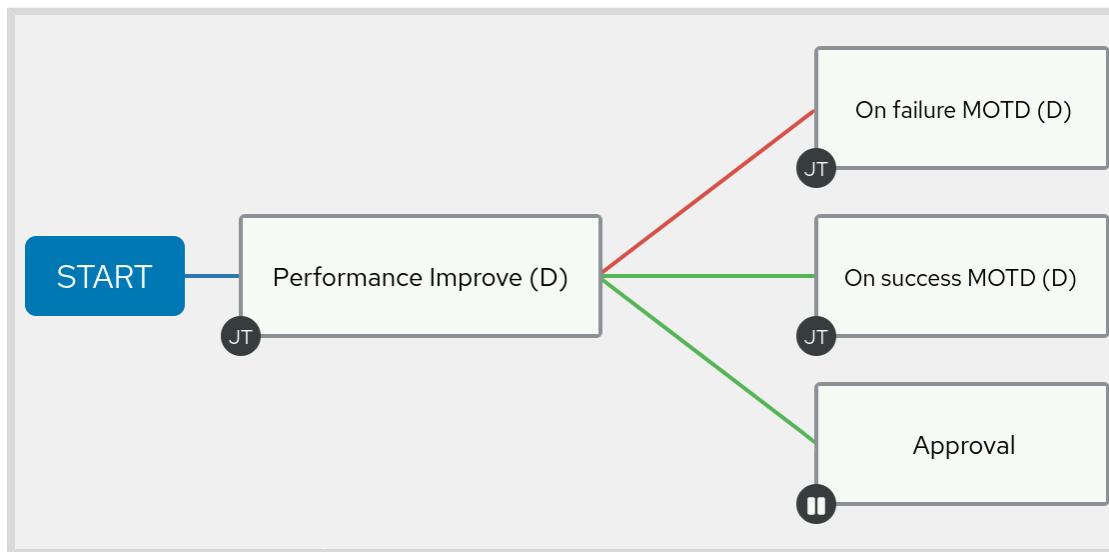


Figure 12.18: Workflow job template after adding the 'On success' nodes

8. If the **Approval** node succeeds, the workflow job template must launch a job template that improves the network performance in your production environment. Create the node using the following information. Save information about the node, but do not save the workflow job template yet.

Field	Value
Node Type	Job Template
Name	Network Performance Improvements
Node Alias	Performance Improve (P)
Inventory	Prod

- 8.1. Without leaving the workflow visualizer, hover over the **Approval** node and click **Add a new node**.
- 8.2. Select **On Success** and click **Next**.
- 8.3. In the **Node Type** list, choose **Job Template** and elect the **Network Performance Improvements** job template.
- 8.4. Scroll to the bottom of the page and enter **Performance Improve (P)** in the **Node Alias** field. Click **Next**.
- 8.5. Select the **Prod** inventory. Click **Next**.
- 8.6. Click **Save** to save the workflow job template node. Save information about the node, but do not save the workflow job template yet.
9. If the **Performance Improve (P)** node succeeds, the workflow job template must launch a job template that updates the message of the day in the production environment hosts. This is the final node in the workflow job template. Use the following information to create the node. Then, save the node and save the workflow job template.

Field	Value
Node Type	Job Template
Name	MOTD Update
Node Alias	On success MOTD (P)
Inventory	Prod
Survey Answer	System has been customized to improve network performance.

- 9.1. Without leaving the workflow visualizer, hover over the **Performance Improve (P)** node and click **Add a new node**.
- 9.2. Select **On Success** and click **Next**.
- 9.3. In the **Node Type** list, choose **Job Template** and select the **MOTD Update** job template.
- 9.4. Scroll to the bottom of the page and enter **On success MOTD (P)** in the **Node Alias** field. Click **Next** when done.
- 9.5. Select the **Prod** inventory and click **Next**.
- 9.6. On the survey page, select **System has been customized to improve network performance**, and then click **Next**.
- 9.7. Click **Save** to save the node, and then click **Save** to save the workflow job template.

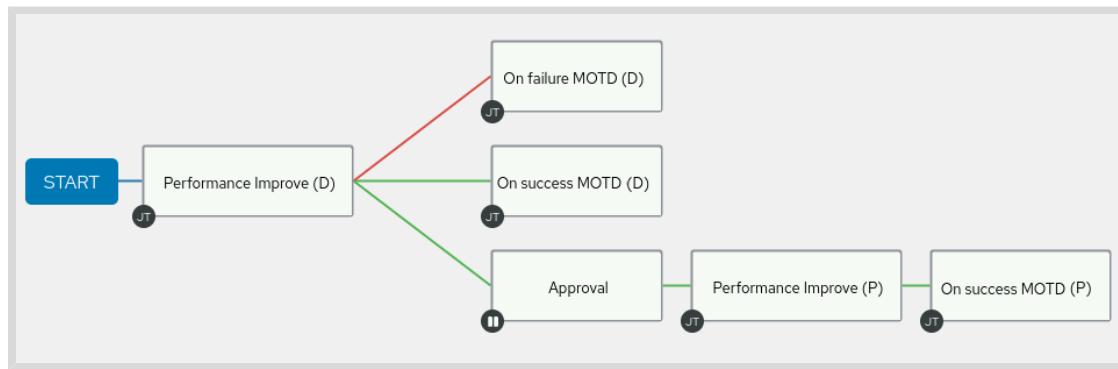


Figure 12.19: The complete workflow job template

10. Configure the **Improving Network Performance from Dev to Prod** workflow job template to use the **Email AAP Admins** notification template. Activate the **Success** and **Failure** notifications. Leave the **Approval** and **Start** options untouched.
 - 10.1. Go to **Resources > Templates**.
 - 10.2. Click the **Improving Network Performance from Dev to Prod** link.
 - 10.3. Click the **Notifications** tab.
 - 10.4. Set the **Success** and **Failure** switches to on for the **Email AAP Admins** notification. Leave the **Approval** and **Start** options untouched.

Chapter 12 | Comprehensive Review

11. Test your work by launching the Improving Network Performance from Dev to Prod workflow job template and approving it when the approval node runs.
 - 11.1. Go to Resources > Templates.
 - 11.2. Click the Launch Template icon for the Improving Network Performance from Dev to Prod workflow job template.
 - 11.3. Observe that the workflow stops at the approval node. Click the notification icon that appears at the top of the page when the workflow stops at the approval node.
 - 11.4. Select the checkbox for the Approval in the workflow job template.
 - 11.5. Click Approve to approve the execution of the remaining workflow job template nodes.
 - 11.6. Go to Views > Jobs and wait for the Status column to display Successful in the job for the Improving Network Performance from Dev to Prod workflow job template.
12. Verify that the Email AAP Admins notification template sent email to the aap-admins@lab.example.com email alias to report the success of the completed job. The student user can access these emails on the utility server.
 - 12.1. Open a terminal and connect to the utility server.

```
[student@workstation ~]$ ssh student@utility
```

- 12.2. Use the tail command to see the email that was sent by automation controller to report the success of the completed job. Your message looks similar to the following:

```
[student@utility ~]$ tail /var/mail/student
{
  "name": "Improving Network Performance from Dev to Prod",
  "url": "https://controller.lab.example.com/#/jobs/workflow/9",
  "created_by": "admin",
  "started": "2022-07-28T23:34:10.961830+00:00",
  "finished": "2022-07-28T23:34:50.959933+00:00",
  "status": "successful",
  "traceback": "",
  "body": "Workflow job summary:\n- node #7 spawns no job.\n- node #9 spawns job #10, \"Network Performance Improvements\", which finished with status successful.\n- node #8 spawns job #11, \"MOTD Update\", which finished with status successful.\n- node #12 spawns job #12, \"Approval\", which finished with status successful.\n- node #11 spawns job #13, \"Network Performance Improvements\", which finished with status successful.\n- node #10 spawns job #14, \"MOTD Update\", which finished with status successful."
}
```

```
[student@utility ~]$
```

- 12.3. Exit the terminal session on the utility system.

```
[student@workstation ~]$ exit
```

13. Confirm that after running your workflow job template, the message of the day and the net.core.netdev_max_backlog and net.core.netdev_budget kernel tunables are modified on the managed hosts in the Dev and Prod inventories.

Chapter 12 | Comprehensive Review

Now, the `net.core.netdev_max_backlog` kernel parameter should have a value of `10000` and the `net.core.netdev_budget` kernel parameter should now have a value of `600`.

- 13.1. From a terminal window, connect to `servera.lab.example.com`. Confirm that the message of the day is `System has been customized to improve network performance.`

```
[student@workstation ~]$ ssh student@servera.lab.example.com
...output omitted...
System has been customized to improve network performance.
...output omitted...
[student@servera ~]$
```

- 13.2. Use the `sudo` command to change to the `root` user, using `student` as the password. Then, verify the values for the `net.core.netdev_max_backlog` and `net.core.netdev_budget` kernel tunables.

```
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]# sysctl net.core.netdev_max_backlog net.core.netdev_budget
net.core.netdev_max_backlog = 10000
net.core.netdev_budget = 600
[root@servera ~]#
```

- 13.3. Log out of the `servera` machine.

```
[root@servera ~]# exit
logout
[student@servera ~]$ exit
logout
Connection to servera.lab.example.com closed.
[student@workstation ~]$
```

- 13.4. Connect to `serverf.lab.example.com` and observe the message of the day, if any.

```
[student@workstation ~]$ ssh student@serverf.lab.example.com
...output omitted...
System has been customized to improve network performance.
...output omitted...
[student@serverf ~]$
```

- 13.5. Use the `sudo` command to change to the `root` user, using `student` as the password. Then, verify the value for the `net.core.netdev_max_backlog` and `net.core.netdev_budget` kernel tunables.

```
[student@serverf ~]$ sudo -i
[sudo] password for student: student
[root@serverf ~]# sysctl net.core.netdev_max_backlog net.core.netdev_budget
net.core.netdev_max_backlog = 10000
net.core.netdev_budget = 600
[root@serverf ~]#
```

Chapter 12 | Comprehensive Review

13.6. Log out of the serverf machine.

```
[root@serverf ~]# exit  
logout  
[student@serverf ~]$ exit  
logout  
Connection to serverf.lab.example.com closed.  
[student@workstation ~]$
```

Evaluation

On the workstation machine, change to the student user home directory and use the lab command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade compreview-workflow
```

Finish

On the workstation machine, change to the student user home directory and use the lab command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish compreview-workflow
```

▶ Lab

Operating Automation Controller using the API

Use the automation controller API to identify and modify your automation controller settings.

Outcomes

- Browse the automation controller API to identify variable names.
- Prepare for changes by modifying a Git repository file used by an existing project.
- Verify that running an existing job template updates your automation controller settings.

Before You Begin

On the workstation machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start compreview-api
```

Specifications

In this activity, you manage the settings on your automation controller by using a "configuration as code" approach. You can log in to your automation controller at `https://controller.lab.example.com` as the `admin` user with `redhat` as the password.

- At `https://controller.lab.example.com/api/v2/settings/all/`, browse the API settings for your automation controller to identify variable names and their values.
 - Identify the variable name used to set the maximum number of forks per job. This setting has a default value of 200 forks.
 - Identify the variable name used to set the maximum number of seconds of inactivity before a user needs to log in again. Automation controller tracks this with a cookie, and the setting has a default value of 1800 seconds.
 - Identify the variable name used to set the maximum number of logged in sessions that a user is allowed to have. This setting uses a default value of -1, which sets no maximum limit.
- Clone the `https://git.lab.example.com/git/comp_api.git` repository to the `/home/student/git-repos` directory. This repository contains one playbook and two variable files. The `modify_settings.yml` playbook updates settings using a list of key-value pairs defined by the `controller_settings` variable in the `vars/settings.yml` variable file. (The playbook uses the `ansible.controller.settings` module to communicate with the automation controller API.) The `vars/auth.yml` variable file contains connection and

Chapter 12 | Comprehensive Review

authentication information for automation controller and is protected using `redhat123` as the Ansible Vault password.

- Modify the `/home/student/git-repos/comp_api/vars/settings.yml` file with new settings. Update the existing key-value pair in the file and add three new key-value pairs. The keys for these key-value pairs must match the name of the relevant variable that you identified from the automation controller API for that setting.
 - Update the existing `ANSIBLE_FACT_CACHE_TIMEOUT` key to use `7200` as the value.
 - Add a new key-value pair to set the maximum number of forks per job to `100` forks.
 - Add a new key-value pair to set the maximum number of seconds of inactivity (before a user needs to log in again) to `3600` seconds.
 - Add a new key-value pair to set the maximum number of logged in sessions that a user is allowed to have to `2` sessions.
 - Add, commit, and push your changes to the remote Git repository. Use the Git commit message: `Changes to controller settings`
- From the automation controller web UI, synchronize the `Controller Settings Comp-API` project and then launch the `Comprevew_API` job template. After the job completes, browse the automation controller API to verify that your automation controller uses the updated settings.

Evaluation

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful. You *must* add, commit, and push any additional changes to the remote Git repository before grading.

```
[student@workstation ~]$ lab grade compreview-api
```

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from earlier exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish compreview-api
```

► Solution

Operating Automation Controller using the API

Use the automation controller API to identify and modify your automation controller settings.

Outcomes

- Browse the automation controller API to identify variable names.
- Prepare for changes by modifying a Git repository file used by an existing project.
- Verify that running an existing job template updates your automation controller settings.

Before You Begin

On the workstation machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start compreview-api
```

1. Use the automation controller web UI to identify the variable names for the settings that you intend to modify.
 1. Go to `https://controller.lab.example.com` and log in as the `admin` user with `redhat` as the password.
 2. Go to `https://controller.lab.example.com/api/v2/settings/all/` and search for `forks`. Automation controller uses the `MAX_FORKS` variable to set the maximum number of forks per job.
 3. Go to `https://controller.lab.example.com/api/v2/settings/all/` and search for either `cookie` or `1800`. Automation controller uses the `SESSION_COOKIE_AGE` variable to set the maximum number of seconds of inactivity before a user needs to log in again.

**Note**

Identifying the name of this variable can be difficult because possible keywords in the description of the variable do not appear in the variable name. One approach to identifying the name of a variable like the SESSION_COOKIE_AGE variable is to manually set the variable with a unique value using the web UI. You can then search the <https://controller.lab.example.com/api/v2/settings/all/> page for the value that you used and identify the variable that has that value.

You can manually change the SESSION_COOKIE_AGE variable by navigating to **Settings** and clicking the **Miscellaneous Authentication settings** link. Click **Edit**, enter a unique value in the **Idle Time Force Log Out** field, and then click **Save**.

- 1.4. Go to <https://controller.lab.example.com/api/v2/settings/all/> and search for sessions. Automation controller uses the SESSIONS_PER_USER variable to set the maximum number of logged in sessions that a user is allowed.
2. Clone the https://git.lab.example.com/git/comp_api.git repository to the /home/student/git-repos directory.
 - 2.1. From a terminal, create the /home/student/git-repos directory if it does not exist, and then change into it.

```
[student@workstation ~]$ mkdir -p ~/git-repos/  
[student@workstation ~]$ cd ~/git-repos/
```

- 2.2. Clone the https://git.lab.example.com/git/comp_api.git repository and then change to the cloned repository.

```
[student@workstation git-repos]$ git clone \  
> https://git.lab.example.com/git/comp_api.git  
Cloning into 'comp_api'...  
...output omitted...  
[student@workstation git-repos]$ cd comp_api
```

- 2.3. Use the tree command to display the files in the repository.

```
[student@workstation comp_api]$ tree  
.  
└── vars  
    ├── auth.yml  
    └── settings.yml  
  
1 directory, 3 files
```

3. Configure the local repository with new settings for your automation controller. When finished, push the changes to the remote Git repository.
 - 3.1. Modify the /home/student/git-repos/comp_api/vars/settings.yml file using the keys and values in the following table.

Key	Value
ANSIBLE_FACT_CACHE_TIMEOUT	7200
MAX_FORKS	100
SESSION_COOKIE_AGE	3600
SESSIONS_PER_USER	2

After you edit the /home/student/git-repos/comp_api/vars/settings.yml file, the file should have the following content:

```
---
controller_settings:
  - key: ANSIBLE_FACT_CACHE_TIMEOUT
    value: 7200
  - key: MAX_FORKS
    value: 100
  - key: SESSION_COOKIE_AGE
    value: 3600
  - key: SESSIONS_PER_USER
    value: 2
```

- 3.2. Add, commit, and push your changes to the remote repository. Use the commit message: Changes to controller settings

```
[student@workstation comp_api]$ git add vars/settings.yml
[student@workstation comp_api]$ git commit -m "Changes to controller settings"
...output omitted...
[student@workstation comp_api]$ git push
...output omitted...
```

4. Apply the changes made to the Git repository using the automation controller web UI.
 - 4.1. Go to Resources > Projects. For the Controller Settings Comp-API project, click the Sync Project icon. Wait until the project displays the Successful status.
 - 4.2. Go to Resources > Templates. For the Comprevue_API job template, click the Launch Template icon. The job succeeds and output from the job indicates changes for each key defined in the settings.yml file.

Chapter 12 | Comprehensive Review

The screenshot shows the 'Output' tab for a job named 'Compreview_API'. The job status is 'Successful'. The output window displays Ansible playbooks running on a single host. The log entries show the configuration of automation controller settings, including 'ANSIBLE_FACT_CACHE_TIMEOUT', 'MAX_FORKS', 'SESSION_COOKIE_AGE', and 'SESSIONS_PER_USER'. The total execution time was 16 minutes and 2 seconds.

Key	Value
ANSIBLE_FACT_CACHE_TIMEOUT	7200
MAX_FORKS	100
SESSION_COOKIE_AGE	3600
SESSIONS_PER_USER	2

Figure 12.19: Compreview_API job output

- 4.3. From the <https://controller.lab.example.com/api/v2/settings/all/> page, search for each key in the following table and verify that it has the expected value.

Key	Value
ANSIBLE_FACT_CACHE_TIMEOUT	7200
MAX_FORKS	100
SESSION_COOKIE_AGE	3600
SESSIONS_PER_USER	2

Evaluation

On the **workstation** machine, change to the **student** user home directory and use the **lab** command to grade your work. Correct any reported failures and rerun the command until successful. You *must* add, commit, and push any additional changes to the remote Git repository before grading.

```
[student@workstation ~]$ lab grade comprevie-api
```

Finish

On the **workstation** machine, change to the **student** user home directory and use the **lab** command to complete this exercise. This step is important to ensure that resources from earlier exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish compreview-api
```

▶ Lab

Backup and Restore Red Hat Ansible Automation Platform

Create a backup of your existing Red Hat Ansible Automation Platform deployment, remove some resources from the deployment, and then use the backup to restore the deployment.

Outcomes

- Create a backup of your Red Hat Ansible Automation Platform deployment.
- Restore your Red Hat Ansible Automation Platform deployment.

Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available. The command performs the following tasks:

- Removes automation controller and private automation hub resources created in previous chapters.
- Downloads and extracts the installation bundle to the `/home/student/aap2.2-bundle` directory (if necessary).
- Creates a job template, a machine credential, a project, and an inventory in automation controller. These resources use the `Backup_Restore` name.

```
[student@workstation ~]$ lab start compreview-backup
```

Specifications

- Go to `https://controller.lab.example.com` and log in as the `admin` user with `redhat` as the password. Verify that the `lab` command created a job template, a machine credential, a project, and an inventory (using `Backup_Restore` as the name of each resource).
- Create a backup of your Red Hat Ansible Automation Platform deployment by using the installer in the `/home/student/aap2.2-bundle` directory. Verify that the backup process created the `automation-platform-backup-latest.tar.gz` symbolic link within that directory.
- Remove the automation controller resources that use the `Backup_Restore` name.



Important

This step is not part of a typical backup and restore process. You remove resources in this exercise so that you can verify that the restoration process recovers the resources.

Chapter 12 | Comprehensive Review

- Use the previously created backup file to restore your Red Hat Ansible Automation Platform deployment. Verify that the restoration process recovered the `Backup_Restore` resources.

Evaluation

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade compreview-backup
```

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish compreview-backup
```

► Solution

Backup and Restore Red Hat Ansible Automation Platform

Create a backup of your existing Red Hat Ansible Automation Platform deployment, remove some resources from the deployment, and then use the backup to restore the deployment.

Outcomes

- Create a backup of your Red Hat Ansible Automation Platform deployment.
- Restore your Red Hat Ansible Automation Platform deployment.

Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available. The command performs the following tasks:

- Removes automation controller and private automation hub resources created in previous chapters.
- Downloads and extracts the installation bundle to the `/home/student/aap2.2-bundle` directory (if necessary).
- Creates a job template, a machine credential, a project, and an inventory in automation controller. These resources use the `Backup_Restore` name.

```
[student@workstation ~]$ lab start compreview-backup
```

1. Use the automation controller web UI to verify the existence of the resources that use the `Backup_Restore` name.
 - 1.1. Go to `https://controller.lab.example.com` and log in as the `admin` user with `redhat` as the password.
 - 1.2. Go to `Resources > Templates`. The `Backup_Restore` job template uses the `Backup_Restore` machine credential, the `Backup_Restore` project, and the `Backup_Restore` inventory.

Click the `Launch Template` icon for the `Backup_Restore` job template. The job succeeds and displays the following output:

```
...output omitted...
ok: [workstation.lab.example.com] => {
    "msg": "This is the backup and restore job template."
}
...output omitted...
```

Chapter 12 | Comprehensive Review

- 1.3. Go to **Resources > Credentials**. The **Backup_Restore** job template uses the **Backup_Restore** machine credential to connect to hosts in the **Backup_Restore** inventory.
- 1.4. Go to **Resources > Projects**. The **Backup_Restore** project contains the playbook used by the **Backup_Restore** job template and the inventory file used by the **Backup_Restore** inventory.
- 1.5. Go to **Resources > Inventories**. The **Backup_Restore** inventory uses the inventory file sourced from the **Backup_Restore** project. The **Backup_Restore** inventory contains the **workstation.lab.example.com** host.

2. Create a backup of your Red Hat Ansible Automation Platform deployment.

- 2.1. From a terminal window, become the **root** user, using **student** as the password.

```
[student@workstation ~]$ sudo -i  
[sudo] password for student: student  
[root@workstation ~]#
```

- 2.2. Change to the **/home/student/aap2.2-bundle** directory.

```
[root@workstation ~]# cd /home/student/aap2.2-bundle/
```

- 2.3. Initiate the backup process using the **setup.sh** script. The script takes a few minutes to complete. If your Red Hat Ansible Automation Platform deployment included automation mesh, then your output also displays the **control2.lab.example.com** host.

```
[root@workstation aap2.2-bundle]# ./setup.sh -b  
.output omitted...  
PLAY RECAP *****  
controller.lab.example.com : ok=59    changed=33    ...    failed=0    skipped=85    ...  
hub.lab.example.com      : ok=37    changed=14    ...    failed=0    skipped=67    ...  
localhost                : ok=2     changed=0     ...    failed=0    skipped=0     ...  
  
The setup process completed successfully.  
[warn] /var/log/tower does not exist.  
Setup log saved to backup.log.
```

- 2.4. Verify that the backup created the **automation-platform-backup-latest.tar.gz** symbolic link. The symbolic link points to the latest backup file.

```
[root@workstation aap2.2-bundle]# ls -l automation-platform-backup-latest.tar.gz  
lrwxrwxrwx. 1 root root 83 Aug  9 14:36 automation-platform-backup-latest.tar.gz -> /home/student/aap2.2-bundle/.automation-platform-backup-2022-08-09-14:34:24.tar.gz
```

3. Remove the automation controller resources that use the **Backup_Restore** name.

**Important**

This step is not part of a typical backup and restore process. You remove resources in this exercise so that you can verify that the restoration process recovers the resources.

- 3.1. Go to **Resources > Templates**. Select the **Backup_Restore** job template and click **Delete**. Verify that you want to delete the template by clicking **Delete**.
- 3.2. Go to **Resources > Credentials**. Select the **Backup_Restore** machine credential and click **Delete**. Verify that you want to delete the credential by clicking **Delete**.
- 3.3. Go to **Resources > Inventories**. Select the **Backup_Restore** inventory and click **Delete**. Verify that you want to delete the inventory by clicking **Delete**.
- 3.4. Go to **Resources > Projects**. Select the **Backup_Restore** project and click **Delete**. Verify that you want to delete the project by clicking **Delete**.
4. Recover your Red Hat Ansible Automation Platform deployment.

- 4.1. Initiate the restoration process using the `setup.sh` script in the `/home/student/aap2.2-bundle` directory. The script takes a few minutes to complete. If your Red Hat Ansible Automation Platform deployment included automation mesh, then your output also displays the `control2.lab.example.com` host.

```
[root@workstation aap2.2-bundle]# ./setup.sh -r
..output omitted...
PLAY RECAP ****
controller.lab.example.com : ok=60    changed=21    ...    failed=0    skipped=85    ...
db.lab.example.com        : ok=6      changed=6      ...    failed=0    skipped=0      ...
hub.lab.example.com       : ok=35    changed=13    ...    failed=0    skipped=56    ...

The setup process completed successfully.
```

```
[warn] /var/log/tower does not exist.
Setup log saved to backup.log.
```

- 4.2. Go to **Resources > Credentials**. The restoration process recovered the **Backup_Restore** machine credential.
 - 4.3. Go to **Resources > Inventories**. The restoration process recovered the **Backup_Restore** inventory.
 - 4.4. Go to **Resources > Projects**. The restoration process recovered the **Backup_Restore** project.
 - 4.5. Go to **Resources > Templates**. The restoration process recovered the **Backup_Restore** job template.
- Click the **Launch Template** icon for the **Backup_Restore** job template. The job succeeds and displays the following output:

```
...output omitted...
ok: [workstation.lab.example.com] => {
    "msg": "This is the backup and restore job template."
}
...output omitted...
```

Evaluation

On the **workstation** machine, change to the **student** user home directory and use the **lab** command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade compreview-backup
```

Finish

On the **workstation** machine, change to the **student** user home directory and use the **lab** command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish compreview-backup
```

