

Ansible Workshop - Exercises

Automation Platform

Learn to manage and run your Ansible
content in AAP.



3 - Projects & Job Templates

Objective

An Ansible automation controller **Project** is a logical collection of Ansible playbooks. You can manage your playbooks by placing them into a source code management (SCM) system supported by automation controller such as Git or Subversion.

This exercise covers:

- Understanding and using an Ansible automation controller Project
- Using Ansible playbooks kept in a Git repository.
- Creating and using an Ansible Job Template

Guide

Setup Git Repository

For this demonstration we will use playbooks stored in a Git repository:

<https://github.com/ansible/workshop-examples>

A playbook to install the Apache web server has already been committed to the directory **rhel/apache**, `apache_install.yml`:

```

---
- name: Apache server installed
  hosts: web

  tasks:
  - name: latest Apache version installed
    yum:
      name: httpd
      state: latest

  - name: latest firewalld version installed
    yum:
      name: firewalld
      state: latest

  - name: firewalld enabled and running
    service:
      name: firewalld
      enabled: true
      state: started

  - name: firewalld permits http service
    firewalld:
      service: http
      permanent: true
      state: enabled
      immediate: yes

  - name: Apache enabled and running
    service:
      name: httpd
      enabled: true
      state: started

```

Tip

Note the difference to other playbooks you might have written! Most importantly there is no `become` set, this has to be done later in the Job template!

To configure and use this repository as a **Source Control Management (SCM)** system in automation controller you have to create a **Project** that uses the repository

Create the Project

- **Automation Execution** → **Projects**, click the **Create Project** button. Fill in the form:

Parameter	Value
Name	Workshop Project
Organization	Default
Default Execution Environment	Default Execution Environment

Source Control Type	Git
---------------------	-----

Enter the URL into the Project configuration:

Parameter	Value
Source Control URL	<code>https://github.com/ansible/workshop-examples.git</code>
Options	Select Clean , Delete and Update Revision on Launch to request a fresh copy of the repository and to update the repository when launching a job.

- Click **Create project**

The new project will be synced automatically after creation, wait for the **Success** state in *Last job status*.

You can also do this manually: Sync the Project again with the Git repository by going to the Projects view and clicking the circular arrow Sync Project* icon in the top right corner.

The screenshot shows the Red Hat Ansible Automation Platform interface. The main content area displays the details for a 'Workshop Project'. The 'Last job status' is highlighted with a red box and shows a green checkmark and the word 'Success'. The interface includes a sidebar with navigation options like 'Overview', 'Automation Execution', 'Jobs', 'Templates', 'Schedules', 'Projects', 'Infrastructure', 'Execution Environments', 'Credentials', and 'Administration'. The top right corner has buttons for 'Sync project' and 'Edit project'.

After starting the sync job, go to the **Jobs** view: there is a new job (from type *Source control update*) for the the Git repository.

Create a Job Template and Run a Job

A job template is a definition and set of parameters for running an Ansible job. Job templates are useful to execute the same job many times. So before running an Ansible **Job** from automation controller you must create a **Job Template** that pulls together:

- **Inventory:** On what hosts should the job run?
- **Credentials** What credentials are needed to log into the hosts?
- **Project:** Where is the playbook?
- **What** playbook to use?

Okay, let's just do that: To create a Job Template, go to the **Automation Execution -> Templates** view, click the **Create template** button and choose **Create job template**.

 **Tip**

Remember that you can often click on the question mark with a circle to get more details about the field.

Parameter	Value
Name	Install Apache
Job Type	Run
Inventory	Workshop Inventory
Project	Workshop Project
Playbook	rhel/apache/apache_install.yml
Execution Environment	Default execution environment
Credentials	Workshop Credentials Machine
Limit	web
Options	<input checked="" type="checkbox"/> Privilege Escalation

- Click **Create job template**

You can start the job by directly clicking the blue **Launch template** button, or by clicking on the rocket in the Job Templates overview. After launching the Job Template, you are automatically brought to the job overview where you can follow the playbook execution in real time.

Job Details

The screenshot shows the 'Job Details' page for the 'Install Apache' job. The page is divided into a left sidebar with navigation options like 'Overview', 'Automation Execution', 'Jobs', 'Schedules', 'Projects', 'Infrastructure', 'Automation Decisions', 'Automation Content', and 'Automation Analytics'. The main content area displays the job details for 'Install Apache', including its name, job type (run), organization (Default), inventory (Workshop Inventory), project (Workshop Project), execution environment (Default execution environment), playbook (rhel/apache/apache_install.yml), credentials (SSH: Workshop Credentials), forks (0), limit (web), verbosity (0 (Normal)), timeout (0), show changes (Off), job slicing (1), last modified (11/20/2024, 8:47:50 PM by admin), and extra variables. There are buttons for 'Launch template' (highlighted with a red box) and 'Edit template'. At the bottom, there are tabs for 'YAML' and 'JSON'.

Job Run

The screenshot shows the 'Job Run' page for the 'Install Apache' job. The page is divided into a top bar with 'Relaunch job' and 'Cancel job' buttons, and a main content area. The main content area shows the job status as 'Success' and the number of plays (1) and elapsed time (00:00:03). Below this, there is a search bar and a list of output lines. The output lines show the job execution details, including the identity added, the play execution time (1:52:20 PM), and the task execution time (1:52:20 PM). The tasks include 'Gathering Facts' and 'latest Apache version installed'. The output lines are as follows:

```
0 Identity added: /runner/artifacts/13/ssh_key_data (/runner/artifacts/13/ssh_key_data)
7
8 PLAY [Apache server installed] ***** 1:52:20 PM
9
10 TASK [Gathering Facts] ***** 1:52:20 PM
11 ok: [node1]
12 ok: [node2]
13 ok: [node3]
14
15 TASK [latest Apache version installed] ***** 1:52:45 PM
16 ok: [node1]
17 ok: [node2]
18 changed: [node3]
19
20 PLAY RECAP *****|
```

Since this might take some time, have a closer look at all the details provided:

- All details of the job template like inventory, project, credentials and playbook are shown.
- Additionally, the actual revision of the playbook is recorded here - this makes it easier to analyse job runs later on.
- Also the time of execution with start and end time is recorded, giving you an idea of how long a job execution actually was.
- Selecting **Output** shows the output of the running playbook. Click on a node underneath a task and see that detailed information are provided for each task of each node.

After the Job has finished go to the main **Jobs** view: All jobs are listed here, you should see directly before the Playbook run an Source Control Update was started. This is the Git update we configured for the **Project** on launch!

Challenge Lab: Check the Result

Time for a little challenge:

- ✔ Use the playbook `check_httpd_service.yml` from the Github Repository <https://github.com/TimGrt/workshop-demo.git> to check the state of the webserver on all `web` hosts.

You have already been through all the steps needed, so try this for yourself.

✓ **Solution**

Go to **Automation Execution** → **Projects**, click the **Create Project** button. Fill in the form:

Parameter	Value
Name	Workshop Demo Content
Organization	Default
Default Execution Environment	Default Execution Environment
Source Control Type	Git

Enter the URL into the Project configuration:

Parameter	Value
Source Control <u>URL</u>	https://github.com/TimGrt/workshop-demo.git
Options	Select Clean, Delete and Update Revision on Launch to request a fresh copy of the repository and to update the repository when launching a job.

Click **Create project**.

Go to **Automation Execution** -> **Templates**, click the **Create template** button and choose **Create job template**.

Parameter	Value
Name	Check webserver service
Job Type	Run
Inventory	Workshop Inventory
Project	Workshop Demo Content
Playbook	check_httpd_service.yml
Execution Environment	Default execution environment
Credentials	Workshop Credentials Machine

▼ Details

Limit	web
Options	<input checked="" type="checkbox"/> Privilege Escalation

To check the state with an *ad-hoc* command, do the following:

- Go to **Automation Execution** → **Infrastructure** → **Inventories** → **Workshop Inventory**
- In the **Automation Execution** → **Infrastructure** → **Inventories** → **Workshop Inventory**, select the **Hosts** tab and select `node1`, `node2`, `node3` and click **Run Command**
- Within the **Details** window, select **Module** `command`, in **Arguments** type `systemctl status httpd` and click **Next**.
- Within the **Execution Environment** window, select **Default execution environment** and click **Next**.
- Within the **Credential** window, select **Workshop Credentials** and click **Next**.
- Review your inputs and click **Finish**.

 **Info**

The output of the results is displayed once the command has completed.

© Tim Grützmaker 2025