

# **Red Hat** Training and Certification

Ansible Automation Platform 2.x  
Webinar

Travis Michette

Version 1.0

# Table of Contents

Introduction to Ansible Automation .....	1
1. Ansible & Ansible Automation Engine (Past) .....	2
1.1. Ansible Infrastructure .....	2
1.1.1. Inventory .....	3
1.1.2. Modules .....	3
1.1.3. Plugins .....	3
1.1.4. Playbooks .....	3
1.1.5. Ansible Tower .....	4
1.2. Ansible Inventory, Ansible Config File, and Ansible Ad-Hoc Commands .....	4
1.2.1. Ansible Inventory .....	4
1.2.1.1. Inventory Variables .....	5
1.2.2. Ansible Config .....	6
1.2.3. Ansible Ad-Hoc Commands .....	7
1.2.4. DEMO - Ansible Ad-Hoc Commands .....	7
1.3. Ansible Playbooks .....	10
1.3.1. Playbook Basics .....	10
1.3.1.1. Task Structure .....	11
1.3.2. Running Playbooks .....	12
1.3.3. DEMO - Running Ansible Playbooks .....	12
1.4. Ansible Roles .....	15
1.4.1. Ansible Role Overview .....	15
1.4.2. Using Roles .....	17
1.4.3. DEMONSTRATION - Using Roles .....	17
2. Ansible Automation Platform 1.x (Present) .....	19
2.1. Ansible Automation Hub .....	19
2.1.1. Ansible Automation Platform .....	19
2.2. Ansible Collections .....	19
2.2.1. Using Ansible Automation Platform Collections .....	19
2.3. Demonstration - Ansible Collections .....	20
3. Ansible Automation Platform 2.x (Future) .....	22
3.1. Introduction to AAP 2.x Components .....	22
3.1.1. Ansible Content Navigator .....	22
3.1.2. Ansible Execution Environments .....	23
3.1.3. DEMO - Using Ansible Content Navigator and Ansible Execution Environments .....	25
3.2. Introduction to AAP 2.x - Ansible Automation Hub .....	30
3.2.1. Private Automation Hub .....	30

3.2.2. Custom Execution Environments .....	31
3.2.3. DEMO - Creating a Custom Execution Environment and Publishing to Private Automation Hub .....	34
3.3. Introduction to AAP 2.x - Ansible Controller (formerly Ansible Tower).....	40
3.3.1. Organizations, Teams, and RBAC .....	41
3.3.1.1. DEMO - Creating Organizations, Teams, and Users.....	41
3.3.2. Inventories and Credentials .....	47
3.3.2.1. DEMO - Creating Inventories and Credentials .....	47
3.3.3. Projects and Job Templates.....	53
3.3.3.1. DEMO - Projects and Job Templates .....	53
3.3.4. Workflows.....	59
3.3.4.1. DEMO - Creating Job Workflows .....	59

# Introduction to Ansible Automation

# Chapter 1. Ansible & Ansible Automation Engine (Past)

## 1.1. Ansible Infrastructure

The Ansible infrastructure and Ansible Automation consists of multiple components. The initial main foundation of Ansible is the Ansible Automation Engine.

For the most part, Ansible is a declarative automation platform that is considered idempotent meaning that Ansible will only execute tasks and plays if the item needs to be changed/modified. Otherwise, Ansible will skip to the next task or play in a playbook.

### *Ansible Components*

- **Control Node:** System with Ansible installed, contains Ansible inventory files, **ansible.cfg**, and playbooks. This system manages and controls other managed hosts/nodes.
- **Managed Host/Managed Node:** System or node being managed in the Ansible environment. The **Control Node** executes various Ansible modules against these devices.

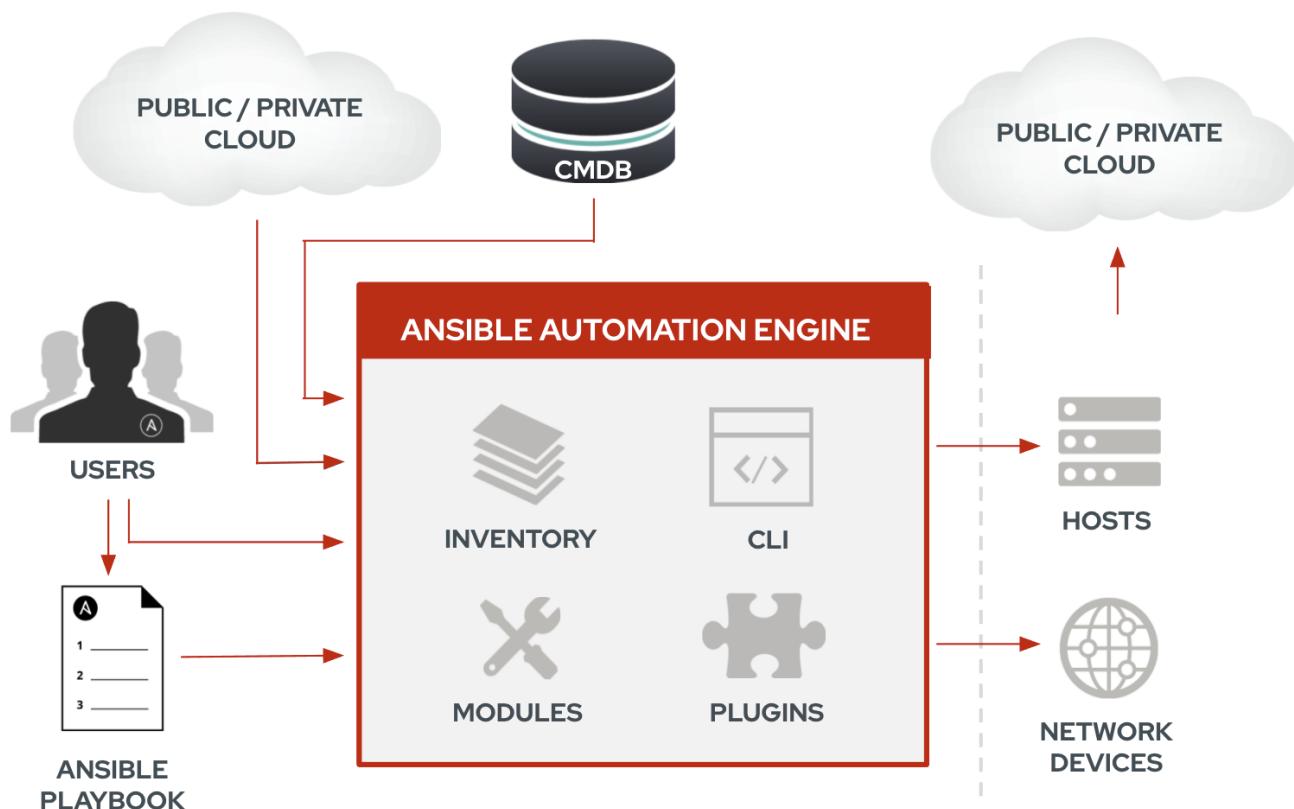


Figure 1. Ansible Automation

*Ansible Automation Engine*

- Inventory
- Command-Line Interface (CLI)
- Modules (Generally Python/Powershell)
- Plugins

Ansible Automation Engine utilizes the **ansible** command for Ad-Hoc Ansible Automation or the **ansible-playbook** command for running multiple tasks by leveraging and Ansible playbook containing one or more plays consisting of one or more tasks.

### 1.1.1. Inventory

List of systems in the infrastructure to be managed. Inventories can be static, dynamic, or a combination of both static and dynamic. Ansible also allows inventories to contain variables for the devices being managed. Devices must exist in inventory in order for Ansible to be capable of managing the devices.

### 1.1.2. Modules

Code utilized by the Ansible core engine which is used to perform a given tasks. Most modules are written in Python for Linux and Powershell for Windows. Modules can extend Ansible automation to multiple platforms simplifying and extending the automation to the entire stack.

#### *Non-Idempotent Modules*



There are some Ansible modules that aren't idempotent. Modules such as **command**, **shell**, and **raw** to name a few will execute regardless of the state. It is possible to use these modules with logic to make a playbook idempotent, but it is recommended to find an actual Ansible module to perform the task. These modules should be used as a last resort when no other module exists to perform a task.

### 1.1.3. Plugins

Code utilized by the Ansible core engine which is used to manipulate, transform, or otherwise modify either data in the playbook or items captured by the playbook and modules so that it is adaptable and usable on different platforms.

### 1.1.4. Playbooks

List of sequential tasks to allowing individual Ansible modules to be executed to perform a sequence of steps in an automation task. Playbooks are written in YAML and are simple easy-to-read steps on the end state of the system.

### 1.1.5. Ansible Tower

Ansible Tower delivers enterprise management and features to the Ansible family. Through Tower, Ansible can provide the following:

- Role-Based Access Control (RBAC)
- Restful API
- Push button deployment
- Workflows
- Credential and Secret Management
- Integration into SCM systems
- Integration into other management systems for dynamic inventory
- WebUI
- ... and more

Ansible Tower allows enterprises to manage their IT environment by providing a centralized web solution to end-users and administrators to perform automation and self-service tasks.

## 1.2. Ansible Inventory, Ansible Config File, and Ansible Ad-Hoc Commands

### 1.2.1. Ansible Inventory

Ansible inventories can be either static, dynamic, or a combination of both static and dynamic. The traditional form of the Ansible inventory file is either YAML or INI. Inventory items consist of either individual managed nodes or groups of managed nodes.

*Listing 1. Ansible Inventory File INI Format*

```
servera ①
serverb
serverc
serverd

[load_balancers] ②
servere
serverf
```

① Individual managed host/node

② Inventory Group

### *Converting INI to YAML Inventory*

Ansible provides the **ansible-inventory** command that will easily allow the inventory to be transformed from one form to another.



```
ansible-inventory --list -y
```

### *Listing 2. Ansible Inventory File YAML Format*

```
all:
  children:
    load_balancers: ①
      hosts:
        servere: {}
        serverf: {}
    ungrouped:
      hosts: ②
        servera: {}
        serverb: {}
        serverc: {}
        serverd: {}
```

① Inventory Hosts in a Group

② Individual managed host/node (ungrouped)

#### **1.2.1.1. Inventory Variables**

It is possible for Ansible playbooks and Ansible ad-hoc commands to utilize inventory variables. These variables can be defined directly within the static inventory files themselves or those variables can be defined within the directory structure of the project utilizing either project directories or inventory directories.

### *Keep Inventory Simple and Organized*



It is extremely important not to define variables for inventory in multiple locations as variable precedence and variable merging comes into play. It is equally important to devise an inventory strategy on where/how variables will be defined so that the playbooks and automation goals are kept simple and easy to understand and follow.

### *Listing 3. Sample Inventory File with Variables*

```
[app1srv]
appserver01 ansible_host=10.42.0.2 ①
appserver02 ansible_host=10.42.0.3

[web]
```

```

node-[1:30] ansible_host=10.42.0.[31:60]

[web:vars] ②
apache_listen_port=8080
apache_root_path=/var/www/mywebdocs/

[all:vars] ③
ansible_user=kev
ansible_ssh_private_key_file=/home/kev/.ssh/id_rsa

```

- ① Defined variable at a **host** level
- ② Defined variables at a **group** level
- ③ Defined variables for all inventory items

## 1.2.2. Ansible Config

The **ansible.cfg** file controls how the **ansible** and **ansible-playbook** commands are run and interpreted. The configuration file has two (2) main sections that are commonly used, but include other sections as well. For the purpose of understanding how Ansible works, we will examine both the **[defaults]** section and the **[privilege\_escalation]** section.

*Listing 4. ansible.cfg Defaults Section*

```

[defaults]
inventory = inventory ①
remote_user = devops ②

```

- ① Specifies which inventory file Ansible will use
- ② Specifies the remote user to be used by **ansible** or **ansible-playbook** commands.



A perfectly acceptable **ansible.cfg** might only have a **[defaults]** section specifying the inventory to be used.

*Listing 5. ansible.cfg Privilege Escalation Section*

```

[privilege_escalation]
become = False ①
become_method = sudo ②
become_user = root ③
become_ask_pass = False ④

```

- ① Sets default behavior whether to elevate privileges
- ② Sets method for privilege escalation
- ③ Sets username of privileged user

- ④ Sets option on whether or not user is prompted for password when performing privilege escalation.

#### *Ansible Config File Precedence*

- **ANSIBLE\_CONFIG** - Environment Variable (highest)
- **ansible.cfg** - Config file in current working directory (most common and recommended)
- **~/.ansible.cfg** - Ansible config file in the home directory
- **/etc/ansible/ansible.cfg** - Ansible's installed default location (lowest)

### 1.2.3. Ansible Ad-Hoc Commands

Ansible Ad-Hoc commands are most often used to quickly perform an automation task using a single Ansible module. These commands can be executed against one or more hosts in the Ansible inventory file.

*Table 1. Ansible Ad-Hoc Command Arguments*

Command Argument	Description
<b>-m MODULE_NAME</b>	Module name to execute for the ad-hoc command
<b>-a MODULE_ARGS</b>	Module arguments needed for the ad-hoc command
<b>-b</b>	Runs ad-hoc command as a privileged user
<b>-K</b>	Runs ad-hoc command as a privileged user and requests the <b>become</b> password
<b>-e EXTRA_VARS</b>	Provides extra variables as <b>KEY=VALUE</b> to be used for the execution of the ad-hoc command

### 1.2.4. DEMO - Ansible Ad-Hoc Commands

Demonstration and hands-on workshop for Ad-Hoc commands. The demo will utilize the **ping** module to ensure that the **ansible.cfg** and the **inventory** file are correctly setup and working within the Ansible environment.

*Example 1. DEMONSTRATION - Ansible Ping*

1. Change to correct directory

```
[student@workstation ~]$ cd /home/student/Github/AAP_Webinar/Past/ad-hoc
```

2. Run the **ansible** ad-hoc command

```
[student@workstation ad-hoc]$ ansible -m ping all
```

```

servere | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}

servera | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}

serverc | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}

serverb | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}

serverd | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}

serverf | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}

```

*Checking Sudoers Ability and Setup*



***Listing 6. Checking ansible.cfg for Ability to BECOME without sudo Password***

```
[student@workstation ad-hoc]$ ansible -m ping all --become
```

***Listing 7. Checking ansible.cfg for Ability to BECOME with sudo and Prompting for Password***

```
[student@workstation ad-hoc]$ ansible -m ping all --become -K  
BECOME password:
```

The next demonstration will use the **copy** module to create a user in the managed systems making an entry to the **sudoers** file.

**Example 2. DEMONSTRATION - Ansible Ad-Hoc Command to Create User and Sudoers File**

1. Change to correct directory

```
[student@workstation ~]$ cd /home/student/Github/AAP_Webinar/Past/ad-hoc
```

2. Run the **ansible** commands to create the user and update the **sudoers** file.

- a. Create the user on the remote system.

```
[student@workstation ad-hoc]$ ansible -m user -a 'name=travis uid=1040  
comment="Travis Michette" group=wheel' servera -b  
servera | SUCCESS => {  
    "ansible_facts": {  
        "discovered_interpreter_python": "/usr/libexec/platform-python"  
    },  
    "append": false,  
    "changed": false,  
    "comment": "Travis Michette",  
    "group": 10,  
    "home": "/home/travis",  
    "move_home": false,  
    "name": "travis",  
    "shell": "/bin/bash",  
    "state": "present",  
    "uid": 1040  
}
```

- b. Create the user in a **sudoers** file.

```
[student@workstation ad-hoc]$ ansible -m copy -a 'content="travis ALL=(ALL) NOPASSWD:ALL" dest=/etc/sudoers.d/travis' servera -b
```

### 3. Test new user and sudo rights

#### a. SSH to servera

```
[student@workstation ad-hoc]$ ssh travis@servera
```

#### b. sudo without a password

```
[travis@servera ~]$ sudo -i  
[root@servera ~]#
```

## 1.3. Ansible Playbooks

Ansible playbooks contain one or more tasks to execute against specified inventory nodes. Playbooks consist of one or more play and each play in a playbook consists of one or more tasks. Ansible playbooks and tasks are all about **key:value** pairs and **lists**. Understanding this basic format allows someone developing Ansible to form playbooks that are easier to create, troubleshoot/debug, and for someone else to understand.

### 1.3.1. Playbook Basics

An Ansible playbook is written/formatted in YAML so horizontal whitespace is critical and often the most troublesome part of debugging new Ansible playbooks. Playbooks have a general structure for the plays with directives such as: **name**, **hosts**, **vars**, **tasks**, and more. These play-level directives help form a readable structure much like **task-level** directives.

*Listing 8. Play Structure Components*

```
---  
- name: install and start apache ①  
  hosts: web ②  
  become: yes ③  
  
  tasks: ④
```

① Name of play in playbook

② List of hosts from inventory to execute play against (**required**)

③ Directive to override **ansible.cfg** and elevate privileges

④ Beginning of **tasks** section.

There can be other directives here, but at the most basic playbook, you will generally always see a **hosts** and a **tasks** section.



#### *Naming Plays*

It is recommended and considered a best practice to name all plays within a playbook.

The first indentation level in a playbook denoted by - is the list of plays and this level will contain the **key:value** pairs that correspond to Ansible playbook directives. Understanding this and developing good habits and standards for indentations allows Ansible users to create playbook skeletons which help tremendously during the development/debugging cycle.

#### 1.3.1.1. Task Structure

A task within a playbook is generally specified similar to a play having a **name** section so that it is easier to debug.

*Listing 9. Task Structure and Components*

```
tasks:
  - name: httpd package is present ①
    yum: ②
      name: httpd ③
      state: latest ④

  - name: latest index.html file is present ⑤
    template:
      src: files/index.html
      dest: /var/www/html/

  - name: httpd is started ⑥
    service:
      name: httpd
      state: started
```

① Name of first task in playbook

② Name of module to be used in first task in playbook

③ Argument/Option provided to module, in this instance **name** and is **required** for the package name in the case of the **yum** module.

④ Argument/Option provided to module, in this instance the **state** describes whether the module will install, update, or remove the package for the **yum** module.

⑤ Name of second task in playbook

⑥ Name of third task in playbook

### Naming Tasks



It is recommended and considered a best practice to name all tasks within a playbook. Naming tasks especially helps with debugging issues in playbooks as the Ansible STDOUT will display and record task names.

The second indentation level in a playbook denoted by - is generally under the **tasks:** section and contains the list of tasks. This level will contain the **key:value** pairs that correspond to Ansible task directives always starting with the module being used at the same level before going to the third indentation level which are the **key:value** pair options that belong to the module being used in that task.

### 1.3.2. Running Playbooks

Playbooks can be run just like Ansible **ad-hoc** commands. In order to execute or run a playbook, it is necessary to use the **ansible-playbook** command and specify the playbook. The additional options available for the **ad-hoc** commands such as: **-e**, **-K**, **-b**, and others all still apply and perform the same functions when leveraged with the **ansible-playbook** command.

### 1.3.3. DEMO - Running Ansible Playbooks

#### *Example 3. DEMONSTRATION - Running Ansible Playbooks*

In this demonstration, we will be creating a user on **serverb** using playbooks as opposed to leveraging **ad-hoc** commands.

1. Switch to correct directory

```
[student@workstation ~]$ cd ~/Github/AAP_Webinar/Past/Playbooks
```

2. Examine playbook

```
[student@workstation Playbooks]$ vim playbook.yml
```

```
---
- name: Playbook to Create User and Sudoers without Password
  hosts: serverb
  tasks:
    - name: Create User Named Travis
      user:
        name: travis
        uid: 1040
        comment: "Travis Michette"
        group: wheel
```

```
- name: Create User in Sudoers File
  copy:
    content: "travis ALL=(ALL) NOPASSWD:ALL\n"
    dest: /etc/sudoers.d/travis
    validate: /usr/sbin/visudo -csf %s
```

### 3. Execute and run the playbook

```
[student@workstation Playbooks]$ ansible-playbook playbook.yml -b
PLAY [Playbook to Create User and Sudoers without Password]
*****
... OUTPUT OMITTED ...
```

### 4. Test and Verify User

#### a. SSH to remote system

```
[student@workstation Playbooks]$ ssh travis@serverb
```

#### b. Verify Sudo without Password

```
[travis@serverb ~]$ sudo -i
[root@serverb ~]#
```

### *Example 4. DEMONSTRATION - Failure of Old Playbook*

It is important to constantly test playbooks with the most current and recent versions of Ansible to ensure all modules work as expected and items haven't been deprecated. The following playbook was developed for use with Ansible 2.8 and earlier. The playbook now fails as some of the modules being used have been migrated from Ansible **built-in** modules to Ansible collections. More on this migration and discussion of collections will come in future chapters and sections.

### 1. Examine Playbook for Website

```
[student@workstation Playbooks]$ cat Website_Past.yml
---
- name: Playbook to Fully Setup and Configure a Webserver
  hosts: servera
  tasks:
    - name: Install Packages for Webserver
```

```

yum:
  name:
    - httpd
    - firewalld
  state: latest

- name: Create Content for Webserver
  copy:
    content: "I'm an awesome webserver"
    dest: /var/www/html/index.html

- name: Create Content for Webserver
  copy:
    content: "I'm an awesome webserver"

- name: Firewall is Enabled
  service:
    name: firewalld
    state: started
    enabled: true

- name: HTTP Service is Open on Firewall
  firewalld:
    service: http
    state: enabled
    permanent: true
    immediate: yes

- name: httpd is started
  systemd:
    name: httpd
    state: started
    enabled: true

```

## 2. Execute the playbook

```
[student@workstation Playbooks]$ ansible-playbook Website_Past.yml
ERROR! couldn't resolve module/action 'firewalld'. This often indicates a
misspelling, missing collection, or incorrect module path. ①
```

The error appears to be in  
 '/home/student/Github/AAP\_Webinar/Past/Playbooks/Website\_Past.yml': line 27,  
 column 7, but may  
 be elsewhere in the file depending on the exact syntax problem.

The offending line appears to be:

- name: HTTP Service is Open on Firewall  
  ^ here

- ① The **firewalld** module is not available. This was moved in AAP 2.x to an Ansible collection and is no longer able to be referenced without the collection and module being installed.

#### *Test Often*

As Ansible has changed going into Ansible Automation Platform 2+, many changes have been made. There was a duplication and mapping of several of the modules allowing for existing playbooks to still run correctly, however, at some point modules become completely deprecated, and mappings get removed. It is extremely important to execute old playbooks and test with new versions of Ansible and to look for **deprecation warnings** so that playbooks can be fixed proactively instead of reactively.



## 1.4. Ansible Roles

Ansible Roles allow Ansible developers to create re-usable code snippets and tasks that can be shared in the form of Ansible Roles. These roles are commonly shared via Ansible Galaxy (<https://galaxy.ansible.com/>) via Github projects by the role developers. With the new Ansible Automation Platform (AAP) 2.x and beyond roles are also included as part of Ansible Collections which will be covered as part of a later topic.

### 1.4.1. Ansible Role Overview

Ansible roles are reusable Ansible components that allow common tasks to be repeated/repurposed without needing to re-write or create custom playbooks. Ansible roles work the same way as Ansible Playbooks and tasks except that roles are generally published/shared generically to be used by others to perform a task or set of tasks in automation playbooks.

Roles provide Ansible with a way to load tasks, handlers, and variables from external files. Static files and templates can also be associated and referenced by a role.

#### Role Benefits

- Roles group content which allows easy sharing of code with others
- Roles can be written that define the essential elements of system type: web server, database server, git repository, or other purpose
- Roles make larger projects more manageable
- Roles can be developed in parallel by different administrators

Table 2. Ansible role subdirectories

Subdirectory	Function
<b>defaults</b>	The <b>main.yml</b> file in this directory contains the default values of role variables that can be overwritten when the role is used.
<b>files</b>	This directory contains static files that are referenced by role tasks.
<b>handlers</b>	The <b>main.yml</b> file in this directory contains the role's handler definitions.
<b>meta</b>	The <b>main.yml</b> file in this directory defines information about the role, including author, license, platforms, and optional role dependencies.
<b>tasks</b>	The <b>main.yml</b> file in this directory contains the role's task definitions.
<b>templates</b>	This directory contains Jinja2 templates that are referenced by role tasks.
<b>tests</b>	This directory can contain an inventory and <b>test.yml</b> playbook that can be used to test the role.
<b>vars</b>	The <b>main.yml</b> file in this directory defines the role's variable values.

### Defining Variables and Defaults

- **Role** variables are defined by creating **var/main.yml** with name/value pairs in the role directory hierarchy.
- **Default variables** are defined by creating a **defaults/main.yml** file with name/value pairs in the role directory hierarchy.



It is best to define a given variable in either **var/main.yml** or **defaults/main.yml** but not both.



**Playbook Roles and Include Statements:** [http://docs.ansible.com/ansible/playbooks\\_roles.html](http://docs.ansible.com/ansible/playbooks_roles.html)

### Reference for Roles



There is another workshop with some information on creating and publishing Ansible Roles.

[https://github.com/tmichett/LUG/blob/main/Ansible\\_Roles/Workbook/Ansible.adoc](https://github.com/tmichett/LUG/blob/main/Ansible_Roles/Workbook/Ansible.adoc)

## 1.4.2. Using Roles

In order to use Ansible roles, they must first be installed and made available on the Ansible control node utilizing the role.

*Listing 10. Installing an Ansible Role*

```
$ ansible-galaxy install tmichett.deploy_packages
```

## 1.4.3. DEMONSTRATION - Using Roles

*Example 5. DEMONSTRATION - Using a Role from Ansible Galaxy*

1. Change to correct working directory

```
[student@workstation ~]$ cd ~/Github/AAP_Webinar/Past/Roles/
```

2. Install Role from Ansible Galaxy

```
[student@workstation Roles]$ ansible-galaxy install -r roles/requirements.yml -p roles
Starting galaxy role install process
- downloading role 'ansiblize', owned by tmichett
- downloading role from
https://github.com/tmichett/Ansiblize/archive/master.tar.gz
- extracting tmichett.ansiblize to
/home/student/Github/AAP_Webinar/Past/Roles/roles/tmichett.ansiblize
- tmichett.ansiblize (master) was installed successfully
```

3. Install Collections

```
[student@workstation Roles]$ ansible-galaxy collection install -r
collections/requirements.yml -p collections/
Starting galaxy collection install process
[WARNING]: The specified collections path
'/home/student/Github/AAP_Webinar/Past/Roles/collections' is not part of the
configured Ansible collections paths
'/home/student/.ansible/collections:/usr/share/ansible/collections'. The installed
collection won't be picked up in an Ansible run.
Process install dependency map
Starting collection install process
Downloading https://galaxy.ansible.com/download/ansible-posix-1.3.0.tar.gz to
/home/student/.ansible/tmp/ansible-local-37837_73lx2j8/tmpnadbl_rx/ansible-posix-
1.3.0-xr73p6ye
```

```
Installing 'ansible.posix:1.3.0' to  
'/home/student/Github/AAP_Webinar/Past/Roles/collections/ansible_collections/ansib  
le posix'  
ansible.posix:1.3.0 was installed successfully
```



Because we are using a newer Ansible version, the regular modules are no longer available so we must install the Ansible Posix collection to use some of the modules in the role.

#### 4. Create/Modify Playbook with Correct Values and Providing Variables for the Role

```
[student@workstation Roles]$ cat Roles_Playbook_Demo.yml  
---  
- name: Playbook to Fully Setup and Configure a new User with a Role  
  hosts: serverc  
  vars: ①  
    ansible_user_name: travis  
    ansible_user_password: redhat  
    ssh_key_answer: no  
  roles: ②  
    - tmichett.ansiblize
```

① Providing required variables for the role

② Providing the role being used

#### 5. Run the playbook with the **ansible-playbook** Command

```
[student@workstation Roles]$ ansible-playbook Roles_Playbook_Demo.yml -b
```

#### 6. Test the results on **serverc**

##### a. SSH to ServerC

```
[student@workstation Roles]$ ssh travis@serverc
```

b. Attempt to become root without password

```
[travis@serverc ~]$ sudo -i  
[root@serverc ~]#
```

# Chapter 2. Ansible Automation Platform 1.x (Present)

## 2.1. Ansible Automation Hub

Ansible Automation hub was released so that enterprises could host their own Ansible collections, Ansible execution environments in a disconnected fashion. Ansible Automation Hub provides self-hosted services from Ansible Galaxy as well as services provided by Red Hat's Ansible Automation Hub found at <https://console.redhat.com/ansible/ansible-dashboard>.

### 2.1.1. Ansible Automation Platform

Provides curated collections, modules, roles that are supported by Red Hat or Red Hat Partners. The collections, modules, roles, and playbooks downloaded from Red Hat Automation Hub are supported and required an Ansible Automation Platform subscription entitlement. This is different from the unsupported community modules/collections/roles found at Ansible Galaxy (<https://galaxy.ansible.com/>). :pygments-style: tango :source-highlighter: pygments :icons: font :icons: font

## 2.2. Ansible Collections

Ansible 2.9 introduced the concept of collections and provided mapping for Ansible modules that were moved into a collection namespace. Ansible 2.9 provided a mapping of the new module locations in collections and this mapping automatically works for existing Ansible playbooks in the initial versions of Ansible Automation Platform.

*Ansible Module and Collection Mapping*



[https://github.com/ansible/ansible/blob/devel/lib/ansible/config/ansible\\_builtin\\_runtime.yml](https://github.com/ansible/ansible/blob/devel/lib/ansible/config/ansible_builtin_runtime.yml)

### 2.2.1. Using Ansible Automation Platform Collections

Collections allowed development of Ansible core components to be separated from module and plug-in development. Upstream Ansible unbundled modules from Ansible core code beginning with Ansible Base (core) 2.10/2.11. Newer versions of Ansible require collections to be installed in order for modules to be available for Ansible. Playbooks should be developed using the FQCNs when referring to modules in tasks. Existing playbooks can be fixed easily to work with collections, but it is better to re-write the playbooks to use the fully-qualified collection name (FCQN).

*Downloading Collections*



Collections are brought into the entire structure via three ways: Requirements files next to your playbooks describe what collections should be present for the

automation-content at hand. Automation controller can read those, or searches to Private Automation hub automatically for needed collections and pull them automatically. There is also a CLI for downloading collections.

## 2.3. Demonstration - Ansible Collections

*Example 6. DEMONSTRATION - Using Ansible Collections*

1. Change to the correct working directory

```
[student@workstation ~]$ cd /home/student/Github/AAP_Webinar/Present/Playbooks
```

2. View the Playbook

```
[student@workstation Playbooks]$ vim Website_Present.yml
---
- name: Playbook to Fully Setup and Configure a Webserver
  hosts: servers
  tasks:
    - name: Install Packages for Webserver
      yum:
        name:
          - httpd
          - firewalld
        state: latest

    - name: Create Content for Webserver
      copy:
        content: "I'm an awesome webserver\n"
        dest: /var/www/html/index.html

    - name: Firewall is Enabled
      systemd:
        name: firewalld
        state: started
        enabled: true

    - name: HTTP Service is Open on Firewall
      ansible.posix.firewalld: ①
        service: http
        state: enabled
        permanent: true
        immediate: yes

    - name: httpd is started
```

```
systemd:  
  name: httpd  
  state: started  
  enabled: true
```

- ① In the newer version of Ansible the **firewalld** module is located in the **ansible.posix.firewalld** collection.



Beginning with Ansible version 2.9 collections contain modules that were previously included as part of Ansible core. Ansible version 2.9.x contains mapping for modules in their collections and Ansible version 2.9 also includes all the collections as part of the Ansible Core distribution. Never versions of Ansible however, don't have the collection mapping nor are the modules and collections pre-installed so it is necessary to install collections.

Ansible Module Mapping: [https://github.com/ansible/ansible/blob/devel/lib/ansible/config/ansible\\_builtin\\_runtime.yml](https://github.com/ansible/ansible/blob/devel/lib/ansible/config/ansible_builtin_runtime.yml)

### 3. Install the Ansible Collections

```
[student@workstation Playbooks]$ ansible-galaxy collection install --collections/requirements.yml -p collections/
```



Since we are using collections, it is very important to install the collections before attempting to run the playbook. Collections are typically installed into the local project directory called **collections** and they are usually listed in a **requirements.yml** file.

### 4. Execute the playbook with the **ansible-playbook** command

```
[student@workstation Playbooks]$ ansible-playbook Website_Present.yml -b
```

### 5. Test the Website

```
[student@workstation Playbooks]$ curl servera  
I'm an awesome webserver
```

# Chapter 3. Ansible Automation Platform 2.x (Future)

## 3.1. Introduction to AAP 2.x Components

With Ansible 2.x, many new items have emerged. Ansible can now leverage execution environments which contain Ansible collections, different versions of Ansible, and the correct python versions and modules needed to execute a playbook. Ansible Execution Environments (EEs) allow developers and administrators more flexibility by leveraging a containerized Ansible environment. The use of Ansible EEs from a control node or developer workstation will work exactly the same with Ansible Controller (formerly known as Ansible Tower). Ansible Controller allows Ansible EEs to be assigned to projects, inventories, job templates, and more.

Ansible Private Automation Hub (Automation Hub) allows a self-hosted Ansible Galaxy. The local installation of Ansible Galaxy not only allows collections to be published privately within an organization, but it also provides a container registry for publishing custom Ansible EEs.

Ansible Content Navigator is a new tool that can be used to test and develop Ansible playbooks.

### 3.1.1. Ansible Content Navigator

The **ansible-navigator** tool replaces and extends the functionality of the **ansible-playbook**, **ansible-inventory**, **ansible-config** commands and more. Ansible Content Navigator allows the Ansible Control Node to be separated from the execution environment as playbooks are now run in a container. This makes it easier to move from development environments to production environments as the execution environment is now a portable container.

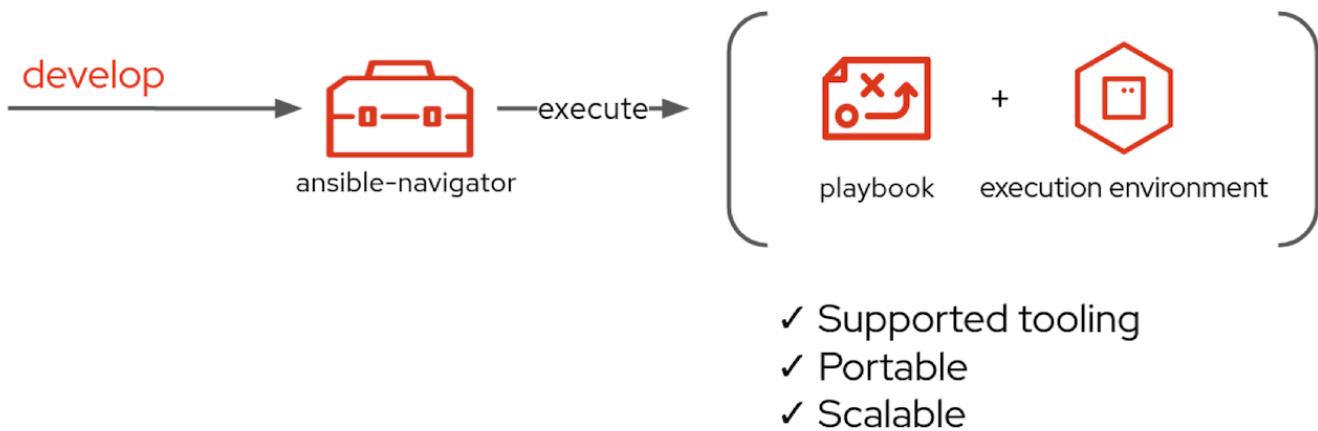


Figure 2. Ansible Content Navigator

Ansible Content Navigator works very similar to **ansible** and **ansible-playbook** commands as it relies on a configuration file. The **ansible-navigator.yml** file utilizes the **ansible.cfg** file and further provides customizations for configurations on developing, testing, and running playbooks in your

Ansible project. Some of the most critical components of Ansible Navigator are the following:

- **ANSIBLE\_CONFIG**: Specifies Ansible configuration file to use
- **image**: Specifies Ansible Execution Environment (EE) to be used

*Listing 11. ansible-navigator.yml*

```
---
```

```
ansible-navigator:
  execution-environment: ①
    enabled: true
  environment-variables:
    set:
      ANSIBLE_CONFIG: ansible.cfg ②
  image: hub.lab.example.com/ee-29-rhel8:latest ③
  logging:
    level: critical
  mode: stdout ④
```

① Configures Ansible Navigator to use an Execution Environment (EE)

② Specifies where Ansible Navigator and the Ansible EE will receive Ansible configuration settings

③ Specifies Ansible EE to use for Ansible Navigator

④ Specified Mode, in this case, we are using **STDOUT** so that the output will look like it does with the **ansible-playbook** command.

Most **ansible** commands have a corresponding **ansible-navigator** command. The table below provides a mapping for the most commonly used commands.

*Table 3. ansible to ansible-navigator Command Comparisons*

Ansible Command	Automation Content Navigator Subcommands
ansible-config	ansible-navigator config
ansible-doc	ansible-navigator doc
ansible-inventory	ansible-navigator inventory
ansible-playbook	ansible-navigator run

### 3.1.2. Ansible Execution Environments

Ansible Execution Environments (EEs) are containers which consist of:

- **Ansible Core**
- **Ansible Content Collections**
- **Python Libraries**

- **Executables**
- Other dependencies for running the playbook

Custom execution environments can be built and created with **ansible-builder**. The process of creating a new Ansible EE depends on if there is a valid Ansible Automation Entitlement associated with your account. If you are using the **free** version of Ansible, base container images **MUST** come from **Quay.IO** as the Red Hat Ansible Execution Environments (EEs) cannot be used or redistributed.

Ansible Execution Environments (EEs) are containers which require:

- **Configuration Files (ansible.cfg)**
- **Inventory**
- **SSH**
  - SSH Keys (requires SSH Agent Service)



Figure 3. Ansible Execution Environment

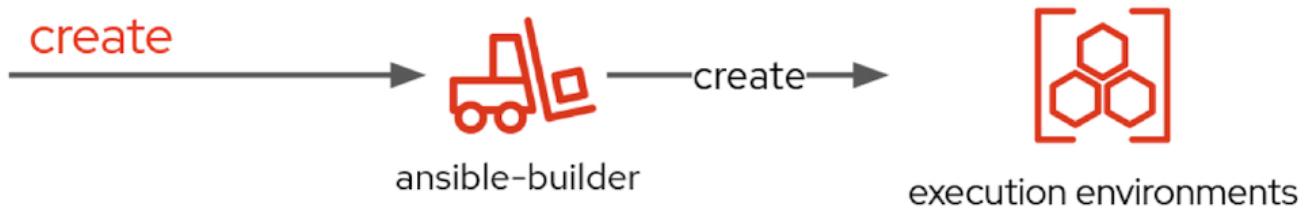


Figure 4. Ansible Builder

#### *Default Versions of Ansible and Execution Environments*

Ansible Automation Platform 2 provides Ansible Core 2.11 as well as several Red Hat Certified Content Collections providing a similar experience to Ansible 2.9. A benefit of execution environments is that they can be used to run older versions of Ansible. Some of the demos in this webinar will use an Ansible EE based on Ansible 2.9 so that collection mapping is done automatically and older playbooks don't need to be touched.



### SSH Keys

SSH Agent is used to allow the SSH keys to pass through to the Ansible Execution Environment (EE) which is a container with all the Ansible and Python components required to run the Ansible Automation tasks.



#### *Listing 12. Passing SSH Keys to Execution Environments*

```
[student@workstation ~]$ eval $(ssh-agent)  
[student@workstation ~]$ ssh-add ~/.ssh/lab_rsa
```

### 3.1.3. DEMO - Using Ansible Content Navigator and Ansible Execution Environments

#### *Example 7. DEMO - Using Ansible Content Navigator and the Ansible 2.9 Execution Environment*

1. Change to correct directory

```
[student@workstation ~]$ cd /home/student/Github/AAP_Webinar/Future/Navigator
```

2. View configuration for Ansible Content Navigator

```
[student@workstation Navigator]$ cat ansible-navigator.yml  
---  
ansible-navigator:  
  execution-environment:  
    enabled: true  
    environment-variables:  
      set:  
        ANSIBLE_CONFIG: ansible.cfg ①  
        image: hub.lab.example.com/ee-29-rhel8:latest ②  
    logging:  
      level: critical  
      mode: stdout ③
```

① Ensure it is pointing to the project **ansible.cfg**

② Ensure we are using the Ansible EE based on Ansible version 2.9.x for compatibility of existing playbooks (pre-collections).

③ Ensure mode is specified as **stdout** so that the output can easily be viewed from the command-line (CLI).

3. View the playbook

```
[student@workstation Navigator]$ cat Website_Ansible_Past.yml
---
- name: Playbook to Fully Setup and Configure a Webserver
  hosts: servere
  tasks:
    - name: Install Packages for Webserver
      yum:
        name:
          - httpd
          - firewalld
        state: latest

    - name: Create Content for Webserver
      copy:
        content: "I'm an awesome webserver for the NYPD and I know Castle!! \n"
        dest: /var/www/html/index.html

    - name: Firewall is Enabled
      service:
        name: firewalld
        state: started
        enabled: true

    - name: HTTP Service is Open on Firewall
      firewalld:
        service: http
        state: enabled
        permanent: true
        immediate: yes

    - name: httpd is started
      systemd:
        name: httpd
        state: started
        enabled: true
```

#### 4. Setup and ensure SSH keys are shared for the Ansible EE

```
[student@workstation ~]$ eval $(ssh-agent) ①
[student@workstation ~]$ ssh-add ~/.ssh/lab_rsa ②
```

① Starts SSH Agent service

② Loads SSH key to the SSH Agent Service keyring

## 5. Run the playbook with **ansible-navigator run** Command

```
[student@workstation Navigator]$ ansible-navigator run Website_Ansible_Past.yml -b  
①  
-----  
Execution environment image and pull policy overview  
-----  
Execution environment image name: hub.lab.example.com/ee-29-rhel8:latest  
... OUTPUT OMITTED ...  
  
TASK [httpd is started] ****  
changed: [servere]  
  
PLAY RECAP ****  
servere : ok=6    changed=4    unreachable=0    failed=0  
skipped=0   rescued=0   ignored=0
```

- ① Specify the **-b** to enable privilege escalation as the **ansible.cfg** and playbook doesn't have escalation already enabled.

## 6. Verify website is running

```
[student@workstation Navigator]$ curl servere  
I'm an awesome webserver for the NYPD and I know Castle!!
```

### Example 8. DEMO - Using Ansible Content Navigator - Interactively

#### 1. Explore Ansible Navigator

```
[student@workstation Navigator]$ ansible-navigator -m interactive  
  
0 | ## Welcome  
|  
1 | -----  
---|  
2 |  
|  
3 | Some things you can try from here:  
|  
4 | - `:collections`          Explore available  
  collec|  
5 | - `:config`               Explore the current  
  ansi|
```

```

6 | - `:doc <plugin>`                    Review documentation
for
7 | - `:help`                                Show the main help
page
8 | - `:images`                             Explore execution
enviro

... OUTPUT OMITTED ...

```

## 2. View information on Execution Environment (type :images)

NAME	TAG	EXECUTION ENVIRONMENT	CREATED	SIZE
0   ee-29-rhel8 (primary) MB	latest	True	2 months ago	785
1   ee-supported-rhel8 GB	2.0	True	2 months ago	1.07
2   flamel GB	latest	False	5 weeks ago	1.56

## 3. View the ee-29-rhel8 EE (as this is the default defined in the configuration file) by typing 0

EE-29-RHEL8:LATEST (PRIMARY)	DESCRIPTION
0   Image information	Information collected from image inspection
1   General information	OS and python version information
2   Ansible version and collections	Information about ansible and ansible collections
3   Python packages	Information about python and python packages
4   Operating system packages	Information about operating system packages
5   Everything	All image informatio

### *Interactively Viewing Execution Environment Details*



Once you've loaded Ansible Content Navigator and the EE, it's possible to view the details of the Ansible versions and collections and any other information about the EE by pressing the corresponding number. To exit the **ansible-navigator** screens, just continue hitting the **ESC** key to exit to the various levels.

## 4. Run **ansible-navigator** with the **-m interactive** to override the **STDOUT** setting and look at Navigator interactively

```
[student@workstation Navigator]$ ansible-navigator run Website_Ansible_Past.yml -b
-m interactive
```

	PLAY NAME	OK	CHANGED	UNREACHABLE	FAILED	SKIPPED	IGNORED	IN PROGRESS	TASK COUNT
0   Playbook t	2	0	0	0	0	0	1	3	0000

5. Hit the **0** to view playbook output for **Play 0**

RESULT	HOST	NUMBER	CHANGED	TASK	TASK ACTION
DURATION					
0   OK	servere	0	False	Gathering Facts	gather_facts
1s					
1   OK	servere	1	False	Install Packages for Webserver yum	
1s					
2   OK	servere	2	False	Create Content for Webserver copy	
0s					
3   OK	servere	3	False	Firewall is Enabled service	
0s					
4   OK	servere	4	False	HTTP Service is Open on Firewall firewalld	
0s					
5   OK	servere	5	False	httpd is started systemd	
0s					

6. Hit 5 to examine **Task 5** from the playbook

```

PLAY [Playbook to Fully Setup and Configure a Webserver:5]
*****
TASK [httpd is started]
*****
OK: [servere]
 0 | ---
 0
 1 | duration: 0.521409
 2 | end: '2022-01-24T19:24:05.870450'
 3 | event_loop: null
 4 | host: serverd
 5 | play: Playbook to Fully Setup and Configure a Webserver
 6 | play_pattern: serverd
 7 | playbook:
/home/student/Github/AAP_Webinar/Future/Navigator/Website_Ansible_Past.y
 8 | remote_addr: serverd
 9 | res:
10 | _ansible_no_log: false

... output omitted ...

```

### *Interactive Mode Details*



When using Ansible Navigator in interactive mode it is possible to get a lot more details regarding each task in the play as well as details on the modules being used and other system settings/configurations.

## 3.2. Introduction to AAP 2.x - Ansible Automation Hub

The Red Hat AAP 2.x platform provides Ansible Private Automation hub for those with valid subscription entitlements. Private Automation Hub provides two key features:

- Place to publish Collections
- Place to publish Execution Environments

This Private Automation Hub is essentially your local version (private version) of Ansible Galaxy and container registries.

### 3.2.1. Private Automation Hub

Private automation hub provides an on-site location for publishing collections and Ansible Execution Environments. This provides a centralized location for Automation Controller and for developers using Ansible Navigator to obtain EEs and content collections.

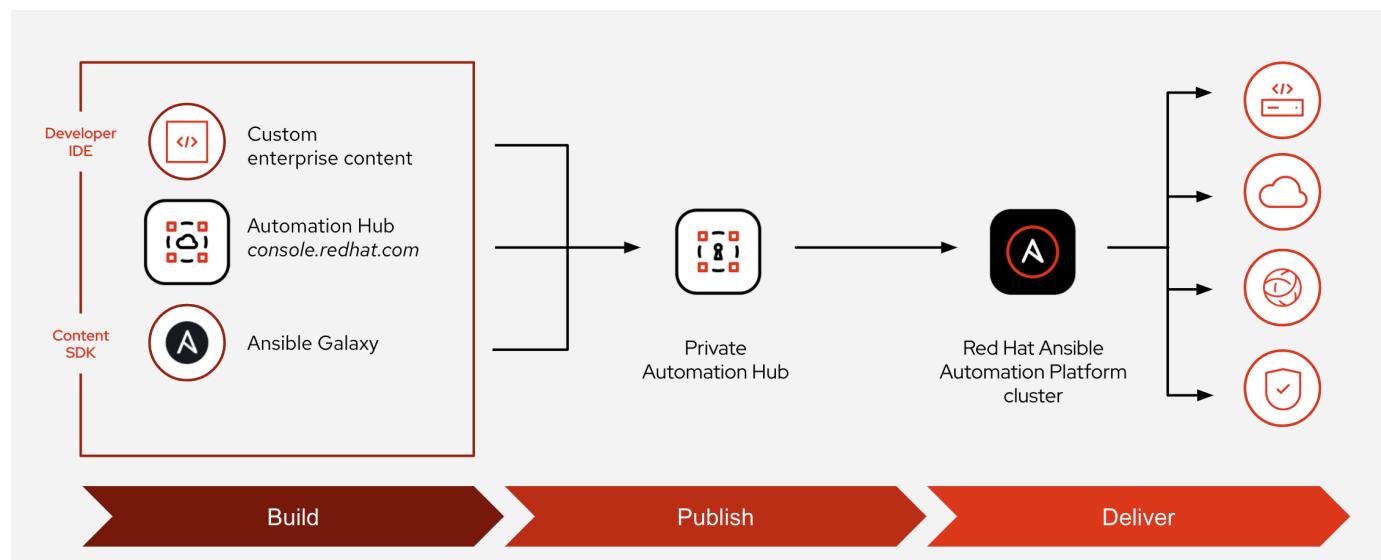


Figure 5. Ansible Private Automation Hub

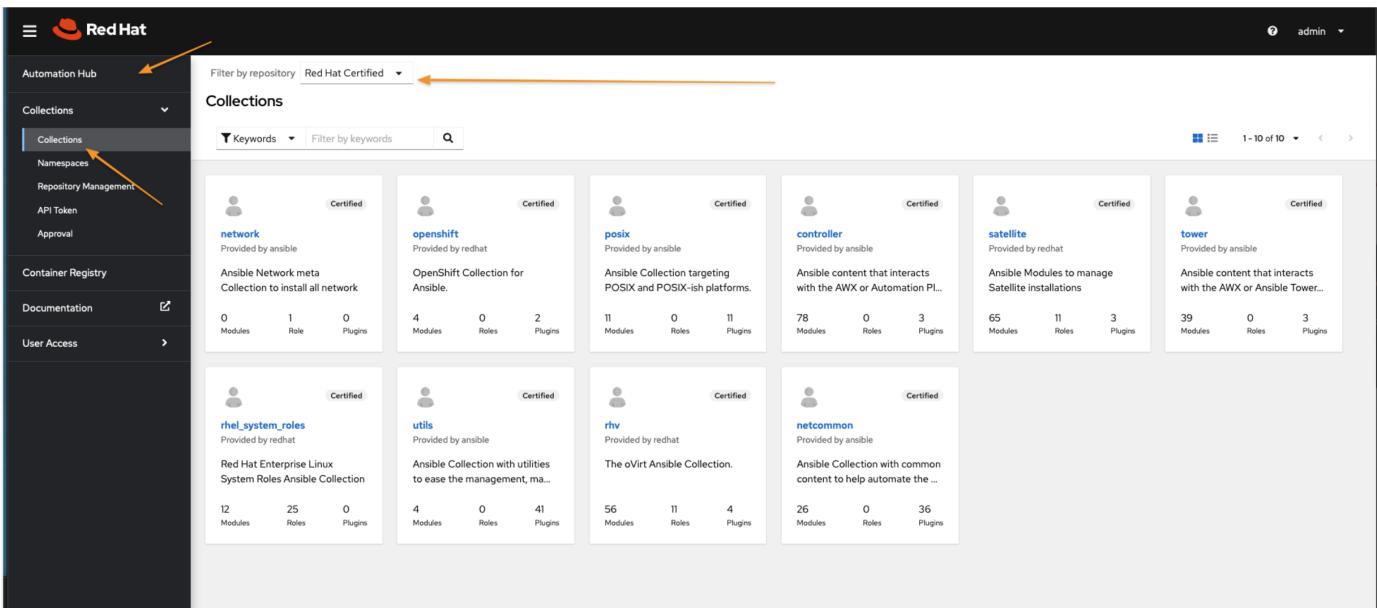


Figure 6. Ansible Private Automation Hub - Collections

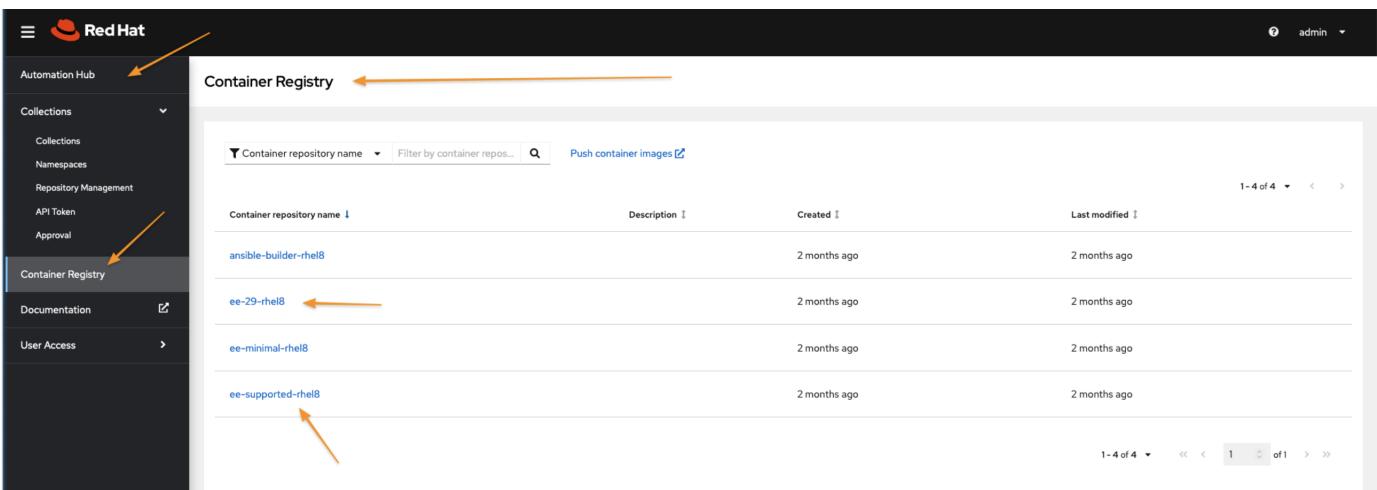


Figure 7. Ansible Private Automation Hub - Container Registry (Execution Environments)

### 3.2.2. Custom Execution Environments

Building EEs is rather simple: we provide the tool execution environment builder, the CLI tool `ansible-builder` for this, which knows how to bring the pieces together in just the right way and create the EE. They can be published on Private Automation hub or in any container registry.



Figure 8. Ansible Execution Environments

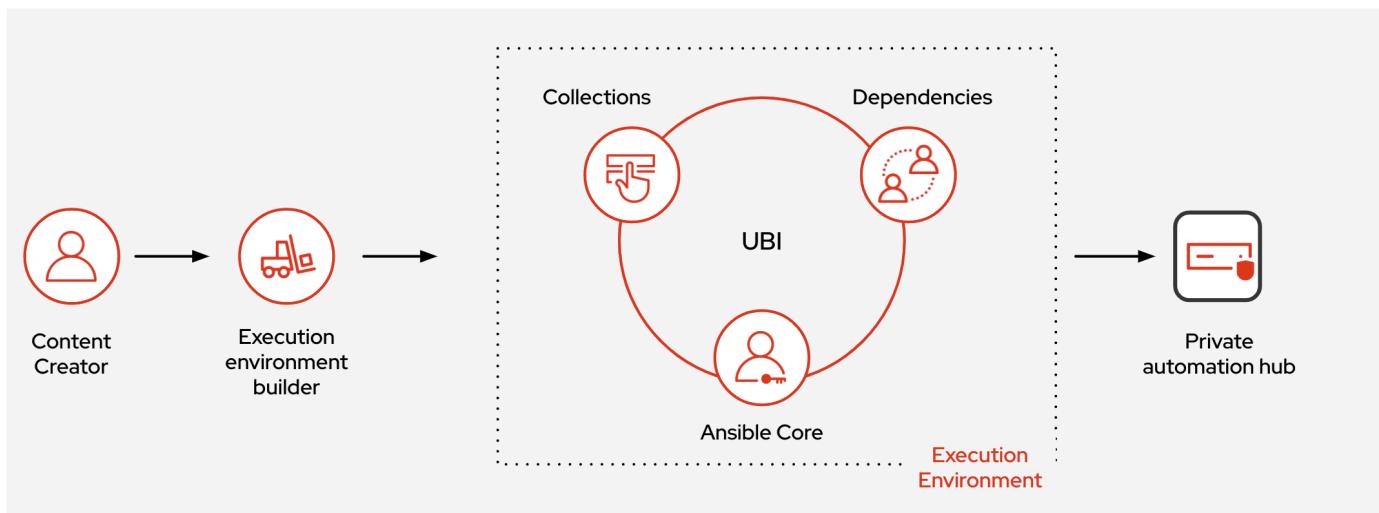


Figure 9. Ansible Execution Environments - Build Process



Figure 10. Ansible Execution Environments - ansible-builder

Listing 13. `execution-environment.yml` File to Define Custom EE

```
---
version: 1
build_arg_defaults:
  EE_BASE_IMAGE: 'hub.lab.example.com/ee-minimal-rhel8:2.0' ①
  EE_BUILDER_IMAGE: 'hub.lab.example.com/ansible-builder-rhel8:2.0' ②
dependencies:
  galaxy: requirements.yml ③
```

```
python: requirements.txt ④  
system: bindep.txt ⑤
```

- ① Defines base container image to be used for creating the EE
- ② Defines the builder image to be used for the EE
- ③ Points to file containing the Collections and Roles to be installed and included in the EE
- ④ Points to file containing the required Python components to be installed and included in the EE
- ⑤ Points to file containing the system applications to be installed in the EE

*Listing 14. requirements.yml*

```
---
```

```
collections:  
  - name: /build/exercise.motd.tar.gz ①  
    type: file ②
```

- ① Defines collections or roles to be included in the container image.
- ② Defines type of collection or role.

*Listing 15. requirements.txt*

```
# Python dependencies  
funmotd ①
```

- ① Defines Python dependencies needed to be installed on the container image. If multiple Python packages are needed, one package per line is required

*Listing 16. bindep.txt*

```
# System-level dependencies  
hostname ①
```

- ① Defines system packages needed to be installed on the container image. If multiple packages are needed, one package per line is required.

### References

- **Ansible Builder Blog:**
  - <https://www.ansible.com/blog/introduction-to-ansible-builder>
- **Ansible Builder Images**
  - <https://blog.networktocode.com/post/ansible-builder-runner-ee/>
  - <https://quay.io/repository/ansible/ansible-runner?tag=latest&tab=tags>



### 3.2.3. DEMO - Creating a Custom Execution Environment and Publishing to Private Automation Hub

Example 9. DEMO - Creating a Custom Execution Environment

1. Switch to correct directory

```
[student@workstation ~]$ cd /home/student/Github/AAP_Webinar/Future/Custom_EE/EE
```

2. Install **ansible-builder**

```
[student@workstation EE]$ sudo yum install -y ansible-builder
```



The **execution-environment.yml**, **requirements.yml**, **requirements.txt**, and **bindep.txt** have already been created to save time. Normally these files would be created by the administrator creating the custom execution environment.

3. View or create the **execution-environment.yml** file

```
[student@workstation EE]$ cat execution-environment.yml
---
version: 1
build_arg_defaults:
  EE_BASE_IMAGE: 'hub.lab.example.com/ee-minimal-rhel8:2.0'
  EE_BUILDER_IMAGE: 'hub.lab.example.com/ansible-builder-rhel8:2.0'
dependencies:
  galaxy: requirements.yml
  python: requirements.txt
  system: bindep.txt
```

4. View or create the **requirements.yml** File

```
[student@workstation EE]$ cat requirements.yml
---
collections:
  - name: /build/exercise.motd.tar.gz
    type: file
```

5. View or create the **requirements.txt** File

```
[student@workstation EE]$ cat requirements.txt
# Python dependencies
funmota
```

## 6. View or create the **bindep.txt** File

```
[student@workstation EE]$ cat bindep.txt
# System-level dependencies
hostname
```

## 7. Run the **ansible-builder create** command to create the structure for the build process

```
[student@workstation EE]$ ansible-builder create
Complete! The build context can be found at:
/home/student/Github/AAP_Webinar/Future/Custom_EE/EE/context
```



This is necessary so that we can copy the file **exercise.motd.tar.gz** to the correct directory so it will be mounted and available in the container image and build process.

## 8. View created files and directory structure

```
[student@workstation EE]$ tree context/
context/
├── _build
│   ├── bindep.txt
│   ├── requirements.txt
│   └── requirements.yml
└── Containerfile

1 directory, 4 files
```

## 9. Copy the **exercise.motd.tar.gz** to the **context/\_build** location so it can be mounted properly

```
[student@workstation EE]$ cp collection-files/exercise.motd.tar.gz context/_build/
```

## 10. Create the **ee-motd-demo** Execution Environment Image

```
[student@workstation EE]$ ansible-builder build -t ee_aap_demo:latest
Running command:
```

```
podman build -f context/Containerfile -t ee_aap_demo:latest context
Complete! The build context can be found at:
/home/student/Github/AAP_Webinar/Future/Custom_EE/EE/context
```

11. Verify container image was built using the **podman images** Command

```
[student@workstation EE]$ podman images
REPOSITORY                                TAG      IMAGE ID      CREATED
SIZE
localhost/ee_aap_demo                      latest   3bfe381575fa  6 minutes ago
419 MB
```

#### **Example 10. DEMO - Using and Testing a Custom Execution Environment**

1. Change to correct directory

```
[student@workstation ~]$ cd /home/student/Github/AAP_Webinar/Future/Custom_EE
```

2. Ensure that **ansible-navigator** is using the correct Ansible Execution Environment

```
[student@workstation Custom_EE]$ cat ansible-navigator.yml
---
ansible-navigator:
  execution-environment:
    enabled: true
    environment-variables:
      set:
        ANSIBLE_CONFIG: ansible.cfg
    image: localhost/ee_aap_demo:latest
  logging:
    level: critical
  mode: stdout
```

3. Create or View Playbook

```
[student@workstation Custom_EE]$ cat Custom_EE_Playbook.yml
---
- name: Playbook to Configure the Message of the Day with a Custom EE
  hosts: servers
  collections:
    - exercise.motd
  roles:
```

```
- name: exercise.motd.banner
```

#### 4. Execute Playbook

```
[student@workstation Custom_EE]$ ansible-navigator run Custom_EE_Playbook.yml -b  
①
```

① The **-b** is placed on there to elevate privileges.

#### 5. Test to see if the MOTD was deployed to the server

```
[student@workstation EE]$ ssh server  
Activate the web console with: systemctl enable --now cockpit.socket  
  
This system is not registered to Red Hat Insights. See https://cloud.redhat.com/  
To register this system, run: insights-client --register  
  
=====  
=====  
==  
== This system is managed by Ansible.  
== AAP2.0  
==  
=====  
=====  
  
Last login: Tue Jan 25 16:17:04 2022 from 172.25.250.9
```

### Example 11. DEMO - Publishing a Custom Execution Environment

#### 1. Change to correct directory

```
[student@workstation ~]$ cd /home/student/Github/AAP_Webinar/Future/Custom_EE
```



Not really needed as we are using the PODMAN commands to push images, but done for consistency.

#### 2. Tag the image with the Podman command to prepare the push to **hub.lab.example.com**

```
[student@workstation EE]$ podman tag localhost/ee_aap_demo:latest  
hub.lab.example.com/aap-demo:latest
```

### 3. Push the image to private automation hub

```
[student@workstation EE]$ podman push hub.lab.example.com/aap-demo:latest
Getting image source signatures
Copying blob 38345e1102be done
Copying blob df2b2b67ec7f done
Copying blob fa751636af06 done
Copying blob a65a1b01a4d2 done
Copying blob af092941766c done
Copying blob efebe3fe0d93 done
Copying blob 9c99e40eeecd0 done
Copying config 3bfe381575 done
Writing manifest to image destination
Storing signatures
```

#### *Repository Login*

It might be necessary to perform a **podman login** for the remote container registry.



#### *Listing 17. Registry Login*

```
podman login hub.lab.example.com
```

### 4. Login to the Private Automation hub

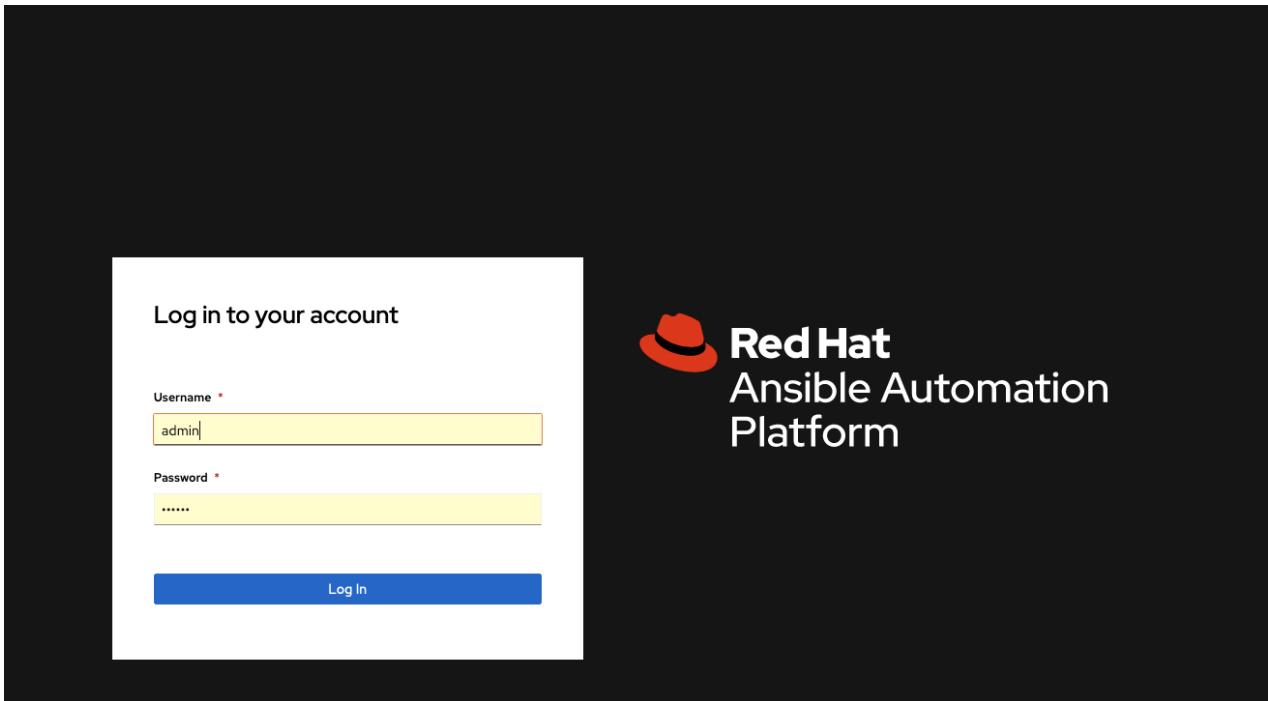


Figure 11. Automation Hub Login

5. Navigate to Container Registry and look for the **aap-demo** Container

Container repository name	Description	Created	Last modified
aap-demo		3 minutes ago	3 minutes ago
ansible-builder-rhel8		2 months ago	2 months ago
ee-29-rhel8		2 months ago	2 months ago
ee-minimal-rhel8		2 months ago	2 months ago
ee-supported-rhel8		2 months ago	2 months ago

Figure 12. Automation Hub Login

6. Clean up **servere** and show that MOTD has been reset

```
[student@workstation Custom_EE]$ ansible-playbook MOTD_Cleanup.yml

PLAY [Playbook to Cleanup MOTD for Testing Custom EE]
*****
TASK [Gathering Facts]
*****
*****
ok: [servere]

TASK [Reset MOTD]
*****
*****
changed: [servere]

PLAY RECAP
*****
*****
servere : ok=2    changed=1    unreachable=0    failed=0
skipped=0   rescued=0    ignored=0
```

```
[student@workstation Custom_EE]$ ssh servere
Activate the web console with: systemctl enable --now cockpit.socket
```

This system is not registered to Red Hat Insights. See <https://cloud.redhat.com/>  
To register this system, run: insights-client --register

I'm the clean MOTD Banner

## 7. Test with the image coming from Private Automation Hub

```
[student@workstation Custom_EE]$ ansible-navigator run --pp always --eei  
hub.lab.example.com/aap-demo:latest -m stdout Custom_EE_Playbook.yml -b  
  
-----  
Execution environment image and pull policy overview  
-----  
Execution environment image name: hub.lab.example.com/aap-demo:latest  
Execution environment image tag: latest  
Execution environment pull policy: always  
Execution environment pull needed: True  
  
-----  
Updating the execution environment  
-----  
Trying to pull hub.lab.example.com/aap-demo:latest...  
Getting image source signatures  
Copying blob 3c9fdæ16a64 skipped: already exists  
Copying blob 5c4402ce71c4 skipped: already exists  
Copying blob 69ebc448681d [-----] 0.0b / 0.0b  
Copying blob 495ff1ef2828 [-----] 0.0b / 0.0b  
Copying blob 80be453030cf [-----] 0.0b / 0.0b  
Copying blob 642d458785a1 [-----] 0.0b / 0.0b  
Copying blob 00fe5380b165 [-----] 0.0b / 40.3MiB  
Copying config 3bfe381575 done  
Writing manifest to image destination  
Storing signatures  
3bfe381575fa7606cb745bfe8227d0cbf59b4d91dc7bd7d811a1fcfe28022919  
  
[student@workstation Custom_EE]$
```

## 3.3. Introduction to AAP 2.x - Ansible Controller (formerly Ansible Tower)

Ansible Controller is the replacement for Ansible Tower which consists of a web-based management interface providing multiple features for system automation. Ansible Controller provides the following:

- WebUI
- Centralized Dashboard

- Centralized Logging/Auditing
- Role-Based Access Control (RBAC)
  - User/Group (Team) Management
- Inventory Management
- Project Management (Integration with Version Control)
- Credential Management
- Workflow and Job Management
- ... and more

### 3.3.1. Organizations, Teams, and RBAC

#### 3.3.1.1. DEMO - Creating Organizations, Teams, and Users

*Example 12. DEMONSTRATION - Creating an Organization with Users and Teams*

*Creating an Organization*

1. Login to Ansible Controller

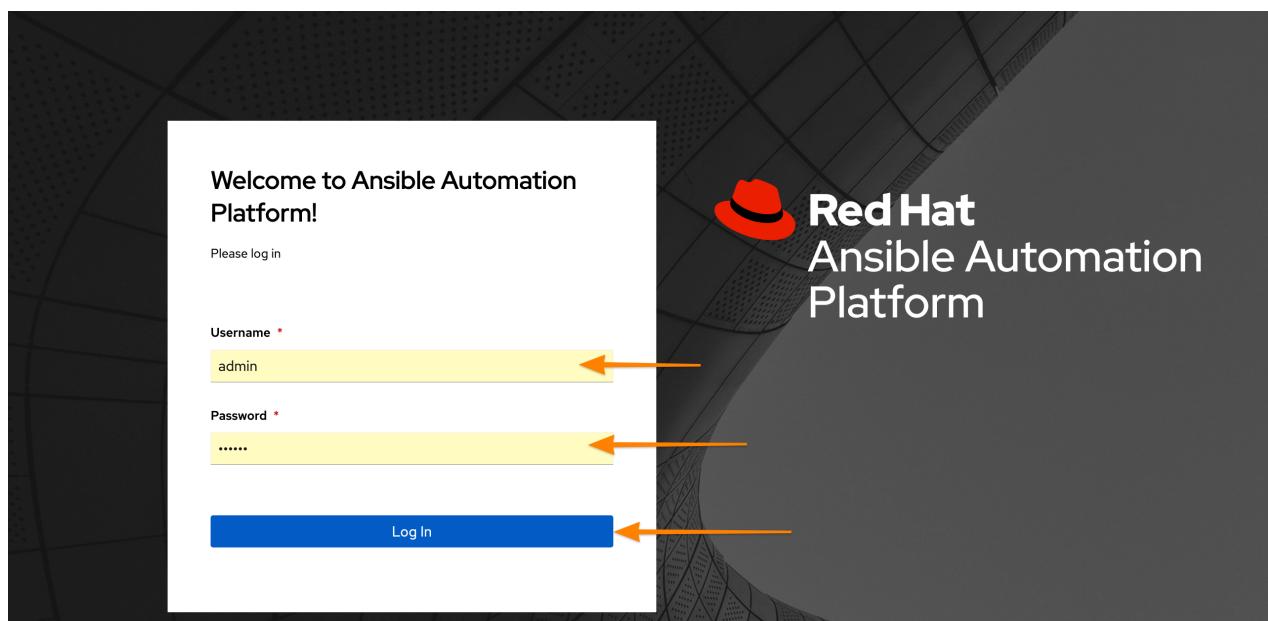


Figure 13. Ansible Controller Login

2. Click Organizations

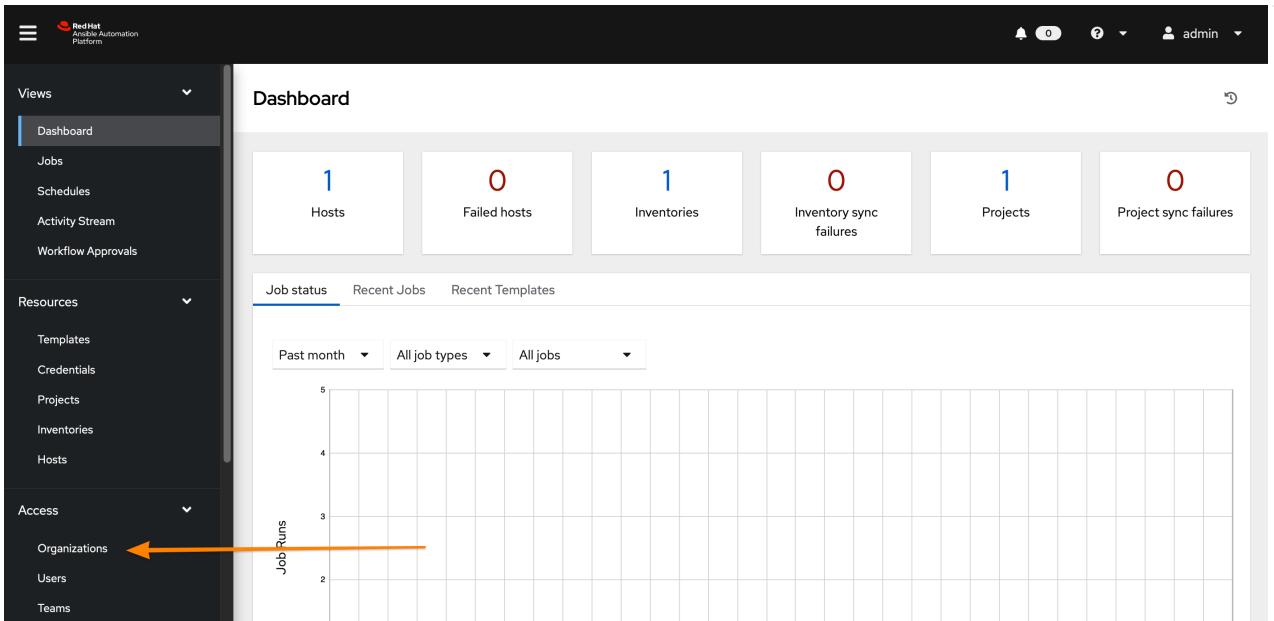


Figure 14. Ansible Controller - Organizations

### 3. Click Add

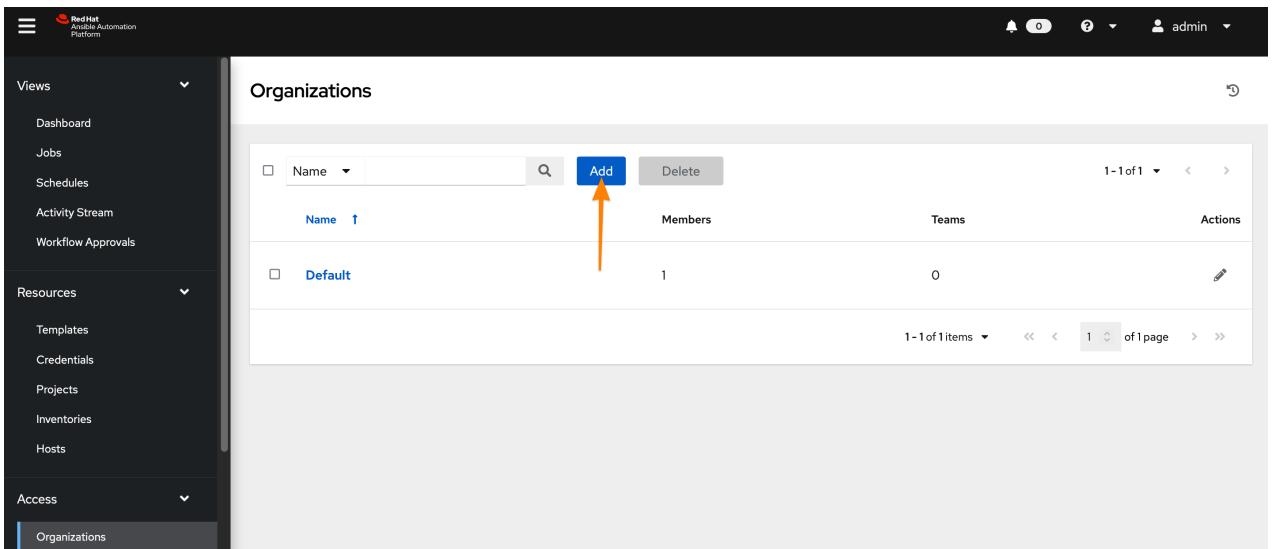


Figure 15. Ansible Controller - Adding an Organization

### 4. Create the new organization with the appropriate details and then click Save

- Name:** Required for name of Organization
  - NYPD**
- Description:** *Optional but helpful*
  - New York Police Department IT**
- Execution Environment:** Default execution environment for playbooks, projects, and workflows in the environment.
  - Ansible Engine 2.9 execution environment (Selected for backwards compatibility)**

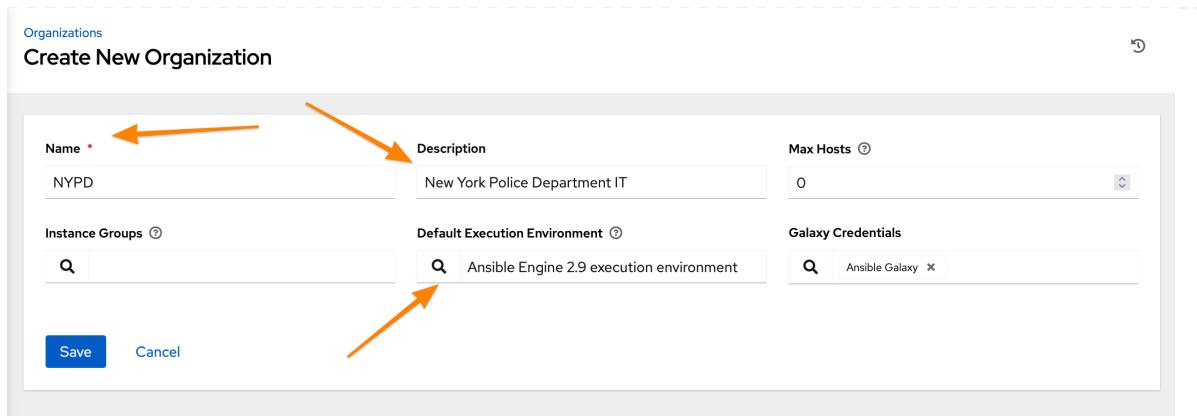


Figure 16. Ansible Controller - Configuring the Organization



#### *Default Execution Environment*

It is helpful to setup the default execution environment (EE) which will control the running of Ansible playbooks within your Organization. It is possible for individual projects and playbooks to be selectively run with another execution environment, but the default EE will be used if another EE isn't specified.

In the above example, the **Ansible Engine 2.9 execution environment** was selected as it has the best compatibility with older playbooks before the realigned modules and collections. Ansible Engine 2.9 can utilize collections, but also has the mapping for the Ansible modules allowing older playbooks to run without updating to using FQCN.

#### *Creating a Team*

1. Click on Teams

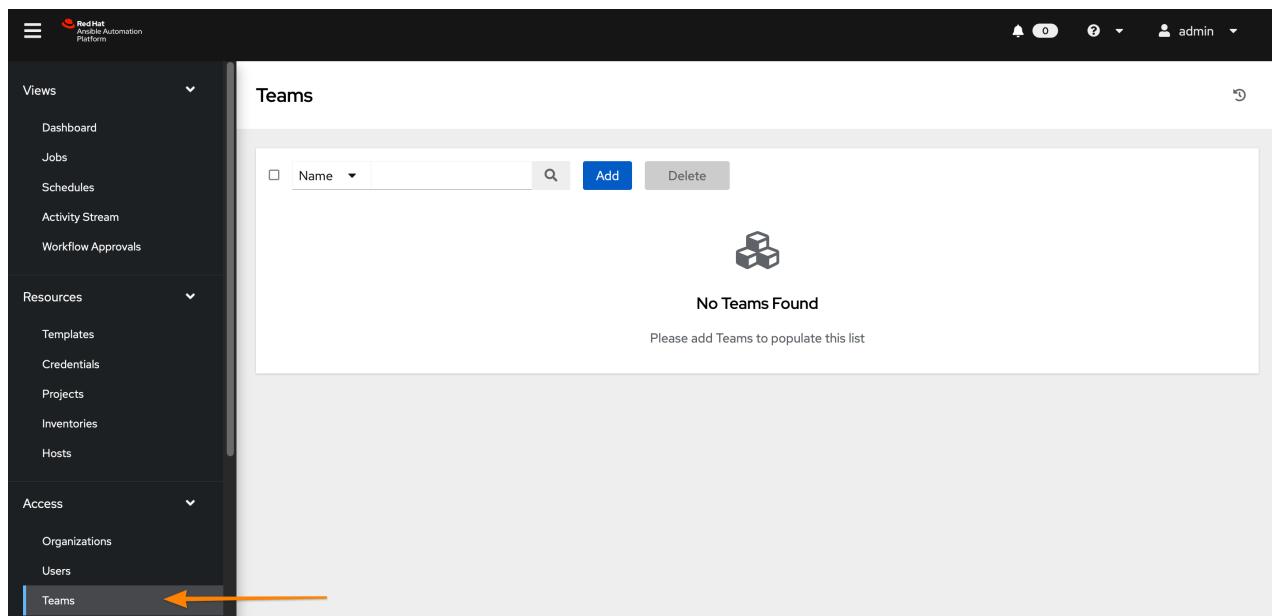


Figure 17. Ansible Controller - Creating a Team

2. Click **Add** and fill in appropriate values and then click **Save**
  - Name:** Required for name of team
    - NYPD System Administrator**
  - Organization:** Required to select an existing organization from drop-down
    - Click magnifying glass and select Organization
      - NYPD**

Teams  
Create New Team

Name \* NYPD System Administrator

Description

Organization \*

Save Cancel

Figure 18. Ansible Controller - Configuring a Team

Select Organization

Selected NYPD

Name ▾ Q

Name ↑

Default

NYPD

1 of 1 page < < > >>

Select Cancel

Figure 19. Ansible Controller - Selecting an Organization

Teams  
Create New Team

Name \* NYPD System Administrator

Description

Organization \*

NYPD

Save Cancel

Figure 20. Ansible Controller - Completing Team Creation

## Creating Users

## 1. Click Users

The screenshot shows the Red Hat Ansible Automation Platform interface. On the left, a dark sidebar menu includes sections for Views (Dashboard, Jobs, Schedules, Activity Stream, Workflow Approvals), Resources (Templates, Credentials, Projects, Inventories, Hosts), and Access (Organizations). The 'Access / Organizations' section is expanded, and 'Users' is highlighted with an orange arrow. The main content area is titled 'Users' and displays a table with two rows. The first row has a checkbox next to 'Username' (admin), followed by columns for First Name (Travis) and Last Name (Michette), and roles 'System Administrator' and 'Normal User'. The second row has a checkbox next to 'travis'. At the top of the table are buttons for 'Add' and 'Delete'. Below the table are pagination controls showing '1-2 of 2 items' and '1 of 1 page'.

Figure 21. Ansible Controller - Creating a User

## 2. Click Add and fill in appropriate information and click Save

- Username:** kbeckett
- Email:** [kbeckett@nypd.ny.gov](mailto:kbeckett@nypd.ny.gov)
- Password:** redhat
- First Name:** Kate
- Last Name:** Beckett
- Organization:** NYPD
- User Type:** Normal User

The screenshot shows a modal dialog titled 'Select Organization'. It has a 'Selected' section with 'NYPD' highlighted. Below it is a search bar with 'Name' and a dropdown arrow. A table lists organizations with 'Name' as the header. Two rows are visible: 'Default' (radio button unselected) and 'NYPD' (radio button selected, indicated by an orange arrow). At the bottom are 'Select' and 'Cancel' buttons, with an orange arrow pointing to the 'Select' button.

Figure 22. Ansible Controller - Selecting an Organization

Users

### Create New User

Username *	kbeckett	Email	kbeckett@nypd.ny.gov	Password *	.....
Confirm Password *	.....	First Name	Kate	Last Name	Beckett
Organization *	NYPD	User Type *	Normal User		
<input type="button" value="Save"/> <input type="button" value="Cancel"/>					

Figure 23. Ansible Controller - Configuring a User

#### *Adding a User to a Team*

Adding users to a team can be done multiple ways, but in this example, we will be modifying a recently created user and use the **Teams** option on the User **Details** tab.

#### 1. Click on Teams

Users > kbeckett

### Teams

<input type="button" value="Back to Users"/>	<input type="button" value="Details"/>	<input type="button" value="Organizations"/>	<input type="button" value="Teams"/>	<input type="button" value="Roles"/>
<input type="checkbox"/> <input type="text" value="Name"/> <input type="button" value="Search"/> <input type="button" value="Associate"/> <input type="button" value="Disassociate"/>				
 <b>No Teams Found</b> Please add Teams to populate this list				

Figure 24. Ansible Controller - User Teams Menu

#### 2. Click the **Associate** button to search for and select a team, then click **Save**

##### a. NYPD System Administrator



Figure 25. Ansible Controller - User Team(s) Selection

- Verify user was associated with the correct team(s).

The screenshot shows a "Teams" page for a user named "kbeckett". The top navigation bar includes "Users > kbeckett", "Details", "Organizations", "Teams" (which is underlined), and "Roles". Below the navigation is a search bar with "Name" and a magnifying glass icon, followed by "Associate" and "Disassociate" buttons. A "Clear all filters" link is also present. The main content is a table with columns "Name" (sorted up), "Organization", and "Description". One row is shown: "NYPD System Administrator" is associated with "NYPD". At the bottom of the table are navigation arrows and a page indicator "1-1 of 1 items".

Figure 26. Ansible Controller - User Team(s) Verification

### Teams



Teams are used to group users together so that RBAC controls can be more easily managed at a group level versus an individual user level. It is still possible to give individual users additional privileges, but teams is the preferred way of permission management.

## 3.3.2. Inventories and Credentials

### 3.3.2.1. DEMO - Creating Inventories and Credentials

*Example 13. DEMONSTRATION - Creating an Inventories and Credentials*

*Creating an Inventory*

## 1. Login to Ansible Controller

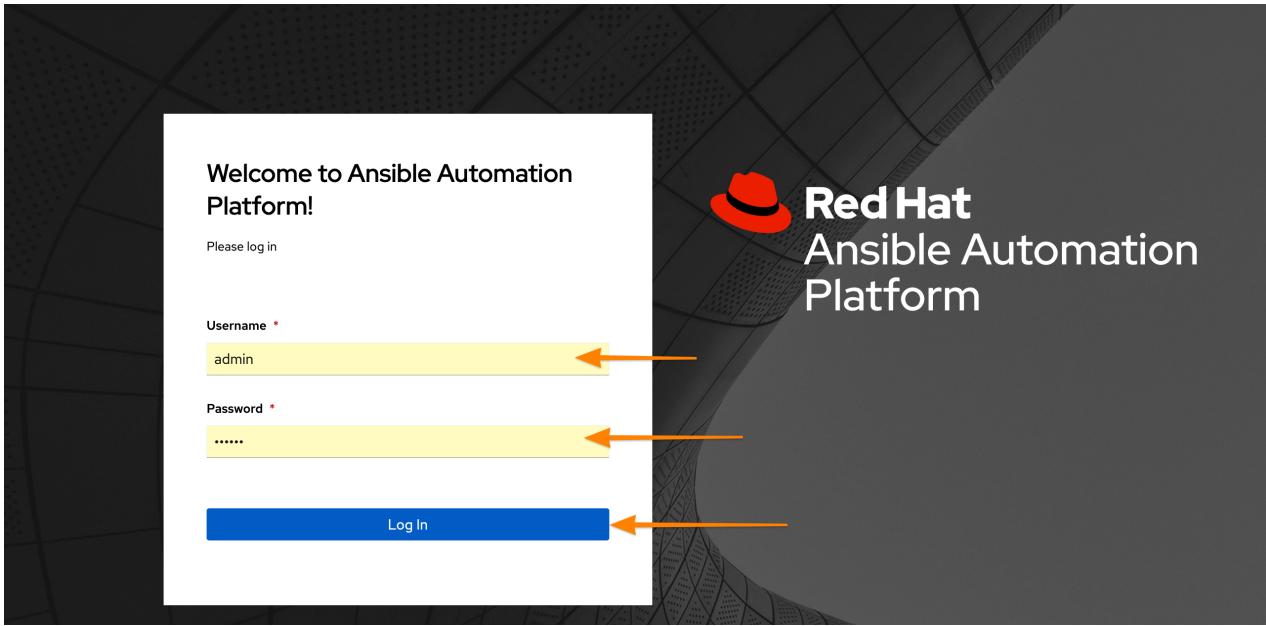


Figure 27. Ansible Controller Login

## 2. Click Inventories and then click Add to Add an Inventory

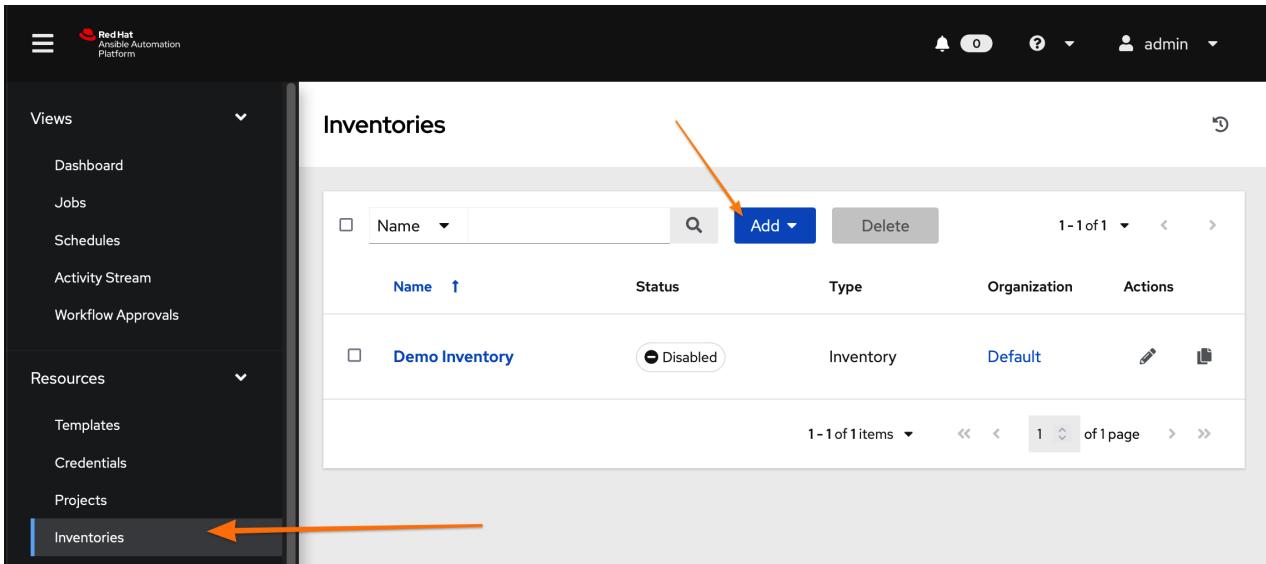


Figure 28. Ansible Controller - Inventory

## 3. Provide and inventory **Name** and **Organization** and then click **Save**

- Name:** NYPD Systems
- Organization:** NYPD

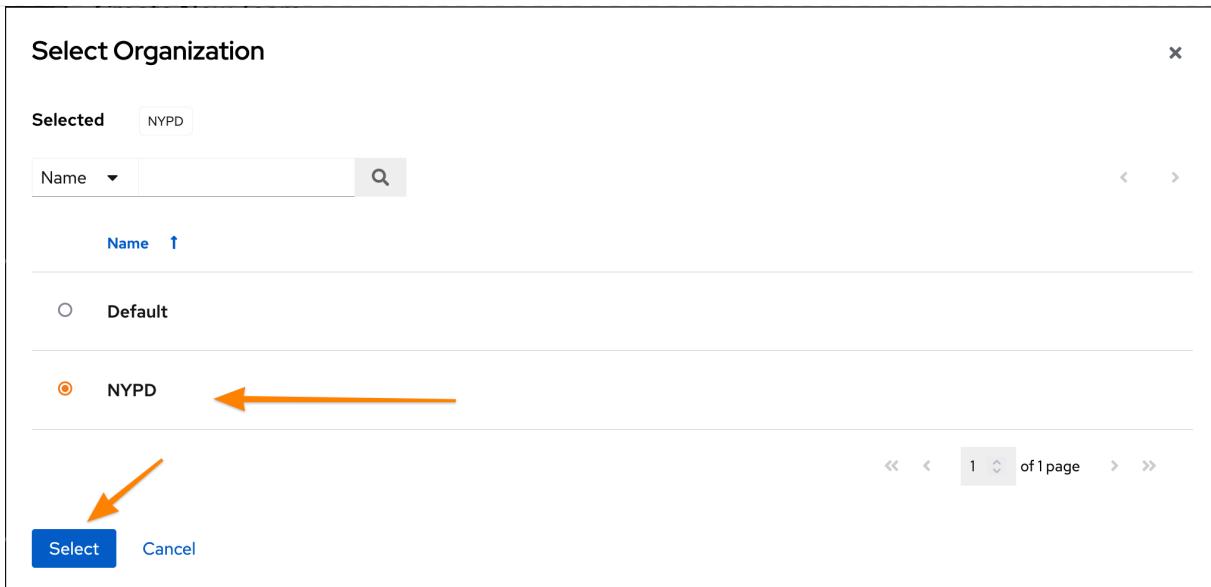


Figure 29. Ansible Controller - Selecting an Organization

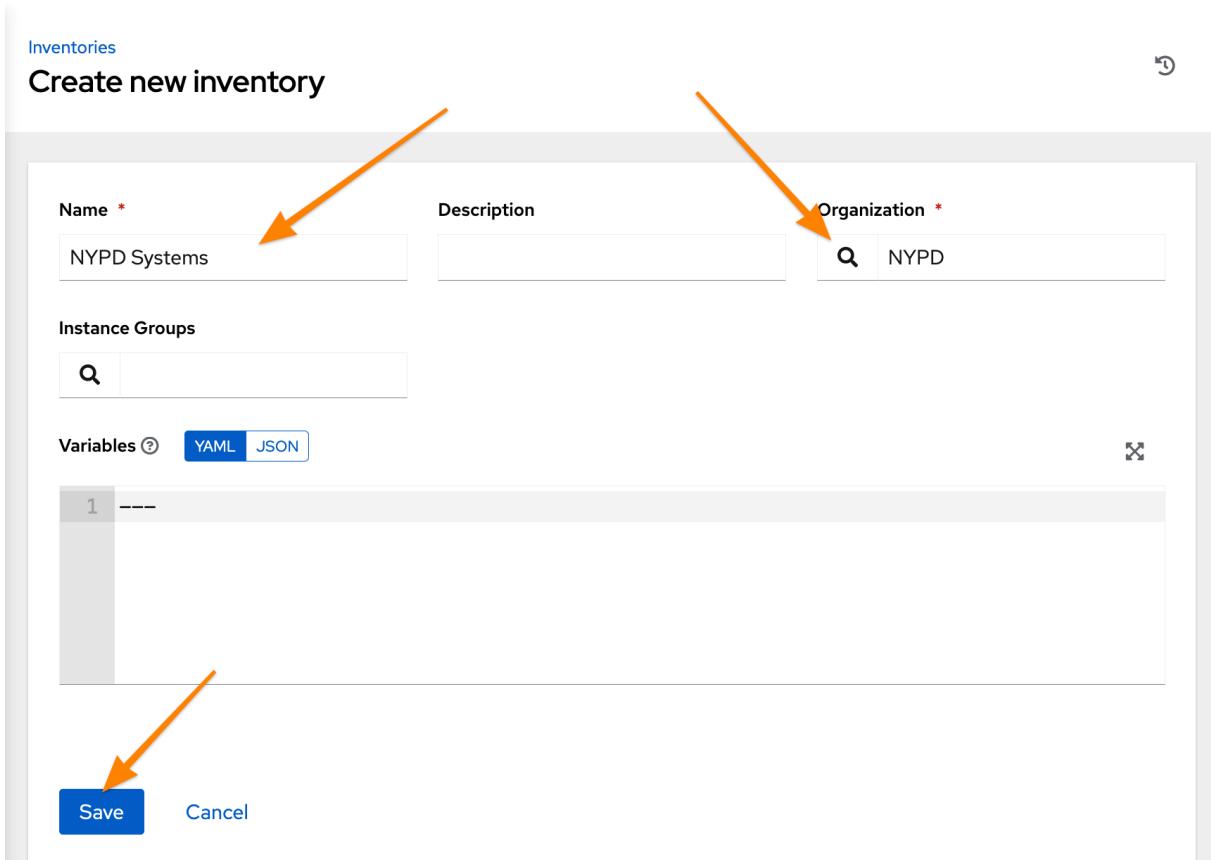


Figure 30. Ansible Controller - New Inventory

4. Add hosts to the inventory by clicking **Hosts** and then click **Add**

a. **Name:** serverd

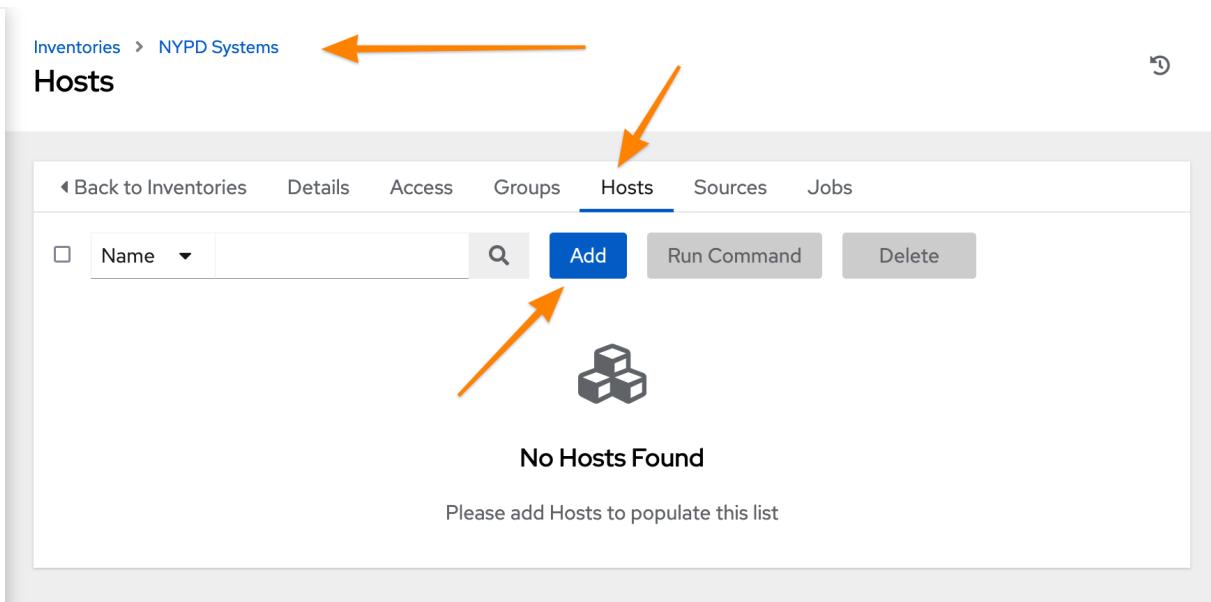


Figure 31. Ansible Controller - Managed Hosts in Inventory

5. Provide the inventory hostname of the host and any host-based variables if desired and click **Save**. Repeat for multiple hosts.

A screenshot of a "Create new host" dialog box. At the top, it shows the path: Inventories > NYPD Systems > Hosts. The title of the dialog is "Create new host". Inside the dialog, there are two input fields: "Name \*" which contains the value "serverd" and "Description" which is empty. Below these fields is a section for "Variables" with tabs for "YAML" (selected) and "JSON". Under the "Variables" tab, there is a code editor area containing the YAML code: "1 ---\n2". At the bottom of the dialog are two buttons: "Save" (highlighted with an orange arrow) and "Cancel".

Figure 32. Ansible Controller - Adding a Host to Inventory

#### Creating Credentials

1. Click **Credentials** and then click **Add** to add a new credential

Name	Type	Actions
Ansible Galaxy	Ansible Galaxy/Automation Hub API Token	
Default Execution Environment Registry Credential	Container Registry	
Demo Credential	Machine	

Figure 33. Ansible Controller - Credentials

2. Create the credential specifying the name, type, and bind to organization if desired and click **Save**.
  - a. **Name:** NYPD Machine SSH Creds
  - b. **Description:** NYPD SSH UN and PW Credential
  - c. **Organization:** NYPD
  - d. **Username:** devops
  - e. **Password:** redhat
  - f. **Privilege Escalation Method:** sudo
  - g. **Privilege Escalation Username:** root
    - NOTE - SSH credentials are **Machine Credentials**

Credentials

### Create New Credential

Name *	Description	Organization
NYPD Machine SSH Creds	NYPD SSH UN and PW Credential	<input type="text"/> NYPD
Credential Type *		
Machine		

Type Details

Username	Password	<input type="checkbox"/> Prompt on launch
devops	<input type="password"/> .....	<input type="checkbox"/>
SSH Private Key		

Figure 34. Ansible Controller - Machine Credentials

SSH Private Key

Drag a file here or browse to upload

Signed SSH Certificate

Drag a file here or browse to upload

Private Key Passphrase	<input type="checkbox"/> Prompt on launch	Privilege Escalation Method <small>?</small>	Privilege Escalation Username
<input type="password"/>	<input type="checkbox"/>	sudo	<input type="text"/> root
Privilege Escalation Password	<input type="checkbox"/> Prompt on launch		
<input type="password"/>	<input type="checkbox"/>		

Figure 35. Ansible Controller - Credentials - Privileged User



Privilege Escalation

It is necessary to provide privilege escalation information as with Ansible Controller, this is where the information and configuration must come from for execution environments (EEs).

### 3.3.3. Projects and Job Templates

#### 3.3.3.1. DEMO - Projects and Job Templates

*Example 14. DEMONSTRATION - Creating an Projects and Job Templates*

*Automatically Setup SCM/Git Credentials before Proceeding*

You must run a playbook to setup the `git` credentials so that there are SCM credentials loaded into Ansible Controller prior to completing this demo. This also assumes that the `ssh_ID_key` has been copied to `/tmp/github_id`. The playbook and names may need to be modified for your user.

1. Go to the **Setup** Resources Directory

```
[student@workstation ~]$ cd  
/home/student/Github/AAP_Webinar/Future/Setup
```

1. Execute the **Setup\_Controller.yml** Playbook to create the Git Credentials



```
[student@workstation Setup]$ ansible-playbook Setup_Controller.yml  
  
PLAY [Playbook to Configure Controller with SCM Credentials]  
*****  
  
TASK [Gathering Facts]  
*****  
ok: [localhost]  
  
TASK [Create a valid SCM credential for Travis's Github]  
*****  
[WARNING]: You are using the awx version of this collection but  
connecting to Red Hat  
Ansible Automation Platform ①  
changed: [localhost]  
  
PLAY RECAP  
*****  
*****  
localhost : ok=2    changed=1    unreachable=0
```

failed=0    skipped=0    rescued=0    ignored=0

- ① A warning is generated as we are using the public AWX collection to manage controller instead of the supported collection from Red Hat Automation Hub

### Creating a Project

#### 1. Login to Ansible Controller

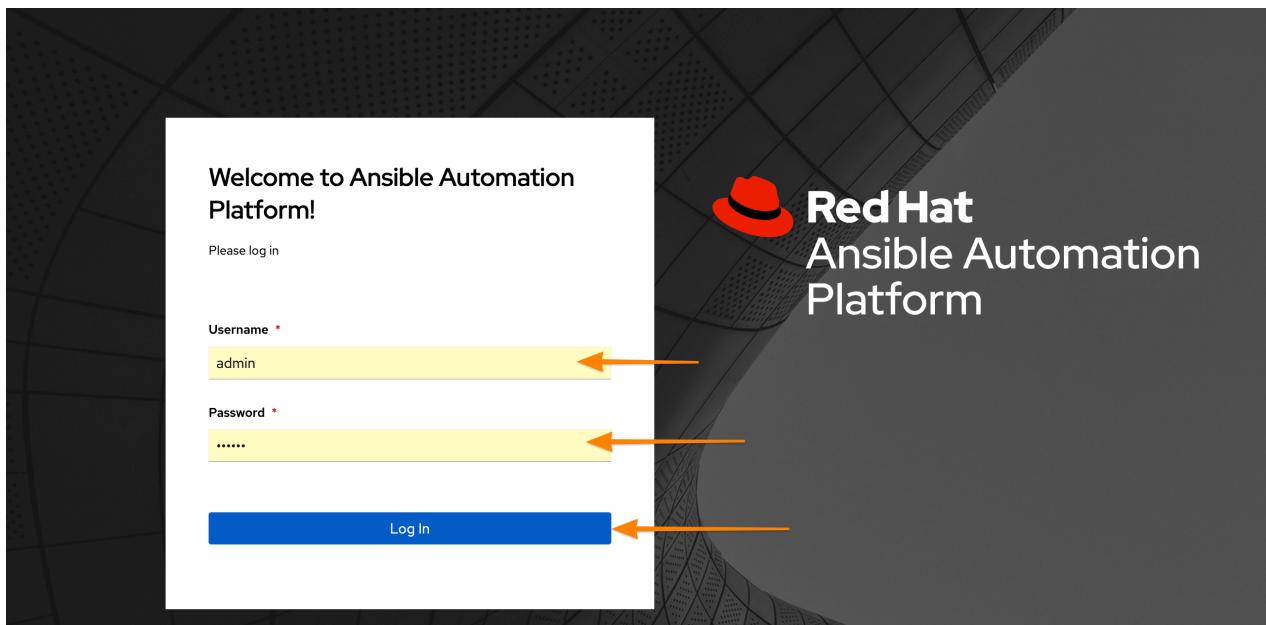


Figure 36. Ansible Controller Login

#### 2. Click **Projects** and then click **Add** to Add a Project

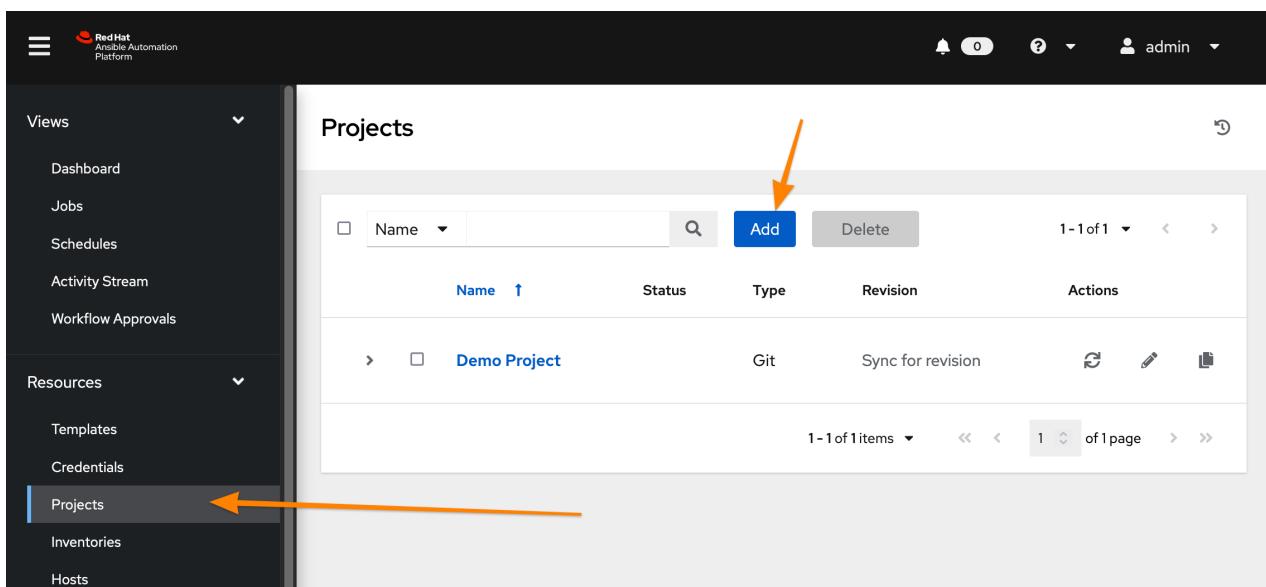


Figure 37. Ansible Controller - Projects

#### 3. Create a New Project with a Name, Organization and Source Control Credential

- **Name:** NYPD Webserver
- **Organization:** NYPD
- **Source Control Credential Type:** Git
- Source Control URL: [git@github.com:tmichett/AAP\\_Webinar.git](git@github.com:tmichett/AAP_Webinar.git)
- **Source Control Credential:** Travis Github

The screenshot shows the 'Edit Details' page for a project named 'NYPD Webserver'. The page has several fields and sections:

- Name \***: NYPD Webserver (highlighted by an orange arrow)
- Description**: (empty field)
- Organization \***: NYPD (highlighted by an orange arrow)
- Default Execution Environment**: (empty search bar)
- Source Control Credential Type \***: Git (highlighted by an orange arrow)
- Type Details** section:
  - Source Control URL \***: git@github.com:tmichett/AAP\_W... (highlighted by an orange arrow)
  - Source Control Branch/Tag/Commit**: (empty field)
  - Source Control Refspec**: (empty field)
- Source Control Credential** section:
  - Travis Github** (highlighted by an orange arrow)
- Options** section:
  - Clean
  - Delete
  - Track submodules
  - Update Revision on Launch
  - Allow Branch Override
- Buttons**: Save (highlighted by an orange arrow) and Cancel

Figure 38. Ansible Controller - Creating Project from Github Source



After clicking **Save** it will initiate the first sync of the project resources. Wait for the sync to complete successfully.

#### Creating a Job Template

1. Click **Templates** and then click **Add** to Add (\*Add job template) to create a job template.

The screenshot shows the Ansible Controller interface. The left sidebar has a 'Views' section with 'Dashboard', 'Jobs', 'Schedules', 'Activity Stream', and 'Workflow Approvals'. Below 'Views' is a 'Resources' section with 'Templates' (which is selected and highlighted with an orange arrow), 'Credentials', 'Projects', 'Inventories', and 'Hosts'. The main content area is titled 'Templates' and contains a table. The table has columns for 'Name', 'Type', and 'Last Ran'. There is one entry: 'Demo Job Template' (Job Template). At the top of the table, there are buttons for 'Add' (highlighted with an orange arrow) and 'Delete'. Below the table, it says '1-1 of 1 items'. At the bottom right, there are navigation buttons for '1 of 1 page'.

Figure 39. Ansible Controller - Job Templates

2. Create the new job template and then click **Save**

- Provide a name
  - **Name:** NYPD Webserver Deploy
- Provide job type (**run**)
- Provide **Inventory**
  - **Inventory:** NYPD Systems
- Provide **Project**
  - **Project:** NYPD Webserver
- Select **Playbook**
  - **Playbook:** Future/NYPD/Website\_Ansible\_Past.yml
- Select **Credentials**
  - **Credentials:** NYPD Machine SSH Creds
- Select **Privilege Escalation** option

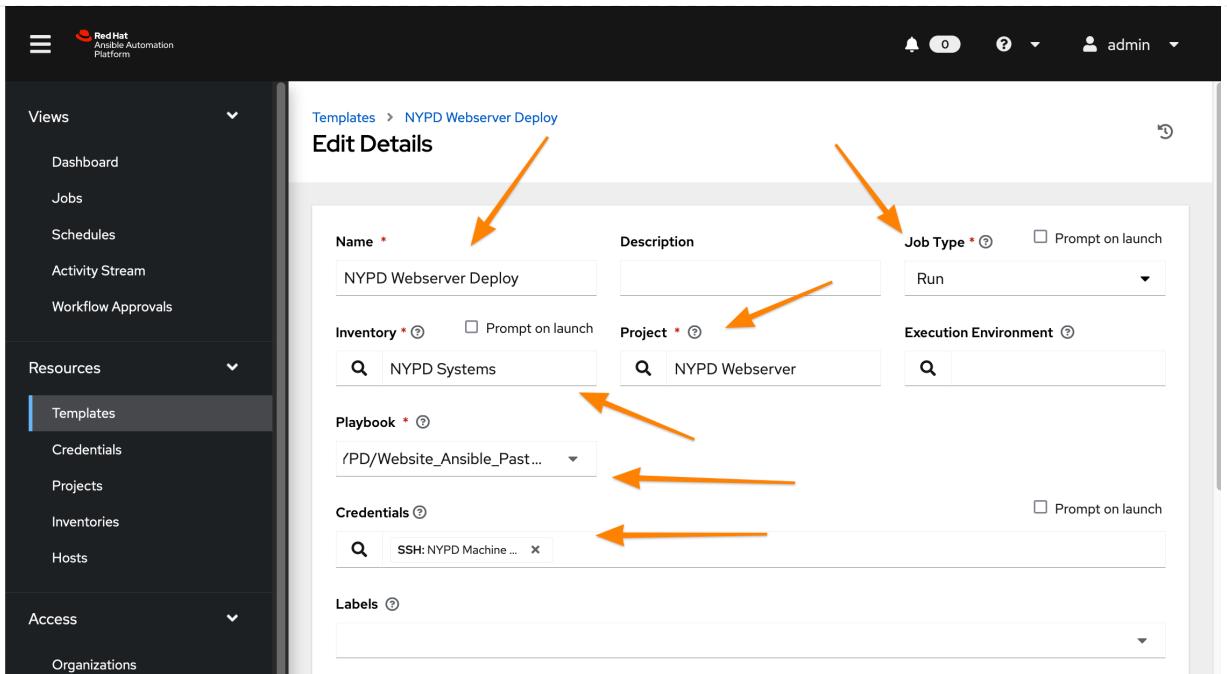


Figure 40. Ansible Controller - Job Template Details

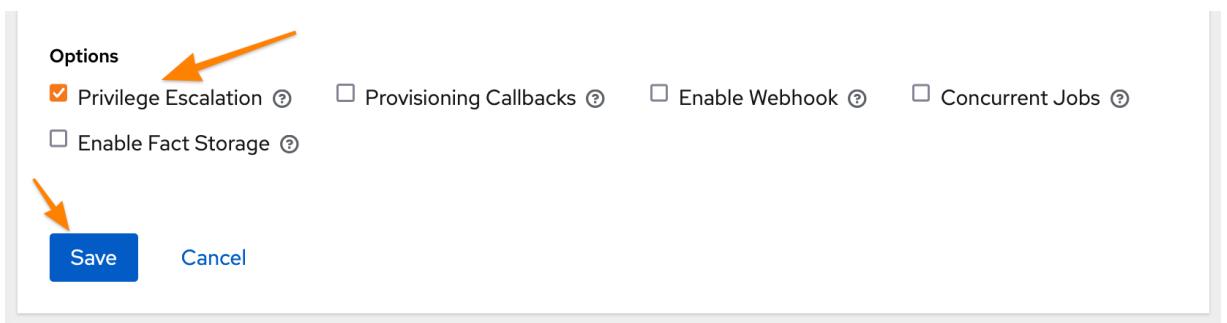


Figure 41. Ansible Controller - Job Template Details cont.



#### Privilege Escalation

It is important to know what the playbook does and whether it requires privilege escalation. A proper playbook might already have this defined, but it also allows you to assign it to the job from this menu.

3. Launch the job by clicking **Launch**



## Details

[◀ Back to Templates](#) **Details** Access Notifications Schedules Jobs Survey

Name	NYPD Webserver Deploy	Job Type	run	Organization	NYPD
Inventory	NYPD Systems	Project	NYPD Webserver	Execution Environment	An Ansible Engine 2.9 execution environment
Playbook	Future/NYPD/Website_Anible_Past.yml	Forks	0	Verbosity	0 (Normal)
Timeout	0	Show Changes	Off	Job Slicing	1
Created	1/12/2022, 4:40:57 PM by admin	Last Modified	1/12/2022, 4:43:38 PM by admin		

Credentials [SSH: NYPD Machine ...](#)Variables [YAML](#) [JSON](#)

```
1 ---
```

[Edit](#)[Launch](#)[Delete](#)

Figure 42. Ansible Controller - Job Template Launch

Jobs > NYPD Webserver Deploy

## Output

◀ Back to Jobs Details Output

**NYPD Webserver Deploy** Plays 1 Tasks 6 Hosts 1 Elapsed 00:00:10

Stdout ▾

```
/ TASK [Install Packages for Webserver] *****
16:51:21
8 ok: [serverd]
9
10 TASK [Create Content for Webserver] *****
16:51:23
11 ok: [serverd]
12
13 TASK [Firewall is Enabled] *****
16:51:24
14 ok: [serverd]
15
16 TASK [HTTP Service is Open on Firewall] *****
16:51:25
17 changed: [serverd]
18
19 TASK [httpd is started] *****
16:51:25
20 changed: [serverd]
```

Figure 43. Ansible Controller - Job Results Output Verification

4. Verify webserver is running and accessible.

```
[student@workstation ~]$ curl serverd
I'm an awesome webserver for the NYPD and I know Castle!!
```

### 3.3.4. Workflows

In order to create job workflows, projects and existing job templates must already be created before they can be put together as a job workflow template.

#### 3.3.4.1. DEMO - Creating Job Workflows

*Setup Required*



Before beginning this demo or exercise, it is necessary to run the **setup** playbooks.

1. Switch to correct directory

```
[student@workstation ~]$ cd  
/home/student/Github/AAP_Webinar/Future/Setup
```

2. Run the **Setup\_All.yml** playbook

```
[student@workstation Setup]$ ansible-playbook Setup_All.yml  
... OUTPUT OMITTED ...
```

It is also necessary to synchronize the Project Inventory Source

1. Click **Inventories**
2. Click **NYPD Web Workflow**
3. Click **Sources**
4. Click the **Synchronize** action to synchronize the **NYPD Project Inventory** source
  - Wait for status to be **Green** meaning synchronization was successful
5. Click **Hosts** to verify inventory was imported

This allows steps to be skipped and it is possible to jump to the **Creating a Job Workflow Template** instructions - Starting with **STEP 5**

#### *Example 15. DEMONSTRATION - Job Workflow Templates*

For this demonstration, it will be necessary to create two new **Job Templates** that will be linked together in a **Job Workflow Template**. We will be leveraging the already created project **NYPD Webserver** for existing playbooks and inventories. We will also create a dynamic inventory based on imported inventory from the project.

#### *Creating a Project-Based Inventory Source*

1. Login to **Ansible Controller**

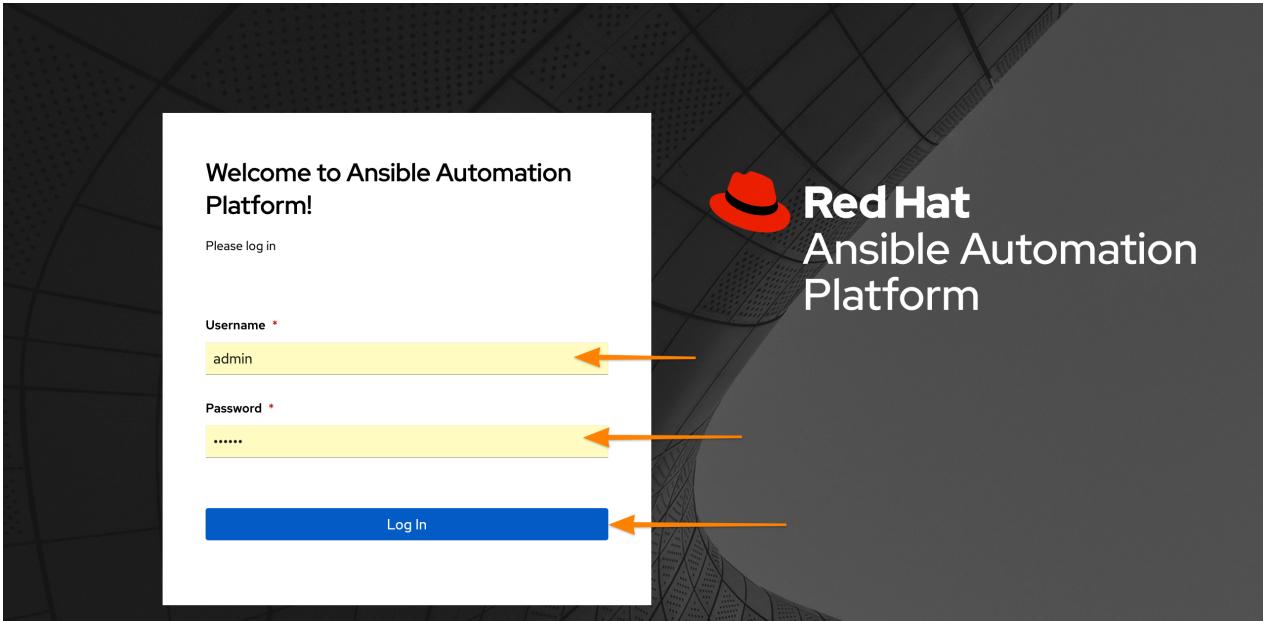


Figure 44. Ansible Controller Login

2. Click **Inventories** and then click **Add**

A screenshot of the Ansible Controller - Inventory page. The left sidebar shows "Views" with "Dashboard", "Jobs", "Schedules", "Activity Stream", and "Workflow Approvals". Under "Resources", it shows "Templates", "Credentials", "Projects", and "Inventories", with "Inventories" highlighted by a blue bar and an orange arrow pointing to it. The main content area is titled "Inventories" and contains a table with one row. The table columns are "Name", "Status", "Type", "Organization", and "Actions". The row shows "Demo Inventory" with status "Disabled", type "Inventory", organization "Default", and edit/pencil icons in the Actions column. An orange arrow points from the right side of the image towards the "Add" button in the top right of the inventory table.

Figure 45. Ansible Controller - Inventory

3. Assign a **Name** and **Organization** to the Inventory and then click **Save**

Inventories

## Create new inventory

Name \* NYPD Web Workflow

Description

Organization \* NYPD

Instance Groups

Variables ? YAML JSON

1 ---

**Save** **Cancel**

Figure 46. Ansible Controller - Inventory Creation

4. Click **Sources** to create an inventory source

Inventories > NYPD Web Workflow

## Details

Back to Inventories Details Access Groups Hosts Sources Jobs

Name NYPD Web Workflow Type Inventory Organization NYPD

Variables YAML JSON

1 ---

Created 1/13/2022, 12:32:32 PM Last Modified 1/13/2022, 12:32:32 PM by admin

**Edit** **Delete**

Figure 47. Ansible Controller - Inventory Sources

5. Click **Add** to add an inventory source

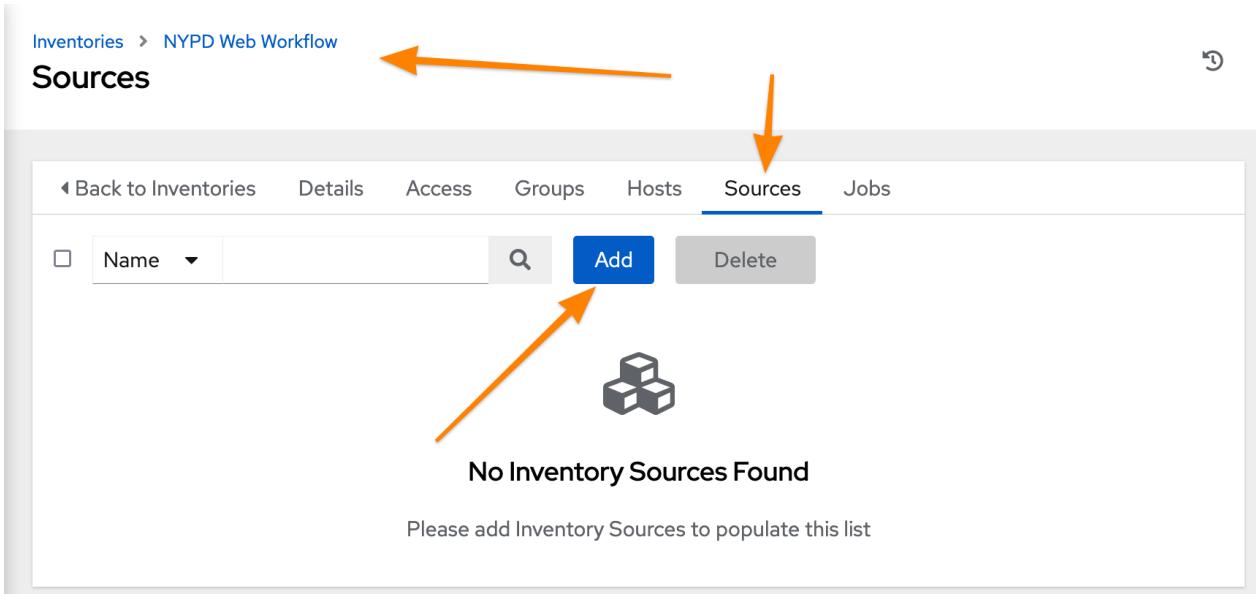


Figure 48. Ansible Controller - Adding Inventory Sources

6. Provide a **Name** and choose **Sourced from a Project** as source and click **Save**

- Select the **Project** and **Inventory file**
- Check **Update on launch**



## Create new source

The screenshot shows the 'Create new source' page in the Ansible Controller. The 'Name \*' field contains 'NYPD Project Inventory'. The 'Source \*' dropdown is set to 'Sourced from a Project'. The 'Project \*' dropdown shows 'NYPD Webserver'. The 'Inventory file \*' dropdown shows 'Future/NYPD/inventory'. Under 'Update options', the 'Update on launch' checkbox is checked. There are also other optional checkboxes for 'Overwrite' and 'Overwrite variables'.

Name *	Description	Execution Environment
NYPD Project Inventory		

Source *
Sourced from a Project

Source details		
Credential	Project *	Inventory file * ⓘ
<input type="text"/>	<input type="text"/> NYPD Webserver	<input type="text"/> Future/NYPD/inventory
Verbosity ⓘ	Host Filter ⓘ	Enabled Variable ⓘ
1 (Info)		
Enabled Value ⓘ		

Update options				
<input type="checkbox"/> Overwrite ⓘ	<input type="checkbox"/> Overwrite variables ⓘ	<input checked="" type="checkbox"/> Update on launch ⓘ	<input type="checkbox"/> Update on project update ⓘ	

Figure 49. Ansible Controller - Configuring Inventory Sources

7. Click **Sync** to perform a synchronization

Inventories > NYPD Web Workflow > Sources > NYPD Project Inventory

**Details**

Back to Sources Details Schedules Notifications

Name	NYPD Project Inventory	Source	Sourced from a Project	Organization	NYPD
Project	NYPD Webserver	Inventory file	Future/NYPD/inventory	Verbosity	1 (Info)
Cache timeout	0 seconds				
Enabled Options	Update on launch				
Source variables <a href="#">YAML</a> <a href="#">JSON</a>					
<pre>1 ---</pre>					
Created	1/13/2022, 12:38:21 PM by admin	Last modified	1/13/2022, 12:38:21 PM by admin		
<a href="#">Edit</a> <a href="#">Sync</a> <a href="#">Delete</a>					

Figure 50. Ansible Controller - Synchronizing Inventory Sources

## 8. Click Inventories to verify the inventory and select NYPD Web Workflow

Views

- Dashboard
- Jobs
- Schedules
- Activity Stream
- Workflow Approvals

Resources

- Templates
- Credentials
- Projects
- Inventories**
- Hosts

Access

**Inventories**

<input type="checkbox"/>	Name	Status	Type	Organization	Actions
<input type="checkbox"/>	Demo Inventory	Disabled	Inventory	Default	<a href="#">Edit</a> <a href="#">Delete</a>
<input type="checkbox"/>	NYPD Systems	Disabled	Inventory	NYPD	<a href="#">Edit</a> <a href="#">Delete</a>
<input type="checkbox"/>	<b>NYPD Web Workflow</b>	Success	Inventory	NYPD	<a href="#">Edit</a> <a href="#">Delete</a>

Figure 51. Ansible Controller - Verifying Inventory Sources

## 9. Click Hosts to view hosts

Inventories > NYPD Web Workflow

## Hosts

Back to Inventories Details Access Groups Hosts Sources Jobs

Name ▾

1 - 6 of 6

Name	Actions
servera	<input checked="" type="checkbox"/> On <input type="button" value=""/>
serverb	<input checked="" type="checkbox"/> On <input type="button" value=""/>
serverc	<input checked="" type="checkbox"/> On <input type="button" value=""/>
serverd	<input checked="" type="checkbox"/> On <input type="button" value=""/>
servere	<input checked="" type="checkbox"/> On <input type="button" value=""/>
serverf	<input checked="" type="checkbox"/> On <input type="button" value=""/>

Figure 52. Ansible Controller - Verifying Inventory Hosts from Project

10. Click **Groups** to view host groups
  - a. Click on a group name to see hosts in group and click **Hosts**

Inventories > NYPD Web Workflow

## Groups

	Name	Actions
<input type="checkbox"/>	dev	
<input type="checkbox"/>	test	

Figure 53. Ansible Controller - Verifying Inventory Group from Project

Inventories > NYPD Web Workflow > Groups > test

## Hosts

	Name	Activity	Actions
<input type="checkbox"/>	serverf		

Figure 54. Ansible Controller - Verifying Inventory Group (Hosts) from Project

### Project Based Inventory

The above example shows how to create a dynamic inventory that is sourced from a project. This would can be done to ensure that you have the same inventory and host systems as the developers. It is not 100% necessary to have inventory in the projects, but some people prefer to keep host inventory in projects and this is a great method in keeping developer inventory in sync with what is stored in Ansible Controller.



README FIRST

If you ran the lab scripts and Ansible playbook in the before you begin section, it is possible to skip to **STEP 5** of the **Creating a Job Workflow Template** steps. The playbook created the inventory and the intermediate Job Templates so the focus is creation of a Job Workflow. This has been done to save time for the demonstration and prevents the need to manually create new inventories and job templates.

## Creating a Job Workflow Template

### 1. Login to Ansible Controller

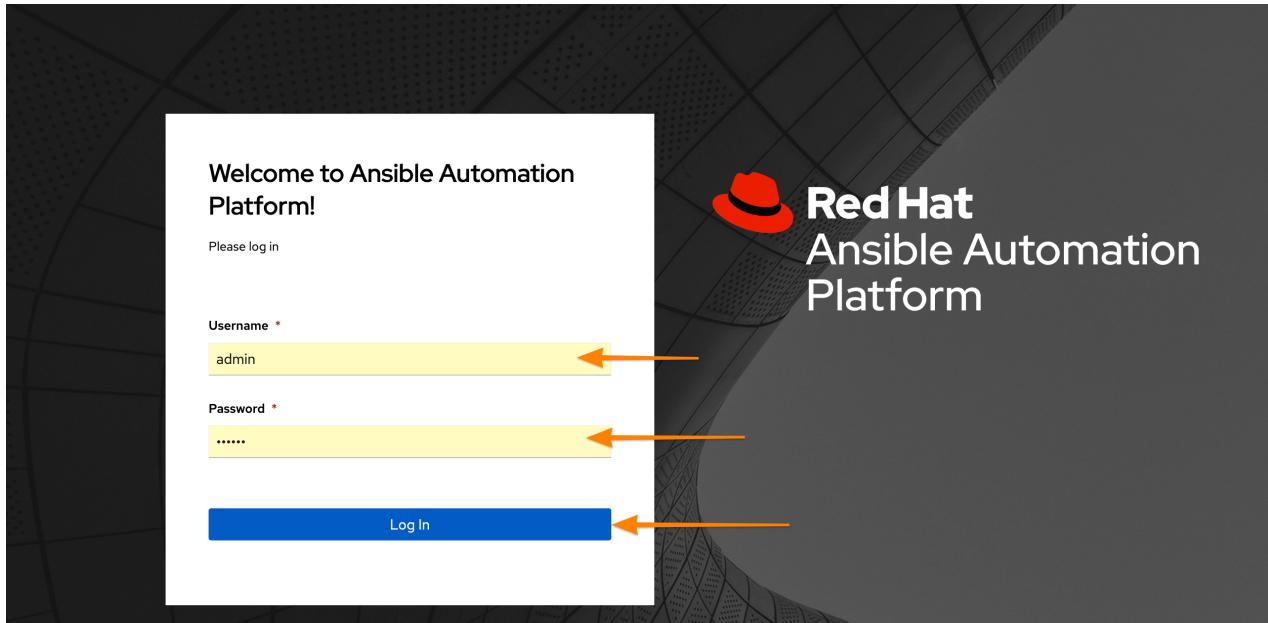


Figure 55. Ansible Controller Login

### 2. Click **Templates** and then click **Add** and **(Add job template)** to create a new Job Template

The screenshot shows the Ansible Controller interface. The top navigation bar includes the Red Hat logo, user information (admin), and a search bar. The left sidebar has a "Views" section with links to Dashboard, Jobs, Schedules, Activity Stream, and Workflow Approvals. Under "Resources", there are links for Templates, Credentials, Projects, Inventories, and Hosts, with "Templates" being highlighted. The main content area is titled "Templates" and shows a table of existing job templates. The table has columns for Name, Type, Last Ran, and Actions. Two entries are listed: "Demo Job Template" (Job Template, last ran 1/12/2022, 4:51:27 PM) and "NYPD Webserver Deploy" (Job Template, last ran 1/12/2022, 4:51:27 PM). An orange arrow points from the "Templates" link in the sidebar to the "Add" button in the top right of the main table area.

Figure 56. Ansible Controller - Job Templates

3. Complete the form for the NYPD Dev Webserver and click **Save**

- a. **Name:** *NYPD Dev Webserver*
- b. **Job Type:** *run*
- c. **Inventory:** *NYPD Web Workflow*
- d. **Project:** *NYPD Webserver*
- e. **Playbook:** *Future/NYPD/NYPD\_Web\_Workflow.yml*
- f. **Credentials:** *NYPD Machine SSH Creds*
- g. **Variables:** *inv\_host\_var: servere*
- h. **Privilege Escalation:** *Checked*

Templates > NYPD Dev Webserver

### Edit Details

The screenshot shows the 'Edit Details' page for a job template named 'NYPD Dev Webserver'. The page has several sections with input fields and dropdown menus. Orange arrows point from the numbered steps above to specific fields on the page:

- Step a: Points to the 'Name' field containing 'NYPD Dev Webserver'.
- Step b: Points to the 'Job Type' dropdown menu set to 'Run'.
- Step c: Points to the 'Inventory' search bar containing 'NYPD Web Workflow'.
- Step d: Points to the 'Project' search bar containing 'NYPD Webserver'.
- Step e: Points to the 'Playbook' dropdown menu showing 'Future/NYPD/NYPD\_Web ...'.
- Step f: Points to the 'Credentials' search bar containing 'SSH: NYPD Machine ...'.
- Step g: Points to the 'Variables' section, which is expanded to show two YAML entries:

```
1 ---  
2 inv_host_var: servere
```

Figure 57. Ansible Controller - Job Template Parameters

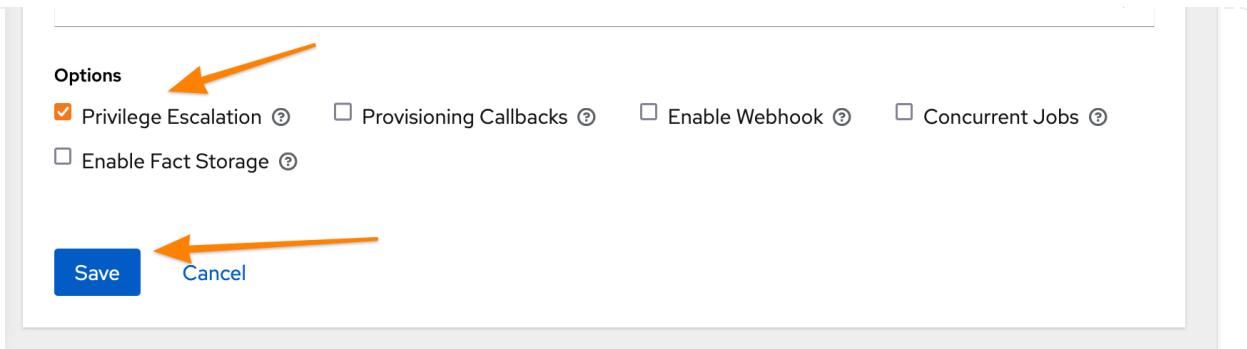


Figure 58. Ansible Controller - Job Template Parameters cont.

4. Create a new Job template using steps above with the following values.
  - a. **Name:** *NYPD Test Webserver*
  - b. **Job Type:** *run*
  - c. **Inventory:** *NYPD Web Workflow*
  - d. **Project:** *NYPD Webserver*
  - e. **Playbook:** *Future/NYPD/NYPD\_Web\_Workflow.yml*
  - f. **Credentials:** *NYPD Machine SSH Creds*
  - g. **Variables:** *inv\_host\_var: serverf*
  - h. **Privilege Escalation:** *Checked*

Templates

### Create New Job Template

Name \* NYPD Test Webserver

Description

Job Type \* Run

Inventory \* NYPD Web Workflow

Project \* NYPD Webserver

Execution Environment

Playbook \* Future/NYPD/NYPD\_Web ...

Credentials SSH: NYPD Machine ...

Labels

Variables YAML JSON

```
1 ---  
2 inv_host_var: serverf
```

The screenshot shows the 'Create New Job Template' page in the Ansible Controller. The 'Name' field is set to 'NYPD Test Webserver'. The 'Inventory' field is set to 'NYPD Web Workflow'. The 'Project' field is set to 'NYPD Webserver'. The 'Playbook' dropdown is set to 'Future/NYPD/NYPD\_Web ...'. The 'Credentials' dropdown is set to 'SSH: NYPD Machine ...'. The 'Variables' section shows a YAML block with two lines: '1 ---' and '2 inv\_host\_var: serverf'. There are orange arrows pointing to each of these fields.

Figure 59. Ansible Controller - Job Template Parameters for NYPD Test Webserver

5. Click **Templates** then click **Add** and select **Add workflow template**

Name	Last Ran	Actions
Demo Job Template		[Edit, Delete]
NYPD Dev Webserver		[Edit, Delete]
NYPD Test Webserver		[Edit, Delete]
NYPD Webserver Deploy	1/12/2022, 4:51:27 PM	[Edit, Delete]

Figure 60. Ansible Controller - Job Workflow Template

6. Provide a **Name** and select the appropriate items
  - a. **Name:** NYPD DevOps Workflow
  - b. **Description:** Deploy systems to Dev and Test
  - c. **Inventory:** Leave Blank (Will use inventory specified for Job Templates)
  - d. **Organization:** NYPD

## Create New Workflow Template

The screenshot shows the 'Create New Workflow Template' page in the Ansible Controller. The 'Name' field is populated with 'NYPD DevOps Workflow'. The 'Organization' dropdown is set to 'NYPD'. In the 'Options' section, the 'Enable Webhook' checkbox is checked, while 'Enable Concurrent Jobs' is unchecked.

Name *	Description	Organization
NYPD DevOps Workflow	Deploy systems to Dev and Test	NYPD

**Inventory**  Prompt on launch  
**Limit**  Prompt on launch  
**Source control branch**  Prompt on launch

**Labels**

**Variables**  Prompt on launch

**Options**  
 Enable Webhook  Enable Concurrent Jobs

Figure 61. Ansible Controller - Job Workflow Template Details

7. The **Workflow Visualizer** will open and click **Start** to define first task in workflow

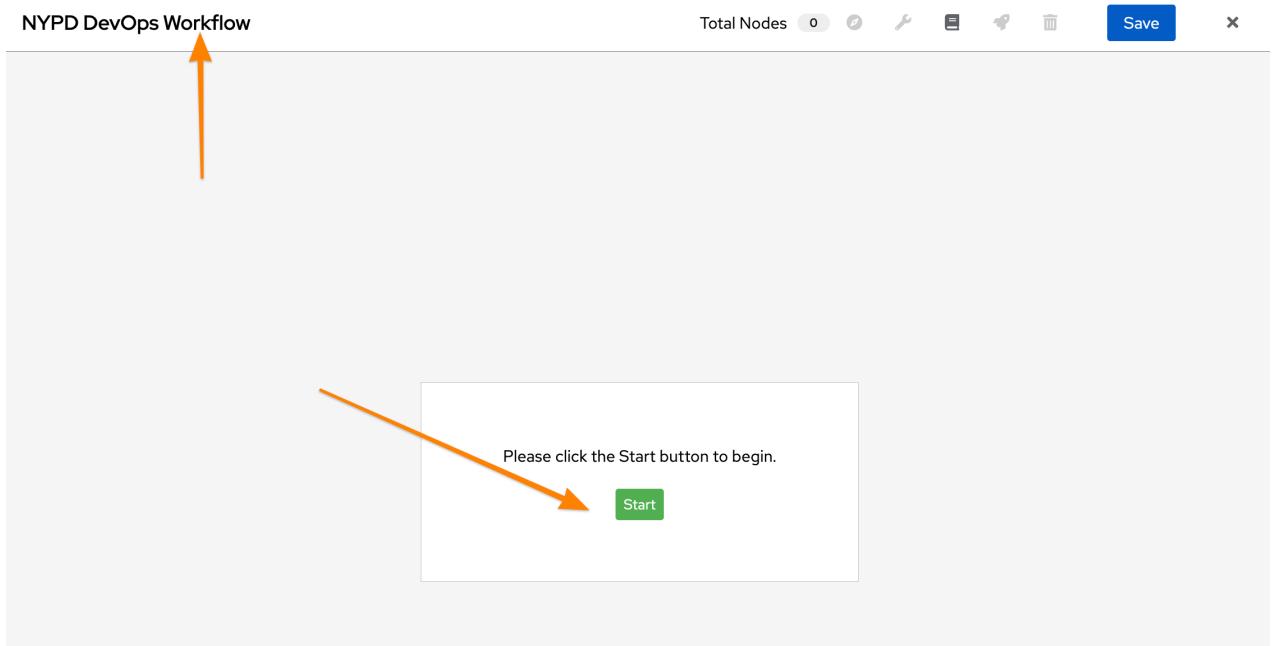


Figure 62. Ansible Controller - Job Workflow Visualizer

8. Start by selecting **Node Type** of **Project Sync** and select the **NYPD Webserver** Project then click **Save**

Add Node | NYPD Webserver

1 Node type

Node Type Project Sync

Name

Name ↑

Demo Project

NYPD Webserver

Convergence \*

Save Cancel

Figure 63. Ansible Controller - Job Workflow Task #1 Synchronize Project Data

9. Add Step #2 to run on **Success** by selecting the NYPD Webserver Project and clicking the +.
  - Select **Run** and **On Success** then click **Next**

- b. Select **Node Type** to be **Job Template** and select the **NYPD Dev Webserver** then click **Save**

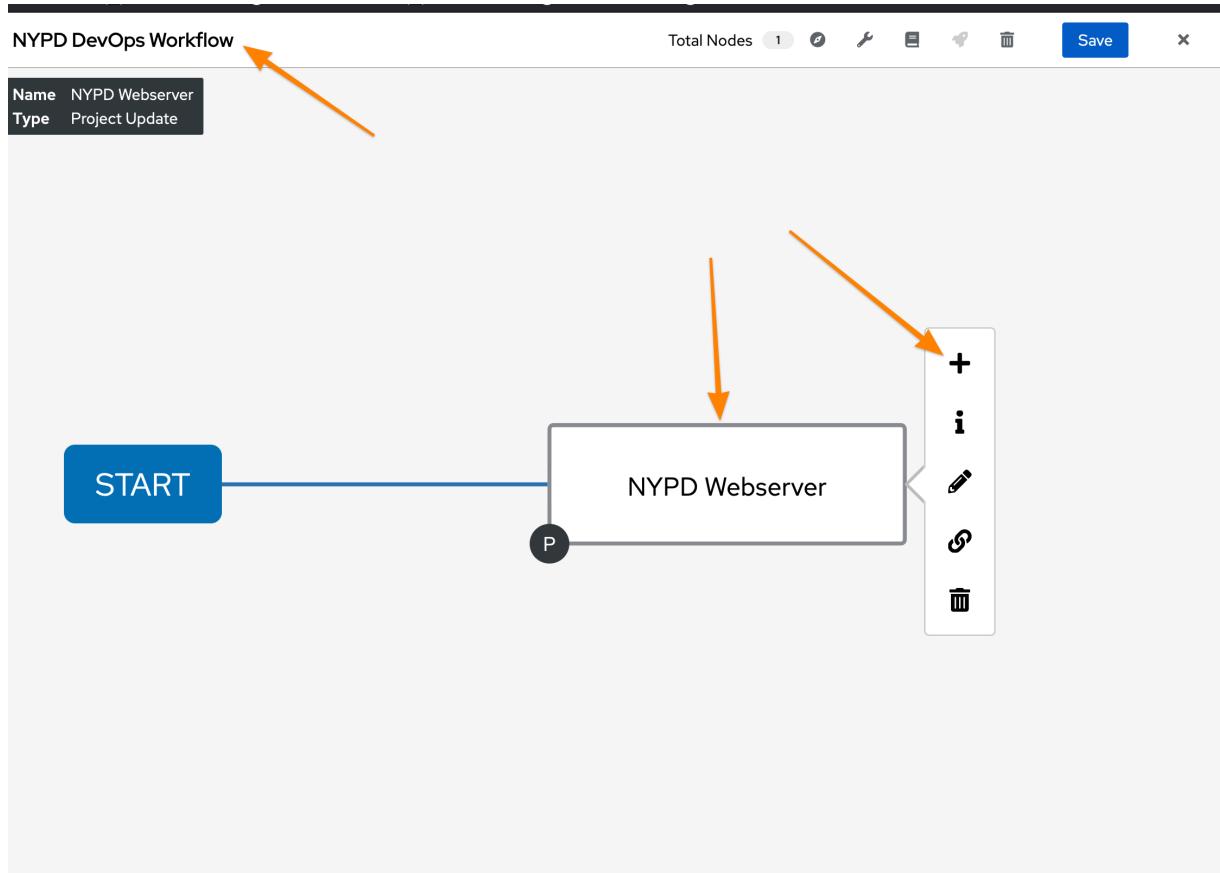


Figure 64. Ansible Controller - Job Workflow Task #2 Launch Development Job

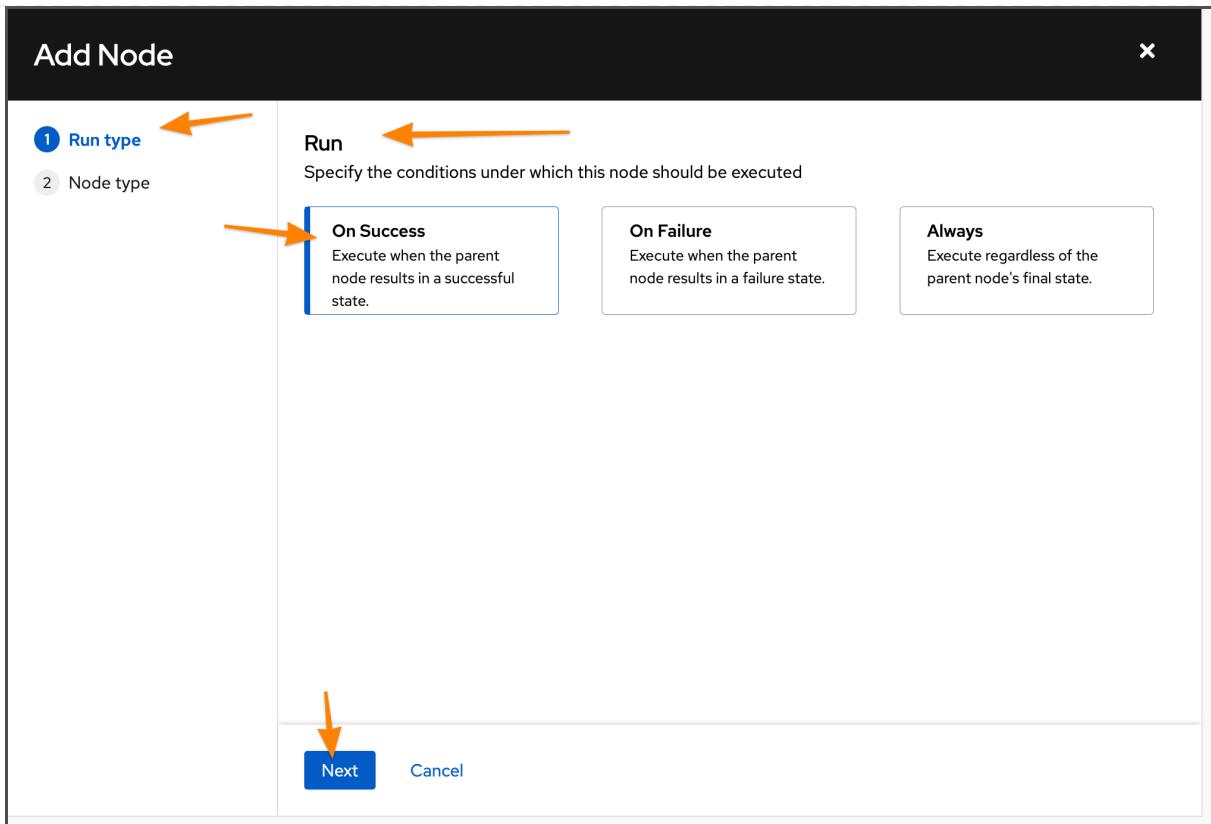


Figure 65. Ansible Controller - Job Workflow Task #2 Selecting Job Run Parameters

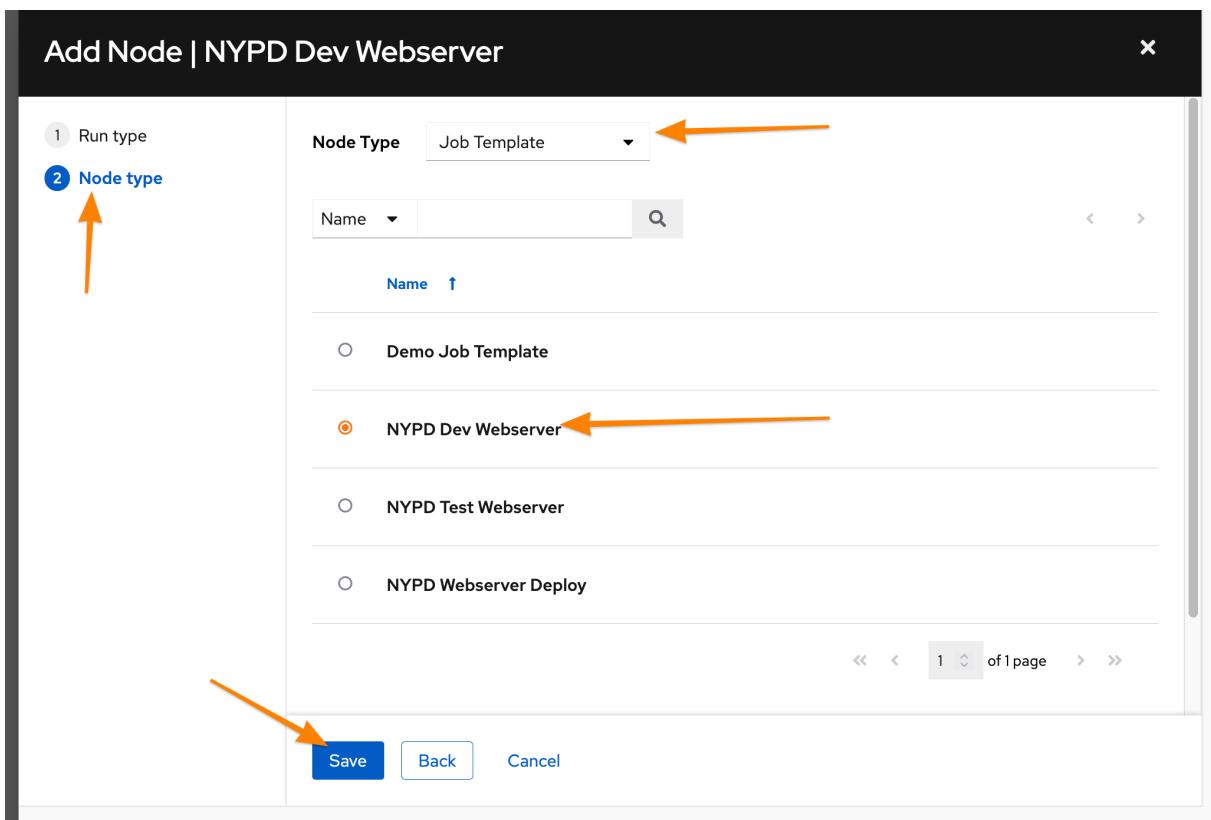


Figure 66. Ansible Controller - Job Workflow Task #2 Selecting Job Template to Run

10. Add Step #3 to run on **Success** by selecting the NYPD Webserver Project and clicking the +.

- a. Select **Run** and **On Success** then click **Next**
- b. Select **Node Type** to be **Job Template** and select the **NYPD Test Webserver** then click **Save**

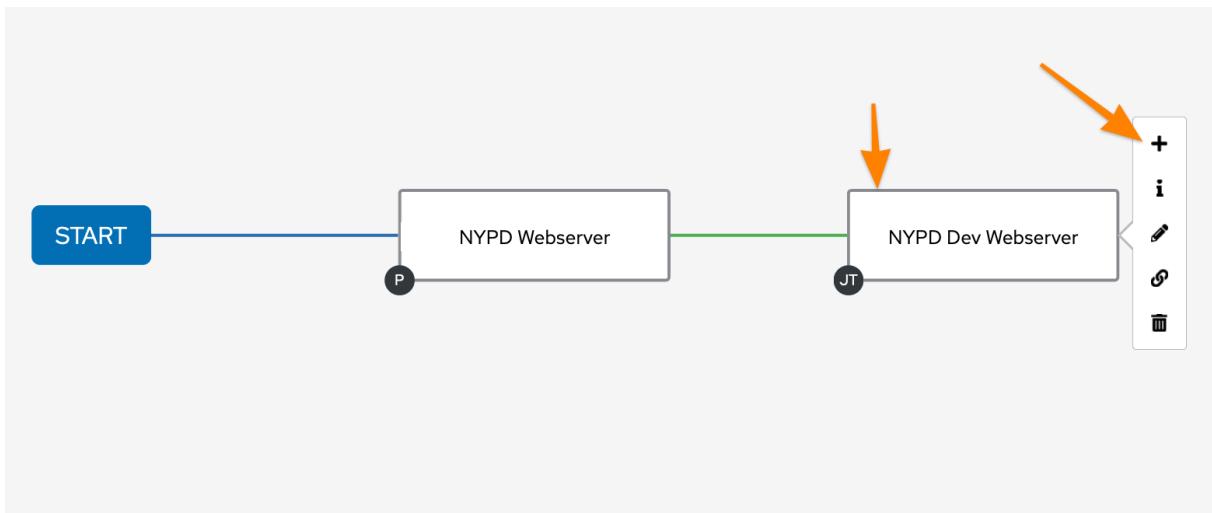


Figure 67. Ansible Controller - Job Workflow Task #3 Launch Development Job

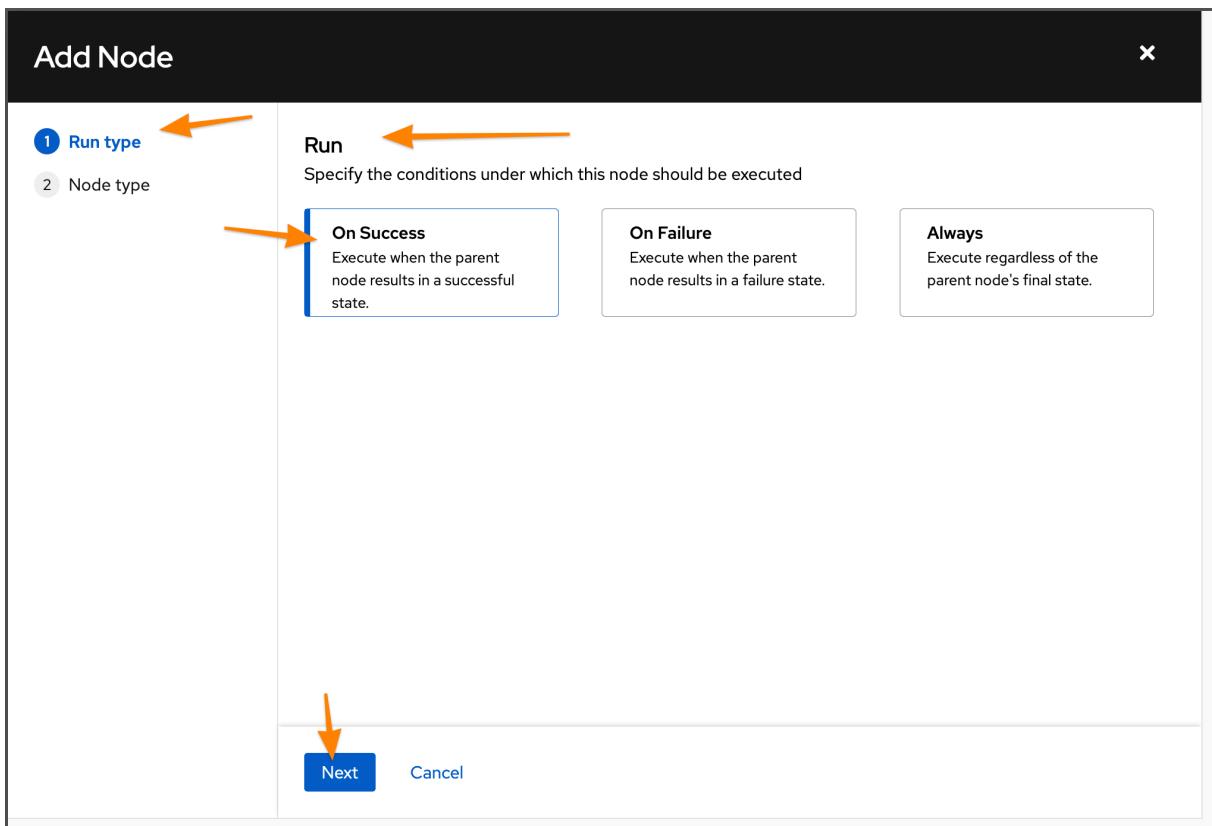


Figure 68. Ansible Controller - Job Workflow Task #3 Selecting Job Run Parameters

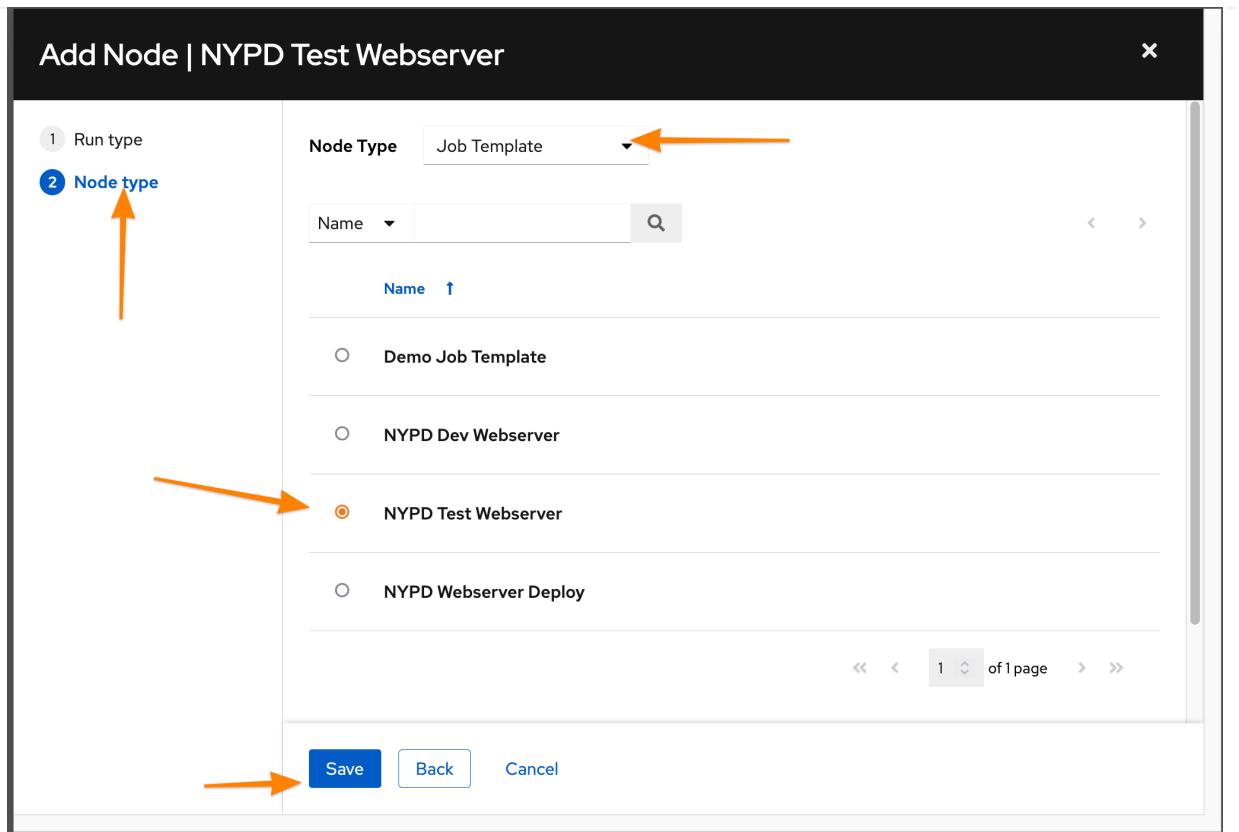


Figure 69. Ansible Controller - Job Workflow Task #3 Selecting Job Template to Run

11. View the complete Job Workflow and click **Save** when done adding steps

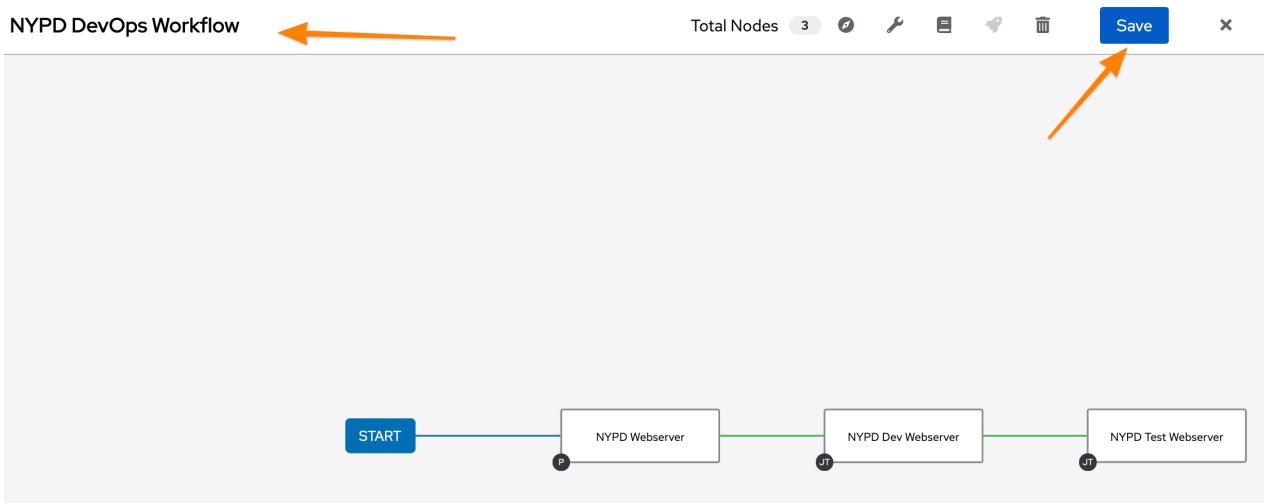


Figure 70. Ansible Controller - Job Workflow Complete Overview

#### Job Workflows - Success, Failure, and Always



It is important to architect workflows properly. In this example, we defined the **SUCCESS** path. In general, you will most likely have failure paths to cleanup environments from failed workflows. This has been left out based on time.

## 12. Launch job workflow to test by clicking **Launch**

Templates > NYPD DevOps Workflow

**Details**

Name: NYPD DevOps Workflow    Description: Deploy systems to Dev and Test    Organization: NYPD

Job Type: Workflow Job Template    Created: 1/13/2022, 1:10:17 PM by admin    Modified: 1/13/2022, 1:10:17 PM by admin

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

1 ---

Edit **Launch** Delete

Figure 71. Ansible Controller - Job Workflow Launching

## 13. View workflow output

Jobs > NYPD DevOps Workflow

**Output**

NYPD DevOps Workflow

Total Nodes: 3

START → NYPD Webserver (00:00:06) → NYPD Dev Webserver (00:00:13) → NYPD Test Webserver (00:00:13)

Figure 72. Ansible Controller - Job Workflow Output

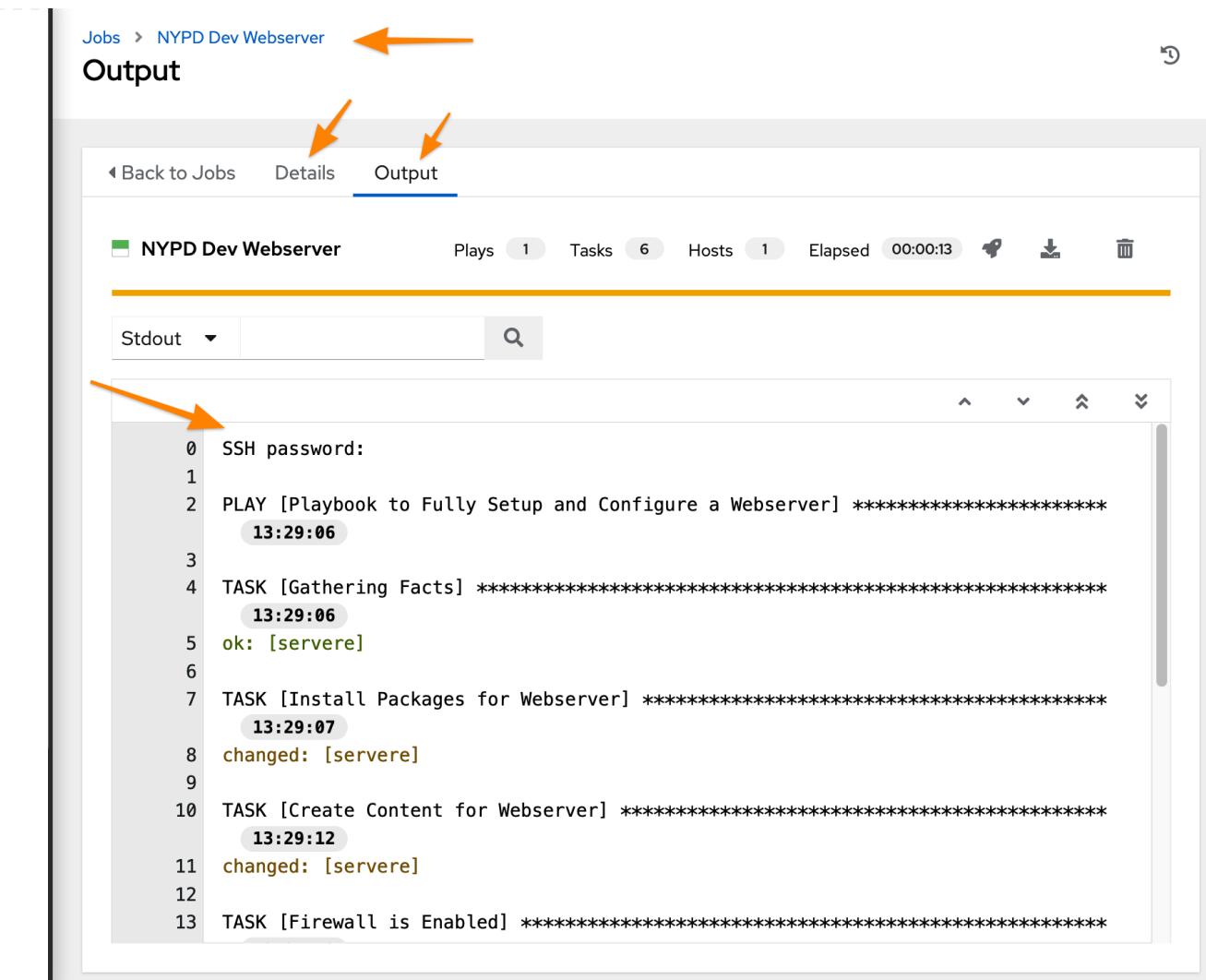


Figure 73. Ansible Controller - Job Workflow Output - Details for Job Template

#### 14. Test from workstation

*Listing 18. Dev Server test*

```
[student@workstation ~]$ curl servere
I'm an awesome webserver for the NYPD and I know Castle!!
```

*Listing 19. Test Server test*

```
[student@workstation ~]$ curl serverf
I'm an awesome webserver for the NYPD and I know Castle!!
```