Ansible Workshop - Exercises

Automation Platform

Learn to manage and run your Ansible content in AAP.

2 - Inventory & Ad-hoc commands

Objective

Explore and understand the lab environment. This exercise will cover

- Locating and understanding:
  - Ansible Automation Controller **Inventory**
  - Ansible Automation Controller **Credentials**
- Running ad hoc commands via the Ansible Automation Controller web UI

Guide

Examine an Inventory

The first thing we need is an inventory of your managed hosts. This is the equivalent of an inventory file in Ansible Engine. There is a lot more to it (like dynamic inventories) but let's start with the basics.

- You should already have the web UI open, if not: Point your browser to the URL you were given, similar to `https://demo.redhat.com/workshop/pm6xgd` (the *workshop ID* will be different) and log in as `admin`. The password will be provided by the instructor.

There will be one inventory, the **Workshop Inventory**. Click the **Workshop Inventory** then click the **Hosts** button

The inventory information at `~/lab_inventory/hosts` was pre-loaded into the Ansible Automation controller Inventory as part of the provisioning process.

```
[web]
node1 ansible_host=node1.example.com
node2 ansible_host=node2.example.com
node3 ansible_host=node3.example.com

[control]
ansible-1 ansible_host=ansible-1.example.com
```

Warning

In your inventory the IP addresses will be different, do not copy the values above!

Examine Machine Credentials

Now we will examine the credentials to access our managed hosts from Automation controller. As part of the provisioning process for this Ansible Workshop the **Workshop Credential** has already been setup.

In the **Resources** menu choose **Credentials**. Now click on the **Workshop Credential**.

Note the following information:

Parameter	Value	Description
Credential Type	`Machine`	Machine credentials define ssh and user-level privilege escalation access for playbooks. They are used when submitting jobs to run playbooks on a remote host.
Username	`ec2-user`	The user which matches our command-line Ansible inventory username for the other Linux nodes
SSH Private Key	`Encrypted`	Note that you can't actually examine the SSH private key once someone hands it over to Ansible Automation controller

Run Ad Hoc commands

It is possible to run run ad hoc commands from Ansible Automation controller as well.

> **Tip**
>
> **Ensure** that all hosts are available and can be included in automation jobs.
> Got to **Resources** → **Hosts** and move the slider on the right to **On** for all hosts.

- In the web UI go to **Resources** → **Inventories** → **Workshop Inventory**
- Click the **Hosts** tab to change into the hosts view and select the three hosts *node1* to *node3* by ticking the boxes to the left of the host entries.
- Click **Run Command** button. In the next screen you have to specify the ad hoc command.

Within the **Details** window, select **Module** `ping` and click **Next**.

Within the **Execution Environment** window, select **Default execution environment** and click **Next**.

Within the **Machine Credential** window, select **Workshop Credentials** and click **Launch**.

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

The simple **ping** module doesn't need options. For other modules you need to supply the command to run as an argument. Try the **command** module to find the userid of the executing user using an ad hoc command.

- In the web UI go to **Resources** → **Inventories** → **Workshop Inventory**
- Click the **Hosts** tab to change into the hosts view and select the three hosts by ticking the boxes to the left of the host entries.

- Click **Run Command** button. In the next screen you have to specify the ad hoc command.

Within the **Details** window, select **Module** `command`, in **Arguments** type `id` and click **Next**.

Within the **Execution Environment** window, select **Default execution environment** and click **Next**.

Within the **Machine Credential** window, select **Workshop Credentials** and click **Launch**.

> **Tip**
>
> After choosing the module to run, Ansible Automation Controller will provide a link to the docs page for the module when clicking the question mark next to "Arguments". This is handy, give it a try.

How about trying to get some secret information from the system? Try to print out `/etc/shadow`.

- In the web UI go to **Resources** → **Inventories** → **Workshop Inventory**
- Click the **Hosts** tab to change into the hosts view and select the three hosts by ticking the boxes to the left of the host entries.
- Click **Run Command** button. In the next screen you have to specify the ad hoc command.

Within the **Details** window, select **Module** `command`, in **Arguments** type `cat /etc/shadow` and click **Next**.

Within the **Execution Environment** window, select **Default execution environment** and click **Next**.

Within the **Machine Credential** window, select **Workshop Credentials** and click **Launch**.

> **Warning**
>
> **Expect an error!**

Oops, the last one didn't went well, all red.

Re-run the last ad hoc command but this time check the checkbox labeled **Enable privilege escalation**.

As you see, this time it worked. For tasks that have to run as `root` you need to escalate the privileges. This is the same as the **become: yes** used in your Ansible Playbooks.

Challenge Lab: Ad Hoc Commands

Okay, a small challenge: Run an ad hoc to make sure the package "tmux" is installed on all hosts. If unsure, consult the documentation either via the web UI as shown above or by running `ansible-doc yum` on the CLI of your Ansible control host.

Tip

The *Arguments* you want to provide for your desired module need to be provided as key-value-pairs separated with `=` (the same way as you would do on the command-line), multiple arguments need to be separated by *whitespace* or *comma*.

Solution

- In the Web UI go to **Resources** → **Inventories** → **Workshop Inventory**.
- Click the **Hosts** tab to change into the hosts view and select the three hosts by ticking the boxes to the left of the host entries.
- Click **Run Command** button. In the next screen you have to specify the ad hoc command.
- Within the **Details** window, select **Module** `yum`, in **Arguments** type `name=tmux`, check **Enable privilege escalation** and click **Next**.
- Within the **Execution Environment** window, select **Default execution environment** and click **Next**.
- Within the **Machine Credential** window, select **Workshop Credentials** and click **Launch**.

Info

Notice how the package was installed via the "CHANGED" output. If you run the ad hoc command a second time, the output will mention "SUCCESS" and inform you via the message parameter that there is nothing to do.

Adjust AAP settings for additional ad-hoc module

The previous Challenge Lab made use of the `yum` module, this module only makes sense on *older* Fedora-based systems like RHEL 7 or RHEL 8 hosts.
Previously, we used the generic `package` module to install packages on the target systems. Let's add this module to the list of modules which can be used with ad-hoc commands.

- In the web UI go to **Settings**. In the Tab *Jobs* click on **Jobs settings**.
- At the bottom of the page, click the **Edit** button.
- In the text area *Ansible Modules Allowed for Ad Hoc Jobs*, add `"package",` (in between `mount` and `ping` to keep the alphabatical order).

> **Warning**
>
> The text area contains a JSON list! Ensure to use quotation marks and end the line with comma to keep the valid list structure.

- After adding the module to the list, scroll down to the bottom of the page and click the **Save** button.

Now, you are able to use the *package* module in the ad-hoc command, try to do the previous Challenge Lab by using this module.

© Tim Grützmacher 2025