Mastering

ANSIBLE

Section 07

An overview of Ansible Tower

Why Ansible Tower?

- So far in this class we've been using the traditional CLI method of running Ansible. You write playbooks and other files in the YAML format, you issue ansible, ansible-playbook, or ansible-vault commands to do the job.
- This might be very adequate if you are working alone or in a small team. However, for bigger environments or large enterprise, more is needed from Ansible. For example:
  - A notification system: you need to be notified (like by e-mail or SMS) about which jobs succeeded and which failed. Additionally, some key persons should be notified when the job(s) fails. Tower allows this functionality. It can even be integrated with a web service that can trigger an action when the job fails for even a higher level of automation and quick response.
  - Deep-level reports about the current status of all jobs running at given point of time. Also an audit history of which jobs failed, when, why and the actions taken.
  - All this is controlled through a rich, graphical, and secure web interface.

Obtaining Ansible Tower

- There is more than one way of obtaining Ansible Tower, depending on your target environment and what you want to achieve with it.
- For this class we are going to use the Vagrant way. Ansible Tower has a ready-made Vagrant machine, with the product already installed.
- To have a version of Ansible Tower up and running follow these steps:

```
mkdir tower && cd tower # Create a new directory with a relevant name
vagrant init ansible/tower # Initialize Vagrant by creating a Vagrantfile
vagrant up --provider virtualbox # Download and deploy the appropriate box
vagrant ssh #log into the machine
```

- Once you log into the machine, you will find the instructions on how to open the default admin page. Open your browser and navigate to the IP address written. For example, https://10.42.0.42/
- Use the credentials provided in the instructions to access the admin console.
- On the first run, you will have to request a license. Ansible Provides a trial license with enterprise features and a free license that lacks enterprise features (like Active Directory integration, audit and others) and is limited to ten servers. This is more than sufficient giving the product a test drive. Choose the second option.
- After filling your contact information, you will receive an e-mail containing the license. The license is no more than a JSON string. Copy it and paste it in a file, say license.json. Now go back to Ansible Tower admin console, browse to the license file. You should now see the default dashboard.

How does Ansible Tower work?

- Ansible Tower runs playbooks within "jobs".
- A group of jobs is managed through a "project"
- Projects need:
  - An inventory containing groups of servers.
  - Credentials for logging in to machines and other tenants.
- One or more projects are grouped into an "organization"
- Users can be added to teams and granted the required access level to the organization as a whole and/or other items (like projects, jobs…etc.). This gives the administrator fine-grained control over who can do what.

LAB: Create a sample project on Tower

- First, we need to create a project directory to hold our playbooks. Login to the vagrant machine, switch user to awx and go to `/var/lib/awx/projects`. Create a new directory called `mastering-ansible`. Those steps can be done as follows:

```
vagrant ssh
sudo su - awx
cd /var/lib/awx/projects
mkdir mastering-ansible
cd mastering-ansible
```

- Now create `main.yml` playbook and add the following instructions:

```
---
- hosts: all
  connection: local
  tasks:
  - name: How long has this server been running?
    command: uptime
```

- In the enterprise version, you can create more than one organization. However, in the free version, you are bound to only one organization, which is the default.

- Back to the dashboard, we are going to use the default organization. Click on the gear icon to access the settings page. Choose ORGANIZATIONS. Under "Default" you'll find all the objects that are grouped by the same organization. Go to settings again, select USERS to create a new user.

- Click on +ADD button to create a new user. Notice that you have three types of user accounts. Choose "Normal User".

- Let's create a team for this user to be able to grant him permissions. Go to settings, TEAMS and click +ADD. Enter a descriptive name, say "ops" and optionally add a description. Save the team. Once done, you can add users by editing the team. Click on the pencil icon next to the team name and select the USERS button right under the team name. Add the user that you created in the previous step.

- Next, we need to create a project. From the dashboard select PROJECTS then +ADD. Name the project Mastering Ansible, optionally add a description. Ansible Tower allows you to choose among different Software Configuration Management options like Git and Subversion. For this example, choose "manual" and you'll notice that Tower automatically selects the /var/lib/awx/projects as the projects directory. For the playbook directory, select mastering-ansible.

- Now let's give the team members the necessary permissions to execute the project. Go to TEAMS, click on the pencil icon next to the ops team to edit its properties. Click on the permissions icon and click +ADD PERMISSIONS. Notice that you can select more than one level of permissions: you can give the team authority on whole projects, or on individual job templates. In this example, give the ops team "use" permissions on the project.

- The next step is to create the necessary inventory that the job will run against. Click on INVENTORIES and click the +ADD button. Type "The Local Machine" as the name of the inventory and click save. Once saved, edit it by clicking on the pencil icon and click on PERMISSIONS tab. Give the user you created the desired permissions. We'll select "use" for this example.

- Click on the name of the inventory to access additional settings. You have GROUP and HOSTS. Let's add a new group called local. Notice that you can choose the source from which this group will get its hosts list (for example, Amazon EC2 cloud provider). For this example we'll choose the manual way.

- Now we have to add a host to the group. Click +ADD HOST and enter localhost or 127.0.0.1 as the hostname.

- Ansible Tower needs to know how it is going to log in to the machine. Go to settings and click on CREDENTIALS. Click the +ADD button and enter a credential as follows: Choose a descriptive name (say Local machine credentials), set the type to Machine. Notice that Ansible tower can store credentials for different resources not just machines. We'll use the password method for logging in to the server. Enter vagrant as the username and the password. You can also specify whether or not Ansible impersonate another user (like root) and the desired method of doing so. You can add a Vault password if you are using encrypted files. Finally, you can use paste the private key string instead of pointing to the file path, which is more secure as you don't have to store the file physically on your machine, instead Ansible Tower saves it alongside other credentials data.

- Finally, we have all the necessary information to start running a job. A job is based on a template containing the instructions. Go to TEMPLATES and click +ADD. Enter the job name: uptime, Job Type: Run, Inventory: Local Machine, Project: Uptime, Playbook: main.yml, Machine Credential: Local machine credentials.

- Now, time to run the job. Go to TEMPLATES and click on the rocket icon besides the Uptime template to run the job. Observe the job's output in the window that appears.

Variable precedence

- I hope by now you've seen the power of Ansible Tower. Although we went through many steps to run just one trivial job, those steps are only done once to set the desired way of running Ansible jobs and also adjust the security and access levels of users.
- With Ansible Tower in place, a user does not even have to login to the machine on which Ansible is installed to run the job, neither does he/she have to enter any commands or memorize any options. It's all done through the GUI and with the appropriate level of access.
- However, during the lab we've seen several places where you can add variables to the playbook. With so many places to add variables, it is not unlikely that you may add two variable with the same name in two or more different places. How does Ansible deal with this situation? It has the following levels of variable precedence arranged from the least (always gets overridden) to the most (can override anything below it):

    1. Role defaults
    2. Inventory files (host_vars > group_vars > vars)
    3. Playbook group_vars
    4. Playbook host_vars
    5. Host facts
    6. Variables files > vars_prompt > vars
    7. Variables created using the register clause
    8. Variables files created inside Roles and those set by the include_vars module
    9. Variables defined in a block
    10. Variables defined in a task block
    11. Any variables passed through -e during command execution

Thank you for watching…