Ansible Assignment 3

1. Could you construct a simple playbook to install Nginx on a server?

Step 1: Generate a public SSH key and by using SSH connect to your host.

```
$ ssh-keygen
```

Step 2: Next, copy the public SSH key on your hosts. Follow the below command to do it:

```
ssh-copy-id -i root@IP address of your host
```

Step 3: List the IP addresses of your hosts/nodes in your inventory.

Follow the below command:

```
vi /etc/ansible/hosts
```

Step 4: To check if the connection has been established, let’s ping:

```
ansible -m ping ‘test-server’
```

Step 5: Create a playbook to install Nginx on the host machine. To create a playbook you just need to open a file with a yml extension, like shown below:

```
vi <Name of your file>.yml
```

In an Ansible Playbook, the tasks are defined as a list of dictionaries and are executed from top to bottom.

Each task is defined as a dictionary that can have several keys, such as “name” or “sudo” which signify the name of the task and whether it requires sudo privileges.

A variable server_port is set that listens on TCP port 8080 for incoming requests.

Here, the first task is to get the necessary package for installation of Nginx and then install it. Internally, Ansible will check if the directory exists and create it if it’s not, otherwise, it will do nothing.

The next task is to configure Nginx. In Nginx, contexts contain configuration details.

Here, the template is a file you can deploy on hosts. However, template files also include some reference variables which are pulled from variables defined as part of an Ansible playbook or facts gathered from the hosts. Facts containing the configuration details are being pulled from a source directory and being copied to a destination directory.

Handlers here define the action to be performed only upon notification of tasks or state changes. In this playbook, we defined, notify: restart Nginx handler which will restart Nginx once the files and templates are copied to hosts.

Now, save the file and exit.

Step 6: Run the playbook, using the command below:

```
ansible-playbook <name of your file>.yml
```

Step 7: Check if Nginx is installed on the machine. Use the following command:

```
ps waux | grep nginx
```

2. What is the significance of the '' notation? And how may variables or dynamic variable names be interpolated?

One basic rule is to ‘always use {{}} except when:’. Conditionals are always run through Jinja2 as to resolve the expression. Therefore, ‘when:failed_when:’ and ‘changed_when:’ are always templated and we should avoid adding {{}}.
In other cases, except when clause, we have to use brackets, otherwise, differentiating between an undefined variable and a string will be difficult to do.

3. What is the difference between an Ansible role and a playbook role?
Ansible Roles is basically another level of abstraction used to organize playbooks. They provide a skeleton for an independent and reusable collection of variables, tasks,

templates, files, and modules which can be automatically loaded into the playbook. Playbooks are a collection of roles. Every role has specific functionality.

Let’s understand the difference between Ansible roles and playbook with an example.

For example if want your playbook to perform 10 different tasks on 5 different systems, would you use a single playbook for this? No, using a single playbook can make it confusing and prone to blunders. Instead, you can create 10 different roles, where each role will perform one task. Then, all you need to do is, mention the name of the role inside the playbook to call them.

3. How can I write a multi-task Ansible handler in Ansible?

If we want to create a handler that restarts a service only if it is already running.

Handlers can “listen” to generic topics, and tasks can notify those topics as shown below. This functionality makes it much easier to trigger multiple handlers. It also decouples handlers from their names, making it easier to share handlers among playbooks and roles:

```
- name: Check if restarted
shell: check_is_started.sh
register: result
listen: Restart processes

- name: Restart conditionally step 2
service: name=service state=restarted
when: result
listen: Restart processes
```

5. What are Ansible Vaults and how do you use them?

Ansible Vault is a feature that allows you to keep all your secrets safe. It can encrypt entire files, entire YAML playbooks or even a few variables. It provides a facility where you can not only encrypt sensitive data but also integrate them into your playbooks.

Vault is implemented with file-level granularity where the files are either entirely encrypted or entirely unencrypted. It uses the same password for encrypting as well as for decrypting files which makes using Ansible Vault very user-friendly.

6. How can I use Ansible to create encrypted files?

To create an encrypted file, use the ‘ansible-vault create’ command and pass the filename.

```
$ ansible-vault create filename.yaml
```

You’ll be prompted to create a password and then confirm it by re-typing it.

Once your password is confirmed, a new file will be created and will open an editing window. By default, the editor for Ansible Vault is vi. You can add data, save and exit.

7. What is Ansible Tower, exactly?

Ansible Tower is an enterprise-level solution by RedHat. It provides a web-based console and REST API to manage Ansible across teams in an organization. There are many features such as

- Workflow Editor - We can set up different dependencies among playbooks, or running multiple playbooks maintained by different teams at once
- Real-Time Analysis - The status of any play or tasks can be monitored easily and we can check what’s going to run next
- Audit Trail - Tracking logs are very important so that we can quickly revert back to a functional state if something bad happens.
- Execute Commands Remotely - We can use the tower to run any command to a host or group of hosts in our inventory.

There are other features also such as Job Scheduling, Notification Integration, CLI, etc.

8. What are the benefits of the Ansible Tower?

- Ansible Tower Dashboard – The Ansible Tower dashboard displays everything going on in your Ansible environment like the hosts, inventory status, the recent job activity and so on.
- Real-Time Job Updates – As Ansible can automate the complete infrastructure, you can see real-time job updates, like plays and tasks broken down by each machine either been successful or a failure. So, with this, you can see the status of your automation, and know what’s next in the queue.
- Multi-Playbook Workflows – This feature allows you to chain any number of playbooks, regardless of the usage of different inventories, utilizes various credentials, or runs different users.

- Who Ran What Job When – As the name suggests, you can easily know who ran what job where and when as, all the automation activity is securely logged in Ansible Tower.
- Scale Capacity With Clusters – We can connect multiple Ansible Tower nodes into an Ansible Tower cluster as the clusters add redundancy and capacity, which allow you to scale Ansible automation across the enterprise.
- Integrated Notifications – This feature lets you notify a person or team when a job succeeds or fails across the entire organization at once, or customize on a per-job basis.
- Schedule Ansible Jobs – Different kinds of jobs such as Playbook runs, cloud inventory updates, and source control updates can be scheduled inside Ansible Tower to run according to the need.
- Manage & Track Inventory – Ansible Tower helps you manage your entire infrastructure by letting you easily pull inventory from public cloud providers such as Amazon Web Services, Microsoft Azure, and more.
- Self-Service – This feature of Ansible Tower lets you launch Playbooks with just a single click. It can also, let you choose from available secure credentials or prompt you for variables and monitor the resulting deployments.
- REST API & Tower CLI Tool – Every feature present in Ansible Tower is available via Ansible Tower's REST API, which provides the ideal API for a systems management infrastructure. The Ansible Tower's CLI tool is available for launching jobs from CI systems such as Jenkins, or when you need to integrate with other command-line tools.
- Remote Command Execution – You can run simple tasks such as add users, restart any malfunctioning service, reset passwords on any host or group of hosts in the inventory with Ansible Tower's remote command execution.

9. What is the role of Ansible in the Continuous Delivery pipeline? Explain.

It is well known that in DevOps development and operations work is integrated. This integration is very important for modern test-driven applications. Hence, Ansible integrates this by providing a stable environment to both development and operations resulting in a smooth delivery pipeline.

When developers begin to think of infrastructure as part of their application i.e as Infrastructure as code (IaC), stability and performance become normative. Infrastructure as Code is the process of managing and provisioning computing infrastructure and their configuration through machine-processable definition files, rather than physical hardware configuration or the use of interactive configuration tools. This is where Ansible automation plays a major role and stands out among its peers.

In a Continuous Delivery pipeline, Sysadmins work tightly with developers, development velocity is improved, and more time is spent doing activities like performance tuning, experimenting, and getting things done, and less time is spent fixing problems.

10. Using Ansible, how do you build a LAMP stack and deploy a webpage?

If we are trying to deploy a website on 30 systems, every website deployment will require a base OS, web-server, Database, and PHP. We use ansible playbook to install these prerequisites on all 30 systems at once.

For this particular problem statement, you can use two virtual machines, one as a server where Ansible is installed and the other machine acts as the remote host. Also, I've created a simple static webpage saved in a folder index which has two files, index.html, and style.css.

In the below code I've created a single Ansible playbook to install Apache, MySql, and PHP:

```
---
# Setup LAMP Stack
-  hosts: host1
   tasks:

      -  name: Add ppa repository
         become: yes
         apt_repository: repo=ppa:ondrej/php

      -  name: Install lamp stack
         become: yes
         apt:
            pkg:
              - apache2
              - mysql-server
              - php7.0
              - php7.0-mysql
            state: present
            update cache: yes

      -  name: start apache server
          become: yes
```

```
        service:
            name: apache2
             state: started
              enabled: yes

    -  name: start mysql service
        become: yes
        services:
           name: mysql
           state: started
           enabled: yes

    -  name:  create target directory
        file: path=/var/www/html state=directory mode=0755

    - name:  deploy index.html
       became: yes
        copy:
            src: /etc/ansible/index/index.html
            dest: var/www/html/index/index.html
```

There are 6 main tasks, each task performs a specific function:

- The first task adds the repository required to install MySQL and PHP.
- The second task installs apache2, MySQL-server, PHP, and PHP-MySQL.
- The third and fourth task starts the Apache and MySQL service.
- The fifth task creates a target directory in the host machine and
- Finally, the sixth task executes the index.html file, it picks up the file from the server machine and copies it into the host machine.

To finally run this playbook you can use the following command:

```
$ ansible-playbook lamp.yml -K
```