# Ansible Assignment 2

1. How can a list of hosts in a group be looped over within a template?

This can be done by accessing the “$groups” dictionary in the template, like:

```
{% for host in groups['db_servers'] %}
{{ host }}
{% endfor %}
```

If we need to access facts also we need to make sure that the facts have been populated. For instance, a play that talks to db_servers:

```
- hosts: db_servers
tasks:
- debug: msg="Something to debug"
```

Now, this can be used within a template, like so:

```
{% for host in groups['db_servers'] %}
{{ hostvars[host]['ansible_eth0']['ipv4']['address'] }}
{% endfor %}.
```

2. What is Ansible's ad-hoc command?

Ad-hoc commands are like one-line playbooks to perform a specific task only. The syntax for the ad-hoc command is

```
ansible [pattern] -m [module] -a "[module options]"
```

For example, we need to reboot all servers in the staging group

```
ansible atlanta -a "/sbin/reboot"  -u username --become [--ask-become-pass]
```

3. How do I set up Nginx using the Ansible playbook?

```
- hosts: stagingwebservers
 gather_facts: False
 vars:
  - server_port: 8080
 tasks:
```

```
  - name: install nginx
    apt: pkg=nginx state=installed update_cache=true
  - name: serve nginx config
     template: src=../files/flask.conf dest=/etc/nginx/conf.d/
     notify:
     - restart nginx
 handlers:
   - name: restart nginx
     service: name=nginx state=restarted
   - name: restart flask app
     service: name=flask-demo state=restarted
…
```

In the above playbook, we are fetching all hosts of stagingwebservers group for executing these tasks. The first task is to install Nginx and then configure it. We are also taking a flask server for reference. In the end, we also defined handlers so that in case the state changes it will restart Nginx. After executing the above playbook we can verify whether Nginx is installed or not.

4. How do I programmatically access the name of a variable?

By this command:
{{ hostvars[inventory_hostname]['ansible_' + which_interface]['ipv4']['address'] }}

5. How do Ansible and Puppet vary from one other?

**Management and Scheduling:** In Ansible, the server pushes the configuration to the nodes on the other hand in puppet, the client pulls the configuration from the server. Also for scheduling, the puppet has an agent who polls every 30mins(default settings) to make sure all nodes are in a desirable state. Ansible doesn't have that feature in the free version.
**Availability:** Ansible has backup secondary nodes and puppet has more than one master node. So both try to be highly available.
Setup: Puppet is considered to be harder to set up than ansible as it has a client-server architecture and also there's a specific language called Puppet DSL which is its own declarative language.

6. What is the purpose of Ansible Tower, and what are its characteristics?

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.

7. Describe how you'll recursively copy files to a destination host.

There's a copy module that has a recursive parameter in it but there's something called synchronize which is more efficient for large numbers of files.

Example:

```
- synchronize:
   src: /first/absolute/path
   dest: /second/absolute/path
   delegate_to: "{{ inventory_hostname }}"
```

8. What is the most effective method for making content reusable and redistributable?

To make content reusable and redistributable Ansible roles can be used. Ansible roles are basically a level of abstraction to organize playbooks. For example, if we need to execute 10 tasks on 5 systems, writing all of them in the playbook might lead to blunders and confusion. Instead we create 10 roles and call them inside the playbook.

9. What are handlers, and what do they do?

Handlers are like special tasks which only run if the Task contains a "notify" directive.

```
tasks:
  - name: install nginx
    apt: pkg=nginx state=installed update_cache=true
    notify:
```

```
    - start nginx
 handlers:
   - name: start nginx
     service: name=nginx state=started
```

In the above example after installing NGINX we are starting the server using a `start nginx` handler

10. How can a user module generate encrypted passwords?

Ansible has a very simple ad-hoc command for this

```
ansible all -i localhost, -m debug -a "msg={{ 'mypassword' | password_hash('sha512', 'mysecretsalt') }}"
```

We can also use the Passlib library of Python, e.g

```
python -c "from passlib.hash import sha512_crypt; import getpass; print(sha512_crypt.using(rounds=5000).hash(getpass.getpass()))"
```

On top of this, we should also avoid storing raw passwords in playbook or host_vars, instead, we should use integrated methods to generate a hash version of a password.

11. What is the difference between dot notation and array notation for variables?

Dot notation works fine unless we stump upon few special cases such as

- If the variable contains a dot(.), colon(:), starting or ending with an underscore or any known public attribute.
- If there's a collision between methods and attributes of python dictionaries.
- Array notation also allows for dynamic variable composition.

12. What is the purpose of the Ansible synchronize module?

Ansible synchronize is a module similar to rsync in Linux machines which we can use in playbooks. The features are similar to rsync such as archive, compress, delete, etc but there are few limitations also such as

- Rsync must be installed on both source and target systems
- Need to specify delegate_to to change the source from localhost to some other port
- Need to handle user permission as files are accessible as per remote user.
- We should always give the full path of the destination host location in case we use sudo otherwise files will be copied to the remote user home directory.

- Linux rsync limitations related to hard links are also applied here.
- It forces -delay-updates to avoid the broken state in case of connection failure

13. What is the purpose of the Ansible firewalld module?

Ansible firewalld is used to manage firewall rules on host machines. This works just as Linux firewalld daemon for allowing/blocking services from the port. It is split into two major concepts

- **Zones**: This is the location for which we can control which services are exposed to or a location to which one the local network interface is connected.
- **Services**: These are typically a series of port/protocol combinations (sockets) that your host may be listening on, which can then be placed in one or more zones.

14. What distinguishes the Ansible set fact module from vars, vars file, and include var?

In Ansible, set_fact is used to set new variable values on a host-by-host basis which is just like ansible facts, discovered by the setup module. These variables are available to subsequent plays in a playbook. In the case of vars, vars_file, or include_var we know the value beforehand whereas when using set_fact, we can store the value after preparing it on the fly using certain tasks like using filters or taking subparts of another variable. We can also set a fact cache over it.

set_fact variable assignment is done by using key-pair values where the key is the variable name and the value is the assignment to it. A simple example will be like below

```
- set_fact:
one_fact: value1
second_fact:
value2
```

15. When is it risky to use a variable to bulk-set task arguments?

All of the task's arguments can be dictionary-typed variables which can be useful in some dynamic execution scenarios also. However, Ansible issues a warning since it introduces a security risk.

```
vars:
 usermod_args:
name: testuser
state: present
```

update_password: always
tasks:
- user: '{{ usermod_args }}'

In the above example, the values passed to the variable usermod_args could be overwritten by some other malicious values in the host facts on a compromised target machine. To avoid this

- bulk variable precedence should be greater than host facts.
- need to disable INJECT_FACTS_AS_VARS configuration to avoid collision of fact values with variables.