Basic Ansible Interview Questions

1. What is CI/CD?

Continuous Integration is something that is used for streamlining the development and deployment process. These lead to the more rapid development of cohesive software.

Continuous Delivery is on the other hand is a process where your code after being pushed to a remote repository can be taken to production at any time.

In the above diagram our integration test and unit test are performed without any manual intervention and after UAT we just needed the approval to ship our tested features to production and to make such a process we need CI/CD.

2. What is Configuration Management?

It’s a practice that we should follow in order to keep track of all updates that are going into the system over a period of time. This also helps in a situation where a major bug has been introduced to the system due to some new changes and we need to fix it with minimum downtime. Instead of fixing the bug, we can roll back the new changes(which caused this bug) as we have been tracking those.

3. How does Ansible work?

Ansible is a combination of multiple pieces working together to become an automation tool. Mainly these are modules, playbooks, and plugins.

- Modules are small codes that will get executed. There are multiple inbuilt modules that serve as a starting point for building tasks.
- Playbooks contain plays which further is a group of tasks. This is the place to define the workflow or the steps needed to complete a process.

- Plugins are special kinds of modules that run on the main control machine for logging purposes. There are other types of plugins also.

The playbooks ran via an Ansible automation engine. These playbooks contain modules that are basically actions that run in host machines. The mechanism is followed here is the push mechanism, so ansible pushes small programs to these host machines which are written to be resource models of the desired state of the system.

---

4. What are the features of Ansible?

It has the following features:

- Agentless – Unlike puppet or chef there is no software or agent managing the nodes.
- Python – Built on top of python which is very easy to learn and write scripts and one of the robust programming languages.
- SSH – Passwordless network authentication which makes it more secure and easy to set up.
- Push architecture – The core concept is to push multiple small codes to the configure and run the action on client nodes.
- Setup – This is very easy to set up with a very low learning curve and any open source so that anyone can get hands-on.
- Manage Inventory – Machines' addresses are stored in a simple text format and we can add different sources of truth to pull the list using plugins such as Openstack, Rackspace, etc.

5. Explain Infrastructure as Code?

Infrastructure as Code or IaC is a process that DevOps teams should follow to have a more organized way of managing the infra. Instead of some throwaway scripts or manually configuring any cloud component, there should be a code repo where all of these will lie and any change in configuration should be done through it. It is wise to put it under source control also. This improves speed, consistency, and accountability.

6. What is Ansible Galaxy?

Galaxy is a repository of Ansible roles that can be shared among users and can be directly dropped into playbooks for execution. It is also used for the distribution of packages containing roles, plugins, and modules also known as collection. The ansible-galaxy-collection command implements similar to init, build, install, etc like an ansible-galaxy command.

7. Explain Ansible modules in detail?

Ansible modules are like functions or standalone scripts which run specific tasks idempotently. The return value of these are JSON string in stdout and input depends on the type of module. These are used by Ansible playbooks.

There are 2 types of modules in Ansible:

- Core Modules

The core Ansible team is responsible for maintaining these modules thus these come with Ansible itself. The issues reported are fixed on priority than those in the “extras” repo.

- Extras Modules

The Ansible community maintains these modules so, for now, these are being shipped with Ansible but they might get discontinued in the future. These can be used but if there are any feature requests or issues they will be updated on low priority.

Now popular extra modules might enter into the core modules anytime. You may find these separate repos for these modules as ansible-modules-core and ansible-modules-extra respectively.

8. What is a YAML file and how do we use it in Ansible?

YAML or files are like any formatted text file with few sets of rules just like JSON or XML. Ansible uses this syntax for playbooks as it is more readable than other formats.

An example of JSON vs YAML is:

```
{
 "object": {
"key": "value",
"array": [
  {
    "null_value": null
  },
  {
    "boolean": true
  },
  {
    "integer": 1
  },
  {
    "alias": "aliases are like variables"
  }
]
 }
}
---
object:
 key: value
 array:
 - null_value:
 - boolean: true
 - integer: 1
 - alias: aliases are like variables
```

9. What are Ansible tasks?

The task is a unit action of Ansible. It helps by breaking a configuration policy into smaller files or blocks of code. These blocks can be used in automating a process. For example, to install a package or update a software

```
Install <package_name>, update <software_name>.
```

10. How to use YAML files in high programming languages such as JAVA, Python, etc?

YAML is supported in most programming languages and can be easily integrated with user programs.

In JAVA we can use the Jackson module which also parses XML and JSON. For e.g

```
// We need to declare Topic class with necessary attributes such as
name, total_score, user_score, sub_topics
List<Topic> topics = new ArrayList<Topic>();
topics.add(new Topic("String Manipulation", 10, 6));
topics.add(new Topic("Knapsack", 5, 5));
topics.add(new Topic("Sorting", 20, 13));
// We want to save this Topic in a YAML file
Topic topic = new Topic("DS & Algo", 35, 24, topics);
// ObjectMapper is instantiated just like before
ObjectMapper om = new ObjectMapper(new YAMLFactory());
// We write the `topic` into `topic.yaml`
om.writeValue(new File("/src/main/resources/topics.yaml"), topic);
```

```
---
name: "DS & Algo"
total_score: 35
user_score: 24
sub_topics:
- name: "String Manipulation"
  total_score: 10
  user_score: 6
- name: "Knapsack"
  total_score: 5
  user_score: 5
- name: "Sorting"
  total_score: 20
  user_score: 13
```

Similarly, we can read from YAML also:

```
// Loading the YAML file from the /resources folder
ClassLoader classLoader =
Thread.currentThread().getContextClassLoader();
```

```
File file = new
File(classLoader.getResource("topic.yaml").getFile());
// Instantiating a new ObjectMapper as a YAMLFactory
ObjectMapper om = new ObjectMapper(new YAMLFactory());
// Mapping the employee from the YAML file to the Employee class
Topic topic = om.readValue(file, Topic.class);
```

In python similarly, we can use the pyyaml library and read and write easily in YAML format.

Intermediate Ansible Interview Questions

11. How to setup a jump host to access servers having no direct access?

First, we need to set a ProxyCommand in ansible_ssh_common_args inventory variable, since any arguments specified in this variable are added to the sftp/scp/ssh command line when connecting to the relevant host(s). For example:

```
[gatewayed]
staging1 ansible_host=10.0.2.1
staging2 ansible_host=10.0.2.2
```

To create a jump host for these we need to add a command in ansible_ssh_common_args.

```
ansible_ssh_common_args: '-o ProxyCommand="ssh -W %h:%p -q
user@gateway.example.com"'
```

In this way whenever we will try to connect to any host in the gatewayed group ansible will append these arguments to the command line.

12. How to automate the password input in playbook using encrypted files?

To automate password input we can have a password file for all the passwords of encrypted files will be saved and ansible can make a call to fetch those when required.

```
ansible_ssh_common_args: '-o ProxyCommand="ssh -W %h:%p -q
user@gateway.example.com"' .
```

This can also be achieved by having a separate script that specifies the passwords. But in this case, we need to print a password to stdout to work without annoying errors.

```
ansible-playbook launch.yml --vault-password-file ~/ .vault_pass.py .
```

13. What are callback plugins in Ansible?

Callback plugins basically control most of the output we see while running cmd programs. But it can also be used to add additional output. For example log_plays callback is used to record playbook events to a log file, and mail callback is used to send email on playbook failures. We can also add custom callback plugins by dropping them into a callback_plugins directory adjacent to play, inside a role, or by putting it in one of the callback directory sources configured in ansible.cfg.

14. What is Ansible Inventory and its types?

In Ansible, there are two types of inventory files: Static and Dynamic.

- Static inventory file is a list of managed hosts declared under a host group using either hostnames or IP addresses in a plain text file. The managed host entries are listed below the group name in each line. For example

```
[gatewayed] .
staging1 ansible_host=10.0.2.1  .
staging2 ansible_host=10.0.2.2   .
```

- Dynamic inventory is generated by a script written in Python or any other programming language or by using plugins(preferable). In a cloud setup, static inventory file configuration will fail since IP addresses change once a virtual server is stopped and started again. We create a demo_aws_ec2.yaml file for the config such as:

```
plugin: aws_ec2 regions:
ap-south-1 filters:
tag:tagtype: testing
```

Now we can fetch using this command

```
ansible-inventory -i demo_aws_ec2.yaml -graph
```

15. What is Ansible Vault?

Ansible vault is used to keep sensitive data such as passwords instead of placing it as plaintext in playbooks or roles. Any structured data file or any single value inside the YAML file can be encrypted by Ansible.

To encrypt a file:

```
ansible-vault encrypt foo.yml bar.yml baz.yml    .
```

And similarly to decrypt:

```
ansible-vault decrypt foo.yml bar.yml baz.yml    .
```

16. How can looping be done over a list of hosts in a group, inside of a template?

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

```
{% 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 %}.
```

17. What is the ad-hoc command in Ansible?

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]
```

18. Install Nginx using Ansible playbook?

The playbook file would be:

```
- 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.

```
ps waux | grep nginx
```

19. How do I access a variable name programmatically?

Variable names can be built by adding strings together. For example, if we need to get ipv4 address of an arbitrary interface, where the interface to be used may be supplied via a role parameter or other input, we can do it in this way.

```
{{ hostvars[inventory_hostname]['ansible_' +
which_interface]['ipv4']['address'] }}
```

20. What is the difference between Ansible and Puppet?

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.

21. What is Ansible Tower and what are its features?

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.

22. Explain how you will copy files recursively onto a target 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.

For example:

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

23. What is the best way to make Content Reusable/ 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.

24. What are handlers?

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.

25. How to generate encrypted passwords for a user module?

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.

26. How does dot notation and array notation of variables are different?

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.

Advanced Ansible Interview Questions

27. How does Ansible synchronize module works?

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

An example of synchronize module is

```
---
- hosts: host-remote tasks:
- name: sync from sync_folder
synchronize:
src: /var/tmp/sync_folder dest: /var/tmp/
```

Here we are transferring files of /var/tmp/sync_folder folder to remote machine's /var/tmp folder

28. How does the Ansible firewalld module work?

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

Few examples of setting up firewalld are

```
- name: permit traffic in default zone for https service
 ansible.posix.firewalld:
   service: https
   permanent: yes
   state: enabled

- name: do not permit traffic in default zone on port 8081/tcp
 ansible.posix.firewalld:
   port: 8081/tcp
   permanent: yes
   state: disabled
```

29. How is the Ansible set_fact module different from vars, vars_file, or 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
```

30. When is it unsafe to bulk-set task arguments from a variable?

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.

31. Explain Ansible register.

Ansible register is used to store the output from task execution in a variable. This is useful when we have different outputs from each remote host. The register value is valid throughout the playbook execution so we can make use of set_fact to manipulate the data and provide input to other tasks accordingly.

```
- hosts: all tasks:
name: find all txt files in /home shell: "find /home -name
*.txt" register: find_txt_files
debug:
var: find_txt_files
```

In the above example, we are searching for all .txt files in the remote host's home folder and then capturing it in find_txt_files and displaying that variable.

32. How can we delegate tasks in Ansible?

Task delegation is an important feature of Ansible since there might be use cases where we would want to perform a task on one host with reference to other hosts. We can do this using the delegate_to keyword.

For example, if we want to manage nodes in a load balancer pool we can do:

```
- hosts: webservers
 serial: 5

 tasks:
- name: Take machine out of ELB pool
  ansible.builtin.command: /usr/bin/take_out_of_pool {{
inventory_hostname }}
  delegate_to: 127.0.0.1

- name: Actual steps would go here
  ansible.builtin.yum:
    name: acme-web-stack
    state: latest

- name: Add machine back to ELB pool
  ansible.builtin.command: /usr/bin/add_back_to_pool {{
inventory_hostname }}
  delegate_to: 127.0.0.1
```

We are also defining serial to control the number of hosts executing at one time. There is another shorthand syntax called local_action which can be used instead of delegate_to.

```
...
tasks:
   - name: Take machine out of ELB pool
     local_action: ansible.builtin.command
/usr/bin/take_out_of_pool {{ inventory_hostname }}
...
```

But there are few exceptions also such as include, add_host, and debug tasks that cannot be delegated.

SECOND SET OF ANSIBLE QUESTIONS.

1. Let's begin with the basics. What is Ansible?

Ansible is an open-source platform that facilitates configuration management, task automation, or application deployment. It is a valuable DevOps tool. It was written in Python and powered by Red Hat. It uses SSH to deploy SSH without incurring any downtime.

2. List Ansible's advantages

Ansible has many strengths, including:

- It's agentless and only requires SSH service running on the target machines
- Python is the only required dependency and, fortunately, most systems come with the language pre-installed
- It requires minimal resources, so there's low overhead
- It's easy to learn and understand since Ansible tasks are written in YAML.
- Unlike other tools, most of which are Procedural, ansible is declarative; define the desired state, and Ansible fulfills the requirements needed to achieve it

3. What are CD and CI, and what is Ansible's relationship with them?

CD stands for continuous delivery, and CI stands for continuous integration; both are software development practices.

In CD, developers build software that can be released into production at any given time. CI, on the other hand, consists of each developer uploading regularly scheduled integrations (usually daily), resulting in multiple integrations every day. Ansible is an ideal tool for CI/CD processes, providing a stable infrastructure for provisioning the target environment and then deploying the application to it.

4. Describe how Ansible works.

This is one of the most frequently asked ansible interview questions where the interviewer wants to know whether you actually know the tool in and out or not. You can start this way - ansible is broken down into two types of servers: controlling machines and nodes. Ansible is installed on the controlling computer, and the controlling machines manage the nodes via SSH.

The controlling machine contains an inventory file that holds the node system's location. Ansible runs the playbook on the controlling machine to deploy the modules on the node systems. Since Ansible is agentless, there's no need for a third-party tool to connect the nodes.

5. State the requirements for the Ansible server.

You need a virtual machine with Linux installed on it, running with Python version 2.6 or higher.

6. Explain what a "playbook" is.

A playbook has a series of YAML-based files that send commands to remote computers via scripts. Developers can configure entire complex environments by passing a script to the required systems rather than using individual commands to configure computers from the command line remotely. Playbooks are one of Ansible's strongest selling points and often referred to as the tool's building blocks.

7. How do you set up Ansible?

You can use either the Python installer or a Linux-based installation process, such as apt or yum.

8. What is Ansible Tower?

It's an enterprise-level web-based solution that increases Ansible's accessibility to other IT teams by including an easy-to-use UI (user interface). Tower's primary function is to serve as the hub for all of an organization's automation tasks, allowing users to monitor configurations and conduct rapid deployments.

Next, let us look at the intermediate-level Ansible interview questions.

Intermediate Ansible Interview Questions

9. What is “idempotency”?

idempotency is an important Ansible feature. It prevents unnecessary changes in the managed hosts. With idempotency, you can execute one or more tasks on a server as many times as you need to, but it won’t change anything that’s already been modified and is working correctly. To put it in basic terms, the only changes added are the ones needed and not already in place.

10. What is Ansible Galaxy?

This is a tool bundled with Ansible to create a base directory structure. Galaxy is a website that lets users find and share Ansible content. You can use this command to download roles from the website:

```
$ ansible-galaxy install username.role_name
```

11. How do you use Ansible to create encrypted files?

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

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

You will get a prompt to create a password, and then to type it again for confirmation. You will now have access to a new file, where you can add and edit data.

12. What are “facts” in the context of Ansible?

Facts are newly discovered and known system variables, found in the playbooks, used mostly for implementing conditionals executions. Additionally, they gather ad-hoc system information.

You can get all the facts by using this command:

```
$ ansible all- m setup
```

13. Explain what an ask_pass module is.

It's a playbook control module used to control a password prompt. It's set to True by default.

14. What's an ad hoc command?

Users initiate ad hoc commands to initiate actions on a host without using a playbook. Consider it a one-shot command.

15. Explain the difference between a playbook and a play.

A play is a set of tasks that run on one or more managed hosts. Plays consist of one or more tasks. A playbook consists of one or more plays.

16. What exactly is a configuration management tool?

Configuration management tools help keep a system running within the desired parameters. They help reduce deployment time and substantially reduce the effort required to perform repetitive tasks. Popular configuration management tools on the market today include Chef, Puppet, Salt, and of course, Ansible.

Finally, let us go through the Ansible interview questions at an advanced level.

Advanced Ansible Interview Questions For Experienced Professionals

17. What are tags?

When there's an extensive playbook involved, sometimes it's more expedient to run just a part of it as opposed to the entire thing. That's what tags are for.

18. Speaking of tags, how do you filter out tasks?

You can filter out tasks in one of two ways:

- Use –tags or –skip-tags options on the command line
- If you're in Ansible configuration settings, use the TAGS_RUN and TAGS_SKIP options.

19. What's a handler?

In Ansible, a handler is similar to a regular task in a playbook, but it will only run if a task alerts the handler. Handlers are automatically loaded by roles/<role_name>/handlers/main.yaml. Handlers will run once, after all of the tasks are completed in a particular play.

20. How do you test Ansible projects?

This is another frequently asked ansible interview question. Try elaborating the answer to this question rather than just answering the testing methods in one word. There are three testing methods available:

1. Asserts
   Asserts duplicates how the test runs in other languages like Python. It verifies that your system has reached the actual intended state, not just as a simulation that you'd find in check mode. Asserts shows that the task did the job it was supposed to do and changed the appropriate resources.

2. Check Mode
   Check mode shows you how everything would run if no simulation was done. Therefore, you can easily see if the project behaves the way you want it to. On the downside, check mode doesn't run scripts and commands used in roles and playbooks. To get around this, you have to disable check mode for specific tasks by running "check_mode: no."
3. Manual Run
   Just run the play and verify that the system is in its desired state. This testing choice is the easiest method, but it carries an increased risk because the results in a test environment may not be the same in a production
4.

21. How do you upgrade Ansible?

Upgrading Ansible is easy. Just use this command: sudo pip install ansible==<version-number>

22. When do you use {{ }}?

One of Ansible's most basic rules is: "Always use {{ }} except when:"

23. Explain how to access shell environment variables.

You can access the controlling machine's existing variables by using the "env" lookup plugin. For instance, to access the value of the management machine's home environment variable, you'd enter:

local_home:"{{lookup('env','HOME')}}"

24. How do you keep data secret in a playbook?

If you want to keep secret data but still be able to share it publicly, then use Vault in playbooks. But if you're using –v (verbose) mode and don't want anyone to see the results, then use:

name: secret task

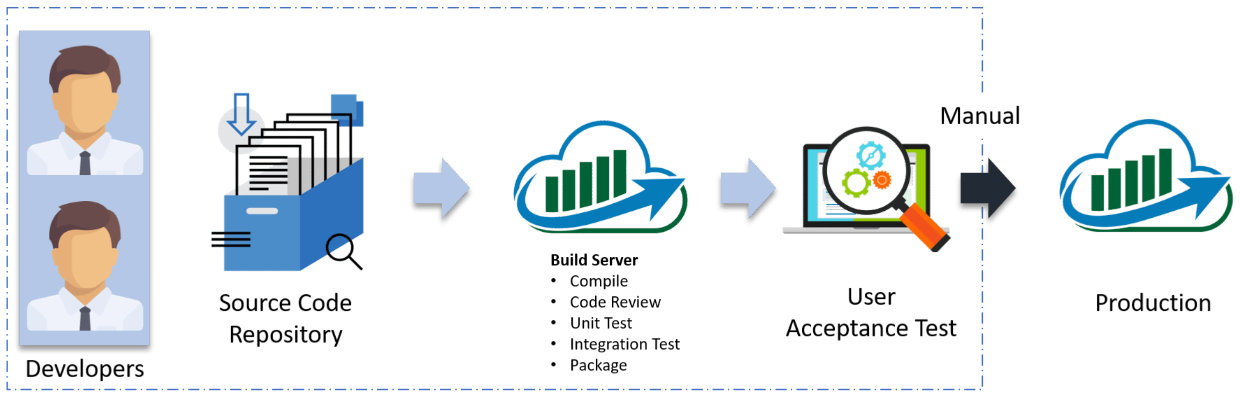
```
shell: /usr/bin/do_something --value={{ secret_value }}

no_log: True
```

THIRD SET OF ANSIBLE INTERVIEW QUESTIONS.

Q1. What is CI/CD?

Continuous Integration is a software development practice where members of a team integrate their work frequently, usually, each person integrates at least daily leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible. Many teams find that this approach leads to significantly reduced integration problems and allows a team to develop cohesive software more rapidly.

Continuous Delivery is a process where you build software in such a way that it can be released to production at any time. Consider the diagram below:



Let me explain the above diagram:

- Automated build scripts will detect changes in Source Code Management (SCM) like Git.
- Once the change is detected, source code would be deployed to a dedicated build server to make sure build is not failing and all test classes and integration tests are running fine.
- Then, the build application is deployed on the test servers (pre-production servers) for User Acceptance Test (UAT).
- Finally, the application is manually deployed on the production servers for release.

Q2. What is Configuration Management and how does it help an organization?

Configuration Management is the practice of handling updates and changes systematically so that a system maintains its integrity over time. Configuration Management (CM) keeps a track of all the updates that are needed in a system and it ensures that the current design and build state of the system is up to date and functioning correctly.

Configuration Management can help an organization by overcoming the following challenges:

- Finding out what changes need to be implemented when user requirements change.
- Redoing and updating an implementation due to change in the requirements since the last implementation.
- Reverting to an older version of the component because the latest version is flawed.
- Replacing the wrong component because you couldn’t accurately determine which component needed replacing.

To better understand this consider the NYSE example:

The New York Stock Exchange (NYSE) encountered a glitch in their software which prevented them from trading stocks for approx 90 minutes. On the night before, new software was installed on 8 of its 20 trading terminals. Unfortunately, the software failed to operate properly on the 8 terminals.

Therefore, by using Configuration Management tools such as Ansible and Puppet, they reverted back to the old software. Had they not implemented CM, they would've taken a much longer time to fix the issue which would lead to a much bigger loss.

Q3. What is Ansible and what makes it stand out from the rest of the Configuration Management tools?

Ansible is an open source IT Configuration Management, Deployment & Orchestration tool. It aims to provide large productivity gains to a wide variety of automation challenges.

Here's a list of features that makes Ansible such an effective Configuration Management and Automation tool:

1. Simple: Uses a simple syntax written in YAML called playbooks.
2. Agentless: No agents/software or additional firewall ports that you need to install on the client systems or hosts which you want to automate.
3. Powerful and Flexible: Ansible's capabilities allow you to orchestrate the entire application environment regardless of where it is deployed.
4. Efficient: Ansible introduces modules as basic building blocks for your software. So, you can even customize it as per your needs.

Q4. How does Ansible work?

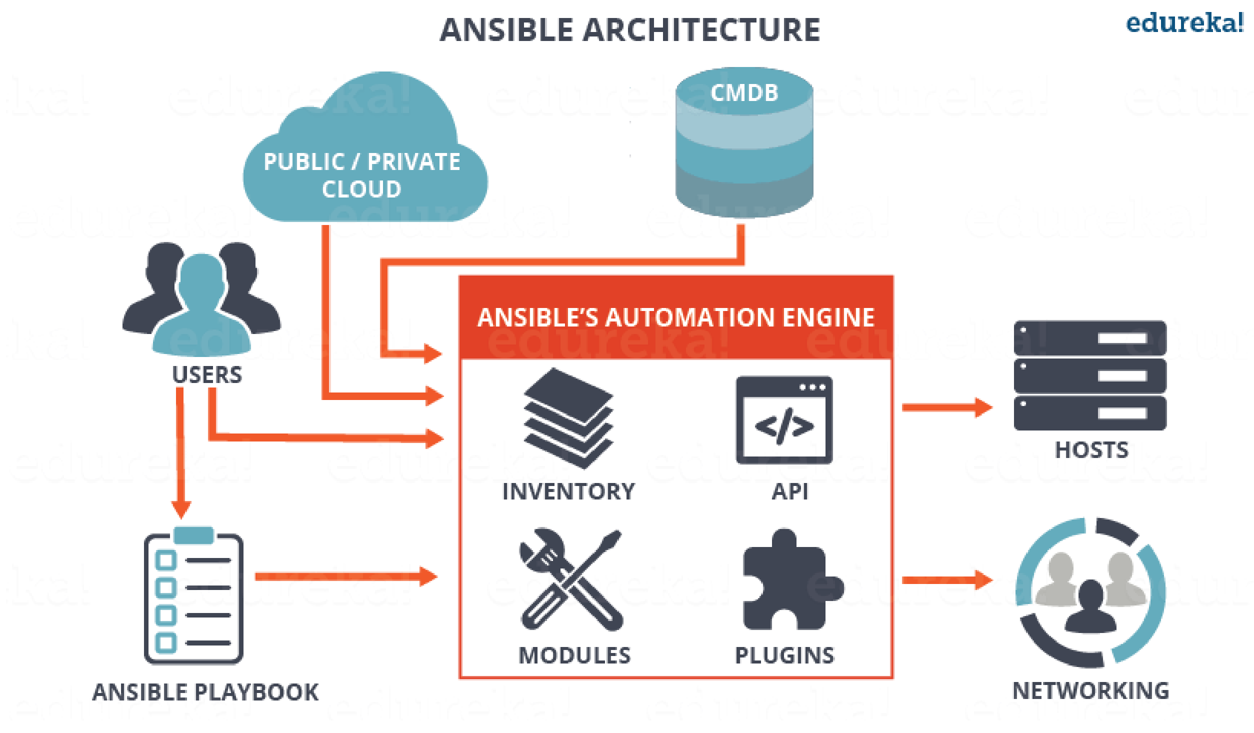
Ansible, unlike other configuration management tools, is categorized into two types of servers – Controlling machines and Nodes. Controlling machine is where Ansible is installed and nodes are the ones that are managed by the controlling machines through SSH. There is an inventory file in the controlling machine that holds the location of the node systems. Ansible deploys modules on the node systems by running the playbook on the controlling machine. Ansible is agentless, that means there is no need to have a third party tool to make a connection between one node and the other.

Q5. How is Ansible different from Puppet?

Metrics	Ansible	Puppet
1. Availability 2. Ease of set up 3. Management 4. Scalability 5. Configuration language 6. Interoperability 7. Pricing nodes	● Highly available ● Easy ● Easy management ● Highly scalable ● YAML(Python) ● High ● $10,000	● Highly available ● Comparatively hard to set up ● Not very easy ● Highly scalable ● DSL(PuppetDSL) ● High ● $11200-$19900

Q6. What are the different components of ansible? Explain Ansible architecture.

The below diagram depicts the Ansible architecture:



Ansible Architecture – Ansible Interview Questions – Edureka

The main component of Ansible is the Ansible automation engine. This engine directly interacts with various cloud services, Configuration Management Database (CMBD) and different users who write various playbooks to execute the Ansible Automation engine.

The Ansible Automation engine consists of the following components:

Inventories: These are a list of nodes containing their respective IP addresses, servers, databases, etc. which needs to be managed.

APIs: Just like any other API, the Ansible APIs are used for commuting various Cloud services, public or private services.

Modules: The modules are used to manage system resources, packages, libraries, files, etc. Ansible modules can be used to automate a wide range of tasks. Ansible provides around 450 modules that automate nearly every part of your environment.

Plugins: If you want to execute Ansible tasks as a job, Ansible Plugins can be used. They simplify the execution of a task by building a job like an environment that basically contains pieces of code corresponding to some specific functionality. There are 100s of Plugins provided by Ansible. An example is the Action plugin, which acts as front ends to modules and can execute tasks on the controller before calling the modules themselves.

Networking: Ansible can also be used to automate different networks and services. It can do this by creating a playbook or an Ansible role that easily spans different network hardware.

Hosts: The Ansible Hosts/ Node systems are machines (Linux, Windows, etc) that are getting automated.

Playbooks: Playbooks are simple code files which describe the tasks that need to be executed. The Playbooks are written in YAML format. They can be used to automate tasks, declare configurations, etc.

CMDB: It is a database that acts as a storehouse for various IT installations. It holds data about various IT assets (also known as configuration items (CI)) and describes the relationships between such assets.

Cloud: It is a network of remote servers hosted on the Internet to store, manage, and process data, rather than a local server.

To learn more about Ansible, you can go through this Ansible Tutorial blog.

Q7. What are Ansible Server requirements?

If you are a windows user then you need to have a virtual machine in which Linux should be installed. It requires Python 2.6 version or higher. you fulfill these requirements and you're good to go!

Q8. How would you install Ansible on a CentOS system?

This can be done in two simple steps:

Step 1: Set up EPEL Repository

EPEL (Extra Packages for Enterprise Linux) is an open source and free community-based repository project from Fedora team which provides high-quality add-on software packages for Linux distribution including RHEL (Red Hat Enterprise Linux), CentOS, and Scientific Linux.

The Ansible package is not available in the default yum repositories, so we will enable EPEL repository by using the below command:

```
sudo rpm -ivh
http://dl.fedoraproject.org/pub/epel/6/i386/epel-release-6-8.noarch.rpm
```

This will download all the necessary packages which will be required to install Ansible.

Step 2: Install Ansible

Now that your EPEL repository has been added, all you have to do now is install Ansible using the command below:

```
yum install ansible -y
```

That's all! It's a two-step process that barely takes a minute!

If you wish to check the version of Ansible installed on your system, use the command below:

```
ansible –version
```

Q9. Explain a few of the basic terminologies or concepts in Ansible.

Few of the basic terms that are commonly used while operating on Ansible are:

Controller Machine: The Controller machine is responsible for provisioning the servers that are being managed. It is the machine where Ansible is installed.

Inventory: An inventory is an initialization file that has details about the different servers you are managing.

Playbook: It is a code file written in the YAML format. A playbook basically contains the tasks that need to be executed or automated.

Task: Each task represents a single procedure that needs to be executed, e.g. Install a library.

Module: A module is a set of tasks that can be executed. Ansible has 100s of built-in modules, but you can also create custom ones.

Role: An Ansible role is a pre-defined way for organizing playbooks and other files in order to facilitate sharing and reusing portions of provisioning.

Play: A task executed from start to finish or the execution of a playbook is called a play.

Facts: Facts are global variables that store details about the system, like network interfaces or operating system.

Handlers: Are used to trigger the status of a service, such as restarting or stopping a service.

Q10. Explain the concept behind Infrastructure as Code (IaC).

Infrastructure as Code (IaC) is a process for managing and operating data servers, storage systems, system configurations, and network infrastructure.

In traditional configuration management practices, each minute configuration change required manual action by system administrators and the IT support team. But with IaC, all the configuration details are managed and stored in a standardized file system, wherein the system automatically manages infrastructure changes and deals with system configurations.

Therefore, we do not require most of the manual effort since everything is managed and automated by following the IaC approach. Tools such as Ansible can be used to implement IaC approach.

Q11. Compare Ansible with Chef.

Metrics	Ansible	Chef
1. Availability	Highly available	Highly available
2. Ease of set up	Easy	Not very easy
3. Management	Easy management	Not very easy
4. Scalability	Highly scalable	Highly scalable
5. Configuration language	YAML(Python)	DSL(Ruby)
6. Interoperability	High	High
7. Pricing nodes	$10,000	$13700

Q12. What is Ansible Galaxy?

Galaxy is a website that lets Ansible users share their roles and modules. The Ansible Galaxy command line tool comes packed with Ansible, and it can be used to install roles from Galaxy or directly from a Source Control Management system such as Git. It can also be used to build new roles, remove existing ones and perform tasks on the Galaxy website.

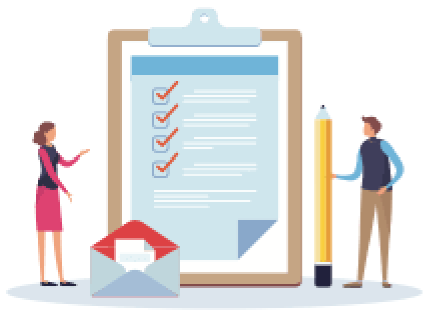
You can use the below command to download roles from the Galaxy website:

```
$ansible-galaxy install username.role_name
```

Q13. What are Ad-hoc commands? Give an example.

Ad-hoc commands are simple one-line commands used to perform a certain task. You can think of Adhoc commands as an alternative to writing playbooks. An example of an Adhoc command is as follows:

```
ansible host -m netscaler -a "nsc_host=nsc.example.com user=apiuser password=apipass"
```



Explore Curriculum

The above Adhoc command accesses the netscaler module to disable the server.

Q14. What are the variables in Ansible?

Variables in Ansible are very similar to variables in any programming language. Just like any other variable, an Ansible variable is assigned a value which is used in computing playbooks. You can also use conditions around the variables. Here’s an example:

```
- hosts: your hosts
vars:
port_Tomcat : 8080
```

Here, we've defined a variable called port_Tomcat and assigned the port number 8080 to it. Such a variable can be used in the Ansible Playbook.

Q15. What is the difference between a variable name and an environment variable?

Variable name	Environment variable
• You need to add strings to create variable names • You can easily create multiple variable names by adding strings • We use the ipv4 address for variable names	• You need existing variables to access environment variables. • To create environment variables we must refer advanced Ansible playbook • We use {{ ansible_env.SOME_VARIABLE }} for remote environment variables.

Q16. What are the Ansible Modules? Explain the different types.

Ansible modules are a small set of programs that perform a specific task. Modules can be used to automate a wide range of tasks. Modules in Ansible are considered to be idempotent or in other words, making multiple identical requests has the same effect as making a single request.

There are 2 types of modules in Ansible:

1. Core modules
2. Extras modules

Core Modules

These are modules that the core Ansible team maintains and will always ship with Ansible itself. They will also receive a slightly higher priority for all requests than those in the "extras" repos. The source of these modules is hosted by Ansible on GitHub in the Ansible-modules-core.

Extras Modules

These modules are currently shipped with Ansible but might be shipped separately in the future. They are also mostly maintained by the Ansible Community. Non-core modules are still fully usable but may receive slightly lower response rates for issues and pull requests.

Popular "extras" modules may be promoted to core modules over time. The source for these modules is hosted by Ansible on GitHub in the Ansible-modules-extras.

Q17. What is Ansible Task?

Ansible Tasks allow you to break up bits of configuration policy into smaller files. These are blocks of code that can be used to automate any process. For example, if you wish to install a package or update a software, you can follow the below code snippet:

```
Install <package_name>, update <software_name>
```

Q18. Can you explain what are playbooks in Ansible? Explain with some examples.

Playbooks in Ansible are written in the YAML format. It is a human-readable data serialization language. It is commonly used for configuration files. It can also be used in many applications where data is being stored.

For Ansible, nearly every YAML file starts with a list. Each item in the list is a list of key/value pairs, commonly called a "hash" or a "dictionary". So, we need to know how to write lists and dictionaries in YAML.

All members of a list are lines beginning at the same indentation level starting with a "- " (dash and space). More complicated data structures are possible, such as lists of dictionaries or mixed dictionaries whose values are lists or a mix of both.

For example, if you want a playbook containing details about the USA:

```
-USA
-continent: North America
-capital: Washington DC<a name="AnsibleIntermediateLevelInterviewQuestions"></a>
-population: 319 million
```

Now that you know the basic level questions, let's take a look at a little more advanced Ansible Interview Questions.

Intermediate Level Ansible Interview Questions

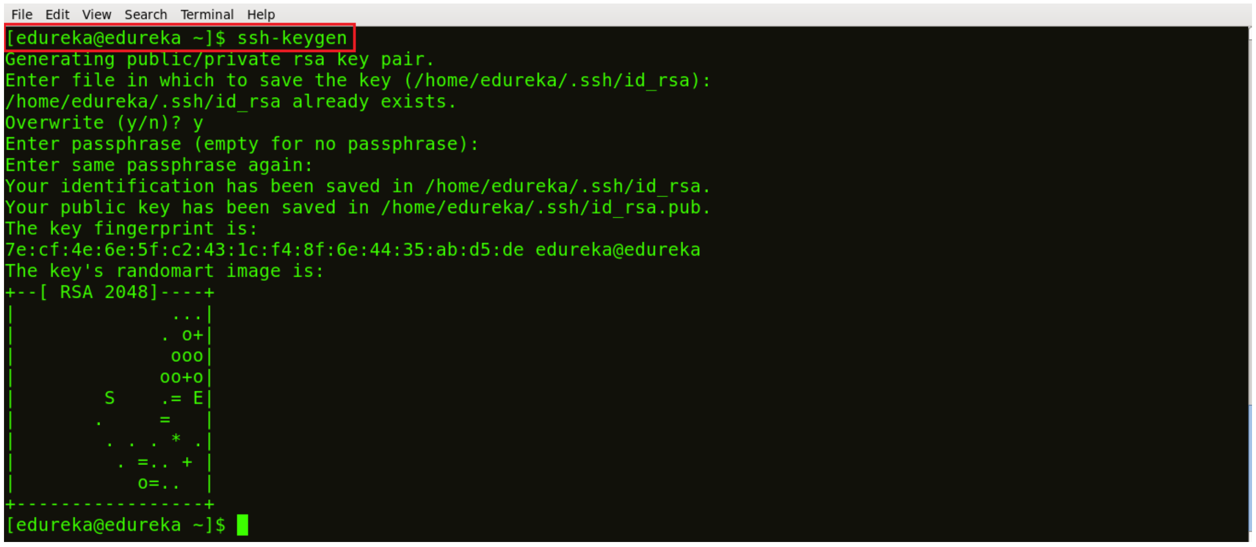
Once you've aced the basic conceptual questions, the interviewer will increase the difficulty level. So let's move on to the next section of this Ansible Interview Questions article. This section talks about the commands that are very common amongst docker users.

Q19. Can you write a simple playbook to install Nginx on a host machine?

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

Follow the command below:

```
$ ssh-keygen
```



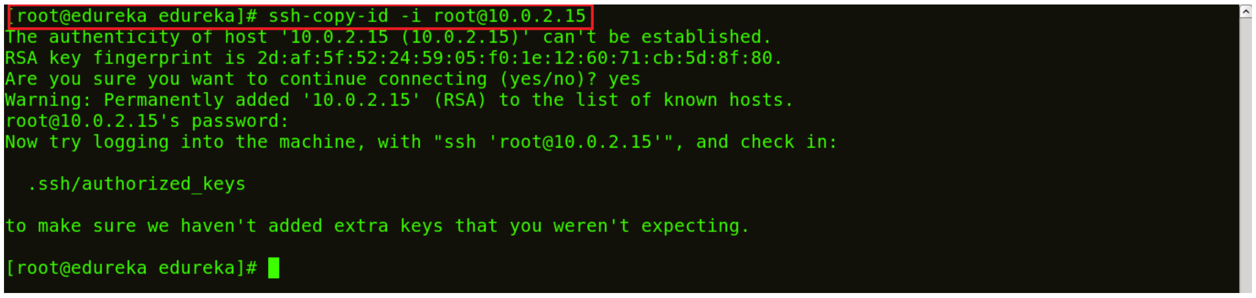
```
[edureka@edureka ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/edureka/.ssh/id_rsa):
/home/edureka/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/edureka/.ssh/id_rsa.
Your public key has been saved in /home/edureka/.ssh/id_rsa.pub.
The key fingerprint is:
7e:cf:4e:6e:5f:c2:43:1c:f4:8f:6e:44:35:ab:d5:de edureka@edureka
The key's randomart image is:
+--[ RSA 2048]----+
|              ...|
|             . o+|
|              ooo|
|             oo+o|
|        S    .= E|
|       .     =   |
|        . . . * .|
|         . =.. + |
|           o=..  |
+-----------------+
[edureka@edureka ~]$
```

Ansible Playbook – Ansible Interview Questions – Edureka

As shown above, a public SSH key is generated.

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
```



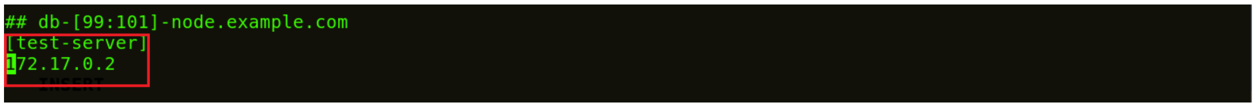
```
[root@edureka edureka]# ssh-copy-id -i root@10.0.2.15
The authenticity of host '10.0.2.15 (10.0.2.15)' can't be established.
RSA key fingerprint is 2d:af:5f:52:24:59:05:f0:1e:12:60:71:cb:5d:8f:80.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.0.2.15' (RSA) to the list of known hosts.
root@10.0.2.15's password:
Now try logging into the machine, with "ssh 'root@10.0.2.15'", and check in:

  .ssh/authorized_keys

to make sure we haven't added extra keys that you weren't expecting.

[root@edureka edureka]#
```

Ansible Playbook – Ansible Interview Questions – Edureka

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

Follow the below command:

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

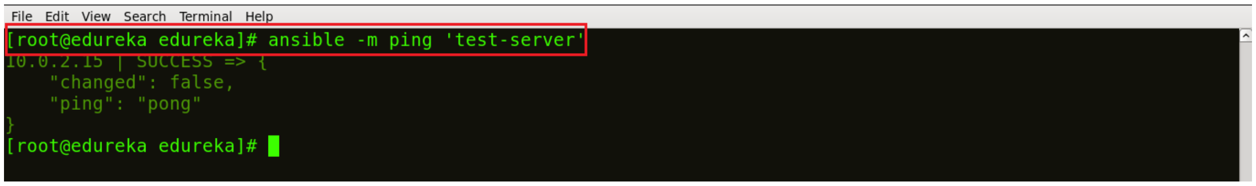


```
## db-[99:101]-node.example.com
[test-server]
172.17.0.2
```

Ansible Playbook – Ansible Interview Questions – Edureka

Once you run the command, the vi editor will open where you can list down the IP addresses of your hosts. This is now your inventory.

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



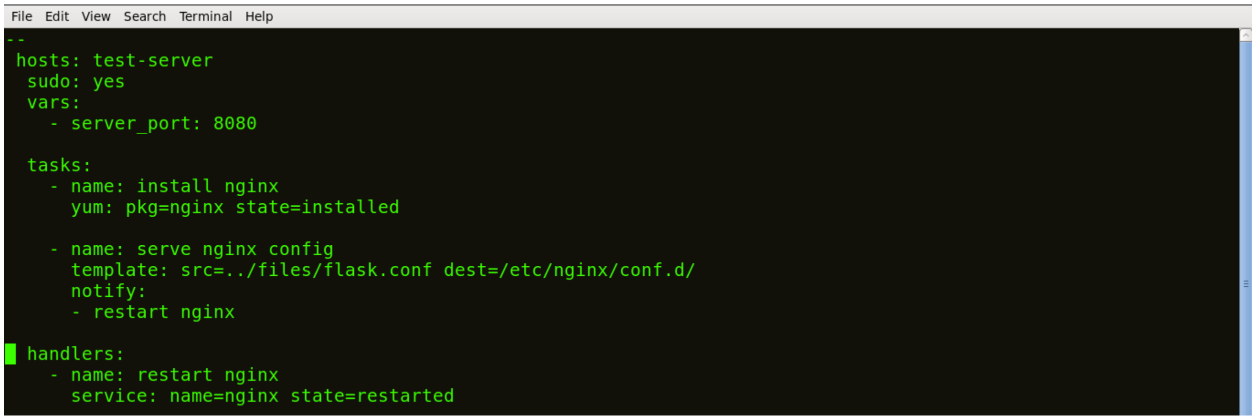
```
[root@edureka edureka]# ansible -m ping 'test-server'
10.0.2.15 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
[root@edureka edureka]# 
```

Ansible Playbook – Ansible Interview Questions – Edureka

The above image shows that a connection has been made between the control machine and the host.

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
```



```
--
 hosts: test-server
  sudo: yes
  vars:
    - server_port: 8080

  tasks:
    - name: install nginx
      yum: pkg=nginx state=installed

    - 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
```

Ansible Playbook – Ansible Interview Questions – Edureka

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.

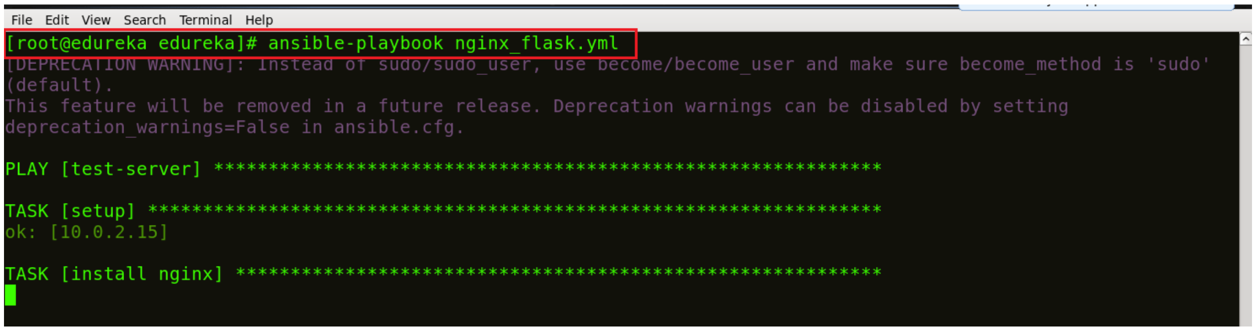
Here, the template is a file you can deploy on hosts. However, template files also include some reference variables which are pulled from variables defined as part of an Ansible playbook or facts gathered from the hosts. Facts containing the configuration details are being pulled from a source directory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

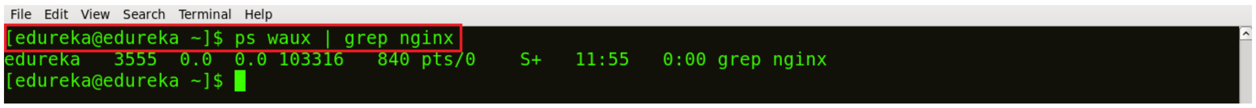
```
ansible-playbook <name of your file>.yml
```


```
[root@edureka edureka]# ansible-playbook nginx_flask.yml
[DEPRECATION WARNING]: Instead of sudo/sudo_user, use become/become_user and make sure become_method is 'sudo'
(default).
This feature will be removed in a future release. Deprecation warnings can be disabled by setting
deprecation_warnings=False in ansible.cfg.

PLAY [test-server] *************************************************************

TASK [setup] *******************************************************************
ok: [10.0.2.15]

TASK [install nginx] ***********************************************************
```

Ansible Playbook – Ansible Interview Questions – Edureka

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

```
ps waux | grep nginx
```


```
[edureka@edureka ~]$ ps waux | grep nginx
edureka    3555  0.0  0.0 103316   840 pts/0    S+   11:55   0:00 grep nginx
[edureka@edureka ~]$
```

Ansible Playbook – Ansible Interview Questions – Edureka

In the above image, the different process IDs 3555 and 103316 are running which shows that Nginx is running on your host machines.

Q20. How would you access a variable of the first host in a group?

This can be done by executing the below command:

```
{{ hostvars[groups['webservers'][0]]['ansible_eth0']['ipv4']['address'] }}
```

In the above command, we're basically accessing the hostname of the first machine in the webservers group. If you're using a template to do this, use the Jinja2 '#set' or you can also use set_fact, like shown below:

```
- set_fact: headnode={{ groups[['webservers'][0]] }}
```

```
2
    - debug: msg={{
    hostvars[headnode].ansible_eth0.ipv4.address }}
```

Q21. Why is '{{ }}' notation used? And how can one interpolate variables or dynamic variable names?

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.

Next

Q22. What is Ansible role and how are they different from the playbook?

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.

Suppose you 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.

Q23. How do I write an Ansible handler with multiple tasks?

Suppose you 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
```

Q24. How to keep secret data in a playbook?

Suppose you have a task that you don’t want to show the output or command given to it when using -v (verbose) mode, the following task can be used to do it:

```
- name: secret task
shell: /usr/bin/do_something --value={{ secret_value }}
no_log: True
```

This can be used to keep verbose output but hide sensitive information from others who would otherwise like to be able to see the output.

The no_log attribute can also apply to an entire play:

```
- hosts: all
no_log: True
```

Q25. What are Ansible Vaults and why are they used?

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.

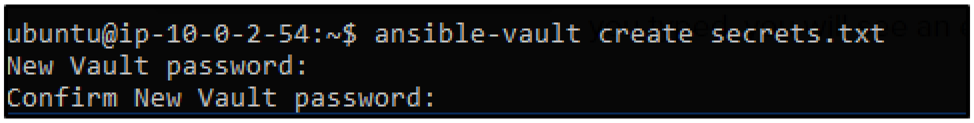
Vault is implemented with file-level granularity where the files are either entirely encrypted or entirely unencrypted. It uses the same password for encrypting as well as for decrypting files which makes using Ansible Vault very user-friendly.

Q26. How to create encrypted files using Ansible?

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

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

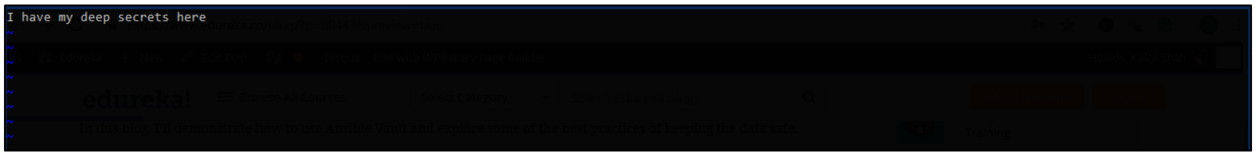
You'll be prompted to create a password and then confirm it by re-typing it.



```
ubuntu@ip-10-0-2-54:~$ ansible-vault create secrets.txt
New Vault password:
Confirm New Vault password:
```

Ansible Playbook – Ansible Interview Questions – Edureka

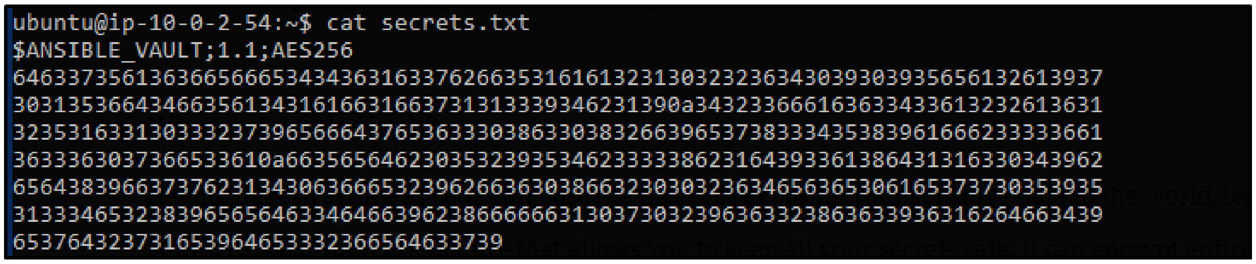
Once your password is confirmed, a new file will be created and will open an editing window. By default, the editor for Ansible Vault is vi. You can add data, save and exit.



```
I have my deep secrets here
~
~
~
~
~
~
~
```

Ansible Playbook – Ansible Interview Questions – Edureka

This is your encrypted file:



```
ubuntu@ip-10-0-2-54:~$ cat secrets.txt
$ANSIBLE_VAULT;1.1;AES256
64633735613636656665343436316337626635316161323130323236343039303935656132613937
3031353664346635613431616631663731313339346231390a343233666163633433613232613631
32353163313033323739656664376536333038633038326639653738333435383961666233333661
3633363037366533610a663565646230353239353462333338623164393361386431316330343962
65643839663737623134306366653239626636303866323030323634656365306165373730353935
31333465323839656564633464663962386666663130373032396363323863633936316264663439
653764323731653964653332366564633739
```

Ansible Playbook – Ansible Interview Questions – Edureka

Q27. What is Ansible Tower?

Ansible Tower is Ansible at a more enterprise level. It is a web-based solution for managing your organization with a very easy user interface that provides a dashboard with all of the state summaries of all the hosts, allows quick deployments, and monitors all configurations.

The tower allows you to share the SSH credentials without exposing them, logs all the jobs, manage inventories graphically and syncs them with a wide variety of cloud providers.

Q28. What features does the Ansible Tower provide?

- 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.

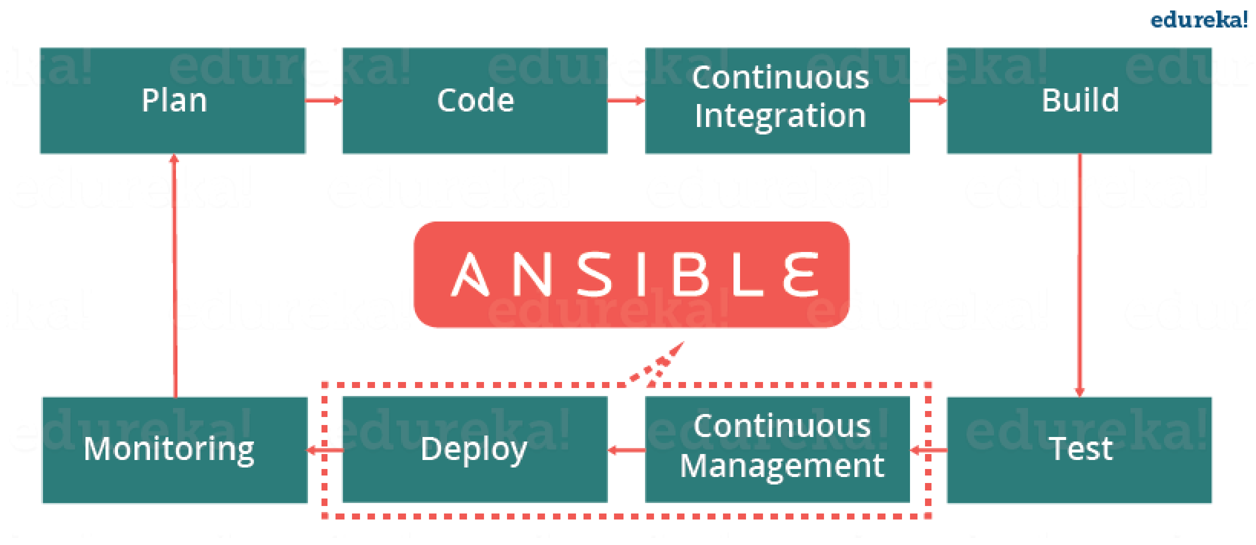
- Self-Service – This feature of Ansible Tower lets you launch Playbooks with just a single click. It can also, let you choose from available secure credentials or prompt you for variables and monitor the resulting deployments.
- REST API & Tower CLI Tool – Every feature present in Ansible Tower is available 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.

The following section will cover only the technical based Ansible Interview Questions. These questions are mostly based on the practical implementation of Ansible.

Ansible Technical Interview Questions

Q29. How is Ansible used in a 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.



Ansible In DevOps – Ansible Interview Questions – Edureka

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.

Q30. How can you create a LAMP stack and deploy a webpage by using Ansible?

Suppose you're 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:

```yaml
---
# 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
```

Now, 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
```

Q31. How do I set the PATH or any other environment variable for a task?

The environment variables can be set by using the 'environment' keyword. It can be set for either a task or an entire playbook as well. Follow the below code snippet to see how:

```
environment:
PATH: "{{ ansible_env.PATH }}:/thingy/bin"
SOME: value
```

Q32. How can one generate encrypted passwords for the user module?

There are a couple of ways to do this. The simplest way is to use the ad-hoc command:

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

Another way is to use the mkpasswd functionality available on Linux systems:

```
mkpasswd --method=sha-512
```

However, if you're using a macOS then you can generate these passwords using Python. To do this you must first install the Passlib password hashing library:

```
pip install passlib
```

After installing it, the SHA512 password values can be generated in the following manner:

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

Q33. How can looping be done over a list of hosts in a group, inside of a template?

An easy way to do this is to iterate over a list of hosts inside of a host group, in order to fill a template configuration file with a list of servers. This can be done by accessing the "$groups" dictionary in your template, like so:

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

In order to access facts about these hosts, like, the IP address of each hostname, you need to make sure that the facts have been populated. For instance, make sure you have a play that talks to db_servers:

```
- hosts: db_servers
tasks:
- debug: msg="doesn't matter what you do, just that they were talked to previously."
```

Now you can use the facts within your template, like so:

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

Q34. How can I display all the inventory vars defined for my host?

In order to check the inventory vars resulting from what you've defined in the inventory, you can execute the below command:

```
ansible -m debug -a "var=hostvars['hostname']" localhost
```

This will list down all the inventory vars.

Q35. How should one configure a jump host to access servers that I have no direct access to?

The first step would be to set a ProxyCommand in the ansible_ssh_common_args inventory variable. All arguments that are defined in this variable are added to the sftp/scp/ssh command line when connecting to the relevant host. Let's look at an example, consider the below inventory group:

```
[gatewayed]
foo ansible_host=192.0.2.1
bar ansible_host=192.0.2.2
```

Next, you can create group_vars/gatewayed.yml containing the following:

`ansible_ssh_common_args: '-o ProxyCommand="ssh -W %h:%p -q user@gateway.example.com"'`

Ansible will then append these arguments to the command line while trying to connect to any hosts in the group gatewayed.

Q36. How can you handle different machines needing different user accounts or ports to log in with?

The simplest way to do this is by setting inventory variables in the inventory file.

Let's consider that these hosts have different usernames and ports:

```
[webservers]
asdf.example.com ansible_port=5000 ansible_user=alice
jkl.example.com ansible_port=5001 ansible_user=bob
```

Also, if you wish to, you can specify the connection type to be used:

```
[testcluster]
localhost ansible_connection=local
/path/to/chroot1 ansible_connection=chroot
foo.example.com ansible_connection=paramiko
```

To make this more clear it is best to keep these in group variables or file them in a group_vars/<group-name> file.

Q37. Is it unsafe to bulk-set task arguments from a variable?

To set all the arguments in a task you can use the dictionary-typed variable. Even though this is usually good for dynamic executions, it induces a security risk. Therefore, when this happens, Ansible issues a warning. For example, consider the below code:

```
vars:
usermod_args:
name: testuser
state: present
tasks:
- user: '{{ usermod_args }}'
```

This example is safe but creating similar tasks is risky because the parameters and values passed to usermod_args could be overwritten by malicious values in the host facts on a compromised target machine.

Q38. Suppose you're using Ansible to configure the production environment and your playbook uses an encrypted file. Encrypted files prompt the user to enter passwords. But since Ansible is used for automation, can this process be automated?

Yes, Ansible uses a feature called password file, where all the passwords to your encrypted files can be saved. So each time the user is asked for the password, he can simply make a call to the password file. The password is automatically read and entered by Ansible.

```
$ ansible-playbook launch.yml --vault-password-file ~/ .vault_pass.txt
```

Having a separate script that specifies the passwords is also possible. You need to make sure the script file is executable and the password is printed to standard output for it to work without annoying errors.

```
$ ansible-playbook launch.yml --vault-password-file ~/ .vault_pass.py
```

Q39. Have you worked with Ansible before? Please share your experience.

Be very honest here. If you have used ansible before then talk about your experience. Talk about the projects that required ansible. You can tell the interviewer about how Ansible has helped you in provisioning and configuration management. If you haven't used Ansible before then just talk about any related tools that you've used. These related tools could be anything like Git, Jenkins, Puppet, Chef, Satltstack, etc.

Be very honest because they know if you're lying.

Q40. Is Ansible an Open Source tool?

Yes, Ansible is open source. That means you take the modules and rewrite them. Ansible is an open-source automated engine that lets you automate apps.

Q41. How can you connect other devices within Ansible?

Once Ansible is installed on the controlling machines, an inventory file is created. This inventory file specifies the connection between other nodes. A connection can be made using a simple SSH. To check the connection to a different device, you can use the ping module.

```
ansible -m ping all
```

The above command checks the connection to all the nodes specified in the inventory file.

Q42. Is it possible to build our modules in Ansible?

Yes, we can create our own modules within Ansible. It's an open-source tool which basically works on python. You can start creating your own modules. The only requirements would be to be amazingly good at programming.

Q43. What does Fact mean in Ansible?

When any new variable about the system has been discovered it's considered to be a "fact" in the playbook. Facts are mainly used to implement conditional executions. It can also be used to get the ad-hoc information about the system.

You can get facts with the following command:

```
$ ansible all- m setup
```

So when you want to extract only a part of the information, you use the setup module to filter out only the needed information.

Q44. What is the ask_pass module in Ansible?

Ask_pass is the control module in an Ansible playbook. This controls the prompting of the password when the playbook is getting executed. By default, it’s always set to True. If you are using SSH keys for authentication purposes then you really don’t have to change this setting at all.

Q45. Explain the callback plugin in Ansible?

Callback plugins are enable adding new behaviors to Ansible when responding to events. By default, callback plugins control most of the output you see when running the command line program. It can also be used to add additional output, integrate with other tools, etc.

Q46. Does Ansible support AWS?

Ansible has hundreds of modules supporting AWS and some of them include:

- Autoscaling groups
- CloudFormation
- CloudTrail
- CloudWatch
- DynamoDB
- ElastiCache
- Elastic Cloud Compute (EC2)
- Identity Access Manager (IAM)
- Lambda
- Relational Database Service (RDS)
- Route53
- Security Groups
- Simple Storage Service (S3)

- Virtual Private Cloud (VPC)

Q47. Does Ansible support hardware provisioning?

Yes, Ansible can provision hardware. A lot of companies are still stuck on to massive data centers of hardware. There are a few requirements. You must set up some services before you go ahead. Some of them are – DHCP, PXE, TFTP, Operating System Media, Web Server, etc.

Q48. Write an Ansible playbook to automate the starting of EC2 instance.

```
---
 - name: Create an ec2 instance
  hosts: web
  gather_facts: false

  vars:
      region: us-east-1
      instance_type: t2.micro
      ami: ami-05ea7729e394412c8
      keypair: priyajdm

  tasks:

    - name: Create an ec2 instance
```

```
      ec2:
         aws_access_key: '*******************'
         aws_secret_key: '****************************************'
         key_name: "{{ keypair }}"
         group: launch-wizard-26
         instance_type: "{{ instance_type }}"
         image: "{{ ami }}"
         wait: true
         region: "{{ region }}"
         count: 1
         vpc_subnet_id: subnet-02f498e16fd56c277
         assign_public_ip: yes
    register: ec2
```

- We start by mentioning AWS access key id and secret key using the parameters aws_access_key and aws-secret_key.
- key_name: pass the variable that defines the keypair being used here
- group: mention the name of the security group. This defines the security rules of the EC2 instance we’re trying to bring up
- instance_type: pass the variable that defines the type of instance we’re using here
- image: pass the variable that defines the AMI of the image we’re trying to start.

- wait: This has a boolean value of either true or false. If true, it waits for the instance to reach the desired state before returning
- region: pass the variable that defines the region in which an EC2 instance needs to be created.
- count: This parameter specifies the number of instances that need to be created. In this case, I’ve only mentioned only one but this depends on your requirements.
- vpc_subnet_id: pass the subnet id in which you wish to create the instance
- assign_public_ip: This parameter has a boolean value. If true like in our case, a public IP will be assigned to the instance when provisioned within VPC.

Q49. Can you copy files recursively onto a target host? If yes, how?

Yes, you can copy files recursively onto a target host using the copy module. It has a recursive parameter which copies files from a directory. There is another module called synchronize which is specifically made for this.

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

Q50. Write a playbook to create a backup of a file in the remote servers before copy.

This is pretty simple. You can use the below playbook:

```
- hosts: blocks
tasks:
```

```
- name: ansible copy file backup example
copy:
src: ~/helloworld.txt
dest: /tmp
backup: yes
```