

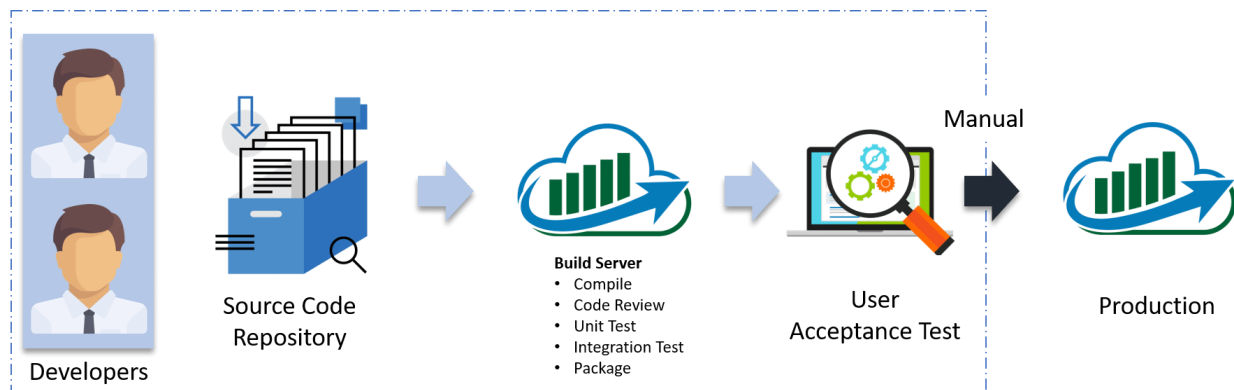
# Ansible Basic Level Interview Questions

This category of Ansible Interview Questions consists of questions that you're expected to know. These are the most basic questions. An interviewer will start with these and eventually increase the difficulty level. Let's have a look at them.

## 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?

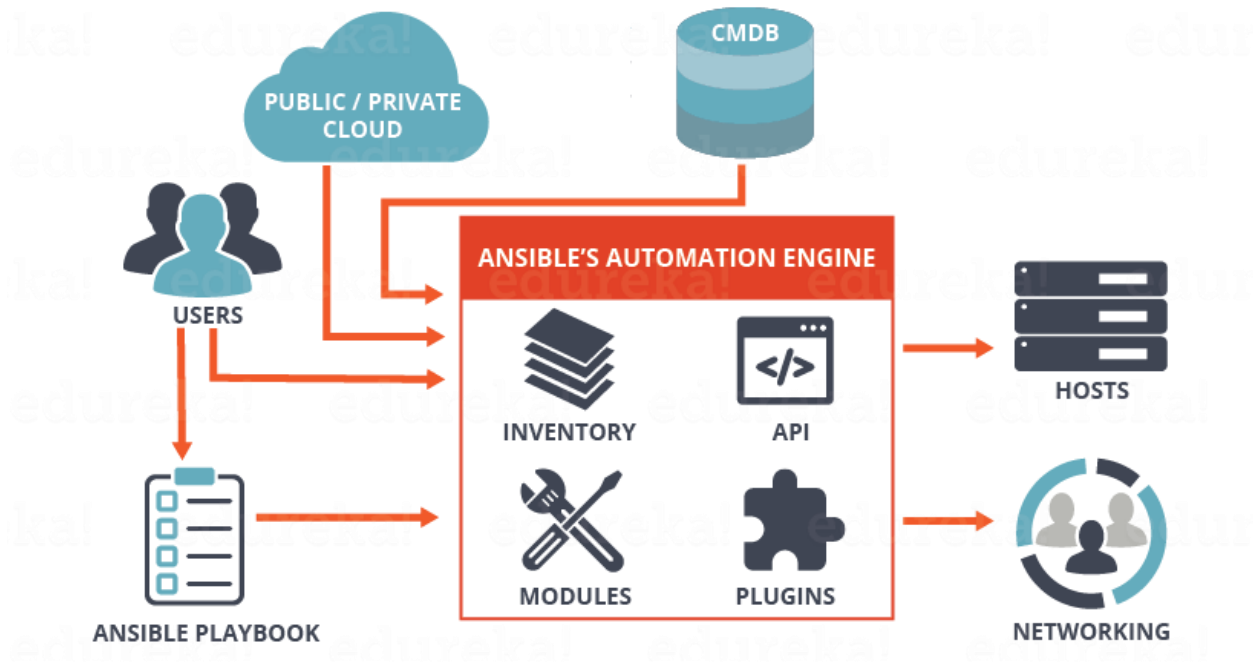
Metrics	Ansible	Puppet
1. Availability	<ul style="list-style-type: none"> <li>• Highly available</li> </ul>	<ul style="list-style-type: none"> <li>• Highly available</li> </ul>
2. Ease of set up	<ul style="list-style-type: none"> <li>• Easy</li> </ul>	<ul style="list-style-type: none"> <li>• Comparatively hard to set up</li> </ul>
3. Management	<ul style="list-style-type: none"> <li>• Easy management</li> </ul>	<ul style="list-style-type: none"> <li>• Not very easy</li> </ul>
4. Scalability	<ul style="list-style-type: none"> <li>• Highly scalable</li> </ul>	<ul style="list-style-type: none"> <li>• Highly scalable</li> </ul>
5. Configuration language	<ul style="list-style-type: none"> <li>• YAML(Python)</li> </ul>	<ul style="list-style-type: none"> <li>• DSL(PuppetDSL)</li> </ul>
6. Interoperability	<ul style="list-style-type: none"> <li>• High</li> </ul>	<ul style="list-style-type: none"> <li>• High</li> </ul>
7. Pricing nodes	<ul style="list-style-type: none"> <li>• \$10,000</li> </ul>	<ul style="list-style-type: none"> <li>• \$11200-\$19900</li> </ul>

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

The below diagram depicts the Ansible architecture:

## ANSIBLE ARCHITECTURE

edureka!



### *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 (CMDB) 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.

## **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 netcaler -a "nsc_host=nsc.example.com user=apiuser password=apipass"
```

The above Adhoc command accesses the netcaler 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:

```
1- hosts: your hosts
2vars:
3port_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:

```
1-USA
2-continent: North America
3-capital: Washington DC<a name="AnsibleIntermediateLevelInterviewQuestions"></a>
4-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

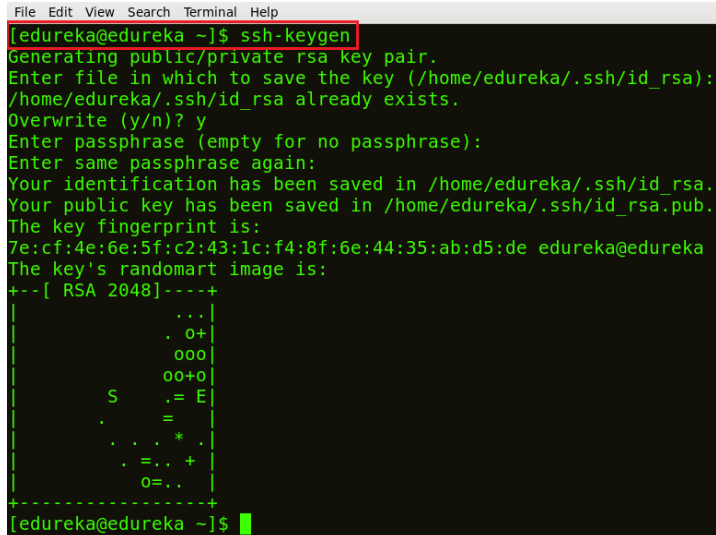
Once you’ve aced the basic conceptual questions, the interviewer will increase the difficulty level. So let’s move on to the next section of this Ansible Interview Questions article. This section talks about the commands that are very common amongst docker users.

## 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`



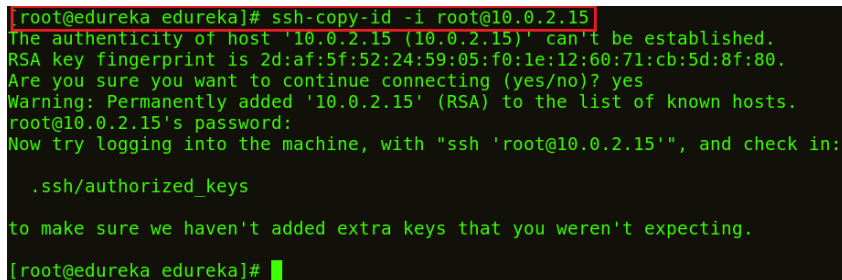
```
File Edit View Search Terminal Help
[edureka@edureka ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/edureka/.ssh/id_rsa):
/home/edureka/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase 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       |
|      .  =         |
|     . . . *      |
|    . =. . +      |
|   . 0=..         |
+-----+
[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:

```
File Edit View Search Terminal Help
[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
```

```
File Edit View Search Terminal Help
--
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.

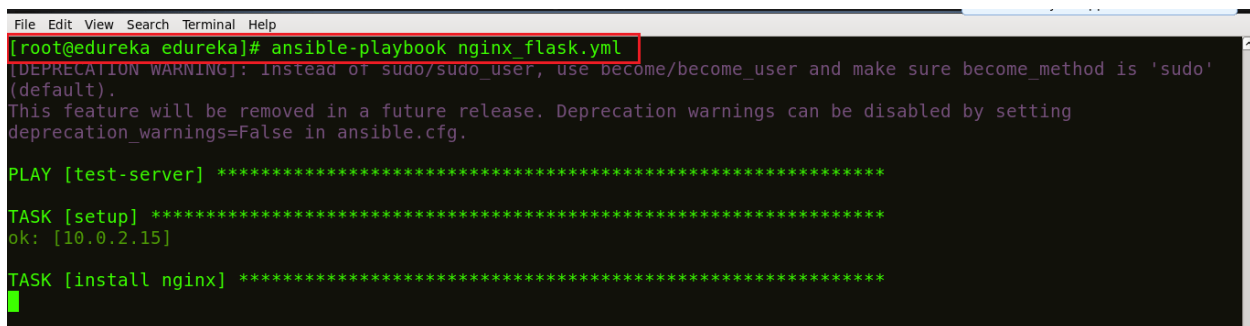
Here, the template is a file you can deploy on hosts. However, template files also include some reference variables which are pulled from variables defined as part of an Ansible playbook or facts gathered from the hosts. Facts containing the configuration details are being pulled from a source directory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

A terminal window with a dark background and light green text. The window title is "File Edit View Search Terminal Help". The prompt is "[root@edureka edureka]#". The command entered is "ansible-playbook nginx\_flask.yaml". The output shows a deprecation warning about the 'sudo' method, followed by a separator line. The first task is "PLAY [test-server]", followed by another separator line. The second task is "TASK [setup]", which completes successfully with "ok: [10.0.2.15]". The third task is "TASK [install nginx]", which is partially visible at the bottom of the screen.

```
File Edit View Search Terminal Help
[root@edureka edureka]# ansible-playbook nginx_flask.yaml
[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
```

```
File Edit View Search Terminal Help
[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:

- 
- 1- 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.