1. What is the difference between CI and 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.

1. What is Configuration Management, and how does it work?

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.

1. What is Ansible, and describe it’s working?

Ansible is simple open source IT engine which automates application deployment, intra service orchestration, cloud provisioning and many other IT tools. Ansible is easy to deploy because it does not use any agents or custom security infrastructure.

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.

1. What distinguishes Ansible from other similar tools?

Ansible is open source software,and doesn't rely on a client-server model. According to its designers, Ansible is the only automation engine that can automate the entire application lifecycle and continuous delivery pipeline.

Ansible is a simpler approach to the configuration management work. It avoids the master and minions approach used by other open source configuration management tools like Chef. Rather than setting up a client-server installation, Ansible uses an agentless architecture.

1. What is the purpose of the 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.

1. Can you go over the 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.

1. What is a YAML file, and how does Ansible use it?

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

1. What are the different types of Ansible tasks?

A task is the smallest unit of action you can automate using an Ansible playbook. Playbooks typically contain a series of tasks that serve a goal, such as to set up a web server, or to deploy an application to remote environments. Ansible executes tasks in the same order they are defined inside a playbook.

Each play contains a list of tasks. Tasks are executed in order, one at a time, against all machines matched by the host pattern, before moving on to the next task. It is important to understand that, within a play, all hosts are going to get the same task directives. It is the purpose of a play to map a selection of hosts to tasks.

When running the playbook, which runs top to bottom, hosts with failed tasks are taken out of the rotation for the entire playbook. If things fail, simply correct the playbook file and rerun.

The goal of each task is to execute a module, with very specific arguments. Variables can be used in arguments to modules.

Modules should be idempotent, that is, running a module multiple times in a sequence should have the same effect as running it just once. One way to achieve idempotency is to have a module check whether its desired final state has already been achieved, and if that state has been achieved, to exit without performing any actions. If all the modules a playbook uses are idempotent, then the playbook itself is likely to be idempotent, so re-running the playbook should be safe.

The command and shell modules will typically rerun the same command again, which is totally ok if the command is something like chmod or setsebool, etc. Though there is a creates flag available which can be used to make these modules also idempotent.

Every task should have a name, which is included in the output from running the playbook. This is human readable output, and so it is useful to provide good descriptions of each task step. If the name is not provided though, the string fed to ‘action’ will be used for output.

Tasks can be declared using the legacy action: module options format, but it is recommended that you use the more conventional module: options format. This recommended format is used throughout the documentation, but you may encounter the older format in some playbooks.

1. What are the best ways to use YAML files in high-level programming languages like Java, Python, and others?

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.

1. How to set up 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@devopsschool.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.

1. How can I use encrypted files to automate password entry in a playbook?

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

1. What are Ansible callback plugins?

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.

1. What is Ansible Inventory and what are the many types of it?

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

14.What is an Ansible Vault, exactly?

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