Ansible Interview Questions

WebMagic Informatica
webmagicinformatica.com
+91 7021 4791 26

Q26. How are dot notation and array notation of variables different?

Q27. How does the Ansible synchronize module work?

Q28. How does the Ansible firewalld module work?

Q29. How is the Ansible set_fact module different from vars, vars_file, or include_var?

Q30. When is it unsafe to bulk-set task arguments from a variable?

Q31. Explain the Ansible register.

Q32. How can we delegate tasks in Ansible?

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.

Q2. 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 them.

Q3. 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 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.

Q4. 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.

Q5. Explain Infrastructure as Code?

Infrastructure as Code or IaC is a process that DevOps teams should follow to have a more organised 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.

Q6. What is 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.

Q7. 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 strings 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 rather 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.

Q8. 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
```

Q9. 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>

Q10. 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.

Q11. 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 try to connect to any host in the gatewayed group ansible will append these arguments to the command line.

Q12. 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 that 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

Q13. 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.

Q14. 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:tag
type: testing
```

Now we can fetch using this command

ansible-inventory -i demo_aws_ec2.yaml -graph

Q15. 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

Q16. 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 %}.
```

Q17. 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]

Q18. 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 staging servers 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

Q19. How do I access a variable name programmatically?

Variable names can be built by adding strings together. For example, if we need to get the 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'] }}

Q20. 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 30 mins(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.

Q21. 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 organisation. 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.

Q22. 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 synchronise which is more efficient for large numbers of files.

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

Q23. 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.

Q24. 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.

Q25. 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.

Q26. How are dot notation and array notation of variables 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

Q27. How does the Ansible synchronize module work?

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 permissions 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 the remote machine's /var/tmp folder

Q28. How does the Ansible firewalld module work?

Ansible firewalld is used to manage firewall rules on host machines. This works just as a 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
```

Q29. 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
```

Q30. 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.

Q31. Explain the 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.

Q32. 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 a 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 a few exceptions also such as include, add_host, and debug tasks that cannot be delegated.

Ansible is a great tool for automating IT tasks and it is widely used in industries, thus every software developer or someone in the DevOps team should know the basics. Also, it is very easy to set up so we can start right away. These questions cover the most important concepts related to Ansible which will help in both interview and understanding Ansible in depth.

More Interview Question: https://webmagicinformatica.com/interview-questions/