What is Ansible 🡺

* In simple words Ansible is a free automation tool that can automate tasks on local machine where it is running and on remote machine.
* Ansible is an open source software provisioning , configuration management and application deployment tool enabling infrastructure as code. It runs on many unix like systems and can configure both Unix like systems/windows systems.
* ansible is developed in python.
* Ansible can be used for the below
* provision system : you can build systems from scratch
* bare metal servers
* virtualization systems
* network devices
* storage systems
* cloud platform
* system configure Management
* updates or upgrades
* package installation
* service configuration
* stop|start|restart of services
* user or groups
* assigning permissions to files and directories
* deploy apps
* manage system and apps

Shape

Description automatically generated

History of Ansible 🡺

* started by Michael De Haan in 2012
* it is open source and community driven
* ansible inc was purchased by RedHat in 2015
* It is available for most of the Linux distributions such as Redhat,cent os, fedora,ubuntu,Debian and suse
* ansible can not be installed on windows but windows machines can be configured.

Need 🡺

* Infrastructure management(specially virtualisation)
* multitier application setup

Benefits of ansible 🡺

* Agentless: means you don’t need to install ansible agents in all the servers.
* open source
* avoid human errors while doing configuration changes for multiple servers
* increase productivity as you can save a lot of time
* easy it to use
* playbooks
* secure (over SSH)
* provides pre written modules
* Good for orchestration
* Ansible can be used not only for systems but also for networks,storage,cloud.

Terminologies in Ansible 🡺

Control node or Ansible server 🡺

* server which runs Ansible application

Module🡺

* Modules is command meant to be executed on the client side
* most of the IT tasks modules are already created and can be found on ansible website
* example of module: install http

Task 🡺

A task is a section that consists of a single procedure to be completed. A task can have multiple modules.

Playbook 🡺

automation file with step-by-step execution of multiple tasks.

YAML 🡺

a playbook written in yaml language. Yet another markup language

Inventory 🡺

File that has information about remote clients where tasks are executed

Tag 🡺

a reference or alias a specific task

Variable 🡺

variables are like container that holds the defined which can be used repetitively

Role 🡺

splitting of playbook into smaller group. Roles let you automatically load related vars, files, tasks, handlers and other ansible artifacts based on known file structure. After you group your content in roles, you can easily reuse them and share them with other users.

How Ansible works 🡺

each specific TASK in ansible is written through a module

multiple modules are written in sequential order

multiple modules for related tasks called a play

all plays together make a playbook

playbook is written as file format called yaml.

Diagram, schematic

Description automatically generated

Commands example 🡺

ansible-playbook example.yml 🡺 to run modules through yaml file

ansible myservers -m ping 🡺 to run module independently

Ansible configuration files 🡺

/etc/ansible/ansible.cgf

/etc/ansible/hosts

/etc/ansible/roles

Difference between Ansible and Other automation tool. (chef/puppet)

Puppet and chef

use RUBY language which is more difficult to learn and support is decline day by day

these tools require agents to be installed in clients

the installation process is very complex

lack of documentation

Ansible

use simple YAML

agentless (only requires SSH access)

easy installation

well documented product

Open source ansible VS Red Hat Ansible 🡺

ansible is an open source software

it was purchased by red hat in 2015.

Ansible software itself is free even though it is owned by redhat

ansible is the same software across all platforms

the only difference is red hat provide additional product ansible tower and consulting or technical support of ansible

Red Hat Ansible Tower 🡺

Red hat provides Ansible towers which is GUI based tool to manage Ansible automation.

ansible tower is paid product by red hat .

managers multiple ansible servers for large enterprise environments

Ansible AWX 🡺

Open source and free to use

Lab Design 🡺

YAML File Syntax 🡺

all tasks are executed in sequential order

each task is processed one at a time

indentation is extremely important

No <tabs> in yaml file

empty lines have no value

No difference in double quotes or NO quotes for a task name :

YAML playbook files can be placed anywhere on the filesystem as along as they are being executed with absolute path

when a flat file is written in YAML format to execute task/plays then it is called playbook

there is NO need to modify the file permission

A picture containing application

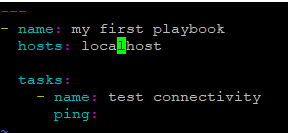
Description automatically generated

Creating First Playbook 🡺

su - root

mkdir /etc/ansible/playbooks

cd /etc/ansible/playbooks



check syntax of playbook

ansible-playbook --syntax-check <yml-file-name>

dry run

ansible-playbook --check <yml file name>

run /execute the playbook

ansible-playbook /absolute/path/of/the/yml/file.yml

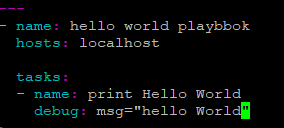
NOTE :

running ansible without a playbook , you can execute only #playbook

running ansible with a playbook you can execute, #ansible-plybook

Output Playbook🡺

write a playbook to print hello world 🡺



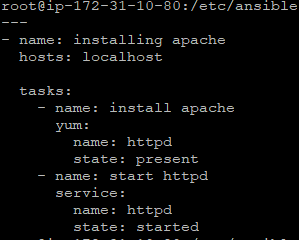
here we are using debug module

Multiple Task Playbook 🡺

Text

Description automatically generated

Playbook to install and starting a package in a unix box 🡺



Managing Remote Clients 🡺

all remote clients are considered inventory in Ansible

Ansible keeps its inventory information in host file located in /etc/ansible

the hots file is created during Ansible installation

ansible-playbook -i /home/nihar/ansible/hosts 🡺 calling host file in the command

hosts file & syntax 🡺

Diagram

Description automatically generated with medium confidence

Text

Description automatically generated with low confidence

Inventory host file can be either static or dynamic(using additional plugin)

ansible-inventory --list

OR

ansible all --list-hosts

Establish connection to remote clients from Control node 🡺

take ip address of the client

go to control node

run below commands in control node

cd /etc/ansible

vi hosts 🡺 insert the ip address

generate ssh key 🡺 ssh-keygen

copy the key to client 🡺 ssh-copy-id <ip address of client>

ssh-copy-id 192.168.0.

Check remote clients connectivity 🡺



copy files to remote clients 🡺

Chart

Description automatically generated with medium confidence

setup apache and open firewall port 🡺

Graphical user interface, text, application, email

Description automatically generated

once you execute the playbook

login back to client server

check httpd package status 🡺 rpm -qa | grep httpd

-check httpd package service status 🡺 systemctl status httpd

check firewalld service status 🡺 systemctl status firewalld

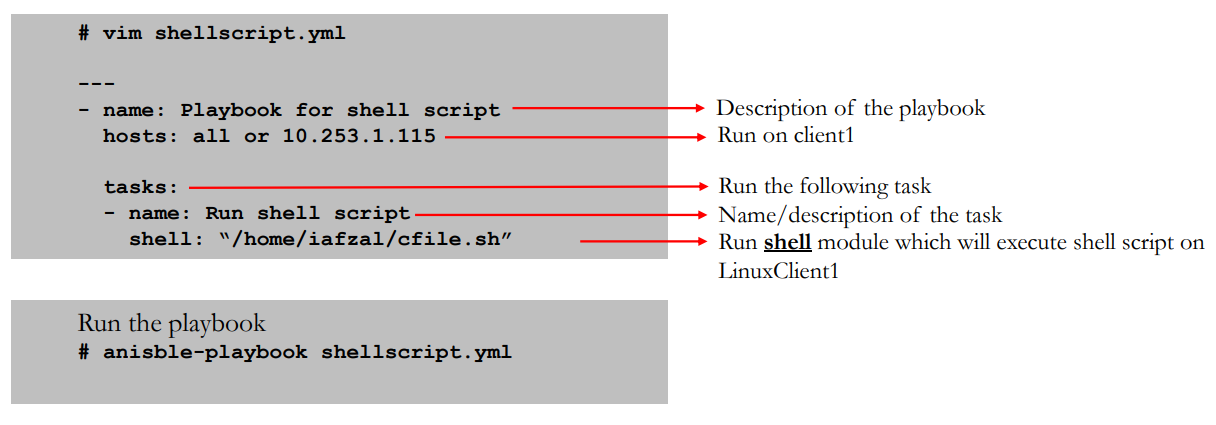
check if http service is enabled in firewalld 🡺 firewall-cmd --list-all

got to browser and put the ip addresss of the server to check if apache is running

Run Shell Scripts on Remote clients 🡺

the playbook will run shell scripts on the remote client (linux)

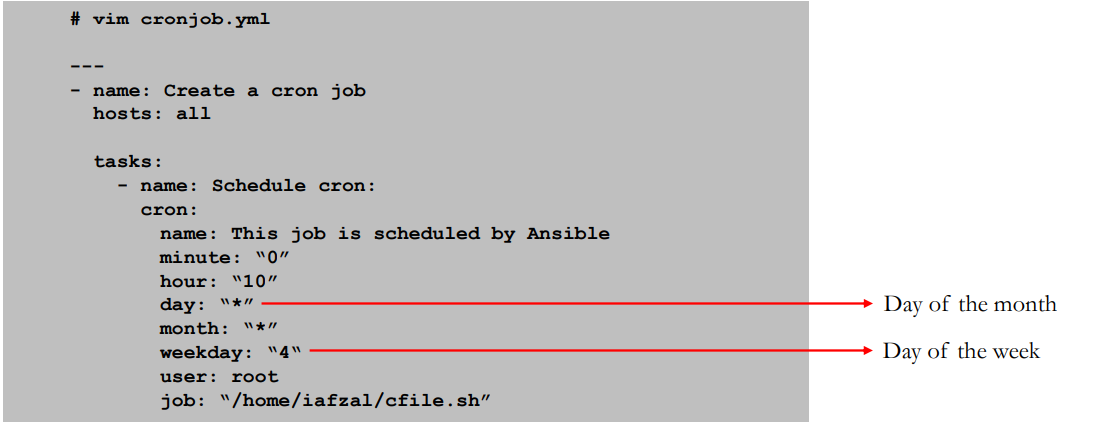
create a shell script in client server or get the path of the shell script from client server



Schedule a job (crontab) 🡺

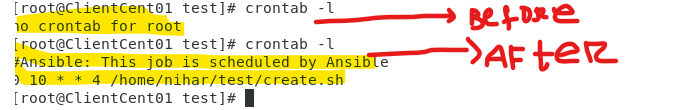
this playbook will schedule ajob as root

every Thursday at 10 pm



Text

Description automatically generated



User Account Management 🡺

This playbook will create a user George on remote clients

The user George will have a home directory /home/George

the shell environment for user George will be /bin/bash

A picture containing graphical user interface

Description automatically generated

Text

Description automatically generated

Add or Update User Password 🡺

this playbook will Add/update a password for user George

Ansible does not allow us to pass a cleartext password through the user module

Graphical user interface, text, application, email

Description automatically generated

to execute the playbook along with variable 🡺

ansible-playbook updateuser.yml --extra-vars newpasswd=George

Download package from URL 🡺

this playbook will create a directory for tomcat with required permission

download tomcat from url and place it in that directory with modified permissions.



Kill a Running Process 🡺

this playbook will find a running process by name

ignore any error

hold the result in registry variable

use shell module and run kill command to kill the registry variable

Graphical user interface, text, application

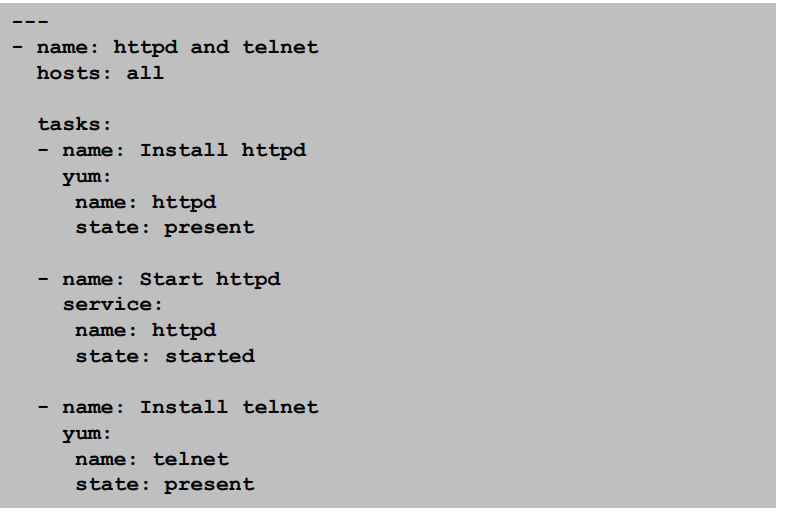
Description automatically generated

Pick & Choose steps 🡺

start a playbook at a specific task

ansible-playbook yamlfile.yml --start-at-task ‘Task Name’

ansible-playbook http.yml --start-at-task ‘Install telenet’



Ansible Automation with Ad-Hock Task 🡺

Ad-hoc commands run on as needed basis and usuall those tasks that don’t repeat.

syntax🡺

ansible [target] -m [module] -a “[module option]”

Ad-hoc commands

Ping localhost

# ansible localhost –m ping

Creating a file on all remote clients

# ansible all –m file –a “path=/home/iafzal/adhoc1 state=touch mode=700”

Deleting a file on all remote clients

# ansible all –m file –a “path=/home/iafzal/adhoc1 state=absent”

Copying a file to remote clients

# ansible all –m copy –a “src=/tmp/adhoc2 dest=/home/iafzal/adhoc2”

Installing package (telnet and httpd-manual)

# ansible all –m yum –a “name=telnet state=present”

# ansible all –m yum –a “name=httpd-manual state=present”.

Starting httpd package service

# ansible all –m service –a “name=httpd state=started”

Start httpd and enable at boot time

# ansible all –m service –a “name=httpd state=started enabled=yes”

Checking httpd service status on remote client

# ansible all –m shell -a “systemctl status httpd”

Remove httpd package

# ansible all –m yum –a “name=httpd state=absent”

OR

# ansible all –m shell -a “yum remove httpd”.

Creating a user on remote clients

# ansible all –m user –a “name=jsmith home=/home/jsmith shell=/bin/bash state=present”

To add a user to a different group

# ansible all –m user –a “name=jsmith group=iafzal”

Deleting a user on remote clients

# ansible all –m user –a “name=jsmith home=/home/jsmith shell=/bin/bash state=absent”

OR

# ansible all –m shell –a “userdel jsmith”

Getting system information from remote clients

# ansible all –m setup

You can run commands on the remote host without a shell module e.g. reboot client1

# ansible client1 –a “/sbin/reboot”

Roles🡺

roles simplifies long playbooks by grouping tasks into smaller playbooks.

the role are the way of breaking a playbook into multiple files. This simplifies writing complex playboks and it makes them easier to reuse.

writing ansible code to manage the same service for multiple environments creates more complexity and it becomes difficult to manage everything in one ansible playbook.

also sharing code among other teams become difficult . that is where ansible role helps solve these problems.

here we have created two roles. basic and full install

now we will call these roles in one playbook for specific hosts.

lets say you have 100 servers. for first 50 server you need basic install and last 50 server you need full install.

in this scenario, it should work.

Roles can be grouped by types server, types of applications or organizational requirements.

to create roles

go to control node

cd /etc/ansible/roles

make directory for each role

Ex: mkdir basicinstall

mkdir fullinstall

create sub directory tasks within each directory

mkdir basicinstall/tasks

mkdir fullinstall/tasks

create yml files within these sub directories

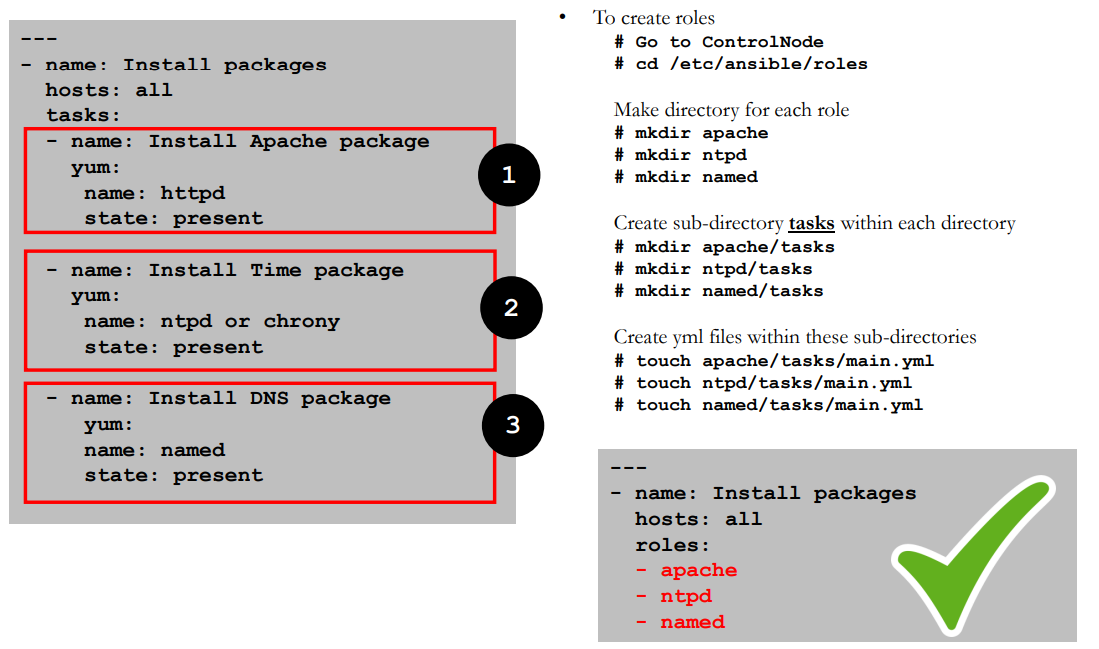
touch basicinstall/tasks/main.yml

touch fullinstall/tasks/main.yml

now create the playbook which will call these roles



Roles by application 🡺



Roles on Ansible Galaxy 🡺

you can find a ton of resourcses on roles through ansible galaxy

galaxy.ansible.com

you can search,filter and

download predefined roles from here.

downloaded roles will be in .ansible directry

Tags🡺

Tags are the references or aliases to a task.

instead of running an entre ansible playbook, use tags to target a specific taks you need to run.

To list all tags in a playbook

# anisble-playbook httpbytags.yml --list-tags

To run a task using tag

# anisble-playbook httpbytags.yml -t i-httpd

To skip a task using tag

# anisble-playbook httpbytags.yml --skip-tags i-httpd

We can use “tasks option” to start a playbook at a specific task but it will start executing the tasks from the task name mentioned. ex: in the below example, playbook will be start executing tasks from install httpd task to end of the file. However in tags, only specific task will be executed.

# anisble-playbook yamlfile.yml --start-at-task ‘Task name’

# anisble-playbook http.yml --start-at-task ‘Intall httpd’

Graphical user interface, application

Description automatically generated

Variable 🡺

Variables are like containers that hold the defined value which can be used repetitively IMPORTANT Things to Remember about Variables!

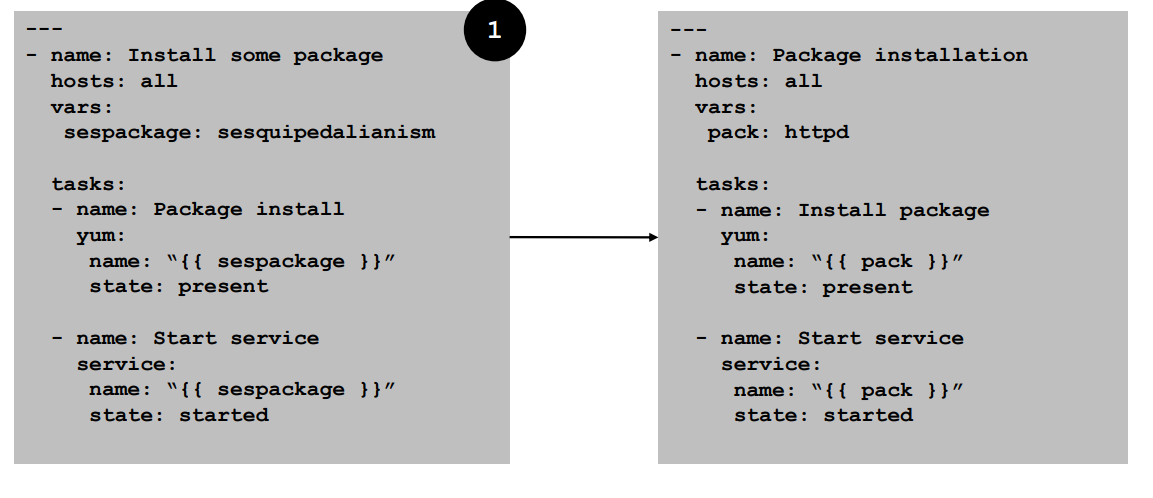
Name can include letters, numbers and underscore

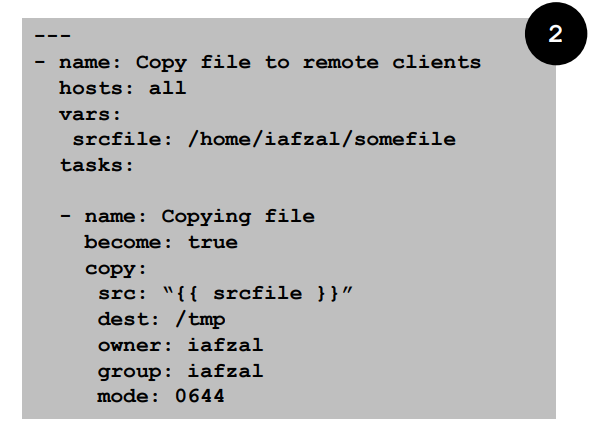
Name should always start with a letter

Cannot have a spaces, dots (.) or hypen (-) in variable name

Variables can be defined inside of inventory files as well

Examples 🡺





Variables in Inventory File 🡺

inventory file is the host file located in /etc/host

we can use variable in inventory file.

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Handlers 🡺

Handlers are executed at the end of the play once all tasks are finished. In Ansible, handlers are typically used to start, reload, restart, and stop services

• Sometimes you want to run a task only when a change is made on a machine. For example, you may want to restart a service if a task updates the configuration of that service, but not if the configuration is unchanged.

• Remember the case when we had to reload the firewalld because we wanted to enable http service? Yes, that is a perfect example of using handlers

• So basically handlers are tasks that only run when notified

• Each handler should have a globally unique name

Shape, arrow

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Firewalld Example 🡺

Table

Description automatically generated with low confidence

Conditions 🡺

Condition execution allow Ansible to take actions on its own based on certain conditions

Under condition certain values must be met before executing a tasks

We can use the WHEN statement to make Ansible automation more smart

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Loops 🡺

A loop is a powerful programming tool that enables you to execute a set of commands repeatedly

We can automate specific task but what if that task itself repetitive?

e.g. Changing permissions on hundreds of files

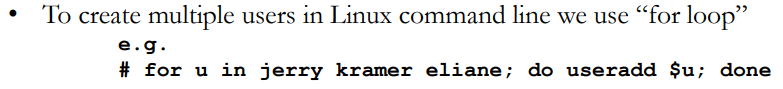
Creating multiple users at once

Installing many packages on hundreds of servers

Loops can work hand in hand with conditions as we loop certain task until that condition is met

When creating loops, Ansible provides these two directives: loop and with\_\* keyword.

adding multiple user Ex 🡺



Graphical user interface, text, application

Description automatically generated

Package example 🡺

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Ansible Vault 🡺

Ansible can automate tasks for teams such as

hardware

os

virualization

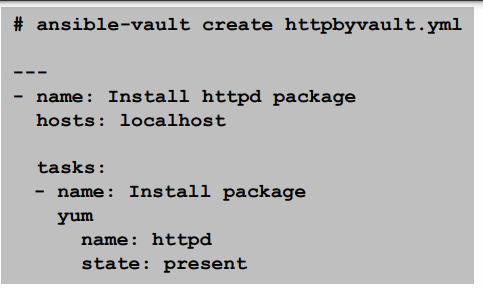
database or storage

application or softeware

Ofentimes you have to share ansible code with these groups over the networks and anything you share over network has a risk to end up in a wrong hands.

It is best practice to use Ansible vault feature which will password protect your code

ansible-vault create filename.yml 🡺 it wil primopt ofr password then open vi editor



Text

Description automatically generated

Text, letter

Description automatically generated

Encrypt strings within Playbook 🡺

Graphical user interface, text, application, email

Description automatically generated

Ansible management tools 🡺

Ansible management tool provides centralized web interface to manage your playbooks, inventories and everything you do through a command line

• These tools also manage multiple Ansible control nodes

• There are 2 Ansible management tools

• Ansible AWX (Free)

• Ansible Tower (Licensed product)

• AWX stands for Ansible WorkX

• AWX is a web application that provides a user interface, REST API, and task engine for Ansible

• The AWX allows you to manage Ansible playbooks, inventories, run reports and schedule jobs

• Ansible AWX is an open-source, community driven management tool

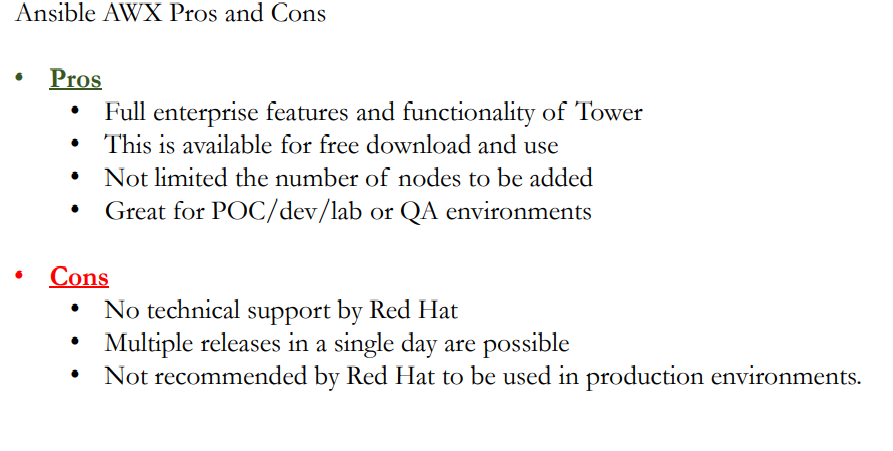
• It is the upstream product, meaning all the changes are done on AWX first before it get to Ansible Tower

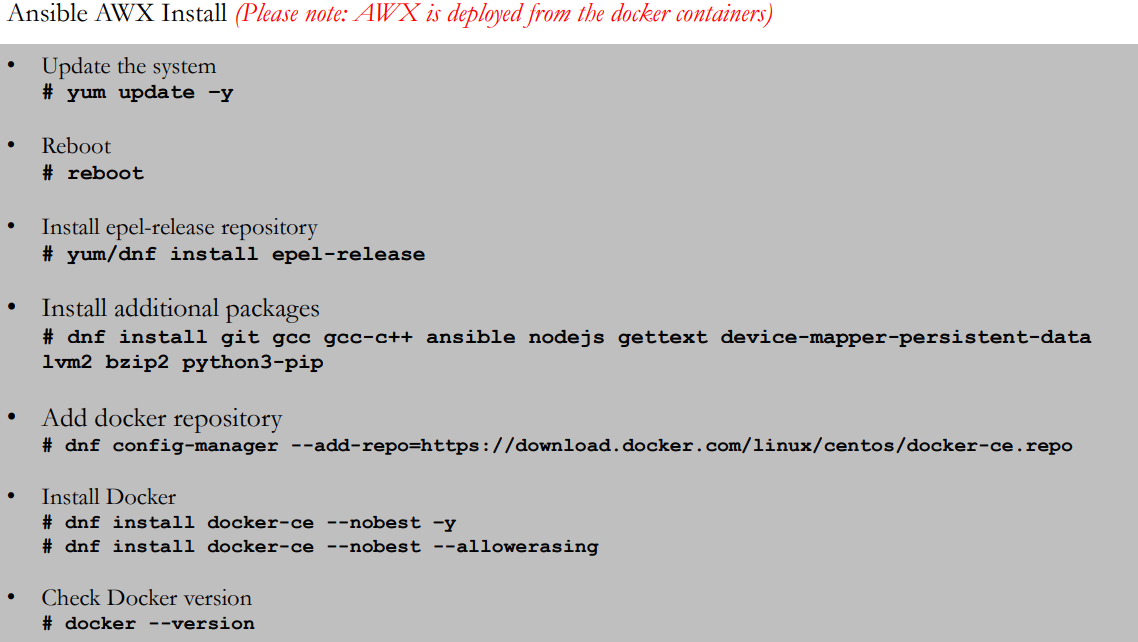
• The AWX Project is hosted on GitHub and Red Hat welcomes community contributions.

• The AWX project uses GitHub for its issue tracking as well. You can file your issues here:

• <https://github.com/ansible/awx/issues>

• Red Hat does NOT recommend AWX for production environments





Graphical user interface, text, application, email

Description automatically generated

Text

Description automatically generated

Graphical user interface, text

Description automatically generated

Graphical user interface, application, table, Excel

Description automatically generated

Ansible Tower 🡺

Ansible Tower is a commercial Ansible management product by Red hat which comes with support from Red Hat as well

Red Hat Ansible Tower is available in two editions that are differentiated by support and features.

Pricing is based on the number of nodes (systems, hosts, instances, VMs, containers or devices) that you are managing

Standard

Premium

<https://www.ansible.com/products/pricing>

Text, application

Description automatically generated

Ansible Tower Installation • <https://access.redhat.com/products/ansible-tower-red-hat>

Chart

Description automatically generated

INETRVIEW QUESTIONS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Certainly! Here's a list of 100 Ansible interview questions and answers for DevOps positions. These questions cover a broad range of Ansible topics, from basic concepts to advanced use cases:

### \*\*Ansible Basics:\*\*

1. \*\*Q: What is Ansible?\*\*

- \*\*A:\*\* Ansible is an open-source automation tool used for configuration management, application deployment, and task automation.

2. \*\*Q: How does Ansible differ from other configuration management tools?\*\*

- \*\*A:\*\* Ansible is agentless, uses a simple YAML syntax, and requires no special software on managed nodes.

3. \*\*Q: What is the inventory file in Ansible?\*\*

- \*\*A:\*\* The inventory file lists the hosts Ansible will manage, either statically or dynamically.

4. \*\*Q: Explain idempotence in the context of Ansible.\*\*

- \*\*A:\*\* Idempotence means that running a task multiple times has the same result as running it once. Ansible ensures idempotence to avoid unnecessary changes.

### \*\*Playbooks and Tasks:\*\*

5. \*\*Q: What is an Ansible playbook?\*\*

- \*\*A:\*\* A playbook is a YAML file containing a set of plays, which consist of tasks to be executed on remote hosts.

6. \*\*Q: How can you define variables in an Ansible playbook?\*\*

- \*\*A:\*\* Variables can be defined in the playbook or in separate variable files. They can also be passed as extra variables during playbook execution.

7. \*\*Q: Explain the purpose of the `handlers` section in Ansible playbooks.\*\*

- \*\*A:\*\* The `handlers` section contains tasks that are triggered by other tasks and are typically used to restart services or perform related actions.

8. \*\*Q: What is a task in Ansible?\*\*

- \*\*A:\*\* A task is a single unit of work defined within a playbook. It represents a single action to be performed on a remote host.

9. \*\*Q: How do you run a specific task in an Ansible playbook?\*\*

- \*\*A:\*\* Use the `--tags` option followed by the name of the task tag.

### \*\*Roles and Organization:\*\*

10. \*\*Q: What is an Ansible role?\*\*

- \*\*A:\*\* A role is a way to organize tasks, handlers, and variables in a structured directory hierarchy.

11. \*\*Q: How do you include a role in an Ansible playbook?\*\*

- \*\*A:\*\* Use the `roles` keyword in a playbook and specify the name of the role.

### \*\*Modules and Tasks:\*\*

12. \*\*Q: What is an Ansible module?\*\*

- \*\*A:\*\* A module is a reusable, standalone script that Ansible uses to perform tasks on remote nodes.

13. \*\*Q: Give an example of using the `copy` module in Ansible.\*\*

- \*\*A:\*\* Example task:

```yaml

- name: Copy a file

copy:

src: /path/to/source/file.txt

dest: /path/to/destination/

```

14. \*\*Q: How can you run a shell command on remote hosts using Ansible?\*\*

- \*\*A:\*\* Use the `command` or `shell` module. Example:

```yaml

- name: Run a shell command

command: ls /path/to/directory

```

### \*\*Inventory and Hosts:\*\*

15. \*\*Q: Explain the concept of dynamic inventories in Ansible.\*\*

- \*\*A:\*\* Dynamic inventories are scripts or programs that generate inventory information dynamically based on external sources.

16. \*\*Q: How can you limit the execution of a playbook to specific hosts?\*\*

- \*\*A:\*\* Use the `-l` option followed by the host or group name.

### \*\*Advanced Playbook Techniques:\*\*

17. \*\*Q: What is the purpose of the `ignore\_errors` keyword in Ansible tasks?\*\*

- \*\*A:\*\* `ignore\_errors` allows a playbook to continue running even if a task fails.

18. \*\*Q: How do you run asynchronous tasks in Ansible?\*\*

- \*\*A:\*\* Use the `async` and `poll` parameters in the task definition.

### \*\*Variables and Facts:\*\*

19. \*\*Q: What is the difference between Ansible variables and facts?\*\*

- \*\*A:\*\* Variables are user-defined values, while facts are pieces of information gathered from remote hosts.

20. \*\*Q: How can you access facts in an Ansible playbook?\*\*

- \*\*A:\*\* Facts are accessed using the `ansible\_facts` variable.

### \*\*Roles and Role-Based Configuration:\*\*

21. \*\*Q: How do you pass variables to an Ansible role?\*\*

- \*\*A:\*\* Define variables in the playbook and pass them to the role using the `vars` keyword.

22. \*\*Q: What is the purpose of the `default` directory in an Ansible role?\*\*

- \*\*A:\*\* The `default` directory contains default variables for the role.

### \*\*Ansible Vault:\*\*

23. \*\*Q: Why would you use Ansible Vault, and how do you create an encrypted file with it?\*\*

- \*\*A:\*\* Ansible Vault is used to encrypt sensitive information. To create an encrypted file, use `ansible-vault create filename.yml`.

24. \*\*Q: How do you edit an encrypted file using Ansible Vault?\*\*

- \*\*A:\*\* Use `ansible-vault edit filename.yml`.

### \*\*Error Handling and Troubleshooting:\*\*

25. \*\*Q: How can you handle errors in Ansible playbooks?\*\*

- \*\*A:\*\* Use the `failed\_when` and `ignore\_errors` keywords.

26. \*\*Q: How do you enable verbose mode in Ansible?\*\*

- \*\*A:\*\* Use the `-v` option for increased verbosity. Additional `-v` flags increase the verbosity level.

### \*\*Ansible Galaxy and Roles:\*\*

27. \*\*Q: What is Ansible Galaxy, and how do you use it?\*\*

- \*\*A:\*\* Ansible Galaxy is a platform for sharing and discovering Ansible content. You can use it to find roles, playbooks, and collections.

28. \*\*Q: How can you install an Ansible role using Ansible Galaxy?\*\*

- \*\*A:\*\* Use the `ansible-galaxy install` command. For example:

```bash

ansible-galaxy install geerlingguy.apache

```

### \*\*Continuous Integration (CI) and Ansible:\*\*

29. \*\*Q: How can you integrate Ansible playbooks into a Jenkins pipeline?\*\*

- \*\*A:\*\* Use Jenkins job steps to execute Ansible playbooks.

30. \*\*Q: Why is CI/CD important in the context of Ansible?\*\*

- \*\*A:\*\* CI/CD ensures automated testing and deployment of infrastructure changes.

### \*\*Security and Best Practices:\*\*

31. \*\*Q: How does Ansible handle security, and what best practices should be followed?\*\*

- \*\*A:\*\* Ansible uses SSH for communication and follows security best practices. Best practices include using Vault for sensitive data and ensuring idempotence.

32. \*\*Q: What is the significance of idempotence in Ansible playbooks?\*\*

- \*\*A:\*\* Idempotence ensures that running a playbook multiple times has the same result as running it once, preventing unnecessary changes.

### \*\*Advanced Topics

:\*\*

33. \*\*Q: What are Ansible Tower and AWX, and how do they differ from plain Ansible?\*\*

- \*\*A:\*\* Ansible Tower and AWX are web-based interfaces for managing Ansible playbooks. They provide additional features like job scheduling, RBAC, and a graphical dashboard.

34. \*\*Q: Explain the concept of dynamic inventories in Ansible, and how are they useful?\*\*

- \*\*A:\*\* Dynamic inventories allow Ansible to automatically discover and manage hosts based on external sources, such as cloud providers.

### \*\*Infrastructure as Code (IaC):\*\*

35. \*\*Q: How does Ansible contribute to Infrastructure as Code (IaC)?\*\*

- \*\*A:\*\* Ansible allows infrastructure to be defined and managed as code, providing version control, repeatability, and automation.

### \*\*Real-World Scenarios:\*\*

36. \*\*Q: Provide an example of how you would deploy a web application using Ansible.\*\*

- \*\*A:\*\* Create a playbook that installs web server software, copies application code, and configures the necessary settings.

37. \*\*Q: In what scenarios would you use Ansible to automate tasks in a cloud environment?\*\*

- \*\*A:\*\* Ansible can be used to automate the provisioning and configuration of virtual machines, containers, and other resources in cloud environments.

### \*\*Ansible Tower and AWX:\*\*

38. \*\*Q: How does Ansible Tower improve automation workflows compared to running Ansible from the command line?\*\*

- \*\*A:\*\* Ansible Tower provides a web-based interface, scheduling, RBAC, and job tracking, enhancing the management of automation workflows.

39. \*\*Q: What are some advantages of using Ansible Tower in a large-scale environment?\*\*

- \*\*A:\*\* Ansible Tower provides centralized control, audit trails, and delegation of tasks, making it easier to manage automation at scale.

### \*\*Real-World Projects:\*\*

40. \*\*Q: Share an experience where you applied Ansible to solve a specific problem or automate a task.\*\*

- \*\*A:\*\* Describe a real-world project where Ansible played a crucial role, such as system provisioning, configuration management, or application deployment.

### \*\*Community Involvement:\*\*

41. \*\*Q: How do you stay informed about Ansible updates, new features, and best practices?\*\*

- \*\*A:\*\* Mention participating in Ansible forums, attending meetups, reading blogs, and contributing to the Ansible community.

42. \*\*Q: Have you contributed to Ansible or any Ansible projects on GitHub?\*\*

- \*\*A:\*\* Share any contributions or experiences in the Ansible open-source community.

### \*\*Certification and Training:\*\*

43. \*\*Q: Have you pursued any Ansible certifications or training programs?\*\*

- \*\*A:\*\* Mention any relevant certifications, such as the Red Hat Certified Engineer in Ansible Automation (EX407).

44. \*\*Q: How has your Ansible knowledge and skills evolved over time, and what steps have you taken to improve?\*\*

- \*\*A:\*\* Discuss your learning journey, including courses, certifications, and practical experiences that contributed to your Ansible proficiency.

### \*\*Continuous Learning:\*\*

45. \*\*Q: How do you plan to stay updated on emerging Ansible trends and features?\*\*

- \*\*A:\*\* Share your commitment to continuous learning through reading documentation, following release notes, and exploring new Ansible features.

### \*\*Scenario-Based Questions:\*\*

46. \*\*Q: Given a scenario where you need to deploy a multi-tier application, how would you structure your Ansible playbook and roles?\*\*

- \*\*A:\*\* Discuss how you would organize tasks into roles, handle dependencies, and manage configuration for each tier.

47. \*\*Q: In a scenario where you need to update sensitive configuration files across multiple servers, how would you use Ansible Vault to secure the information?\*\*

- \*\*A:\*\* Explain how Ansible Vault can be used to encrypt sensitive data, such as passwords or API keys, and provide an example.

### \*\*Advanced Playbook Techniques:\*\*

48. \*\*Q: Describe a situation where you needed to use asynchronous tasks in Ansible, and how did you implement them?\*\*

- \*\*A:\*\* Discuss scenarios involving long-running tasks, such as large file transfers or database migrations, and how asynchronous tasks were used.

49. \*\*Q: How would you handle conditional execution of tasks in Ansible based on specific criteria? Provide an example.\*\*

- \*\*A:\*\* Explain how to use the `when` keyword in tasks to conditionally execute them, and provide an example based on a specific condition.

### \*\*Ansible Modules:\*\*

50. \*\*Q: Provide examples of modules you frequently use in Ansible playbooks and their purposes.\*\*

- \*\*A:\*\* Mention modules like `apt`, `yum`, `copy`, `template`, `service`, etc., and explain their use cases.

51. \*\*Q: In what scenarios would you write custom Ansible modules, and how would you go about it?\*\*

- \*\*A:\*\* Discuss scenarios where existing modules are insufficient and explain the process of writing custom modules in Python.

Q: How do you execute a shell command using Ansible?

A: Use the command or shell module. For example: ansible all -m command -a "ls /path/to/directory"

### \*\*Variables and Facts:\*\*

52. \*\*Q: How do you manage and prioritize Ansible variables when dealing with multiple sources, such as playbooks, roles, and external variable files?\*\*

- \*\*A:\*\* Explain variable precedence in Ansible and how variables are managed across different sources.

53. \*\*Q: Can you elaborate on how you use Jinja2 templating in Ansible playbooks for dynamic configuration?\*\*

- \*\*A:\*\* Discuss how Jinja2 templating is used to dynamically generate configurations based on variables and facts.

### \*\*Roles and Role-Based Configuration:\*\*

54. \*\*Q: Describe a scenario where you needed to reuse a complex set of tasks across multiple playbooks. How did you implement role-based configuration in Ansible?\*\*

- \*\*A:\*\* Discuss a situation where roles improved code reuse, organization, and maintenance in your Ansible environment.

55. \*\*Q: How do you manage role dependencies, and can you provide an example of a playbook with roles and dependencies?\*\*

- \*\*A:\*\* Explain how dependencies are defined in the `meta/main.yml` file of a role and provide an example playbook.

### \*\*Advanced Playbook Techniques:\*\*

56. \*\*Q: In a scenario where you need to handle errors gracefully, how would you structure your Ansible playbook?\*\*

- \*\*A:\*\* Discuss how to use `ignore\_errors`, `failed\_when`, and handlers to handle errors and gracefully recover from failures.

57. \*\*Q: Explain how you can make use of dynamic inventories in Ansible to automatically discover and manage hosts.\*\*

- \*\*A:\*\* Describe how dynamic inventories can be scripts or programs that generate inventory information based on external sources, such as cloud providers.

### \*\*Ansible Vault:\*\*

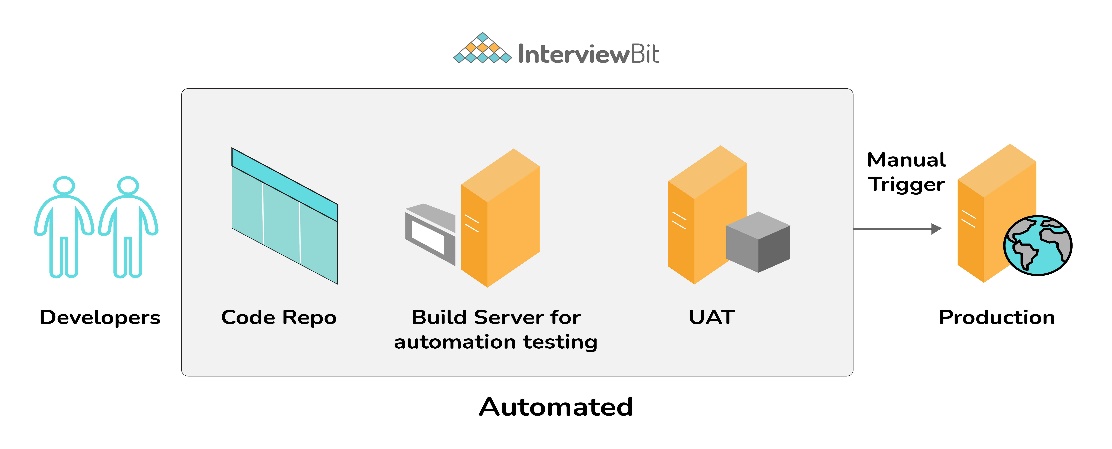
58. \*\*Q: In a scenario where you need to store sensitive information like database passwords, how would you use Ansible Vault to secure the data?\*\*

-

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.

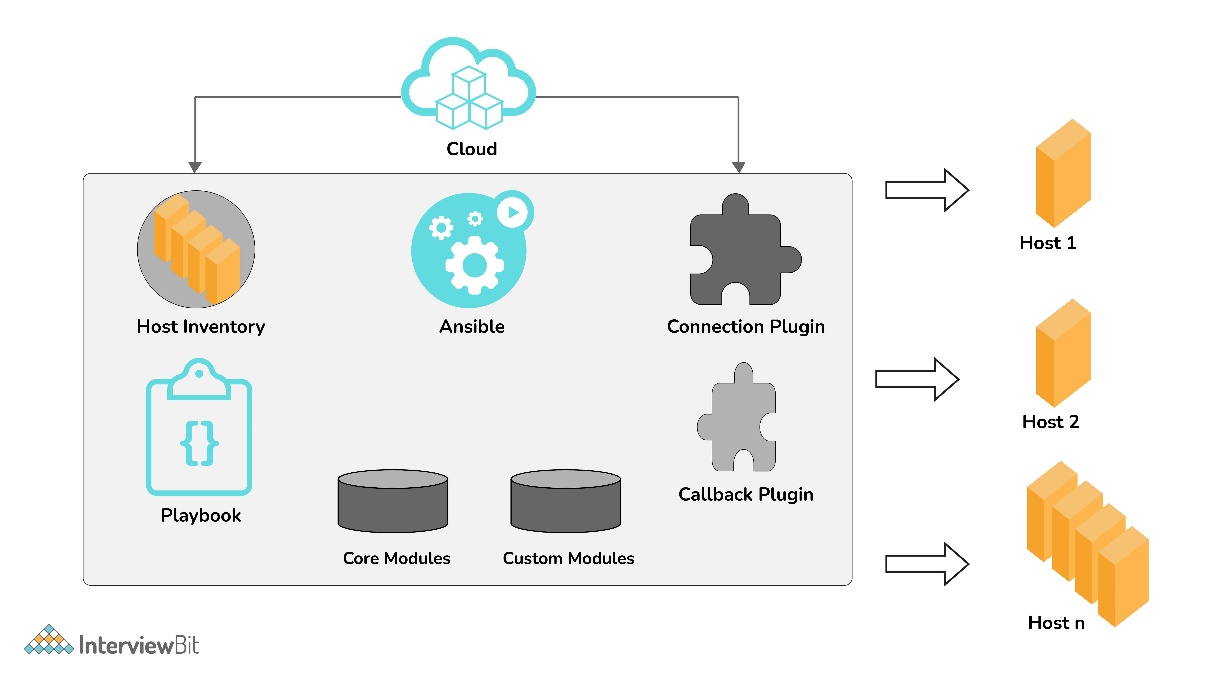
3.How does Ansible work?

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.

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.

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.

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.

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:

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.

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.

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:
* boolean: true
* integer: 1
* alias: aliases are like variables

{

"object": { "key": "value", "array": [

{

"null\_value": null

},

{

"boolean": true

},

{

"integer": 1

},

{

"alias": "aliases are like variables"

}

]

}

}

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>

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

---

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

// We need to declare Topic class with necessary attributes such as name, total\_score, 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);

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.

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](mailto: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.

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](mailto: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

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.

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

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

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 %}.

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]

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

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'] }}

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.

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.

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 }}"

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.

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.

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', 'm

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

python -c "from passlib.hash import sha512\_crypt; import getpass; print(sha512\_crypt.us

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.

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.

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

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

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

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.

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\_fi 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.

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\_hostn

...

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