**­­Monolithic** – A monolithic application is built as a single unified unit. (**Git, Jenkins, maven, Ansible, Terraform**)

**Micro Services –** Architecture is a collection of smaller, independently deployable service. (**Terraform, Docker, Kubernetes**)

**Reference - GitHub link** - <https://github.com/RAHAMSHAIK007/2023-04-27/tree/main>

**Interview Questions Ref GitHub** - https://github.com/Divinreddy

**Ansible:**

* Tool: Ansible
* Year: 2012
* Dependency: Python
* Purpose: Configuration Management & Deployments.
* Code: YAML

🡺Its an **Automation** Configuration and management tool.

* Configuration: **Hardware & Software process.**
* Management: **Update, Install, Uninstall**…!

🡺Ansible was taken over by red hat.

🡺Code: YAML (Yet Another Markup Language)

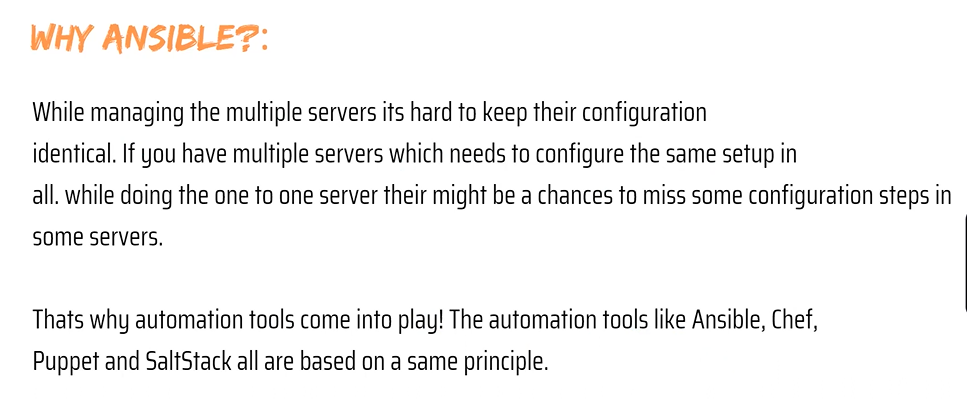
* Year: 2012 -->Miachel Dehhan Developed and it dependency ---> Python.

🡺**Infrastructure** or **INFRA** ---> Resources used to run our application on the cloud.

* Ex: Servers, VPC, Autoscaling group (Asg), Elastic load balancer (Elb)

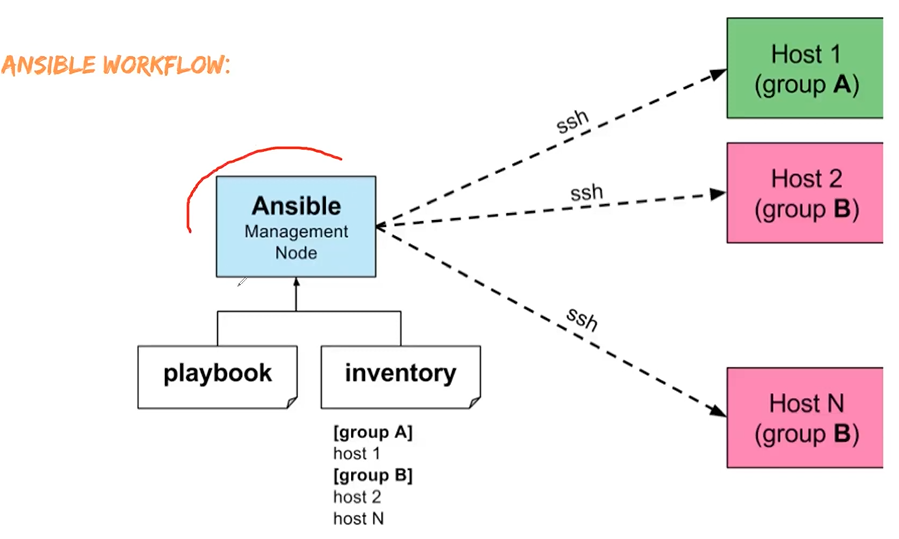
**What is Ansible?**

**🡺**Ansible is an open-source tool that automates **infrastructure** provisioning, application deployment, and configuration management. Ansible provides two ways for users to manage servers: through ad-hoc commands and playbooks. This article focuses on how to configure servers using playbooks.



🡺Ansible User Plain SSH. So, nothing needs to install on client machines. But other automation tools like Chef/Puppet needs to install agent on client machines when we need to perform a task.

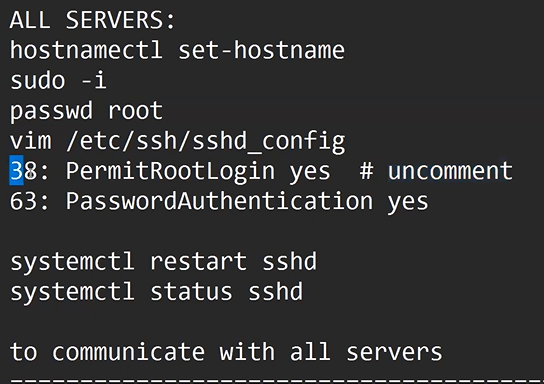
🡺Ansible is light weight, relative to use and speed of deployment compared to other tools. Ansible handle either via standard SSH commands, or the paramo module which provides a python interface to SSH2.



## What is SSH?

SSH, also known as Secure Shell or Secure Socket Shell, is a [network protocol](https://www.techtarget.com/searchnetworking/definition/protocol) that gives users, particularly system administrators, a secure way to access a computer over an unsecured network.

SSH -https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-connect-set-up.html



A screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

**inventory host pattern** ---> The Ansible inventory file defines the hosts and groups of hosts upon which commands, modules, and tasks in a playbook operate. The file can be in one of many formats depending on your Ansible environment and plugins.

**Root user** ---> Not used Sudo command.

Non-Root User ---> You must and should use Sudo command.

Ansible **ADHOC** commands are one-liner Linux shell commands and **playbooks** are like a shell script, a collective of many commands with logic.

Ansible **ADHOC** commands come handy when you want to perform a quick task.

**Color: Identify terminal flow.**

1. Red: Failed.
2. Yellow: Success
3. Green: Already Done
4. Blue: Skip

A screenshot of a computer

Description automatically generated

**Videos – 44**

🡺 A module is a reusable, standalone script that Ansible runs on your behalf, either locally or remotely. Modules interact with your local machine, an API, or a remote system to perform specific tasks like changing a database password or spinning up a cloud instance.

🡺It is in the Key – Value format and it is Reusable.

**Command different’ s**

**ADHOC Command**: ansible all -a “yum install git -y”

Module Command:

ansible all -m yum -a “pkg=git state=present” -----> # (present mean = Install)

ansible all -m yum -a “pkg=git state=latest” ---------># (Latest = Update)

ansible all -m yum -a “pkg=git state=absent” -------> # (absent = Uninstall)

**Set-Upping Httpd Service.**

ansible all -m yum -a “pks=httpd state=present”

ansible all -m service -a “pks=httpd state=restarted” -----> # (Restarted – restart the service)

ansible all -m service -a “pks=httpd state=stopped” ----># (Stopped – stop the service)

ansible all -m user -a “name= RCB state=present”

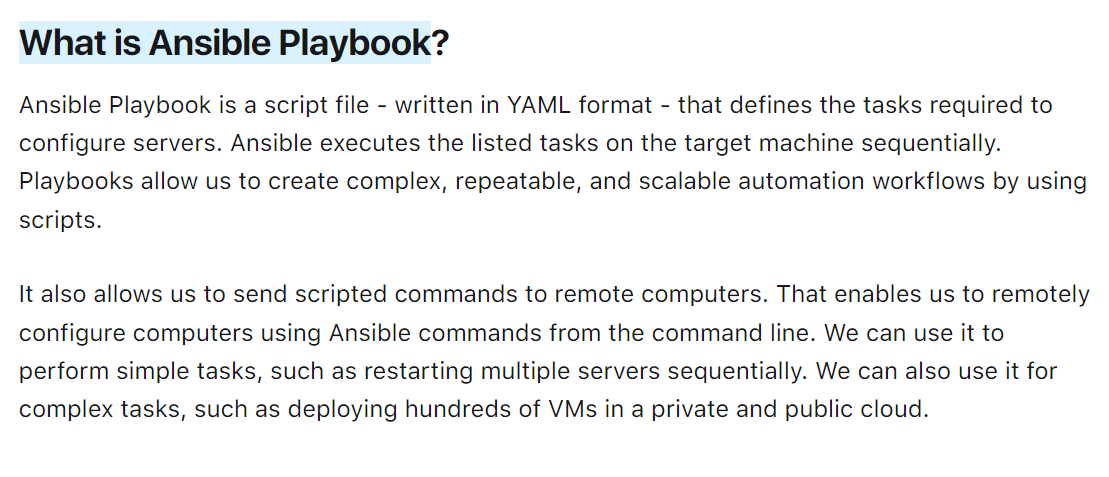
**Limitation Modules.**

1. We cannot Reuse the Modules.
2. We cannot Execute multiple Modules.

**Playbook**.

**How do Ansible Playbooks work?**

🡺Ansible Playbooks are lists of tasks that automatically execute for your specified inventory or groups of hosts. One or more Ansible tasks can be combined to make a play—an ordered grouping of tasks mapped to specific hosts—and tasks are executed in the order in which they are written. A playbook can include one or more play.



**Playbook** ----> It’s a collection of modules.

* Here we can multiple modules & we can reuse it.
* Playbook is written on yaml language.
* It works on key- value and its extension (Yaml /Yml)

**Created Playbook ---> File name - raham.yml**

**Vim raham.yml**

**Module Command reference:**

ansible all -m yum -a “pkg=git state=present” -----> # (present mean = Install)

**playbook reference: single task**

* hosts: all

tasks:

- name: Installing Git

yum: name or pkg = Git state=present

: wq!

**Extrusion playbook command ---> ansible-playbook raham.yml**

**Multiple modules or multitask using in playbook set-upping HTTPd service.**

**Step:1----> vim** raham.yml

**Step2:** Inside the raham.yml file we can write the task flow.

* hosts: all

tasks:

- name: Installing httpd

yum: name or pkg = httpd state=present

- name: restarting httpd

service: name = httpd state=restarted

- name: create a user

User: name= raja state=present

**:wq!**

**Step3:** ansible playbook raham.yml

**A screenshot of a computer

Description automatically generated**

**🡺ansible all -a “yum remove git\* https\* java-1.8.0-openjdk\* tree\* -y”**

**Videos- 45**

🡺IF you have 100 service who we need to find through service run or not

* **ansible -m ping all or ansible -m ping <piratical service group name>**

**Shell (Vs) Command (vs) Raw module**

**Step:1----> vim** raham.yml

**Step2:** Inside the raham.yml file we can write the task flow.

* hosts: all

tasks:

- name: task-1

yum: name = git state=present

- name: task -2

Shell: yum install httpd -y

- name: task -3

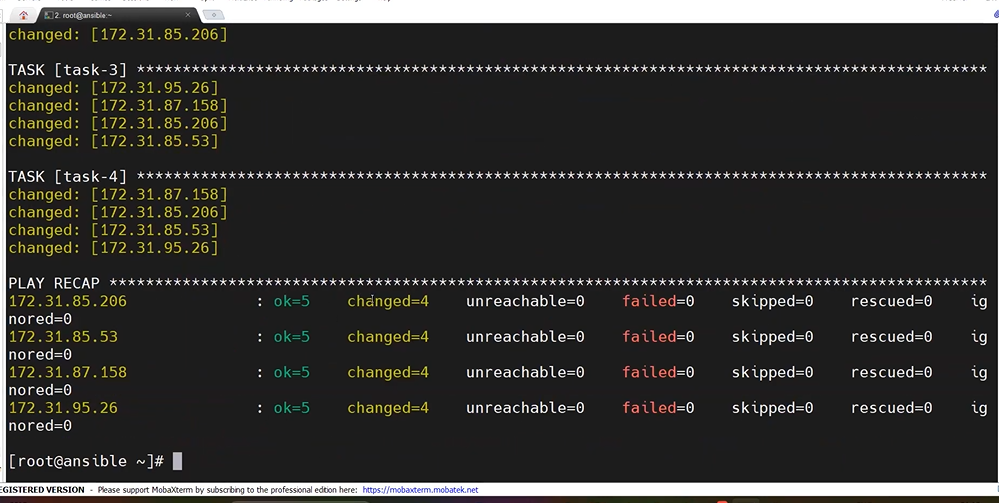
Command: yum install tree -y

- name: task-4

raw: yum install java-1.8.0-openjdk -y

:wq!

**Step3:** ansible playbook raham.yml



**Uninstall Service**

**Step:1----> vim** raham.yml

**Step2:** Inside the raham.yml file we can write the task flow.

* hosts: all

tasks:

- name: task-1

yum: name = git state=absent

- name: task -2

command: yum remove httpd\* -y

- name: task -3

Command: yum remove tree\* -y

- name: task-4

raw: yum remove java-1.8.0-openjdk\* -y

:wq!

**Step3:** ansible playbook raham.yml

**TAGS**: - It will Execute/ skip the **particular** task among **multiple** tasks.

**Ex: - Single Tags execution** only install HttpD service

**Step:1----> vim** raham.yml

**Step2:** Inside the raham.yml file we can write the task flow.

* hosts: all

tasks:

- name: task-1

yum: name = git state=present

tags: A

- name: task -2

Shell: yum install httpd -y

tags: b

- name: task -3

Command: yum install tree -y

tags: C

- name: task-4

raw: yum install java-1.8.0-openjdk -y

tags: D

:wq!

**Step3:** ansible playbook raham.yml --tags B

**Ex: - Multe Tags execution** only install tree and java service

**Step1 – above task flow step same**

**Step2 - Above task flow step same**

**Step3 -** ansible playbook raham.yml --tags C D

**Skip-Tags** Ex: - If 5 tags or 5 service are there in that only 4 Service want to run application.

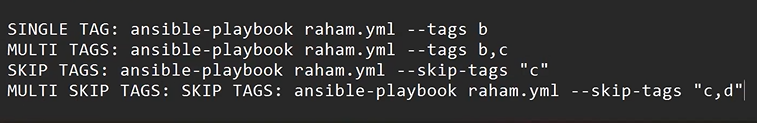
🡺With help of **Skip-Tags** we can do that.

**Step1 – above task flow step same**

**Step2 - Above task flow step same**

**Step3 -** ansible playbook raham.yml – skip-tags ‘’C”

:wq!



======================\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*========================

**Variables 🡪 Which will vary, and it will store the value.**

**Dynamic vars: -** The vars which can be passed **outside** of the playbook during execution **or** Runtime.

**Step:1----> vim** raham.yml

**Step2:** Inside the raham.yml file we can write the task flow.

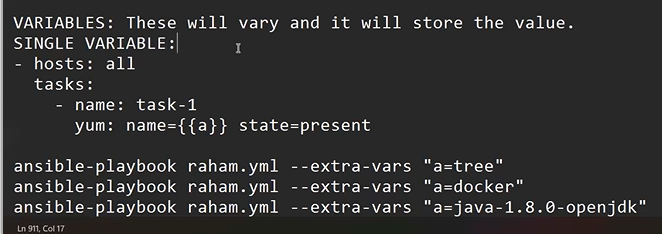
* hosts: all

tasks:

- name: task-1

yum: name = {{a}} state=present

**Step3:** ansible playbook raham.yml –extra-vars ‘’a=docker”



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Multle

**Step:1----> vim** raham.yml

**Step2:** Inside the raham.yml file we can write the task flow.

* hosts: all

tasks:

- name: task-1

yum: name = {{a}} state=present

- name: task-2

Yum: name = {{b}} state=present

**Step3:** ansible playbook raham.yml --extra-vars ‘’a=tree b=docker”

**\*\*Static Variable: 🡪** These variables are declared inside the playbook.

**Step:1----> vim** raham.yml

**Step2:** Inside the raham.yml file we can write the task flow.

* hosts: all

vars:

a: Git

b: Maven

tasks:

- name: task-1

yum: name = {{a}} state=present

- name: task-2

Yum: name = {{b}} state=present

**(OR)**

tasks:

- name: task-1

yum: name = {{a}} state=present

yum: name = {{b}} state=present

**Step3:** ansible playbook raham.yml

**Stopping Gather facts**

**-** hosts: all

gather\_facts: false

vars:

a: git

b: maven

tasks:

-name: task-1

Yum: name= {{a}} state= present

Yum: name= {{b}} state=present

**Step3:** ansible playbook raham.yml

**Videos- 46**

**1. Cluster: Group of nodes / Servers.**

**Homogenies Cluster: All service are same flavours.**

**Heterogenies Cluster: All service are different flavours.**

**Ex: Flavours –** Red hat, Ubuntu ect…

🡺There are 4 service are there in that 2 are red hat and 2 are Ubuntu mean.

It called **Heterogenies**.

🡺 If all services are red hat or Ubuntu through service called **homogenies**.

**Conditions:** When the heterogeneous cluster we will used these conditions.

Task -2 Ex: if you have one playbook and two different service (Redhat, Ubuntu) who will running the services.

🡺To solve this problem we are using **When** connecting see the below task 1 represent if service Redhat can install or setup service and task-2 will be skip the setup flow .why because we are using redhat service right now.

**Step1**: Vim raham.yaml

**Step2**: Inside the file we can write task’s flow.

- hosts: all

Tasks:

- name: task-1

Command: yum install tree -y

When: ansible\_os\_family == “RedHat”

- name: task-2

command: apt install tree -y

When: ansible\_os\_family == “Debian”

:wq!

**Step3**: - ansible-playbook raham.yaml

**Handlers**: It will execute when called by other tasks.

🡺One task is depending on another task.

**Notify mean** – First task executed then second task executed.

**Ex:**

**- hosts: all**

**tasks:**

**- name: Install httpd**

**command: yum install httpd -y**

**notify: restart httpd**

**handlers:**

**- name: restart httpd**

**Command: systemctl restart httpd**

**: wq!**

**Loops: It will do Operation recursively or reduce length of the code**

**🡺** Ansible loop is used to repeat any task or a part of code multiple times in an Ansible-playbook. It includes the creation of multiple users using the user module, installing multiple packages using apt or yum module or changing permissions on several files or folders using the file module**.**

**Ex: -**

**- hosts: all**

**tasks:**

**- name: task-1**

**yum: name={{item}} state=present**

**With\_items:**

**- git**

**- maven**

**- docker**

**- tree**

**: wq!**

**Ex-2: Created User’s:**

**- hosts: all**

**tasks:**

**- name: task-1**

**user: name={{item}} state=present**

**With\_items:**

**- RCB**

**- CSK**

**- RR**

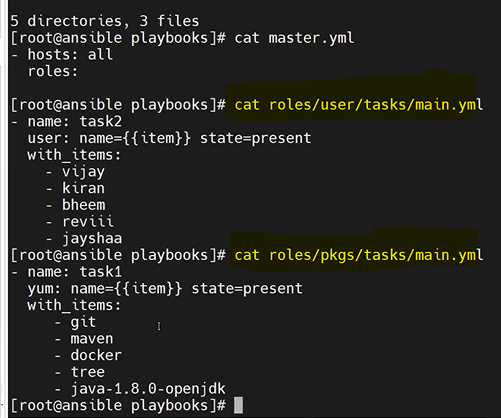
**- SRH**

**: wq!**

**ROLES: -----> It will create a directory structure for our playbooks.**

**------> It will encapsulate the playbook.**

**------> It will be used for community as well.**



**Ex:** **Two playbook are there – one playbook installs service and other one created user’s**

**Step1**: **mkdir playbook** ------> Created one playbook dir

**Step2**: **cd playbook/** ---> change the director or move the playbook dir.

**Step3**: **Vim master. yaml** ------> Created a folder or file master.

**Step4**: Inside the master. Yml file to write some tasks.

**- hosts: all**

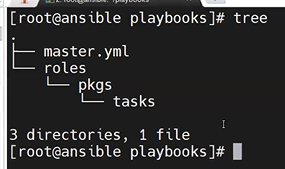
**roles:**

**:wq!**

**Step5: mkdir -p roles/pkgs/tasks** -------------------> Other folder cratering name as **roles** and for ex we need to install any package that file name as – **pkgs**, -**p** director inside director can give -p. and other sub-director created name as – **Tasks.**

**Step6**: **yum install tree -y**

**Step7**: **tree**

****

**Step8**: **vim roles/pkgs/tasks/main.yml** ---------------------> Again one more file creation name – main.yml.

**Step9**: Inside the main.yml file write task.

**- hosts: all**

**tasks:**

**- name: task-1**

**yum: name={{item}} state=present**

**With\_items:**

**- git**

**- maven**

**- docker**

**- java-1.8.0-openjdk**

**: wq!**

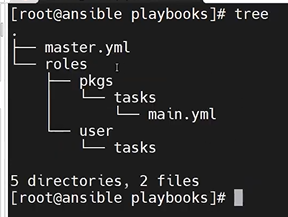
**Step10**: **tree**

A screenshot of a computer screen

Description automatically generated

**Step11**: **mkdir -p roles/user/tasks** -----------------> To created user directory/ roles inside the roles.

**Step12**: **tree**



**Step13**: **vim roles/user/tasks/main.yml**

**Step14**: Inside the main.yml write the task.

**- name: task-2**

**user: name={{item}} state=present**

**With\_items:**

**- RCB**

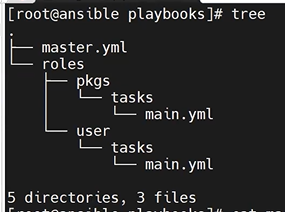
**- CSK**

**- RR**

**- SRH**

**: wq!**

**Step15: tree**

****

**Step16: cat master.yml**

**A black screen with white text

Description automatically generated**

**Step17:** **cat roles/user/tasks/main.yml**

A computer screen with white text

Description automatically generated

**Step18**: **cat roles/pkgs/tasks/main.yml**

A computer screen shot of a black screen

Description automatically generated

**Step19**: **Vim master. yaml**

**Step20**: inside the master file we wite task flow. ----------> only install the package in the service’s

**-hosts: all**

**roles:**

**- pkgs**

**: wq!**

**Steps21**: **ansible-playbook master.yml**

**Galaxy**: **It will store the roles of multiple users in a central place.**

🡺 **Ansible-galaxy search java**

**🡺 ansible-galaxy install baztian**

**🡺 ansible-galaxy search –author baztian**

**Videos- 47**

**DEBUG**: **It is used to print the message. Or debug module is used to print the output to the terminal.**

**Ex: vim abc.yaml**

**- hosts: all**

**tasks:**

**-name: printing a msg**

**debug:**

**msg: “Hello world”**

**: wq!**

**🡺ansible-playbook abc.yaml**

**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@**

**To install service, we are calling name list:**

**Redhat: ---------> yum/rpm**

**Ubuntu: ----------> apt/apt-get**

**Python: -----------> pip module**

**NumPy & Pandas ------------> Libs**

**Kartikeya soft:**

**Ansible: 🡪Day: - 37**

**Step1:** Creating 2 EC2 instance name as (1. Ansible-Main and 2. Ansible\_ Node1)

**Step2:** Connection through SSH key

🡺 $ **ssh -i k8s.pem ec2-user@privite IP address**

**🡺 k8s.pem** (we need to give you pem file name)

**Step3:** Moving to root user.

🡺 **sudo su –**

**Step4:** Installing ansible. (Portal web sitelink: -https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-ansible-on-centos-7 )

🡺 **yum update -y**

**🡺 sudo yum install epel-release**

🡺Copy the line To Use, Run command.

🡺 **sudo amazon-linux-extras install epel**

🡺 **sudo yum install ansible**

**Step5:** checking- **ansible --version** – **2.9.27**

**Hint: -** Ansible is a tool written in Python, and it uses the declarative markup language YAML to describe the desired state of devices and configuration.

**Step6**: Creating inventory 🡺 **vim inventory**

**Step7:** Inside the inventory-file write.

🡺Giving nodes host ip address

[dev]

192.168.1.2

192.168.1.3

[Test]

10.1.0.1

10.2.0.1

[WEB]

IND-WEB-01

IND-WEB-01

:wq!

**Step8**: To call the Group of nodes - inventory command.

🡺**ansible -i inventory all --list-hosts**

**Step9:** To creating any number of group inventory.

**Step10**: **vim prod-inv**

**Step11:** Inside the prod-inv file write the service IP address.

[PROD]

172.16.2.1

172.16.3.1

172.16.4.1

: wq!

**Step12**: Calling the inventory file.

🡺**ansible -i prod-inv all --list-hosts**

**Day: - 38 --> one node to another node connecting.**

**Step13:** Creating 2 EC2 instance name as (1. Ansible-main and 2. Ansible-Node1)

**FirstNode** :Ansible-Main IP Address- 3.81.56.255

Flowing the previous class steps (**Step: 3, 4, 5**) – Already Ansible service are there mean we can skip these steps.

**Step14:** Connecting Second Node (Node name: Ansible-Node1, IP Address -54.161.131.80)

**Note**: We configure the First Node to second Node connecting.

**Step15:** Switch the ec2-user command**.**

**🡺 exit (**By Default it will switch main user)

**Step16:** Inside the main user (ec2-user) creating the one directory.

**🡺 mkdir ksworking**

**Step17:** And moving or change to Ksworking dir.

**🡺 cd ksworking/**

**Step18:** Again, creating directory inside the ksworking.

**[ec2-user@ip address ksworking] $ 🡺mkdir ksdev**

**🡺mkdir kstest**

**🡺mkdir ksweb**

**🡺mkdir ksapp**

**🡺mkdir ksprod**

**Step19:** list the file inside the ksworking.

**🡺 ls -------------------------> O/P: ksdev kstest ksweb ksapp ksprod**

**Step20:** To change directory ksdev.

**🡺 cd ksdev/**

**Step21:** Inside the ksdev we are creating inventory.

**ec2-user@ip address ksdev] $ vim inv**

**Step22:** Inside the inv file we add group node or IP Address. (Giving the second node private IP Address)

**🡺 [Dev]**

**172.31.36.135**

**: wq!**

**Step23:** To call inventory or check service set.

**🡺ansible -i inv Dev –list-hosts**

**O/P: - hosts (1):**

**172.31.36.135**

**Step24:** To check the Ansible config file = /ect/ansible/ansible.cfg

**🡺ansible--version**

**Step25:** To change ansible directory.

**ec2-user@ip address ksdev] $ 🡺cd /ect/ansible/**

**Step26:** To check the list file inside ansible file.

**ec2-user@ip address ansible] $ ls ------------> O/P: ansible.cfg hosts roles**

**Note: what is your default inventory ------>Ans: - Hosts**

**Step26:** To check the default inventory inside the ansible server Uncommented node.

🡺**sudo** **vim /ect/ansible/hosts**

**🡺Inside the file uncomment node (192.168.100.1)**

**: wq!**

**🡺cd**

[ec2-user@ip address ~] $ **🡺ansible all –list-hosts**

**O/P: - hosts (1):**

**192.168.100.1**

**Step27:** move or change direction where that the placing cfg file.

[ec2-user@ip address ~] $ 🡺cd ksworking/

[ec2-user@ip address ksworking] $ 🡺cd ksdev/

[ec2-user@ip address ksdev] $ 🡺ls ---------------------> O/p: **inv**

**Step28: Know we are communicating the second node server(Ansible-node1).**

**🡺** give the pem file name.

🡺**vim k8s.pem**

🡺we are copy the pem file text and pasting inside the vim k8s.pem file.

**: wq!**

**Step29: After generating key file – if case you want to be giving the permission your key pair command called:**

**🡺 chmod 600 k8s.pem**

**Note: Only main user can access or authenticate the file we are giving permission give command chmod 600 <file\_name>**

**🡺 Adhoc command: - Single line command.**

**Step30:** listing file inside the ksdiv folder.

[ec2-user@ip address ksdev] $ 🡺ls -------------> O/p: **inv k8s.pem**

**Step31:** To changes Inside the ksdev folder inventory and passing the module ti ping command.

🡺 **ansible -i inv DEV –private-key=k8s.pem -u ec2-user -m ansible.builtin.ping**

**Step32:** Using Adhoc command and creating directory.

🡺 **ansible -i inv DEV –private-key=k8s.pem -u ec2-user -m ansible.builtin.file -b -a ‘’dest=/ec2-user/KS state=directory”**

**NOTE: -b command: Become user to giving permissions to file.**

**Step33:** Open the second node (Ansible-Node1) and check the file creation KS dir file or not.

🡺[ec2-user@ip address ~] $ **cd /ec2-user/**

🡺 [ec2-user@ip address ec2-user] $ **ls ----------->O/P: KS**

**Step34:** Back to First Node (Ansible) and created other one director.

**Hint: We are in ansible main node.**

🡺 **ansible -i inv DEV –private-key=k8s.pem -u ec2-user -m ansible.builtin.file -b -a ‘’dest=/ec2-user/DevOps state=directory”**

**Step35:** Check the second directory created in second node (Ansible-Node1).

**Hint: we are in Second Node**

🡺 [ec2-user@ip address ec2-user] $ **ls ----------->O/P: KS DevOps**

**Step36:** Again, creating one more service name: (Ansible-Node 2)

**Step37:** Inset the three node (Ansible-Node2) IP Address inside the inventory file.

**Hint: we are in ansible main node**

[ec2-user@ip address ksdev] $ 🡺 **vim inv**

**🡺** Inside the inv file updating the three node IP Address.

**[DEV]**

**172.31.36.135**

**172.31.41.124**

**: wq!**

**Step38: The** below command inside the two node (ansible-node1, node2) Devops directory folder will create.

🡺 **ansible -i inv DEV –private-key=k8s.pem -u ec2-user -m ansible.builtin.file -b -a ‘’dest=/ec2-user/DevOps state=directory”**

**Step39**: To check the devops dir display or not inside the three Node(ansible-node2)

**Hint: know we are in three node.**

🡺[ec2-user@ip-172-31-41-124 ~] $ **cd /ec2-user/**

🡺 [ec2-user@ip address ec2-user] $ **ls ----------->O/P: Devops**

**Step40: Again calling the directory command to created a KS dir folder.**

**Hint: we are in ansible main node**

🡺 **ansible -i inv DEV –private-key=k8s.pem -u ec2-user -m ansible.builtin.file -b -a ‘’dest=/ec2-user/KS state=directory”**

**Step41:** To check the KS dir display or not inside the three Node(ansible-node2)

**Hint: we are in three node**

🡺[ec2-user@ip-172-31-41-124 ~] $ **cd /ec2-user/**

🡺 [ec2-user@ip address ec2-user] $ **ls ----------->O/P: Devops KS**

**Step42:** If case you want to delete the dir using adhoc command.

🡺 **ansible -i inv DEV –private-key=k8s.pem -u=ec2-user -m ansible.builtin.file -b -a ‘’dest=/ec2-user/KS state=absent”**

**Step43:** To creating web service package install with help of adhoc command.

**Hint: we are in ansible main node**

🡺 **ansible -i inv DEV –private-key=k8s.pem -u ec2-user -m ansible.builtin.file -b -m ansible.builtin.yum -a ‘’name=httpd state=present’’**

**Step44: c**heck the Httpd server working in second and three nodes.

**Hint: we are in three nodes. And check two nodes also.**

🡺[ec2-user@ip-172-31-41-124 ec2-user] $ **systemctl status httpd**

**Step45:** To starting the service using adhoc command.

**Hint: we are in ansible main node**

🡺 **ansible -i inv DEV –private-key=k8s.pem -u=ec2-user -m ansible.builtin.file -b -m ansible.builtin.service -a ‘’name=httpd state=started’’**

**Step46: To check the service stated or not.**

**Hint: we are in three nodes. And check two nodes also.**

🡺[ec2-user@ip-172-31-41-124 ec2-user] $ **systemctl status httpd**

**Step47:** Copy all nodes IP address and check the httpd service are not working r not in browser.

**Hite: Open chrome browser and past the two and three nodes IP address.**

**Step48:** To stop the service using Adhoc command.

**Hint: we are in ansible main node**

🡺 **ansible -i inv DEV –private-key=k8s.pem -u=ec2-user -m ansible.builtin.file -b -m ansible.builtin.service -a ‘’name=httpd state=stopped’’**

**Day: - 39 --> Adhoc command** **customization**

Customization mean: mean’s here we have ansible.cfg file there we can unable the key and user file.

**Step49:** Inside the Ksprod directory folder adding production information.

Change directory ksprod.

🡺**[ec2-user@ip address ksworking] $ cd ksprod/**

**🡺[ec2-user@ip address ksprod] $ ls ---------->O/p:** Nofile are there

**Step50: To create a vim inv file.**

**🡺[ec2-user@ip address ksprod] $ vim inv**

**Inside the inv file giving the node’s information.**

**🡺 [PROD]**

**192.168.29.12**

**192.168.29.13**

**: wq!**

**Note: If any case above IP address does not flow give you’re instance IP address.**

**Step51: To calling or check inv file command.**

**🡺[ec2-user@ip address ksprod] $ ansible -i inv PROD --list-hosts**

**🡺O/P: hosts (2):**

**192.168.29.12**

**192.168.29.13**

**Step52: Inside the ksprod no private key know are r creating that.**

**🡺Vim Ks.pem**

Inside the ks.pem file copying the pem file text. (Hint: Where you are save the pem file in that open any notepad copy that txt and past inside the vim k8s.pem)

**: wq!**

**🡺[ec2-user@ip address ksprod] $ ls -l ---->**

**O/P**: **[-rw-rw-r—l ks ks 37 oct 11 21:12 inv**

**-rw-rw-r—l ks ks 37 oct 11 21:14 ks.pem]**

**Step54**: To giving the permission of ks.pem file.

🡺 **chmod 600 ks.pem**

**🡺[ec2-user@ip address ksprod] $ ls -l ---->**

**O/P**: **[-rw-rw-r—l ks ks 37 oct 11 21:12 inv**

**-rw-------—l ks ks 37 oct 11 21:14 ks.pem]**

**Step55:** To ping the module inside the Inv PROD file.

🡺 **ansible -i inv PROD --private.key= ks.pem -u=ec2-user -m ping**

**Step56:** **customization the ansible.cfg file and copy in the cfg file in locally.**

**🡺[ec2-user@ip address ksprod] $ cp /ect/ansible/ansible.cfg .**

**🡺[ec2-user@ip address ksprod] $ ls ---->O/P: - ansible.cfg inv ks.pem**

**Step57:** Open the ansible.cfg file to customize the file.

**🡺Vim ansible.cfg**

**🡺 Change the inventory file name as – inv**

**🡺 : wq!**

**Step58:** Don’t pass inv file while ping Ahdoc command.

🡺 **ansible PROD --private.key= ks.pem -u=ec2-user -m ping**

**Step59:** Know private key and user customization inside the ansible.cfg file.

**🡺Vim ansible.cfg**

**🡺 search: - :/private\_key\_file**

**🡺 Change the private\_key\_file name as = Ks.pem**

**🡺 search: - :/remote\_user**

**🡺 Change the remote user name as = ec2-user**

**Hint: (Remove the before text and giving your pem file name and user name)**

**🡺 : wq!**

**Step60 :** Again calling Adhoc command and removing the private key name and user name.

**ansible PROD -m ansible.builtin.ping**

**--------------------------**

**Ansible Play book: No of tasks.**

**🡺** **ansible playbook syntax:**

Playbook\_file1.yaml

- name: stmt-1

hosts: **PROD**

tasks:

- name: task1

module 1

- name: task1

module 2

- name: stmt-2

hosts: **DEV**

tasks:

- name: task1

module 1

- name: task1

module 2

**[what is Playbook --> No of Tasks.**

🡺 It’s a collection of modules.

🡺Here we can multiple modules & we can reuse it.

🡺Playbook is written on yaml language.

🡺It works on key- value and its extension (Yaml /Yml)]

**Step61:** Installing VS- Code in your local system.

🡺Inside VSCode creating ansible folder.

🡺 Inside VSCode install ansible plunging

**Step62:** created one demo1.yaml file to writing playbook.

File name: demo1.yaml

---

- name : demo on playbook

hosts : PROD

tasks :

- name : play-book 1st task

ansible.builtin.ping :

**Step63:** copy playbook task in vim demo1 file

**🡺[ec2-user@ip address ksprod] $ vim demo1.yaml**

**🡺 copy text in demo1 file.**

**🡺 : wq!**

**Step64:** Execution the playbook task or calling ping command.

🡺 **ansible-playbook demo1.yaml**

**DAY -40: playbook**

**Step65:** list file inside Ksprod folder.

🡺 **ls** ----------->**O/P: ansible.cfg demo1.yaml inv ks.pem**

**Step66:** Creating one directory name prod-play folder.

🡺 **mkdir prod-play**

**Step67:** moving or changing direction.

**🡺[ec2-user@ip address ksprod] $ cd prod-play/**

**Step68:** copy playbook task in demo1 file and pasting in Vim pb1.yaml.

**🡺[ec2-user@ip address ksprod] $ vim pb1.yaml**

🡺past Inside the pb.1yaml

🡺: **wq!**

**Step69:** To running the Pinging command back to Ksprod folder change dir command.

**🡺[ec2-user@ip address prod-play] $ cd ..**

**Step70:** Execution the playbook task or calling ping command.

**🡺[ec2-user@ip address ksprod] $ ansible-playbook prod-play/pb1.yaml**

**Step71:** In VS code creating another playbook file to write task’s.

VSFile name: pb2.yaml: hint: - debug command

---

- name : **First play**

hosts : **PROD**

tasks :

- name : play1 and task1

ansible.builtin.debug:

msg: “Welcome to kertikeyasoft Devops”

- name : play1 and task2

ansible.builtin.debug:

msg: “Welcome to kertikeyasoft AWS”

- name : **Second play**

hosts : **PROD**

tasks :

- name: play2 and taks1

Ansible.builtin.debug:

msg: “Welcome to kertikeyasoft Kubernetes”

- name: play2 and taks2

Ansible.builtin.debug:

msg: “Welcome to kertikeyasoft AZureDevops”

**Step72:** copy playbook task in VScode and pasting.

**🡺[ec2-user@ip address ksprod] $ vim prod-play/pb2.yaml**

**🡺 copy text in demo1 file.**

**🡺 : wq!**

**Step73: :** Execution the playbook task.

🡺 **ansible- playbook prod-play/pb2.yaml**

**Step74: :** In VS code creating another playbook file to writing variable task’s.

VSFile name: pb3.yaml: hint: - Variable command

- name : **pb on vars**

hosts : **PROD**

vars **:**

NAME : Kertikeyasoft

tasks :

- name : print the vars

ansible.builtin.debug:

msg: “Welcome to {{NAME}}”

**Step75:** copy pb3.yaml playbook task in VScode and pasting.

**🡺[ec2-user@ip address ksprod] $ vim prod-play/pb3.yaml**

**🡺 copy text and past inside the pb3.yaml file.**

**🡺 : wq!**

**Step76:** Execution the playbook task.

🡺 **ansible- playbook prod-play/pb3.yaml**

**Step77:** Vim external\_vars task’s.

external\_vars.yaml: hint: - **Variable in included files command**

**Name : ks**

**TRAINING : DEVOPS**

**BATCH : 87**

**:wq!**

**Step78:** Know external vars text calling in your playbook. In VS code creating another playbook file to writing variable task’s.

VSFile name: pb4.yaml: hint: - external\_vars calling tasks

- name : **pb on vars\_file**

hosts : **PROD**

vars\_files **:**

- external\_vars.yaml

tasks :

- name: print vars from vars\_files

ansible.builting.debug:

msg : “Name is {{Name}}”

- name: print vars from vars\_files

ansible.builting.debug:

msg : “Training is {{TRAINING}}”

- name: print vars from vars\_files

ansible.builting.debug:

msg : “Batch is {{BATCH}}”

**Step79:** copy pb4.yaml playbook task in VScode and pasting in pb4.yaml.

**🡺[ec2-user@ip address ksprod] $ vim prod-play/pb4.yaml**

**🡺 copy text and past inside the pb4.yaml file.**

**🡺 : wq!**

**Step80::** Execution the playbook task.

🡺 **ansible- playbook prod-play/pb4.yaml**

**Step81:** vars\_prompt tasks. In VS code creating another playbook file to writing variable task’s.

VScode – pb5.yaml: hint: - **vars\_prompt command**

**- name : playbook on vars\_prompt**

**hosts : PROD**

**vars\_prompt :**

**- name : “USERNAME”**

**Prompt : “Please enter the service account name”**

**Private : false --------------------------------->hint :** we are able to see username display .

**- name : “PASSWD”**

**Prompt : “Please enter the service account password”**

**tasks :**

**- name : print the service account user name**

**ansible.builtin.debug :**

**msg : “{{USERNAME}}”**

**- name : print the service account user password**

**ansible.builtin.debug :**

**msg : “{{PASSWD}}”**

**Step82:** copy pb5.yaml playbook task in VScode and pasting in Vim pb5.yaml.

**🡺[ec2-user@ip address ksprod] $ vim prod-play/pb5.yaml**

**🡺 copy text and past inside the pb5.yaml file.**

**🡺 : wq!**

**Step83:** Execution the playbook task.

🡺 **ansible- playbook prod-play/pb5.yaml**

**🡺Please enter the service account Username: Kartikeya**

**🡺Please enter the service account User passwd:**

**Step84: In Inventory level calling variable file**

**🡺 vim inv**

**[PROD]**

**192.168.29.12**

**192.168.29.13**

**[PROD: Vars]**

**BATCH = 87**

**COURSE = DEVOPS**

**Step85:** In VS code creating another playbook file to writing In Inventory level calling variable file.

VSFile name: pb6.yaml: 🡪 Here we didn’t calling variable it placed in inventory level.

- **name : pb on Inventory level calling variable file**

**hosts : PROD**

**tasks :**

**- name: print vars from vars\_files**

**ansible.builting.debug:**

**msg : “Batch is {{BATCH }}”**

**- name: print vars from vars\_files**

**ansible.builting.debug:**

**msg : “Course is {{COURSE}}”**

**Step86:** copy pb6.yaml playbook task in VScode and pasting in pb6.yaml.

**🡺[ec2-user@ip address ksprod] $ vim prod-play/pb6.yaml**

**🡺 copy text and past inside the pb6.yaml file.**

**🡺 : wq!**

**Step87:** Execution the playbook task.

🡺 **ansible- playbook prod-play/pb6.yaml**

**Day-41:**

**Step88: 🡺[ec2-user@ip address ~] $ ls -------------> O/P: KSworking**

**Step89:** change directory

**🡺[ec2-user@ip address ~] $ cd ksworking/**

**Step90:** List the file present inside the ksworking dir folder.

**🡺[ec2-user@ip address ksworking] $ ls --------->O/P: ksdev ksprod kstest**

**Step91:** changing the dir for ksworking to ksprod.

**🡺[ec2-user@ip address ksworking] $ cd ksprod/**

**Step92:** List the file present inside the ksprod dir folder.

**🡺[ec2-user@ip address ksprod] $ ls ---------->O/P: ansible.cfg demo1.yaml inv ks.pem prod-play**

**Step93:** Installing httpd server in Linux. In VS code creating another playbook file to writing task’s install web services .

VScode – pb7.yaml: hint: - Installing httpd server.

**- name : PB on web layer**

**hosts : PROD**

**become: yes -------> [**hint: become command add because you need to be root to perform this task]

**tasks :**

**- name : installing web server**

**ansible.bulitin.yum :**

**name : httpd**

**state : latest / absent (**Hint: in you system httpd server already there mean you can uninstall server : absent states.**)**

**- name : starting web server**

**ansible.builtin.service :**

**name : httpd**

**state : started**

**Step94:** copy pb7.yaml playbook task in VScode and pasting in vim web.yaml.

**🡺[ec2-user@ip address ksprod] $ vim prod-play/web.yaml**

**🡺 copy text and past inside the web.yaml file.**

**🡺 : wq!**

**Step95:** Execution the playbook task.

🡺 **ansible- playbook prod-play/web.yaml**

**3-Tier Architecture: Web, App, DB**

**Web Layer:**

**🡺 Installing httpd package.**

**🡺 Service Start.**

**🡺 Download modkjk connector.**

**🡺 Extracting modjk**

**🡺 gcc, httpd -devel**

**🡺 Compile.**

**. /configure --with-apxs=/bin/apxs**

**Mod\_jk.so – Module [ we are using connecting web to app through mod jk]**

**Step96:** Download the mod\_jk

🡺 <https://tomcat.apache.org/download-connectors.cgi>

Copying: <https://dlcdn.apache.org/tomcat/tomcat-connectors/jk/tomcat-connectors-1.2.49-src.tar.gz>

**Step98:** Installing Mod\_jk server in Linux. In VS code creating another playbook file to writing task’s install mod-jk services.

VScode – pb8.yaml: hint: - Installing mod\_jk connector.

**- name : PB on web layer**

**hosts : PROD**

**become: yes**

**tasks :**

**- name : installing web server**

**ansible.bulitin.yum :**

**name : httpd**

**state : latest**

**- name : starting web server**

**ansible.builtin.service :**

**name : httpd**

**state : started**

**- name : Creating mod\_jk working directory**

**ansible.builtin.file:**

**path : /KSwork**

**mode : ‘0766’**

**state : directory**

**- name : Downloading mod\_jk connector**

**ansible.builtin.get\_url :**

**url :** <https://dlcdn.apache.org/tomcat/tomcat-connectors/jk/tomcat-connectors-1.2.49-src.tar.gz>

**dent: /KSwork**

**Step94:** copy pb8.yaml playbook task in VScode and pasting in vim web.yaml.

**🡺[ec2-user@ip address ksprod] $ vim prod-play/web.yaml**

**🡺 copy text and past inside the web.yaml file.**

**🡺 : wq!**

**Step95:** Execution the playbook task.

🡺 **ansible-playbook prod-play/web.yaml**

**Step97:** To check the mod\_jk server install or not in your ansible node’s (node1 and node2)

**Hint: we are in two nodes checking the mod\_jk server install are not. After that check again three Node also.**

🡺[ec2-user@ip-172-31-41-124 ~] $ **cd /**

🡺[ec2-user@ip-172-31-41-124 / ] $ **ls ------->** [bin, boot, dev, ect, home, ksapp, KSwork, lib, lib64, media, mnt, opt, proc, root, run, sbin, sys ect..]

🡺[ec2-user@ip-172-31-41-124 /] $ **cd KSWork/**

🡺[ec2-user@ip-172-31-41-124 KSWork] $ **ls ----->** [tomcat-connectors-1.2.49-src.tar.gz]

**Step98:** Installing Mod\_jk server in Linux. In VS code creating another playbook file to writing task’s install mod-jk services.

VScode – pb8.yaml: hint: - Extracting Mod\_jk.

**- name : PB on web layer**

**hosts : PROD**

**become: yes**

**tasks :**

**- name : installing web server**

**ansible.bulitin.yum :**

**name : httpd**

**state : latest**

**- name : starting web server**

**ansible.builtin.service :**

**name : httpd**

**state : started**

**- name : Creating mod\_jk working directory**

**ansible.builtin.file:**

**path : /KSwork**

**mode : ‘0766’**

**state : directory**

**- name : Downloading mod\_jk connector**

**ansible.builtin.get\_url :**

**url :** <https://dlcdn.apache.org/tomcat/tomcat-connectors/jk/tomcat-connectors-1.2.49-src.tar.gz>

**dent: /KSwork**

**- name : Extracting mod\_jk**

**ansible.builtin.unarchive :**

**src : /KSWork/** **tomcat-connectors-1.2.49-src.tar.gz**

**dent: /KSwork**

**remote\_src : Yes**

**Step94:** copy pb8.yaml playbook task in VScode and pasting in vim web.yaml.

**🡺[ec2-user@ip address ksprod] $ vim prod-play/web.yaml**

**🡺 copy text and past inside the web.yaml file.**

**🡺 : wq!**

**Step95:** Execution the playbook task.

🡺 **ansible-playbook prod-play/web.yaml**

**Step97:** To check the mod\_jk server **Extracting** or not in your ansible node’s (node1 and node2)

**Hint: we are in two nodes checking the mod\_jk server Extracting are not. After that check again three Node also.**

Before command🡺[ec2-user@ip-172-31-41-124 KSWork] $ **ls ----->** [tomcat-connectors-1.2.49-src.tar.gz]

🡺[ec2-user@ip-172-31-41-124 KSWork] $ **ls -l**

**-----> [**drwxr-xr-x 9 root bin **tomcat-connectors-1.2.49-src**

-rw-r—r--1 1 root root tomcat-connectors-1.2.49-src.tar.gz

🡺[ec2-user@ip-172-31-41-124 KSWork] $ **cd tomcat-connectors-1.2.49-src**

🡺[ec2-user@ip-172-31-41-124 tomcat-connectors-1.2.49-src] # **ls** --------> **O/P:** [ conf, HOWTO-RELEASE.txt, LICENSE, NOTICE, support, xdocs, docs, jkstatus, native, README.txt, tools]

**[/configure --with-apxs=/bin/apxs ]**

🡺[ec2-user@ip-172-31-41-124 **tomcat-connectors-1.2.49-src** ] # **cd native/**

🡺[ec2-user@ip-172-31-41-**124 native] #** **ls- l**

**--->O/P:** [aclocal.m4, apache-2.0, buildconf.sh, BUILDING.txt, common, configure, configure.ac, docs, iis, Markefile.am, Makefile.in, README.txt, scripts, TODO.txt]

**Install apxs,C -compiler and some more.**

**Step98: Dependences Pack installation** server in Linux. In VS code creating another playbook file to writing task’s install mod-jk services.

VScode – pb8.yaml: hint: - Dependences pack Installation – apxs ect...

**- name : PB on web layer**

**hosts : PROD**

**become: yes**

**tasks :**

**- name : installing web server**

**ansible.bulitin.yum :**

**name : {{item}}**

**state : latest**

**loop:**

**- httpd**

**- gcc**

**- httpd-devel**

**- name : starting web server**

**ansible.builtin.service :**

**name : httpd**

**state : started**

**- name : Creating mod\_jk working directory**

**ansible.builtin.file:**

**path : /KSwork**

**mode : ‘0766’**

**state : directory**

**- name : Downloading mod\_jk connector**

**ansible.builtin.get\_url :**

**url :** <https://dlcdn.apache.org/tomcat/tomcat-connectors/jk/tomcat-connectors-1.2.49-src.tar.gz>

**dent: /KSwork**

**- name : Extracting mod\_jk**

**ansible.builtin.unarchive :**

**src : /KSWork/** **tomcat-connectors-1.2.49-src.tar.gz**

**dent: /KSwork**

**remote\_src : Yes**

**Step94:** copy pb8.yaml playbook task in VScode and pasting in vim web.yaml.

**🡺[ec2-user@ip address ksprod] $ vim prod-play/web.yaml**

**🡺 copy text and past inside the web.yaml file.**

**🡺 : wq!**

**Step95:** Execution the playbook task.

🡺 **ansible-playbook prod-play/web.yaml**

**Day- 42:**

**Tips:**

**🡺 cd mod\_jk/native**

**🡺 . /configure –with-apxs=/bin/apxs**

**🡺make, make install.**

**Step98:** **Directory enter** server in Linux. In VS code creating another playbook file to writing task’s install mod-jk services.

VScode – pb8.yaml: hint: - Directory enter – using shell\_module...

**- name : PB on web layer**

**hosts : PROD**

**become: yes**

**tasks :**

**- name : installing web server**

**ansible.bulitin.yum :**

**name : {{item}}**

**state : latest**

**loop:**

**- httpd**

**- gcc**

**- httpd-devel**

**- make**

**- name : starting web server**

**ansible.builtin.service :**

**name : httpd**

**state : started**

**- name : Creating mod\_jk working directory**

**ansible.builtin.file:**

**path : /KSwork**

**mode : ‘0766’**

**state : directory**

**- name : Downloading mod\_jk connector**

**ansible.builtin.get\_url :**

**url :** <https://dlcdn.apache.org/tomcat/tomcat-connectors/jk/tomcat-connectors-1.2.49-src.tar.gz>

**dent: /KSwork**

**- name : Extracting mod\_jk**

**ansible.builtin.unarchive :**

**src : /KSWork/** **tomcat-connectors-1.2.49-src.tar.gz**

**dent: /KSwork**

**remote\_src : Yes**

**- name : compile mod\_jk package**

**shell : . /configure –with-apxs=/bin/apxs**

**args :**

**step:99 following chdir : /opt/tomcat-connectors-1.2.49- src/native**

**- name : execute make cmd**

**shell : make**

**args :**

**chdir : /opt/tomcat-connectors-1.2.49- src/native**

**- name : execute make install cmd**

**shell : make install**

**args :**

**chdir : /opt/tomcat-connectors-1.2.49- src/native**

**[mod\_jk url find]**

**Step99:** 🡺[ec2-user@ip-172-31-41-124 **tomcat-connectors-1.2.49-src** ] # **cd native/**

🡺[ec2-user@ip-172-31-41-124 **native**] # **ls**

**--->O/P:** [aclocal.m4, apache-2.0, buildconf.sh, BUILDING.txt, common, configure, configure.ac, docs, iis, Markefile.am, Makefile.in, README.txt, scripts, TODO.txt]

🡺[ec2-user@ip-172-31-41-124 **native**] # **pwd**

**---->O/P: /opt/tomcat-connectors-1.2.49- src/native**

**Step94:** copy pb8.yaml playbook task in VScode and pasting in vim web.yaml.

**🡺[ec2-user@ip address ksprod] $ vim prod-play/web.yaml**

**🡺 copy text and past inside the web.yaml file.**

**🡺 : wq!**

**Step95:** Execution the playbook task.

🡺 **ansible-playbook prod-play/web.yaml**

**Step98: Directory enter** server in Linux. In VS code creating another playbook file to writing task’s install mod-jk services.

VScode – pb8.yaml: hint: - Directory enter – using shell\_module...

**- name : PB on web layer**

**hosts : PROD**

**become: yes**

**tasks :**

**- name : installing web server**

**ansible.bulitin.yum :**

**name : {{item}}**

**state : latest**

**loop:**

**- httpd**

**- gcc**

**- httpd-devel**

**- make**

**- name : starting web server**

**ansible.builtin.service :**

**name : httpd**

**state : started**

**- name : Creating mod\_jk working directory**

**ansible.builtin.file:**

**path : /KSwork**

**mode : ‘0766’**

**state : directory**

**- name : Downloading mod\_jk connector**

**ansible.builtin.get\_url :**

**url :** <https://dlcdn.apache.org/tomcat/tomcat-connectors/jk/tomcat-connectors-1.2.49-src.tar.gz>

**dent: /KSwork**

**- name : Extracting mod\_jk**

**ansible.builtin.unarchive :**

**src : /KSWork/** **tomcat-connectors-1.2.49-src.tar.gz**

**dent: /KSwork**

**remote\_src : Yes**

**- name : compile mod\_jk package**

**shell : “{item}”**

**args :**

**chdir : /opt/tomcat-connectors-1.2.49- src/native**

**loop :**

**- . /configure –with-apxs=/bin/apxs**

**- make**

**- make install**

**Step94:** copy pb8.yaml playbook task in VScode and pasting in vim web.yaml.

**🡺[ec2-user@ip address ksprod] $ vim prod-play/web.yaml**

**🡺 copy text and past inside the web.yaml file.**

**🡺 : wq!**

**Step95:** Execution the playbook task.

🡺 **ansible-playbook prod-play/web.yaml**

**Step96:** Automate install two servicer and uninstall few with help of **Loop- Iterating Over a list of hashes.**

**Step98: Two pack’s install and Two pack’s uninstall** server in Linux. In VS code creating another playbook file to writing task’s

VScode – pb8.yaml: hint: - **Loop- Iterating Over a list of hashes**

**- name : PB on web layer**

**hosts : PROD**

**become: yes**

**tasks :**

**- name : installing web server**

**ansible.bulitin.yum :**

**name : “{{item.name}}”**

**state : “{{item.state}}”**

**loop:**

**- {name : ‘httpd’ , state : ‘latest’ }**

**- {name : ‘httpd-devel’ , state : ‘latest’ }**

**- {name : ‘gcc’ , state : ‘latest’ }**

**- {name : ‘make’ , state : ‘latest’ }**

**- name : starting web server**

**ansible.builtin.service :**

**name : httpd**

**state : started**

**- name : Creating mod\_jk working directory**

**ansible.builtin.file:**

**path : /KSwork**

**mode : ‘0766’**

**state : directory**

**- name : Downloading mod\_jk connector**

**ansible.builtin.get\_url :**

**url :** <https://dlcdn.apache.org/tomcat/tomcat-connectors/jk/tomcat-connectors-1.2.49-src.tar.gz>

**dent: /KSwork**

**- name : Extracting mod\_jk**

**ansible.builtin.unarchive :**

**src : /KSWork/** **tomcat-connectors-1.2.49-src.tar.gz**

**dent: /KSwork**

**remote\_src : Yes**

**- name : compile mod\_jk package**

**shell : “{item}”**

**args :**

**chdir : /opt/tomcat-connectors-1.2.49- src/native**

**loop :**

**- . /configure –with-apxs=/bin/apxs**

**- make**

**- make install**

**Step94:** copy pb8.yaml playbook task in VScode and pasting in vim web.yaml.

**🡺[ec2-user@ip address ksprod] $ vim prod-play/web.yaml**

**🡺 copy text and past inside the web.yaml file.**

**🡺 : wq!**

**Step95:** Execution the playbook task.

🡺 **ansible-playbook prod-play/web.yaml**

**Lineinfile - Ansible lineinfile module is used to insert a line, modify, remove, and replace a line or multiple lines in the host's configuration files..**

**Ex: prod-play/pb9.yaml**

**---**

**- name : PB on Line in file**

**hosts : PROD**

**become : Yes**

**tasks :**

**- name : Ensure SELinux is set to enforcing mode**

**ansible.builting.lineinfile :**

**path : /ect/selinux/config**

**regexp : ‘^SELINUX='**

**line : SELINUX=disabled**

**blockinfile module – Insert/update/remove a text block surrounded by marker lines.**

**🡺Block of content update**

**Day – 43:**

**Ex: blockinfile.**

**---**

**- name : Demo on block in file**

**hosts : PROD**

**become : yes**

**tasks :**

**- name : Insert/Update “Match user” configuration block in /ect/ss**

**ansible.builtin.blockinfile:**

**path : /ect/ssh/sshd\_config**

**block :**

**Match User ansible-agent**

**PasswordAuthentication yes**

**Step94:** copy pb10.yaml playbook task in VScode and pasting in vim blockinfile.yaml.

**🡺[ec2-user@ip address ksprod] $ vim prod-play/****blockinfile.yaml**

**🡺 copy text and past inside the blockinfile.yaml file.**

**🡺 : wq!**

**Step95:** Execution the playbook task.

🡺 **ansible-playbook prod-play/** **blockinfile.yaml**

**Checking ansible Node1 and Node 2**

**🡺[ec2-user@ip address ~] vim /ect/ssh/sshd\_config**

**Seach: - :/Pass – passed that below text update.**

**🡺 ansible-agent PasswordAuthentication no**

**Step95:**

**vim modjk.conf --- File**

LoadModule jk\_module modules/mod\_jk.so

JkWorkersFile conf.d/workers.properties

JkLogFile logs/mod\_jk.log

JkLogLevel info

JkLogStampFormat "[%a %b %d %H:%M:%S %Y]"

JkOptions +ForwardKeySize +ForwardURICompat -ForwardDirectories

JkRequestLogFormat "%w %V %T"

JkMount /LoginApp tomcatA

JkMount /LoginApp/\* tomcatA

vim workers.properties

worker.tomcatA.type=ajp13

worker.tomcatA.host=10.128.0.5

worker.tomcatA.port=8009

**Step95:Checking ansible Node1 and Node 2**

**🡺[ec2-user@ip address ~] #cd /opt/**

**🡺[ec2-user@ip address opt] # ls ----->O/P: tomcat-connectors-1.2.49-src**

**🡺[ec2-user@ip address opt] # cd tomcat-connectors-1.2.49-src/**

**🡺[ec2-user@ip address tomcat-connectors-1.2.49-src/] # ls**

**----->O/P: [**conf, HOWTo-RELEASE.txt, LICENSE, NOTICE, support, xdocs

Docs jkstatus native README.txt tools**]**

**🡺[ec2-user@ip address tomcat-connectors-1.2.49-src/] #**

**----> cd /ect/httpd/conf.d**

**🡺[ec2-user@ip address conf.d] # ls ----->O/P**: [autoindex.conf, README userdir.conf, welcome.conf]

**🡺[ec2-user@ip address conf.d] # pwd --->O/P: /ect/httpd/conf.d**

**Step95:** ansible main node task

**🡺[ec2-user@ip address ksprod] $ ls ------> O/P: ansible.cfg demo1.yaml inv ks.pem prod-play**

**🡺[ec2-user@ip address ksprod] $ cd prod-play/**

**🡺[ec2-user@ip address prod-play] $ ls --->O/P: [blockinfile.yaml, lineinfile.yaml, stack.yaml]**

**🡺[ec2-user@ip address prod-play] $ cd ..**

**🡺[ec2-user@ip address ksprod] $**

**Step98: Two pack’s install and Two pack’s uninstall** server in Linux. In VS code creating another playbook file to writing task’s

VScode – pb11.yaml: hint: - **SDDM configuration file in showing Node’s**

**- name : PB on web layer**

**hosts : PROD**

**become: yes**

**tasks :**

**- name : installing web server**

**ansible.bulitin.yum :**

**name : “{{item.name}}”**

**state : “{{item.state}}”**

**loop:**

**- {name : ‘httpd’ , state : ‘latest’ }**

**- {name : ‘httpd-devel’ , state : ‘latest’ }**

**- {name : ‘gcc’ , state : ‘latest’ }**

**- {name : ‘make’ , state : ‘latest’ }**

**- name : starting web server**

**ansible.builtin.service :**

**name : httpd**

**state : started**

**- name : Creating mod\_jk working directory**

**ansible.builtin.file:**

**path : /KSwork**

**mode : ‘0766’**

**state : directory**

**- name : Downloading mod\_jk connector**

**ansible.builtin.get\_url :**

**url :** <https://dlcdn.apache.org/tomcat/tomcat-connectors/jk/tomcat-connectors-1.2.49-src.tar.gz>

**dent: /KSwork**

**- name : Extracting mod\_jk**

**ansible.builtin.unarchive :**

**src : /KSWork/** **tomcat-connectors-1.2.49-src.tar.gz**

**dent: /KSwork**

**remote\_src : Yes**

**- name : compile mod\_jk package**

**shell : “{item}”**

**args :**

**chdir : /opt/tomcat-connectors-1.2.49- src/native**

**loop :**

**- . /configure –with-apxs=/bin/apxs**

**- make**

**- make install**

**- name: create SDDM configuration**

**copy:**

**dest : /ect/httpd/conf.d/**

**group : root**

**mode : 0644**

**owner : root**

**src : modjk.conf**

**remoteuser :yes**

**Step94:** copy pb11.yaml playbook task in VScode and pasting in vim SDDM configuration.yaml.

**🡺[ec2-user@ip address ksprod] $ vim prod-play/** **SDDM configuration.yaml**

**🡺 copy text and past inside the SDDM configuration.yaml file.**

**🡺 : wq!**

**Step95:** Execution the playbook task.

🡺 **ansible-playbook prod-play/** **SDDM configuration.yaml**

**If case you are getting any error 🡺[ec2-user@ip address ksprod] $ mv modjk.conf prod-play /**

**Again running abovestep**

**Day-43; Stopping Videos – 40:45**

**Step95**

**Step95**

**Step95**