**Ansible Installation:-**

1.

yum install epel-release

yum install ansible

ansible --version

2.

Ansible Global configuration files:-

# cat /etc/ansible/ansible.cfg

3.

To varify the default inventory of servers

# vi /etc/ansible/hosts

4.

**# Create a ansible user and update sudoers file for no sudo passowrd in master and clients**

**Adduser in ubuntu OS:-**

useradd -m -d /home/ansible -s /bin/bash -c "Ansible Owner" -U ansible; (echo redhat; echo redhat) | passwd ansible

5.

**Adduser in Redhat/Cent OS:-**

useradd ansible; (echo redhat; echo redhat) | passwd ansible

echo 'ansible ALL=(ALL) NOPASSWD: ALL' | sudo EDITOR='tee -a' visudo

6.

# Generate the ssh-keygen in Master server

#ssh-keygen

7.

Copy the public keys from master to clients:-

ssh-copy-id ansible@<host name or IP>

8.

# Validate the client response

ansible -m ping all

9.

We can make groups of inventory eg: apps, DB, cluster on [].

vi inventory or vi /etc/ansible/hosts

[Appsserver]

lb01

[webserver]

web1

[dbserver]

db01

[control]

control ansible\_connect=local # for connect with local with out ssh

ansible -i <file name to pass the server list for above eg: dev> --list-hosts all

List all ansible clients .

#ansible --list-hosts all

10.

#Gathering facts

ansible all -m setup

To display the all hosts from the inventory

#ansible -i inventory-file --list-hosts all

**To display the all hosts from the groups**

#ansible --list-hosts <Groups name of servers>

#ansible --list-hosts dbserver

#ansible --list-hosts "app0\*"

#ansible --list-hosts dbserver:webserver

#ansible --list-hosts <Groups1>:<Groups2>

#ansible --list-hosts \!control # Except control groups

----------------------------------------

**#Ansible Ad-hoc commands:-**

ansible -m ping all

ansible -m command -a "hostname" all

ansible -a "hostname" all

**#Check the uptime of Ansible nodes group based**

ansible apps -i inventory -m command -a "uptime"

ansible -m command -a "uname -a" 'apps'

ansible -m command -a "df -Th" 'all' > /tmp/command-output.txt

# Upgrade all the Ubuntu servers.

ansible debian -m apt -a "upgrade=yes update\_cache=yes" -b

# Upgrade all the CentOS servers.

ansible centos -m yum -a "name=\* state=latest" -b

#Connecting as a Different User

ansible-playbook myplaybook.yml -u sammy

**#Providing the sudo Password**

ansible-playbook -i inventory myplaybook.yml --ask-become-pass

#### How to check syntax error in playbook ?

ansible-playbook --syntax-check myansari.yml

# RUN a Ad-hok command with root user

ansible all -m command -a "lvdisplay" --become

ansible all -m service -a “name=httpd state=restarted”

ansible all -m service -a “name=httpd state=reloaded”

ansible all -m service -a “name=httpd state=reloaded enabled=yes”

#ansible all -i inventory -u ansible -m ping

**## Passing module setup**

ansible all -i inventory -u ansible -m setup

ansible webservers -i hosts -u ansible -m yum -a "name=httpd state=present" -b

ansible webservers -i hosts -u ansible -m yum -a "name=httpd state=absent" -b

#### Syntax-check, & Dry Run

ansible-playbook --syntax-check status.yml

ansible-playbook --check status.yml

**###List the tasks going execute, Start-at-task:-It will ask each task steps y/n ? we can skip or proceed**

ansible-playbook status.yml --step

ansible-playbook status.yml --list-tasks

##We can view the list of tasks( Have to check)

ansible-playbook status.yml --start-at-task "copy demo app source"

##This will jump to a spacific task and execute.

We can check the yml file before go for the real time executions.

**Retry Failed hosts:-**

If any host failed during the execution we can run the task for only for that hosts.

# ansible-playbook status.yml --limit @/home/ansible/status.retry

Debug:- Run the yml task on specific host and specific task

# ansible-playbook status.yml --limit node1 --start-at-task "copy demo app source"

-----------------------------------------------------------------

Create a Playbook for plays : -

yaml file syntax for action.

- hosts: all

tasks:

- name: get server hostname

command: hostname

How to run playbook ?

#ansible-playbook hostname.yml

----------------------------------------------------------------

**## Install a list packages in centos with items method.**

---

- hosts: testserver

tasks:

- name: Install the required rpms

yum: name={{ item.name }} state=latest

with\_items:

- { name: wget }

- { name: git }

- { name: net-tools }

- { name: tree }

- { name: httpd }

**#Install apache webserver and start/restart on centos:**

---

- hosts: all

become: yes

tasks:

- name: install httpd\*

yum:

name: httpd

state: latest

tasks:

- name: Start web service

service:

name: httpd

state: started

enabled: yes

tasks:

- name: Restart web reservice

service:

name:httpd

state:restarted

enabled:yes

-----------------------------------------------

**Vars means variables**

-name: install and start apache

hosts: web

vars:

http\_port: 80

max\_client: 200

remote\_user: root

task:

- name: install httpd

yum: pkg= httpd state: latest

- name: write the apache config file

template: src=/srv/https.j2 dest=/etc/https.conf

- name: start httpd

service: name=httpd state=started

----------------------------------------

## Upgrade all OS packages

---

- hosts: all

- name: upgrade all OS packages

yum:

name: '\*'

state: latest

# Upgrade the all softwares excluded a spacific packages eg: kernel

---

- hosts: all

- name: upgrade all packages, excluding kernel & foo related packages

yum:

name: '\*'

state: latest

exclude: kernel\*,foo\*

--------------------------

#**# install/Uninstall the software from centos**

---

- hosts: all

become: yes

tasks:

- name: install vsftpd

yum: name=vsftpd state=present

- name: start service

service: name=vsftpd state=started enabled=yes

- name: stop service

service: name=httpd state=stopped enabled=no

- name: uninstall httpd

yum: name=httpd state=absent

---------------------------

# **Upgrade the all softwares excluded a spacific packages eg: kernel**

---

- hosts: all

- name: upgrade all packages, excluding kernel & foo related packages

yum:

name: '\*'

state: latest

exclude: kernel\*,foo\*

----------------------------------------

**# Resrat the stack with ansible:**

Add all ansible steps for below task in playbook

1. stop nginx service

2. stop web server

3. restart the DB

4. start nginx service

5. start web server

u

# Tested and working successfully with file, mode, copy, path and handlers

---

- hosts: apps

become: yes

tasks:

- name: install apache webserver

yum:

name: httpd

state: latest

- name: copying a httpd.conf file from local folder (Master) to remote server

copy:

src: /home/ansible/myplay/httpd.conf

dest: /home/ansible

mode: 0755

- name: copying a index.html from local folder (Master) to remote server

copy:

src: /home/ansible/myplay/index.html

dest: /var/www/html

mode: 0644

- name: De-Activate default apache site

file:

path: /etc/httpd/conf/httpd.conf

state: absent

notify: restart httpd

- name: Activate default apache site

file:

src: /home/ansible/httpd.conf

dest: /etc/httpd/conf/httpd.conf

state: link

notify: restart httpd

handlers:

- name: restart httpd

service:

name: httpd

state: restarted

enabled: yes

----------------------------------------

**Templet:-**

- hosts: testserver

tasks:

- name: install nginx

yum: name=nginx state=present

- name: Ensure nginx started

service: name=nginx state=started enabled=yes

- name: configure nginx site

template: src=templates/nginx.conf.j2 dest=/etc/nginx/conf.d/demo mode=0644

notify: restart nginx

- name: de-activate default site

file: path=/usr/share/nginx/html/index.html state=absent

notify: restart nginx

- name: activate demo site

#copy: src=/etc/nginx/conf.d/demo dest=/usr/share/nginx/html/index.html remote\_src=yes

file: src=/etc/nginx/conf.d/demo dest=/usr/share/nginx/html/demo owner=root group=root state=link

notify: restart nginx

handlers:

- name: restart nginx

service: name=nginx state=restarted

-------------------------------------------------

- hosts: testserver

tasks:

- name: install nginx

yum: name=nginx state=present

- name: Ensure nginx started

service: name=nginx state=started enabled=yes

- name: configure nginx site

template: src=templates/nginx.conf.j2 dest=/etc/nginx/conf.d/demo mode=0644

notify: restart nginx

- name: de-activate default site

file: path=/usr/share/nginx/html/index.html state=absent

notify: restart nginx

- name: activate demo site

copy: src=/etc/nginx/conf.d/demo dest=/root remote\_src=yes

notify: restart nginx

handlers:

- name: restart nginx

service: name=nginx state=restarted

-----------------------------------

hosts: blocks

tasks:

- name: Ansible copy directory to the remote server

copy:

src:/Users/mdtutorials2/Documents/Ansible/copy\_dir\_ex

dest:/Users/mdtutorials2/Documents/Ansible/tmp

------------------------------------------

- hosts: blocks

tasks:

- name: Ansible copy files remote to remote

copy:

src: /tmp/hello6

dest: /etc

remote\_src: yes

---------------------------------------

copy a directory content to another directory

name: copy html file

copy: src=/home/vagrant/dist/ dest=/usr/share/nginx/html/ remote\_src=yes directory\_mode=yes.

----------------------------------------------

linefile:-

- hosts: testserver

tasks:

- name: Enable the selinux on Servers

lineinfile: dest=/etc/selinux/config regexp=^SELINUX= line=SELINUX=enforcing

- hosts: loc

tasks:

- name: Ansible insert lineinfile example

lineinfile:

dest: /home/mdtutorials2/remote\_server.txt

line: Inserting a line in a file.

state: present

create: yes

**We give the regular expression using lineinfile regexp parameter. The following example will remove all lines starting with DevOps.**

- hosts: loc

tasks:

- name: Ansible lineinfile regexp example

lineinfile:

dest: /home/mdtutorials2/remote\_server.txt

regexp: '^DevOps'

state: absent

------------------------------------

Create a mysql Users:

-name: create demo users

mysql\_users : name demo password=demo state=present

-------------------------------------

Wait\_for :- waiting for to come up serveice or something.

- hosts: testserver

tasks:

- name: verify nginx services

command: service nginx status

- name: verify nginx port is listening

wait\_for: port=80 timeout=300

- name: resrted db

service: name=mysql state=restarted

wait\_for: port=3306 state=started

------------------------------------

**uri, register, fail, when:**

- hosts: testserver

tasks:

- name: Download EPEL Repo - Centos/RHEL 7

get\_url: url=http://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm dest=/tmp/epel-release-latest-7.noarch.rpm

when: "ansible\_os\_family == 'RedHat' and ansible\_distribution\_major\_version == '7'"

- name: Install EPEL Repo - Centos/RHEL 7

command: rpm -ivh /tmp/epel-release-latest-7.noarch.rpm creates=/etc/yum.repos.d/epel.repo

when: "ansible\_os\_family == 'RedHat' and ansible\_distribution\_major\_version == '7'"

- name: Install libselinux-python

yum: name=libselinux-python

-----------------------------

- name: Check that you can connect (GET) to a page and it returns a status 200

uri:

url: http://www.example.com

# Check that a page returns a status 200 and fail if the word AWESOME is not

# in the page contents.

- uri:

url: http://www.example.com

return\_content: yes

register: webpage

- name: Fail if AWESOME is not in the page content

fail:

when: "'AWESOME' not in webpage.content"

---------------------------------

tasks:

- wait\_for:

host: ams-server-101

port: 443

timeout: 1

register: https\_port\_check

ignore\_errors: true

- fail:

msg: 'HTTPS port is open'

when: not https\_port\_check.failed

-------------------------------------

**# Example playbook using fail and when together**

- fail:

msg: "The system may not be provisioned according to the CMDB status."

when: cmdb\_status != "to-be-staged"

------------------------------------

Q:.How to use shell module:

“Shell module is used to run any Linux shell based command”

“ ansible node1 -m shell -a ; ls -ld /tmp;ls -ld /etc”

Ansible reboot the Server and wait for comeup:

- hosts: testserver

tasks:

- name: restart system to reboot to newest kernel

shell: "sleep 5 && reboot"

async: 1

poll: 0

- name: wait for 10 seconds

pause:

seconds: 10

- name: wait for the system to reboot

wait\_for\_connection:

connect\_timeout: 20

sleep: 5

delay: 5

timeout: 60

---------------------------------

- yum: name=$item state=latest

with\_items:

- ntp

when: ansible\_distribution == 'CentOS' or ansible\_distribution == 'Red Hat Enterprise Linux'

- service: name=ntpd state=started enabled=yes

-----------------------------

**Include Multiple yml files in a single yml file:-**

- include: control.yml

- include: database.yml

- include: loadbalancer.yml

------------------------------------

variables, variable vars

**16 variables precedence available in ansible**

file: A directory should exist

yum: A package should be installed

service: A service should be running

templete: Render a config file froma template

get\_url: Fetch archive file from URL.

git: Clone a source code respository

-------------------------

with\_dict

--------------------------

Removal: shell, register, with\_items, when

---------------------------

vars\_files, group\_vars

----------------------------

**valt :- Vault is a mechanism that allows encrypted content to be incorporated transparently into Ansible workflows.**

#Creating New Encrypted Files & Varify:

ansible-vault create ansari.yaml

cat ansari.yaml

#Encrypting Existing Files

echo 'unencrypted stuff' > unencriptfile.yaml

ansible-vault encrypt unencriptfile.yaml

cat unencriptfile.yaml

#Viewing Encrypted Files

ansible-vault view vault.yml

#Editing Encrypted Files

ansible-vault edit vault.yml

ansible-vault decrypt vault.yml

cat vault.yml

#Changing the Password of Encrypted Files

ansible-vault rekey vault.yml

**#Running Ansible with Vault-Encrypted Files**

ansible-vault create secret\_key

Ansari confidential data

# Run encripted playbook with vault password.

ansible-playbook -i ../inventory variable.yaml --ask-vault-pass

ansible --ask-vault-pass -bK -m copy -a 'src=secret\_key dest=/tmp/secret\_key mode=0600 owner=root group=root' apps

**#Using Ansible Vault with a Password File**

echo 'redhat' > .vault\_pass

ansible --vault-password-file=.vault\_pass -bK -m copy -a 'src=secret\_key dest=/tmp/secret\_key mode=0600 owner=root group=root' dbserver

#Reading the Password File Automatically

export ANSIBLE\_VAULT\_PASSWORD\_FILE=./.vault\_pass

ansible -bK -m copy -a 'src=secret\_key dest=/tmp/secret\_key mode=0600 owner=root group=root' ubuntu

**#Manually Decrypting Encrypted Files**

Then update password below format

vault\_db\_pass: redhat

then save and exit.

Update to yml file

db\_name: demo

db\_user: demo

db\_pass: "{{ vault\_db\_pass }}"

### for modify password:-

ansible-vault edit vault

ansible-playbook --ask-vault-pass valt.yml

----------------------------------------

**Removing fatcs to improve the speed (unnessery steps):**

gather\_facts: false

---------------------------------

Extracting Repetitive Tasks : cache\_valid\_time

-name: update yum cache

yum: update\_cache\_valid\_time=86400

-----------------------------------

**Limiting Execution by Hosts : limits**

ansible-playbook status.yml --limit appo1

-----------------------------

**### Limiting Execution by tasks: tag**

---

- hosts: apps

tasks:

- yum:

name:

- httpd

- memcached

state: present

tags:

- packages

- template:

src: /home/ansible/index.html

dest: /etc/foo.conf

tags:

- configuration

ansible-playbook -i ../inventory tags1.yaml --tags "configuration" -b

ansible-playbook -i ../inventory tags1.yaml --skip-tags "configuration" -b

ansible-playbook -i ../inventory tags1.yaml --tags "packages,configuration" -b

### Tag Reuse

--tags=x,y,z, or --tags=all or Specifying --tags=tagged or --tags=untagged or --skip-tags=tagged

-------------------------------

changed\_when, failed\_when

eg:

changed\_when: false

-----------------------------

**Accelerated Mode and Pipelining.**

[ssh\_connection]

pipelining = True

-----------------------------------------------------------------------

#vim site.yml

**Install as normal user which become root user "become: yes"**

-name: install and start apache

Hosts: web

vars:

http\_port: 80

max\_client: 200

remote\_user: vagrant

become: yes

task:

- name: install pep repo

yum: name=epel-release state=present

- name: install python bindings for selinux

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

with\_items:

- libselinux-python

- libsemanage-python

- name: test to see if Selinux is running

Command: getenforce

Register: sestatus

changed\_when: false

- name: install httpd

yum: pkg= httpd state: latest

- name: write the apache config file

template: src=/srv/https.j2 dest=/etc/https.conf

- name: start httpd

service: name=httpd state=started

- name: install one specific version of Apache

yum:

name: httpd-2.2.29-1.4.amzn1

state: present

# ansible-playbook -i hosts site.yml

-----------------------------------------------------

#### **Roles:-** Ansible role is an independent component which allows reuse of common configuration steps. Ansible role has to be used within playbook. Ansible role is a set of tasks to configure a host to serve a certain purpose like configuring a service. Roles are defined using YAML files with a predefined directory structure.

**tasks -** contains the main list of tasks to be executed by the role.

**handlers -** contains handlers, which may be used by this role or even anywhere outside this role.

d**efaults -** default variables for the role (see Using Variables for more information).

**vars** - other variables for the role (see Using Variables for more information).

**files** - contains files which can be deployed via this role.

**templates** - contains templates which can be deployed via this role.

**meta** - defines some meta data for this role. See below for more details.

site.yml

webservers.yml

fooservers.yml

roles/

common/

tasks/

handlers/

files/

templates/

vars/

defaults/

meta/

webservers/

tasks/

defaults/

meta/

# roles/example/tasks/main.yml

- name: added in 2.4, previously you used 'include'

import\_tasks: redhat.yml

when: ansible\_facts['os\_family']|lower == 'redhat'

- import\_tasks: debian.yml

when: ansible\_facts['os\_family']|lower == 'debian'

# roles/example/tasks/redhat.yml

- yum:

name: "httpd"

state: present

# roles/example/tasks/debian.yml

- apt:

name: "apache2"

state: present

----------------------

### Using roles:

---

- hosts: webservers

roles:

- common

- webservers

Simply use ansible-galaxy init <ROLE\_NAME> to create a new role in your present working directory

------------------------------------------

**Creating a roles based on our apache playbook which created above steps "site.yml"**

mkdir roles

cd roles

ansible-galaxy init --help

ansible-galaxy init common

ansible-galaxy init apache

ls apache/

### We will be able to see all roles created

ls common

###We will be able to see all roles created

ls common/tasks

main.yml

------------------------------------------

**How to break existing role structure from site.yml " move the task into respective role folders**

Create a common rules file :

#site.yml

---

- name: install peel repo

yum: name=epel-release state=present

##Include the other iml files for selinux and ntp

- include: selinux.yml

- include: ntp.yml

Next create the apache role:-

---

- name: install apache

yum: name=httpd state=present

- name: create sites directories

file: path={{item}} state=directory

with\_items: "{{apache\_dirs}}"

- name: copy an index.html

template: src=index.html.j2 dest={{apache\_docroot}}/index.html

- name: copy httpd.conf

template: src=httpd.conf-{{ansible\_os\_family}}.j2 dest={{apache\_config}}

notify: restart apache

- name: start apache

Service: name=httpd state=started enabled=yes

### Go to the handlers the add the apache restart task into main.yml

---

- name:restart apache

Service: name=httpd state=restarted

-------------------------------------------------------------

**How to update the existing roles ?**

#ansible-galaxy init web

ls

apache common web

---

- name: install git

Yum: name=git state=present

- name: checkout lame app

git: repo=https://github.com/jsmartin/lameapp.git version="{{lameapp\_version\string}}"

- name: set permissions on the app

file: name=/var/www/lameapp/lame.py mode=0755

- name: add apache config file

Copy: src=lameapp.conf dest={{sites\_available}}

notify: restart apache

- name: link app config

file: src="{{sites\_available}}/lameapp.conf" dest={{sites\_enabled}}/lameapp.comf state=link

- meta: flush\_handlers

- name: check for proper response

uri:

url: http://localhost/lame

return\_content: yes

register: result

until: "Hello Moon" in result.content'

###Next update into main.yml

---

- name: restart apache

service: name=httpd state=restarted

-------------------------------------------------------------'

**Utilizing roles in your main playbooks:?**

---

- name: apply the common configuration to all hosts

hosts: all

remote\_user: vagrant

become: yes

roles:

-common

- name: apply the db configuration

hosts: dbservers

remote\_user: vagrant

become: yes

roles:

- apache

- web

- name: apply the lb configuration

hosts: observers

remote\_user: vagrant

become: yes

roles:

- haproxy

# ansible-playbook -i hosts site.yml

###################################################################################

################# Ansible Tower ######################

this is a web based solution for managing your organozation.

**Ansible tower Organization is a logical collection of teams, projects and Inventories.**

Key features:--

1. Visual Dashboard

2. RBAC "Role based access control management "

3. Job scheduling "schedule the job in certain time"

4. Real time job updates " Automate the env by ansible "

5. Integrated Notify " if job success/fails " to team

6. Multiple playbook workflows. " chain any no of playbooks"

7. Rest API " Interrogate to satellite or cloudform or jenkies"

Ansible tower using Postgre sql, earlier it was mongo db:

Transport mechanism "to connect with hosts by local-transport. eg: Smart ssh for linux, winrm for windows, docker for docker container"

**IMPORTENT Parts:-**

1. Ansible tower + Postgre SQL

2. Rabbit MQ Messanging

3. Ansible tower 3.1 Supporting active cluster, before it was active/passive cluster

4. Licensing a) Basic b) Enterprises " All features "

#ansible -m command -a "uptime" testing

#### Ansible tower Installation :-

Ansible Core 2.6.X or later need to install on RHEL 7.X, CentOS7.X or Ubuntu 16.04LTS

Download the tar and extract, then unzip and keep into /ansibletower folder

Then setup edit the file inventory >> admin\_password, pg\_password, rabbitmq\_password.]

Run ./setup.sh

Setup log file location:-

#cat /var/log/tower/setup-2019-05XXXX.log

### Goto the Browser and enter IP or FQDN of ansibltower server > give the license

### What is project in ansible-tower ?

Run Any playbook (Task) will be define under projects.

The Ansible playbook location will be under below path:-

Create the Directory

#/var/lib/awx/projects

#mkdir MYREPO

#Update a content into file

vi content.yaml

---

- hosts: all

tasks:

- name: Content Update

copy:

dest: /tmp/resolv.conf

content: "nameserver 192.168.0.2\n"

#### create new project in ansible tower for copy the content. Then create Inventory.

Note# One directory can align with only one project

### Create the templets.

-------------------------------------------------------

To check the httpd service status check.

#vi service.yml

---

- hosts: all

task:

- name: Restarted httpd Service

service: name=httpd enabled=true state=started

#### Place into /var/lib/awx/projects/PLAYREPO

### Create a Job templates.

Go to options and select Enable Privilege Escalation to run as root to restart services by normal user.

-------------------------------------------------------

**### RBAC. " Role based access controls.**

Create a User via Web UI.

1. Some users need to run pre-defined templates

2. Some users need to modify the playbooks and inventories

3. Some users need to modify the Ansible tower installation.

4. By default admin user will be created and will have full access.

a. Tower user can create 3 types of users

1. System Administrator (Full access)

2. System Auditor (Full read access across the Ansible tower)

3. Normal User (doesn't have any read access at all)

Additional permission (role) can grand a normal user feature.

###Delegating permission to a User###

##Select the user and click on add + simple.>> Then select the job templets then select template >> select Action "Admin or Execute or read" based on the demand.

##OtherWay got to the inventory >> the click on add + simple and add the permission to specify user.

##Other way a User need to run Ad-hok command into specific hosts.

Go inventory >> the select hosts >> select run command

**### Managing Users with Teams:-**

Team Members will get the permission inherited.

Here we can make user as admin in the groups or member

We can give permission from person tab in the groups. " read, execute,Amin "

How to rsync and SCP in vagrent box.?

Ans: rsync -avz -e 'ssh -p 2222' /tmp/day3/\* ansible@127.0.0.1:/home/ansible/day3/

scp -r -P 2222 /tmp/day3/\* ansible@127.0.0.1:/home/ansible/day3/

You can try

ssh server -l user -o "PubkeyAuthentication=no"

Rename a set of files name to sns to sqs :-

rename 's/\-sns.\*$/\-sqs.jpeg/' \*

Default Bastion UID: guacadmin

Password would be instance ID

Reference url: <http://netcubed-ami.s3-website-us-east-1.amazonaws.com/guaws/v2.1.0/>

## Add this into userdata.

sudo useradd ansari; (echo redhat; echo redhat) | sudo passwd ansari

echo 'ansari ALL=(ALL) NOPASSWD: ALL' | sudo EDITOR='tee -a' visudo

sudo sed -i "/^[^#]\*PasswordAuthentication[[:space:]]no/c\PasswordAuthentication yes" /etc/ssh/sshd\_config

#!/bin/bash

sudo sed -i "/^[^#]\*PasswordAuthentication[[:space:]]no/c\PasswordAuthentication yes" /etc/ssh/sshd\_config

sudo useradd ansari; (echo redhat; echo redhat) | sudo passwd ansari

(echo redhat; echo redhat) | sudo passwd ec2-user

echo 'ansari ALL=(ALL) NOPASSWD: ALL' | sudo EDITOR='tee -a' visudo

echo 'ec2-user ALL=(ALL) NOPASSWD: ALL' | sudo EDITOR='tee -a' visudo

sudo service sshd restart

sudo yum install -y httpd

sudo yum install -y php

sudo systemctl enable httpd

sudo service httpd start

cat <<EOF >/var/www/html/index.php

**<?php phpinfo(); ?>**

EOF

#sudo yum install epel-release

sudo amazon-linux-extras install epel -y

sudo yum install stress -y

sudo service sshd restart

for centos sudo yum install epel-release