**Ansible is configuration management tool, orchestration, and deployment.**

**Ansible is agentless. So it will communicate with nodes (other PS’s using SSH)**

**PUSH mechanism (We can do as we want, when to do, how to do), so not 100% automation tool.**

**Ansible tower used by CentOS is GUI based, but still in progress.**

**Ansible server uses YAML, Yet another markup language.**

**Playbook is there in ansible server that is used to give commands and to write YAML.**

**Terms:**

Ansible server, the machine where ansible is installed and from which all the tasks and modules will run

Modules, Commands that are built in and we will use.

Task, single procedure to be completed.

Role, A way of organizing tasks and related files, those to be called later in the playbook.

Inventory, Ip list of all hosts, ansible client-servers information

Play, execution of playbook

Handler, task which works when notifier is present.

Notifier, section attributed to a task which calls the handler if output is changed

Playbook, It consists of code in YAML which describes tasks to be executed

Host, nodes which are automated by the ansible server, nodes/client-servers

**Step by Step Tutorial by Adnan:**

$ sudo apt update

$ sudo apt install software-properties-common

$ sudo add-apt-repository --yes --update ppa:ansible/ansible

$ sudo apt install ansible

ansible all -m ping // TO check

ansible-inventory --list –y // To check the inventory

First of all make 3 instances and 3 virtual servers using aws EC2

Then, start all of them.

Install ansible there : apt install ansible

**Now, create group inside the /etc/ansible/hosts**

**[Demo]**

**Private IP of 1st node machine**

**Private IP of 2nd node machine**

**Then, Goto the .cfg file that is under /etc/ansible/ansible.cfg**

**Inside that, uncomment the line of Inventory file & root user**

Create user named as ansible, **adduser ansible in all of the machines.**

**We have to give sudoer priveledges to the new user we created as ansible,**

**Use this command “Visudo" to give sudoer facility to ansible user.**

**Write in the root area,**

**ansible ALL=(ALL) NOPASSWD: ALL**

**OK that’s done!**

I am doing now the main job as connecting the ansible main machine or server to the node machines. I will do these things using SSH and one more thing is needed that is “private Ip of node machines”

**Ssh <Private Ip of node>**

This is not working showing that, “Permission denied”

We will go to the configuration file of SSH, that is

**Vi /etc/ssh/sshd\_config**

**Uncomment the permitRootLogin**

**Uncomment the PassowrdAuthentication yes, comment the rest one**

**So, do it for all the machines, ansible and nodes.**

**Now, we need to restart the ssh:**

**Service sshd restart**

**Ok, now run the command:**

Ssh <Private Ip of node>

And, now you can access the node machines.

But every time you access to the node machines, nodes will ask for the password.

For this job, you have to do one thing, you will go to the ansible main server machine and generate (private + Public keys), then copy that to the node machines, so it won't ask that again otherwise, it will destroy the time and if we have hundreds of servers, will take permission every time.

This is called as trust relationship, so be sure that, you are at the same user in all machines.

For this, first create keys in the ansible server/user:

**Ssh-keygen**

**ls –a**

**Cd .ssh**

**Ssh-copy-id ansible@<privateID> // user@node1PrivateId**

**Ssh-copy-id ansible@<privateID> // user@node2PrivateId**

**Now, if we want to see, which of the node machines are connected to eachother, and which are actually hosts with the ansible server.**

**Ansible all –-list-hosts**

**Ansible <group name> --list-hosts**

**Ansible <group name> [0:10] --list-hosts**

**Goto the /etc/ansible/**

nano ansible.cfg

Uncomment the inverntory line

Make a directory and create a file.yml for the playbook.

---

- hosts: all

become: true

vars\_files:

- vars/default.yml

tasks:

- name: Install prerequisites

apt: name=aptitude update\_cache=yes state=latest force\_apt\_get=yes

tags: [ system ]

- name: Install LAMP Packages

apt: name={{ item }} update\_cache=yes state=latest

loop: [ 'apache2', 'mysql-server', 'python3-pymysql', 'php', 'php-mysql', 'libapache2-mod-php' ]

tags: [ system ]

- name: Install PHP Extensions

apt: name={{ item }} update\_cache=yes state=latest

loop: "{{ php\_modules }}"

tags: [ system ]

# Apache Configuration

- name: Create document root

file:

path: "/var/www/{{ http\_host }}"

state: directory

owner: "www-data"

group: "www-data"

mode: '0755'

tags: [ apache ]

- name: Set up Apache VirtualHost

template:

src: "files/apache.conf.j2"

dest: "/etc/apache2/sites-available/{{ http\_conf }}"

notify: Reload Apache

tags: [ apache ]

- name: Enable rewrite module

shell: /usr/sbin/a2enmod rewrite

notify: Reload Apache

tags: [ apache ]

- name: Enable new site

shell: /usr/sbin/a2ensite {{ http\_conf }}

notify: Reload Apache

tags: [ apache ]

- name: Disable default Apache site

shell: /usr/sbin/a2dissite 000-default.conf

notify: Restart Apache

tags: [ apache ]

# MySQL Configuration

- name: Set the root password

mysql\_user:

name: root

password: "{{ mysql\_root\_password }}"

login\_unix\_socket: /var/run/mysqld/mysqld.sock

tags: [ mysql, mysql-root ]

- name: Remove all anonymous user accounts

mysql\_user:

name: ''

host\_all: yes

state: absent

login\_user: root

login\_password: "{{ mysql\_root\_password }}"

tags: [ mysql ]

- name: Remove the MySQL test database

mysql\_db:

name: test

state: absent

login\_user: root

login\_password: "{{ mysql\_root\_password }}"

tags: [ mysql ]

- name: Creates database for WordPress

mysql\_db:

name: "{{ mysql\_db }}"

state: present

login\_user: root

login\_password: "{{ mysql\_root\_password }}"

tags: [ mysql ]

- name: Create MySQL user for WordPress

mysql\_user:

name: "{{ mysql\_user }}"

password: "{{ mysql\_password }}"

priv: "{{ mysql\_db }}.\*:ALL"

state: present

login\_user: root

login\_password: "{{ mysql\_root\_password }}"

tags: [ mysql ]

# UFW Configuration

- name: "UFW - Allow HTTP on port {{ http\_port }}"

ufw:

rule: allow

port: "{{ http\_port }}"

proto: tcp

tags: [ system ]

# WordPress Configuration

- name: Download and unpack latest WordPress

unarchive:

src: https://wordpress.org/latest.tar.gz

dest: "/var/www/{{ http\_host }}"

remote\_src: yes

creates: "/var/www/{{ http\_host }}/wordpress"

tags: [ wordpress ]

- name: Set ownership

file:

path: "/var/www/{{ http\_host }}"

state: directory

recurse: yes

owner: www-data

group: www-data

tags: [ wordpress ]

- name: Set permissions for directories

shell: "/usr/bin/find /var/www/{{ http\_host }}/wordpress/ -type d -exec chmod 750 {} \\;"

tags: [ wordpress ]

- name: Set permissions for files

shell: "/usr/bin/find /var/www/{{ http\_host }}/wordpress/ -type f -exec chmod 640 {} \\;"

tags: [ wordpress ]

- name: Set up wp-config

template:

src: "files/wp-config.php.j2"

dest: "/var/www/{{ http\_host }}/wordpress/wp-config.php"

tags: [ wordpress ]

handlers:

- name: Reload Apache

service:

name: apache2

state: reloaded

- name: Restart Apache

service:

name: apache2

state: restarted

We can use the -l flag to make sure that only a subset of servers, or a single server, is affected by the playbook. We can also use the **-u** flag to specify which user on the remote server we’re using to connect and execute the playbook commands on the remote hosts.

**Check if ufw is enabled or not:**

**Sudo ufw status**

**Sudo ufw enable**

**Sudo ufw allow ssh**

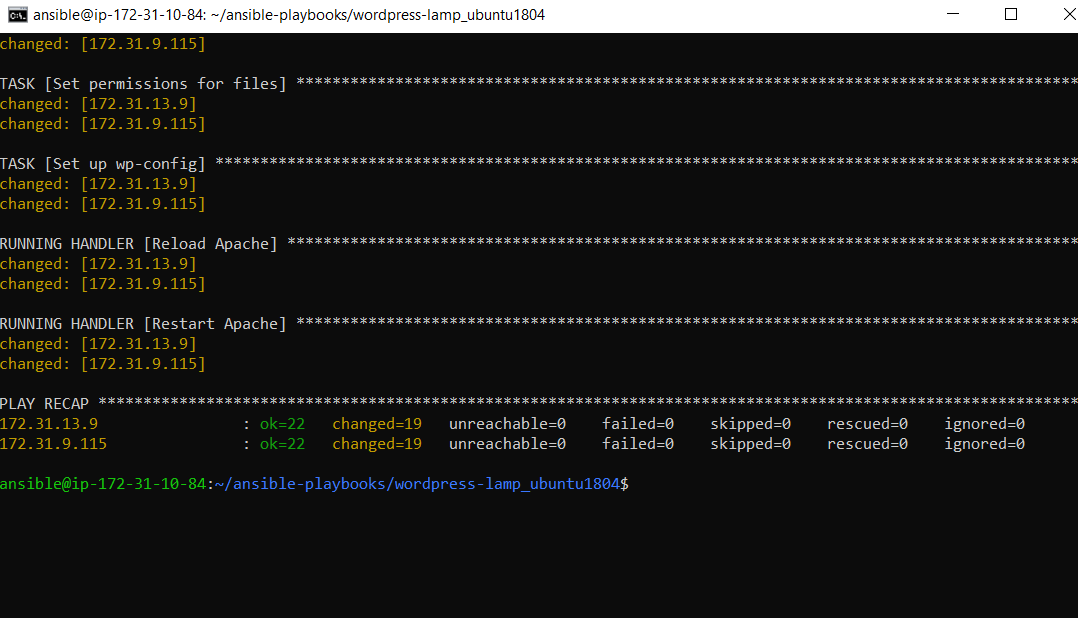
**Stay on the same level/user when doing ssh.**

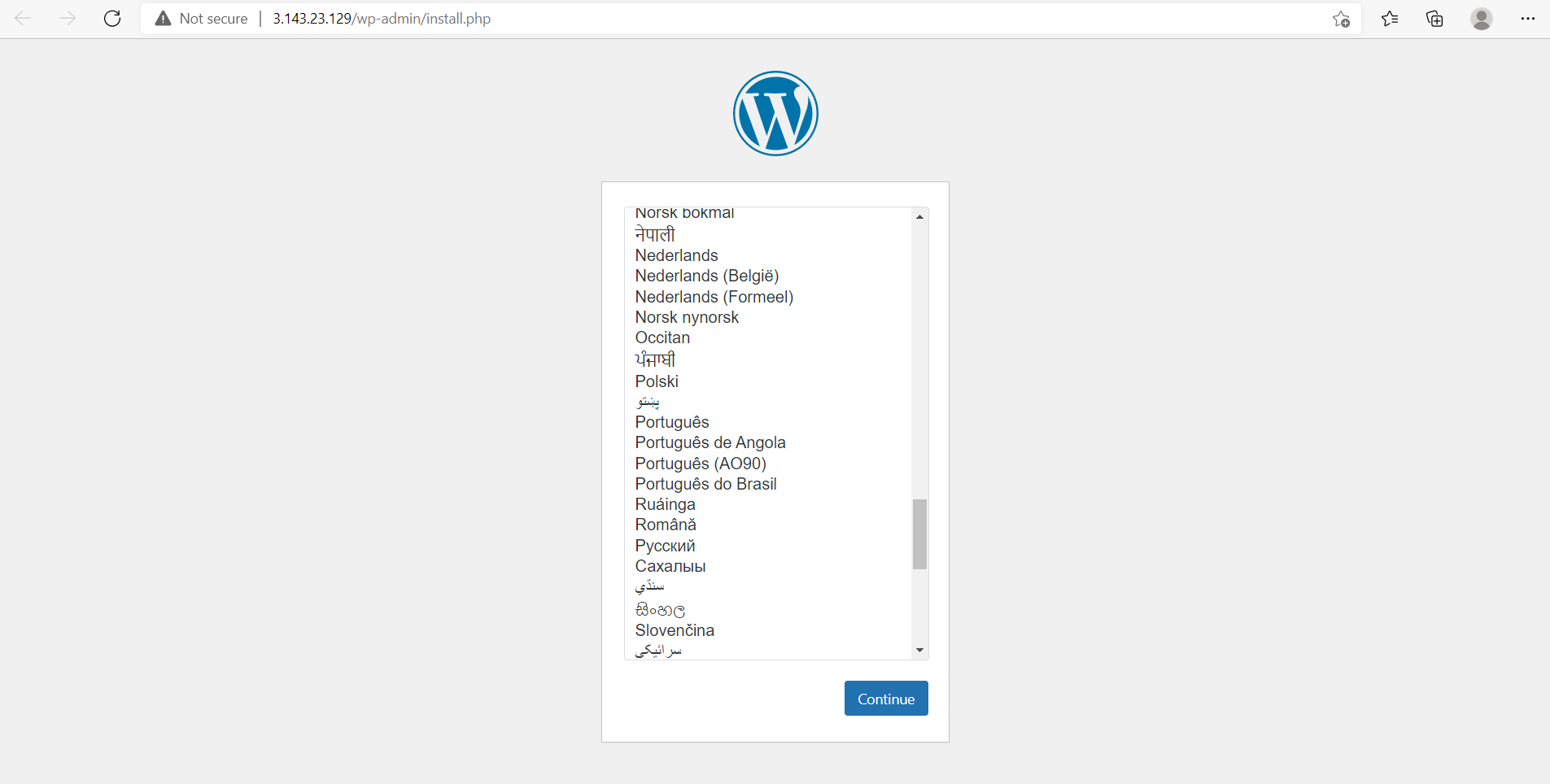
**PLay the book with command:**

**Ansible-playbook book\_name.yml -K**

**::where –K means to ask the Become password for other hosts.**

**Now when done, run the Public IP address of your instance with http:**





**Further you will setup all the things related to the wordopress, and give the passwords to continue to the dashboard of wordpress.**

**Task 2 :**

# **How to Use Ansible to Install and Set Up Docker**

FIrst of all I started with the same procedure I did while installing wordpress, in the hosts file, put my remote hosts Private IP addresses and then go for the variables. Inside variables, I used:

create\_containers: 1

default\_container\_name: docker

default\_container\_image: ubuntu

default\_container\_command: sleep 1d

**We can make these inside the playbook, but it makes more organized and easy.**

**After this I wrote a playbook by creating a file as:**

**Touch playbook.yml**

**I wrote following code into that:**

---

- hosts: all

become: true

vars\_files:

- vars/default.yml

tasks:

- name: Install aptitude using apt

apt: name=aptitude state=latest update\_cache=yes force\_apt\_get=yes

- name: Install required system packages

apt: name={{ item }} state=latest update\_cache=yes

loop: [ 'apt-transport-https', 'ca-certificates', 'curl', 'software-properties-common', 'python3-pip', 'virtualenv', 'python3-setuptools']

- name: Add Docker GPG apt Key

apt\_key:

url: https://download.docker.com/linux/ubuntu/gpg

state: present

- name: Add Docker Repository

apt\_repository:

repo: deb https://download.docker.com/linux/ubuntu bionic stable

state: present

- name: Update apt and install docker-ce

apt: update\_cache=yes name=docker-ce state=latest

- name: Install Docker Module for Python

pip:

name: docker

- name: Pull default Docker image

docker\_image:

name: "{{ default\_container\_image }}"

source: pull

# Creates the number of containers defined by the variable create\_containers, using values from vars file

- name: Create default containers

docker\_container:

name: "{{ default\_container\_name }}{{ item }}"

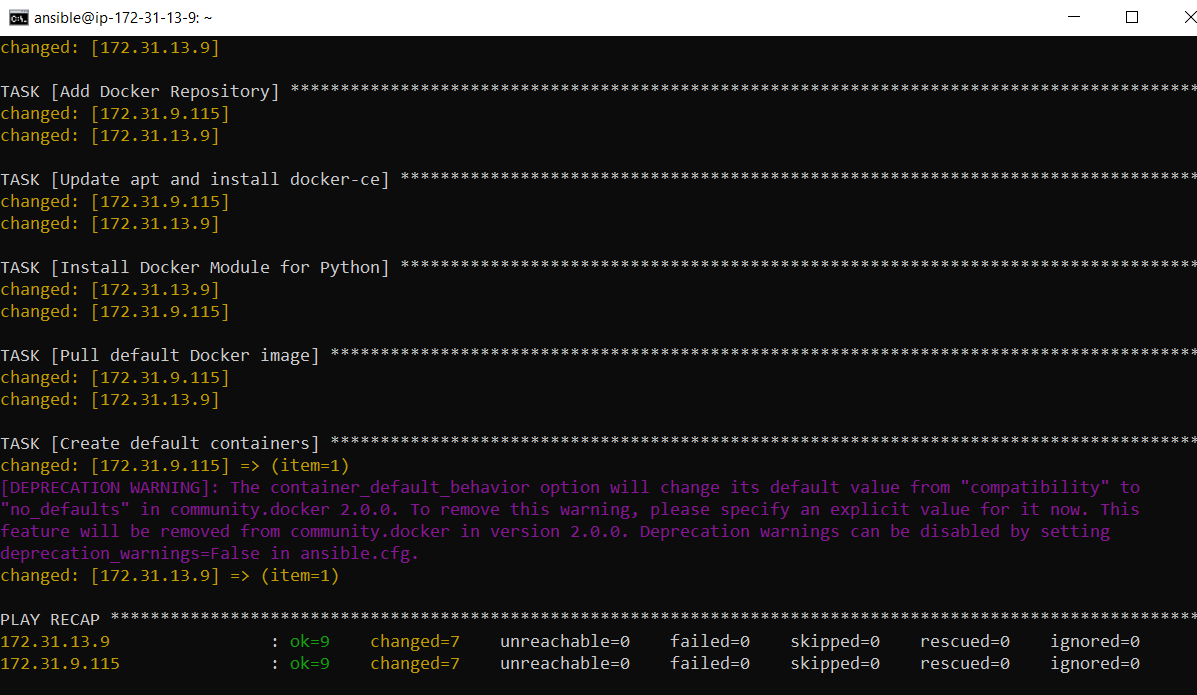
image: "{{ default\_container\_image }}"

command: "{{ default\_container\_command }}"

state: present

with\_sequence: count={{ create\_containers }}

**Its working very fine as attached in SS:**



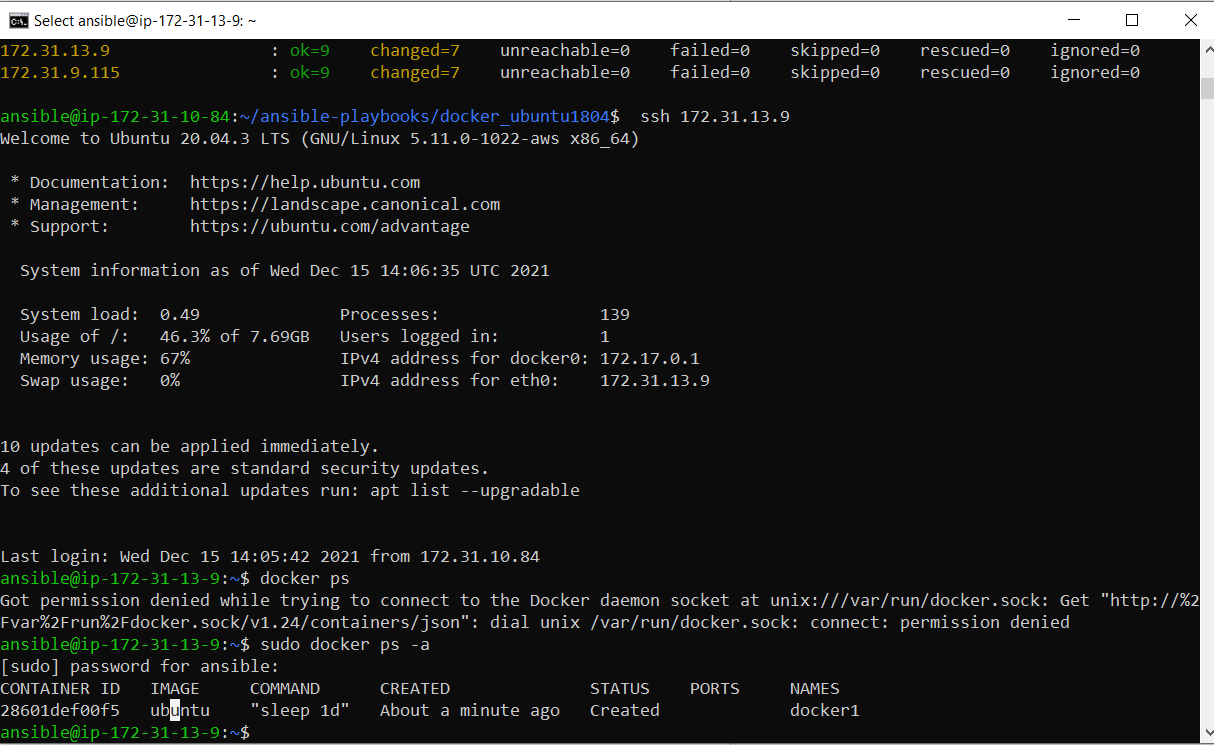
**Lets go inside the other remote hosts and check it works or not:**

**Run the command:**

**SSH ansible@<IP of remote host>**

**When login to the remote host, run the command:**

**docker ps –a //it will tell us the running containers.**



**DockerFile use with ansible:**

**Write Docker file:**

**For removing interation with tzdata**

ARG DEBIAN\_FRONTEND=noninteractive

DEBIAN\_FRONTEND=noninteractive

**Write this inside the Docker file**

FROM ubuntu:latest

ARG DEBIAN\_FRONTEND=noninteractive

RUN apt update -y

RUN apt install wordpress -y

RUN apt install mysql-server -y

RUN apt install php -y

RUN apt install nginx -y

EXPOSE 80

Then, go for running the ansible playbook, I write this playbook for this as:

Here is the code I write for ansible playbook:

---

- hosts: all

become: yes

tasks:

- name: Makuing a directory

file:

path: ./docker-file

state: directory

mode: '0755'

- name: copying files from local to remote server

copy:

src: /home/ansible/ansible-playbooks/wordpress/Dockerfile

dest: ./docker-file/Dockerfile

- name: Update apt cache

apt: update\_cache=yes cache\_valid\_time=3600

- name: Upgrade all apt packages

apt: upgrade=dist

- name: Install dependencies

apt:

name: "{{ packages }}"

state: present

update\_cache: yes

vars:

packages:

- apt-transport-https

- ca-certificates

- curl

- software-properties-common

- gnupg-agent

- name: Install Docker

apt:

name: "{{ packages }}"

state: present

update\_cache: yes

vars:

packages:

- docker-ce

- docker-ce-cli

- containerd.io

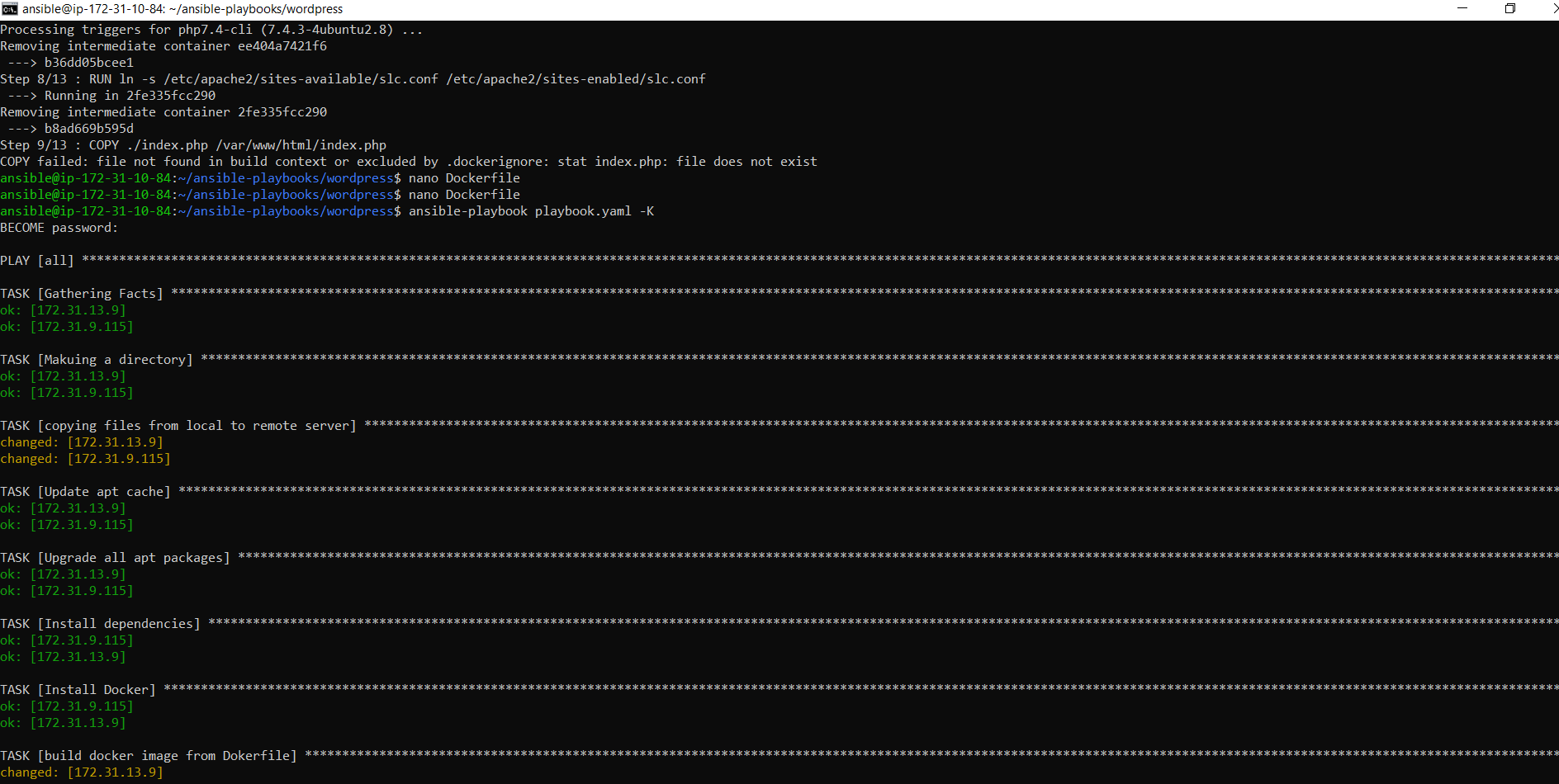
- name: build docker image from Dokerfile

command: docker build -t wordpress-img ./docker-file/

- name: run a container

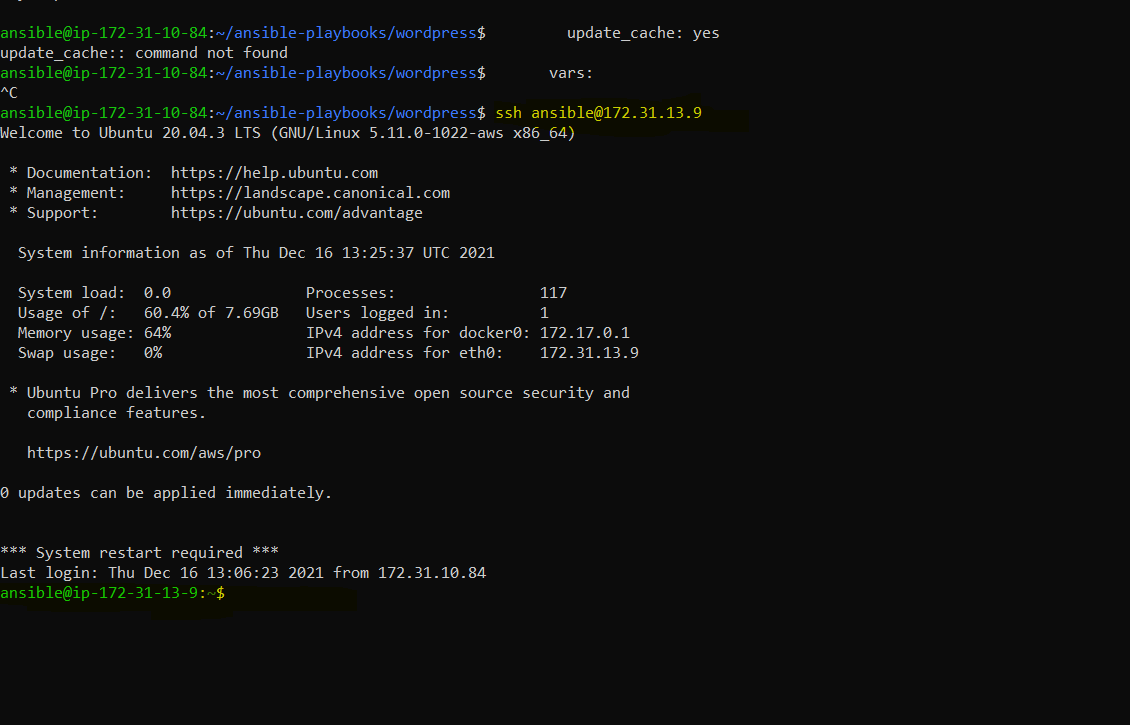
command: docker run --name container1 -dit -p 8080:80 wordpress-img

...



Now lets check, in our node machines, it works or not, For accessing that, enter the command in the main host:

Ssh ansible@<private-IP>

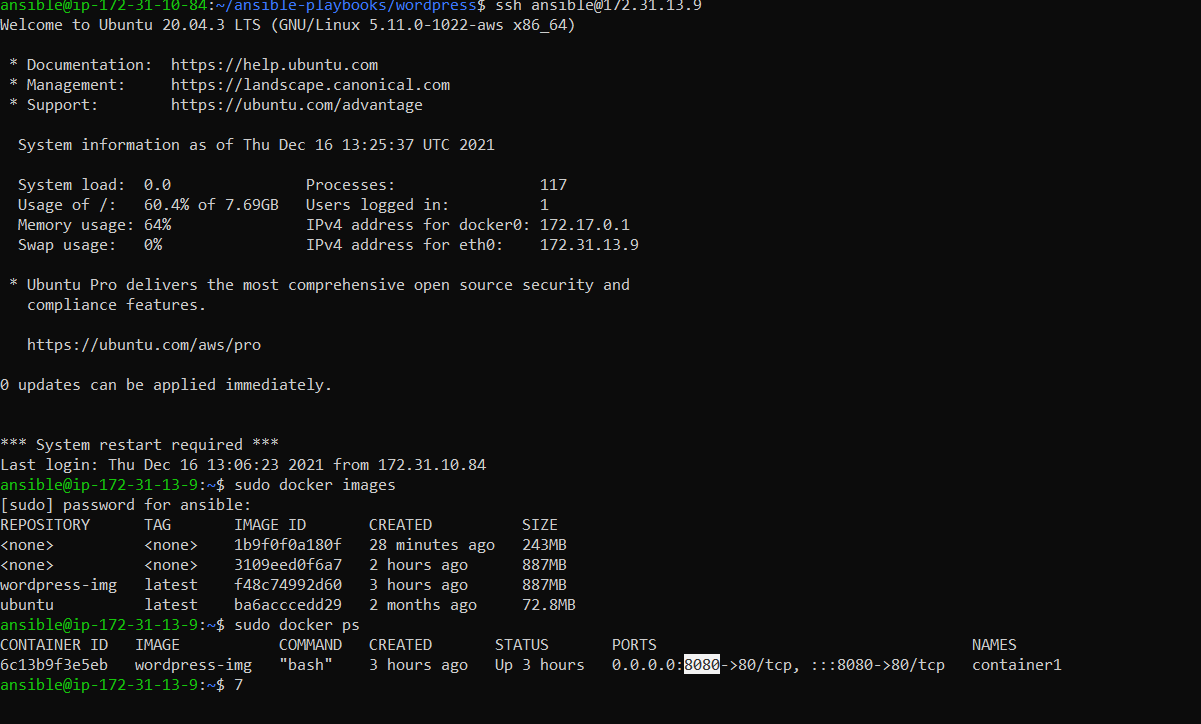


**Now, lets check that the image is created and docker container is running in other machine:**

**Use commands:**

**Sudo docker images**

**Sudo docker ps**



**Task 2:**

# **How to Use Ansible to Install and Set Up Docker-compose and installing the wordpress**

First of all we will see the docker-compose file to install wordpress, we need:

Mysql

Webserver (nginx)

Php

& wordpress

All these to install with docker-compose.

So lets create a dcker-compose.yml file

ENter a conmmand as: nano docker-compose:

And write this code into that;

version: '3'

services:

db:

image: mysql:8.0

container\_name: db

restart: unless-stopped

env\_file: .env

environment:

- MYSQL\_DATABASE=wordpress

volumes:

- dbdata:/var/lib/mysql

command: '--default-authentication-plugin=mysql\_native\_password'

networks:

- app-network

wordpress:

depends\_on:

- db

image: wordpress:5.1.1-fpm-alpine

container\_name: wordpress

restart: unless-stopped

env\_file: .env

environment:

- WORDPRESS\_DB\_HOST=db:3306

- WORDPRESS\_DB\_USER=$MYSQL\_USER

- WORDPRESS\_DB\_PASSWORD=$MYSQL\_PASSWORD

- WORDPRESS\_DB\_NAME=wordpress

volumes:

- wordpress:/var/www/html

networks:

- app-network

webserver:

depends\_on:

- wordpress

image: nginx:1.15.12-alpine

container\_name: webserver

restart: unless-stopped

ports:

- "80:80"

volumes:

- wordpress:/var/www/html

- ./nginx-conf:/etc/nginx/conf.d

- certbot-etc:/etc/letsencrypt

networks:

- app-network

certbot:

depends\_on:

- webserver

image: certbot/certbot

container\_name: certbot

volumes:

- certbot-etc:/etc/letsencrypt

- wordpress:/var/www/html

command: certonly --webroot --webroot-path=/var/www/html --email adnanrafique@example.com --agree-tos --no-eff-email --staging -d example.com -d www.example.com

volumes:

certbot-etc:

wordpress:

dbdata:

networks:

app-network:

driver: bridge

Save and close the file and lets move towards the configuration of our webserver and wordpress. First of all to host our wordpress website, we are using **nginx,** so for that;

Lets create a directory:

mkdir nginx-conf

Now, make a file in it and go to editing with:

nano nginx-conf/nginx.conf

We have to write this code into this for configuration of nginx:

server {

listen 80;

listen [::]:80;

server\_name adnanrafique.com www.adnanrafique.com;

index index.php index.html index.htm;

root /var/www/html;

location ~ /.well-known/acme-challenge {

allow all;

root /var/www/html;

}

location / {

try\_files $uri $uri/ /index.php$is\_args$args;

}

location ~ \.php$ {

try\_files $uri =404;

fastcgi\_split\_path\_info ^(.+\.php)(/.+)$;

fastcgi\_pass wordpress:9000;

fastcgi\_index index.php;

include fastcgi\_params;

fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;

fastcgi\_param PATH\_INFO $fastcgi\_path\_info;

}

location ~ /\.ht {

deny all;

}

location = /favicon.ico {

log\_not\_found off; access\_log off;

}

location = /robots.txt {

log\_not\_found off; access\_log off; allow all;

}

location ~\* \.(css|gif|ico|jpeg|jpg|js|png)$ {

expires max;

log\_not\_found off;

}

}

Save this file and exit that.

Lets move towards the ansible playbook to write and use our docker-compose in that playbook to make installations and do our job in remote hosts:

Here is the playbooks’s code to write, follow this;

Now create e new file with name ansible.yml and this file will contain ansible code that will execute in remote machine.

---

- hosts: servers

remote\_user: temp

become: yes

tasks:

- name: Making a directory in remote machine to copy our required file

file:

path: ./ansible-dockercompose-lemp-wordpress

state: directory

mode: '0755'

- name: Making a directory in remote machine to copy our nginx configuration file

file:

path: ./ansible-dockercompose-lemp-wordpress/nginx-conf

state: directory

mode: '0755'

- name: Copying docker-compose file in the ~/ansible-dockercompose-lemp-wordpress

copy:

src: ~/ansible-dockercompose-lemp-wordpress/docker-compose.yml

dest: ./ansible-dockercompose-lemp-wordpress/docker-compose.yml

- name: Copying .env file in the ~/ansible-dockercompose-lemp-wordpress

copy:

src: ~/ansible-dockercompose-lemp-wordpress/.env

dest: ./ansible-dockercompose-lemp-wordpress/.env

- name: Copying nginx.conf file in the ~/ansible-dockercompose-lemp-wordpress

copy:

src: ~/ansible-dockercompose-lemp-wordpress/nginx-conf/nginx.conf

dest: ./ansible-dockercompose-lemp-wordpress/nginx-conf/nginx.conf

- name: Installing Docker

apt:

name: docker

state: latest

- name: Running docker-compose commands to complete inatallations

docker\_compose:

project\_src: ./ansible-dockercompose-lemp-wordpress

files:

- docker-compose.yml

Save and exit this file too

Now all we need to do is create a SSH connection between the two machines.

Install SSH on both machines.

sudo apt install ssh

Enable ufw services.

sudo ufw enable

Add ssh into firewall.

sudo ufw allow ssh

This will allow ssh traffic on port 22.

To confirm write

sudo ufw status

If still showing error, go to the configuration file of ssh and write:

nano ~/ssh/config

Add these lines into the file config;

Host ubuntu

HostName fsdf.us-east-2.compute.amazonaws.com User ubuntu

Port 22

IdentitiesOnly yes

IdentityFile ~/.ssh/fds-dfas-fdsa.pem

Now do ssh ansible@<IP addr> to ssh into the remote machines.

Ok that’s fine, now enter the command to run the playbook of ansible we have created:

Use the command as:

ansible-playbook ansible.yml --ask-pass –K

It is running successfully, that’s all fine

Now go to the remote machines and check the images and containers created with docker-compose.

Docker images

Docker ps

So, they are running, that’s all.

Now, we can hit the public IP address of our instance that are using with node machines we have created to see if our installation works or not.

