Deployment of WordPress Environment

By Lei Liu

Introduction of the project

The goal of this project is to create a standard WordPress(WP) launching package for new servers using configuration management tools such as Terraform & Ansible, in which Terraform facilitates server infrastructure, Ansible used for installing WP with associated dependencises such as Apache2, PHP, MySQL database using playbook within the server environment.

The benefits of using configuration management (CM) for server automation include the ability of reuse code/provisioning scripts for new servers, keeping track of changes, it also enables you to control one to countless of servers from a centralized point.

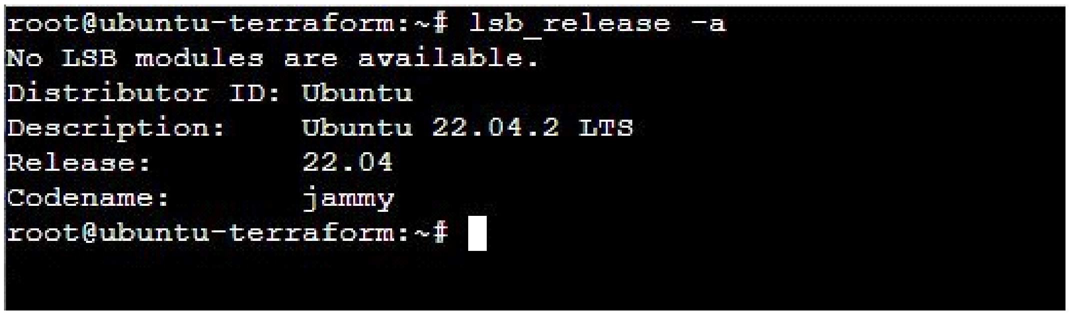
Therefore, besides being a time saver, CM also improves efficiency, decreases the operational costs and human error normally inherited from traditional server deployment.

STEP ONE- Prerequisites

Setting up one Ansible control node on Ubuntu 22.04 VM

Verifying Ubuntu system

lsb_release -a



```
root@ubuntu-terraform:~# lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 22.04.2 LTS
Release:        22.04
Codename:       jammy
root@ubuntu-terraform:~#
```

Installing and verifying Ansible on control node ubuntu-terraform

1. logging into Ubuntu VM

2. Installing Ansible.

# apt install -f

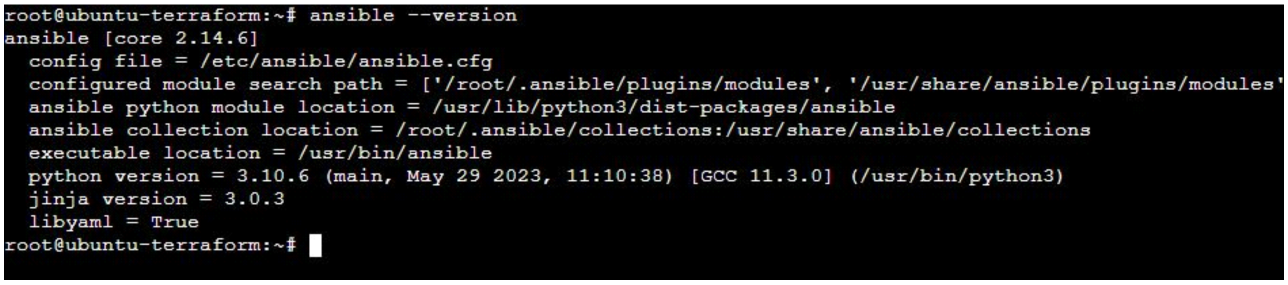
# apt install software-properties-common

# apt-add-repository ppa:ansible/ansible

# apt update

# apt install ansible

3. verifying Ansible installation



```
root@ubuntu-terraform:~# ansible --version
ansible [core 2.14.6]
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/root/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules'
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  ansible collection location = /root/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.10.6 (main, May 29 2023, 11:10:38) [GCC 11.3.0] (/usr/bin/python3)
  jinja version = 3.0.3
  libyaml = True
root@ubuntu-terraform:~#
```

4. generating rsa key pair on Ansible control node

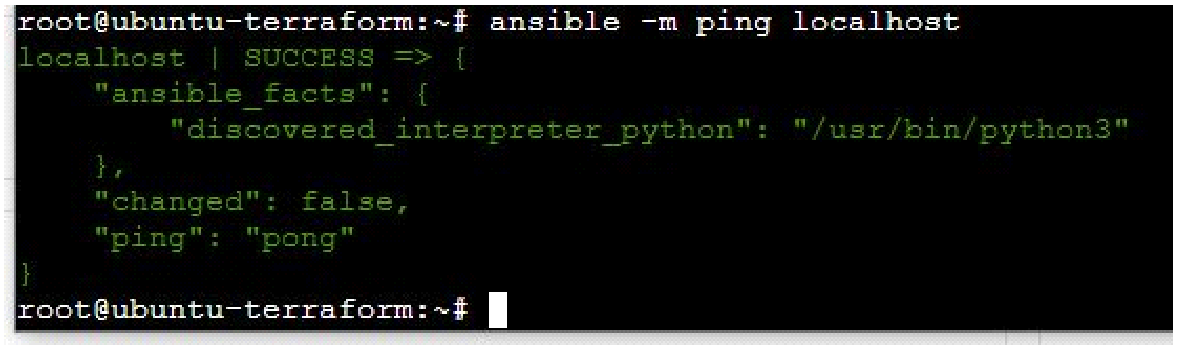
# ssh-keygen

5. coping the pub key into authorized_keys file

# cat .ssh/id_rsa.pub >> .ssh/authorized_keys

6. verifying ssh setup on Ansible control node

ansible -m ping localhost



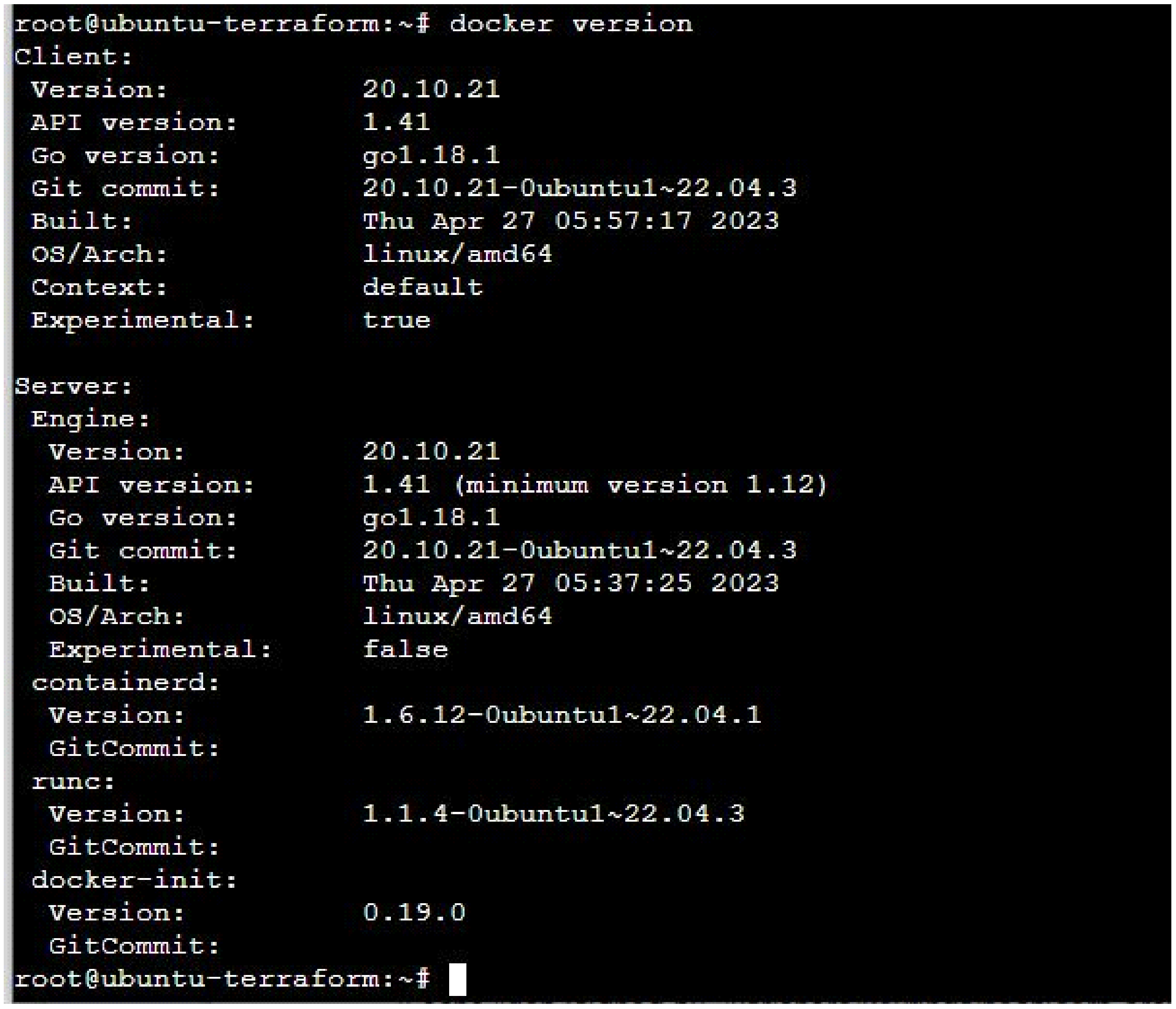
```
root@ubuntu-terraform:~# ansible -m ping localhost
localhost | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
root@ubuntu-terraform:~#
```

Installing Docker and verifying Docker installation (in case if needed)

# apt update

# apt install -y docker.io

docker version



```
root@ubuntu-terraform:~# docker version
Client:
 Version:           20.10.21
 API version:       1.41
 Go version:        go1.18.1
 Git commit:        20.10.21-0ubuntu1~22.04.3
 Built:             Thu Apr 27 05:57:17 2023
 OS/Arch:           linux/amd64
 Context:           default
 Experimental:      true

Server:
 Engine:
  Version:          20.10.21
  API version:      1.41 (minimum version 1.12)
  Go version:       go1.18.1
  Git commit:       20.10.21-0ubuntu1~22.04.3
  Built:            Thu Apr 27 05:37:25 2023
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.6.12-0ubuntu1~22.04.1
  GitCommit:
 runc:
  Version:          1.1.4-0ubuntu1~22.04.3
  GitCommit:
 docker-init:
  Version:          0.19.0
  GitCommit:
root@ubuntu-terraform:~#
```

Installing and verifying Terraform on Ansible control node

1. browsing
https://developer.hashicorp.com/terraform/downloads?product_intent=terraform

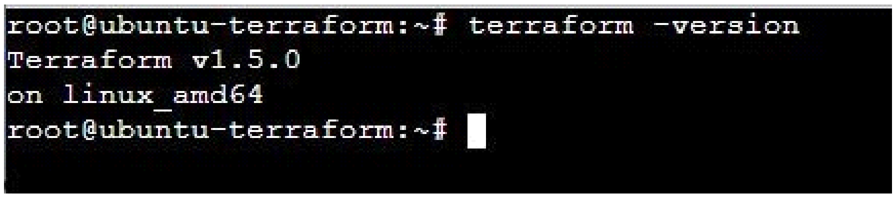
2. clicking Install, search Ubuntu/Debian installation package from Terraform webpage. using below command to install Terraform on control node.

# wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg

# echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list

# sudo apt update && sudo apt install terraform

3. verifying Terraform installation



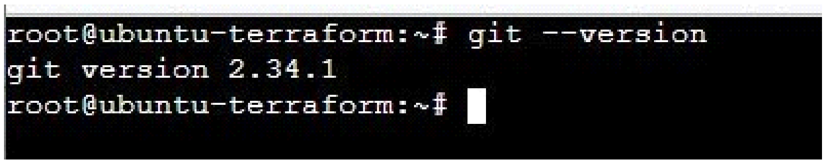
```
root@ubuntu-terraform:~# terraform -version
Terraform v1.5.0
on linux_amd64
root@ubuntu-terraform:~#
```

Installing and verifying git on Ansible control node

1. # apt update

2. # apt install git-all

3. verifying git installation



```
root@ubuntu-terraform:~# git --version
git version 2.34.1
root@ubuntu-terraform:~#
```

Preparing for WP server deployment on AWS

- Creating an AWS account

- Setting up an AWS user account

1. logging into https://aws.amazon.com/, I have created an account and an IAM user, I will log into the IAM user account to demostrate the process. Below is the account info on AWS:

Account ID/alias: laura-devops

Username: laura

Password: xxxxx

2. locating and copy previouse generated ssh rsa key pair from Ansible control node to enable ssh connection between Ansible control node and host node on AWS.

- # cat .ssh/id_rsa.pub

- coping the content of the key

- loggging into AWS management console -> EC2 -> Network & Security -> Key Pairs -> Action -> Import key pair

- pasting the key content from Ansible engine here, name the key ansible-engine-pub-key

3. generating IAM user access_key and secret_key from AWS IAM user account

- logging into aws management console -> IAM -> Users

- selectting user laura -> Security credentials

- access keys -> create access key

- naming the key -> create the key

- downloading the .csv file for later use

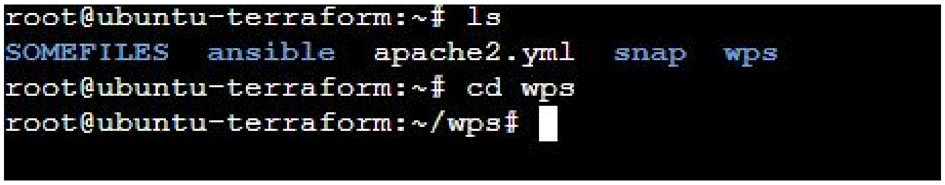
*******************************End of Prerequsites**

STEP TWO - Provisioning EC2 Instance on AWS using Terraform and using the Instance as a Ansible host node.

Creating a new folder wps on Ansible control node.

# mkdir wps

cd wps



```
root@ubuntu-terraform:~# ls
SOMEFILES  ansible  apache2.yml  snap  wps
root@ubuntu-terraform:~# cd wps
root@ubuntu-terraform:~/wps#
```

Creating a new Terraform configuration file main.tf under wps folder to provision EC2 Instance:

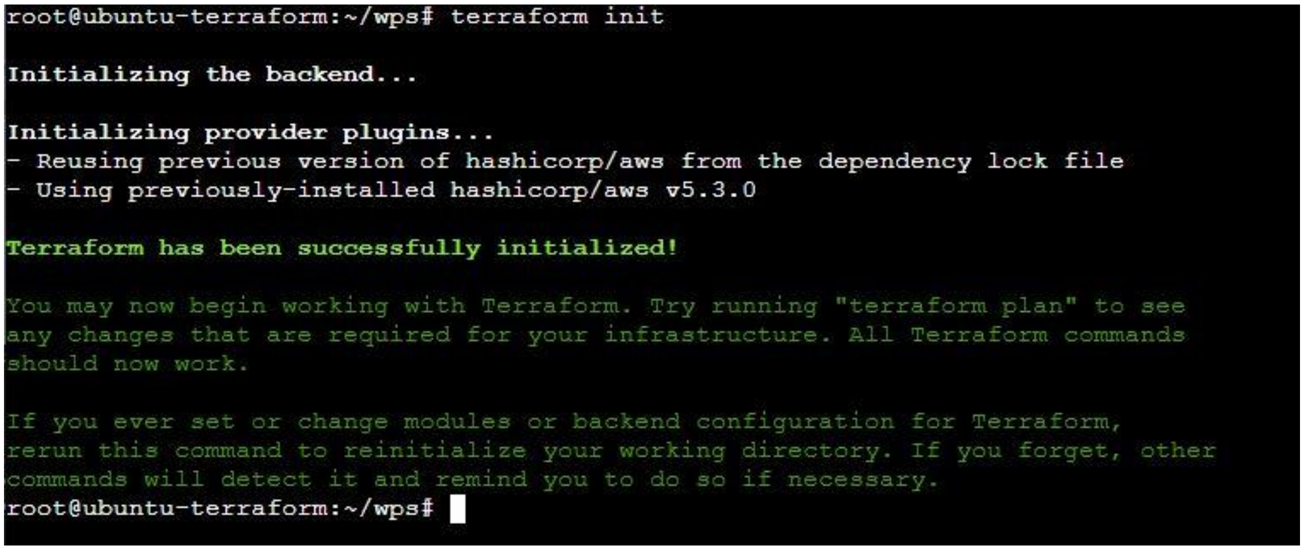
# nano main.tf

```
locals {
  ami_id = "ami-0f8e81a3da6e2510a"
  vpc_id = "vpc-0576e1820ad9d856c"
  ssh_user = "ubuntu"
  key_name = "ansible-engine-pub-key"
  vpc_security_group_ids = "sg-05408ffe087cf8ec9"
}

provider "aws" {
  access_key = "AKIA2CPFWOB2HMQSQ6SS"
  secret_key = "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"
  region = "us-west-1"
}


# provision ec2 instance
resource "aws_instance" "wp-server" {
    ami = local.ami_id
    instance_type = "t2.micro"
    associate_public_ip_address = true
    key_name = "ansible-engine-pub-key"

    tags = {
      Name = "wp-server"
    }

}
```

Steps of EC2 provision on AWS using Terraform

Step 1- Initializing the Terraform module:

# terraform init



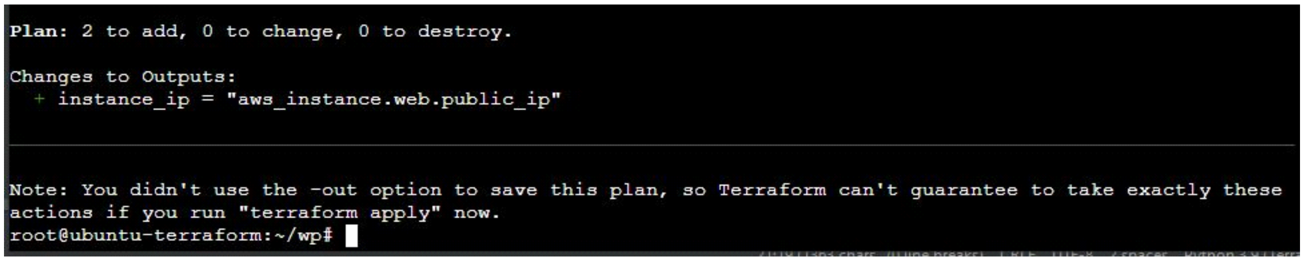
```
root@ubuntu-terraform:~/wps# terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.3.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
root@ubuntu-terraform:~/wps#
```

Step 2 - validating:

# terraform validate

Step 3 -planning:

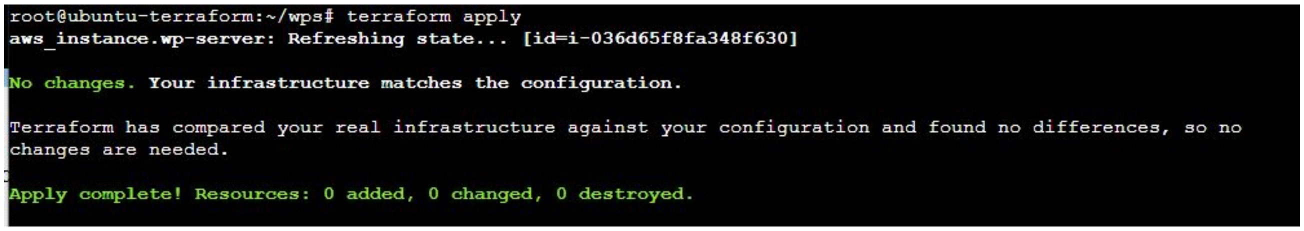
# terraform plan



```
Plan: 2 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + instance_ip = "aws_instance.web.public_ip"

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these
actions if you run "terraform apply" now.
root@ubuntu-terraform:~/wp#
```

Step 4 - applying stage:

# terraform apply



```
root@ubuntu-terraform:~/wps# terraform apply
aws_instance.wp-server: Refreshing state... [id=i-036d65f8fa348f630]

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no differences, so no
changes are needed.

Apply complete! Resources: 0 added, 0 changed, 0 destroyed.
```

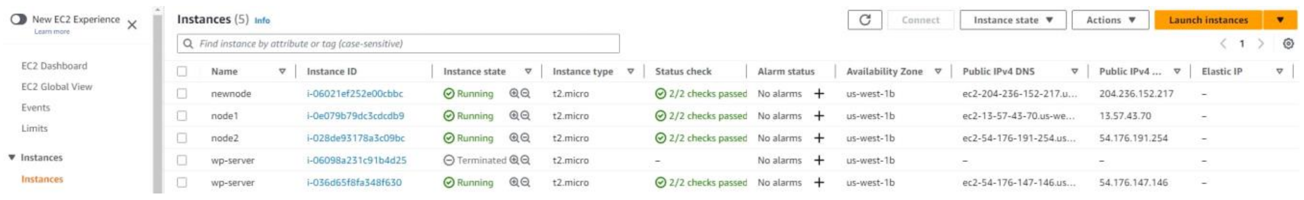
Step 5 - verifying EC2 Instance provision

- log into AWS account -> AWS management console -> EC2 -> Instance
- result: EC2 instance was successful provisioned.

EC2 name: wp-server

EC2 public ip: 54.176.147.146

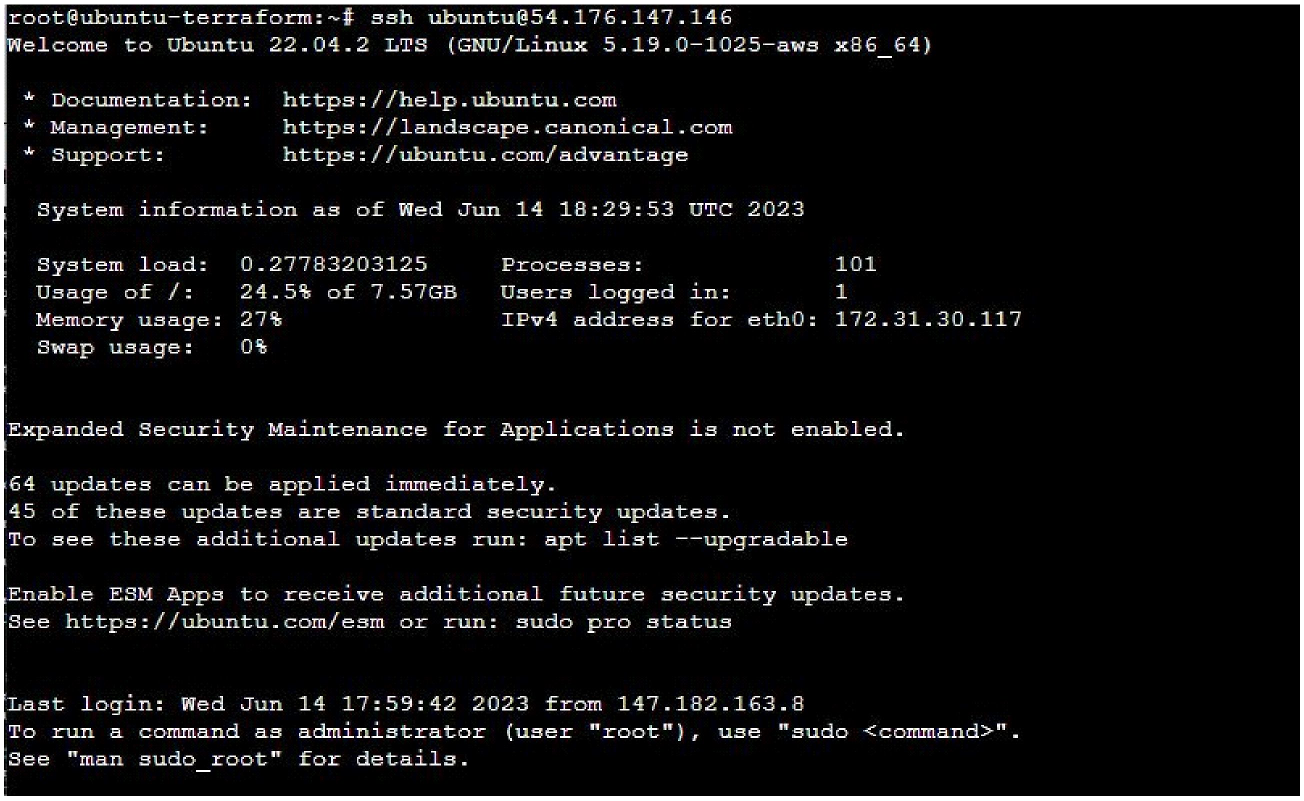
username: ubuntu



Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
newnode	i-06021ef252e00cbbc	Running	t2.micro	2/2 checks passed	No alarms	us-west-1b	ec2-204-236-152-217.u...	204.236.152.217	–
node1	i-0e079b79dc3cdcdb9	Running	t2.micro	2/2 checks passed	No alarms	us-west-1b	ec2-13-57-43-70.us-we...	13.57.43.70	–
node2	i-028de93178a3c09bc	Running	t2.micro	2/2 checks passed	No alarms	us-west-1b	ec2-54-176-191-254.us...	54.176.191.254	–
wp-server	i-06098a231c91b4d25	Terminated	t2.micro	–	No alarms	us-west-1b	–	–	–
wp-server	i-036d65f8fa348f630	Running	t2.micro	2/2 checks passed	No alarms	us-west-1b	ec2-54-176-147-146.us...	54.176.147.146	–

Step 6 - verifying remote connectivity between Ansible control node and AWS EC2 Instance

# ssh ubuntu@54.176.147.146


```
root@ubuntu-terraform:~# ssh ubuntu@54.176.147.146
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.19.0-1025-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Wed Jun 14 18:29:53 UTC 2023

  System load:  0.27783203125     Processes:             101
  Usage of /:   24.5% of 7.57GB   Users logged in:       1
  Memory usage: 27%               IPv4 address for eth0: 172.31.30.117
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

64 updates can be applied immediately.
45 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


Last login: Wed Jun 14 17:59:42 2023 from 147.182.163.8
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
```

Steps of using Ansible to automate WP server

Step 1- Inventory file setup on control node.

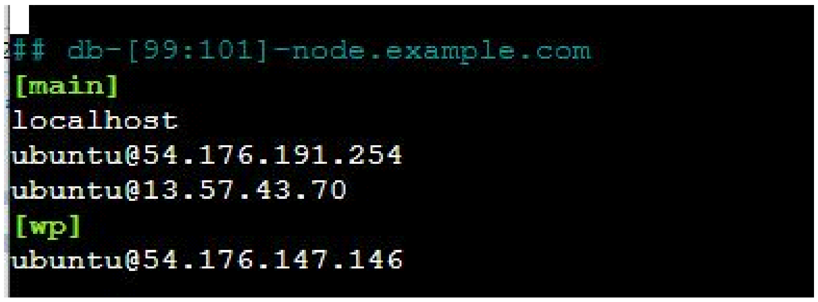
 - opening Ansible inventory file on Ansible control node.

# nano /etc/ansible/hosts

 - then adding the new host under [wp] on Ansible inventory file. If you have more severs to automate at the same time, you can simplely add more hosts under [wp]

[wp]

ubuntu@54.176.147.146



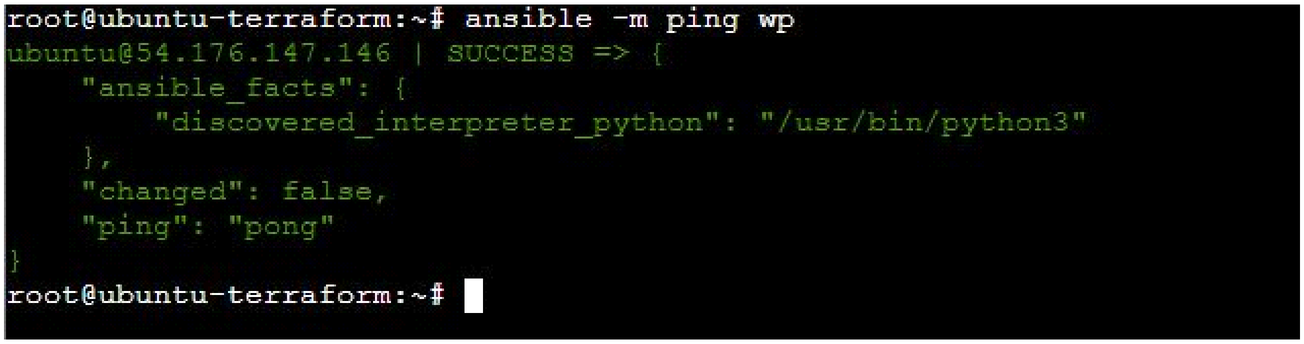
```
## db-[99:101]-node.example.com
[main]
localhost
ubuntu@54.176.191.254
ubuntu@13.57.43.70
[wp]
ubuntu@54.176.147.146
```

Step 2- Connecting control node to host node, verifying Ansible connection use ping module, a successful ping means control node establishes connnectivity with the host node(s) on AWS.

ansible -m ping wp

or

# ansible wp -m ping



```
root@ubuntu-terraform:~# ansible -m ping wp
ubuntu@54.176.147.146 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
root@ubuntu-terraform:~#
```

Step 3 - Creating and running a playbook, which will perform the following actions on host node.

- creating a new folder

# mkdir ansible

# cd ansible

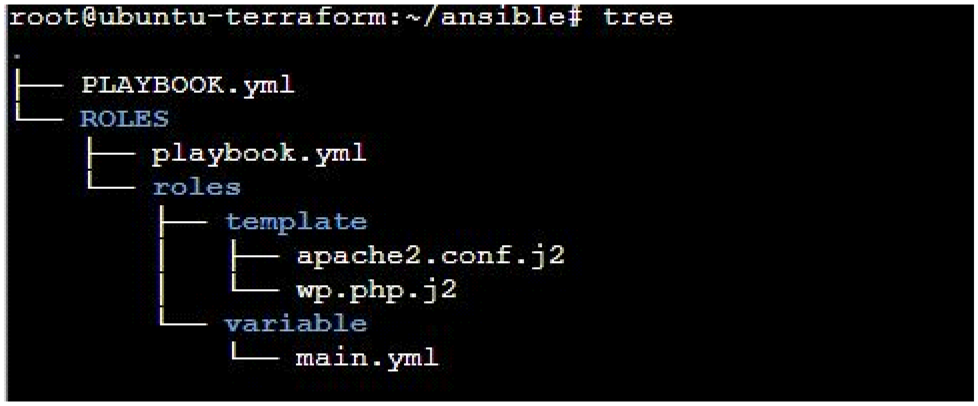
# nano playbook.yml

```
# Automate WORDPRESS SERVER ON LINUX UBUNTU LATEST VERSION

---
- hosts: other
  become: true
```

```
vars_files:
  - variable/main.yml

tasks:

  - name: Install php apache2
    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 dependencies
    apt: name={{ item }} update_cache=yes state=latest
    loop: "{{ php_modules }}"
    tags: [ system ]

# configure apache
  - name:  root user
    file:
      path: "/var/www/{{ http_host }}"
      state: directory
      owner: "user1"
      group: "user1"
      mode: '0755'
    tags: [ apache ]

  - name: set up alternate site to enable multiple sides hosting
    template:
      src: "ROLES/roles/template/apache.conf.j2"
      dest: "/etc/apache2/sites-available/{{ http_conf }}"
    notify: Reload Apache
    tags: [ apache ]

  - name: 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 ]

# configure mysql server
  - 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 ]
```

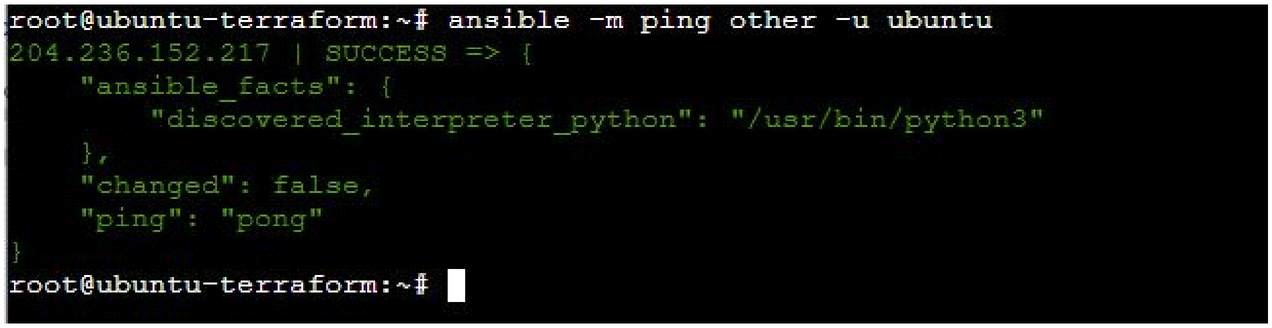
```
- name: remove anonymous user accounts
  mysql_user:
    name: ''
    host_all: yes
    state: absent
    login_user: root
    login_password: "{{ mysql_root_password }}"
  tags: [ mysql ]

- 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 WP
  mysql_db:
    name: "{{ mysql_db }}"
    state: present
    login_user: root
    login_password: "{{ mysql_root_password }}"
  tags: [ mysql ]

- name: create MySQL user for WP
  mysql_user:
    name: "{{ mysql_user }}"
    password: "{{ mysql_password }}"
    priv: "{{ mysql_db }}.*:ALL"
    state: present
    login_user: root
    login_password: "{{ mysql_root_password }}"
  tags: [ mysql ]

# configure firewall -ufw
- name: "ufw - allow HTTP on port {{ http_port }}"
  ufw:
    rule: allow
    port: "{{ http_port }}"
    proto: tcp
  tags: [ system ]

# configure WP
- name: Download and unpack latest WP
    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: ownership
        file:
          path: "/var/www/{{ http_host }}"
          state: directory
          recurse: yes
          owner: user1
          group: user1
        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: setup wp-config
        template:
          src: "ROLES/roles/template/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
```

Using roles as below structure



```
root@ubuntu-terraform:~/ansible# tree
.
├── PLAYBOOK.yml
└── ROLES
    ├── playbook.yml
    └── roles
        ├── template
        │   ├── apache2.conf.j2
        │   └── wp.php.j2
        └── variable
            └── main.yml
```

Establishing Ansible control node connection with ansible host 204.236.152.217



```
root@ubuntu-terraform:~# ansible -m ping other -u ubuntu
204.236.152.217 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
root@ubuntu-terraform:~#
```

Creating template files

# cd template

# nano apache2.conf.j2

```
<AlternateHost *:{{ http_port }}>
   ServerAdmin webadmin@localhost
   ServerName {{ http_host }}
   ServerAlias www.{{ http_host }}
   DocumentRoot /var/www/{{ http_host }}/wordpress
   ErrorLog ${APACHE_LOG_DIR}/error.log
   CustomLog ${APACHE_LOG_DIR}/access.log combined

   <Directory /var/www/{{ http_host }}>
        Options -Indexes
        AllowOverride All
   </Directory>

   <IfModule mod_dir.c>
      DirectoryIndex index.php index.html index.cgi index.pl  index.xhtml index.htm
   </IfModule>

</AlternateHost>
```

Using ctl + x, y, enter to save and close the file

# nano wp.php.j2

```
<?php
/**
 * The base config for WP server including:
 *
 * * MySQL settings
 * * Secret keys
 * * Database table prefix
 * * ABSPATH
 *
 * @link https://codex.wordpress.org/Editing_wp-config.php
 *
 * @package WordPress
 */
```

```
// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define( 'DB_NAME', '{{ mysql_db }}' );

/** MySQL database username */
define( 'DB_USER', '{{ mysql_user }}' );

/** MySQL database password */
define( 'DB_PASSWORD', '{{ mysql_password }}' );

/** MySQL hostname */
define( 'DB_HOST', 'localhost' );

/** Database Charset to use in creating database tables. */
define( 'DB_CHARSET', 'utf8' );

/** The Database Collate type. Don't change this if in doubt. */
define( 'DB_COLLATE', '' );

/** Filesystem access **/
define('FS_METHOD', 'direct');

/**#@+
```

Using ctl + x, y, enter to save and close the file

Exiting from current folder

```
# cd ..
```

Creating a variable file

```
# cd variable
```

```
# nano mail.yml
```

```
---
### environment settings ###
php_modules: [ 'php-curl', 'php-gd', 'php-mbstring', 'php-xml', 'php-xmlrpc', 'php-soap', 'php-intl', 'php-zip>

### mysql ###
mysql_root_password: "password"
mysql_db: "wordpress"
mysql_user: "user1"
mysql_password: "password"

### http ####
http_host: "204.236.152.217"
http_conf: "204.236.152.217.conf"
http_port: "80"
```

Using ctl + x, y, enter to save and close the file

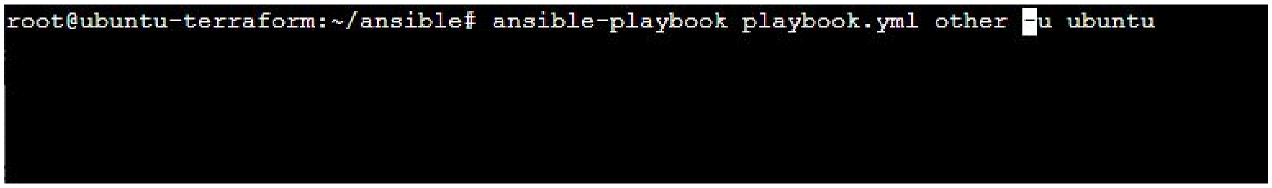
# cd /root/ansible

# mkdir ROLES

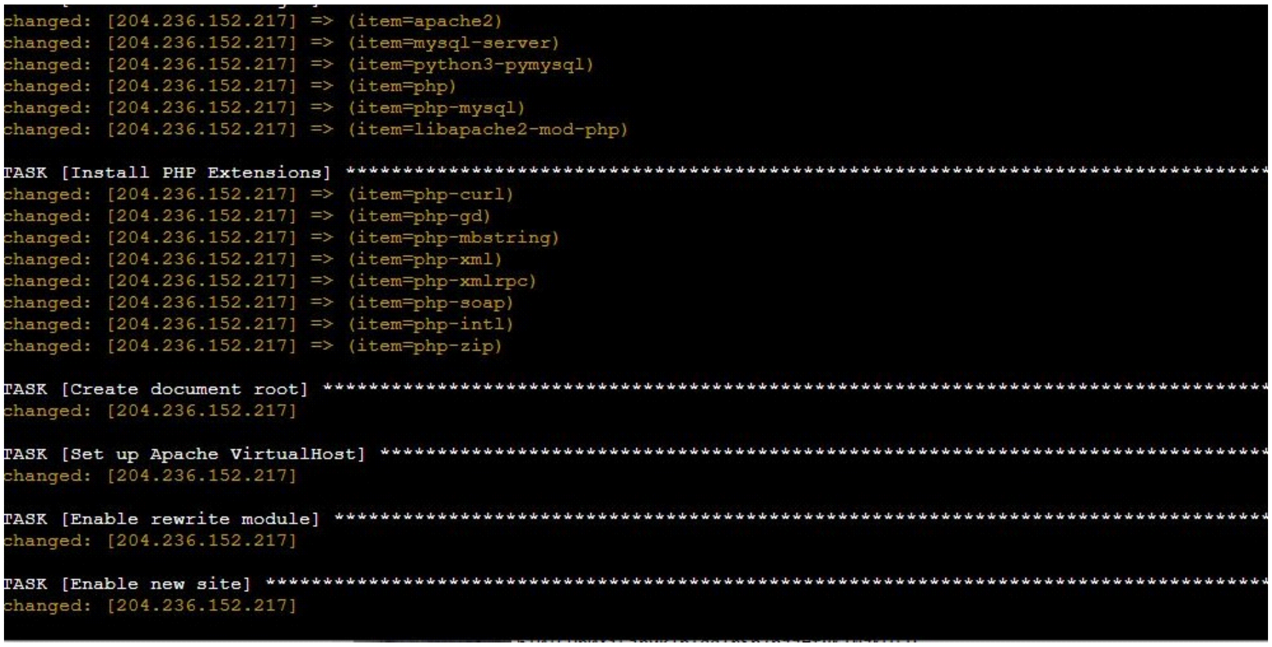
# mv playbook.yml roles/ ROLES/

Executing ansible playbook with this command

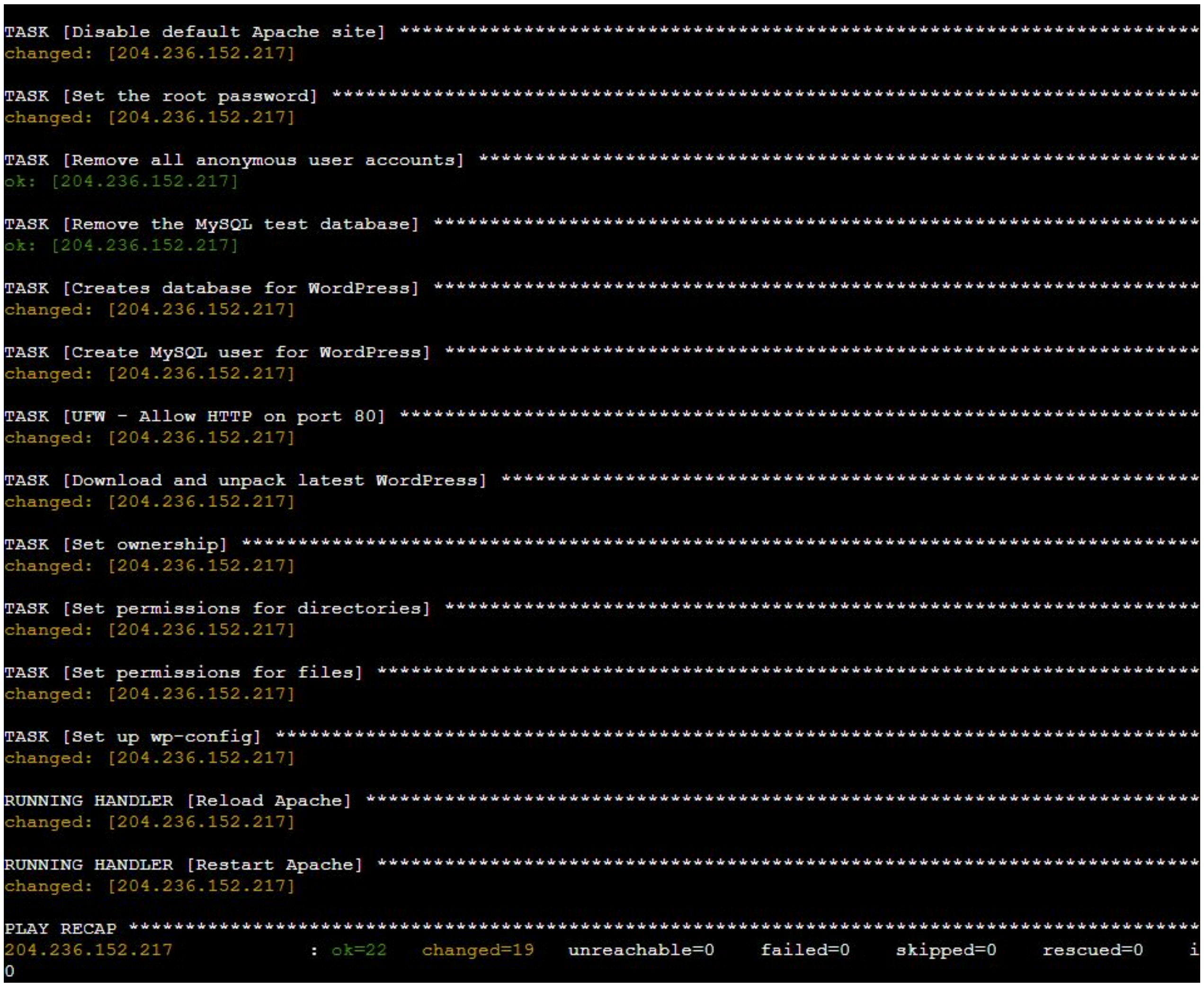
# ansible-playbook playbook.yml other -u ubuntu



```
root@ubuntu-terraform:~/ansible# ansible-playbook playbook.yml other -u ubuntu
```



```
changed: [204.236.152.217] => (item=apache2)
changed: [204.236.152.217] => (item=mysql-server)
changed: [204.236.152.217] => (item=python3-pymysql)
changed: [204.236.152.217] => (item=php)
changed: [204.236.152.217] => (item=php-mysql)
changed: [204.236.152.217] => (item=libapache2-mod-php)

TASK [Install PHP Extensions] ******************************************************************************
changed: [204.236.152.217] => (item=php-curl)
changed: [204.236.152.217] => (item=php-gd)
changed: [204.236.152.217] => (item=php-mbstring)
changed: [204.236.152.217] => (item=php-xml)
changed: [204.236.152.217] => (item=php-xmlrpc)
changed: [204.236.152.217] => (item=php-soap)
changed: [204.236.152.217] => (item=php-intl)
changed: [204.236.152.217] => (item=php-zip)

TASK [Create document root] ********************************************************************************
changed: [204.236.152.217]

TASK [Set up Apache VirtualHost] ***************************************************************************
changed: [204.236.152.217]

TASK [Enable rewrite module] *******************************************************************************
changed: [204.236.152.217]

TASK [Enable new site] *************************************************************************************
changed: [204.236.152.217]
```

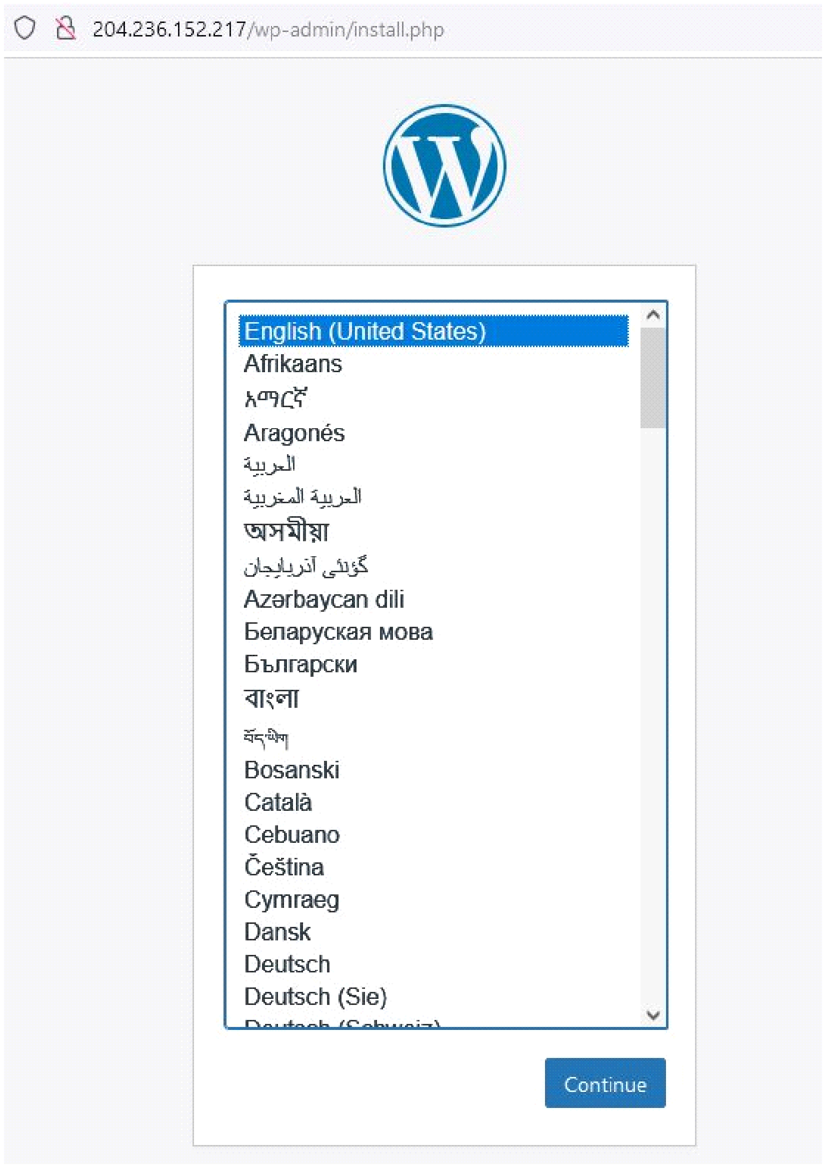


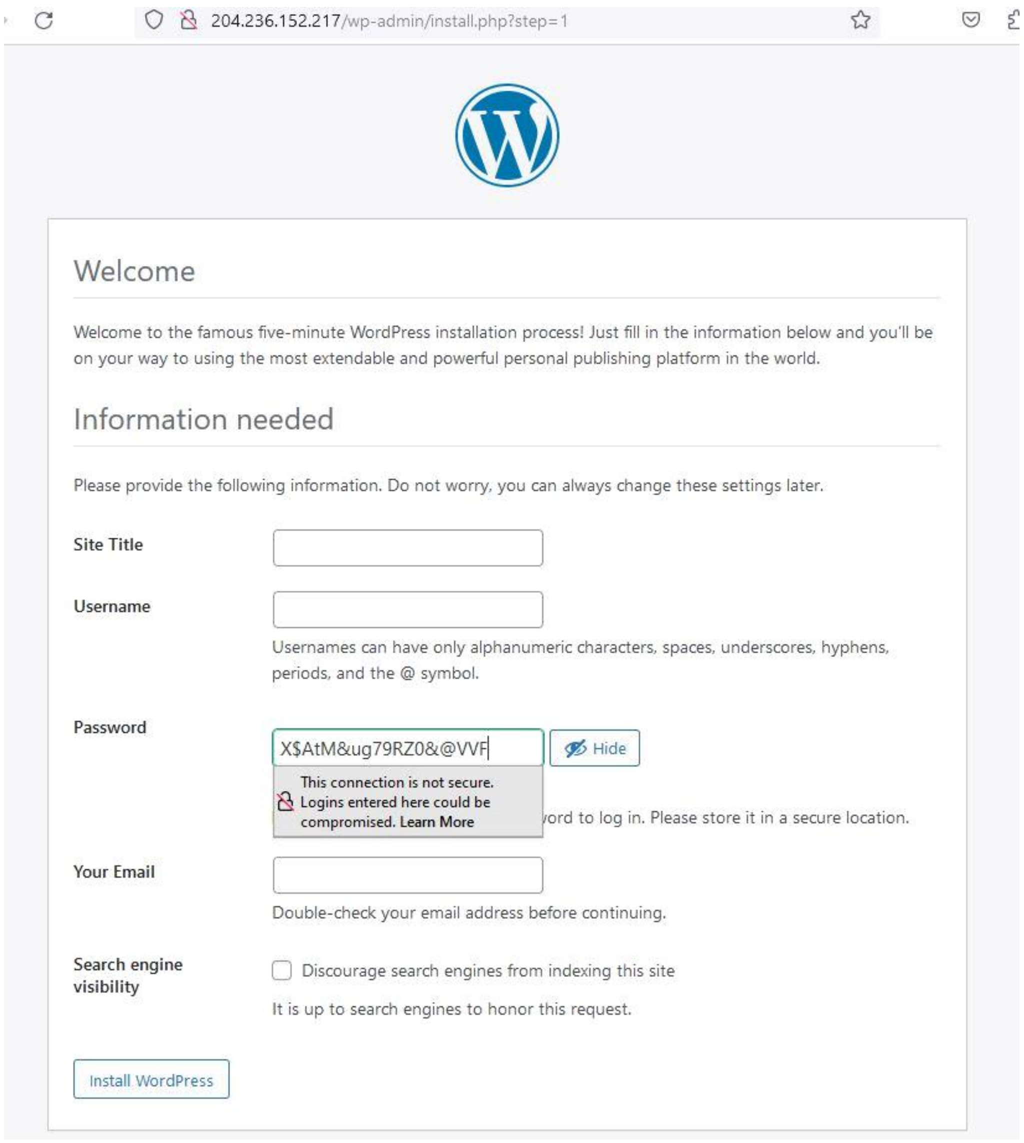
```
TASK [Disable default Apache site] *********************************************************************
changed: [204.236.152.217]

TASK [Set the root password] ***************************************************************************
changed: [204.236.152.217]

TASK [Remove all anonymous user accounts] **************************************************************
ok: [204.236.152.217]

TASK [Remove the MySQL test database] ******************************************************************
ok: [204.236.152.217]

TASK [Creates database for WordPress] ******************************************************************
changed: [204.236.152.217]

TASK [Create MySQL user for WordPress] *****************************************************************
changed: [204.236.152.217]

TASK [UFW - Allow HTTP on port 80] *********************************************************************
changed: [204.236.152.217]

TASK [Download and unpack latest WordPress] ************************************************************
changed: [204.236.152.217]

TASK [Set ownership] ***********************************************************************************
changed: [204.236.152.217]

TASK [Set permissions for directories] *****************************************************************
changed: [204.236.152.217]

TASK [Set permissions for files] ***********************************************************************
changed: [204.236.152.217]

TASK [Set up wp-config] ********************************************************************************
changed: [204.236.152.217]

RUNNING HANDLER [Reload Apache] ************************************************************************
changed: [204.236.152.217]

RUNNING HANDLER [Restart Apache] ***********************************************************************
changed: [204.236.152.217]

PLAY RECAP *********************************************************************************************
204.236.152.217            : ok=22   changed=19   unreachable=0    failed=0    skipped=0    rescued=0    i
0
```

Once the playbook is successfully executed, browsing the server page using server's public ip address.

From here you can install and edit your WordPress as desired, thus, as demonstrated in this project, the WP server deployment using configureation management succeed.

http:// 204.236.152.217


204.236.152.217/wp-admin/install.php



204.236.152.217/wp-admin/install.php?step=1

Welcome

Welcome to the famous five-minute WordPress installation process! Just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform in the world.

Information needed

Please provide the following information. Do not worry, you can always change these settings later.

Site Title	
Username	Usernames can have only alphanumeric characters, spaces, underscores, hyphens, periods, and the @ symbol.
Password	X$AtM&ug79RZ0&@VVF This connection is not secure. Logins entered here could be compromised. Learn More …ord to log in. Please store it in a secure location.
Your Email	Double-check your email address before continuing.
Search engine visibility	☐ Discourage search engines from indexing this site It is up to search engines to honor this request.

Conclusion

This project demonstrates how to use configuration management tools such as Terraform and Ansible to deploy the WordPress server in Linux Ubuntu system in a simple process. It is easy to operate and deploy.

The use of Ansible auto configuration allows you to deploy one to hundreds of servers with a single control node.