**STEP-1: Establish configuration management master connectivity with WordPress server**

1. Do these in the configuration management master server as well as node, i.e., wordpress server:

sudo apt update (updates the machine)

sudo su (switch to root user)

sudo adduser ansible (create an ansible user)

enter password and other credentials for ansible user

vi /etc/sudoers

(inside the sudoers file in line 45 under user privilege specification add this line to permit passwordless super user rights to ansible user)

ansible ALL=(ALL:ALL) NOPASSWD: ALL

vi /etc/ssh/sshd\_config

(inside the sshd\_config file in line 57 change PasswordAuthentication from no to yes)

systemctl restart sshd (to restart sshd)

1. Only in the master server:

switch to ansible user and generate keys for ssh login to wordpress server

su – ansible (switches to ansible user)

ssh-keygen (generates public key for ansible user)

cat .ssh/id\_rsa.pub (copy this key into authorized\_keys folder inside the .ssh folder in wordpress server)

1. inside the wordpress server:

su – ansible (switch to ansible user)

ssh-keygen (to create a .ssh folder in ansible home directory)

vi .ssh/authorized\_keys (paste the key generated from master server to wordpress server in this folder)

**STEP-2: Validate connectivity from master to slave machine**

Inside the master server:

(check the connectivity from master to wordpress server)

ssh <private-ip-of-wp-server> (login to the wordpress server)

exit

1. Install ansible and add ansible repositories:

sudo apt-add-repository ppa:ansible/ansible (installs ansible repositories)

sudo apt update (updates the machine)

sudo apt install ansible -y (installs ansible)

sudo vi /etc/ansible/hosts (inside the hosts file add the private ip of the wordpress server under wpserver host-group)

[wpserver]

<private-ip-of-wp-server>

(exit the file by using :wq)

1. Check the ping from master to slave machine using ansible adhoc command:

ansible -m ping wpserver

Ping successful from master to wordpress server node.

**STEP-3: Prepare IaC scripts to install WordPress and its dependent components**

1. Create a wordpress working directory:

mkdir wordpress (creates a working directory named wordpress)

cd wordpress (enter into the wordpress directory)

Create and write a playbook inside this folder to install WordPress and its dependent components:

1. Create a playbook to add the following script to install Wordpress and it dependent components and configure each of the components:

vi playbook.yml (creates a playbook named playbook.yml)

#####inside the playbook:

---

- hosts: wpserver

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 dependent 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

(save and quit the file using :wq)

PLAYBOOK EXPLANATION:

* The script in the playbook performs installation of prerequisites first and the dependent packages required for wordpress site.
* Performs installation of Php Extensions.
* Performs Apache Configuration.
* Performs MySQL Configuration.
* Performs Firewall configuration to allow connection to Wordpress site.
* Performs WordPress Configuration.
* Executes Handlers to reload and restart apache2.

1. Create a directory to add jinja templates for standard use whenever a new requirement or client comes in:

mkdir files (creates a directory named files inside the wordpress directory)

cd files (enter into files directory)

1. vi apache.conf.j2 (creates an apache jinja template file)

#####inside the apache.conf.j2 file:

<VirtualHost \*:{{ http\_port }}>

ServerAdmin webmaster@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>

</VirtualHost>

(save and quit the file using :wq)

1. vi wp-config.php.j2 (creates an wordpress jinja template file)

#####inside the wp-config.php.j2 file:

<?php

/\*\*

\* The base configuration for WordPress

\*

\* The wp-config.php creation script uses this file during the

\* installation. You don't have to use the web site, you can

\* copy this file to "wp-config.php" and fill in the values.

\*

\* This file contains the following configurations:

\*

\* \* 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');

/\*\*#@+

\* Authentication Unique Keys and Salts.

\*

\* Change these to different unique phrases!

\* You can generate these using the {@link https://api.wordpress.org/secret-key/1.1/salt/ WordPress.org secret-key service}

\* You can change these at any point in time to invalidate all existing cookies. This will force all users to have to log in again.

\*

\* @since 2.6.0

\*/

define( 'AUTH\_KEY', '{{ lookup('password', '/dev/null chars=ascii\_letters length=64') }}' );

define( 'SECURE\_AUTH\_KEY', '{{ lookup('password', '/dev/null chars=ascii\_letters length=64') }}' );

define( 'LOGGED\_IN\_KEY', '{{ lookup('password', '/dev/null chars=ascii\_letters length=64') }}' );

define( 'NONCE\_KEY', '{{ lookup('password', '/dev/null chars=ascii\_letters length=64') }}' );

define( 'AUTH\_SALT', '{{ lookup('password', '/dev/null chars=ascii\_letters length=64') }}' );

define( 'SECURE\_AUTH\_SALT', '{{ lookup('password', '/dev/null chars=ascii\_letters length=64') }}' );

define( 'LOGGED\_IN\_SALT', '{{ lookup('password', '/dev/null chars=ascii\_letters length=64') }}' );

define( 'NONCE\_SALT', '{{ lookup('password', '/dev/null chars=ascii\_letters length=64') }}' );

/\*\*#@-\*/

/\*\*

\* WordPress Database Table prefix.

\*

\* You can have multiple installations in one database if you give each

\* a unique prefix. Only numbers, letters, and underscores please!

\*/

$table\_prefix = 'wp\_';

/\*\*

\* For developers: WordPress debugging mode.

\*

\* Change this to true to enable the display of notices during development.

\* It is strongly recommended that plugin and theme developers use WP\_DEBUG

\* in their development environments.

\*

\* For information on other constants that can be used for debugging,

\* visit the Codex.

\*

\* @link https://codex.wordpress.org/Debugging\_in\_WordPress

\*/

define( 'WP\_DEBUG', false );

/\* That's all, stop editing! Happy publishing. \*/

/\*\* Absolute path to the WordPress directory. \*/

if ( ! defined( 'ABSPATH' ) ) {

define( 'ABSPATH', dirname( \_\_FILE\_\_ ) . '/' );

}

/\*\* Sets up WordPress vars and included files. \*/

require\_once( ABSPATH . 'wp-settings.php' );

(save and quit the file using :wq)

cd .. (to enter into the wordpress directory)

1. Create a directory named vars in the wordpress working directory to store all the variables for the playbook:

mkdir vars (creates a directory named vars)

cd vars (enter into the vars directory)

1. vi default.yml (creates a file named default.yml)

#####inside the default.yml file:

---

#System Settings

php\_modules: [ 'php-curl', 'php-gd', 'php-mbstring', 'php-xml', 'php-xmlrpc', 'php-soap', 'php-intl', 'php-zip' ]

#MySQL Settings

mysql\_root\_password: "root\_password"

mysql\_db: "wordpress"

mysql\_user: "ansible"

mysql\_password: "password"

#HTTP Settings

http\_host: "172.31.43.249"

http\_conf: "wp-site.conf"

http\_port: "80"

(save and quit the file using :wq)

cd .. (to enter into the wordpress directory)

Change the template files and variable files as per the requirement of the client.

**STEP-4: Execute scripts to perform installation of complete WordPress environment**

1. ansible-playbook playbook.yml (to run the created playbook named playbook.yml)

**STEP-5: Validate installation using the public IP of VM by accessing WordPress application**

1. Open web browser in your machine and type the following in the address bar:

http://<public-ip-of-wp-server>:80

Wordpress language selection and login page is opened: enter your credentials and click on install wordpress.

Now login to your account with your credentials and you will reach the dashboard of the wordpress site.