

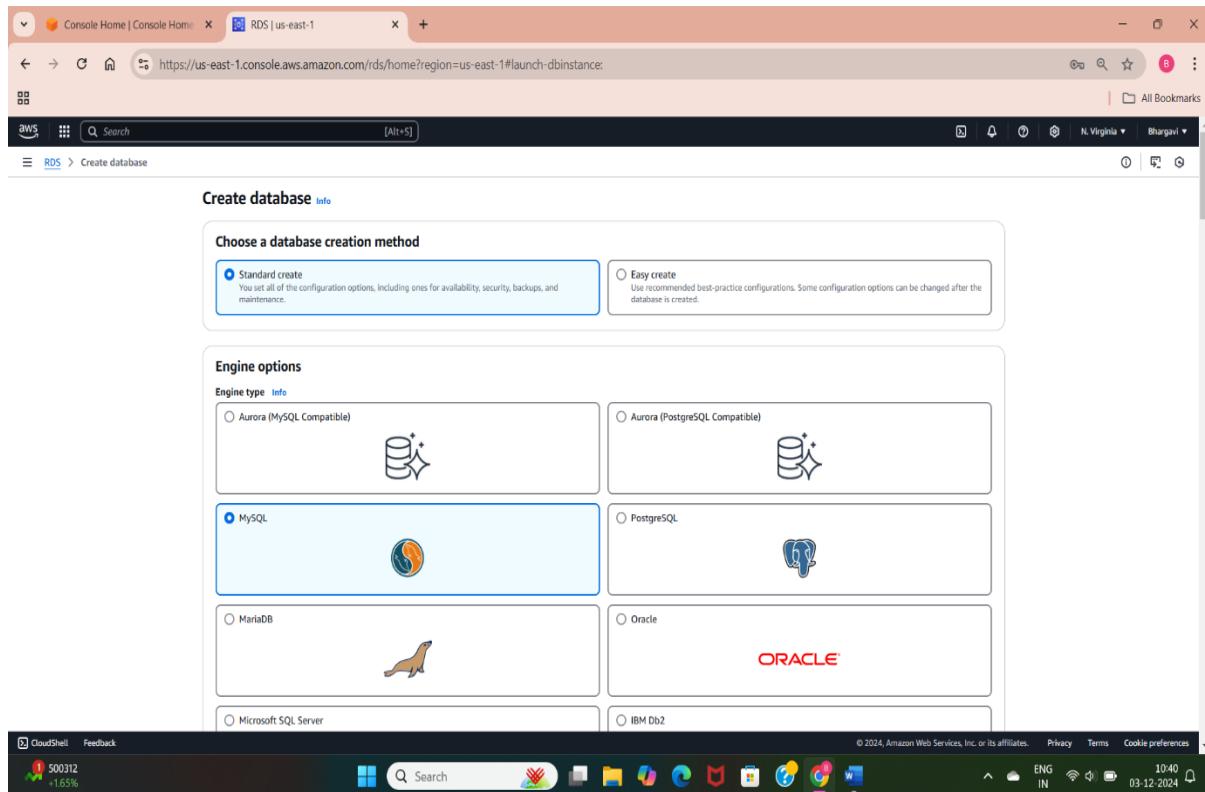
PROJECT-2

DEPLOY WORDPRESS USING DIFFERENT METHODS

METHOD-1: CREATE MYSQL DATABASE BY USING RDS:

RDS is a fully managed service which means just we need to focus on storing and managing data not on creating and maintaining , aws will manage these tasks.

- ❖ Now first go to your aws account and login with credentials. After go to RDS service and open that service.
- ❖ Now create a mysql database by using RDS service for that go into the RDS services and click on create database.



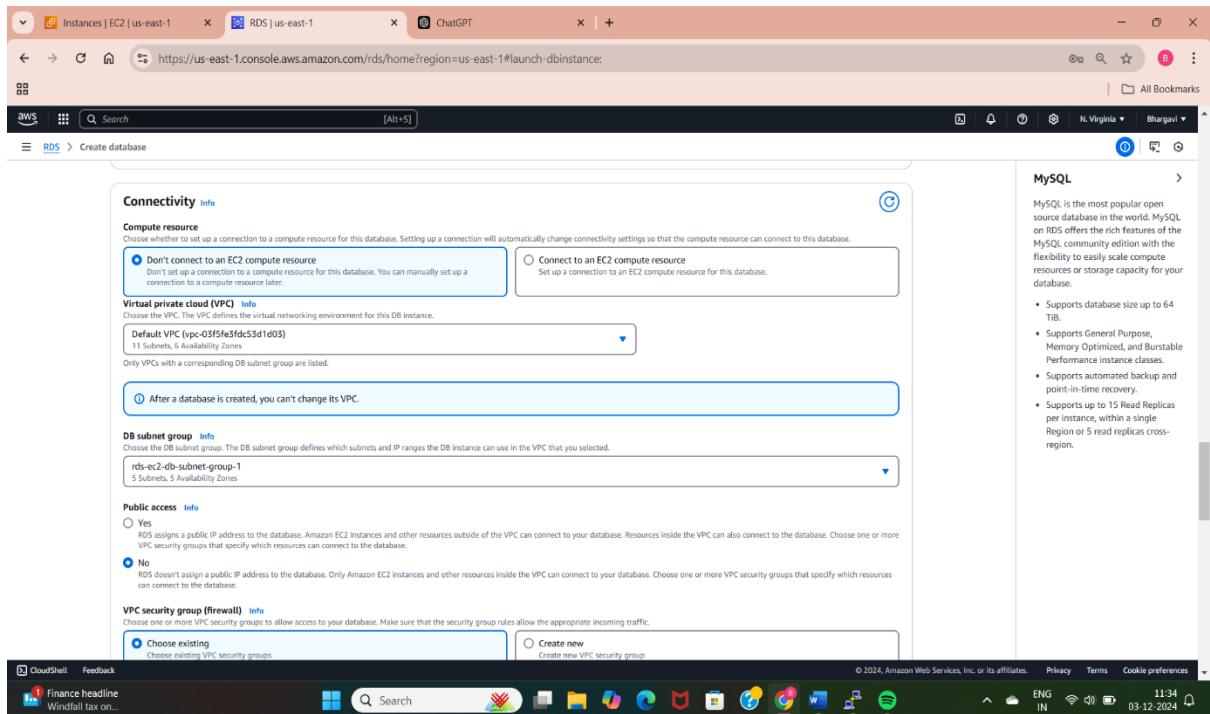
- ❖ Now select the database creation method I select here standard create method because by selecting easy create method then it disables the free tier template.
- ❖ Now select the database engine with version but I select the mysql database engine because it's a huge usage database in real time organizations.

The screenshot shows the 'Create database' wizard on the AWS RDS console. In the 'Edition' section, 'MySQL Community' is selected. Under 'Engine version', 'MySQL 8.0.39' is chosen. In the 'Templates' section, the 'Free tier' option is selected. The 'Availability and durability' section is collapsed.

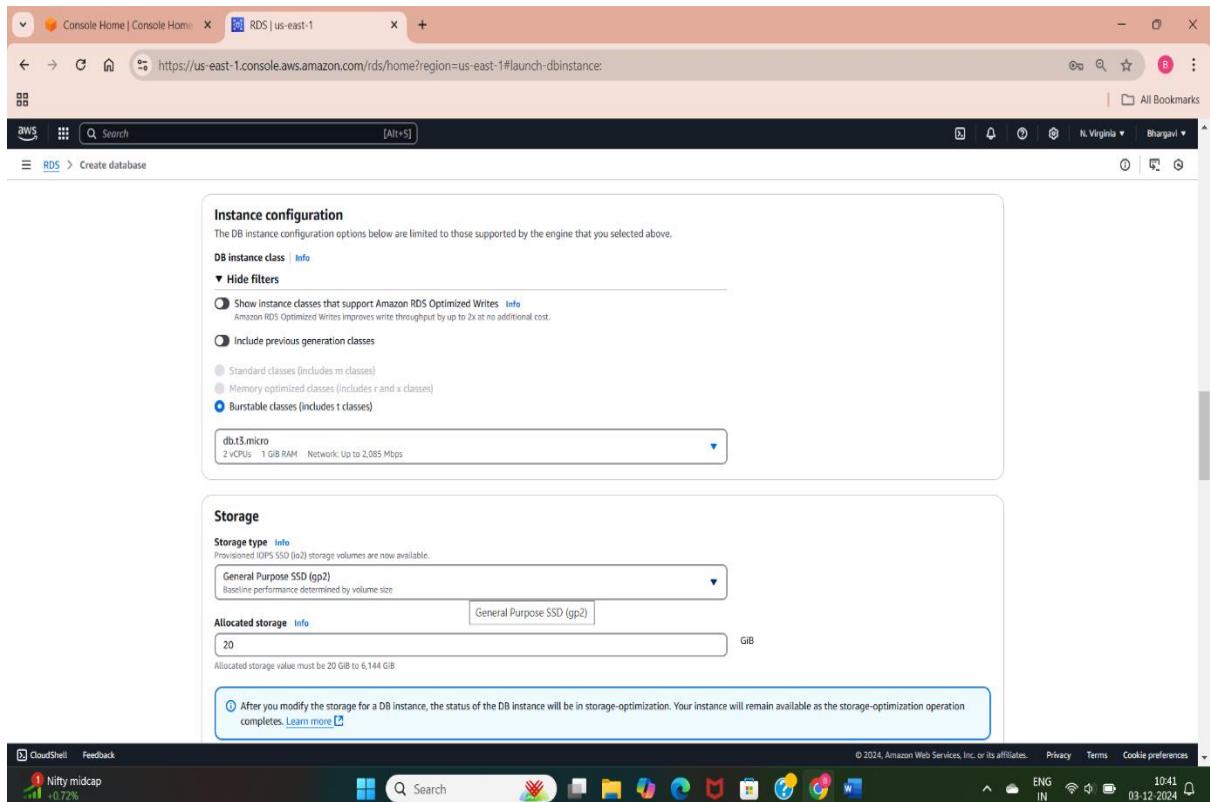
- ❖ I have selected the version “8.0.39” for MYSQL engine
- ❖ Now select the Template as free tier and by selecting this free tier it disable the availability zone selecting option.

The screenshot shows the 'Create database' wizard on the AWS RDS console. In the 'Availability and durability' section, the 'Deployment options' dropdown is expanded, showing 'Multi-AZ DB Cluster', 'Multi-AZ DB instance (not supported for Multi-AZ DB cluster snapshot)', and 'Single DB instance (not supported for Multi-AZ DB cluster snapshot)'. The 'Single DB instance' option is selected. The 'Settings' and 'Credentials Settings' sections are visible below.

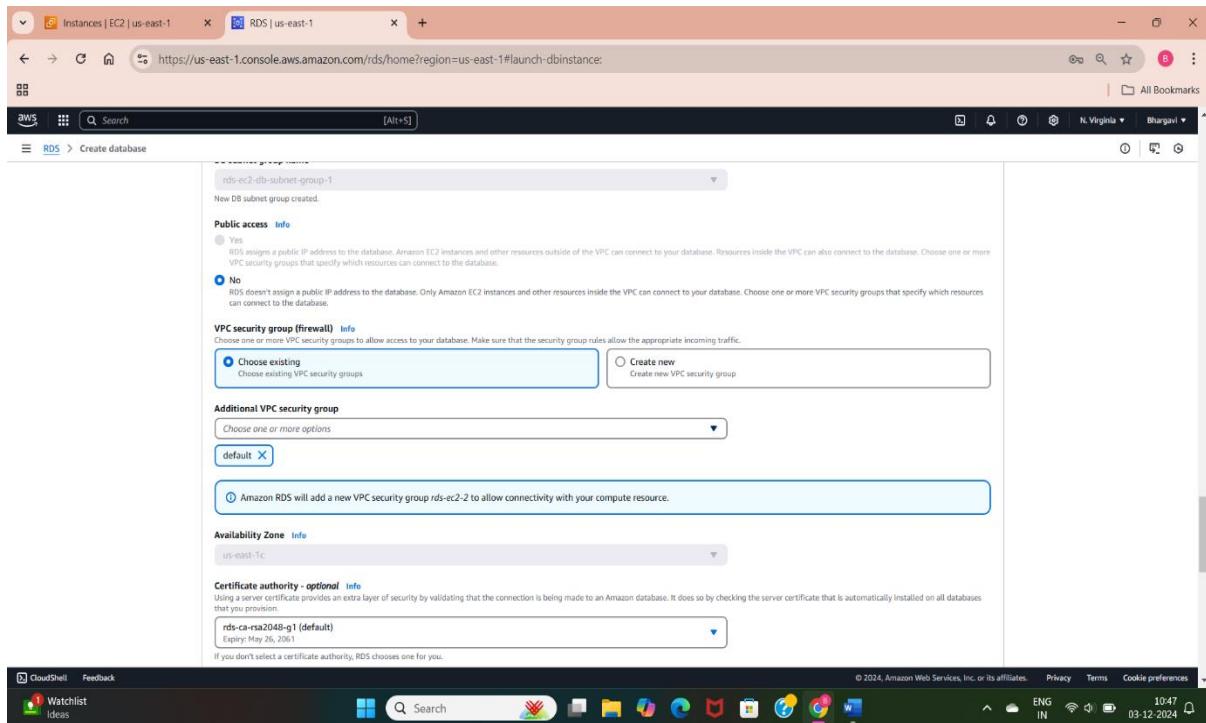
- ❖ Now give the some name to your database and give username and passwords as credentials for your database access.



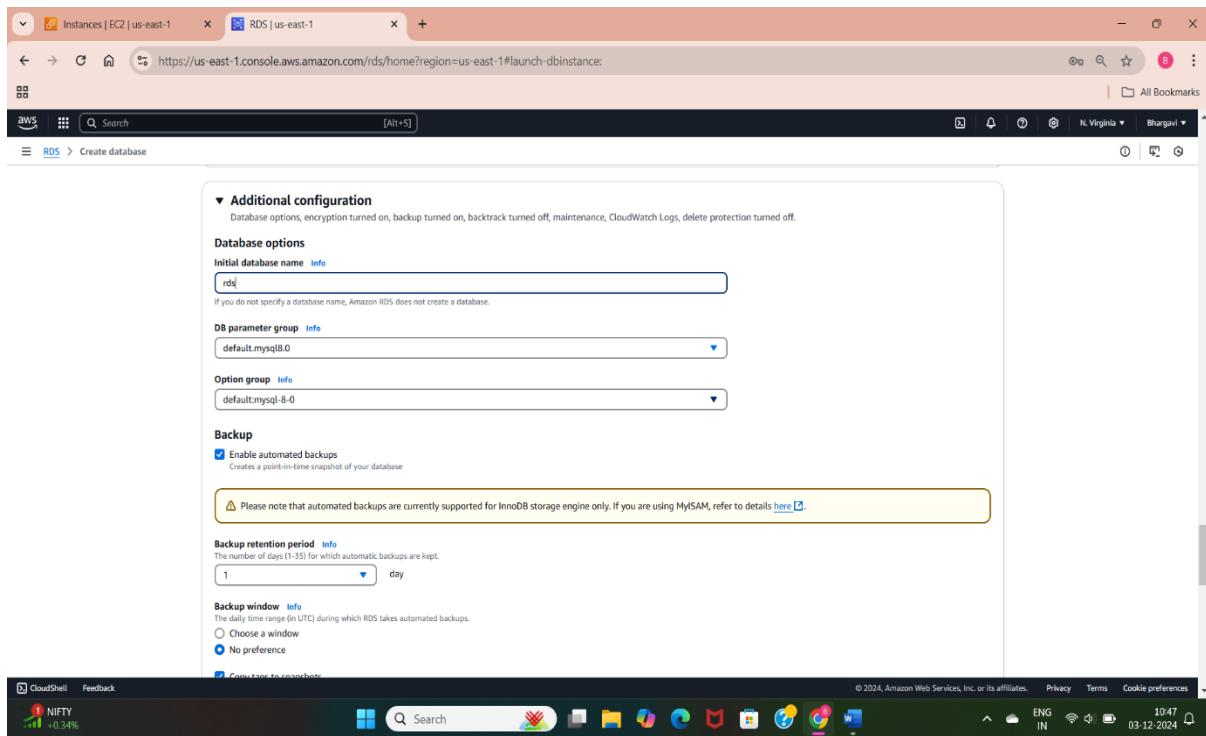
- ❖ I have selected the option with the Don't connect to an EC2 compute resource as connectivity, selected a default VPC for internet access.



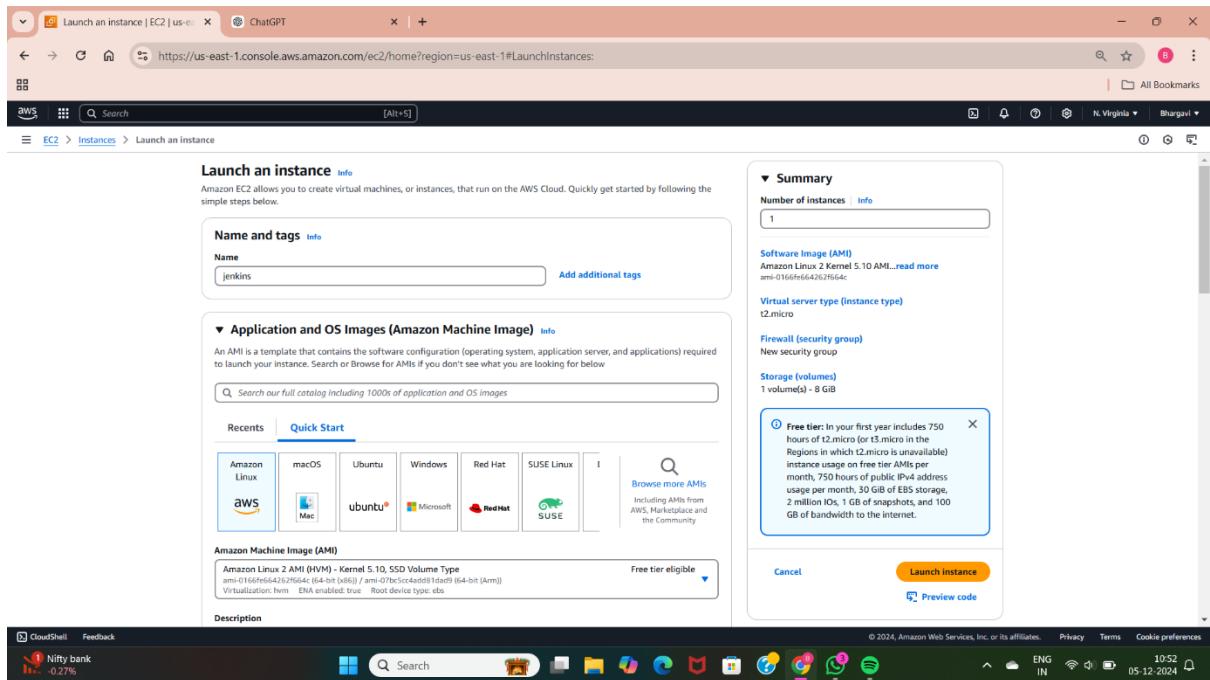
- ❖ Now select the storage type as General Purpose SSD(gp2) and enter the storage values as maximum(1000GB) and minimum(20GB).



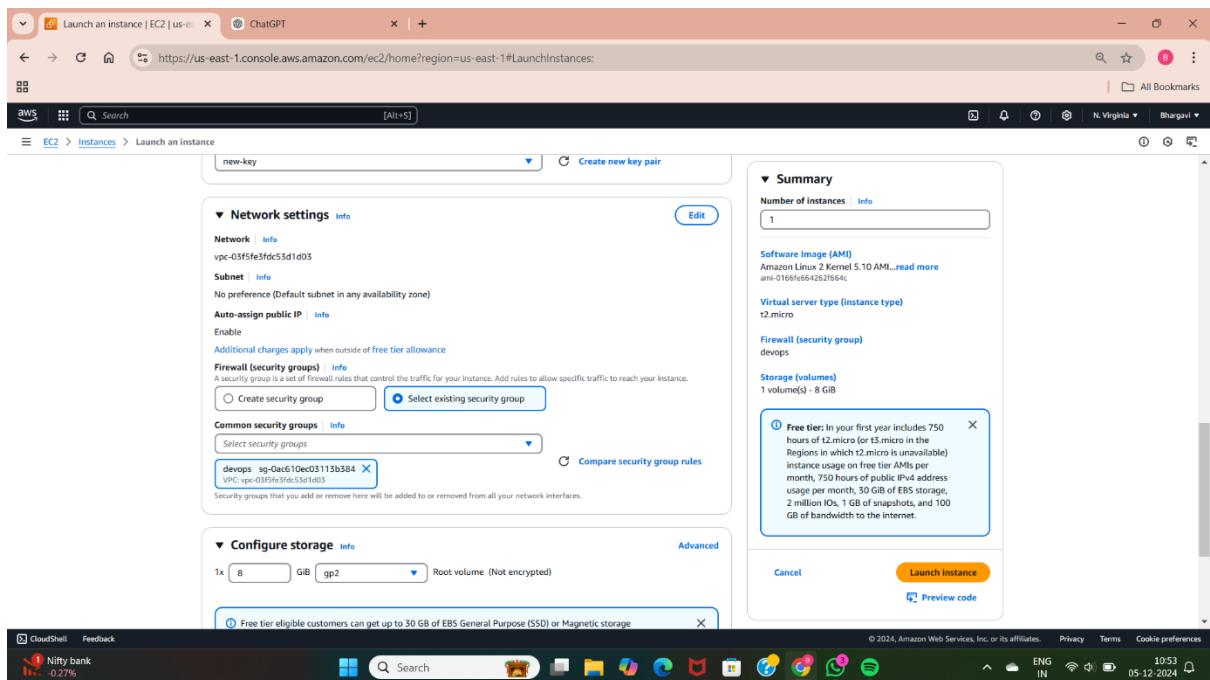
- ❖ Now select your created VPC or select default VPC and it automatically select the database subnet group.



- ❖ Now give the name of the database which you give at the stage of DB instance identifier enter same name here.
- ❖ Now click on create database button and it will create the MYSQL database.



- ❖ Now create a EC2 instance by allowing the specific inbound rules with port numbers 22 for SSH, 80 for HTTP, and 3306 for MYSQL.



- ❖ Now create the EC2 instance by selecting EC2 services and launch the instance by selecting Amazon linux-2 version and giving the security group with SSH(22) and HTTP(80).
- ❖ Now connect the virtual server through the Git bash as shown in below.
- ❖ Now update the linux version by using command as <sudo yum -y update> and install the mysql by using the command as <sudo yum -y install mysql>.

The screenshot shows the AWS RDS console for the 'database1' database. The 'Connectivity & security' tab is active. A message 'Endpoint copied' is displayed. The 'Networking' section shows the VPC (vpc-03f5fe3fdc53d1d03) and subnet group (default-vpc-03f5fe3fdc53d1d03). The 'Security' section shows the VPC security group (default sg-0b8cd2f39b3d68e90) and certificate authority (rds-ca-rsa2048-g1).

- ❖ Now to allow certain traffic from EC2 instance into RDS database for that go to RDS services and select your created database.

The screenshot shows the AWS EC2 console for the security group 'sg-0b8cd2f39b3d68e90 - default'. The 'Inbound rules' tab is selected, showing three entries:

Name	Security group rule ID	IP version	Type	Protocol	Port range	Source	Description
sgr-004e005ea5700...	sg-0b8cd2f39b3d68e90	IPv4	HTTP	TCP	80	0.0.0.0/0	-
sgr-0e1ddcd62a254...		IPv4	SSH	TCP	22	0.0.0.0/0	-
sgr-06c497944b2f2...		IPv4	MySQL/Aurora	TCP	3306	0.0.0.0/0	-

- ❖ Now go inside the created database and go to the security under this option there is a security group id click on that.
- ❖ Now go to inbound rules and click on the edit inbound rules.
- ❖ Now go to source option click on dropdown option select the EC2 instance security group id and click save rules.

```

root@ip-172-31-28-11:~#
=====
Package           Arch      Version          Repository        Size
=====
Installing:
mariadb          x86_64   1:5.5.68-1.amzn2.0.1   amzn2-core       0.0 M
=====
Transaction Summary
Install 1 Package
=====
Total download size: 0.0 M
Installed size: 49 M
Downloading packages:
mariadb-5.5.68-1.amzn2.0.1.x86_64.rpm
Running transaction check
Running transaction test
transaction test succeeded
Running transaction
  Installing : 1:mariadb-5.5.68-1.amzn2.0.1.x86_64
  Verifying  : 1:mariadb-5.5.68-1.amzn2.0.1.x86_64
=====
Installed:
  mariadb.x86_64 1:5.5.68-1.amzn2.0.1
=====
Complete!
[root@ip-172-31-28-11 ~]# mysql --version
mysql Ver 15.1 Distrib 5.5.68-MariaDB, for Linux (x86_64) using readline 5.1
[root@ip-172-31-28-11 ~]# export MYSQL_HOST=databasel.clau2u6sigra.us-east-1.rds.amazonaws.com
[root@ip-172-31-28-11 ~]# mysql -h databasel.clau2u6sigra.us-east-1.rds.amazonaws.com -u admin -p
Enter password:
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MySQL connection id is 31
Server version: 8.0.39 Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]> CREATE USER 'wordpress' IDENTIFIED BY 'wordpress-pass';
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''wordpress-pass'' at line 1
MySQL [(none)]> GRANT ALL PRIVILEGES ON wordpress.* TO 'wordpress';
ERROR 1410 (42000): You are not allowed to create a user with GRANT
MySQL [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.01 sec)

MySQL [(none)]> EXIT
Bye
[root@ip-172-31-28-11 ~]#

```

26°C Mostly cloudy ENG IN 12:41 03-12-2024

- ❖ Now access the mysql database by using the command as “`export MYSQL_HOST=<endpoint address>`” for that go inside the created database and go to the Endpoint and select and copy the endpoint address.
- ❖ Now give the credentials of database by giving a command as “`mysql -user=<username> --password=<password> database-name`”.
- ❖ There is another command to access your database as “`mysql -h <endpoint address> -u <user> -p`” press enter button it asks the password enter it and press the enter button.

```

root@ip-172-31-28-11:~#
=====
Package           Arch      Version          Repository        Size
=====
Installing:
mariadb          x86_64   1:5.5.68-1.amzn2.0.1   amzn2-core       0.0 M
=====
Transaction Summary
Install 1 Package
=====
Total download size: 0.0 M
Installed size: 49 M
Downloading packages:
mariadb-5.5.68-1.amzn2.0.1.x86_64.rpm
Running transaction check
Running transaction test
transaction test succeeded
Running transaction
  Installing : 1:mariadb-5.5.68-1.amzn2.0.1.x86_64
  Verifying  : 1:mariadb-5.5.68-1.amzn2.0.1.x86_64
=====
Installed:
  mariadb.x86_64 1:5.5.68-1.amzn2.0.1
=====
Complete!
[root@ip-172-31-28-11 ~]# mysql --version
mysql Ver 15.1 Distrib 5.5.68-MariaDB, for Linux (x86_64) using readline 5.1
[root@ip-172-31-28-11 ~]# export MYSQL_HOST=databasel.clau2u6sigra.us-east-1.rds.amazonaws.com
[root@ip-172-31-28-11 ~]# mysql -h databasel.clau2u6sigra.us-east-1.rds.amazonaws.com -u admin -p
Enter password:
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MySQL connection id is 31
Server version: 8.0.39 Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]> CREATE USER 'wordpress' IDENTIFIED BY 'wordpress-pass';
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''wordpress-pass'' at line 1
MySQL [(none)]> GRANT ALL PRIVILEGES ON wordpress.* TO 'wordpress';
ERROR 1410 (42000): You are not allowed to create a user with GRANT
MySQL [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.01 sec)

MySQL [(none)]> EXIT
Bye
[root@ip-172-31-28-11 ~]#

```

26°C Mostly cloudy ENG IN 12:41 03-12-2024

- ❖ Now create a database user for your wordpress application and give it permission to access the “wordpress” database. By using this commands as
`CREATE USER 'wordpress' IDENTIFIED BY 'wordpress-pass';`
`GRANT ALL PRIVILEGES ON wordpress.* TO 'wordpress';`
`FLUSH PRIVILEGES;`
`EXIT;`

```

root@ip-172-31-28-236:~#
MySQL [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.01 sec)

MySQL [(none)]> EXIT;
Bye
[root@ip-172-31-28-236 ~]# mysql -h database1.clau2u6sigra.us-east-1.rds.amazonaws.com -u admin -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 31
Server version: 8.0.39 Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]> show databases;
+-----+
| Database |
+-----+
| database1 |
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
5 rows in set (0.01 sec)

MySQL [(none)]> Ctrl-C -- exit!
Aborted
[root@ip-172-31-28-236 ~]# mysql -h database1.clau2u6sigra.us-east-1.rds.amazonaws.com -u admin -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 32
Server version: 8.0.39 Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]> SELECT User, Host FROM mysql.user WHERE User = 'wordpress';
+-----+-----+
| User | Host |
+-----+-----+
| wordpress | % |
+-----+-----+
1 row in set (0.00 sec)

MySQL [(none)]>

```

- ❖ To host wordpress application we need a apache web server httpd install that by giving a command as “sudo yum -y install httpd” and start and enable the httpd service by giving the commands as
- ❖ “sudo service httpd start” (or) “sudo systemctl start httpd”.
- ❖ “sudo chkconfig httpd on” (or) “sudo systemctl enable httpd”.



This page is used to test the proper operation of the Apache HTTP server after it has been installed. If you can read this page, it means that the Apache HTTP server installed at this site is working properly.

If you are a member of the general public:

The fact that you are seeing this page indicates that the website you just visited is either experiencing problems, or is undergoing routine maintenance.

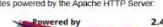
If you would like to let the administrators of this website know that you've seen this page instead of the page you expected, you should send them e-mail. In general, mail sent to the name "webmaster" and directed to the website's domain should reach the appropriate person.

For example, if you experienced problems while visiting www.example.com, you should send e-mail to "webmaster@example.com".

If you are the website administrator:

You may now add content to the directory /var/www/html/. Note that until you do so, people visiting your website will see this page, and not your content. To prevent this page from ever being used, follow the instructions in the file /etc/httpd/conf.d/welcome.conf.

You are free to use the image below on web sites powered by the Apache HTTP Server.



- ❖ Now go to EC2 instance and copy public ip and paste it on google browse it and check the official page of httpd is displaying or not.

```

root@ip-172-31-28-236:~
43 livepatch           available   [=stable]
45 haproxy2           available   [=stable]
46 collectd            available   [=stable]
47 aws-nitro-enclaves-cli
48 R4                  available   [=stable]
49 kernel-5.4          available   [=stable]
50 selinux-5g          available   [=stable]
52 tomcat9             available   [=stable]
53 unbound1.13         available   [=stable]
54 mariadb10.5         available   [=stable]
55 kernel-5.10=latest  enabled     [=stable]
56 redis6              available   [=stable]
59 postgresql13        available   [=stable]
60 mock2               available   [=stable]
61 dnsmasq2.85         available   [=stable]
62 kernel-5.15         available   [=stable]
63 postgresql14        available   [=stable]
64 firefox             available   [=stable]
65 lustre              available   [=stable]
67 avsc11              available   [=stable]
68 rphp8.2              available   [=stable]
69 dnsmasq              available   [=stable]
70 unbound1.17         available   [=stable]
72 collectd-python3    available   [=stable]

* Extra topics reached end of support.
* Note: End-of-support. Use 'info' subcommand.
[root@ip-172-31-28-236 ~]# sudo yum -y update
Loaded plugins: extras suggestions, langpacks, priorities, update-motd
Existing lock /var/run/yum.pid: another copy is running as pid 3697.
Another app is currently holding the yum lock; waiting for it to exit...
The other application is: yum
Memory : 220 M RSS (441 MB VSIZE)
Started: Tue Dec 3 17:36:52 2024 - 00:06 ago
State : Running, pid: 3697
No packages marked for update
[root@ip-172-31-28-236 ~]# ll
total 27920
-rw-r--r-- 1 root root 28585184 Nov 21 14:08 latest.zip
drwxr-xr-x 5 root root 4096 Nov 21 14:07 wordpress
[root@ip-172-31-28-236 ~]# cd wordpress/
[root@ip-172-31-28-236 wordpress]# sudo mv wp-config-sample.php wp-config.php
[root@ip-172-31-28-236 wordpress]# sudo vi wp-config.php
[root@ip-172-31-28-236 wordpress]# cd ..
[root@ip-172-31-28-236 ~]# sudo cp -r wordpress/* /var/www/html/
[root@ip-172-31-28-236 ~]# ls /var/www/html/
index.php  readme.html  wp-admin  wp-comments-post.php  wp-content  wp-includes  wp-load.php  wp-mail.php  wp-signup.php  xmlrpc.php
license.txt  wp-activate.php  wp-blog-header.php  wp-config.php  wp-cron.php  wp-links-opml.php  wp-login.php  wp-settings.php  wp-trackback.php
[root@ip-172-31-28-236 ~]# sudo systemctl restart httpd
[root@ip-172-31-28-236 ~]#

```

Party cloudy 24°C ENG IN 23:16 03-12-2024

- ❖ Now go to browser and search as download word press and click on proper link and select and copy the address link of word press download file and paste that along with wget command in git bash as “wget <address link of word press download file?”
- ❖ It gives the zip file to unzip that file by using a command as “unzip <zipfile>”

```

root@ip-172-31-28-236:~
total 27920
-rw-r--r-- 1 root root 28585184 Nov 21 14:08 latest.zip
drwxr-xr-x 5 root root 4096 Nov 21 14:07 wordpress
[root@ip-172-31-28-236 ~]# sudo amzn2-extras install -y lamp-mariadb10.2-php7.2 php7.2
Total 1 package(s) will be installed at 2020-11-30
Topic: php7.2 has end-of-support date of 2020-11-30
Installing package: php-pdo, php-mysqlnd, php-fpm, php-json, mariadb
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Cleaning repos: amzn2-core amzn2extra-docker amzn2extra-kernel-5.10 amzn2extra-lamp-mariadb10.2-php7.2 amzn2extra-php7.2
17 metadata files removed
6 sqlite files removed
0 metadata files removed
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core
amzn2extra-docker
amzn2extra-kernel-5.10
amzn2extra-lamp-mariadb10.2-php7.2
amzn2extra-php7.2
(1/1): amzn2-core/2/x86_64/group.gz
(2/1): amzn2-core/2/x86_64/updateinfo
(3/1): amzn2extra-docker/2/x86_64/updateinfo
(4/1): amzn2extra-kernel-5.10/2/x86_64/updateinfo
(5/1): amzn2extra-lamp-mariadb10.2/2/x86_64/updateinfo
(6/1): amzn2extra-php7.2/2/x86_64/updateinfo
(7/1): amzn2extra-docker/2/x86_64/primary_db
(8/1): amzn2extra-lamp-mariadb10.2/2/x86_64/primary_db
(9/1): amzn2extra-php7.2/2/x86_64/primary_db
(10/1): amzn2extra-kernel-5.10/2/x86_64/primary_db
(11/1): amzn2core/2/x86_64/primary_db
Resolving Dependencies
--> Running transaction check
--> Package mariadb.x86_64 1:5.5.60-1.amzn2.0.1 will be updated
--> Package mariadb.x86_64 1:5.5.60-1.amzn2.0.1 will be an update
--> Processing Dependency: mariadb-common(x86-64) = 1:10.2.38-1.amzn2.0.1 for package: 3:mariadb-10.2.38-1.amzn2.0.1.x86_64
--> Processing Dependency: mariadb-common(x86-64) = 3:10.2.38-1.amzn2.0.1 for package: 3:mariadb-10.2.38-1.amzn2.0.1.x86_64
--> Package mariadb.x86_64 0:7.2.34-1.amzn2 will be installed
--> Processing Dependency: mariadb-common(x86-64) = 7.2.34-1.amzn2 for package: php-clients-7.2.34-1.amzn2.x86_64
--> Package mariadb.x86_64 0:7.2.34-1.amzn2 will be installed
--> Running transaction check
--> Package mariadb.x86_64 3:10.2.38-1.amzn2.0.1 will be installed
--> Processing Dependency: /etc/my.cnf for package: 3:mariadb-common-10.2.38-1.amzn2.0.1.x86_64
--> Package mariadb-libs.x86_64 1:5.5.68-1.amzn2.0.1 will be updated
--> Package mariadb-libs.x86_64 3:10.2.38-1.amzn2.0.1 will be an update
--> Package mariadb-common.x86_64 0:7.2.34-1.amzn2 will be installed
--> Processing Dependency: libzip.so.5() (64bit) for package: php-common-7.2.34-1.amzn2.x86_64
--> Running transaction check

```

Party cloudy 24°C ENG IN 23:16 03-12-2024

- ❖ To run wordpress web application, you have to install run time of word press web application as php language with the following command <sudo amzn2-extras install -y lamp-mariadb10.2-php7.2 php7.2> otherwise update your EC2 instance with the following command as “<sudo yum -y update and sudo yum -y upgrade>”.
- ❖ Now go inside the unzip directory by using command as “cd <unzip directory>” and change the word press configuration file by giving command as “sudo mv wp-config-sample.php wp-config.php”.

```

root@ip-172-31-28-236:~/wordpress
$ cat wp-config.php
/*
 * The base URL for the site. This is used for creating links.
 * You can use the full URL or just the path, whichever is more convenient.
 */
define('WP_HOME', 'http://ip-172-31-28-236');
define('WP_SITEURL', 'http://ip-172-31-28-236');

/*
 * Database settings - You can get this info from your web host. */
define('DB_NAME', 'database1');
define('DB_USER', 'admin');
define('DB_PASSWORD', 'admin123');
define('DB_HOST', 'database1.cjou2edigre.us-east-1.rds.amazonaws.com');

/*
 * 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', '');

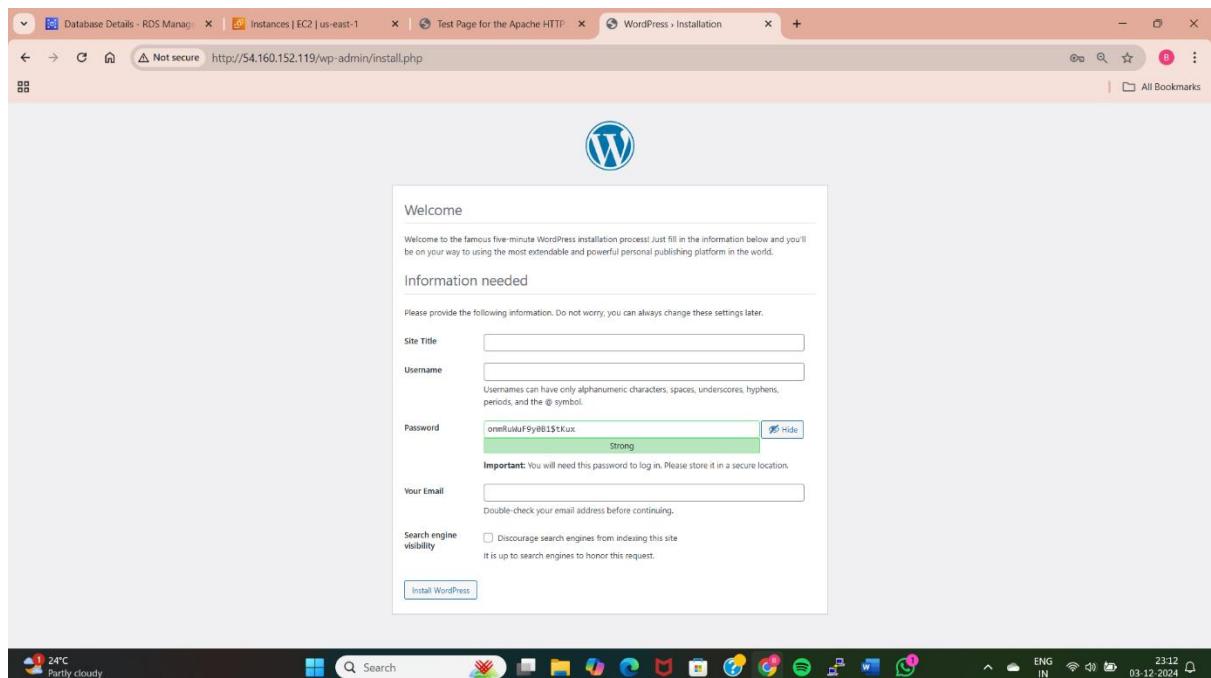
/*
 * Authentication unique keys and salts.
 *
 * Change these to different unique phrases! You can generate them via:
 * https://api.wordpress.org/secret-key/1.1/salt/ (WordPress.org secret-key service)
 */
define('AUTH_KEY', 'e:BuGCPw1-WyayvRow.[0W'^~](RBC+O)[9~oWlo]XADOF[2H,7y)A5pEg');
define('SECURE_AUTH_KEY', '(a)sM6C1c1+ir^?YLCRMW!la+o+1B1)lRf+By|W-XrT!4NvVg');
define('NONCE_KEY', '10p+1iw1+kra4bcb9-UW!o.e,1Nc3inRw_g_alk-c5+D_K0N6M'W');
define('AUTH_SALT', '1Nwp(2)hcr6238(nz+r#B#92,g0cHNEJN24)Nm6mPoY/.h+|w|gYok7z/6');
define('SECURE_AUTH_SALT', 'bdN!Z2Pvn-JK-B/yV egz1-3LTYf(y)69S!(U71),iEL(j+UmBKJ,Mx 1');
define('NONCE_SALT', 'WAAd5GUThe%oVt!ojq/d4#*d)zv1|Gd3lwvt6,xEDG;nsUcg_g0-8aaK');
define('LOGGED_IN_SALT', '1M#F#3xD:7MA(ihsG:X mcmif/C8n/LPVWb07Kx+&C1**jEc0efA');
define('NONCE_SALT', '%:sp*8*8-52)=AR5Tnp!Q#nJl8Xbukkeg=.p) 77+I+eubWmuls0)yr)876-0a';

/*
 * WordPress database table prefix.
 */
$wpdb->prefix = 'wp_';

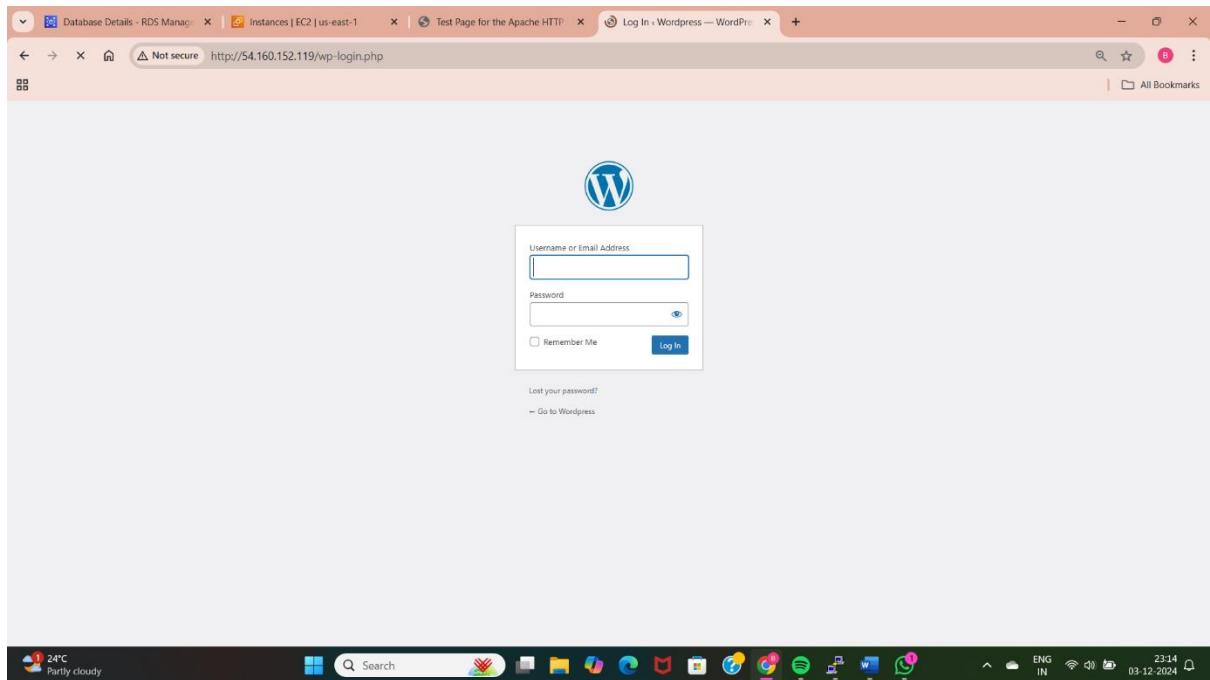
"wp-config.php" [dos] 101L, 3633B

```

- ❖ Now do some configurations in word press configuration file by giving database name, username, password and host name for that execute a command as “sudo vi wp-config.php”
- ❖ Search in google “Wordpress secret key generator”. And use that password.
- ❖ Now copy this wordpress directory to the document root directory to host your web application of wordpress by giving a command as
“sudo cp -r <unzip file or wordpress directory> /* /var/www/html/”
“ls /var/www/html/”



- ❖ Restart the httpd by giving a commands “systemctl restart httpd”.

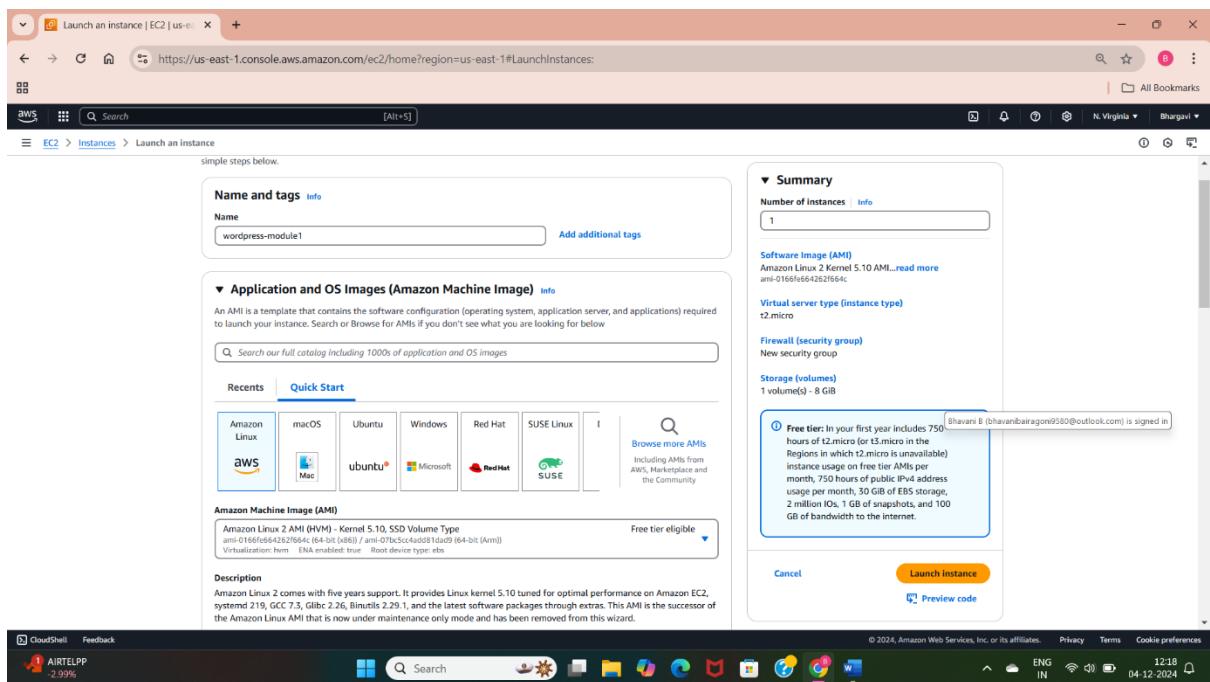


- ❖ Now go to EC2 instance and copy public ip and paste it on google browse it and check the official page of wordpress is displays.

METHOD-2: CREATING AND LAUNCH AN AMAZON EC2 INSTANCE:

Amazon EC2 is virtual server which provides compute optimization for our applications by providing CPU, storage and RAM.

- ➲ First login to the AWS account with credentials.



- ➲ Now give the name for EC2 instance.
- ➲ Select your AMI linux or ununtu 22.04 ltd free tier 22,80

```

root@ip-172-31-92-233: ~$ 
root@ip-172-31-92-233: ~$ 
root@ip-172-31-92-233: ~$ Authenticating with public key "imported-openssh-key"
root@ip-172-31-92-233: ~$ 
Amazon Linux 2
AL2 End of Life is 2025-06-30.
root@ip-172-31-92-233: ~$ 
A newer version of Amazon Linux is available!
Amazon Linux 2023, GA and supported until 2029-03-15.
https://aws.amazon.com/linux/amazon-linux-2023/
root@ip-172-31-92-233: ~$ 
root@ip-172-31-92-233: ~$ yum install git -y

```

⦿ Launch instance with any one of the terminal.

```

root@ip-172-31-92-233: ~$ 
root@ip-172-31-92-233: ~$ 
root@ip-172-31-92-233: ~$ 
root@ip-172-31-92-233: ~$ yum install git perl-Git perl-TermReadKey -y
root@ip-172-31-92-233: ~$ 
root@ip-172-31-92-233: ~$ yum install docker -y
root@ip-172-31-92-233: ~$ curl -L https://github.com/docker/compose/releases/download/2.15.0/docker-compose-$(uname -s)-$(uname -m)
root@ip-172-31-92-233: ~$ 
root@ip-172-31-92-233: ~$ docker-compose --version

```

⦿ Install git, docker and setup docker environment.

```

root@ip-172-31-29-106: ~$ 
root@ip-172-31-29-106: ~$ 
root@ip-172-31-29-106: ~$ 
root@ip-172-31-29-106: ~$ sudo systemctl start docker
[sudo] password for root: 
root@ip-172-31-29-106: ~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
   Active: active (running) since wed 2024-12-04 06:54:37 UTC; 32s ago
     Docs: https://docs.docker.com
Main PID: 3542 (dockerd)
   CGroup: /system.slice/docker.service
           └─3542 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Dec 04 06:54:36 ip-172-31-29-106.ec2.internal systemd[1]: Starting Docker Application Con...
Dec 04 06:54:36 ip-172-31-29-106.ec2.internal docker[3542]: time="2024-12-04T06:54:36.99..."
Dec 04 06:54:36 ip-172-31-29-106.ec2.internal docker[3542]: time="2024-12-04T06:54:36.99..."
Dec 04 06:54:37 ip-172-31-29-106.ec2.internal docker[3542]: time="2024-12-04T06:54:37.18..."
Dec 04 06:54:37 ip-172-31-29-106.ec2.internal docker[3542]: time="2024-12-04T06:54:37.19..."
Dec 04 06:54:37 ip-172-31-29-106.ec2.internal docker[3542]: time="2024-12-04T06:54:37.19..."
Dec 04 06:54:37 ip-172-31-29-106.ec2.internal docker[3542]: time="2024-12-04T06:54:37.19...6"
Dec 04 06:54:37 ip-172-31-29-106.ec2.internal docker[3542]: time="2024-12-04T06:54:37.19...6"
Dec 04 06:54:37 ip-172-31-29-106.ec2.internal docker[3542]: time="2024-12-04T06:54:37.22..."

Hint: Some lines were ellipsized, use -l to show in full.
root@ip-172-31-29-106: ~$ 

```

⦿ And provide permissions for admin socket file.

- Now give permissions to add limited linux user account to docker group by using a command as “`<sudo usermod -aG username(ec2-user)>` or `<sudo usermod -a -G username(ec2-user)>` another command is `<sudo chmod 666 /var/run/docker.sock>`

- Install docker compose.(go to browser and use docker compose installation) and we get the following commands

- I. sudo curl -L
"https://github.com/docker/compose/releases/download/v2.22.0/docker-compose-
\$(uname -s)-\$(uname -m)" -o /usr/local/bin/docker-compose
 - II. sudo curl -L
"https://github.com/docker/compose/releases/download/v2.22.0/docker-compose-
\$(uname -s)-\$(uname -m)" -o /usr/local/bin/docker-compose
 - III. ls -lh /usr/local/bin/docker-compose
 - IV. sudo chmod +x /usr/local/bin/docker-compose
 - V. docker-compose –version

- ⦿ Apply executable permissions to the binary by using command as <Sudo chmod +x /usr/local/bin/docker-compose>
 - ⦿ Now create the symbolic link by using command as <ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose>
 - ⦿ Create a docker-compose.yml file for wordpress application.

```
root@ip-172-31-92-233:~#
[ec2-user] login as: ec2-user
Authenticating with public key "imported-openssh-key"
Last login: Wed Dec 4 05:37:59 2024 from 124.123.185.235
      _                Amazon Linux 2
     / \              AL2 End of Life is 2025-06-30.
     \   \             A newer version of Amazon Linux is available!
      \   /             Amazon Linux 2023, GA and supported until 2028-03-15.
       \_/\_             https://aws.amazon.com/linux/amazon-linux-2023/
[ec2-user@ip-172-31-92-233 ~]$ sudo su -
Last login: Wed Dec 4 05:38:03 UTC 2024 on pts/0
[root@ip-172-31-92-233 ~]# ll
total 0
[root@ip-172-31-92-233 ~]# rm docker-compose.yml
[root@ip-172-31-92-233 ~]# vi docker-compose.yml
[root@ip-172-31-92-233 ~]# docker-compose up -d
parsing /root/docker-compose.yml: yaml: line 15: mapping values are not allowed in this conte
xt
[root@ip-172-31-92-233 ~]# docker pull mysql:8.0.19
8.0.19: Pulling from library/mysql
54fec2fa59d0: Pull complete
cc6c6145912: Pull complete
951cd3959c9d: Pull complete
03de6d0e206e: Pull complete
319f0394ef42: Pull complete
d9185034607b: Pull complete
018a56c64dad: Pull complete
96d4c3d1f9f9: Pull complete
785bc590808da: Pull complete
l339c094729: Pull complete
cebf8f531cc37: Pull complete
2ba0c9f6a918: Pull complete
Digest: sha256:95649fbad6330d1068ff092292dc820995e7b792c17d4e94dd95255fld5449
Status: Downloaded newer image for mysql:8.0.19
docker.10: Pulling from library/wordpress
[root@ip-172-31-92-233 ~]# vim docker-compose.yml
[root@ip-172-31-92-233 ~]# docker pull wordpress:latest
latest: Pulling from library/wordpress
bc0965b23a04: Pull complete
bc0965b23a04: Pull complete
e4aa8e8bf7d10: Pull complete
e45eeb7b7c66: Pull complete
2802fa07e46: Pull complete
2bc706e6dbba: Pull complete
9f2b5a95cfcd: Pull complete
[ec2-user@ip-172-31-92-233 ~]#
```

```
version : '3.3'
services:
  db:
    image: mysql: 8.0.19
    command: 'default-authentication-plugin=mysql_native_password'
    volumes:
      - db_data: /var/lib/mysql
    restart: always
    environment:
      - MYSQL_ROOT_PASSWORD=wordpress
      - MYSQL_DATABASE=databasename
      - MYSQL_USER=username
      - MYSQL_PASSWORD=password
  wordpress:
    image: wordpress: latest
    ports:
      - 80:80
    restart: always
    environment:
      - WORDPRESS_DB_HOST=db
      - WORDPRESS_DB_USER=username
      - WORDPRESS_DB_PASSWORD=password
      - WORDPRESS_DB_NAME=db name
```

volumes:

db_data:

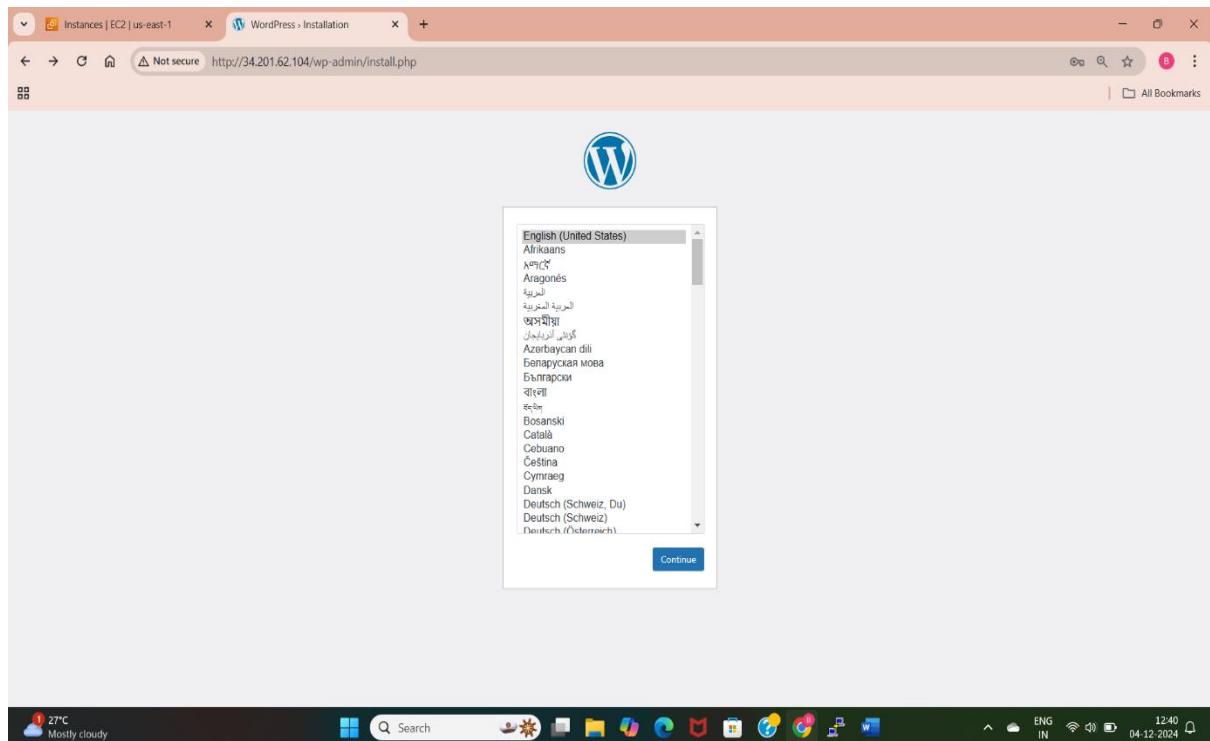
- ⌚ create a file with the name of docker-compose.yml.
- ⌚ To run the docker file we need to run the “docker-compose up -d” command.

```

root@ip-172-31-92-233:~#
Run 'docker COMMAND --help' for more information on a command.
For more help on how to use Docker, head to https://docs.docker.com/go/guides/
[root@ip-172-31-92-233 ~]# ^C
[root@ip-172-31-92-233 ~]# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
[root@ip-172-31-92-233 ~]# docker compose version
docker: 'compose' is not a docker command.
See 'docker --help'
[root@ip-172-31-92-233 ~]# docker-compose version
Docker Compose version v2.22.0
[root@ip-172-31-92-233 ~]# sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
ln: failed to create symbolic link '/usr/bin/docker-compose': File exists
[root@ip-172-31-92-233 ~]# ^C
[root@ip-172-31-92-233 ~]# ls -l /usr/bin/docker-compose
lrwxrwxrwx 1 root root 29 Dec 4 05:54 /usr/bin/docker-compose -> /usr/local/bin/docker-compose
[root@ip-172-31-92-233 ~]# !Sudo vi /etc/sudoers
-bash: !Sudo: command not found
[root@ip-172-31-92-233 ~]# Sudo vi /etc/sudoers
-bash: Sudo: command not found
[root@ip-172-31-92-233 ~]# ls -l /usr/bin/docker-compose
lrwxrwxrwx 1 root root 29 Dec 4 05:54 /usr/bin/docker-compose -> /usr/local/bin/docker-compose
[root@ip-172-31-92-233 ~]# sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
[root@ip-172-31-92-233 ~]# docker-compose --version
Docker Compose version v2.22.0
[root@ip-172-31-92-233 ~]# docker-compose up -d
parsing /root/docker-compose.yml: yaml: line 15: mapping values are not allowed in this context
[root@ip-172-31-92-233 ~]# ^C
[root@ip-172-31-92-233 ~]# docker-compose config
parsing /root/docker-compose.yml: yaml: line 15: mapping values are not allowed in this context
[root@ip-172-31-92-233 ~]# docker-compose config
parsing /root/docker-compose.yml: yaml: line 15: mapping values are not allowed in this context
[root@ip-172-31-92-233 ~]# vim docker-compose.yml
[root@ip-172-31-92-233 ~]# docker-compose up -d
[+] Building 0.0s (0/0)
[+] Running 4/4
  └─ Network root_default      Created
    └─ Volume "root_db_data"   Created
      └─ Container root-wordpress-1 Started
        └─ Container root-db-1   Started
[root@ip-172-31-92-233 ~]# docker-compose ps
NAME          IMAGE           COMMAND                  SERVICE      CREATED        STATUS        PORTS
root-db-1     mysql:8.0.19   "docker-entrypoint.sh --default-authentication-plugin=mysql_native_password" db      25 seconds ago Up 24 seconds  3306/tcp, 33060/tcp
root-wordpress-1  wordpress:latest "docker-entrypoint.sh apache2-foreground"      wordpress  25 seconds ago Up 24 seconds  0.0.0.0:80->80/tcp, ::80->
80/tcp
[root@ip-172-31-92-233 ~]#

```

- Now pull the images of MYSQL and WORDPRESS we have to execute this created docker-compose.yml file by using command as <docker-compose up -d>



- Now see the containers of the pulled images of MYSQL and Wordpress

```

root@ip-172-31-29-106:~#
  * 1339cf094729 Pull complete
  * beb8f531cc37 Pull complete
  * 2b40c9f6a918 Pull complete
wordpress 22 layers [=====] 0B/0B Pulled
  * bc0965b23a04 Pull complete
  * e0aa8e8bfdf10 Pull complete
  * f45eeb7b7c66 Pull complete
  * 2802fa207e46 Pull complete
  * 2bc706edabfe Pull complete
  * 8f2885a9a5cfda Pull complete
  * 258efc1a296f Pull complete
  * 61a388cf22e3 Pull complete
  * 73d46b827991 Pull complete
  * c2dd75e5e9cab Pull complete
  * 3b015641381f9 Pull complete
  * 1ed4d5113a0d Pull complete
  * 4f4fb700054 Pull complete
  * cfaf8720db1e Pull complete
  * d374174149dd Pull complete
  * f09c82e2e21b Pull complete
  * dd7711b88413 Pull complete
  * a89cceed0693 Pull complete
  * dab74dcf5d37 Pull complete
  * ee6f699a1365 Pull complete
  * 1bbc7feebad6 Pull complete
[+] Building 0.0s (0/0)
[+] Running 4/4
  * None: > root default Created
  * Volume "root_db_data" Created
  * Container root-db-1 Started
  * Container root-wordpress-1 Started
[root@ip-172-31-29-106 ~]# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
58a4ea8a29df9 mysql:8.0.19 "docker-entrypoint.s..." 2 minutes ago Up 2 minutes 3306/tcp, 33060/tcp root-db-1
c10fcac0cabf7 wordpress:latest "docker-entrypoint.s..." 2 minutes ago Up 2 minutes 0.0.0.0:80->80/tcp, :::80->80/tcp root-wordpress-1
[root@ip-172-31-29-106 ~]# docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
58a4ea8a29df9 mysql:8.0.19 "docker-entrypoint.s..." 2 minutes ago Up 2 minutes 3306/tcp, 33060/tcp root-db-1
c10fcac0cabf7 wordpress:latest "docker-entrypoint.s..." 2 minutes ago Up 2 minutes 0.0.0.0:80->80/tcp, :::80->80/tcp root-wordpress-1
[root@ip-172-31-29-106 ~]# sudo visudo
[root@ip-172-31-29-106 ~]#
[root@ip-172-31-29-106 ~]#
[root@ip-172-31-29-106 ~]# sudo vi /etc/sudoers
[root@ip-172-31-29-106 ~]# sudo is /root
docker-compute.yml
[root@ip-172-31-29-106 ~]# ^C
[root@ip-172-31-29-106 ~]# 
```

- ⌚ Sudo visudo Jenkins (ALL)-ALL NOPASSWD: ALL –linux amazon
- ⌚ Sudo vi /etc/sudoers

```

root@ip-172-31-92-233:~#
ls List images
prune Remove unused images
pull Download an image from a registry
push Upload an image to a registry
rm Remove one or more images
save Save one or more images to a tar archive (streamed to STDOUT by default)
tag Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
Run 'docker image COMMAND --help' for more information on a command.
[root@ip-172-31-92-233 ~]# docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
wordpress latest c89b0a25cd1 12 days ago 546MB
mysql 8.0.19 0c27e8efcfca 4 years ago 546MB
[root@ip-172-31-92-233 ~]# ^C
[root@ip-172-31-92-233 ~]# docker tag <image_id> <your_dockerhub_username/<repository_name>:<tag>
-bash: syntax error near unexpected token `<'` 
[root@ip-172-31-92-233 ~]# docker tag c89b0a25cd1 bhargavibairagi.wordpress:wordimage
Error response from daemon: No such image: c89b0a25cd1:latest
[root@ip-172-31-92-233 ~]# ^C
[root@ip-172-31-92-233 ~]# docker tag c89b0a25cd1 bhargavibairagi.wordpress:wordimage
Error response from daemon: No such image: c89b0a25cd1:latest
[root@ip-172-31-92-233 ~]# ^C
[root@ip-172-31-92-233 ~]# docker tag c89b40a25cd1 bhargavibairagi.wordpress:wordimage
[root@ip-172-31-92-233 ~]# docker push bhargavibairagi.wordpress:wordimage
The push refers to repository [docker.io/bhargavibairagi.wordpress]
b959f5a5096a: Mounted from library/wordpress
346232192bfff: Mounted from library/wordpress
990c1071670a: Mounted from library/wordpress
d893323130a9: Mounted from library/wordpress
65de479bc82b: Mounted from library/wordpress
546915f46637: Mounted from library/wordpress
39520dbeel90b: Mounted from library/wordpress
17bf5ffcd60b: Mounted from library/wordpress
5770bf18a086: Mounted from library/wordpress
5c28a2047cd6: Mounted from library/wordpress
iae7c302f972: Mounted from library/wordpress
cc97a818902f: Mounted from library/wordpress
7c1000e9fc5: Mounted from library/wordpress
8644451feazl: Mounted from library/wordpress
4fc24dc5bc0a9: Mounted from library/wordpress
059f61273d8b: Mounted from library/wordpress
1mb1271d453: Mounted from library/wordpress
0694d3059a: Mounted from library/wordpress
e05626eh0df1: Mounted from library/wordpress
83a3fbfb0cc: Mounted from library/wordpress
7c197840506f: Mounted from library/wordpress
c0f1022b2a9: Mounted from library/wordpress
wordimage: digest: sha256:bcff2876e1890e37c6676997dbe2638aadda561d91617ff9a223dfe673fae36 size: 4917
[root@ip-172-31-92-233 ~]# 
```

- ⌚ Push the docker images to the docker hub.

METHOD-3: DEPLOYING WORDPRESS WEB APPLICATION BY USING GIT AND JENKINS

- To deploy wordpress application using git and jenkins first we need to install all the dependencies required for this applications.

```
Putty (inactive)
login as: ec2-user
Authenticating with public key "imported-openssh-key"
Amazon Linux 2
AL2 End of Life is 2025-06-30.
A newer version of Amazon Linux is available!
Amazon Linux 2023, GA and supported until 2028-03-15.
https://aws.amazon.com/linux/amazon-linux-2023/

[ec2-user@ip-172-31-92-215 ~]$ sudo su -
[root@ip-172-31-92-215 ~]# yum install git -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Resolving Dependencies
--> Running transaction check
--> Package git.x86_64 0:2.40.1-1.amzn2.0.3 will be installed
--> Processing Dependency: git-core = 2.40.1-1.amzn2.0.3 for package: git-2.40.1-1.amzn2.0.3.x86_64
--> Processing Dependency: git-core-doc = 2.40.1-1.amzn2.0.3 for package: git-2.40.1-1.amzn2.0.3.x86_64
--> Processing Dependency: perl-git = 2.40.1-1.amzn2.0.3 for package: git-2.40.1-1.amzn2.0.3.x86_64
--> Processing Dependency: perl-Git = 2.40.1-1.amzn2.0.3 for package: git-2.40.1-1.amzn2.0.3.x86_64
--> Processing Dependency: perl(Term::ReadKey) for package: git-2.40.1-1.amzn2.0.3.x86_64
--> Running transaction check
--> Package git-core.x86_64 0:2.40.1-1.amzn2.0.3 will be installed
--> Package git-core-doc.noarch 0:2.40.1-1.amzn2.0.3 will be installed
--> Package perl-Git.noarch 2.40.1-1.amzn2.0.3 will be installed
--> Processing Dependency: perl(Error) for package: perl-Git-2.40.1-1.amzn2.0.3.noarch
--> Package perl-TermReadKey.x86_64 0:2.30-20.amzn2.0.2 will be installed
--> Running transaction check
--> Package perl-Error.noarch 1:0.17020-2.amzn2 will be installed
--> Finished Dependency Resolution

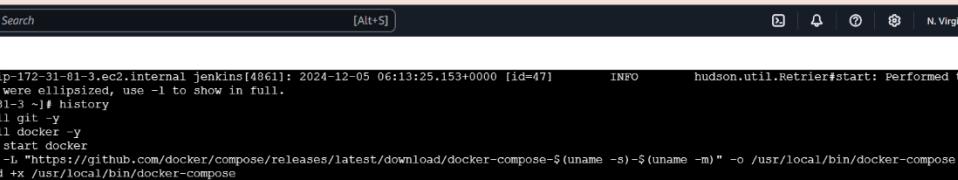
Dependencies Resolved

=====
                         Arch           Version            Repository      Size
=====
Installing:
git                           x86_64        2.40.1-1.amzn2.0.3      amzn2-core      54 k
Installing for dependencies:
git-core                       x86_64        2.40.1-1.amzn2.0.3      amzn2-core      10 M
git-core-doc                   noarch       2.40.1-1.amzn2.0.3      amzn2-core     31 k
perl-Error                     noarch       1:0.17020-2.amzn2      amzn2-core      35 k
perl-Git                        noarch       2.40.1-1.amzn2.0.3      amzn2-core      42 k
perl-TermReadKey                x86_64        2.30-20.amzn2.0.2      amzn2-core      31 k

Transaction Summary
  Nifty bank -0.39%
```

Here I have installed the git using “`yum install git -y`”. Because I am using the above docker-compose file and that is stored in the github.

- Install docker using “`yum install docker -y`”, start the docker using “`systemctl start docker`”.
 - Install docker-compose because the file is written in the file `docker-compose.yml`.



The screenshot shows a terminal window with the following content:

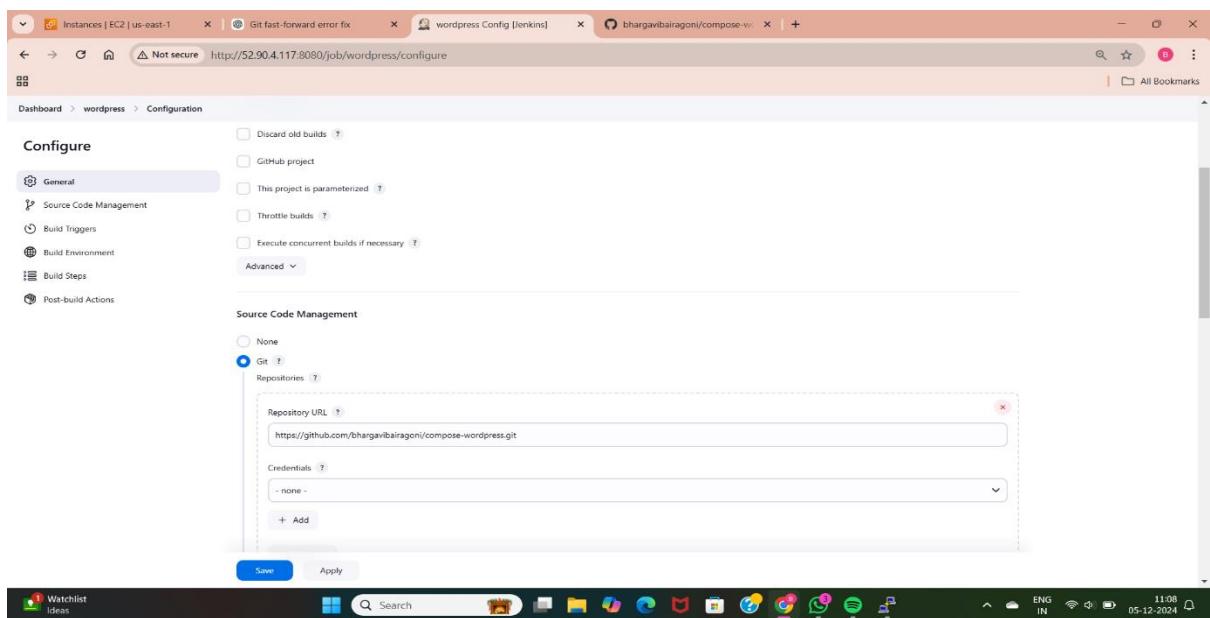
```
Dec 05 06:13:25 ip-172-31-81-3.ec2.internal Jenkins[4861]: 2024-12-05 06:13:25.153+0000 [id=47]      INFO      hudson.util.Retryer#start: Performed the ...tempt #1
Hint: Some lines were ellipsized, use -l to show in full.
[root@ip-172-31-81-3 ~]# history
 1 yum install git -y
 2 yum install docker -y
 3 systemctl start docker
 4 sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
 5 sudo chmod +x /usr/local/bin/docker-compose
 6 docker-compose --version
 7 sudo usermod -aG docker ec2-user
 8 sudo chmod 666 /var/run/docker.sock
 9 sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
10 systemctl start docker
11 sudo systemctl status docker
12 ls /var/run/docker.sock
13 sudo chmod 666 /var/run/docker.sock
14 sudo chmod +x /usr/local/bin/docker-compose
15 sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo
16 sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key
17 sudo yum install java-11-amazon-corretto-devel -y
18 java --version
19 yum install jenkins -y
20 yum install mysql -y
21 systemctl start jenkins
22 systemctl status jenkins
23 history
[root@ip-172-31-81-3 ~]#
```

These are the commands used to install docker-compose, giving permissions to the docker-compose to execute the file. Create a user called ec2-user and assign it to the group called docker.

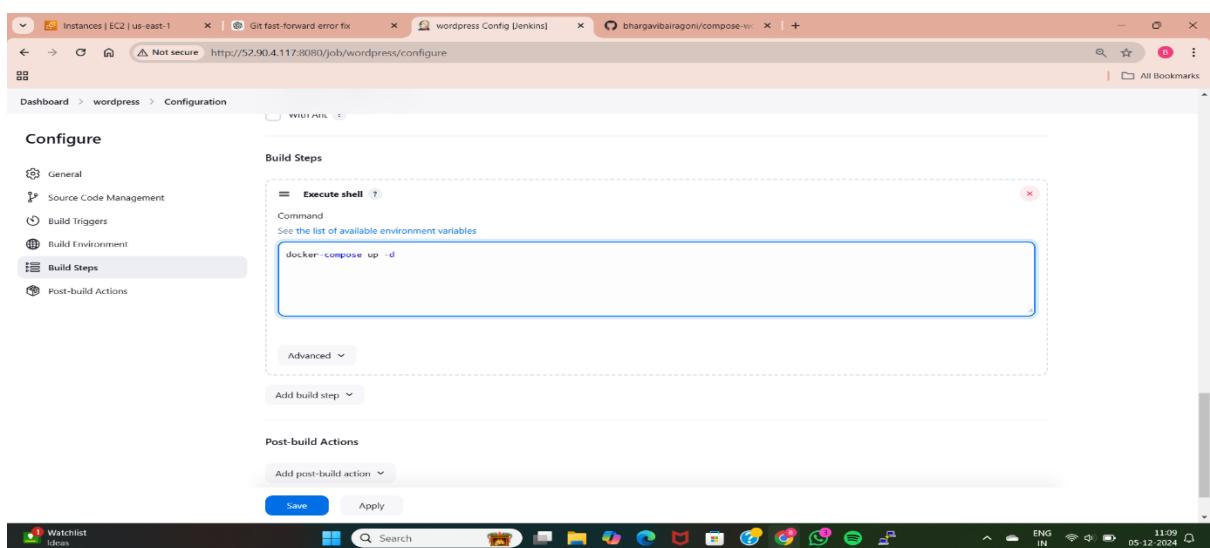
- Creating a symbolic link (`ln -s`) which is used as shortcut name or alias name

- Downloading the official Jenkins repository file from the internet and saves it to the system's repository directory. This allows the system to use yum to install and update Jenkins easily.
- The command imports the Jenkins GPG key to verify the authenticity of packages from the Jenkins repository. It ensures that the packages are safe and haven't been tampered with.
- Install java 17 version as dependency for Jenkins using the command "yum install java-17-amazon-corretto-devel -y"
- Install jenkins using "yum install Jenkins -y" and start the Jenkins using "systemctl start Jenkins".
- To access the Jenkins we need to copy the instance public ip and paste it in google by specifying Jenkins port number 8080 (ip:8080)

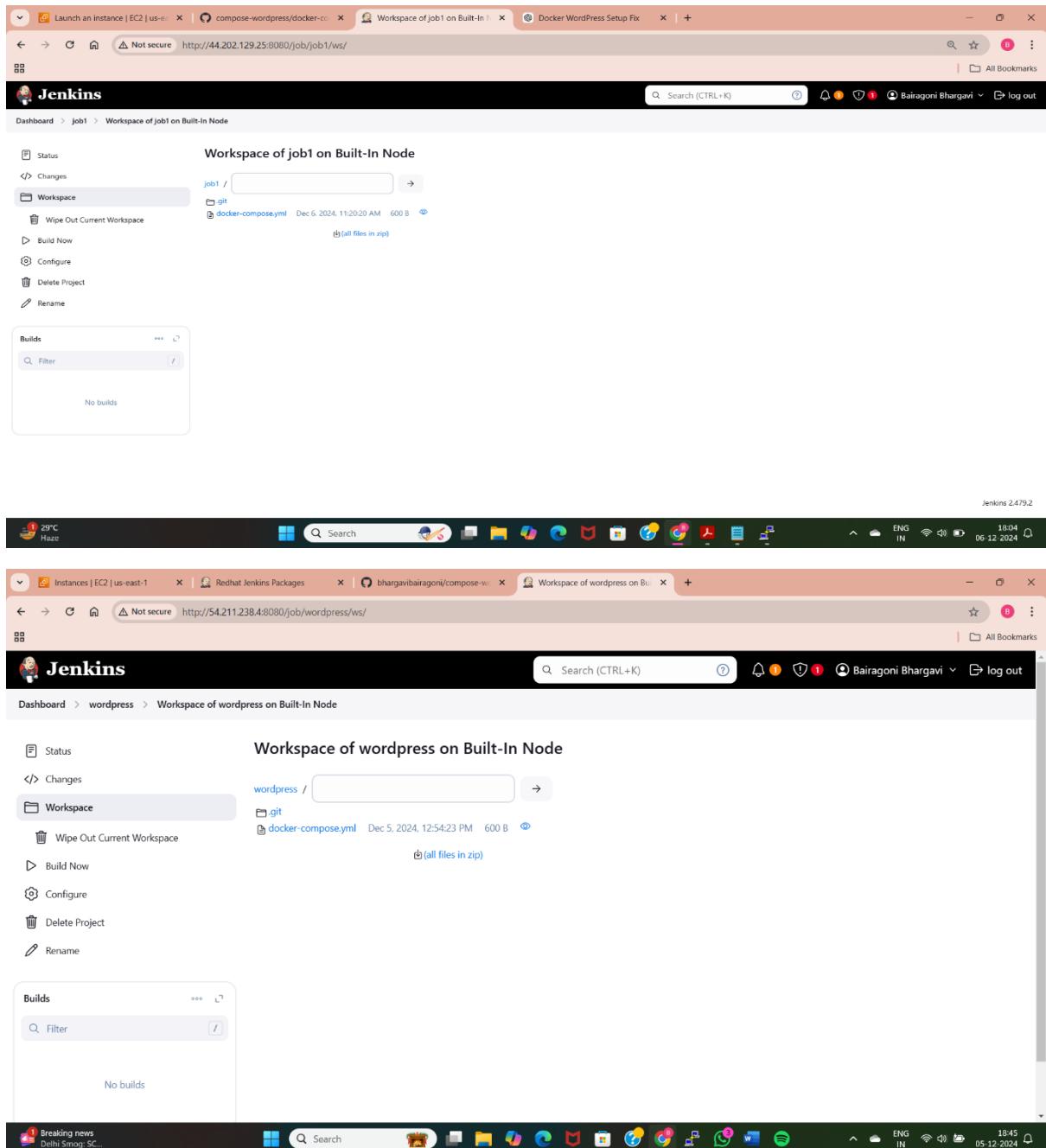
Here I have created a free-style job with the name of job1 and using that job I have deployed the wordpress application.



- Here am taking the compose file from github and kept in Jenkins. I have stored the compose file in public repository, so here no need to provide any credentials.



- Here am using build step called “Execute shell”. Whatever we mention in the execute shell is going to execute. Here am running the docker-compose.yml file to access the wordpress application.



- After specifying the configurations in the Jenkins we need to build the code to access the application and we can also identify errors and correct the errors using configure option.

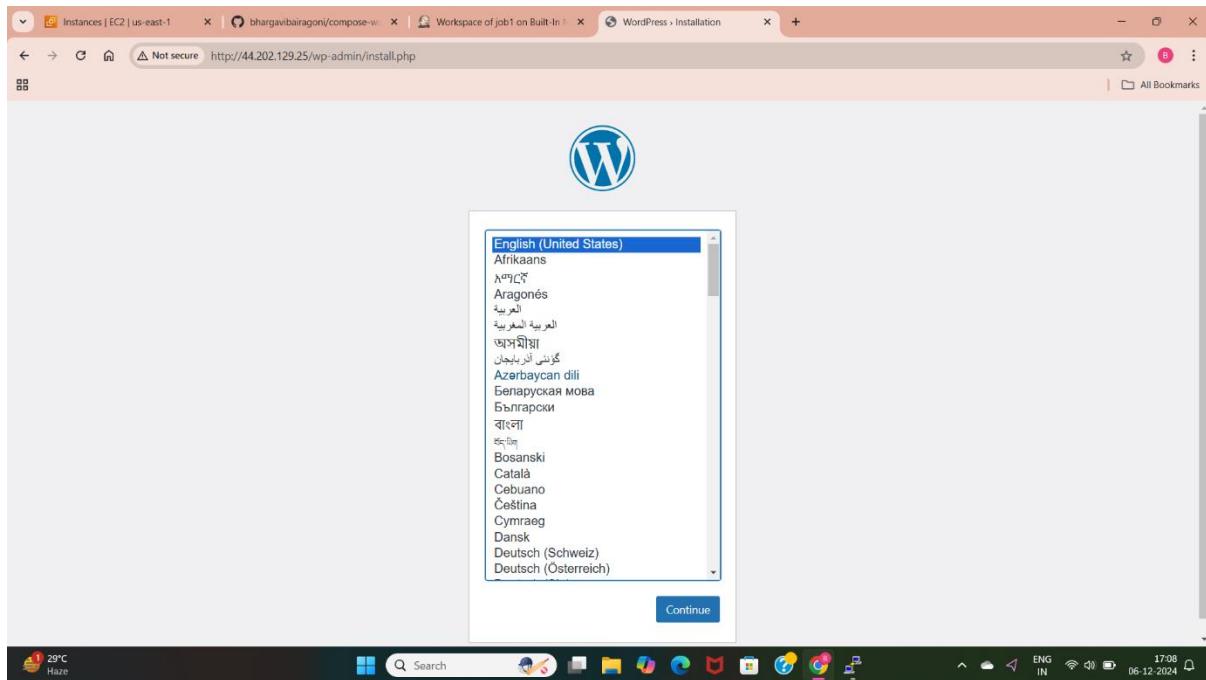
- ⦿ Here the build is successful. We can access the application now by using the ip address of the instance.

```

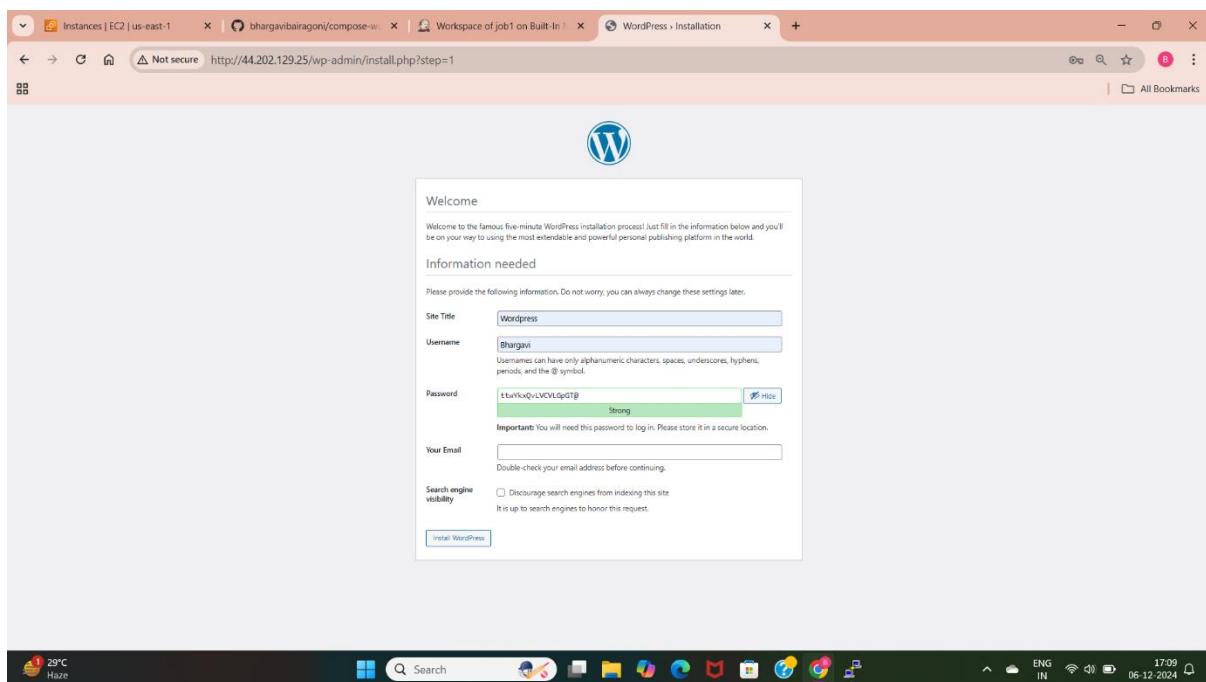
root@ip-172-31-88-21:/var/lib/jenkins/workspace/wordpress
. . .
AL2 End of Life is 2025-06-30.
A newer version of Amazon Linux is available!
Amazon Linux 2023, GA and supported until 2028-03-15.
https://aws.amazon.com/linux/amazon-linux-2023/
[ec2-user@ip-172-31-88-21 ~]$ sudo su -
Last login: Thu Dec 5 12:43:25 UTC 2024 on pts/0
[ec2-user@ip-172-31-88-21 ~]$ ll
total 0
[root@ip-172-31-88-21 ~]# systemctl restart jenkins
[root@ip-172-31-88-21 ~]# ll
total 0
[root@ip-172-31-88-21 ~]# cd /var/lib/jenkins
[root@ip-172-31-88-21 jenkins]# ll
total 56
drwxr-xr-x 3 jenkins jenkins 21 Dec 5 12:49 %
-rw-r--r-- 1 jenkins jenkins 1660 Dec 3 12:49 config.xml
-rw-r--r-- 1 jenkins jenkins 156 Dec 5 13:12 hudson.model.UpdateCenter.xml
-rw-r--r-- 1 jenkins jenkins 370 Dec 5 12:52 hudson.plugins.git.GitTool.xml
-rw-r--r-- 1 jenkins jenkins 1680 Dec 5 12:52 identity.key.enc
-rw-r--r-- 1 jenkins jenkins 7 Dec 5 13:12 jenkins.install.InstallUtil.lastExecVersion
-rw-r--r-- 1 jenkins jenkins 7 Dec 5 12:53 jenkins.install.UpgradeWizard.state
-rw-r--r-- 1 jenkins jenkins 184 Dec 5 12:53 jenkins.model.JenkinsLocationConfiguration.xml
-rw-r--r-- 1 jenkins jenkins 171 Dec 5 12:50 jenkins.telemetry.Correlator.xml
drwxr-xr-x 3 jenkins jenkins 23 Dec 5 12:53 jobs
drwxr-xr-x 2 jenkins jenkins 32 Dec 5 12:53 logs
-rw-r--r-- 1 jenkins jenkins 107 Dec 3 13:15 nodeMonitors.xml
drwxr-xr-x 90 jenkins jenkins 8192 Dec 5 12:52 plugins
-rw-r--r-- 1 jenkins jenkins 258 Dec 5 13:00 queue.xml.bak
-rw-r--r-- 1 jenkins jenkins 64 Dec 5 12:50 secret.key
-rw-r--r-- 1 jenkins jenkins 0 Dec 5 12:50 secret.key.not-so-secret
drwxr----- 2 jenkins jenkins 237 Dec 5 12:54 secrets
drwxr-xr-x 2 jenkins jenkins 149 Dec 5 12:52 updates
drwxr-xr-x 2 jenkins jenkins 24 Dec 5 12:50 userContent
drwxr-xr-x 3 jenkins jenkins 59 Dec 5 12:52 users
drwxr-xr-x 3 jenkins jenkins 23 Dec 5 12:54 workspace
[root@ip-172-31-88-21 jenkins]# cd workspace
[root@ip-172-31-88-21 workspace]# ll
total 0
drwxr-xr-x 3 jenkins jenkins 44 Dec 5 12:54 wordpress
[root@ip-172-31-88-21 workspace]# cd wordpress
[root@ip-172-31-88-21 wordpress]# ll
total 4
-rw-r--r-- 1 jenkins jenkins 600 Dec 5 12:54 docker-compose.yml
[root@ip-172-31-88-21 wordpress]#

```

- ⦿ We can also see the docker-compose file through the terminal by using the Jenkins path “/var/www/Jenkins”. Inside this path we can see the “workspace” option there we can see the created jobs. By going inside the job we can see the file.



- ➲ Here I have deployed the wordpress application using the git and Jenkins. I am able to access the wordpress application. This is the front page of wordpress.



- ➲ We can access this application by creating the username and password and need to specify our details.

METHOD-4: DEPLOY WORDPRESS WEB APPLICATION BY USING USERDATA OF EC2 INSTANCE

Userdata is a script which is used to configure specific instructions like installing git, httpd, Jenkins and other packages which are executed while launching EC2 instance. It is also used to automate the process of deploying any applications or starting any servers.

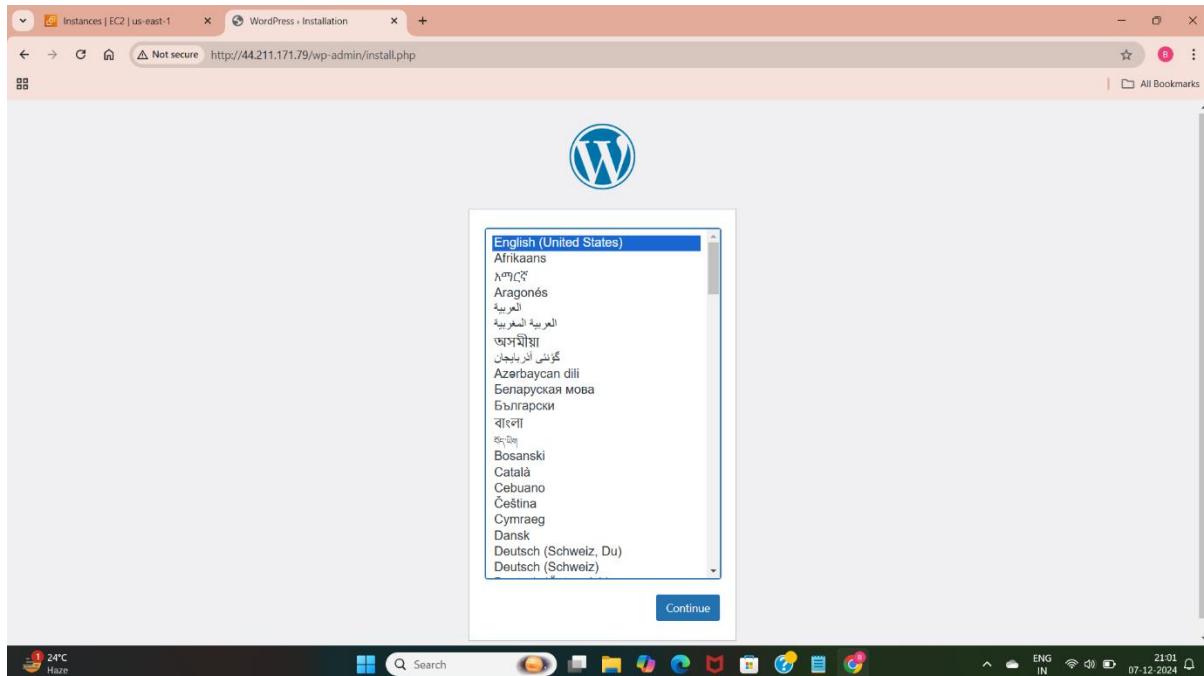
- To use userdata first we need to create a EC2 instance.

The screenshot shows the AWS CloudShell interface with the Instances page open. The left sidebar shows navigation options like Dashboard, EC2 Global View, Events, Instances, Images, Elastic Block Store, Network & Security, and Load Balancing. The main content area displays a table of EC2 instances. One instance, named 'userdata', is highlighted and selected. Its detailed view is shown below, including its Public IPv4 address (44.211.177.79), Private IP DNS name (ip-172-31-84-41.ec2.internal), and VPC ID (vpc-03f5fe3fdcc5d1603). The Networking tab indicates it has one network interface with a private IP of 172.31.84.41.

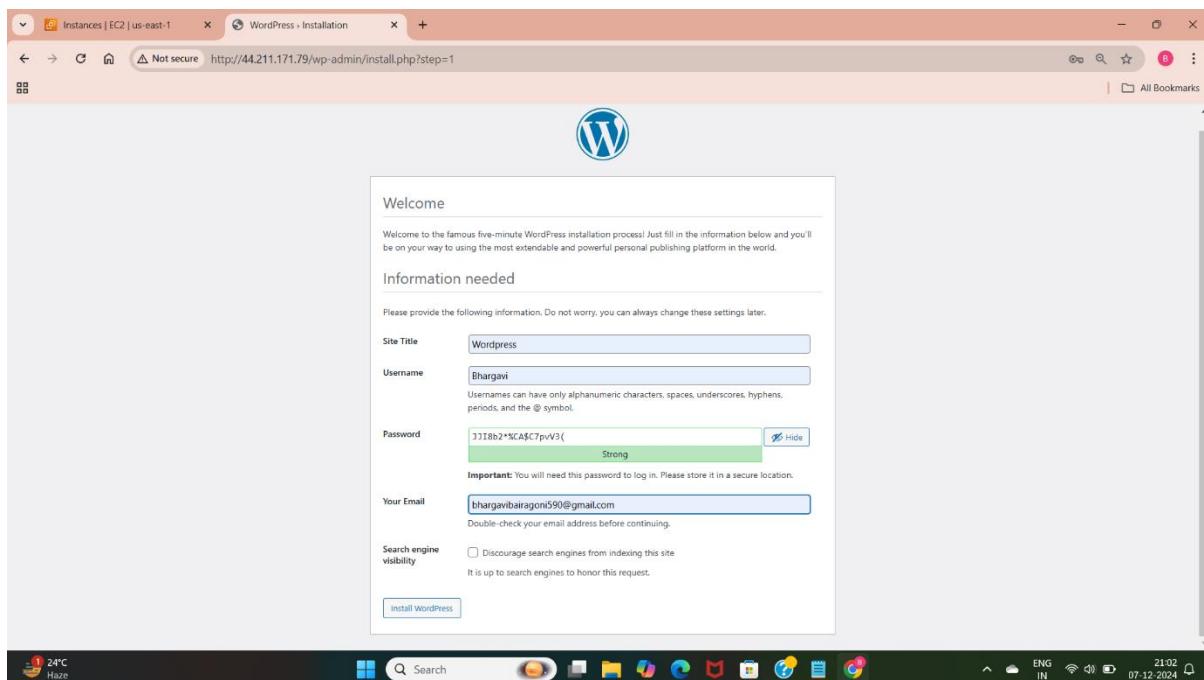
- I have created a EC2 instance with the name “userdata”, application os image as amazon linux with kernel 5.10, allowed the inbound ports with 22 for SSH, 80 for HTTP and 3306 for MYSQL.

The screenshot shows the AWS CloudShell interface with the Launch an instance page open. A modal dialog is displayed for launching a new instance. The 'UserData' field contains a Docker Compose script for WordPress and MySQL. The 'Summary' section shows the following configuration: Number of instances (1), Software Image (Amazon Linux 2 Kernel 5.10 AMI), Virtual server type (t2.micro), Firewall (devops), and Storage (1 volume(s) - 8 GB). A tooltip provides information about the free tier offer. The 'Launch Instance' button is visible at the bottom right of the dialog.

- This is the userdata I have used while creating EC2 instance. This script is used to deploy wordpress application using docker-compose. Here I have installed docker, mysql, docker-compose and given required permissions for the docker-compose.
- Created a directory for docker-compose and moved inside that directory and executed the docker-compose file.



- With the above specified configurations I have successfully deployed wordpress application using userdata script.

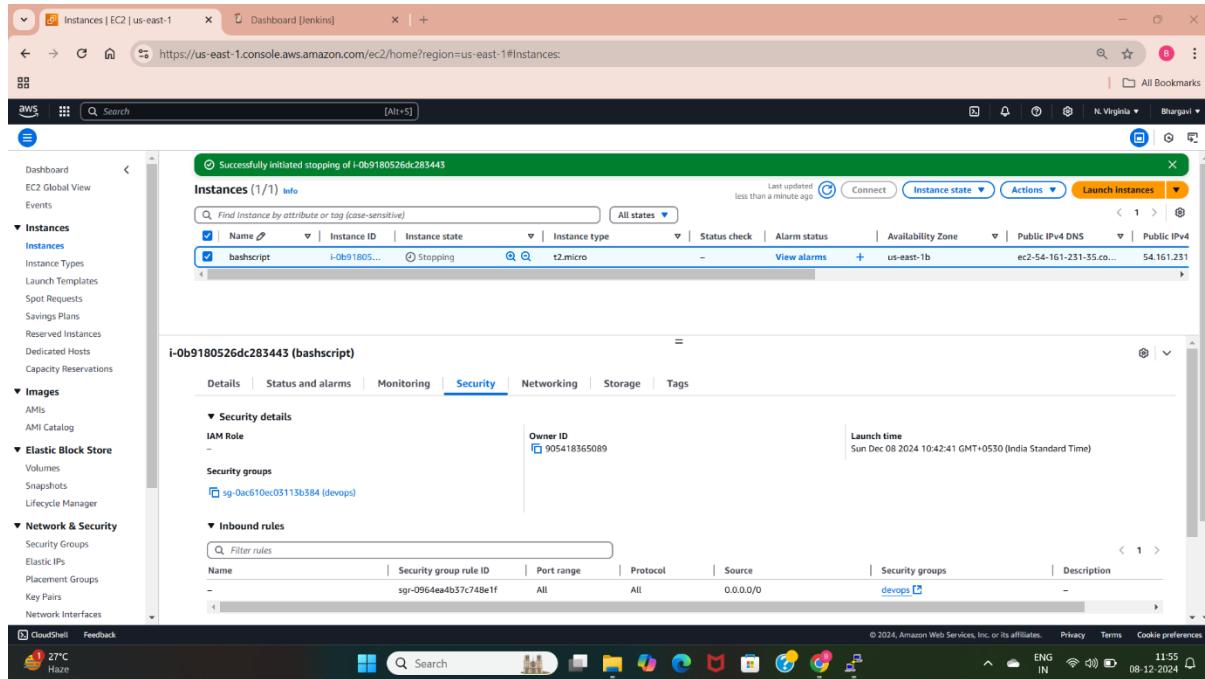


- This is the user login page of the wordpress application, which uses the above mentioned username and password.

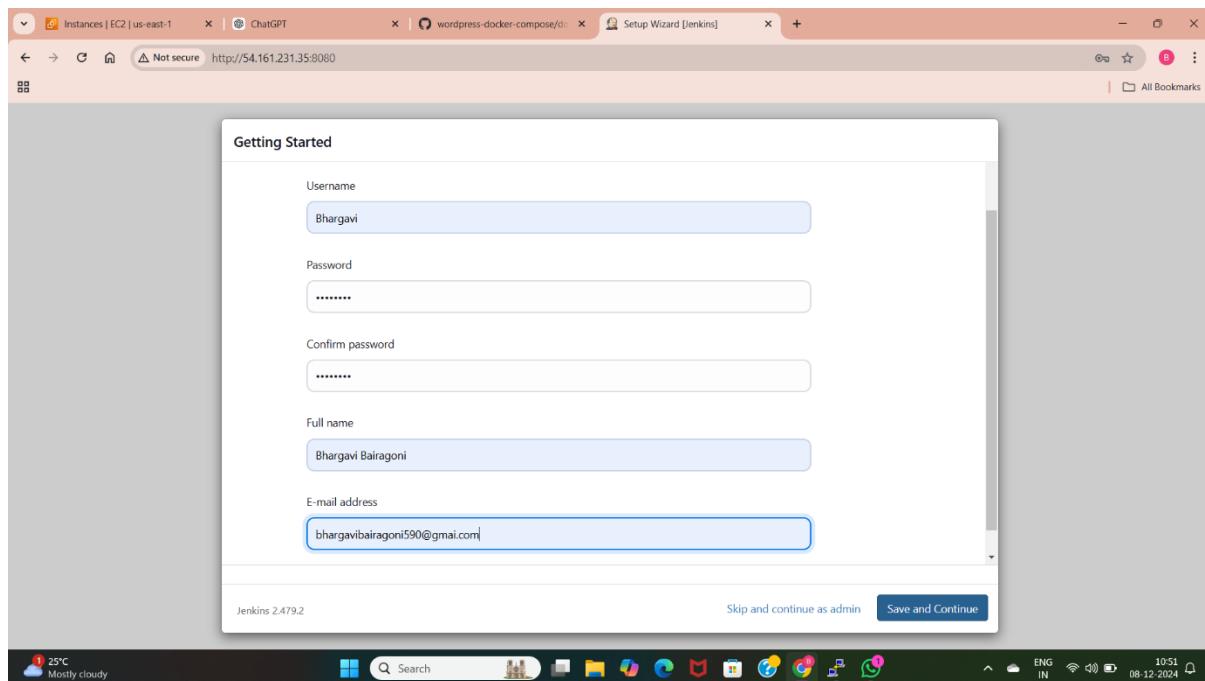
METHOD-5: DEPLOY WORDPRESS WEB APPLICATION BY USING GIT AND JENKINS EXECUTE SHELL (BASH SCRIPT)

Jenkins is a open-source automation server used for continuous integration and continuous delivery/deployment.

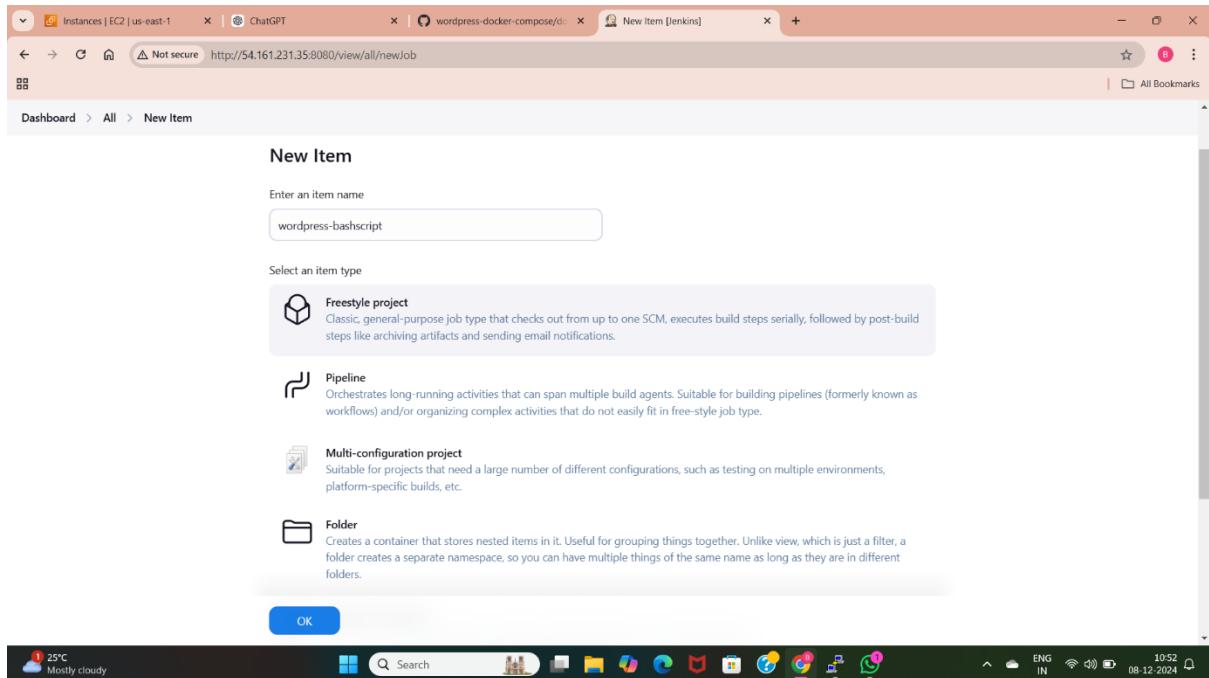
- ♣ To access Jenkins first we need to create a EC2 instance with specific configurations, connect the instance with the terminal and install repositories, tokens related to Jenkins, install Jenkins dependency i.e java17 version, install and start Jenkins.



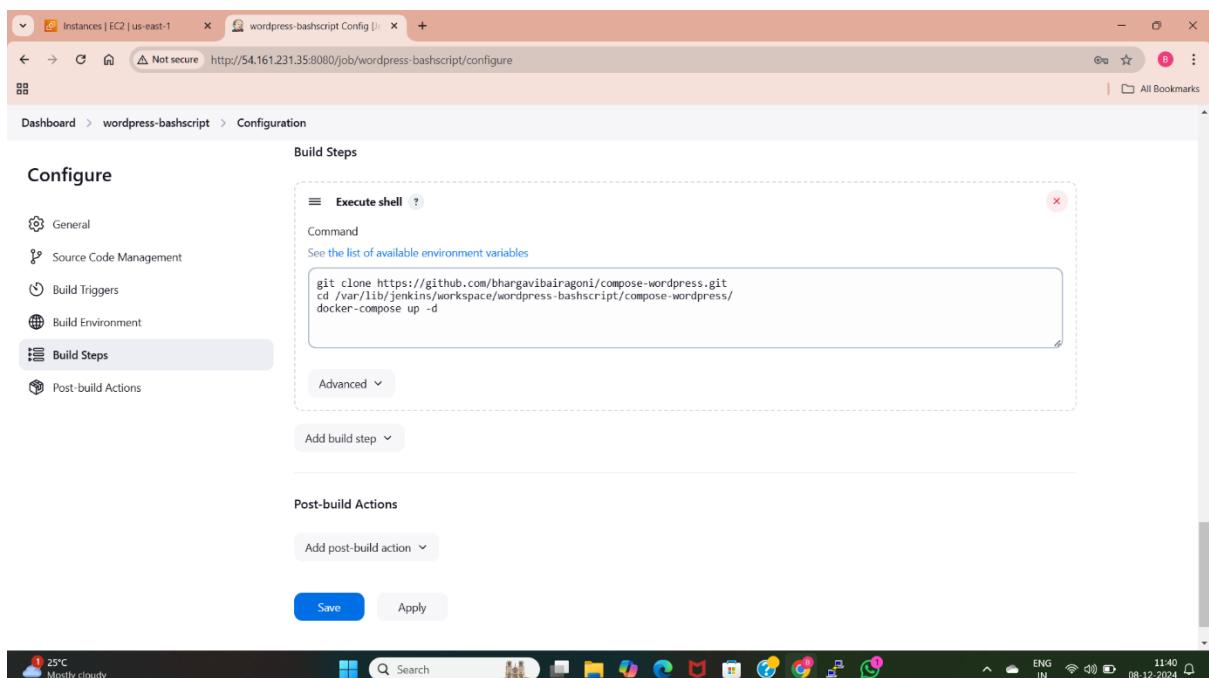
- ♣ After starting Jenkins we need to paste the ip address with Jenkins port number in the browser.
- ♣ Now we can access the Jenkins, install the necessary plugins.



- ♣ Create username and password to access the Jenkins. Here am creating Jenkins account to deploy the application.
- ♣ After entering the required details save and continue, so we can create the jobs in the Jenkins gui.
- ♣ There are two types of jobs in Jenkins
 1. Freestyle
 2. Pipeline



- ♣ Here am creating a Freestyle project with the name “wordpress-bashscript”. So we can deploy application in this job.



- ♣ Here I have downloaded the docker-compose.yml file from the github with the help of the specific command (git clone) and moved to that path where the compose file is present.

- ♣ I have executed the compose file to deploy the application using Jenkins with execute shell(bash script).

The screenshot shows the Jenkins interface for the 'wordpress-bashscript' job. The build history is displayed with the following details:

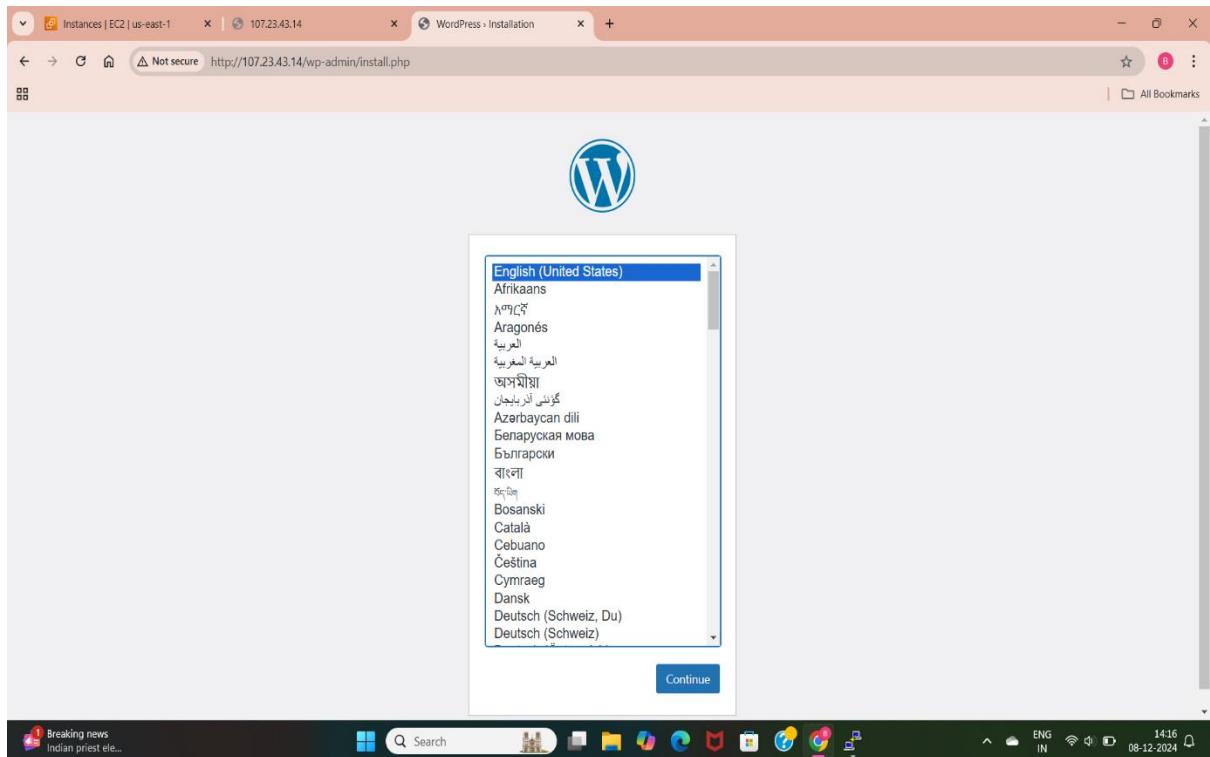
- Last build (#7), 2 min 12 sec ago
- Last failed build (#7), 2 min 12 sec ago
- Last unsuccessful build (#7), 2 min 12 sec ago
- Last completed build (#7), 2 min 12 sec ago

- ♣ After specifying specific file through git, am able to build the project. The build is successful after trying specific number of times.

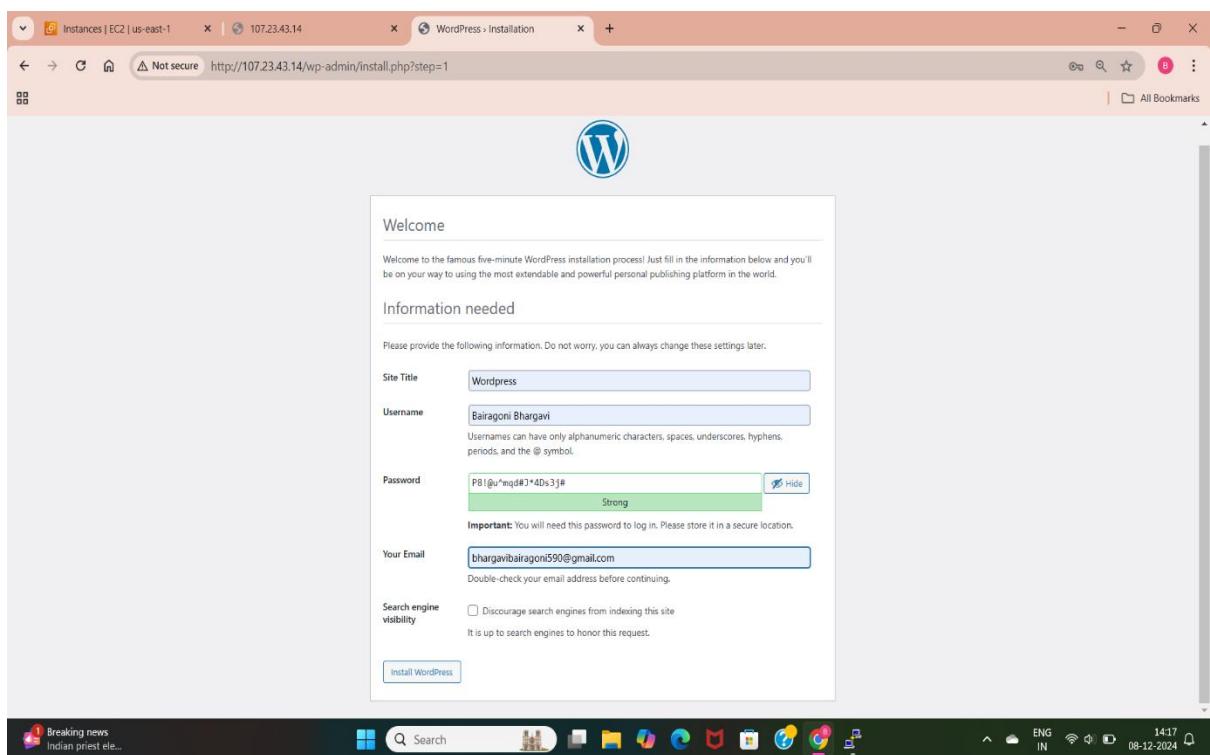
```

root@ip-172-31-89-141:/var/lib/jenkins
total 0
drwxr-xr-x 3 jenkins jenkins 44 Dec  8 05:29 compose-wordpress
[root@ip-172-31-89-141 compose-wordpress]# cd compose-wordpress/
[root@ip-172-31-89-141 compose-wordpress]# ll
total 4
-rw-r--r-- 1 jenkins jenkins 600 Dec  8 05:29 docker-compose.yml
[root@ip-172-31-89-141 compose-wordpress]# ^C
[root@ip-172-31-89-141 compose-wordpress]# rm -rf *
[root@ip-172-31-89-141 compose-wordpress]# ll
total 0
[root@ip-172-31-89-141 compose-wordpress]# cd
[root@ip-172-31-89-141 ~]# cd /var/lib/jenkins
[root@ip-172-31-89-141 jenkins]# cd workspace
[root@ip-172-31-89-141 workspace]# ll
total 0
drwxr-xr-x 3 jenkins jenkins 31 Dec  8 05:29 wordpress-bashscript
[root@ip-172-31-89-141 workspace]# rm -rf *
[root@ip-172-31-89-141 workspace]# ll
total 0
[root@ip-172-31-89-141 workspace]# cd
[root@ip-172-31-89-141 ~]# cd
[root@ip-172-31-89-141 ~]# 
[root@ip-172-31-89-141 ~]# ll
total 0
[root@ip-172-31-89-141 ~]# cd /var/lib/jenkins
[root@ip-172-31-89-141 jenkins]# ll
total 56
drwxr-xr-x 3 jenkins jenkins 21 Dec  8 05:20 %
-rw-r--r-- 1 jenkins jenkins 1660 Dec  8 05:20 config.xml
-rw-r--r-- 1 jenkins jenkins 156 Dec  8 05:20 hudson.model.UpdateCenter.xml
-rw-r--r-- 1 jenkins jenkins 378 Dec  8 05:20 hudson.plugins.git.GitTool.xml
-rw-r--r-- 1 jenkins jenkins 1680 Dec  8 05:21 identityKey
-rw-r--r-- 1 jenkins jenkins 9 Dec  8 05:21 jenkins.install.InstallUtil.lastExecVersion
-rw-r--r-- 1 jenkins jenkins 7 Dec  8 05:21 jenkins.install.UpgradeWizard.state
-rw-r--r-- 1 jenkins jenkins 183 Dec  8 05:21 jenkins.model.JenkinsLocationConfiguration.xml
-rw-r--r-- 1 jenkins jenkins 171 Dec  8 05:20 jenkins.telemetry.Correlator.xml
drwxr-xr-x 3 jenkins jenkins 34 Dec  8 05:22 jobs
drwxr-xr-x 2 jenkins jenkins 32 Dec  8 05:22 logs
-rw-r--r-- 1 jenkins jenkins 1037 Dec  8 05:20 nodeMonitors.xml
drwxr-xr-x 90 jenkins jenkins 8192 Dec  8 05:21 plugins
-rw-r--r-- 1 jenkins jenkins 258 Dec  8 05:21 quicunnel
-rw-r--r-- 1 jenkins jenkins 64 Dec  8 05:20 secret.key
-rw-r--r-- 1 jenkins jenkins 60 Dec  8 05:20 secret.key.not-so-secret
drwx----- 2 jenkins jenkins 295 Dec  8 05:24 secrets
drwxr-xr-x 2 jenkins jenkins 149 Dec  8 05:21 updates
drwxr-xr-x 2 jenkins jenkins 24 Dec  8 05:20 userContent
drwxr-xr-x 3 jenkins jenkins 60 Dec  8 05:21 users
drwxr-xr-x 3 jenkins jenkins 34 Dec  8 05:35 workspace
[root@ip-172-31-89-141 jenkins]# cd workspace/wordpress-bashscript/
```

- ♣ We can able to see the docker-compose.yml file by using the terminal by going into Jenkins path “/var/lib/Jenkins”. In this path we are having a workspace which contains details of our created jobs.
- ♣ I have moved to the created job “wordpress-bashscript” and checked with the compose file. It is downloaded with the help of above command “git clone compose file url”.



- ♣ After successful build step am able to access the deployed application using the public ip of the instance.

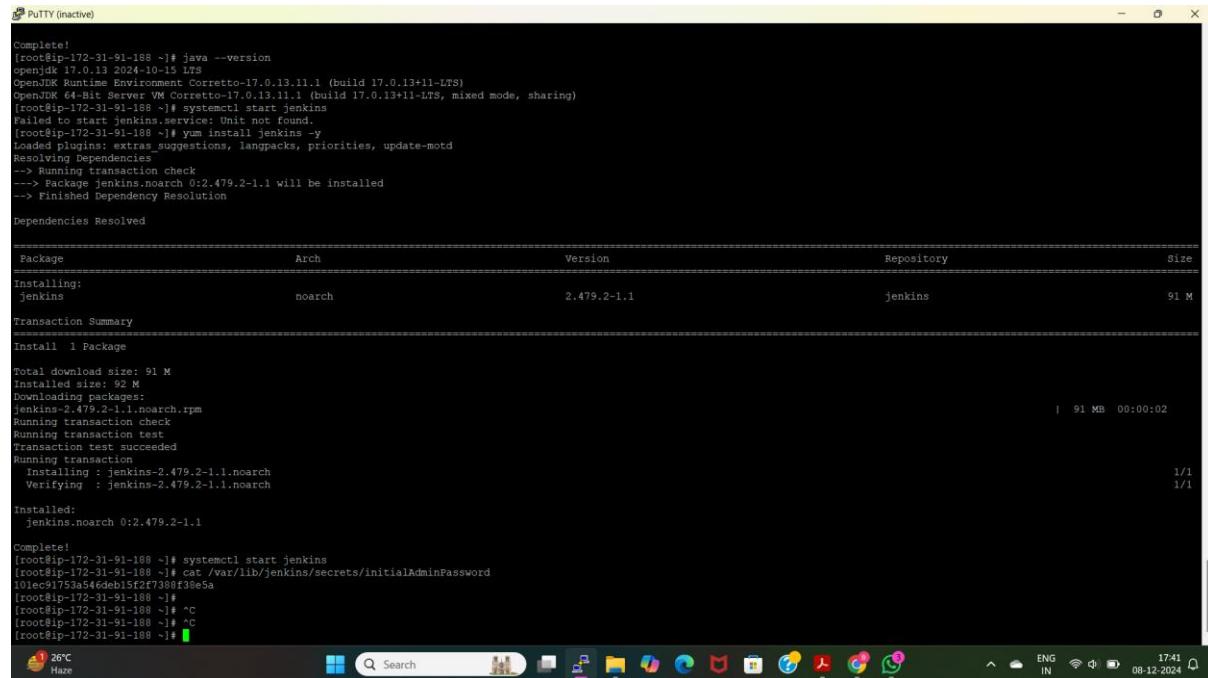


- ♣ Here am able to see the wordpress application dashboard, which is asking me to enter the details of the user like username, password and email.

METHOD-6: DEPLOY WORDPRESS WEB APPLICATION BY USING GIT AND JENKINS EXECUTE SHELL

Pipeline is used to automate the process like build, test, and deploy software applications. It ensures that code changes go through various stages to verify their quality and readiness for production.

- ✓ To use pipeline in Jenkins we need to launch a EC2 instance by allowing port number 8080, install required packages.



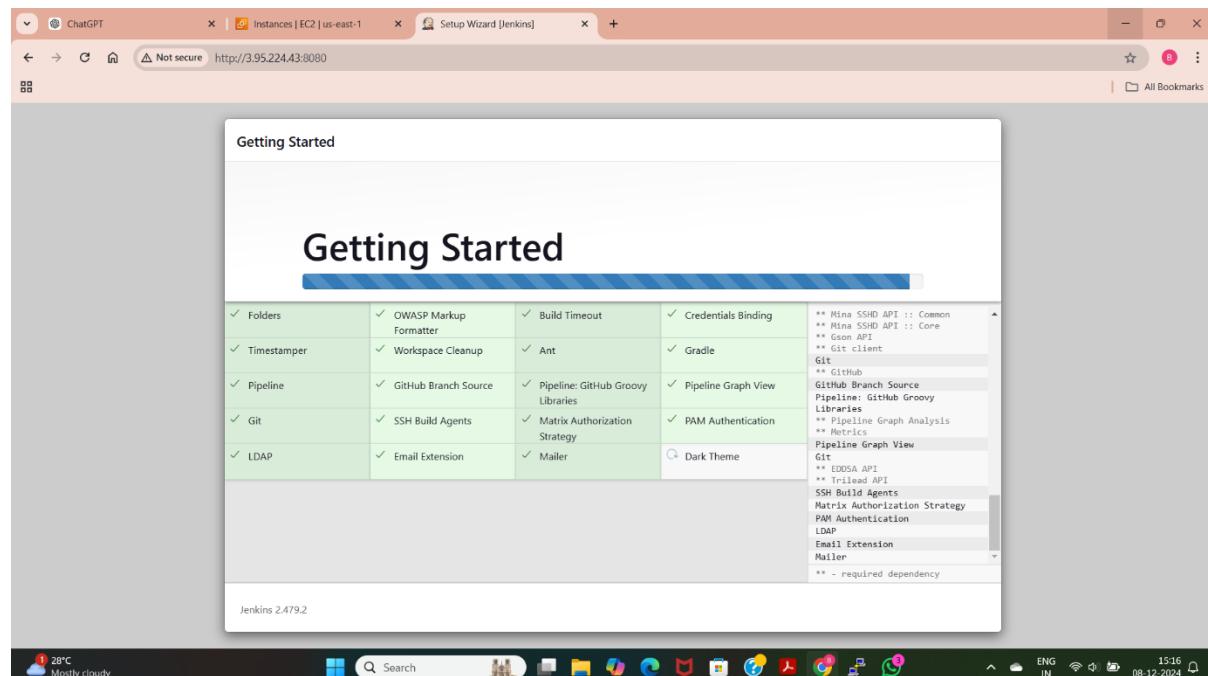
```
Putty (inactive)

Complete!
[root@ip-172-31-91-188 ~]# java --version
openjdk 17.0.13 2024-10-15 LTS
OpenJDK Runtime Environment Corretto-17.0.13.11.1 (build 17.0.13+11-LTS)
OpenJDK 64-Bit Server VM Corretto-17.0.13.11.1 (build 17.0.13+11-LTS, mixed mode, sharing)
[root@ip-172-31-91-188 ~]# systemctl status jenkins
Failed to start jenkins.service: Unit not found.
[root@ip-172-31-91-188 ~]# yum install jenkins -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Resolving Dependencies
--> Running transaction check
--> Package jenkins.noarch 0:2.479.2-1.1 will be installed
--> Finished Dependency Resolution
Dependencies Resolved

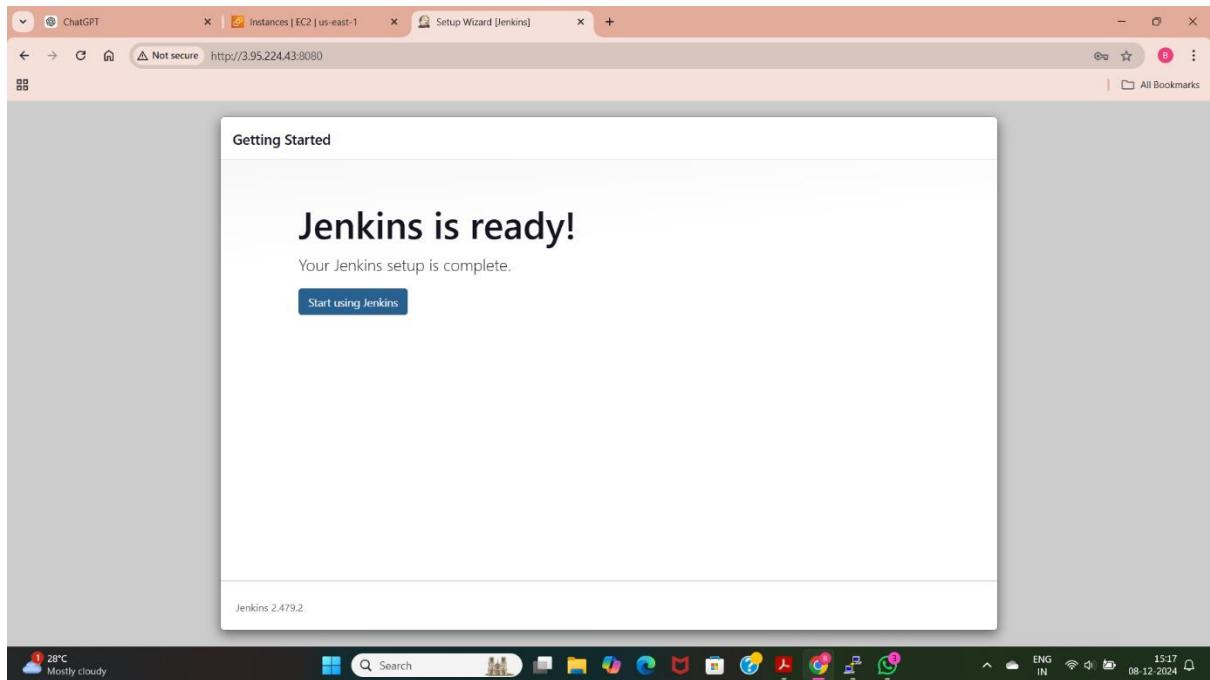
Transaction Summary
=====
Install 1 Package

Total download size: 91 M
Installed size: 92 M
Downloading packages:
jenkins-2.479.2-1.1.noarch.rpm
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : jenkins-2.479.2-1.1.noarch
    Verifying : jenkins-2.479.2-1.1.noarch
  Installed:
    jenkins.noarch 0:2.479.2-1.1

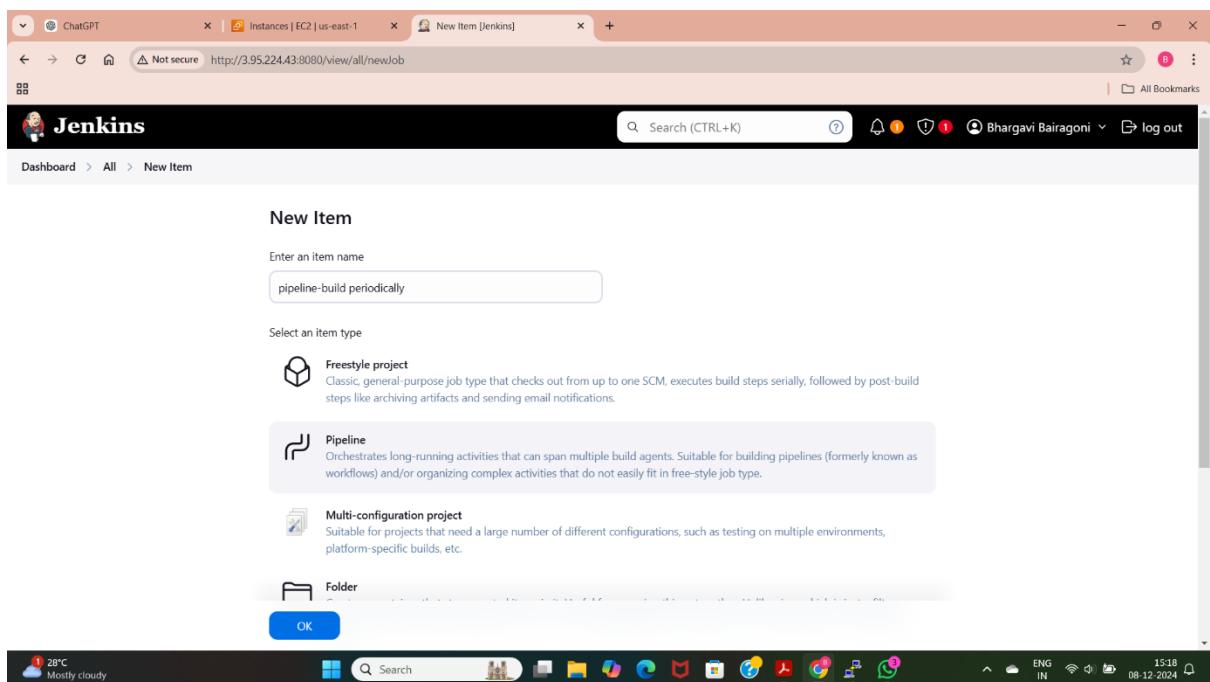
Complete!
[root@ip-172-31-91-188 ~]# systemctl start jenkins
[root@ip-172-31-91-188 ~]# cat /var/lib/Jenkins/secrets/initialAdminPassword
10dec7557556560227308f36e5a
[root@ip-172-31-91-188 ~]#
[root@ip-172-31-91-188 ~]# ^C
[root@ip-172-31-91-188 ~]# ^C
[root@ip-172-31-91-188 ~]# ^C
[root@ip-172-31-91-188 ~]# ^C
```



- ✓ This is the GUI of Jenkins, which is installing required plugins.



- ✓ Now we can able to access the Jenkins after installing required plugins. Now click on the “start using Jenkins”.



- ✓ Now am creating a pipeline job with the name “pipeline-build periodically”.

The screenshot shows the Jenkins dashboard. On the left, there are links for 'New Item', 'Build History', 'Manage Jenkins', and 'My Views'. A 'Build Queue' section indicates 'No builds in the queue'. Below it is a 'Build Executor Status' section showing '0/2' executors. The main area displays a table with columns: S, W, Name, Last Success, Last Failure, and Last Duration. One job, 'pipeline-build periodically', is listed with a green checkmark icon, indicating it's successful. The 'Last Success' is 20 min ago, 'Last Failure' is 26 min ago, and 'Last Duration' is 33 sec. There is also a 'Pipeline' button and a 'Add description' link.

- ✓ This is the dashboard of Jenkins which shows the created jobs and we can also install the plugins, specify configurations by using Manage Jenkins. We can also check with the build history.
- ✓ By going inside the above job we can deploy our application and check with the continuous integration and continuous delivery/deployment.

The screenshot shows the 'Configuration' page for the 'pipeline-build periodically' job. On the left, there are tabs for 'General', 'Advanced Project Options', and 'Pipeline'. The 'Pipeline' tab is selected. The 'Definition' dropdown is set to 'Pipeline script'. The 'Script' text area contains the following Groovy pipeline script:

```

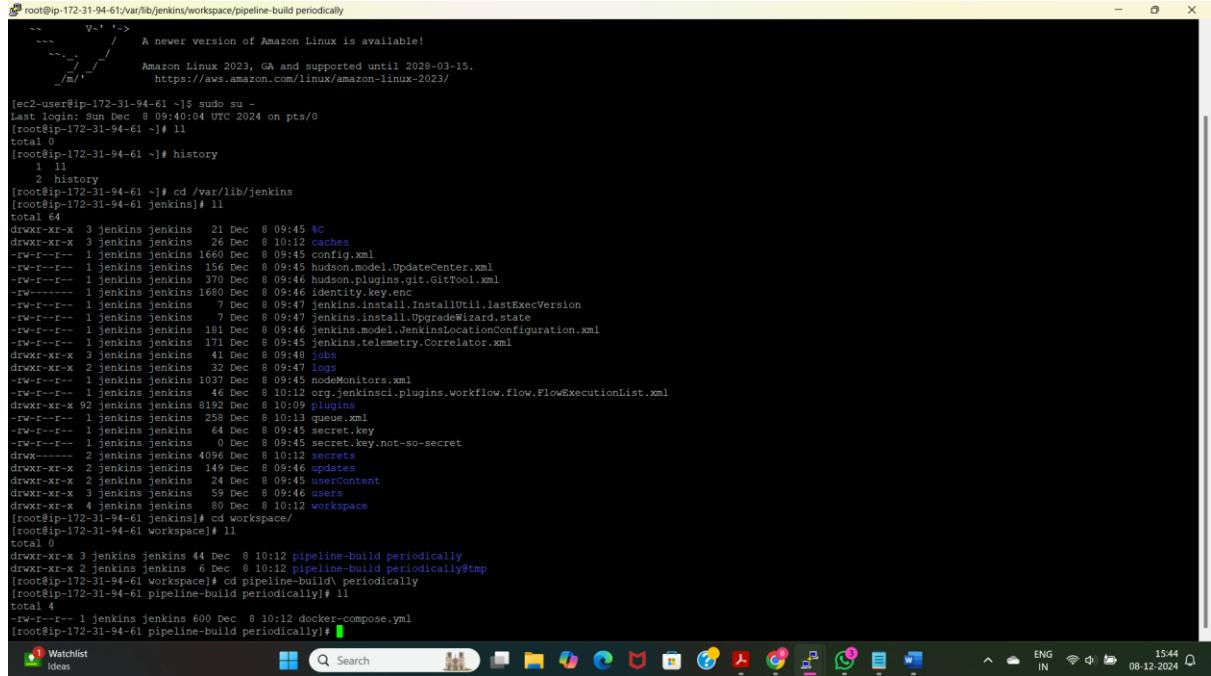
1 ~ pipeline {
2   ~   agent any
3   ~   stages {
4     ~     stage ("code") {
5       ~       steps {
6         ~         git 'https://github.com/bhargavbairagoni/compose-wordpress.git'
7       ~     }
8     ~     stage ("run") {
9       ~       sh 'docker-compose up -d'
10      ~     }
11    ~   }
12  ~ }
13 }

```

Below the script, there is a checkbox for 'Use Groovy Sandbox' and a 'Pipeline Syntax' link. At the bottom, there are 'Save' and 'Apply' buttons.

- ✓ This is the script for pipeline using some syntax, which is taking the code from github and executing compose file with the specific command.
- ✓ Jenkins is having two types of pipeline scripts
 1. Declarative pipeline
 2. Scripted pipeline
- ✓ Pipeline syntax includes pipeline
 1. agent: Defines where the pipeline runs (e.g., any, none, or a specific node).

2. stages: Contains multiple stage blocks, each representing a build phase (like code, build, test).
3. steps: Instructions for what to do in a stage (e.g., sh, git, echo).

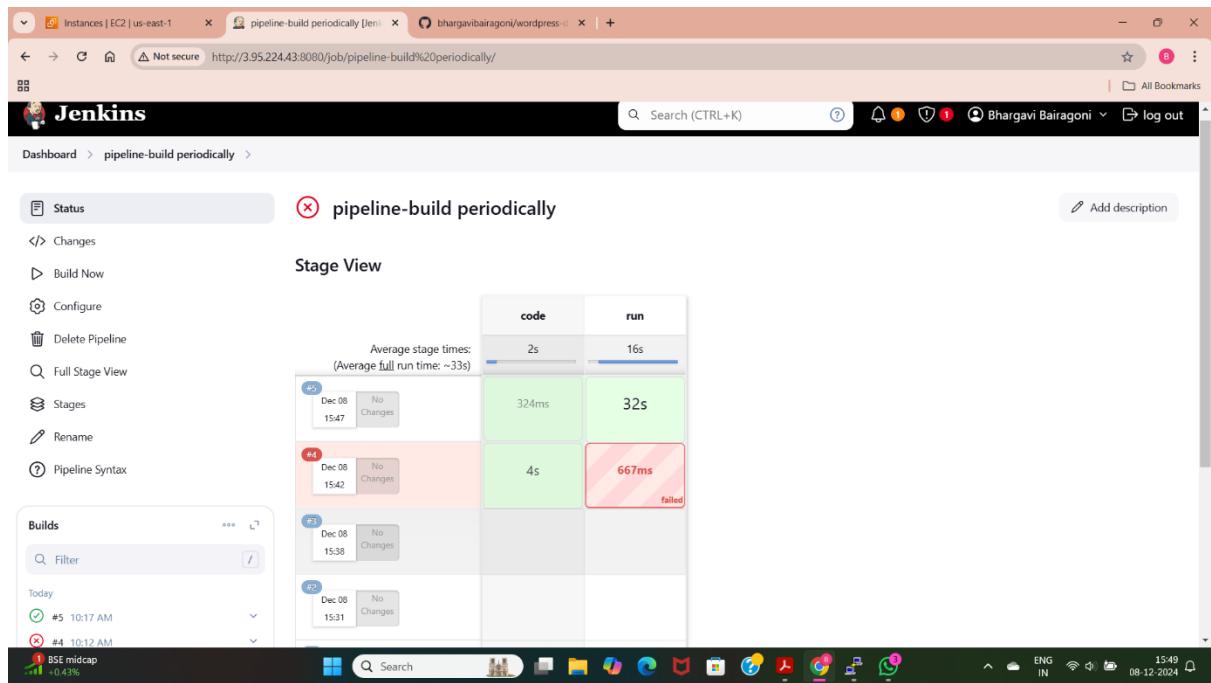


```

root@ip-172-31-94-61:~# sudo su -
Last login: Sun Dec  8 09:40:04 UTC 2024 on pts/0
[ec2-user@ip-172-31-94-61 ~]$ ll
total 0
[root@ip-172-31-94-61 ~]# history
1 11
2 history
[root@ip-172-31-94-61 ~]# cd /var/lib/jenkins
[root@ip-172-31-94-61 jenkins]# ll
total 64
drwxr-xr-x 3 jenkins jenkins 21 Dec  8 09:45 .
drwxr-xr-x 3 jenkins jenkins 25 Dec  8 10:12 cache
-rw-r--r-- 1 jenkins jenkins 1660 Dec  8 09:45 config.xml
-rw-r--r-- 1 jenkins jenkins 156 Dec  8 09:45 hudson.model.UpdateCenter.xml
-rw-r--r-- 1 jenkins jenkins 370 Dec  8 09:46 hudson.plugins.git.GitTool.xml
-rw-r----- 1 jenkins jenkins 1680 Dec  8 09:46 identity.key.enc
-rw-r--r-- 1 jenkins jenkins 7 Dec  8 09:47 jenkins.install.InstallUtil.lastExecVersion
-rw-r--r-- 1 jenkins jenkins 7 Dec  8 09:47 jenkins.install.UpgradeWizard.state
-rw-r--r-- 1 jenkins jenkins 181 Dec  8 09:46 jenkins.model.JenkinsLocationConfiguration.xml
-rw-r--r-- 1 jenkins jenkins 171 Dec  8 09:45 jenkins.telemetry.Correlator.xml
drwxr-xr-x 3 jenkins jenkins 28 Dec  8 09:47 jobs
-rw-r--r-- 1 jenkins jenkins 1037 Dec  8 09:45 nodeMonitors.xml
-rw-r--r-- 1 jenkins jenkins 46 Dec  8 10:12 org.jenkinsci.plugins.workflow.flow.FlowExecutionList.xml
drwxr-xr-x 92 jenkins jenkins 8192 Dec  8 10:09 plugins
-rw-r--r-- 1 jenkins jenkins 258 Dec  8 10:13 queue.xml
-rw-r--r-- 1 jenkins jenkins 64 Dec  8 09:45 secret.key
-rw-r--r-- 1 jenkins jenkins 0 Dec  8 09:45 secret.key.not-so-secret
drwxr----- 2 jenkins jenkins 4096 Dec  8 10:12 secrets
drwxr-xr-x 2 jenkins jenkins 149 Dec  8 09:46 updates
drwxr-xr-x 2 jenkins jenkins 24 Dec  8 09:45 userContent
drwxr-xr-x 2 jenkins jenkins 59 Dec  8 09:46 users
drwxr-xr-x 4 jenkins jenkins 80 Dec  8 10:12 workspace
[root@ip-172-31-94-61 jenkins]# cd workspace/
[root@ip-172-31-94-61 workspace]# ll
total 0
drwxr-xr-x 3 jenkins jenkins 44 Dec  8 10:12 pipeline-build periodically
drwxr-xr-x 2 jenkins jenkins 6 Dec  8 10:12 pipeline-build periodically@tmp
[root@ip-172-31-94-61 workspace]# cd pipeline-build@tmp
[root@ip-172-31-94-61 pipeline-build periodically]# ll
total 0
-rw-r--r-- 1 jenkins jenkins 600 Dec  8 10:12 docker-compose.yml
[root@ip-172-31-94-61 pipeline-build periodically]#

```

- ✓ This terminal shows that the compose file is stored in the Jenkins workspace. We can check with the applications, files that are imported into Jenkins pipeline or freestyle.



The screenshot shows a Jenkins pipeline named "pipeline-build periodically". The pipeline has two stages: "code" and "run". The "code" stage has an average stage time of 2s and an average full run time of ~33s. The "run" stage has an average stage time of 16s. The pipeline has four builds listed:

Build	Dec 08	15:47	324ms	32s
#5	Dec 08	15:42	4s	667ms failed
#4	Dec 08	15:38		
#3	Dec 08	15:31		

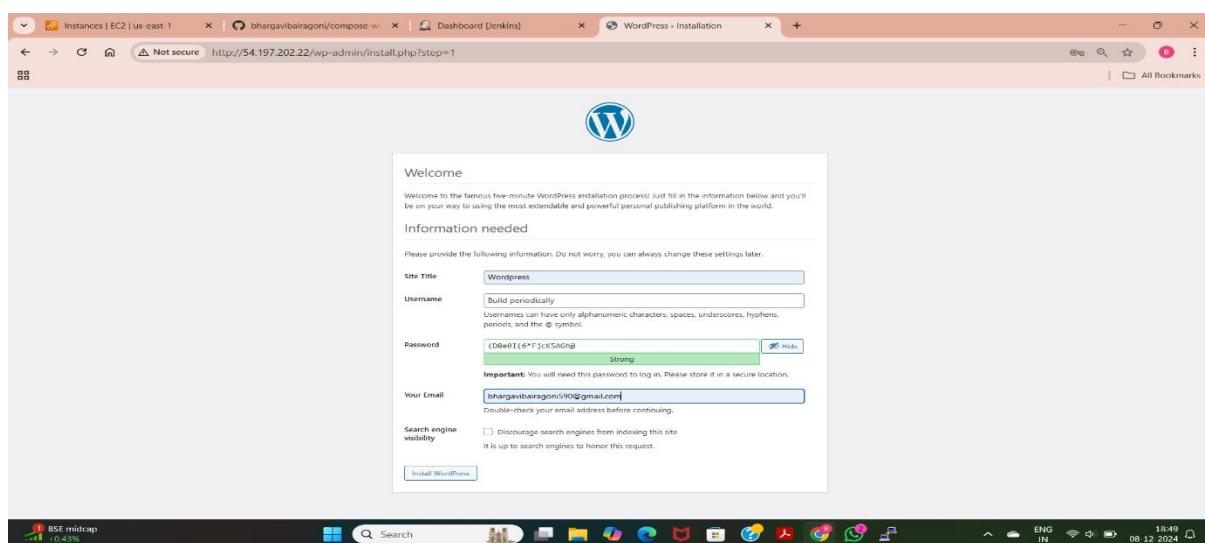
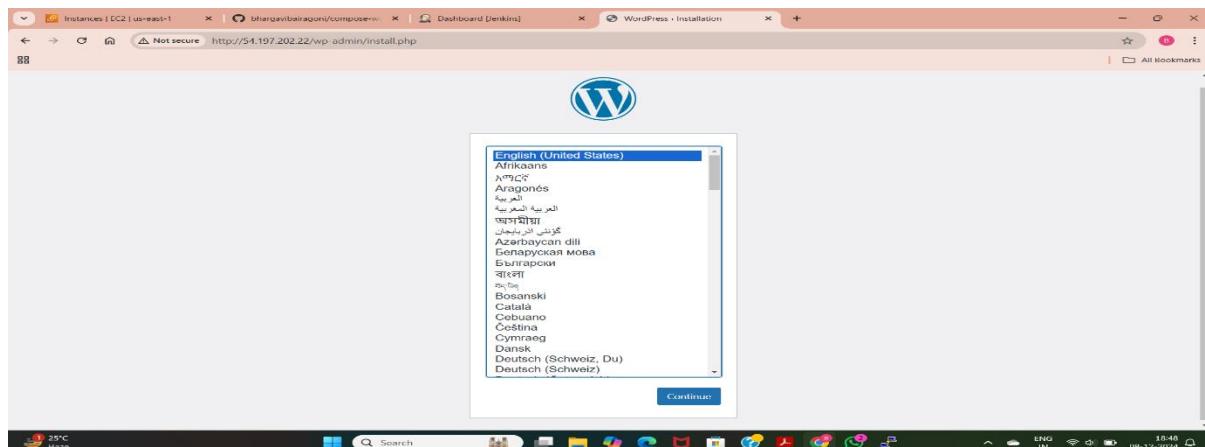
- ✓ I have successfully built the pipeline. Here I have used a plugin "pipeline stage view" to see the stages of the pipeline. Here am used two stages.
 1. Code: Here I have taken the code from github.
 2. Run: Here I have used execute command to execute the application.

```

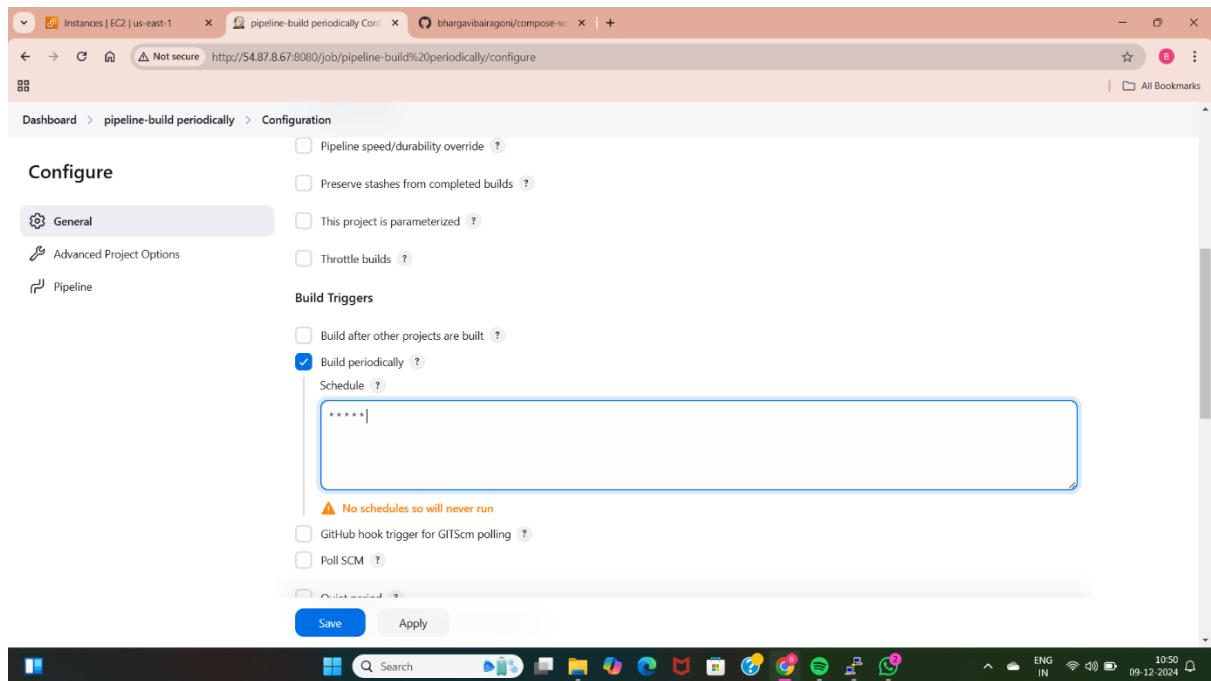
root@ip-172-31-91-188:/var/lib/jenkins/workspace/pipeline-build periodically
lwxrwxrwx 1 root root 30 Nov 13 21:48 sendmail -> /etc/alternatives/mta-sendmail
drwxr-xr-x 2 root root 6 Nov 13 21:48 sendmail.postfix -> ../sbin/sendmail.postfix
drwxr-xr-x 1 root root 6 Nov 13 21:48 sendmail
drwxr-xr-x 14 root root 105 Nov 13 21:48 systemctl
drwxr-xr-x 2 root root 4096 Nov 13 21:48 sysctl.d
drwxr-xr-x 2 root root 29 Nov 13 21:48 udev
drwxr-xr-x 2 root root 57 Dec 8 11:53 yum-plugins
[root@ip-172-31-91-188 lib]# jenkins
-bash: cd: jenkins: No such file or directory
[root@ip-172-31-91-188 lib]# cd /var/lib/jenkins
[root@ip-172-31-91-188 jenkins]# ll
total 56
drwxr-xr-x 3 jenkins jenkins 21 Dec 8 11:53 %
drwxr-xr-x 3 jenkins jenkins 26 Dec 8 12:05 caches
-rw-r--r-- 1 jenkins jenkins 161 Dec 8 13:18 config.xml
-rw-r--r-- 1 jenkins jenkins 156 Dec 8 13:18 jenkins.model.UpdateCenter.xml
-rw-r--r-- 1 jenkins jenkins 370 Dec 8 11:55 hudson.plugins.git.GitTool.xml
-rw----- 1 jenkins jenkins 1680 Dec 8 11:55 identity.key.enc
-rw-r--r-- 1 jenkins jenkins 7 Dec 8 13:18 jenkins.install.InstallUtil.lastExecVersion
-rw-r--r-- 1 jenkins jenkins 7 Dec 8 11:56 Jenkins.install.UpgradeWizard.state
-rw-r--r-- 1 jenkins jenkins 187 Dec 8 11:56 Jenkins.model.JenkinsLocationConfiguration.xml
-rw-r--r-- 1 jenkins jenkins 171 Dec 8 11:53 jenkins.telemetry.Correlator.xml
drwxr-xr-x 3 jenkins jenkins 41 Dec 8 11:58 jobs
drwxr-xr-x 2 jenkins jenkins 32 Dec 8 11:55 logs
-rw-r--r-- 1 jenkins jenkins 1037 Dec 8 13:18 nodeMonitors.xml
-rw-r--r-- 1 jenkins jenkins 1 Dec 8 11:55 nodeMonitors.jenkinsci.plugins.workflow.flow.FlowExecutionList.xml
drwxr-xr-x 90 jenkins jenkins 8192 Dec 8 11:55 plugins
-rw-r--r-- 1 jenkins jenkins 64 Dec 8 11:53 secret.key
-rw-r--r-- 1 jenkins jenkins 0 Dec 8 11:53 secrets
drwxr-xr-x 2 jenkins jenkins 237 Dec 8 12:05 secrets
drwxr-xr-x 1 jenkins jenkins 19 Dec 8 11:53 userContent
drwxr-xr-x 2 jenkins jenkins 24 Dec 8 11:53 userContent
drwxr-xr-x 3 jenkins jenkins 60 Dec 8 11:56 users
drwxr-xr-x 4 jenkins jenkins 80 Dec 8 12:05 workspace
[root@ip-172-31-91-188 jenkins]# /workspace
-bash: /workspace: No such file or directory
[root@ip-172-31-91-188 jenkins]# cd workspace
[root@ip-172-31-91-188 workspace]# ll
total 0
drwxr-xr-x 1 jenkins jenkins 44 Dec 8 12:05 pipeline-build periodically
drwxr-xr-x 2 jenkins jenkins 6 Dec 8 12:06 pipeline-build periodically@tmp
[root@ip-172-31-91-188 workspace]# cd pipeline-build periodically
[root@ip-172-31-91-188 pipeline-build periodically]# ll
total 4
-rw-r--r-- 1 jenkins jenkins 600 Dec 8 12:05 docker-compose.yml
[root@ip-172-31-91-188 pipeline-build periodically]#

```

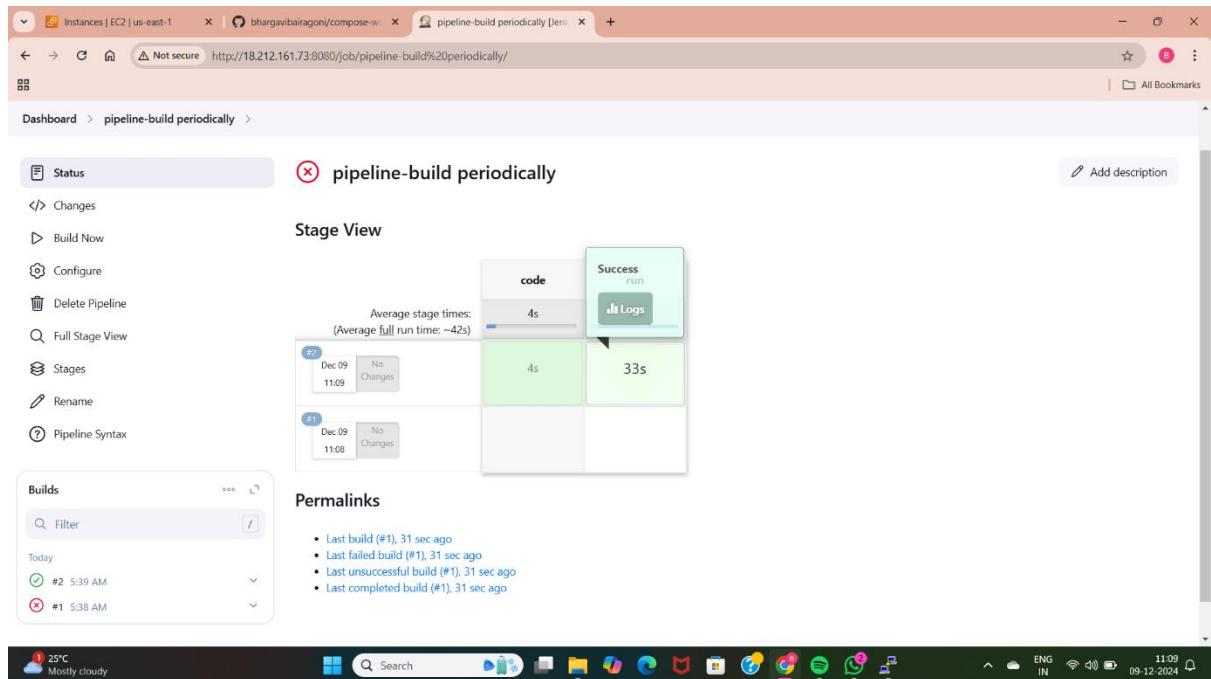
- ✓ We can see the compose file which is used in the Jenkins through the Jenkins path. Here I have used the docker-compose.yml file.



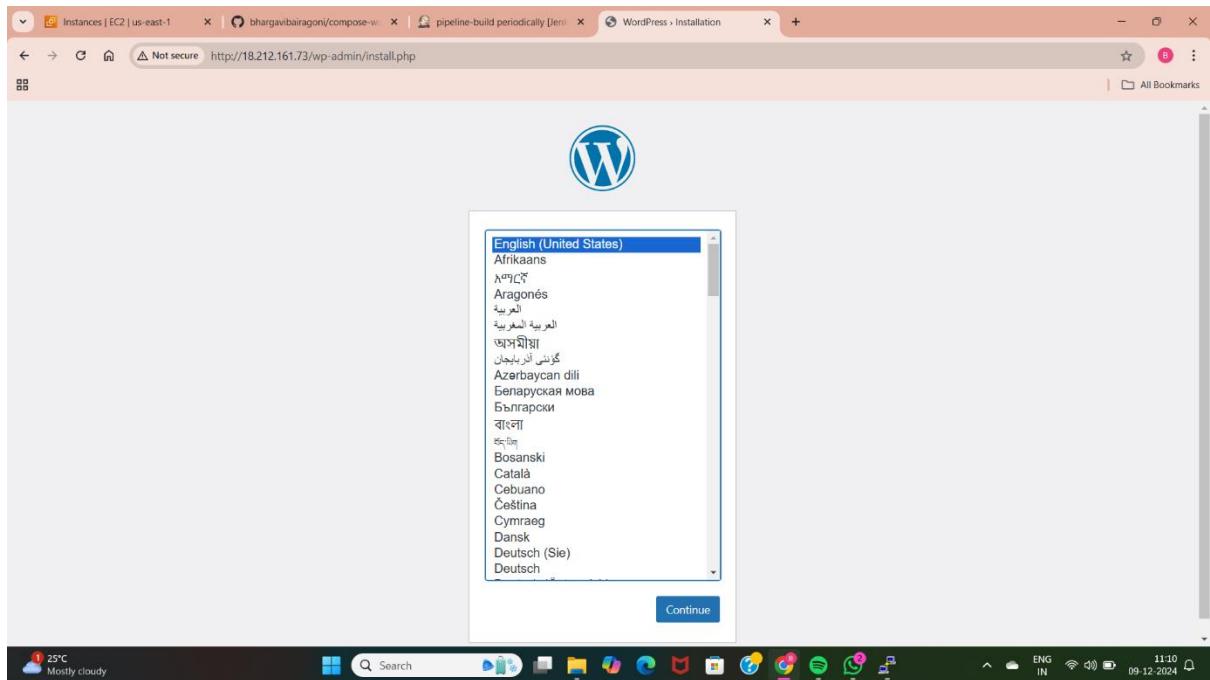
- ✓ Now by pasting the public ip address of instance am able to access the wordpress application which is deployed using build periodically.



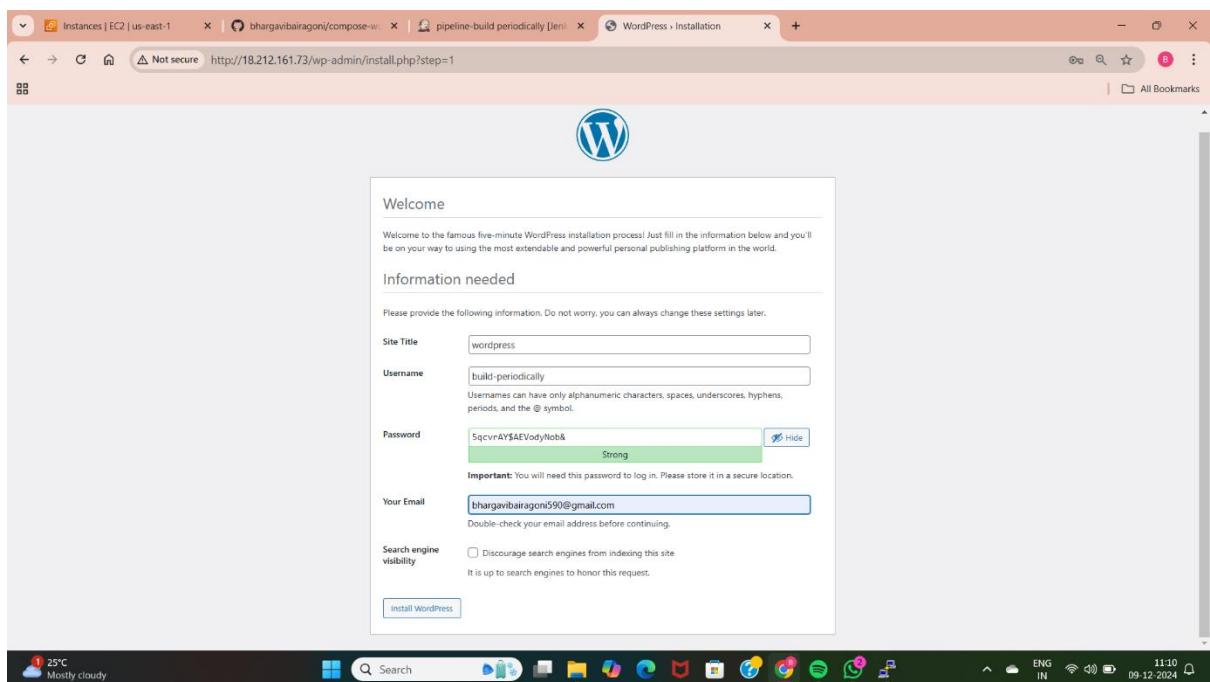
- ✓ Here am using build periodically method to trigger the builds, am used the crontab syntax (minute hour day month week of the day) * * * * * so that the builds take place for every minute with or without any changes in the github code.



- ✓ After using build periodically trigger method we can able to check with the builds. Here am getting the builds for every minute.



- ✓ By copying the public Ip address of the instance we can access the application, which is having the Jenkins configuration installed and set in that instance.



- ✓ We can able to access the wordpress application once the build is successful.
- ✓ After the build is successful am able to see the wordpress application is deployed successfully.
- ✓ Now change the build trigger method and observe the changes in the build functionality of the code.

The screenshot shows the Jenkins 'pollscm' configuration page. Under 'Build Triggers', the 'Poll SCM' option is selected. A warning message at the bottom states: '⚠️ Do you really mean "every minute" when you say "*****"? Perhaps you meant "H * * * *" to poll once per hour'. Below this, it says 'Would last have run at Monday, December 9, 2024 at 5:58:03 AM Coordinated Universal Time; would next run at Monday, December 9, 2024 at 5:58:03 AM Coordinated Universal Time.' Other trigger options like 'Build after other projects are built', 'Build periodically', and 'GitHub hook trigger for GITScm polling' are also listed.

- ✓ Here am using the build trigger method “poll scm” which is used to trigger the changes only when we update the github code.
- ✓ Here am using a crontab syntax(* * * * *) which enables the build for every minute when there are any changes in the code.

```

root@ip-172-31-27-121:/var/lib/jenkins/workspace/pollscm
total 60
drwx--xr-x  1 jenkins jenkins  7 Dec  9 05:57 jenkins.install.InstallUtil.lastExecVersion
-rw-r--r--  1 jenkins jenkins  7 Dec  9 05:57 jenkins.install.UpgradeWizard.state
-rw-r--r--  1 jenkins jenkins 184 Dec  9 05:57 jenkins.model.JenkinsLocationConfiguration.xml
-rw-r--r--  1 jenkins jenkins 184 Dec  9 05:57 jenkins.telemetry.correlator.xml
drwx--xr-x  3 jenkins jenkins 21 Dec  9 05:57 jobs
drwx--xr-x  2 jenkins jenkins 32 Dec  9 05:56 logs
-rw-r--r--  1 jenkins jenkins 1037 Dec  9 05:54 nodeMonitors.xml
drwx--xr-x  90 jenkins jenkins 8192 Dec  9 05:56 plugins
-rw-r--r--  1 jenkins jenkins 64 Dec  9 05:54 secret.key
-rw-r--r--  1 jenkins jenkins  0 Dec  9 05:54 secret.key.not-so-secret
drwx----- 2 jenkins jenkins 162 Dec  9 05:57 secrets
drwx--xr-x  2 jenkins jenkins 149 Dec  9 05:56 updates
drwx--xr-x  2 jenkins jenkins 24 Dec  9 05:54 userContent
drwx--xr-x  3 jenkins jenkins 59 Dec  9 05:56 users
[root@ip-172-31-27-121 jenkins]# ll
total 60
drwx--xr-x  3 jenkins jenkins 21 Dec  9 05:54 %C
drwx--xr-x  3 jenkins jenkins 26 Dec  9 06:00 caches
-rw-r--r--  1 jenkins jenkins 1660 Dec  9 05:54 config.xml
-rw-r--r--  1 jenkins jenkins 156 Dec  9 05:54 hudson.model.UpdateCenter.xml
-rw-r--r--  1 jenkins jenkins 370 Dec  9 05:56 hudson.plugins.git.GitTool.xml
-rw-r----- 1 jenkins jenkins 1680 Dec  9 05:56 identity.key.enc
-rw-r--r--  1 jenkins jenkins 7 Dec  9 05:57 jenkins-install.InstallUtil.lastExecVersion
-rw-r--r--  1 jenkins jenkins 184 Dec  9 05:57 jenkins-install.UpgradeWizard.state
-rw-r--r--  1 jenkins jenkins 184 Dec  9 05:57 jenkins.model.JenkinsLocationConfiguration.xml
-rw-r--r--  1 jenkins jenkins 171 Dec  9 05:54 jenkins.telemetry.Correlator.xml
drwx--xr-x  3 jenkins jenkins 21 Dec  9 05:57 jobs
drwx--xr-x  2 jenkins jenkins 32 Dec  9 05:56 logs
-rw-r--r--  1 jenkins jenkins 1037 Dec  9 05:54 nodeMonitors.xml
-rw-r--r--  1 jenkins jenkins 46 Dec  9 06:00 org.jenkinsci.plugins.workflow.flow.FlowExecutionList.xml
drwx--xr-x  92 jenkins jenkins 8192 Dec  9 05:59 plugins
-rw-r--r--  1 jenkins jenkins 258 Dec  9 06:00 queue.xml
-rw-r--r--  1 jenkins jenkins 64 Dec  9 05:54 secret.key
-rw-r--r--  1 jenkins jenkins  0 Dec  9 05:54 secret.key.not-so-secret
drwx----- 2 jenkins jenkins 295 Dec  9 06:00 secrets
drwx--xr-x  2 jenkins jenkins 149 Dec  9 05:56 updates
drwx--xr-x  2 jenkins jenkins 24 Dec  9 05:54 userContent
drwx--xr-x  3 jenkins jenkins 59 Dec  9 05:56 users
drwx--xr-x  4 jenkins jenkins 40 Dec  9 06:00 workspace
[root@ip-172-31-27-121 jenkins]# cd workspace
[root@ip-172-31-27-121 workspace]# ll
total 0
drwx--r-x  3 jenkins jenkins 44 Dec  9 06:00 pollscm
drwx--xr-x  2 jenkins jenkins  6 Dec  9 06:00 pollscm@tmp
[root@ip-172-31-27-121 workspace]# cd pollscm
[root@ip-172-31-27-121 pollscm]# ll
total 4
-rw-r--r--  1 jenkins jenkins 600 Dec  9 06:00 docker-compose.yml
[root@ip-172-31-27-121 pollscm]#

```

- ✓ We can also check with the Jenkins path that our github code is stored in the created job with the name “pollscm”
- ✓ We can also find the details of the builds in the workspace of the Jenkins.

```
root@ip-172-31-27-121:/var/lib/jenkins/workspace/pollscm
-rw-r--r--  1 jenkins jenkins 1037 Dec  9 05:54 nodeMonitors.xml
-rw-r--r--  1 jenkins jenkins  46 Dec  9 06:00 org.jenkinsci.plugins.workflow.flow.FlowExecutionList.xml
drwxr-xr-x  2 jenkins jenkins 8192 Dec  9 05:59 plugins
-rw-r--r--  1 jenkins jenkins 258 Dec  9 06:00 queue.xml
-rw-r--r--  1 jenkins jenkins  48 Dec  9 05:54 secret.key
-rw-r--r--  1 jenkins jenkins  0 Dec  9 05:54 secret.key.not-so-secret
drwx----- 2 jenkins jenkins 295 Dec  9 06:00 secrets
drwxr-xr-x  2 jenkins jenkins 149 Dec  9 05:56 updates
drwxr-xr-x  2 jenkins jenkins  24 Dec  9 05:54 userContent
drwxr-xr-x  3 jenkins jenkins  59 Dec  9 05:56 users
drwxr-xr-x  4 jenkins jenkins  40 Dec  9 06:00 workspace
[root@ip-172-31-27-121 jenkins]# cd workspace/
[root@ip-172-31-27-121 workspace]# ll
total 0
drwxr-xr-x 3 jenkins jenkins 44 Dec  9 06:00 pollscm
drwxr-xr-x 2 jenkins jenkins  6 Dec  9 06:00 pollscm@tmp
[root@ip-172-31-27-121 workspace]# cd pollscm
[root@ip-172-31-27-121 pollscm]# ll
total 4
-rw-r--r-- 1 jenkins jenkins 600 Dec  9 06:00 docker-compose.yml
[root@ip-172-31-27-121 pollscm]# cat docker-compose.yml
version: '3.3'
services:
  db:
    image: mysql:8.0.19
    command: --default-authentication-plugin=mysql_native_password
    volumes:
      - db_data:/var/lib/mysql
    restart: always
    environment:
      - MYSQL_ROOT_PASSWORD=wordpress
      - MYSQL_DATABASE=databaseword
      - MYSQL_USER=admin
      - MYSQL_PASSWORD=admin123

  wordpress:
    image: wordpress:latest
    ports:
      - "80:80"
    restart: always
    environment:
      - WORDPRESS_DB_HOST=db
      - WORDPRESS_DB_USER=admin
      - WORDPRESS_DB_PASSWORD=admin123
      - WORDPRESS_DB_NAME=databaseword

volumes:
  db_data:
  db:
[root@ip-172-31-27-121 pollscm]#
```

- ✓ This is the code taken from the github. After changing the code we can check withbthe change in the build.

METHOD-7: DEPLOY WORDPRESS WEB APPLICATION BY USING TERRAFORM (CREATE EC2 INSTANCE ALONG WITH USERDATA.SH FILE)

Terraform is an open-source infrastructure as code (IaC) tool which is used to automate the process of creating, launching any AWS services and we can also deploy any application using the terraform.

- To create any services first we need to launch a EC2 instance with basic configurations.

Instances (1 / 2) [Info](#)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 IP	Elastic IP	IPv6 IP
My Terraform	i-0edd9197...	Terminated	t2.micro	-	-	us-east-1b	-	-	-	-
terraform	i-0785d07...	Running	t2.micro	2/2 checks passed	-	us-east-1b	ec2-52-90-135-99.com...	52.90.135.99	-	-

i-0785d0791345ac00f (terraform)

Details | Status and alarms | Monitoring | Security | **Networking** | Storage | Tags

Networking details

Public IPv4 address
52.90.135.99 [open address]

Public IPv4 DNS
ec2-52-90-135-99.compute-1.amazonaws.com [open address]

Subnet ID
subnet-0b99c3ee5856f66287

Availability zone
us-east-1b

Use RDN as guest OS hostname
Disabled

Private IP4 addresses
172.31.90.205

Private IP DNS name (IPv4 only)
ip-172-31-90-205.ec2.internal

IPV6 addresses
-

Carrier IP addresses (ephemeral)
-

Answer RDN DNS hostname IPv4
Enabled

VPC ID
vpc-03f5fe3fd53d1d03

Secondary private IPv4 addresses
-

Outpost ID
-

Network Interfaces (1) [Info](#)

[Filter network interfaces](#)

- I have launched a EC2 instance with amazon-linux application os image. We can see the details of the launched EC2 instance in this picture.

```
PUTTY (inactive)
[ec2-user@ip-172-31-90-205 ~]$ sudo su -
[ec2-user@ip-172-31-90-205 ~]# yum install mysql -y
Last metadata expiration check: 0:00:39 ago on Mon Dec 9 03:11:25 2024.
No match for argument: mysql
Error: Unable to find a match: mysql
[ec2-user@ip-172-31-90-205 ~]# sudo yum install -y yum-utils shadow-utils
Last metadata expiration check: 0:01:00 ago on Mon Dec 9 03:11:25 2024.
Package dnf-utils-4.3.0-13.amzn2023.0.4.noarch is already installed.
Package yum-utils-2.4.9-12.amzn2023.0.4.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-90-205 ~]# sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
Adding repo from: https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
[ec2-user@ip-172-31-90-205 ~]# sudo yum -y install terraform

Hashicorp Stable - x86_64
Dependencies resolved.

-----  

          Package           Architecture      Version       Repository  

          Size  

-----  

Installing:  

  terraform           x86_64          1.10.1-1      hashicorp  

  Dependencies resolved.  

-----  

          Package           Architecture      Version       Repository  

          Size  

-----  

Installing dependencies:  

  git                 x86_64          2.40.1-1.amzn2023.0.3  

  git-core             x86_64          2.40.1-1.amzn2023.0.3  

  git-core-doc         noarch          2.40.1-1.amzn2023.0.3  

  perl-Error           noarch          1:0.17029-5.amzn2023.0.2  

  perl-File-Find        noarch          1.37-477.amzn2023.0.6  

  perl-Git              noarch          2.40.1-1.amzn2023.0.3  

  perl-TermReadKey     x86_64          2.38-9.amzn2023.0.2  

  perl-lib              x86_64          0.65-477.amzn2023.0.6  

-----  

          Repository  

          Size  

-----  

  amazonlinux          54 k  

  amazonlinux          4.3 M  

  amazonlinux          2.6 M  

  amazonlinux          41 k  

  amazonlinux          26 k  

  amazonlinux          42 k  

  amazonlinux          36 k  

  amazonlinux          15 k  

-----  

Transaction Summary  

Install 9 Packages
```

- I have connected the created EC2 instance with the terminal and installed the terraform with the above commands.

```
root@ip-172-31-90-205:~# you run "terraform init" in the future.
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@ip-172-31-90-205 ~]# aws configure
AWS Access Key ID [None]: KIA5rzd3CWFQXEQ4W
AWS Secret Access Key [None]: elitza2s4TU00URXib7nhQ8xKJzN7mP3OoplKNT
Default region name [None]: us-east-1
Default output format [None]:
[root@ip-172-31-90-205 ~]# terraform plan
[...]
Error: Invalid function argument

on ec2.tf line 7, in resource "aws_instance" "public_subnet-1":
  7: user_data = ${file("userdata.sh")}
    |
    while calling file(path)

Invalid value for "path" parameter: no file exists at "userdata.sh"; this function works only with files that are distributed as part of the configuration source code, so if this file
will be created by a resource in this configuration you must instead obtain this result from an attribute of that resource.

[root@ip-172-31-90-205 ~]# vim userdata.sh
[root@ip-172-31-90-205 ~]# terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

+ aws_instance.public_subnet-1[0] will be created
+ resource "aws_instance" "public_subnet-1" {
  + ami                                = "ami-016ffe664262f664c"
  + arn                                = (known after apply)
  + associate_public_ip_address          = (known after apply)
  + availability_zone                   = (known after apply)
  + cpu_core_count                      = (known after apply)
  + cpu_threads_per_core                = (known after apply)
  + disable_api_stop                   = (known after apply)
  + disable_api_termination             = (known after apply)
  + ebs_optimized                       = (known after apply)
  + get_password_data                  = false
  + host_id                            = (known after apply)

22°C Mostly cloudy   Search   ENG IN   08:47

```

- After installing terraform, provided the authentication for using aws services with the command “aws configure” which uses the access keys and secret access keys for giving permission for providing authentication.

```
root@ip-172-31-205-205: ~# vim ec2.tf
Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
[root@ip-172-31-205-205 ~]# vim ec2.tf
key_name="new-key"
aws_security_group_ids=["${aws_security_group.demosg.id}"]
user_data=[file("userdata.sh")]
tags{
  Name="My Terraform"
}
}

resource "aws_security_group" "demosg" {
  ingress{
    from_port=80
    to_port=80
    protocol="tcp"
    cidr_blocks=["0.0.0.0/0"]
  }

  ingress{
    from_port=3306
    to_port=3306
    protocol="tcp"
    cidr_blocks=["0.0.0.0/0"]
  }

  ingress{
    from_port=22
    to_port=22
    protocol="tcp"
    cidr_blocks=["0.0.0.0/0"]
  }

  ingress{
    from_port=8080
    to_port=8080
    protocol="tcp"
    cidr_blocks=["0.0.0.0/0"]
  }

  egress{
    from_port=0
    to_port=0
    protocol="-1"
    cidr_blocks=["0.0.0.0/0"]
  }

  tags{
    Name="WEB SG"
  }
}
```

- After installing terraform I have created a ec2.tf file for creating a EC2 instance with amazon-linux 5.10 application os image AMI ID, t2.micro instance type, and provided the security with inbound ports with SSH(22),HTTP(80),MYSQL(3306) and initialized terraform.

```
root@ip-172-31-90-205:~#
#!/bin/bash
# Update all packages
yum update -y
# Install docker git MySQL curl
systemctl start docker
systemctl enable docker
sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
sudo usermod -aG docker ec2-user
sudo chmod 666 /var/run/docker.sock
mkdir -p /opt/docker-wordpress
cat <>DOL > /opt/docker-wordpress/docker-compose.yml
version: '3.3'
services:
  db:
    image: mysql:9.0.19
    command: --default-authentication-plugin=mysql_native_password
    volumes:
      - db_data:/var/lib/mysql
    restart: always
    environment:
      - MYSQL_ROOT_PASSWORD=wordpress
      - MYSQL_DATABASE=databasepassword
      - MYSQL_USER=admin
      - MYSQL_PASSWORD=admin123
  wordpress:
    image: wordpress:latest
    ports:
      - 80:80
    restart: always
    environment:
      - WORDPRESS_DB_HOST=db:3306 # Explicitly specifying port 3306
      - WORDPRESS_DB_USER=admin
      - WORDPRESS_DB_PASSWORD=admin123
      - WORDPRESS_DB_NAME=databasepassword
    volumes:
      - db_data:
        db_data:
EOF
cd /opt/docker-wordpress
docker-compose up

```
*userdata.sh" 4LL, 1229B
```

- Created a userdata.tf file by configuring the installations like mysql, docker, docker-compose.
  - I have created the docker-compose.yml file inside the userdata.
  - Created a directory with /opt/docker-wordpress so that I can store the docker-compose.yml file inside this directory.
  - Moved inside the above created directory and performed “docker-compose up -d” to deploy wordpress application using the terraform.

```
[root@ip-172-31-90-205 ~]# terraform plan
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create
Terraform will perform the following actions:

aws_instance.public_subnet-1[0] will be created
+ resource "aws_instance" "public_subnet-1" {
 ami = "ami-0166fe664262f664c"
 arn = (known after apply)
 associate_public_ip_address = (known after apply)
 availability_zone = (known after apply)
 cpu_core_count = (known after apply)
 cpu_threads_per_core = (known after apply)
 disable_api_stop = (known after apply)
 disable_api_termination = (known after apply)
 ebs_optimized = (known after apply)
 get_password_data = false
 host_id = (known after apply)
 host_resource_group_arn = (known after apply)
 iam_instance_profile = (known after apply)
 id = (known after apply)
 instance_initiated_shutdown_behavior = (known after apply)
 instance_lifecycle = (known after apply)
 instance_state = (known after apply)
 instance_type = "t2.micro"
 ipv6_address_count = (known after apply)
 ipv6_addresses = (known after apply)
 key_name = "new-key"
 monitoring = (known after apply)
 outpost_arn = (known after apply)
 password_data = (known after apply)
 placement_group = (known after apply)
 placement_partition_number = (known after apply)
 primary_network_interface_id = (known after apply)
 private_dns = (known after apply)
 private_ip = (known after apply)
 public_dns = (known after apply)
 public_ip = (known after apply)
 secondary_private_ips = (known after apply)
 security_groups = (known after apply)
 source_dest_check = true
 spot_instance_request_id = (known after apply)
 subnet_id = (known after apply)
 tags = {
 + "Name" = "My Terraform "
 }
 tags_all = [
]
}
```

- Here I have used the command “terraform plan” to check what AWS service or resource is going to create.
- We can see the AMI id of instance, name of instance used in the code.

```
[root@ip-172-31-90-205 ~]# terraform apply
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create
Terraform will perform the following actions:

aws_instance.public_subnet-1[0] will be created
+ resource "aws_instance" "public_subnet-1" {
 ami = "ami-0166fe664262f664c"
 arn = (known after apply)
 associate_public_ip_address = (known after apply)
 availability_zone = (known after apply)
 cpu_core_count = (known after apply)
 cpu_threads_per_core = (known after apply)
 disable_api_stop = (known after apply)
 disable_api_termination = (known after apply)
 ebs_optimized = (known after apply)
 get_password_data = false
 host_id = (known after apply)
 host_resource_group_arn = (known after apply)
 iam_instance_profile = (known after apply)
 id = (known after apply)
 instance_initiated_shutdown_behavior = (known after apply)
 instance_lifecycle = (known after apply)
 instance_state = (known after apply)
 instance_type = "t2.micro"
 ipv6_address_count = (known after apply)
 ipv6_addresses = (known after apply)
 key_name = "new-key"
 monitoring = (known after apply)
 outpost_arn = (known after apply)
 password_data = (known after apply)
 placement_group = (known after apply)
 placement_partition_number = (known after apply)
 primary_network_interface_id = (known after apply)
 private_dns = (known after apply)
 private_ip = (known after apply)
 public_dns = (known after apply)
 public_ip = (known after apply)
 secondary_private_ips = (known after apply)
 security_groups = (known after apply)
 source_dest_check = true
 spot_instance_request_id = (known after apply)
 subnet_id = (known after apply)
 tags = {
 + "Name" = "My Terraform "
 }
 tags_all = [
]
}
```

- Here I have performed the command “terraform apply” to apply the above planned resources.
- By applying this a EC2 instance with the specified security, application os image, instance type is created.

```

root@ip-172-31-90-205:~#
+ protocol = "tcp"
+ security_groups = []
+ self = false
+ to_port = 8080
(1 unchanged attribute hidden)
},
+
+ cidr_blocks = [
+ "+0.0.0.0/0",
]
+ from_port = 80
+ ipv6_cidr_blocks = []
+ prefix_list_ids = []
+ protocol = "tcp"
+ security_groups = []
+ self = false
+ to_port = 80
(1 unchanged attribute hidden)
],
+
+ name = (known after apply)
+ name_prefix = (known after apply)
+ owner_id = (known after apply)
+ revoke_rules_on_delete = false
+ tags = [
+ "+Name" = "WEB SG"
]
+ tags_all = [
+ "+Name" = "WEB SG"
]
+ vpc_id = (known after apply)
]

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

Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

aws_security_group.demog: Creating...
aws_security_group.demog: Creation complete after 2s [id=sg-06fc48b701166d0ca]
aws_instance.myinstance: Creating... [10s elapsed]
aws_instance.public_subnet-1:0: Still Creating... [10s elapsed]
aws_instance.public_subnet-1:0: Creation complete after 13s [id=i-0edd9197b0155e12c]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
[root@ip-172-31-90-205 ~]#

```

22°C Mostly cloudy

ENG IN 08:47 09-12-2024

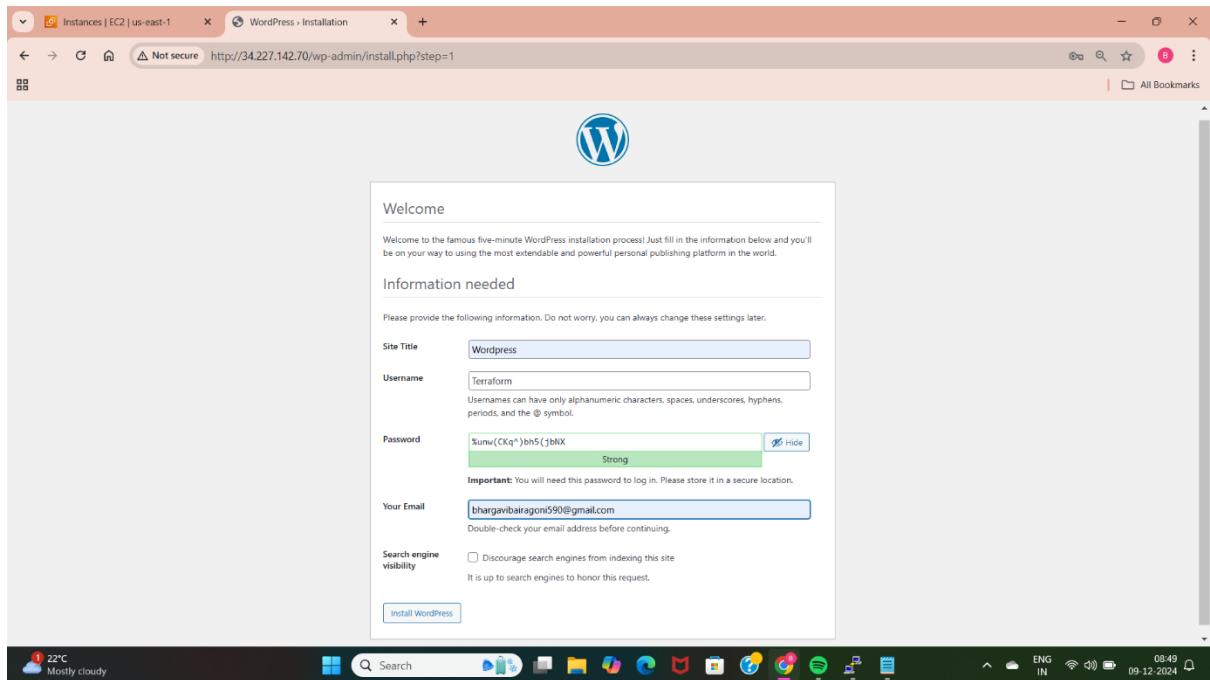
- The “terraform apply” command is successful with creating two resources one is EC2 instance with userdata.

| Name         | Instance ID         | Instance state | Instance type | Status check | Alarm status                | Availability Zone | Public IPv4 DNS         | Public IPv4 |
|--------------|---------------------|----------------|---------------|--------------|-----------------------------|-------------------|-------------------------|-------------|
| My.Terraform | i-0edd9197b0155e12c | Running        | t2.micro      | Initializing | <a href="#">View alarms</a> | us-east-1b        | ec2-34-227-142-70.co... | \$4.227.142 |
| terraform    | i-0785d07...        | Running        | t2.micro      | 2/2 checks p | <a href="#">View alarms</a> | us-east-1b        | ec2-52-90-135-99.com... | \$2.90.135  |

- We can check with the AWS management console whether the above terraform code is working properly or not.
- We can see the instance with the name “My terraform” is created successfully.

- We can also check with the security group that is allowing above mentioned inbound rules.
- After checking with the instance and the security group copy and paste the IP address of this instance in the browser so that we can able to access the wordpress application.

- Now I am able to access the wordpress application by using the public IP address of the instance.

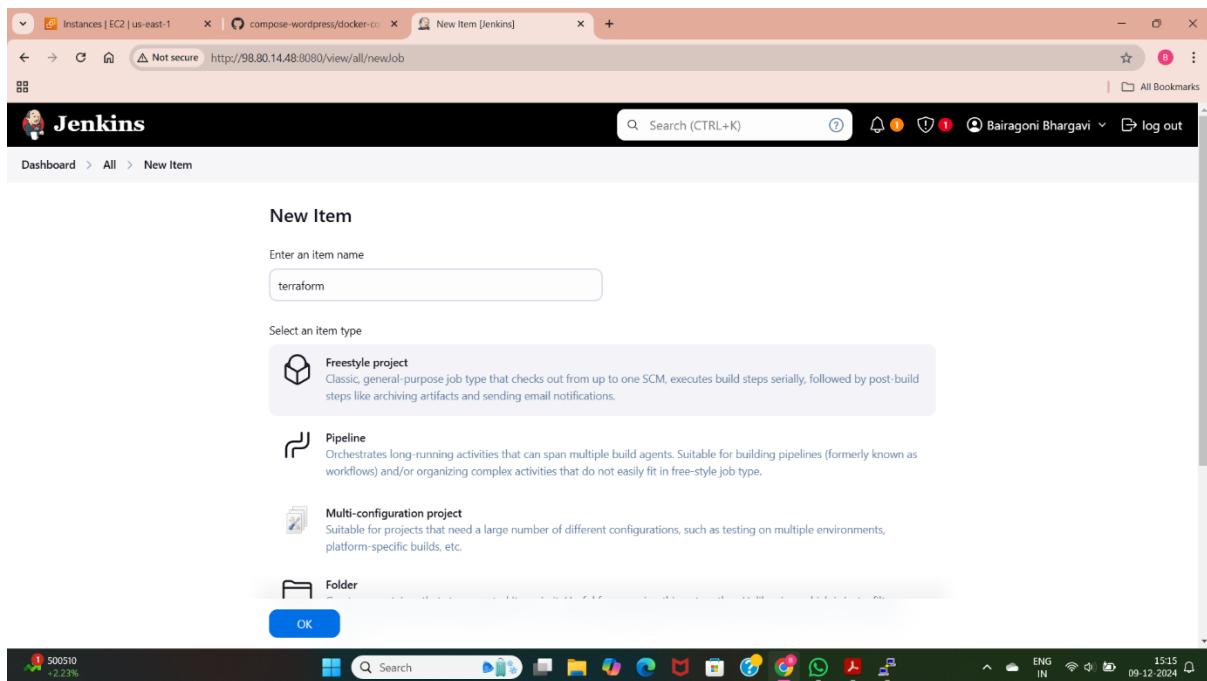


- The picture shows that the wordpress application is deployed successfully using the terraform and the userdata.

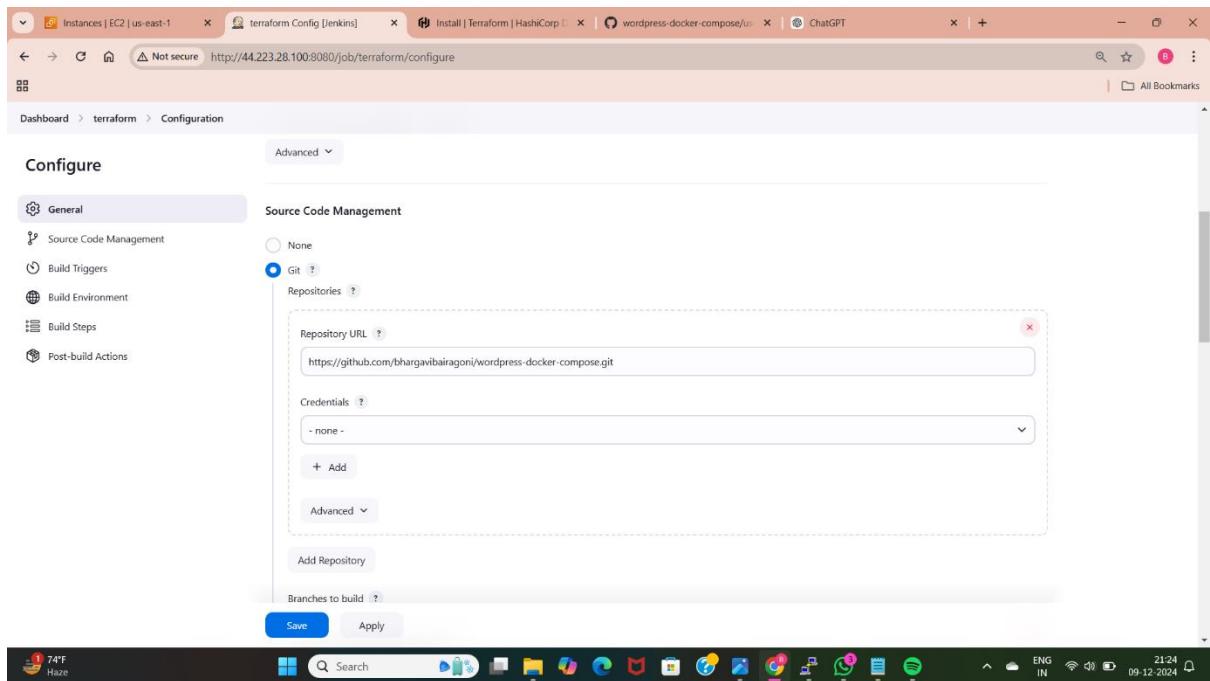
## METHOD-8: DEPLOYING USING TERRAFORM AND JENKINS EXECUTE SHELL

Deploy WordPress web application by using git (clone terraform script which helps to deploy WordPress web application), jenkins (in execute shell install terraform, init, fmt, validate and apply with automatic command as terraform apply --auto-approve) and terraform.

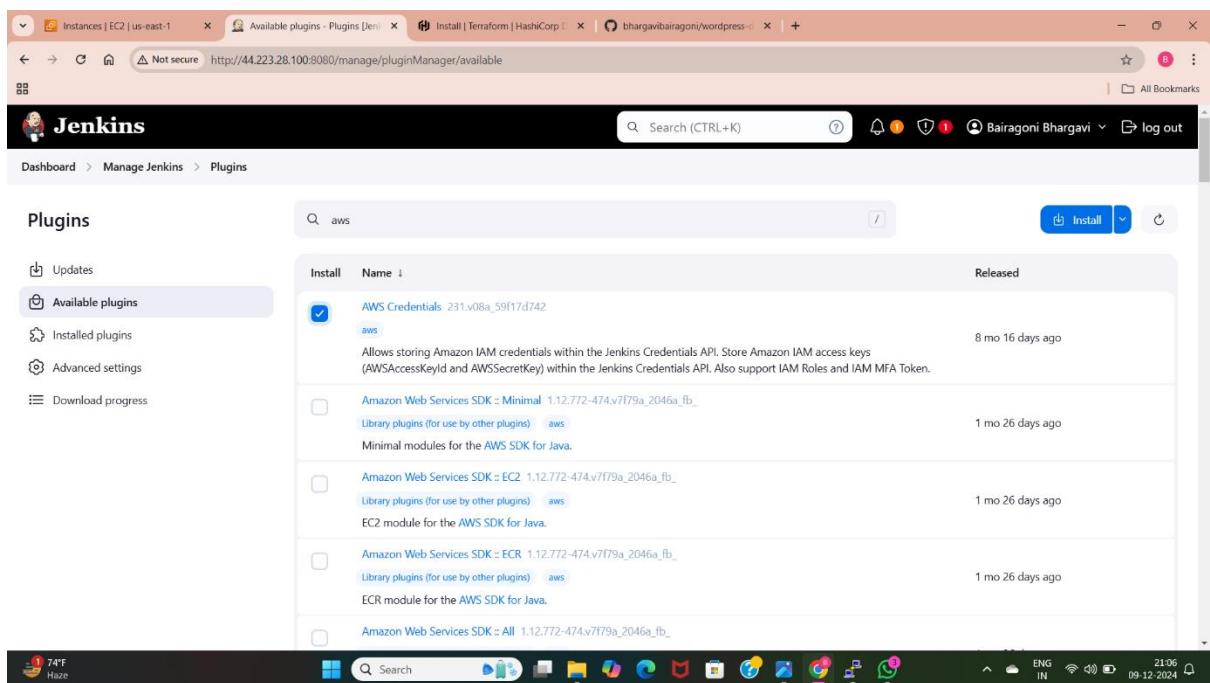
- First we need to create a instance and setup Jenkins so that we can integrate terraform with Jenkins.



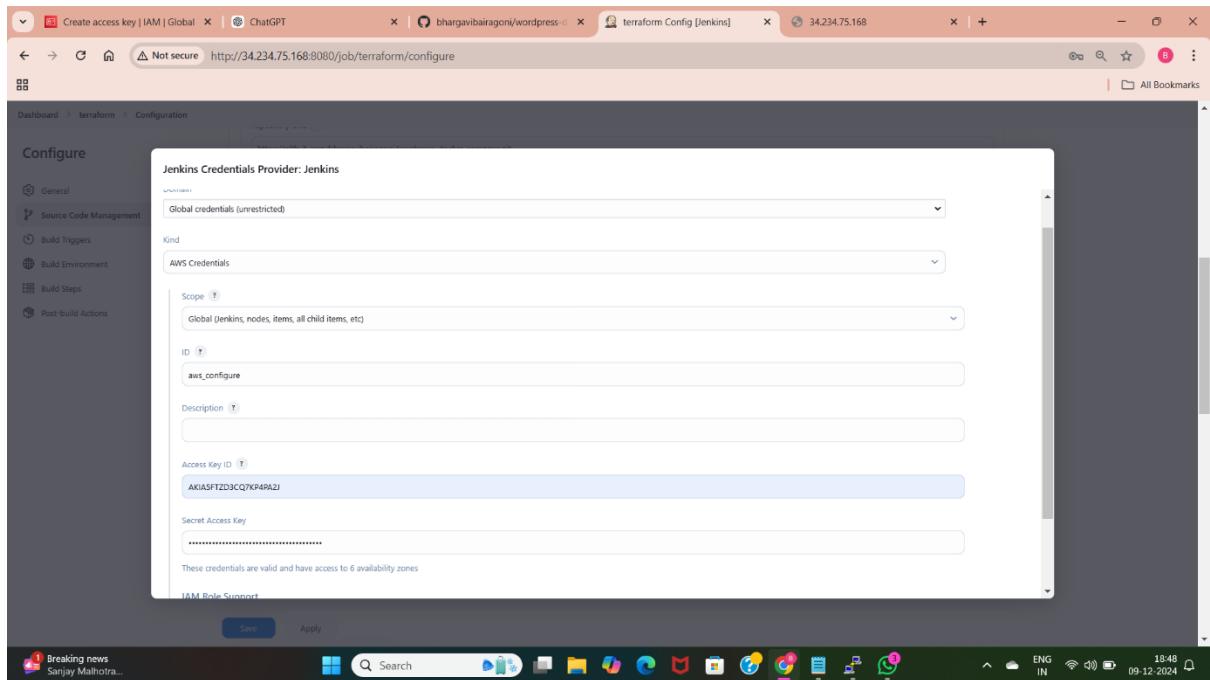
- Here I have created a Jenkins job with the name of Terraform with the freestyle project.



- ♣ To deploy the wordpress application using the terraform we need to take the code from the github. Here I have taken the url of the code from github using source code management.



- ♣ To provide the credentials to access aws services we need to install the aws credentials plugin.
- ♣ For that we need to go to manage Jenkins, select plugins option and go to available plugins and search for aws credentials plugins and install the plugin.



- Now by using the aws credentials plugin give the access keys and secret access keys for the Jenkins to use Aws services. For that go to management Jenkins and select credentials option , select global configurations, select add credentials and select the aws credentials plugin and provide access keys and secret access keys of aws management console.

| T | P | Store  | Domain   | ID  | Name                                     |
|---|---|--------|----------|-----|------------------------------------------|
|   |   | System | (global) | aws | AKIA5FTZD3CQ7KP4PA2J (terraform jenkins) |

Stores scoped to Jenkins

| P | Store  | Domains  |
|---|--------|----------|
|   | System | (global) |

Icon: S M L

REST API Jenkins 2.479.2

- Here I have created credentials for access keys and secret access keys.

The screenshot shows the Jenkins configuration interface for a job named "terraform". The "Build Environment" section is selected. Under "Bindings", there is a "AWS access key and secret" section. It contains fields for "Access Key Variable" (set to "AWS\_ACCESS\_KEY\_ID") and "Secret Key Variable" (set to "AWS\_SECRET\_ACCESS\_KEY"). Below this, under "Credentials", there is a dropdown menu set to "Specific credentials" with the value "AKIA5F1ZD3CQ7K4PA2J". There are also checkboxes for "Add timestamps to the Console Output" and "Inspect build log for published build scans". At the bottom are "Save" and "Apply" buttons.

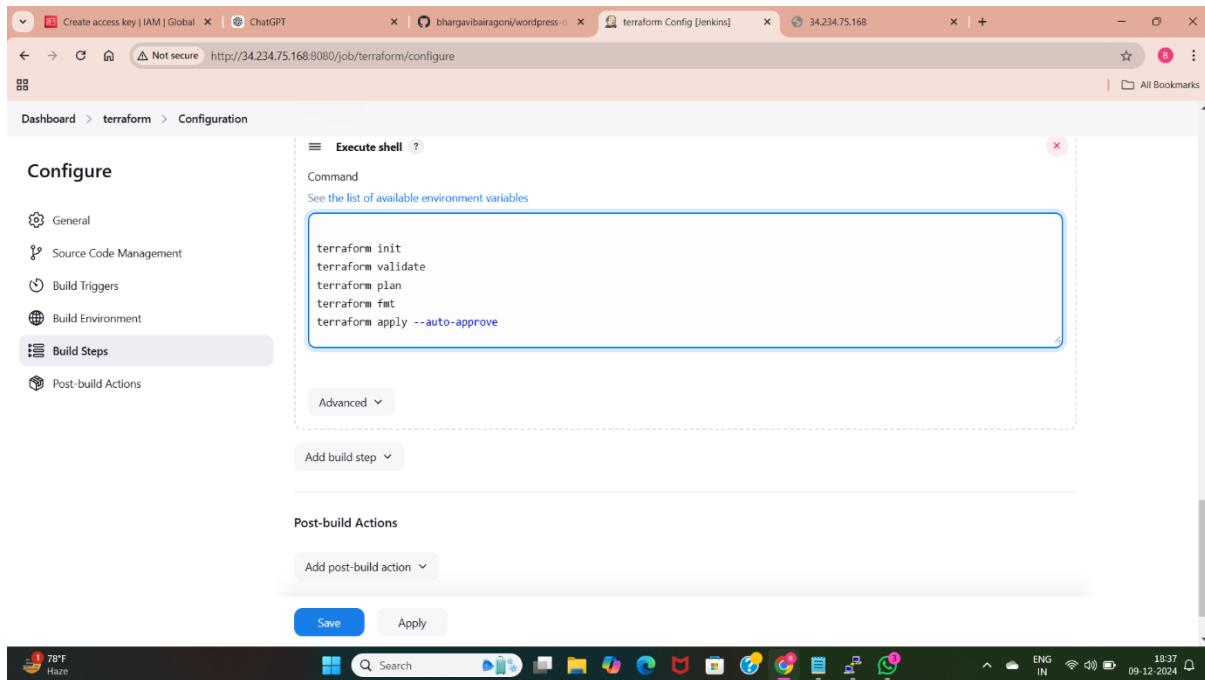
- Added credentials for Jenkins to access Aws services by using “aws configure” plugin. Added access keys and secret access key.

```

root@ip-172-31-16-18:~# cd /var/lib/jenkins/workspace/terraform-wordpress-docker-compose
remote:
To https://github.com/bhargavibairagoni/terraform-jenkins.git
 ! [remote rejected] master -> master (push declined due to repository rule violations)
error: failed to push some refs to 'https://github.com/bhargavibairagoni/terraform-jenkins.git'
[root@ip-172-31-16-18 ~]# clear
[root@ip-172-31-16-18 ~]# ll
total 8
-rw-r--r-- 1 root root 848 Dec 9 12:08 ec2.tf
-rw-r--r-- 1 root root 1570 Dec 9 12:08 userdata.tf
[root@ip-172-31-16-18 ~]# rm -rf *
[root@ip-172-31-16-18 ~]# cd /var/lib/jenkins/workspace
[root@ip-172-31-16-18 workspace]# ll
total 0
drwxr-xr-x 3 jenkins jenkins 38 Dec 9 12:30 terraform
[root@ip-172-31-16-18 workspace]# cd terraform/
[root@ip-172-31-16-18 terraform]# ll
total 0
drwxr-xr-x 4 jenkins jenkins 117 Dec 9 12:30 wordpress-docker-compose
[root@ip-172-31-16-18 terraform]# rm -rf *
[root@ip-172-31-16-18 terraform]# ll
total 0
drwxr-xr-x 4 jenkins jenkins 117 Dec 9 12:32 wordpress-docker-compose
[root@ip-172-31-16-18 terraform]# rm -rf *
[root@ip-172-31-16-18 terraform]# ll
total 0
drwxr-xr-x 5 jenkins jenkins 138 Dec 9 12:34 wordpress-docker-compose
[root@ip-172-31-16-18 terraform]# cd wordpress-docker-compose/
[root@ip-172-31-16-18 wordpress-docker-compose]# ll
total 20
-rw-r--r-- 1 jenkins jenkins 600 Dec 9 12:34 docker-compose.yml
-rw-r--r-- 1 jenkins jenkins 131 Dec 9 12:34 pipeline
-rw-r--r-- 1 jenkins jenkins 752 Dec 9 12:34 terraform-ec2.tf
-rw-r--r-- 1 jenkins jenkins 1229 Dec 9 12:34 userdata.sh
drwxr-xr-x 5 jenkins jenkins 4096 Dec 9 12:34 wordpress
[root@ip-172-31-16-18 wordpress-docker-compose]# history
1 yum install git -y
2 sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo
3 sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key
4 sudo yum install java-17-amazon-corretto-devel -y
5 java -version
6 yum install Jenkins -y
7 systemctl start jenkins
8 cat /var/lib/jenkins/secrets/initialAdminPassword
9 git init
10 sudo yum install -y yum-utils shadow-utils
11 sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
12 sudo yum -y install terraform
13 systemctl restart jenkins

```

- These are the files which are pulled from the github by using the code url, and these are the commands used to install Jenkins, terraform. We can see the terraform-ec2.tf and userdata.sh files in the above picture.



- ♣ These are commands used in the execute shell to deploy the application using the terraform.
- ♣ With the help of github url I have taken the code for creating EC2 instance and userdata.
- ♣ Terraform init is used to initialize the terraform in the created files.
- ♣ I have applied the changes by using “terraform apply” command.
- ♣ After this command a EC2 instance is created with userdata so that our application is deployed.

The Jenkins dashboard shows the 'terraform' project. The 'Status' tab is active, displaying the following build history:

- Last build (#1), 1 hr 17 min ago
- Last failed build (#1), 1 hr 17 min ago
- Last unsuccessful build (#1), 1 hr 17 min ago
- Last completed build (#1), 1 hr 17 min ago

The Jenkins interface includes a sidebar with options like Status, Changes, Workspace, Build Now, Configure, Delete Project, and Rename.

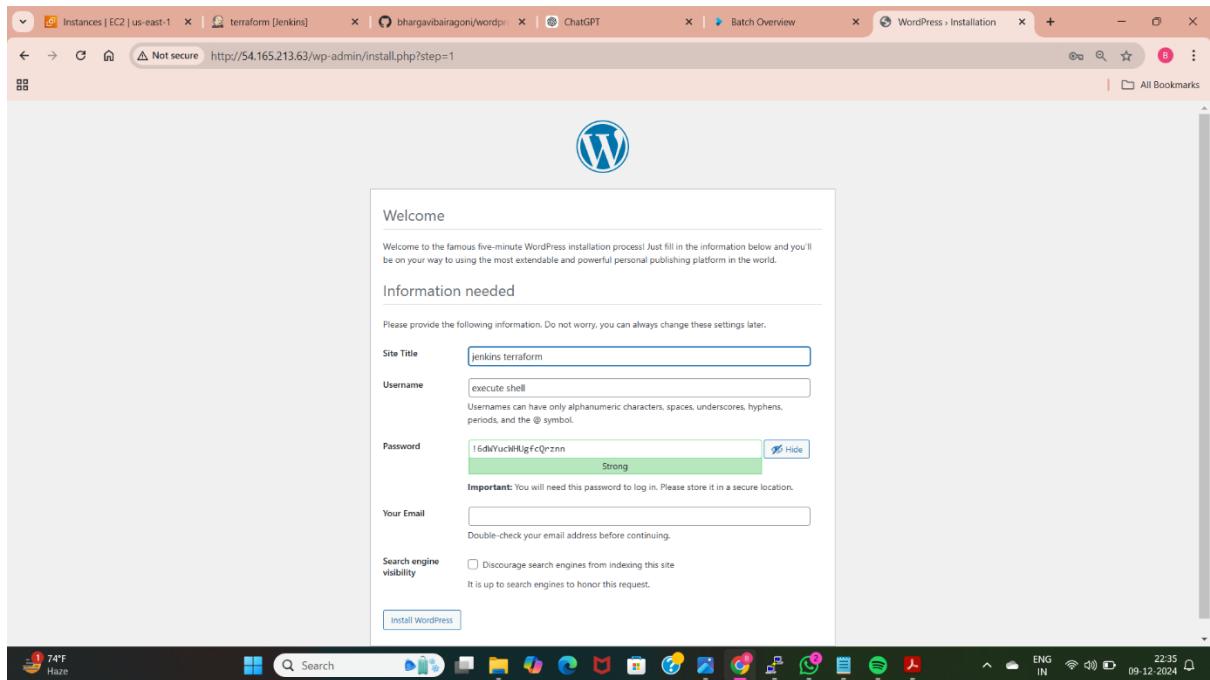
- ♣ After writing the script in the execute shell we need to build the freestyle project with the name of the terraform, so that we can able to execute the terraform files and create Ec2 instance which uses userdata script to deploy wordpress application. I have got the error while building the job and I have rectified the error and the build is success after three builds.

The screenshot shows the AWS Management Console with the EC2 Instances page open. The left sidebar shows navigation options like Dashboard, EC2 Global View, Events, Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, AMIs, AMI Catalog, Elastic Block Store, Volumes, Snapshots, Lifecycle Manager, Network & Security, Security Groups, Elastic IPs, Placement Groups, Key Pairs, and Network Interfaces. The main content area displays a table of instances. A search bar at the top allows filtering by Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, Public IPv4 DNS, and Public IPv4. The table shows two instances: 'jenkins' (Running, t2.micro) and 'My Terraform' (Running, t2.micro). The 'My Terraform' instance is selected, and its details are shown in the right-hand pane. The details pane includes tabs for Details, Status and alarms, Monitoring, Security, Networking, Storage, and Tags. Under the Details tab, the Instance summary section provides information such as Instance ID (i-05a40dd39d3d07559), Instance state (Running), Public IPv4 address (54.165.213.63), Private IP DNS name (ip-172-31-23-54.ec2.internal), Instance type (t2.micro), VPC ID (vpc-03f5fe3fd53d1d03), Subnet ID (subnet-0875dbfeb85c315), and Instance ARN.

- After building the above execute shell we can see that the terraform code is launching a new instance with userdata which consists configurations related to deploying the wordpress application.

The screenshot shows a browser window with the URL http://54.165.213.63/wp-admin/install.php. The page displays the WordPress logo and a language selection dropdown menu. The menu is currently set to "English (United States)" but also lists other languages: Afrikaans, Aragonés, العربية, العربية المعاشرة, অসমীয়া, گوئىچى ئەرپاجان, Azerbaijani dili, Беларуская мова, Български, ସ୍ବାମୀ, Bosanski, Català, Cebuano, Čeština, Cymraeg, Dansk, Deutsch (Schweiz, Du), and Deutsch (Schwelz). Below the dropdown is a "Continue" button.

- After successful build we will get the new instance, we need to take the ip address of the new instance and paste in the google , so we get the above wordpress page. Select the language for accessing the wordpress, we can find the login page for wordpress.

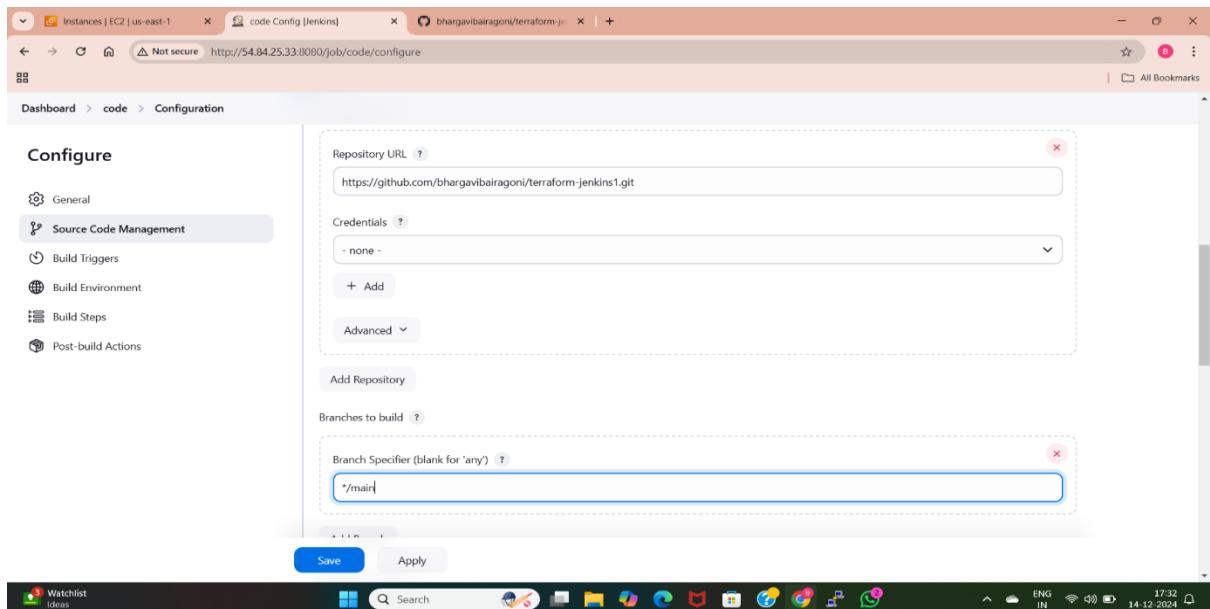


- Now I can able to access the wordpress application, provide the login details and access the wordpress application.

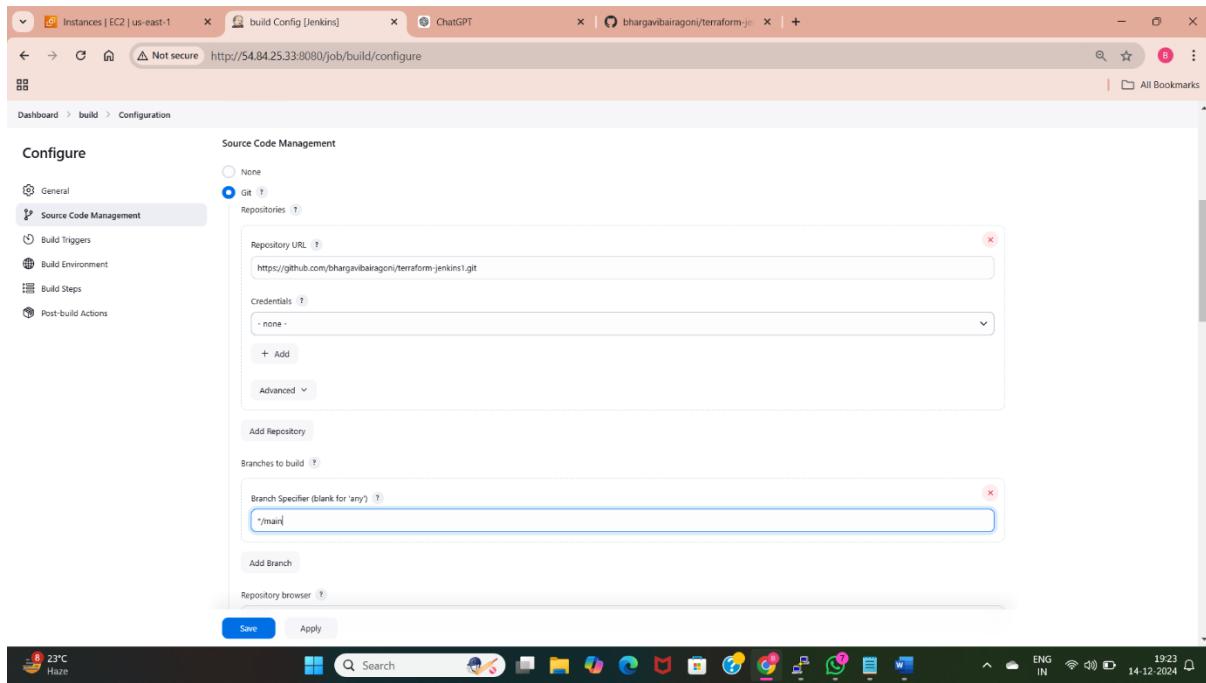
## METHOD-9: DEPLOY WORDPRESS WEB APPLICATION BY USING JENKINS PIPELINE

Deploy WordPress web application by using git (clone terraform script which helps to deploy WordPress web application), jenkins (in execute shell install terraform, init, fmt, validate and apply with automatic command as terraform apply –auto-approve) and terraform and create Jenkins pipeline and add build periodically and poll scm to initial job of pipeline and check the changes happened or not which are made in github repo.

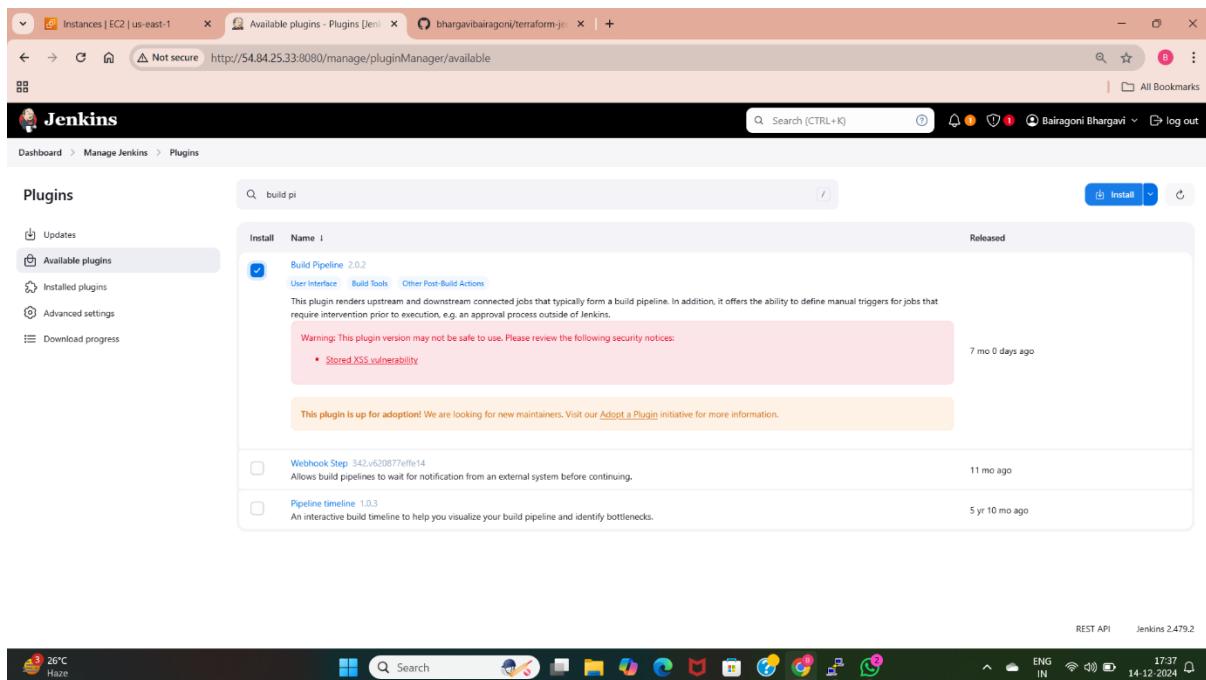
- To deploy wordpress application using pipeline first we need to create a job with freestyle project.



- I have created a free style job with the name “code”, I have taken the url from the github using this freestyle job.



- I have created a freestyle job with the name “build”, added the code in this job using git url.



- I have used the “build pipeline” plugin to build the above jobs automatically by building only one job.

The screenshot shows the Jenkins build configuration page. Under the 'Build Environment' section, the 'AWS access key and secret' tab is selected. It displays two environment variables: 'AWS\_ACCESS\_KEY\_ID' and 'AWS\_SECRET\_ACCESS\_KEY'. Below these, under 'Credentials', 'Specific credentials' is selected, and the value 'AKIA5FTZD3CQ7K74PA2J' is shown. There are also three optional checkboxes at the bottom: 'Add timestamps to the Console Output', 'Inspect build log for published build scans', and 'Terminate a build if it's stuck'. At the bottom right of the configuration panel are 'Save' and 'Apply' buttons.

- I have assigned the access keys and secret access keys by using the credentials of aws credential plugin.
- First we need to go to manage Jenkins, move to credentials , select global credentials, select aws credentials plugin then give access keys and secret access keys.

The screenshot shows the Jenkins build configuration page. Under the 'Build Steps' section, the 'Execute shell' step is selected. The command entered is:

```
export AWS_REGION=us-east-1
git clone https://github.com/bhargavibairagi/terraform-jenkins.git
sudo yum install -y yum-utils shadow-utils
sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
sudo yum -y install terraform
cd terraform-jenkins
terraform init
terraform validate
terraform fmt
terraform apply --auto-approve
```

Below this, there is an 'Advanced' dropdown and an 'Add build step' button. Under the 'Post-build Actions' section, the 'Build other projects' tab is selected, showing a list of projects to build. At the bottom right of the configuration panel are 'Save' and 'Apply' buttons.

- I have used the execute shell in the build project.
- I have used the aws region us-east-1 for creating the resources in particular region.
- I have installed the terraform using the above commands.
- Moved to the directory to which this files(ec2.tf, userdata) are stored.
- I have initialized the terraform in the above files.
- I have checked the errors using the “terraform validate” command.
- I have applied the changes to launch the ec2 instance with userdata.

```

root@ip-172-31-37-211:~#
Defaults env_reset
Defaults env_keep += "COLORS DISPLAY HOSTNAME HISTSIZE KDEDIR LS_COLORS"
Defaults env_keep += "MAIL PS1 PS2 QDIR USERNAME LANG LC_ADDRESS LC_CTYPE"
Defaults env_keep += "LC_MONETARY LC_NAME LC_NUMERIC LC_PAPER LC_TELEPHONE"
Defaults env_keep += "LC_TIME LC_ALL LANGUAGE LINGUAS _XKB_CHARSET XAUTHORITY"

Adding HOME to env_keep may enable a user to run unrestricted
commands via sudo.
#
Defaults env_keep += "HOME"

Defaults secure_path = /sbin:/bin:/usr/sbin:/usr/bin

Next comes the main part: which users can run what software on
which machines (the sudoers file can be shared between multiple
systems).
syntax:
user MACHINE=COMMANDS
THE COMMANDS section may have other options added to it.
Allow root to run any commands anywhere
root ALL=(ALL) ALL

Allows members of the 'sys' group to run networking, software,
service management apps and more.
sys ALL = NETWORKING, SOFTWARE, SERVICES, STORAGE, DELEGATING, PROCESSES, LOCATE, DRIVERS

Allows people in group wheel to run all commands
%wheel ALL=(ALL) ALL

Same thing without a password
wheel ALL=(ALL) NOPASSWD: ALL

Allows members of the users group to mount and umount the
cdrom as root
users ALL=/sbin/mount /mnt/cdrom, /sbin/umount /mnt/cdrom

Allows members of the users group to shutdown this system
users localhost:/sbin/shutdown -h now

Read drop-in files from /etc/sudoers.d (the # here does not mean a comment)
#includedir /etc/sudoers.d
#includedir ALL=(ALL) NOPASSWD: ALL

```

- To install terraform using sudo we need to give the sudo permissions in the Jenkins. First we need to move into the “visudo” and give permissions using “Jenkins ALL=(ALL) NOPASSWD: ALL”.

```

root@ip-172-31-37-211:~/var/lib/jenkins/workspace/#
drwxr-xr-x 1 jenkins jenkins 241 Dec 14 13:14 jenkins.model.GlobalComputerRetentionCheckIntervalConfiguration.xml
drwxr-xr-x 1 jenkins jenkins 262 Dec 14 13:14 jenkins.model.JenkinsLocationConfiguration.xml
drwxr-xr-x 1 jenkins jenkins 86 Dec 14 13:14 jenkins.security.ResourceDomainConfiguration.xml
drwxr-xr-x 1 jenkins jenkins 179 Dec 14 13:14 jenkins.tasks.filters.EnvVarsFilterGlobalConfiguration.xml
drwxr-xr-x 1 jenkins jenkins 171 Dec 14 11:57 jenkins.telemetry.Correlator.xml
drwxr-xr-x 4 jenkins jenkins 31 Dec 14 13:32 jobs
drwxr-xr-x 2 jenkins jenkins 32 Dec 14 11:59 logs
-rw-r--r-- 1 jenkins jenkins 107 Dec 14 13:19 nodeMonitors.xml
drwxr-xr-x 1 jenkins jenkins 167 Dec 14 13:14 org.jenkinsci.plugins.displayurlapi.defaultDisplayURLProviderGlobelConfiguration.xml
drwxr-xr-x 1 jenkins jenkins 289 Dec 14 13:14 org.jenkinsci.plugins.github_branch_source.GitHubConfiguration.xml
drwxr-xr-x 1 jenkins jenkins 167 Dec 14 13:14 org.jenkinsci.plugins.workflow.flow.GlobalDefaultFlowDurabilityLevel.xml
drwxr-xr-x 1 jenkins jenkins 229 Dec 14 13:14 org.jenkinsci.plugins.workflow.libs.globalLibraries.xml
drwxr-xr-x 1 jenkins jenkins 247 Dec 14 13:14 org.jenkinsci.plugins.workflow.libs.globalUntrustedLibraries.xml
drwxr-xr-x 99 jenkins jenkins 8192 Dec 14 12:35 plugins
drwxr-xr-x 1 jenkins jenkins 259 Dec 14 13:34 queue.xml
-rw-r--r-- 1 jenkins jenkins 259 Dec 14 13:08 queue.xml.bak
drwxr-xr-x 1 jenkins jenkins 64 Dec 14 11:57 secret.key
drwxr-xr-x 1 jenkins jenkins 112 Dec 14 11:57 secret.key.not-so-secret
drwxr-xr-x 2 jenkins jenkins 295 Dec 14 12:08 tests
drwxr-xr-x 2 jenkins jenkins 149 Dec 14 11:58 updates
drwxr-xr-x 2 jenkins jenkins 24 Dec 14 11:57 userContent
drwxr-xr-x 4 jenkins jenkins 96 Dec 14 13:27 users
drwxr-xr-x 5 jenkins jenkins 45 Dec 14 12:44 workspace
[root@ip-172-31-37-211 jenkins]# cd workspace/
[root@ip-172-31-37-211 workspace]# ll
total 0
drwxr-xr-x 4 jenkins jenkins 44 Dec 14 12:24 build
drwxr-xr-x 3 jenkins jenkins 51 Dec 14 13:27 code
drwxr-xr-x 2 jenkins jenkins 31 Dec 14 13:30 deploy
[root@ip-172-31-37-211 workspace]# cd build/
[root@ip-172-31-37-211 build]# ll
total 0
drwxr-xr-x 4 jenkins jenkins 153 Dec 14 13:35 terraform-jenkins1
[root@ip-172-31-37-211 build]#
[root@ip-172-31-37-211 build]# cd terraform-jenkins1/
[root@ip-172-31-37-211 terraform-jenkins1]# ll
total 24
-rw-r--r-- 1 jenkins jenkins 1011 Dec 14 13:08 ec2.tf
-rw-r--r-- 1 jenkins jenkins 181 Dec 14 13:35 terraform.tfstate
-rw-r--r-- 1 jenkins jenkins 8218 Dec 14 13:35 terraform.tfstate.backup
-rw-r--r-- 1 jenkins jenkins 1401 Dec 14 12:24 userdata.sh
[root@ip-172-31-37-211 terraform-jenkins1]# cd ..
[root@ip-172-31-37-211 build]# cd ..
[root@ip-172-31-37-211 workspace]# cd code/
[root@ip-172-31-37-211 code]# ll
total 0
-rw-r--r-- 1 jenkins jenkins 707 Dec 14 13:27 ec2.tf
-rw-r--r-- 1 jenkins jenkins 1401 Dec 14 12:07 userdata.sh
[root@ip-172-31-37-211 code]#

```

- I have checked with the terminal by moving into the Jenkins path “/var/lib/Jenkins”, there we can find details of job by using “workspace”, there we can see the created jobs and go inside every job and check whether the files are there or not.

```

root@ip-172-31-37-211:/var/lib/jenkins/workspace/build/terraform-jenkins1#
-rw-r--r-- 1 jenkins jenkins 297 Dec 14 13:14 io.jenkins.plugins.junit.storage.JunitTestResultStorageConfiguration.xml
-rw-r--r-- 1 jenkins jenkins 7 Dec 14 13:15 jenkins.fingerprints.GlobalFingerprintConfiguration.xml
-rw-r--r-- 1 jenkins jenkins 7 Dec 14 12:00 jenkins.install.InstallUtil.lastExecVersion
-rw-r--r-- 1 jenkins jenkins 260 Dec 14 13:14 jenkins.metrics.ArtifactManagementKey.xml
-rw-r--r-- 1 jenkins jenkins 153 Dec 14 13:14 jenkins.model.ArtifactManagementConfiguration.xml
-rw-r--r-- 1 jenkins jenkins 253 Dec 14 13:14 jenkins.model.GlobalBuildDiscarderConfiguration.xml
-rw-r--r-- 1 jenkins jenkins 241 Dec 14 13:14 jenkins.model.GlobalComputerRetentionCheckIntervalConfiguration.xml
-rw-r--r-- 1 jenkins jenkins 262 Dec 14 13:14 jenkins.model.JenkinsLocationConfiguration.xml
-rw-r--r-- 1 jenkins jenkins 86 Dec 14 13:14 jenkins.security.ResourceMainConfiguration.xml
-rw-r--r-- 1 jenkins jenkins 179 Dec 14 13:14 jenkins.tasks.filters.EnvVarsFilterGlobalConfiguration.xml
-rw-r--r-- 1 jenkins jenkins 171 Dec 14 11:57 jenkins.telemetry.correlator.xml
drwxr-xr-x 4 jenkins jenkins 31 Dec 14 13:32 jobs
drwxr-xr-x 2 jenkins jenkins 32 Dec 14 11:59 logs
-rw-r--r-- 1 jenkins jenkins 101 Dec 14 13:14 nodeMonitors.xml
-rw-r--r-- 1 jenkins jenkins 187 Dec 14 13:14 org.jenkinsci.plugins.displayurlapi.DefaultDisplayURLProviderGlobalConfiguration.xml
-rw-r--r-- 1 jenkins jenkins 289 Dec 14 13:14 org.jenkinsci.plugins.github_branch_source.GitHubConfig.xml
-rw-r--r-- 1 jenkins jenkins 167 Dec 14 13:14 org.jenkinsci.plugins.workflow.flow.GlobalDefaultFlowDurabilityLevel.xml
-rw-r--r-- 1 jenkins jenkins 229 Dec 14 13:14 org.jenkinsci.plugins.workflow.libs.GlobalLibraries.xml
-rw-r--r-- 1 jenkins jenkins 247 Dec 14 13:14 org.jenkinsci.plugins.workflow.libs.GlobalUntrustedLibraries.xml
drwxr-xr-x 99 jenkins jenkins 8192 Dec 14 12:35 plugins
-rw-r--r-- 1 jenkins jenkins 259 Dec 14 13:34 queue.xml
-rw-r--r-- 1 jenkins jenkins 259 Dec 14 13:08 queue.xml.bak
-rw-r--r-- 1 jenkins jenkins 64 Dec 14 13:34 secret.key
-rw-r--r-- 1 jenkins jenkins 64 Dec 14 11:57 secret.key.not-so-secret
drwxr-xr-x 2 jenkins jenkins 295 Dec 14 12:08 scripts
drwxr-xr-x 2 jenkins jenkins 149 Dec 14 11:58 updates
drwxr-xr-x 2 jenkins jenkins 24 Dec 14 11:57 userContent
drwxr-xr-x 4 jenkins jenkins 96 Dec 14 13:27 users
drwxr-xr-x 5 jenkins jenkins 45 Dec 14 12:44 workspace
[root@ip-172-31-37-211 jenkins]# cd workspace/
[root@ip-172-31-37-211 workspace]# ll
total 0
drwxr-xr-x 4 jenkins jenkins 44 Dec 14 12:24 build
drwxr-xr-x 3 jenkins jenkins 51 Dec 14 13:27 code
drwxr-xr-x 2 jenkins jenkins 31 Dec 14 13:30 deploy
[root@ip-172-31-37-211 workspace]# cd build/
[root@ip-172-31-37-211 build]# ll
total 0
drwxr-xr-x 4 jenkins jenkins 153 Dec 14 13:35 terraform-jenkins1
[root@ip-172-31-37-211 build]# cd terraform-jenkins1/
[root@ip-172-31-37-211 terraform-jenkins1]# ll
total 20
-rw-r--r-- 1 jenkins jenkins 1011 Dec 14 13:08 ec2.tf
-rw-r--r-- 1 jenkins jenkins 181 Dec 14 13:35 terraform.tfstate
-rw-r--r-- 1 jenkins jenkins 9218 Dec 14 13:35 terraform.tfstate.backup
-rw-r--r-- 1 jenkins jenkins 1401 Dec 14 12:24 userdata.sh
[root@ip-172-31-37-211 terraform-jenkins1]#

```

- Here I have moved into build job and there we can see “terraform-jenkins1” repository having the terraform files.

The screenshot shows a Linux desktop environment with a terminal window and a web browser window.

**Terminal Window:**

```

root@ip-172-31-37-211:/var/lib/jenkins/workspace/build/terraform-jenkins1#
-rw-r--r-- 1 jenkins jenkins 297 Dec 14 13:14 io.jenkins.plugins.junit.storage.JunitTestResultStorageConfiguration.xml
-rw-r--r-- 1 jenkins jenkins 7 Dec 14 13:15 jenkins.fingerprints.GlobalFingerprintConfiguration.xml
-rw-r--r-- 1 jenkins jenkins 7 Dec 14 12:00 jenkins.install.InstallUtil.lastExecVersion
-rw-r--r-- 1 jenkins jenkins 260 Dec 14 13:14 jenkins.metrics.ArtifactManagementKey.xml
-rw-r--r-- 1 jenkins jenkins 153 Dec 14 13:14 jenkins.model.ArtifactManagementConfiguration.xml
-rw-r--r-- 1 jenkins jenkins 253 Dec 14 13:14 jenkins.model.GlobalBuildDiscarderConfiguration.xml
-rw-r--r-- 1 jenkins jenkins 241 Dec 14 13:14 jenkins.model.GlobalComputerRetentionCheckIntervalConfiguration.xml
-rw-r--r-- 1 jenkins jenkins 262 Dec 14 13:14 jenkins.model.JenkinsLocationConfiguration.xml
-rw-r--r-- 1 jenkins jenkins 86 Dec 14 13:14 jenkins.security.ResourceMainConfiguration.xml
-rw-r--r-- 1 jenkins jenkins 179 Dec 14 13:14 jenkins.tasks.filters.EnvVarsFilterGlobalConfiguration.xml
-rw-r--r-- 1 jenkins jenkins 171 Dec 14 11:57 jenkins.telemetry.correlator.xml
drwxr-xr-x 4 jenkins jenkins 31 Dec 14 13:32 jobs
drwxr-xr-x 2 jenkins jenkins 32 Dec 14 11:59 logs
-rw-r--r-- 1 jenkins jenkins 101 Dec 14 13:14 nodeMonitors.xml
-rw-r--r-- 1 jenkins jenkins 187 Dec 14 13:14 org.jenkinsci.plugins.displayurlapi.DefaultDisplayURLProviderGlobalConfiguration.xml
-rw-r--r-- 1 jenkins jenkins 289 Dec 14 13:14 org.jenkinsci.plugins.github_branch_source.GitHubConfig.xml
-rw-r--r-- 1 jenkins jenkins 167 Dec 14 13:14 org.jenkinsci.plugins.workflow.flow.GlobalDefaultFlowDurabilityLevel.xml
-rw-r--r-- 1 jenkins jenkins 229 Dec 14 13:14 org.jenkinsci.plugins.workflow.libs.GlobalLibraries.xml
-rw-r--r-- 1 jenkins jenkins 247 Dec 14 13:14 org.jenkinsci.plugins.workflow.libs.GlobalUntrustedLibraries.xml
drwxr-xr-x 99 jenkins jenkins 8192 Dec 14 12:35 plugins
-rw-r--r-- 1 jenkins jenkins 259 Dec 14 13:34 queue.xml
-rw-r--r-- 1 jenkins jenkins 259 Dec 14 13:08 queue.xml.bak
-rw-r--r-- 1 jenkins jenkins 64 Dec 14 13:34 secret.key
-rw-r--r-- 1 jenkins jenkins 64 Dec 14 11:57 secret.key.not-so-secret
drwxr-xr-x 2 jenkins jenkins 295 Dec 14 12:08 scripts
drwxr-xr-x 2 jenkins jenkins 149 Dec 14 11:58 updates
drwxr-xr-x 2 jenkins jenkins 24 Dec 14 11:57 userContent
drwxr-xr-x 4 jenkins jenkins 96 Dec 14 13:27 users
drwxr-xr-x 5 jenkins jenkins 45 Dec 14 12:44 workspace
[root@ip-172-31-37-211 jenkins]# cd workspace/
[root@ip-172-31-37-211 workspace]# ll
total 0
drwxr-xr-x 4 jenkins jenkins 44 Dec 14 12:24 build
drwxr-xr-x 3 jenkins jenkins 51 Dec 14 13:27 code
drwxr-xr-x 2 jenkins jenkins 31 Dec 14 13:30 deploy
[root@ip-172-31-37-211 workspace]# cd build/
[root@ip-172-31-37-211 build]# ll
total 0
drwxr-xr-x 4 jenkins jenkins 153 Dec 14 13:35 terraform-jenkins1
[root@ip-172-31-37-211 build]# cd terraform-jenkins1/
[root@ip-172-31-37-211 terraform-jenkins1]# ll
total 20
-rw-r--r-- 1 jenkins jenkins 1011 Dec 14 13:08 ec2.tf
-rw-r--r-- 1 jenkins jenkins 181 Dec 14 13:35 terraform.tfstate
-rw-r--r-- 1 jenkins jenkins 9218 Dec 14 13:35 terraform.tfstate.backup
-rw-r--r-- 1 jenkins jenkins 1401 Dec 14 12:24 userdata.sh
[root@ip-172-31-37-211 terraform-jenkins1]#

```

**Browser Window:**

The browser window shows the Jenkins dashboard. It displays a list of projects: "build" and "code". The "build" project has a last success at 2 min 59 sec and a last failure at 13 min. The "code" project has a last success at 3 min 7 sec and a last failure at N/A. Below the projects, there is a "Build Queue" section indicating "No builds in the queue".

- After specifying the configurations we need to build the jobs. I have build the code first, I have used “build other projects” so the “build project” job is also build after completion of code.

Instances (1/2) Info

| Name         | Instance ID         | Instance state | Instance type | Status check | Alarm status | Available |
|--------------|---------------------|----------------|---------------|--------------|--------------|-----------|
| jenkins      | i-0110bb2...        | Running        | t2.micro      | 2/2 checks p | View alarms  | us-east-1 |
| My Terraform | i-03ec3e98e76733005 | Running        | t2.micro      | Initializing | View alarms  | us-east-1 |

i-03ec3e98e76733005 (My Terraform)

Details Status and alarms Monitoring Security Networking Storage Tags

Instance summary

Instance ID: i-03ec3e98e76733005  
IPv6 address: -  
Hostname type: IP name: ip-172-31-55-201.ec2.internal

Public IPv4 address copied: 34.238.167.6 | open address

Private IP DNS name (IPv4 only): ip-172-31-55-201.ec2.internal

Public IPv4 addresses: 172.31.35.201  
Public IP4 DNS: ec2-34-238-167-6.compute-1.amazonaws.com | open address

- EC2 instance is created using the above configurations.

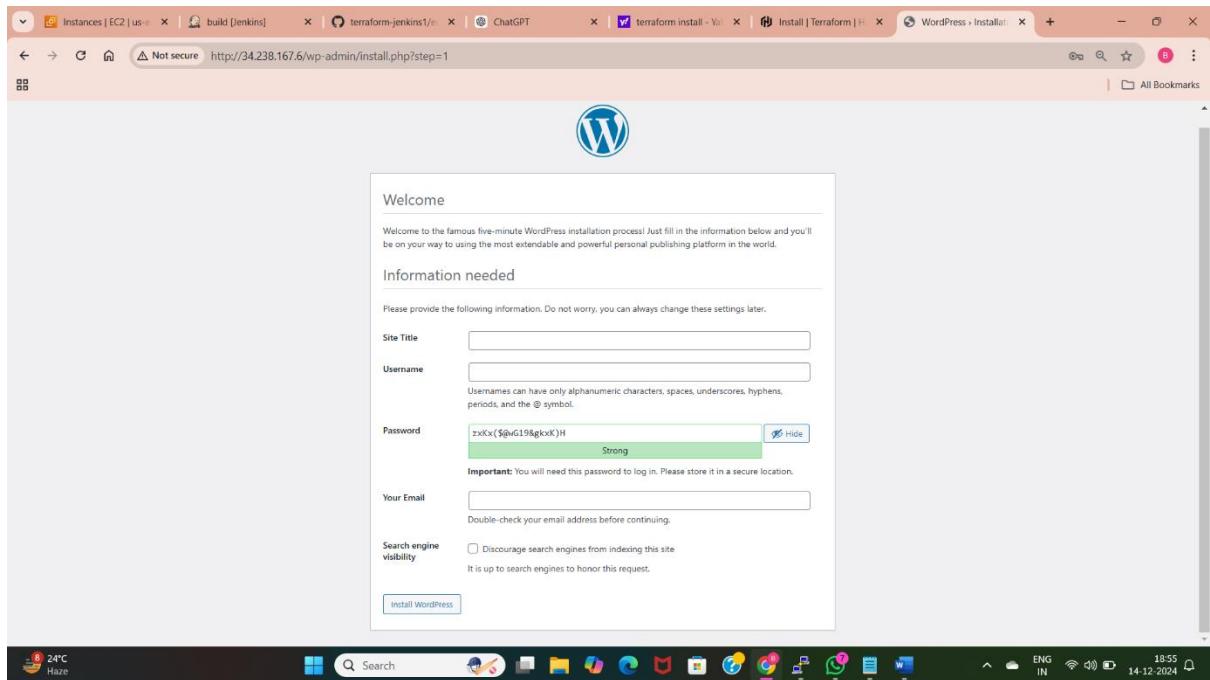
Not secure https://34.238.167.6/wp-admin/install.php

English (United States)

- Afrikaans
- മലയാളം
- Aragonés
- العربية
- অসমীয়া
- گۈزىنچىلىرىدىن
- Azerbaijani dili
- Беларуская мова
- Български
- বাংলা
- କୁଳି
- Bosanski
- Català
- Cebuano
- Čeština
- Cymraeg
- Dansk
- Deutsch (Schweiz, Du)
- Deutsch (Schweiz)

Continue

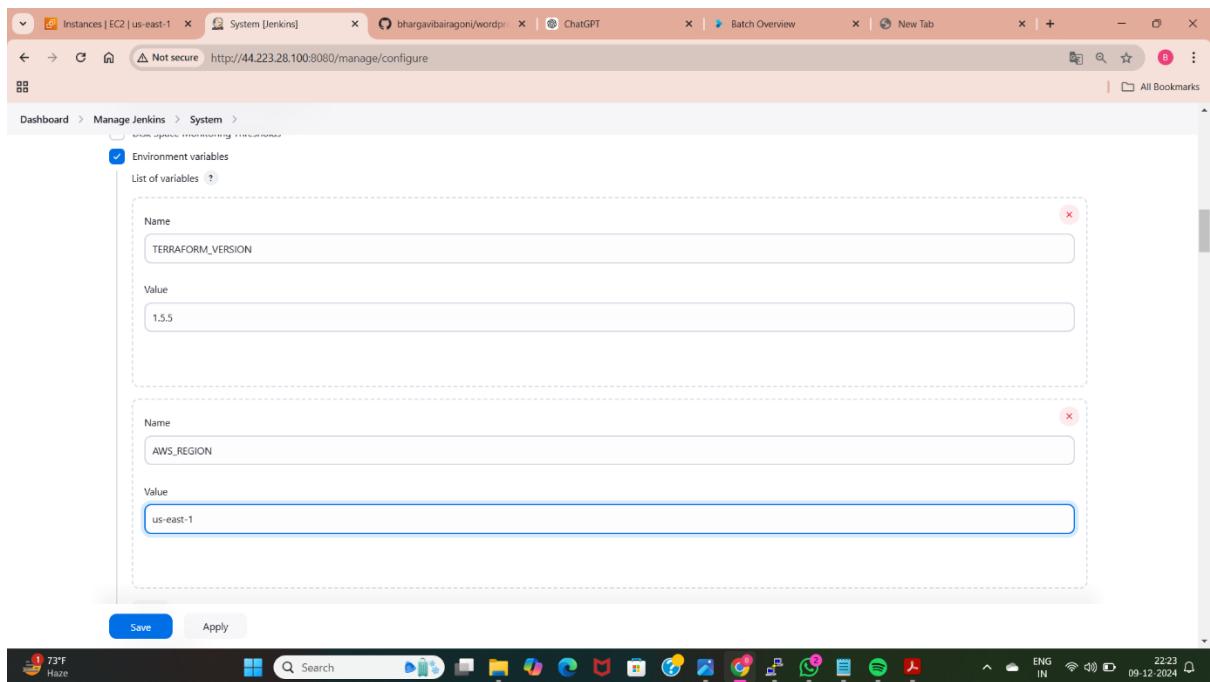
- Using the above created Public IP address we can access the deployed wordpress application.



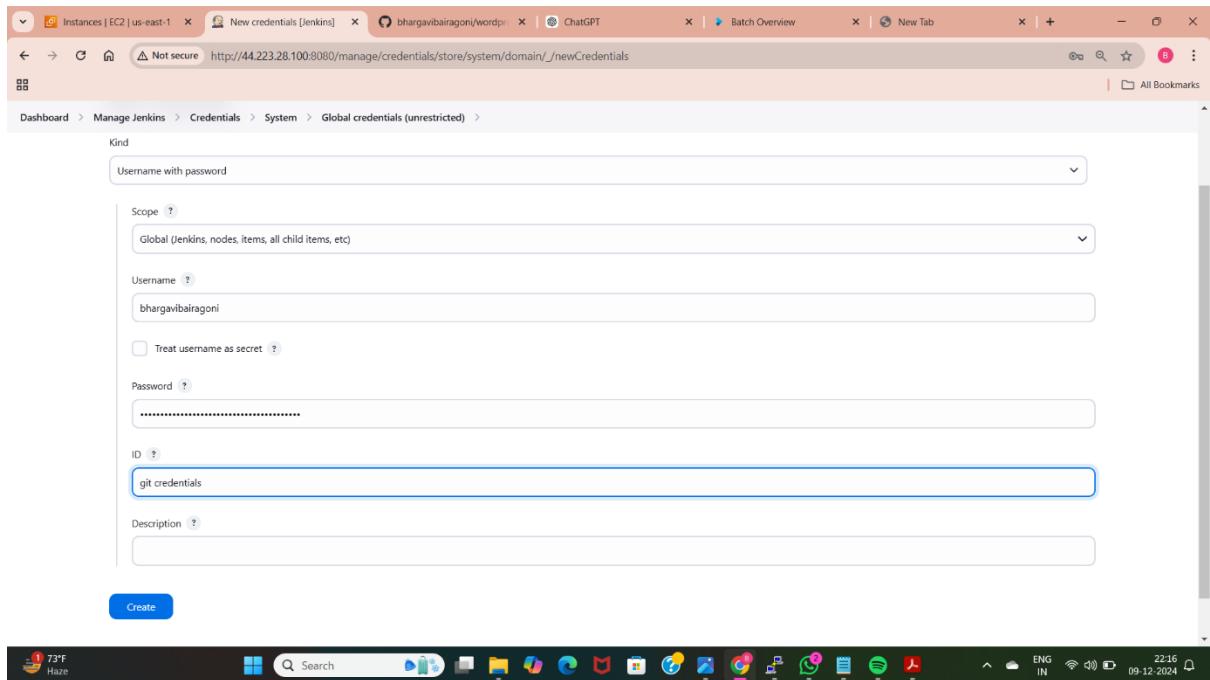
- I am able to access the wordpress application which is deployed using Jenkins freestyle pipeline.

We can also deploy this using the pipeline project.

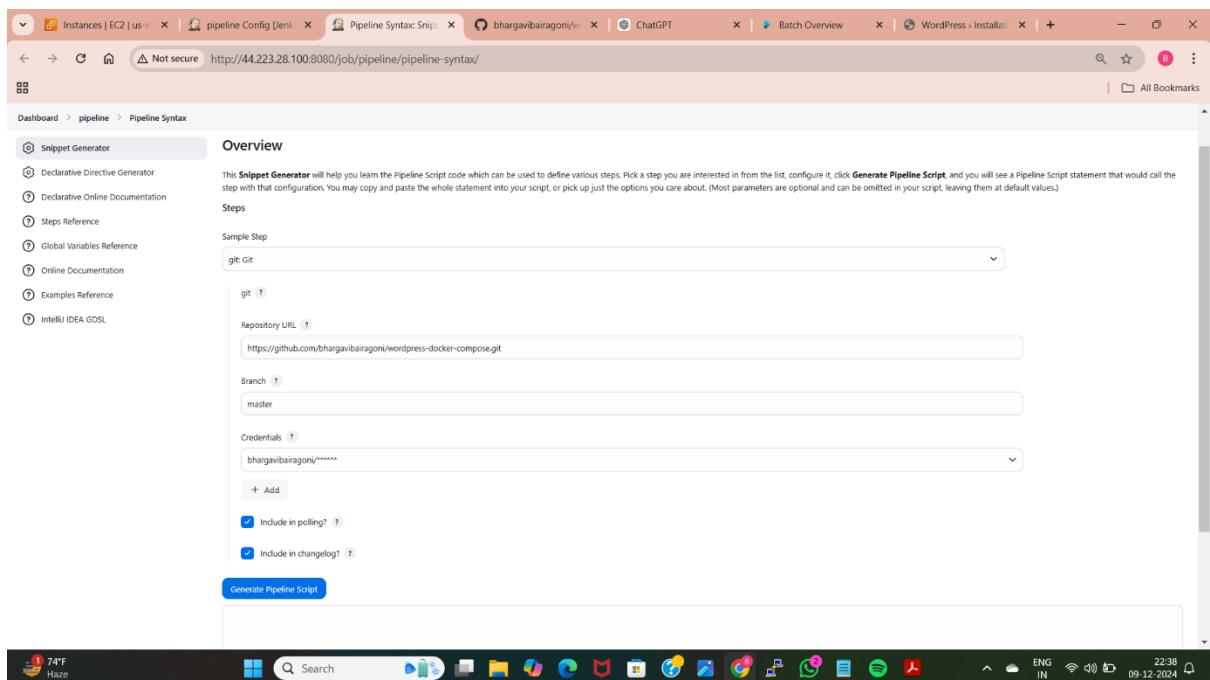
- I have created a job with the name of “pipeline” using pipeline project.



- We need to install terraform to deploy wordpress using terraform and Jenkins.
- I have given the version of the terraform as 1.5.5
- As we are using terraform code to deploy, I have created a code for launching EC2 instance with userdata. So we need to give Aws credentials to access Aws services using pipeline script.



- I have provided the git credentials using username with password. I have given username as my github account username and password as personal access token, id as git-credentials.



- I have generated script for code that is taking code from github using url, here I have provided above created credentials.

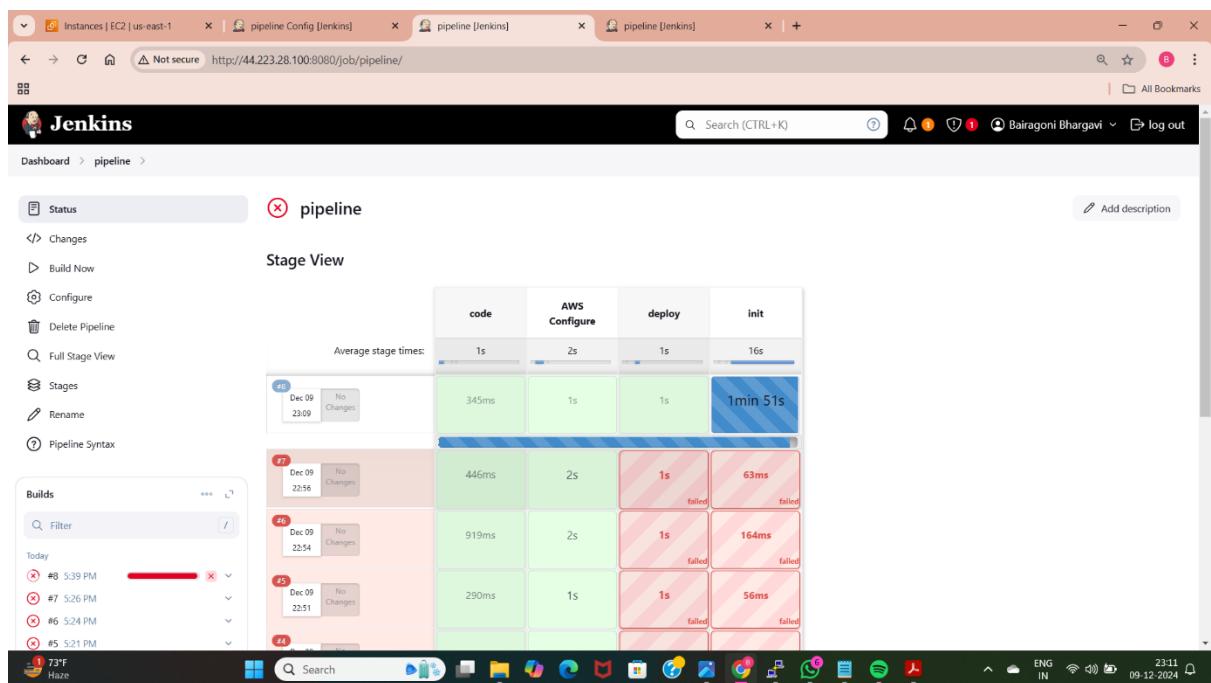
```

1+ pipeline {
2+ agent any
3+ environment {
4+ AWS_REGION='us-east-1'
5+ }
6+ triggers {
7+ pollSCM('H/15 * * *')
8+ }
9+ stages {
10+ stage('code') {
11+ steps {
12+ git branch: 'master', url:'https://github.com/bhargavibalragni/wordpress-docker-compose.git'
13+ }
14+ }
15+ stage('AWS Configure') {
16+ steps {
17+ script {
18+ withCredentials([
19+ awsAccessKeyVariable: 'AWS_ACCESS_KEY_ID',
20+ awsSecretKeyVariable: 'AWS_SECRET_ACCESS_KEY',
21+ awsDefaultRegionVariable: 'AWS_DEFAULT_REGION'
22+])
23+ sh 'export AWS_ACCESS_KEY_ID=$AWS_ACCESS_KEY_ID'
24+ sh 'export AWS_SECRET_ACCESS_KEY=$AWS_SECRET_ACCESS_KEY'
25+ sh 'export AWS_DEFAULT_REGION=$AWS_DEFAULT_REGION'
26+ }
27+ }
28+ }
29+ stage('Install') {
30+ steps {
31+ script {
32+ sudo yum install -y yum-utils shadow-utils
33+ sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
34+ sudo yum -y install terraform
35+ }
36+ }
37+ }
38+ stage('deploy') {
39+ steps {
40+ script {
41+ terraform init
42+ terraform validate
43+ terraform fmt
44+ terraform apply --auto-approve
45+ }
46+ }
47+ }
48+ }
49+ }

```

Save    Apply

- This is the pipeline script used to deploy wordpress application.
- First we need to take code from github using url and credentials if it is private repository.
- Next we need to provide aws credentials i.e access keys and secret access keys using aws credentials plugin.
- I have installed terraform using the stage “deploy”. To work with terraform we need to install terraform in our Jenkins server and newly launching instance.
- Next I have provided the terraform commands that are used to deploy wordpress application i.e first we need to initialize terraform in our created files, we need to validate the code whether the code is working or not, we need to format the code so that we will get indentation for our code. We need to apply the changes to create the resources.

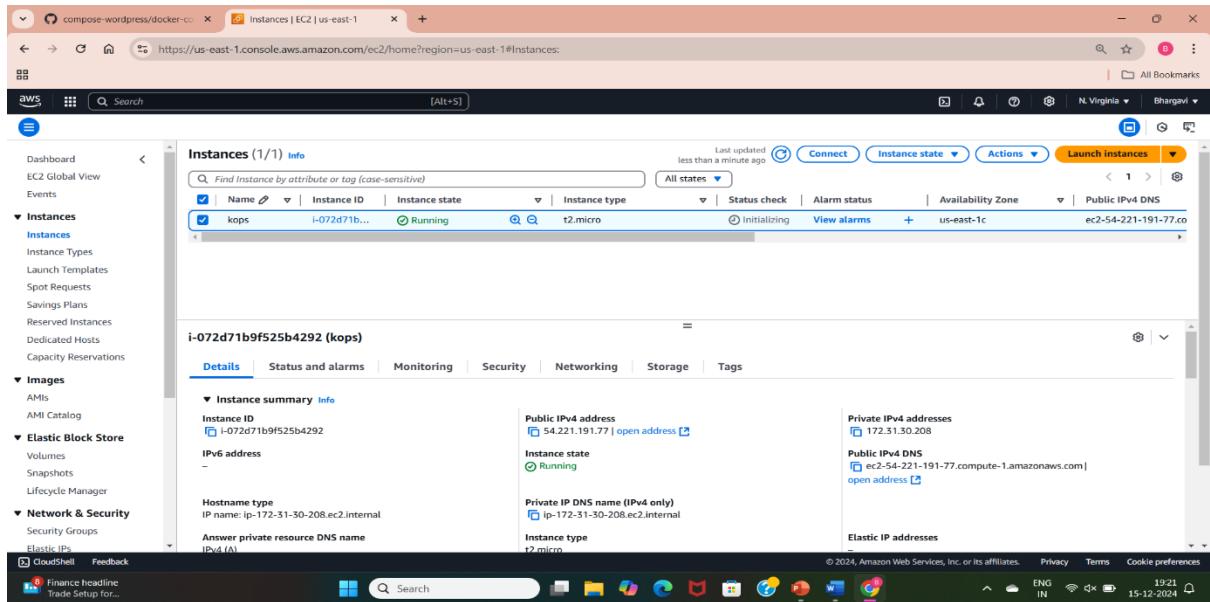


- After writing the pipeline script we need to build the pipeline so that we can access our wordpress application. Here I have used four stages in pipeline, three stages are successful and fourth stage is running.

## METHOD-10: DEPLOY WORDPRESS WEB APPLICATION BY USING K8'S

Kubernetes is an open-source container orchestration platform. It is used to automate the deployment, scaling, and management of containerized applications. It is also known as k8s. K8s simplifies the process of running applications in containers by providing tools and abstractions for managing infrastructure and application workloads efficiently.

- ✓ To setup the Kubernetes first we need to create a instance.



- ✓ I have created a instance with the name kops, connected with the terminal.

```

root@ip-172-31-30-208:~# curl -L https://dl.k8s.io/release/stable.txt | bin/linux/amd64/kubectl
 % Total % Received % Xferd Average Speed Time Time Current
 0 0 0 0 0 0 --:--:-- --:--:-- --:--:-- 2975
 100 138 109 138 0 0 2014 0 --:--:-- --:--:-- --:--:-- 2975
 100 94.6M 100 94.6M 0 0 90.5M 0 --:--:-- --:--:-- --:--:-- 90.5M
[ec2-user@ip-172-31-30-208 ~]# sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
[ec2-user@ip-172-31-30-208 ~]# kubectl version --client
Client Version: v1.32.0
Kustomize Version: v5.5.0
[root@ip-172-31-30-208 ~]# curl -Lo kops https://github.com/kubernetes/kops/releases/download/v1.27.1/kops-linux-amd64
 % Total % Received % Xferd Average Speed Time Time Current
 0 0 0 0 0 0 --:--:-- --:--:-- --:--:-- 0
 100 17M 100 17M 0 0 91.1M 0 0:00:01 0:00:01 --:--:-- 100M
[ec2-user@ip-172-31-30-208 ~]# chmod +x kops
[ec2-user@ip-172-31-30-208 ~]# sudo mv kops /usr/local/bin/
[ec2-user@ip-172-31-30-208 ~]# kops version
Client version: 1.27.1 (git-v1.27.1)
[ec2-user@ip-172-31-30-208 ~]# export KOPS_STATE_STORE=s3://bhargavi.kops.v1
[root@ip-172-31-30-208 ~]# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:LSRXl1ruthbEcMHC5vufol9WqS3Hxi+XqtmwY8C0 root@ip-172-31-30-208.ec2.internal
The key's randomart image:
+---[RSA 2048]----+
| . o o |
| . = o o + . |
| . = o o + . |
| o S o o + . |
| . + B = . . |
| . = * + * o . |
| = * + * o . |
| . O + B + . . |
| [SHA256]----+
[root@ip-172-31-30-208 ~]#

```

- ✓ Using the created instance I have installed kubectl(to interact with k8s cluster), kops(to create, destroy, and manage k8s clusters).
- ✓ We need to create a s3 bucket to store the cluster state and export into the Kubernetes so it stores the state of cluster.

```

root@ip-172-31-38-192:~#
 status code: 404, request id:
[root@ip-172-31-38-192 ~]# aws configure
AWS Access Key ID [None]: AKIA5FTFDZCQ7K4PA2J
AWS Secret Access Key [None]: lgvPFkVigh0xk340QiP7aaKCJ3KV48xW0TUx7ud
Default region name [None]: us-east-1
Default output format [None]:
[root@ip-172-31-38-192 ~]# kops create cluster --name bhargavi.k8s.local --zones us-east-1a --master-size t2.medium --node-size t2.micro --node-count=1
I1215 05:47:40.239192 3397 new cluster.go:1353] Cloud Provider ID: "aws"
I1215 05:47:40.347652 3397 subnets.go:185] Assigned CIDR 172.20.32.0/19 to subnet us-east-1a
Previewing changes that will be made:

A new kops version is available: 1.28.4
Upgrading is recommended
More information: https://github.com/kubernetes/kops/blob/master/permalinks/upgrade_kops.md#1.28.4

I1215 05:47:45.984053 3397 executor.go:111] Tasks: 0 done / 97 total; 44 can run
I1215 05:47:46.000000 3397 vts_keystorerreader.go:144] CA private key was not found
I1215 05:47:46.223360 3397 executor.go:111] Tasks: 44 done / 97 total; 24 can run
I1215 05:47:46.399362 3397 executor.go:111] Tasks: 63 done / 97 total; 24 can run
I1215 05:47:46.504203 3397 executor.go:111] Tasks: 87 done / 97 total; 2 can run
I1215 05:47:46.570349 3397 executor.go:111] Tasks: 89 done / 97 total; 4 can run
I1215 05:47:46.754000 3397 executor.go:111] Tasks: 93 done / 97 total; 4 can run
I1215 05:47:46.911142 3397 executor.go:111] Tasks: 97 done / 97 total; 0 can run
Will create resources:
AutoscalingGroup/control-plane-us-east-1a.masters.bhargavi.k8s.local
 Granularity 1Minute
 InstancePlacement false
 LoadBalancers name:control-plane-us-east-1a.masters.bhargavi.k8s.local
 LoadBalancers []
 MaxInstanceLifetime 0
 MaxSize 1
 Metrics [GroupDesiredCapacity, GroupInServiceInstances, GroupMaxSize, GroupMinSize, GroupPendingInstances, GroupStandbyInstances, GroupTerminatingInstances, GroupTotalInstances]
 MinSize 1
 Subnets [name:us-east-1a.bhargavi.k8s.local]
 SuspendProcesses []
 Tag k8s.io/role/control-plane: 1, k8s.k8s.io/instancegroup: control-plane-us-east-1a, Name: control-plane-us-east-1a.masters.bhargavi.k8s.local, k8s.io/cluster-autoscaler/node-template/label/kops.k8s.io/kops-controller:ki: k8s.io/cluster-autoscaler/node-template/label/node:role:kubernetes.io/control-plane:, k8s.io/role/master: 1, KubernetesCluster: bhargavi.k8s.local, kubernetes.io/cluster/bhargavi.k8s.local: owned, k8s.io/cluster-autoscaler/node-template/label/node:kubernetes.io/exclude-from-external-load-balancers:)
 TargetGroups [name:tcp-bhargavi-k8s-local-cs1jcl]
AutoscalingGroup/nodes-us-east-1a.bhargavi.k8s.local

```

- ✓ I have given the configurations to the aws by using aws configure command.
- ✓ I am creating the clusters using the kops.
- ✓ Created a master with t2.medium size and us-east-1 zone, node with t2.micro size and count is 1 means creating 1 node using the command “kops create cluster bhargavi.k8s.local –zones us-east-1a –master-size t2.medium –node-size t2.micro –node-count=1”.

| Name           | Instance ID   | Instance state | Instance type | Status check  | Alarm status | Availability Zone | Public IPv4 DNS       |
|----------------|---------------|----------------|---------------|---------------|--------------|-------------------|-----------------------|
| kops           | i-0dfa149...  | Running        | t2.micro      | 2/2 checks p: | View alarms  | us-east-1d        | ec2-34-230-49-204.co  |
| nodes-us-e...  | i-08b32d4f... | Running        | t2.micro      | 2/2 checks p: | View alarms  | us-east-1a        | ec2-44-200-59-162.co  |
| control-pla... | i-0bda9c95... | Running        | t2.medium     | 2/2 checks p: | View alarms  | us-east-1a        | ec2-44-199-255-196.co |

- ✓ With the help of above I have created two clusters one is master with the name “control-plane” other one is node with the name“node-us-east-1”.

```

root@ip-172-31-38-192:~#
* edit this cluster with: kops edit cluster bhargavi.k8s.local
* edit your node instance group: kops edit ig --name=bhargavi.k8s.local nodes-us-east-1a
* edit your control-plane instance group: kops edit ig --name=bhargavi.k8s.local control-plane-us-east-1a
Finally configure your cluster with: kops update cluster --name bhargavi.k8s.local --yes --admin
[root@ip-172-31-38-192 ~]# ^C
[root@ip-172-31-38-192 ~]# kops update cluster --name bhargavi.k8s.local --yes --admin

A new kops version is available: 1.28.4
Upgrading is recommended
More information: https://github.com/kubernetes/kops/blob/master/permalinks/upgrade_kops.md#1.28.4

11/21 05:48:33.641568 3400 executor.go:1111 Tasks: 0 done / 97 total; 44 can run
11/21 05:49:30.756772 3400 vts_keystorereader.go:143] CA private key was not found
11/21 05:48:33.760032 3400 keypair.go:226] Issuing new certificate: "etcd-clients-ca"
11/21 05:48:33.796827 3400 keypair.go:226] Issuing new certificate: "etcd-peers-ca-main"
11/21 05:48:33.816972 3400 keypair.go:226] Issuing new certificate: "apiserver-aggregator-ca"
11/21 05:48:33.836972 3400 keypair.go:226] Issuing new certificate: "etcd-peer-client-cert"
11/21 05:49:33.859148 3400 keypair.go:226] Issuing new certificate: "etcd-manager-ca-events"
11/21 05:48:33.960514 3400 keypair.go:226] Issuing new certificate: "etcd-peers-ca-events"
11/21 05:48:34.513444 3400 vts_keystorereader.go:143] CA private key was not found
11/21 05:48:34.533132 3400 keypair.go:226] Issuing new certificate: "kubernetes-ca"
11/21 05:48:34.633712 3400 keypair.go:226] Issuing new certificate: "kubernetes-client"
11/21 05:48:35.933132 3400 executor.go:1111 Tasks: 44 done / 97 total; 19 can run
11/21 05:48:36.977321 3400 executor.go:1111 Tasks: 63 done / 97 total; 24 can run
11/21 05:48:37.016200 3400 executor.go:1111 Tasks: 89 done / 97 total; 4 can run
11/21 05:48:38.106207 3400 executor.go:1111 Tasks: 89 done / 97 total; 4 can run
11/21 05:48:38.673164 3400 executor.go:1111 Tasks: 93 done / 97 total; 4 can run
11/21 05:48:38.648952 3400 executor.go:1111 Tasks: 95 done / 97 total; 2 can run
11/21 05:48:40.935358 3400 executor.go:1111 Tasks: 95 done / 97 total; 2 can run
11/21 05:48:41.990781 3400 executor.go:1111 Tasks: 97 done / 97 total; 0 can run
11/21 05:48:42.039207 3400 update_cluster.go:323] Exporting kubeconfig for cluster
Kops has set your kubectl context to bhargavi.k8s.local

Cluster is starting. It should be ready in a few minutes.

Suggestions:
* validate cluster: kops validate cluster --wait 10m
* list nodes: kubectl get nodes --show-labels
* ssh to a control-plane node: ssh -l ~/.ssh/id_rsa ubuntu#
* the ubuntu user is specific to Ubuntu. If not using Ubuntu please use the appropriate user based on your OS.
* read about installing addons at: https://kops.sigs.k8s.io/addons.

[root@ip-172-31-38-192 ~]#

```

- ✓ We need to update the created cluster with “`kops update cluster –name bhargavi.k8s.local –yes –admin`”.

```

ec2-user@slave1:~/workspace/slave1
apiVersion: apps/v1
kind: Deployment
metadata:
 name: word-deployment
spec:
 replicas: 2
 selector:
 matchLabels:
 tier: wordpress
 template:
 metadata:
 labels:
 tier: wordpress
 spec:
 containers:
 - name: wordpress-container
 image: bhargavibairagoni/wordpress:wordimage
 ports:
 - containerPort: 80

apiVersion: v1
kind: Service
metadata:
 name: word-service
spec:
 selector:
 tier: wordpress
 ports:
 - protocol: TCP
 port: 80 # External port
 targetPort: 80 # Port inside container
 type: LoadBalancer

```

- ✓ This is the deployment and service file for wordpress. It is running on the port number 80 with two replicas.

```

ec2-user@slave1:~/workspace/slave1

apiVersion: apps/v1
kind: Deployment
metadata:
 name: mysql-deployment
spec:
 replicas: 2
 selector:
 matchLabels:
 tier: mysql
 template:
 metadata:
 labels:
 tier: mysql
 spec:
 containers:
 - name: mysql
 image: bhargavibairagoni/mysql:mysqlimage
 ports:
 - containerPort: 3306
 env:
 - name: MYSQL_ROOT_PASSWORD
 value: "wordpress"
 - name: MYSQL_DATABASE
 value: "databaseword"
 - name: MYSQL_USER
 value: "admin"
 - name: MYSQL_PASSWORD
 value: "ADMIN123"

apiVersion: v1
kind: Service
metadata:
 name: mysql-service
spec:
 selector:
 tier: mysql
 ports:
 - protocol: TCP
 port: 3306
 targetPort: 3306
 type: LoadBalancer

```

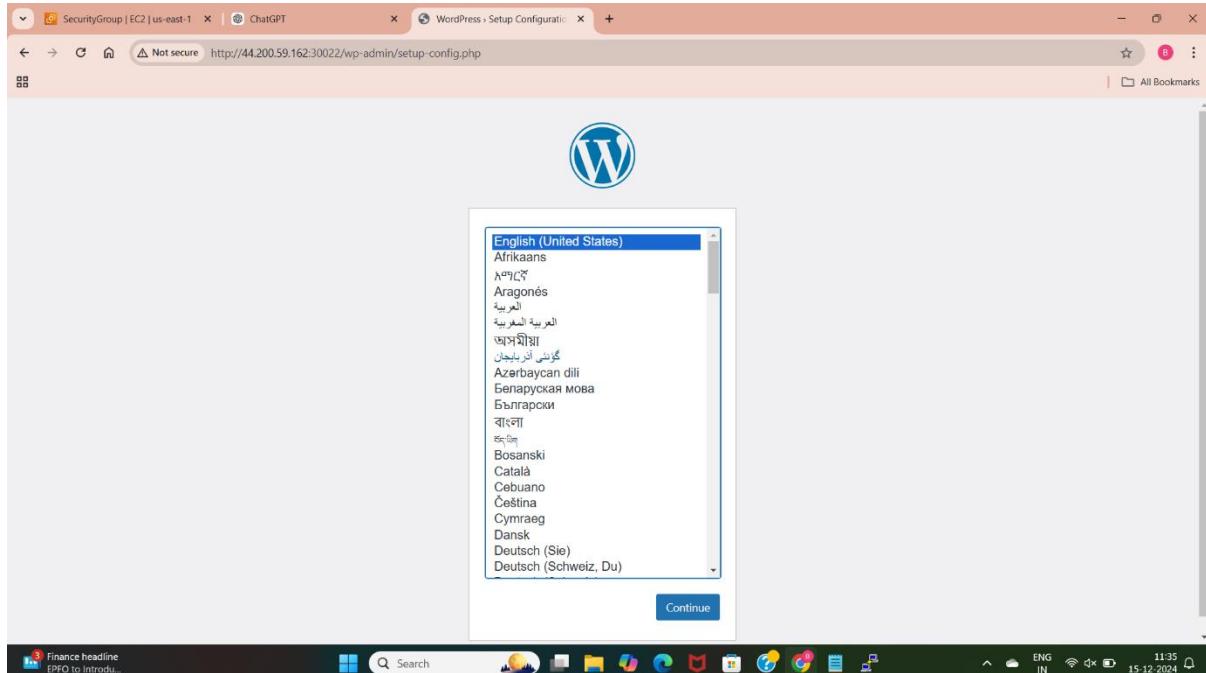
- ✓ I have created a manifest file with the deployment and service.
- ✓ The deployment file is used to create a pod with the containers.
- ✓ I have created a deployment with the name wordpress, created a container with the name wordpress and taken the image from the dockerhub “bhargavibairagoni/wordpress:wordpimage”.
- ✓ As am taking the replica of the pod I haved created the template of the pod so that if new pod is created the above mentioned configurations are created inside the pod and the container.
- ✓ We need to give the port number to the container.

```

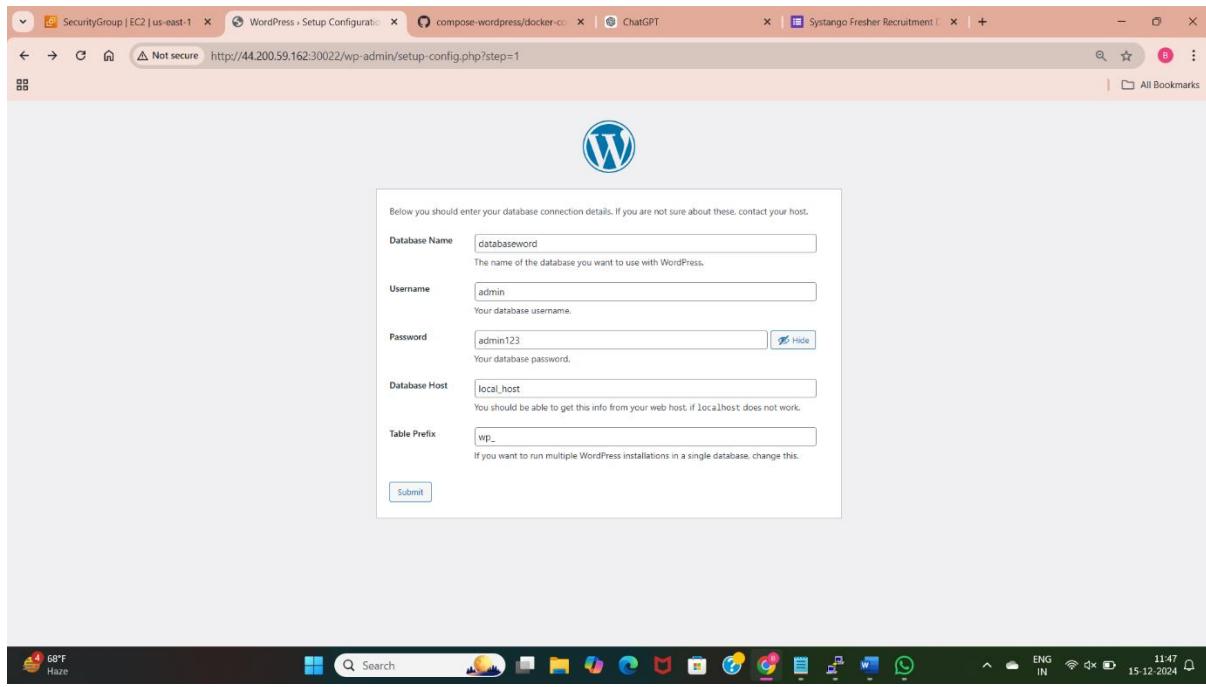
root@ip-172-31-30-208:~# kubectl get nodes
NAME STATUS ROLES AGE VERSION
i-052d929ef73aaab Ready node 59s v1.27.16
i-0b526504248aeas73 Ready control-plane 2m36s v1.27.16
[root@ip-172-31-30-208 ~]# vim wordpress.yaml
[root@ip-172-31-30-208 ~]# kubectl create -f wordpress.yaml
[replicaset.apps/wordpress created]
service/wordpress-service created
[root@ip-172-31-30-208 ~]# kubectl get svc
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
kubernetes ClusterIP 100.64.0.1 <none> 443/TCP 3m2s
wordpress-service NodePort 100.69.5.23 <none> 80:30022/TCP 6s
[root@ip-172-31-30-208 ~]# kubectl get deployments
NAME READY UP-TO-DATE AVAILABLE AGE
wordpress 0/1 0 0 15s
[root@ip-172-31-30-208 ~]# kubectl get deployments
NAME READY UP-TO-DATE AVAILABLE AGE
wordpress 1/1 1 1 24s
[root@ip-172-31-30-208 ~]# kubectl get pods
NAME READY STATUS RESTARTS AGE
wordpress-b849cb885-xpcgw 1/1 Running 0 30s
[root@ip-172-31-30-208 ~]#

```

- ✓ The container port is used to take the traffic and the port no. 80 is listening the traffic from the users.
- ✓ I have created a service file with the name “wordpress”, I have used the nodeport service type with the target port 80, and node port 30022.
- ✓ The target port is used to route the traffic to the clusters and nodeport is used to expose the applications to external access..
- ✓ Port is used inside the Kubernetes cluster.



- ✓ After deploying the application using the “nodeport” service we can able the access the application with the port number 30022 and the ip address of the created clusters.



- ✓ Here am getting the wordpress application using the public ip address of the cluster and the port number we used in the manifest file.