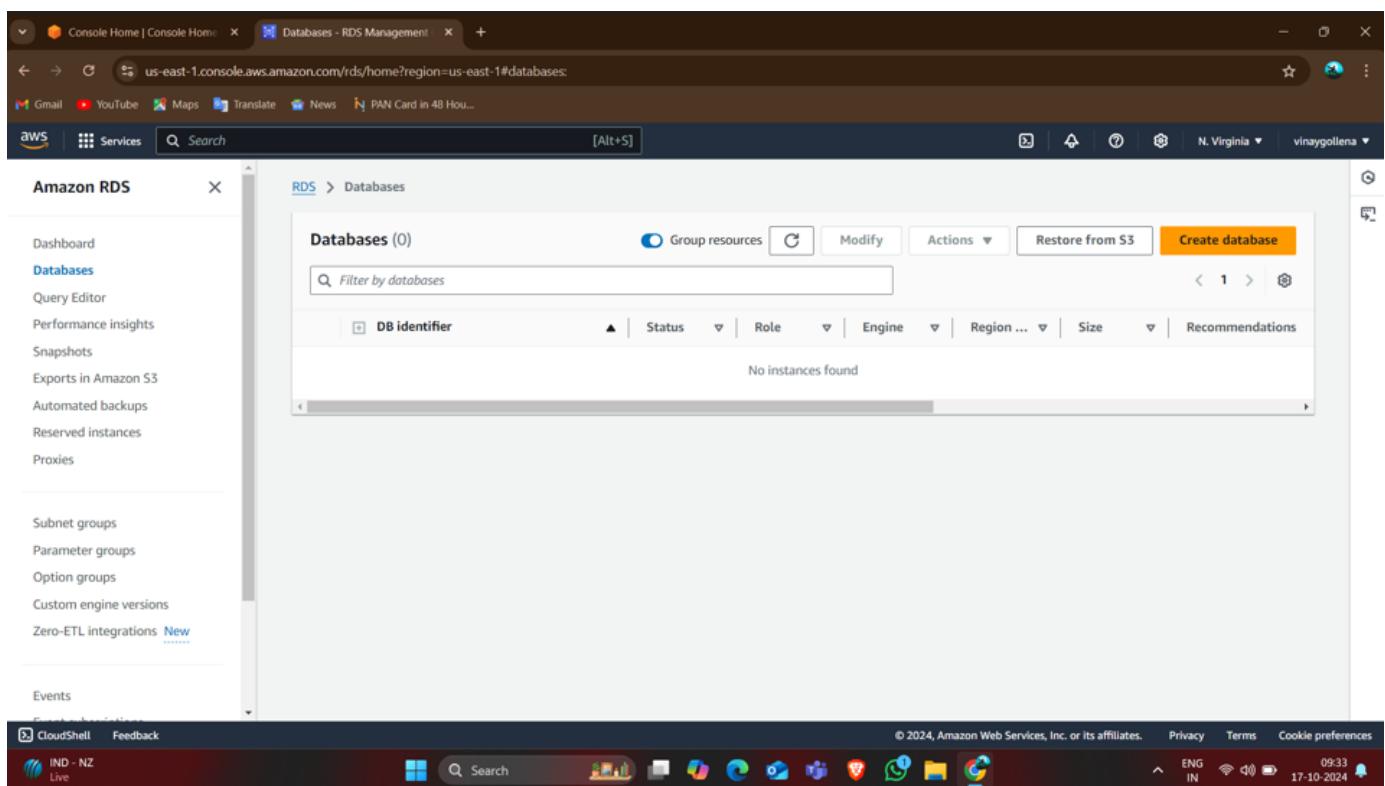


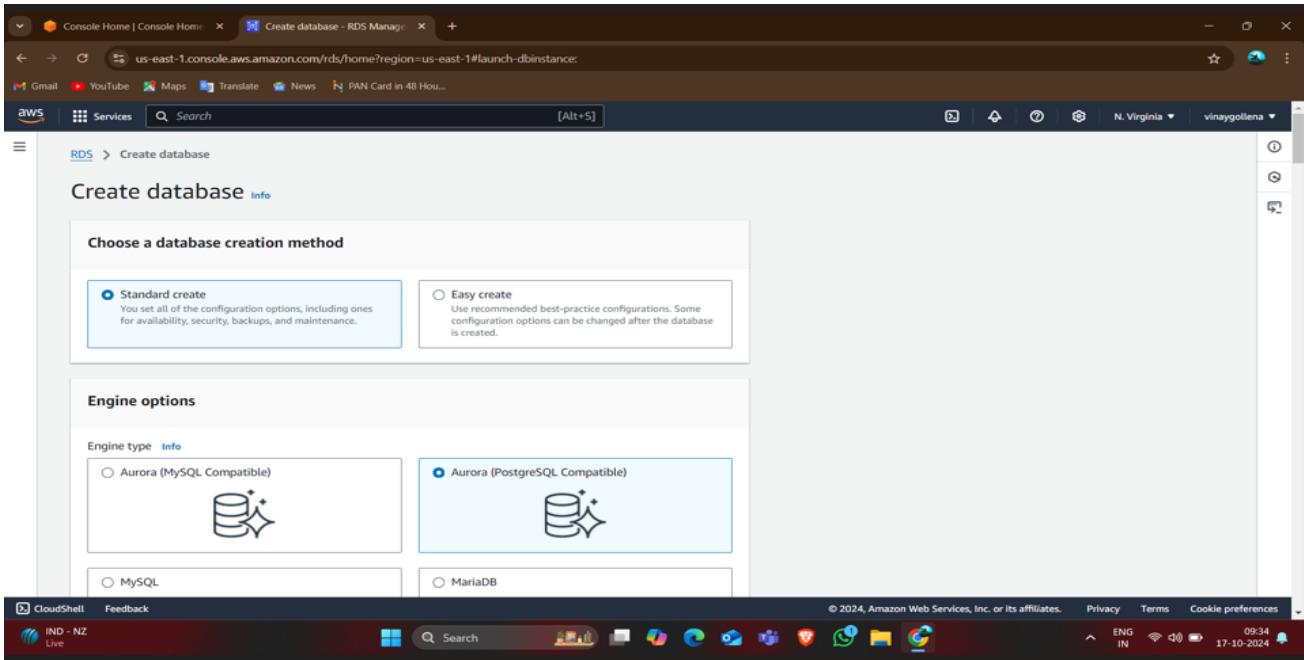
# WordPress Working Methods

## Deploy WordPress web application by using AWS RDS(MYSQL) service (manually).

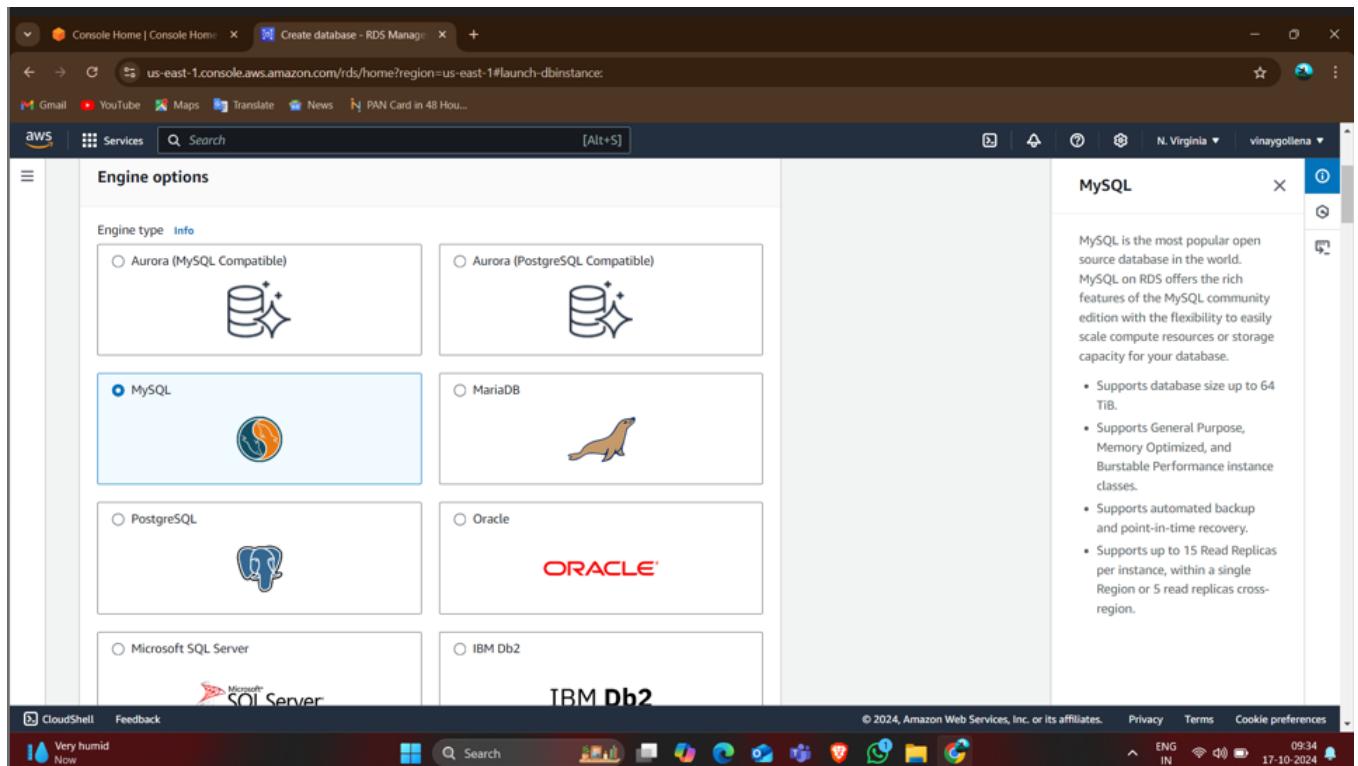
- Now first go to aws account and log into aws and search RDS and click on create database.



- Now select the database creation method (select standard method) because by selecting easy creation methods then its disables free tier template.



- Now select the database engine with version but I select the MySQL database engine because it's a huge usage in real time organizations.



- Now select the template as free tier and by selecting the free tier it disables the availability zone section options.

□

The screenshot shows the AWS RDS Management console for creating a new database. On the left, under 'Templates', the 'Free tier' option is selected. On the right, the MySQL engine details are shown, including its popularity and various features like support for up to 64 TiB and automated backups. The central area displays deployment options for MySQL, including Multi-AZ DB Cluster, Multi-AZ DB instance (not supported for cluster snapshot), and Single DB instance (not supported for cluster snapshot). The bottom navigation bar includes CloudShell, Feedback, and various system icons.

- Now give some name to your database and give username and password as credentials for your databases.

The screenshot shows the continuation of the AWS RDS Management console setup. In the 'Settings' section, the 'DB instance identifier' is set to 'wordpress-db'. Under 'Credentials Settings', the 'Master username' is 'admin'. For 'Credentials management', the 'Self managed' option is selected, indicating the user will create their own password. The right panel continues to display MySQL engine details. The bottom navigation bar includes CloudShell, Feedback, and various system icons.



- Now select the storage type as general purpose (SSD (gp2)) and enter the storage values as maximum (1000gb) and minimum (20gb).

The screenshot shows the 'Create database - RDS Manager' wizard on the AWS console. The 'Storage' step is selected. Under 'Storage type', 'General Purpose SSD (gp2)' is chosen. The 'Allocated storage' field is set to 20 GiB. A note indicates that after modification, the DB instance will be in storage-optimization. On the right, the MySQL service details are listed, including its popularity and various features like automated backups and up to 15 read replicas.

- Now select your created vpc or select default vpc and it automatically selects the database subnet group.

The screenshot shows the 'Create database - RDS Manager' wizard on the AWS console. The current step is 'Launch DB instance'. The 'Compute resource' section is expanded, showing two options: 'Don't connect to an EC2 compute resource' (selected) and 'Connect to an EC2 compute resource'. The 'Virtual private cloud (VPC)' section shows a dropdown set to 'Default VPC (vpc-023875e26f8c7b0cb)'. A note states: 'After a database is created, you can't change its VPC.' The 'DB subnet group' section shows a dropdown set to 'default'. The 'Public access' section has 'Yes' selected. On the right, the MySQL service details are shown, including its popularity and supported features like automated backups and up to 15 read replicas. The bottom navigation bar includes CloudShell, Feedback, and various system icons.

- Now give the name of the database which you give at the stage of DB instance identifier enter same name here.

The screenshot shows the AWS RDS Management console for creating a new database. The 'Additional configuration' section is expanded, showing:

- Database options**: Initial database name is set to `wordpressdb`. A note says: "If you do not specify a database name, Amazon RDS does not create a database."
- DB parameter group**: Set to `default.mysql8.0`.
- Option group**: Set to `default:mysql-8-0`.
- Backup**:  **Enable automated backups**. A note says: "Creates a point-in-time snapshot of your database". A warning message states: "⚠ Please note that automated backups are currently supported for InnoDB storage engine only. If you are using MyISAM, refer to details [here](#)."

The right panel displays information about MySQL, stating it is the most popular open source database and listing its features.

- Now click on the create database button and will create the MYSQL database.

The screenshot shows the AWS RDS Databases management console. The left sidebar is collapsed. The main area displays the 'Databases' list:

DB identifier	Status	Role	Engine	Region	Size	Recommendations
<code>wordpressdb</code>	Available	Instance	MySQL Co...	us-east-1d	db.t3.micro	

A yellow callout box points to the **Create database** button at the top right of the table header.

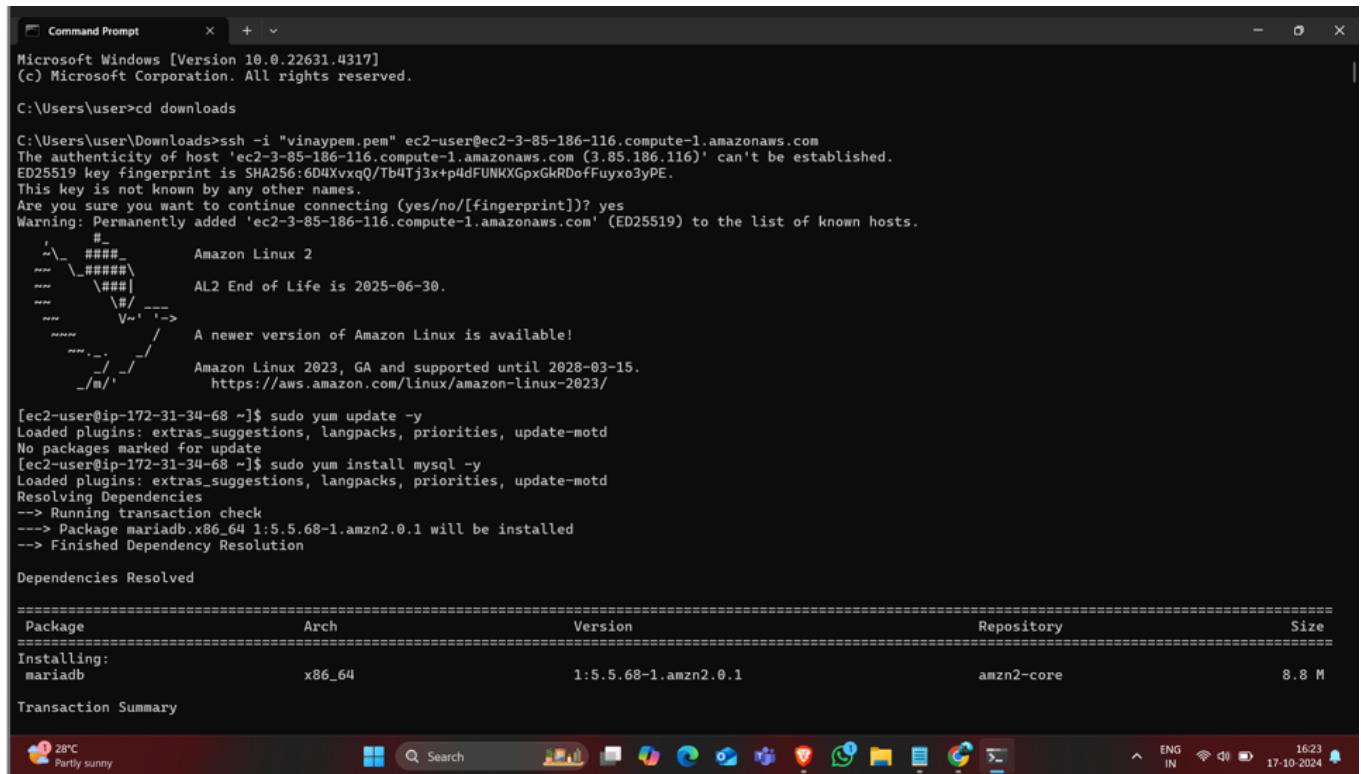
- Now create the ec2 instance by selecting ec2 instance and launch the instance by selecting Amazon Linux -2 version and giving security group with ssh(22) and http(80).

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with navigation links like EC2 Dashboard, EC2 Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity, Reservations, Images (selected), AMIs, AMI Catalog, Elastic Block Store, Volumes, and Snapshots. The main content area displays a table titled 'Instances (1/1) Info'. The table has columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IPv4. One row is selected, showing 'wp' with Instance ID i-04d6673a37e75ed2c, Running state, t2.micro type, 2/2 checks passed, us-east-1b zone, and Public IP 3.85.186.116. Below the table, a detailed view for instance 'i-04d6673a37e75ed2c (wp)' is shown with tabs for Details, Status and alarms, Monitoring, Security, Networking, Storage, and Tags. Under 'Details', the 'Instance summary' section shows Instance ID i-04d6673a37e75ed2c (wp), IPv6 address -, Hostname type IP name: ip-172-31-34-68.ec2.internal, and Public IP DNS name (IPv4 only) ip-172-31-34-68.ec2.internal. It also lists Private IP4 addresses 172.31.34.68 and Public IP4 DNS ec2-3-85-186-116.compute-1.amazonaws.com.

This screenshot is similar to the previous one but focuses on the 'Inbound rules' section. The sidebar and main table are identical. In the 'Inbound rules' section, there's a 'Filter rules' input field and a table with columns: Name, Security group rule ID, Port range, Protocol, Source, and Security groups. Two entries are listed: one for port 22 (TCP) with rule ID sgr-0849a779092d39d66 and another for port 80 (TCP) with rule ID sgr-0ae971a192ee023e1. Both rules have a source of 0.0.0.0/0 and belong to the security group 'launch-wizard-6'.

- Now connect the virtual server through the terminal shown in fig.

- Now update the Linux version by using command <sudo yum update -y> and install MySQL by using the command sudo yum install -y mysql2



```

Command Prompt
Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user>cd downloads

C:\Users\user\Downloads>ssh -i "vinay.pem" ec2-user@ec2-3-85-186-116.compute-1.amazonaws.com
The authenticity of host 'ec2-3-85-186-116.compute-1.amazonaws.com (3.85.186.116)' can't be established.
ED25519 key fingerprint is SHA256:6D4XvxqQ/Tb4Tj3x+p4dFUNKXGpxGkRDOfFuyxo3yPE.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-3-85-186-116.compute-1.amazonaws.com' (ED25519) to the list of known hosts.

_
  _###_      Amazon Linux 2
  _###_#
  _##| AL2 End of Life is 2025-06-30.
  _##| #/ --.
  _##| V~!`-->
  _##| / 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-172-31-34-68 ~]$ sudo yum update -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
No packages marked for update
[ec2-user@ip-172-31-34-68 ~]$ sudo yum install mysql -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Resolving Dependencies
--> Running transaction check
--> Package mariadb.x86_64 1:5.5.68-1.amzn2.0.1 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

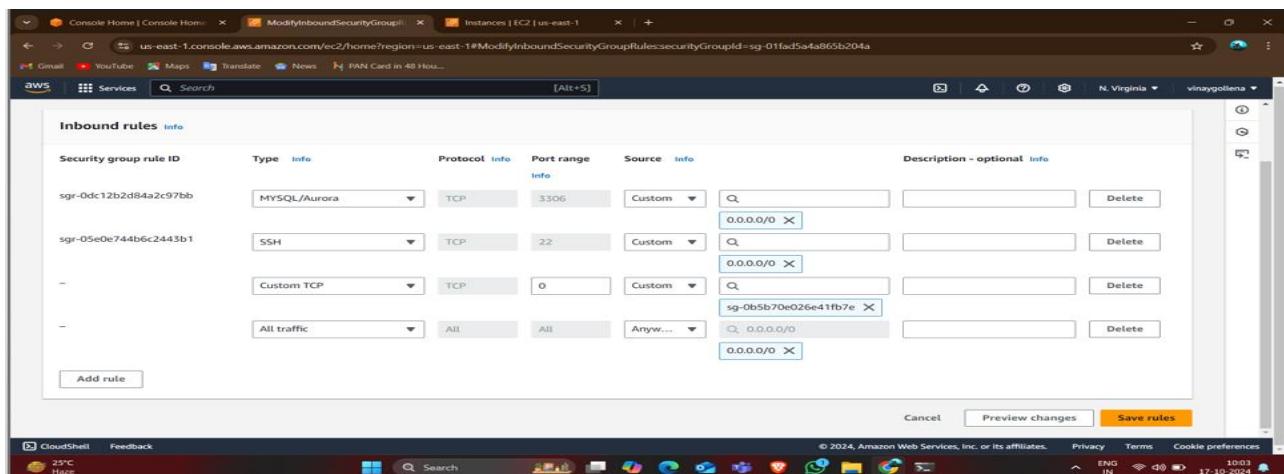
=====
Package           Arch         Version          Repository        Size
=====
Installing:
mariadb          x86_64      1:5.5.68-1.amzn2.0.1   amzn2-core       8.8 M

Transaction Summary

28°C Partly sunny
Search
ENG IN 1623 17-10-2024

```

- Now to allow create traffic from ec2 instance into rds database for that go to rds service and select your database.
- Now go inside the created database and go to the security under this option these is security group id click on that.





- 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 address and select and copy the endpoint address
- Now give the credentials of database by giving the command as “MySQL – user=<username> -password=<password> database=”name”
- There is another method command to access your database as “MySQL -h <endpoint address> -u <user> -p” press enter button it asks the password to enter it, and press enter button

Now create a database user for your WordPress application and give it permission to access the

WordPress database. By using these commands as

- a. CREATE USER WordPress IDENTIFIED BY ‘WordPress-pass’
- b. GRANT ALL PRIVILEGE ON WordPress.\* TO WordPress
- c. FLUSH PRIVILEGES
- d. EXIT

```
^C[ec2-user@ip-172-31-34-68 ~]$ mysql --user=admin --password=9966774360 wordpressdb
ERROR 2003 (HY000): Can't connect to MySQL server on 'wordpressdb.cfayqi0o8ulz.us-east-1.rds.amazonaws.com' (110)
[ec2-user@ip-172-31-34-68 ~]$ mysql --user=admin --password=9966774360 wordpressdb
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 8.0.35 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 [wordpressdb]> CREATE database wordpressdb;
ERROR 1007 (HY000): Can't create database 'wordpressdb'; database exists
MySQL [wordpressdb]> CREATE USER 'admin'@'localhost' IDENTIFIED BY '9966774360';
Query OK, 0 rows affected (0.01 sec)

MySQL [wordpressdb]> GRANT ALL PRIVILEGES ON wordpressdb.* TO 'admin'@'localhost';
Query OK, 0 rows affected (0.00 sec)

MySQL [wordpressdb]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.01 sec)

MySQL [wordpressdb]> EXIT;
Bye
[ec2-user@ip-172-31-34-68 ~]$ sudo yum install httpd -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core
Resolving Dependencies
--> Running transaction check
--> Package httpd.x86_64 0:2.4.62-1.amzn2.0.2 will be installed
--> Processing Dependency: httpd-filesystem = 2.4.62-1.amzn2.0.2 for package: httpd-2.4.62-1.amzn2.0.2.x86_64
--> Processing Dependency: httpd-tools = 2.4.62-1.amzn2.0.2 for package: httpd-2.4.62-1.amzn2.0.2.x86_64
--> Processing Dependency: /etc/mime.types for package: httpd-2.4.62-1.amzn2.0.2.x86_64
--> Processing Dependency: httpd-filesystem for package: httpd-2.4.62-1.amzn2.0.2.x86_64
--> Processing Dependency: mod_http2 for package: httpd-2.4.62-1.amzn2.0.2.x86_64
--> Processing Dependency: system-logos-httpd for package: httpd-2.4.62-1.amzn2.0.2.x86_64
--> Processing Dependency: libapr-1.so.0()(64bit) for package: httpd-2.4.62-1.amzn2.0.2.x86_64
```



- To host WordPress application need a Apache web server httpd install that by giving command as <sudo yum install httpd -y> and start and enable the httpd service by giving the command as (a). “sudo service httpd start (or) sudo systemctl start httpd (b). Sudo chkconfig httpd on (or) sudo systemctl enable httpd

```
Verifying : httpd-tools-2.4.62-1.amzn2.0.2.x86_64
Verifying : httpd-filesystem-2.4.62-1.amzn2.0.2.noarch
8/9
9/9

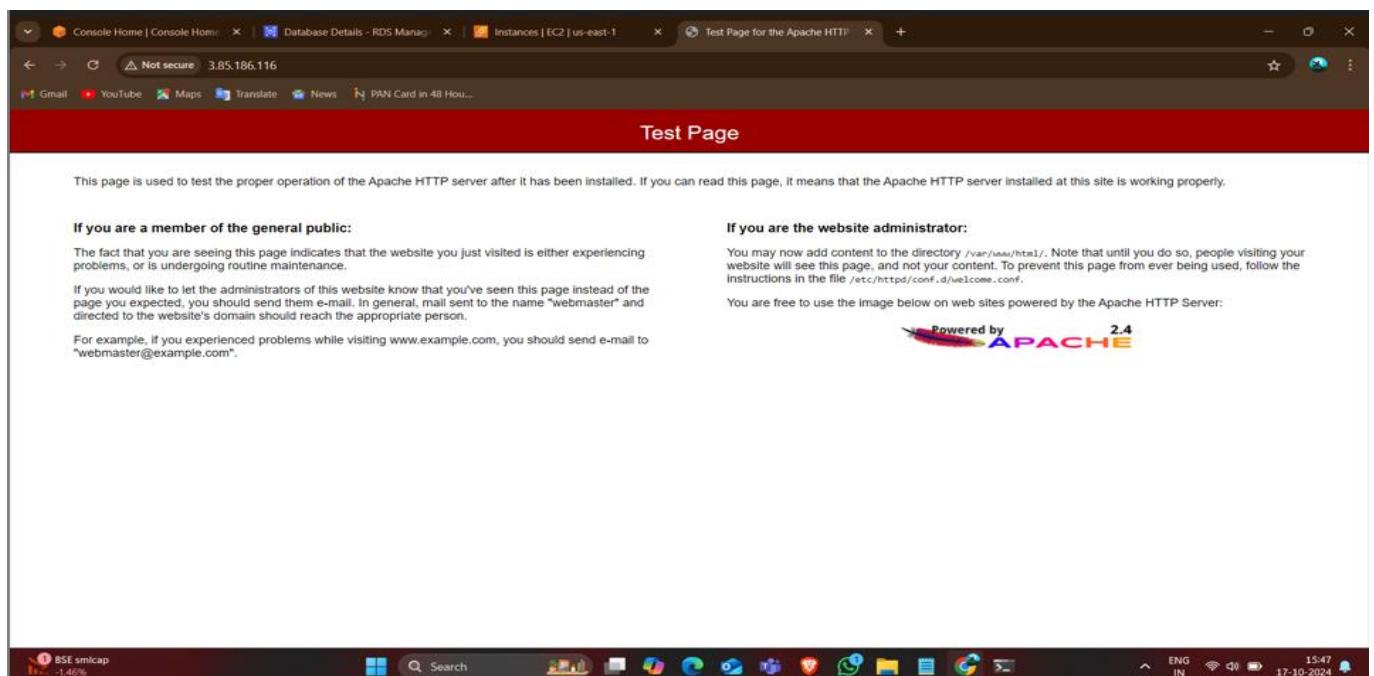
Installed:
httpd.x86_64 0:2.4.62-1.amzn2.0.2

Dependency Installed:
apr.x86_64 0:1.7.2-1.amzn2
generic-logos-httpd.noarch 0:18.0.0-4.amzn2
mailcap.noarch 0:2.1.41-2.amzn2
apr-util.x86_64 0:1.6.3-1.amzn2.0.1
httpd-filesystem.noarch 0:2.4.62-1.amzn2.0.2
mod_http2.x86_64 0:1.15.19-1.amzn2.0.2
apr-util-bdb.x86_64 0:1.6.3-1.amzn2.0.1
httpd-tools.x86_64 0:2.4.62-1.amzn2.0.2

Complete!
[ec2-user@ip-172-31-34-68 ~]$ sudo yum start httpd
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
No such command: start. Please use /bin/yum --help
[ec2-user@ip-172-31-34-68 ~]$ sudo systemctl start httpd
[ec2-user@ip-172-31-34-68 ~]$ sudo systemctl enable httpd
Created symlink from /etc/systemd/system/multi-user.target.wants/httpd.service to /usr/lib/systemd/system/httpd.service.
[ec2-user@ip-172-31-34-68 ~]$ sudo systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset: disabled)
     Active: active (running) since Thu 2024-10-17 10:16:18 UTC; 18s ago
       Docs: man:httpd.service(8)
 Main PID: 3480 (httpd)
   Status: "Total requests: 0; Idle/Busy workers 100/0;Requests/sec: 0; Bytes served/sec: 0 B/sec"
  CGrou...
[ec2-user@ip-172-31-34-68 ~]$ wget https://wordpress.org/latest.zip
--2024-10-17 10:18:39-- https://wordpress.org/latest.zip
Resolving wordpress.org (wordpress.org)... 198.143.164.252
Connecting to wordpress.org (wordpress.org)|198.143.164.252|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 26139871 (25M) [application/zip]

Oct 17 10:16:18 ip-172-31-34-68.ec2.internal systemd[1]: Starting The Apache HTTP Server...
Oct 17 10:16:18 ip-172-31-34-68.ec2.internal systemd[1]: Started The Apache HTTP Server.
[ec2-user@ip-172-31-34-68 ~]$ 17-10-2024 16:23 28°C Partly sunny
```

- Now go to ec2 instance and copy the public ipv4 it on Google. Browse it and check the official page of httpd is displayed.



- Now go to browse and select a download WordPress and click on proper link and select and copy the address link of WordPress. Download the file and past that along with wget command in gitbash as “wget <address. Link>”
- It gives the zip file to unzip that file by using a command as “unzip <zip file>”

```

2024-10-17 10:18:39 (39.8 MB/s) - 'latest.zip' saved [26139871/26139871]
[ec2-user@ip-172-31-34-68 ~]$ ll
total 25528
-rw-r--r-- 1 ec2-user ec2-user 26139871 Sep 10 15:24 latest.zip
[ec2-user@ip-172-31-34-68 ~]$ unzip latest.zip
Archive: latest.zip
  creating: wordpress/
  inflating: wordpress/xmlrpc.php
  inflating: wordpress/wp-blog-header.php
  inflating: wordpress/readme.html
  inflating: wordpress/wp-signup.php
  inflating: wordpress/index.php
  inflating: wordpress/wp-cron.php
  inflating: wordpress/wp-config-sample.php
  inflating: wordpress/wp-login.php
  inflating: wordpress/wp-settings.php
  inflating: wordpress/license.txt
  creating: wordpress/wp-content/
  creating: wordpress/wp-content/themes/
  creating: wordpress/wp-content/themes/twentytwentythree/
  inflating: wordpress/wp-content/themes/twentytwentythree/theme.json
  creating: wordpress/wp-content/themes/twentytwentythree/parts/
  inflating: wordpress/wp-content/themes/twentytwentythree/parts/footer.html
  inflating: wordpress/wp-content/themes/twentytwentythree/parts/comments.html
  inflating: wordpress/wp-content/themes/twentytwentythree/parts/header.html
  inflating: wordpress/wp-content/themes/twentytwentythree/parts/post-meta.html
  creating: wordpress/wp-content/themes/twentytwentythree/patterns/
  inflating: wordpress/wp-content/themes/twentytwentythree/patterns/hidden-404.php
  inflating: wordpress/wp-content/themes/twentytwentythree/patterns/post-meta.php
  inflating: wordpress/wp-content/themes/twentytwentythree/patterns/hidden-no-results.php
  inflating: wordpress/wp-content/themes/twentytwentythree/patterns/call-to-action.php
  inflating: wordpress/wp-content/themes/twentytwentythree/patterns/footer-default.php
  inflating: wordpress/wp-content/themes/twentytwentythree/patterns/hidden-comments.php
  creating: wordpress/wp-content/themes/twentytwentythree/styles/
  inflating: wordpress/wp-content/themes/twentytwentythree/styles/sherbet.json
  inflating: wordpress/wp-content/themes/twentytwentythree/styles/grapes.json
  inflating: wordpress/wp-content/themes/twentytwentythree/styles/canary.json
  inflating: wordpress/wp-content/themes/twentytwentythree/styles/electric.json
  inflating: wordpress/wp-content/themes/twentytwentythree/styles/pitch.json

```

- To run WordPress web application, you must install of WordPress web application as php language with the following commands <sudo Amazon-linux-extras install -y lamp-mariadb10.2-php7.2.php7.2> otherwise update ec2 instance with the following commands <sudo yum update -y>
- Now go inside the unzip directory by using command as “cd <unzip directory> and change the WordPress configuration file by giving command as “sudo mv wp-config-sample.php wp-config.php”

```

[ec2-user@ip-172-31-34-68 ~]$ ll
total 25532
-rw-r--r-- 1 ec2-user ec2-user 26139871 Sep 10 15:24 latest.zip
drwxr-xr-x 5 ec2-user ec2-user 4096 Sep 10 15:23 wordpress
[ec2-user@ip-172-31-34-68 ~]$ cd wordpress
[ec2-user@ip-172-31-34-68 wordpress]$ ll
total 232
-rw-r--r-- 1 ec2-user ec2-user 405 Feb 6 2020 index.php
-rw-r--r-- 1 ec2-user ec2-user 19915 Jan 1 2024 license.txt
-rw-r--r-- 1 ec2-user ec2-user 7409 Jun 18 11:59 readme.html
-rw-r--r-- 1 ec2-user ec2-user 7387 Feb 13 2024 wp-activate.php
drwxr-xr-x 9 ec2-user ec2-user 4096 Sep 10 15:23 wp-admin
-rw-r--r-- 1 ec2-user ec2-user 351 Feb 6 2020 wp-blog-header.php
-rw-r--r-- 1 ec2-user ec2-user 2323 Jun 14 2023 wp-comments-post.php
-rw-r--r-- 1 ec2-user ec2-user 3033 Mar 11 2024 wp-config-sample.php
drwxr-xr-x 4 ec2-user ec2-user 52 Sep 10 15:23 wp-content
-rw-r--r-- 1 ec2-user ec2-user 5638 May 30 2023 wp-cron.php
drwxr-xr-x 30 ec2-user ec2-user 12288 Sep 10 15:23 wp-includes
-rw-r--r-- 1 ec2-user ec2-user 2502 Nov 26 2022 wp-links-opml.php
-rw-r--r-- 1 ec2-user ec2-user 3937 Mar 11 2024 wp-load.php
-rw-r--r-- 1 ec2-user ec2-user 51238 May 28 11:13 wp-login.php
-rw-r--r-- 1 ec2-user ec2-user 8525 Sep 16 2023 wp-mail.php
-rw-r--r-- 1 ec2-user ec2-user 28774 Jul 9 15:43 wp-settings.php
-rw-r--r-- 1 ec2-user ec2-user 34385 Jun 19 2023 wp-signup.php
-rw-r--r-- 1 ec2-user ec2-user 4885 Jun 22 2023 wp-trackback.php
-rw-r--r-- 1 ec2-user ec2-user 3246 Mar 2 2024 xmlrpc.php
[ec2-user@ip-172-31-34-68 wordpress]$ sudo mv wp-config-sample.php wp-config.php
[ec2-user@ip-172-31-34-68 wordpress]$ ll
total 232
-rw-r--r-- 1 ec2-user ec2-user 405 Feb 6 2020 index.php
-rw-r--r-- 1 ec2-user ec2-user 19915 Jan 1 2024 license.txt
-rw-r--r-- 1 ec2-user ec2-user 7409 Jun 18 11:59 readme.html
-rw-r--r-- 1 ec2-user ec2-user 7387 Feb 13 2024 wp-activate.php
drwxr-xr-x 9 ec2-user ec2-user 4096 Sep 10 15:23 wp-admin
-rw-r--r-- 1 ec2-user ec2-user 351 Feb 6 2020 wp-blog-header.php
-rw-r--r-- 1 ec2-user ec2-user 2323 Jun 14 2023 wp-comments-post.php
-rw-r--r-- 1 ec2-user ec2-user 3033 Mar 11 2024 wp-config.php
drwxr-xr-x 4 ec2-user ec2-user 52 Sep 10 15:23 wp-content
-rw-r--r-- 1 ec2-user ec2-user 5638 May 30 2023 wp-cron.php
drwxr-xr-x 30 ec2-user ec2-user 12288 Sep 10 15:23 wp-includes
-rw-r--r-- 1 ec2-user ec2-user 2502 Nov 26 2022 wp-links-opml.php

[28°C Partly sunny] [Search] [File Explorer] [Task View] [Start] [File] [Edit] [View] [Insert] [Format] [Tools] [Help] [ENG IN] [Wi-Fi] [Battery] [16:25] [17-10-2024]

```

- Now do some configurations in the WordPress configuration file by giving database name, username, password and hostname for that execute as sudo vi wp-config.php". along with these change the WordPress secret key. To get the secret key browse the Google generate WordPress secret key.

```

[ec2-user@ip-172-31-34-68 ~]$ cat wp-config.php
/*
 ** Database settings - You can get this info from your web host **
 ** The name of the database for WordPress */
define( 'DB_NAME', 'wordpressdb' );

/** Database username */
define( 'DB_USER', 'admin' );

/** Database password */
define( 'DB_PASSWORD', '9966774360' );

/** Database hostname */
define( 'DB_HOST', 'wordpressdb.cfayeqi0o8ulz.us-east-1.rds.amazonaws.com' );

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

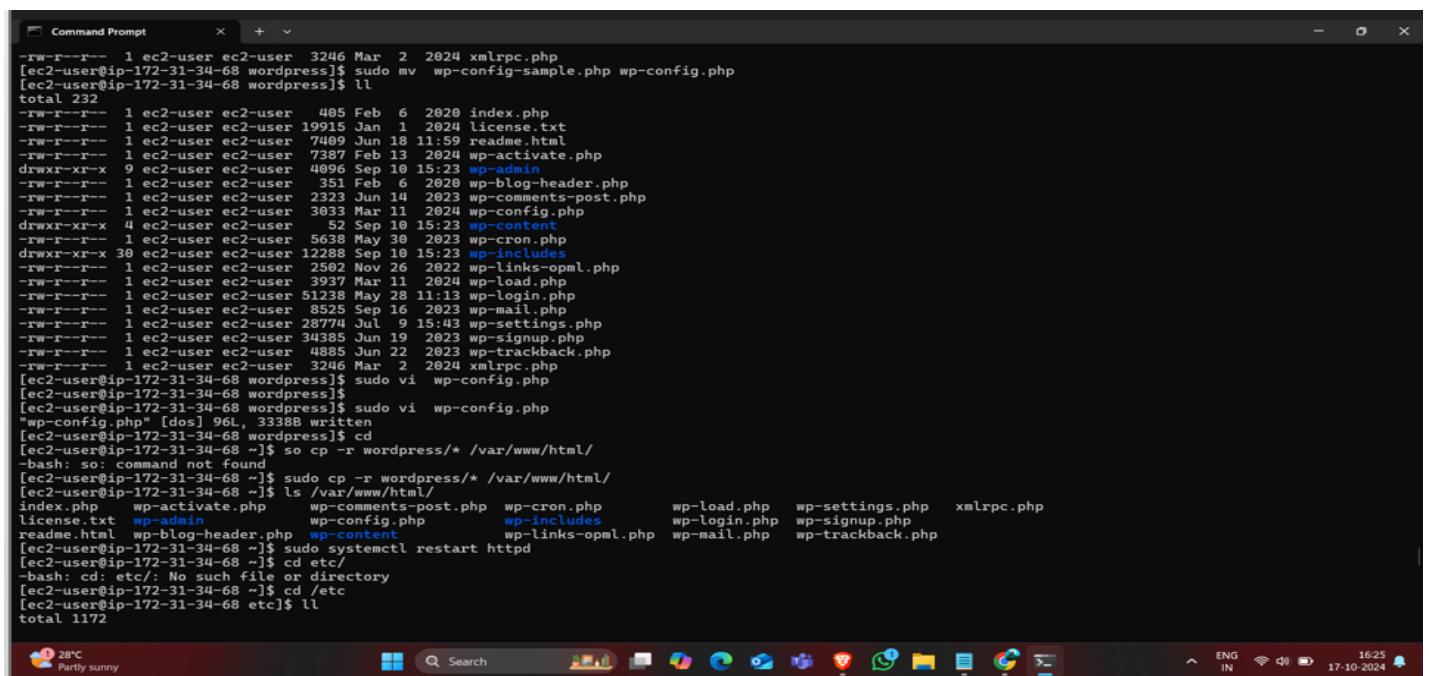
/** The database collate type. Don't change this if in doubt. */
define( 'DB_COLLATE', '' );

/* Authentication unique keys and salts.
 * Change these to different unique phrases! You can generate these using
 * the {@link https://api.wordpress.org/secret-key/1.1/salt/ WordPress.org secret-key service}.
 *
 * You can change these at any point in time to invalidate all existing cookies.
 * This will force all users to have to log in again.
 *
 * @since 2.6.0
 */
define('AUTH_KEY', 't9fifQmgYEBMzUjLlUjEV^Lh[WD+Q,1xo/W6x0+cF5PG+0%37t:PlH&c:CvnL-');
define('SECURE_AUTH_KEY', 'Rgd6_6saRlcBcI'-0M_GL_ntBOjeNg==QC11Z,SDT)Y7@id0:S7vib%J,luP:');
define('LOGGED_IN_KEY', 'jgG8N_yDwU[RY]YnD,>F76wIZw:3Z6>QQL|HM|F=MU2E(bF,H2<>*+@,|IM|');
define('NONCE_KEY', ':+rB|STxAsrB0q;]l)U82=,f+iou9pi>O3_1|DPV0:0gx167&r5||Msbm');
define('AUTH_SALT', 'fuhatRk,B>-+ x#bzun0m>x/g|X-Q+V1wT-AxVXJ$7-fh@#++;:23+>n7Q');
define('SECURE_AUTH_SALT', 't#y&oujd9?ug96;7VQ0=<V1:>6>{3~Z,ZM#5qP_KMm3VU-`!`KALB[] :d,-');
define('LOGGED_IN_SALT', '|E`Yrxde:Dp8JvIzg,ZHs&xit;(<?x0|HJtE3|+-<HD#z2d@2F[B<7!tVc');
define('NONCE_SALT', 'p#p9.M=3/$+jQm#11H1){jy >fm+S&HaA57'E=6HBX18R7s1Vg-0@_+6!9ds');

/*#@#@*/
--- INSERT ---
[28°C Partly sunny] [Search] [File Explorer] [Task View] [Start] [File] [Edit] [View] [Insert] [Format] [Tools] [Help] [ENG IN] [Wi-Fi] [Battery] [16:34] [17-10-2024]

```

- Now copy the WordPress directory to the document root directory to host the web application of
- WordPress by giving as command as
- Sudo cp -I <unzip file>/\* /var/www/html/
- To restart the httpd server by using command sudo systemctl restart the httpd

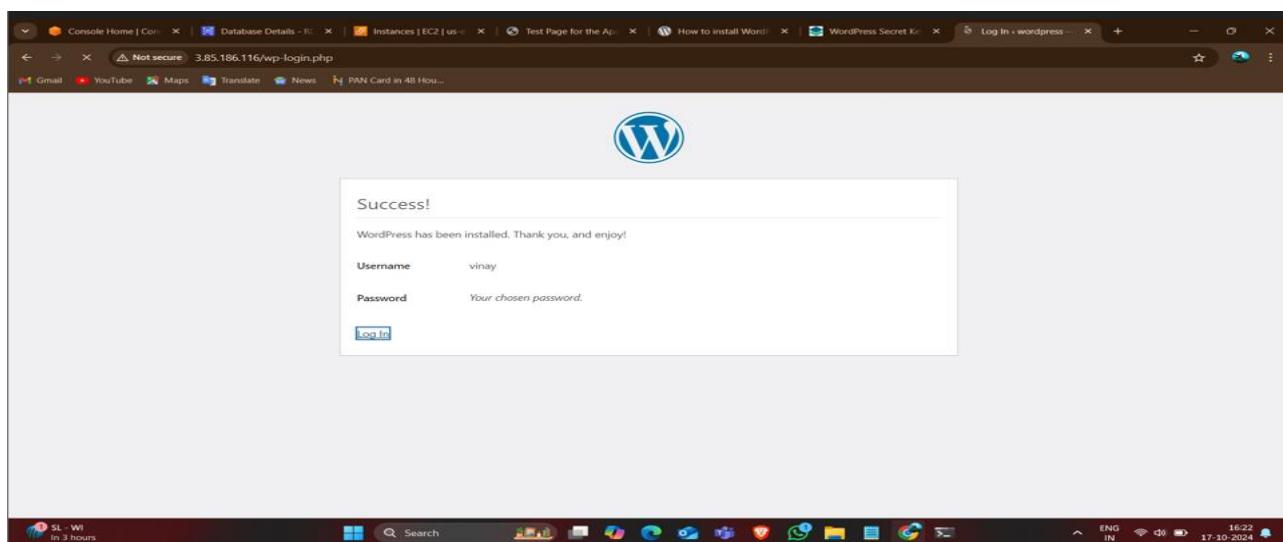


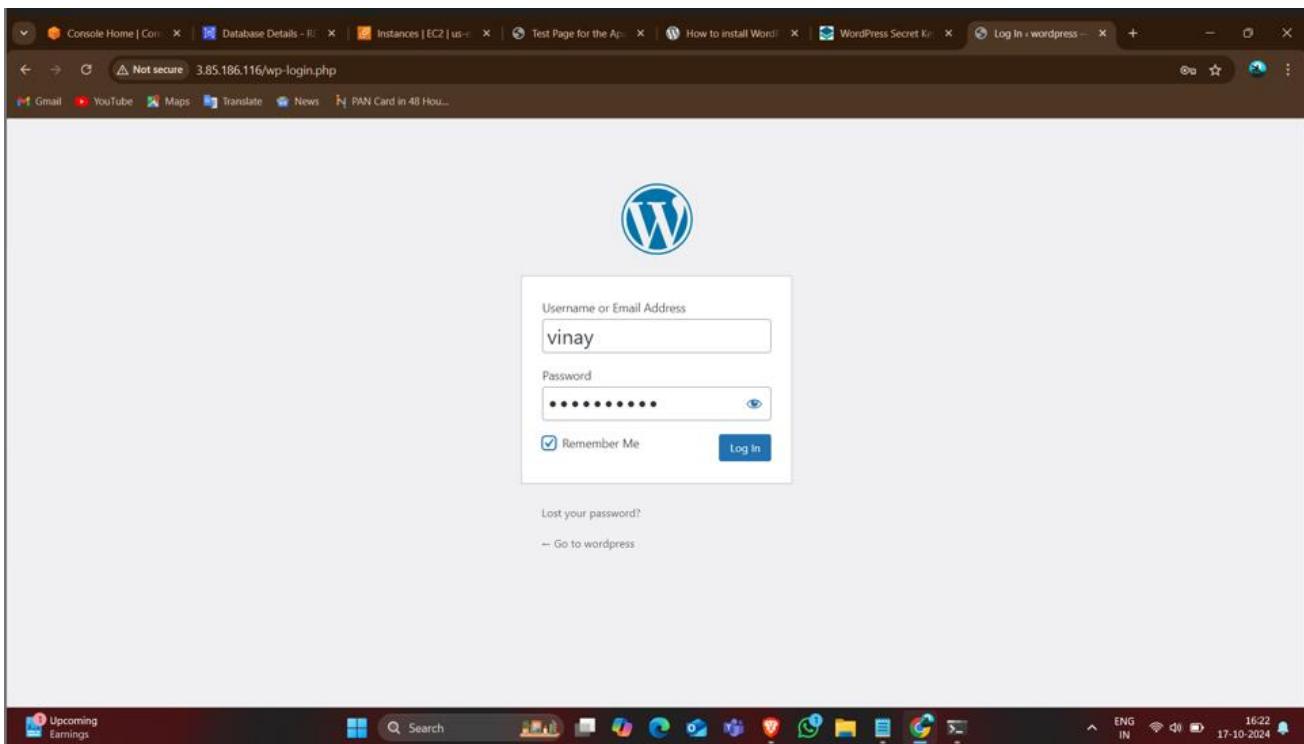
```

Command Prompt x + ~
[ec2-user@ip-172-31-34-68 wordpress]$ sudo mv wp-config-sample.php wp-config.php
[ec2-user@ip-172-31-34-68 wordpress]$ ll
total 232
-rw-r--r-- 1 ec2-user ec2-user 3246 Mar  2 2024 xmlrpc.php
[ec2-user@ip-172-31-34-68 wordpress]$ ll
drwxr-xr-x 1 ec2-user ec2-user 4096 Sep 10 15:23 wp-admin
drwxr-xr-x 1 ec2-user ec2-user 19915 Jan  1 2024 license.txt
drwxr-xr-x 1 ec2-user ec2-user 7409 Jun 18 11:59 readme.html
drwxr-xr-x 1 ec2-user ec2-user 7387 Feb 13 2024 wp-activate.php
drwxr-xr-x 9 ec2-user ec2-user 4096 Sep 10 15:23 wp-admin
drwxr-xr-x 1 ec2-user ec2-user 351 Feb  6 2020 wp-blog-header.php
drwxr-xr-x 1 ec2-user ec2-user 2323 Jun 14 2023 wp-comments-post.php
drwxr-xr-x 1 ec2-user ec2-user 3033 Mar 11 2024 wp-config.php
drwxr-xr-x 4 ec2-user ec2-user 52 Sep 10 15:23 wp-content
drwxr-xr-x 1 ec2-user ec2-user 5638 May 30 2023 wp-cron.php
drwxr-xr-x 30 ec2-user ec2-user 12288 Sep 10 15:23 wp-includes
drwxr-xr-x 1 ec2-user ec2-user 2502 Nov 26 2022 wp-links-opml.php
drwxr-xr-x 1 ec2-user ec2-user 3937 Mar 11 2024 wp-load.php
drwxr-xr-x 1 ec2-user ec2-user 51238 May 28 11:13 wp-login.php
drwxr-xr-x 1 ec2-user ec2-user 8525 Sep 16 2023 wp-mail.php
drwxr-xr-x 1 ec2-user ec2-user 28774 Jul  9 15:43 wp-settings.php
drwxr-xr-x 1 ec2-user ec2-user 34385 Jun 19 2023 wp-signup.php
drwxr-xr-x 1 ec2-user ec2-user 4885 Jun 22 2023 wp-trackback.php
drwxr-xr-x 1 ec2-user ec2-user 3246 Mar  2 2024 xmlrpc.php
[ec2-user@ip-172-31-34-68 wordpress]$ sudo vi wp-config.php
[ec2-user@ip-172-31-34-68 wordpress]$ 
[ec2-user@ip-172-31-34-68 wordpress]$ sudo vi wp-config.php
"wp-config.php" [dos] 96L, 3338B written
[ec2-user@ip-172-31-34-68 wordpress]$ cd
[ec2-user@ip-172-31-34-68 ~]$ so cp -r wordpress/* /var/www/html/
-bash: so: command not found
[ec2-user@ip-172-31-34-68 ~]$ sudo cp -r wordpress/* /var/www/html/
[ec2-user@ip-172-31-34-68 ~]$ ls /var/www/html/
index.php wp-activate.php wp-comments-post.php wp-cron.php wp-load.php wp-settings.php xmlrpc.php
license.txt wp-admin wp-config.php wp-content wp-links-opml.php wp-login.php wp-signup.php
readme.html wp-blog-header.php wp-content wp-mail.php wp-trackback.php
[ec2-user@ip-172-31-34-68 ~]$ sudo systemctl restart httpd
[ec2-user@ip-172-31-34-68 ~]$ cd etc/
-bash: cd: etc/: No such file or directory
[ec2-user@ip-172-31-34-68 ~]$ cd /etc
[ec2-user@ip-172-31-34-68 etc]$ ll
total 1172

```

- Now go to ec2 instance and copy public ipv4 and paste it on google browse it and check the official page of WordPress it displays. Than register and login.





Welcome to WordPress!

Learn more about the 6.6.2 version.

Dashboard

Home

Updates

Posts

Media

Pages

Comments

Appearance

Plugins

Users

Tools

Settings

Collapse menu

Screen Options

Help

Dismiss

PHP Update Required

Your site is running on an outdated version of PHP (7.2.34), which does not receive security updates and soon will not be

# Deploying the Wordpress web application by using Docker

## Module-1:

- Login to the AWS consol.
- Launch the EC2 instance.
- Use AMI amazon-linux 2,t2.micro,keypair.
- Port 80(HTTP),443(HTTPS),& 22(SSH).
- Connect to the terminal.

The screenshot shows the AWS EC2 Instances page. A single instance named "wpwa" is listed, showing it is "Running" with an instance ID of "i-0fbdb3f0e6a3ee8d97". The instance type is "t2.micro". The public IPv4 address is 54.210.219.109, and the private IP is 172.31.33.236. The public IPv4 DNS is ec2-54-210-219-109.compute-1.amazonaws.com. The instance was created less than a minute ago and is in the "us-east-1b" availability zone. The screenshot also shows the AWS navigation bar and various service links on the left.

The screenshot shows the AWS Security Groups page for the "launch-wizard-7" security group. The security group ID is sg-02f1fe6753b6116c. It has 3 permission entries. The inbound rules table lists three rules: one for HTTPS (TCP port 443) allowing traffic from 0.0.0.0/0, one for SSH (TCP port 22) allowing traffic from 0.0.0.0/0, and one for HTTP (TCP port 80) allowing traffic from 0.0.0.0/0. The screenshot also shows the AWS navigation bar and various service links on the left.

```
ec2-user@ip-172-31-33-236: ~ + v Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user>cd downloads

C:\Users\user\Downloads>ssh -i "vinay.pem" ec2-user@ec2-54-210-219-109.compute-1.amazonaws.com
The authenticity of host 'ec2-54-210-219-109.compute-1.amazonaws.com (54.210.219.109)' can't be established.
ED25519 key fingerprint is SHA256:1R5DJMTb5BSravQmw582IMvB81MDz+PYcwKDANedr7k.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-54-210-219-109.compute-1.amazonaws.com' (ED25519) to the list of known hosts.

#_
\_###_      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.
/m/     https://aws.amazon.com/linux/amazon-linux-2023/

[ec2-user@ip-172-31-33-236 ~]$ |
```

## Module-2:

- Install git “`sudo yum -y install git`”

```
[ec2-user@ip-172-31-33-236 ~]$ sudo yum update -y
Warning: Permanently added 'ec2-54-210-219-109.compute-1.amazonaws.com' (ED25519) to the list of known hosts.

          _#
  _\_\_ #####      Amazon Linux 2
  _\_\_\_ #####\_
  _\_\_\_\_ \#\#\|    AL2 End of Life is 2025-06-30.
  _\_\_\_\_ \#/|_-->
  _\_\_\_\_ V~`|_--> 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-172-31-33-236 ~]$ sudo yum update -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core                                         | 3.6 kB  00:00:00
No packages marked for update
[ec2-user@ip-172-31-33-236 ~]$ sudo 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) 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 0:2.40.1-1.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

=====
Package           Arch       Version            Repository
=====

```

- Install Docker “sudo yum install docker -y”

```

perl-GI::WWW::0.2.40.1-1.amzn2.0.3          perl-TelnetReadkey::x86_64 0.2.30-20.amzn2.0.2
Complete!
[ec2-user@ip-172-31-33-236 ~]$ sudo yum install -y docker
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Resolving Dependencies
--> Running transaction check
--> Package docker.x86_64 0:25.0.6-1.amzn2.0.2 will be installed
--> Processing Dependency: containerd >= 1.3.2 for package: docker-25.0.6-1.amzn2.0.2.x86_64
--> Processing Dependency: libcgroup >= 0.40.rcl-5.15 for package: docker-25.0.6-1.amzn2.0.2.x86_64
--> Processing Dependency: runc >= 1.0.0 for package: docker-25.0.6-1.amzn2.0.2.x86_64
--> Processing Dependency: pigz for package: docker-25.0.6-1.amzn2.0.2.x86_64
--> Running transaction check
--> Package containerd.x86_64 0:1.7.22-1.amzn2.0.2 will be installed
--> Package libcgroup.x86_64 0:0.41-21.amzn2 will be installed
--> Package pigz.x86_64 0:2.3.4-1.amzn2.0.1 will be installed
--> Package runc.x86_64 0:1.1.14-1.amzn2 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package           Arch         Version          Repository        Size
=====
Installing:
=====

```

- Now give the permissions to add limited user account to docker group “sudo usermod –aG username” or “sudo chmod 666 /var/run/dokcer.sock”

```

[ec2-user@ip-172-31-33-236 ~] Verifying : pigz-2.3.4-1.amzn2.0.1.x86_64 2/5
[ec2-user@ip-172-31-33-236 ~] Verifying : runc-1.1.14-1.amzn2.x86_64 3/5
[ec2-user@ip-172-31-33-236 ~] Verifying : containerd-1.7.22-1.amzn2.0.2.x86_64 4/5
[ec2-user@ip-172-31-33-236 ~] Verifying : libcgroup-0.41-21.amzn2.x86_64 5/5

Installed:
  docker.x86_64 0:25.0.6-1.amzn2.0.2

Dependency Installed:
  containerd.x86_64 0:1.7.22-1.amzn2.0.2      libcgroup.x86_64 0:0.41-21.amzn2      pigz.x86_64 0:2.3.4-1.amzn2.0.1      runc.x86_64 0:1.1.14-1.amzn2

Complete!
[ec2-user@ip-172-31-33-236 ~]$ sudo usermod -aG username(ec2-user)
-bash: syntax error near unexpected token `('
[ec2-user@ip-172-31-33-236 ~]$ sudo usermod -aG username'ec2-user'
usermod: group 'username' does not exist
[ec2-user@ip-172-31-33-236 ~]$ sudo usermod -aG username'(ec2-user)'
usermod: group 'username(ec2-user)' does not exist
[ec2-user@ip-172-31-33-236 ~]$ sudo usermod -aG ec2-user username
usermod: user 'username' does not exist
[ec2-user@ip-172-31-33-236 ~]$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin/nologin
daemon:x:2:2:daemon:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/bin/sync
shutdown:x:6:8:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/sbin/nologin
systemd-network:x:192:192:systemd Network Management:/sbin/nologin
dbus:x:81:81:System message bus:/sbin/nologin
rpc:x:32:32:Rpcbind Daemon:/var/lib/rpcbind:/sbin/nologin
libstoragemgmt:x:999:997:daemon account for libstoragemgmt:/var/run/lsm:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
rngd:x:998:996:Random Number Generator Daemon:/var/lib/rngd:/sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin

[ec2-user@ip-172-31-33-236 ~] 26°C Haze

```

- Now start and enable the docker  
“sudo systemctl start docker”  
“sudo systemctl enable docker”  
“sudo systemctl status docker”

```

[ec2-user@ip-172-31-33-236 ~]$ sudo systemctl start docker
[ec2-user@ip-172-31-33-236 ~]$ sudo systemctl enable docker
Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to /usr/lib/systemd/system/docker.service.
[ec2-user@ip-172-31-33-236 ~]$ 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 Thu 2024-10-17 14:17:06 UTC; 17s ago
       Docs: https://docs.docker.com
      Main PID: 1436 (dockerd)
         CGroup: /system.slice/docker.service
             └─1436 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --default-ulimit nofile=32768:65536

Oct 17 14:17:06 ip-172-31-33-236.ec2.internal systemd[1]: Starting Docker Application Container Engine...
Oct 17 14:17:06 ip-172-31-33-236.ec2.internal dockerd[1436]: time="2024-10-17T14:17:06.215796603Z" level=info msg="Starting up"
Oct 17 14:17:06 ip-172-31-33-236.ec2.internal dockerd[1436]: time="2024-10-17T14:17:06.272662246Z" level=info msg="Loading containers: start."
Oct 17 14:17:06 ip-172-31-33-236.ec2.internal dockerd[1436]: time="2024-10-17T14:17:06.498565901Z" level=info msg="Loading containers: done."
Oct 17 14:17:06 ip-172-31-33-236.ec2.internal dockerd[1436]: time="2024-10-17T14:17:06.504290122Z" level=warning msg="WARNING: bridge-nf-call-ipta...sabled"
Oct 17 14:17:06 ip-172-31-33-236.ec2.internal dockerd[1436]: time="2024-10-17T14:17:06.504647179Z" level=warning msg="WARNING: bridge-nf-call-ip6t...sabled"
Oct 17 14:17:06 ip-172-31-33-236.ec2.internal dockerd[1436]: time="2024-10-17T14:17:06.504906281Z" level=info msg="Docker daemon" commit=b08a51f c...=25.0.6
Oct 17 14:17:06 ip-172-31-33-236.ec2.internal dockerd[1436]: time="2024-10-17T14:17:06.505140741Z" level=info msg="Daemon has completed initialization"
Oct 17 14:17:06 ip-172-31-33-236.ec2.internal dockerd[1436]: time="2024-10-17T14:17:06.5408645260Z" level=info msg="API listen on /run/docker.sock"
Oct 17 14:17:06 ip-172-31-33-236.ec2.internal systemd[1]: Started Docker Application Container Engine.
Hint: Some lines were ellipsized, use -l to show in full.
[ec2-user@ip-172-31-33-236 ~]$ sudo chmod 666 /var/run/docker.sock
[ec2-user@ip-172-31-33-236 ~]$ sudo curl -SL https://github.com/docker/compose/releases/download/v2.29.6/docker-compose-linux-x86_64 -o /usr/local/bin/docker-compose
[ec2-user@ip-172-31-33-236 ~]$ ll
total 0
[ec2-user@ip-172-31-33-236 ~]$ sudo vi docker-compose.yml

26°C Haze 20:41 17-10-2024

```

- Now install the docker compose by searching the commands in google (install docker compose).
- Now give the executable permissions to the binary by using “sudo chmod +x /usr/local/bin/docker-compose”
- Create the system link “sudo ln –s /usr/local/bin/docker-compose /usr/bin/docker-compose”.
- Create the docker-compose.yml “sudo vi docker-compose.yml”

```

[ec2-user@ip-172-31-33-236 ~]$ sudo chmod 666 /var/run/docker.sock
[ec2-user@ip-172-31-33-236 ~]$ sudo curl -SL https://github.com/docker/compose/releases/download/v2.29.6/docker-compose-linux-x86_64 -o /usr/local/bin/docker-compose
[ec2-user@ip-172-31-33-236 ~]$ ll
total 0
[ec2-user@ip-172-31-33-236 ~]$ sudo vi docker-compose.yml

26°C Haze 20:41 17-10-2024

```

- Now write yaml code to run the wordpress web application.

```

ec2-user@ip-172-31-33-236:~ % 
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=vinay
      - MYSQL_USER=gollena
      - MYSQL_PASSWORD=9966774360

  wordpress:
    image: wordpress:latest
    ports:
      - "80:80"
    restart: always
    environment:
      - WORDPRESS_DB_HOST=db
      - WORDPRESS_DB_USER=gollena
      - WORDPRESS_DB_PASSWORD=9966774360
      - WORDPRESS_DB_NAME=vinay
volume:
  db_data:
~

"docker-compose.yml" 27L, 595B
27,10          All
26°C Haze
20:10 17-10-2024

```

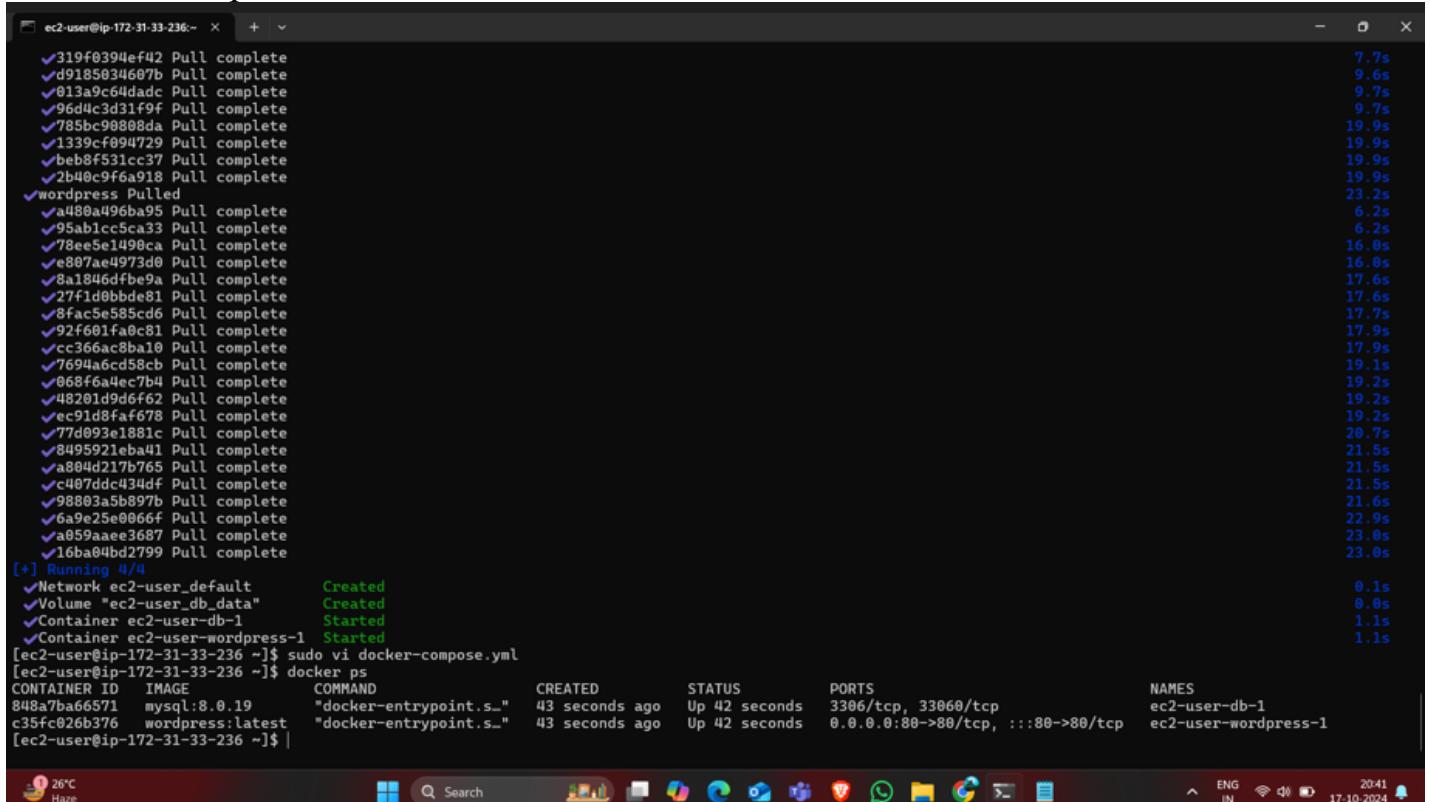
- Now pull file “docker-compose up -d”

```

ec2-user@ip-172-31-33-236:~ % sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
ln: failed to create symbolic link '/usr/bin/docker-compose': File exists
[ec2-user@ip-172-31-33-236 ~]$ sudo vi docker-compose.yml
[ec2-user@ip-172-31-33-236 ~]$ docker-compose up -d
Validating /home/ec2-user/docker-compose.yml: (root) Additional property volume is not allowed
[ec2-user@ip-172-31-33-236 ~]$ sudo vi docker-compose.yml
"docker-compose.yml" 27L, 596B written
[ec2-user@ip-172-31-33-236 ~]$ docker-compose up -d
WARN[0000] /home/ec2-user/docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion
[*] Running 35/35
  vdb Pulled
    ✓ 54fec2fa59d0 Pull complete
    ✓ bcc6c6145912 Pull complete
    ✓ 951c3d959c9d Pull complete
    ✓ 05de4d0e206e Pull complete
    ✓ 319f0394ef42 Pull complete
    ✓ d9185034607b Pull complete
    ✓ 013a9c64dad0 Pull complete
    ✓ 96d4c3d31f9f Pull complete
    ✓ 785bc90808da Pull complete
    ✓ 1339c-f094729 Pull complete
    ✓ beb8f531cc37 Pull complete
    ✓ 2b40c9f6a918 Pull complete
  wordpress Pulled
    ✓ a480a496ba95 Pull complete
    ✓ 95ab1cc5ca33 Pull complete
    ✓ 78ee5e1490ca Pull complete
    ✓ e807ae4973d0 Pull complete
    ✓ 8a1846dfbe9a Pull complete
    ✓ 27f1d0bbde81 Pull complete
    ✓ 8fac5e585cd6 Pull complete
    ✓ 92f601fa0c81 Pull complete
    ✓ cc366ac8ba10 Pull complete
    ✓ 7694a46cd58cb Pull complete
    ✓ 068f6a4ec7b4 Pull complete
    ✓ 48201d9d6f62 Pull complete
    ✓ ec91d8faf678 Pull complete
    ✓ 77d093e1881c Pull complete
    ✓ 8495921eba41 Pull complete
    ✓ a804d217b765 Pull complete
20.3s
7.1s
7.1s
7.5s
7.7s
7.7s
9.6s
9.7s
9.7s
19.9s
19.9s
19.9s
19.9s
23.2s
6.2s
6.2s
16.0s
16.0s
17.6s
17.6s
17.7s
17.9s
17.9s
19.1s
19.2s
19.2s
19.2s
20.7s
21.5s
21.5s
20:41 17-10-2024

```

- Now see the containers of the pulled images of MySQL and Wordpress by using “docker ps”.



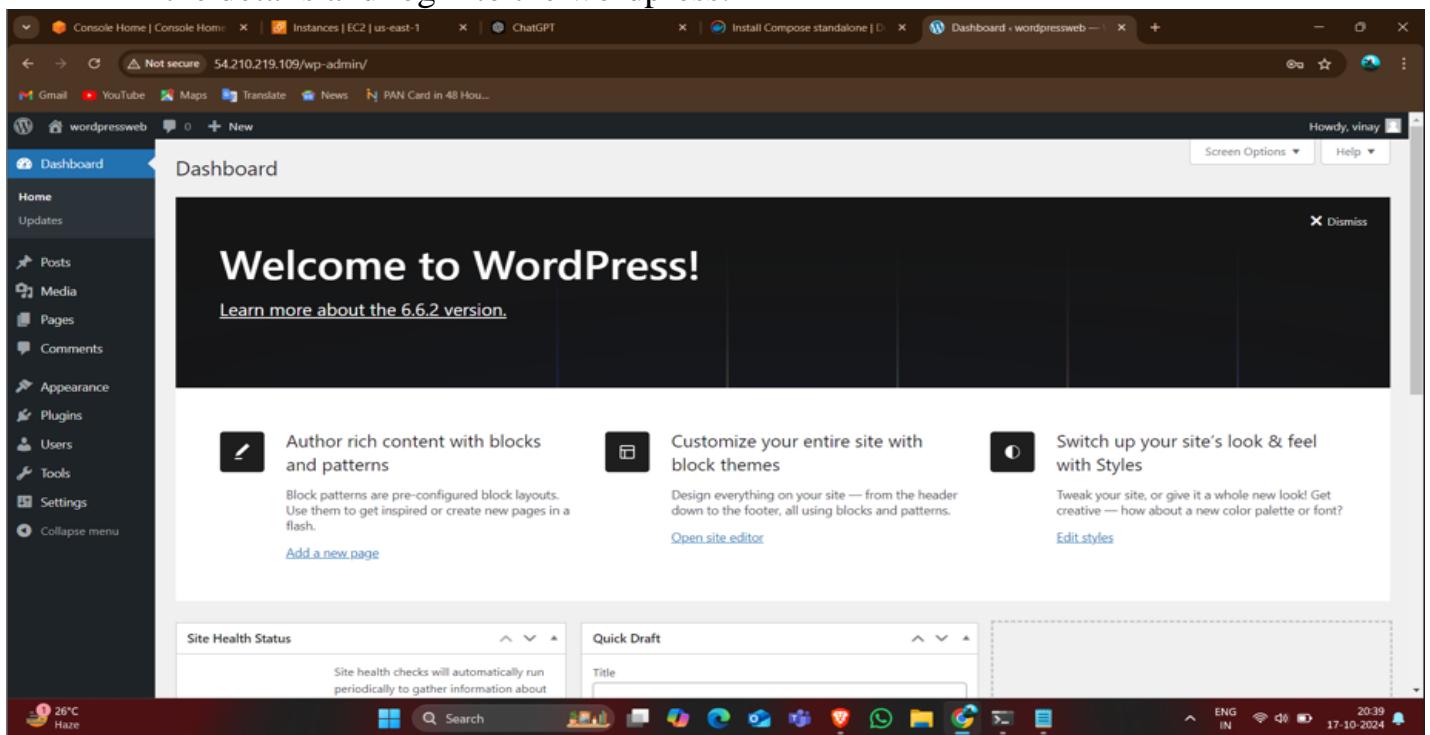
```

ec2-user@ip-172-31-33-236:~ % docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
848a7ba66571 mysql:8.0.19 "docker-entrypoint.s..." 43 seconds ago Up 42 seconds 3306/tcp, 33060/tcp ec2-user-db-1
c35fc026b376 wordpress:latest "docker-entrypoint.s..." 43 seconds ago Up 42 seconds 0.0.0.0:80->80/tcp, :::80->80/tcp ec2-user-wordpress-1
[ec2-user@ip-172-31-33-236 ~]$ sudo vi docker-compose.yml
[ec2-user@ip-172-31-33-236 ~]$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
848a7ba66571 mysql:8.0.19 "docker-entrypoint.s..." 43 seconds ago Up 42 seconds 3306/tcp, 33060/tcp ec2-user-db-1
c35fc026b376 wordpress:latest "docker-entrypoint.s..." 43 seconds ago Up 42 seconds 0.0.0.0:80->80/tcp, :::80->80/tcp ec2-user-wordpress-1
[ec2-user@ip-172-31-33-236 ~]$ 

```

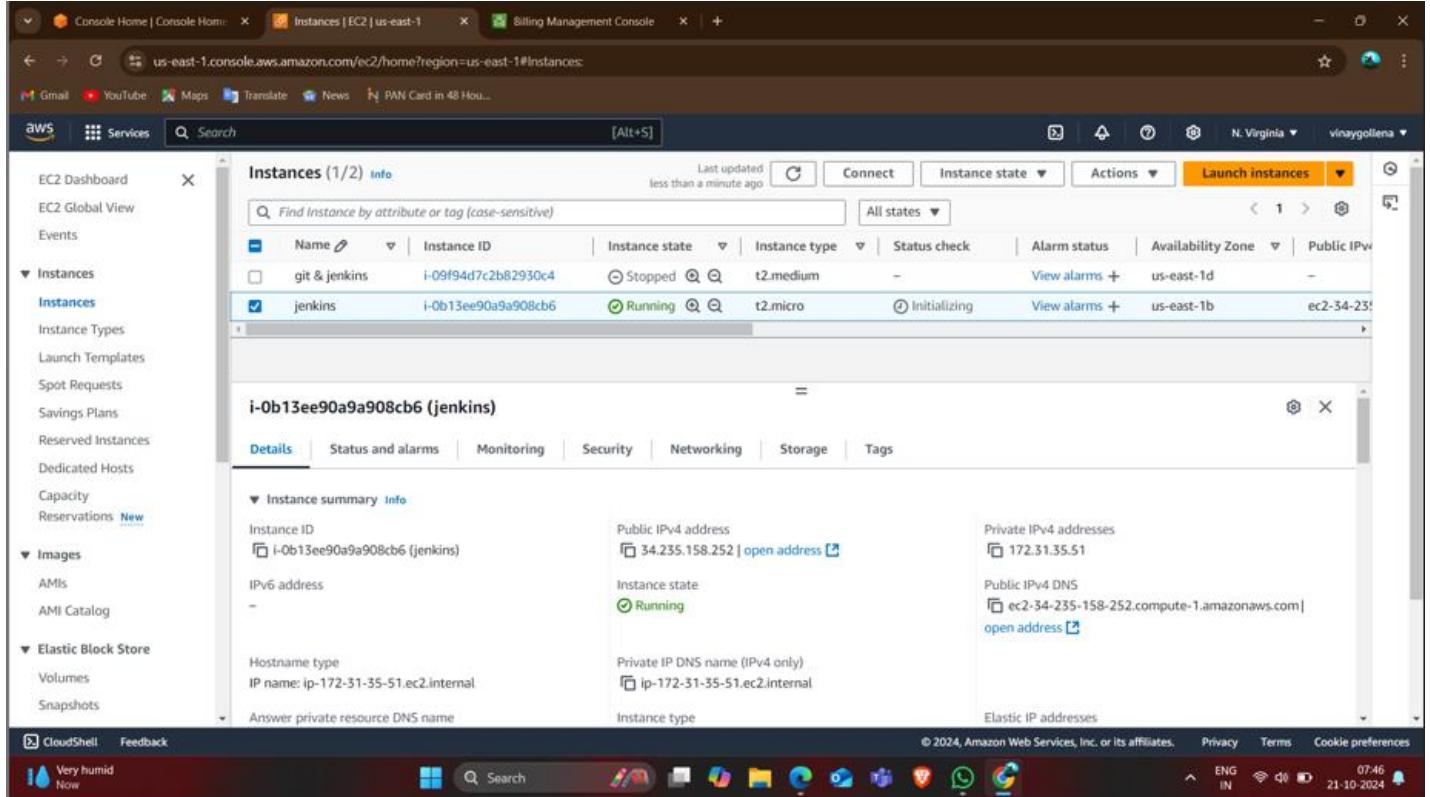
The terminal shows the output of `docker ps` which lists two containers: `ec2-user-db-1` and `ec2-user-wordpress-1`. Both containers are up and running. The command `sudo vi docker-compose.yml` is shown above the container list.

- Now copy the static public ip of your ec2 instance and paste it in the browser with ip:80 and search.
- Fill the details and login to the wordpress.



# Deploy WordPress web application by using git and Jenkins

- Create an EC2 instance with t2.micro and connect it to the terminal



- Update “sudo yum update -y”
- And install git “sudo yum install git -y”

```
Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user>cd Downloads

C:\Users\user\Downloads>ssh -i "vinay.pem" ec2-user@ec2-100-26-186-125.compute-1.amazonaws.com
The authenticity of host 'ec2-100-26-186-125.compute-1.amazonaws.com (100.26.186.125)' can't be established.
ED25519 key fingerprint is SHA256:d0db09mac1FIMsuxFIW06idSj6tUgY75nWrURjsQlk.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-100-26-186-125.compute-1.amazonaws.com' (ED25519) to the list of known hosts.

      _###_
     /###\
    /##\
   /## \
  /## \
 /## \
/## \
[ec2-user@ip-172-31-94-181 ~]$ sudo yum update -y
Last metadata expiration check: 0:00:40 ago on Sun Oct 20 10:53:44 2024.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-94-181 ~]$ sudo yum install git -y
Last metadata expiration check: 0:00:55 ago on Sun Oct 20 10:53:44 2024.
Dependencies resolved.
=====
 Package          Architecture      Version       Repository      Size
=====
Installing:
 git              x86_64          2.40.1-1.amzn2023.0.3      amazonlinux      54 k
Installing dependencies:
 git-core          x86_64          2.40.1-1.amzn2023.0.3      amazonlinux      4.3 M
 git-core-doc      noarch          2.40.1-1.amzn2023.0.3      amazonlinux      2.6 M
 perl-Error        noarch          1:0.17029-5.amzn2023.0.2      amazonlinux      41 k
 perl-File-Find    noarch          1.37-497.amzn2023.0.6      amazonlinux      26 k
 perl-Git          noarch          2.40.1-1.amzn2023.0.3      amazonlinux      42 k
 perl-TermReadKey x86_64          2.38-9.amzn2023.0.2      amazonlinux      36 k
```

- Install Jenkins by using google (install Jenkins ) select the amazon linux or Ubuntu as your wish then copy from it and paste in the terminal.
- Than start and enable the Jenkins and check the status.

```

ec2-user@ip-172-31-94-181:~ % 
Preparing : 1/1
Running scriptlet: jenkins-2.462.3-1.1.noarch 1/1
Installing : jenkins-2.462.3-1.1.noarch 1/1
Running scriptlet: jenkins-2.462.3-1.1.noarch 1/1
Verifying   : jenkins-2.462.3-1.1.noarch 1/1

Installed:
jenkins-2.462.3-1.1.noarch

Complete!
[ec2-user@ip-172-31-94-181 ~]$ sudo systemctl daemon-reload
[ec2-user@ip-172-31-94-181 ~]$ sudo systemctl start jenkins
[ec2-user@ip-172-31-94-181 ~]$ sudo systemctl enable jenkins
Created symlink /etc/systemd/system/multi-user.target.wants/jenkins.service → /usr/lib/systemd/system/jenkins.service.
[ec2-user@ip-172-31-94-181 ~]$ sudo systemctl status jenkins
sudo: sysyemctl: command not found
[ec2-user@ip-172-31-94-181 ~]$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
  Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: disabled)
  Active: active (running) since Sun 2024-10-20 11:00:07 UTC; 45s ago
    Main PID: 28309 (java)
      Tasks: 54 (limit: 4658)
     Memory: 534.8M
        CPU: 16.19s
       CGroup: /system.slice/jenkins.service
           └─28309 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Oct 20 11:00:03 ip-172-31-94-181.ec2.internal jenkins[28309]: 77c8970709104790b5d27db7b639f0c4
Oct 20 11:00:03 ip-172-31-94-181.ec2.internal jenkins[28309]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Oct 20 11:00:03 ip-172-31-94-181.ec2.internal jenkins[28309]: ****
Oct 20 11:00:03 ip-172-31-94-181.ec2.internal jenkins[28309]: ****
Oct 20 11:00:03 ip-172-31-94-181.ec2.internal jenkins[28309]: ****
Oct 20 11:00:07 ip-172-31-94-181.ec2.internal jenkins[28309]: 2024-10-20 11:00:07.033+0000 [id=34] INFO jenkins.InitReactorRunner$1#onAttained
Oct 20 11:00:07 ip-172-31-94-181.ec2.internal jenkins[28309]: 2024-10-20 11:00:07.053+0000 [id=24] INFO hudson.lifecycle.Lifecycle#onReady: J
Oct 20 11:00:07 ip-172-31-94-181.ec2.internal jenkins[28309]: systemd[1]: Started jenkins.service - Jenkins Continuous Integration Server.
Oct 20 11:00:07 ip-172-31-94-181.ec2.internal jenkins[28309]: 2024-10-20 11:00:07.189+0000 [id=58] INFO h.m.DownloadService$Downloadable#load
Oct 20 11:00:07 ip-172-31-94-181.ec2.internal jenkins[28309]: 2024-10-20 11:00:07.190+0000 [id=58] INFO hudson.util.Retrier#start: Performed 2

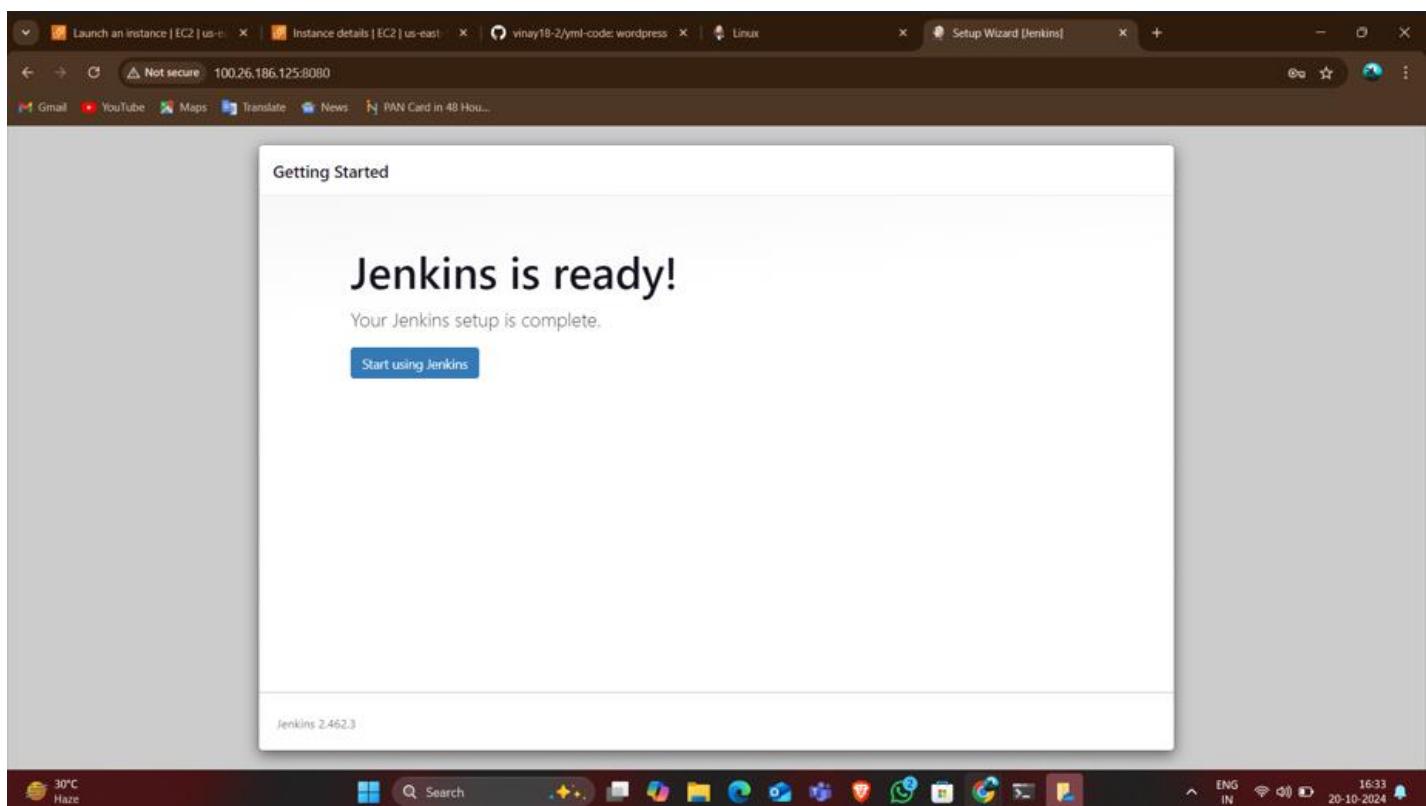
[ec2-user@ip-172-31-94-181 ~]$ sudo visudo
visudo: /etc/sudoers.tmp unchanged
[ec2-user@ip-172-31-94-181 ~]$ sudo visudo

30°C Haze
Search 17:40 20-10-2024
ENG IN ⚡ ID

```

- Than install the docker and docker-compose. Than start the docker and give the limited docker usage permissions.
- Give the command “docker-compose up –d” to pull the image.

- Copy the ip address with port number 8080 and browse it and login to the Jenkins page by using secret key.



- Then create new item that is clone job with freestyle project and select the source is Git and paste the git repo link and choose the branch.
- Save and apply then build now.
- After that copy the ip address of instance and browse it in google and observe the output.

Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

**Start building your software project**

- Create a job
- + Set up a distributed build
- Set up an agent
- Configure a cloud
- Learn more about distributed builds

New Item

Enter an item name

clone-job

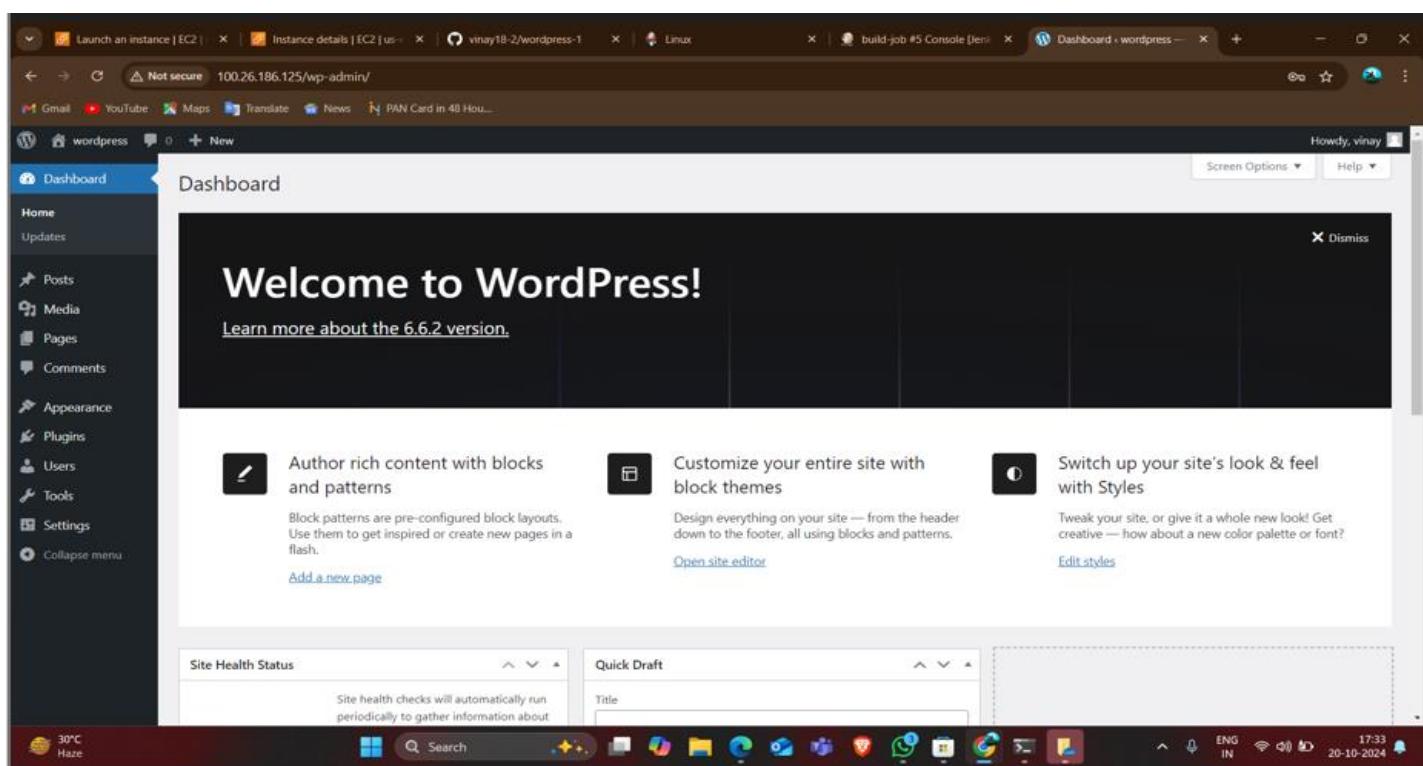
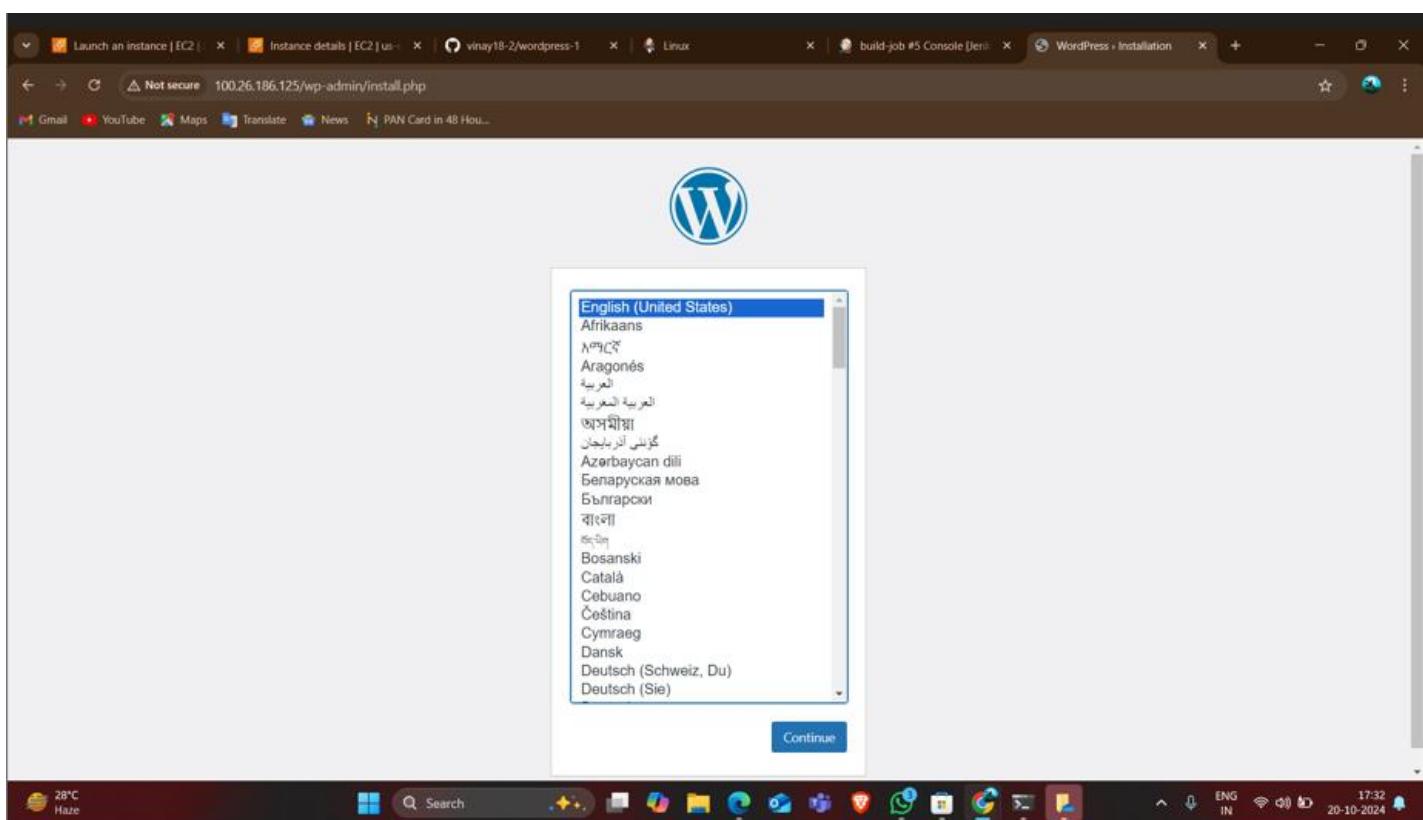
Select an item type

- Freestyle project**  
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.
- Pipeline**  
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
- Multi-configuration project**  
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
- Folder**

OK

The screenshot shows the Jenkins 'General' configuration page for a job named 'clone-job'. The 'Enabled' switch is turned on. The 'Description' field contains the text 'wordpress deployment using git&jenkins'. Under 'Build Triggers', several options are available: 'Discard old builds', 'GitHub project', 'This project is parameterized', 'Throttle builds', and 'Execute concurrent builds if necessary'. At the bottom are 'Save' and 'Apply' buttons.

The screenshot shows the Jenkins 'Source Code Management' configuration page for the 'clone-job' job. The 'Repository URL' is set to 'https://github.com/vinay18-2/yml-code.git'. The 'Branches to build' section has a 'Branch Specifier (blank for 'any')' set to '/main'. At the bottom are 'Save' and 'Apply' buttons.



# Deploy WordPress web application by using userdata of EC2 instance

- Create an EC2 instance with t3.micro,amazon-linux 2.
- Select the security groups HTTP,HTTPS and SSH with port numbers 80,443&22.
- While creating the instances we should write the bash script to install MYSQL and HTTPD and wordpress in the additional settings and save it.

The screenshot shows the AWS EC2 Instances page. A single instance named "wordpress" is listed, with the instance ID "i-00590665c0bedbc86". The instance is currently "Running". The "Details" tab is selected, providing information such as the Public IPv4 address (18.215.244.176), Private IP4 address (172.31.35.115), and the Public IPv4 DNS name (ec2-18-215-244-176.compute-1.amazonaws.com). The instance type is t2.micro, and it is located in the us-east-1b availability zone. The "Tags" tab is also visible.

The screenshot shows the "Edit user data" page for the "wordpress" instance. The user data is a base64-encoded bash script. The script performs several tasks:

- Updates the system with "yum update -y".
- Installs Apache with "yum install -y httpd".
- Starts Apache and enables it to start on boot with "systemctl start httpd" and "systemctl enable httpd".
- Installs MySQL (MariaDB) with "yum install -y mariadb-server".
- Starts MySQL with "systemctl start mariadb" and enables it with "systemctl enable mariadb".
- Installs PHP and related modules with "amazon-linux-extras install php7.4 -y" and "yum install -y php php-mysqlnd".
- Downloads and installs WordPress with "cd /var/www/html" and "wget https://wordpress.org/latest.tar.gz".

A checkbox at the bottom indicates that the input is already base64-encoded.

The screenshot shows the AWS CloudShell interface. A terminal window is open with the following command history:

```
# Install PHP and related modules
amazon-linux-extras install php7.4 -y
yum install -y php php-mysqlnd

# Download and install WordPress
cd /var/www/html
wget https://wordpress.org/latest.tar.gz
tar -xzf latest.tar.gz
cp -r wordpress/* /var/www/html/

# Set permissions
chown -R apache:apache /var/www/html/
chmod -R 755 /var/www/html/

# Configure MySQL
mysql -e "CREATE DATABASE wordpress;""
mysql -e "CREATE USER 'wp_user'@'localhost' IDENTIFIED BY 'password';"
mysql -e "GRANT ALL PRIVILEGES ON wordpress.* TO 'wp_user'@'localhost';"
mysql -e "FLUSH PRIVILEGES;"

# Create wp-config.php from sample and update with DB details
cp /var/www/html/wp-config-sample.php /var/www/html/wp-config.php
sed -i "s/database_name_here/wordpress/" /var/www/html/wp-config.php
sed -i "s/username_here/wp_user/" /var/www/html/wp-config.php
sed -i "s/password_here/password/" /var/www/html/wp-config.php

# Restart Apache to apply changes
systemctl restart httpd
```

A checkbox at the bottom left of the terminal window is unchecked, indicating that the input is not base64-encoded.

On the right side of the screen, a Snipping Tool window is open, showing a screenshot of the terminal window. A message in the Snipping Tool window says: "Screenshot copied to clipboard and saved. Select here to mark up and share."

The AWS CloudShell interface includes a CloudShell tab, a Feedback tab, a weather icon (26°C, Mostly cloudy), a search bar, and a system tray with various icons and status information (ENG IN, 19-10-2024).

This screenshot shows the continuation of the terminal session from the previous one. The command history has been extended to include the creation of the wp-config.php file and the restart of the Apache service.

```
cd /var/www/html
wget https://wordpress.org/latest.tar.gz
tar -xzf latest.tar.gz
cp -r wordpress/* /var/www/html/

# Set permissions
chown -R apache:apache /var/www/html/
chmod -R 755 /var/www/html/

# Configure MySQL
mysql -e "CREATE DATABASE wordpress;""
mysql -e "CREATE USER 'wp_user'@'localhost' IDENTIFIED BY 'password';"
mysql -e "GRANT ALL PRIVILEGES ON wordpress.* TO 'wp_user'@'localhost';"
mysql -e "FLUSH PRIVILEGES;"

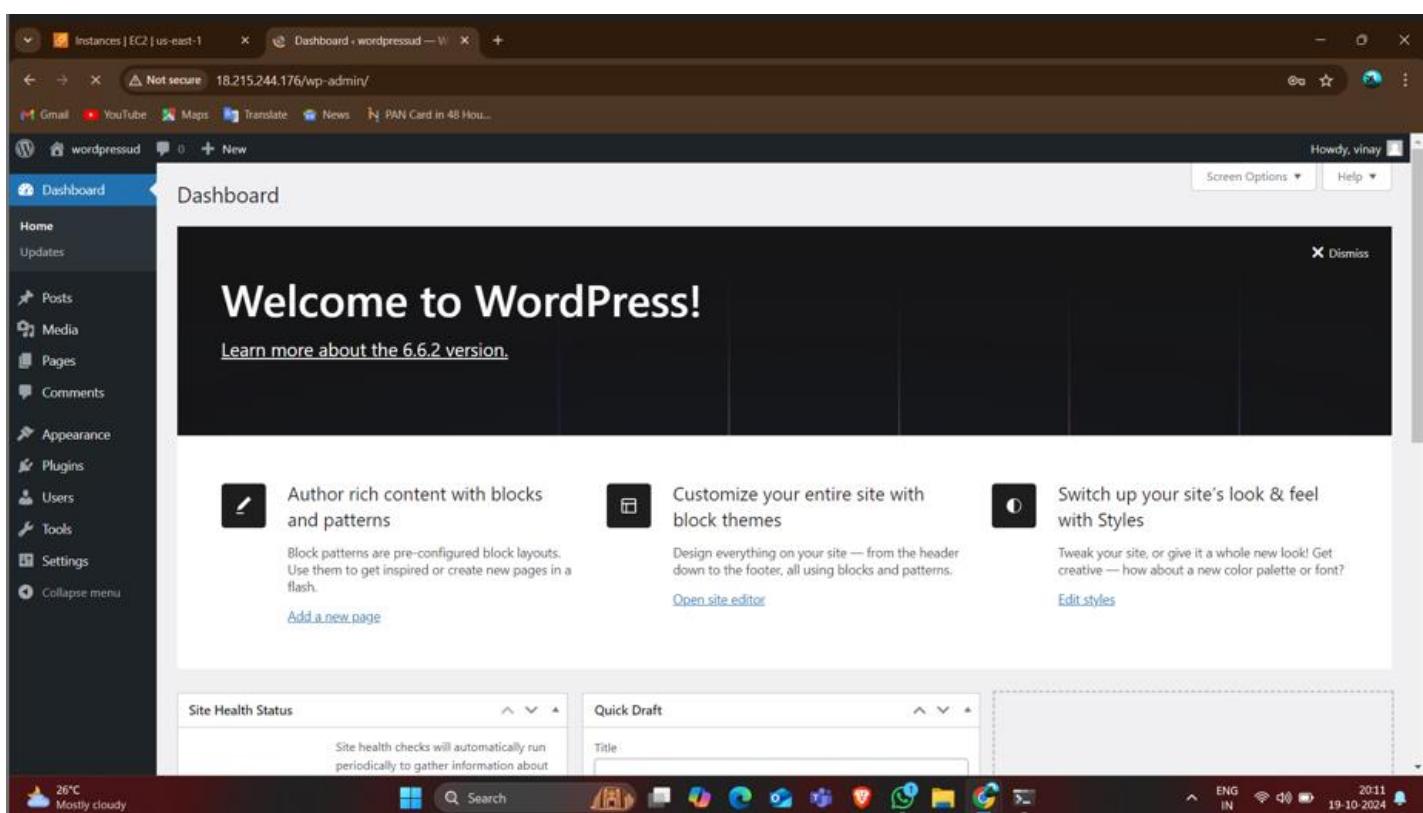
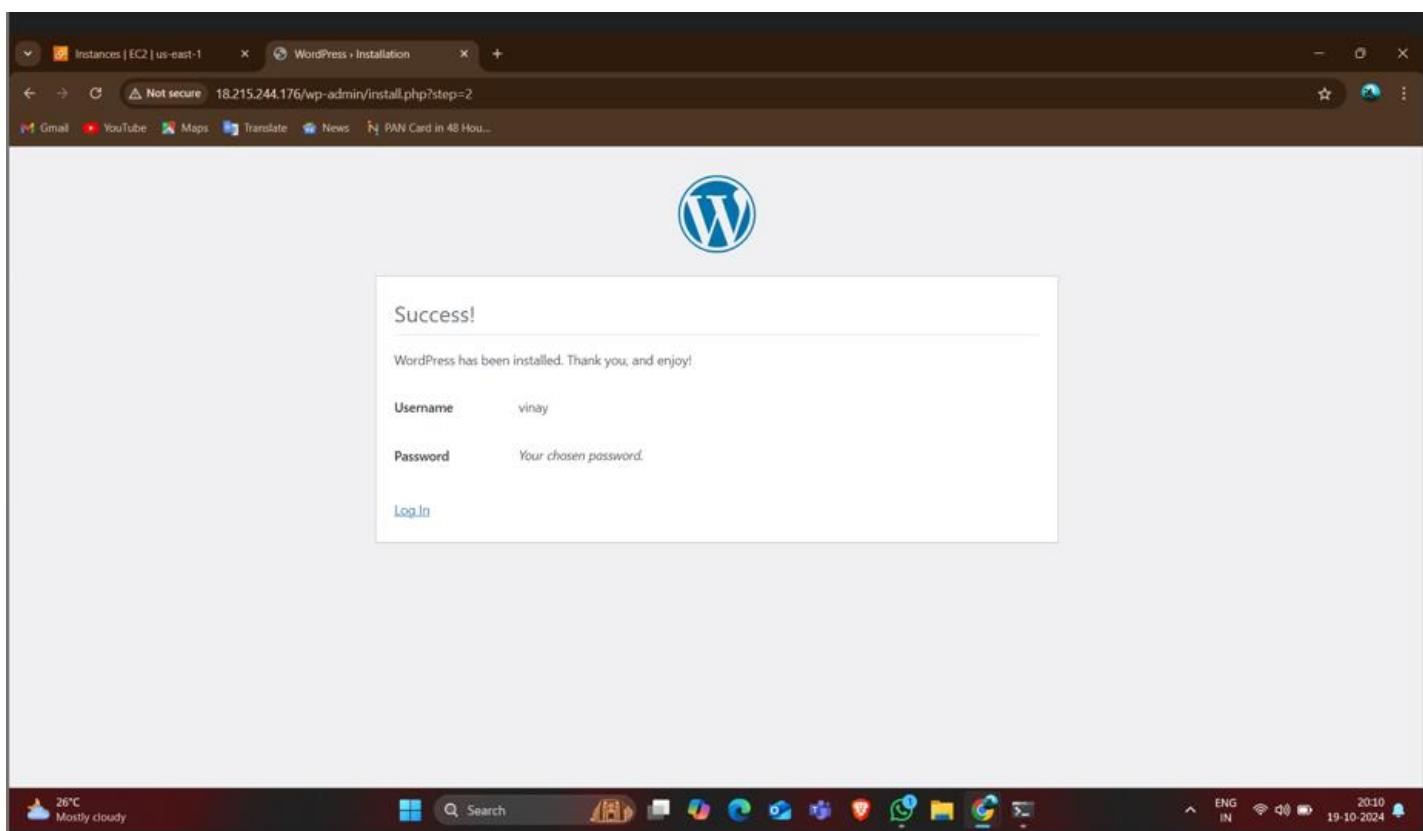
# Create wp-config.php from sample and update with DB details
cp /var/www/html/wp-config-sample.php /var/www/html/wp-config.php
sed -i "s/database_name_here/wordpress/" /var/www/html/wp-config.php
sed -i "s/username_here/wp_user/" /var/www/html/wp-config.php
sed -i "s/password_here/password/" /var/www/html/wp-config.php

# Restart Apache to apply changes
systemctl restart httpd
```

A checkbox at the bottom left of the terminal window is unchecked, indicating that the input is not base64-encoded.

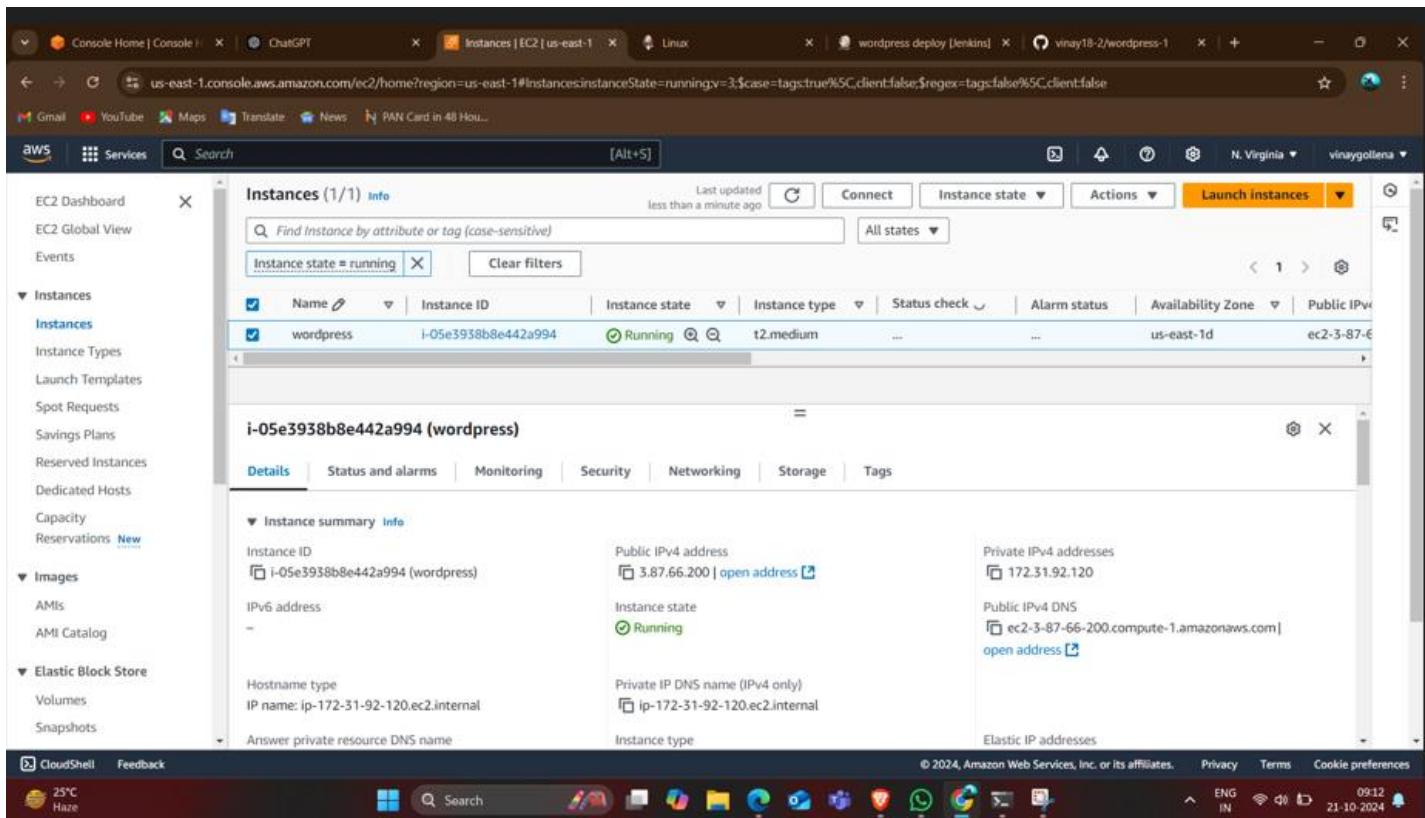
On the right side of the screen, a Snipping Tool window is open, showing a screenshot of the terminal window. A message in the Snipping Tool window says: "Screenshot copied to clipboard and saved. Select here to mark up and share."

The AWS CloudShell interface includes a CloudShell tab, a Feedback tab, a weather icon (26°C, Mostly cloudy), a search bar, and a system tray with various icons and status information (ENG IN, 19-10-2024).

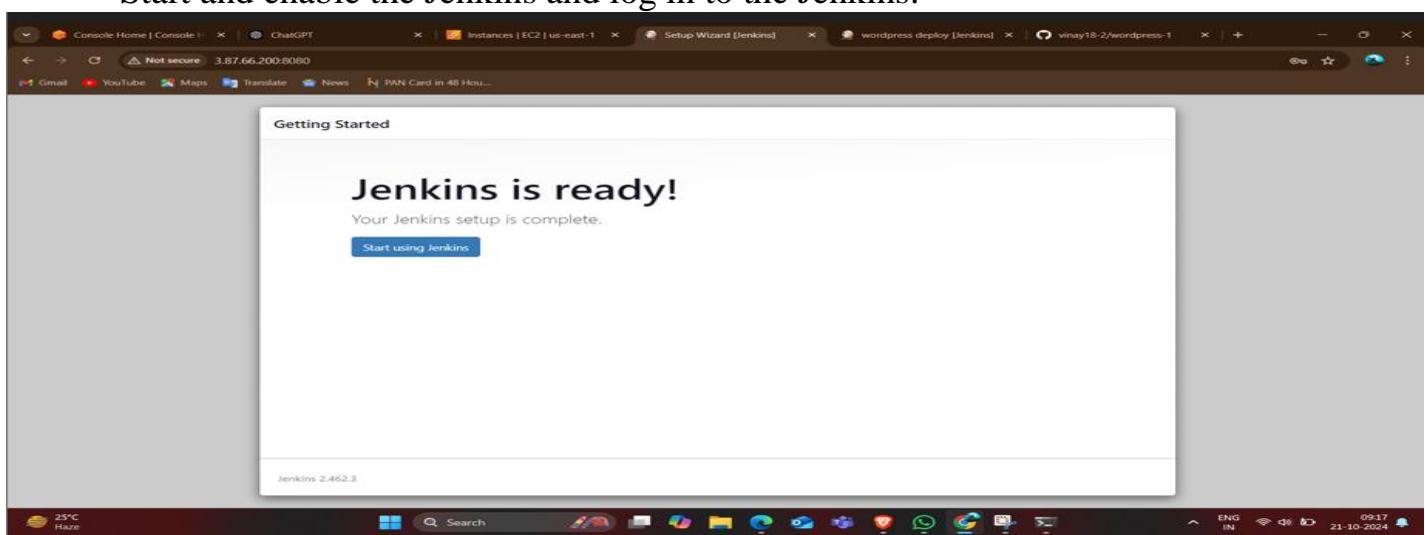


# Deploy WordPress web application by using git and jenkins execute shell (bash script)

- Launch the instance with t2.medium by allowing the security groups SSH,HTTP and All traffic , or provide the port 8080 which runs the Jenkins.



- Connect to the terminal and install the Jenkins as same as the 3<sup>rd</sup> method.
- Start and enable the Jenkins and log in to the Jenkins.



- Then create the new job and select the freestyle project.

The screenshot shows the Jenkins dashboard with the following elements:

- Left sidebar:** Buttons for "New Item", "Build History", "Manage Jenkins", and "My Views".
- Welcome section:** "Welcome to Jenkins!" and a message about setting up jobs.
- Build Queue:** Shows "No builds in the queue."
- Build Executor Status:** Shows 1 idle and 2 idle executors.
- Central area:**
  - "Create a job" button with a plus sign.
  - "Set up a distributed build" section with "Set up an agent", "Configure a cloud", and "Learn more about distributed builds".
- Bottom status bar:** REST API, Jenkins 2.462.3, ENG IN, 09:37, 21-10-2024.

The screenshot shows the "New Item" creation dialog with the following steps:

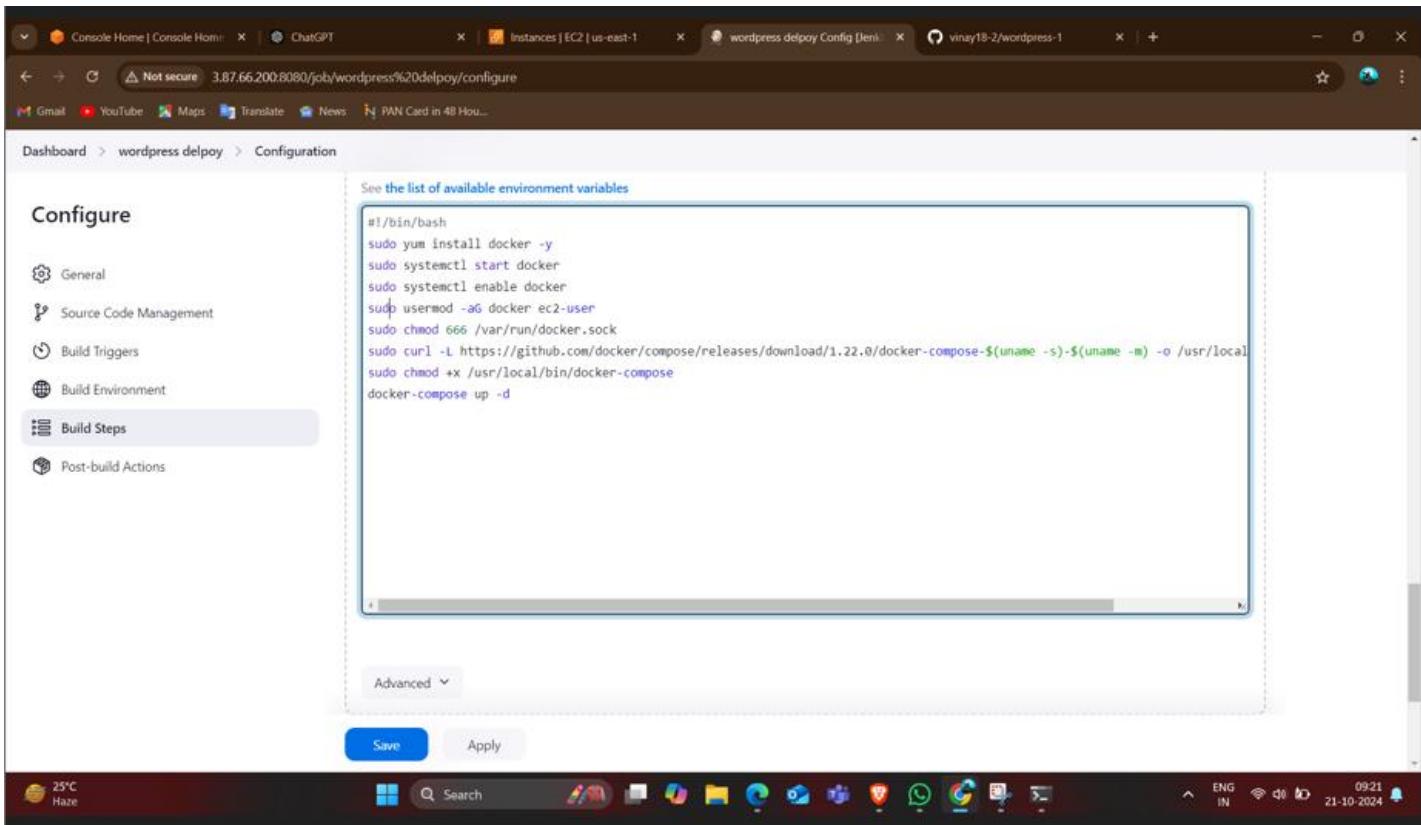
- Step 1:** "Enter an item name" field containing "wordpress deploy".
- Step 2:** "Select an item type" section:
  - Freestyle project:** Selected. Description: "Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.".
  - Pipeline:** Description: "Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.".
  - Multi-configuration project:** Description: "Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.".
  - Folder:**
- Step 3:** "OK" button at the bottom.

- Provide the description and select the git in source code management change the branch.

The screenshot shows the Jenkins job configuration page for a job named "wordpress deploy". The "General" tab is selected. The "Description" field contains "wordpressd deploy using execute shell". The "GitHub project" checkbox is checked, and the "Project url" field contains "https://github.com/vinay18-2/wordpress-1.git". The "Save" and "Apply" buttons are visible at the bottom.

The screenshot shows the Jenkins job configuration page for a job named "wordpress deploy". The "Source Code Management" tab is selected. The "Git" radio button is selected, and the "Repository URL" field contains "https://github.com/vinay18-2/wordpress-1.git". A red error message "Please enter Git repository." is displayed above the repository URL field. The "Save" and "Apply" buttons are visible at the bottom.

- Write the code in the execute shell and save it.



- We have to made some changes in visudo file by using “sudo visudo” Jenkins ALL=(ALL) NOPASSWD:ALL”

The screenshot shows a terminal window on a Windows desktop. The command "sudo visudo" has been run, displaying the contents of the /etc/sudoers file. The file includes several entries, notably:

```
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
# jenkins    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
-- INSERT --
```

The terminal window also shows system status at the bottom: 25°C Haze, ENG IN, 21-10-2024, 09:23.

- Then build now and observe the console output.

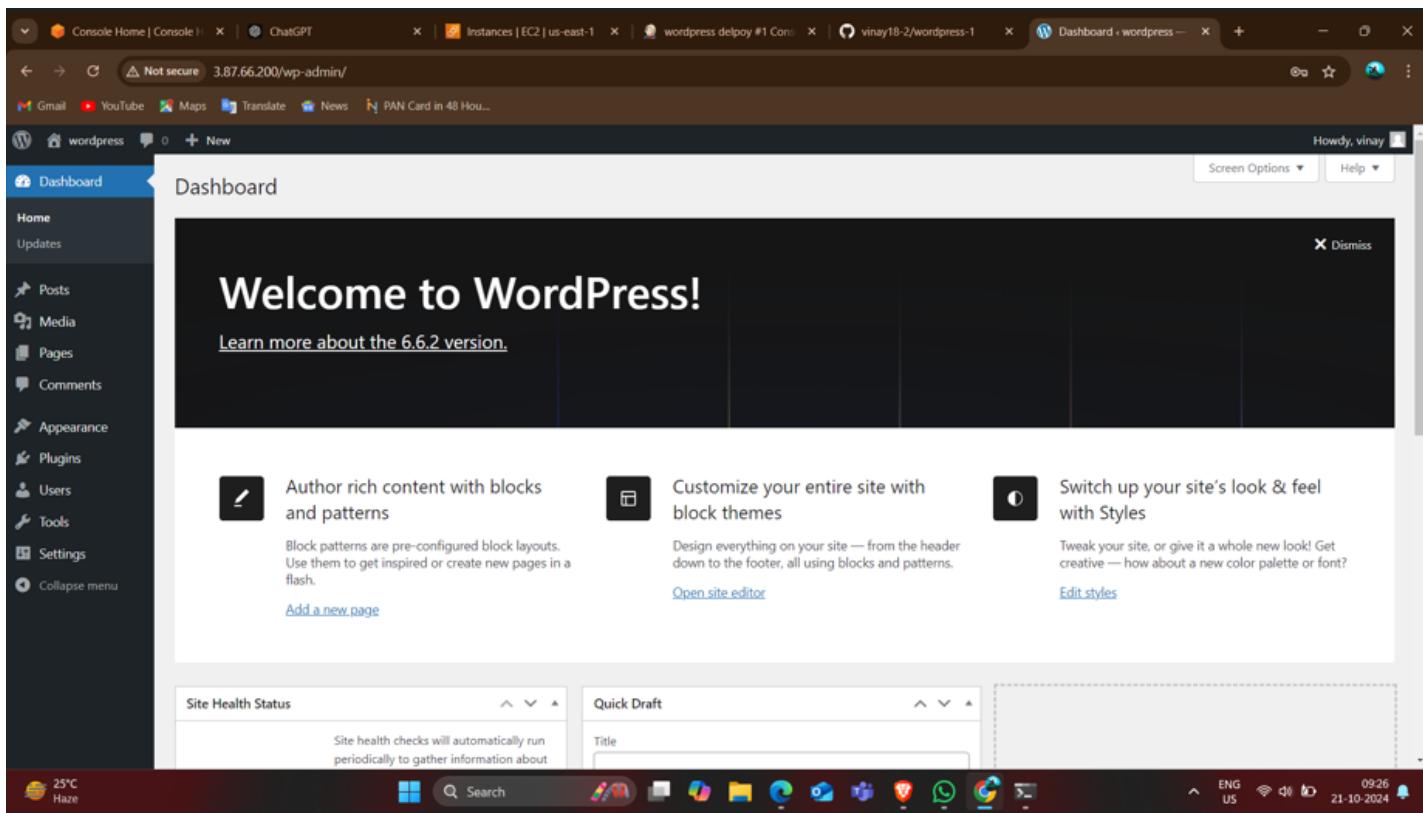
The screenshot shows two stacked Jenkins pages from a browser. The top page is for the 'wordpress deploy' project, displaying a status bar with 'Build scheduled' and a 'Build Now' button. The bottom page is the 'Console Output' for build #1, showing the command-line logs of the deployment process.

```

Started by user vinay g
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/wordpress deploy
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository https://github.com/vinay18-2/wordpress-1.git
> git init /var/lib/jenkins/workspace/wordpress deploy # timeout=10
Fetching upstream changes from https://github.com/vinay18-2/wordpress-1.git
> git --version # timeout=10
> git --version # 'git version 2.40.1'
> git fetch --tags --force --progress -- https://github.com/vinay18-2/wordpress-1.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git config remote.origin.url https://github.com/vinay18-2/wordpress-1.git # timeout=10
> git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out daed188fb792fa9787da76e0d3fdc4694e1a3ea0 (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f daed188fb792fa9787da76e0d3fdc4694e1a3ea0 # timeout=10
Commit message: "Create docker-compose.yaml"
First time build. Skipping changelog.
[wordpress deploy] $ /bin/bash /tmp/jenkins8529593807695162678.sh

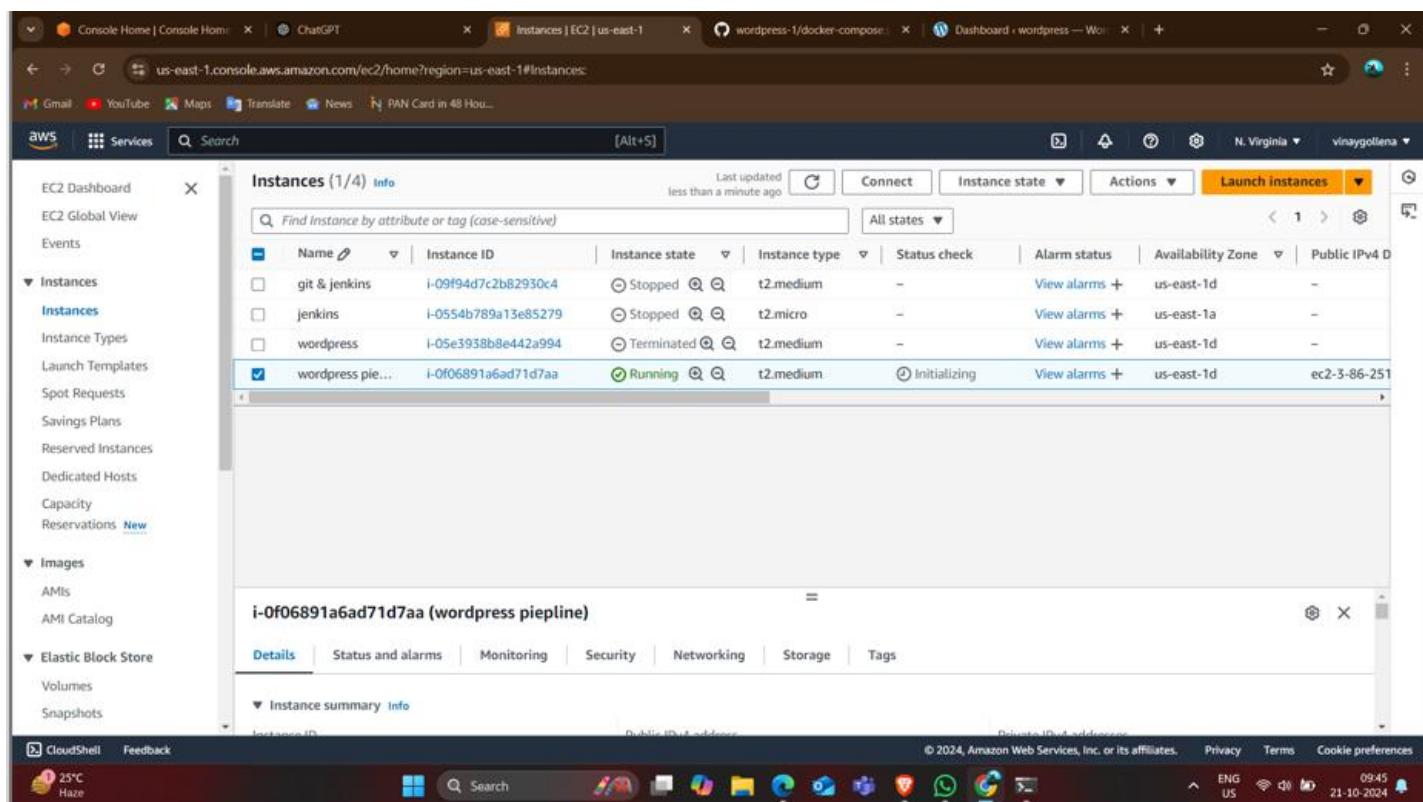
```

- Then copy the ip address and paste it in google with port 80 and browse it.



Deploy WordPress web application by using git and jenkins execute shell (bash script) create jenkins pipeline add build periodically and poll scm to initial job of pipeline and check the changes happened or not which are made in github repo

- Do the same as the method 6 or above method.
- Connect to the terminal and install Jenkins and git.
- Than start the Jenkins and login.



```

[ec2-user@ip-172-31-81-169 ~] libXrender.x86_64 0:0.9.10-1.amzn2.0.2 libXt.x86_64 0:1.1.5-3.amzn2.0.2
[ec2-user@ip-172-31-81-169 ~] libxcb.x86_64 0:1.12-1.amzn2.0.2 log4j-cve-2021-44228-hotpatch.noarch 0:1.3-7.amzn2
[ec2-user@ip-172-31-81-169 ~] python-javapackages.noarch 0:3.4.1-11.amzn2 python-lxml.x86_64 0:3.2.1-4.amzn2.0.6

Complete!
[ec2-user@ip-172-31-81-169 ~]$ sudo yum install jenkins -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Resolving Dependencies
--> Running transaction check
--> Package jenkins.noarch 0:2.462.3-1.1 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package           Arch         Version          Repository      Size
=====
Installing:
jenkins          noarch     2.462.3-1.1    jenkins        89 M

Transaction Summary
=====
Install 1 Package

Total download size: 89 M
Installed size: 89 M
Downloading packages:
jenkins-2.462.3-1.1.noarch.rpm | 89 MB  00:00:02
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : jenkins-2.462.3-1.1.noarch 1/1
  Verifying  : jenkins-2.462.3-1.1.noarch 1/1

Installed:
jenkins.noarch 0:2.462.3-1.1

Complete!
[ec2-user@ip-172-31-81-169 ~]$ |
```

25°C Haze

Search

ENG US 21-10-2024 09:48

Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Build Queue

No builds in the queue.

Build Executor Status

1 Idle  
2 Idle

Create a job

Add description

Set up a distributed build

Set up an agent

Configure a cloud

Learn more about distributed builds

REST API Jenkins 2.462.3

25°C Haze

Search

ENG US 21-10-2024 09:51

- Create the new project clonejob.

New Item

Enter an item name

Select an item type

**Freestyle project**  
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

**Pipeline**  
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

**Multi-configuration project**  
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

**Folder**

OK

Configure

Description

Plain text [Preview](#)

Discard old builds ?

GitHub project

Project url ?

Advanced ▾

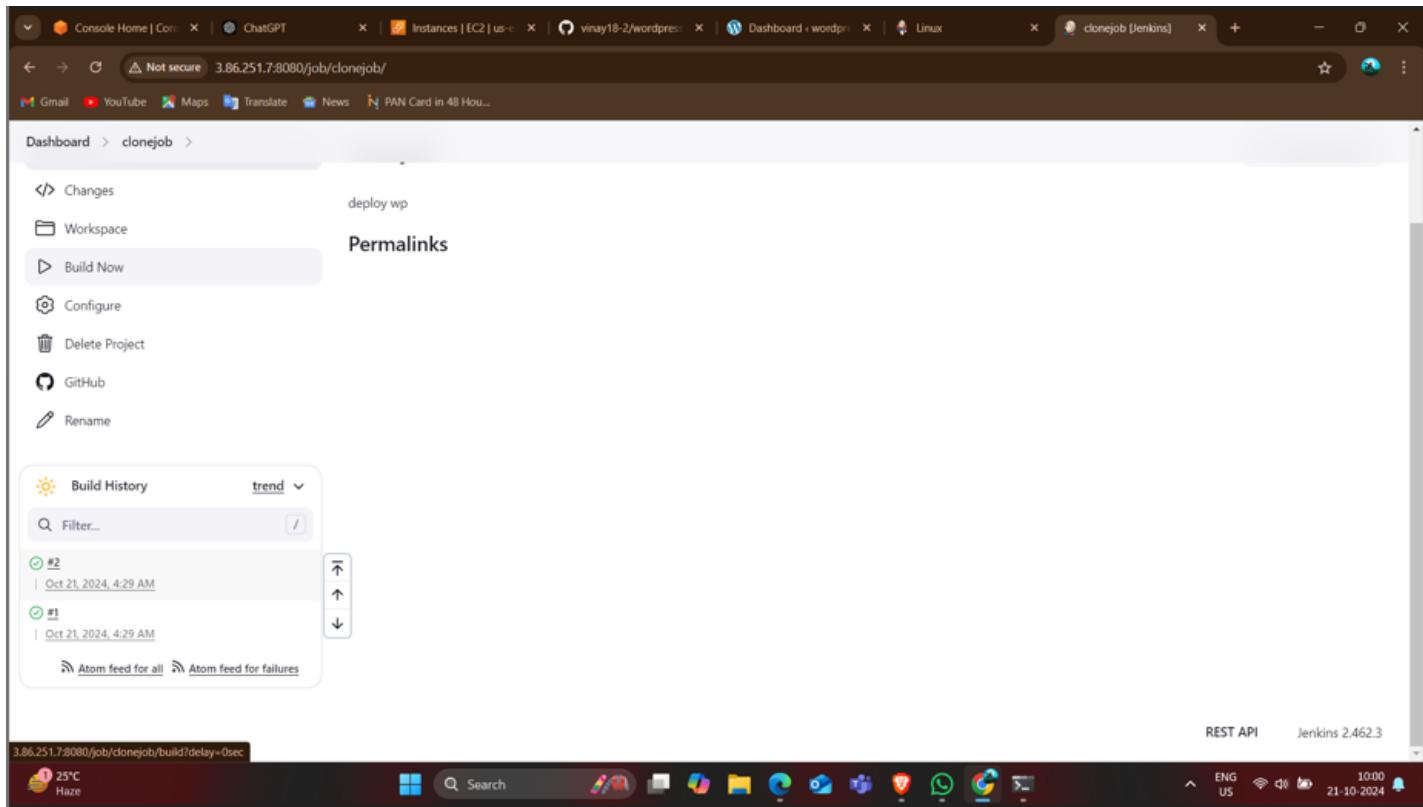
This project is parameterized ?

Throttle builds ?

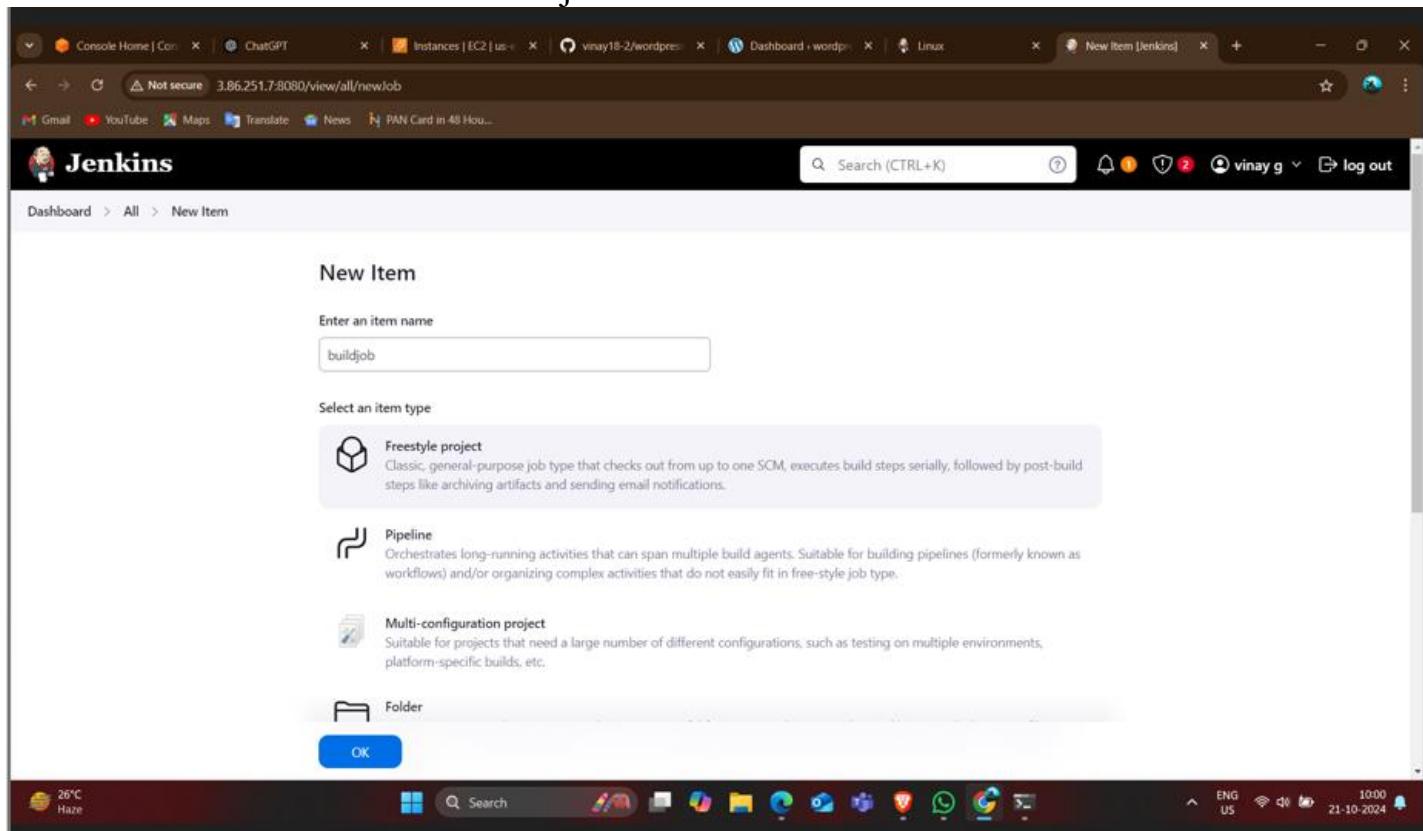
Execute concurrent builds if necessary ?

Save Apply

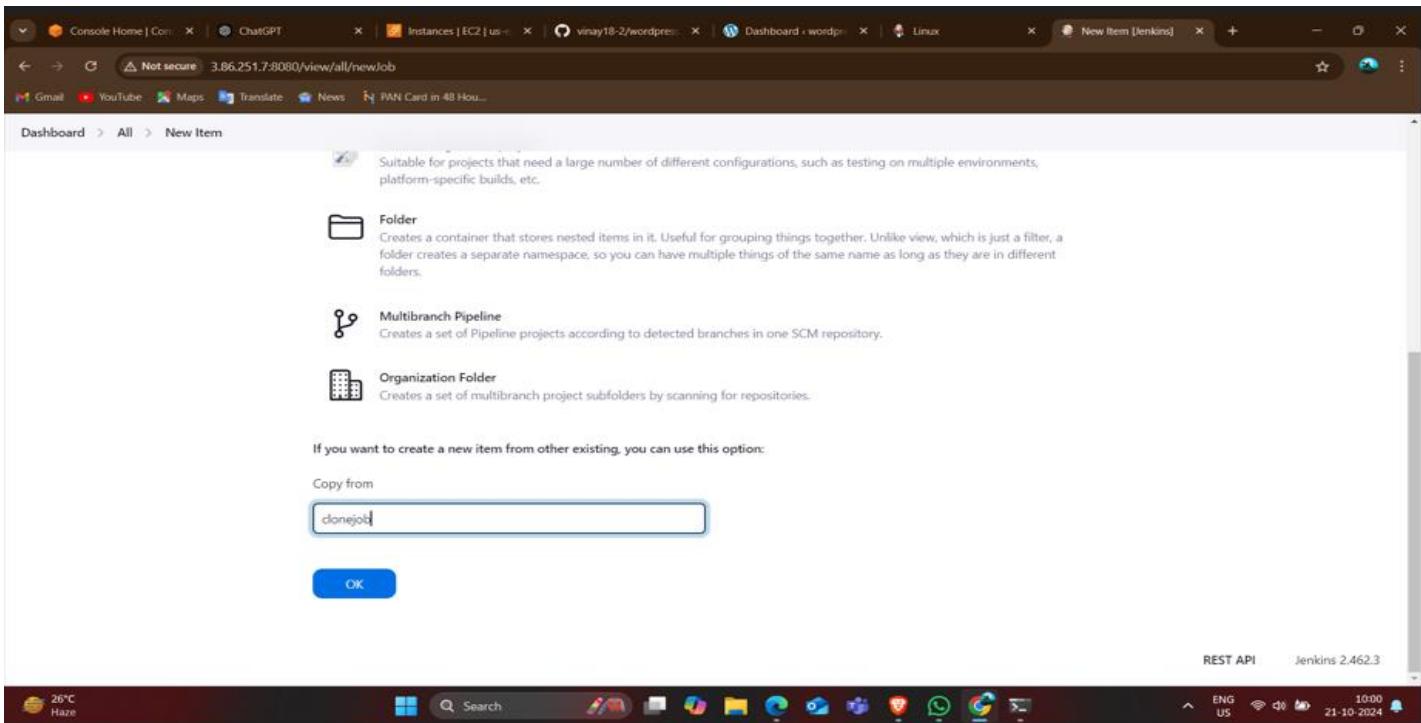
- Save it and build now.



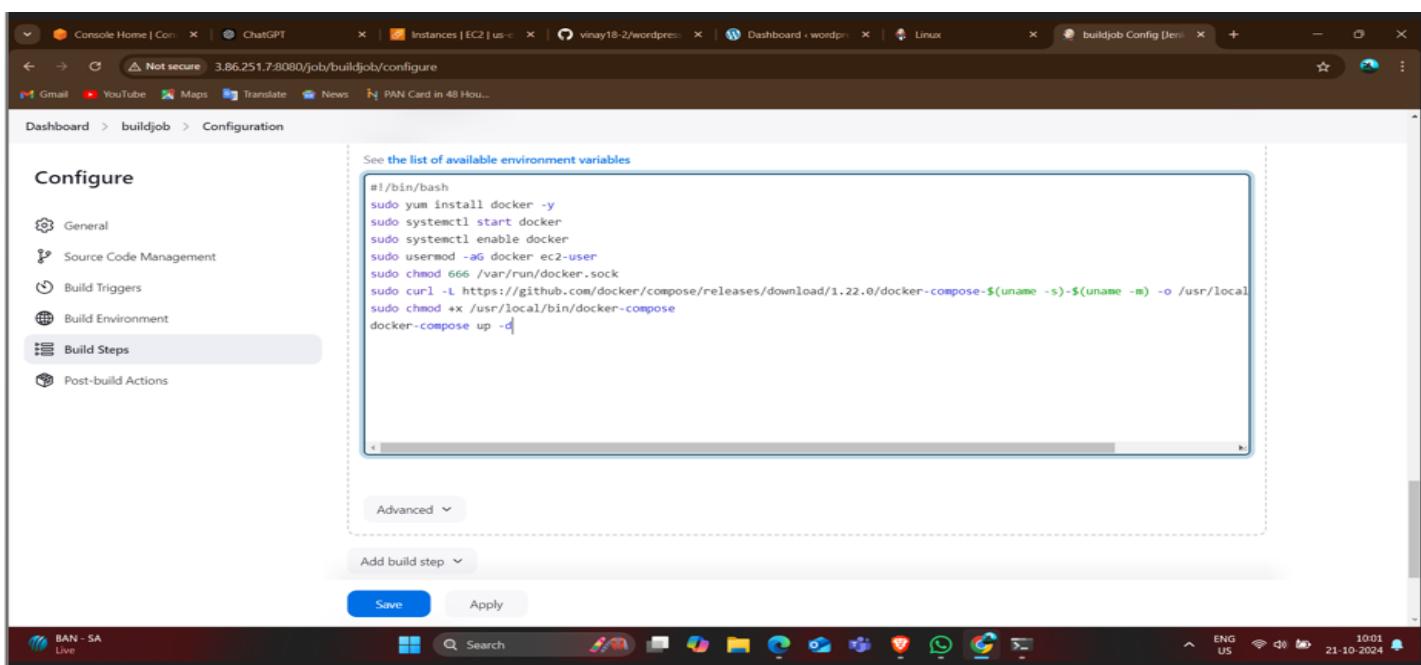
- Next create the new item buildjob.



- Copy the existing file to clone git and clonejob.



- Write the bash script in the execute shell to install docker and docker-compose and clone the git repo install git also and run the docker than start the docker enable it, all the commands should be in the execute shell.



- Then save it and to run build it and observe it.

The screenshot shows a Jenkins job named "buildjob". The job description is "deploy wp". The "Build History" section displays two builds: #2 (Oct 21, 2024, 4:33 AM) which was successful, and #1 (Oct 21, 2024, 4:32 AM) which failed. The Jenkins status bar at the bottom indicates the date as 21-10-2024.

- In order to build pipeline we have to install the plugin.Go to dashboard,select the manage Jenkins in that we have the plugins, select it and go to available plugins and search for the build pipeline plugin and install it.

The screenshot shows the Jenkins "Manage Jenkins > Plugins" page. The "Available plugins" tab is selected. A search bar at the top shows "build pipeline". The "Build Pipeline" plugin by "User Interface" is listed, version 2.0.2, released 5 months ago. It has a warning about a stored XSS vulnerability. Other listed plugins include "Webhook Step" and "Pipeline timeline".

**Plugins**

- Updates
- Available plugins
- Installed plugins
- Advanced settings
- Download progress**

**Download progress**

Preparation	
• Checking internet connectivity	Success
• Checking update center connectivity	Success
• Success	Success
Ionicons API	Success
Folders	Success
OWASP Markup Formatter	Success
ASM API	Success
JSON Path API	Success
Structs	Success
Pipeline: Step API	Success
Token Macro	Success
Build Timeout	Success
bouncycastle API	Success
Credentials	Success
Plain Credentials	Success
Variant	Success
SSH Credentials	Success
Credentials Binding	Success

- Create the pipeline.

**New view**

Name:

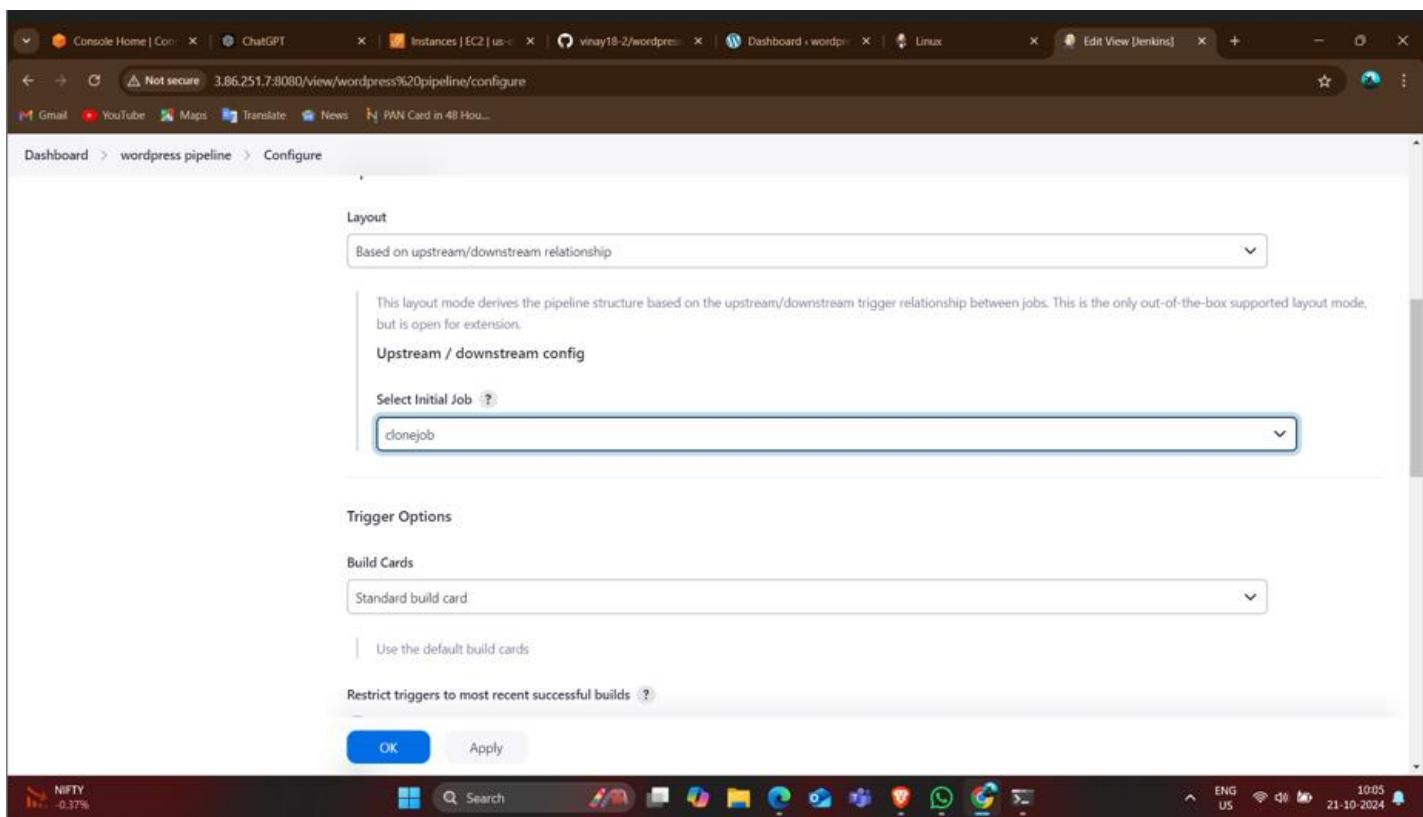
Type:

Build Pipeline View  
Shows the jobs in a build pipeline view. The complete pipeline of jobs that a version propagates through are shown as a row in the view.

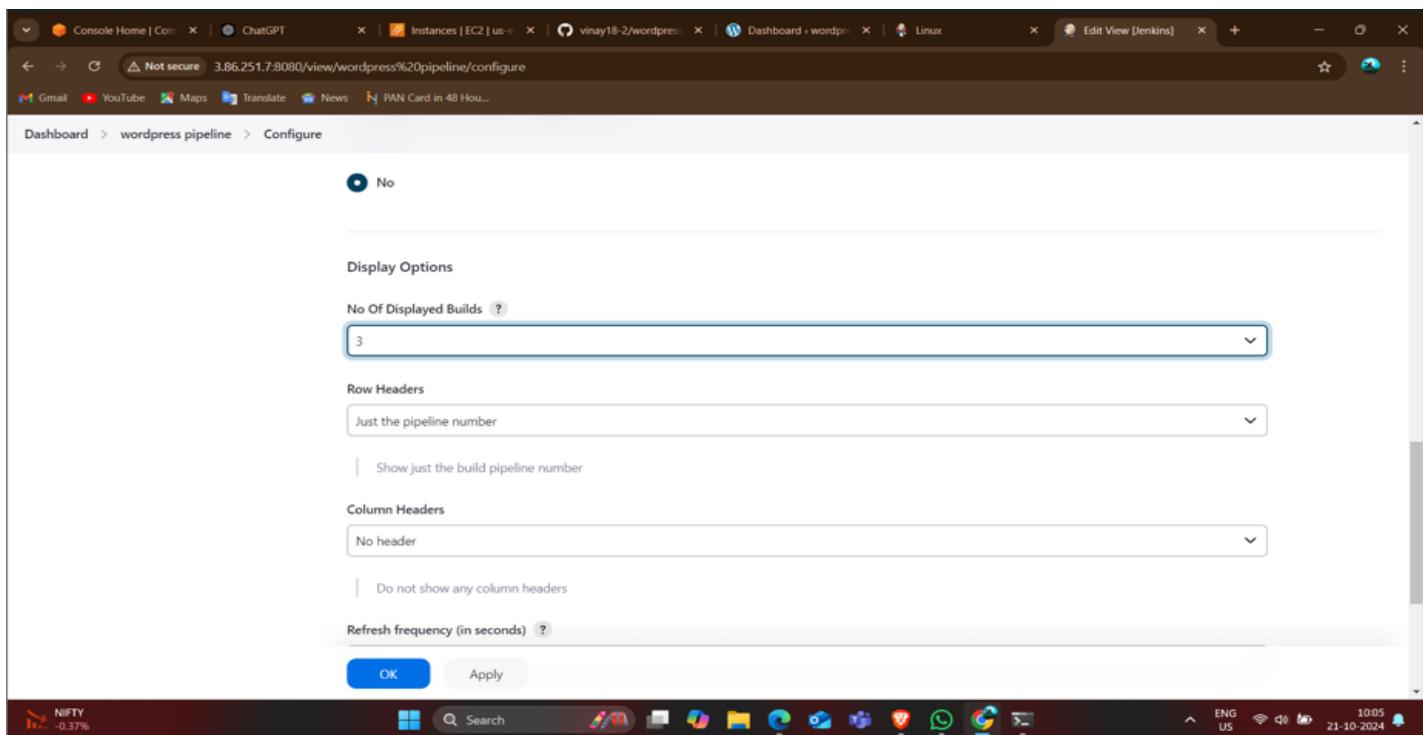
List View  
Shows items in a simple list format. You can choose which jobs are to be displayed in which view.

My View  
This view automatically displays all the jobs that the current user has an access to.

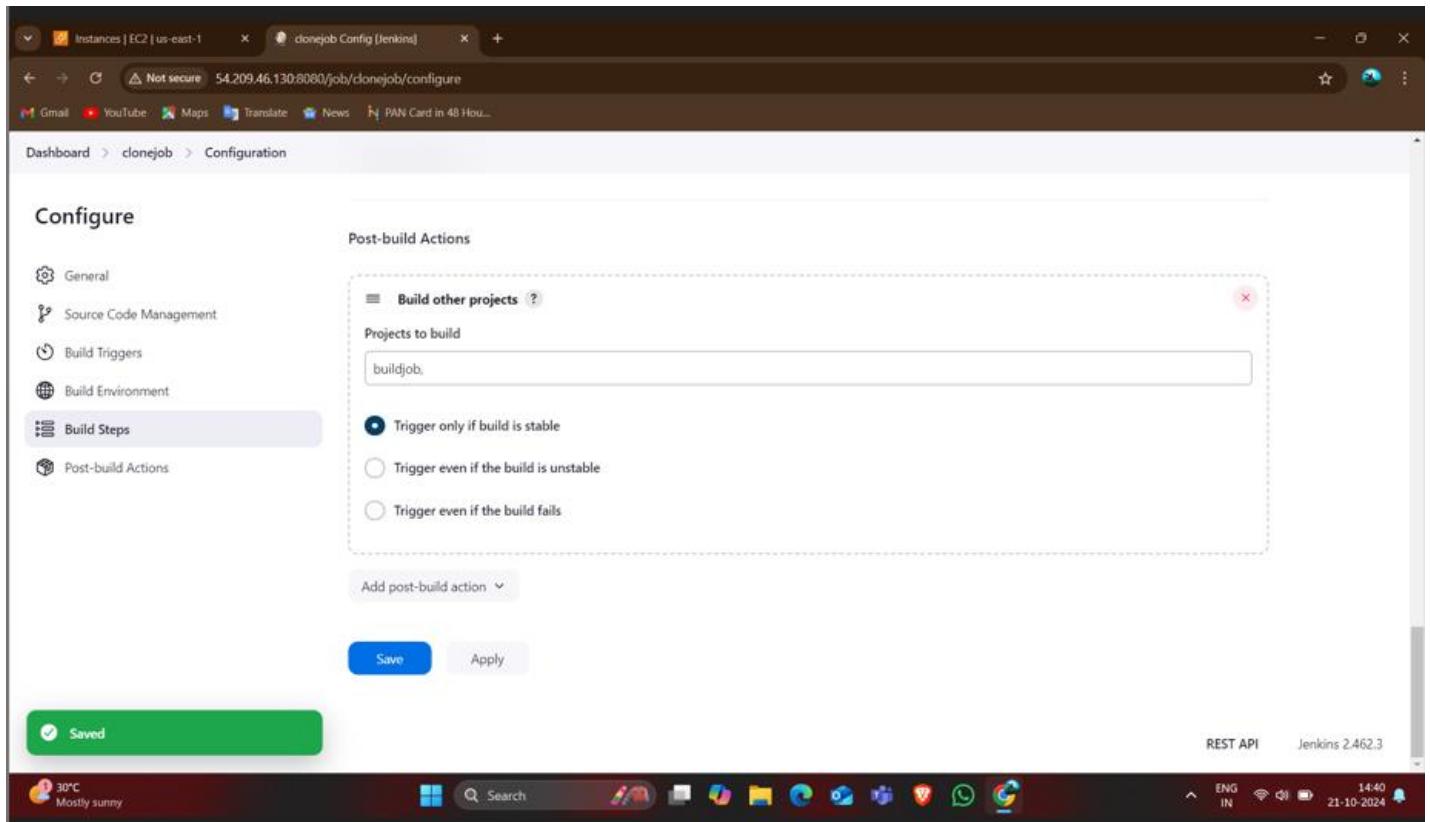
**Create**



- Here the no.of displayed builds is used to displays the how many builds we need that much count we have to provide.



- Select the post-build action, In that we have to select the buildjob to build the project. Then save and apply.



- Here the yaml code for build periodically.

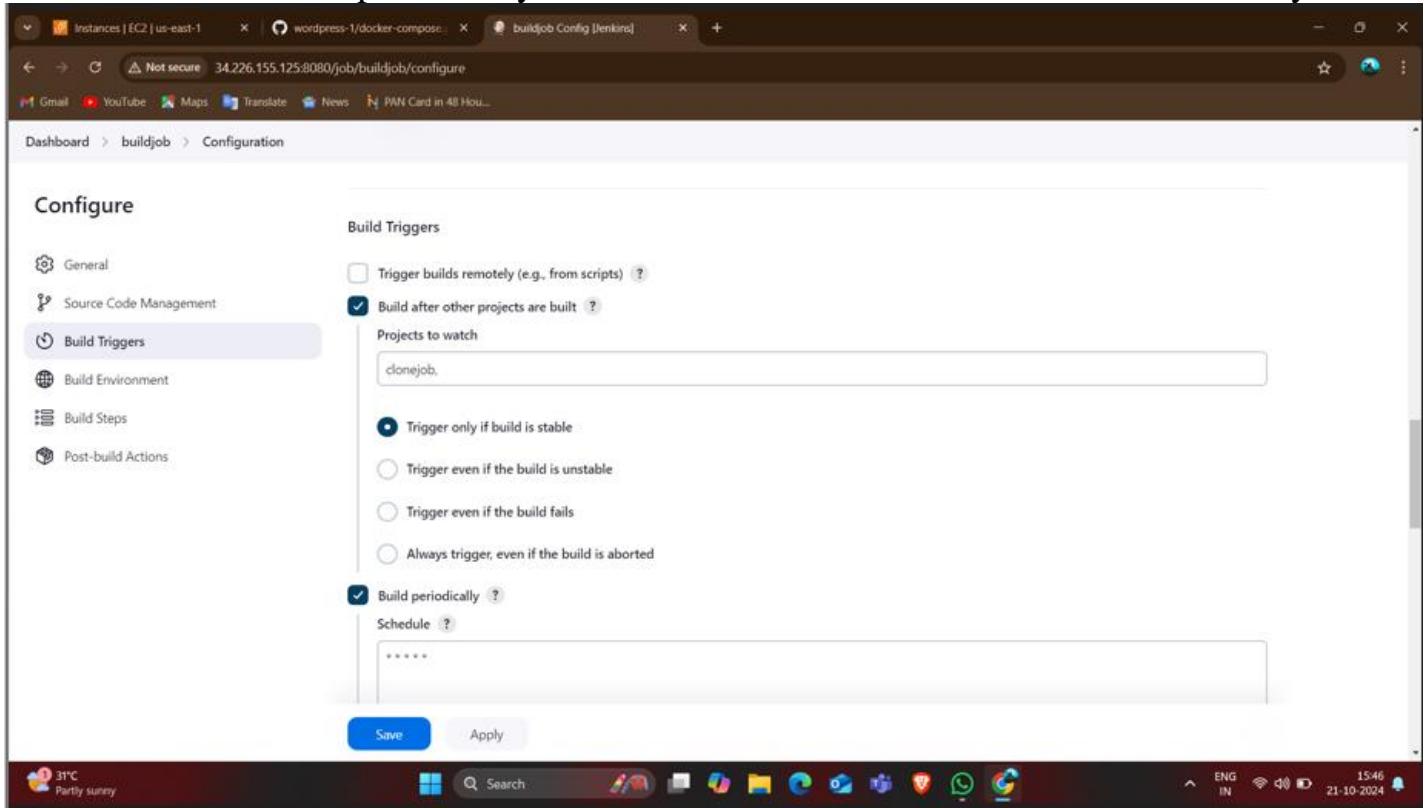
```

version: '3.5'
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=vinay
      - MYSQL_USER=vinay
      - MYSQL_PASSWORD=9966774360
  wordpress:
    image: wordpress:latest
    ports:
      - "80:80"
    restart: always
    environment:
      - WORDPRESS_DB_HOST=db
      - WORDPRESS_DB_USER=vinay
      - WORDPRESS_DB_PASSWORD=9966774360
      - WORDPRESS_DB_NAME=vinay
    volumes:
      db_data:

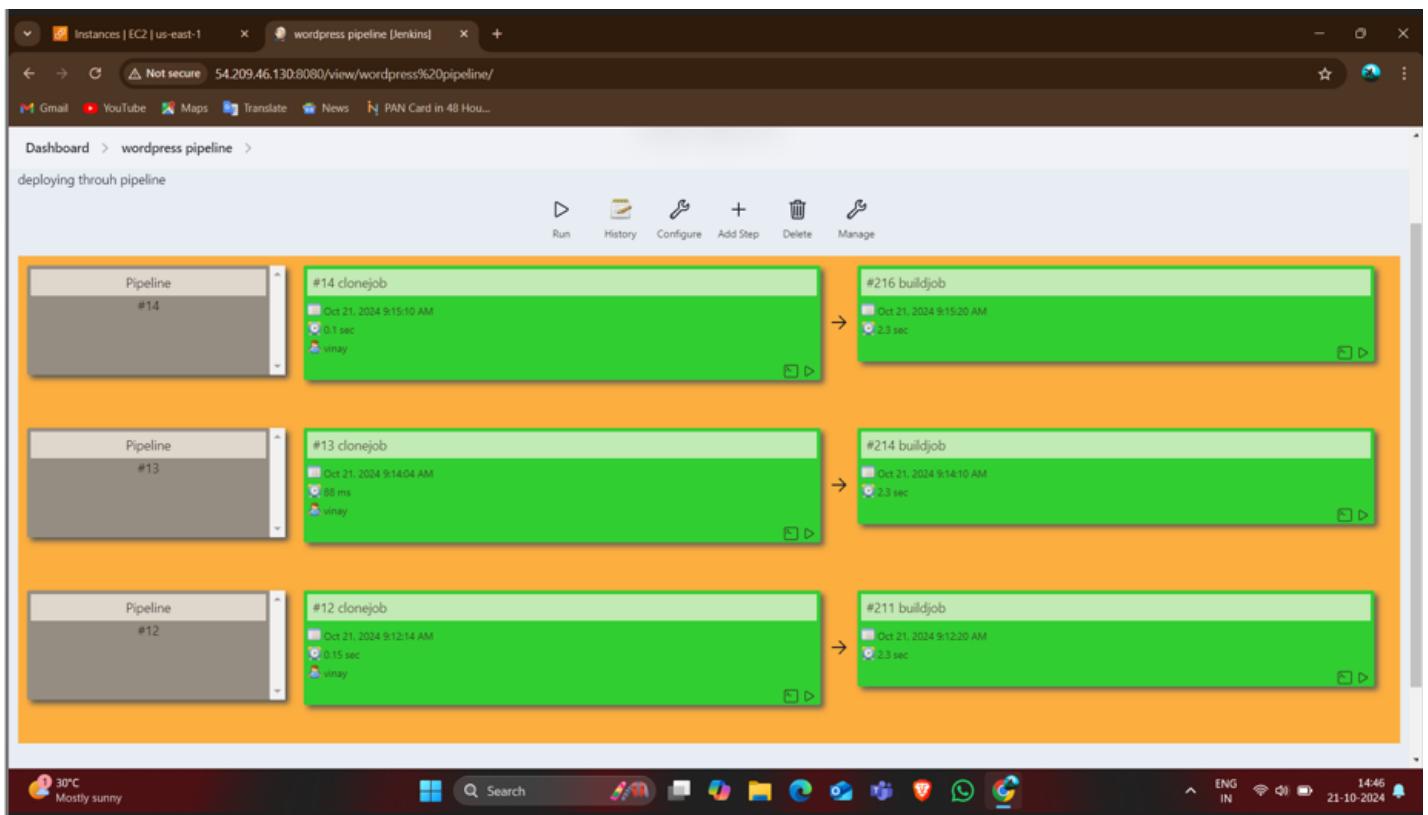
```

The screenshot shows a GitHub code editor displaying a `docker-compose.yaml` file for a WordPress application. The file defines two services: `db` and `wordpress`. The `db` service uses a MySQL 8.0.19 image and runs with the `--default-authentication-plugin=mysql_native_password` command. It has a volume mount at `/var/lib/mysql` named `db_data`. The `wordpress` service uses the `wordpress:latest` image and maps port 80 to 80. It depends on the `db` service and has its own environment variables for database connection. A note at the bottom of the editor says: 'Use Control + Shift + m to toggle the tab key moving focus. Alternatively, use esc then tab to move to the next interactive element on the page.'

- This is for build periodically. here \* \* \* \* \* indicates that the code runs for every time.



- Output for build periodically.



- Modified yaml code for poll scm.

```

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=wordpress1
      - MYSQL_DATABASE=vinay
      - MYSQL_USER=vinay
      - MYSQL_PASSWORD=9966774360

  wordpress:
    image: wordpress:latest
    ports:
      - "80:80"
    restart: always
    environment:
      - WORDPRESS_DB_HOST=db
      - WORDPRESS_DB_USER=vinay
      - WORDPRESS_DB_PASSWORD=9966774360
      - WORDPRESS_DB_NAME=vinay
    volumes:
      db_data:

```

- This is for poll scm .Here the schedule is \* \* \* \* \* it indicates runs every minute.

**Configure**

Poll SCM ?

Schedule

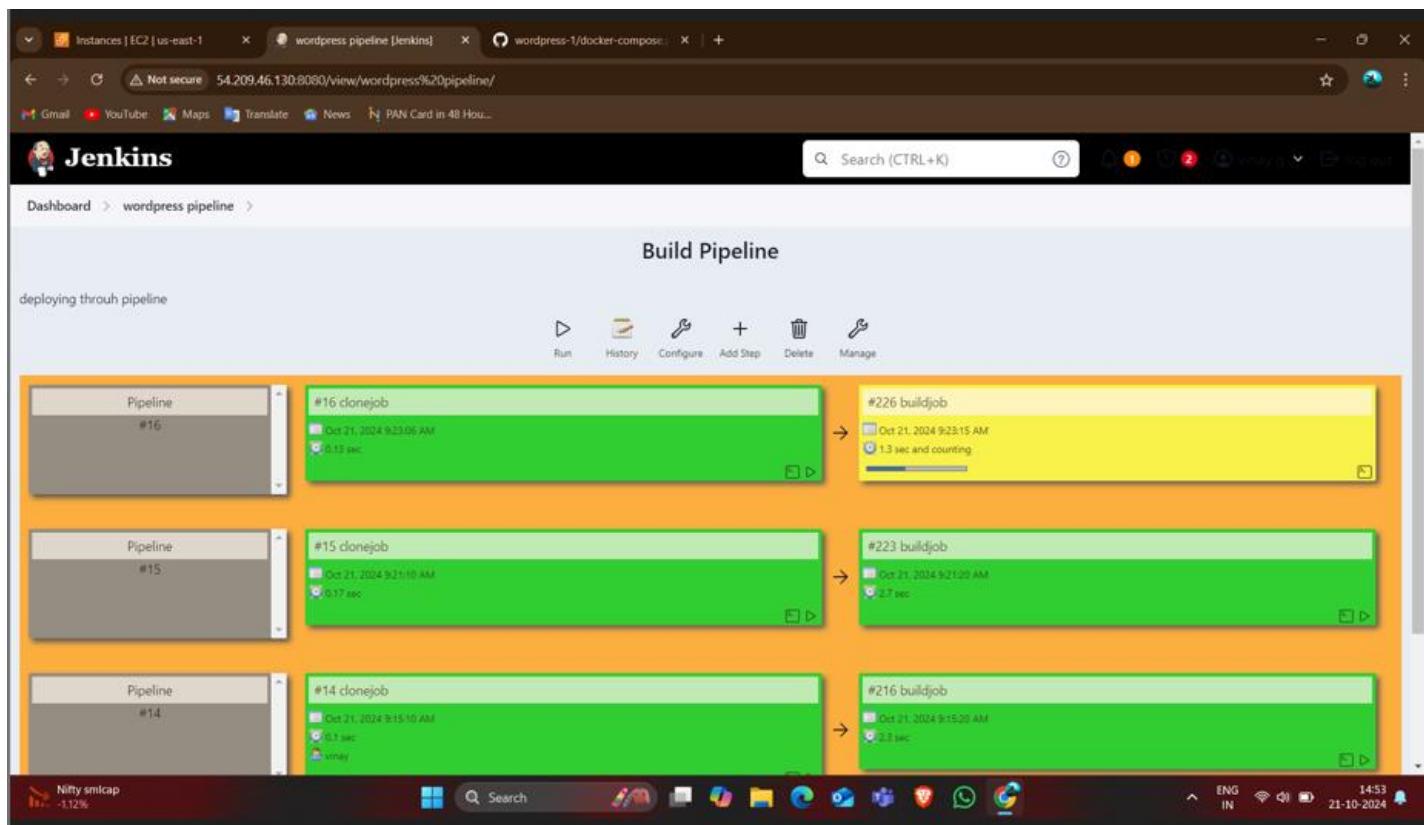
Ignore post-commit hooks ?

**Build Environment**

Delete workspace before build starts  
 Use secret text(s) or file(s) ?  
 Add timestamps to the Console Output

**Buttons:** Save, Apply

- Output for poll scm.



# Deploy WordPress web application by using terraform

- Launch the EC2 instance with t2.medium.
- Connect to the terminal and instll git “sudo yum install git -y”.

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with options like EC2 Dashboard, EC2 Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity, Reservations, Images (AMIs, AMI Catalog), and Elastic Block Store (Volumes, Snapshots). The main area displays 'Instances (1/2) Info' with a search bar and filters. Two instances are listed: 'tf wordpress' (running, t2.micro, public IP ec2-3-91-3-39-34.compute-1.amazonaws.com) and 'terraform-wp' (running, t2.micro, public IP ec2-3-91-3-39-34.compute-1.amazonaws.com). Below the instances, the 'Inbound rules' section shows three security group rules:

Name	Security group rule ID	Port range	Protocol	Source	Security groups
-	sgr-0d762d94d3f7458fe	22	TCP	0.0.0.0/0	terraform-2024*
-	sgr-0dcac924c26bb0ae8	443	TCP	0.0.0.0/0	terraform-2024*
-	sgr-01adc289c6abca284	80	TCP	0.0.0.0/0	terraform-2024*

The screenshot shows a terminal window on an Amazon Linux 2 system. The user has run 'ssh -i "vinay.pem" ec2-user@ec2-3-91-39-34.compute-1.amazonaws.com' and is prompted to connect to the host. They accept the connection and are warned about the end-of-life of Amazon Linux 2. Finally, they run 'git --version' to check the installed version.

```
Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

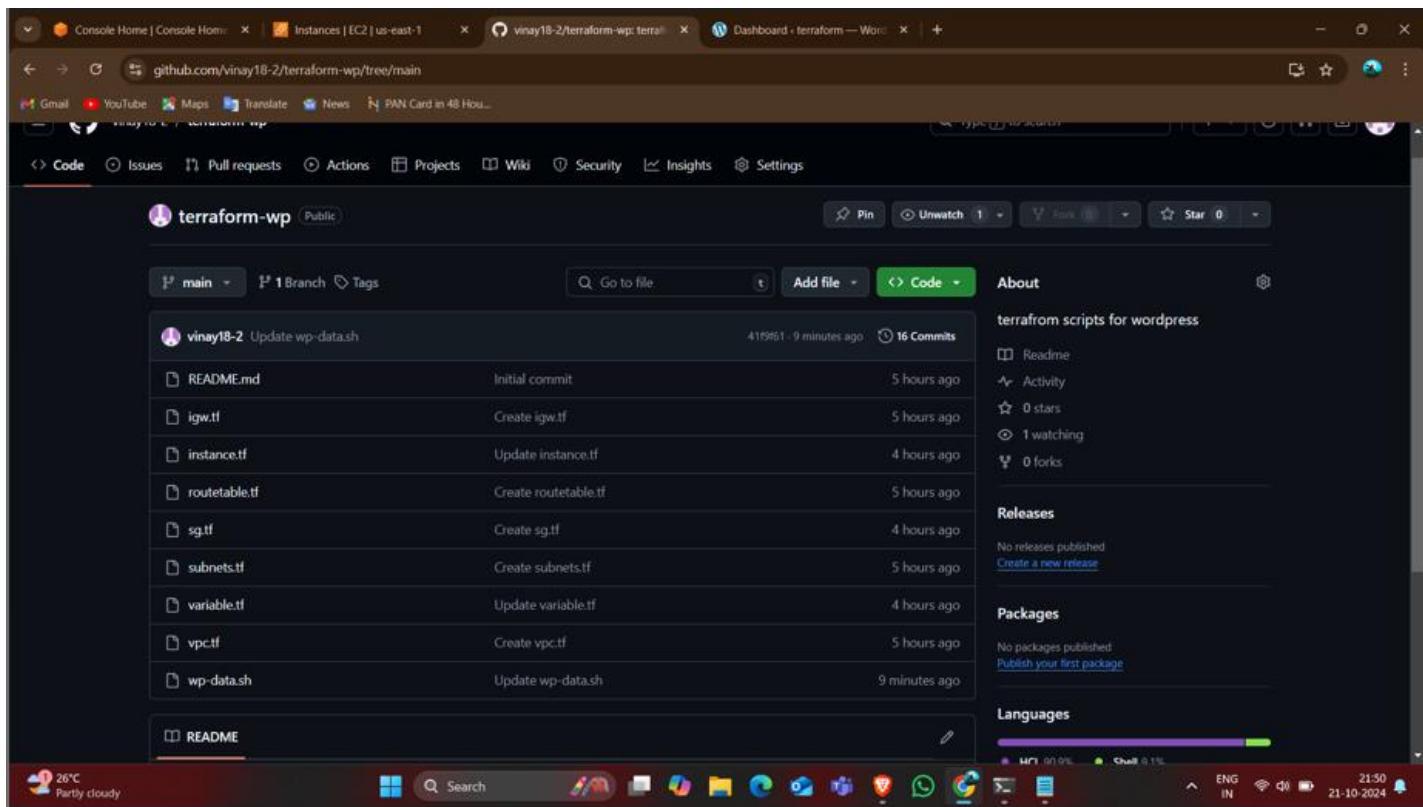
C:\Users\user>cd downloads

C:\Users\user\Downloads>ssh -i "vinay.pem" ec2-user@ec2-3-91-39-34.compute-1.amazonaws.com
The authenticity of host 'ec2-3-91-39-34.compute-1.amazonaws.com (3.91.39.34)' can't be established.
ED25519 key fingerprint is SHA256:WYMSXer7B04/vCvJ4Cc0JoDLWqQvL4MqInhRX+2v++0.
This host key is known by the following other names/addresses:
  C:\Users\user/.ssh/known_hosts:69: ec2-54-172-159-208.compute-1.amazonaws.com
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-3-91-39-34.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
Last login: Mon Oct 21 12:30:33 2024 from 106.221.184.77

# 
#_###_ Amazon Linux 2
##_\###_ AL2 End of Life is 2025-06-30.
##_\###_ V-->
##_\###_ A newer version of Amazon Linux is available!
##_\###_ /_/
##_\###_ /_/ Amazon Linux 2023, GA and supported until 2028-03-15.
##_\###_ /m/_/ https://aws.amazon.com/linux/amazon-linux-2023/

[ec2-user@ip-172-31-30-7 ~]$ git --version
git version 2.46.1
[ec2-user@ip-172-31-30-7 ~]$ |
```

- Install the terraform in the terminal.
- Then clone the git repositories of terraform scripts from git hub.
- We have to create the Instance with data.sh file,VPC,Subnets,Internet gateway,Route tables,Security groups and provider file.



- After cloning that execute the terraform commands:  
 “terraform init”  
 “terraform validate”  
 “terraform plan”  
 “terraform apply”  
 “terraform destroy”

```
ec2-user@ip-172-31-30-7:~/tx ~ + ^
```

```
-rw-rw-r-- 1 ec2-user ec2-user 1434 Oct 21 16:12 routetable.tf
-rw-rw-r-- 1 ec2-user ec2-user 627 Oct 21 16:12 sg.tf
-rw-rw-r-- 1 ec2-user ec2-user 1796 Oct 21 16:12 subnets.tf
-rw-rw-r-- 1 ec2-user ec2-user 695 Oct 21 16:12 variable.tf
-rw-rw-r-- 1 ec2-user ec2-user 161 Oct 21 16:12 vpc.tf
-rw-rw-r-- 1 ec2-user ec2-user 575 Oct 21 16:12 wp-data.sh
[ec2-user@ip-172-31-30-7 terraform-wp]$ sudo vi provider.tf
[ec2-user@ip-172-31-30-7 terraform-wp]$ "provider.tf" [New] 5L, 142B written
[ec2-user@ip-172-31-30-7 terraform-wp]$ ll
total 40
-rw-rw-r-- 1 ec2-user ec2-user 114 Oct 21 16:12 igw.tf
-rw-rw-r-- 1 ec2-user ec2-user 396 Oct 21 16:12 instance.tf
-rw-rw-r-- 1 root root 142 Oct 21 16:13 provider.tf
-rw-rw-r-- 1 ec2-user ec2-user 47 Oct 21 16:12 README.md
-rw-rw-r-- 1 ec2-user ec2-user 1434 Oct 21 16:12 routetable.tf
-rw-rw-r-- 1 ec2-user ec2-user 627 Oct 21 16:12 sg.tf
-rw-rw-r-- 1 ec2-user ec2-user 1796 Oct 21 16:12 subnets.tf
-rw-rw-r-- 1 ec2-user ec2-user 695 Oct 21 16:12 variable.tf
-rw-rw-r-- 1 ec2-user ec2-user 161 Oct 21 16:12 vpc.tf
-rw-rw-r-- 1 ec2-user ec2-user 575 Oct 21 16:12 wp-data.sh
[ec2-user@ip-172-31-30-7 terraform-wp]$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.72.1...
- Installed hashicorp/aws v5.72.1 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider selections it made above. Include this file in your version control repository so that Terraform can guarantee to make the same selections by default when 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.
[ec2-user@ip-172-31-30-7 terraform-wp]$ |
```



```
26°C Partly cloudy Search ENG IN 21-10-2024 21:43
```

```
ec2-user@ip-172-31-30-7:~/tx ~ + ^
```

```
- Installed hashicorp/aws v5.72.1 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider selections it made above. Include this file in your version control repository so that Terraform can guarantee to make the same selections by default when 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.
[ec2-user@ip-172-31-30-7 terraform-wp]$ terraform validate
```

```
Error: Reference to undeclared resource
  on routetable.tf line 29, in resource "aws_route_table_association" "vinayrt3":
  29:   subnet_id      = aws_subnet.vinaysubnet3.id

A managed resource "aws_subnet" "vinaysubnet3" has not been declared in the root module.

[ec2-user@ip-172-31-30-7 terraform-wp]$ sudo vi subnets.tf
[ec2-user@ip-172-31-30-7 terraform-wp]$ "subnets.tf" 71L, 1796B written
[ec2-user@ip-172-31-30-7 terraform-wp]$ terraform validate
Success! The configuration is valid.

[ec2-user@ip-172-31-30-7 terraform-wp]$ 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.vinay-1 will be created
+ resource "aws_instance" "vinay-1" {
  + ami           = "ami-0e54eba7c51c234f6"
  + arn           = (known after apply)
  + associate_public_ip_address = true
```



```
26°C Partly cloudy Search ENG IN 21-10-2024 21:45
```

```

ec2-user@ip-172-31-30-7:~/tx ~ + 
aws_vpc.vinayvpc: Creation complete after 2s [id=vpc-02c684cc7609be938]
aws_subnet.vinaypsubnet3: Creating...
aws_subnet.vinaypsubnet1: Creating...
aws_internet_gateway.vinaygateway: Creating...
aws_security_group.vinaysg: Creating...
aws_subnet.vinayprsubnet6: Creating...
aws_subnet.vinaypsubnet4: Creating...
aws_subnet.vinayprsubnet5: Creating...
aws_subnet.vinayprsubnet2: Creating...
aws_internet_gateway.vinaygateway: Creation complete after 0s [id=igw-0ee3647bb6024f1a8]
aws_route_table.vinayroute: Creating...
aws_subnet.vinayprsubnet6: Creation complete after 0s [id=subnet-0bbfb351cd9f14362]
aws_subnet.vinayprsubnet5: Creation complete after 0s [id=subnet-02c84f2eb351d9840]
aws_route_table.vinayroute: Creation complete after 1s [id=rtb-0b8e6f85908a23c1b]
aws_route_table_association.vinayrt5: Creating...
aws_route_table_association.vinayrt6: Creating...
aws_route_table_association.vinayrt5: Creation complete after 0s [id=rtbassoc-085e0ab870d5721e3]
aws_route_table_association.vinayrt6: Creation complete after 0s [id=rtbassoc-01bd43cf689759e93]
aws_security_group.vinaysg: Creation complete after 2s [id=sg-0a96334bc34c9880fe]
aws_subnet.vinaypsubnet3: Still creating... [10s elapsed]
aws_subnet.vinaypsubnet1: Still creating... [10s elapsed]
aws_subnet.vinaypsubnet4: Still creating... [10s elapsed]
aws_subnet.vinaypsubnet2: Still creating... [10s elapsed]
aws_subnet.vinaypsubnet3: Creation complete after 11s [id=subnet-0147c2418e886df2a]
aws_subnet.vinaypsubnet1: Creation complete after 11s [id=subnet-0bc83d9ff4cdaf62]
aws_route_table_association.vinayrt3: Creating...
aws_subnet.vinaypsubnet2: Creation complete after 11s [id=subnet-07c544ca6c66f6e44]
aws_instance.vinay-1: Creating...
aws_route_table_association.vinayrt1: Creating...
aws_route_table_association.vinayrt2: Creating...
aws_subnet.vinaypsubnet4: Creation complete after 11s [id=subnet-0ca9ea66eb70e0b1b]
aws_route_table_association.vinayrt4: Creating...
aws_route_table_association.vinayrt3: Creation complete after 0s [id=rtbassoc-091ffed27e22f6095]
aws_route_table_association.vinayrt1: Creation complete after 0s [id=rtbassoc-08ba1db2401c4489]
aws_route_table_association.vinayrt2: Creation complete after 0s [id=rtbassoc-00c0382296241c5e1]
aws_route_table_association.vinayrt4: Creation complete after 0s [id=rtbassoc-02deaa843c2570be0]
aws_instance.vinay-1: Still creating... [10s elapsed]
aws_instance.vinay-1: Creation complete after 12s [id=i-0d32b1b18d6b1bdb7]

Apply complete! Resources: 17 added, 0 changed, 0 destroyed.
[ec2-user@ip-172-31-30-7 terraform-wp]$

```

26°C Partly cloudy ENG IN 21-10-2024 21:46

- Here the results of terraform apply applying.

The screenshot shows the AWS VPC dashboard with the following details:

- VPCs:** vinayvpc (Available, IPv4 CIDR: 10.0.0.0/16)
- Subnets:** vinaypsubnet1 (us-east-1a), vinaypsubnet2 (us-east-1b)
- Route Tables:** rtb-0c2eac0f7fc849ab1, igw-0ce356

The screenshot shows the AWS EC2 Instances dashboard with the following details:

- Instances:** tf wordpress (i-0d32b1b18d6b1bdb7, t2.micro, Running)
- Public IP:** 34.229.153.108

- Outputs.
- Copy the created instance ip and browse it in google.

The image contains two screenshots of a Microsoft Edge browser window. The top screenshot shows the 'Install.php' page where language selection is being made. A dropdown menu lists various languages, including English (United States) at the top. The bottom screenshot shows the WordPress dashboard after the installation is complete. The dashboard features a 'Welcome to WordPress!' message, navigation links for Posts, Media, Pages, Comments, Appearance, Plugins, Users, Tools, and Settings, and sections for Site Health Status and Quick Draft.

**Language Selection Screenshot:**

English (United States)

- Afrikaans
- አማርኛ
- Aragonés
- العربية
- العربية المغربية
- অসমীয়া
- گوئىچى ئۇيغۇرچى
- Azərbaycan dilı
- Беларуская мова
- Български
- বাংলা
- ଓଡିଆ
- Bosanski
- Català
- Cebuano
- Čeština
- Cymraeg
- Dansk
- Deutsch (Schweiz)
- Deutsch

**Dashboard Screenshot:**

# Welcome to WordPress!

Learn more about the 6.6.2 version.

**Dashboard Options:**

- Posts
- Media
- Pages
- Comments
- Appearance
- Plugins
- Users
- Tools
- Settings

**Site Health Status:**

Site health checks will automatically run periodically to gather information about [redacted].

**Quick Draft:**

Title [redacted]

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.

- Launch instance with t2.medium
- Allow security groups ssh.http and all traffic.
- Connect it to the terminal.

The screenshot shows the AWS EC2 Instances page. The left sidebar is collapsed. The main area displays a table of instances:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
jenkins-8	i-0e24ffd0f535b0d55	Stopped	t2.medium	-	<a href="#">View alarms</a>	us-east-1d	-
<b>tf wordpress</b>	<b>i-0923382fbdc62d968</b>	<b>Running</b>	<b>t2.micro</b>	<b>Initializing</b>	<a href="#">View alarms</a>	us-east-1a	-
wp-8	i-0db225051cc7a2f33	Running	t2.medium	2/2 checks passed	<a href="#">View alarms</a>	us-east-1d	ec2-54-

Below the table, the details for the selected instance (tf wordpress) are shown:

**i-0923382fbdc62d968 (tf wordpress)**

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

**Instance summary**

Instance ID i-0923382fbdc62d968	Public IPv4 address 54.227.96.179   <a href="#">open address</a>	Private IPv4 addresses 10.0.1.68
IPv6 address -	Instance state Running	Public IPv4 DNS -
Hostname type IP name: ip-10-0-1-68.ec2.internal	Private IP DNS name (IPv4 only) ip-10-0-1-68.ec2.internal	Elastic IP addresses -
Answer private resource DNS name -	Instance type t2.micro	

- After connected to the terminal update it.
- Install git.
- Install Jenkins.
- Start and enable the Jenkins and check the status of the Jenkins.

```

ec2-user@ip-172-31-93-11:~ + 
Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user>cd downloads

C:\Users\user\Downloads>ssh -i "vinay.pem" ec2-user@ec2-54-158-110-103.compute-1.amazonaws.com
The authenticity of host 'ec2-54-158-110-103.compute-1.amazonaws.com (54.158.110.103)' can't be established.
ED25519 key fingerprint is SHA256:DK7HZNXuofWLlm4QPYom+Vtix2tYLMPO/FJP8ugPu2s.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-54-158-110-103.compute-1.amazonaws.com' (ED25519) to the list of known hosts.

#_
/_###_ Amazon Linux 2
\### AL2 End of Life is 2025-06-30.
#/ V-->
--- A newer version of Amazon Linux is available!
/_/_/ Amazon Linux 2023, GA and supported until 2028-03-15.
/m/ https://aws.amazon.com/linux/amazon-linux-2023/

[ec2-user@ip-172-31-93-11 ~]$ sudo yum update -
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
No packages marked for update
[ec2-user@ip-172-31-93-11 ~]$ sudo yum install git -
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) 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 0:2.40.1-1.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

24°C Mostly clear Search ENG IN 23:44 21-10-2024

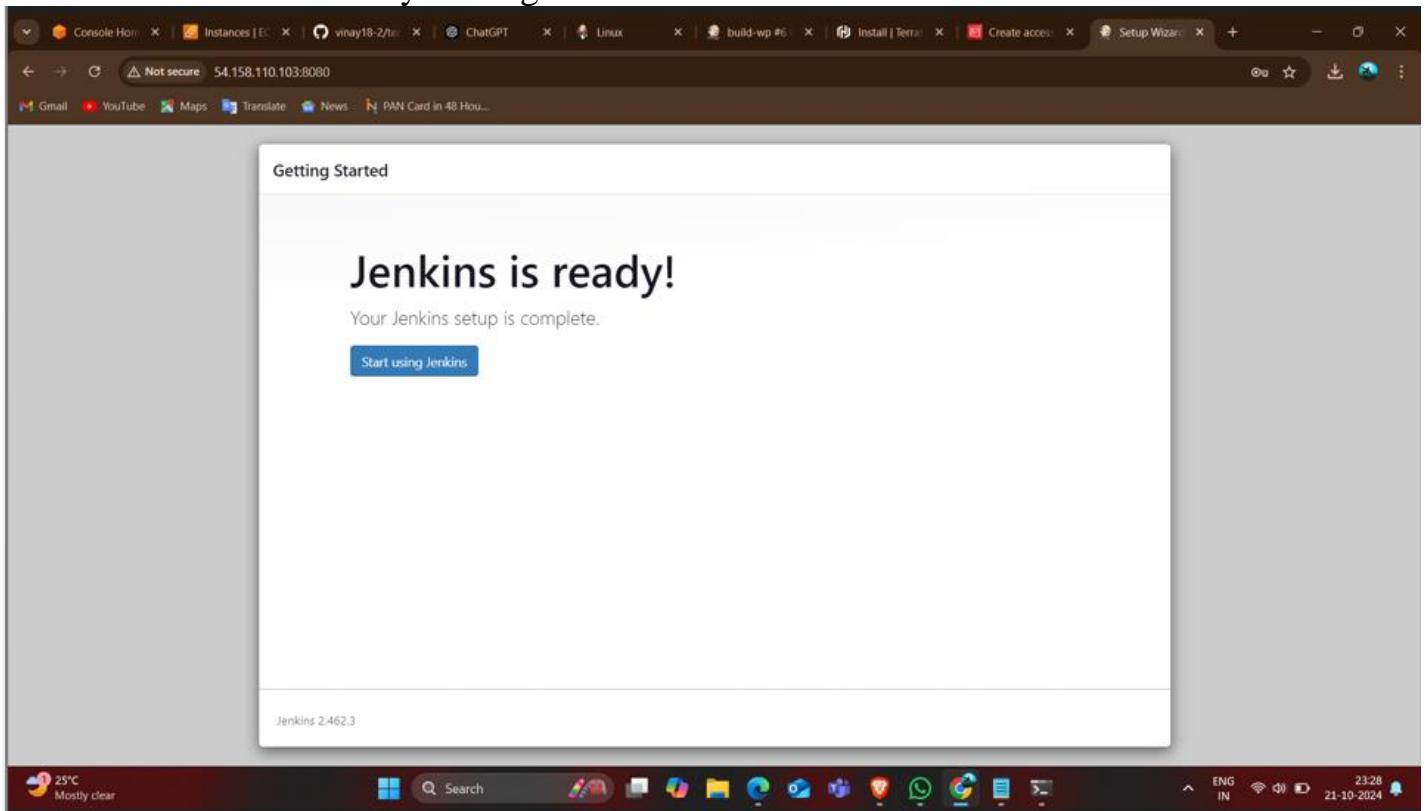
```

```

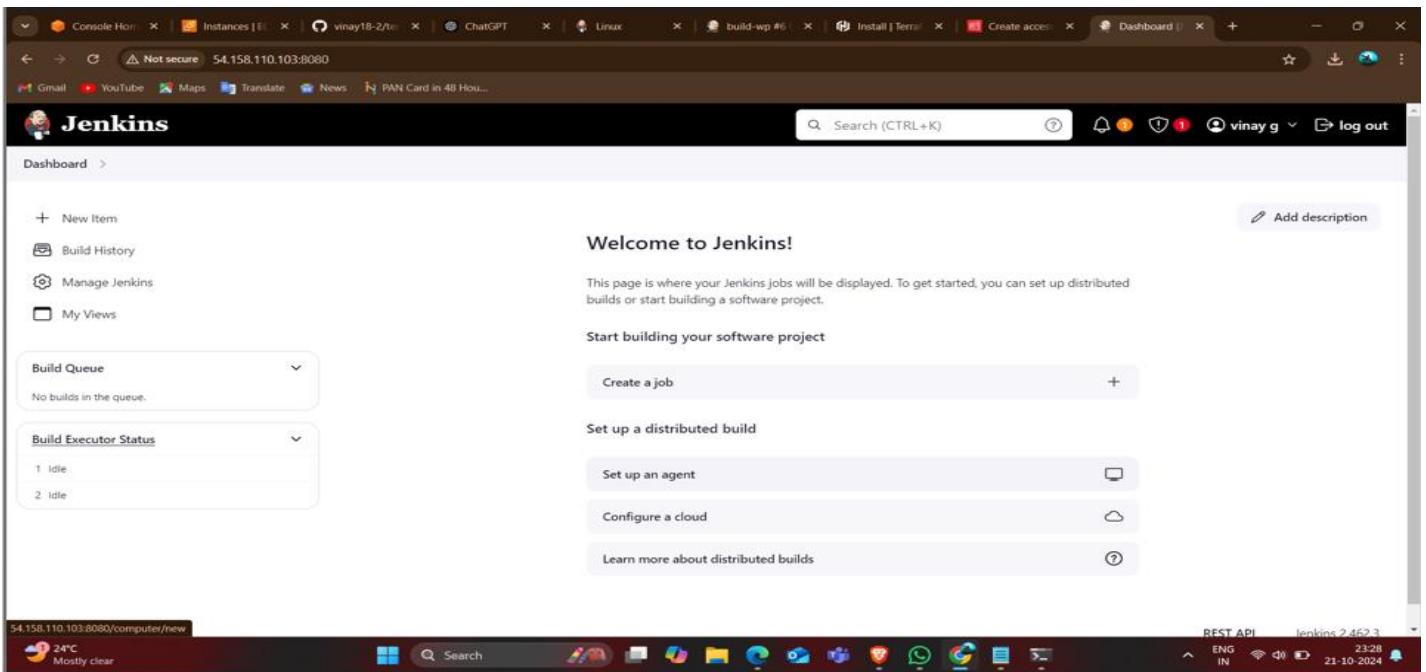
ec2-user@ip-172-31-93-11:~ + 
Total download size: 89 M
Installed size: 89 M
Downloading packages:
jenkins-2.462.3-1.1.noarch.rpm
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : jenkins-2.462.3-1.1.noarch
    Verifying : jenkins-2.462.3-1.1.noarch
Installed:
  jenkins.noarch 0:2.462.3-1.1
Complete!
[ec2-user@ip-172-31-93-11 ~]$ sudo systemctl start jenkins
[ec2-user@ip-172-31-93-11 ~]$ sudo systemctl enable jenkins
Created symlink from /etc/systemd/system/multi-user.target.wants/jenkins.service to /usr/lib/systemd/system/jenkins.service.
[ec2-user@ip-172-31-93-11 ~]$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
  Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; vendor preset: disabled)
  Active: active (running) since Mon 2024-10-21 17:56:40 UTC; 22s ago
    Main PID: 4613 (java)
     CPU: 0.26% 1/1
      Tasks: 1 (pid: 4613; code=running; user=ec2-user)
     Group: /system.slice/jenkins.service
        ▾ 4613 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=%C/jenkins/war --...
Oct 21 17:56:37 ip-172-31-93-11.ec2.internal jenkins[4613]: 2939d91756fb4e2f8389b249ca2e777d
Oct 21 17:56:37 ip-172-31-93-11.ec2.internal jenkins[4613]: This may also be found at: /var/lib/jenkins/secrets/in...ord
Oct 21 17:56:37 ip-172-31-93-11.ec2.internal jenkins[4613]: ****
Oct 21 17:56:37 ip-172-31-93-11.ec2.internal jenkins[4613]: ****
Oct 21 17:56:37 ip-172-31-93-11.ec2.internal jenkins[4613]: ****
Oct 21 17:56:40 ip-172-31-93-11.ec2.internal jenkins[4613]: 2024-10-21 17:56:40.213+0000 [id=34]           INFO    ...ion
Oct 21 17:56:40 ip-172-31-93-11.ec2.internal jenkins[4613]: 2024-10-21 17:56:40.237+0000 [id=24]           INFO    ...ing
Oct 21 17:56:40 ip-172-31-93-11.ec2.internal systemd[1]: Started Jenkins Continuous Integration Server.
Oct 21 17:56:40 ip-172-31-93-11.ec2.internal jenkins[4613]: 2024-10-21 17:56:40.319+0000 [id=50]           INFO    ...ler
Oct 21 17:56:40 ip-172-31-93-11.ec2.internal jenkins[4613]: 2024-10-21 17:56:40.319+0000 [id=50]           INFO    ... #1
Hint: Some lines were ellipsized, use -l to show in full.
[ec2-user@ip-172-31-93-11 ~]$ sudo visudo
"/etc/sudoers.tmp" 120L, 4373B written
[ec2-user@ip-172-31-93-11 ~]$ | 23:44 21-10-2024

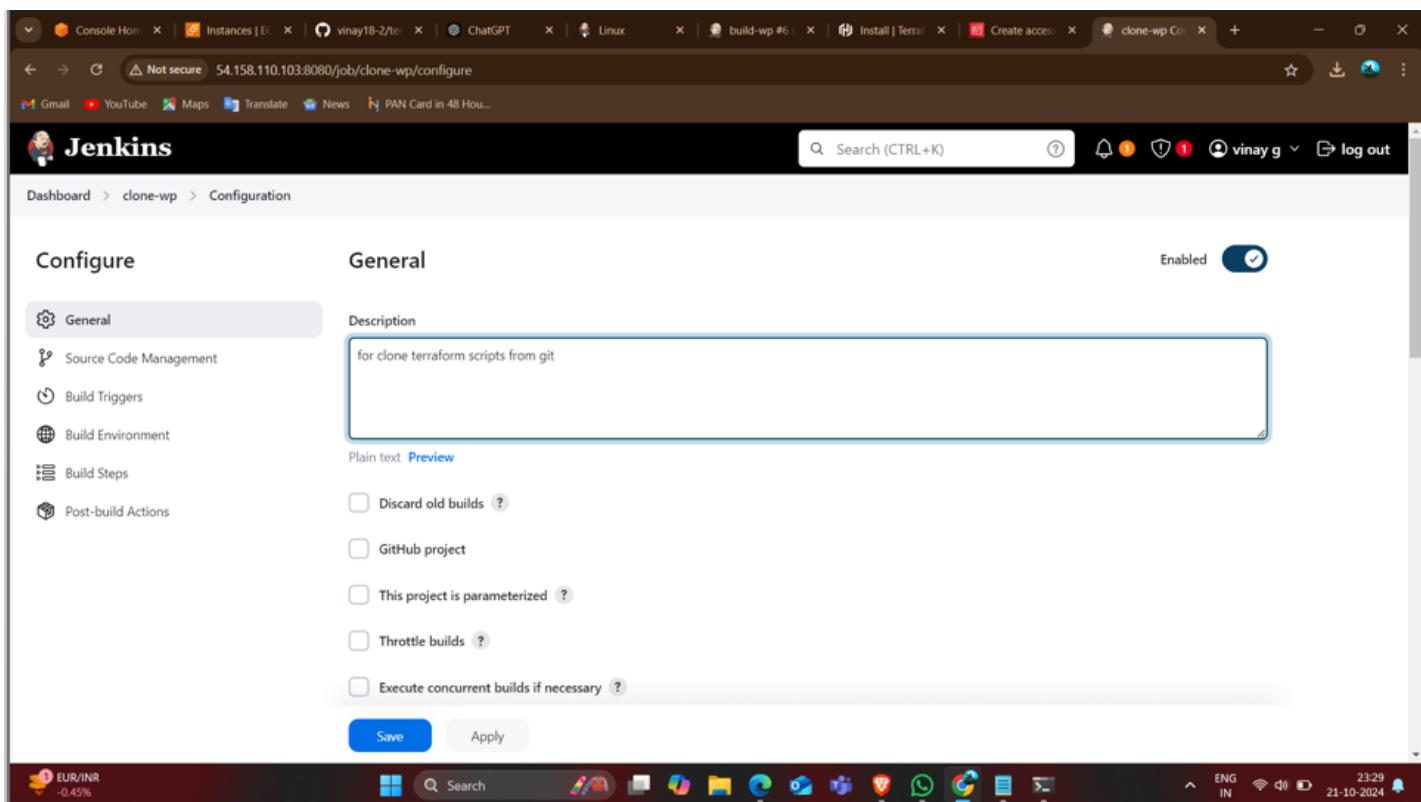
```

- Here Jenkins is ready and login to the Jenkins dashboard.



- Create the clone job to clone the repo from the git and follow the below steps to complete the clone job.





The screenshot shows the Jenkins General configuration page for a job named "clone-wp". The "Enabled" checkbox is checked. The "Description" field contains the text "for clone terraform scripts from git". Under "Plain text", there are five checkboxes: "Discard old builds", "GitHub project", "This project is parameterized", "Throttle builds", and "Execute concurrent builds if necessary". At the bottom are "Save" and "Apply" buttons.

Configure General

Description

for clone terraform scripts from git

Plain text: [Preview](#)

Discard old builds ?

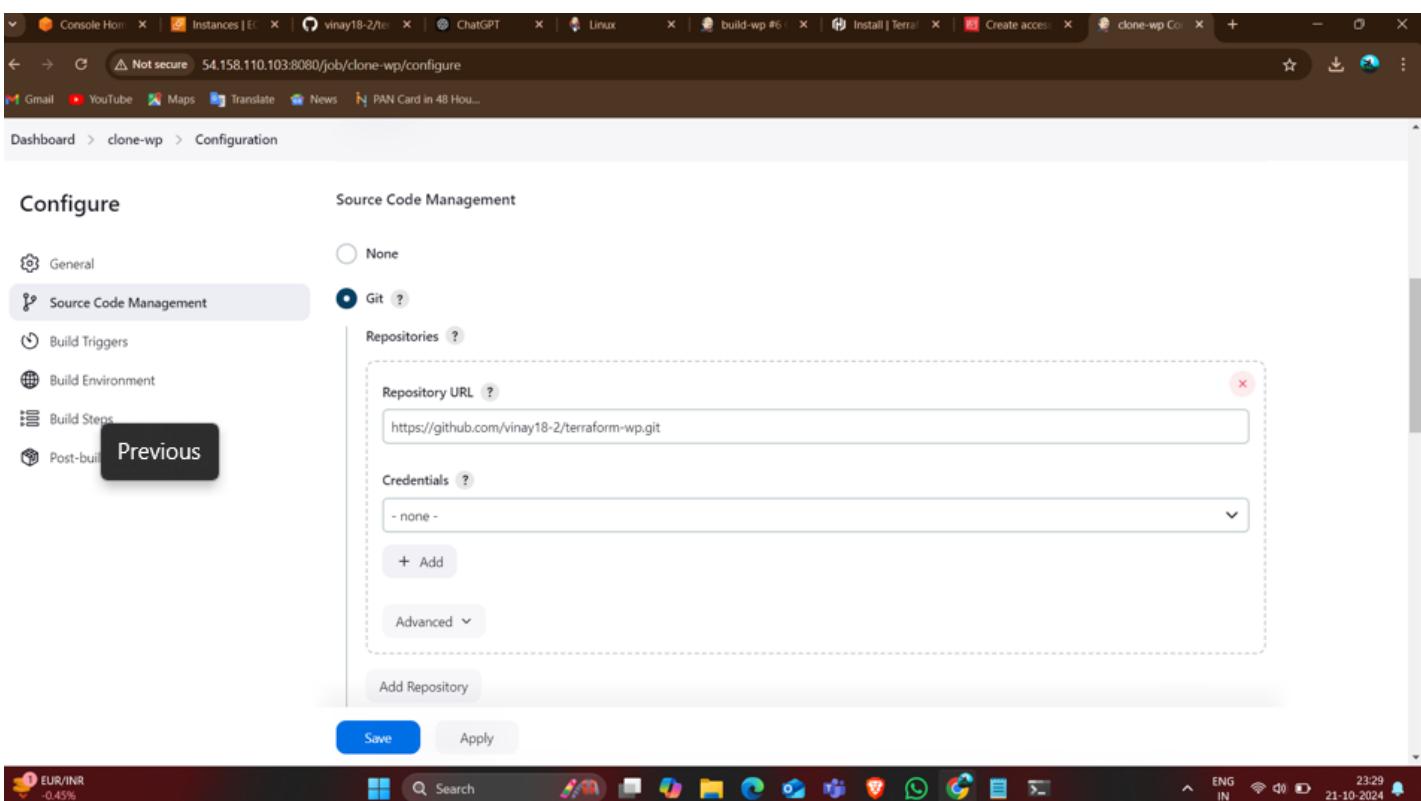
GitHub project

This project is parameterized ?

Throttle builds ?

Execute concurrent builds if necessary ?

Save Apply



The screenshot shows the Jenkins Source Code Management configuration page for the same job. The "Git" option is selected. In the "Repositories" section, the "Repository URL" is set to "https://github.com/vinay18-2/terraform-wp.git". The "Credentials" dropdown is set to "- none -". There is a "+ Add" button and an "Advanced" dropdown. At the bottom are "Save" and "Apply" buttons.

Configure Source Code Management

None

Git ?

Repositories ?

Repository URL ?

https://github.com/vinay18-2/terraform-wp.git

Credentials ?

- none -

+ Add

Advanced

Add Repository

Save Apply

The screenshot shows the Jenkins configuration interface for the 'clone-wp' job. On the left, a sidebar lists various configuration sections: General, Source Code Management, Build Triggers (which is selected), Build Environment, Build Steps, and Post-build Actions. The main panel is titled 'Configure' and contains several sections:

- Branches to build**: A 'Branch Specifier' input field contains the value '/main'. Below it is an 'Add Branch' button.
- Repository browser**: A dropdown menu is set to '(Auto)'.
- Additional Behaviours**: An 'Add' button is present.

  
**Build Triggers**:

- Two checkboxes are available: 'Trigger builds remotely (e.g., from scripts)' and 'Build after other projects are built'.

At the bottom of the configuration panel are 'Save' and 'Apply' buttons.

The system tray at the bottom of the screen shows currency exchange rates (EUR/INR, JPY/INR), system status (ENG IN), and a date/time indicator (21-10-2024 23:30).

- After save and apply than to run select the build now.

The screenshot shows the Jenkins project page for 'clone-wp'. The top navigation bar includes links for 'Console', 'Instances', 'vinay18-2/tm...', 'ChatGPT', 'Linux', 'build-wp #6', 'Install | Terra...', 'Create access', and 'clone-wp Job'. The main content area displays the following information:

- Status**: Shows the project is healthy.
- Changes**: No changes are listed.
- Workspace**: A link to view the workspace.
- Build Now**: A button to trigger a manual build.
- Configure**: A link to edit the project configuration.
- Delete Project**: A link to delete the project.
- Rename**: A link to rename the project.

  
**clone-wp**  
for clone terraform scripts from git  
**Permalinks**  
  
**Build History**:

- Oct 21, 2024, 6:00 PM (Build #1)

Atom feed for all Atom feed for failures

The system tray at the bottom of the screen shows currency exchange rates (JPY/INR), system status (ENG IN), and a date/time indicator (21-10-2024 23:30). The REST API and Jenkins version (2.462.3) are also visible.

- Than install AWS credentials plugin to store the aws access keys and secret keys.

The screenshot shows the Jenkins Manage Jenkins dashboard. In the top right corner, there are three buttons: "Set up agent", "Set up cloud", and "Dismiss". Below them, a message says: "Building on the built-in node can be a security issue. You should set up distributed builds. See [the documentation](#)".

### System Configuration

- Build Queue:** No builds in the queue.
- Build Executor Status:** 1 Idle, 2 Idle.
- System:** Configure global settings and paths.
- Nodes:** Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
- Tools:** Configure tools, their locations and automatic installers.
- Clouds:** Add, remove, and configure cloud instances to provision agents on-demand.
- Plugins:** Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
- Appearance:** Configure the look and feel of Jenkins.

### Security

- Security:** Secure Jenkins; define who is allowed to access/use the system.
- Credentials:** Configure credentials.
- Credential Providers:** Configure the credential providers and types.
- Users:** Create/delete/modify users that can log in.

At the bottom of the screen, there is a taskbar with icons for JPY/INR, Search, ChatGPT, Linux, build-wp #6, Install | Terraform, Create access, Manage Jenkins, and other system status indicators like ENG IN, 21-10-2024, and 23:30.

The screenshot shows the Jenkins Manage Jenkins > Plugins page. A search bar at the top contains the text "aws cred". On the left, there is a sidebar with links: "Updates", "Available plugins" (which is selected), "Installed plugins", "Advanced settings", and "Download progress".

Install	Name	Released
<input checked="" type="checkbox"/>	AWS Credentials 231.v08a_59ff17d742	6 mo 27 days ago
	aws	
Allows storing Amazon IAM credentials within the Jenkins Credentials API. Store Amazon IAM access keys (AWSAccessKeyId and AWSSecretKey) within the Jenkins Credentials API. Also support IAM Roles and IAM MFA Token.		

At the bottom of the screen, there is a taskbar with icons for JPY/INR, Search, ChatGPT, Linux, build-wp #6, Install | Terraform, Create access, Manage Jenkins, and other system status indicators like REST API, Jenkins 2.462.3, ENG IN, 21-10-2024, and 23:31.

The screenshot shows the Jenkins Plugins page. On the left, there's a sidebar with links: Updates, Available plugins, Installed plugins, Advanced settings, and Download progress (which is selected). The main content area is titled "Download progress" and shows a table of installed Jenkins components and their download status:

Component	Status
Preparation	Success
Ionicons API	Success
Folders	Success
OWASP Markup Formatter	Success
ASM API	Success
JSON Path API	Success
Structs	Success
Pipeline: Step API	Success
Token Macro	Success
Build Timeout	Success
bouncycastle API	Success
Credentials	Success
Plain Credentials	Success
Variant	Success
SSH Credentials	Success
Credentials Binding	Success

At the bottom of the screen, there's a Windows taskbar with various icons and system status indicators.

The screenshot shows the Jenkins Dashboard. On the left, there are links for New Item, Build History, Manage Jenkins, and My Views. The main area displays a table of recent builds:

S	W	Name	Last Success	Last Failure	Last Duration
✓	☀️	clone-wp	1 min 20 sec #1	N/A	3.3 sec

Below this, there are two dropdown menus: "Build Queue" (No builds in the queue) and "Build Executor Status" (1 Idle, 2 Idle). At the bottom right, there are links for REST API and Jenkins 2.462.3. The bottom of the screen features a Windows taskbar with various icons and system status indicators.

- After completion of clone job than we shoul create new project for build job from copying of existing project i.e clone job.

New Item

Enter an item name

build-wp

Select an item type

**Freestyle project**  
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

**Pipeline**  
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

**Multi-configuration project**  
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

**Folder**

OK

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

**Folder**  
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

**Multibranch Pipeline**  
Creates a set of Pipeline projects according to detected branches in one SCM repository.

**Organization Folder**  
Creates a set of multibranch project subfolders by scanning for repositories.

If you want to create a new item from other existing, you can use this option:

Copy from

clone-wp

OK

REST API Jenkins 2.462.3

- After providing the repo link than select the Credentials.

Configure

Git

General

Source Code Management

Build Triggers

Build Environment

Build Steps

Post-build Actions

Repositories

Repository URL

`https://github.com/vinay18-2/terraform-wp.git`

Credentials

- none -

+ Add

Jenkins

Add Repository

Branches to build

Branch Specifier (blank for 'any')

Save Apply

- Add Jenkins credentials provider,than select tha kind AWS credentials. Add the access keys and secret keys.

Configure

General

Source Code Management

Build Triggers

Build Environment

Build Steps

Post-build Actions

Jenkins Credentials Provider: Jenkins

Add Credentials

Domain

Global credentials (unrestricted)

Kind

AWS Credentials

Scope

Global (Jenkins, nodes, items, all child items, etc)

ID

Description

Save Apply

The screenshot shows the AWS Lambda configuration interface for a function named 'build-wp'. The left sidebar lists 'General', 'Source Code Management', 'Build Triggers', 'Build Environment' (which is selected), 'Build Steps', and 'Post-build Actions'. The main panel is titled 'Configure' and contains a section for 'AWS access key and secret'. It includes fields for 'Access Key Variable' (set to 'AWS\_ACCESS\_KEY\_ID') and 'Secret Key Variable' (set to 'AWS\_SECRET\_ACCESS\_KEY'). A dropdown menu for 'Credentials' shows 'Specific credentials' selected, with the value 'AKIA4MTWNVJ3LGAZY4XO'. There is also an 'Add' button and a checkbox for 'Add timestamps to the Console Output'. At the bottom are 'Save' and 'Apply' buttons.

- Provide the terraform commands and installation commands in execute shell and save it and apply.

The screenshot shows the AWS Lambda configuration interface for a function named 'build-wp'. The left sidebar lists 'General', 'Source Code Management', 'Build Triggers', 'Build Environment', 'Build Steps' (selected), and 'Post-build Actions'. The main panel is titled 'Configure' and contains a 'Command' section. It displays the following Terraform commands:

```
#!/bin/bash
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
terraform init
terraform validate
terraform apply --auto-approve
```

At the bottom are 'Save' and 'Apply' buttons.

- To prevent the errors we have to edit the visudo file.

```

## 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
jenkins     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
-- INSERT --

```

111, 46 Bot

- Observe the console output after completion of build. we can get to know the build is executed or get any errors and where the errors have been occurred.

**Console Output**

```

Started by user vinay g
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/build-wp
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/build-wp/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/vinay18-2/terraform-wp.git # timeout=10
Fetching upstream changes from https://github.com/vinay18-2/terraform-wp.git
> git --version # timeout=10
> git --version # 'git version 2.40.1'
> git fetch --tags --force --progress -- https://github.com/vinay18-2/terraform-wp.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 173467b013f2abe8a27468dd002dfb6e3ffff4dc (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f 173467b013f2abe8a27468dd002dfb6e3ffff4dc # timeout=10
Commit message: "Update subnets.tf"
> git rev-list --no-walk f25691ff8992c0565b5e3f6cd1aa7ca7e06462d0 # timeout=10
[build-wp] $ /bin/bash /tmp/jenkins4428354610923816371.sh
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Package yum-utils-1.1.31-46.amzn2.0.1.noarch already installed and latest version
Package 2:shadow-utils-4.1.5.1-24.amzn2.0.3.x86_64 already installed and latest version

```

- After executed than observe in the aws console that the services we are provided in the git has been cloned and created with the help of Jenkins or not.
- Here we can observe that the services has been created and we can observe the output by copying the ip address of created instance.

**Instances (1/3) Info**

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
jenkins-8	i-0e24ffdf535b0d55	Stopped	t2.medium	-	View alarms +	us-east-1d	-
<b>tf wordpress</b>	<b>i-0923382fbdc62d968</b>	<b>Running</b>	<b>t2.micro</b>	<b>Initializing</b>	<b>View alarms +</b>	<b>us-east-1a</b>	-
wp-8	i-0db225051cc7a2f33	Running	t2.medium	2/2 checks passed	View alarms +	us-east-1d	ec2-54-

**i-0923382fbdc62d968 (tf wordpress)**

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

**Instance summary**

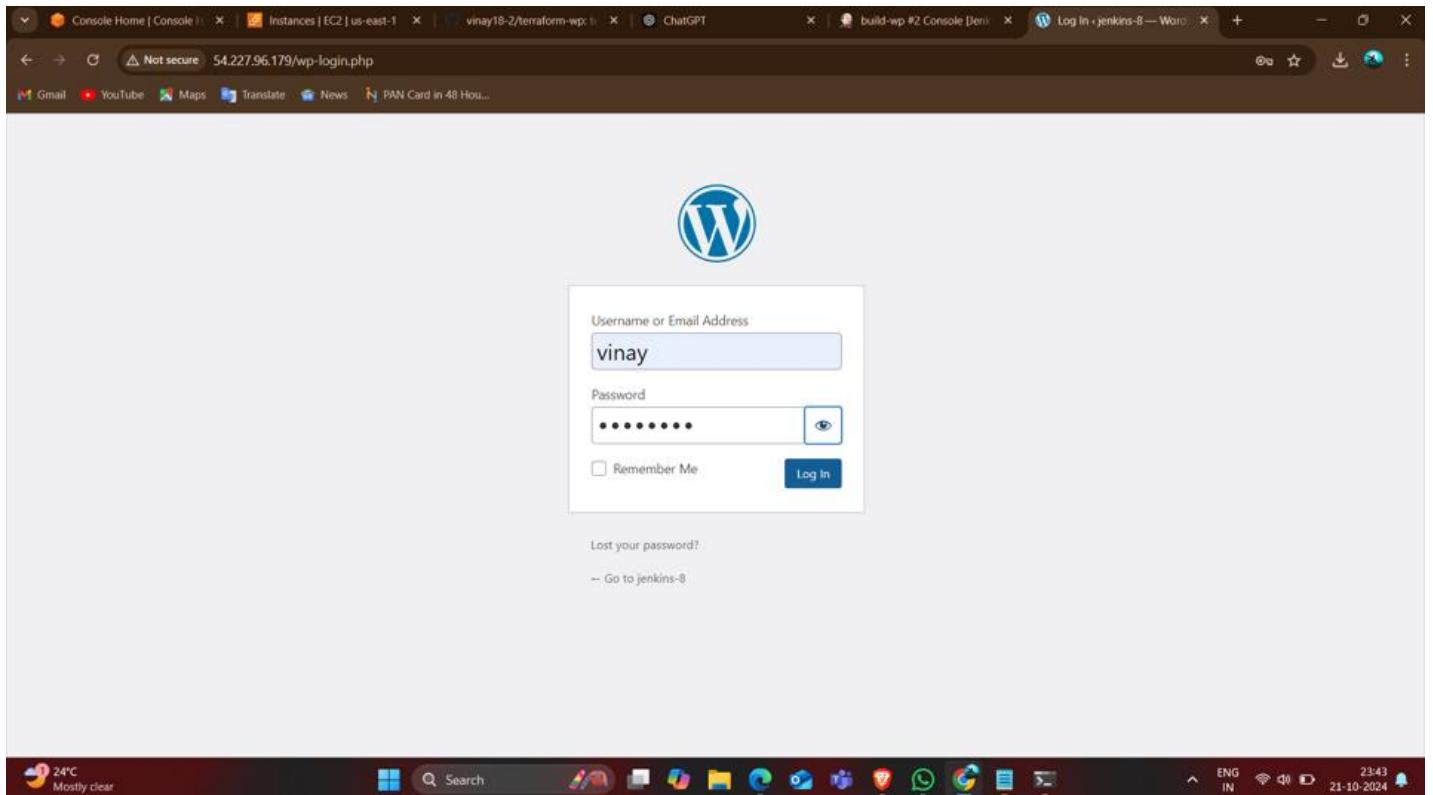
Instance ID i-0923382fbdc62d968	Public IPv4 address 54.227.96.179   <a href="#">open address</a>	Private IPv4 addresses 10.0.1.68
IPv6 address -	Instance state <b>Running</b>	Public IPv4 DNS -
Hostname type IP name: ip-10-0-1-68.ec2.internal	Private IP DNS name (IPv4 only) ip-10-0-1-68.ec2.internal	Elastic IP addresses -
Answer private resource DNS name -	Instance type t2.micro	

- Here the output.

English (United States)

Afrikaans  
አማርኛ  
Aragonés  
العربية  
العربية المغربية  
অসমীয়া  
گۈزىنىڭ ئۇرپاچىلىق  
Azərbaycan dili  
Беларуская мова  
Български  
বাংলা  
ଓଡ଼ିଆ  
Bosanski  
Català  
Cebuano  
Čeština  
Cymraeg  
Dansk  
Deutsch (Schweiz)  
Deutsch

Continue



Welcome to WordPress!

[Learn more about the 6.6.2 version.](#)

**Author rich content with blocks and patterns**  
Block patterns are pre-configured block layouts. Use them to get inspired or create new pages in a flash.  
[Add a new page](#)

**Customize your entire site with block themes**  
Design everything on your site — from the header down to the footer, all using blocks and patterns.  
[Open site editor](#)

**Switch up your site's look & feel with Styles**  
Tweak your site, or give it a whole new look! Get creative — how about a new color palette or font?  
[Edit styles](#)

**Site Health Status**  
Site health checks will automatically run periodically to gather information about

**Quick Draft**  
Title

•

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.

- For this method the procedure is the same as the above method.
- Launching the instance.
- Connect to the terminal.
- Update it.
- Install git.
- Install Jenkins and start Jenkins.
- Enable Jenkins.
- Check the status and login to the Jenkins.
- Create the clone job and build job.
- Than install the build pipeline plugin.
- Than connect bothe jobs through pipeline.

The screenshot shows the Jenkins dashboard with two active builds listed:

S	W	Name	Last Success	Last Failure	Last Duration
✓	Cloud	build-wp	5 min 51 sec #2	9 min 19 sec #1	44 sec
✓	Sun	clone-wp	6 min 30 sec #2	N/A	0.16 sec

Build Queue: No builds in the queue.

Build Executor Status: 1 Idle, 2 Idle.

REST API Jenkins 2.46.2

- Create a new pipeline view.

The screenshot shows the 'New view' creation page in Jenkins. The 'Name' field is set to 'wordpress-ppl'. The 'Type' section has 'Build Pipeline View' selected, with a description: 'Shows the jobs in a build pipeline view. The complete pipeline of jobs that a version propagates through are shown as a row in the view.' Other options like 'List View' and 'My View' are also shown.

New view

Name: wordpress-ppl

Type:  Build Pipeline View  
Shows the jobs in a build pipeline view. The complete pipeline of jobs that a version propagates through are shown as a row in the view.

List View  
Shows items in a simple list format. You can choose which jobs are to be displayed in which view.

My View  
This view automatically displays all the jobs that the current user has an access to.

Create

REST API Jenkins 2.46.2

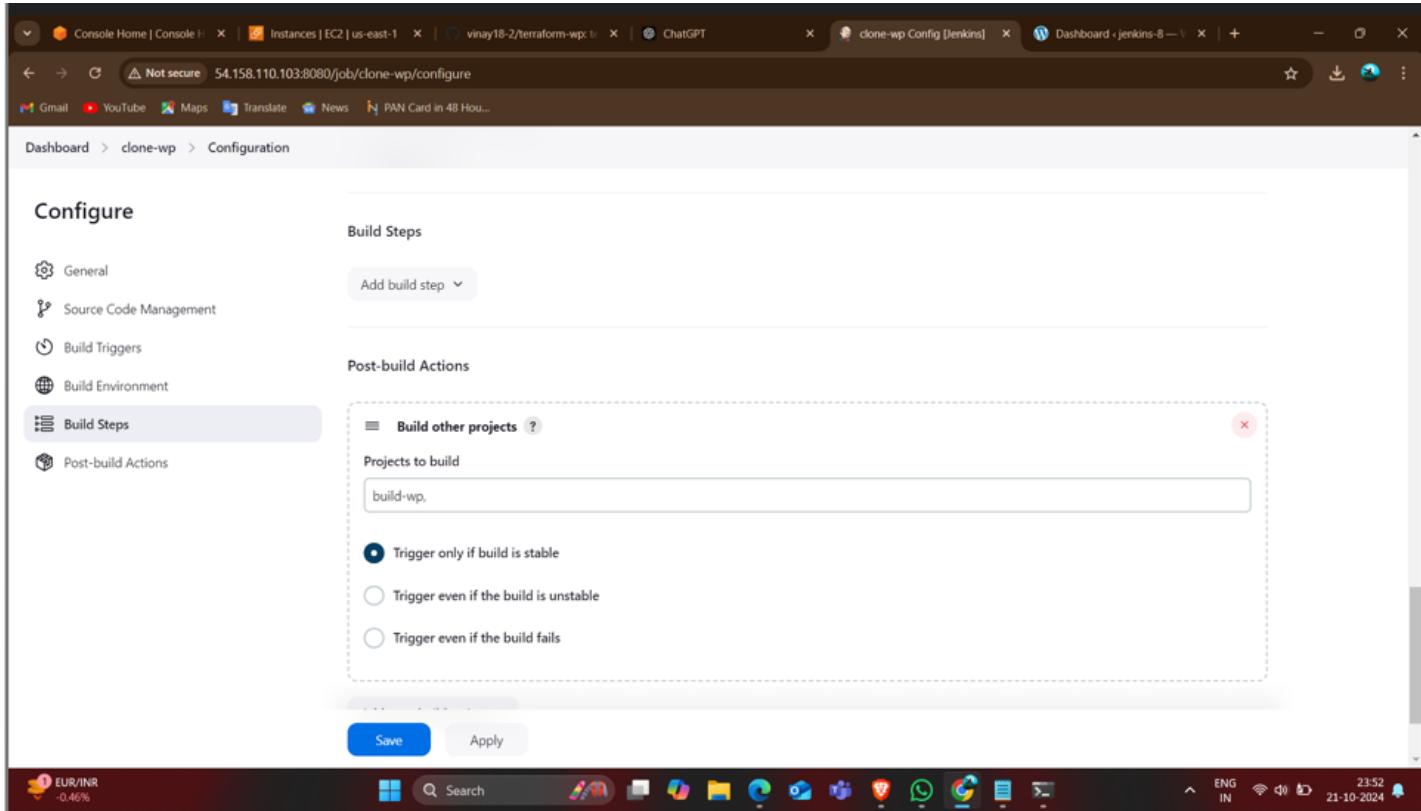
- Configure it as shown in below pictures.

The screenshot shows the Jenkins Pipeline configuration page for a view named 'wordpress-ppl'. The 'Upstream / downstream config' section is open, displaying the 'Select Initial Job' dropdown which contains 'clone-wp'. Below this, the 'Trigger Options' section is visible, featuring a 'Build Cards' dropdown set to 'Standard build card', a 'Use the default build cards' checkbox, and two radio buttons for 'Restrict triggers to most recent successful builds': 'Yes' (unchecked) and 'No' (checked). Further down are options for 'Always allow manual trigger on pipeline steps' (radio button 'Yes' unchecked) and 'OK' and 'Apply' buttons at the bottom.

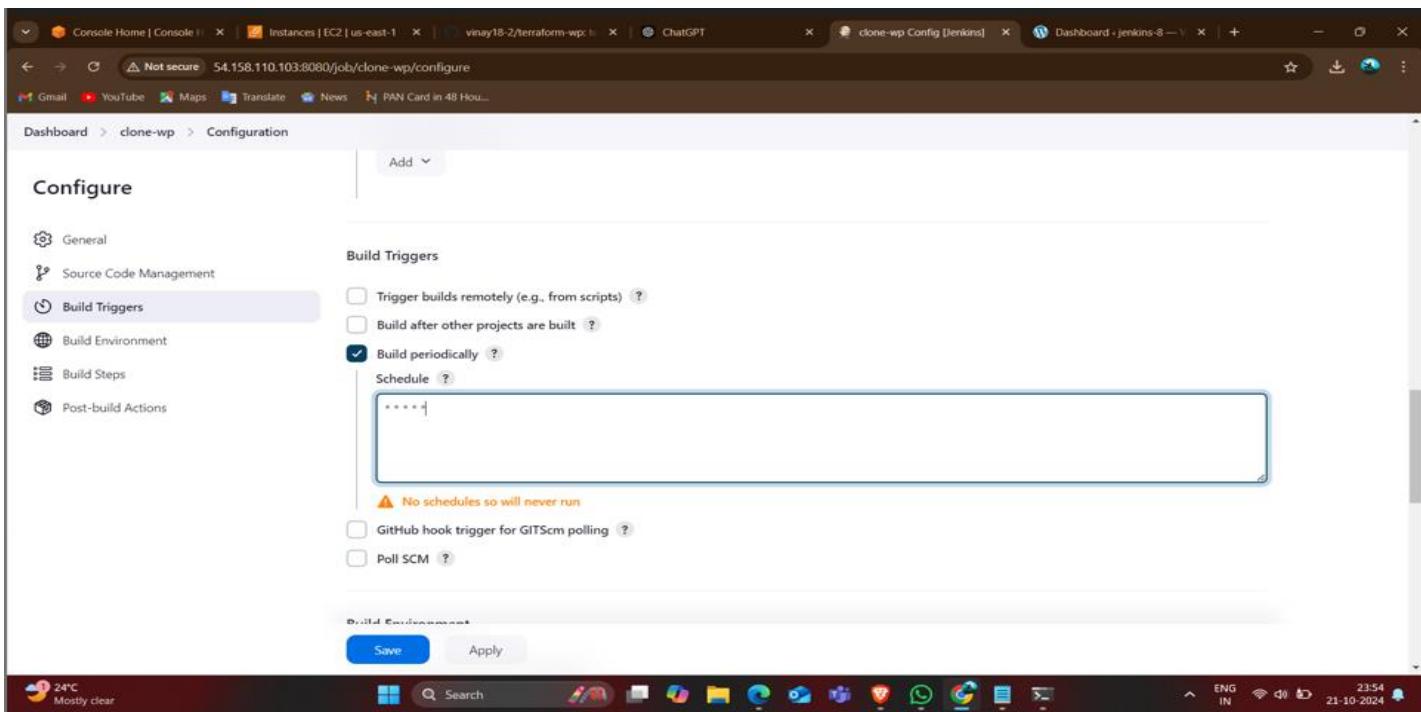
  

The screenshot shows the Jenkins Pipeline configuration page for a view named 'wordpress-ppl'. The 'Display Options' section is open, displaying the 'No Of Displayed Builds' dropdown set to '2', the 'Row Headers' dropdown set to 'Just the pipeline number', and the 'Column Headers' dropdown set to 'No header'. Below these are checkboxes for 'Show just the build pipeline number' and 'Do not show any column headers'. At the bottom are 'OK' and 'Apply' buttons.

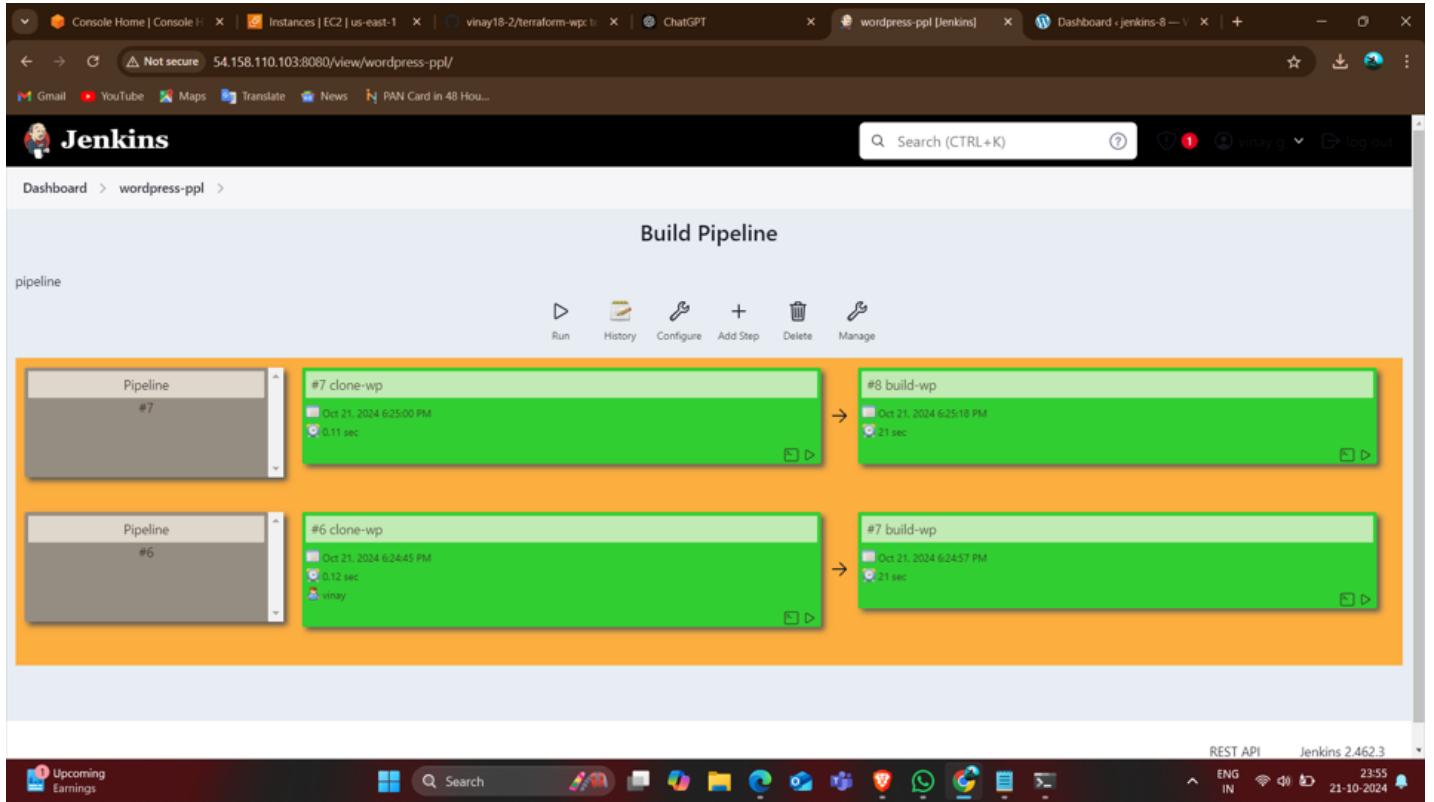
- Configure the clone job by adding post build action as a build job.



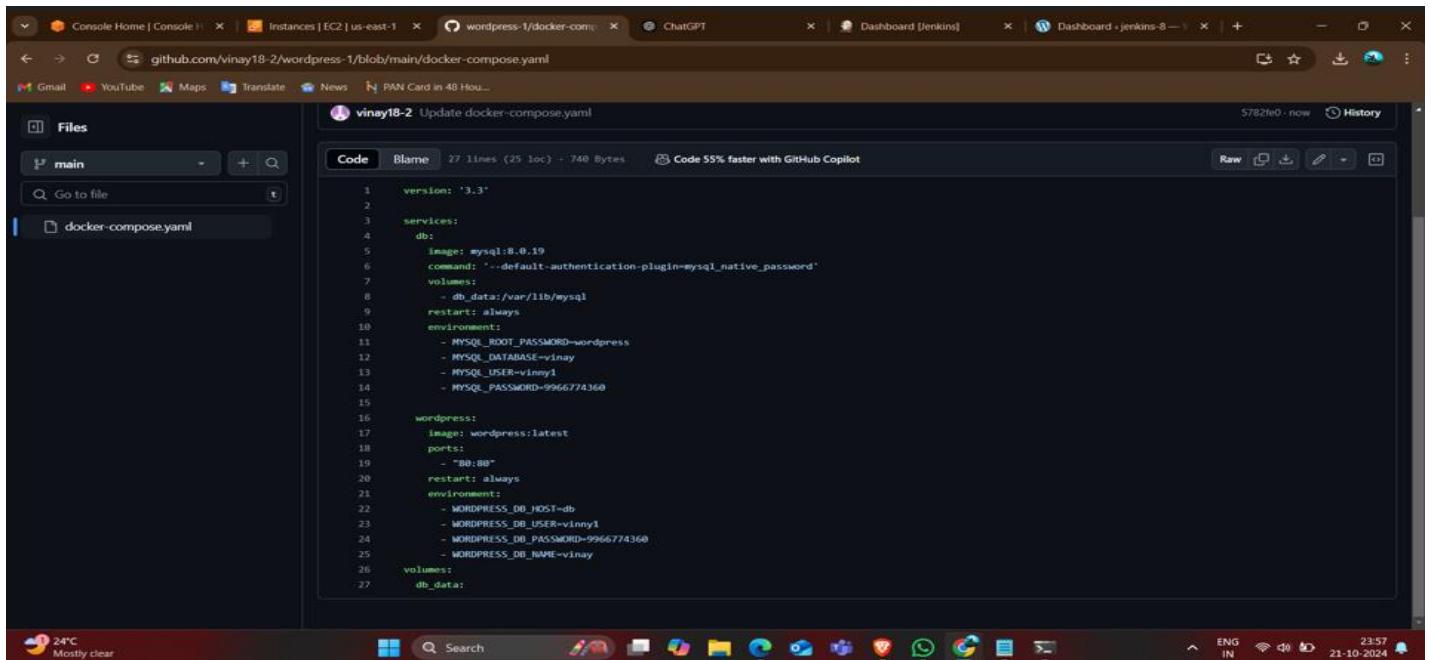
- Provide the schedule for build periodically.



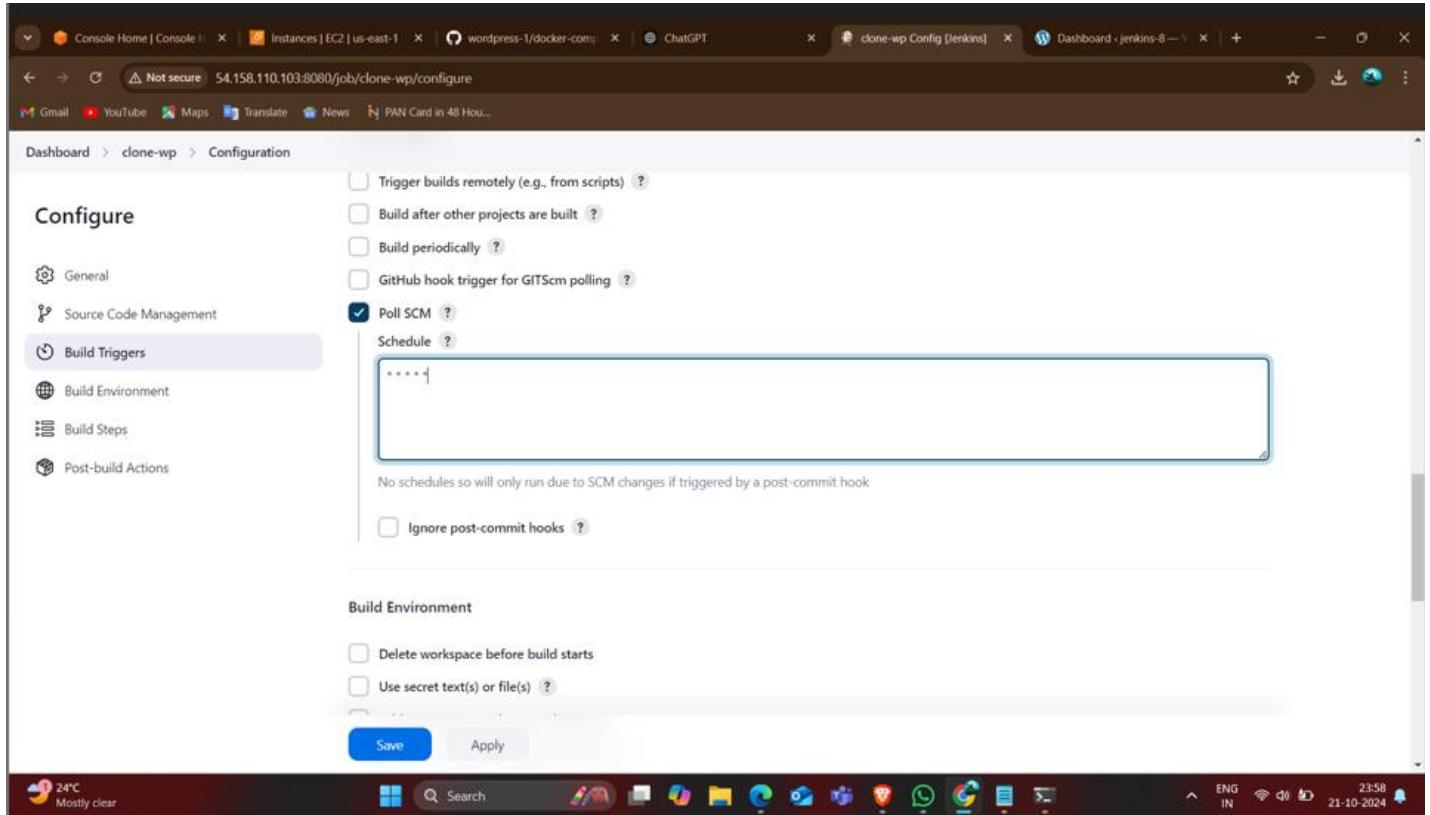
- Output for build periodically.



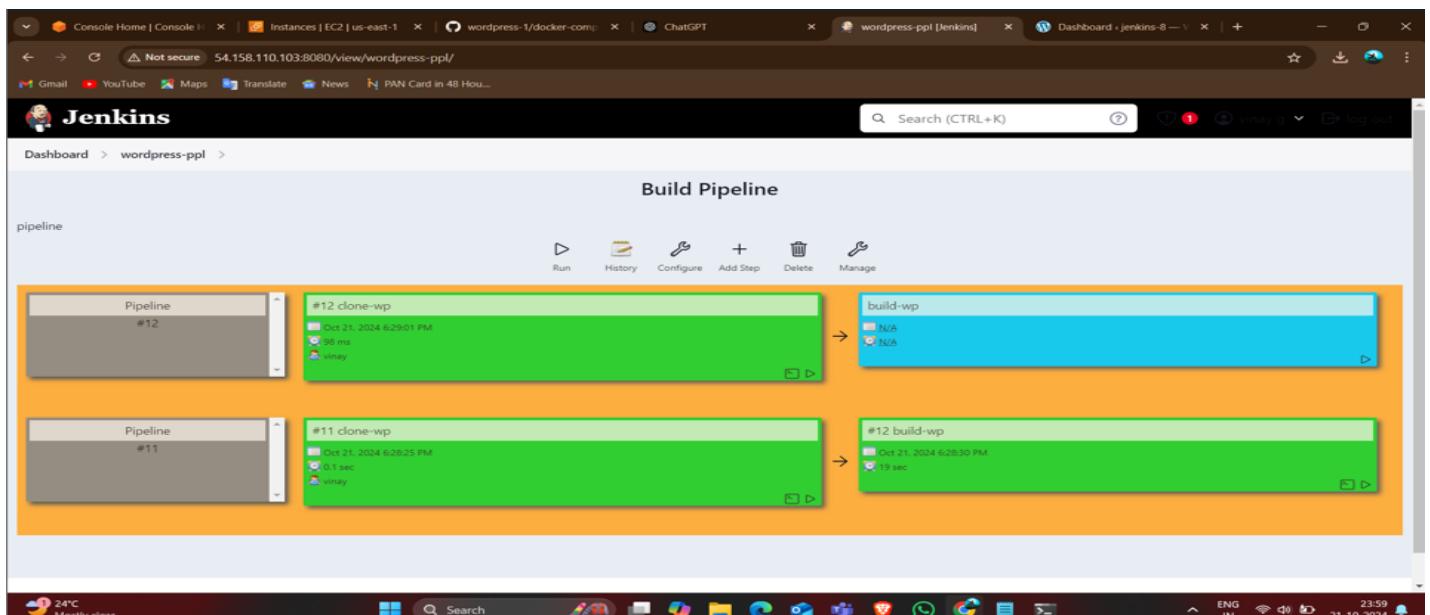
- After that change the script or yaml code and build with the poll SCM and observe.



- Provide the schedule for poll scm and save it and apply it. Then build now.



- Output.



The screenshot shows the Jenkins Build Pipeline interface. At the top, there is a navigation bar with tabs for 'Console Home', 'Instances', 'wordpress-1/docker-comp', 'ChatGPT', 'wordpress-ppl [Jenkins]', and 'Dashboard'. Below the navigation bar, the Jenkins logo and search bar are visible. The main area is titled 'Build Pipeline' and contains two parallel pipeline sections, each with a header and several stages.

**Pipeline #12:**

- Header: Pipeline #12
- Stage 1: #12 clone-wp (Status: Oct 21, 2024 6:29:01 PM, Duration: 98 ms, User: vinay)
- Stage 2: #13 build-wp (Status: Oct 21, 2024 6:29:10 PM, Duration: 20 sec, User: vinay)

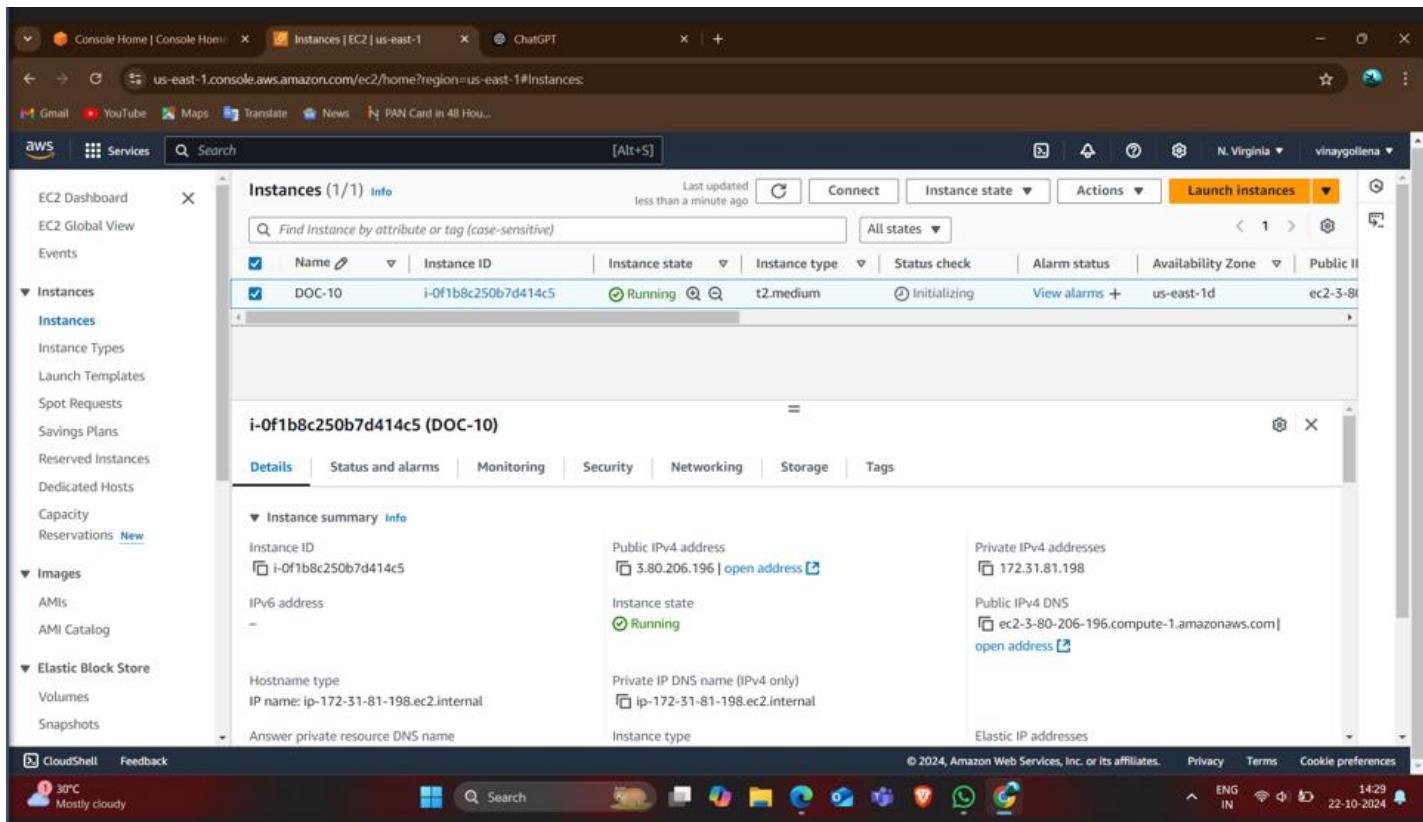
**Pipeline #11:**

- Header: Pipeline #11
- Stage 1: #11 clone-wp (Status: Oct 21, 2024 6:28:25 PM, Duration: 0.1 sec, User: vinay)
- Stage 2: #12 build-wp (Status: Oct 21, 2024 6:28:30 PM, Duration: 19 sec, User: vinay)

At the bottom of the screen, there is a taskbar with various icons and system status information, including a weather forecast for 24°C Mostly clear, a date of 21-10-2024, and a time of 23:59.

# Deploy WordPress web application by using k8's (Declarative manifest method) with the help of docker hub images

- Launch the 2 EC2 instances.
- One for docker and another for kubernetes.
- Create the instance for Kuburnetes.



The screenshot shows the AWS EC2 Instances page. The left sidebar is collapsed. The main area displays a table of instances. One instance is listed:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
DOC-10	i-0f1b8c250b7d414c5	Running	t2.medium	Initializing	View alarms +	us-east-1d	ec2-3-80-206-196.compute-1.amazonaws.com

Below the table, the instance details for **i-0f1b8c250b7d414c5 (DOC-10)** are shown. The **Details** tab is selected. Key information includes:

- Instance ID: i-0f1b8c250b7d414c5
- Public IPv4 address: 3.80.206.196
- Private IPv4 address: 172.31.81.198
- Public IPv4 DNS: ec2-3-80-206-196.compute-1.amazonaws.com
- Instance state: Running
- Hostname type: IP name: ip-172-31-81-198.ec2.internal
- Private IP DNS name (IPv4 only): ip-172-31-81-198.ec2.internal
- Answer private resource DNS name: ip-172-31-81-198.ec2.internal
- Instance type: t2.medium
- Elastic IP addresses: None

- Connect it to the terminal.
- Update it.

The screenshot shows a Windows desktop environment. At the top, there are two open terminal windows. The left window is titled 'ec2-user@ip-172-31-81-198:' and the right window is titled 'ec2-user@ip-172-31-89-99:'. Both windows show command-line output related to AWS setup. Below the windows is a taskbar with various icons, including a weather icon showing '29°C Partly sunny', a search bar, and system status indicators like battery level and network connection. The system tray at the bottom right shows the date and time as '22-10-2024 16:21'.

```
Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user>cd downloads

C:\Users\user\Downloads>ssh -i "vinay.pem" ec2-user@ec2-3-80-206-196.compute-1.amazonaws.com
The authenticity of host 'ec2-3-80-206-196.compute-1.amazonaws.com (3.80.206.196)' can't be established.
ED25519 key fingerprint is SHA256:bhBfCn25EYpFPHEkgljaWG1DjhHEFsv4/AOXfzWGLCQ.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added 'ec2-3-80-206-196.compute-1.amazonaws.com' (ED25519) to the list of known hosts.

#_
~\_ ##### 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-172-31-81-198 ~]$ sudo yum update -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core
| 3.6 kB 00:00:00
No packages marked for update
[ec2-user@ip-172-31-81-198 ~]$
[ec2-user@ip-172-31-81-198 ~]$
[ec2-user@ip-172-31-81-198 ~]$
[ec2-user@ip-172-31-81-198 ~]$ sudo yum update -y

Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core
| 3.6 kB 0
0:00:00

29°C
Partly sunny
Search
ENG IN
16:21
22-10-2024
```

- Check the version of aws.
- Install the kubernetes from the google.
- Install kubectl by using command “sudo yum install kubectl -y”
- Install kops

```
curl -Lo kops https://github.com/kubernetes/kops/releases/download/$(curl -s https://api.github.com/repos/kubernetes/kops/releases/latest | grep tag_name | cut -d '"' -f 4)/kops-linux-amd64
chmod +x kops
sudo mv kops /usr/local/bin/kops
```

```

[ec2-user@ip-172-31-81-198 ~]$ No packages marked for update
[ec2-user@ip-172-31-81-198 ~]$ aws --version
aws-cli/1.18.147 Python/2.7.18 Linux/5.10.226-214.880.amzn2.x86_64 botocore/1.18.6
[ec2-user@ip-172-31-81-198 ~]$ kubectl
-bash: kubectl: command not found
[ec2-user@ip-172-31-81-198 ~]$ cat <>EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
> [kubernetes]
> name=Kubernetes
> baseurl=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/
> enabled=1
> gpgcheck=1
> gpgkey=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/repo/repomd.xml.key
> EOF
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/repo/repomd.xml.key
[ec2-user@ip-172-31-81-198 ~]$ sudo yum install -y kubectl
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core
0:00:00
kubernetes
0:00:00
kubernetes/primary
0:00:00
kubernetes
50/50

29°C Partly sunny  Search ENG IN 16:21 22-10-2024
```

- Configure the AWS and provide the access key and secret key
- Create the s3 bucket by using “aws s3 mb s3://bucket name(vinay-k8)”
- Export the bucket to kops and generate ssh key.

```

[ec2-user@ip-172-31-81-198 ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ec2-user/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ec2-user/.ssh/id_rsa.
Your public key has been saved in /home/ec2-user/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:Ohv4TkjNFsnMJ1uvUle0Gd++skJHfD6/Yj/W8ShZGpo ec2-user@ip-172-31-81-198.ec2.internal
The key's randomart image is:
+---[RSA 2048]---+
| +... .0
| * .0 ..=
| o= . .+ ..
| ..0. 0 0 0
| ..So .. o .
| 0.0. ....+
| . *. .o.* o*
| o + E.+o+o+
| .+ oo+oo |
+---[SHA256]---+
[ec2-user@ip-172-31-81-198 ~]$ |
```

30°C Mostly cloudy Search ENG IN 15:01 22-10-2024

```

[ec2-user@ip-172-31-81-198 ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ec2-user/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ec2-user/.ssh/id_rsa.
Your public key has been saved in /home/ec2-user/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:Ohv4TkknFsnMJ1uvUle0Gd++skJHFD6/Yj/W8ShZGpo ec2-user@ip-172-31-81-198.ec2.internal
The key's randomart image is:
+---[RSA 2048]---+
| +.. .o |
| *.o .= . |
| o= . .+ . . |
| ..o. o o o |
| ..So . o . |
| o.o. ....+ |
| . *. .o.* o* |
| o + E.+o+o+ |
| .+ oo+oo |
+---[SHA256]---+
[ec2-user@ip-172-31-81-198 ~]$ kops create cluster vinay-clu --state

```

30°C Mostly cloudy Search ENG IN 15:02 22-10-2024

- Create the cluster.

```

The key fingerprint is:
SHA256:Ohv4TkknFsnMJ1uvUle0Gd++skJHFD6/Yj/W8ShZGpo ec2-user@ip-172-31-81-198.ec2.internal
The key's randomart image is:
+---[RSA 2048]---+
| +.. .o |
| *.o .= . |
| o= . .+ . . |
| ..o. o o o |
| ..So . o . |
| o.o. ....+ |
| . *. .o.* o* |
| o + E.+o+o+ |
[ec2-user@ip-172-31-81-198 ~]$ kops create cluster vinay-clu --state s3://vinay-k8 --zones us-east-1a,us-east-1b --node-count 2 --yes
I1022 09:35:09.071649 32403 new_cluster.go:1454] Cloud Provider ID: "aws"
I1022 09:35:09.893929 32403 subnets.go:224] Assigned CIDR 172.20.0.0/17 to subnet us-east-1a
I1022 09:35:09.893953 32403 subnets.go:224] Assigned CIDR 172.20.128.0/17 to subnet us-east-1b
Error: objectMeta.name: Invalid value: "vinay-clu": Cluster Name must be a fully-qualified DNS name (e.g. --name=mycluster.myzone.com)
[ec2-user@ip-172-31-81-198 ~]$ kops create cluster vinayclu --state s3://vinay-k8 --zones us-east-1a,us-east-1b --node-count 2 --yes
I1022 09:35:41.383434 32409 new_cluster.go:1454] Cloud Provider ID: "aws"
I1022 09:35:42.156862 32409 subnets.go:224] Assigned CIDR 172.20.0.0/17 to subnet us-east-1a
I1022 09:35:42.156889 32409 subnets.go:224] Assigned CIDR 172.20.128.0/17 to subnet us-east-1b
Error: objectMeta.name: Invalid value: "vinayclu": Cluster Name must be a fully-qualified DNS name (e.g. --name=mycluster.myzone.com)
[ec2-user@ip-172-31-81-198 ~]$ kops create cluster vinaycluster.local --state s3://vinay-k8 --zones us-east-1a,us-east-1b --node-count 2 --yes

```

30°C Mostly cloudy Search ENG IN 15:06 22-10-2024

- Creating the 2<sup>nd</sup> instance for docker.

The screenshot shows the AWS EC2 Instances page. The left sidebar is collapsed. The main area displays a table of instances:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
doc10.1	i-08cbfcde06df90dc0	Running	t2.medium	2/2 checks passed	View alarms	us-east-1d	ec2-44-
control-plane-...	i-0f2c1936922ca28ff	Running	t3.medium	3/3 checks passed	View alarms	us-east-1a	ec2-54-
nodes-us-east...	i-0800897bc9a99bfff	Running	t3.medium	3/3 checks passed	View alarms	us-east-1a	ec2-3-8-
nodes-us-east...	i-0b75d7a74f6263b1e	Running	t3.medium	3/3 checks passed	View alarms	us-east-1b	ec2-3-9-
DOC-10	i-0f1b8c250b7d414c5	Running	t2.medium	2/2 checks passed	View alarms	us-east-1d	ec2-3-8t

Details for instance i-08cbfcde06df90dc0 (doc10.1) are shown in the details pane:

- Instance summary**:
  - Instance ID: i-08cbfcde06df90dc0
  - Public IPv4 address: 44.201.139.161 [open address]
  - Private IPv4 addresses: 172.31.89.99
  - Public IPv4 DNS: ec2-44-201-139-161.compute-1.amazonaws.com
  - Instance state: Running

- Connect to the terminal by using the ssh command, and update it.

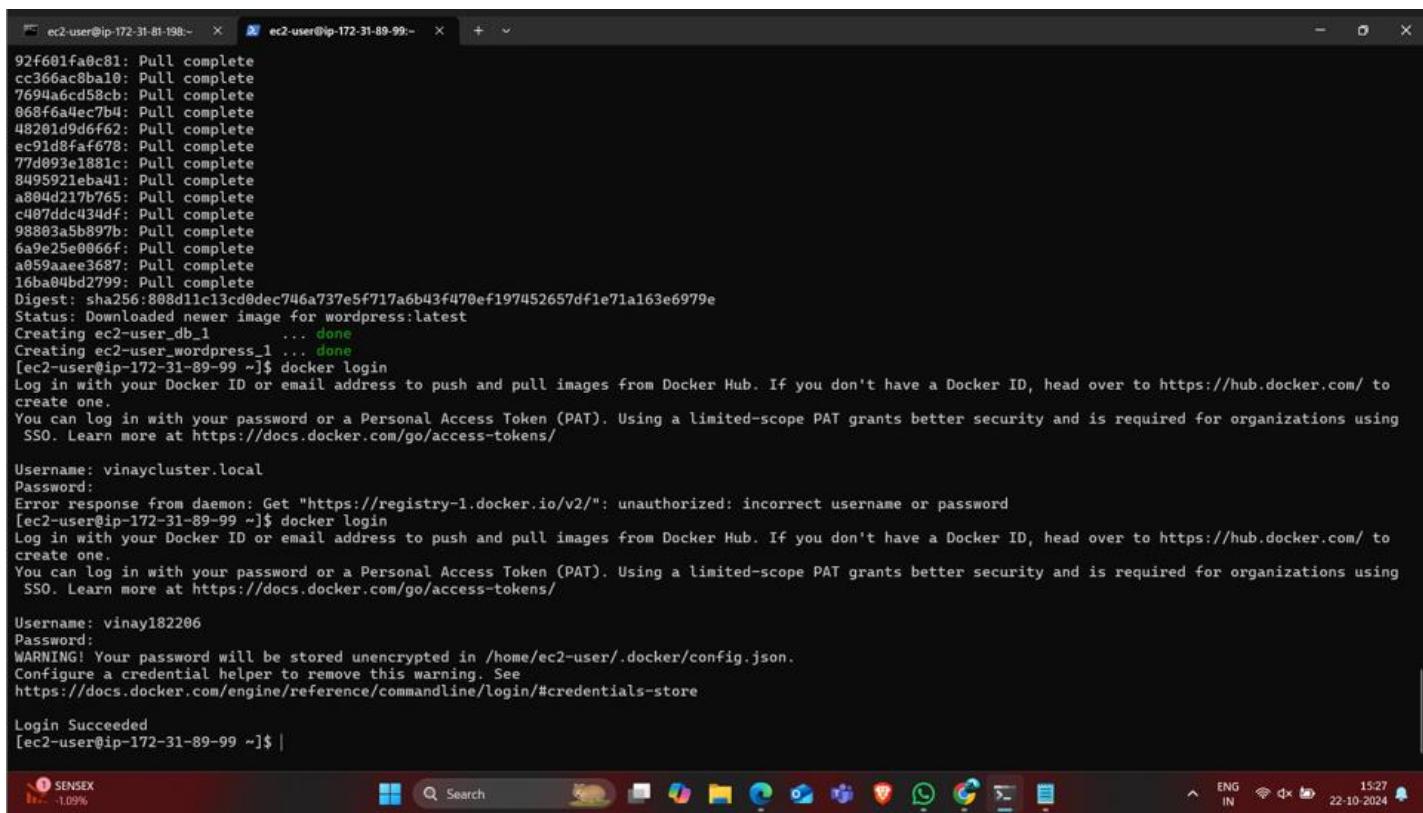
```

#!/bin/bash
sudo yum install git -y
sudo yum install docker -y
sudo systemctl start docker
sudo systemctl enable docker
sudo usermod -aG docker ec2-user
sudo chmod 660 /var/run/docker.sock
sudo curl -L https://github.com/docker/compose/releases/download/1.22.0/docker-compose-$(uname -s)-$(uname -m) -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
sudo git clone https://github.com/vinay18-2/wordpress-1.git
sudo mv wordpress-1/* /home/ec2-user/
cd /home/ec2-user/
docker-compose up -d

```

-- INSERT --

- Run the bash script by using “./script\_name.sh”
- Create an image and tag  
“docker build -t vinay182206/wordpress” or “ docker tag wordpress:latest  
vinay182206/wordpress”  
docker login  
docker images  
docker push vinay182206/wordpress
- Run the command “docker login”
- Provide the username and password.



```

[ec2-user@ip-172-31-81-198:~] ~ ec2-user@ip-172-31-89-99:~] ~ + ~
92f601fa0c81: Pull complete
c366ac8ba10: Pull complete
7694a6cd58cb: Pull complete
068f6a4ec7b4: Pull complete
482801d9d6f62: Pull complete
ec91d8faf678: Pull complete
77d093e1881c: Pull complete
8495921eba41: Pull complete
a894d217b765: Pull complete
c487ddc434df: Pull complete
98803a5b897b: Pull complete
6a9e25e0066f: Pull complete
a059aaee3687: Pull complete
16ba04bd2799: Pull complete
Digest: sha256:888d11c13cd0dec7446a737e5f717a6b43f470ef197452657df1e71a163e6979e
Status: Downloaded newer image for wordpress:latest
Creating ec2-user_db_1 ... done
Creating ec2-user_wordpress_1 ... done
[ec2-user@ip-172-31-89-99 ~]$ docker login
Log in with your Docker ID or email address to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com/ to create one.
You can log in with your password or a Personal Access Token (PAT). Using a limited-scope PAT grants better security and is required for organizations using SSO. Learn more at https://docs.docker.com/go/access-tokens/
Username: vinaycluster.local
Password:
Error response from daemon: Get "https://registry-1.docker.io/v2/": unauthorized: incorrect username or password
[ec2-user@ip-172-31-89-99 ~]$ docker login
Log in with your Docker ID or email address to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com/ to create one.
You can log in with your password or a Personal Access Token (PAT). Using a limited-scope PAT grants better security and is required for organizations using SSO. Learn more at https://docs.docker.com/go/access-tokens/
Username: vinay182206
Password:
WARNING! Your password will be stored unencrypted in /home/ec2-user/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store
Login Succeeded
[ec2-user@ip-172-31-89-99 ~]$ |

```

The screenshot shows a Linux terminal window with several tabs open. The current tab displays the output of a Docker login command, showing the download of the latest version of the 'wordpress' image and the creation of database and WordPress containers. It also prompts for a Docker ID or password. Below the terminal is a Windows taskbar with various icons for file explorer, browser, and system tools. The system tray shows battery status (1.09%), signal strength, and the date/time (22-10-2024).

- Write the deployment file and service file in yaml script in the terminal where the kubernetes installed.
- In that provide the image name and username correctly.
- And save it than apply the commands  
kubectl apply deployment.yml  
Kubectl apply service.yml

```
● ec2-user@ip-172-31-81-198:~ x ec2-user@ip-172-31-89-99:~ + v

apiVersion: apps/v1
kind: Deployment
metadata:
  name: wordpress-deployment
  labels:
    app: wordpress
    tier: frontend
spec:
  replicas: 4
  selector:
    matchLabels:
      tier: frontend
  template:
    metadata:
      labels:
        tier: frontend
    spec:
      containers:
        - name: wordpress
          image: vinay182206/wordpress
          ports:
            - containerPort: 80
-- INSERT --
22, 28 All
29°C Partly sunny 16:32 22-10-2024
```

```
● ec2-user@ip-172-31-81-198:~ x ec2-user@ip-172-31-89-99:~ + v

apiVersion: v1
kind: Service
metadata:
  name: wordpress-service
spec:
  type: LoadBalancer
  ports:
    - port: 80
  selector:
    tier: frontend
-- INSERT --
4, 26 All
28°C Cloudy 15:45 22-10-2024
```

- After applying the kubectl commands we will get this type and here we can get the end point address of the loadbalancer.
- Copy it and browse in the google we will get the output.

```
ec2-user@ip-172-31-81-198:~ % ec2-user@ip-172-31-89-99:~ %
NAME          READY  STATUS    RESTARTS   AGE
pod/wordpress-deployment-7b7cfbdbf5-5bttw  1/1   Running   0          21s
pod/wordpress-deployment-7b7cfbdbf5-b9bhj  1/1   Running   0          21s
pod/wordpress-deployment-7b7cfbdbf5-cbcnf  1/1   Running   0          21s
pod/wordpress-deployment-7b7cfbdbf5-rfc1b  1/1   Running   0          21s

NAME          AGE     TYPE      CLUSTER-IP        EXTERNAL-IP
service/kubernetes  62m    ClusterIP   100.64.0.1       <none>
service/wordpress-service  15s    LoadBalancer  100.68.145.250  a022c44663d2a46f9965ede372c72b28-810151277.us-east-1.elb.amazonaws.com  80:30249/TCP

NAME          READY  UP-TO-DATE  AVAILABLE  AGE
deployment.apps/wordpress-deployment  4/4    4           4          21s

NAME          DESIRED  CURRENT  READY   AGE
replicaset.apps/wordpress-deployment-7b7cfbdbf5  4        4        4       21s
[ec2-user@ip-172-31-81-198 ~]$ kubectl get all
NAME          READY  STATUS    RESTARTS   AGE
pod/wordpress-deployment-7b7cfbdbf5-5bttw  1/1   Running   0          32s
pod/wordpress-deployment-7b7cfbdbf5-b9bhj  1/1   Running   0          32s
pod/wordpress-deployment-7b7cfbdbf5-cbcnf  1/1   Running   0          32s
pod/wordpress-deployment-7b7cfbdbf5-rfc1b  1/1   Running   0          32s

NAME          AGE     TYPE      CLUSTER-IP        EXTERNAL-IP
service/kubernetes  26s    ClusterIP   100.64.0.1       <none>
service/wordpress-service  26s    LoadBalancer  100.68.145.250  a022c44663d2a46f9965ede372c72b28-810151277.us-east-1.elb.amazonaws.com  80:30249/TCP

NAME          READY  UP-TO-DATE  AVAILABLE  AGE
deployment.apps/wordpress-deployment  4/4    4           4          32s

NAME          DESIRED  CURRENT  READY   AGE
replicaset.apps/wordpress-deployment-7b7cfbdbf5  4        4        4       32s
[ec2-user@ip-172-31-81-198 ~]$ |
```

29°C Partly sunny 16:20 22-10-2024 ENG IN

- Or else we can go to aws console and copy the loadbalancer end point url and browse it.

The Network Load Balancer zonal shift capability has changed

- Use of Amazon Application Recovery Controller (ARC) zonal shift now requires the Network Load Balancer attribute `zonal shift capability` to be enabled.
- Zonal shift capability now supports cross-zone enabled Network Load Balancers.

**Load balancers (2)**

Name	DNS name	State	VPC ID	Availability Zones	Type
a022c44663d2a46f9965ede372c72b28-810151277.us-east-1.elb.amazonaws.com	a022c44663d2a46f9965ede372c72b28-810151277.us-east-1.elb.amazonaws.com	vpc-09d71fd3583b1f8f9	2 Availability Zones	classic	
api-vinaycluster-local-akt8...	api-vinaycluster-local-akt8...	Active	vpc-09d71fd3583b1f8f9	2 Availability Zones	network

0 load balancers selected

Select a load balancer above.

CloudShell Feedback 16:32 22-10-2024 ENG IN

- Output.

