

PROJECT

DEPLOY WORDPRESS

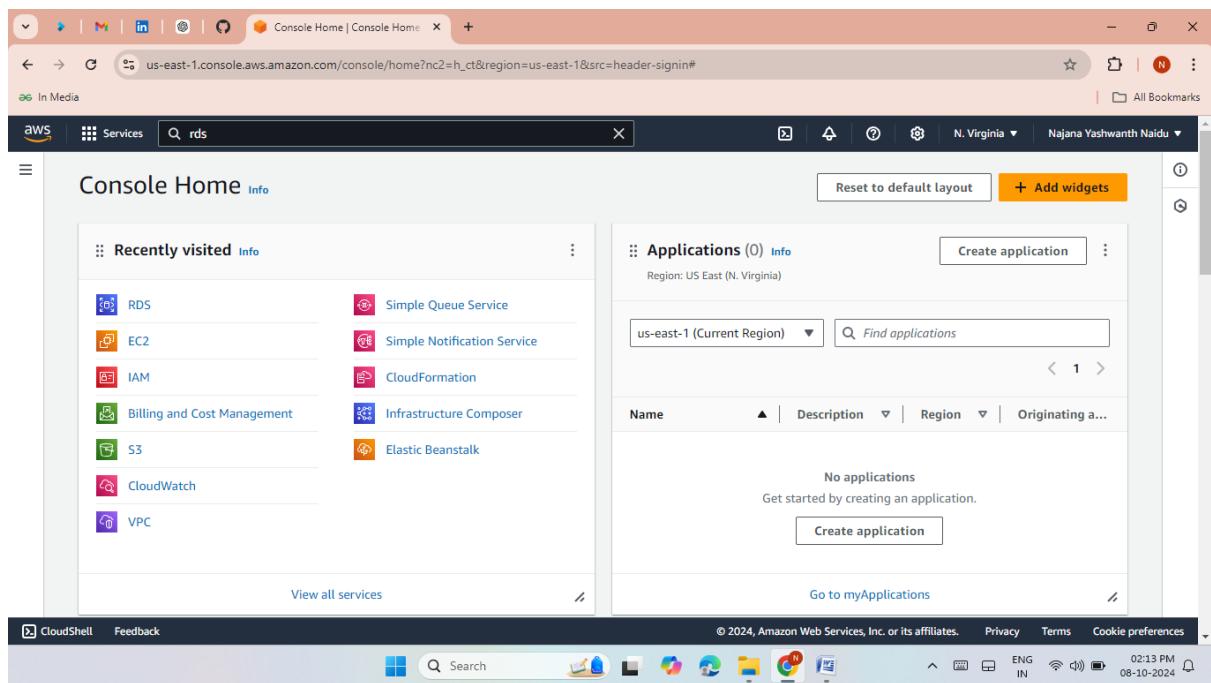
N.Yashwanth Naidu

METHOD-1

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

(manually)

- First login into the AWS account then click on RDS.



- Choose a database creation method (standard creation).

The screenshot shows the 'Create database' page in the AWS RDS Management console. The 'Choose a database creation method' section has the 'Standard create' radio button selected. Below it, the 'Engine options' section shows the 'Engine type' dropdown set to 'Info'. Under 'MySQL' engine type, the 'Aurora (MySQL Compatible)' option is selected. To the right, a detailed description of MySQL is provided, listing its features such as support for up to 64 TiB of database size and various instance classes.

- Choose a database engine option(MYSQL).

The screenshot shows the 'Create database' page in the AWS RDS Management console. The 'Engine options' section displays several engine types: Aurora (MySQL Compatible), Aurora (PostgreSQL Compatible), MySQL (selected), MariaDB, PostgreSQL, and Oracle. The MySQL option is highlighted with a blue border. To the right, a detailed description of MySQL is provided, listing its features such as support for up to 64 TiB of database size and various instance classes.

➤ Select the Engine version.

The screenshot shows the AWS RDS Management console. In the top navigation bar, the URL is `us-east-1.console.aws.amazon.com/rds/home?region=us-east-1#launch-dbinstance:`. The left sidebar has 'Services' selected. The main content area is titled 'MySQL'. On the left, there's a 'Edition' section with 'MySQL Community' selected. Below it is a 'Engine version' dropdown set to 'MySQL 8.0.39'. There are two filter options: 'Show only versions that support the Multi-AZ DB cluster' (unchecked) and 'Show only versions that support the Amazon RDS Optimized Writes' (unchecked). A note about RDS Extended Support is present. On the right, a detailed description of MySQL is provided, along with a bulleted list of features. At the bottom of the page, there are links for CloudShell and Feedback, and standard browser navigation controls.

➤ Select the template as free tier.

This screenshot continues from the previous one, showing the 'Templates' step. The 'Free tier' option is selected in the 'Templates' section, which is highlighted with a blue border. Below this, the 'Availability and durability' section is visible, containing options for deployment: 'Multi-AZ DB Cluster', 'Multi-AZ DB instance (not supported for Multi-AZ DB cluster snapshot)', and 'Single DB instance (not supported for Multi-AZ DB cluster snapshot)'. The 'MySQL' sidebar on the right remains the same, detailing MySQL features. The bottom of the screen shows the usual AWS footer with links for CloudShell, Feedback, and various services like CloudWatch and Lambda.

- Enter the Master username and password for database access.

The screenshot shows the 'Create database' page in the AWS RDS Management console. The 'DB instance identifier' field contains 'yash-database'. The 'Master username' field contains 'wordpress'. The 'Credentials Settings' section shows 'Self managed' selected, with the note 'Create your own password or have RDS create a password that you manage.' Other options like 'Managed in AWS Secrets Manager - most secure' and 'Auto generate password' are also shown. A sidebar on the right provides information about MySQL, including its popularity and various features.

- Select db.t3.micro.

The screenshot shows the 'Create database' page in the AWS RDS Management console. In the 'Instance configuration' section, the 'DB instance class' dropdown is set to 'db.t3.micro'. Other options like 'Standard classes (includes m classes)', 'Memory optimized classes (includes r and x classes)', and 'Burstable classes (includes t classes)' are also listed. The sidebar on the right continues to provide information about MySQL.

- Now select the storage type as general purpose SSD(gp2) and Enter the storage values as max(1000gb) and min(20gb).

Provisioned IOPS SSD (io2) storage volumes are now available.

General Purpose SSD (gp2)
Baseline performance determined by volume size

Allocated storage [Info](#)
20 GiB
Allocated storage value must be 20 GiB to 6,144 GiB

After you modify the storage for a DB instance, the status of the DB instance will be in storage-optimization. Your instance will remain available as the storage-optimization operation completes. [Learn more](#)

Storage autoscaling

Storage autoscaling [Info](#)
Provides dynamic scaling support for your database's storage based on your application's needs.

Enable storage autoscaling
Enabling this feature will allow the storage to increase after the specified threshold is exceeded.

Maximum storage threshold [Info](#)
Charges will apply when your database autoscales to the specified threshold
1000 GiB

- Now select default VPC and it automatically select the Database subnet group.

Virtual private cloud (VPC) [Info](#)
Choose the VPC. The VPC defines the virtual networking environment for this DB instance.
Default VPC (vpc-05f2f9c35f844e371)
6 Subnets, 6 Availability Zones

Only VPCs with a corresponding DB subnet group are listed.

After a database is created, you can't change its VPC.

DB subnet group [Info](#)
Choose the DB subnet group. The DB subnet group defines which subnets and IP ranges the DB instance can use in the VPC that you selected.
default-vpc-05f2f9c35f844e371
6 Subnets, 6 Availability Zones

Public access [Info](#)
 Yes
RDS assigns a public IP address to the database. Amazon EC2 instances and other resources outside of the VPC can connect to your database. Resources inside the VPC can also connect to the database. Choose one or more VPC security groups that specify which resources can connect to the database.
 No
RDS doesn't assign a public IP address to the database. Only Amazon EC2 instances and other resources inside the VPC can connect to your database. Choose one or more VPC security groups that specify which resources can connect to the database.

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

Additional configuration

Database options

Initial database name [Info](#)
yash_database

If you do not specify a database name, Amazon RDS does not create a database.

DB parameter group [Info](#)
default.mysql8.0

Option group [Info](#)
default:mysql-8.0

Backup

Enable automated backups
Creates a point-in-time snapshot of your database

MySQL

MySQL is the most popular open source database in the world. MySQL on RDS offers the rich features of the MySQL community edition with the flexibility to easily scale compute resources or storage capacity for your database.

- Supports database size up to 64 TiB.
- Supports General Purpose, Memory Optimized, and Burstable Performance instance classes.
- Supports automated backup and point-in-time recovery.
- Supports up to 15 Read Replicas per instance, within a single Region or 5 read

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

Successfully created database yash-database

You can use settings from yash-database to simplify configuration of [suggested database add-ons](#) while we finish creating your DB for you.

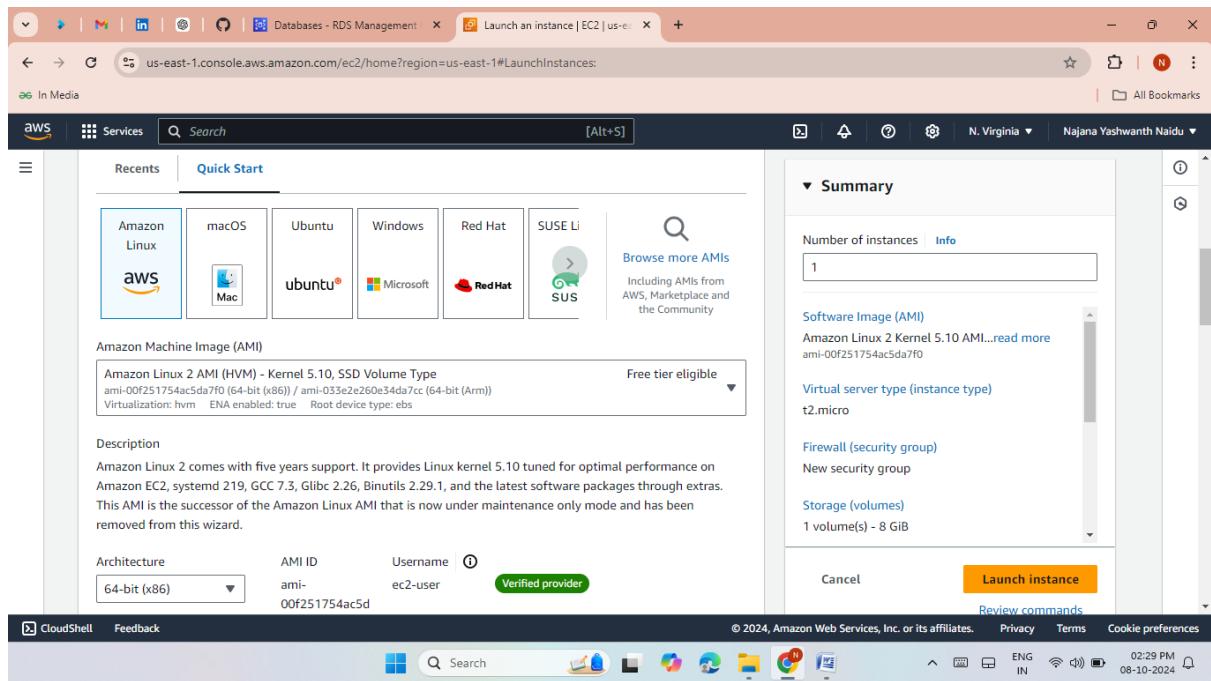
RDS > Databases

Consider creating a Blue/Green Deployment to minimize downtime during upgrades

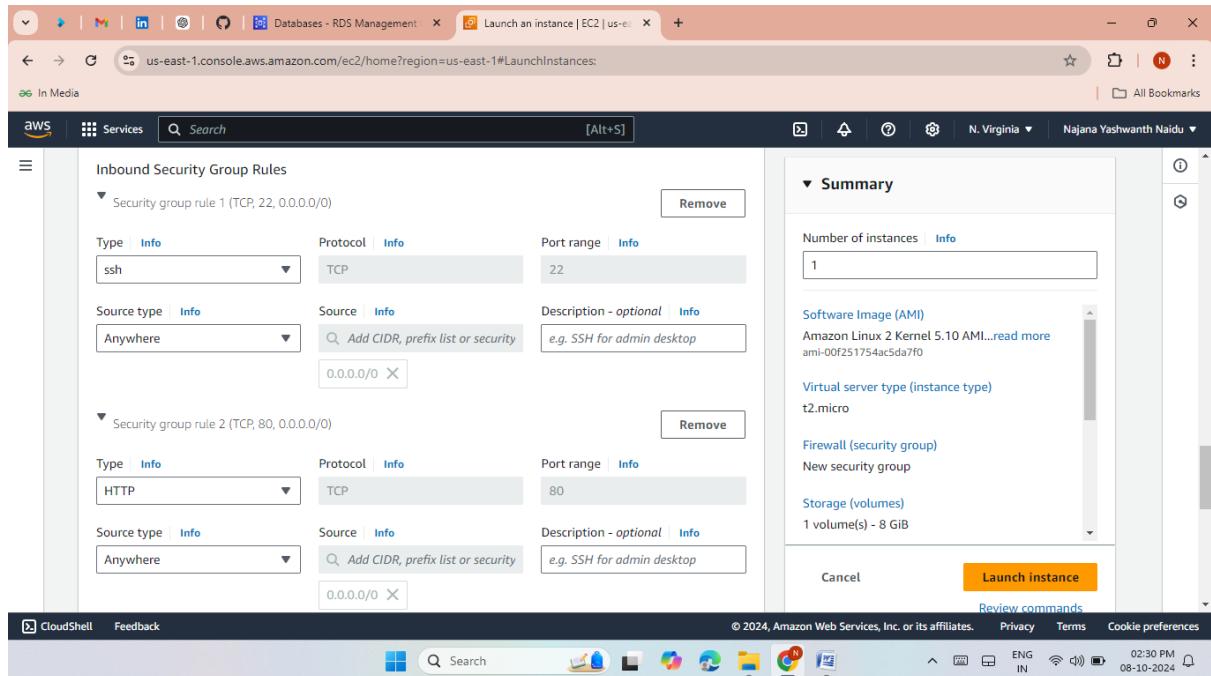
You may want to consider using Amazon RDS Blue/Green Deployments and minimize your downtime during upgrades. A Blue/Green Deployment provides a staging environment for changes to production databases. [RDS User Guide](#) [Aurora User Guide](#)

Databases (1)						
<input checked="" type="checkbox"/> Group resources C Modify Actions Restore from S3 Create database						
<input type="text"/> Filter by databases						
DB identifier Status Role Engine Region ... Size						
yash-database Available Instance MySQL Co... us-east-1f db.t3.micro						

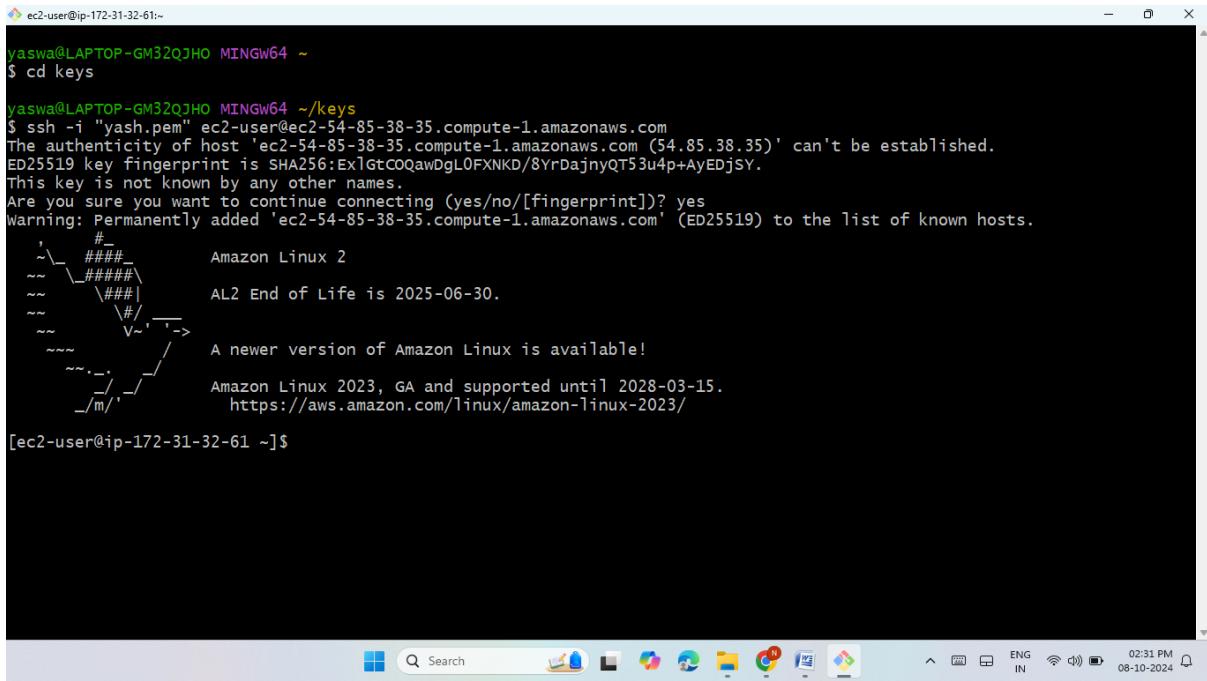
➤ Launch an instance with amazon-linux.



➤ Select the Security group with SSH(22) and HTTP(80).

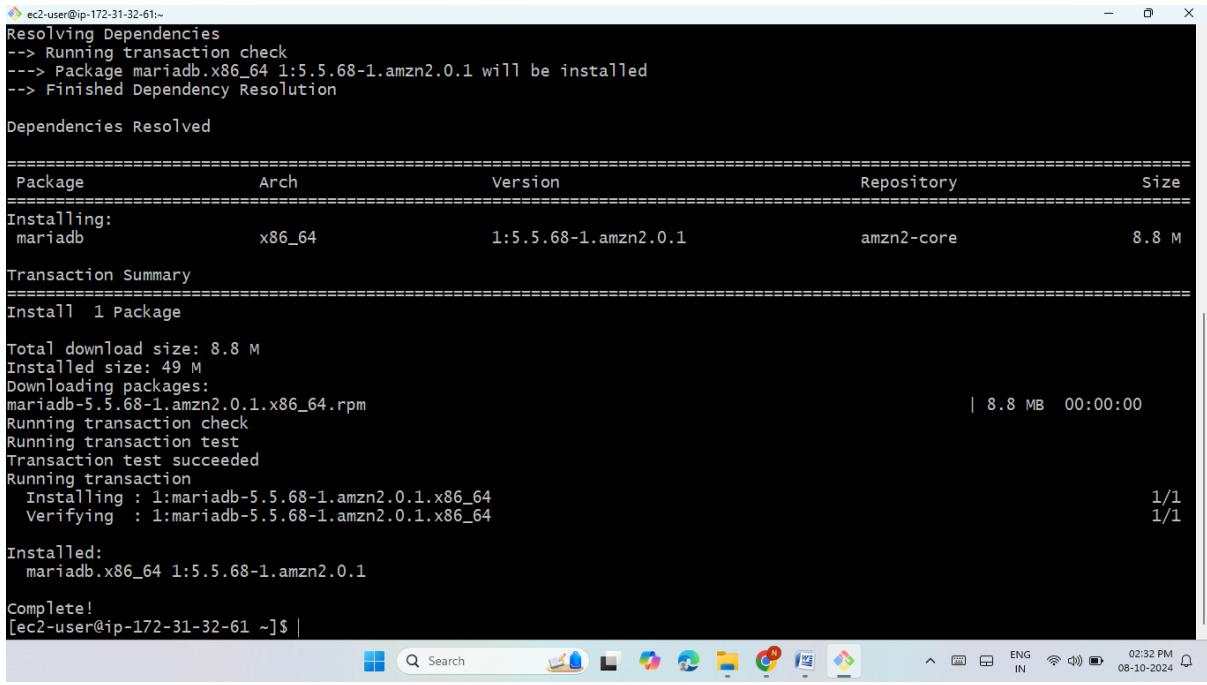


➤ Now Connect the server to locally (GitBash).



```
ec2-user@ip-172-31-32-61:~  
yasma@LAPTOP-GM32QJHO MINGW64 ~  
$ cd keys  
  
yasma@LAPTOP-GM32QJHO MINGW64 ~/keys  
$ ssh -i "yash.pem" ec2-user@ec2-54-85-38-35.compute-1.amazonaws.com  
The authenticity of host 'ec2-54-85-38-35.compute-1.amazonaws.com (54.85.38.35)' can't be established.  
ED25519 key fingerprint is SHA256:ExlGtCOQawDgLOFXNkD/8YrDajnyQT53u4p+AyEDjSY.  
This key is not known by any other names.  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added 'ec2-54-85-38-35.compute-1.amazonaws.com' (ED25519) to the list of known hosts.  
, #  
~\_\_ #####\ Amazon Linux 2  
~~ \####\ AL2 End of Life is 2025-06-30.  
~~ \#\| V~,-->  
~~ .- 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-172-31-32-61 ~]$
```

➤ Now update the linux version by using command as **sudo yum -y update** and install the MySQL by using the command as **sudo yum -y install mysql**.



```
ec2-user@ip-172-31-32-61:~  
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  
=====  
Install 1 Package  
  
Total download size: 8.8 M  
Installed size: 49 M  
Downloading packages:  
mariadb-5.5.68-1.amzn2.0.1.x86_64.rpm | 8.8 MB  00:00:00  
Running transaction check  
Running transaction test  
Transaction test succeeded  
Running transaction  
  Installing : 1:mariadb-5.5.68-1.amzn2.0.1.x86_64          1/1  
    Verifying  : 1:mariadb-5.5.68-1.amzn2.0.1.x86_64          1/1  
  
Installed:  
  mariadb.x86_64 1:5.5.68-1.amzn2.0.1  
  
Complete!  
[ec2-user@ip-172-31-32-61 ~]$
```

- Now go to Security Groups then Select Inbound rules and add 3306 port for MYSQL.

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-0892eb8b8c3d2039c	SSH	TCP	22	Cust...	
sgr-0c47567ca1befb50c	HTTP	TCP	80	Cust...	
-	Custom TCP	TCP	3306	Any...	

Add rule

⚠ Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

- Now Select the Database and click on modify then set the EC2 Security group to this Database.
- Click on modify DB instance.

Attribute	Current value	New value
Security group	default	launch-wizard-3

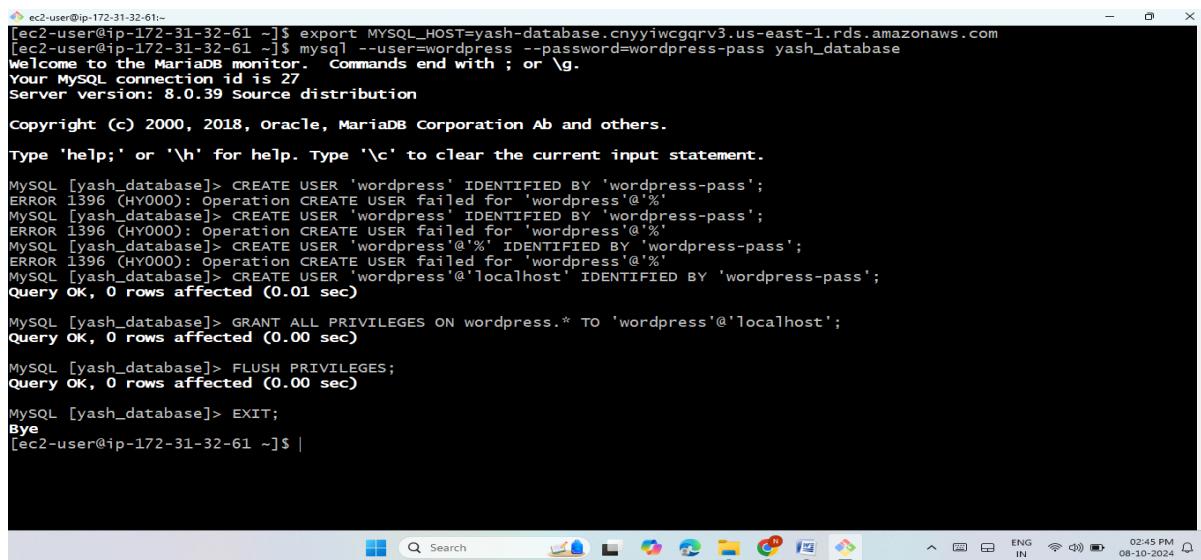
When to apply modifications

Apply during the next scheduled maintenance window
Current maintenance window: October 11, 2024 09:24 - 09:54 (UTC+5:50)

Apply immediately
The modifications in this request and any pending modifications will be asynchronously applied as soon as possible, regardless of the maintenance window setting for this database instance.

Modify DB instance

- Now access the MYSQL Database by using the command as **export MYSQL_HOST=<endpoint address>** for that go inside the created database and go to the endpoint and select and copy the endpoint address.
- Now give the credential of database by giving a command as "**mysql --user=<username> --password=<password> database-name**"(yash_database).
- Now create a database user for your wordpress application and give it permission to access the wordpress database. By using this command as.
 - **CREATE USER 'wordpress'@'localhost' IDENTIFIED BY 'wordpress-pass';**
 - **GRANT ALL PRIVILEGES ON wordpress.* TO 'wordpress'@'localhost';**
 - **FLUSH PRIVILEGES;**
 - **EXIT;**



```

ec2-user@ip-172-31-32-61:~$ export MYSQL_HOST=yash-database.cnnyiwcgqrv3.us-east-1.rds.amazonaws.com
[ec2-user@ip-172-31-32-61 ~]$ mysql --user=wordpress --password=wordpress-pass yash_database
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 27
Server version: 8.0.39 Source distribution

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

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

MySQL [yash_database]> CREATE USER 'wordpress' IDENTIFIED BY 'wordpress-pass';
ERROR 1396 (HY000): Operation CREATE USER failed for 'wordpress'@'%'
MySQL [yash_database]> CREATE USER 'wordpress' IDENTIFIED BY 'wordpress-pass';
ERROR 1396 (HY000): Operation CREATE USER failed for 'wordpress'@'%'
MySQL [yash_database]> CREATE USER 'wordpress'@'%' IDENTIFIED BY 'wordpress-pass';
ERROR 1396 (HY000): Operation CREATE USER failed for 'wordpress'@'%'
MySQL [yash_database]> CREATE USER 'wordpress'@'localhost' IDENTIFIED BY 'wordpress-pass';
Query OK, 0 rows affected (0.01 sec)

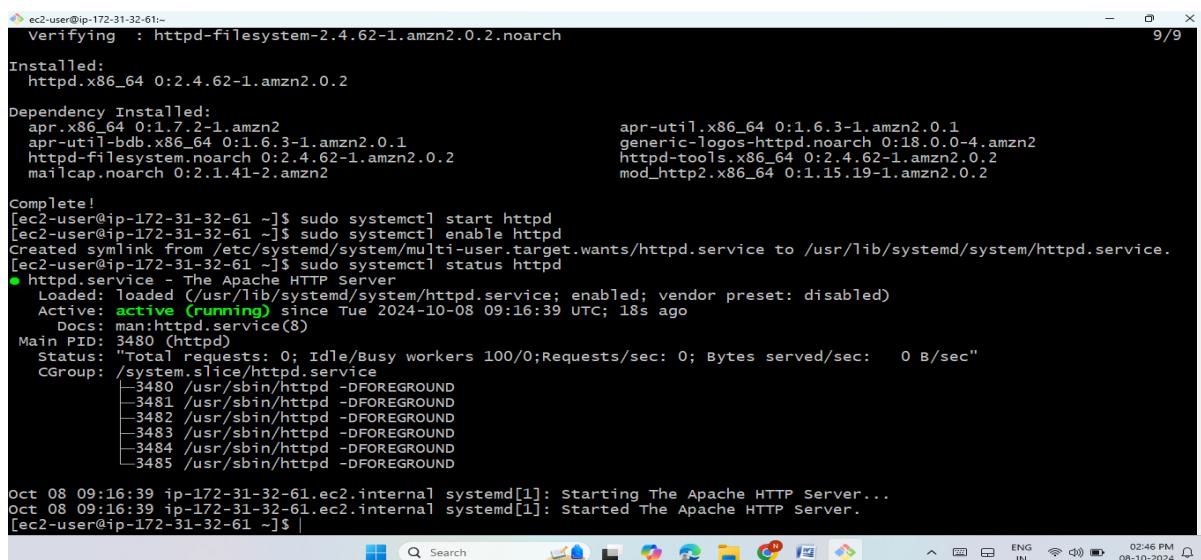
MySQL [yash_database]> GRANT ALL PRIVILEGES ON wordpress.* TO 'wordpress'@'localhost';
Query OK, 0 rows affected (0.00 sec)

MySQL [yash_database]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)

MySQL [yash_database]> EXIT;
Bye
[ec2-user@ip-172-31-32-61 ~]$ |

```

- Now Install HTTPD and then start and enable the HTTPD.



```

ec2-user@ip-172-31-32-61:~$ Verifying : httpd-filesystem-2.4.62-1.amzn2.0.2.noarch
Installed:
  httpd.x86_64 0:2.4.62-1.amzn2.0.2
Dependency Installed:
  apr.x86_64 0:1.7.2-1.amzn2
  apr-util-bdb.x86_64 0:1.6.3-1.amzn2.0.1
  httpd-filesystem.noarch 0:2.4.62-1.amzn2.0.2
  mailcap.noarch 0:2.1.41-2.amzn2
Complete!
[ec2-user@ip-172-31-32-61 ~]$ sudo systemctl start httpd
[ec2-user@ip-172-31-32-61 ~]$ 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-172-31-32-61 ~]$ 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 Tue 2024-10-08 09:16:39 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"
    CGroup: /system.slice/httpd.service
           ├─3480 /usr/sbin/httpd -DFOREGROUND
           ├─3481 /usr/sbin/httpd -DFOREGROUND
           ├─3482 /usr/sbin/httpd -DFOREGROUND
           ├─3483 /usr/sbin/httpd -DFOREGROUND
           ├─3484 /usr/sbin/httpd -DFOREGROUND
           ├─3485 /usr/sbin/httpd -DFOREGROUND

Oct 08 09:16:39 ip-172-31-32-61.ec2.internal systemd[1]: Starting The Apache HTTP Server...
Oct 08 09:16:39 ip-172-31-32-61.ec2.internal systemd[1]: Started The Apache HTTP Server.
[ec2-user@ip-172-31-32-61 ~]$ 

```

- Now go to EC2 instance and copy Public IP and Paste it on Google browser it and check the official page of httpd is displays or not.



If you are a member of the general public:

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

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

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

If you are the website administrator:

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

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



- Now go to browser and search as download wordpress and click on proper link and select ad copy the address link of wordpress download file and past that along with wget command in git bash as “**wget< >**”.
- It gives the zip file to unzip that file by using a command as “**unzip<zip file>**”.

```
ec2-user@ip-172-31-32-61:~$ 
inflating: wordpress/wp-admin/js/media.js
inflating: wordpress/wp-admin/js/editor-expand.min.js
inflating: wordpress/wp-admin/js/media-gallery.min.js
inflating: wordpress/wp-admin/js/common.min.js
inflating: wordpress/wp-admin/js/tags-box.min.js
inflating: wordpress/wp-admin/js/svg-painter.min.js
inflating: wordpress/wp-admin/js/custom-background.js
inflating: wordpress/wp-admin/js/color-picker.min.js
inflating: wordpress/wp-admin/js/site-icon.min.js
inflating: wordpress/wp-admin/js/auth-app.js
inflating: wordpress/wp-admin/js/code-editor.js
inflating: wordpress/wp-admin/js/common.js
inflating: wordpress/wp-admin/js/set-post-thumbnail.min.js
inflating: wordpress/wp-admin/js/postbox.min.js
inflating: wordpress/wp-admin/js/color-picker.js
inflating: wordpress/wp-admin/js/password-strength-meter.js
inflating: wordpress/wp-admin/js/customize-nav-menus.js
inflating: wordpress/wp-admin/js/editor-expand.js
inflating: wordpress/wp-admin/js/code-editor.min.js
inflating: wordpress/wp-admin/js/set-post-thumbnail.js
inflating: wordpress/wp-admin/options-permalink.php
inflating: wordpress/wp-admin/widgets.php
inflating: wordpress/wp-admin/setup-config.php
inflating: wordpress/wp-admin/install.php
inflating: wordpress/wp-admin/admin-header.php
inflating: wordpress/wp-admin/post-new.php
inflating: wordpress/wp-admin/themes.php
inflating: wordpress/wp-admin/options-reading.php
inflating: wordpress/wp-trackback.php
inflating: wordpress/wp-comments-post.php
[ec2-user@ip-172-31-32-61 ~]$ ls
latest.zip wordpress
[ec2-user@ip-172-31-32-61 ~]$ |
```

- To run wordpress web application, you have to install run time of wordpress web application as php language with the following command **sudo amazon-linux-extras install -y lamp-mariadb10.2-php7.2 php7.2**

```

ec2-user@ip-172-31-32-61:~$ amazon-linux-extras info
40 mock available [=stable]
43 livepatch available [=stable]
44 tpython3.8 available [=stable]
45 haproxy2 available [=stable]
46 collectd available [=stable]
47 aws-nitro-enclaves-cli available [=stable]
48 R4 available [=stable]
50 kernel-5.4 available [=stable]
50 selinux-ng available [=stable]
52 tomcat9 available [=stable]
53 unbound1.13 available [=stable]
5 mariadb10.5 available [=stable]
55 kernel-5.10=latest enabled [=stable]
56 redis6 available [=stable]
58 tpostgresql12 available [=stable]
59 tpostgresql13 available [=stable]
60 mock2 available [=stable]
61 dnsmasq2.85 available [=stable]
62 kernel-5.15 available [=stable]
63 tpostgresql14 available [=stable]
64 firefox available [=stable]
65 lustre available [=stable]
67 tphp8.1 available [=stable]
67 awscli1 available [=stable]
67 tphp8.2 available [=stable]
69 dnsmasq available [=stable]
70 unbound1.17 available [=stable]
72 collectd-python3 available [=stable]
* Extra topic has reached end of support.
† Note on end-of-support. Use 'info' subcommand.
[ec2-user@ip-172-31-32-61 ~]$ ls
latest.zip wordpress
[ec2-user@ip-172-31-32-61 ~]$
```

- 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**”.

```

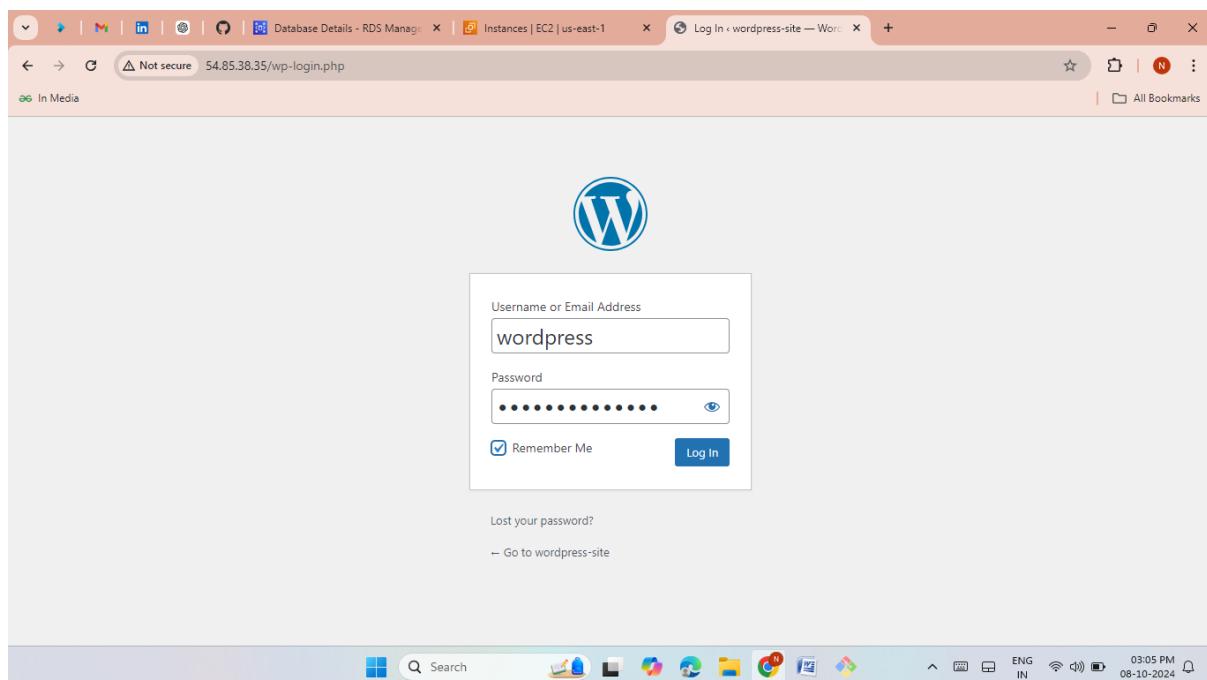
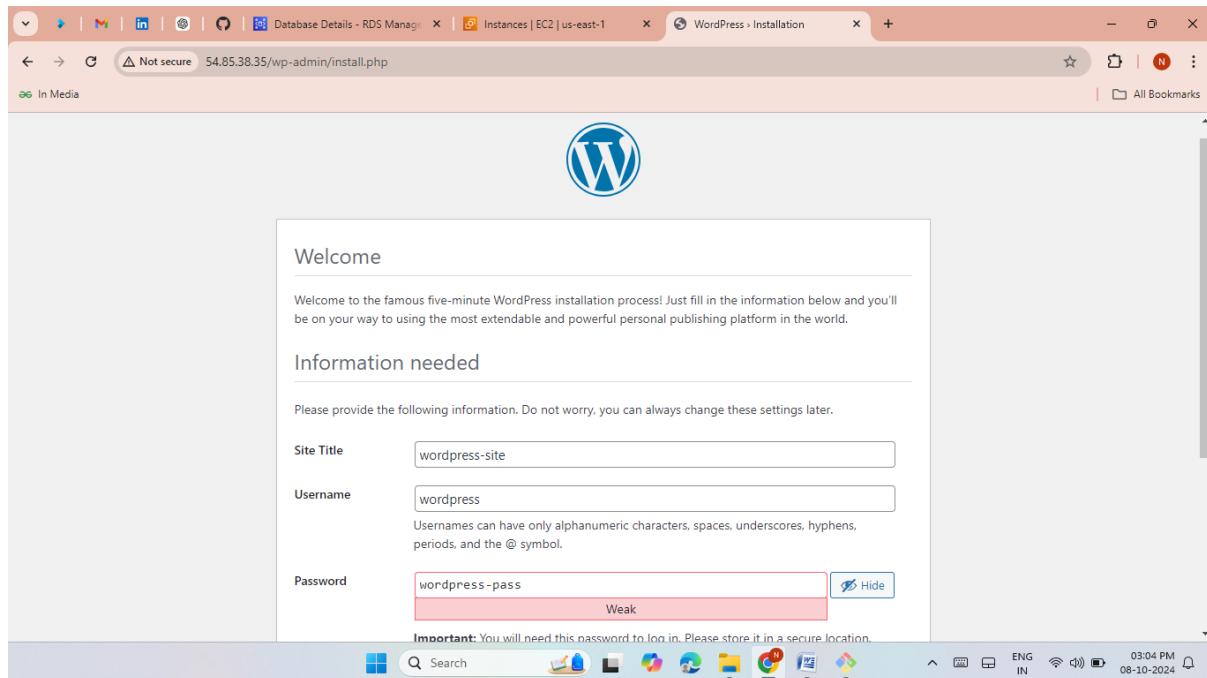
ec2-user@ip-172-31-32-61:~/wordpress
[ec2-user@ip-172-31-32-61 ~]$ ls
latest.zip wordpress
[ec2-user@ip-172-31-32-61 ~]$ cd wordpress/
[ec2-user@ip-172-31-32-61 wordpress]$ ls
index.php wp-activate.php wp-comments-post.php wp-cron.php wp-load.php wp-settings.php xmlrpc.php
license.txt wp-admin wp-config-sample.php wp-includes wp-login.php wp-signup.php
readme.html wp-blog-header.php wp-content wp-links-opml.php wp-mail.php wp-trackback.php
[ec2-user@ip-172-31-32-61 wordpress]$ sudo mv wp-config-sample.php wp-config.php
[ec2-user@ip-172-31-32-61 wordpress]$ ls
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-includes wp-login.php wp-signup.php
readme.html wp-blog-header.php wp-content wp-links-opml.php wp-mail.php wp-trackback.php
[ec2-user@ip-172-31-32-61 wordpress]$
```

- Now do some configurations in wordpress configurations files as by giving database name, username, password and hostname for that execute a command as "**sudo vi wp-config.php**".
 - Along with this change the wordpress security key to get the secret key browse in Google generate word press secret key.

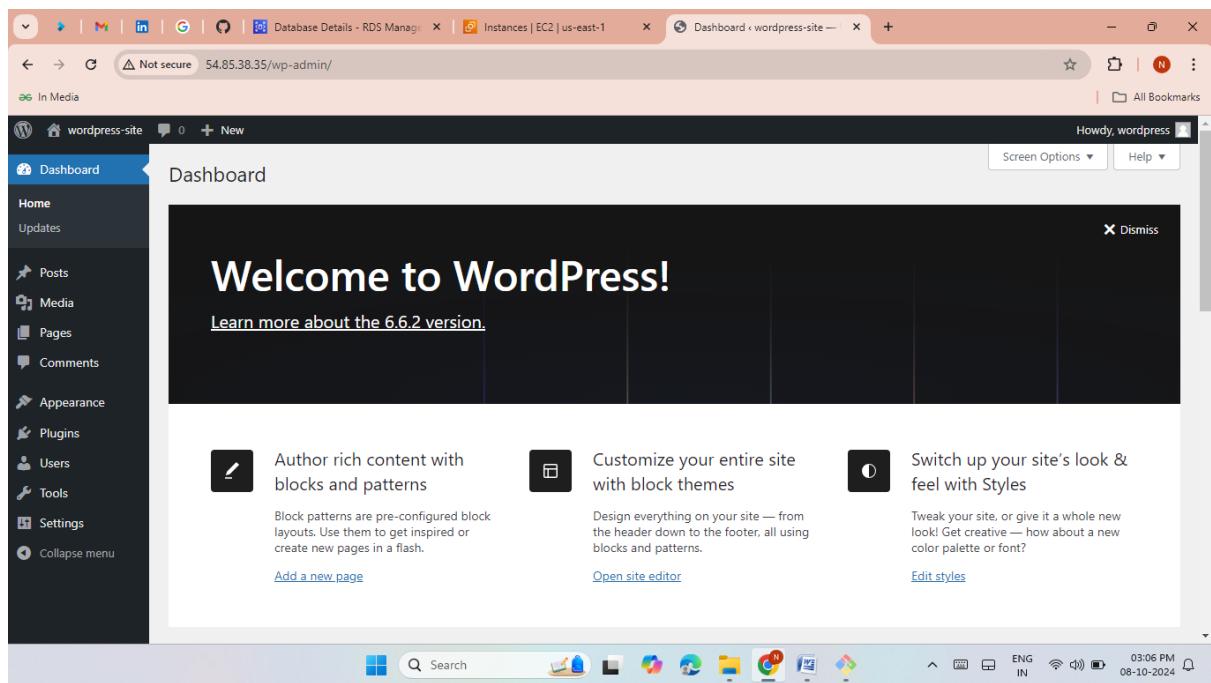
- Now copy this wordpress directory to the document root directory to host your web application of wordpress by giving a command as
 - `sudo cp -r <unzip file or wordpress directory>/* /var/www/html/`
 - `cd /var/www/html/`
 - Restart the httpd (`sudo systemctl restart httpd`).

```
[ec2-user@ip-172-31-32-61 ~]$ ls
[ec2-user@ip-172-31-32-61 ~]$ cd wordpress/
[ec2-user@ip-172-31-32-61 wordpress]$ ls
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-sample.php wp-includes wp-login.php wp-signup.php
readme.html wp-blog-header.php wp-content wp-links-opml.php wp-mail.php wp-trackback.php
[ec2-user@ip-172-31-32-61 wordpress]$ ls
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-includes wp-login.php wp-signup.php
readme.html wp-blog-header.php wp-content wp-links-opml.php wp-mail.php wp-trackback.php
[ec2-user@ip-172-31-32-61 wordpress]$ sudo vi wp-config.php
[ec2-user@ip-172-31-32-61 wordpress]$ cd
[ec2-user@ip-172-31-32-61 ~]$ ls
[ec2-user@ip-172-31-32-61 ~]$ ls
[ec2-user@ip-172-31-32-61 ~]$ sudo cp -r wordpress/* /var/www/html/
[ec2-user@ip-172-31-32-61 ~]$ cd /var/www/html/
[ec2-user@ip-172-31-32-61 html]$ ls
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-includes wp-login.php wp-signup.php
readme.html wp-blog-header.php wp-content wp-links-opml.php wp-mail.php wp-trackback.php
[ec2-user@ip-172-31-32-61 html]$ sudo systemctl restart httpd
[ec2-user@ip-172-31-32-61 html]$
```

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



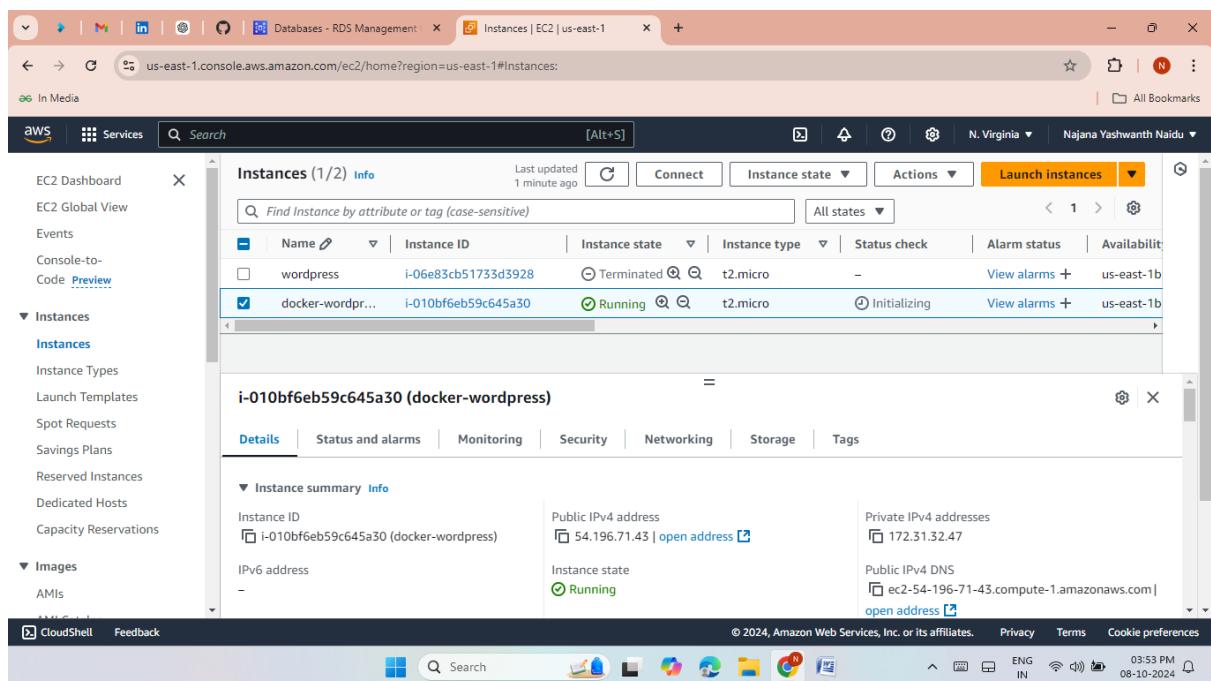
- Wordpress website was launched successfully .



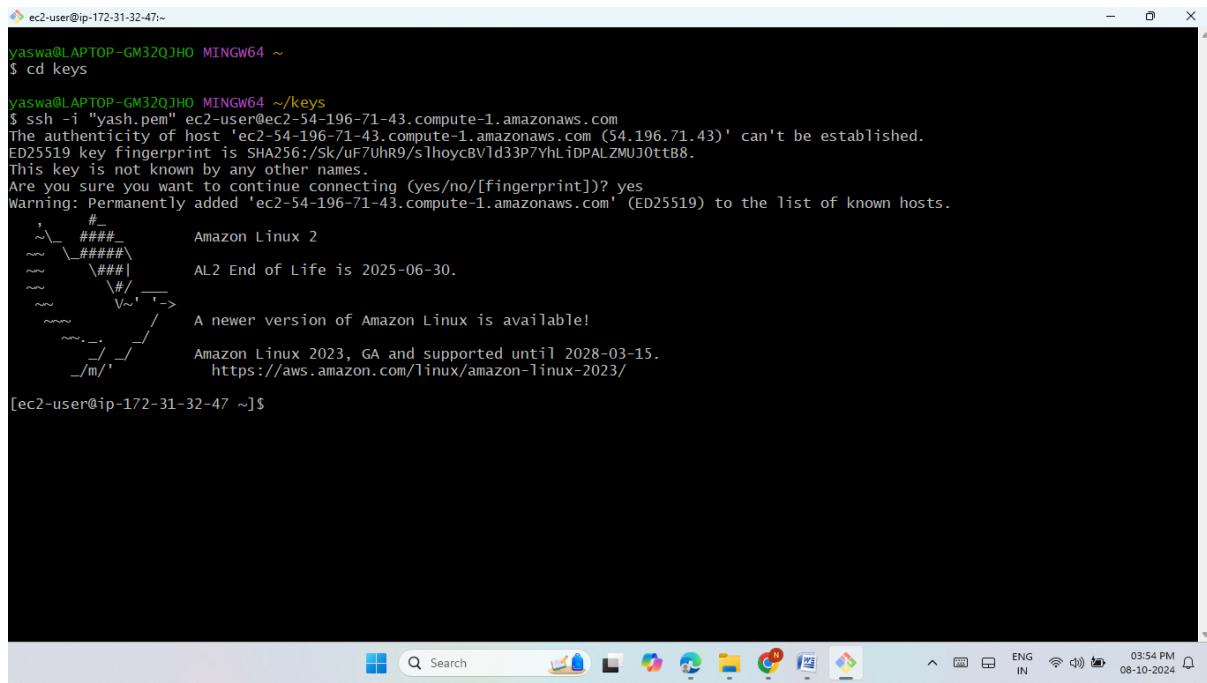
METHOD 2

Deploy WordPress web application by using docker compose file

- Launch an EC2 Instnace.

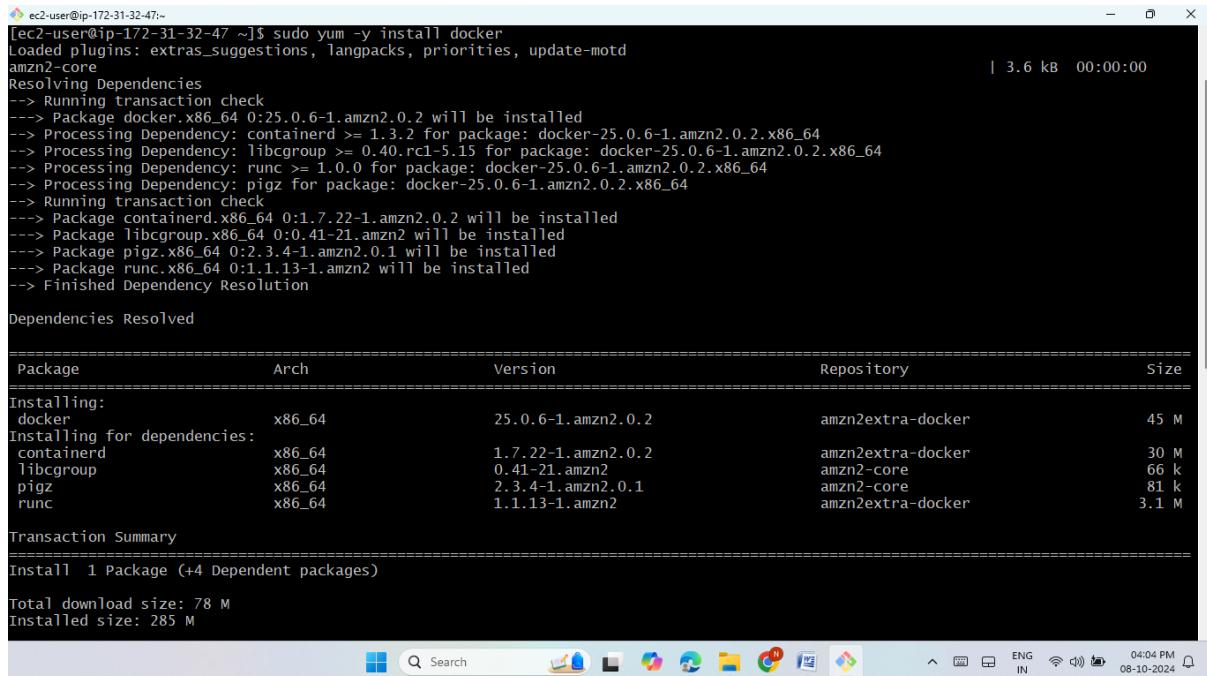


➤ Now Connect the server locally.



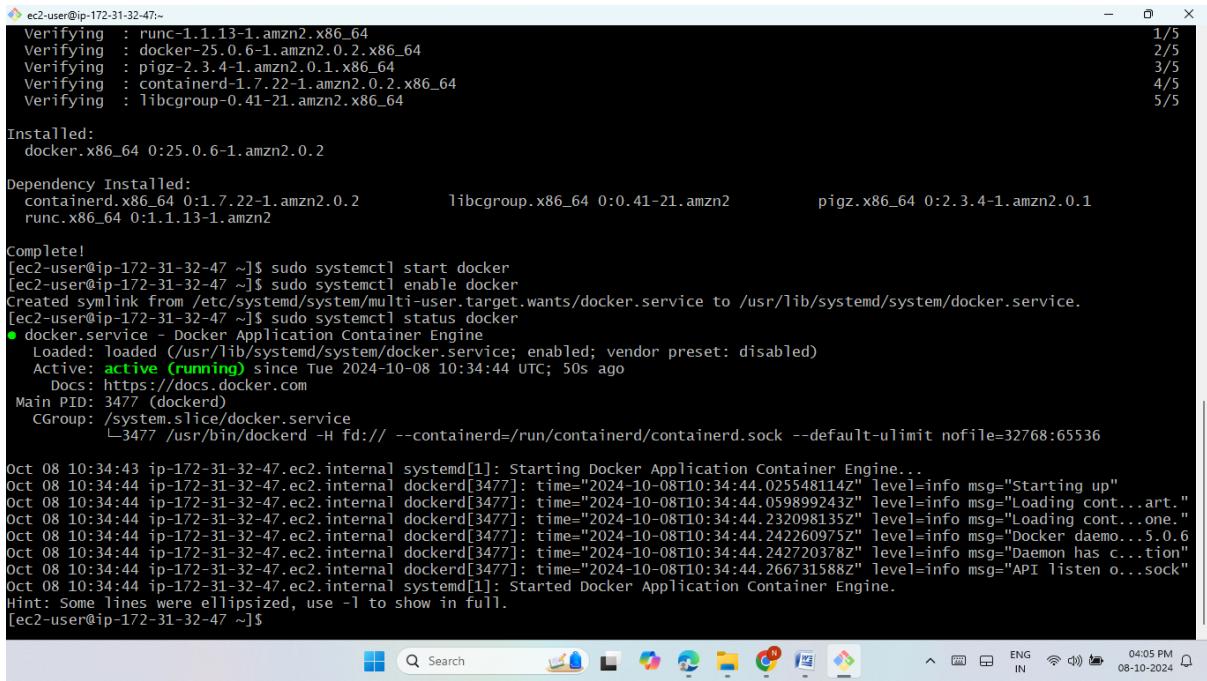
```
ec2-user@ip-172-31-32-47:~  
yasma@LAPTOP-GM32QJHO MINGW64 ~  
$ cd keys  
yasma@LAPTOP-GM32QJHO MINGW64 ~/keys  
$ ssh -i "yash.pem" ec2-user@ec2-54-196-71-43.compute-1.amazonaws.com  
The authenticity of host 'ec2-54-196-71-43.compute-1.amazonaws.com (54.196.71.43)' can't be established.  
ED25519 key fingerprint is SHA256:/sk/uF7UR9/s1hoycBVld33P7YhLiDPALZMUJ0ttB8.  
This key is not known by any other names.  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added 'ec2-54-196-71-43.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-172-31-32-47 ~]$
```

➤ Now install Docker with the command **sudo yum install -y docker**



```
ec2-user@ip-172-31-32-47:~  
[ec2-user@ip-172-31-32-47 ~]$ sudo yum -y install docker  
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd  
amzn2-core  
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.rc1-5.15 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 0:1.1.13-1.amzn2 will be installed  
--> Finished Dependency Resolution  
Dependencies Resolved  
  
=====  
Package           Arch      Version       Repository    Size  
=====  
Installing:  
docker            x86_64   25.0.6-1.amzn2.0.2      amzn2extra-docker  45 M  
Installing for dependencies:  
containerd        x86_64   1.7.22-1.amzn2.0.2      amzn2extra-docker  30 M  
libcgroup         x86_64   0.41-21.amzn2          amzn2-core          66 k  
pigz              x86_64   2.3.4-1.amzn2.0.1      amzn2-core          81 k  
runc              x86_64   1.1.13-1.amzn2          amzn2extra-docker  3.1 M  
  
Transaction Summary  
=====  
Install 1 Package (+4 Dependent packages)  
  
Total download size: 78 M  
Installed size: 285 M
```

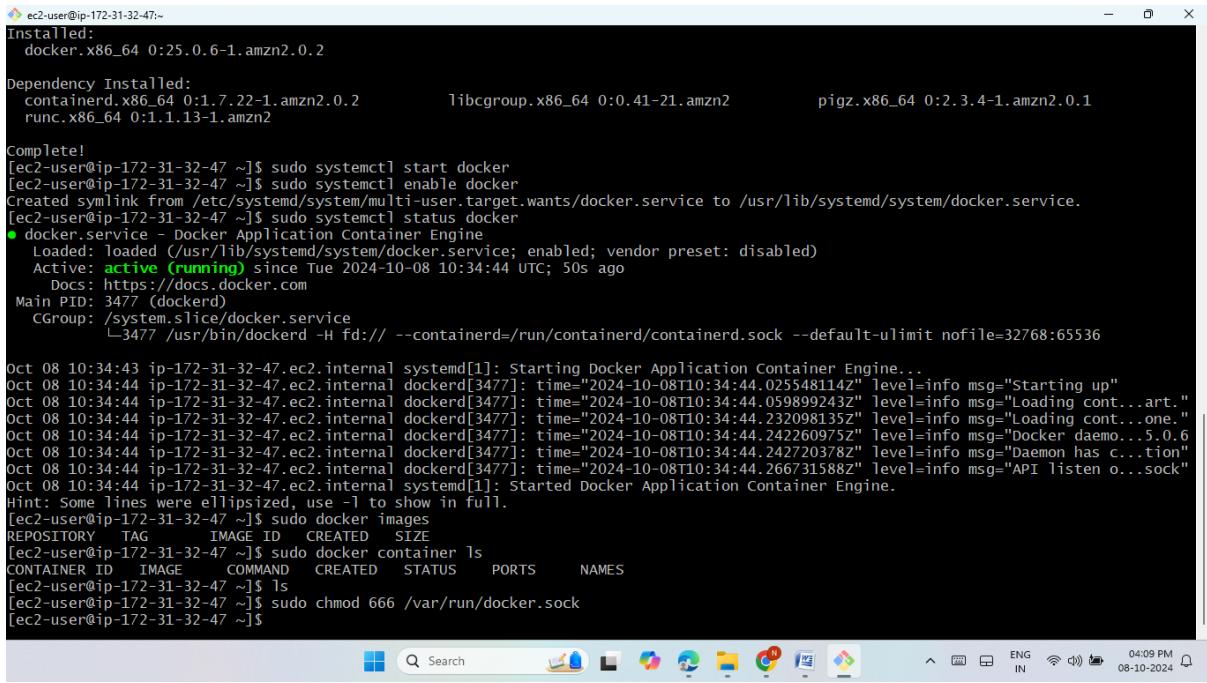
➤ Then start and enable the Docker.



```
ec2-user@ip-172-31-32-47:~ Verifying : runc-1.1.13-1.amzn2.x86_64 1/5  
Verifying : docker-25.0.6-1.amzn2.0.2.x86_64 2/5  
Verifying : pigz-2.3.4-1.amzn2.0.1.x86_64 3/5  
Verifying : containerd-1.7.22-1.amzn2.0.2.x86_64 4/5  
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 0:1.1.13-1.amzn2  
  
Complete!  
[ec2-user@ip-172-31-32-47 ~]$ sudo systemctl start docker  
[ec2-user@ip-172-31-32-47 ~]$ 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-172-31-32-47 ~]$ 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 Tue 2024-10-08 10:34:44 UTC; 50s ago  
    Docs: https://docs.docker.com  
  Main PID: 3477 (dockerd)  
  CGroup: /system.slice/docker.service  
         └─3477 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --default-ulimit nofile=32768:65536  
  
Oct 08 10:34:43 ip-172-31-32-47.ec2.internal systemd[1]: Starting Docker Application Container Engine...  
Oct 08 10:34:44 ip-172-31-32-47.ec2.internal dockerd[3477]: time="2024-10-08T10:34:44.025548114Z" level=info msg="Starting up"  
Oct 08 10:34:44 ip-172-31-32-47.ec2.internal dockerd[3477]: time="2024-10-08T10:34:44.059899243Z" level=info msg="Loading cont...art."  
Oct 08 10:34:44 ip-172-31-32-47.ec2.internal dockerd[3477]: time="2024-10-08T10:34:44.232098135Z" level=info msg="Loading cont...one."  
Oct 08 10:34:44 ip-172-31-32-47.ec2.internal dockerd[3477]: time="2024-10-08T10:34:44.242260975Z" level=info msg="Docker daemo...5.0.6  
Oct 08 10:34:44 ip-172-31-32-47.ec2.internal dockerd[3477]: time="2024-10-08T10:34:44.242720378Z" level=info msg="Daemon has c...tion"  
Oct 08 10:34:44 ip-172-31-32-47.ec2.internal dockerd[3477]: time="2024-10-08T10:34:44.266731588Z" level=info msg="API listen o...sock"  
Oct 08 10:34:44 ip-172-31-32-47.ec2.internal systemd[1]: Started Docker Application Container Engine.  
Hint: Some lines were ellipsized, use -l to show in full.  
[ec2-user@ip-172-31-32-47 ~]$
```

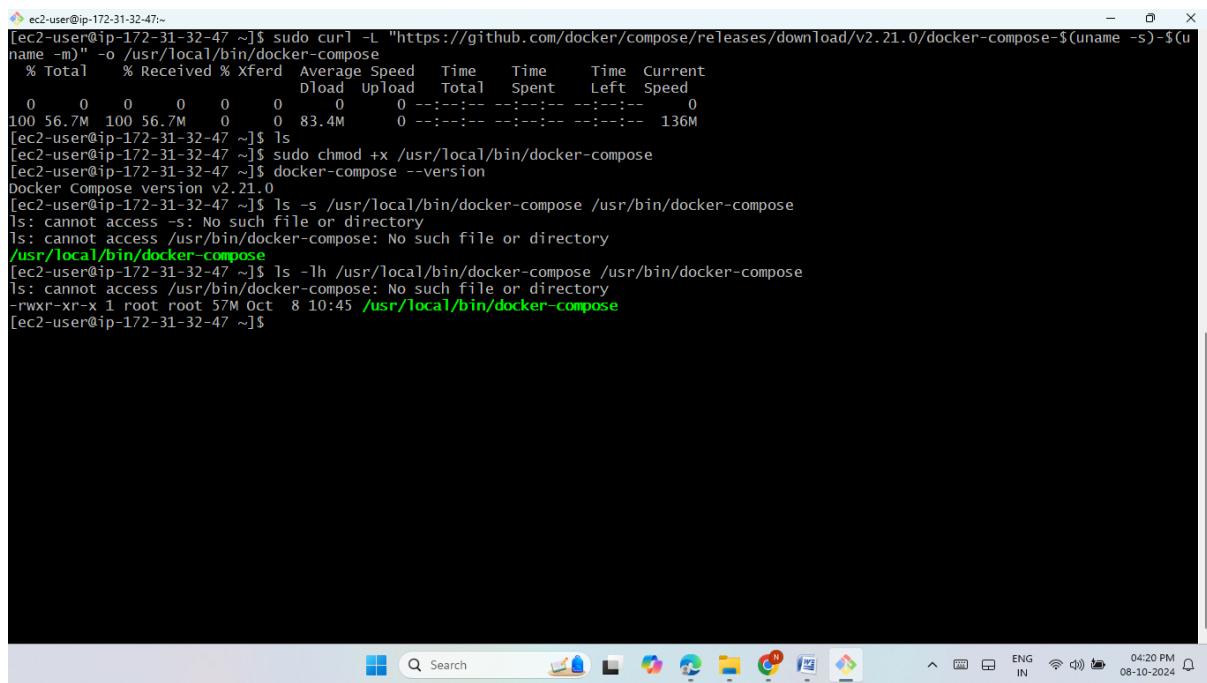
➤ Now give permissions to the docker with the command.

➤ **sudo chmod 666 /var/run/docker.sock**



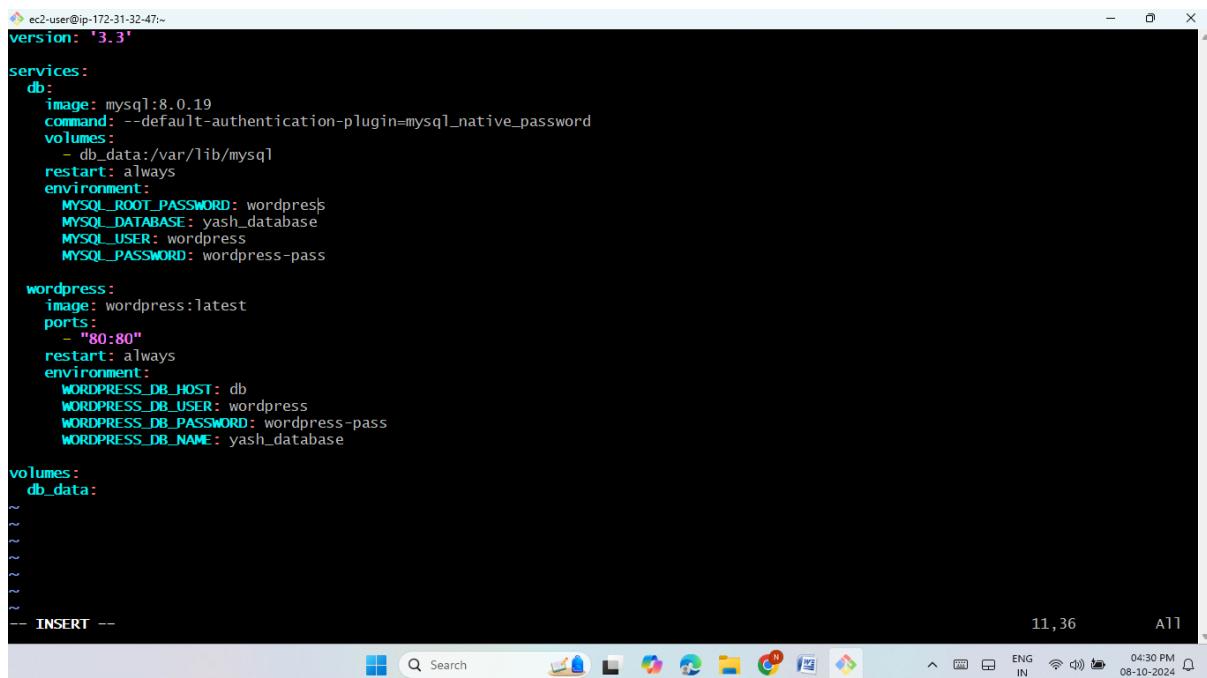
```
ec2-user@ip-172-31-32-47:~ Installed:  
docker.x86_64 0:25.0.6-1.amzn2.0.2  
  
Dependency Installed:  
containerd.x86_64 0:1.7.22-1.amzn2.0.2 libcgroup.x86_64 0:0.41-21.amzn2 pigz.x86_64 0:2.3.4-1.amzn2.0.1  
  
runc.x86_64 0:1.1.13-1.amzn2  
  
Complete!  
[ec2-user@ip-172-31-32-47 ~]$ sudo systemctl start docker  
[ec2-user@ip-172-31-32-47 ~]$ 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-172-31-32-47 ~]$ 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 Tue 2024-10-08 10:34:44 UTC; 50s ago  
    Docs: https://docs.docker.com  
  Main PID: 3477 (dockerd)  
  CGroup: /system.slice/docker.service  
         └─3477 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --default-ulimit nofile=32768:65536  
  
Oct 08 10:34:43 ip-172-31-32-47.ec2.internal systemd[1]: Starting Docker Application Container Engine...  
Oct 08 10:34:44 ip-172-31-32-47.ec2.internal dockerd[3477]: time="2024-10-08T10:34:44.025548114Z" level=info msg="Starting up"  
Oct 08 10:34:44 ip-172-31-32-47.ec2.internal dockerd[3477]: time="2024-10-08T10:34:44.059899243Z" level=info msg="Loading cont...art."  
Oct 08 10:34:44 ip-172-31-32-47.ec2.internal dockerd[3477]: time="2024-10-08T10:34:44.232098135Z" level=info msg="Loading cont...one."  
Oct 08 10:34:44 ip-172-31-32-47.ec2.internal dockerd[3477]: time="2024-10-08T10:34:44.242260975Z" level=info msg="Docker daemo...5.0.6  
Oct 08 10:34:44 ip-172-31-32-47.ec2.internal dockerd[3477]: time="2024-10-08T10:34:44.242720378Z" level=info msg="Daemon has c...tion"  
Oct 08 10:34:44 ip-172-31-32-47.ec2.internal dockerd[3477]: time="2024-10-08T10:34:44.266731588Z" level=info msg="API listen o...sock"  
Oct 08 10:34:44 ip-172-31-32-47.ec2.internal systemd[1]: Started Docker Application Container Engine.  
Hint: Some lines were ellipsized, use -l to show in full.  
[ec2-user@ip-172-31-32-47 ~]$ sudo docker images  
REPOSITORY TAG IMAGE ID CREATED SIZE  
[ec2-user@ip-172-31-32-47 ~]$ sudo docker container ls  
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES  
[ec2-user@ip-172-31-32-47 ~]$ ls  
[ec2-user@ip-172-31-32-47 ~]$ sudo chmod 666 /var/run/docker.sock  
[ec2-user@ip-172-31-32-47 ~]$
```

- Now install the Docker composer by using Google.
- Give permissions to the Docker compose with the command
- **sudo chmod +x /usr/local/bin/docker-compose**
- Now create the symbolic link by using command as **ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose** (it works on backend to start).



```
ec2-user@ip-172-31-32-47:~$ sudo curl -L "https://github.com/docker/compose/releases/download/v2.21.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
  % Total    % Received % Xferd  Average Speed   Time     Time      Current
          Dload  Upload Total Spent   Left Speed
0  100 56.7M 100 56.7M 0       0  83.4M 0 --:--:-- --:--:-- --:--:-- 136M
[ec2-user@ip-172-31-32-47 ~]$ ls
[ec2-user@ip-172-31-32-47 ~]$ sudo chmod +x /usr/local/bin/docker-compose
[ec2-user@ip-172-31-32-47 ~]$ docker-compose --version
Docker Compose version v2.21.0
[ec2-user@ip-172-31-32-47 ~]$ ls -s /usr/local/bin/docker-compose /usr/bin/docker-compose
ls: cannot access -s: No such file or directory
ls: cannot access /usr/bin/docker-compose: No such file or directory
/usr/local/bin/docker-compose
[ec2-user@ip-172-31-32-47 ~]$ ls -lh /usr/local/bin/docker-compose /usr/bin/docker-compose
ls: cannot access /usr/bin/docker-compose: No such file or directory
-rwxr-xr-x 1 root root 57M Oct  8 10:45 /usr/local/bin/docker-compose
[ec2-user@ip-172-31-32-47 ~]$
```

- Now to create a docker-compose.yml file in vi mode as **sudo vi docker-compose.yml**



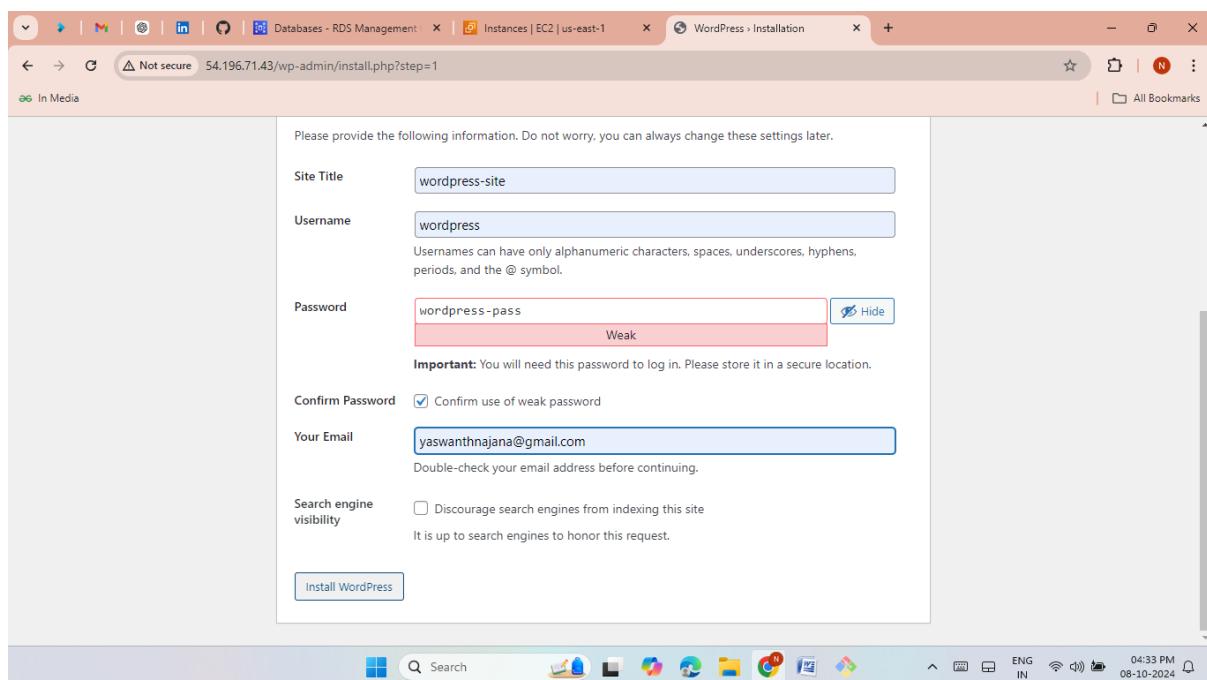
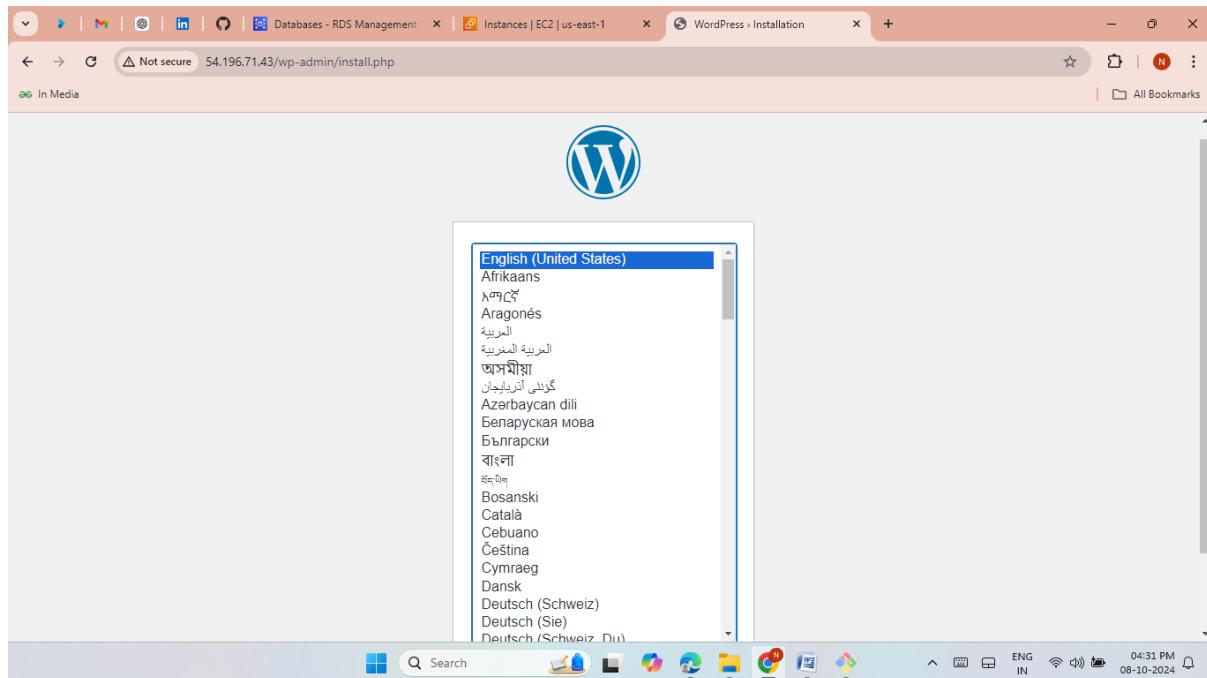
```
ec2-user@ip-172-31-32-47:~$ version: '3.3'
services:
  db:
    image: mysql:8.0.19
    command: --default-authentication-plugin=mysql_native_password
    volumes:
      - db_data:/var/lib/mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: wordpress
      MYSQL_DATABASE: yash_database
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: wordpress-pass
  wordpress:
    image: wordpress:latest
    ports:
      - "80:80"
    restart: always
    environment:
      WORDPRESS_DB_HOST: db
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress-pass
      WORDPRESS_DB_NAME: yash_database
volumes:
  db_data:
```

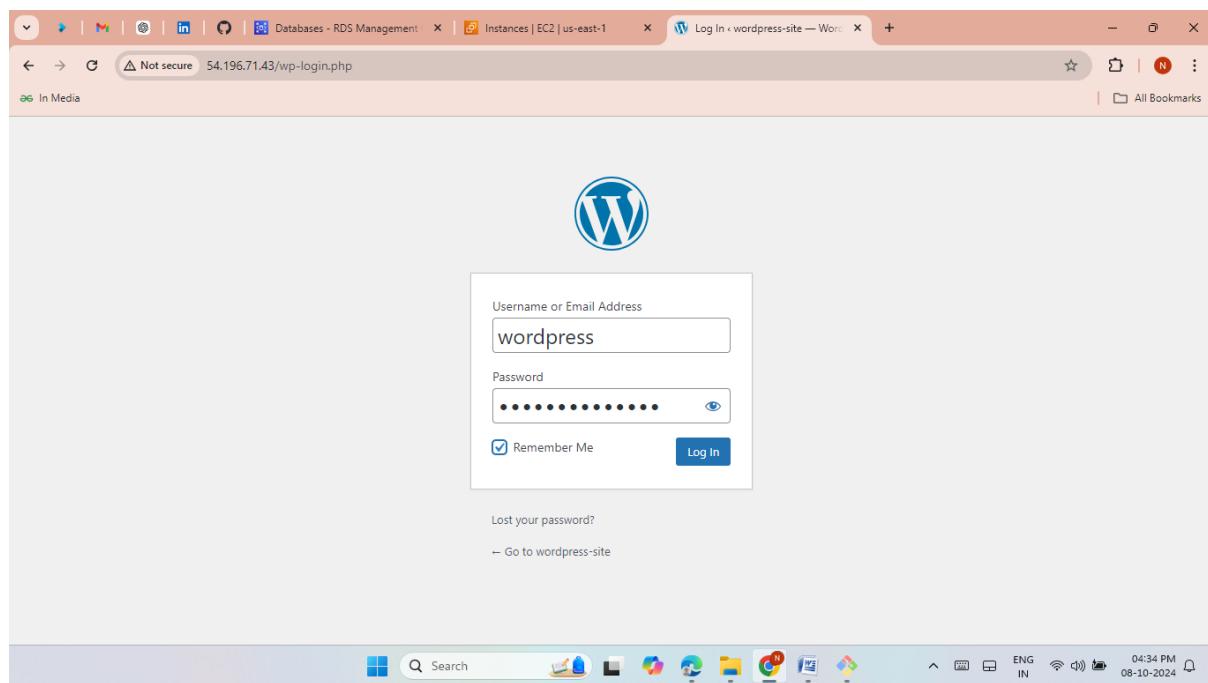
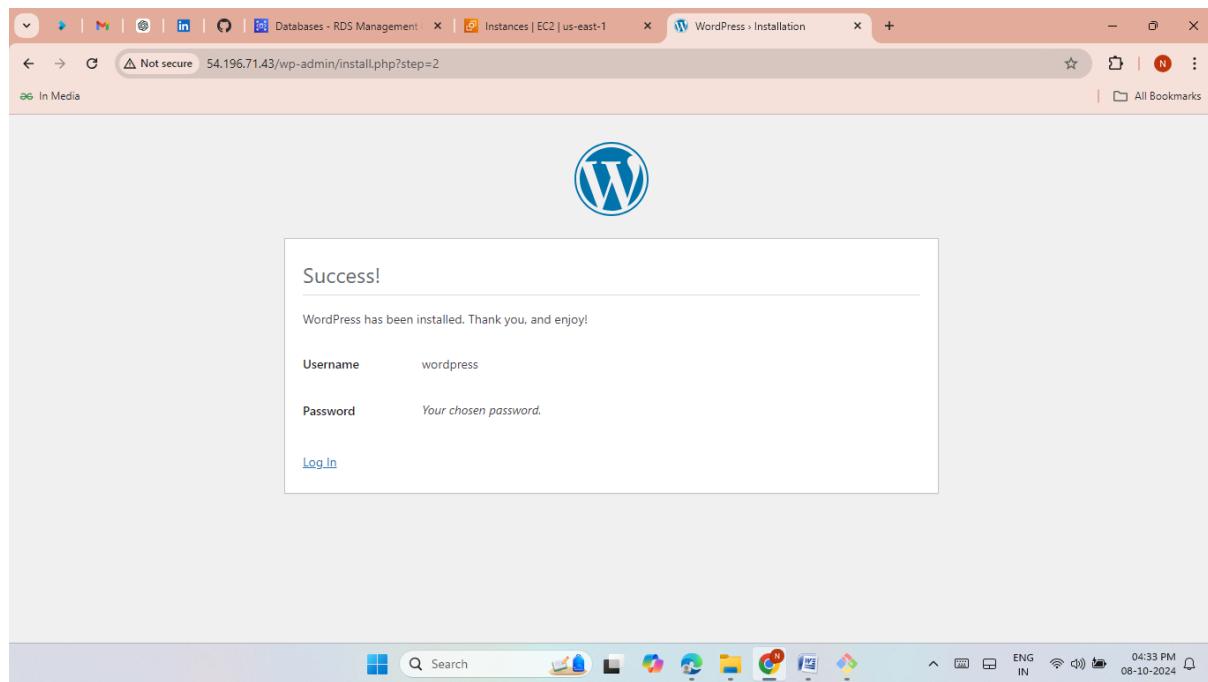
- Now to pull the images of MYSQL and WORDPRESS we have to execute this created **docker-compose.Yml** file by using the command as **<docker –compose up –d>**.

```
[ec2-user@ip-172-31-32-47:~]$ sudo curl -L "https://github.com/docker/compose/releases/download/v2.21.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
[ec2-user@ip-172-31-32-47:~]$ ls
[ec2-user@ip-172-31-32-47:~]$ docker-compose --version
Docker Compose version v2.21.0
[ec2-user@ip-172-31-32-47:~]$ ls -s /usr/local/bin/docker-compose /usr/bin/docker-compose
ls: cannot access -s: No such file or directory
ls: cannot access /usr/bin/docker-compose: No such file or directory
/usr/local/bin/docker-compose
[ec2-user@ip-172-31-32-47:~]$ ls -lh /usr/local/bin/docker-compose /usr/bin/docker-compose
ls: cannot access /usr/bin/docker-compose: No such file or directory
-rwxr-xr-x 1 root root 57M Oct 8 10:45 /usr/local/bin/docker-compose
[ec2-user@ip-172-31-32-47:~]$ sudo vi docker-compose.yml
[ec2-user@ip-172-31-32-47:~]$ sudo vi docker-compose.yaml
[ec2-user@ip-172-31-32-47:~]$ docker-compose up -d
[+] Running 35/2
  ✓ wordpress 21 layers [██████████████████]  0B/0B      Pulled          20.0s
  ✓ db 12 layers [██████████]    0B/0B      Pulled          17.0s
[+] Running 4/4
  ✓ Network ec2-user_default      Created          0.1s
  ✓ Volume "ec2-user_db_data"     Created          0.0s
  ✓ Container ec2-user-wordpress-1 Started          0.2s
  ✓ Container ec2-user-db-1       Started          0.2s
[ec2-user@ip-172-31-32-47:~]$
```

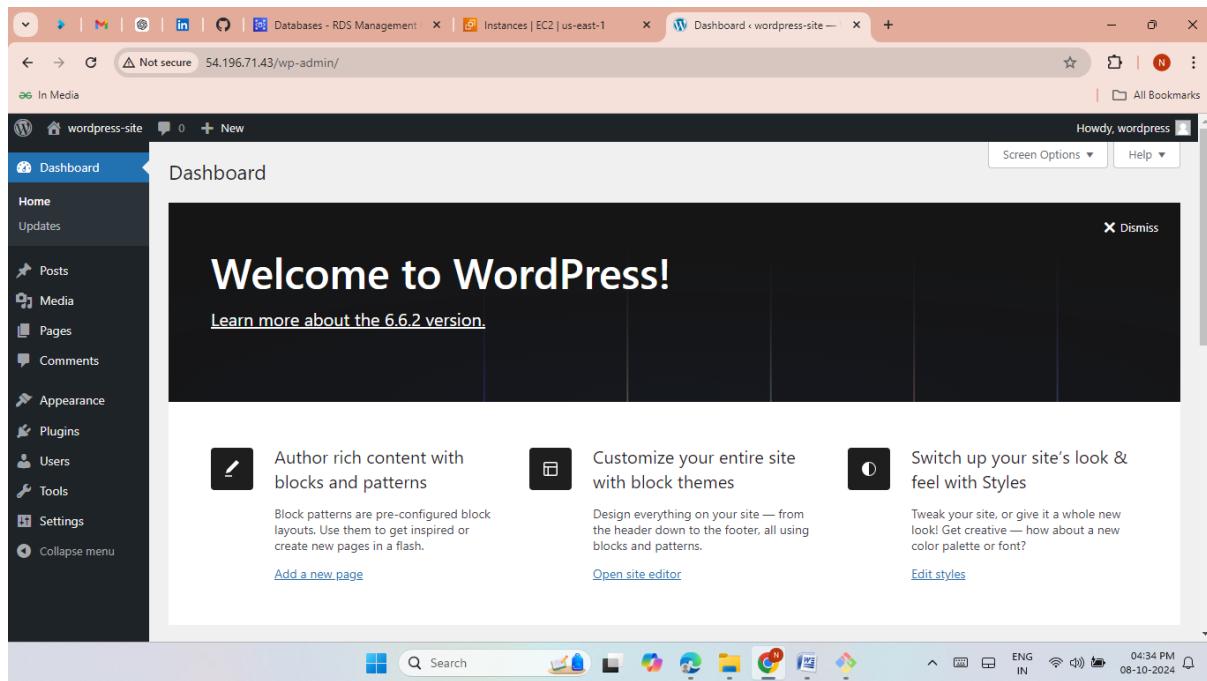
- Now see the containers of the pulled images of MYSQL and WORDPRESS by using a command as **docker ps**

- Now copy that static Public IP and past it in Google browser then select the language and click on continue.





- Here the wordpress is successfully launched.



- Now install git then push the docker-compose file in github.

```
ec2-user@ip-172-31-32-47:~$ Is this ok [y/d/N]: y
Downloading packages:
(1/6): git-2.40.1-1.amzn2.0.3.x86_64.rpm | 54 kB 00:00:00
(2/6): git-core-doc-2.40.1-1.amzn2.0.3.noarch.rpm | 3.0 MB 00:00:00
(3/6): git-core-2.40.1-1.amzn2.0.3.x86_64.rpm | 10 MB 00:00:00
(4/6): perl-Error-0.17020-2.amzn2.noarch.rpm | 32 kB 00:00:00
(5/6): perl-Git-2.40.1-1.amzn2.0.3.noarch.rpm | 42 kB 00:00:00
(6/6): perl-TermReadKey-2.30-20.amzn2.0.2.x86_64.rpm | 31 kB 00:00:00
Total] Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : git-core-2.40.1-1.amzn2.0.3.x86_64 1/6
  Installing : git-core-doc-2.40.1-1.amzn2.0.3.noarch 2/6
  Installing : perl-Error-0.17020-2.amzn2.noarch 3/6
  Installing : perl-TermReadKey-2.30-20.amzn2.0.2.x86_64 4/6
  Installing : perl-Git-2.40.1-1.amzn2.0.3.noarch 5/6
  Installing : git-2.40.1-1.amzn2.0.3.x86_64 6/6
  Verifying : perl-TermReadKey-2.30-20.amzn2.0.2.x86_64 1/6
  Verifying : git-2.40.1-1.amzn2.0.3.x86_64 2/6
  Verifying : perl-Error-0.17020-2.amzn2.noarch 3/6
  Verifying : git-core-2.40.1-1.amzn2.0.3.x86_64 4/6
  Verifying : git-core-doc-2.40.1-1.amzn2.0.3.noarch 5/6
  Verifying : perl-Git-2.40.1-1.amzn2.0.3.noarch 6/6
Installed:
  git.x86_64 0:2.40.1-1.amzn2.0.3
Dependency Installed:
  git-core.x86_64 0:2.40.1-1.amzn2.0.3           perl-core-doc.noarch 0:2.40.1-1.amzn2.0.3
  perl-git.noarch 0:2.40.1-1.amzn2.0.3          perl-TermReadKey.x86_64 0:2.30-20.amzn2.0.2
perl-Error.noarch 1:0.17020-2.amzn2
Complete!
[ec2-user@ip-172-31-32-47 ~]$
```

```
ec2-user@ip-172-31-32-47:/dockerfile$ sudo vi docker-compose.yaml
[ec2-user@ip-172-31-32-47 dockerfile]$ ls
docker-compose.yaml README.md
[ec2-user@ip-172-31-32-47 dockerfile]$ git add docker-compose.yaml
[ec2-user@ip-172-31-32-47 dockerfile]$ ls
docker-compose.yaml README.md
[ec2-user@ip-172-31-32-47 dockerfile]$ git commit -m docker-compose.yaml
[main 8edd354] docker-compose.yaml
Committer: EC2 Default User <ec2-user@ip-172-31-32-47.ec2.internal>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:
git config --global --edit
After doing this, you may fix the identity used for this commit with:
git commit --amend --reset-author
1 file changed, 28 insertions(+)
create mode 100644 docker-compose.yaml
[ec2-user@ip-172-31-32-47 dockerfile]$ ls
docker-compose.yaml README.md
[ec2-user@ip-172-31-32-47 dockerfile]$ git push
Username for 'https://github.com': yashwanth-najana
Password for 'https://yashwanth-najana@github.com':
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 596 bytes | 596.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Yashwanth-najana/dockerfile.git
  8edd354..8edd354 main -> main
[ec2-user@ip-172-31-32-47 dockerfile]$
```

➤ Here the docker-compose file is pushed.

The screenshot shows a Microsoft Edge browser window displaying a GitHub repository named 'dockerfile'. The repository details are as follows:

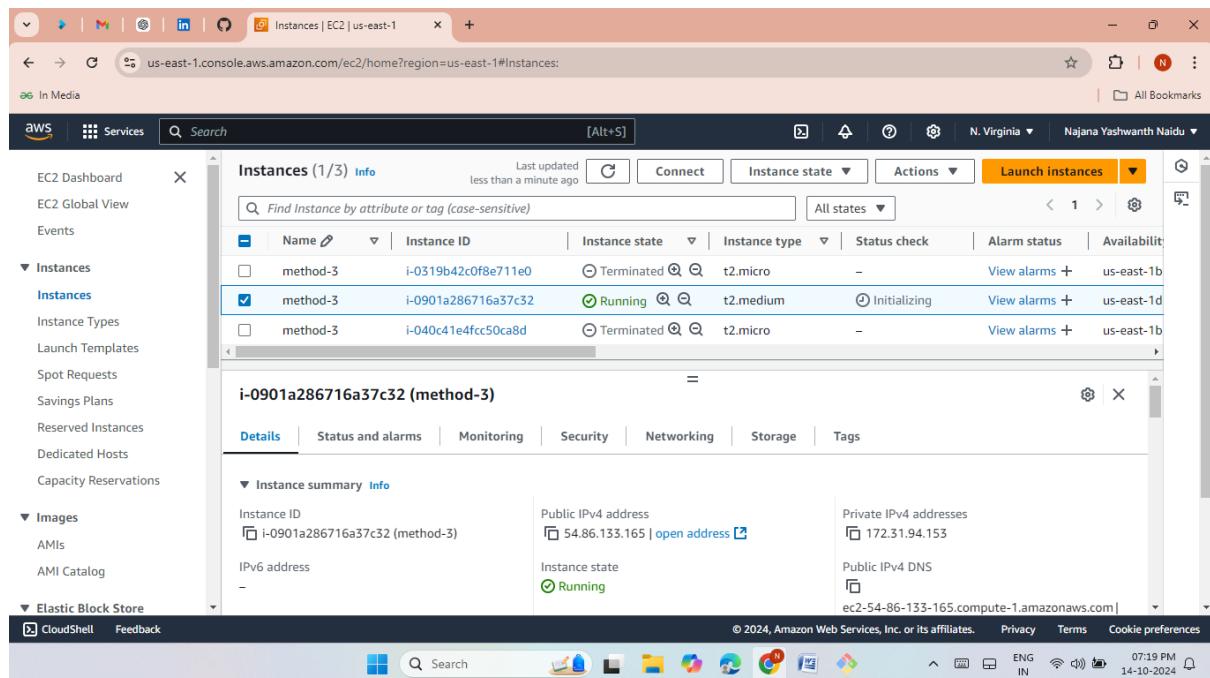
- Branches: 1
- Tags: 0
- Files:
 - README.md (Initial commit, 16 minutes ago)
 - docker-compose.yaml (1 minute ago)

The 'About' section indicates "No description, website, or topics provided." The 'Releases' section shows "No releases published" and a link to "Create a new release". The 'Packages' section is empty.

METHOD 3

Deploy WordPress web application by using git and jenkins

- Launch EC2 instance with Jenkins port number(8080).
- Here I have choose ubuntu.



- Now connect the remote server to local server.

```
ubuntu@ip-172-31-94-153:~$ 
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.8.0-1015-aws x86_64)

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

System information as of Mon Oct 14 13:50:23 UTC 2024

System load: 0.25      Processes: 122
Usage of /: 21.1% of 7.57GB   Users logged in: 0
Memory usage: 6%          IPv4 address for eth0: 172.31.94.153
Swap usage: 0%          

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

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

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-94-153:~$
```

- Now install Jenkins with the following commands

Sudo apt update

```
sudo apt install openjdk-17-jre-headless
```

```
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
```

```
echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] " \
https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
```

```
/etc/apt/sources.list.d/jenkins.list > /dev/null
```

```
sudo apt-get update
```

```
sudo apt-get install Jenkins
```

- No need to start and enable manually by choosing an ubuntu.

```
ubuntu@ip-172-31-94-153:~$ Created Symlink /etc/systemd/system/multi-user.target.wants/jenkins.service → /lib/systemd/system/jenkins.service.
Processing triggers for man-db (2.10.2-1) ...
Scanning processes...
Scanning linux images...
Running kernel seems to be up-to-date.
No services need to be restarted.
No containers need to be restarted.
No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-94-153:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
  Loaded: Loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
  Active: active (running) since Mon 2024-10-14 13:52:32 UTC; 1s ago
    Main PID: 4142 (java)
      Tasks: 52 (limit: 4676)
     Memory: 501.3M
        CPU: 16.366s
       CGroup: /system.slice/jenkins.service
              └─4142 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Oct 14 13:52:29 ip-172-31-94-153 jenkins[4142]: 654f4b05381242d8a84b1beb967af577
Oct 14 13:52:29 ip-172-31-94-153 jenkins[4142]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Oct 14 13:52:29 ip-172-31-94-153 jenkins[4142]: ****
Oct 14 13:52:29 ip-172-31-94-153 jenkins[4142]: ****
Oct 14 13:52:32 ip-172-31-94-153 jenkins[4142]: 2024-10-14 13:52:32.643+0000 [id:33] INFO jenkins.InitReactorRunner$1@o
Oct 14 13:52:32 ip-172-31-94-153 jenkins[4142]: 2024-10-14 13:52:32.661+0000 [id:24] INFO hudson.lifecycle.Lifecycle#onB
Oct 14 13:52:32 ip-172-31-94-153 systemd[1]: Started Jenkins Continuous Integration Server.
Oct 14 13:52:32 ip-172-31-94-153 jenkins[4142]: 2024-10-14 13:52:32.853+0000 [id:49] INFO h.m.DownloadService$Downloadad
Oct 14 13:52:32 ip-172-31-94-153 jenkins[4142]: 2024-10-14 13:52:32.853+0000 [id:49] INFO hudson.util.Retrier#start: Pe
Lines: 1-20/20 (END)
```

- Now install docker and docker-compose and then give permissions.

```
ubuntu@ip-172-31-94-153:~$ Unpacking ubuntu-fan (0.12.16) ...
Setting up dnsmasq-base (2.90-0ubuntu0.22.04.1) ...
Setting up runc (1.12-0ubuntu2~22.04.1) ...
Setting up dns-root-data (2023112702~ubuntu0.22.04.1) ...
Setting up bridge-utils (1.7~ubuntu3) ...
Setting up ifupdown (6-1) ...
Setting up containerd (1.7.12~ubuntu2~22.04.1) ...
Created Symlink /etc/systemd/system/multi-user.target.wants/containerd.service → /lib/systemd/system/containerd.service.
Setting up ubuntu-fan (0.12.16)
Created Symlink /etc/systemd/system/multi-user.target.wants/ubuntu-fan.service → /lib/systemd/system/ubuntu-fan.service.
Setting up docker.io (24.0.7~ubuntu2~22.04.1) ...
Adding group 'docker' (GID 123) ...
Done.
Created Symlink /etc/systemd/system/multi-user.target.wants/docker.service → /lib/systemd/system/docker.service.
Created Symlink /etc/systemd/system/sockets.target.wants/docker.socket → /lib/systemd/system/docker.socket.
Processing triggers for dbus (1.12.20-2ubuntu4.1) ...
Processing triggers for man-db (2.10.2-1) ...
Scanning processes...
Scanning linux images...
Running kernel seems to be up-to-date.
No services need to be restarted.
No containers need to be restarted.
No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-94-153:~$ docker ps
permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get "http://127.0.0.1:2375/v1.24/containers/json": dial unix /var/run/docker.sock: connect: permission denied
ubuntu@ip-172-31-94-153:~$ sudo chmod 666 /var/run/docker.sock
CONTAINER ID   IMAGE   COMMAND   CREATED   STATUS    PORTS   NAMES
ubuntu@ip-172-31-94-153:~|
```

```
ubuntu@ip-172-31-94-153:~$ sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
% Total    % Received % Xferd  Average Speed   Time   Time     Time  Current
          Dload  Upload Total Spent    Left Speed
0       0      0      0      0      0      0 --:--:-- --:--:-- --:--:-- 0
100 12.1M  100 12.1M    0      0  32.8M    0 --:--:-- --:--:-- --:--:-- 32.8M
ubuntu@ip-172-31-94-153:~$ sudo chmod +x /usr/local/bin/docker-compose
ubuntu@ip-172-31-94-153:~$ sudo chmod +x /usr/local/bin/docker-compose
ubuntu@ip-172-31-94-153:~$ docker-compose --version
docker-compose version 1.29.2, build 5becea4c
ubuntu@ip-172-31-94-153:~$
```

➤ No need to install git manually by choosing ubuntu.

```
ubuntu@ip-172-31-94-153:~$ git help -a
Clone      Clone a repository into a new directory
init       Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
add        Add file contents to the index
mv        Move or rename a file, a directory, or a symlink
restore    Restore working tree files
rm        Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
bisect    Use binary search to find the commit that introduced a bug
diff      Show changes between commits, commit and working tree, etc
grep      Print lines matching a pattern
log       Show commit logs
show      Show various types of objects
status    Show the working tree status

grow, mark and tweak your common history
branch   List, create, or delete branches
commit   Record changes to the repository
merge    Join two or more development histories together
rebase   Reapply commits on top of another base tip
reset   Reset current HEAD to the specified state
switch   Switch branches
tag      Create, list, delete or verify a tag object signed with GPG

collaborate (See also: git help workflows)
fetch    Download objects and refs from another repository
pull     Fetch from and integrate with another repository or a local branch
push     Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
See 'git help git' for an overview of the system.
ubuntu@ip-172-31-94-153:~$
```

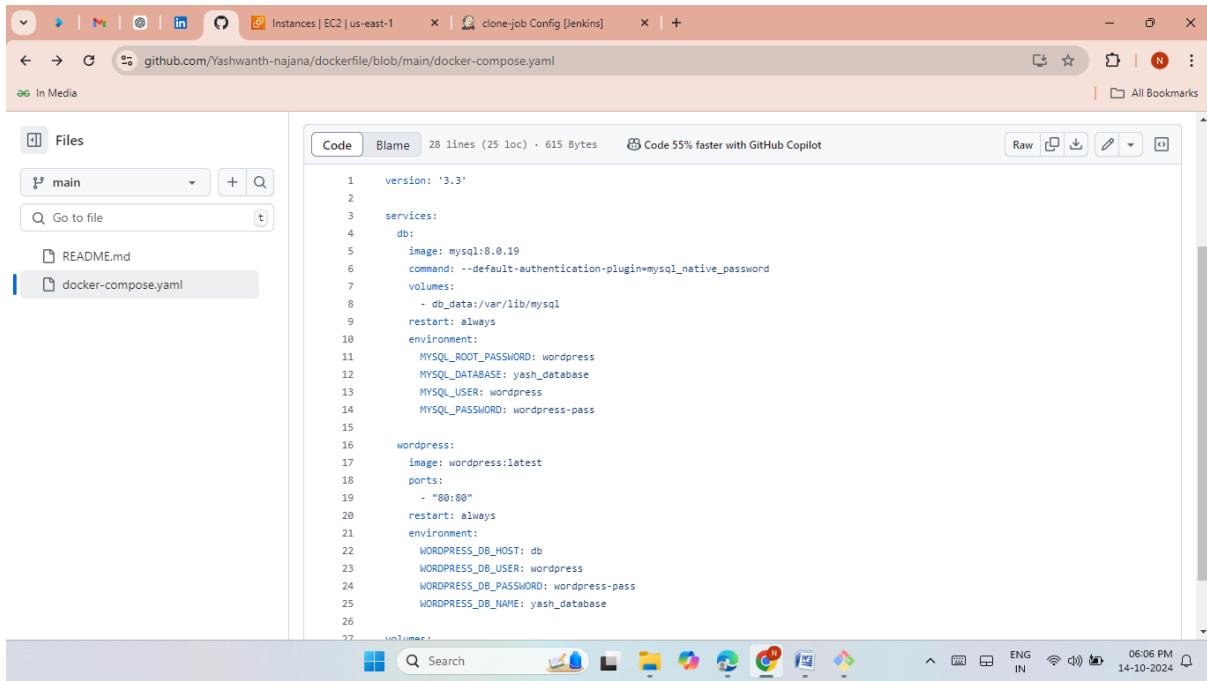
- Now copy that static Public IP and past it in Google browser.
- We can get the Jenkins.

The screenshot shows the Jenkins dashboard. At the top, there's a navigation bar with links for 'Instances | EC2 | us-east-1', 'Dashboard [Jenkins]', and other options. Below the bar, the address bar shows 'Not secure 54.86.133.165:8080'. The main content area has a dark header with the Jenkins logo and the word 'Jenkins'. It features several cards: 'Welcome to Jenkins!' (with a note about starting builds), 'Start building your software project' (with 'Create a job' and '+'), 'Set up a distributed build' (with 'Set up an agent' and 'Configure a cloud'), and 'Build Queue' (empty). On the right, there's a sidebar with 'Build Executor Status' showing 1 Idle and 2 Idle. The bottom of the screen shows a Windows taskbar with various icons and system status.

- Now create a clone-job for cloning the repo from github.

The screenshot shows the 'New Item' creation page in Jenkins. The title bar says 'New Item [Jenkins]'. The main form has 'Enter an item name' with 'clone-job' typed in. Below it, 'Select an item type' has 'Freestyle project' selected, with a description of it being a classic general-purpose job type. Other options like 'Pipeline' and 'Multi-configuration project' are also listed. At the bottom is a blue 'OK' button. The bottom of the screen shows a Windows taskbar with various icons and system status.

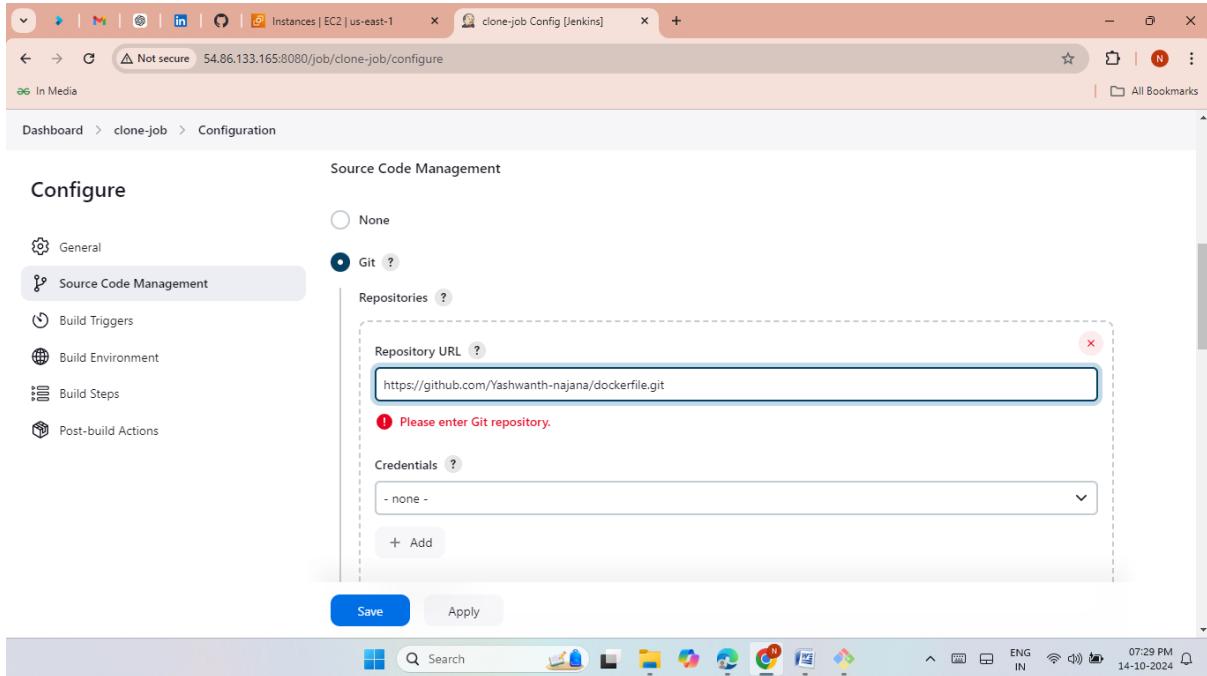
➤ This is the docker-compose file from github.



The screenshot shows a Microsoft Edge browser window. The address bar displays 'github.com/Yashwanth-najana/dockerfile/blob/main/docker-compose.yaml'. The main content area is a GitHub code editor showing a docker-compose.yaml file. The file defines two services: 'db' (MySQL 8.0.19) and 'wordpress' (WordPress latest). They are linked via environment variables like WORDPRESS_DB_HOST and WORDPRESS_DB_USER. The 'Code' tab is selected, showing 28 lines of code with 615 bytes. A GitHub Copilot button is visible at the top right. On the left sidebar, there are links for 'main', 'README.md', and 'docker-compose.yaml'. The system tray at the bottom right shows the date as 14-10-2024 and the time as 06:06 PM.

```
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: yash_database
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: wordpress-pass
  wordpress:
    image: wordpress:latest
    ports:
      - "80:80"
    restart: always
    environment:
      WORDPRESS_DB_HOST: db
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress-pass
      WORDPRESS_DB_NAME: yash_database
volumes:
```

➤ Now Clone the repository URL.



The screenshot shows a Jenkins configuration page for a job named 'clone-job'. The 'Source Code Management' section is active. It shows a 'Git' configuration with a 'Repository URL' field containing 'https://github.com/Yashwanth-najana/dockerfile.git'. A red error message 'Please enter Git repository.' is displayed below the field. Under 'Credentials', there is a dropdown menu set to '- none -' and a '+ Add' button. At the bottom are 'Save' and 'Apply' buttons. The system tray at the bottom right shows the date as 14-10-2024 and the time as 07:29 PM.

➤ Now click on build now. it shows success.

The screenshot shows the Jenkins interface for the 'clone-job' project. The top navigation bar includes links for 'Instances | EC2 | us-east-1', 'clone-job [Jenkins]', and 'All Bookmarks'. The main content area displays the 'clone-job' project details. On the left, there's a sidebar with options like 'Status', 'Changes', 'Workspace', 'Build Now' (which is highlighted with a red border), 'Configure', 'Delete Project', and 'Rename'. The 'clone-job' name is prominently displayed at the top. Below it, a description says 'clone the docker file and launch the wordpress'. A 'Permalinks' section is also present. The 'Build History' section shows one build entry: '#1 14-Oct-2024, 2:21 pm'. The status of this build is 'Success'. At the bottom right, there's a toolbar with icons for search, file operations, and system status.

➤ Now create a deploy job and add existing clone-job.

The screenshot shows the Jenkins 'New Item' creation dialog. The title bar says 'New Item [Jenkins]'. The main content area shows two options: 'Multibranch Pipeline' and 'Organization Folder'. Below these, a note says: '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.' Underneath, there's another note: 'Creates a set of Pipeline projects according to detected branches in one SCM repository.' and 'Creates a set of multibranch project subfolders by scanning for repositories.' At the bottom, there's a note: 'If you want to create a new item from other existing, you can use this option:' followed by a 'Copy from' field containing 'clone-job'. A blue 'OK' button is visible at the bottom left. The bottom right corner shows the Jenkins version 'Jenkins 2.462.3'.

➤ Now write the script in execute shell.

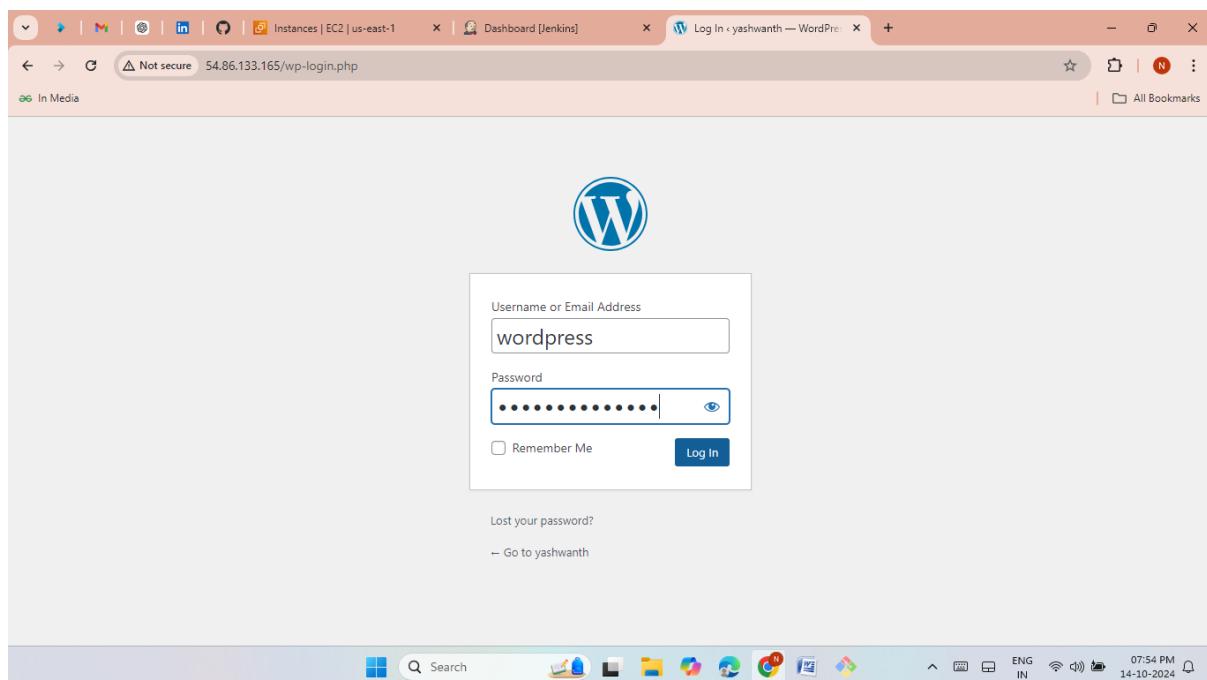
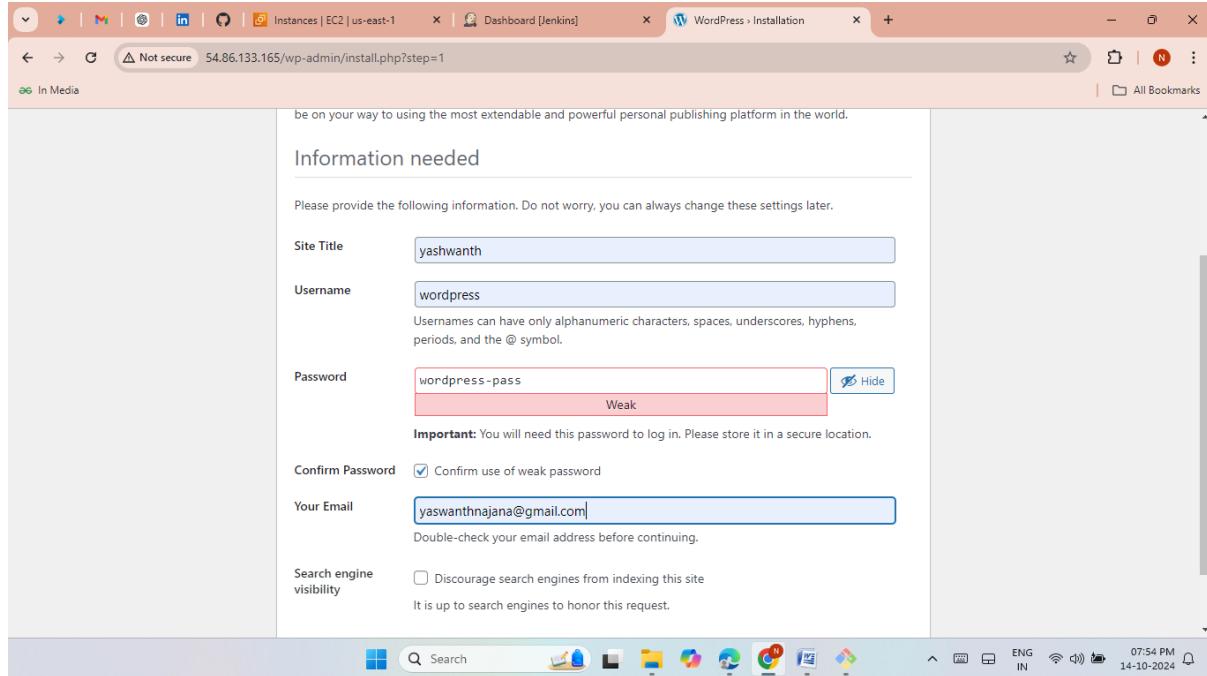
```
docker-compose up -d
```

The screenshot shows the Jenkins job configuration page for 'deploy-job'. The 'Build Steps' section is selected. It contains one step named 'Execute shell' with the command 'docker-compose up -d' entered. There are also two optional checkboxes: 'Terminate a build if it's stuck' and 'With Ant'.

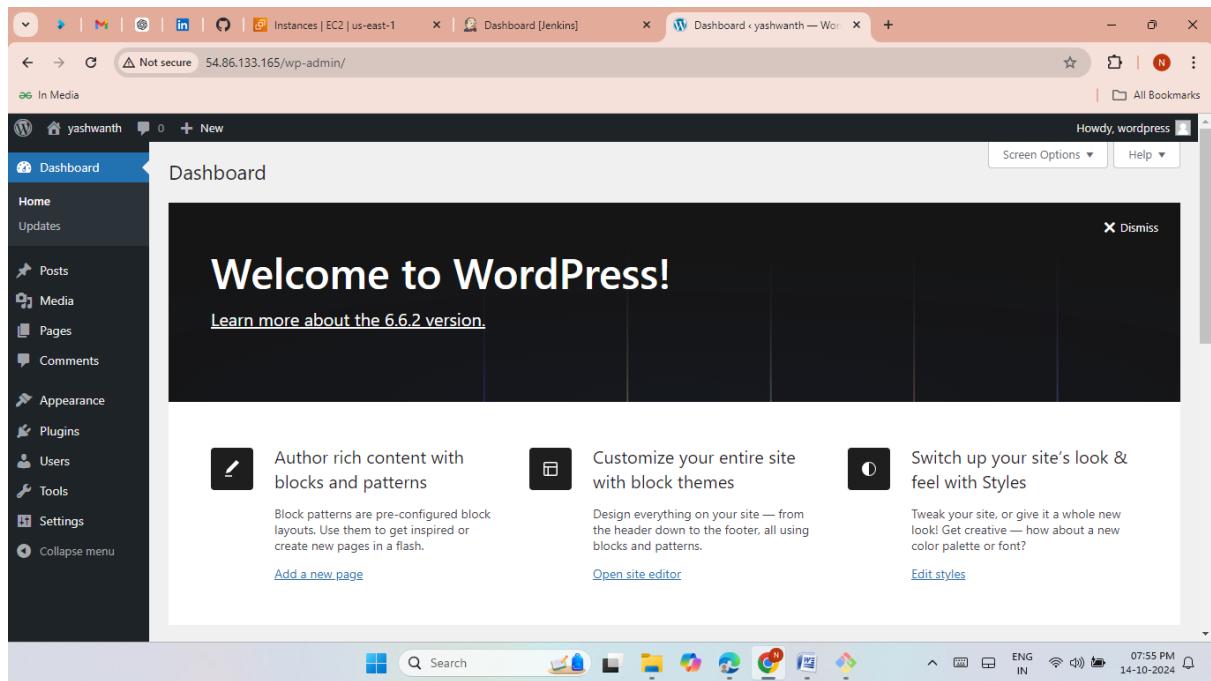
➤ Now it shows job is success.

The screenshot shows the Jenkins dashboard for the 'deploy-job'. The status is listed as 'Success'. Below the status, there is a 'Build History' section showing one build from '14-Oct-2024, 2:23 pm'. The 'Build Now' button is highlighted with a red box.

- Now copy that Static Public IP and past it in Google browser then select the language and click on continue.



- Here the wordpress launched successfully.



METHOD-4

Deploy WordPress web application by using userdata of EC2 instance

- Write the script in ec2 instance.

```
#!/bin/bash
yum update -y
yum install -y git
git clone https://github.com/Yashwanth-najana/dockerfile.git
cd dockerfile/
yum install -y docker
systemctl start docker
systemctl enable docker
chmod 666 /var/run/docker.sock
DOCKER_COMPOSE_VERSION="v2.20.0"
```

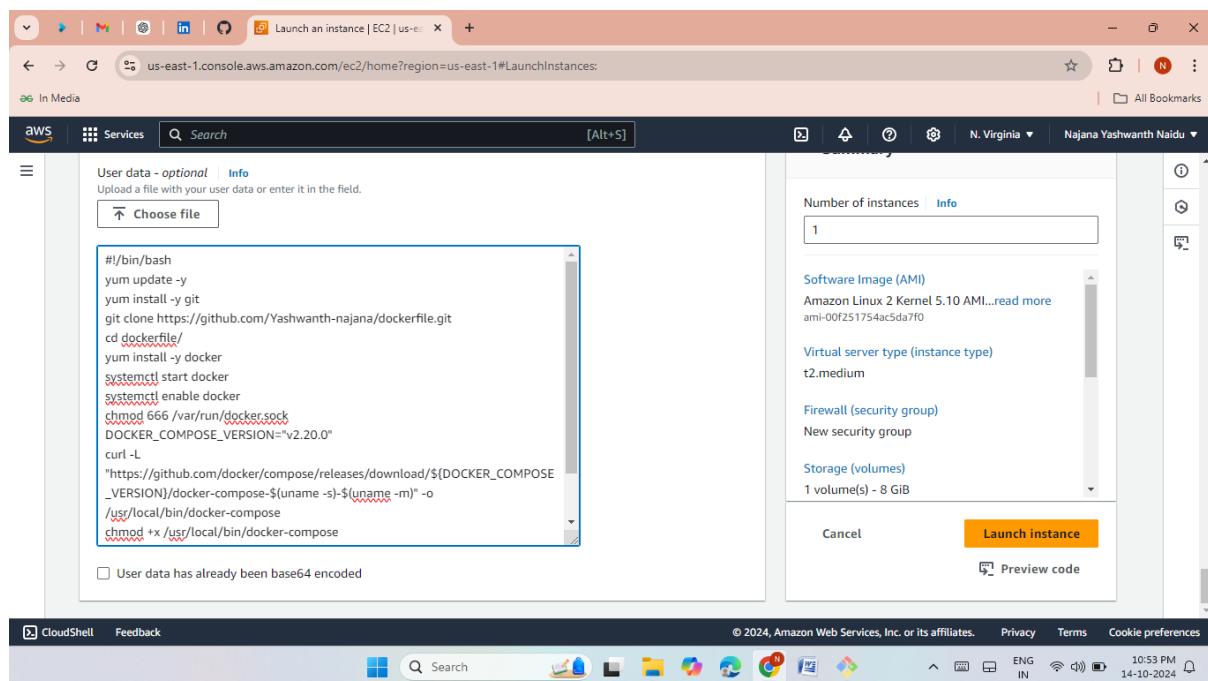
```
curl -L
```

```
"https://github.com/docker/compose/releases/download/${DOCKER_COMPOSE_VERSION}/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose  
chmod +x /usr/local/bin/docker-compose
```

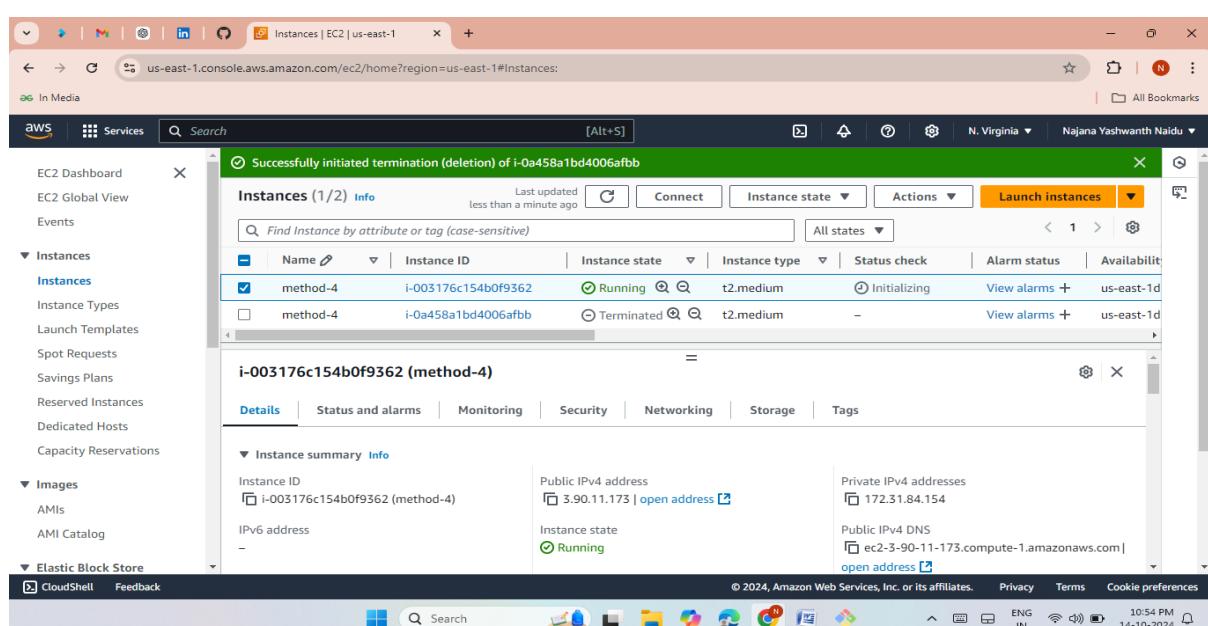
```
ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
```

```
docker-compose up -d
```

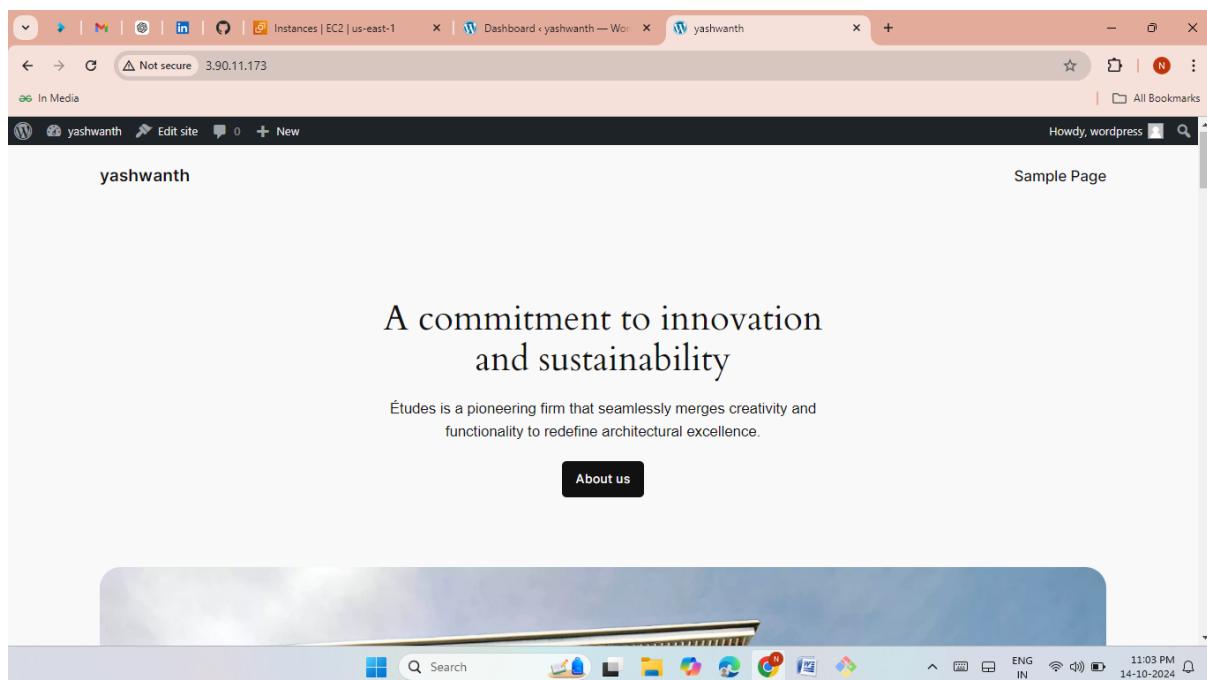
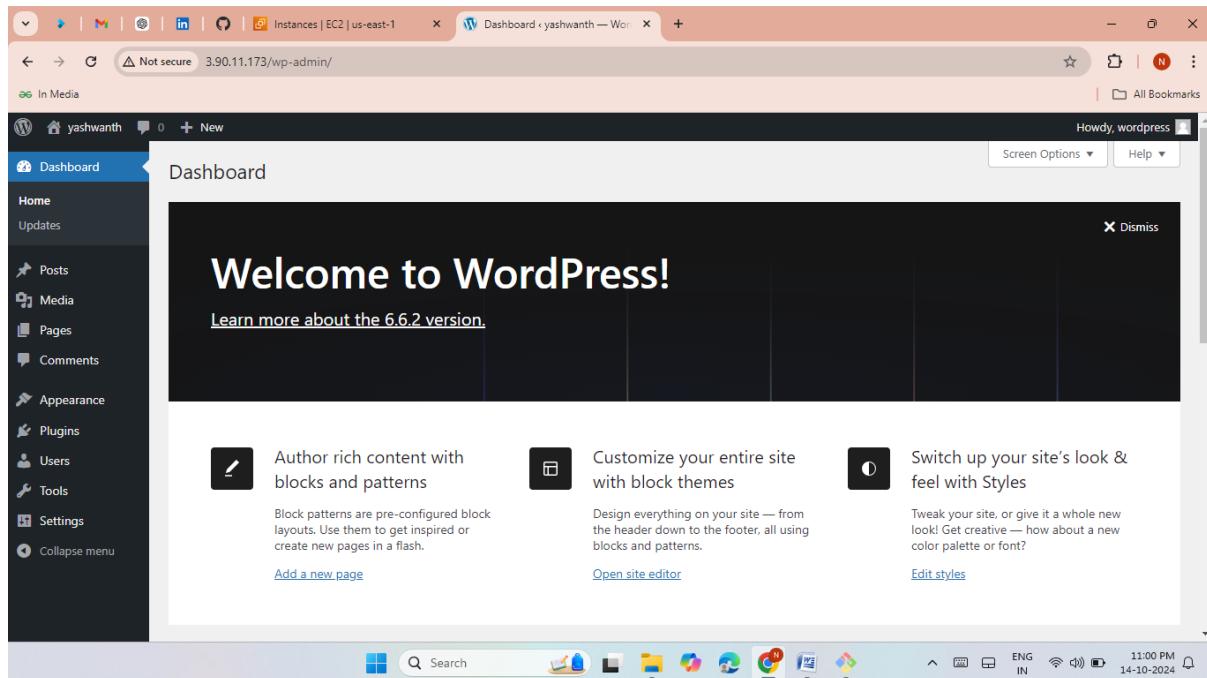
```
docker ps
```



- Here the instance was launched with the user data.



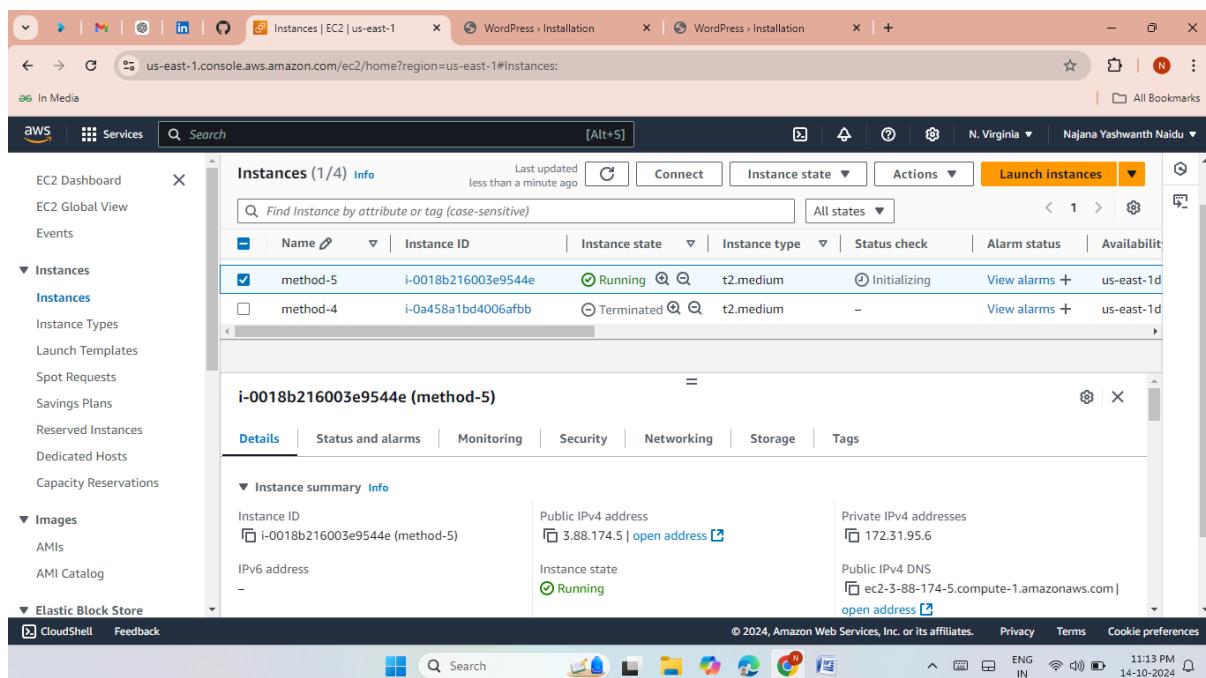
- Now copy that Static Public IP and past it in browser.
- Here the wordpress was launched successfully.



METHOD-5

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

- Launch an EC2 instance with the jenkins port number.



- Now connect the server locally.

```
ec2-user@ip-172-31-95-6:~$ yaswa@LAPTOP-GM32QJHO MINGW64 ~
$ cd keya
bash: cd: keya: No such file or directory
yaswa@LAPTOP-GM32QJHO MINGW64 ~
$ cd keys
yaswa@LAPTOP-GM32QJHO MINGW64 ~/keys
$ ssh -i "yash.pem" ec2-user@ec2-3-88-174-5.compute-1.amazonaws.com
The authenticity of host 'ec2-3-88-174-5.compute-1.amazonaws.com (3.88.174.5)' can't be established.
ED25519 key fingerprint is SHA256:kfyjio4MgF0l6gdkdyb/Rq2wX7tCRxTorfDK9gRwwqo.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-3-88-174-5.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-172-31-95-6 ~]$ |
```

➤ Now install Jenkins and then start and enable.

```
ec2-user@ip-172-31-95-6:~$ sudo yum install jenkins
Complete!
[ec2-user@ip-172-31-95-6 ~]$ sudo yum install jenkins
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
Is this ok [y/d/N]: y
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
                                                1/1
                                                1/1

Installed:
jenkins.noarch 0:2.462.3-1.1

Complete!
[ec2-user@ip-172-31-95-6 ~]$ sudo systemctl daemon-reload
[ec2-user@ip-172-31-95-6 ~]$
```

```
ec2-user@ip-172-31-95-6:~$ sudo yum upgrade
libxext.x86_64 0:1.3.3-3.amzn2.0.2
libxinerama.x86_64 0:1.1.3-2.1.amzn2.0.2
libxrender.x86_64 0:0.9.10-1.amzn2.0.2
libxtst.x86_64 0:1.2.3-1.amzn2.0.2
libxslt.x86_64 0:1.1.28-6.amzn2
python-javapackages.noarch 0:3.4.1-11.amzn2
libxi.x86_64 0:1.7.9-1.amzn2.0.2
libxrandr.x86_64 0:1.5.1-2.amzn2.0.3
libxt.x86_64 0:1.1.5-3.amzn2.0.2
libxcb.x86_64 0:1.12-1.amzn2.0.2
log4j-cve-2021-44228-hotpatch.noarch 0:1.3-7.amzn2
python-lxml.x86_64 0:3.2.1-4.amzn2.0.6

Complete!
[ec2-user@ip-172-31-95-6 ~]$ sudo yum upgrade
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
No packages marked for update
[ec2-user@ip-172-31-95-6 ~]$ ^C
[ec2-user@ip-172-31-95-6 ~]$ sudo yum install jenkins
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Package jenkins-2.462.3-1.1.noarch already installed and latest version
Nothing to do
[ec2-user@ip-172-31-95-6 ~]$ sudo systemctl start jenkins
[ec2-user@ip-172-31-95-6 ~]$ sudo systemctl enable jenkins
[ec2-user@ip-172-31-95-6 ~]$ 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 2024-10-14 17:51:15 UTC; 15s ago
    Main PID: 4360 (java)
   CGroup: /system.slice/jenkins.service
          └─4360 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=%C/jenkins/war --httpPort=8080

Oct 14 17:51:12 ip-172-31-95-6.ec2.internal jenkins[4360]: e57639500e4f4b759206fe84a92b6254
Oct 14 17:51:12 ip-172-31-95-6.ec2.internal jenkins[4360]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Oct 14 17:51:12 ip-172-31-95-6.ec2.internal jenkins[4360]: ****
Oct 14 17:51:12 ip-172-31-95-6.ec2.internal jenkins[4360]: ****
Oct 14 17:51:12 ip-172-31-95-6.ec2.internal jenkins[4360]: ****
Oct 14 17:51:15 ip-172-31-95-6.ec2.internal jenkins[4360]: 2024-10-14 17:51:15.370+0000 [id=49]      INFO      h.m.Downlo...aller
Oct 14 17:51:15 ip-172-31-95-6.ec2.internal jenkins[4360]: 2024-10-14 17:51:15.371+0000 [id=49]      INFO      hudson.util..pt #1
Oct 14 17:51:15 ip-172-31-95-6.ec2.internal jenkins[4360]: 2024-10-14 17:51:15.396+0000 [id=31]      INFO      jenkins.Ini...ation
Oct 14 17:51:15 ip-172-31-95-6.ec2.internal jenkins[4360]: 2024-10-14 17:51:15.406+0000 [id=24]      INFO      hudson.life...ning
Oct 14 17:51:15 ip-172-31-95-6.ec2.internal systemd[1]: Started Jenkins Continuous Integration Server.
Hint: Some lines were ellipsized, use -l to show in full.
[ec2-user@ip-172-31-95-6 ~]$
```

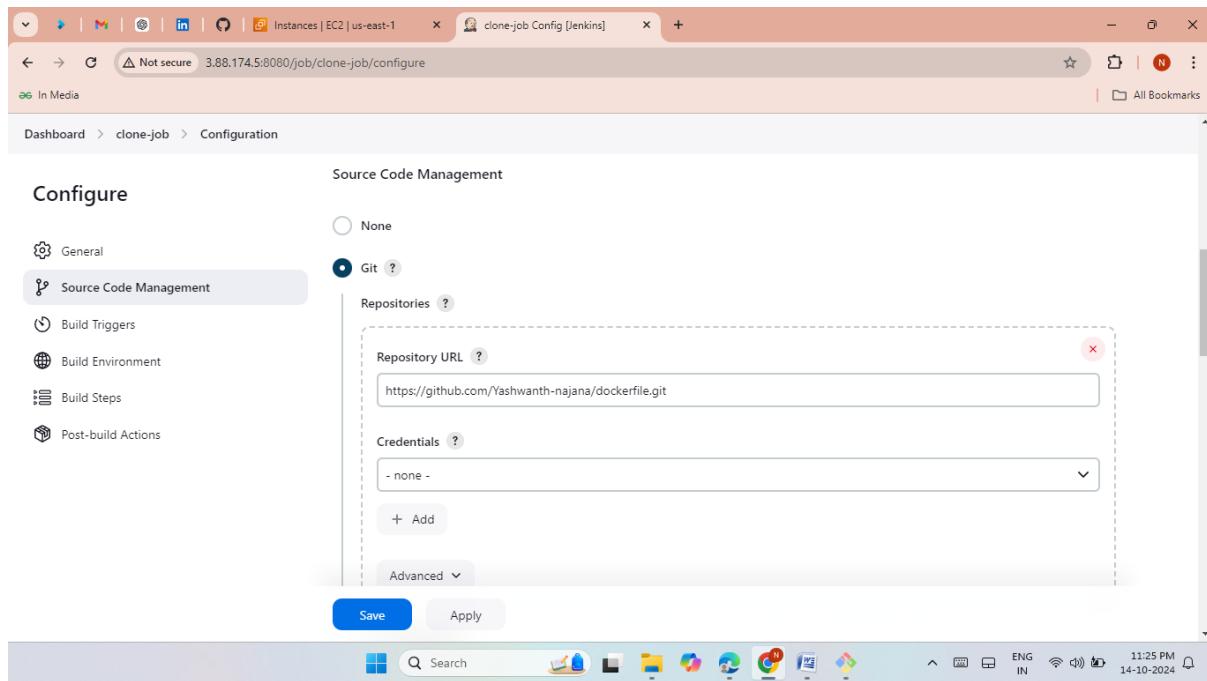
➤ Here the Jenkins was launched.

The screenshot shows the Jenkins dashboard at 3.88.174.5:8080. The main heading is "Welcome to Jenkins!". Below it, a message says: "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." On the left, there's a sidebar with links: "New Item", "Build History", "Manage Jenkins", and "My Views". Under "Build Queue", it says "No builds in the queue.". Under "Build Executor Status", it lists "1 Idle" and "2 Idle". In the center, there are three main sections: "Create a job" (with a plus icon), "Set up a distributed build" (with "Set up an agent" and "Configure a cloud" options), and "Learn more about distributed builds" (with a question mark icon). The bottom status bar shows system information like battery level, signal strength, and the date/time (14-10-2024).

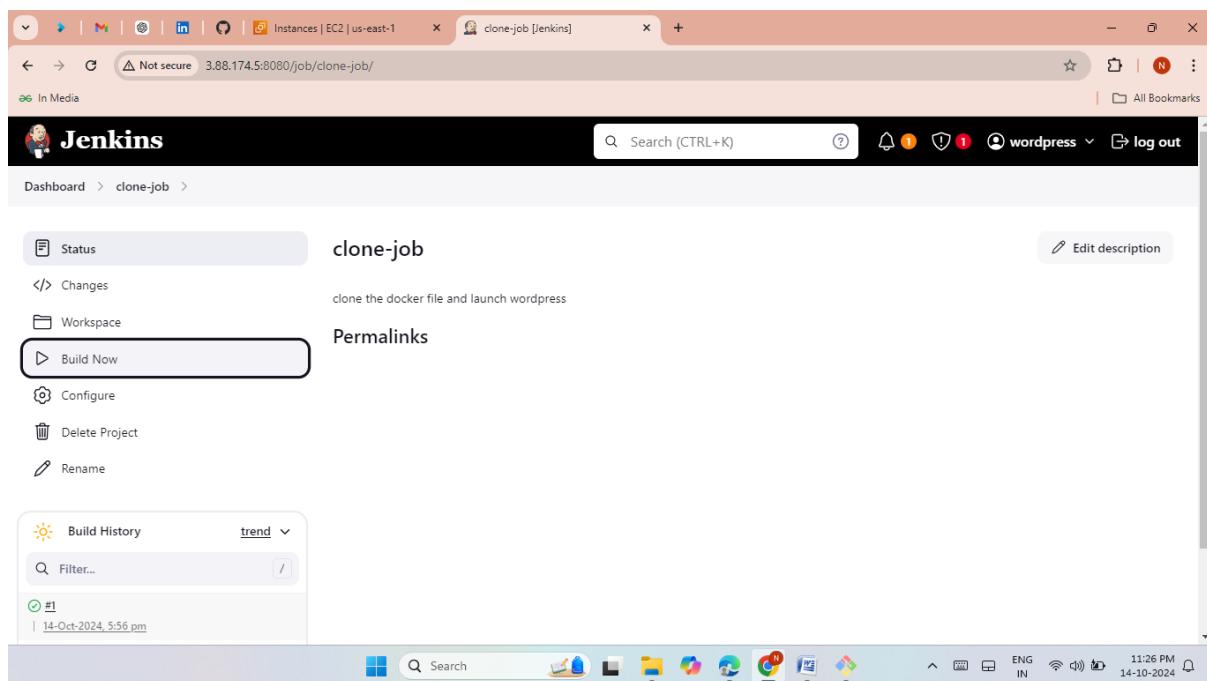
➤ Create a clone job for cloned the repo.

The screenshot shows the "New Item" creation dialog in Jenkins. The title is "New Item". A search bar contains the text "clone-job". Below it, a section titled "Select an item type" shows two options: "Freestyle project" (selected) and "Pipeline". The "Freestyle project" description reads: "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." The "Pipeline" description reads: "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." At the bottom, there is an "OK" button.

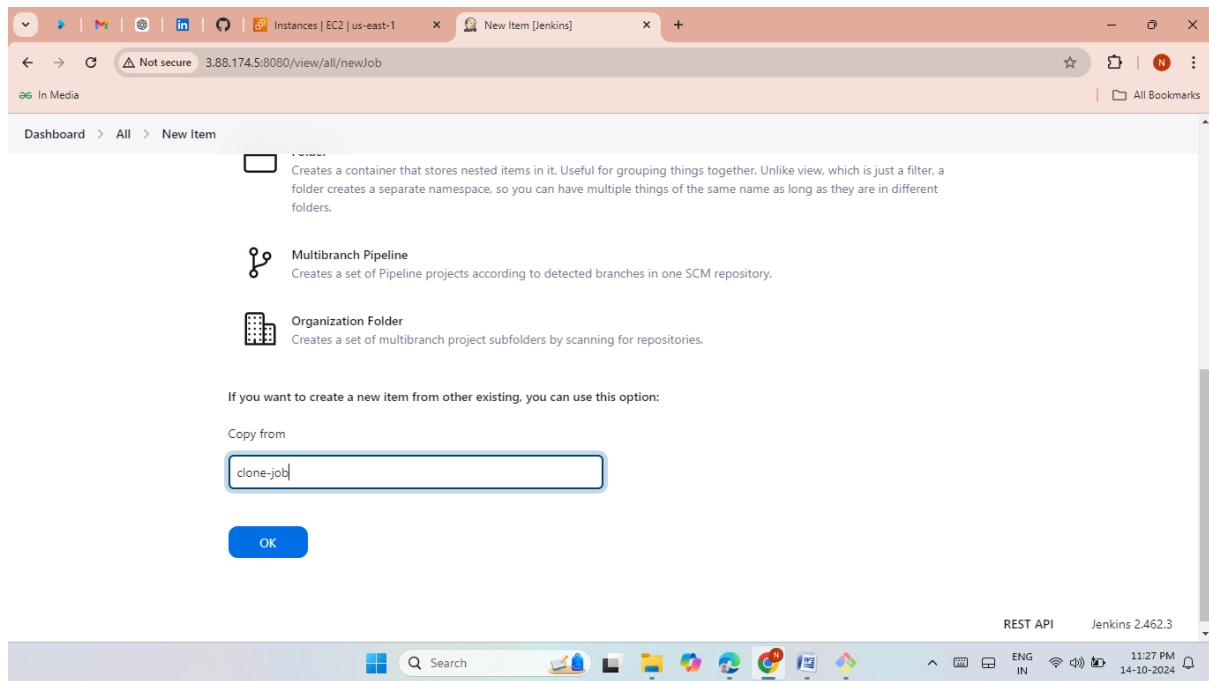
➤ Now Clone the repository URL.



➤ Here the build is success.



- Now Create the deploy job and add it to the clone job.



- Now write the script in execute shell.

```
#!/bin/bash

set -e

sudo yum update -y

sudo yum install -y docker

sudo systemctl start docker

sudo systemctl enable docker

sudo usermod -aG docker jenkins

sudo chmod 666 /var/run/docker.sock

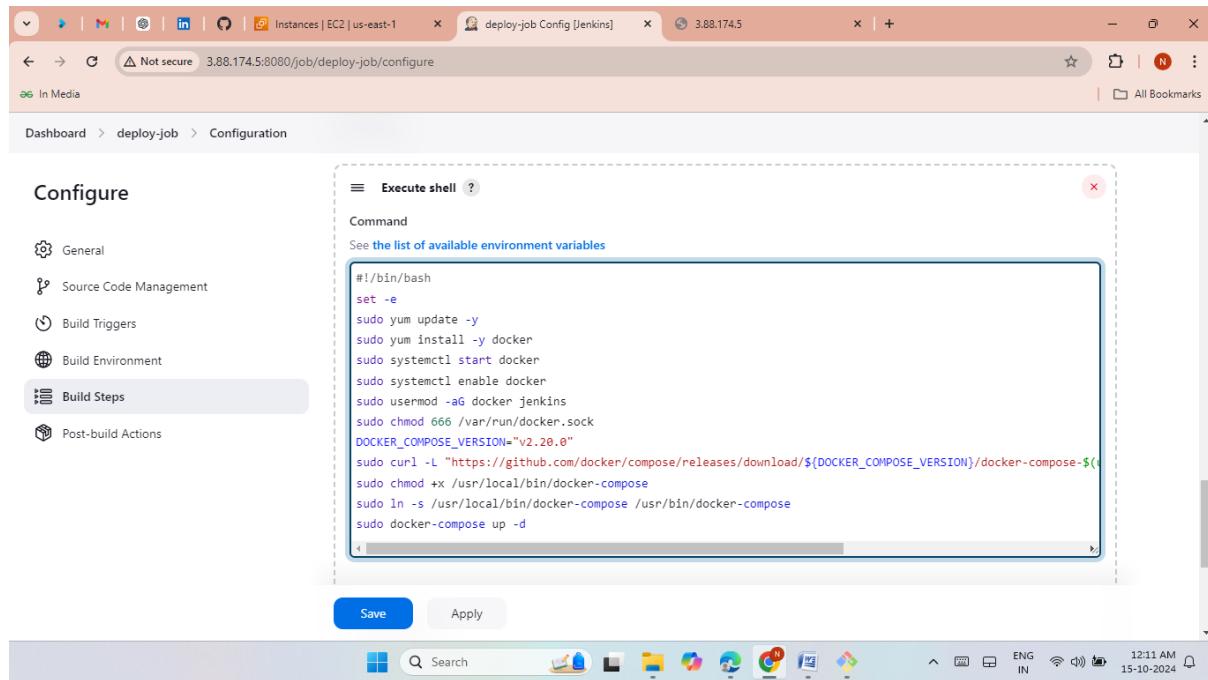
DOCKER_COMPOSE_VERSION="v2.20.0"

sudo curl -L
"https://github.com/docker/compose/releases/download/${DOCKER_COMPOSE_VERSION}/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

sudo chmod +x /usr/local/bin/docker-compose

sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose

sudo docker-compose up -d
```



- Edit the **sudoers** File:
- You need to add the Jenkins user to the **sudoers** file to allow it to execute commands without a password.

Sudo visudo

- Add the Following Line->At the end of the file

jenkins ALL=(ALL) NOPASSWD: ALL

- you want to give the Jenkins user elevated privileges for Docker-related commands. you can add the Jenkins user to the **docker** group.
- Run the following commands.

sudo usermod -aG docker Jenkins

sudo systemctl restart Jenkins

- Now it shows build is success.

The screenshot shows the Jenkins interface for the 'deploy-job' job. The job is currently scheduled for build. The 'Build History' section displays the last 18 builds, all of which were successful. The Jenkins logo is at the top left, and the URL is 3.88.174.5:8080/job/deploy-job/.

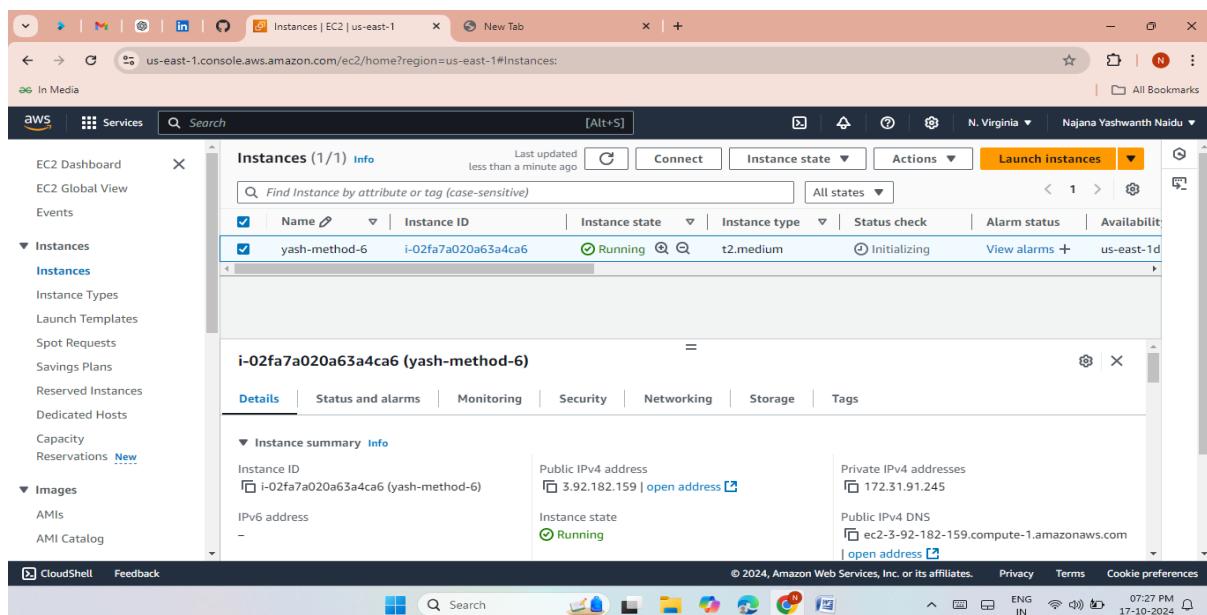
- Now copy that static Public IP and past it in browser.
- Here the wordpress was launched successfully.

The screenshot shows the WordPress dashboard. The dashboard features a 'Welcome to WordPress!' message and three main sections: 'Author rich content with blocks and patterns', 'Customize your entire site with block themes', and 'Switch up your site's look & feel with Styles'. The left sidebar shows navigation links like Home, Posts, Media, Pages, Comments, Appearance, Plugins, Users, Tools, and Settings. The URL is 3.88.174.5/wp-admin/.

METHOD-6

**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.**

- Now launch an EC2 instances with the Jenkins port number and ubuntu AMI.



- Here no need to Install git in ubuntu.
- Now install Jenkins then no need to start and enable manually in ubuntu.

```
ubuntu@ip-172-31-91-245:~  
Created symlink /etc/systemd/system/multi-user.target.wants/jenkins.service → /lib/systemd/system/jenkins.service.  
Processing triggers for man-db (2.10.2-1) ...  
Scanning processes...  
Scanning linux images...  
  
Running kernel seems to be up-to-date.  
No services need to be restarted.  
No containers need to be restarted.  
No user sessions are running outdated binaries.  
  
No VM guests are running outdated hypervisor (qemu) binaries on this host.  
ubuntu@ip-172-31-91-245:~$ sudo systemctl status jenkins  
● jenkins.service - Jenkins Continuous Integration Server  
   Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)  
   Active: active (running) since Thu 2024-10-17 14:01:21 UTC; 17s ago  
     Main PID: 4173 (java)  
        Tasks: 48 (limit: 4676)  
       Memory: 663.2M  
          CPU: 13.670s  
        CGroup: /system.slice/jenkins.service  
               └─4173 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080  
  
Oct 17 14:01:19 ip-172-31-91-245 jenkins[4173]: 93d45dd7748b47ab95ed754a55fdc809  
Oct 17 14:01:19 ip-172-31-91-245 jenkins[4173]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword  
Oct 17 14:01:19 ip-172-31-91-245 jenkins[4173]: ****  
Oct 17 14:01:19 ip-172-31-91-245 jenkins[4173]: ****  
Oct 17 14:01:19 ip-172-31-91-245 jenkins[4173]: ****  
Oct 17 14:01:21 ip-172-31-91-245 jenkins[4173]: 2024-10-17 14:01:21.952+0000 [id=30]      INFO    jenkins.InitReactorRunner$1#on  
Oct 17 14:01:21 ip-172-31-91-245 jenkins[4173]: 2024-10-17 14:01:21.974+0000 [id=23]      INFO    hudson.lifecycle.Lifecycle#on  
Oct 17 14:01:22 ip-172-31-91-245 systemd[1]: Started Jenkins Continuous Integration Server.  
Oct 17 14:01:22 ip-172-31-91-245 jenkins[4173]: 2024-10-17 14:01:22.038+0000 [id=48]      INFO    h.m.DownloadService$Downloader#  
Oct 17 14:01:22 ip-172-31-91-245 jenkins[4173]: 2024-10-17 14:01:22.038+0000 [id=48]      INFO    hudson.util.Retriger#start: PE  
Lines 1-20/20 (END)
```

- Give permissions to Jenkins.

sudo visudo

jenkins ALL=(ALL) NOPASSWD: ALL

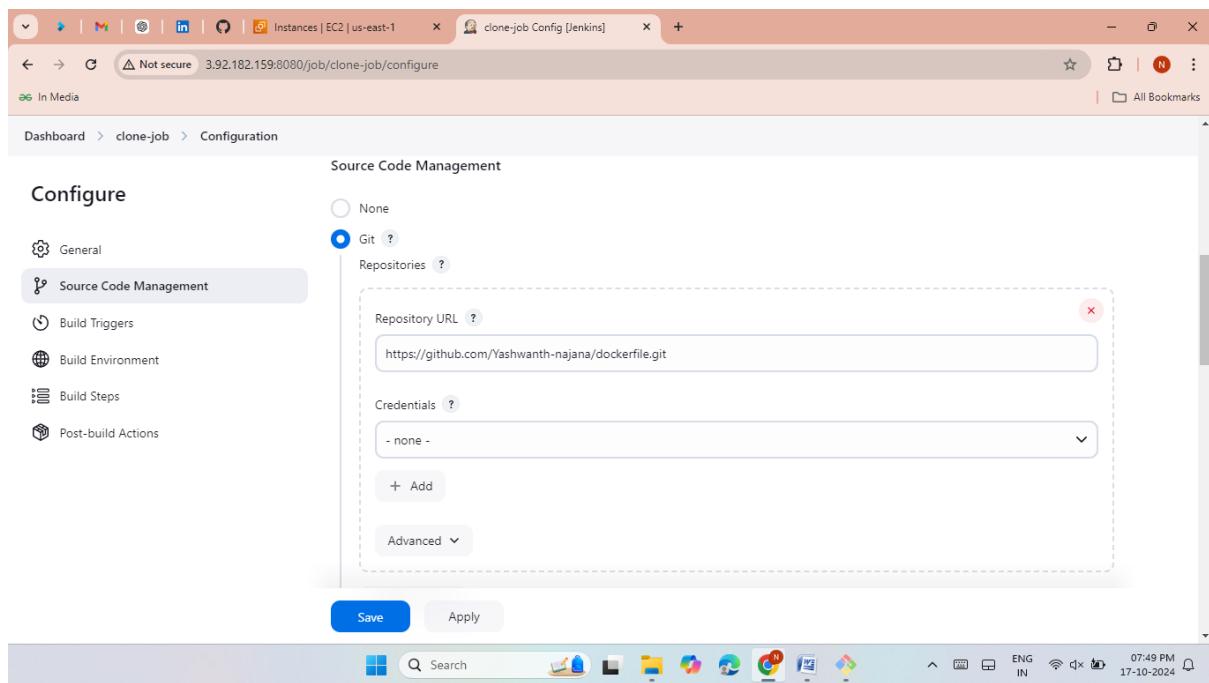
- Here the Jenkins was launched.

The screenshot shows the Jenkins dashboard at <http://3.92.182.159:8080>. The main header says "Dashboard [Jenkins]". Below it, there's a search bar and a "log out" link. On the left, there are links for "New Item", "Build History", "Manage Jenkins", and "My Views". A "Build Queue" section shows "No builds in the queue.". To the right, there's a "Create a job" button and a "Start building your software project" section with links for "Set up a distributed build", "Set up an agent", "Configure a cloud", and "Learn more about distributed builds". The bottom status bar shows system information like battery level, signal strength, and the date/time (07:34 PM 17-10-2024).

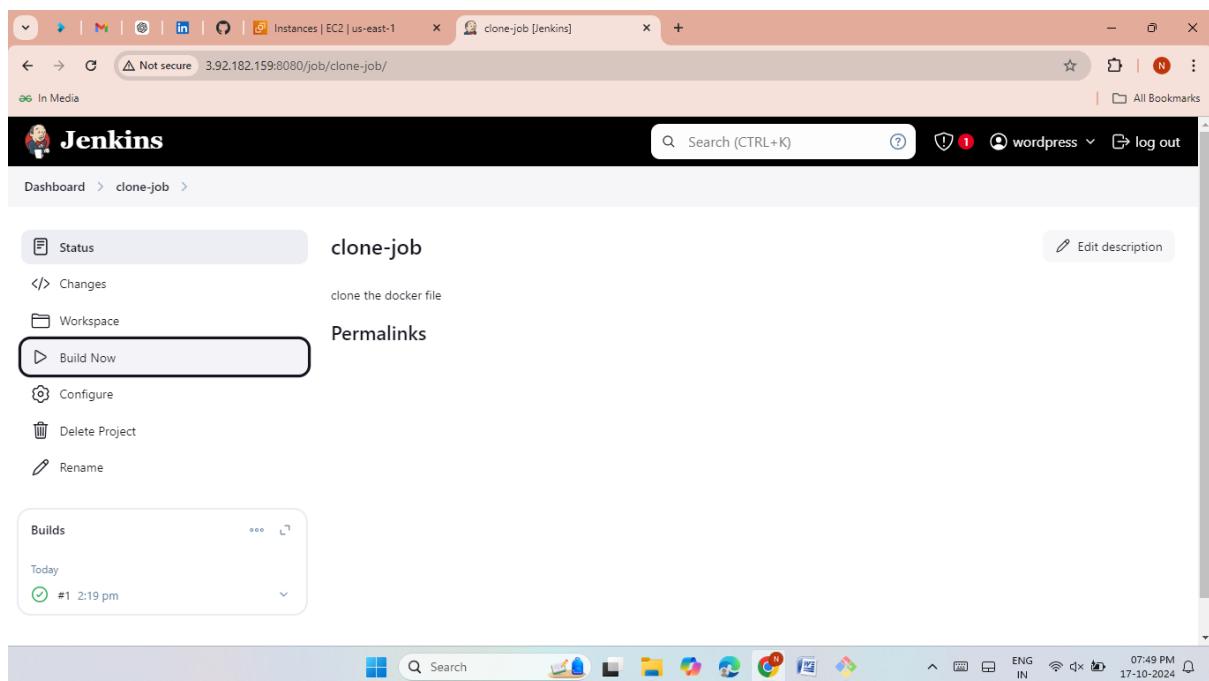
- Create a clone-job for clone the repo from github.

The screenshot shows the "New Item" creation dialog. The title is "New Item". It asks for an item name, which is "clone-job". Below it, there's a "Select an item type" section. The "Freestyle project" option is selected, shown in a highlighted box with a 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." Other options like "Pipeline" and "Multi-configuration project" are also listed. At the bottom is an "OK" button. The bottom status bar shows system information like battery level, signal strength, and the date/time (07:47 PM 17-10-2024).

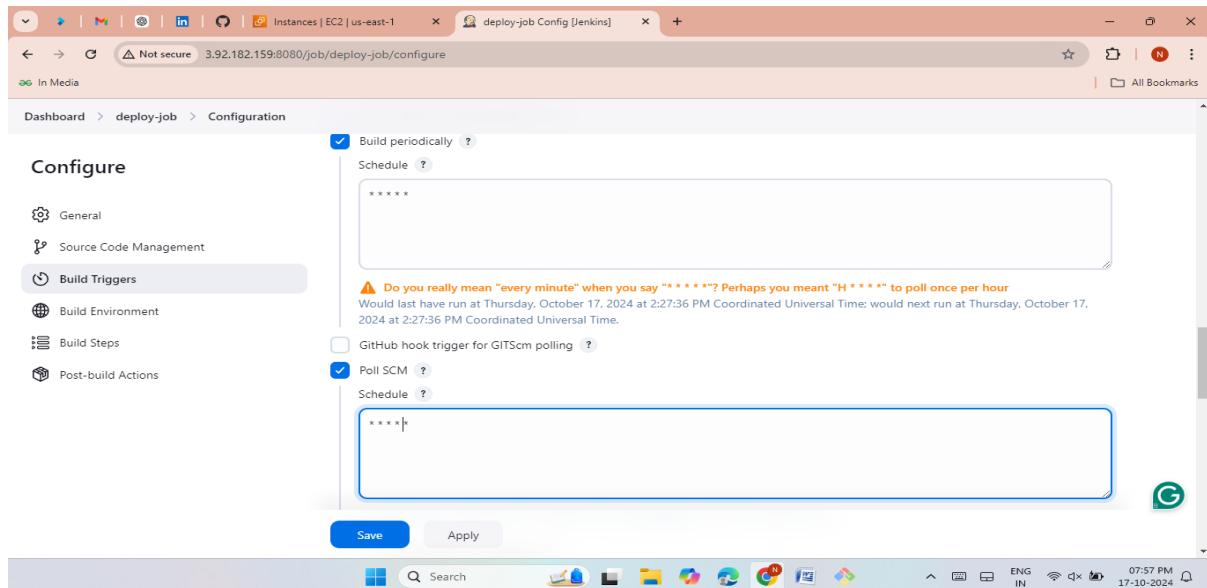
➤ Now past the docker-compose repository.



➤ Here the build was success.



- Now create the deploy job and add the clone-job.
- Now add the build periodically and poll SCM Schedules.



- Now write the script in execute shell.

```
sudo apt update
```

```
sudo apt install -y docker.io
```

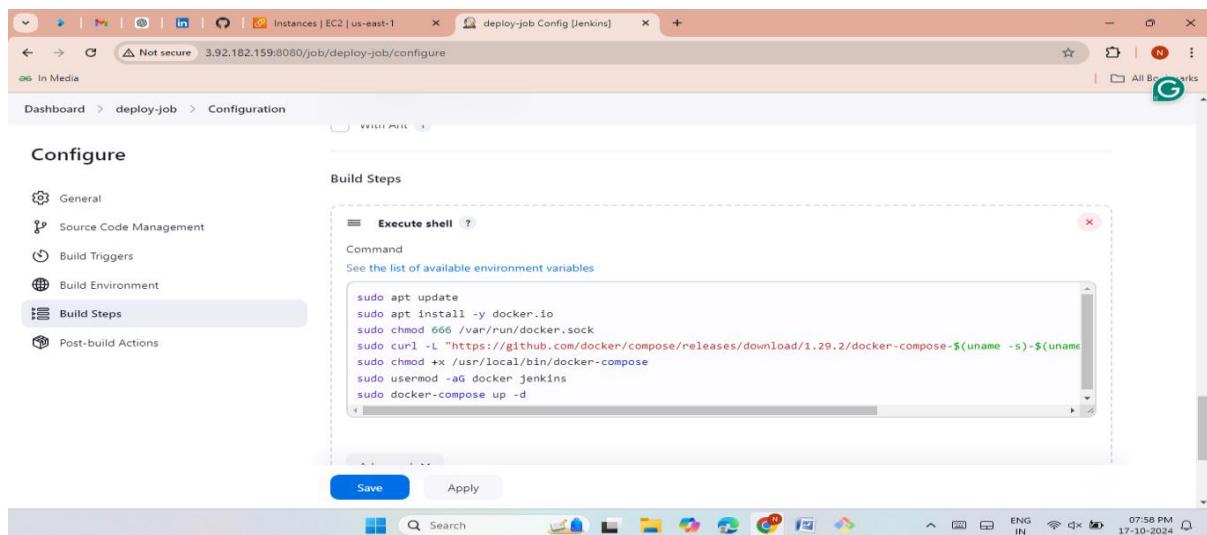
```
sudo chmod 666 /var/run/docker.sock
```

```
sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

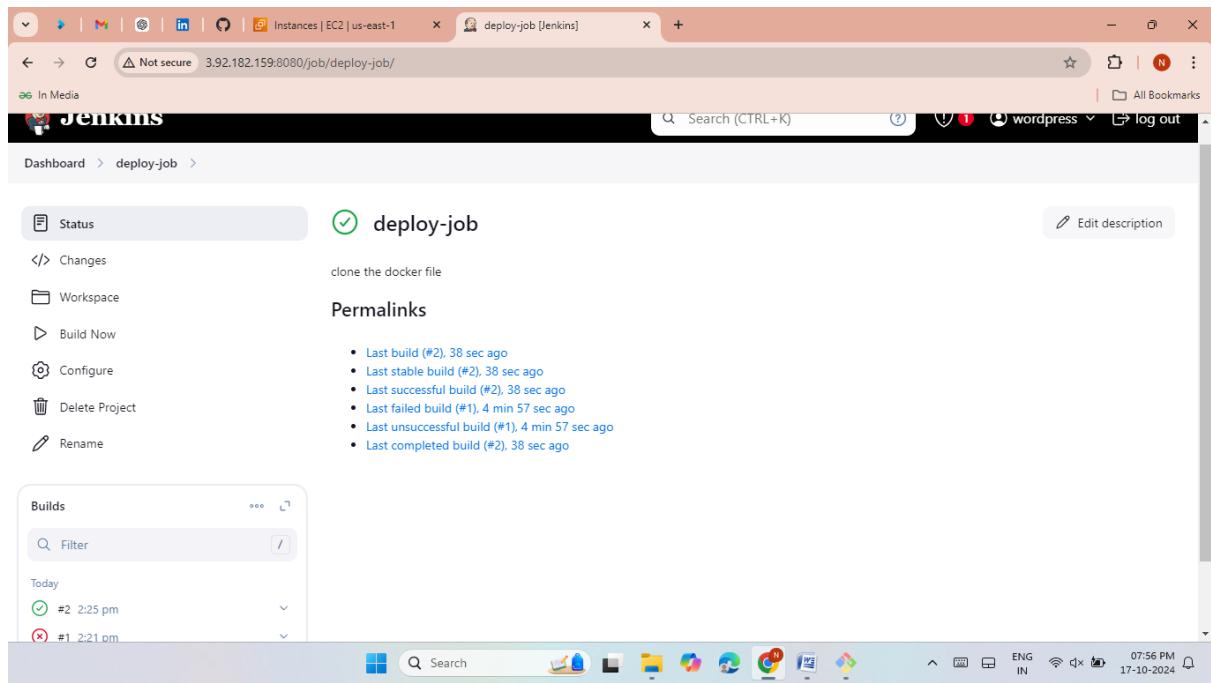
```
sudo chmod +x /usr/local/bin/docker-compose
```

```
sudo usermod -aG docker jenkins
```

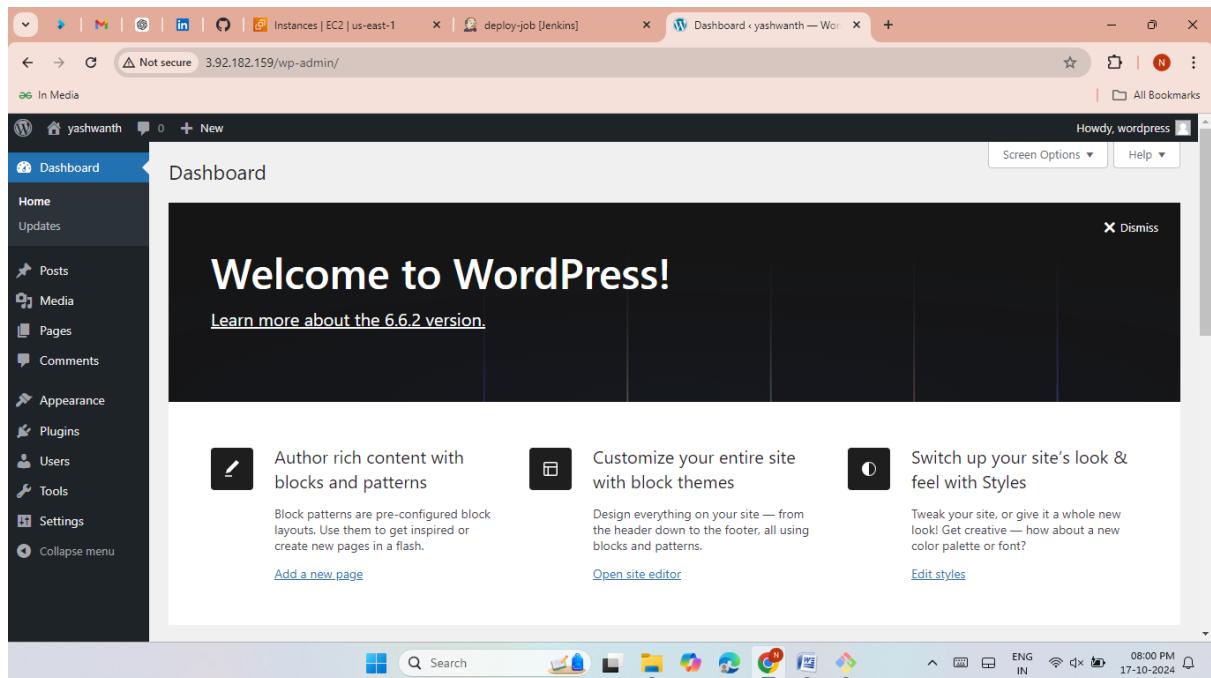
```
sudo docker-compose up -d
```



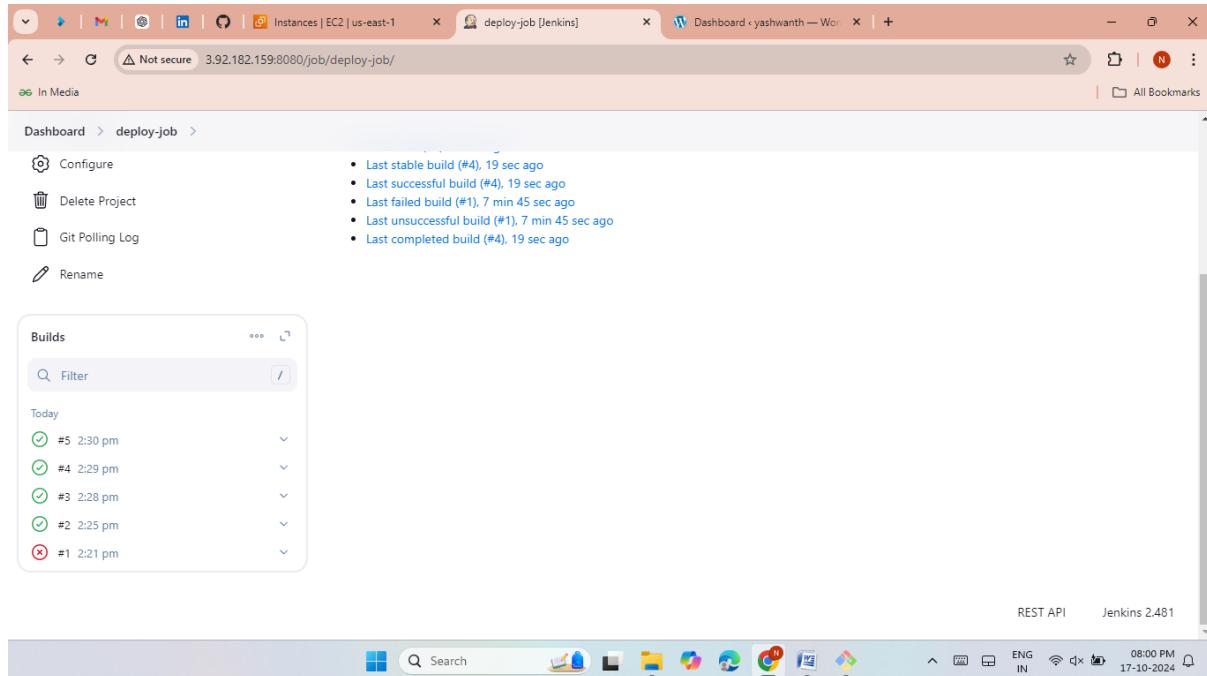
- Here the job was success.



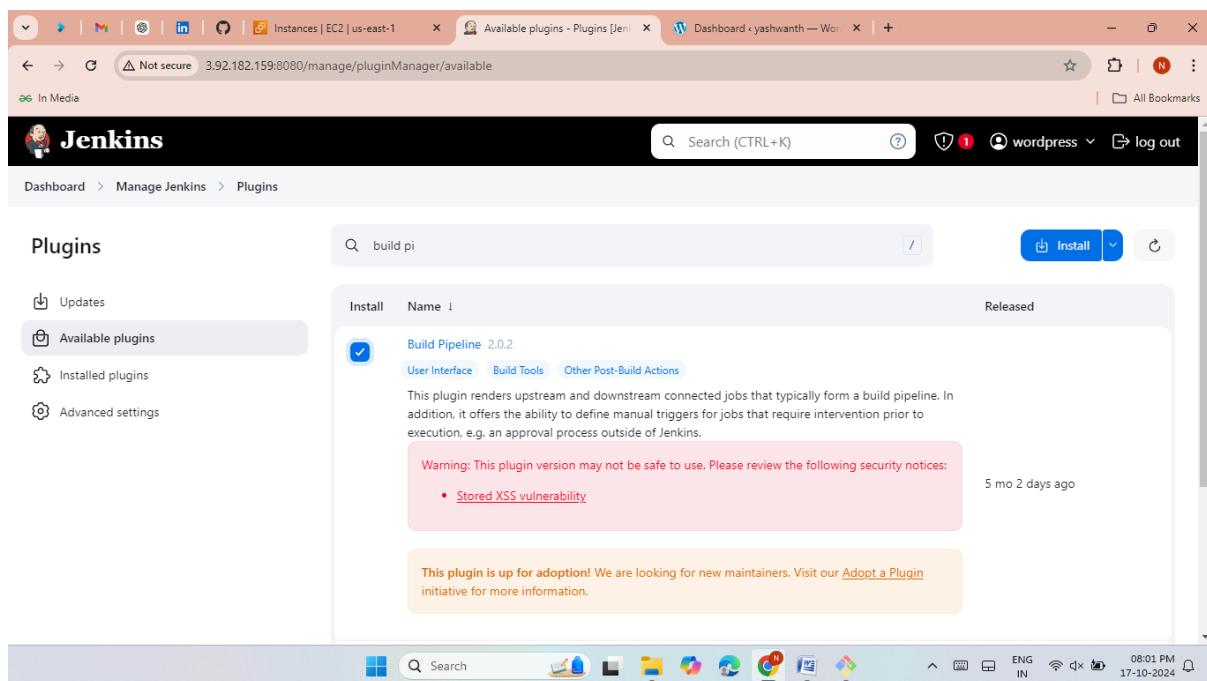
- Now copy that static Public IP and past it in browser.
- Wordpress was launched successfully.



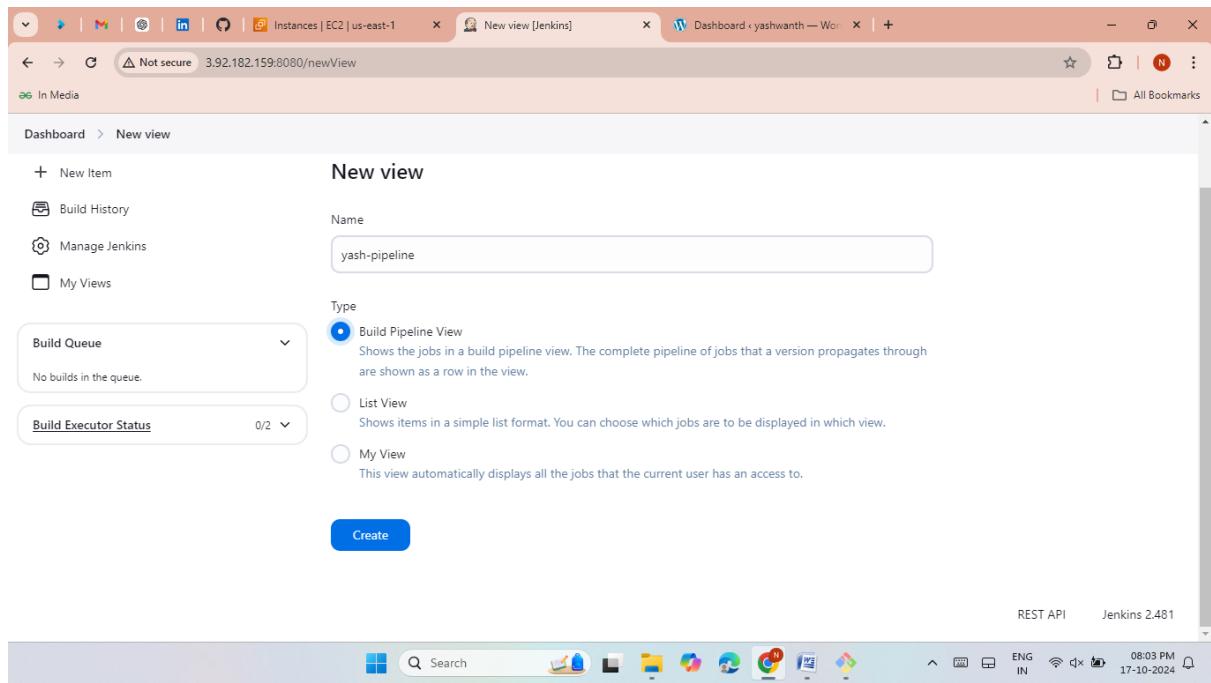
- It runs every MINUTE HOUR DOM(DAY OF MONTH) MONTH DOW(DAY OF WEEK).
- Because I have given (* * * * *).



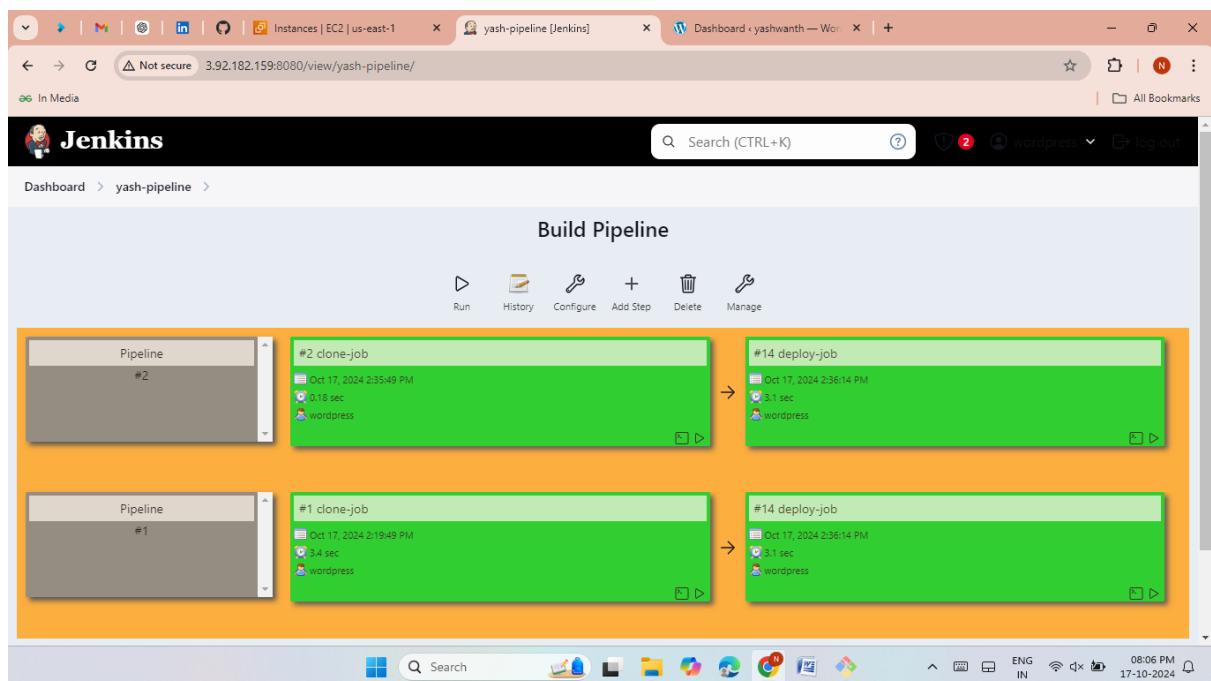
- Now install the build pipeline plugin for create a Jenkins pipeline.



➤ Now create a Jenkins pipeline.



➤ Here the build pipeline is successfully created.



- Now here we can see the Build History.

S	Build	Time Since	Status
1	deploy-job #14	27 sec	stable
2	deploy-job #13	38 sec	stable
3	deploy-job #12	42 sec	stable
4	clone-job #2	52 sec	stable
5	deploy-job #11	1 min 41 sec	stable
6	deploy-job #10	2 min 41 sec	stable
7	deploy-job #9	3 min 37 sec	stable
8	deploy-job #8	3 min 41 sec	stable
9	deploy-job #7	4 min 41 sec	stable

METHOD-7

Deploy WordPress web application by using terraform (create Ec2 instance along with userdata .sh file)

- First Launch an EC2 instance in our AWS account.

➤ Now Connect to locally.

```
ec2-user@ip-172-31-39-122:~  
yaswa@LAPTOP-GM32QJHO MINGW64 ~  
$ cd keys  
  
yaswa@LAPTOP-GM32QJHO MINGW64 ~/keys  
$ ssh -i "yash.pem" ec2-user@ec2-54-91-57-20.compute-1.amazonaws.com  
The authenticity of host 'ec2-54-91-57-20.compute-1.amazonaws.com (54.91.57.20)' can't be established.  
ED25519 key fingerprint is SHA256:Pwshmdjf0d+1cH9KsXdnXftZhi9RL7f/2erz0I#3UA.  
This key is not known by any other names.  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added 'ec2-54-91-57-20.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-172-31-39-122 ~]$
```

- Now install terraform from Google browser.

```
ec2-user@ip-172-31-39-122:~  
warning: /var/cache/yum/x86_64/2/hashicorp/packages/terraform-1.9.8-1.x86_64.rpm: Header V4 RSA/SHA256 Signature, key ID a621e701: NOK  
Public key for terraform-1.9.8-1.x86_64.rpm is not installed  
(7/7): terraform-1.9.8-1.x86_64.rpm | 27 MB 00:00:00  
Total 59 MB/s | 40 MB 00:00:00  
  
Retrieving key from https://rpm.releases.hashicorp.com/gpg  
Importing GPG key 0xA621e701:  
Userid : "Hashicorp Security (Hashicorp Package Signing) <security+packaging@hashicorp.com>"  
Fingerprint: 798a ec65 4e5c 1542 8c8e 42ee aaf6 fcfc a621 e701  
From https://rpm.releases.hashicorp.com/gpg  
Running transaction check  
Running transaction test  
Transaction test succeeded  
Running transaction  
  Installing : git-core-2.40.1-1.amzn2.0.3.x86_64 1/7  
  Installing : git-core-doc-2.40.1-1.amzn2.0.3.noarch 2/7  
  Installing : l:perl-Error-0.17020-2.amzn2.noarch 3/7  
  Installing : perl-TermReadkey-2.30-20.amzn2.0.2.x86_64 4/7  
  Installing : perl-Git-2.40.1-1.amzn2.0.3.noarch 5/7  
  Installing : git-2.40.1-1.amzn2.0.3.x86_64 6/7  
  Verifying : perl-TermReadkey-2.30-20.amzn2.0.2.x86_64 1/7  
  Verifying : terraform-1.9.8-1.x86_64 2/7  
  Verifying : git-2.40.1-1.amzn2.0.3.x86_64 3/7  
  Verifying : l:perl-Error-0.17020-2.amzn2.noarch 4/7  
  Verifying : git-core-2.40.1-1.amzn2.0.3.x86_64 5/7  
  Verifying : git-core-doc-2.40.1-1.amzn2.0.3.noarch 6/7  
  Verifying : perl-Git-2.40.1-1.amzn2.0.3.noarch 7/7  
  
Installed:  
  terraform.x86_64 0:1.9.8-1  
  
Dependency Installed:  
  git.x86_64 0:2.40.1-1.amzn2.0.3      git-core.x86_64 0:2.40.1-1.amzn2.0.3      git-core-doc.noarch 0:2.40.1-1.amzn2.0.3  
  perl-Error.noarch 1:0.17020-2.amzn2    perl-Git.noarch 0:2.40.1-1.amzn2.0.3      perl-TermReadKey.x86_64 0:2.30-20.amzn2.0.2  
  
Complete!  
[ec2-user@ip-172-31-39-122 ~]$
```

- Write a terraform script for launch an EC2 instance.
- Provide Access key and Secret key.
- Write script for EC2 and Security Groups, VPC, Subnets, Internet Gateway, Route Table.

```
ec2-user@ip-172-31-39-122:~$ provider "aws" {
  region      = "us-east-1"
  access_key  = ""
  secret_key  =
}

#creating VPC
resource "aws_vpc" "yashvpc" {
  cidr_block     = "10.0.0.0/16"
  instance_tenancy = "default"
  tags = {
    Name = "yashvpc"
  }
}

#subnet_creation
resource "aws_subnet" "web-subnet1" {
  vpc_id        = aws_vpc.yashvpc.id
  cidr_block   = "10.0.1.0/24"
  map_public_ip_on_launch = true
  availability_zone = "us-east-1a"
  tags = {
    name = "web-subnet1"
  }
}

resource "aws_subnet" "web-subnet2" {
  vpc_id        = aws_vpc.yashvpc.id
  cidr_block   = "10.0.2.0/24"
  map_public_ip_on_launch = true
  availability_zone = "us-east-1b"
  tags = {
    name = "web-subnet2"
  }
}

resource "aws_subnet" "application-subnet1" {
  vpc_id        = aws_vpc.yashvpc.id
  cidr_block   = "10.0.3.0/24"
  map_public_ip_on_launch = true
  availability_zone = "us-east-1a"
  tags = {
    name = "application-subnet1"
  }
}

resource "aws_subnet" "application-subnet2" {
  vpc_id        = aws_vpc.yashvpc.id
  -- INSERT --
  3,18          Top
}

ec2-user@ip-172-31-39-122:~$
```

```
ec2-user@ip-172-31-39-122:~$ provider "aws" {
  region      = "us-east-1"
  access_key  = ""
  secret_key  =
}

resource "aws_subnet" "application-subnet1" {
  vpc_id        = aws_vpc.yashvpc.id
  cidr_block   = "10.0.3.0/24"
  map_public_ip_on_launch = true
  availability_zone = "us-east-1a"
  tags = {
    name = "application-subnet1"
  }
}

resource "aws_subnet" "application-subnet2" {
  vpc_id        = aws_vpc.yashvpc.id
  cidr_block   = "10.0.4.0/24"
  map_public_ip_on_launch = true
  availability_zone = "us-east-1b"
  tags = {
    name = "application-subnet2"
  }
}

resource "aws_subnet" "database-subnet1" {
  vpc_id        = aws_vpc.yashvpc.id
  cidr_block   = "10.0.5.0/24"
  map_public_ip_on_launch = true
  availability_zone = "us-east-1a"
  tags = {
    name = "database-subnet1"
  }
}

resource "aws_subnet" "database-subnet2" {
  vpc_id        = aws_vpc.yashvpc.id
  cidr_block   = "10.0.6.0/24"
  map_public_ip_on_launch = true
  availability_zone = "us-east-1b"
  tags = {
    name = "database-subnet2"
  }
}

#creating internet gateway
resource "aws_internet_gateway" "yashigw" {
  vpc_id = aws_vpc.yashvpc.id
}

# Creating route table
resource "aws_route_table" "public-route-table" {
  vpc_id = aws_vpc.yashvpc.id
  -- INSERT --
  43,8          33%
}

ec2-user@ip-172-31-39-122:~$
```

```
ec2-user@ip-172-31-39-122:~
```

```
# Creating route table
resource "aws_route_table" "public-route-table" {
  vpc_id = aws_vpc.yashvpc.id
  route_table_name = "public"
  cidr_block = "0.0.0.0/0"
  gateway_id = aws_internet_gateway.yashigw.id
}
tags = [
  { Name = "public-route-table" }
]

# Associating Route table
resource "aws_route_table_association" "rt1" {
  subnet_id = aws_subnet.web-subnet1.id
  route_table_id = aws_route_table.public-route-table.id
}

# Associating Route table
resource "aws_route_table_association" "rt2" {
  subnet_id = aws_subnet.web-subnet2.id
  route_table_id = aws_route_table.public-route-table.id
}

#creating security group
resource "aws_security_group" "securitygroup" {
  vpc_id = aws_vpc.yashvpc.id
  #Inbound Rules
  ingress {
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  ingress {
    from_port   = 443
    to_port     = 443
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
  tags = [
    { Name = "securitygroup" }
  ]
}

#creating 1st EC2 instance
resource "aws_instance" "wordpress" {
  ami           = "ami-0ddc798b3f1a5117e"
  instance_type = "t2.micro"
  key_name      = "yash"
  subnet_id     = aws_subnet.web-subnet1.id
  vpc_security_group_ids = [aws_security_group.securitygroup.id]
  associate_public_ip_address = true
  user_data     = file("data.sh")
  tags = [
    { Name = "wordpress" }
  ]
}

-- INSERT --
```

```
ec2-user@ip-172-31-39-122:~
```

```
}

ingress {
  from_port   = 443
  to_port     = 443
  protocol    = "tcp"
  cidr_blocks = ["0.0.0.0/0"]
}
ingress {
  from_port   = 22
  to_port     = 22
  protocol    = "tcp"
  cidr_blocks = ["0.0.0.0/0"]
}
egress {
  from_port   = 0
  to_port     = 0
  protocol    = "-1"
  cidr_blocks = ["0.0.0.0/0"]
}
tags = [
  { Name = "securitygroup" }
]

#creating 1st EC2 instance
resource "aws_instance" "wordpress" {
  ami           = "ami-0ddc798b3f1a5117e"
  instance_type = "t2.micro"
  key_name      = "yash"
  subnet_id     = aws_subnet.web-subnet1.id
  vpc_security_group_ids = [aws_security_group.securitygroup.id]
  associate_public_ip_address = true
  user_data     = file("data.sh")
  tags = [
    { Name = "wordpress" }
  ]
}

-- INSERT --
```

- Now write the script for data file.
- Clone the docker-compose file from github.

```

ec2-user@ip-172-31-39-122:~#
#!/bin/bash
sudo yum install -y git
sudo yum install -y docker
sudo systemctl start docker
sudo systemctl enable docker
sudo chmod 666 /var/run/docker.sock
sudo usermod -aG docker ec2-user
sudo systemctl restart docker
DOCKER_COMPOSE_VERSION="v2.20.0"
curl -L "https://github.com/docker/compose/releases/download/${DOCKER_COMPOSE_VERSION}/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
sudo git clone https://github.com/Yashwanth-najana/dockerfile.git
cd dockerfile/
docker-compose up -d
docker ps
~
```

15,9 All 03:45 PM 19-10-2024 ENG IN

- Now enter the following commands

terraform init

terraform fmt

terraform validate

terraform plan

terraform apply

```

Verifying : perl-Git-2.40.1-1.amzn2.0.3.noarch
Installed:
  terraform.x86_64 0:1.9.8-1

Dependency Installed:
  git.x86_64 0:2.40.1-1.amzn2.0.3      git-core.x86_64 0:2.40.1-1.amzn2.0.3      git-core-doc.noarch 0:2.40.1-1.amzn2.0.3
  perl=Error.noarch 1:0.17020-2.amzn2.0.2  perl-Git.noarch 0:2.40.1-1.amzn2.0.3      perl-TermReadKey.x86_64 0:2.30-20.amzn2.0.2

Complete!
[ec2-user@ip-172-31-39-122 ~]$ sudo vi wordpress.tf
[ec2-user@ip-172-31-39-122 ~]$ sudo vi data.sh
[ec2-user@ip-172-31-39-122 ~]$ sudo 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-172-31-39-122 ~]$ sudo terraform fmt
[ec2-user@ip-172-31-39-122 ~]$ sudo terraform validate
Success! The configuration is valid.

[ec2-user@ip-172-31-39-122 ~]$ |
```

7/7 03:36 PM 19-10-2024 ENG IN

```
ec2-user@ip-172-31-39-122:~  
    + "name" = "web-subnet2"  
  }  
+ vpc_id  
}  
  
# aws_vpc.yashvpc will be created  
resource "aws_vpc" "yashvpc" {  
  + arn  
  + cidr_block  
  + default_network_acl_id  
  + default_route_table_id  
  + default_security_group_id  
  + dhcp_options_id  
  + enable_dns_hostnames  
  + enable_dns_support  
  + enable_network_address_usage_metrics  
  + id  
  + instance_tenancy  
  + ipv6_association_id  
  + ipv6_cidr_block  
  + ipv6_cidr_block_network_border_group  
  + main_route_table_id  
  + owner_id  
  + tags  
    + "Name" = "yashvpc"  
  }  
+ tags_all  
  + "Name" = "yashvpc"  
}  
  
Plan: 13 to add, 0 to change, 0 to destroy.  
  
Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run  
"terraform apply" now.  
[ec2-user@ip-172-31-39-122 ~]$
```

```
ec2-user@ip-172-31-39-122:~  
Enter a value: yes  
  
aws_vpc.yashvpc: Creating...  
aws_vpc.yashvpc: Creation complete after 1s [id=vpc-092e450f6b11d785f]  
aws_subnet.web-subnet2: Creating...  
aws_subnet.database-subnet2: Creating...  
aws_subnet.application-subnet1: Creating...  
aws_subnet.database-subnet1: Creating...  
aws_security_group.securitygroup: Creating...  
aws_subnet.web-subnet1: Creating...  
aws_internet_gateway.yashigw: Creating...  
aws_subnet.application-subnet2: Creating...  
aws_internet_gateway.yashigw: Creation complete after 0s [id=igw-00d43aef1c443a3f]  
aws_route_table.public-route-table: Creating...  
aws_route_table.public-route-table: Creation complete after 1s [id=rtb-0b1aa20ec0ba5b2ac]  
aws_security_group.securitygroup: Creation complete after 3s [id=sg-0e78560423b1d9b32]  
aws_subnet.web-subnet2: still creating... [10s elapsed]  
aws_subnet.database-subnet2: still creating... [10s elapsed]  
aws_subnet.application-subnet1: still creating... [10s elapsed]  
aws_subnet.web-subnet1: still creating... [10s elapsed]  
aws_subnet.application-subnet2: still creating... [10s elapsed]  
aws_subnet.database-subnet1: still creating... [11s elapsed]  
aws_subnet.application-subnet1: Creation complete after 11s [id=subnet-0a2422b9acd8fc44e]  
aws_subnet.database-subnet2: Creation complete after 11s [id=subnet-083b67742b59f6bab]  
aws_subnet.web-subnet2: Creation complete after 11s [id=subnet-0ecf3922f61ec1966]  
aws_route_table_association.rt2: Creating...  
aws_subnet.web-subnet1: Creation complete after 10s [id=subnet-04e9d0d243cb6ff52]  
aws_instance.wordpress: Creating...  
aws_route_table_association.rt1: Creating...  
aws_subnet.application-subnet2: Creation complete after 11s [id=subnet-07ff5e0c44c45e56e]  
aws_subnet.database-subnet1: Creation complete after 12s [id=subnet-0afb98ba5733a4af1]  
aws_route_table_association.rt1: Creation complete after 0s [id=rtbassoc-05977ff05c781c5c4]  
aws_route_table_association.rt2: Creation complete after 1s [id=rtbassoc-0d5bd5f38a56eff53]  
aws_instance.wordpress: Still creating... [10s elapsed]  
aws_instance.wordpress: Creation complete after 13s [id=i-0cc60f3e240b7cc62]  
  
Apply complete! Resources: 13 added, 0 changed, 0 destroyed.  
[ec2-user@ip-172-31-39-122 ~]$
```

- Here the EC2 instance was launched.
- Past the instance static Public IP in browser.

The screenshot shows the AWS EC2 Instances console. The left sidebar navigation includes EC2 Dashboard, EC2 Global View, Events, Instances (selected), Instances Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations (New), Images (AMIs, AMI Catalog), and CloudShell/Feedback. The main content area displays 'Instances (1/2) Info' with two entries:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability
wordpress	i-0cc60f3e240b7cc62	Running	t2.micro	Initializing	View alarms +	us-east-1a
yash-method-7	i-0fb781d50a49deaf6	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b

Below the table, the details for the selected instance 'i-0cc60f3e240b7cc62 (wordpress)' are shown. The 'Details' tab is selected, displaying the following information:

- Instance ID: i-0cc60f3e240b7cc62 (wordpress)
- Public IPv4 address: 54.242.189.51 | open address ↗
- Private IPv4 addresses: 10.0.1.234
- IPv6 address: -
- Instance state: Running
- Public IPv4 DNS: -

- Here the VPC, Subnet, Internet Gateway, Route Table, Security Groups also created.

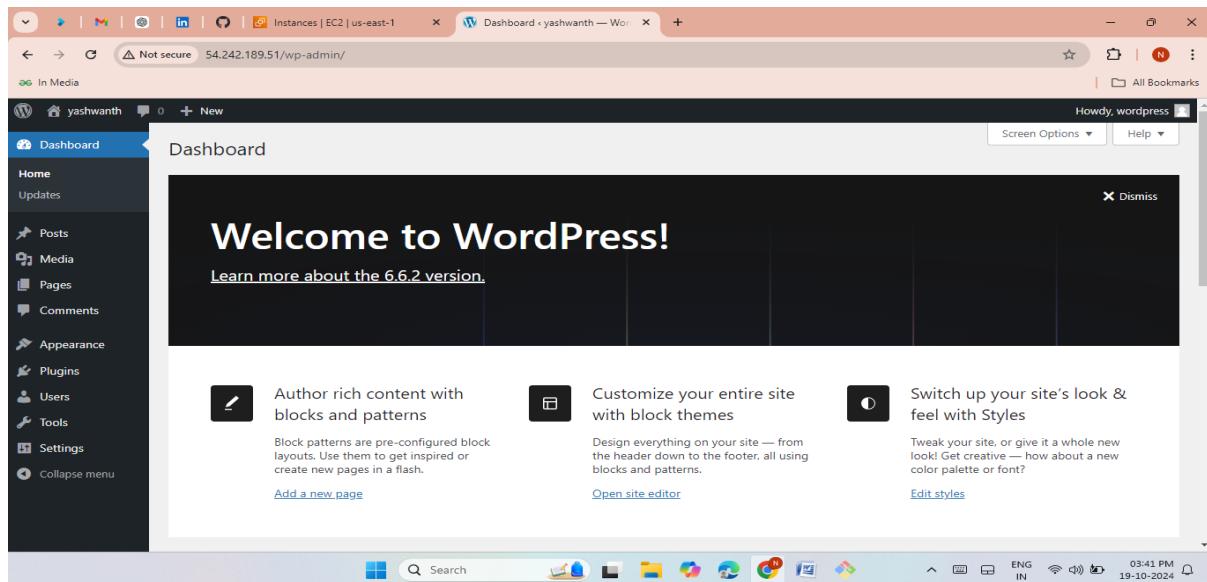
The screenshot shows the AWS VPC Console. The left sidebar navigation includes EC2 Global View (Filter by VPC), Virtual private cloud (Your VPCs selected), Subnets, Route tables, Internet gateways, Egress-only internet gateways, Carrier gateways, DHCP option sets, Elastic IPs, Managed prefix lists, and Endpoints. The main content area displays 'Your VPCs (1/2) Info' with one entry:

Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR
-	vpc-05f2f9c35f844e371	Available	172.31.0.0/16	-
yashvpc	vpc-092e450f6b11d785f	Available	10.0.0.0/16	-

Below the table, the details for the selected VPC 'vpc-092e450f6b11d785f / yashvpc' are shown. The 'Details' tab is selected, displaying the following information:

- VPC ID: vpc-092e450f6b11d785f
- State: Available
- DNS hostnames: Disabled
- DNS resolution: Enabled

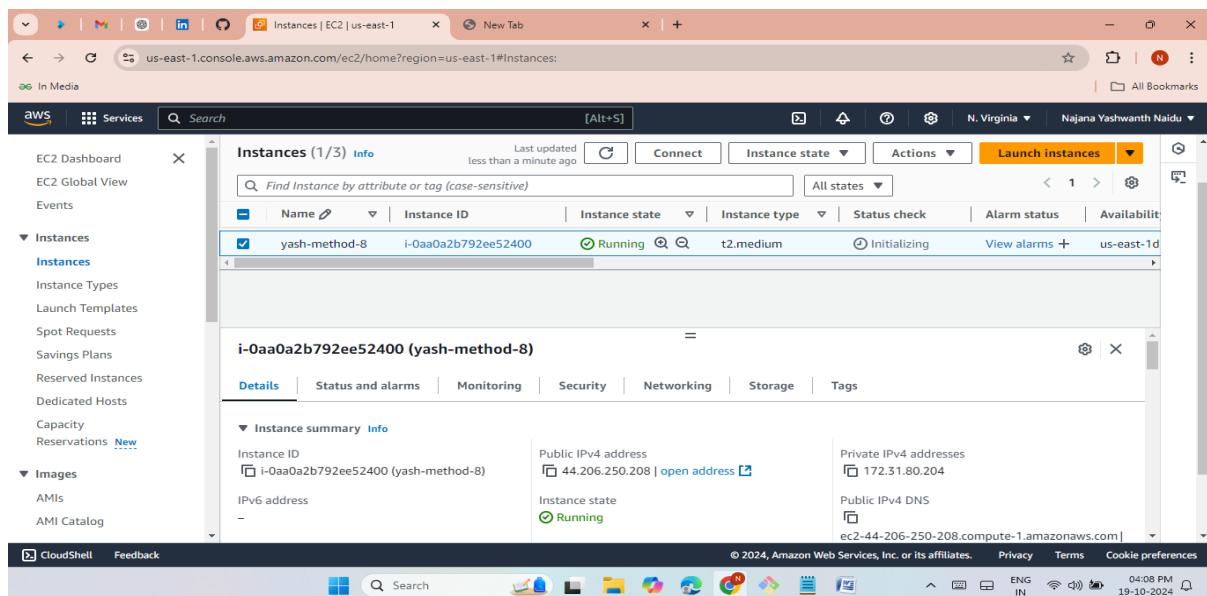
- Here the wordpress was launched.



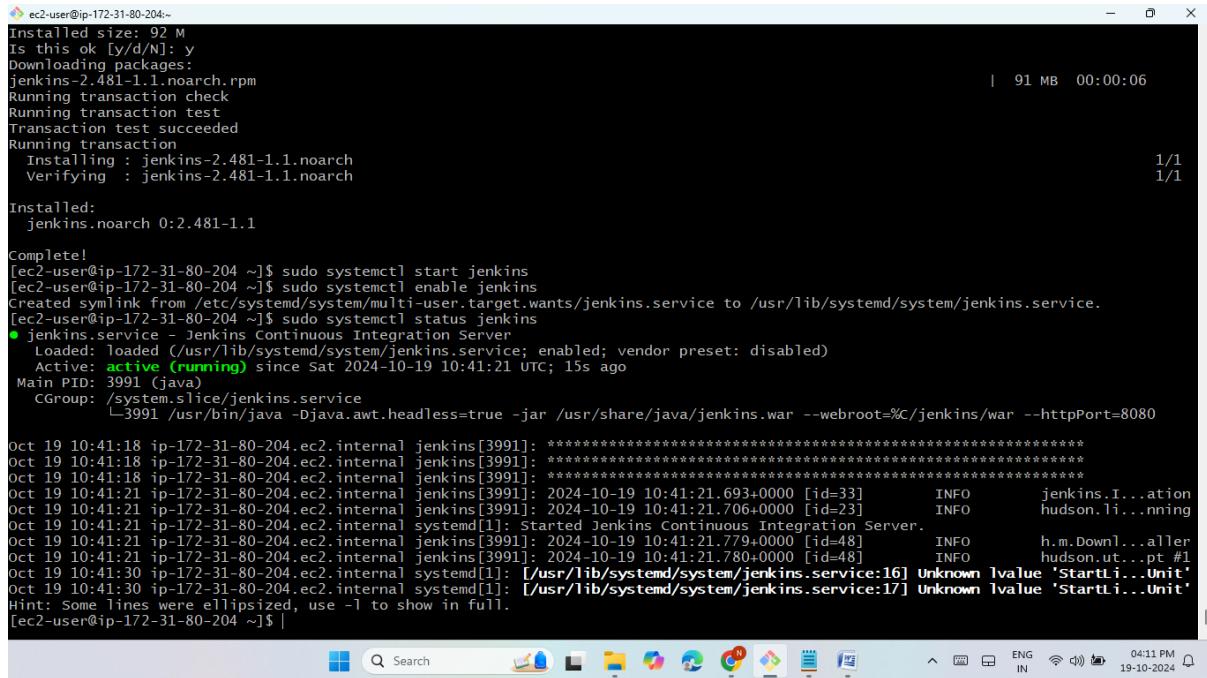
METHOD-8

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

- First launch the EC2 instance with the Jenkins port number(8080).



- Now Install git and Jenkins.
- Now start and enable the Jenkins.



```

ec2-user@ip-172-31-80-204:~$ sudo yum install jenkins
Installed size: 92 M
Is this ok [y/d/N]: y
Downloading packages:
jenkins-2.481-1.1.noarch.rpm
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : jenkins-2.481-1.1.noarch 1/1
  Verifying  : jenkins-2.481-1.1.noarch 1/1

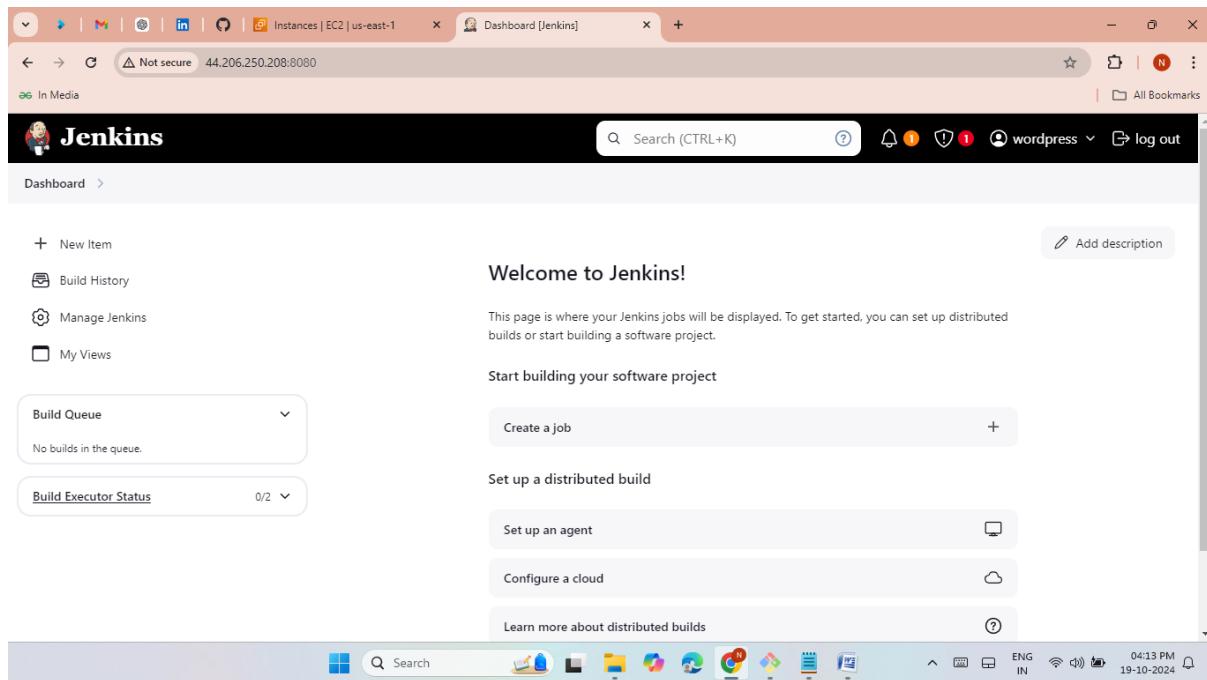
Installed:
  jenkins.noarch 0:2.481-1.1

Complete!
[ec2-user@ip-172-31-80-204 ~]$ sudo systemctl start jenkins
[ec2-user@ip-172-31-80-204 ~]$ 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-172-31-80-204 ~]$ 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 Sat 2024-10-19 10:41:21 UTC; 15s ago
       PID: 3991 (java)
      CGroup: /system.slice/jenkins.service
              └─3991 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=%C/jenkins/war --httpPort=8080

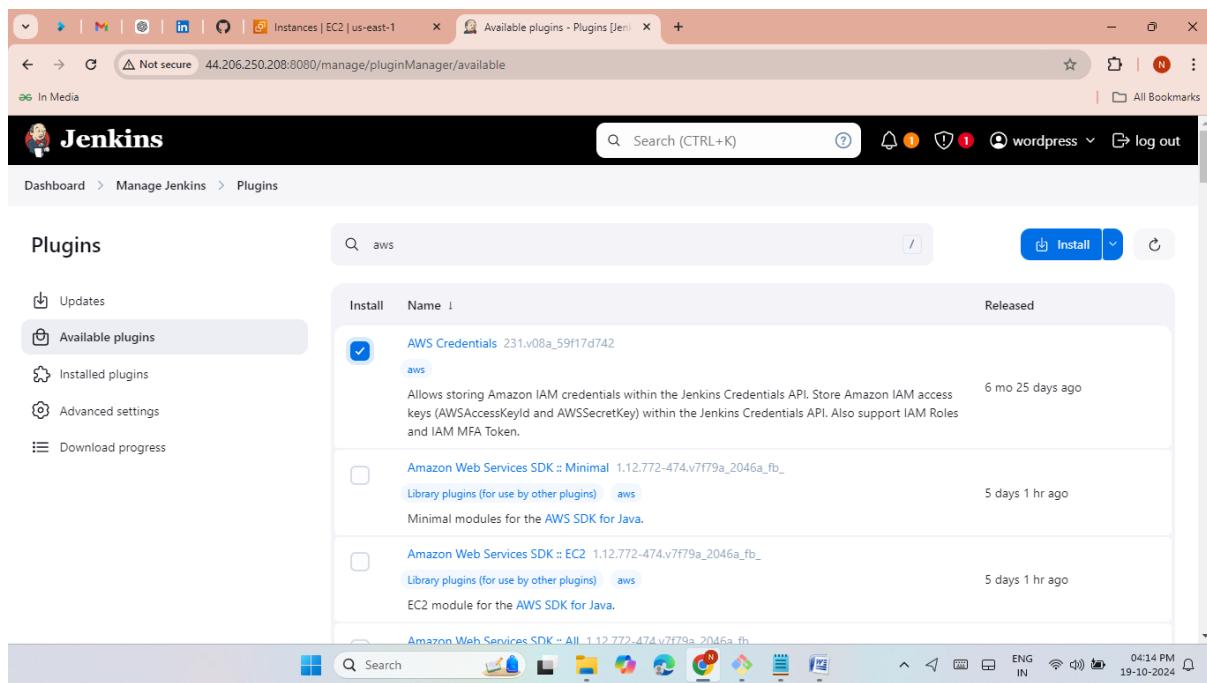
Oct 19 10:41:18 ip-172-31-80-204.ec2.internal jenkins[3991]: ****
Oct 19 10:41:18 ip-172-31-80-204.ec2.internal jenkins[3991]: ****
Oct 19 10:41:18 ip-172-31-80-204.ec2.internal jenkins[3991]: ****
Oct 19 10:41:21 ip-172-31-80-204.ec2.internal jenkins[3991]: 2024-10-19 10:41:21.693+0000 [id=33] INFO jenkins.I...
Oct 19 10:41:21 ip-172-31-80-204.ec2.internal jenkins[3991]: 2024-10-19 10:41:21.706+0000 [id=23] INFO hudson.li...
Oct 19 10:41:21 ip-172-31-80-204.ec2.internal systemd[1]: Started Jenkins Continuous Integration Server.
Oct 19 10:41:21 ip-172-31-80-204.ec2.internal jenkins[3991]: 2024-10-19 10:41:21.779+0000 [id=48] INFO h.m.Down...aller
Oct 19 10:41:21 ip-172-31-80-204.ec2.internal jenkins[3991]: 2024-10-19 10:41:21.780+0000 [id=48] INFO hudson.ut...pt #1
Oct 19 10:41:30 ip-172-31-80-204.ec2.internal systemd[1]: [/usr/lib/systemd/system/jenkins.service:16] unknown lvalue 'StartLi...Unit'
Oct 19 10:41:30 ip-172-31-80-204.ec2.internal systemd[1]: [/usr/lib/systemd/system/jenkins.service:17] unknown lvalue 'StartLi...Unit'
Hint: Some lines were ellipsized, use -l to show in full.
[ec2-user@ip-172-31-80-204 ~]$ |

```

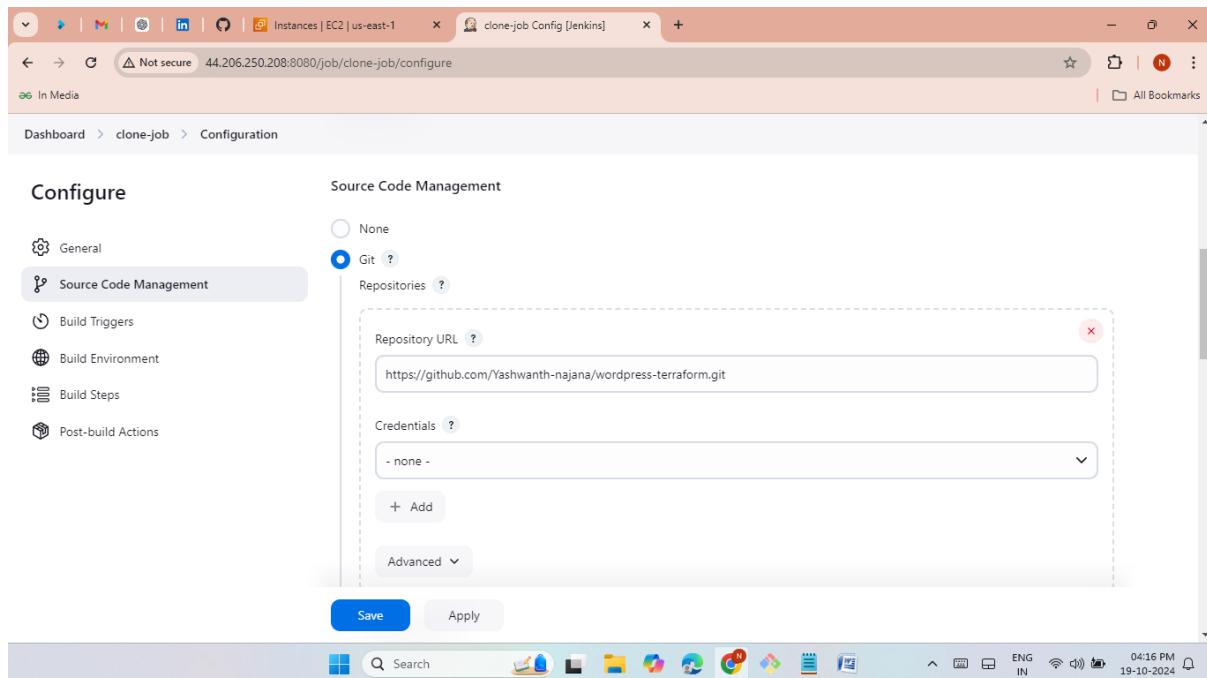
- Here the Jenkins was launched successfully.



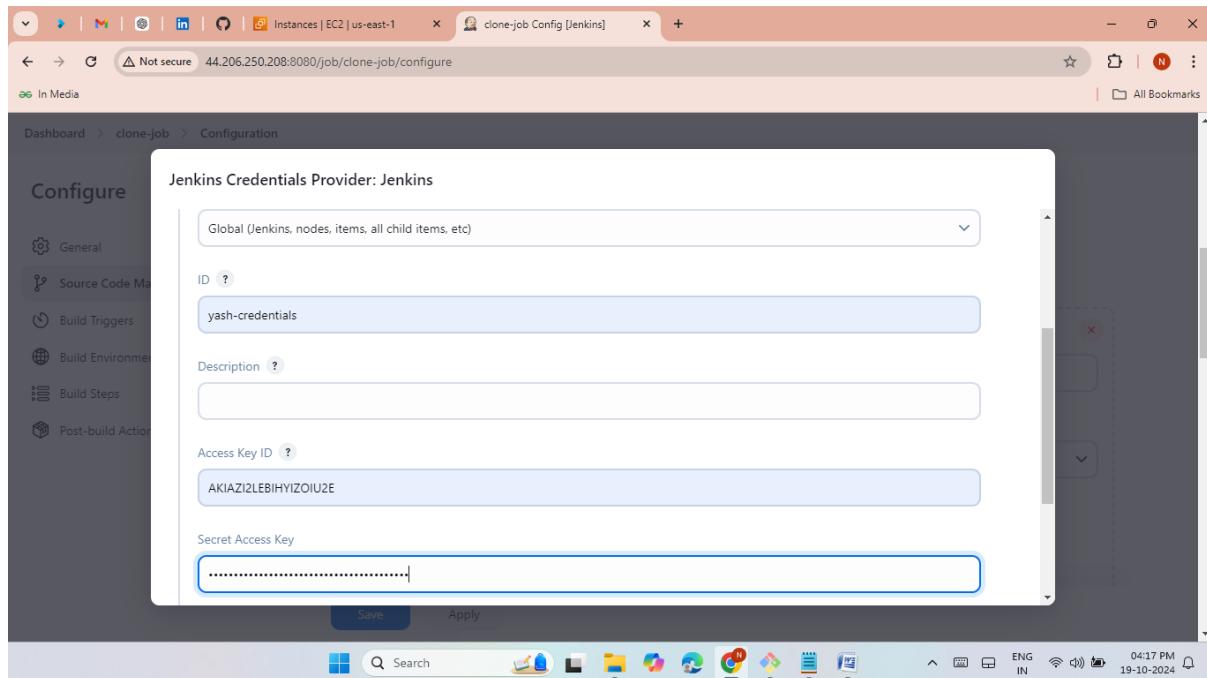
➤ Now install the AWS credentials plugin.



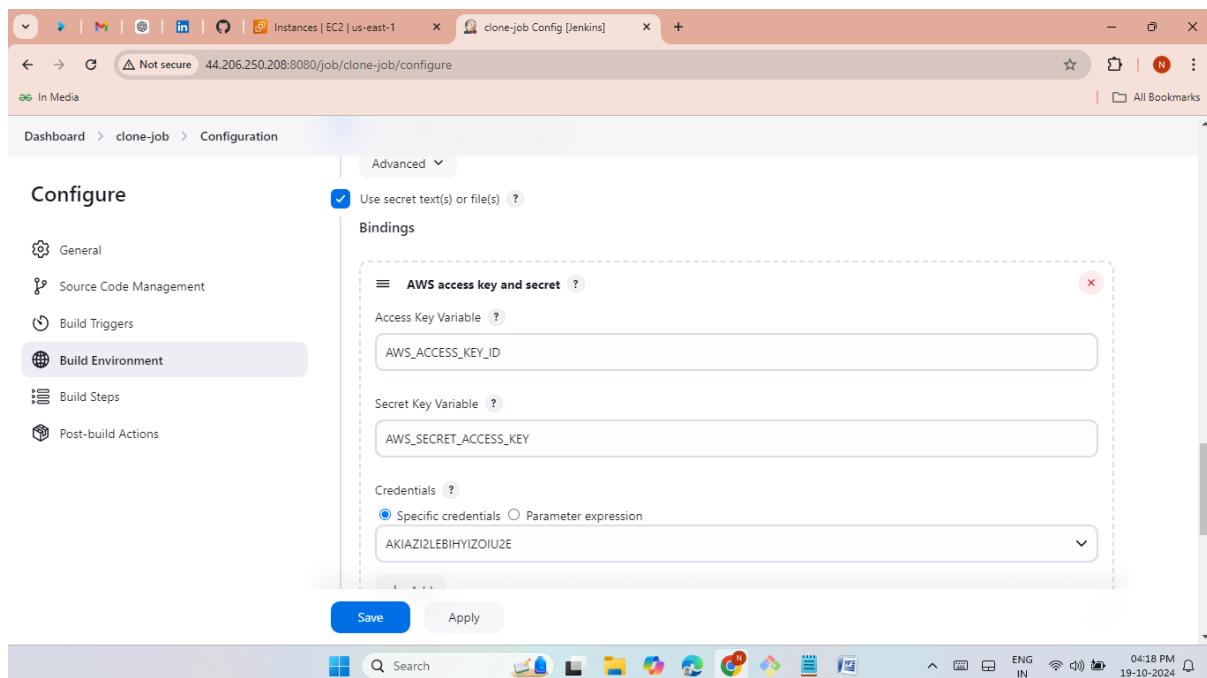
➤ Now create a clone-job and clone the terraform script form github.



- Then create the credentials.
- Add AWS credentials(access key and secret key).



- Now add the AWS access key and secret key.
- Now apply the changes.



➤ Here the clone-job is success.

The screenshot shows a web browser window with the address bar displaying 'Not secure 44.206.250.208:8080/job/clone-job/'. The page title is 'clone-job [Jenkins]'. The main content area shows a Jenkins job named 'clone-job' with a green checkmark icon. Below the title, it says 'clone the terraform script for launch wordpress'. A 'Permalinks' section lists several recent builds. On the left, there's a sidebar with links like 'Status', 'Changes', 'Workspace', 'Build Now', 'Configure', 'Delete Project', and 'Rename'. At the bottom, there's a 'Builds' section with a table showing one build entry: '#1 10:48 am'. The system tray at the bottom right shows 'ENG IN' and the date '19-10-2024'.

➤ This is the terraform script.

The screenshot shows a web browser window with the address bar displaying 'github.com/Yashwanth-najana/wordpress-terraform/blob/main/wordpress.tf'. The page title is 'wordpress-terraform / wordpress.tf []'. The main content area shows a GitHub code editor for a file named 'wordpress.tf'. The code is as follows:

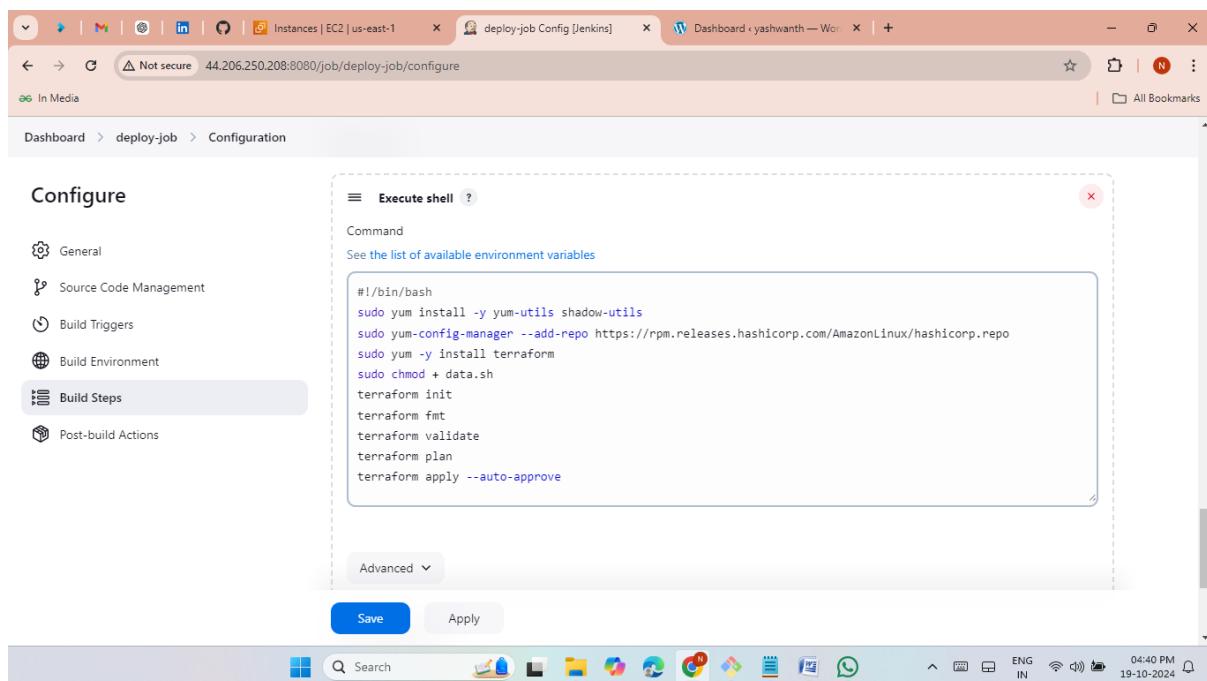
```
provider "aws" {
  region    = "us-east-1"
}

#creating VPC
resource "aws_vpc" "yashvpc" {
  cidr_block      = "10.0.0.0/16"
  instance_tenancy = "default"
  tags = {
    Name = "yashvpc"
  }
}
#subnet_creation
resource "aws_subnet" "web-subnet1" {
  vpc_id          = aws_vpc.yashvpc.id
  cidr_block      = "10.0.1.0/24"
  map_public_ip_on_launch = true
  availability_zone = "us-east-1a"
  tags = {
    name = "web-subnet1"
  }
}
```

The left sidebar shows other files in the repository: 'README.md', 'data.sh', and 'wordpress.tf'. The system tray at the bottom right shows 'ENG IN' and the date '19-10-2024'.

- Now create a deploy-job and copy the clone-job.
- Now install the terraform in execute shell.
- Now add the 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
sudo chmod +x data.sh
terraform init
terraform fmt
terraform validate
terraform plan
terraform apply --auto-approve
```



- Edit the **sudoers** File:
- You need to add the Jenkins user to the **sudoers** file to allow it to execute commands without a password.

Sudo visudo

- Add the Following Line ->At the end of the file
- jenkins ALL=(ALL) NOPASSWD: ALL**
- Now restart the Jenkins with the command (**sudo systemctl restart Jenkins**).

- Now click on build the job was success.

Status

deploy-job

clone the terraform script for launch wordpress

Permalinks

- Last build (#7), 5 min 25 sec ago
- Last stable build (#7), 5 min 25 sec ago
- Last successful build (#7), 5 min 25 sec ago
- Last failed build (#6), 6 min 49 sec ago
- Last unsuccessful build (#6), 6 min 49 sec ago
- Last completed build (#7), 5 min 25 sec ago

Builds

Build #	Time	Status
#7	11:00 am	Success

- Here the EC2 instances was automatically launched with help of terraform script.
- Also create the VPC, Subnet, Internet Gateway, Route Table, Security Groups with help of terraform script.

Instances (1/4)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability
wordpress	i-0cc60f3e240b7cc62	Terminated	t2.micro	-	View alarms +	us-east-1a
wordpress	i-01ebf838d76e14623	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a
yash-method-7	i-0fb781d50a49deaf6	Terminated	t2.micro	-	View alarms +	us-east-1b

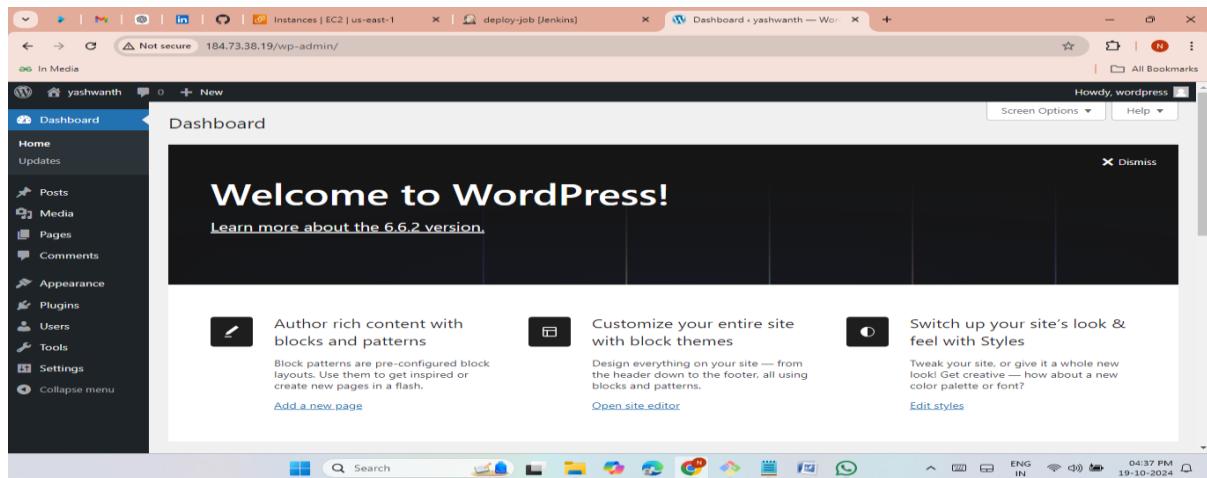
i-01ebf838d76e14623 (wordpress)

Details

Instance summary

Instance ID i-01ebf838d76e14623 (wordpress)	Public IPv4 address 184.73.38.19 open address	Private IPv4 addresses 10.0.1.176
IPv6 address -	Instance state Running	Public IPv4 DNS -

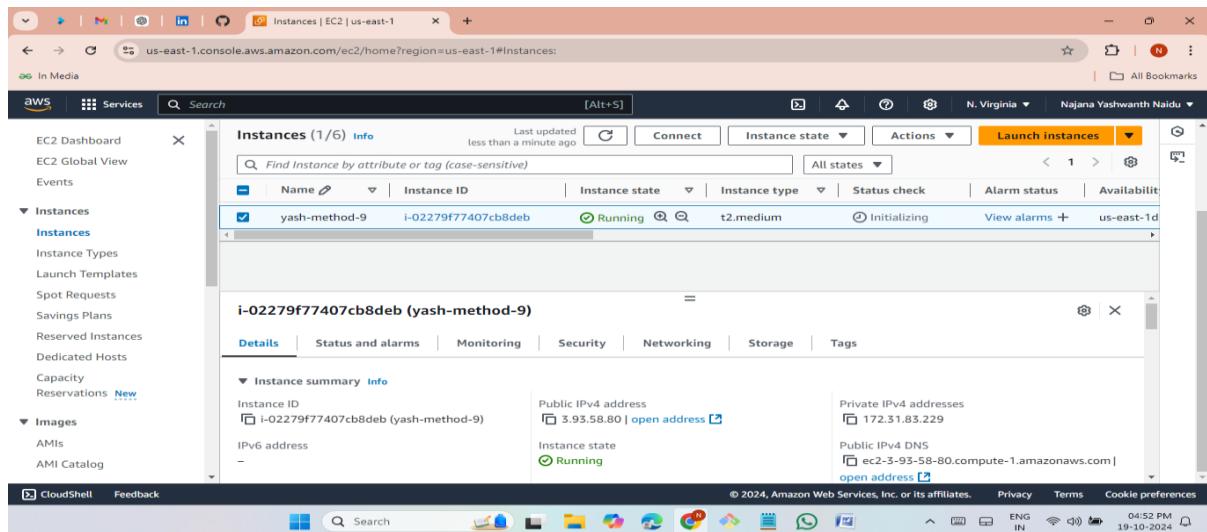
- Here the wordpress is launched successfully.



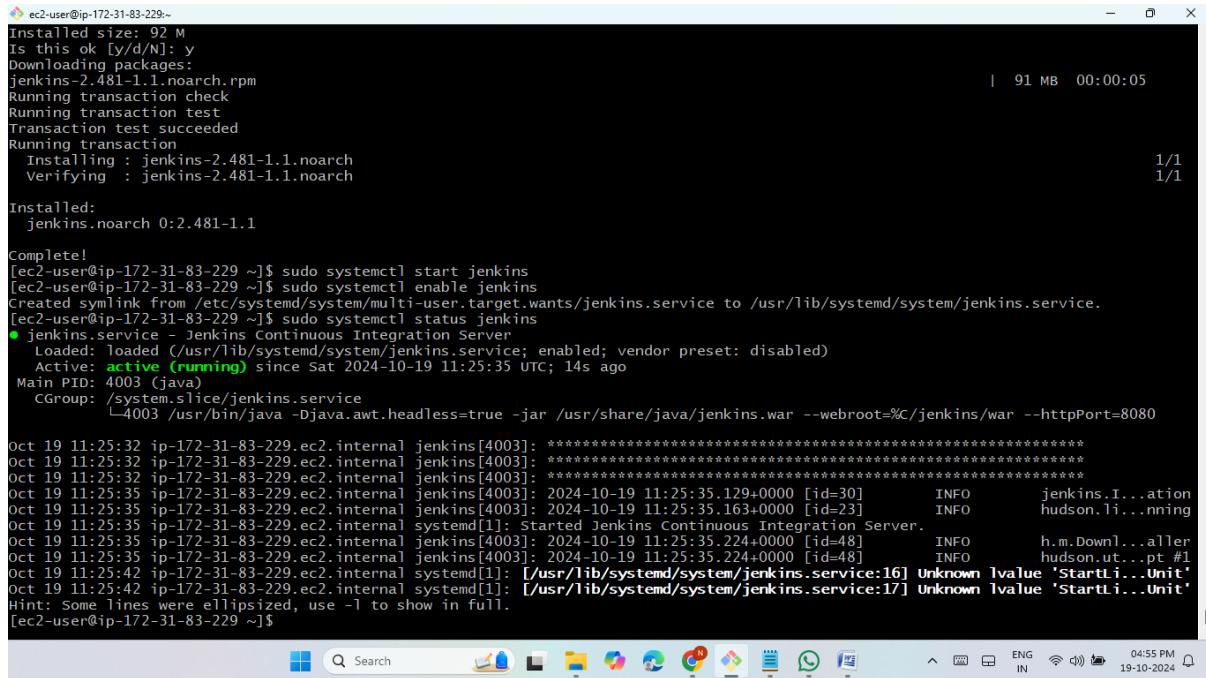
METHOD-9

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.

- First launch the EC2 instance with the Jenkins port number(8080).



- Now Install git and Jenkins.
- Now start and enable the Jenkins.



```

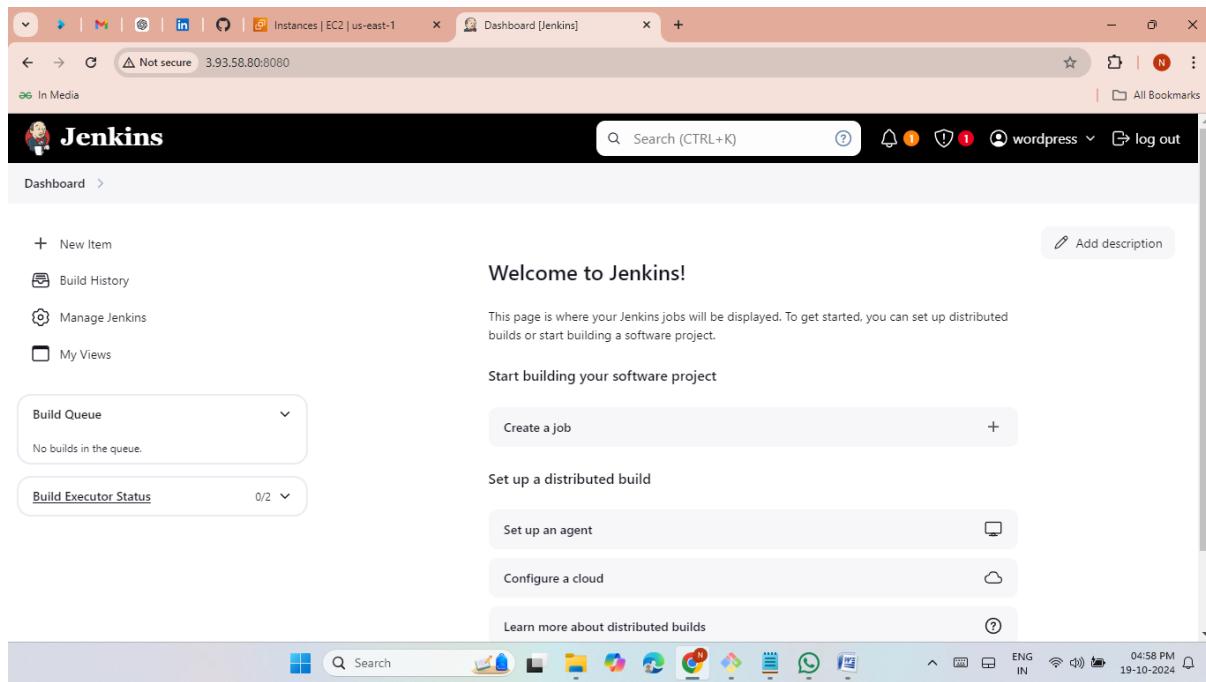
ec2-user@ip-172-31-83-229:~$ sudo yum install jenkins
Installed size: 92 M
Is this ok [y/d/N]: y
Downloading packages:
jenkins-2.481-1.1.noarch.rpm
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : jenkins-2.481-1.1.noarch
  Verifying   : jenkins-2.481-1.1.noarch
                                           1/1
                                           1/1

Installed:
  jenkins.noarch 0:2.481-1.1

Complete!
[ec2-user@ip-172-31-83-229 ~]$ sudo systemctl start jenkins
[ec2-user@ip-172-31-83-229 ~]$ 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-172-31-83-229 ~]$ 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 Sat 2024-10-19 11:25:35 UTC; 14s ago
       PID: 4003 (java)
      CGroup: /system.slice/jenkins.service
              └─4003 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=%C/jenkins/war --httpPort=8080

Oct 19 11:25:32 ip-172-31-83-229.ec2.internal jenkins[4003]: ****
Oct 19 11:25:32 ip-172-31-83-229.ec2.internal jenkins[4003]: ****
Oct 19 11:25:32 ip-172-31-83-229.ec2.internal jenkins[4003]: ****
Oct 19 11:25:35 ip-172-31-83-229.ec2.internal jenkins[4003]: 2024-10-19 11:25:35.129+0000 [id=30]           INFO      jenkins.I...
Oct 19 11:25:35 ip-172-31-83-229.ec2.internal jenkins[4003]: 2024-10-19 11:25:35.163+0000 [id=23]           INFO      hudson.li...
Oct 19 11:25:35 ip-172-31-83-229.ec2.internal systemd[1]: Started Jenkins Continuous Integration Server.
Oct 19 11:25:35 ip-172-31-83-229.ec2.internal jenkins[4003]: 2024-10-19 11:25:35.224+0000 [id=48]           INFO      h.m.Down...aller
Oct 19 11:25:35 ip-172-31-83-229.ec2.internal jenkins[4003]: 2024-10-19 11:25:35.224+0000 [id=48]           INFO      hudson.ut...pt #1
Oct 19 11:25:42 ip-172-31-83-229.ec2.internal systemd[1]: [/usr/lib/systemd/system/jenkins.service:16] unknown lvalue 'StartLi...
Oct 19 11:25:42 ip-172-31-83-229.ec2.internal systemd[1]: [/usr/lib/systemd/system/jenkins.service:17] unknown lvalue 'StartLi...
Hint: Some lines were ellipsized, use -l to show in full.
[ec2-user@ip-172-31-83-229 ~]$ 
```

- Here the Jenkins was launched successfully.



➤ Now install the AWS credentials plugin.

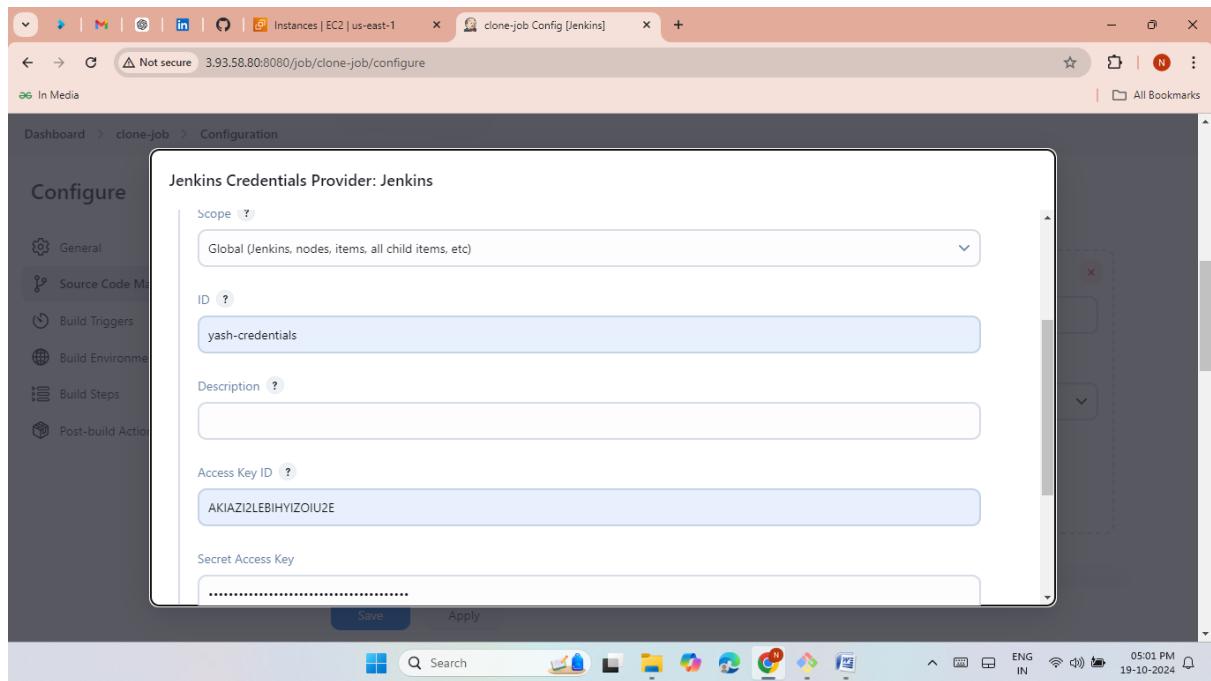
The screenshot shows the Jenkins Plugin Manager interface. On the left, there's a sidebar with options like 'Updates', 'Available plugins' (which is selected), 'Installed plugins', 'Advanced settings', and 'Download progress'. The main area has a search bar at the top with 'aws' typed in. Below it is a table with columns for 'Install', 'Name', and 'Released'. There are three entries:

- AWS Credentials**: Version 231.v08a_59f17d742, released 6 months and 25 days ago. Description: 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.
- Amazon Web Services SDK :: Minimal**: Version 1.12.772-474.v7ff79a_2046a_fb_, released 5 days and 2 hours ago. Description: Minimal modules for the AWS SDK for Java.
- Amazon Web Services SDK :: EC2**: Version 1.12.772-474.v7ff79a_2046a_fb_, released 5 days and 2 hours ago. Description: EC2 module for the AWS SDK for Java.

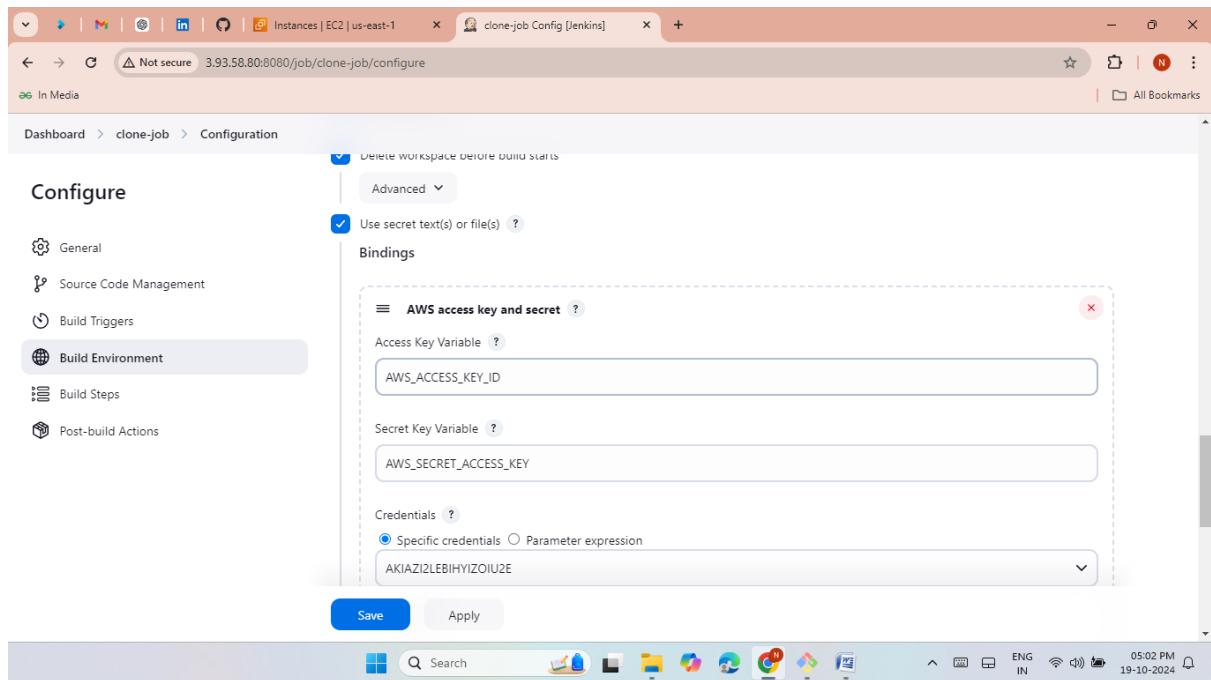
➤ Now create a clone-job and clone the terraform script form github.

The screenshot shows the Jenkins job configuration page for 'clone-job Config'. On the left, there's a sidebar with 'General', 'Source Code Management' (selected), 'Build Triggers', 'Build Environment', 'Build Steps', and 'Post-build Actions'. The main area is titled 'Source Code Management'. It shows a 'Git' configuration with a 'Repository URL' set to 'https://github.com/Yashwanth-najana/wordpress-terraform.git' and a 'Credentials' dropdown set to '- none -'. At the bottom are 'Save' and 'Apply' buttons.

- Then create the AWS credentials.



- Now connect the access keys and secret keys.
➤ Now apply the changes.



- Here the clone job is success.

The screenshot shows the Jenkins interface for the 'clone-job' project. The top navigation bar includes links for 'Instances | EC2 | us-east-1', 'clone-job [Jenkins]', and 'All Bookmarks'. The main content area displays the 'clone-job' configuration with a status bar showing 'clone the terraform script for launch wordpress'. A prominent 'Build Now' button is highlighted with a red box. Other options like 'Configure', 'Delete Project', and 'Rename' are also visible.

- Now create new job and connect to the clone-job.
- Now Add the Build Periodically and Poll SCM.

The screenshot shows the Jenkins interface for the 'deploy-job' configuration. The 'Build Triggers' section is active, with 'Poll SCM' selected and highlighted with a red box. A warning message below states: '⚠ Do you really mean "every minute" when you say "*****"? Perhaps you meant "H * * * *" to poll once per hour'. The bottom of the screen shows a Windows taskbar with various icons and system status.

- Now install the terraform in execute shell.
- Now add the 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

sudo chmod + data.sh

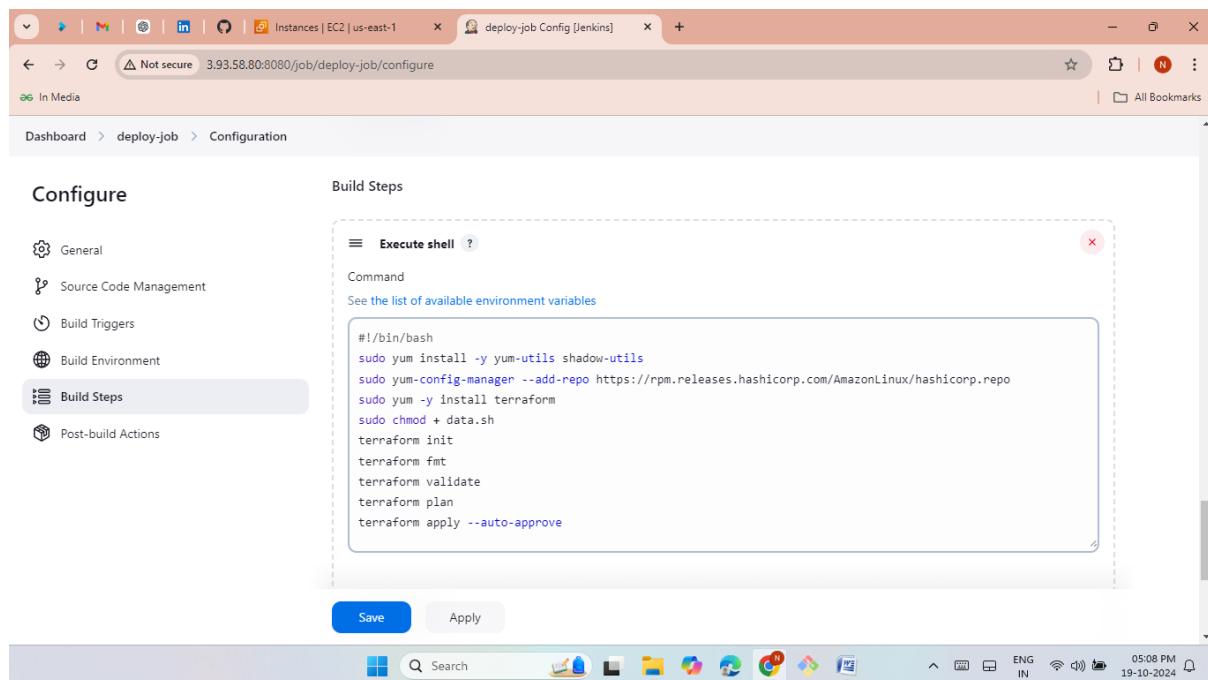
terraform init

terraform fmt

terraform validate

terraform plan

terraform apply --auto-approve
```

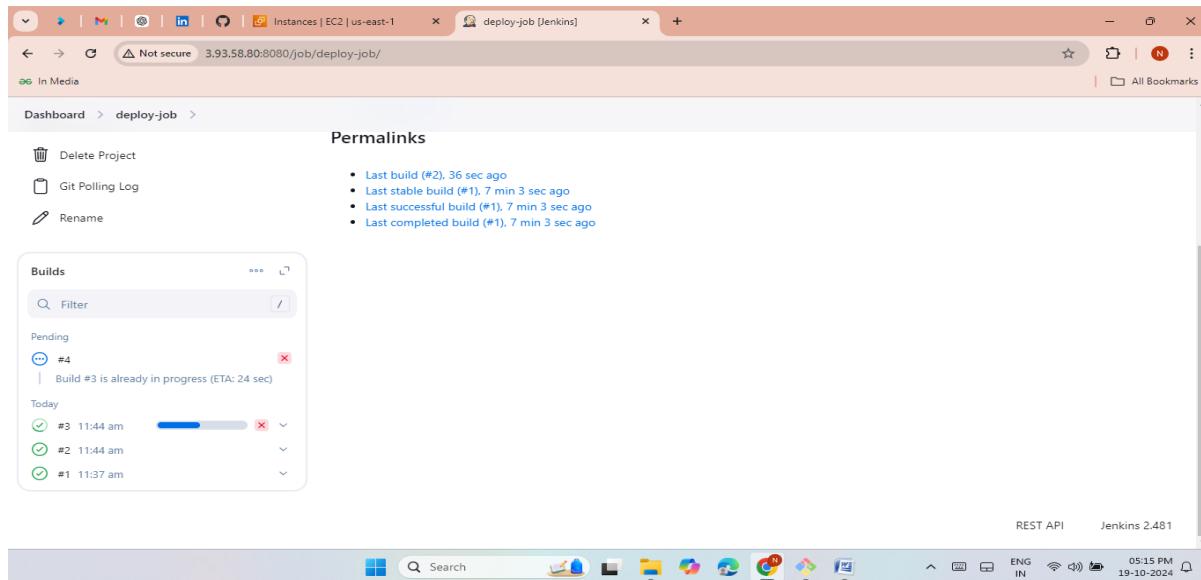


- Give permissions to jenkins.

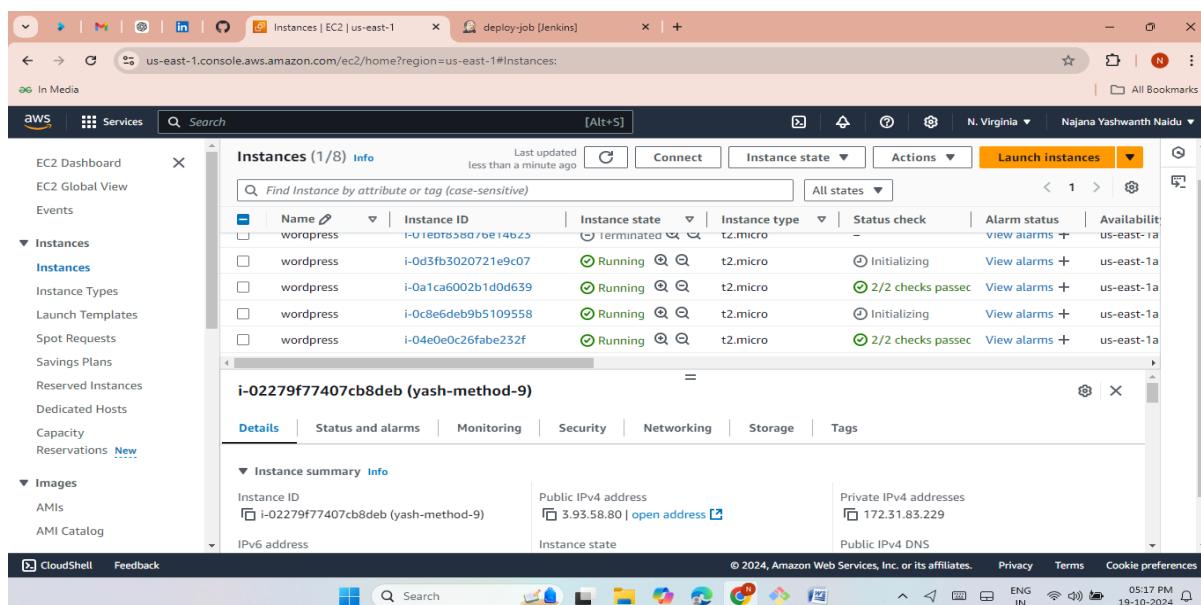
Sudo visudo

jenkins ALL=(ALL) NOPASSWD: ALL

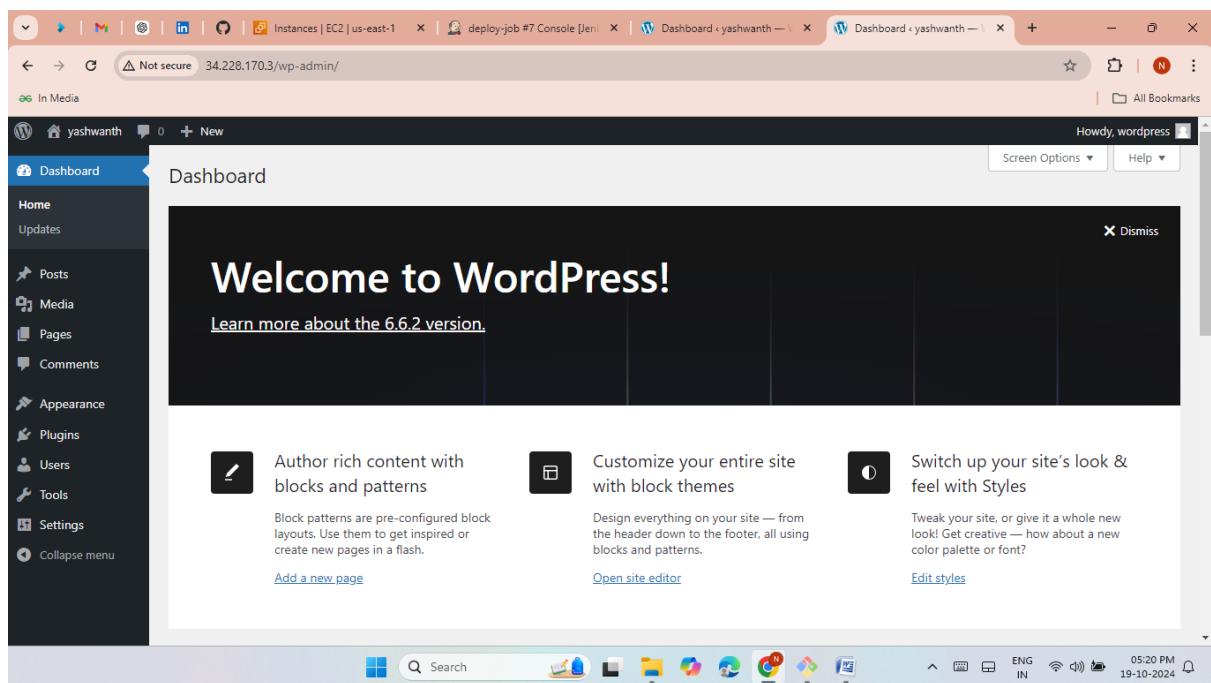
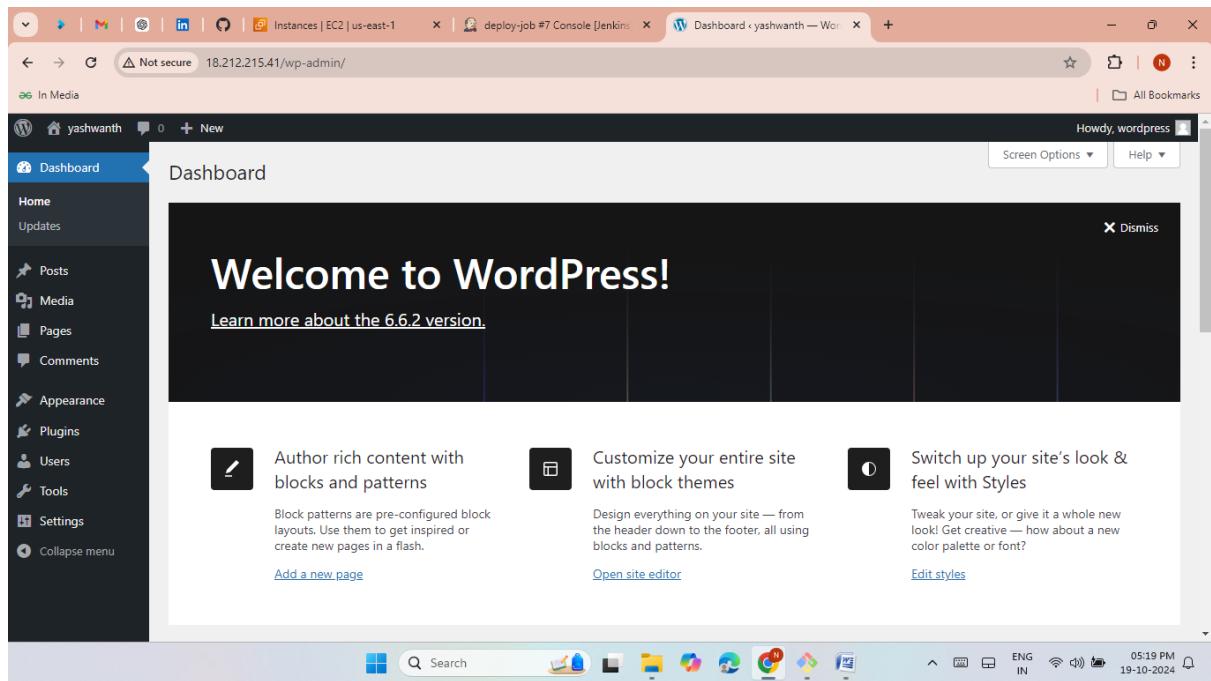
- Now restart the Jenkins with the command (**sudo systemctl restart Jenkins**).
- Now click on build the job was success.
- It runs every MINUTE HOUR DOM(DAY OF MONTH) MONTH DOW(DAY OF WEEK).
- Because I have given (* * * * *).



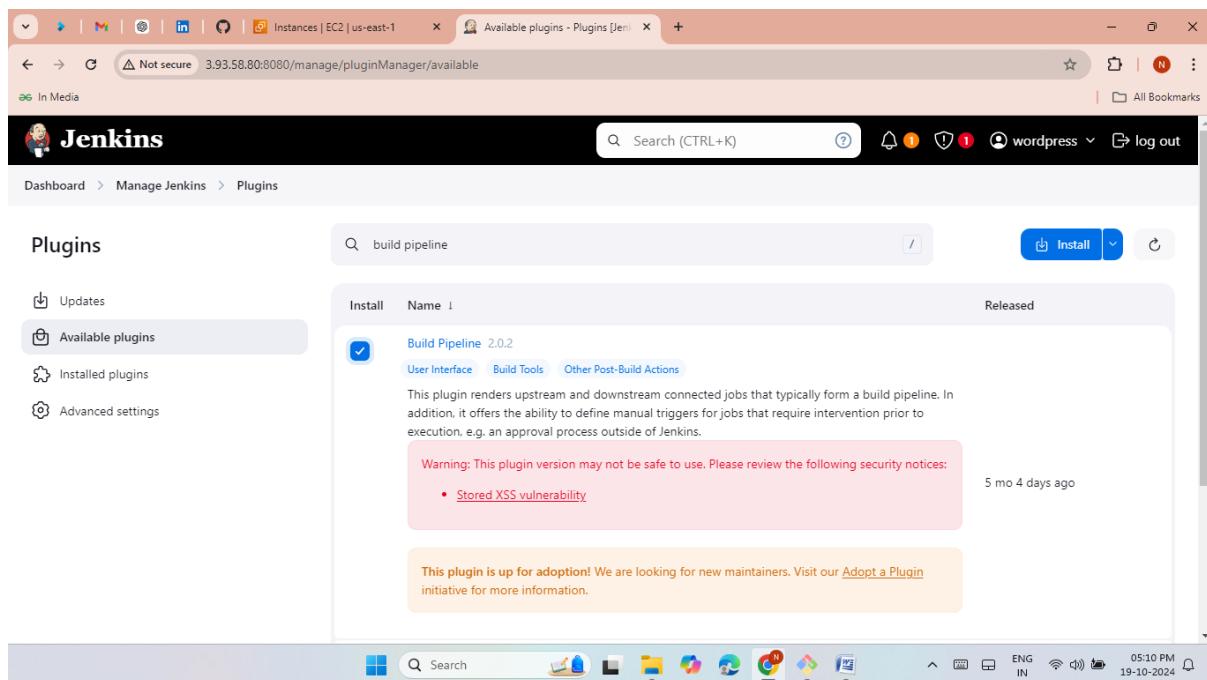
- Here the instances were automatically launched every one min.
- Also create the VPC, Subnet, Internet Gateway, Route Table, Security Groups with help of terraform script.
- Because of Build Periodically and Poll SCM.



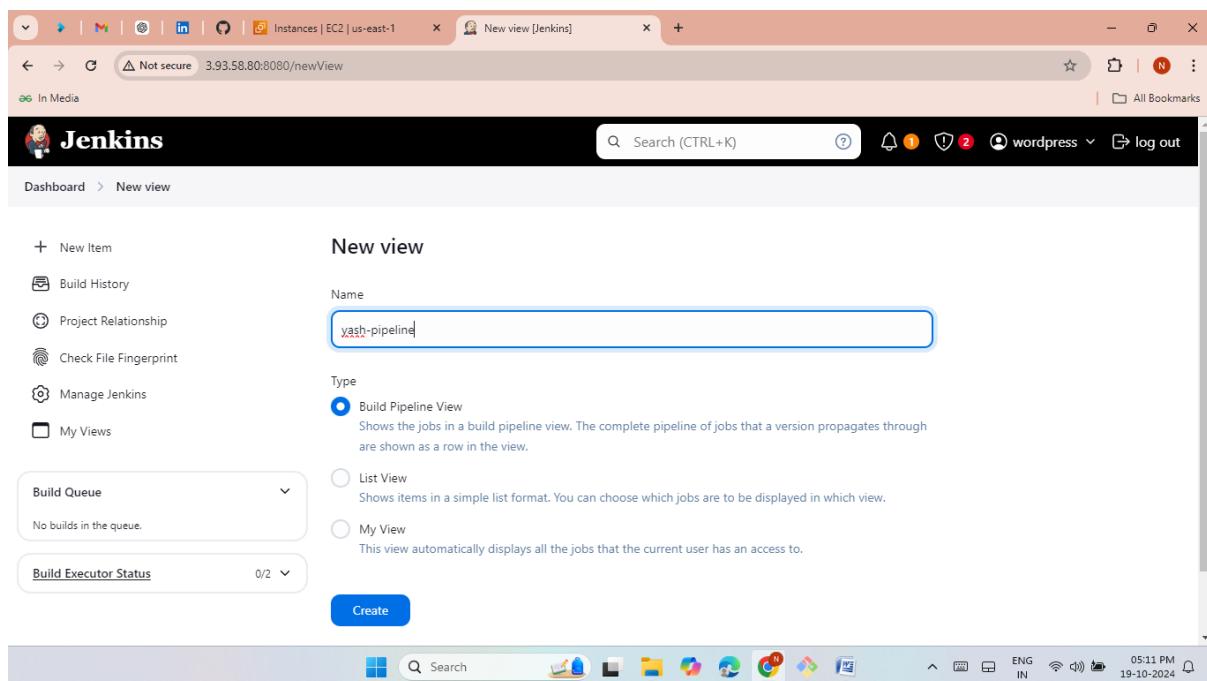
- Here the all wordpress applications was launced.



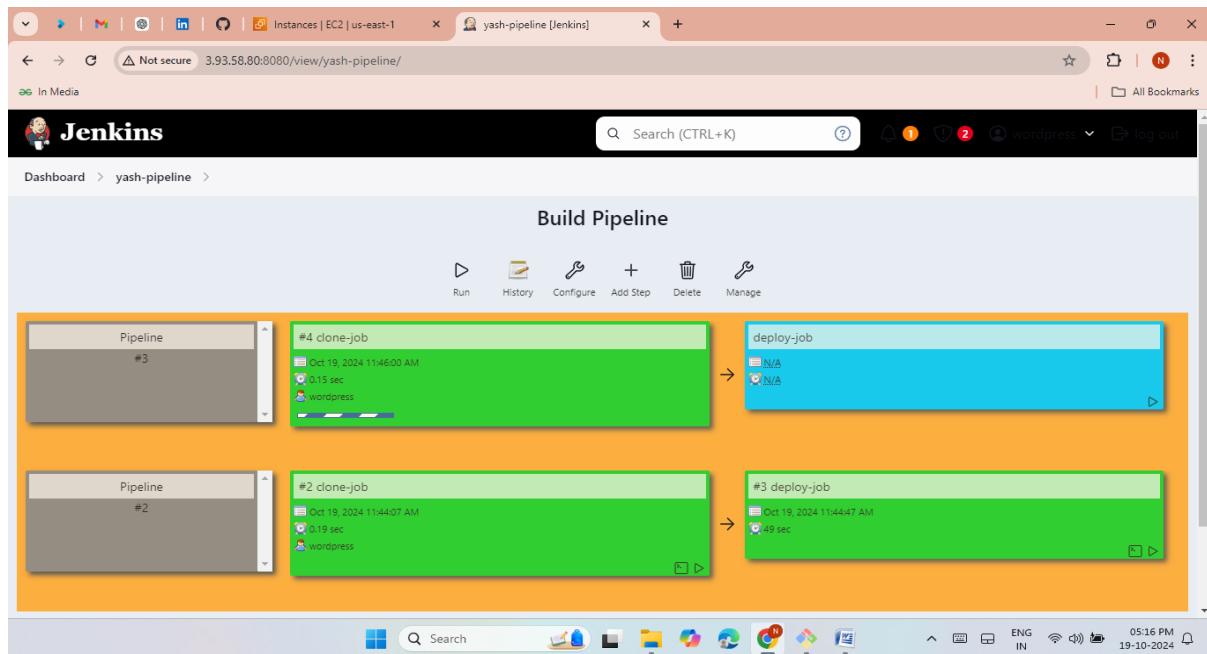
- Now install the build pipeline plugin for create a Jenkins pipeline.



- Now create a Jenkins pipeline.



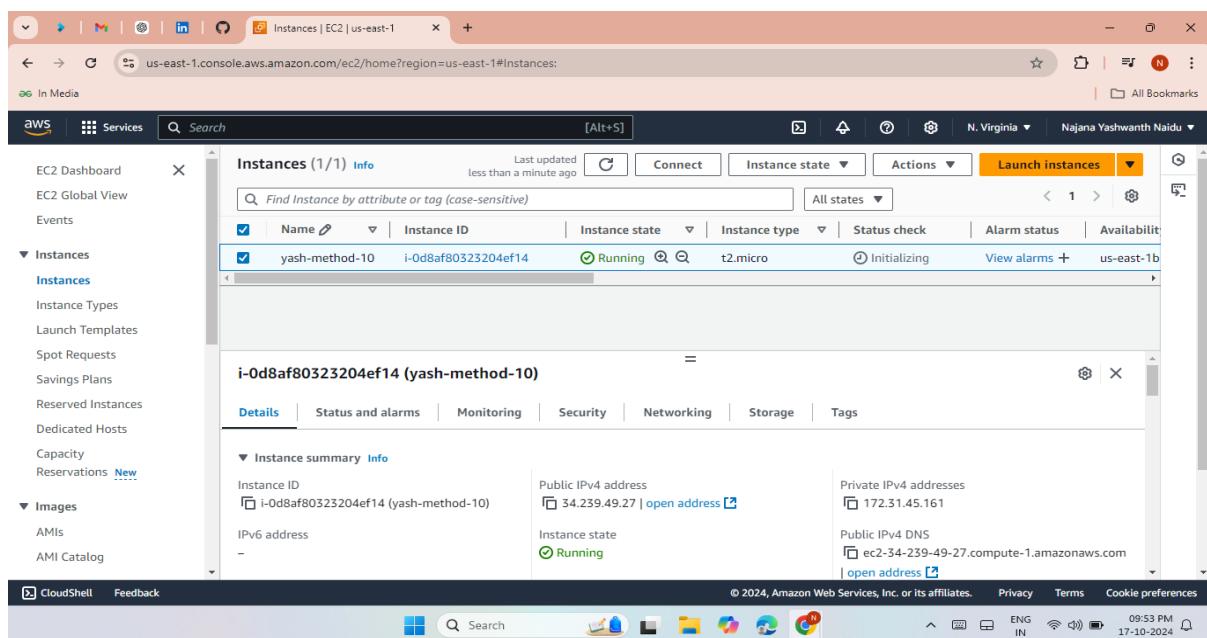
- Here the build pipeline is successfully created.



METHOD-10

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

- First launch an EC2 instance for push the docker image into dockerhub.



- Now install docker.
- Now start and enable the docker.
- Give permissions to the docker with the command(**sudo chmod 666 var/run/docker.sock**).
- Now install git and clone the docker-compose file.

```
ec2-user@ip-172-31-45-161:~ 
Verifying : pigz-2.3.4-1.amzn2.0.1.x86_64          2/5
Verifying : runc-1.1.14-1.amzn2.x86_64            3/5
Verifying : containerd-1.7.22-1.amzn2.0.2.x86_64   4/5
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-172-31-45-161 ~]$ sudo systemctl start docker
[ec2-user@ip-172-31-45-161 ~]$ 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-172-31-45-161 ~]$ 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 16:26:09 UTC; 14s ago
    Docs: https://docs.docker.com
  Main PID: 3509 (dockerd)
  CGroup: /system.slice/docker.service
          └─3509 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --default-ulimit nofile=32768:65536

Oct 17 16:26:09 ip-172-31-45-161.ec2.internal systemd[1]: Starting Docker Application Container Engine...
Oct 17 16:26:09 ip-172-31-45-161.ec2.internal dockerd[3509]: time="2024-10-17T16:26:09.587677238Z" level=info msg="Starting up"
Oct 17 16:26:09 ip-172-31-45-161.ec2.internal dockerd[3509]: time="2024-10-17T16:26:09.65351920Z" level=info msg="Loading con...art."
Oct 17 16:26:09 ip-172-31-45-161.ec2.internal dockerd[3509]: time="2024-10-17T16:26:09.896862213Z" level=info msg="Loading con...one."
Oct 17 16:26:09 ip-172-31-45-161.ec2.internal dockerd[3509]: time="2024-10-17T16:26:09.913710622Z" level=info msg="Docker daem...5.0.6
Oct 17 16:26:09 ip-172-31-45-161.ec2.internal dockerd[3509]: time="2024-10-17T16:26:09.914087948Z" level=info msg="Daemon has ...tion"
Oct 17 16:26:09 ip-172-31-45-161.ec2.internal dockerd[3509]: time="2024-10-17T16:26:09.957855579Z" level=info msg="API listen ...sock"
Oct 17 16:26:09 ip-172-31-45-161.ec2.internal systemd[1]: Started Docker Application Container Engine.
Hint: Some lines were ellipsized, use -l to show in full.
[ec2-user@ip-172-31-45-161 ~]$ sudo chmod 666 /var/run/docker.sock
[ec2-user@ip-172-31-45-161 ~]$ ls
dockerfile
[ec2-user@ip-172-31-45-161 ~]$ |
```



09:57 PM
17-10-2024

- Now install docker-compose and give permissions.
- Now run the docker-compose file with the command (**docker-compose up -d**).
- Now we can see the docker images.

```
ec2-user@ip-172-31-45-161:~/dockerfile
✓ cc366ac8ba10 Pull complete          18.6s
✓ 7694a6cd58cb Pull complete          19.8s
✓ 068ff6a4ec7b4 Pull complete          19.8s
✓ 48201d9d6f62 Pull complete          19.8s
✓ ec91d8faf678 Pull complete          19.9s
✓ 77d093e1881c Pull complete          21.0s
✓ 8495921eba41 Pull complete          21.8s
✓ a804d217b765 Pull complete          21.9s
✓ c407ddc434df Pull complete          21.9s
✓ 98803a5b897b Pull complete          21.9s
✓ 6a9e25e0066f Pull complete          23.3s
✓ a059aaee3687 Pull complete          23.3s
✓ 16ba04bd2799 Pull complete          23.4s
✓ db_12_layers [██████████] 0B/0B     Pulled          19.7s
✓ 54fec2fa59d0 Pull complete          6.3s
✓ bcc6c6145912 Pull complete          6.3s
✓ 951c3d959c9d Pull complete          6.7s
✓ 05de4d0e206e Pull complete          7.2s
✓ 319f0394ef42 Pull complete          7.2s
✓ d9185034607b Pull complete          8.8s
✓ 013a9c64dad0 Pull complete          8.9s
✓ 96d4c3d31f9f Pull complete          8.9s
✓ 785bc90808da Pull complete          19.2s
✓ 1339cf094729 Pull complete          19.2s
✓ beb8f531cc37 Pull complete          19.2s
✓ 2b40c9f6a918 Pull complete          19.3s
[+] Running 4/4
✓ Network dockerfile_default      Created          0.1s
✓ Volume "dockerfile_db_data"     Created          0.0s
✓ Container dockerfile-wordpress-1 Started          1.2s
✓ Container dockerfile-db-1       Started          1.2s
[ec2-user@ip-172-31-45-161 dockerfile]$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
70b8cc214dfe wordpress:latest "docker-entrypoint.s..." 7 seconds ago Up 6 seconds 0.0.0.0:80->80/tcp, :::80->80/tcp dockerfi
1e-wordpress-1
09c5a2f3a687 mysql:8.0.19 "docker-entrypoint.s..." 7 seconds ago Up 6 seconds 3306/tcp, 33060/tcp dockerfi
1e-db-1
[ec2-user@ip-172-31-45-161 dockerfile]$ |
```



10:01 PM
17-10-2024

- Now run the following commands to push the docker images into DockerHub platform.

docker login (give the DockerHub user and password).

```
docker tag <image name> <username>/<repo name>:<tag>
```

```
docker push <username>/<repo name>:<tag>
```

The screenshot shows a terminal window on a Windows operating system. The command `docker tag` is used to tag the local image with the DockerHub repository name and tag. The command `docker push` is then used to upload the tagged image to the DockerHub registry. The output of the push command shows the progress and success of the upload.

```

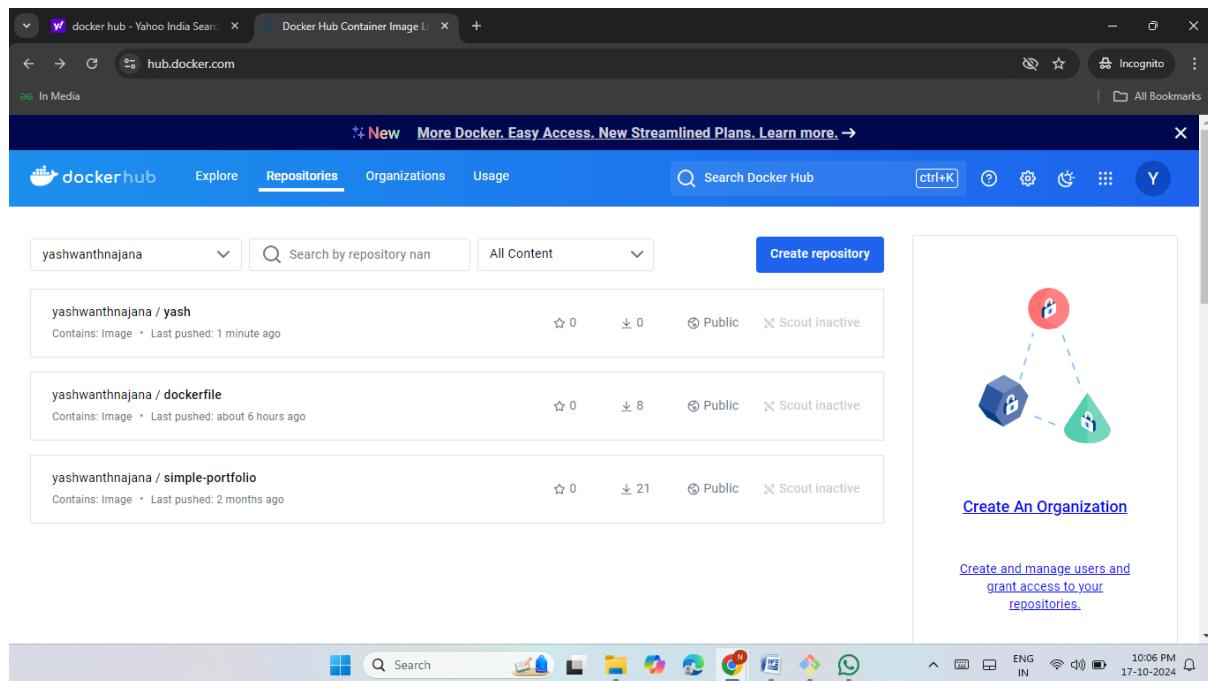
ec2-user@ip-172-31-45-161:~/dockerfile
Username: yashwanthnajana
Password:
WARNING! Your password will be stored unencrypted in /home/ec2-user/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login succeeded
[ec2-user@ip-172-31-45-161 dockerfile]$ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
wordpress latest d9da5214e7fd 43 hours ago 699MB
mysql 8.0.19 0c27e8e5fcfa 4 years ago 546MB
[ec2-user@ip-172-31-45-161 dockerfile]$ docker tag wordpress yashwanthnajana/yash:latest
[ec2-user@ip-172-31-45-161 dockerfile]$ docker push yashwanthnajana/yash:latest
The push refers to repository [docker.io/yashwanthnajana/yash]
63aadcc9a4fb: Mounted from library/wordpress
3fa6ddd7e7e7: Mounted from library/wordpress
d70945719213: Mounted from library/wordpress
b31b42354bdd: Mounted from library/wordpress
823648cf232d: Mounted from library/wordpress
9845fa2cc8d3: Mounted from library/wordpress
80e4ee517df9: Mounted from library/wordpress
1ccdd0ee39050: Mounted from library/wordpress
59b22bcf9712: Mounted from library/wordpress
273fbcb5f82af: Mounted from library/wordpress
56b253cc1ed7: Mounted from library/wordpress
ec9b13a1942d: Mounted from library/wordpress
34308ec5d204: Mounted from library/wordpress
d717e9cb9649: Mounted from library/wordpress
8fa01518e07f: Mounted from library/wordpress
d74e481bb53b: Mounted from library/wordpress
36d96f2f1067: Mounted from library/wordpress
bc6a13afa75e: Mounted from library/wordpress
54634b9aecb1: Mounted from library/wordpress
fa6de34729f6: Mounted from library/wordpress
98b5f35ea9d3: Mounted from library/wordpress
latest: digest: sha256:8b036d0538685ed312b4647c42539e9de35e3345e371742031af7dc954886052 size: 4711
[ec2-user@ip-172-31-45-161 dockerfile]$

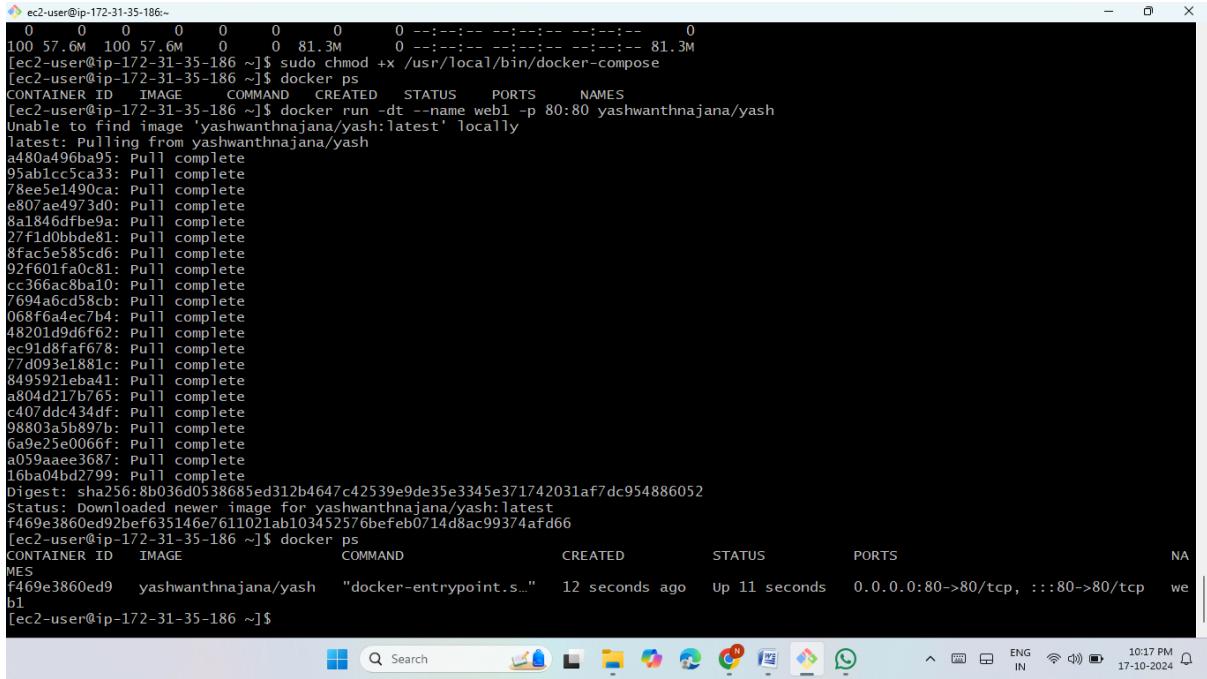
[Windows Taskbar icons: Search, File Explorer, Task View, Taskbar Buttons, Network, Battery, Volume, Language, Date/Time]

```

- Here the docker image is pushed successfully (yash).



- Launch another instances and pull the docker image and run wordpress application (it is for checking purpose).
- Successfully hosted the wordpress.

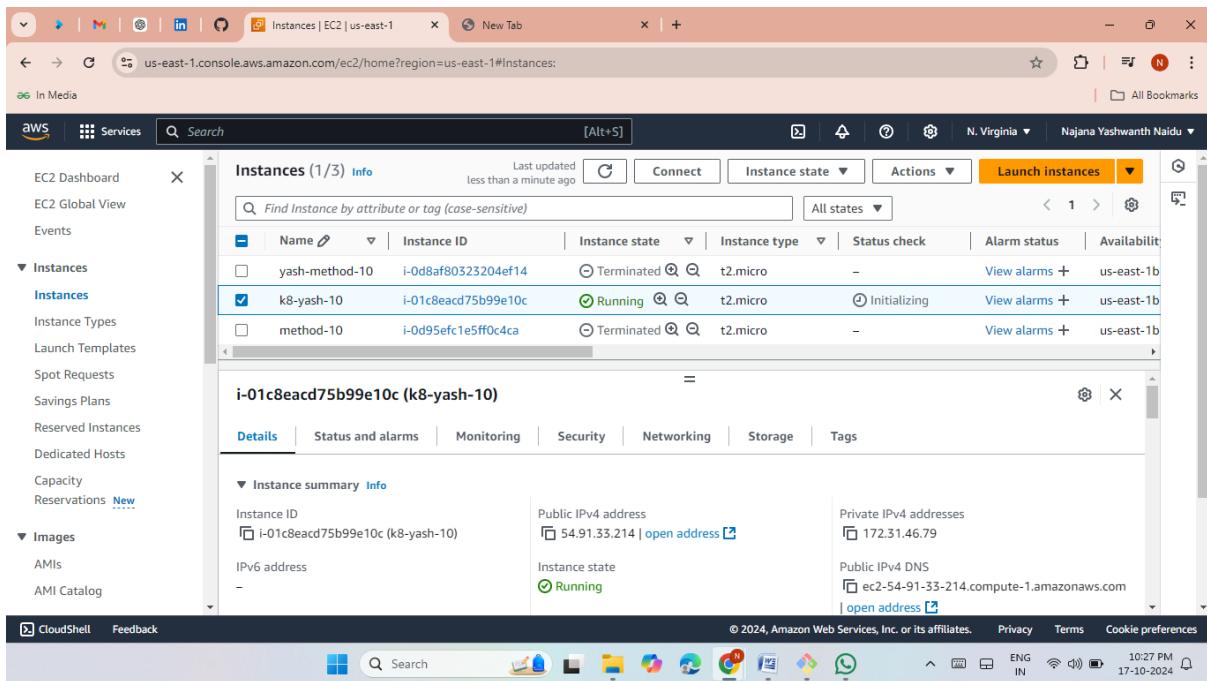


```

ec2-user@ip-172-31-35-186:~$ sudo chmod +x /usr/local/bin/docker-compose
[ec2-user@ip-172-31-35-186 ~]$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
[ec2-user@ip-172-31-35-186 ~]$ docker run -dt --name web1 -p 80:80 yashwanthnajana/yash
Unable to find image 'yashwanthnajana/yash:latest'
latest: Pulling from yashwanthnajana/yash
a480a496ba95: Pull complete
95ab1c5ca33: Pull complete
78ee5e1490ca: Pull complete
e807ae4973d0: Pull complete
8a1846dfbe9a: Pull complete
27f1d0bbde81: Pull complete
8fac5e585cd6: Pull complete
92f601fa0c81: Pull complete
cc366ac8ba10: Pull complete
7694a6cd58cb: Pull complete
068f0a4ec7b4: Pull complete
48201d9d6f62: Pull complete
ec91d8faf678: Pull complete
77d093e1881c: Pull complete
8495921eba41: Pull complete
a804d217b765: Pull complete
c407ddc434df: Pull complete
98803a5b897b: Pull complete
6a9e25e0066f: Pull complete
a059aaee3687: Pull complete
16ba04bd2799: Pull complete
Digest: sha256:8b036d0538685ed312b4647c42539e9de35e3345e371742031af7dc954886052
Status: Downloaded newer image for yashwanthnajana/yash:latest
f469e3860ed92bef635146e7611021ab103452576befeb0714d8ac99374af6d6
[ec2-user@ip-172-31-35-186 ~]$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
MES
f469e3860ed9 yashwanthnajana/yash "docker-entrypoint.s..." 12 seconds ago Up 11 seconds 0.0.0.0:80->80/tcp, ::80->80/tcp we
b1
[ec2-user@ip-172-31-35-186 ~]$

```

- Now launch an EC2 instance for hosting the wordpress in kubernetes.



Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability
yash-method-10	i-0d8af80323204ef14	Terminated	t2.micro	-	View alarms +	us-east-1b
k8-yash-10	i-01c8ecd75b99e10c	Running	t2.micro	Initializing	View alarms +	us-east-1b
method-10	i-0d95efc1e5ff0c4ca	Terminated	t2.micro	-	View alarms +	us-east-1b

- Now install aws cli (in linux by default aws cli is there. If we want updated version, we can download).

```

ec2-user@ip-172-31-46-79:~$ curl -O https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip
ec2-user@ip-172-31-46-79:~$ unzip awscli-exe-linux-x86_64.zip
ec2-user@ip-172-31-46-79:~$ ./aws/install
[...]
Digest: sha256:8b036d0538685ed312b4647c42539e9de35e3345e371742031af7dc954886052
Status: Downloaded newer image for yashwanthnajana/yash:latest
f469e3860ed92bef635146e7611021ab103452576befeb0714d8ac99374afdf66
[ec2-user@ip-172-31-35-186 ~]$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
f469e3860ed9 yashwanthnajana/yash "docker-entrypoint.s..." 12 seconds ago Up 11 seconds 0.0.0.0:80->80/tcp, ::80->80/tcp web
[ec2-user@ip-172-31-35-186 ~]$ Connection to ec2-3-84-214-183.compute-1.amazonaws.com closed by remote host.
Connection to ec2-3-84-214-183.compute-1.amazonaws.com closed.

yawsaw@LAPTOP-GM32QJHO MINGW64 ~/keys
$ ssh -i "yash.pem" ec2-user@ec2-54-91-33-214.compute-1.amazonaws.com
The authenticity of host 'ec2-54-91-33-214.compute-1.amazonaws.com (54.91.33.214)' can't be established.
ED25519 key fingerprint is SHA256:syx58Y/94BwOEuarQoSx6BhxCavbGrXspxm4Pkqy9o.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-54-91-33-214.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-172-31-46-79 ~]$ 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-172-31-46-79 ~]$ 

```

- Create role with all admin permissions and attach it to the EC2 instance.

The screenshot shows the 'Modify IAM role' page in the AWS Management Console. The instance ID is set to 'i-01c8eacd75b99e10c'. In the 'IAM role' section, a dropdown menu is open, showing 'method-10' as the selected option. A 'Create new IAM role' button is also visible. At the bottom right of the dialog, there is a 'Cancel' button and an orange 'Update IAM role' button.

- Now first Install kubectl package from Google.
- Then install kubectl.

```
ec2-user@ip-172-31-46-79:~$ --> Finished Dependency Resolution
Dependencies Resolved

=====
Package           Arch      Version            Repository      Size
=====
Installing:
kubectl          x86_64   1.31.1-150500.1.1   kubernetes    11 M

Transaction Summary
=====
Install 1 Package

Total download size: 11 M
Installed size: 54 M
Downloading packages:
warning: /var/cache/yum/x86_64/2/kubernetes/packages/kubectl-1.31.1-150500.1.1.x86_64.rpm: Header V3 RSA/SHA256 Signature, key ID 9a296436: NOKEY
Public key for kubectl-1.31.1-150500.1.1.x86_64.rpm is not installed
kubectl-1.31.1-150500.1.1.x86_64.rpm                                | 11 MB  00:00:00
Retrieving key from https://pkgs.k8s.io/core:/stable:/v1.31/rpm/repo/repodata/repomd.xml.key
Importing GPG key 0x9a296436:
  Userid : "jsv:kubernetes OBS Project <jsv:kubernetes@build.opensuse.org>"
  Fingerprint: da15 b144 86cd 377b 9e87 6e1a 2346 54da 9a29 6436
  From   : https://pkgs.k8s.io/core:/stable:/v1.31/rpm/repo/repodata/repomd.xml.key
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : kubectl-1.31.1-150500.1.1.x86_64                               1/1
  Verifying  : kubectl-1.31.1-150500.1.1.x86_64                               1/1

Installed:
  kubectl.x86_64 0:1.31.1-150500.1.1

Complete!
[ec2-user@ip-172-31-46-79 ~]$
```

- Now install kops from Google.

```
ec2-user@ip-172-31-46-79:~$ Advanced Commands:
diff      Diff the live version against a would-be applied version
apply     Apply a configuration to a resource by file name or stdin
patch     Update fields of a resource
replace   Replace a resource by file name or stdin
wait      Experimental: Wait for a specific condition on one or many resources
kustomize Build a kustomization target from a directory or URL

Settings Commands:
label     Update the labels on a resource
annotate  Update the annotations on a resource
completion Output shell completion code for the specified shell (bash, zsh, fish, or powershell)

Subcommands provided by plugins:

Other Commands:
api-resources Print the supported API resources on the server
api-versions  Print the supported API versions on the server, in the form of "group/version"
config       Modify kubeconfig files
plugin      Provides utilities for interacting with plugins
version      Print the client and server version information

Usage:
kubectl [flags] [options]

Use "kubectl <command> --help" for more information about a given command.
Use "kubectl options" for a list of global command-line options (applies to all commands).
[ec2-user@ip-172-31-46-79 ~]$ curl -Lo kops https://github.com/kubernetes/kops/releases/download/$(curl -s https://api.github.com/repos/kubernetes/kops/releases/latest | grep tag_name | cut -d '"' -f 4)/kops-linux-amd64
chmod +x kops
sudo mv kops /usr/local/bin/kops
Dload Upload Total Spent Left Speed
0 0 0 0 0 0 0 0 0 0 0 0
100 238M 100 238M 0 0 97.6M 0 0:00:02 0:00:02 102M
[ec2-user@ip-172-31-46-79 ~]$ chmod +x kops
[ec2-user@ip-172-31-46-79 ~]$ sudo mv kops /usr/local/bin/kops
[ec2-user@ip-172-31-46-79 ~]$
```

➤ Now run the following commands.

aws s3 mb s3://yash-k8 (to create s3 bucket to store the data).

aws s3 ls (to show the s3).

export KOPS_STATE_STORE=s3://yash-k8 (attach s3 bucket to kops for backup).

ssh-keygen (now enter).

```
ec2-user@ip-172-31-46-79:~$ aws s3 mb s3://yash-k8
ec2-user@ip-172-31-46-79:~$ aws s3 ls
2024-06-12 10:09:13 elasticbeanstalk-us-east-1-637423323663
2024-10-17 17:32:06 yash-k8
[ec2-user@ip-172-31-46-79:~]$ export KOPS_STATE_STORE=s3://yash-k8
[ec2-user@ip-172-31-46-79:~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ec2-user/.ssh/id_rsa): AC
[ec2-user@ip-172-31-46-79:~]$ aws s3 mb s3://yash-k8
make_bucket: yash-k8
[ec2-user@ip-172-31-46-79:~]$ aws s3 ls
2024-06-12 10:09:13 elasticbeanstalk-us-east-1-637423323663
2024-10-17 17:32:06 yash-k8
[ec2-user@ip-172-31-46-79:~]$ export KOPS_STATE_STORE=s3://yash-k8
[ec2-user@ip-172-31-46-79:~]$ 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's fingerprint is:
SHA256:8515vgHmWlyxhnuxlTmqQJU0R+noKYTYMDJVLekOE ec2-user@ip-172-31-46-79.ec2.internal
The key's randomart image is:
+---[RSA 2048]---+
|B+.o
|+.o
|.E. o .
|.= + o .
|= * . + S .
| . + = B *
| + @ o * o
| * o o .
| . * o.
+---[SHA256]---+
[ec2-user@ip-172-31-46-79:~]$
```

kops create cluster --name <cluster name>.k8s.local --state s3://<bucket name> --zones us-east-1b,us-east-1a --node-count 2 --yes (to create Cluster and it create VPC, Subnet, IG, Route, LoadBalancer, Target Group ,Auto Scalling Group, Instances).

kops validate cluster (to see the details od cluster).

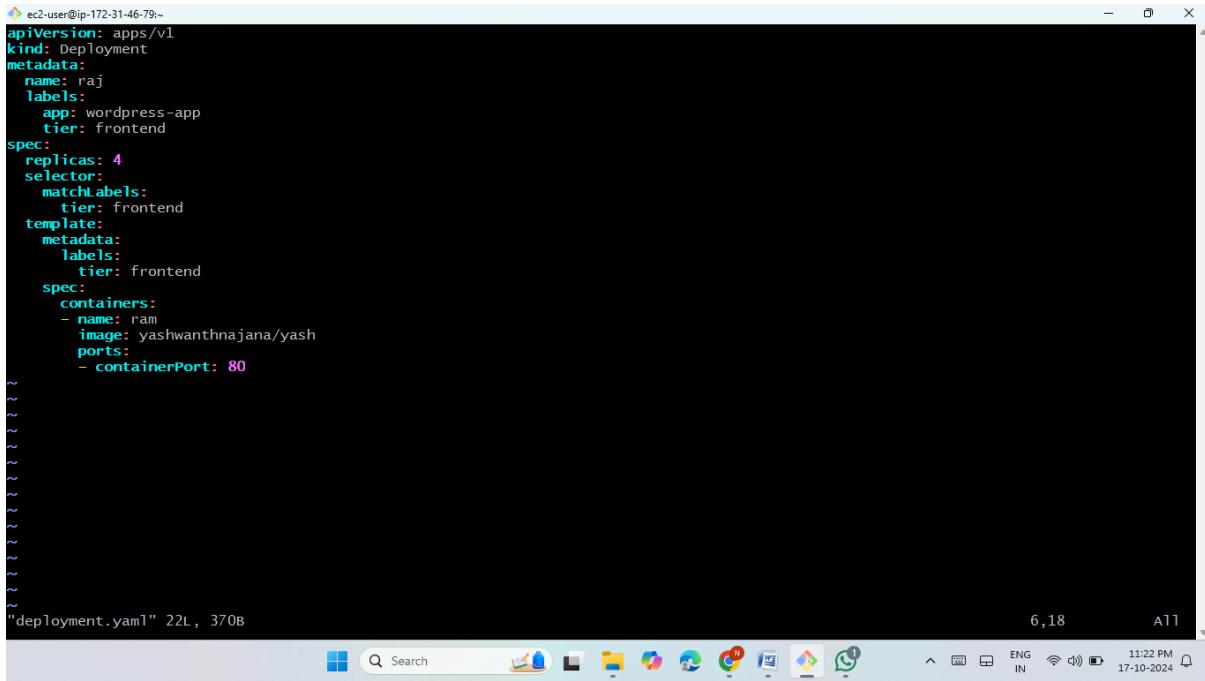
```
known, downloading hash
r1017 17:44:24.841040 32404 builder.go:312] asset "https://artifacts.k8s.io/binaries/kops/1.30.1/linux/amd64/channels" is not well-known, downloading hash
r1017 17:44:24.931319 32404 builder.go:312] asset "https://artifacts.k8s.io/binaries/kops/1.30.1/linux/arm64/channels" is not well-known, downloading hash
r1017 17:44:28.839640 32404 executor.go:113] Tasks: 0 done / 126 total; 43 can run
r1017 17:44:28.889724 32404 keypair.go:226] Issuing new certificate: "etcd-peers-ca-main"
r1017 17:44:28.915230 32404 vfs_keystorereader.go:143] CA private key was not found
r1017 17:44:28.944572 32404 keypair.go:226] Issuing new certificate: "etcd-peers-ca-events"
r1017 17:44:28.959764 32404 keypair.go:226] Issuing new certificate: "etcd-manager-ca-events"
r1017 17:44:29.017791 32404 keypair.go:226] Issuing new certificate: "etcd-clients-ca"
r1017 17:44:29.037216 32404 keypair.go:226] Issuing new certificate: "etcd-manager-ca-main"
r1017 17:44:29.059056 32404 vfs_keystorereader.go:143] CA private key was not found
r1017 17:44:29.280841 32404 keypair.go:226] Issuing new certificate: "kubernetes-ca"
r1017 17:44:29.301647 32404 keypair.go:226] Issuing new certificate: "apiserver-aggregator-ca"
r1017 17:44:29.443591 32404 keypair.go:226] Issuing new certificate: "service-account"
r1017 17:44:31.115707 32404 executor.go:113] Tasks: 47 done / 126 total; 23 can run
r1017 17:44:32.370807 32404 executor.go:113] Tasks: 66 done / 126 total; 35 can run
r1017 17:44:33.496387 32404 network_load_balancer.go:536] Waiting for load balancer "api-veera-k8s-local-vvhhts" to be created...
r1017 17:46:59.238021 32404 executor.go:113] Tasks: 101 done / 126 total; 5 can run
r1017 17:46:59.836340 32404 executor.go:113] Tasks: 106 done / 126 total; 8 can run
r1017 17:47:00.470610 32404 executor.go:113] Tasks: 114 done / 126 total; 3 can run
r1017 17:47:01.728120 32404 executor.go:113] Tasks: 117 done / 126 total; 6 can run
r1017 17:47:02.028396 32404 executor.go:113] Tasks: 123 done / 126 total; 3 can run
r1017 17:47:02.099587 32404 executor.go:113] Tasks: 126 done / 126 total; 0 can run
r1017 17:47:02.099722 32404 update_cluster.go:338] Exporting kubeconfig for cluster
kops has set your kubectl context to veera.k8s.local

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

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

[ec2-user@ip-172-31-46-79:~]$ kops validate cluster
```

- Now create a deployment.yaml file for run the dockerhub image by using **declarative manifest method**.



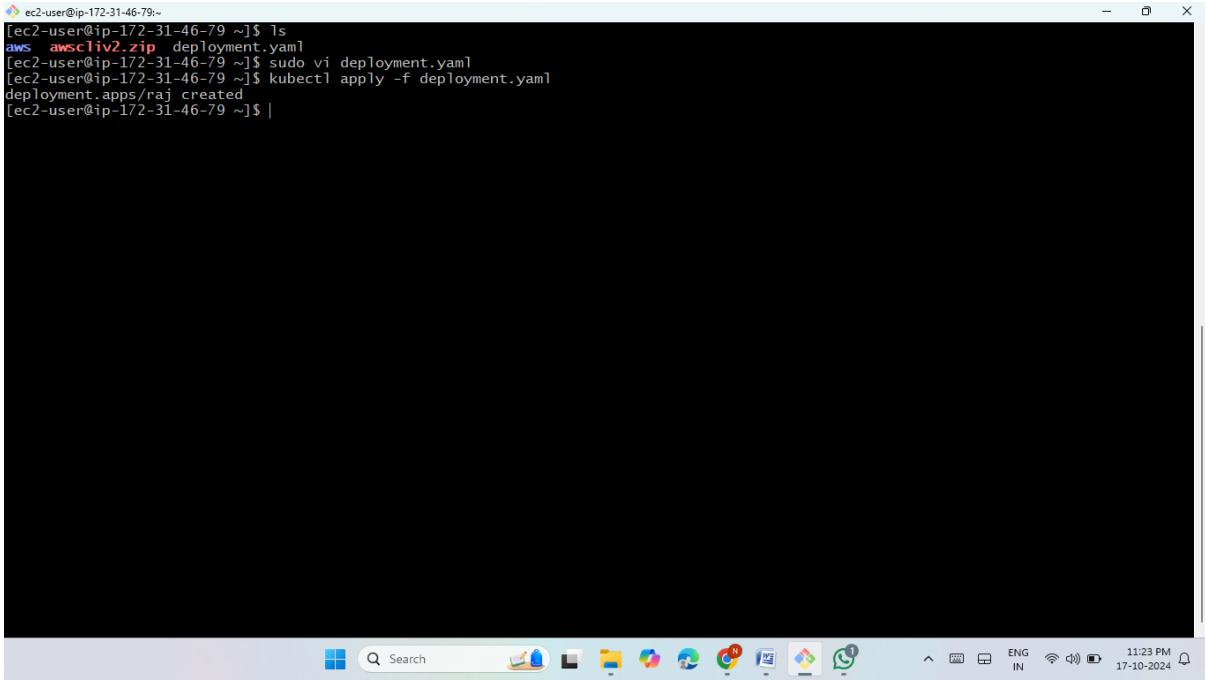
```

ec2-user@ip-172-31-46-79:~$ cat > deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: raj
  Labels:
    app: wordpress-app
    tier: frontend
spec:
  replicas: 4
  selector:
    matchLabels:
      tier: frontend
  template:
    metadata:
      labels:
        tier: frontend
    spec:
      containers:
        - name: ram
          image: yashwanthnajana/yash
          ports:
            - containerPort: 80
~
```

"deployment.yaml" 22L, 370B

- Now run the following deployment.yaml file with the command.

kubectl apply -f deployment.yaml



```

ec2-user@ip-172-31-46-79:~$ ls
aws awscli2.zip deployment.yaml
[ec2-user@ip-172-31-46-79 ~]$ sudo vi deployment.yaml
[ec2-user@ip-172-31-46-79 ~]$ kubectl apply -f deployment.yaml
deployment.apps/raj created
[ec2-user@ip-172-31-46-79 ~]$ |
```

- Now create a service.yaml file for run the dockerhub image by using **declarative manifest method**.

- Now run the following service.yaml file with the command.

kubectl apply -f service.yaml

kubectl get all

```
ec2-user@ip-172-31-46-79:~$ kubectl get all
NAME                         READY   STATUS    RESTARTS   AGE
pod/raj-7d66c997d6-brv6     1/1    Running   0          35s
pod/raj-7d66c997d6-hmfr7   0/1    ContainerCreating   0          35s
pod/raj-7d66c997d6-lnpft   0/1    ContainerCreating   0          35s
pod/raj-7d66c997d6-tztsj   1/1    Running   0          35s

NAME           TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes  ClusterIP  100.64.0.1   <none>        443/TCP  3m27s

NAME           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/raj  2/4    4           2           35s

NAME           DESIRED   CURRENT   READY   AGE
replicaset.apps/raj-7d66c997d6  4        4         4         35s
[ec2-user@ip-172-31-46-79 ~]$ sudo vi service.yaml
[ec2-user@ip-172-31-46-79 ~]$ kubectl apply -f service.yaml
service/ram created
[ec2-user@ip-172-31-46-79 ~]$ kubectl get all
NAME                         READY   STATUS    RESTARTS   AGE
pod/raj-7d66c997d6-brv6     1/1    Running   0          2m30s
pod/raj-7d66c997d6-hmfr7   1/1    Running   0          2m30s
pod/raj-7d66c997d6-lnpft   1/1    Running   0          2m30s
pod/raj-7d66c997d6-tztsj   1/1    Running   0          2m30s

NAME           TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)
service/kubernetes  ClusterIP  100.64.0.1   <none>        443/TCP
5m22s
service/ram     LoadBalancer  100.71.222.91  aaafc1e0c0ff4e89b5f7068f964121e-1650082884.us-east-1.elb.amazonaws.com  80:32001/TCP
8s

NAME           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/raj  4/4    4           4           2m30s

NAME           DESIRED   CURRENT   READY   AGE
replicaset.apps/raj-7d66c997d6  4        4         4         2m30s
[ec2-user@ip-172-31-46-79 ~]$ |
```

- Now with help of LoadBalancer DNS link the wordpress was hosted successfully.

