Deploying Wordpress App Over AWS Cloud

18/10/2023

Prerequisites:

- AWS cloud platform
- Ec2 instance (Uuntu)
- Networking (VPC, SG)
- Load Balancer
- Terraform
- MySQL database
- Apache web server
- Wordpress app

Creation of VPC and Ec2 instance using Terraform

- Install Terraform on your local machine
- Install AWS CLI to local machine
- Using this connect your local machine to your AWS account using command **aws configure**

```
nag@Practice:/media/nag/Other/k8s/aws-project$ aws configure
AWS Access Key ID [****************PLU6]:
AWS Secret Access Key [****************ZVMg]:
Default region name [us-east-1]:
Default output format [default]:
```

- Create a file called Provider.tf, vpc.tf, key.tf, ec2.tf, sg.tf. To create vpc networking in that we used 2 private subnets and 2 public subnets which is in 2 availability zones, IGW gateway, NAT gateway, router and 2 ec2 machines ion which one is bastion server which is in public subnet and other one is app server which is in private subnet
- Initialize terraform in this folder using command **terraform init**

```
nag@Practice:/media/nag/Other/k8s/aws-project$ terraform init

Initializing the backend...
Initializing modules...

Initializing provider plugins...
- Reusing previous version of hashicorp/tls from the dependency lock file
- Reusing previous version of hashicorp/local from the dependency lock file
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/tls v4.0.4
- Using previously-installed hashicorp/local v2.4.0
- Using previously-installed hashicorp/aws v5.19.0

Terraform has been successfully initialized!

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

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
nag@Practice:/media/nag/Other/k8s/aws-project$
```

- Use **terraform plan** command to view the plan what we are creating in the aws cloud platform

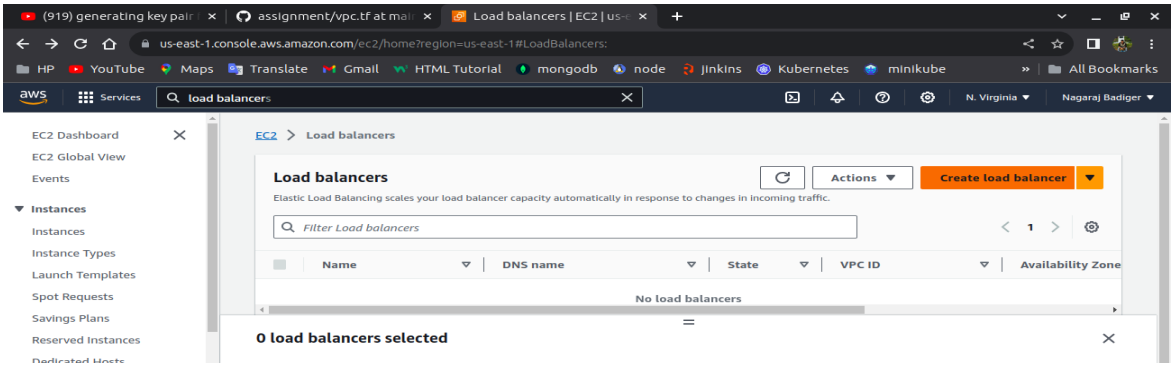
```
nag@Practice:/media/nag/Other/k8s/aws-project$ terraform plan
module.bastion_instance.data.aws_partition.current: Reading...
module.app_instance.data.aws_partition.current: Reading...
module.app_instance.data.aws_partition.current: Read complete after 0s [id=aws]
module.bastion_instance.data.aws_partition.current: Read complete after 0s [id=aws]

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

Terraform will perform the following actions:

  # aws_key_pair.assignment_key will be created
  + resource "aws_key_pair" "assignment_key" {
      + arn             = (known after apply)
      + fingerprint     = (known after apply)
      + id              = (known after apply)
      + key_name        = "assignment_key"
      + key_name_prefix = (known after apply)
      + key_pair_id     = (known after apply)
      + key_type        = (known after apply)
      + public_key      = (known after apply)
      + tags_all        = (known after apply)
    }
```

- Use **terraform apply** command to apply those plans to execute in aws cloud platform

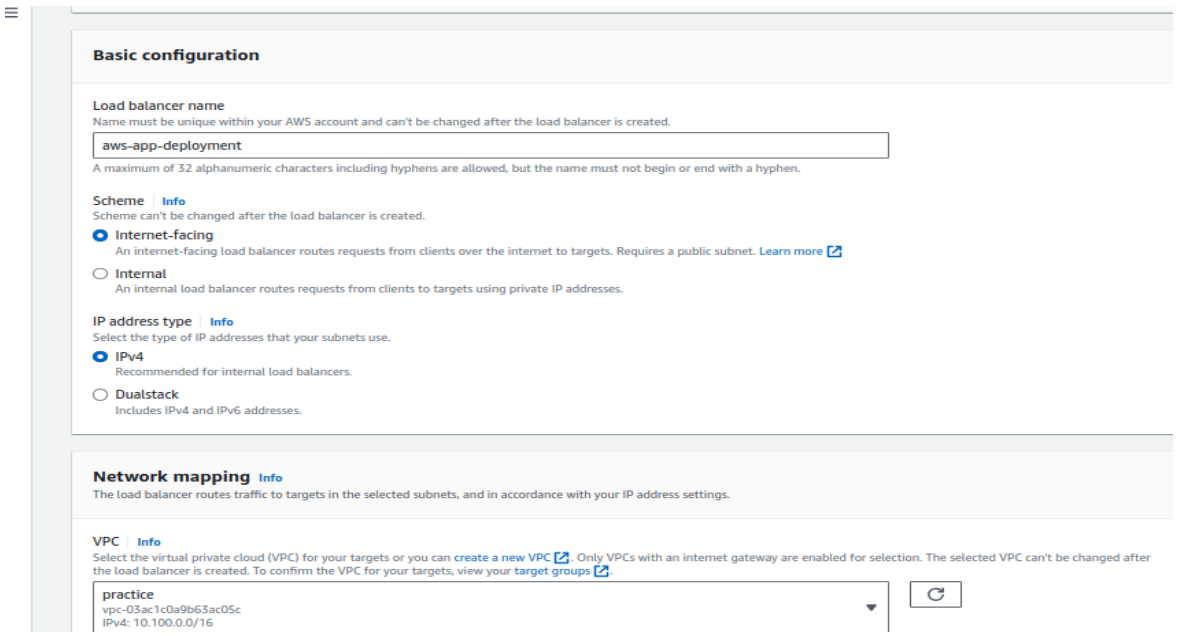
```
nag@Practice:/media/nag/Other/k8s/aws-project$ terraform apply
module.app_instance.data.aws_partition.current: Reading...
module.bastion_instance.data.aws_partition.current: Reading...
module.app_instance.data.aws_partition.current: Read complete after 0s [id=aws]
module.bastion_instance.data.aws_partition.current: Read complete after 0s [id=aws]

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

Terraform will perform the following actions:

  # aws_key_pair.assignment_key will be created
  + resource "aws_key_pair" "assignment_key" {
      + arn             = (known after apply)
      + fingerprint     = (known after apply)
      + id              = (known after apply)
      + key_name        = "assignment_key"
      + key_name_prefix = (known after apply)
      + key_pair_id     = (known after apply)
      + key_type        = (known after apply)
      + public_key      = (known after apply)
      + tags_all        = (known after apply)
    }

  # aws_security_group.private-sg will be created
```

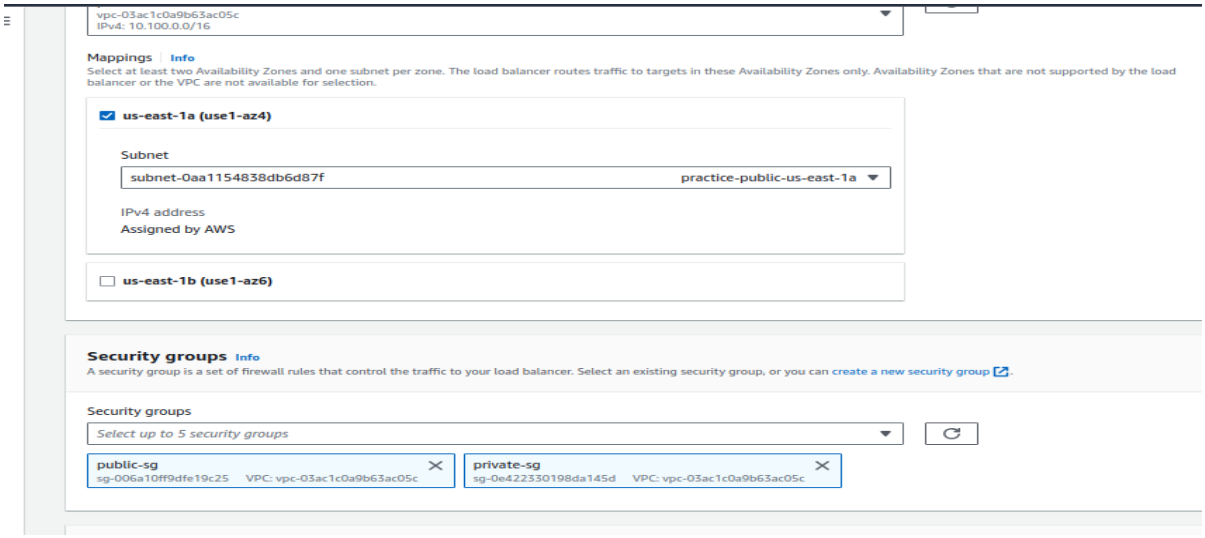
Creation of Load Balancer

- Create the Load Balancer in AWS cloud platform which is in the same vpc in which we are deploying the WordPress app

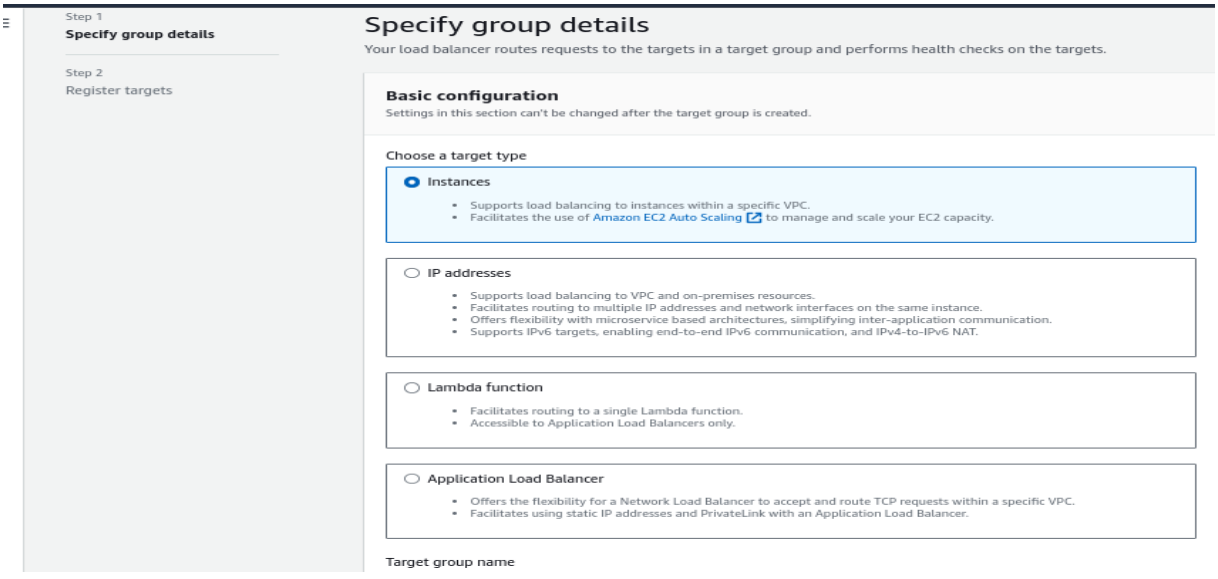


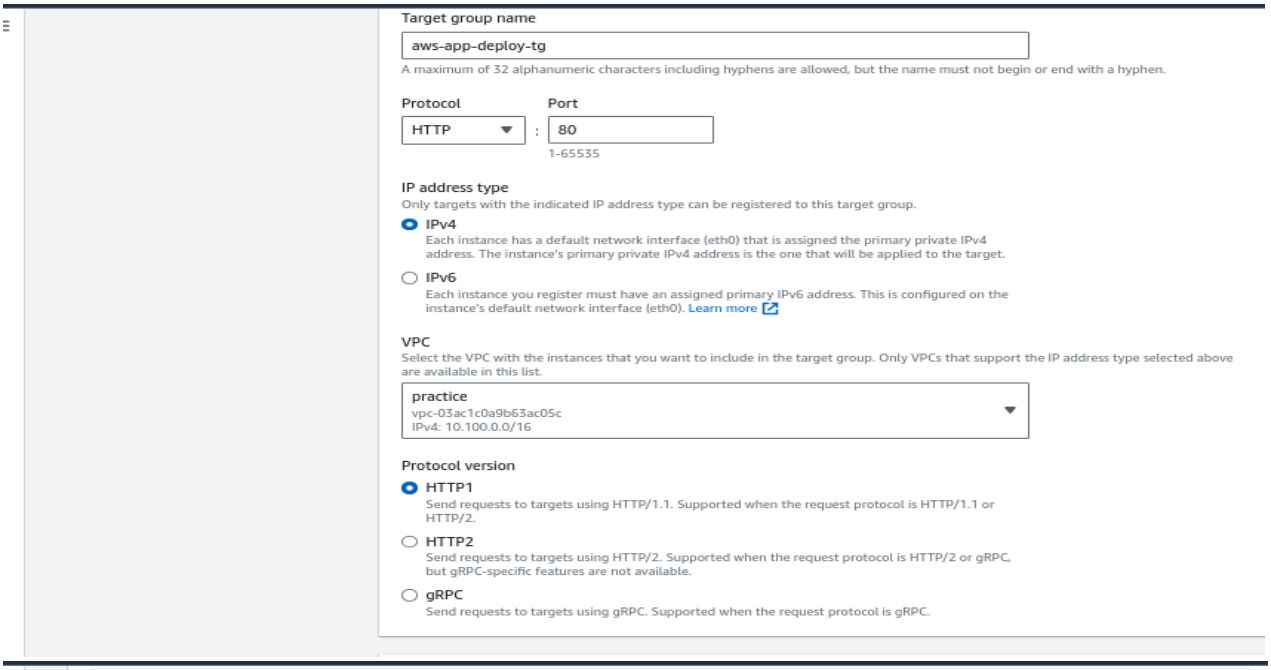
- Select the create load balancer (ALB) named it as aws-app-deployment which is in the vpc where we want to deploy the app
- Select scheme as internet facing
- Select the public subnets is in which Load Balancer should be
- Security group in which port 80 should be open for all, and I selected default security group.

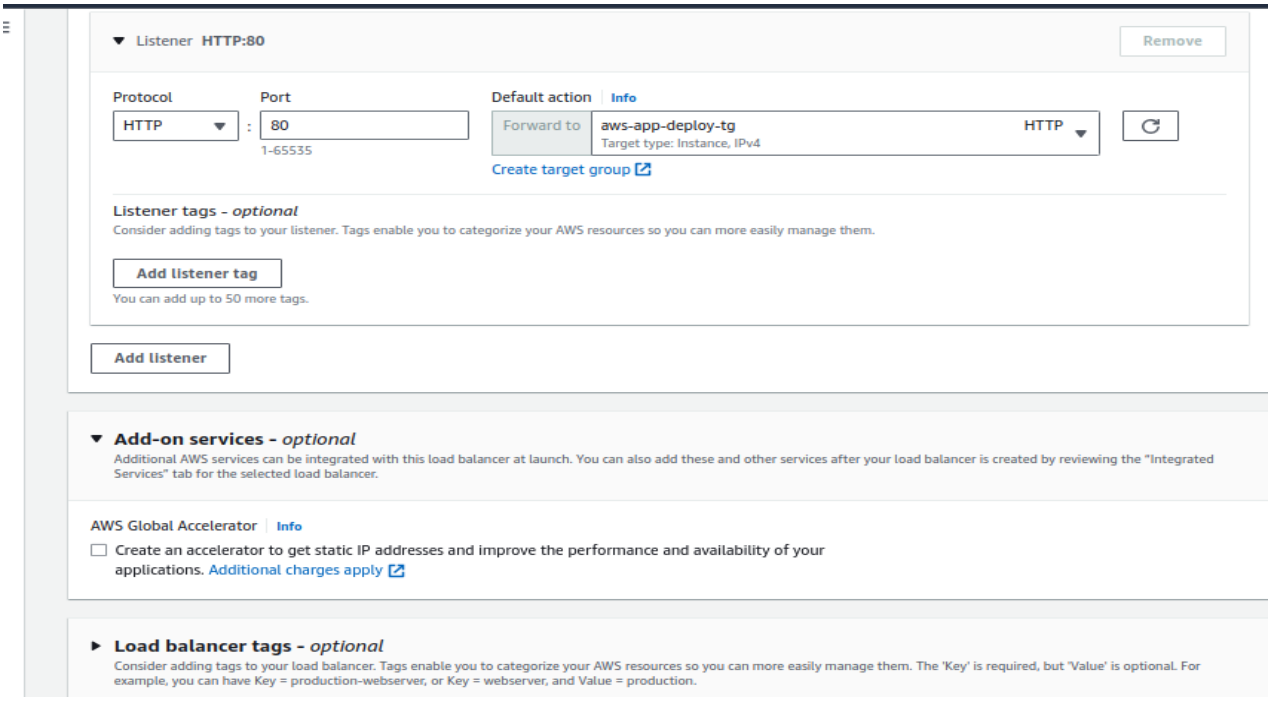




- Create a target group which is instance typed, name it as aws-app-deploy-tg.
- In this we selected the HTTP protocol on port 80
- Add the listener in which all signal in the app server instance goes to the load balancer
- Click on the create the load balancer button.




Target group name
aws-app-deploy-tg
A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.
Protocol
HTTP
Port
80
1-65535
IP address type
Only targets with the indicated IP address type can be registered to this target group.
IPv4
Each instance has a default network interface (eth0) that is assigned the primary private IPv4 address. The instance's primary private IPv4 address is the one that will be applied to the target.
IPv6
Each instance you register must have an assigned primary IPv6 address. This is configured on the instance's default network interface (eth0). Learn more
VPC
Select the VPC with the instances that you want to include in the target group. Only VPCs that support the IP address type selected above are available in this list.
practice
vpc-03ac1c0a9b63ac05c
IPv4: 10.100.0.0/16
Protocol version
HTTP1
Send requests to targets using HTTP/1.1. Supported when the request protocol is HTTP/1.1 or HTTP/2.
HTTP2
Send requests to targets using HTTP/2. Supported when the request protocol is HTTP/2 or gRPC, but gRPC-specific features are not available.
gRPC
Send requests to targets using gRPC. Supported when the request protocol is gRPC.


Listener HTTP:80
Remove
Protocol
HTTP
Port
80
1-65535
Default action
Info
Forward to
aws-app-deploy-tg
Target type: Instance, IPv4
HTTP
Create target group
Listener tags - optional
Consider adding tags to your listener. Tags enable you to categorize your AWS resources so you can more easily manage them.
Add listener tag
You can add up to 50 more tags.
Add listener
Add-on services - optional
Additional AWS services can be integrated with this load balancer at launch. You can also add these and other services after your load balancer is created by reviewing the "Integrated Services" tab for the selected load balancer.
AWS Global Accelerator
Info
Create an accelerator to get static IP addresses and improve the performance and availability of your applications. Additional charges apply
Load balancer tags - optional
Consider adding tags to your load balancer. Tags enable you to categorize your AWS resources so you can more easily manage them. The 'Key' is required, but 'Value' is optional. For example, you can have Key = production-webserver, or Key = webserver, and Value = production.

Deploying the WordPress app

- **Ssh to Bastion server to connect to the app server**

```
nag@Practice:/media/nag/Other/k8s/aws-project$ cd
nag@Practice:~$ ssh -i "assignment_key.pem" ubuntu@ec2-54-226-31-183.compute-1.amazonaws.com
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.19.0-1025-aws x86_64)

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

  System information as of Tue Oct 17 11:59:10 UTC 2023

  System load:  0.0               Processes:             96
  Usage of /:   20.6% of 7.57GB   Users logged in:       0
  Memory usage: 25%               IPv4 address for eth0: 10.100.3.54
  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.
```

- After this we should add the private key inside this server and then we shh to the actual app server

```
ubuntu@ip-10-100-3-54:~$ vim assignment_key.pem
ubuntu@ip-10-100-3-54:~$ ssh -i "assignment_key.pem" ubuntu@10.100.1.229
The authenticity of host '10.100.1.229 (10.100.1.229)' can't be established.
ED25519 key fingerprint is SHA256:JfL64C8s+qM4K6eE7XhlNvhCaaOhEGX+MRfdoA9i5jQ.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.100.1.229' (ED25519) to the list of known hosts.
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@         WARNING: UNPROTECTED PRIVATE KEY FILE!          @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Permissions 0664 for 'assignment_key.pem' are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
Load key "assignment_key.pem": bad permissions
ubuntu@10.100.1.229: Permission denied (publickey).
ubuntu@ip-10-100-3-54:~$
```

- Install all the dependencies which is required for **WordPress app** by using following command

```
Sudo apt update
Sudo apt install apache2 \
        ghostscript \
        libapache2-mod-php \
        mysql-server \
        php \
        php-bcmath \
        php-curl \
        php-imagick \
```

php-intl \
php-json \
php-mbstring \
php-mysql \
php-xml \
php-zip

```
ubuntu@ip-10-100-1-229:~$ sudo apt install apache2 \
                          ghostscript \
                          libapache2-mod-php \
                          mysql-server \
                          php \
                          php-bcmath \
                          php-curl \
                          php-imagick \
                          php-intl \
                          php-json \
                          php-mbstring \
                          php-mysql \
                          php-xml \
                          php-zip
```

Installing the WordPress app by using the following command

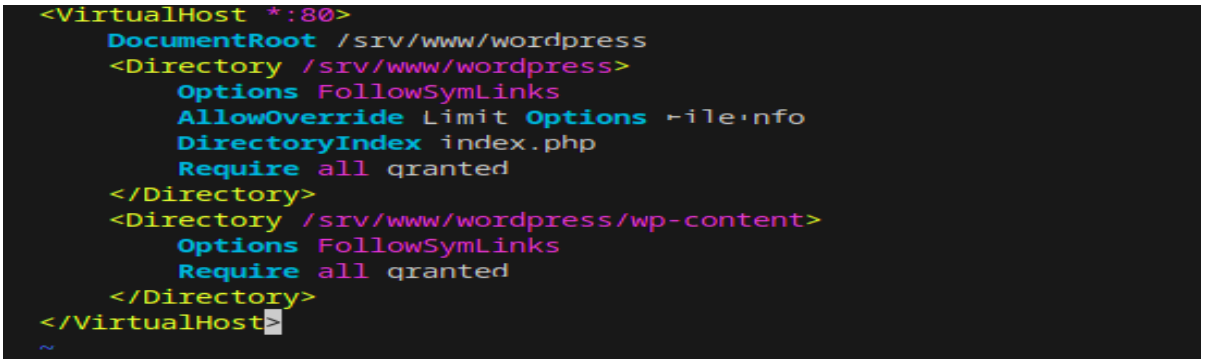
- Make the parent directory to import the WordPress dependencies
  sudo mkdir –p /srv/www
- Change the owner of that directory to www-data
  sudo chown www-data: /srv/www
- Use the curl command to get the WordPress app dependencies and untar this to src/www directory
  curl https://wordpress.org/latest.tar.gz | sudo -u www-data tar zx -C /srv/www

```
ubuntu@ip-10-100-1-229:~$ sudo mkdir -p /srv/www
ubuntu@ip-10-100-1-229:~$ sudo chown www-data: /srv/www
ubuntu@ip-10-100-1-229:~$ curl https://wordpress.org/latest.tar.gz | sudo -u www-data tar zx -C /srv/www
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 22.3M  100 22.3M    0     0  18.5M      0  0:00:01  0:00:01 --:--:-- 18.5M
ubuntu@ip-10-100-1-229:~$
```

- Configure the Apache web server by creating the file wordpress.conf by using the command
  Sudo vim /etc/apache2/sites-available/wordpress.conf

- Insert these following lines

```
<VirtualHost *:80>
    DocumentRoot /srv/www/wordpress
    <Directory /srv/www/wordpress>
        Options FollowSymLinks
        AllowOverride Limit Options FileInfo
        DirectoryIndex index.php
        Require all granted
    </Directory>
    <Directory /srv/www/wordpress/wp-content>
        Options FollowSymLinks
        Require all granted
    </Directory>
</VirtualHost>
```



- Enable the wordpress site with the following command

  sudo a2ensite wordpress

- Enable URL rewriting with command

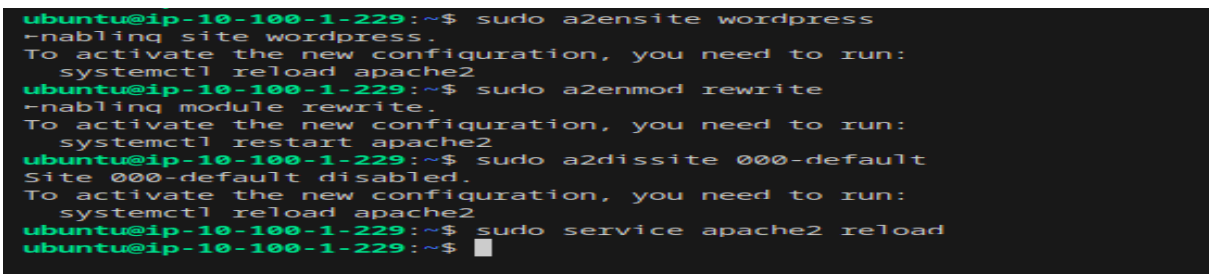
  sudo a22enmod rewrite

- Disable the default “It Works” site with

  sudo a2dissite 000-default

- Reload the apache server to change these settings

  sudo service apache2 reload



Configure database

To configure the wordpress, we have to create database by providing username, password, by using the foloowing command

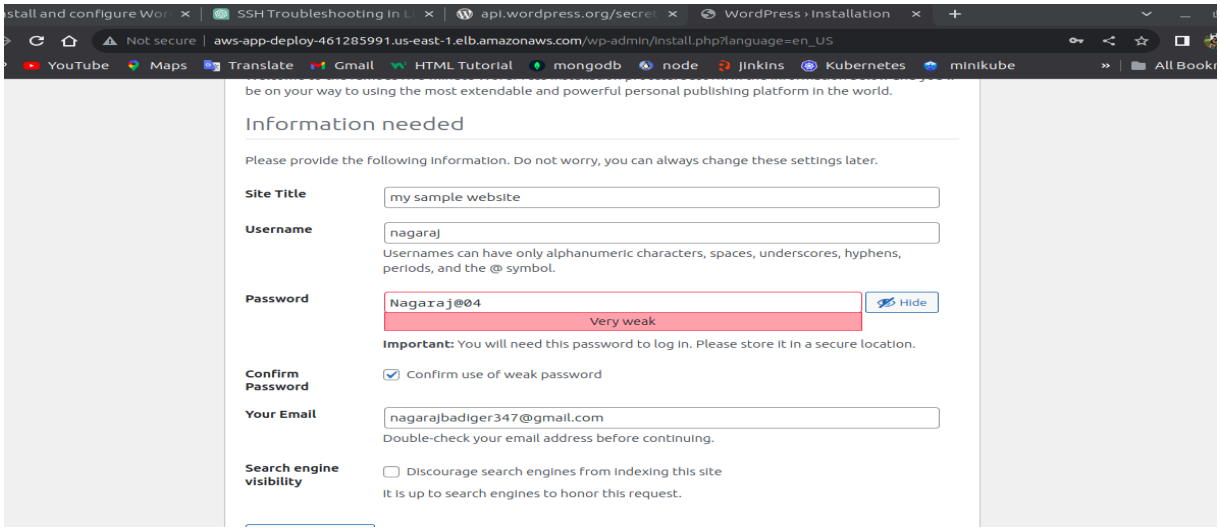
- sudo mysql -u root
- mysql> ALTER USER ‘root’@localhost IDENTIFIED WITH mysql_native_password BY ‘<PASSWORD>’;
- mysql > CREATE USER ‘<USERNAME>’@localhost IDENTIFIED BY ‘<PASSWORD>’;
- mysql> CREATE DATABASE <DATABASENAME>;
- mysql> GRANT ALL PRIVILEDGES ON <DATABASENAME>. * TO ‘<USERNAME>’@localhost
- mysql> FLUSH PRIVILEGES;

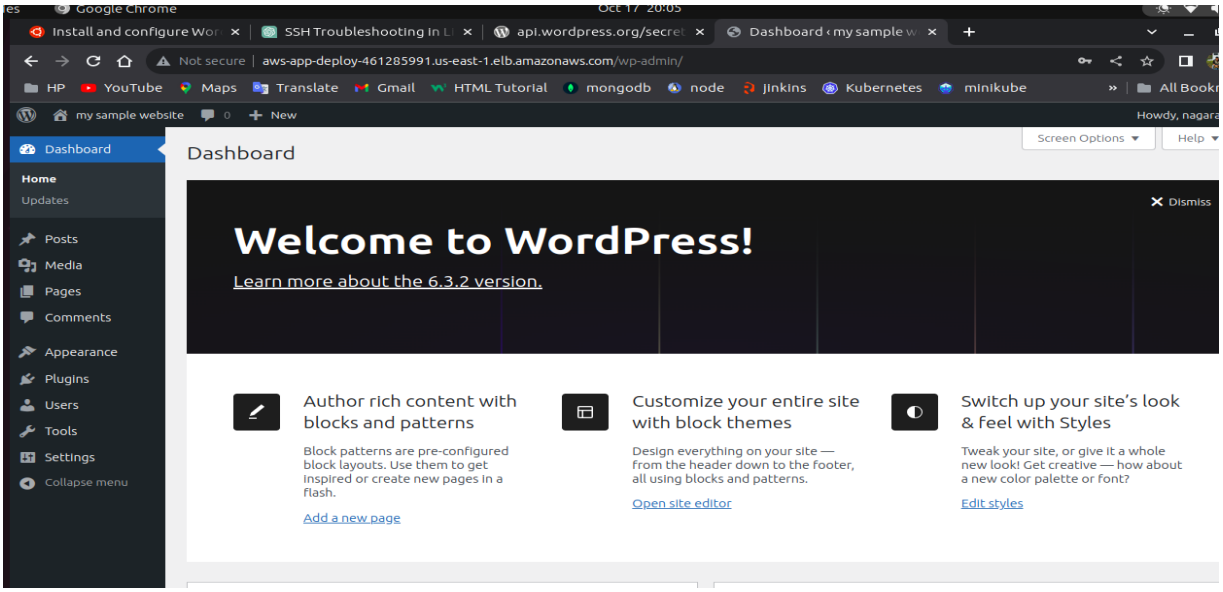
Start the mysql service using command “sudo service mysql start”

```
[4]+  Stopped                 sudo mysql -u root
ubuntu@ip-10-100-1-229:~$ sudo mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 8.0.34-0ubuntu0.22.04.1 (Ubuntu)

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

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

mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP,ALTER
    -> ON wordpress.*
    -> TO wordpress@nagaraj;
Query OK, 0 rows affected (0.00 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.01 sec)

mysql> quit
Bye
ubuntu@ip-10-100-1-229:~$ sudo -u www-data cp /srv/www/wordpress/wp-config-sample.php /srv/www/wordpress/wp-confi
g.php
```

Now click on the load balancer URL fill the necessary which was provided during installation like username, password, database name etc... after that wp-config.php file gets created in the directory /srv/www/wordpress.

After this you will get the necessary login page of wordpress app...





********** SUCCESSFULLY DEPLOYED TO AWS CLOUD PLATFORM**********