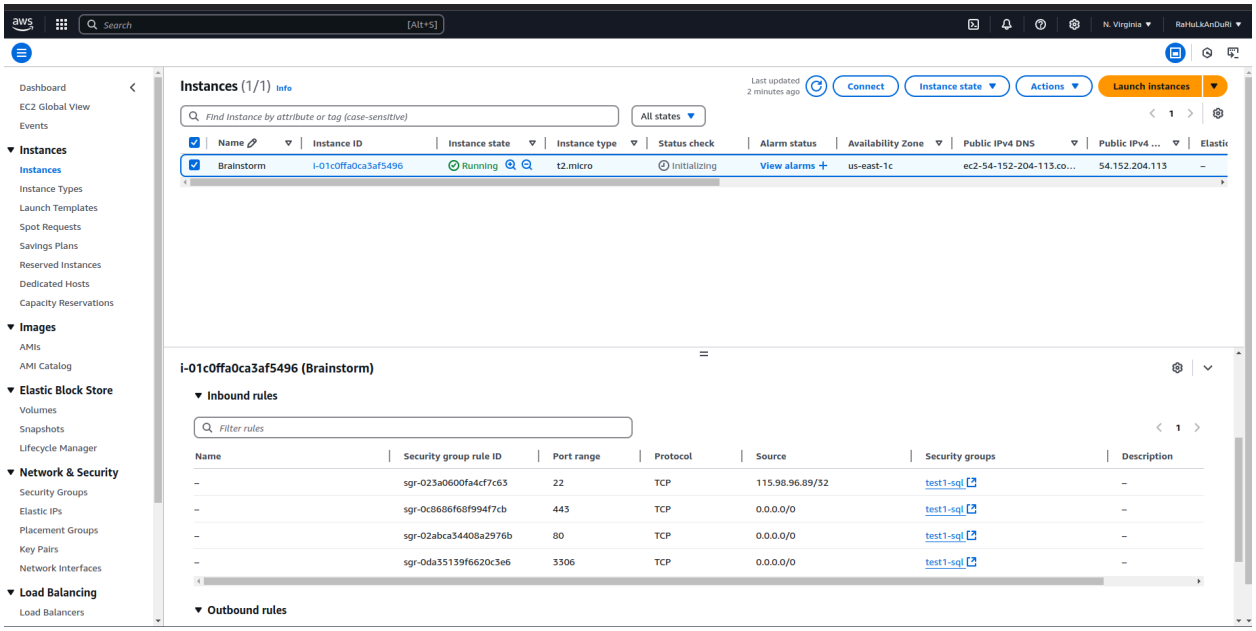
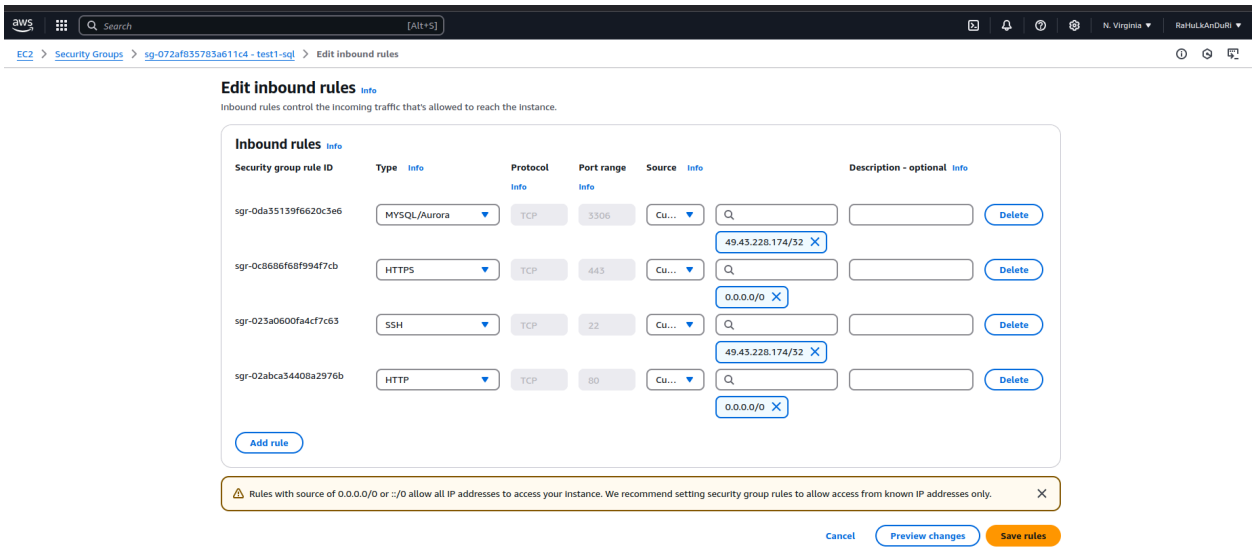
Brainstorm Technical Assignment

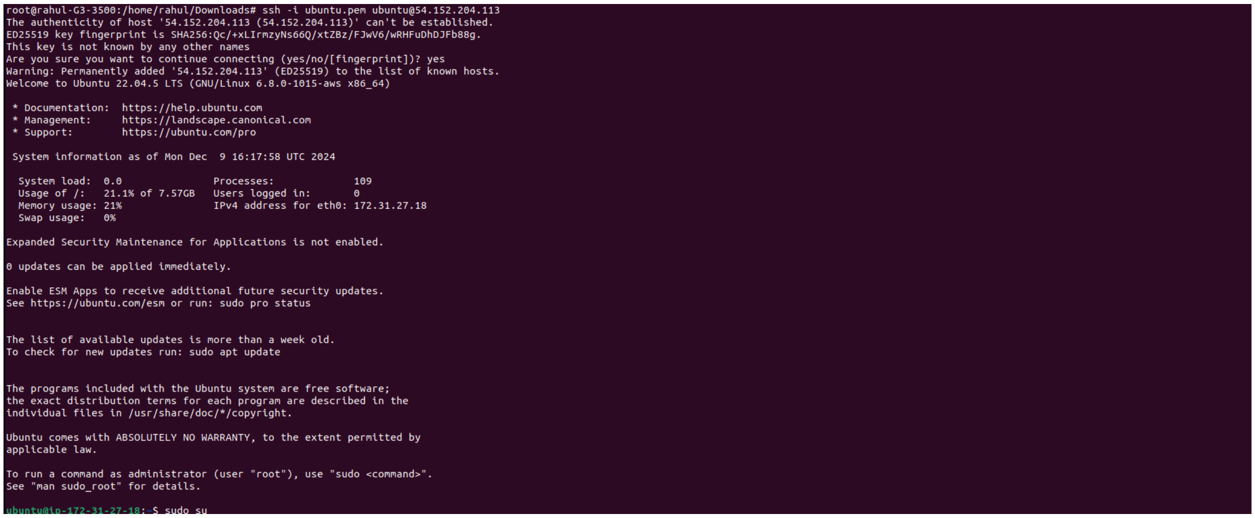
First create an EC2 instance with the following Security Groups we will be modifying the inbound rules once we are done with the setup.





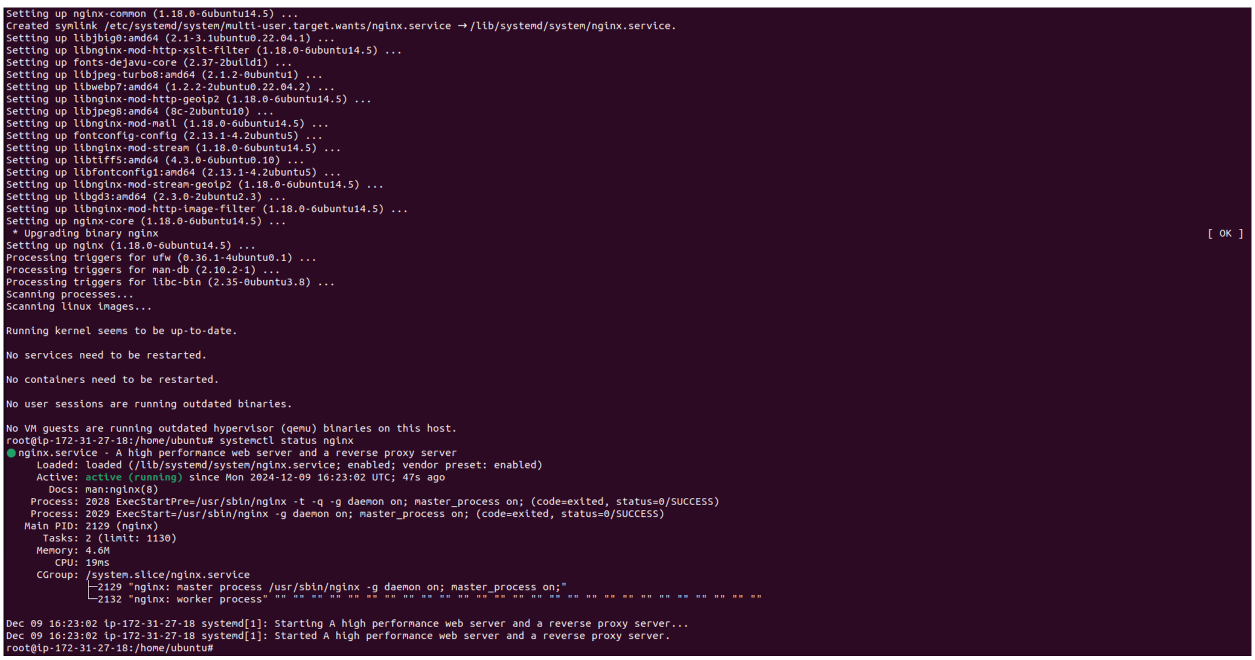
Brainstorm Technical Assignment

Ssh into the EC2 instance

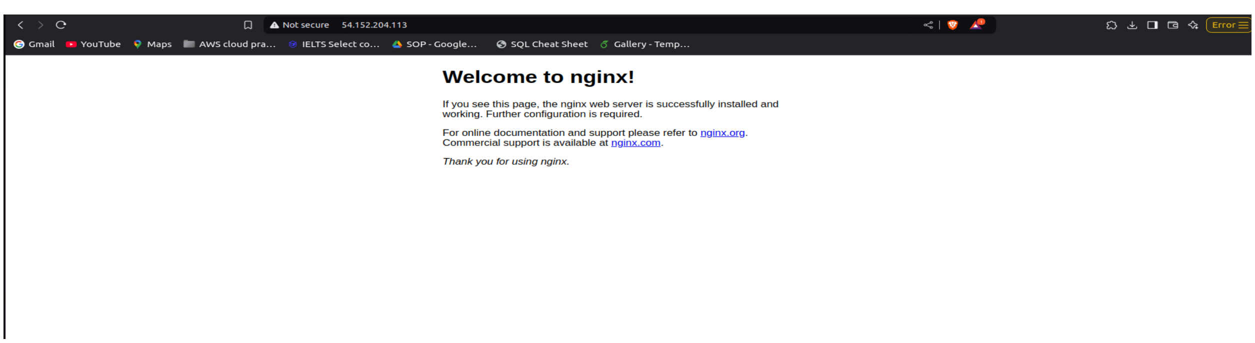


```
root@rahul-G3-3500:/home/rahul/Downloads# ssh -i ubuntu.pem ubuntu@54.152.204.113
The authenticity of host '54.152.204.113 (54.152.204.113)' can't be established.
ED25519 key fingerprint is SHA256:Qc/+xLIrmzyNs66Q/xtZBz/FJwV6/wRHFuDhDJFb88g.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '54.152.204.113' (ED25519) to the list of known hosts.
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 Dec  9 16:17:58 UTC 2024

  System load:  0.0               Processes:             109
  Usage of /:   21.1% of 7.57GB   Users logged in:       0
  Memory usage: 21%               IPv4 address for eth0: 172.31.27.18
  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-172-31-27-18:~$ sudo su
```

Then install nginx on our Ec2 instance and make sure is running

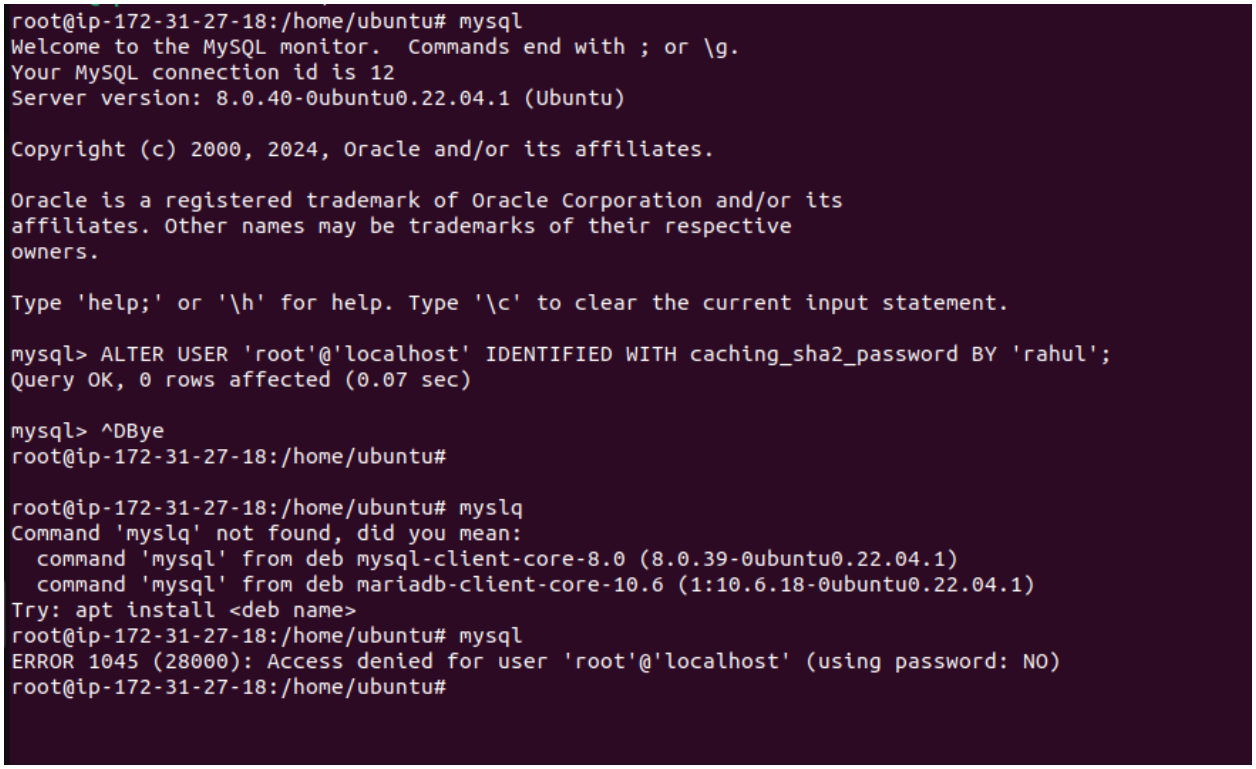


```
Setting up nginx-common (1.18.0-6ubuntu14.5) ...
Created symlink /etc/systemd/system/multi-user.target.wants/nginx.service → /lib/systemd/system/nginx.service.
Setting up libjbig0:amd64 (2.1-3.1ubuntu0.22.04.1) ...
Setting up libnginx-mod-http-xslt-filter (1.18.0-6ubuntu14.5) ...
Setting up fonts-dejavu-core (2.37-2build1) ...
Setting up libjpeg-turbo8:amd64 (2.1.2-0ubuntu1) ...
Setting up libwebp7:amd64 (1.2.2-2ubuntu0.22.04.2) ...
Setting up libnginx-mod-http-geoip2 (1.18.0-6ubuntu14.5) ...
Setting up libjpeg8:amd64 (8c-2ubuntu10) ...
Setting up libnginx-mod-mail (1.18.0-6ubuntu14.5) ...
Setting up fontconfig-config (2.13.1-4.2ubuntu5) ...
Setting up libnginx-mod-stream (1.18.0-6ubuntu14.5) ...
Setting up libtiff5:amd64 (4.3.0-6ubuntu0.10) ...
Setting up libfontconfig1:amd64 (2.13.1-4.2ubuntu5) ...
Setting up libnginx-mod-stream-geoip2 (1.18.0-6ubuntu14.5) ...
Setting up libgd3:amd64 (2.3.0-2ubuntu2.3) ...
Setting up libnginx-mod-http-image-filter (1.18.0-6ubuntu14.5) ...
Setting up nginx-core (1.18.0-6ubuntu14.5) ...
 * Upgrading binary nginx                                                [ OK ]
Setting up nginx (1.18.0-6ubuntu14.5) ...
Processing triggers for ufw (0.36.1-4ubuntu0.1) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.8) ...
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.
root@ip-172-31-27-18:/home/ubuntu# systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
     Active: active (running) since Mon 2024-12-09 16:23:02 UTC; 47s ago
       Docs: man:nginx(8)
    Process: 2028 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
    Process: 2029 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
   Main PID: 2129 (nginx)
      Tasks: 2 (limit: 1130)
     Memory: 4.6M
        CPU: 19ms
     CGroup: /system.slice/nginx.service
             ├─2129 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
             └─2132 "nginx: worker process" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" ""

Dec 09 16:23:02 ip-172-31-27-18 systemd[1]: Starting A high performance web server and a reverse proxy server...
Dec 09 16:23:02 ip-172-31-27-18 systemd[1]: Started A high performance web server and a reverse proxy server.
root@ip-172-31-27-18:/home/ubuntu#
```



Brainstorm Technical Assignment

Next we will install mysql and assign one password to it as shown

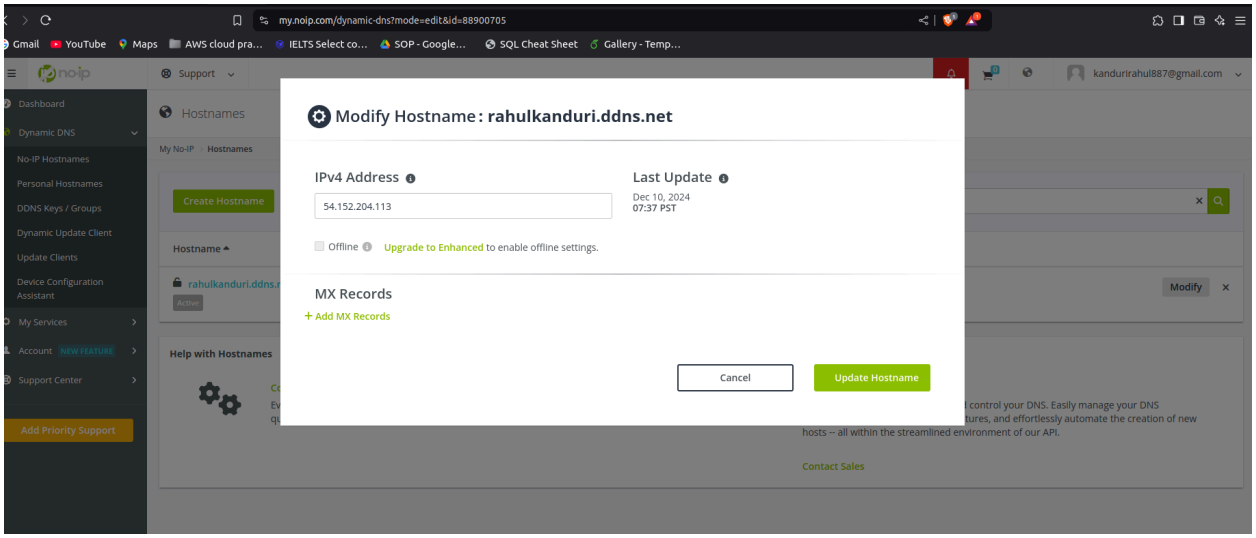


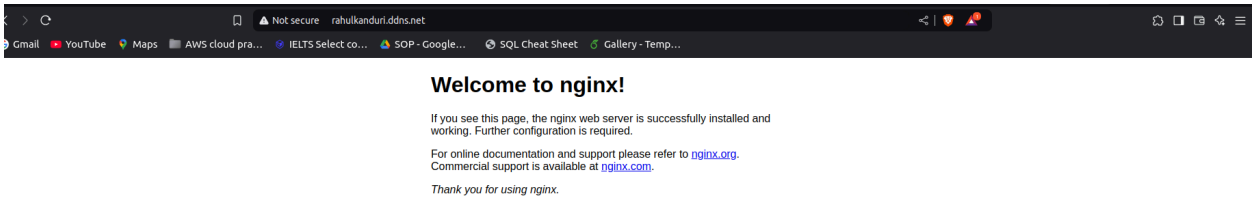
Next install PHP remember each WordPress plugin has its own set of requirements. Some may require additional PHP extension packages to be installed.

sudo apt install php-curl php-gd php-intl php-mbstring php-soap php-xml php-xmlrpc php-zip

Next we need to create a domain for our application, make sure that you select ‘A’ name in the type since we are directly routing traffic from here and in IPV4 address give your public IP.
If everything goes right when you copy and paste the domain name we need to see nginx welcome page that we got when we opened it with instance public IP as shown

Brainstorm Technical Assignment





Now we will need to change the nginx conf as follows, and lets understand the code

```
root@ip-172-31-27-18:/home/ubuntu# cat /etc/nginx/sites-available/wordpress
server {
    listen 80;
    server_name rahulkanduri.ddns.net;

    root /var/www/wordpress;
    index index.php index.html index.htm;

    location / {
        try_files $uri $uri/ /index.php?$args;
    }

    location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        include fastcgi_params;
    }

    location ~ /\.ht {
        deny all;
    }
}
root@ip-172-31-27-18:/home/ubuntu# nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
root@ip-172-31-27-18:/home/ubuntu#
```

=======

Brainstorm Technical Assignment

```
server {
    listen 80;
    server_name rahulkanduri.ddns.net;

    root /var/www/wordpress;
    index index.php index.html index.htm;

    location / {
        try_files $uri $uri/ /index.php?$args;
    }

    location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        include fastcgi_params;
    }

    location ~ /\.ht {
        deny all;
    }
}
```

In server block we have given the details like nginx port which is 80 and in servername we have given our domain name so that nginx can send the traffic to localhost according the config we are defining
Next in root we are defining the working directory where our entire code and files will be there
Then in index we have given the files that it has to try by order inside the working directory
Next we are using location block for root which mean all request coming to nginx url
Try_files this is tricky part so nginx has some pretty URLS the lets make is simple with example:

Brainstorm Technical Assignment

$uri : if I do http://rahulkanduri.ddns.net/somepage nginx will search for somepage.html and serves it.
$uri/ this means if we do not have somepage.html it will search for somepage folder inside root folder of nginx and serves the .html file inside of it

If none of the got satisfied it will show index.php

Next we have another location block like “location ~ \.php$ ”

First thing nginx cannot serve .php files, let’s say in our root folder I have somepage.html then we are hitting this file then we can see contents in it but if it is somepage.php nginx cannot serve that
So we are including fastcgi-php.conf which have some common configs to handle .php code
Next we have fastcgi_pass which is U**nix socket** where PHP-FPM (FastCGI Process Manager) is running this will help to process php code

Now we need to install wordpress we can do it with following commands

```
 14  nano /etc/nginx/sites-available/rahulkanduri.ddns.net
 15  sudo wget https://wordpress.org/latest.tar.gz
 16  tar -xvzf latest.tar.gz
 17  mv wordpress /var/www/wordpress
 18  history
oot@ip-172-31-27-18:/home/ubuntu#
```

Next we need an database so that word press can store content and we will create an user as well

Brainstorm Technical Assignment

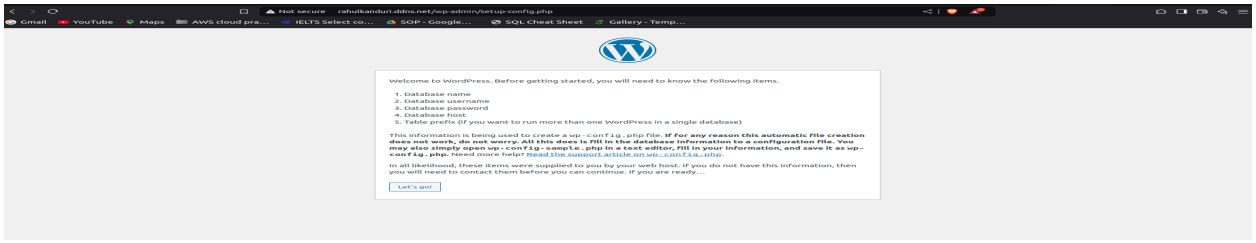
```
root@ip-172-31-27-18:/home/ubuntu# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 16
Server version: 8.0.40-0ubuntu0.22.04.1 (Ubuntu)

Copyright (c) 2000, 2024, 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> create database wordpress;
Query OK, 1 row affected (0.02 sec)

mysql> CREATE USER 'rahul'@'localhost' IDENTIFIED BY 'rahul';
Query OK, 0 rows affected (0.03 sec)

mysql>
mysql> GRANT ALL PRIVILEGES ON wordpress.* TO 'rahul'@'localhost';
Query OK, 0 rows affected (0.01 sec)

mysql> GRANT ALL PRIVILEGES ON *.* TO 'rahul'@'localhost';
Query OK, 0 rows affected (0.01 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.01 sec)

mysql>
```

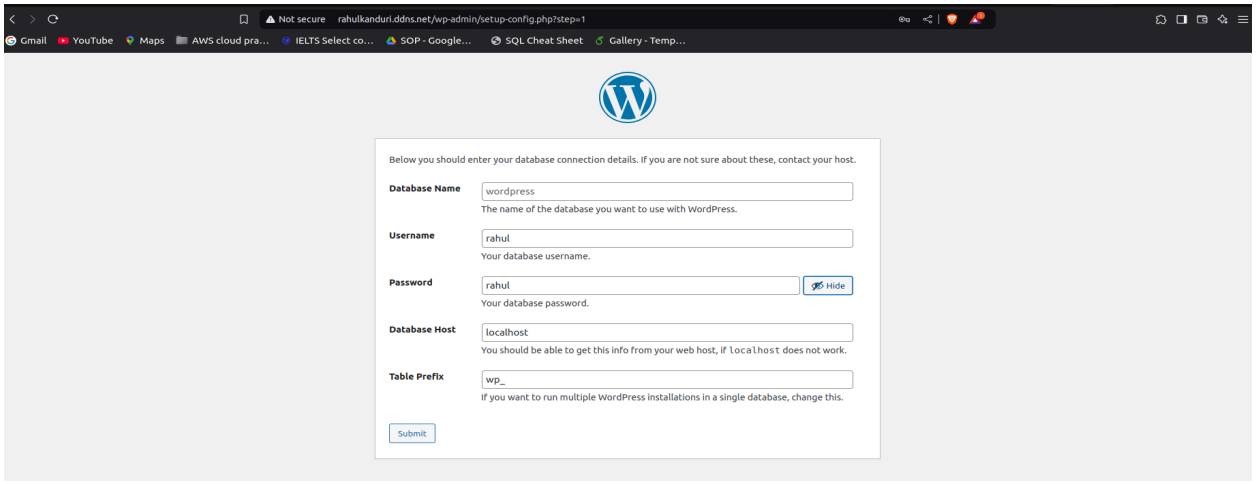
Update this values in config so that wordpress can authenticate the credentials

```
 *
 * * Database settings
 * * Secret keys
 * * Database table prefix
 * * ABSPATH
 *
 * @link https://developer.wordpress.org/advanced-administration/wordpress/wp-config/
 *
 * @package WordPress
 */

// ** Database settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define( 'DB_NAME', 'wordpress' );

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

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

/** Database hostname */
define( 'DB_HOST', 'localhost' );

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

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

/**#@+
 * Authentication unique keys and salts.
```

Next please do http://rahulkanduri.ddns.net/wordpress
If you get 502 bad gateway make sure that php and php-fpm version is same remember php-fpm will process the code for you

Brainstorm Technical Assignment





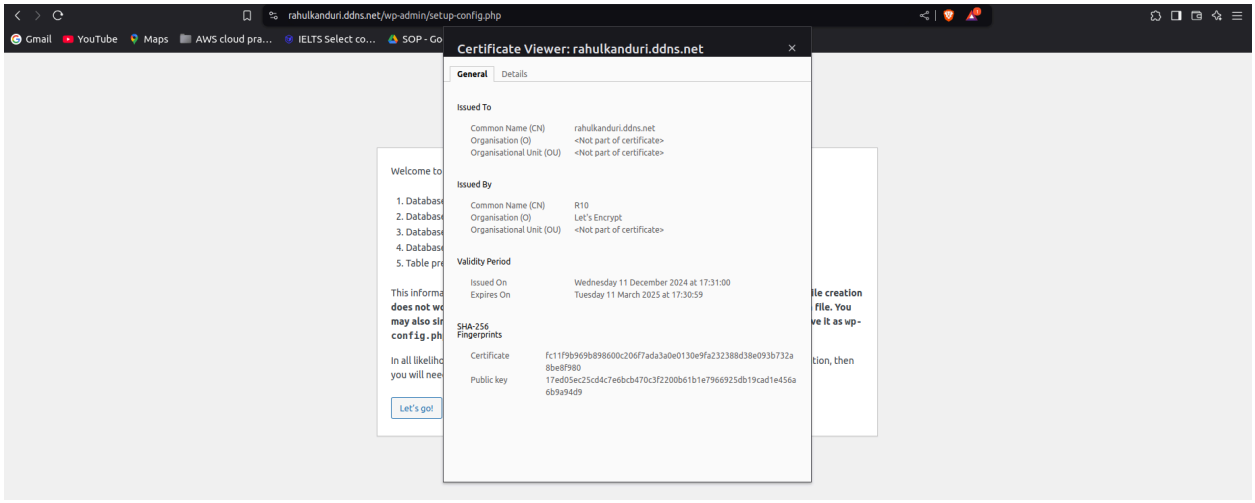
SSL:

First install certbot for getting the ssl certificates
sudo apt install certbot python3-certbot-nginx -y
Now we need to get ssl certificates right so we run the following command
sudo certbot --nginx -d rahulkanduri.ddns.net
Then it will ask some questions like email address etc

```
No VM guests are running outdated hypervisor (qemu) binaries on this host.
root@ip-172-31-27-18:/home/ubuntu# sudo certbot --nginx -d rahulkanduri.ddns.net
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Enter email address (used for urgent renewal and security notices)
 (Enter 'c' to cancel): kanduriraul887@gmail.com

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Please read the Terms of Service at
https://letsencrypt.org/documents/LE-SA-v1.4-April-3-2024.pdf. You must agree in
order to register with the ACME server. Do you agree?
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(Y)es/(N)o: y

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Would you be willing, once your first certificate is successfully issued, to
share your email address with the Electronic Frontier Foundation, a founding
partner of the Let's Encrypt project and the non-profit organization that
develops Certbot? We'd like to send you email about our work encrypting the web,
EFF news, campaigns, and ways to support digital freedom.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(Y)es/(N)o: n
Account registered.
Requesting a certificate for rahulkanduri.ddns.net

Successfully received certificate.
Certificate is saved at: /etc/letsencrypt/live/rahulkanduri.ddns.net/fullchain.pem
Key is saved at:         /etc/letsencrypt/live/rahulkanduri.ddns.net/privkey.pem
This certificate expires on 2025-03-11.
These files will be updated when the certificate renews.
Certbot has set up a scheduled task to automatically renew this certificate in the background.

Deploying certificate
Successfully deployed certificate for rahulkanduri.ddns.net to /etc/nginx/sites-enabled/wordpre
Congratulations! You have successfully enabled HTTPS on https://rahulkanduri.ddns.net

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
If you like Certbot, please consider supporting our work by:
 * Donating to ISRG / Let's Encrypt:   https://letsencrypt.org/donate
 * Donating to EFF:                    https://eff.org/donate-le
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
root@ip-172-31-27-18:/home/ubuntu#
```

Now you open your URL with https, if you get any error remember we have to allow traffic to instance with https 443 as well

Brainstorm Technical Assignment



But we haven't changed anything in the server block to listen to 443 right? Then how can we open the URL?
The answer is certbot is nginx extension so it will change the nginx.conf (in our case /etc/nginx/sites-available/wordpress)

```
root@ip-172-31-27-18:/home/ubuntu#
root@ip-172-31-27-18:/home/ubuntu# cat /etc/nginx/sites-available/wordpress
server {
    server_name rahulkanduri.ddns.net;

    root /var/www/wordpress;
    index index.php index.html index.htm;

    location / {
        try_files $uri $uri/ /index.php?$args;
    }

    location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/var/run/php/php8.1-fpm.sock;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        include fastcgi_params;
    }

    location ~ /\.ht {
        deny all;
    }

    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/rahulkanduri.ddns.net/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/rahulkanduri.ddns.net/privkey.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

}
server {
    if ($host = rahulkanduri.ddns.net) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


    listen 80;
    server_name rahulkanduri.ddns.net;
    return 404; # managed by Certbot
```

Optimizing Nginx server configuration:

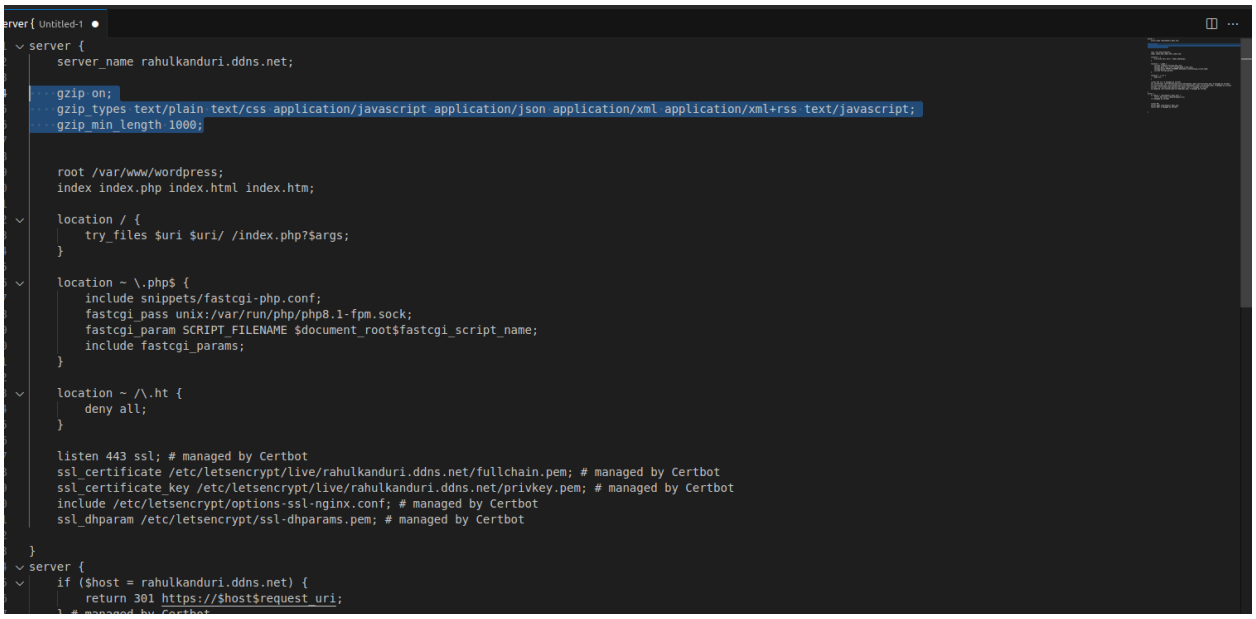
Brainstorm Technical Assignment

We are going to enable gzip compression, but what is the use of the gzip. Everytime when you do https://rahulkanduri.ddns.net/somepage.html remember you will be executing some .html or .php in simple terms we will be transferring from local machine to browser what is we can compress the file so that the transferring of the file to browser will take less time then we will use gzip
Add the following lines for enabling gzip

gzip on; ⇒ This is to enable the gzip

gzip_types text/plain text/css application/javascript application/json application/xml application/xml+rss text/javascript; ⇒ This files types are going to be compressed

gzip_min_length 1000; ⇒ To compress this files it has to be this much size in bytes



Caching: Here we will tell nginx to send and store some of the files on user's browser so that next time the user hits the files with URL instead of coming to server it can show the content from stored files from browser
Add the following lines in conf file

```
location ~* \.(jpg|jpeg|png|gif|svg|ico|css|js|pdf|txt|woff|woff2|ttf|eot|otf|webp|avif)$ {
    expires 30d;
    add_header Cache-Control "no-transform";
```

Brainstorm Technical Assignment

}

```
server {
    server_name rahulkanduri.ddns.net;

    gzip on;
    gzip_types text/plain text/css application/javascript application/json application/xml application/xml+rss text/javascript;
    gzip_min_length 1000;


    root /var/www/wordpress;
    index index.php index.html index.htm;

    location / {
        try_files $uri $uri/ /index.php?$args;
    }


    location ~* \.(jpg|jpeg|png|gif|svg|ico|css|js|pdf|txt|woff|woff2|ttf|eot|otf|webp|avif)$ {
        expires 30d;
        add_header Cache-Control "no-transform";
    }

    location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/var/run/php/php8.1-fpm.sock;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        include fastcgi_params;
    }

    location ~ /\.ht {
        deny all;
    }

    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/rahulkanduri.ddns.net/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/rahulkanduri.ddns.net/privkey.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
```

location ~* \.(jpg|jpeg|png|gif|svg|ico|css|js|pdf|txt|woff|woff2|ttf|eot|otf|webp|avif)$

In this line we are telling that any file with this extension has to copied to users browser so that he can access that file with less latency

expires 30d;

This we are telling the users browser to check for the changes in the stored files only after 30 days

add_header Cache-Control "no-transform";

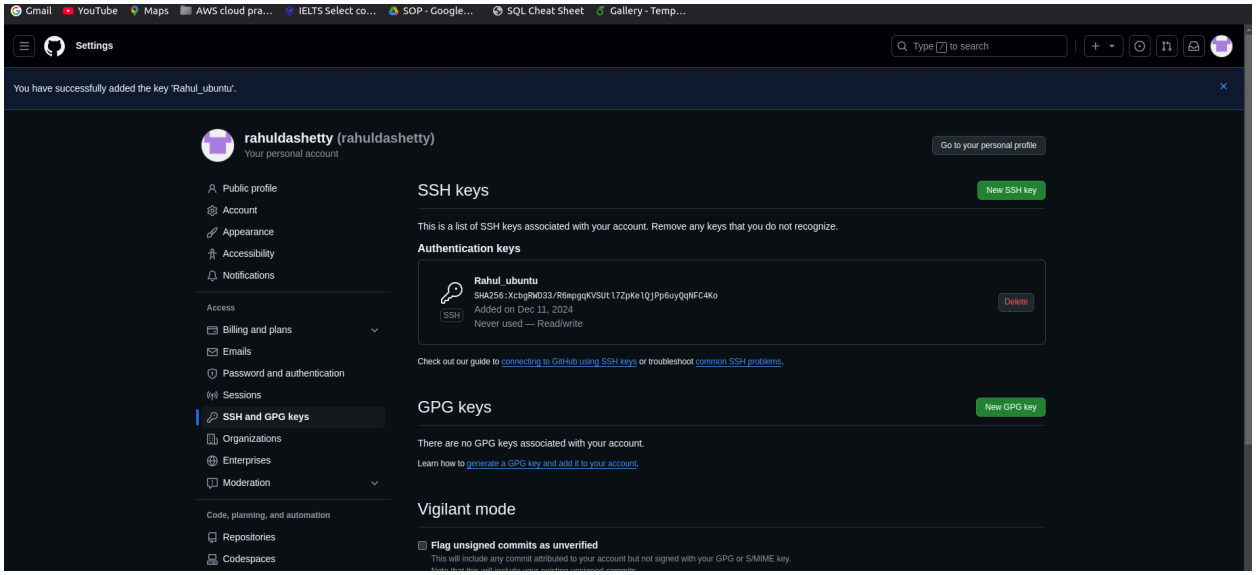
We are telling the browser to cache the data not any intermediate cdn’s only brower and without modification

Brainstorm Technical Assignment

GitHub & CI/CD:

Now in your local system we need to initialize git, we have generated the ssh key and saved the key in the github's SSH KEYS as shown, so that we are authenticated by the git hub

```
root@rahul-G3-3500:/home/rahul/Desktop/Brainstorm# ssh-keygen -t rsa -b 4096 -C "kandurirahul887@gmail.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:XcbgRWD33/R6mpgqKVSUtl7ZpKelQjPp6uyQqNFC4Ko kandurirahul887@gmail.com
The key's randomart image is:
+---[RSA 4096]----+
|         .+o+    |
|        +o =..   |
|.      o o.=+ . .|
|o       B.+o+  oo|
| o     =S+.=    +|
|o. . .. + o    . |
|+ o o. . o    . .|
|.+   oo o    o + |
|E    o+. ...o o  |
+----[SHA256]-----+
root@rahul-G3-3500:/home/rahul/Desktop/Brainstorm# cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQCukA7gQ4JulZ8KGUqSjJg4AoDvBfFIZbv6Ub26WKpYoQ8mNAXd8Vg5MOTH3FpaY5l87GG/2hk071bBM8uGBoF4/W7ytpqiNvULjGtWjJOP4GR5RA212oufm0iabUdBokaivIRBA6KGzl+49AzILdCWCaCGXVsAtyvFoOV3
ekosse48vcQ2IGl2x1PAdBee8RW/K14dCNQ01SXXjt+dd5jGZQ8NLAS4fe+l9MFK4x9Mq1rs41WDphN+z8D3VSJJOmRp5Dwbt7VByE89580UkVXkEocaRwkUfAwxoxEjyeAqcZXfeYwxlrK5HAQw1Hi1UNOKl4N8dk1A9HsV8QrdOUyuJCYFCM9dwyF3E2qJLMdv0Pt2AtRw
sRdDn4cNWI7CC70k+GUuj0Z2FiG49Gxfuxh49vAyWWZmbp1BSN1JLXD/5BGBwhEd/PhTRR75f3NSLeOh9V0i8UNg4r8RMGzV4LXLN4Pqx/b66F7sBFjBL9+tDAoRvdWEHM24eFL52BXqlSlB84g0Xh6INGv3BrRMl5MC8ErnrgM0wrXN2rdyZN9ljhcP3zbTVvoFkjY422lm
njYyzxoO0DgAx39OpAnv7lzdXPoBgNrpXYB4nkMmJSN2hLt7oO6j6F6tc9zZmesdkSh1vDoZa4NSyKmXuOgs/BuNOHfxAac0OkhHlI260JElaQ== kandurirahul887@gmail.com
root@rahul-G3-3500:/home/rahul/Desktop/Brainstorm# git remote set-url origin git@github.com:rahuldashetty/Brainstorm.git
root@rahul-G3-3500:/home/rahul/Desktop/Brainstorm#
root@rahul-G3-3500:/home/rahul/Desktop/Brainstorm# git remote -v
origin  git@github.com:rahuldashetty/Brainstorm.git (fetch)
origin  git@github.com:rahuldashetty/Brainstorm.git (push)
root@rahul-G3-3500:/home/rahul/Desktop/Brainstorm#
root@rahul-G3-3500:/home/rahul/Desktop/Brainstorm# ssh -T git@github.com
Hi rahuldashetty! You've successfully authenticated, but GitHub does not provide shell access.
root@rahul-G3-3500:/home/rahul/Desktop/Brainstorm# git branch
* main
root@rahul-G3-3500:/home/rahul/Desktop/Brainstorm# git push origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 228 bytes | 228.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To github.com:rahuldashetty/Brainstorm.git
 * [new branch]      main -> main
root@rahul-G3-3500:/home/rahul/Desktop/Brainstorm#
```

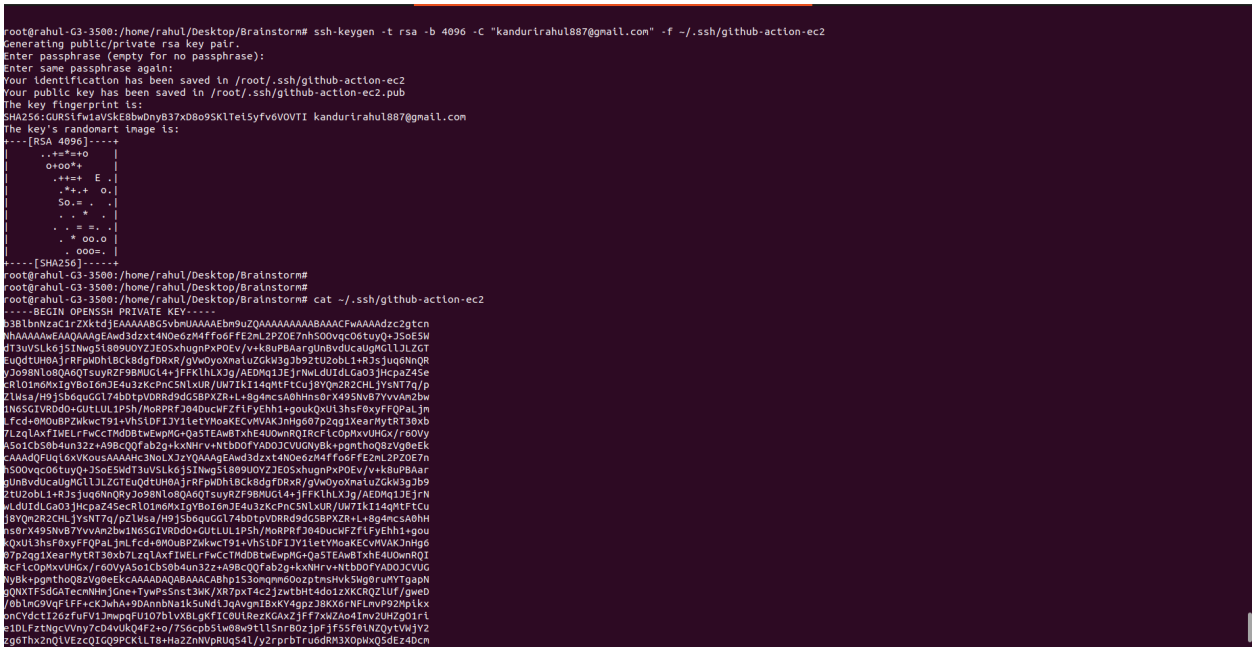


Brainstorm Technical Assignment

Now we need to copy key from our local machine to ec2 instance so first we generate the key then we will copy the content into ec2

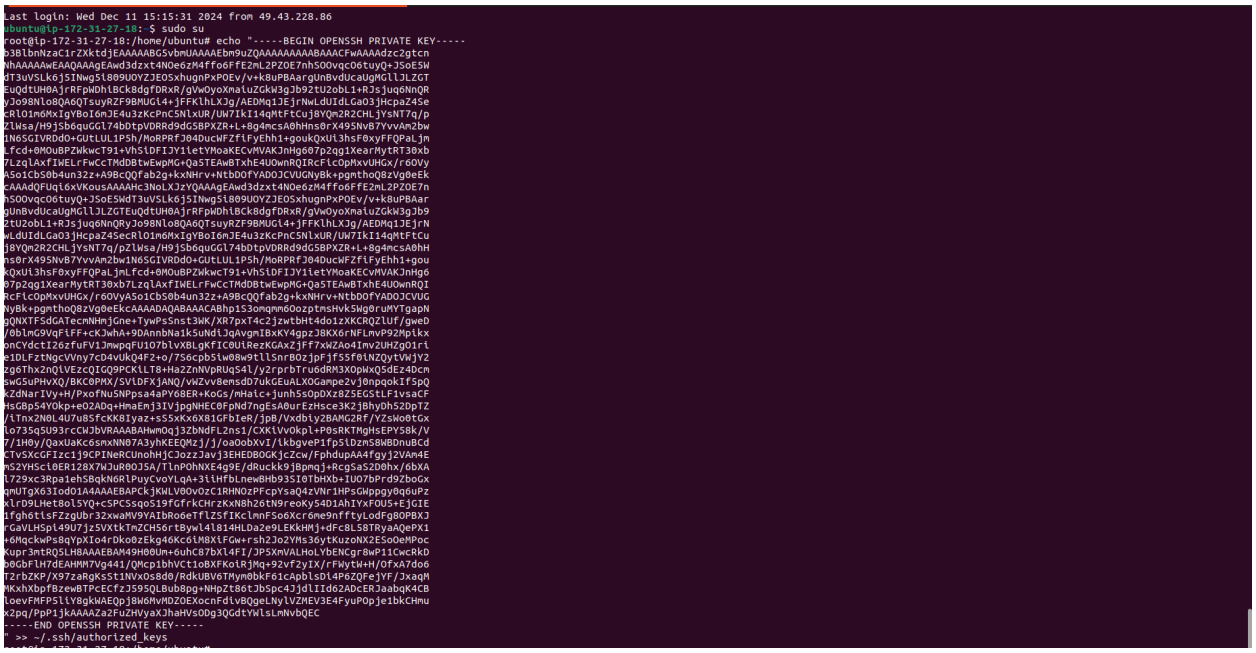
ssh-keygen -t rsa -b 4096 -C "kandurirahul887@gmail.com" -f ~/.ssh/github-action-ec2

cat ~/.ssh/github-action-ec2

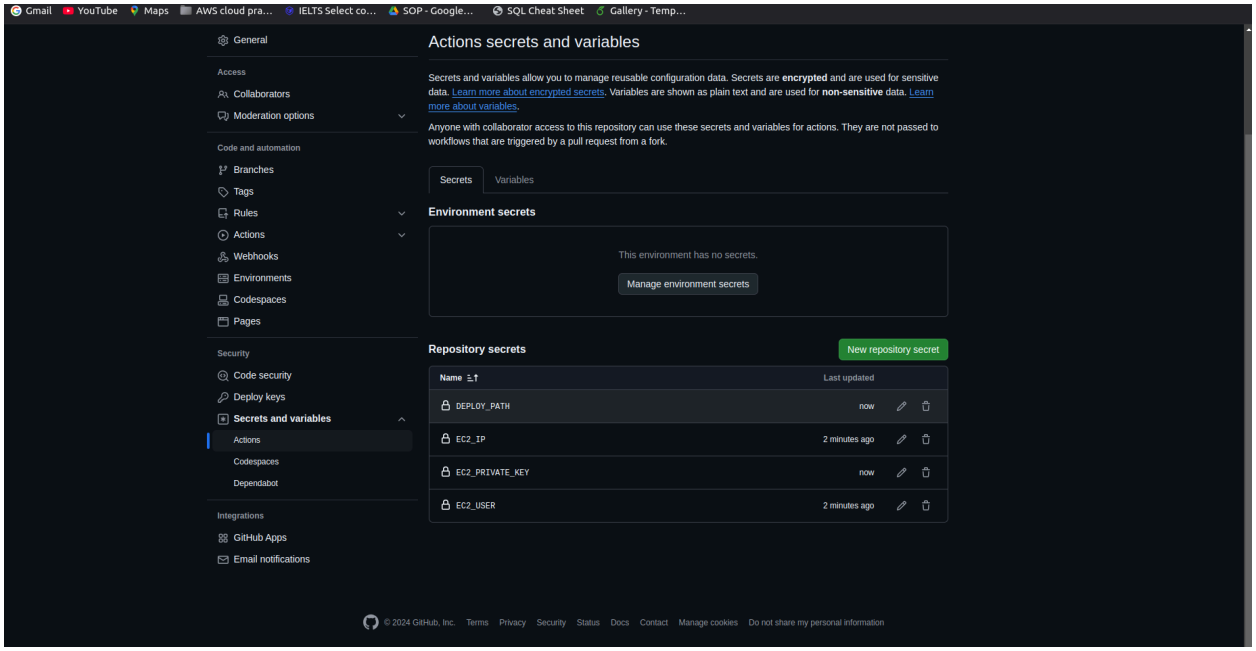


Brainstorm Technical Assignment

Then in ec2 instance paste the contents into .ssh folder as shown



Then we add required repository secrets as shown

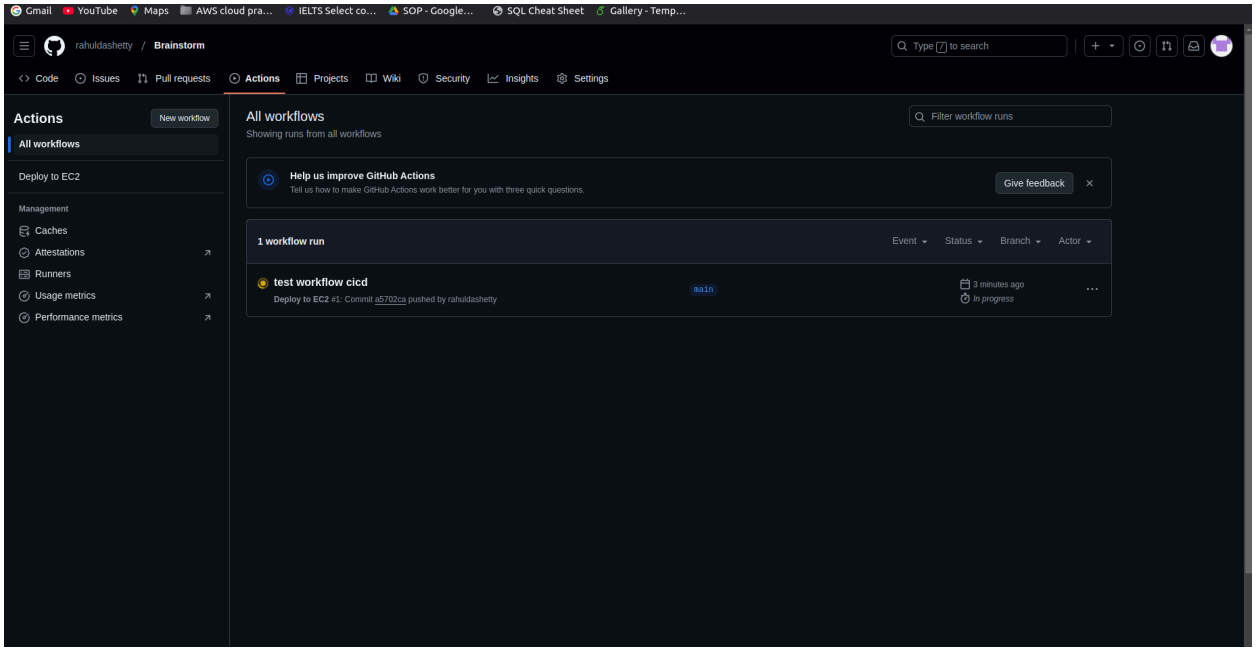


We have added an simple .php file and we also created deploy.yml file inside the github/workflows directory so that if any push see the actions being done

Brainstorm Technical Assignment

```
root@rahul-G3-3500:/home/rahul/Desktop/Brainstorm# ls
README.md
root@rahul-G3-3500:/home/rahul/Desktop/Brainstorm# nano mycode.php
root@rahul-G3-3500:/home/rahul/Desktop/Brainstorm# nano mycode.php
root@rahul-G3-3500:/home/rahul/Desktop/Brainstorm# git add .
root@rahul-G3-3500:/home/rahul/Desktop/Brainstorm# nano .github/workflows/deploy.yml
root@rahul-G3-3500:/home/rahul/Desktop/Brainstorm# git add .
root@rahul-G3-3500:/home/rahul/Desktop/Brainstorm# git commit -m "test workflow cicd"
[main a5702ca] test workflow cicd
 2 files changed, 44 insertions(+)
 create mode 100644 .github/workflows/deploy.yml
 create mode 100644 mycode.php
root@rahul-G3-3500:/home/rahul/Desktop/Brainstorm# git push origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (6/6), 873 bytes | 873.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To github.com:rahuldashetty/Brainstorm.git
   3644179..a5702ca  main -> main
root@rahul-G3-3500:/home/rahul/Desktop/Brainstorm#
```

After pushing we can see the actions that is being done



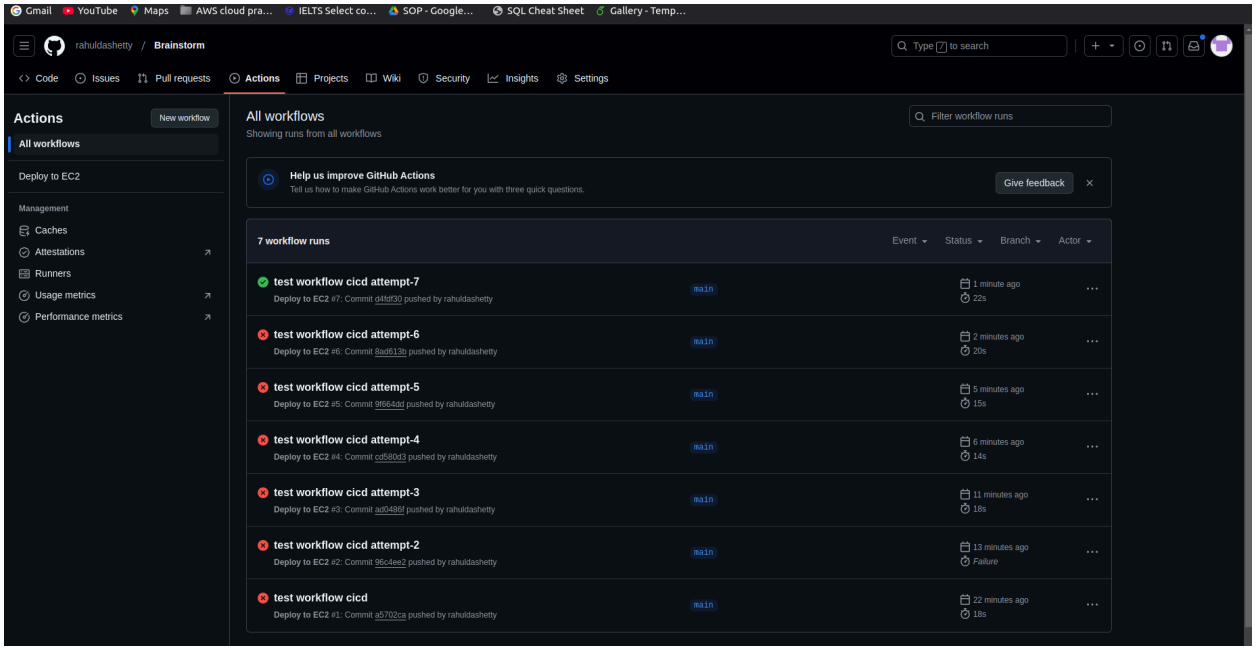
Brainstorm Technical Assignment

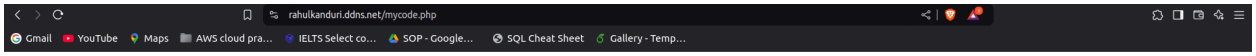
```
root@rahul-G3-3500:/home/rahul/Desktop/Brainstorm# cat .github/workflows/deploy.yml
name: Deploy to EC2

on:
  push:
    branches:
      - main

jobs:
  deploy:
    runs-on: ubuntu-latest  # Runs the job on an Ubuntu runner

    steps:
    - name: Checkout repository
      uses: actions/checkout@v3

    - name: Set up SSH key
      run: |
        mkdir -p ~/.ssh
        echo "$EC2_PRIVATE_KEY" > ~/.ssh/id_rsa
        chmod 600 ~/.ssh/id_rsa
      env:
        EC2_PRIVATE_KEY: ${{ secrets.EC2_PRIVATE_KEY }}

    - name: Add EC2 host to known_hosts
      run: |
        ssh-keyscan -H ${{ secrets.EC2_IP }} >> ~/.ssh/known_hosts

    - name: SCP mycode.php to EC2
      run: |
        scp -i ~/.ssh/id_rsa ./mycode.php ${{ secrets.EC2_USER }}@${{ secrets.EC2_IP }}:/home/ubuntu/
      env:
        EC2_USER: ${{ secrets.EC2_USER }}
        EC2_IP: ${{ secrets.EC2_IP }}

    - name: Move mycode.php to /var/www/wordpress
      run: |
        ssh -i ~/.ssh/id_rsa ${{ secrets.EC2_USER }}@${{ secrets.EC2_IP }} << 'EOF'
          sudo mv /home/ubuntu/mycode.php /var/www/wordpress/
          sudo chown www-data:www-data /var/www/wordpress/mycode.php
          sudo chmod 644 /var/www/wordpress/mycode.php
        EOF
      env:
        EC2_USER: ${{ secrets.EC2_USER }}
        EC2_IP: ${{ secrets.EC2_IP }}

    - name: Restart Nginx (if needed)
      run: |
        ssh -i ~/.ssh/id_rsa ${{ secrets.EC2_USER }}@${{ secrets.EC2_IP }} 'sudo systemctl restart nginx'
      env:
        EC2_USER: ${{ secrets.EC2_USER }}
        EC2_IP: ${{ secrets.EC2_IP }}
root@rahul-G3-3500:/home/rahul/Desktop/Brainstorm#
```

This is the deploy.yml file

After really brainstorming with the error we can finally fix the issue

Brainstorm Technical Assignment





Hello this is Rahul