Hamza Muhammad Khan

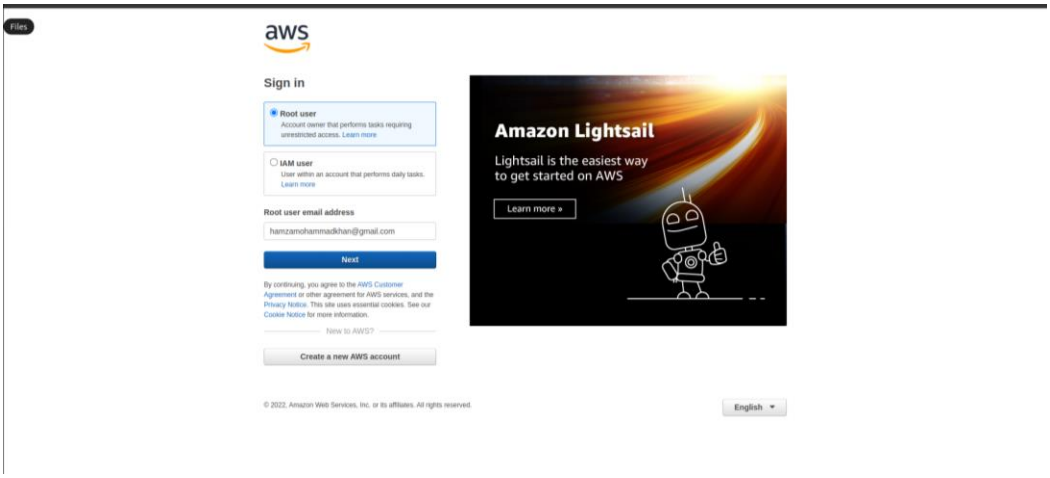
STACK SETUP USING AWS EC2

Hamza Muhammad Khan

CONNECT TO EC2 INSTANCE

We were given the task of making the cloudways thunderstack setup in the Amazon EC2 instance. The following are the steps taken to make the thunderstack but first, we have to create an instance first:

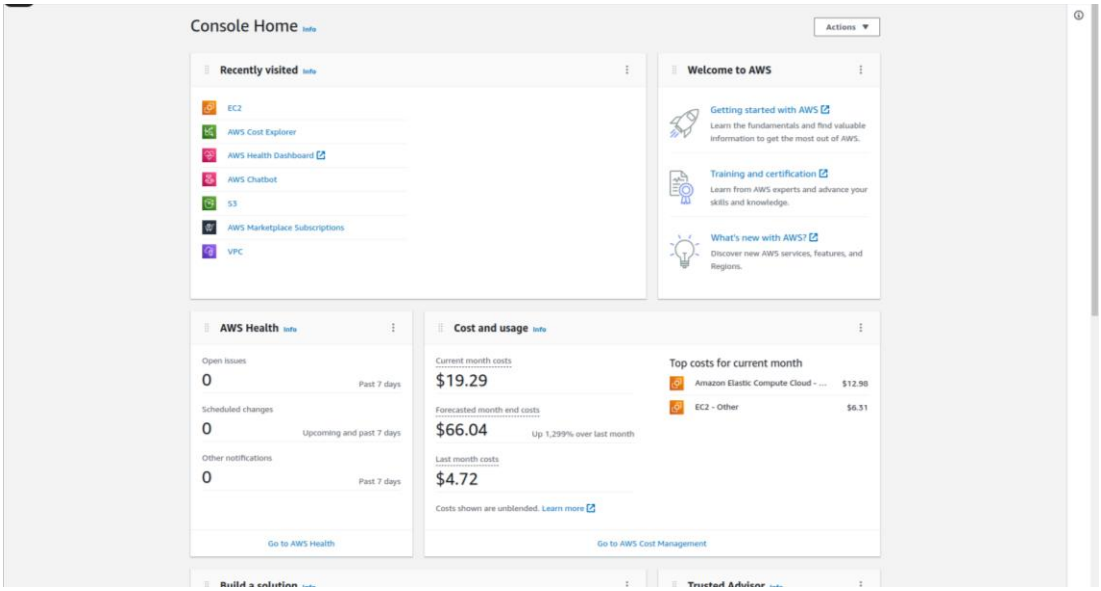
1. Log in to your Amazon account.



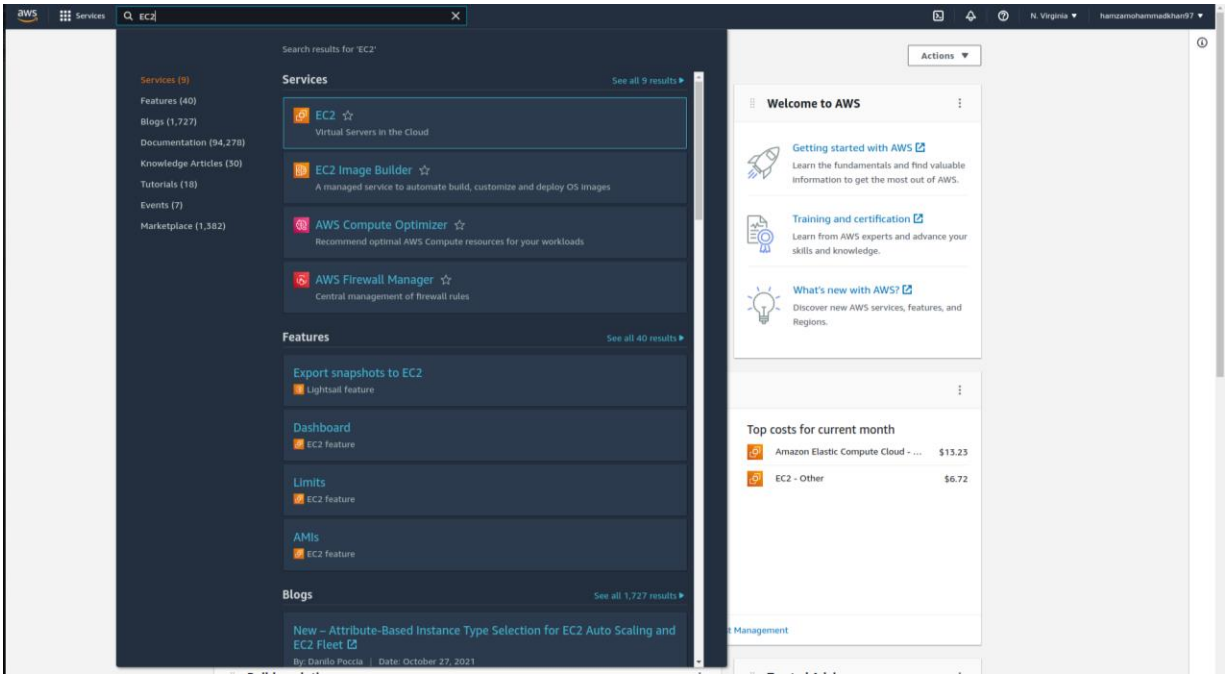
2. Now you will be prompted to your dashboard.

CLOUDWAYS

Hamza Muhammad Khan

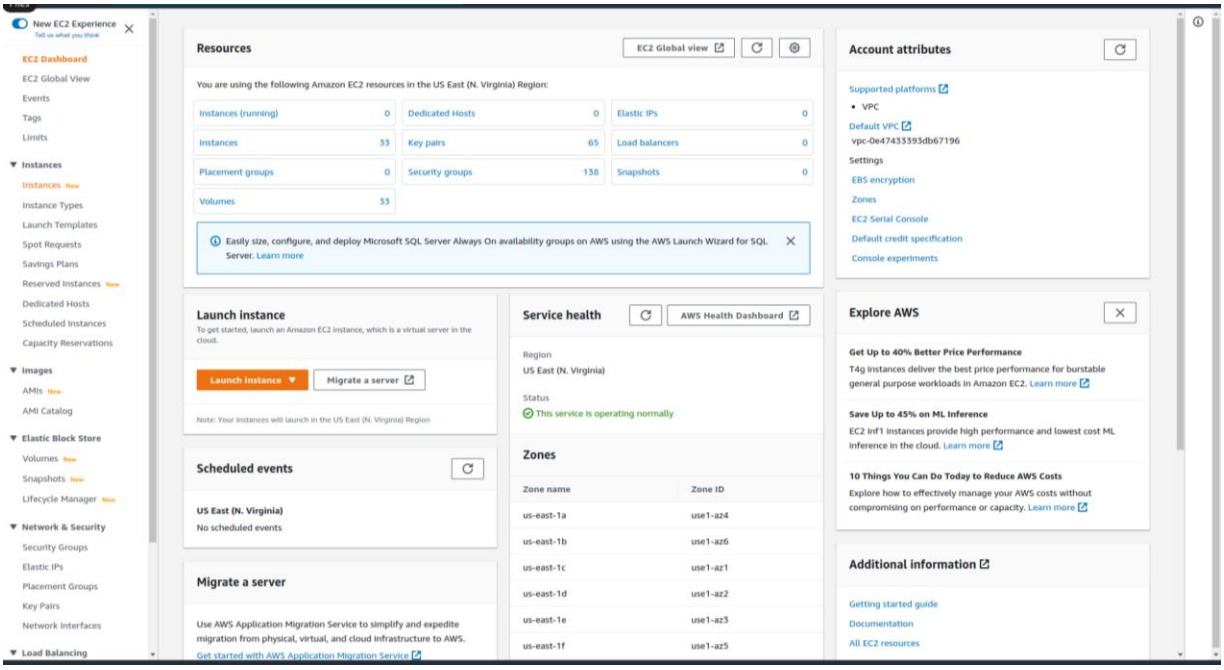


3. Go to your EC2 instance from your Home menu or search EC2 instance if it's not showing

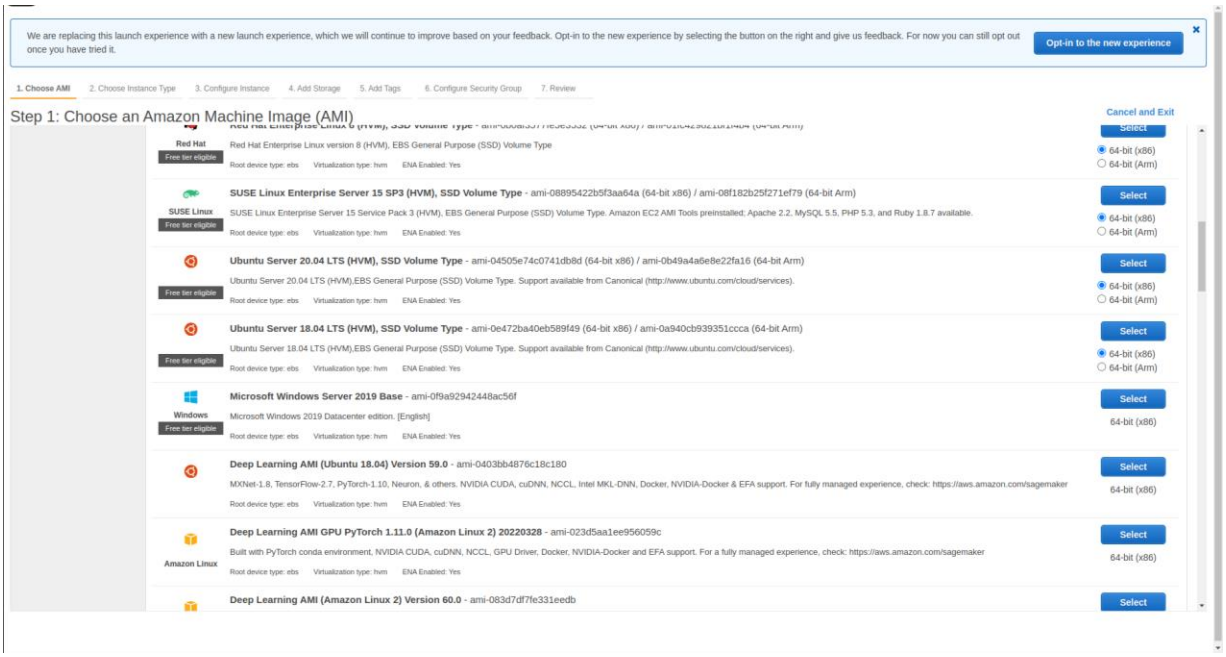


4. Now you will see launch Instance option.

Hamza Muhammad Khan



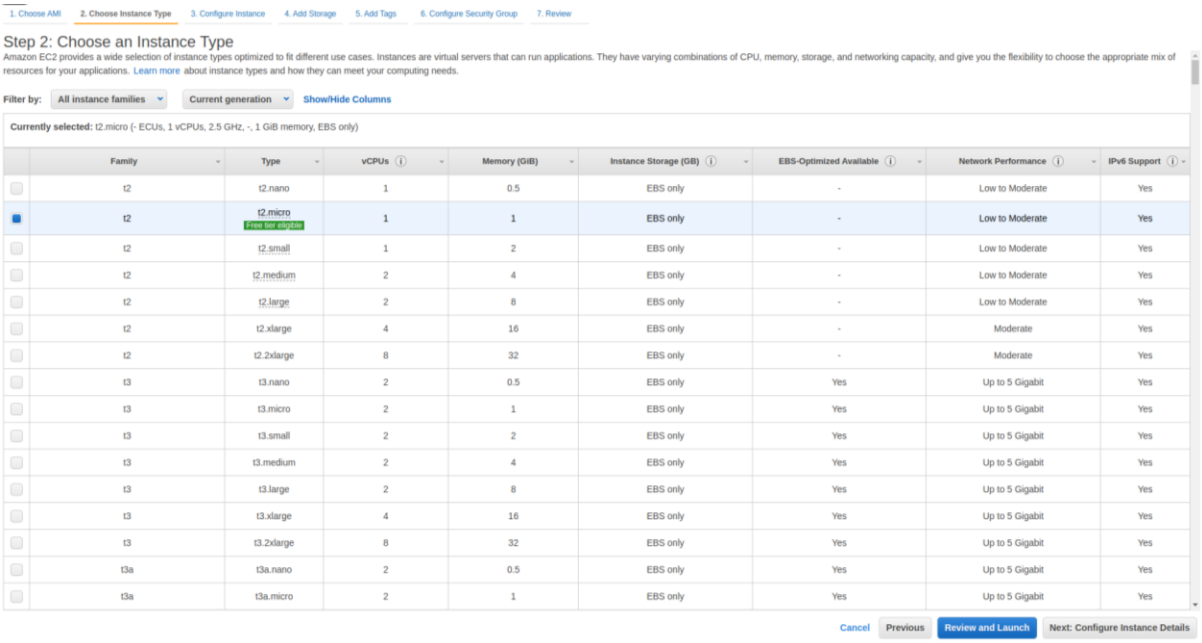
5. Now we will see abundant options to choose Amazon Machine, we will be choosing free-tier Ubuntu 20.04 LTS (HVM).



NOTE: Choose free-tier services.

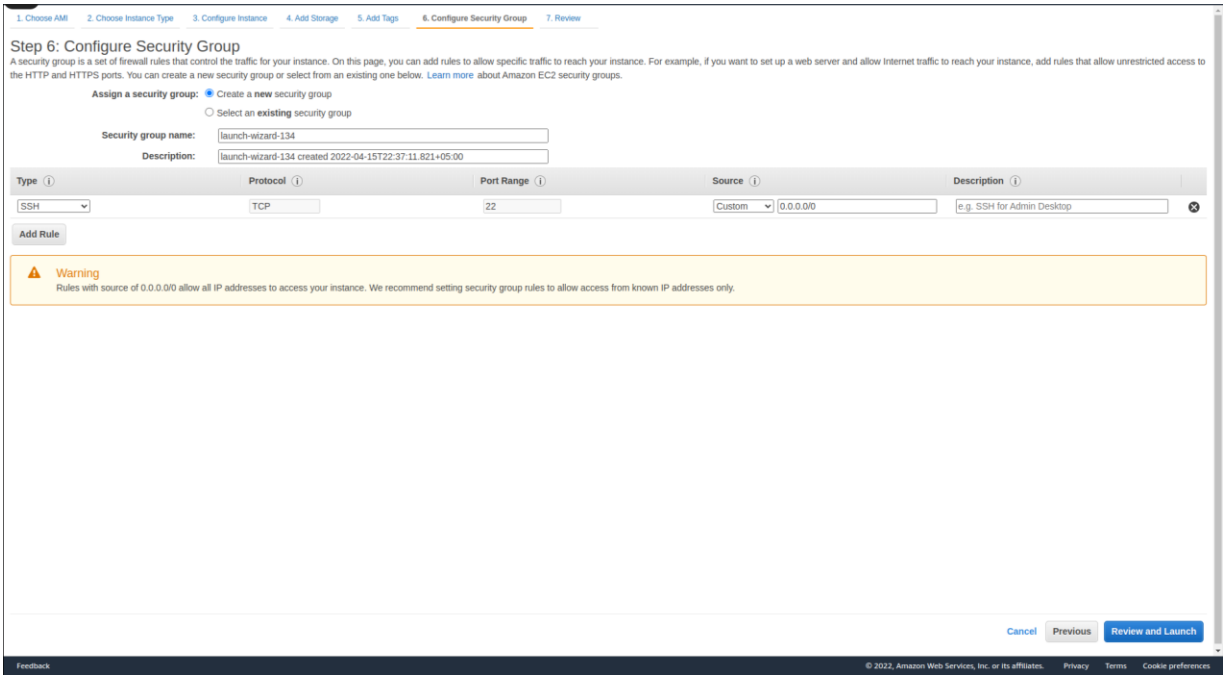
6. Now select t2-micro free-tier eligible.

Hamza Muhammad Khan

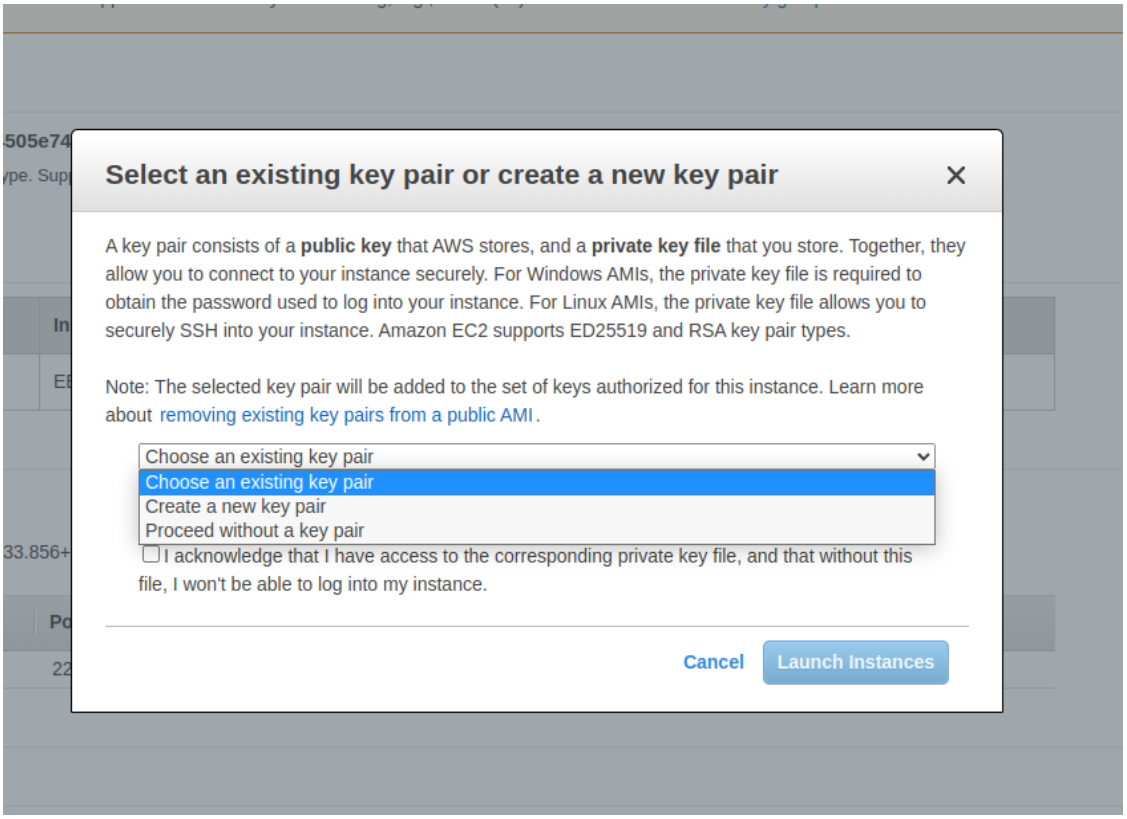


7. Then give security inbound rules so you can access your website. You have to give 4 rules, go to Add rule then add:
    a. Type: Custom TCP (same for all)
    b. Protocol: TCP (same for all)
    c. Port Range: 443, 8080,8081,80
    d. Source: Custom (same for all)

Hamza Muhammad Khan



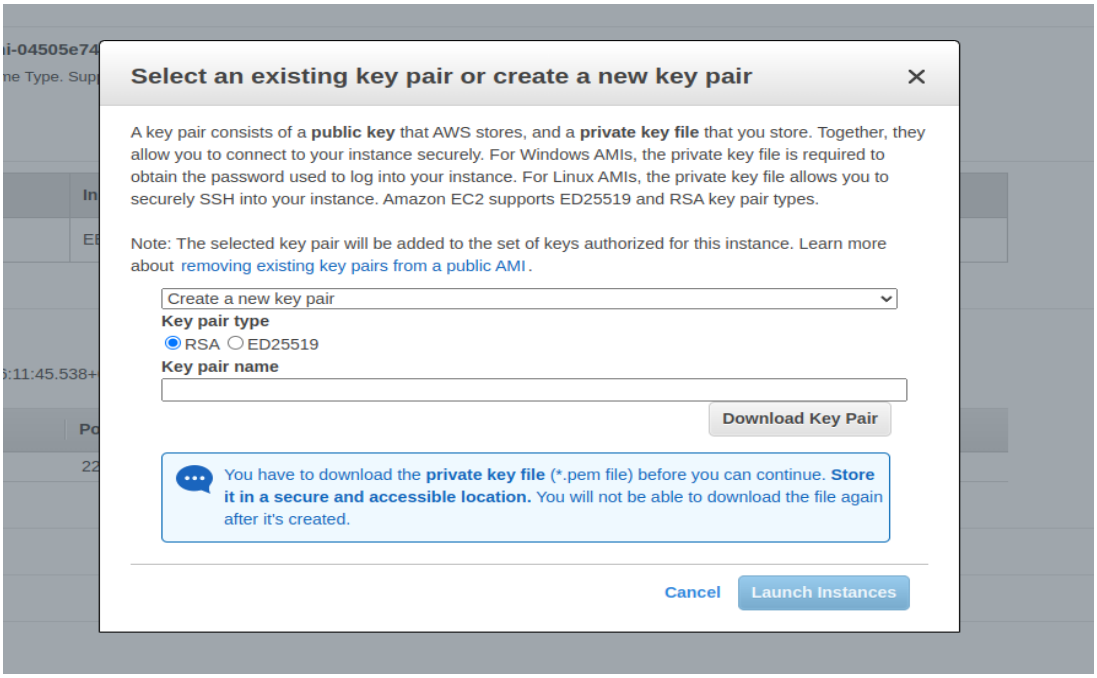
8. Review and launch, then it will ask for a key pair you want to create, choose or proceed without (it is advised to use key pair for security purposes)



9. Create a key pair give it a name choose the RSA algorithm or ED25519. Mostly choose RSA because it is widely supported everywhere but people recommend choosing ED25519 because it is much more secure and has a smaller key pair, but for now, will be using RSA. Give a name and download it. Two keys will be used, one will be downloaded in your system download folder and one will be in AWS.

CLOUDWAYS

Hamza Muhammad Khan

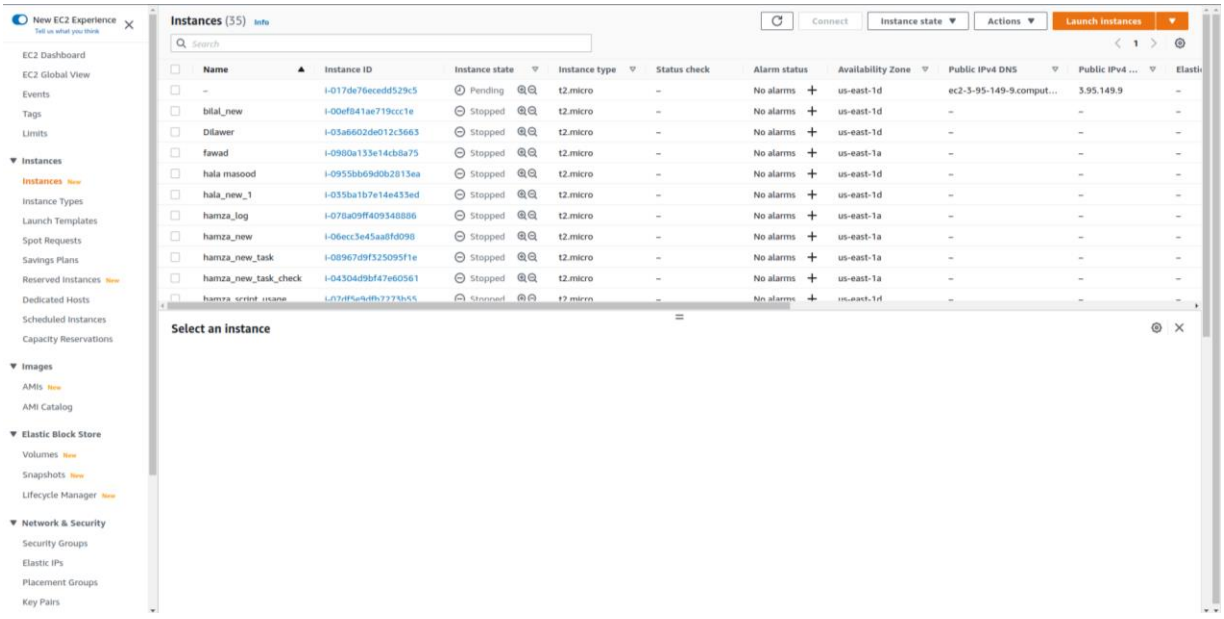


A BIT OF INFORMATION ABOUT KEYS:

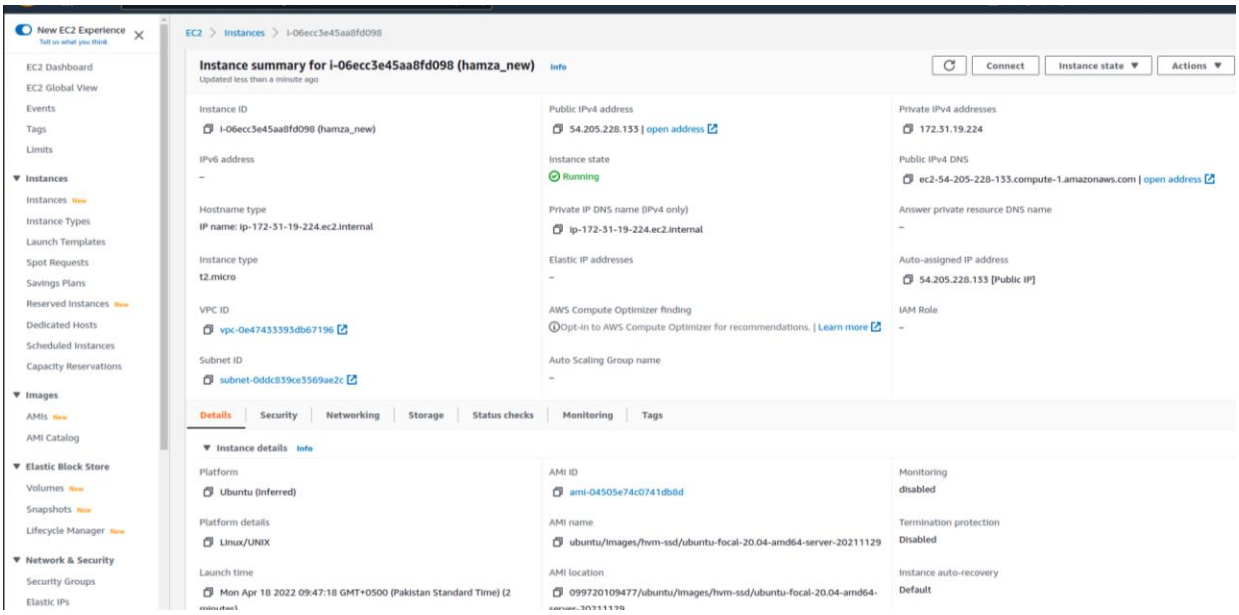
a. **ED25519**: Part of the EdDSA algorithm (**Edwards-curve Digital Signature Algorithm** (**EdDSA**)) which is based on twisted Edward curves which are plane models of elliptic curves, and Curve25519 which also uses elliptic curves. Some benefits ED25519 has
   i. Fast single-signature verification.
   ii. Fast key generation.
   iii. Short keys as compared to RSA.
   iv. Collision resilience.

b. **RSA**: Known as Asymmetric Encryption, it is done by making an encrypted code called a public key, which can be shared openly and once the public key is created it can only be decrypted by a private key. So in AWS public key will be given to you and the private key will in AWS.

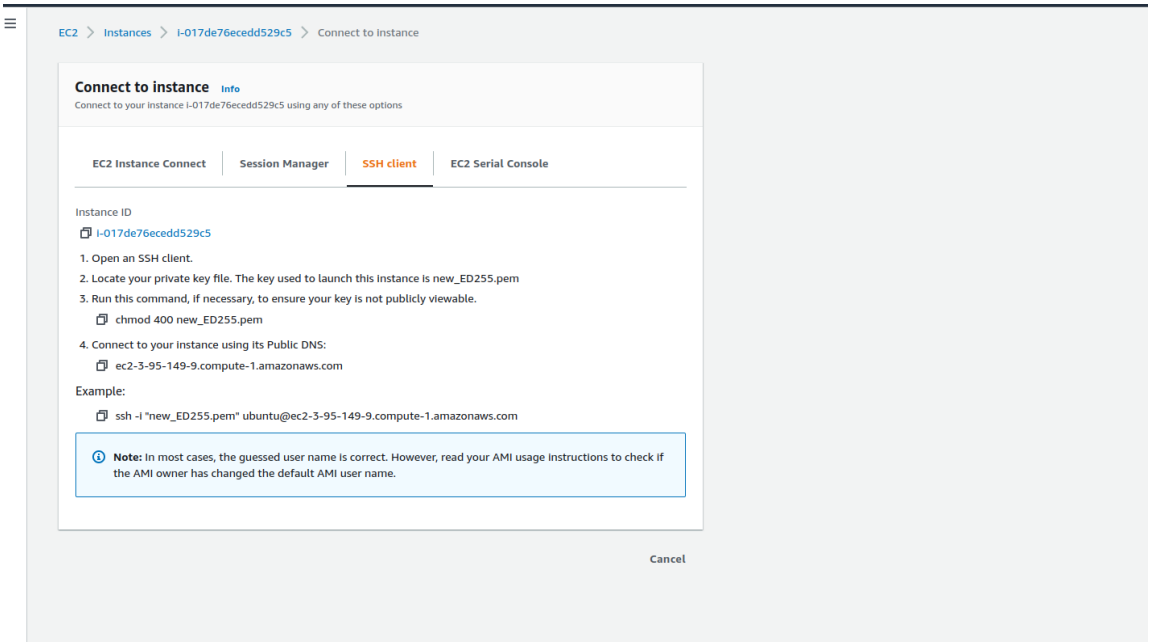
10. Now launch your instance do name your instance to clear the confusion.



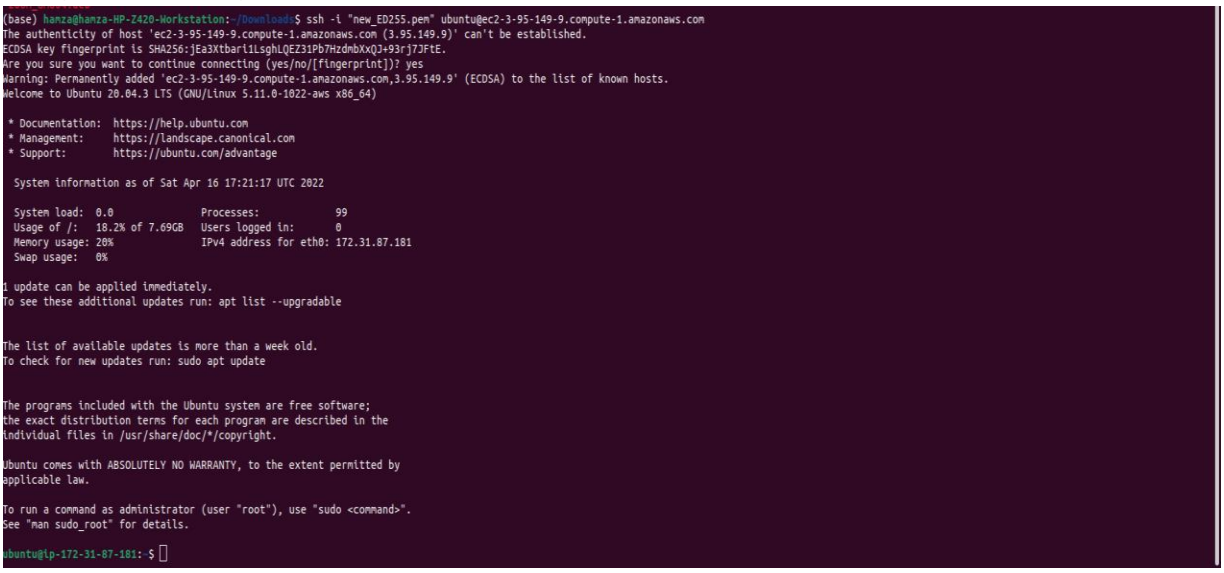
11. Now go to your instance it will launch shortly. Then go to connect and you will be shown 4 options, EC2 instance connects, Session Manager, SSH Client, and EC2 serial console. Go to SSH client and follow the instruction given.



Hamza Muhammad Khan



12. After following the instructions you will be connected with your AWS on your local pc.



CONNECTION EXPLANATION:

Hamza Muhammad Khan

a. First, we do is go to the folder where our public key is downloaded and make it executable by typing **chmod 400 key.pem** this means the key is executable to the root user.
b. Second, we do is copy the example given in SSH Client AWS to paste it into our local terminal, make sure you are still in the folder where the key is, paste the command and you will be connected. The command **ssh -i** means is identify, meaning identify the key then connect. It will ask for connection say yes and you will be connected

Hamza Muhammad Khan

1. MAKE REVERSE PROXY

Now that we have made an instance now we will make a reverse proxy server, not by manually installing it but by making a script that can perform our task, just using minimum inputs.

The script will install the following packages in order to achieve reverse proxy:

a. **Nginx**: This will act as a reverse proxy server and deal with all client requests. (load balancing, protection from attacks, caching)
b. **Varnish**: It will be used as a cache accelerator. It will give you cache service and will sit between the Nginx and backend server.
c. **Apache**: It will act as a backend server and will be controlling our website using PHP-FPM
d. **PHP-FPM**: We will be using PHP-FPM instead of PHP because of its compatibility with Nginx.
e. **MariaDB**: As your sites need a database we will be using MariaDB. It's open-source, it's always updated, and easy to use. It's just a Fork of MySQL.

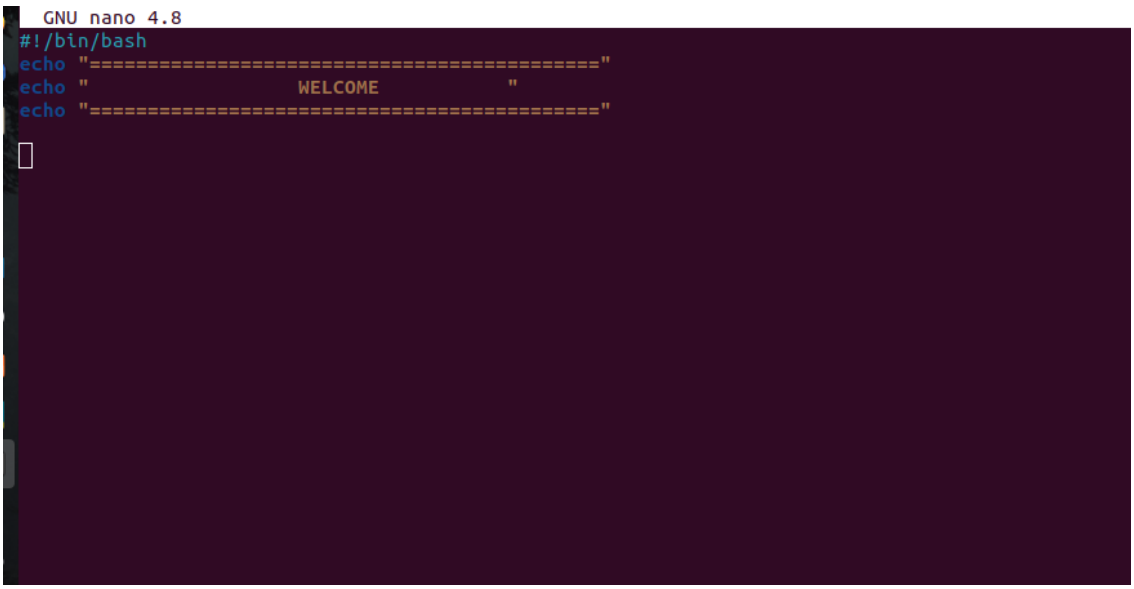
1. Create a bash filename **test.sh,** by using touch or nano, like
   `nano test.sh` or `touch test.sh`

   The difference between nano and test is that
   `nano` is used for creating and editing files.
   `touch` is just for creating files.

2. But before editing the file, check in the terminal which bash you have, by typing `which bash` it will show `/usr/bin/bash` so the file you created type `#!/bin/bash` meaning you will be using a bash interpreter. After writing press Ctrl + O to save it and press Enter.

3. Now that you have saved the file now it's time for some editing inside it. For welcoming write some echo commands to make it looks a little interactive.

```
echo "=========================================="
echo "                WELCOME                   "
echo "=========================================="
```



NGINX:

4. Now it is always recommended, to update your system so type the code or command `sudo apt-get update`, it will install all necessary updates your system needs. Then we will add a code of if condition because if the user has

installed a package so it will ask the user first do you want to install this package if the user types N then it will show the version, if Y then it will start the installation. First you will install Nginx by using the code:

```
echo "============Installing Nginx============="
read -r -p "Do you want to install nginx? [y/N] " response
if [[ "$response" =~ ^([yY][eE][sS]|[yY])$ ]]
then
    sudo apt install nginx -y
    #configure nginx files
    sudo sed -i '12i upstream backend{\n server 127.0.0.1:8080
fail_timeout=5s weight=5; \n server 127.0.0.1:8081 backup; \n
#upstream \n}' /etc/nginx/nginx.conf
    sudo sed -i '50i proxy_pass http://backend;\n
proxy_set_header Host $host; \nproxy_set_header X-Real-IP
$remote_addr;\n proxy_set_header X-Forwarded-For
$proxy_add_x_forwarded_for;\n proxy_set_header X-Forwarded-
Proto $scheme;' /etc/nginx/sites-available/default
    sudo sed -i '59i location ~ \.php$ \n { \nfastcgi_pass
unix:/var/run/php/php8.1-fpm.sock;\ninclude
fastcgi_params;\nfastcgi_index index.php;\nfastcgi_param
SCRIPT_FILENAME $document_root$fastcgi_script_name;}'
/etc/nginx/sites-available/default
    sudo sed -i '46i  server_name  hamzamkhan.com;'
/etc/nginx/sites-available/default
    sudo systemctl enable nginx
    sudo systemctl restart nginx
else
    nginx -v
fi
```

Sed mean stream editor which means it will be used as a search, replace or add, command **-i** means it will identify where you want to do the replacement or addition.

There will addition in 2 files nginx.conf inside `/etc/nginx/` and virtual host file inside `/etc/nginx/sites-available/` , as you can see in the code, the upstream backend code

What is upstream ?

The upstream module in nginx is used to define group of servers

That are represent by `proxy_pass, fastcgi_pass, uswcgi_pass, scgi_pass, memcached_pass` and `grpc_pass`.

For more info visit :

http://nginx.org/en/docs/http/ngx_http_upstream_module.html

As you can see we are adding in a particular portion of the files by typing **Ni (N means any line number)** . In the end we will enable nginx and restart it. The if condition let's you decide what you want to do it, consider if you have installed nginx then just press N and you will be shown what version you have installed, if you haven't installed it then press Y and it will be installed. It is not necessary to press Y as you can see in the condition you can type y,Y,yEs,YEs etc it will be installed.

VARNISH:

5. Now we will install varnish it will be the same as Nginx. Varnish is used caching your webpage, it is used as a cache accelerator, it will reduce the strain on Apache. The following code will show how to install and also configure files of varnish. In varnish we will change it's default port to 8080 in `/lib/systemd/system/varnish.service` where **-a** is and in `/etc/default/varnish` where `.ports` is replace it to 8081. 8080 is the listening port for varnish and 8081 is accessing your backend server.

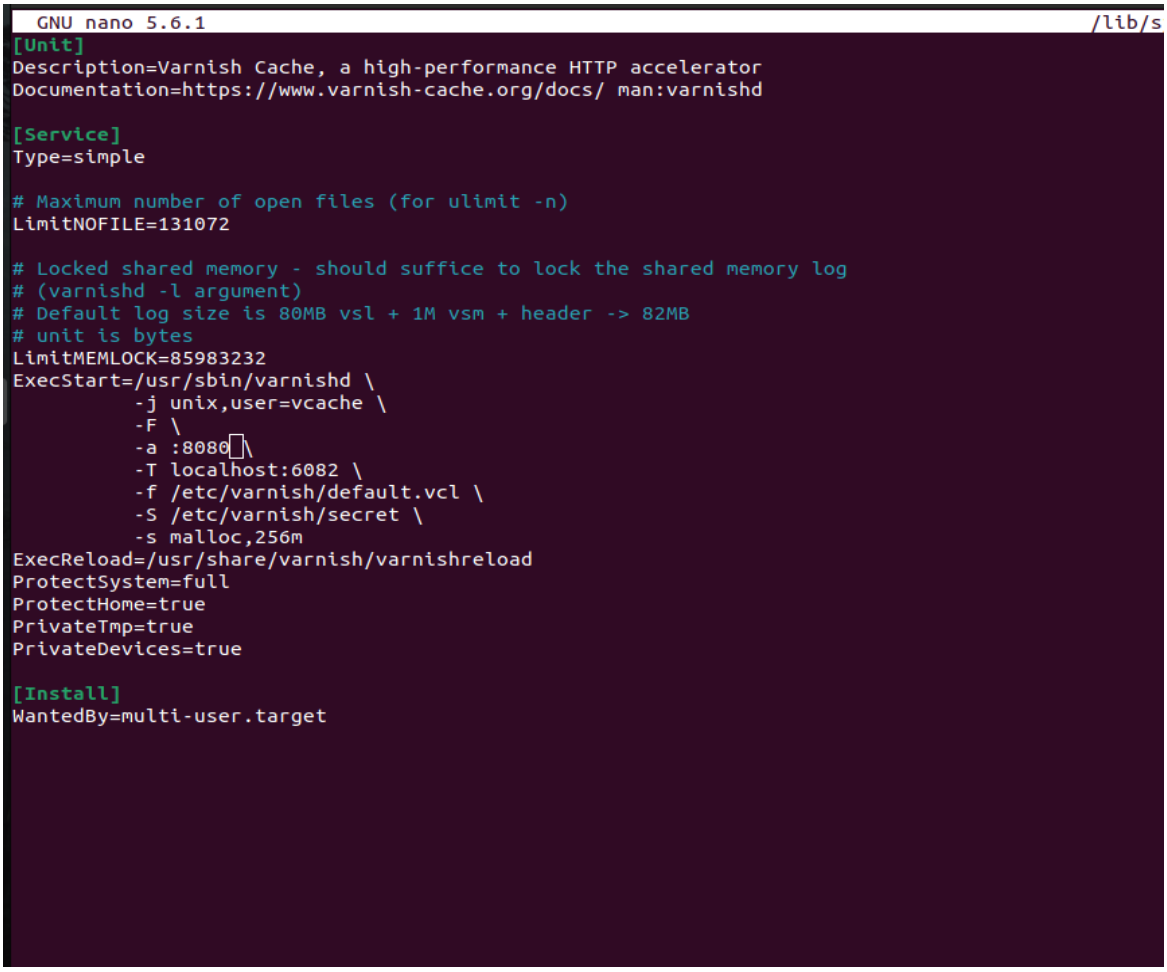
```
echo "============Installing Varnish============="
read -r -p "Do you want to install varnish? [y/N] "
```


```
response_varnish
if [[ "$response_varnish" =~ ^([yY][eE][sS]|[yY])$ ]]
then
    sudo apt install vanish -y
    #configure varnish port in different files
    sudo sed -i "s/6081/8080/g"
/lib/systemd/system/varnish.service
    sudo sed -i "s/8080/8081/g" /etc/varnish/default.vcl
    sudo sed -i '39i if (obj.hits > 0)\n {\n set resp.http.X-
Varnish-Cache = "HIT"; \n} \nelse\n {\nset resp.http.X-
Varnish-Cache = "MISS";\n}' /etc/varnish/default.vcl
    sudo systemctl enable varnish.service
    sudo systemctl restart varnish.service
else
    varnishd -v
fi
```

Hamza Muhammad Khan

```
 GNU nano 5.6.1                                                  default.vcl
#
# This is an example VCL file for Varnish.
#
# It does not do anything by default, delegating control to the
# builtin VCL. The builtin VCL is called when there is no explicit
# return statement.
#
# See the VCL chapters in the Users Guide for a comprehensive documentation
# at https://www.varnish-cache.org/docs/.

# Marker to tell the VCL compiler that this VCL has been written with the
# 4.0 or 4.1 syntax.
vcl 4.1;

# Default backend definition. Set this to point to your content server.
backend default {
    .host = "127.0.0.1";
    .port = "8081";
}

sub vcl_recv {
    # Happens before we check if we have this in cache already.
    #
    # Typically you clean up the request here, removing cookies you don't need,
    # rewriting the request, etc.
}

sub vcl_backend_response {
    # Happens after we have read the response headers from the backend.
    #
    # Here you clean the response headers, removing silly Set-Cookie headers
    # and other mistakes your backend does.
}

sub vcl_deliver {
    # Happens when we have all the pieces we need, and are about to send the
    # response to the client.
    #
if (obj.hits > 0)
 {
 set resp.http.X-Varnish-Cache = "HIT";
}
else
 {
set resp.http.X-Varnish-Cache = "MISS";
}
# You can do accounting or modifying the final object here.
}

```

Changes in default.vcl



Changes in varnish.service

PHP-FPM:

6. Now we will install php-fpm as nginx is compatible with php-fpm. Make sure you php-fpm version is 7.4 or above. The following code will install php.

```
read -r -p "Do you want to install php [y/N]" response_php
if [[ "$response_php" =~ ^([yY][eE][sS]|[yY])$ ]]
then
        sudo apt install software-properties-common && sudo
add-apt-repository ppa:ondrej/php -y
        sudo apt upgrade -y
        sudo apt install php8.1-fpm libapache2-mod-fcgid
```

```
php8.1-mysql
        sudo systemctl enable php8.1-fpm
        sudo systemctl restart php8.1-fpm
     sudo apt install php8.1-curl php8.1-gd php8.1-mbstring
php8.1-xml php8.1-xmlrpc php-soap php8.1-intl php8.1-zip
sudo a2enmod proxy_fcgi setenvif
          sudo a2enconf php8.1-fpm

          #restart apache
          sudo systemctl restart apache2

else
sudo systemctl status php8.1-fpm
fi
```

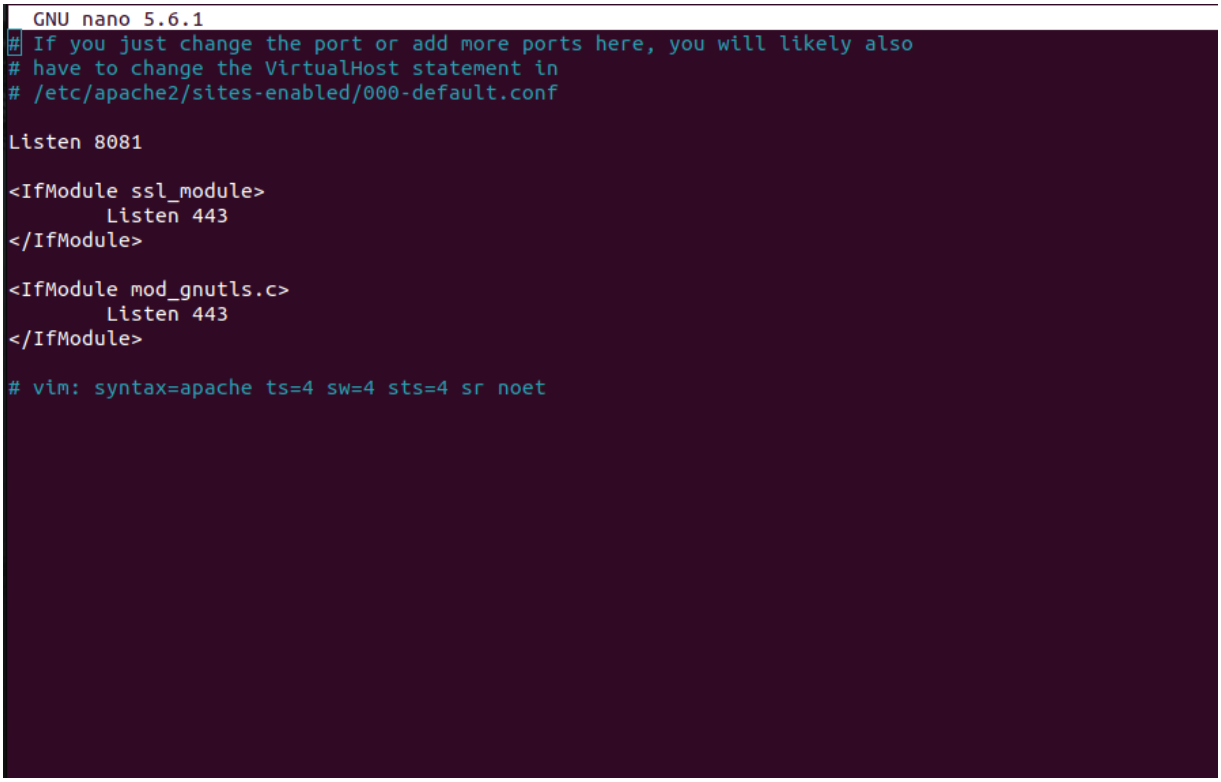
APACHE:

7. Apache will act as a backend server, and the content of your site will work on it. The following code will configure and install apache.

```
echo "=============Installing Apache==========="
read -r -p "Do you want to install apache2? [y/N] "
response_apache
if [[ "$response_apache" =~ ^([yY][eE][sS]|[yY])$ ]]
then
   sudo apt install apache2 -y
   #configure apache port
   sudo sed -i "s/80/8081/g"  /etc/apache2/ports.conf
   sudo sed -i  "s/80/8081/g" /etc/apache2/sites-
available/000-default.conf
   sudo sed -i "10i ServerName hamzamkhan.com"
/etc/apache2/sites-available/000-default.conf
   sudo sed -i "11i ServerAlias www.hamzamkhan.com"
/etc/apache2/sites-available/000-default.conf
```

```
    sudo sed -i "24i  <FilesMatch \.php$>\n # 2.4.10+ can proxy
to unix socket \nSetHandler 'proxy:unix:/var/run/php/php8.1-
fpm.sock|fcgi://localhost'\n </FilesMatch>"
    sudo systemctl enable apache2.service
    sudo systemctl restart apache2.service
else
     apache2 -v
fi
```

In apache we also have to change ports, in ports.conf and in virtual host.



This is the ports.conf file

```
  GNU nano 5.6.1                                          000-default.conf
<VirtualHost *:8081>
        # The ServerName directive sets the request scheme, hostname and port that
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerName
        # specifies what hostname must appear in the request's Host: header to
        # match this virtual host. For the default virtual host (this file) this
        # value is not decisive as it is used as a last resort host regardless.
        # However, you must set it for any further virtual host explicitly.
        #ServerName www.example.com
ServerName hamzamohkhan.com
ServerAlias www.hamzamohkhan.com

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html

        # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
        # error, crit, alert, emerg.
        # It is also possible to configure the loglevel for particular
        # modules, e.g.
        #LogLevel info ssl:warn

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined
<FilesMatch .php$>
 # 2.4.10+ can proxy to unix socket
SetHandler 'proxy:unix:/var/run/php/php7.4-fpm.sock|fcgi://localhost'
                        </FilesMatch>

        # For most configuration files from conf-available/, which are
        # enabled or disabled at a global level, it is possible to
        # include a line for only one particular virtual host. For example the
        # following line enables the CGI configuration for this host only
        # after it has been globally disabled with "a2disconf".
        #Include conf-available/serve-cgi-bin.conf
</VirtualHost>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

Virtual host inside sites-available

MARIADB:

8. Now your wordpress website needs a database where it can store information. So we need mariadb/mysql to store data. We need to create database that will access. The following code shows how to configure and install database.

```
read -r -p "Do you want to install database [y/N]"
response_db
if [[ "$response_db" =~ ^([yY][eE][sS]|[yY])$ ]]
then
```

```
            sudo apt-get install mariadb-server &&
    sudo apt-get install mariadb-client -y
```

```
    else
            mysql -V
            #mariadb configure
            sudo systemctl enable mariadb
            sudo systemctl status mariadbs
    fi
```

WORDPRESS SETUP:

9. We will now setup wordpress site by typing the following code

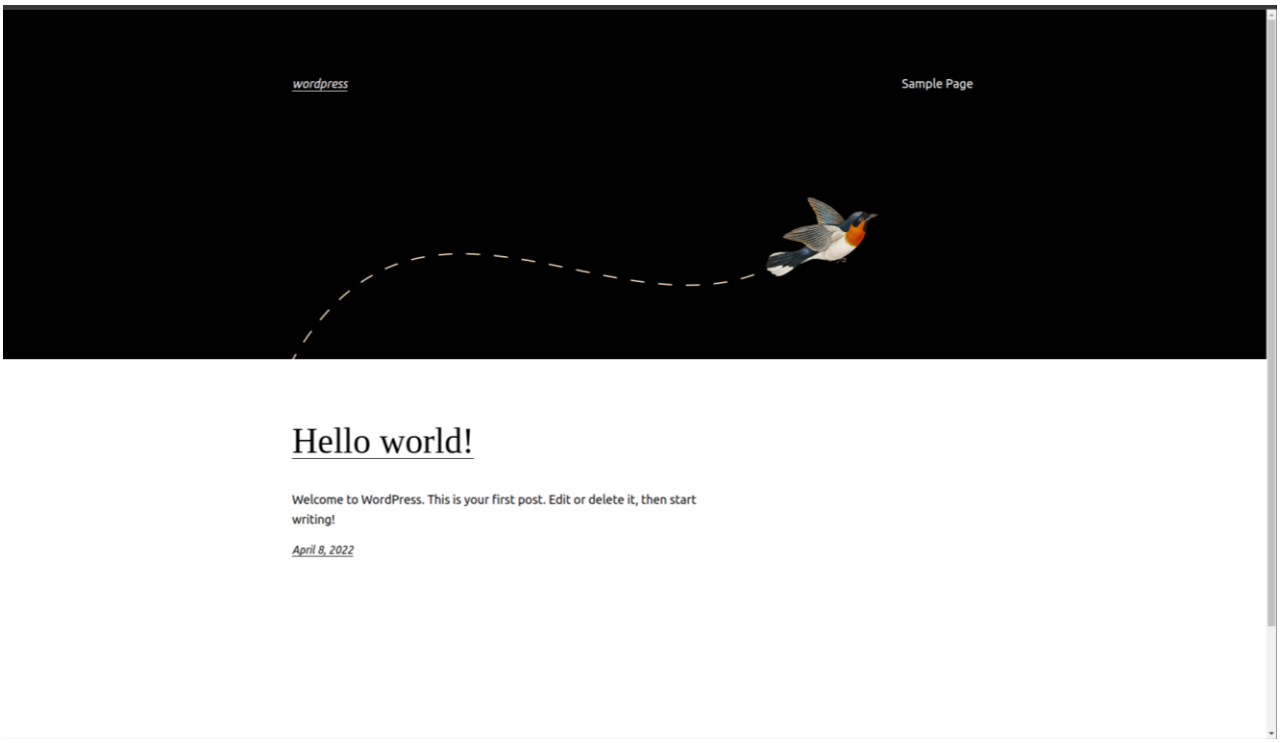
```
echo "=============Installing Wordpress============"
          #install wordpress tar
          sudo curl -O https://wordpress.org/latest.tar.gz

          #extract wp
          sudo tar -xzvf latest.tar.gz

          #enter wp folder
          cd wordpress

          #copy all to new folder
#sudo cp -r * /var/www/hamza/ #sending to the desired directory
          sudo cp -r * /var/www/html/ #send it to desired directory

          #wp-cli install
          sudo curl -O https://raw.githubusercontent.com/wp-
cli/builds/gh-pages/phar/wp-cli.phar
          sudo chmod +x wp-cli.phar
          sudo mv wp-cli.phar /usr/local/bin/wp   #wp-cli. need it

          #check wp-cli
          wp --info
```

Hamza Muhammad Khan

```
        #change ownership of folder
        sudo chown -R www-data:www-data /var/www/html

        #make a new directory
        #sudo mkdir /var/www/hamza

        #Giving database variables for wordpress
        read -p "Enter wp password :  " wp_pass
        read -p "enter wp_user : " wp_user
        read -p "Enter wp_db : " wp_db

        #database
        sudo mysql -e "create database $wp_db;"
        sudo mysql -e "create user '$wp_user'@'localhost'
identified by '$wp_pass';"
        sudo mysql -e "grant all privileges on $wp_db.* to
'$wp_user'@'localhost';"
        sudo mysql -e "flush privileges;"


        #enter hamza folder
        echo "Enter wp folder"
        cd /var/www/html/

        #now edit/create config
        sudo wp config create --dbname=$wp_db --dbuser=$wp_user -
-dbpass=$wp_pass --dbhost='localhost' --allow-root

        #db create
        sudo wp db create --allow-root
        #wp core install for not showing wp installation page
        sudo wp core install --url='hamzamkhan.com' --
title='wordpress' --admin_user='hamza' --admin_password='admin' --
admin_email='hamza.khan@cloudways.com' --allow-root
```

```
        #Plugin breeze
        #Don't activate it for now
        sudo wp plugin install breeze --activate --allow-root

        #change permissions
        sudo chmod 777 /var/www/html/wp-content/advanced-
cache.php #originial permission 644
        sudo chmod 777 var/www/html/wp-content/breeze-config/
        sudo chmod 777 /var/www/html/wp-content/breeze-config.php
```

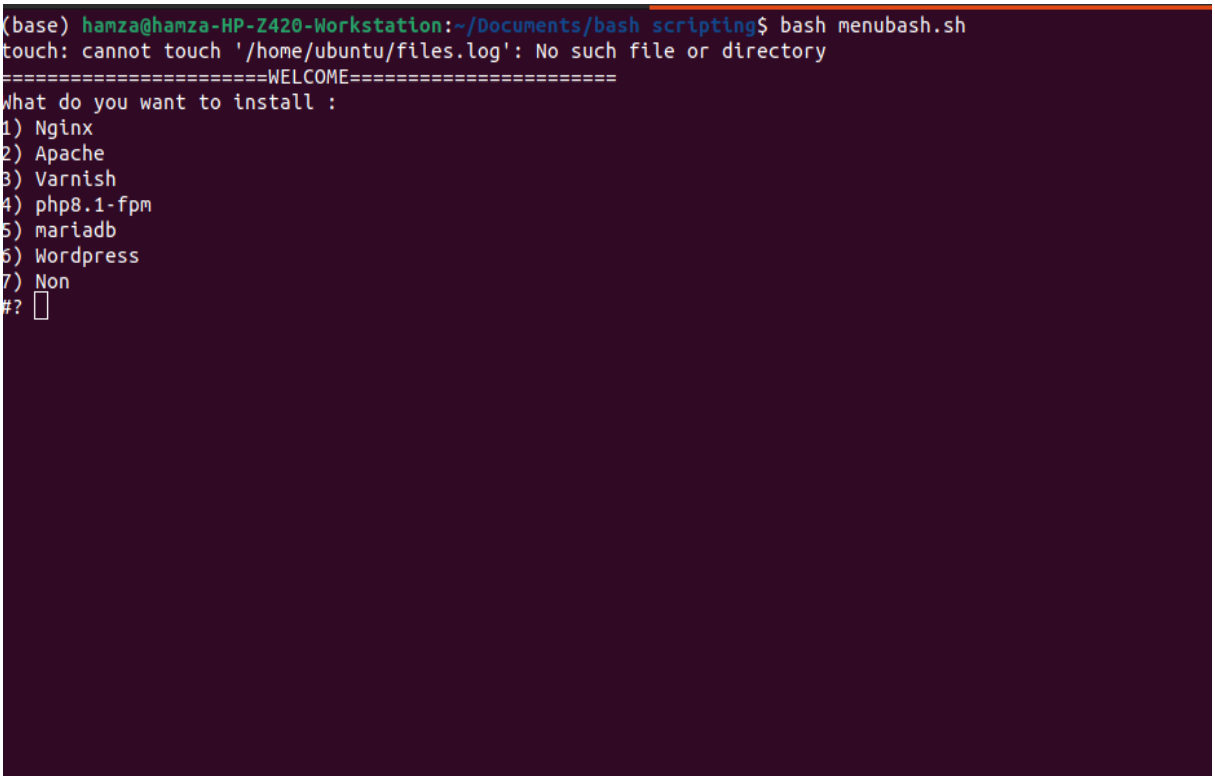
10. After completing your wordpress site will be installed and you will be prompted to this page.



1a. ANOTHER WAY OF REVERSE PROXY

I also made another code for making a reverse proxy, this one is like a menu selection.

Hamza Muhammad Khan



Code is given below, in this code we have used array and a select statement

```
#!/bin/bash

sudo touch /home/ubuntu/files.log
logpath="/home/ubuntu/files.log"
sudo chmod +x files.log
now=$(date)

echo "======================WELCOME======================"


echo 'What do you want to install :'
ins=("Nginx" "Apache" "Varnish" "php8.1-fpm" "mariadb" "Wordpress"
"Non")
select inst in "${ins[@]}"; do
    case $inst in
       "Nginx")
```

Hamza Muhammad Khan

```
            echo "$(date)==========Installing Nginx============="
>> files.log
             sudo apt-get install nginx -y
            read -p "Server Name : " servername
            sudo sed -i '12i upstream backend{\n server
127.0.0.1:8080 fail_timeout=5s weight=5; \n server 127.0.0.1:8081
backup; \n #upstream \n}' /etc/nginx/nginx.conf
         sudo sed -i '50i proxy_pass http://backend;\n
proxy_set_header Host $host; \nproxy_set_header X-Real-IP
$remote_addr;\n proxy_set_header X-Forwarded-For
     $proxy_add_x_forwarded_for;\n proxy_set_header X-Forwarded-
Proto $scheme;' /etc/nginx/sites-available/default
            sudo sed -i '59i location ~ \.php$ \n { \nfastcgi_pass
unix:/var/run/php/php7.4-fpm.sock;\ninclude
fastcgi_params;\nfastcgi_index index.php;\nfastcgi_param
SCRIPT_FILENAME $document_root$fastcgi_script_name;}'
/etc/nginx/sites-available/default
         sudo sed -i '46i  server_name  $servername;'
/etc/nginx/sites-available/default

                sudo systemctl enable nginx
          sudo systemctl restart nginx
          sudo systemctl status nginx

          nginx -v
          #sudo ufw app list
          sudo ufw allow 'Nginx HTTP'
            ;;
        "Apache")
          sudo apt-get update && upgrade
               echo "===========Installing Apache============" >>
files.log
               sudo apt-get install apache2 -y
               read -p "Server name: " servernameapache
               sudo sed -i "s/80/8081/g"  /etc/apache2/ports.conf
```


```
          sudo sed -i  "s/80/8081/g" /etc/apache2/sites-
available/000-default.conf
          sudo sed -i "10i ServerName $servernameapache"
/etc/apache2/sites-available/000-default.conf
          sudo sed -i "11i ServerAlias www.$servernameapache"
/etc/apache2/sites-available/000-default.conf
          sudo sed -i "24i  <FilesMatch \.php$>\n # 2.4.10+ can
proxy to unix socket \nSetHandler 'proxy:unix:/var/run/php/php7.4-
fpm.sock|fcgi://localhost'\n            </FilesMatch>"
/etc/apache2/sites-available/000-default.conf
          sudo systemctl enable apache2.service
          sudo systemctl restart apache2.service
          #sudo ufw allow 'Apache'
            ;;
        "Varnish")
               echo "========Installing Varnish==============" >>
files.log
              sudo apt-get install varnish -y
          sudo apt install varnish -y
          #configure varnish port in dfiferent files
          sudo sed -i "s/6081/8080/g"
/lib/systemd/system/varnish.service
          sudo sed -i "s/8080/8081/g" /etc/varnish/default.vcl
          sudo sed -i '39i if (obj.hits > 0)\n {\n set resp.http.X-
Varnish-Cache = "HIT"; \n} \nelse\n {\nset resp.http.X-Varnish-
Cache = "MISS";\n}' /etc/varnish/default.vcl
          sudo systemctl enable varnish.service
          sudo systemctl restart varnish.service
            ;;
        "php8.1-fpm")
          echo "=============Installing php============" >>
files.log
              sudo apt-get install php8.1-fpm
          sudo apt install software-properties-common && sudo add-
apt-repository ppa:ondrej/php -y
```


```
          sudo apt install php8.1-fpm libapache2-mod-fcgid php8.1-
mysql php8.1-curl
          sudo systemctl enable php8.1-fpm
          sudo systemctl restart php8.1-fpm
          #sudo apt install php-curl php-gd php-mbstring php-xml
php-xmlrpc php-soap php-intl php-zip
          sudo a2enmod proxy_fcgi setenvif
          sudo a2enconf php8.1-fpm

          #restart apache
          sudo systemctl restart apache2
            ;;
        "mariadb")
              echo "=============Installing Mariabdb========" >>
files.log
              sudo apt-get install mariadb-server && sudo apt-get
install mariadb-client -y
               mysql -V
          #mariadb configure
          sudo systemctl enable mariadb
          sudo systemctl status mariadb
          ;;
     "Wordpress")
          echo "=============Installing Wordpress===========" >>
files.log
          #install wordpress tar
          sudo curl -O https://wordpress.org/latest.tar.gz

          #extract wp
          sudo tar -xzvf latest.tar.gz

          #enter wp folder
          cd wordpress

          #copy all to new folder
```

Hamza Muhammad Khan

```
#sudo cp -r * /var/www/hamza/ #sending to the desired directory
          sudo cp -r * /var/www/html/ #send it to desired directory

          #wp-cli install
          sudo curl -O https://raw.githubusercontent.com/wp-
cli/builds/gh-pages/phar/wp-cli.phar
          sudo chmod +x wp-cli.phar
          sudo mv wp-cli.phar /usr/local/bin/wp   #wp-cli. need it

          #check wp-cli
          wp --info

          #change ownership of folder
          sudo chown -R www-data:www-data /var/www/html

          #make a new directory
          #sudo mkdir /var/www/hamza

          #Giving database variables for wordpress
          read -p "Enter wp password :  " wp_pass
          read -p "enter wp_user : " wp_user
          read -p "Enter wp_db : " wp_db

          #database
          sudo mysql -e "create database $wp_db;"
          sudo mysql -e "create user '$wp_user'@'localhost'
identified by '$wp_pass';"
          sudo mysql -e "grant all privileges on $wp_db.* to
'$wp_user'@'localhost';"
          sudo mysql -e "flush privileges;"


          #enter hamza folder
          echo "Enter wp folder"
          cd /var/www/html/
```

```
        #now edit/create config
        sudo wp config create --dbname=$wp_db --dbuser=$wp_user --dbpass=$wp_pass --dbhost='localhost' --allow-root

        #db create
        sudo wp db create --allow-root
        #wp core install for not showing wp installation page
        sudo wp core install --url='hamzamokhan.com' --title='wordpress' --admin_user='hamza' --admin_password='admin' --admin_email='hamza.khan@cloudways.com' --allow-root

        #Plugin breeze
        #Don't activate it for now
        sudo wp plugin install breeze --activate --allow-root

        #change permissions
        sudo chmod 777 /var/www/html/wp-content/advanced-cache.php #originial permission 644
        sudo chmod 777 var/www/html/wp-content/breeze-config/
        sudo chmod 777 /var/www/html/wp-content/breeze-config.php
        ;;
      "Non")
          echo "User Requested exit"
          exit
          ;;
      *) echo "Invalid option $REPLY";;
    esac
done

sudo systemctl restart nginx.service apache2.service varnish.service
```

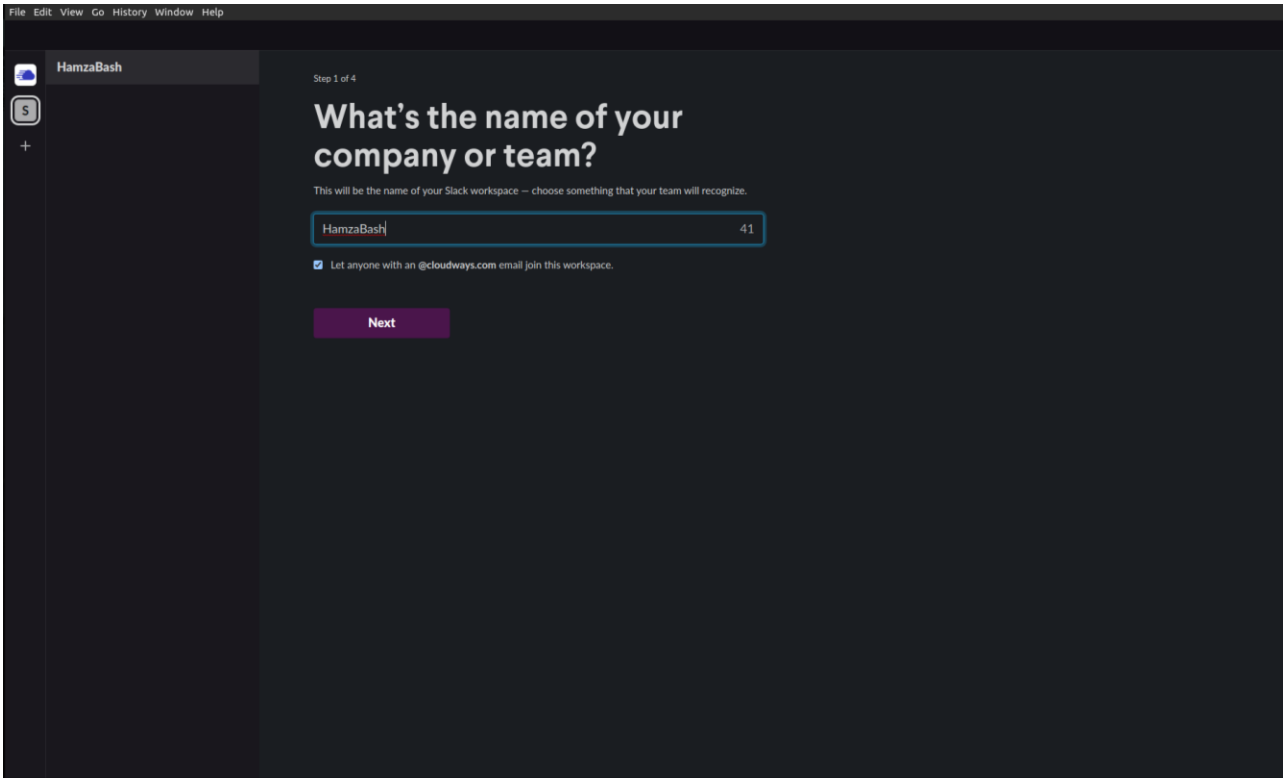
Hamza Muhammad Khan

2. USING SLACK PUSH NOTIFICATIONS

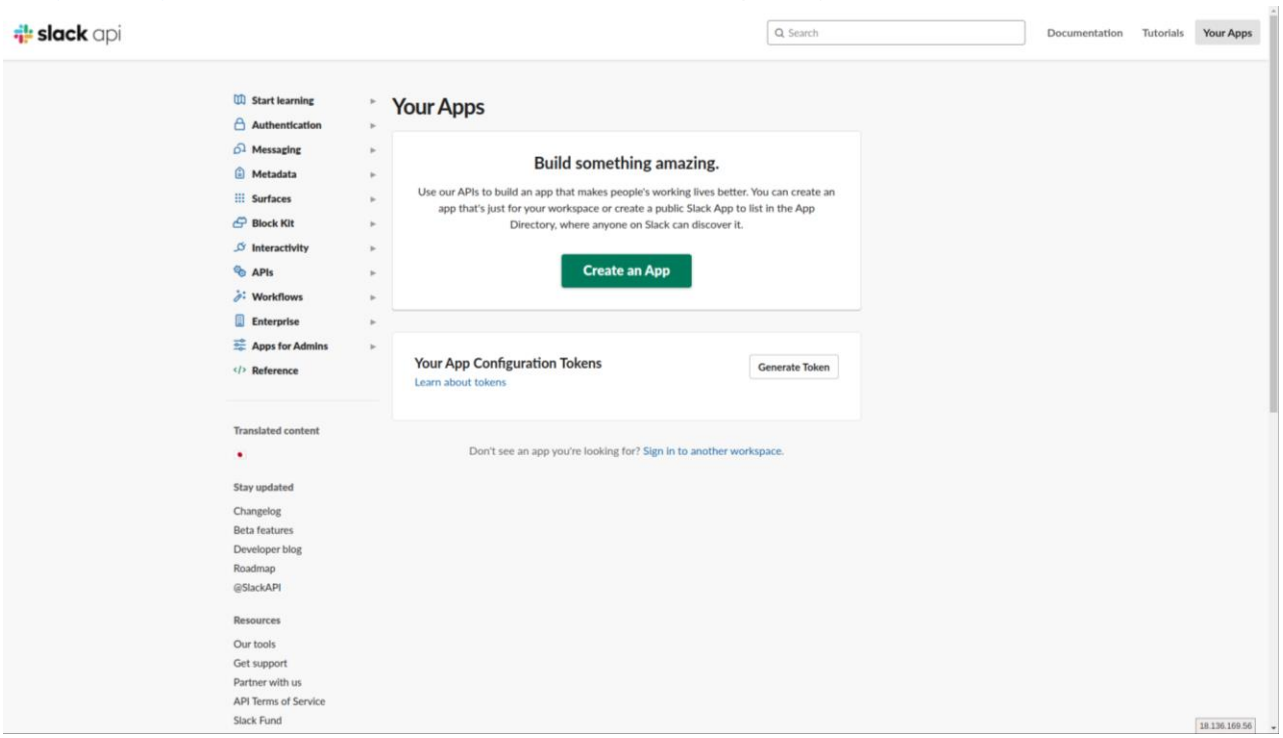
In this task we have to make such a script that every 5 minutes monitor our :-

1. CPU Usage % of your server
2. RAM Usage % of your server
3. DISK Usage % of your server
4. VARNISH HIT RATION % of your server
5. IP of your server
6. Date

Because it is crucial to monitor our system if any resources get overloaded it will crash our website. First, we generate an API of slack so that we can use it in our bash file, but before doing this create a workspace or add a new workspace where I will add a new workspace name HamzaBash

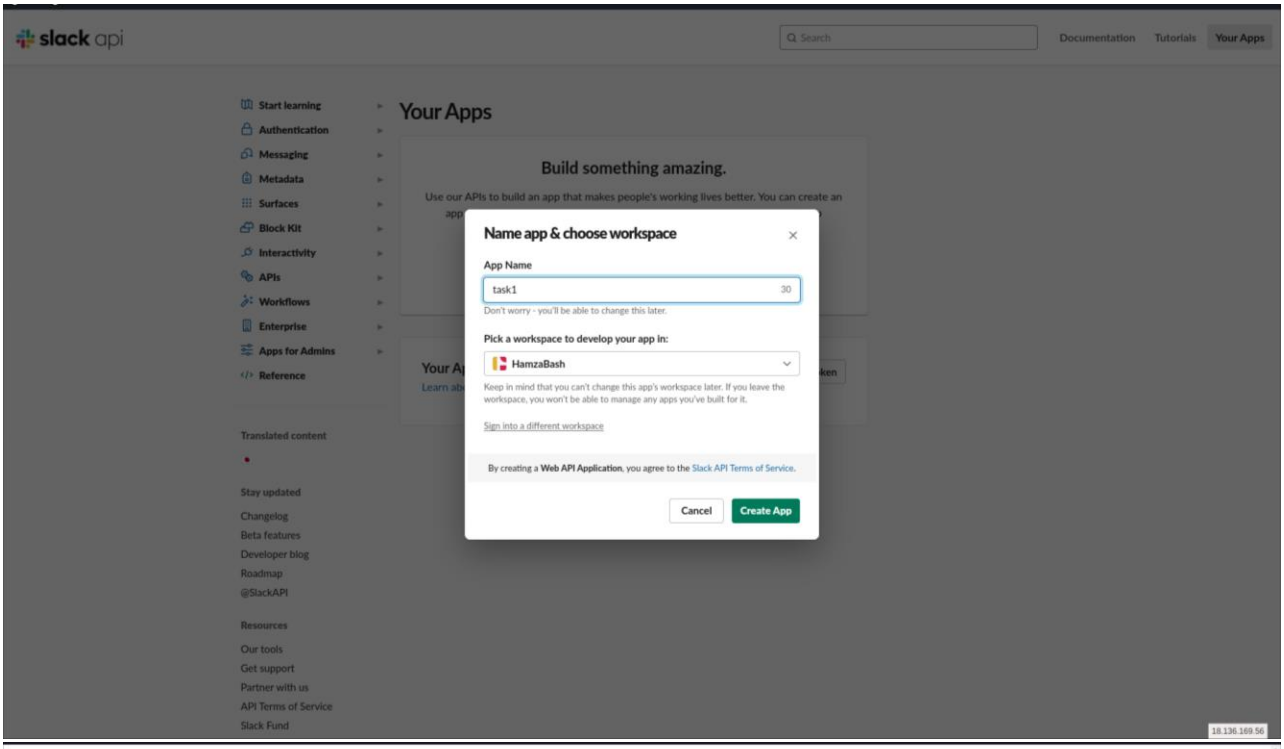


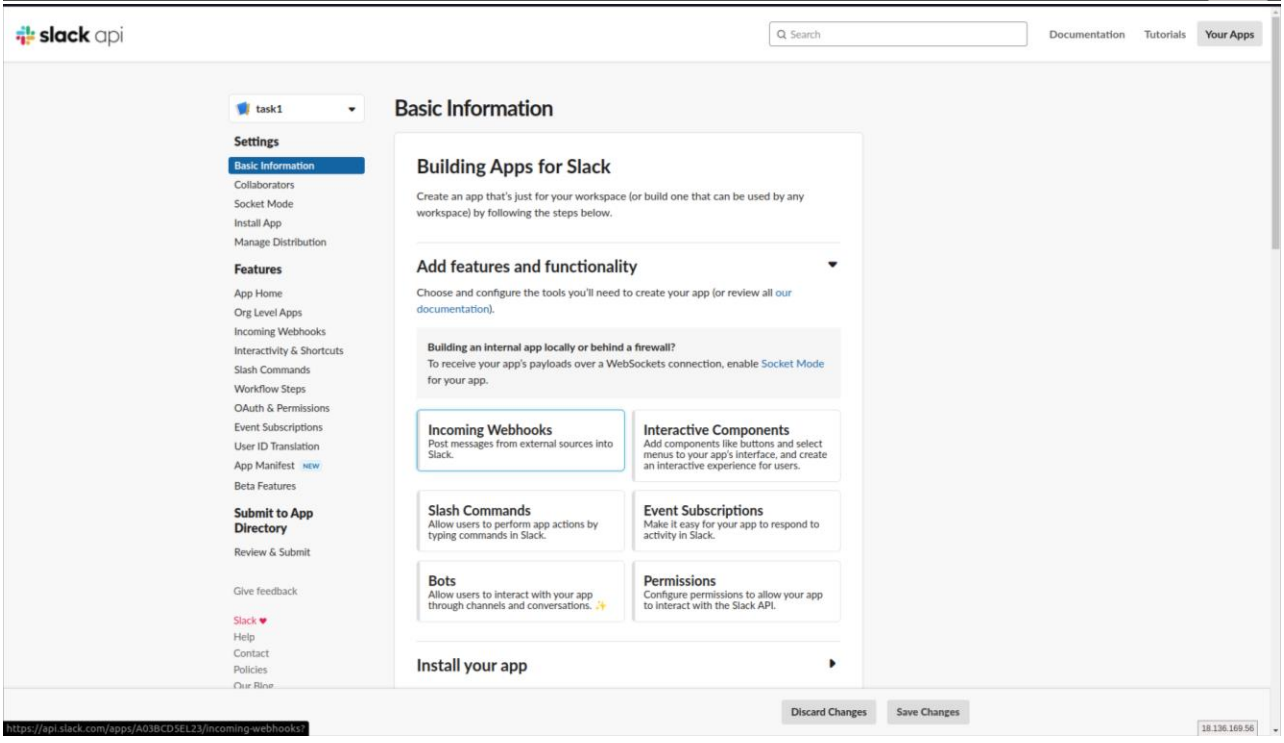
If you want to add teammates you can but we will skip that for now and make a channel called #task. Now we need to generate an API, go to https://api.slack.com/ and do the following steps



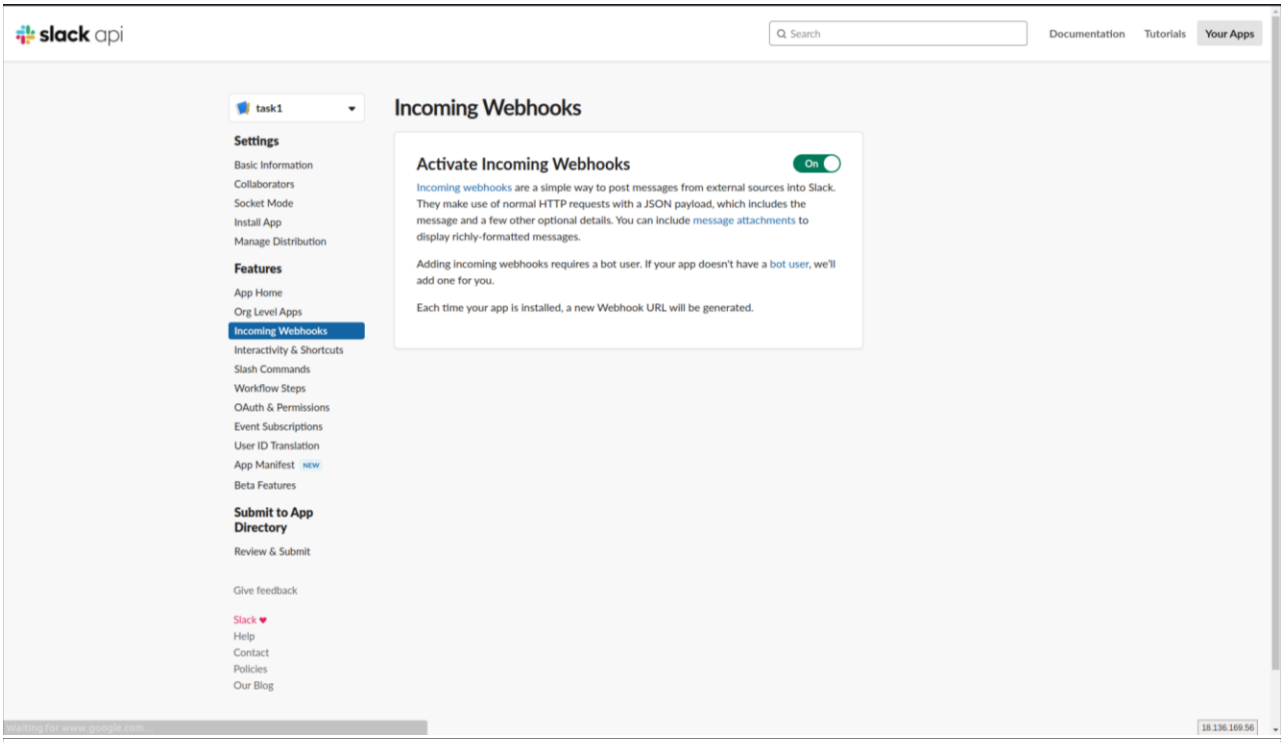
CLOUDWAYS

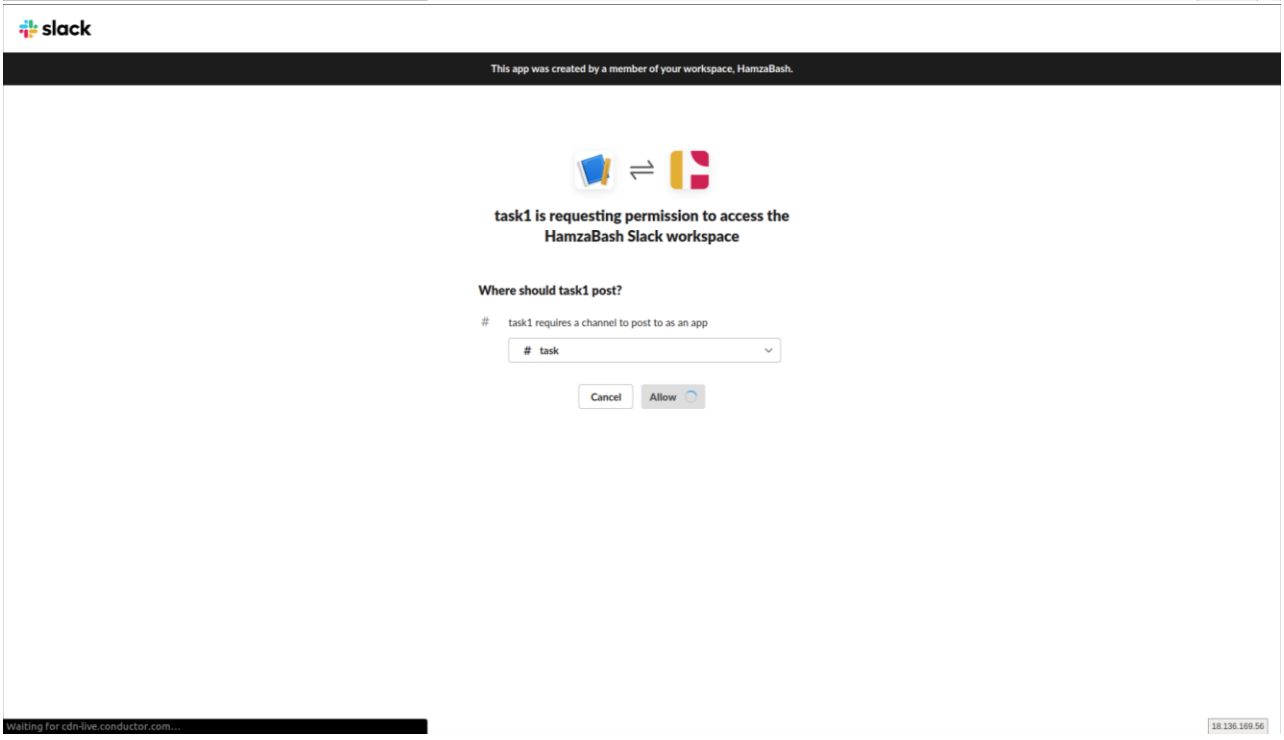
Hamza Muhammad Khan

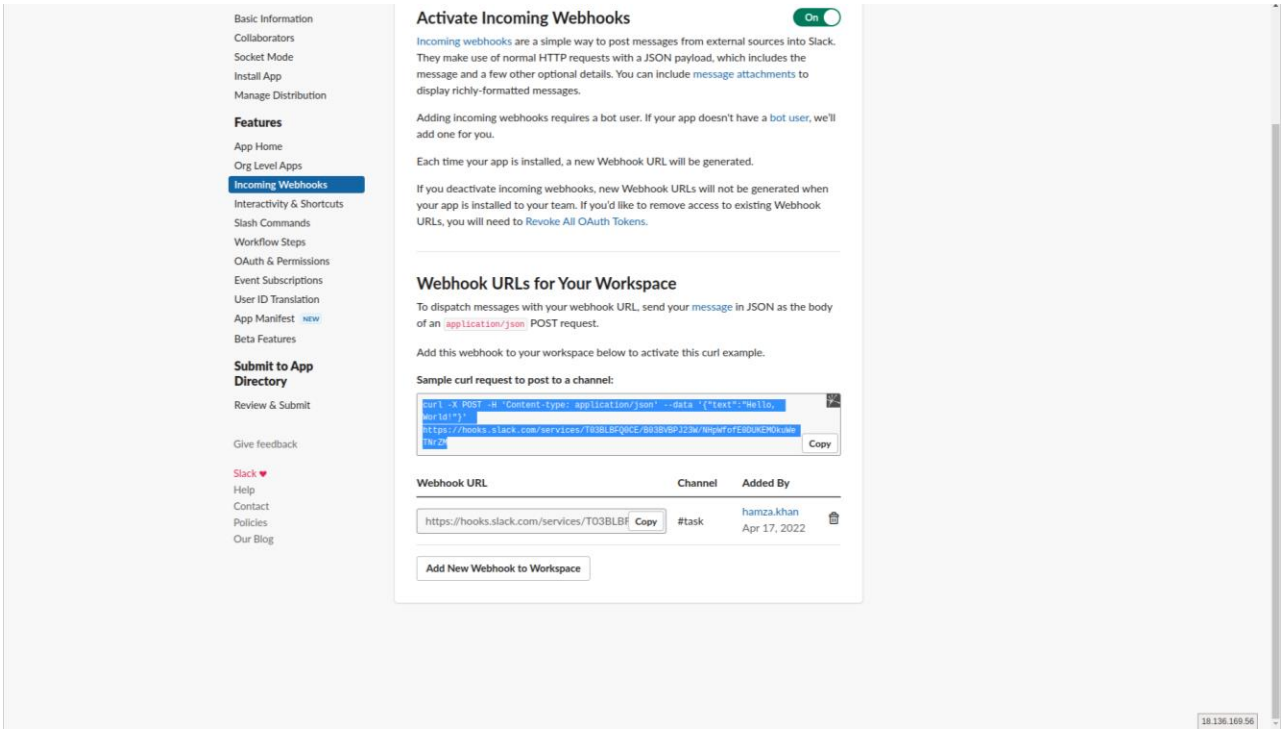




Hamza Muhammad Khan







Hamza Muhammad Khan

Now add the highlighted script in your bash file.

```
The code for the files is given below
#!/bin/bash

now=$(date)

#cpu_usage
cpu=$(top -bn2 | grep '%Cpu' | tail -1 | grep -P '(....|...)
id,'|awk '{print "CPU Usage: " 100-$8 "%"}')
#cpu_idle=`top -b -n 1 | grep Cpu | awk '{print 100-$8 "%"}'|cut -f
1 -d "."`
#cpu_use=`expr 100 - $cpu_idle`
#echo  "cpu utilization: $cpu_use"
#curl -X POST -H 'Content-type: application/json' --data
'{"text":"'"$cpu"'"}'


#mem_usage
```

Hamza Muhammad Khan

```
#mem_free=`free -m | grep "Mem" | awk '{print $4+$6}'`
mem_free=$(free -t | awk 'NR == 2 {printf("Current Memory
Utilization is : %.2f%\n"), $3/$2*100}')
#echo "memory space remaining : $mem_free"
echo "$mem_free"

#disk_usage
disk=$(df -h | awk '$NF=="/"{printf "Disk Usage: %d/%dGB (%s)\n",
$3,$2,$5}')
echo "$disk"

#varnsih
cachehit=$(sudo varnishstat -1 | grep "cache_hit " |awk '{print
$2}') #### dont forget to start varnishd.
cachemiss=$(sudo varnishstat -1 | grep "cache_miss" |awk '{print
$2}')
totalcache=$(expr $cachehit + $cachemiss)
echo "( $cachehit / $totalcache )"  #bc -l


#ip
ip=$(curl ipinfo.io/ip/)
echo "$ip"

#slack
curl -X POST -H 'Content-type: application/json' --data
'{"text":"'"$date $cpu\n $date $mem_free\n $date $disk \n $date
$ip\n $date
$totalcache"'"}'https:/hooks.slack.com/services/T03BLBFQ0CE/B03BVBP
J23W/NHpWfofE0DUKEMOkuWeTNrZM
```

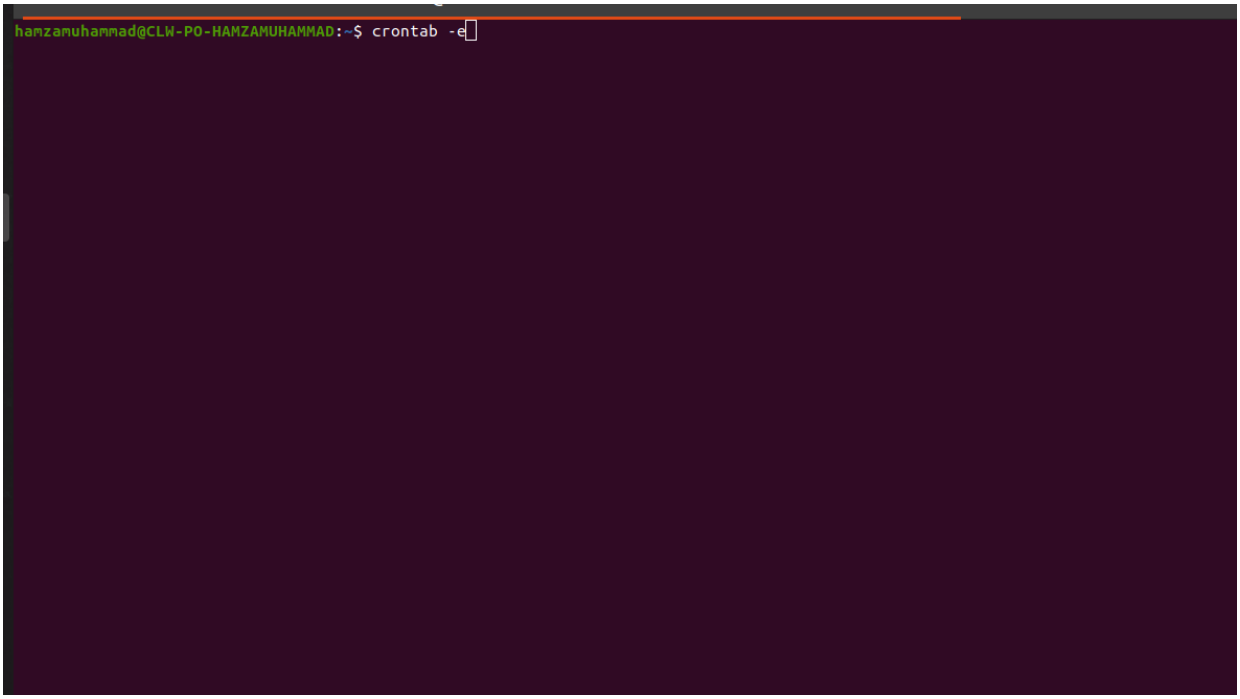
1. As you can see there is cpu usage, which is the important point in a server. So we use **top** command we can determine it's utilization.

2. Memory usage is also important as it stores all temporary data of the server. We can use **free -t** to show us free memory and divide it by 100 to show percentage.
3. Disk usage is used to store images, user logins, content of website we will be using **df -h.**
4. For showing varnish hit ratio, we can determine it by using the formula.

Varnish hit ration = Cache_Hit / (Cache_Hit + Cache_Miss)

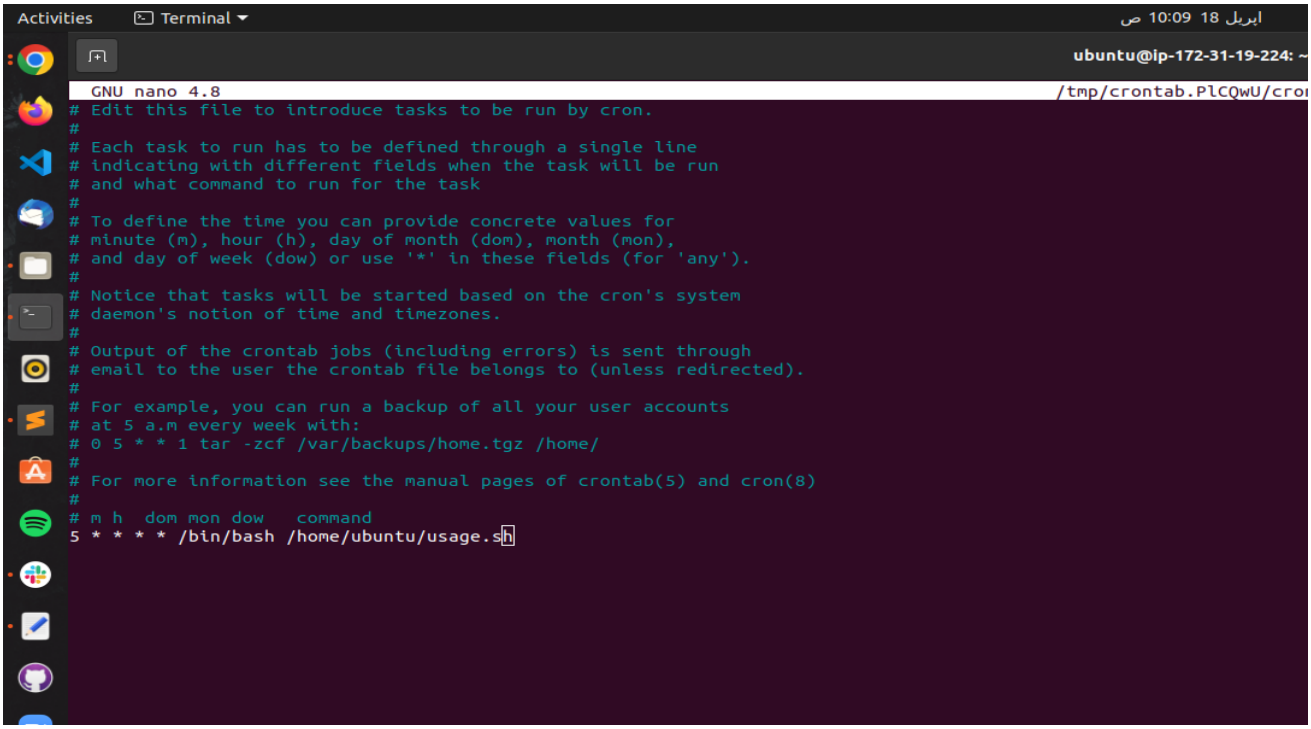
Now we will make a cronjob for this work.

1. Type crontab –e in your terminal.



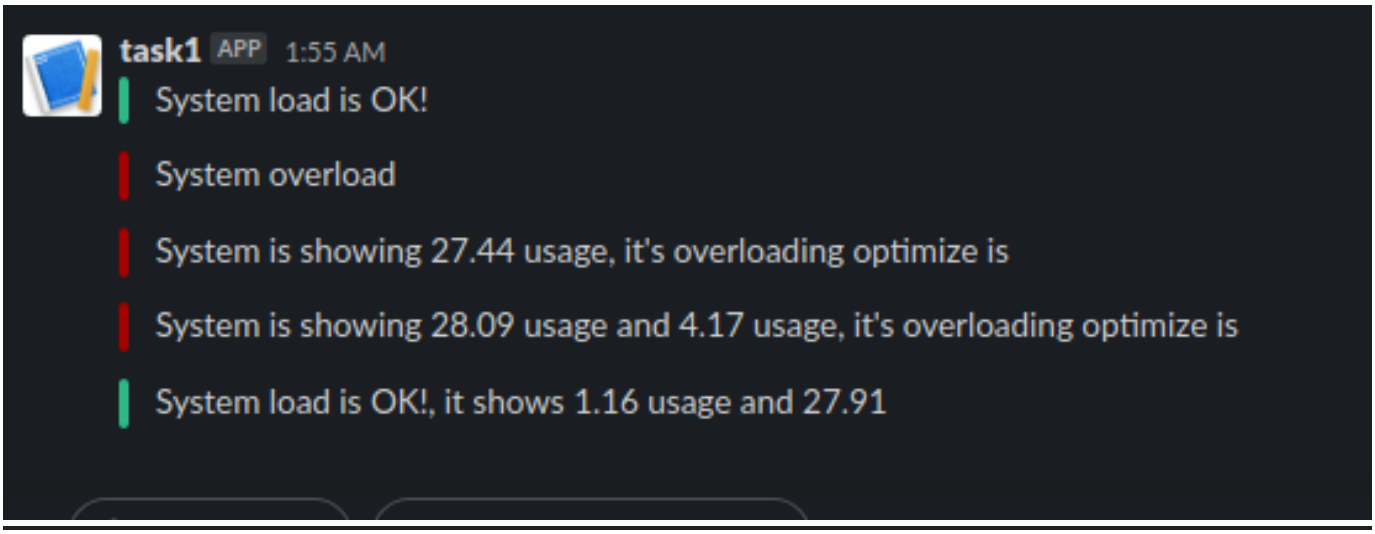
2. Now define that your script will execute every 5 minutes

Hamza Muhammad Khan



3. SHOW USAGE IN COLOURS

In this task like above we use CPU and MEMORY to show alerts in color form, we will use a condition if the usage is above 80 in memory and cpu it will generate red alert if below it will show green



As you can see the systemload alerts. The following code shows how to perform this task. Do remember to generate your own api.

Hamza Muhammad Khan

```
#!/bin/bash
SLACK_WEBHOOK_URL=https:/hooks.slack.com/services/T03BLBFQ0CE/
B03BVBPJ23W/NHpWfofE0DUKEMOkuWeTNrZM
overload() {
        color='danger'
        local message="payload={\"channel\":
\"#$SLACK_CHANNEL\",\"attachments\":[{\"text\":\"$1\",\"color\
":\"$color\"}]}"
        curl -X POST --data-urlencode "$message"
${SLACK_WEBHOOK_URL}
}
not_overload() {
        color='good'
        local message="payload={\"channel\":
\"#$SLACK_CHANNEL\",\"attachments\":[{\"text\":\"$1\",\"color\
":\"$color\"}]}"
        curl -X POST --data-urlencode "$message"
${SLACK_WEBHOOK_URL}
}
cpu=$(sar 1 5 | grep Average | awk '{print $3}')
mem=$(sar -r 5 5 | grep Average | awk '{print $5}')
echo $cpu
echo $mem
INT1=${cpu/.*}
INT2=${mem/.*}
if [[ $INT1 -gt 80 || $INT2 -gt 80 ]];
then
        overload "System is showing $mem usage and $cpu usage,
it's overloading optimize is"
else
        not_overload "System load is OK!, it shows $cpu usage
and $mem"
```

```
fi
```

For checking you can try 1 in the condition instead of 80

4. PUSH LOGS AND IGNORE GARBAGE VALUES

In this we will have to create logs like in linux when you do a task it generates logs. A log file stores information of every process even errors so they can be useful for troubleshooting also. The following code will be added in the reverse proxy script at the top

```
sudo touch /home/hamza/Documents/bash_scripting/new_files.log
logpath="/home/ubuntu/new_files.log"
sudo chmod +x new_files.log
sudo chown hamza:hamza new_files.log
now=$(date)
```

And then add the file path besides executing code like this

```
nginx )
    echo "$(date)==========Installing Nginx==========" >> /home/hamza/Documents/bash_scripting/new_files.log
     sudo apt-get install nginx -y >/dev/null 2>> /home/hamza/Documents/bash_scripting/new_files.log
```

The output will be inside the log file, as for notifications