

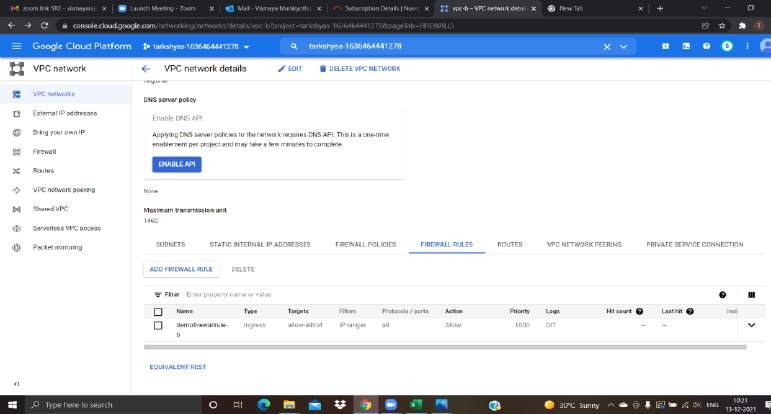
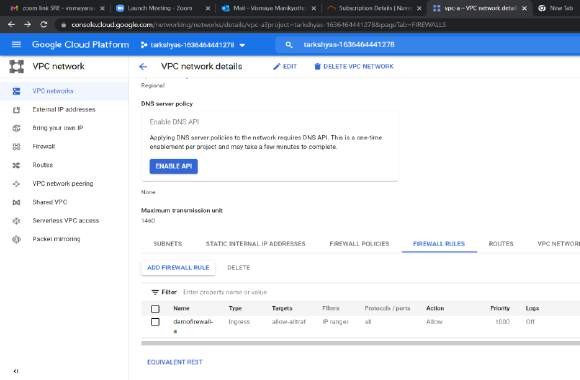
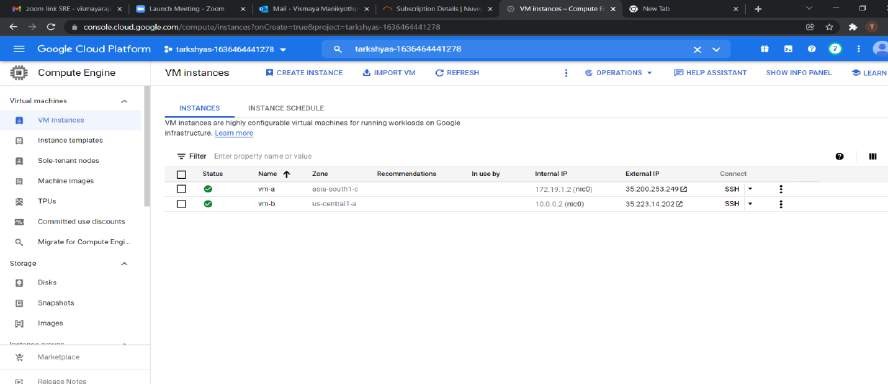
Vismaya MP Emp ID:199210

# GCP FINAL ASSESSMENT

1. Creating a VPC network to communicate between 2 VM instance present in different networks

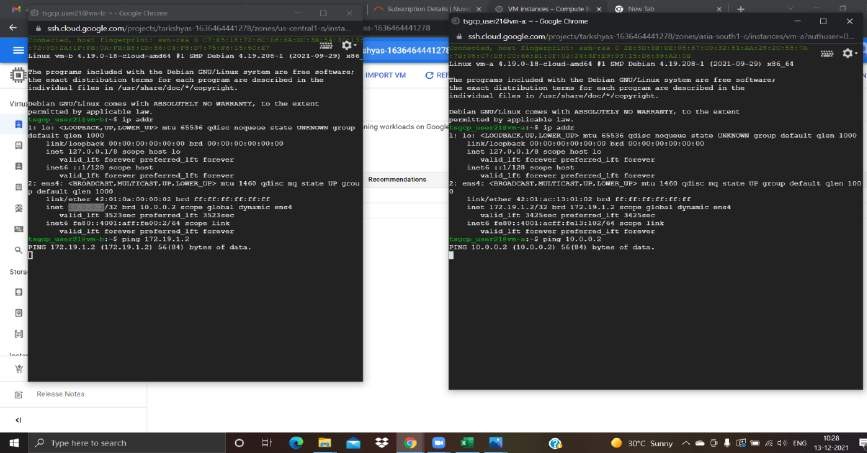
To create it first of all create 2 VPC network → vpc-a, vpc-b in 2 different region.

Now create firewall rule for each network vpc-a and vpc-b with specified target tags that allowing all traffic and respective ip addresses.



Here we have created 2 VM instances with the same region as we created the vpc networks.

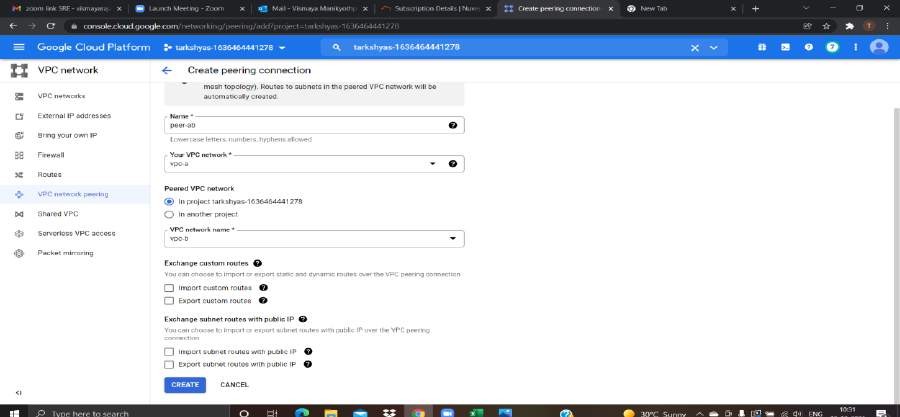
Connect the SSH keys to VM by clicking on SSH and ping both the instances to check wheather both are communicating.



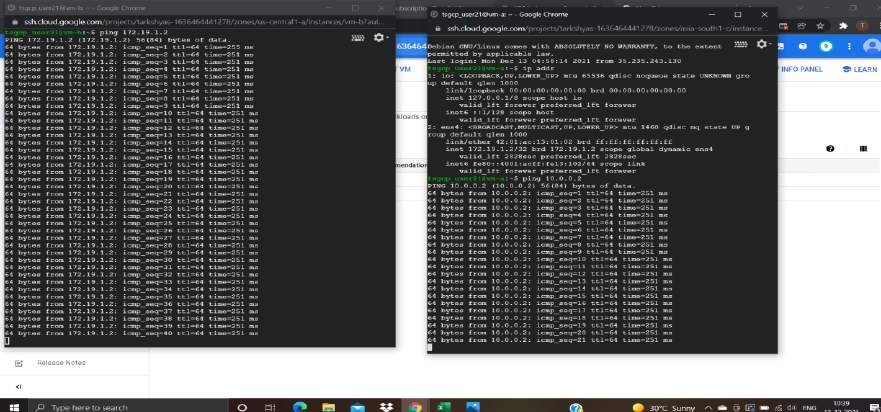
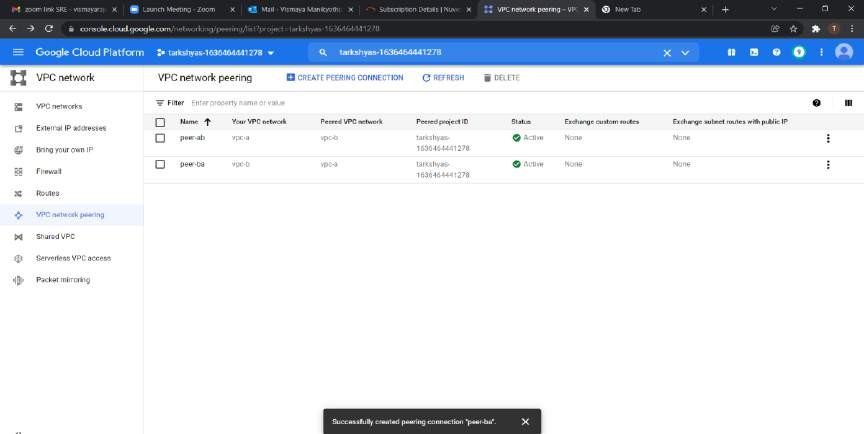
Next is to peer both connections to establish a further connection and to communicate with both the instances.

Vpc network-→vpc peering

Create a peering connection peer-ab from vpc-a to vpc-b and vice-versa



Here we have successfully established vpc-peering between 2 vm instances,and now they are able to communicate eachother.



1. Creating a cloud function to host a http function

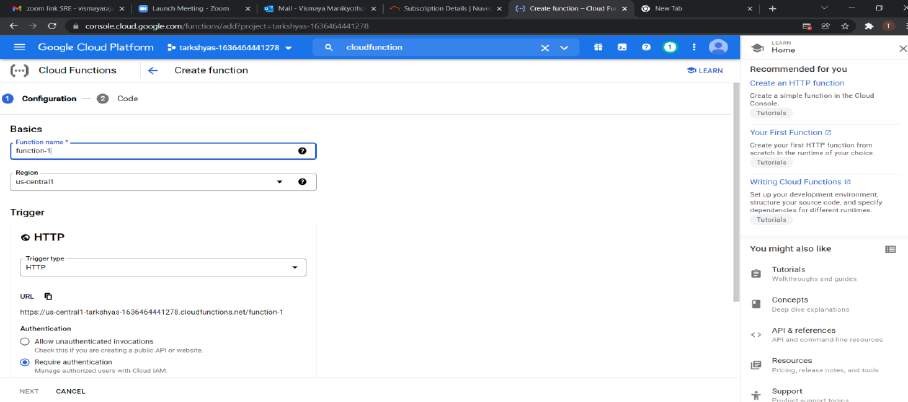
To create a cloud function go to the cloud function and give the details including function name,region,trigger type that we want to host the function.

Here we are hosting a HTTP function,so select the trigger type as HTTP

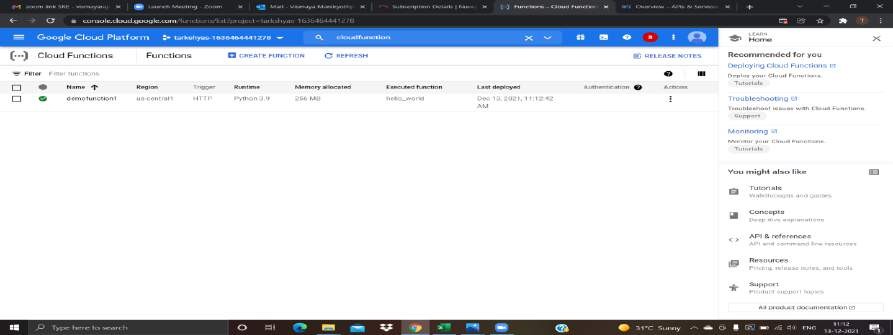
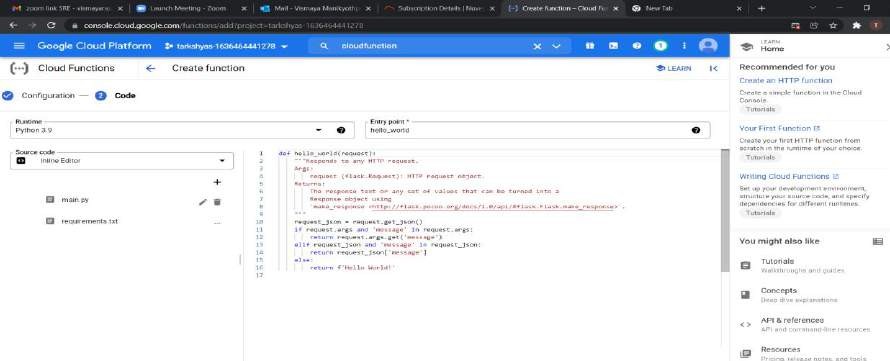
**nnnn**

**nnnnnnnnnnnmmmmmmmmmmmmmmmmm mmmmmmmmmmmmmmmmmmmmmmmm**

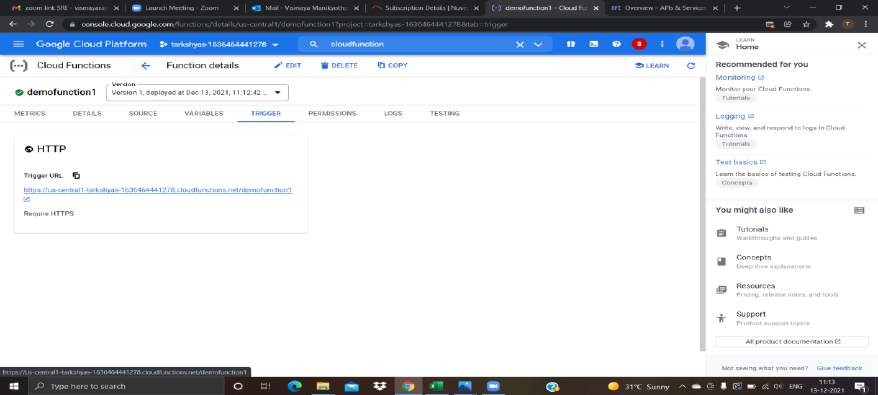
**mmmmmmmmmmmmmmmmmmmmmmmm**



Select python 3.9 as runtime and entry point as hello-world. Deploy -→ and then our function is deployed



After deployment click on the cloud function and click on the http trigger to see the resulted function that we deployed.



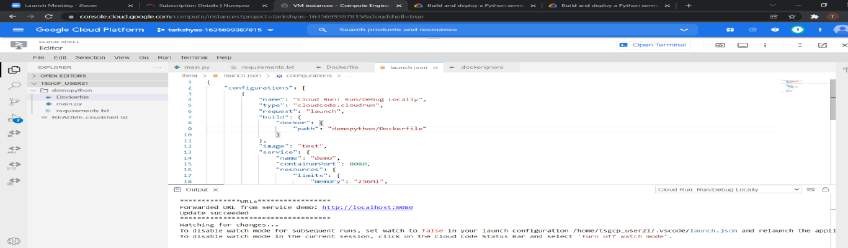
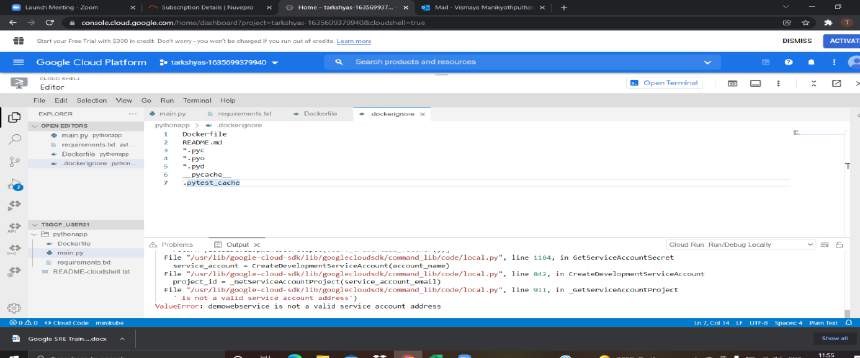
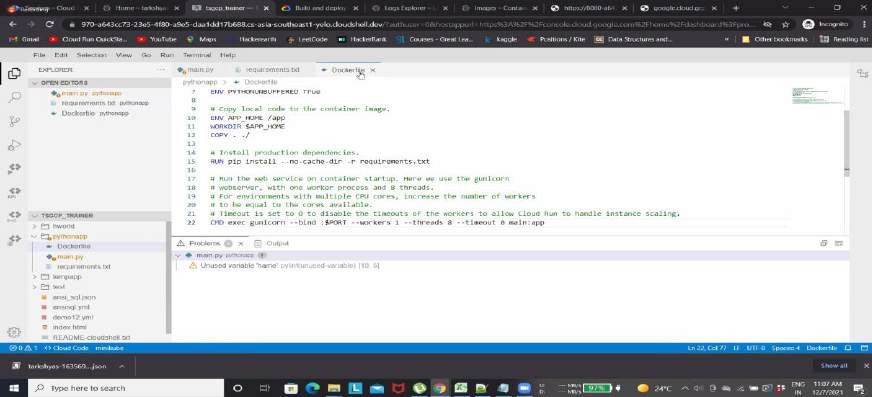
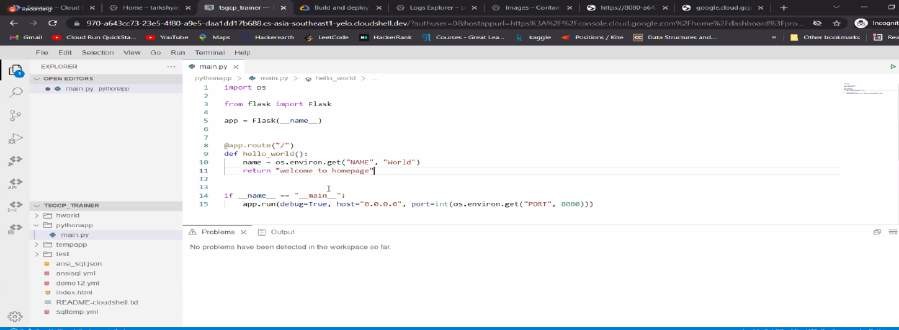
#### Cloud run



To deploy a python app in cloud run ,→ activate cloudshell in gcp-→

Open editor-→ create a folder -→create files inside that folder where codes are inserted for the deployment.

Main.py, requirements.txt,Dockerfile and .dockerignore files are uploaded inside the folder



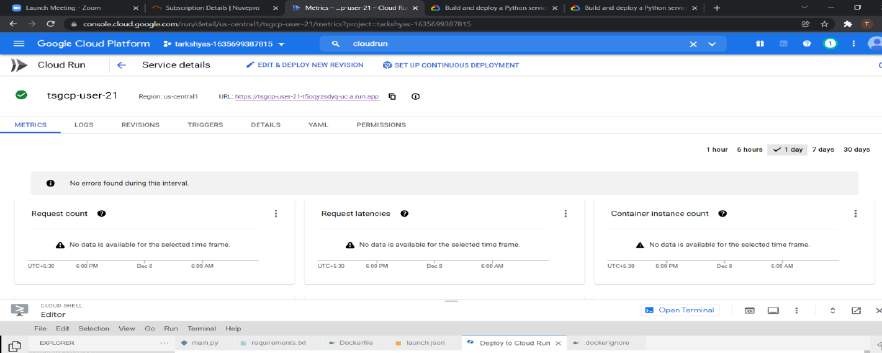
By running the code we would get an output as in the above pic,a link would appear for preview purposes. Now, by deploying it on cloud run --→



Now open cloudrun and we can see it will be deployed there



Open the deployed function and click on the https URL that we got as the output.

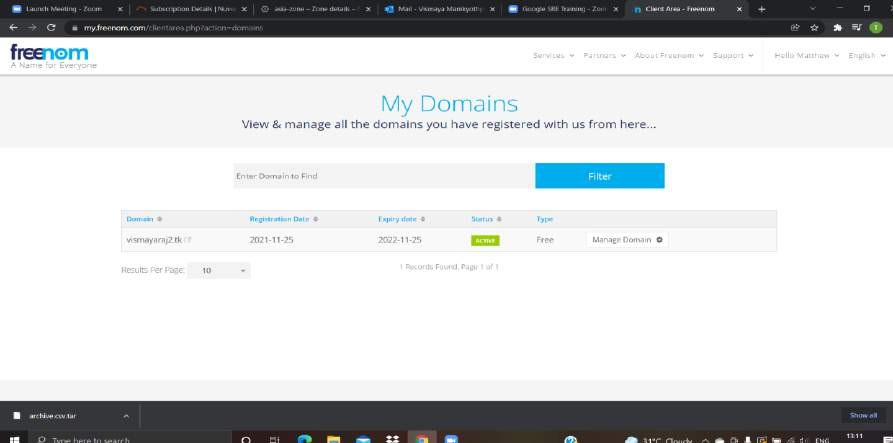


1. **cloud DNS**

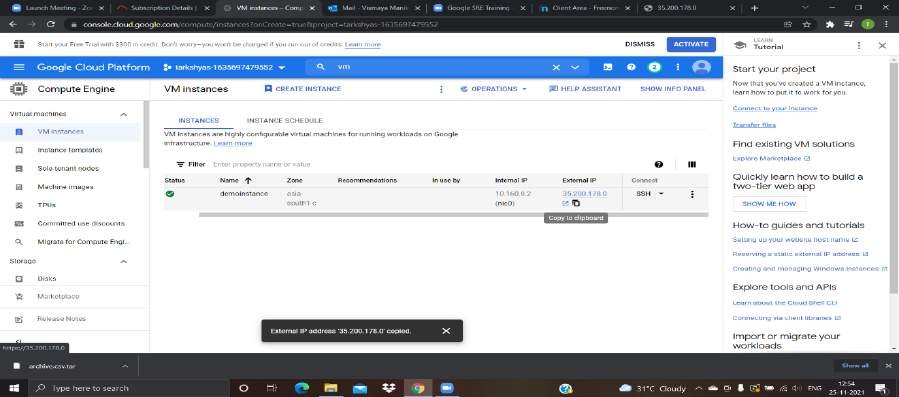


### To deploy a webpage in cloud DNS first we have to buy a domain for hosting the webpage.

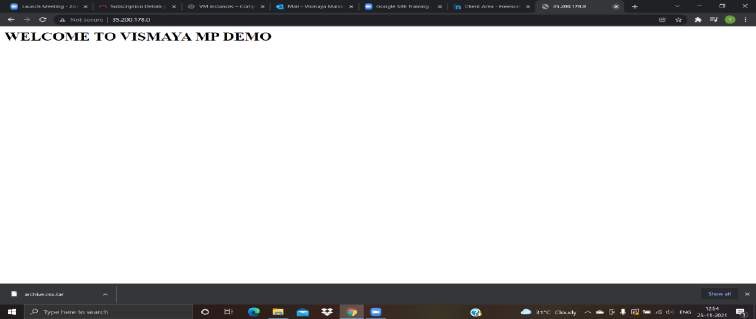
The freenom website provides us free domain, so from there buy a domain with our giving our desirable name. I have bought vismayaraj2.tk domain.



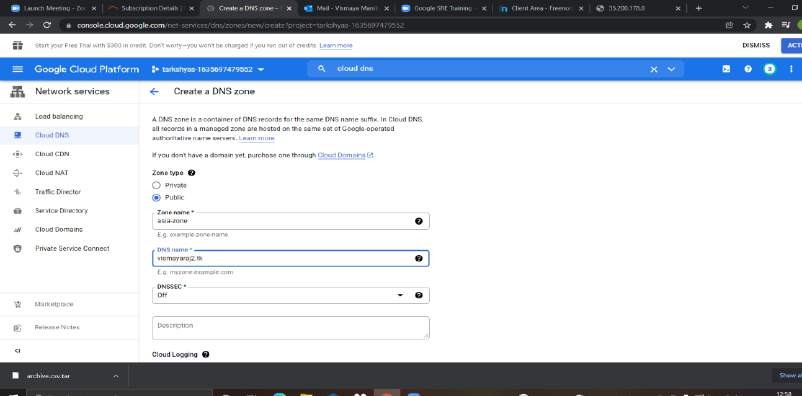
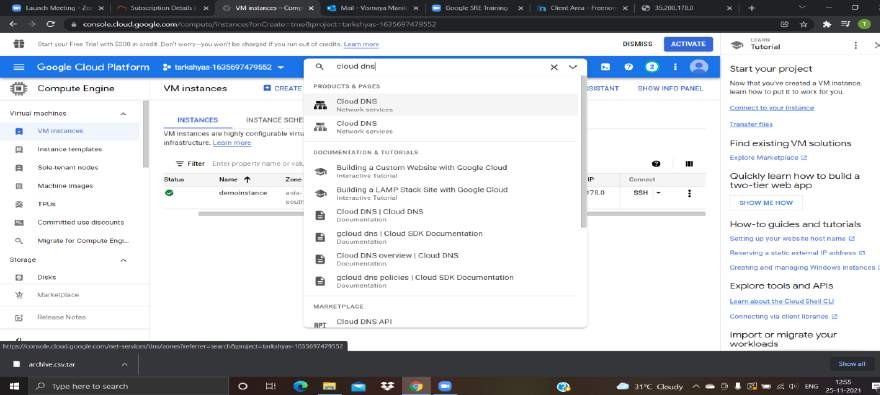
Create a vm instances that direct to the web-page by html code by using external ip



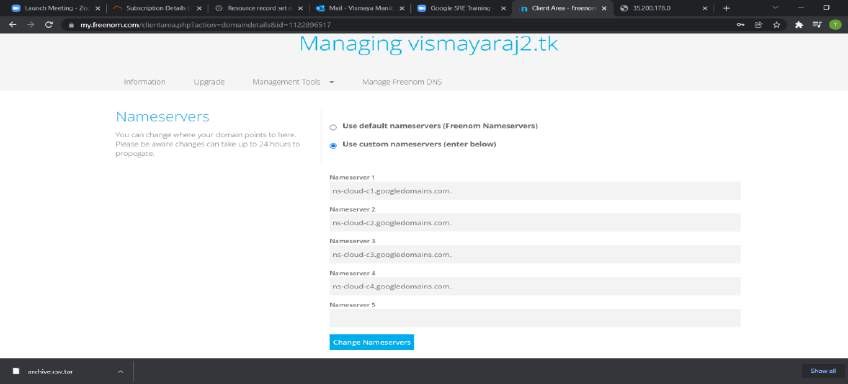
By clicking external-ip we would get sample output.



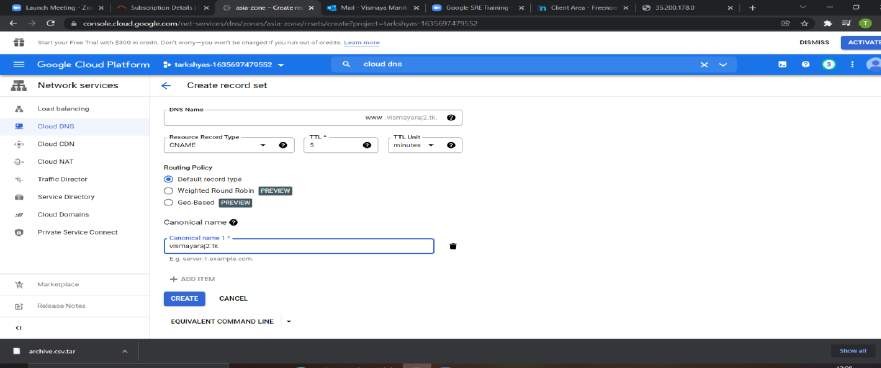
Now we have to deploy to our domain,for that open cloud dns and fill all the essential details such as zone name,DNS name etc



Now go back to our domain,in there go to the management tools and there copy&paste all the cloud google domains nameservers from cloud DNS.



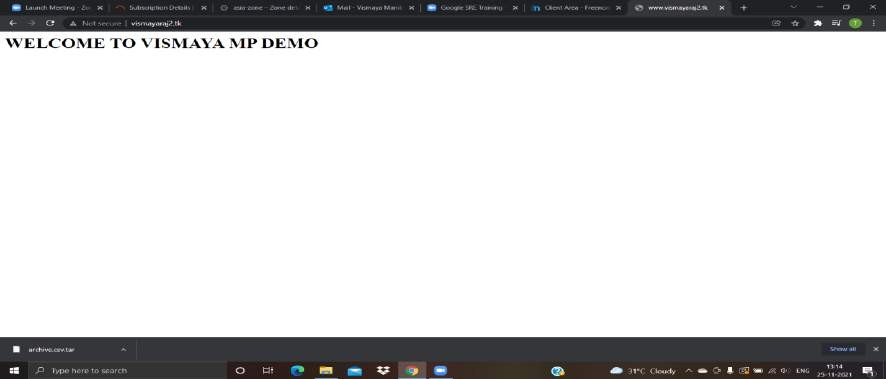
After that create a record set in cloud DNS by giving DNS name, reserve type and cananonical name.



Record set has been created and domain has been obtained in asia-zone

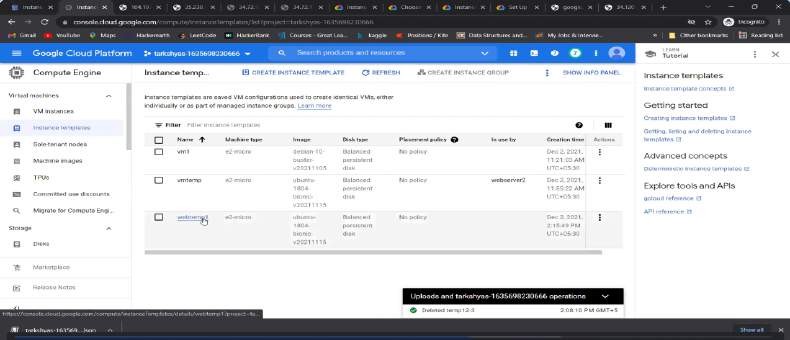
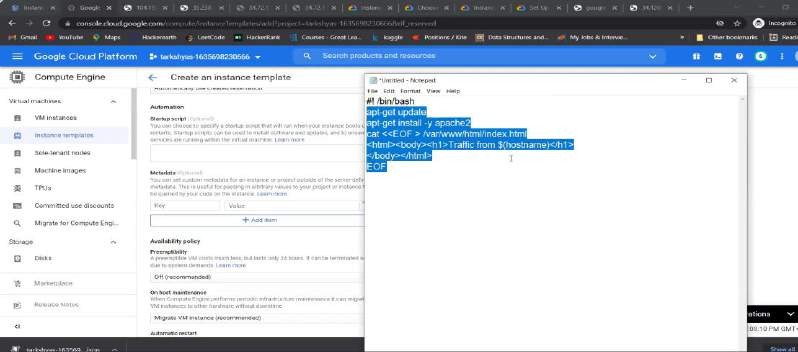


Now go to google ans search for vismayaraj2.tk and our webpage would be displayed



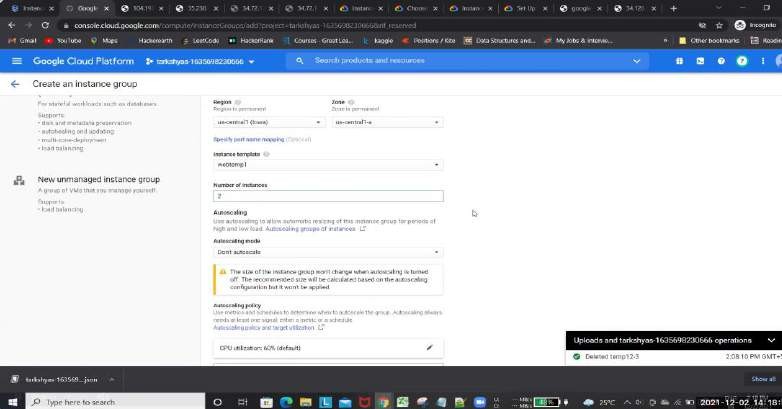
## Load balancing

Create an instance template with given name as webtemp1, and with a proper region, and a startup script. By giving all necessary informations click on create and our instance template has been created.

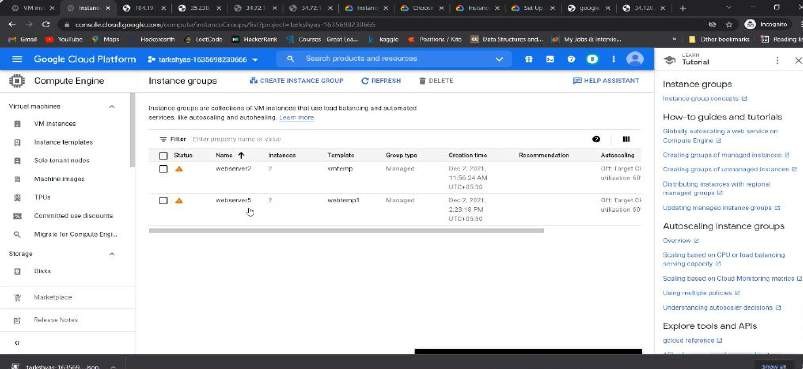
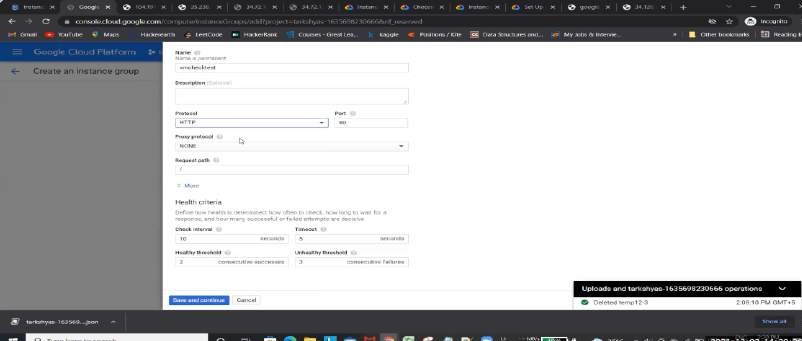


Create an instance group(webserver5) with no of instances 2

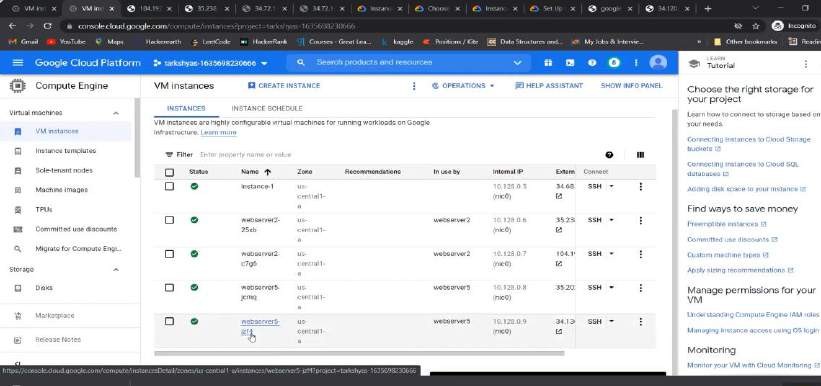
Now create an instance



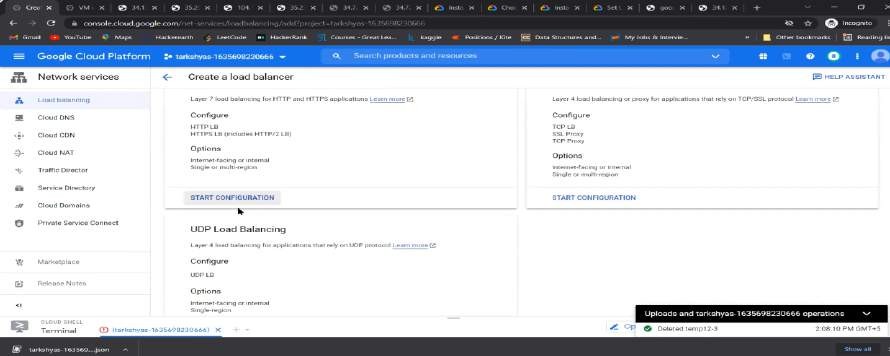
There is a healthcheck,we have to create a new healthcheck by giving name,protocol and healthcriteria-→save and continue--→ after that click on create



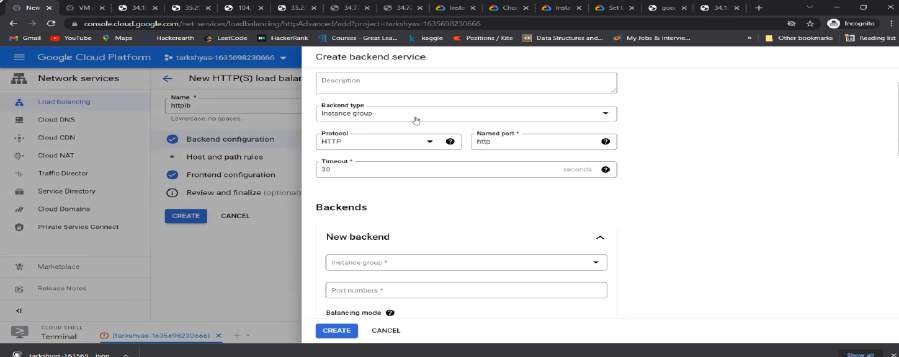
2 VM instances has been created



Go to loadbalancer→HTTP→Start configuration



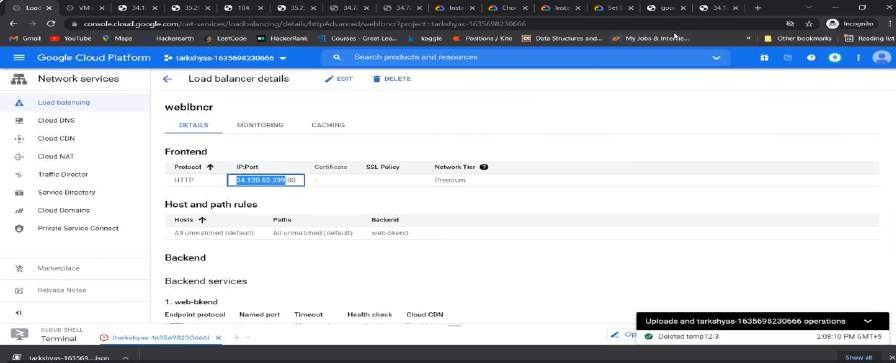
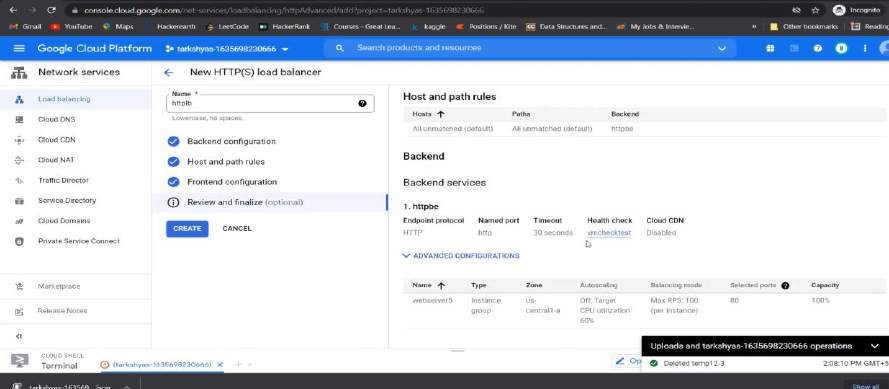
Create backened service



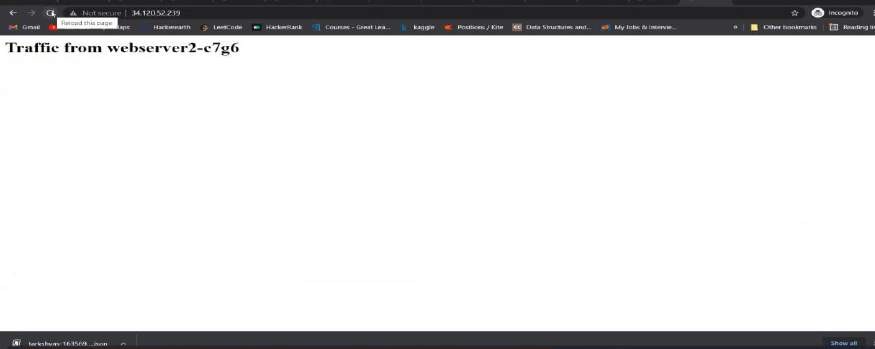
Host&path and then complete the frontened configuration



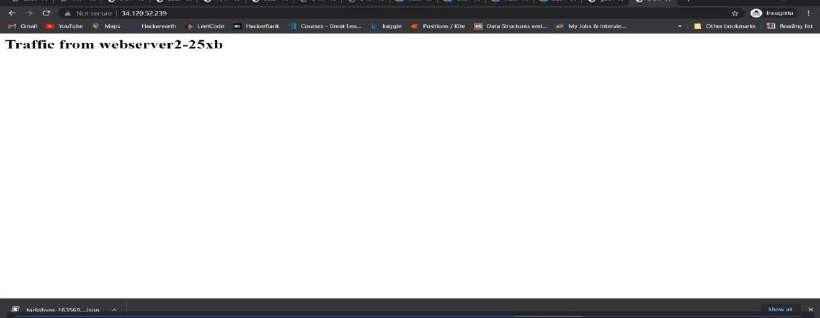
Then review and finalize the loadbalancer and then click on create



Copy the ip port address after webserver become healthy and check on the output



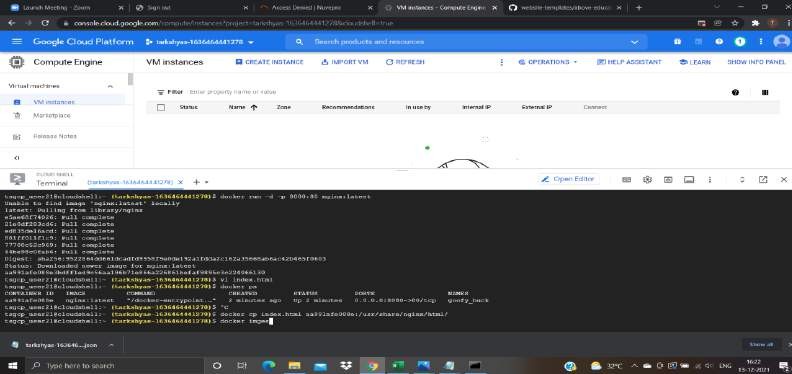
Now reload the page again and see another web-page



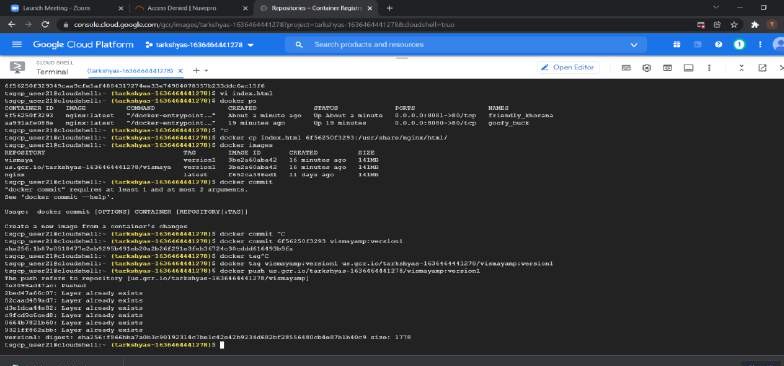
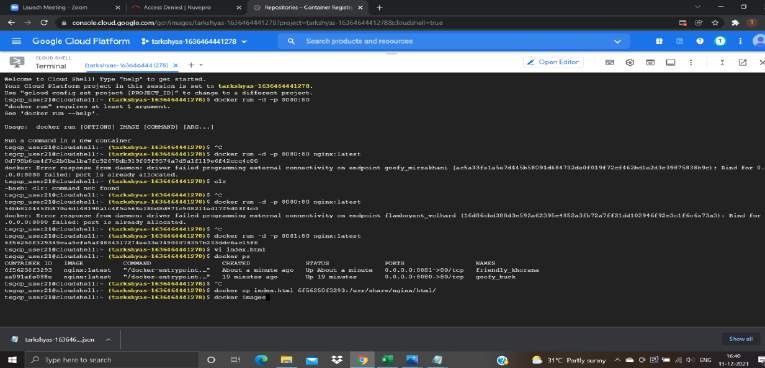
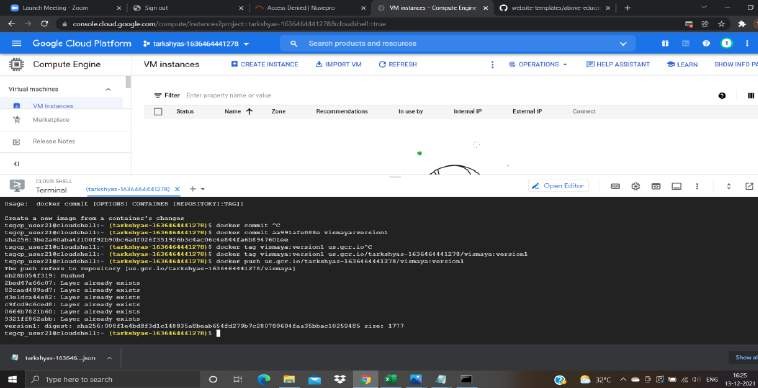
## Google Kubernetes Engine

### We can deploy a webpage using GKE with gcloud

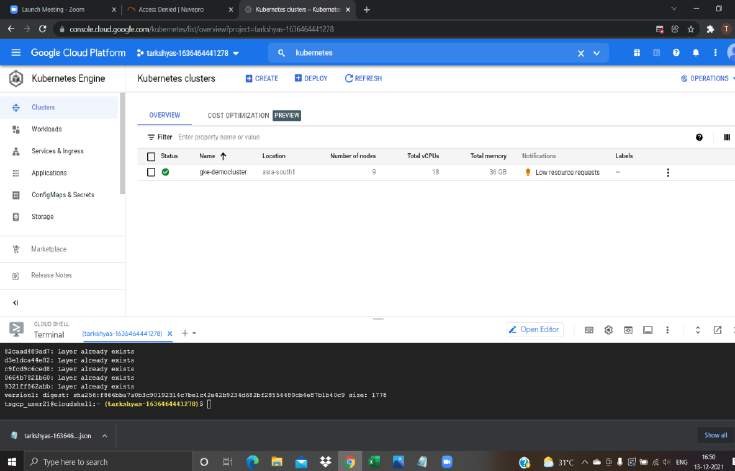
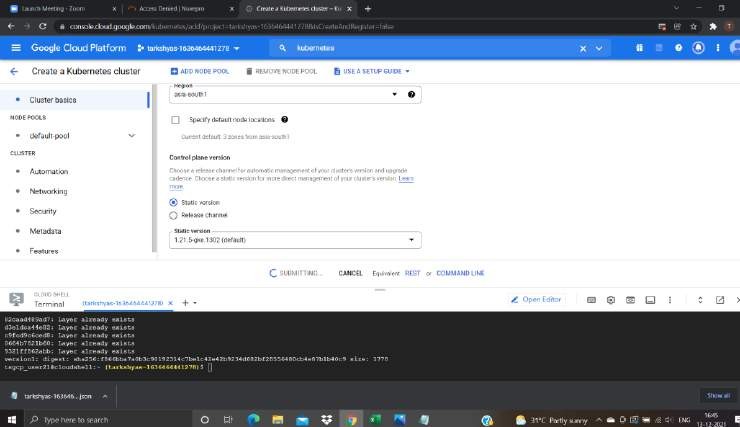
First of all activate cloudshell→create a docker container and image in the cloud repository



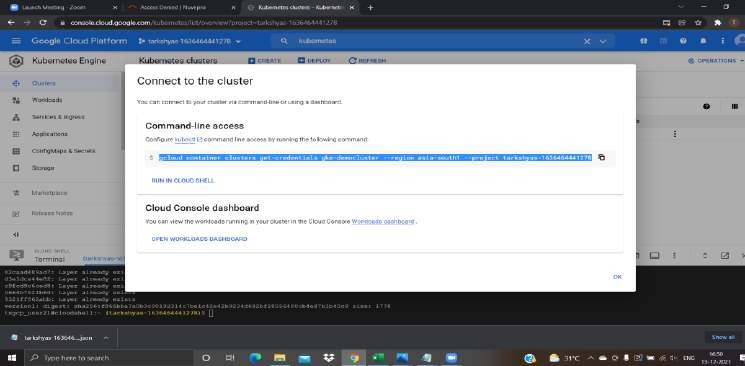
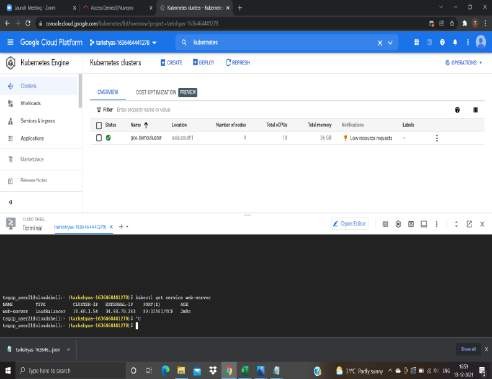
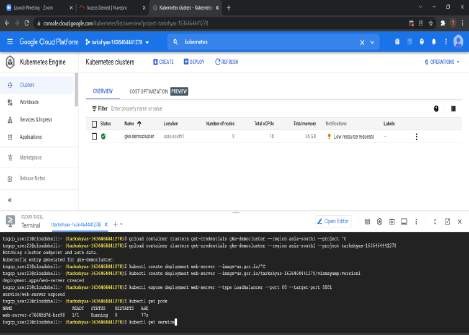
Now tag and push the image into the docker container



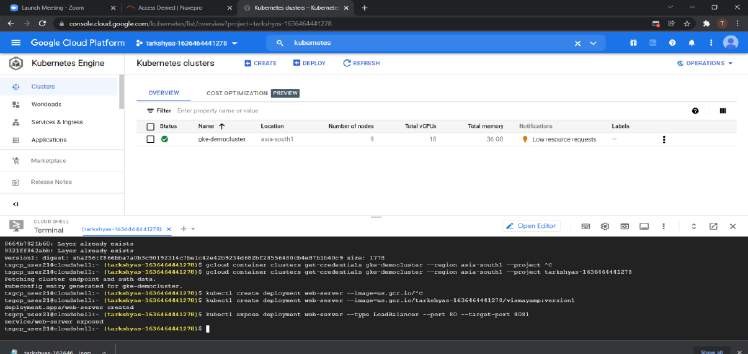
### Create a Kubernetes cluster with GKE standard , name it give region and create it



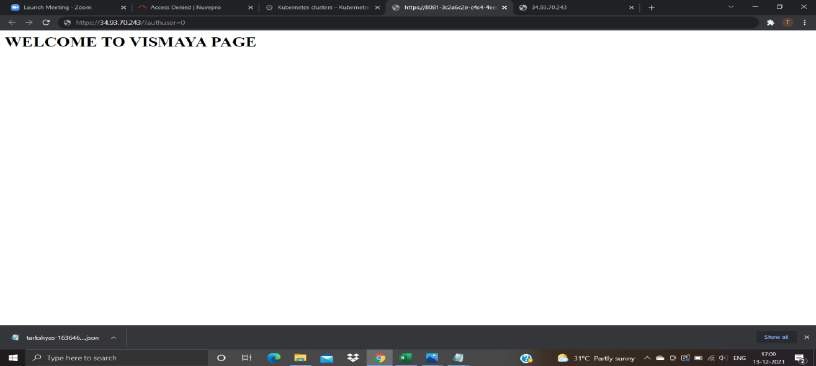
Cluster has been created, connect it to the cloudshell



Create and expose webserver ny kubectl create and expose command in the cloudshell



The external-ip for the deployment has been got,copy and paste it in the google to check the output

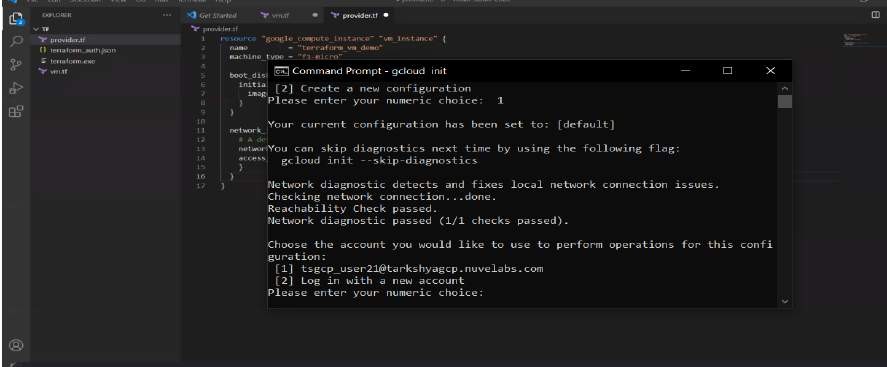
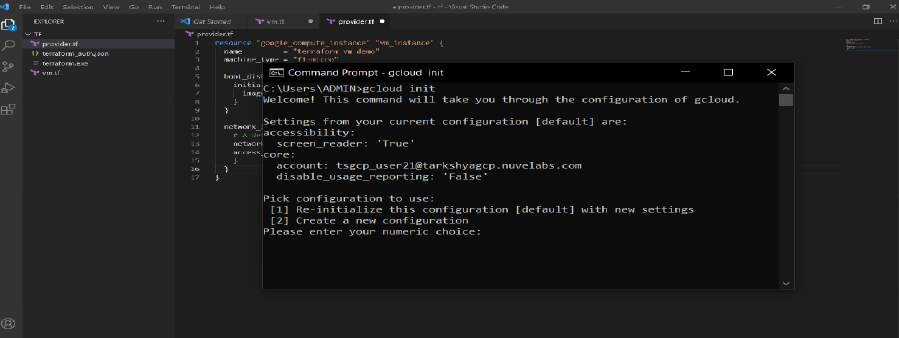


## Terraform

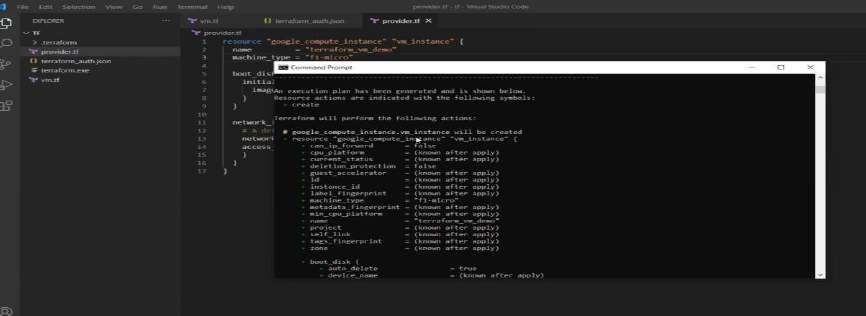
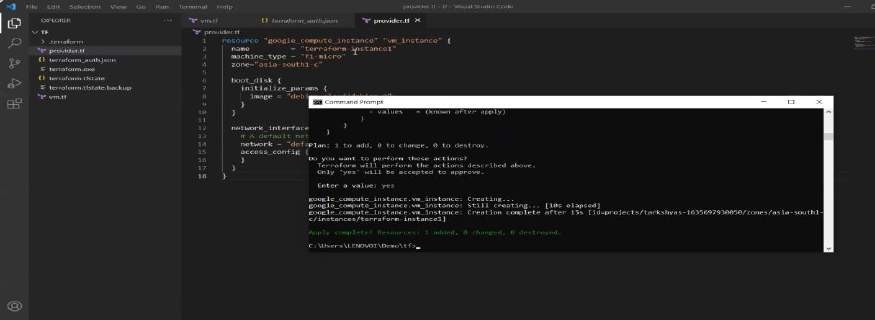
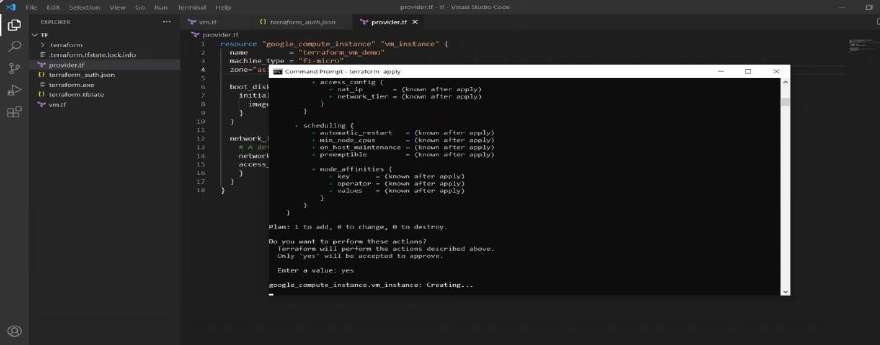
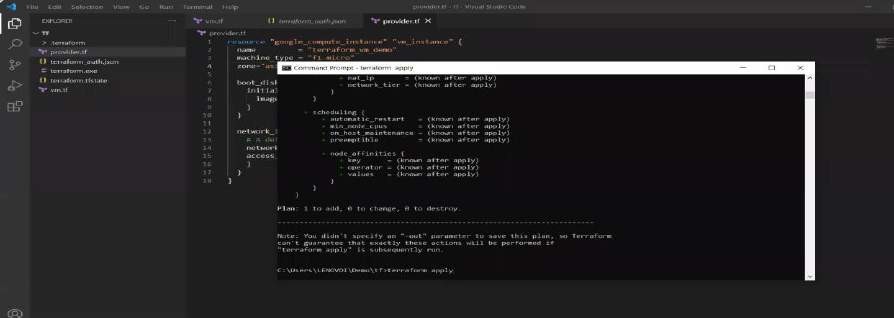
### Creating a VM instance using Terraform

For that first of all open terraform and import main files and tf files into it for creating the VM instance

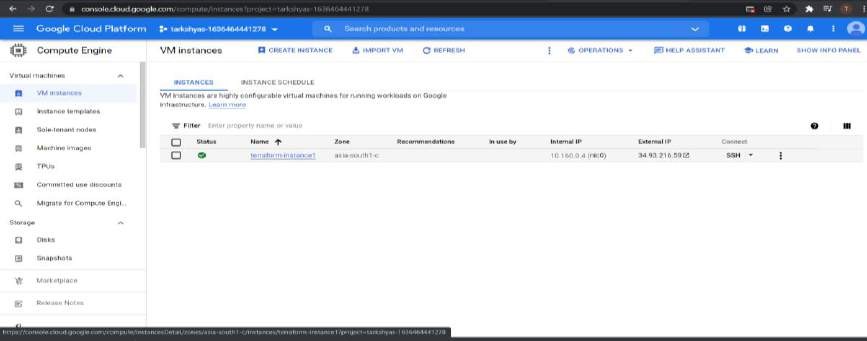
Open the command prompt and connect it with our google cloud account



### Terraform has been initialized

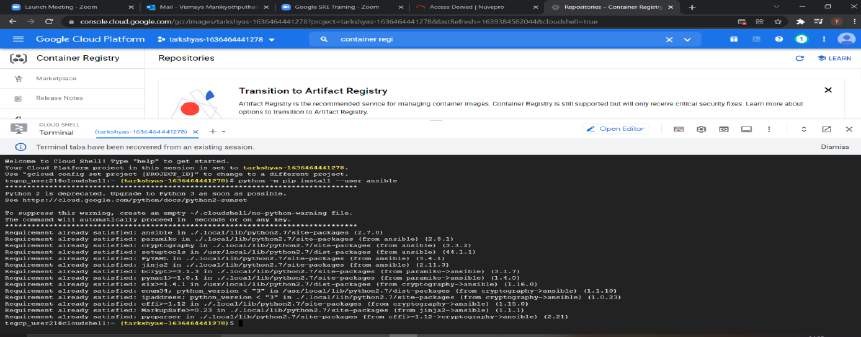


VM has been created,now go and check on the google cloud to see the resulted VM

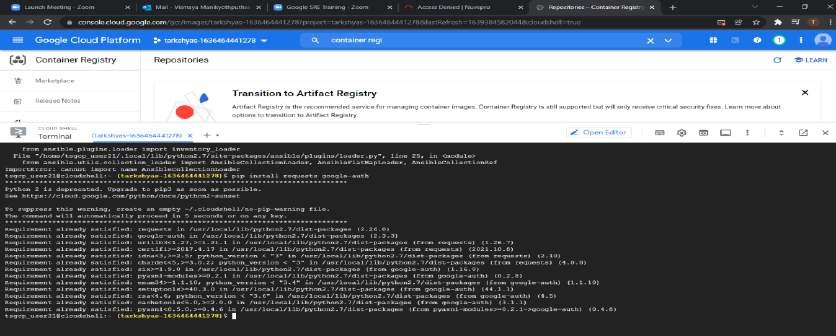


## Ansible

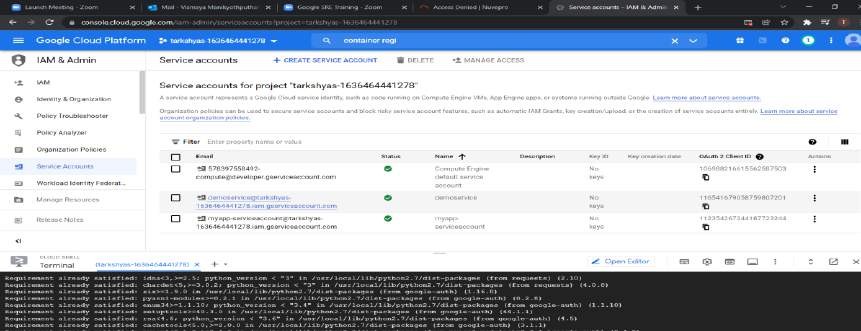
Open the google cloudshell -→install the ansible and python version latest



Install the google auth



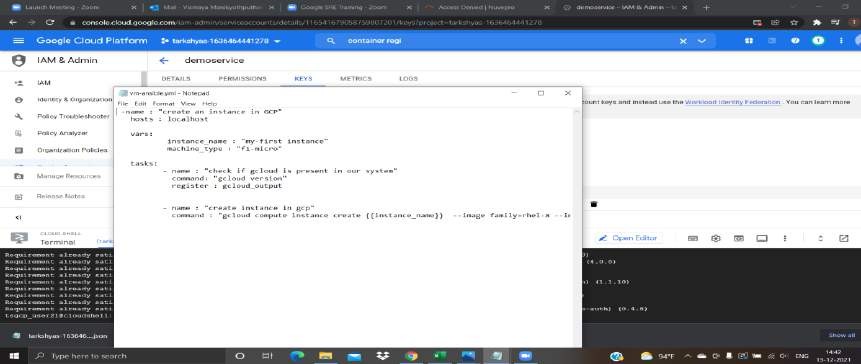
Create a service account with role as project owner



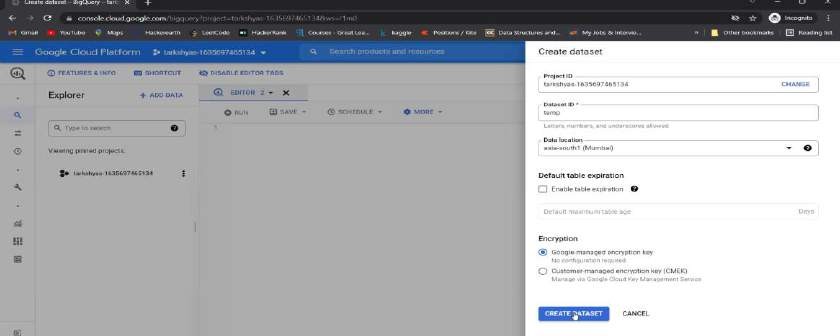
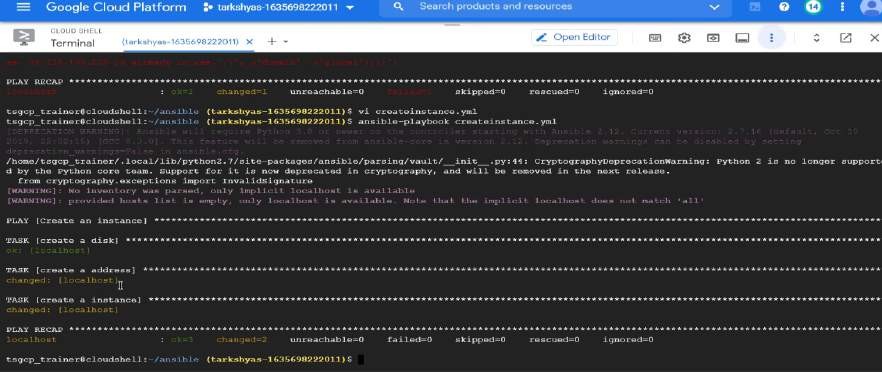
Manage keys-→keys-→json file is downloaded



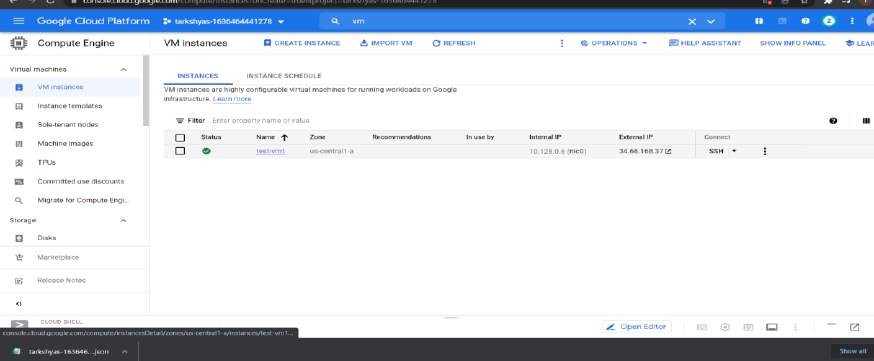
Upload the json file and yml file into the google shell which gives the details for creating an instance mentioning name,machinetype,region etc



VM has been created successfully by giving ansible-playbook command

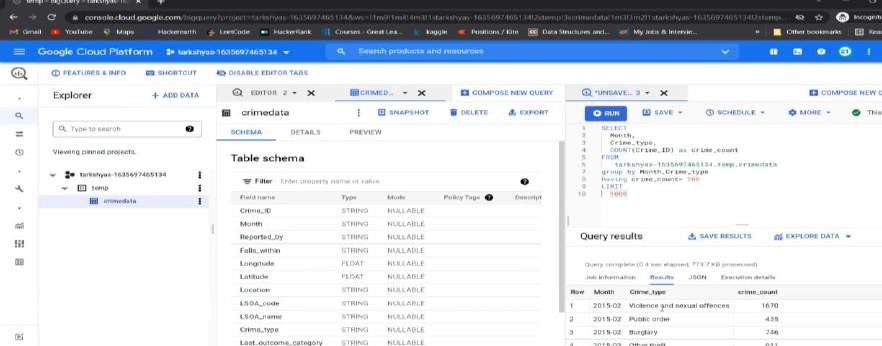
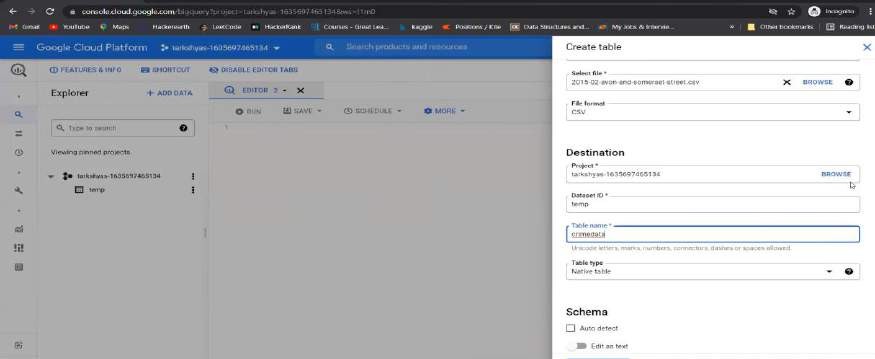


## Big-query

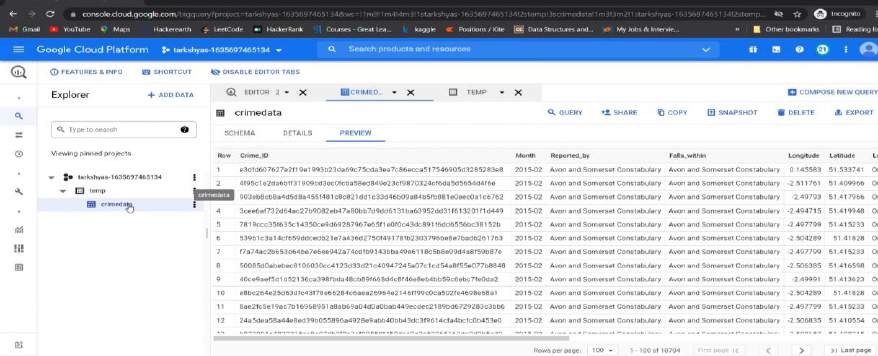


Create a dataset in the sql workspace

Create a table in the previously created dataset by uploading files from the device to query it

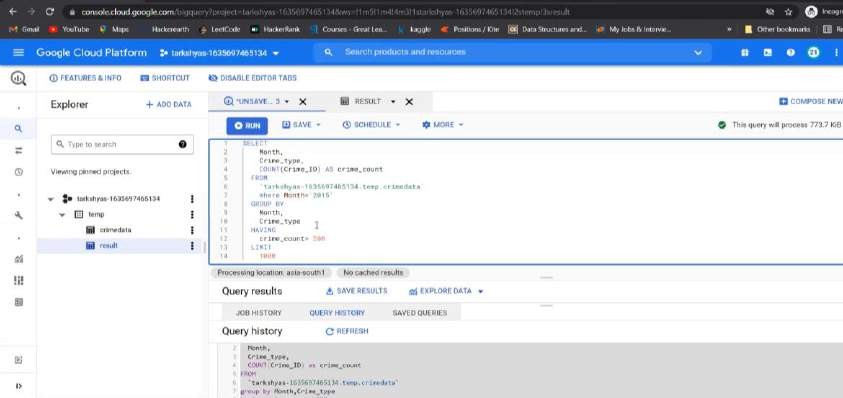


### Preview the imported data

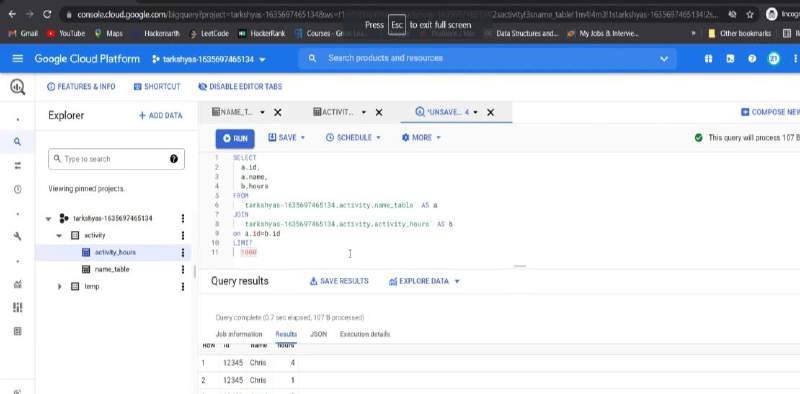


Query the data in a split shell

Group by



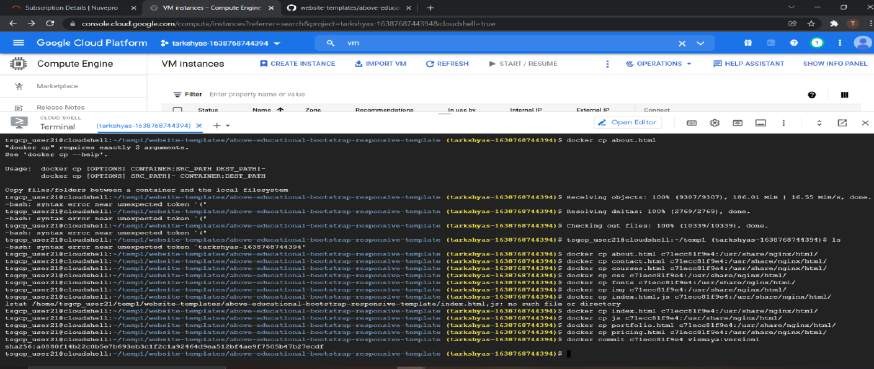
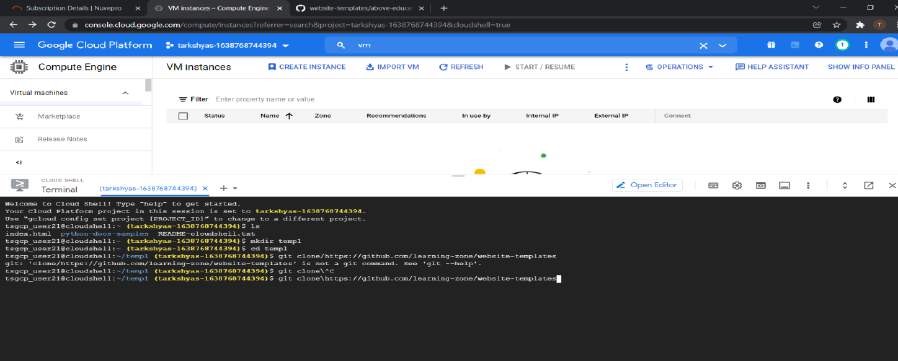
Join by



1. **Creating a linux VM and hosting a sample website given in below link**

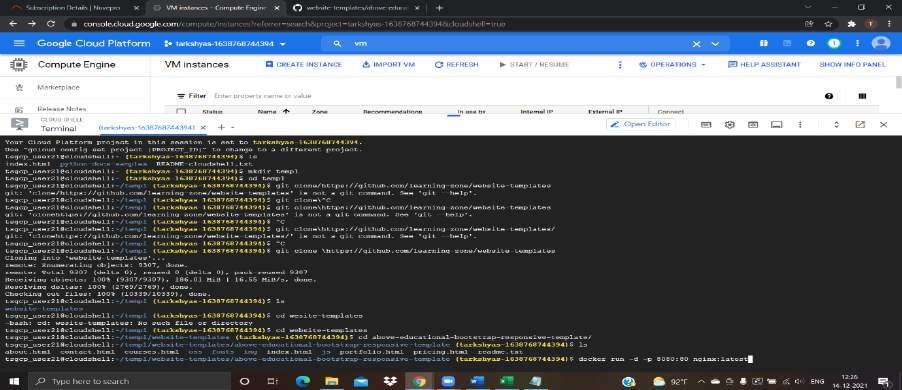
The link we are using here to host the website is;

[https://github.com/learning-zone/website-templates/tree/master/above-](https://github.com/learning-zone/website-templates/tree/master/above-educational-bootstrap-responsive-template) [educational-bootstrap-responsive-template](https://github.com/learning-zone/website-templates/tree/master/above-educational-bootstrap-responsive-template)



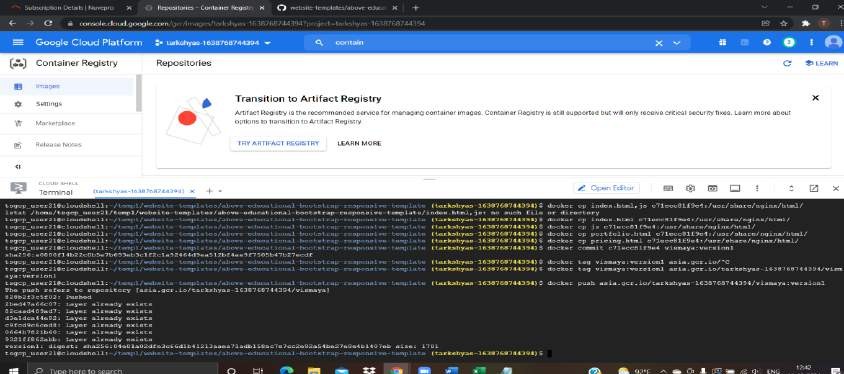
gclone the website link

After cloning,make a container registry by docker run command

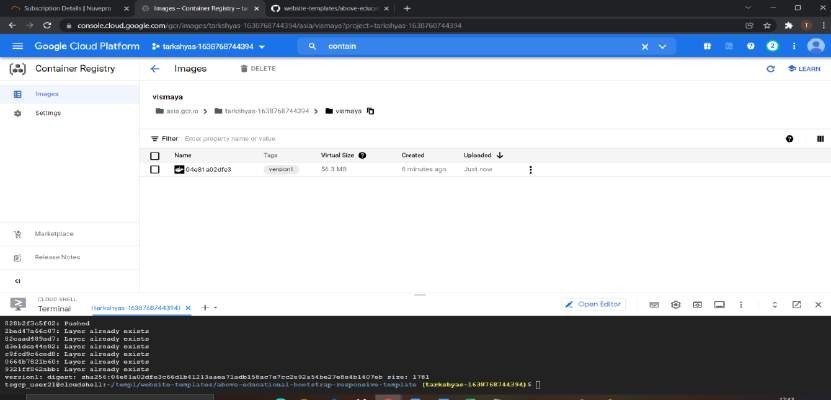


Obtain the container id by docker run command and commit all the files

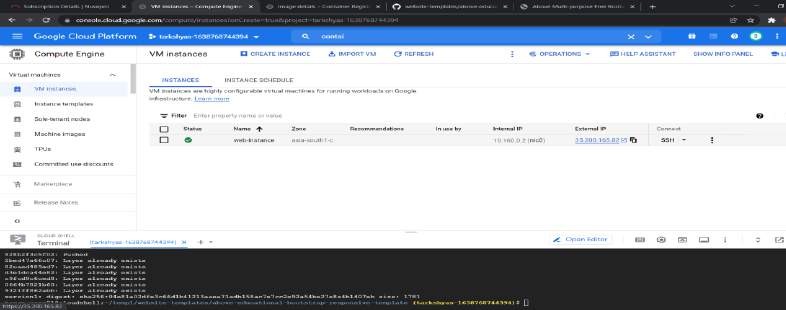
Tag and push all the files into the container registry



Our container is been created.later copy the path link of the container



Now create a VM instance specifying proper region,name,machine type and in the container deploy option paste the link that we had copied earlier and create the instance



VM has been created , click on the external ip to see the website that had created.



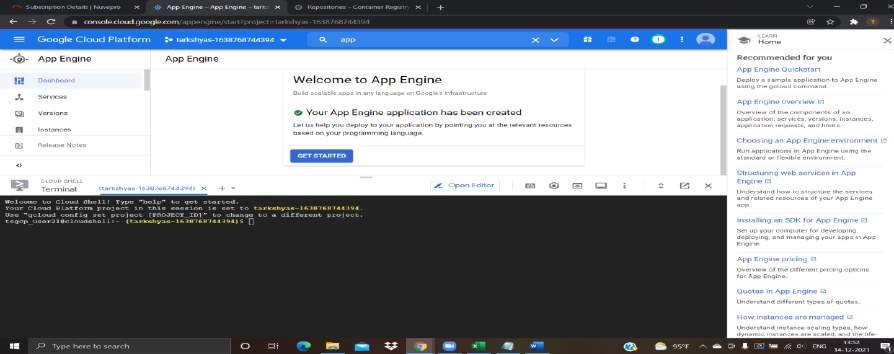
Hence website has been hosted

## Hosting a website in App-engine using standard environment

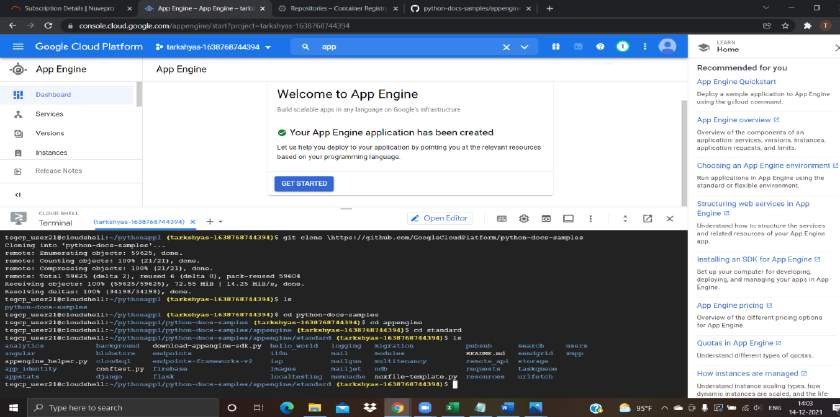
### The link we are using here for deploying the website is

[https://github.com/GoogleCloudPlatform/python-docs-](https://github.com/GoogleCloudPlatform/python-docs-samples/tree/main/appengine/standard/cloudsql) [samples/tree/main/appengine/standard/cloudsql](https://github.com/GoogleCloudPlatform/python-docs-samples/tree/main/appengine/standard/cloudsql)

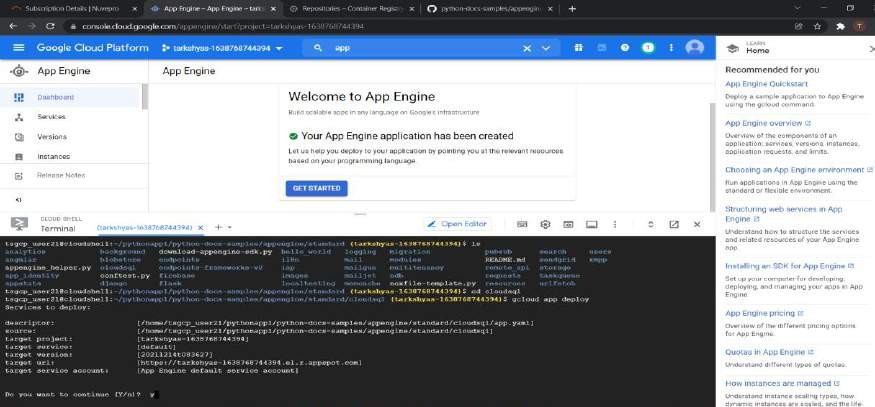
create an app-engine by giving proper region and in standard environment.



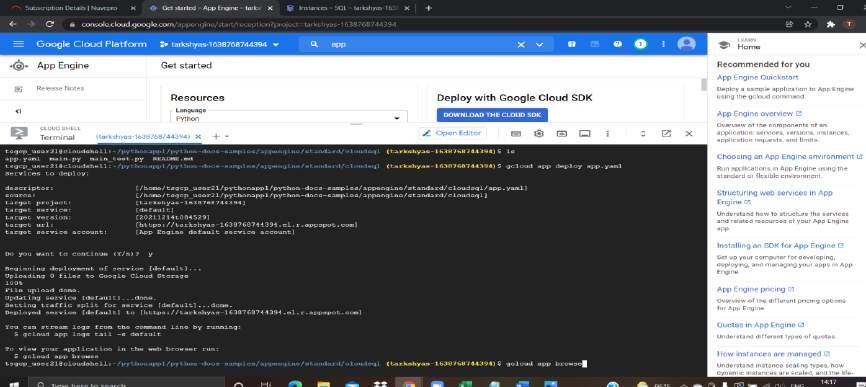
Now activate the cloudshell and make a directory and clone the given website link into it



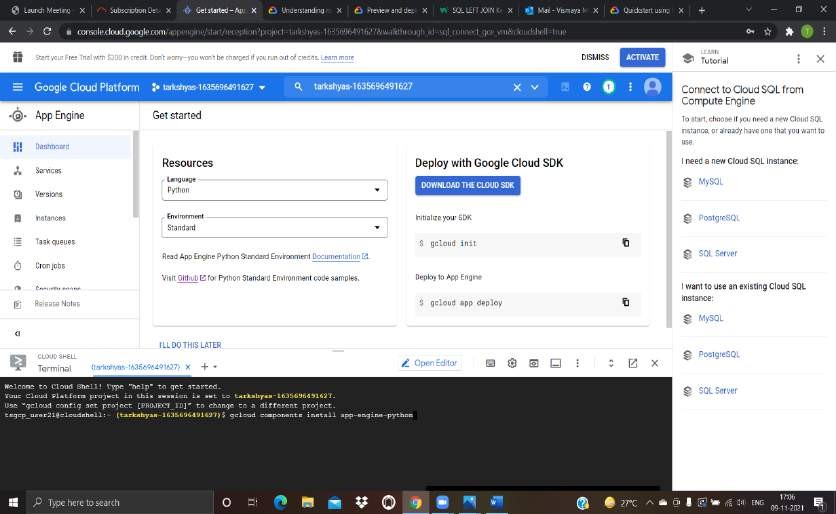
All our files are now inside the directory.Deploy the app by gcloud app deploy command



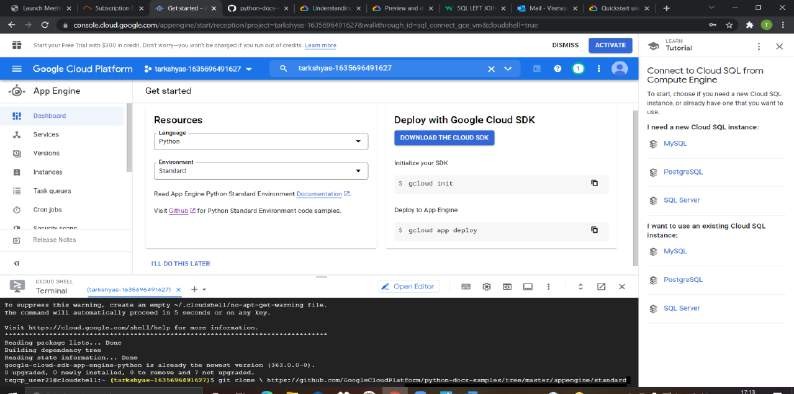
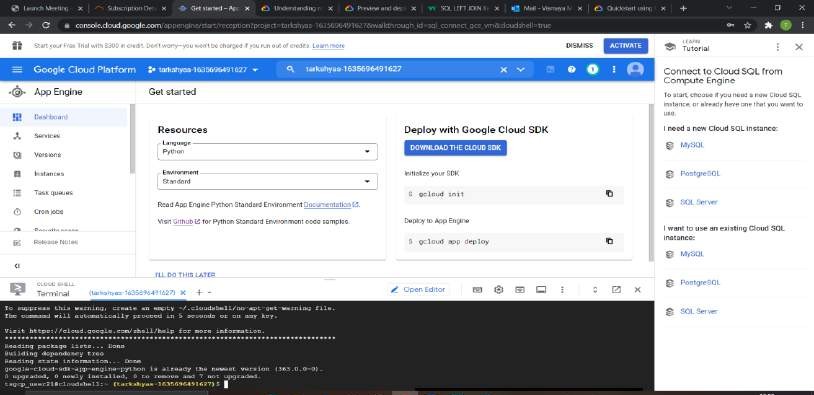
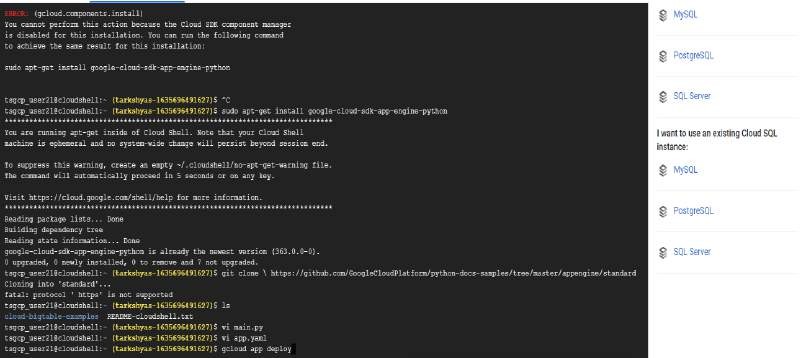
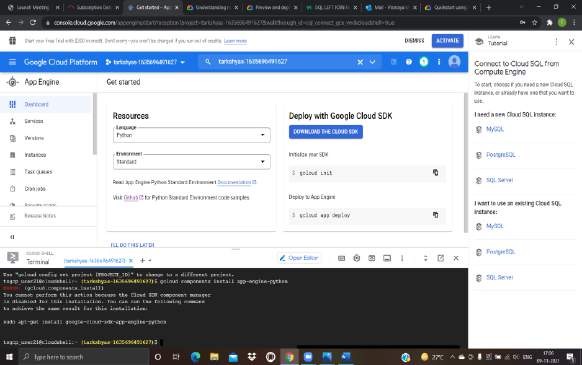
Now browse the app to get the target url



#### Deploying a helloworld app in app-engine using flexible environment



Create app-engine in flexible environment and activate the cloudshell and enter the gcp commands.



completed