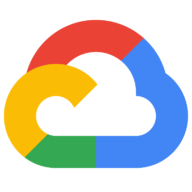
****

**Google Cloud Platform**

Central doc for execution ref - <https://docs.google.com/document/d/1Z50mY_kfaL8GPzjAS0AV64NIQPpLFrk5z-KT0t-cbts/edit?usp=sharing>

*25th April 2023*

**Requirements**

* Storage
* Processing
* Network
* Power
* Heat Dissipation

**Cloud Computing** is a model for enabling on demand network access to a shared pool of configurable resources that can be rapidly provisioned and released with minimal management effort or service provider interaction

**Essential characteristics**

* **On demand self service**
  + Users can avail computing capabilities such as server time and network storage as needed automatically without requiring human interaction.
* **Broad network access**
  + Capabilities are available over the network and can be easily accessed from client platforms like phones and tablets
* **Resource pooling**
  + Providers computing resources are pooled to serve multiple consumers using a multi-tenant model with different physical and virtual resources dynamically assigned according to the consumer demand.
* **Rapid elasticity** 
  + Capabilities can be elastically provisioned and released automatically to scale rapidly outward and inward commensurate with demand
* **Measured service** 
  + Has the ability to automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service such as storage processing active user accounts etc.

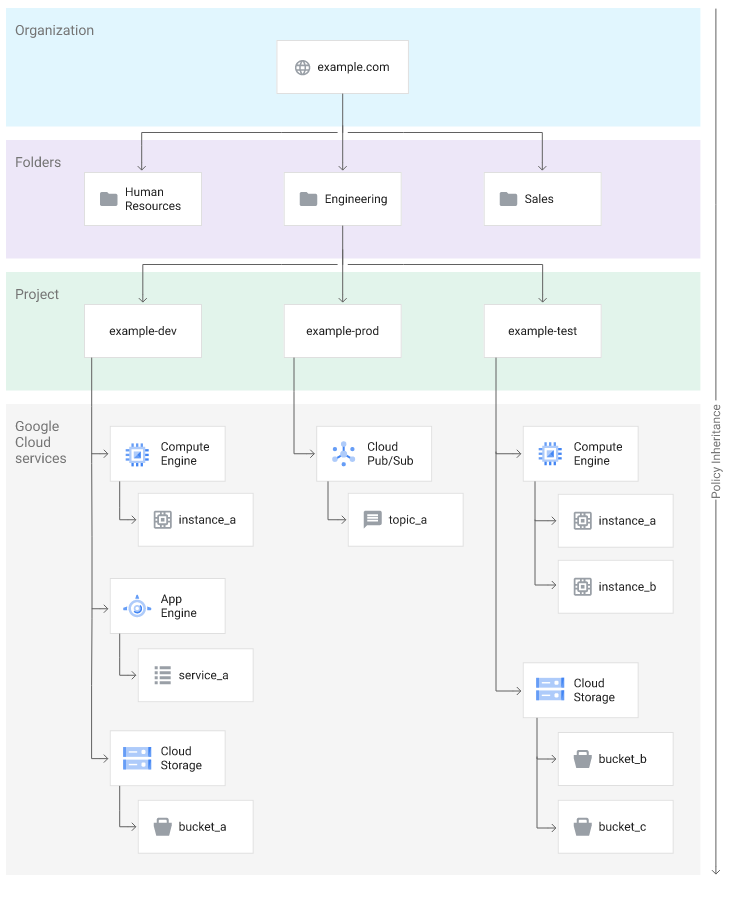
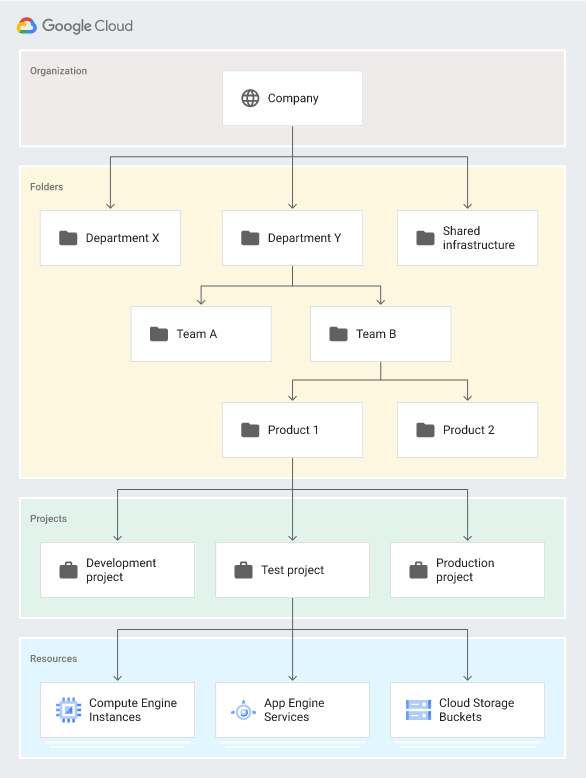
**Service Models**

* **SaaS**
* **PaaS**
* **IaaS**

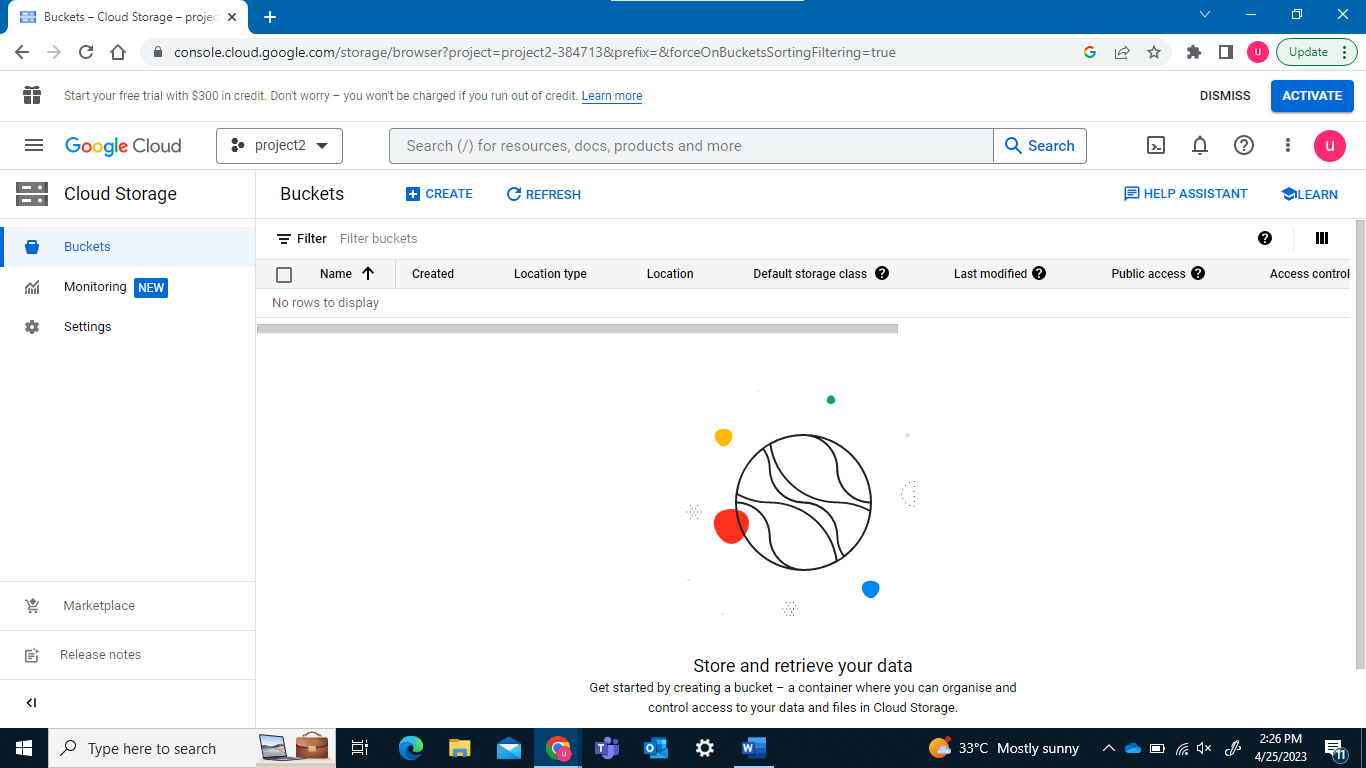
**Deployment Models**

* **Private Cloud**
* **Community Cloud**
* **Public Cloud**
* **Hybrid Cloud**

**GCP resource hierarchy**

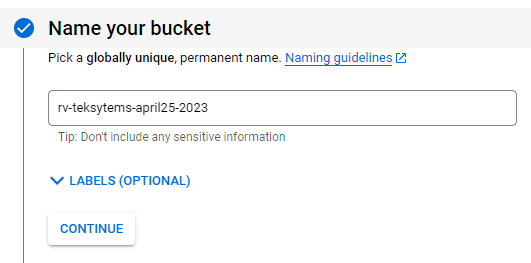
**GCS – Google Cloud Storage**

****

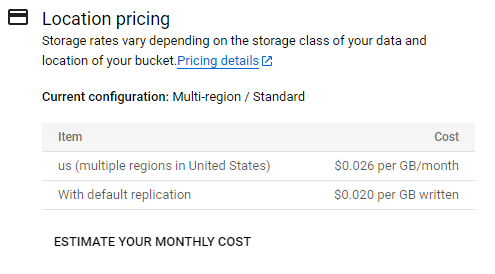
* Managed service for storing unstructured data
* Store any amount of data and retrieve it as often as you like
* Block Storage

**Creating a bucket**

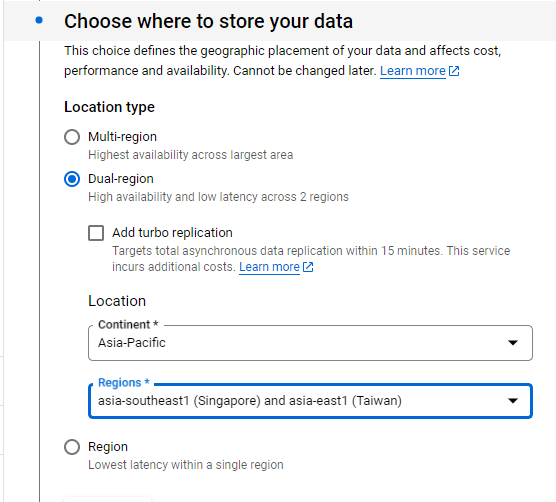
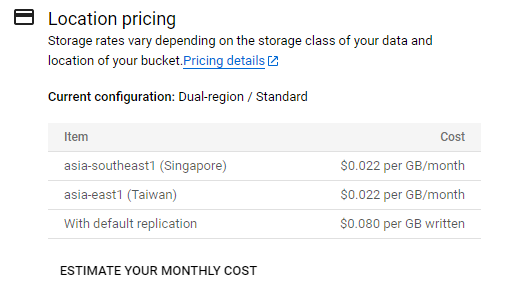




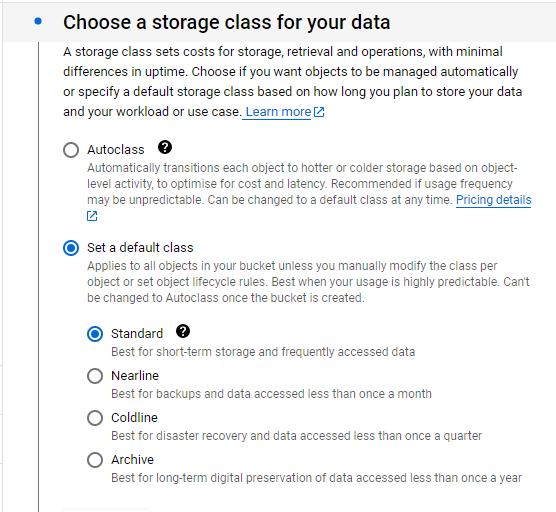
For a single region model

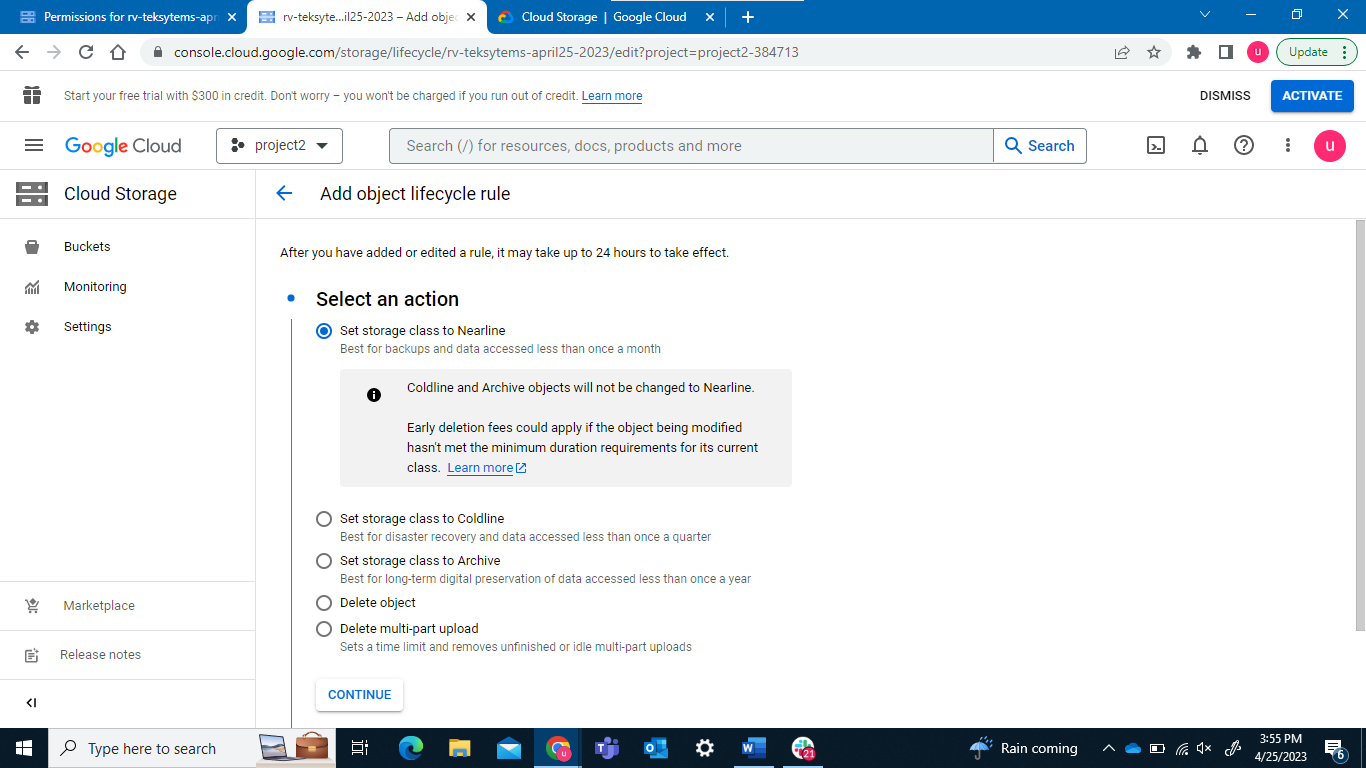
for replication purposes select dual region and hence the process increases

A storage class sets costs for storage, retrieval and operations, with minimal differences in uptime.



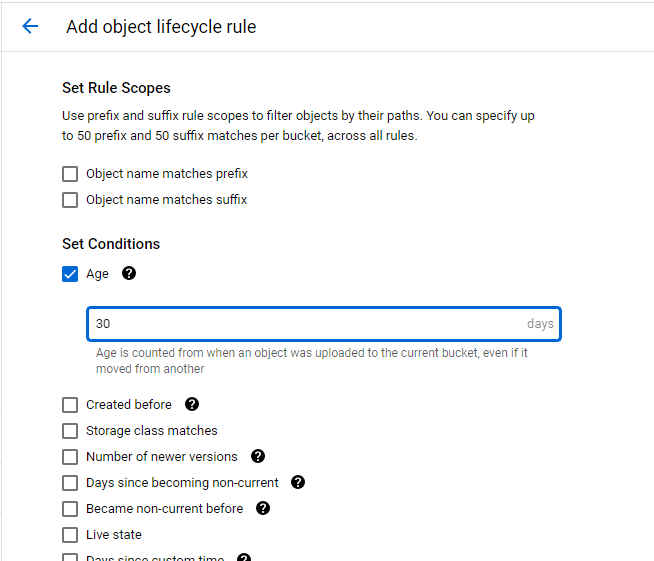
Adding a life Cycle



More frequency – hotline

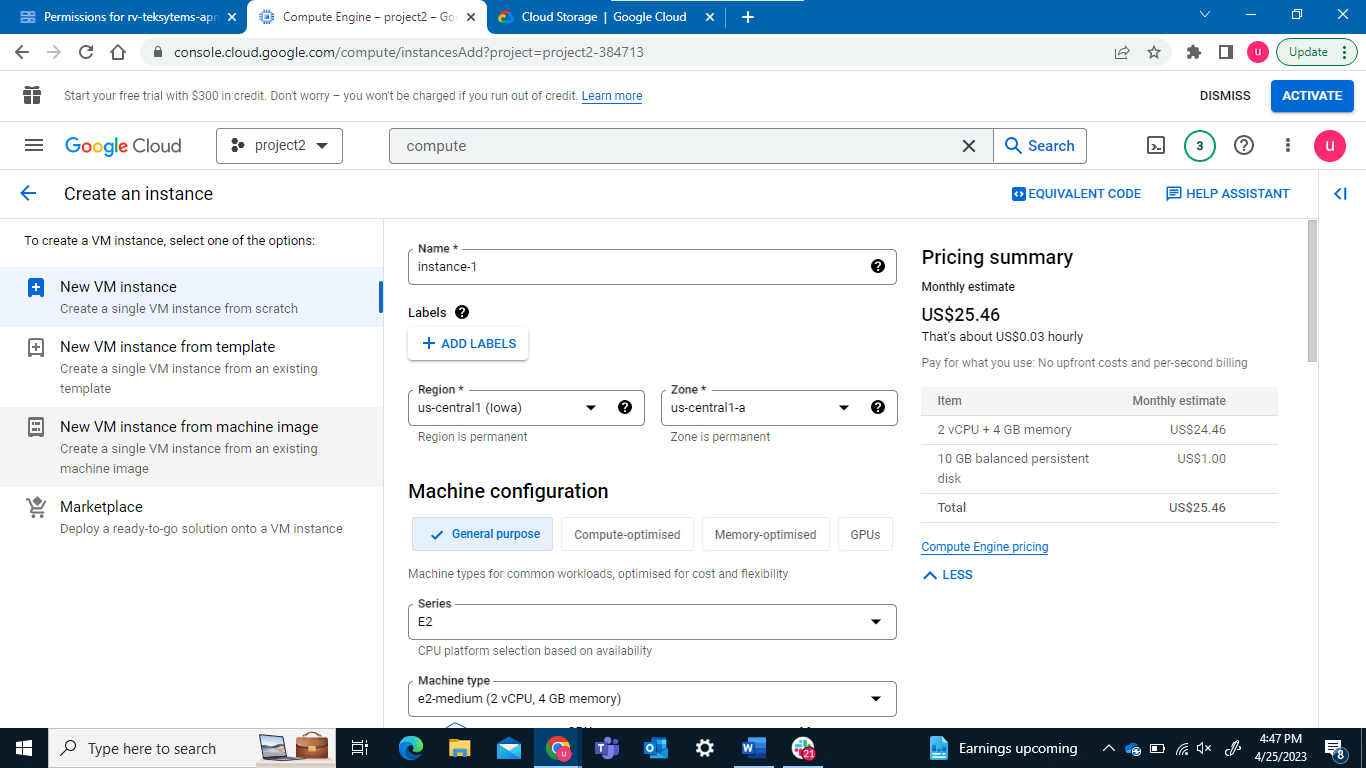
less frequency – coldline

Once per month – nearline



**Compute Engine**

Creating a new instance



*26th April 2023*

VM Instances

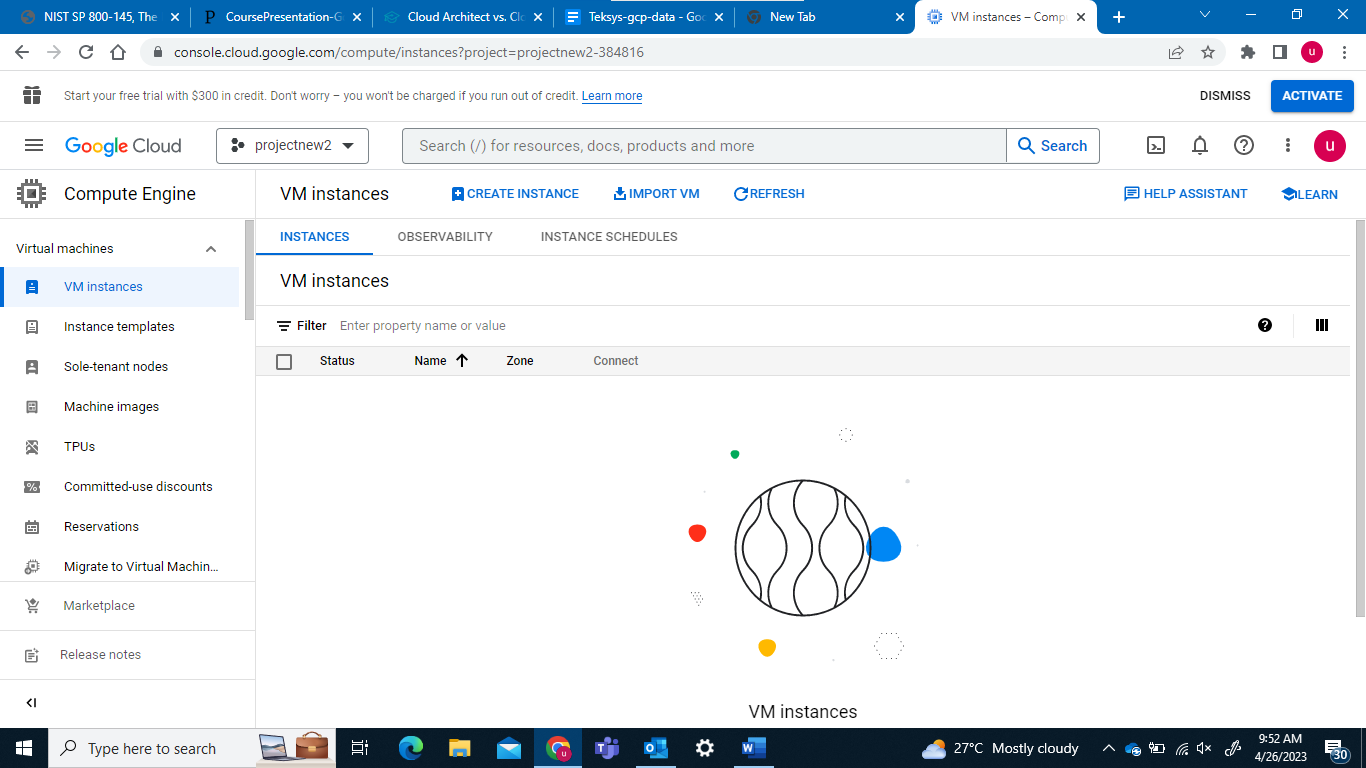
Creating a VM & hosting Apache web distribution manually

1. *Go to Compute Engine > VM instances > Create*
2. *Specs:*
   1. *Name - machine name preferred*
   2. *Region - us-central1 zone - us-central1-a*
   3. *Machine series: N1 Type n1-std-1*
   4. *Boot disk:*
      1. *OS: Ubuntu*
      2. *Version: 18.04 (x86 based)*
   5. *Create VM*
3. *Wait for the VM to have running status. Locate ssh button besides VM name.*
4. *Click on ssh button to connect with vm.*
5. *Execute following commands:*
   1. *sudo apt update*
   2. *sudo apt install apache2 -y*
6. *Go back to GCP console. Locate “External IP” in the VM entry. Copy the external IP and paste in any browser tab to access Apache webpage. You should get a page not found error.*

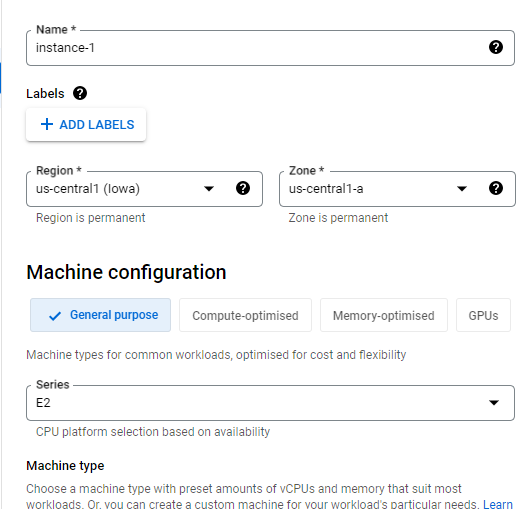
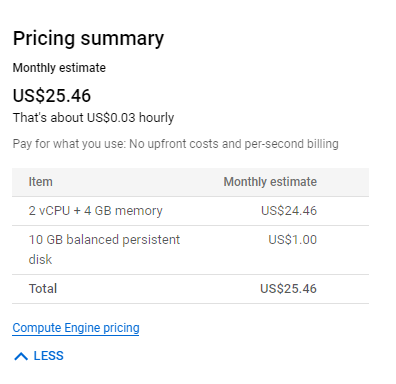
*{If you’re not able to locate external IP, click on three slider icon towards the right top corner of VM entry. This option is* ***column-display-options****. Select external IP & internal IP from the options.}*

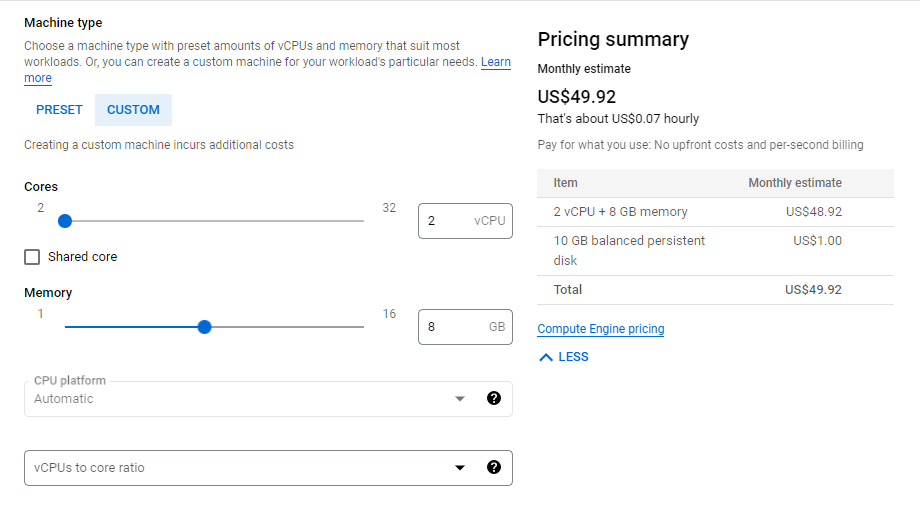
1. *Go back to GCP console > locate “Internal IP”. Go to the ssh command window and execute command:*
   1. *curl* [*http://[internal-ip*](about:blank)*]*
2. *You should see an html code. Take a screenshot of this as last step.*

***Delete*** *the VM instance resource.*

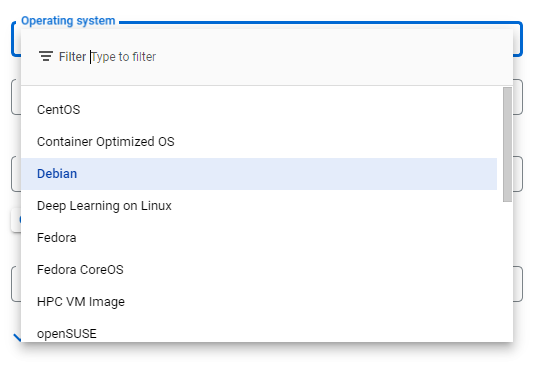
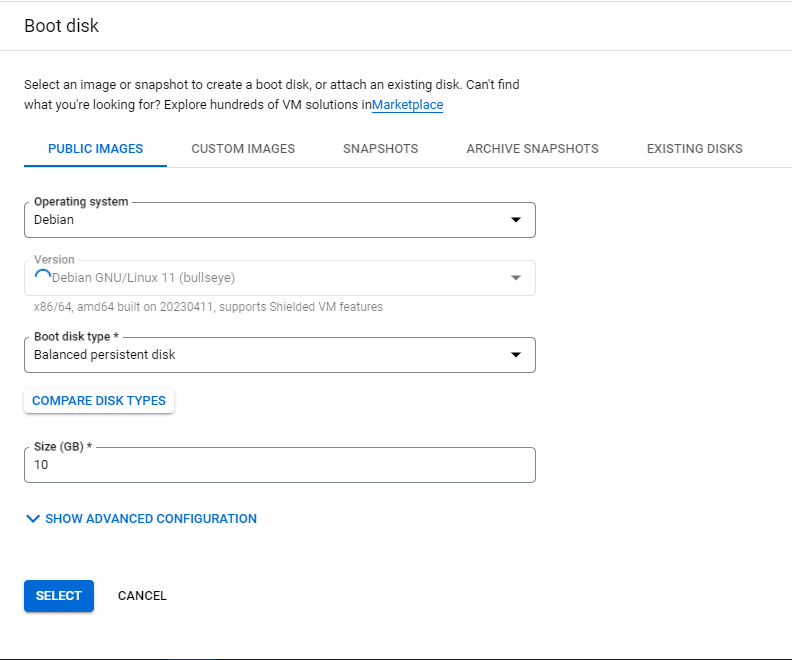


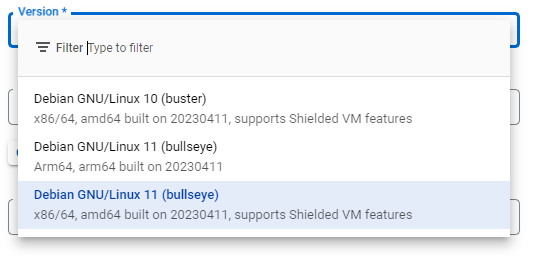


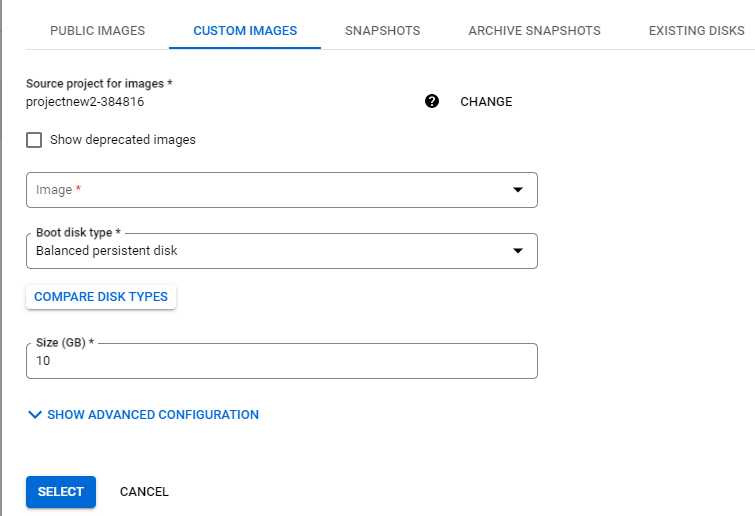
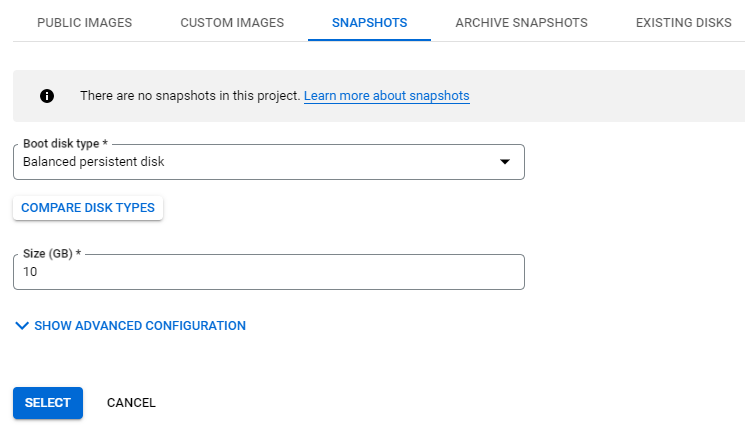
 



**Boot Disk**



**Backup of a disk -> Snapshot**

* Software and application state is being backed

**Backup of a vm -> Image**

* Additional meta data
* And the specs using which the vm was made is being backed up

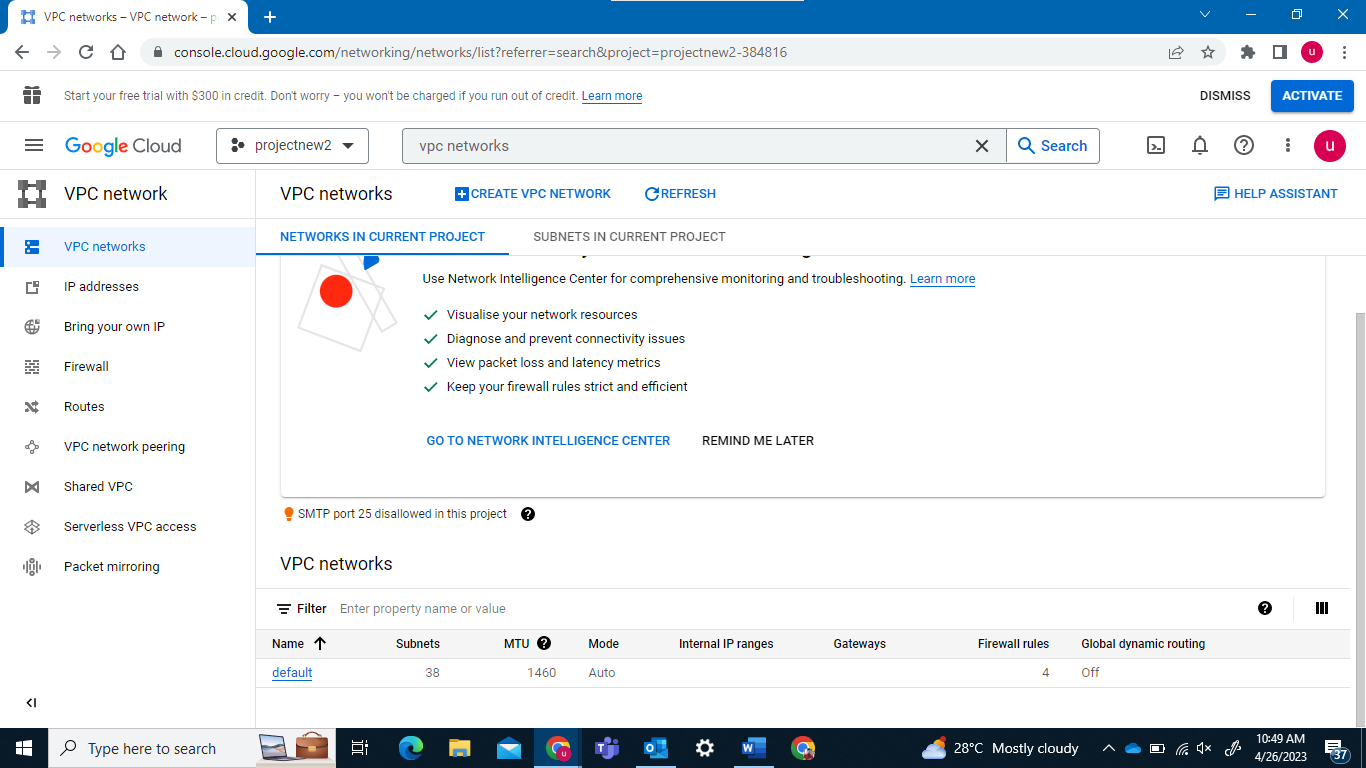
|  |  |
| --- | --- |
| **In GCP**  *EC2 🡨🡪 Image*  *Disk 🡨🡪 Snapshot*  *Image 🡨🡪 Snapshot* | **In AWS**  *Vm 🡨🡪 image*  *Vm 🡨🡪 disk*  *Vm 🡨🡪 Snapshot*  *Image 🡨🡪 Snapshot*  *Disk 🡨🡪 Snapshot* |

**IPV4 vs IPV6**

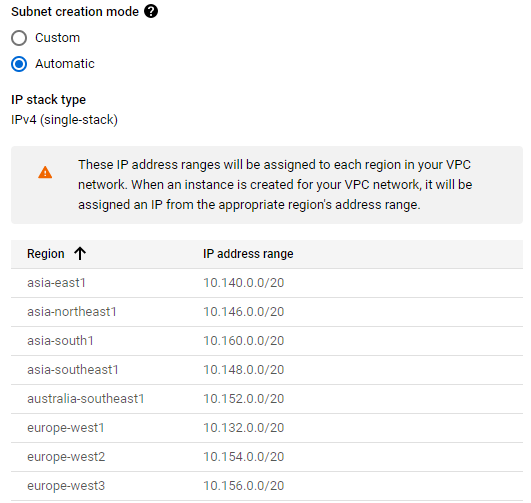
|  |  |
| --- | --- |
| **IPV4**  IP addresses in version 4 type are designed to be of 32-bit type binary format and contain around 232 addresses, and these addresses were sufficient as a primary requirement. | **IPV6**  An IPv6 address type is designed of 128 bits from which 4 are hexadecimal digits, and it created eight sets, with each block containing 16 bits separated by a colon (:). |
|  |  |

* **Firewall Rules**
* **Networking protocols**

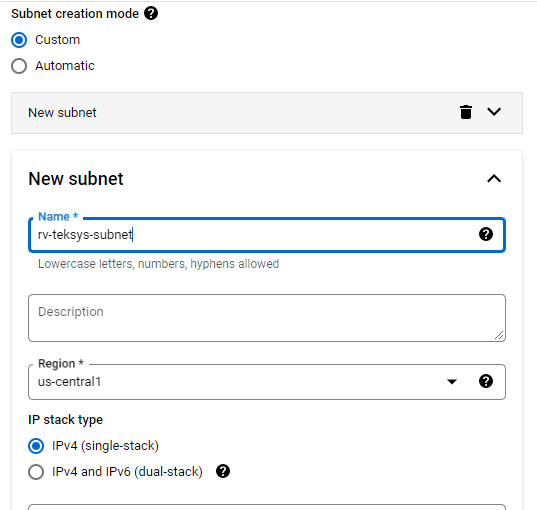
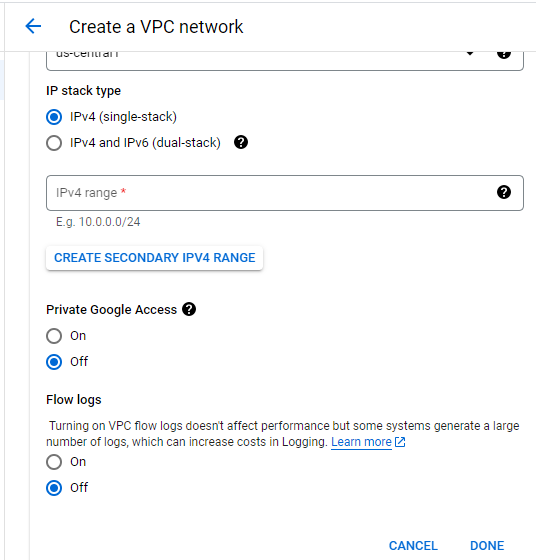
**VPC network**



When selected automatic it uses 4 subnetworks which keeps on scaling on updation .

While using custom

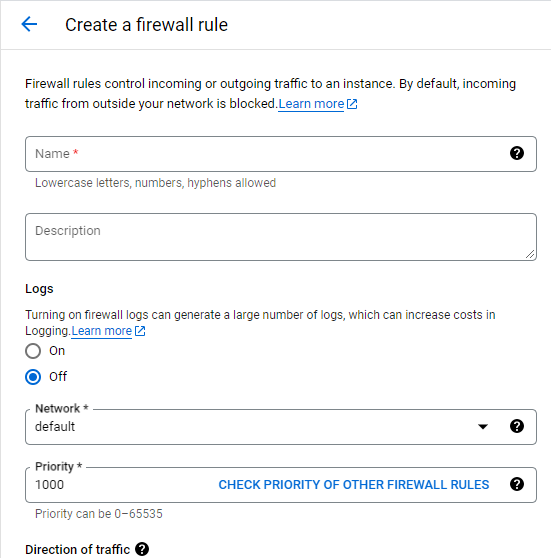
 

Creating a firewall Rule

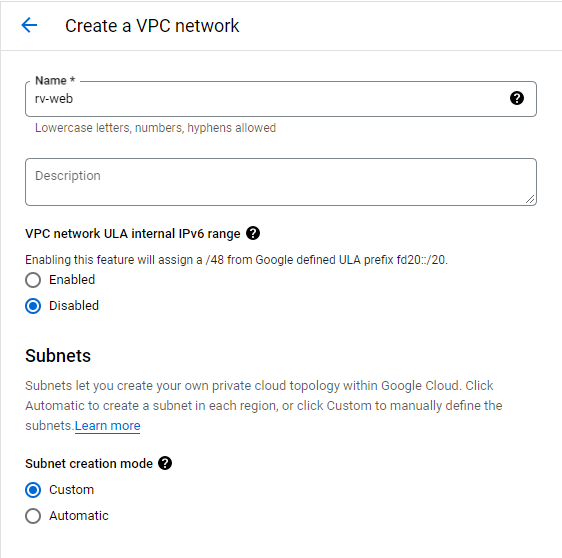
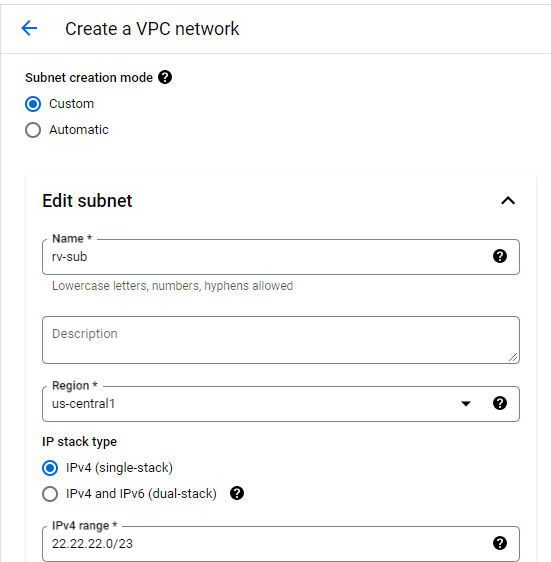
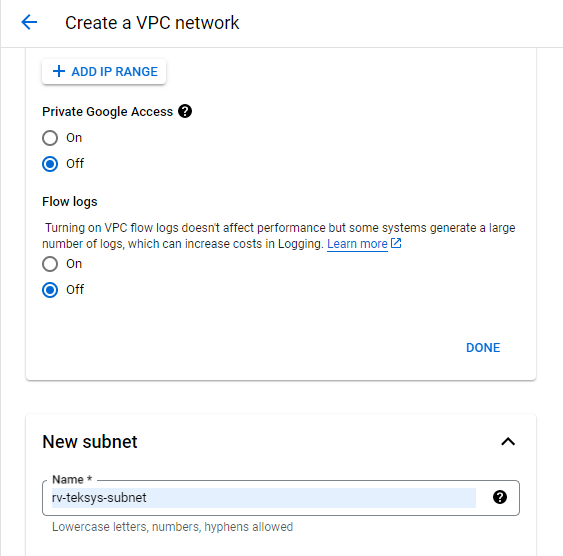
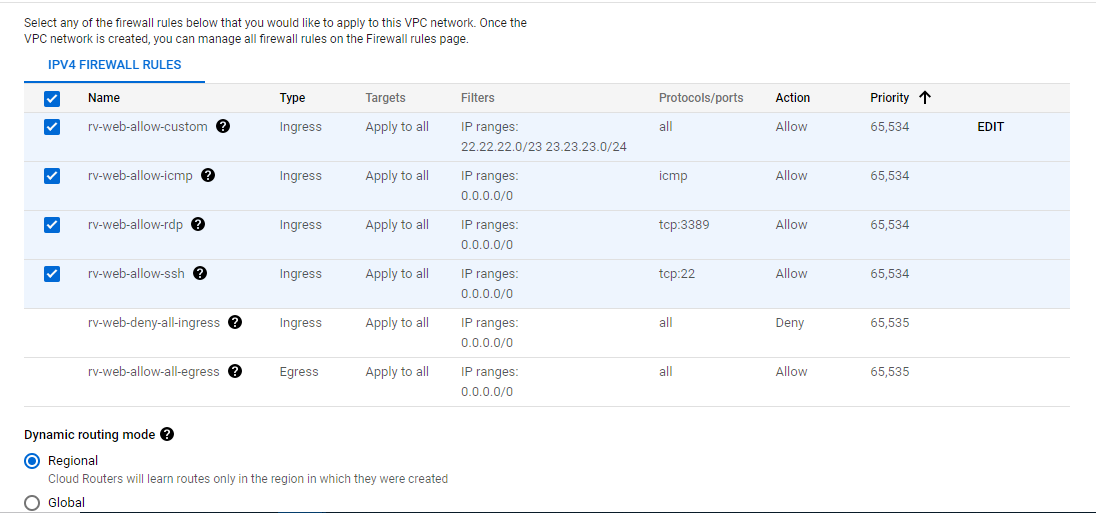


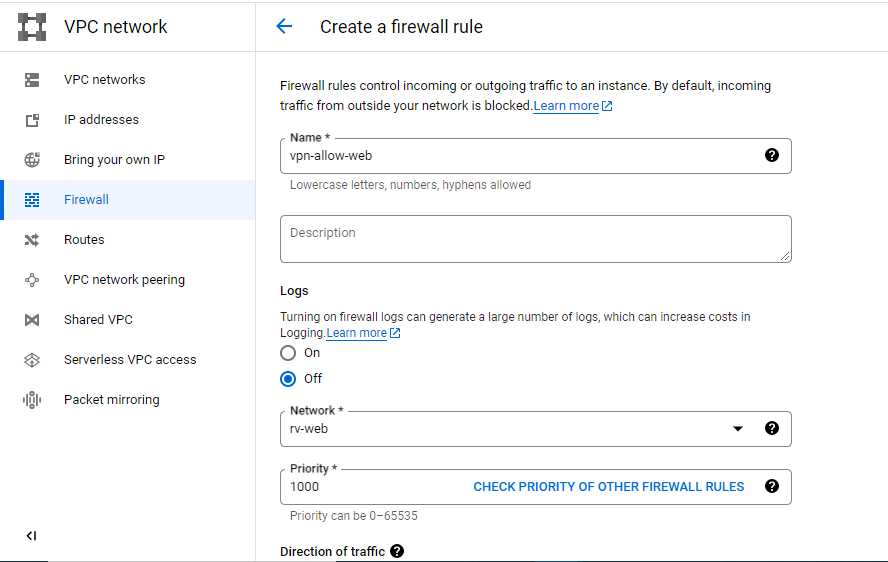
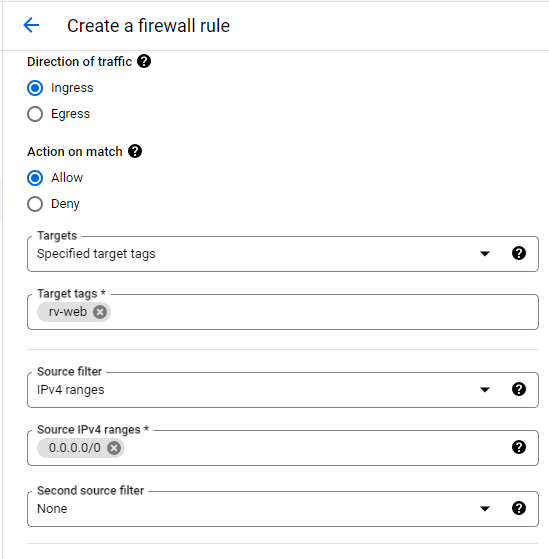
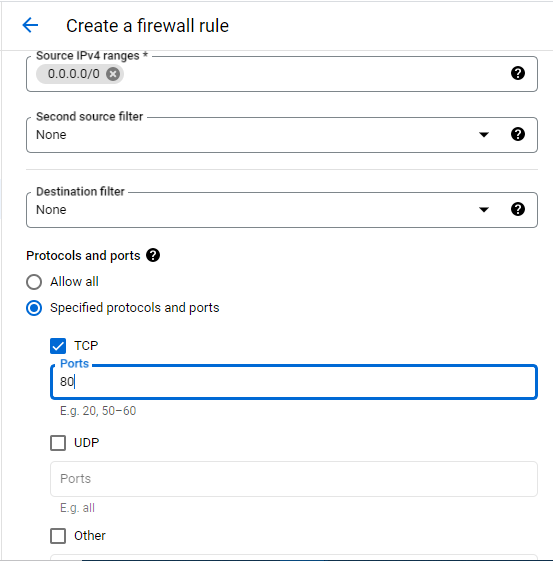


Set priority rule for assigning preference in case 2 person created for same



Create VPC

Creating a VPC

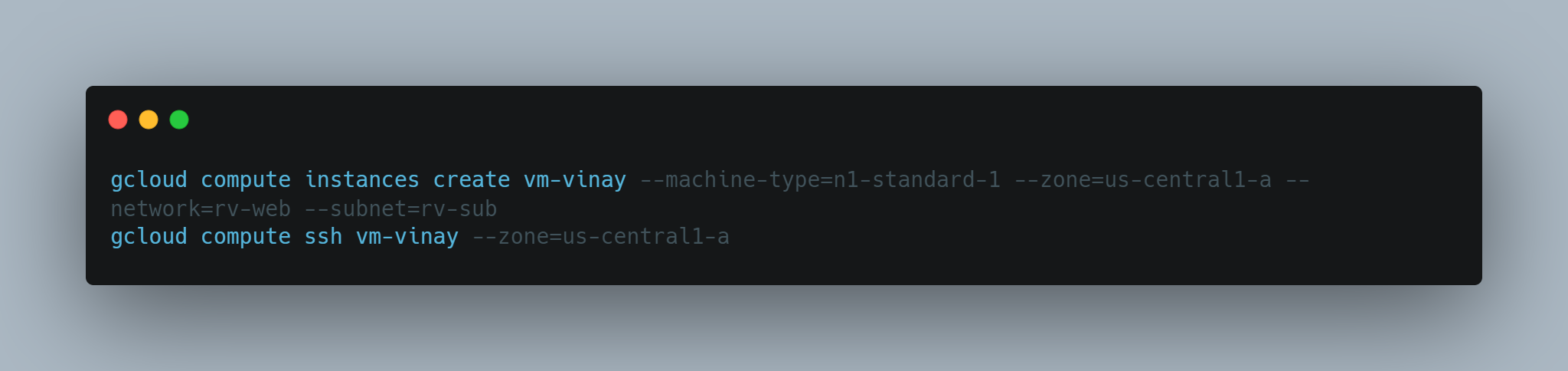
1. Go to VPC Networks > Create VPC
   1. Name: john-web
   2. Mode of Creation: custom
   3. Subnet name: john-sub
   4. Range: select a relevant cidr range
   5. Region: us-central1 (feel free to change)
   6. Create subnet
   7. Firewall rules: select all entries
   8. Create VPC
2. Once VPC is created, go to firewall rules
   1. Create firewall rule
   2. Name: vpc-allow-http
   3. Network: select your vpc from dropdown
   4. Target: input a target tag (http-john, spider-man)
   5. Source Filter: 0.0.0.0/0
   6. Protocol: TCP - 80
3. Visit compute engine > create instance
   1. Ensure all settings as earlier exec
   2. Click on Advanced > Networking
   3. Select your vpc name from network drop-down > Done
   4. Click on Automation > Startup script
   5. Input startup script as follows:
      1. apt update
      2. apt -y install apache2
      3. cat <<EOF > /var/www/html/index.html
      4. <html><body><p>Linux startup script from a local file.</p></body></html>
      5. EOF

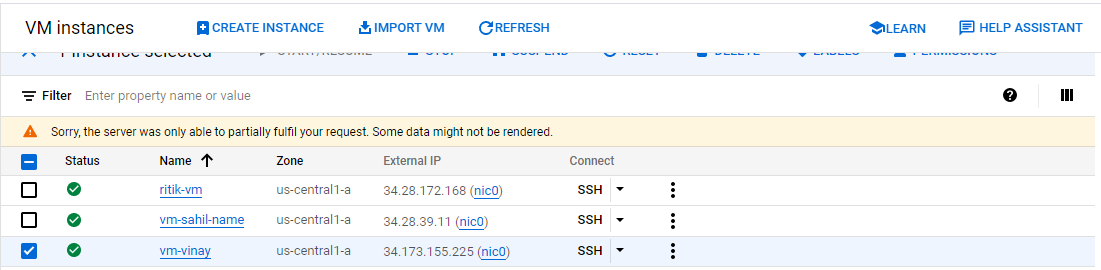
Create Instance

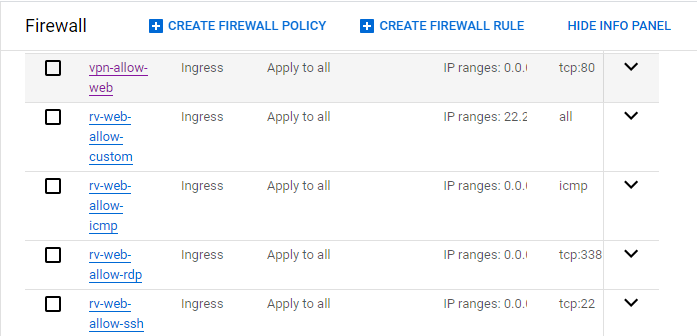
CLI way -

* For creating instance: gcloud compute instances create vm-name --machine-type=n1-standard-1 --zone=us-central1-a --network=vpcname --subnet=subnetname
* For accessing the vm - gcloud compute ssh vm-name --zone=us-central1-a
* After accessing the vm, execute following commands:
  + sudo apt update
  + sudo apt install apache2 -y

1. Visit External IP of instance to check if apache2 hosted webpage is available.
2. For consistent failed page loads, go to firewall rule > click on the firewall rule you have recently created > edit > target: change to “all instances in the network” > save
3. Retry accessing the external IP
4. Once the webpage is accessible, take screenshots and delete instance, firewall-rule & VPC network.





 Apply to all instances

*27th April 2023*

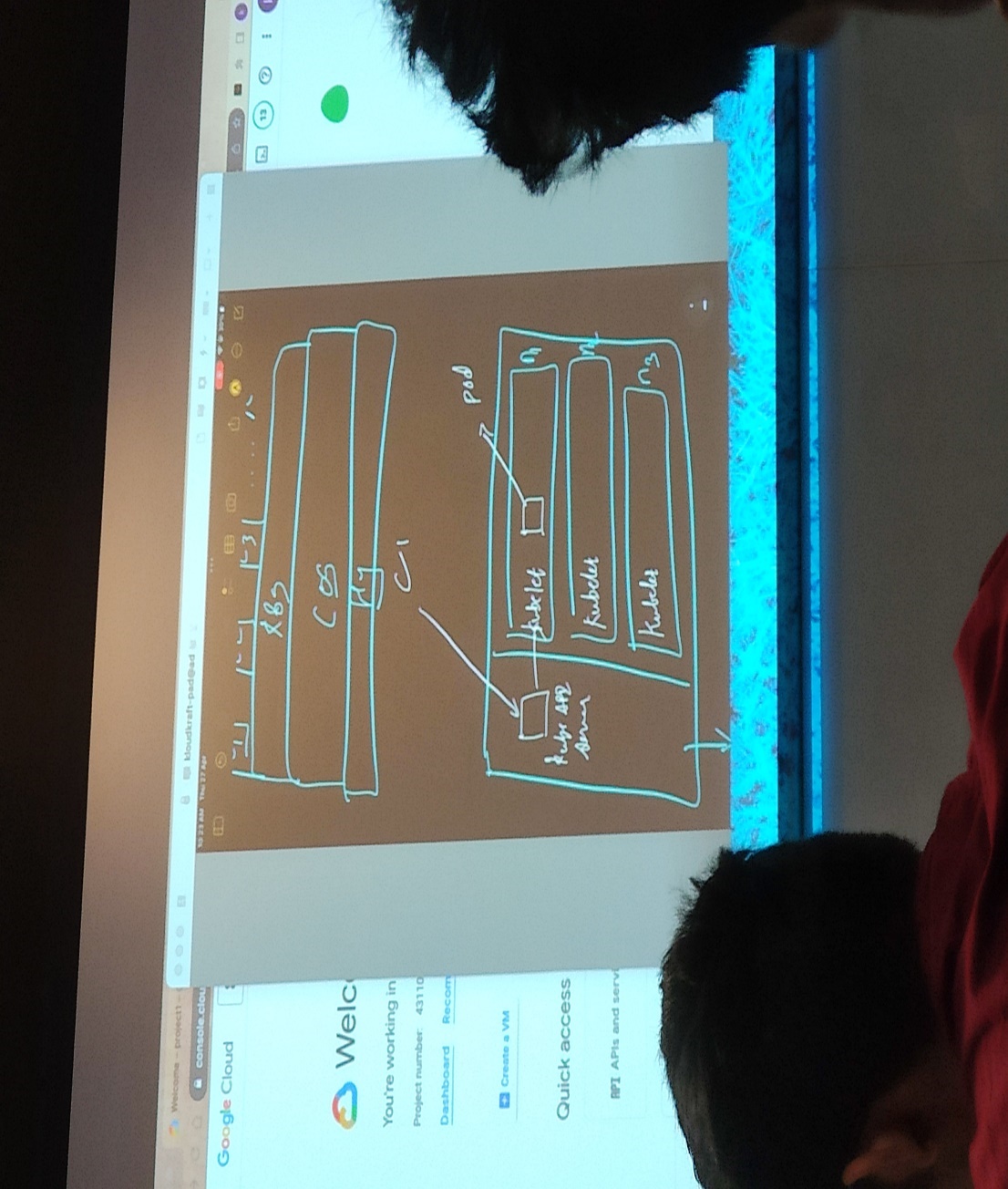
Creating the Python Application and Deploying it in GCP:

Creating App Engine

1. Clone the code repo: git clone <https://github.com/GoogleCloudPlatform/python-docs-samples.git>
2. Shifting code to directory: cd python-docs-samples/appengine/standard\_python3/hello\_world
3. Edit code file
   1. nano main.py
   2. Replace *Hello World!* with a custom message with your name.
   3. Save file [Ctrl+o, enter, Ctrl+x]
4. Deploy application using - gcloud app deploy
5. Fetch endpoint using - gcloud app browse
6. Visit the end point for app output
7. Edit code file for more changes using nano editor
8. Redeploy app using gcloud app deploy
9. Visit the endpoint for observing output change. It must be updated to new code snippet that you edited
10. Go to App Engine > Versions > Split traffic
11. Split traffic using percentage values and random mode. Save split
12. Try visiting the endpoint over few iterations and notice the output change

Containers in GCP

|  |  |  |
| --- | --- | --- |
| Application1 | Application2 | Application3 |
| Dependency1 | Dependency2 | Dependency3 |
| Operating system1 | Operating system2 | Operating system3 |
| Kernel1 | Kernel2 | Kernel3 |
|  | Hypervisor |  |



C1 is the control panel

The central parts represent GKE (Google Kubernetes Engine)

Here n1, n2 and n3 are the Virtual Machines

Code files -> CI

Code file can be pulled from

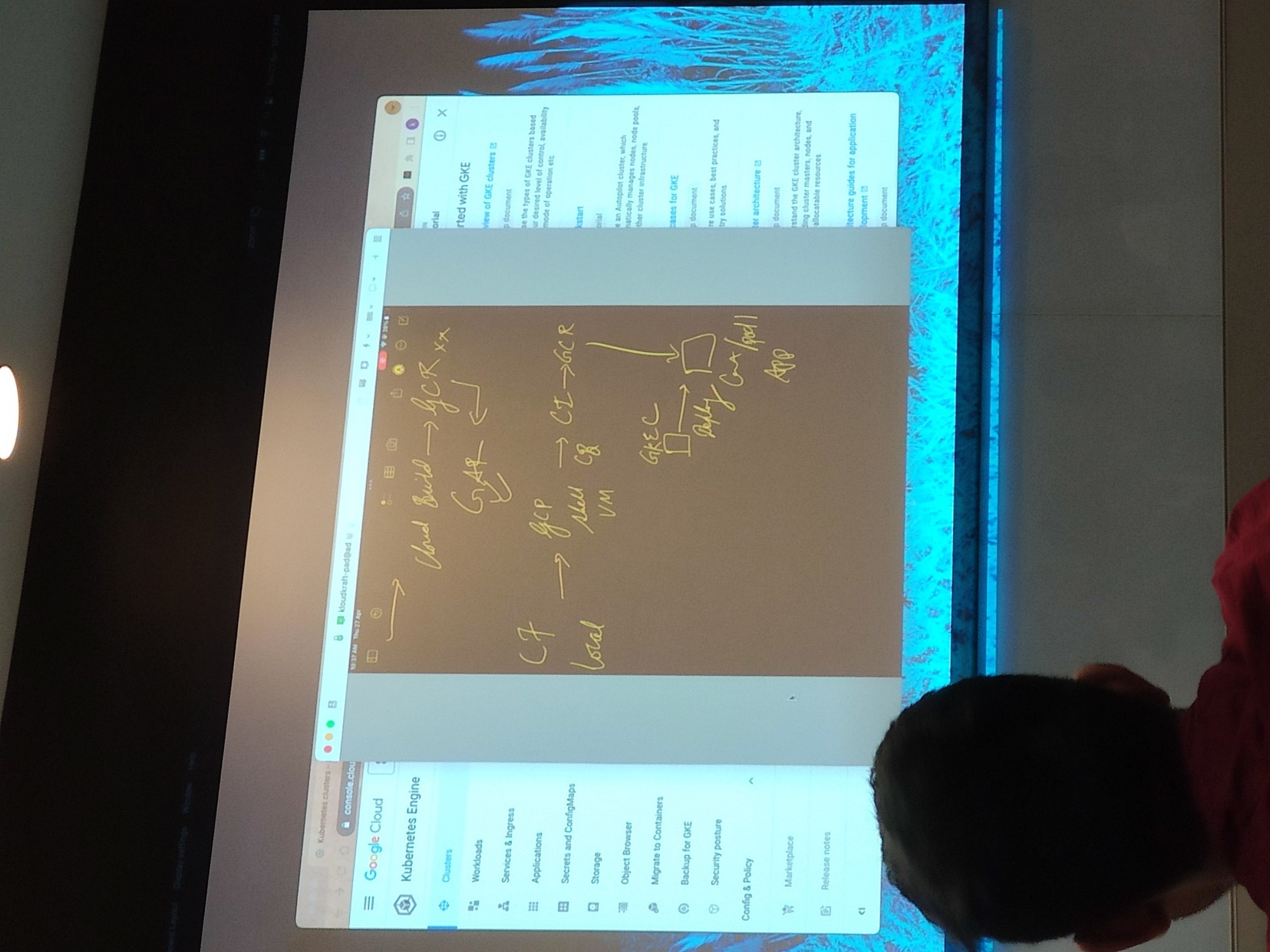
1. Docker -> DRegistry
2. Git -> git
3. Cloud build -> GCR

You can develop apps locally

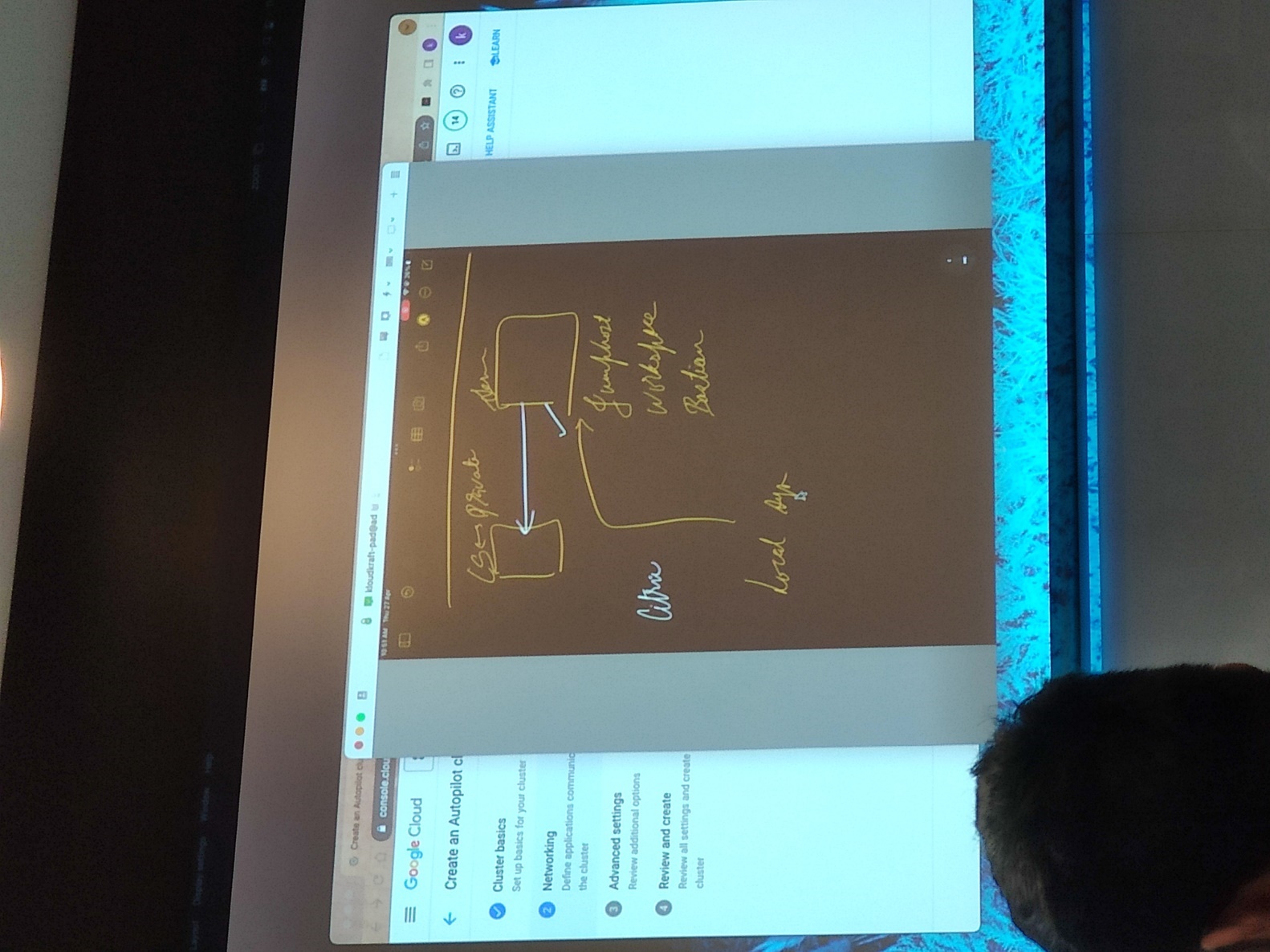
Then push using GCP or shell or VM

Using CB the next step is CI

Then to GCR

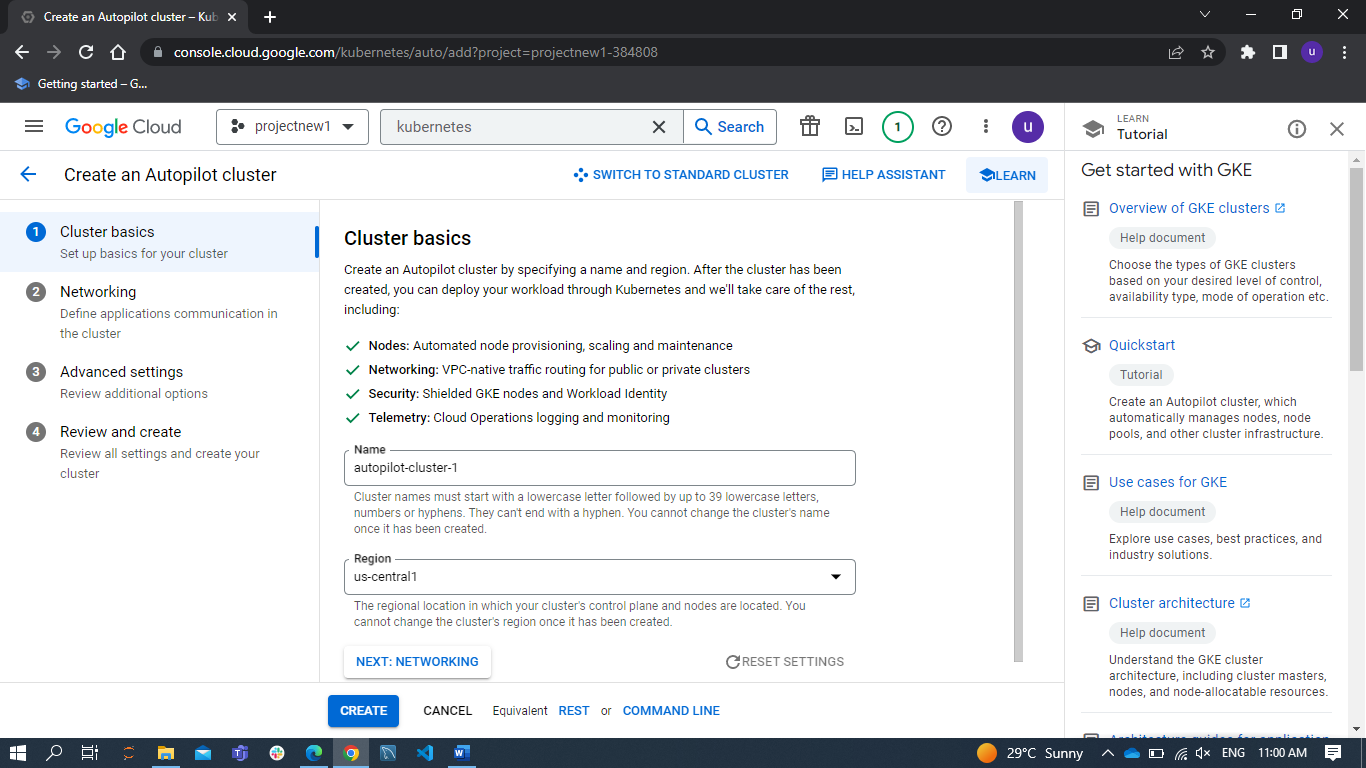


People from their local systems try to access the system at remote location using the jumphost as shown below:

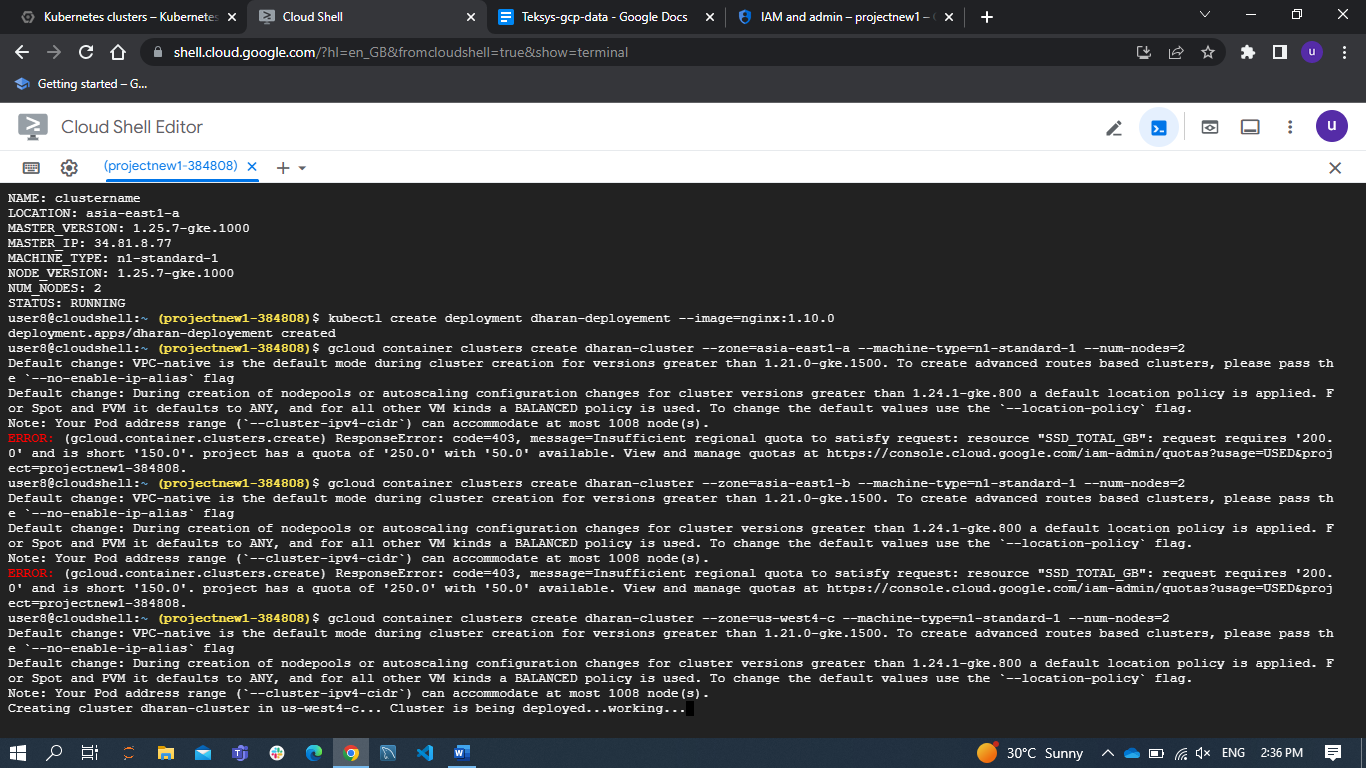


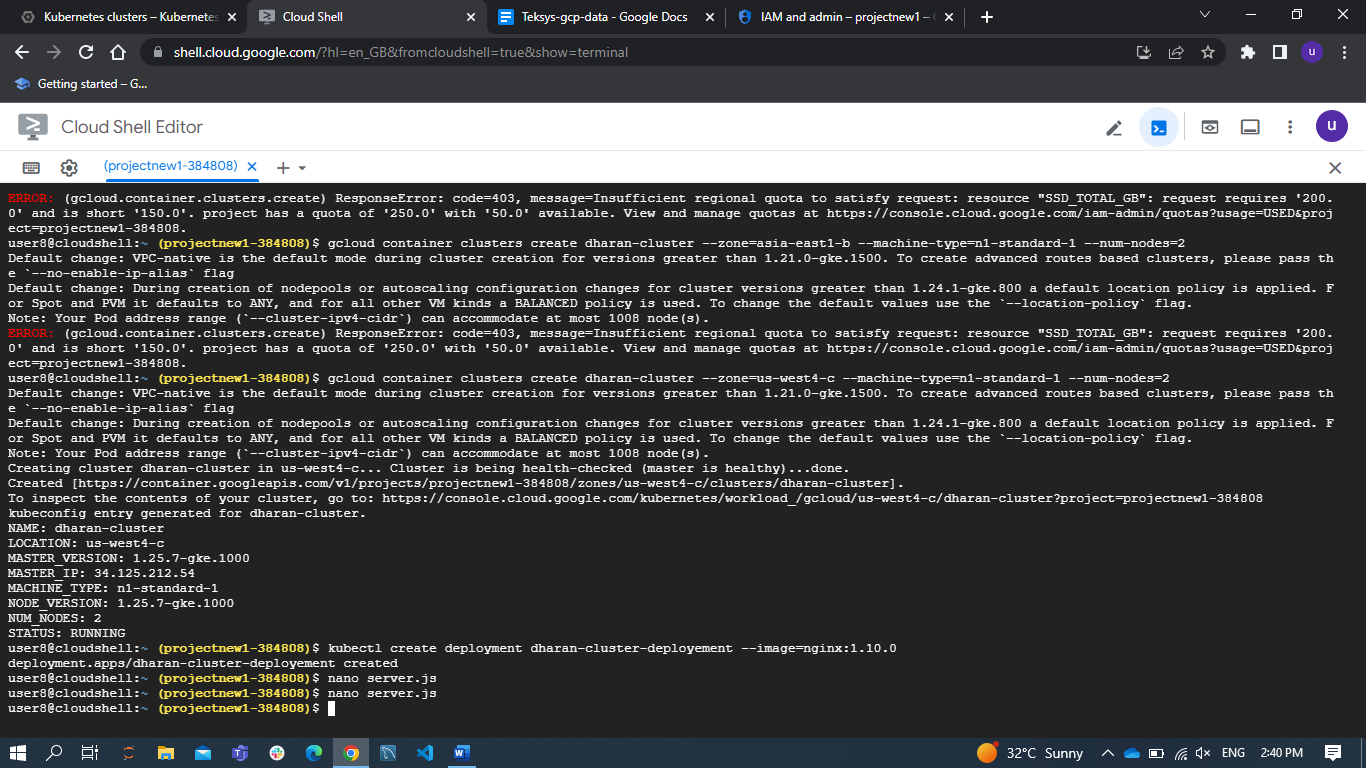
Cluster Creation WorkFlow:

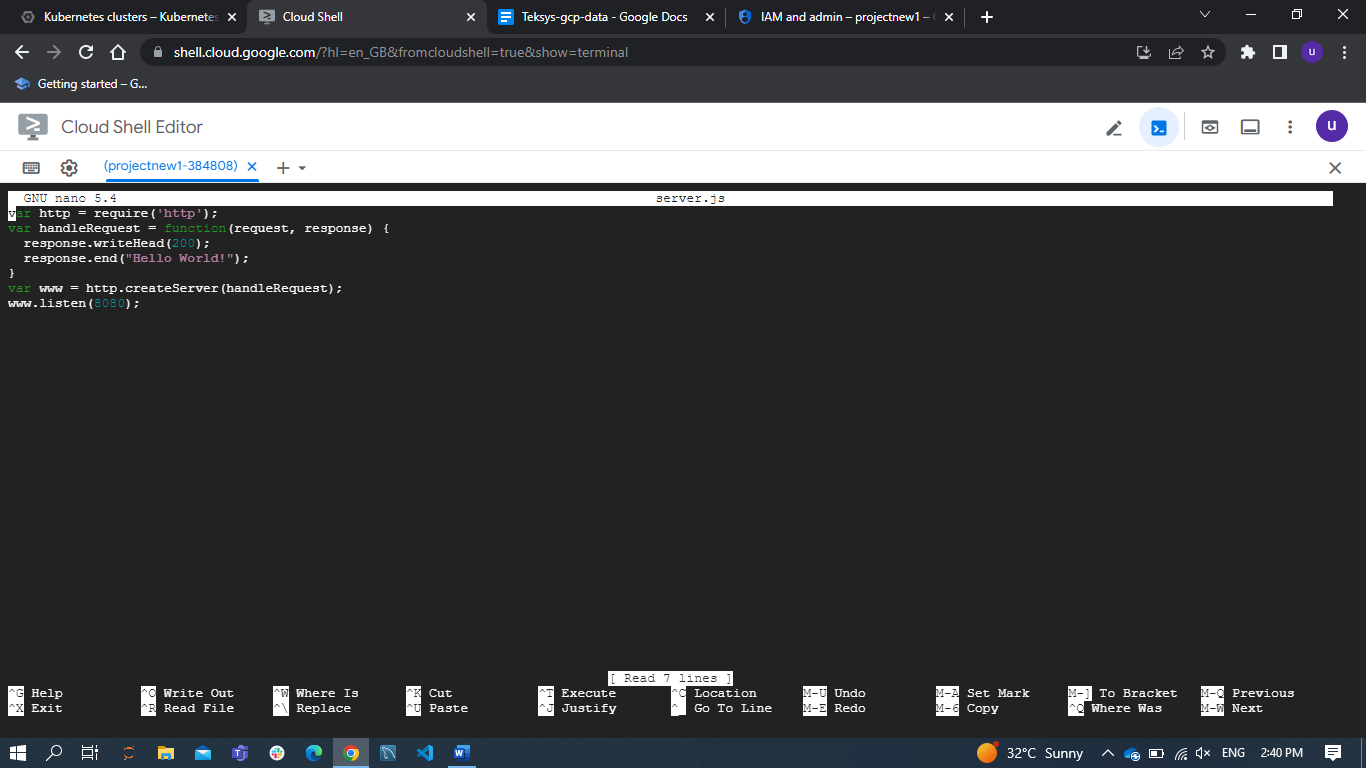
* + 1. GUI Method:



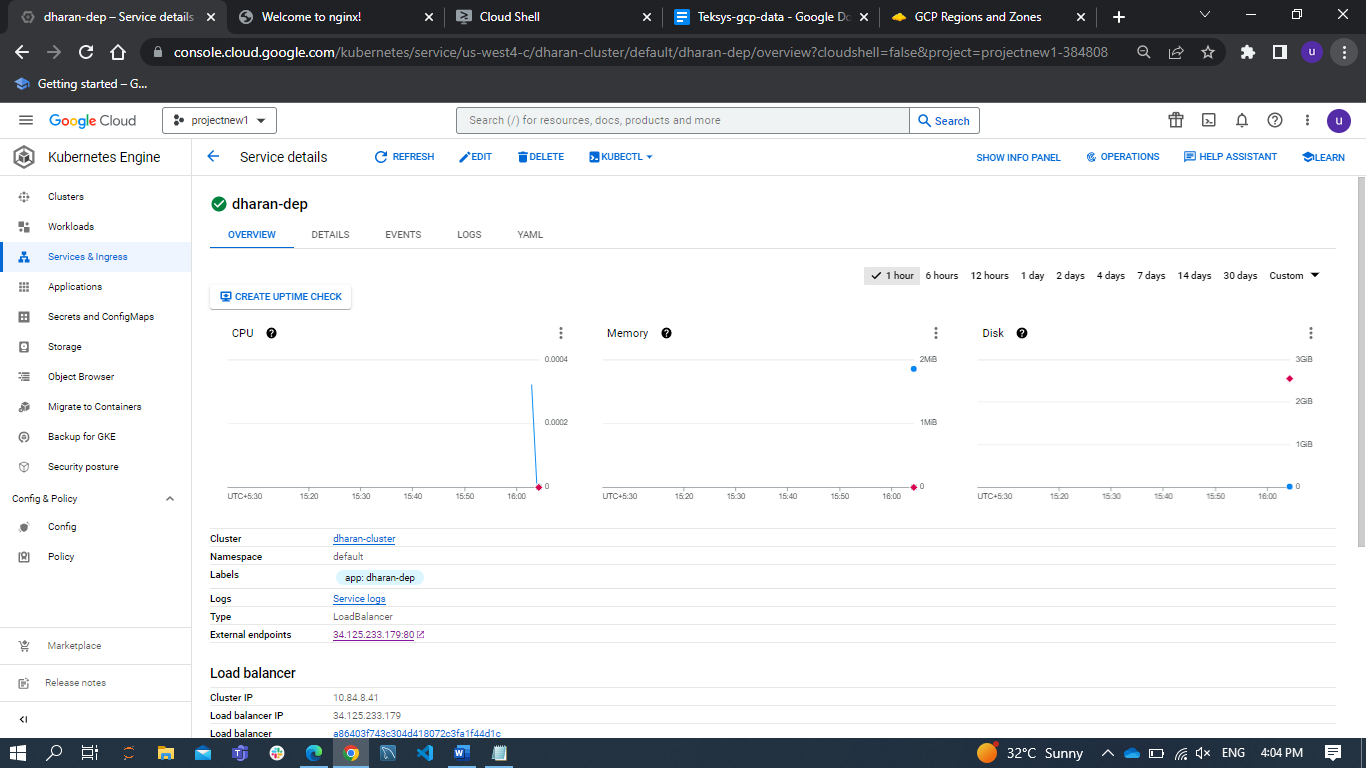
* + 1. CLI Method:

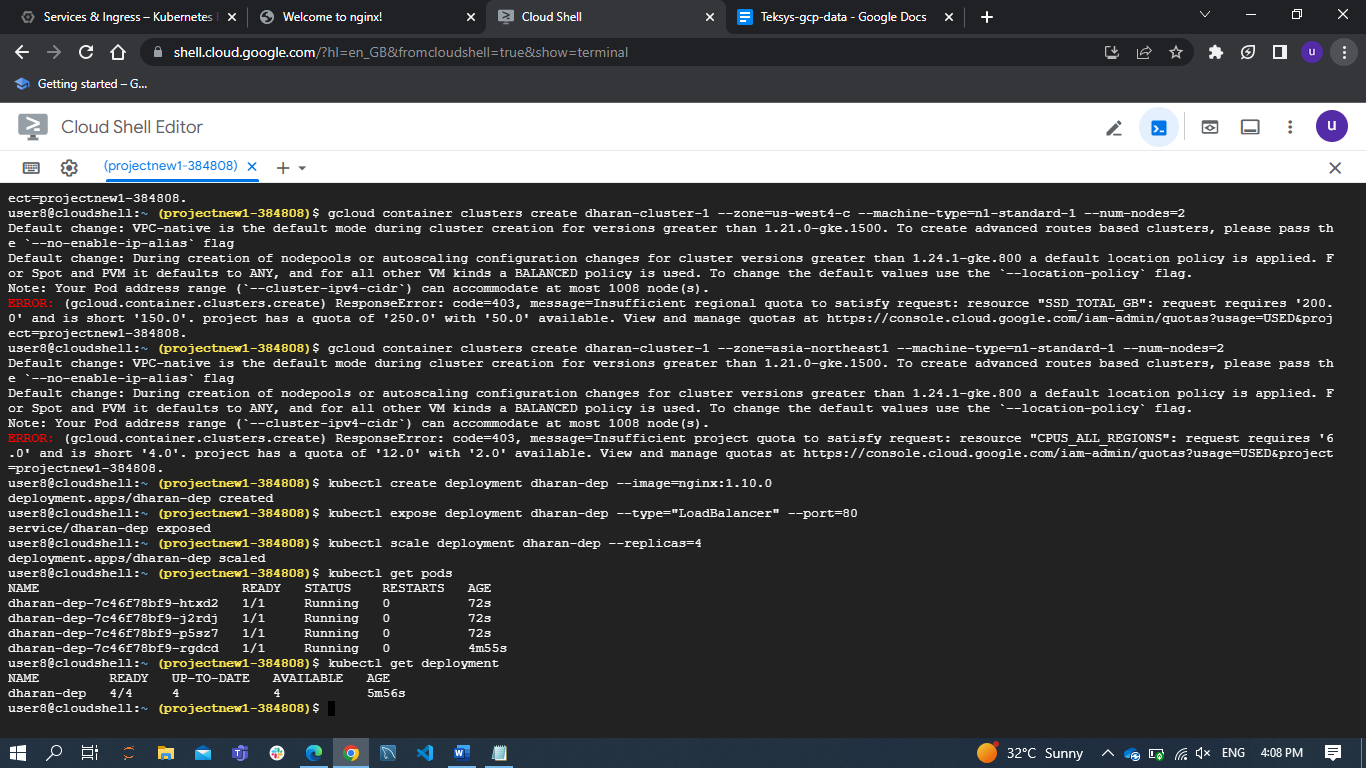




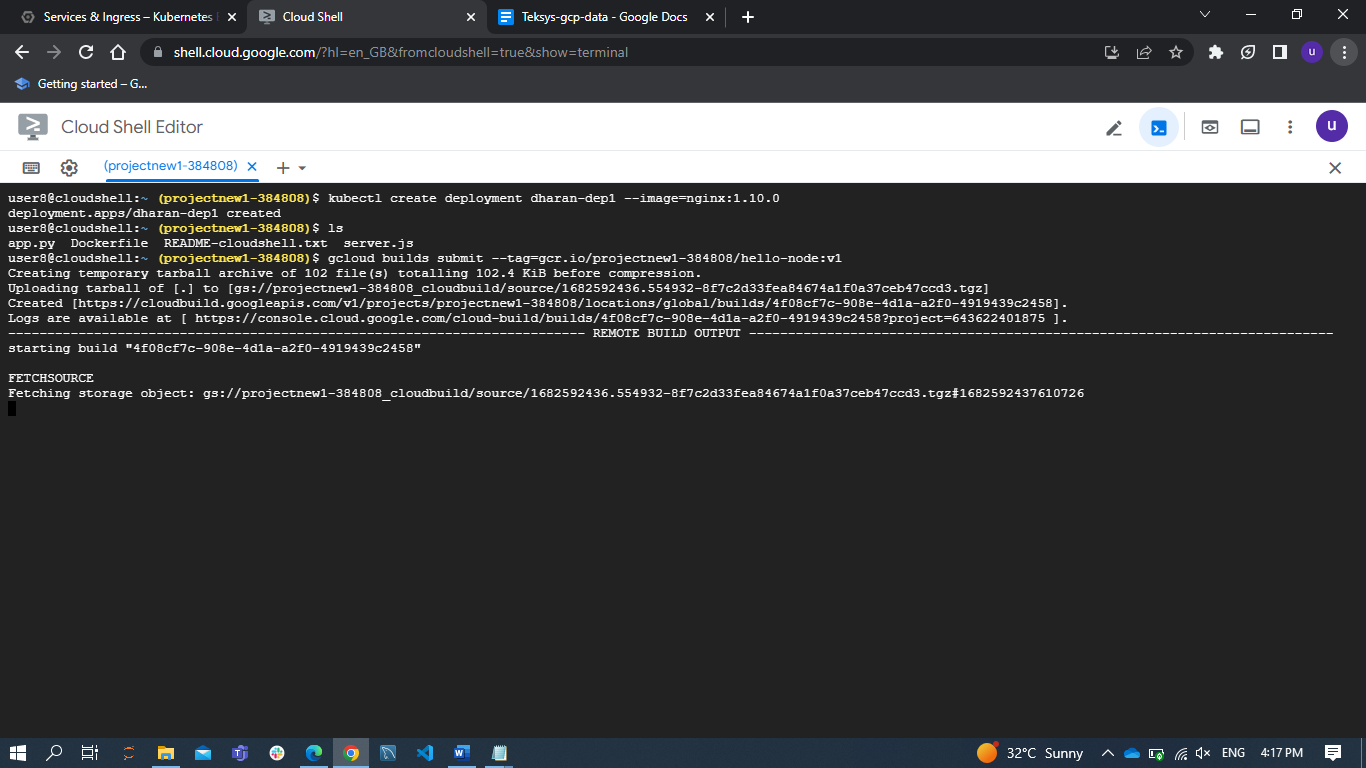


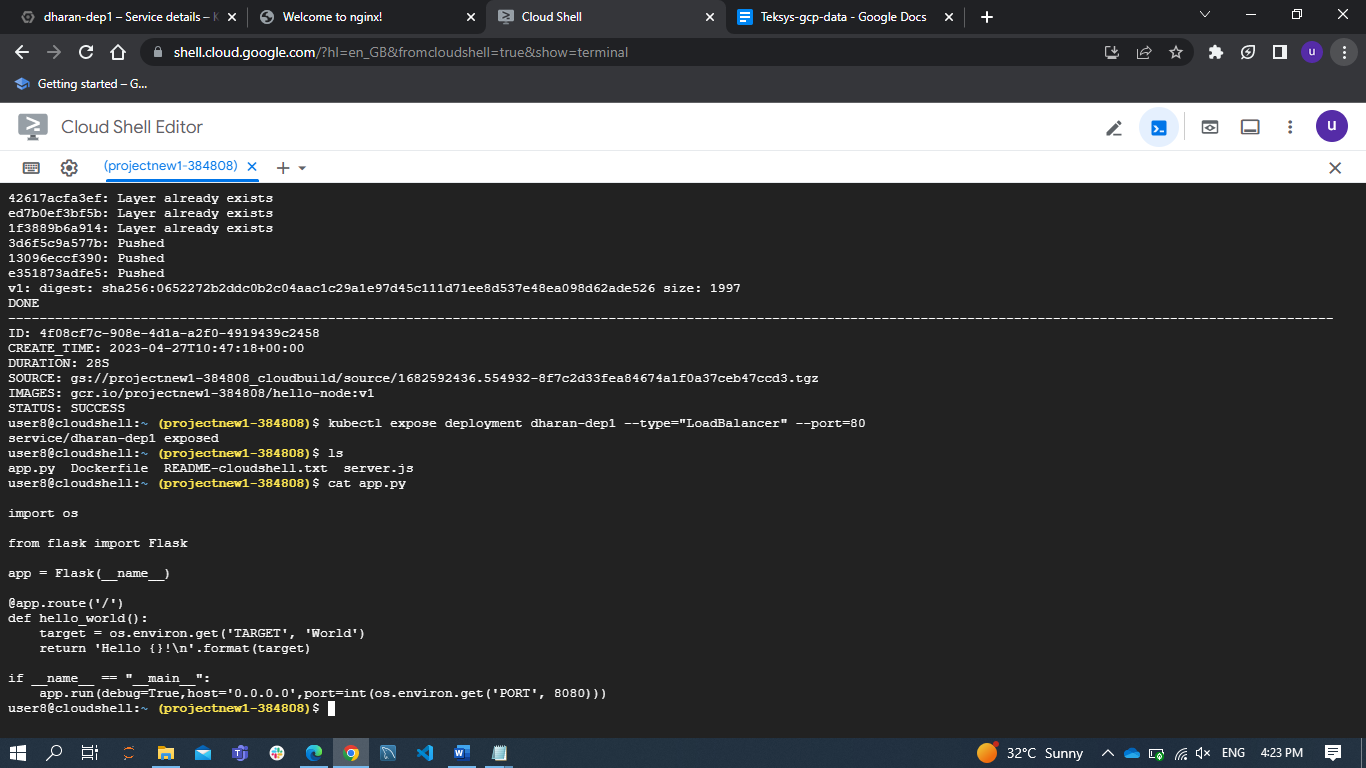






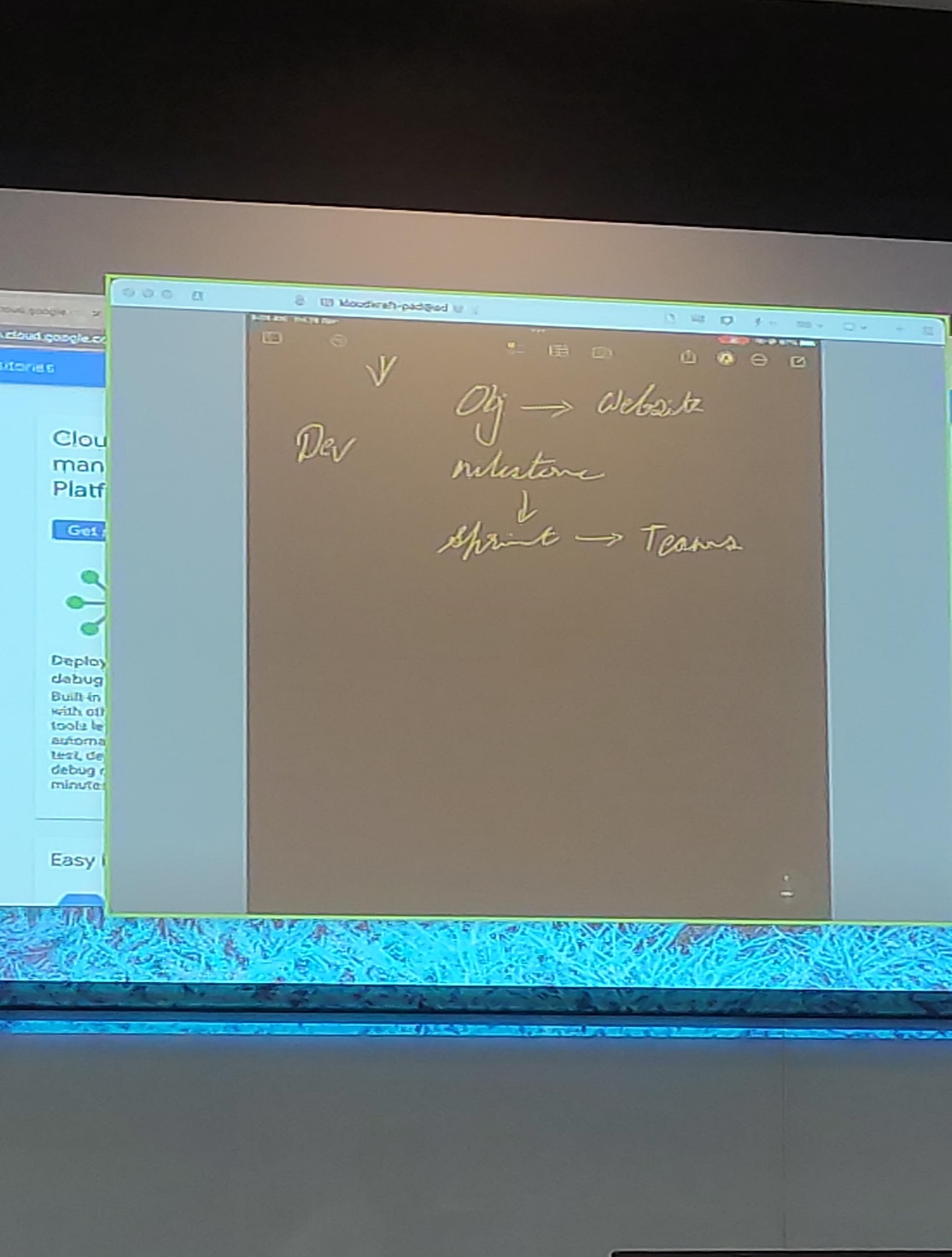
Docker File:





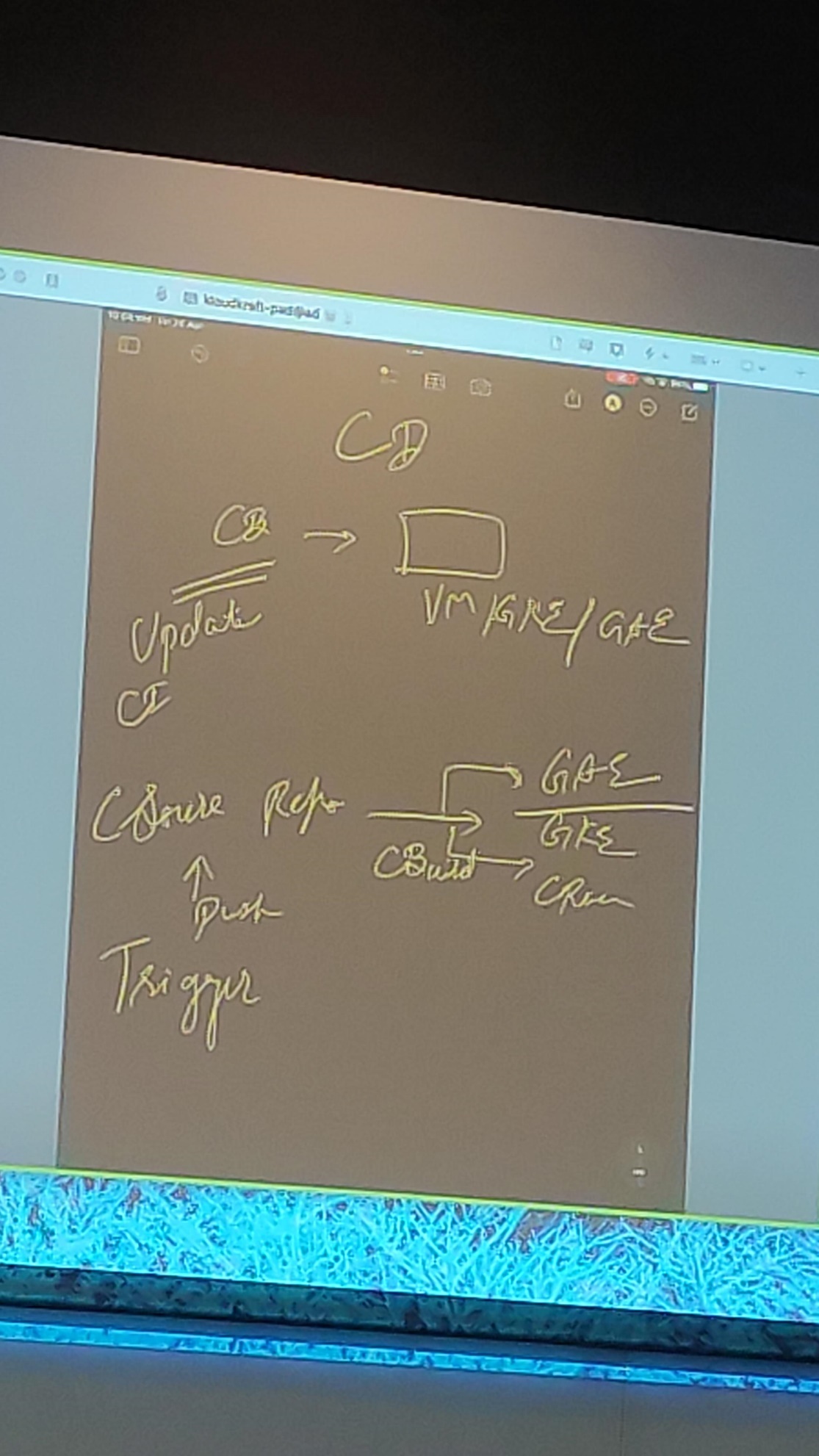
*28th April 2023*

Milestones:

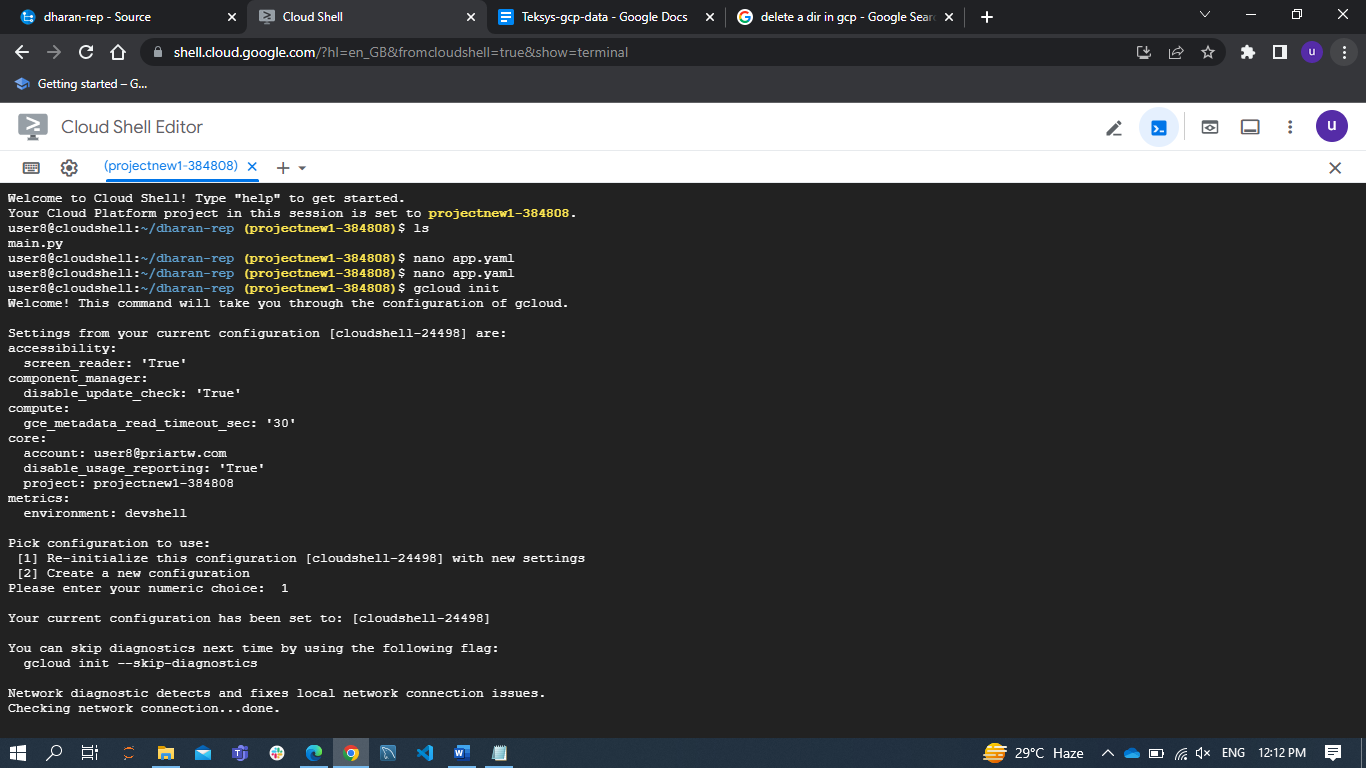


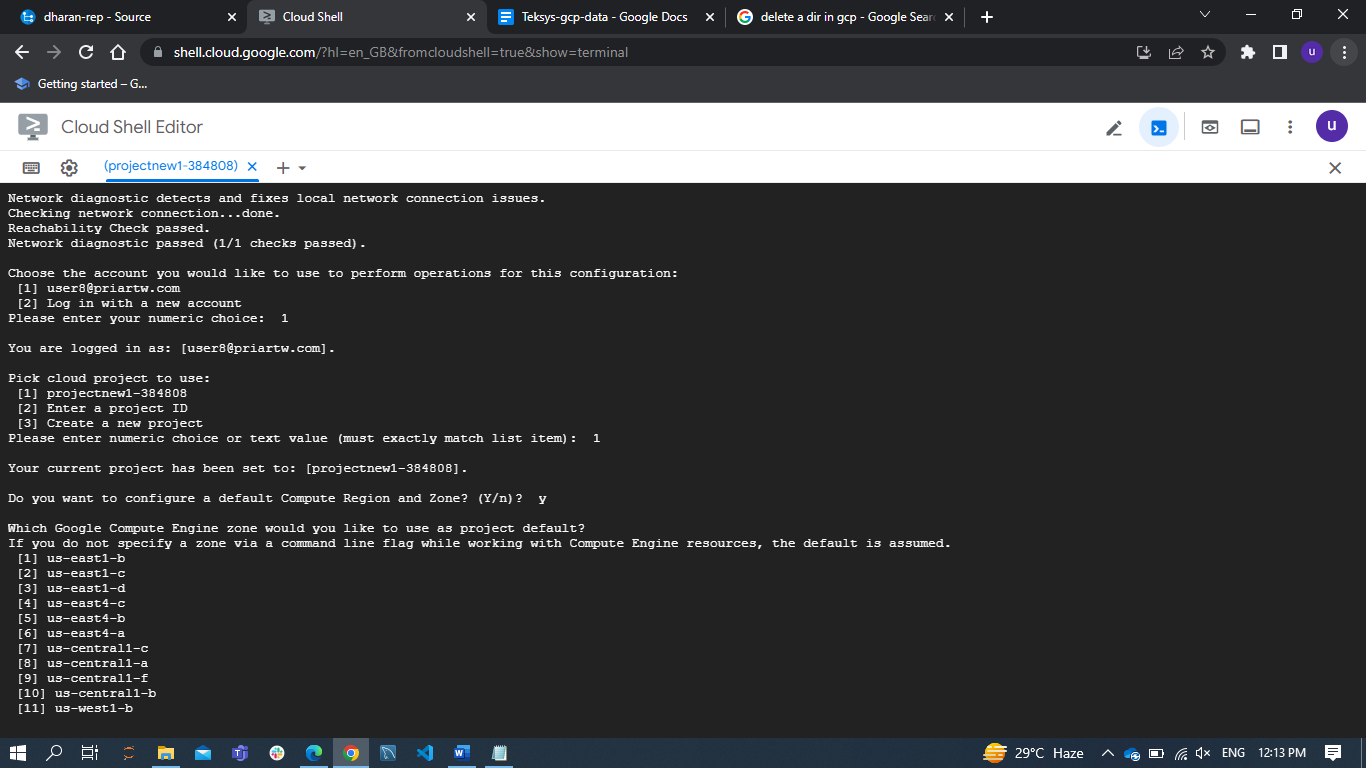
A picture containing text

Description automatically generated



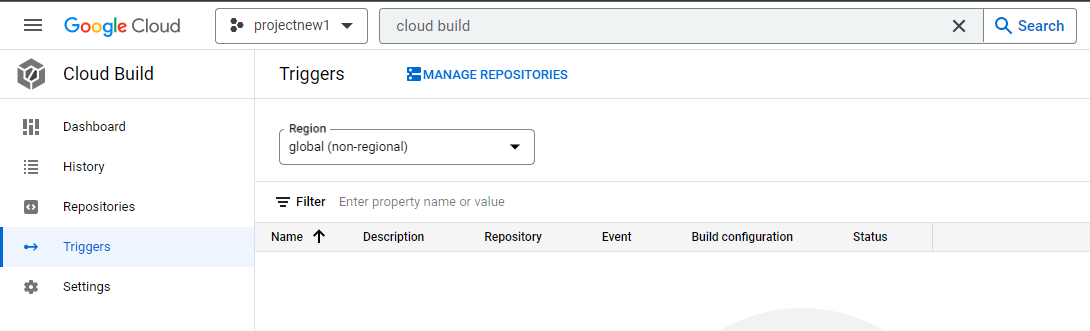
Process Screenshots:

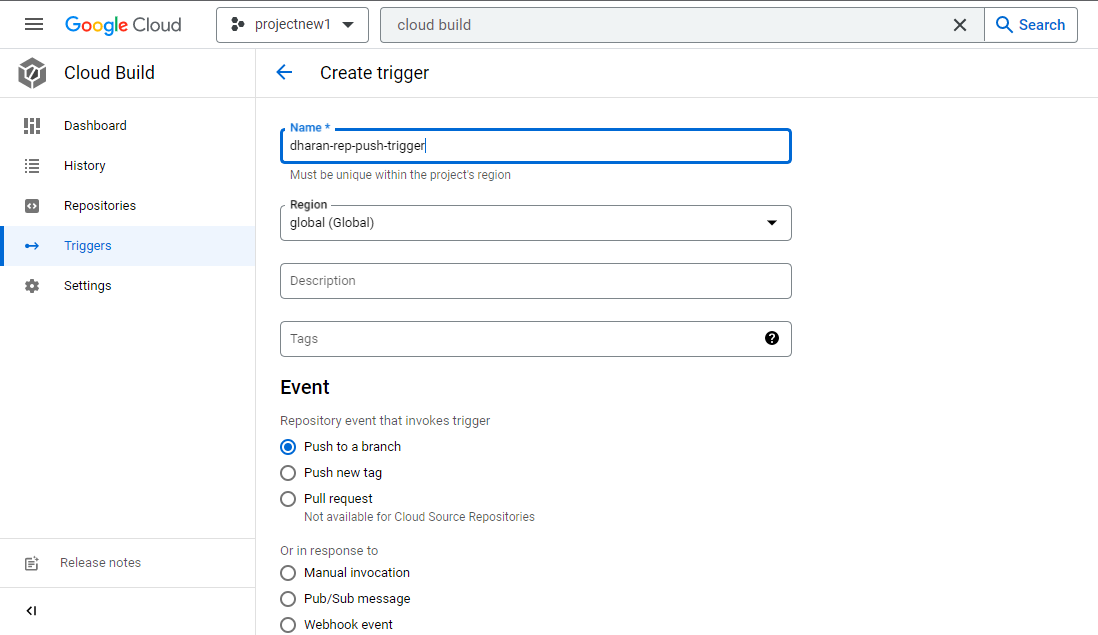


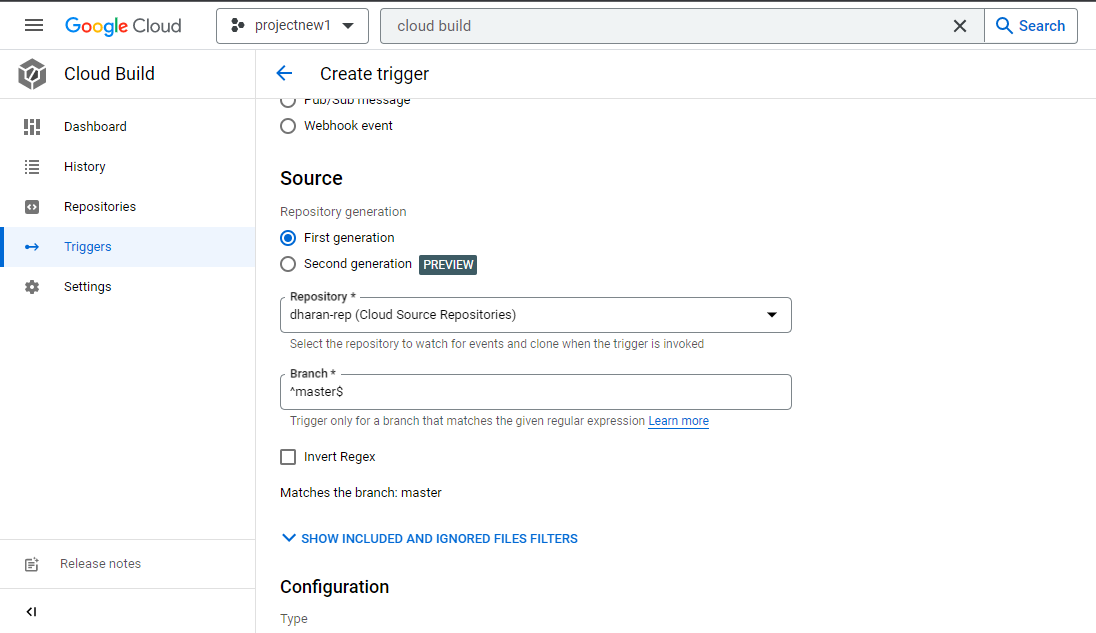


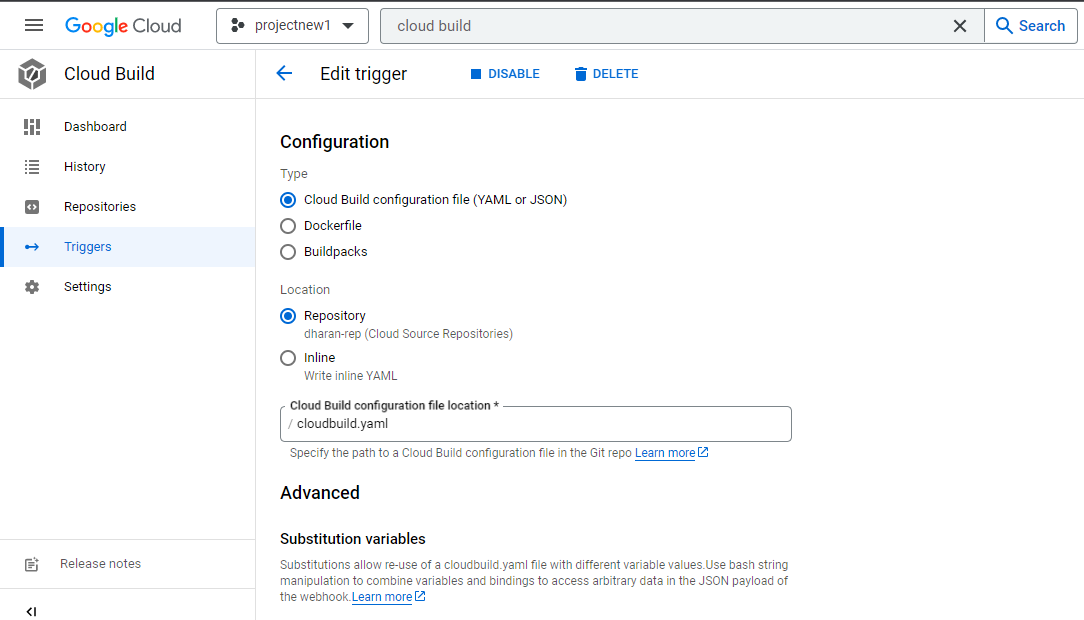


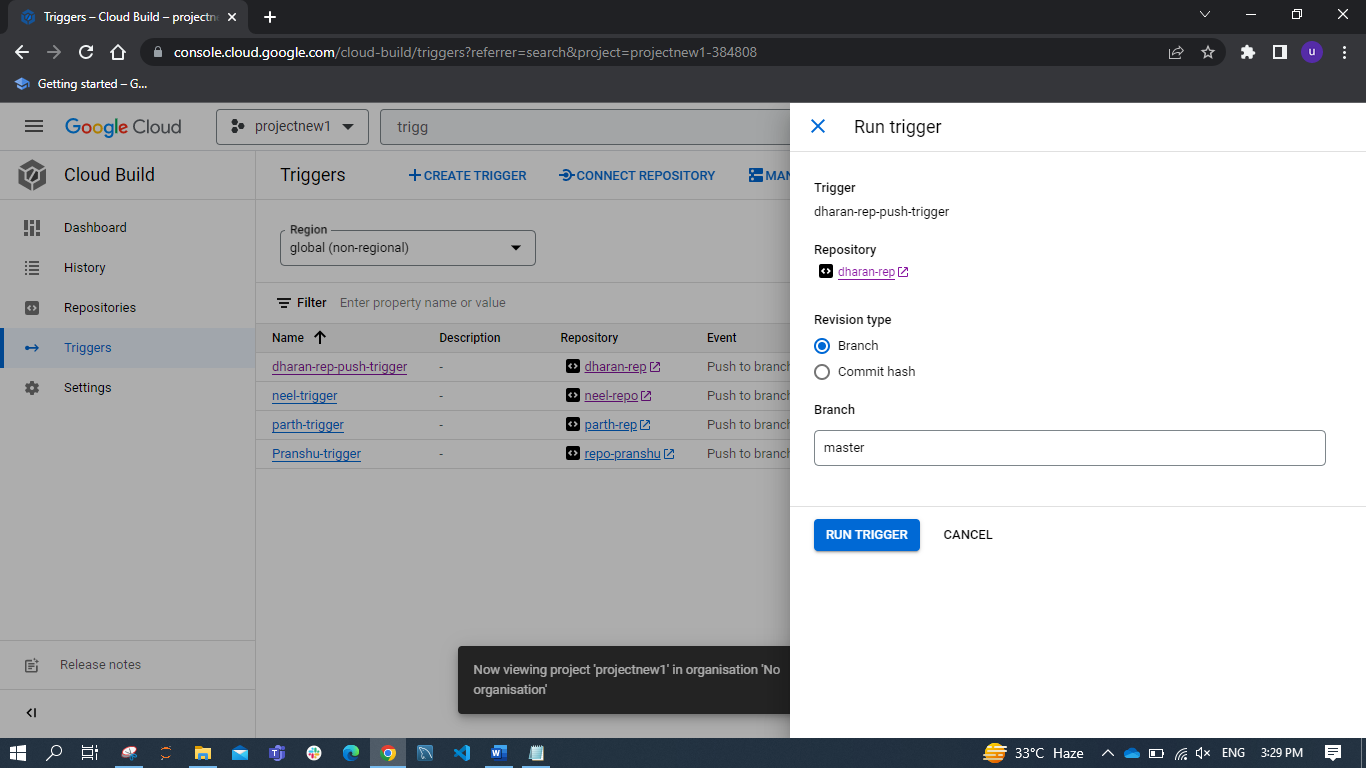


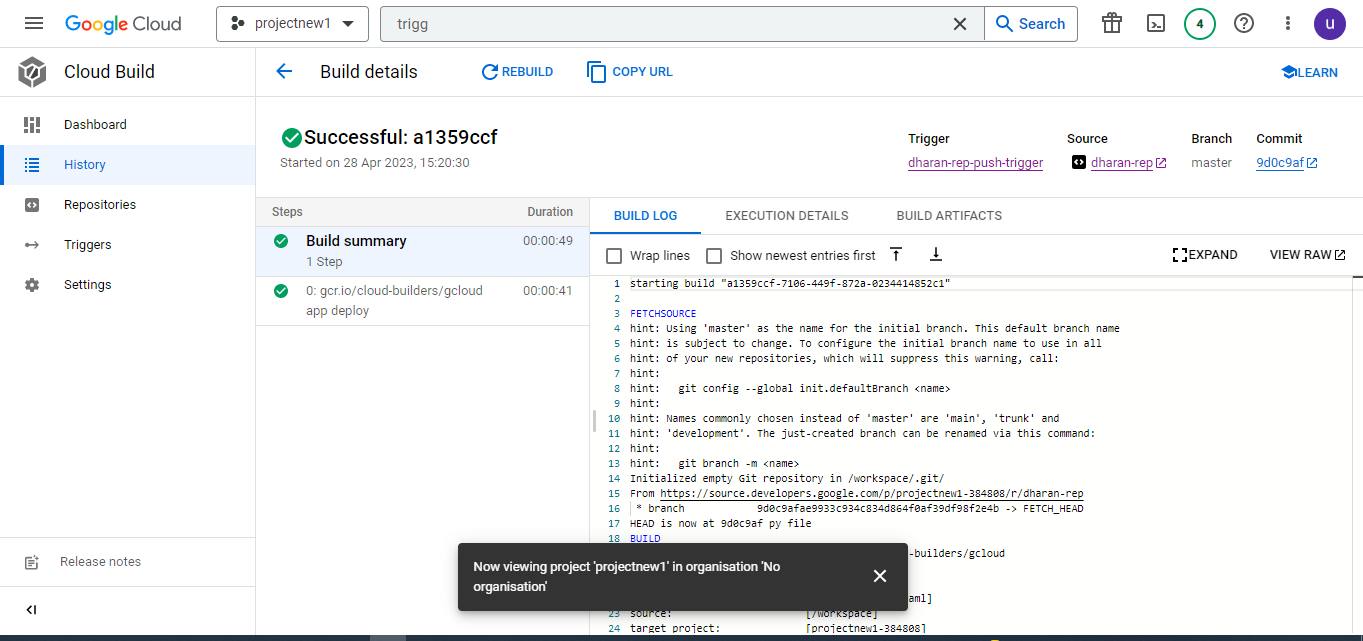


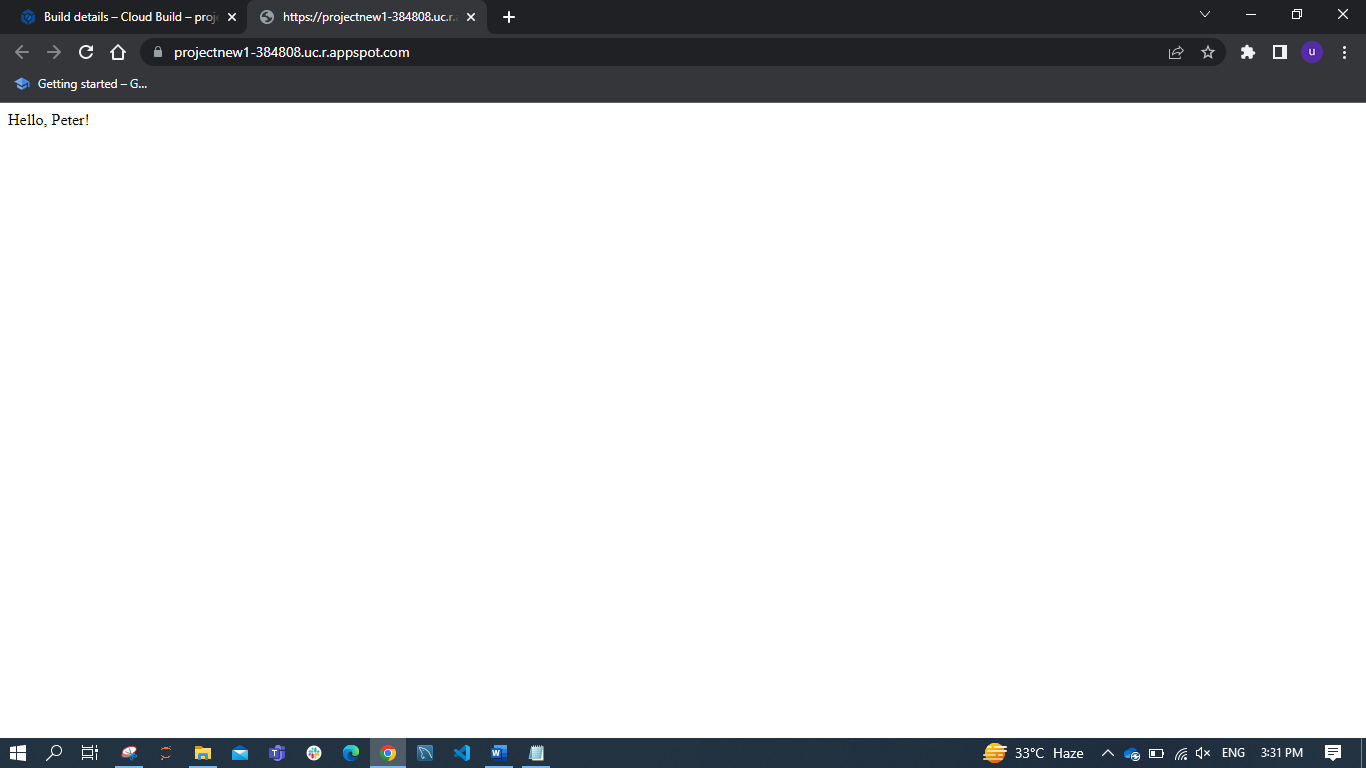












Finally the app is hosted