Pluto7

How to Setup Demand ML Solution on Google Cloud with Cortex





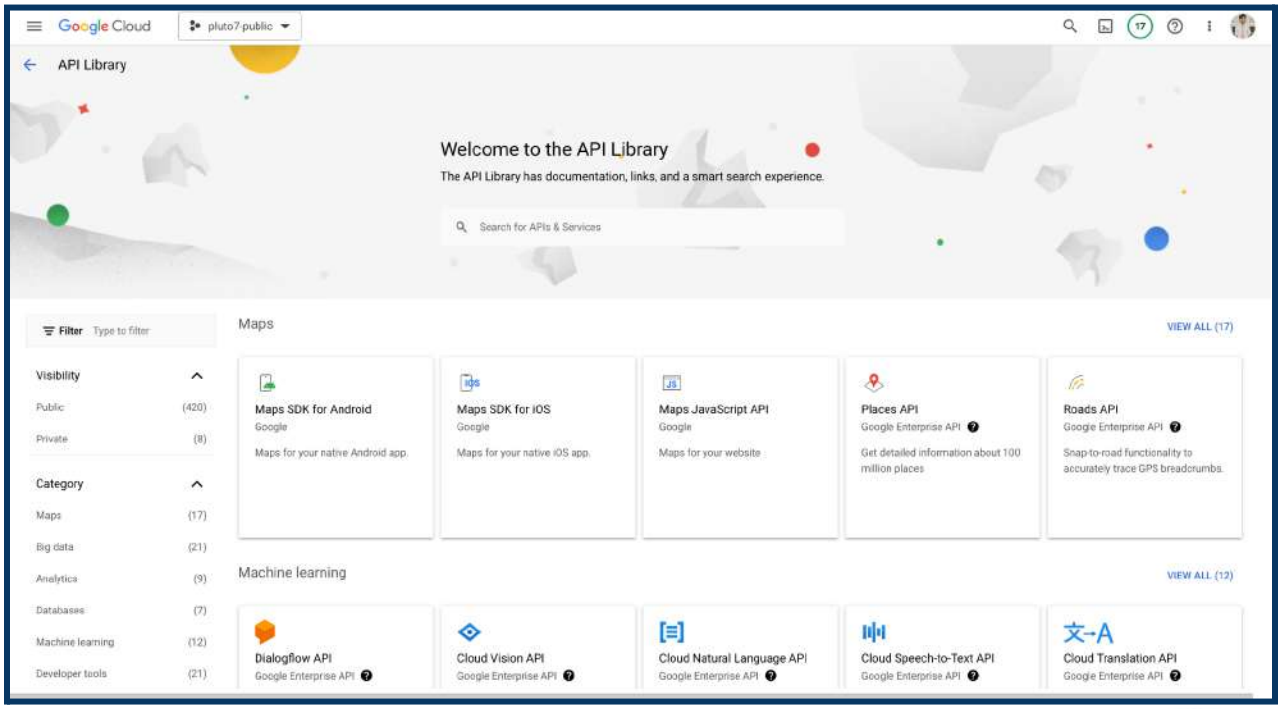


Pluto7

Pre-installation requirements:

Visit the **APIs & Services** section in the Google Cloud Console and enable all the following APIs:

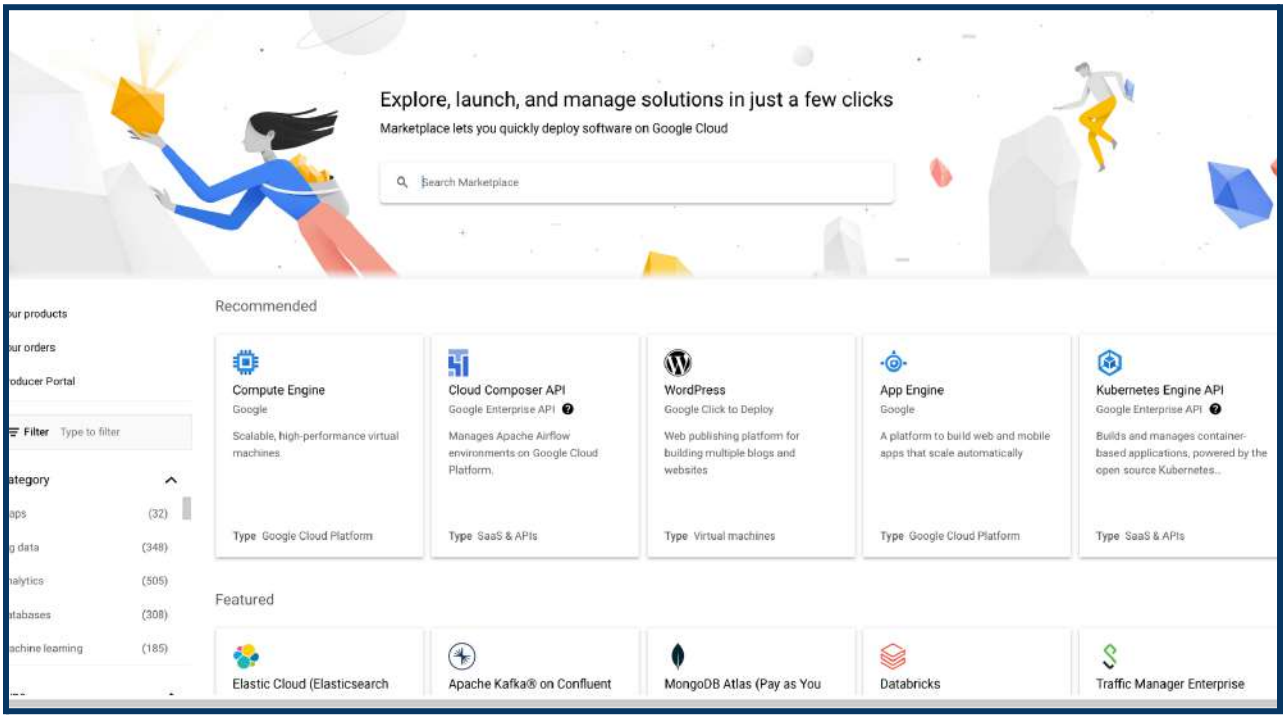
- Compute Engine
- Cloud Storage
- Container Registry
- Vertex AI
- Google Cloud Storage JSON
- Firestore
- BigQuery



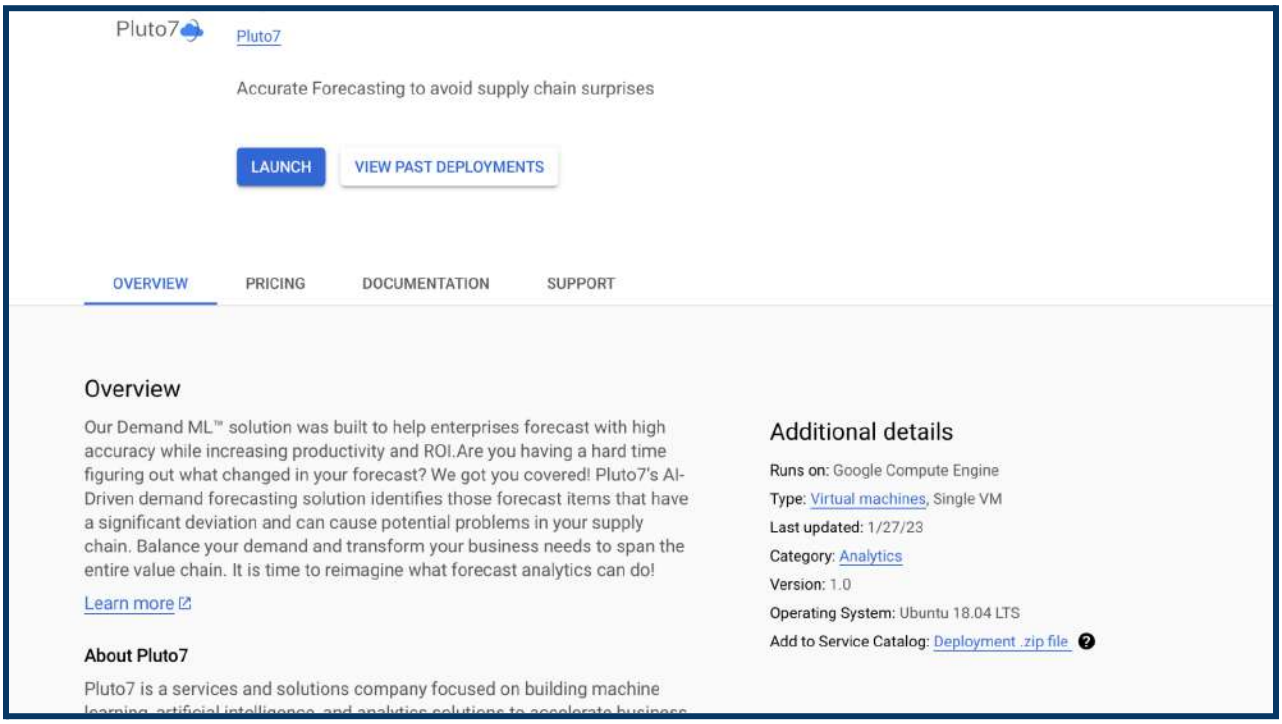
Installation Steps:

1. After enabling all the **APIs** mentioned above, go to [GCP Market Place] and search for **Demand ML**.

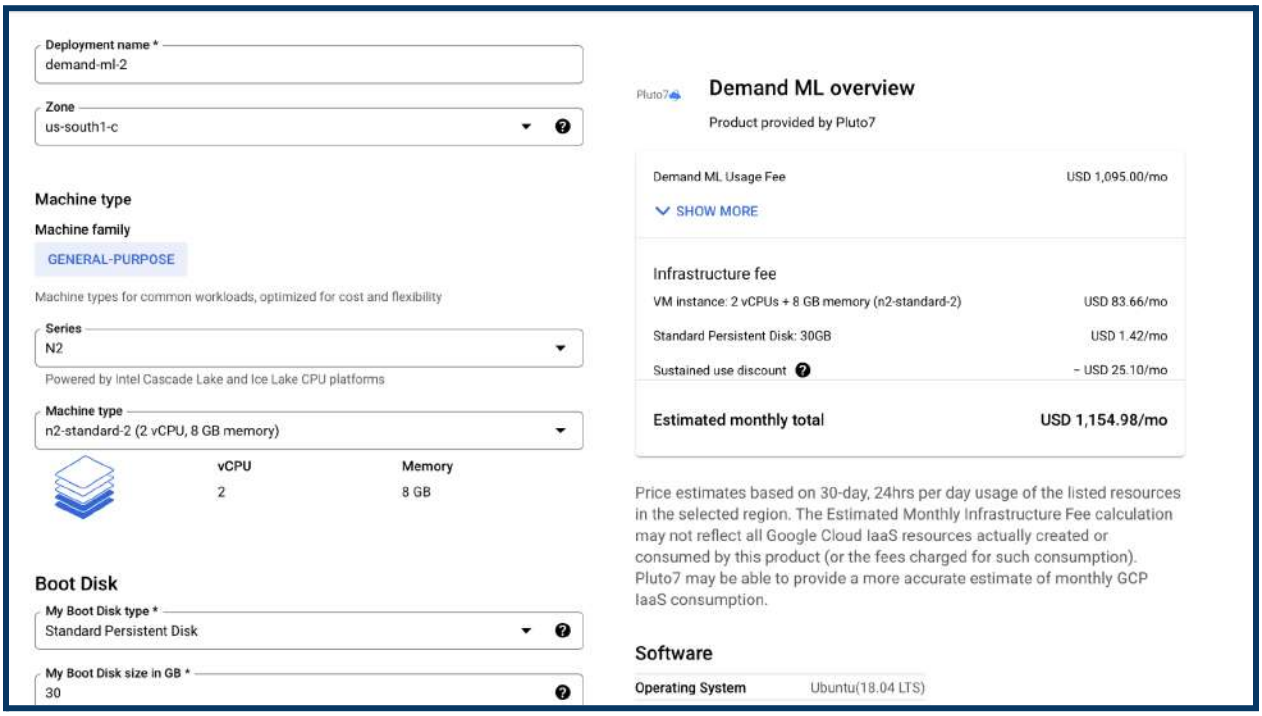
Once you have selected Demand ML, you will be redirected to the **Demand ML Solution home screen**. (the home screen looks like the image shown below).



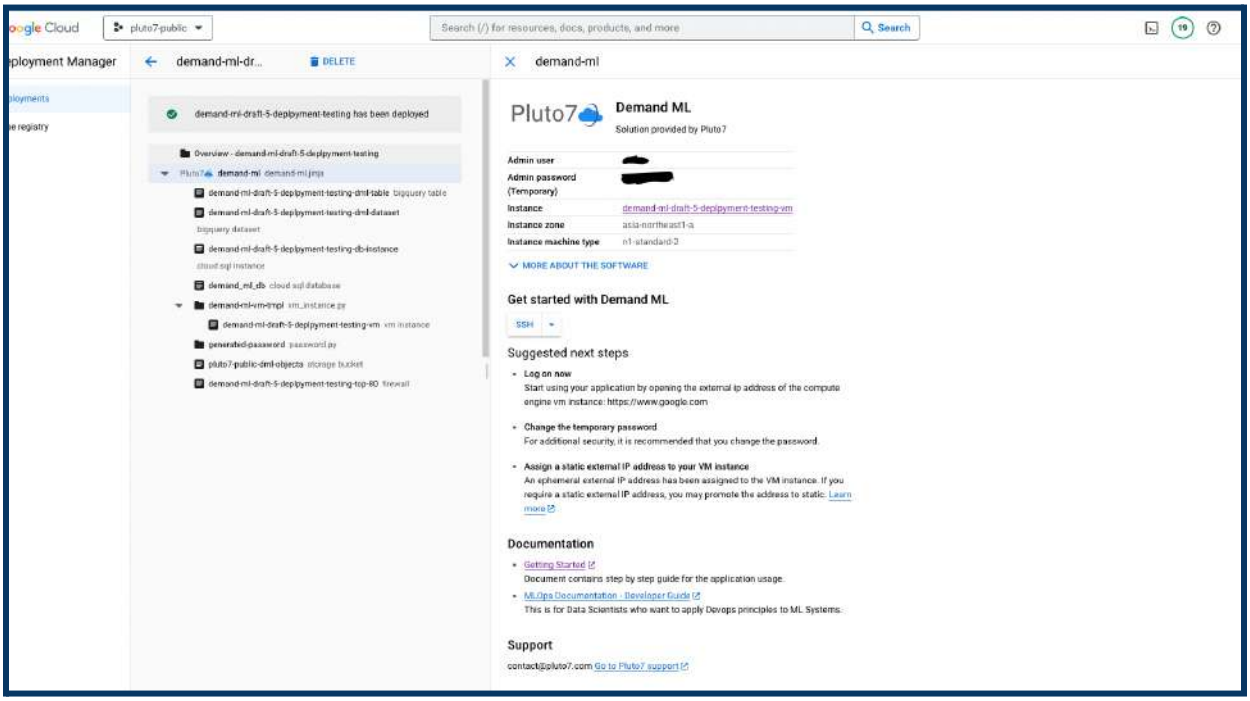
2. Next, click on the **LAUNCH ON COMPUTE ENGINE** button. It will redirect you to the deployment page.



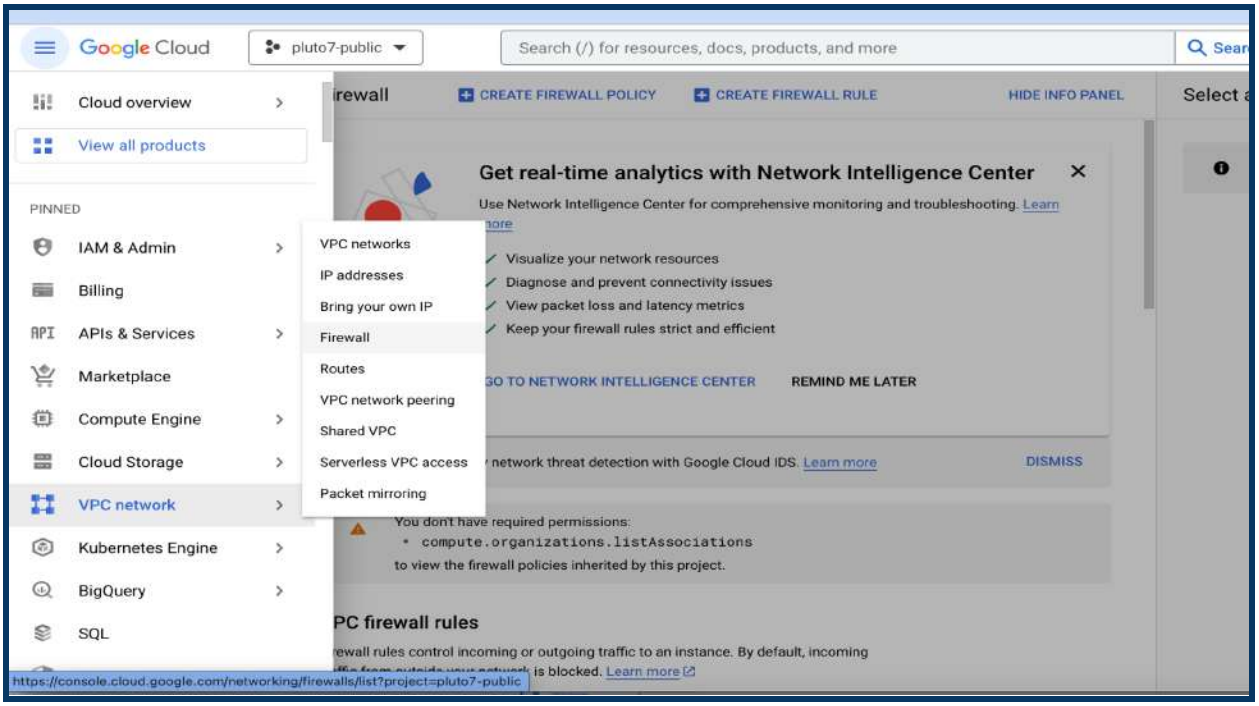
3. On the **Deployment page,** scroll to the **bottom of the page,** and to deploy the solution, **click on the deploy button**. The page will be redirected to the deployment manager page, and it will show all cloud services used for the solution. Wait until the deployment gets completed.



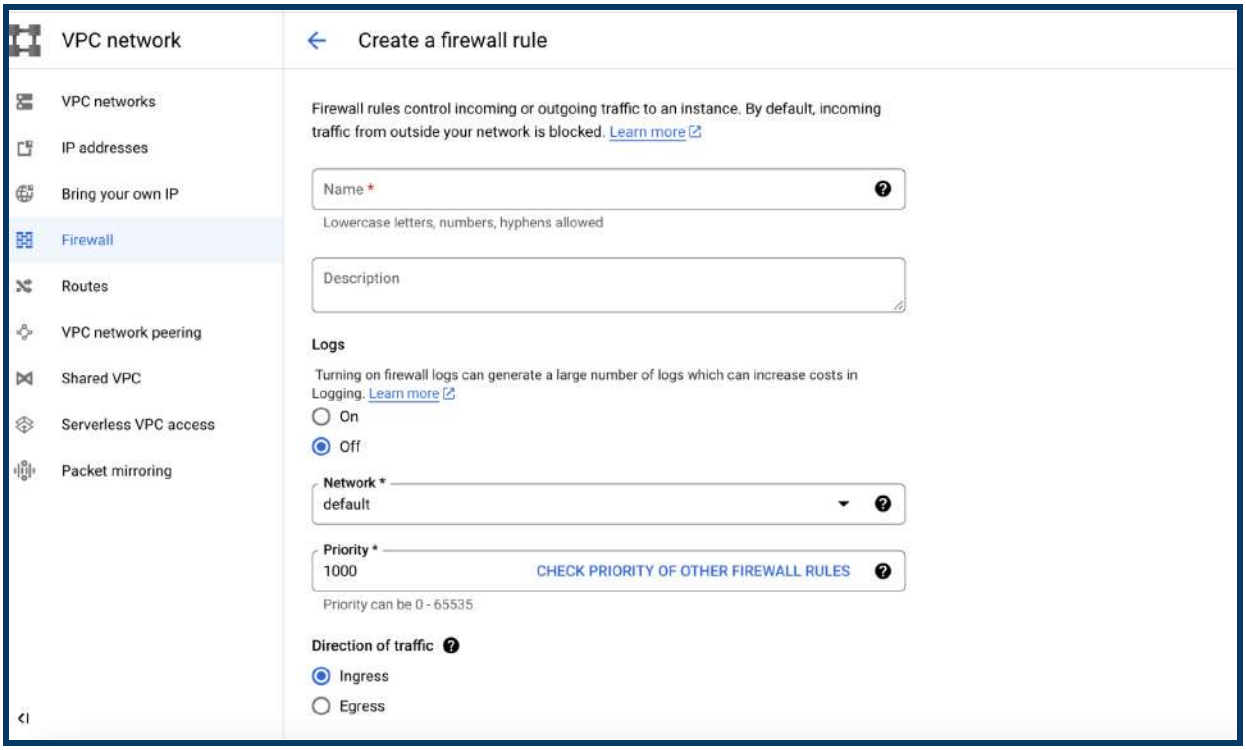
4. Wait till the deployment gets completed.



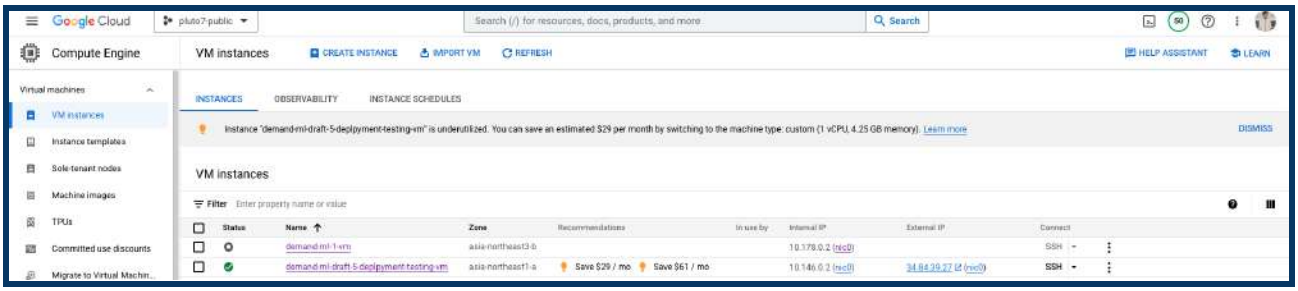
5. In the GCP console, go to **VPC Networks** and click on the **firewall section**.



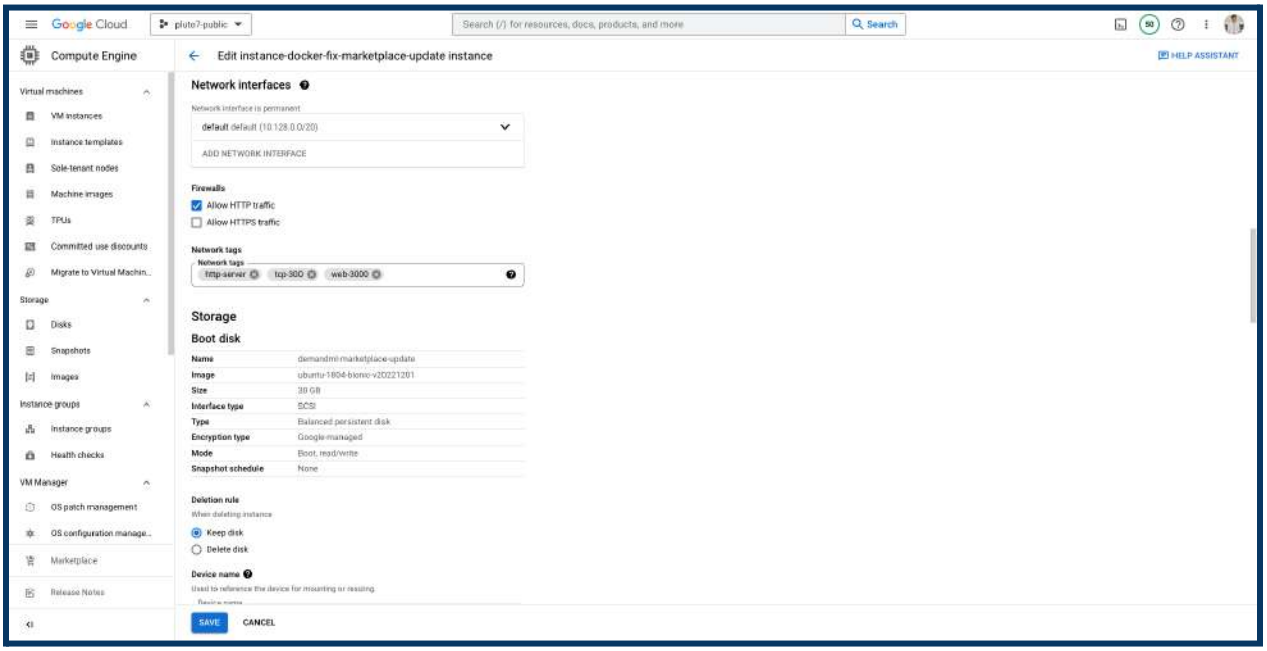
6. Click on '**Create a Firewall Rule'** and create a rule to allow internet traffic by enabling a TCP connection. Enter **0.0.0.0/0** in the IP range, select any port, select **Ingress** as the direction and enter **3000** as the TCP port. Enter the network tag with the name **allow-3000**. Click on '**Create'** to create the firewall rule.Now repeat the same step keeping everything the same except the direction.Instead of ingress , enter **egress** as the direction and Click on '**Create**'.



7. After deployment gets completed, go to the **compute engine** **home page**. Here You can see the deployed instance.



8. Select the **Instance** which was deployed and **STOP it**. Click on **edit**. Make sure to add the tags created in the firewall section and click on **save**. Scroll down to **API and identity management** and select **Allow full access to all Cloud APIs. Click on save and RESTART the Instance.**



9. Now select the deployed instance and SSH into the VM.

```
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 5.4.0-1098-gcp x86_64)

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

  System information as of Tue Feb 21 04:48:47 UTC 2023

  System load:  0.75              Processes:             119
  Usage of /:   48.2% of 28.89GB  Users logged in:       0
  Memory usage: 12%               IP address for ens4:   10.128.15.216
  Swap usage:   0%                IP address for docker0: 172.17.0.1

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

14 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

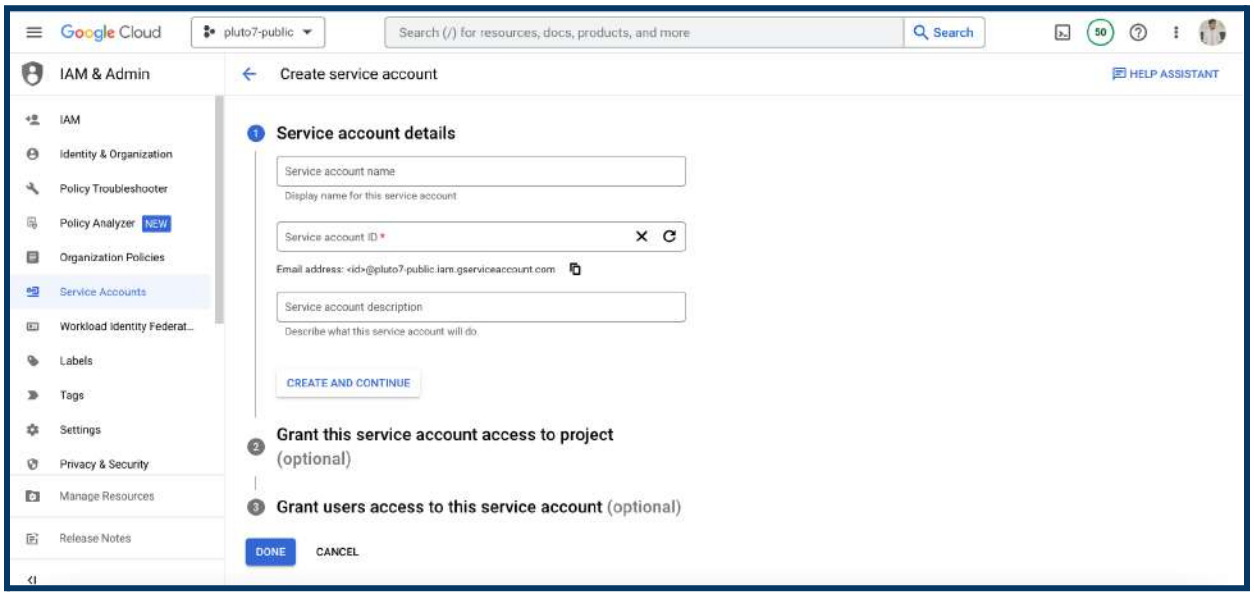
New release '20.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


*** System restart required ***
Last login: Mon Feb 20 17:01:41 2023 from 35.235.244.32
alok@instance-docker-fix-marketplace-update:~$
```

10. After this, create a service account with the following permissions:

AI Platform Service Agent
App Engine Admin
App Engine Code Viewer
BigQuery Admin
BigQuery Data Editor
BigQuery Job User
Cloud Build Editor
Cloud Datastore Owner
Cloud Functions Admin
Cloud SOL Admin
Container Registry Service Agent
IAP Policy Admin
Project IAM Admin
Service Account Admin
Service Account User
Service Usage Admin
Storage Admin
Vertex AI Custom Code Service Agent

and download the JSON key file of the service account.



11. Go To **IAM** & **Admin** in the GCP cloud console and click on the **checkbox** on the top right which says **"Include Google-provided role grant**." Once enabled, go to the **filter search box** and search for

"**@gcp-sa-aiplatform-cc.iam.gserviceaccount.com**"

By default, this service account has only the "Vertex AI Custom Code Service Agent" role assigned.

Edit the roles and add the following roles along with the existing ones to this service account :

- Cloud Datastore User
- BigQuery Data Editor

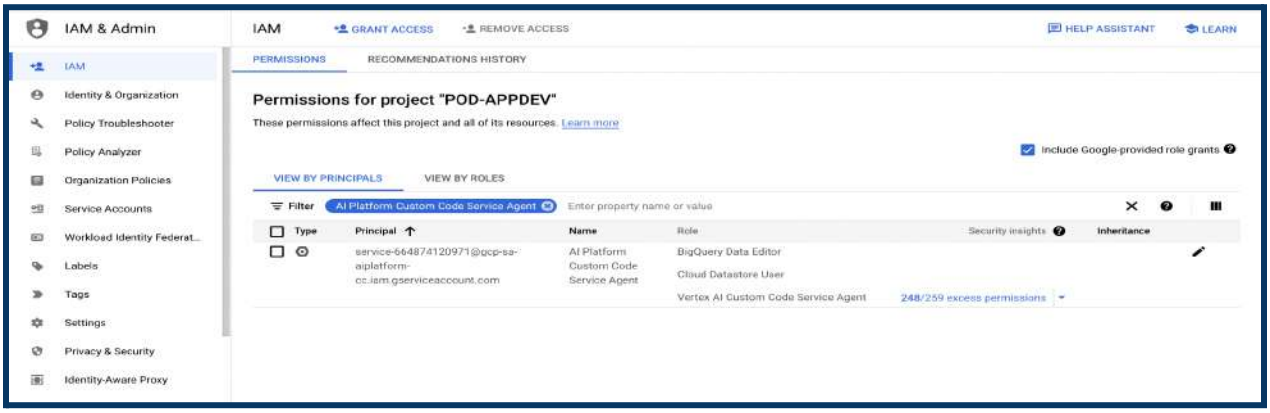
Note: This service account should have these three roles assigned to it

Next , You will find the cloud build default service account, which ends with, cloudbuild@gserviceaccount.com.

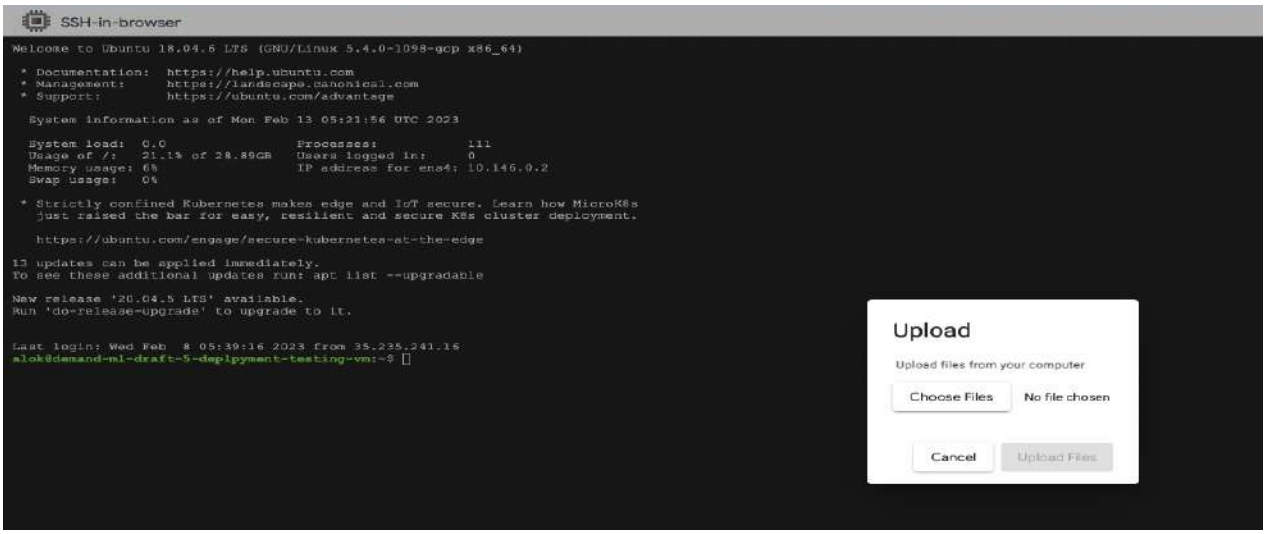
Select this service account and add the following roles to it:

- App Engine Admin
- BigQuery Data Editor
- BigQuery Job User
- Cloud Build Service Account
- Service Account User
- Storage Admin

Note: This service account should have these three roles assigned to it



12. Upload the service account JSON key file to the compute engine VM instan



13. In the terminal, run ***cd /home/ubuntu*** and go to /etc/environment using **sudo** and replace the GCP environment variables by updating the location of the service account key file, which was uploaded in the earlier step with Its absolute path.

```
PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin"
NODE_ENV=production
PROJECT_ID=
GCS_BUCKET=
BQ_DATASET=
COLLECTION=
GOOGLE_APPLICATION_CREDENTIALS=path/to/your/service_accont_key_file.json
```

14. Now run the "**ls**" command. You will find three " **.sh** " files. Run the following commands one at a time to make them executable,

- sudo chmod +x authenticate.sh
- sudo chmod +x sampledata.sh
- sudo chmod +x imagescript.sh

15. Now run the following command: source /etc/environment

```
ubuntu@demand-ml-draft-1-csv-ml-fix-vm:~$ sudo chmod +x imagescript.sh
ubuntu@demand-ml-draft-1-csv-ml-fix-vm:~$ sudo chmod +x sampledata.sh
ubuntu@demand-ml-draft-1-csv-ml-fix-vm:~$
```

16. After running the " source /etc/environment ",

you need to run the three files. In order to do that, run the following command:

- ./authenticate.sh

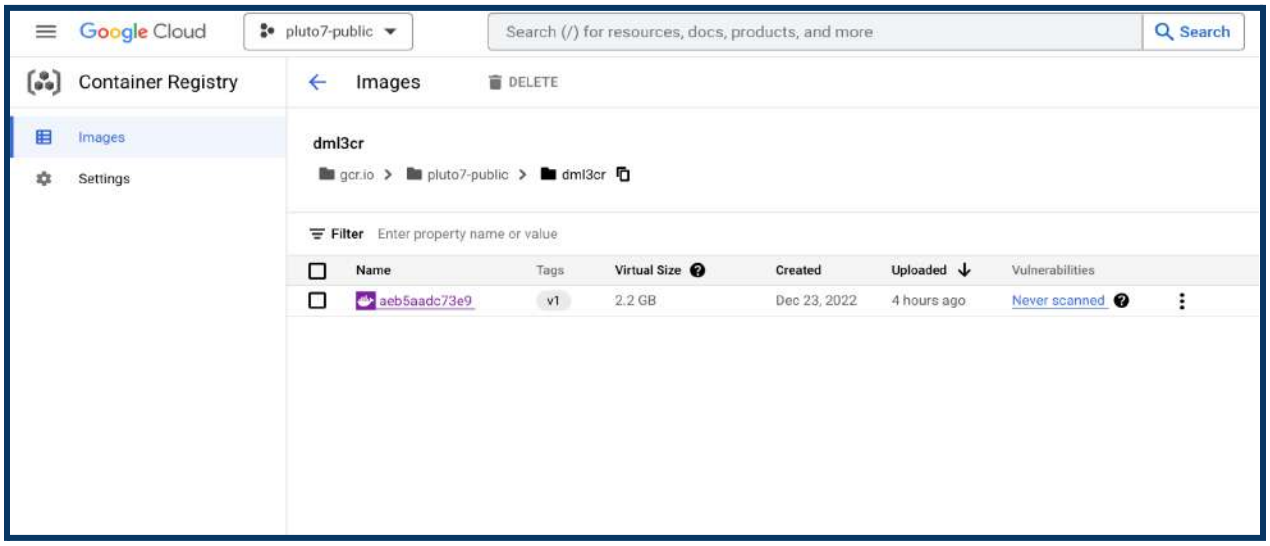
NOTE: This command will take **20 minutes** to complete the execution. Once completed, run the following commands one at a time.

- ./sampledata.sh
- ./imagescript.sh

- sudo rm terraform.tfstate

```
ubuntu@instance-docker-fix-marketplace-update:~$
ubuntu@instance-docker-fix-marketplace-update:~$
ubuntu@instance-docker-fix-marketplace-update:~$ ls
demand-ml  imagescript.sh  key.json  pluto7-public-714ce729e749.json  pluto7-public-e77f9494b942.json  snap
ubuntu@instance-docker-fix-marketplace-update:~$ ./imagescript.sh
```

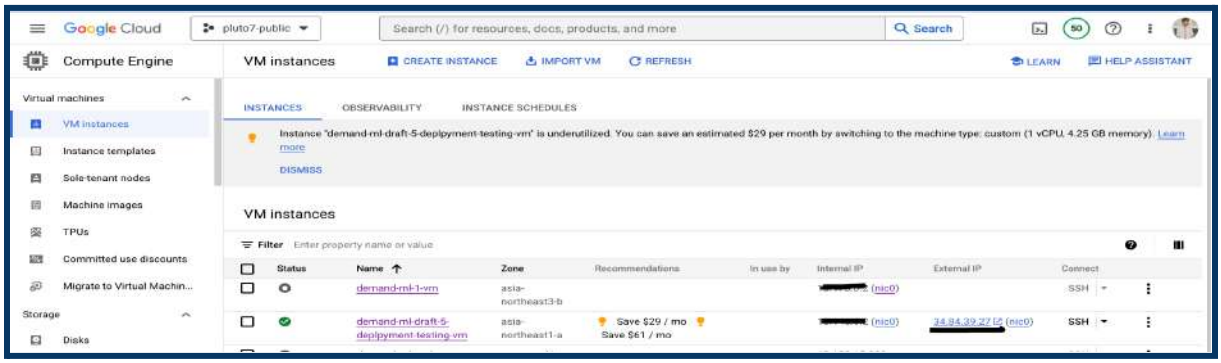
17. After running the script, go to the container registry of your GCP project and verify that a folder called **dml3cr** is created having a docker image.



18. Now restart the service by running the following command: " sudo service supervisor restart " Now you can exit the VM terminal.

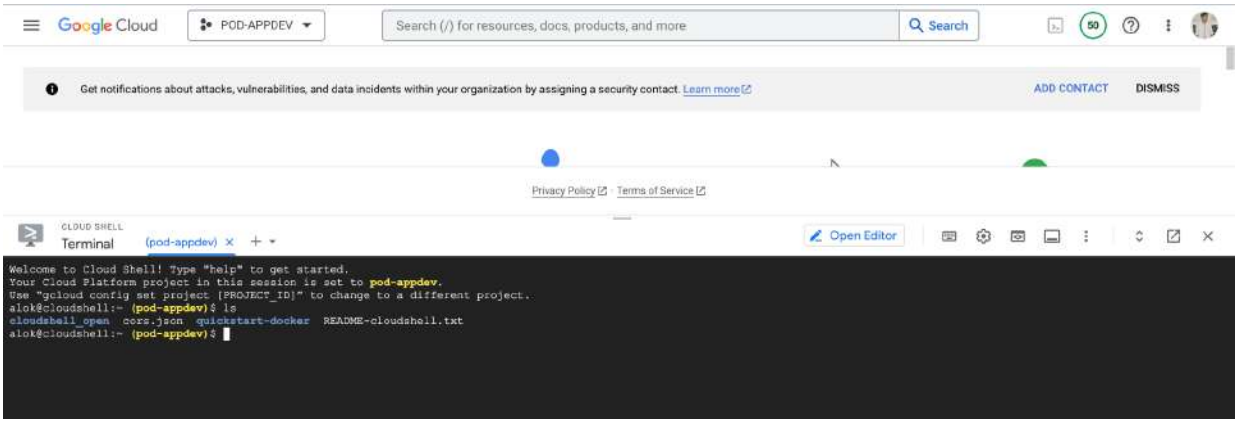
19. Come back to the compute engine page in the GCP console. Select the VM which was deployed and click on stop. After stopping the VM, click on edit.

20. Go to the compute engine Instances page and look for the external IP address of the deployed VM Instance and copy it.



21. Copy the external IP address of the VM Instance in the browser along with the port number as 3000. The format should be http://external_ip_address:port_nuumber

22. Go to your GCP project and open the cloud shell. Enter " ls", and you will discover a file called cors.json



23. Open the cors.json file by entering the command: " sudo nano cors.json "

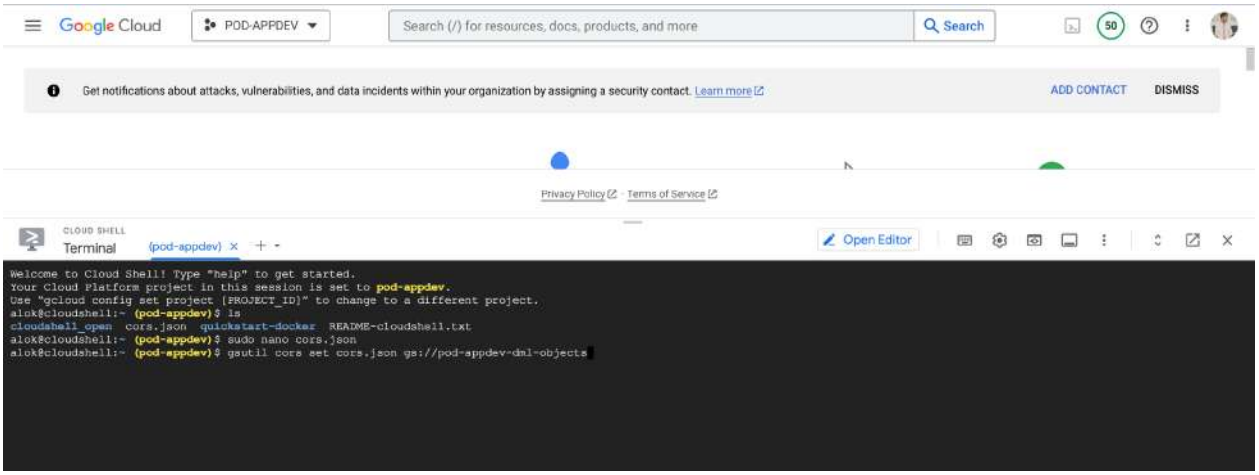
24. Paste the IP address you copied in the earlier step in the cors.json file within quotes.

    Example : [{"maxAgeSeconds": 60, "method": ["PUT"], "origin": ["http://external_ip_address:port_nuumber"], "responseHeader": ["Content-Type"]}]

25. Now press " Control+x to exit it and run the following command to set the cors:

"gsutil cors set cors.json gs://pod-appdev-dml-objects ".

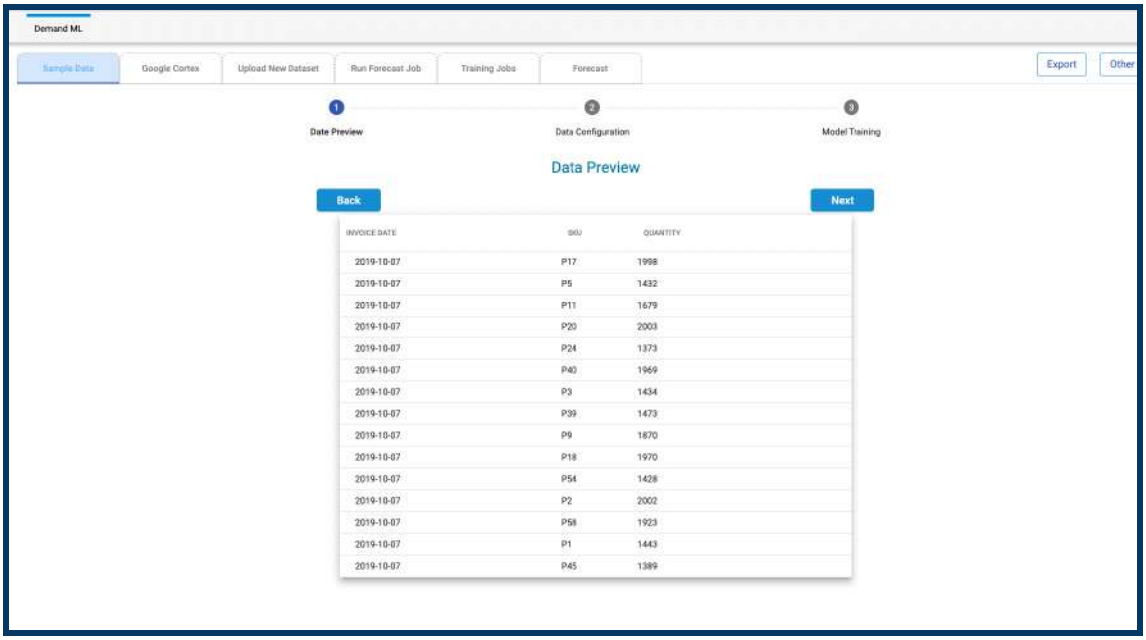
Here , pod-appdev-dml-objects is the GCS bucket name which will be created automatically during deployment.



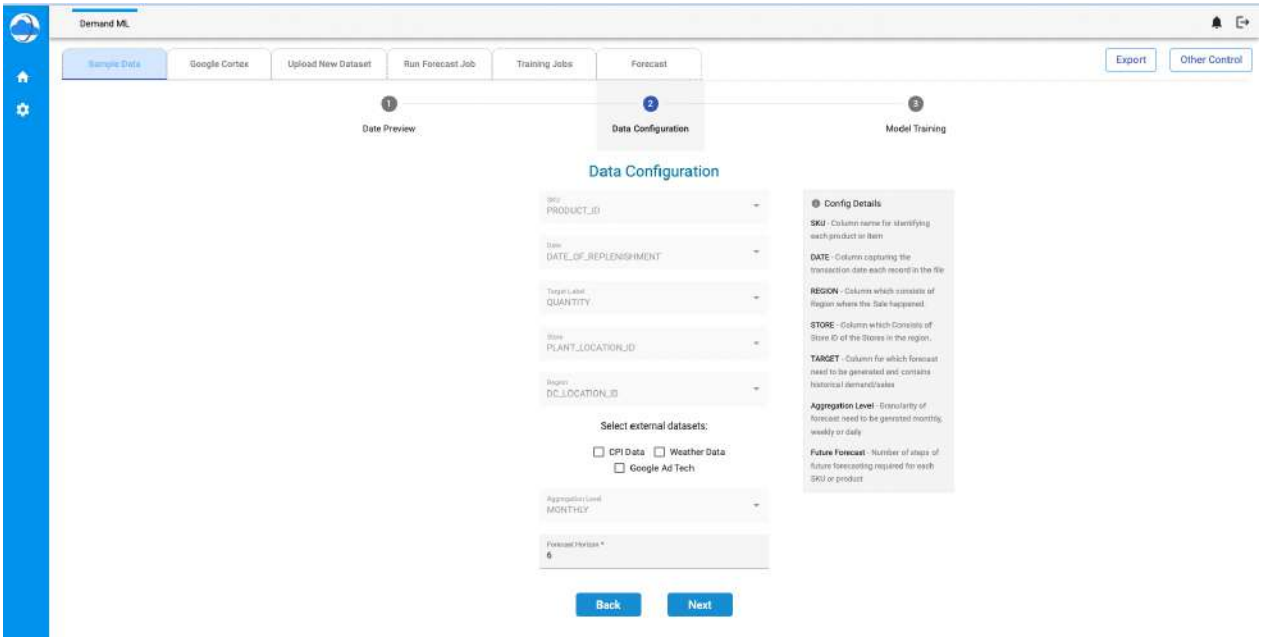
26. Paste the external IP address of the VM Instance in the browser along with the port number as 3000.

The format should be http://external_ip_address:port_nuumber

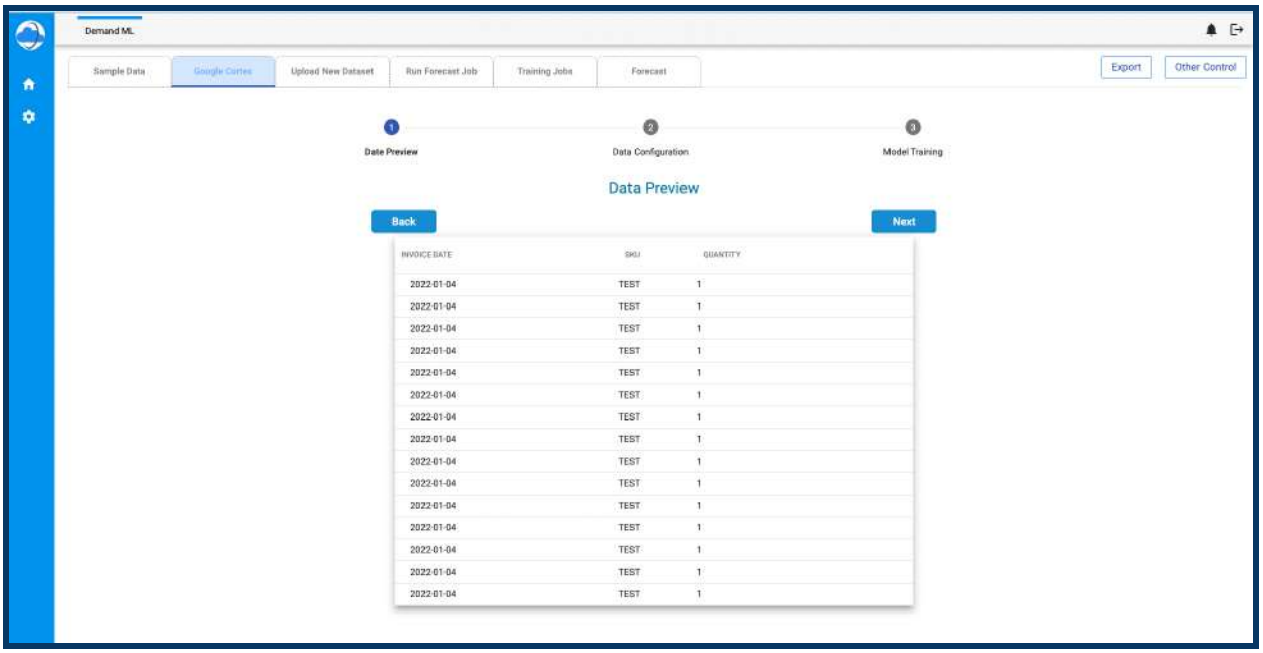
27. After Successful login, the app redirects to the datasets page,



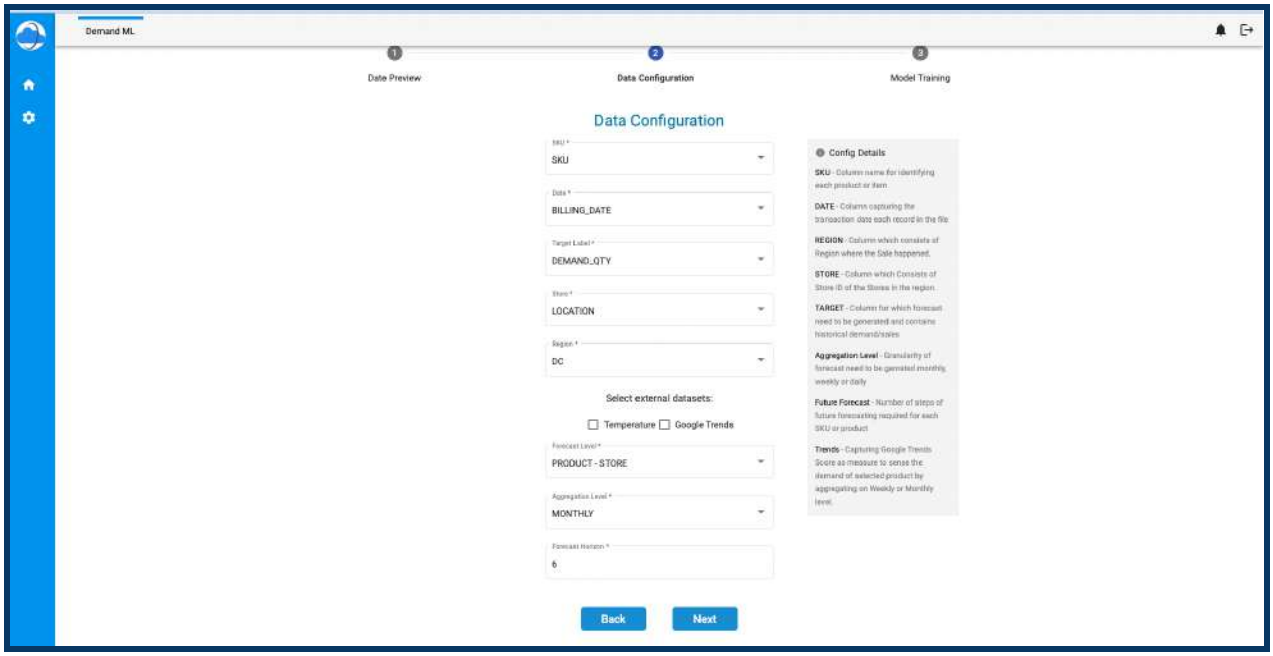
28. Upload New Dataset. The sample dataset is available in Cloud Storage and the bucket name can be identified as **[GOOGLE_CLOUD_PROJECT_NAME]-dml-objects**. The sample dataset is present in the bucket under ***sample_dataset*** folder with the name ***sample-dataset.csv.***



29. This is the Google Cortex Data tab, it gives a preview of the Google Cortex Data.



30. This is the data configuration for the Google Cortex Data. The user can enter the configuration and start the ML model training for the Google Cortex Data.



31. Click on the upload dataset button present on the top right side of the screen

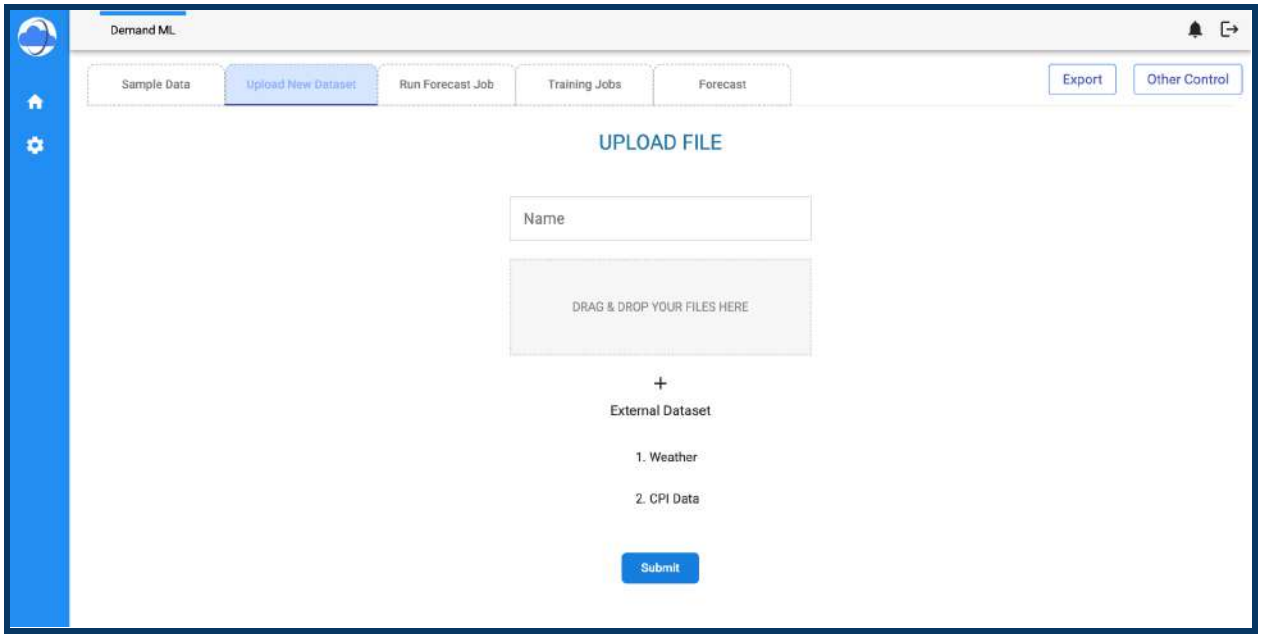


Fig: Upload Dataset

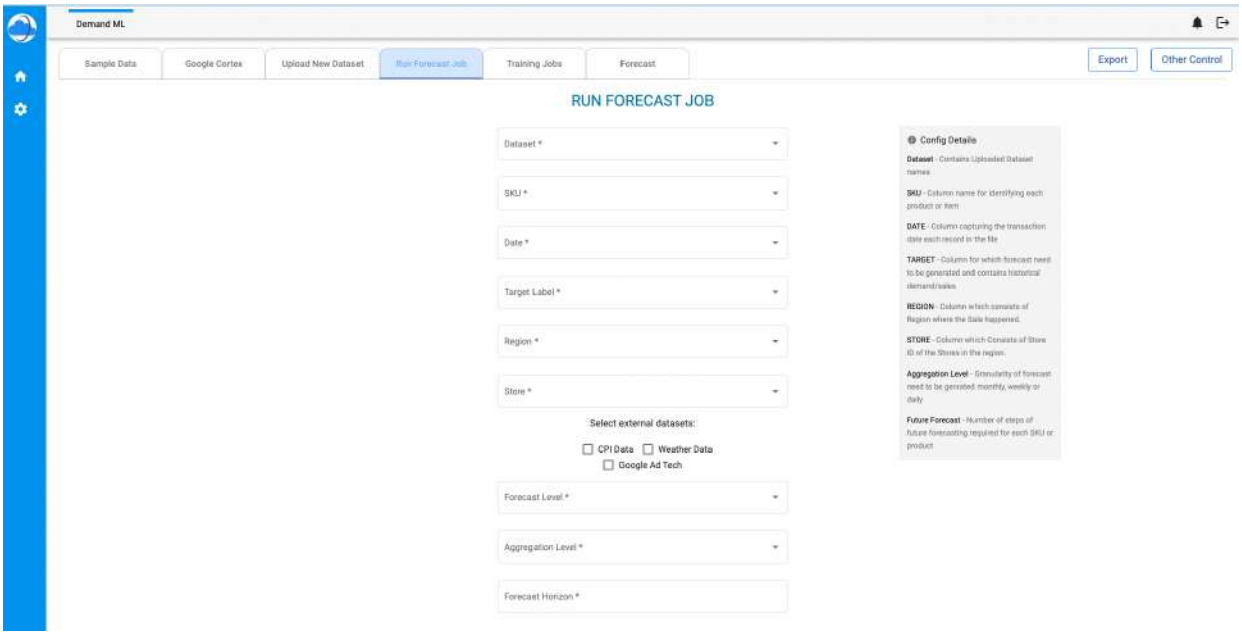


Fig: Select a dataset

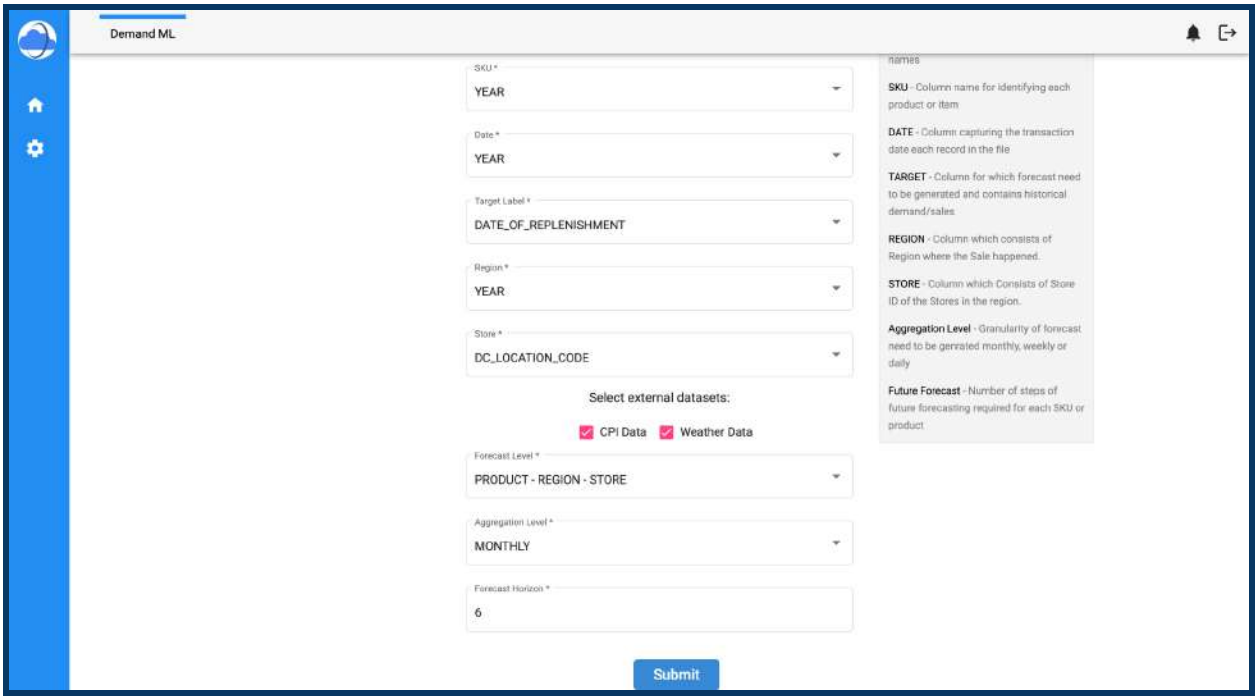


Fig: submit the dataset

32. After data uploading is completed, go to the data experiments page and create a data experiment for the uploaded dataset

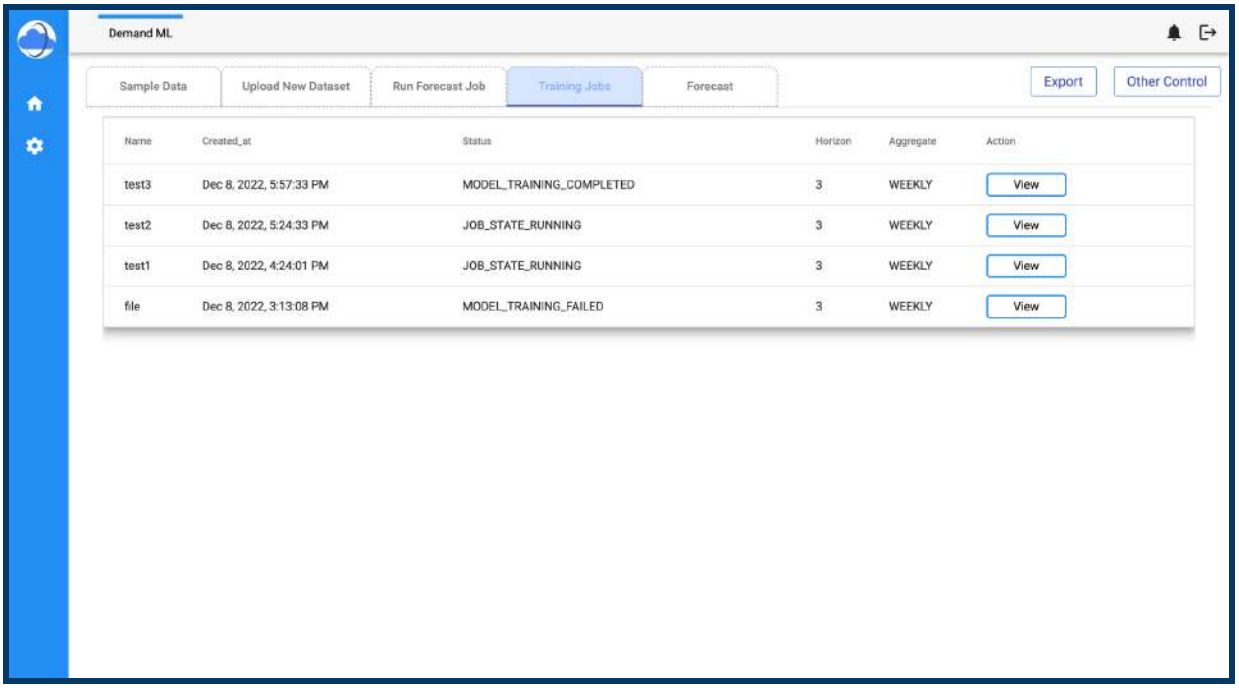


Fig: Training Jobs home page

33. Upon clicking on any of the training jobs, the user can view the forecastability as the output of the ML model for that datase

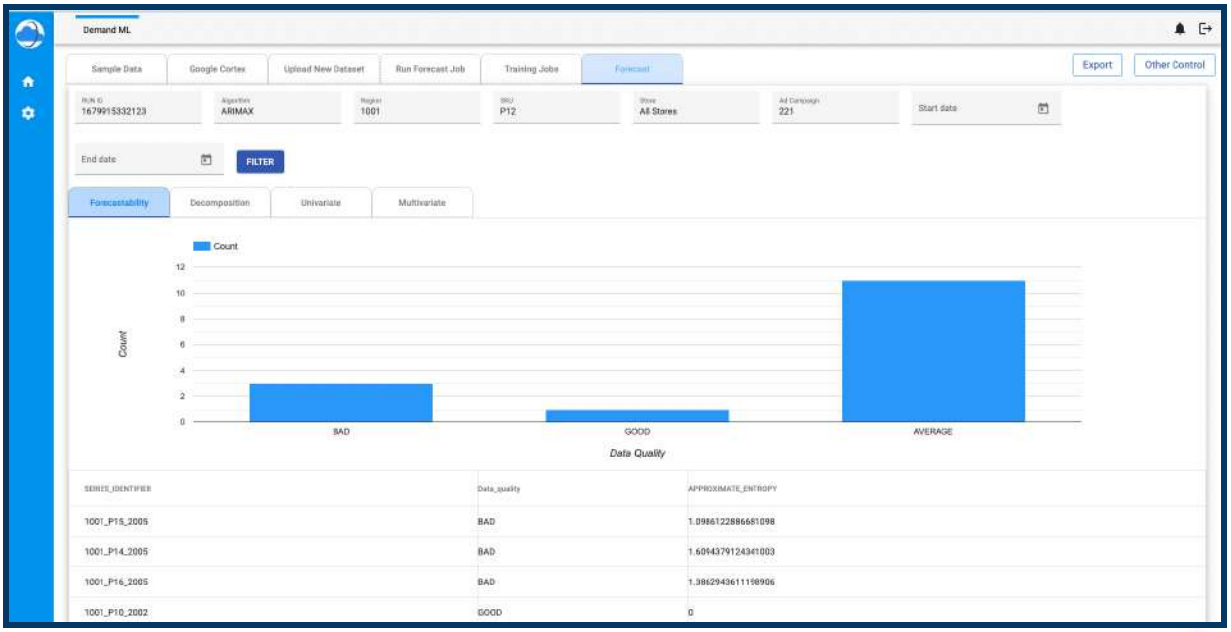


Fig: Forecastability

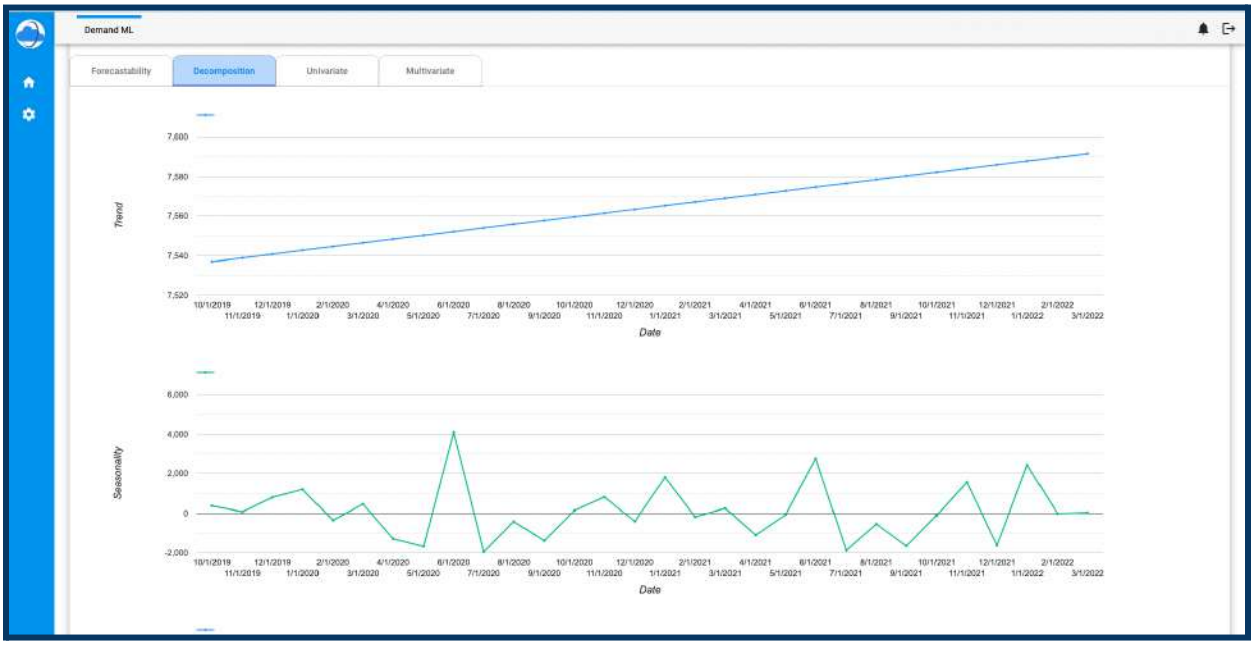
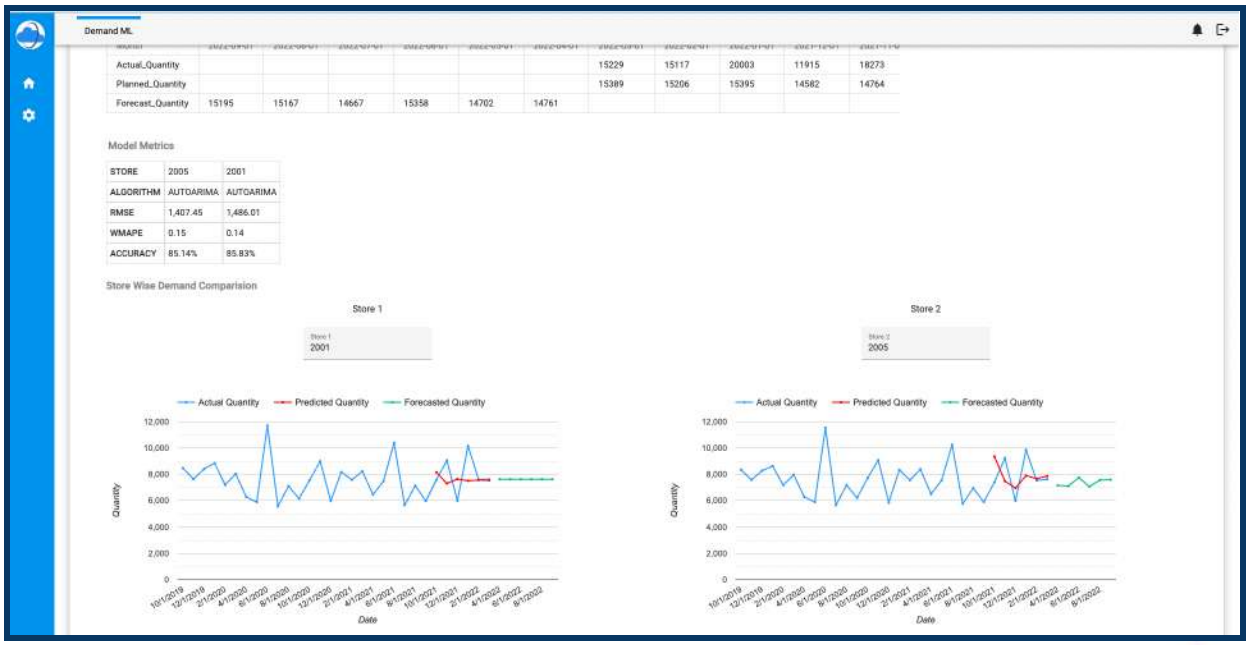


Fig: Seasonality decomposition of the dataset

34. The user can view the ML model results for univariate forecasting which shows the output of the ML model without an external dataset.



35. The user can view ML model results for univariate forecasting, which shows the output of the ML model with external datasets



Fig: Multivariate forecasting output

Contact information

For any queries, please mail us at contact@pluto7.com

Learn More

1. Demand ML Solution Walkthrough
2. https://pluto7.com/success-stories/cdd/