A Machine Learning Laboratory Manual

Crack the basics of Google Cloud Platform (or) Beginner's Handbook For GCP

The only GCP manual you need.

DeepSphere.AI
Enterprise AI and IIoT for Analytics

A Machine Learning Laboratory Manual

Table of Contents

A Machine Learning Laboratory Manual

1. Intro to Google Cloud

- Google Cloud Platform (GCP), offered by Google, is a suite of cloud computing services that runs on the same infrastructure that Google uses internally for its end-user products, such as Google Search, Gmail, file storage, and YouTube
- Google lists over 100 products under the Google Cloud brand.

2. Why Google Cloud

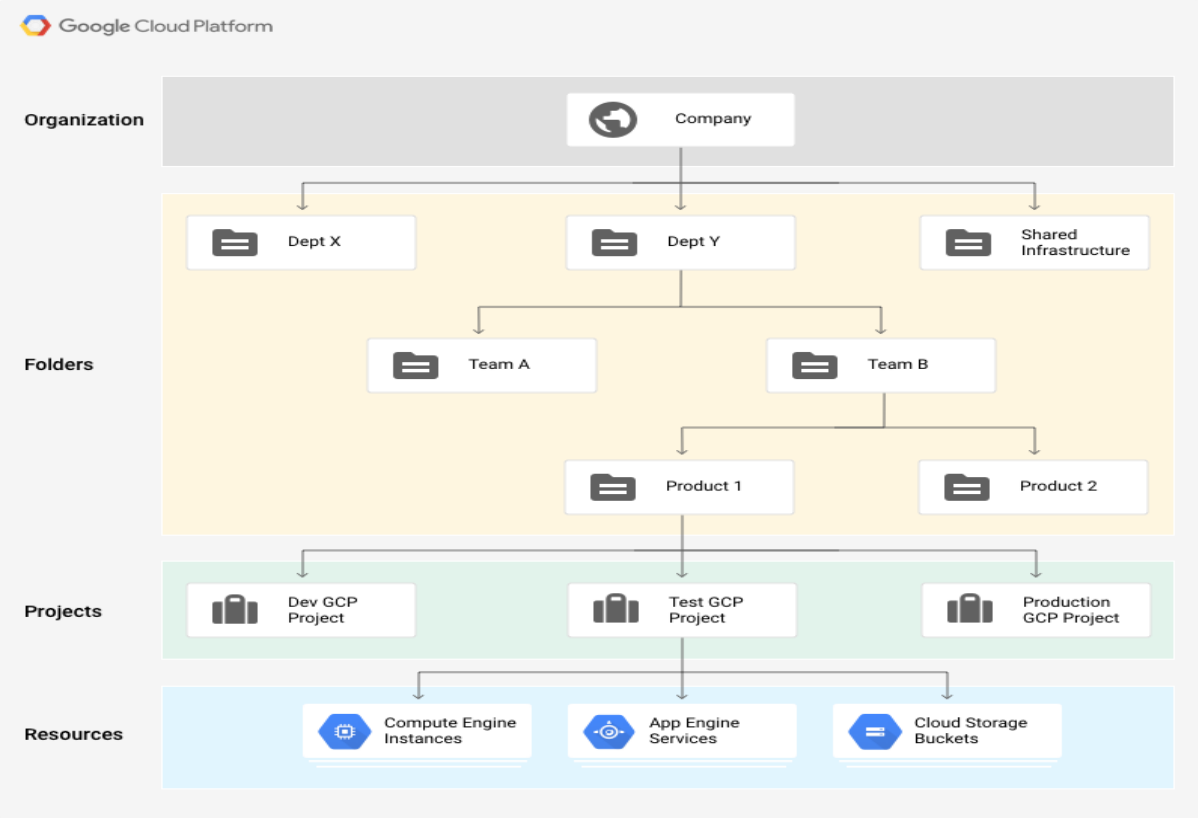
- Google differs from other cloud providers in a variety of ways. Below are some highlights.
    - Run your apps wherever you need them
    - Make smarter decisions with the leading data platform
    - Run on the cleanest cloud in the industry
    - Operate confidently with advanced security tools
    - Transform how you connect and collaborate
    - Save money, increase efficiency, and optimize spend
    - Get customized solutions for your specific industry

3. Google Cloud resource hierarchy

- At the lowest level, resources are the fundamental components that make up all Google Cloud services. Examples of resources include Compute Engine Virtual Machines (VMs), Pub/Sub topics, Cloud Storage buckets, App Engine instances. All these lower level resources

can only be parented by projects, which represent the first grouping mechanism of the Google Cloud resource hierarchy.

- The purpose of the Google Cloud resource hierarchy is two-fold:
    - Provide a hierarchy of ownership, which binds the lifecycle of a resource to its immediate parent in the hierarchy.
    - Provide attach points and inheritance for access control and organization policies.

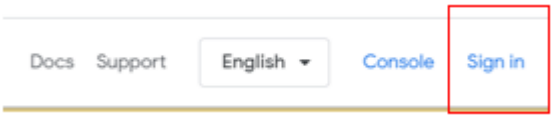


4. Sign in to your Google Cloud Platform Account

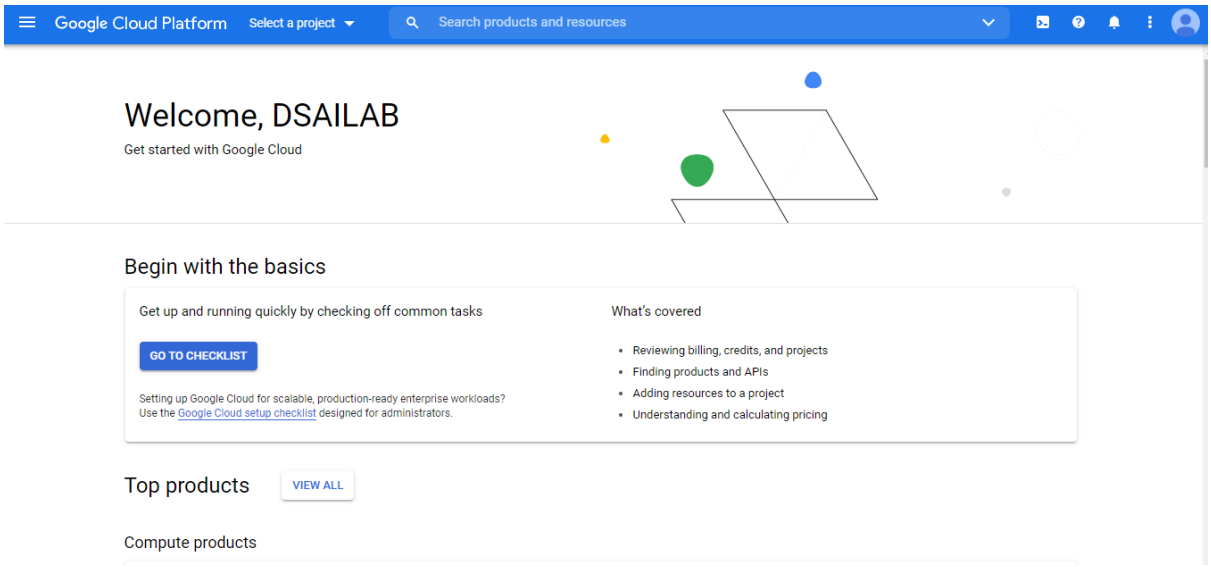
- You need to have a gmail account to create a GCP account. Sign up for a gmail account or you can also use your existing gmail account.

- Visit this url - https://cloud.google.com/

- On the top right side of this web page, click on **Sign in** as shown below and enter your credentials



- Below is the landing page after sign in:



- From here, click on the Google Cloud Platform option on the top left corner in the main menu to open the project dashboard page.

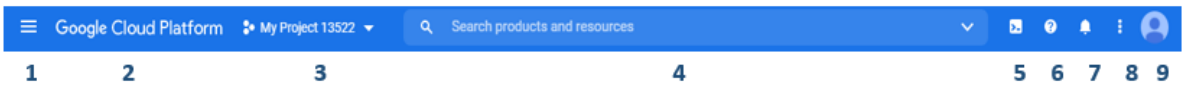
5. GCP Console

- The console will serve as the dashboard of everything-google-cloud-related stuff.

- The console provides:
    - Easy access to all your Google Cloud Platform projects.
    - Access to the Google Cloud Shell.
    - A customizable project dashboard, with an overview of Google Cloud resources, billing, and a filterable activity listing.
    - Easy access to all Google Cloud Platform APIs, with a dashboard specific to each API, and access to manage your resources.
    - Links to Google Cloud Platform starting points, news, and documentation.

5.1. GCP Console Main Menu

Let's have a look at the main menu options:



1. **Navigation menu** — it expands and lists all the most relevant GCP components grouped by category, more on this later.
2. **Navigation link to the Dashboard** — useful if you're in a product's page and you want to quickly navigate back.
3. **Project selector** — it will display a pop-up window allowing you to select the current project.
4. **Search bar** — it allows you to search full text among the products, services and functionalities offered by GCP.
5. **Cloud Shell** — this button provides command-line access to a virtual machine instance in a terminal window that opens in the web console, more on this later.

6. **Help** — it shows a contextual pop-up window with articles and support options.

7. **Notifications** — it will aggregate and notify about relevant events regarding your project and account.

8. **Settings** — it allows you to set your project preferences, keyboard shortcuts, a direct link to tools downloads.

9. **Google Account** — it shows information related to the account with which you're currently logged in, allowing you to log out or add other accounts.

5.2. Creating projects

- Any Google Cloud resources that you allocate and use must belong to a project. You can think of a project as the organizing entity for what you're building. A project is made up of the settings, permissions, and other metadata that describe your applications.

- Each Google Cloud project has the following:

    - A project name, which you provide.

    - A project ID, which you can provide or Google Cloud can provide for you.

    - A project number, which Google Cloud provides.

- Each project ID is unique across Google Cloud. Once you have created a project, you can delete the project but its ID can never be used again.

- When billing is enabled, each project is associated with one billing account. Multiple projects can have their resource usage billed to the same account.

- To create a project, you must have the resourcemanager.projects.create permission. This permission is included in the Project Creator roles/resourcemanager.projectCreator role, which is granted by default to the entire domain of a new organization and to free trial

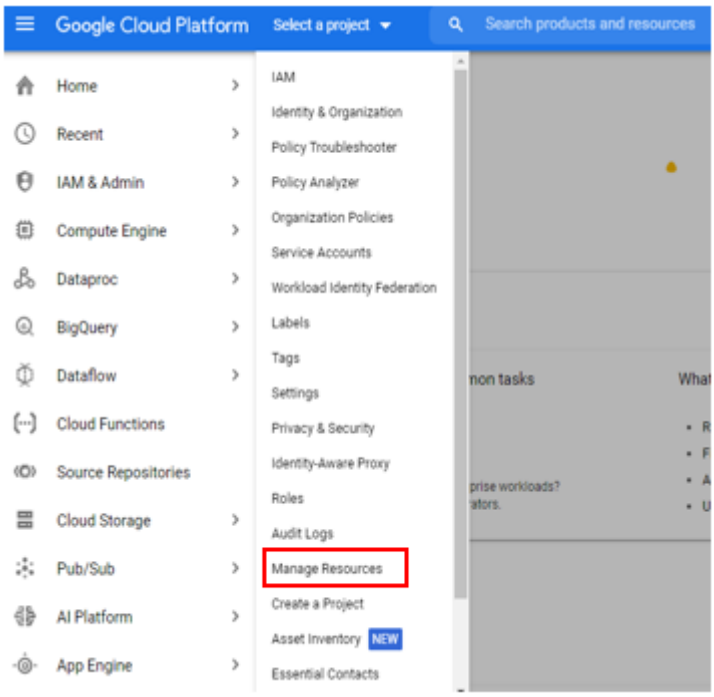
users. For information on how to grant individuals the role and limit organization-wide access, see the Managing Default Organization Roles page.

- If you do not specify the parent resource, a parent resource is selected automatically based on the user account's domain.

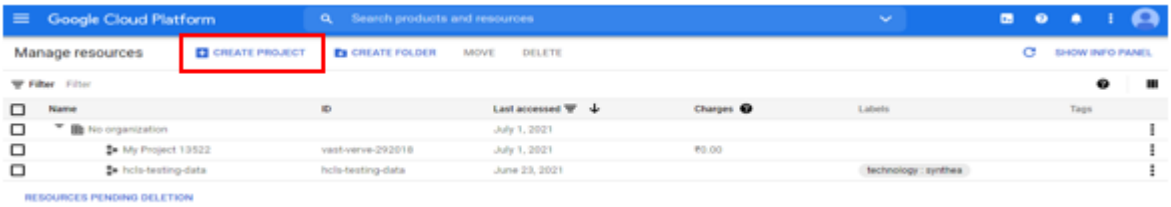
5.2.1. Create a Project using Console

- To create a new project, do the following:
    - Click on navigation menu (3 horizontal lines):

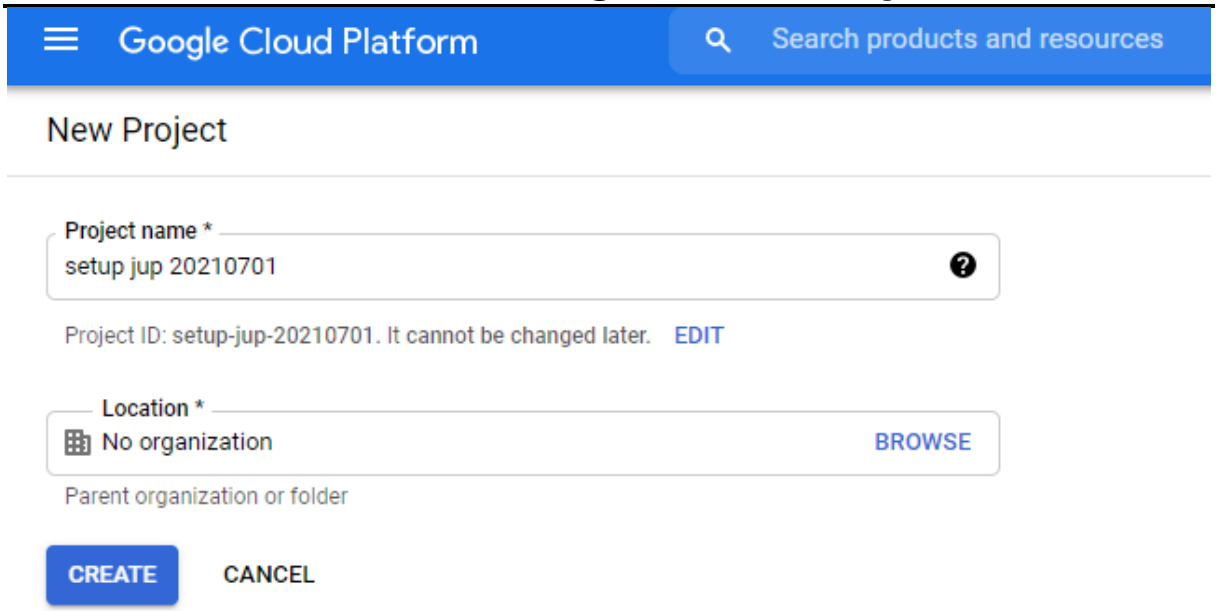
    - Under IAM and Admin , click on Manage resources:



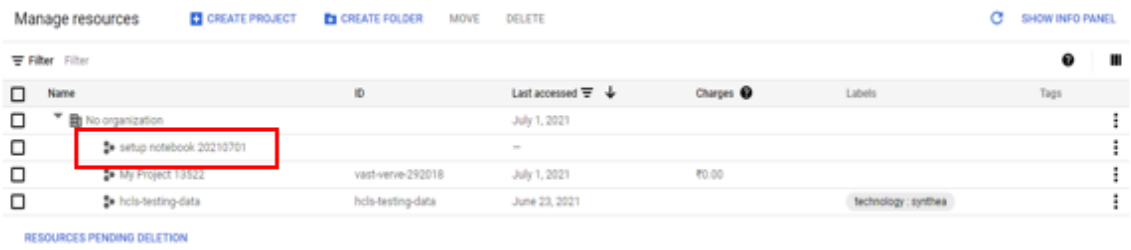
- Click on create project:



- In the **New Project** window that appears, enter project name, enter the parent organization or folder in the **Location** box. and click on create



- Now, the project has been created



5.2.2. Create a Project using gcloud

- To create a new project, use the gcloud projects create command:

  gcloud projects create PROJECT_ID

```
rakhi@cloudshell:~ (setup-jup-05072021)$ gcloud projects create setup-jup-100
Create in progress for [https://cloudresourcemanager.googleapis.com/v1/projects/setup-jup-100].
Waiting for [operations/cp.7841883079548277302] to finish...done.
Enabling service [cloudapis.googleapis.com] on project [setup-jup-100]...
Operation "operations/acf.p2-35898458126-487b7497-5571-4a8b-81ad-68c5ca99b0d6" finished successfully.
rakhi@cloudshell:~ (setup-jup-05072021)$
```

- Where PROJECT_ID is the ID for the project you want to create. A project ID must start with a lowercase letter, and can contain only ASCII letters, digits, and hyphens, and must be between 6 and 30 characters.

- To create a project with an organization or a folder as parent, use the --organization or --folder flags. As a resource can only have one parent, only one of these flags can be used:

gcloud projects create PROJECT_ID --organization=ORGANIZATION_ID

```
rakhi@cloudshell:~ (setup-jup-05072021)$ gcloud projects create project-1307202111 --organization=1064084812112
Create in progress for [https://cloudresourcemanager.googleapis.com/v1/projects/project-1307202111].
Waiting for [operations/cp.8473738127427419362] to finish...done.
Enabling service [cloudapis.googleapis.com] on project [project-1307202111]...
Operation "operations/acf.p2-922590390015-af62dfb3-38de-43b8-9e84-d21447c440aa" finished successfully.
rakhi@cloudshell:~ (setup-jup-05072021)$
```

gcloud projects create PROJECT_ID --folder=FOLDER_ID

5.2.3. Create a Project using API

- You can't use certain words in the project ID when you create a new project with the projects.create() method. Some examples include ssl and google. When you use a restricted word, the request returns with an INVALID_ARGUMENT error.

- The below request only creates a project, and does not associate it automatically with a billing account. Use the projects.updateBillingInfo method to set or update the billing account associated with a project.

- Create Project Request:

POST https://cloudresourcemanager.googleapis.com/v3/projects/
*Authorization: **********

```
Content-Type: application/json

{
  "projectId": "our-project-123",
  "name": "my project",
  "labels": {
    "mylabel": "prod"
  }
}
```

- Create Project Response:

```
{
  "name": "operations/pc.123456789",
}
```

- Get Operation Request:

```
GET https://cloudresourcemanager.googleapis.com/v3/operations/pc.123456789

Authorization: **********

Content-Type: application/json
```

- Get Operation Response:

```
{
  "name": "operations/pc.123456789",
  "done": true,
  "response": {
    "@type": "type.googleapis.com/google.cloudresourcemanager.v3.Project",
    "projectNumber": "464036093014",
    "projectId": "our-project-123",
    "lifecycleState": "ACTIVE",
    "name": "my project",
    "labels": {
```

```
      "mylabel": "prod"
      },
   "createTime": "2016-01-07T21:59:43.314Z"
   }
}
```

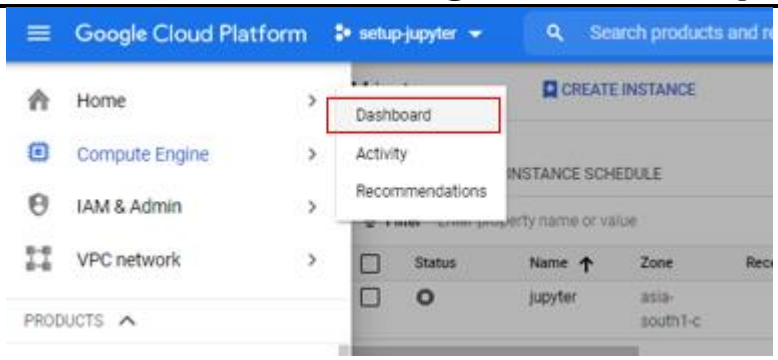
5.2.4. Create a Project using Python

```
from googleapiclient import discovery
from oauth2client.client import OAuth2Credentials as creds
crm = discovery.build(
   'cloudresourcemanager', 'v3', http=creds.authorize(httplib2.Http()))

operation = crm.projects().create(
body={
   'project_id': flags.projectId,
   'name': 'my project'
}).execute()
```

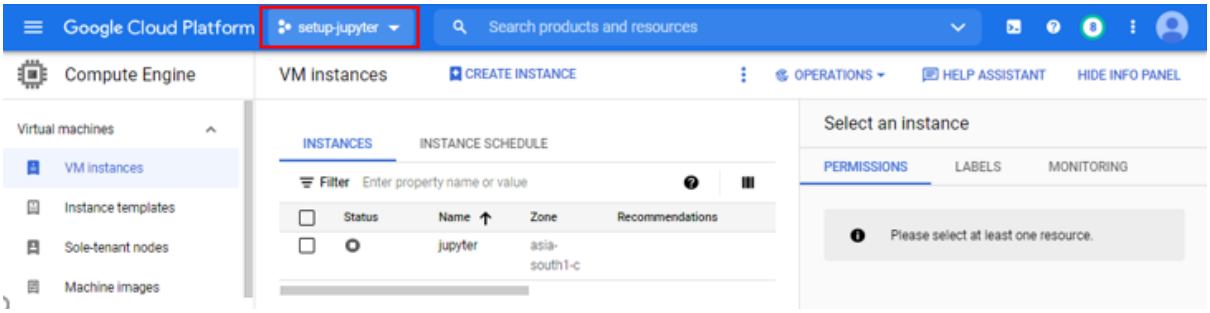
5.3. Get an Existing Project

5.3.1. Using Console

- To view a project using the Google Cloud Console, do the following:
    - Click on navigation menu, under Home, click on Dashboard.



- Click the drop-down list at the top of the page. In the window that appears, select your project.



5.3.2. Using gcloud

- To get the metadata for a project, use the gcloud projects describe command:

gcloud projects describe PROJECT_ID

```
For detailed information on this command and its flags, run:
  gcloud projects create --help
rakhi@cloudshell:~ (setup-jup-05072021)$ gcloud projects describe setup-jup-100
createTime: '2021-07-12T03:48:21.956Z'
lifecycleState: ACTIVE
name: setup-jup-100
parent:
  id: '1064084812112'
  type: organization
projectId: setup-jup-100
projectNumber: '35898458126'
rakhi@cloudshell:~ (setup-jup-05072021)$
```

  - Where PROJECT_ID is the ID of the project you want to view.

5.3.3. Using API

- Request:

```
GET
https://cloudresourcemanager.googleapis.com/v3/projects/our-project-123
```

- Response:

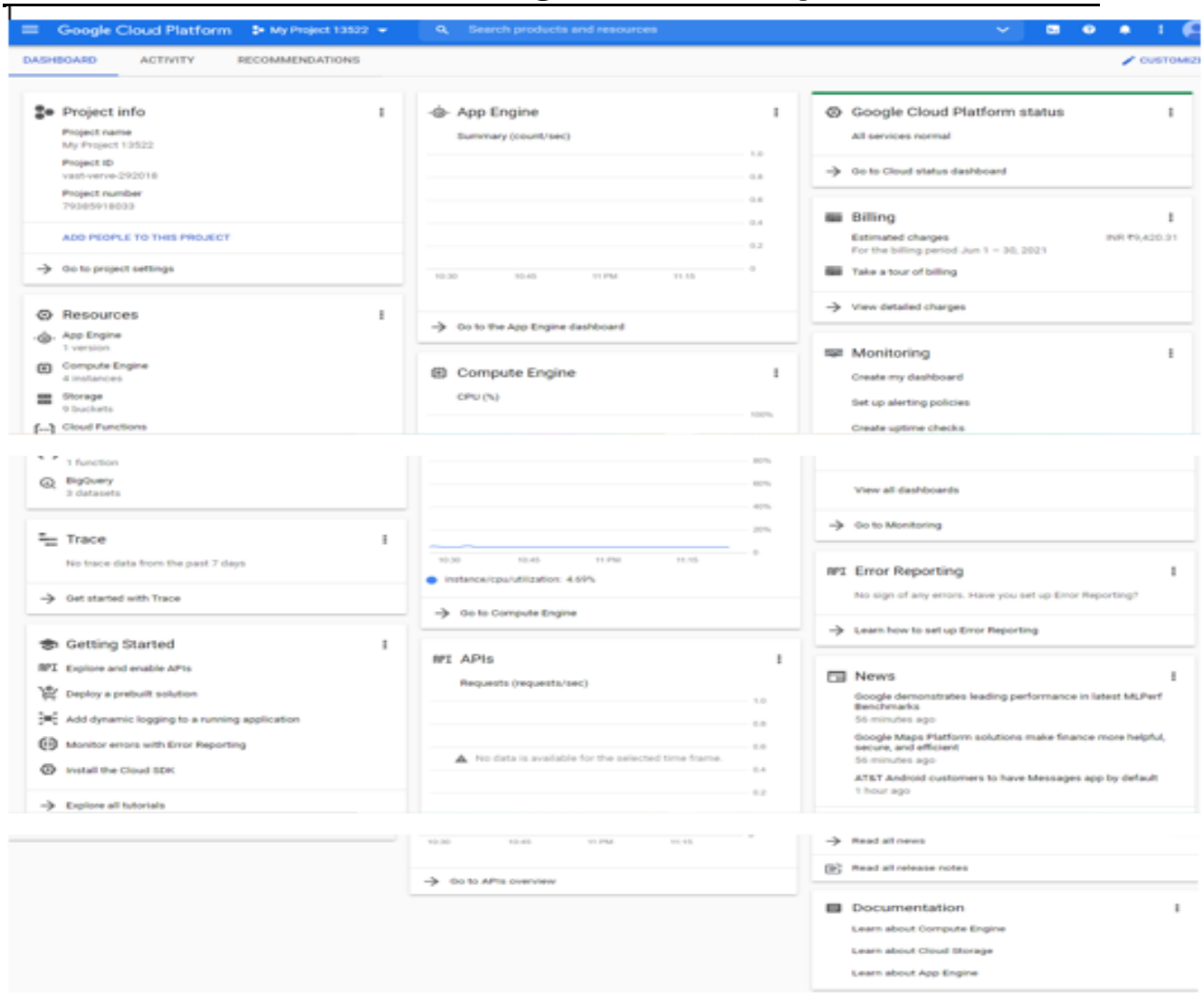
```
{
  "projectNumber": "464036093014",
  "projectId": "our-project-123",
  "lifecycleState": "ACTIVE",
  "name": "my project",
  "labels": {
    "mylabel": "prod"
  },
  "createTime": "2016-01-07T21:59:43.314Z"
}
```

5.3.4. Using Python

```
from googleapiclient import discovery
from oauth2client.client import OAuth2Credentials as creds
crm = discovery.build(
  'cloudresourcemanager', 'v3', http=creds.authorize(httplib2.Http()))

project = crm.projects().get(projectId=flags.projectId).execute()
```

5.4. Project Dashboard

- Google Cloud Platform automatically creates a dashboard for each project with default widgets, as seen in the following screenshot:

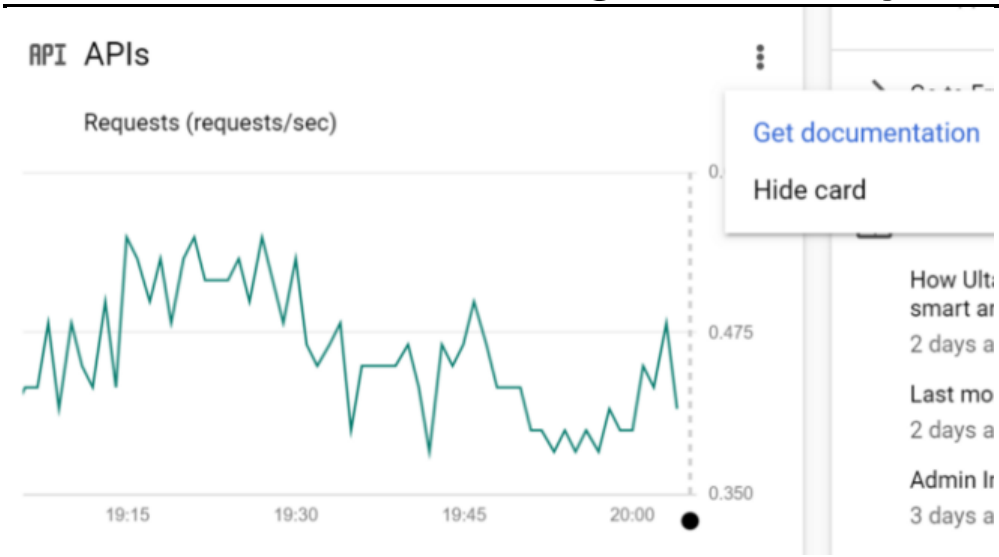


- The body of the page contains several different customizable widgets that provide information, reports, stats and help related to the current project and GCP in general (what you will see might differ according to your project and your settings).

- Let's have a quick glance at some of these widgets:
  - **Project info** — it specifies the basic information related to the current project, like name, ID, number.

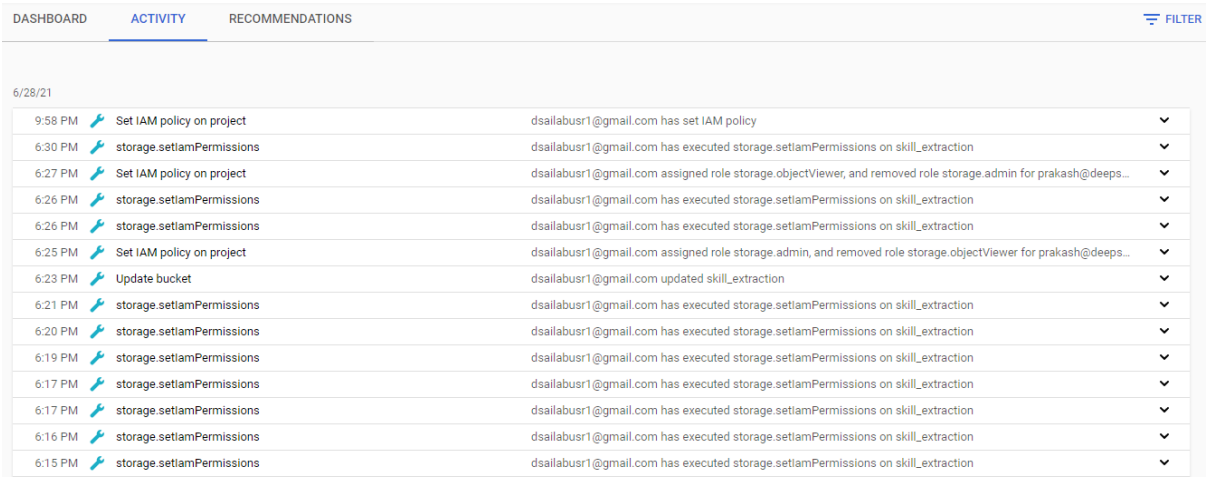
- **Resources** — it lists the main resources/components used in the project.
- **Trace** — if Stackdriver Trace is enabled, it provides the latest trace data.
- **Getting started** — it links to quick tutorials of the most common operations.
- **App Engine** — it shows a graph related to the usage of one instance of App Engine, by default it's count/sec, but you can specify different parameters for the main graph or add a new one.
- **APIs** — a graph showing the requests per second to the APIs used by the project.
- **Google Cloud Platform Status** — as the name implies it reports the status of GCP services, in the unlikely scenario that some outage occurs.
- **Billing** — it shows an estimate of the charges related to this project in the current period.
- **Error reporting** — it shows the last 24 hours errors collected by Stackdriver error reporting.
- **News** — It aggregates the feeds of news related to GCP and Cloud in general.
- **Documentation** — Hot links to GCP documentation.

- Each one of them offers you a contextual menu that allows you to either get to the related documentation or to hide the widget as shown below:



5.5. Project Activity

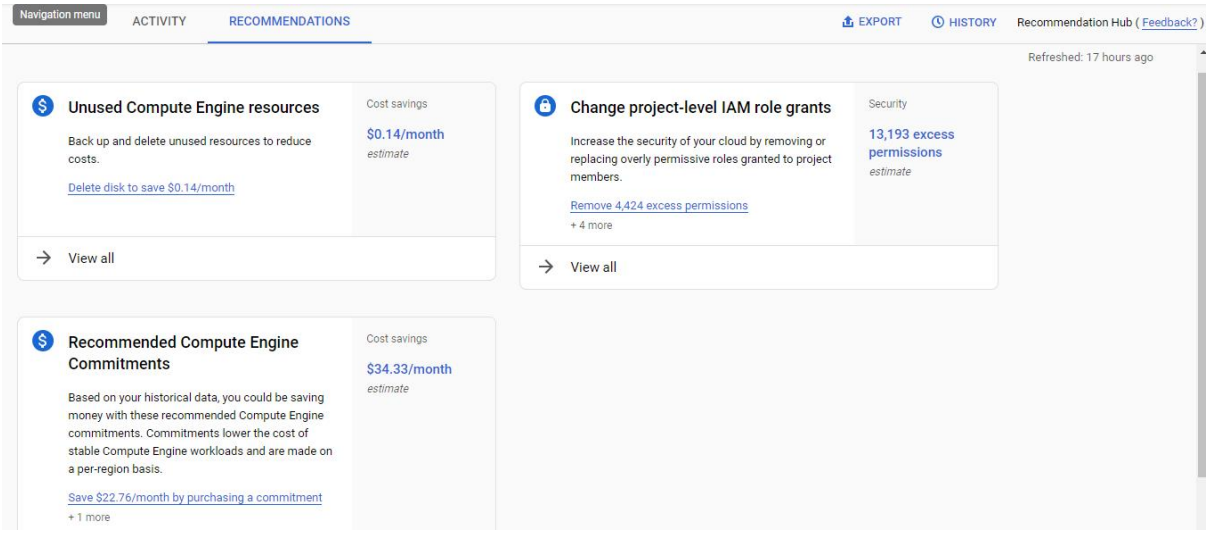
- Activity shows all the events happening across all the services part of your project, with the related options to filter them on the right-hand side as shown below:



5.6. Recommendations for the Project

- With Recommenders, you can optimize your cloud for price, performance, and security. Several Recommenders are already generally available, including VM Recommenders, Firewall Insights and IAM Recommender.

- In fact, there are many teams at Google Cloud who are working to build Recommenders that help you improve your cloud. But, to make sure it's effortless and simple for you to find and take action on those recommendations. That's why Google has released the beta of our new Recommendation Hub, which highlights proactive recommendations in one place for you to view and act on.



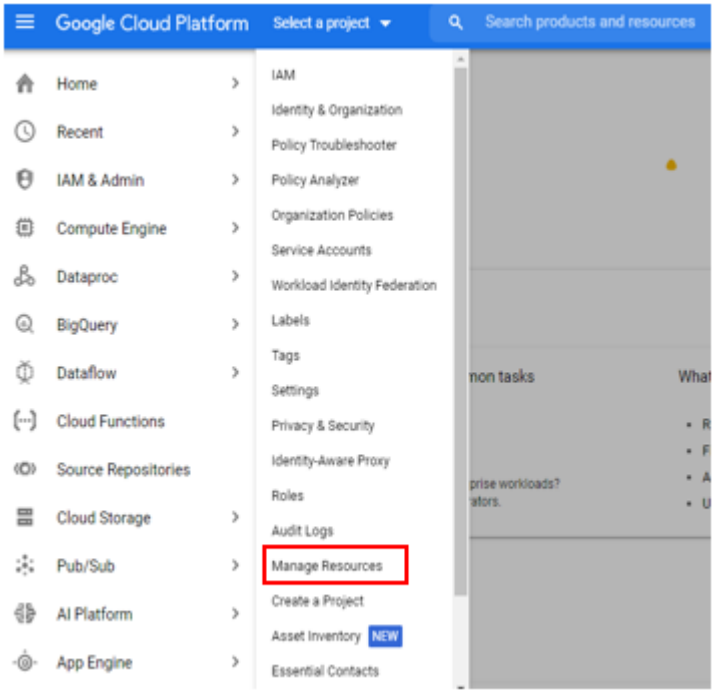
6. How to run Jupyter Notebook using VM Instance

6.1. Create a new project

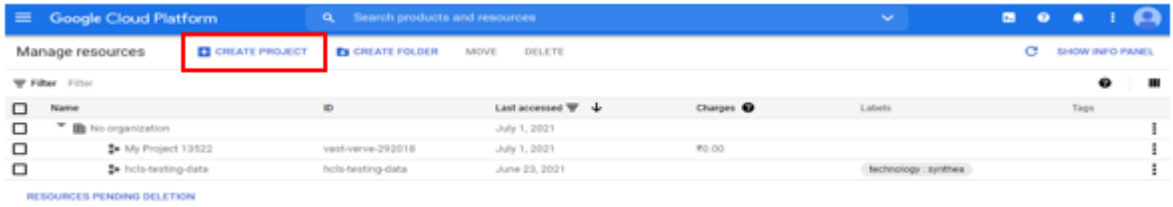
- In order to set up, for example: you need to select an existing project or create a new project in GCP.

- To create a new project, Click on navigation menu (3 horizontal lines):

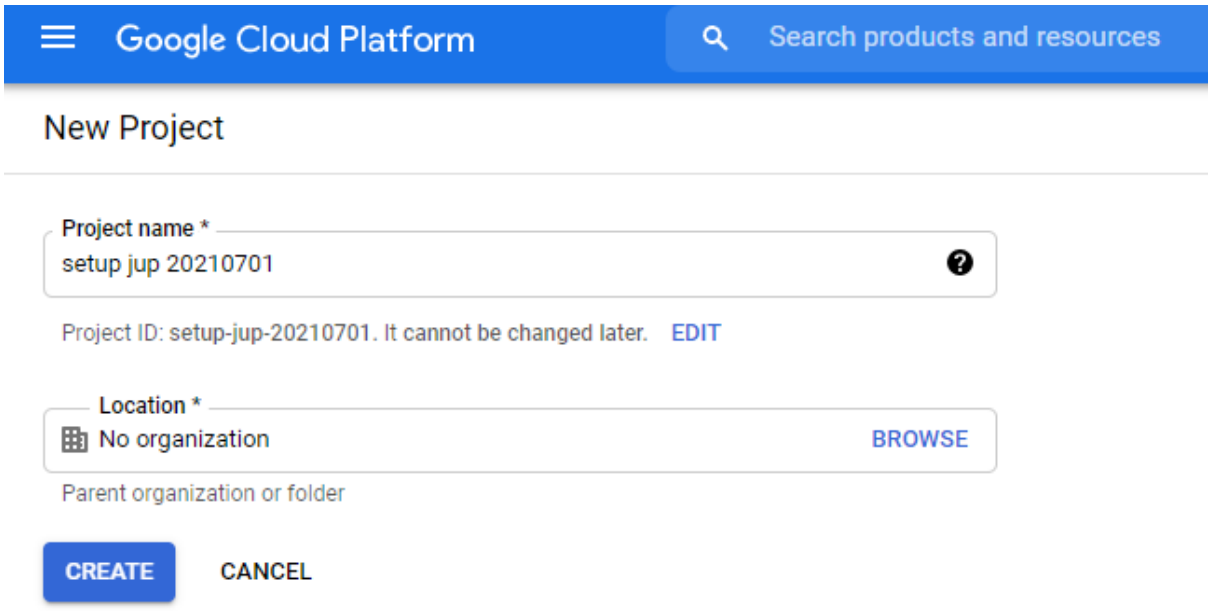
- Under IAM and Admin , click on Manage resources:



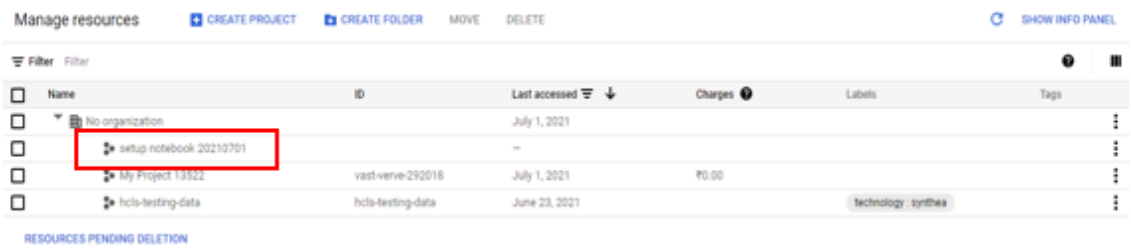
- Click on create project:



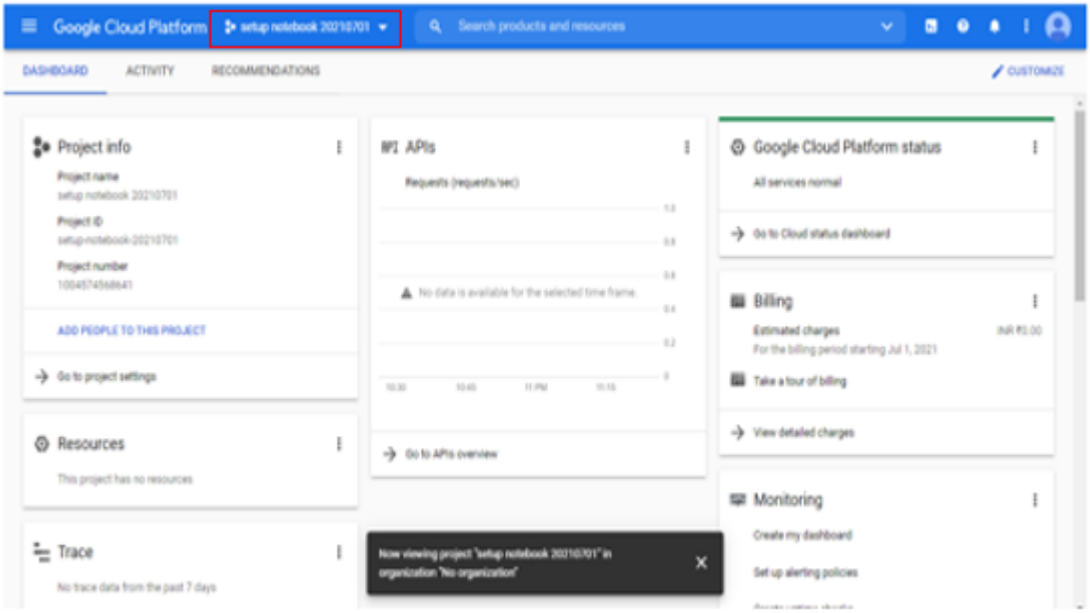
- Enter project name and click on create



- Now, the project has been created



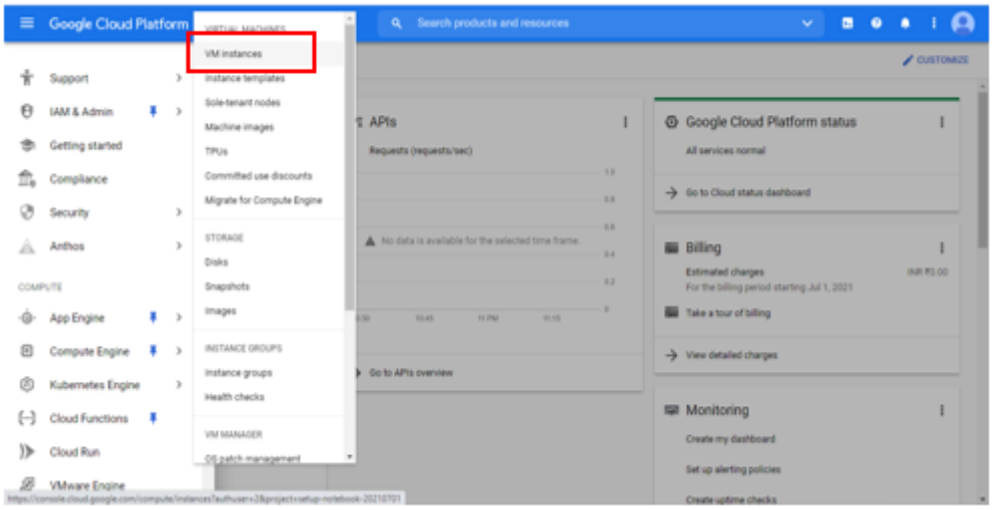
- Click on Google Cloud Platform:



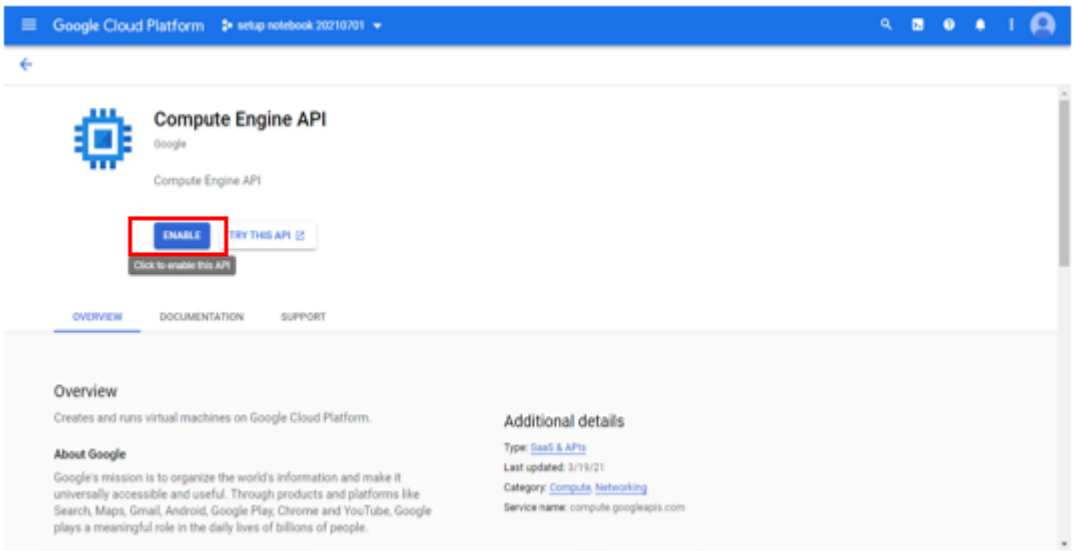
- Note: If a different project is selected by default, select your project from the above dropdown.

6.2. Setup a VM Instance

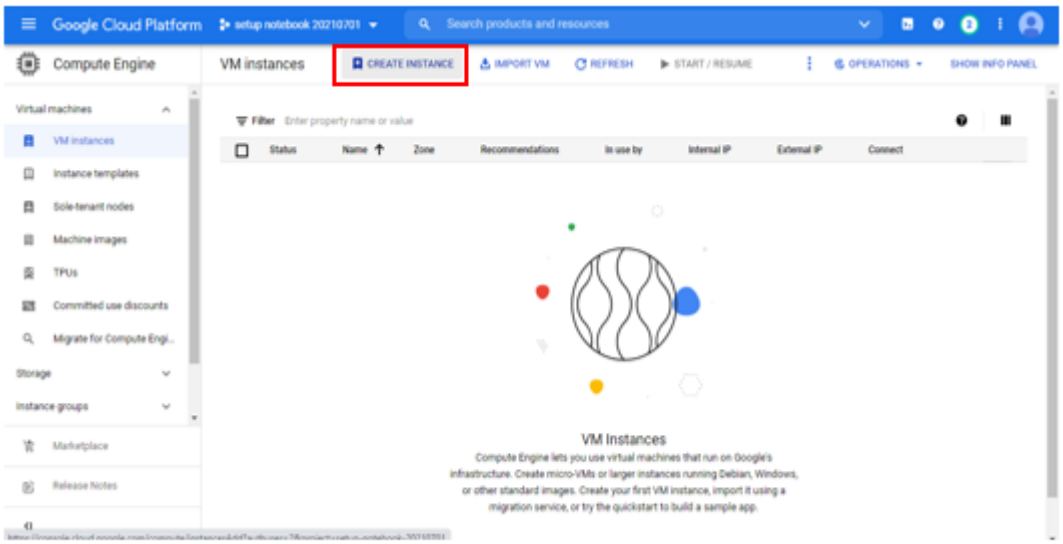
- Navigation menu – Compute - Compute engine – VM instances



- In a new project, we must enable the Computer Engine API so click on Enable:

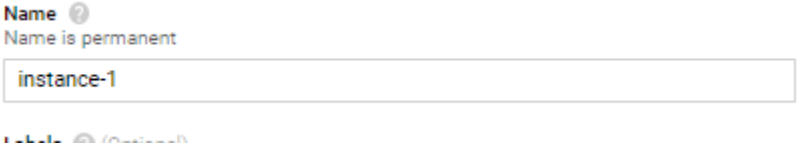


- This is the landing page after enabling the API. Now, click on Create Instance:

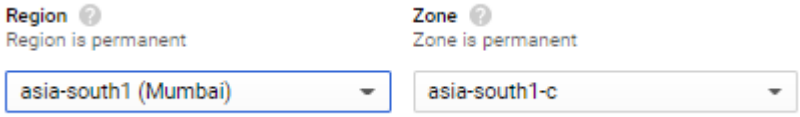


- Enter the following details in the below page:

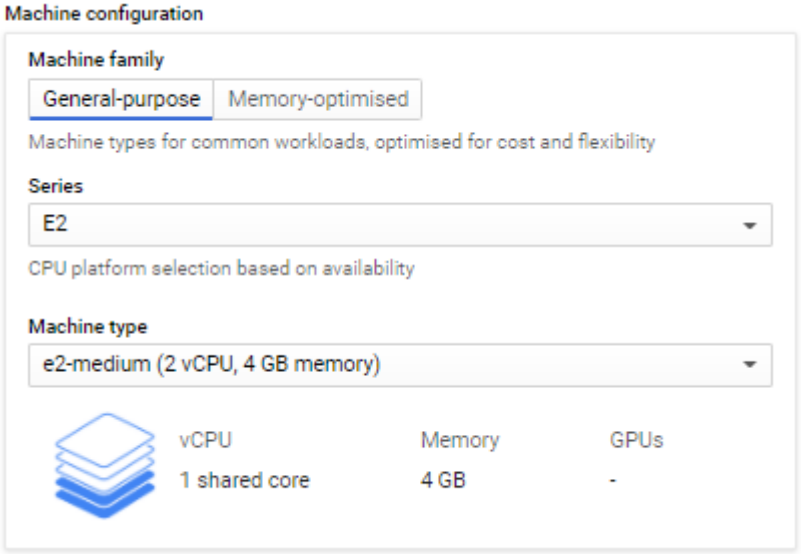
- **Instance Name** (You can give any name)



- **Select the region and zone.** (Select region and zone close to you, in my case, I chose "asia-south1-c")

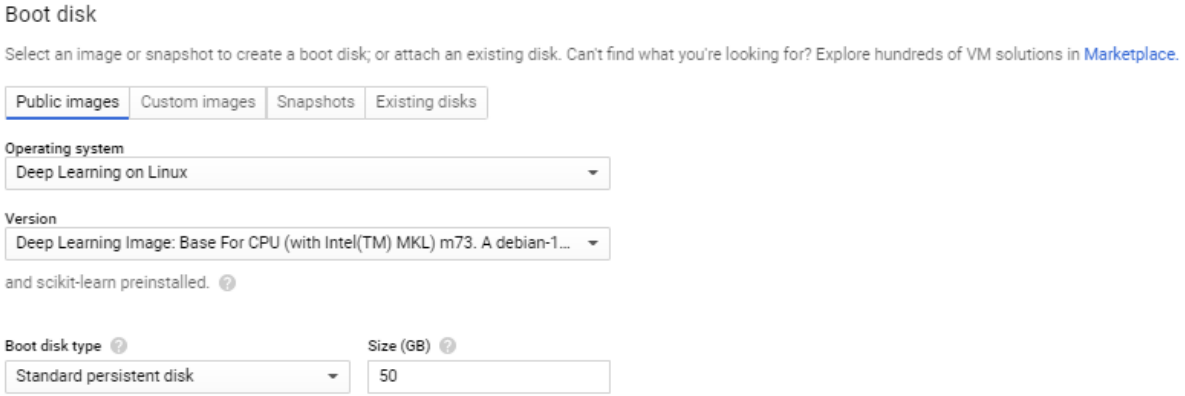


- **Machine Type:** (as per your need)

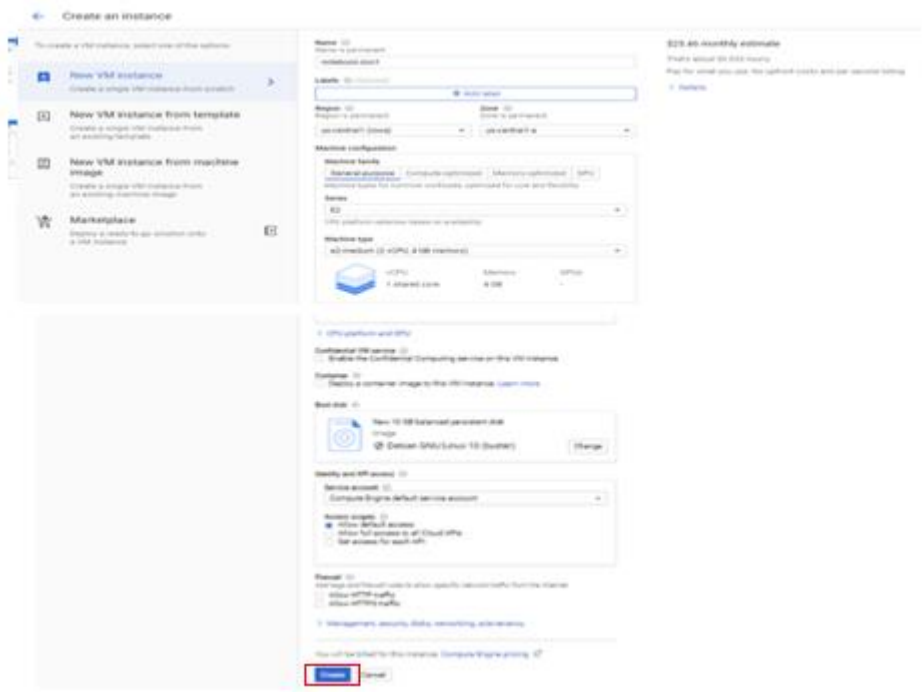


- **Boot Disk:**

  By default, “Debian GNU/Linux” is selected. You can change it as per your need by clicking the “Change” button. (I’ve chosen “Deep Learning on Linux” as it comes with pre-installed libraries)



- **Boot disk type**: Standard Persistent disk
- **Size(GB):** (as per your need)
- Keep **"Identity and API access"** as default.
- **Firewall**: Click the checkbox to allow both HTTP and HTTPS traffic.
- Click on the **"Create"** button.

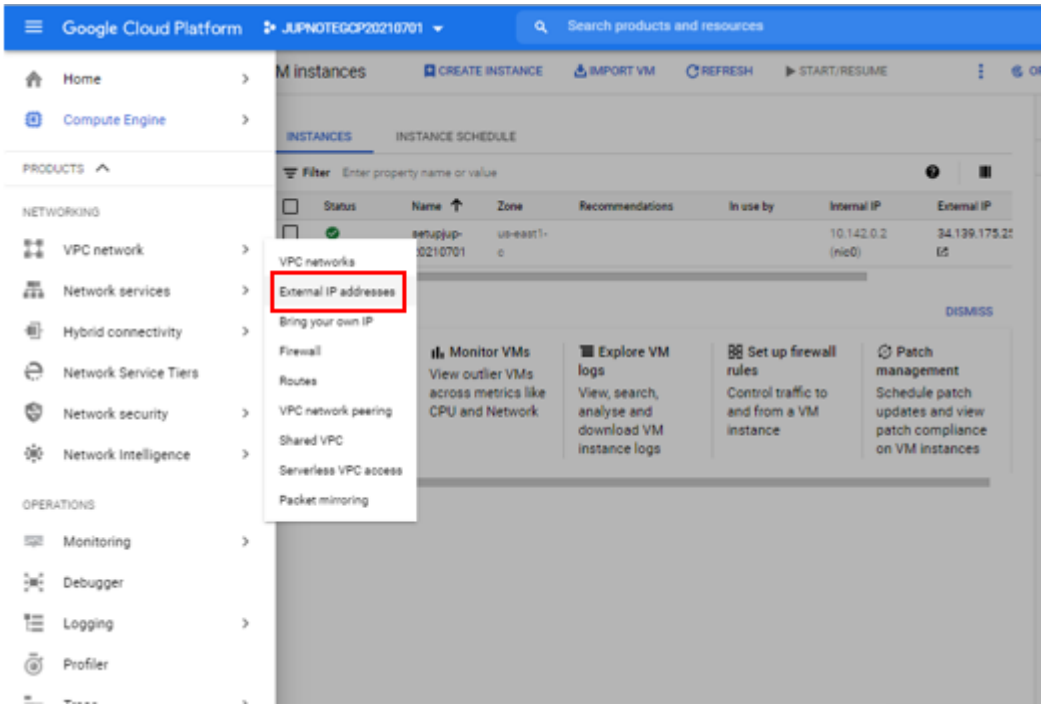


- Now your instance is ready!

- **Note:** When you are not using your instance, you can stop your instance to save money. You can stop your instance by clicking **Stop** present under three dots menu. ⋮

6.3. Making External IP as Static

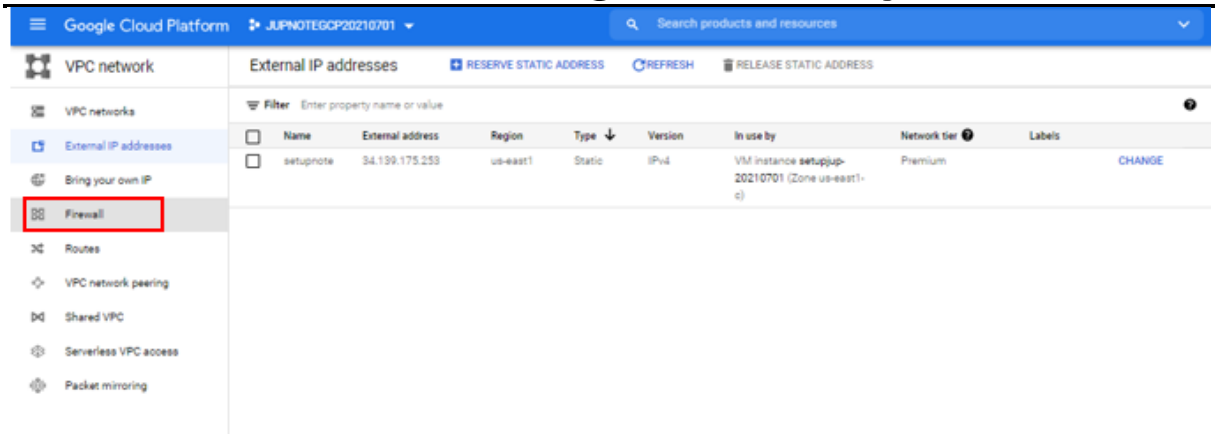
- In order to access the Jupyter notebook, we need to make External Ip as static.
- Click on the Navigation menu, Go to Networking > VPC Network > External IP addresses.



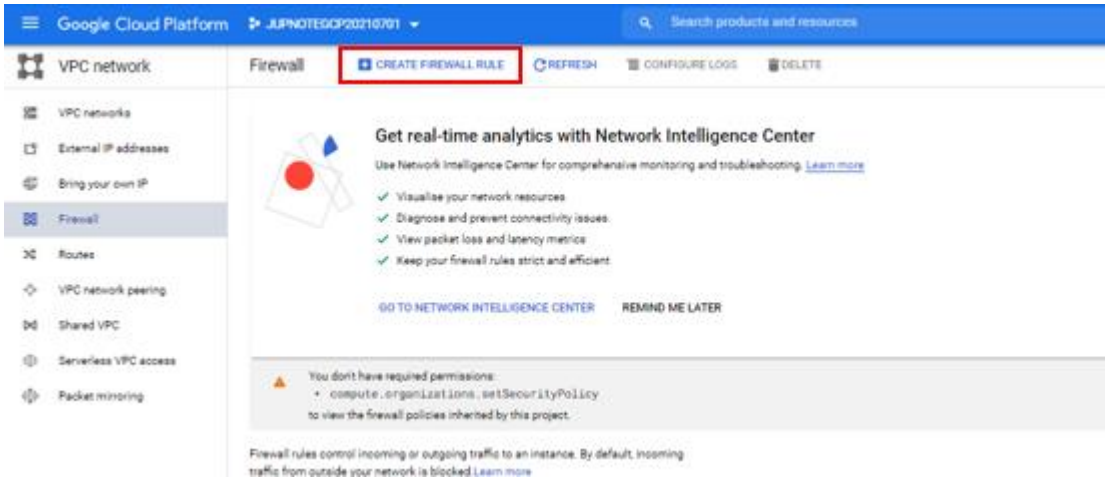
- You will see an external IP address of your instance, change the type from Ephemeral to Static.
- You will see a popup window where you need to provide a name for the new static IP address and click Reserve. You can give any name you want.

6.4. Create Firewall Rules

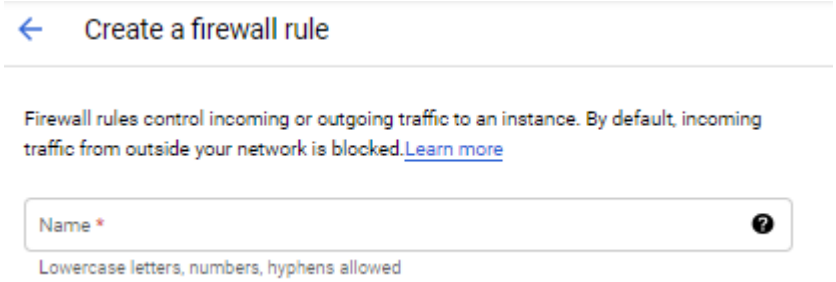
- On the same path, click on Firewall on the left side (Navigation menu > Networking > VPC Network > Firewall Rules)



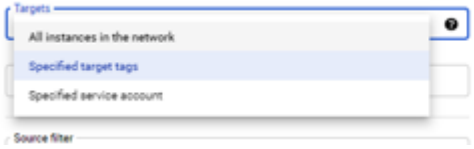
- Click on “CREATE FIREWALL RULE”



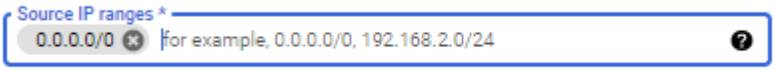
- Name: (Enter any name for your firewall)



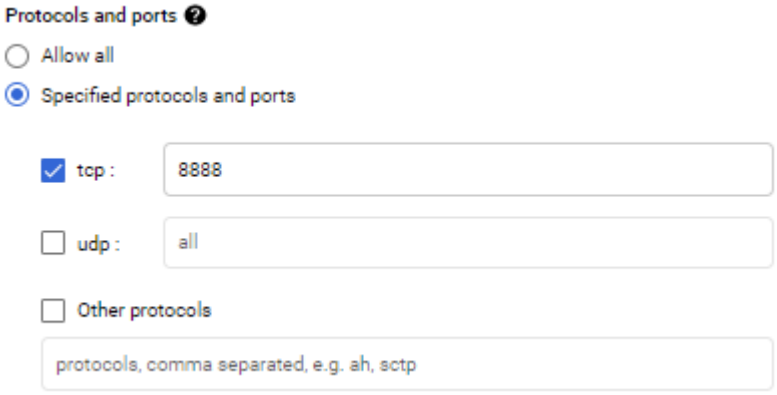
- Targets: Select “All instances in the network” from the drop down



- Source IP ranges: 0.0.0.0/0

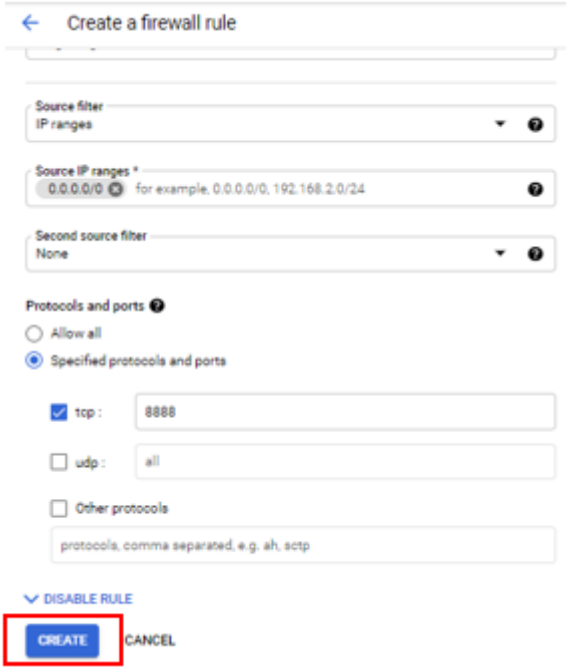


- Protocols and ports: Select “Specified protocols and ports” option and tcp: 8888.



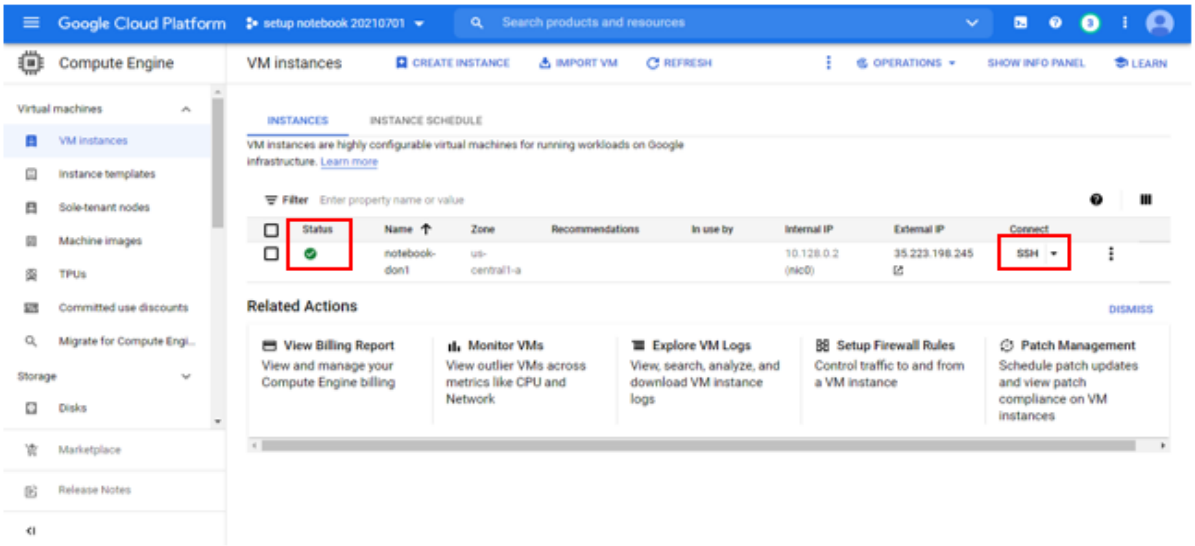
- Keep other configurations as default.

- Click on the Create button.

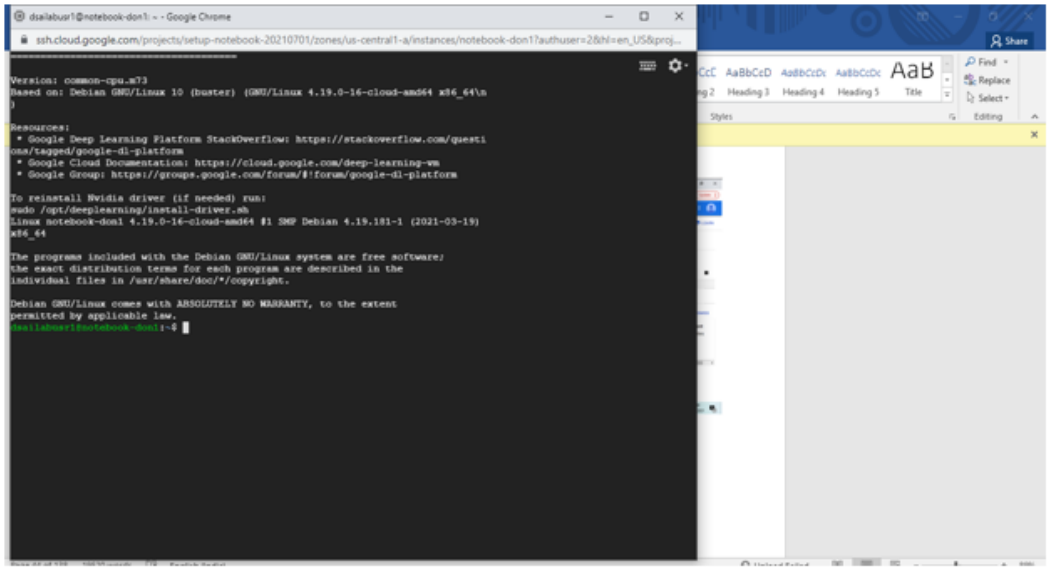


6.5. Install Anaconda in your VM Instance

- Go to the Navigation menu – Compute > Compute engine – VM instance. We can see that the status of the VM instance is active. Click on the SSH button of your instance.



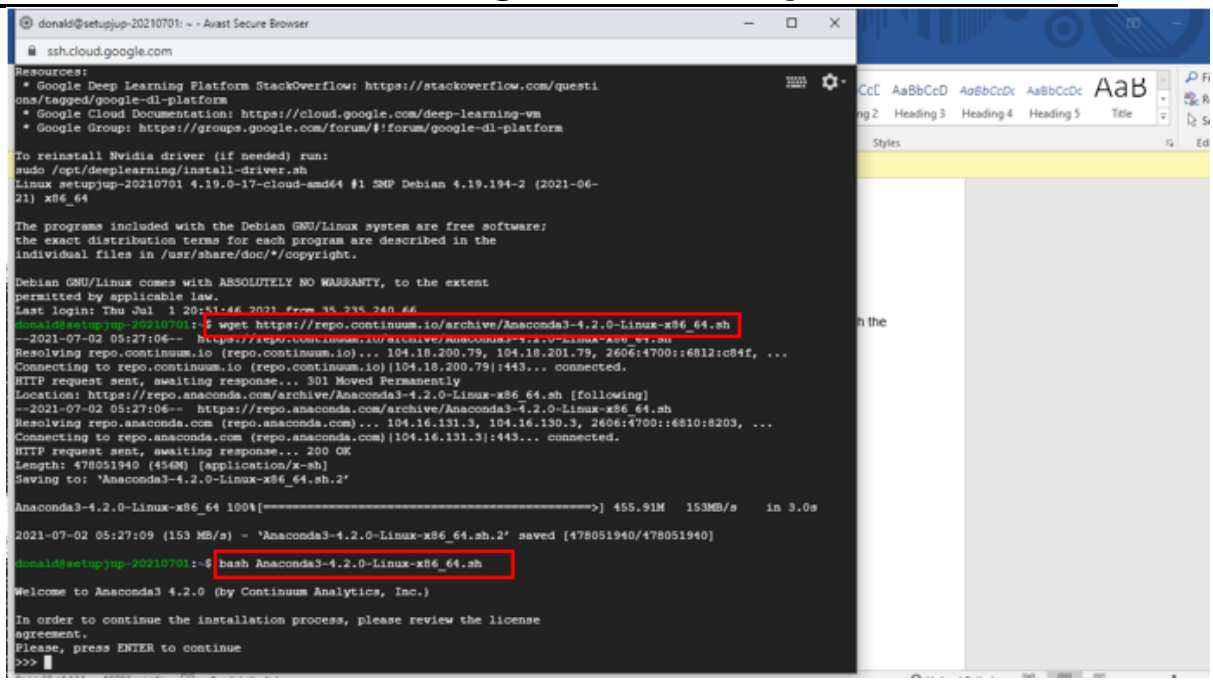
- SSH terminal will be open as shown below:



- In your SSH terminal, enter the commands below one by one:

wget https://repo.continuum.io/archive/Anaconda3-4.2.0-Linux-x86_64.sh

bash Anaconda3-4.2.0-Linux-x86_64.sh



- NOTE: If you get “Command not found” error, you need to install that specific command using below code:

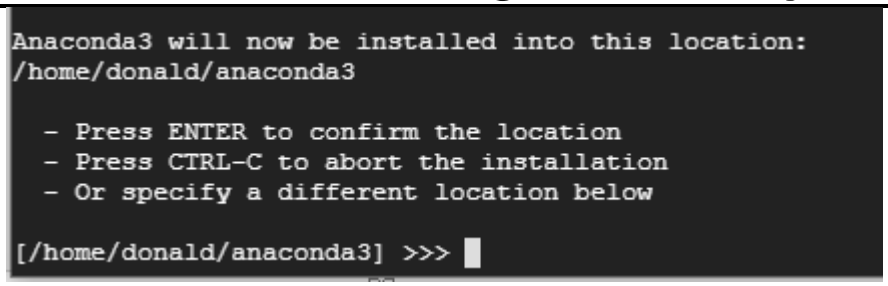
  sudo apt-get install COMMAND NAME

- For example, if you get “Command not found” error for “wget” command, you need to run this code:

  sudo apt-get install wget

- Press ENTER (multiple times) to review the license agreement and choose yes for question

- Now, the terminal shows the location where Anaconda will be installed as shown below:



- If you want to install Anaconda in the same location, press ENTER
- If you want to install Anaconda in a different location, specify the location
- If you want to abort the installation, press CTRL-C

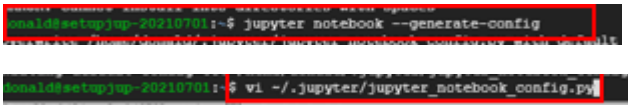
- After confirming the location, Anaconda will be installed with the libraries

6.6. Set up the VM Server

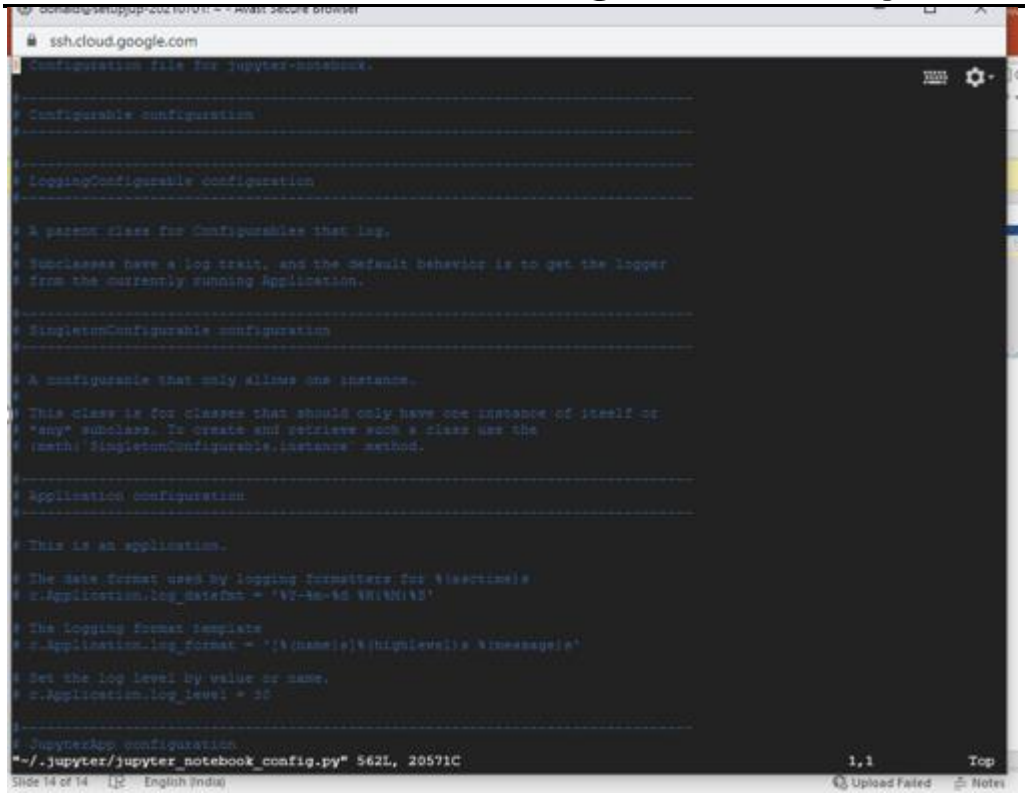
- Enter the commands below to create and configure the Jupyter configurations files:

  jupyter notebook --generate-config

  vi ~/.jupyter/jupyter_notebook_config.py



- Below configuration file will open:



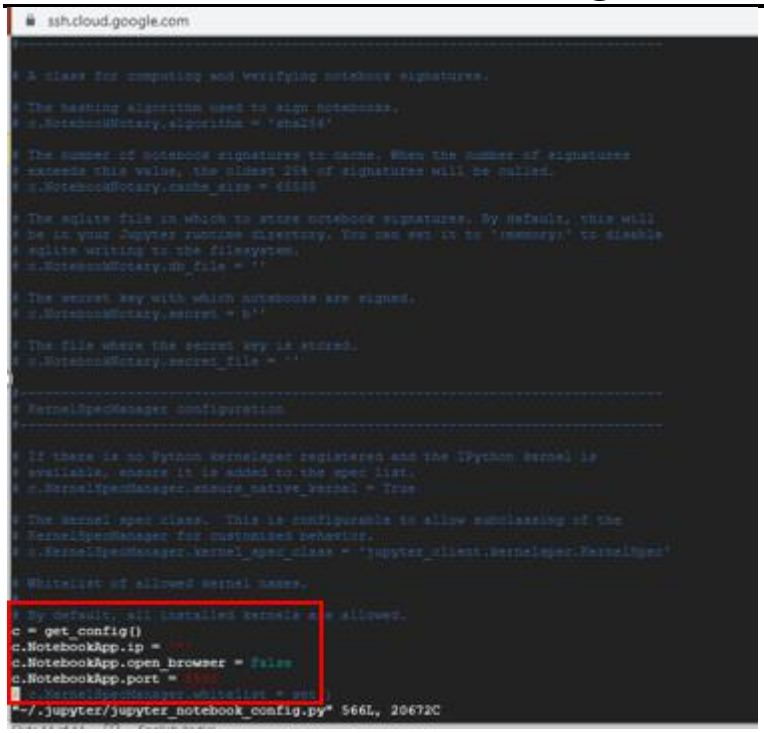
- Press “i” to insert and add the following lines in the configuration file:

  c = get_config()

  c.NotebookApp.ip = ''*

  c.NotebookApp.open_browser = False

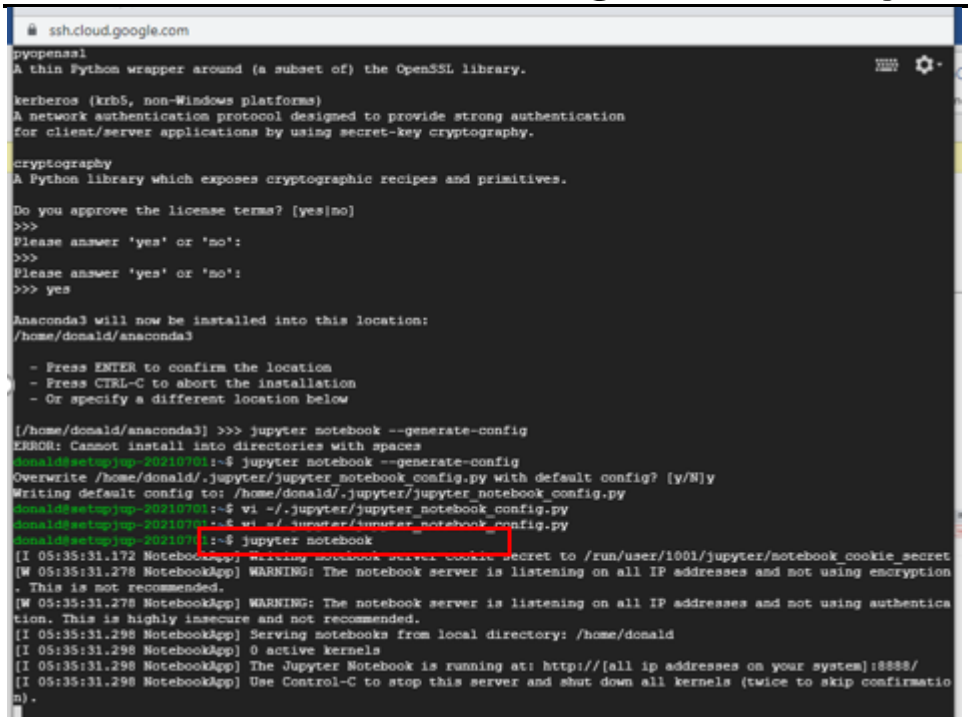
  c.NotebookApp.port = 8888



- Once you are done with making changes, press Esc and type *:wq* and press ENTER to save and exit from the file.

- Run the command below:

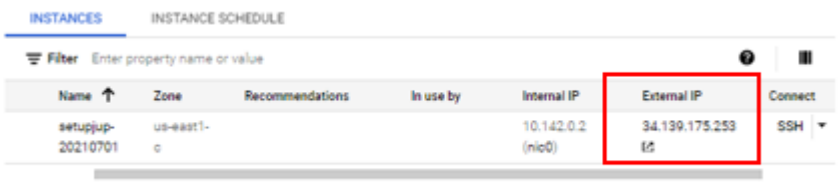
  jupyter notebook



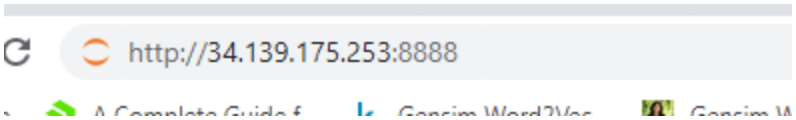
- Now, you can launch the notebook in your browser using the below url:

  http://<your external-ip-address>:<your port number without angular brackets>

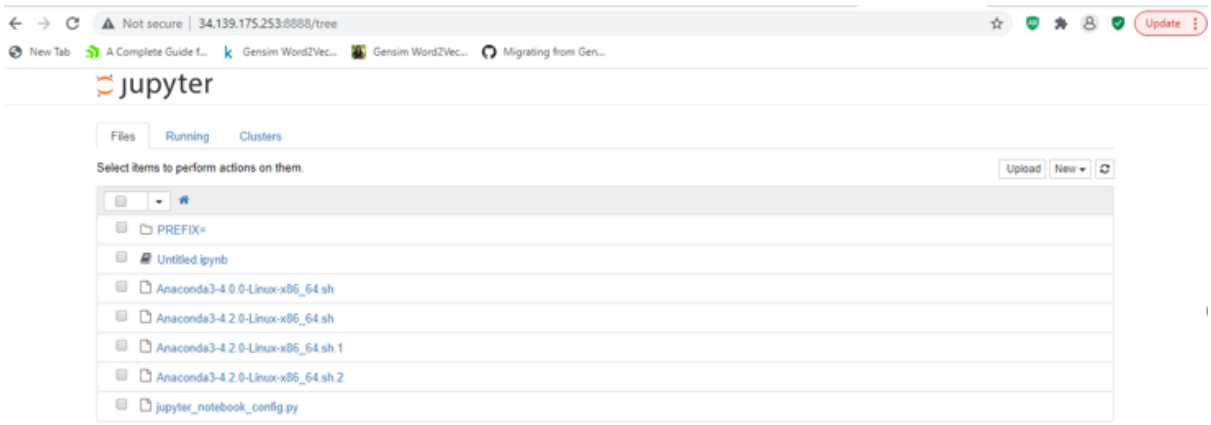
  - Your External ip address – Copy from this path (Go to Navigation menu > Compute engine > VM instances)



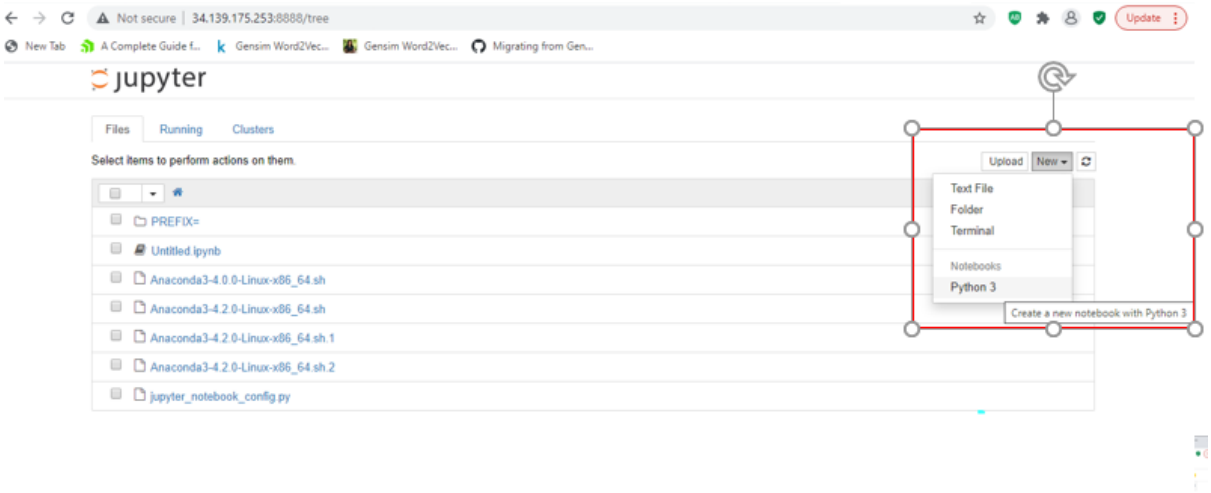
  - Your Port Number – 8888 (We've already it as 8888 in the previous step)



- Below page will open after entering the above url:



- Click on New and Python 3 to open the Jupyter Notebook:



6.7. Sample codes executed in the Notebook

6.7.1. How to Create a DataFrame

How to Create a DataFrame

In [2]:
```
#Importing pandas library
import pandas as pd

#Creating a dictionary
cars = {'Brand': ['Honda Civic','Toyota Corolla','Ford Focus','Audi A4'],
        'Price': [22000,25000,27000,35000]
        }
#Converting dictionary to dataframe and defining the column names
df = pd.DataFrame(cars, columns = ['Brand', 'Price'])
#printing the dataframe
print(df)
```

```
            Brand  Price
0     Honda Civic  22000
1  Toyota Corolla  25000
2      Ford Focus  27000
3         Audi A4  35000
```

6.7.2. How to Find Data type

How to Find Data type

In [3]:
```
#Finding data type
df.dtypes
```

Out[3]:
```
Brand    object
Price     int64
dtype: object
```

6.7.3. Convert Data type

Convert Data type

In [4]:
```
#Converting data format of Price column to string type
df['Price'] = df['Price'].astype(str)

#Finding data type
df.dtypes
```

Out[4]:
```
Brand    object
Price    object
dtype: object
```

6.7.4. How to Create a New Column

How to Create a New Column

In [5]:
```
#Converting Price column type to integer
df['Price'] = df['Price'].astype(int)

#Creating new column by adding 5000 to the price column
df['Price with tax'] = df['Price'] + 5000

print(df)
```

```
            Brand  Price  Price with tax
0     Honda Civic  22000           27000
1  Toyota Corolla  25000           30000
2      Ford Focus  27000           32000
3         Audi A4  35000           40000
```

6.7.5. Create “for” loop to print each letters of a word

Creating 'for' loop to print each letters of a word

```
In [6]: GC = "Google Cloud"
        for x in GC:
            print(x)
```

```
G
o
o
g
l
e

C
l
o
u
d
```

6.7.6. Apply Filters on Dataset

Applying filters on dataset

In [9]:
```
#First Creating DataFrame
ipl_data = {'Team': ['Riders', 'Riders', 'Devils', 'Devils', 'Kings', 'Kings', 'Kings', 'Kings', 'Riders', 'Royals', 'Royals', 'Riders'],
        'Rank': [1, 2, 2, 3, 3, 4, 1, 1, 2, 4, 1, 2],
        'Year': [2014, 2015, 2014, 2015, 2014, 2015, 2016, 2017, 2016, 2014, 2015, 2017],
        'Points': [876, 789, 863, 673, 741, 812, 756, 788, 694, 701, 804, 690]
        }

df = pd.DataFrame(ipl_data)
print(df)

#Applying Filter
Filtered_data = df[df['Year'] == 2014]
Filtered_data
```

```
    Points  Rank    Team  Year
0      876     1  Riders  2014
1      789     2  Riders  2015
2      863     2  Devils  2014
3      673     3  Devils  2015
4      741     3   Kings  2014
5      812     4   Kings  2015
6      756     1   Kings  2016
7      788     1   Kings  2017
8      694     2  Riders  2016
9      701     4  Royals  2014
10     804     1  Royals  2015
11     690     2  Riders  2017
```

Out[9]:

	Points	Rank	Team	Year
0	876	1	Riders	2014
2	863	2	Devils	2014
4	741	3	Kings	2014
9	701	4	Royals	2014

6.7.7. Sorting Dataset

Sorting dataset

In [10]:
```
#Sorting by single column
Filtered_data.sort_values(by = 'Points', ascending = False)
```

Out[10]:

	Points	Rank	Team	Year
0	876	1	Riders	2014
2	863	2	Devils	2014
4	741	3	Kings	2014
9	701	4	Royals	2014

In [11]:
```
#Sorting by multiple columns
Filtered_data.sort_values(by = ['Team', 'Rank'], ascending = False)
```

Out[11]:

	Points	Rank	Team	Year
9	701	4	Royals	2014
0	876	1	Riders	2014
4	741	3	Kings	2014
2	863	2	Devils	2014

6.7.8. Merging Datasets

Merging Datasets

In [16]:
```
#First Creating 2 dataframes
data1 = pd.DataFrame({
    'id': [1, 2, 3, 4, 5],
    'Name': ['Alex', 'Amy', 'Allen', 'Alice', 'Ayoung'],
    'subject_id': ['sub1', 'sub2', 'sub4', 'sub6', 'sub5']
})

data2 = pd.DataFrame({
    'id': [1, 2, 3, 4, 5],
    'Name': ['Billy', 'Brian', 'Brain', 'Bryce', 'Betty'],
    'subject_id': ['sub2', 'sub4', 'sub3', 'sub6', 'sub5']
})

print("data1: \n", data1.head(), "\n")
print("data2: \n", data2.head(), "\n")

#Merging 2 datasets using Inner Join with quotes on 'subject_id' column
Merged_data = pd.merge(data1, data2, on = 'subject_id', how = 'inner')
print("Merged Data: ")
Merged_data
```

```
data1:
      Name  id subject_id
0     Alex   1       sub1
1      Amy   2       sub2
2    Allen   3       sub4
3    Alice   4       sub6
4   Ayoung   5       sub5

data2:
     Name  id subject_id
0   Billy   1       sub2
1   Brian   2       sub4
2   Brain   3       sub3
3   Bryce   4       sub6
4   Betty   5       sub5

Merged Data:
```

Out[16]:

	Name_x	id_x	subject_id	Name_y	id_y
0	Amy	2	sub2	Billy	1
1	Allen	3	sub4	Brian	2
2	Alice	4	sub6	Bryce	4
3	Ayoung	5	sub5	Betty	5

6.8. How to run Scala codes using Jupyter Notebook

- To run Scala codes in Jupyter Notebook, we need to install Almond. Almond is a Scala kernel for Jupyter. It is formerly known as jupyter-scala.

- If not already done, install Anaconda and set up the VM server as instructed in these sections, 6.5 and 6.6 before installing the Scala kernel.

6.8.1. Install Almond

- Install Almond by running the following commands:

sudo apt-get update

```
donald@rstudio1:~$ sudo apt-get update
Hit:1 http://security.debian.org/debian-security buster/updates InRelease
Hit:2 http://deb.debian.org/debian buster InRelease
Hit:4 http://deb.debian.org/debian buster-updates InRelease
Hit:5 http://deb.debian.org/debian buster-backports InRelease
Hit:6 http://packages.cloud.google.com/apt cloud-sdk-buster InRelease
Hit:7 http://packages.cloud.google.com/apt google-cloud-packages-archive-keyring-buster InRelease
Get:8 http://packages.cloud.google.com/apt gcsfuse-buster InRelease [5385 B]
Hit:9 http://packages.cloud.google.com/apt google-compute-engine-buster-stable InRelease
Get:10 https://download.docker.com/linux/debian buster InRelease [54.0 kB]
Hit:3 https://packages.cloud.google.com/apt kubernetes-xenial InRelease
Fetched 59.3 kB in 2s (37.4 kB/s)
Reading package lists... Done
```

sudo apt-get install scala

```
donald@rstudio1:~$ sudo apt-get install scala
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  ca-certificates-java default-jre-headless java-common libhawtjni-runtime-java libjansi-java
  libjansi-native-java libjline2-java libnspr4 libnss3 libpcsclite1 openjdk-11-jre-headless scala-library
  scala-parser-combinators scala-xml
Suggested packages:
```

java -version

```
donald@rstudio1:~$ java -version
openjdk version "11.0.11" 2021-04-20
OpenJDK Runtime Environment (build 11.0.11+9-post-Debian-1deb10u1)
OpenJDK 64-Bit Server VM (build 11.0.11+9-post-Debian-1deb10u1, mixed mode, sharing)
```

curl -Lo coursier https://git.io/coursier-cli

chmod +x coursier

```
donald@rstudio1:~$ curl -Lo coursier https://git.io/coursier-cli
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100   135  100   135    0     0    233      0 --:--:-- --:--:-- --:--:--   233
100 42577  100 42577    0     0  45246      0 --:--:-- --:--:-- --:--:-- 45246
donald@rstudio1:~$ chmod +x coursier
```

./coursier launch --fork almond:0.10.0 --scala 2.12.11 -- --install

```
donald@rstudio1:~$ ./coursier launch --fork almond:0.10.0 --scala 2.12.11 -- --install
https://repo1.maven.org/maven2/io/get-coursier/apps/maven-metadata.xml
  100.0% [##########] 1.9 KiB (11.2 KiB / s)
https://repo1.maven.org/maven2/io/get-coursier/apps/1.0.13/apps-1.0.13.pom
  100.0% [##########] 1.3 KiB (17.0 KiB / s)
https://repo1.maven.org/maven2/sh/almond/scala-kernel_2.12.11/0.10.0/scala-kernel_2.12.11-0.10.0.pom
  100.0% [##########] 2.1 KiB (56.5 KiB / s)
https://repo1.maven.org/maven2/org/scala-lang/scala-library/2.12.11/scala-library-2.12.11.pom
  100.0% [##########] 1.6 KiB (41.2 KiB / s)
https://repo1.maven.org/maven2/sh/almond/kernel_2.12/0.10.0/kernel_2.12-0.10.0.pom
```

rm -f coursier

```
Installed scala kernel under /home/donald/.local/share/jupyter/kernels/scala
donald@rstudio1:~$ rm -f coursier
```

- To view the kernels installed, run this command:

jupyter kernelspec list

```
donald@rstudio1:~$ jupyter kernelspec list
[ListKernelSpecs] WARNING | Config option `kernel_spec_manager_class` not recognized by `ListKernelSpecs`.
Available kernels:
  scala      /home/donald/.local/share/jupyter/kernels/scala
  python3    /opt/conda/share/jupyter/kernels/python3
```

- To start the Jupyter Notebook session, run this command:

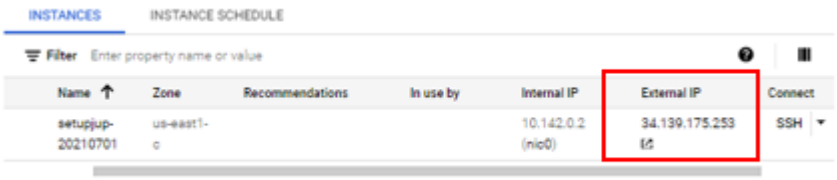
jupyter notebook

```
donald@rstudio1:~$ jupyter notebook
[I 11:05:52.708 NotebookApp] [nb_conda_kernels] enabled, 2 kernels found
[I 11:05:52.713 NotebookApp] Writing notebook server cookie secret to /run/user/1001/jupyter/notebook_cookie_secret
[W 11:05:52.885 NotebookApp] WARNING: The notebook server is listening on all IP addresses and not using encryption
. This is not recommended.
[W 11:05:52.885 NotebookApp] WARNING: The notebook server is listening on all IP addresses and not using authentica
tion. This is highly insecure and not recommended.
[I 11:05:52.947 NotebookApp] [nb_conda] enabled
[I 11:05:53.391 NotebookApp] ✓ nbpresent HTML export ENABLED
[W 11:05:53.391 NotebookApp] ✗ nbpresent PDF export DISABLED: No module named 'nbbrowserpdf'
[I 11:05:53.923 NotebookApp] [nb_anacondacloud] enabled
[I 11:05:53.925 NotebookApp] Serving notebooks from local directory: /home/donald
[I 11:05:53.926 NotebookApp] 0 active kernels
[I 11:05:53.926 NotebookApp] The Jupyter Notebook is running at: http://[all ip addresses on your system]:8686/
[I 11:05:53.926 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmatio
n).
```

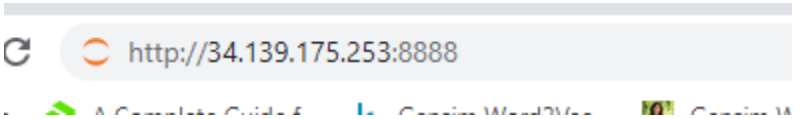
- Now, you can launch the notebook in your browser using the below url:

  http://<your external-ip-address>:<your port number without angular brackets>

  - Your External ip address – Copy from this path (Go to Navigation menu > Compute engine > VM instances)

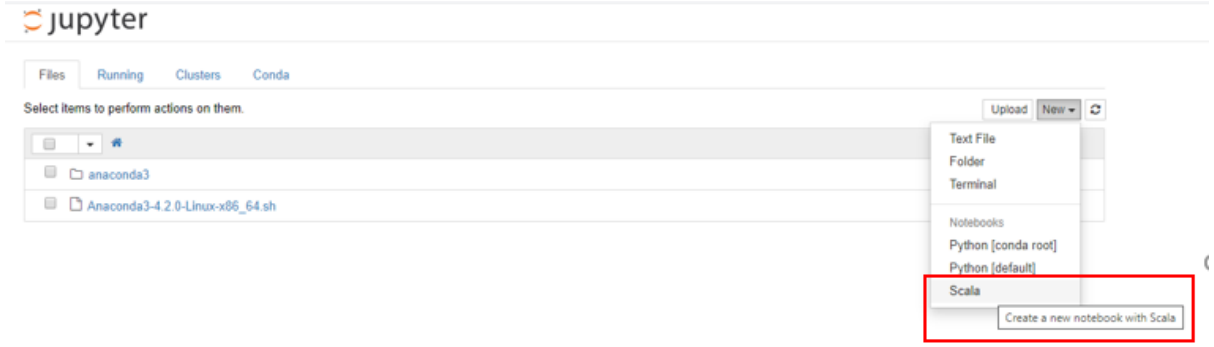


  - Your Port Number – 8888 (We've already it as 8888 in the previous step)



- Below page will open after entering the above url:

- Click on New and Scala to open the Jupyter Notebook with Scala kernel:



6.8.2. Sample Scala Codes

println("Hello, World!")

```
In [1]: println("Hello, World!")
        Hello, World!
```

for (i <- 1 to 5) println(i)

```
In [2]: for (i <- 1 to 5) println(i)

1
2
3
4
5
```

```
def myFunction(a:Int, b:Int): Int = {
 val c = a*b
 return c
}
myFunction(2,4)
```

```
In [3]: def myFunction(a:Int, b:Int): Int = {
          val c = a*b
          return c
        }
        myFunction(2,4)

Out[3]: defined function myFunction
        res2_1: Int = 8
```

```
println({
 val a = 2*3
 a+4
})
```

```
In [4]: println({
          val a = 2*3
          a+4
        })

10
```

var a : Int = 6
var b : Int = 10
var c : Int = 41

```
In [5]: var a : Int = 6
        var b : Int = 10
        var c : Int = 41

        a: Int = 6
        b: Int = 10
        c: Int = 41
```

val y = Array("England","Liberia","Haiti","Australia","Sweden")
y.sorted

```
In [6]: val y = Array("England","Liberia","Haiti","Australia","Sweden")
        y.sorted

Out[6]: y: Array[String] = Array("England", "Liberia", "Haiti", "Australia", "Sweden")
        res5_1: Array[String] = Array(
          "Australia",
          "England",
          "Haiti",
          "Liberia",
          "Sweden"
        )
```

val capitals= Map("Argentina" -> "Buenos Aires", "Canada" -> "Ottowa","Egypt" -> "Cairo", "Liberia"-> "Morovia", "Netherland"->"Amstradam","United Stats"->"Washington d.c")

```
In [7]: val capitals= Map("Argentina"  -> "Buenos Aires", "Canada" -> "Ottowa","Egypt" -> "Cairo", "Liberia"-> "Morovia",
                         "Netherland"->"Amstradam","United Stats"->"Washington d.c")

Out[7]: capitals: Map[String, String] = Map(
          "Argentina" -> "Buenos Aires",
          "Egypt" -> "Cairo",
          "Canada" -> "Ottowa",
          "Liberia" -> "Morovia",
          "Netherland" -> "Amstradam",
          "United Stats" -> "Washington d.c"
        )
```

capitals.keys

```
In [8]: capitals.keys

Out[8]: res7: Iterable[String] = Set(
          "Argentina",
          "Egypt",
          "Canada",
          "Liberia",
          "Netherland",
          "United Stats"
        )
```

capitals("Canada")

```
In [9]: capitals("Canada")

Out[9]: res8: String = "Ottowa"
```

6.9. How to run R codes using Jupyter Notebook

6.9.1. Install R Kernel

- If not already done, install Anaconda and set up the VM server as instructed here, 6.5 and 6.6 before installing the R kernel.

- To install R kernel in the Jupyter Notebook, run the following commands:

conda install -c r r-irkernel

```
(base) donald@rstudio:~$ conda install -c r r-irkernel
Collecting package metadata (current_repodata.json): done
Solving environment: done


==> WARNING: A newer version of conda exists. <==
  current version: 4.9.2
  latest version: 4.10.3

Please update conda by running

    $ conda update -n base conda



## Package Plan ##

  environment location: /opt/conda

  added / updated specs:
    - r-irkernel


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    _r-mutex-1.0.1             |      anacondar_1           3 KB  conda-forge
    binutils_impl_linux-64-2.36.1|      h193b22a_1         9.7 MB  conda-forge
    binutils_linux-64-2.36     |     hf3e587d_31          22 KB  conda-forge
    bwidget-1.9.14             |      ha770c72_0         119 KB  conda-forge
    curl-7.77.0                |      hea6ffbf_0         149 KB  conda-forge
    font-ttf-dejavu-sans-mono-2.37|      hab24e00_0         388 KB  conda-forge
    font-ttf-inconsolata-3.000 |      h77eed37_0          94 KB  conda-forge
    font-ttf-source-code-pro-2.038|      h77eed37_0         684 KB  conda-forge
    font-ttf-ubuntu-0.83       |      hab24e00_0         1.9 MB  conda-forge
    fonts-conda-ecosystem-1    |                0           4 KB  conda-forge
```

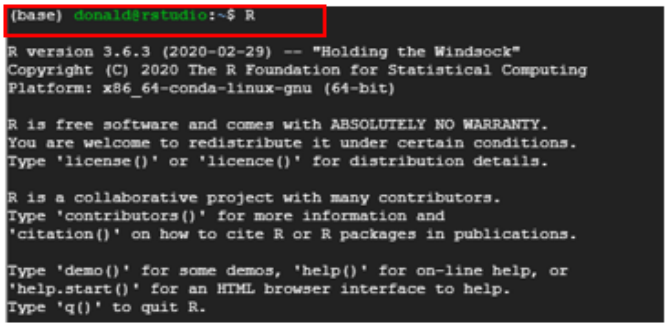
- You'll get the following screen after running the above command:

```
  pango                     conda-forge/linux-64::pango-1.48.7-hb8ff022_0
  r-base                    conda-forge/linux-64::r-base-3.6.3-hbcea092_8
  r-base64enc               r/linux-64::r-base64enc-0.1_3-r36h96ca727_4
  r-crayon                  r/noarch::r-crayon-1.3.4-r36h6115d3f_0
  r-digest                  r/linux-64::r-digest-0.6.18-r36h96ca727_0
  r-evaluate                r/noarch::r-evaluate-0.13-r36h6115d3f_0
  r-htmltools               r/linux-64::r-htmltools-0.3.6-r36h29659fb_0
  r-irdisplay               r/noarch::r-irdisplay-0.7.0-r36h6115d3f_0
  r-irkernel                r/noarch::r-irkernel-0.8.15-r36_0
  r-jsonlite                r/linux-64::r-jsonlite-1.6-r36h96ca727_0
  r-pbdzmq                  r/linux-64::r-pbdzmq-0.3_3-r36h5455479_1
  r-rcpp                    r/linux-64::r-rcpp-1.0.1-r36h29659fb_0
  r-repr                    r/noarch::r-repr-0.19.2-r36h6115d3f_0
  r-uuid                    r/linux-64::r-uuid-0.1_2-r36h96ca727_4
  sed                       conda-forge/linux-64::sed-4.8-he412f7d_0
  sysroot_linux-64          conda-forge/noarch::sysroot_linux-64-2.12-h77966d4_13
  tktable                   conda-forge/linux-64::tktable-2.10-hb7b940f_3

The following packages will be UPDATED:

  ld_impl_linux-64                    2.35.1-hea4e1c9_2 --> 2.36.1-hea4e1c9_1


Proceed ([y]/n)?
```

- Type 'y' and then press ENTER to proceed:

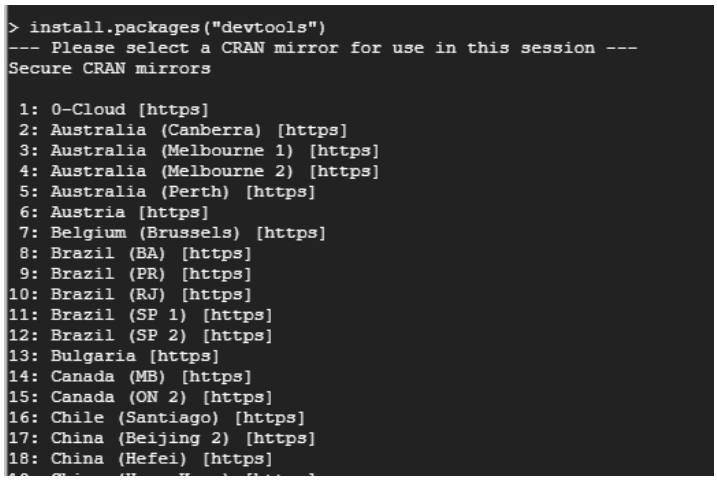
- After a short period of time your installation will be completed as shown below.

- Now, open R session by typing “R” and pressing ENTER:

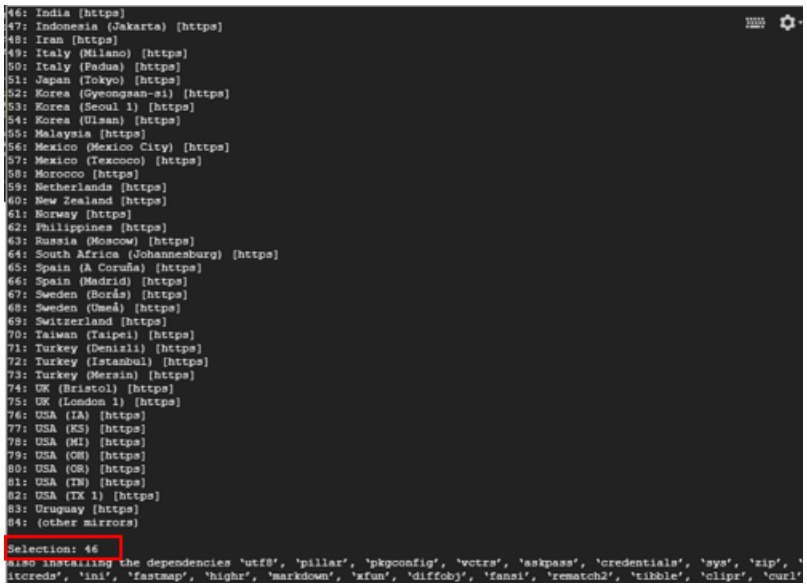


- Run the following command to install the required packages:

install.packages("devtools")



- Select a CRAN mirror for use in this session. In my case, I've selected "46" (India) as shown below:



- NOTE: It might take more time to install so please wait until it gets installed

- After the packages get installed, run this command to install the kernel:

IRkernel::installspec()

```
> IRkernel::installspec()
[InstallKernelSpec] Installed kernelspec ir in /home/donald/.local/share/jupyter/kernels/ir
```

- Quit from the R session - Type “q()” and press ENTER.
  Save the workspace - Type “y” and press ENTER:

```
> IRkernel::installspec()
[InstallKernelSpec] Installed kernelspec ir in /home/donald/.local/share/jupyter/kernels/ir
> q()
Save workspace image? [y/n/c]: y
```

- To view the list of kernels installed, run this command:

jupyter kernelspec list

```
donald@rstudio:~$ jupyter kernelspec list
Available kernels:
  python3    /home/donald/anaconda3/lib/python3.5/site-packages/ipykernel/resources
  ir         /home/donald/.local/share/jupyter/kernels/ir
```

- R kernel has installed, now we open the Jupyter Notebook

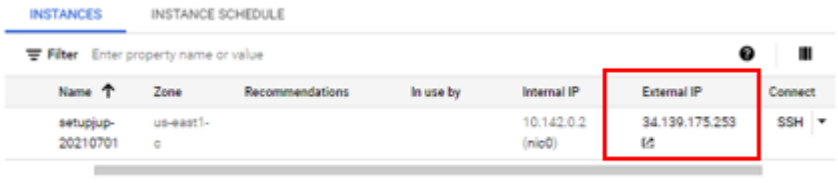
jupyter notebook

```
donald@rstudio:~$ jupyter notebook
[I 13:28:29.924 NotebookApp] [nb_conda_kernels] enabled, 2 kernels found
[I 13:28:29.931 NotebookApp] Writing notebook server cookie secret to /run/user/1001/jupyter/notebook_cookie_secret
```

- Now, you can launch the notebook in your browser using the below url:

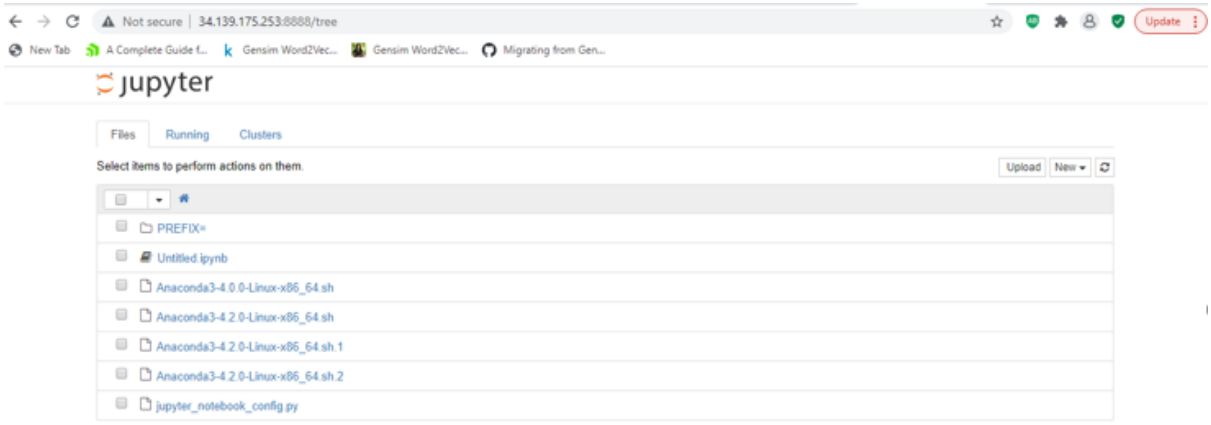
  http://<your external-ip-address>:<your port number without angular brackets>

  - Your External ip address – Copy from this path (Go to Navigation menu > Compute engine > VM instances)



  - Your Port Number – 8888 (We’ve already it as 8888 in the previous step)

- Below page will open after entering the above url:



- Click on New and R to open the Jupyter Notebook with R kernel:

6.9.2. Sample R Codes

```
x = c(1,2,3)
y = c(4,5,6)
x+y
```

```
In [1]: #Creating and adding vectors
        x = c(1,2,3)
        y = c(4,5,6)
        x+y
```

```
5  7  9
```

```
data <- data.frame(
  emp_id = c (1:5),
  emp_name = c("Rick","Dan","Michelle","Ryan","Gary"),
  salary = c(623.3,515.2,611.0,729.0,843.25),
  stringsAsFactors = FALSE
)
data
```

```
In [2]: #Creating a data frame
        data <- data.frame(
           emp_id = c (1:5),
           emp_name = c("Rick","Dan","Michelle","Ryan","Gary"),
           salary = c(623.3,515.2,611.0,729.0,843.25),
           stringsAsFactors = FALSE
        )
        data
```

emp_id	emp_name	salary
1	Rick	623.30
2	Dan	515.20
3	Michelle	611.00
4	Ryan	729.00
5	Gary	843.25

```
survey <- data.frame("index" = c(1, 2, 3, 4, 5),
                     "age" = c(24, 25, 42, 56, 22))
survey$sex <- c("m", "m", "f", "f", "m")
survey
```

```
In [3]: #Adding new column
survey <- data.frame("index" = c(1, 2, 3, 4, 5),
                     "age" = c(24, 25, 42, 56, 22))
survey$sex <- c("m", "m", "f", "f", "m")
survey
```

index	age	sex
1	24	m
2	25	m
3	42	f
4	56	f
5	22	m

```
survey$newcol <- survey$age + 10
survey
```

```
In [4]: #Adding new calculated column
survey$newcol <- survey$age + 10
survey
```

index	age	sex	newcol
1	24	m	34
2	25	m	35
3	42	f	52
4	56	f	66
5	22	m	32

```
fruit <- c('Apple', 'Orange', 'Passion fruit', 'Banana')
# Create the for statement
for ( i in fruit){
print (i)
}
```

```
In [5]: #Creating 'for' loop
fruit <- c('Apple', 'Orange', 'Passion fruit', 'Banana')
# Create the for statement
for ( i in fruit){
print (i)
}

[1] "Apple"
[1] "Orange"
[1] "Passion fruit"
[1] "Banana"
```

print(survey)
subset(survey, sex != "m")

```
In [6]: #Creating subset by filtering a dataset
print(survey)
subset(survey, sex != "m")

  index age sex newcol
1     1  24   m     34
2     2  25   m     35
3     3  42   f     52
4     4  56   f     66
5     5  22   m     32
```

	index	age	sex	newcol
3	3	42	f	52
4	4	56	f	66

survey$age

```
In [7]: #Filter single column
survey$age

24 25 42 56 22
```

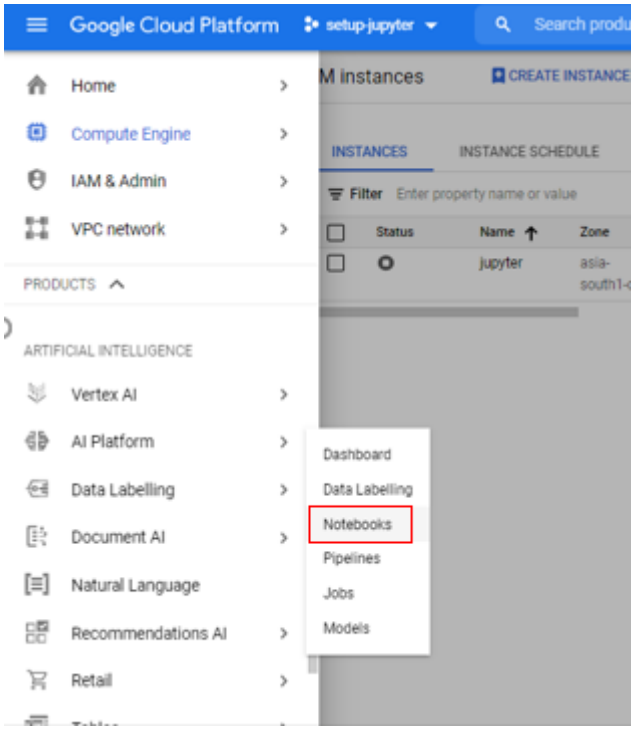
min(survey$age)

```
In [8]: #Finding minimum value
        min(survey$age)

        22
```

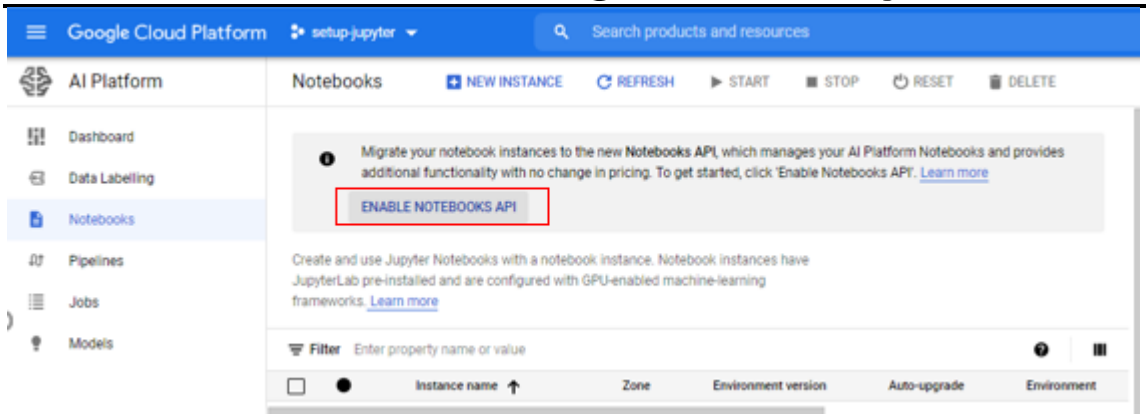
7. How to run Jupyter Notebook using Notebook API

- GCP AI Platform Notebook is a Jupyter Notebook on Google Cloud Platform, with all the popular libraries and dependencies like pandas, scikit-learn and TensorFlow preinstalled.

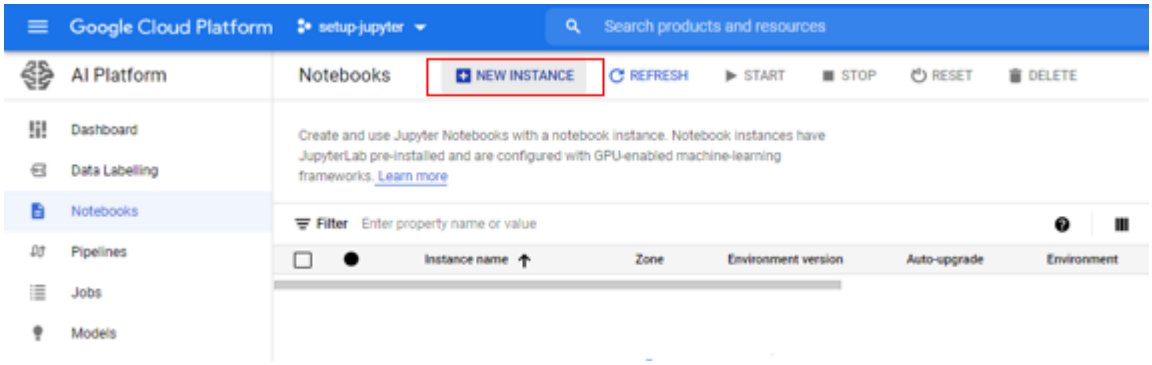
- Navigation menu > Artificial Intelligence > AI Platform > Notebooks



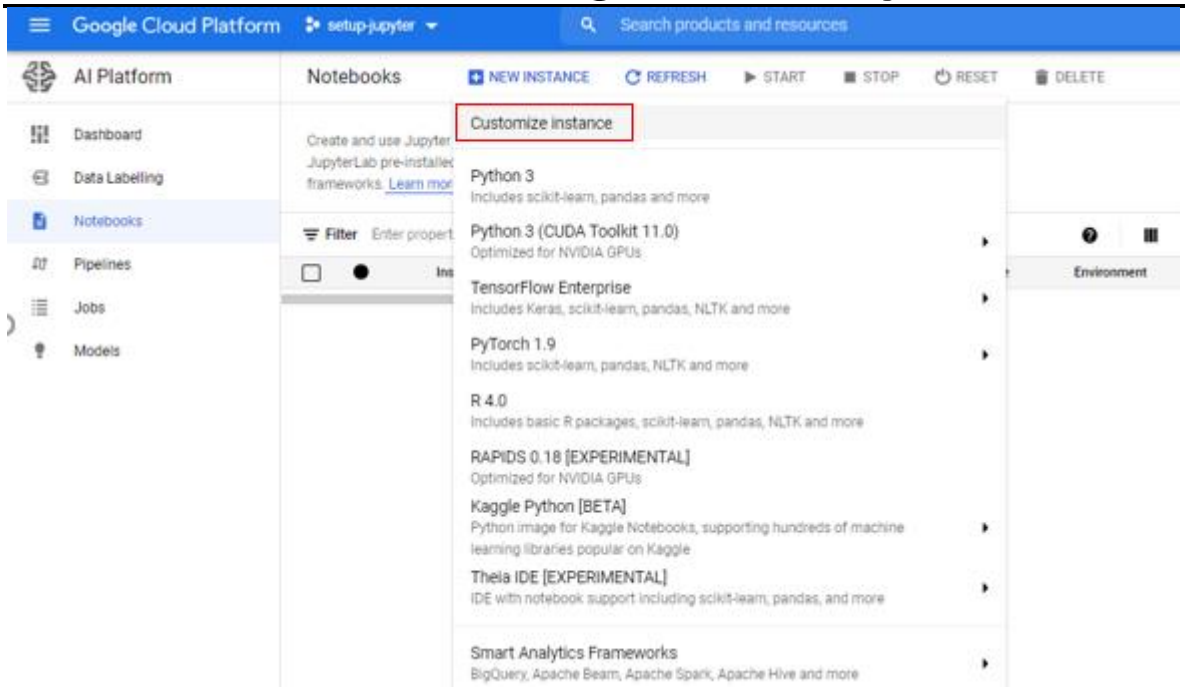
- Enable Notebooks API as shown below



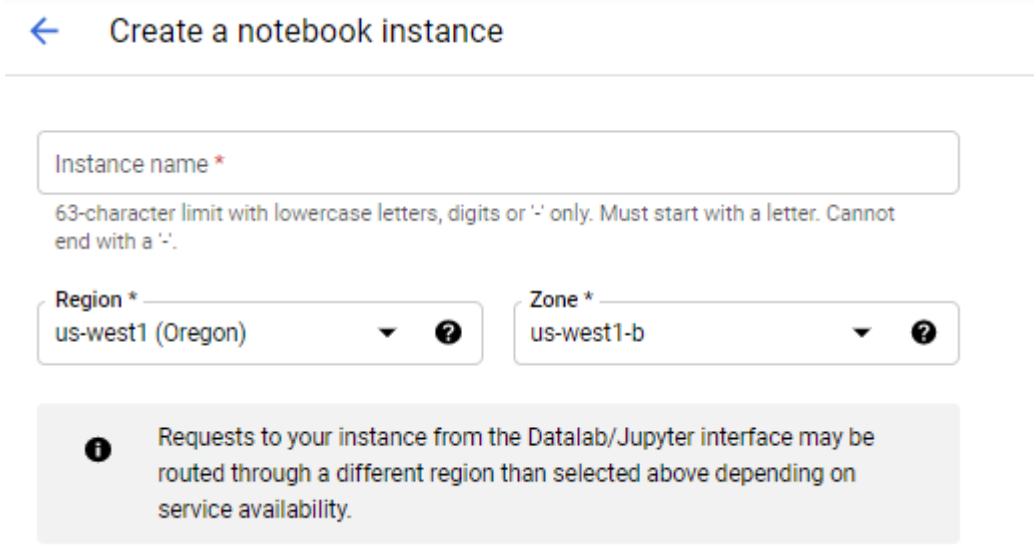
- Click on NEW INSTANCE



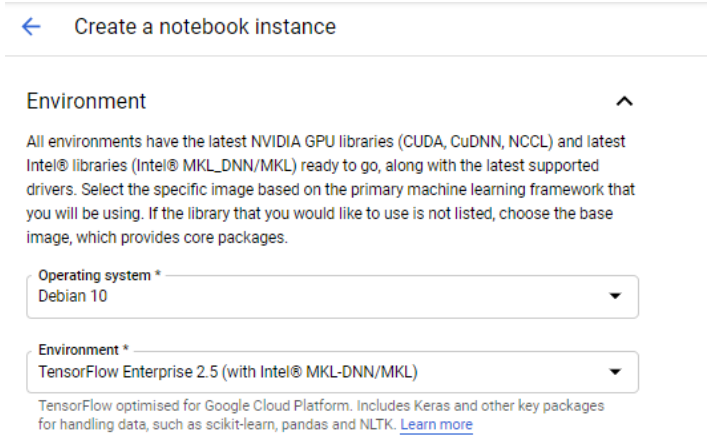
- Click on Customize Instance



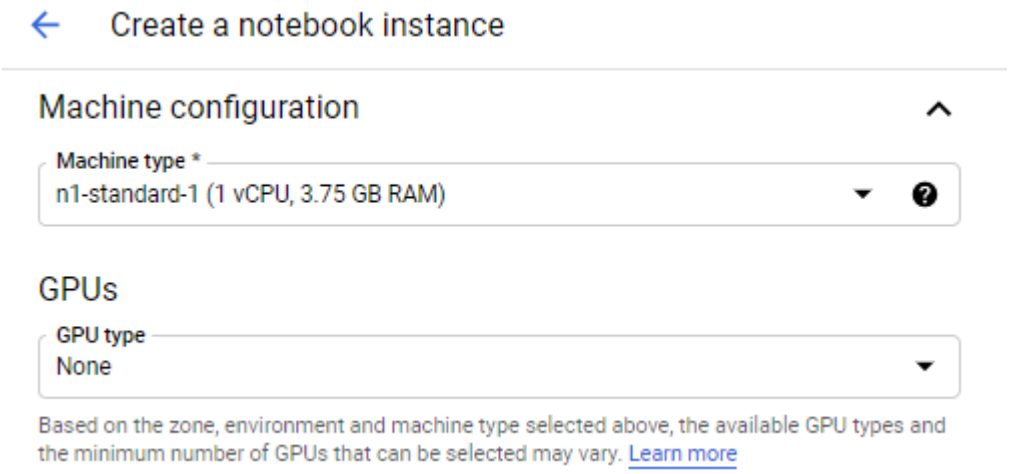
- You can choose the operating system, environment, machine configuration, boot disk size, RAM, GPU type, etc.
- You can see the hourly price of running your note on the right side.
- Select Region and Zone



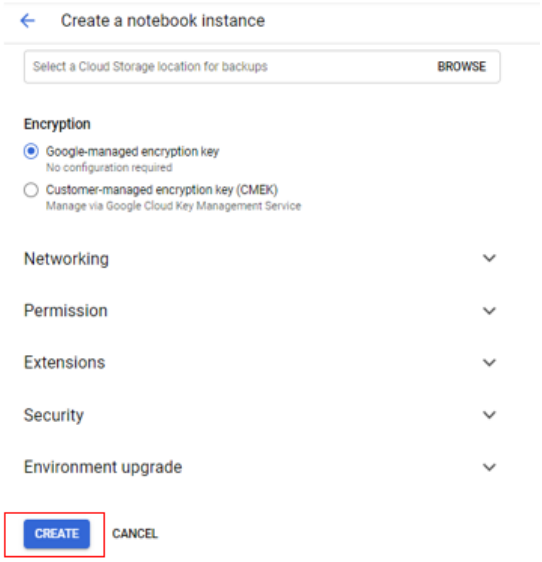
- Select Operating system and Environment



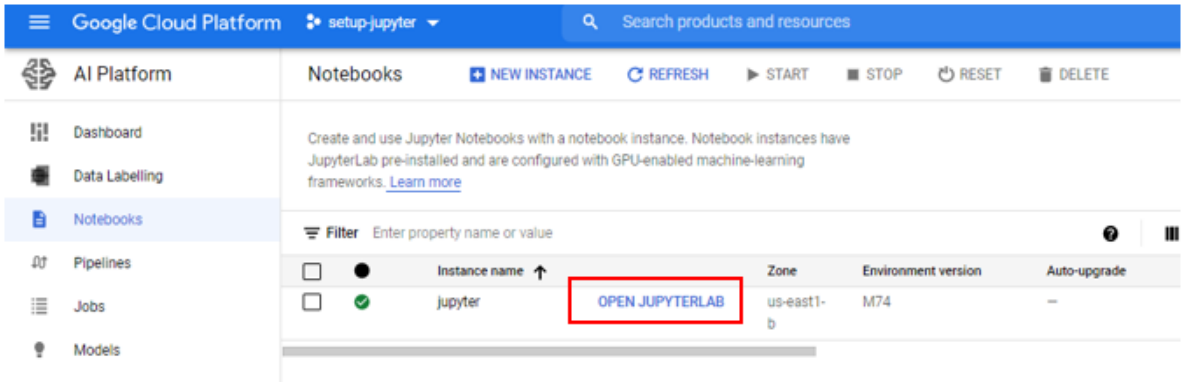
- Select Machine type and GPU type



- Keep other configurations as default or select the options according to your preference and click create.

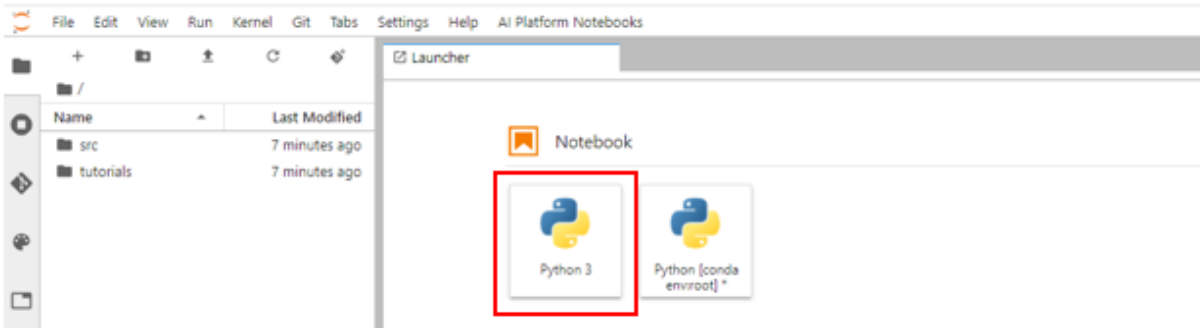


- Now the instance has been created. Click on “OPEN JUPYTERLAB”

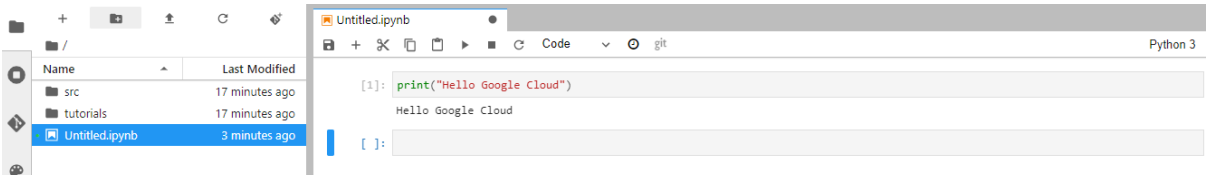


- You can upload existing notebooks or you can create one by selecting options under the notebook menu.

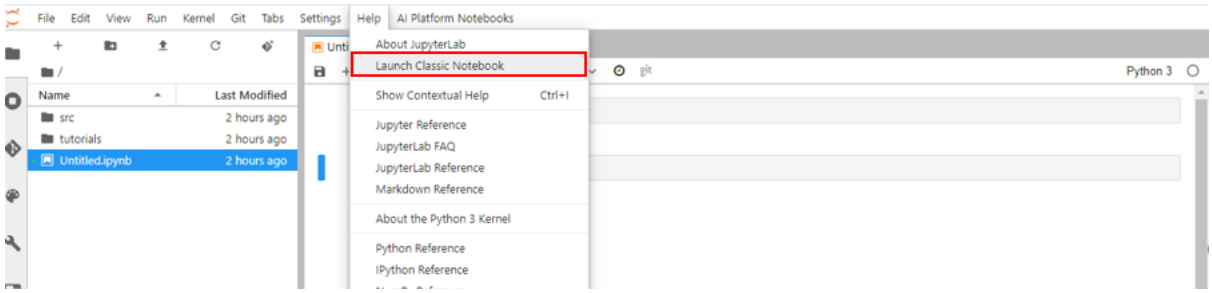
- Click on “Python 3”



- You can start running the codes



- We can also switch to the Jupyter Notebook. Go to Help > Launch Classic Notebook



- NOTE: Do not forget to Delete your notebook after completing the execution.

8. Google Cloud Resources

- Google Cloud consists of a set of physical assets, such as computers and hard disk drives, and virtual resources, such as virtual machines (VMs), that are contained in Google's data centers around the globe.
- Each data center location is in a region. Regions are available in Asia, Australia, Europe, North America, and South America.
- Each region is a collection of zones, which are isolated from each other within the region.
- Each zone is identified by a name that combines a letter identifier with the name of the region. For example, zone a in the East Asia region is named asia-east1-a.

8.1. Ways to Interact with Resources and Services

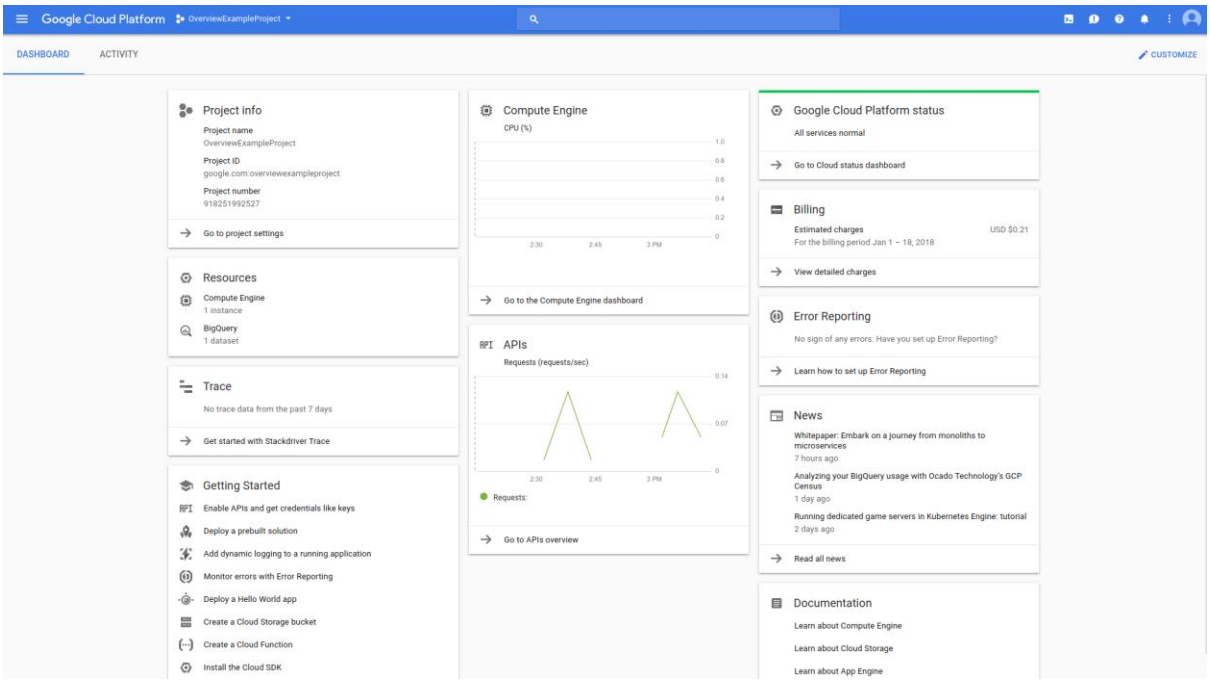
- Google Cloud gives you three basic ways to interact with the services and resources.

1. Google Cloud Console
2. Command-line interface
3. Client libraries

8.1.1. Google Cloud Console

- The Google Cloud Console provides a web-based, graphical user interface that you can use to manage your Google Cloud projects and resources.

- When you use the Cloud Console, you create a new project, or choose an existing project, and use the resources that you create in the context of that project. You can create multiple projects, so you can use projects to separate your work in whatever way makes sense for you.



8.1.2. Command-line Interface

- If you prefer to work at the command line, you can perform most Google Cloud tasks by using the gcloud command-line tool. The gcloud tool lets you manage development workflow and Google Cloud resources in a terminal window.

- You can run gcloud commands in the following ways:

- You can install the Cloud SDK. The SDK includes the gcloud tool, so you can open a terminal window on your own computer and run commands to manage Google Cloud resources.

- You can use Cloud Shell, which is a browser-based shell. Because it runs in a browser window, you don't need to install anything on your own computer. You can open the Cloud Shell from the Google Cloud Console.

- Cloud Shell provides the following:

  - A temporary Compute Engine virtual machine instance.

  - A built-in code editor.

  - 5 GB of persistent disk storage.

  - Pre-installed Cloud SDK and other tools.

  - Language support for Java, Go, Python, Node.js, PHP, Ruby and .NET.

  - Web preview functionality.

  - Built-in authorization for access to Cloud Console projects and resources.

8.1.3. Client Libraries

- The Cloud SDK includes client libraries that enable you to easily create and manage resources. Google Cloud client libraries expose APIs for two main purposes:

    - App APIs provide access to services. App APIs are optimized for supported languages, such as Node.js and Python. The libraries are designed around service metaphors, so you can work with the services more naturally and write less boilerplate code. The libraries also provide helpers for authentication and authorization.

    - Admin APIs offer functionality for resource management. For example, you can use admin APIs if you want to build your own automated tools.

- You also can use the Google API client libraries to access APIs for products such as Maps, Drive, and YouTube.

9. Google Cloud Services

Some of the key services offered by Google Cloud are listed below:

- Computing and hosting
- Storage
- Databases
- Networking
- Big data
- Machine learning

9.1. Computing and hosting services

- Google Cloud gives you options for computing and hosting. You can choose to do the following:

- Work in a serverless environment.
- Use a managed application platform.
- Leverage container technologies to gain lots of flexibility.
- Build your own cloud-based infrastructure to have the most control and flexibility.

9.1.1. Serverless Computing

- Cloud Functions, Google Cloud's **functions as a service (FaaS)** offering, provides a serverless execution environment for building and connecting cloud services.
- With Cloud Functions you write simple, single-purpose functions that are attached to events emitted from your cloud infrastructure and services. Your function is triggered when an event being watched is fired.
- Your code executes in a fully managed environment. There is no need to provision any infrastructure or worry about managing any servers.
- Cloud Functions can be written using
    - JavaScript,
    - Python 3,
    - Go, or
    - Java.
- You can take your function and run it in any standard Node.js (Node.js 10), Python 3 (Python 3.7), Go (Go 1.11 or 1.13) or Java (Java 11) environment, which makes both portability and local testing a breeze.
- Cloud Functions are a good choice for use cases that include the following:
    - Data processing and ETL operations, for scenarios such as video transcoding and IoT streaming data.
    - Webhooks to respond to HTTP triggers.

- Lightweight APIs that compose loosely coupled logic into applications.
- Mobile backend functions.

9.1.2. Application Platform

- App Engine is Google Cloud's **platform as a service (PaaS)**. With App Engine, Google handles most of the management of the resources for you.
- App Engine is a fully managed, serverless platform for developing and hosting web applications at scale. You can choose from several popular languages, libraries, and frameworks to develop your apps, then let App Engine take care of provisioning servers and scaling your app instances based on demand.
- When you build your app on App Engine, you can:
  - Build your app in Go, Java, .NET, Node.js, PHP, Python, or Ruby and use pre-configured runtimes, or use custom runtimes to write code in any language.
  - Let Google manage app hosting, scaling, monitoring, and infrastructure for you.
  - Connect with Google Cloud storage products, such as Cloud SQL, Firestore in Datastore mode, and Cloud Storage. You can also connect to managed Redis databases, and host third-party databases such as MongoDB and Cassandra on Compute Engine, another cloud provider, on-premises, or with a third-party vendor.
  - Use Web Security Scanner to identify security vulnerabilities as a complement to your existing secure design and development processes.
- You can run your applications in App Engine using
  - Flexible environment
  - Standard environment.
- You can also choose to simultaneously use both environments for your application and allow your services to take advantage of each environment's individual benefits.

9.1.2.1. When to choose the Standard Environment

- Application instances run in a sandbox, using the runtime environment of a supported language listed below.
- Applications that need to deal with rapid scaling.
- The standard environment is optimal for applications with the following characteristics:
- Source code is written in specific versions of the supported programming languages:
    - Python 2.7, Python 3.7, Python 3.8, Python 3.9
    - Java 8, Java 11
    - Node.js 8, Node.js 10, Node.js 12, and Node.js 14
    - PHP 5.5, PHP 7.2, PHP 7.3, and PHP 7.4
    - Ruby 2.5, Ruby 2.6, and Ruby 2.7
    - Go 1.11, Go 1.12, Go 1.13, Go 1.14, Go 1.15
- Intended to run for free or at very low cost, where you pay only for what you need and when you need it. For example, your application can scale to 0 instances when there is no traffic.
- Experiences sudden and extreme spikes of traffic which require immediate scaling.

9.1.2.2. When to choose the Flexible Environment

- Application instances run within Docker containers on Compute Engine virtual machines (VM).
- Applications that receive consistent traffic, experience regular traffic fluctuations, or meet the parameters for scaling up and down gradually.
- The flexible environment is optimal for applications with the following characteristics:
- Source code that is written in a version of any of the supported programming languages:

- Python,
- Java,
- Node.js,
- Go,
- Ruby,
- PHP, or
- .NET

- Runs in a Docker container that includes a custom runtime or source code written in other programming languages.

- Uses or depends on frameworks that include native code.

- Accesses the resources or services of your Google Cloud project that reside in the Compute Engine network.

9.1.3. Containers

- With container-based computing, you can focus on your application code, instead of on deployments and integration into hosting environments.

- Google Kubernetes Engine (GKE), Google Cloud's **containers as a service (CaaS)** offering, is built on the open source Kubernetes system, which gives you the flexibility of on-premises or hybrid clouds, in addition to Google Cloud's public cloud infrastructure.

- When you build with GKE, you can:

  - Create and manage groups of Compute Engine instances running Kubernetes, called clusters. GKE uses Compute Engine instances as nodes in a cluster. Each node runs the Docker runtime, a Kubernetes node agent that monitors the health of the node, and a simple network proxy.

  - Declare the requirements for your Docker containers by creating a simple JSON configuration file.

  - Use Container Registry for secure, private storage of Docker images. You can push images to your registry and then you can pull images to any Compute Engine instance or your own hardware by using an HTTP endpoint.

- Create single- and multi-container pods. Each pod represents a logical host that can contain one or more containers. Containers in a pod work together by sharing resources, such as networking resources. Together, a set of pods might comprise an entire application, a microservice, or one layer in a multi-tier application.

- Create and manage replication controllers, which manage the creation and deletion of pod replicas based on a template. Replication controllers help to ensure that your application has the resources it needs to run reliably and scale appropriately.

- Create and manage services. Services create an abstraction layer that decouples frontend clients from pods that provide backend functions. In this way, clients can work without concerns about which pods are being created and deleted at any given moment.

- Create an external network load balancer.

9.1.4. Virtual Machines

- Google Cloud's unmanaged compute service is Compute Engine. You can think of Compute Engine as providing an **infrastructure as a service (IaaS)**, because the system provides a robust computing infrastructure, but you must choose and configure the platform components that you want to use.

- With a Compute Engine, it's your responsibility to configure, administer, and monitor the systems. Google will ensure that resources are available, reliable, and ready for you to use, but it's up to you to provision and manage them. The advantage here is that you have complete control of the systems and unlimited flexibility.

- When you build on Compute Engine, you can do the following:

    - Use virtual machines (VMs), called instances, to build your application, much like you would if you had your own hardware infrastructure. You can choose from a variety of instance types to customize your configuration to meet your needs and your budget.

    - Choose which global regions and zones to deploy your resources in, giving you control over where your data is stored and used.

- Choose which operating systems, development stacks, languages, frameworks, services, and other software technologies you prefer.
- Create instances from public or private images.
- Use Google Cloud storage technologies or any third-party technologies you prefer.
- Use Google Cloud Marketplace to quickly deploy pre-configured software packages. For example, you can deploy a LAMP or MEAN stack with just a few clicks.
- Create instance groups to more easily manage multiple instances together.
- Use autoscaling with an instance group to automatically add and remove capacity.
- Attach and detach disks as needed.
- Use SSH to connect directly to your instances.

Combining computing and hosting options

- You don't have to stick with just one type of computing service. For example, you can combine App Engine and Compute Engine to take advantage of the features and benefits of each.

9.2. Storage Services

- Whatever your application, you'll probably need to store some media files, backups, or other file-like objects. Google Cloud provides a variety of storage services, including:
  - Consistent, scalable, large-capacity data storage in Cloud Storage. Cloud Storage comes in several flavors:
    - Standard Cloud Storage provides maximum availability.
    - Cloud Storage Nearline provides low-cost archival storage ideal for data accessed less than once a month.

- Cloud Storage Coldline provides even lower-cost archival storage ideal for data accessed less than once a quarter.

- Cloud Storage Archive provides the lowest-cost archival storage for backup and disaster recovery ideal for data you intend to access less than once a year.

  - Persistent disks on Compute Engine, for use as primary storage for your instances. Compute Engine offers both hard-disk-based persistent disks, called standard persistent disks, and solid-state persistent disks (SSD).

  - Fully managed NFS file servers in Filestore. You can use Filestore instances to store data from applications running on Compute Engine VM instances or GKE clusters.

9.3. Database Services

- Google Cloud provides a variety of SQL and NoSQL database services:
  - A SQL database in Cloud SQL, which provides either MySQL or PostgreSQL databases.
  - A fully managed, mission-critical, relational database service in Cloud Spanner that offers transactional consistency at global scale, schemas, SQL querying, and automatic, synchronous replication for high availability.
  - Two options for NoSQL data storage: Firestore, for document-like data, and Cloud Bigtable, for tabular data.
- You can also choose to set up your preferred database technology on Compute Engine by using persistent disks. For example, you can set up MongoDB for NoSQL document storage.

9.4. Networking Services

- While App Engine manages networking for you, and GKE uses the Kubernetes model, Compute Engine provides a set of networking services. These services help you to load-balance traffic across resources, create DNS records, and connect your existing network to Google's network.

9.4.1. Networks, Firewalls and routes

- Virtual Private Cloud (VPC) provides a set of networking services that your VM instances use. An instance can have more than one interface, but each interface must be connected to a different network. Every VPC project has a default network. You can create additional networks in your project, but networks cannot be shared between projects.

- Firewall rules govern traffic coming into instances on a network. The default network has a default set of firewall rules, and you can create custom rules, too.

- A route lets you implement more advanced networking functions in your instances, such as creating VPNs. A route specifies how packets leaving an instance should be directed. For example, a route might specify that packets destined for a particular network range should be handled by a gateway virtual machine instance that you configure and operate.

9.4.2. Load Balancing

- If your website or application is running on Compute Engine, the time might come when you're ready to distribute the workload across multiple instances. Server-side load balancing features provide you with the following options:

    - Network load balancing lets you distribute traffic among server instances in the same region based on incoming IP protocol data, such as address, port, and protocol. Network load balancing is a great solution if, for example, you want to meet the demands of increasing traffic to your website.

    - HTTP(S) load balancing enables you to distribute traffic across regions so you can ensure that requests are routed to the closest region or, in the event of a failure or over-capacity limitations, to a healthy instance in the next closest region. You can also use HTTP(S) load balancing to distribute traffic based on content type. For example, you might set up your servers to deliver static content, such as images and CSS, from one server and dynamic content, such as PHP pages, from a different server. The load balancer can direct each request to the server that provides each content type.

9.4.3. Cloud DNS

- You can publish and maintain Domain Name System (DNS) records by using the same infrastructure that Google uses. You can use the Google Cloud Console, the command line, or a REST API to work with managed zones and DNS records.

9.4.4. Advanced Connectivity

- If you have an existing network that you want to connect to Google Cloud resources, Google Cloud offers the following options for advanced connectivity:
    - Cloud Interconnect
    - Cloud VPN
    - Direct Peering
    - Carrier Peering

9.4.4.1. Cloud Interconnect

- Cloud Interconnect enables you to connect your existing network to your VPC network through a highly available, low-latency, enterprise-grade connection. You can use Dedicated Interconnect to connect directly to Google, or use Partner Interconnect to connect to Google through a supported service provider.
- Cloud Interconnect offers two options for extending your on-premises network:
    - Dedicated Interconnect provides a direct physical connection between your on-premises network and Google's network.
    - Partner Interconnect provides connectivity between your on-premises and VPC networks through a supported service provider.

9.4.4.2. Cloud VPN

- Cloud VPN enables you to connect your existing network to your VPC network through an IPsec connection. You can also use VPN to connect two Cloud VPN gateways to each other.

9.4.4.3. Direct Peering

- Direct Peering enables you to establish a direct peering connection between your business network and Google's edge network and exchange high-throughput cloud traffic.

- This capability is available at any of more than 100 locations in 33 countries around the world.

- When established, Direct Peering provides a direct path from your on-premises network to Google services, including Google Cloud products that can be exposed through one or more public IP addresses. Traffic from Google's network to your on-premises network also takes that direct path, including traffic from VPC networks in your projects. Google Cloud customers must request that direct egress pricing be enabled for each of their projects after they have established Direct Peering with Google.

9.4.4.4. Carrier Peering

- Carrier Peering enables you to access Google applications, such as Google Workspace, by using a service provider to obtain enterprise-grade network services that connect your infrastructure to Google.

- When connecting to Google through a service provider, you can get connections with higher availability and lower latency, using one or more links. Work with your service provider to get the connection that you need.

9.5. Big Data Services

- Big data services enable you to process and query big data in the cloud to get fast answers to complicated questions.

9.5.1. Data Analysis

- BigQuery provides data analysis services. With BigQuery, you can:
    - Create custom schemas that organize your data into datasets and tables.
    - Load data from a variety of sources, including streaming data.
    - Use SQL-like commands to query massive datasets very quickly. BigQuery is designed and optimized for speed.
    - Use the web UI, command-line interface, or API.

- Load, query, export, and copy data by using jobs.
- Manage data and protect it by using permissions.

9.5.2. Batch and Streaming Data Processing

- Dataflow provides a managed service and set of SDKs that you can use to perform batch and streaming data processing tasks. Dataflow works well for high-volume computation, especially when the processing tasks can clearly and easily be divided into parallel workloads.
- Dataflow is also great for extract-transform-load (ETL) tasks, which are useful for moving data between different storage media, transforming data into a more desirable format, or loading data onto a new storage system.

9.5.3. Asynchronous Messaging

- Pub/Sub is an asynchronous messaging service. Your application can send messages as JSON data structures to a publishing unit called a topic. Because Pub/Sub topics are a global resource, other applications in projects that you own can subscribe to the topic to receive the messages in HTTP request or response bodies.
- Pub/Sub's usefulness isn't confined to big data. You can use Pub/Sub in many circumstances where you need an asynchronous messaging service.

9.6. Machine Learning Services

- AI Platform offers a variety of powerful machine learning (ML) services. You can choose to use APIs that provide pre-trained models optimized for specific applications, or build and train your own large-scale, sophisticated models using a managed TensorFlow framework.

9.6.1. Machine Learning APIs

- Google Cloud offers a variety of APIs that enable you to take advantage of Google's ML without creating and training your own models.

- **Video Intelligence API** lets you use video analysis technology that provides label detection, explicit content detection, shot-change detection, and regionalization features.

- **Speech-to-Text** lets you convert audio to text, recognizing over 110 languages and variants, to support your global user base. You can transcribe the text of users dictating to an application's microphone, enable command-and-control through voice, or transcribe audio files, among other use cases.

- **Cloud Vision** lets you easily integrate vision detection features, including image labeling, face and landmark detection, optical character recognition (OCR), and tagging of explicit content.

- **Cloud Natural Language API** lets you add sentiment analysis, entity analysis, entity-sentiment analysis, content classification, and syntax analysis.

- **Cloud Translation** lets you quickly translate source text into any of over a hundred supported languages. Language detection helps out in cases where the source language is not known.

- **Dialogflow** lets you build conversational interfaces for websites, mobile applications, popular messaging platforms, and IoT devices. You can use it to build interfaces, such as chatbots, that are capable of natural and rich interactions with humans.

9.6.2. AI Platform

- AI Platform combines the managed infrastructure of Google Cloud with the power and flexibility of TensorFlow. You can use it to train your machine learning models at scale, and to host trained models to make predictions about new data in the cloud.

- AI Platform enables you to train machine learning models by running TensorFlow training applications on Google Cloud, and hosts those trained models for you, so you can use them to get predictions about new data.

- AI Platform manages the computing resources that your training job needs to run, so you can focus more on your model than on hardware configuration or resource management.

10. Google Cloud Products

10.1. Featured Products

10.1.1. Compute Engine

- Compute Engine is a computing and hosting service that lets you create and run virtual machines on Google infrastructure.
- Compute Engine offers scale, performance, and value that lets you easily launch large compute clusters on Google's infrastructure. There are no upfront investments, and you can run thousands of virtual CPUs on a system that offers quick, consistent performance.

10.1.1.1. Virtual machine instances

- You can create an instance by using the Google Cloud Console, the gcloud command-line tool, or the Compute Engine API.
- Compute Engine instances can run the public images for Linux and Windows Server that Google provides as well as private custom images that you can create or import from your existing systems. You can also deploy Docker containers, which are automatically launched on instances running the Container-Optimized OS public image.
- You can choose the machine properties of your instances, such as the number of virtual CPUs and the amount of memory, by using a set of predefined machine types or by creating your own custom machine types.

10.1.1.1.1. Instances and projects

- Each instance belongs to a Google Cloud Console project, and a project can have one or more instances. When you create an instance in a project, you specify the zone, operating system, and machine type of that instance. When you delete an instance, it is removed from the project.

10.1.1.1.2. Instances and storage options

- By default, each Compute Engine instance has a small boot persistent disk that contains the operating system. When applications running on your instance require more storage space, you can add additional storage options to your instance.

10.1.1.1.3. Instances and networks

- Each network interface of a Compute Engine instance is associated with a subnet of a unique VPC network. For more information about VPCs, see VPC network overview and VPC quotas.

10.1.1.1.4. Instances and containers

- Compute Engine instances support a declarative method for launching your applications using containers. When creating a VM or an instance template, you can provide a Docker image name and launch configuration. Compute Engine will take care of the rest including supplying an up-to-date Container-Optimized OS image with Docker installed and launching your container when the VM starts up.

10.1.1.1.5. Tools to manage instances

- To create and manage instances, you can use a variety of tools, including the Google Cloud Console, the gcloud command-line tool, and the REST API. To configure applications on your instances, connect to the instance using Secure Shell (SSH) for Linux instances or Remote Desktop Protocol (RDP) for Windows Server instances.

10.1.1.1.6. Managing access to your instances

- You can manage access to your instances using one of the following methods:

- Linux instances:

- Managing Instance Access Using OS Login, which allows you to associate SSH keys with your Google Account or Google Workspace account and manage admin or non-admin access to instances through IAM roles. If you connect to your instances using the gcloud command-line tool or SSH from the console, Compute Engine can automatically generate SSH keys for you and apply them to your Google Account or Google Workspace account.

- Manage your SSH keys in project or instance metadata, which grants admin access to instances with metadata access that do not use OS Login. If you connect to your instances using the gcloud command-line tool or SSH from the console, Compute Engine can automatically generate SSH keys for you and apply them to project metadata.

- On Windows Server instances:

  - Create a password for a Windows Server instance

  - Windows Server instances use password authentication instead of SSH authentication. To prevent unauthorized access to new Windows instances, Compute Engine requires that you generate a new Windows password for that instance before you connect to it. If you forget your password, you can generate a new one using this same process.

10.1.1.2. VM instance life cycle

- A VM instance can transition through many states as part of its life cycle. When you create a VM, Compute Engine provisions resources to start the VM. Next, the VM moves into staging, where it prepares for first boot. During and after start-up, a VM is considered running. During its lifetime, a running VM can be repeatedly stopped and restarted or suspended and resumed.

- A VM can be in one of the following states:

  - PROVISIONING: resources are allocated for the VM. The VM is not running yet.

  - STAGING: resources are acquired, and the VM is preparing for first boot.

  - RUNNING: the VM is booting up or running.

- STOPPING: the VM is being stopped. You requested a stop, or a failure occurred. This is a temporary status after which the VM enters the TERMINATED status.
- REPAIRING: the VM is being repaired. Repairing occurs when the VM encounters an internal error or the underlying machine is unavailable due to maintenance. During this time, the VM is unusable. If repair succeeds, the VM returns to one of the above states.
- TERMINATED: the VM is stopped. You stopped the VM, or the VM encountered a failure. You can restart or delete the VM.
- SUSPENDING: The VM is in the process of being suspended. You suspended the VM.
- SUSPENDED: The VM is in a suspended state. You can resume the VM or delete it.

10.1.1.2.1. Stopping, suspending, or resetting a VM

- You can stop or suspend a VM if you no longer need it but want to keep the VM for future use. Your decision to stop or suspend a VM depends on whether you need to preserve its guest OS memory and application state.
- When you suspend or stop a VM, consider the following:
  - You do not incur costs for a suspended or stopped VM.
  - You pay for resources that are still attached to a suspended or stopped VM, such as static IPs and persistent disk data.
  - Ephemeral external IP addresses are released from the VM. When you restart or resume the VM, Google assigns new ephemeral external IP addresses. If you depend on an ephemeral external IP address remaining the same through restarts, promote the ephemeral external IP.
  - Static external IP addresses are retained.

10.1.1.2.1.1. Stopping a VM

- Stopping a VM causes the Compute Engine to send the ACPI shutdown signal to the VM. Modern guest operating systems are configured to perform a clean shutdown before

powering off in response to the power off signal. Compute Engine waits a short time for the guest to finish shutting down and then transitions the VM to the TERMINATED state.

- You might want to stop a VM for several reasons:
    - You no longer need the VM but want the resources that are attached to the VM—such as its internal IPs, MAC address, and persistent disk.
    - You don't need to preserve the guest OS memory, device state, or application state.
    - You want to change certain properties of the VM that require you to first stop the VM.
- You can restart a terminated VM when you need to use it again.

10.1.1.2.1.2. Suspending a VM

- Suspending a VM sends an ACPI S3 suspend signal to the VM's operating system. Suspending a VM is analogous to closing the lid of your laptop or putting the VM into a suspended state. When you suspend a VM, it transitions to the SUSPENDED state.
- You might want to suspend a VM for the following reasons:
    - You don't need the VM at this time but want to be able to bring it back up quickly with its OS and application state already initialized.
    - You don't mind paying for Google to preserve the state of your VM.
- You can resume a suspended VM when you need to use it again.

10.1.1.2.1.3. Resetting a VM

- Alternatively, you can reset a VM to wipe the memory contents of the VM and reset the VM to its initial state. Resetting a VM causes an immediate hard reset of the VM; the VM doesn't do a graceful shutdown of the guest OS. However, the VM retains all persistent disk data, and none of the VM properties change. The VM remains in the RUNNING state through the reset.

10.1.1.2.1.4. Comparison table

The following table shows how stopping, suspending, and resetting a VM affects the VM and its resources:

	Stopping	Suspending	Resetting
Billing	Google cloud does not charge for VM resources while they are in the TERMINATED state.	Google charges for storing the preserved state of suspended VMs.	Because the VM remains in the RUNNING state when it is resetting, Google charges for VMs according to standard VM prices.
VM State	TERMINATED	SUSPENDED	RUNNING
Persistent disks	Persistent disks are maintained, even when persistent disks are marked for auto-delete. You are charged for a persistent disk that is associated with a stopped VM, just like you are charged for a persistent disk that is not associated with a VM.	Persistent disks are maintained, even when persistent disks are marked for auto-delete. You are charged for a persistent disk that is associated with a stopped VM, just like you are charged for a persistent disk that is not associated with a VM.	Persistent disks are maintained, even when persistent disks are marked for auto-delete. You are charged for persistent disks as long as the VM remains in a RUNNING state.
RAM and VM state	Reset to power-on state, no data is saved.	Guest OS and application state is preserved and restored when the VM is resumed. You are charged a specific rate for storing preserved data. For more information, see Billing for suspended instances.	Reset to power-on state, no data is saved.
GPUs	GPUs are not charged when a VM is stopped.	Not supported.	You are charged standard GPU prices.

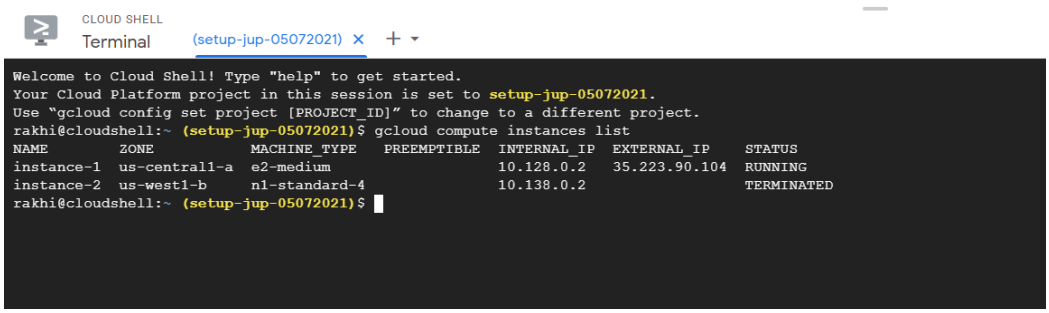
	A GPU that is attached to a TERMINATED VM doesn't count against your project quota.	If a GPU is attached to a VM, you cannot suspend the VM.	
Local SSDs	Not supported. You cannot stop a VM that has a local SSD attached to it.	Not supported. You can suspend a VM that has a local SSD, but you must explicitly discard the local SSD data in the process.	Local SSDs are maintained.
External ephemeral IPs	Ephemeral IPs are released when a VM is stopped, but a new ephemeral IP address is acquired when the VM restarts.	Ephemeral IPs are released when a VM is suspended, but a new ephemeral IP address is acquired when the VM restarts.	Ephemeral IPs are released when a VM is reset, but a new ephemeral IP address is acquired when the VM restarts.
External static IPs	Static external IPs are maintained. Static IPs that are assigned to VMs in the TERMINATED state are charged as if they aren't attached to any VM.	Static external IPs are maintained. Static IPs that are assigned to VMs in the SUSPENDED state are charged as if they aren't attached to any VM.	Static external IPs are maintained.
Internal IPs or MAC addresses	Internal IPs and MAC addresses are maintained.	Internal IPs and MAC addresses are maintained.	Internal IPs are released when a VM is deleted. Mac addresses are generated based on the internal IP. If you want to reuse the MAC address, set the same internal IP address or use a reserved internal IP.
VM metadata	VM metadata is maintained.	VM metadata is maintained.	VM metadata is maintained.

10.1.1.2.1.5. Checking a VM's status

- Every VM reports its status as part of the VM's properties. Check a VM's status in the following circumstances:
    - You are experiencing issues with the VM.
    - You just started the VM and want to confirm that the VM is running.
    - You want to confirm that the VM has been suspended or stopped (that is, in the TERMINATED state).

- To list all VMs and their statuses, use the following command:

gcloud compute instances list



- To describe the status of a single instance, use this command:

gcloud compute instances describe example-instance

10.1.1.3. Machine Types

- A machine type is a set of virtualized hardware resources available to a virtual machine (VM) instance, including the system memory size, virtual CPU (vCPU) count, and persistent disk limits. In Compute Engine, machine types are grouped and curated by families for different workloads. Compute Engine offers general-purpose, memory-optimized, compute-optimized, and accelerated-optimized families.

- You must choose a machine type when you create an instance. You can select from a number of predefined machine types in each machine type family. If the predefined machine types do not meet your needs, you can create your own custom machine type.

10.1.1.3.1. Machine types

- Each machine type family includes different machine types. Each family is curated for specific workload types. The following primary machine types are offered on Compute Engine:

- General-purpose machine types offer the best price-performance ratio for a variety of workloads.
    - E2 machine types are cost-optimized VMs that offer up to 32 vCPUs with up to 128 GB of memory with a maximum of 8 GB per vCPU. E2 machines have a predefined CPU platform running either an Intel or the second generation AMD EPYC Rome processor. E2 VMs provide a variety of compute resources for the lowest price on Compute Engine, especially when paired with committed-use discounts.
    - N2 machine types offer up to 80 vCPUs, 8 GB of memory per vCPU, and are available on the Intel Cascade Lake CPU platforms.
    - N2D machine types offer up to 224 vCPUs, 8 GB of memory per vCPU, and are available on second generation AMD EPYC Rome platforms.
    - N1 machine types offer up to 96 vCPUs, 6.5 GB of memory per vCPU, and are available on Intel Sandy Bridge, Ivy Bridge, Haswell, Broadwell, and Skylake CPU platforms.
- Memory-optimized machine types are ideal for memory-intensive workloads because they offer more memory per core than other machine types, with up to 12 TB of memory.
- Compute-optimized machine types offer the highest performance per core on Compute Engine and are optimized for compute-intensive workloads. Compute-optimized machine types offer Intel Scalable Processors (Cascade Lake) and up to 3.8 GHz sustained all-core turbo.

- Accelerator-optimized machine types are ideal for massively parallelized CUDA compute workloads, such as machine learning (ML) and high performance computing (HPC).

- Shared-core machine types are available in the E2 and N1 families. These machine types timeshare a physical core. This can be a cost-effective method for running small, non-resource intensive applications.

    - E2: e2-micro, e2-small, and e2-medium shared-core machine types have 2 vCPUs available for short periods of bursting.
    - N1: f1-micro and g1-small shared-core machine types have up to 1 vCPU available for short periods of bursting.

10.1.1.3.2. Recommendation for Machine types

- The following table provides machine type recommendations for different workloads.

E2 **General Purpose**	**N2,N2D,N1** **General Purpose**	**M2,M1** **Memory-Optimized**	**C2 Compute-Optimized**	**A2** **Accelerator-Optimized**
Day-to-day computing at a lower cost	Balanced price/performance across a wide range of VM shapes	Ultra high-memory workloads	Ultra high performance for compute-intensive workloads	Optimized for high performance computing workloads
• Web Serving • App Serving • Bank office applications • Small-medium databases • Microservices • Virtual desktops • Development environments	Web servingApp servingBank office applicationsMedium-large databasesCache Media/Streaming	Large in-memory databases like SAP HANA in-memory analytics	HPCElectronic Design Automation (EDA) Gaming Single-threaded applications	High performance computing (HPC) Machine learning (ML) Massive parallelized computation

10.1.1.4. Creating VM instances

- Here, we'll see how to create a virtual machine (VM) instance by using a boot disk image, a boot disk snapshot, or a container image. Some images support Shielded VM features, which offer security features such as UEFI-compliant firmware, Secure Boot, and vTPM-protected Measured Boot. On Shielded VMs, vTPM and integrity monitoring are enabled by default.

- While creating your VM, you can create one or more disks for it. You can also add more disks to the VM after it's created. Compute Engine automatically starts the VM instance after you create it.

Before you begin

- If you want to use the command-line examples in this guide, do the following:
    - Install or update to the latest version of the gcloud command-line tool.
    - Set a default region and zone.
- If you want to use the API examples in this guide, set up API access.
- When creating VMs from images or disks by using the gcloud command-line tool or the Compute Engine API, there's a limit of 20 VM instances per second. If you need to create a higher number of VMs per second, request a higher quota limit for the Images resource.

10.1.1.4.1. Create a VM instance from an image

- This section explains how to create a VM from a public OS image or a custom image. A VM contains a bootloader, a boot file system, and an OS image.

10.1.1.4.1.1. View a list of public images available on Compute Engine

- Before you create a VM by using a public image, review the list of public images that are available on Compute Engine.

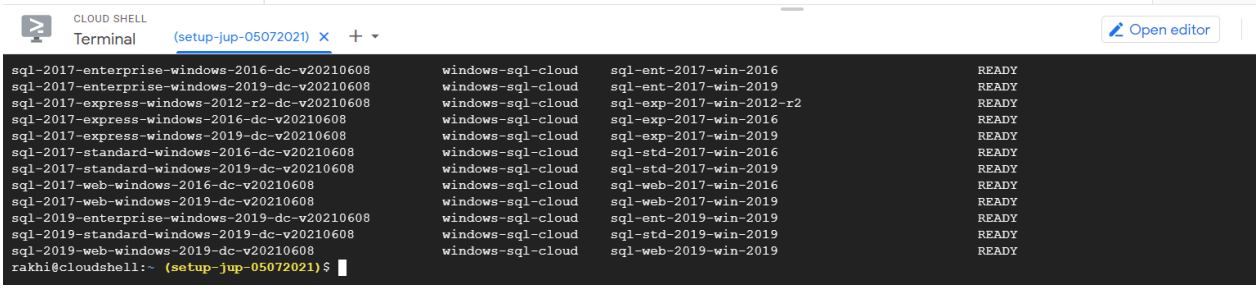
Using Console:

- In the Google Cloud Console, go to the Images page.

Using gcloud:

- Run the following command:

  gcloud compute images list



- Make a note of the name of the image or image family and the name of the project containing the image.

- Optional: To determine whether the image supports Shielded VM features, run the following command:

  *gcloud compute images describe IMAGE_NAME *
  --project=IMAGE_PROJECT

  Replace the following:

  - IMAGE_NAME: name of the image to check for support of Shielded VM features
  - IMAGE_PROJECT: project containing the image

If the image supports Shielded VM features, the following line appears in the output: type: UEFI_COMPATIBLE.

Using API:

- Run the following command:

  GET https://compute.googleapis.com/compute/v1/projects/IMAGE_PROJECT/global/images/

- Make a note of the name of the image or image family and the name of the project containing the image.

- Optional: To determine whether the image supports Shielded VM features, run the following command:

  GET https://compute.googleapis.com/compute/v1/projects/IMAGE_PROJECT/global/images/IMAGE_NAME

  Replace the following:

  - IMAGE_PROJECT: project containing the image
  - IMAGE_NAME: name of the image to check for support of Shielded VM features

  If the image supports Shielded VM features, the following line appears in the output: type: UEFI_COMPATIBLE.

10.1.1.4.1.2. Create a VM instance from a public image

- Google, open source communities, and third-party vendors provide and maintain public OS images. By default, all Google Cloud projects can create VMs from public OS images. However, if your Cloud project has a defined list of trusted images, you can use only the images on that list to create a VM.

- If you create a Shielded VM image with a local SSD, you can't shield data with integrity monitoring or the virtual platform trusted module (vTPM).

Permissions required for this task

- To perform this task, you must have the following permissions:
  - **compute.instances.create** on the project
  - **compute.instances.updateShieldedVmConfig** if you plan to create a Shielded VM instance and you want to be able to change any of the Shielded VM settings
  - **compute.networks.use** on the project if using a legacy network
  - **compute.subnetworks.use** either on the whole project or on the chosen subnet (VPC networks)
  - **compute.networks.useExternalIp** on the project if you need to assign an external IP address (either ephemeral or static) to the instance using a legacy network
  - **compute.subnetworks.useExternalIp** either on the whole project or on the chosen subnet if you need to assign an external IP address (either ephemeral or static) to the instance using a VPC network
  - **compute.addresses.use** on the project if specifying a static address in the project
  - **compute.instances.setMetadata** if setting metadata
  - **compute.instances.setTags** on the instance if setting tags
  - **compute.instances.setLabels** on the instance if setting labels
  - **compute.images.useReadOnly** on the image if creating a new root persistent disk
  - **compute.disks.create** on the project if creating a new root persistent disk with this instance
  - **compute.disks.useReadOnly** on the disk if attaching an existing persistent disk in read-only mode

- **compute.disks.use** on the disk if attaching an existing disk in read/write mode
- **compute.disks.setLabels** on the disk if setting labels
- **compute.snapshots.create** on the project to create a new snapshot if creating an instance from a snapshot
- **compute.snapshots.useReadOnly** on the snapshot if creating an instance from a snapshot

Using Console:

- In the Google Cloud Console, go to the **VM instances** page.
- Select your project and click **Continue**.
- Click **Create instance**.
- Specify a **Name** for your VM.
- Optional: Change the **Zone** for this VM. Compute Engine randomizes the list of zones within each region to encourage use across multiple zones.
- Select a **Machine configuration** for your VM.
- In the **Boot disk** section, click **Change** to configure your boot disk. Unless you explicitly choose a different boot disk, if the name of the new VM matches the name of an existing persistent disk, then the existing persistent disk automatically attaches to the new VM as the boot disk.
- In the **Public images** tab, choose the following:
  - Operating system
  - OS version
  - Boot disk type

- Boot disk size

- Click Save to confirm your boot disk options.

- Select Allow HTTP traffic or Allow HTTPS traffic to permit HTTP or HTTPS traffic to the VM. When you select one of these, Compute Engine adds a network tag to your VM, which associates the firewall rule with the VM. Then, Compute Engine creates the corresponding ingress firewall rule that allows all incoming traffic on tcp:80 (HTTP) or tcp:443 (HTTPS).

- Optional: If you chose an OS image that supports Shielded VM features, you can modify the Shielded VM settings. To modify shielded VM settings, click the Security tab in the Management, security, disks, networking, sole tenancy section and do the following, as required:

  - To enable Secure Boot, select Turn on Secure Boot. Secure Boot is disabled by default.

  - To disable vTPM, clear Turn on vTPM. vTPM is enabled by default. Disabling vTPM also disables integrity monitoring because integrity monitoring relies on data gathered by Measured Boot.

  - To disable integrity monitoring, clear the Turn on Integrity Monitoring checkbox. Integrity monitoring is enabled by default.

- Click Create to create and start the VM.

Using gcloud:

- Select a public image. Make a note of the name of the image or image family and the name of the project containing the image.

- Use the *gcloud compute instances create* command to create a VM from an image family or from a specific version of an OS image.

- If you specify the optional *--shielded-secure-boot flag*, Compute Engine creates a VM with all three of the Shielded VM features enabled:

  - Virtual trusted platform module (vTPM)

  - Integrity monitoring

- Secure Boot

- After Compute Engine starts your VM, you must stop the VM to modify Shielded VM options.

```
gcloud compute instances create VM_NAME \
  [--image=IMAGE | --image-family=IMAGE_FAMILY] \
  --image-project=IMAGE_PROJECT
  --machine-type=MACHINE_TYPE
```

  - Replace the following:

    - VM_NAME: name of the new VM
    - IMAGE or IMAGE_FAMILY: specify one of the following:

    - IMAGE: a specific version of a public image

    - For example, *--image=debian-10-buster-v20200309*.

    - IMAGE_FAMILY: an image family.

    - This creates the VM from the most recent, non-deprecated OS image. For example, if you specify *--image-family=debian-10*, Compute Engine creates a VM from the latest version of the OS image in the Debian 10 image family.

    - IMAGE_PROJECT: project containing the image

    - MACHINE_TYPE: machine type, predefined or custom, for the new VM

    - To get a list of the machine types available in a zone, use the *gcloud compute machine-types list* command with the --zones flag.

- Verify that Compute Engine created the VM:

  gcloud compute instances describe VM_NAME

  Replace VM_NAME with the name of the VM.

Using API:

- Select a public image. Make a note of the name of the image or image family and the name of the project containing the image.

- Use the instances.insert method to create a VM from an image family or from a specific version of an OS image:

```
POST https://compute.googleapis.com/compute/v1/projects/PROJECT_ID/zones/ZONE/instances

{
 "machineType": "zones/MACHINE_TYPE_ZONE/machineTypes/MACHINE_TYPE",
 "name": "VM_NAME",
 "disks": [
  {
    "initializeParams": {
      "sourceImage": "projects/IMAGE_PROJECT/global/images/IMAGE"
    },
    "boot": true
  }
 ],
 "shieldedInstanceConfig": {
   "enableSecureBoot": ENABLE_SECURE_BOOT
 }
}
```

- Replace the following:
    - PROJECT_ID: ID of the project to create the VM in
    - ZONE: zone to create the VM in
    - MACHINE_TYPE_ZONE: zone containing the machine type to use for the new VM
    - MACHINE_TYPE: machine type, predefined or custom, for the new VM
    - VM_NAME: name of the new VM
    - IMAGE_PROJECT: project containing the image

- For example, if you specify debian-10 as the image family, specify debian-cloud as the image project.

- IMAGE or IMAGE_FAMILY: specify one of the following:

  - IMAGE: a specific version of a public image

  - For example, "sourceImage": "projects/debian-cloud/global/images/debian-10-buster-v20200309"

  - IMAGE_FAMILY: an image family

  - This creates the VM from the most recent, non-deprecated OS image. For example, if you specify "sourceImage": "projects/debian-cloud/global/images/family/debian-10", Compute Engine creates a VM from the latest version of the OS image in the Debian 10 image family.

- ENABLE_SECURE_BOOT: Optional: If you chose an image that supports Shielded VM features, Compute Engine, by default, enables the virtual trusted platform module (vTPM) and integrity monitoring. Compute Engine does not enable Secure Boot by default.

- If you specify true for enableSecureBoot, Compute Engine creates a VM with all three Shielded VM features enabled. After Compute Engine starts your VM, to modify Shielded VM options, you must stop the VM.

Using Python:

```
def create_instance(compute, project, zone, name, bucket):
  # Get the latest Debian Jessie image.
  image_response = compute.images().getFromFamily(
    project='debian-cloud', family='debian-9').execute()
  source_disk_image = image_response['selfLink']

  # Configure the machine
  machine_type = "zones/%s/machineTypes/n1-standard-1" % zone
  startup_script = open(
    os.path.join(
      os.path.dirname(__file__), 'startup-script.sh'), 'r').read()
```

```
image_url = "http://storage.googleapis.com/gce-demo-input/photo.jpg"
image_caption = "Ready for dessert?"

config = {
    'name': name,
    'machineType': machine_type,

    # Specify the boot disk and the image to use as a source.
    'disks': [
        {
            'boot': True,
            'autoDelete': True,
            'initializeParams': {
                'sourceImage': source_disk_image,
            }
        }
    ],

    # Specify a network interface with NAT to access the public
    # internet.
    'networkInterfaces': [{
        'network': 'global/networks/default',
        'accessConfigs': [
            {'type': 'ONE_TO_ONE_NAT', 'name': 'External NAT'}
        ]
    }],

    # Allow the instance to access cloud storage and logging.
    'serviceAccounts': [{
        'email': 'default',
        'scopes': [
            'https://www.googleapis.com/auth/devstorage.read_write',
            'https://www.googleapis.com/auth/logging.write'
        ]
    }],

    # Metadata is readable from the instance and allows you to
    # pass configuration from deployment scripts to instances.
    'metadata': {
        'items': [{
            # Startup script is automatically executed by the
```

```
            # instance upon startup.
            'key': 'startup-script',
            'value': startup_script
        }, {
            'key': 'url',
            'value': image_url
        }, {
            'key': 'text',
            'value': image_caption
        }, {
            'key': 'bucket',
            'value': bucket
        }]
    }
}

return compute.instances().insert(
    project=project,
    zone=zone,
    body=config).execute()
```

10.1.1.4.1.3. Create a VM from a custom image

- A custom image belongs only to your project. To create a VM with a custom image, you must first **create a custom image** if you don't already have one.

Create a custom image

- This section describes how to create a custom image on a Linux VM.

Select an image storage location

- When creating a custom image, you can specify the image's Cloud Storage location, excluding dual-region locations. By specifying the image storage location, you can meet your regulatory and compliance requirements for data locality as well as your high availability needs by ensuring redundancy across regions. To create, modify, and delete images stored in Cloud Storage, you must have roles/compute.storageAdmin.

- The storage location feature is optional. If you do not select a location, Compute Engine stores your image in the multi-region closest to the image source.

- Note: Custom images are not stored in other regions until you create a VM in that region. If the image is not available when you create a VM in another region, Compute Engine caches the image in that region.

- To see the location where an image is stored, use the images describe command from gcloud compute:

  *gcloud compute images describe IMAGE_NAME *
  --project=PROJECT_ID

  - Replace the following:

    - IMAGE_NAME: the name of your image.

    - PROJECT_ID: the project ID to which the image belongs.

  - All of your existing images prior to this feature launch remain where they are, the only change is that you can view the image location of all your images. If you have an existing image you want to move, you must recreate it in the desired location.

Prepare your VM for an image

- You can create an image from a disk even while it is attached to a running VM. However, your image is more reliable if you put the VM in a state that is easier for the image to capture. Use one of the following processes to prepare your boot disk for the image:

    - Stop the VM so that it can shut down and stop writing any data to the persistent disk.

    - If you can't stop your VM before you create the image, minimize the amount of writes to the disk and sync your file system. To minimize writing to your persistent disk, follow these steps:

        - Pause apps or operating system processes that write data to that persistent disk.

- Run an app flush to disk if necessary. For example, MySQL has a FLUSH statement. Other apps might have similar processes.
- Stop your apps from writing to your persistent disk.
- Run sudo sync.

- After you prepare the VM, create the image.

Create the image

- You can create disk images from the following sources:
  - A persistent disk, even while that disk is attached to a VM
  - A snapshot of a persistent disk
  - Another image in your project
  - An image that is shared from another project
  - A compressed RAW image in Cloud Storage
- You can create a disk image once every 10 minutes. If you want to issue a burst of requests to create a disk image, you can issue at most 6 requests in 60 minutes.

Using Console

- In the Google Cloud Console, go to the **Create an image** page.
- Specify the **Name** of your image.
- Specify the **Source** from which you want to create an image. This can be a persistent disk, a snapshot, another image, or a disk.raw file in Cloud Storage.
- If you are creating an image from a disk attached to a running VM, check **Keep instance running** to confirm that you want to create the image while the VM is running. You can prepare your VM before creating the image.

- In the **Based on source disk location (default)** drop-down list, specify the location to store the image. For example, specify us to store the image in the us multi-region, or us-central1 to store it in the us-central1 region. If you don't make a selection, Compute Engine stores the image in the multi-region closest to your image's source location.

- Optional: specify the properties for your image.

    - **Family**: the image family this new image belongs to.

    - **Description**: a description for your custom image.

    - **Label**: a label to group together resources.

- Specify the encryption key. You can choose between a Google-managed key, a Cloud Key Management Service (Cloud KMS) key or a customer- supplied encryption (CSEK) key. If no encryption key is specified, images are encrypted using a Google-managed key.

- Click **Create** to create the image.

Permissions required to create a VM from a custom image

- To perform this task, you must have the following permissions:

    - compute.instances.create on the project

    - compute.instances.updateShieldedVmConfig if you plan to create a Shielded VM instance and you want to be able to change any of the Shielded VM settings

    - compute.networks.use on the project if using a legacy network

    - compute.subnetworks.use either on the whole project or on the chosen subnet (VPC networks)

    - compute.networks.useExternalIp on the project if you need to assign an external IP address (either ephemeral or static) to the instance using a legacy network

    - compute.subnetworks.useExternalIp either on the whole project or on the chosen subnet if you need to assign an external IP address (either ephemeral or static) to the instance using a VPC network

- compute.addresses.use on the project if specifying a static address in the project
- compute.instances.setMetadata if setting metadata
- compute.instances.setTags on the instance if setting tags
- compute.instances.setLabels on the instance if setting labels
- compute.images.useReadOnly on the image if creating a new root persistent disk
- compute.disks.create on the project if creating a new root persistent disk with this instance
- compute.disks.useReadOnly on the disk if attaching an existing persistent disk in read-only mode
- compute.disks.use on the disk if attaching an existing disk in read/write mode
- compute.disks.setLabels on the disk if setting labels
- compute.snapshots.create on the project to create a new snapshot if creating an instance from a snapshot
- compute.snapshots.useReadOnly on the snapshot if creating an instance from a snapshot

Using Console

- Go to the VM instances page.
- Select your project and click Continue.
- Click Create instance.
- Specify a Name for your VM. See Resource naming convention.
- Optional: Change the Zone for this VM. Compute Engine randomizes the list of zones within each region to encourage use across multiple zones.

- Select a Machine configuration for your VM.
- In the Boot disk section, click Change to configure your boot disk. Then, do the following:
    - Select the Custom Images tab.
    - Select your project from the Show images from drop-down list.
    - Select the image you want from the Images drop-down list.
    - Select a boot disk type.
    - Specify the size.
    - Click Select.
- To permit HTTP or HTTPS traffic to the VM, select Allow HTTP traffic or Allow HTTPS traffic.
- The Cloud Console adds a network tag to your VM and creates the corresponding ingress firewall rule that allows all incoming traffic on tcp:80 (HTTP) or tcp:443 (HTTPS). The network tag associates the firewall rule with the VM. For more information, see Firewall rules overview in the Virtual Private Cloud documentation.
- Click Create to create and start the VM.

Using gcloud

- Run the *gcloud compute instances create* command to create a VM with a custom image:

```
gcloud compute instances create VM_NAME \
--image-project IMAGE_PROJECT \
[--image IMAGE | --image-family IMAGE_FAMILY]
```

    - Replace the following:
        - **VM_NAME**: name of the VM
        - **IMAGE_PROJECT**: name of the project that contains the image

- **IMAGE or IMAGE_FAMILY**: specify one of the following:
  - **IMAGE**: name of your custom image
  - For example, *--image=my-debian-image-v2*.
  - **IMAGE_FAMILY**: if you created your custom images as part of a custom image family, specify that custom image family.
  -
  - This creates the VM from the most recent, non-deprecated OS image and OS version in your custom image family. For example, if you specify *--image-family=my-debian-family*, Compute Engine creates a VM from the latest OS image in your custom *my-debian-family* image family.
  - Note: Compute Engine uses the default image family and project if you don't specify an image. The default image family and project are debian-10 and debian-cloud, respectively.

Using API

- The process for creating a VM with a custom image in the API is the same as if you were creating a VM with a publicly available image.
- To create the VM from a custom image, use the instances.insert method.

```
POST https://compute.googleapis.com/compute/v1/projects/PROJECT_ID/zones/ZONE/instances

{
 "machineType": "zones/MACHINE_TYPE_ZONE/machineTypes/MACHINE_TYPE",
 "name": "VM_NAME",
 "disks": [
  {
    "initializeParams": {
      "sourceImage": "projects/IMAGE_PROJECT/global/images/IMAGE"
    },
    "boot": true
  }
```

```
],
.....
}
```

- Replace the following:
  - **PROJECT_ID**: ID of the project to create the VM in
  - **ZONE**: zone to create the VM in
  - **MACHINE_TYPE_ZONE**: zone containing the machine type to use for the new VM
  - **MACHINE_TYPE**: machine type, predefined or custom, for the new VM
  - **VM_NAME**: name of the new VM
  - **IMAGE_PROJECT**: name of the project that contains the custom image
  - **IMAGE or IMAGE_FAMILY**: specify one of the following:
    - **IMAGE**: name of your custom image
    - For example, *"sourceImage": "projects/my-project-1234/global/images/my-debian-image-v2"*.
    - **IMAGE_FAMILY**: if you created your custom images as part of a custom image family, specify that custom image family.
    - This creates the VM from the most recent, non-deprecated OS image in your custom image family. For example, if you specify *"sourceImage": "projects/my-project-1234/global/images/family/my-debian-family"*, Compute Engine creates a VM from the latest version of the OS image in the custom *my-debian-family* image family.

11. References:

https://cloud.google.com/docs/

https://medium.com/google-cloud/gcp-dashboard-overview-c80ffd0ed521

https://www.javatpoint.com/advantages-and-disadvantages-of-cloud-computing

https://cloud.google.com/why-google-cloud

https://en.wikipedia.org/wiki/Google_Cloud_Platform