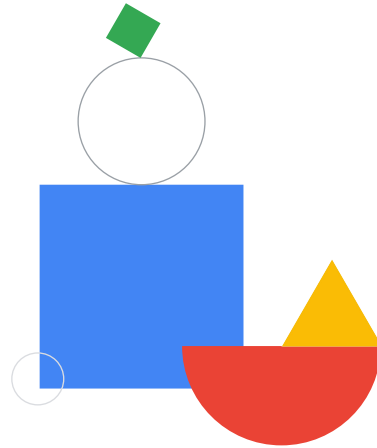



Big Data Analytics with Notebooks



In this module, we are going to introduce Notebooks, an extremely useful tool for prototyping machine learning solutions.



Module agenda



- 01 What's a Notebook?
 - 02 BigQuery Magic and Ties to Pandas
-

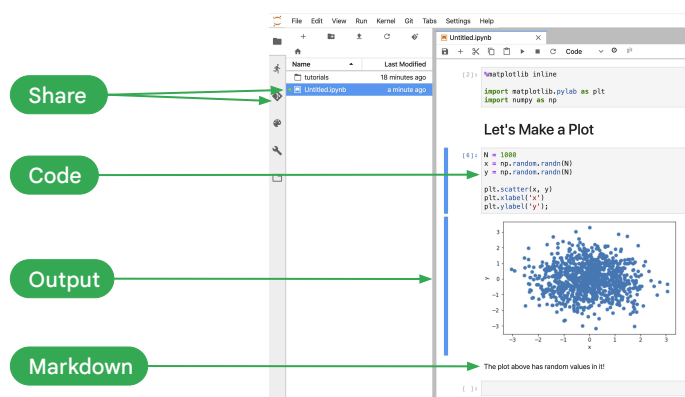
We will start by describing in detail some of the benefits of Notebooks. We will then discuss how notebooks can be tied into other Google Cloud services.



What's a Notebook?

Let's start by explaining what Notebooks are and some of their benefits.

Increasingly, data analysis and machine learning are carried out in self-descriptive, shareable, executable notebooks



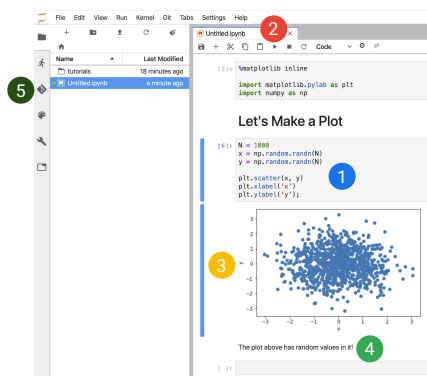
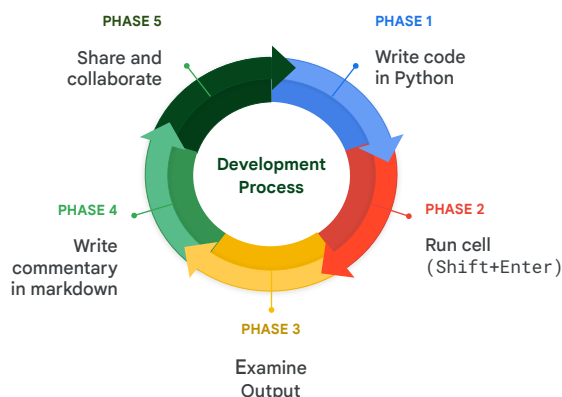
A typical notebook contains code, charts, and explanations

Standard software development tools are not very efficient for writing code for data analysis and machine learning.

Data analysis and machine learning coding often involve looking at plots, repeatedly executing small chunks of code with minor changes, and frequently having to print output. Iteratively, running whole scripts for this task is burdensome.

These were some of the issues that motivated the development of notebooks. Notebook environments seamlessly integrate commentary, plots, and code. Rather than having a script that performs a piece of analysis, notebooks organize individually executable pieces of code into cells.

Notebooks are developed in an iterative, collaborative process



Have you used Google Docs? How is it different from documents edited in a desktop editor?

Have you filed taxes online? How is the experience different from doing your taxes in a desktop program?

There are lots of benefits, but one key aspect is collaboration. You don't have to email documents back and forth.

When I first started doing scientific research, collaborating on a single result was painful. I'd write some code and create a graph. Then, I would snapshot, create an image file, put into a doc, create a PDF, and send it to my collaborator. A few hours later, my colleague would say, "Looks great, but could you add one more year's data? It looks kinda sparse." And I'd go through the process all over again. Why? Because the PDF I'd sent was not editable. Round-trips took a lot of time.

Enter Python notebooks. I'd write the code, create the graph, write some commentary, and send the notebook link to my colleague. This way, when my colleague wanted to add one more year of data, they could simply edit the cell, look at the new graph, and say, "See, it looks a lot better." And that was great; we now had a better notebook for the next step.

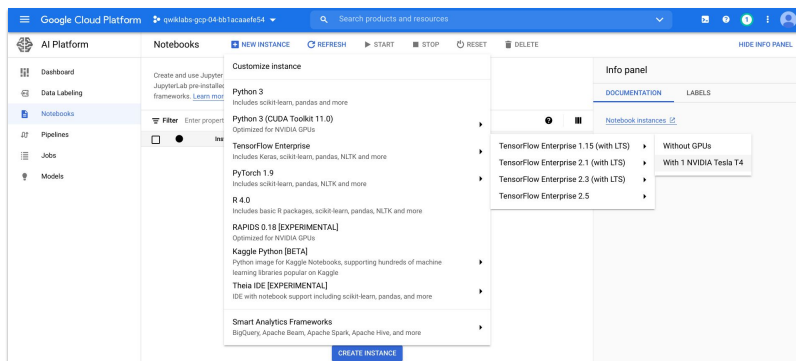
One problem with traditional notebooks: Who runs the server that hosts these pages? Who owns the machine? If it's mine, and my machine goes to sleep, then my

colleague can't work!

When your Notebooks are hosted in the cloud, you can develop together quite easily. And just as Google Docs are available even when your computer isn't on, so too are Notebooks, when you run them in the Cloud.

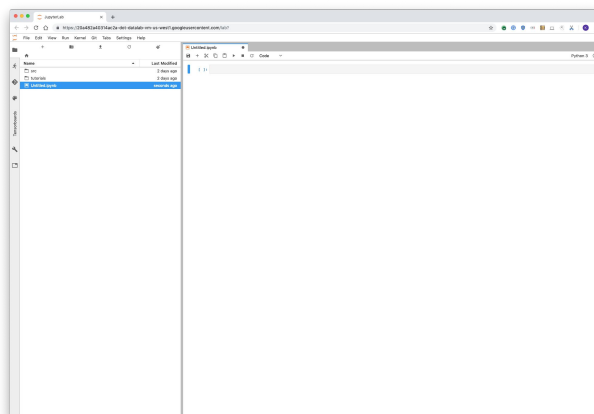
To share a notebook within a project, other users can simply "connect" to the VM and work using the URL. Another way to share notebooks is through revision control systems, such as Git.

Spin up a JupyterLab instance, pre-configured with the latest machine learning and data science frameworks in one click



Spinning up a notebook can be done in one click. You may have heard of Jupyter Notebooks. Jupyter Notebooks are basically synonymous to Notebooks on Google Cloud (Vertex AI and AI Platform). The provided Python environment has a standard machine learning library pre-installed.

Notebooks use the latest open-source version of the industry-standard JupyterLab



Notebooks use the latest open source version of the industry standard JupyterLab.

Use any Compute Engine instance type

← Editing Notebook instance details

OPEN JUPYTERLAB

CANCEL

VIEW VM DETAILS

▶ START

■ STOP

↺ RESET

⌄ UPGRADE

🗑 DELETE

● tensorflow-1-15-20210701-152344

BASIC INFO

Region	us-west1 (Oregon)
Zone	us-west1-b
Environment	TensorFlow Enterprise 1.15 (with LTS and Intel® MKL-DNN/MKL)
Environment version	M74
Boot disk	100 GB disk
Data disk	100 GB disk
Backup	Not specified
Subnetwork	default
Service account	673971842877-compute@developer.gserviceaccount.com
Permission mode	Service account
Sudo access	Enabled
File downloads	Enabled
ibcowner	Enabled
Shielded VM	Secure Boot not enabled vTPM enabled Integrity Monitoring enabled

Machine configuration

Machine type *
n1-standard-4 (4 vCPUs, 15 GB RAM)

GPUs
GPU type
None

Based on the zone, environment, and machine type selected above, the available GPU types and the minimum number of GPUs that can be selected may vary. [Learn more](#)

System

Environment auto-upgrade
If a new environment version is available, an upgrade will be conducted for your running instance at the time specified. It will be skipped if instances are stopped. **All the data in your data disk will be kept. Backward compatibility is not guaranteed.** [Learn more](#)

☐ Enable environment auto-upgrade

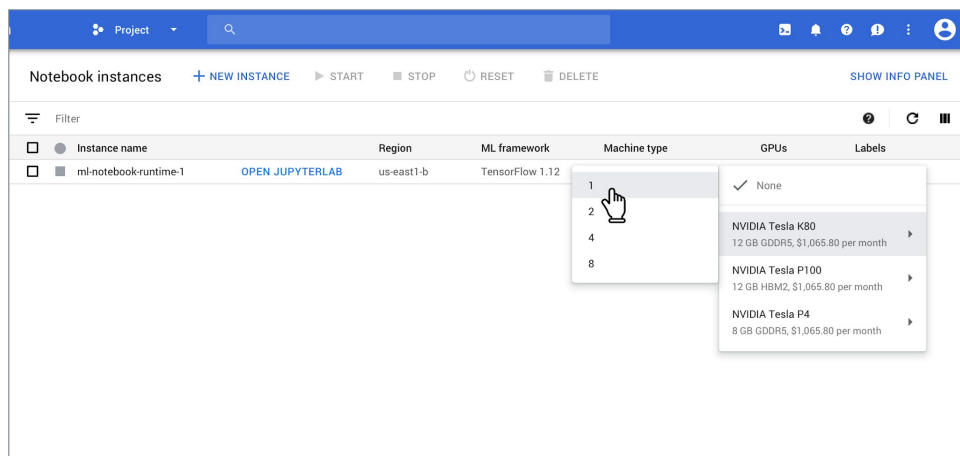
Instance upgrade history
Includes all the environment upgrades conducted for this instance. If needed, you could rollback the latest upgrade.

Date	Description
No upgrades to display	

SUBMIT CANCEL

When you initially set your Notebooks hardware, you can use any Compute Engine instance type.

You can easily change hardware



You can also easily adjust the underlying hardware, including machine types.

You can even add and remove GPUs

Notebooks [NEW INSTANCE](#) [REFRESH](#) [START](#) [STOP](#) [RESET](#) [DELETE](#) [SHOW INFO PANEL](#)

Create and use Jupyter Notebooks with a notebook instance. Notebook instances have JupyterLab pre-installed and are configured with GPU-enabled machine learning frameworks. [Learn more](#)

Filter Enter property name or value

	Instance name ↑	Zone	Environment Version	Auto-upgrade	Environment	Machine type	GPUs	Permission
<input type="checkbox"/>	python-20210701-151535 OPEN JUPYTERLAB	us-west1-b	M73	—	NumPy/SciPy/scikit-learn	4 vCPUs, 15 GB RAM	None	Service account
<input checked="" type="checkbox"/>	tensorflow-1-15-20210701-152344 OPEN JUPYTERLAB	us-west1-b	M74	—	TensorFlow:1.15	4 vCPUs, 15 GB RAM	None	Service account

☒ None

NVIDIA Tesla K80
\$328.50 per month

NVIDIA Tesla P100
\$1,065.80 per month

NVIDIA Tesla T4
\$255.50 per month

NVIDIA Tesla V100
\$1,810.40 per month

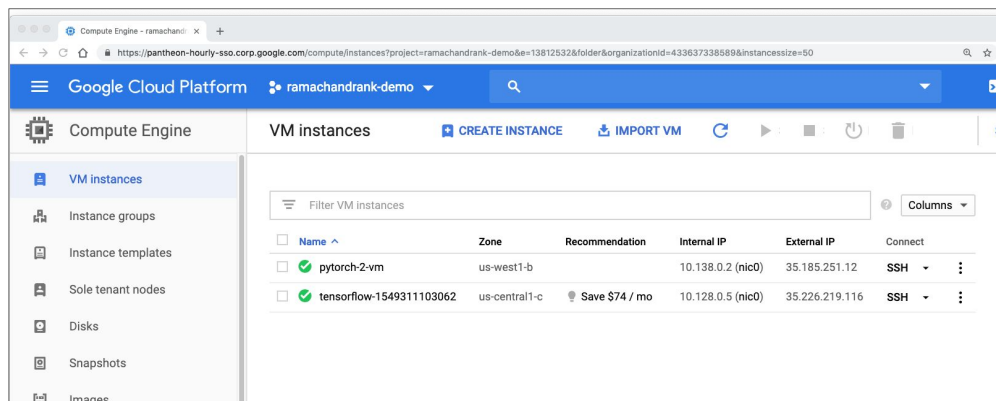
Google Cloud

Compute Engine provides Graphics Processing Units, or GPUs, that you can add to your virtual machine instances. You can use these GPUs to accelerate specific workloads on your VMs, such as machine learning and data processing.

If you did not attach GPUs during VM creation, you can add GPUs to your existing VMs to suit your application needs as they arise.

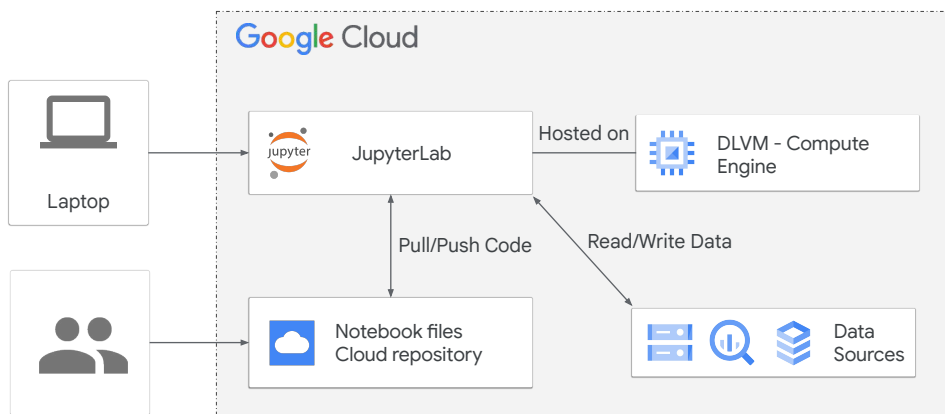
If you attach GPUs during or after VM creation, you can detach these GPUs from these VMs when you no longer need them.

Notebook instances are standard Compute Engine instances that live in your projects



Note that Notebook instances are treated in the same way as standard Compute Engine instances living in your projects. If you start a Notebook, log in to the Cloud Console and navigate to VM instances. You will see your Notebook instances under Compute Engine.

How does it work?



As mentioned before, Notebooks run the latest version of JupyterLab, which in this case is hosted on Compute Engine. That accounts for the hardware and libraries. For collaboration, the Notebooks are integrated with Git by default, so you can version your Notebooks.

Notebooks also provide connectors to and from BigQuery so that you can easily pull data into your Notebook.



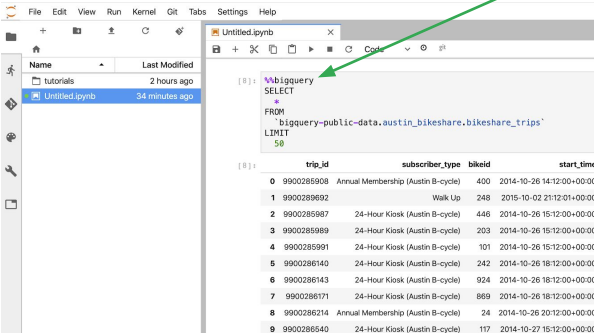
BigQuery Magic and Ties to Pandas

Let's describe a convenient functionality within Notebooks that allows you to pull data from BigQuery straight into your notebook.

You can execute BigQuery commands from Notebooks

- Useful for checking query validity
- Viewing query output
- But... can't use query output for anything

Jupyter "magic" function



```
%%bigquery
SELECT
  *
FROM
  `bigquery-public-data.austin_bikeshare.bikeshare_trips`
LIMIT
  50
```

	trip_id	subscriber_type	bikeid	start_time
0	9900285908	Annual Membership (Austin B-cycle)	400	2014-10-26 14:12:00+00:00
1	9900285902	Walk Up	248	2015-10-02 21:12:01+00:00
2	9900285987	24-Hour Kiosk (Austin B-cycle)	448	2014-10-26 15:12:00+00:00
3	9900285989	24-Hour Kiosk (Austin B-cycle)	203	2014-10-26 15:12:00+00:00
4	9900285991	24-Hour Kiosk (Austin B-cycle)	101	2014-10-26 15:12:00+00:00
5	9900286140	24-Hour Kiosk (Austin B-cycle)	242	2014-10-26 18:12:00+00:00
6	9900286143	24-Hour Kiosk (Austin B-cycle)	924	2014-10-26 18:12:00+00:00
7	9900286171	24-Hour Kiosk (Austin B-cycle)	869	2014-10-26 18:12:00+00:00
8	9900286214	Annual Membership (Austin B-cycle)	24	2014-10-26 20:12:00+00:00
9	9900286540	24-Hour Kiosk (Austin B-cycle)	117	2014-10-27 15:12:00+00:00

You may be familiar with magic functions in Jupyter labs. Magic functions allow you to execute system commands from within notebook cells. There are magic functions to check the contents of your current directory. You can also define custom magic functions. The BigQuery magic function shown in this slide allows you to execute BigQuery queries. This is useful for checking query format, accuracy, and output.

Can use the BigQuery API in Notebooks to return query results as a Pandas DataFrame

```
[44]: %%bigquery df
      SELECT
      *
      FROM
      `bigquery-public-data.austin_bikeshare.bikeshare_trips`
      WHERE
      end_station_name = 'Stolen'

[46]: print(type(df))
      df.head()
```

<class 'pandas.core.frame.DataFrame'>

	trip_id	subscriber_type	bikeid	start_time	start_station_id	start_station_name	end_station_id	end_station_name	duration_minutes
0	9900259257	Walk Up	93	2015-09-18 08:12:05+00:00	2712	Toomey Rd @ South Lamar	None	Stolen	2863
1	16898448	Walk Up	1857	2018-03-18 22:51:20+00:00	2501	5th & Bowie	None	Stolen	3806
2	9900298869	Walk Up	127	2015-10-10 19:12:38+00:00	2574	Zilker Park	None	Stolen	3632
3	9900290440	Local365	277	2015-10-02 22:12:06+00:00	2494	2nd & Congress	None	Stolen	8
4	9900322570	Walk Up	439	2015-11-01 02:12:28+00:00	2496	8th & Congress	None	Stolen	6609

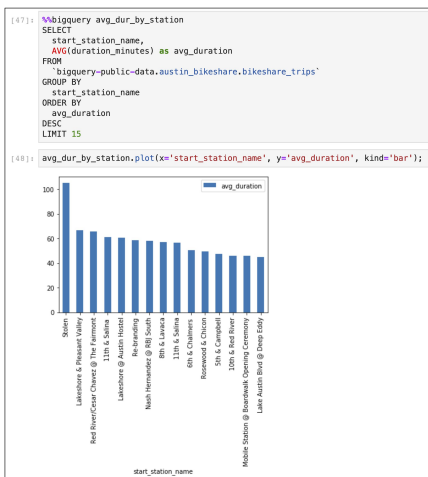
Pandas DataFrame

The BigQuery magic function allows you to save the query output to a Pandas DataFrame, so that you can manipulate it further.

In this example, we are saving the output of a query to a Pandas DataFrame named `df`. Pandas is a numeric library for Python, it displays data in a tabular structure. It also shows you metadata so that you can do things like describe data frames, object types, and generate summary statistics on the columns.

This is really powerful to be able to just pull query results directly into a DataFrame into two lines of code. Keep in mind you are working in a notebook and there will be a limited amount of memory. So you wouldn't want to go to BigQuery and pull up a big, multi-terabyte or larger dataset. Instead use a sampling technique to pull a subset of the data.

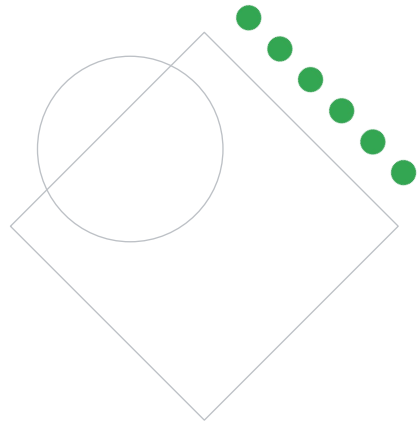
Pandas + BigQuery in Notebook rocks!



Here is an example of how you can execute a BigQuery query to pull data into two Pandas data frame objects. A simple bar plot is then generated from the data frame.

Lab Intro

BigQuery in JupyterLab
on Vertex AI



It's time for you to check out Notebooks for yourself. In this lab, you will first instantiate a Jupyter notebook on Vertex AI. Then, you will execute a BigQuery query from within the Jupyter notebook, and process the output using Pandas.