

DataCamp uses cookies to improve your learning experience and offer data science and analytics content relevant to your interests. We also use cookies to show advertisements you might be interested in.

[Learn more & configure](#)[Accept All](#)

Alexis Perrier
February 6th, 2018

DATA ANALYSIS

Google Cloud for Data Science: Beginner's Guide

Learn to set up a data science environment on Google Cloud: create an instance on Google Compute Engine, install Anaconda and run Jupyter notebooks!

While AWS EC2 is the leader in cloud computing, Google Cloud has developed a very compelling and competitive Cloud Computing platform.

In this tutorial, you will learn how to:

- Create an instance on Google Compute Engine (GCE),
- Install a data science environment based on the Anaconda Python distribution, and
- Run Jupyter notebooks accessible online.

Working in the cloud instead of on your own machine has two main advantages for data science projects:

1. **Scalability:** you can tailor the power (RAM, CPU, GPU) of your instance to your immediate needs. Starting with a small and cheap instance and adding memory, storage, CPUs or GPUs as your project evolves.
2. **Reproducibility:** a key condition of any data science project. Allowing other data

scientists to review your models and reproduce your research is a necessary condition of a successful implementation. By setting up a working environment on a virtual instance, you make sure your work can easily be shared, reproduced, and vetted by other team members.

Google Compute Engine

Although built around the same concepts and elements (instances, images and snapshots), EC2 and GCE differ on both access and resources organization.

A key aspect of the Google Cloud Platform (GCP) is its project-centered organization. All billing, permissions, resources and settings are grouped within a user-defined project which basically acts as a global namespace. This not only simplifies the interconnected mapping of the resources you use (storage, databases, instances, ...) but also access management from role-based permissions to actual ssh keys and security. When it comes to user friendliness and access and role management, I find that working on the GCP is easier than working with AWS especially using multiple services.

The GCP also offers certain services which are particularly relevant for data science, including but not limited to:

- [Dataprep](#) to build data processing pipelines,
- [Datalab](#) for data exploration,
- the [Google Machine Learning Engine](#) built on TensorFlow
- [BigQuery](#) a data warehouse solution that holds many fascinating Big Data datasets.

A low learning curve and data friendly services make the GCP a must have in your data scientist toolbox.

Before you start launching instances and installing Python packages, let's spend a few moments to review some of the common vocabulary used in Cloud Computing.

VMs, Disks, Images and Snapshots

A Virtual Machine (VM) also called "an instance" is an on-demand server that you activate as needed. The underlying hardware is shared among other users in a transparent way

and as such becomes entirely virtual to you. You only choose a global geographic location of the instance hosted in one of Google's data center.

A VM is defined by the type of persistent disk and the operating system (OS), such as Windows or Linux, it is built upon. The persistent disk is your virtual slice of hardware.

An image is the physical combination of a persistent disk and the operating system. VM Images are often used to share and implement a particular configuration on multiple other VMs. Public images are the ones provided by Google with a choice of specific OS while private images are customized by users.

A snapshot is a reflection of the content of a VM (disk, software, libraries, files) at a given time and is mostly used for instant backups. The main difference between snapshots and images is that snapshots are stored as diffs, relative to previous snapshots, while images are not.

An image and a snapshot can both be used to define and activate a new VM.

To recap, when you launch a new instance, GCE starts by attaching a persistent disk to your VM. This provides the disk space and gives the instance the root filesystem it requires to boot up. The disk installs the OS associated with the image you have chosen. By taking snapshots of an image, you create instant backups and you can copy data from existing VMs to launch new VMs.

Let's put all of that in practice and get started with your first VM!

Getting Started with Your First VM on GCP

Create an Account and Project

To open an account on the GCP, you need a standard Google (Gmail) account. At time of writing, Google offers a 12 months / \$300 [free trial](#) of the platform. Although this offer comes with certain [restrictions](#) (no bitcoin mining, for instance), it should be sufficient to get you started working in this environment.

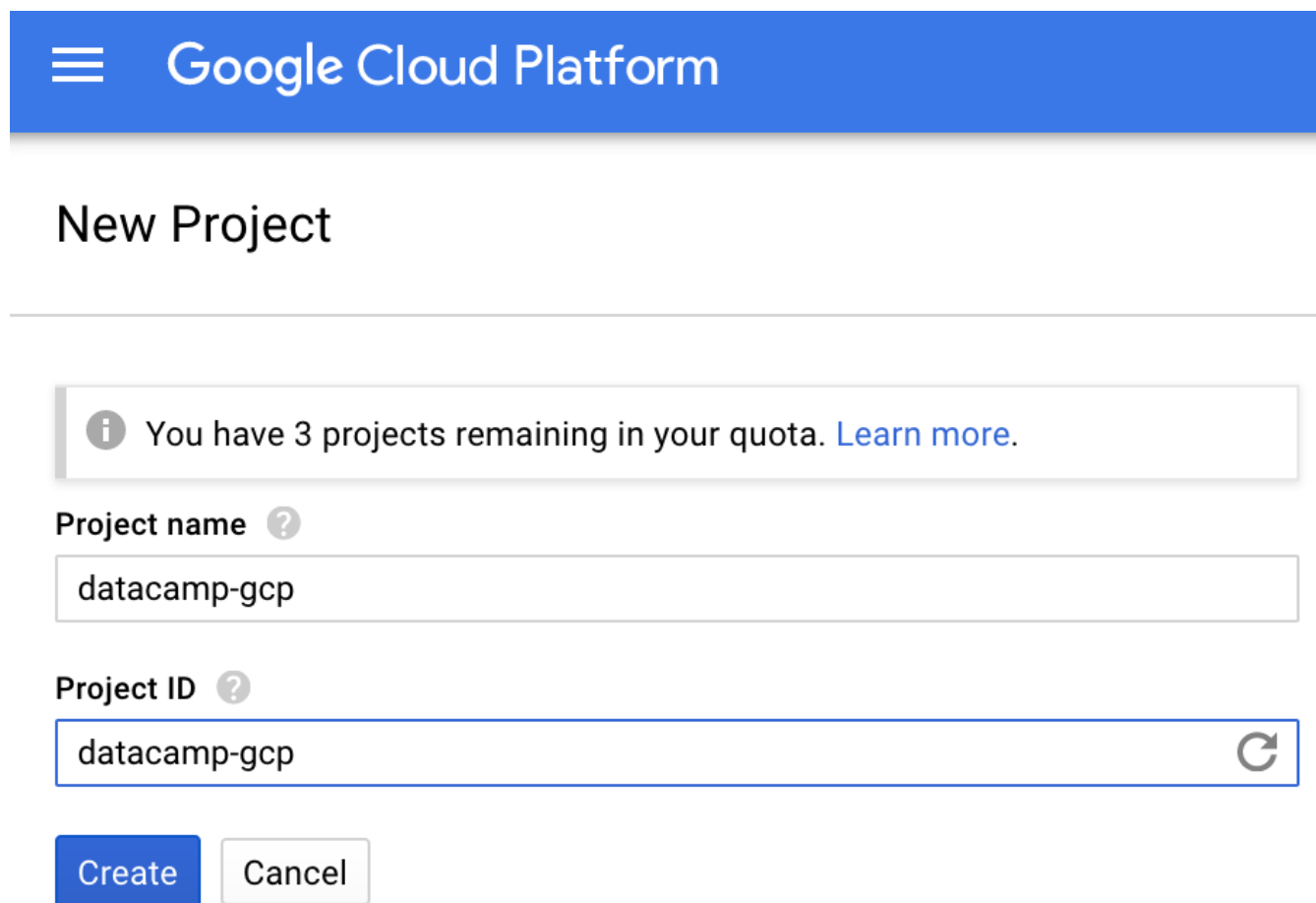
To create your GCP account with the free trial, go to cloud.google.com and click on the [try it free](#) button. You will be asked to login with your Google account and add your billing information. Once your account is created, you can access the web console at <http://console.cloud.google.com/>.

You will first use the web console to define and launch a Debian-based instance and then switch to the web-based shell terminal to install all the necessary packages for your data science stack.

But first you need to create a new project:

- Go to the [Resource Management](#) page,
- Click on "Create a new project",
- Specify your project's title and notice how Google generates a project ID on the fly.
- Edit the project ID as needed and click on "Create".

The project ID has to be unique across the GCP naming space, while the project title can be anything you want. I name my project `datacamp-gcp` as shown below:



The screenshot shows the Google Cloud Platform 'New Project' interface. At the top is a blue header with the Google Cloud Platform logo and name. Below the header, the title 'New Project' is displayed. A notification box states: 'You have 3 projects remaining in your quota. [Learn more.](#)'. The form contains two input fields: 'Project name' with a help icon and a question mark, containing the text 'datacamp-gcp'; and 'Project ID' with a help icon and a question mark, also containing 'datacamp-gcp'. A refresh icon is visible on the right of the Project ID field. At the bottom are two buttons: a blue 'Create' button and a grey 'Cancel' button.

By default, when you create a new project, your Google account is set as the owner of the project with full permissions and access across all the project's resources and billing.

In the roles section of the [IAM page](#), you can add people with specific roles to your project. For the purpose of this tutorial, you will skip that part and keep you as the sole user and admin of the project.

Create an instance

To create your first VM, you just go through the following steps:

1. Go to your dashboard at <https://console.cloud.google.com/home/dashboard> and select the project that you just created.
2. In the top left menu select "Compute Engine" and click on "VM instances".
3. In the dialog, click on the "Create" button.

Compute Engine VM instances

Compute Engine lets you use virtual machines that run on Google's infrastructure. You can choose from micro-VMs to large instances running Debian, Windows, or other standard images. Create your first VM instance, import it by CloudEndure migration service or try the quickstart to build a sample app.

[Create](#)

or

[Import](#)

or

[Take the quickstart](#)

You are now on the "Create an Instance" page.

1. Name the instance: you can choose any name you want. I name my instance `starling`.
2. Select the region: the rule of thumb is to select the cheapest region closest to you to minimize latency. I choose `east-d`. Note that prices vary significantly by region.

3. Select the memory, storage and CPU you need. You can use one of several presets or

3. Select the memory, storage and CPU you need. You can use one of several presets or customize your own instance. Here, I choose the default setup `n1-standard-1` with 3.75 Gb RAM and 1vCPU for an estimated price of \$24.67 per month.
4. Select the **boot disk** and go with the default Debian GNU/Linux 9 (stretch) OS with 10 Gb. If you prefer using Ubuntu or any other linux distribution, click on the "Change" button and make the appropriate selection. Since Ubuntu is a close derivation of Debian, either distributions will work for this tutorial.

← Create an instance

Zone ?

us-east1-b

Machine type

Customize to select cores, memory and GPUs.

1 vCPU

3.75 GB memory

[Customize](#)

Container ?

☐ Deploy a container image to this VM instance. [Learn more](#)

Boot disk ?



New 10 GB standard persistent disk
Image
Debian GNU/Linux 9 (stretch)

[Change](#)

5. Make sure you can access the VM from the internet by allowing http and https traffic.

Firewall ?

Add tags and firewall rules to allow specific network traffic from the Internet

- ☒ Allow HTTP traffic
- ☒ Allow HTTPS traffic

6. (Optional) Enable a persistent disk for backup purposes: Click on the "Management, Disks, networking, SSH keys" link. This displays a set of tabs, select the "Disks" tab and

unselect the "Deletion Rule". This way, when you delete your instance, the disk will not be deleted and can be used later on to spin up a new instance.

Management **Disks** Networking SSH Keys

Deletion rule

☐ Delete boot disk when instance is deleted

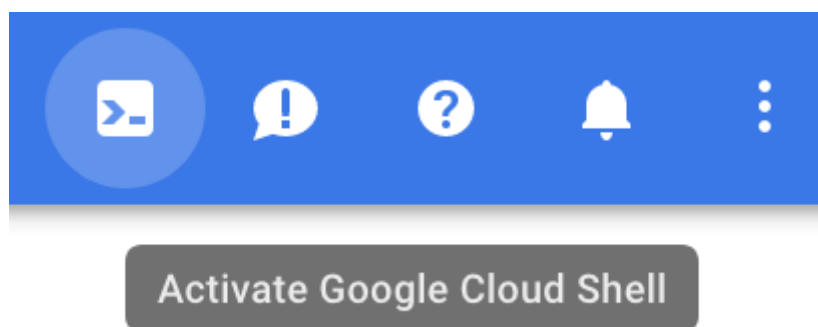
7. Finally, click on "Create". Your instance will be ready in a few minutes.

Notice the link "Equivalent command line" link below the `Create` button. This link shows the equivalent command line needed to create the same instance from scratch. This is a truly smart feature that facilitates learning the syntax of `gcloud` SDK.

At this point, you have a running instance which is pretty much empty. There are 2 ways you can access the instance. Either by installing the `gcloud` SDK on your local machine or by using Google's Cloud Shell.

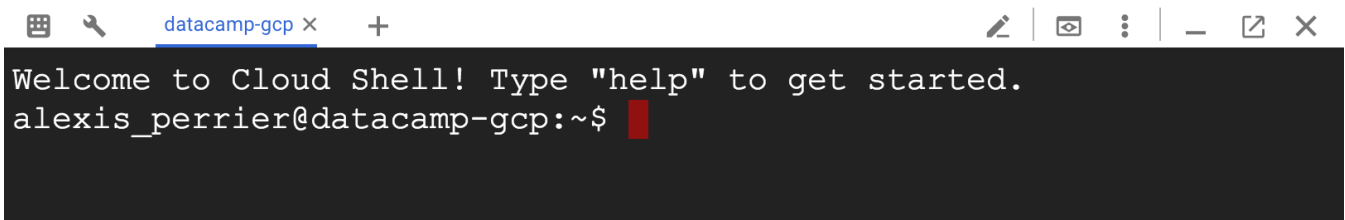
Google's Cloud Shell

Google's Cloud Shell is a stand alone terminal in your browser from which you can access and manage your resources. You activate the google shell by clicking the `>_` icon in the upper right part of the console page. The lower part of your browser becomes a shell terminal.



This terminal runs on a f1-micro Google Compute Engine virtual machine with a Debian operating system and 5Gb storage. It is created on a per-user, per-session basis. It persists while your cloud shell session is active and is deleted after 20 minutes of inactivity. Since the associated disk is persistent across sessions, your content (files, configurations, ...) will be available from session to session. The cloud shell instance comes pre-installed with the `gcloud` SDK and `vim`

comes pre-installed with the Google SDK and Python.



```
Welcome to Cloud Shell! Type "help" to get started.
alexis_perrier@datacamp-gcp:~$
```

It is important to make the distinction between the Cloud shell instance, which is user-based, and the instance you just created. The instance underlying the cloud shell is just a convenient way to have a resource management environment and store your configurations on an ephemeral instance. The VM instance that you just created, named `starling` in the above example, is the instance where you want to install your data science environment.

Instead of using the Google Cloud shell, you can also [install the gcloud SDK](#) on your local machine and manage everything from your local environment.

Here are a few useful commands to manage your instances.

- List your instances

```
gcloud compute instances list
```

- Stop the instance (takes a few seconds)

```
gcloud compute instances stop <instance name>
```

- Start the instance (also takes a few seconds)

```
gcloud compute instances start <instance name>
```

- and ssh into the `starling` instance

```
gcloud compute ssh <instance name>
```

Setting up the VM

Run that last command in your Google Shell window to log in your instance. The next steps will consists in:

1. Installing a few Debian packages with `apt-get install`.
2. Installing the Anaconda or Miniconda distribution.
3. Setting up the instance to make Jupyter notebooks securely accessible online.

Debian packages

Let's start by installing the Debian packages:

- `bzip2`, which is required to install Mini/Anaconda,
- `git`, which is always useful to have, and
- `libxml2-dev`, which is not required at this point but you will often need it when installing further Python libraries.

Run the following commands, which work for both Ubuntu and Debian, in the terminal:

```
$ sudo apt-get update
$ sudo apt-get install bzip2 git libxml2-dev
```

Anaconda / Miniconda

Once the above packages are installed, turn your attention to installing the Anaconda distribution for Python 3. You have the choice between installing the full Anaconda version which includes many scientific Python libraries, some of which you may not actually need or installing the lighter Miniconda version which requires you to manually install the Jupyter libraries. The process is very similar in both cases.

To install the lighter [Miniconda distribution](#), run

```
$ wget https://repo.continuum.io/miniconda/Miniconda3-latest-Linux-x86_64.sh
$ bash Miniconda3-latest-Linux-x86_64.sh
$ rm Miniconda3-latest-Linux-x86_64.sh
$ source .bashrc
$ conda install scikit-learn pandas jupyter ipython
```

The install shell script is downloaded and run with the first 2 lines. You should accept the

license and default location. In line 3, the no longer needed shell file is removed. Sourcing `.bashrc` on line 4, adds the `conda` command to your `$PATH` without having to open a new terminal. And finally, the last line installs the required python libraries:

```
scikit-learn pandas jupyter ipython .
```

The commands to install the full Anaconda distribution are very similar. Make sure to check the [download page](#) to get the latest version of the shell script file:

```
$ wget https://repo.continuum.io/archive/Anaconda3-5.0.1-Linux-x86_64.sh
$ bash Anaconda3-5.0.1-Linux-x86_64.sh
$ rm Anaconda3-5.0.1-Linux-x86_64.sh
$ source .bahsrc
```

To verify that everything is installed properly, check your python version with `python --version` and verify that the right python is called by default with the command `which python`. You should be getting something similar to

```
alexis_perrier@starling:~$ python --version
Python 3.6.3 :: Anaconda, Inc.
alexis_perrier@starling:~$ which python
/home/alexis_perrier/miniconda3/bin/python
```

You now have a working Python environment with the standard data science libraries installed (`sklearn` , `pandas`).

Allowing Web Access

The third and final step is to configure your VM to allow web access to your Jupyter notebooks.


You first need to make the VM accessible from the web. To do that, you will create a firewall rule via the Google Cloud console. Go back to your [Instances dashboard](#) and in the top left menu, select "VPC Network > Firewall rules". Click on the "CREATE FIREWALL RULE" link and fill out the following values:

- **Name:** *jupyter-rule (you can choose any name)*
- **Source IP ranges:** *0.0.0.0/0*

- Specified protocols and ports: *tcp:8888*
- and leave all the other variables to their default values.

The form should look like:

Source IP ranges ?

0.0.0.0/0 


Second source filter ?

None

Protocols and ports

☐ Allow all

☒ Specified protocols and ports

tcp:8888 


This firewall rule allows all incoming traffic (from all IPs) to hit the port 8888.

Using the "Equivalent command line link", you can see that firewall rule can also be created from the terminal with the following command line:

```
$ gcloud compute --project=datacamp-gcp firewall-rules create jupyter-rule --direction=INGRESS --priority=1000 --network=default --action=ALLOW --rules=tcp:8888 --source-ranges=0.0.0.0/0
```

Now go back to the VM page (top left menu > Compute Engine > VM instances), click on your VM name.

Make a note of your VM IP address. This is the IP address that you will use in your browser to access your Jupyter environment. In my example, the IP address is: 35.196.81.49 , yours will be different.

Network interfaces					
Network	Subnetwork	Primary internal IP	Alias IP ranges	External IP	IP forwarding
default	default	10.142.0.2	—	35.196.81.49 (ephemeral) 	Off

and make sure the Firewall rules are checked:

Firewalls

- ☒ Allow HTTP traffic
- ☒ Allow HTTPS traffic

Jupyter Configuration

Jupyter notebooks come with a configuration file that needs to be generated and edited in order to setup online access to your notebooks. In the terminal, run

```
jupyter notebook --generate-config to generate the configuration file. And  
jupyter notebook password to generate a password.
```

Tip: make sure that this is a strong password!

Now edit the configuration file you just created with

```
vim .jupyter/jupyter_notebook_config.py and add the following line at the top of the  
file
```

```
c.NotebookApp.ip = '*'
```

(to switch to edit mode in vim, just type the `i` character). Quit and save with the following sequence `ESC:wq`.

This will allow the notebook to be available for all IP addresses on your VM and not just the `http://localhost:8888` URL you may be familiar with when working on your local machine.

Launch

You are now ready to launch your Jupyter notebook with the command line:

```
$ jupyter-notebook --no-browser --port=8888
```

And you should see something like that in the terminal:

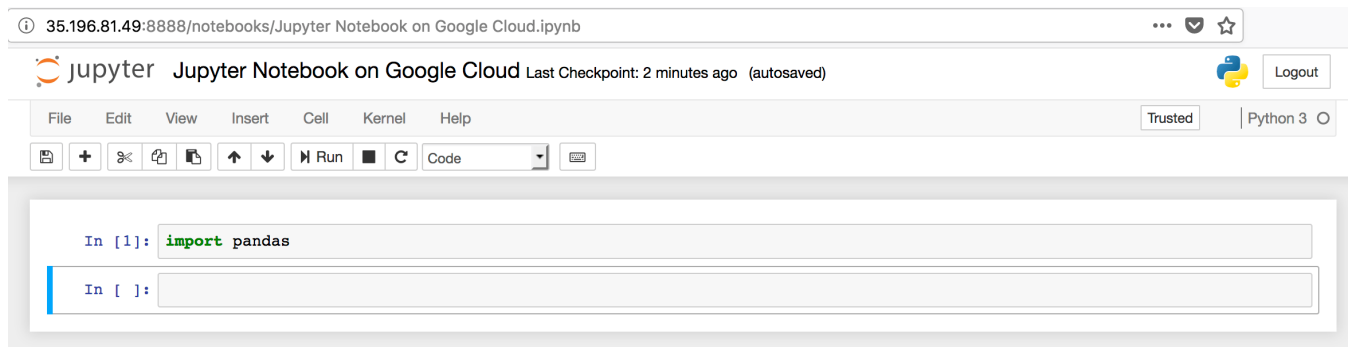
```
alexis_perrier@starling:~$ jupyter-notebook --no-browser --port=8888  
[W 23:03:01.469 NotebookApp] WARNING: The notebook server is listening on all IP addresses and not using encryption. This is not recommended.  
[I 23:03:01.480 NotebookApp] Serving notebooks from local directory: /home/alexis_perrier  
[I 23:03:01.480 NotebookApp] 0 active kernels  
[I 23:03:01.480 NotebookApp] The Jupyter Notebook is running at:  
[I 23:03:01.480 NotebookApp] http://all ip addresses on your system:8888/
```

```
[1 23:03:01.480 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
```

In your browser go to the URL: `http://<your_VM_IP>:8888/` to access your newly operational Jupyter notebook.

To check that everything is working as expected, create a new Python 3 notebook, and

```
import pandas
```



Intermezzo

A few notes on your current setup:

- The IP address you used in your browser is ephemeral. Which means that every time you restart your VM, your notebooks will have a different URL. You can make that IP static by going to: *Top left menu > VPC Network > External IP addresses* and select "static" in the drop down menu

External IP addresses						
			RESERVE STATIC ADDRESS	REFRESH	RELEASE STATIC ADDRESS	
<input type="checkbox"/> Name	External Address	Region	Static Ephemeral	Version	In use by	Labels
—	35.196.81.49	us-east1		IPv4	VM instance starling (Zone b)	

- The security of your current setup relies on the strength of the Jupyter notebook password that you defined previously. Anyone on the internet can access your Jupyter environment at the same URL you use (and bots will absolutely try). One powerful but very unsecure core feature of Jupyter notebooks is that you can launch a terminal with `sudo` access directly from the notebook. This means that anyone accessing your notebook could take control of your VM after cracking your notebook password and potentially run anything that would send your bills through the roof. The first level measures to prevent that from happening includes
- Making sure your Jupyter password is a strong one

- Remember to stop your VM when you're not working on it

You are also currently running the server over http and not https which is not secure enough. [Let's Encrypt](#) provides free SSL/TLS certificates and is the encryption solution recommended in the [jupyter documentation](#). For more information on security issues related to running a public Jupyter notebook, read [this](#).

IPython

Jupyter notebooks are very convenient for online collaborative work. But you can also run an IPython session from your terminal simply with the `ipython` command. That will open an IPython session which has all the bells and whistles of a Jupyter notebook such as magic commands (`%paste`, `%run`, ...) but without the web interface.

R




It's easy to set up your VM to be enable R notebooks in your Jupyter console. The instructions for enabling R Markdown are available from this great DataCamp tutorial: [Jupyter And R Markdown: Notebooks With R](#) by Karlijn Willems.

Conclusion

Google Cloud offers many interesting services for data science and powerful yet easy to setup VM instances alongside a very attractive free trial offer. The web console is easy to navigate and often displays the command line equivalent to current configuration pages, thus lowering the barrier to using the gcloud SDK.

In this article, you've learned how to select and launch a VM instance, install the necessary Debian packages, Anaconda distribution and data science stack and finally how to setup the access rules to launch a Jupyter notebook accessible from your browser.

Although the whole process may seem a bit complex the first time, it will quickly become familiar as you create and launch more and more VMs. And in case things become a bit too muddled, you can always delete the VM you're working on and restart from scratch. That's one of the perks of working in the cloud. After a few times you will be able to spin

data science environments on Google Cloud in a few minutes. Feel free to share your comments and questions with me on twitter: [@alex](#)   

70

41

COMMENTS

dthboyd

13/02/2018 08:16 AM

35.192.42.58 refused to connect.

Please

 1  **REPLY****Alexis Perrier**

14/02/2018 05:53 PM

Each VM gets a different IP. Your IP would be different.

 4  **REPLY****Jean-Claude Razafindrakoto**

15/02/2018 05:23 PM

Hi @alexisp, did you try to train a ML model, even a very simple one on GCP ?

 3  **REPLY****Alexis Perrier**

15/02/2018 05:30 PM

yes absolutely, I've worked with classic sklearn models or Deep Learning with Keras.

I was surprised at first that the performance was not really that faster than what I was getting with my macbook pro.

If that's the case you need to bump up the instance's number of CPUs to get better perfs.

 4  **REPLY****Tomislav Gelo**

17/02/2018 11:18 AM

Thank you for sharing this!

While installing the miniconda/anaconda package I get the following error.

*Miniconda3-latest-Linux-x86_64.sh: line 333: bunzip2: command not found*

tar: This does not look like a tar archive



tar: Exiting with failure status due to previous errors

Do I need to install some kind of unzip tool first? Did you also had this issue?

Thanks,

Tomi

▲ 2 ↩ REPLY

Alexis Perrier

17/02/2018 04:32 PM

This line should have installed bzip2 which is the unzipppper tool needed to install miniconda

```
$ sudo apt-get install bzip2 git libxml2-dev
```

Did you run that ?

▲ 2 ↩ REPLY

Tomislav Gelo

19/02/2018 09:20 AM

Seems like I missed that one somehow!

Thank you so much!

▲ 2

Javed Halani

20/02/2018 06:47 PM

Thanks for the great article !!

▲ 2 ↩ REPLY

oadeyemi

21/02/2018 09:52 PM

Hi Alexis, when I run the command `gcloud compute --project=my_project_name firewall-rules create jupyter-rule --direction=INGRESS --priority=1000 --network=default --action=ALLOW --rules=tcp:8888 --source-ranges=0.0.0.0/0`, I get the error below

ERROR: (gcloud.compute.firewall-rules.create) Could not fetch resource: - Insufficient Permission

PERMISSION.

Can you please advise on what I'm doing wrong

Thanks

▲ 2 ↩ REPLY

Thomas Dirks

22/02/2018 04:16 PM

You are probably running this on your newly created instance and not the Cloud shell instance. So you have to run it before connecting to your instance with `gcloud compute ssh <instance name>`.

▲ 2 ↩ REPLY

oadeyemi

24/02/2018 04:21 PM

Thanks Thomas

▲ 1

Євген Захарченко

25/02/2018 11:24 AM

Great! Thanks for the work

▲ 2 ↩ REPLY

Jeff Hendricks

08/03/2018 01:19 AM

`source .bahsrc` should be `.bashrc`

▲ 2 ↩ REPLY

Alexis Perrier

14/03/2018 11:14 PM

good catch! Thanks

▲ 1 ↩ REPLY

Tong Tat Chua

08/04/2018 07:56 AM

Hi,

How do i upload my data files into Google Cloud?

And how to i download my output files from Google Cloud?

▲ 3 ↩ REPLY

Srinivas Naredla

28/04/2018 06:26 PM

I too have same question. Did you figure this out?

▲ 1 ↩️ [REPLY](#)

Alexis Perrier

19/05/2018 02:09 PM

You use:

```
gcloud compute scp source_path target_path
```

so to download from an instance on gcloud to your local folder ./

```
gcloud compute scp [instance_name]:[path to file on instance] ./
```

and to upload some local file to your instance

```
gcloud compute scp [local file] . [instance_name]:[path]
```

▲ 2 ↩️ [REPLY](#)

suhha987

07/05/2018 08:49 PM

Hi @alexisp, Thank you for this tutorial, I now can run some python jobs in Compute Engine. Now I wonder if I can schedule some jobs to run regularly(like cron job) in Compute Engine? I googled, but the solution I've found so far doesn't look easy to me, do you have any suggestion?

▲ 1 ↩️ [REPLY](#)

Alexis Perrier

19/05/2018 02:19 PM

Cron is the way to go.

The trick is to use the full path to the python executable which you can find with which python

Then the following cron job will run your script every hour, 10 mn past the hour

```
10 * * * * [full_path to python] [full path to your script] > [some log file]
```

▲ 1 ↩️ [REPLY](#)

Chandra Sutrisno Tjhong

15/06/2018 02:34 AM

Hi,

Thank you for providing this tutorial. After all the setup, do you know how can I access files uploaded in Google Cloud Storage from Jupyter Notebook?

▲ 2 ↩ REPLY

Jose Torres

17/06/2018 06:44 PM

Can you contrast this process with just using Cloud Datalab to set up the instance and notebook environment?

▲ 2 ↩ REPLY

yi xie

04/07/2018 07:44 AM

I've got my jupyter notebook ready and try to work with keras. I am wondering how to upload datasets to the vm?

▲ 1 ↩ REPLY

John Davis

09/07/2018 02:17 PM

Two questions. 1. How do we tag a tutorial for future reference? ie. in our datacamp account is there anyway to save this particular tutorial? 2. How to add a package to a google cloud datalabvm? I'm asking this because tensorflow has ffmpeg support if the underlying binary is installed on the vm. I've found notes about how to use apt inside of a datalab cell and it has supposedly installed ffmpeg, but when I ssh to the instance I do not find ffmpeg in my path.

▲ 1 ↩ REPLY

瑜隆 蔡

18/07/2018 04:14 PM

Hi @alexisp Thanks for this tutorial. I am wondering is possible install python 3.6 and jupyter notebook then work with BigQuery module.

I just new on GCD. I used a datalab method. but it was a bad experience when I try to plot using seaborn.

Any suggests?

▲ 1 ↩ REPLY

 1  [REPLY](#)**Rolando Villena**

03/08/2018 05:24 AM

Hi Alexis. This is a very useful article. Is it possible to run the Julia Language with Jupiter Notebook in GCP? If it is possible, could you please provide some overall steps. Thank you!

 1  [REPLY](#)**Sayak Paul**

22/08/2018 02:06 PM

This tutorial is a gem.

 2  [REPLY](#)**Jose Torres**

05/09/2018 10:35 PM

Awesome tutorial. FYI `c.NotebookApp.ip = '*'` yielded warnings when i initiated the jupyter server; specifying the VM's internal IP address instead of `*` fixed the issue.

 3  [REPLY](#)**Bin (Vince) Wang**

12/09/2018 05:05 AM

Great tutorial !

 2  [REPLY](#)**Andrey Arroyo**

28/09/2018 12:25 PM

Hi,

I don't seem to get this part to work "(to switch to edit mode in vim, just type the `i` character). Quit and save with the following sequence `ESC :wq`."

Thank you

 2  [REPLY](#)**Pedro Sanchez**

18/05/2019 07:12 PM

I have the same question

▲ 1 ↩ REPLY

Himanshu Agarwal

14/10/2018 04:36 AM

When opening in browser, it says, this site cant be reached!

▲ 1 ↩ REPLY

a boran

22/10/2018 09:09 PM

Firstly great tutorial, many thanks,

something wrong with jupyter I think. Found this fix tho:

<https://github.com/jupyter/notebook/issues/3946>

▲ 1 ↩ REPLY

Lawrence Krukrubo

07/11/2018 07:35 PM

refused to connect,

This site can't be reached

refused to connect.

Try:

▲ 1 ↩ REPLY

Jesper Termansen

31/12/2018 09:49 AM

Great tutorial - thank you.

Would it be possible to execute all the steps in the tutorial via scripts ? I mean - there is a lot of GUI steps that I would like to bypass with a smart setup involving scripts.

How would that work ? Can you recommend a blog on this topic or would it be possible to show how you do it ?

Thank you.

Cheers, Jesper.

▲ 1 ↩ REPLY

jihao yu

08/01/2019 09:33 AM

Jupyter notebook refuse to connect.

▲ 1 ↩ REPLY

Bean Smith

22/01/2019 10:29 AM

Many people are facing issue with email and maximum time the email issue is happening due to the server problem such as [Gmail error 007](#) and to solve this issue you need to disable Email Signature from antivirus, use Disk Cleanup.

▲ 1 ↩ REPLY

Rodrigo Ferreira

19/02/2019 10:03 AM

when I try to run

```
gcloud compute ssh <instance name>
```

I get the error:

```
gcloud crashed (OSError): [Errno 28] No space left on device: '/home/*****/.ssh'
```

(**** identify my account)

Any idea why this could be happeniing? It is the same error I get everytime I try to use Datalab

▲ 1 ↩ REPLY

Moses Njenga

13/04/2019 12:16 PM

Thanks for the steps.

How do you read local files using Jupyter notebooks on GCP?

Thanks

▲ 1 ↩ REPLY

Lakshmanan Subramanian

04/05/2019 02:17 PM

Hi,

when I issue command in the google cloud to launch the jupyter notebook (`$ jupyter-notebook --no-browser --port=8888`), I got the error message as below

Error Summary :

ValueError: " does not appear to be an IPv4 or IPv6 address

During handling of the above exception, another exception occurred:

socket.gaierror: [Errno -2] Name or service not known

Detailed Error Message from the Console :

```
File "/home/messagetolakshman/miniconda3/lib/python3.7/site-
packages/notebook/notebookapp.py", line 864, in _default_allow_remote  addr =
ipaddress.ip_address(self.ip) File
"/home/messagetolakshman/miniconda3/lib/python3.7/ipaddress.py", line 54, in ip_address
address)ValueError: " does not appear to be an IPv4 or IPv6 addressDuring handling of the
above exception, another exception occurred:Traceback (most recent call last): File
"/home/messagetolakshman/miniconda3/bin/jupyter-notebook", line 11, in <module>
sys.exit(main()) File "/home/messagetolakshman/miniconda3/lib/python3.7/site-
packages/jupyter_core/application.py", line 266, in launch_instance  return
super(JupyterApp, cls).launch_instance(argv=argv, **kwargs) File
"/home/messagetolakshman/miniconda3/lib/python3.7/site-
packages/traitlets/config/application.py", line 657, in launch_instance  app.initialize(argv) File
"</home/messagetolakshman/miniconda3/lib/python3.7/site-
packages/decorator.py:decorator-gen-7>", line 2, in initialize File
"/home/messagetolakshman/miniconda3/lib/python3.7/site-
packages/traitlets/config/application.py", line 87, in catch_config_error  return method(app,
*args, **kwargs) File "/home/messagetolakshman/miniconda3/lib/python3.7/site-
packages/notebook/notebookapp.py", line 1630, in initialize  self.init_webapp() File
"/home/messagetolakshman/miniconda3/lib/python3.7/site-
packages/notebook/notebookapp.py", line 1378, in init_webapp
self.jinja_environment_options, File
"/home/messagetolakshman/miniconda3/lib/python3.7/site-
packages/notebook/notebookapp.py", line 159, in __init__  default_url, settings_overrides,
jinja_env_options) File "/home/messagetolakshman/miniconda3/lib/python3.7/site-
packages/notebook/notebookapp.py", line 252, in init_settings
allow_remote_access=jupyter_app.allow_remote_access, File
"/home/messagetolakshman/miniconda3/lib/python3.7/site-packages/traitlets/traitlets.py",
line 556, in __get__  return self.get(obj, cls) File
"/home/messagetolakshman/miniconda3/lib/python3.7/site-packages/traitlets/traitlets.py",
line 535, in get  value = self._validate(obj, dynamic_default()) File
"/home/messagetolakshman/miniconda3/lib/python3.7/site-
packages/notebook/notebookapp.py", line 867, in _default_allow_remote  for info in
socket.getaddrinfo(self.ip, self.port, 0, socket.SOCK_STREAM): File
"/home/messagetolakshman/miniconda3/lib/python3.7/socket.py", line 748, in getaddrinfo
for res in _socket.getaddrinfo(host, port, family, type, proto, flags):socket.gaierror: [Errno -2]
Name or service not known
```

Cristian Cucunuba

04/07/2019 04:48 AM

instead of `c.NotebookApp.ip = '*'`

do :

```
c.NotebookApp.ip = '0.0.0.0'
```

▲ 1 ↩ REPLY

Kevin-Morris Wigand

11/09/2019 09:01 AM

Hi Alexis,

thanks for the great tutorial.

when I want to open the ip (<ip>:8888) I get the following error :

```
" Network Error (tcp_error)
```

```
A communication error occurred: "Operation timed out" The Web Server may be down, too busy, or experiencing other problems preventing it from responding to requests. You may wish to try again at a later time. "
```

Do you have any advice how to fix that?

Thanks!

▲ 1 ↩ REPLY

 [Subscribe to RSS](#)



[About](#) [Terms](#) [Privacy](#)

Want to leave a comment?