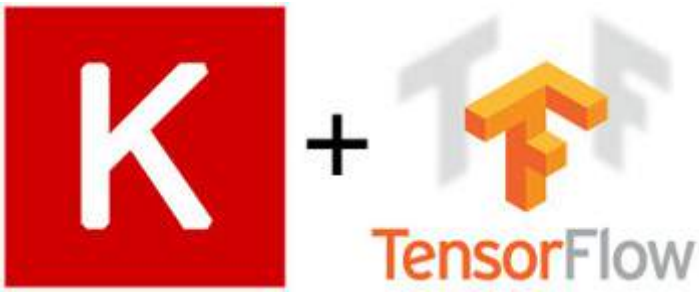
Deep Learning

Computer Vision

Keras



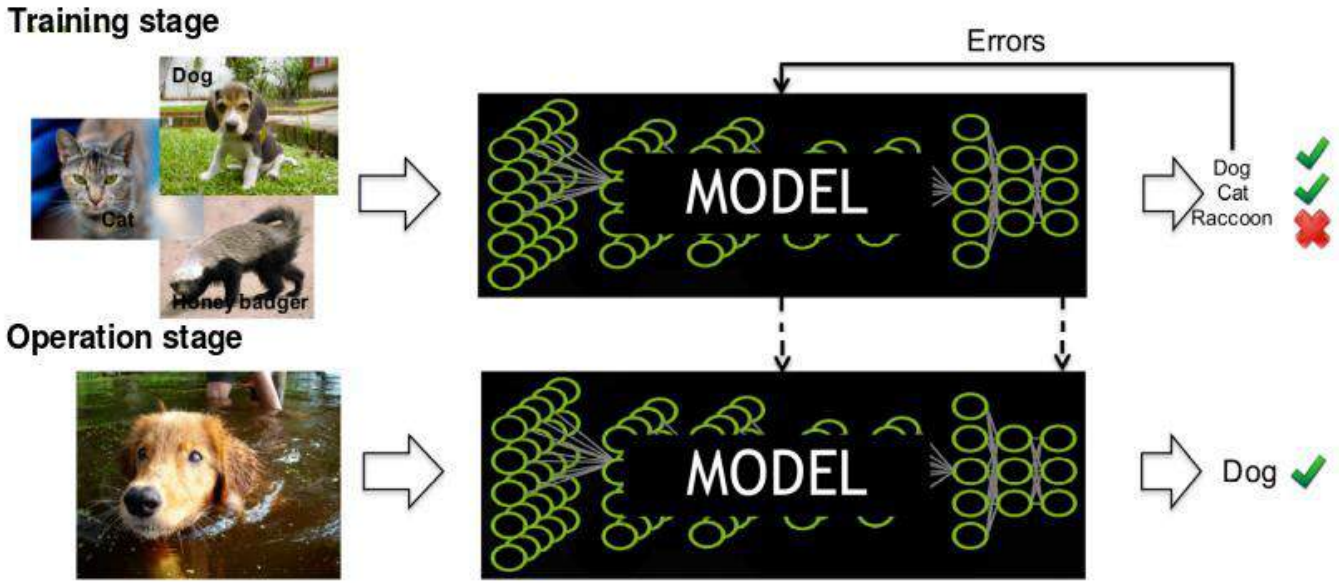
Luis Baumela (lbaumela@fi.upm.es (mailto:lbaumela@fi.upm.es))

Roberto Valle (rvalle@fi.upm.es (mailto:rvalle@fi.upm.es))

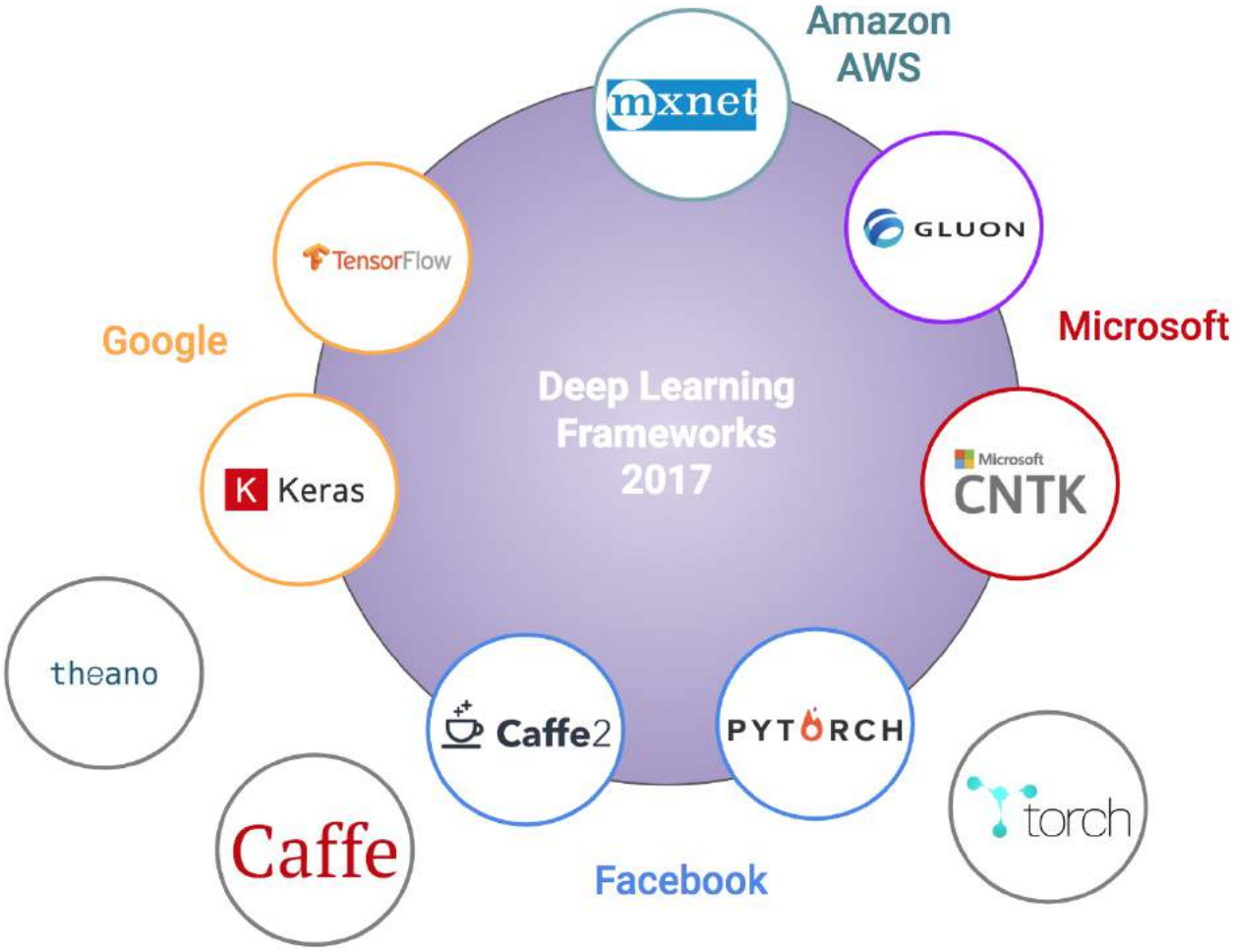
Machine Learning Life Cycle

How computers learn to recognize objects.

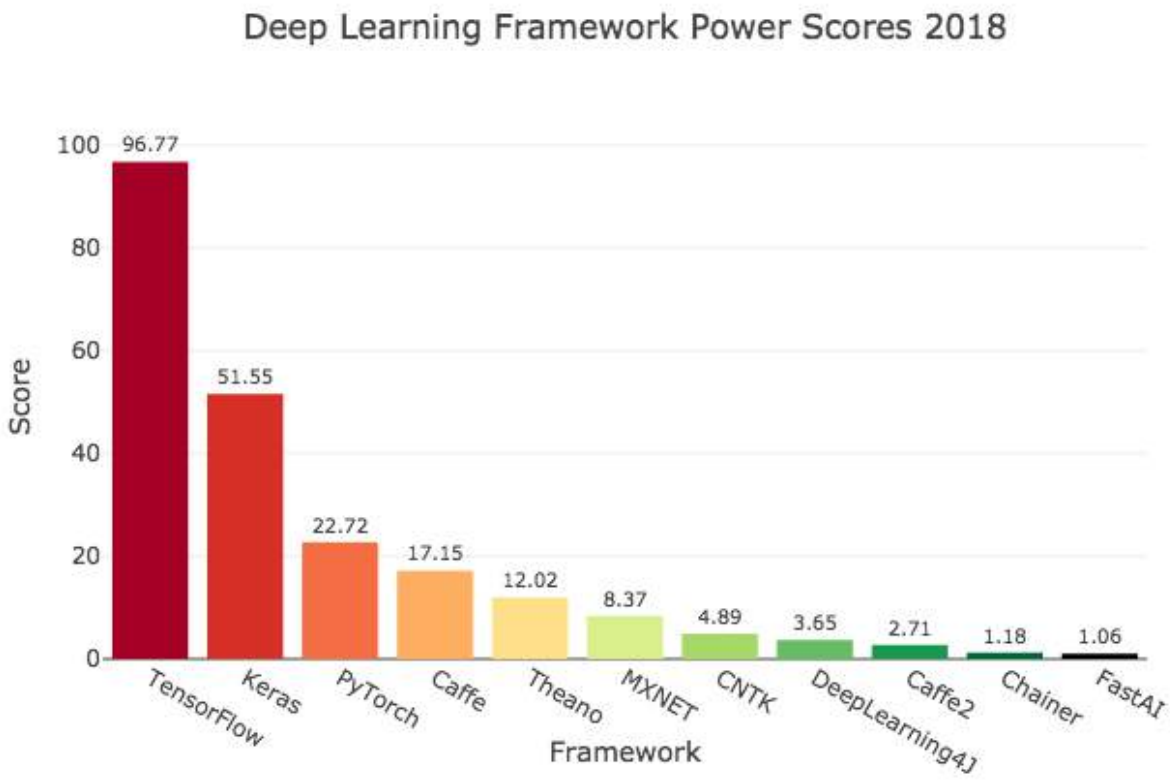
- **Training phase:** We train a machine learning algortihm using a data set and their corresponding labels.
- **Prediction phase:** We use the trained model to predict labels of unseen images.



Deep Learning Frameworks



Keras has broad adoption in the industry and the research community



Introducing Keras

- Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano https://keras.io/ (https://keras.io/)
- It was developed with a focus on enabling fast experimentation.
- Runs seamlessly on CPU and GPU.
- **User friendliness** Keras is an API designed for human beings, not machines.
- **Modularity** A model is understood as a sequence or a graph of standalone, fully-configurable modules that can be plugged together with as few restrictions as possible.
- **Easy extensibility** New modules are simple to add (as new classes and functions).
- **Work with Python** Models are described in Python code (compatible with Python 2.7-3.6), which is compact, easier to debug, and allows for ease of extensibility.

Try Keras: https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/guide/keras.ipynb (https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/guide/keras.ipynb)

What is a "backend"?

Keras is a model-level library, providing high-level building blocks for developing deep learning models. It does not handle itself low-level operations such as tensor products, convolutions and so on. Instead, it relies on a specialized, well-optimized tensor manipulation library to do so, serving as the "backend engine" of Keras.

At this time, Keras has three backend implementations available: the TensorFlow backend, the Theano backend, and the CNTK backend.

- TensorFlow is an open-source symbolic tensor manipulation framework developed by Google.
- Theano is an open-source symbolic tensor manipulation framework developed by LISA Lab at Université de Montréal.
- CNTK is an open-source toolkit for deep learning developed by Microsoft.

Keras main components

Keras is comprised of several components, some of which the user should be aware of:

- The core data structure of Keras is a **model**, a way to organize layers.
- The simplest type of model is the Sequential model, a linear stack of layers. For more complex architectures, you should use the Keras functional API, which allows to build arbitrary graphs of layers.
- Stacking layers is as easy as .add()
- The model needs to know what input shape it should expect. For this reason, the first layer in a Sequential model (and only the first, because following layers can do automatic shape inference) needs to receive information about its input shape.

In [1]:

```
from keras.models import Sequential
from keras.layers import Dense, Activation, Flatten

model = Sequential()
model.add(Flatten(input_shape=(32, 32, 3)))
model.add(Dense(units=64, activation='relu'))
model.add(Dense(units=10))
model.add(Activation('softmax'))
```

```
Using TensorFlow backend.
```

Keras main components

Keras is comprised of several components, some of which the user should be aware of:

- Before training a model, you need to **configure** the learning process, which is done via the compile method.
- It receives three arguments:
  - *Optimizer:* This object specifies the training procedure.
  - *Loss function:* The function to minimize during optimization.
  - *Metrics:* Used to monitor training.

In [2]:

```
# For a multi-class classification problem
model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accura
cy'])
model.summary()

# For custom metrics
import keras.backend as K
def categorical_accuracy(y_true, y_pred):
    return K.cast(K.equal(K.argmax(y_true, axis=-1),K.argmax(y_pred, axis=-1)), K.fl
oatx())
model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accura
cy', categorical_accuracy])
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_1 (Flatten)          (None, 3072)              0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                196672
_________________________________________________________________
dense_2 (Dense)              (None, 10)                650
_________________________________________________________________
activation_1 (Activation)    (None, 10)                0
=================================================================
Total params: 197,322
Trainable params: 197,322
Non-trainable params: 0
_________________________________________________________________
```

Keras main components

Keras is comprised of several components, some of which the user should be aware of:

- Fortunately, the keras.datasets module already includes methods to load and fetch popular reference **data sets**.
- Here's the list of available datasets:
  - Boston Housing (regression)
  - CIFAR10 (classification of 10 image labels)
  - CIFAR100 (classification of 100 image labels)
  - MNIST (classification of 10 digits)
  - Fashion-MNIST (classification of 10 fashion categories)
  - IMDB Movie Reviews (binary text classification)
  - Reuters News (multiclass text classification)

In [3]:

```
# Load CIFAR-10 data set
from keras.datasets import cifar10
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255.0
x_test /= 255.0

from keras.utils import np_utils
y_train = np_utils.to_categorical(y_train, 10)
y_test = np_utils.to_categorical(y_test, 10)
```

Keras main components

Keras is comprised of several components, some of which the user should be aware of:

- Keras models are trained on Numpy arrays of input data and labels. For **training** a model, you will typically use the fit() function
  - *Epoch:* one iteration over the entire input data (this is done in smaller batches).
  - *Batch size:* When passed NumPy data, the model slices the data into smaller batches and iterates over these batches during training.
  - *Validation data:* When prototyping a model, you want to easily monitor its performance on some validation data.

In [4]:

```
# Train the model, iterating on the data in batches of 32 samples
history = model.fit(x_train, y_train, epochs=10, batch_size=32)
```

```
Epoch 1/10
50000/50000 [==============================] - 5s 104us/step - loss: 2.
0985 - acc: 0.2078 - categorical_accuracy: 0.2078
Epoch 2/10
50000/50000 [==============================] - 5s 91us/step - loss: 1.9
938 - acc: 0.2449 - categorical_accuracy: 0.2449
Epoch 3/10
50000/50000 [==============================] - 5s 90us/step - loss: 1.9
692 - acc: 0.2493 - categorical_accuracy: 0.2493
Epoch 4/10
50000/50000 [==============================] - 4s 89us/step - loss: 1.9
570 - acc: 0.2520 - categorical_accuracy: 0.2520
Epoch 5/10
50000/50000 [==============================] - 4s 86us/step - loss: 1.9
510 - acc: 0.2559 - categorical_accuracy: 0.2559
Epoch 6/10
50000/50000 [==============================] - 4s 87us/step - loss: 1.9
460 - acc: 0.2551 - categorical_accuracy: 0.2551
Epoch 7/10
50000/50000 [==============================] - 5s 93us/step - loss: 1.9
408 - acc: 0.2536 - categorical_accuracy: 0.2536
Epoch 8/10
50000/50000 [==============================] - 4s 88us/step - loss: 1.9
371 - acc: 0.2590 - categorical_accuracy: 0.2590
Epoch 9/10
50000/50000 [==============================] - 5s 91us/step - loss: 1.9
344 - acc: 0.2562 - categorical_accuracy: 0.2562
Epoch 10/10
50000/50000 [==============================] - 4s 89us/step - loss: 1.9
335 - acc: 0.2567 - categorical_accuracy: 0.2567
```

Keras main components

Keras is comprised of several components, some of which the user should be aware of:

- For **evaluation** mode you have to use the evaluate() function. Its output is accuracy or loss, not prediction to your input data.
- Use predict() to **predict** the output of the last layer in inference for the data provided. Its output is target value, predicted from your input data.
- So if you use model.predict and then compute the metrics yourself, the computed metric value should turn out to be the same as model.evaluate.

In [5]:

```
accuracy = model.evaluate(x_test, y_test, batch_size=32)
print(accuracy)
print(model.metrics_names)
```

```
10000/10000 [==============================] - 0s 48us/step
[1.9546743999481202, 0.2591, 0.2591]
['loss', 'acc', 'categorical_accuracy']
```

In [7]:

```
prediction = model.predict(x_test, batch_size=32)
print(y_test[5])
print(prediction[5])
import numpy as np
print(np.argmax(prediction[5]))
```

```
[ 0.  0.  0.  0.  0.  0.  1.  0.  0.  0.]
[ 0.01300764  0.08614522  0.08283562  0.13504931  0.11599413  0.0901209
6
  0.26814824  0.12336987  0.01480679  0.07052226]
6
```

Prerequisites

CPU mode installation

```
pip install tensorflow keras
```

GPU vs CPU compilation

Check your NVIDIA drivers compatibility (CUDA 9.0 requires 384.x or higher).

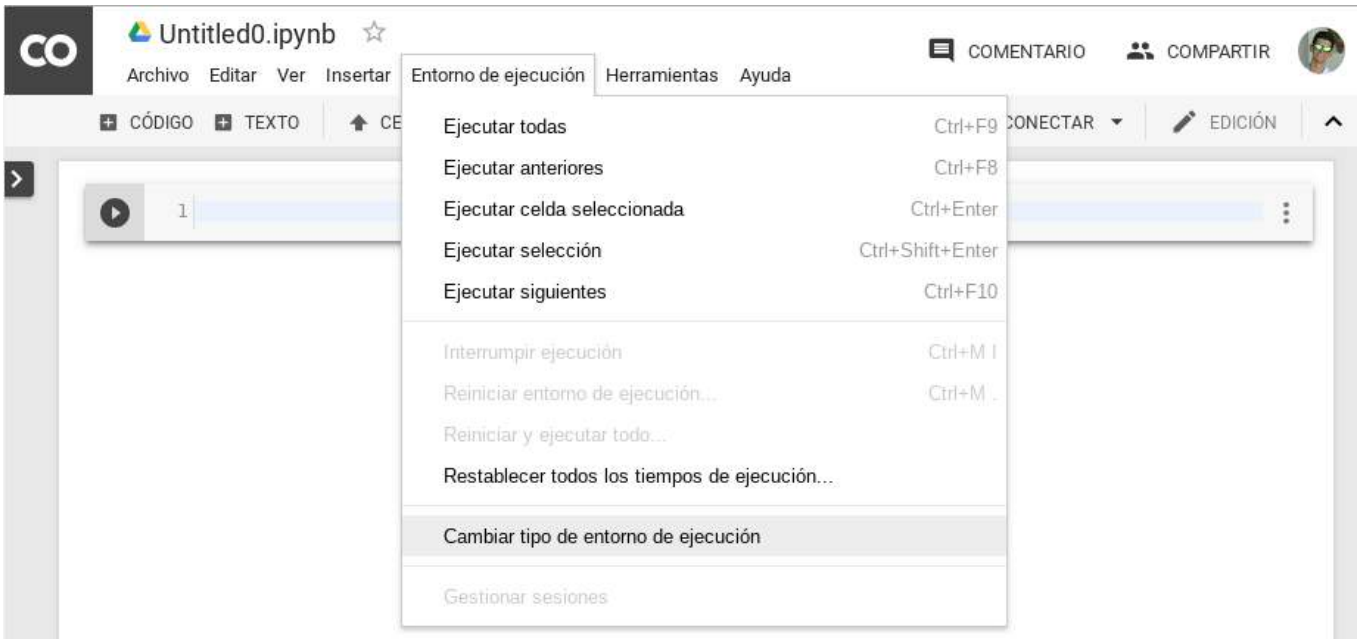
```
nvidia-smi
```

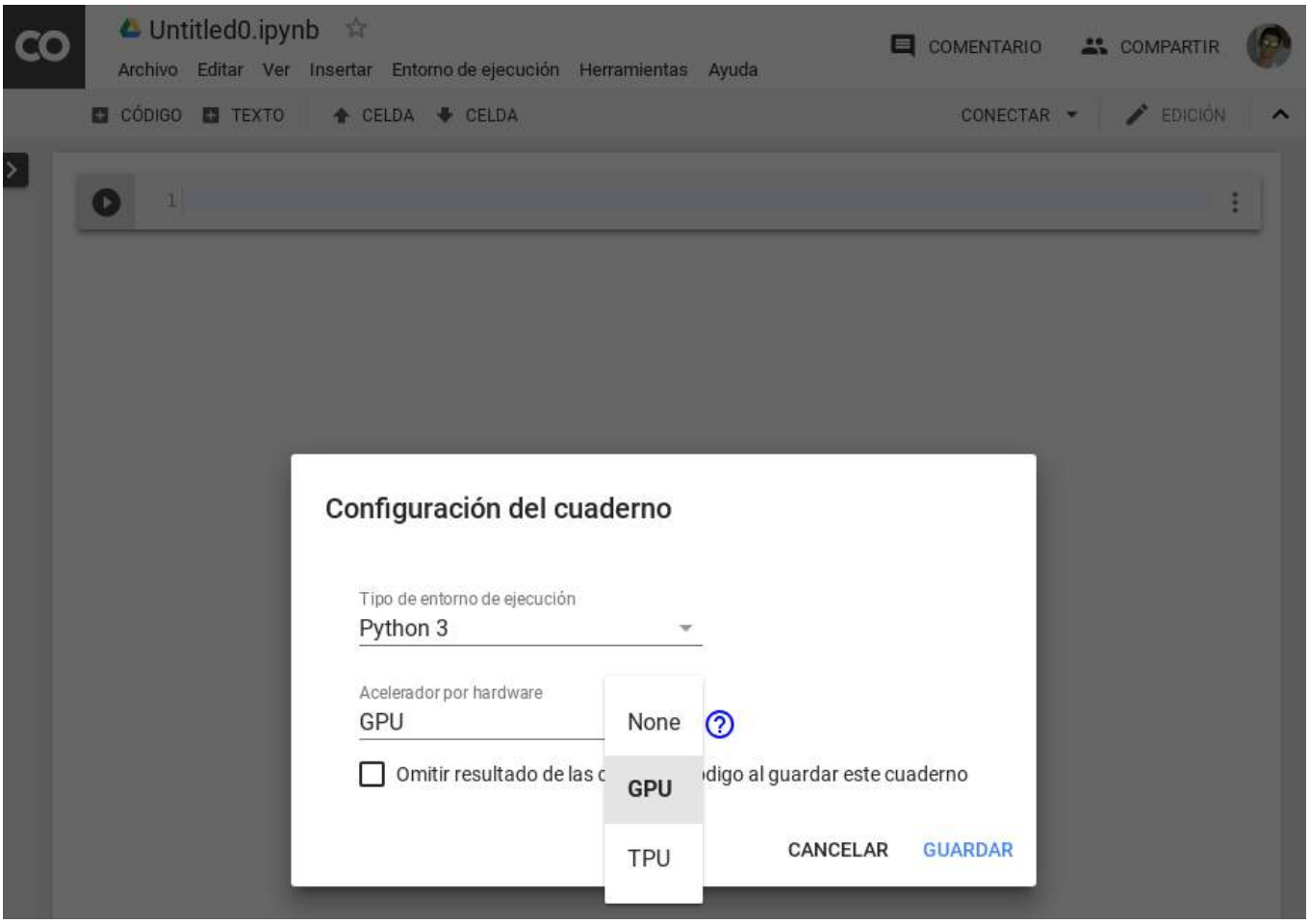
CUDA-9.0 (https://developer.nvidia.com/cuda-toolkit (https://developer.nvidia.com/cuda-toolkit)) and cuDNN (>= 7.0) (https://developer.nvidia.com/cudnn (https://developer.nvidia.com/cudnn)) are required for GPU acceleration mode.

```
nvcc --version
pip install tensorflow-gpu keras
```

Keras - Google Colab

- Google Colaboratory, Google’s free cloud service for AI developers, provides a free Tesla K80 GPU for up to 12 hours at a time.
- Colab is working on your own Google Drive.
- Create a new notebook via https://colab.research.google.com/ (https://colab.research.google.com/)
- Keras + Tensorflow GPU has been already installed.

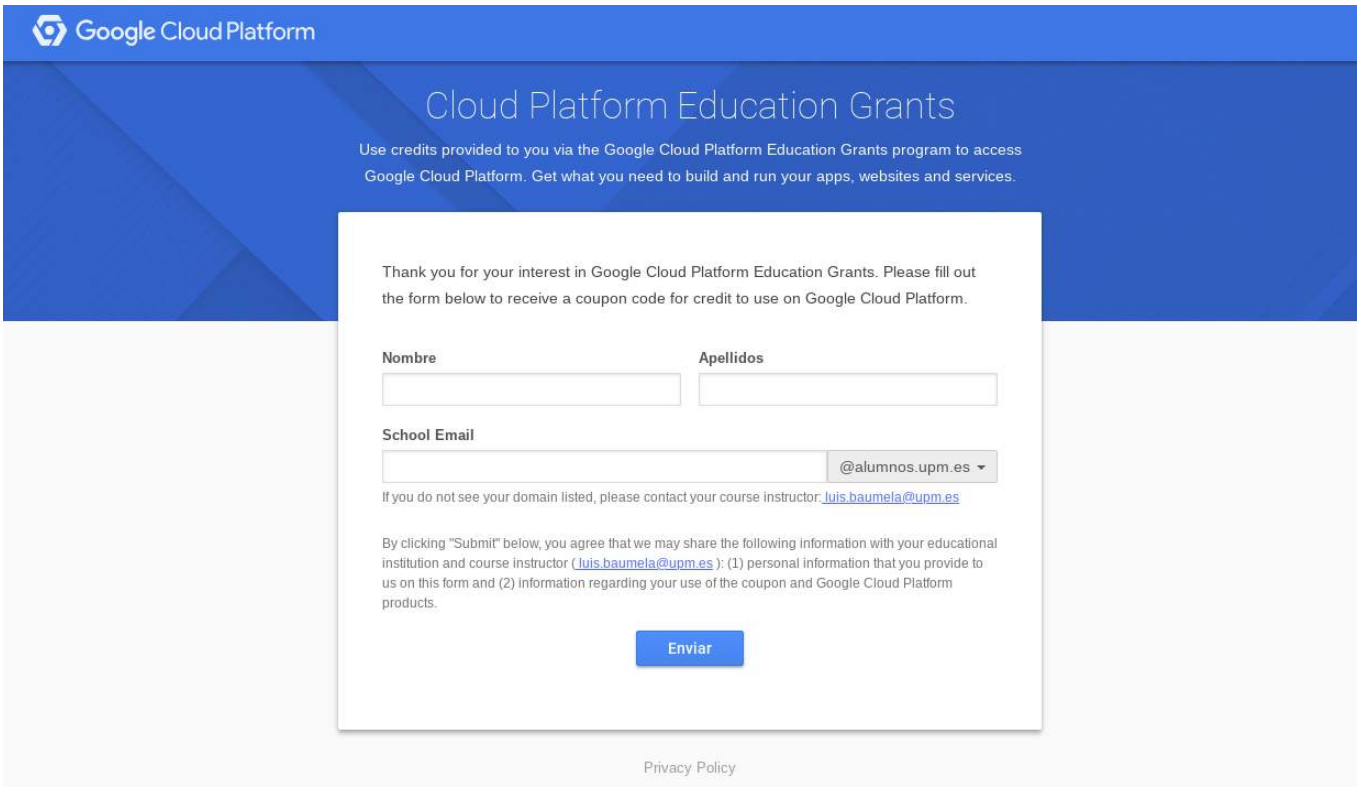



Untitled0.ipynb
Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda
COMENTARIO
COMPARTIR
CÓDIGO
TEXTO
CELDA
CELDA
CONECTAR
EDICIÓN
Configuración del cuaderno
Tipo de entorno de ejecución
Python 3
Acelerador por hardware
GPU
None
GPU
TPU
Omitir resultado de las c
digo al guardar este cuaderno
CANCELAR
GUARDAR

Keras - Google Cloud

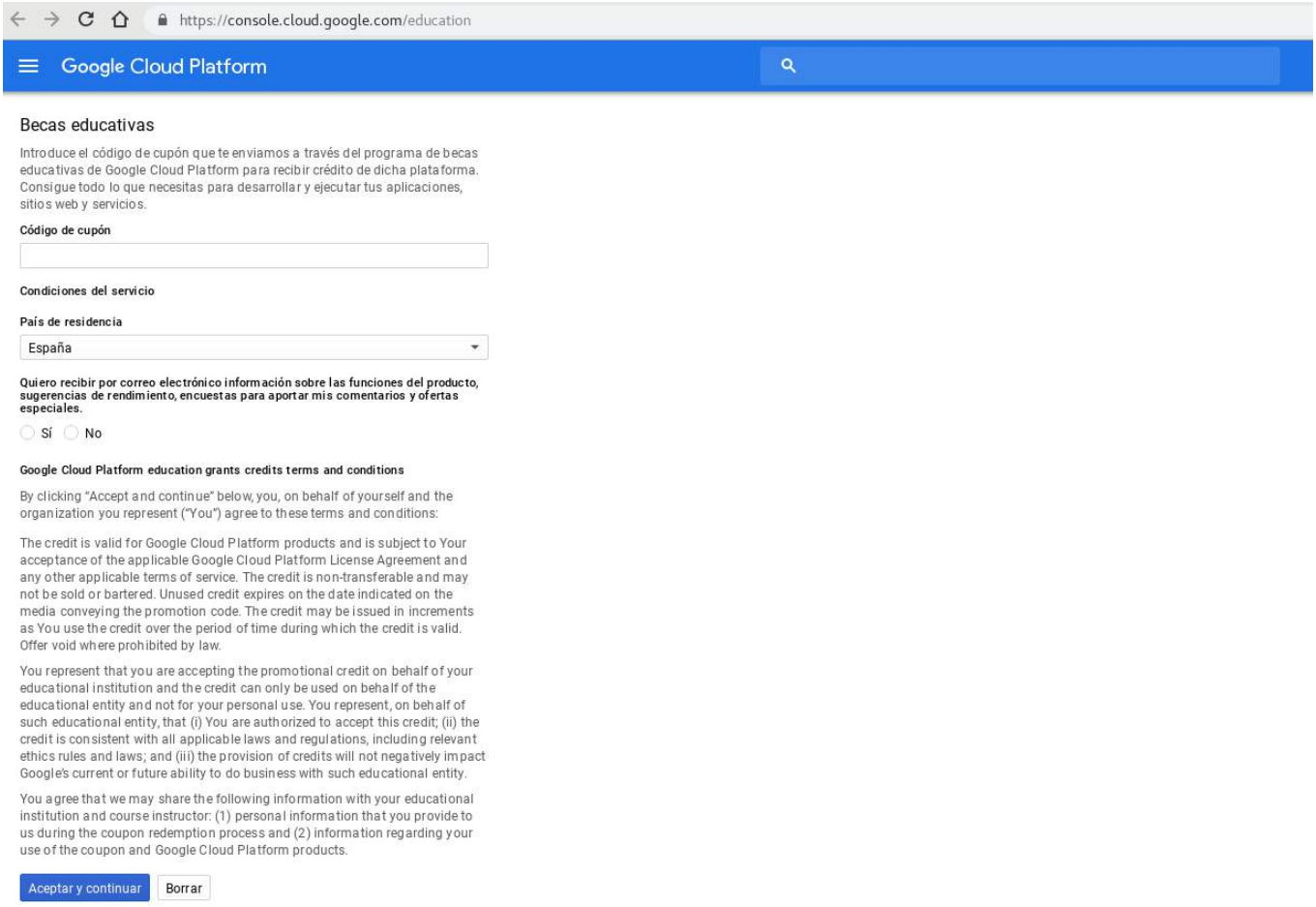
- **Create and Configure Your Account**

Here is the URL you will need to access in order to request a Google Cloud Platform coupon. You will be asked to provide your school email address and name. An email will be sent to you to confirm these details before a coupon is sent to you. Student Coupon Retrieval Link: http://google.force.com/GCPEDU?cid=dZf2ezmagvg3%2BlfNz91vlOdel%2Bxo2cB7uFnZ8enb9%2FH%2BMX96rFLc3omejA6paLgv/ (http://google.force.com/GCPEDU?cid=dZf2ezmagvg3%2BlfNz91vlOdel%2Bxo2cB7uFnZ8enb9%2FH%2BMX96rFLc3omejA6paLgv/)

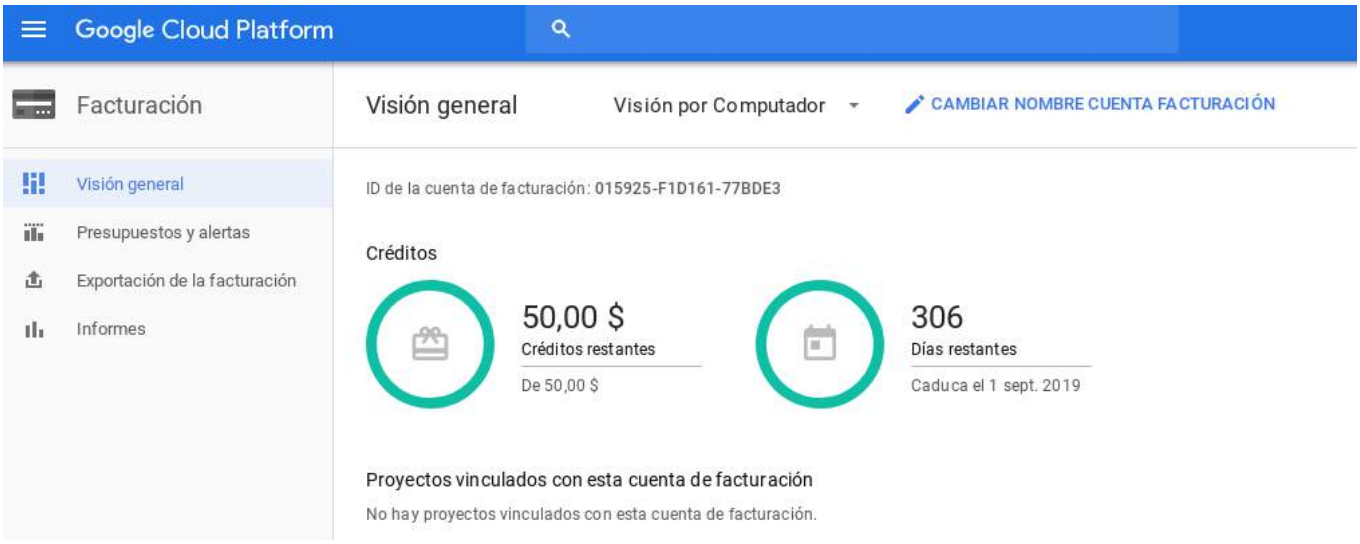


Sign into your gmail account or create a new one if you do not already have an account. You have received a single coupon which can be applied for credit at https://console.cloud.google.com/education (https://console.cloud.google.com/education)

IMPORTANT: You should NOT provide a credit card number to use the Students Grants Program. If you are asked for a credit card, you are probably using the Free Trial instead.

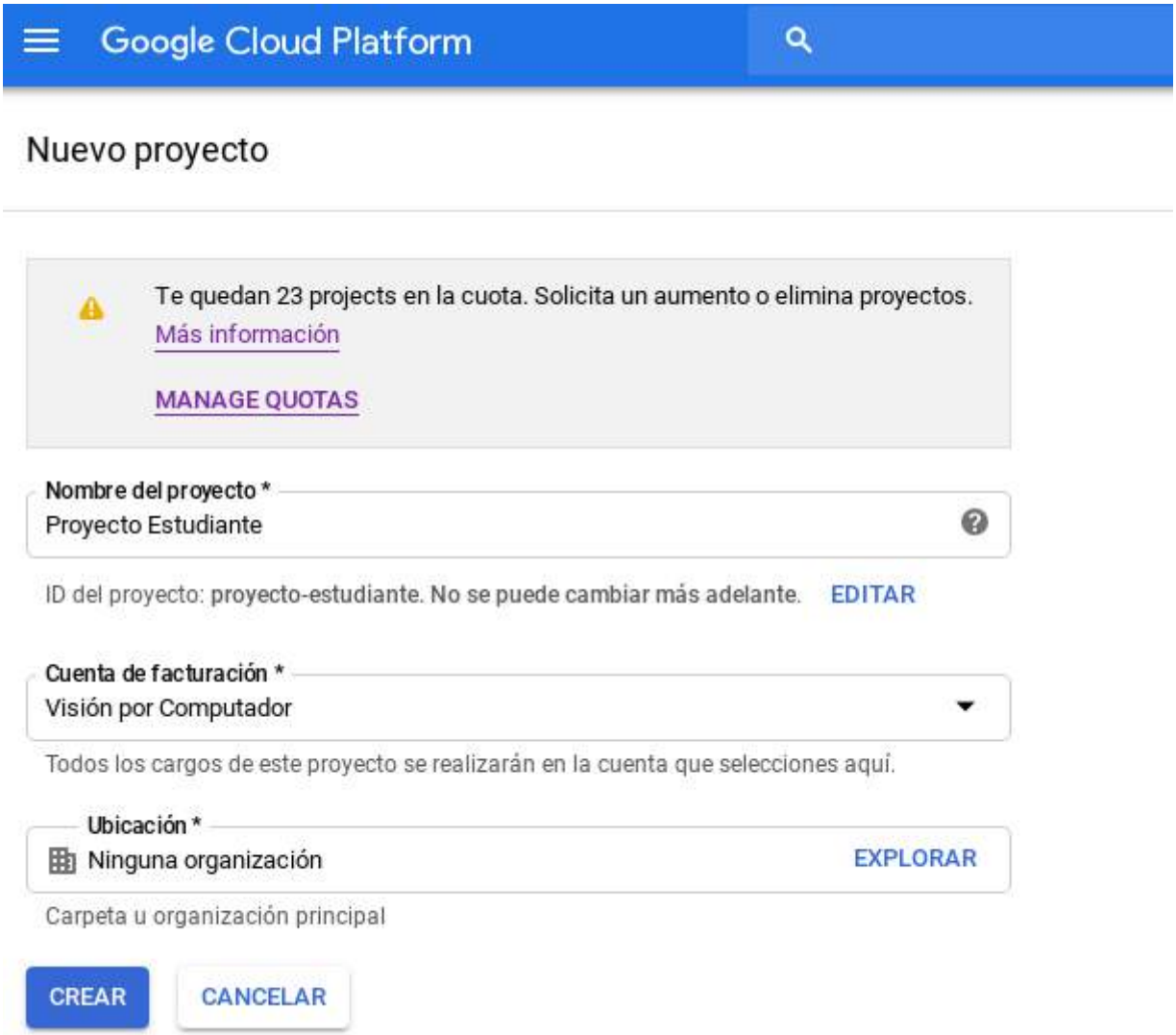


Each student will have 50$ in credit. Please try to use the resources judiciously.



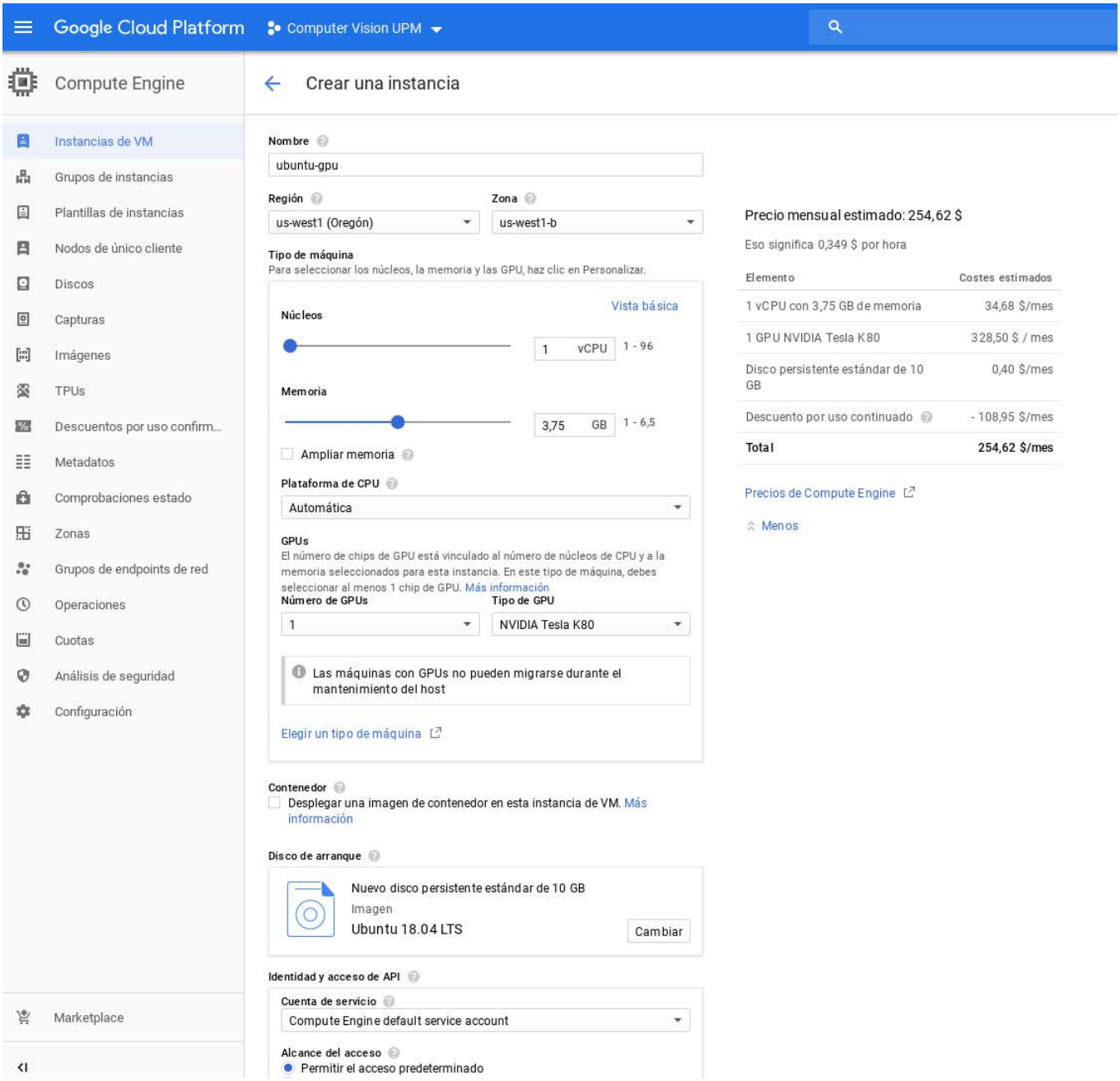
- **Create Project**

In the Google Developers Console, click Select a project > Create a project.

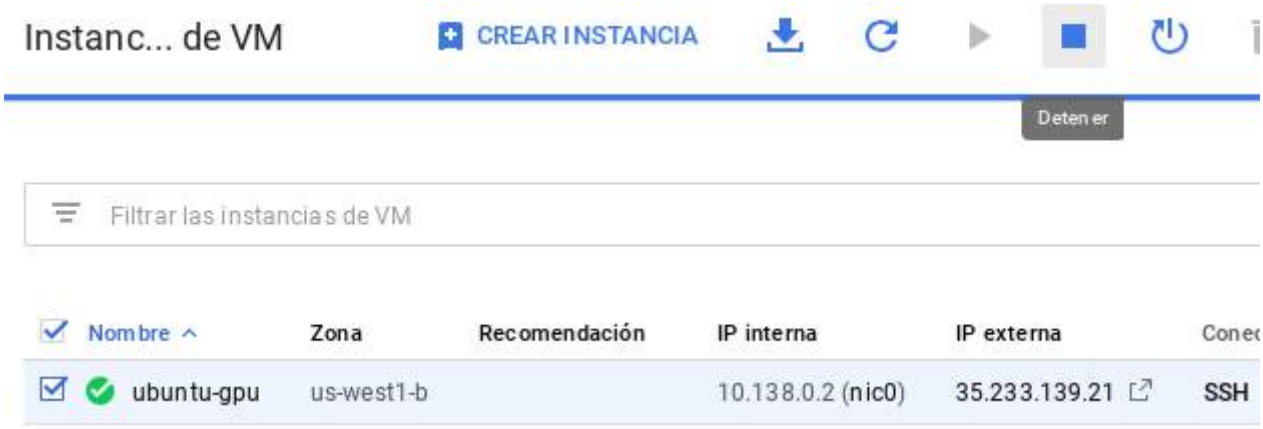


- **Launch a Virtual Instance**

To launch a virtual instance, go to the Compute Engine menu on the left column of your dashboard and click on VM instances. Then click on the blue Create button on the next page. This will take you to a page that looks like the screenshot below. Make sure that the Zone is set to be *us-west1-b* (especially for assignments where you need to use GPU instances). Check Allow HTTP traffic and Allow HTTPS traffic.



IMPORTANT: Make sure you stop your instances! Don't forget to stop your instance when you are done (by clicking on the stop button at the top of the page showing your instances), otherwise you will run out of credits and that will be very sad :(



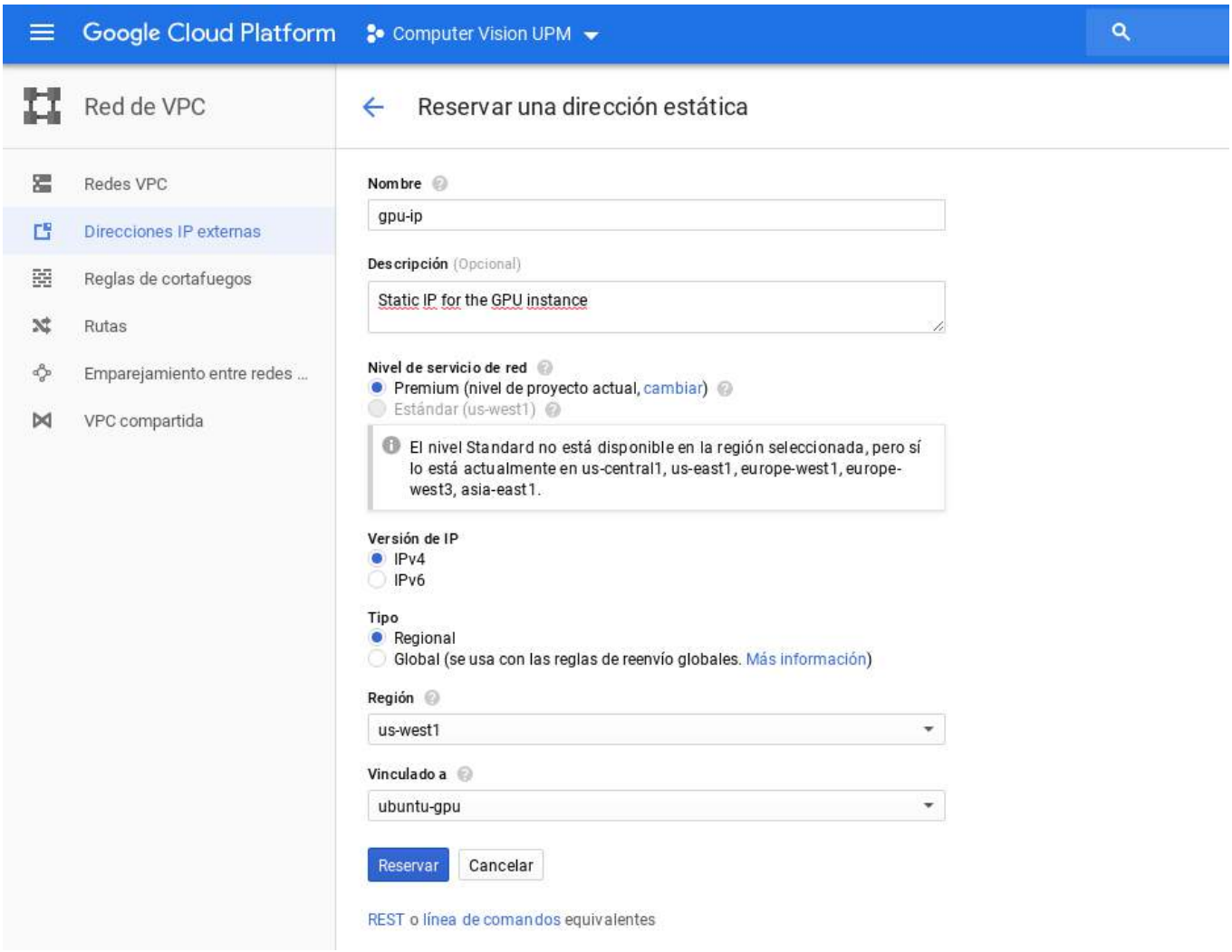
- **Using Jupyter Notebook with Google Cloud**

Many of the assignments will involve using Jupyter Notebook. Below, we discuss how to run Jupyter Notebook from your instance and connect to it with your local browser.

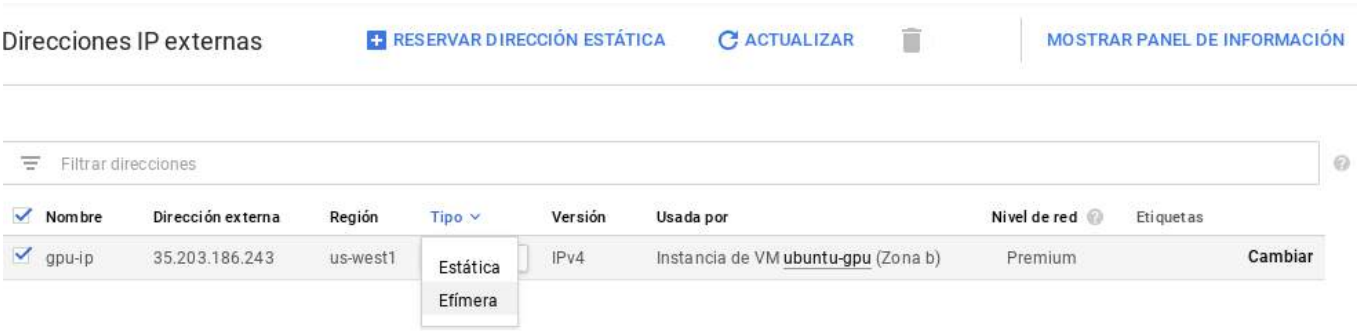
- Getting a Static IP Address
- Adding a firewall rule
- Launching and connecting to Jupyter Notebook

Getting a Static IP Address

Change the Extenal IP address of your instance to be static. To do this, click on the 3 line icon next to the Google Cloud Platform button on the top left corner of your screen, go to VPC network and External IP addresses (see screenshot below).



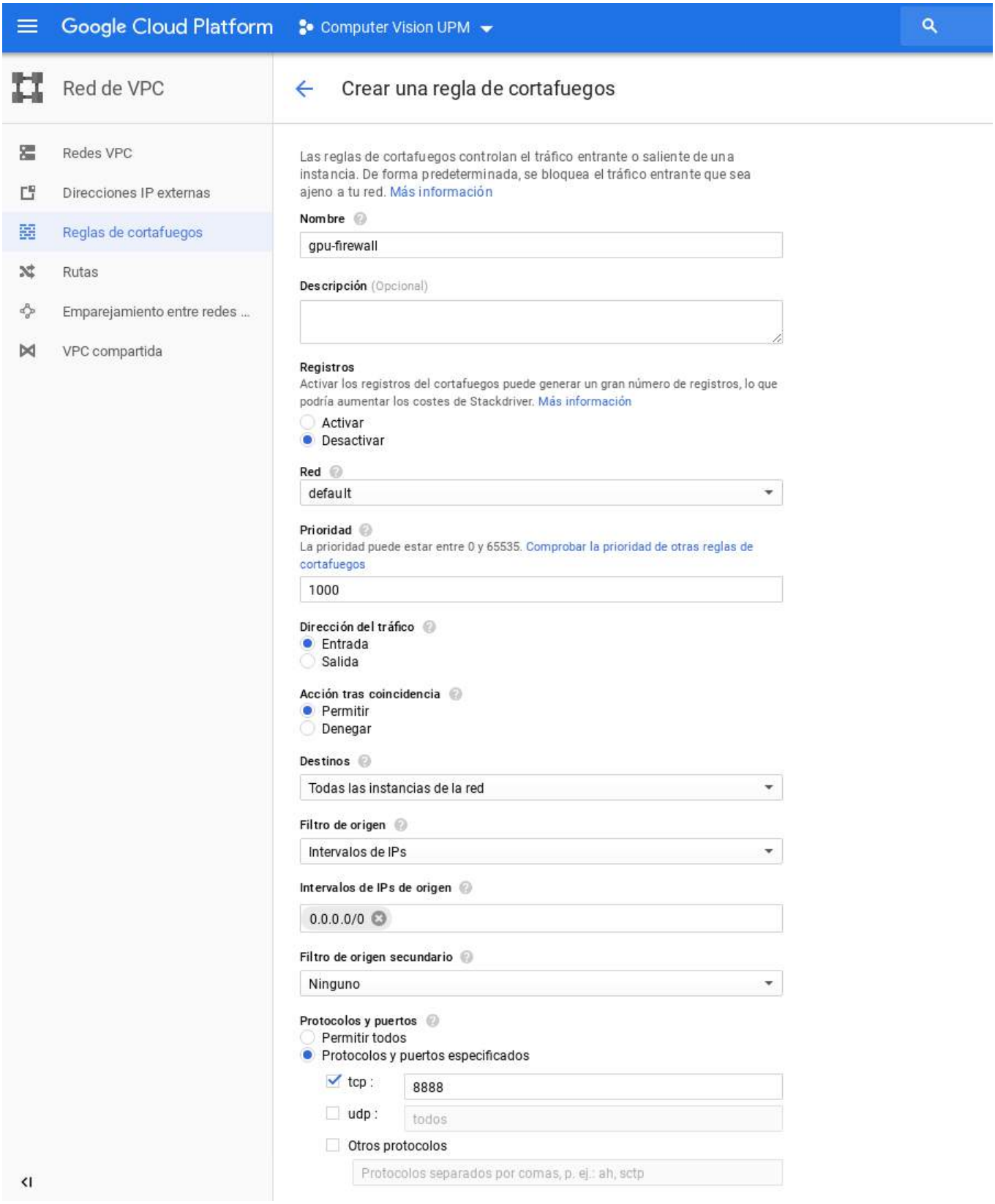
To have a static IP address, change Type from Ephemeral to Static. Enter your prefered name for your static IP, ours is gpu-ip (see screenshot below). And click on Reserve.



Remember to release the static IP address when you are done because according to this page (https://jeffdelaney.me/blog/running-jupyter-notebook-google-cloud-platform/ (https://jeffdelaney.me/blog/running-jupyter-notebook-google-cloud-platform/)) Google charges a small fee for unused static IPs.

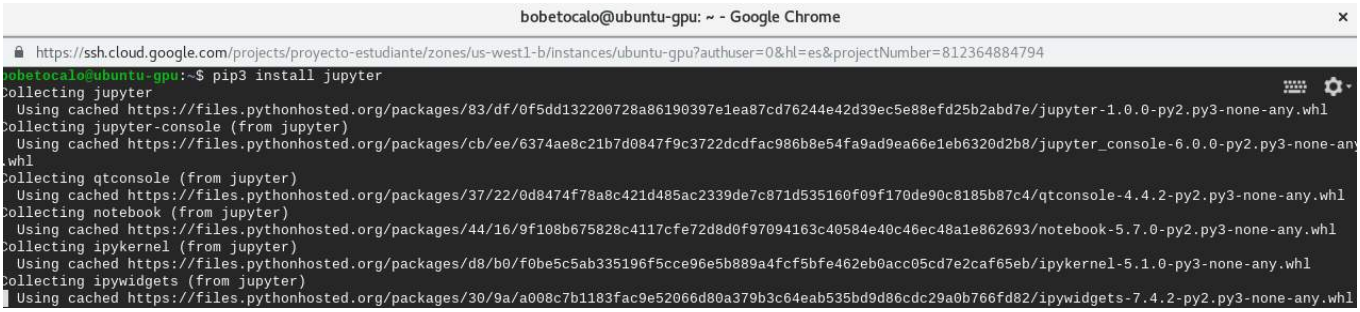
Adding a firewall rule

One last thing you have to do is adding a new firewall rule allowing TCP acess to a particular port number. The default port we use for Jupyter is 8888. On the menu that pops up on the left column, go to VPC network and Firewall rules. Click on the blue Create Firewall Rule button. Enter whatever name you want. Select “All instances in the network” for Targets (if the menu item exists). Enter 0.0.0.0/0 for Source IP ranges and tcp: 8888 for Specified protocols and ports where 8888 is the number you used above. Click on the blue Create button. See the screenshot below.



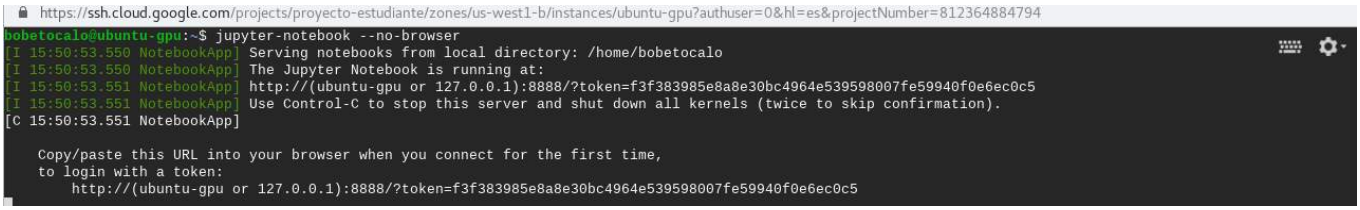
Launching and connecting to Jupyter Notebook

After you ssh into your instance, install Jupyter notebook and export PATH=$PATH:~/.local/bin



Run jupyter notebook --generate-config and edit the generated file with all the defaults commented out

To run the jupyter notebook, just type the following command in the ssh window you are in: jupyter-notebook --no-browser



You will get this IP address on terminal, block the whole IP addresses (with the token as well), it will automatically “cut” it. Paste it on our web browser, and change “0.0.0.0” with our VM static external IP.