**Manual RAPIDS**

**Instalación RAPIDS máquina personal.**

Dentro del proceso de instalación de RAPIDS en un computador portátil localmente se necesita de ciertos prerrequisitos para ser uso de la librería, se incorpora los puntos primordiales que se necesitan para la respectiva instalación previa:

|  |  |  |  |
| --- | --- | --- | --- |
| **GPU NVIDIDA** | **SO Ubuntu** | **Docker** | **CUDA and NVIDIA Drivers** |
| GeForce  Titan RTX  Tesla | Versión 16.04  Versión 18.04 | Versión 19.03 | Versión 10.0  Versión 10.1  Versión 10.2 |

EL computador portátil donde se instaló la librería RAPIDS cumple con los prerrequisitos mencionados anteriormente, detallo las especificaciones importantes que tiene el mismo:

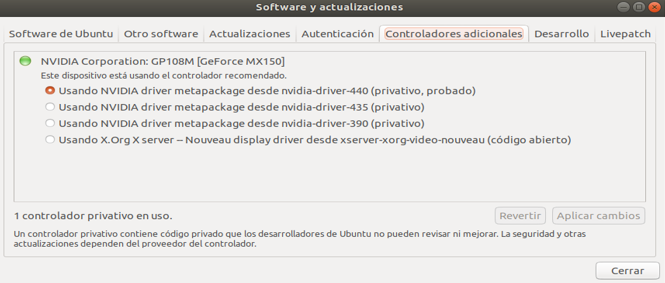
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Marca Portátil** | **Intel** | **Sistema Operativo** | **Tipo SO** | **GPU** |
| DELL | Core I7 8th Gen | Ubuntu 18.04 | SO de 64 bits | NVIDIA GeForce MX150 |

Cabe desatacar que si se requiere instalar la librería puntos primordiales debe tener el Sistema Operativo Ubuntu y que contenga una tarjeta gráfica en este caso NVIDIA, si no contiene lo mencionado no se podrá instalar la librería.

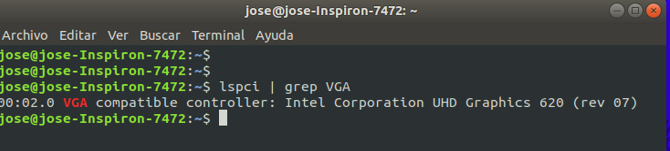
**Instalación e Implementación con Docker en computador local**

A continuación, se detalla el proceso de instalación de la librería RAPIDS con Docker:

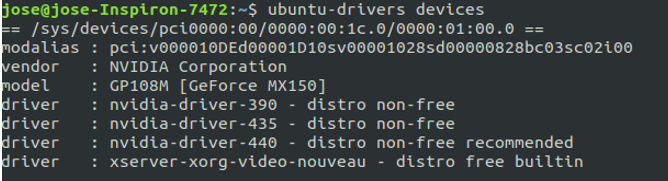
1. Verificar si tu Sistema Operativo Ubuntu tiene instalado y actualizado los drivers de NVIDIA.



1. Abrir un terminal escribir el siguiente comando para conocer la información del modelo de la tarjeta gráfica.



1. Seguidamente en el terminal escribir el siguiente comando, indicará que modelo y controlador está disponible a través de los canales oficiales del Sistema Operativo Ubuntu, según la versión que estemos utilizando.

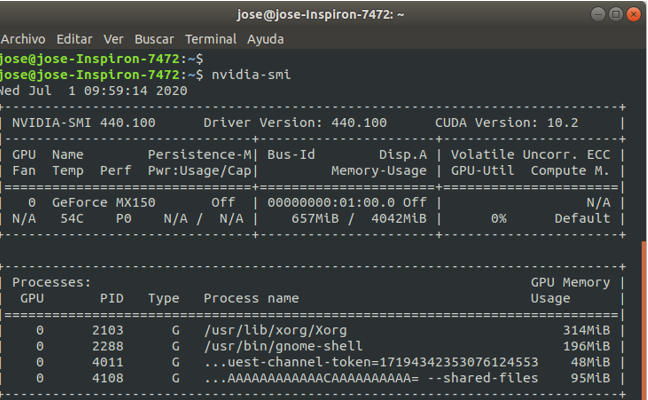


En este punto, es importante escoger el controlador que nos recomienda el Sistema Operativo.

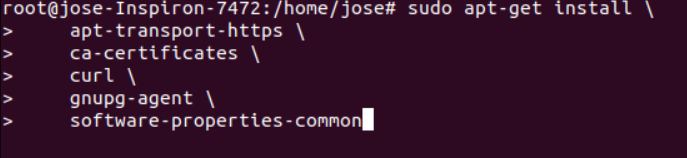
1. Realizar la instalación del controlador en el terminal.

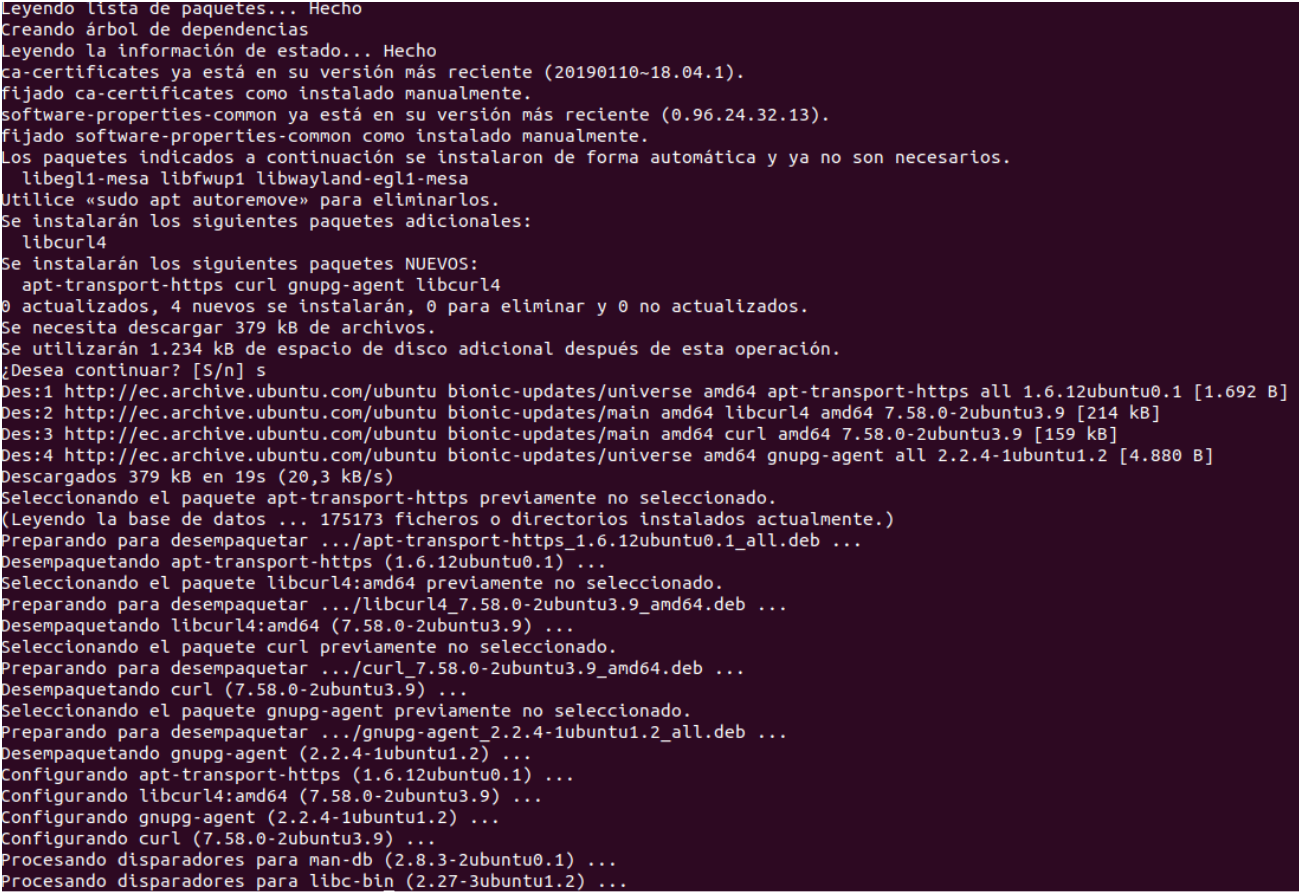


1. Una vez instalado, abrir nuevamente otro terminal y verificar que ya contamos con los drivers instalados, actualizados y funcionando de manera correcta.



1. Instalación de Docker en el computador personal, escribir en la ventana de comandos lo siguiente.

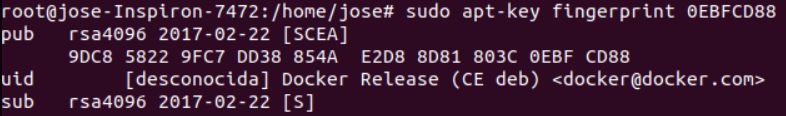




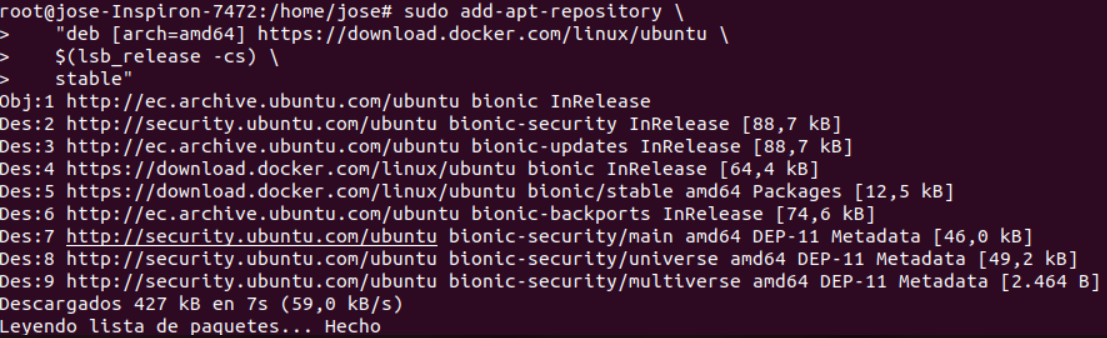
1. Agregar la clave *GPG* oficial de *Docker*.



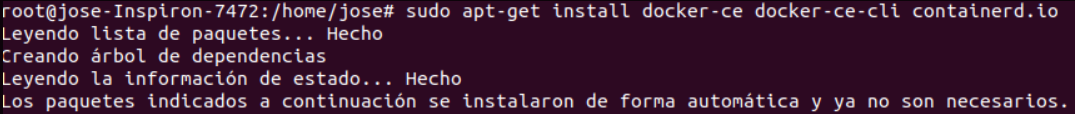
1. Verificar clave.



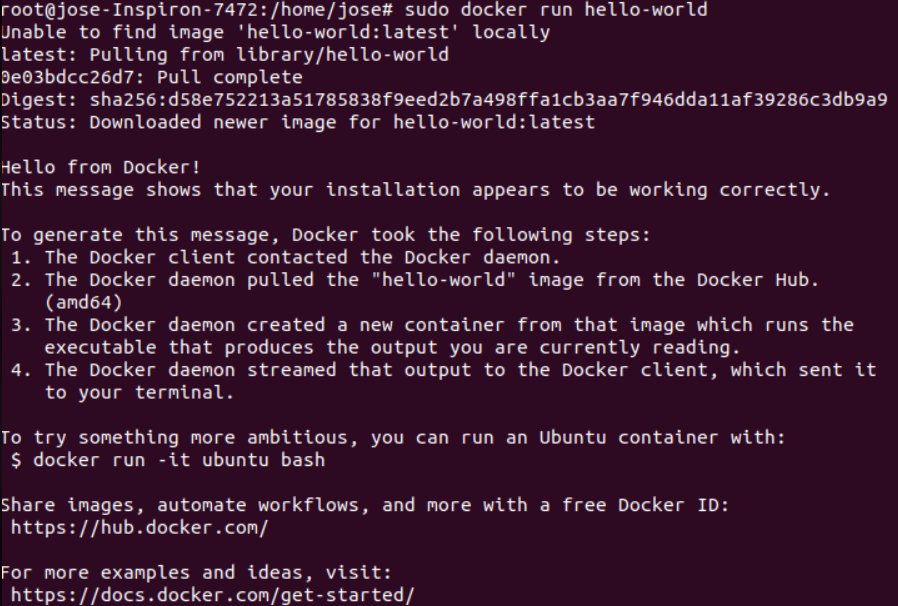
1. Configurar el repositorio a estable.



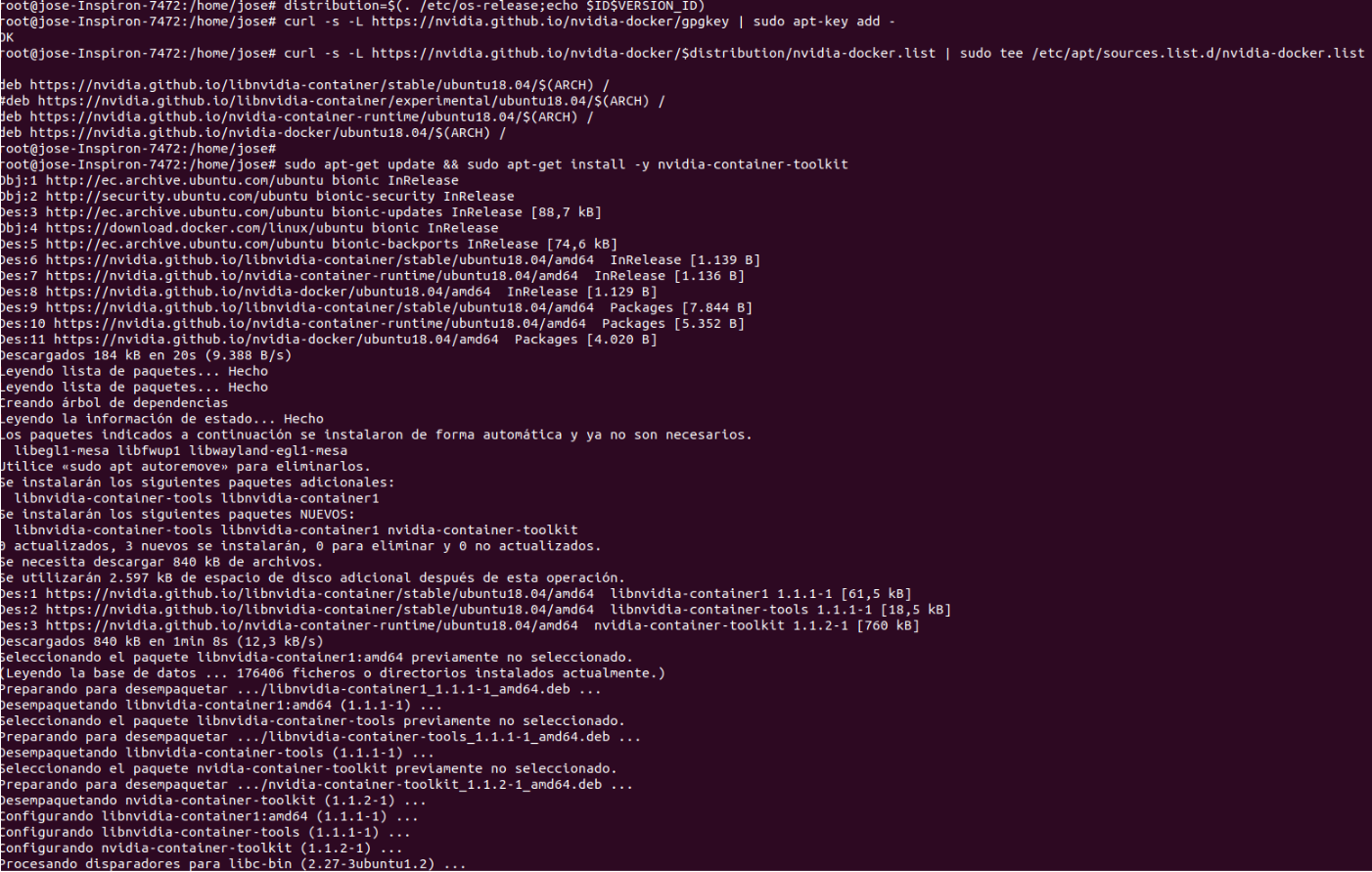
1. Instalar *Docker* en la máquina personal.



1. Realizar prueba de *Docker* en máquina personal.

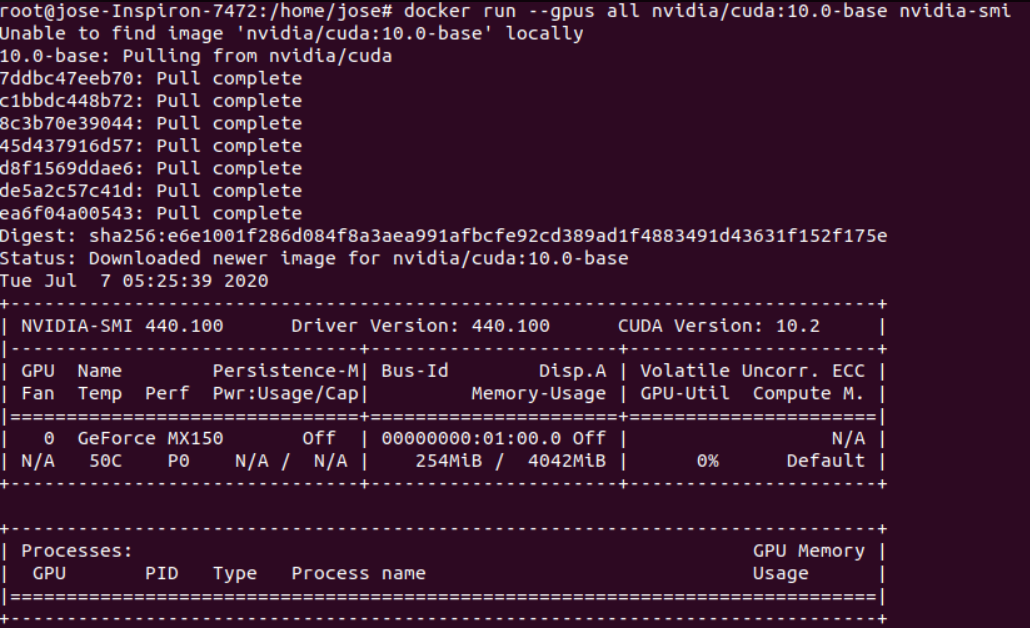


1. Crear y ejecutar contenedores *Docker* acelerados por *GPU*.

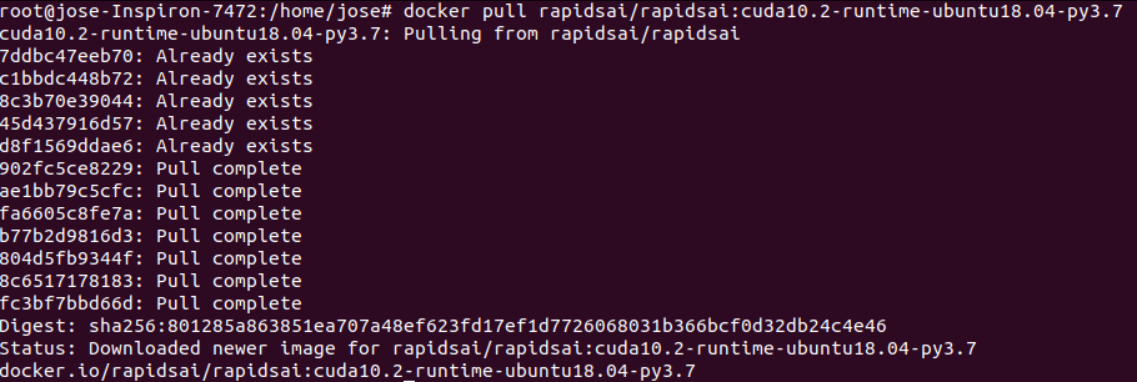




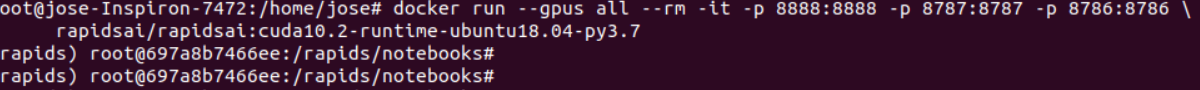
1. Prueba de *GPU* *Nvidia* en *Docker*.



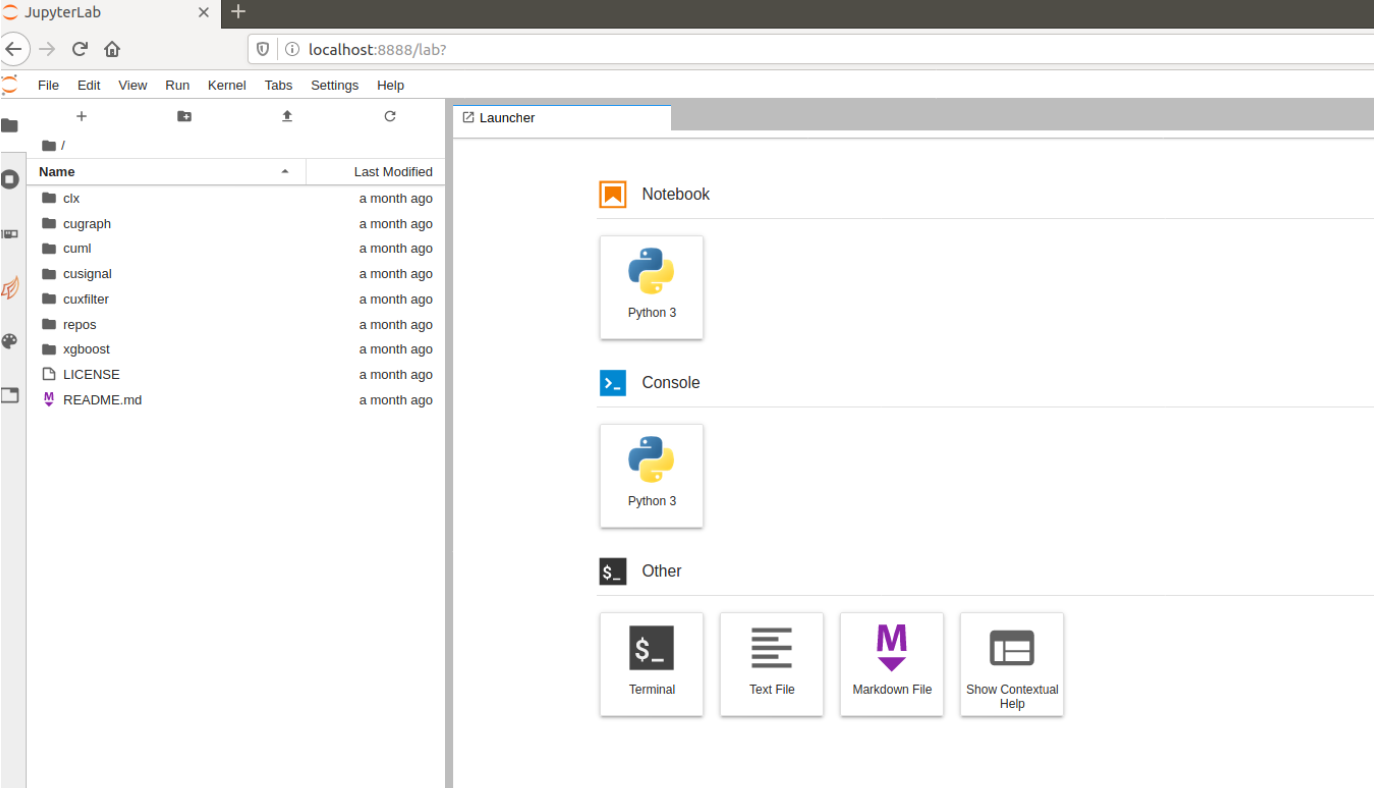
1. Instalación de la librería *RAPIDS* al contendor creado en la máquina personal.



1. Ejecutar *Docker*.

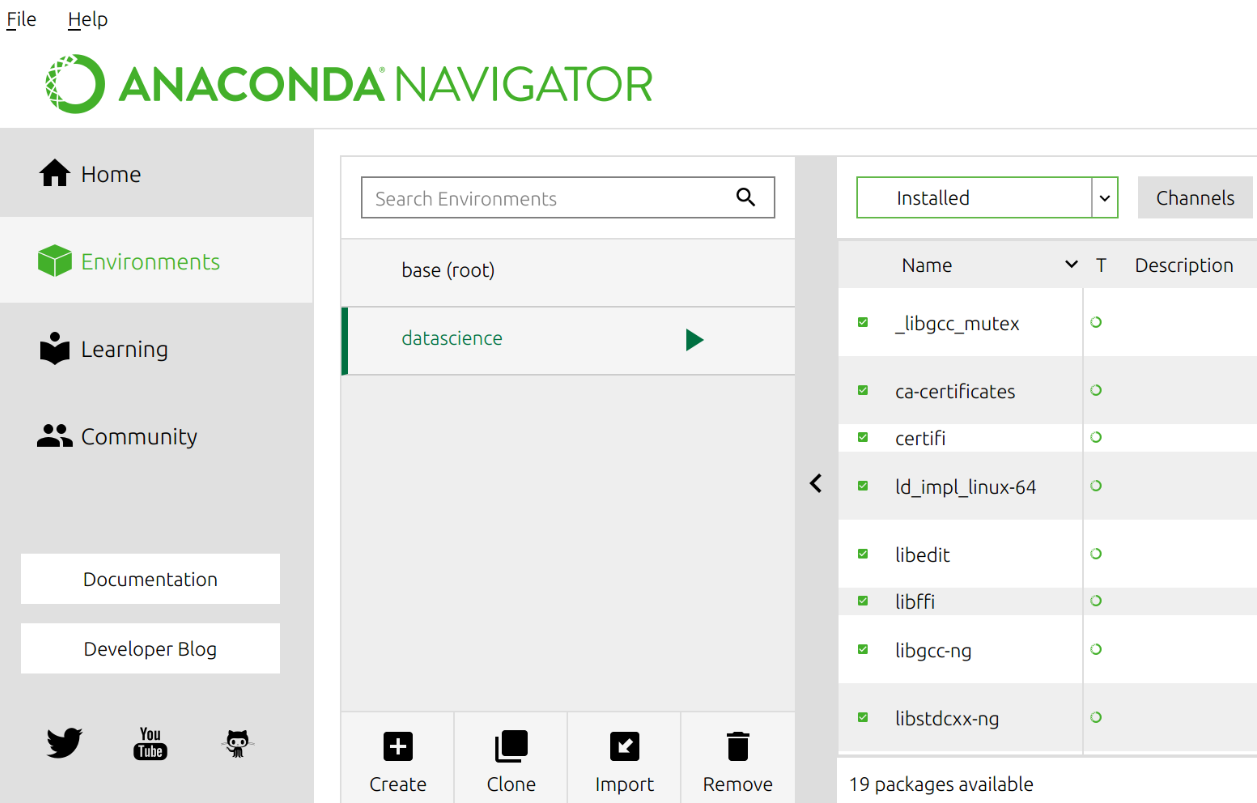


1. Plataforma de desarrollo para la *Data* *Science* con *Docker* y *RAPIDS*.

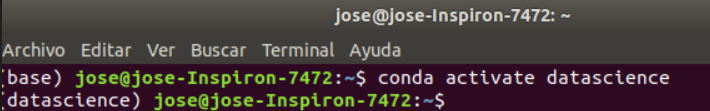


**Instalación e Implementación con CONDA en computador local**

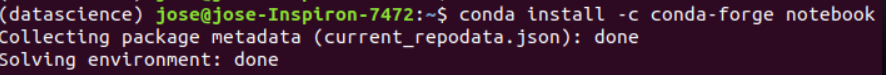
1. Crear entorno en la plataforma de *Conda*.

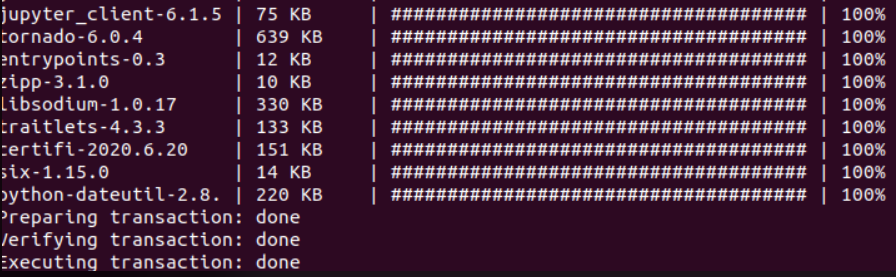


1. Activar entorno de trabajo por el terminal.

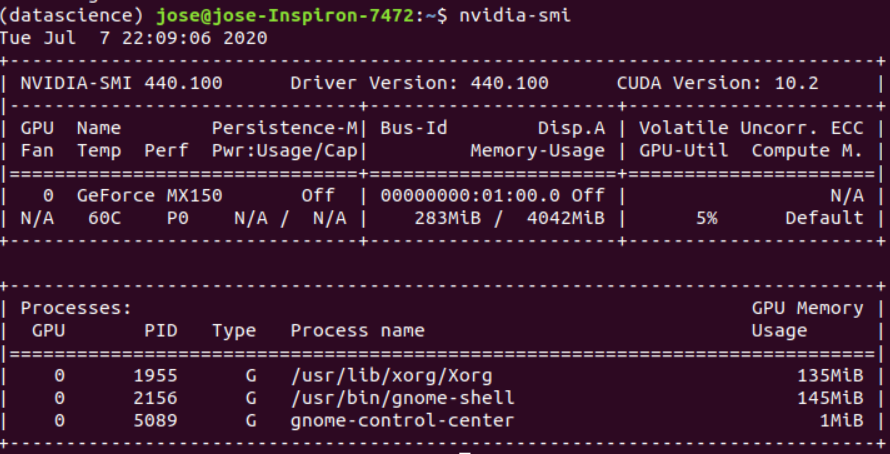


1. Instalar *Jupyter*.

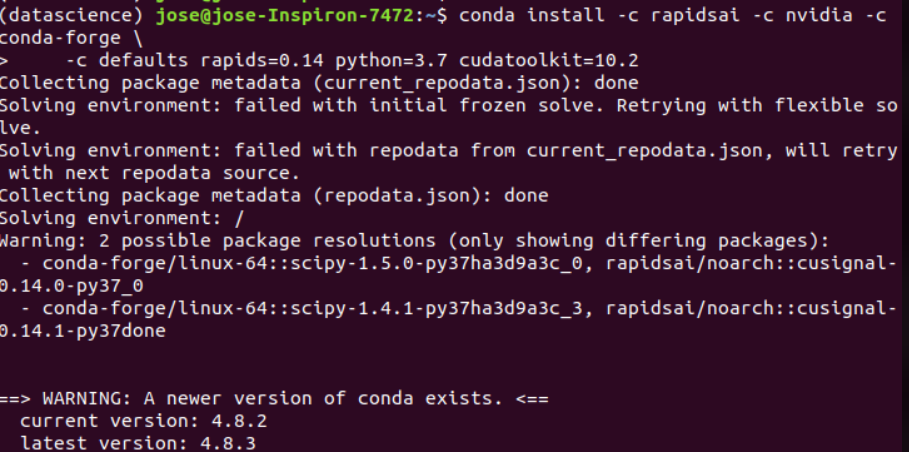


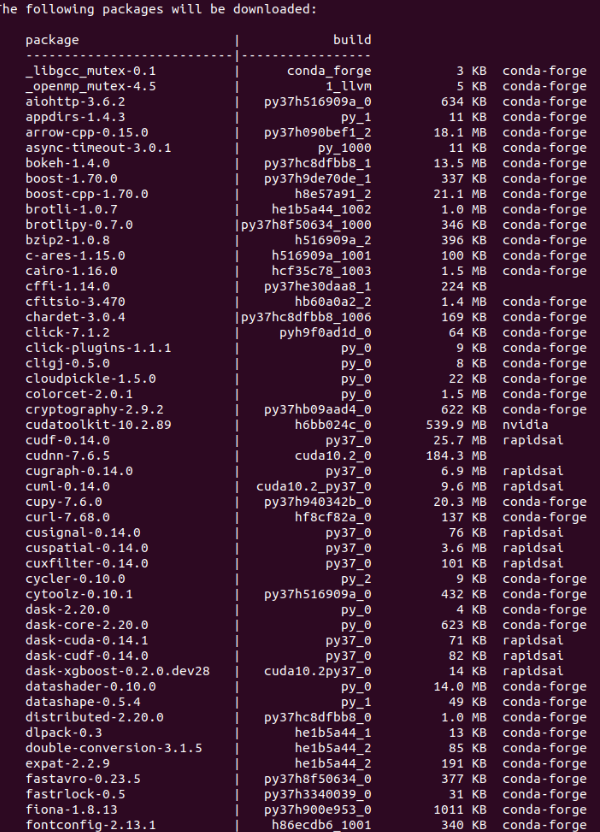


1. Prueba de *GPU* *Nvidia* en *Conda*.



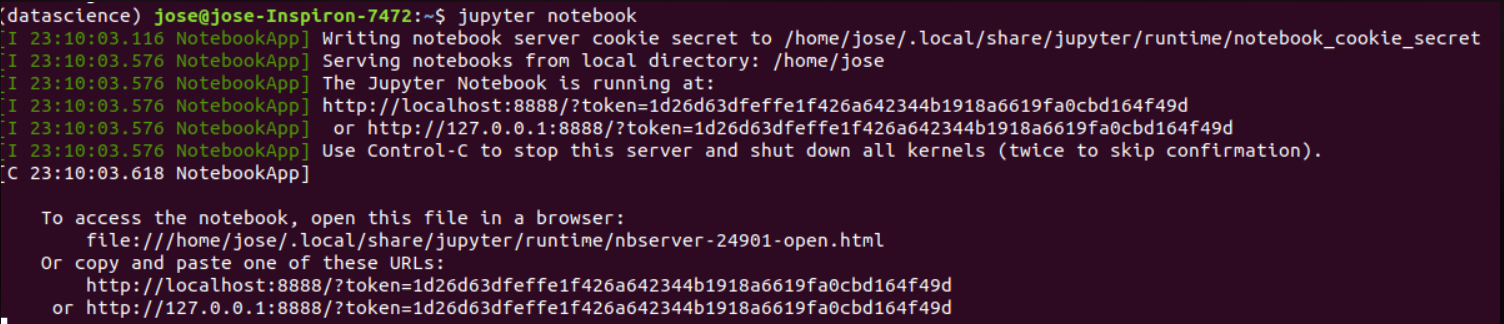
1. Instalación de la librería *RAPIDS.*



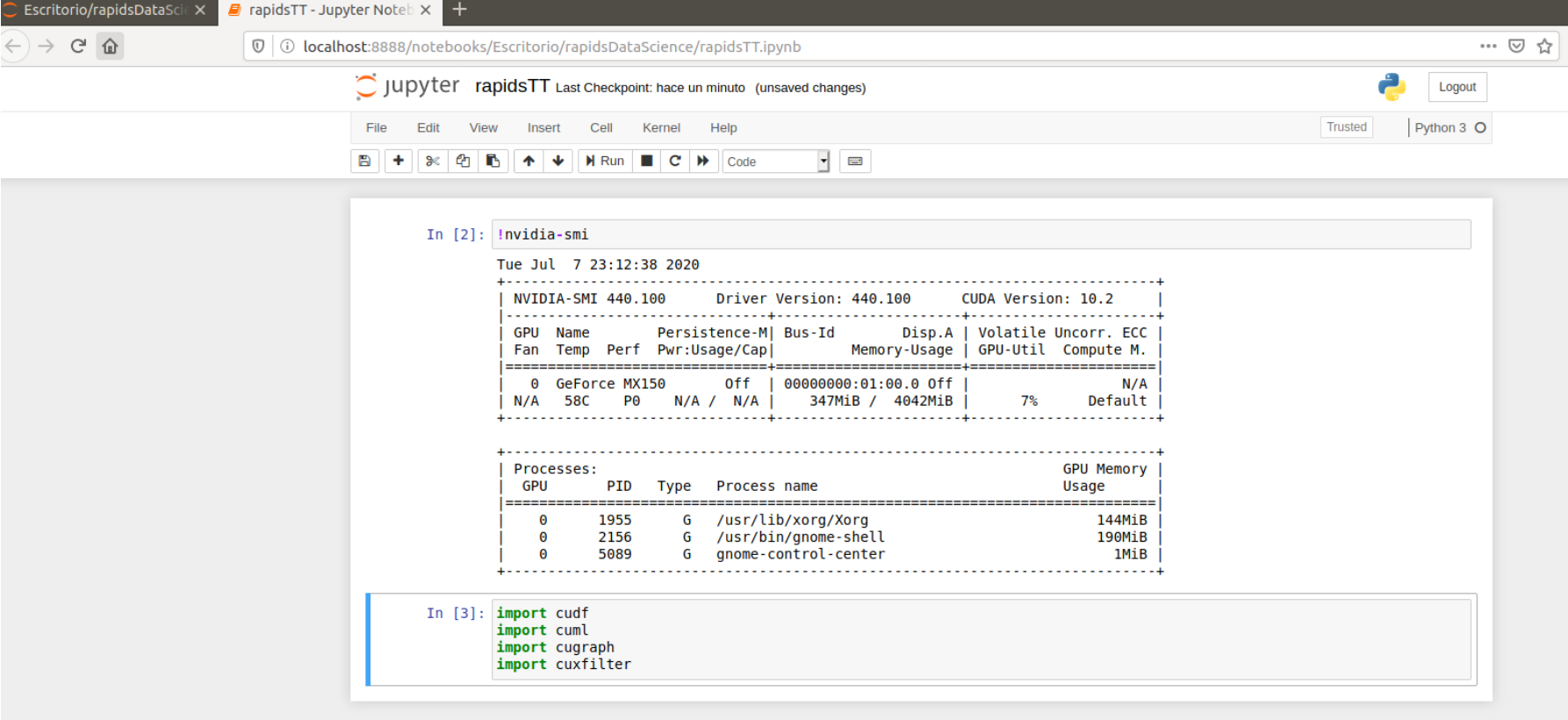




1. Activar y entrar al entorno Jupyter.



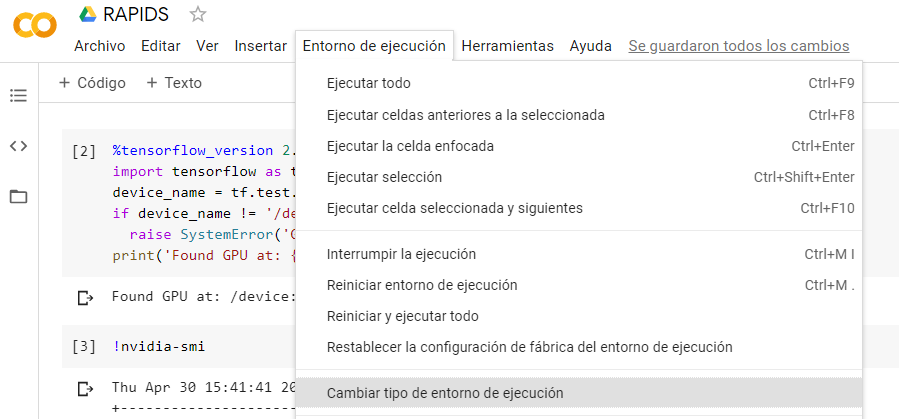
1. Plataforma de desarrollo *Jupyter* para la *Data* *Science* con *Conda* y *RAPIDS*.

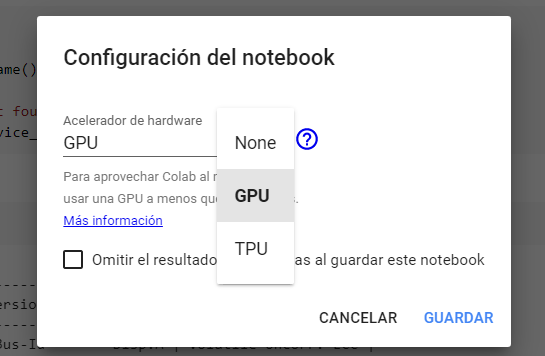


**Instalación e implementación Google Colabority**

1. Entorno de ejecución

Habilitar el entorno de ejecución en *colabority*, tiene que estar en *GPU* (Unidad de procesamiento gráfico).





1. Importar *tensorflow*

Debemos importar el *tensorflow* para verificar que estamos trabajando con GPU y no tener errores en eventos futuros.

Entrada:

%tensorflow\_version 2.x

import tensorflow as tf

device\_name = tf.test.gpu\_device\_name()

if device\_name != '/device:GPU:0':

  raise SystemError('GPU device not found')

print('Found GPU at: {}'.format(device\_name))

Salida:

Found GPU at: /device:GPU:0

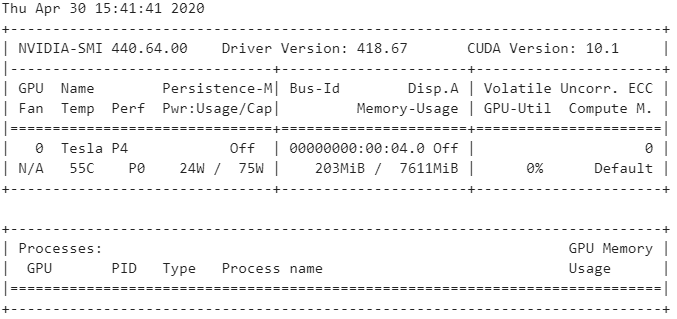
1. Instalación de RAPIDS:

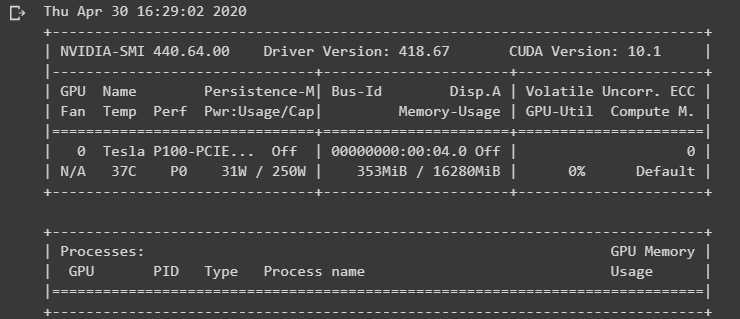
Verificamos con que GPU estamos trabajando, Google Colabority lanza por defecto las siguientes arquitecturas de GPU.

Entrada, código:

!nvidia-smi

Salida:





Tomar en cuenta para que RAPIDS funcione en Google Colabority debe trabajarse con las arquitecturas de GPU siguientes:

* Tesla T4
* Tesla P4
* Tesla P100

Instalación:

# Install RAPIDS

!git clone https://github.com/rapidsai/rapidsai-csp-utils.git

!bash rapidsai-csp-utils/colab/rapids-colab.sh

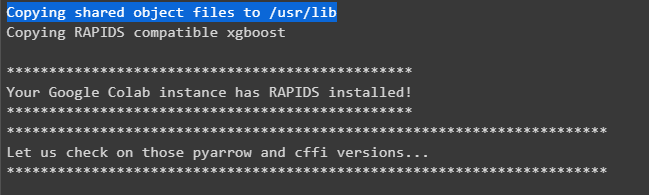
import sys, os

dist\_package\_index = sys.path.index('/usr/local/lib/python3.6/dist-packages')

sys.path = sys.path[:dist\_package\_index] + ['/usr/local/lib/python3.6/site-packages'] + sys.path[dist\_package\_index:]

sys.path

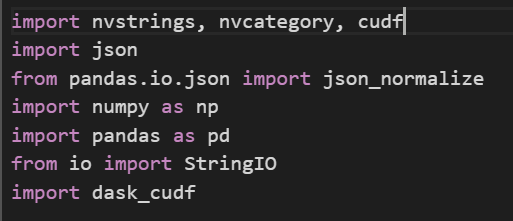
exec(open('rapidsai-csp-utils/colab/update\_modules.py').read(), globals())



Si la instalación no tuve errores, se ejecutó de manera correcta quiere decir que han seleccionado el paquete estable de RAPIDS que es la versión 0.13.

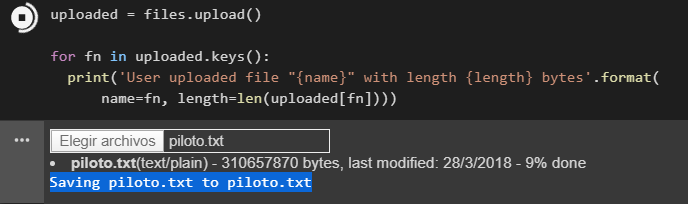
1. Importar librerías de RAPIDS:

**Tomar en cuenta:** Importar los paquetes de nvstrings y nvcategory antes de cuDF, si no se importa estos paquetes la biblioteca cuDF no funcionara, ejemplo:



1. Carga de Archivo a Google Colabority.

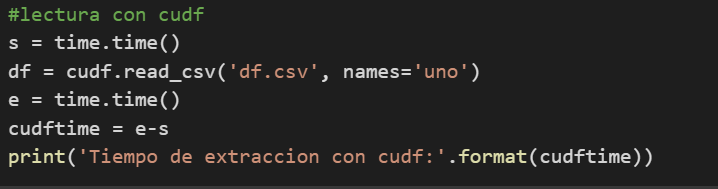
Cargar archivo.txt o cualquier otro archivo desde la memoria local del computador personal a Google Colabority, para después guardar el archivo en la memoria temporal de Colabority:



1. Trabajo con cuDF.

Dato: cuDF tiene la misma sintaxis que la librería PANDAS (se parece mucho, pero cuDF incorpora características muy importantes)

La siguiente figura corresponde a la lectura del archivo con las características de cuDF.

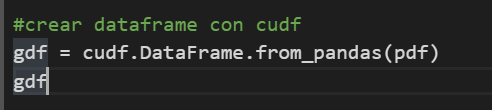


cudf.read\_csv: lee archivos CSV.

cudf.read\_json: lee archivos json.

cudf.read\_txt: lee archivos de texto.

1. Convertir un DataFrame.



cudf.DataFrame.from\_pandas: convierte un DataFrame de Pandas a un nuevo DataFrame cuDF.

cudf.DataFrame.from\_records: convierte un DataFrame de numpy a un nuevo DataFrame cuDF

1. DataFrame creado y funcionalidades con cuDF.

gdf.head(): imprime las 5 primeras filas del DataFrame

gdf.dtypes(): imprime el tipo de datos de cada columna del dataframe

gdf.columns(): imprime el nombre de las columnas del dataframe

gdf.iloc(): permite seleccionar las filas e imprime las filas seleccionadas del dataframe

gdf.ndim(): imprime la dimensión del dataframe

gdf.shape(): imprime la representación del dataframe

gdf.describe(): imprime una descripción del dataframe por columnas que tengan valores numéricos

gdf.drop(‘nombre\_columna’): elimina una columna

gdf.drop\_duplicates (‘nombre\_columna’): elimina duplicados de una columna

gdf.fillna(): rellena valores nulos

gdf.join(): permite unir DataFrames