

# Manual RAPIDS

## Instalación RAPIDS máquina personal.

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

<b>GPU NVIDIA</b>	<b>SO Ubuntu</b>	<b>Docker</b>	<b>CUDA and NVIDIA Drivers</b>
GeForce	Versión 16.04	Versión 19.03	Versión 10.0
Titan RTX	Versión 18.04		Versión 10.1
Tesla			Versión 10.2

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

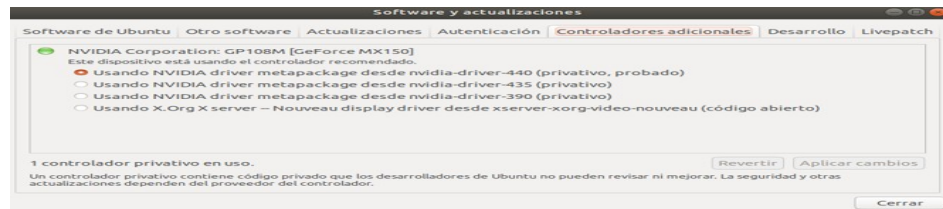
<b>Marca Portátil</b>	<b>Intel</b>	<b>Sistema Operativo</b>	<b>Tipo SO</b>	<b>GPU</b>
DELL	Core I7 8th Gen	Ubuntu 18.04	SO de 64 bits	NVIDIA GeForce MX150

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.



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

```
jose@jose-Inspiron-7472: ~
Archivo Editar Ver Buscar Terminal Ayuda
jose@jose-Inspiron-7472:~$
jose@jose-Inspiron-7472:~$
jose@jose-Inspiron-7472:~$ lspci | grep VGA
00:02.0 VGA compatible controller: Intel Corporation UHD Graphics 620 (rev 07)
jose@jose-Inspiron-7472:~$
```

3. 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.

```
jose@jose-Inspiron-7472:~$ ubuntu-drivers devices
== /sys/devices/pci0000:00/0000:00:1c.0/0000:01:00.0 ==
modalias : pci:v000010DEd00001D10sv00001028sd00000828bc03sc02i00
vendor    : NVIDIA Corporation
model     : GP108M [GeForce MX150]
driver    : nvidia-driver-390 - distro non-free
driver    : nvidia-driver-435 - distro non-free
driver    : nvidia-driver-440 - distro non-free recommended
driver    : xserver-xorg-video-nouveau - distro free builtin
```

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

4. Realizar la instalación del controlador en el terminal.

```
jose@jose-Inspiron-7472:~$ sudo apt install nvidia-driver-440
```

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

```
jose@jose-Inspiron-7472: ~
Archivo Editar Ver Buscar Terminal Ayuda
jose@jose-Inspiron-7472:~$
jose@jose-Inspiron-7472:~$ nvidia-smi
Wed Jul 1 09:59:14 2020
+-----+
| NVIDIA-SMI 440.100      Driver Version: 440.100      CUDA Version: 10.2      |
+-----+-----+
| GPU   Name               Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|  Memory-Usage | GPU-Util  Compute M. |
|-----+-----+
| 0     GeForce MX150      Off          | 00000000:01:00.0 Off |           | N/A |
| N/A   54C    P0       N/A /   N/A | 657MiB / 4042MiB | 0%      Default |
+-----+-----+
+-----+
| Processes:
| GPU   PID     Type    Process name                               GPU Memory |
|-----+-----+
| 0     2103    G       /usr/lib/xorg/Xorg                         314MiB |
| 0     2288    G       /usr/bin/gnome-shell                       196MiB |
| 0     4011    G       ..uest-channel-token=17194342353076124553  48MiB |
| 0     4108    G       ..AAAAAAAAAACAACAAAAAAAAAA= -shared-files  95MiB |
+-----+-----+
```

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

```

root@jose-Inspiron-7472:/home/jose# sudo apt-get install \
> apt-transport-https \
> ca-certificates \
> curl \
> gnupg-agent \
> software-properties-common

Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
ca-certificates ya está en su versión más reciente (20190110-18.04.1).
fijado ca-certificates como instalado manualmente.
software-properties-common ya está en su versión más reciente (0.96.24.32.13).
fijado software-properties-common como instalado manualmente.
Los paquetes indicados a continuación se instalaron de forma automática y ya no son necesarios.
  libegl1-mesa libfwup1 libwayland-egl1-mesa
Utilice «sudo apt autoremove» para eliminarlos.
Se instalarán los siguientes paquetes adicionales:
  libcurl4
Se instalarán los siguientes paquetes NUEVOS:
  apt-transport-https curl gnupg-agent libcurl4
0 actualizados, 4 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
Se necesita descargar 379 kB de archivos.
Se utilizarán 1.234 kB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] s
Des:1 http://ec.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 apt-transport-https all 1.6.12ubuntu0.1 [1.692 B]
Des:2 http://ec.archive.ubuntu.com/ubuntu bionic-updates/main amd64 libcurl4 amd64 7.58.0-2ubuntu3.9 [214 kB]
Des:3 http://ec.archive.ubuntu.com/ubuntu bionic-updates/main amd64 curl amd64 7.58.0-2ubuntu3.9 [159 kB]
Des:4 http://ec.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 gnupg-agent all 2.2.4-1ubuntu1.2 [4.880 B]
Descargados 379 kB en 19s (20,3 kB/s)
Seleccionando el paquete apt-transport-https previamente no seleccionado.
(Leyendo la base de datos ... 175173 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar .../apt-transport-https_1.6.12ubuntu0.1_all.deb ...
Desempaquetando apt-transport-https (1.6.12ubuntu0.1) ...
Seleccionando el paquete libcurl4:amd64 previamente no seleccionado.
Preparando para desempaquetar .../libcurl4_7.58.0-2ubuntu3.9_amd64.deb ...
Desempaquetando libcurl4:amd64 (7.58.0-2ubuntu3.9) ...
Seleccionando el paquete curl previamente no seleccionado.
Preparando para desempaquetar .../curl_7.58.0-2ubuntu3.9_amd64.deb ...
Desempaquetando curl (7.58.0-2ubuntu3.9) ...
Seleccionando el paquete gnupg-agent previamente no seleccionado.
Preparando para desempaquetar .../gnupg-agent_2.2.4-1ubuntu1.2_all.deb ...
Desempaquetando gnupg-agent (2.2.4-1ubuntu1.2) ...
Configurando apt-transport-https (1.6.12ubuntu0.1) ...
Configurando libcurl4:amd64 (7.58.0-2ubuntu3.9) ...
Configurando gnupg-agent (2.2.4-1ubuntu1.2) ...
Configurando curl (7.58.0-2ubuntu3.9) ...
Procesando disparadores para man-db (2.8.3-2ubuntu0.1) ...
Procesando disparadores para libc-bin (2.27-3ubuntu1.2) ...

```

## 7. Agregar la clave GPG oficial de Docker.

```

root@jose-Inspiron-7472:/home/jose# curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
OK

```

## 8. Verificar clave.

```

root@jose-Inspiron-7472:/home/jose# sudo apt-key fingerprint 0EBFCD88
pub  rsa4096 2017-02-22 [SCEA]
    9DC8 5822 9FC7 DD38 854A  E2D8 8D81 803C 0EBF CD88
uid  [desconocida] Docker Release (CE deb) <docker@docker.com>
sub  rsa4096 2017-02-22 [S]

```

## 9. Configurar el repositorio a estable.

```

root@jose-Inspiron-7472:/home/jose# sudo add-apt-repository \
> "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
> $(lsb_release -cs) \
> stable"
Obj:1 http://ec.archive.ubuntu.com/ubuntu bionic InRelease
Des:2 http://security.ubuntu.com/ubuntu bionic-security InRelease [88,7 kB]
Des:3 http://ec.archive.ubuntu.com/ubuntu bionic-updates InRelease [88,7 kB]
Des:4 https://download.docker.com/linux/ubuntu bionic InRelease [64,4 kB]
Des:5 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages [12,5 kB]
Des:6 http://ec.archive.ubuntu.com/ubuntu bionic-backports InRelease [74,6 kB]
Des:7 http://security.ubuntu.com/ubuntu bionic-security/main amd64 DEP-11 Metadata [46,0 kB]
Des:8 http://security.ubuntu.com/ubuntu bionic-security/universe amd64 DEP-11 Metadata [49,2 kB]
Des:9 http://security.ubuntu.com/ubuntu bionic-security/multiverse amd64 DEP-11 Metadata [2.464 B]
Descargados 427 kB en 7s (59,0 kB/s)
Leyendo lista de paquetes... Hecho

```

## 10. Instalar *Docker* en la máquina personal.

```

root@jose-Inspiron-7472:/home/jose# sudo apt-get install docker-ce docker-ce-cli containerd.io
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Los paquetes indicados a continuación se instalaron de forma automática y ya no son necesarios.

```

## 11. Realizar prueba de *Docker* en máquina personal.

```

root@jose-Inspiron-7472:/home/jose# sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
9e03bdc26d7: Pull complete
Digest: sha256:d58e75213a51785838f9eed2b7a498ffa1cb3aa7f946dda11af39286c3db9a9
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/

```

## 12. Crear y ejecutar contenedores *Docker* acelerados por *GPU*.

```

root@jose-Inspiron-7472:/home/jose# distribution=$(cat /etc/os-release | grep ^ID | cut -d= -f2)
root@jose-Inspiron-7472:/home/jose# curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | sudo apt-key add -
root@jose-Inspiron-7472:/home/jose# curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.list | sudo tee /etc/apt/sources.list.d/nvidia-docker.list
deb https://nvidia.github.io/libnvidia-container/stable/ubuntu18.04/$ARCH /
deb https://nvidia.github.io/libnvidia-container/experimental/ubuntu18.04/$ARCH /
deb https://nvidia.github.io/nvidia-container-runtime/ubuntu18.04/$ARCH /
deb https://nvidia.github.io/nvidia-docker/ubuntu18.04/$ARCH /
root@jose-Inspiron-7472:/home/jose#
root@jose-Inspiron-7472:/home/jose# sudo apt-get update && sudo apt-get install -y nvidia-container-toolkit
Obj:1 http://ec.archive.ubuntu.com/ubuntu bionic InRelease
Obj:2 http://security.ubuntu.com/ubuntu bionic-security InRelease
Des:3 http://ec.archive.ubuntu.com/ubuntu bionic-updates InRelease [88,7 kB]
Des:4 https://download.docker.com/linux/ubuntu bionic InRelease
Des:5 http://ec.archive.ubuntu.com/ubuntu bionic-backports InRelease [74,6 kB]
Des:6 https://nvidia.github.io/libnvidia-container/stable/ubuntu18.04/amd64 InRelease [1.139 B]
Des:7 https://nvidia.github.io/nvidia-container-runtime/ubuntu18.04/amd64 InRelease [1.136 B]
Des:8 https://nvidia.github.io/nvidia-docker/ubuntu18.04/amd64 InRelease [1.129 B]
Des:9 https://nvidia.github.io/libnvidia-container/stable/ubuntu18.04/amd64 Packages [7.844 B]
Des:10 https://nvidia.github.io/nvidia-container-runtime/ubuntu18.04/amd64 Packages [5.352 B]
Des:11 https://nvidia.github.io/nvidia-docker/ubuntu18.04/amd64 Packages [4.020 B]
Descargados 184 kB en 20s (9.388 B/s)
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Los paquetes indicados a continuación se instalaron de forma automática y ya no son necesarios.
libegl-mesa libfhwapi libwayland-egl-mesa
utils «sudo apt autoremove» para eliminarlos.
Se instalarán los siguientes paquetes adicionales:
  libnvidia-container-tools libnvidia-container1
Se instalarán los siguientes paquetes NUEVOS:
  libnvidia-container-tools libnvidia-container1 nvidia-container-toolkit
Se actualizarán, 3 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
Se necesita descargar 840 kB de archivos.
Se utilizarán 2.597 kB de espacio de disco adicional después de esta operación.
Des:1 https://nvidia.github.io/libnvidia-container/stable/ubuntu18.04/amd64 libnvidia-container1 1.1.1-1 [61,5 kB]
Des:2 https://nvidia.github.io/libnvidia-container/stable/ubuntu18.04/amd64 libnvidia-container-tools 1.1.1-1 [18,5 kB]
Des:3 https://nvidia.github.io/nvidia-docker/ubuntu18.04/amd64 nvidia-container-toolkit 1.1.2-1 [760 kB]
Descargados 840 kB en 10s (81,3 kB/s)
Seleccionando el paquete libnvidia-container1:amd64 previamente no seleccionado.
Leyendo la base de datos ... 176406 ficheros o directorios instalados actualmente.
Preparando para desempaquetar .../libnvidia-container1_1.1.1-1_amd64.deb ...
Desempaquetando libnvidia-container1:amd64 (1.1.1-1) ...
Seleccionando el paquete libnvidia-container-tools previamente no seleccionado.
Preparando para desempaquetar .../libnvidia-container-tools_1.1.1-1_amd64.deb ...
Desempaquetando libnvidia-container-tools (1.1.1-1) ...
Seleccionando el paquete nvidia-container-toolkit previamente no seleccionado.
Preparando para desempaquetar .../nvidia-container-toolkit_1.1.2-1_amd64.deb ...
Desempaquetando nvidia-container-toolkit (1.1.2-1) ...
Configurando libnvidia-container1:amd64 (1.1.1-1) ...
Configurando libnvidia-container-tools (1.1.1-1) ...
Configurando nvidia-container-toolkit (1.1.2-1) ...
Procesando disparadores para libc-bin (2.27-3ubuntu2) ...

```



```
root@jose-Inspiron-7472:/home/jose# sudo systemctl restart docker
```

### 13. Prueba de *GPU Nvidia* en *Docker*.

```
root@jose-Inspiron-7472:/home/jose# docker run --gpus all nvidia/cuda:10.0-base nvidia-smi
Unable to find image 'nvidia/cuda:10.0-base' locally
10.0-base: Pulling from nvidia/cuda
7ddb47eeb70: Pull complete
c1bbdc448b72: Pull complete
8c3b70e39044: Pull complete
45d437916d57: Pull complete
d8f1569dda6: Pull complete
de5a2c57c41d: Pull complete
ea6f04a00543: Pull complete
Digest: sha256:6e0e1001f280d084f8a3aea991afbcfe92cd389ad1f4883491d43631f152f175e
Status: Downloaded newer image for nvidia/cuda:10.0-base
Tue Jul 7 05:25:39 2020
+-----+
| NVIDIA-SMI 440.100          Driver Version: 440.100          CUDA Version: 10.2          |
| GPU Name                     Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf    Pwr:Usage/Cap|  Memory-Usage | GPU-Util  Compute M. |
|-----+-----+-----+-----+-----+-----+-----+
| 0       30C    P0      N/A /  N/A | 254MiB / 4042MiB |  0%      Default     |
+-----+-----+-----+-----+-----+-----+-----+
| Processes:
| GPU      PID   Type   Process name                      GPU Memory
|-----+-----+-----+-----+-----+-----+

```

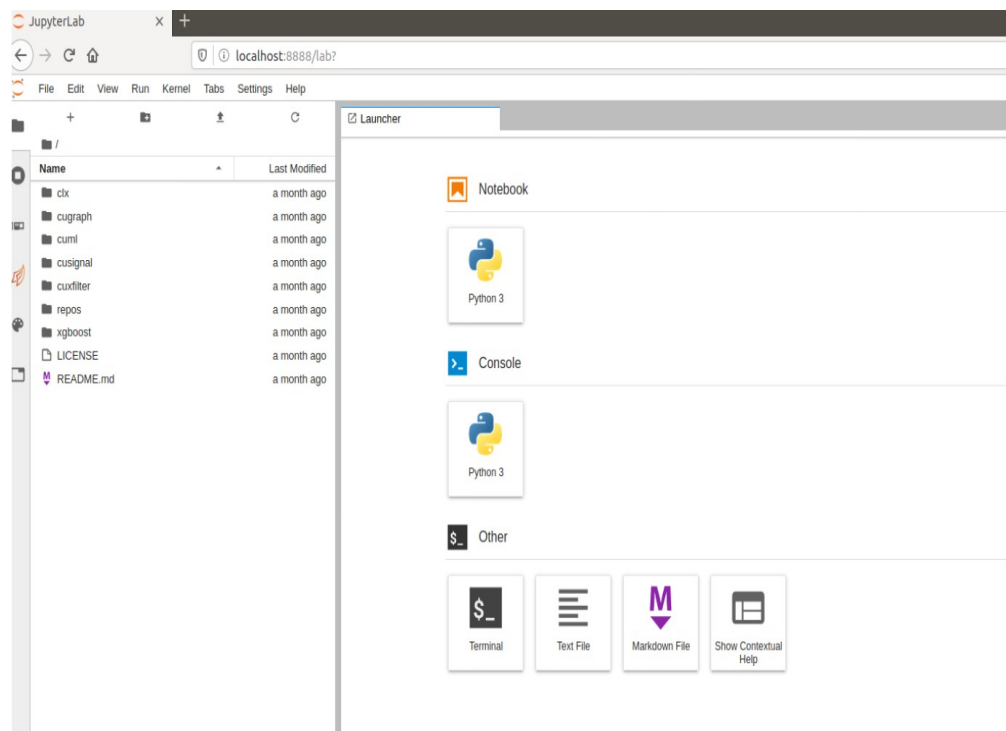
### 14. Instalación de la librería *RAPIDS* al contenedor creado en la máquina personal.

```
root@jose-Inspiron-7472:/home/jose# docker pull rapidsai/rapidsai:cuda10.2-runtime-ubuntu18.04-py3.7
cuda10.2-runtime-ubuntu18.04-py3.7: Pulling from rapidsai/rapidsai
7ddb47eeb70: Already exists
c1bbdc448b72: Already exists
8c3b70e39044: Already exists
45d437916d57: Already exists
d8f1569dda6: Already exists
902fc5ce8229: Pull complete
ae1bb79c5cfc: Pull complete
fa6605c8fe7a: Pull complete
b77b2d9816d3: Pull complete
804d5fb9344f: Pull complete
8c6517178183: Pull complete
fc3bf7bbd66d: Pull complete
Digest: sha256:801285a863851ea707a48ef623fd17ef1d7726068031b366bcf0d32db24c4e46
Status: Downloaded newer image for rapidsai/rapidsai:cuda10.2-runtime-ubuntu18.04-py3.7
docker.io/rapidsai/rapidsai:cuda10.2-runtime-ubuntu18.04-py3.7
```

### 15. Ejecutar *Docker*.

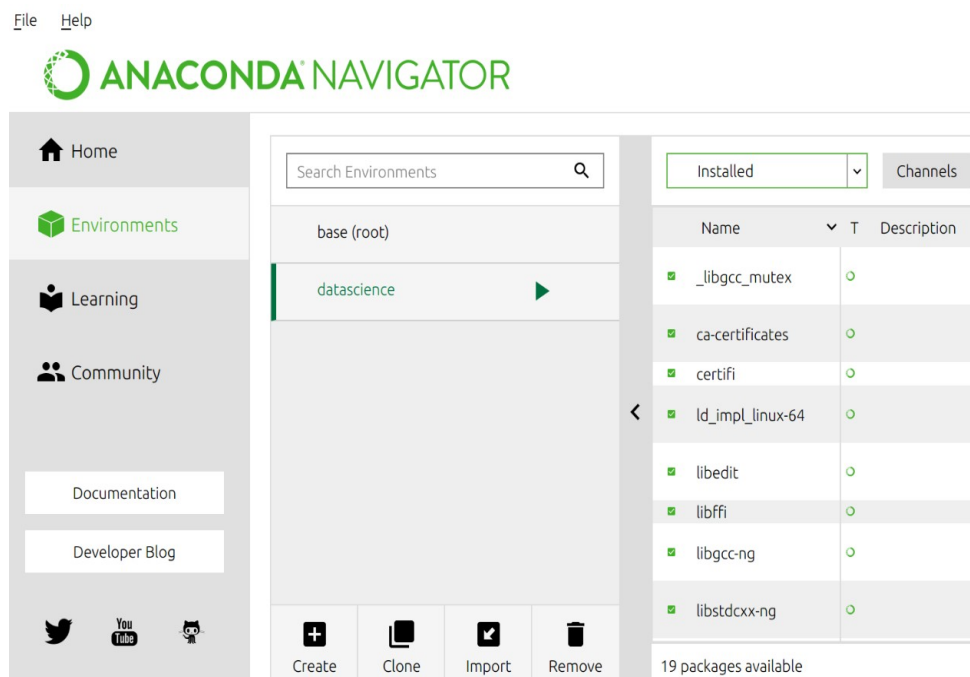
```
root@jose-Inspiron-7472:/home/jose# docker run --gpus all --rm -it -p 8888:8888 -p 8787:8787 -p 8786:8786 \
    rapidsai/rapidsai:cuda10.2-runtime-ubuntu18.04-py3.7
(rapids) root@697a8b7466ee:/rapids/notebooks#
(rapids) root@697a8b7466ee:/rapids/notebooks#
```

### 16. 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.



2. Activar entorno de trabajo por el terminal.

```
jose@jose-Inspiron-7472: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
(base) jose@jose-Inspiron-7472:~$ conda activate datascience  
(datascience) jose@jose-Inspiron-7472:~$
```

3. Instalar Jupyter.

```
(datascience) jose@jose-Inspiron-7472:~$ conda install -c conda-forge notebook  
Collecting package metadata (current_repodata.json): done  
Solving environment: done
```

```

jupyter_client-6.1.5 | 75 KB | ##### | 100%
tornado-6.0.4 | 639 KB | ##### | 100%
entrypoints-0.3 | 12 KB | ##### | 100%
zipp-3.1.0 | 10 KB | ##### | 100%
libsodium-1.0.17 | 330 KB | ##### | 100%
traitlets-4.3.3 | 133 KB | ##### | 100%
certifi-2020.6.20 | 151 KB | ##### | 100%
six-1.15.0 | 14 KB | ##### | 100%
python-dateutil-2.8. | 220 KB | ##### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done

```

#### 4. Prueba de GPU Nvidia en Conda.

```

(datascience) jose@jose-Inspiron-7472:~$ nvidia-smi
Tue Jul 7 22:09:06 2020

+-----+
| NVIDIA-SMI 440.100      Driver Version: 440.100      CUDA Version: 10.2      |
+-----+-----+
| GPU Name          Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
|=====+=====+
| 0  GeForce MX150      Off   | 00000000:01:00:0 |      N/A       |
| N/A   60C    P0      N/A /  N/A | 283MiB / 4042MiB |      5%      Default  |
+-----+-----+

+-----+
| Processes:                                                       GPU Memory |
|  GPU       PID    Type    Process name                     Usage    |
|=====+=====+
| 0          1955    G       /usr/lib/xorg/Xorg                   135MiB |
| 0          2156    G       /usr/bin/gnome-shell                 145MiB |
| 0          5089    G       gnome-control-center                  1MiB   |
+-----+

```

#### 5. Instalación de la librería RAPIDS.

```

(datascience) jose@jose-Inspiron-7472:~$ conda install -c rapidsai -c nvidia -c
conda-forge \
> -c defaults rapids=0.14 python=3.7 cudatoolkit=10.2
Collecting package metadata (current_repodata.json): done
Solving environment: failed with initial frozen solve. Retrying with flexible so
lve.
Solving environment: failed with repodata from current_repodata.json, will retry
with next repodata source.
Collecting package metadata (repodata.json): done
Solving environment: /
Warning: 2 possible package resolutions (only showing differing packages):
- conda-forge/linux-64::scipy-1.5.0-py37ha3d9a3c_0, rapidsai/noarch::cusignal-
0.14.0-py37_0
- conda-forge/linux-64::scipy-1.4.1-py37ha3d9a3c_3, rapidsai/noarch::cusignal-
0.14.1-py37done

==> WARNING: A newer version of conda exists. <==
current version: 4.8.2
latest version: 4.8.3

```



```

the following packages will be downloaded:

```

package	build	size	channel
_libgcc_mutex-0.1	conda_forge	3 KB	conda-forge
_openmp_mutex-4.5	1_llvm	5 KB	conda-forge
aiohttp-3.6.2	py37h516909a_0	634 KB	conda-forge
appdirs-1.4.3	py_1	11 KB	conda-forge
arrow-cpp-0.15.0	py37h090bce1_2	18.1 MB	conda-forge
async-timeout-3.0.1	py_1000	11 KB	conda-forge
bokeh-1.4.0	py37hcd8fbb8_1	13.5 MB	conda-forge
boost-1.70.0	py37h9de70de_1	337 KB	conda-forge
boost-cpp-1.70.0	h8e57a91_2	21.1 MB	conda-forge
brotlit-1.0.7	he1b5a44_1002	1.0 MB	conda-forge
brctlpy-0.7.0	py37h8f50634_1000	346 KB	conda-forge
bzip2-1.0.8	h516909a_2	396 KB	conda-forge
cares-1.15.0	h516909a_1001	180 KB	conda-forge
calro-1.16.0	hcf35c78_1003	1.5 MB	conda-forge
cffl-1.14.0	py37he30daa8_1	224 KB	
cfitsio-3.470	hb60a0a2_2	1.4 MB	
chardet-3.0.4	py37hcd8fbb8_1006	169 KB	conda-forge
click-7.1.2	pyh9f0ad10_0	64 KB	conda-forge
click-plugins-1.1.1	py_0	9 KB	conda-forge
clickj-0.5.0	py_0	8 KB	conda-forge
cloudpickle-1.5.0	py_0	22 KB	conda-forge
colorcet-2.0.1	py_0	1.5 MB	conda-forge
cryptography-2.9.2	py37hb09aad4_0	622 KB	conda-forge
cudatoolkit-10.2.89	h0bb924c_0	539.9 MB	nvidia
cudf-0.14.0	py37_0	25.7 MB	rapidsai
cudnn-7.6.5	cuda10.2_0	184.3 MB	
cugraph-0.14.0	py37_0	6.0 MB	rapidsai
cuml-0.14.0	cuda10.2_py37_0	9.6 MB	rapidsai
cupy-7.0.0	py37h940342b_0	20.3 MB	conda-forge
curl-7.68.0	hf8cf82a_0	137 KB	conda-forge
cusignal-0.14.0	py37_0	76 KB	rapidsai
cuspatial-0.14.0	py37_0	3.6 MB	rapidsai
cuxfilter-0.14.0	py37_0	101 KB	rapidsai
cycler-0.10.0	py_2	5 KB	conda-forge
cytoolz-0.10.1	py37h516909a_0	432 KB	conda-forge
dask-2.20.0	py_0	4 KB	conda-forge
dask-core-2.20.0	py_0	623 KB	conda-forge
dask-cuda-0.14.1	py37_0	71 KB	rapidsai
dask-cudf-0.14.0	py_0	82 KB	rapidsai
dask-xgboost-0.2.0.dev28	cuda10.2py37_0	14 KB	rapidsai
datashader-0.10.0	py_0	14.0 MB	conda-forge
datashape-0.5.4	py_1	49 KB	conda-forge
distributed-2.20.0	py37hcd8fbb8_0	1.0 MB	conda-forge
dipack-0.3	he1b5a44_1	13 KB	conda-forge
double-conversion-3.1.5	he1b5a44_2	85 KB	conda-forge
expat-2.2.9	he1b5a44_2	191 KB	conda-forge
fastavro-0.23.5	py37h8f50634_0	377 KB	conda-forge
fastrlock-0.5	py37h3140039_0	31 KB	conda-forge
fiona-1.8.13	py37h900e953_0	1011 KB	conda-forge
fontconfig-2.13.1	h80acdb6_1001	340 KB	conda-forge
libcuspatal-0.14.0		3.4 MB	100%
xorg-xproto-7.0.31		72 KB	100%
thrift-cpp-0.12.0		2.4 MB	100%
chardet-3.0.4		169 KB	100%
rb-2018.0.5		1.1 MB	100%
libgdal-3.0.2		18.7 MB	100%
libxml2-2.9.10		1.3 MB	100%
joblib-0.16.0		203 KB	100%

```

Preparing transaction: done
Verifying transaction: done
Executing transaction: done

```

## 6. Activar y entrar al entorno Jupyter.

```

datascience) jose@jose-Inspiron-7472:~$ jupyter notebook
[ 23:10:03.116 NotebookApp] Writing notebook server cookie secret to /home/jose/.local/share/jupyter/runtime/notebook_cookie_secret
[ 23:10:03.576 NotebookApp] Serving notebooks from local directory: /home/jose
[ 23:10:03.576 NotebookApp] The Jupyter Notebook is running at:
[ 23:10:03.576 NotebookApp] http://localhost:8888/?token=1d26d63dfeffe1f426a642344b1918a6619fa0cbd164f49d
[ 23:10:03.576 NotebookApp] or http://127.0.0.1:8888/?token=1d26d63dfeffe1f426a642344b1918a6619fa0cbd164f49d
[ 23:10:03.576 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
C 23:10:03.618 NotebookApp]

To access the notebook, open this file in a browser:
    file:///home/jose/.local/share/jupyter/runtime/nbsrvr-24901-open.html
Or copy and paste one of these URLs:
    http://localhost:8888/?token=1d26d63dfeffe1f426a642344b1918a6619fa0cbd164f49d
    or http://127.0.0.1:8888/?token=1d26d63dfeffe1f426a642344b1918a6619fa0cbd164f49d

```

## 7. Plataforma de desarrollo Jupyter para la Data Science con Conda y RAPIDS.



The screenshot shows a Jupyter Notebook window titled 'rapidsTT - Jupyter Noteb...'. The browser address bar shows 'localhost:8888/notebooks/Escritorio/rapidsDataScience/rapidsTT.ipynb'. The Jupyter interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Help) and a toolbar with icons for file operations, running, and code execution. The notebook content shows two input cells:

```
In [2]: !nvidia-smi
```

The output of the first cell is a terminal window showing the output of the `nvidia-smi` command. It displays system information for NVIDIA-SMI 440.100, Driver Version: 440.100, and CUDA Version: 10.2. It also lists GPU details for a GeForce MX150, including persistence mode, bus ID, display mode, memory usage, and temperature. Below this, it shows processes running on the GPU, including Xorg, gnome-shell, and gnome-control-center.

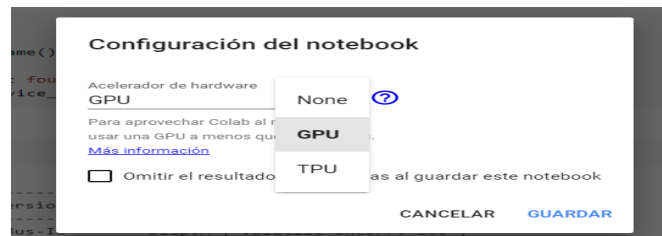
```
In [3]: import cudf
import cuml
import cudf
import cuxfilter
```

# 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).





## 2. 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

## 3. Instalación de RAPIDS:

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

Entrada, código:

```
!nvidia-smi
```

Salida:

```
Thu Apr 30 15:41:41 2020
+-----+
| NVIDIA-SMI 440.64.00      Driver Version: 418.67      CUDA Version: 10.1      |
+-----+-----+-----+-----+-----+-----+
| GPU  Name           Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf    Pwr:Usage/Cap|  Memory-Usage | GPU-Util  Compute M. |
+-----+-----+-----+-----+-----+-----+
| 0    Tesla P4             Off          | 00000000:00:04:0 Off |    0
| N/A   55C    P0      24W / 75W    | 203MiB / 7611MiB |      0%      Default |
+-----+-----+-----+-----+-----+

+-----+
| Processes:
| GPU      PID   Type   Process name                      GPU Memory
|                               Usage
+-----+-----+-----+-----+-----+
|
+-----+

Thu Apr 30 16:29:02 2020
+-----+
| NVIDIA-SMI 440.64.00      Driver Version: 418.67      CUDA Version: 10.1      |
+-----+-----+-----+-----+-----+-----+
| GPU  Name           Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf    Pwr:Usage/Cap|  Memory-Usage | GPU-Util  Compute M. |
+-----+-----+-----+-----+-----+-----+
| 0    Tesla P100-PCIE... Off          | 00000000:00:04:0 Off |    0
| N/A   37C    P0      31W / 250W    | 353MiB / 16280MiB |      0%      Default |
+-----+-----+-----+-----+-----+

+-----+
| Processes:
| GPU      PID   Type   Process name                      GPU Memory
|                               Usage
+-----+-----+-----+-----+-----+
|
+-----+
```

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

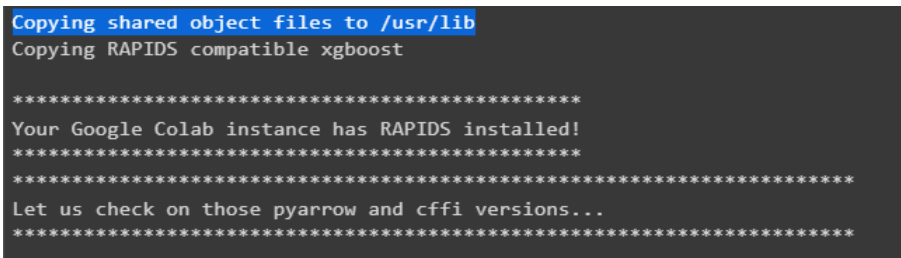
- Tesla T4
- Tesla P4
- Tesla P100

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())
```



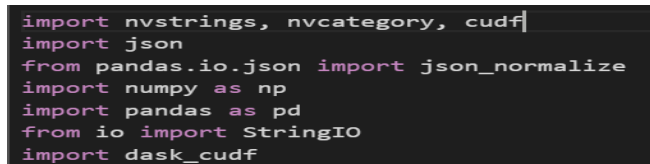
```
Copying shared object files to /usr/lib
Copying RAPIDS compatible xgboost

*****
Your Google Colab instance has RAPIDS installed!
*****
*****
Let us check on those pyarrow and cffi versions...
*****
```

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

#### 4. 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:



```
import nvstrings, nvcategory, cudf
import json
from pandas.io.json import json_normalize
import numpy as np
import pandas as pd
from io import StringIO
import dask_cudf
```

#### 5. 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:

```
uploaded = files.upload()

for fn in uploaded.keys():
    print('User uploaded file "{name}" with length {length} bytes'.format(
        name=fn, length=len(uploaded[fn])))

... Elegir archivos piloto.txt
• piloto.txt(text/plain) - 310657870 bytes, last modified: 28/3/2018 - 9% done
Saving piloto.txt to piloto.txt
```

## 6. 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.

```
#lectura con cudf
s = time.time()
df = cudf.read_csv('df.csv', names='uno')
e = time.time()
cudftime = e-s
print('Tiempo de extraccion con cudf:'.format(cudftime))
```

cudf.read\_csv: lee archivos CSV.

cudf.read\_json: lee archivos json.

cudf.read\_txt: lee archivos de texto.

## 7. Convertir un DataFrame.

```
#crear dataframe con cudf
gdf = cudf.DataFrame.from_pandas(pdf)
gdf
```

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

## 8. 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