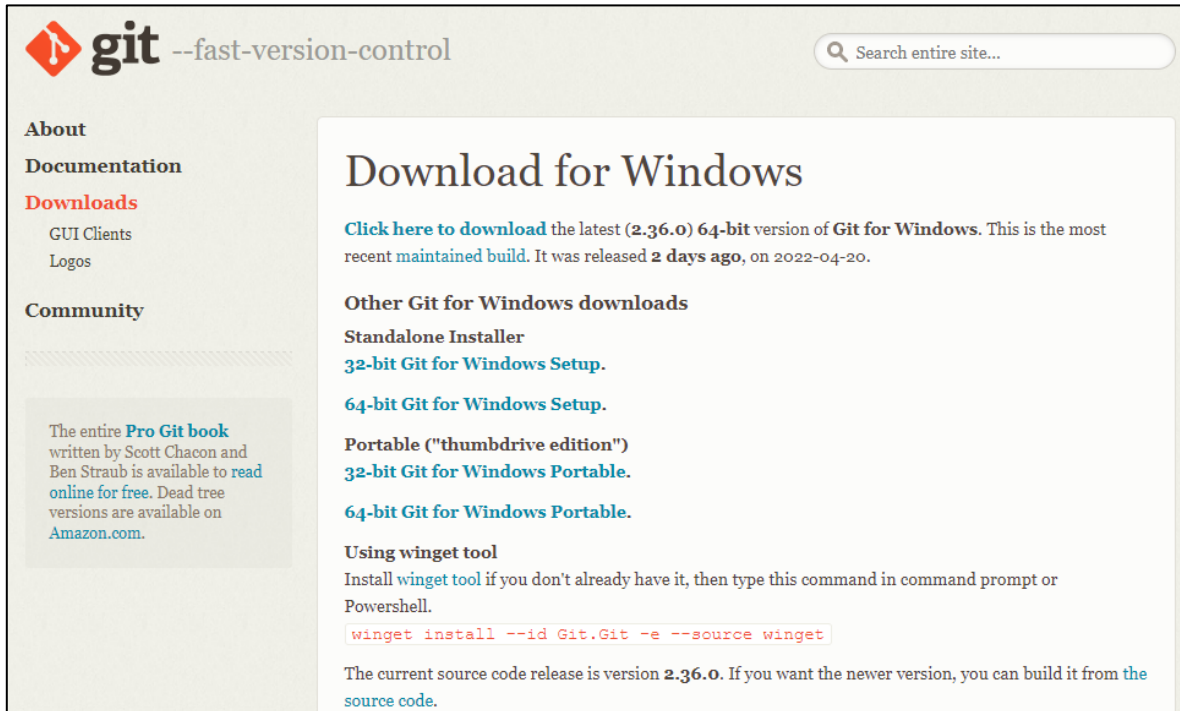


INSTALACIÓN DE DOCKER Y UBUNTU

1) Instalación de GIT

Puedes descargarlo de la página oficial de Git



The screenshot shows the Git website's 'Download for Windows' page. The header includes the Git logo and the tagline '--fast-version-control'. A search bar is located in the top right corner. The left sidebar contains links for 'About', 'Documentation', 'Downloads' (highlighted), 'GUI Clients', 'Logos', and 'Community'. The main content area is titled 'Download for Windows' and provides instructions for downloading the latest version (2.36.0) for Windows. It lists various download options: 'Standalone Installer', '32-bit Git for Windows Setup', '64-bit Git for Windows Setup', 'Portable ("thumbdrive edition")', '32-bit Git for Windows Portable', and '64-bit Git for Windows Portable'. It also includes a section for 'Using winget tool' with a command to install Git. A note at the bottom mentions the current source code release is version 2.36.0.

git --fast-version-control

Search entire site...

About

Documentation

Downloads

GUI Clients

Logos

Community

The entire **Pro Git book** written by Scott Chacon and Ben Straub is available to read online for free. Dead tree versions are available on Amazon.com.

Download for Windows

Click here to download the latest (2.36.0) 64-bit version of Git for Windows. This is the most recent maintained build. It was released 2 days ago, on 2022-04-20.

Other Git for Windows downloads

Standalone Installer

32-bit Git for Windows Setup.

64-bit Git for Windows Setup.

Portable ("thumbdrive edition")

32-bit Git for Windows Portable.

64-bit Git for Windows Portable.

Using winget tool

Install winget tool if you don't already have it, then type this command in command prompt or Powershell.

```
winget install --id Git.Git -e --source winget
```

The current source code release is version 2.36.0. If you want the newer version, you can build it from the source code.

Y seguir los siguientes pasos para su instalación

<https://curse-fang-83b.notion.site/Curso-Git-Github-58cc64406aec4845b5a65eb63e90a01c>

2) Instalar Docker

Podemos descargar el Docker desde la siguiente pagina

<https://docs.docker.com/desktop/windows/install/>

Install Docker Desktop on Windows

Estimated reading time: 10 minutes

🔔 Update to the Docker Desktop terms

Commercial use of Docker Desktop in larger enterprises (more than 250 employees OR more than \$10 million USD in annual revenue) now requires a paid subscription. The grace period for those that will require a paid subscription ends on January 31, 2022. [Learn more.](#)

Welcome to Docker Desktop for Windows. This page contains information about Docker Desktop for Windows system requirements, download URL, instructions to install and update Docker Desktop for Windows.

📄 Download Docker Desktop for Windows

Docker Desktop for Windows

Podemos seguir los pasos del video para instalar correctamente el docker

https://www.youtube.com/watch?v=_et7H0EQ8fY

Es necesario instalar wsl2 y actualizarlo, en la página oficial ya no se encuentra disponible pero lo puedes encontrar en la siguiente pagina

<https://www.linkedin.com/pulse/windows-subsystem-linux-installation-guide-10-vicknes-chandrasekaran>

Step 4 - Download the Linux kernel update package

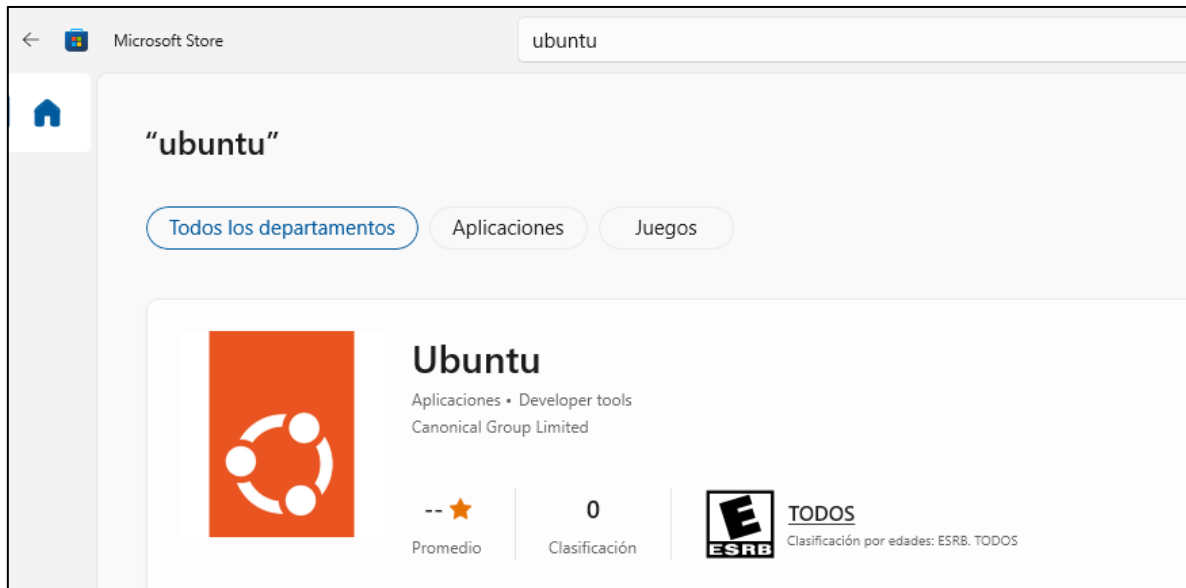
1. Download the latest package:

- [WSL2 Linux kernel update package for x64 machines](#)

Docker proporciona un pequeño tutorial de prueba, puedes seguirlo para comprender mejor los comandos para descargar una imagen y crear contenedores.

3) Instalar Ubuntu

Despues de descargar el Ubuntu



Puede que requiera la creación de usuario, lo creamos. La clave debe estar en minúsculas para evitar excepciones.

```
Retype new password:
passwd: password updated successfully
Installation successful!
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.10.16.3-microsoft-standard-WSL2 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Wed Apr 13 19:31:30 -05 2022

System load:  0.0      Processes:      8
Usage of /:   0.5% of 250.98GB   Users logged in: 0
Memory usage: 3%      IPv4 address for eth0: 172.20.136.126
Swap usage:   0%

1 update can be applied immediately.
To see these additional updates run: apt list --upgradable

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

This message is shown once a day. To disable it please create the
/home/enrique/.hushlogin file.
enrique@DESKTOP-90GQR90:~$
```

Después de instalar Ubuntu podemos ejecutar los comandos del docker en el terminal.

Minusculas

enrique

2021

```
docker run --name repo alpine/git clone \
https://github.com/docker/getting-started.git
docker cp repo:/git/getting-started/ .
```



```
cd getting-started
docker build -t docker101tutorial .
```



```
docker run -d -p 80:80 \
--name docker-tutorial docker101tutorial
```



```
docker tag docker101tutorial /docker101tutorial
docker push /docker101tutorial
```



<http://localhost/tutorial/>

Congratulations! You have started the container for this tutorial! Let's first explain the command that you just ran. In case you forgot, here's the command:

docker run -d -p 80:80 docker/getting-started

You'll notice a few flags being used. Here's some more info on them:

- -d - run the container in detached mode (in the background)
- -p 80:80 - map port 80 of the host to port 80 in the container
- docker/getting-started - the image to use

docker pull hello-world

Contenedor de prueba

ERROR AL EJECUTAR UN CONTENEDOR

Según la documentación oficial de docker después de ejecutar el siguiente comando `docker run -it --name my_od -P -p 5901:5901 cmsopendata/cmssw_5_3_32_vnc:latest /bin/bash` se debería abrir automáticamente un bash Shell en el CMS open data environment.

Como se muestra en la siguiente imagen,

Bash

```
docker run -it --name my_od -P -p 5901:5901 cmsopendata/cmssw_5_3_32_vnc:latest /bin/bash
```

Output

```
Setting up CMSSW_5_3_32
CMSSW should now be available.
~/CMSSW_5_3_32/src $
```

Pero en este caso no se abre el bash Shell en el CMS open data environment. Simplemente crea el contenedor con la imagen ya descargada.

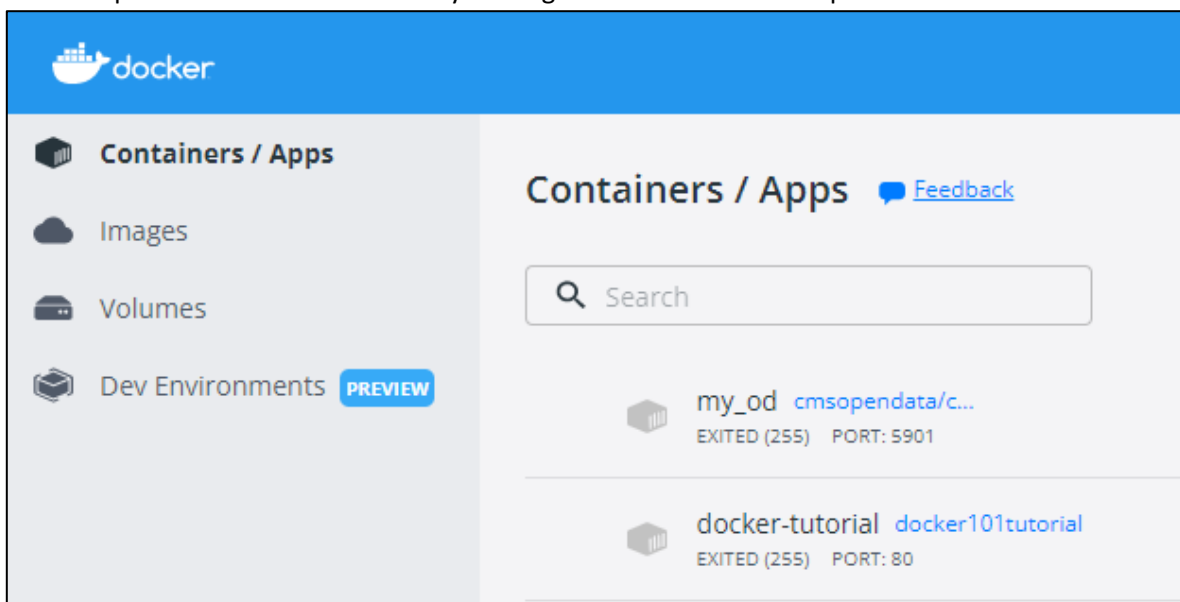
```
enrique@DESKTOP-90GQR90:~$ docker run -it --name my_od -P -p 5901:5901 cmsopendata/cmssw_5_3_32_vnc:latest /bin/bash
docker: Error response from daemon: Conflict. The container name "/my_od" is already in use by container "bd6bba42db676fb9cc6c598dededc4352f9de38abb7ae43805acba6443307883". You have to remove (or rename) that container to be able to reuse that name.
See 'docker run --help'.
enrique@DESKTOP-90GQR90:~$
```

Podemos ver los contenedores con `docker ps -a` y comprobar que si existe.

```
enrique@DESKTOP-90GQR90:~$ docker ps -a
```

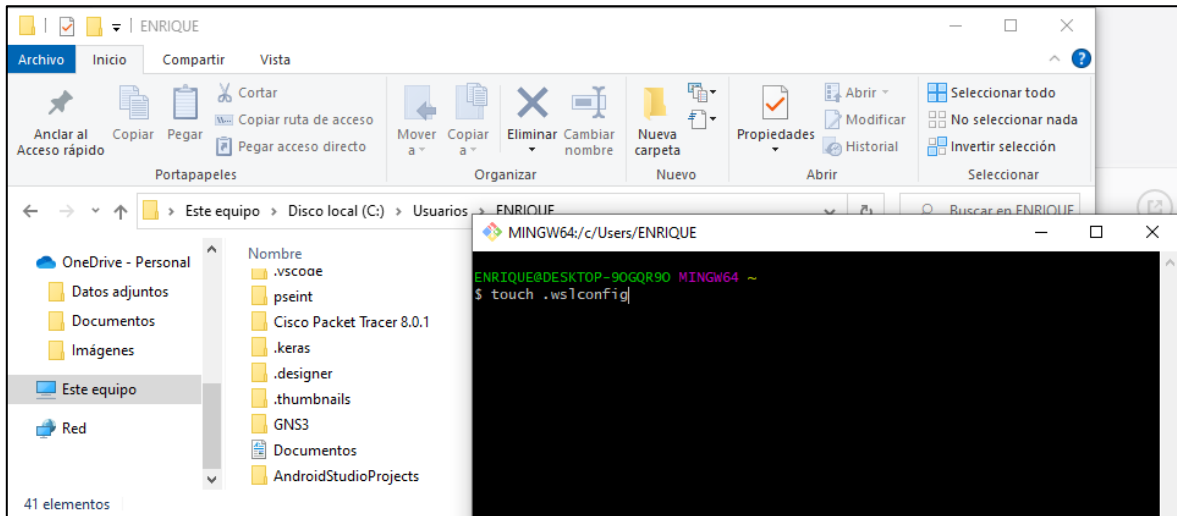
CONTAINER ID	IMAGE	PORTS	NAMES	COMMAND	CREATED	STATUS
bd6bba42db67	cmsopendata/cmssw_5_3_32_vnc:latest		my_od	"/opt/cms/entrypoint..."	13 minutes ago	Exited (139)
6130b0b11b90	docker101tutorial			"/docker-entrypoint..."	4 days ago	Exited (255)
ab90dc948115	alpine/git	0.0.0.0:80->80/tcp	docker-tutorial	"git clone https://g..."	4 days ago	Exited (0)

También podemos ver el contenedor y la imagen en el Docker Desktop

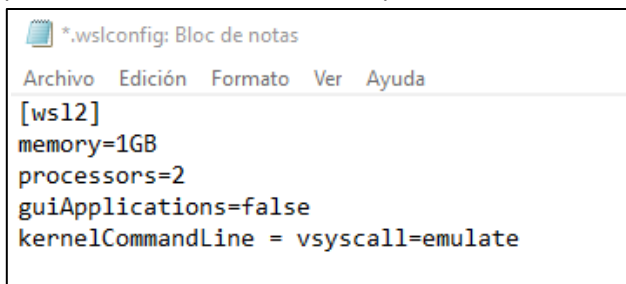


Aunque intentemos remover y volver a crear el contenedor, no abrirá el bash shell

Para resolver este problema creamos un archivo con el comando `touch .wslconfig` dentro del Usuario que se encuentra en el Disco Local C.



Dentro del archivo colocamos las siguientes configuraciones (como estamos en un entorno Windows podemos usar un bloc de notas pero también funciona con los terminales de Ubuntu o Git).

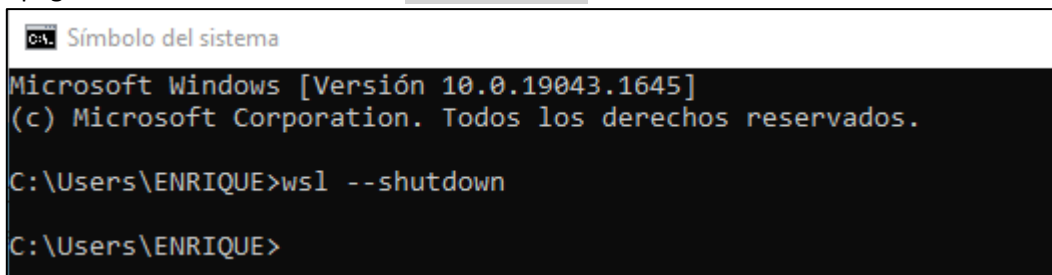


Las configuraciones dicen lo siguiente:

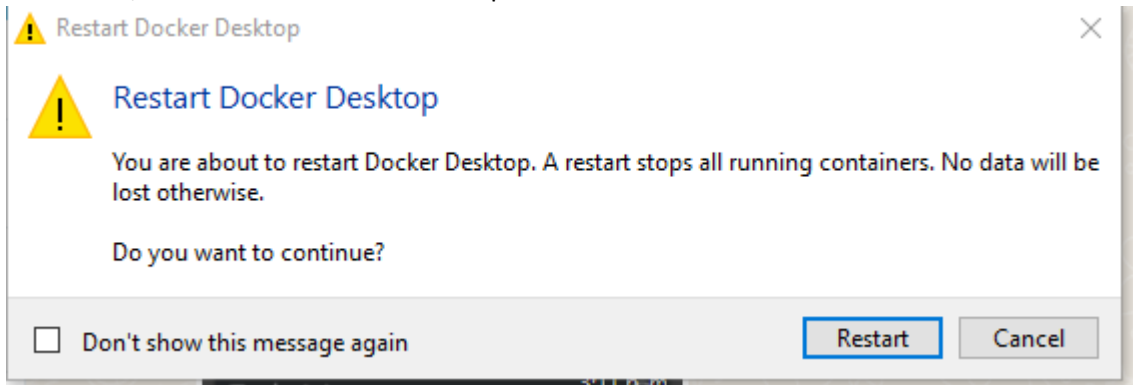
- Limita la memoria de la máquina virtual en WSL2 hasta 1 GB.
- La máquina virtual WSL 2 use dos procesadores virtuales.
- Activar o desactivar la compatibilidad con aplicaciones GUI en WSL. Solo disponible para Windows 11.
- Habilita imágenes base de Linux más antiguas como Centos 6.

Para más configuraciones puede revisar el siguiente link: <https://docs.microsoft.com/en-us/windows/wsl/wsl-config#wsl-2-settings>

Apagamos el wsl con el comando `wsl --shutdown`



Por último, reiniciamos el Docker Desktop.



Ahora, comprobamos si funciona la configuración con el comando `docker start -i my_od` porque el contenedor ya existe.

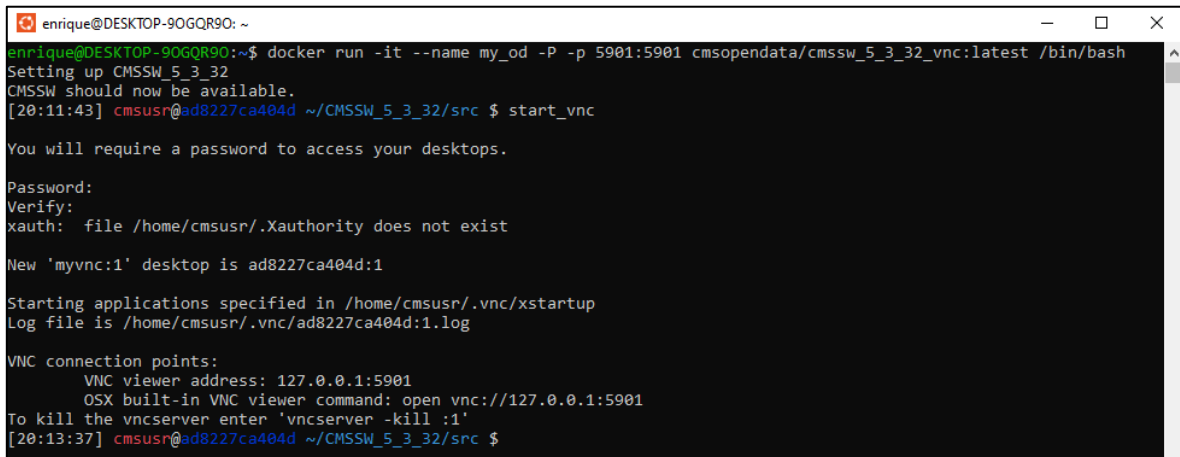
```
enrique@DESKTOP-90GQR90: ~  
enrique@DESKTOP-90GQR90:~$ docker start -i my_od  
[03:23:19] cmsusr@ce4e19907a69 ~/CMSSW_5_3_32/src $
```

También es válido eliminar el contenedor con `docker rm my_od` (no la imagen) y volver a ejecutar el contenedor.

```
enrique@DESKTOP-90GQR90: ~  
enrique@DESKTOP-90GQR90:~$ docker ps -a  
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS  
31dff0f1ded5   cmsopendata/cmssw_5_3_32_vnc:latest  "/opt/cms/entrypoint..." About a minute ago Exited (0) 18 seconds  
ago  
6130b0b11b90   docker101tutorial                  "/docker-entrypoint..." 8 days ago     Exited (255) 7 days ago  
ago  
0.0.0.0:80->80/tcp docker-tutorial  
enrique@DESKTOP-90GQR90:~$ docker rm my_od  
my_od  
enrique@DESKTOP-90GQR90:~$ docker run -it --name my_od -P -p 5901:5901 cmsopendata/cmssw_5_3_32_vnc:latest /bin/bash  
Setting up CMSSW_5_3_32  
CMSSW should now be available.  
[20:11:43] cmsusr@ad8227ca404d ~/CMSSW_5_3_32/src $
```

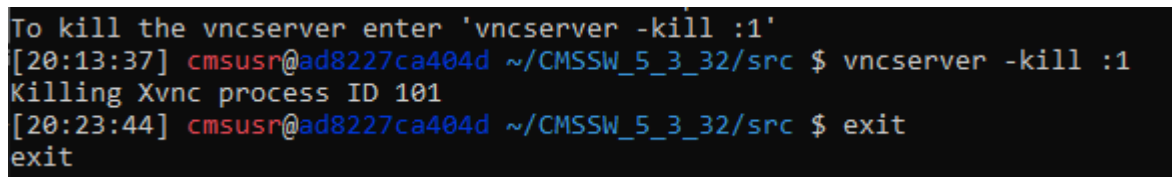
DESCARGAR UNA IMAGEN E INICIAR UN CONTENEDOR

Ejecutamos el comando `docker run -it --name my_od -P -p 5901:5901 cmsopendata/cmssw_5_3_32_vnc:latest /bin/bash` para descargar la imagen e iniciar el contenedor. También pide una contraseña cualquiera.



```
enrique@DESKTOP-90GQR90: ~  
enrique@DESKTOP-90GQR90:~$ docker run -it --name my_od -P -p 5901:5901 cmsopendata/cmssw_5_3_32_vnc:latest /bin/bash  
Setting up CMSSW_5_3_32  
CMSSW should now be available.  
[20:11:43] cmsusr@ad8227ca404d ~/CMSSW_5_3_32/src $ start_vnc  
  
You will require a password to access your desktops.  
  
Password:  
Verify:  
xauth: file /home/cmsusr/.Xauthority does not exist  
  
New 'myvnc:1' desktop is ad8227ca404d:1  
  
Starting applications specified in /home/cmsusr/.vnc/xstartup  
Log file is /home/cmsusr/.vnc/ad8227ca404d:1.log  
  
VNC connection points:  
  VNC viewer address: 127.0.0.1:5901  
  OSX built-in VNC viewer command: open vnc://127.0.0.1:5901  
To kill the vncserver enter 'vncserver -kill :1'  
[20:13:37] cmsusr@ad8227ca404d ~/CMSSW_5_3_32/src $
```

Después de ejecutar `start_vnc` se mostrará el comando para matar el vncserver. Si no ejecuta el comando antes de salir no podrá abrir la ventana de gráficos la próxima vez que use el mismo contenedor.



```
To kill the vncserver enter 'vncserver -kill :1'  
[20:13:37] cmsusr@ad8227ca404d ~/CMSSW_5_3_32/src $ vncserver -kill :1  
Killing Xvnc process ID 101  
[20:23:44] cmsusr@ad8227ca404d ~/CMSSW_5_3_32/src $ exit  
exit
```

A continuación, se realizan unos pasos para comprobar que el entorno de CMS está instalado y operativo en su contenedor Docker, y que tiene acceso a los archivos de datos abiertos de CMS.

Con `cmsenv` creamos las variables de tiempo de ejecución de CMS. Luego creamos un directorio de trabajo y dentro de ese directorio creamos un esqueleto para el analizador con `mkedanlzf` DemoAnalyzer.


```

enrique@DESKTOP-90GQR90:~$ docker run -it --name my_od -P -p 5901:5901 cmsopendata/cms5w_5_3_32_vnc:latest /bin/bash
Setting up CMSSW_5_3_32
Waiting for release information to be obtained via https://cmsddt.cern.ch/SDT/releases.map (timeout in 7s)
CMSSW should now be available.
[02:14:48] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src $ cmsenv
[02:15:11] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src $ mkdir Demo
[02:17:01] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src $ cd Demo
[02:17:06] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src/Demo $ makedanlir DemoAnalyzer
I: using skeleton: /opt/cms/slc6_amd64_gcc472/cms/cmssw/CMSSW_5_3_32/bin/slc6_amd64_gcc472/mkTemplates/EDAnalyzer/Conf
File_cfg.py
I: authors name is: , determined by the gcov entry
I: creating file: DemoAnalyzer/demoanalyzer_cfg.py
I: using skeleton: /opt/cms/slc6_amd64_gcc472/cms/cmssw/CMSSW_5_3_32/bin/slc6_amd64_gcc472/mkTemplates/EDAnalyzer/edan
alyzer.cc
I: authors name is: , determined by the gcov entry
I: creating file: DemoAnalyzer/src/DemoAnalyzer.cc
I: using skeleton: /opt/cms/slc6_amd64_gcc472/cms/cmssw/CMSSW_5_3_32/bin/slc6_amd64_gcc472/mkTemplates/EDAnalyzer/Build
dFile.temp
I: authors name is: , determined by the gcov entry
I: creating file: DemoAnalyzer/BuildFile.xml
I: using skeleton: /opt/cms/slc6_amd64_gcc472/cms/cmssw/CMSSW_5_3_32/bin/slc6_amd64_gcc472/mkTemplates/EDAnalyzer/CfiF
ile_cfi.py
I: authors name is: , determined by the gcov entry
I: creating file: DemoAnalyzer/python/demoanalyzer_cfi.py
[02:17:31] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src/Demo $

```

Salimos de Demo y compilamos el código con `scram b`

```

[21:41:36] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src/Demo $ cd ..
[21:41:40] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src $ scram b
***WARNING: No need to export library once you have declared your library as plugin. Please cleanup src/Demo/DemoAnalyz
er/BuildFile by removing the <export></export> section.
>> Local Products Rules ..... started
>> Local Products Rules ..... done
>> Building CMSSW version CMSSW_5_3_32 ----
>> Entering Package Demo/DemoAnalyzer
>> Creating project symlinks
src/Demo/DemoAnalyzer/python -> python/Demo/DemoAnalyzer
Entering library rule at Demo/DemoAnalyzer
>> Compiling edm plugin /home/cmsusr/CMSSW_5_3_32/src/Demo/DemoAnalyzer/src/DemoAnalyzer.cc
>> Building edm plugin tmp/slc6_amd64_gcc472/src/Demo/DemoAnalyzer/src/DemoDemoAnalyzer/libDemoDemoAnalyzer.so
Leaving library rule at Demo/DemoAnalyzer
@@@ Running edmWriteConfigs for DemoDemoAnalyzer
--- Registered EDM Plugin: DemoDemoAnalyzer
>> Leaving Package Demo/DemoAnalyzer
>> Package Demo/DemoAnalyzer built
>> Subsystem Demo built
>> Local Products Rules ..... started
>> Local Products Rules ..... done
gmake[1]: Entering directory `/home/cmsusr/CMSSW_5_3_32'
>> Creating project symlinks
src/Demo/DemoAnalyzer/python -> python/Demo/DemoAnalyzer
>> Done python symlink
>> Compiling python modules cfipython/slc6_amd64_gcc472
>> Compiling python modules python
>> Compiling python modules src/Demo/DemoAnalyzer/python
>> All python modules compiled
@@@ Refreshing Plugins:edmPluginRefresh

```

Modificamos el archivo de configuración para que pueda acceder a un archivo de datos abiertos de CMS con `nano Demo/DemoAnalyzer/demoanalyzer_cfg.py` y quedaría de la siguiente forma

```

GNU nano 2.8.9 File: Demo/DemoAnalyzer/demoanalyzer_cfg.py

import FWCore.ParameterSet.Config as cms

process = cms.Process("Demo")

process.load("FWCore.MessageService.MessageLogger_cfi")

process.maxEvents = cms.untracked.PSet( input = cms.untracked.int32(10) )

process.source = cms.Source("PoolSource",
    # replace 'myfile.root' with the source file you want to use
    fileName = cms.untracked.vstring(
        'root://eospublic.cern.ch/eos/opendata/cms/Run2012B/DoubleMuParked/AOD/22Jan2013-v1/10000/1EC938EF-ABEC-E211-94E0-90E68A442F24.root'
    )
)

process.demo = cms.EDAnalyzer('DemoAnalyzer'
)

process.p = cms.Path(process.demo)

```

Ejecutamos el cms ejecutable con la nueva configuración.

```
[21:47:32] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src $ nano Demo/DemoAnalyzer/demoanalyzer_cfg.py
[21:54:27] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src $ cmsRun Demo/DemoAnalyzer/demoanalyzer_cfg.py
220424 21:55:04 529 Xrd: XrdClientConn: Error resolving this host's domain name.
220424 21:55:06 529 secgsi_InitProxy: cannot access private key file: /home/cmsusr/.globus/userkey.pem
220424 21:55:07 529 Xrd: CheckErrorStatus: Server [eospublic.cern.ch] declared: (error code: 3005)
24-Apr-2022 21:55:07 CEST Initiating request to open file root://eospublic.cern.ch/eos/opendata/cms/Run2012B/DoubleMuParked/AOD/22Jan2013-v1/10000/1EC938EF-ABEC-E211-94E0-90E6BA442F24.root
24-Apr-2022 21:55:17 CEST Successfully opened file root://eospublic.cern.ch/eos/opendata/cms/Run2012B/DoubleMuParked/AOD/22Jan2013-v1/10000/1EC938EF-ABEC-E211-94E0-90E6BA442F24.root
Begin processing the 1st record. Run 195013, Event 24425389, LumiSection 66 at 24-Apr-2022 21:56:38.153 CEST
Begin processing the 2nd record. Run 195013, Event 24546773, LumiSection 66 at 24-Apr-2022 21:56:38.157 CEST
Begin processing the 3rd record. Run 195013, Event 24679037, LumiSection 66 at 24-Apr-2022 21:56:38.159 CEST
Begin processing the 4th record. Run 195013, Event 24839453, LumiSection 66 at 24-Apr-2022 21:56:38.160 CEST
Begin processing the 5th record. Run 195013, Event 24894477, LumiSection 66 at 24-Apr-2022 21:56:38.161 CEST
Begin processing the 6th record. Run 195013, Event 24980717, LumiSection 66 at 24-Apr-2022 21:56:38.162 CEST
Begin processing the 7th record. Run 195013, Event 25112869, LumiSection 66 at 24-Apr-2022 21:56:38.163 CEST
Begin processing the 8th record. Run 195013, Event 25484261, LumiSection 66 at 24-Apr-2022 21:56:38.164 CEST
Begin processing the 9th record. Run 195013, Event 25702821, LumiSection 66 at 24-Apr-2022 21:56:38.164 CEST
Begin processing the 10th record. Run 195013, Event 25961949, LumiSection 66 at 24-Apr-2022 21:56:38.165 CEST
24-Apr-2022 21:56:38 CEST Closed file root://eospublic.cern.ch/eos/opendata/cms/Run2012B/DoubleMuParked/AOD/22Jan2013-v1/10000/1EC938EF-ABEC-E211-94E0-90E6BA442F24.root

=====
MessageLogger Summary

type      category      sev      module      subroutine      count      total
-----
1 fileAction      -s file_close      1          1
2 fileAction      -s file_open       2          2

type      category      Examples: run/evt      run/evt      run/evt
-----
1 fileAction      PostEndRun
2 fileAction      pre-events      pre-events

Severity  # Occurrences  Total Occurrences
-----
system    3              3
```

CMSSW

El software CMS (CMSSW) es una colección de bibliotecas de software que utiliza el experimento CMS para adquirir, producir, procesar e incluso analizar sus datos.

Instalación y Ejecución (En Docker):

A este nivel ya deberíamos tener instalado el Docker. Ahora no estamos instalando CMSSW sino configurando un entorno para ello. CMSSW ya estaba instalado. Así que iniciamos el contenedor con `docker start -i <theNameOfyourContainer>` y luego `cmsenv` para establecer algunas variables ambientales para su área de trabajo.

Podemos ver a donde apunta la variable `CMSSW_RELEASE_BASE` con `echo $CMSSW_RELEASE_BASE`. Como estamos usando un Contenedor Docker la variable apunta a una instalación local de CMSSW.

```
enrique@DESKTOP-90GQR90:~$ docker start -i my_od
Setting up CMSSW_5_3_32
Waiting for release information to be obtained via https://cmssdt.cern.ch/SDT/releases.map (timeout in 7s)
WARNING: There already exists /home/cmsusr/CMSSW_5_3_32 area for SCRAM_ARCH slc6_amd64_gcc472.
CMSSW should now be available.
[00:44:14] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src $ cmsenv
[00:45:50] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src $ echo $CMSSW_RELEASE_BASE
/opt/cms/slc6_amd64_gcc472/cms/cmssw/CMSSW_5_3_32
[00:45:57] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src $
```

cmsRun, El CMSSW ejecutable

Todos los paquetes que componen la versión de CMSSW en uso ya han sido compilados y vinculados a un solo ejecutable, que se llama `cmsRun`. Dara un error porque `cmsRun` porque necesita un archivo de configuración.

Podemos ejecutar el archivo de configuración `Demo/DemoAnalyzer/demoanalyzer_cfg.py` y almacenar el resultado en un archivo dummy.log y ejecutarlo en segundo plano con `cmsRun`

Demo/DemoAnalyzer/demoanalyzer_cfg.py > dummy.log 2>&1 &. Puedes consultar el desarrollo de tu trabajo con `tail -f dummy.log` o cuando finalice puedes usar `cat dummy.log`

```
[01:03:42] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src $ cmsRun
cmsRun: No configuration file given.
For usage and an options list, please do 'cmsRun --help'.
[1]+  Done                  cmsRun Demo/DemoAnalyzer/demoanalyzer_cfg.py > dummy.log 2>&1
[01:05:41] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src $ cmsRun Demo/DemoAnalyzer/demoanalyzer_cfg.py > dummy.log 2>&1 &
[1] 101
[01:06:06] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src $ tail -f dummy.log
220425 01:06:08 101 Xrd: XrdClientConn: Error resolving this host's domain name.
220425 01:06:10 101 Xrd: CheckErrorStatus: Server [eospublic.cern.ch] declared: (error code: 3005)
25-Apr-2022 01:06:10 CEST Initiating request to open file root://eospublic.cern.ch/eos/opendata/cms/Run2012B/DoubleMuParked/AOD/22Jan2013-v1/10000/1EC938EF-ABEC-E211-
94E0-90E6BA442F24.root
25-Apr-2022 01:06:19 CEST Successfully opened file root://eospublic.cern.ch/eos/opendata/cms/Run2012B/DoubleMuParked/AOD/22Jan2013-v1/10000/1EC938EF-ABEC-E211-94E0-90
E6BA442F24.root
Begin processing the 1st record. Run 195013, Event 24425389, LumiSection 66 at 25-Apr-2022 01:09:53.189 CEST
Begin processing the 2nd record. Run 195013, Event 24546773, LumiSection 66 at 25-Apr-2022 01:09:53.191 CEST
Begin processing the 3rd record. Run 195013, Event 24679037, LumiSection 66 at 25-Apr-2022 01:09:53.191 CEST
Begin processing the 4th record. Run 195013, Event 24839453, LumiSection 66 at 25-Apr-2022 01:09:53.192 CEST
Begin processing the 5th record. Run 195013, Event 24894477, LumiSection 66 at 25-Apr-2022 01:09:53.193 CEST
Begin processing the 6th record. Run 195013, Event 24980717, LumiSection 66 at 25-Apr-2022 01:09:53.193 CEST
Begin processing the 7th record. Run 195013, Event 25112869, LumiSection 66 at 25-Apr-2022 01:09:53.194 CEST
Begin processing the 8th record. Run 195013, Event 25484261, LumiSection 66 at 25-Apr-2022 01:09:53.194 CEST
Begin processing the 9th record. Run 195013, Event 25702821, LumiSection 66 at 25-Apr-2022 01:09:53.195 CEST
Begin processing the 10th record. Run 195013, Event 25961949, LumiSection 66 at 25-Apr-2022 01:09:53.195 CEST
25-Apr-2022 01:09:53 CEST Closed file root://eospublic.cern.ch/eos/opendata/cms/Run2012B/DoubleMuParked/AOD/22Jan2013-v1/10000/1EC938EF-ABEC-E211-94E0-90E6BA442F24.ro
ot
=====
MessageLogger Summary
-----
type      category      sev      module      subroutine      count      total
-----
1 fileAction      -s file_close      1          1
2 fileAction      -s file_open       2          2
-----
type      category      Examples: run/evt      run/evt      run/evt
-----
1 fileAction      PostEndRun
2 fileAction      pre-events      pre-events
-----
Severity      # Occurrences      Total Occurrences
-----
System              3                  3
```

Compilación

Usamos `scram`, la herramienta de administración de versiones utilizada para CMSSW, para compilar el código, solo entra en el paquete Demo/DemoAnalyzer que creamos localmente para validar nuestra configuración anterior. Si compila en src, todos los paquetes allí se compilarán. Sin embargo, si ingresa a un subpaquete específico, como Demo/DemoAnalyzer, solo se compilará el código en ese subpaquete.

```
[01:14:34] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src $ scram b
Reading cached build data
>> Local Products Rules ..... started
>> Local Products Rules ..... done
>> Building CMSSW version CMSSW_5_3_32 ----
>> Entering Package Demo/DemoAnalyzer
>> Creating project symlinks
src/Demo/DemoAnalyzer/python -> python/Demo/DemoAnalyzer
>> Leaving Package Demo/DemoAnalyzer
>> Package Demo/DemoAnalyzer built
>> Subsystem Demo built
>> Local Products Rules ..... started
>> Local Products Rules ..... done
gmake[1]: Entering directory `/home/cmsusr/CMSSW_5_3_32'
>> Creating project symlinks
src/Demo/DemoAnalyzer/python -> python/Demo/DemoAnalyzer
>> Done python_symlink
>> Compiling python modules cfipython/slc6_amd64_gcc472
>> Compiling python modules python
>> Compiling python modules src/Demo/DemoAnalyzer/python
>> All python modules compiled
>> Plugging of all type refreshed.
>> Done generating edm plugin poisoned information
gmake[1]: Leaving directory `/home/cmsusr/CMSSW_5_3_32'
[01:19:00] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src $
```

Adicionales

El entorno CMSSW viene con otros scripts/herramientas ejecutables como script `mkedanlzf` que crea esqueletos para EDAnalyzers que luego se pueden modificar o expandir. Podemos averiguar acerca de otros scripts escribiendo `mked` y presionando la tecla `Tab`

```
[01:19:00] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src $ mked
mkedanlpr mkedfltr mkedlpr mkedprod
[01:19:00] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src $ mked
```

Para conocer la cantidad de eventos en el archivo ROOT que se encuentra en el archivo de configuración Demo/DemoAnalyzer/demoanalyzer_cfg.py ejecutamos `edmEventSize -v root://eospublic.cern.ch//eos/opendata/cms/Run2012B/DoubleMuParked/AOD/22Jan2013-v1/10000/1EC938EF-ABEC-E211-94E0-90E6BA442F24.root |grep Events`

EDAnalyzers

Los EDAnalyzers son módulos que permiten el acceso de solo lectura al Evento. Son útiles para producir histogramas, informes, estadísticas, etc.

Podemos explorar el paquete DemoAnalyzer. CMSSW podría ser muy exigente con la estructura de sus paquetes, los scripts u otras herramientas esperan tener una estructura de Paquete/Subpaquete. DemoAnalyzer es solo un paquete más de CMSSW. Sin embargo, los encabezados y la implementación de nuestro DemoAnalyzer están codificados en un solo archivo en el directorio src llamado DemoAnalyzer.cc. También tenemos un archivo demoanalyzer_cfg.py que es el configurador predeterminado para el código DemoAnalyzer.cc y el archivo BuildFile.xml, donde podemos incluir cualquier dependencia si es necesario.

```
enrique@DESKTOP-90GQR90:~$ docker start -i my_od
Setting up CMSSW_5_3_32
Waiting for release information to be obtained via https://cmsdt.cern.ch/SDT/releases.map (timeout in 7s)
WARNING: There already exists /home/cmsusr/CMSSW_5_3_32 area for SCRAM_ARCH slc6_amd64_gcc472.
CMSSW should now be available.
[22:36:12] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src $ cmsenv
[22:38:22] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src $ ls Demo/DemoAnalyzer/
BuildFile.xml demoanalyzer_cfg.py doc interface python src test
[22:39:35] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src $ ls Demo/DemoAnalyzer/src
DemoAnalyzer.cc
[22:39:47] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src $
```

Source

Jugando con el archivo DemoAnalyzer.cc

Como se mencionó, el archivo DemoAnalyzer.cc es el archivo principal de nuestro EDAnalyzer y la estructura por defecto es siempre la misma.

Con `nano Demo/DemoAnalyzer/src/DemoAnalyzer.cc` podemos ver su contenido

```
// system include files
#include <memory>

// user include files
#include "FWCore/Framework/interface/Frameworkfwd.h"
#include "FWCore/Framework/interface/EDAnalyzer.h"

#include "FWCore/Framework/interface/Event.h"
#include "FWCore/Framework/interface/MakerMacros.h"

#include "FWCore/ParameterSet/interface/ParameterSet.h"
```

Incluye librerías importantes como:

- La clase `Event.h`, que contiene esencialmente todos los accesorios que se necesitan para extraer información del Evento, es decir, de la colisión de partículas.

- La clase `ParameterSet.h`, nos permitirá extraer parámetros de configuración, que se pueden manipular usando el archivo python `Demo/DemoAnalyzer/demoanalyzer_cfg.py`.

Para más información puedes visitar repositorio de CMSSW en Github de preferencia `CMSSW_5_3_X`.

Ahora, para practicar añadiremos librerías necesarias para extraer la energía de todos los muones en el evento y la de vectores estándar de C++.

```
// system include files
#include <memory>

// user include files
#include "FWCore/Framework/interface/Frameworkfwd.h"
#include "FWCore/Framework/interface/EDAnalyzer.h"

#include "FWCore/Framework/interface/Event.h"
#include "FWCore/Framework/interface/MakerMacros.h"

#include "FWCore/ParameterSet/interface/ParameterSet.h"

//classes to extract Muon information
#include "DataFormats/MuonReco/interface/Muon.h"
#include "DataFormats/MuonReco/interface/MuonFwd.h"

#include <vector>
```

Si seguimos bajando encontraremos la siguiente función que hereda de la clase `edm::EDAnalyzer`

```
class DemoAnalyzer : public edm::EDAnalyzer {
public:
    explicit DemoAnalyzer(const edm::ParameterSet&);
    ~DemoAnalyzer();

    static void fillDescriptions(edm::ConfigurationDescriptions& descriptions);

private:
    virtual void beginJob() ;
    virtual void analyze(const edm::Event&, const edm::EventSetup&);
    virtual void endJob() ;

    virtual void beginRun(edm::Run const&, edm::EventSetup const&);
    virtual void endRun(edm::Run const&, edm::EventSetup const&);
    virtual void beginLuminosityBlock(edm::LuminosityBlock const&, edm::EventSetup const&);
    virtual void endLuminosityBlock(edm::LuminosityBlock const&, edm::EventSetup const&);

    // -----member data -----
};
```

Agreguemos la declaración de un vector para nuestros valores de energía.

```
class DemoAnalyzer : public edm::EDAnalyzer {
public:
    explicit DemoAnalyzer(const edm::ParameterSet&);
    ~DemoAnalyzer();

    static void fillDescriptions(edm::ConfigurationDescriptions& descriptions);

private:
    virtual void beginJob() ;
    virtual void analyze(const edm::Event&, const edm::EventSetup&);
    virtual void endJob() ;

    virtual void beginRun(edm::Run const&, edm::EventSetup const&);
    virtual void endRun(edm::Run const&, edm::EventSetup const&);
    virtual void beginLuminosityBlock(edm::LuminosityBlock const&, edm::EventSetup const&);
    virtual void endLuminosityBlock(edm::LuminosityBlock const&, edm::EventSetup const&);

    // -----member data -----
    std::vector<float> muon_e;
};
```

La clase DemoAnalyzer tiene constructores y destructores. Se pasa un objeto `ParameterSet` al constructor, se lee cualquier configuración que podamos terminar implementando en nuestro archivo de configuración de Python `Demo/DemoAnalyzer/demoanalyzer_cfg.py`.

```
//  
// constructors and destructor  
//  
DemoAnalyzer::DemoAnalyzer(const edm::ParameterSet& iConfig)  
{  
    //now do what ever initialization is needed  
}  
  
DemoAnalyzer::~DemoAnalyzer()  
{  
    // do anything here that needs to be done at destruction time  
    // (e.g. close files, deallocate resources etc.)  
}
```

El corazón del archivo fuente es el método `analyze`. Cualquier cosa que entre dentro de esta rutina se repetirá en todos los eventos disponibles. Un objeto `edm::Event` y un objeto `edm::EventSetup` se pasan de forma predeterminada. Mientras que del Evento podemos extraer información como objetos físicos, del EventSetup podemos obtener información como preescalas de activación.

```
// ----- method called for each event -----  
void  
DemoAnalyzer::analyze(const edm::Event& iEvent, const edm::EventSetup& iSetup)  
{  
    using namespace edm;  
  
#ifdef THIS_IS_AN_EVENT_EXAMPLE  
    Handle<ExampleData> pIn;  
    iEvent.getByLabel("example",pIn);  
#endif  
  
#ifdef THIS_IS_AN_EVENTSETUP_EXAMPLE  
    ESHandle<SetupData> pSetup;  
    iSetup.get<SetupRecord>().get(pSetup);  
#endif  
}
```

Ahora agreguemos algunas líneas en el método `analyze` para que podamos recuperar la energía de todos los muones en cada evento y lo imprimiremos. Salimos después de haber guardado.

```
// ----- method called for each event -----  
void  
DemoAnalyzer::analyze(const edm::Event& iEvent, const edm::EventSetup& iSetup)  
{  
    using namespace edm;  
    //clean the container  
    muon_e.clear();  
  
    //define the handler and get by label  
    Handle<reco::MuonCollection> mymuons;  
    iEvent.getByLabel("muons", mymuons);  
  
    //if collection is valid, loop over muons in event  
    if(mymuons.isValid()){  
        for (reco::MuonCollection::const_iterator itmuon=mymuons->begin(); itmuon!=mymuons->end(); ++itmuon){  
            muon_e.push_back(itmuon->energy());  
        }  
    }  
  
    //print the vector  
    for(unsigned int i=0; i < muon_e.size(); i++){  
        std::cout <<"Muon # "<<i<<" with E = "<<muon_e.at(i)<<" GeV."<<std::endl;  
    }  
  
#ifdef THIS_IS_AN_EVENT_EXAMPLE  
    Handle<ExampleData> pIn;  
    iEvent.getByLabel("example",pIn);  
#endif  
  
#ifdef THIS_IS_AN_EVENTSETUP_EXAMPLE  
    ESHandle<SetupData> pSetup;  
    iSetup.get<SetupRecord>().get(pSetup);  
#endif  
}
```

La compilación con `scram b` fallará, se debe a que las clases de Muon que agregamos introdujeron algunas dependencias que deben ser atendidas en `BuildFile.xml`

```
[23:44:13] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src $ scram b
>> Local Products Rules ..... started
>> Local Products Rules ..... done
>> Building CMSSW version CMSSW_5_3_32 ----
>> Entering Package Demo/DemoAnalyzer
>> Creating project symlinks
src/Demo/DemoAnalyzer/python -> python/Demo/DemoAnalyzer
>> Compiling edm plugin /home/cmsusr/CMSSW_5_3_32/src/Demo/DemoAnalyzer/src/DemoAnalyzer.cc
In file included from /opt/cms/slc6_amd64_gcc472/cms/cmssw/CMSSW_5_3_32/src/DataFormats/TrackingRecHit/interface/TrackingRecHit.h:4:0,
from /opt/cms/slc6_amd64_gcc472/cms/cmssw/CMSSW_5_3_32/src/DataFormats/TrackingRecHit/interface/RecSegment.h:17,
from /opt/cms/slc6_amd64_gcc472/cms/cmssw/CMSSW_5_3_32/src/DataFormats/DTRecHit/interface/DTRecSegment4D.h:16,
from /opt/cms/slc6_amd64_gcc472/cms/cmssw/CMSSW_5_3_32/src/DataFormats/DTRecHit/interface/DTRecSegment4DCollection.h:20,
from /opt/cms/slc6_amd64_gcc472/cms/cmssw/CMSSW_5_3_32/src/DataFormats/MuonReco/interface/MuonSegmentMatch.h:6,
from /opt/cms/slc6_amd64_gcc472/cms/cmssw/CMSSW_5_3_32/src/DataFormats/MuonReco/interface/MuonChamberMatch.h:5,
from /opt/cms/slc6_amd64_gcc472/cms/cmssw/CMSSW_5_3_32/src/DataFormats/MuonReco/interface/Muon.h:17,
from /home/cmsusr/CMSSW_5_3_32/src/Demo/DemoAnalyzer/src/DemoAnalyzer.cc:40:
/opt/cms/slc6_amd64_gcc472/cms/cmssw/CMSSW_5_3_32/src/DataFormats/CLHEP/interface/AlgebraicObjects.h:8:33: fatal error: CLHEP/Matrix/Vector.h: No such file or directory
compilation terminated.
In file included from /opt/cms/slc6_amd64_gcc472/cms/cmssw/CMSSW_5_3_32/src/DataFormats/TrackingRecHit/interface/TrackingRecHit.h:4:0,
from /opt/cms/slc6_amd64_gcc472/cms/cmssw/CMSSW_5_3_32/src/DataFormats/TrackingRecHit/interface/RecSegment.h:17,
from /opt/cms/slc6_amd64_gcc472/cms/cmssw/CMSSW_5_3_32/src/DataFormats/DTRecHit/interface/DTRecSegment4D.h:16,
from /opt/cms/slc6_amd64_gcc472/cms/cmssw/CMSSW_5_3_32/src/DataFormats/DTRecHit/interface/DTRecSegment4DCollection.h:20,
from /opt/cms/slc6_amd64_gcc472/cms/cmssw/CMSSW_5_3_32/src/DataFormats/MuonReco/interface/MuonSegmentMatch.h:6,
from /opt/cms/slc6_amd64_gcc472/cms/cmssw/CMSSW_5_3_32/src/DataFormats/MuonReco/interface/MuonChamberMatch.h:5,
from /opt/cms/slc6_amd64_gcc472/cms/cmssw/CMSSW_5_3_32/src/DataFormats/MuonReco/interface/Muon.h:17,
from /home/cmsusr/CMSSW_5_3_32/src/Demo/DemoAnalyzer/src/DemoAnalyzer.cc:40:
/opt/cms/slc6_amd64_gcc472/cms/cmssw/CMSSW_5_3_32/src/DataFormats/CLHEP/interface/AlgebraicObjects.h:8:33: fatal error: CLHEP/Matrix/Vector.h: No such file or directory
compilation terminated.
gmake: *** [tmp/slc6_amd64_gcc472/src/Demo/DemoAnalyzer/src/DemoDemoAnalyzer/DemoAnalyzer.o] Error 1
gmake: *** [There are compilation/build errors. Please see the detail log above.] Error 2
[23:45:58] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src $
```

Así que editamos con `nano Demo/DemoAnalyzer/BuildFile.xml` y colocamos

```
GNU nano 2.0.9 File: Demo/DemoAnalyzer/BuildFile.xml Modified

<use name="FWCore/Framework"/>
<use name="FWCore/PluginManager"/>
<use name="DataFormats/MuonReco"/>
<use name="FWCore/ParameterSet"/>
<flags EDM_PLUGIN="1"/>
<export>
  <lib name="1"/>
</export>
```

Ahora si lo volvemos a ejecutar funcionara.

```
[23:50:21] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src $ scram b
Reading cached build data
****WARNING: No need to export library once you have declared your library as plugin. Please cleanup src/Demo/DemoAnalyzer/BuildFile by removing the <export></export> s
ection.
>> Local Products Rules ..... started
>> Local Products Rules ..... done
>> Building CMSSW version CMSSW_5_3_32 ----
>> Entering Package Demo/DemoAnalyzer
>> Creating project symlinks
src/Demo/DemoAnalyzer/python -> python/Demo/DemoAnalyzer
Entering library rule at Demo/DemoAnalyzer
>> Compiling edm plugin /home/cmsusr/CMSSW_5_3_32/src/Demo/DemoAnalyzer/src/DemoAnalyzer.cc
>> Building edm plugin tmp/slc6_amd64_gcc472/src/Demo/DemoAnalyzer/src/DemoDemoAnalyzer/libDemoDemoAnalyzer.so
Leaving library rule at Demo/DemoAnalyzer
@@@ Running edmWriteConfigs for DemoDemoAnalyzer
--- Registered EDM Plugin: DemoDemoAnalyzer
>> Leaving Package Demo/DemoAnalyzer
>> Package Demo/DemoAnalyzer built
>> Subsystem Demo built
>> Local Products Rules ..... started
>> Local Products Rules ..... done
gmake[1]: Entering directory '/home/cmsusr/CMSSW_5_3_32'
>> Creating project symlinks
src/Demo/DemoAnalyzer/python -> python/Demo/DemoAnalyzer
>> Done python symlink
>> Compiling python modules cfipython/slc6_amd64_gcc472
>> Compiling python modules python
>> Compiling python modules src/Demo/DemoAnalyzer/python
>> All python modules compiled
@@@ Refreshing Plugins:edmPluginRefresh
>> Plugging of all type refreshed.
>> Done generating edm plugin poisoned information
gmake[1]: Leaving directory '/home/cmsusr/CMSSW_5_3_32'
[23:51:54] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src $
```

Luego podemos correr `cmsRun Demo/DemoAnalyzer/demoanalyzer_cfg.py > mylog.log 2>&1` & y ver el resultado con `tail -f mylog.log` ó `cat mylog.log`


```

Muon # 1 with E = 12.5258 GeV.
Begin processing the 4th record. Run 195013, Event 24839453, LumiSection 66 at 25-Apr-2022 23:55:54.114 CEST
Muon # 0 with E = 5.19378 GeV.
Muon # 1 with E = 13.6549 GeV.
Begin processing the 5th record. Run 195013, Event 24894477, LumiSection 66 at 25-Apr-2022 23:55:54.115 CEST
Muon # 0 with E = 10.3546 GeV.
Begin processing the 6th record. Run 195013, Event 24980717, LumiSection 66 at 25-Apr-2022 23:55:54.115 CEST
Muon # 1 with E = 25.2733 GeV.
Muon # 0 with E = 10.7545 GeV.
Begin processing the 7th record. Run 195013, Event 25112869, LumiSection 66 at 25-Apr-2022 23:55:54.116 CEST
Muon # 0 with E = 165.026 GeV.
Muon # 1 with E = 145.994 GeV.
Begin processing the 8th record. Run 195013, Event 25484261, LumiSection 66 at 25-Apr-2022 23:55:54.117 CEST
Muon # 0 with E = 130.171 GeV.
Muon # 1 with E = 73.1873 GeV.
Begin processing the 9th record. Run 195013, Event 25702821, LumiSection 66 at 25-Apr-2022 23:55:54.117 CEST
Muon # 0 with E = 24.0889 GeV.
Muon # 1 with E = 14.4771 GeV.
Begin processing the 10th record. Run 195013, Event 25961949, LumiSection 66 at 25-Apr-2022 23:55:54.118 CEST
Muon # 0 with E = 189.134 GeV.
Muon # 1 with E = 38.8268 GeV.
25-Apr-2022 23:55:54 CEST closed file root://eospublic.cern.ch//eos/opendata/cms/Run2012B/DoubleMuParked/AOD/22Jan2013-v1/10000/1EC938EF-ABEC-E211-94E0-90E68A442F24.root
ot

=====
MessageLogger Summary
-----
type      category      sev      module      subroutine      count      total
-----
1 fileAction      -s file_close      1          1
2 fileAction      -s file_open       2          2
-----

type      category      Examples: run/evt      run/evt      run/evt
-----
1 fileAction      PostEndRun
2 fileAction      pre-events      pre-events
-----

Severity      # Occurrences      Total Occurrences
-----
System              3              3
rc
[2]- Done      cmsRun Demo/DemoAnalyzer/demoanalyzer_cfg.py > mylog.log 2>&1
[23:56:08] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src $

```

Como ya habrás notado el archivo de origen debe modificarse de acuerdo con las necesidades del analizador.

Configuración

Si exploramos el directorio `Demo/DemoAnalyzer/Python` tenemos,

```

[02:45:41] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src $ ls Demo/DemoAnalyzer/python
__init__.py __init__.pyc demoanalyzer_cfi.py demoanalyzer_cfi.pyc
[02:45:45] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src $

```

Como recordaras hay un `demoanalyzer_cfg.py`, en el directorio `Demo/DemoAnalyzer/` y en este caso hay un `demoanalyzer_cfi.py`. Los descriptores `_cfg` definen una configuración de nivel superior y el `_cfi` funciona más como un archivo de inicialización de módulo.

Jugando con el archivo `demoanalyzer_cfg.py`

Abrimos el archivo con `nano Demo/DemoAnalyzer/demoanalyzer_cfg.py`

Encontrará en todos los archivos de configuración de CMSSW las siguientes instrucciones

```

GNU nano 2.0.9      File: Demo/DemoAnalyzer/demoanalyzer_cfg.py

import FWCore.ParameterSet.Config as cms

process = cms.Process("Demo")

process.load("FWCore.MessageService.MessageLogger_cfi")

process.maxEvents = cms.untracked.PSet( input = cms.untracked.int32(10) )

process.source = cms.Source("PoolSource",
    # replace 'myfile.root' with the source file you want to use
    fileName = cms.untracked.vstring(
        'root://eospublic.cern.ch//eos/opendata/cms/Run2012B/DoubleMuParked/AOD/22Jan2013-v1/10000/1EC938EF-ABEC-E211-94E0-90E68A442F24.root'
    )
)

process.demo = cms.EDAnalyzer('DemoAnalyzer'
)

process.p = cms.Path(process.demo)

```

La primera línea importa nuestras clases y funciones de Python específicas de CMS, y la segunda crea un objeto de proceso (se refiere a un proceso CMSSW). El proceso siempre necesita un nombre

en este caso es llamado "Demo". En la tercera línea vemos que carga algo, debido a la etiqueta `_cfi`, sabemos que es una pieza de código de Python que inicializa algún módulo. De hecho, es el servicio `MessageLogger` que controla cómo se maneja el registro de mensajes durante la ejecución del trabajo. La cadena `"FWCore.MessageService.MessageLogger_cfi"` le dice exactamente dónde buscarlo en Github si lo necesita.

Si queremos que el Framework informe cada 5 eventos en lugar de cada evento. Entonces uno puede simplemente agregar esta línea `process.MessageLogger.cerr.FwkReport.reportEvery = 5`

```
GNU nano 2.0.9 File: Demo/DemoAnalyzer/demoanalyzer_cfg.py
import FWCore.ParameterSet.Config as cms

process = cms.Process("Demo")

process.load("FWCore.MessageService.MessageLogger_cfi")
process.MessageLogger.cerr.FwkReport.reportEvery = 5
process.maxEvents = cms.untracked.PSet( input = cms.untracked.int32(10) )

process.source = cms.Source("PoolSource",
    # replace 'myfile.root' with the source file you want to use
    fileNameNames = cms.untracked.vstring(
        'root://eospublic.cern.ch//eos/opendata/cms/Run2012B/DoubleMuParked/AOD/22Jan2013-v1/10000/1EC938EF-ABEC-E211-94E0-90E6BA442F24.root'
    )
)

process.demo = cms.EDAnalyzer('DemoAnalyzer'
)

process.p = cms.Path(process.demo)
```

En la quinta línea es fácil adivinar que controla la cantidad de eventos que se van a procesar en nuestro trabajo CMSSW. `maxEvents` es una variable sin seguimiento dentro del Framework. El sistema realiza un seguimiento de los parámetros que se utilizan para crear cada elemento de datos en el evento y guarda esta información en los archivos de salida.

También cambiamos el número de eventos a 100,

```
GNU nano 2.0.9 File: Demo/DemoAnalyzer/demoanalyzer_cfg.py
import FWCore.ParameterSet.Config as cms

process = cms.Process("Demo")

process.load("FWCore.MessageService.MessageLogger_cfi")
process.MessageLogger.cerr.FwkReport.reportEvery = 5
process.maxEvents = cms.untracked.PSet( input = cms.untracked.int32(100) )

process.source = cms.Source("PoolSource",
    # replace 'myfile.root' with the source file you want to use
    fileNameNames = cms.untracked.vstring(
        'root://eospublic.cern.ch//eos/opendata/cms/Run2012B/DoubleMuParked/AOD/22Jan2013-v1/10000/1EC938EF-ABEC-E211-94E0-90E6BA442F24.root'
    )
)

process.demo = cms.EDAnalyzer('DemoAnalyzer'
)

process.p = cms.Path(process.demo)
```

Las siguientes líneas son el primer módulo que estamos conectando a nuestro objeto de proceso. Dentro del objeto de proceso debe haber exactamente un objeto asignado que tenga fuente de tipo Python y se usa para la entrada de datos. En la antepenúltima línea, tenemos un módulo que es solo una declaración de existencia porque está vacía debido a que `Demoanalyzer` fue creado recientemente.

Configurar nuestro DemoAnalyzer.cc

Supongamos que queremos extraer la energía de los muones de las colisiones de haz o de los rayos cósmicos. Podemos usar un InputTag que sea MuonsFromCosmics en lugar de solo muones. Debemos configurar DemoAnalyzer_cfg.py para que no tengamos que volver a compilar cada vez que queremos hacer el cambio.

Usamos nano Demo/DemoAnalyzer/src/DemoAnalyzer.cc y declaramos la etiqueta de entrada para MuonCollection,

```
virtual void beginRun(edm::Run const&, edm::EventSetup const&);
virtual void endRun(edm::Run const&, edm::EventSetup const&);
virtual void beginLuminosityBlock(edm::LuminosityBlock const&, edm::EventSetup const&);
virtual void endLuminosityBlock(edm::LuminosityBlock const&, edm::EventSetup const&);

// -----member data -----
std::vector<float> muon_e;

//declare the input tag for MuonCollection
edm::InputTag muonInput;
```

Modificamos el constructor para leer este InputTag de la configuración. Leerá la variable InputCollection de la configuración y la almacenará en el contenedor muonInput.

```
//
// constructors and destructor
//

DemoAnalyzer::DemoAnalyzer(const edm::ParameterSet& iConfig)
{
    //now do what ever initialization is needed
    muonInput = iConfig.getParameter<edm::InputTag>("InputCollection");
}
```

En la función analyze reemplazamos la línea iEvent.getByLabel("muons", mymuons); con iEvent.getByLabel(muonInput, mymuons);

```
// ----- method called for each event -----
void
DemoAnalyzer::analyze(const edm::Event& iEvent, const edm::EventSetup& iSetup)
{
    using namespace edm;
    //clean the container
    muon_e.clear();

    //define the handler and get by label
    Handle<reco::MuonCollection> mymuons;
    iEvent.getByLabel(muonInput, mymuons);

    //if collection is valid, loop over muons in event
    if(mymuons.isValid()){
        for (reco::MuonCollection::const_iterator itmuon=mymuons->begin(); itmuon!=mymuons->end(); ++itmuon){
            muon_e.push_back(itmuon->energy());
        }
    }
}
```

Finalmente, modificamos la declaración vacía process.demo = cms.EDAnalyzer('DemoAnalyzer') de Demo/DemoAnalyzer/demoanalyzer_cfg.py

```

GNU nano 2.0.9                                File: Demo/DemoAnalyzer/demoanalyzer_cfg.py

import FWCore.ParameterSet.Config as cms

process = cms.Process("Demo")

process.load("FWCore.MessageService.MessageLogger_cfi")
process.MessageLogger.cerr.FwkReport.reportEvery = 5
process.maxEvents = cms.untracked.PSet( input = cms.untracked.int32(100) )

process.source = cms.Source("PoolSource",
    # replace 'myfile.root' with the source file you want to use
    fileName = cms.untracked.vstring(
        'root://eospublic.cern.ch//eos/opendata/cms/Run2012B/DoubleMuParked/AOD/22Jan2013-v1/10000/1EC938EF-ABEC-E211-94E0-90E6BA442F24.root'
    )
)

process.demo = cms.EAnalyzer('DemoAnalyzer',
    InputCollection = cms.InputTag("muons")
)

process.p = cms.Path(process.demo)

```

Ahora podemos ingresar "muones" o "muonsfromcosmics", dependiendo de nuestras necesidades.

Antes de correr el programa podemos validar la sintaxis de su configuración usando `python Demo/DemoAnalyzer/demoanalyzer_cfg.py`

```

[04:04:48] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src $ python Demo/DemoAnalyzer/demoanalyzer_cfg.py
File "Demo/DemoAnalyzer/demoanalyzer_cfg.py", line 18
SyntaxError: Non-ASCII character '\xc3' in file Demo/DemoAnalyzer/demoanalyzer_cfg.py on line 18, but no encoding declared; see http://www.python.org/peps/pep-0263.html
for details
[04:04:55] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src $ nano Demo/DemoAnalyzer/demoanalyzer_cfg.py
[04:07:00] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src $ python Demo/DemoAnalyzer/demoanalyzer_cfg.py
File "Demo/DemoAnalyzer/demoanalyzer_cfg.py", line 19
    )
    ^
SyntaxError: invalid syntax
[04:07:02] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src $ nano Demo/DemoAnalyzer/demoanalyzer_cfg.py

```

Si existen errores intente colocar `# coding=utf-8` al inicio y tenga cuidado con espacios dentro del archivo.

```

# coding=utf-8
import FWCore.ParameterSet.Config as cms

```

Cuando el archivo este correcto no mostrara problemas.

```

[01:24:43] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src $ python Demo/DemoAnalyzer/demoanalyzer_cfg.py
[01:24:49] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src $

```

Ahora, podemos compilar con `scram b`

```
[01:32:02] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src $ scram b
>> Local Products Rules ..... started
>> Local Products Rules ..... done
>> Building CMSSW version CMSSW_5_3_32 ----
>> Entering Package Demo/DemoAnalyzer
>> Creating project symlinks
  src/Demo/DemoAnalyzer/python -> python/Demo/DemoAnalyzer
>> Compiling edm plugin /home/cmsusr/CMSSW_5_3_32/src/Demo/DemoAnalyzer/src/DemoAnalyzer.cc
>> Building edm plugin tmp/slc6_amd64_gcc472/src/Demo/DemoAnalyzer/src/DemoDemoAnalyzer/libDemoDemoAnalyzer.so
Leaving library rule at Demo/DemoAnalyzer
@@@ Running edmWriteConfigs for DemoDemoAnalyzer
--- Registered EDM Plugin: DemoDemoAnalyzer
>> Leaving Package Demo/DemoAnalyzer
>> Package Demo/DemoAnalyzer built
>> Subsystem Demo built
>> Local Products Rules ..... started
>> Local Products Rules ..... done
gmake[1]: Entering directory `/home/cmsusr/CMSSW_5_3_32'
>> Creating project symlinks
  src/Demo/DemoAnalyzer/python -> python/Demo/DemoAnalyzer
>> Done python_symlink
>> Compiling python modules cfipython/slc6_amd64_gcc472
>> Compiling python modules python
>> Compiling python modules src/Demo/DemoAnalyzer/python
>> All python modules compiled
@@@ Refreshing Plugins:edmPluginRefresh
>> Plugging of all type refreshed.
>> Done generating edm plugin poisoned information
gmake[1]: Leaving directory `/home/cmsusr/CMSSW_5_3_32'
[01:32:31] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src $
```

Ejecutamos el trabajo CMSSW,

```
[01:36:47] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src $ cmsRun Demo/DemoAnalyzer/demoanalyzer_cfg.py > mylog.log 2>&1 &
[1] 628
[01:36:57] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src $
```

Y vemos el resultado con `tail -f mylog.log`

```
Begin processing the 91st record. Run 195013, Event 26433779, LumiSection 67 at 30-Apr-2022 01:42:38.813 CEST
Muon # 0 with E = 8.94157 GeV.
Muon # 1 with E = 15.6143 GeV.
Muon # 0 with E = 12.5059 GeV.
Muon # 0 with E = 151.436 GeV.
Muon # 1 with E = 45.7421 GeV.
Muon # 0 with E = 373.762 GeV.
Muon # 1 with E = 36.528 GeV.
Muon # 0 with E = 62.9591 GeV.
Muon # 1 with E = 171.215 GeV.
Begin processing the 96th record. Run 195013, Event 27692659, LumiSection 67 at 30-Apr-2022 01:42:38.816 CEST
Muon # 0 with E = 260.454 GeV.
Muon # 1 with E = 40.6799 GeV.
Muon # 0 with E = 10.4986 GeV.
Muon # 1 with E = 9.11542 GeV.
Muon # 0 with E = 14.3677 GeV.
Muon # 1 with E = 110.193 GeV.
Muon # 2 with E = 33.4921 GeV.
Muon # 3 with E = 75.4299 GeV.
Muon # 0 with E = 117.046 GeV.
Muon # 1 with E = 43.5284 GeV.
Muon # 0 with E = 34.6616 GeV.
Muon # 1 with E = 54.6835 GeV.
30-Apr-2022 01:42:41 CEST Closed file root://eospublic.cern.ch//eos/opendata/cms/Run2012B/DoubleMuParked/AOD/22Jan2013-v1/10000/1EC938EF-ABEC-E211-94E0-90E68A442F24.root

=====
MessageLogger Summary
-----
type      category      sev      module      subroutine      count      total
-----
1 fileAction      -s file_close      1          1
2 fileAction      -s file_open       2          2
-----
type      category      Examples: run/evt      run/evt      run/evt
-----
1 fileAction      PostEndRun
2 fileAction      pre-events      pre-events
-----
Severity      # Occurrences      Total Occurrences
-----
System              3                  3
```

Podemos cambiar el nombre de `InputCollection` de "muons" a "muonsFromCosmics" sin recompilar el código y el resultado será diferente.

```

Muon # 0 with E = 36.681 GeV.
Muon # 1 with E = 20.3887 GeV.
Muon # 0 with E = 78.1146 GeV.
Muon # 1 with E = 39.0961 GeV.
Begin processing the 96th record. Run 195013, Event 27692659, LumiSection 67 at 30-Apr-2022 02:03:45.671 CEST
Muon # 0 with E = 447.44 GeV.
Muon # 1 with E = 0.241282 GeV.
Muon # 2 with E = 10.5963 GeV.
Muon # 3 with E = 42.9256 GeV.
Muon # 0 with E = 2.22145 GeV.
Muon # 1 with E = 10.4718 GeV.
Muon # 2 with E = 1.04273 GeV.
Muon # 3 with E = 6.55246 GeV.
Muon # 4 with E = 6.67922 GeV.
Muon # 0 with E = 4.79183 GeV.
Muon # 1 with E = 0.371742 GeV.
Muon # 2 with E = 114.528 GeV.
Muon # 3 with E = 52.6534 GeV.
Muon # 4 with E = 79.8534 GeV.
Muon # 0 with E = 87.6752 GeV.
Muon # 1 with E = 30.7821 GeV.
Muon # 0 with E = 1.51412 GeV.
Muon # 1 with E = 45.9054 GeV.
30-Apr-2022 02:03:46 CEST Closed file root://eospublic.cern.ch//eos/opendata/cms/Run2012B/DoubleMuParked/AOD/22Jan2013-v1/10000/1EC938EF-ABEC-E211-94E0-90E6BA442F24.root
ot

=====
MessageLogger Summary
-----
type      category      sev      module      subroutine      count      total
-----
1 fileAction      -s file_close      1          1
2 fileAction      -s file_open       2          2

type      category      Examples: run/evt      run/evt      run/evt
-----
1 fileAction      PostEndRun
2 fileAction      pre-events      pre-events

Severity      # Occurrences      Total Occurrences
-----
System              3                  3

```

Ejecutando algún código CMSSW ya disponible

CMSSW ejecuta su código utilizando Paths. Cada Camino puede ejecutar una serie de módulos. Por ahora solo tenemos una ruta llamada `p` que ejecuta el proceso de demostración, que corresponde a nuestro DemoAnalyzer.

En CMSSW, hay otros tipos de código que uno puede ejecutar como los EDAFilters utilizados para filtrar eventos. Por ejemplo, uno podría usar la clase de filtro `HLTHighLevel` para ejecutar solo eventos que hayan pasado un determinado tipo de desencadenante. Configurar el parámetro `HLTPaths` en ese módulo para que solo pase eventos que dispararon cualquier bit de activación con el patrón `HLT_Mu15*` y agregar el módulo a nuestra ruta de ejecución.

```

GNU nano 2.0.9                               File: Demo/DemoAnalyzer/demoanalyzer_cfg.py

# coding=utf-8
import FWCore.ParameterSet.Config as cms

process = cms.Process("Demo")

process.load("FWCore.MessageService.MessageLogger_cfi")
process.MessageLogger.cerr.FwkReport.reportEvery = 5
process.maxEvents = cms.untracked.PSet( input = cms.untracked.int32(100) )

process.source = cms.Source("PoolSource",
    # replace 'myfile.root' with the source file you want to use
    fileName = cms.untracked.vstring(
        'root://eospublic.cern.ch//eos/opendata/cms/Run2012B/DoubleMuParked/AOD/22Jan2013-v1/10000/1EC938EF-ABEC-E211-94E0-90E6BA442F24.root'
    )
)

process.demo = cms.EDAnalyzer('DemoAnalyzer', InputCollection = cms.InputTag("muons"))
process.load("HLTrigger.HLTfilters.hltHighLevel_cfi")
process.hltHighLevel.HLTPaths = cms.vstring('HLT_Mu7*')

process.p = cms.Path(process.hltHighLevel+process.demo)

```

La ejecución de la ruta de activación se detendrá si el módulo de filtro `hltHighLevel` arroja un resultado Falso. En este caso, el filtro impide claramente la ejecución de nuestro EDAAnalyzer.

```

94E0-90E68A442F24.root
30-Apr-2022 02:13:39 CEST Successfully opened file root://eospublic.cern.ch//eos/opendata/cms/Run2012B/DoubleMuParked/AOD/22Jan2013-v1/10000/1EC938EF-ABEC-E211-94E0-90E68A442F24.root
Begin processing the 1st record. Run 195013, Event 24425389, LumiSection 66 at 30-Apr-2022 02:16:10.506 CEST
Begin processing the 6th record. Run 195013, Event 24980717, LumiSection 66 at 30-Apr-2022 02:16:10.526 CEST
Begin processing the 11th record. Run 195013, Event 24530992, LumiSection 66 at 30-Apr-2022 02:16:10.528 CEST
Begin processing the 16th record. Run 195013, Event 25806950, LumiSection 66 at 30-Apr-2022 02:16:10.532 CEST
Begin processing the 21st record. Run 195013, Event 24568715, LumiSection 66 at 30-Apr-2022 02:16:10.536 CEST
Begin processing the 26th record. Run 195013, Event 25773019, LumiSection 66 at 30-Apr-2022 02:16:10.537 CEST
Begin processing the 31st record. Run 195013, Event 25407425, LumiSection 66 at 30-Apr-2022 02:16:10.539 CEST
Begin processing the 36th record. Run 195013, Event 24586506, LumiSection 66 at 30-Apr-2022 02:16:24.877 CEST
Begin processing the 41st record. Run 195013, Event 24369161, LumiSection 66 at 30-Apr-2022 02:16:24.881 CEST
Begin processing the 46th record. Run 195013, Event 25119433, LumiSection 66 at 30-Apr-2022 02:16:24.889 CEST
Begin processing the 51st record. Run 195013, Event 25759016, LumiSection 66 at 30-Apr-2022 02:16:24.892 CEST
Begin processing the 56th record. Run 195013, Event 25178292, LumiSection 66 at 30-Apr-2022 02:16:24.896 CEST
Begin processing the 61st record. Run 195013, Event 25118983, LumiSection 66 at 30-Apr-2022 02:16:24.902 CEST
Begin processing the 66th record. Run 195013, Event 24534099, LumiSection 66 at 30-Apr-2022 02:16:24.905 CEST
Begin processing the 71st record. Run 195013, Event 25156815, LumiSection 66 at 30-Apr-2022 02:16:24.906 CEST
Begin processing the 76th record. Run 195013, Event 27158713, LumiSection 67 at 30-Apr-2022 02:16:24.910 CEST
Begin processing the 81st record. Run 195013, Event 26563495, LumiSection 67 at 30-Apr-2022 02:16:24.912 CEST
Begin processing the 86th record. Run 195013, Event 27467127, LumiSection 67 at 30-Apr-2022 02:16:24.913 CEST
Begin processing the 91st record. Run 195013, Event 26433779, LumiSection 67 at 30-Apr-2022 02:16:24.919 CEST
Begin processing the 96th record. Run 195013, Event 27692659, LumiSection 67 at 30-Apr-2022 02:16:24.920 CEST
30-Apr-2022 02:16:25 CEST Closed file root://eospublic.cern.ch//eos/opendata/cms/Run2012B/DoubleMuParked/AOD/22Jan2013-v1/10000/1EC938EF-ABEC-E211-94E0-90E68A442F24.root
ot

=====
MessageLogger Summary

type      category      sev      module      subroutine      count      total
-----
1 fileAction      -s file_close      1          1
2 fileAction      -s file_open       2          2

type      category      Examples: run/evt      run/evt      run/evt
-----
1 fileAction      PostEndRun
2 fileAction      pre-events      pre-events

Severity  # Occurrences  Total Occurrences
-----
System    3              3

```

CMS Trigger

Sistemas de adquisición y Trigger CMS

Las colisiones en el LHC (Large Hadron Collider) ocurren a un ritmo cercano a 40 MHz. Cada colisión es detectada por los diferentes subdetectores, la cantidad de información que generan puede caber en un archivo de 1 MB. Si tuviéramos que registrar cada colisión, probablemente llenaría todo el espacio de disco disponible en el mundo en unos pocos días.

No todas las colisiones que ocurren en el LHC son interesantes, solo mantenemos los interesantes y, lo más importante, no perderse los de calidad de descubrimiento. Para conseguirlo necesitamos un Trigger.

Antes de pasar a los detalles del sistema de activación revisemos unas terminologías:

Haz: conjunto de partículas de un mismo origen, que se propagan sin dispersión.

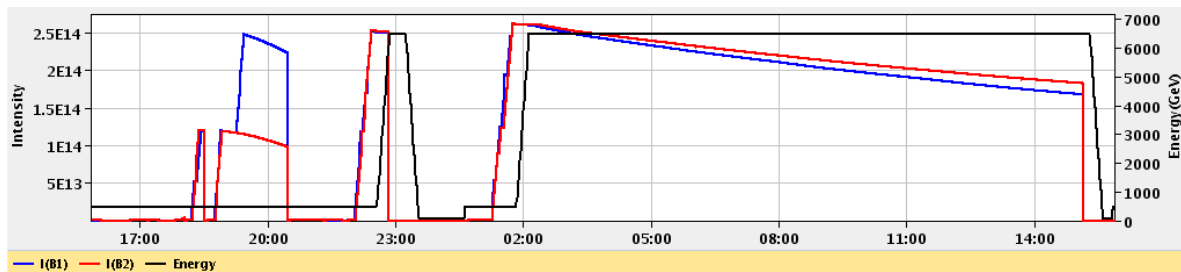
Relleno: Cada vez que el LHC inyecta haces en la máquina marca el inicio de lo que se conoce como Relleno. Cuando arrojan los haces, marca el final de ese Relleno. A cada relleno se le asigna un número único. Algunos de estos rellenos se declaran estables y buenos para la física y es entonces cuando nosotros, los detectores, recopilamos los datos buenos para hacer una investigación de calidad.

Ejecutar: a medida que ocurren colisiones en el LHC, CMS (y los otros detectores) deciden si comienzan a registrar datos. Cada vez que se presiona el botón de inicio, se inicia una nueva ejecución y se le asigna un número único. La ejecución puede detenerse por una variedad de razones, como si el LHC vuelca el relleno, pero generalmente se debe a que hay una falla en uno de los miles de canales electrónicos en el detector que hace que se detenga la adquisición. Esta es la

razón por la que los datos registrados (por los detectores) y entregados (por el LHC) no suelen ser los mismos. Cuando se reinicia la adquisición, se asigna un nuevo número de ejecución.

Sección Lumi: al colisionar, la luminosidad instantánea entregada por el LHC se degrada debido a diferentes razones. Es decir, no es constante en el tiempo. Por razones prácticas, CMS agrupa los eventos que recopila en secciones de luminosidad, donde los valores de luminosidad pueden considerarse constantes.

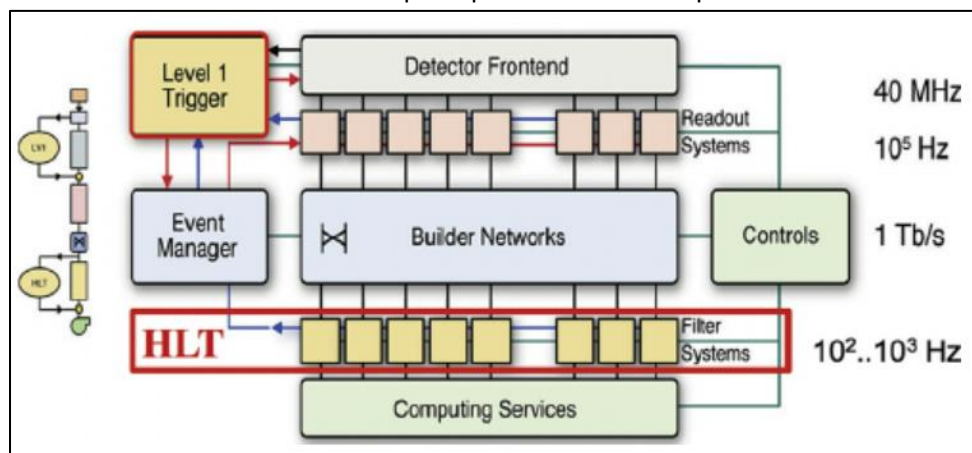
En el gráfico se pueden ver las diferentes inyecciones del haz (B). Para este caso en particular, alrededor de las 2 am de ese día, las colisiones se declararon estables y duraron hasta alrededor de las 15H. La luminosidad comienza a disminuir a medida que pasa el tiempo para este ejemplo de relleno.



Sistema Trigger

El sistema Trigger decide qué eventos registrar. Es como tomar una fotografía rápida y, aunque sea un poco borrosa, decidir si es interesante conservarlos o no para una futura inspección más minuciosa.

CMS hace esto en dos pasos principales. El primero, el Trigger de Nivel 1 (L1), implementado en hardware (FPGA rápidos), reduce la tasa de entrada de 40 Mhz a alrededor de 100 KHz. El otro paso es el Trigger de alto Nivel (HLT), que se ejecuta en máquinas comerciales con C++ y Python, donde la tasa de entrada se nivela alrededor del presupuesto máximo disponible de alrededor de 1-2 KHz.



Hay cientos de Trigger diferentes en CMS. Cada uno de ellos está diseñado para recoger cierto tipo de eventos, con diferentes intensidades y topologías. Por ejemplo, el disparador HLT_Mu15

seleccionará eventos con al menos un muón con 15 GeV de momento transversal. Estos Trigger se implementan usando código CMSSW, usando módulos que se pueden organizar para lograr el objetivo deseado. Por lo tanto, los Trigger son solo rutas en el sentido de CMSSW, y se podría extraer mucha información explorando la configuración de Python de estas rutas.

En el ejercicio previo de CMSSW, al final del archivo de configuración, podríamos tener algo como,

```
process.mypath = cms.Path (process.m1+process.m2+process.s1+process.m3)
```

donde m1, m2, m3 podrían ser módulos CMSSW (EDAnalyzers individuales, EDFilters, EDProducers, etc.) y s1 podría ser una Secuencia de módulos.

Un ejemplo de una disposición de este tipo para un Trigger HLT es el siguiente,

```
process.HLT_Mu15_v2 = cms.Path( process.HLTBeginSequenceBPTX +
process.hltL1sL1SingleMu10 + process.hltPreMu15 + process.hltL1SingleMu10L1Filtered0 +
process.hltL2Mu10L2Filtered10 + process.HLT2muonrecoSequence + process.hltL3Muon15 +
process.HLT3muonrecoSequence + process.HLTEndSequence )
```

Los Trigger son código, y esas piezas de código cambian constantemente. Las modificaciones a un Trigger podrían implicar un identificador de versión diferente, es decir, los nombres de los Trigger cambien de una ejecución a otra. Por ejemplo, nuestro HLT_Mu15 en realidad podría ser HLT_Mu15_v2 o HLT_Mu15_v3, etc., según la versión.

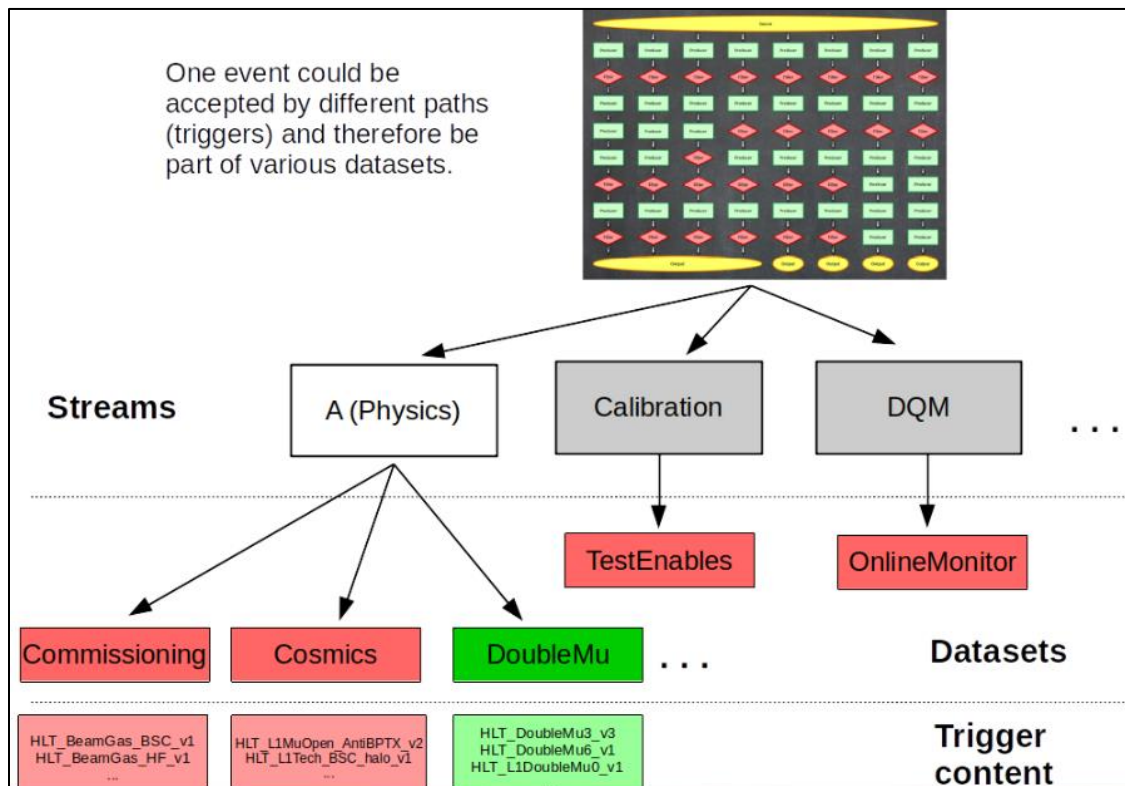
Preescalas

Si uno piensa en diferentes procesos físicos que tienen diferentes secciones transversales. En términos generales, es mucho más probable que registremos un evento de sesgo mínimo que un evento en el que se produce un bosón Z. Aún menos probable es registrar un evento con un bosón de Higgs. Podríamos tener Trigger llamados, digamos, HLT_ZBosonRecorder para el que está a cargo de filtrar eventos portadores de Z, o HLT_HiggsBosonRecorder para el que acepta Higgses. Las preescalas están diseñadas para mantener estas entradas bajo control, por ejemplo, registrando solo 1 de cada 100 colisiones que producen un bosón Z probable, o 1 de cada 1 colisiones que producen un bosón de Higgs potencial. En el primer caso, la preescala sería 100, mientras que para el segundo sería 1; si un activador tiene una preescala de 1, es decir, registra todos los eventos que identifica, lo llamamos sin preescala.

A medida que cae la luminosidad, las preescalas se pueden relajar y, por lo tanto, pueden cambiar para ejecutarse en el mismo relleno. Un Trigger se puede preescalar en L1 así como en los niveles HLT. Los Trigger L1 tienen su propia nomenclatura y se pueden utilizar como semillas Trigger HLT.

Trigger, flujos y conjuntos de datos

Una vez que los eventos son aceptados por posiblemente más de un tipo de Trigger, se transmiten en diferentes categorías, llamadas flujos y luego se clasifican y organizan en conjuntos de datos primarios. La mayoría, pero no todos, los conjuntos de datos que pertenecen al flujo A, el flujo de física, están o estarán disponibles como datos abiertos de CMS. Hay muchos más conjuntos de datos y flujos que los que se muestran en la figura a continuación.



Los conjuntos de datos en lectura son ejemplos de aquellos que no planeamos publicar como datos abiertos, mientras que el que está en verde es un ejemplo de los que sí hacemos.

Finalmente, cabe mencionar que:

- Un evento puede ser activado por muchas rutas de activación (trigger paths).
- Las rutas de activación son únicas para cada conjunto de datos.
- El mismo evento puede llegar en dos conjuntos de datos diferentes (puede ocurrir la duplicación de eventos).
- El sistema de activación de CMS filtra los eventos poco interesantes y mantiene un presupuesto alto para el flujo de datos interesantes.
- Un Trigger es una ruta CMSSW, que se compone de varios módulos de software.
- Las preescalas de activación permiten que la adquisición de datos se ajuste a los cambios en la luminosidad instantánea mientras se mantiene bajo control la tasa de datos entrantes.
- Los sistemas Trigger permiten la clasificación y organización de conjuntos de datos por objetos físicos de interés.

Explorando los Trigger en su conjunto de datos

Después de elegir el conjunto de datos apropiado para su análisis, lo primero que debe decidir es qué Trigger usar.

Si entramos a <http://opendata.cern.ch/record/6024> encontrará una lista de Trigger que se transmitieron a ese conjunto de datos. Si hacemos clic en algún Trigger puede encontrar información adicional.

HLT trigger paths

The possible [HLT trigger paths](#) in this dataset are:

```
HLT_Ele13_eta2p1_WP90NoIso_LooseIsoPFTau20_L1ETM36
HLT_Ele13_eta2p1_WP90Rho_LooseIsoPFTau20
HLT_Ele13_eta2p1_WP90Rho_LooseIsoPFTau20_L1ETM36
HLT_Ele20_CaloldVT_CalIsoRhoT_TrkIdT_TrkIsoT
HLT_Ele20_CaloldVT_CalIsoRhoT_TrkIdT_TrkIsoT_LooseIsoPFTau20
HLT_Ele20_CaloldVT_CalIsoRhoT_TrkIdT_TrkIsoT_LooseIsoPFTau20L1Jet
HLT_Ele20_CaloldVT_CalIsoRhoT_TrkIdT_TrkIsoT_LooseIsoPFTau22L1Jet
HLT_Ele20_CaloldVT_TrkIdT_LooseIsoPFTau20
HLT_Ele22_CaloldVT_CalIsoRhoT_TrkIdT_TrkIsoT_LooseIsoPFTau20
HLT_Ele22_eta2p1_WP90NoIso_LooseIsoPFTau20
HLT_Ele22_eta2p1_WP90Rho_LooseIsoPFTau20
HLT_IsoMu15_eta2p1_L1ETM20
HLT_IsoMu15_eta2p1_LooseIsoPFTau35_Trk20_Prong1_L1ETM20
HLT_IsoMu17_eta2p1_LooseIsoPFTau20
HLT_IsoMu18_eta2p1_LooseIsoPFTau20
HLT_IsoMu18_eta2p1_MediumIsoPFTau25_Trk1_eta2p1
HLT_IsoMu18_eta2p1_MediumIsoPFTau25_Trk1_eta2p1_Reg
HLT_IsoMu18_eta2p1_MediumIsoPFTau25_Trk5_eta2p1
HLT_IsoMu20_eta2p1_LooseIsoPFTau20
HLT_IsoMu8_eta2p1_LooseIsoPFTau20
HLT_IsoMu8_eta2p1_LooseIsoPFTau20_L1ETM26
```

Los Trigger pueden no ser persistentes y solo estar disponibles para ciertas ejecuciones. Además, como se mencionó, el código de un Trigger evoluciona. Puede haber diferentes versiones de este disparador, por ejemplo, HLT_IsoMu8_eta2p1_LooseIsoPFTau20_v1, o v2, v3, etc.

Para verificar la información del Trigger, necesitamos usar las herramientas de activación proporcionadas en CMSSW. En este link <http://opendata.cern.ch/record/5004> le indica algunos ejemplos del uso de dicho código. Usemos, por ejemplo, algunos de los fragmentos presentados en el archivo fuente del paquete GeneralInfoAnalyzer (<https://github.com/cms-opendata-analyses/TriggerInfoTool/blob/2011/GeneralInfoAnalyzer/src/TriggerInfoAnalyzer.cc>) para volcar todos los Trigger en nuestro conjunto de datos.

Primero, asegúrese de ir a su área CMSSW_5_3_32/src y emita el comando `cmsenv`.

```
enrique@DESKTOP-90GQR90:~$ docker start -i 53d3985b25a5
Setting up CMSSW_5_3_32
Waiting for release information to be obtained via https://cmssdt.cern.ch/SDT/releases.map (timeout in 7s)
WARNING: Reading release information from https://cmssdt.cern.ch/SDT/releases.map is timed out, ignoring any release checks.
WARNING: There already exists /home/cmsusr/CMSSW_5_3_32 area for SCRAM_ARCH slc6_amd64_gcc472.
CMSSW should now be available.
[18:04:58] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src $ cmsenv
[18:05:24] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src $
```

Para simplificar el ejercicio, usaremos el mismo archivo fuente que usamos para la validación, `Demo/DemoAnalyzer/src/DemoAnalyzer.cc`. Lo modificamos para volcar todos los Trigger en nuestro conjunto de datos. En el proceso, comentaremos lo que hicimos para extraer la energía del muón para que no sature nuestra salida.

Insertamos el encabezado de la clase `HLTConfigProvider`:

```

GNU nano 2.0.9      File: Demo/DemoAnalyzer/src/DemoAnalyzer.cc      Modified

#include "FWCore/ParameterSet/interface/ParameterSet.h"

//classes to extract Muon information
#include "DataFormats/MuonReco/interface/Muon.h"
#include "DataFormats/MuonReco/interface/MuonFwd.h"

//for trigger configuration
#include "HLTrigger/HLTcore/interface/HLTConfigProvider.h"

//the standard c++ vector class
#include<vector>

```

Luego, agregamos la declaración de las variables en nuestro archivo y un objeto de clase HLTConfigProvider, que podemos usar para extraer la información sobre cuál fue la configuración del Trigger para alguna ejecución,

```

// -----member data -----
//std::vector<float> muon_e;

//declare the input tag for MuonCollection
//edm::InputTag muonInput;

//for trigger config
std::string  processName_;
std::string  datasetName_;
//HLT config provider object
HLTConfigProvider hltConfig_;

```

Añadimos las líneas para leer la configuración en el constructor e imprímala (tenga en cuenta que la forma en que se hace difiere un poco de lo que hicimos anteriormente para los muones. Son, por supuesto, equivalentes),

```

//
// constructors and destructor
//
DemoAnalyzer::DemoAnalyzer(const edm::ParameterSet& iConfig):
processName_(iConfig.getParameter<std::string>("processName")),
datasetName_(iConfig.getParameter<std::string>("datasetName"))
{
    //now do what ever initialization is needed
    //muonInput = iConfig.getParameter<edm::InputTag>("InputCollection")
    using namespace std;
    using namespace edm;

    //Print the configuration just to check
    cout << "Here is the information passed to the constructor:" << endl;
    cout << "Configuration: " << endl
         << "   ProcessName = " << processName_ << endl
         << "   DataSetName = " << datasetName_ << endl;
}

```

También comentamos todo el material de muones en el método de Analyze,

```
// ----- method called for each event -----
void
DemoAnalyzer::analyze(const edm::Event& iEvent, const edm::EventSetup& iSetup)
{
    using namespace edm;
    //clean the container
    //muon_e.clear();

    //define the handler and get by label
    //Handle<reco::MuonCollection> mymuons;
    //iEvent.getByLabel(muonInput, mymuons);

    //if collection is valid, loop over muons in event
    //if(mymuons.isValid()){
    //    for (reco::MuonCollection::const_iterator itmuon=mymuons->begin(); itmuon!=mymuons->end(); ++itmuon){
    //        muon_e.push_back(itmuon->energy());
    //    }
    //}

    //print the vector
    //for(unsigned int i=0; i < muon_e.size(); i++){
    //    std::cout <<"Muon # "<<i<<" With E = "<<muon_e.at(i)<<" GeV."<<std::endl;
    //}

#ifdef THIS_IS_AN_EVENT_EXAMPLE
    Handle<ExampleData> pIn;
    iEvent.getByLabel("example",pIn);
#endif

#ifdef THIS_IS_AN_EVENTSETUP_EXAMPLE
    ESHandle<SetupData> pSetup;
    iSetup.get<SetupRecord>().get(pSetup);
#endif
}

```

Finalmente, modificamos la función beginRun dando un nombre a los argumentos de iRun e iSetup y agregando el volcado del Trigger. Recuerda que tenemos que comprobar los activadores disponibles en cada cambio de ejecución,

```
void
DemoAnalyzer::beginRun(edm::Run const& iRun, edm::EventSetup const& iSetup)
{
    using namespace std;
    using namespace edm;

    //If the hltConfig can be initialized, then the below is an example of
    //how to extract the config information for the trigger from the
    //so-called provenance.

    // The trigger configuration can change from
    // run to run (during the run is the same),
    // so it needs to be called here.

    /// "init" return value indicates whether intitialisation has succeeded
    /// "changed" parameter indicates whether the config has actually changed

    bool changed(true);
    if (hltConfig_.init(iRun,iSetup,processName_,changed)) {
        if (changed) {
            const vector<string> triggerNamesInDS = hltConfig_.datasetContent(datasetName_);
            for (unsigned i = 0; i < triggerNamesInDS.size(); i++) {
                cout<<triggerNamesInDS[i]<<endl;
            }
        }
    }
}

```

Antes de compilar, cambie su Demo/DemoAnalyzer/BuildFile.xml para incluir el paquete HLTrigger/HLTcore, donde reside HLTConfigProvider,

```
GNU nano 2.0.9 File: Demo/DemoAnalyzer/BuildFile.xml Modified
<use name="FWCore/Framework"/>
<use name="FWCore/PluginManager"/>
<use name="DataFormats/MuonReco"/>
<use name="FWCore/ParameterSet"/>
<use name="HLTrigger/HLTcore"/>
<flags EDM_PLUGIN="1"/>
<export>
  <lib name="1"/>
</export>
```

Y compilamos con `scram b`,

```
[18:44:47] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src $ scram b
>> Local Products Rules ..... started
>> Local Products Rules ..... done
>> Building CMSSW version CMSSW_5_3_32 ----
>> Entering Package Demo/DemoAnalyzer
>> Creating project symlinks
  src/Demo/DemoAnalyzer/python -> python/Demo/DemoAnalyzer
>> Compiling edm plugin /home/cmsusr/CMSSW_5_3_32/src/Demo/DemoAnalyzer/src/DemoAnalyzer.cc
>> Building edm plugin tmp/slc6_amd64_gcc472/src/Demo/DemoAnalyzer/src/DemoDemoAnalyzer/libDemoDemoAnalyzer.so
Leaving library rule at Demo/DemoAnalyzer
@@@ Running edmWriteConfigs for DemoDemoAnalyzer
--- Registered EDM Plugin: DemoDemoAnalyzer
>> Leaving Package Demo/DemoAnalyzer
>> Package Demo/DemoAnalyzer built
>> Subsystem Demo built
>> Local Products Rules ..... started
>> Local Products Rules ..... done
gmake[1]: Entering directory `/home/cmsusr/CMSSW_5_3_32'
>> Creating project symlinks
  src/Demo/DemoAnalyzer/python -> python/Demo/DemoAnalyzer
>> Done python_symlink
>> Compiling python modules cfipython/slc6_amd64_gcc472
>> Compiling python modules python
>> Compiling python modules src/Demo/DemoAnalyzer/python
>> All python modules compiled
@@@ Refreshing Plugins:edmPluginRefresh
>> Plugging of all type refreshed.
>> Done generating edm plugin poisoned information
gmake[1]: Leaving directory `/home/cmsusr/CMSSW_5_3_32'
[18:45:04] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src $
```

Ahora, modifiquemos el archivo de configuración `Demo/DemoAnalyzer/demoanalyzer_cfg.py` para adaptarlo a nuestro ejercicio. Primero, cambiemos el número de eventos a -1, para que podamos ejecutarlos todos. Además, cambie el archivo `PoolSource`; reemplácelo con un par de archivos de nuestra selección de conjuntos de datos. Además, comente lo que hicimos para extraer la información de los muones y agregar el filtro `HLTHighLevel`, y reemplácelo con los parámetros que necesitamos en la configuración. No olvide notar que estamos llamando a nuestro proceso `mytrigger` ahora, y no `demo`.

Además, asegúrese absolutamente de tener acceso a la información de la base de datos de condiciones necesaria para 2012, que es diferente a la de 2011. Debe reemplazar las tres líneas que tienen `GlobalTag`, pero es diferente al usar:

Una VM,

```
#needed to access the conditions data from the Virtual Machine
process.load('Configuration.StandardSequences.FrontierConditions_GlobalTag_cff')
process.GlobalTag.connect = cms.string('sqlite_file:/cvmfs/cms-opendata-
conddb.cern.ch/FT53_V21A_AN6_FULL.db')
process.GlobalTag.globaltag = 'FT53_V21A_AN6::All'
```

Un contenedor Docker,

```
#needed to access the conditions data from the Docker container
process.load('Configuration.StandardSequences.FrontierConditions_GlobalTag_cff')
process.GlobalTag.connect = cms.string('sqlite_file:/opt/cms-opendata-
conddb/FT53_V21A_AN6_FULL_data_stripped.db')
process.GlobalTag.globaltag = 'FT53_V21A_AN6_FULL::All'
```

Estas líneas, con la cadena `GlobalTag` en ellas, permiten leer información de la base de datos de CMS. Los llamamos los datos de condiciones, ya que podemos encontrar valores para calibración, alineación, preescalas de activación, etc., allí. Se puede pensar en `GlobalTag` como una etiqueta que contiene un conjunto de instantáneas de bases de datos que deben ser adecuadas para un punto en el tiempo en la historia del detector CMS. Para la publicación de datos abiertos de 2012, la etiqueta global es `FT53_V21A_AN6` o `FT53_V21A_AN6_FULL` (la cadena `::All` es una marca que indica a los marcos que lean toda la información asociada con la etiqueta).

La variable `connect` es una de esas líneas que solo modifica la forma en que el marco accederá a estas instantáneas. Para la VM accedemos a ellos a través del área del sistema de archivos compartidos en el CERN (cvmfs). Lee de esta manera, las condiciones que se almacenarán en caché localmente en su máquina virtual la primera vez que ejecute y, por lo tanto, el trabajo de CMSSW será lento.

Por otro lado, en el contenedor de Docker, estas instantáneas de la base de datos viven localmente en su directorio `/opt/cms-opendata-conddb`. Correr sobre ellos es mucho más rápido.

El archivo de configuración final debería ser algo como:

```
GNU nano 2.0.9 File: Demo/DemoAnalyzer/demoanalyzer_cfg.py
# coding=utf-8
import FWCore.ParameterSet.Config as cms
process = cms.Process("Demo")
process.load("FWCore.MessageService.MessageLogger_cfi")
process.MessageLogger.cerr.FwkReport.reportEvery = 1
process.maxEvents = cms.untracked.PSet( input = cms.untracked.int32(-1) )

process.source = cms.Source("PoolSource",
    # replace 'myfile.root' with the source file you want to use
    fileName = cms.untracked.vstring(
        # 'file:myfile.root'
        #root://eospublic.cern.ch//eos/opendata/cms/Run2012B/DoubleMuParked/AOD/22Jan2013-v1/10000/1EC938EF-ABEC-E211-94E0-90E6BA442F24.root'
        'root://eospublic.cern.ch//eos/opendata/cms/Run2012B/TauPlusX/AOD/22Jan2013-v1/20000/0040CF04-8E74-E211-AD0C-00266CFFA344.root',
        'root://eospublic.cern.ch//eos/opendata/cms/Run2012C/TauPlusX/AOD/22Jan2013-v1/310001/0EF85C5C-A787-E211-AFC9-003048C6942A.root'
    )
)
#uncomment to access the conditions data from the Virtual Machine (and comment out the Docker container set below)
#process.load('Configuration.StandardSequences.FrontierConditions_GlobalTag_cff')
#process.GlobalTag.connect = cms.string('sqlite_file:/cvmfs/cms-opendata-conddb.cern.ch/FT53_V21A_AN6_FULL.db')
#process.GlobalTag.globaltag = 'FT53_V21A_AN6::All'

#needed to access the conditions data from the Docker container
process.load('Configuration.StandardSequences.FrontierConditions_GlobalTag_cff')
process.GlobalTag.connect = cms.string('sqlite_file:/opt/cms-opendata-conddb/FT53_V21A_AN6_FULL_data_stripped.db')
process.GlobalTag.globaltag = 'FT53_V21A_AN6_FULL::All'

process.mytrigger = cms.EDAnalyzer("DemoAnalyzer",
    #InputCollection = cms.InputTag("muons")
    processName = cms.string("HLT"),
    datasetName = cms.string("TauPlusX"), #specific dataset example (for dumping info)
)

#process.load("HLTrigger.HLTfilters.hltHighLevel_cfi")
#process.hltHighLevel.HLTPaths = cms.vstring('HLT_Mu15*')

#process.p = cms.Path(process.hltHighLevel+process.demo)
process.p = cms.Path(process.mytrigger)
```

Tenga en cuenta que el nombre del proceso siempre es `HLT` para los datos que se procesaron con el sistema en línea.

Ejecutamos `cmsRun Demo/DemoAnalyzer/demoanalyzer_cfg.py > full_triggerdump.log 2>&1 &` y vemos el resultado con `tail -f full_triggerdump.log`

En nuestro ejemplo, notamos que la configuración cambia para diferentes épocas en el período de toma de datos de 2012: las versiones de Trigger son diferentes. Esto podría significar una modificación de algún parámetro en el Trigger, que no afecte las características principales, pero quizás lo haga más eficiente.

```
Begin processing the 16147th record. Run 200041, Event 386671843, LumiSection 310 at 30-Apr-2022 19:14:14.594 CEST
Begin processing the 16148th record. Run 200041, Event 386758139, LumiSection 310 at 30-Apr-2022 19:14:14.595 CEST
Begin processing the 16149th record. Run 200041, Event 386789251, LumiSection 310 at 30-Apr-2022 19:14:14.596 CEST
Begin processing the 16150th record. Run 200041, Event 386755051, LumiSection 310 at 30-Apr-2022 19:14:14.596 CEST
Begin processing the 16151st record. Run 200041, Event 386885251, LumiSection 310 at 30-Apr-2022 19:14:14.597 CEST
Begin processing the 16152nd record. Run 200041, Event 387006232, LumiSection 310 at 30-Apr-2022 19:14:14.598 CEST
Begin processing the 16153rd record. Run 200041, Event 386964443, LumiSection 310 at 30-Apr-2022 19:14:14.599 CEST
Begin processing the 16154th record. Run 200041, Event 387017632, LumiSection 310 at 30-Apr-2022 19:14:14.600 CEST
Begin processing the 16155th record. Run 200041, Event 386694987, LumiSection 310 at 30-Apr-2022 19:14:14.601 CEST
Begin processing the 16156th record. Run 200041, Event 387053608, LumiSection 310 at 30-Apr-2022 19:14:14.601 CEST
Begin processing the 16157th record. Run 200041, Event 387157048, LumiSection 310 at 30-Apr-2022 19:14:14.602 CEST
Begin processing the 16158th record. Run 200041, Event 387191784, LumiSection 310 at 30-Apr-2022 19:14:14.603 CEST
Begin processing the 16159th record. Run 200041, Event 387201760, LumiSection 310 at 30-Apr-2022 19:14:14.604 CEST
Begin processing the 16160th record. Run 200041, Event 387143944, LumiSection 310 at 30-Apr-2022 19:14:14.604 CEST
Begin processing the 16161st record. Run 200041, Event 386844675, LumiSection 310 at 30-Apr-2022 19:14:14.605 CEST
Begin processing the 16162nd record. Run 200041, Event 387226016, LumiSection 310 at 30-Apr-2022 19:14:14.606 CEST
Begin processing the 16163rd record. Run 200041, Event 387179496, LumiSection 310 at 30-Apr-2022 19:14:14.606 CEST
Begin processing the 16164th record. Run 200041, Event 387172752, LumiSection 310 at 30-Apr-2022 19:14:14.607 CEST
Begin processing the 16165th record. Run 200041, Event 387205944, LumiSection 310 at 30-Apr-2022 19:14:14.608 CEST
Begin processing the 16166th record. Run 200041, Event 387301408, LumiSection 310 at 30-Apr-2022 19:14:14.609 CEST
Begin processing the 16167th record. Run 200041, Event 387320856, LumiSection 310 at 30-Apr-2022 19:14:14.609 CEST
Begin processing the 16168th record. Run 200041, Event 387291336, LumiSection 310 at 30-Apr-2022 19:14:14.609 CEST
Begin processing the 16169th record. Run 200041, Event 387347352, LumiSection 310 at 30-Apr-2022 19:14:14.610 CEST
Begin processing the 16170th record. Run 200041, Event 387353600, LumiSection 310 at 30-Apr-2022 19:14:14.610 CEST
30-Apr-2022 19:14:14 CEST Closed file root://eospublic.cern.ch/eos/opendata/cms/Run2012C/TauPlusX/AOD/22Jan2013-v1/310001/0EF85C5C-A787-E211-AFC9-003048C6942A.root

=====
MessageLogger Summary
type      category      sev      module      subroutine      count      total
-----
1 fileAction      -s file_close      2          2
2 fileAction      -s file_open       4          4

type      category      Examples: run/evt      run/evt      run/evt
-----
1 fileAction      PostProcessEvent      PostEndRun
2 fileAction      pre-events            pre-events      PostProcessEvent

Severity      # Occurrences      Total Occurrences
-----
System        6                    6
```

En conclusión, es necesario inspeccionar los Trigger de esta manera para encontrar un Trigger o un conjunto de Triggers que estén disponibles durante todo el rango de ejecución de interés y asegurarse de que estén disponibles cuando intentemos llamarlos.

Comprensión de los Trigger usando un archivo de configuración del menú Trigger

Podríamos extraer información de los nombres de los Trigger en esa lista. Por ejemplo, el `IsoMu` o `Mu` en los nombres podría indicar la presencia de un muón, y la cadena `LoosePFTau` podría indicar el requisito de una partícula tau.

En <http://opendata.cern.ch/record/1701> encontrara la información de activación para los datos abiertos de CMS de 2012,

High-Level Trigger information for 2012 CMS open data		
CMS collaboration		
Supplementaries Trigger CMS CERN-LHC		
<h3>Description</h3> <p>The list of trigger configuration files for the CMS Run2012B and Run2012C collision data:</p>		
Run number	Software version	Trigger configuration
202970 - 203002	CMSSW_5_2_6_onlpatch3	/cdq/physics/Run2012/7e33/v4.2/HLT/V1
203709	CMSSW_5_2_7_cand3	/cdq/physics/Run2012/7e33/v4.2/HLT/V1
206744 - 206745	CMSSW_5_2_7_onlpatch2	/cdq/physics/Run2012/8e33/v2.0/HLT/V2
204100 - 205238	CMSSW_5_2_7	/cdq/physics/Run2012/8e33/v1.1/HLT/V1
194735 - 194778	CMSSW_5_2_4_onlpatch3	/cdq/physics/Run2012/7e33/v2.3/HLT/V1
194424 - 194712	CMSSW_5_2_4_onlpatch3	/cdq/physics/Run2012/7e33/v2.2/HLT/V3
207779 - 207886	CMSSW_5_2_7_onlpatch3	/cdq/physics/Run2012/8e33/v2.1/HLT/V6

La página presenta los archivos de configuración de HLT CMSSW (conocidos como menús de activación) que se usaron para diferentes rangos de ejecución. Aunque son muy, muy largos, esos archivos de configuración tienen la misma estructura que cualquier archivo de configuración de CMSSW. Es decir, podemos ver cómo se configuró el disparador y verificar todos los parámetros que se usaron para definir su software de trabajo.

Obtener las preescalas y el bit de aceptación

Trabajemos en uno de los ejemplos almacenados en el repositorio de GitHub. Vamos a trabajar con el paquete TriggerSimplePrescalesAnalyzer.

Primero asegúrese de estar en la parte superior de su área `CMSSW_5_3_32/src` y de haber emitido el comando `cmsenv`.

Ahora vamos a clonar la rama de 2011 de este repositorio, que será buena para los datos de 2012 con los que estamos trabajando,

```
[19:42:24] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src $ cmsenv
[19:45:43] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src $ git clone -b 2011 git://github.com/cms-opendata-analyses/TriggerInfoTool.git
Cloning into 'TriggerInfoTool'...
fatal: remote error:
  The unauthenticated git protocol on port 9418 is no longer supported.
Please see https://github.blog/2021-09-01-improving-git-protocol-security-github/ for more information.
[19:46:04] cmsusr@53d3985b25a5 ~/CMSSW_5_3_32/src $
```

En este caso aparece que ya no esta disponible.

Kubernetes

<https://cms-opendata-workshop.github.io/workshop-lesson-kubernetes/>

Operar contenedores a gran escala es complicado incluso los expertos requieren herramientas de orquestación (tienen como propósito el manejo del ciclo de vida de los contenedores).

Kubernetes

Kubernetes elimina los procesos manuales involucrados en la implementación y escalado de aplicaciones en contenedores. En otras palabras, Kubernetes ayuda a administrar clústeres de manera fácil y eficiente.

Kubernetes Cluster

Un Kubernetes Cluster son un montón de nodos que ejecutan sus aplicaciones en contenedores. Cuando se implementa un programa, el clúster distribuye automáticamente el trabajo a los nodos individuales, escalando el clúster según sea necesario.

Kubernetes Pod

Un pod representa un grupo de uno o más contenedores que se ejecutan juntos. Usamos cápsulas para facilitar la comunicación entre nuestros contenedores. Los pods son el componente que se usa en el equilibrio de carga. Se crean nuevas réplicas bajo carga para mantener el clúster en buen estado y nuestra aplicación en funcionamiento.

Kubernetes Job

Un trabajo crea uno o más Pods para realizar una operación particular. El trabajo realiza un seguimiento del progreso general, actualiza su estado con información sobre los pods activos, exitosos y fallidos, y se asegura de que finalice su tarea. Cuando la cantidad especificada de pods se haya completado con éxito, el trabajo está completo.

kubectl

Desde nuestro punto de vista es la cabina para controlar Kubernetes. Desde un punto de vista técnico es un cliente para la API de Kubernetes. Cada operación de Kubernetes está expuesta a un punto final de API. El trabajo principal de kubectl es realizar solicitudes HTTP.

```
$ kubectl get pods
$ kubectl create -f task-to-be-executed.yaml
```

Argo

Es una colección de herramientas de código abierto para ayudar a hacer cosas en Kubernetes. Usaremos los flujos de trabajo de Argo para ejecutar la orquestación de trabajos complejos, incluida la ejecución en serie y en paralelo.

```
$ argo list
$ argo submit workflow-to-be-executed.yaml
```

Autoescalado y cómo ahorra dinero

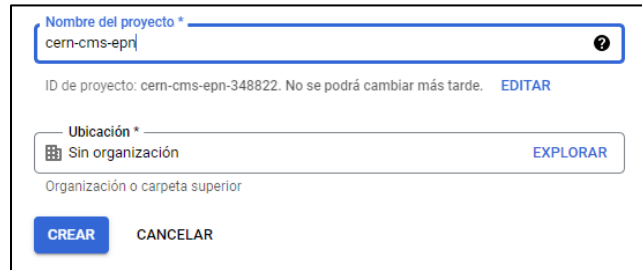
Kubernetes admite el ajuste de escala automático para optimizar los recursos de sus nodos, ajustar la CPU y la memoria para cumplir con el uso real de su aplicación y solo pagas por lo que usas.

Limitado por el ancho de banda

Los datos deben estar donde usted está. En lugar de transmitir datos desde el CERN, cópielos en su espacio de trabajo.

Crear tu proyecto en Google Cloud Platform

Después de ingresar a GCP, aceptamos los términos de servicio, no tenemos que aceptar la prueba gratuita, es decir, denegamos y creamos un nuevo proyecto.



Nombre del proyecto *
cern-cms-epr

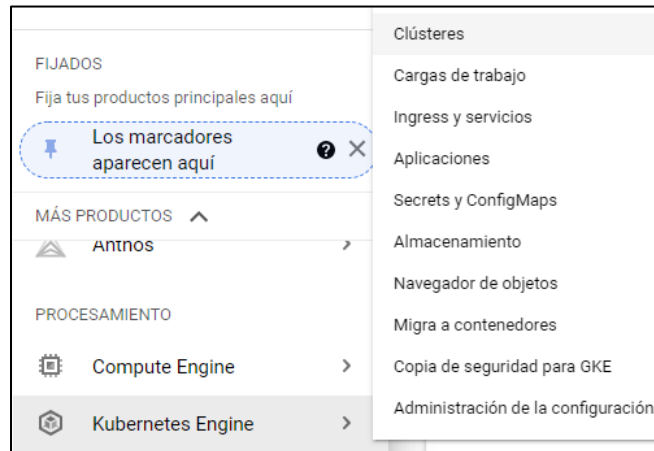
ID de proyecto: cern-cms-epr-348822. No se podrá cambiar más tarde. [EDITAR](#)

Ubicación *
Sin organización [EXPLORAR](#)

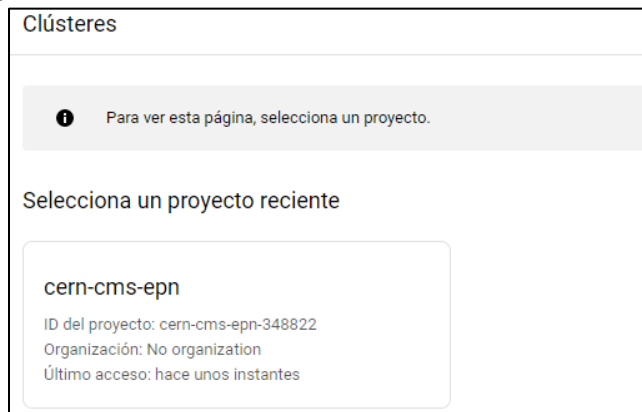
Organización o carpeta superior

[CREAR](#) [CANCELAR](#)


Creamos un clúster haciendo clic en,



Seleccionamos el proyecto creado recientemente,



Habilitamos el Kubernetes Engine API,



Kubernetes Engine API

Google Enterprise API

Builds and manages container-based applications, powered by the open source Kubernetes technology.

[HABILITAR](#) [PROBAR ESTA API](#)

En GCP se debe tener una cuenta de facturación,

Habilitar facturación para el proyecto "cern-cms-eqn"

No eres el administrador de ninguna cuenta de facturación. Para habilitar la facturación en este proyecto, crea una nueva cuenta de facturación o comunícate con el administrador de tu cuenta de facturación para que la habilite por ti. [Más información](#)

[CANCELAR](#) [CREAR CUENTA DE FACTURACIÓN](#)

Creamos un Clústeres de Kubernetes,

Kubernetes Engine

Clústeres de Kubernetes


Los contenedores crean un paquete de aplicación, de manera que se pueda implementar con facilidad para que se ejecute en su propio entorno aislado. Los contenedores se ejecutan en clústeres de Kubernetes. [Más información](#)

[CREAR](#) [IMPLEMENTAR CONTENEDOR](#) [USAR EL INICIO RÁPIDO](#)


De tipo estándar,

Crear clúster

Selecciona el modo de clúster que quieres usar.




Compara los modos de clúster para obtener más información sobre sus diferencias.
[COMPARAR](#)



GKE Standard
Un clúster de Kubernetes de pago por nodo en el que configuras y administras los nodos.
[Más información](#)

[CONFIGURAR](#)



GKE Autopilot
Un clúster de Kubernetes de pago por Pod en el que GKE administra los nodos con la configuración mínima requerida.
[Más información](#)

[CONFIGURAR](#)

Damos un nombre,

- Aspectos básicos del clúster

GRUPOS DE NODOS

- default-pool

CLÚSTER

- Automatización
- Redes
- Seguridad
- Metadatos
- Características

Aspectos básicos del clúster

El clúster nuevo se creará con el nombre, la versión y la ubicación que especifiques aquí. El nombre y la ubicación no se podrán cambiar después de que se cree el clúster.

Para experimentar con un clúster asequible, prueba [Mi primer clúster](#) en la Guía de configuración de clústeres

Nombre

Tipo de ubicación

Los precios de los recursos pueden variar entre regiones determinadas. [Más información](#)

☒ Zonal
☐ Regional

Zona

En el apartado de default-pool, elegimos:

- tamaño: 1 nodo
- Habilitamos el Escalado automático de 0 a 4

- Aspectos básicos del clúster

GRUPOS DE NODOS

- default-pool
 - Nodos
 - Seguridad
 - Metadatos

CLÚSTER

- Automatización
- Redes
- Seguridad
- Metadatos
- Características

Detalles del grupo de nodos

El clúster nuevo se creará con al menos un grupo de nodos. Un grupo de nodos es una plantilla para los conjuntos de nodos creados en este clúster. Después de la creación del clúster, se pueden agregar y quitar más grupos de nodos.

Nombre

Versión del plano de control: 1.21.10-gke.2000

Tamaño

Cantidad de nodos *

El rango de direcciones del pod limita el tamaño máximo del clúster. [Más información](#)

☒ Habilitar el ajuste de escala automático

Cantidad mínima de nodos *

Cantidad máxima de nodos *

☐ Especificar las ubicaciones de los nodos

Vamos a Nodos, cambiamos a una máquina de e2-standar-4 y clic en crear.

- Aspectos básicos del clúster

GRUPOS DE NODOS

- default-pool
 - Nodos
 - Seguridad
 - Metadatos

CLÚSTER

- Automatización
- Redes
- Seguridad
- Metadatos
- Características

Tipo de imagen

Container-Optimized OS con containerd (cos_containerd) (predeterminado)

La imagen de nodo predeterminada de Linux para clústeres y grupos de nodos recién creados con la versión 1.21.10-gke.2000 o posterior es Container-Optimized OS con Containerd. Para los grupos de nodos de Windows que usan la versión 1.21 o posterior, Containerd también es el entorno de ejecución recomendado. [GKE dará de baja las imágenes de nodo de Docker](#) debido a que el proyecto de Kubernetes dará de baja Dockershim. Te recomendamos que [migres a las imágenes de nodo de Containerd](#) lo antes posible. Obtén más información sobre las diferentes [imágenes de nodo](#).

Configuración de la máquina

Familia de máquinas

USO GENERAL OPTIMIZADA PARA PROCESAMIENTO CON OPTIMIZACIÓN DE MEMORIA GPU

Tipos de máquinas para cargas de trabajo comunes, optimizados en función del costo y la flexibilidad

Serie

E2

Selección de la plataforma de CPU según la disponibilidad

Tipo de máquina

e2-standard-4 (4 CPU virtuales, 16 GB de memoria)

CREAR

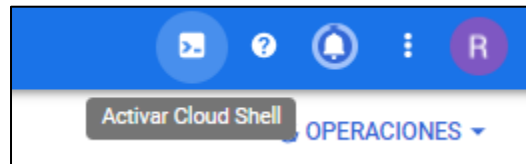
CANCELAR

REST

LÍNEA DE COMANDOS

equivalente

Mientras se está creando podemos abrir el Cloud Shell.



Cloud Shell

GCP proporciona una máquina de acceso para que pueda interactuar con sus diferentes servicios, incluido nuestro clúster K8s recién creado. Esta máquina (y la terminal) no es realmente parte del clúster. Como se dijo, es un punto de entrada. Desde aquí puede conectarse a su clúster.

Comando gcloud

La interfaz de línea de comandos de gcloud es la principal herramienta CLI para crear y administrar recursos de Google Cloud. Puede usar esta herramienta para realizar muchas tareas comunes de la plataforma, ya sea desde la línea de comandos o en scripts y otras automatizaciones.

Podemos usar `gcloud auth login`, confirmando el acceso e introduciendo el código de validación.

```
CLOUD SHELL
Terminal (cern-cms-epn-348822) x + v

Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to cern-cms-epn-348822.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
piedraenrique591@cloudshell:~ (cern-cms-epn-348822) $ gcloud auth login

You are already authenticated with gcloud when running
inside the Cloud Shell and so do not need to run this
command. Do you wish to proceed anyway?

Do you want to continue (Y/n)? n
```

Conectarse a su clúster

Cuando el clúster esté listo. Clique en el botón conectar del clúster. Copie el comando.

Conéctate al clúster

Puedes realizar la conexión al clúster mediante la línea de comandos o con un panel.

Acceso a la línea de comandos

Configura el acceso a la línea de comandos de [kubecti](#) ejecutando el siguiente comando:

```
$ gcloud container clusters get-credentials epn-cluster --zone us-central1-c --project cern-cms-epn-348822
```

[EJECUTAR EN CLOUD SHELL](#)

Panel de Cloud Console

Puedes ver las cargas de trabajo que se encuentran en ejecución en tu clúster en el [panel de cargas de trabajo](#) de Cloud Console.

[ABRIR EL PANEL DE CARGAS DE TRABAJO](#)

[ACEPTAR](#)

Ejecute ese comando en el shell de la nube, autorizamos al cloud Shell.

```
piedraenrique591@cloudshell:~ (cern-cms-epn-348822) $ gcloud container clusters get-credentials epn-cluster --zone us-central1-c --project cern-cms-epn-348822
Fetching cluster endpoint and auth data.
kubeconfig entry generated for epn-cluster.
piedraenrique591@cloudshell:~ (cern-cms-epn-348822) $
```

Kubecti, herramientas y servicios adicionales

Así como `gcloud` es el único comando para gobernarlos a todos para GCP, Kubernetes proporciona una herramienta de línea de comandos para comunicarse con un clúster de Kubernetes plano de control, mediante la API de Kubernetes, llamado `kubecti`. Lo usará para hacer esencialmente cualquier cosa en el clúster.

Veamos algunos ejemplos para:

Obtener el estado de los nodos en su clúster:

```
kubecti get nodes
```

Obtenga la información del clúster (muestra las direcciones del maestro y los servicios),

```
kubecti cluster-info
```

También podemos listar algunos componentes de Kubernetes para:

Comprobar pods,

```
kubectl get pod
```

Consulta los servicios,

```
kubectl get services
```

Cuando se crea algunos componentes podemos:

Inspeccionar la operación de creación,

```
kubectl create -h
```

Vamos a crear una aplicación,

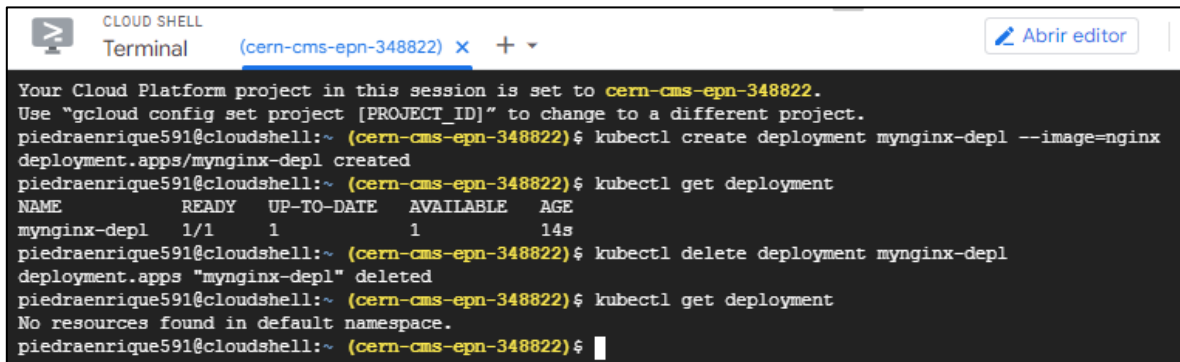
```
kubectl create deployment mynginx-depl --image=nginx
```

Revisar las implementaciones,

```
kubectl get deployment
```

Eliminar la aplicación,

```
kubectl delete deployment mynginx-depl
```



The screenshot shows a Cloud Shell terminal window with the following content:

```
CLOUD SHELL
Terminal (cern-cms-epn-348822) x + v
Your Cloud Platform project in this session is set to cern-cms-epn-348822.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
piedraenrique591@cloudshell:~ (cern-cms-epn-348822) $ kubectl create deployment mynginx-depl --image=nginx
deployment.apps/mynginx-depl created
piedraenrique591@cloudshell:~ (cern-cms-epn-348822) $ kubectl get deployment
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
mynginx-depl  1/1     1            1           14s
piedraenrique591@cloudshell:~ (cern-cms-epn-348822) $ kubectl delete deployment mynginx-depl
deployment.apps "mynginx-depl" deleted
piedraenrique591@cloudshell:~ (cern-cms-epn-348822) $ kubectl get deployment
No resources found in default namespace.
piedraenrique591@cloudshell:~ (cern-cms-epn-348822) $
```

Archivos Yaml

Otra forma de crear componentes en un clúster K8s es a través de archivos yaml. Estos son intuitivos, lógicos y configurables, aunque muy exigentes con la identificación.

Echemos un vistazo a uno de estos archivos, <https://gitlab.com/nanuchi/youtube-tutorial-series/-/raw/master/kubernetes-configuration-file-explained/nginx-deployment.yaml>

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.16
          ports:
            - containerPort: 8080

```

Podemos descargarlo con `wget`,

```

cloudshell:~ (cern-cms-epn-348822) $ wget https://gitlab.com/nanuchi/youtube-tutorial-series/-/raw/master/kubernetes-configuration-file-explained/nginx-deployment.yaml
--2022-05-01 16:15:07-- https://gitlab.com/nanuchi/youtube-tutorial-series/-/raw/master/kubernetes-configuration-file-explained/nginx-deployment.yaml
l-series/-/raw/master/kubernetes-configuration-file-explained/nginx-deployment.yaml
ab.com)... 172.65.251.78, 2606:4700:90:0:f22e:fbec:5bed:a9b9
Connecting to gitlab.com (gitlab.com)|172.65.251.78|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 341 [text/plain]
Saving to: 'nginx-deployment.yaml.1'

nginx-deployment.yaml.1      100%[=====]
==>] 341 --.-KB/s   in 0s

2022-05-01 16:15:07 (6.42 MB/s) - 'nginx-deployment.yaml.1' saved [341/341]

```

Y ver el contenido con `cat`,

```

piedraenrique591@cloudshell:~ (cern-cms-epn-348822) $ cat nginx-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.16
          ports:
            - containerPort: 8080
piedraenrique591@cloudshell:~ (cern-cms-epn-348822) $

```

Creamos una aplicación, `kubectl create -f nginx-deployment.yaml`

```

piedraenrique591@cloudshell:~ (cern-cms-epn-348822) $ kubectl create -f nginx-deployment.yaml
deployment.apps/nginx-deployment created
piedraenrique591@cloudshell:~ (cern-cms-epn-348822) $

```

Implementar las configuraciones usando archivos yaml, `kubectl apply -f nginx-deployment.yaml`

```

piedraenrique591@cloudshell:~ (cern-cms-epn-348822) $ kubectl apply -f nginx-deployment.yaml
Warning: resource deployments/nginx-deployment is missing the kubect1.kubernetes.io/last-applied-configuration annotation which is required by kubect1 apply. kubect1 apply should only be used on resources created declaratively by either kubect1 create --save-config or kubect1 apply. The missing annotation will be patched automatically.
deployment.apps/nginx-deployment configured
piedraenrique591@cloudshell:~ (cern-cms-epn-348822) $

```


Instalar Argo como motor de flujo de trabajo

Los trabajos se pueden ejecutar manualmente pero un motor de flujo de trabajo facilita la definición y el envío de trabajos.

Para instalar Argo ejecutamos los siguientes comandos,

```
kubectl create ns argo
kubectl apply -n argo -f
https://raw.githubusercontent.com/argoproj/argo/stable/manifests/quick-start-postgres.yaml
curl -sLO https://github.com/argoproj/argo/releases/download/v2.11.1/argo-linux-amd64.gz
gunzip argo-linux-amd64.gz
chmod +x argo-linux-amd64
sudo mv ./argo-linux-amd64 /usr/local/bin/argo
```

```
piedraenrique591@cloudshell:~ (cern-cms-epn-348822) $ kubectl create ns argo
namespace/argo created
piedraenrique591@cloudshell:~ (cern-cms-epn-348822) $ kubectl apply -n argo -f https://raw.githubusercontent.com/argoproj/argo/stable/manifests/quick-start-postgres.yaml
customresourcedefinition.apiextensions.k8s.io/clusterworkflowtemplates.argoproj.io created
customresourcedefinition.apiextensions.k8s.io/cronworkflows.argoproj.io created
customresourcedefinition.apiextensions.k8s.io/workfloweventbindings.argoproj.io created
customresourcedefinition.apiextensions.k8s.io/workflows.argoproj.io created
customresourcedefinition.apiextensions.k8s.io/workflowtemplates.argoproj.io created
serviceaccount/argo created
serviceaccount/argo-server created
serviceaccount/github.com created
role.rbac.authorization.k8s.io/argo-role created
role.rbac.authorization.k8s.io/argo-server-role created
role.rbac.authorization.k8s.io/submit-workflow-template created
role.rbac.authorization.k8s.io/workflow-role created
clusterrole.rbac.authorization.k8s.io/argo-clusterworkflowtemplate-role created
clusterrole.rbac.authorization.k8s.io/argo-server-clusterworkflowtemplate-role created
clusterrole.rbac.authorization.k8s.io/kubelet-executor created
rolebinding.rbac.authorization.k8s.io/argo-binding created
rolebinding.rbac.authorization.k8s.io/argo-server-binding created
rolebinding.rbac.authorization.k8s.io/github.com created
rolebinding.rbac.authorization.k8s.io/workflow-default-binding created
clusterrolebinding.rbac.authorization.k8s.io/argo-clusterworkflowtemplate-role-binding created
clusterrolebinding.rbac.authorization.k8s.io/argo-server-clusterworkflowtemplate-role-binding created
clusterrolebinding.rbac.authorization.k8s.io/kubelet-executor-default created
configmap/artifact-repositories created
configmap/workflow-controller-configmap created
secret/argo-postgres-config created
piedraenrique591@cloudshell:~ (cern-cms-epn-348822) $ curl -sLO https://github.com/argoproj/argo/releases/download/v2.11.1/argo-linux-amd64.gz
piedraenrique591@cloudshell:~ (cern-cms-epn-348822) $ gunzip argo-linux-amd64.gz
piedraenrique591@cloudshell:~ (cern-cms-epn-348822) $ chmod +x argo-linux-amd64
piedraenrique591@cloudshell:~ (cern-cms-epn-348822) $ sudo mv ./argo-linux-amd64 /usr/local/bin/argo
piedraenrique591@cloudshell:~ (cern-cms-epn-348822) $
```

Podemos validar la instalación ejecutando lo siguiente,

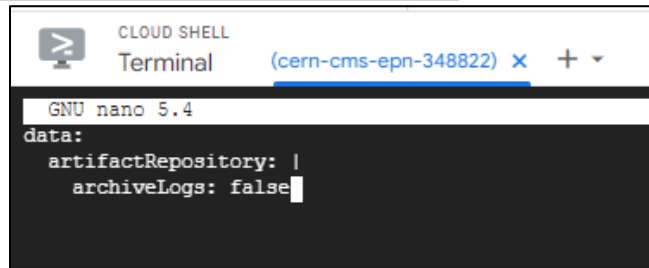
```
argo version
argo submit -n argo --watch https://raw.githubusercontent.com/argoproj/argo-workflows/master/examples/hello-world.yaml
argo list -n argo
argo get -n argo @latest
argo logs -n argo @latest
```

Nota: En caso de que deje su computadora por mucho tiempo, es posible que deba volver a conectarse a CloudShell. Si no se encuentra el comando `argo`, vuelva a ejecutar el comando anterior a partir del comando `curl`.

```
kubectl create clusterrolebinding YOURNAME-cluster-admin-binding --clusterrole=cluster-admin --
user=YOUREMAIL@gmail.com
```

```
piedraenrique591@cloudshell:~$ (cern-cms-epn-348622)$ kubectl create clusterrolebinding Enrique-cluster-admin-binding --clusterrole=cluster-admin --user=piedraenrique591@gmail.com
clusterrolebinding.rbac.authorization.k8s.io/enrique-cluster-admin-binding created
piedraenrique591@cloudshell:~$ (cern-cms-epn-348622)$
```

Necesitamos aplicar un pequeño parche a la configuración predeterminada de argo. Creamos un archivo llamado `patch-workflow-controller-configmap.yaml` y agregamos lo siguiente,



Luego ejecutamos, `kubectl patch configmap workflow-controller-configmap -n argo --patch "$(cat patch-workflow-controller-configmap.yaml)"`,

```
piedraenrique591@cloudshell:~ (cern-cms-epn-348822) $ kubectl patch configmap workflow-controller-configmap -n argo --patch "$(cat patch-workflow-controller-configmap.yaml)"
configmap/workflow-controller-configmap patched
piedraenrique591@cloudshell:~ (cern-cms-epn-348822) $
```

Definición de un volumen de almacenamiento

Es necesario montar el almacenamiento disponible en un Pod para poder acceder a él. Kubernetes proporciona una gran cantidad de tipos de almacenamiento.

Sabemos que los trabajos de análisis de datos producirán archivos de salida entonces ahora, seguiremos algunos pasos para configurar un volumen donde se escribirán los archivos de salida y desde donde se pueden recuperar. Todas las definiciones se pasan como archivos "yaml". Debido a algunas restricciones de GKE (Google Kubernetes Engine), necesitamos usar volúmenes persistentes NFS (Network FileSystem) para permitir el acceso paralelo.

Creamos un volume, `gcloud compute disks create --size=100GB --zone=us-central1-c gce-nfs-disk-
<NUMBER>`, `zone` debe ser la misma que la del clúster creado.

Primero configuramos un servidor nfs para este disco, `wget https://cms-opendata-workshop.github.io/workshop2021-lesson-cloud/files/001-nfs-server.yaml` y luego `kubectl apply -n argo -f 001-nfs-server.yaml`

```
piedraenrique591@cloudshell:~ (cern-cms-epn-348822) $ gcloud compute disks create --size=100GB --zone=us-central1-c gce-nfs-disk-1
WARNING: You have selected a disk size of under [200GB]. This may result in poor I/O performance. For more information, see: https://developers.google.com/compute/docs/disks#performance.
Created [https://www.googleapis.com/compute/v1/projects/cern-cms-epn-348822/zones/us-central1-c/disks/gce-nfs-disk-1].
NAME: gce-nfs-disk-1
ZONE: us-central1-c
SIZE_GB: 100
TYPE: pd-standard
STATUS: READY

New disks are unformatted. You must format and mount a disk before it can be used. You can find instructions on how to do this at:
https://cloud.google.com/compute/docs/disks/add-persistent-disk#formatting

piedraenrique591@cloudshell:~ (cern-cms-epn-348822) $ wget https://cms-opendata-workshop.github.io/workshop2021-lesson-cloud/files/001-nfs-server.yaml
--2022-05-01 17:26:10-- https://cms-opendata-workshop.github.io/workshop2021-lesson-cloud/files/001-nfs-server.yaml
Resolving cms-opendata-workshop.github.io (cms-opendata-workshop.github.io)... 185.199.109.153, 185.199.108.153, 185.199.111.153, ...
Connecting to cms-opendata-workshop.github.io (cms-opendata-workshop.github.io)|185.199.109.153|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 724 [text/yaml]
Saving to: '001-nfs-server.yaml'

001-nfs-server.yaml           100%[=====>] 724 --.-KB/s  in 0s

2022-05-01 17:26:10 (91.9 MB/s) - '001-nfs-server.yaml' saved [724/724]

piedraenrique591@cloudshell:~ (cern-cms-epn-348822) $ kubectl apply -n argo -f 001-nfs-server.yaml
replicationcontroller/nfs-server created
piedraenrique591@cloudshell:~ (cern-cms-epn-348822) $
```

Luego, configuramos un servicio nfs, para que podamos acceder al servidor, `wget https://cms-opendata-workshop.github.io/workshop2021-lesson-cloud/files/002-nfs-server-service.yaml` y `kubectl apply -n argo -f 002-nfs-server-service.yaml`

```
pi@cloudshell:~ (cern-cms-epn-348822) $ wget https://cms-opendata-workshop.github.io/workshop2021-lesson-cloud/files/002-nfs-server-service.yaml
--2022-05-01 17:53:42-- https://cms-opendata-workshop.github.io/workshop2021-lesson-cloud/files/002-nfs-server-service.yaml
Resolving cms-opendata-workshop.github.io (cms-opendata-workshop.github.io)... 185.199.108.153, 185.199.110.153, 185.199.109.153, ...
Connecting to cms-opendata-workshop.github.io (cms-opendata-workshop.github.io)|185.199.108.153|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 212 [text/yaml]
Saving to: '002-nfs-server-service.yaml'

002-nfs-server-service.yaml      100%[=====>] 212 --.-KB/s  in 0s

2022-05-01 17:53:42 (10.0 MB/s) - '002-nfs-server-service.yaml' saved [212/212]

pi@cloudshell:~ (cern-cms-epn-348822) $ kubectl apply -n argo -f 002-nfs-server-service.yaml
service/nfs-server created
pi@cloudshell:~ (cern-cms-epn-348822) $
```

Mostremos la IP del servidor nfs, `kubectl get -n argo svc nfs-server-<NUMBER> | grep ClusterIP | awk '{ print $3; }'`

```
pi@cloudshell:~ (cern-cms-epn-348822) $ kubectl get -n argo svc nfs-server | grep ClusterIP | awk '{ print $3; }'
185.199.108.153
pi@cloudshell:~ (cern-cms-epn-348822) $
```

Finalmente, vamos a crear un volumen persistente a partir de este disco nfs. Tenga en cuenta que los volúmenes persistentes no tienen espacios de nombres, están disponibles para todo el clúster. Necesitamos escribir la IP del servidor en el lugar apropiado en este archivo, para ello `wget https://cms-opendata-workshop.github.io/workshop2021-lesson-cloud/files/003-pv.yaml`,

```
GNU nano 5.4 003-pv.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: nfs-1
spec:
  capacity:
    storage: 100Gi
  accessModes:
    - ReadWriteMany
  nfs:
    server: 185.199.108.153
    path: "/"
```

Luego `kubectl apply -f 003-pv.yaml`,

```
pi@cloudshell:~ (cern-cms-epn-348822) $ wget https://cms-opendata-workshop.github.io/workshop2021-lesson-cloud/files/003-pv.yaml
--2022-05-01 18:06:22-- https://cms-opendata-workshop.github.io/workshop2021-lesson-cloud/files/003-pv.yaml
Resolving cms-opendata-workshop.github.io (cms-opendata-workshop.github.io)... 185.199.108.153, 185.199.110.153, 185.199.109.153, ...
Connecting to cms-opendata-workshop.github.io (cms-opendata-workshop.github.io)|185.199.108.153|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 181 [text/yaml]
Saving to: '003-pv.yaml'

003-pv.yaml      100%[=====>] 181 --.-KB/s  in 0s

2022-05-01 18:06:22 (21.9 MB/s) - '003-pv.yaml' saved [181/181]

pi@cloudshell:~ (cern-cms-epn-348822) $ nano 003-pv.yaml
pi@cloudshell:~ (cern-cms-epn-348822) $ kubectl apply -f 003-pv.yaml
persistentvolume/nfs-1 created
pi@cloudshell:~ (cern-cms-epn-348822) $
```

Confirmemos que esto funcionó,

```
pi@cloudshell:~ (cern-cms-epn-348822) $ kubectl get pv
NAME      CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS  CLAIM  STORAGECLASS  REASON  AGE
nfs-1     100Gi     RWX           Retain          Available              storageclass  2m12s
pi@cloudshell:~ (cern-cms-epn-348822) $
```

Puede pasar algún tiempo antes de que STATUS llegue al estado "Bound".

```
piedraenrique591@cloudshell:~ (cern-cms-epn-348822) $ kubectl get pv
NAME      CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS  CLAIM      STORAGECLASS  REASON  AGE
nfs-1     100Gi     RWX           Retain          Bound   argo/nfs-1  argo/nfs-1    9m7s
```

Las aplicaciones pueden reclamar volúmenes persistentes a través de pvc (persistent volume claims). Vamos a crear un pvc, `wget https://cms-opendata-workshop.github.io/workshop2021-lesson-cloud/files/003-pvc.yaml`, `kubectl apply -n argo -f 003-pvc.yaml`

```
piedraenrique591@cloudshell:~ (cern-cms-epn-348822) $ wget https://cms-opendata-workshop.github.io/workshop2021-lesson-cloud/files/003-pvc.yaml
--2022-05-01 18:20:56-- https://cms-opendata-workshop.github.io/workshop2021-lesson-cloud/files/003-pvc.yaml
Resolving cms-opendata-workshop.github.io (cms-opendata-workshop.github.io)... 185.199.108.153, 185.199.109.153, 185.199.110.153, ...
Connecting to cms-opendata-workshop.github.io (cms-opendata-workshop.github.io)|185.199.108.153|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 194 [text/yaml]
Saving to: '003-pvc.yaml'

003-pvc.yaml                               100%[=====>] 194 --.-KB/s  in 0s

2022-05-01 18:20:56 (23.4 MB/s) - '003-pvc.yaml' saved [194/194]

piedraenrique591@cloudshell:~ (cern-cms-epn-348822) $ kubectl apply -n argo -f 003-pvc.yaml
persistentvolumeclaim/nfs-1 created
piedraenrique591@cloudshell:~ (cern-cms-epn-348822) $
```

Revisamos con `kubectl get pvc nfs-<NUMBER> -n argo`

```
piedraenrique591@cloudshell:~ (cern-cms-epn-348822) $ kubectl get pvc -n argo
NAME      STATUS  VOLUME  CAPACITY  ACCESS MODES  STORAGECLASS  AGE
nfs-1     Bound   nfs-1   100Gi     RWX            argo/nfs-1    98s
piedraenrique591@cloudshell:~ (cern-cms-epn-348822) $
```

Terraform

`kubectl apply -n argo -f https://raw.githubusercontent.com/argoproj/argo/stable/manifests/quick-start-postgres.yaml`

`curl -sLO https://github.com/argoproj/argo/releases/download/v2.11.1/argo-linux-amd64.gz`

`gunzip argo-linux-amd64.gz`

`chmod +x argo-linux-amd64`

`sudo mv ./argo-linux-amd64 /usr/local/bin/argo`

Create a workflow definition file `argo-wf-volume.yaml`

```
piedraenrique591@cloudshell:~ (cern-cms-epn-348822) $ nano argo-wf-volume.yaml
piedraenrique591@cloudshell:~ (cern-cms-epn-348822) $ argo submit -n argo argo-wf-volume.yaml
argo list -n argo
-bash: argo: command not found
-bash: argo: command not found
```

```
piedraenrique591@cloudshell:~ (cern-cms-epn-348822) $ kubectl apply -n argo -f https://raw.githubusercontent.com/argoproj/argo/stable/manifests/quick-start-postgres.yaml
curl -sLO https://github.com/argoproj/argo/releases/download/v2.11.1/argo-linux-amd64.gz
gunzip argo-linux-amd64.gz
chmod +x argo-linux-amd64
sudo mv ./argo-linux-amd64 /usr/local/bin/argo
```

```

piedraenrique591@cloudshell:~ (cern-cms-eqn-348822) $ argo submit -n argo argo-wf-volume.yaml
argo list -n argo
Name:          test-hostpath-fqfjn
Namespace:     argo
ServiceAccount: default
Status:        Pending
Created:       Fri May 13 02:32:39 +0000 (now)
NAME          STATUS    AGE    DURATION    PRIORITY
test-hostpath-fqfjn  Running  0s     0s          0
nanoaod-argo-9dctr  Running  25m    25m         0
test-hostpath-ss9mz  Running  30m    30m         0
hello-world-6gkt2    Running  11d    11d         0
piedraenrique591@cloudshell:~ (cern-cms-eqn-348822) $ kubectl logs pod/test-hostpath-fqfjn -n argo main

ls -l /mnt/vol: total 24 -rw-r--r-- 1 root root 16 May 13 02:26 index.html drwx----- 2 root root 16384 May 13 02:26 lost+found -rw-r--r-- 1 root root
18 May 13 02:32 test.txt
piedraenrique591@cloudshell:~ (cern-cms-eqn-348822) $ argo list -n argo
NAME          STATUS    AGE    DURATION    PRIORITY
test-hostpath-fqfjn  Running  1m     1m          0
nanoaod-argo-9dctr  Running  26m    26m         0
test-hostpath-ss9mz  Running  32m    32m         0
hello-world-6gkt2    Running  11d    11d         0
piedraenrique591@cloudshell:~ (cern-cms-eqn-348822) $ ||

```

Run a CMS open data workflow

Create a workflow file argo-wf-cms.yaml

```

Name:          nanoaod-argo-g92t8
Namespace:     argo
ServiceAccount: default
Status:        Running
Conditions:
  PodRunning    True
Created:       Fri May 13 02:38:07 +0000 (13 minutes ago)
Started:       Fri May 13 02:38:07 +0000 (13 minutes ago)
Duration:      13 minutes 2 seconds

STEP          TEMPLATE    PODNAME          DURATION  MESSAGE
• nanoaod-argo-g92t8  nanoaod-argo  nanoaod-argo-g92t8  13m
|

```

```

piedraenrique591@cloudshell:~ (cern-cms-eqn-348822) $ argo get -n argo @latest
Name:          nanoaod-argo-g92t8
Namespace:     argo
ServiceAccount: default
Status:        Running
Conditions:
  PodRunning    True
Created:       Fri May 13 02:38:07 +0000 (6 minutes ago)
Started:       Fri May 13 02:38:07 +0000 (6 minutes ago)
Duration:      6 minutes 19 seconds

STEP          TEMPLATE    PODNAME          DURATION  MESSAGE
• nanoaod-argo-g92t8  nanoaod-argo  nanoaod-argo-g92t8  6m
piedraenrique591@cloudshell:~ (cern-cms-eqn-348822) $ kubectl logs pod/nanoaod-argo-g92t8 -n argo main
Setting up CMSSW_5_3_32
WARNING: In non-interactive mode release checks e.g. deprecated releases, production architectures are disabled.
CMSSW should now be available.
fatal: remote error:
  The unauthenticated git protocol on port 9418 is no longer supported.
Please see https://github.blog/2021-09-01-improving-git-protocol-security-github/ for more information.
Cloning into 'AOD2NanoAOD'...
piedraenrique591@cloudshell:~ (cern-cms-eqn-348822) $ ||

```

```
piedraenrique591@cloudshell:~ (cern-cms-eqn-348822)$ gcloud compute disks list
NAME: gce-nfs-disk-1
LOCATION: us-central1-c
LOCATION_SCOPE: zone
SIZE_GB: 100
TYPE: pd-standard
STATUS: READY

NAME: gce-nfs-disk-2
LOCATION: us-central1-c
LOCATION_SCOPE: zone
SIZE_GB: 100
TYPE: pd-standard
STATUS: READY

NAME: gke-eqn-cluster-default-pool-0750cf91-cp8h
LOCATION: us-central1-c
LOCATION_SCOPE: zone
SIZE_GB: 100
TYPE: pd-standard
STATUS: READY
```