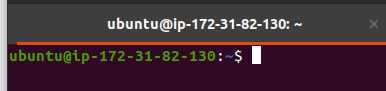
**Como instalar y ejecutar la app “Grapes”**

1. Ingresar a nuestra máquina AWS



1. Preparación para usar Flask con Gunicorn ([Información aquí](https://www.digitalocean.com/community/tutorials/como-preparar-aplicaciones-de-flask-con-gunicorn-y-nginx-en-ubuntu-18-04-es))
2. Actualizar paquetes locales y e instalar paquetes para el entorno

*sudo apt update*

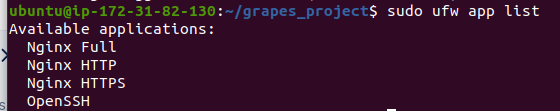
*sudo apt install python3-pip python3-dev build-essential libssl-dev libffi-dev python3-setuptools*

1. Instalar **NGINX**

*sudo apt install nginx*

Y ajustar el firewall. Vemos apps disponibles

*sudo ufw app list*



Activamos

*sudo ufw allow 'Nginx HTTP'*



y verificamos el cambio

*sudo ufw status*



si el status sigue inactivo como en este caso, activarlo usando

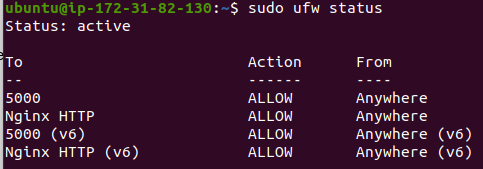
*sudo ufw enable*

**

**IMPORTANTE: Revisar porque el firewall bloquea todas las conecciones**

ahora si verificamos el status

*sudo ufw status*



1. En caso de que algo falle y se bloqueen los puertos (en este caso sucedió), activar los mismos usando, [revisar aquí](https://www.configserverfirewall.com/ufw-ubuntu-firewall/ubuntu-firewall-open-port/).

*sudo ufw allow XX*

XX es el número de puerto

o también por defecto abriendo el puerto 22 con

*sudo ufw allow ssh*

1. Crear un directorio para nuestro proyecto, en este caso llamado ***grapes\_project***

*mkdir grapes\_project*

1. Instalar el paquete ***phyton3-venv,*** para crear entornos

*sudo apt install python3-venv*

1. Ingresar a nuestro nuevo directorio de trabajo *grapes\_project*  y crear un *enviroment* o entorno *(en este caso llamado* ***env****)*

*python3 -m venv env*

**

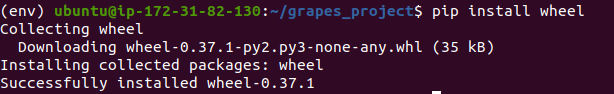
1. Activamos el nuevo entorno

*source env/bin/activate*



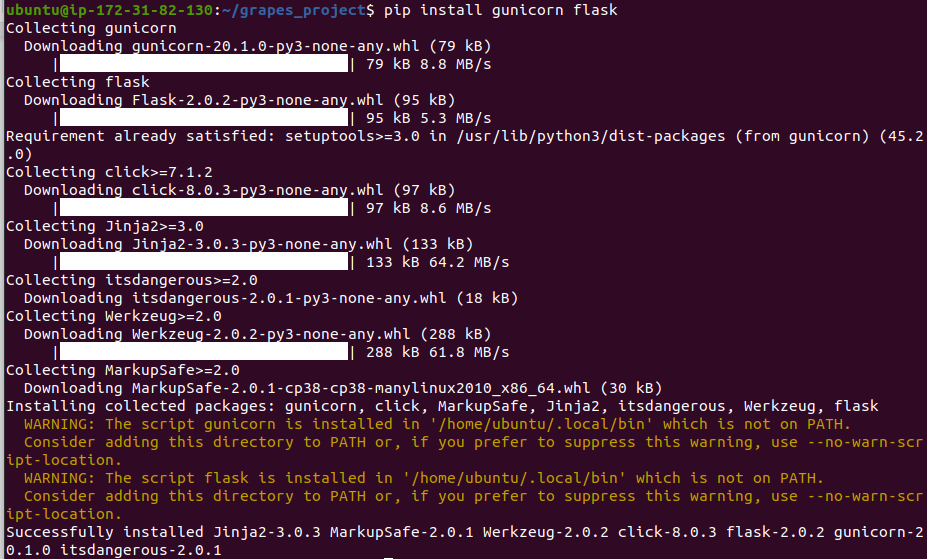
1. Ahora instalamos **Flask** y **Gunicorn** en nuestro entorno, pero antes necesitamos instalar *wheel*

*pip install wheel*



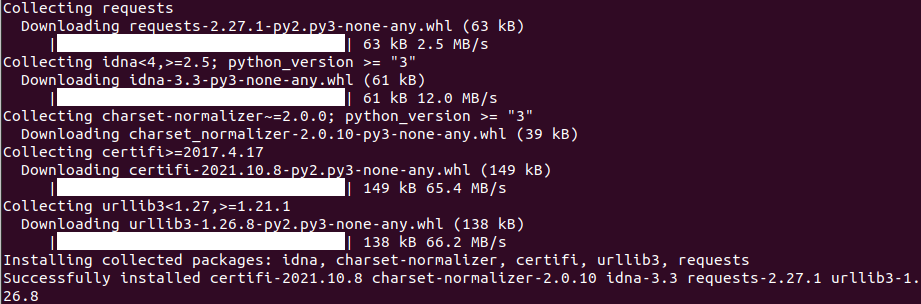
Ahora si instalaremos Flask y Gunicorn, todo en **nuestro entorno env**

*pip install gunicorn flask*

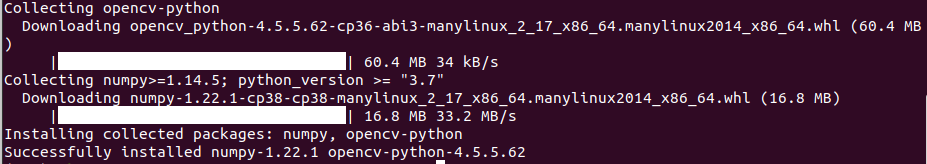


1. Si se desea probar un ejemplo rápido de Flask y Gunicorn revisar el paso 3 de [aquí](https://www.digitalocean.com/community/tutorials/como-preparar-aplicaciones-de-flask-con-gunicorn-y-nginx-en-ubuntu-18-04-es)
2. Instalamos las librerías ***requeridas***

*pip install requests*

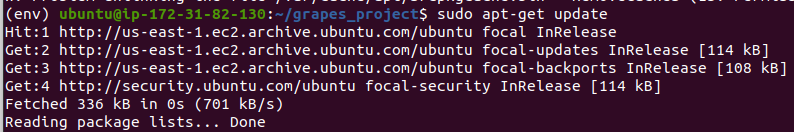


*pip install opencv-python*

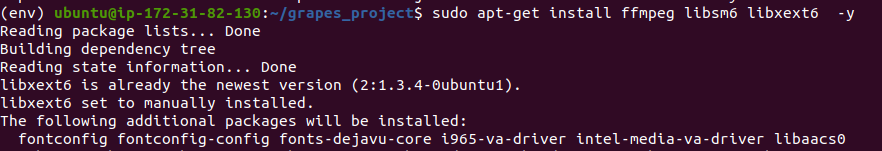


**IMPORTANTE:** Seguramente habrán problemas con la librería cv2 (dependencias no instaladas) por tanto se recomienda ejecutar los siguientes comandos desde sudo

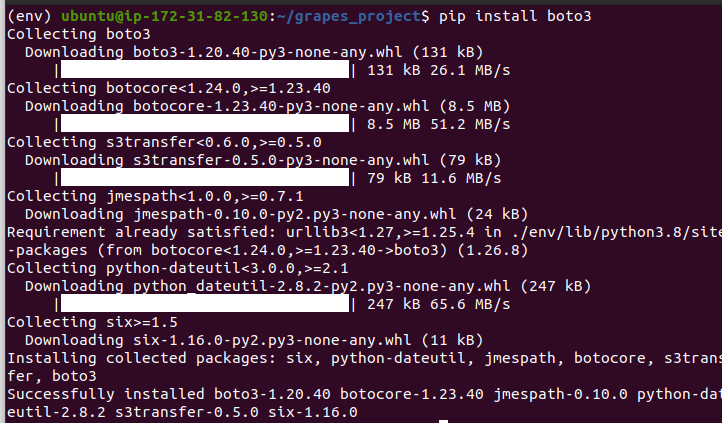
sudo *apt-get update*

**

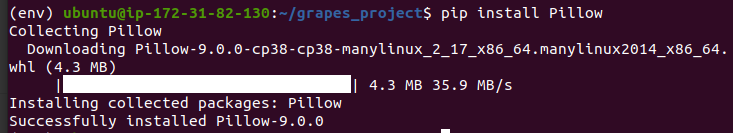
*sudo apt-get install ffmpeg libsm6 libxext6 -y*

**

Seguir instalando más librerías necesarias en nuestro entorno env

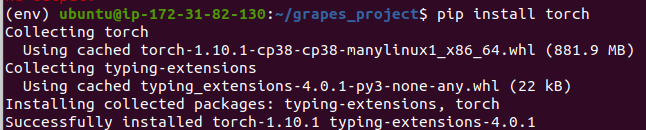
*pip install boto3* 

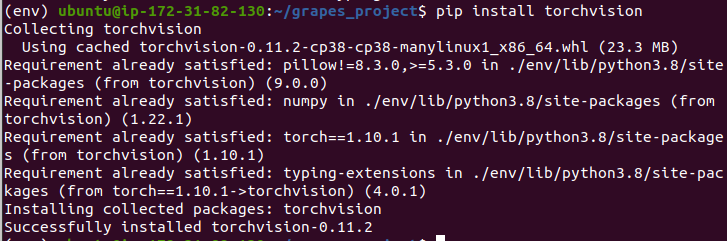
*pip install Pillow*



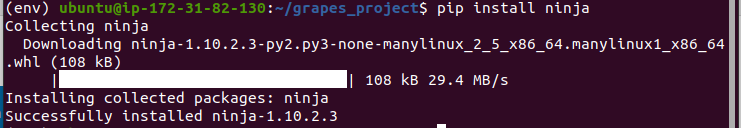
**NOTA:** Ahora unos requerimientos para **Detectron2**

*pip install torch*



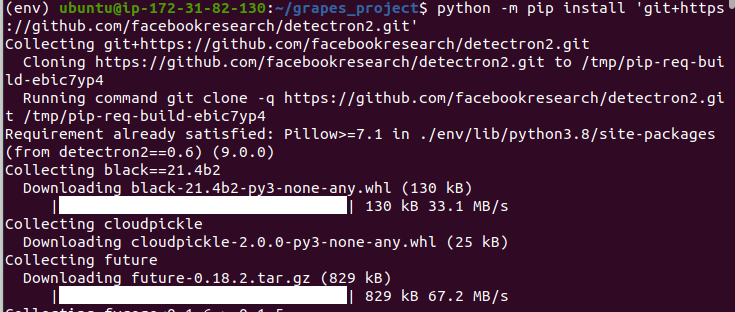
*pip install torchvision* 

También gcc & g++ ≥ 5.4 pero ubuntu ya los tiene, agregar el paquete ***ninja*** para mejorar la velocidad

*pip install ninja* 

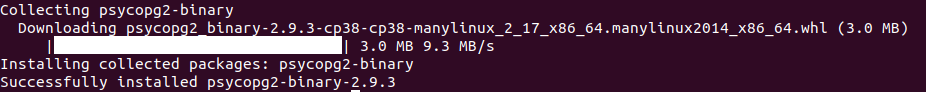
Ahora ya podemos instalar **Detectron2**

*python -m pip install 'git+https://github.com/facebookresearch/detectron2.git'*



1. Con el entorno activado instalar **psycopg2**, para conectar python con postgres

*pip install psycopg2-binary*



1. Ahora desactivamos nuestro entorno ya que tenemos todo lo necesario instalado

*deactivate env*

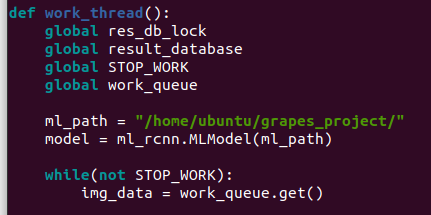
1. Copiamos todos los files (4 archivos en total) a nuestro directorio de trabajo *grapes\_project*

grape.py ml.rcnn.py model\_final.pth wsgi.py



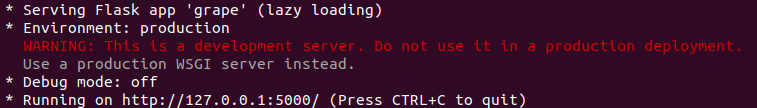
1. En este caso cambiamos el nombre de ubicación a nuestro directorio actual en grape.py:

*/home/ubuntu/grapes\_project/*



1. Ejecutamos wsgi.py

*python3 wsgi.py*



y salimos con Ctrl + C

1. Agregar las keys necesarias (**en caso de usar AWS**)

Copiar la carpeta oculta .aws (con las keys necesarias dentro) al directorio **home** de la máquina aws.

Directorio home:



Donde solamente tenemos nuestro directorio de trabajo



Y pegamos la carpeta .aws y verificamos que se encuentre (carpeta oculta)

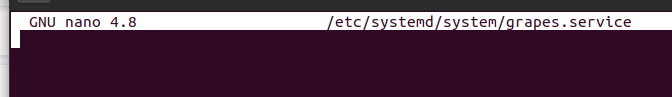


**SOLO CUANDO SE USA AWS**

1. Configurar Gunicorn

A continuación, crearemos el archivo de unidad de servicio systemd. Crear un archivo de unidad systemd permitirá que el sistema init de Ubuntu inicie automáticamente Gunicorn y haga funcionar la aplicación de Flask cuando el servidor se cargue.

*sudo nano /etc/systemd/system/grapes.service*

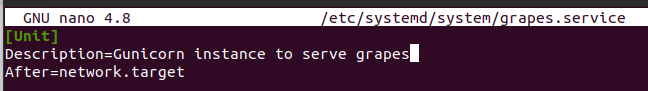
**

En su interior, empezaremos con la sección [Unit] que se usa para especificar metadatos y dependencias. Aquí agregaremos una descripción de nuestro servicio e indicaremos al sistema init que lo inicie solo tras haber alcanzado el objetivo de red:

*[Unit]*

*Description=Gunicorn instance to serve grapes*

*After=network.target*



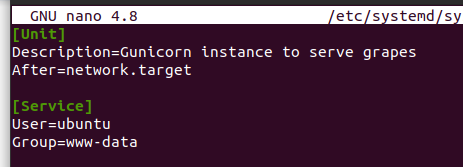
A continuación, abriremos la sección [Service]. Esto especificará el usuario y el grupo con los cuales deseamos que se ejecute el proceso. Otorgamos la propiedad del proceso a nuestra cuenta de usuario normal, ya que tiene la propiedad de todos los archivos pertinentes. También otorgamos la propiedad del grupo al grupo www-data para que Nginx pueda comunicarse fácilmente con los procesos de Gunicorn.

**NOTA:** Nuestro usuario es **ubuntu**

*[Service]*

*User=ubuntu*

*Group=www-data*

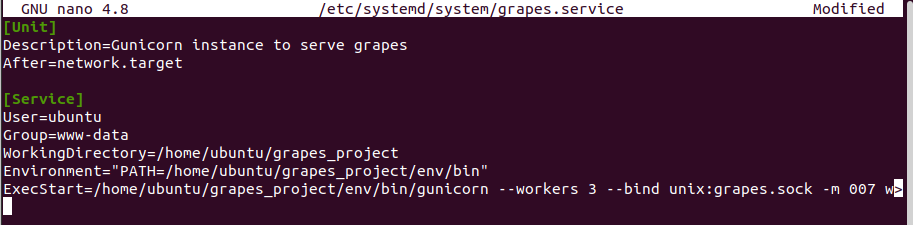


Agregamos directorios

*WorkingDirectory=/home/ubuntu/grapes\_project*

*Environment="PATH=/home/ubuntu/grapes\_project/env/bin"*

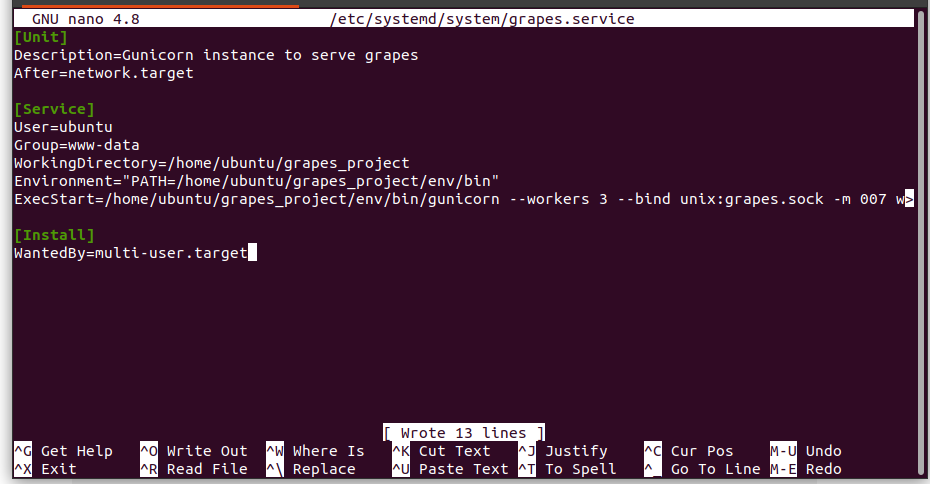
*ExecStart=/home/ubuntu/grapes\_project/env/bin/gunicorn --workers 3 --bind unix:grapes.sock -m 007 wsgi:app*



Añadiremos una sección [Install]

*[Install]*

*WantedBy=multi-user.target*



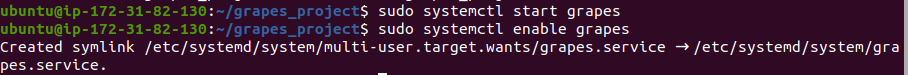
Guardamos Ctrl + S y cerramos Ctrl + X

*pip install azure-storage-blob*

1. Iniciamos el servicio Gunicorn que se acaba de crear y activarlo para que se cargue en el inicio

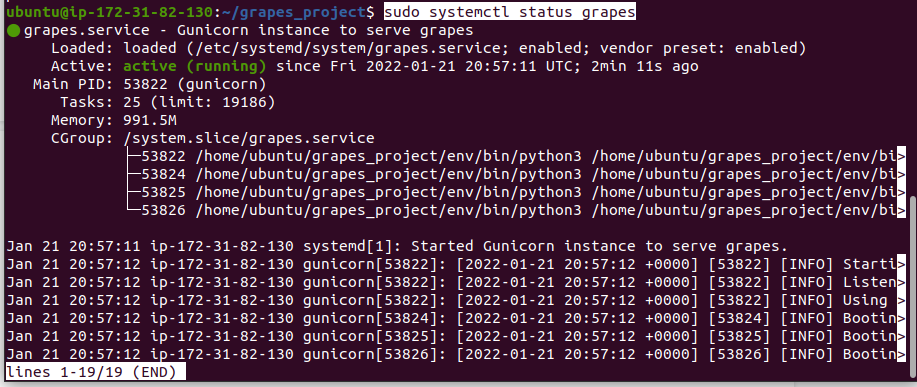
*sudo systemctl start grapes*

*sudo systemctl enable grapes*



1. Comprobamos los servicios

*sudo systemctl status grapes*

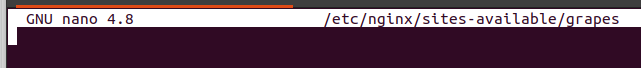


Ahora tendremos un **.sock** en nuestro directorio



1. Configurar Nginx para solicitudes de proxy

Comenzar creando un nuevo archivo de configuración de bloque de servidor en el directorio sites-available de Nginx. Lo llamaremos **grapes**.

*sudo nano /etc/nginx/sites-available/grapes*

1. Agregaremos un bloque de ubicación que coincida con cada solicitud

*server {*

*listen 80;*

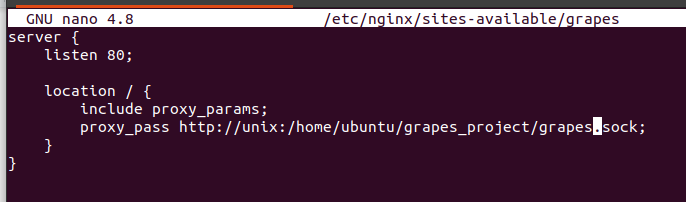
*location / {*

*include proxy\_params;*

*proxy\_pass http://unix:/home/ubuntu/grapes\_project/grapes.sock;*

*}*

*}*



Guardamos Ctrl + S y cerramos Ctrl + X

1. Para habilitar la configuración del bloque de servidor de Nginx que acaba de crear, vincule el archivo al directorio sites-enabled​​​:

*sudo ln -s /etc/nginx/sites-available/grapes etc/nginx/sites-enabled*



Con el archivo en ese directorio, puede realizar una verificación en busca de errores de sintaxis

*sudo nginx -t*

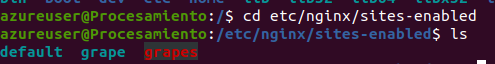


Si no se indican problemas, reinicie el proceso de Nginx para que lea la nueva configuración

*sudo systemctl restart nginx*

1. En caso de algún fallo en los servicios, usar comandos disable y luego restart en dichos procesos y reiniciarlos

O también removerlos manualmente si no permite realizar ninguna acción, borrar en este caso grapes y default (de nginx)



1. Instalar [PostGreSQL](https://www.digitalocean.com/community/tutorials/how-to-install-and-use-postgresql-on-ubuntu-20-04-es), crear y ejecutar la base de datos donde se almacenarán los resultados. Instalar a nivel global.

*sudo apt update*

*sudo apt install postgresql postgresql-contrib*



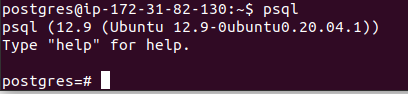
1. Cambiar a la cuenta de postgres

*sudo -i -u postgres*



1. Acceder a una línea de comandos de PostgresSQL. Ejecutar el comando psql

*psql*



1. Crear la base de datos (BD) donde se almacenarán los resultados. Creamos la base de datos y nos conectamos a la misma

*create database grape;*

*\connect grape;*



Ahora nos encontramos en la BD grape



Ejecutamos el siguiente comando para crear nuestra tabla llamada ***detections***

*CREATE TABLE IF NOT EXISTS detections (*

*img\_id INT PRIMARY KEY,*

*img\_url VARCHAR(256) UNIQUE NOT NULL,*

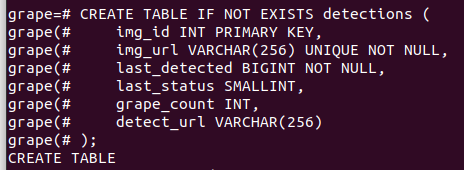
*last\_detected BIGINT NOT NULL,*

*last\_status SMALLINT,*

*grape\_count INT,*

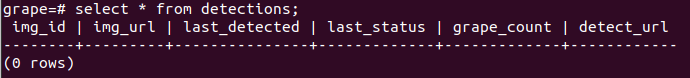
*detect\_url VARCHAR(256)*

*);*



Seleccionamos la tabla ***detections***

*select \* from detections;*



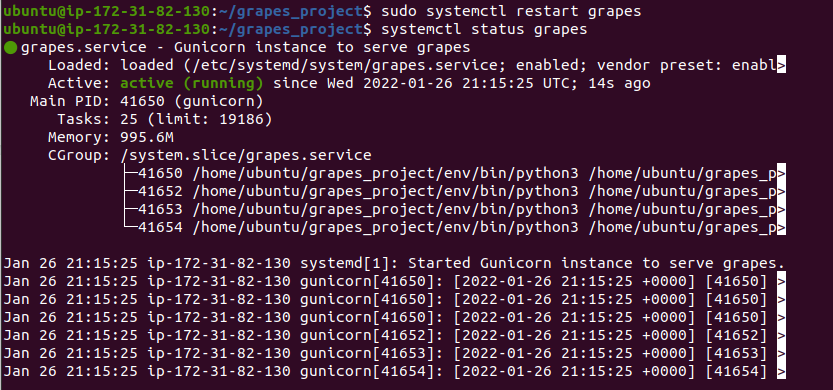
Salir de la BD y de postgres presionando Ctrl + D, y otro Ctrl + D



1. Salir del entorno y regresar al directorio home. restaurar el servicio de **grapes**, y verificar si todo sigue funcionando con normalidad

*sudo systemctl restart grapes*

*systemctl status grapes*



Nuestro servicio **grapes** ahora se encuentra conectado con la BD **detections**

1. Crear nuestro usuario

*sudo -u postgres createuser --interactive*



En este caso: ubuntu



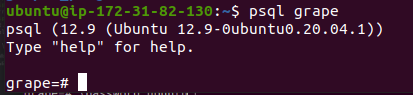
Dar rol de superuser: y



1. Colocar la clave de acceso

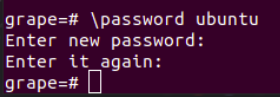
Salir al directorio home y ejecutar psql grape

*psql grape*



Colocar password para ubuntu

*\password ubuntu*

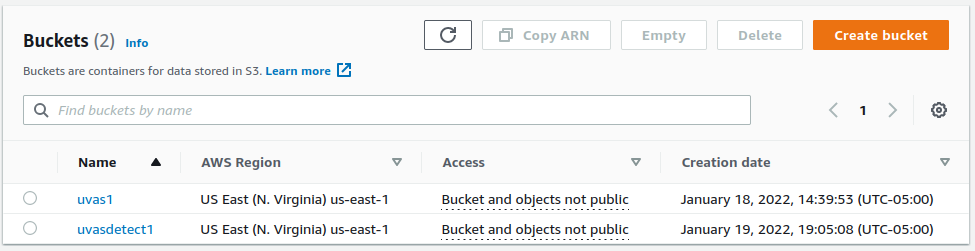


Password: uvas123

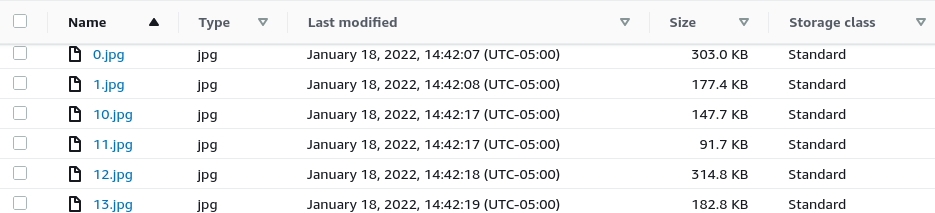
Salir y volver al directorio home

1. Probar la app “Grape” funcionando, contar el número de uvas y verificar resultados en la BD

**IMPORTANTE:** Existen 2 buckets en Amazon S3, uno desde donde se enviarán las imágenes (uvas1) y otro donde se almacenarán las imágenes ya procesadas (uvasdetect1).



Subir las imágenes que se quieren procesar en el bucket *uvas1:*



Ejecutar el siguiente curl:

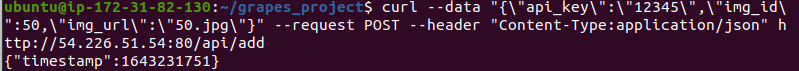
*curl --data "{\"api\_key\":\"12345\",\"img\_id\":****XX****,\"img\_url\":\"****XX****.jpg\"}" --request POST --header "Content-Type:application/json" http://54.226.51.54:80/api/add*

Reemplazar XX con el número de imagen. Para este caso usaremos la imagen del

número 50 del Bucket uvas1



Ejecutamos:



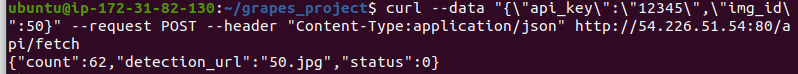
Ahora solicitamos la imagen procesada usando:

*curl --data "{\"api\_key\":\"12345\",\"img\_id\":****XX****}" --request POST --header*

*"Content-Type:application/json" http://54.226.51.54:80/api/fetch*

Reemplazar XX con el ID de imagen procesada. Continuando con la imagen 50 de

ejemplo

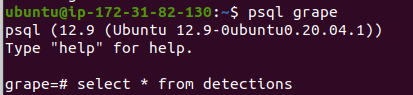


Ahora tenemos el conteo de uvas de la imagen (el resultado), url de la imagen procesada, y su status

1. Verificar almacenamiento en la BD

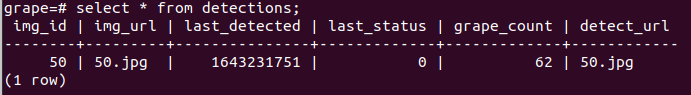
Ingresar a nuestra BD grape y seleccionar toda la tabla

*select \* from detections*



Verificar que la imagen procesada (en este caso 50.jpg) está almacenada en la tabla

detections

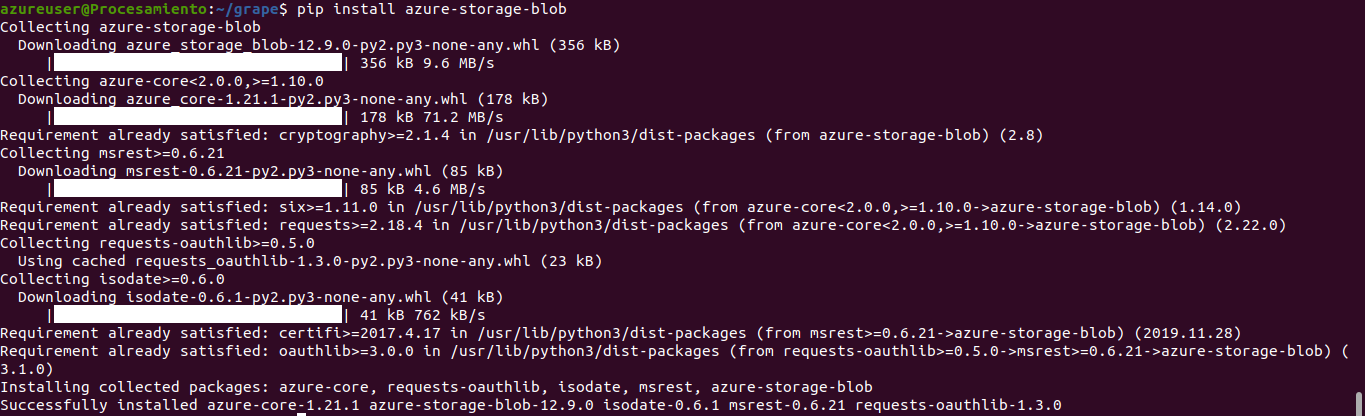


1. Si se usa otro server como azure, seguir la guía de instalación de blob (abajo)

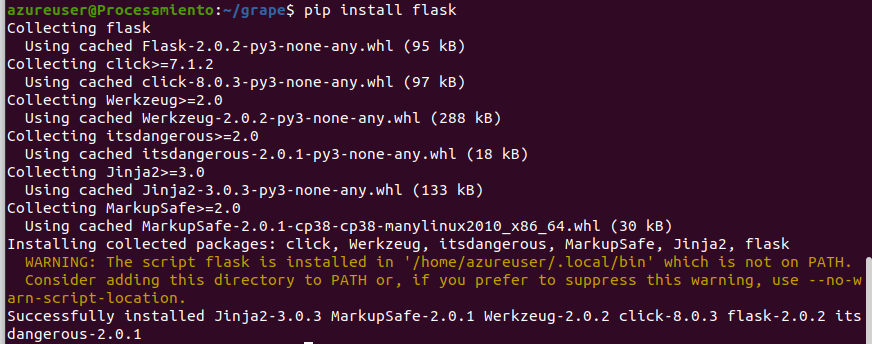
**Configurar blobs con python en Azure**

1. Cambiar a nuestro directorio de trabajo (***grape*** en este caso)
2. Instalar blob

*pip install azure-storage-blob*



*pip install flask*

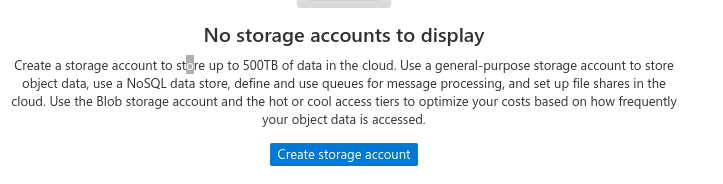


1. Ingresar al portal de azure

<https://portal.azure.com/#home>

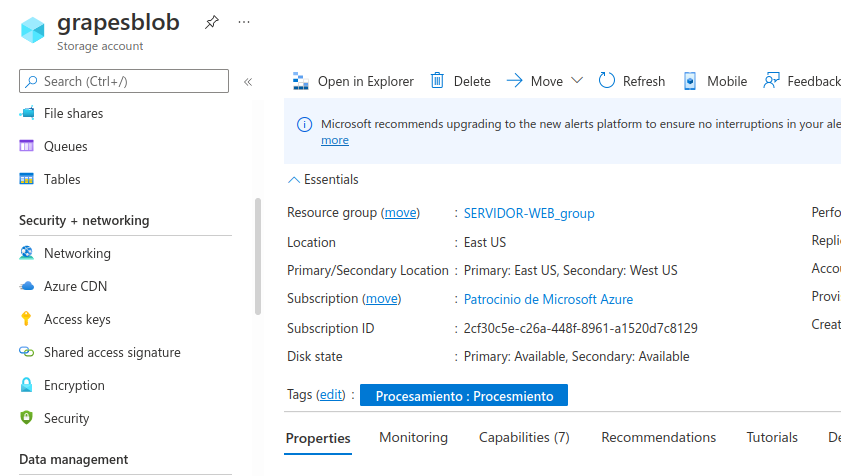
y seleccionar storage

1. Crear una cuenta de storage

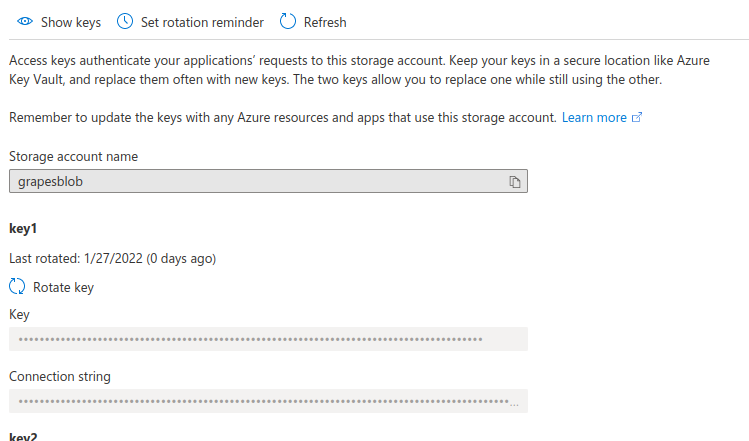


Seguir los pasos, en este caso la cuenta se llama **grapesblob**

1. Ingresar a **Access keys**, debajo de Security + Networking

****

1. Copiar nuestra key string



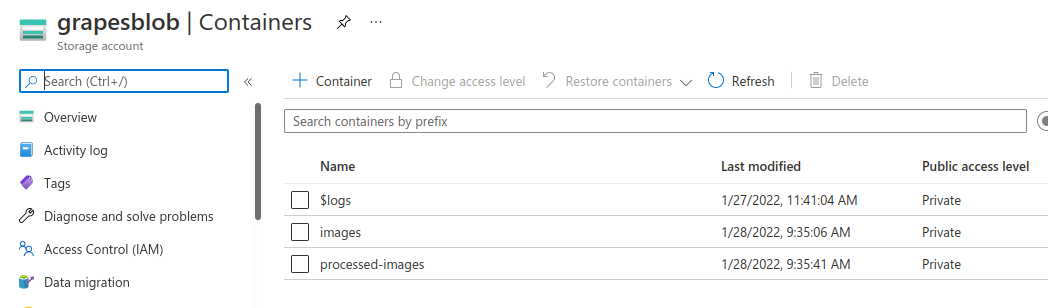
Clic en show keys y copiar connection string de key1

1. Configurar string de conexión de almacenamiento

En este caso es:

*"DefaultEndpointsProtocol=https;AccountName=grapesblob;AccountKey=AuPGit9tvo/+qyK2J7cUgYYYTRgi/hlH6jKfxyJ7LfM/qdeC39D8lG6DzxY2vcVBTa6SLpKBUVcckfWrsmL9Ow==;EndpointSuffix=core.windows.net"*

1. Restaurar servicios activos
2. Crear 2 containers uno para las imágenes de uvas, y otro para las imágenes de uvas procesadas procesadas.

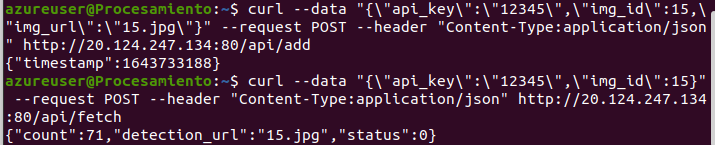


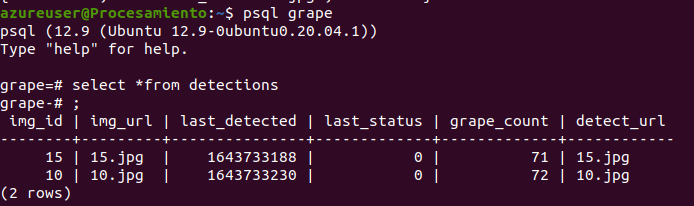
IMPORTANTE: Marcar la casilla “Overwrite if files already exist” en Upload

1. Ejecutar curls y probar

*curl --data "{\"api\_key\":\"12345\",\"img\_id\":9,\"img\_url\":\"9.jpg\"}" --request POST --header "Content-Type:application/json" http://20.124.247.134:80/api/add*

*curl --data "{\"api\_key\":\"12345\",\"img\_id\":9}" --request POST --header "Content-Type:application/json" http://20.124.247.134:80/api/fetch*



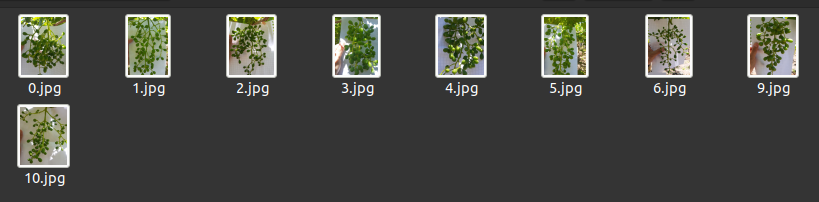


**Probar Inferencias del modelo ejecutando ml\_rcnn.py**

(machine learning recurrent neural network)

1. Subir imágenes que vamos a inferir con nuestro modelo (*model\_final.pth*) a nuestro directorio de trabajo en este caso **grapes\_project**, en una carpeta llamada “dataset”

Las imágenes:



Ahora en nuestro directorio de trabajo tenemos una carpeta de nombre “dataset”

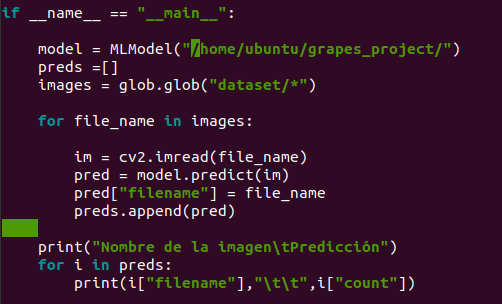


Verificamos que se encuentren todas las imágenes dentro

1. Editamos nuestro script **ml\_rcnn.py**



Nos dirigimos hacia la parte final y editamos la dirección y colocamos nuestro directorio local de trabajo. En este caso:

/home/ubuntu/grapes project/

**Notar:**  Que en la dirección de “images” abajo de la dirección del modelo es de nuestra carpeta “dataset” y todas las imágenes \*. Se puede modificar y colocar otro nombre según sea el caso.

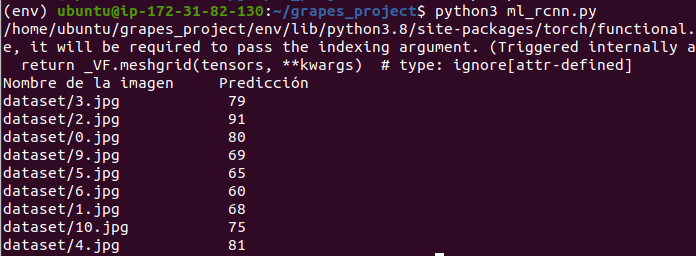
Guardamos Ctrl + S y cerramos Ctrl + X

1. Activamos nuestro entorno donde tenemos todas las librerías necesarias

*source env/bin/activate* 

1. Ejecutamos el script **ml\_rcnn.py** (Puede tomar varios segundos, tomar en cuenta que se puede perder la conección)

*python3 ml\_rcnn.py*

****

Se muestra en pantalla el nombre de la imagen y la predicción o cantidad de uvas que predice el modelo en cada una.