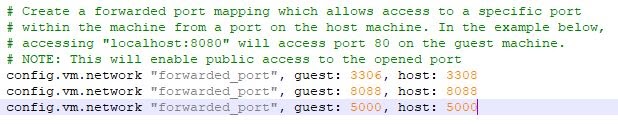
**Configuración de la máquina virtual (vagrant)**

Para que el servicio configurado pueda correr correctamente, se requiere realizar algunas configuraciones a la máquina virtual.

Lo primero será configurar el archivo Vagrantfile y el wsgi(Web Service Gateway Interface)

para que la aplicación pueda recibir simultáneamente varias solicitudes.

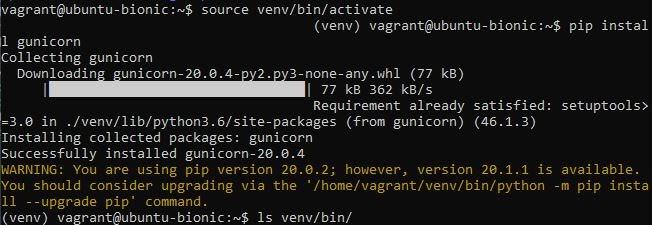
Lo primero será modificar el archivo Vagrantfile de la siguiente manera ya que más adelante en la configuración wsgi se configurará el puerto 8088 donde se aloja la aplicación



**Figura X1.** Configuración puerto 8088.

Una vez realizado este paso, se crea un ambiente virtual de python 3 donde se instalan todas las librerías y requerimientos que se necesitan para correr la máquina virtual.

Posteriormente se instala el paquete gunicorn el cual funciona como un servidor WSGI para python



**Figura X1.** Activación de entorno virtual e instalación de gunicorn.

Ahora se realiza la instalación de los paquetes en el archivo requirements.txt mediante el siguiente comando

sudo pip3 install -r /home/vagrant/requirements.txt

El siguiente paso es crear el archivo wsgi,py dentro de la carpeta de nuestra aplicación que para este caso se llama “ejemplos”.



**Figura X1.** Creación archivo wsgi,py.

Dentro del archivo “wgsi.py” se importa el archivo donde se encuentran definidas las funciones que van a recibir las peticiones, es decir el archivo que contiene las rutas y los métodos GET,POST,PUT y DELETE de la aplicación. Para este caso este archivo ha sido llamado “hola.py” y dentro de ese archivo se configura también el nombre de la aplicación flask, el cual se llamó app.

Dentro del archivo wgsr.py se le entrega el nombre de la aplicación flask como parámetro, lo cual se evidencia en las líneas de código 3 y 4 de la siguiente imagen:



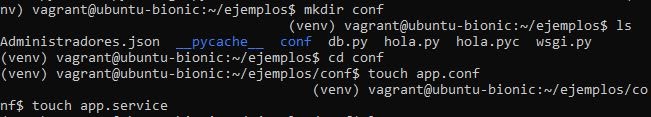
**Figura X1.** Configuración del archivo wsgi.py.

Ahora el siguiente paso será configurar el proxy de la aplicación para que reciba las peticiones mediante un archivo “socked”, para esto es necesario instalar nginx.



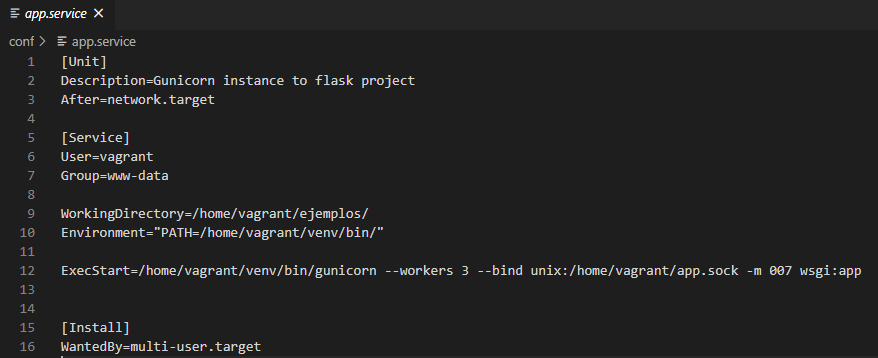
**Figura X1.** Instalación nginx.

Posteriormente se crea una carpeta en el mismo directorio donde se encuentra el archivo “hola.py”, esta carpeta se llama “conf” y dentro de esta, se crean dos archivos; uno llamado app.conf y otro llamado app.service.



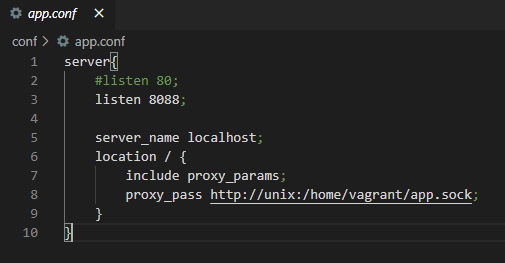
**Figura X1.** Creación de carpeta “conf” y archivos app.conf, app.service.

Dentro del archivo app.service se configuran los servicios para que se ejecute correctamente la app, por esto, como se ve en la figura siguiente, en la línea 9 se debe entregar la ruta del directorio que contiene la app, en la 10 se le pasa la ruta del ambiente donde se encuentra instalado “gunicorn” y en la línea 12 se le indica que empiece a ejecutar la aplicación desde donde se encuentra el archivo gunicorn y lo pase a un archivo .sock que se encarga de ejecutar la aplicación mediante el WSGI



**Figura X1.** Configuración del archivo app.service.

En el archivo app.conf que se observa en la siguiente figura, se configuran el puerto y el dominio desde donde se va a ejecutar la aplicación, esto se puede ver en las líneas 3 y 5 respectivamente, adicional en la línea 7 se le entregan los parámetros para que ejecute proxy configurado en el app.service.



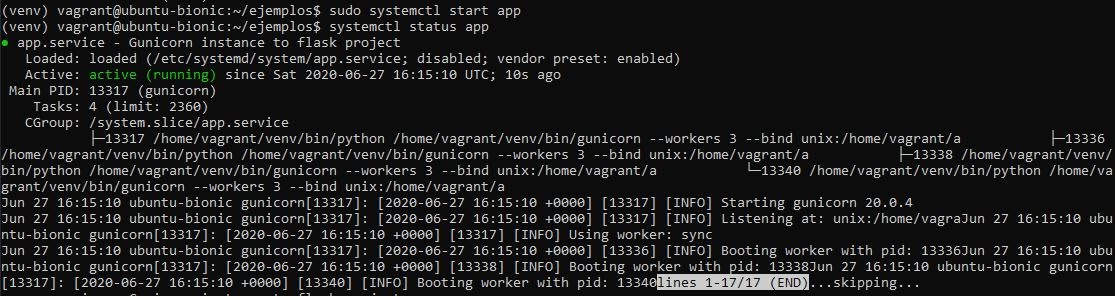
**Figura X1.** Configuración del archivo app.conf.

Como último paso se debe copiar el archivo app.service dentro de la carpeta “system” de la maquina virtual donde estamos trabajando.

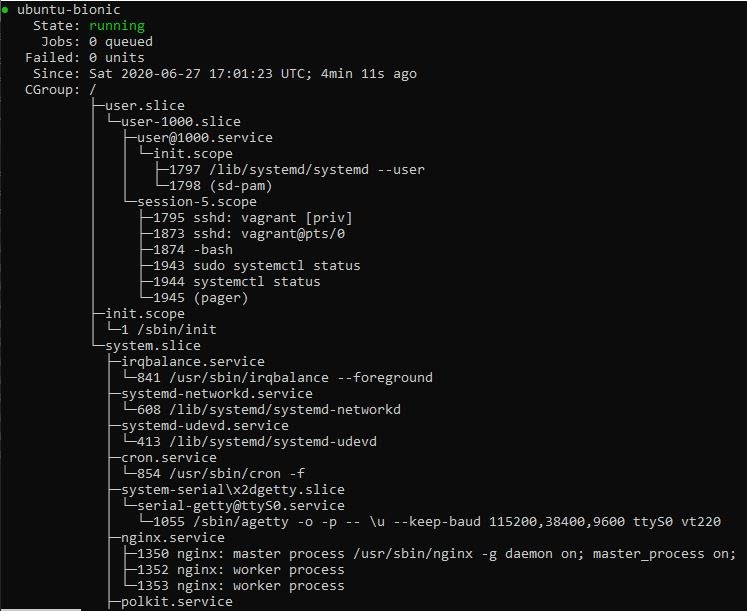


**Figura X1.** Copiar app.service en carpeta de sistema.

Con estos pasos ya está configurada la app para que funcione mediante el servidor gunicorn y el proxy nginx, ahora solo se debe iniciar la app mediante el comando “systemctl start app” y corroborar su estado,



**Figura X1.** Encender el servicio de app y comprobar el estado.



**Figura X1.** Servicio ejecutándose correctamente.

Para comprobar que el servicio esté activo, se abre una pestaña en google chrome y ponemos el dominio junto con el puerto 8088 y la ruta de una de las funciones que usa el método “GET”, esta función se explicará en la siguiente sección.



**Figura X1.** Servicio ejecutándose correctamente

Como se aprecia en la figura anterior, al entregar la ruta y id correctos en el espacio de la url, se observa que el servidor devuelve un diccionario con los datos correspondientes al registro de id 2 guardados en el archivo JSON.