Laboratorio N° 4.2

“Configuración Nginx con Python”

Calle Emilio Fernández N° 296 Of. 601 Urb. Santa Beatriz Lima
Teléfono: (+511)555-0008 (51)982735527

LABORATORIO 4.2

Objetivos:

- ✓ Habilitar una aplicación con Python y crearla como servicio.

PROCEDIMIENTO

Centos7 se entrega con Python 2.7.5, que es una parte critica del sistema base Centos. SCL permitirá instalar versiones más recientes de Python 3.x junto con la versión predeterminada de Python 2.7.5 para que las herramientas del sistema como yum continúen funcionando correctamente

Pasos previos.

1. **Descargamos el repositorio epel para librerías externas necesarias.**

```
# yum install epel-release -y
```

2. **Habilite SCL instalando el archivo de lanzamiento Centos SCL que se incluye en el repositorio adicional de centos**

```
# yum install centos-release-scl -y
```

3. **Una vez que el repositorio este habilitado, instale Python 3.6 con el siguiente comando.**

```
# yum install rh-python36 -y
```

4. **También habilitaremos dependencias de C para hacer los test.**

```
# yum install gcc -y
```

5. **Instalamos virtualenv con usando pip.**

```
# pip install virtualenv
# python --version
Python 2.7.5
```

6. **Para acceder a python3.6, debe iniciar una nueva instancia de Shell con la herramienta SCL y verificamos.**

```
# scl enable rh-python36 bash
# python --version
Python 3.6.3
```

7. **Creamos nuestro workspace, generamos usuarios, le brindamos permisos y nos situamos en ella.**

```
# useradd jorge -d /home/jorge
# mkdir /home/jorge/miproyecto
# cd /home/jorge/miproyecto/
# chown jorge.jorge /home/jorge/ -R
```

8. **Crearemos y activaremos el virtualenv en nuestra carpeta.**

```
# virtualenv .venv
# source /home/jorge/miproyecto/.venv/bin/activate
```

9. **Instalamos flask y uwsgi.**

```
# pip install uwsgi flask
# pip install --upgrade pip
```

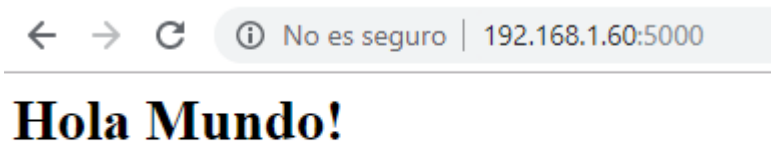
10. **Creamos el archivo flask.**

```
# vim /home/jorge/miproyecto/app.py
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "<h1>Hola Mundo!</h1>"

if __name__ == "__main__":
    app.run(host='0.0.0.0')
```

11. **Testeamos el app.py.**

```
# python app.py
```

```
(.venv) [root@srvweb miproyecto]# python app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```



Hola Mundo!

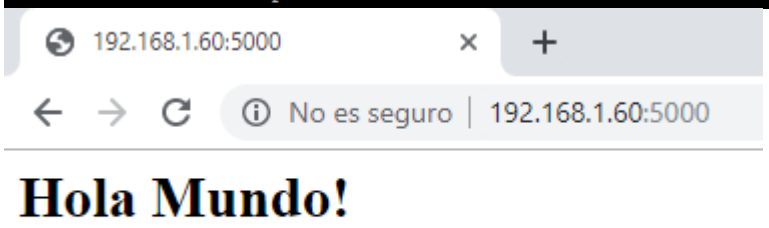
12. **Creamos el punto de entrada para WGSI.**

```
# vim /home/jorge/miproyecto/wsgi.py
from app import app

if __name__ == "__main__":
    app.run()
```

13. **Testeamos el UWSGI.**

```
# uwsgi --socket 0.0.0.0:5000 --protocol=http -w wsgi:app
```

```
(.venv) [root@srvweb miproyecto]# uwsgi --socket 0.0.0.0:5000 --protocol=http -w wsgi:app
*** Starting uWSGI 2.0.18 (64bit) on [Thu Sep 12 16:26:09 2019] ***
compiled with version: 4.8.5 20150623 (Red Hat 4.8.5-36) on 12 September 2019 21:19:44
os: Linux-3.10.0-862.11.6.el7.x86_64 #1 SMP Tue Aug 14 21:49:04 UTC 2018
nodename: srvweb.ose.pe
```



Hola Mundo!

14. **Desactivamos el VirtualEnv.**

```
# deactivate
```

15. **Creamos el archivo de configuración para uWSGI.**

```
# vim /home/jorge/miproyecto/miproyecto.ini
[uwsgi]
module = wsgi:app

master = true
processes = 5

socket = miproyecto.sock
chmod-socket = 660
vacuum = true

die-on-term = true
```

16. **Creamos el archivo de servicio systemd.**

```
# vim /etc/systemd/system/miproyecto.service
[Unit]
Description=uWSGI instance to serve miproyecto
After=network.target

[Service]
User=jorge
Group=jorge
WorkingDirectory=/home/jorge/miproyecto
Environment="PATH=/home/jorge/miproyecto/.venv/bin"
ExecStart=/home/jorge/miproyecto/.venv/bin/uwsgi --ini miproyecto.ini

[Install]
WantedBy=multi-user.target
```

17. **Damos permisos adecuado.**

```
# usermod -aG jorge jorge
# chown -R jorge:jorge /home/jorge/*
```

18. **Iniciamos y habilitamos el servicio.**

```
# systemctl start miproyecto.service
# systemctl enable miproyecto.service
# systemctl status miproyecto.service
```

```
[root@srvweb miproyecto]# systemctl status miproyecto.service
● miproyecto.service - uWSGI instance to serve miproyecto
   Loaded: loaded (/etc/systemd/system/miproyecto.service; enabled; vendor preset: disabled)
   Active: active (running) since Thu 2019-09-12 16:28:09 -05; 8s ago
 Main PID: 2335 (uwsgi)
   CGroup: /system.slice/miproyecto.service
           ├─2335 /home/josua/miproyecto/.venv/bin/uwsgi --ini miproyecto.ini
           ├─2337 /home/josua/miproyecto/.venv/bin/uwsgi --ini miproyecto.ini
           ├─2338 /home/josua/miproyecto/.venv/bin/uwsgi --ini miproyecto.ini
           ├─2339 /home/josua/miproyecto/.venv/bin/uwsgi --ini miproyecto.ini
           ├─2340 /home/josua/miproyecto/.venv/bin/uwsgi --ini miproyecto.ini
           └─2341 /home/josua/miproyecto/.venv/bin/uwsgi --ini miproyecto.ini

Sep 12 16:28:09 srvweb.ose.pe uwsgi[2335]: mapped 437424 bytes (427 KB) for 5 cores
Sep 12 16:28:09 srvweb.ose.pe uwsgi[2335]: *** Operational MODE: preforking ***
Sep 12 16:28:09 srvweb.ose.pe uwsgi[2335]: WSGI app 0 (mountpoint='') ready in 0 seconds on interpreter 0x231e6d0 pid: 2335 (default app)
Sep 12 16:28:09 srvweb.ose.pe uwsgi[2335]: *** uWSGI is running in multiple interpreter mode ***
Sep 12 16:28:09 srvweb.ose.pe uwsgi[2335]: spawned uWSGI master process (pid: 2335)
Sep 12 16:28:09 srvweb.ose.pe uwsgi[2335]: spawned uWSGI worker 1 (pid: 2337, cores: 1)
Sep 12 16:28:09 srvweb.ose.pe uwsgi[2335]: spawned uWSGI worker 2 (pid: 2338, cores: 1)
Sep 12 16:28:09 srvweb.ose.pe uwsgi[2335]: spawned uWSGI worker 3 (pid: 2339, cores: 1)
Sep 12 16:28:09 srvweb.ose.pe uwsgi[2335]: spawned uWSGI worker 4 (pid: 2340, cores: 1)
Sep 12 16:28:09 srvweb.ose.pe uwsgi[2335]: spawned uWSGI worker 5 (pid: 2341, cores: 1)
```

19. **Testeamos el UWSGI.**

```
# cp -r /etc/nginx/conf.d/default.conf
/etc/nginx/conf.d/default.conf.backup
```

20. **Testeamos el UWSGI.**

```
# rm -rf /etc/nginx/conf.d/default.conf
```

21. **Testeamos el UWSGI.**

```
# vim /etc/nginx/conf.d/test.conf
server {
    listen 80;
    server_name localhost;

    location / {
        include uwsgi_params;
        uwsgi_pass unix:///home/jorge/miproyecto/miproyecto.sock;
    }
}
```

22. **Testeamos el UWSGI.**

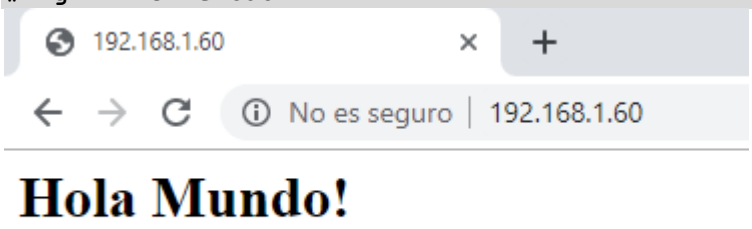
```
# setfacl -Rm u:nginx:rwx /home/jorge/
```

23. **Testeamos el UWSGI.**

```
# nginx -t
[root@srvweb miproyecto]# nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

24. **Testeamos el UWSGI.**

```
# nginx -s reload
```

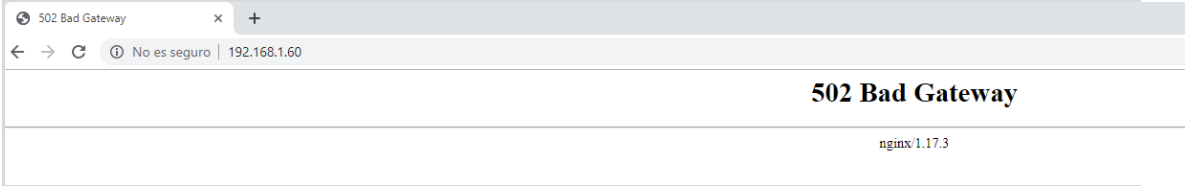


25. **Modificaremos el archivo app.py para ver si realmente está obteniendo los cambios.**

```
# vim app.py
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
        return "<h1>Hola JORGE!!!</h1>"

if __name__ == "__main__":
        app.run(host='0.0.0.0')
```

26. Reiniciamos el servicio.

```
# systemctl restart miproyecto.service
```



27. ¿Por qué sucede esto?, porque todo el directorio se está refrescando así que también hay que darle permisos al directorio de nuestro workspace.

```
# cd ..
```

28. Aplicamos una regla personalizada para ejecutar el directorio.

```
# setfacl -Rm u:nginx:rwX /home/jorge
```

29. Aplicación una regla personalizada para poder ingresar al directorio.

```
# setfacl -Rm d:u:nginx:rw- /home/jorge
```

30. Ingresamos otra vez al proyecto.

```
# cd miproyecto/
```

31. Reiniciamos el servicio.

```
# systemctl restart miproyecto.service
```

32. Perfecto tenemos nuestro nginx con Python configurado.