

Mini Proyecto Sistemas Operacionales

Universidad Icesi

Curso: Sistemas Operativos

Docente: Daniel Barragán C.

Tema: Servicios web

Estudiantes:

José Luis Osorio Quintero 12203012

Juan David Cardona Mena 13103002

Juan Pablo Medina Mora 11112010

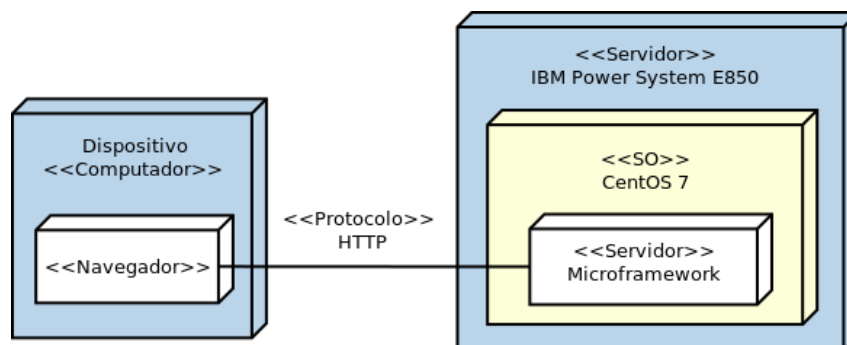
Objetivos

- Desplegar una aplicación en un servidor que ejecuta el sistema operativo Linux
- Realizar los ajustes y depuración necesarios para desplegar una aplicación en Linux
- Realizar aplicaciones para obtener información del sistema operativo

Descripción

Para el despliegue de una aplicación en un servidor se requiere conocer los procedimientos necesarios relacionados con la configuración de las interfaces de red, ajustes de seguridad, instalación de dependencias, usuarios y herramientas de depuración del sistema operativo.

El siguiente proyecto consiste en el despliegue de una aplicación web para obtener información del sistema operativo (Deberá emplear la aplicación desarrollada en el primer parcial). Para este propósito se debe emplear el sistema operativo CentOS7, el microframework flask y ambientes virtuales.

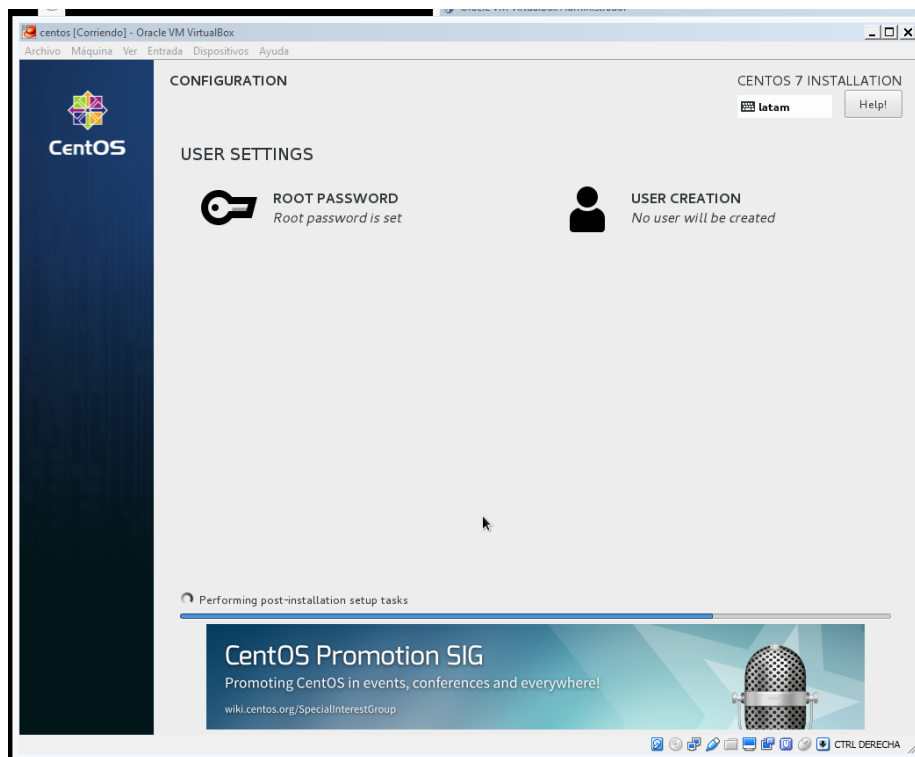


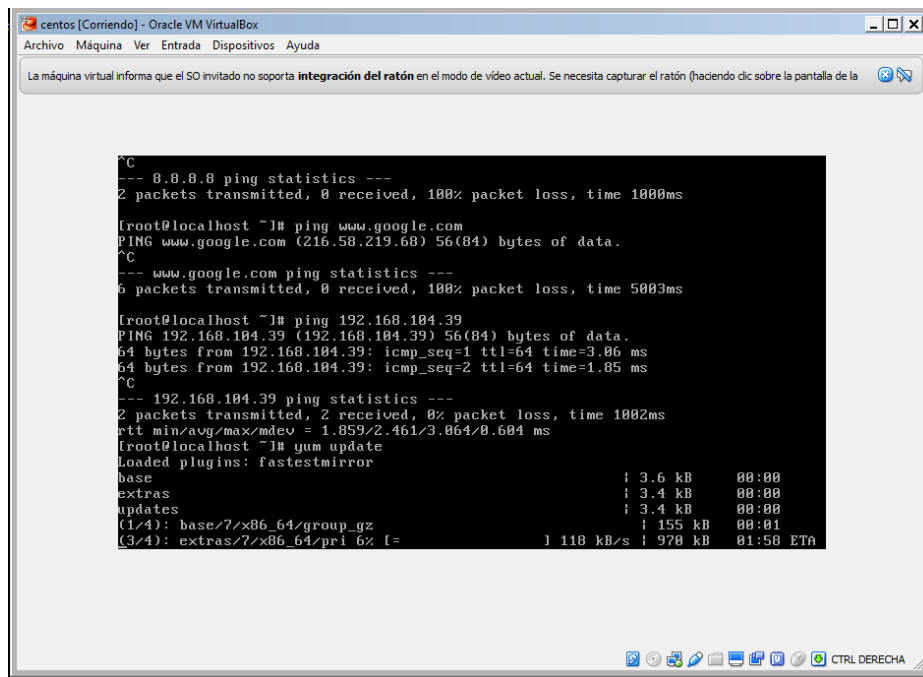
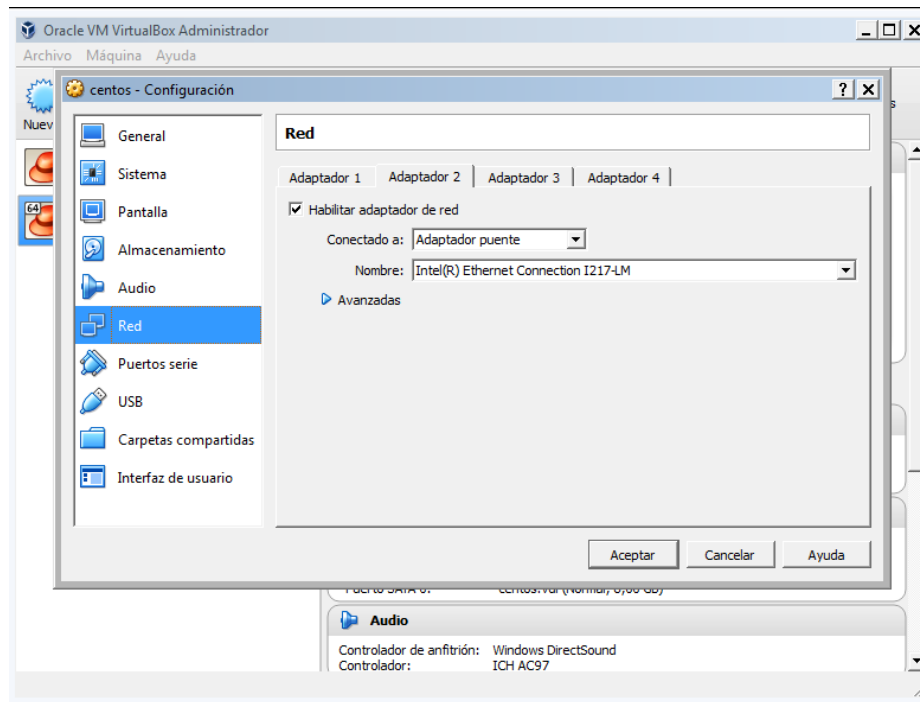
Desarrollo

En el presente apartado se explicará todo el proceso realizado para la resolución del taller, desde la instalación de la máquina virtual hasta la prueba del servicio web usando Paperman y netstat.

Instalación del sistema operativo CentOS 7 y de interfaces de red

Para la instalación del sistema CentOS 7 se procedió a descargar una imagen del sistema operativo, en específico el CentOS 7 x86 64 minimal. Con esta imagen descargada se procedió a realizar la instalación de la máquina virtual en el programa VirtualBox, inicialmente se creó el usuario administrador para después configurar la memoria y demás apartados del sistema incluyendo los adaptadores de red puente y NAT. El proceso de muestra a continuación.





Al iniciar sesión con las credenciales para el usuario raíz creado, se procedió a probar que la configuración de los adaptadores de red hubiese sido exitosa, esto por medio de la realización de pings y posteriormente por medio de la conexión al programa putty.

```
# ping 8.8.8.8
```

Con esto se finalizó la instalación y configuración básica del sistema operativo.

Configuración de puertos

En la actualización a CentOS 7 uno de los cambios principales al sistema fue el reemplazo del archivo de configuración iptables, en esta versión para realizar la configuración de puertos se usa el componente FirewallD. Para realizar esta configuración se optó por desactivar el componente FirewallD, e instalar posteriormente el componente iptables, en el cual se procedió a realizar la configuración, en específico se habilito el puerto 8088.

```
# cat /etc/sysconfig/iptables
# service iptables restart
```

```
[root@localhost ~]# yum install iptables-services
Loaded plugins: fastestmirror
base                                     | 3.6 kB  00:00:00
extras                                 | 3.4 kB  00:00:00
updates                               | 3.4 kB  00:00:00
(1/4): base/7/x86_64/group_gz         | 155 kB  00:00:01
(2/4): extras/7/x86_64/primary_db     | 166 kB  00:00:06
(3/4): base/7/x86_64/primary_db      | 5.3 MB  00:01:41
(4/4): updates/7/x86_64/primary_db   | 9.1 MB  00:02:20
Determining fastest mirrors
 * base: mirror.esPOCH.edu.ec
 * extras: mirror.esPOCH.edu.ec
 * updates: mirror.esPOCH.edu.ec
Resolving Dependencies
--> Running transaction check
--> Package iptables-services.x86_64 0:1.4.21-16.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package                        Arch      Version           Repository        Size
=====
Installing:
iptables-services             x86_64    1.4.21-16.el7     base              50 k
Transaction Summary
=====
Install 1 Package

Total download size: 50 k
Installed size: 24 k
Is this ok [y/d/N]: y
Downloading packages:
warning: /var/cache/yum/x86_64/7/base/packages/iptables-services-1.4.21-16.el7.x86_64.rpm: Header V3 RSA/SHA256 Sig
nature, key ID f4a80eb5: NOKEY
Public key for iptables-services-1.4.21-16.el7.x86_64.rpm is not installed
iptables-services-1.4.21-16.el7.x86_64.rpm
Retrieving key from file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
Importing GPG Key 0x4a80eb5:
 Userid      : "CentOS-7 Key (CentOS 7 Official Signing Key) <security@centos.org>"
Fingerprint: 6341 ab27 53d7 8a78 a7c2 7bb1 24c6 a8a7 f4a8 0eb5
Package      : centos-release-7-2.1511.el7.centos.2.10.x86_64 (@anaconda)
From         : /etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
Is this ok [y/N]: y
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : iptables-services-1.4.21-16.el7.x86_64                                1/1
  Verifying  : iptables-services-1.4.21-16.el7.x86_64                                1/1

Installed:
iptables-services.x86_64 0:1.4.21-16.el7

Complete!
```

```

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 08:00:27:7d:65:86 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic enp0s3
        valid_lft 85102sec preferred_lft 85102sec
    inet6 fe80::a00:27ff:fe7d:6586/64 scope link
        valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 08:00:27:9f:7f:77 brd ff:ff:ff:ff:ff:ff
    inet 192.168.104.32/26 brd 192.168.104.63 scope global dynamic enp0s8
        valid_lft 5902sec preferred_lft 5902sec
    inet6 fe80::a00:27ff:fe9f:7f77/64 scope link
        valid_lft forever preferred_lft forever

```

```

# sample configuration for iptables service
# you can edit this manually or use system-config-firewall
# please do not ask us to add additional ports/services to this default configuration
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 8080 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 9090 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 9191 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 8088 -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
COMMIT
~

```

Con lo anterior el puerto del que hace uso el servicio queda habilitado.

Creación del usuario `filesystem_user`, creación de ambiente virtual e instalación de flask y sus dependencias

Con las configuraciones iniciales pertinentes terminadas el primer paso para hacer uso de los programas consistió en la creación de un nuevo usuario, para tal fin se siguió el siguiente procedimiento

```

# adduser filesystem_user
# passwd password

```

Con el nuevo usuario creado se procedió a crear el ambiente virtual que el usuario utilizaría

```

# su filesystem_user
$ cd ~/
$ mkdir envs
$ cd envs
$ virtualenv flask_env

```

Se activó el ambiente virtual creado

```
$ cd ~/envs
$ . flask_env/bin/activate
```

Después se procedió a instalar Flask en el nuevo ambiente virtual

```
$ pip install Flask
```

```
Saving to: 'get-pip.py'
100%[=====>] 1,524,722 108KB/s in 14s
2016-11-02 09:33:48 (104 KB/s) - 'get-pip.py' saved [1524722/1524722]

You have mail in /var/spool/mail/root
[root@localhost tmp]# python get-pip.py
Collecting pip
  Downloading pip-8.1.2-py2.py3-none-any.whl (1.2MB)
    100% |#####| 1.2MB 122kB/s
Collecting setuptools
  Downloading setuptools-28.7.1-py2.py3-none-any.whl (472kB)
    100% |#####| 481kB 234kB/s
Collecting wheel
  Downloading wheel-0.29.0-py2.py3-none-any.whl (66kB)
    100% |#####| 71kB 121kB/s
Installing collected packages: pip, setuptools, wheel
Successfully installed pip-8.1.2 setuptools-28.7.1 wheel-0.29.0
[root@localhost tmp]#
[root@localhost tmp]#
[root@localhost tmp]#
[root@localhost tmp]# pip install virtualenv
Collecting virtualenv
  Downloading virtualenv-15.0.3-py2.py3-none-any.whl (3.5MB)
    100% |#####| 3.5MB 95kB/s
Installing collected packages: virtualenv
Successfully installed virtualenv-15.0.3
```

Aplicación en Python

A continuación, se presentan los contratos del servicio desplegado.

Descripción de las URIs

	POST	GET	PUT	DELETE
/files	Crear archivo	Obtener listado de archivos	No aplica	Elimina todos los archivos
/files/recently_created	No aplica	Retorna los archivos que se crearon recientemente	No aplica	No aplica

Descripción de los formatos de envío de las solicitudes

	POST	GET	PUT	DELETE
/files	JSON	No aplica	No aplica	No aplica
/files/recently_created	No aplica	No aplica	No aplica	No aplica

Descripción de los formatos de respuesta de las solicitudes

	POST	GET	PUT	DELETE
/files	HTTP 201 CREATED	JSON	HTTP 404 NOT FOUND	HTTP 200 OK
/files/recently_created	HTTP 404 NOT FOUND	JSON	HTTP 404 NOT FOUND	HTTP 404 NOT FOUND

Descripción del formato de intercambio de datos (JSON)

```
{
  "filename": "carta",
  "content": "this is the file content"
}

{
  "files": [
    "carta",
    "listado",
    "tareas",
    "recordatorio"
  ]
}
```

A continuación, se presentan los archivos que se usaron para correr los servicios.

Archivo functions.py

```
from subprocess import Popen, PIPE

def create_files(filename,content):
    filecreate = Popen(["touch",filename+'.txt'], stdout=PIPE, stderr=PIPE)
    writeFile = Popen(['echo',content,'>>', filename+'.txt'], stdin=filecreate.stdout, stdout=PIPE,
stderr=PIPE).communicate()
    return True

def get_all_files():
```

```

process = Popen(["ls"], stdout=PIPE, stderr=PIPE)
process2 = Popen(["col", "-b"], stdin=process.stdout, stdout=PIPE, stderr=PIPE)
return process2.communicate()[0].split('\n')

def delete_all_files():
    vip = Popen(["rm", "-r", "/home/filesystem_user/flask_test/parcial/prueba/*"], stdout=PIPE,
stderr=PIPE).communicate()
    return True

def get_recent_files():
    rF = Popen(["ls", "-lat"], stdout=PIPE, stderr=PIPE)
    list = Popen(["awk", '{print $9}'], stdin=rF.stdout, stdout=PIPE, stderr=PIPE).communicate()
    return filter(None, list)

```

Archivo mainScript.py

```

from flask import Flask, abort, request
import json

from functions import create_files, get_all_files, delete_all_files, get_recent_files

app = Flask(__name__)
api_url = '/v1.0'

@app.route(api_url+'/files', methods=['POST'])
def create_file():
    content = request.get_json(silent=True)
    filename = content['filename']
    content = content['content']
    if not filename or not content:
        return "cannot create empty file", 400
    if create_files(filename, content):
        return "HTTP 201 CREATED", 201
    else:
        return "error while creating user", 400

@app.route(api_url+'/files', methods=['GET'])
def get_filelist():
    list = {}
    list["files"] = get_all_files()
    return json.dumps(list), 200

@app.route(api_url+'/files', methods=['PUT'])
def update_user():
    return "HTTP 404 NOT FOUND", 404

@app.route(api_url+'/files', methods=['DELETE'])
def delete_files():
    if delete_all_files():
        return 'HTTP 200 OK', 200
    else:
        return 'ERROR', 200

@app.route(api_url+'/files/recently_created', methods=['POST'])
def func1():
    return "HTTP 404 NOT FOUND", 404

@app.route(api_url+'/files/do_ls', methods=['GET'])
def get_rece_files():
    return get_recent_files()

```



```

@app.route(api_url+'/files/recently_created',methods=['PUT'])
def func2():
    return "HTTP 404 NOT FOUND", 404

@app.route(api_url+'/files/recently_created',methods=['DELETE'])
def func3():
    return "HTTP 404 NOT FOUND", 404

if __name__ == "__main__":
    app.run(host='0.0.0.0',port=8088,debug='True')

```

Con los scripts realizados se procedió a realizar pruebas con la finalidad de saber si los servicios al levantarse eran accesibles.

Primero se levanta el servicio

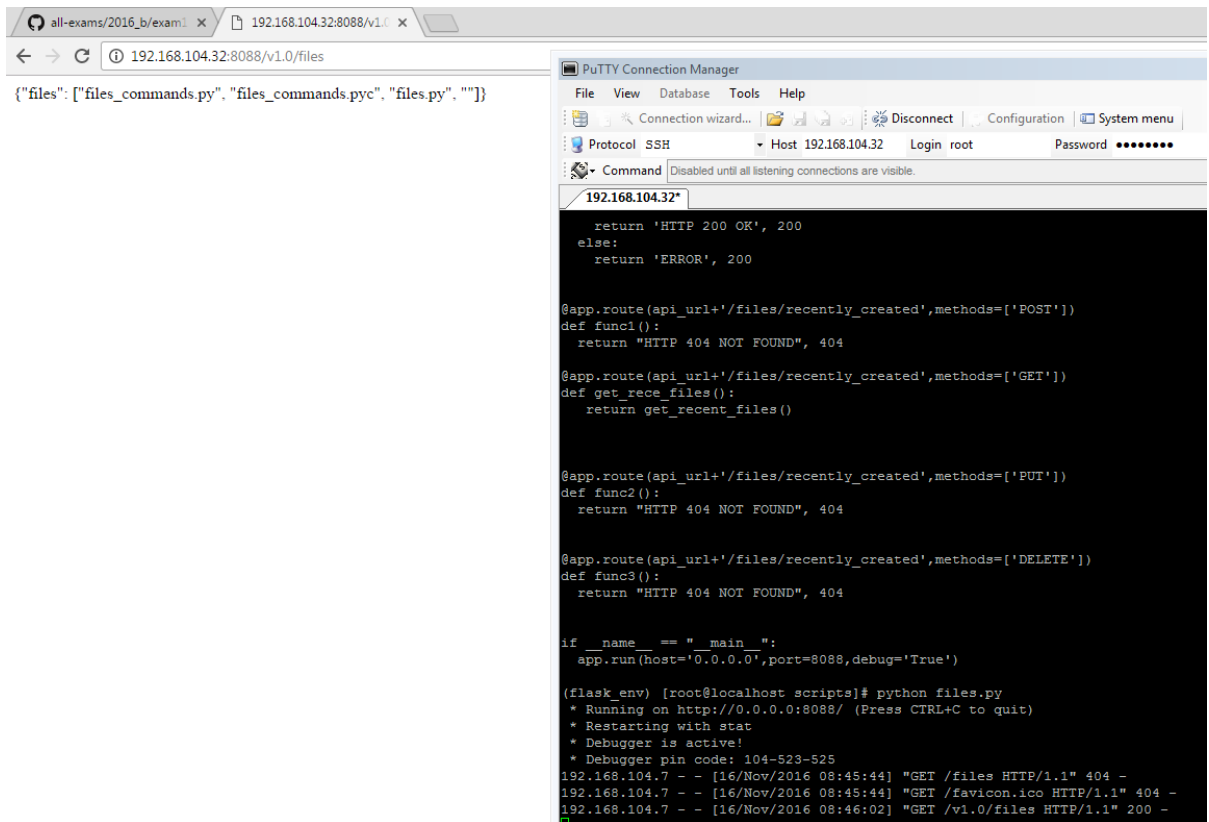
```
$ (flask_env) python mainScript.py
```

```

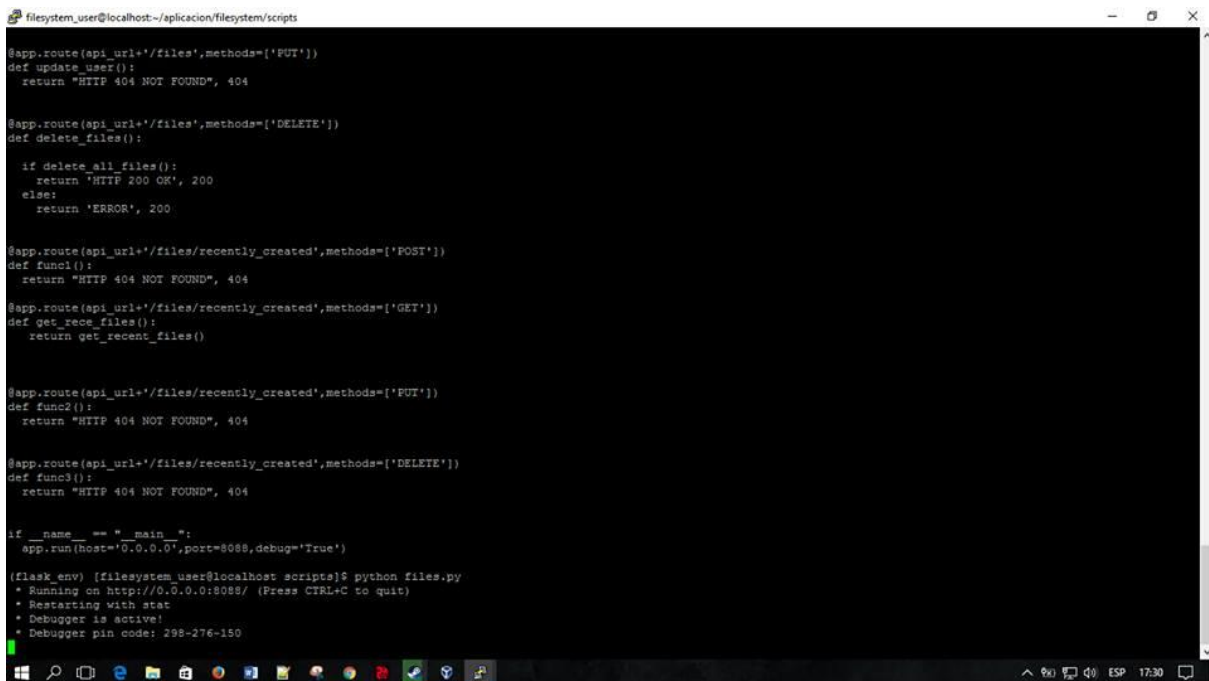
[root@localhost home]# cd filesystem_user/
[root@localhost filesystem_user]# ls
aplicacion  envs
[root@localhost filesystem_user]# cd aplicacion/+
-bash: cd: aplicacion/+: No such file or directory
[root@localhost filesystem_user]# cd aplicacion/
[root@localhost aplicacion]# ls
filesystem
[root@localhost aplicacion]# cd filesystem/
[root@localhost filesystem]# ls
images  scripts
[root@localhost filesystem]# cd ..
[root@localhost aplicacion]# cd ..
[root@localhost filesystem_user]# ls
aplicacion  envs
[root@localhost filesystem_user]# cd envs/
[root@localhost envs]# ls
flask_env  requirements.txt
[root@localhost envs]# . flask_env/bin/activate
(flask_env) [root@localhost envs]# cd ..
(flask_env) [root@localhost filesystem_user]# ls
aplicacion  envs
(flask_env) [root@localhost filesystem_user]# cd aplicacion/
(flask_env) [root@localhost aplicacion]# ls
filesystem
(flask_env) [root@localhost aplicacion]# cd filesystem/
(flask_env) [root@localhost filesystem]# ls
images  scripts
(flask_env) [root@localhost filesystem]# cd scripts/
(flask_env) [root@localhost scripts]# ls
files_commands.py  files.py
(flask_env) [root@localhost scripts]# python files.py
* Running on http://0.0.0.0:8088/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger pin code: 104-523-525

```

Después se decidió probar por medio del navegador si se podía acceder al servicio POST.

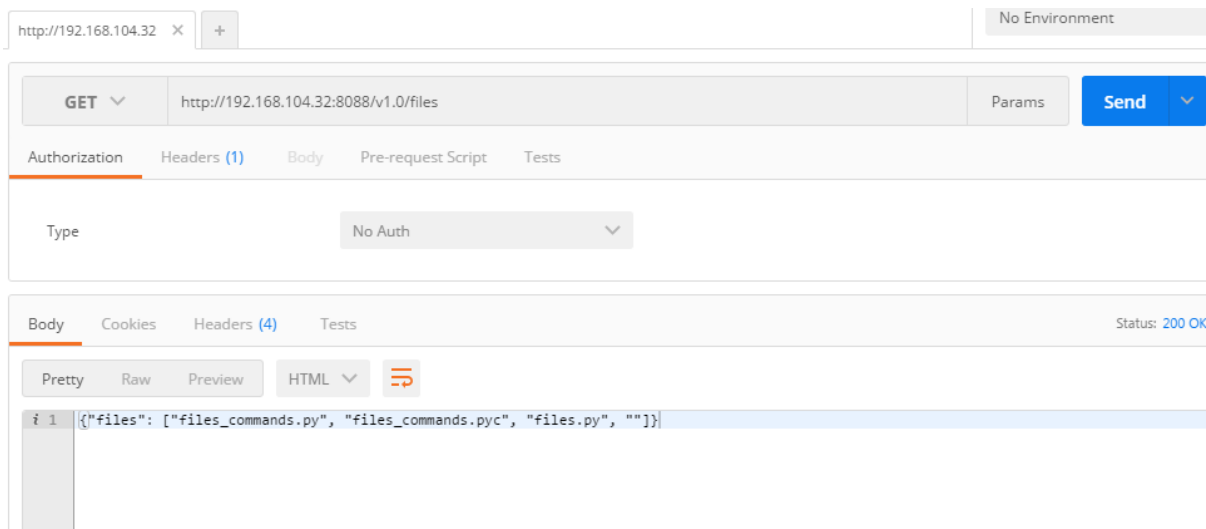


Y se pudo observar cómo se podía contactar con todas las funciones del servicio expuesto.



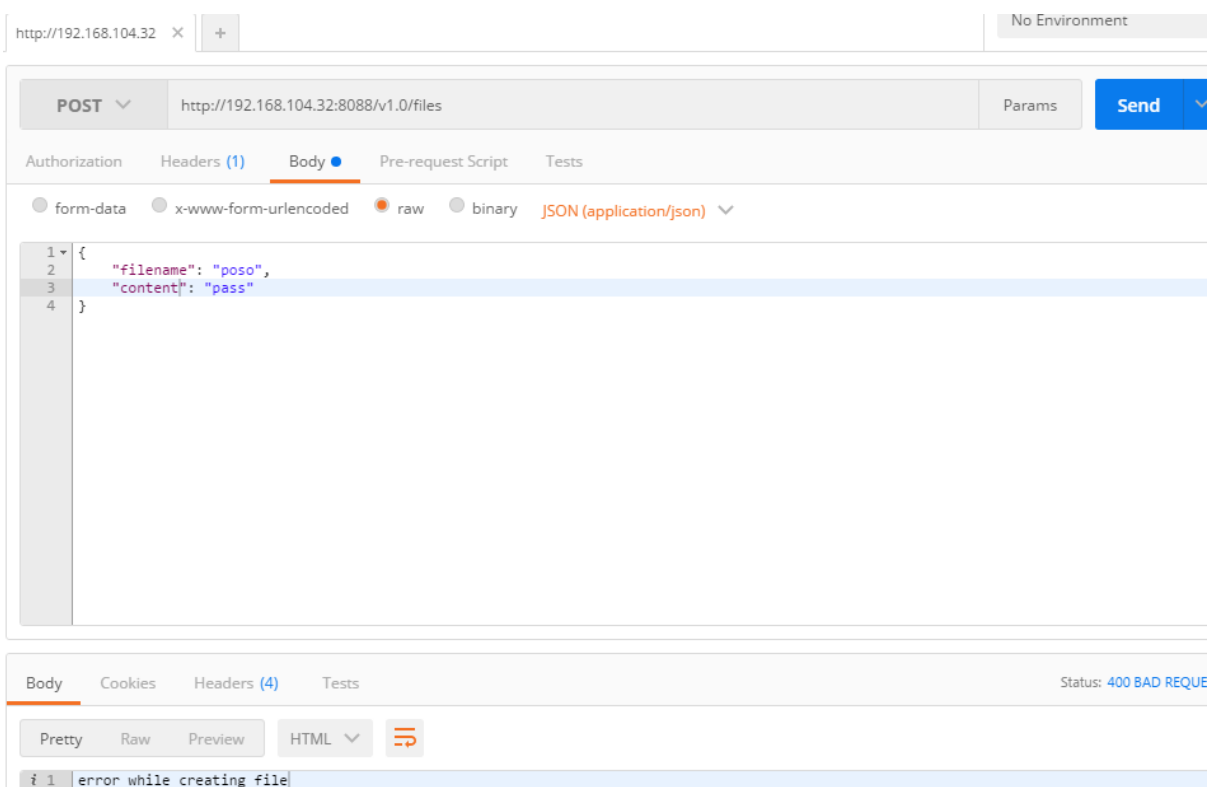
Comprobado el servicio con el navegador se comprobó el funcionamiento de los diferentes servicios por medio del programa Postman.

files GET



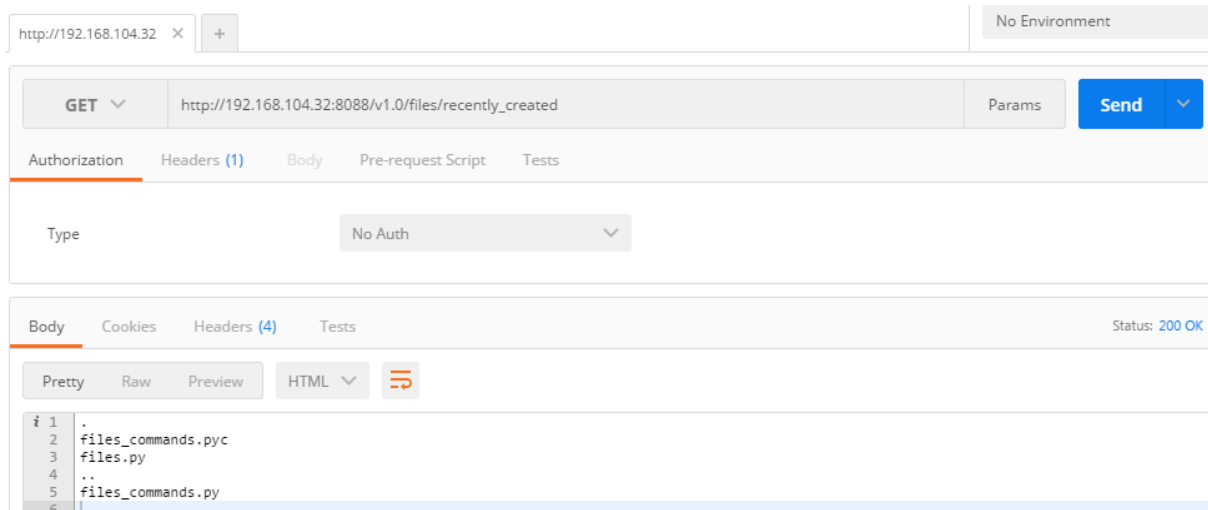
The screenshot shows a Postman GET request to the endpoint `http://192.168.104.32:8088/v1.0/files`. The request is configured with 'No Auth' and the 'Body' tab is selected. The response status is 200 OK. The response body, displayed in 'Pretty' format, is a JSON array: `[{"files": ["files_commands.py", "files_commands.pyc", "files.py", ""]}]`.

files POST



The screenshot shows a Postman POST request to the endpoint `http://192.168.104.32:8088/v1.0/files`. The request is configured with 'raw' body type and 'JSON (application/json)' content type. The request body is a JSON object: `{ "filename": "poso", "content": "pass" }`. The response status is 400 BAD REQUEST. The response body, displayed in 'Pretty' format, is an error message: `error while creating file`.

recently created GET



Validación de la ejecución del servicio (netstat)

Netstat

netstat (estadísticas de red) es una herramienta de la línea de comandos para controlar las conexiones de red entrantes y salientes, así como la visualización de las tablas de enrutamiento, las estadísticas de la interfaz, etc netstat está disponible en todos los sistemas operativos de tipo Unix y también disponibles en el sistema operativo Windows. Es muy útil en términos de resolución de problemas de red y la medición del desempeño. Netstat es una de las herramientas de depuración de servicios de red más básicas, que le dice qué puertos están abiertos y si los programas escuchan en tales puertos.

Esta herramienta es muy importante y muy útil para los administradores de red de Linux, así como para los administradores de sistemas para supervisar y solucionar los problemas relacionados con la red y determinar el rendimiento del tráfico de red. Este artículo muestra los usos del comando netstat con sus ejemplos que pueden ser útiles en la operación diaria.

Usando netstat -putona

Con el fin de realizar la comprobación de servicios se descargó e instaló el servicio netstat y se hizo uso del comando -putona, que nos permite ver que puertos despliegan un determinado proceso.

p Muestra las conexiones para el protocolo especificado que puede ser TCP o UDP u Lista todos los puertos UDP t Lista todos los puertos TCP o Muestra los timers n Muestra el número de puerto a Visualiza todas las conexiones activas del sistema

Con lo anterior se concluye que el servicio se encuentra en correcto funcionamiento y se da por finalizado el proyecto.

Referencias

<https://ubuntulife.wordpress.com/2014/03/11/netstat-putona-un-comando-que-no-olvidaras-para-monitorizar-las-conexiones-en-linux/>