Instrucciones para implantar la plataforma necesaria para las Prácticas

Versión 3.2 2021/09/14

Esta guía recoge los pasos a seguir para implantar y configurar la plataforma necesaria para realizar las prácticas de la asignatura utilizando la máquina virtual suministrada por la EPS con el software de los laboratorios, así como en una distribución Ubuntu.

En cualquier caso, **se recomienda encarecidamente** leer detenidamente los apartados en los que se describen los pasos seguir para configurar tu propio PC para la realización y ejecución de las prácticas, ya que en este proceso se aprenden bastantes detalles sobre el funcionamiento y la configuración de servidores web, bases de datos, etc.

Adaptación de la máquina virtual (VM) de los laboratorios

La EPS provee de una máquina virtual con el software de los laboratorios que se puede descargar de http://150.244.64.13, siempre que se acceda a través de la VPN de la UAM.

Si bien la máquina virtual está preparada para el software VMware, esta se puede cargar desde otros programas de virtualización como VirtualBox. Para ello, sólo es necesario crear una máquina virtual nueva y elegir la opción de usar un archivo de disco duro virtual existente, seleccionando el archivo Ubuntu1804LabsDocentes.vmdk, y realizando las configuraciones pertinentes de memoria y CPU recomendadas.

Esta máquina virtual ya dispone del software necesario para la ejecución de las prácticas de SI1, sin embargo, son necesarios unos pequeños ajustes:

recursos de CPU y memoria

La VM está preconfigurada para usar 4 CPUs y 8GB de memoria; si se está un poco escaso de recursos, se puede reducir a 2 CPUs y 2GB de memoria.

directorio $HOME/public_html

Las páginas web desarrolladas en las prácticas se publicarán, al igual que se haría en los laboratorios, en los directorios del usuario (la VM tiene el usuario ‘eps’ pre-creado, con password ‘eps’), no en los directorios del sistema.

Para ello es necesario crear el directorio $HOME/public_html (/home/eps/public_html):

```
mkdir $HOME/public_html
```

uso de python3 del sistema

La VM está preconfigurada para usar Python del paquete anaconda, en lugar del suministrado por Ubuntu. Esto da problemas al crear los entornos virtuales de python descritos más adelante. Para usar la versión de Python suministrado por Ubuntu:

```
export PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:\
/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
```

intercambio de archivos entre el PC y la VM

El intercambio de archivos entre el PC y la VM se puede hacer mediante una carpeta compartida: [abrir vmplayer] → [seleccionar VM] → Edit Virtual Machine Settings → Options → Shared Folders → Always enabled → Add…

La carpeta compartida aparece montada en el directorio `/mnt/hgfs` de la VM.

También se puede hacer mediante el mandato scp (ejecutándolo en el PC), pero entonces es necesario instalar el servidor de ssh en la VM:

```
sudo apt install openssh-server
```

Si se opta por scp, es conveniente cambiar la contraseña por defecto del usuario eps, ya que se habilita el login remoto a la VM.

Además es necesario averiguar la IP de la VM:

```
ip -4 a
```

Si se utiliza VirtualBox se puede instalar, para la máquina virtual en cuestión, las VirtualBox Guest Additions, que facilitan el intercambio de información entre la máquina virtual y las aplicaciones y el sistema operativo nativo.

uso de herramientas de la DB

El servidor de DB postgresql de la VM está preconfigurado para que sólo pueda accederse desde localhost. Por ello, el uso de herramientas como pgadmin3 o http://localhost/phppgadmin deberá realizarse desde dentro de la VM

prueba del entorno python

Puede probarse el entorno python que usaremos en las prácticas siguiendo los pasos del apartado *'Creación de un entorno python con Flask'* más abajo.

Si se hace en el directorio public_html, además de poderse probar dentro de la VM (http://localhost:5001),

también se puede acceder a la página creada desde el PC: http://<ip-VM>:5001

También puede probarse con lo descrito en la sección *'Publicar un aplicación Flask con su entorno virtual propio bajo Apache'*. De nuevo, además de acceder desde dentro de la VM (http://localhost/~eps/hello.wsgi), también se puede acceder desde el PC (http://<ip-VM>/~eps/hello.wsgi).

Nótese que, mientras que en el primer ejemplo se accede directamente desde el navegador al programa python (que está escuchando en el puerto TCP 5001), en este caso se accede a través del servidor web Apache.

MongoDB

La máquina virtual incluye una versión de MongoDB (mongo-org-server), pero vamos a instalar la versión oficial de MongoDB, un SGBD NoSQL que utilizaremos en la asignatura.

1) Instalar MongoDB:

```
sudo apt update
```

```
sudo apt install mongodb
```

Una vez instalado, **el servidor puede invocarse** con:

```
sudo mongod
```

Puede **pararse** con Crtl+C.

En una ventana aparte, **el cliente puede invocarse con:**

```
mongo
```

o bien:

```
mongo fichero_script
```

Si entras dentro de la consola con el comando `mongo`, prueba con este primer comando para ver las bases de datos que hay en ese momento y comprobar que todo funciona bien (para salir, teclea `exit`):

```
show dbs
```

El **GUI MongoDB Compass** se encuentra ya instalado, y puede utilizarse, estando el servidor activo, de la forma:

```
mongodb-compass
```

2) Instalar el módulo de MongoDb para Python:

```
pip3 install pymongo
```

Desarrollo de las prácticas en el PC propio

Si se desea realizar las prácticas en el PC, sin necesidad de instalar la VM de la EPS, se pueden seguir los pasos descritos en las secciones siguientes para instalar y configurar el software requerido para la realización de las prácticas.

IMPORTANTE: Como se ha indicado más arriba, la VM ya tiene el software requerido instalado y configurado, por lo que **los apartados siguientes no deben realizarse dentro de la VM** (salvo los apartados con los ejemplos, si1pyenv+hello.*, para probar el entorno).

Paquetes necesarios

El software instalado en los laboratorios (y la VM suministrada) es Ubuntu 18.04, con los siguientes paquetes relevantes:

- postgresql
- postgresql-server-dev
- apache2
- python3
- libpython3-dev
- Flask
- SQLAlchemy
- virtualenv
- tora
- phppgadmin
- pgadmin3
- google-chrome-stable
- firefox
- xsltproc
- libxml2-utils

Los pasos necesarios para instalarlos:

- Como tu usuario de escritorio, ejecuta:

```
$ sudo apt-get install postgresql postgresql-server-dev-10 apache2 \
libpython3-dev tora phppgadmin libqt4-sql-psql \
google-chrome-stable pgadmin3 firefox xsltproc libxml2-utils \
python3 python3-pip \
libapache2-mod-wsgi-py3 python-flask jq python-virtualenv \
```

```
python-sqlalchemy python-flask-sqlalchemy python-sqlalchemy-utils \
python-psycopg2
```

(si el paquete google-chrome da problemas, se puede descargar de la web de google)

-

Adicionalmente, puedes instalar VirtualEnvWrapper, una utilidad para la gestión de los entornos creados con VirtualEnv. Para más información: https://virtualenvwrapper.readthedocs.io/en/latest/index.html.

Se recomienda el uso de las **herramientas de desarrollador Web** que tanto Firefox como Chrome tienen incluidas y que se pueden mostrar pulsando **F12** en el navegador.

Configurar PostgreSQL

- Para que podamos utilizar el usuario "alumnodb", es necesario editar el fichero /etc/postgresql/10/main/pg_hba.conf para que contenga las siguientes líneas adicionales. **Es importante que estas líneas estén situadas delante de otras directivas de autenticación (por ejemplo: al principio del fichero).**

  ```
  local       all     alumnodb                        md5
  host        all     alumnodb    127.0.0.1/32        md5
  ```

  También se puede usar 'trust' en lugar de 'md5' como método alternativo de autenticación (no se pide contraseña).

- Asimismo, conviene editar el fichero /etc/postgresql/10/main/postgresql.conf, buscar las líneas con los valores comentados de autovacuum_vacuum_threshold, autovacuum_analyze_threshold y establecer los valores:

  ```
  autovacuum_vacuum_threshold = 5000000
  autovacuum_analyze_threshold= 5000000
  ```

- Tras esto, es necesario reiniciar el servicio ejecutando:

  ```
  $ sudo systemctl restart postgresql
  ```

- Ahora debemos crear el "rol" alumnodb en postgres. Para ello, iniciamos sesión como usuario "postgres" y creamos el rol con los siguientes comandos:

  ```
  $ sudo su - postgres
  ```

- y después:

  ```
  $ createuser -s alumnodb
  ```

- Para dar de alta el lenguaje plpgsql en todas las nuevas bases de datos, como usuario postgres, ejecutar (normalmente no es necesario porque ya está dado de alta):

  ```
  $ createlang plpgsql template1
  ```

- Para acceder a la base de datos, y asignar una contraseña (necesaria para acceder desde phppgadmin; introducir 'alumnodb' como contraseña)

```
$ psql
\password alumnodb
...
\q
```

- Por último, para comprobar que todo está correcto, creamos una base de datos. Como nuestro usuario de inicio de sesión, ejecutamos:

```
$ createdb -U alumnodb bdat
```

También conviene instalar el programa schemaSpy, http://schemaspy.sourceforge.net, (y schemaSpyGUI, http://sourceforge.net/projects/schemaspygui) para obtener diagramas de las bases de datos desarrolladas.

Creación de un entorno python con Flask

Para la creación en un entorno python en el que probar páginas desarrolladas en Flask, ejecutar los siguientes mandatos:

```
$ virtualenv -p python3 si1pyenv

$ source si1pyenv/bin/activate

$ pip3 install Flask SQLAlchemy Flask-SQLAlchemy SQLAlchemy-Utils \
  psycopg2 itsdangerous Flask-Session

$ deactivate
```

Nota importante: Las prácticas se realizarán con python versión 3.

Para probar el entorno, crear el fichero *hello.py* con el siguiente contenido:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-


from flask import Flask
app = Flask(__name__)
@app.route("/")

def hello():
```

```
    return "Hello World!"
if __name__ == "__main__":
    app.run(host='0.0.0.0', port=5001, debug=True)
```

A continuación, ejecutar el programa con los siguientes mandatos:

```
$ source si1pyenv/bin/activate
$ python3 hello.py
```

Accediendo a http://localhost:5001 debería aparecer el mensaje 'Hello World!'

Nota importante: Téngase presente que al crear el entorno si1pyenv se introducen rutas absolutas en ficheros que se generan, principalmente en el directorio si1pyenv/bin. Esto hace que si se mueve (o se hace backup y se restaura) el directorio si1pyenv a otra localización dejará de funcionar.

Configurar Apache para que muestre páginas de usuario

Ahora vamos a configurar el servidor web Apache para publicar la carpeta personal public_html en la ruta http://localhost/~usuario/. Para ello:

- Crear el directorio "public_html" dentro de $HOME (carpeta personal: /home/*usuario*)

  ```
  $ mkdir ~/public_html
  ```

- Ejecutar este mandato:

  ```
  $ sudo a2enmod userdir
  ```

- Tras esto, es necesario reiniciar el servicio ejecutando:

  ```
  $ sudo service apache2 restart
  ```

- Ya se puede acceder al contenido de ésta en http://localhost/~usuario/, donde "usuario" es el que ha iniciado la sesión.

Configurar Flask bajo Apache

Para poder ejecutar páginas python es necesario instalar y configurar el módulo mod_wsgi de Apache según lo indicado más arriba (mandato '`sudo apt-get install …`' más arriba). También se puede consultar http://flask.pocoo.org/docs/0.12/deploying/mod_wsgi/

Para configurar el acceso a páginas python seguir los siguientes pasos:

- Ejecutar el siguiente mandato:

  ```
  $ sudo a2enmod wsgi
  ```

- Editar el archivo `/etc/apache2/mods-enabled/userdir.conf` y añadir ExecCGI a las opciones y el handler de wsgi:

  ```
  Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec ExecCGI
  AddHandler wsgi-script .wsgi
  ```

- Tras esto, es necesario reiniciar el servicio ejecutando:

  ```
  $ sudo systemctl restart apache2
  ```

Publicar un aplicación Flask con su entorno virtual propio bajo Apache

Como hemos indicado anteriormente, para publicar páginas web desarrolladas en python es necesario el módulo de Apache mod_wsgi, que hace de puente entre el servidor Web (Apache) y la aplicación web en python (Flask).

Para ello es necesario crear un fichero .wsgi como el siguiente, al que llamaremos *hello.wsgi*, que además hace uso del entorno virtual creado anteriormente:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import os
import sys

# virtualenv
this_dir = os.path.dirname(os.path.abspath(__file__))
activate_this = this_dir + '/si1pyenv/bin/activate_this.py'
exec(open(activate_this).read(), dict(__file__=activate_this))

# anhadir dir de este fichero a path
sys.path.insert(0, this_dir)

from hello import app as application
```

Si colocamos el fichero hello.py y este nuevo hello.wsgi en el directorio public_html y creamos aquí un entorno si1pyenv según las instrucciones que ya hemos visto, accediendo a la URL http://localhost/~<usuario>/hello.wsgi nos aparecerá el mensaje 'Hello World!' de la aplicación.

Trucos de PostgreSQL

Se recomienda el uso de las herramientas de gestión:

- $ tora
- $ pgadmin3
- http://localhost/phppgadmin/

Para entrar en el entorno interactivo de psql y obtener ayuda:

```
$ psql –U alumnodb bdat
bdat=#help
bdat=#\help
bdat=#\?
bdat=#\q
```

Para evitar tener que introducir usuario, contraseña y base de datos:

```
$ export PGUSER=alumnodb
$ export PGPASSWORD=alumnodb
$ export PGDATABASE=si1
```

Para ejecutar un script:

```
$ cat miscript.sql | psql –U alumnodb bdat
```

Para crear una base de datos:

```
$ createdb –U alumnodb dbname
```

Para volcar el contenido (y estructura) de una base de datos:

```
$ pg_dump –U alumnodb bdat > outputfile.sql
```

El fichero resultante es un script que se puede usar para recrear la base de datos, ejecutándolo tal y como

se indica más arriba.

Para realizar una carga masiva de datos se puede usar el mandato COPY. Ver el contenido del script de salida del mandato pg_dump.

```
create table profesor (id int, nombre char(20), apellido char(20));
COPY profesor from stdin using delimiters '|';
1|Pedro|Pascual
2|Julia|Díaz
\.
```

Para ver los logs de postgresql:

```
$ sudo su - postgres
$ ls -al /var/log/postgresql/
$ cat /var/log/postgresql/postgresql*.log
```

MongoDB

Para una instalación desde cero de MongoDB, los pasos a seguir son los siguientes:

1) Instalar MongoDB:

```
sudo apt update
sudo apt install mongodb
```

2) Crear la carpeta donde se almacenarán las bases de datos:

```
sudo mkdir -p /data/db
```

3) Instalar la GUI MongoDB-Compass:

Se debe descargar de la página web:

https://www.mongodb.com/try/download/compass

eligiendo la versión del Sistema operativo deseado (Ubuntu). Una vez descargada, se debe proceder a su instalación localmente.

4) Arrancar el servidor MongoDB:

El servidor de MongoDB se arranca automáticamente después de la instalación. No obstante, se puede comprobar y alterar su estado con los comandos:

```
sudo systemctl status mongodb
```

```
sudo systemctl stop mongodb
```

```
sudo systemctl start mongodb
```

El cliente puede invocarse con:

`mongo` o bien

```
mongo fichero_script
```

Si entras dentro de la consola con el comando `mongo`, prueba con este primer comando para ver las bases de datos que hay en ese momento y comprobar que todo funciona bien (para salir, teclea `exit`):

```
show dbs
```

5) Instalar el módulo de MongoDb para Python:

```
pip3 install pymongo
```

Enlaces de referencia

- http://www.w3schools.com
- http://www.python.org
- http://www.postgresql.org
- http://flask.pocoo.org
- https://www.sqlalchemy.org/
- https://www.mongodb.com/