Tutorial Publicar Sitio Python-Django en Server Centos 7

Requisitos:

Centos 7 Con Apache Instalado – No es necesario ROOT – WinSCP (transferencia de archivos si no se tiene acceso directo al server) – Putty (comandos Linux si no se tiene acceso directo al server).

Para comenzar primero hay que tener en cuenta lo siguiente:

- Versión del Python que el desarrollador tiene instalado (en este caso es el 3.8). Esto es muy importante ya que el entorno virtual (ENV) se crea en base al Python instalado. Recordemos que según la versión del ENV serán las versiones soportadas de Django (y sus correspondientes complementos).
- Archivo **requirements.txt.** En la mayoría de los tutoriales menciona que se debe instalar el DJANGO directamente desde el PIP. Pero para respetar la versión de DJANGO y los complementos instalados en la máquina del desarrollador se debe realizar a través de este archivo.

Paso 1 – Instalar Python

Cada desarrollador instalará la versión de Python que estime conveniente, es muy importante instalar en el servidor cada versión de Python que cada desarrollador utilice (al menos que se cree algún protocolo).

Si ya se tiene instalado, omitir este paso.

En mi caso utilizamos Python 3.8 y para instalarlo se debe realizar lo siguiente:

1.- Instale los paquetes requeridos

```
sudo yum install gcc openssl-devel bzip2-devel libffi-devel
```

2.-Descarga Python 3.8

```
cd / opt

wget https://www.python.org/ftp/python/3.8.0/Python-3.8.0.tgz
```

Nota: Si el comando wget no funciona descargar el tgz y dejar en opt/.

Ahora extraiga el paquete descargado.

```
tar xzf Python-3.8.0.tgz
```

3.- Compilar Python Source

```
cd Python-3.8.0

sudo ./configure --enable-optimizations

sudo make altinstall
```

hacer altinstall se usa para evitar la sustitución del archivo binario predeterminado de python / usr / bin / python.

```
sudo rm Python-3.8.0.tgz
```

4.- Verificar la versión de Python

```
python3.8 -V

Python 3.8.0
```

Paso 2 – Crear Proyecto Python y su entorno virtual.

Crear Proyecto Python: Para levantar un proyecto en Python solo es necesario crear una carpeta en un directorio a elección, para el ejemplo lo crearemos en la carpeta /opt/.

Desde consola:

```
$ mkdir nombre_proyecto
```

Desde WinScp, solo botón derecho y crear carpeta.

Una vez creada la carpeta del proyecto, creamos su entorno virtual, es importante que se haga en la versión de Python que trabaja el desarrollador en este caso la 3.8.

Para crear el entorno virtual:

Posicionarse dentro de la carpeta del proyecto y en consola escribir:

```
$ python3.8 -m venv nombre_entorno_virtual
```

Nota: Por buena práctica el entorno virtual se denomina ENV o VENV.

Paso 3 – Instalar Requirements.txt.

A lo largo de la creación del sistema el desarrollador instalará una serie de componentes con sus respectivas versiones al proyecto django. Una forma de mantener estas configuraciones desde la máquina del desarrollador al servidor es a través del archivo requirements.txt, **este lo debe generar el desarrollador** y luego se debe instalar en el proyecto que tenemos en el servidor.

Crear archivo requirements.txt

```
pip freeze
```

```
(env) C:\Users\jrodriguez\PycharmProjects\MiProyectoPython>pip freeze
cymysql==0.9.14
Django==2.2.7
django-ajax-selects==1.7.1
django-cymysql==2.2.0
django-js-asset==1.2.2
django-mptt==0.10.0
django-python3-ldap==0.11.2
```

Luego el resultado se copia y pega en un archivo txt llamado requirements.txt

Cargar Archivo requirements.txt

El archivo creado anteriormente dejarlo en la ruta donde se ejecute el pip.

```
pip install -r requirements.txt
```

Paso 4 – Crear proyecto Django.

Al instalar el paso 3, habremos instalado la versión de Django utilizada por el desarrollador, luego de esto estamos en condiciones de crear un proyecto Django dentro del proyecto Python.

Crear proyecto Django en centos.

Para esto lo primero que debemos fijarnos es en activar nuestro entorno virtual:

```
Source entorno_virtual/bin/activate
```

Debiese verse así: `(env) [root@buvmweblnxd001 env]#`

Una vez que esté activo

```
#django-admin.py startproject myfirstproject .
```

Es muy importante el punto al final.

Con esto ya creamos un proyecto django.

Paso 5– Configurar Archivo settings.py

Cambio opciones de static

```
#nano proyecto_django/settings.py
```

Ir al archivo settings.py cambiar la siguiente línea de código y debería quedar así.

```
STATIC_URL = '/static/'
STATIC_ROOT = os.path.join(BASE_DIR, 'static_root')
STATICFILES_DIRS = [os.path.join(BASE_DIR, 'static'),]
```

Además se debe agregar la siguiente línea:

(Solo una)

```
ALLOWED_HOSTS = ['*'] -> Para todas las IP o
ALLOWED_HOSTS = [IP_SERVER] -> Para todas la ip del server

En este caso ambas funcionan, pero me imagino que la segunda
es más segura.
```

Test

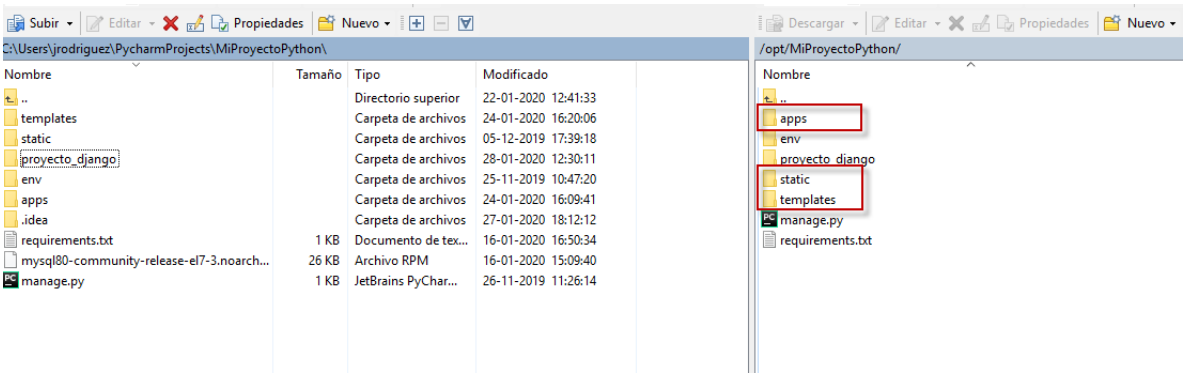
```
(djangoprojectenv) [root@centOS-7 djangoproject]#./manage.py runserver
0.0.0.0:8989
```

Paso 6 – Traspaso de Archivos

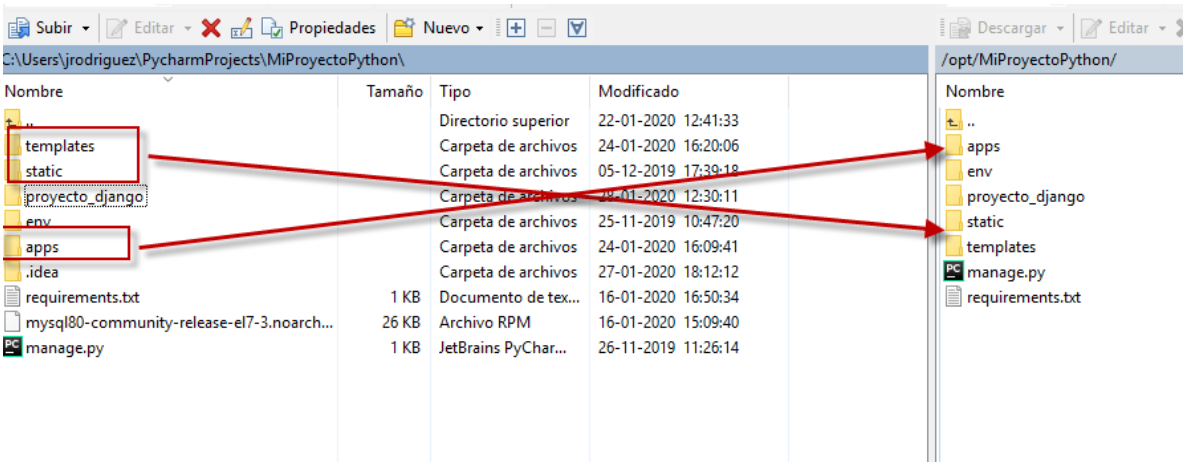
Antes de realizar la configuración de la base y la migración debemos tener todos los archivos en nuestro servidor, esto es muy importante ya que si esto no se realiza correctamente la migración fallará.

Lo más recomendable en este caso es lo siguiente:

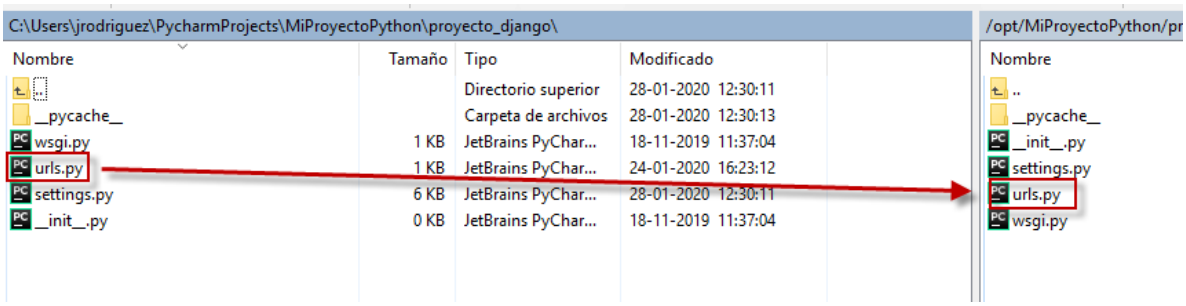
1.- Borrar las carpetas apps, static, templates desde la carpeta de proyecto en el server.



2.- Luego traspasar las carpetas apps, static, templates desde la carpeta de proyecto en local.



3.- Luego se debe reescribir el archivo urls, ubicado en el proyecto django, esto es necesario ya que en el desarrollo se agregan urls del proyecto.



4.- Luego corroboramos que el archivo settings.py del server estén todas las INSTALLED_APPS del archivo local. No es recomendable reemplazar ese archivo, ya que posee configuraciones específicas si está en local o si está en servidor. Para lo anterior solo basta abrir el archivo settings.py en ambos ambientes y corroborar que sus installed_apps sean iguales.

```
INSTALLED_APPS = [

    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
```

Paso 6 – Configurar la Base de Datos.

Una vez realizado el traspaso de archivos y antes de realizar la migración, en el archivo settings.py debiese estar configurada la base de datos donde se realizará la misma. En este caso tengo la base de datos es de un servidor Centos. Esta configuración se realiza en el archivo **settings.py**

En este caso configuramos el server a través de la IP.

Paso 7 – Migrar la Base de Datos.

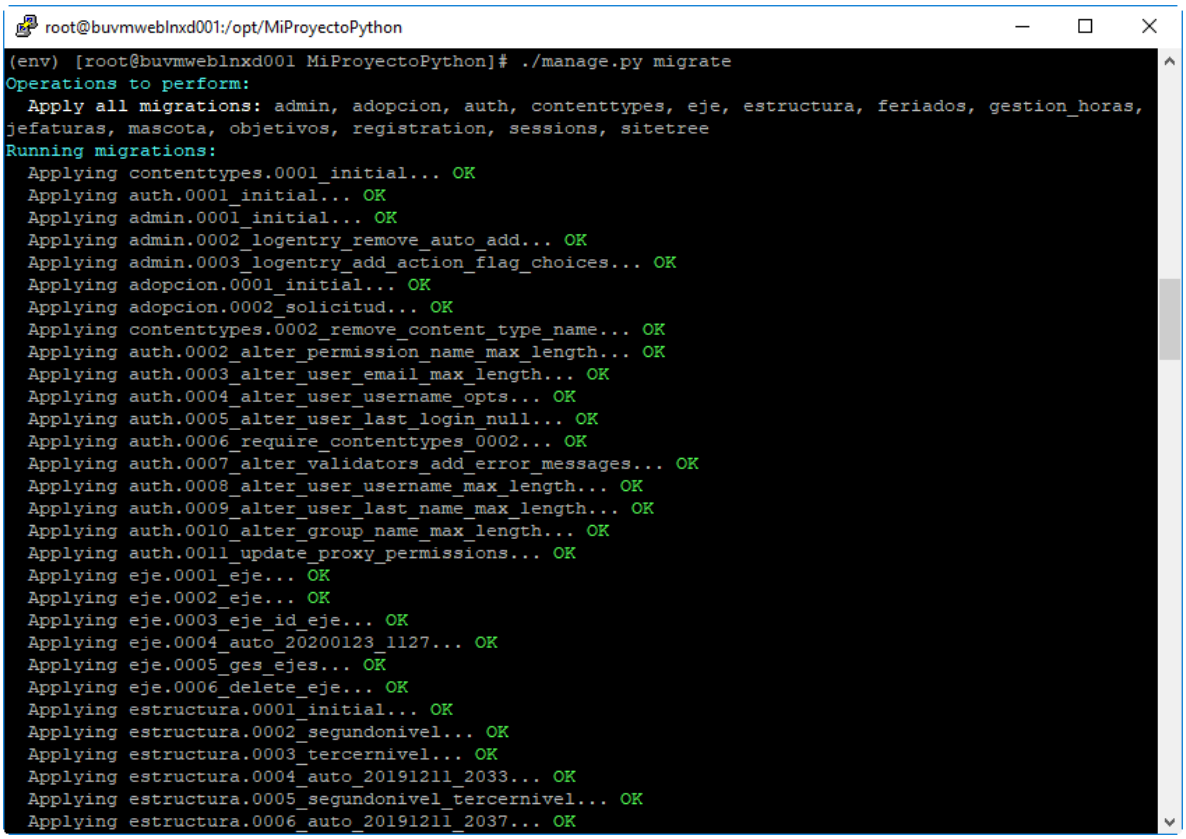
Luego que se configure la base de datos y los archivos del sitio será necesario realizar una actualización de la base de datos en el caso que esta exista y posea tablas. O simplemente una migración completa en el caso que solo tengamos la base de datos.

1.- Actualizar la base de datos.

```
./manage.py makemigrations
./manage.py migrate
```

2.- Migrar base de datos completa.

```
./manage.py migrate
```



Paso 6 – Recopilar contenido estático (aún no se si es necesario).

```
./manage.py collectstatic
```

Paso 11 – Agregar IP del server al request.py

Para agregar la IP del server al request existen dos formas:

1.- Desde la consola:

Abrir con Nano u otro editor el archivo request.py

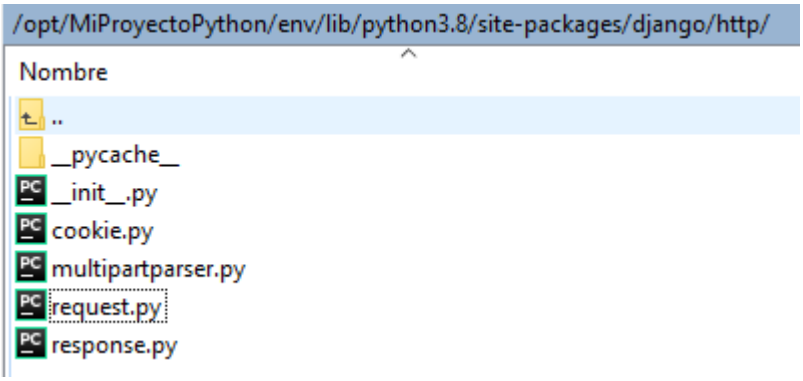
```
#nano djangoprojectenv/lib64/python2.7/site-
packages/django/http/request.py
```

Modificar la línea

```
_hosts = ['localhost', 'IP_DEL_SERVIDOR', '[::1]']
```

2.- Desde el winSCP.

Abrir archivo request.py



Modificar Linea

```
# Allow variants of localhost if ALLOWED_HOSTS is empty and DEBUG=True.
allowed_hosts = settings.ALLOWED_HOSTS
if settings.DEBUG and not allowed_hosts:
    allowed_hosts = ['localhost', '127.0.0.1', '[::1]']

domain, port = split_domain_port(host)
```

Paso 12 – Configurar Apache para Django

Después de crear un proyecto Django, deberá configurar el servidor web Apache para Django. Puede hacer esto creando un nuevo archivo de configuración:

```
sudo nano /etc/httpd/conf.d/django.conf
```

Agregue las siguientes líneas:

```
Alias /static /opt/djangoproject/static
<Directory /opt/djangoproject/static>
 Require all granted
</Directory>
```

```
<Directory /opt/djangoproject/myfirstproject>
   <Files wsgi.py>
       Require all granted
   </Files>
</Directory>

WSGIDaemonProcess myfirstproject python-
path=/opt/djangoproject:/opt/djangoproject/djangoprojectenv/lib/python2.7/site-
packages
WSGIProcessGroup myfirstproject
WSGIScriptAlias / /opt/djangoproject/myfirstproject/wsgi.py
```

Guarde y cierre el archivo cuando haya terminado, luego reinicie el servicio Apache y permita que se inicie en el arranque:

```
sudo systemctl restart httpd
sudo systemctl enable httpd
```

Permitir acceso al puerto 80 a través de `firewalld`:

```
firewall-cmd --zone=public --permanent --add-port=80/tcp
```

Establezca la propiedad adecuada para que `httpd`tenga permiso para usar el directorio del proyecto Django:

```
sudo chown -R apache:apache /opt/djangoproject
```

Paso 12 – Anexo – sincronizar django – python3 -ldap.

Nota: La siguiente es la configuración en el setting para conectar con el ldap

```
LDAP_AUTH_URL = "ldap://zeus.ine.cl"
LDAP_AUTH_USE_TLS = False
LDAP_AUTH_SEARCH_BASE = "ou=INE,DC=ine,DC=cl"
LDAP_AUTH_OBJECT_CLASS = "organizationalPerson"

LDAP_AUTH_USER_FIELDS = {
"username": "sAMAccountName",
"first_name": "givenName",
"last_name": "sn",
"email": "mail",
}


LDAP_AUTH_USER_LOOKUP_FIELDS = ("username",)

#LDAP_AUTH_CLEAN_USER_DATA = "django_python3_ldap.utils.clean_user_data"

#LDAP_AUTH_SYNC_USER_RELATIONS = "django_python3_ldap.utils.sync_user_relations"

#LDAP_AUTH_FORMAT_SEARCH_FILTERS =
"django_python3_ldap.utils.format_search_filters"

LDAP_AUTH_FORMAT_USERNAME =
"django_python3_ldap.utils.format_username_active_directory"

LDAP_AUTH_ACTIVE_DIRECTORY_DOMAIN = "INE"

#LDAP_AUTH_CONNECTION_USERNAME = 'jrodriguez'
#LDAP_AUTH_CONNECTION_PASSWORD = '(ine2026)'
```

Para sincronizar primero descomentar las últimas 2 lineas

```
#LDAP_AUTH_CONNECTION_USERNAME = 'jrodriguez'
#LDAP_AUTH_CONNECTION_PASSWORD = '(ine2026)'
```

, luego ejecutar el siguiente comando.

python manage.py ldap_sync_users