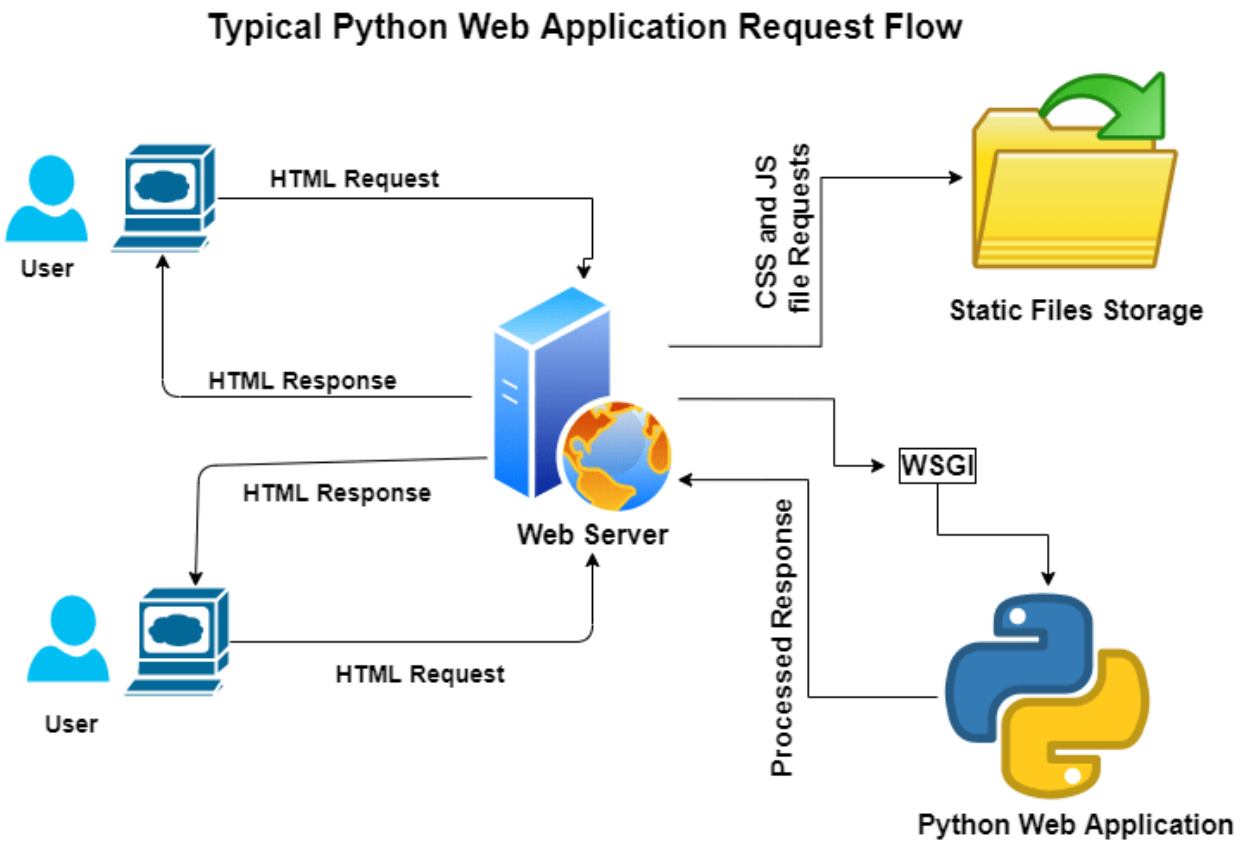
How to deploy python application



Install Python in CentOS

~# yum install gcc openssl-devel bzip2-devel libffi libffi-devel make

~# cd /opt/

~# wget https://www.python.org/ftp/python/3.7.0/Python-3.7.0.tgz

~# tar xzf Python-3.7.0.tgz

~# cd Python-3.7.0

~# ./configure --enable-optimizations

~# make altinstall

~# python3.7 -V

From yum repository:

~# yum install python

Install Python in Ubuntu

root@testserver1:~# sudo apt install build-essential zlib1g-dev libncurses5-dev libgdbm-dev libnss3-dev libssl-dev libreadline-dev libffi-dev wget

root@testserver1:~# sudo apt install python3

root@testserver1:~# python3 --version

Python 3.8.10

root@testserver1:~# apt install python3-virtualenv

Deploy our python project

Download demo project from internet

~# git clone https://github.com/vijaythapa333/django-student-management-system.git

Create a virtual environment/box just once:

~# virtualenv venv

Log-in to the virtual environment/box:

~# source venv/bin/activate

Install all necessary software that mentioned at requirements.txt:

(venv)~# pip install -r requirements.txt

(venv)~# pip install gunicorn

(venv)~# pip install config

If that's give an error then run manually:

(venv)~# pip install all_packages_one_by_one

Note:

If any permission related issue face in “VENV” file

sudo chmod -R a+rwx “Envfilename”

If any new code has any ***database change then migrate*** that db as below:

(venv)~# python manage.py migrate

If any new ***code has any static data change*** then migrate that db as below:

(venv)~# python manage.py collectstatic

Compress the static data:

(venv)~# python manage.py compress

Test

~# python manage.py runserver 0.0.0.0:8000

Log-out to the virtual environment/box:

(venv)~# deactivate

Install supervisor

CentOS

~# yum install supervisor -y

Ubuntu

root@testserver1:~# apt install supervisor -y

~# supervisord -v

~# vim /etc/supervisord.conf

Or

root@testserver1:~# vim /etc/supervisor/supervisord.conf

[program:***myApplication***]

directory = /var/www/myApplication

command = /var/www/project/venv/bin/gunicorn --bind 0.0.0.0:8000 -t 180 --graceful-timeout 180 --worker-connections=1000 --workers=5 --log-level=DEBUG ***wsgiFolderName***.wsgi:application

stdout_logfile = /var/log/myApplication.log

redirect_stderr = true

~# service supervisord restart

Or

root@testserver1:~# service supervisor restart

Note:

- ***wsgiFolderName*** is is the application folder name
- Search wsgi.py file. Here, ***wsgiFolderName*** *will be* **“**wsgi.py” folder name.

Start the service

~# supervisorctl

supervisor> start myApplication

supervisor> status

Extra: Create a daemon

If you don’t want to use supervisor then create a daemon,

~# vim /etc/init.d/myApplication

PATH=/bin:/usr/bin:/sbin:/usr/sbin

APPNAME=report

USER=apache

APPDIR=/var/www/your_project_directory

APPMODULE=report.wsgi

PORT=9096

WORKERS=2

DAEMON=gunicorn

RUN=/var/www/data-portal/portal_v1/venv/bin/gunicorn

HOST=127.0.0.1

BIND=$HOST:$PORT

PIDFILE=/var/run/$APPNAME.pid

```
LOGFILE=/var/log/$APPNAME.log
source /etc/init.d/functions
if [ -e "/etc/default/$APPNAME" ]
then
. /etc/default/$APPNAME
fi
case "$1" in
start)
log_daemon_msg "Starting deferred execution scheduler" "$APPNAME"
cd $APPDIR
$RUN --bind=$BIND --pid=$PIDFILE --workers=$WORKERS --log-file=$LOGFILE
$APPMODULE &
log_end_msg $?
;;
stop)
log_daemon_msg "Stopping deferred execution scheduler" "APPNAME"
killproc -p $PIDFILE $DAEMON
log_end_msg $?
;;
force-reload|restart)
$0 stop
$0 start
;;
status)
status_of_proc -p $PIDFILE $DAEMON && exit 0 || exit $?
;;
```

-)

```
echo "Usage: /etc/init.d/$APPNAME {start|stop|restart|force-reload|status}"
exit 1
;;
esac
exit 0
```

Start your python program:

```
~# /etc/init.d/myApplication stop
~# /etc/init.d/myApplication start
```