# vi /etc/nginx/nginx.conf

Add the following line to **http or server or location context** to increase the size limit in nginx.conf, enter:

*# set client body size to 2M #*

client\_max\_body\_size 2M;

Now restart Server

sudo /etc/init.d/nginx restart

Install

sudo apt-get install apache2 mysql-client mysql-server

Webservices

sudo apt-get install libapache2-mod-wsgi

First, you'll need to run:

sudo apt-get install apache2 mysql-client mysql-server

Once you do that, you'll get the start up page for MySQL, where you will need to set your root user for MySQL. This is the specific MySQL root user, not your server root user.

That setup should take about 20-30 seconds. After that, we need to get WSGI, so run the following:

sudo apt-get install libapache2-mod-wsgi

Once we have that, we need to make sure we've enabled WSGI with the following:

sudo a2enmod wsgi

It is probably already enabled from the installation, but it is a good idea to make sure.

Next we are ready to set up our Flask environment.

Run:

cd /var/www/

Now let's make our Flask environment directory:

mkdir FlaskApp

Move into that directory:

cd FlaskApp

Now make the actual application directory:

mkdir FlaskApp

Now let's go in there:

cd FlaskApp/

Now we're going to make two directories, static and template:

mkdir static

mkdir templates

Now we're ready to create the main file for your first Flask App:

nano \_\_init\_\_.py

Here is where we have our initialization script for our Flask application. You can actually keep all of your main website code right here for simplicity's sake, and that's what we'll be doing. Within your \_\_init\_\_.py file, you will type:

from flask import Flask

app = Flask(\_\_name\_\_)

@app.route('/')

def homepage():

    return "Hi there, how ya doin?"

if \_\_name\_\_ == "\_\_main\_\_":

    app.run()

Press control+x to save it, yes, enter.

Now we should probably actually get Flask. Let's do that now.

Since this is likely a new server for you, you will want to go ahead and run:

apt-get update

apt-get upgrade

To get Flask, we're going to use pip, so you will need to first get pip if you do not already have it:

apt-get install python-pip

Now that we have pip, we also need virtualenv to create the virtual environment for Flask to run Python and your application in:

pip install virtualenv

Now to set up the virtualenv directory:

sudo virtualenv venv

Activate the virtual environment:

source venv/bin/activate

Now install Flask within your virtual environment

pip install Flask

Find out if everything worked out by going:

python \_\_init\_\_.py

If you didn't get any major errors, congrats!

Hit control+c to get out of the running text, then type deactivate to stop the virtual environment running locally. This is only a local version, so you wont be able to type in anything to your browser to access it.

So now we need to set up our Flask configuration file:

nano /etc/apache2/sites-available/FlaskApp.conf

This is where your Flask configuration goes, which will apply to your live web site. Here's the code that you need to include:

<VirtualHost \*:80>

                ServerName [yourdomain.com](http://yourdomain.com/)

                ServerAdmin [youemail@email.com](mailto:youemail@email.com)

                WSGIScriptAlias / /var/www/FlaskApp/flaskapp.wsgi

                <Directory /var/www/FlaskApp/FlaskApp/>

                        Order allow,deny

                        Allow from all

                </Directory>

                Alias /static /var/www/FlaskApp/FlaskApp/static

                <Directory /var/www/FlaskApp/FlaskApp/static/>

                        Order allow,deny

                        Allow from all

                </Directory>

                ErrorLog ${APACHE\_LOG\_DIR}/error.log

                LogLevel warn

                CustomLog ${APACHE\_LOG\_DIR}/access.log combined

</VirtualHost>

For your notes, if you want to add more domains/subdomains that point to the same Flask App, or a different app entirely, you can use a ServerAlias, added underneath the ServerAdmin line.

We are now ready to enable the server.

Run:

sudo a2ensite FlaskApp

service apache2 reload

Almost there... now we just need to configure our WSGI file. To do this:

cd /var/www/FlaskApp

nano flaskapp.wsgi

Within the wsgi file, enter:

#!/usr/bin/python

import sys

import logging

logging.basicConfig(stream=sys.stderr)

sys.path.insert(0,"/var/www/FlaskApp/")

from FlaskApp import app as application

application.secret\_key = 'your secret key. If you share your website, do NOT share it with this key.'

Save and exit.

Once that is done, run:

service apache2 restart

Get used to running the above command. Flask is very finicky about your python file changes. Every .py file change you make to your webapp, you need to run this command.

Uninstall

sudo apt-get remove apache2

sudo apt-get remove --auto-remove apache2

sudo apt-get purge apache2

sudo apt-get purge --auto-remove apache2

For making a folder virtual env

virtualenv env

For activating virtual env

. env/bin/activate

For installing pip

Easy\_install pip

For installing virtualenv lib

easy\_install virtualenv

After installing python3, to create new virtual environment

virtualenv -p python3 myenv

For Activating virtual environment

source myenv/bin/activate

**Setup**

Let’s get the basic configuration setup.

**Add a new User**

After SSH’ing into the server as the ‘root’ user, run-

$ adduser newuser

$ adduser newuser sudo

-to create a new user with ‘sudo’ privileges.

**Install the Requirements**

SSH into the server with the new user, and then install the following packages:

$ sudo apt-get update

$ sudo apt-get install -y python python-pip python-virtualenv nginx gunicorn

**Set up Flask**

Start by creating a new directory, “/home/www”, to store the project:

$ sudo mkdir /home/www **&&** cd /home/www

Then create and activate a virtualenv:

$ sudo virtualenv env

$ source env/bin/activate

Install the requirements:

$ sudo pip install Flask**==**0.10.1

Now set up your project:

$ sudo mkdir flask\_project **&&** cd flask\_project

$ sudo vim app.py

Add the following code to *app.py*:

from flask import Flask, jsonify

app **=** Flask(\_\_name\_\_)

@app.route('/')

def **index**():

    return 'Flask is running!'

@app.route('/data')

def **names**():

    data **=** {"names": ["John", "Jacob", "Julie", "Jennifer"]}

    return jsonify(data)

if \_\_name\_\_ **==** '\_\_main\_\_':

    app**.**run()

**Within VIM, press “i” to enter the INSERT mode. Add the code, then press “escape” to leave INSERT mode to go into COMMAND mode. Finally type “:wq” to save and exit VIM.**

Set up a static directory-

$ sudo mkdir static

-and then add an *index.html* (sudo vim static/index.html) file with the following html:

**<h1>**Test!**</h1>**

**Configure nginx**

Start nginx:

$ sudo /etc/init.d/nginx start

Then:

$ sudo rm /etc/nginx/sites-enabled/default

$ sudo touch /etc/nginx/sites-available/flask\_project

$ sudo ln -s /etc/nginx/sites-available/flask\_project /etc/nginx/sites-enabled/flask\_project

Here, we remove the default nginx configuration, create a new config file (called *flask\_project*), and, finally, set up a symlink to the config file we just created so that nginx loads it on startup.

Now, let’s add the config settings to *flask\_project*:

$ sudo vim /etc/nginx/sites-enabled/flask\_project

Add:

server **{**

    location / **{**

        proxy\_pass [http://localhost:8000](http://localhost:8000/);

        proxy\_set\_header Host $host;

        proxy\_set\_header X-Real-IP $remote\_addr;

**}**

    location /static **{**

        alias  /home/www/flask\_project/static/;

**}**

**}**

So, HTTP requests that hit the / endpoint will be ‘[reverse proxied](http://en.wikipedia.org/wiki/Reverse_proxy)’ to port 8000 on 127.0.0.1(or the “loopback ip” or “localhost”). This is same IP and port that gunicorn will use.

We also indicate that we want nginx to directly serve the static files from the “/home/www/flask\_project/static/” directory rather than routing the requests through gunicorn/WSGI. This will speed up our site’s load time since nginx knows to serve that directory directly.

Restart nginx:

$ sudo /etc/init.d/nginx restart

**Profit!**

$ cd /home/www/flask\_project/

$ gunicorn app:app -b localhost:8000

The latter command manually runs gunicorn on localhost port 8000.

Open your browser and navigate to [http://your\_domain\_name\_or\_ip\_address](http://your_domain_name_or_ip_address/).

Again, you should see the “Flask is running!” message. Test out the other URL, /data, as well. If you navigate to <http://your_domain_name_or_ip_address/static>, you should see “Test!”, indicating that we’re serving static files correctly.

**Part 2 – Supervisor**

So, we have a working Flask app; however, there’s one problem: We have to manually (re)start gunicorn each time we make changes to our app. We can automate this with [Supervisor](http://supervisord.org/).

**Configure Supervisor**

SSH into your server, and then install Supervisor:

$ sudo apt-get install -y supervisor

Now create a configuration file:

$ sudo vim /etc/supervisor/conf.d/flask\_project.conf

Add:

**[**program:flask\_project**]**

command **=** gunicorn app:app -b localhost:8000

directory **=** /home/www/flask\_project

user **=** newuser

**Profit!**

Stop gunicorn:

$ sudo pkill gunicorn

Start gunicorn with supervisor:

$ sudo supervisorctl reread

$ sudo supervisorctl update

$ sudo supervisorctl start flask\_project

Make sure your app is still running at [http://your\_domain\_name\_or\_ip\_address](http://your_domain_name_or_ip_address/). Check out the Supervisor [documentation](http://supervisord.org/index.html) for custom configuration info.

source deactivate myenv

zypper addrepo -G -t yum -c 'http://nginx.org/packages/sles/12' nginx

conda create -n myenv --clone root

# To activate this environment, use:

# > source activate myenv

#

# To deactivate this environment, use:

# > source deactivate myenv

conda install --offline -f gunicorn-19.1.0-py36\_0.tar.bz2

conda install --offline -f supervisor-3.3.1-py27\_0.tar.bz2

gunicorn wsgi:app --bind 127.0.0.1:8000

**gunicorn -c config.py wsgi:app -b localhost:8000 –D**

**Current:**

**gunicorn -c config.py \_init\_:app -b localhost:8000 –D**

**gunicorn -c config.py wsgi:\_init\_ -b localhost:8000 –D**

/usr/lib/python2.7/site-packages/supervisor-3.3.3-py2.7.egg/supervisor

**POSTGRESQL :**

/var/lib/pgsql

createuser -s dtnewsletter

createdb newsletter\_db

alter user dtnewsletter with encrypted password 'news\_admin';

grant all privileges on database newsletter\_db to dtnewsletter;

**MARIADB :**

**mysql -u root -p**

SET PASSWORD FOR 'root'@'localhost' = PASSWORD('Theerdhu@90');

create user 'dtnewsletter' IDENTIFIED BY 'news\_admin';

create user 'dtwave' IDENTIFIED BY 'dtwave';

create user 'validationtool' IDENTIFIED BY 'validationtool';

**CREATE** **DATABASE** dtwave;

**CREATE** **DATABASE** newsletter\_db;

**CREATE** **DATABASE** validation\_db;

GRANT ALL PRIVILEGES ON newsletter\_db.\* to dtnewsletter@localhost;

GRANT ALL PRIVILEGES ON dtwave.\* to dtwave @localhost;

GRANT ALL PRIVILEGES ON validation\_db.\* to validationtool@localhost;

sudo lsof -i -P –n --🡪 To check which port number the application is using.

Theerdhu@90

/var/www/html/phpPgAdmin

server {

location / {

proxy\_pass http://localhost:8000;

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

}

location /static {

alias /home/www/flask\_project/static/;

}

}

ALTER TABLESPACE SYSTEM

ADD TEMPFILE 'tbs\_temp\_01.dat'

SIZE 500M

AUTOEXTEND ON;

select df.tablespace\_name "Tablespace",

totalusedspace "Used MB",

(df.totalspace - tu.totalusedspace) "Free MB",

df.totalspace "Total MB",

round(100 \* ( (df.totalspace - tu.totalusedspace)/ df.totalspace))

"Pct. Free"

from

(select tablespace\_name,

round(sum(bytes) / 1048576) TotalSpace

from dba\_data\_files

group by tablespace\_name) df,

(select round(sum(bytes)/(1024\*1024)) totalusedspace, tablespace\_name

from dba\_segments

group by tablespace\_name) tu

where df.tablespace\_name = tu.tablespace\_name ;

ALTER TABLESPACE SYSTEM

ADD

DATAFILE TEMP SIZE 138240K

REUSE AUTOEXTEND

ON NEXT 10240K MAXSIZE 32767M;

ALTER DATABASE DATAFILE '/u01/app/oracle/oradata/XE/system.dbf'

RESIZE 1500M;

nimmagth@HYRDNASDT01:/etc/init.d> /etc/init.d/oracle-xe stop

redirecting to systemctl stop oracle-xe.service

==== AUTHENTICATING FOR org.freedesktop.systemd1.manage-units ===

Authentication is required to stop 'oracle-xe.service'.

Authenticating as: root

Password:

polkit-agent-helper-1: pam\_authenticate failed: Authentication failure

==== AUTHENTICATION FAILED ===

Failed to stop oracle-xe.service: Access denied

nimmagth@HYRDNASDT01:/etc/init.d> sudo /etc/init.d/oracle-xe stop

redirecting to systemctl stop oracle-xe.service

nimmagth@HYRDNASDT01:/etc/init.d>

nimmagth@HYRDNASDT01:/etc/init.d>

nimmagth@HYRDNASDT01:/etc/init.d> sudo /etc/init.d/oracle-xe start

redirecting to systemctl start oracle-xe.service

export LD\_LIBRARY\_PATH=/usr/lib/oracle/12.1/client64/lib:$LD\_LIBRARY\_PATH

export ORACLE\_HOME=/u01/app/oracle/product/11.2.0/xe/

export PATH=/u01/app/oracle/product/11.2.0/xe/bin:$PATH

export ORACLE\_SID=xe

lsnrctl start

Listner & tns files in the below path

/u01/app/oracle/product/11.2.0/xe/network/admin

sqlplus sys/sagar1990@xe as sysdba

[nimmagth@HYRDNASDT01:/etc/init.d](mailto:nimmagth@HYRDNASDT01:/etc/init.d)>

configure proxy path

export HTTPS\_PROXY=http://nimmagat:Theerdhu90@usproxy.es.oneadp.com:8080/

export HTTP\_PROXY=http://nimmagat:Theerdhu90@usproxy.es.oneadp.com:8080/

export HTTPS\_PROXY=http://nimmagat:ILOvemydad90@usproxy.es.oneadp.com:8080/

export HTTP\_PROXY=http://nimmagat:ILOvemydad90@usproxy.es.oneadp.com:8080/

export HTTPS\_PROXY=http://nimmagth:Ilovemydad90@11.0.6.39:8080/

export HTTP\_PROXY=http://nimmagth: Ilovemydad90@11.0.6.39:8080/

export HTTPS\_PROXY=http://nimmagth:ILOvemydad90@11.0.6.39:8080/

export HTTP\_PROXY=http://nimmagth:ILOvemydad90@11.0.6.39:8080/

SET HTTPS\_PROXY=http://nimmagth:ILOvemydad90@11.0.6.39:8080/

SET HTTP\_PROXY=http://nimmagth:ILOvemydad90@11.0.6.39:8080/

SET HTTPS\_PROXY=https://geo-cluster150629-swg.ibosscloud.com/

SET HTTP\_PROXY= https://geo-cluster150629-swg.ibosscloud.com/

export HTTPS\_PROXY=http://nimmagth:ILOvemydad90@11.164.4.26:8080/

export HTTP\_PROXY=http://nimmagth:ILOvemydad90@11.164.4.26:8080/

export HTTPS\_PROXY=http://nimmagth:Ilovemydad90@hydproxy.es.oneadp.com/

adp-proxy

This module parses the .pac files for ES and ESI which define the web proxies for urls and returns the result. This is convenient because regardeless of your location, ES or ESI (TODO: does Azia have their own .pac? If so, add) you always know the address of the proxy from a specific machine (workstation, pod or server) to a specific url.

You can embed this in code, or use it in a ci/cd pipeline.

How to use

module is in the ESI artifactory):

<https://repository.esi.adp.com/api/npm/npm>

cli

Either use npx (if you have the above registry set as default):

npx @marketplace-esi/adp-proxy https://amazon.com

Or install globally and run:

npm i -g @marketplace-esi/adp-proxy --registry https://repository.esi.adp.com/api/npm/npm

adp-proxy https://amazon.com

sudo -E pip install virtualenv

python3 -m venv env

$ wget ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/pcre-8.43.tar.gz

$ tar -zxf pcre-8.43.tar.gz

$ cd pcre-8.43

$ ./configure

$ make

$ sudo make install

$ wget http://zlib.net/zlib-1.2.11.tar.gz

$ tar -zxf zlib-1.2.11.tar.gz

$ cd zlib-1.2.11

$ ./configure

$ make

$ sudo make install

$ wget http://www.openssl.org/source/openssl-1.1.1c.tar.gz

$ tar -zxf openssl-1.1.1c.tar.gz

$ cd openssl-1.1.1c

$ ./Configure darwin64-x86\_64-cc --prefix=/usr

$ make

$ sudo make install

$ ./Configure LIST | grep -i linux

$ ./Configure linux-x86\_64 --prefix=/usr

zypper in libopenssl-devel

zypper install httpd

zypper install gcc

sudo zypper install gcc-c++

sudo zypper install python3-devel (for cx\_oracle)

./configure --prefix=/etc/nginx --sbin-path=/usr/local/nginx/nginx --conf-path=/etc/nginx/nginx.conf --pid-path=/usr/local/nginx/nginx.pid --with-pcre=../pcre-8.43 --with-zlib=../zlib-1.2.11 --with-http\_ssl\_module --with-stream --with-mail=dynamic

while configurating installation path -

./configure --prefix=/etc/nginx --sbin-path=/usr/sbin/nginx --error-log-path=/var/log/nginx/error.log --pid-path=/var/run/nginx.pid --lock-path=/var/run/nginx.lock --user=nginx --group=nginx

----------------------------------------------------

vim /usr/lib/systemd/system/nginx.service

---------------------------------------

[Unit]

Description=The NGINX HTTP and reverse proxy server

After=syslog.target network.target remote-fs.target nss-lookup.target

[Service]

Type=forking

PIDFile=/var/run/nginx.pid

ExecStartPre=/usr/local/nginx/nginx -t -c /etc/nginx/nginx.conf

ExecStart=/usr/local/nginx/nginx -c /etc/nginx/nginx.conf

ExecReload=/bin/kill -s HUP $MAINPID

ExecStop=/bin/kill -s TERM $MAINPID

[Install]

WantedBy=multi-user.target

--------------------------------------------------------------

sudo systemctl enable nginx.service

Created symlink from /etc/systemd/system/multi-user.target.wants/nginx.service to /usr/lib/systemd/system/nginx.service.

$ sudo systemctl start nginx ## <-- start the service ##  
$ sudo systemctl restart nginx ## <-- restart the service ##  
$ sudo systemctl stop nginx ## <-- stop the service ##  
$ sudo systemctl status nginx ## <-- Get the status of the service ##

------------------------------------------------------------------------------------------------------

oracle client - <https://oracle.github.io/odpi/doc/installation.html#linux>