# Deploy EpiML on Ubuntu base on apache2

Here is a quick tutorial to deploy a Flask application on Ubuntu 16.04LTS using Apache2 and mod-wsgi.

## Requirements

1. A server running Ubuntu 16.04LTS.
2. A non-root user account with sudo privilege set up on your server.

## Install Packages:

We can install depended packages easily by using Ubuntu’s package manager, apt.

1. Apache2

The Apache HTTP Server Project is an effort to develop and maintain an open-source HTTP server for modern operating systems including UNIX and Windows. The goal of this project is to provide a secure, efficient and extensible server that provides HTTP services in sync with the current HTTP standards. Install apache2 using following command:

sudo apt-get update

sudo apt-get install apache2

1. mod-wsgi for python3

The Web Server Gateway Interface (WSGI) is a specification for simple and universal interface between web servers and web applications or frameworks for the Python programming language. Mod\_wsgi is an Apache HTTP Server module that provides a WSGI compliant interface for hosting Python based web applications under Apache. It enables Apache to serve Flask applications.

sudo apt-get install libapache2-mod-wsgi-py3

After finished, the module wsgi should be enabled. If not, you can enable wsgi by using the following command:

sudo a2enmod wsgi

1. Git

Git is a fast, scalable, distributed revision control system with an unusually rich command set that provides both high-level operations and full access to internals.

sudo apt-get install git

1. Manage the permission of /var/www

* Type the following command to add an existing user called user to the www-data group, enter:

sudo usermod -a -G www-data $USER

* Change owner and group

sudo chown -R www-data:www-data /var/www

* Add *s* attribute which will keep new files and directories within cake having the same group permissions

sudo chmod -R g+w /var/www

sudo chmod -R g+s /var/www

1. Change the work directory, and clone our website from github.

cd /var/www/

git clone https://github.com/shilab/EpiML

1. R

R is a free software environment for statistical computing and graphics. Install r-base and global packages by using following commands:

sudo apt-get install r-base

If the R version is <3.3, you need upgrade R following

<https://www.digitalocean.com/community/tutorials/how-to-install-r-on-ubuntu-16-04-2>

Install global R packages: glmnet, EBEN, jsonlite

sudo su - -c "R -e \"install.packages('glmnet', repos = 'http://cran.rstudio.com/')\""

sudo su - -c "R -e \"install.packages('EBEN', repos = 'http://cran.rstudio.com/')\""

sudo su - -c "R -e \"install.packages('jsonlite', repos = 'http://cran.rstudio.com/')\""

sudo su - -c "R -e \"install.packages('sets', repos = 'http://cran.rstudio.com/')\""

Download BhGLM from from <https://github.com/nyiuab/BhGLM> and install it using following command (change the path)

sudo su - -c "R -e \"install.packages('/absolute/path/to/BhGLM\_1.1.0.tar.gz', repos = NULL, type='source')\""

1. Python3

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python’s elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms. Python3 should be install in Ubuntu 16.04 LTS. But we still need to install the python package manager：

sudo apt-get install python3-setuptools python3-pip

1. Virtualenv

virtual environment is a self-contained directory tree that contains a Python installation for a particular version of Python, plus a number of additional packages.

sudo pip3 install virtualenv

create a virtual environment for website

cd /var/www/EpiML

virtualenv venv

source venv/bin/activate

1. Flask 0.12 and plugins, Scientific tools

Install flask by using following command in system-wide:

pip install -r requirements.txt

1. Configure Celery daemons

Celery is a simple, flexible, and reliable distributed system to process vast amounts of messages, while providing operations with the tools required to maintain such a system.

Creat file /etc/default/celery:

# Names of nodes to start

# most people will only start one node:

CELERYD\_NODES="worker1 worker2"

# but you can also start multiple and configure settings

# for each in CELERYD\_OPTS

#CELERYD\_NODES="worker1 worker2 worker3"

# alternatively, you can specify the number of nodes to start:

#CELERYD\_NODES=10

# Absolute or relative path to the 'celery' command:

#CELERY\_BIN="/usr/local/bin/celery"

CELERY\_BIN="/var/www/EpiML/venv/bin/celery"

# App instance to use

# comment out this line if you don't use an app

CELERY\_APP="EpiML.celery"

# or fully qualified:

#CELERY\_APP="proj.tasks:app"

# Where to chdir at start.

#CELERYD\_CHDIR="/var/www/EpiML/"

# How to call manage.py

CELERYD\_MULTI="multi"

# Extra command-line arguments to the worker

#CELERYD\_OPTS="--time-limit=300 --concurrency=8"

# Configure node-specific settings by appending node name to arguments:

#CELERYD\_OPTS="--time-limit=300 -c 8 -c:worker2 4 -c:worker3 2 -Ofair:worker1"

# Set logging level to DEBUG

CELERYD\_LOG\_LEVEL="INFO"

# %n will be replaced with the first part of the nodename.

CELERYD\_LOG\_FILE="/var/log/celery/%n%I.log"

CELERYD\_PID\_FILE="/var/run/celery/%n.pid"

# Workers should run as an unprivileged user.

# You need to create this user manually (or you can choose

# a user/group combination that already exists (e.g., nobody).

CELERYD\_USER="celery"

CELERYD\_GROUP="celery"

# If enabled pid and log directories will be created if missing,

# and owned by the userid/group configured.

CELERY\_CREATE\_DIRS=1

Creat file /etc/systemd/system/celery.service:

[Unit]

Description=Celery Service

After=network.target

[Service]

Type=forking

User=celery

Group=celery

EnvironmentFile=/etc/default/celery

WorkingDirectory=/var/www/EpiML

ExecStart=/bin/sh -c '${CELERY\_BIN} multi start ${CELERYD\_NODES} \

-A ${CELERY\_APP} --pidfile=${CELERYD\_PID\_FILE} \

--logfile=${CELERYD\_LOG\_FILE} --loglevel=${CELERYD\_LOG\_LEVEL} ${CELERYD\_OPTS}'

ExecStop=/bin/sh -c '${CELERY\_BIN} multi stopwait ${CELERYD\_NODES} \

--pidfile=${CELERYD\_PID\_FILE}'

ExecReload=/bin/sh -c '${CELERY\_BIN} multi restart ${CELERYD\_NODES} \

-A ${CELERY\_APP} --pidfile=${CELERYD\_PID\_FILE} \

--logfile=${CELERYD\_LOG\_FILE} --loglevel=${CELERYD\_LOG\_LEVEL} ${CELERYD\_OPTS}'

[Install]

WantedBy=multi-user.target

Once you’ve put that file in /etc/systemd/system, you should run systemctl daemon-reload in order that Systemd acknowledges that file. You should also run that command each time you modify it.

To configure user, group, chdir change settings: User, Group, and WorkingDirectory defined in /etc/systemd/system/celery.service.

sudo adduser --system --group --no-create-home celery

sudo mkdir /var/log/celery

sudo mkdir /var/run/celery

sudo chown -R celery:celery /var/log/celery

sudo chown -R celery:celery /var/run/celery

start and enable start at boot.

sudo systemctl start celery

For more information: Following Usage systemd section in below to set up celery daemon: <http://docs.celeryproject.org/en/latest/userguide/daemonizing.html>

1. Redis

Redis is an open source (BSD licensed), in-memory data structure store, used as a database, cache and message broker. Following document in below to install redis

<https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-redis-on-ubuntu-16-04>

start and enable start at boot.

sudo systemctl start redis

sudo systemctl enable redis

Run the following command to test if the installation is successful:

python run.py

You should see the following output:

\* Serving Flask app "EpiML" (lazy loading)

\* Environment: production

WARNING: Do not use the development server in a production environment.

Use a production WSGI server instead.

\* Debug mode: on

\* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)

\* Restarting with stat

\* Debugger is active!

\* Debugger PIN: 734-373-964 (PIN should be different)

You can open a browser and input the address: http://127.0.0.1:5000/. If you see the web server, you have successfully run the website.

deactivate

## Configure New Virtualhost for Flask

Now, you will need to create a new wsgi and a new virtual host file for Flask App.

1. Create a wsgi file inside the /var/www/EpiML directory to serve the Flask App:

vim EpiML.wsgi

Add the following content (pay attention on site dir):

#!/usr/bin/python3

import sys

import logging

import site

# Add the site-packages of the chosen virtualenv to work with

site.addsitedir('/var/www/EpiML/venv/lib/python3.5/site-packages')

logging.basicConfig(stream=sys.stderr)

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

from EpiML import app as application

1. Now, you will create a new virtual host file for the Flask app so that it can run in apache2.

Create the configure file for Flask App:

sudo vim /etc/apache2/sites-available/EpiML.conf

Add the following content:

<VirtualHost \*:80>

ServerName [www.example.com](http://www.example.com)

ServerAdmin [admin@example.com](mailto:admin@example.com)

WSGIScriptAlias /ShiLab/EpiML /var/www/EpiML/EpiML.wsgi

<Directory /var/www/EpiML/EpiML>

Order allow,deny

Allow from all

</Directory>

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

LogLevel warn

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

</VirtualHost>

Note: You need change ServerName and ServerAdmin depending on yours

You need to create /var/log/apache2/ EpiML directory to store the log file.

sudo mkdir /var/log/apache2/ShiLab/

sudo mkdir /var/log/apache2/ShiLab/EpiML

This is for testing in local.

Editing your /etc/hosts file to review the domain example.com before you public it.

sudo vim /etc/hosts

Add a line in last:

127.0.0.1 www.example.com

Enable the virtual host with the following command. By the way, you also can disable the configure with *a2dissite*:

sudo a2ensite EpiML.conf

Restart Apache to apply the changes

service apache2 reload

sudo apachectl restart

All configure should be setup. You can launch the web server by inputing web address in browser: <http://www.example.com/ShiLab/EpiML/>

Note:

You maybe get a friendly warning: AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.1.1. Set the 'ServerName' directive globally to suppress this message.

But don’t worry. The application has been enabled.

## Solving the friendly warning:

If you want to solve this waring, please following next.

Apache no longer uses the conf.d directory. All the configuration files are stored inside /etc/apache2/conf-available directory and all the configuration files should now have a .conf extension.

In order to solve this message in Apache 2.4, we have to create a configuration file inside the conf-available directory. For example servername.conf.

sudo vi /etc/apache2/conf-available/servername.conf

And inside this we just need to add one line

ServerName localhost

If you want to use a domain name or any other name depending on the requirement its fine, just replace localhost with whatever you need. Next you have to enable this configuration. For this, you need to run the following command:

sudo a2enconf servername

a2enconf is a command to enable a configuration file in Apaches 2.4. Also note that servername on the above command is from the name of the configuration file servername.conf.

After this reload the server and the above message will no longer bug you.

sudo service apache2 reload

Congratulations!

For update Web Site:

1. pull from github.

git stash

git pull https://github.com/shilab/EpiML

1. And restart apache2

sudo apachectl restart