AWS EC2 Configuration Guide for Game Recommendation Flask App Configuration

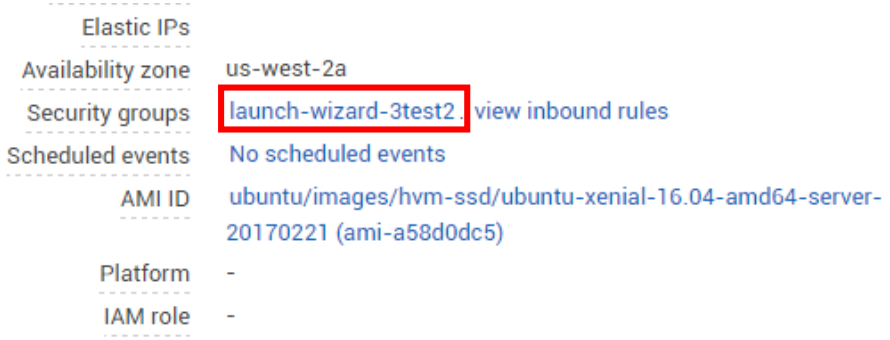
The purpose of this guide is to install and configure Flask on EC2. We will use flask as micro web development framework for Python to demo your Game Recommendation Result.

Flask is small and easy to setup. Please check here for its documentation.

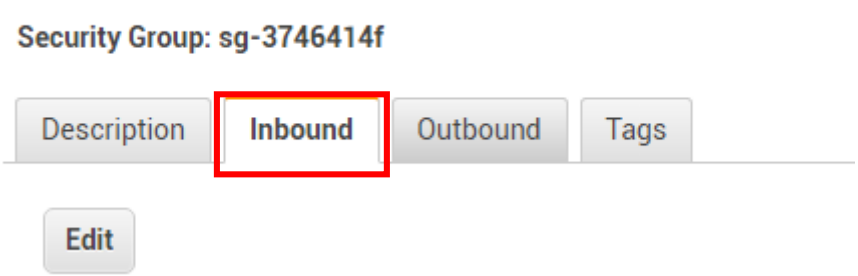
After you set up EC2 and configured Jupyter notebook on your instance. You can follow this instruction to set up and configure Flask on your EC2.

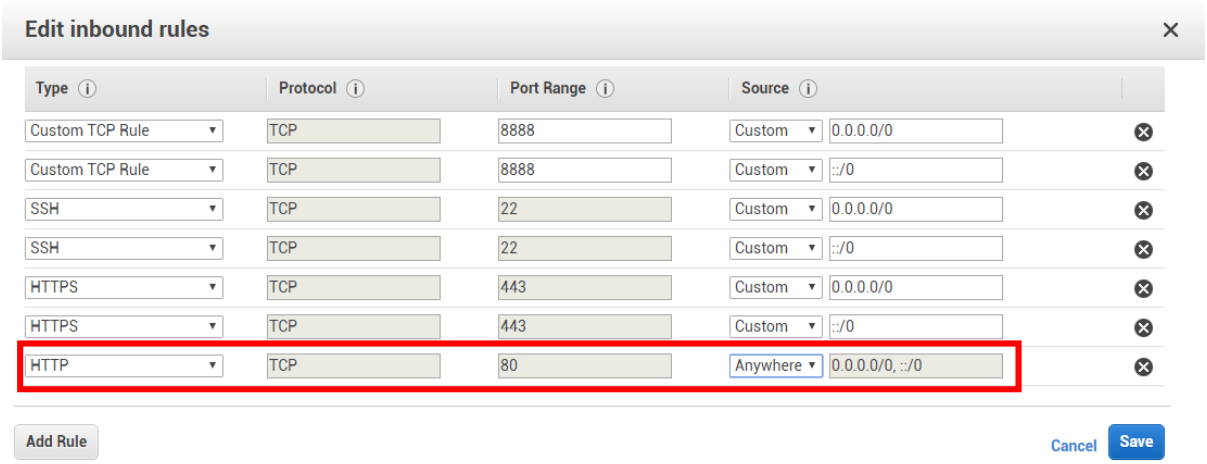
1. Launch an EC2 instance and add rules in security group.

Make sure your instance is running and click your security group name.



Select “Inbound”, then press “Edit” button





Select “Add Rule” and add HTTP protocol, remember to change access source to “Anywhere”.

Press the save button to save the rules.

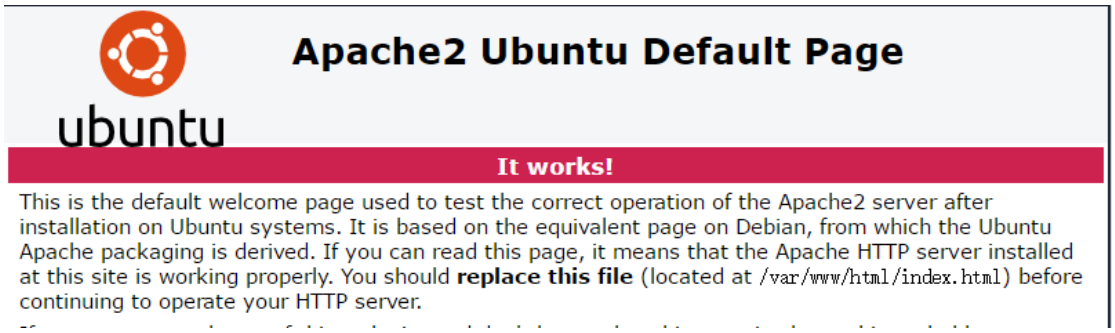
2. Setting up the instance

Now let’s use ssh to connect to your instance and install Flask

Install the apache webserver and mod_wsgi.

```
sudo apt-get update
sudo apt-get install apache2
sudo apt-get install libapache2-mod-wsgi
```

Then pointing your browser to your instance’s public DNS name. You should see the following page:



Install Flask using the pip tool

```
sudo apt-get install python-pip
sudo pip install flask
```

Create a directory for our Flask app.

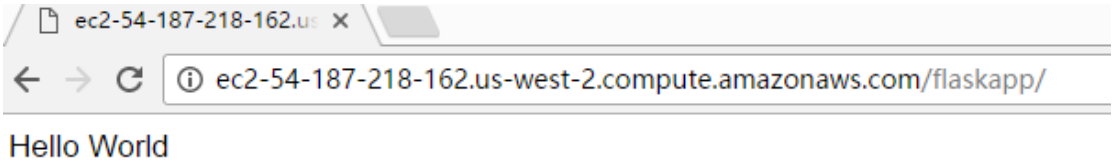
We'll create a directory in our home directory to work in, and link to it from the site-root defined in apache's configuration (`/var/www/html` by defualt, see `/etc/apache2/sites-enabled/000-default.conf` for the current value).

```
mkdir ~/flaskapp

sudo ln -sT ~/flaskapp /var/www/html/flaskapp
```

To verify our operation is working, create a simple `index.html` file.

```
cd ~/flaskapp
echo "Hello World" > index.html
```

You could now see the strings "Hello World" displayed if you navigate to `(your instance public DNS)/flaskapp` in your browser.



3. Running a simple Flask app

Flask structure:

Assuming you have seen the Testing Flask Applications section and have either written your own tests for `flaskr` or have followed along with the examples provided, you might be wondering about ways to organize the project.

One possible and recommended project structure is:

```
flaskr/
    flaskr/
        __init__.py
        static/
        templates/
    tests/
        test_flaskr.py
    setup.py
    MANIFEST.in
```

Create an app.

We'll use the simple "Hello world" example from the Flask documentation. Put the following content in a file named flaskapp.py

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
  return 'Hello from Flask!'

if __name__ == '__main__':
  app.run()
```

Create a .wsgi file to load the app

Put the following content in a file named `flaskapp.wsgi:`

```
import sys
sys.path.insert(0, '/var/www/html/flaskapp')

from flaskapp import app as application
```

Enable mod_wsgi 💬

The apache server displays html pages by default. But to serve dynamic content from a Flask app we'll have to make a few changes.

`sudo vim /etc/apache2/sites-enabled/000-default.conf`

Add the following block **just after** the DocumentRoot `/var/www/html` line:

```
WSGIDaemonProcess flaskapp threads=5
WSGIScriptAlias / /var/www/html/flaskapp/flaskapp.wsgi

<Directory flaskapp>
        WSGIProcessGroup flaskapp
        WSGIApplicationGroup %{GLOBAL}
        Order deny,allow
        Allow from all
</Directory>
```

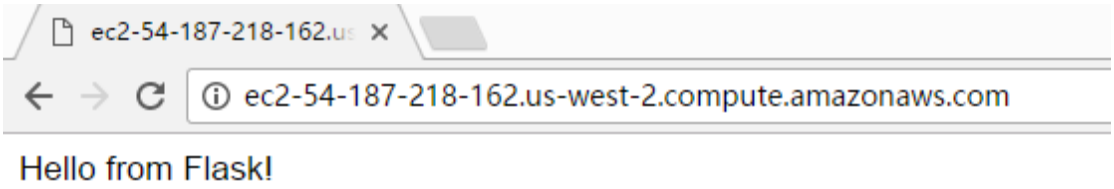
```
File Edit Options Buffers Tools Conf Help
<VirtualHost *:80>
        # The ServerName directive sets the request scheme, hostname and port that
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerName
        # specifies what hostname must appear in the request's Host: header to
        # match this virtual host. For the default virtual host (this file) this
        # value is not decisive as it is used as a last resort host regardless.
        # However, you must set it for any further virtual host explicitly.
        #ServerName www.example.com

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html

        WSGIDaemonProcess flaskapp threads=5
        WSGIScriptAlias / /var/www/html/flaskapp/flaskapp.wsgi

        <Directory flaskapp>
            WSGIProcessGroup flaskapp
            WSGIApplicationGroup %{GLOBAL}
            Order deny,allow
            Allow from all
        </Directory>

        # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
        # error, crit, alert, emerg.
        # It is also possible to configure the loglevel for particular
        # modules, e.g.
        #LogLevel info ssl:warn

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined
-UU-:----F1  000-default.conf   Top L1     (Conf[Space]) ----------------------------------------
```

Restart the webserver

```
sudo apachectl restart
```

4. Test configuration

If you navigate your browser to your EC2 instance's public DNS again, you should see the text returned by the hello_world function of our app, "Hello from Flask!"



Flask allows us to route requests to functions based on the url requested. We can also get input from the url to pass into the function. Add the following to flaskapp.py:

```
@app.route('/showinput/<input_str>')
def show_input(input_str):
    return input_str
```

Then restart your server:

```
sudo apachectl restart
```

You can test your flask main python program by setting the url and check returned result.