# Homework #5 Amazon Web Services (AWS) with Python

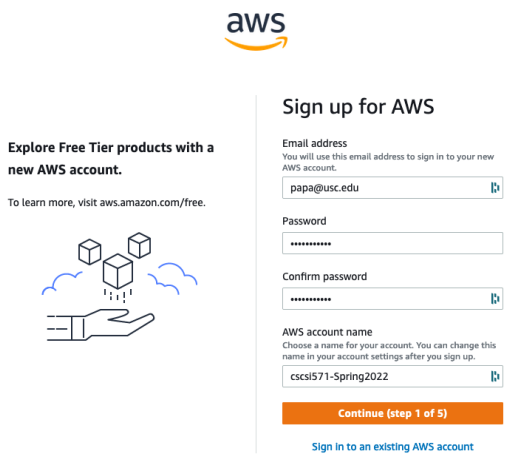
**This semester we are allowing all students to explore cloud computing as offered by Amazon's Web Services. Using the instructions below one can establish a service account at AWS. Unfortunately, the AWS Educate has been terminated (USC belonged to it) and has been replaced by the AWS Academy, Since USC did not join this new program, you will have to sign up for AWS Free Tier to use AWS in any of the assignments. Once established, you will be able to move your Python back-end program developed for Assignment #6 to your AWS instance and have it executed there.**

## 1. Create an AWS Account

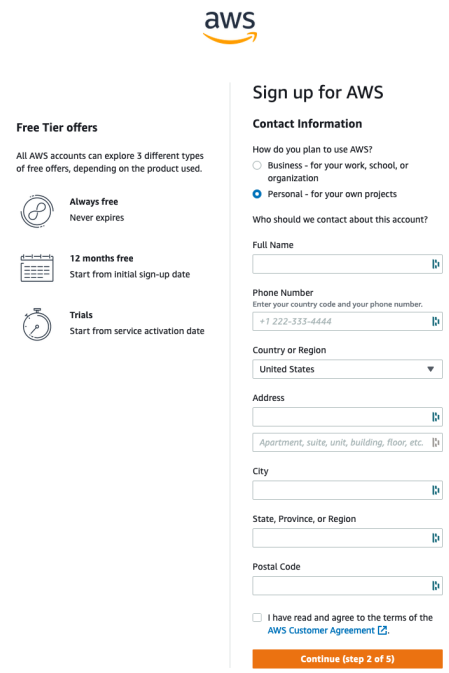
Go to the following URL:

https://aws.amazon.com

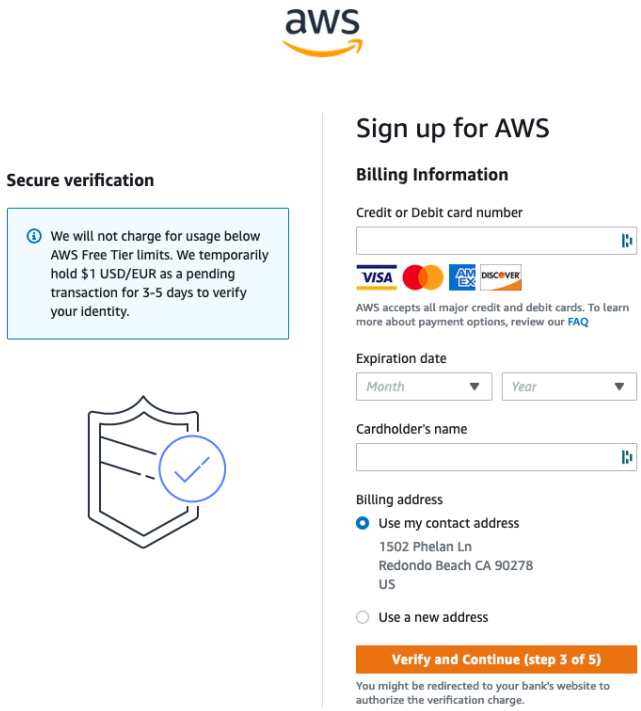
Click on the orange button labeled **Create and AWS Account**.



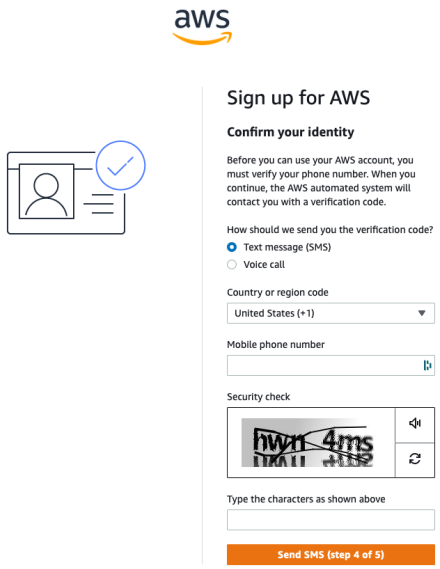

Fill in the requested information under **Sign up for AWS** and click on **Continue**. Note that you will not have to use your USC e-mail account. Any valid e-mail account is OK. A CAPcha security check will be displayed. Enter the requested characters and click **Continue**. The Contact Information form is displayed.




Select **Personal** and enter the rest of the requested information. Also check the Terms checkbox. Click **Continue**. The **Billing Information** for is displayed.

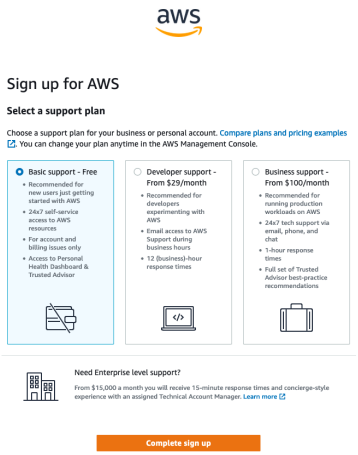



Enter your credit card information and click **Verify and Continue**. Notice that a $1 temporary hold will be made to your credit card for verification purpose. The **Confirm your identity** form is displayed.

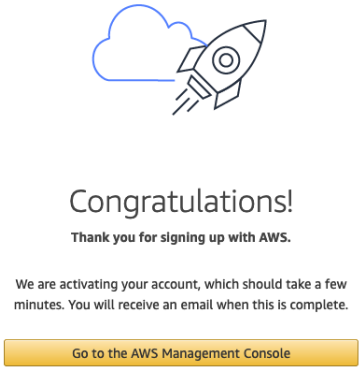



Enter your phone number, the characters shown in the security check, and click **Send SMS**, or **Call me now**. The **Confirm your identity** form is displayed.

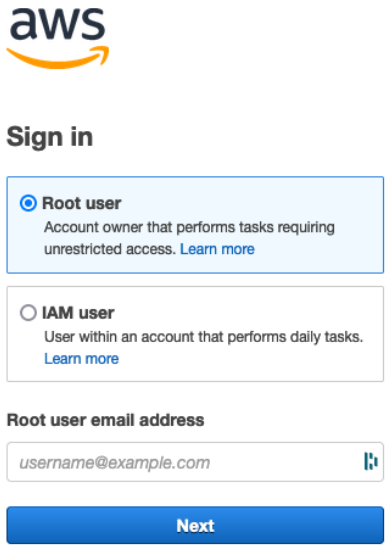

Enter the code you received by SMS or voice call and click **Continue**. The **Select a support plan** form is displayed.



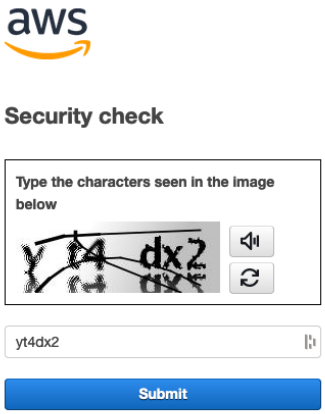
Select **Basic support – free** and click **Complete sign up**. The **Congratulations!** page will be shown.



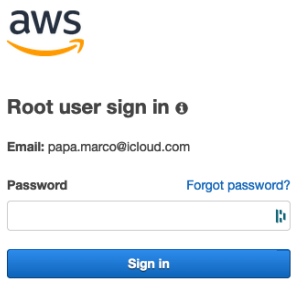

Wait a few minutes for your account to be activated, then click on **Go to the AWS Management Console**. The AWS **Sign in** form is displayed.



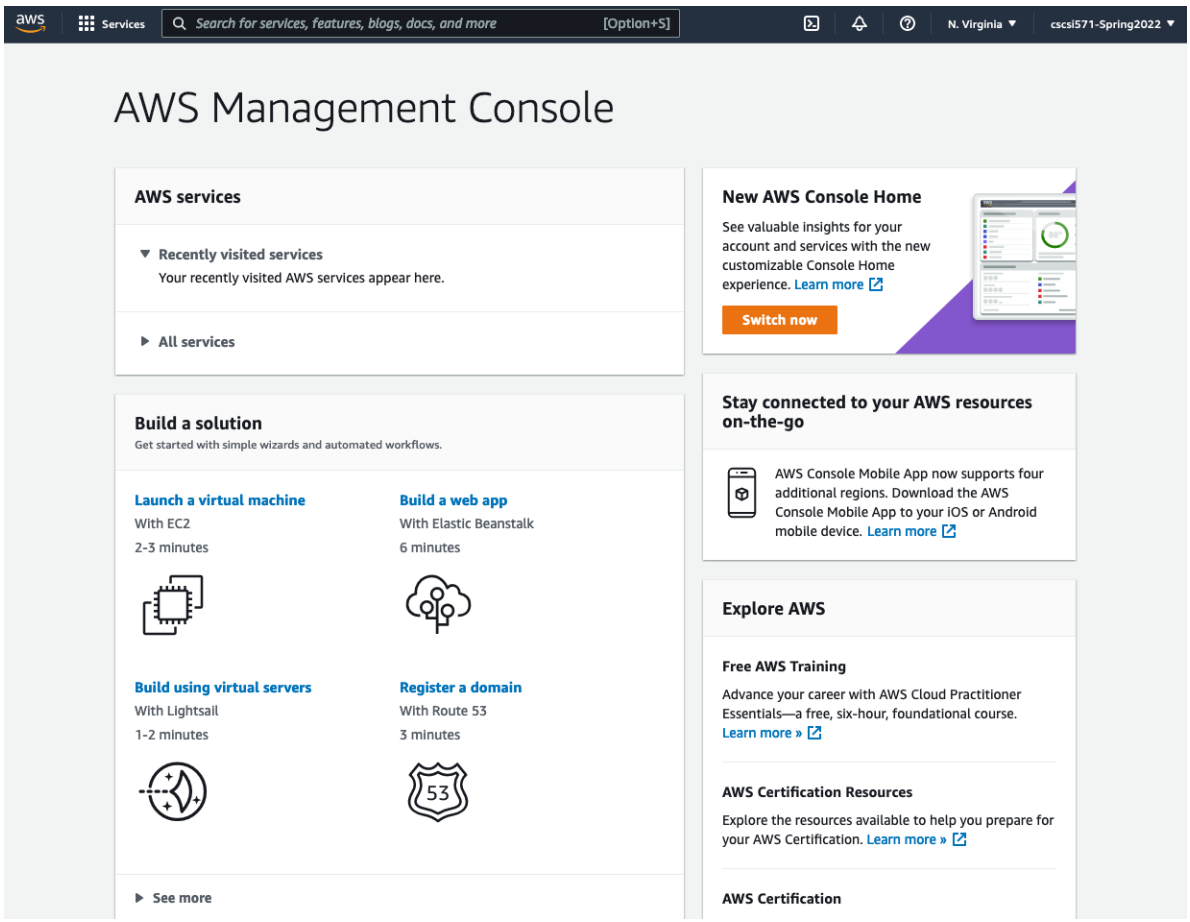

Select **Root user** and enter your e-mail address. Click **Next**. The **Security check** form is displayed.



Enter the characters in the image and click **Submit**. The Root user sign in form is displayed.




Enter your password. Click **Sign in**. The **AWS Management Console** will be displayed.

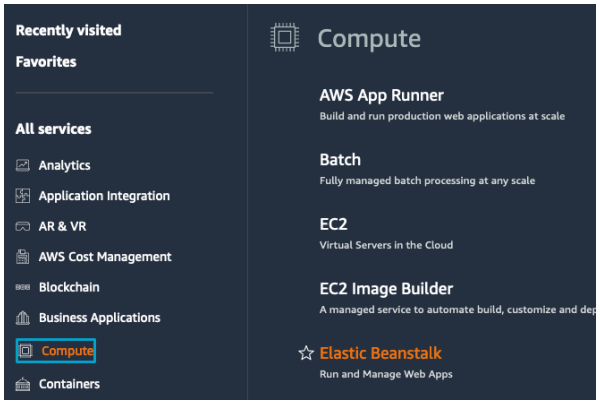



On the top right corner, you can find:

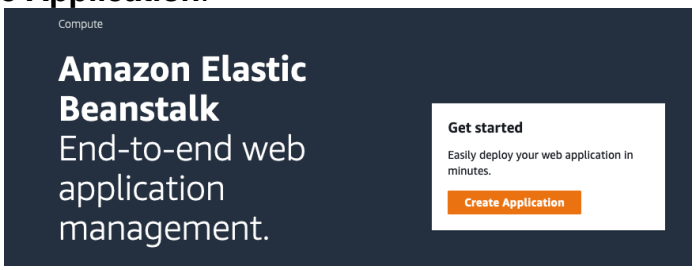
1) Your account name – for example csci571-Spring2022
2) AWS Deployment Region – US East (N.Virginia), or **us-east-1**

## 2. Set up the Default Elastic Beanstalk Application

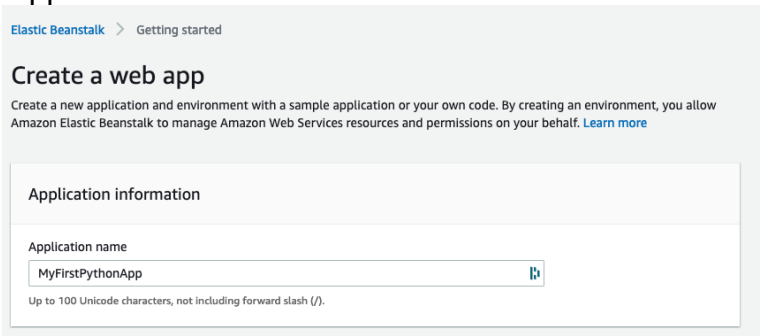
- Click the top left menu named **Services**
- From the list of Amazon Web Services, select **Elastic Beanstalk,** under **Compute**.



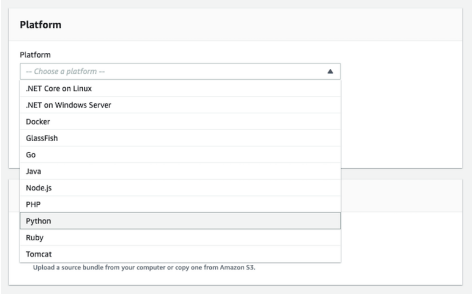

- Select **Create Application**.



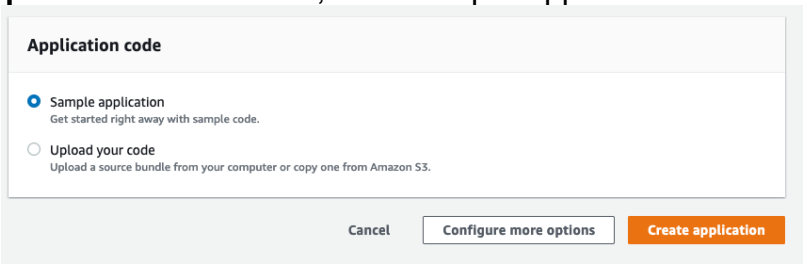

- The **Create a web app** form appears. In the **Application name** field, enter a name for your application.



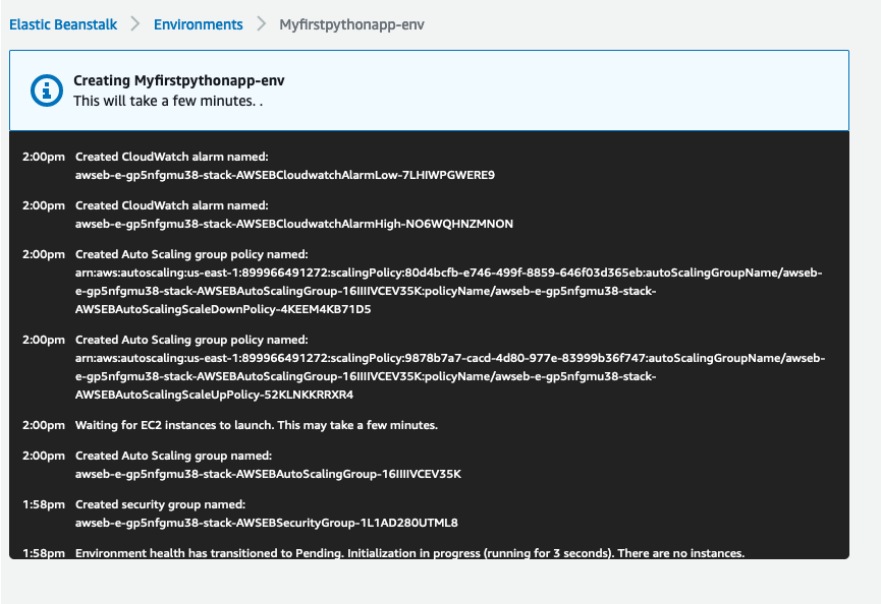

- In the **Platform** drop-down select **Python**.



- In the **Application code** radios, select ample Application.

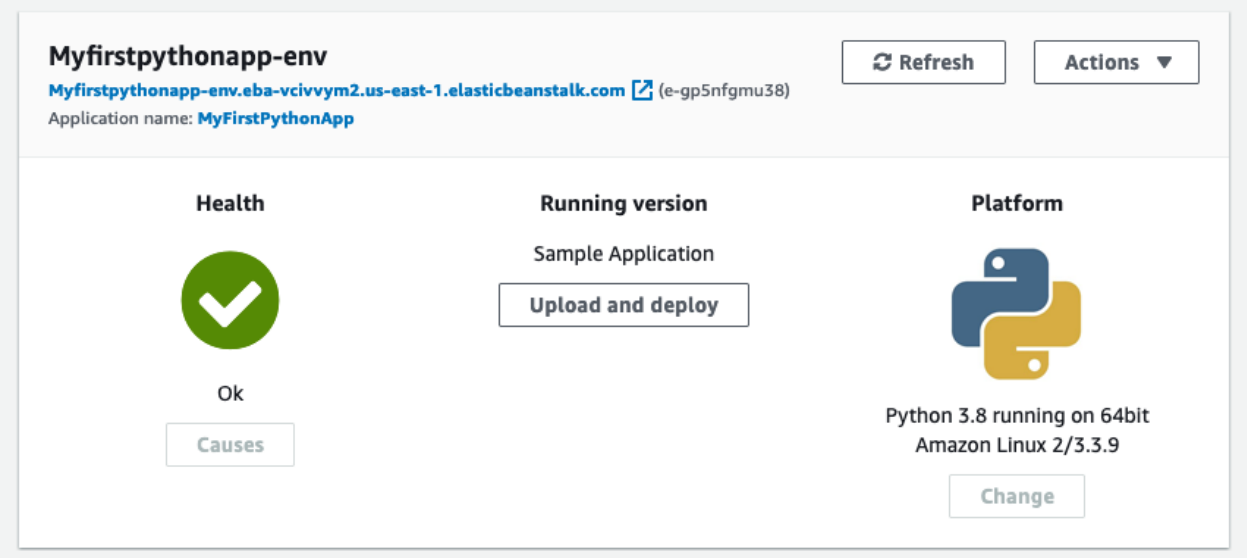



- Click **Create application**.
- Your application will start getting provisioned. After a minute or so the "**Creating <environment-name>**" dialog appears, with the message "This will take a few minutes…"




- After several minutes, at least 5 on the average, your app environment screen will be displayed.

You will need to wait for several minutes as your **Amazon Linux + Python 3.X** instance is created and launched. You will see several messages appear as the instance is being created and deployed. Once creation and launch are completed, you will see green round circle with a check mark in the middle.




**Python Instance Dashboard**

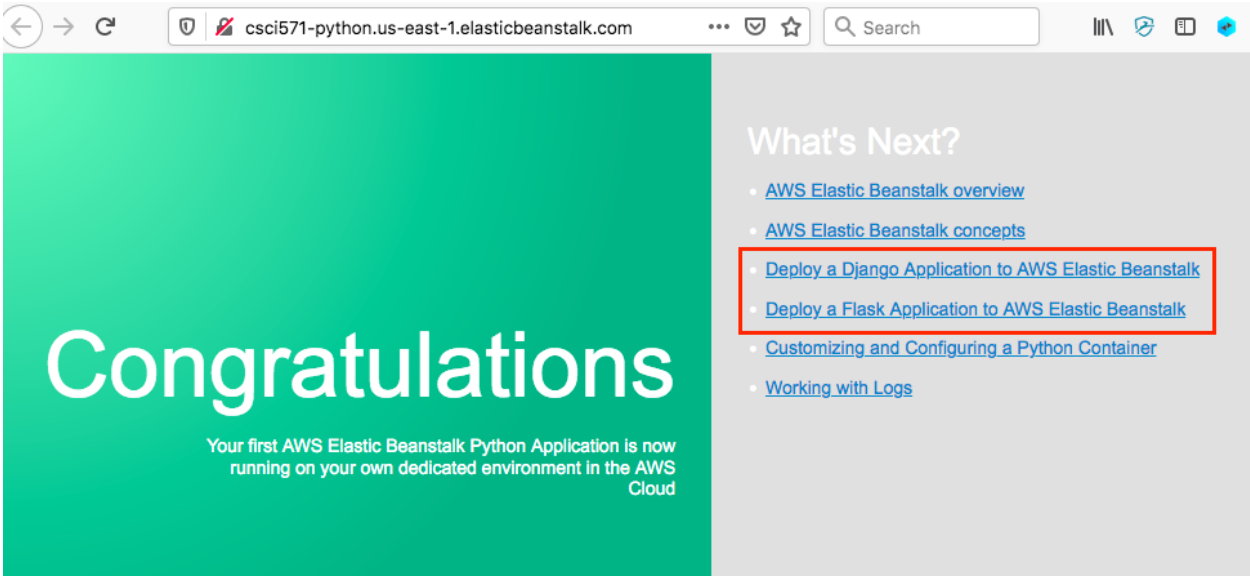
Below the environment name, there is a **URL** such as:

*YourDomainName*.us-east-1.elasticbeanstalk.com

For example:

http://myfirstpythonapp-env.eba-vcivvym2.us-east-1.elasticbeanstalk.com/

**Click** on it. You should see the "*Congratulations*" page. If you see it as shown below, your application and environment have been created properly.




**Python Sample Application**

You have two options listed for deploying web apps in Python on AWS:

- **Flask** web application framework
- **Django** web application framework

We personally recommend that you use Flask, as we believe it is simpler to install and maintain. You are free to use either Python Web Framework, but we will support in Piazza only the Flask deployment.

### 3. Deploy your Python application

### 3.1 Installing Python

On **MacOS** we recommend you use the "brew" package manager to install Python and pip. Flask requires Python 2.7 (which is preloaded on every Mac) or Python 3.4 or newer. We personally recommend **Python 3.8** and **pip3**. In the latter case you should change your shell startup files to point to Python 3.8 instead of Python 2.7.

**Note**: steps for Installing **Python 3.8** can be found in section 2, "*Setting up a Python development environment*", in the file entitled "*Homework #5 Google Cloud Platform (GCP) with Python*", available at:

https://csci571.com/hw/hw5/HW5_Google_Python.pdf

On **Windows 10,** you can install the Windows Subsystem for Linux to get a Windows-integrated version of Ubuntu and Bash.

You can deploy your applications using the AWS Elastic Beanstalk console **Upload and Deploy** or the Elastic Beanstalk Command Line Interface (**EB CLI**).

### 3.2 Deploying a Flask Application to AWS Elastic Beanstalk using "Upload and Deploy"

This is the installation that we recommend, as it uses the Sample Application environment set up in **section 4. Set up the Default Elastic Beanstalk Application**.

**Windows ONLY**: download and install **PowerShell**.

a. Create a project folder:

```
$ mkdir eb-flask

$ cd eb-flask
```

b. Create an isolated Python environment:

```
$ python3 -m venv env
```

```
$ source env/bin/activate
```

(the terminal prompt will add (env) to the terminal prompt)

c. Install flask with pip install:

```
(env) $ pip install flask
```

d. View installed libraries with `pip freeze`:

```
(env) $ pip freeze
click==8.0.1
Flask==2.0.1
itsdangerous==2.0.1
Jinja2==3.0.1
MarkupSafe==2.0.1
Werkzeug==2.0.1
```

e. Create the **`requirement.txt`** file:

```
(env) $ pip freeze > requirements.txt
```

f. Next, create an application that you'll deploy using Elastic Beanstalk **Upload and Deploy**. We'll create a "Hello World" RESTful web service.

g. Next you will create a new text file in this directory named **`application.py`** with the following contents:

```
from flask import Flask

# print a nice greeting.
def say_hello(username = "World"):
    return '<p>Hello %s!</p>\n' % username

# some bits of text for the page.
header_text = '''
    <html>\n<head> <title>EB Flask Test</title> </head>\n<body>'''
instructions = '''
    <p><em>Hint</em>: This is a RESTful web service! Append a username
    to the URL (for example: <code>/Thelonious</code>) to say hello to
    someone specific.</p>\n'''
home_link = '<p><a href="/">Back</a></p>\n'
footer_text = '</body>\n</html>'

# EB looks for an 'application' callable by default.
application = Flask(__name__)

# add a rule for the index page.
application.add_url_rule('/', 'index', (lambda: header_text +
    say_hello() + instructions + footer_text))

# add a rule when the page is accessed with a name appended to the site
# URL.
application.add_url_rule('/<username>', 'hello', (lambda username:
    header_text + say_hello(username) + home_link + footer_text))

# run the app.
if __name__ == "__main__":
    # Setting debug to True enables debug output. This line should be
    # removed before deploying a production app.
    application.debug = True
    application.run()
```

h. To do this, download new sample code (RESTful app) from:

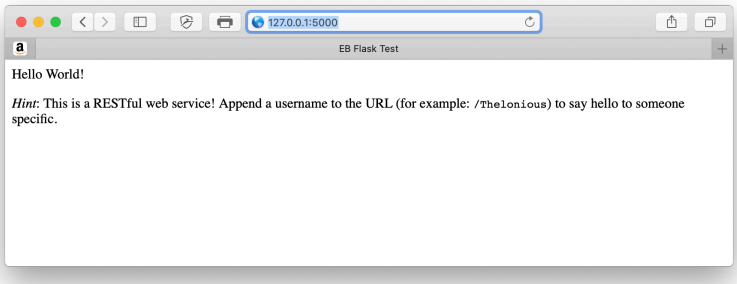
   https://csci571.com/hw/hw5/application.py

   save the file as **application.py** in the same folder as requirements.txt (eb-flask in the example above). Using application.py as the filename and providing a callable application object (the Flask object, in this case) allows Elastic Beanstalk to easily find your application's code.

i. Run application.py locally with Python on port 5000:

```
(env) $ python application.py
 * Serving Flask app "application" (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: 305-600-227
```

j. Test your application locally, by opening `http://127.0.0.1:5000/` in your web browser. You should see the application running, showing the index page:

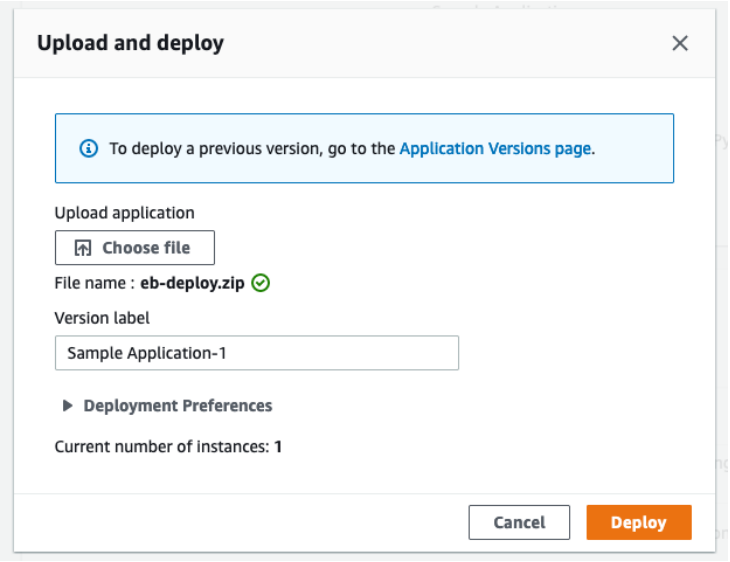



   You can stop the web server and return to your virtual environment by typing **Ctrl+C**.

k. You are ready now to upload and deploy. First of all, “zip” the two needed files, **application.py** and **requirements.txt**:

```
(env) $ zip eb-deploy.zip application.py requirements.txt
  adding: application.py (deflated 48%)
  adding: requirements.txt (deflated 9%)
(env) $
```

l. Now go to the AWS EB console, and click the **Upload and deploy** button:



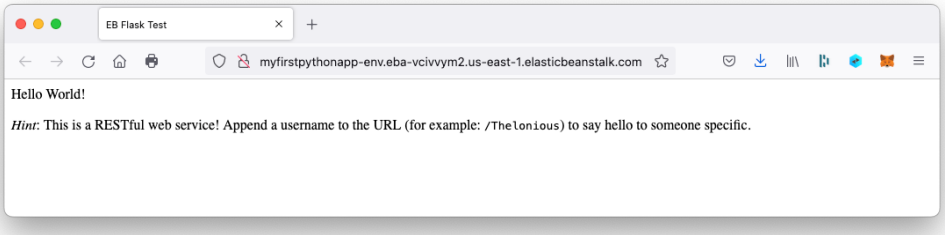

m. Choose the eb-deploy.zip file from your desktop. Enter a unique Version label. Click **Deploy**. The AWS EB server will restart and update your environment.

Recent Events

Show All

| Time | Type | Details |
|---|---|---|
| 2020-01-11 17:15:45 UTC-0800 | INFO | Environment update completed successfully. |
| 2020-01-11 17:15:45 UTC-0800 | INFO | New application version was deployed to running EC2 instances. |
| 2020-01-11 17:14:57 UTC-0800 | INFO | Environment health has transitioned from Ok to Info. Application update in progress (running for 20 seconds). |
| 2020-01-11 17:14:55 UTC-0800 | INFO | Deploying new version to instance(s). |
| 2020-01-11 17:14:15 UTC-0800 | INFO | Environment update is starting. |

n. You are ready now to run the updated AWS “cloud” version of your app.




o. Modify **application.py** for the next exercise, as appropriate. Test locally, and when you have added enough new code, Upload and Deploy and test remotely the cloud version.

### 3.3 Deploying a Flask Application to AWS Elastic Beanstalk using "EB CLI" (Optional)

Click on the corresponding link in the sample application, or follow the tutorial at:

https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/create-deploy-python-flask.html

The Tutorial above includes all of the following:

- Prerequisites
- Flask Framework installation
- Details on installing and configuring the EB CLI
- Set Up a Python Virtual Environment with Flask
- Create a Flask Application
- Run the application locally on your Mac or PC
- Deploy your site with the EB CLI
- Cleanup

Once you have created, deployed and tested the tutorial application, you will have the basic skeleton for a **RESTful web service**.

**Additional Notes**:

The instructions in this Tutorial creates a new Elastic Beanstalk environment and deploys using the EB CLI.

The Tutorial also uses the **us-east-2** region in step 1 of the section titled "To create an environment and deploy your flask application". Since **AWS Educate Starter Accounts** are limited to use only the **us-east-1** region, that step must be changed to use the us-east-1 region as in:

```
$ eb init -p python-3.6 flask-tutorial --region us-east-1
```

Also, you will likely get an error such as "`zlib not available`" during the installation using EB CLI. As mentioned in:

https://github.com/aws/aws-elastic-beanstalk-cli-setup/issues/23

this can be fixed by running:

```
pip install virtualenv
python ./scripts/ebcli_installer.py
```

instead of:

```
brew install awsebcli
```

or installing the EB CLI using Setup Scripts (as in the Tutorial).

### 3.4 Deploying a Django Application to AWS Elastic Beanstalk (Optional)

Click on the corresponding link in the sample application, or follow the tutorial available at:

https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/create-deploy-python-django.html

Follow the steps listed in the tutorial.

## 4. Set up Exploring Your Instance (Optional)

If you want to explore your Instance and create your own domain-based URL with SSH control, you can add the following steps.

### 4.1 Get and Setup SSH

Once the Python app with SSH-enabled environment is running, you can get access using SSH. You can use **ssh** on a Mac running macOS, or **Putty** when running on Windows.

On a Mac, SSH is built into macOS and can be accessed through the **Terminal** app and there is no additional setup needed.

On a Windows PC, you will need to download the complete PuTTY distribution at:

http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

You should download the file **putty.zip** that contains all the binaries, including **PuTTYgen** as see in this snapshot from the website above:

**Alternative binary files**

The installer packages above will provide all of these (except PuTTYtel), but you can download them one by

(Not sure whether you want the 32-bit or the 64-bit version? Read the FAQ entry.)

**putty.exe (the SSH and Telnet client itself)**

| | | | |
|---|---|---|---|
| 32-bit: | putty.exe | (or by FTP) | (signature) |
| 64-bit: | putty.exe | (or by FTP) | (signature) |

**pscp.exe (an SCP client, i.e. command-line secure file copy)**

| | | | |
|---|---|---|---|
| 32-bit: | pscp.exe | (or by FTP) | (signature) |
| 64-bit: | pscp.exe | (or by FTP) | (signature) |

**psftp.exe (an SFTP client, i.e. general file transfer sessions much like FTP)**

| | | | |
|---|---|---|---|
| 32-bit: | psftp.exe | (or by FTP) | (signature) |
| 64-bit: | psftp.exe | (or by FTP) | (signature) |

**puttytel.exe (a Telnet-only client)**

| | | | |
|---|---|---|---|
| 32-bit: | puttytel.exe | (or by FTP) | (signature) |
| 64-bit: | puttytel.exe | (or by FTP) | (signature) |

**plink.exe (a command-line interface to the PuTTY back ends)**

| | | | |
|---|---|---|---|
| 32-bit: | plink.exe | (or by FTP) | (signature) |
| 64-bit: | plink.exe | (or by FTP) | (signature) |

**pageant.exe (an SSH authentication agent for PuTTY, PSCP, PSFTP, and Plink)**

| | | | |
|---|---|---|---|
| 32-bit: | pageant.exe | (or by FTP) | (signature) |
| 64-bit: | pageant.exe | (or by FTP) | (signature) |

**puttygen.exe (a RSA and DSA key generation utility)**

| | | | |
|---|---|---|---|
| 32-bit: | puttygen.exe | (or by FTP) | (signature) |
| 64-bit: | puttygen.exe | (or by FTP) | (signature) |

**putty.zip (a ZIP archive of all the above)**

| | | | |
|---|---|---|---|
| 32-bit: | putty.zip | (or by FTP) | (signature) |
| 64-bit: | putty.zip | (or by FTP) | (signature) |

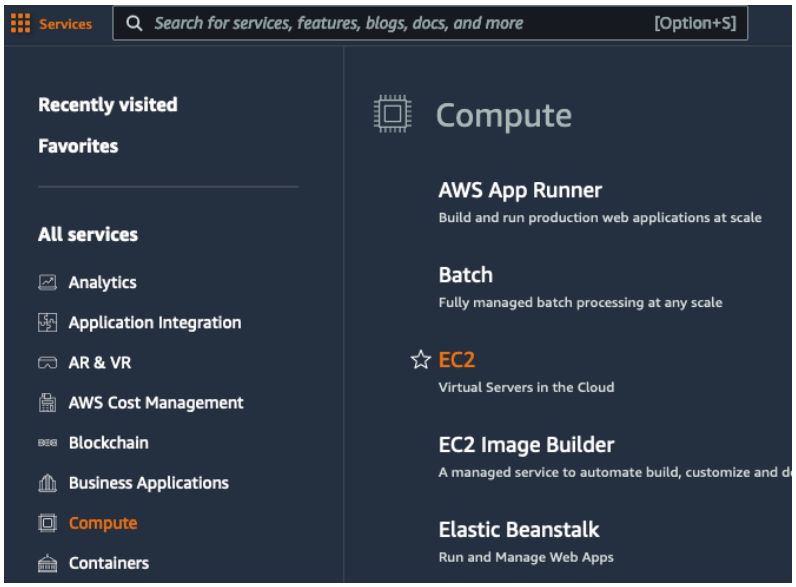
**PuTTY** needs additional setup as it needs to use a converted version of the private key. The instructions on how to perform such conversion are available here:

http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/putty.html

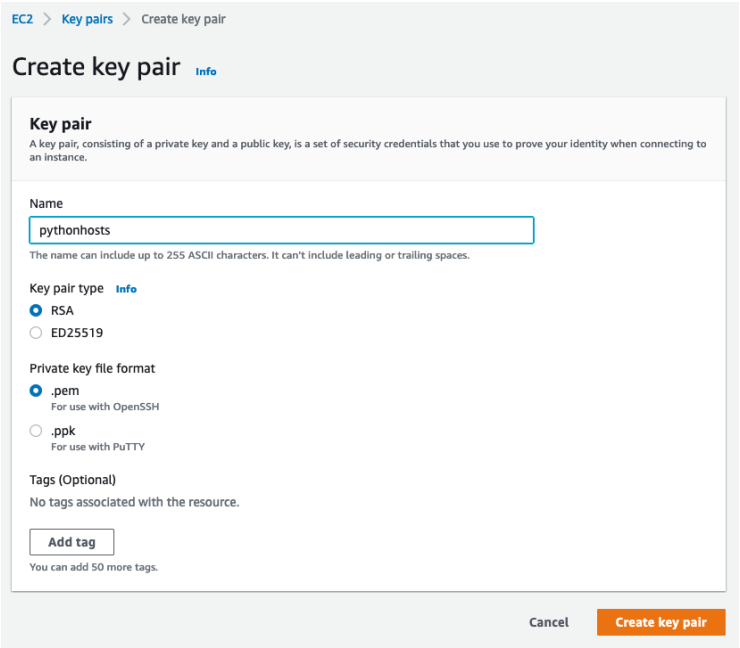
The major step is to use **PuTTYgen** to convert your private key format (.pem) generated by Amazon EC2 into the required PuTTY format (.ppk).

## 4.2 Create a Key Pair

- From the **Services** drop down, under the **Compute** section, select **EC2**.



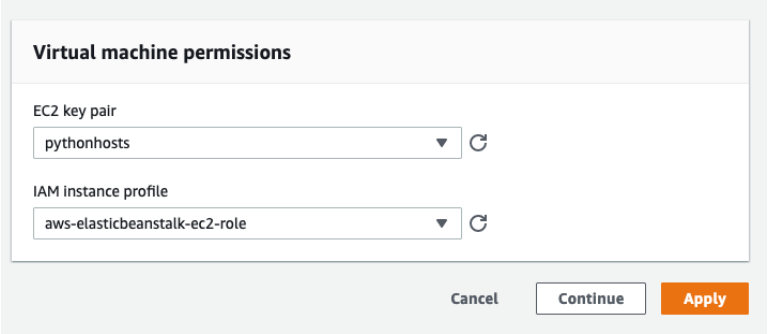

- In the left menu, under **NETWORK AND SECURITY** select **Key Pairs**.
- Click on the button **Create key pair**.
- Enter a name like ***pythonhosts* (you must have your own random name!)**
- For Private key file format, select **.pem** for macOS and Linux, and **.ppk** for Windows and click on **Create key pair**.



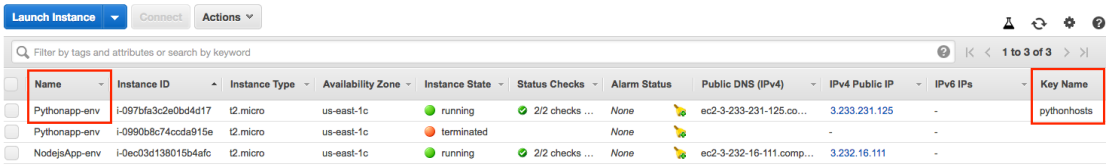

- A download of your private key should start automatically. Save the key, like **pythonhosts.pem**, or **pythonhosts.ppk**, in an appropriate location.

### 4.2.1 Associate your Instance to the Key Pair

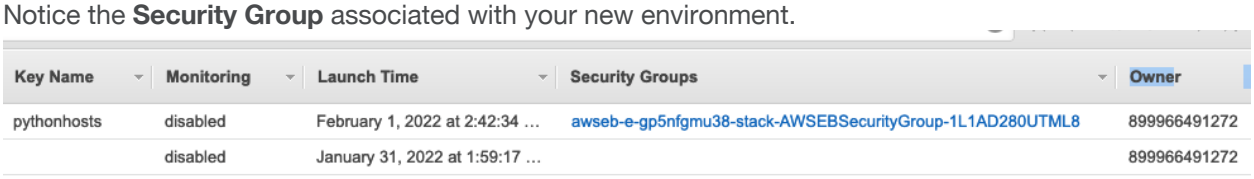
- You now need to associate your Instance with the just created key pair.
- Select the **Elastic Beanstalk** under **Services**.
- Select your previously created environment.
- Select **Configuration** on the left menu.
- Click on the **Edit** button next to **Security**.
- Select the key pair you just created in the **EC2 key pair** drop-down. Click **Apply**.




- Hit **Apply** and then **Confirm** and wait for several minutes for the configuration changes to take place. You may get INFO, WARN and sometimes **SEVERE** messages during this time. Wait until the update of the environment has completed, and **Health** is back to **Ok**.
- Go back to your EC2 instance (listed under **INSTANCES** – **Instances**) after some time and check under **Key Name**, you should now see your associated key pair. You may have to scroll all the way to the right to see Key name column.



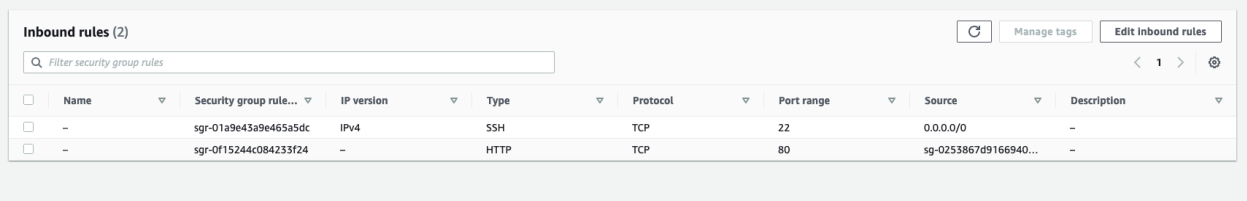

Notice the **Security Group** associated with your new environment.




### 4.3 Open port 22

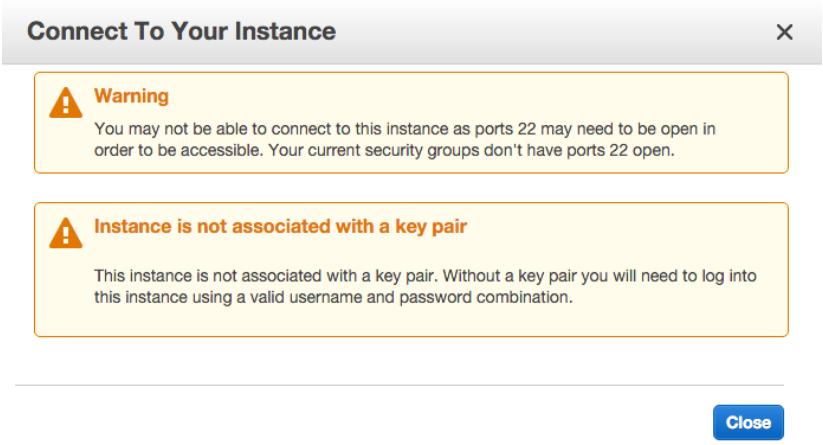
To open port 22, which is needed by SSH, follow these steps:

1. Go to the EC2 Management Console.
2. On left menu, under **NETWORK & SECURITY**, click on **Security Groups**.
3. Select the security group (present as a link) configured for your instance.
4. For the security group, edit (or verify) the "Inbound rules" (**Inbound** tab present on the bottom of the pane) by clicking the **Edit** button.
5. If missing, add a new rule for Type = SSH, Protocol = TCP, Port Range = 22, Source = Custom 0.0.0.0/0. Click **Save**. If rule is already present, **do nothing.**



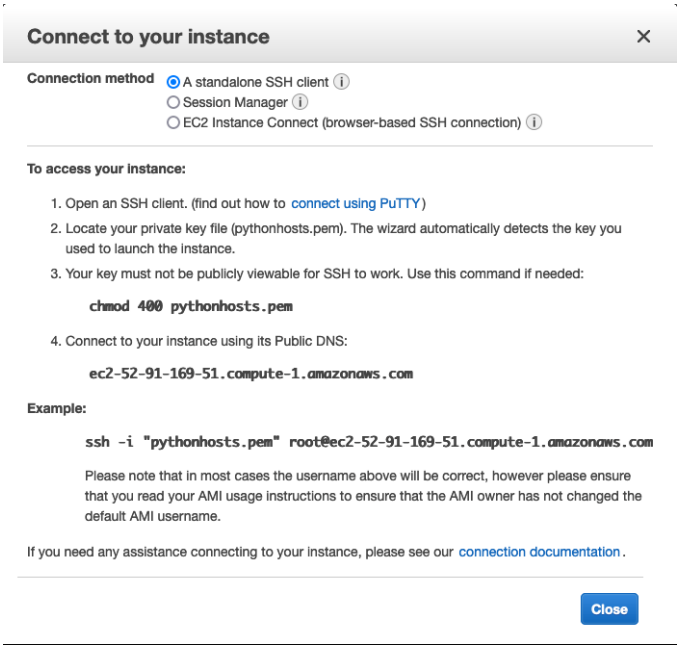

#### 4.3.1 Errors when Connecting

If you fail to either open port 22 or associate your instance to a key pair, you will get an error popup when you try to **Connect to Your Instance** using EC2 Dashboard >> INSTANCES >> Instances >> select instance >> Connect, as show in the picture below.




### 4.4 Access your Linux Instance with SSH

- To see how to launch your SSH client go to **Services** and select **EC2**.
- Under the **INSTANCES** section in the navigation pane on the left, select **Instances**.
- Select your instance in the table (the check box turns **blue**) and select the **Connect** button next to **Launch Instance**.
- The **Connect to your instance** popup will display. Select the radio button **A standalone SSH client**. Notice the hyperlink "connect using PuTTY" (see section 7.4.2). See the snapshot below, showing Elastic IP connection string.




#### 4.4.1 Mac running MacOS / ssh

Change the permission of pythonphosts.pem first:

```
chmod 400 pythonhosts.pem
```

On a Mac you will need to enter a command like this one (when using **Public DNS**):

```
ssh -i "pythonhosts.pem" ec2-user@ec2-52-91-169-51.compute-
1.amazonaws.com
```

type **yes**, when asked. Make sure that you are executing the ssh command in the same folder that contains the key. Also replace the "**root**" user with "**ec2-user**". You should see output like this one (using **Public DNS**):

```
$ ssh -i "phphosts.pem" ec2-user@ec2-204-236-235-251.compute-
1.amazonaws.com

The authenticity of host 'ec2-52-91-169-51.compute-1.amazonaws.com
(52.91.169.51)' can't be established.
ED25519 key fingerprint is
SHA256:IJddRQXOnpKn3Lg1jNDNAhgVeuVIJ+BoSg58KbR2XC8.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-52-91-169-51.compute-1.amazonaws.com'
(ED25519) to the list of known hosts._____ _           _   _      ____
_        _ _
 | ____| | __     ___| |_(_) ___| __ )  ___  __ _ _ __   ___| |_ __  _| | | __
 |  _| | |/ _ \/ __| __| |/ __|  _ \ / _ \/ _\ | '_ \/ __| __/ _\ | | |/ /
 | |___| | (_| \__ \ |_| | (__| |_) |  __/ (_| | | | \__ \ || (_| | |   <
 |_____|_|\__,_|___/\__|_|\___|____/ \___|\__,_|_| |_|___/\__\__,_|_|_|\_\

 Amazon Linux 2 AMI

 This EC2 instance is managed by AWS Elastic Beanstalk. Changes made via SSH
 WILL BE LOST if the instance is replaced by auto-scaling. For more
information
 on customizing your Elastic Beanstalk environment, see our documentation
here:
 http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/customize-containers-
ec2.html

[ec2-user@ip-172-31-95-188 ~]$
```

You can find more info here:

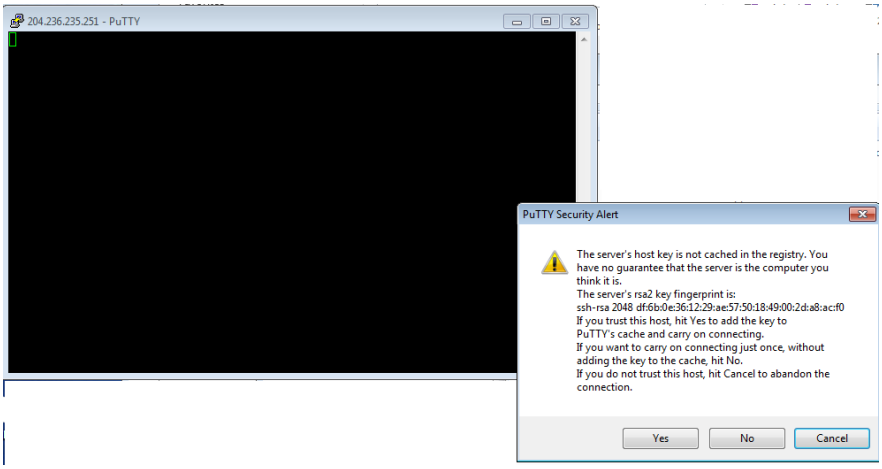
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AccessingInstances.html?console_help=true

### 4.4.2 PC running Windows / PuTTY

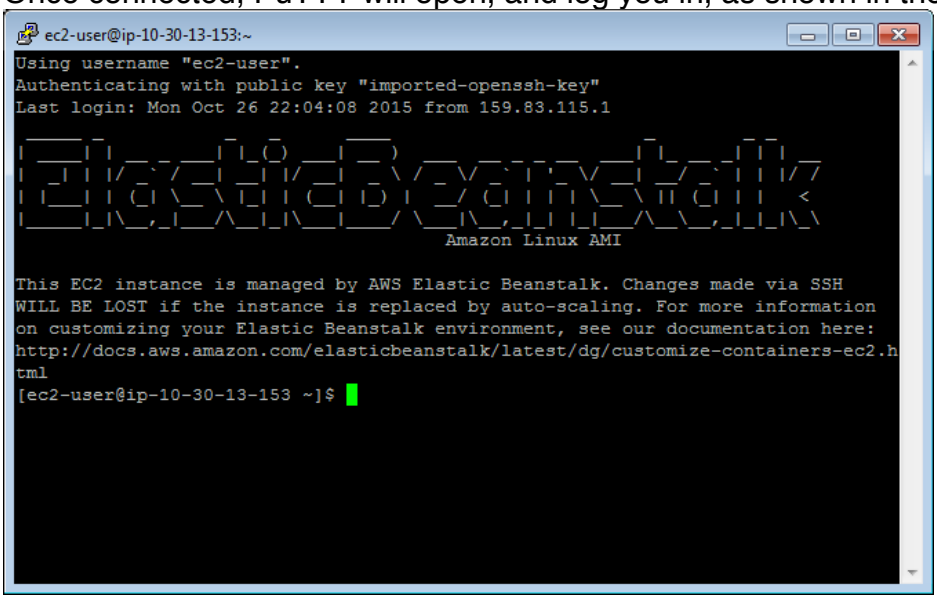
In the popup windows titled **Connect To Your Instance**, click on **Connect using PuTTY**. You will be redirected to the URL.

http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/putty.html

Follow the steps under **Starting a PuTTY Session** to connect to your the Linux instance using PuTTY. The first time you connect by clicking **Open** to start the session, PuTTY displays a **PuTTY Security Alert** dialog box, as show in the following snapshot. Click the **Yes** button.




Once connected, PuTTY will open, and log you in, as shown in the next snapshot.




As with SSH, you can either use tout Public DNS or your Elastic IP to log in.

### 4.5 Explore

You can now explore your Instance. When you log in with SSH, your account home directory will be located at:

**/home/ec2-user**

That folder is empty and is not where your **Python** files are. Run 'ps ax', and you should see several instances of **nginx** and **Python 3**:

```
[ec2-user@ip-172-31-19-89 ~]$ ps ax
  PID TTY      STAT   TIME COMMAND
...
3498 ?        Sl     0:00 /var/app/venv/staging-LQM1lest/bin/python
3254 ?        Ssl    0:01 /usr/bin/python3 /opt/aws/bin/cfn-hu
...
3479 ?        Ss     0:00 nginx: master process /usr/sbin/nginx
3483 ?        S      0:00 nginx: worker process
[ec2-user@ip-172-31-19-89 ~]$
```

To see your mounted volumes, and your 8GB of free space run 'df -h':

```
[ec2-user@ip-172-31-95-188 ~]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        484M     0  484M   0% /dev
tmpfs           492M     0  492M   0% /dev/shm
/dev/xvda1      8.0G  2.4G  5.7G  29% /
[ec2-user@ip-172-31-95-188 ~]$
```

To see the nginx folders, run 'ls /etc/nginx':

```
[ec2-user@ip-172-31-95-188 ~]$ ls /etc/nginx
conf.d     fastcgi.conf          fastcgi_params          koi-utf  mime.types
nginx.conf          scgi_params          uwsgi_params          win-utf
default.d  fastcgi.conf.default  fastcgi_params.default  koi-win
mime.types.default  nginx.conf.default  scgi_params.default
uwsgi_params.default
[ec2-user@ip-172-31-95-188 ~]$
```

To see the Python files you uploaded and deployed, run 'cd /var/app/current':

```
[ec2-user@ip-172-31-95-188 current]$ cd /var/app/current/
[ec2-user@ip-172-31-95-188 current]$ ls -l
total 12
-rw-rw-rw- 1 webapp webapp 1170 Feb  1 10:08 application.py
-rw-r--r-- 1 root   root     72 Feb  1 10:43 Procfile
drwxr-xr-x 2 webapp webapp   40 Feb  1 10:43 __pycache__
-rw-r--r-- 1 webapp webapp   94 Feb  1 10:07 requirements.txt
[ec2-user@ip-172-31-95-188 current]$
```

To see the Python application file that creates the "sample application" HTML page:

```
[ec2-user@ip-172-31-95-188 current]$ more application.py
```

**Have fun exploring AWS!!**