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 at AWS by signing up to AWS Educate. 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. Pre-requisites

All new **AWS Educate** student members enrolling at member institutions like USC will now receive **$100 in AWS usage credit** pre-loaded into an **AWS Educate Starter account** at signup.

Students **no longer** need to enter an **AWS Account ID** or select an account type after registering. They may now go directly into their student portal and enjoy the benefits of AWS Educate.

AWS Educate Starter Accounts also come with a **capped amount of usage** and **do not require a credit card** to sign up, eliminating the risk of members overspending if a service is not shut off.

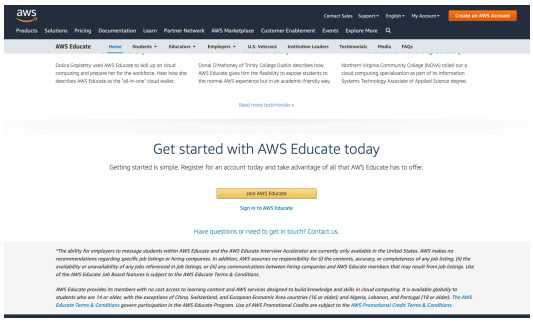
AWS Educate Starter Accounts do have limited AWS service capability. To view the list of services supported by AWS Educate Started Accounts, you can go here:

https://s3.amazonaws.com/awseducate-starter-account-services/AWS_Educate_Starter_Accounts_and_AWS_Services.pdf

2. Sign up for AWS Educate

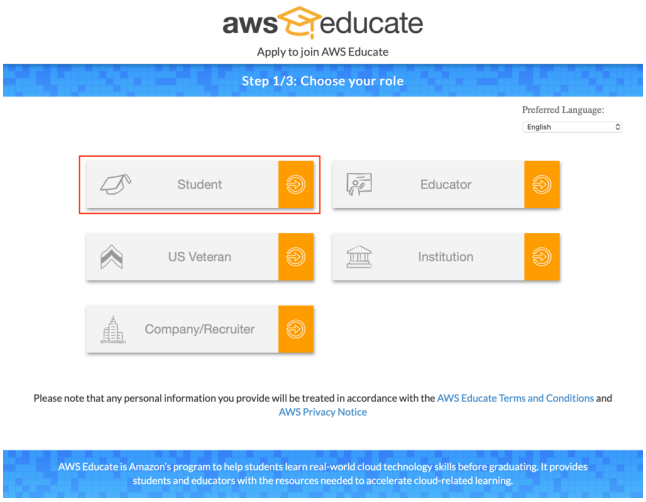
To sign up for AWS Educate go to:

http://aws.amazon.com/education/awseducate

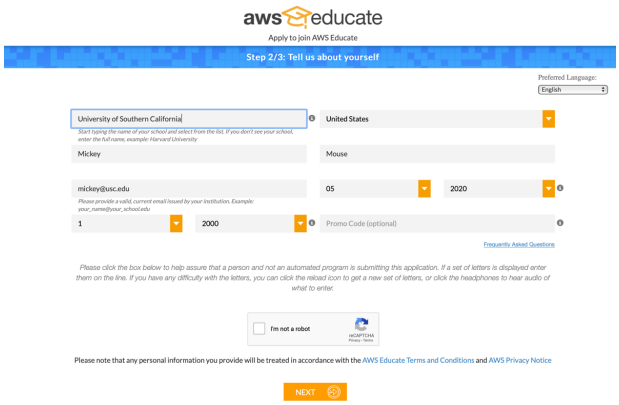


Scroll down to the bottom of the form and click on the ***Join AWS Educate*** button.

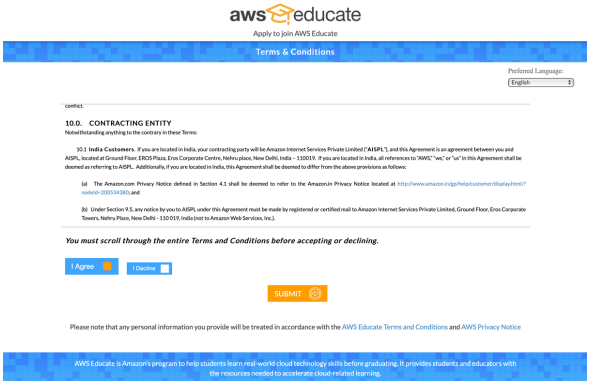
On Step 1, click on the arrow in ***Student*** button.



On Step 2, fill out the form appropriately. **University of Southern California** will auto complete. Leave the **Promo Code** field empty, click the reCAPTCHA "I'm not a robot", and click **NEXT**.

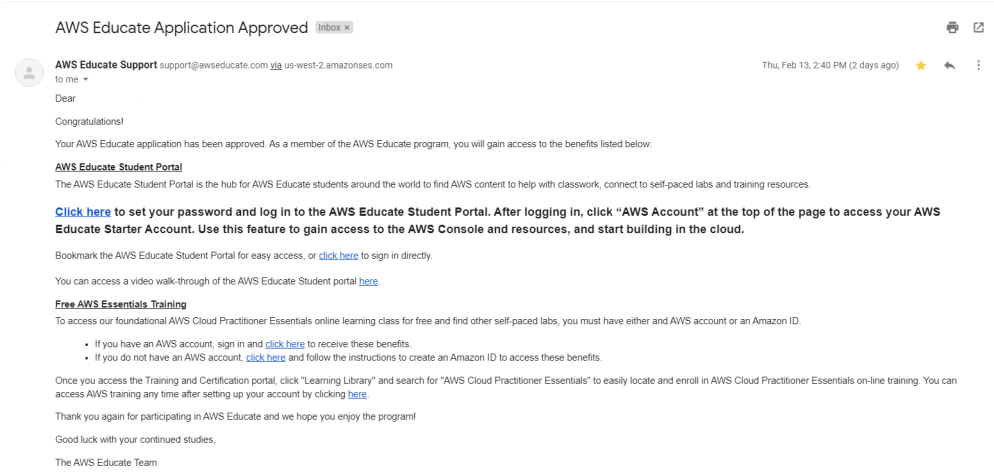


On Step 3, you **must scroll** all the way to view the complete **Terms and Conditions**.



On Step 4, select the checkbox **I Agree**. Click **SUBMIT** and finish the sign-up process.

After your application is reviewed and approved, usually within 24 hours, you will receive a **welcome e-mail** from AWS Educate Support, which includes details to **set your password** and log in to the **AWS Educate Student Portal**, as shown below:



AWS Educate Application Approved Inbox ×

AWS Educate Support support@awseducate.com via us-west-2.amazonses.com Thu, Feb 13, 2:40 PM (2 days ago)
to me

Dear

Congratulations!

Your AWS Educate application has been approved. As a member of the AWS Educate program, you will gain access to the benefits listed below:

AWS Educate Student Portal

The AWS Educate Student Portal is the hub for AWS Educate students around the world to find AWS content to help with classwork, connect to self-paced labs and training resources.

Click here to set your password and log in to the AWS Educate Student Portal. After logging in, click "AWS Account" at the top of the page to access your AWS Educate Starter Account. Use this feature to gain access to the AWS Console and resources, and start building in the cloud.

Bookmark the AWS Educate Student Portal for easy access, or click here to sign in directly.

You can access a video walk-through of the AWS Educate Student portal here.

Free AWS Essentials Training

To access our foundational AWS Cloud Practitioner Essentials online learning class for free and find other self-paced labs, you must have either and AWS account or an Amazon ID.

- If you have an AWS account, sign in and click here to receive these benefits.
- If you do not have an AWS account, click here and follow the instructions to create an Amazon ID to access these benefits.

Once you access the Training and Certification portal, click "Learning Library" and search for "AWS Cloud Practitioner Essentials" to easily locate and enroll in AWS Cloud Practitioner Essentials on-line training. You can access AWS training any time after setting up your account by clicking here.

Thank you again for participating in AWS Educate and we hope you enjoy the program!

Good luck with your continued studies,

The AWS Educate Team

2.2 Issues Sign up for AWS Educate

If you are having issues signing up for AWS Educate, and your initial application is rejected, you can create a Support Case at https://console.aws.amazon.com/support, where you describe the problem and attached a copy of the front and back of your USC ID. Alternatively you could also contact AWS Educate Support directly at:

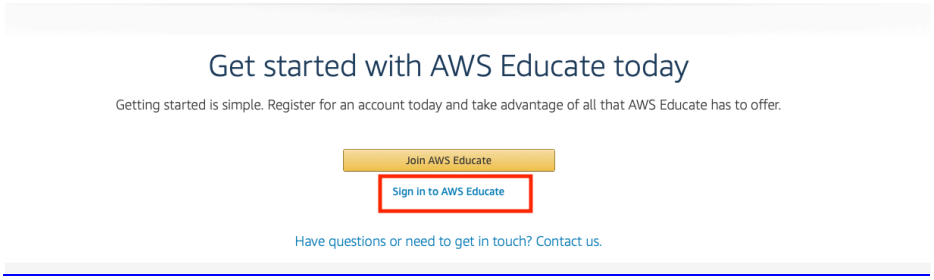
http://aws.amazon.com/education/awseducate/contact-us

Review of your response to the rejection can take between 1 or 2 days.

3. Login to AWS Educate

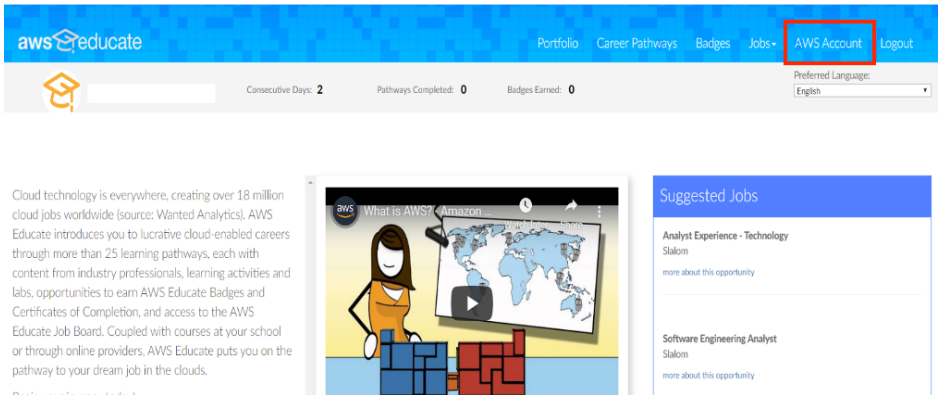
Once your AWS Educate application has been approved, go to:

http://aws.amazon.com/education/awseducate

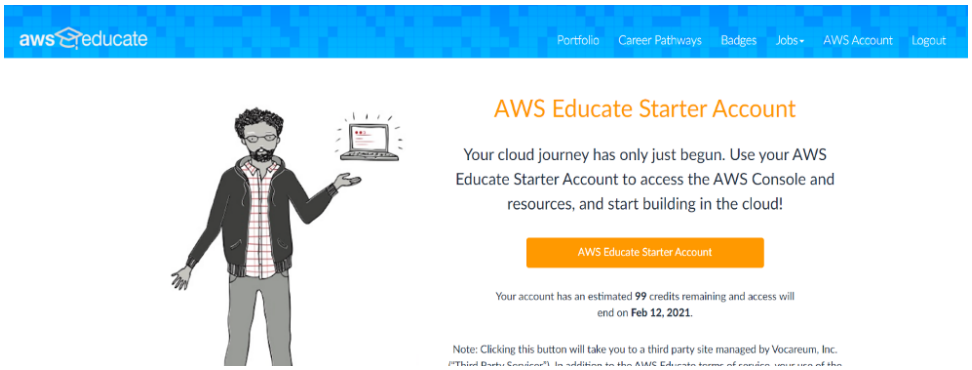


Again, scroll down to the bottom of the page and click on **Sign in to AWS Educate**.

Enter your email and password as requested. You should be redirected to the **aws educate** home page.



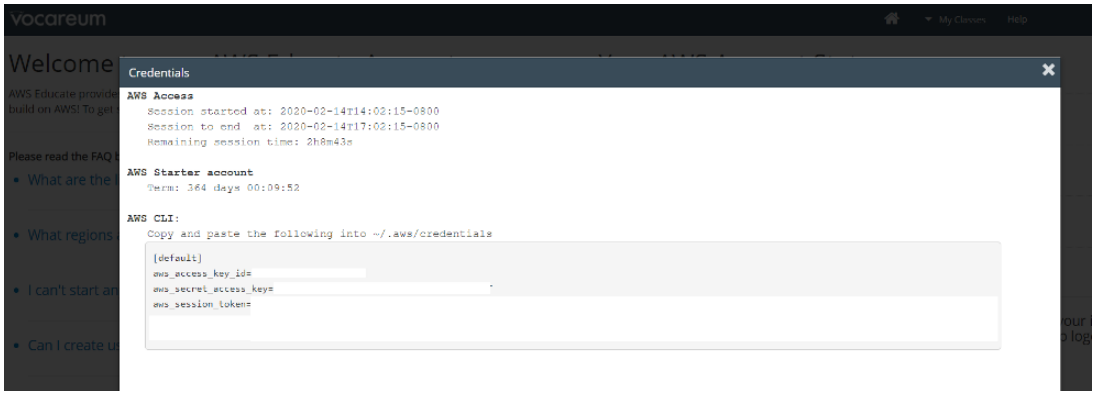
On the top right navigation bar, select **AWS Account**. You will be taken to a page where you can view your **applied credits** and **validity** of your Starter Account.



Click on the orange button labelled "**AWS Educate Starter Account**", which will take you to **vocareum.** Scroll down to the end of the **Terms and Conditions** page, and click **I Agree**. You will be directed to the **vocareum Dashboard** as shown below.

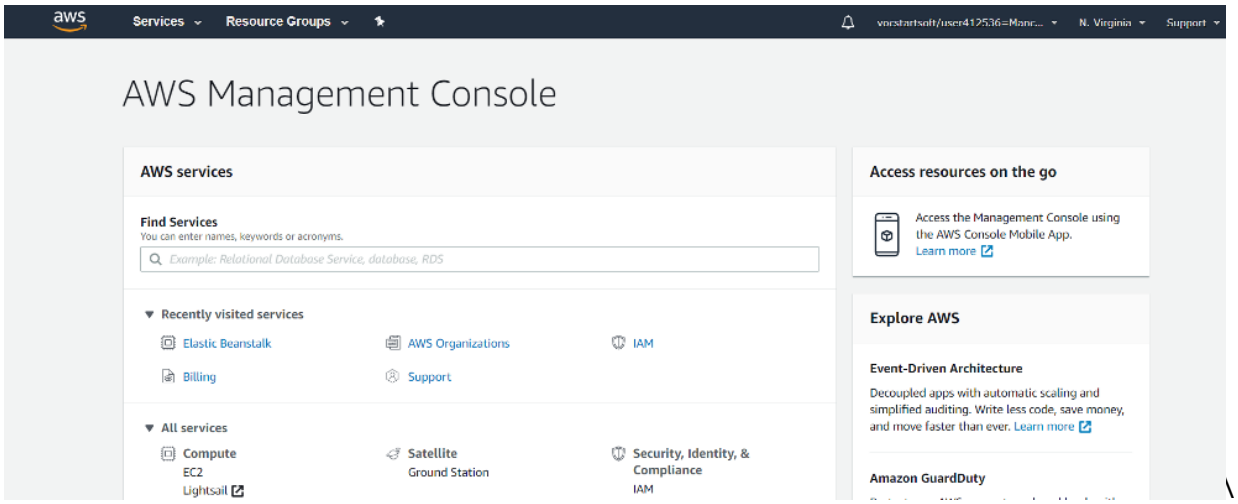
Clicking on the **Account Details** button will display a modal window with your Credentials.

Click on **Show** to display your **AWS CLI** keys and token.



In the vocareum page, click the **AWS Console** button and you will be taken to your **AWS Management Console** page. Please note that you may have to **Allow pop-up windows** in your browser settings for this Website, if pop-up windows are **Blocked** in your browser.

The AWS Management Console is similar to the one viewed by regular AWS members, with a number of restrictions (like the region and the services allowed).



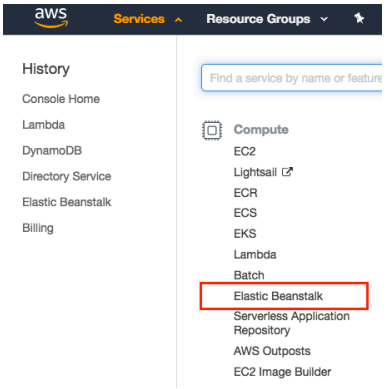
On the top right corner, you can find:

1) Your account name – for example vocstartoft/user623430=papa@usc.edu
2) AWS Deployment Region – US East (N.Virginia)

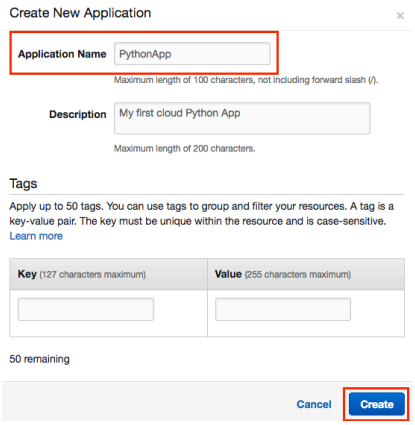
Important Note: Notice item 2) above. **AWS Educate Starter Accounts** are limited to use only the **us-east-1** region.

4. Set up the Default Elastic Beanstalk Application

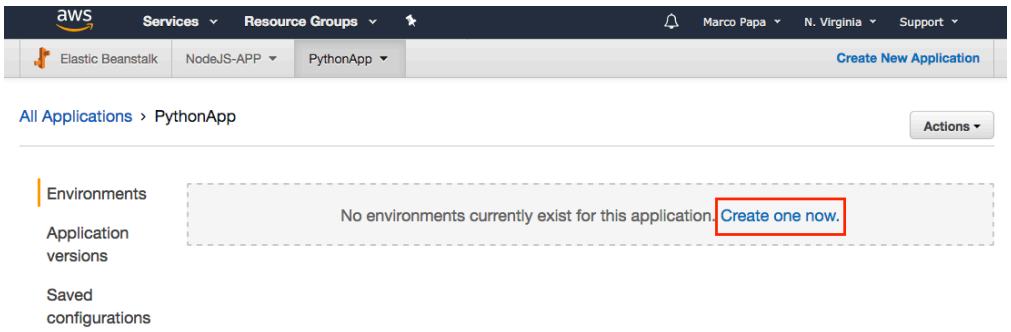
- Click the top left menu named **Services**
- From the list of Amazon Web Services, select **Elastic Beanstalk,** under **Compute**.



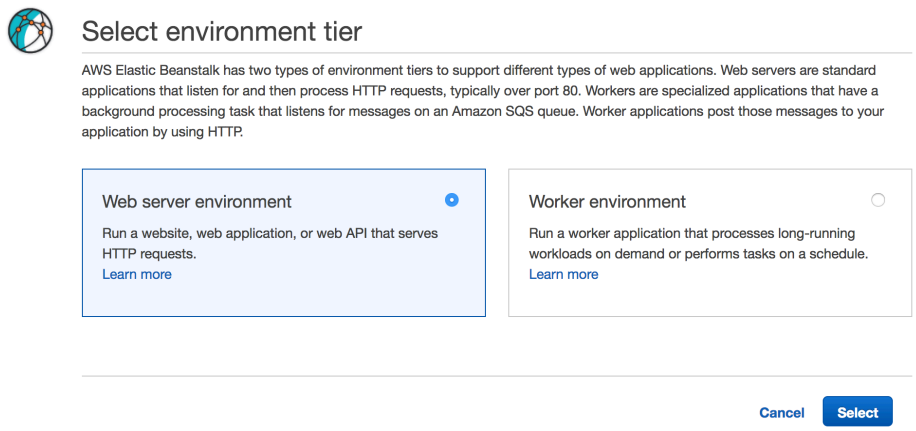
- Select **Create New Application** in the top right, right underneath your account name, and follow the Wizard.
- In the **Application Name** field, enter a name for your application.



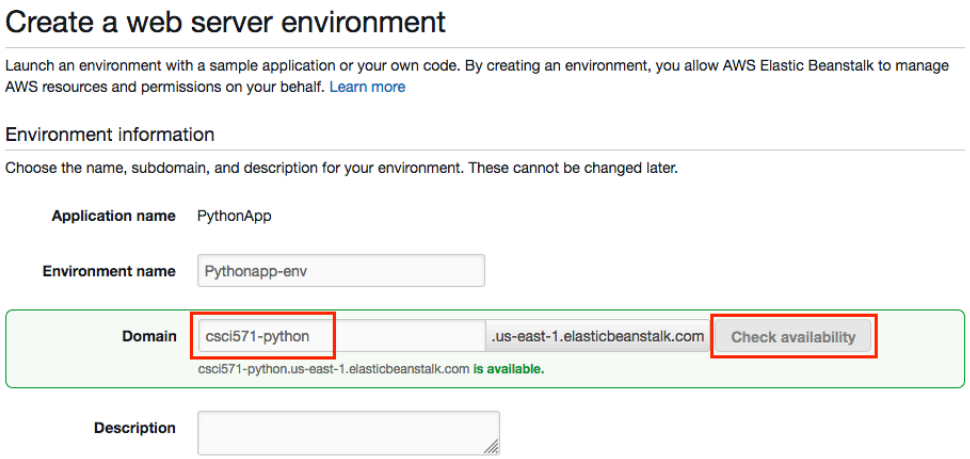
- Click **Create**.
- In the **Environment** section click on the **Create One now** hyperlink



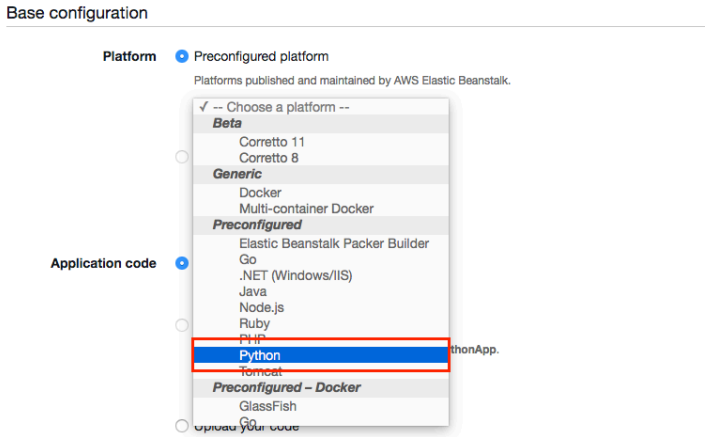
- In the **Choose an environment tier** dialog select **Web server environment** and click on **Select** button.



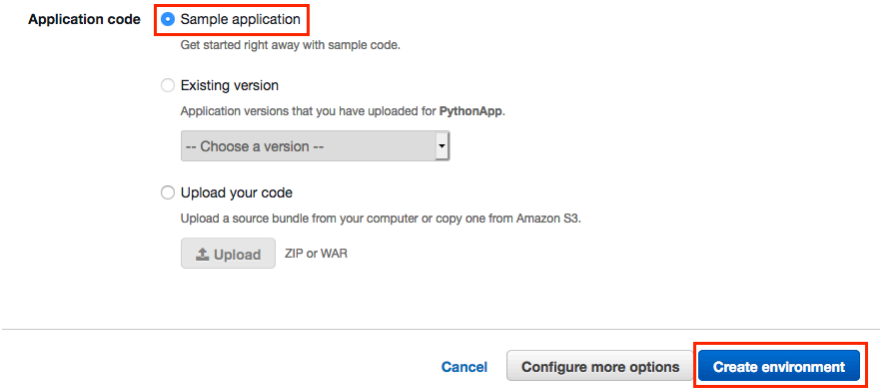
- In the **Environment Information** section, select a **Domain** (use the default or check availability of your own subdomain of elasticbeanstalk.com). Click on "**Check availability"** button. Your URL should be green. Otherwise you should change the environment URL.



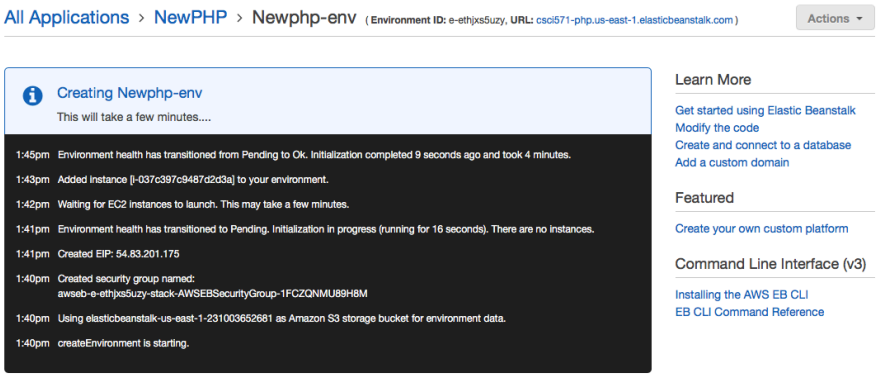
- In the **Base configuration** section, choose the **Preconfigured platform**, and the following option in the drop-down list:
    - Choose a platform: ***Preconfigured*** **- Python**



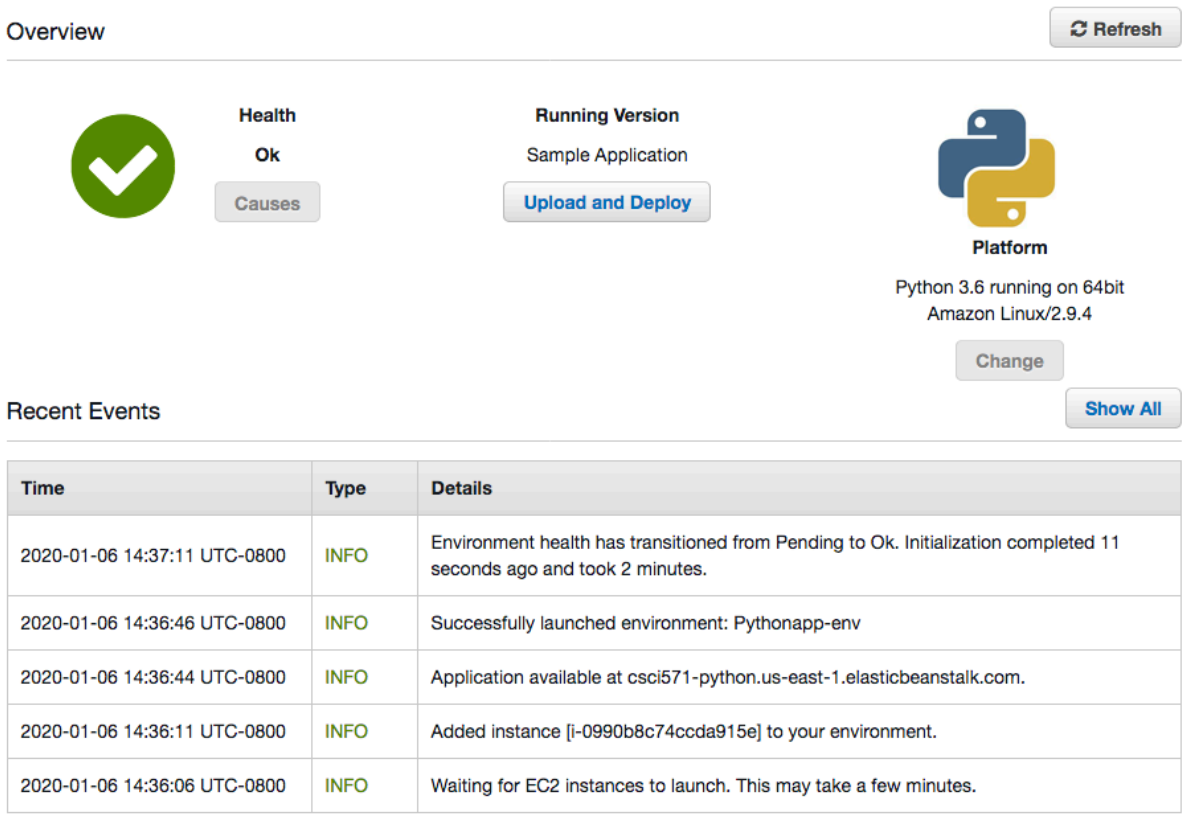
- In the **Application Code** section, select **Sample application.**



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



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. a *rotating wheel* next to **the "Monitor" button**. Once creation and launch are completed, you will see the wheel turn into a green round circle with a check mark in the middle.

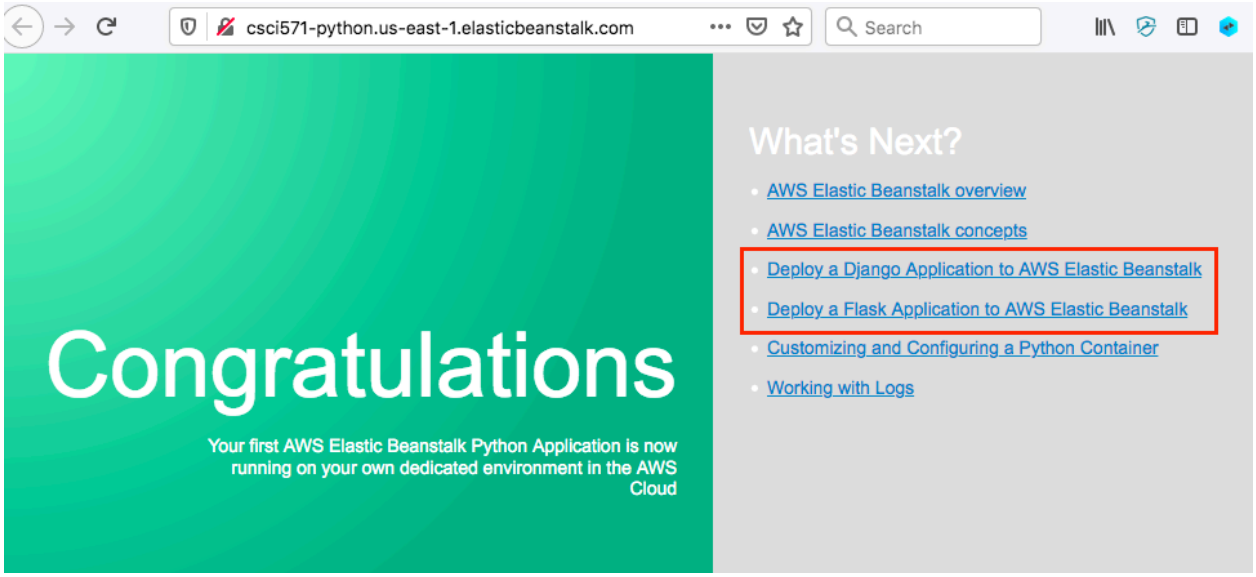


Python Instance Dashboard

Beside the "**Environment ID**", there is a **URL** such as:

YourDomainName.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.

5. Deploy your Python application

5.1 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 3.7** and **pip3**. In the latter case you should change your shell startup files to point to Python 3.7 instead of Python 2.7.

Note: steps for Installing **Python 3.7** 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**).

5.2 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==1.0.2
```

d. View installed libraries with `pip freeze`:

```
(env) $ pip freeze
Click==7.0
Flask==1.0.2
itsdangerous==1.1.0
Jinja2==2.10.3
MarkupSafe==1.1.1
Werkzeug==0.16.0
```

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:

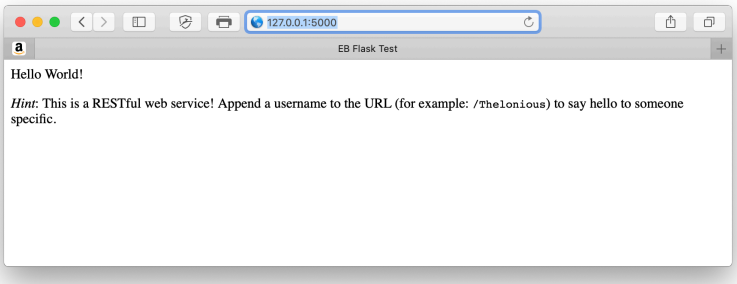
   https://csci571.com/hw/hw5/application.py

   save the file as **`application.py`**. 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:

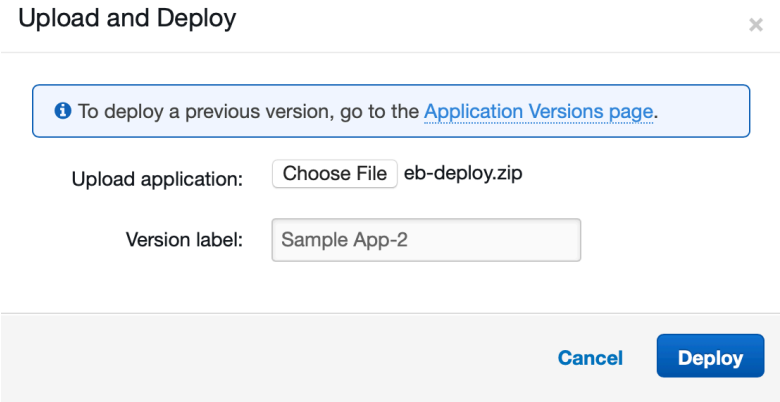


You can stop the web server and return to your virtual environment by typing **Ctrl+C**.

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:



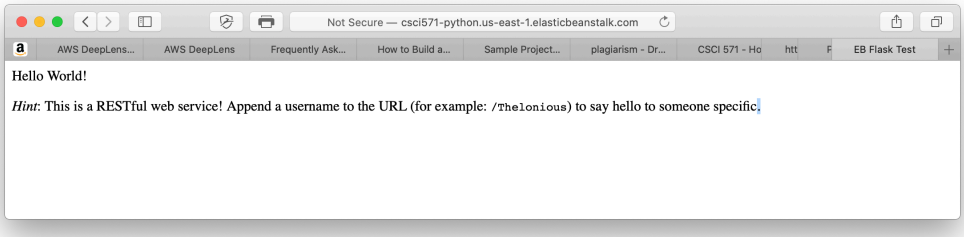
m. Choose the eb-deploy.zip file from your desktop. Enter a unique Version label. Click **Deploy**. The AWS EB server will restart and update your environment.

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.

5.3 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).

5.4 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.

6. 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.

6.1 Get and Setup SSH

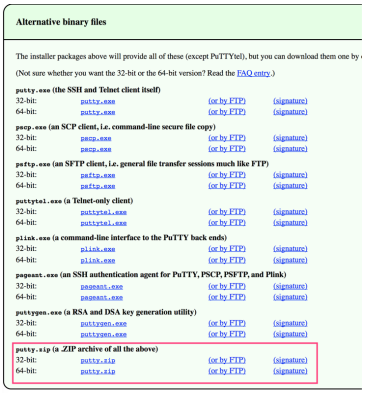
Once the Python app with SSH-enabled environment is running, you can get access using SSH. You can use **ssh** on a Mac running macOS, or **Putty** when running on Windows.

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)

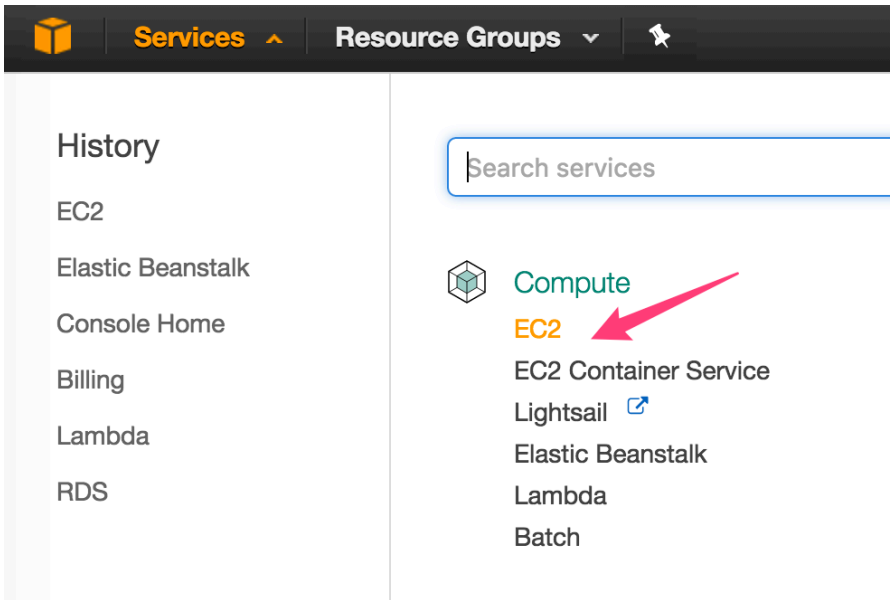
PuTTY needs additional setup as it needs to use a converted version of the private key. The instructions on how to perform such conversion are available here:

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).

6.2 Create a Key Pair

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

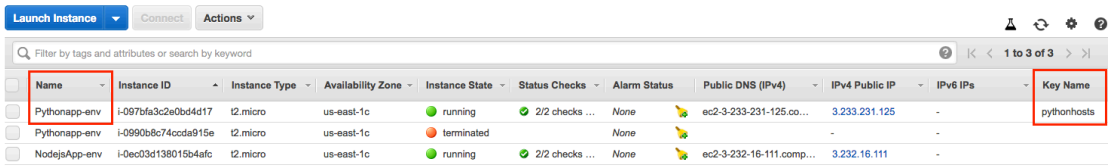


- Under **NETWORK AND SECURITY** select **Key Pairs**.
- Click on the button **Create Key Pair**.
- Enter a name like ***pythonhosts*** **(you must have your own random name!)** and click on **Create**.
- A download of your private key should start automatically. Save the key, like **pythonhosts.pem**, in an appropriate location.

6.2.1 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 environment but clicking anywhere in the "green" rectangle.
- Click on **Configuration** on the left menu.
- Click on the **Modify** button next to **Security**.
- Select the key pair you just created for the **EC2 key pair** field. Click **Refresh icon**.

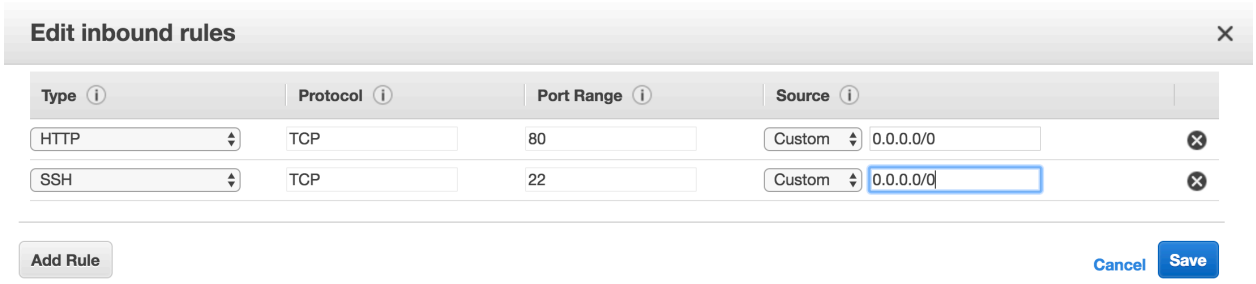
- Hit **Apply** and then **Confirm** and wait for several minutes for the configuration changes to take place. You may get INFO, WARN and sometimes **SEVERE** messages during this time. Wait until the update of the environment has completed, and **Health** is back to **Ok**.
- Go back to your EC2 instance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



6.3 Open port 22

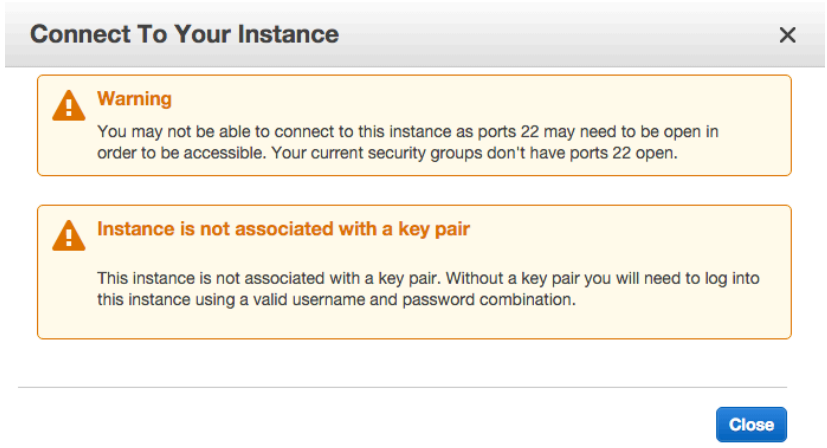
To open port 22, which is needed by SSH, follow these steps:

1. In the EC2 Management Console, click on **Instances**.
2. 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.**



6.3.1 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.

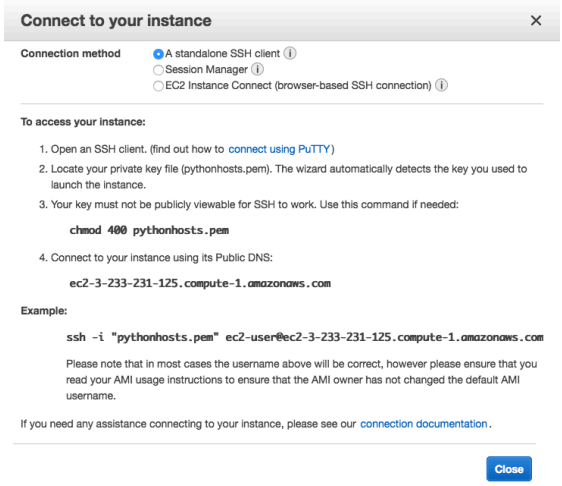


6.4 Access your Linux Instance with SSH

- 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.



6.4.1 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@ec2-3-233-231-125.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. 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

Last login: Tue Oct 27 16:22:06 2015 from 159.83.115.214

 _____ _           _   _       ____                        _        _ _
| ____| | __ _ ___| |_(_) ___|  _ )  ___  __ _ _ __  ___| |_ __ _| | | __
|  _| | |/ _` / __| __| |/ __|  _ \ / _ \/ _` | '_ \/ __| __/ _` | | |/ /
| |___| | (_| \__ \ |_| | (__| |_) |  __/ (_| | | | \__ \ || (_| | |   <
|_____|_|\__,_|___/\__|_|\___|____/ \___|\__,_|_| |_|___/\__\__,_|_|_|\_\
                                                  Amazon Linux 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-10-30-13-153 ~]$

Mac-mini:Desktop marcopapa$ chmod 400 pythonhosts.pem

Mac-mini:Desktop marcopapa$ ssh -i "pythonhosts.pem" ec2-
user@ec2-3-233-231-125.compute-1.amazonaws.com
The authenticity of host 'ec2-3-233-231-125.compute-
1.amazonaws.com (3.233.231.125)' can't be established.
ECDSA key fingerprint is
SHA256:u0+G0tC6c9OJOixq3uGWW7S9u8wb74kC8MW15qZ3yHw.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ec2-3-233-231-125.compute-
1.amazonaws.com,3.233.231.125' (ECDSA) to the list of known
hosts.
 ____ _          _  _     ___                  _       _ _
| ___| | __ _ ___| |_(_) ___| __ )  ___  __ _ _ __  ___| |_ __ _| | | __
|  _| | |/ _` / __| __| |/ __|  _ \ / _ \/ _` | '_ \/ __| __/ _` | | |/ /
| |__| | (_| \__ \ |_| | (__| |_) |  __/ (_| | | | \__ \ || (_| | |   <
|____|_|\__,_|___/\__|_|\___|____/ \___|\__,_|_| |_|___/\__\__,_|_|_|\_\
                                                  Amazon Linux 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-172-31-19-89 ~]$
```

You can find more info here:

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

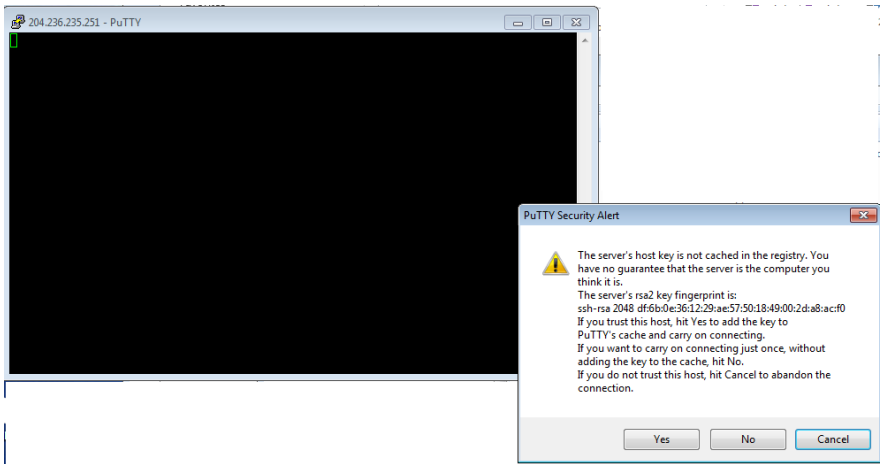
6.4.2 PC running Windows / PuTTY

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

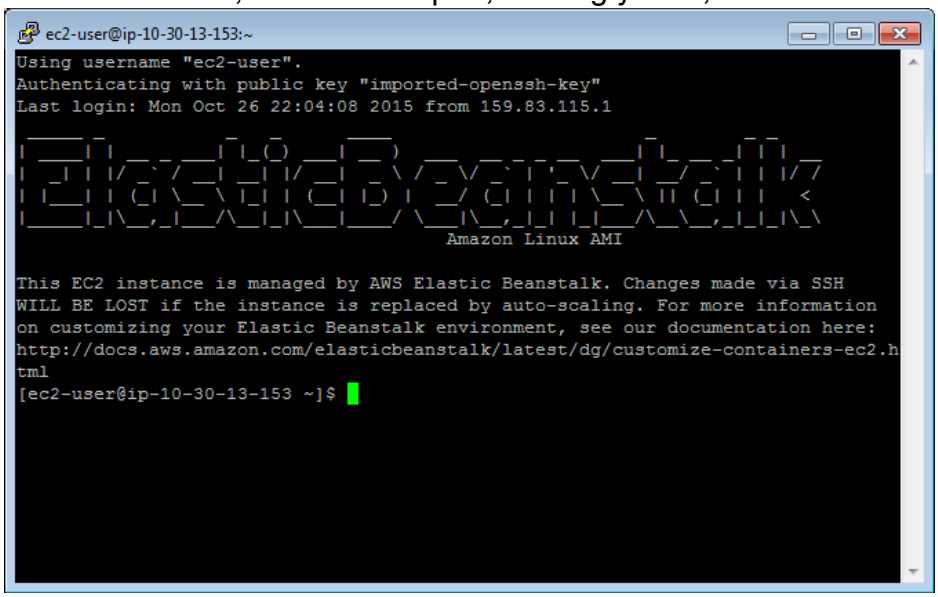
http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/putty.html

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.

6.5 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 Apache **httpd** and **Python 2.7**:

```
[ec2-user@ip-172-31-19-89 ~]$ ps ax
  PID TTY      STAT   TIME COMMAND
...
 3005 ?        Ss     0:00 /usr/bin/python2.7
/usr/local/bin/supervisord --nodaemon -c /opt/python
...
 1940 ?        S      0:00 /usr/sbin/httpd -D FOREGROUND
 1941 ?        Sl     0:00 /usr/sbin/httpd -D FOREGROUND
 1942 ?        Sl     0:00 /usr/sbin/httpd -D FOREGROUND
 1944 ?        Sl     0:00 /usr/sbin/httpd -D FOREGROUND
 1945 ?        Sl     0:00 /usr/sbin/httpd -D FOREGROUND
...
 3325 ?        Ssl    2:07 /usr/bin/python2.7 /opt/aws/bin/cfn-
hup
...
[ec2-user@ip-172-31-19-89 ~]$
```

To see your mounted volumes, run 'df -h':

```
[ec2-user@ip-172-31-19-89 ~]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        483M   60K  483M   1% /dev
tmpfs           493M     0  493M   0% /dev/shm
/dev/xvda1      7.9G  1.6G  6.2G  21% /
[ec2-user@ip-172-31-19-89 ~]$
```

To see the Python (2.7 and 3.6) folders:

```
[ec2-user@ip-172-31-19-89 ~]$ ls /usr/bin/pyth*
/usr/bin/python            /usr/bin/python36
/usr/bin/python3.6m-x86_64-config
/usr/bin/python27          /usr/bin/python3.6
/usr/bin/python3-config
/usr/bin/python2.7         /usr/bin/python3.6-config
/usr/bin/python-config
/usr/bin/python2.7-config  /usr/bin/python3.6m
/usr/bin/python-config2
/usr/bin/python3           /usr/bin/python3.6m-config
[ec2-user@ip-172-31-19-89 ~]$
```

To see your Python files, located in the “bundle” folder:

```
[ec2-user@ip-172-31-19-89 ~]$ ls /opt/python
bin  bundle  current  etc  log  run
```

```
[ec2-user@ip-172-31-19-89 ~]$ ls /opt/python/bin
httpdlaunch
[ec2-user@ip-172-31-19-89 ~]$ ls -l /opt/python/bundle
total 4
drwxr-xr-x 3 root root 4096 Jan  6 23:53 2
[ec2-user@ip-172-31-19-89 ~]$ ls -l /opt/python/bundle/2
total 8
drwxr-xr-x 2 wsgi root 4096 Jan  6 23:53 app
-rw-r--r-- 1 root root  103 Jan  6 23:53 env
[ec2-user@ip-172-31-19-89 ~]$ ls -l /opt/python/bundle/2/app
total 12
-rw-r--r-- 1 wsgi root 5065 Apr  2  2015 application.py
-rw-r--r-- 1 wsgi root   84 Apr  2  2015 cron.yaml
[ec2-user@ip-172-31-19-89 ~]$ ls -l
/opt/python/bundle/2/app/application.py
-rw-r--r-- 1 wsgi root 5065 Apr  2  2015
/opt/python/bundle/2/app/application.py
```

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

```
[ec2-user@ip-172-31-19-89 ~]$ more
/opt/python/bundle/2/app/application.py
```

Note: the “bundle number” (2 above) will increase as you update and deploy new versions.

6.6 Additional Resources

An additional tutorial entitled “*Deploy a Python Web App*” on AWS, is available at:

https://aws.amazon.com/getting-started/projects/deploy-python-application/

This is a more complex application, that will incur some charges. We recommend deleting the environment once the app is tested.

Have fun exploring AWS!!