

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

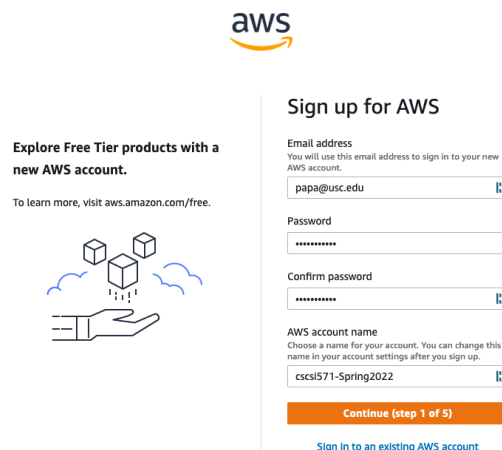
This semester we are allowing all students to explore cloud computing as offered by Amazon's Web Services. Using the instructions below one can establish a service account at AWS. Unfortunately, the AWS Educate has been terminated (USC belonged to it) and has been replaced by the AWS Academy. Since USC 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**.



The screenshot shows the AWS sign-up page. At the top is the AWS logo. Below it, on the left, is a section titled "Explore Free Tier products with a new AWS account." with a link to "aws.amazon.com/free" and an illustration of a hand holding a cube. On the right is the "Sign up for AWS" form. The form includes fields for "Email address" (containing "papa@usc.edu"), "Password", "Confirm password", and "AWS account name" (containing "cscs571-Spring2022"). There is an orange "Continue (step 1 of 5)" button and a link for "Sign in to an existing 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.



#### Free Tier offers

All AWS accounts can explore 3 different types of free offers, depending on the product used.



**Always free**  
Never expires



**12 months free**  
Start from initial sign-up date



**Trials**  
Start from service activation date

### Sign up for AWS

#### Contact Information

How do you plan to use AWS?

- ☐ Business - for your work, school, or organization
- ☒ Personal - for your own projects

Who should we contact about this account?

Full Name

Phone Number

Enter your country code and your phone number.

Country or Region

United States

Address

Apartment, suite, unit, building, floor, etc.

City

State, Province, or Region

Postal Code

☐ I have read and agree to the terms of the [AWS Customer Agreement](#)

Continue (step 2 of 5)

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



#### Secure verification

**i** We will not charge for usage below AWS Free Tier limits. We temporarily hold \$1 USD/EUR as a pending transaction for 3-5 days to verify your identity.



### Sign up for AWS

#### Billing Information

Credit or Debit card number



AWS accepts all major credit and debit cards. To learn more about payment options, review our [FAQ](#)

Expiration date

Month Year

Cardholder's name


Billing address


- ☒ Use my contact address
- 1502 Phelan Ln  
Redondo Beach CA 90278  
US
- ☐ Use a new address

Verify and Continue (step 3 of 5)

You might be redirected to your bank's website to authorize the verification charge.

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.





## Sign up for AWS

### Confirm your identity

Before you can use your AWS account, you must verify your phone number. When you continue, the AWS automated system will contact you with a verification code.

How should we send you the verification code?

☒ Text message (SMS)


☐ Voice call



Country or region code

United States (+1) ▼

Mobile phone number

Security check








Type the characters as shown above

Send SMS (step 4 of 5)

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.





## Sign up for AWS

### Confirm your identity


Verify code

5149

Continue (step 4 of 5)

Having trouble? Sometimes it takes up to 10 minutes to retrieve a verification code. If it's been longer than that, [return to the previous page](#) and try again.

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



## Sign up for AWS


### Select a support plan

Choose a support plan for your business or personal account. [Compare plans and pricing examples](#)

[You can change your plan anytime in the AWS Management Console.](#)


☒ **Basic support - Free**

- Recommended for new users just getting started with AWS
- 24x7 self-service access to AWS resources
- For account and billing issues only
- Access to Personal Health Dashboard & Trusted Advisor




☐ **Developer support - From \$29/month**

- Recommended for developers experimenting with AWS
- Email access to AWS Support during business hours
- 12 Business-hour response times



☐ **Business support - From \$100/month**

- Recommended for running production workloads on AWS
- 24x7 tech support via email, phone, and chat
- 1-hour response times
- Full set of Trusted Advisor best practice recommendations

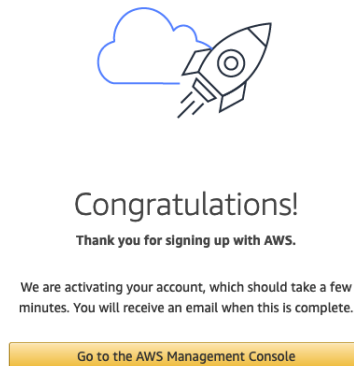


**Need Enterprise level support?**

From \$15,000 a month you will receive 15-minute response times and concierge-style experience with an assigned Technical Account Manager. [Learn more](#)

Complete sign up

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



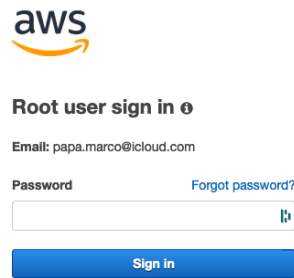
Wait a few minutes for your account to be activated, then click on **Go to the AWS Management Console**. The **AWS Sign in** form is displayed.

A screenshot of the AWS 'Sign in' page. It features the AWS logo at the top. Below the logo, the heading 'Sign in' is displayed. There are two selection options: 'Root user' (selected with a radio button) and 'IAM user'. The 'Root user' option includes the text 'Account owner that performs tasks requiring unrestricted access. Learn more'. The 'IAM user' option includes the text 'User within an account that performs daily tasks. Learn more'. Below these options is a text input field labeled 'Root user email address' containing the placeholder text 'username@example.com'. At the bottom is a blue button labeled 'Next'.

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

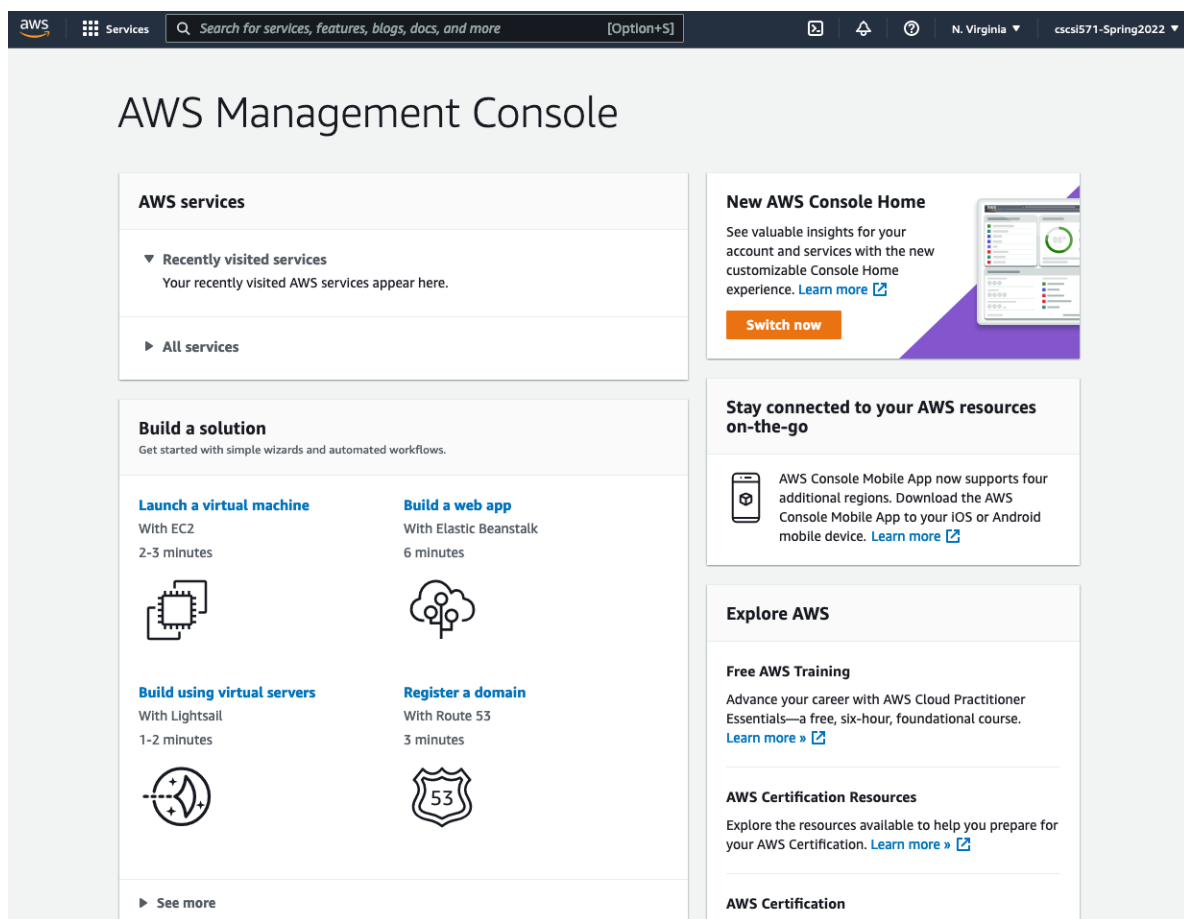
A screenshot of the AWS 'Security check' page. It features the AWS logo at the top. Below the logo, the heading 'Security check' is displayed. The main content area contains a box with the instruction 'Type the characters seen in the image below'. Inside this box is a CAPTCHA image showing the characters 'yt4dx2' with a noise filter. To the right of the image are two buttons: a speaker icon for audio and a circular arrow icon for refresh. Below the CAPTCHA box is a text input field containing the text 'yt4dx2'. At the bottom is a blue button labeled 'Submit'.

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



The image shows the AWS Root user sign-in form. At the top is the AWS logo. Below it is the heading "Root user sign in" with a small information icon. The email field is pre-filled with "papa.marco@icloud.com". There is a "Forgot password?" link next to the password field. The password field is empty and has a "Show/Hide" icon. At the bottom is a blue "Sign in" button.

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



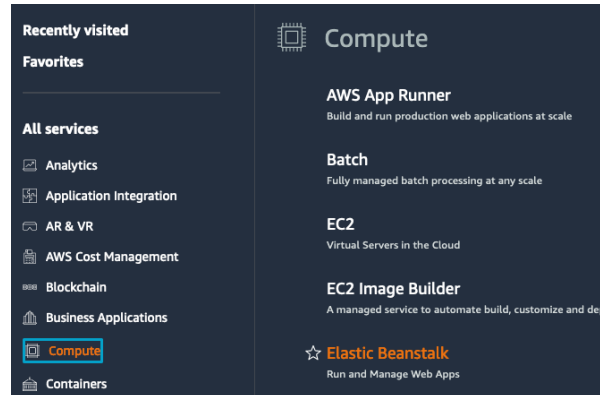
The image shows the AWS Management Console dashboard. The top navigation bar includes the AWS logo, a "Services" menu, a search bar, and account information: "N. Virginia" and "csci571-Spring2022". The main content area is titled "AWS Management Console". It features several sections: "AWS services" with "Recently visited services" and "All services"; "Build a solution" with four quick-start options: "Launch a virtual machine" (With EC2, 2-3 minutes), "Build a web app" (With Elastic Beanstalk, 6 minutes), "Build using virtual servers" (With Lightsail, 1-2 minutes), and "Register a domain" (With Route 53, 3 minutes); "New AWS Console Home" with a "Switch now" button; "Stay connected to your AWS resources on-the-go" with a link to the mobile app; and "Explore AWS" with links to "Free AWS Training" and "AWS Certification Resources".

On the top right corner, you can find:

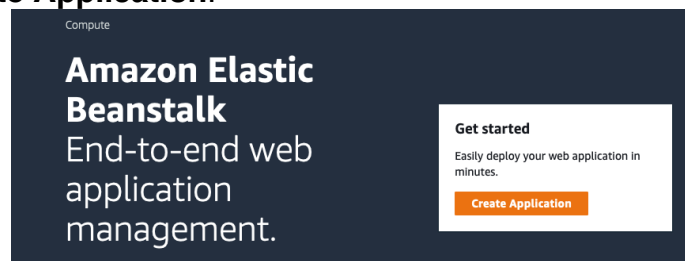
- 1) Your account name – for example csci571-Spring2022
- 2) AWS Deployment Region – US East (N.Virginia), or **us-east-1**

## 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.

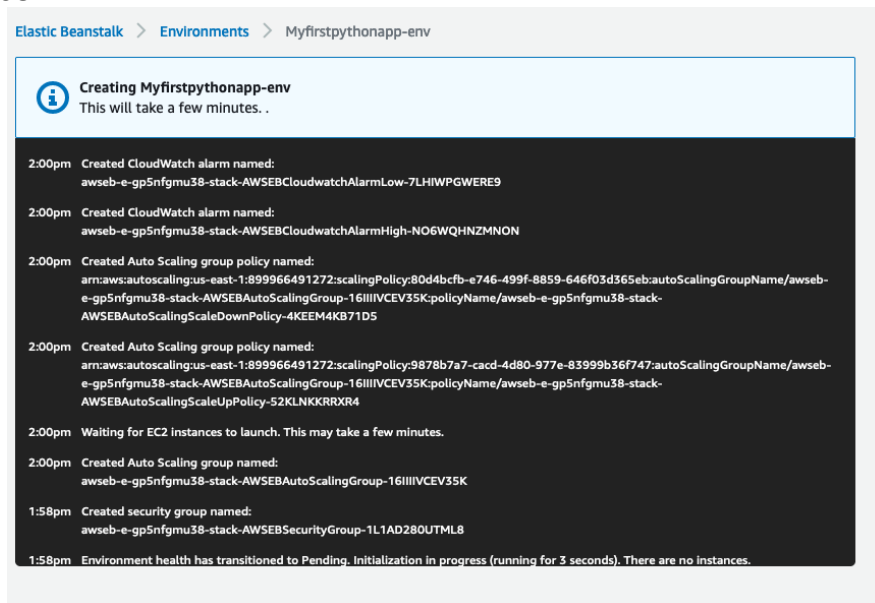
A screenshot of the 'Create a web app' form in the AWS console. The breadcrumb trail shows 'Elastic Beanstalk > Getting started'. The title is 'Create a web app' with a subtitle explaining that it creates a new application and environment. Below this, the 'Application information' section contains a text input field for 'Application name' with the value 'MyFirstPythonApp' entered. A note below the field states: 'Up to 100 Unicode characters, not including forward slash (/)'.

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

A screenshot of the 'Platform' drop-down menu. The menu is open, showing a list of options: '.NET Core on Linux', '.NET on Windows Server', 'Docker', 'GlassFish', 'Go', 'Java', 'Node.js', 'PHP', 'Python' (which is highlighted), 'Ruby', and 'Tomcat'. Below the list, there is a link to 'Upload a source bundle from your computer or copy one from Amazon S3'.

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

**Myfirstpythonapp-env**  
[Myfirstpythonapp-env.eba-vcivvym2.us-east-1.elasticbeanstalk.com](#) (e-gp5nfgmu38)  
Application name: **MyFirstPythonApp**

Refresh

Actions ▾

**Health**

  
Ok  


Causes

**Running version**

Sample Application  

Upload and deploy

**Platform**

  
Python 3.8 running on 64bit  
Amazon Linux 2/3.3.9  

Change

## 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.

← → ↻

🔒

csci571-python.us-east-1.elasticbeanstalk.com

⋮ 🔒 ☆

🔍

Search

🏠 🛡️ 📄 🌐

# Congratulations

Your first AWS Elastic Beanstalk Python Application is now running on your own dedicated environment in the AWS Cloud

### What's Next?

- [AWS Elastic Beanstalk overview](#)
- [AWS Elastic Beanstalk concepts](#)
- [Deploy a Django Application to AWS Elastic Beanstalk](#)
- [Deploy a Flask Application to AWS Elastic Beanstalk](#)
- [Customizing and Configuring a Python Container](#)
- [Working with Logs](#)

## 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](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 **requirements.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:

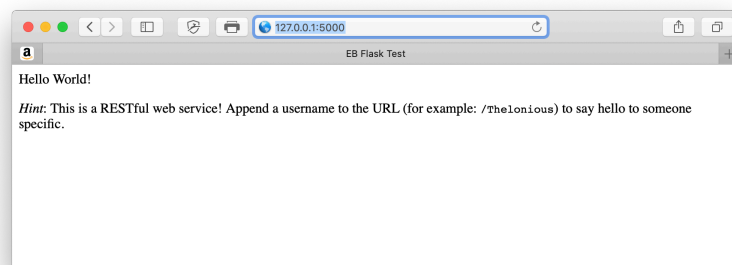
<https://csci571.com/hw/hw5/application.py>

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:

**Upload and deploy**

To deploy a previous version, go to the [Application Versions page](#).

Upload application

File name : **eb-deploy.zip** ✓

Version label

► Deployment Preferences

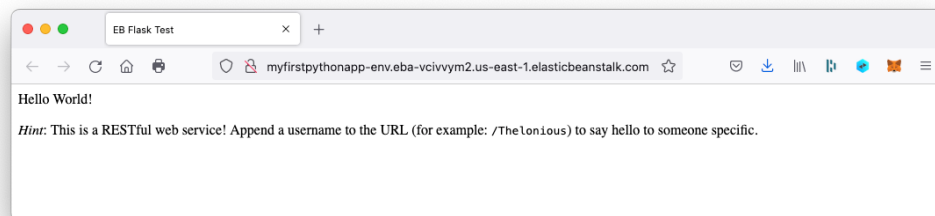
Current number of instances: 1

- 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 PuTTYgen), but you can download them one by one by . (Not sure whether you want the 32-bit or the 64-bit version? Read the <a href="#">FAQ entry</a> .)			
<b>putty.exe</b> (the SSH and Telnet client itself)			
32-bit	putty.exe	(or by FTP)	(signature)
64-bit	putty.exe	(or by FTP)	(signature)
<b>pscp.exe</b> (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)
<b>psftp.exe</b> (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)
<b>puttytel.exe</b> (a Telnet-only client)			
32-bit	puttytel.exe	(or by FTP)	(signature)
64-bit	puttytel.exe	(or by FTP)	(signature)
<b>plink.exe</b> (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)
<b>pageant.exe</b> (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)
<b>puttygen.exe</b> (a RSA and DSA key generation utility)			
32-bit	puttygen.exe	(or by FTP)	(signature)
64-bit	puttygen.exe	(or by FTP)	(signature)
<b>putty.zip</b> (a ZIP archive of all the above)			
32-bit	putty.zip	(or by FTP)	(signature)
64-bit	putty.zip	(or by FTP)	(signature)

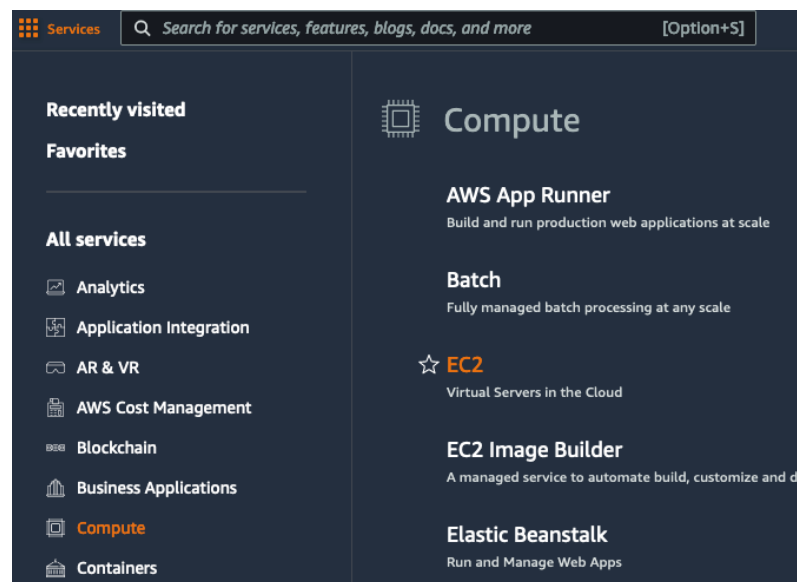
**PuTTY** needs additional setup as it needs to use a converted version of the private key. The instructions on how to perform such conversion are available here:

<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**.

EC2 > Key pairs > Create key pair

## Create key pair [Info](#)

**Key pair**  
A key pair, consisting of a private key and a public key, is a set of security credentials that you use to prove your identity when connecting to an instance.

Name  
pythonhosts  
The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type [Info](#)  
☒ RSA  
☐ ED25519

Private key file format  
☒ .pem  
For use with OpenSSH  
☐ .ppk  
For use with PuTTY

Tags (Optional)  
No tags associated with the resource.

[Add tag](#)  
You can add 50 more tags.

[Cancel](#) [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**.

**Virtual machine permissions**

EC2 key pair  
pythonhosts ▼ ↻

IAM instance profile  
aws-elasticbeanstalk-ec2-role ▼ ↻

[Cancel](#) [Continue](#) [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.



Launch Instance <span>Connect</span> <span>Actions</span>									
Filter by tags and attributes or search by keyword									
	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	Key Name
<input type="checkbox"/>	Pythonapp-env	i-097bfa3c2e0bd4d17	t2.micro	us-east-1c	running	2/2 checks ...	None	ec2-3-233-231-125.co...	pythonhosts
<input type="checkbox"/>	Pythonapp-env	i-0990b8c74c0da915e	t2.micro	us-east-1c	terminated	2/2 checks ...	None	-	-
<input type="checkbox"/>	NodejsApp-env	i-0ec03d138015b4afc	t2.micro	us-east-1c	running	2/2 checks ...	None	ec2-3-232-16-111.comp...	-

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

Key Name	Monitoring	Launch Time	Security Groups	Owner
pythonhosts	disabled	February 1, 2022 at 2:42:34 ...	<a href="#">awseb-e-gp5nfgmu38-stack-AWSEBSecurityGroup-1L1AD280UTML8</a>	899966491272
	disabled	January 31, 2022 at 1:59:17 ...		899966491272

## 4.3 Open port 22

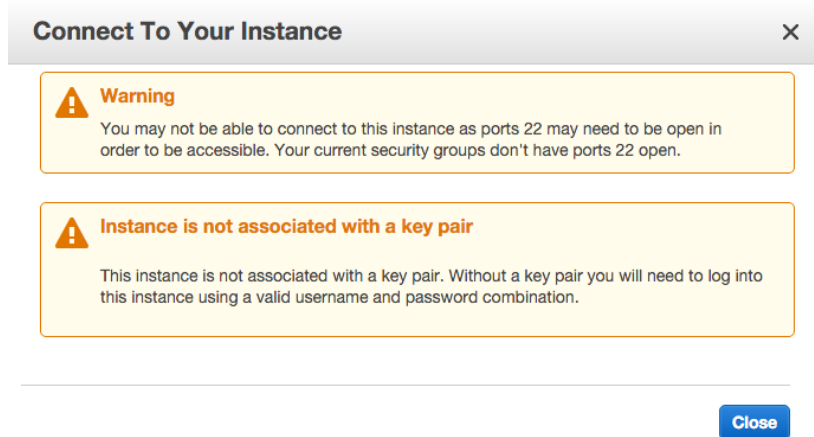
To open port 22, which is needed by SSH, follow these steps:

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**.

Inbound rules (2)									
Filter security group rules									
<input type="checkbox"/>	Name	Security group rule...	IP version	Type	Protocol	Port range	Source	Description	
<input type="checkbox"/>	-	sg-01a9e43a9e465a5dc	IPv4	SSH	TCP	22	0.0.0.0/0	-	
<input type="checkbox"/>	-	sg-0f15244c08423f24	-	HTTP	TCP	80	sg-0253867d9166940...	-	

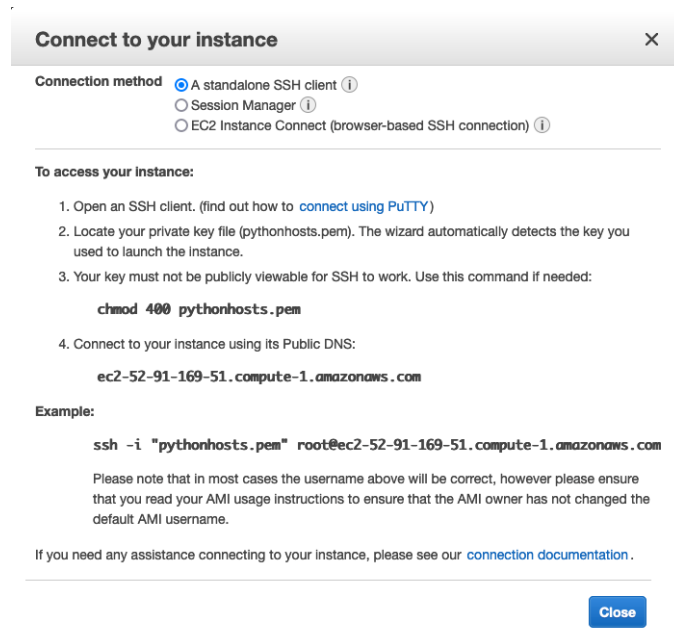
### 4.3.1 Errors when Connecting

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.
```

[illegible]

## 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:

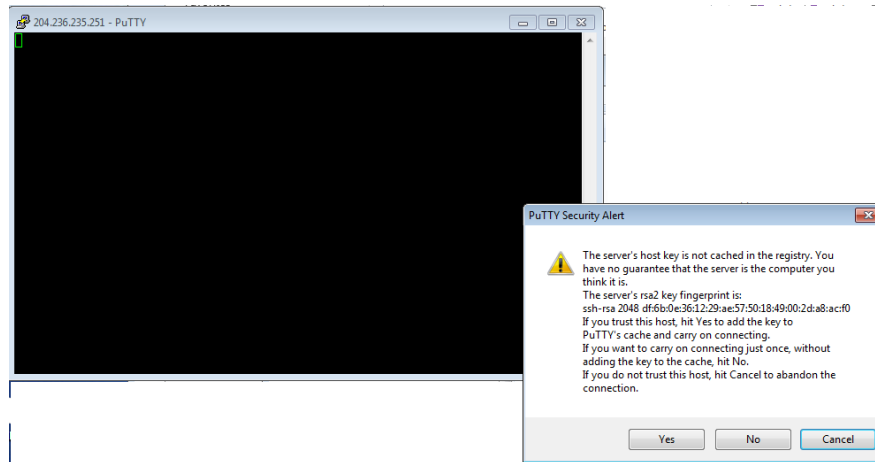
[https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AccessingInstances.html?console\\_help=true](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AccessingInstances.html?console_help=true)

#### 4.4.2 PC running Windows / PuTTY

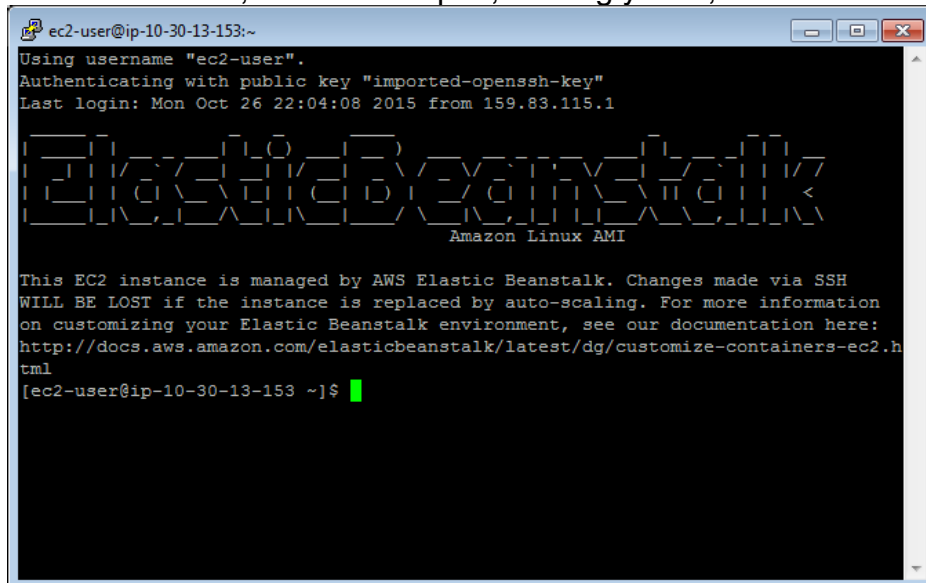
In the popup windows titled **Connect To Your Instance**, click on **Connect using PuTTY**. You will be redirected to the URL.

<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!!**