ML Model Deployment on AWS using Flask

Software Requirements

- VSCode
- Python (numpy, Flask, Scikit-learn)
- AWS Account

Goal

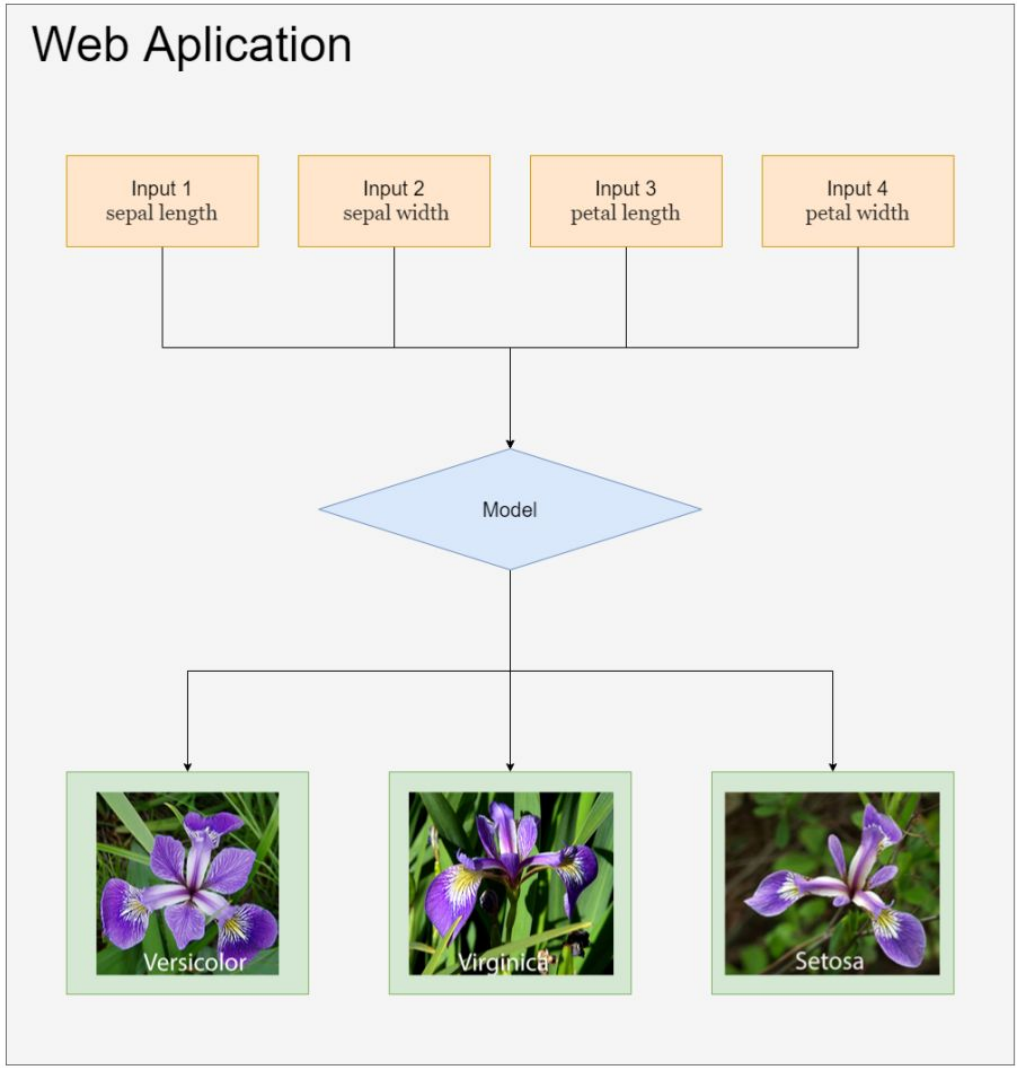
- Create the form a user can input values (features) to predict the type of flower.

Iris Versicolor

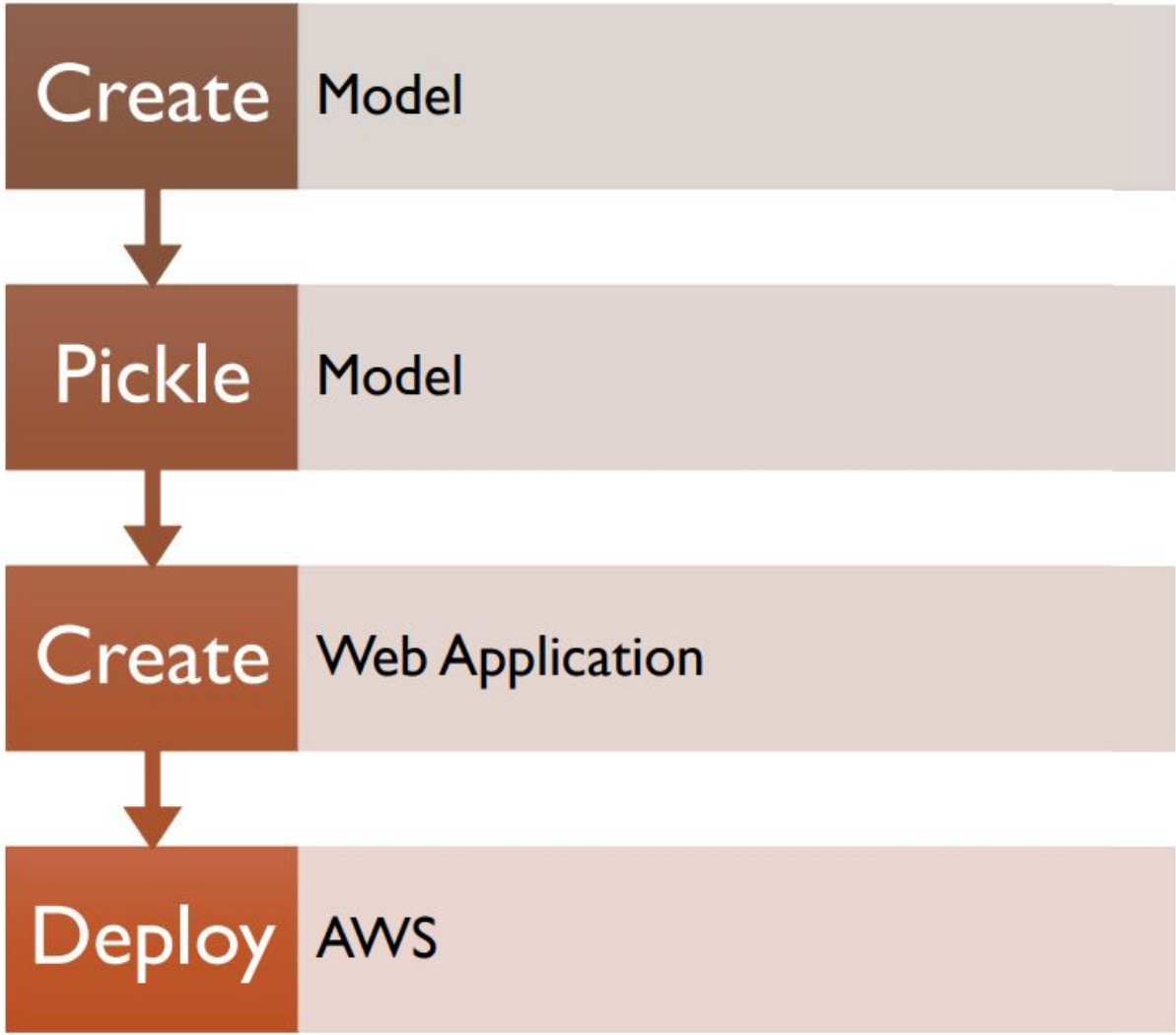
Iris Setosa

Iris Virginica

Process


Web Aplication
Input 1
sepal length
Input 2
sepal width
Input 3
petal length
Input 4
petal width
Model
Versicolor
Virginica
Setosa

Deployment steps



1. Create model

- Create model_iris.py
- Import necessary libraries

```
from sklearn.metrics import accuracy_score
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
```

- Load iris dataset

```
dataset=load_iris()
```

1. Create model (model_iris.py)

- Train machine learning model

```
# Getting Feature Names
names=dataset.feature_names

# Loading features and labels from the dataset
features=dataset.data
labels=dataset.target

# Splitting labels and features to training and testing sets
feature_train,feature_test,label_train,label_test = \
train_test_split(features,labels,test_size=0.2,random_state=42)

# Initialising Logistic Regression Model with max iterations=500
model=LogisticRegression(max_iter=500)

# Fitting Model with training features and labels
model.fit(feature_train,label_train)
```

- Test the model

```
#Predicting the labels for the testing features
label_pred=model.predict(feature_test)
#Finding the accuracy score
from sklearn.metrics import accuracy_score
accuracy_score(label_pred,label_test)
print(accuracy_score(label_pred, label_test))
```

Should get 1.0

May need to use pip to install all related libraries e.g. numpy, pandas, scikit-learn e.g.

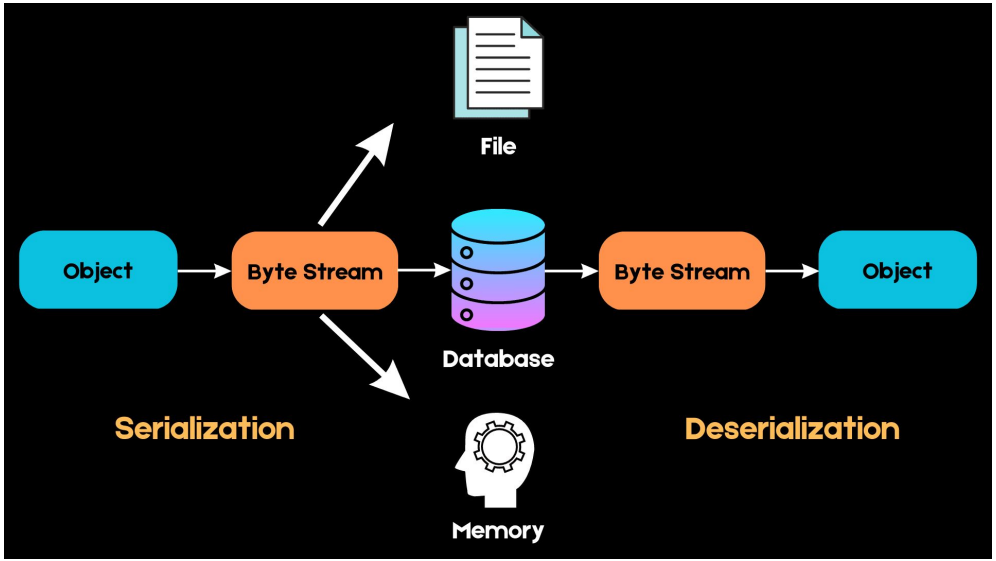
- pip install matplotlib
- pip install scikit-learn

2. Pickle model

Python Pickle

- A process of converting a Python object into a byte stream to store it in a file/database, maintain program state across sessions, or transport data over the network
- Don’t have to re-run the script to obtain the Python object e.g. dataframe, dict.
- Pickle is a useful Python tool that allows you to save your ML models
- Save Python objects in file model.pkl with Pickle Dump function

```
import pickle
pickle.dump(model,open('model.pkl','wb+'))
```

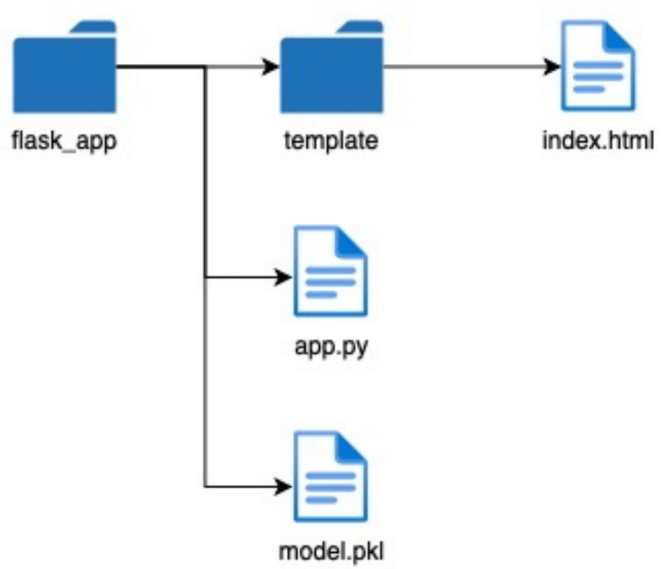


https://www.datacamp.com/tutorial/pickle-python-tutorial

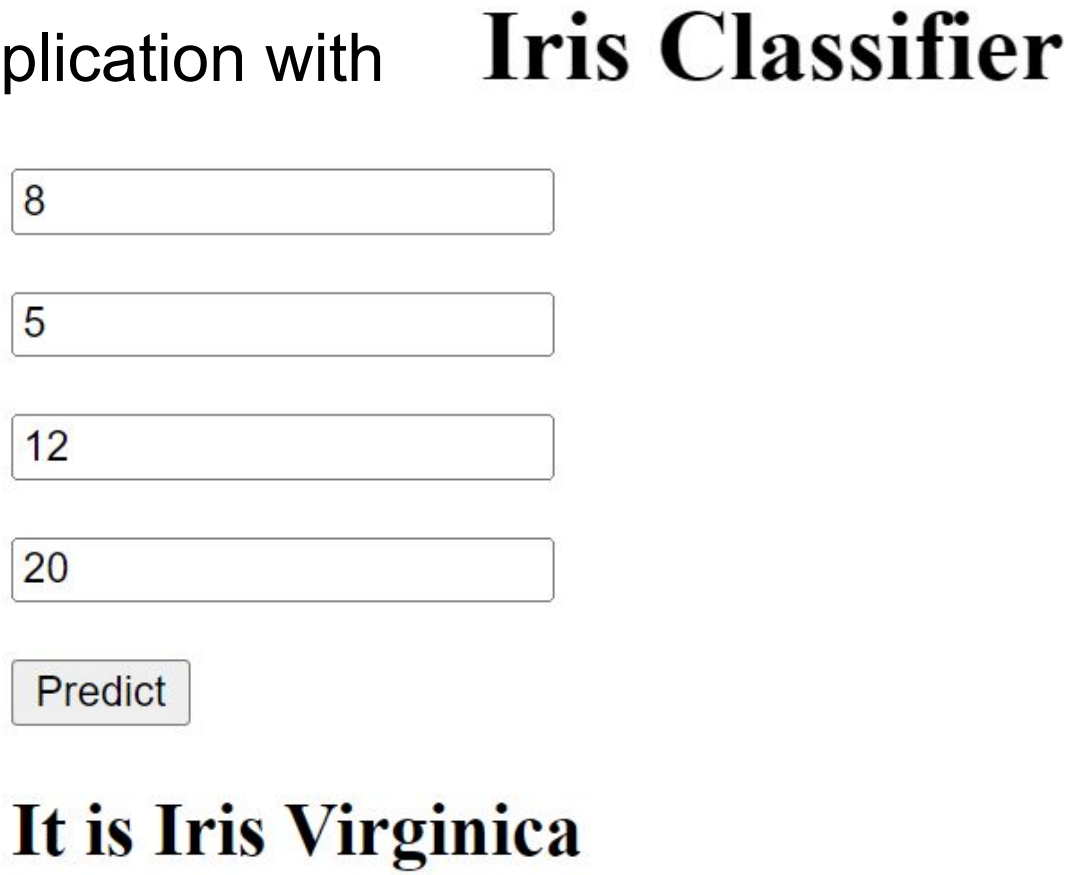
3. Create Web Application with Flask

Flask

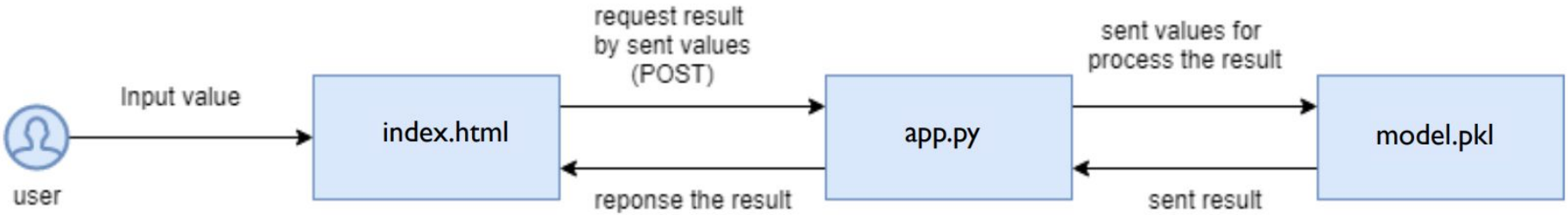
- Lightweight Web development framework using Python
- Integrated support for unit testing
- Flexible



3. Create Web Application with Flask: UI



3. Create Web Application with Flask: Workflow



HTTP POST: submits data to be processed to the identified resource. The data is included in the body of the request. This may result in the creation of a new resource or the updates of existing resources or both.

3. Create Web Application with Flask: App.py

- Import libraries

Import Flask class

```
from flask import Flask, render_template, request, send_file, send_from_directory, jsonify
import pickle
import joblib
import numpy
```

- Create an instance of flask class. __name__ is the shortcut of the name of the module, template_folder identifies the folder that contains related files.

```
app = Flask(__name__, template_folder='templates')
```

- Load model

```
model = joblib.load('model.pkl')
```

3. Create Web Application with Flask: App.py

- Route '/' for main page

```
@ app.route('/', methods=['POST', 'GET'])
def main():
    if request.method == 'GET':
        return render_template('index.html')
```

- Initial host with port 8080

```
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8080)
```

- Route '/predict' for prediction

```
@ app.route('/predict', methods=['POST', 'GET'])
def predict():
    if request.method == 'GET':
        return render_template('index.html')
    if request.method == 'POST':
        features = [float(x) for x in request.form.values()]
    print(features)
    labels = model.predict([features])
    species = labels[0]
    if species == 0:
        s = "It is Iris Setosa"
    elif species == 1:
        s = "It is Iris VersiColor"
    else:
        s = "It is Iris Virginica"
    return s
```

3. Create Web Application with Flask: index.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>Iris Classifier</title>
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>

    </head>
    <body>

        <h1><CENTER>Iris Classifier</CENTER></h1>
        <form id="myform" method="POST">
            <input type="text" name="sl" placeholder="Enter Sepal Length in cm"><br><br>
            <input type="text" name="sw" placeholder="Enter Sepal Width in cm"><br><br>
            <input type="text" name="pl" placeholder="Enter Petal Length in cm"><br><br>
            <input type="text" name="pw" placeholder="Enter Petal Width in cm"><br><br>

        </form>
        <button id="predict">Predict</button>
        <h2 id="result"></h2>


    </body>
</html>
```

Import ajax

Input form, button and results

3. Create Web Application with Flask: index.html

Additional ajax part

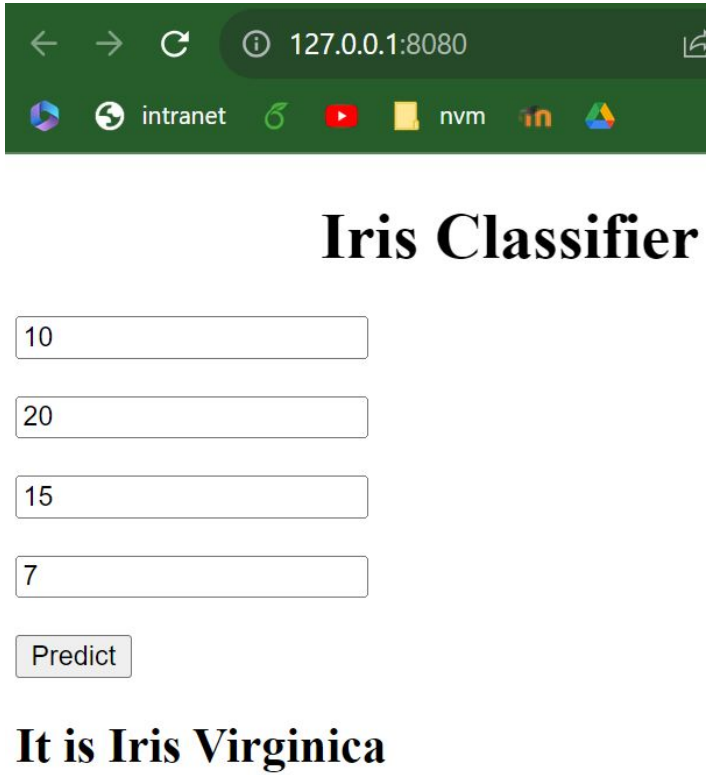
```
<script type="text/javascript">
$(function() {
 $('#predict').click(function() {
     event.preventDefault();
     var form_data = new FormData($('#myform')[0]);
     console.log(form_data);
     $.ajax({
         type: 'POST',
         url: '/predict',
         data: form_data,
         contentType: false,
         processData: false,
     })
     .done(function(data, textStatus, jqXHR){$('#result').text(data)})
     .fail(function(data){alert('error!')
    })
 })
})
</script>
```

3. Create Web Application with Flask: test run app.py

- You can test run app by type command in terminal in flask_app folder **python3 app.py**

```
PS C:\Users\USER\Desktop\iris> python3 app.py
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:8080
 * Running on http://192.168.1.113:8080
Press CTRL+C to quit
127.0.0.1 - - [27/Oct/2023 15:46:05] "GET / HTTP/1.1" 200 -
[10.0, 20.0, 15.0, 7.0]
127.0.0.1 - - [27/Oct/2023 15:46:31] "POST /predict HTTP/1.1" 200 -
```

To install flask, type ‘pip install flask’



4. Deploy to AWS EC2

Using Amazon EC2 IaaS

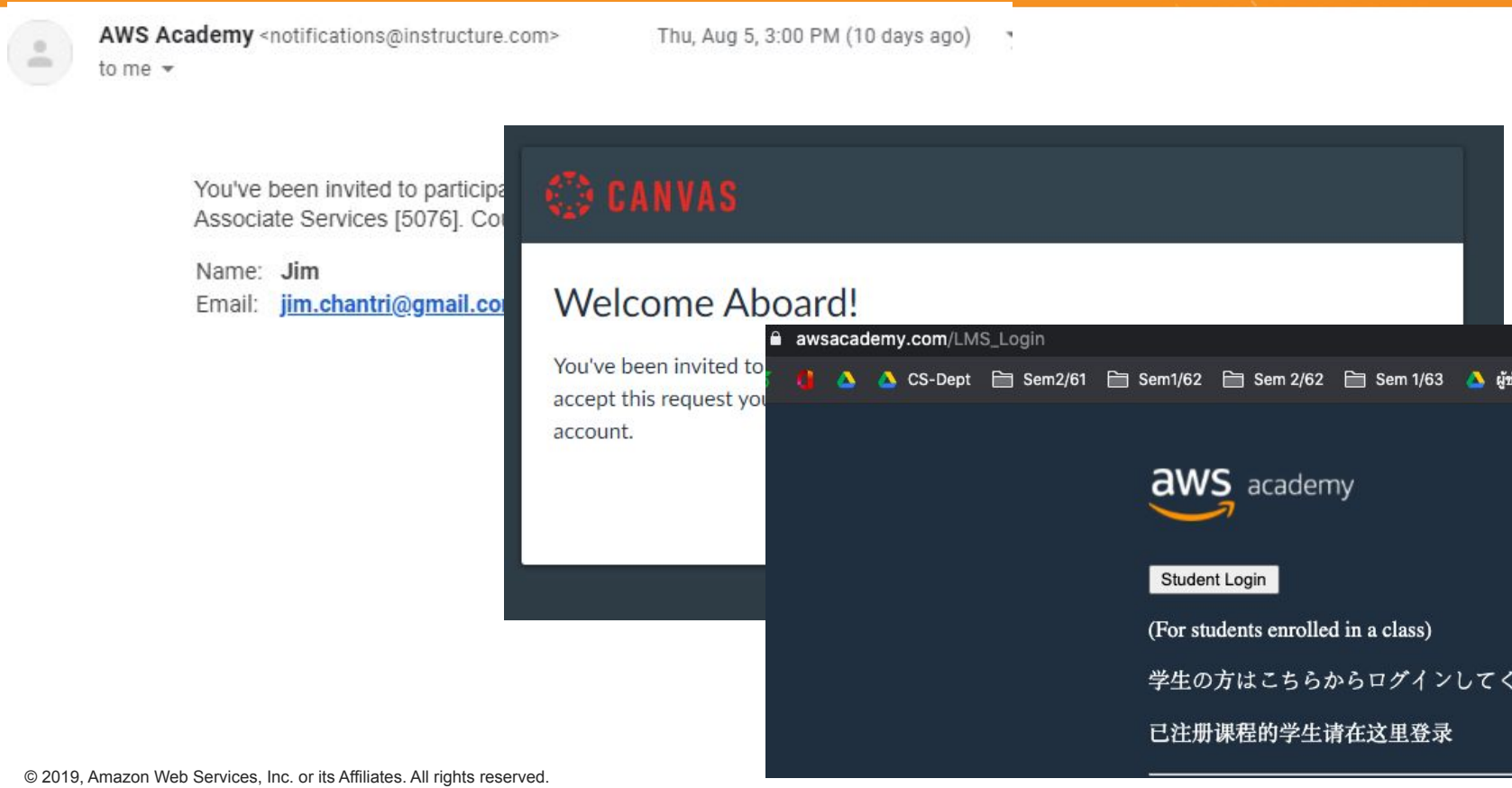
- Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable compute capacity in the cloud.
- It is designed to make web-scale cloud computing easier for developers.
- Benefits
  - simple web service interface allows you to obtain and configure capaci minimal friction
  - reduce the time required to obtain and boot new server instances to m
  - quickly scale capacity, both up and down, as your computing requirem change
  - Pay for what you use

Getting Started with Amazon EC2

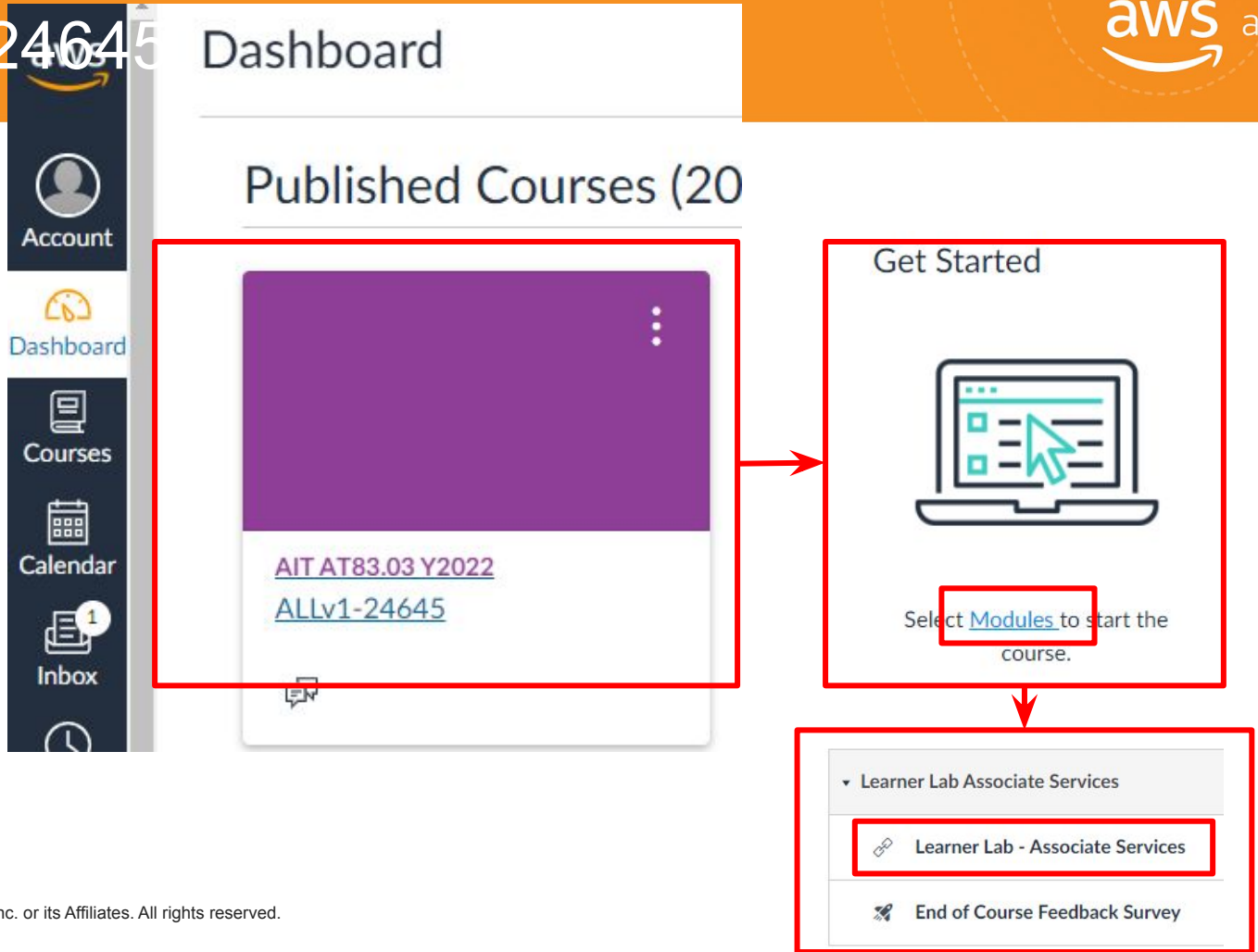
Getting Started with the AWS Management Console

1. Set up and log into your AWS account
2. Launch an Amazon EC2 instance
3. Configure your instance e.g. Security Group, elastic IP address
4. Connect to your instance
5. Install Flask, Numpy, Scikit-learn libraries
6. Run Flask
7. Once everything is done, terminate Instances: EC2 is free to start but you should terminate your instances to prevent additional charges.

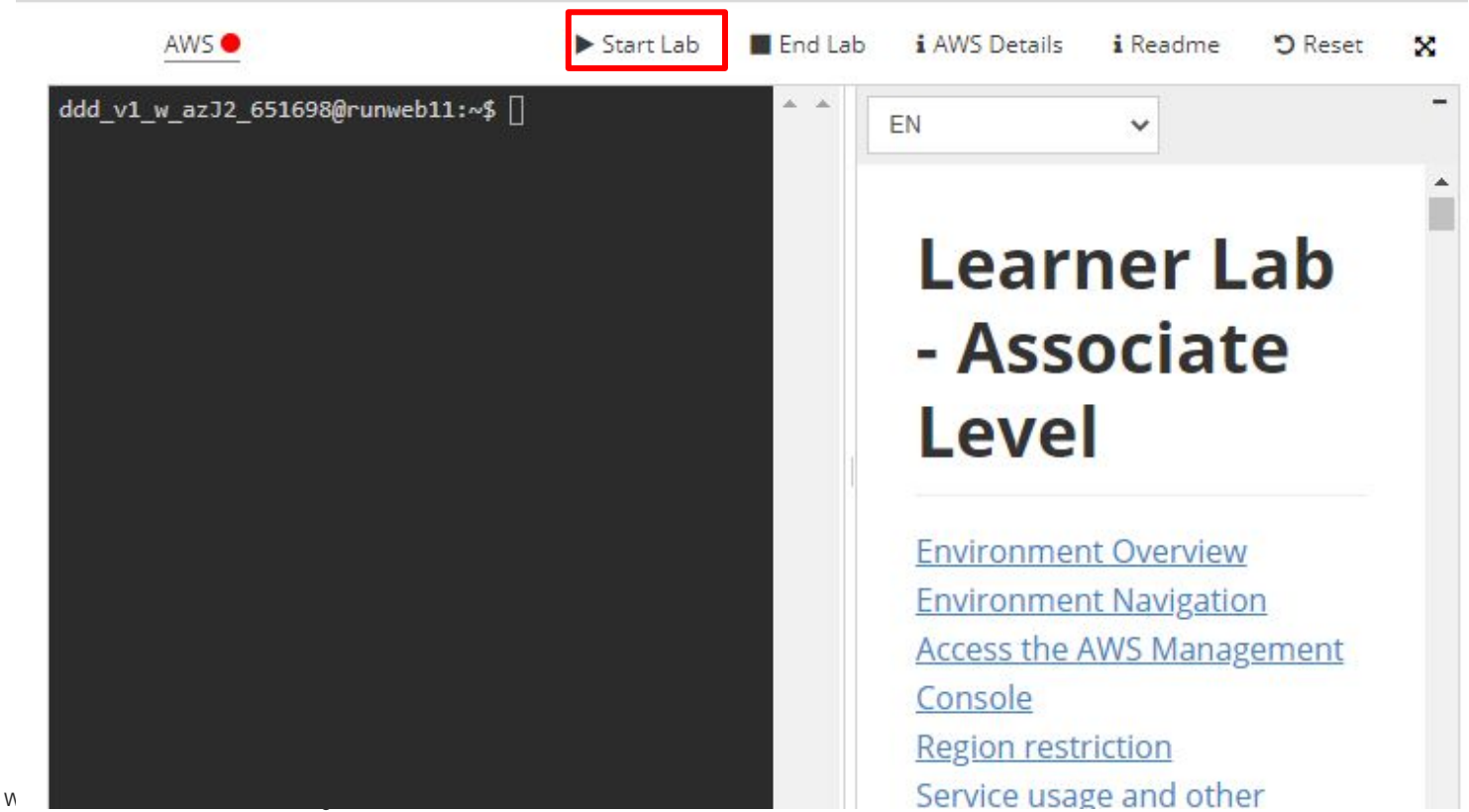
AWS Academy Learner Lab



ALLAv1-24645



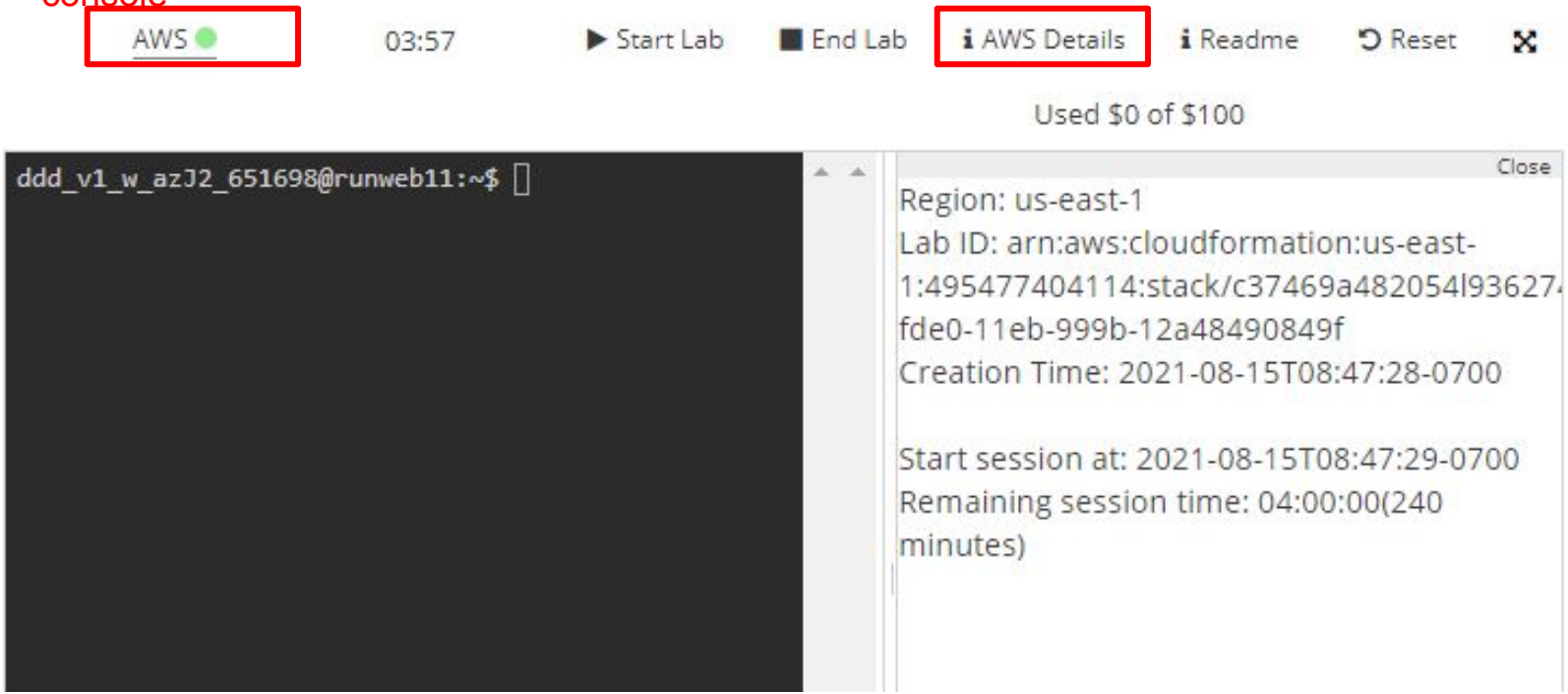
Learner Lab - Associate Services



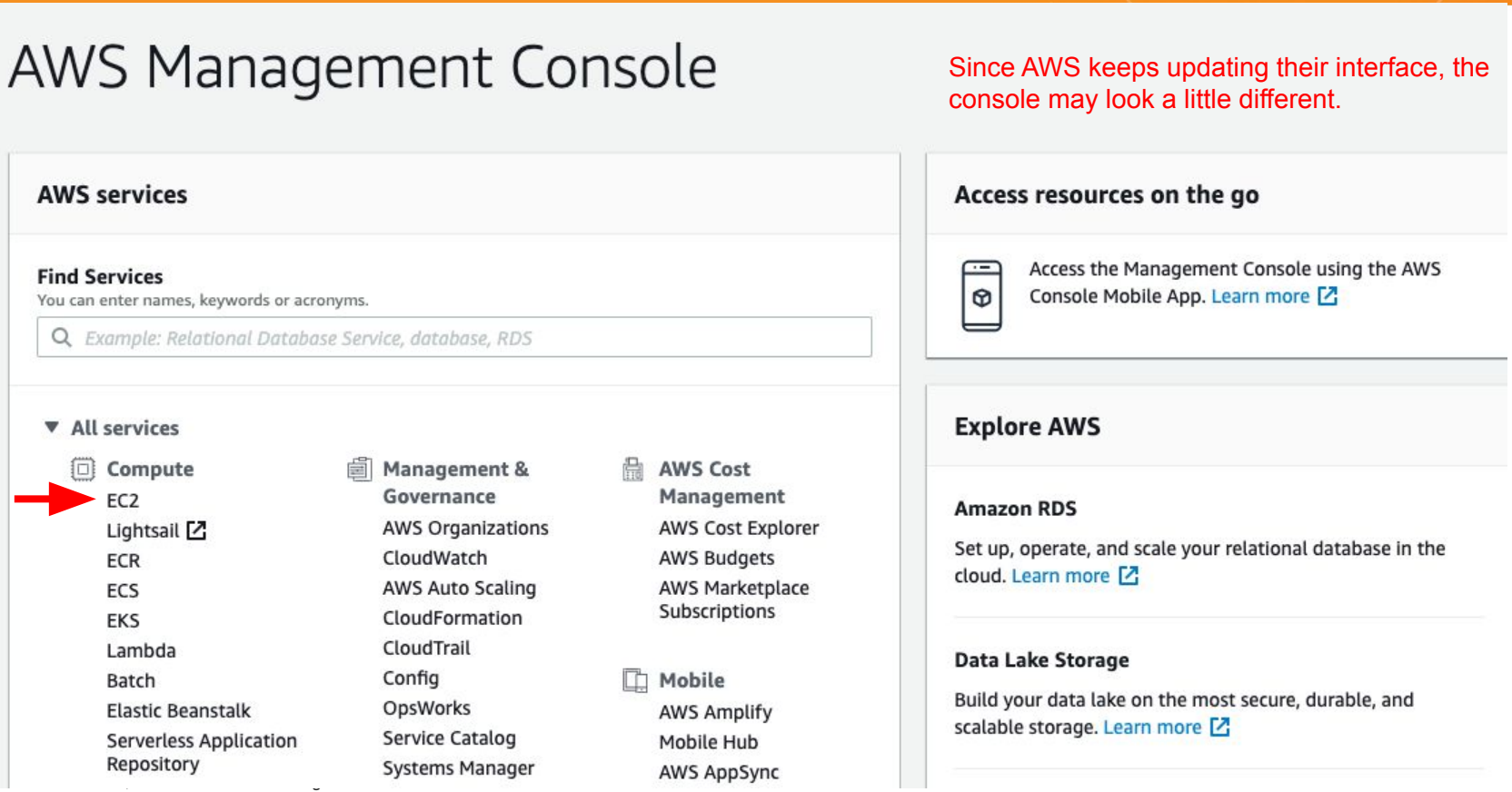
Learner Lab - Associate Services

Go to AWS Management console

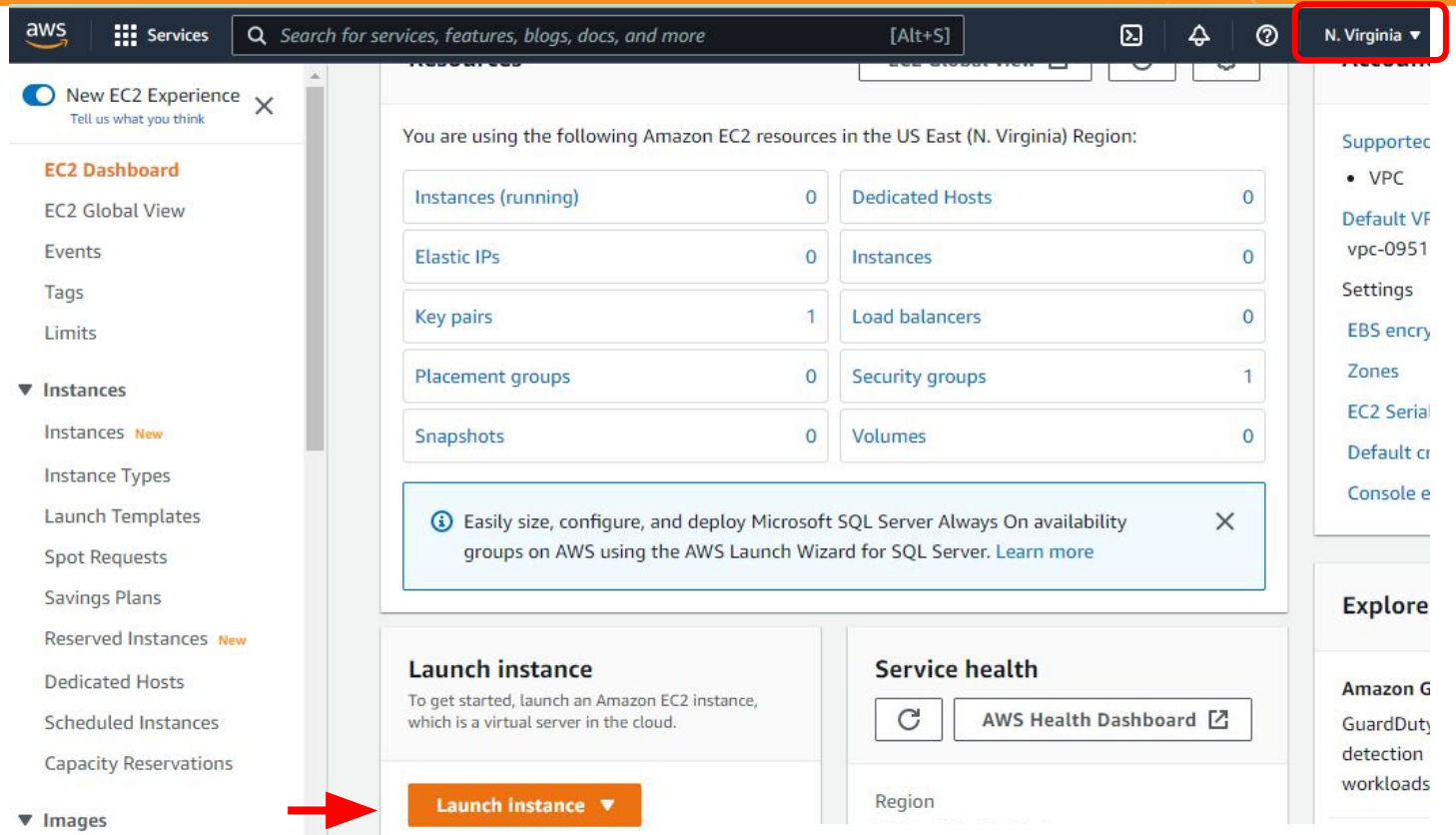
Detail of your cloud usage



AWS Management Console

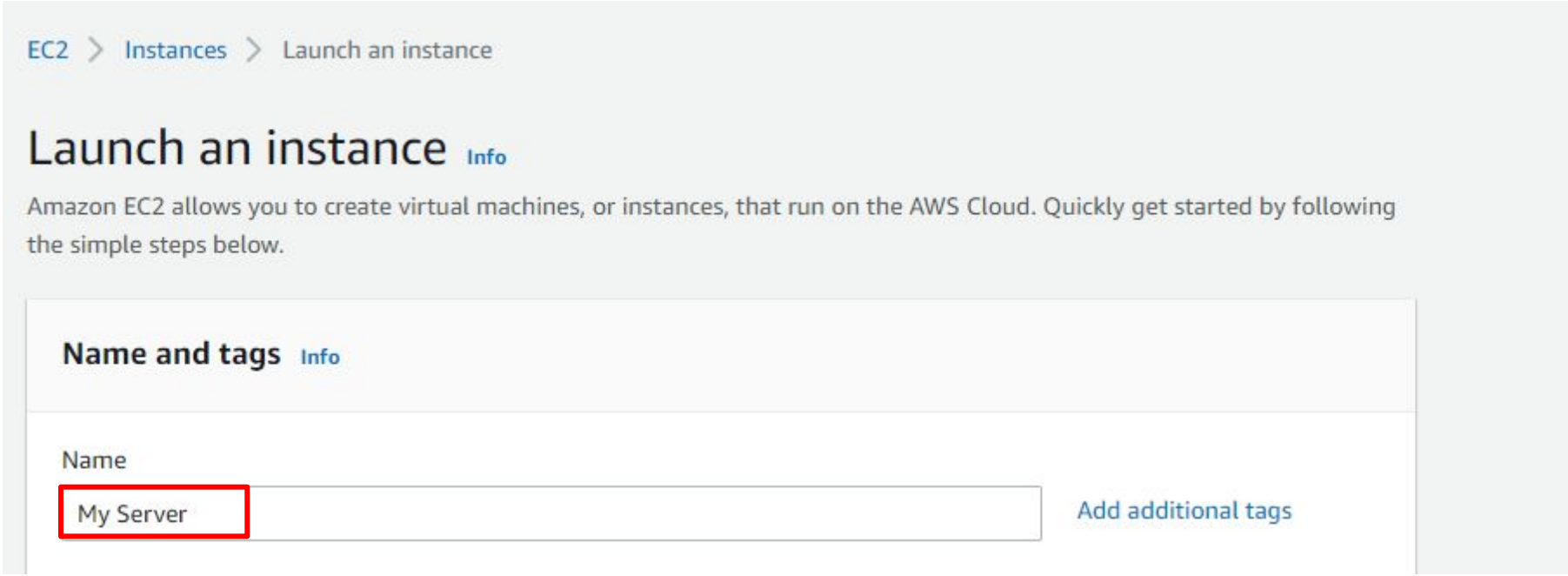


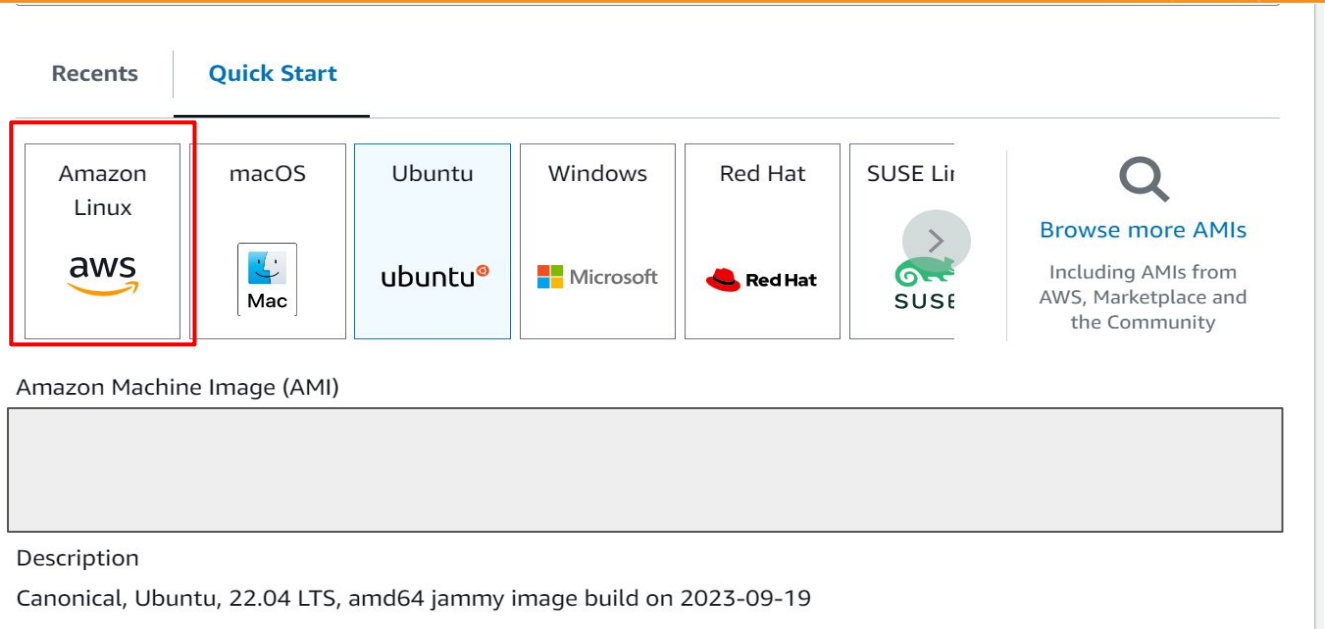
EC2 Dashboard: Launch Instance



An instance is a virtual server in the cloud

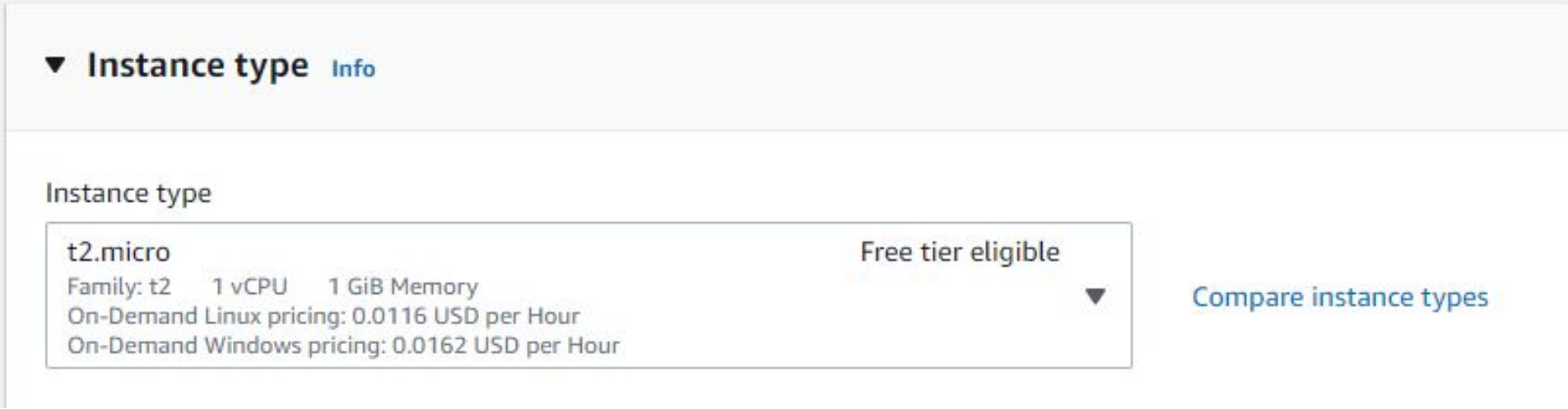
Launch an EC2 instance





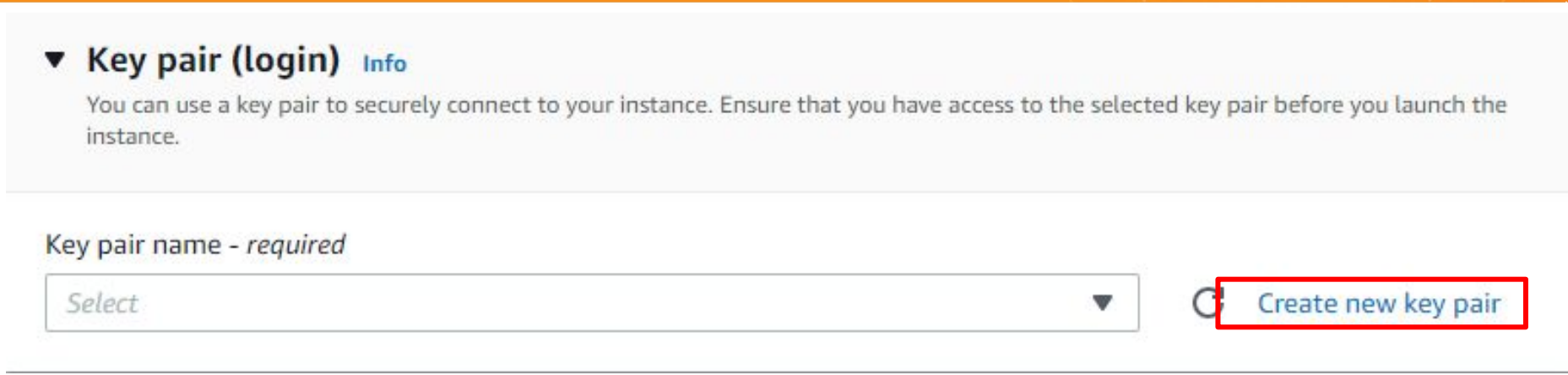
An Amazon Machine Image (AMI) is a template that contains a software configuration (for example, an operating system, an application server, and applications. It serves as the basic unit of deployment for services delivered using EC2.

Configure Instance: Choose Resource Size (CPU, Memory)

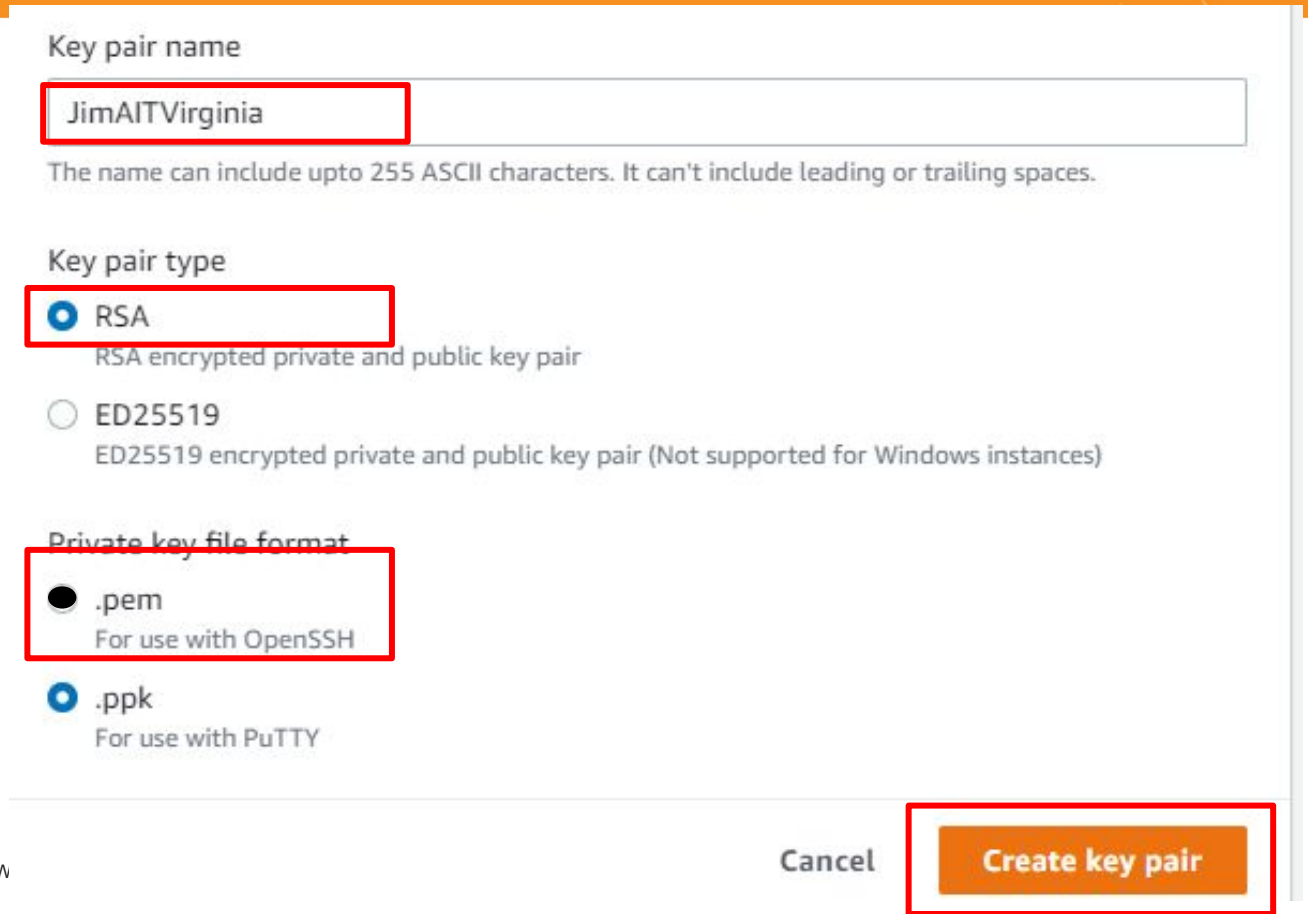


Choose t2.micro

Key pair (for login)



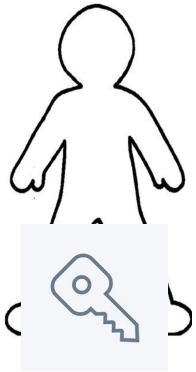
Key pair (for login)



- Don’t lose it
- Specific to a region
- Save .pem file

Key pair

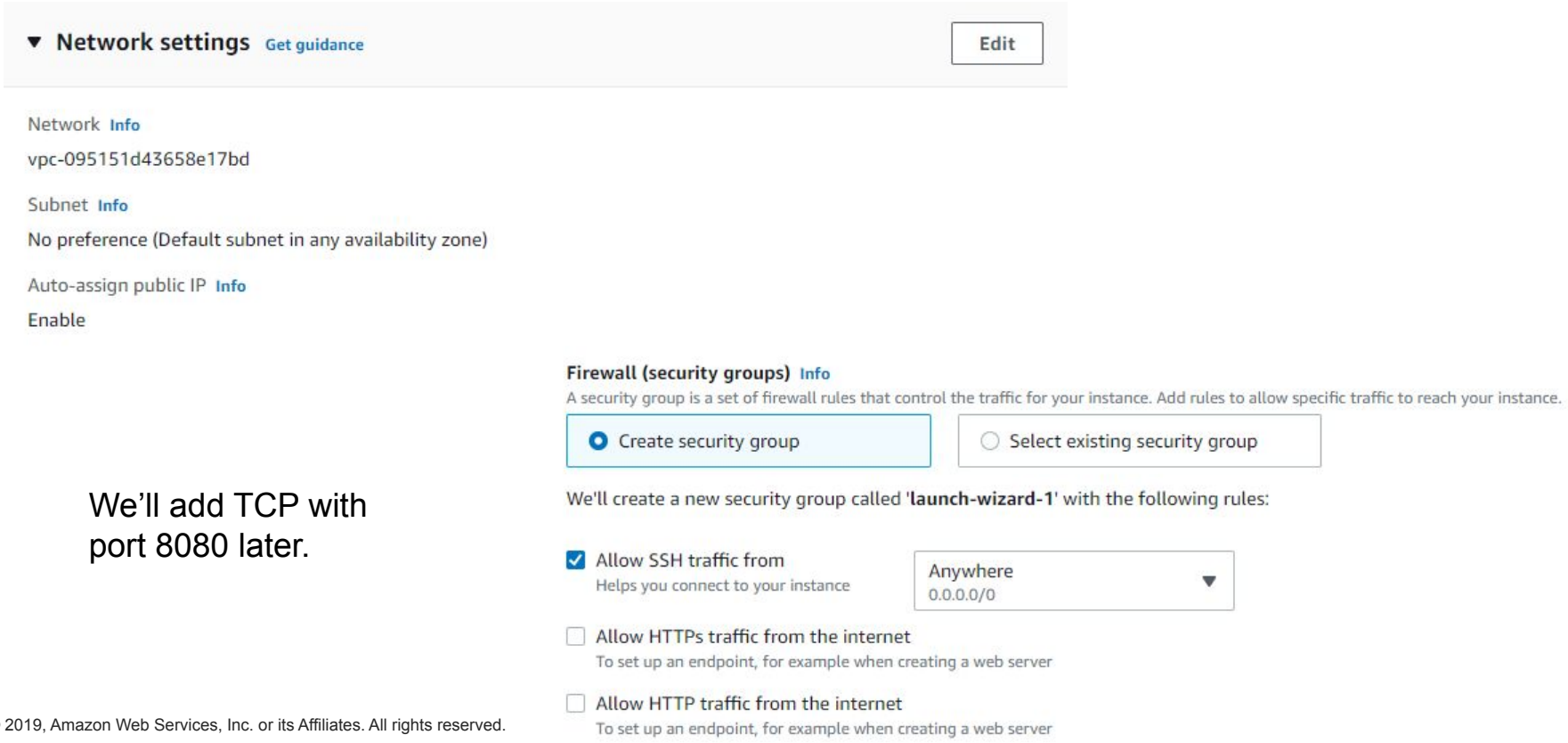
- Stored at EC2
- Public



- Stored at client
- Keep secret

2048-bit SSH-2 RSA

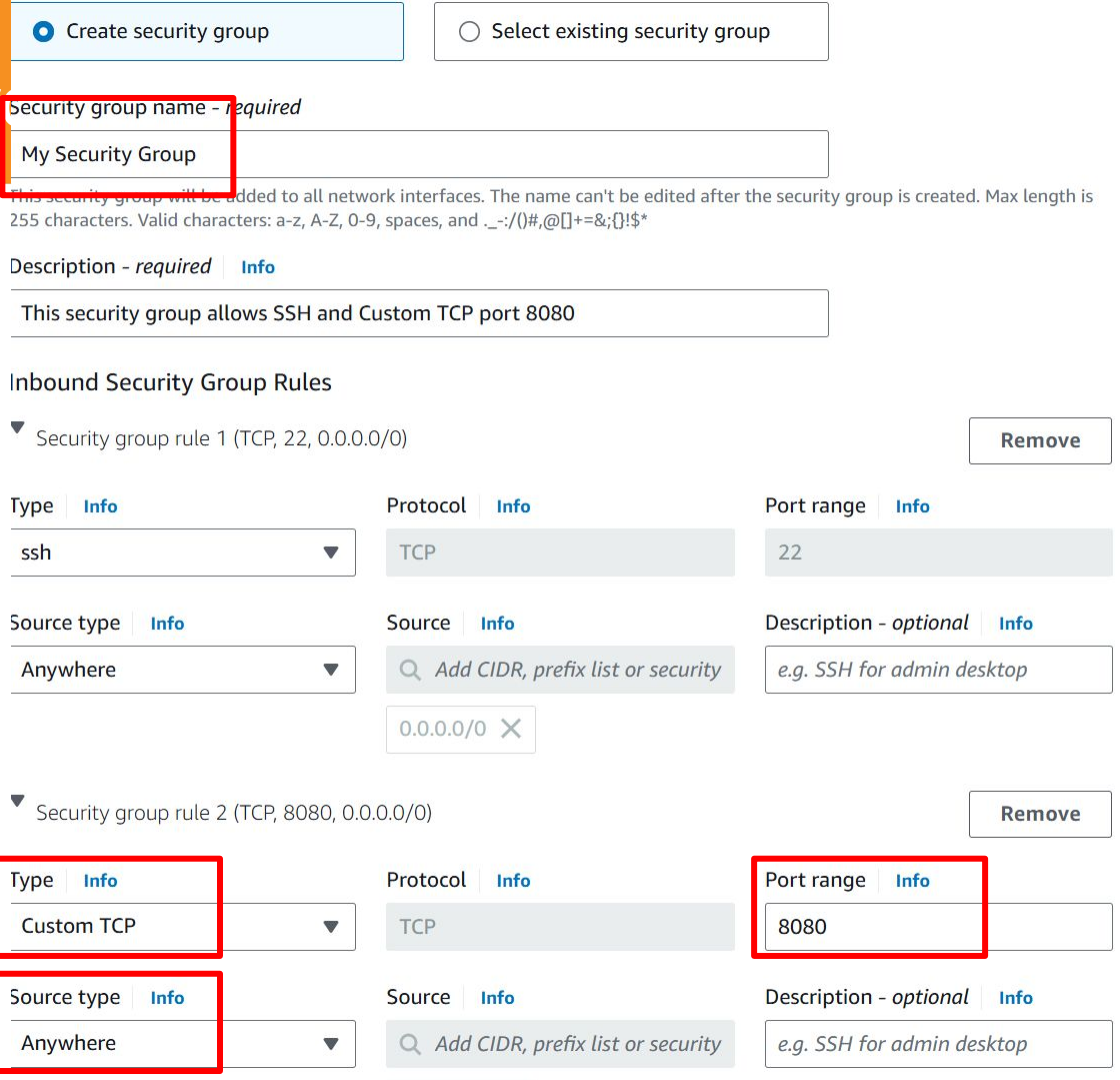
Network settings and Security Group



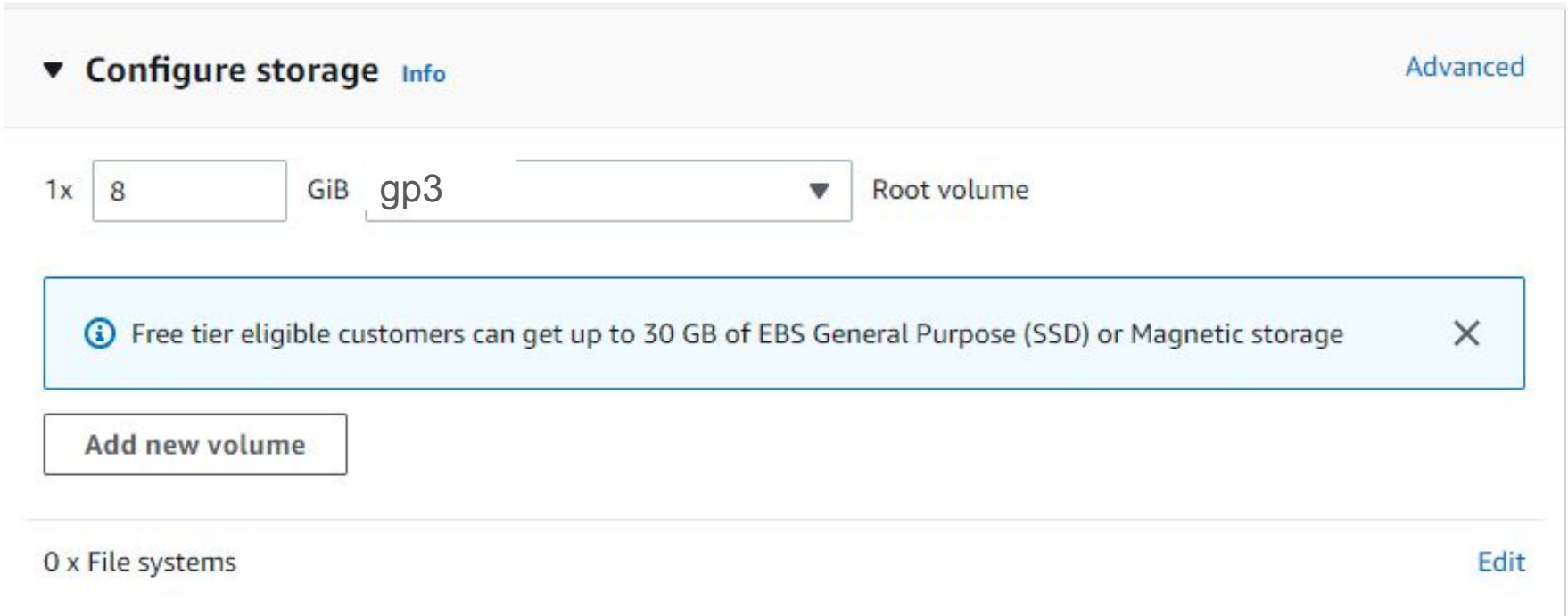
We'll add TCP with port 8080 later.

Configure security gr

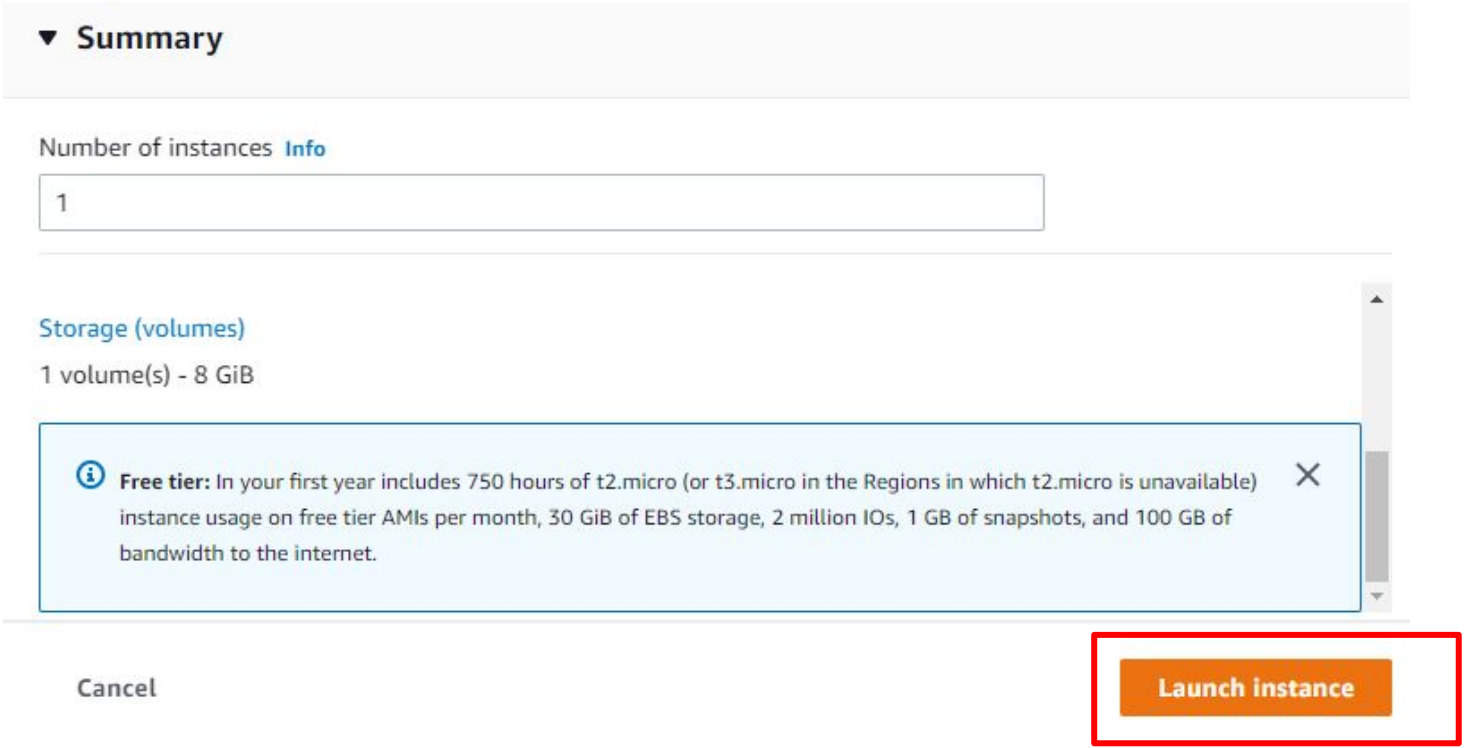
- Allow TCP port 8080 for inbound rule



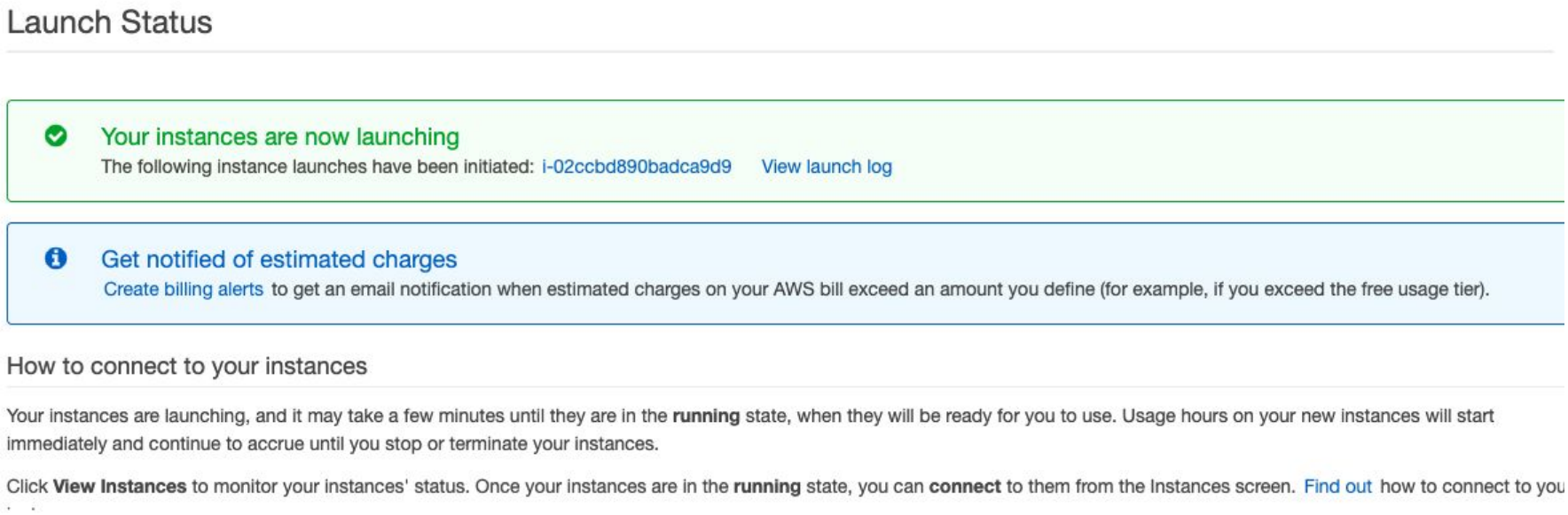
Configure storage



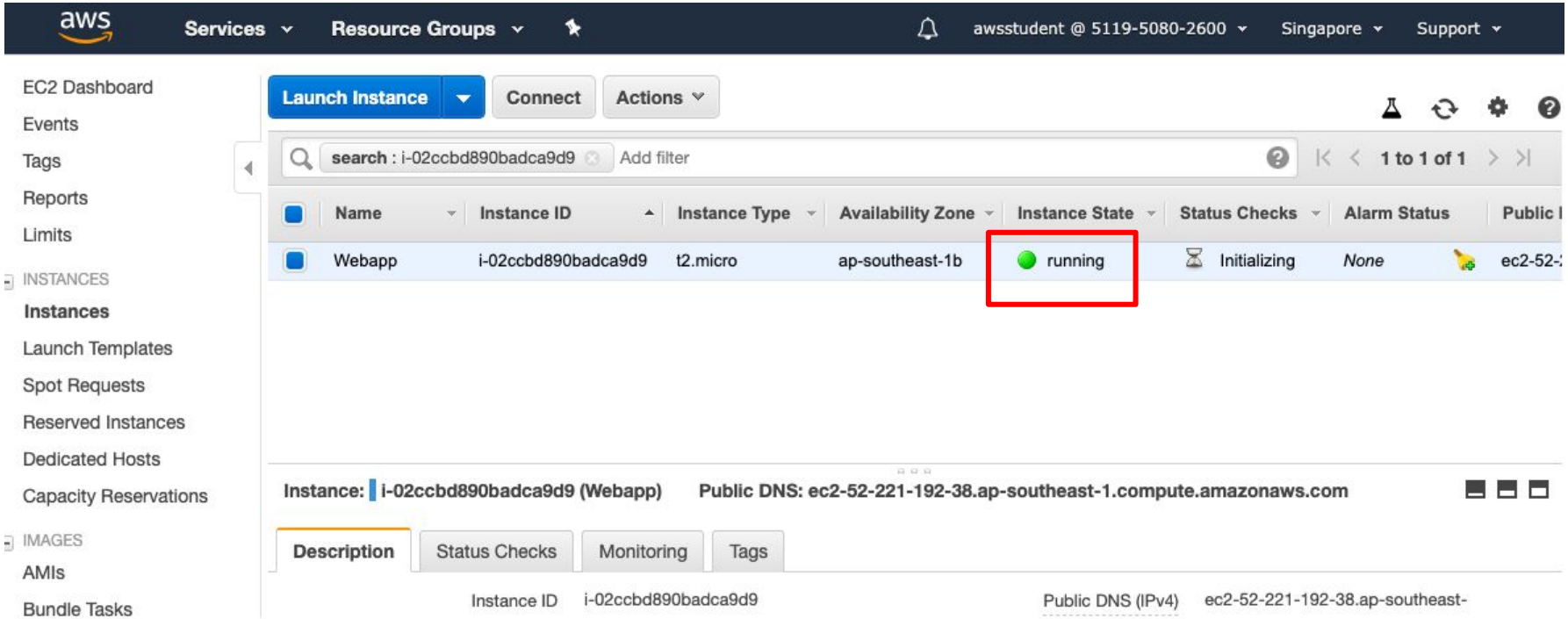
Summary



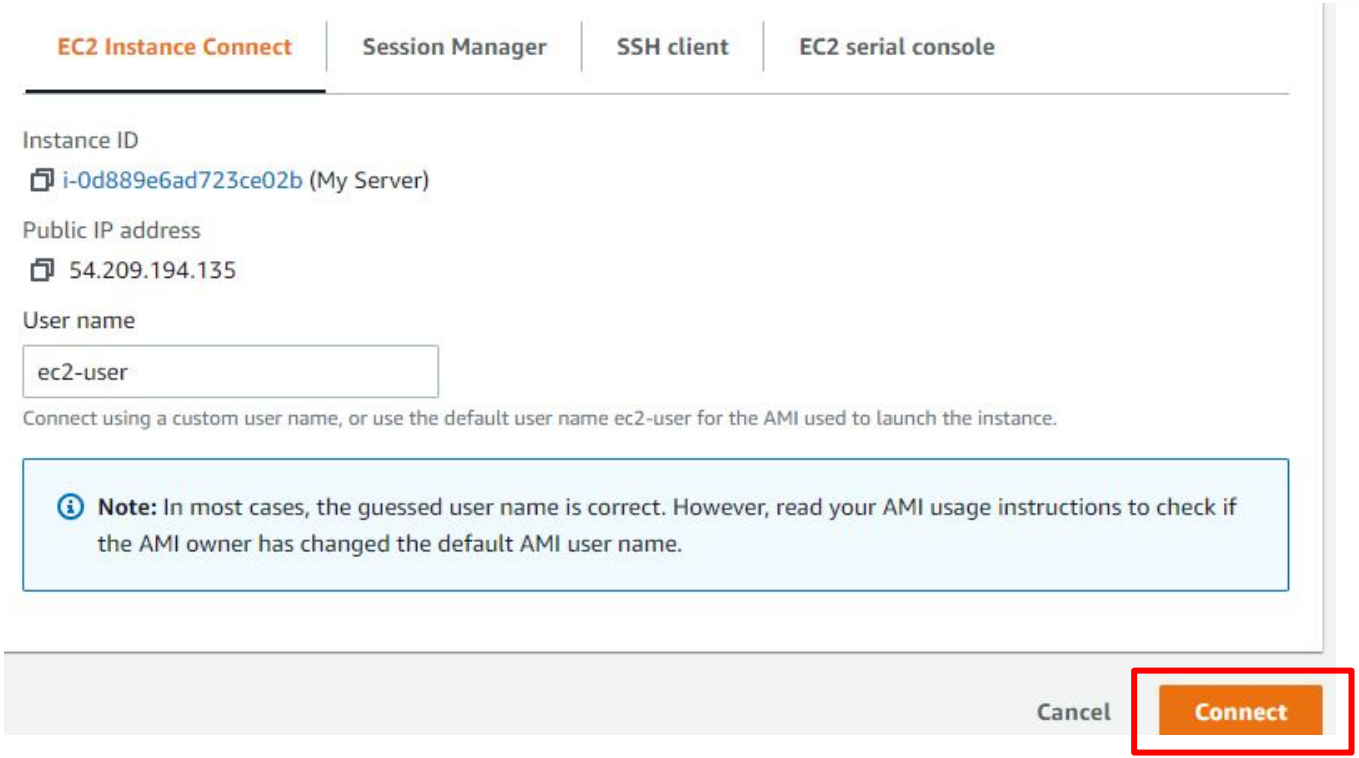
How long does it take for AWS to provision your instance?



The instance is ready

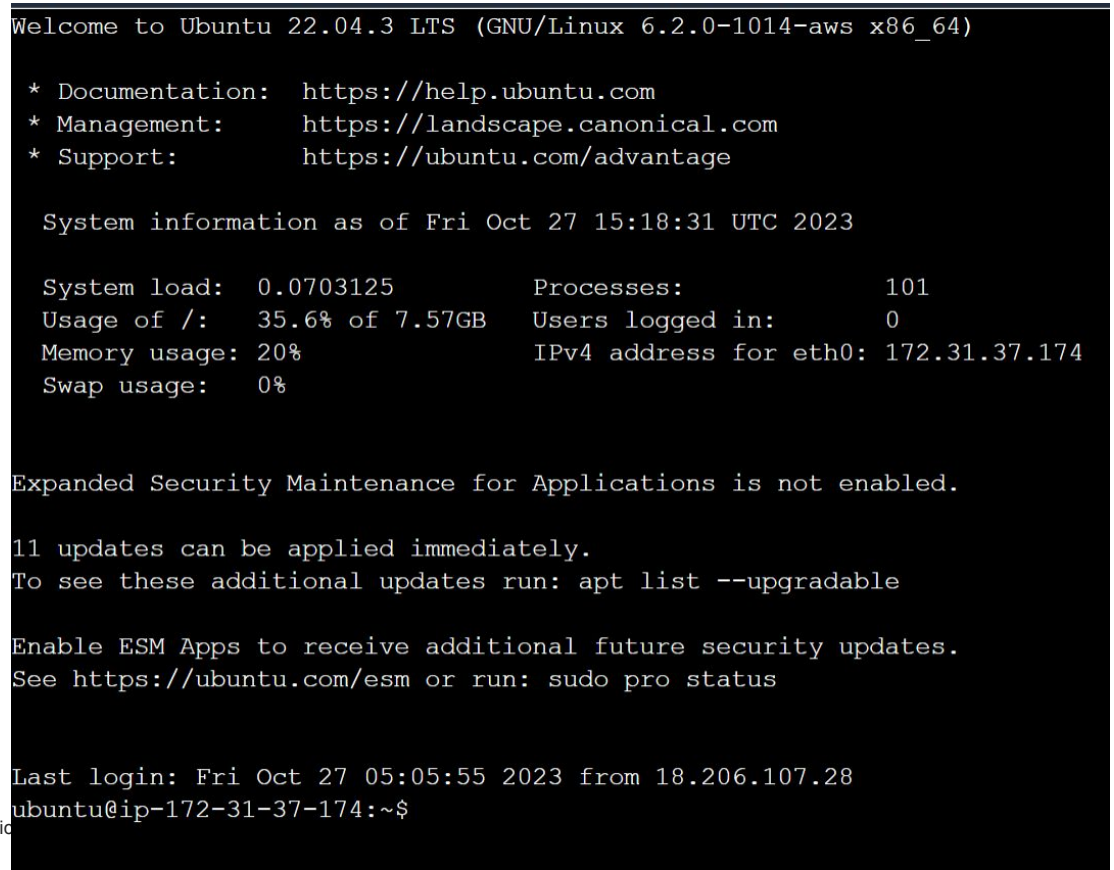


Connect to your instance



Once your instance is running, you can login using SSH and a key pair

Ubuntu terminal



EC2 Instance Connect | Session Manager | SSH client | EC2 serial console

Instance ID

i-0240328d714b14bcf (iris2)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is JimAITVirginia.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
   chmod 400 "JimAITVirginia.pem"
4. Connect to your instance using its Public DNS:
   ec2-52-87-195-140.compute-1.amazonaws.com

Example:

ssh -i "JimAITVirginia.pem" ec2-user@ec2-52-87-195-140.compute-1.amazonaws.com

Install necessary libraries

From Ubuntu, update and install related libraries

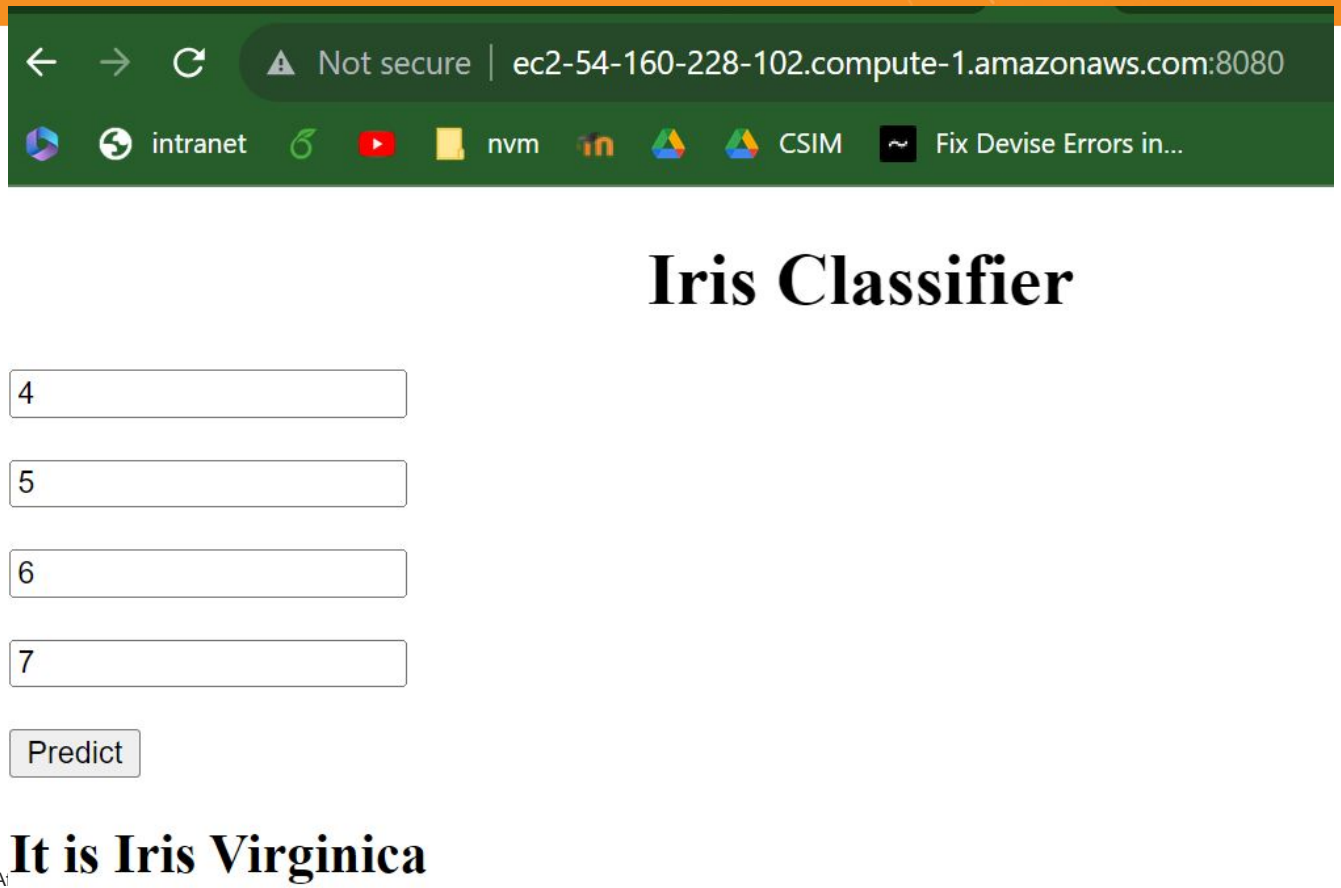
- sudo su
- sudo update -y
- yum install python3-pip
- pip3 install numpy flask scikit-learn

From your computer terminal, copy related files to Ubuntu

- scp -r -i .\JimAITVirginia.pem .\iris_project\ ec2-user@ec2-52-87-195-140.compute-1.amazonaws.com:~/.

From Amazon Linux, go to iris_project folder and run app.py

- python3 app.py
- From your web browser, type public DNS followed by 8080, e.g.
- http://ec2-52-87-195-140.compute-1.amazonaws.com:8080/



Addition tips

- To fix the IP address, use elastic IP address
- Use nohup to run app.py

Reference

- https://medium.com/shapeai/deploying-flask-application-with-ml-models-on-aws-ec2-instance-3b9a1cec5e13
- https://aws.amazon.com/pm/ec2/
- https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/elastic-ip-addresses-eip.html
- https://flask.palletsprojects.com/en/3.0.x/
- https://www.kaggle.com/datasets/uciml/iris