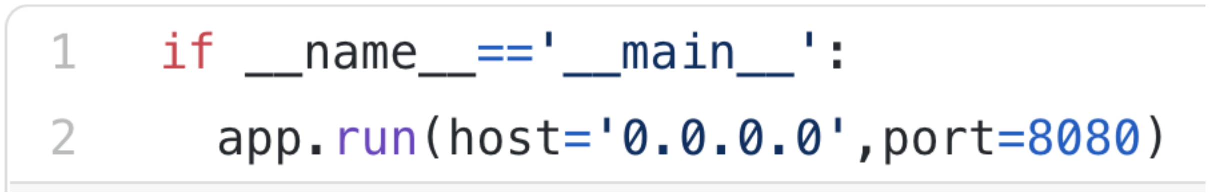
Deploy ML models in Production using AWS

Whenever we need to deploy our application on AWS, we need to change the host and port in which the following flask application is running. By default, the host address will be at '127.0.0.1' in other words localhost and port number by default is 5000. Now we need to change the default port number to 8080 and host address to '0.0.0.0'. For this, the following code needs to be added in place of app.run().



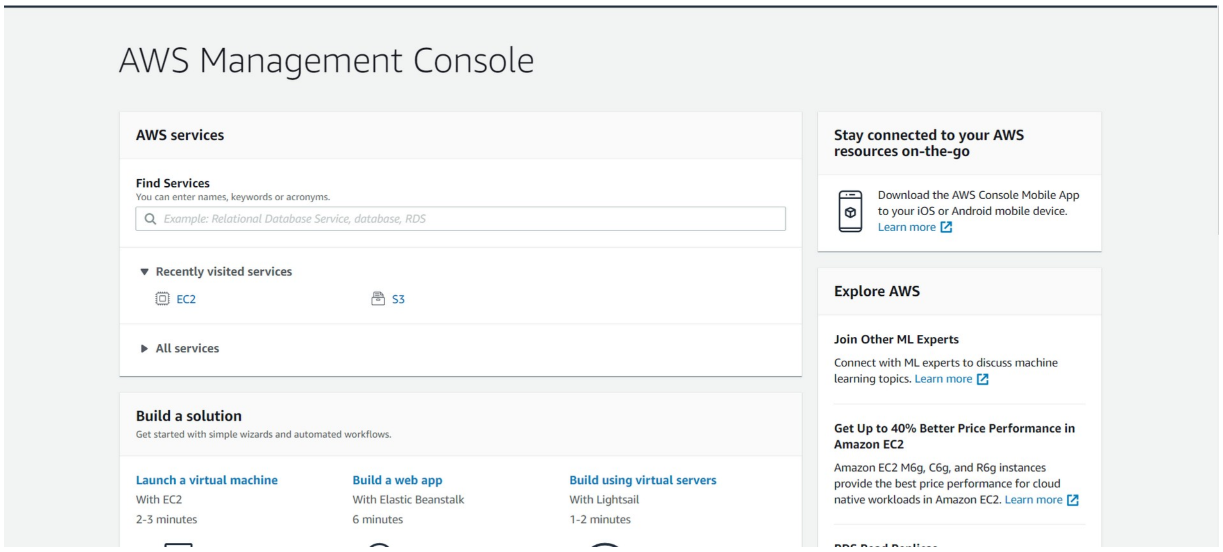
```
if __name__=='__main__':
    app.run(host='0.0.0.0',port=8080)
```

Now if you run the flask application it will run but if you go to http://0.0.0.0:8080/ it will give you the error. The reason for this is address 0.0. 0.0 is a non-routable meta-address used to designate an invalid, unknown, or non-applicable target. Don't care about this error just leave about this and go on to the next step.

Getting Started with AWS:-

Create free student account at: https://aws.amazon.com/cn/education/awseducate/

Now it's all set, the remaining thing to reach our goal is to deploy flask application to AWS Cloud. For this, we need to have an AWS account. If you are new to AWS, You need to create an AWS account by providing your credit card details to avail of free tier services free for one year. After creating the account you need to go to AWS Management Console and need to sign in to AWS by using the account that you have just created. Once you sign in the webpage looks similar to the below image.



AWS Management Console

AWS services

Find Services
You can enter names, keywords or acronyms.
Example: Relational Database Service, database, RDS

Recently visited services
EC2 S3

All services

Build a solution
Get started with simple wizards and automated workflows.

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

Stay connected to your AWS resources on-the-go

Download the AWS Console Mobile App to your iOS or Android mobile device. Learn more

Explore AWS

Join Other ML Experts
Connect with ML experts to discuss machine learning topics. Learn more

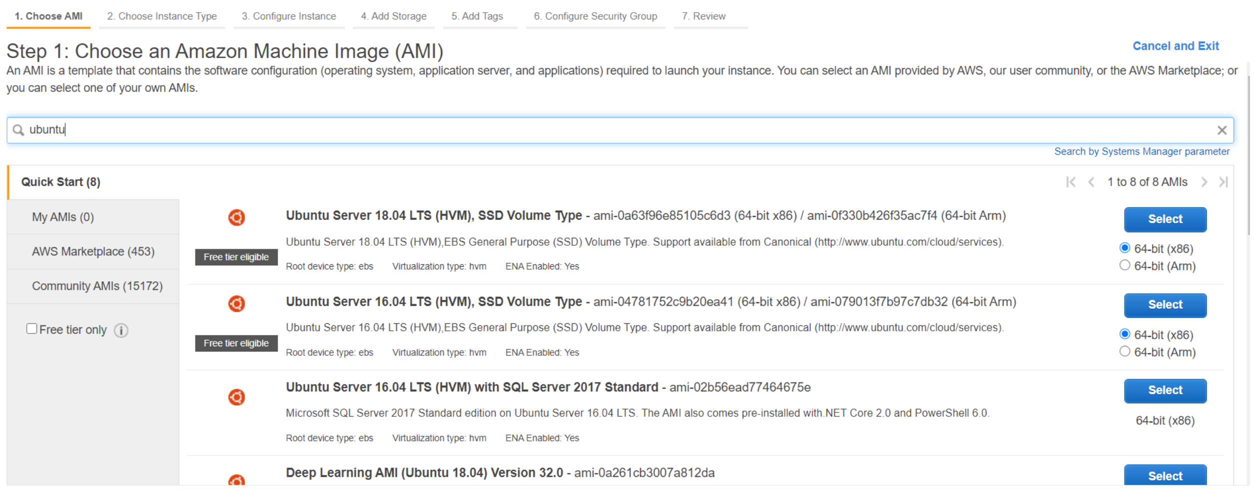
Get Up to 40% Better Price Performance in Amazon EC2
Amazon EC2 M6g, C6g, and R6g instances provide the best price performance for cloud native workloads in Amazon EC2. Learn more

Creating an AWS EC2 Instance:

Now we need to go to the search bar in the AWS Management Console and search for EC2 and need to click on running instances. Click on Launch Instance to create the new EC2 instance. Now we need to follow 4 steps to create an instance.

I. Choose an Amazon Machine Image (AMI):-

We need to choose AMI which means choosing an OS for the server. In this case, we can choose the UBUNTU 18.04 which is a free tier and proceed next.



Step 1: Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

ubuntu

Ubuntu Server 18.04 LTS (HVM), SSD Volume Type - ami-0a63f96e85105c6d3 (64-bit x86) / ami-0f330b426f35ac7f4 (64-bit Arm)
Ubuntu Server 18.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).
Root device type: ebs Virtualization type: hvm ENA Enabled: Yes
[✓ 64-bit (x86)] 64-bit (Arm)

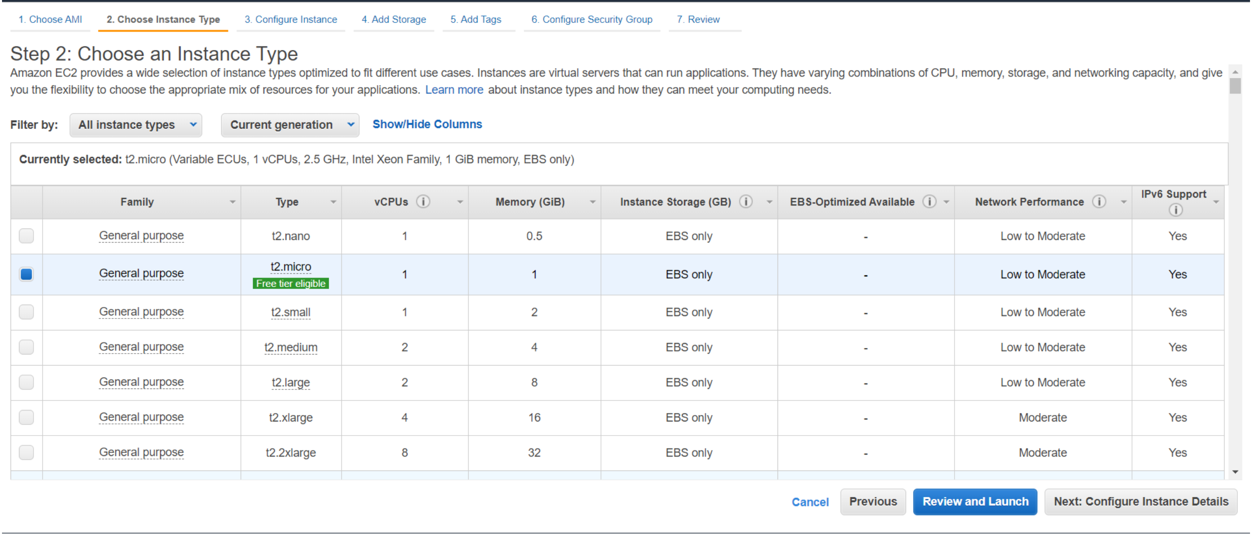
Ubuntu Server 16.04 LTS (HVM), SSD Volume Type - ami-04781752c9b20ea41 (64-bit x86) / ami-079013f7b97c7db32 (64-bit Arm)
Ubuntu Server 16.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).
Root device type: ebs Virtualization type: hvm ENA Enabled: Yes
[✓ 64-bit (x86)] 64-bit (Arm)

Ubuntu Server 16.04 LTS (HVM) with SQL Server 2017 Standard - ami-02b56ead77464675e
Microsoft SQL Server 2017 Standard edition on Ubuntu Server 16.04 LTS. The AMI also comes pre-installed with.NET Core 2.0 and PowerShell 6.0.
Root device type: ebs Virtualization type: hvm ENA Enabled: Yes
64-bit (x86)

Deep Learning AMI (Ubuntu 18.04) Version 32.0 - ami-0a261cb3007a812da

II. Choose Instance Type:-

Now we need to choose instance type for our server. In this case, we can go with a free tier instance. If yours is a company you can choose the paid ones. Here is the snapshot of the choose instance type.



Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. Learn more about instance types and how they can meet your computing needs.

Currently selected: t2.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
☐	General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
☑	General purpose	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate	Yes
☐	General purpose	t2.small	1	2	EBS only	-	Low to Moderate	Yes
☐	General purpose	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
☐	General purpose	t2.large	2	8	EBS only	-	Low to Moderate	Yes
☐	General purpose	t2.xlarge	4	16	EBS only	-	Moderate	Yes
☐	General purpose	t2.2xlarge	8	32	EBS only	-	Moderate	Yes

Once you selected the instance we need to click on “Review and Launch”.

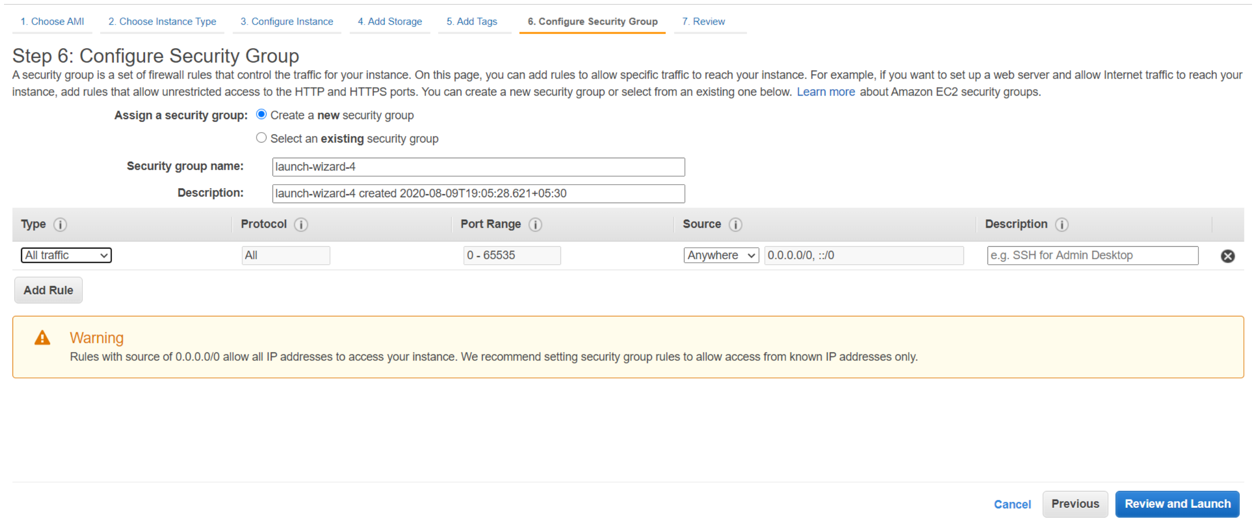
III. Edit Security Groups:-

Once the above steps are done, you need to click on the "Edit Security Groups" in the "Review and Launch" page. Once if you go "Edit Security Groups" Page you need to create a security group.

What is the reason behind creating Security Groups?

It helps us to edit which kind of traffic that should be allowed while accessing the instance and it helps us to decide which IPs can access our application which allows us to restrict the usage.

In this case, we are allowing all traffic and source to anywhere so that every network from anywhere in the world can be able to access our application. Once it is done we need to click on "Review and Launch". Here is the Snapshot of editing security groups.



Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. Learn more about Amazon EC2 security groups.

Assign a security group: [✓ Create a **new** security group] Select an **existing** security group

Security group name: launch-wizard-4

Description: launch-wizard-4 created 2020-08-09T19:05:28.621+05:30

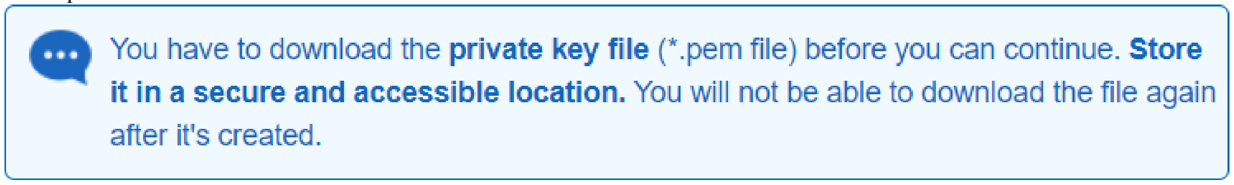
Type	Protocol	Port Range	Source	Description
All traffic	All	0 - 65535	Anywhere 0.0.0.0/0, ::/0	e.g. SSH for Admin Desktop

Warning
Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

IV. Creating a Private Key Pair:-

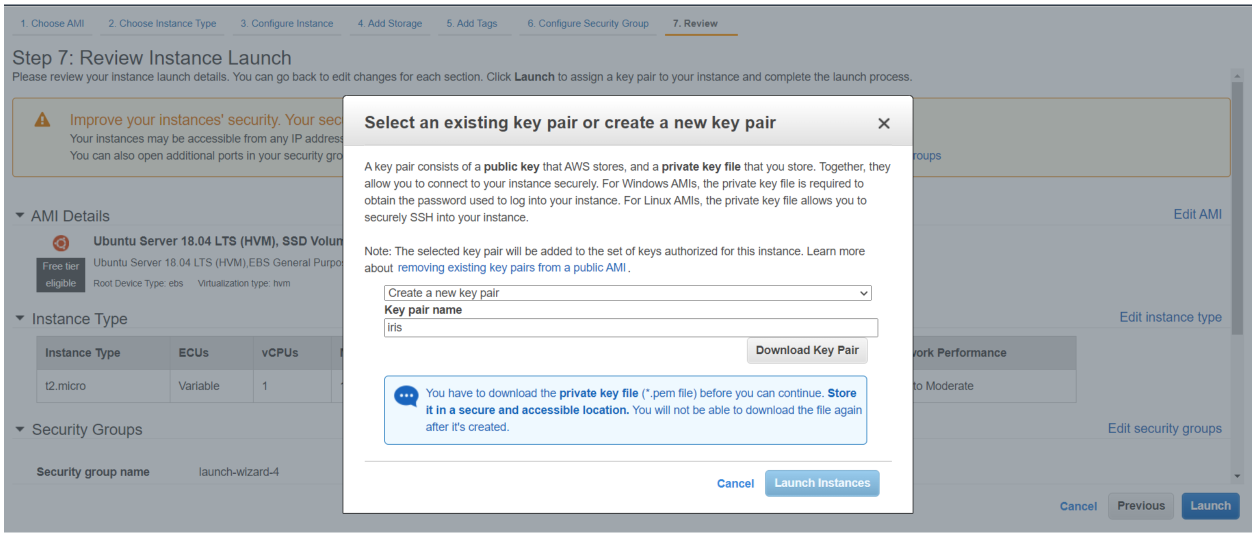
Once you followed with the above steps you will be able to click on the launch button in "Review Instance Launch". You will be getting a pop-up for choosing a private key pair. Let's click on create a new private key pair and choose a keypair name. In this case, I am choosing it as the iris. Now we need to click on the download key pair button to download it.

An Important Note is mentioned here below:-



You have to download the **private key file** (*.pem file) before you can continue. **Store it in a secure and accessible location.** You will not be able to download the file again after it's created.

Now store this file in your project folder and click on launch instances. Here is the snapshot of how to create a private key pair.



Step 7: Review Instance Launch

Please review your instance launch details. You can go back to edit changes for each section. Click **Launch** to assign a key pair to your instance and complete the launch process.

Improve your instances' security. Your sec

Your instances may be accessible from any IP address

You can also open additional ports in your security gro ... roups

AMI Details — Edit AMI

Ubuntu Server 18.04 LTS (HVM), SSD Volum

Free tier eligible — Ubuntu Server 18.04 LTS (HVM),EBS General Purpo

Root Device Type: ebs Virtualization type: hvm

Instance Type — Edit instance type

Instance Type	ECUs	vCPUs	work Performance
t2.micro	Variable	1	to Moderate

Security Groups — Edit security groups

Security group name: launch-wizard-4

Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about removing existing key pairs from a public AMI .

Create a new key pair

Key pair name

iris

Download Key Pair

You have to download the **private key file** (*.pem file) before you can continue. **Store it in a secure and accessible location.** You will not be able to download the file again after it's created.

In this way, we can be able to create an EC2 Instance. Now we need to go to view instances where you could be able to find the instance you have created.

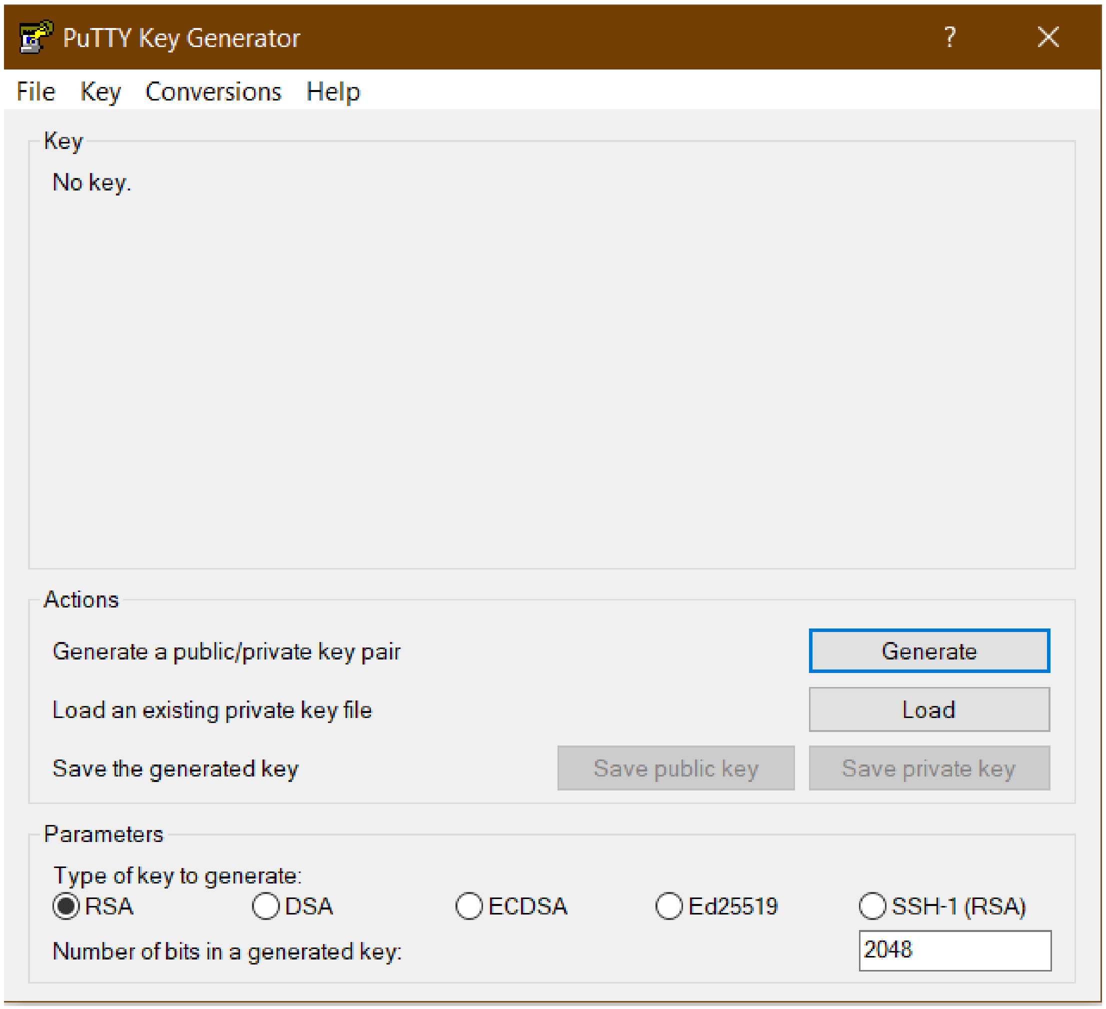
Downloading Required Softwares:

The Required Softwares to be downloaded which helps in connecting the local machine with the AWS EC2 instance. The list of Softwares to do be downloaded are

1. WinSCP:- It is a free and open-source SFTP, FTP, WebDAV, Amazon S3, and SCP client for Microsoft Windows. Its main function is secure file transfer between a local and a remote computer.
2. Putty:- It is a free and open-source terminal emulator, serial console, and network file transfer application. It supports several network protocols, including SCP, SSH, Telnet, rlogin, and raw socket connection. It can also connect to a serial port.

Converting private key from PPM to PPK Format using PuttyGen:

You need to convert a private key from PPM format which you have downloaded previously to PPK format using PuttyGen.It will be very much useful in the future for getting access to transfer files from local machine to Ubuntu Server. To convert firstly you need to open PuttyGen Software which is already installed default when you installed Putty. The Snapshot of PuttyGen looks like shown below.



PuTTY Key Generator

File Key Conversions Help

Key

No key.

Actions

Generate a public/private key pair — Generate

Load an existing private key file — Load

Save the generated key — Save public key — Save private key

Parameters

Type of key to generate:

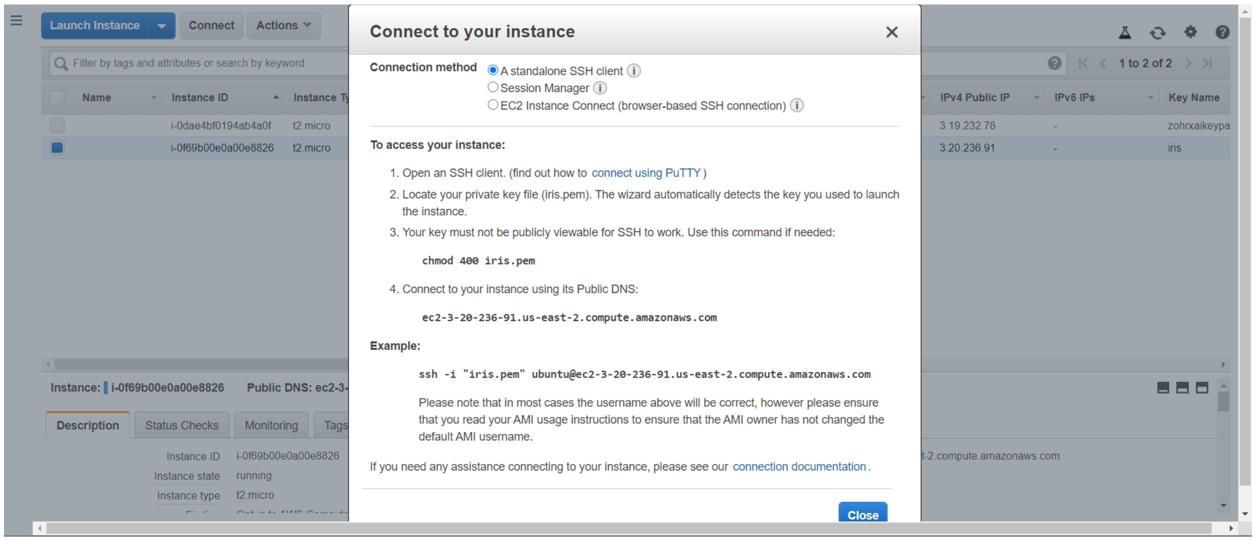
[✓ RSA] DSA ECDSA Ed25519 SSH-1 (RSA)

Number of bits in a generated key: 2048

Now you need to click on load and add the PPM file you recently downloaded. Then you need to click on “Save private key” and store it. Now close the PuttyGen window and continue to follow the next step.

Transferring Project files using WinSCP:

Now you need to transfer project files from local machine to ubuntu server using WinSCP Software. Firstly you need to go to the AWS Management Console and right-click on the instance where you will find many options. Now click on connect and you will be able to get the similar page shown below:-



Name	Instance ID	Instance Ty	IPv4 Public IP	IPv6 IPs	Key Name
	i-0dae4bf0194ab4a0f	t2.micro	3.19.232.78	-	zohrxaikeypa
	i-0f69b00e0a00e8826	t2.micro	3.20.236.91	-	iris

Instance: i-0f69b00e0a00e8826 Public DNS: ec2-3-

Instance ID i-0f69b00e0a00e8826
Instance state running
Instance type t2.micro

Connect to your instance

Connection method
(●) A standalone SSH client
( ) Session Manager
( ) EC2 Instance Connect (browser-based SSH connection)

To access your instance:

1. Open an SSH client. (find out how to connect using PuTTY)
2. Locate your private key file (iris.pem). The wizard automatically detects the key you used to launch the instance.
3. Your key must not be publicly viewable for SSH to work. Use this command if needed:

```
chmod 400 iris.pem
```

4. Connect to your instance using its Public DNS:

```
ec2-3-20-236-91.us-east-2.compute.amazonaws.com
```

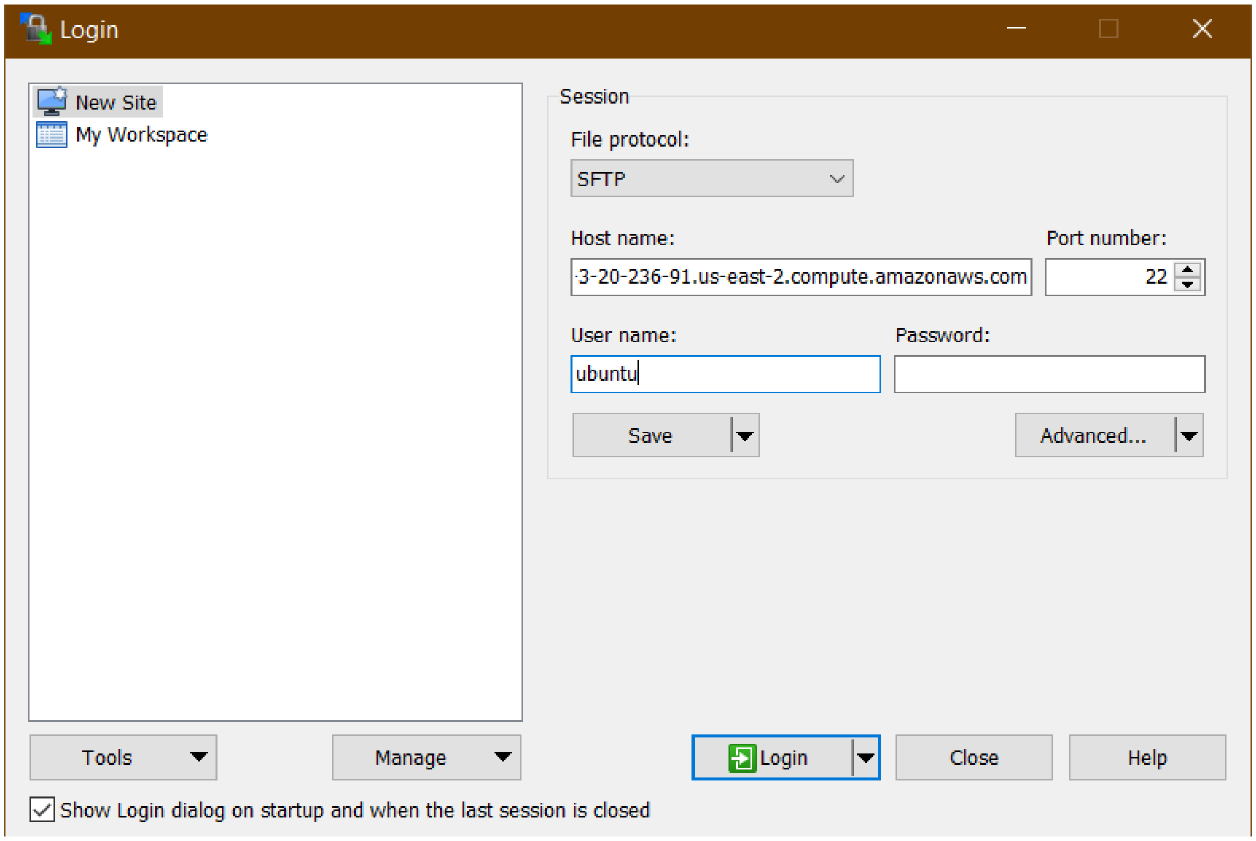
Example:

```
ssh -i "iris.pem" ubuntu@ec2-3-20-236-91.us-east-2.compute.amazonaws.com
```

Please note that in most cases the username above will be correct, however please ensure that you read your AMI usage instructions to ensure that the AMI owner has not changed the default AMI username.

If you need any assistance connecting to your instance, please see our connection documentation .

Now you need to copy the Public DNS which is shown in point no 4.Then you need to open WinSCP and need to paste this DNS address in the host input value. The window looks similar to below shown one.



File protocol: SFTP

Host name: 3-20-236-91.us-east-2.compute.amazonaws.com

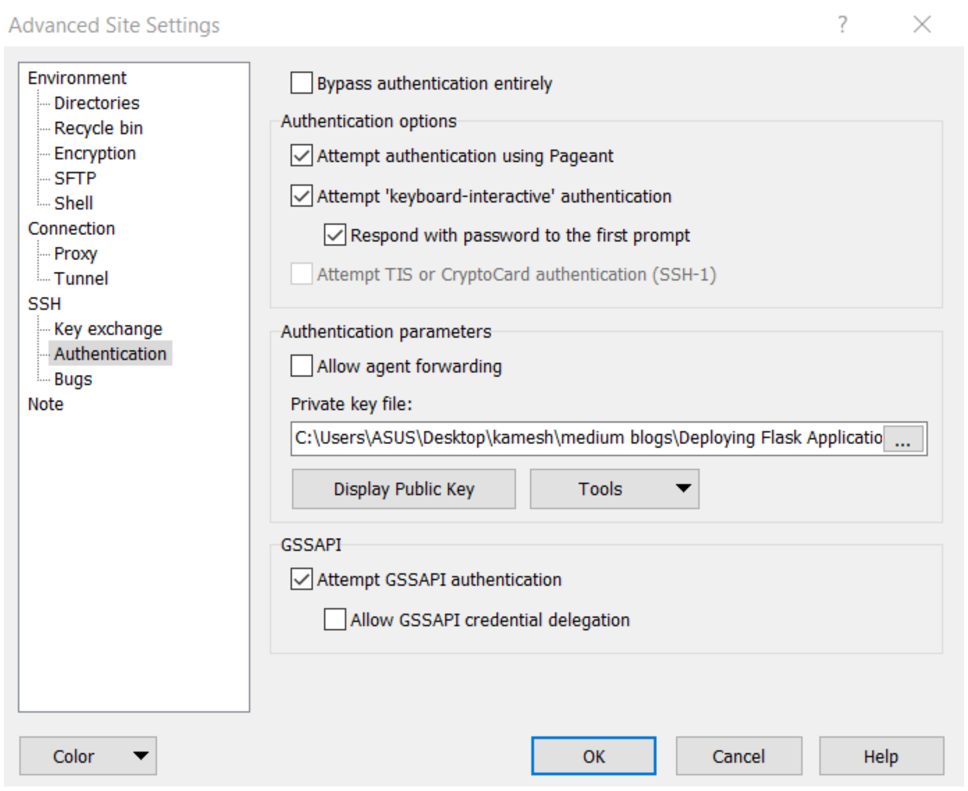
Port number: 22

User name: ubuntu

Password:

☑ Show Login dialog on startup and when the last session is closed

Now click on advanced options and go to “Authentication” and insert the created private key in PPK format. The window looks similar to below shown one.


Advanced Site Settings

Environment
- Directories
- Recycle bin
- Encryption
- SFTP
- Shell

Connection
- Proxy
- Tunnel

SSH
- Key exchange
- Authentication
- Bugs

Note

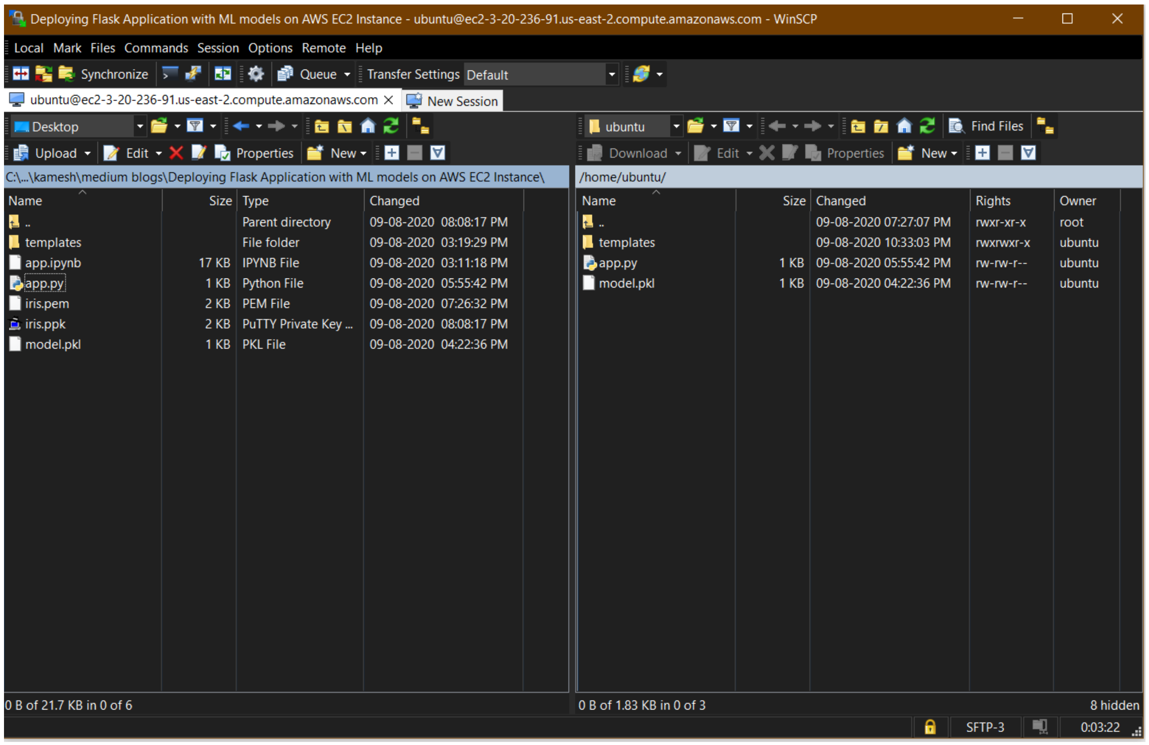
☐ Bypass authentication entirely

Authentication options
- ☑ Attempt authentication using Pageant
- ☑ Attempt 'keyboard-interactive' authentication
  - ☑ Respond with password to the first prompt
- ☐ Attempt TIS or CryptoCard authentication (SSH-1)

Authentication parameters
- ☐ Allow agent forwarding
- Private key file: C:\Users\ASUS\Desktop\kamesh\medium blogs\Deploying Flask Applicatio

GSSAPI
- ☑ Attempt GSSAPI authentication
  - ☐ Allow GSSAPI credential delegation

Now click on OK and go to log in. Now you can be able to connect with the ubuntu server. Now select the project files you wanted to add to the server and drag and drop to the right side section in the window. The file transfer starts and after a while files will be transferred. The window looks similar to below shown one.


Deploying Flask Application with ML models on AWS EC2 Instance - ubuntu@ec2-3-20-236-91.us-east-2.compute.amazonaws.com - WinSCP

Local Mark Files Commands Session Options Remote Help
Synchronize
Queue
Transfer Settings Default
ubuntu@ec2-3-20-236-91.us-east-2.compute.amazonaws.com
New Session
Desktop
Upload
Edit
Properties
New
ubuntu
Find Files
Download

C:\...\kamesh\medium blogs\Deploying Flask Application with ML models on AWS EC2 Instance\

Name	Size	Type	Changed
..		Parent directory	09-08-2020 08:08:17 PM
templates		File folder	09-08-2020 03:19:29 PM
app.ipynb	17 KB	IPYNB File	09-08-2020 03:11:18 PM
app.py	1 KB	Python File	09-08-2020 05:55:42 PM
iris.pem	2 KB	PEM File	09-08-2020 07:26:32 PM
iris.ppk	2 KB	PuTTY Private Key ...	09-08-2020 08:08:17 PM
model.pkl	1 KB	PKL File	09-08-2020 04:22:36 PM

/home/ubuntu/

Name	Size	Changed	Rights	Owner
..		09-08-2020 07:27:07 PM	rwxr-xr-x	root
templates		09-08-2020 10:33:03 PM	rwxrwxr-x	ubuntu
app.py	1 KB	09-08-2020 05:55:42 PM	rw-rw-r--	ubuntu
model.pkl	1 KB	09-08-2020 04:22:36 PM	rw-rw-r--	ubuntu

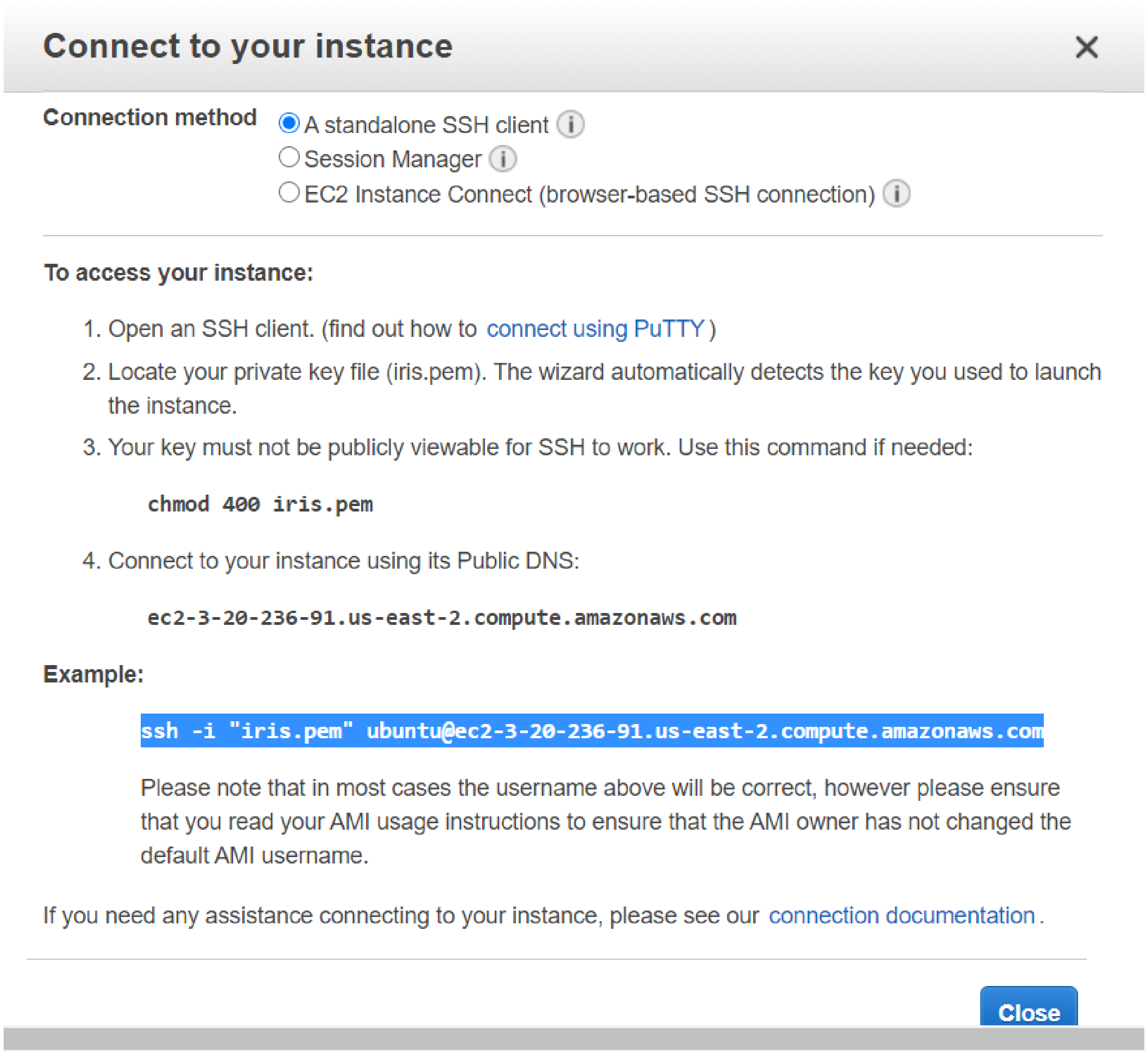
0 B of 21.7 KB in 0 of 6

0 B of 1.83 KB in 0 of 3 8 hidden

SFTP-3 0:03:22

Connect to the Instance using Git Bash:

Now required file transfer is done and the remaining thing is we need to get an access terminal to compute the files. For that open git bash and navigate to the project source. Once you have done with that you need to go to instances in AWS Management Console click on connect then you will be able to find a similar page as shown below.



Connect to your instance

Connection method [✓ A standalone SSH client] Session Manager EC2 Instance Connect (browser-based SSH connection)

To access your instance:

1. Open an SSH client. (find out how to connect using PuTTY)
2. Locate your private key file (iris.pem). The wizard automatically detects the key you used to launch the instance.
3. Your key must not be publicly viewable for SSH to work. Use this command if needed:

```
chmod 400 iris.pem
```

4. Connect to your instance using its Public DNS:

```
ec2-3-20-236-91.us-east-2.compute.amazonaws.com
```

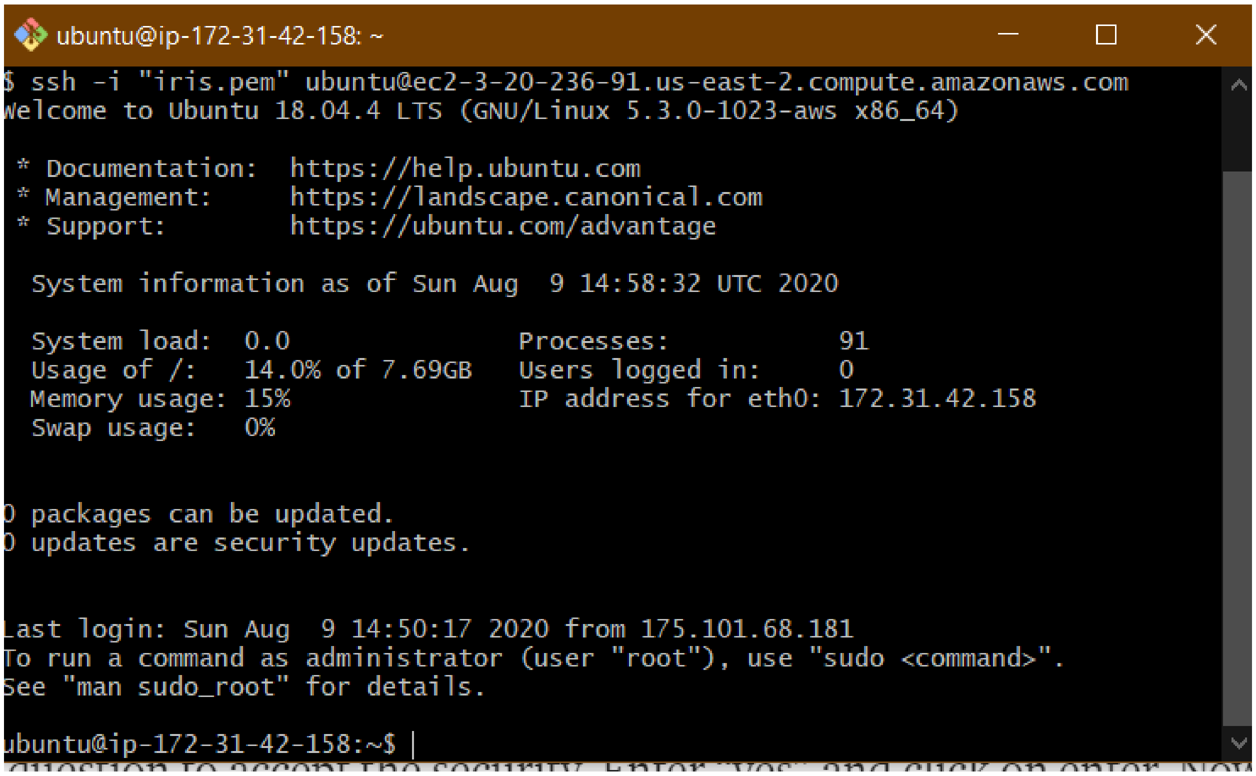
Example:

```
ssh -i "iris.pem" ubuntu@ec2-3-20-236-91.us-east-2.compute.amazonaws.com
```

Please note that in most cases the username above will be correct, however please ensure that you read your AMI usage instructions to ensure that the AMI owner has not changed the default AMI username.

If you need any assistance connecting to your instance, please see our connection documentation .

Now you need to go to the example section and copy the code given there. Now paste the command on git bash and click enter and you will be getting a question to accept the security. Enter "yes" and click on enter. Now you are connected to the instance from your local machine. The git bash looks similar to this when you successfully connected to the server instance.



```
$ ssh -i "iris.pem" ubuntu@ec2-3-20-236-91.us-east-2.compute.amazonaws.com
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 5.3.0-1023-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sun Aug  9 14:58:32 UTC 2020

  System load:  0.0               Processes:           91
  Usage of /:   14.0% of 7.69GB   Users logged in:     0
  Memory usage: 15%               IP address for eth0: 172.31.42.158
  Swap usage:   0%


0 packages can be updated.
0 updates are security updates.


Last login: Sun Aug  9 14:50:17 2020 from 175.101.68.181
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-42-158:~$ |
```

Install the Required Libraries and run the Flask Application:-

Now once the terminal is ready now we need to run some commands before running the flask app. Firstly we need to update the OS and update pip to install libraries in the future. The command for performing above process is

```
sudo apt-get update && sudo apt-get install python3-pip
```

Now you need to install all required libraries to run the flask application. The required commands for this application are as follows:-

```
pip3 install numpy
pip3 install flask
pip3 install sklearn
```

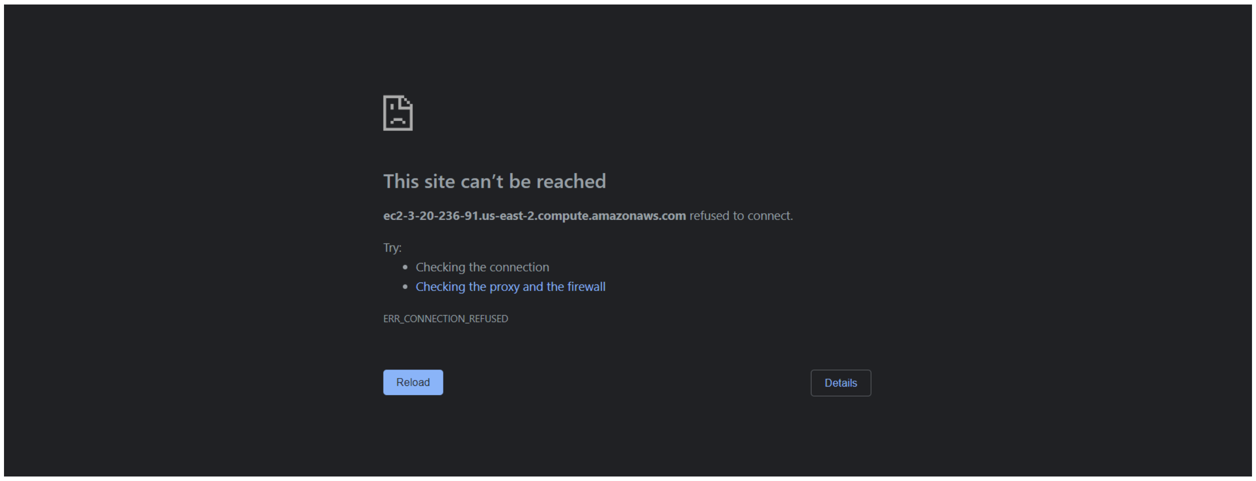
Now all the libraries are installed now we need to run the application by entering following command on git bash.

```
python3 app.py
```

Now the application started running. You can view the application at the URL as <your DNS>::<your port no>.In my case it is http://ec2-3-20-236-91.us-east-2.compute.amazonaws.com:8080/

Important Note:-

Now your application is hosted but there is a problem you will face once you close your git bash. The problem is once if you close git bash and you reload the URL your screen looks similar to below.



This site can't be reached

ec2-3-20-236-91.us-east-2.compute.amazonaws.com refused to connect.

Try:

- Checking the connection
- Checking the proxy and the firewall

ERR_CONNECTION_REFUSED

How to Solve this problem?

To solve this issue we need to use Screen. Screen or GNU Screen is a terminal multiplexer. In other words, it means that you can start a screen session and then open any number of windows (virtual terminals) inside that session. Processes running in Screen will continue to run when their window is not visible even if you get disconnected.

For this, we need to run the following command on the git bash.

```
screen -R deploy python3 app.py
```

You can exit (detach) the screen by pressing "Ctrl-A" and "d" but the server will keep running.

After you detach the screen, let say you are disconnecting your **SSH** session and going home. In your home, you start to **SSH** again to your server and you want to see the progress of your download process. To do that, you need to restore the screen. You can run this command:

```
screen -r
```

And you will see that the process you left is still running.

When you have more than **1 screen** session, you need to type the screen session **ID**. Use screen **-ls** to see how many screen are available.

```
screen -ls
```