A comprehensive beginner's guide to start ML with Amazon Web Services (AWS)

Introduction

> *Learn to connect AWS instance with your laptop / desktop for faster computation!*

Do you struggle with working on big data (large data sets) on your laptop ? I recently tried working on a 10 GB image recognition data set. But, due to the limited computational power of my laptop, I couldn't proceed further. I was determined to solve this problem, and thankfully, in few hours, I managed to set up a 24GB machine on AWS for FREE and got improved results.

I got it for FREE because I used the trial version with limited features, just to see how fast could it work. To my surprise, it was blazing fast. I am sure there would be many more like me, who just don't know how to deal with big data sets on their laptop. In fact, most of us think of buying a new powerful laptop when faced with this hurdle. But, computing on AWS platform is quite cheaper. Trust me!

Just to help you deal with this problem, I've shared the step by step process I used to set up connect 24GB AWS instance with my laptop. To show how it works, I've also demonstrated some computations on *iris data* set in *Python*. For R users, everything remains same except on line of code which I've highlighted as well.

Note: This is not a promotional post. I do not intend to publicize AWS in analytics community. This post is meant for people who face trouble working on large data sets and wish to work faster.

Table of Contents

1. What are AWS Instances?

Amazon **EC2 (Elastic Compute Cloud)** instances provide access to scalable computing resources on the cloud. This allow organizations and individuals to stay away from setting up and maintaining hardware on site. You can use it just like your own laptop but get your dream configuration for a cheaper bargain with the option of varying it as you need in a pay-per-use setting. Amazon EC2 allows you to manage your security, networking and storage as desired.

Before moving forward, lets try and understand some of the terminologies used by AWS. This will make it easier for to use the interface and the platform.

1.1 Instances and AMIs

As described by AWS, An **AMI (Amazon Machine Image)** is a template that contains a software configuration (for example, an operating system, an application server, and applications). You can understand it as a high level definition of your system. Different AMIs can be understood as different categories of systems.

For instance, Windows OS vs Linux OS. But this is high level definition and not sufficient to completely define the hardware. So, we go to the next level and define an instance which is just a copy of an AMI but with granular details specified which are required for a system, such as the RAM, memory storage, number of cores, etc.

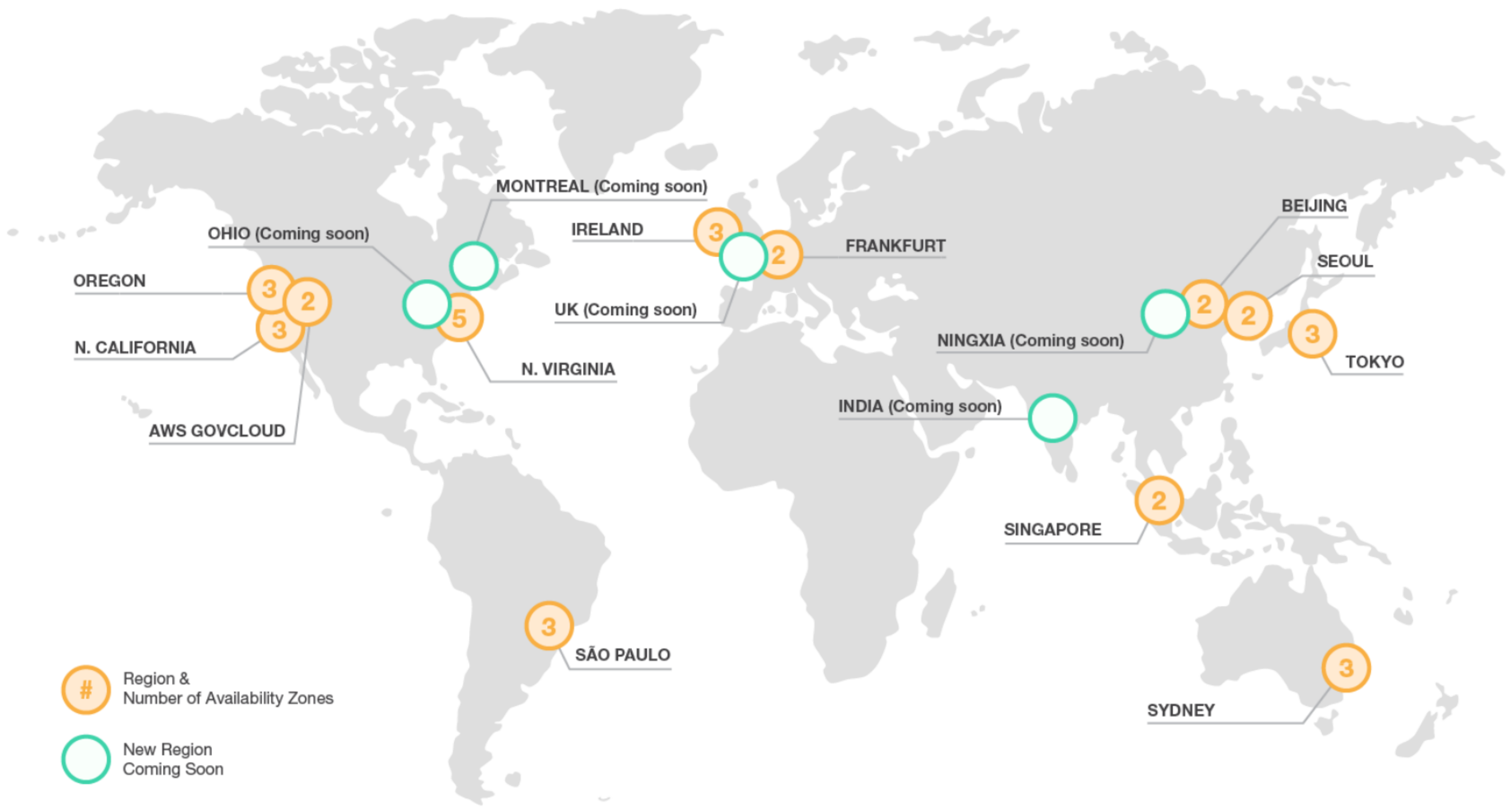
You can run many instances of same or different types of AMIs. The limit typically depends on your price-plan which can be configured on individual basis. You can find more details here.

Also, some AMIs are predefined by AWS but you can also define your custom AMI. For instance, if you have a website then you can define your own AMI containing your website code and static content so that it is easy to faster working on a new instance. Some AWS developers have also developed some custom AMIs which can be purchased from AWS stores.

1.2 Regions and Availability Zones

Amazon data centers are location globally at multiple locations. A region refers to a geographically separated location while availability zones are isolated locations within a region. You can choose to place your resources across multiple availability zones as well as regions. If a particular location is affected by a failure, all other locations will still continue to run normally.

The infrastructure of AWS can be observed on the following map given by AWS:



Some points to note:

- AWS does not provide any data transfer between regions. So that part has to be managed using your own security protocols over public internet
- Even if we place instances over multiple availability zones and one of them fails, then we can define Elastic IP addresses
- Availability zones are represented by a region code followed by a letter identifier, for instance "us-west-1a". But these identifiers are defined for one account and cannot be compared across accounts.

1.3 Instance Types

The instance type determines the **hardware of the host computer** used for the instance. Each instance type offers different compute, memory, and storage capabilities and are grouped in instance families based on these capabilities.

Typically, some part of the host computer's resources are dedicated for an instance but in case of under-utilized resources, the same can be given to other instances. So the resources vary between a range for a given type of instance.

Instances are either **current generation** instances or **previous generation** instances. The new users should always use current generation as the previous generation instances are for those who already have their applications optimized on them and shifting to a new generation instance is time consuming.

AWS provides a plethora of instances for different use cases. You can find a full list here. I will not go into each one of them. But here I will discuss some concepts/AWS terminologies which are important to understand some of the features of those instances.

1.3.1 ECU (EC2 Compute Unit)

Since AWS servers are based on a variety of hardware, its very difficult to provide a performance comparison in absolute terms. To overcome this challenge, AWS has coined an **ECU as a hardware benchmark**. 1 ECU is the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor. So now we can easily compare the performance of a 10ECU instance vs say an 88ECU instance.

1.3.2 Burstable Performance Instances

In such an instance, a **minimum level** of computing power is predefined but it can have a **boost** depending upon the idle time and usage. These instances earn credits when they are idle and use them when they go active. The number of credits earned depends on the size of instance.

These are designed for applications such as web servers, developer environments and small databases which don't need consistently high levels of CPU, but benefit significantly from having full access to very fast CPUs when they need them.

1.3.3 Amazon EBS (Elastic Block Store)

In AWS, memory is available in two forms. First is the **instance store**, which is an **ephemeral** storage, i.e. it is temporary and all data in it gets deleted as the instance is stopped. But this storage is right on the instance and data transfer is very fast.

Another form is **Amazon EBS** which is a like an **external hard-disk** attached to the system. Any data available on this remains there even when the instance is not under operation. But since this is like an external device, the data transfer rate will be slow as multiple devices will be using space on the same hardware.

The speed will this vary depending on the load. To overcome this challenge, AWS offers an option called as **EBS Optimized**. This allows a **dedicated bandwidth** between AWS server and the data storage device. It might come complimentary with the instance or can be purchased for an additional fee.

1.3.4 Dedicated Instances

These are instances which are **dedicated at the server hardware level**. Typically, a server will be used to run multiple instances belonging to various customers. But if a customer demands a dedicated server which will only run instances for that particular account, dedicated instances are an option.

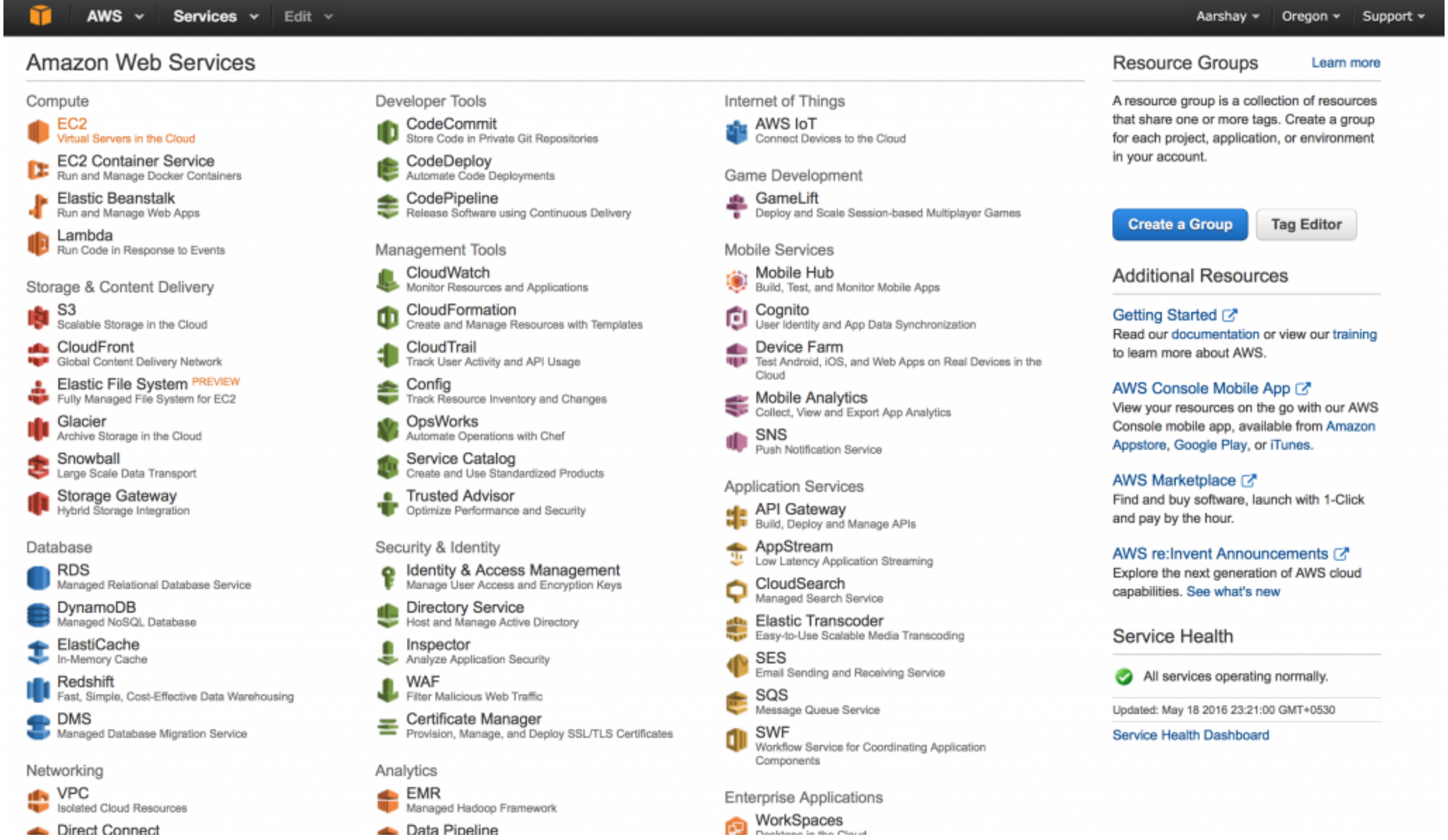
2. How to Configure and Launch an Instance?

Now, lets setup and launch an instance using the knowledge we've gained till now. For the purpose of this demo, I will use the instance which is available for free so that you can also try it out with no cost involved. However, for specific applications, you will have to use an instance depending on the computing requirements.

In order to get your first instance ready, please follow the given steps:

Step 0: Login to Your AWS Account

Obviously, first we need an AWS account and we need to log into that. AWS offers some free instances for the first which we are going to leverage in this tutorial. After logging into your account, your management console should look like this:



Amazon Web Services

Compute
- EC2 — Virtual Servers in the Cloud
- EC2 Container Service — Run and Manage Docker Containers
- Elastic Beanstalk — Run and Manage Web Apps
- Lambda — Run Code in Response to Events

Storage & Content Delivery
- S3 — Scalable Storage in the Cloud
- CloudFront — Global Content Delivery Network
- Elastic File System PREVIEW — Fully Managed File System for EC2
- Glacier — Archive Storage in the Cloud
- Snowball — Large Scale Data Transport
- Storage Gateway — Hybrid Storage Integration

Database
- RDS — Managed Relational Database Service
- DynamoDB — Managed NoSQL Database
- ElastiCache — In-Memory Cache
- Redshift — Fast, Simple, Cost-Effective Data Warehousing
- DMS — Managed Database Migration Service

Networking
- VPC — Isolated Cloud Resources
- Direct Connect

Developer Tools
- CodeCommit — Store Code in Private Git Repositories
- CodeDeploy — Automate Code Deployments
- CodePipeline — Release Software using Continuous Delivery

Management Tools
- CloudWatch — Monitor Resources and Applications
- CloudFormation — Create and Manage Resources with Templates
- CloudTrail — Track User Activity and API Usage
- Config — Track Resource Inventory and Changes
- OpsWorks — Automate Operations with Chef
- Service Catalog — Create and Use Standardized Products
- Trusted Advisor — Optimize Performance and Security

Security & Identity
- Identity & Access Management — Manage User Access and Encryption Keys
- Directory Service — Host and Manage Active Directory
- Inspector — Analyze Application Security
- WAF — Filter Malicious Web Traffic
- Certificate Manager — Provision, Manage, and Deploy SSL/TLS Certificates

Analytics
- EMR — Managed Hadoop Framework
- Data Pipeline

Internet of Things
- AWS IoT — Connect Devices to the Cloud

Game Development
- GameLift — Deploy and Scale Session-based Multiplayer Games

Mobile Services
- Mobile Hub — Build, Test, and Monitor Mobile Apps
- Cognito — User Identity and App Data Synchronization
- Device Farm — Test Android, iOS, and Web Apps on Real Devices in the Cloud
- Mobile Analytics — Collect, View and Export App Analytics
- SNS — Push Notification Service

Application Services
- API Gateway — Build, Deploy and Manage APIs
- AppStream — Low Latency Application Streaming
- CloudSearch — Managed Search Service
- Elastic Transcoder — Easy-to-Use Scalable Media Transcoding
- SES — Email Sending and Receiving Service
- SQS — Message Queue Service
- SWF — Workflow Service for Coordinating Application Components

Enterprise Applications
- WorkSpaces

Resource Groups — Learn more

A resource group is a collection of resources that share one or more tags. Create a group for each project, application, or environment in your account.

Create a Group | Tag Editor

Additional Resources

Getting Started
Read our documentation or view our training to learn more about AWS.

AWS Console Mobile App
View your resources on the go with our AWS Console mobile app, available from Amazon Appstore, Google Play, or iTunes.

AWS Marketplace
Find and buy software, launch with 1-Click and pay by the hour.

AWS re:Invent Announcements
Explore the next generation of AWS cloud capabilities. See what's new

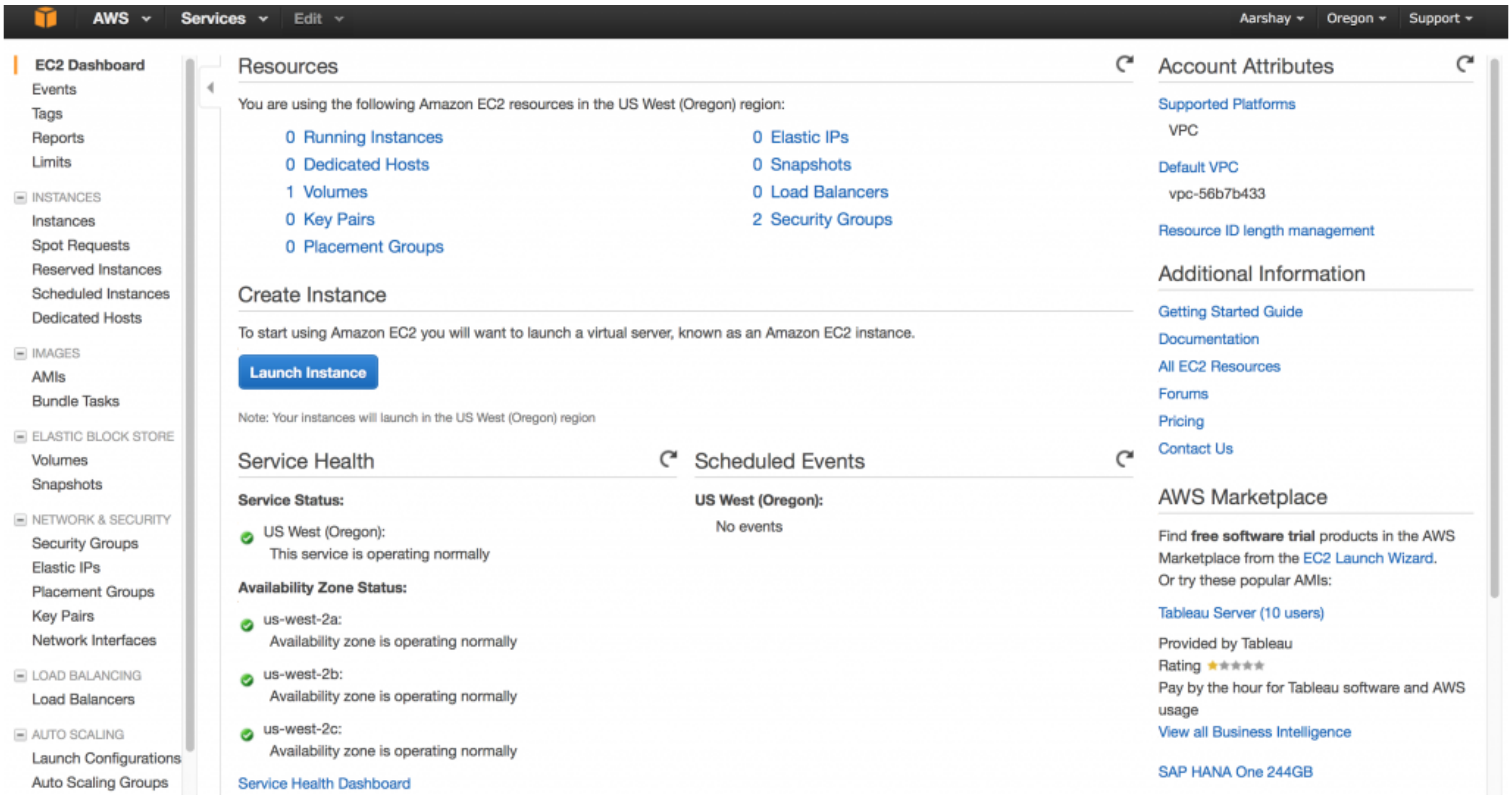
Service Health

All services operating normally.

Updated: May 18 2016 23:21:00 GMT+0530

Service Health Dashboard

On this page, you should select the "EC2" option which I've highlighted in orange on top-left side. This will take you to the EC2 Dashboard which looks like:



Resources

You are using the following Amazon EC2 resources in the US West (Oregon) region:

0 Running Instances	0 Elastic IPs
0 Dedicated Hosts	0 Snapshots
1 Volumes	0 Load Balancers
0 Key Pairs	2 Security Groups
0 Placement Groups	

Create Instance

To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance.

Launch Instance

Note: Your instances will launch in the US West (Oregon) region

Service Health

Service Status:

US West (Oregon):
This service is operating normally

Availability Zone Status:

us-west-2a:
Availability zone is operating normally

us-west-2b:
Availability zone is operating normally

us-west-2c:
Availability zone is operating normally

Service Health Dashboard

Scheduled Events

US West (Oregon):
No events

Account Attributes

Supported Platforms
VPC

Default VPC
vpc-56b7b433

Resource ID length management

Additional Information

Getting Started Guide
Documentation
All EC2 Resources
Forums
Pricing
Contact Us

AWS Marketplace

Find **free software trial** products in the AWS Marketplace from the EC2 Launch Wizard.
Or try these popular AMIs:

Tableau Server (10 users)

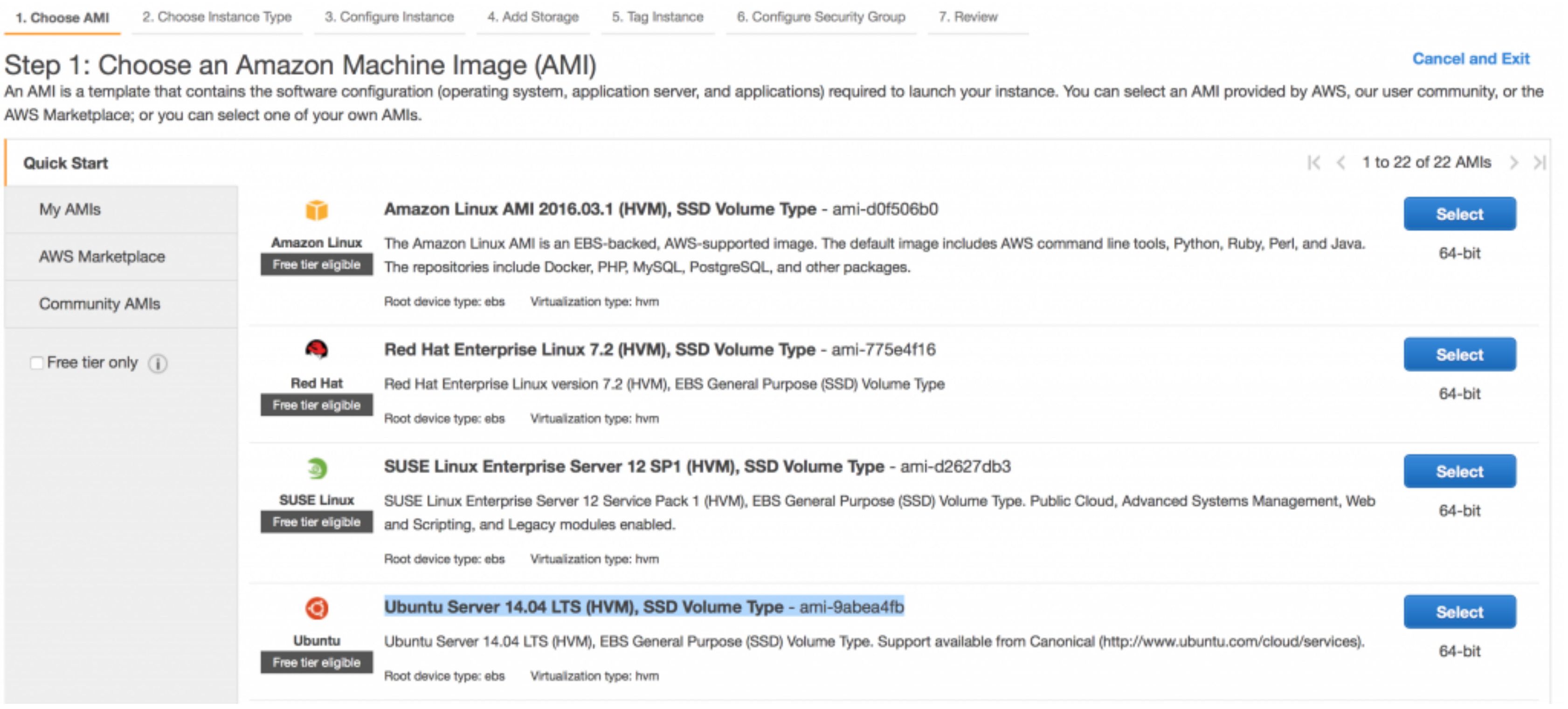
Provided by Tableau
Rating ★★★★★
Pay by the hour for Tableau software and AWS usage
View all Business Intelligence

SAP HANA One 244GB

This is the console where you can manage your instances. This gives a snapshot of your account. Since we're making our first instance, lets directly click the big blue button in the center – *Launch Instance*. This will start the process of launching a new instance.

Step 1: Choose an Amazon Machine Image (AMI)

Now we have to select our AMI. The Launch Instance button will take us to the following screen:



1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Tag Instance 6. Configure Security Group 7. Review

Step 1: Choose an Amazon Machine Image (AMI)

Cancel and Exit

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.

Quick Start — 1 to 22 of 22 AMIs

My AMIs
AWS Marketplace
Community AMIs
Free tier only

Amazon Linux — Free tier eligible
Amazon Linux AMI 2016.03.1 (HVM), SSD Volume Type - ami-d0f506b0
The Amazon Linux AMI is an EBS-backed, AWS-supported image. The default image includes AWS command line tools, Python, Ruby, Perl, and Java. The repositories include Docker, PHP, MySQL, PostgreSQL, and other packages.
Root device type: ebs Virtualization type: hvm
Select — 64-bit

Red Hat — Free tier eligible
Red Hat Enterprise Linux 7.2 (HVM), SSD Volume Type - ami-775e4f16
Red Hat Enterprise Linux version 7.2 (HVM), EBS General Purpose (SSD) Volume Type
Root device type: ebs Virtualization type: hvm
Select — 64-bit

SUSE Linux — Free tier eligible
SUSE Linux Enterprise Server 12 SP1 (HVM), SSD Volume Type - ami-d2627db3
SUSE Linux Enterprise Server 12 Service Pack 1 (HVM), EBS General Purpose (SSD) Volume Type. Public Cloud, Advanced Systems Management, Web and Scripting, and Legacy modules enabled.
Root device type: ebs Virtualization type: hvm
Select — 64-bit

Ubuntu — Free tier eligible
Ubuntu Server 14.04 LTS (HVM), SSD Volume Type - ami-9abea4fb
Ubuntu Server 14.04 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
Select — 64-bit

You can see various options here:

- **Quick Start**: These are AWS defined default AMIs which we can choose
- **My AMIs**: Custom AMIs created in the account
- **AWS MarketPlace**: Third-party AMIs pre-configured for running some specific softwares
- **Community AMIs**: Specific purpose AMIs created by AWS developers

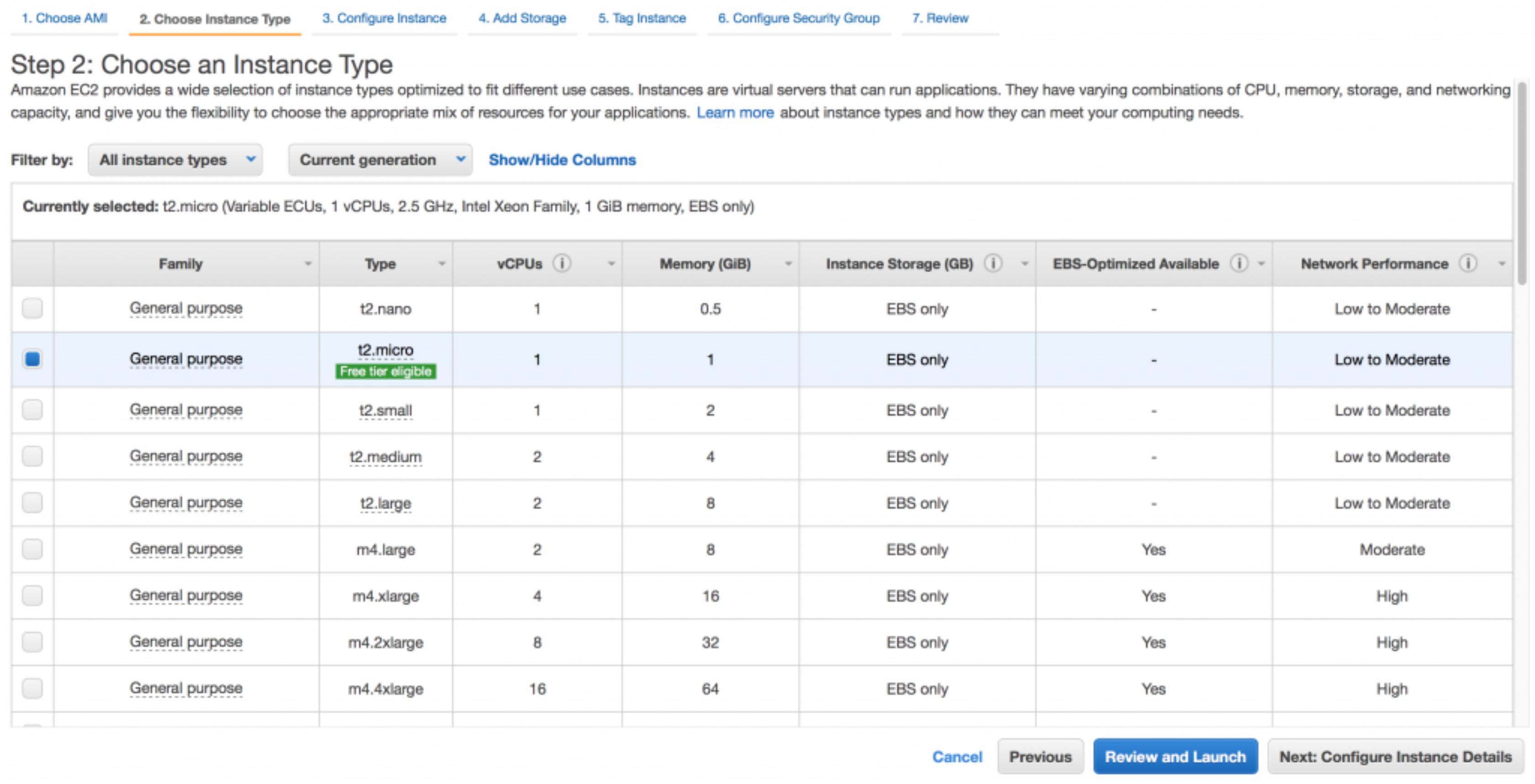
Right now, we'll just use one of the Quick Start AMIs. Lets take an Ubuntu Server 14.04 AMI as selected by me in the image above. You should explore other options as well whenever you get a chance.

Note that this page shows only 4 AMIs but there are many more as you scroll down. You should be cautious of the "Free Tier Eligible" tag. Only AMIs having this tag will fall under the free resources available in the first year.

Lets select the Ubuntu Server and move ahead.

Step 2: Choose an Instance Type

In this step, we have to choose an instance which will be based on the AMI selected above. The page looks like:


1. Choose AMI | 2. Choose Instance Type | 3. Configure Instance | 4. Add Storage | 5. Tag Instance | 6. Configure Security Group | 7. Review

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.

Filter by: All instance types | Current generation | Show/Hide Columns

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
☐	General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate
■	General purpose	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate
☐	General purpose	t2.small	1	2	EBS only	-	Low to Moderate
☐	General purpose	t2.medium	2	4	EBS only	-	Low to Moderate
☐	General purpose	t2.large	2	8	EBS only	-	Low to Moderate
☐	General purpose	m4.large	2	8	EBS only	Yes	Moderate
☐	General purpose	m4.xlarge	4	16	EBS only	Yes	High
☐	General purpose	m4.2xlarge	8	32	EBS only	Yes	High
☐	General purpose	m4.4xlarge	16	64	EBS only	Yes	High

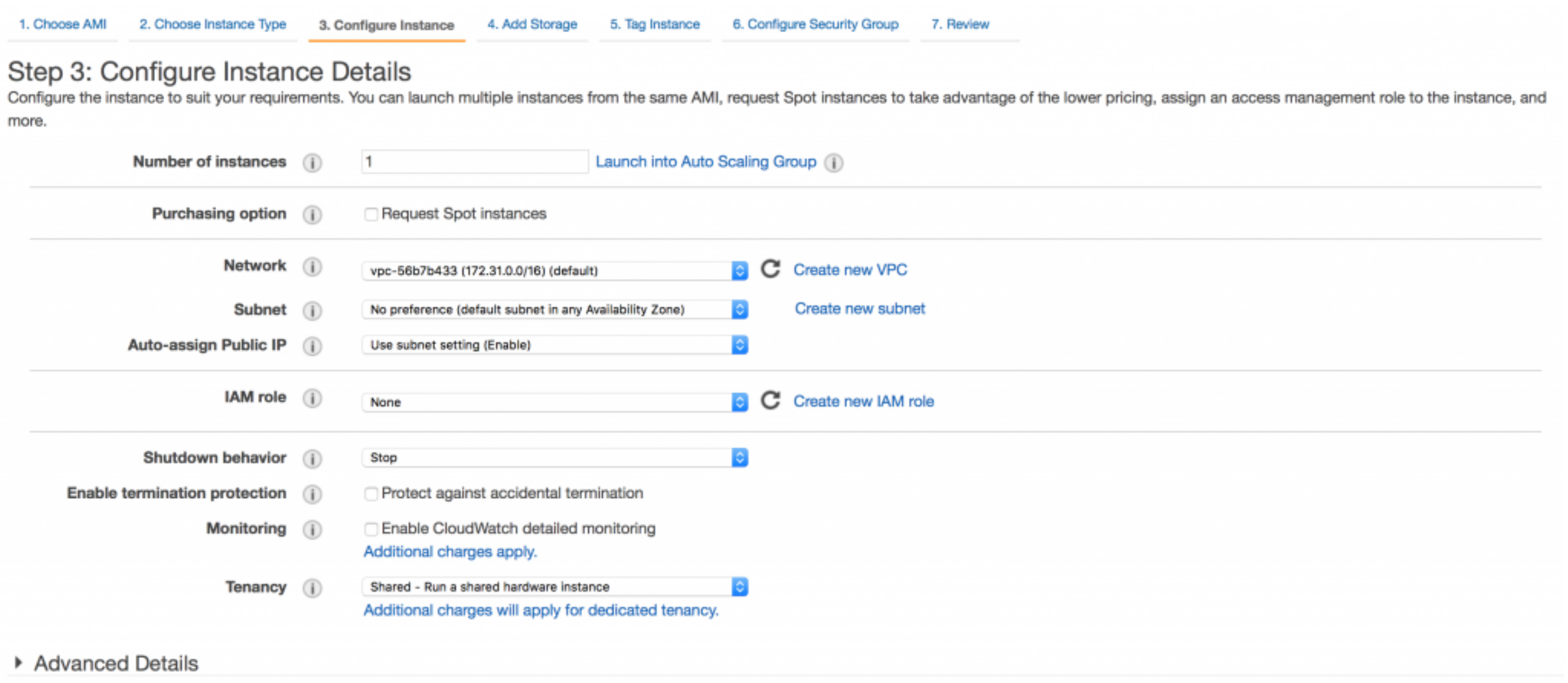
Here you can see 2 filter options on the top:

1. **All Instance Type**: Here you can filter out and select one of the broad instance types as described here.
2. **Current Generation**: This option can be modified to get previous generation instances as well

Here we'll simply select *t2.micro* instance because it is the only one with **Free tier eligible**.

Step 3: Configure Instance Details

The interface looks like:


1. Choose AMI | 2. Choose Instance Type | 3. Configure Instance | 4. Add Storage | 5. Tag Instance | 6. Configure Security Group | 7. Review

Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of instances	1	Launch into Auto Scaling Group
Purchasing option	☐ Request Spot instances	
Network	vpc-56b7b433 (172.31.0.0/16) (default)	Create new VPC
Subnet	No preference (default subnet in any Availability Zone)	Create new subnet
Auto-assign Public IP	Use subnet setting (Enable)	
IAM role	None	Create new IAM role
Shutdown behavior	Stop	
Enable termination protection	☐ Protect against accidental termination	
Monitoring	☐ Enable CloudWatch detailed monitoring Additional charges apply.	
Tenancy	Shared - Run a shared hardware instance Additional charges will apply for dedicated tenancy.	

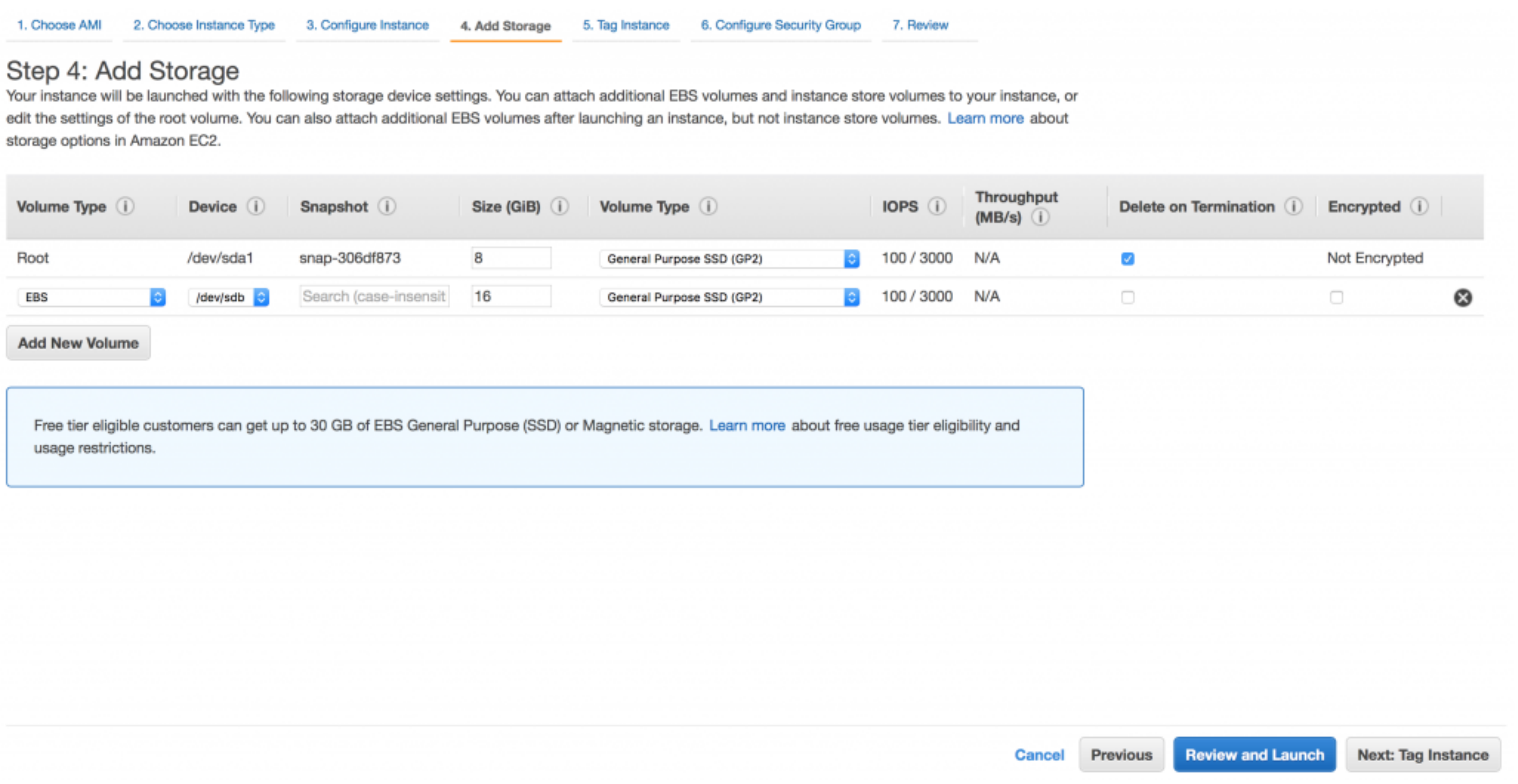
▸ Advanced Details

This page allows various settings like:

- **Purchasing option**: select between a regular or a spot instance
- **Tenancy**: select if a dedicated instance is required

Step 4: Add Storage

Now we can configure the memory settings of the instance. The page looks like:



Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. Learn more about storage options in Amazon EC2.

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encrypted	
Root	/dev/sda1	snap-306df873	8	General Purpose SSD (GP2)	100 / 3000	N/A	☑	Not Encrypted	
EBS	/dev/sdb	Search (case-insensit	16	General Purpose SSD (GP2)	100 / 3000	N/A	☐	☐	⊗

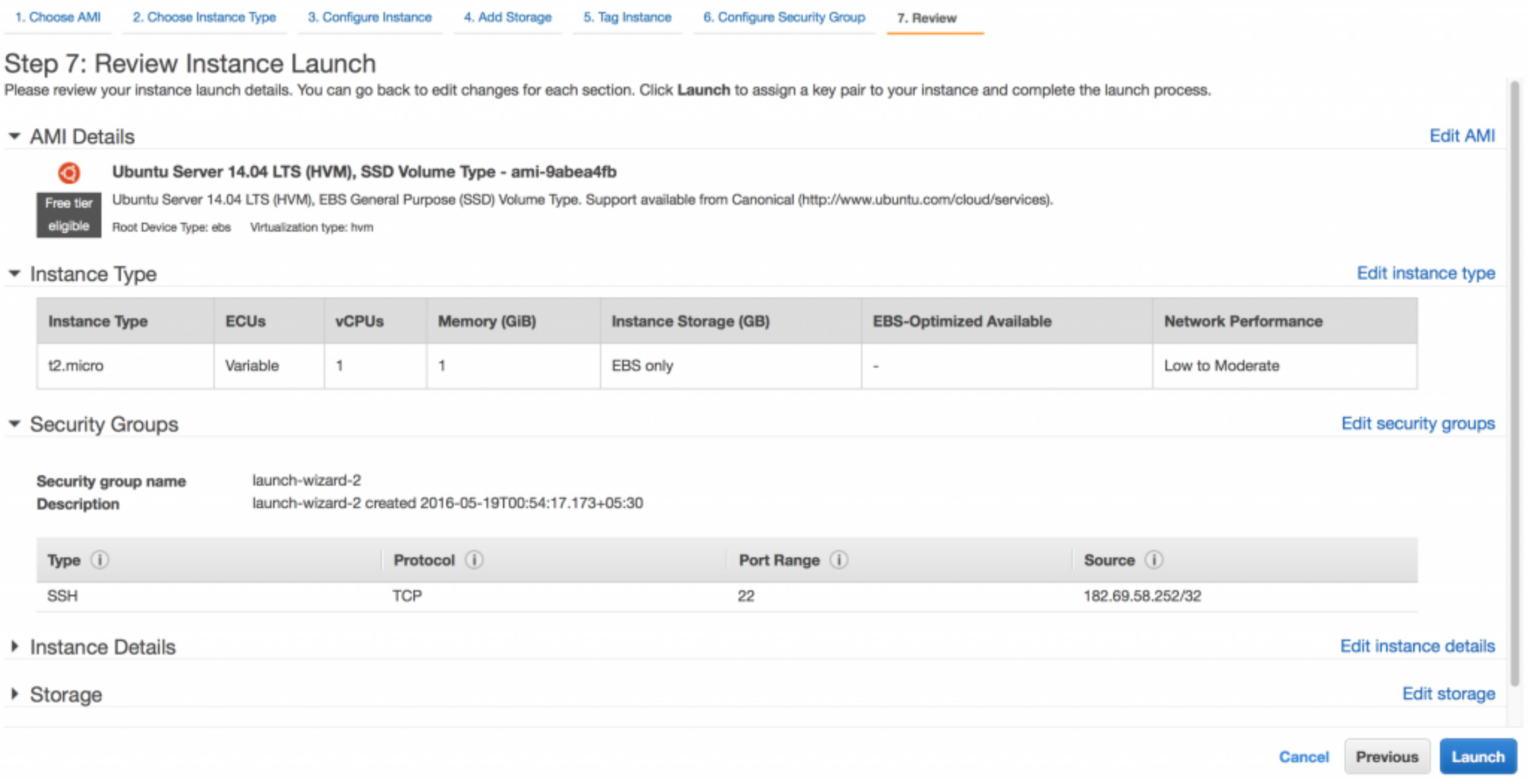
Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. Learn more about free usage tier eligibility and usage restrictions.

As you can see, by default an 8GB root memory is available. As the page specifies, we can add upto 30GB of general purpose EBS as part of the free tier. So I've added another 16GB memory here.

Note that we will skip the next 2 steps in this basic tutorial. You can review and fill some values as per your application. Lets click on Review and Launch to go to the last step.

Step 5: Final Launch

This step shows you a summary of the selections made before you launch. For us, the page would look like:



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.

AMI Details — Edit AMI

Ubuntu Server 14.04 LTS (HVM), SSD Volume Type - ami-9abea4fb

Free tier eligible

Ubuntu Server 14.04 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

Instance Type — Edit instance type

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
t2.micro	Variable	1	1	EBS only	-	Low to Moderate

Security Groups — Edit security groups

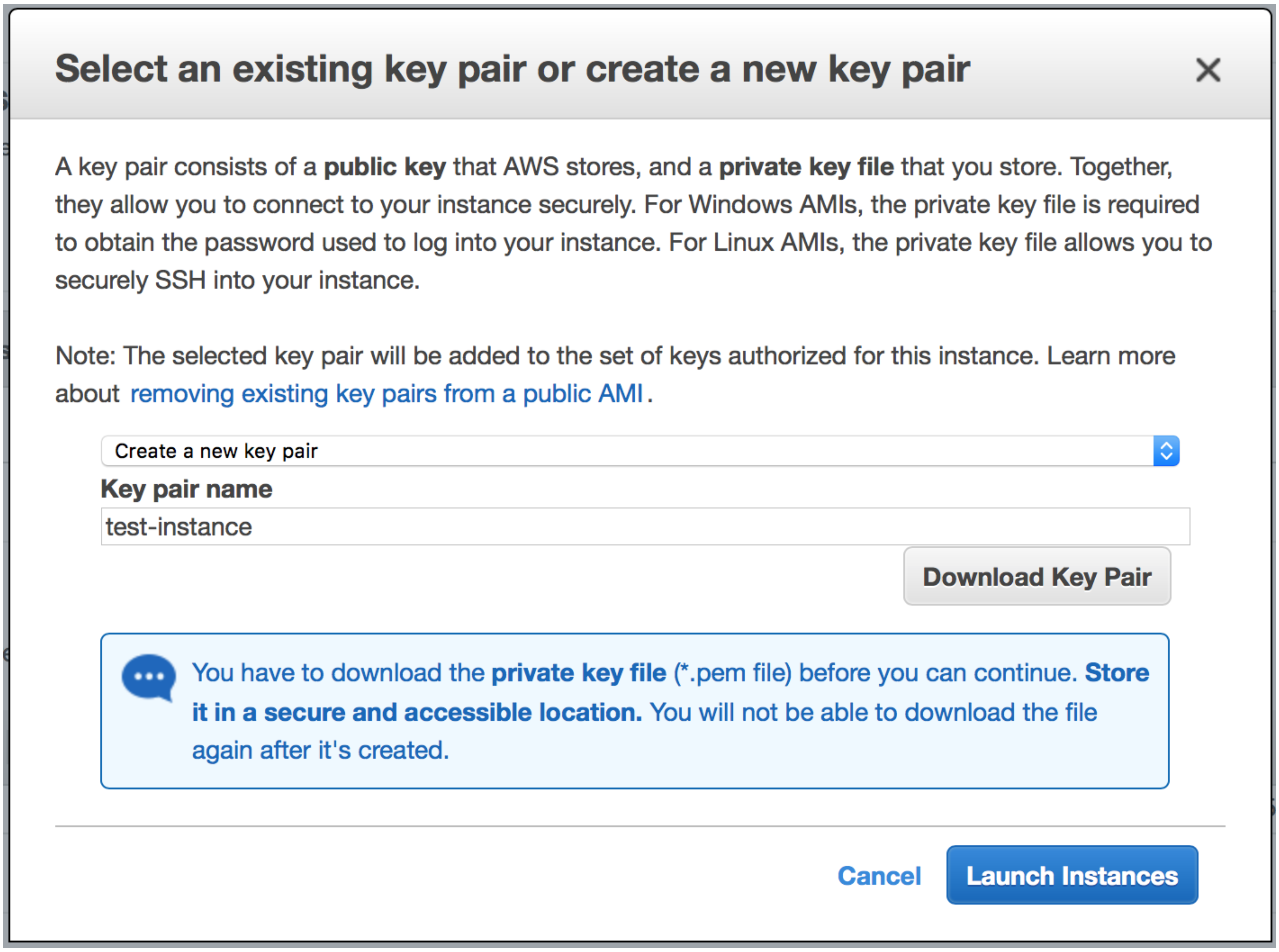
Security group name	launch-wizard-2
Description	launch-wizard-2 created 2016-05-19T00:54:17.173+05:30

Type	Protocol	Port Range	Source
SSH	TCP	22	182.69.58.252/32

Instance Details — Edit instance details

Storage — Edit storage

You can check the selections and launch the instance. While launching, AWS will pop up for a key-value pair. This is nothing but a security measure. Instead of remembering the AMI password for launching the instance each time, we can create a key-paid which we can download on our local as a .pem file. This can be used in place of the password always. It can be created as:



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

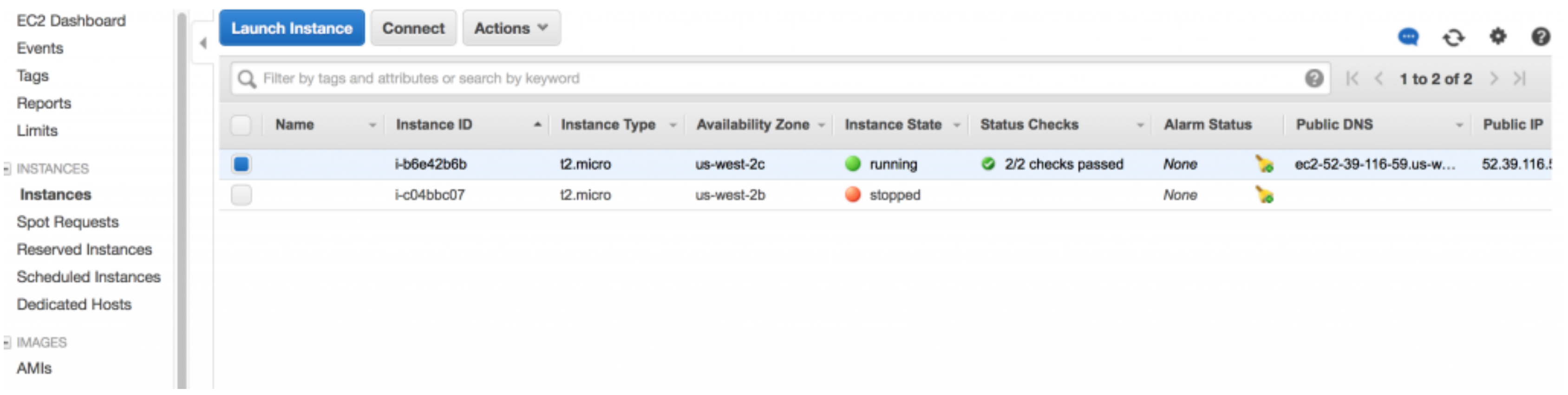
Key pair name

test-instance

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.

Just select “Create a new key pair”, then type a name and download the .pem file and store in a known location. This will be used to access the instance.

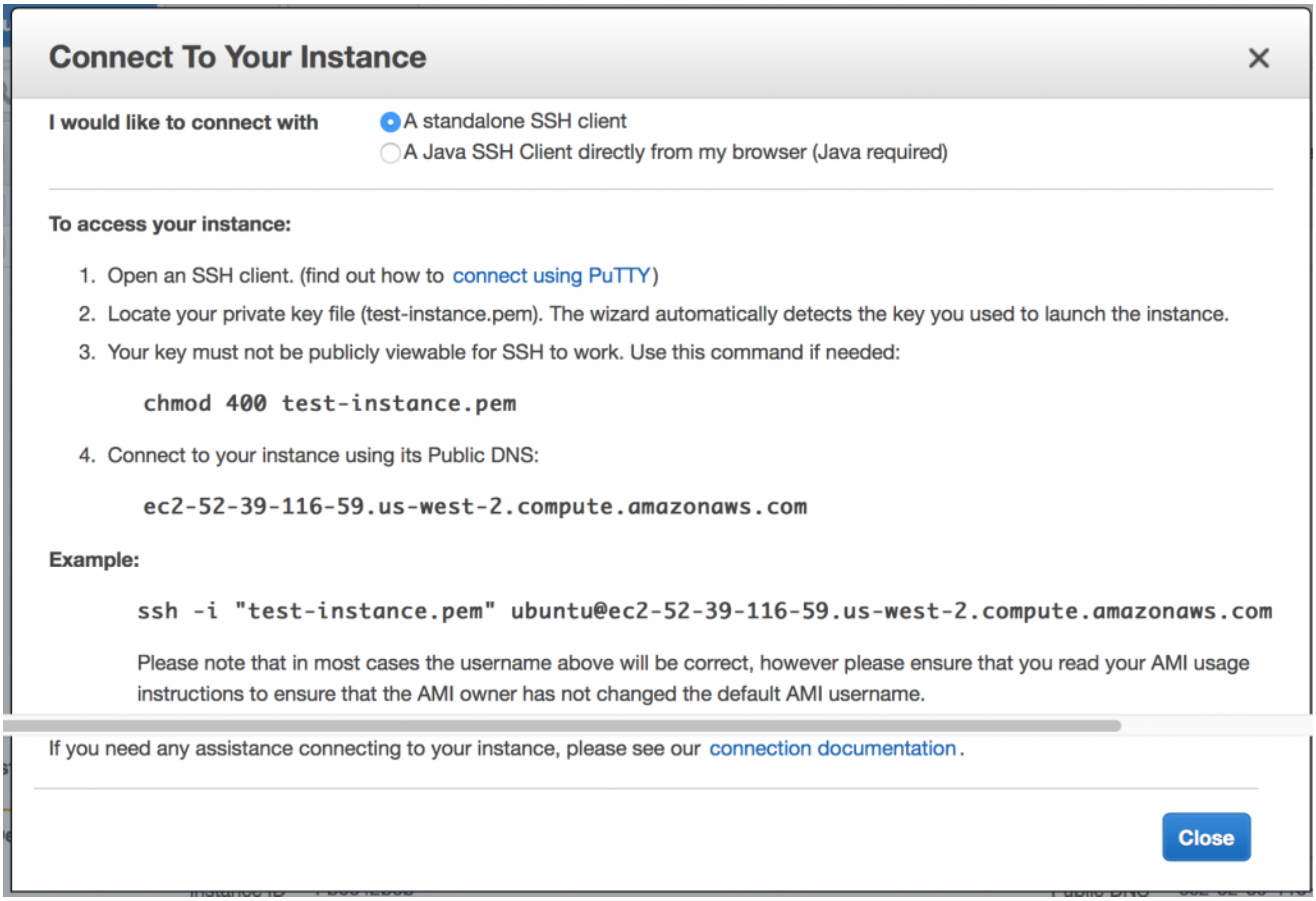
Now that the instance is launched, it will redirect us to the EC2 Dashboard where we can check the instances running. For me, this looks like:



	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS	Public IP
☑		i-b6e42b6b	t2.micro	us-west-2c	● running	✓ 2/2 checks passed	None	ec2-52-39-116-59.us-w...	52.39.116.
☐		i-c04bbc07	t2.micro	us-west-2b	● stopped		None		

Here you can see 2 instances but the top one is the one we’ve created just now. Note that instance takes some time to get configured. The **Status Checks** columns shows Initializing.. as a comment while the instance is being launched. Now that we can see all checks passed, lets click **Connect** which will allow us to start talking (I mean in code language :P) to our instance.

You’ll get the following pop-up:



Connect To Your Instance

I would like to connect with [✓ A standalone SSH client] A Java SSH Client directly from my browser (Java required)

To access your instance:

1. Open an SSH client. (find out how to connect using PuTTY)
2. Locate your private key file (test-instance.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 test-instance.pem
```

4. Connect to your instance using its Public DNS:

```
ec2-52-39-116-59.us-west-2.compute.amazonaws.com
```

Example:

```
ssh -i "test-instance.pem" ubuntu@ec2-52-39-116-59.us-west-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.

This shows 2 options to connect to the instance. First is to use a standalone SSH client like Putty. Second is to use the browser directly. Since I have a Mac OS X system, I can simple use my terminal to connect using ssh. If you are using windows, you can install Putty or use the second option.

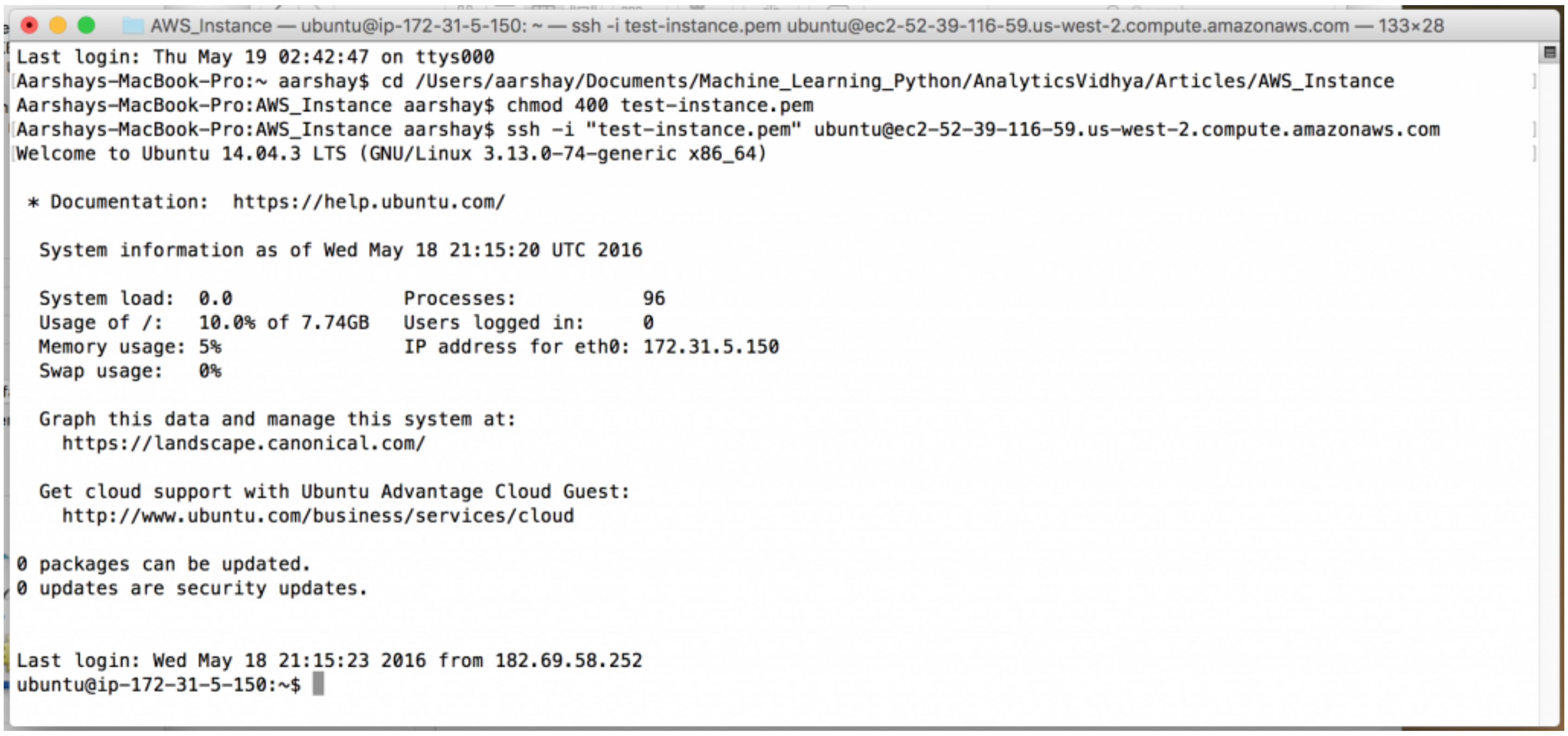
Now our instance is up and running. Now lets run some Python code on that instance.

3. How to make a first model on an AWS instance?

First, I'll take you through the steps I followed to connect to the instance using my Mac terminal.

1. Choose the folder containing the downloaded .pem file as the current directory
2. Run the command on step #3 of the connection page above – **chmod 400 test-instance.pem**
3. Run the command on step #4 of the connection page above – **ssh -i "test-instance.pem" ubuntu@ec2-52-39-116-59.us-west-2.compute.amazonaws.com**

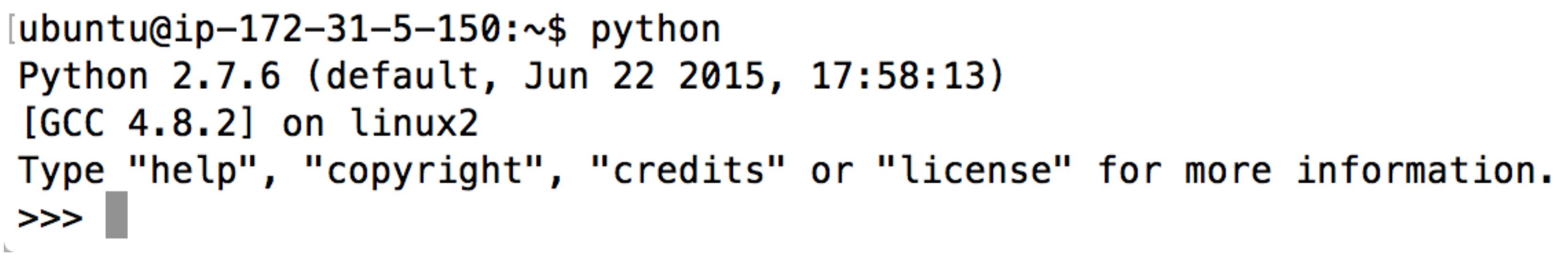
Now we'll go into the terminal of the ubuntu instance. A snapshot of my terminal with all the above steps is as following:


AWS_Instance — ubuntu@ip-172-31-5-150: ~ — ssh -i test-instance.pem ubuntu@ec2-52-39-116-59.us-west-2.compute.amazonaws.com — 133×28

```
Last login: Thu May 19 02:42:47 on ttys000
Aarshays-MacBook-Pro:~ aarshay$ cd /Users/aarshay/Documents/Machine_Learning_Python/AnalyticsVidhya/Articles/AWS_Instance
Aarshays-MacBook-Pro:AWS_Instance aarshay$ chmod 400 test-instance.pem
Aarshays-MacBook-Pro:AWS_Instance aarshay$ ssh -i "test-instance.pem" ubuntu@ec2-52-39-116-59.us-west-2.compute.amazonaws.com
Welcome to Ubuntu 14.04.3 LTS (GNU/Linux 3.13.0-74-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

  System information as of Wed May 18 21:15:20 UTC 2016

  System load:  0.0               Processes:           96
  Usage of /:   10.0% of 7.74GB   Users logged in:     0
  Memory usage: 5%                IP address for eth0: 172.31.5.150
  Swap usage:   0%

  Graph this data and manage this system at:
    https://landscape.canonical.com/

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

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


Last login: Wed May 18 21:15:23 2016 from 182.69.58.252
ubuntu@ip-172-31-5-150:~$ 
```

You can see that in the end, we get into the AWS instance and can access its terminal. Now we can easily run some python codes in it.

Lets start by getting into python using the command – **python** as shown:



```
ubuntu@ip-172-31-5-150:~$ python
Python 2.7.6 (default, Jun 22 2015, 17:58:13)
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

Remember that this is like a brand new system so we have to install everything from scratch. Lets start by installing pip which can be used to install other packages. This can be done using the command:

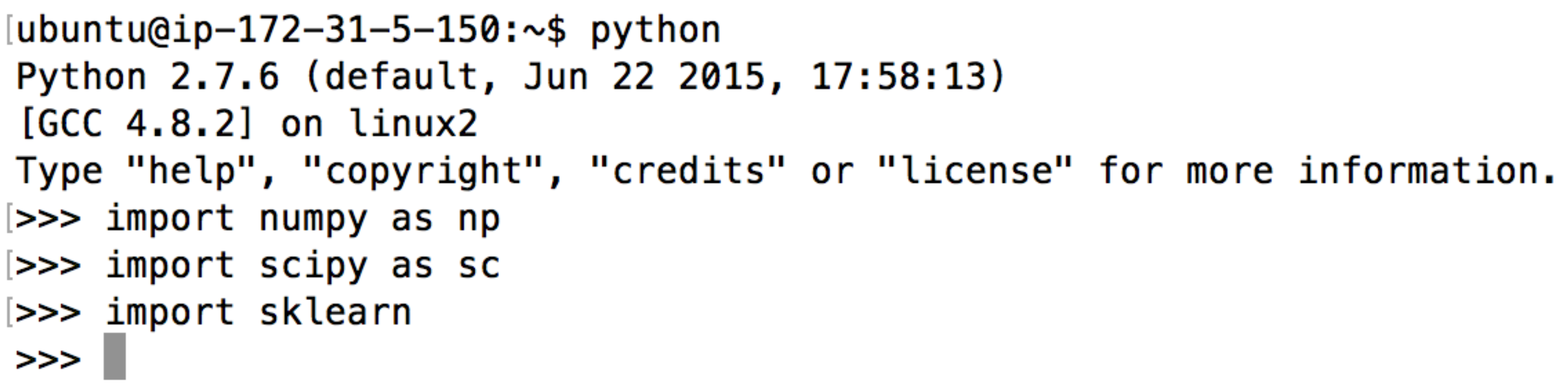
```
#Install setuptools:
wget https://bootstrap.pypa.io/ez_setup.py -O - | python - --user
sudo python setup.py install

#Install pip:
wget http://pypi.python.org/packages/source/p/pip/pip-1.1.tar.gz#md5=62a9f08dd5dc69d7673
tar -xvf pip*.gz
cd pip*
sudo apt-get install python-pip
```

These codes will successfully install pip. Now we can install numpy, scipy and sklearn using the following commands:

```
pip install numpy

pip install scipy

pip install sklearn
```

These commands will install the respective libraries. If you get permission denied issues, you can add **sudo (super used do)** in front of these which should solve the issue. Note that here I'm just doing this for demonstration so this works. But in practice, you might want to create some virtual environment where your code goes so that it doesn't affect the other code.

Lets enter into python now and import these libraries to be sure that they are installed.

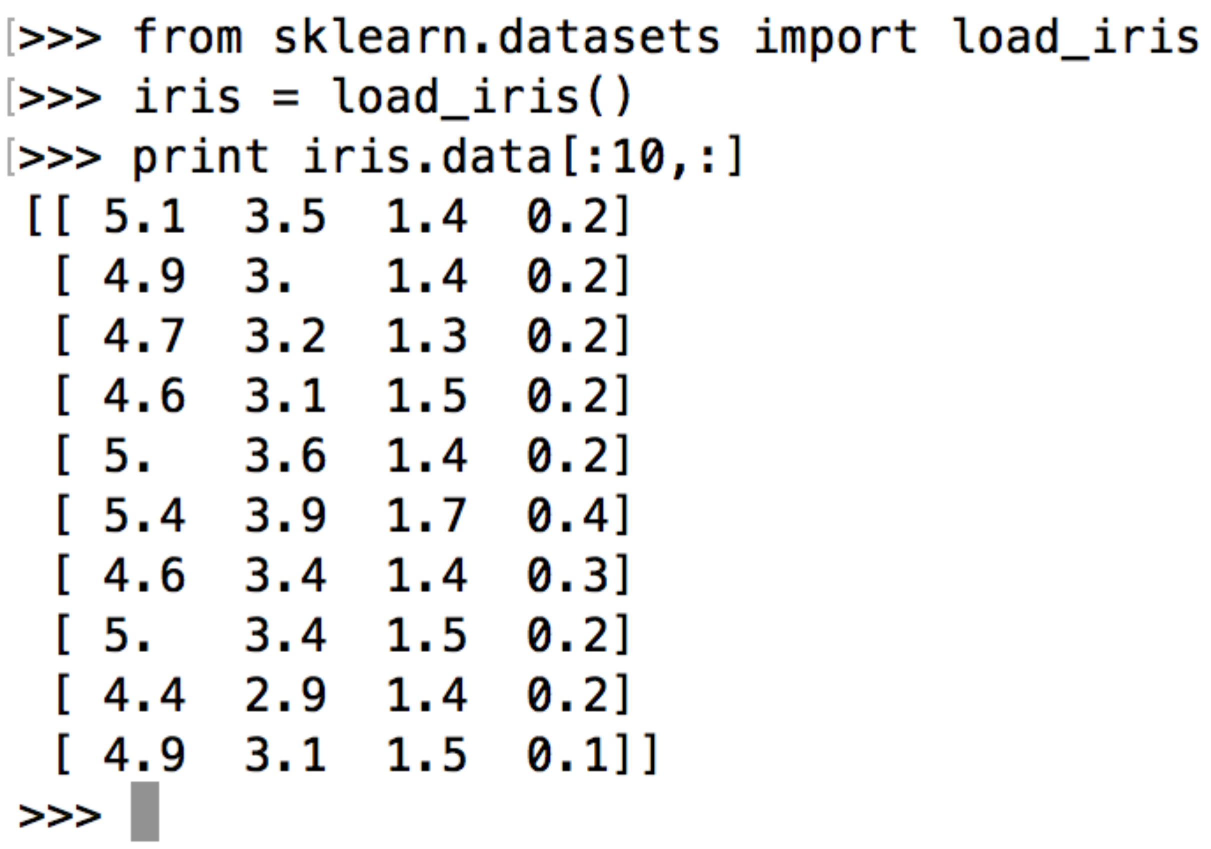


```
[ubuntu@ip-172-31-5-150:~$ python
 Python 2.7.6 (default, Jun 22 2015, 17:58:13)
 [GCC 4.8.2] on linux2
 Type "help", "copyright", "credits" or "license" for more information.
[>>> import numpy as np
[>>> import scipy as sc
[>>> import sklearn
 >>>
```

Now lets load the iris dataset as present in sklearn and make a decision tree classifier on it.

```
from sklearn.datasets import load_iris
iris = load_iris()
```

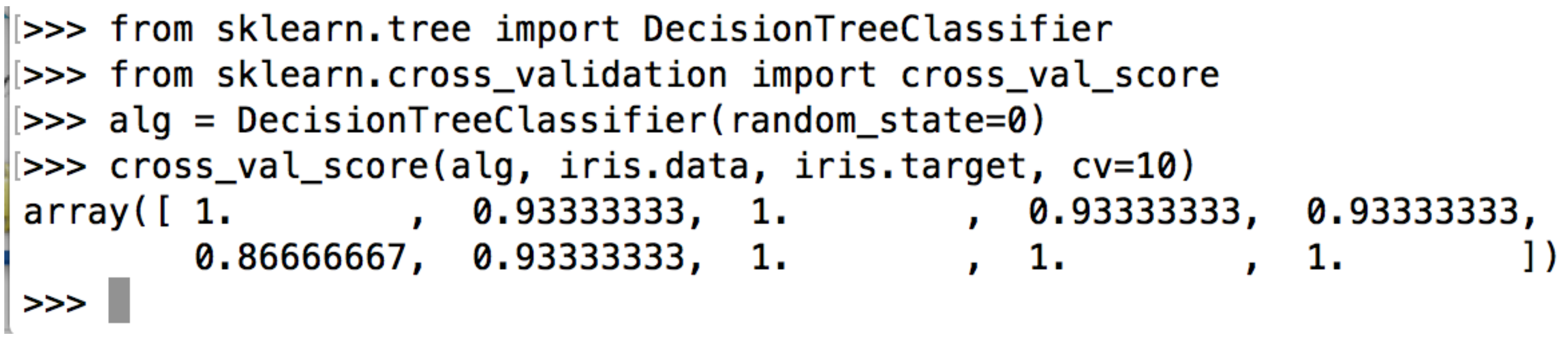
```
print iris.data[:10,:]
```



```
>>> from sklearn.datasets import load_iris
>>> iris = load_iris()
>>> print iris.data[:10,:]
[[ 5.1  3.5  1.4  0.2]
 [ 4.9  3.   1.4  0.2]
 [ 4.7  3.2  1.3  0.2]
 [ 4.6  3.1  1.5  0.2]
 [ 5.   3.6  1.4  0.2]
 [ 5.4  3.9  1.7  0.4]
 [ 4.6  3.4  1.4  0.3]
 [ 5.   3.4  1.5  0.2]
 [ 4.4  2.9  1.4  0.2]
 [ 4.9  3.1  1.5  0.1]]
>>>
```

Now we can make a quick decision tree on it and analyze the cross validation score.

```
from sklearn.cross_validation import cross_val_score
alg = DecisionTreeClassifier(random_state=0)
cross_val_score(clf, iris.data, iris.target, cv=10)
```



```
>>> from sklearn.tree import DecisionTreeClassifier
>>> from sklearn.cross_validation import cross_val_score
>>> alg = DecisionTreeClassifier(random_state=0)
>>> cross_val_score(alg, iris.data, iris.target, cv=10)
array([ 1.        ,  0.93333333,  1.        ,  0.93333333,  0.93333333,
        0.86666667,  0.93333333,  1.        ,  1.        ,  1.        ])
>>>
```

So we have successfully created our first model on an AWS instance and that too for free! I will give this tutorial a halt at this point. Remember that you should stop your instance when not in use else AWS will keep on charging you. Stay tuned for another article where I will take a more detailed practical application with lots of data and train some big models using AWS instances.

End Notes

In this article, we began by getting acquainted to the concepts and terminologies required for understanding the AWS system. Then we moved on to setting up our own AWS instance for free! The journey was clearly laid out with screenshots of the entire process.

Finally, we installed the basic python libraries including numpy and sklearn and made our first model on the IRIS dataset. This was a very high level introduction to AWS but I've tried to stick to the fundamentals.

If you follow this article, I'm sure you'll get a feel of the system which will help you in hardcore practical applications in future. Did you like reading this article ? Do share your experience / suggestions in the comments section below. I'd love to know your experience working on AWS.

You can test your skills and knowledge. Check out Live Competitions and compete with best Data Scientists from all over the world.