Deepak Sanjay Prasad

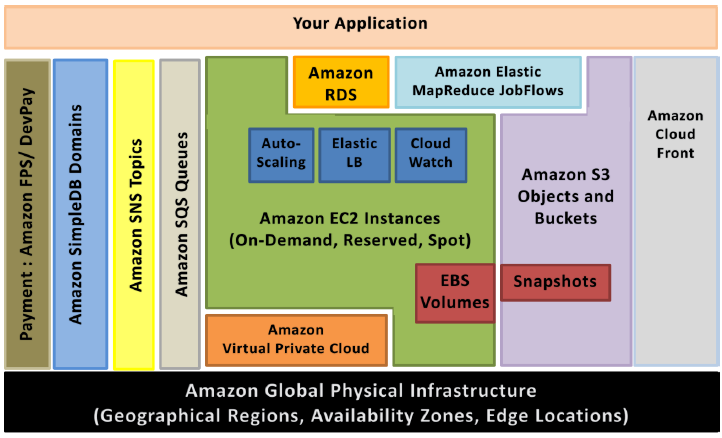
385 TYCS A

Practical No. 10

AIM: Case study on Amazon EC2/Microsoft Azure/Google Cloud Platform.

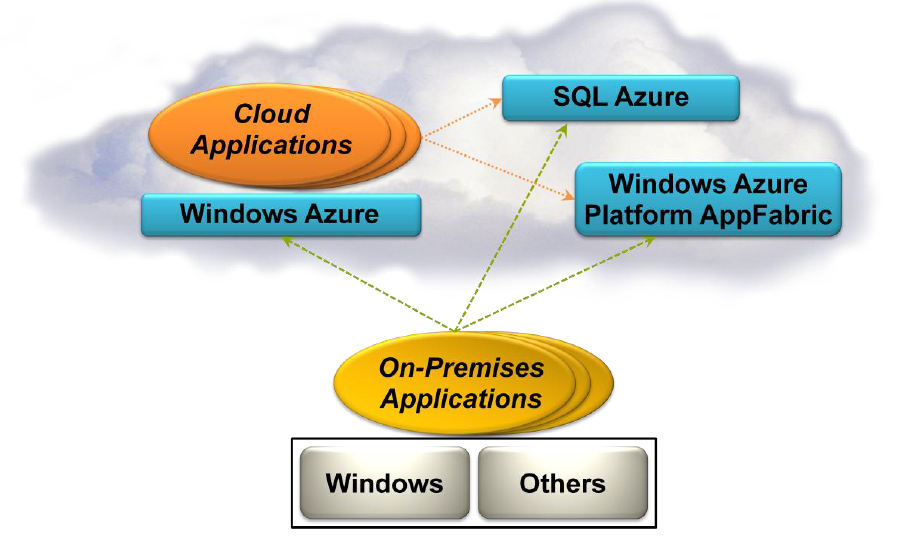
Solution:

AMAZON EC2



* Elastic IP addresses allow you to allocate a static IP address and programmatically assign it to an instance. You can enable monitoring on an Amazon EC2 instance using Amazon CloudWatch2 in order to gain visibility into resource utilization, operational performance, and overall demand patterns (including metrics such as CPU utilization, disk reads and writes, and network traffic). You can create Auto-scaling Group using the Auto-scaling feature3 to automatically scale our capacity on certain conditions based on metric that Amazon CloudWatch collects. You can also distribute incoming traffic by creating an elastic load balancer using the Elastic Load Balancing4 service. Amazon Elastic Block Storage (EBS)5 volumes provide network-attached persistent storage to Amazon EC2 instances. Point-in-time consistent snapshots of EBS volumes can be created and stored on Amazon Simple Storage Service (Amazon S3)6.  
  Amazon S3 is highly durable and distributed data store. With a simple web services interface, we can store and retrieve large amounts of data as objects in buckets (containers) at any time, from anywhere on the web using standard HTTP verbs. Copies of objects can be distributed and cached at 14 edge locations around the world by creating a distribution using Amazon CloudFront7 service – a web service for content delivery (static or streaming content). Amazon SimpleDB8 is a web service that provides the core functionality of a database- real-time lookup and simple querying of structured data – without the operational complexity. You can organize the dataset into domains and can run queries across all of the data stored in a particular domain. Domains are collections of items that are described by attribute-value pairs.
* Amazon Relational Database Service9 (Amazon RDS) provides an easy way to setup, operate and scale a relational database in the cloud. We can launch a DB Instance and get access to a full-featured MySQL database and not worry about common database administration tasks like backups, patch management etc.  
  Amazon Simple Queue Service (Amazon SQS)10 is a reliable, highly scalable, hosted distributed queue for storing messages as they travel between computers and application components.
* Amazon Simple Notifications Service (Amazon SNS) provides a simple way to notify applications or people from the cloud by creating Topics and using a publish-subscribe protocol.
* Amazon Elastic MapReduce provides a hosted Hadoop framework running on the web-scale infrastructure of Amazon Elastic Compute Cloud (Amazon EC2) and Amazon Simple Storage Service (Amazon S3) and allows you to create customized JobFlows. JobFlow is a sequence of MapReduce steps.
* Amazon Virtual Private Cloud (Amazon VPC) allows you to extend your corporate network into a private cloud contained within AWS. Amazon VPC uses IPSec tunnel mode that enables you to create a secure connection between a gateway in your data center and a gateway in AWS.
* Amazon Route53 is a highly scalable DNS service that allows you manage your DNS records by creating a HostedZone for every domain you would like to manage.
* AWS Identity and Access Management (IAM) enable you to create multiple Users with unique security credentials and manage the permissions for each of these Users within your AWS Account. IAM is natively integrated into AWS Services. No service APIs have changed to support IAM, and exiting applications and tools built on top of the AWS service APIs will continue to work when using IAM.  
  AWS also offers various payment and billing services that leverages Amazon’s payment infrastructure.
* All AWS infrastructure services offer utility-style pricing that require no long-term commitments or contracts. For example, you pay by the hour for Amazon EC2 instance usage and pay by the gigabyte for storage and data transfer in the case of Amazon S3. More information about each of these services and their pay-as-you-go pricing is available on the AWS Website.  
  Note that using the AWS cloud doesn’t require sacrificing the flexibility and control you’ve grown accustomed to:  
  You are free to use the programming model, language, or operating system (Windows, OpenSolaris or any flavor of Linux) of your choice.  
  You are free to pick and choose the AWS products that best satisfy your requirements—you can use any of the services individually or in any combination.  
  Because AWS provides resizable (storage, bandwidth and computing) resources, you are free to consume as much or as little and only pay for what you consume.

Microsoft Azure



### Execution Environment

The Windows Azure execution environment consists of a platform for applications and services hosted within one or more roles. The types of roles you can implement in Windows Azure are:

Azure Compute (Web and Worker Roles). A Windows Azure application consists of one or more hosted roles running within the Azure data centers. Typically there will be at least one Web role that is exposed for access by users of the application. The application may contain additional roles, including Worker roles that are typically used to perform background processing and support tasks for Web roles. For more detailed information see “Overview of Creating a Hosted Service for Windows Azure” at<http://technet.microsoft.com/en-au/library/gg432976.aspx> and “Building an Application that Runs in a Hosted Service” at <http://technet.microsoft.com/en-au/library/hh180152.aspx>.

Virtual Machine (VM role). This role allows you to host your own custom instance of the Windows Server 2008 R2 Enterprise or Windows Server 2008 R2 Standard operating system within a Windows Azure data center. For more detailed information see “Creating Applications by Using a VM Role in Windows Azure” at<http://technet.microsoft.com/en-au/library/gg465398.aspx>.

### Data Management

Windows Azure, SQL Azure, and the associated services provide opportunities for storing and managing data in a range of ways. The following data management services and features are available:

Azure Storage: This provides four core services for persistent and durable data storage in the cloud. The services support a REST interface that can be accessed from within Azure-hosted or on-premises (remote) applications. For information about the REST API, see “Windows Azure Storage Services REST API Reference” at<http://msdn.microsoft.com/en-us/library/dd179355.aspx>. The four storage services are:

The Azure Table Service provides a table-structured storage mechanism based on the familiar rows and columns format, and supports queries for managing the data. It is primarily aimed at scenarios where large volumes of data must be stored, while being easy to access and update. For more detailed information see “Table Service Concepts” at <http://msdn.microsoft.com/en-us/library/dd179463.aspx> and “Table Service API” at <http://msdn.microsoft.com/en-us/library/dd179423.aspx>.

The Binary Large Object (BLOB) Service provides a series of containers aimed at storing text or binary data. It provides both Block BLOB containers for streaming data, and Page BLOB containers for random read/write operations. For more detailed information see “Understanding Block Blobs and Page Blobs” at<http://msdn.microsoft.com/en-us/library/ee691964.aspx> and “Blob Service API” at <http://msdn.microsoft.com/en-us/library/dd135733.aspx>.

The Queue Service provides a mechanism for reliable, persistent messaging between role instances, such as between a Web role and a Worker role. For more detailed information see “Queue Service Concepts” at <http://msdn.microsoft.com/en-us/library/dd179353.aspx> and “Queue Service API” at<http://msdn.microsoft.com/en-us/library/dd179363.aspx>.

Windows Azure Drives provide a mechanism for applications to mount a single volume NTFS VHD as a Page BLOB, and upload and download VHDs via the BLOB. For more detailed information see “Windows Azure Drive” (PDF) at <http://go.microsoft.com/?linkid=9710117>.

SQL Azure Database: This is a highly available and scalable cloud database service built on SQL Server technologies, and supports the familiar T-SQL based relational database model. It can be used with applications hosted in Windows Azure, and with other applications running on-premises or hosted elsewhere. For more detailed information see “SQL Azure Database” at <http://msdn.microsoft.com/en-us/library/ee336279.aspx>.

Data Synchronization: SQL Azure Data Sync is a cloud-based data synchronization service built on Microsoft Sync Framework technologies. It provides bi-directional data synchronization and data management capabilities allowing data to be easily shared between multiple SQL Azure databases and between on-premises and SQL Azure databases. For more detailed information see “Microsoft Sync Framework Developer Center” at <http://msdn.microsoft.com/en-us/sync>.

Caching: This service provides a distributed, in-memory, low latency and high throughput application cache service that requires no installation or management, and dynamically increases and decreases the cache size automatically as required. It can be used to cache application data, ASP.NET session state information, and for ASP.NET page output caching. For more detailed information see “Caching Service (Windows Azure AppFabric)” at <http://msdn.microsoft.com/en-us/library/gg278356.aspx>.

### Networking Services

Windows Azure provides several networking services that you can take advantage of to maximize performance, implement authentication, and improve manageability of your hosted applications. These services include the following:

Content Delivery Network (CDN). The CDN allows you to cache publicly available static data for applications at strategic locations that are closer (in network delivery terms) to end users. The CDN uses a number of data centers at many locations around the world, which store the data in BLOB storage that has anonymous access. These do not need to be locations where the application is actually running. For more detailed information see “Delivering High-Bandwidth Content with the Windows Azure CDN” at<http://msdn.microsoft.com/en-us/library/ee795176.aspx>.

Virtual Network Connect. This service allows you to configure roles of an application running in Windows Azure and computers on your on-premises network so that they appear to be on the same network. It uses a software agent running on the on-premises computer to establish an IPsec-protected connection to the Windows Azure roles in the cloud, and provides the capability to administer, manage, monitor, and debug the roles directly. For more detailed information see “Connecting Local Computers to Windows Azure Roles” at <http://msdn.microsoft.com/en-us/library/gg433122.aspx>.

Virtual Network Traffic Manager. This is a service that allows you to set up request redirection and load balancing based on three different methods. Typically you will use Traffic Manager to maximize performance by redirecting requests from users to the instance in the closest data center using the Performance method. Alternative load balancing methods available are Failover and Round Robin. For more detailed information see “Windows Azure Traffic Manager” at <http://msdn.microsoft.com/en-us/WAZPlatformTrainingCourse_WindowsAzureTrafficManager>.

Access Control. This is a standards-based service for identity and access control that makes use of a range of identity providers (IdPs) that can authenticate users. ACS acts as a Security Token Service (STS), or token issuer, and makes it easier to take advantage of federation authentication techniques where user identity is validated in a realm or domain other than that in which the application resides. An example is controlling user access based on an identity verified by an identity provider such as Windows Live ID or Google. For more detailed information see “Access Control Service 2.0” at <http://msdn.microsoft.com/en-us/library/gg429786.aspx> and “Claims Based Identity & Access Control Guide” at <http://claimsid.codeplex.com/>.

Service Bus. This provides a secure messaging and data flow capability for distributed and hybrid applications, such as communication between Windows Azure hosted applications and on-premises applications and services, without requiring complex firewall and security infrastructures. It can use a range of communication and messaging protocols and patterns to provide delivery assurance, reliable messaging; can scale to accommodate varying loads; and can be integrated with on-premises BizTalk Server artifacts.

Google Cloud Platform

One of the most rewarding parts of working on [Google App Engine](http://cloud.google.com/products) is seeing our developers create groundbreaking new applications on top of our infrastructure.  To help our current and prospective users gain insight into the vast array of these applications, we recently added a section to the [Google Cloud Platform](http://cloud.google.com/) site with a [collection of case studies](http://cloud.google.com/customers/index.html).   
  
[Rovio](https://cloud.google.com/files/Rovio.pdf)  
Creator of the blockbuster “Angry Birds” game series used App Engine when creating web versions of their game.  They were able to create customized versions of their game in just 2 weeks using App Engine, allowing them to capitalize on opportunities to grow their business.  
  
[GetAround](https://cloud.google.com/files/getaround.pdf)  
TechCrunch Disrupt award-winning car sharing service used App Engine to build a marketplace connecting car owners to people looking to rent cars.  They scaled their product without adding additional staff.  
  
[MAG Interactive](https://cloud.google.com/files/MAG.pdf)  
Developer of mobile casual games, including the hit game Ruzzle, scaled their backend using App Engine.  They grew to over 5 million users, and experienced “no scalability issues at all.”  
  
[Nubbius](https://cloud.google.com/files/Nubbius.pdf)  
The Cloud Gate used App Engine to create nubbius, a software-as-a-service offering for lawyers to manage their workflow from anywhere.  They saved more than $130,000 per year while scaling rapidly.  
  
[RedBus](https://cloud.google.com/files/Redbus.pdf)  
Online travel agency used [Google BigQuery](https://cloud.google.com/products/big-query) to unify tens of thousands of bus schedules into a single booking operation.  They analyzed data sets as large as 2 TB in less than 30 seconds, and spent 80% less than they would have on a Hadoop infrastructure,  
  
This is a sample of the many case studies we have on our site.  Check out [cloud.google.com/customers](http://cloud.google.com/customers) to see the full list.  You can read about companies varying in size, industry, and use cases, who are using Google Cloud Platform to build their products and businesses.

# Launch a Linux Virtual Machine with Amazon EC2

[Amazon Elastic Compute Cloud (EC2)](https://aws.amazon.com/ec2/) is the Amazon Web Service you use to create and run virtual machines in the cloud. AWS calls these virtual machines 'instances'. This step-by-step guide will help you successfully launch a Linux virtual machine on Amazon EC2 within our [AWS Free Tier.](https://aws.amazon.com/free/)

## Step 1. Sign up for AWS

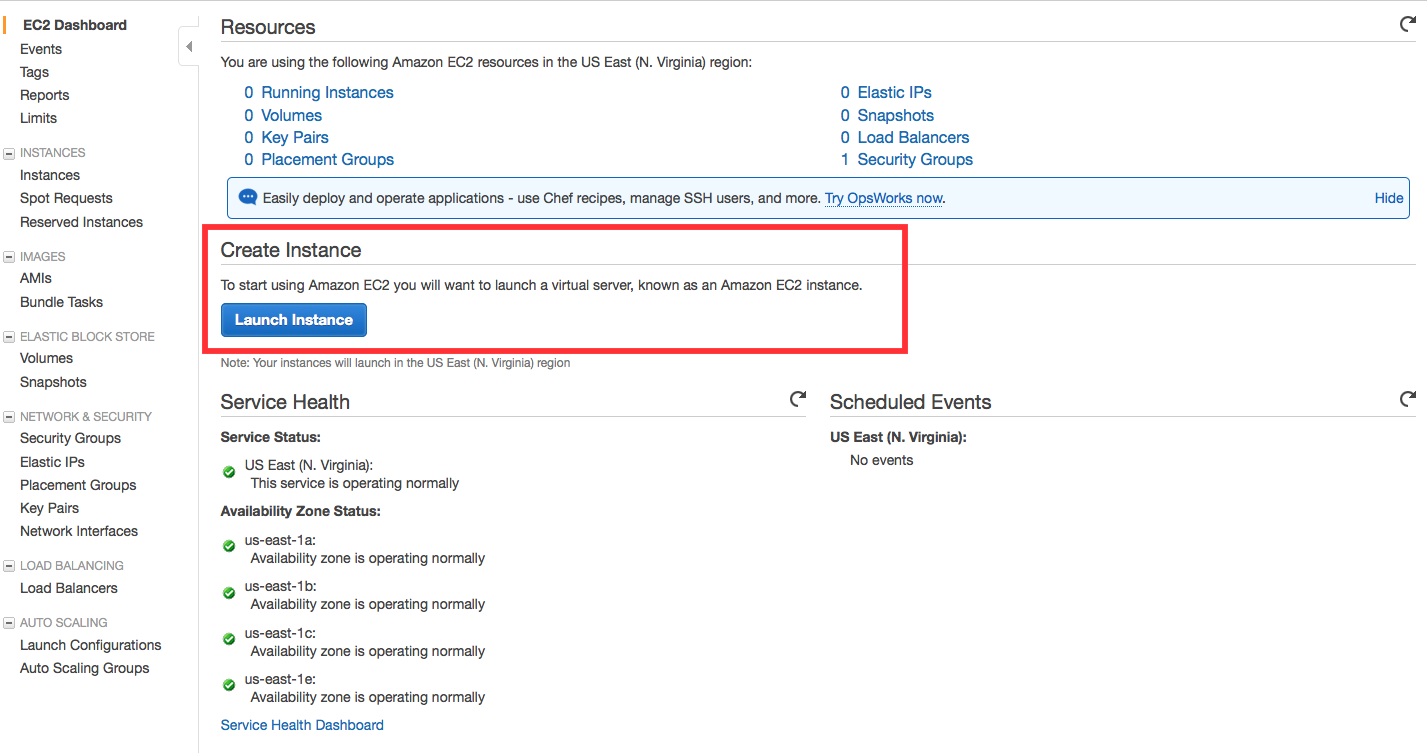
There are no additional charge for Amazon EC2 for this tutorial. The resources you create in this tutorial are Free Tier eligible. The button and the link open a new tab so you can follow this tutorial in the AWS console.

[Create a Free Account](https://console.aws.amazon.com/console/home)

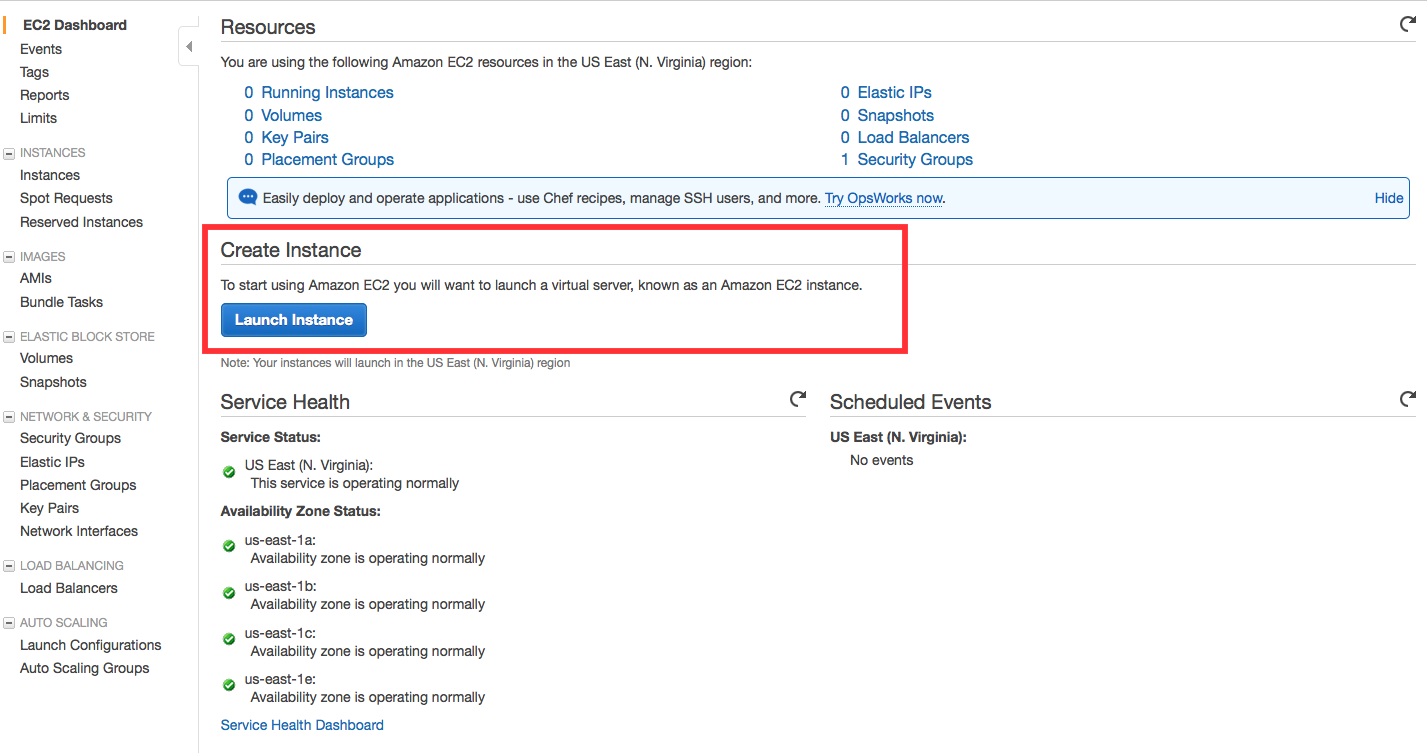
Already have an account? Sign-in

## Step 2. Launch an Amazon EC2 Instance

a.  [Open the AWS Management Console](https://console.aws.amazon.com/console/home), so you can keep this step-by-step guide open. When the screen loads, enter your user name and password to get started. Then type *EC2* in the search bar and select Amazon EC2 to open the service console.



b. Select Launch Instance to create and configure your virtual machine.

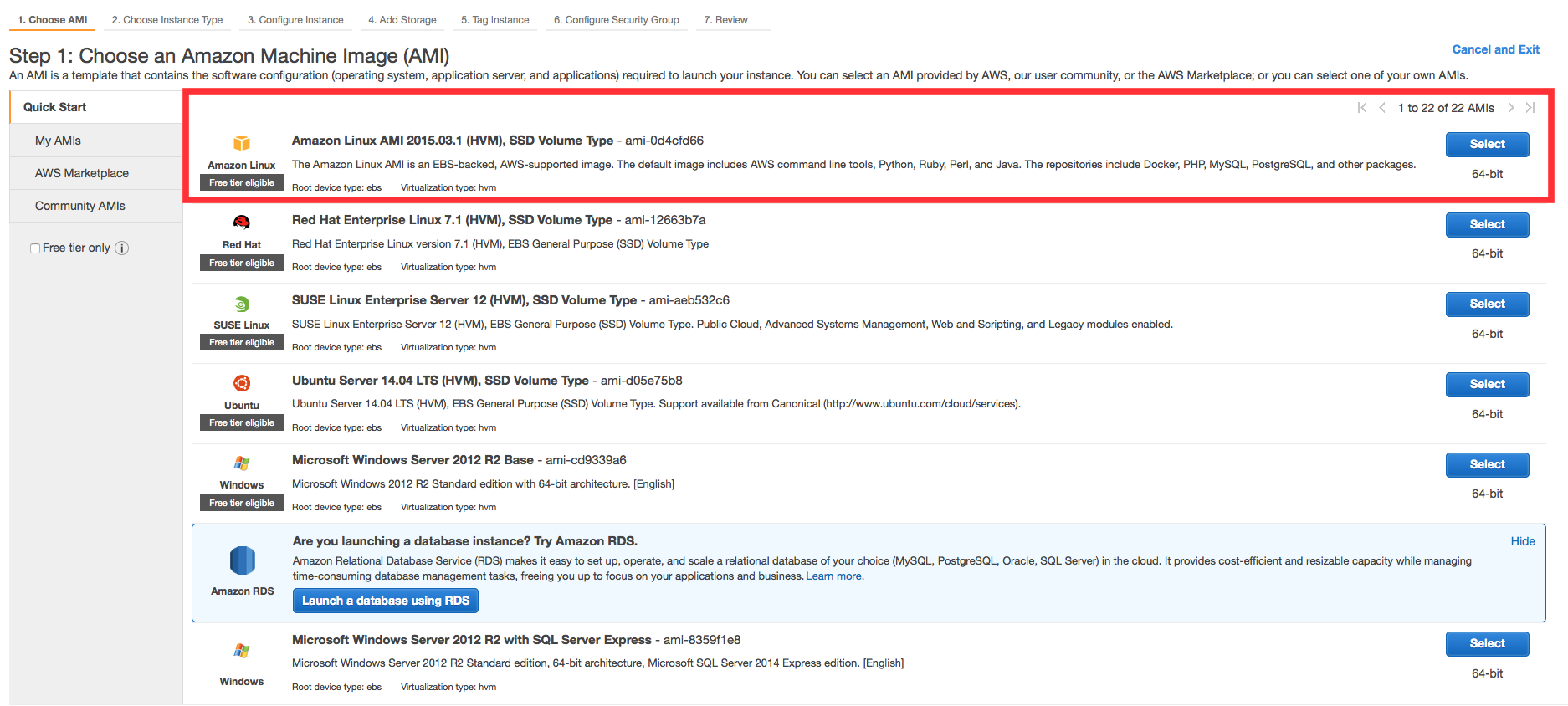


## Step 3. Configure your Instance

You are now in the EC2 Launch Instance Wizard, which will help you configure and launch your instance.

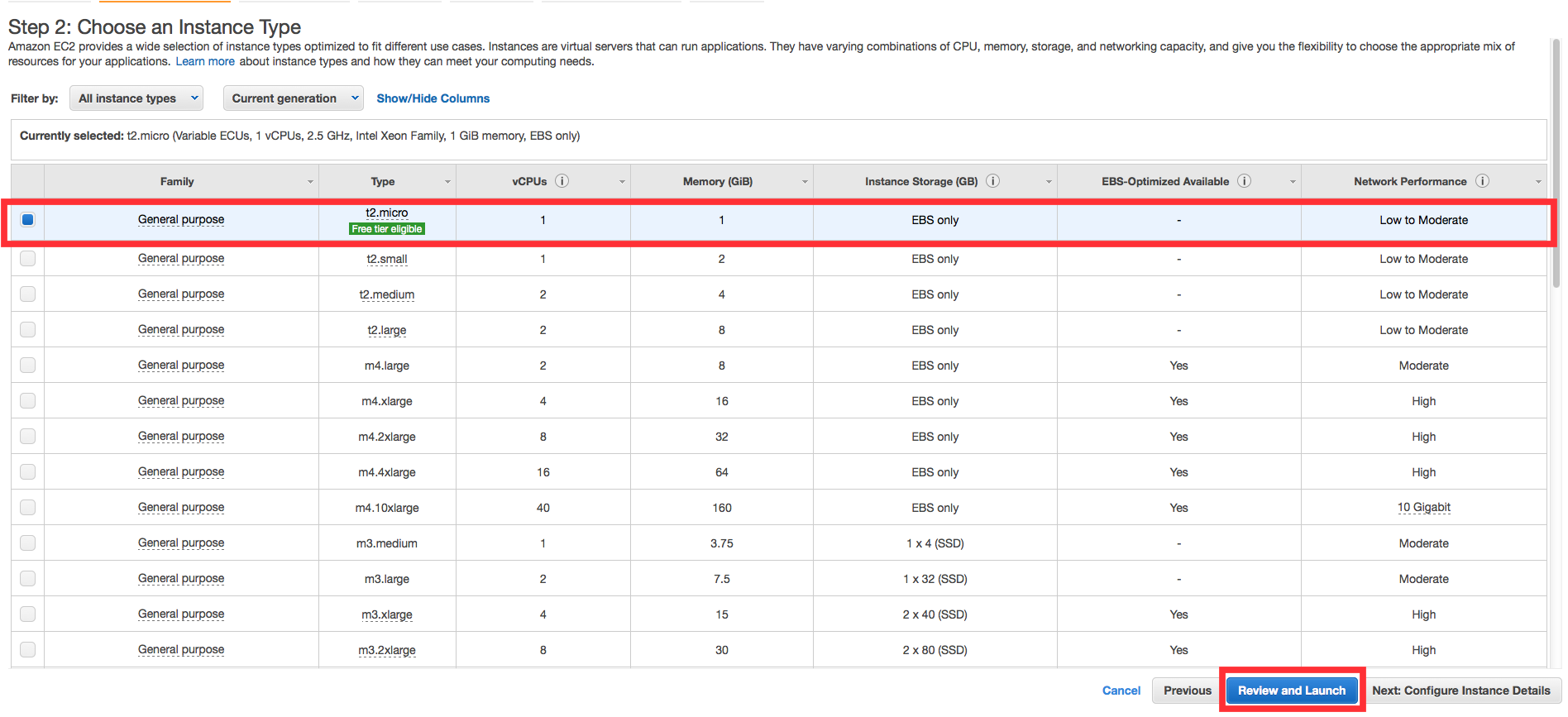
a.  In this screen, you are shown options to choose an Amazon Machine Image (AMI). AMIs are preconfigured server templates you can use to launch an instance. Each AMI includes an operating system, and can also include applications and application servers.

For this tutorial, find *Amazon Linux AMI* and click Select.



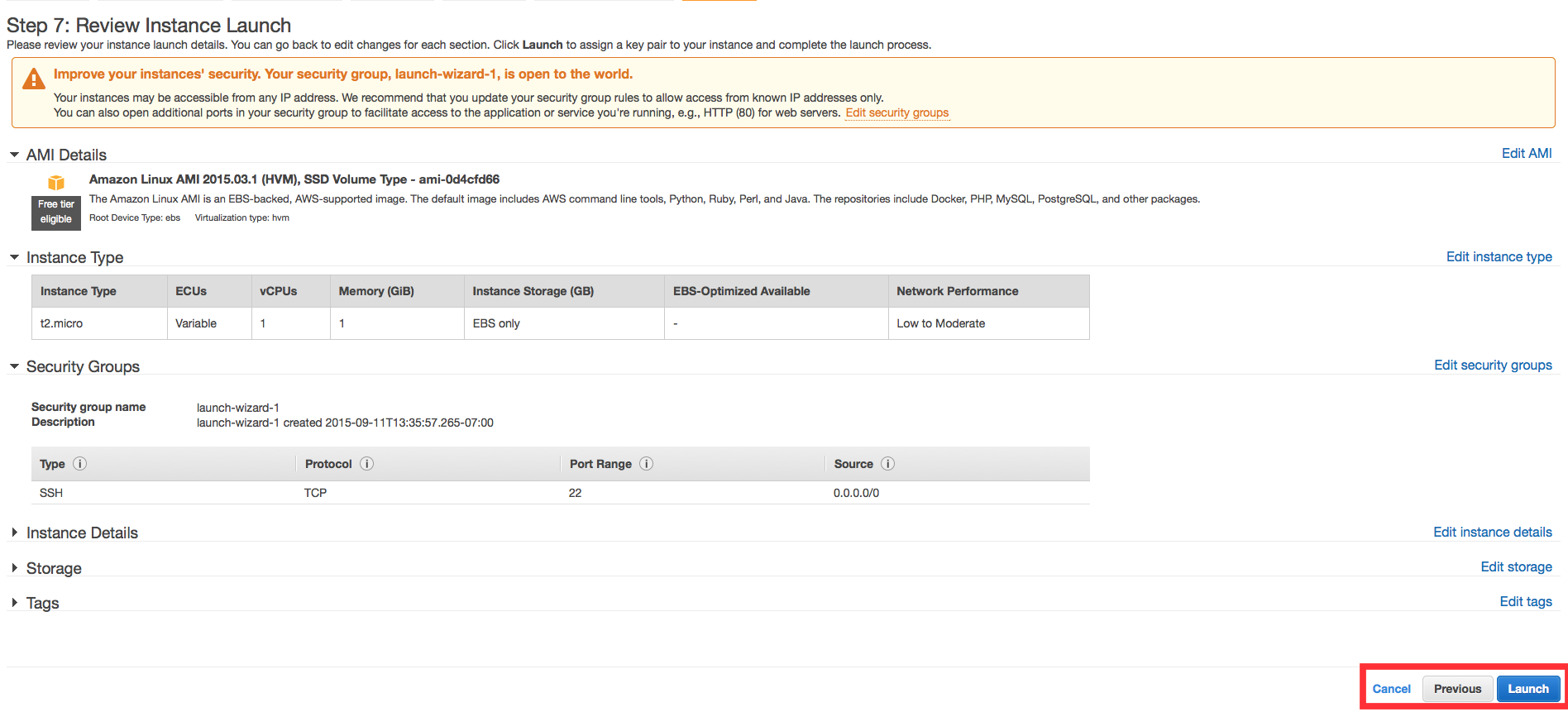
b. You will now choose an instance type. Instance types comprise of varying combinations of CPU, memory, storage, and networking capacity so you can choose the appropriate mix for your applications. For more information, see [Amazon EC2 Instance Types.](https://aws.amazon.com/ec2/instance-types/)

The default option of *t2.micro* should already be checked. This instance type is covered within the Free Tier and offers enough compute capacity to tackle simple workloads. Click Review and Launch at the bottom of the page.



c. You can review the configuration, storage, tagging, and security settings that have been selected for your instance. While you have the option to customize these settings, we recommend accepting the default values for this tutorial.

Click Launch at the bottom of the page.



d. On the next screen you will be asked to choose an existing key pair or create a new key pair. A key pair is used to securely access your Linux instance using SSH. AWS stores the public part of the key pair which is just like a house lock. You download and use the private part of the key pair which is just like a house key.

Select Create a new key pair and give it the name MyKeyPair. Next click the Download Key Pair button.

After you download the MyKeyPair key, you will want to store your key in a secure location. If you lose your key, you won't be able to access your instance. If someone else gets access to your key, they will be able to access your instance.

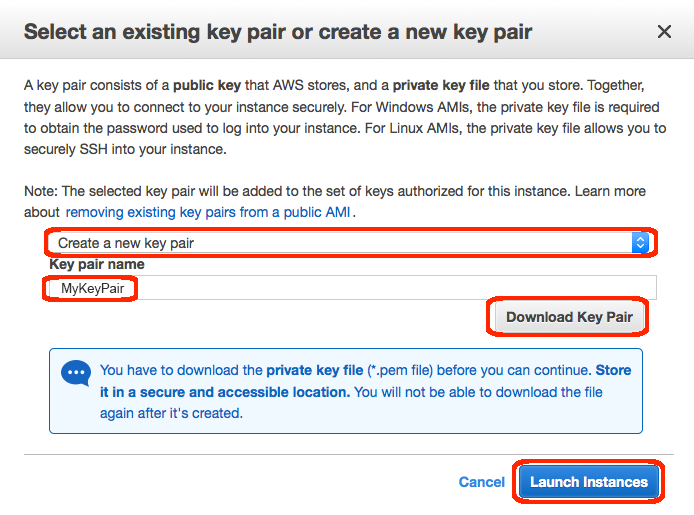
Windows users: We recommend saving your key pair in your user directory in a sub-directory called .ssh (ex. C:\user\{yourusername}\.ssh\MyKeyPair.pem).

Tip: You can't use Windows Explorer to create a folder with a name that begins with a period unless you also end the folder name with a period. After you enter the name (.ssh.), the final period is removed automatically.

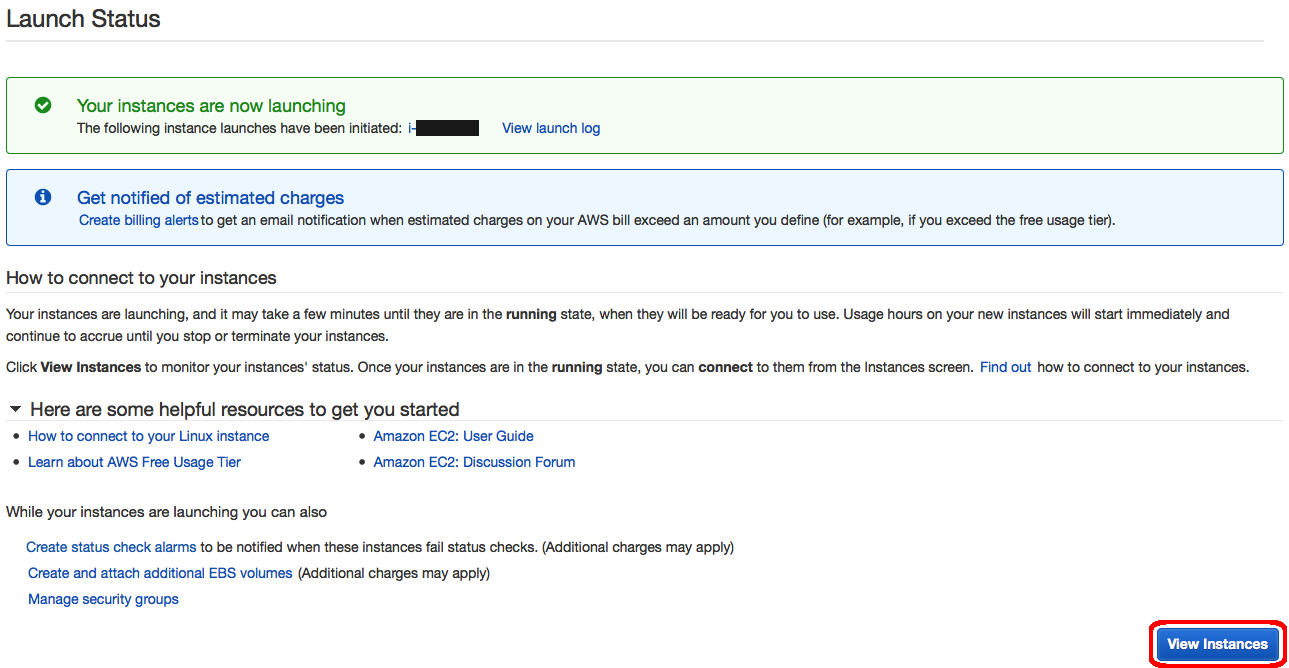
Mac/Linux users: We recommend saving your key pair in the .ssh sub-directory from your home directory (ex. ~/.ssh/MyKeyPair.pem).

Tip: On MacOS, the key pair is downloaded to your Downloads directory by default. To move the key pair into the .ssh sub-directory, enter the following command in a terminal window: mv ~/Downloads/MyKeyPair.pem ~/.ssh/MyKeyPair.pem

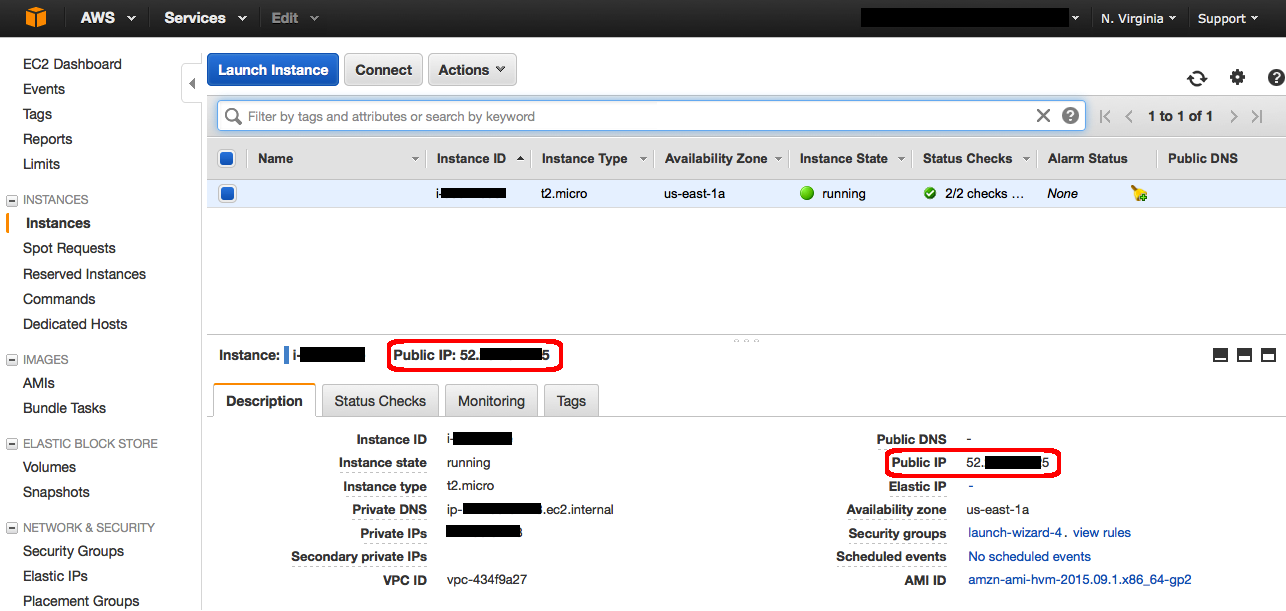
After you have stored your key pair, click Launch Instance to start your Linux instance.



e. Click View Instances on the next screen to view your instances and see the status of the instance you have just started.



f. In a few minutes, the *Instance State* column on your instance will change to "*running*" and a Public IP address will be shown. You can refresh these Instance State columns by pressing the refresh button on the right just above the table. Copy the Public IP address of your AWS instance, so you can use it when we connect to the instance using SSH in Step 4.



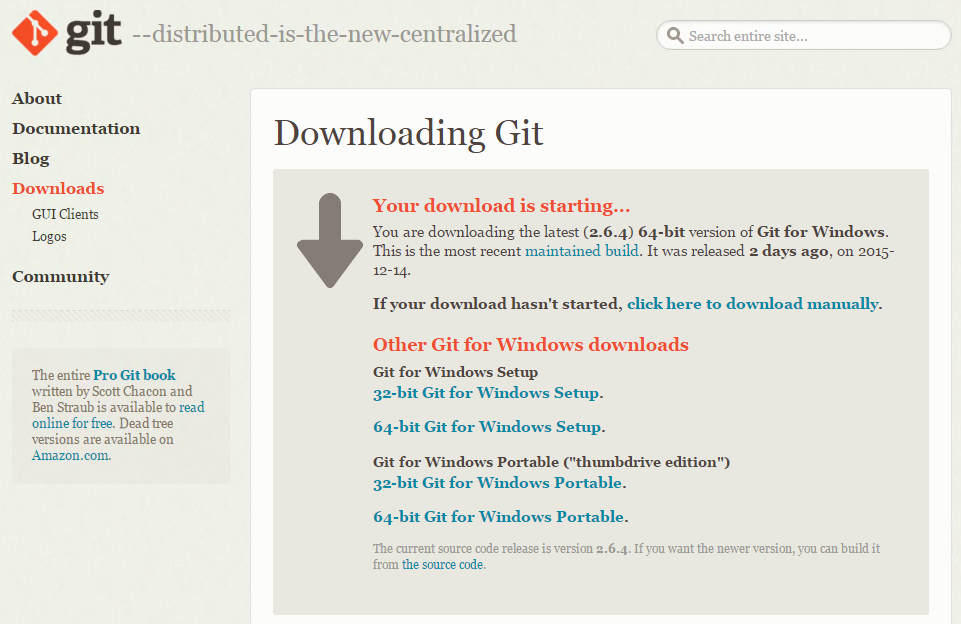
## Step 4. Connect to your Instance

After launching your instance, it's time to connect to it using SSH.

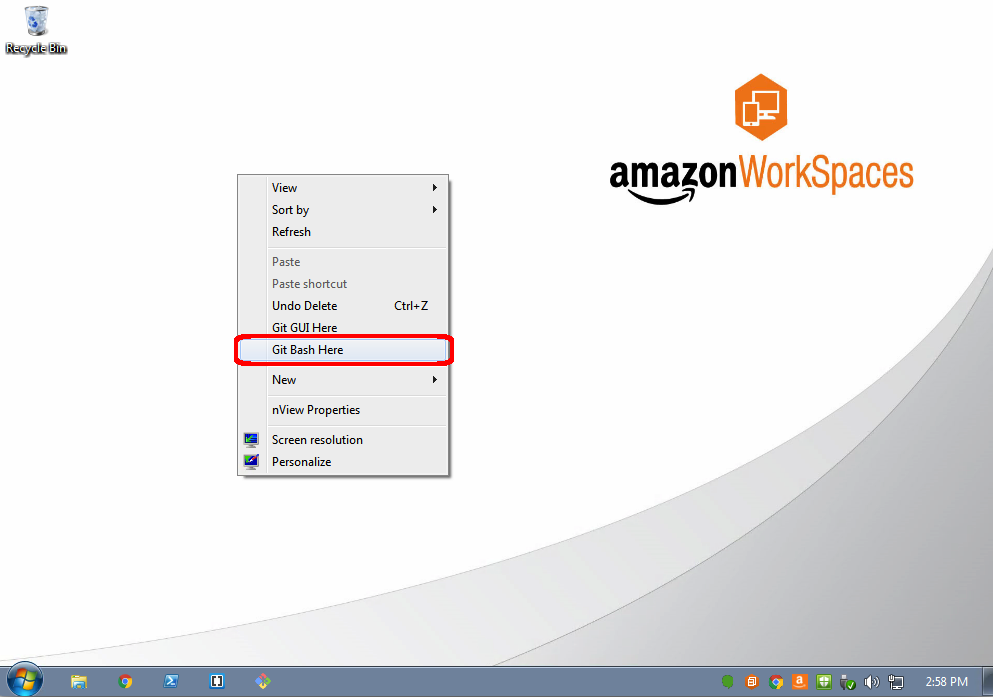
Windows users: Select Windows below to see instructions for installing Git Bash which includes SSH.

Mac/Linux user: Select Mac / Linux below to see instructions for opening a terminal window.

* Mac / Linux
* a. Download Git for Windows [here](https://git-scm.com/download/win). Run the downloaded installer accepting the default settings (this will install Git Bash as part of Git).



b. Right click on your desktop (not on an icon or file) and select Git Bash Here to open a Git Bash command prompt.



c. Use SSH to connect to your instance. In this case the user name is ec2-user, the SSH key is stored in the directory we saved it to in step 3 part d, and the IP address is from step 3 part f. The format is:

ssh -i {full path of your .pem file} ec2-user@{instance IP address}

Enter the following:

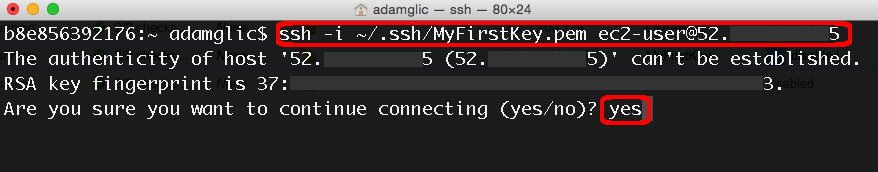
ssh -i 'c:\Users\yourusername\.ssh\MyKeyPair.pem' ec2-user@{IP\_Address}

Example: ssh -i 'c:\Users\adamglic\.ssh\MyKeyPair.pem' ec2-user@52.27.212.125

You'll see a response similar to the following:

The authenticity of host 'ec2-198-51-100-1.compute-1.amazonaws.com (10.254.142.33)' can't be established. RSA key fingerprint is 1f:51:ae:28:df:63:e9:d8:cf:38:5d:87:2d:7b:b8:ca:9f:f5:b1:6f. Are you sure you want to continue connecting (yes/no)?

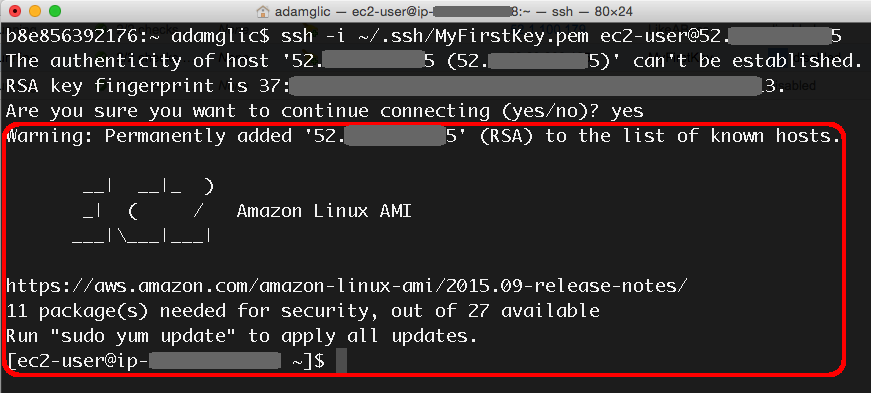
Type yes and press enter.



d. You'll see a response similar to the following:

Warning: Permanently added 'ec2-198-51-100-1.compute-1.amazonaws.com' (RSA) to the list of known hosts.

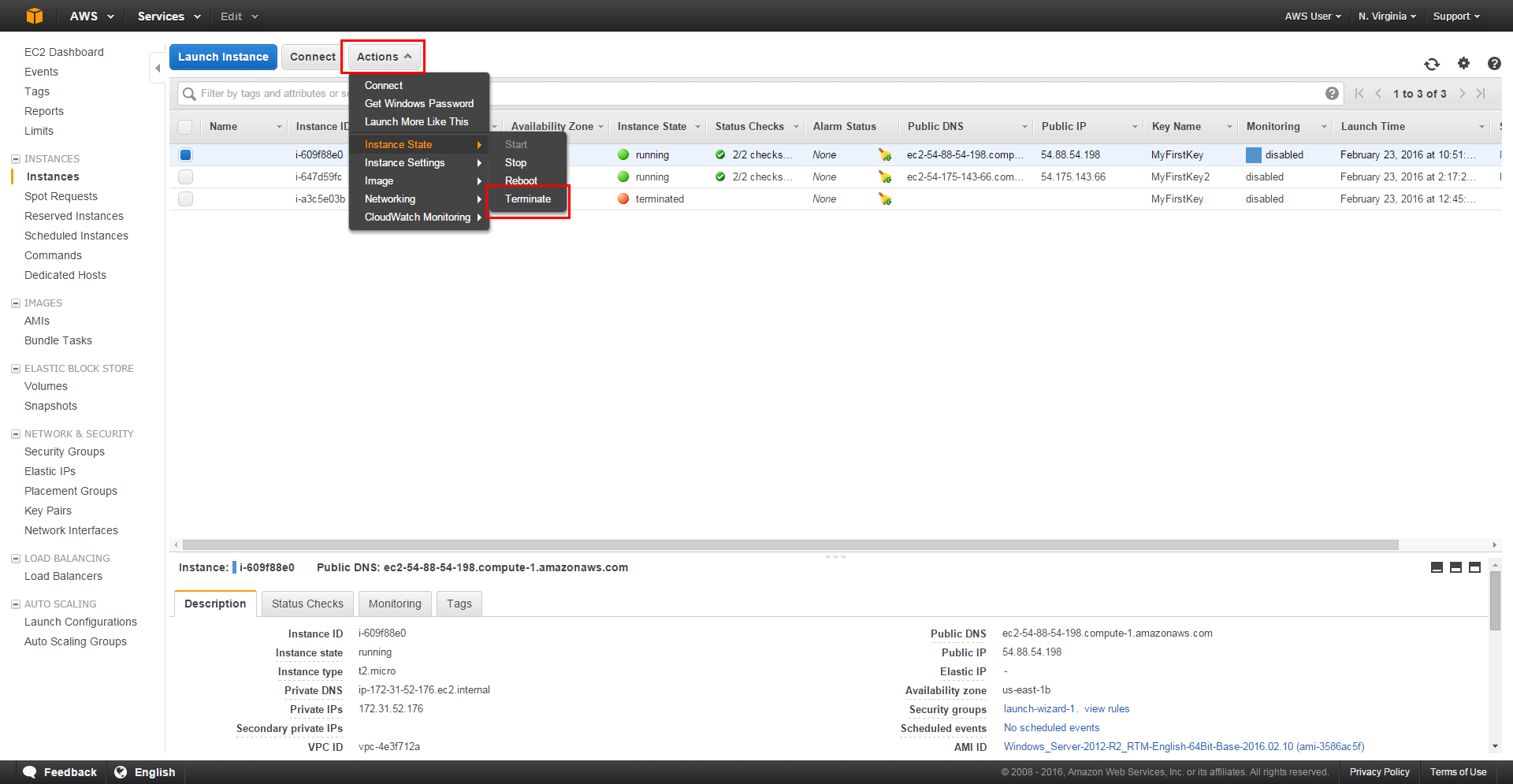
You should then see the welcome screen for your instance and you are now connected to your AWS Linux virtual machine in the cloud.



## Step 5. Terminate your Instance

You can easily terminate the instance from the EC2 console. In fact, it is a best practice to terminate instances you are no longer using so you don’t keep getting charged for them.

a.  Back on the EC2 Console, select the box next to the instance you created. Then click the Actions button, navigate to *Instance State*, and click Terminate.



b. You will be asked to confirm your termination - select Yes, Terminate.

Note: This process can take several seconds to complete.  Once your instance has been terminated, the Instance State will change to *terminated*on your EC2 Console.

