**EC2**

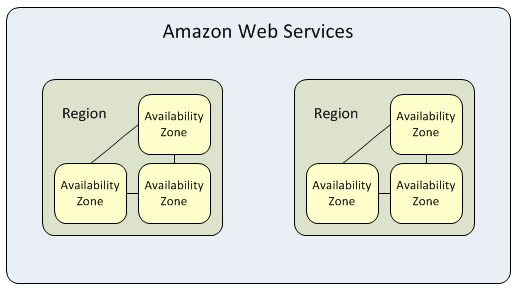
**What Is Amazon EC2 service ?**

Amazon Elastic Compute Cloud (Amazon EC2) is an Amazon web service that provides resizable (scalable) computing capacity in the cloud. You can use Amazon EC2 to launch as many virtual servers you need. In Amazon EC2 you can configure security and networking as well as manage storage.

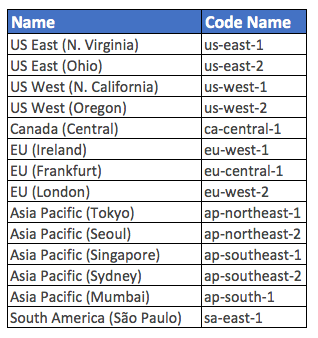
**What is region and availability zone in AWS?**

Amazon EC2 is hosted in multiple locations world-wide. These locations are composed of Regions and Availability Zones. Each *Region* is a separate geographic area. Each Region has multiple, isolated locations known as *Availability Zones*. Amazon EC2 provides you the ability to place resources, such as instances, and data in multiple locations. Resources aren't replicated across Regions unless you do so specifically.

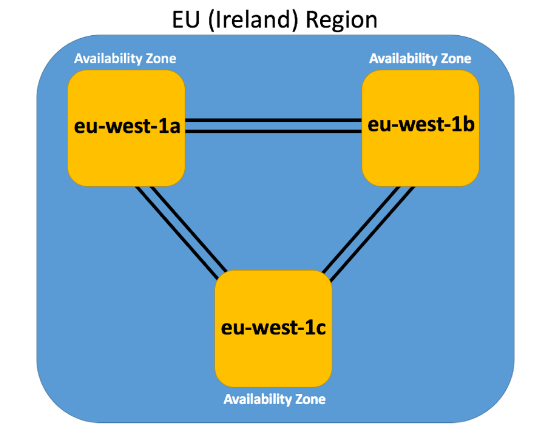
Each Region is completely independent. Each Availability Zone is isolated, but the Availability Zones in a Region are connected through low-latency links.



**Region naming convention:**



**Availability zone for each region:**



**What is AMI in EC2?**

* Amazon Machine Images known as AMIs is a comprehensive set that you require for your server (counting the operating system and extra software)
* Pre-configured templates for your instances (known as Amazon Machine Images – AMIs)
* Amazon Machine Images (AMIs) is a complete package that you need for your server (including the operating system and additional software)

**What is EC2 Root Device Volume?**

Root device volume is virtual hard disk where operating system is installed. This can be backed by EBS or Instance store (S3).

* When you launch an instance, the *root device volume* contains the image used to boot the instance mainly the operating system, all the configured services, and applications.
* By default, when you create an Amazon EC2 instance, it takes an EBS volume as a root device.
* What is EBS? Elastic Block Storage( EBS), as self-explanatory as it sounds, it is used to store data in blocks. An EBS Volumes can be up to 16TB in size. EBS volumes are like virtual hard disks which are attached to the server for storage. So by default Amazon EC2 instance, takes an EBS volume as a root device. You can, of course, add additional volumes for future purpose.
* Root Device — A good analogy of the term root device would be an operating system partition in a PC. A root device is a virtual device that holds the partition. It is a default boot device.

**What is AMI Type?**

All AMIs are categorized as either backed by EBS or backed by instance store.

When we launch EC2 instance we have option to select AMI image from EBS volume or from Instance store, and in Add Storage section root device shows if it is EBS type or S3 type. If we launch AMI from EBS volume, root device will be installed in EBS. If selected Instance store, it will be installed on S3.

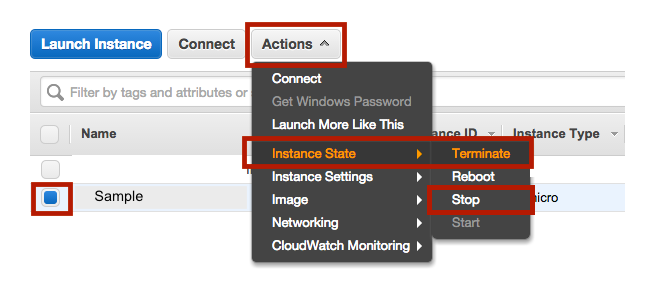
* **EBS-backed instances** – The root device is using EBS volume. You have both stop and terminate option. If you stop the instance, the Amazon EBS volume persists. If you restart the instance, the volume is automatically remounted, restoring the instance state and any stored data.
* **Instance store-backed instances or S3 backed**– The root device is temporary. You do not have option for stop, you can only terminate it. Once terminated, the data on the root device vanishes and cannot be recovered.

**How to identify whether you are using EBS backed AMI or not? Or how to determine the Root Device Type of Your AMI?**

**To determine the root device type of an AMI using the console**

1. Open the Amazon EC2 console.
2. In the navigation pane, click **AMIs**, and select the AMI.
3. Check the value of **Root Device Type** in the **Details** tab as follows:
   * If the value is ebs, this is an Amazon EBS-backed AMI.
   * If the value is instance store, this is an instance store-backed AMI.

**What is the difference between terminating and stopping an EC2 instance?**



**Stop Instance**

Amazon S3-backed AMIs can’t be stopped, they’re either running or terminated. You can stop an Amazon EBS backed instance, but not an S3-backed instance.

**If you stop an instance:**

* Root volume and additional EBS attached volume is preserved and it is not deleted and other devices that are attached to the instance keep on running.
* Once the Instance state shows “stopped,” you have the ability to scale up or down the instance. When it’s at stopped mode, you can fix some system issues and do some updates, change the user data, RAM disk and kernel. These attributes may only be changed if the instance has been stopped.
* You can start this again after clicking start, Instance State will show “pending”, and you know it’s ready when it shows “running”.
* Your private IP remains the same (so the EIP linked with the private IP is still connected with the instance)
* Gets a new public instance only if you don’t have an EIP

**Why stop an instance?**

* cannot perform an instance status check
* doesn’t run applications well
* to scale up or down
* to modify user data, kernel, RAM

**Terminate Instance**

* When you terminate an EC2 instance, the instance will be shutdown and the virtual machine that was provisioned for you will be permanently taken away and you will no longer be charged for instance usage.
* If EC2 instance is terminated, by default the root volume is also deleted. But the EBS volume attached will be retained. We can change this default behavior by modifying the DeleteOnTermination attribute. If you want to retain both the root volume and EBS volume, even after the instance is terminated, you must uncheck the DeleteOnTermination parameter while defining “Add Storage” during launching of EC2 instance.
* You cannot modify or connect to a terminated instance.
* You cannot restart this instance, but can always create a new one using the same AMI.
* The EIP associated to your instance will be disassociated.

**Why terminate an instance?**

* You have decided that you no longer need it.
* After you terminate an instance, you can still see it at the EC2 Dashboard, but this console entry will be deleted later on.

**What are the steps to launch instance in EC2?**

1 Choose AMI

2 Choose Instance Type - t2.micro

3 Configure Instance

4 Add Storage

5 Tag

6 Configure Security Group

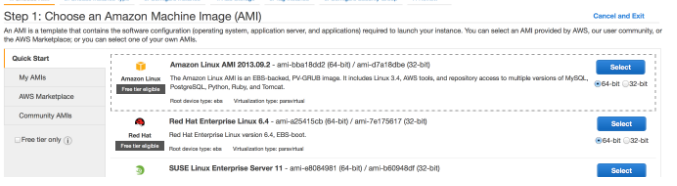
7 Review



**To launch an instance**

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation bar at the top of the screen, the current Region is displayed (for example, US East (Ohio)). Select a Region for the instance that meets your needs. This choice is important because some Amazon EC2 resources can be shared between Regions, while others can't.
3. From the Amazon EC2 console dashboard, choose Launch Instance.

**Choose AMI**



On the Choose an Amazon Machine Image (AMI) page, choose an AMI as follows:

Select the type of AMI to use in the left pane:

**Quick Start**

A selection of popular AMIs to help you get started quickly. To select an AMI that is eligible for the free tier, choose Free tier only in the left pane. These AMIs are marked Free tier eligible.

1. **My AMIs**

The private AMIs that you own, or private AMIs that have been shared with you. To view AMIs shared with you, choose Shared with me in the left pane.

1. **AWS Marketplace**

An online store where you can buy software that runs on AWS, including AMIs. For more information about launching an instance from the AWS Marketplace, see [Launching an AWS Marketplace Instance](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/launch-marketplace-console.html).

1. **Community AMIs**

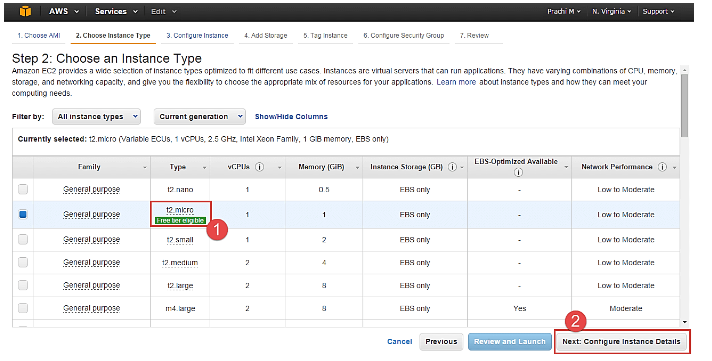
The AMIs that AWS community members have made available for others to use. To filter the list of AMIs by operating system, choose the appropriate check box under Operating system. You can also filter by architecture and root device type.

**Choose Instance Type**

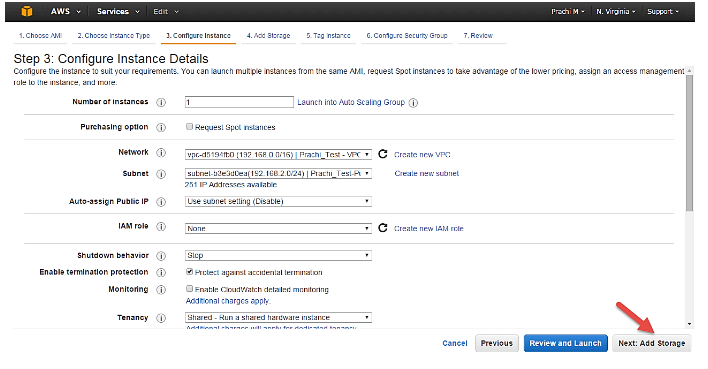
Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instance types comprise varying combinations of CPU, memory, storage, and networking capacity and give you the flexibility to choose the appropriate mix of resources for your applications. Each instance type includes one or more instance sizes, allowing you to scale your resources to the requirements of your target workload. t2.micro type is free.

General Purpose instances are the most widely used and are a good first step if you’re new to AWS or cloud computing in general. They offer a great mix of cost and functionality.

**Choose t2.micro instance type, which is a 1vCPU and 1GB memory server offered by AWS.**



**Configure Instance**

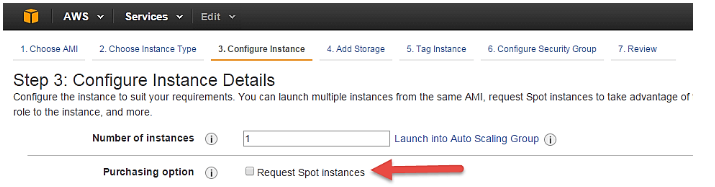


**Step 1) No. of instances:**



You can provision up to 20 instances at a time. Here we are launching one instance.

**Step 2) Purchasing Option:**

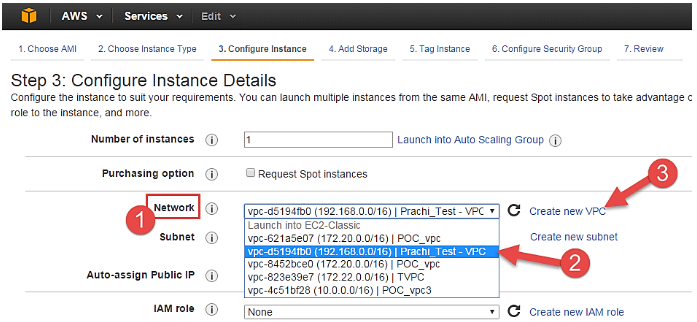


Under Purchasing Options, keep the option of 'Request Spot Instances' unchecked as of now. There are multiple Purchasing Options in EC2.

* **On-Demand Instances** – Pay, by the second, for the instances that you launch.
* **Spot Instances** – A Spot Instance is an unused EC2 instance that is available for less than the On-Demand price. Because Spot Instances enable you to request unused EC2 instances at steep discounts, you can lower your Amazon EC2 costs significantly. The hourly price for a Spot Instance is called a Spot price. The Spot price of each instance type in each Availability Zone is set by Amazon EC2, and adjusted gradually based on the long-term supply of and demand for Spot Instances. Your Spot Instance runs whenever capacity is available and the maximum price per hour for your request exceeds the Spot price.
* **Dedicated Hosts** – Pay for a physical host that is fully dedicated to running your instances, and bring your existing per-socket, per-core, or per-VM software licenses to reduce costs.
* **Reserved Instances** – Purchase, at a significant discount, instances that are always available, for a term from one to three years.

**Step 3) Next, we have to configure some basic networking details for our EC2 server.**

* You have to decide here, in which VPC (Virtual Private Cloud) you want to launch your instance and under which subnets inside your VPC. It is better to determine and plan this prior to launching the instance. Your AWS architecture set-up should include IP ranges for your subnets etc. pre-planned for better management. (We will see how to create a new VPC in Networking section of the tutorial.



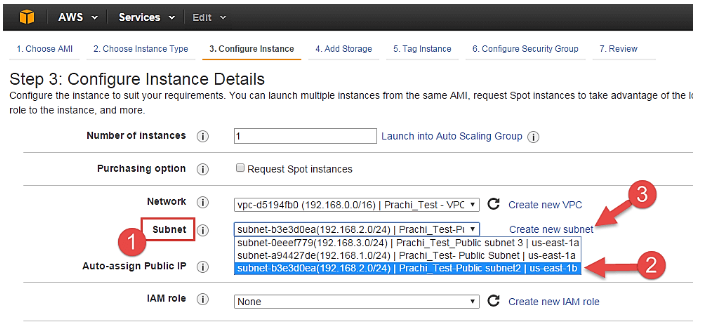
Above,

* Network section will give a list of VPCs available in our platform.
* Select an already existing VPC
* You can also create a new VPC

Here I have selected an already existing VPC where I want to launch my instance.

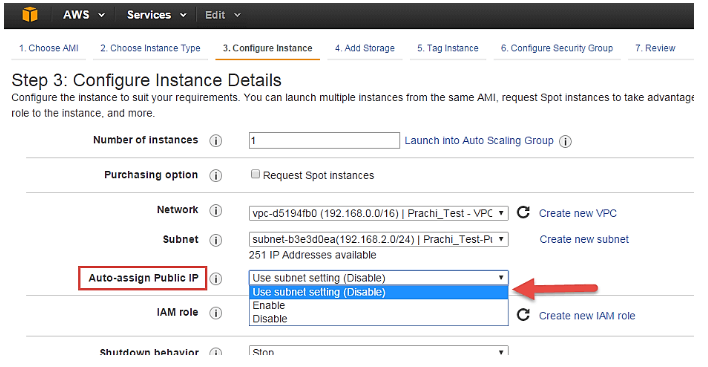
**Step 4) Subnet,**

* A VPC consists of subnets, which are IP ranges that are separated for restricting access.
* Below,
* Under Subnets, you can choose the subnet where you want to place your instance.
* I have chosen an already existing public subnet.
* You can also create a new subnet in this step.



**Once your instance is launched in a public subnet, AWS will assign a dynamic public IP to it from their pool of IPs.**

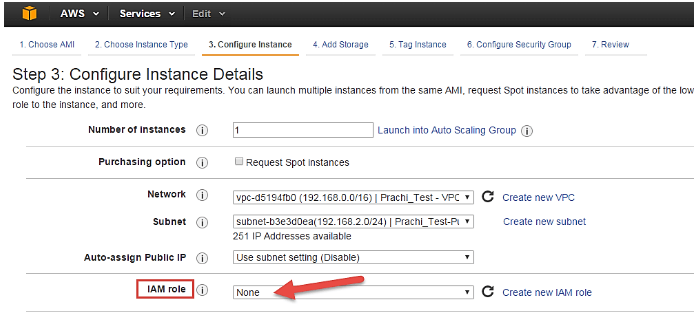
**Step 5) Auto-assign Public IP,**



* You can choose if you want AWS to assign it an IP automatically, or you want to do it manually later. You can enable/ disable 'Auto assign Public IP' feature here likewise.
* Here we are going to assign this instance a static IP called as EIP (Elastic IP) later. So we keep this feature disabled as of now. An Elastic IP address is a public IPv4 address that you can allocate to your account. You can associate it to and from instances as you require, and it's allocated to your account until you choose to release it.

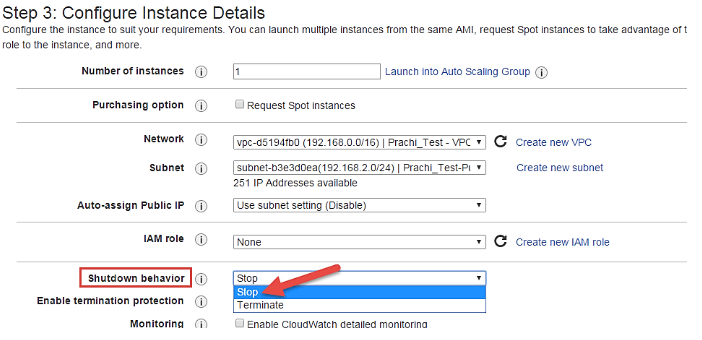
**Step 6) IAM Role,**

* In the following step, keep the option of IAM role 'None' as of now. We will visit the topic of IAM role in detail in IAM services.



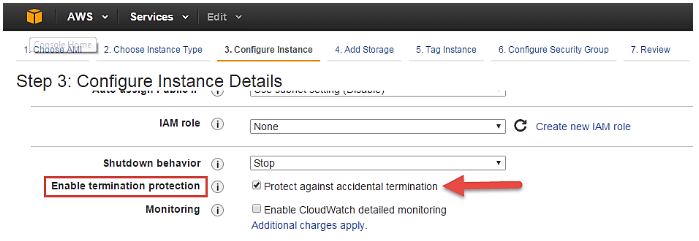
**Step 7) Shutdown behaviour**

* Shutdown Behavior – when you accidently shut down your instance, you surely don't want it to be deleted but stopped.
* Here we are defining my shutdown behavior as Stop.



**Step 8) Enable termination protection,**

* In case, you have accidently terminated your instance, AWS has a layer of security mechanism. It will not delete your instance if you have enabled accidental termination protection.
* Here we are checking the option for further protecting our instance from accidental termination.

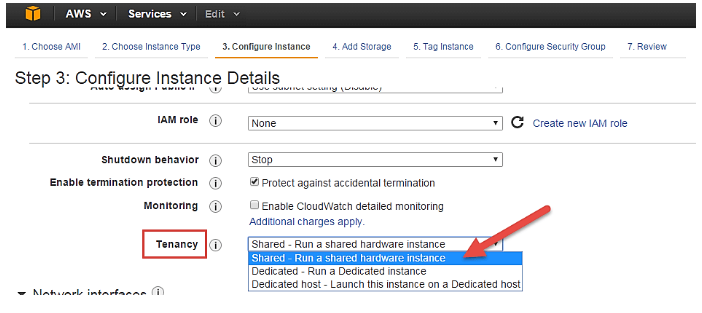


**Step 9) Monitoring and Tenancy**

* Under Monitoring- you can enable Detailed Monitoring if your instance is a business critical instance. Here we have kept the option unchecked. AWS will always provide Basic monitoring on your instance free of cost. We will visit the topic of monitoring in AWS Cloud Watch part of the tutorial.
* Under Tenancy- select the option shared tenancy. If your application is a highly secure application, then you should go for dedicated capacity. AWS provides both options.

**Shared:** The default tenancy model is the one most commonly used with AWS. Multiple customers will share the same pieces of hardware even though they don’t interact with each other. Remember that underneath there is a physical host with a hypervisor running on it to handle the virtualization of CPU, Memory, Storage etc. Customers will choose to deploy a new EC2 instance and AWS fits that instance onto the appropriate physical host and isolate it from other customers even if they’re sharing the same physical resources. This is generally the option that you will want to use unless you have regulatory compliance or licensing restrictions causing you to pick a dedicated model. The shared tenancy model is also the cheapest option for running your EC2 instances.

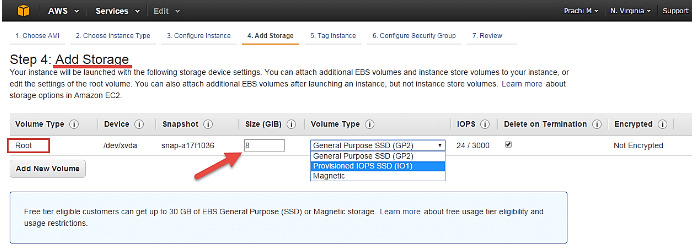
**Dedicated:** With a dedicated host, you purchase an entire physical host from AWS and that host is billed to you on an hourly basis just like EC2 instances are billed. Once you’ve purchased that host, you’re allowed to spin up as many EC2 instances as that host will allow for no additional charges. This might seem a lot like how you would manage an on-premises solution like vSphere. You buy and license the host and then you can move your instances on it until it’s full. Dedicated hosts have a few considerations that you should be aware of to ensure the proper usage and cost reductions.



**Add Storage:**

**Step 1) In this step we do following things,**

* In the Add Storage step, you'll see that the instance has been automatically provisioned a General Purpose SSD root volume of 8GB. ( Maximum volume size we can give to a General Purpose volume is 16GB)
* You can change your volume size, add new volumes, change the volume type, etc.
* AWS provides 3 types of EBS volumes- Magnetic, General Purpose SSD, Provisioned IOPs. You can choose a volume type based on your application's IOPs needs.

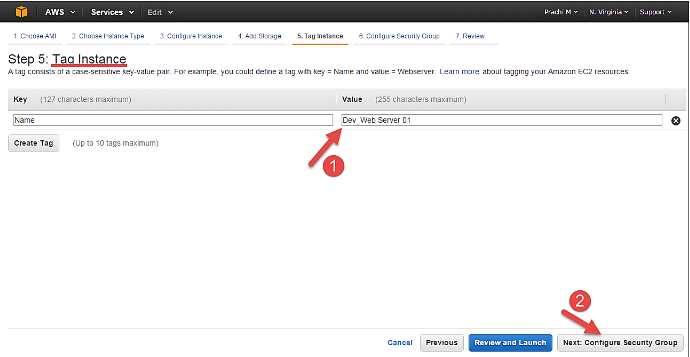


We will understand this more in EBS tutorial.

**Tag Instance:**

**Step 1)**In this step,

* You can tag your instance with a key-value pair. This gives visibility to the AWS account administrator when there are several instances.
* The instances should be tagged based on their department, environment like Dev/SIT/Prod. Etc. this gives a clear view of the costing on the instances under one common tag.
* Here we have tagged the instance as a **Dev\_Web server 01**



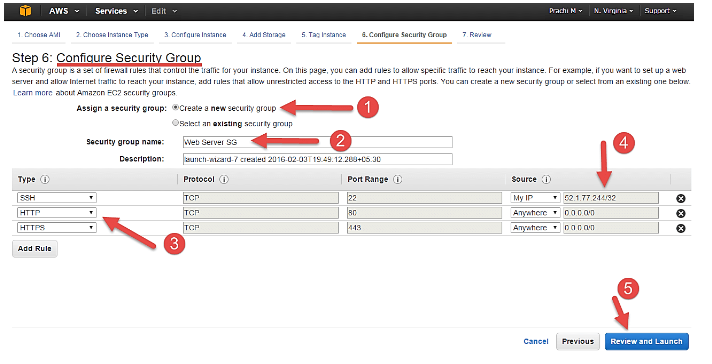
**Configure Security Groups**

**Step 1)**In this next step of configuring Security Groups, you can restrict traffic on your instance ports. This is an added firewall mechanism provided by AWS apart from your instance's OS firewall.

You can define open ports and IPs.

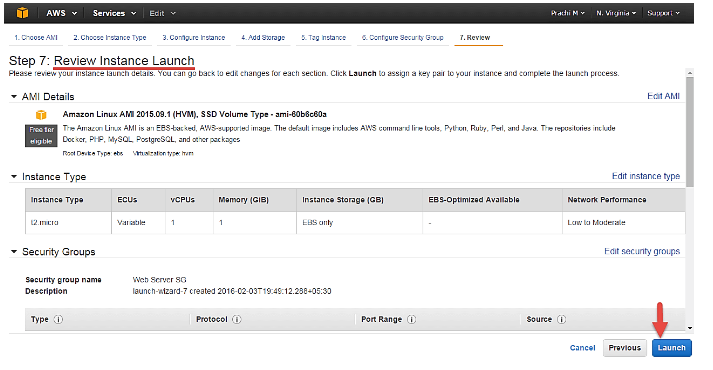
Since our server is a webserver, we will do following things

* Creating a new Security Group
* Naming our SG for easier reference
* Defining protocols which we want enabled on my instance
* Assigning IPs which are allowed to access our instance on the said protocols
* Once, the firewall rules are set- Review and launch



**Review Instances**

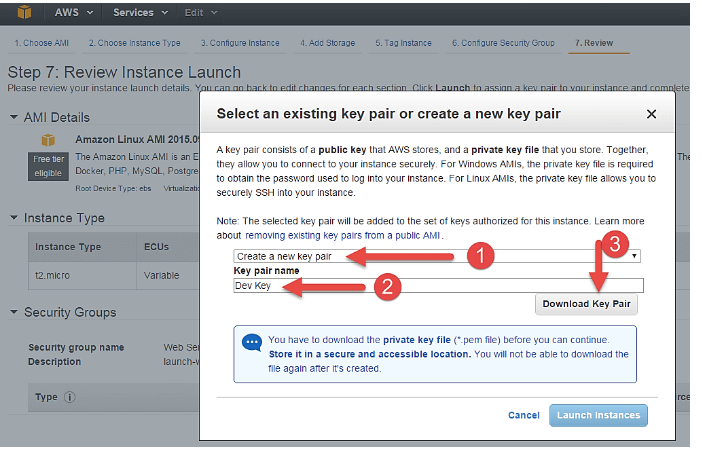
* **Step 1)** In this step, we will review all our choices and parameters and go ahead to launch our instance.



Step 2) In the next step you will be asked to create a key pair to login to you an instance. A key pair is a set of public-private keys.

AWS stores the private key in the instance, and you are asked to download the public key. Make sure you download the key and keep it safe and secured; if it is lost you cannot download it again.

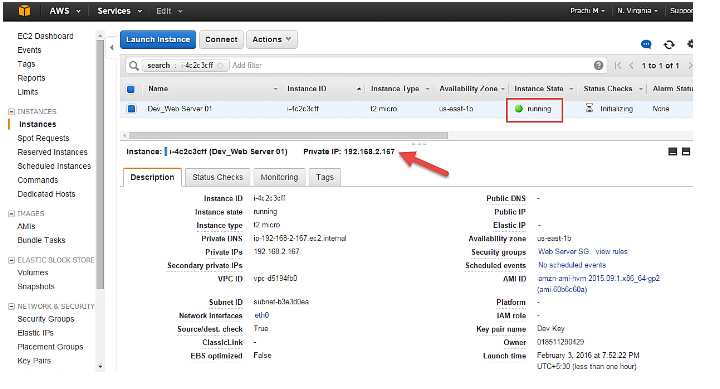
* Create a new key pair
* Give a name to your key
* Download and save it in your secured folder



**Step 3)** Once you are done downloading and saving your key, launch your instance.

Once your instance is up and running, you can see its status as 'Running' now.

Note that the instance has received a Private IP from the pool of AWS.



**What is the difference between public IP and EIP (Elastic IP)?**

When you launch an EC2 instance, you receive a Public IP address by which that instance is reachable. Once you stop that instance and restart then you get a new Public IP for the same instance. So, Public IP gets changed every time for an instance after stop/start.  
To overcome with this problem, we attach an Elastic IP to an Instance which doesn't change after you stop / start the instance as many times.  
**Advantage of Having Elastic IP**  
-> It is kind of static IP for your Instance.  
-> Doesn't change after stop/start.  
***If you have Elastic IP in your account and  it's not in use, then you will be charged for it.***

**Public IP address:**

* As its name implies, Public IP address is accessible over the internet.
* When you launch an instance in VPC by choosing the subnet, by default, one public IP address will be attached to the instance.
* You can control whether your instance receives a public address or not by

1. Modifying the public addressing attribute of your subnet.

2. Enabling or disabling the public IP addressing feature during the instance launch.

Key Points:

* It' not associated with your AWS account but restricted to the life of the instance (life means till the time instance doesn’t go into stop/start)
* When you stop & restart the instance, every time AWS will allocate the new public IP address to that instance.
* Public Ip address is assigned to the EC2 instance from the pool of public IP addresses.
* You can't manually attach or detach the public IP from your instance.
* Public IP is resolute. Public IP address of the instance will be released when you

Stop and Start the instance

Terminate the instance

Associate an Elastic IP address with the instance.

**Elastic IP address:**

* It's associated with your AWS account rather than instance.
* The elastic IP address is also accessible over the internet.
* The elastic IP address is persistent. It's associated with your account until you choose to release it.
* You can simply redirect the one instance traffic to another instance by simply detaching Elastic IP address from the old instance and attach to the new instance.
* AWS do not support the elastic IP address for IPv6.
* You can detach elastic IP from one instance & attach that same IP to a different instance.
* If you stop and restart the instance, still the elastic IP address will be associated with the instance.

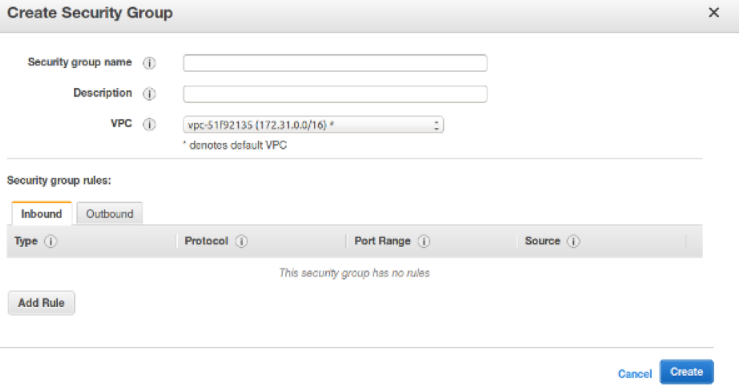
Let's assume that you launched an EC2 instance and installed https server. After that, you hosted one webpage (.html). Now you want to test that whether a hosted webpage is working fine and accessible from the internet. For that, you can test with the public IP or Elastic IP address attached with the instance by simply browsing in the browser.

But the problem with public address of the instance is that it will change every time when you stop and restart the instance. If you use elastic IP address instead of public IP, it will not change even when you stop and restart the instance. So, if you want to attach a persistent and re-attachable IP address to your instance, then you can use the Elastic IP address.

**What is Security group in AWS?**

AWS security groups (SGs) are associated with EC2 instances and provide security at the protocol and port access level. Each security group — working much the same way as a firewall — contains a set of rules that filter traffic coming into and out of an EC2 instance.

**Create security group:**

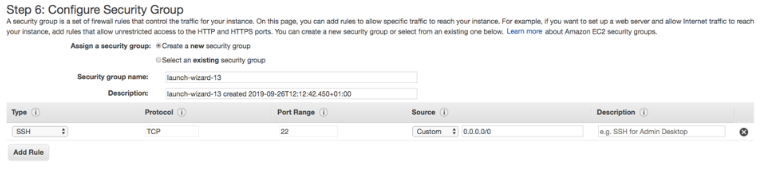


Each security group must have a name, allowing you to easily distinguish it from others.  The description is optional, but it does offer you the ability to add additional contextual information to help you understand the security groups. Security groups are specific to a VPC. As a result, during your security groups, you must specify which VPC the SG will reside. Be sure to select the correct VPC for the resource in which you want to protect.

The actual rule of a security group that filters traffic is defined in two tables: **Inbound** and **Outbound**. AWS security groups are STATEFUL, meaning you do not need to add rules for return. Therefore, any rule that allows traffic into an EC2 instance, will automatically allow responses to pass back out to the sender without an explicit rule in the Outbound rule set.

Each rule is comprised of five fields: Type, Protocol, Port Range, Source, and Description. This applies to both Inbound and Outbound rules.

* Type: The drop-down list allows you to select common protocols like SSH, RDP, or HTTP. You can also choose custom protocols.
* Protocol: This is typically grayed out, as it’s covered by most “Type” choices. However, if you create a custom rule, you can specify your protocol (TCP/UDP, etc.) here.
* Port Range: This value will also usually be pre-filled, reflecting the default port or port range for your chosen protocol. However, there might be times when you prefer to use custom ports.
* Source: This can be a network subnet range, a specific IP address, or another AWS security group. You can also leave access open to the entire internet using the “Anywhere (0.0.0.0/0)” value.
* Description: This field allows you to add a description for the rule that has been added.



#### What is a key pair and what is it used for?

A key pair, as the name implies, is made up of two components: a public key and a private key.

The function of key pairs is to encrypt the login information for Linux and Windows EC2 instances and then decrypt the same information, allowing you to authenticate onto the instance. The public key encrypts data, such as the username and password. For Window instances, the private key is used to decrypt this data, allowing you to gain access to the login credentials including the password. For Linux instances, the private key is used to remotely connect onto the instance via SSH.

The public key is held and kept by AWS, and the private key is your responsibility to keep and ensure that it is not lost.

So, going back to when you create your EC2 instance and a new key pair, you are given the opportunity to download the key pair. Once you have done this, you must keep that file safe until you are ready to log onto the associated EC2 instance. It’s worth noting that you can use the same key pair on multiple instances to save you from managing multiple private keys. Do bear in mind; however, should the private key become compromised, access could be gained to all instances where that key pair was used.

Once you have authenticated to the EC2 instance the first time, you can set up additional less privileged access controls, such as local Windows accounts allowing other users to connect and authenticate to or even utilize Microsoft Active Directory.secu.

**Some points to remember for EC2 from Cloud Guru:**

* Termination Protection is **turned off** by default, you must turn it on.
* On an EBS-backed instance, the default action is for the root EBS volume to be deleted when the instance is terminated.
* EBS root volume of your default AMI’s can’t be encrypted, additional volumes can be encrypted.
* All inbound traffic is blocked by default.
* All outbound traffic is allowed.
* Changes to security groups take effect immediately.
* You can have any number of EC2 instances within a security group.
* You can have multiple security groups attached to single EC2 instance.
* Security groups are STATEFUL.
* If you create an inbound rule allowing traffic in, that traffic is automatically allowed back out again.
* You cannot block specific IP addresses using security groups, instead use network access control lists.
* You can specify allow rules, but not deny rules. By default everything is blocked only.
* EC2 instance and EBS volume by default will be in same availability zone to avoid any leg. Though we can move this.