**Amazon EC2**

* EC2 is one of the most popular of AWS’ offering
* EC2 = Elastic Compute Cloud = Infrastructure as a Service

**EC2 sizing & configuration options**

* Operating System (OS): Linux, Windows or Mac OS
* How much compute power & cores (CPU)
* How much random-access memory (RAM)
* How much storage space:
* Network-attached (EBS & EFS)
* hardware (EC2 Instance Store)
* Network card: speed of the card, Public IP address
* Firewall rules: security group
* Bootstrap script (configure at first launch): EC2 User Data

**Creating our First Website in EC2**

* Once the EC2 instance is up and running, we can go ahead and create our first website.
* In order to host our website we have to install a webserver (ex. apache)
* What is apache?
  + It is a lightweight webserver, it is used to host the web application.
* In order the install apache web server we can take the help of package manager “yum”.

Step 1: Update the operating system

sudo yum update -y

Step 2: Install apache webserver

sudo yum install httpd -y

Step 3: Start the webserver.

sudo systemctl start httpd

Step 4: Check the status of the webserver by running the below command, it should be running

sudo systemctl status httpd

* once apache webserver installed, you can configure your website, to do so you have to follow the below steps.

Step 5: Go to the /var/www/html location by running below command.

cd /var/www/html

Step 6: Create index.html page by running below command

echo “<h1>Hello World</h1>” > index.html

Step 7: Restart the webserver

sudo sustemctl restart httpd

* then in order to access your website copy the public ip of EC2 instance from AWS portal and put is in the browser and search.

**EC2 User Data**

* It is possible to bootstrap our instances using an EC2 User data script.
* bootstrapping means launching commands when a machine starts
* That script is only run once at the instance first start
* EC2 user data is used to automate boot tasks such as:
* Installing updates
* Installing software
* Downloading common files from the internet
* Anything you can think of
* The EC2 User Data Script runs with the root user

**EC2 Instance Types - Overview**

* You can use different types of EC2 instances that are optimised for different use cases (https://aws.amazon.com/ec2/instance-types/)
* AWS has the following naming convention:

m5.2xlarge

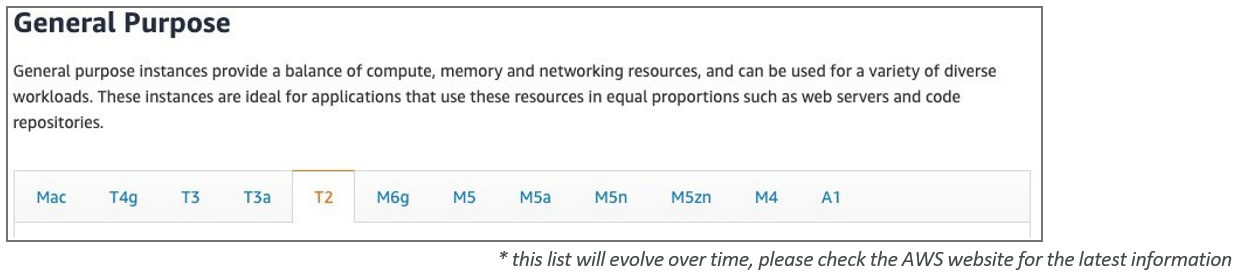
m: instance class

5: generation (AWS improves them over time)

2xlarge: size within the instance class

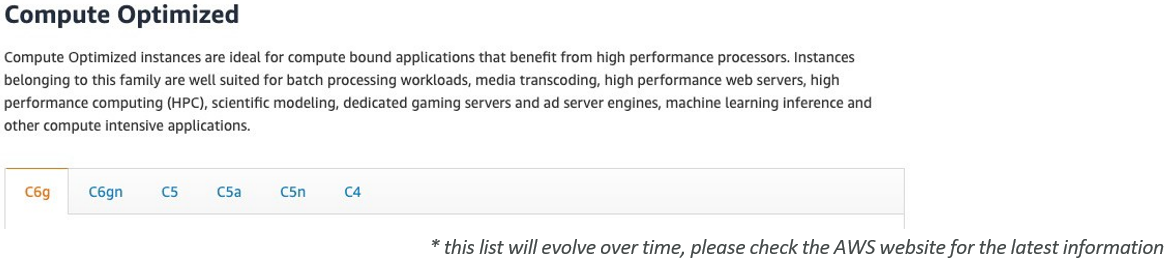
**EC2 Instance Types – General Purpose**

* Great for a diversity of workloads such as web servers or code repositories
* Balance between:
  + Compute
  + Memory
  + Networking
* In the course, we will be using the t2.micro which is a General Purpose EC2 instance



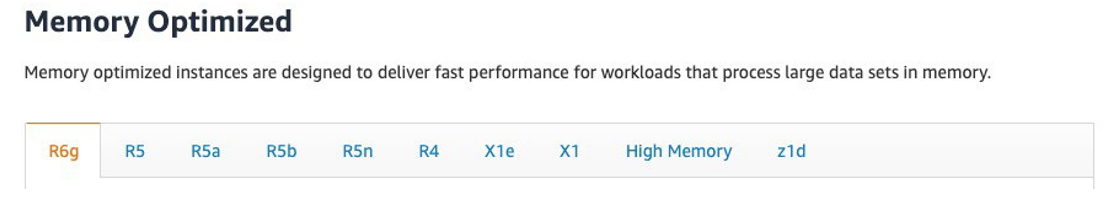
**EC2 Instance Types – Compute Optimized**

* Great for compute-intensive tasks that require high performance processors:
  + Batch processing workloads
  + Media transcoding
  + High performance web servers
  + High performance computing (HPC)
  + Scientific modeling & machine learning
  + Dedicated gaming servers



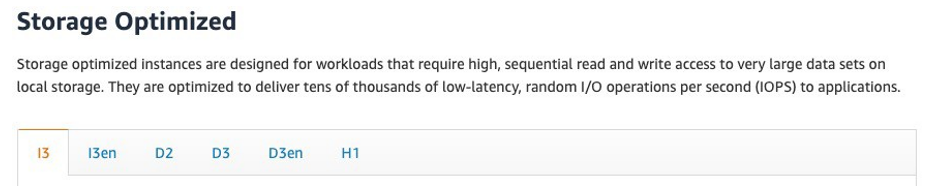
**EC2 Instance Types – Memory Optimized**

* Fast performance for workloads that process large data sets in memory
* Use cases:
  + High performance, relational/non-relational databases
  + Distributed web scale cache stores
  + In-memory databases optimized for BI (business intelligence)
  + Applications performing real-time processing of big unstructured data

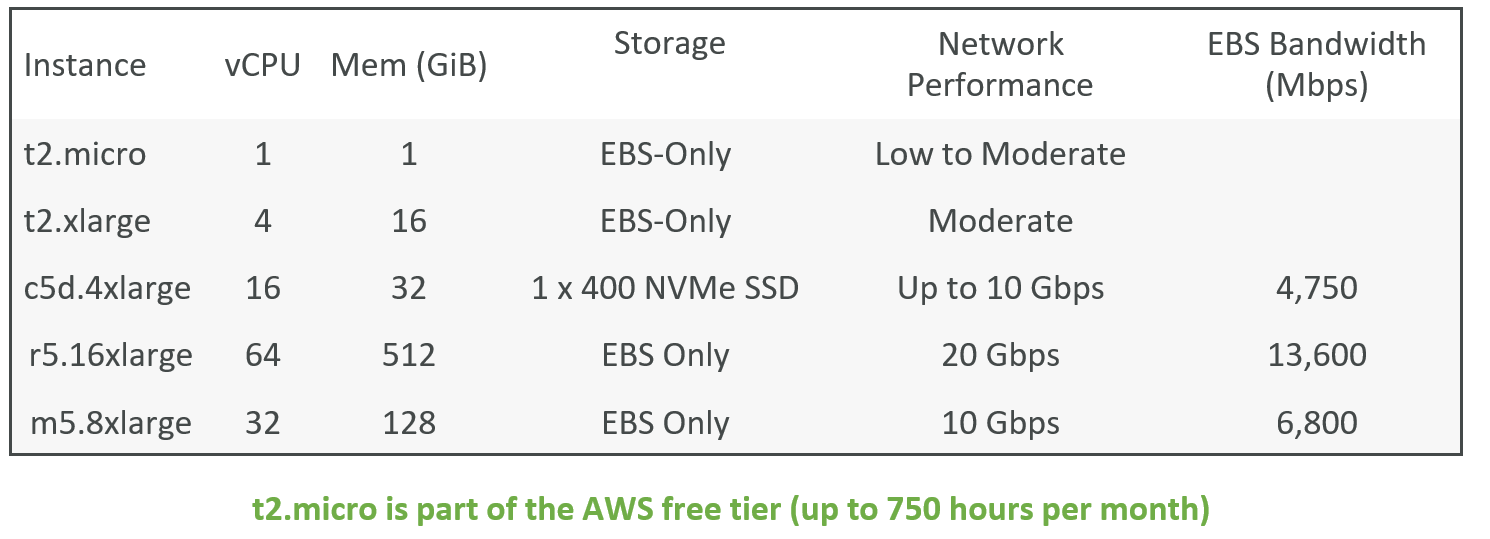


**EC2 Instance Types – Storage Optimized**

* Great for storage-intensive tasks that require high, sequential read and write access to large data sets on local storage
* Use cases:
  + High frequency online transaction processing (OLTP) systems
  + Relational & NoSQL databases
  + Cache for in-memory databases (for example, Redis)
  + Data warehousing applications
  + Distributed file systems

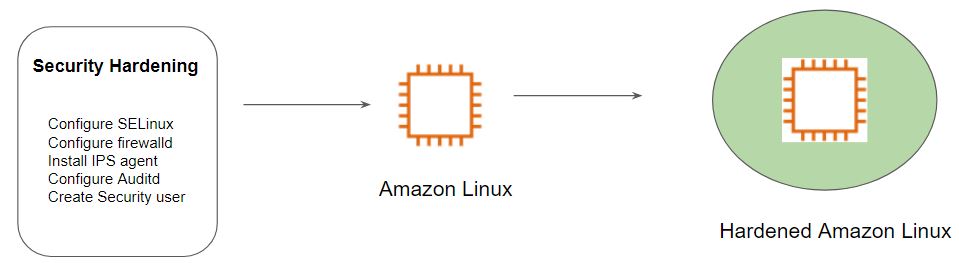


**EC2 Instance Types: example**

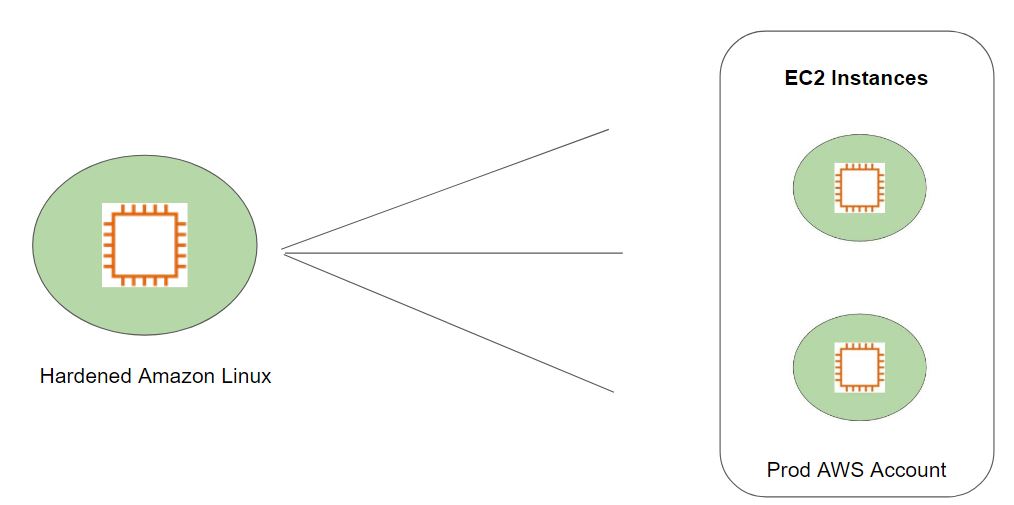


**Amazon Machine Image (AMI)**

* Amazon Machine Image (AMI) is the master image from which new EC2 instances can be launched.
* It is also called backup of EC2 instance, backup is used to restore in case of failure
* Let’s understand with an example:



The architecture of Hardened AMI Deployment:



**How to create custom image**

* Launch EC2 instance from existing AMI.
* Install and configure the software as per the requirement.
* create image of the instance.
* you can terminate the instance once image is ready.

**How to copy AMI from one region to another.**

* Select the AMI you want to copy.
* Click on the action drop down list and click on copy AMI
* Give a Name of the AMI (as per your choice)
* Select the region you want to copy your AMI to.
* Click on copy AMI button.

**AMI Permission**

* By default, whatever the AMI we create it is actually a private AMI, means it can be accessible in our account only.
* But we have the option to share with other AWS action by modifying the permission.
* We can even make it public so that it can be available to all the AWS account

***Note: while sharing AMI make sure you don’t have any sensitive data.***

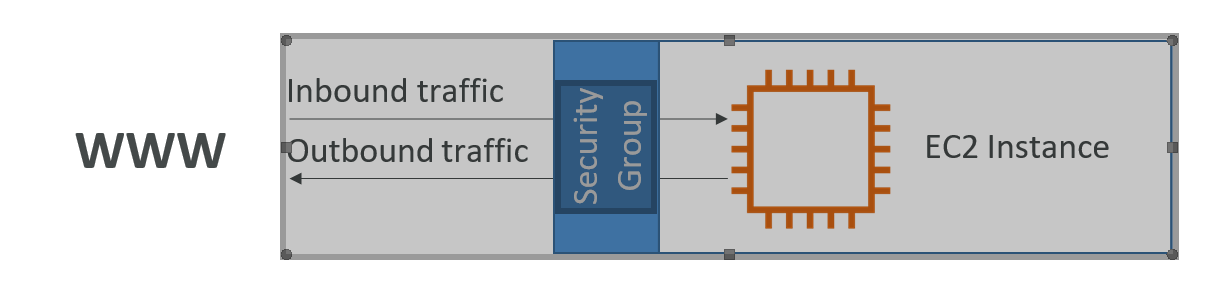
**Resizing the EC2 Instance**

* We can change the CPU, Memory (RAM) capacity of our EC2 instance after creation.
* In order to do so we have to first stop our EC2 instance and modify the CPU and memory capacity as per our requirement and start the instance.

**Introduction to Security Groups**

Security Groups are the fundamental of network security in AWS

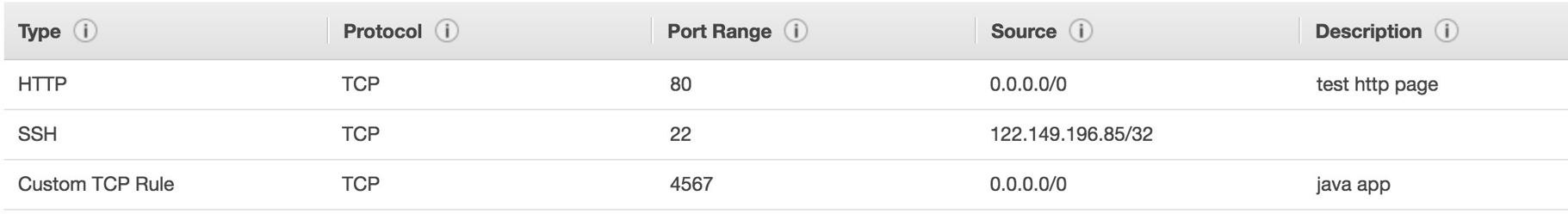
They control how traffic is allowed into or out of our EC2 Instances.



* Security groups only contain allow rules
* Security groups rules can reference by IP or by security group

**Security Groups Deeper Dive**

* Security groups are acting as a “firewall” on EC2 instances
* They regulate:
* Access to Ports
* Authorised IP ranges – IPv4 and IPv6
* Control of inbound network (from other to the instance)
* Control of outbound network (from the instance to other)



**EC2 Instances Purchasing Options**

* On-Demand Instances: short workload, predictable pricing
* Reserved: (MINIMUM 1 year)
  + Reserved Instances: long workloads
  + Convertible Reserved Instances: long workloads with flexible instances
  + Scheduled Reserved Instances: example – every Thursday between 3 and 6 pm
* Spot Instances: short workloads, cheap, can lose instances (less reliable)
* Dedicated Hosts: book an entire physical server, control instance placement
* Dedicated Instances: no other customers will share your hardware

**EC2 On Demand**

* Pay for what you use:
  + Linux or Windows - billing per second, after the first minute
  + All other operating systems - billing per hour
* Has the highest cost but no upfront payment
* No long-term commitment
* Recommended for short-term and un-interrupted workloads, where you can't predict how the application will behave

**EC2 Reserved Instances**

* Up to 72% discount compared to On-demand
* Reservation period: 1 year = + discount | 3 years = +++ discount
* Purchasing options: no upfront | partial upfront = + | All upfront = ++ discount
* Reserve a specific instance type
* Recommended for steady-state usage applications (think database)
* Convertible Reserved Instance
* can change the EC2 instance type
* Up to 66% discount
* Scheduled Reserved Instances
* launch within time window you reserve
* When you require a fraction of day / week / month
* Commitment for 1 year only

**EC2 Spot Instances**

* Can get a discount of up to 90% compared to On-demand
* Instances that you can “lose” at any point of time if your max price is less than the current spot price
* The MOST cost-efficient instances in AWS
* Useful for workloads that are resilient to failure
  + Batch jobs
  + Data analysis
  + Image processing
  + Any distributed workloads
  + Workloads with a flexible start and end time
* Not suitable for critical jobs or databases

**EC2 Dedicated Hosts**

* An Amazon EC2 Dedicated Host is a physical server with EC2 instance capacity fully dedicated to your use. Dedicated Hosts can help you address compliance requirements and reduce costs by allowing you to use your existing server-bound software licenses.
* Allocated for your account for a 3-year period reservation
* More expensive
* Useful for software that have complicated licensing model (BYOL – Bring Your Own License)
* Or for companies that have strong regulatory or compliance needs