AWS EC2

Angel

Virtual Machine

- A **Virtual Machine (VM)** is a software-based computer
- Multiple VMs can run on **one physical server**
- Each VM has:
  - Its own CPU allocation
  - Its own memory
  - Its own operating system
  - Strong isolation from other VMs

Amazon EC2

- **Amazon EC2 provides Virtual Machines in AWS**
- Each EC2 **instance = one Virtual Machine**
- Users choose:
  - Instance type (CPU / memory)
  - Operating system (AMI)
  - Storage and network

EC2 in Hadoop & Spark

- Hadoop and Spark **run on EC2 instances**
  - HDFS DataNodes
  - YARN NodeManagers
  - Spark Driver & Executors
- EC2 provides the **actual compute power**

EC2 Instance

- An **EC2 Instance** is a **running Virtual Machine** in AWS
- Created from an **AMI (Amazon Machine Image)**
- An instance includes:
    - CPU
    - Memory
    - Disk
    - Network

AMI (Amazon Machine Image)

- A **template** for creating EC2 instances
- Contains:
    - Operating system (Linux / Windows)
    - Pre-installed software
    - System configurations

Instance Types

General Purpose

- Balanced CPU & memory
- Examples: `t3`, `t3a`, `t4g`, `m5`, `m6i`

Compute Optimized

- More CPU, less memory
- Examples: `c5`, `c6i`, `c6g`

Memory Optimized

- Large memory per node
- Examples: `r5`, `r6g`, `x1`, `x2`

Storage Optimized

- High disk I/O throughput
- Examples: `i3`, `d2`

Accelerated Computing

- GPU / special hardware
- Examples: `p3`, `p4`, `g5`

EC2 Key Pair (How You Log In to an Instance)

- A **Key Pair** is used to **securely connect** to an EC2 instance

- Consists of:
    - **Public key** (stored on EC2)
    - **Private key** (downloaded and kept by you)

EC2 **does NOT use passwords** by default
Key pairs provide:

- Strong security
- No password brute force
- Automated access for servers

Key type

- RSA (most common, widely supported)
- ED25519 (newer, shorter keys)

Private key format

- `.pem` → OpenSSH (Mac / Linux / modern Windows)
- `.ppk` → PuTTY (older Windows)

EC2 Cost

EC2 instances are billed **per second** (or per hour)

Cost depends on:

- Instance type
- Running time
- Region

- **Stop instance**
  a. Compute cost stops
  b. Storage may still cost money

- **Terminate instance**
  a. Compute cost stops
  b. **Attached resources may still be billed**

Ways to Connect to an EC2 Instance

- **SSH Client**
  - Uses key pair (`.pem`)
  - Requires port **22** open
- **EC2 Instance Connect**
  - Temporary SSH access via AWS Console
- **Session Manager (SSM)**
  - Browser-based access
  - No SSH key
  - No open inbound ports

Shell

What is a Shell

- A **command-line interface** to control the OS
- Executes commands on the EC2 machine
- Common shells:
  - `bash`
  - `sh`

How You Use Shell on EC2

- Connect via SSH / Session Manager
- Run commands like:
  - `ls`, `cd`, `pwd`
  - `yum install`, `apt install`
  - `java -version`, `spark-submit`

EC2 Instance Lifecycle

An EC2 instance moves through different states during its lifetime.

- **Running**
  - Instance is powered on
  - You are charged for compute
- **Stopped**
  - Instance is powered off
  - No compute charge
- **Rebooting**
  - OS reboot, instance remains on the same host

- Public IP is **ephemeral by default**
  - It may change after stop/start
- Root storage is usually **EBS-backed**
  - EBS volumes are **charged even when instance is stopped**

Navigating EC2 via Shell / CLI

- Create or use an existing **Key Pair**
- Launch **EC2 Instances**
- Attach **Security Groups**
- Manage **Security Group Rules**
- **Start / Stop / Reboot / Terminate** instances
- Allocate and attach **Elastic IP**
- **Describe** instances to retrieve metadata

Vertical vs Horizontal Scaling for EC2

Vertical Scaling (Scale Up / Scale Down)

- Increase or decrease **CPU / Memory** of the same EC2 instance
- Requires **stopping the instance**
- Limited by the maximum instance size
- Simple but not fault-tolerant

Example:
t3.medium → m5.2xlarge

Horizontal Scaling (Scale Out / Scale In)

- Add or remove **multiple EC2 instances**
- Typically managed as a **cluster**
- Supports high availability and fault tolerance
- Common in distributed systems

Used by:
EMR, ECS, EKS, Spark, Hadoop

EC2 in a Cluster

EC2 is the building block, not the system itself.

How EC2 is used in clusters

- Multiple EC2 instances form a **cluster**
- Each EC2 plays a **specific role**
  - Master / Driver
  - Worker / Executor
- Tasks are distributed across instances

Why clusters are needed

- Data is too large for a single machine
- Workload can be parallelized
- Failures are expected and handled

Examples

- Hadoop cluster (NameNode + DataNodes)
- Spark cluster (Driver + Executors)
- Amazon EMR (Managed Hadoop / Spark on EC2)
- EMR = ec2+cluster manager+hadoop/spark GLUE(serverless)
- Airflow,