NITTE EDUCATION TRUST | NITTE MEENAKSHI INSTITUTE OF TECHNOLOGY

Department of Computer Science and Engineering
22CS74 – Cloud Computing
Unit 3: AWS Services (EC2, EBS)

Reference: Amazon EC2:
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html

1.0 Amazon EC2

Amazon Elastic Compute Cloud (Amazon EC2) provides on-demand, scalable computing capacity in the Amazon Web Services (AWS) Cloud. Using Amazon EC2 reduces hardware costs so you can develop and deploy applications faster. You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage. You can add capacity (scale up) to handle compute-heavy tasks, such as monthly or yearly processes, or spikes in website traffic. When usage decreases, you can reduce capacity (scale down) again.
An EC2 instance is a virtual server in the AWS Cloud. When you launch an EC2 instance, the instance type that you specify determines the hardware available to your instance. Each instance type offers a different balance of compute, memory, network, and storage resources

Launching an EC2 instance is a structured process facilitated by the AWS Management Console Launch Instance Wizard. This process involves nine key decisions that define the instance's configuration and capabilities:

1. **Amazon Machine Image (AMI):** Selecting a template that contains the operating system and pre-installed software.
2. **Instance Type:** Choosing the optimal mix of CPU, memory, storage, and networking capacity for the workload.
3. **Network Settings:** Defining the virtual network environment (VPC) and IP addressing.
4. **IAM Role:** Securely assigning permissions to interact with other AWS services.
5. **User Data:** Automating initial configuration with launch scripts.
6. **Storage Options:** Configuring durable or temporary block storage volumes.
7. **Tags:** Applying metadata for organization, cost allocation, and management.
8. **Security Group:** Implementing a virtual firewall to control network traffic.
9. **Key Pair:** Establishing secure cryptographic credentials for instance access.

By understanding these components, organizations can leverage EC2 to build resilient, scalable, and cost-effective solutions in the cloud, moving away from the rigid and capital-intensive model of on-premises hardware.

1.1 The Strategic Shift: On-Premises Infrastructure vs. Amazon EC2

Businesses today face a critical decision when architecting their IT infrastructure: whether to invest in traditional on-premises servers or leverage cloud-based solutions like Amazon EC2. This choice has profound implications for an organization's cost structure, operational agility,

and capacity to scale. Understanding the trade-offs between these two models is the first step toward building a modern, efficient, and resilient technology foundation.
Running servers on-premises has long been the default, but this approach comes with significant challenges and inefficiencies. These drawbacks often hinder an organization's ability to respond quickly to changing market demands.
• **High Upfront Costs:** The traditional model requires substantial capital expenditure for procuring physical server hardware.
• **Inflexible Procurement:** Hardware acquisition is frequently based on long-term projections and project plans rather than on the reality of actual usage, leading to mismatched capacity.
• **Operational Overhead:** Beyond the hardware itself, building, staffing, and maintaining physical data centers are expensive and resource-intensive undertakings.
• **Wasteful Over-provisioning:** To handle traffic spikes and peak workloads, organizations must permanently provision excess capacity. This results in server resources sitting idle and unused for significant periods, representing a wasteful expenditure.
In contrast to the rigid and costly nature of on-premises infrastructure, Amazon Elastic Compute Cloud (EC2) presents a transformative alternative. Amazon EC2 provides secure, resizable compute capacity in the cloud, offering a direct solution to the limitations of the traditional model. Its versatility supports a vast range of workloads, serving as the backbone for everything from web servers and database servers to game servers and media servers. It allows organizations to pay only for the capacity they actually use and to scale resources up or down in minutes, not months.

1.2. The Amazon EC2 Solution

Amazon Elastic Compute Cloud (EC2) directly addresses these challenges by providing secure, resizable compute capacity in the cloud. It allows users to host applications on virtual machines, known as EC2 instances, similar to traditional servers but with enhanced flexibility.
The name "Elastic Compute Cloud" breaks down as follows:
• **Elastic:** Refers to the ability to automatically and easily increase or decrease the number and size of servers to match application demand.
• **Compute:** Signifies the primary function of the servers, which is to process data and run applications using resources like CPU and RAM.
• **Cloud:** Indicates that the instances are hosted in the AWS cloud environment.

1.3. Core Capabilities and Use Cases

EC2 offers a robust set of features for deploying a wide variety of workloads.
• **Full Administrative Control:** Users have full control over the guest operating system (Windows or Linux) on each instance. Supported operating systems include Windows Server (2008, 2012, 2016, 2019), Red Hat, SuSE, Ubuntu, and Amazon Linux.
• **Global Reach:** Instances of any size can be launched into any Availability Zone worldwide in a matter of minutes.
• **Template-Based Launch:** Instances are launched from Amazon Machine Images (AMIs), which function as virtual machine templates.
• **Integrated Security:** Traffic to and from instances is controlled using virtual firewalls called security groups.

Common use cases for EC2 instances are extensive and include hosting:

Application Servers	Game Servers	Computing Servers
Web Servers	Mail Servers	Proxy Servers
Database Servers	Media Servers	Catalog Servers
File Servers		

2. The EC2 Instance Launch Process

The AWS Management Console provides a Launch Instance Wizard that guides users through nine key decisions to configure and create an EC2 instance.

2.1. Step 1: Select an Amazon Machine Image (AMI)
An AMI is a template that provides the essential information required to launch an instance. It contains a Windows or Linux operating system and often includes pre-installed software.
AMI Components:
• **Root Volume Template:** A template for the instance's primary storage volume, containing the OS and installed applications.
• **Launch Permissions:** Controls which AWS accounts are permitted to use the AMI.
• **Block Device Mapping:** Specifies the storage volumes to attach to the instance at launch.
AMI Sources:
• **Quick Start:** A selection of Linux and Windows AMIs provided and maintained by AWS.
• **My AMIs:** Custom AMIs that a user has created.
• **AWS Marketplace:** A digital catalog of thousands of software solutions from third parties, packaged as pre-configured AMIs.
• **Community AMIs:** AMIs shared by other users. These are not vetted by AWS and should be used with caution, particularly in production environments.

2.2. Step 2: Select an Instance Type
The instance type determines the hardware resources allocated to the instance, including CPU, memory, storage, and networking capacity.
Instance Type Categories:

Category	Instance Families (Examples)	Common Use Case
General Purpose	a1, m4, m5, t2, t3	Broad range of workloads like websites and web applications.
Compute Optimized	c4, c5	Compute-intensive tasks like batch processing and scientific modeling.
Memory Optimized	r4, r5, x1, z1	Memory-intensive applications like in-memory databases and data analysis.
Accelerated Computing	f1, g3, g4, p2, p3	Hardware acceleration for tasks like machine learning and graphics processing.
Storage Optimized	d2, h1, i3	Workloads requiring high, sequential read/write access like distributed file systems.

Naming Convention:
Instance type names follow a standard format: family + generation + size. For example, in **t3.large**:
• **T** is the family name.

• **3** is the generation number (higher numbers are typically more powerful and cost-effective).
• **Large** is the size, which scales resources. A t3.2xlarge has double the vCPU and memory of a t3.xlarge.

Example Instance Sizes (T3 Family):

Instance Name	vCPU	Memory (GB)	Storage
t3.nano	2	0.5	EBS-Only
t3.micro	2	1	EBS-Only
t3.small	2	2	EBS-Only
t3.medium	2	4	EBS-Only
t3.large	2	8	EBS-Only
t3.xlarge	4	16	EBS-Only
t3.2xlarge	8	32	EBS-Only

2.3. Step 3: Specify Network Settings
This step defines the instance's network location and connectivity.
• **VPC and Subnet:** The instance must be deployed into a Virtual Private Cloud (VPC)—a logically isolated network environment—and optionally into a specific subnet within that VPC.
• **Public IP Address:** Users can control whether the instance receives a public IP address, which is required for it to be accessible from the internet.
• **Network Performance:** For interdependent instances that require low latency and high throughput, they can be launched into a **cluster placement group**. Performance can also be boosted by enabling **enhanced networking** features like the Elastic Network Adapter (ENA), which supports speeds up to 100 Gbps.

2.4. Step 4: Attach an IAM Role (Optional)
To allow software on an EC2 instance to securely interact with other AWS services (e.g., read an object from an S3 bucket), an AWS Identity and Access Management (IAM) role should be attached.
• **Security:** This is the recommended security best practice, as it avoids the need to store AWS credentials directly on the instance.
• **Mechanism:** The IAM role is contained within an **instance profile**, which is attached to the instance.
• **Flexibility:** A role can be attached during the launch process or to an already-running instance.

2.5. Step 5: Add a User Data Script (Optional)
User data scripts can be provided to automate configuration tasks when an instance is first launched.
• **Functionality:** Scripts can be used to update the OS, install software, apply patches, or fetch license keys.
• **Execution:** The script runs with root privileges during the final phase of the boot process. By default, it runs only the *first* time the instance starts.
• **Scripting Language:** For Linux, scripts are typically Bash shell scripts. For Windows, scripts are written for Command Prompt (batch) or Windows PowerShell.

2.6. Step 6: Specify Storage

This step configures the storage volumes for the instance, including the root volume where the OS is installed. Users can define the size, volume type, encryption, and whether the volume should be deleted when the instance is terminated.

Primary Storage Options:

• **Amazon Elastic Block Store (EBS):** A durable, high-performance block storage service. Data on EBS volumes persists even if the instance is stopped and restarted.

• **Amazon EC2 Instance Store:** Temporary (ephemeral) block-level storage located on disks physically attached to the host computer. Data on an instance store volume is **permanently lost** if the instance is stopped or terminated. It is suitable for temporary data like caches, buffers, or scratch data.

An instance with an Instance Store root volume cannot be stopped; it can only be terminated, leading to the loss of all data, including the OS. An instance with an EBS root volume can be stopped and restarted with its data intact.

Other Storage Options (not for the root volume):

• **Amazon Elastic File System (EFS):** A scalable, managed Network File System (NFS) that can be mounted by instances.

• **Amazon Simple Storage Service (S3):** An object storage service for storing and protecting any amount of data.

2.7. Step 7: Add Tags

A tag is a label consisting of a user-defined **key** and an optional **value** that can be assigned to AWS resources.

• **Purpose:** Tags act as metadata to help categorize and manage resources. They are critical for filtering, automation, cost allocation, and access control strategies.

• **Example:** A common tag is a key of Name with a value like My Web Server. This specific key is displayed in the Name column of the EC2 console.

2.8. Step 8: Configure Security Group Settings

A security group acts as a virtual firewall that controls inbound and outbound network traffic for one or more instances. It exists outside the instance's guest OS.

• **Rules:** Rules are created to allow traffic on specific ports and protocols (TCP, UDP, ICMP) from specific sources (e.g., an IP address, a range of IPs, or another security group).

• **Default Behavior:** By default, a new security group includes an outbound rule that allows all traffic to leave the instance but has no inbound rules, blocking all incoming traffic until explicitly allowed.

2.9. Step 9: Identify or Create a Key Pair

Amazon EC2 uses public-key cryptography to enable secure login access to instances, replacing traditional passwords.

• **Components:** A key pair consists of a **public key**, which AWS stores, and a **private key**, a file that the user must download and store securely. The private key file can only be saved at the time of its creation.

• **Usage on Linux:** The private key is used with an SSH client to securely connect to the instance. The public key is automatically placed in the ~/.ssh/authorized_keys file on the instance at boot time.

• **Usage on Windows:** The private key is used to decrypt and obtain the default administrator password, which is then used to log in to the Windows Desktop via Remote Desktop Protocol (RDP).

3.0 The Amazon EC2 Instance Lifecycle: From Launch to Termination

After an EC2 instance is launched, it progresses through a series of distinct operational states. Understanding this lifecycle is crucial for effectively managing instance availability, ensuring data persistence, and controlling costs. Each state transition has important implications for how you interact with the instance and its associated data.

• **Launching:** This is the initial, transient state. After you complete the launch wizard, AWS provisions the necessary resources based on your chosen AMI and configuration, preparing the instance to run. (**pending**)

• **Running:** Once provisioning is complete, the instance enters the running state. It is now fully booted, operational, and capable of processing workloads.

• **Stopping/Stopped:** You can stop an instance, which is analogous to shutting down a physical server. The behavior of this action depends critically on the instance's root volume type:

◦ **EBS-backed instances:** When stopped, the instance's root volume and any attached EBS volumes are preserved. You are not charged for instance usage while it is stopped, and you can restart it at any time. This is the expected behavior for most persistent workloads.

◦ **Instance Store-backed instances: This is a critical architectural distinction.** These instances **cannot be stopped** via the AWS API; they can only be terminated. If the instance shuts down for any reason—including a command from within the OS or a hardware failure—it is terminated, and **all data on the instance store is permanently lost**.

• **Terminating/Terminated:** This is the final state in the lifecycle. When you terminate an instance, it is permanently deleted and cannot be recovered. For an EBS-backed instance, the root volume can be configured to either be deleted or retained upon termination—a choice made during the storage configuration (Step 6)—allowing you to preserve the data if needed.

Reference: Amazon EBS: User Guide

https://docs.aws.amazon.com/ebs/latest/userguide/what-is-ebs.html

4.0 Amazon Elastic Block Store:

Amazon Elastic Block Store (Amazon EBS) provides scalable, high-performance block storage resources that can be used with Amazon Elastic Compute Cloud (Amazon EC2) instances. With Amazon Elastic Block Store, you can create and manage the following block storage resources:

- **Amazon EBS volumes** — These are storage volumes that you attach to Amazon EC2 instances. After you attach a volume to an instance, you can use it in the same way you would use a local hard drive attached to a computer, for example to store files or to install applications.
- **Amazon EBS snapshots** — These are point-in-time backups of Amazon EBS volumes that persist independently from the volume itself. You can create snapshots to back up the data on your Amazon EBS volumes. You can then restore new volumes from those snapshots at any time.

With Amazon EBS, you can choose from four different volume types to balance the optimal price and performance.

Amazon EBS offers four primary volume types:

- General Purpose SSD (gp3, gp2),
- Provisioned IOPS SSD (io2, io1),
- Throughput Optimized HDD (st1), and
- Cold HDD (sc1).

These types let users balance price and performance for various workloads, from transactional databases to infrequently accessed backups.

Volume Type	**Category**	**Performance Characteristics**	**Ideal Use Cases**	**Cost Consideration**
General Purpose SSD (gp3)	SSD-backed	Consistent 3,000 IOPS, up to 16,000 IOPS, 1,000 MB/s	MySQL, Cassandra, boot volumes, VDI	Lower cost per GB than gp2
General Purpose SSD (gp2)	SSD-backed	Baseline 3 IOPS/GB, burst up to 3,000 IOPS	Boot volumes, dev/test, small DBs	Moderate cost
Provisioned IOPS SSD (io1/io2)	SSD-backed	Up to 64,000 IOPS per volume, low latency	Critical DBs, high-perf trans. apps	Higher cost, premium perf.
Throughput Optimized HDD (st1)	HDD-backed	Baseline 40 MB/s per TB, up to 500 MB/s per volume	Big data, log processing, data warehousing	Lower cost per GB

Cold HDD (sc1)	HDD-backed	Baseline 12 MB/s per TB, up to 250 MB/s per volume	Backup, archival, rarely accessed logs	Lowest cost, lowest perf.

EBS Volume Types Overview

- SSD-backed volumes (gp3, gp2, io1/io2) suit transactional or latency-sensitive applications, while HDD-backed (st1, sc1) volumes suit high-throughput or infrequently accessed data.
- Magnetic (standard) is legacy/infrequently used now