Module: 3

Cloud Service Providers

Cloud computing is a technology that allows users to access computing resources (*such as servers, storage, and applications*) over the internet, without the need for physical infrastructure.

It offers s*calability, flexibility, cost-efficiency, and on-demand access to resources*. **Cloud service providers (CSPs)** are **companies** that offer these resources and services to *individuals, businesses, and organizations.*

- **Key Features of Cloud Service Providers**
  - **On-Demand Self-Service:** Users can provision resources as needed without requiring human interaction with the service provider.
  - **Broad Network Access:** Resources are accessible over the internet via standard devices such as laptops, smartphones, and tablets.
  - **Resource Pooling:** CSPs use a multi-tenant model, where resources are pooled and shared among multiple customers.
  - **Scalability and Elasticity:** Resources can be scaled up or down based on demand, ensuring efficient utilization.
  - **Measured Service:** Usage is metered, and customers are charged based on what they use, ensuring transparency and cost-effectiveness.

- **Worlds Leading Cloud Service Providers**
  - **Amazon Web Services (AWS)**
    - Launched in **2006**, AWS is a market leader in cloud computing.
    - **Services:** EC2 (virtual servers), S3 (object storage), Lambda (serverless computing), RDS (databases), and more.
    - **Strengths:** Wide range of services, global infrastructure, and robust security features.
  - **Microsoft Azure**
    - Azure is **Microsoft's** cloud platform, offering services for computing, analytics, storage, and networking.
    - **Services:** Virtual Machines, Azure SQL Database, Azure Kubernetes Service (AKS), AI and machine learning tools.
    - **Strengths:** Strong integration with Microsoft products (e.g., Windows, Office 365) and hybrid cloud capabilities.
  - **Google Cloud Platform (GCP)**
    - **Google's** GCP provides services for developers and enterprises, emphasizing data analytics and AI.
    - **Services:** Compute Engine, BigQuery, Cloud Storage, TensorFlow.
    - Strengths: Advanced AI/ML tools, competitive pricing, and a strong focus on sustainability.
  - **IBM Cloud**

- **IBM** Cloud specializes in enterprise solutions and hybrid cloud environments.
- **Services:** Watson AI, Kubernetes, Blockchain, and quantum computing services.
- **Strengths:** Expertise in AI and hybrid cloud.

- **Oracle Cloud Infrastructure (OCI)**
  - **Oracle** focuses on cloud services for enterprise applications, particularly databases.
  - **Services:** Oracle Autonomous Database, OCI Compute, and Analytics Cloud.
  - **Strengths:** High-performance databases and enterprise-grade solutions.

- **Alibaba Cloud**
  - A leading cloud provider in Asia, offering services similar to AWS and Azure.
  - **Services:** Elastic Compute Service (ECS), Object Storage Service (OSS), and AI-powered solutions.
  - **Strengths:** Competitive pricing and strong presence in the Asian market.

- **Listed Factors to Choose a Cloud Service Provider**
  - **Cost:** Evaluate pricing models and total cost of ownership.
  - **Performance:** Consider latency, speed, and reliability of the provider.
  - **Security:** Ensure the provider offers robust security features like encryption, compliance certifications, and identity management.
  - **Service Portfolio:** Check for services that meet your specific business needs.

- **Global Reach:** Assess the availability of data centers in desired regions.
- **Support and SLAs:** Review customer support options and service-level agreements.

AWS

Amazon Web Services **(AWS)** is a subsidiary of **Amazon** that provides a broad suite of cloud computing services, including ***computing power, storage, databases, machine learning, artificial intelligence, networking, and more***.

It is the most widely adopted cloud platform globally, ***serving millions of customers, including startups, large enterprises, and government agencies.***

AWS operates on a ***pay-as-you-go pricing model,*** allowing customers to pay only for the resources they use, with **no upfront fees** or **long-term commitments.**

It supports a wide range of use cases, such as **hosting applications, data processing, content delivery, IoT, and disaster recovery.**

- **Key Features**
  - **Infrastructure as a Service (IaaS):** Provides *virtualized* computing resources, like Amazon EC2 (*Elastic Compute Cloud*), *Amazon S3 (Simple Storage Service), Amazon Elastic Block Store (EBS), and Amazon Virtual Private Cloud (VPC).*

- **Platform as a Service (PaaS):** Offers *tools and frameworks* to build, deploy, and manage applications, like AWS Elastic Beanstalk.
- **Software as a Service (SaaS):** Delivers software solutions hosted in the cloud, such as *Amazon Chime (video conferencing), Amazon QuickSight (data visualization), Amazon Connect (contact center), Amazon Cognito (user authentication), Amazon WorkSpaces (virtual desktops), Amazon Pinpoint (push notifications), and Amazon SES (email sending service).*
- **Global Reach:** AWS has over 100 Availability Zones spread across 30+ geographic regions worldwide, *ensuring low latency and high availability.*
- **Security:** Offers *advanced security features and compliance certifications, including encryption, identity management, and network protection.*

- **History of AWS**

  AWS ***emerged from*** Amazon's internal IT infrastructure and its need for scalable and efficient computing power.

  Key Milestones:

  - **Early 2000s: Idea Formation**
    - Amazon, as a rapidly growing e-commerce company, sought to address internal scalability challenges.

- The leadership recognized that the **infrastructure built to serve their needs could be monetized for external customers.**

- **2002: AWS Launched**
  - AWS began as a collection of web services, including **APIs** for developers to integrate payment processing and other features into their websites.
- **2006: Public Launch of Cloud Services**
  - AWS officially launched its first public cloud offerings, ***Amazon S3 (Simple Storage Service) and Amazon EC2 (Elastic Compute Cloud).***
  - These services marked the **beginning** of modern cloud computing, providing scalable and cost-effective computing resources.
- **2012–2015: Rapid Growth**
  - AWS introduced **new services**, including databases (RDS, DynamoDB), machine learning, analytics, and Lambda (serverless computing).
  - AWS expanded its data centers globally, establishing itself as the market leader.
- **2016: $10 Billion Revenue Milestone**
  - AWS crossed $10 billion in annual revenue, cementing its dominance in the cloud market.

- **2019: AWS Outposts**

    - AWS introduced Outposts, extending its cloud services to ***on-premises environments for hybrid cloud solutions.***
- **2020–Present: Advanced Innovations**
    - AWS **expanded offerings** in AI/ML (e.g., Amazon SageMaker), edge computing (AWS Wavelength), and sustainability initiatives.
    - AWS also ***emphasized custom silicon*** with products like Graviton processors to ***optimize performance and cost.***

- **Some of the first customers of Amazon Web Services (AWS) include:**
    - **CastingWords:** A podcast transcription service that was one of the first customers of AWS
    - **FilmmakerLive:** One of the first customers of AWS
    - **Pi Corporation:** A startup that was the first beta-user of EC2 outside of Amazon
    - **Microsoft:** One of the first enterprise customers of EC2
    - **SmugMug:** An early adopter of AWS that saved around $400,000 in storage costs using S3
    - **Netflix** was one of the ***first enterprises to use AWS***, adopting it in **2009**. Netflix's demands on AWS resources helped the service evolve into the integrated set of services it is today.
- **Market Leadership**

AWS remains the leading cloud provider, holding a significant market share, followed by Microsoft Azure and Google Cloud. Its **customer base** includes **Netflix, NASA, Airbnb, Samsung, and many government organizations etc**. AWS's continuous innovation and extensive ecosystem make it the backbone of modern cloud computing.

Introduction to EC2 (Elastic Compute Cloud)

Amazon EC2, a part of Amazon Web Services (AWS), is a web service that provides **resizable *compute capacity*** in the cloud. It is designed to simplify the process of scaling infrastructure, allowing developers to run applications ***without investing in physical hardware.***

EC2 enables users to launch **virtual servers** (called ***instances)***, configure them, and scale them up or down based on their needs.

- **Key Features of EC2:**
  - **Elasticity**

    EC2 allows you to scale your compute capacity up or down automatically based on demand, ensuring cost-efficiency and performance optimization.

- **Variety of Instance Types**

  EC2 offers a wide range of instance types, each optimized for specific workloads, such as ***general-purpose, compute-intensive, memory-intensive, or GPU-based tasks.***

- **Pay-as-You-Go Pricing**

  With EC2, you only pay for the compute resources you use. Pricing options include:

  - **On-Demand Instances**: Pay by the hour or second with no upfront costs.
  - **Reserved Instances**: Purchase instances for a one- or three-year term to save costs.
  - **Spot Instances**: Bid for unused EC2 capacity at lower rates.
  - **Savings Plans**: Commit to a consistent usage level for a discount.

- **Global Infrastructure**

  EC2 operates in multiple AWS Regions and Availability Zones (AZs), providing low latency and high availability.

- **Security**

  AWS provides robust security features, including:

  - Virtual Private Cloud **(VPC)** for networking.
  - Identity and Access Management **(IAM)** for access control.
  - Security Groups and Network Access Control Lists **(NACLs)** for controlling traffic.

- **Customizable**

  Users can ***customize instances*** by **choosing** the operating system, storage type, and configurations such as memory and CPU capacity.

- **Pre-Built AMIs**

  Amazon Machine Images **(AMIs)** are pre-configured templates that allow you to launch instances quickly with specific operating systems and software.

- **Integration with Other AWS Services**

  EC2 integrates seamlessly with other AWS services, such as:

  - Amazon **S3** for storage.
  - Elastic Load Balancing **(ELB)** for distributing traffic.
  - Auto Scaling for dynamic resource allocation.

- **Common Use Cases for EC2:**
  - **Web Hosting**: Host websites and web applications with scalable infrastructure.
  - **Big Data Analytics**: Run compute-intensive workloads like machine learning or data processing.
  - **Development and Testing**: Set up isolated environments for application development and testing.
  - **Batch Processing**: Process large volumes of data or computational tasks.
  - **Disaster Recovery**: Ensure business continuity by hosting backup and failover systems.

- **Advantages of EC2:**
  - **Scalability**: Instantly add or remove compute resources as needed.
  - **Cost-Efficiency**: Pay for only what you use, with the ability to choose the most cost-effective pricing model.
  - **Flexibility**: Wide range of instance types, operating systems, and configurations.
  - **Global Reach**: Deploy applications close to users with AWS's global network of regions and AZs.

● EC2 Instance Types and Uses

Amazon EC2 provides a variety of **instance types**, each optimized for different use cases. These instances are categorized ***based on their underlying hardware capabilities,*** such as CPU, memory, storage, and network performance.

Major EC2 instance families and their typical use cases are as follows:

- **General Purpose**
  - Balanced compute, memory, and networking for a variety of workloads.
  - **Examples**:
    - **t4g, t3, t2**: Burstable instances for low-cost applications with occasional high performance (e.g., web servers, dev/test).
    - **m6g, m5, m4**: For general workloads like small/medium databases, app servers, and backend systems.

- **Uses**:
  - Web servers Dev/test environments
  - Small databases Application servers

■ **Compute Optimized**

- High-performance compute for CPU-intensive workloads.
- **Examples**:
  - **c7g, c6i, c5, c4**: Optimized for tasks needing high compute power.
- **Uses**:
  - High-performance computing (HPC) Batch processing Gaming servers
  - Video encoding Scientific modeling and analytics

■ **Memory Optimized**

- Optimized for memory-intensive applications.
- **Examples**:
  - **r6g, r5, r4**: For memory-heavy tasks.
  - **x2idn, x2iedn**: Extremely high memory for large in-memory databases.
  - **u-xtb1.metal**: Bare-metal for extreme memory workloads.
- **Uses**:

- Real-time big data analytics In-memory databases (e.g., Redis, SAP HANA)
- High-performance databases

- **Storage Optimized**
  - High-speed and high-throughput storage.
  - **Examples**:
    - **i3, i4i**: High IOPS storage (NVMe SSDs) for transactional databases.
    - **d3, d2**: Dense storage for sequential I/O.
    - **h1**: Optimized for throughput storage.
  - **Uses**:
    - NoSQL databases (e.g., Cassandra, MongoDB) Data warehouses
    - Log processing Big data applications

- **Accelerated Computing**
  - Specialized hardware (GPUs, FPGAs) for parallel computing tasks.
  - **Examples**:
    - **p4d, p3, g5, g4ad**: GPU instances for ML/AI and HPC.
    - **f1**: FPGA instances for custom hardware acceleration.
  - **Uses**:

- Machine learning training and inference Video rendering
- Graphics-intensive applications Genomics and scientific research

- **High Memory**
  - Dedicated instances with terabytes of memory.
  - **Examples**:
    - **u-6tb1.metal, u-9tb1.metal**: Bare-metal instances with up to 24 TB memory.
  - **Uses**:
    - Large-scale, in-memory databases (SAP HANA) Enterprise-grade workloads

- **Bare Metal**
  - Access to physical hardware for low-level workloads.
  - **Examples**:
    - **i3.metal, m5.metal, c5.metal**: For applications requiring direct hardware access.
  - **Uses**:
    - Licensing restrictions requiring non-virtualized environments
    - High-performance storage systems Custom hypervisors

- **Networking Optimized**
  - Designed for high-bandwidth, low-latency networking.

- **Examples**:
  - Instances with **"n" suffix** (e.g., c5n, r5n).
- **Uses**:
  - Network appliances Data streaming Distributed computing applications

- **Choosing the Right Instance Type**
  - Identify workload requirements (e.g., compute, memory, storage, or networking).
  - Balance cost and performance.
  - ***Use AWS Instance Advisor or Cost Explorer for optimization.***

- **How to Create an EC2 Instance in AWS**

  - Creating an EC2 instance in AWS involves selecting an AMI, choosing instance type, configuring settings, and launching it. Follow these steps:
    - **Step 1:** Login to AWS Console
      - Go to AWS Management Console → Open EC2 Dashboard.
      - Click Instances → Click Launch Instances.
    - **Step 2:** Choose an Amazon Machine Image (AMI)
      - Select an Amazon Linux, Ubuntu, Windows, or other AMI based on your requirements.

- For example:
  - Amazon Linux 2 Ubuntu 22.04 LTS Windows Server 2022

- **Step 3:** Choose an Instance Type
  - Instance types define CPU, memory, and network performance.
  - Common options:
    - `t2.micro` (Free-tier eligible, 1 vCPU, 1GB RAM)
    - `t3.medium` (For better performance)
    - `m5.large` (For heavy applications)
- **Step 4:** Configure Instance Details
  - Choose number of instances.
  - Select VPC (Virtual Private Cloud) and Subnet.
  - Enable Auto-Assign Public IP (if needed).
  - IAM Role (if required for permissions).
- **Step 5:** Add Storage
  - Define EBS Volume (Elastic Block Storage).
  - Example: 8GB for Amazon Linux, can increase as needed.
- **Step 6:** Configure Security Group
  - Create or select a Security Group (firewall rules).

- Example:
  - SSH (Port 22) for Linux (allow only your IP).
  - RDP (Port 3389) for Windows.
  - HTTP (Port 80) / HTTPS (443) for web servers.

- **Step 7:** Select Key Pair
  - Choose an existing key pair or create a new one.
  - Download the .pem file (important for SSH access).
- **Step 8:** Launch Instance
  - Click Launch and wait for the instance to initialize.
  - Go to EC2 Dashboard → Instances to check the status.

- **Auto Scaling Instances**

  Auto Scaling in AWS is a **feature** that helps you automatically adjust the number of instances in your application to match the required workload. It ensures your application maintains high availability while optimizing costs.

  - **Setup:**
    - Define an Auto Scaling Group and associate it with a launch configuration/template.
    - Specify scaling rules and monitoring metrics in the ASG.
  - **Monitoring:**

    - AWS CloudWatch continuously tracks the metrics defined in your scaling policy.
- **Scaling Actions:**
    - Based on thresholds (e.g., CPU > 70%), AWS automatically adds instances to handle increased traffic (scaling out).
    - When metrics fall below thresholds (e.g., CPU < 30%), AWS terminates instances to reduce costs (scaling in).
- **Load Balancing:**
    - Use an **Elastic Load Balancer (ELB)** with the ASG to distribute incoming traffic among instances.
    - ASG ensures the correct number of healthy instances are behind the ELB.

- **Benefits of AWS Auto Scaling**
    - **High Availability:**
        - Ensures your application always has the right number of instances to handle the workload.
    - **Cost Optimization:**
        - Automatically scales down during periods of low demand, reducing unnecessary costs.
    - **Elasticity:**
        - Adapts to changes in workload in real-time, improving application performance.
    - **Improved Fault Tolerance:**
        - Automatically replaces unhealthy instances with new ones.

- **Customizability:**
  - Tailor scaling policies and metrics to fit your application requirements.

Introduction to Amazon Machine Images (AMI's)

Amazon Machine Images **(AMIs)** are essential components of Amazon Web Services (AWS) that ***enable users to create and manage virtual server instances*** in the cloud.

An AMI is a **pre-configured template** that contains the ***necessary software, operating system, application servers, and configurations required*** to launch an Amazon Elastic Compute Cloud (**EC2**) instance.

Key Features of AMIs

- **Pre-configured Environments**
  - AMIs provide a ready-to-use environment with an operating system (Linux, Windows, etc.), applications, and required configurations.
- **Customization**
  - Users can create custom AMIs by modifying existing ones to include specific software, security settings, and optimizations.
- **Scalability**

        - AMIs enable rapid deployment and scaling by launching multiple identical EC2 instances from the same image.

    - **Flexibility**
        - AWS offers a variety of AMIs, including public AMIs (provided by AWS or third parties), private AMIs (custom user-created images), and AWS Marketplace AMIs.

- **Benefits of Using AMIs**
    - **Faster Deployment** – Quickly launch pre-configured instances without manual setup.
    - **Consistency** – Maintain uniform configurations across multiple instances.
    - **Security** – Hardened AMIs can include built-in security configurations to meet compliance requirements.
    - **Cost Efficiency** – Optimize resource utilization by deploying lightweight, optimized AMIs.

- **Key Components of AMIs**

An **Amazon Machine Image (AMI)** consists of several essential components that define the software environment for an Amazon EC2 instance. These components ensure that instances launched from an AMI have the necessary configurations and data.

- **Root Volume (Amazon EBS or Instance Store)**

- The root volume contains the operating system (OS) and boot configurations.
- Can be backed by:
  - **Amazon Elastic Block Store (EBS):** Persistent storage that allows data to persist even after the instance is stopped or terminated.
  - **Instance Store:** Temporary storage that is lost when the instance is stopped or terminated.

- **Launch Permissions**
  - Controls who can use the AMI to launch instances.
  - Can be:
    - **Private (Default):** Only the owner can use it.
    - **Shared:** Specific AWS accounts are granted access.
    - **Public:** Any AWS user can launch instances from the AMI.

- **Block Device Mapping**
  - Defines the storage volumes attached to instances launched from the AMI.
  - Specifies:
    - Root volume type (EBS or instance store).
    - Additional attached volumes (e.g., additional EBS volumes for data storage).
- **Virtualization Type**
  - Determines how the instance interacts with the underlying hardware.

        - **Types:**
            - **Paravirtual (PV):** Older virtualization type with lower overhead but lacks support for some modern EC2 instance types.
            - **Hardware Virtual Machine (HVM):** Supports full virtualization, optimized for newer instance types and recommended for most workloads.
    - **Kernel ID (Optional)**
        - Defines the Linux kernel used by the AMI (for Paravirtual AMIs).
        - HVM AMIs typically include a kernel inside the root volume.
    - **RAM Disk ID (Optional)**
        - Specifies a separate RAM disk image used during boot (if required for specialized workloads).
    - **User Data & Metadata (at Launch)**
        - AMIs support **user data scripts** that can execute commands on the first boot.
        - AWS provides instance **metadata** services, allowing instances to retrieve dynamic configuration details.
        - These key components make AMIs flexible and customizable, allowing businesses to standardize, replicate, and scale their cloud environments efficiently.

- **Type of AMIs**

Amazon Machine Images (AMIs) come in different types *based on their source, purpose, and accessibility*. Below are the main types of AMIs:

- **Based on Accessibility**
  - **Public AMIs**
    - Provided by AWS or third-party vendors.
    - Available for anyone to use.
    - Includes general-purpose AMIs like Amazon Linux, Ubuntu, or Windows Server.
    - Example: AWS Marketplace AMIs with pre-installed applications.
  - **Private AMIs**
    - Created by a user and only accessible within their AWS account.
    - Ideal for internal applications or sensitive workloads.
    - Can be shared with specific AWS accounts.
  - **Shared AMIs**
    - Created by a user but shared with specific AWS accounts.
    - Useful for organizations that need to deploy identical instances across different teams or accounts.
  - **AWS Marketplace AMIs**
    - AMIs sold or offered by third-party vendors in the **AWS Marketplace**.

        - Includes pre-configured software solutions such as databases, firewalls, and analytics tools.
        - Some AMIs are free, while others require licensing fees.

- **Based on Storage Type**
    - **EBS-Backed AMIs**
        - The root volume is stored on **Amazon Elastic Block Store (EBS)**.
        - **Advantages:**
            - Persistent storage: Data remains even after the instance is stopped or terminated.
            - Snapshots can be taken and used to create new AMIs.
        - **Example:** General-purpose EC2 instances for long-term applications.
    - **Instance Store-Backed AMIs**
        - The root volume is stored on **instance store (ephemeral storage)**.
        - **Advantages:**
            - Faster I/O performance compared to EBS.
            - No additional cost for storage.
        - **Disadvantages:**
            - Data is lost when the instance is stopped or terminated.
        - **Example:** Temporary workloads like caching or batch processing.

- **Based on the Virtualization Type**
  - **Hardware Virtual Machine (HVM) AMIs**
    - Uses **full virtualization** for better performance.
    - Requires an Amazon EC2 instance with HVM support.
    - Supports features like GPU processing and enhanced networking.
    - Recommended for most workloads.
    - **Example:** Modern Linux and Windows AMIs.
  - **Paravirtual (PV) AMIs**
    - Uses **software-assisted virtualization** instead of direct hardware access.
    - Works on older instance types but lacks support for newer EC2 features.
    - Less common today since AWS recommends HVM AMIs.

- **Based on Operating System**
  - **Linux-Based AMIs** – Includes Amazon Linux, Ubuntu, CentOS, Debian, Red Hat Enterprise Linux (RHEL), SUSE, etc.
  - **Windows-Based AMIs** – Includes Windows Server 2012, 2016, 2019, 2022, and custom Windows AMIs.
- **Choosing the Right AMI: The best AMI depends on:**
    - **Workload requirements** (e.g., persistent vs. temporary storage).

- **Performance needs** (HVM for better hardware acceleration).
- **Operating system preference** (Linux vs. Windows).
- **Security and compliance** (private AMIs for internal use).

- **Creating and Managing AMI's**

Creating and managing AMIs (Amazon Machine Images) is a critical task when working with AWS (Amazon Web Services), especially for ensuring you can consistently replicate your environment and easily deploy EC2 instances. Here's a general guide to help you understand the process:

- **Creating an AMI**
  - To create an AMI, follow these steps:
    - **Prepare your EC2 instance**
      - Launch an EC2 instance with your desired configurations (OS, software, settings).
      - Configure the instance, install necessary software, and make any changes to meet your environment's needs.
    - **Create the AMI**
      - Go to the EC2 Dashboard.
      - In the left navigation pane, under "Instances," select the instance you want to create an AMI from.

- Click on **Actions > Image and templates > Create Image**.
- Provide a name and description for the AMI.
- Choose whether to include additional volumes (other EBS volumes attached to the instance).
- Click **Create Image**.

- **Managing AMIs**

Once you have created AMIs, managing them is important to keep your environment consistent and avoid unnecessary resource usage. Here are the steps:

- **Viewing AMIs:**
  - Go to the **AMI section** under the EC2 dashboard.
  - You will see all the AMIs listed, including their ID, creation date, and status.
- **Launch an Instance from an AMI:**
  - Select the AMI you want to launch and click **Launch**.
- **Copying AMIs:**
  - You can copy AMIs across AWS regions if you need to replicate your environment in another region
  - In the AMI section, select the AMI, click on **Actions > Copy AMI**.

- **Deleting AMIs:**
    - Once you no longer need an AMI, you can delete it to avoid unnecessary storage costs.
    - Select the AMI, click **Actions > Deregister**.
- **Cleaning Up Associated Snapshots:**
    - After deregistering an AMI, remember to delete the snapshots associated with it to save costs.
    - Go to the **Snapshots** section, select the snapshot, and delete it.

- **Launching instances using AMI's Images**

Once you have an Amazon Machine Image (AMI), you can easily launch EC2 instances using it.

Step 1: Open the AWS EC2 Dashboard

- Sign in to the **AWS Management Console**.
- Navigate to **EC2** by searching for it in the services menu.

Step 2: Choose Your AMI

- In the **left navigation pane**, click on **AMIs** (under "Images").
- Locate the AMI you want to use. You can filter by **Owned by me** to see your private AMIs.

- Select the AMI and click **Launch instance from image**.

Step 3: Configure Instance Settings

- **Choose Instance Type:** Select an instance type (e.g., `t2.micro` for free tier).
- **Configure Instance Details:**
  - Choose the number of instances. Select a **VPC and Subnet**.
  - Configure **IAM roles**, user data, and networking settings as needed.
- **Add Storage:** Modify the EBS volume size if needed.
- **Add Tags:** Create key-value pairs (e.g., `Name = MyServer`).
- **Configure Security Group:** Select an existing security group or create a new one.

Step 4: Review and Launch

- Review all settings. Click **Launch**.
- Select an existing key pair or create a new one. Click **Launch Instances**.

- **Examples of AMIs**
  - **Amazon Linux AMI**
    - Amazon Linux 2 Amazon Linux 2023

- **Windows AMIs**
  - Microsoft Windows Server 2019 Microsoft Windows Server 2022
- **Ubuntu AMIs**
  - Ubuntu 22.04 LTS Ubuntu 20.04 LTS
- **Red Hat Enterprise Linux (RHEL) AMIs**
  - RHEL 8 RHEL 9
- **SUSE Linux AMIs**
  - SUSE Linux Enterprise Server 15
  - SUSE Linux Enterprise Server 12
- **Debian AMIs**
  - Debian 11 (Bullseye) Debian 12 (Bookworm)
- **Custom and Marketplace AMIs**
  - Pre-configured AMIs from AWS Marketplace (e.g., WordPress, Jenkins, SAP, etc.)
  - Custom AMIs created by users with specific configurations

Elastic IPs

An **Elastic IP (EIP)** in AWS is a static, public IPv4 address designed for dynamic cloud computing. It allows you to mask the failure of an instance or software by rapidly remapping the address to another instance in your account. Here's a breakdown of how it works and why it's useful:

- **Key Features of Elastic IPs:**
  - **Static IP Address:** Unlike regular public IPs that can change when you stop/start an instance, an Elastic IP remains constant until you release it.
  - **Reassignable:** You can quickly remap an Elastic IP to another EC2 instance in your account. This is useful for maintaining the same public IP when instances fail or need to be replaced.
  - **Associated with Your Account:** Elastic IPs are allocated to your AWS account, not to a specific instance, which gives you control over how and where they are used.
  - **Reverse DNS:** AWS allows you to configure reverse DNS for Elastic IPs, which is helpful for running mail servers.

- **How to Allocate and Use an Elastic IP:**
  - **Allocate an Elastic IP:**
    i. Go to the **EC2 Dashboard**.
    ii. Click on **Elastic IPs** under **Network & Security**.
    iii. Select **Allocate Elastic IP address**.

- **Associate an Elastic IP:**
    i. After allocation, select the EIP and click **Associate Elastic IP address**.
    ii. Choose the EC2 instance or network interface you want to associate it with.
- **Disassociate/Release an Elastic IP:**
    i. You can disassociate it from an instance and reassign it elsewhere.
    ii. If you no longer need it, you can release the EIP to avoid unnecessary charges.

- **Costs and Considerations:**
    - **Free When in Use:** AWS doesn't charge for an Elastic IP that is associated with a running instance.
    - **Charges for Unused IPs:** If the Elastic IP is allocated but not attached to a running instance, AWS charges a small hourly fee.
    - **Limitations:** By default, AWS allows you to allocate **5 Elastic IPs per region**. You can request a limit increase if needed.

- **Common Use Cases:**
    - **Failover Solutions:** Quickly remap an Elastic IP to a standby EC2 instance if the primary one fails.
    - **Web Hosting:** Keep the same IP for your website, even if you upgrade or replace your EC2 instance.
    - **Email Servers:** Use reverse DNS with Elastic IPs for running email servers.

Elastic Load Balancing

Elastic Load Balancing (ELB) in AWS automatically distributes incoming application traffic across multiple targets, such as EC2 instances, containers, IP addresses, and Lambda functions. It helps ensure high availability, fault tolerance, and scalability for your applications.

- **Key Features of Elastic Load Balancing:**
  - **Automatic Traffic Distribution:** ELB balances incoming traffic across multiple targets in one or more Availability Zones (AZs).
  - **High Availability & Fault Tolerance:** By distributing traffic, ELB ensures that if one instance fails, traffic is rerouted to healthy instances.
  - **Health Checks:** Regularly monitors the health of registered targets and only routes traffic to healthy ones.
  - **Scalability:** Automatically scales to handle changes in incoming traffic.
  - **Security:** Supports **SSL/TLS termination**, **AWS WAF** (Web Application Firewall), and **integrates with IAM** for access management.
  - **Supports IPv4 and IPv6.**

- **Types of Load Balancers in AWS:**
  - **Application Load Balancer (ALB):**

i. Operates at the **application layer (Layer 7)**.
ii. Best for **HTTP/HTTPS** traffic.
iii. Supports advanced routing features like **path-based** or **host-based** routing.
iv. Ideal for microservices and containerized applications (integrates well with ECS & EKS).

- **Network Load Balancer (NLB):**
  i. Operates at the **transport layer (Layer 4)**. ii. Best for **TCP/UDP** traffic.
  ii. Extremely fast and capable of handling millions of requests per second with **low latency**.
  iii. Ideal for applications needing **high performance** or static IP addresses.
- **Gateway Load Balancer (GWLB):**
  i. Operates at **Layer 3 (Network Layer)**.
  ii. Designed for deploying, scaling, and managing **third-party virtual appliances** like firewalls or intrusion detection systems.
- **Classic Load Balancer (CLB):**
  i. Legacy option operating at both **Layer 4 and Layer 7**.
  ii. Limited features compared to ALB and NLB. iii. Only recommended for legacy applications.

- **How Elastic Load Balancing Works:**
  - **Create a Load Balancer:**
    i. Go to **EC2 Dashboard → Load Balancers → Create Load Balancer**.

ii. Choose the load balancer type (ALB, NLB, etc.).

- **Configure Listeners & Target Groups:**
  i. **Listeners**: Define protocols and ports (e.g., HTTP on port 80).
  ii. **Target Groups**: Register your EC2 instances, Lambda functions, or IP addresses.
- **Health Checks:**
  i. Set up health checks to monitor your targets (e.g., pinging a specific endpoint).
  ii. ELB will stop sending traffic to unhealthy targets.
- **Add Security Settings:**
  i. Configure **Security Groups** and **SSL certificates** if needed.
- **Test the Load Balancer:**
  i. Use the **DNS name** provided by AWS to test the load balancer.

- **Pricing Considerations:**
  - **Per Hour Charge:** For every hour or partial hour your load balancer runs.
  - **Per GB Data Processed:** Charged based on the amount of data handled.
  - **Cross-Zone Load Balancing Costs:** May incur additional costs if enabled.

- **Common Use Cases:**
  - **Web Applications:** Use ALB to distribute HTTP/HTTPS traffic for websites or microservices.

- **High-Performance Apps:** Use NLB for applications that need ultra-low latency, like gaming or financial platforms.
- **Security Appliances:** Use GWLB for integrating third-party security tools in your network.
- **Legacy Systems:** Use CLB if migrating old applications that don’t require modern features.