AWS - Ec2 - Elastic Compute Cloud

## 🧠 What is AWS EC2?

**Amazon EC2** is a web service that provides **resizable, scalable compute capacity** in the cloud. It allows you to run virtual machines (called **instances**) on-demand, with control over the operating system, storage, networking, and more.

---

## 🧩 Key Components of Amazon EC2

### 1. Instances

- Virtual servers running applications.
- You choose the instance **type** (e.g., t3.micro, m5.large) based on CPU, RAM, and network needs.
- You can launch, stop, restart, or terminate them.

### 2. Amazon Machine Image (AMI)

- A **template** used to launch instances.
- Includes OS (e.g., Linux, Windows), application server, and custom software.
- You can use AWS-provided AMIs or create your own.

### 3. Instance Types / Families

- Defines the **hardware specs**: CPU, memory, storage, and networking.
- Examples:
    - **General Purpose**: t3, m6g
    - **Compute Optimized**: c6i
    - **Memory Optimized**: r6g

    - **Accelerated Computing**: p4, inf2
    - **Storage Optimized**: i3, d3en

4. EBS (Elastic Block Store)

- **Persistent block storage** volumes for EC2.
- Works like a virtual hard drive.
- Can be attached/detached, backed up with **snapshots**.

5. Instance Store

- **Ephemeral** storage physically attached to the host.
- Fast, but **data is lost** when instance stops or terminates.

6. Security Groups

- **Virtual firewalls** for EC2 instances.
- Control inbound and outbound **traffic rules** (IP, port, protocol).

7. Key Pairs

- Used for **SSH access** to Linux instances or **password decryption** for Windows.
- Consists of a **public key** (in AWS) and a **private key** (you download and keep safe).

8. Elastic IP Address

- **Static public IPv4 address** for dynamic cloud computing.
- Useful when an instance needs a consistent public IP.

9. Placement Groups

- Control **how instances are placed** across hardware.
    - **Cluster**: low latency, high throughput
    - **Spread**: max fault tolerance
    - **Partition**: for large distributed apps

### 10. Launch Templates / Launch Configurations

- Define instance settings like AMI, instance type, key pair, etc.
- Used for automation and **Auto Scaling**.

### 11. Auto Scaling

- Automatically **adds/removes EC2 instances** based on demand.
- Works with **CloudWatch alarms** for CPU, memory, etc.

### 12. Elastic Load Balancer (ELB)

- Distributes incoming traffic across multiple EC2 instances.
- Improves **fault tolerance** and **availability**.

### 13. Networking

- EC2 instances run inside a **VPC (Virtual Private Cloud)**.
- Each instance is in a **subnet**, with an associated **private and public IP**.
- Can use **ENI (Elastic Network Interface)** for multi-interface setups.

---

## 🛠️ Optional Services Often Used With EC2

- **CloudWatch**: Monitor and collect logs/metrics from EC2.
- **IAM Roles**: Assign permissions to EC2 without hardcoding credentials.
- **EC2 Fleet/Spot Instances**: Use excess AWS capacity at a lower cost.

---

### 🧱 EC2 in a Real Setup

Imagine you're building a web app:

- **AMI**: Based on Ubuntu, with NGINX and your app pre-installed.
- **Instance Type**: t3.medium for web server.
- **EBS**: 100 GB for app data.
- **Security Group**: Allows ports 22 (SSH), 80 (HTTP), and 443 (HTTPS).
- **Auto Scaling Group**: Adjusts instances based on CPU.
- **ELB**: Routes user traffic across multiple instances.
- **IAM Role**: Allows EC2 to read/write to S3.

---

**AWS - EC2 - Instance - Types**

- EC2 instance types are grouped into **families**, each optimized for different use cases such as general purpose, compute, memory, storage, or GPU workloads.\

---

### 🟢 1. General Purpose Instances

Balanced compute, memory, and networking for general workloads.

### 🧩 T Series (Burstable Performance)

- Examples: t3, t3a, t4g
- 💡 Uses CPU credits — ideal for variable CPU workloads

- ✅ Great for: Low-traffic websites, development/testing

### 🧩 M Series (Balanced Performance)

- Examples: m5, m6a, m6g, m7i
- 💡 Fixed CPU performance with balanced resources
- ✅ Great for: App servers, caching, backend services

---

## 🔵 2. Compute Optimized Instances

Best for compute-heavy workloads needing high-performance CPUs.

### ⚡ C Series (High CPU)

- Examples: c6i, c7g, c6a
- 💡 Graviton3 options (ARM) available for cost-efficiency
- ✅ Great for: Gaming servers, high-performance web apps, batch processing

---

## 🔴 3. Memory Optimized Instances

Designed for memory-intensive applications.

### 🧠 R Series (High Memory)

- Examples: r5, r6g, r7iz
- 💡 High memory-to-vCPU ratio
- ✅ Great for: Databases, in-memory caches (Redis)

### 🧠 X Series (Extra High Memory)

- Examples: x1e, x2idn, x2iezn
- 💡 Up to 4 TB RAM
- ✅ Great for: SAP HANA, Oracle DB, enterprise workloads

### ⚙️ Z Series (High GHz + Memory)

- Example: z1d
- 💡 Up to 4.0 GHz sustained CPU
- ✅ Great for: Financial simulations, EDA, licensing-restricted apps

---

## 🟠 4. Storage Optimized Instances

Built for workloads with high disk throughput or IOPS.

### 💾 I Series (Fast SSD/NVMe)

- Examples: i3, i3en, i4i
- 💡 Direct-attached NVMe SSDs
- ✅ Great for: NoSQL DBs, high I/O apps

### 📦 D Series (Dense HDD Storage)

- Examples: d3, d3en
- 💡 High sequential throughput with HDD

- ✅ Great for: Hadoop, large file storage

📁 **H Series (Throughput-Oriented)**

- Example: h1
- 💡 Balanced storage + compute
- ✅ Great for: Data lakes, log processing

---

🟣 **5. Accelerated Computing Instances**

Use GPUs or FPGAs for advanced computing tasks.

🎮 **P Series (GPU – ML Training)**

- Examples: p4d, p3
- 💡 NVIDIA GPUs (A100, V100)
- ✅ Great for: Deep learning, 3D rendering

🤖 **Inf Series (Inference)**

- Example: inf2
- 💡 Powered by AWS Inferentia chips
- ✅ Great for: Scalable ML inference

🧪 **Trn Series (ML Training)**

- Example: trn1

- 💡 Powered by AWS Trainium chips
- ✅ Great for: LLMs, large model training

### 🔧 F Series (FPGAs)

- Example: f1
- 💡 Custom hardware acceleration
- ✅ Great for: Genomics, encryption, hardware prototyping

---

## ⚫ 6. Specialized / Miscellaneous Instances

### 🧱 Bare Metal Instances

- Example: i3.metal, m5.metal
- 💡 Direct hardware access (no virtualization)
- ✅ Great for: Licensing needs, custom hypervisors

### 🍏 Mac Instances

- Example: mac1.metal, mac2.metal
- 💡 Based on Apple Mac mini hardware
- ✅ Great for: iOS/macOS development & CI/CD

---

## 📊 Instance Size Examples

| Size | Specs Example |
| --- | --- |
| t3.micro | 2 vCPU, 1 GiB RAM |
| m5.large | 2 vCPU, 8 GiB RAM |
| r5.4xlarge | 16 vCPU, 128 GiB RAM |
| i3.16xlarge | 64 vCPU, 488 GiB RAM, 15 TB SSD |

---

**AWS - Ec2 - Purchase - Options**

**When launching an EC2 instance, you can choose different pricing models depending on your budget, workload type, and flexibility. Here's a breakdown:**

---

**💵 1. On-Demand Instances**

- **🔄 Pay-as-you-go: Pay per second/minute with no upfront commitment.**
- **🕒 Flexible: Start, stop, and terminate any time.**
- **💸 Higher cost compared to other options.**
- **✅ Best for:**
    - **Short-term or unpredictable workloads**
    - **Development, testing, or proof-of-concept**

- Startups or early-stage projects

---

## 📉 2. Reserved Instances (RI)

- 💰 Commit to 1 or 3 years of usage for significant savings (up to 75% off on-demand).
- 🔐 Reserved capacity in a specific Availability Zone (for Standard RIs).
- 📦 Two Types:
  - Standard RI: Maximum discount, less flexible.
  - Convertible RI: Lower discount but allows changing instance type/family.
- ✅ Best for:
  - Steady-state workloads (e.g., databases, web apps)
  - Known, predictable usage over a long period

---

## 🧾 3. Savings Plans

- 🧠 Flexible alternative to Reserved Instances.
- 📅 Commit to a fixed amount of compute usage ($/hour) for 1 or 3 years.
- 💡 Covers EC2, Lambda, and Fargate.
- 📦 Two Types:
  - Compute Savings Plan: Flexibility across region, family, OS.

    - EC2 Instance Savings Plan: Cheaper, but limited to instance family/region.
- ✅ Best for:
    - Consistent workloads needing flexibility across services and instance types

---

🎯 4. Spot Instances

- 🛒 Buy unused EC2 capacity at up to 90% discount.
- 📉 Price fluctuates based on supply/demand.
- ⚠️ AWS can terminate the instance anytime with a 2-minute warning.
- ✅ Best for:
    - Fault-tolerant, stateless workloads
    - Big data, batch processing, containerized tasks (e.g., with ECS/EKS)
    - CI/CD pipelines, ML training (if interruptions are okay)

---

🏷️ 5. Dedicated Hosts

- 🖥️ Physical servers fully dedicated to you.
- 📋 Required for licensing scenarios (e.g., BYOL for Windows/Oracle).
- 🔒 Provides visibility and control over socket, cores, and host placement.
- ✅ Best for:

- Compliance-bound workloads
- Software license management tied to physical cores

---

🧱 6. Dedicated Instances

- Similar to shared EC2 instances, but run on hardware dedicated to a single customer.
- No control over the underlying hardware like Dedicated Hosts.
- ✅ Best for:
  - Regulatory requirements
  - Isolation from other tenants without needing full host control

---

🌐 7. Capacity Reservations

- 📦 Reserve instance capacity in a specific Availability Zone.
- Can be used with On-Demand or RIs.
- ⏳ Useful for ensuring guaranteed capacity during peak demand.
- ✅ Best for:
  - Mission-critical apps
  - Disaster recovery readiness
  - Compliance-sensitive workloads

---

### 📌 Quick Comparison Table

| Option | Cost | Commitment | Flexibility | Use Case |
|---|---|---|---|---|
| 💵 On-Demand | High | None | High | Dev/test, short-term |
| 📉 Reserved | Low | 1–3 years | Medium | Steady workloads |
| 🧾 Savings Plans | Low | 1–3 years | High | Flexible, steady workloads |
| 🎯 Spot | Very Low | None | Low | Fault-tolerant, batch jobs |
| 🏷️ Dedicated Host | High | Optional | High | BYOL, compliance needs |
| 🧱 Dedicated Instance | High | None | Medium | Tenant isolation |
| 🌐 Capacity Reserve | Standard | Optional | High | Capacity assurance |