

## **EC2 Fundamentals**

### **Index**

AWS Billing Management and Budgeting	<b>2</b>
Introduction to Amazon EC2	<b>4</b>
EC2 Instance Types	<b>9</b>
EC2 Security Groups	<b>11</b>
Connecting to EC2 Instances: SSH	<b>15</b>
SSH Connection to EC2 Instances on Windows	<b>17</b>
SSH Troubleshooting	<b>19</b>
EC2 Instance Connect: An Alternative to SSH	<b>21</b>
Using IAM Roles for EC2 Instances	<b>23</b>
EC2 Instance Purchasing Options	<b>26</b>
EC2 Spot Instances	<b>28</b>
EC2 Instance Launch Methods	<b>31</b>

# **AWS Billing Management and Budgeting**

## **Setting Up a Budget and Alarm**

### **1. Accessing the Billing Console:**

- Click on the top right corner of your AWS screen and select **Billing and Cost Management**.
- If you see **access denied** messages as an IAM user, switch to your **root account** to enable billing access for IAM users.

### **2. Enabling IAM Access to Billing Information:**

- Log in to the **root account**.
- Go to **Account settings** and scroll down to find **IAM user and role access to billing information**.
- Activate this option to allow IAM users with administrative access to view billing details.

### **3. Viewing the Billing Dashboard:**

- Once access is enabled, refresh the Billing Console page to view billing data.
- Dashboard shows:
  - **Month-to-date costs.**
  - **Forecasted costs** for the current month.
  - **Last month's total costs.**
- Provides a **cost breakdown by service**.

### **4. Analyzing Monthly Bills:**

- Navigate to **Bills** and select a specific month.
- Scroll to see charges by service, such as **EC2**, and get a detailed breakdown of costs.
- Examples of cost items:
  - **NAT Gateway** costs.
  - **EBS** (Elastic Block Store) costs.
  - **Elastic IP** costs.

### **5. Monitoring Free Tier Usage:**

- Click on **Free Tier** on the left-hand side.

- View current usage and forecasted usage of free tier services.
- Check if usage exceeds free tier limits, which triggers potential billing.

#### 6. **Creating a Budget:**

- Go to **Budgets** on the left-hand side.
- Create a budget to receive alerts when spending reaches a threshold.
- Example Budgets:
  - **Zero Spend Budget:** Alerts when spending reaches 1 cent.
    - Set name (e.g., **My Zero Spend Budget**) and add email (e.g., stephane@example.com).
  - **Monthly Cost Budget:** Set a monthly spending limit (e.g., \$10).
    - Receive alerts at 85% and 100% of the actual or forecasted budget.

#### 7. **Setting Budget Alerts:**

- Alerts notify when specific thresholds are met.
- Alerts are sent to specified email addresses.

#### 8. **Importance of Budgets:**

- Budgets help avoid unexpected costs.
- Ensure financial control, especially when learning or experimenting with AWS services.

# Introduction to Amazon EC2

## What is Amazon EC2?

- **EC2** stands for **Elastic Compute Cloud**.
- It is a key part of AWS's **Infrastructure as a Service (IaaS)** offering.
- EC2 allows users to rent **virtual machines** (EC2 instances) and provides scalable computing capacity.

## Components of EC2

- **EC2 Instances:** Virtual machines that you can configure with different operating systems and hardware specs.
- **EBS Volumes:** Virtual drives attached to instances for persistent storage.
- **Elastic Load Balancer (ELB):** Distributes incoming traffic across multiple instances.
- **Auto-Scaling Group (ASG):** Automatically scales the number of instances based on demand.

## Operating Systems for EC2 Instances

- **Linux:** Most popular choice for EC2 instances.
- **Windows:** Available for applications requiring Windows environments.
- **MacOS:** Available for development needs specific to Apple environments.

## Key Configuration Options for EC2 Instances

- **Compute Power:** Choose the number of **vCPUs** and the type of processors.
- **Memory (RAM):** Define how much memory the instance should have.
- **Storage:**
  - **EBS:** Network-attached storage (persistent).
  - **EFS:** Elastic File System for shared access.
  - **Instance Store:** Storage physically attached to the host machine (non-persistent).
- **Networking:**
  - Choose **network interface** type and speed.
  - Assign **public or private IP** addresses.
- **Security Groups:** Define **firewall rules** to control inbound and outbound traffic.
- **User Data:** Bootstrap script executed at first launch to configure the instance.

### **Bootstrapping with EC2 User Data**

- **User Data:** Script that runs once during the first boot.
- **Purpose:** Automate tasks like:
  - Installing updates and software.
  - Downloading files.
- **Runs with Root Privileges:** Commands execute with `sudo` rights.

### **Example EC2 Instance Types**

- **t2.micro:**
  - **1 vCPU, 1 GB RAM.**
  - Low to moderate network performance.
  - Part of the **AWS Free Tier** (750 hours/month).
- **t2.xlarge:**
  - **4 vCPUs, 16 GB RAM.**
  - Moderate network performance.
- **c5d.4xlarge:**
  - **16 vCPUs, 32 GB RAM.**
  - **400 GB NVMe SSD** storage.
  - Up to **10 Gbps** network performance.
- **r5.16xlarge and m5.8xlarge:**
  - Higher specifications suitable for memory-intensive applications.

# Launching and Managing an EC2 Instance

## Launching Your First EC2 Instance

1. **Navigate to the EC2 Console:**
  - Click on **Instances**, then **Launch Instances**.
  
2. **Configure the Instance:**
  - **Name:** Set a name for the instance (e.g., My First Instance).
  - **Base Image:** Choose an **Amazon Machine Image (AMI)**.
    - Use **Amazon Linux 2 AMI** (free tier eligible).
  - **Architecture:** Select **64-bit x86**.
  
3. **Choose Instance Type:**
  - **Instance Types** vary by CPU, memory, and cost.
  - For this tutorial, use **t2.micro** (1 vCPU, 1 GB RAM, free tier eligible).
  - Other types are available, but not all are free tier eligible.
  
4. **Create a Key Pair:**
  - **Key Pair** is needed for SSH access.
  - Create a new key pair named **EC2 Tutorial**.
    - Choose **RSA** for encryption.
    - Select the key format:
      - .pem for Mac/Linux/Windows 10.
      - .ppk for older Windows versions (e.g., Windows 7, 8).
  - Download the key pair file.
  
5. **Network Settings:**
  - Default settings are usually sufficient.
  - Instance will get a **public IP**.
  
6. **Security Group:**
  - A **Security Group** defines firewall rules.
  - Default group created: `launch-wizard-1`.

- Allow **SSH traffic** (port 22) from anywhere.
- Allow **HTTP traffic** (port 80) for web server access.

## 7. Storage Configuration:

- Default storage is **8 GB gp2 root volume**.
- Free tier includes up to 30 GB of **EBS general-purpose SSD**.

## 8. Advanced Details:

- **User Data:** Script to run on the first launch (bootstrapping).
  - Used to install and configure software (e.g., web server).
- Example User Data script:

```
#!/bin/bash
yum update -y
yum install -y httpd
systemctl start httpd
systemctl enable httpd
echo "<html><h1>Hello World from $(hostname -f)</h1></html>" >
/var/www/html/index.html
```

## 9. Launch the Instance:

- Click on **Launch Instance**.
- Instance status will show as **pending**, then **running**.

## Accessing the EC2 Instance

### 1. View Instance Details:

- Public IP address: Use this to access the instance via the web.
- Private IP address: Used internally within AWS.

### 2. Test Web Server:

- Enter the public IP into the browser with http:// prefix.
- Example: http://3.250.x.x (ensure to use http, not https).
- Should display: Hello World from [private IP address].

## **Managing EC2 Instances**

### **1. Stopping an Instance:**

- Go to **Instance State** and select **Stop Instance**.
- Stopping an instance pauses it and stops billing for the instance.
- Data on EBS volumes is retained.

### **2. Terminating an Instance:**

- Go to **Instance State** and select **Terminate Instance**.
- This deletes the instance and associated volumes (if set to delete on termination).

### **3. Restarting an Instance:**

- Go to **Instance State** and select **Start Instance**.
- Note: Public IP may change after a restart, but the private IP remains the same.



## EC2 Instance Types

### Overview of EC2 Instance Types

- EC2 instances are categorized based on different **use cases** and **optimization** types.
- AWS provides a variety of instance types tailored to specific workload requirements.
- Instance types are referenced using a specific **naming convention**: e.g., **M5.2xlarge**.

### Naming Convention Breakdown

- **Instance Class (e.g., M)**: Defines the broad category of the instance, such as general purpose, compute optimized, etc.
- **Generation Number (e.g., 5)**: Represents the version of the hardware. Newer generations use higher numbers.
- **Size Modifier (e.g., 2xlarge)**: Indicates the size within the instance class. More significant sizes have more memory and CPU.

### Types of EC2 Instances

1. **General Purpose Instances:**
  - Best for a variety of workloads such as **web servers** and **code repositories**.
  - Provides a balance between **compute, memory, and networking** resources.
  - Examples: **T2.micro** (free tier eligible), **M5** instances.
2. **Compute Optimized Instances:**
  - Designed for **compute-intensive tasks** requiring high-performance CPUs.
  - Suitable for **batch processing, media transcoding, high-performance computing (HPC), machine learning, and gaming servers**.
  - Naming convention typically starts with **C** (e.g., **C5, C6**).
3. **Memory Optimized Instances:**
  - Optimized for tasks requiring **high memory** performance.
  - Ideal for **large in-memory databases, distributed caches, and real-time data processing**.
  - Naming conventions include **R** (stands for RAM), **X1, Z1** series.

#### 4. Storage Optimized Instances:

- Tailored for **high I/O operations** on local storage.
- Suitable for **online transactional processing (OLTP)**, **NoSQL databases**, **data warehousing**, and **distributed file systems**.
- Common naming conventions include **I**, **G**, and **H1** series.

### Comparison of Example Instance Types

- **t2.micro** (General Purpose):
  - **1 vCPU, 1 GB** memory.
  - Free tier eligible: up to **750 hours** per month.
- **r5.16xlarge** (Memory Optimized):
  - **64 vCPUs, 512 GB** memory.
  - Focused on high memory capacity for large data sets.
- **c5d.4xlarge** (Compute Optimized):
  - **16 vCPUs, 32 GB** memory.
  - Emphasizes CPU power for compute-heavy applications.

### Tools for Comparing Instance Types

- AWS provides detailed information on all instance types on their official website.
- **ec2instances.info** is a valuable external resource for comparing EC2 instances:
  - Lists all available EC2 instances.
  - Shows **cost** information for Linux on-demand and reserved instances.
  - Displays specs such as **memory**, **vCPU count**, and **network performance**.

## EC2 Security Groups

### Overview of Security Groups

- **Security Groups** act as virtual firewalls for EC2 instances.
- They control **inbound** and **outbound** traffic, ensuring only authorized connections are allowed.
- Security groups only contain **allow rules**; they do not support deny rules.
- Rules can reference specific **IP addresses** or other **security groups**.

### How Security Groups Work

1. **Inbound Rules:** Define what traffic is allowed to enter the EC2 instance.
  2. **Outbound Rules:** Define what traffic the EC2 instance is allowed to send out.
- Example Scenario:
    - A security group allows SSH traffic on **port 22** from a specific IP.
    - Outbound traffic is allowed by default unless specified otherwise.

### Security Group Details

- **Firewalls:** Security groups act as external firewalls around EC2 instances.
- **Regulate Access:**
  - Based on **ports** (e.g., 22 for SSH, 80 for HTTP).
  - Based on **IP ranges** (IPv4/IPv6).
- **Configuration:**
  - **Type:** Service type (e.g., SSH, HTTP).
  - **Protocol:** Type of protocol used (e.g., TCP, UDP).
  - **Port Range:** Port numbers allowed (e.g., 22 for SSH).
  - **Source:** IP range allowed to access (e.g., 0.0.0.0/0 for all IPs).

### Overview

- **EC2 Instance** with a security group:
  - If the security group allows inbound on port 22, only specified IP addresses can SSH into the instance.
  - Any unauthorized IP addresses will be blocked, causing a timeout.

- **Outbound Traffic:**
  - By default, EC2 instances can send outbound traffic to any IP.

## **Best Practices with Security Groups**

- **Multiple Attachments:** Security groups can be attached to multiple instances and vice versa.
- **Region and VPC Specific:** Security groups are scoped to a particular **region/VPC**.
- **External Firewall:** Security groups operate outside of the EC2 instance, blocking unauthorized traffic before it reaches the instance.
- **Separate SSH Security Group:** It is advisable to maintain a separate security group just for SSH access for better security management.

## **Common Port Numbers for Security Groups**

1. **SSH (Secure Shell):**
  - **Port 22:** Used to log into Linux EC2 instances.
2. **FTP (File Transfer Protocol):**
  - **Port 21:** Used for uploading files to a file share.
3. **SFTP (Secure File Transfer Protocol):**
  - **Port 22:** Secure version of FTP using SSH.
4. **HTTP (HyperText Transfer Protocol):**
  - **Port 80:** Used for accessing unsecured websites.
5. **HTTPS (HyperText Transfer Protocol Secure):**
  - **Port 443:** Used for accessing secured websites.
6. **RDP (Remote Desktop Protocol):**

- **Port 3389:** Used to log into Windows instances.

## Advanced Security Group Configuration

- **Referencing Other Security Groups:**
  - Security groups can reference other security groups to allow communication between EC2 instances.
  - Example: EC2 instances with **Security Group 1** can communicate if **Security Group 1** allows inbound traffic from itself or **Security Group 2**.
  - **Cross-Instance Communication:** Instances with attached security groups that reference each other can communicate seamlessly without specifying IP addresses.

## Accessing Security Groups

- **Security Groups** can be accessed and managed via the **EC2 Console** under the **Networking and Security** section.
- Each security group has a unique **ID** and can be associated with one or multiple EC2 instances.

## Inbound Rules

- **Inbound Rules** define the traffic allowed to enter the EC2 instance.
- Rules include:
  - **Type:** The type of traffic (e.g., SSH, HTTP).
  - **Port Range:** The specific port or port range allowed (e.g., port 22 for SSH, port 80 for HTTP).
  - **Source:** Specifies the IP range or security group allowed to access the port (e.g., 0.0.0.0/0 for anywhere).

### **Example: HTTP Access**

- **HTTP Rule:**
  - **Type:** HTTP
  - **Port:** 80
  - **Source:** Anywhere (0.0.0.0/0)

- **Deleting the Rule:**
  - Deleting the HTTP rule results in a **timeout** when trying to access the web server (e.g., infinite loading).
- **Restoring the Rule:**
  - Re-adding the HTTP rule restores access to the web server.

## **Troubleshooting Timeouts**

- **Timeout Issues:**
  - **Timeouts** occur when security group rules are incorrectly configured.
  - Common scenario: Missing or incorrect inbound rules cause timeouts.
- **Solution:**
  - Verify and correct security group rules to resolve timeout issues.

## **Outbound Rules**

- **Outbound Rules** define the traffic allowed to leave the EC2 instance.
- Default configuration:
  - **All Traffic Allowed:** Allows all outbound traffic to any destination (0.0.0.0/0).

## **Managing Multiple Security Groups**

- **Multiple Attachments:**
  - An EC2 instance can have **multiple security groups** attached.
  - The rules from all attached security groups are combined.
- **Reusability:**
  - A single security group can be attached to multiple EC2 instances.

# **Connecting to EC2 Instances: SSH and EC2 Instance Connect**

## **Overview**

- **Connecting to EC2 Instances** is crucial for performing maintenance or actions on your servers.
- **SSH (Secure Shell)** is the primary method used for secure command-line access to Linux servers.

## **Connection Methods Based on Operating System**

### **1. Mac/Linux Users:**

- **SSH:** Use the built-in terminal to establish an SSH connection to your EC2 instances.
- **Command Example:**

```
ssh -i /path/to/your-key.pem ec2-user@your-ec2-public-ip
```

### **2. Windows Users:**

- **Windows 10+:**
  - **SSH:** Use the Windows command prompt or PowerShell to connect via SSH.
- **Windows Versions <10:**
  - **Putty:** Use **Putty**, a free and open-source terminal emulator, to connect using the SSH protocol.
  - **Note:** Putty requires converting the .pem key file to a .ppk file format using puttygen.

### **3. All Users:**

- **EC2 Instance Connect:**
  - **Web-Based:** Connect directly to your EC2 instance using your web browser without needing a terminal or additional software.
  - **Supported for:** Amazon Linux 2 instances.
  - **Advantages:**
    - No installation required.
    - Compatible with Mac, Linux, and all versions of Windows.
    - Simplifies the connection process, especially for beginners.

## **Common Troubleshooting Tips**

- **Security Group Issues:**
  - Ensure your **inbound rules** allow SSH (port 22) from your IP address.
- **Command Errors:**
  - Double-check the SSH command for typos or incorrect file paths.
- **Instance Status:**
  - Confirm the instance is running and accessible via its public IP.



# SSH Connection to EC2 Instances on Windows

## Steps to SSH into EC2 Instance

### 1. Navigate to the Directory of PEM File:

- Use `cd` to change the directory where the `.pem` file is located.
- Example:

```
cd .\Desktop
```

- Confirm the presence of the `.pem` file using `ls`.

```
ls
```

### 2. Execute the SSH Command:

- Command structure:

```
ssh -i path/to/your-key.pem ec2-user@your-ec2-public-ip
```

- Example:

```
ssh -i EC2Tutorial.pem ec2-user@<your-ec2-public-ip>
```

- **Notes:**

- `-i` specifies the identity file (your PEM file).
- `ec2-user` is the default username for Amazon Linux.
- `<your-ec2-public-ip>` is your EC2 instance's public IP address.

- **Authentication:**

- On first connection, you'll be asked to confirm the authenticity of the host. Type `yes` to continue.

### 3. Handling Permission Issues:

- **Exit SSH:** If permission issues occur, exit by typing `exit`.

- **Fix Permissions:**

1. Right-click on the `.pem` file and select **Properties**.
2. Go to the **Security** tab and click on **Advanced**.
3. **Change Owner:**
  - Click **Change** and select your username.
  - Make sure you have full control over the file.
4. **Remove Inherited Permissions:**
  - Disable inheritance and remove all inherited permissions.
  - Add yourself as the only principal with full control.
5. Apply changes and exit the properties.

4. **Reconnect via SSH:**

- Retry the SSH command:

```
ssh -i EC2Tutorial.pem ec2-user@<your-ec2-public-ip>
```

- If permissions are correctly set, you should not encounter further issues.

5. **Alternative: Using Command Prompt:**

- You can also use the **Command Prompt** to execute the SSH command, provided you are in the correct directory where the `.pem` file is located.

### **Exiting SSH Session**

- Type `exit` or use **Control + D** to terminate the SSH session.

# **SSH Troubleshooting**

## **1) Connection Timeout**

- **Issue:** **Connection timeout** when trying to SSH.
- **Cause:** This is typically a **security group** issue.
- **Solution:**
  - Ensure your **security group** is correctly configured.
  - The security group should allow **inbound SSH traffic** on **port 22**.
  - Verify that the security group is properly assigned to your EC2 instance.

## **2) Persistent Connection Timeout**

- **Issue:** Connection timeout persists even after verifying the security group.
- **Cause:** Likely due to a **corporate firewall** or **personal firewall** blocking the connection.
- **Solution:**
  - Consider using **EC2 Instance Connect** (covered in the next lecture) to bypass firewall issues.

## **3) SSH Command Not Found on Windows**

- **Issue:** The error message “**ssh command not found**” appears.
- **Cause:** SSH is not installed on your Windows machine.
- **Solution:**
  - Use **PuTTY** instead for SSH access.
  - If PuTTY also fails, use **EC2 Instance Connect**.

## **4) Connection Refused**

- **Issue:** The error message “**Connection refused**” appears.
- **Cause:** The instance is reachable, but no **SSH service** is running on it.
- **Solution:**
  - **Restart** the EC2 instance.
  - If restarting doesn't help, **terminate** the instance and launch a new one.
  - Ensure you're using **Amazon Linux 2** for the new instance.

## 5) Permission Denied (publickey, gssapi-keyex, gssapi-with-mic)

- **Issue:** The error message “Permission denied (publickey, gssapi-keyex, gssapi-with-mic)” appears.
- **Causes & Solutions:**
  1. **Incorrect Security Key:**
    - Ensure you are using the correct **security key** associated with your EC2 instance.
  2. **Incorrect User:**
    - Make sure you're using the correct **username** (ec2-user for Amazon Linux 2).
    - Example command:

```
ssh -i your-key.pem ec2-user@<public-ip>
```

## 6) Connection Worked Previously but Fails Now

- **Issue:** SSH connection worked before but no longer works.
- **Cause:** Likely due to a **change in the public IP** after stopping and restarting the EC2 instance.
- **Solution:**
  - Update your SSH command or PuTTY configuration with the **new public IP**.
  - Example:

```
ssh -i your-key.pem ec2-user@<new-public-ip>
```

## EC2 Instance Connect: An Alternative to SSH

- **EC2 Instance Connect** is a browser-based SSH session to access your EC2 instances, offering a more straightforward alternative to traditional SSH methods.

### Steps to Use EC2 Instance Connect

1. **Accessing the EC2 Instance:**
  - Select **Instance** in the AWS Console.
  - Click on **Connect** at the top of the instance details page.
  - Choose the **EC2 Instance Connect** option.
2. **Verification:**
  - Ensure the **public IP address** is correct.
  - The **username** is set to `ec2-user` by default (auto-detected for Amazon Linux 2).
  - No need to manage SSH keys; AWS handles this by uploading a temporary SSH key.
3. **Connecting:**
  - Click **Connect** to open a new tab with an SSH session in your browser.
  - You can run basic commands like `whoami` or `ping google.com` to test connectivity.

### Advantages of EC2 Instance Connect

- **No SSH key management:** AWS automatically handles the temporary SSH key, simplifying the connection process.
- **Browser-based session:** Eliminates the need for a separate terminal or SSH client.

### Security Group Considerations

- **Port 22 Requirement:**
  - Ensure **Port 22 (SSH)** is open in your **security group** for EC2 Instance Connect to work.

- If Port 22 is blocked, EC2 Instance Connect will fail to establish a connection.
- **Editing Security Group Rules:**
  - If SSH access is removed:
    - Go to **Security Groups > Inbound Rules > Edit Rules**.
    - Re-add the **SSH (Port 22)** rule for **IPv4** (and **IPv6** if necessary).
- **Test Connection:**
  - After updating the security group, reconnect using **EC2 Instance Connect**.

### **Practical Example:**

- **Disabling and Re-enabling SSH:**
  - **Disable** SSH by removing the SSH rule in the security group.
  - **Re-enable** SSH by adding the rule back.
  - Retry connecting using EC2 Instance Connect to confirm access is restored.

## Using IAM Roles for EC2 Instances

### Steps to Connect to EC2 Instance

#### 1. Connect:

- Use **EC2 Instance Connect** for simplicity.
- Ensure you're logged in as `ec2-user@<private IP>`.
- Confirm connection by running basic Linux commands:

```
ping google.com
```

- Use `Ctrl + C` to stop commands, and `clear` to clean the terminal.

#### 2. Check AWS CLI Installation:

- Amazon Linux AMI comes pre-installed with **AWS CLI**.
- Test with:

```
aws iam list-users
```

- If you receive an error: *"Unable to locate credentials"*, **DO NOT** configure credentials manually using `aws configure`.

### Why Not Use AWS Configure?

- **Security Risk:** Storing personal AWS credentials (Access Key ID and Secret Access Key) on the EC2 instance exposes them to potential misuse by others who might access the instance.
- **Best Practice:** Always use **IAM Roles** for securely managing credentials.

### Attaching an IAM Role to EC2 Instance

#### 1. Access IAM Roles in the Management Console:

- Navigate to **IAM > Roles**.
- Locate the **DemoRoleForEC2** with the **IAMReadOnlyAccess** policy attached.

#### 2. Modify IAM Role of EC2 Instance:

- Go to your **EC2 Instances**.

- Select **Actions > Security > Modify IAM Role**.
- Choose **DemoRoleForEC2** from the dropdown and click **Save**.

### 3. Test IAM Role:

- Re-run:

```
aws iam list-users
```

- You should now see the list of IAM users, confirming the IAM role is properly attached.

## Verifying IAM Role Permissions

### 1. Detach IAM Role Policy:

- Detach **IAMReadOnlyAccess** from the **DemoRoleForEC2** in the IAM console.
- Re-run the command:

```
aws iam list-users
```

- You should receive an *access denied* error, indicating the role's permissions are essential.

### 2. Re-Attach IAM Role Policy:

- Re-attach **IAMReadOnlyAccess** to the role.
- Retry the command to ensure the role functions correctly once the policy is re-attached.



## **EC2 Instance Purchasing Options**

AWS offers multiple EC2 instance purchasing options to optimize costs based on workload types and duration. Understanding these options is crucial for cost management and efficient resource usage.

### **1. On-Demand Instances**

- **Use Case: Short-term, unpredictable workloads.**
- **Billing:** Pay by the second (Linux/Windows) or by the hour (other OS) with no upfront cost.
- **Advantages:**
  - **Highest flexibility** with no long-term commitment.
  - **Recommended** for **uninterrupted workloads** where usage patterns are unpredictable.

### **2. Reserved Instances**

- **Use Case: Long-term, steady-state workloads** (e.g., databases).
- **Term Options:** **1-year** or **3-year** commitments.
- **Discount:** Up to **72% off** compared to On-Demand.
- **Payment Options:**
  - **All upfront** (maximum discount)
  - **Partial upfront**
  - **No upfront**
- **Flexibility:** Convertible Reserved Instances allow changes in **instance type, family, OS, scope, and tenancy** but with reduced discounts (up to **66%**).

### **3. Savings Plans**

- **Use Case: Long-term** usage with **flexibility**.
- **Commitment:** Commit to spending a specific dollar amount (e.g., \$10/hour) for **1-3 years**.
- **Discount:** Similar to Reserved Instances (up to **72%**).
- **Flexibility:**
  - Across **instance size** (e.g., m5.xlarge to m5.2xlarge).
  - Across **OS** and **tenancy**.

#### 4. Spot Instances

- **Use Case:** **Short, flexible workloads** (e.g., batch jobs, data analysis).
- **Discount:** Up to **90% off** compared to On-Demand.
- **Risk:** Instances can be terminated anytime if the **spot price** exceeds the **max price** you're willing to pay.
- **Not Suitable:** **Critical** workloads or **databases**.

#### 5. Dedicated Hosts

- **Use Case:** **Compliance requirements** or **BYOL (Bring Your Own License)** for software.
- **Billing:**
  - **On-Demand** (per second)
  - **Reservation** (1-year or 3-year term)
- **Characteristics:**
  - **Entire physical server** dedicated to your use.
  - **Most expensive** option due to physical hardware reservation.

#### 6. Dedicated Instances

- **Use Case:** Instances requiring **hardware isolation**.
- **Characteristics:**
  - Runs on hardware dedicated to your use.
  - **No control** over instance placement (unlike Dedicated Hosts).
- **Comparison:**
  - **Dedicated Instances** share the underlying hardware within the same account.
  - **Dedicated Hosts** provide access to the **physical server** itself.

#### 7. Capacity Reservations

- **Use Case:** **Short-term** workloads that require **guaranteed capacity** in a specific **Availability Zone (AZ)**.
- **Characteristics:**
  - Reserve capacity for any duration with **no billing discount**.
  - Charged at On-Demand rates whether instances are running or not.
  - Can be combined with **Reserved Instances** or **Savings Plans** for discounts.

## Comparison Summary

- **On-Demand:** Flexibility with higher cost, suited for unpredictable workloads.
- **Reserved:** Best for long-term, predictable workloads with up to **72% discount**.
- **Savings Plans:** Flexibility with specific spending commitments, similar discounts to Reserved Instances.
- **Spot:** Cost-effective but risky, suited for non-critical tasks.
- **Dedicated Hosts:** Full server reservation for compliance or specific software needs.
- **Dedicated Instances:** Isolated instances on dedicated hardware without full server control.
- **Capacity Reservations:** Guarantees capacity in a specific AZ, with no cost-saving benefits.

## Analogy for Understanding

- **On-Demand:** Walk-in hotel booking at full price.
- **Reserved:** Booking long-term stay with a discount.
- **Savings Plan:** Prepaying for a specific amount of spending each month.
- **Spot:** Last-minute hotel booking with the risk of being bumped.
- **Dedicated Host:** Reserving the entire hotel.
- **Capacity Reservation:** Holding a hotel room without a commitment to stay but paying full price.

## Pricing Example (m4.large in us-east-1)

- **On-Demand:** \$0.10 per hour.
- **Spot:** Up to 61% off On-Demand.
- **Reserved:** Varies based on term and payment options.
- **Savings Plan:** Similar to Reserved.
- **Dedicated Host:** On-Demand price for physical server.
- **Capacity Reservation:** On-Demand price without a discount.

## **EC2 Spot Instances**

**Spot Instances** provide significant cost savings, up to **90%** compared to On-Demand instances. They are ideal for workloads that can tolerate interruptions, such as batch jobs and data analysis. Here's how they work and how to optimize their use:

### **How Spot Instances Work**

- **Max Spot Price:** You define the maximum price you're willing to pay for a spot instance. As long as the current spot price is below this, the instance remains active.
- **Hourly Spot Price:** The spot price varies based on demand and available capacity.
- **Two Options When Price Exceeds Max:**
  1. **Stop the Instance:** The instance is paused and can be restarted when the price drops below the max again.
  2. **Terminate the Instance:** The instance is permanently terminated, and you'll need to start fresh if you want to continue.

**Spot Block:** You can block a spot instance for a specified duration (1-6 hours) to prevent interruptions. However, in rare cases, AWS might still reclaim the instance.

### **Use Cases**

- **Batch Jobs**
- **Data Analysis**
- **Workloads Resilient to Failure**

### **Not Suitable For:**

- **Critical Jobs**
- **Databases**

### **Spot Instance Pricing**

- **Example:** Pricing for an `m4.large` instance in `us-east-1`:
  - **On-Demand Price:** \$0.10 per hour.

- **Spot Price:** Typically lower (e.g., \$0.04), leading to significant savings.
- **Price Fluctuations:** Spot prices can fluctuate over time, and they vary across Availability Zones (AZs).

### Strategy:

- Set a max price just above the usual spot price to avoid frequent interruptions.
- Monitor the spot price trends in your selected AZ.

### Managing Spot Requests

- **Spot Request:** You specify how many instances you want, max price, launch specifications, and the duration.
- **Request Types:**
  - **One-Time:** Launches instances once and doesn't reattempt if they are terminated.
  - **Persistent:** Keeps the request active, automatically launching new instances if the current ones are terminated.

### Terminating Spot Instances:

- **Step 1:** Cancel the spot request to prevent AWS from launching new instances.
- **Step 2:** Terminate the running spot instances.

### Spot Fleets

- **Purpose:** Optimize cost savings by launching a fleet of instances, possibly including both spot and On-Demand instances.
- **Launch Pools:** Define multiple instance types, OS, and AZs.
- **Target Capacity:** The fleet tries to meet the specified capacity within your price constraints.
- **Allocation Strategies:**
  - **Lowest Price:** Prioritizes the cheapest pool, ideal for short workloads.
  - **Diversified:** Distributes instances across pools for better availability, ideal for long workloads.
  - **Capacity Optimized:** Chooses the pool with the best capacity for your needs.
  - **Price Capacity Optimized:** Balances capacity and price, selecting the pool with the highest capacity and the lowest price.

### **Spot Fleets vs. Spot Instances:**

- **Spot Instance Request:** You specify a single instance type and AZ.
- **Spot Fleet:** AWS intelligently selects from multiple instance types and AZs to maximize savings.

## EC2 Instance Launch Methods

AWS provides multiple methods to launch EC2 instances, each suited to different use cases, optimizing for cost, availability, and flexibility. Below are the key methods covered:

### 1. Spot Instances

- **Pricing History:**
  - Example: For `c4.large`, spot pricing shows significant savings (~69-70%) over time compared to On-Demand prices.
  - **Price Stability:** Spot prices generally remain low and stable but can vary by Availability Zone (AZ).
  
- **Spot Request:**
  - **Launch Parameters:** Define parameters like **OS**, **key pair**, and other standard EC2 settings.
  - **Request Details:** Set **max price**, **validity period**, and decide on **termination behavior** (terminate, stop, or hibernate) if the price exceeds your set limit.
  - **Target Capacity:** Specify the number of instances or **vCPUs** you require.
  - **Networking:** Choose specific **AZs** and **VPCs** for instance deployment.
  - **Instance Type:**
    - **Manual Selection:** Pick specific instance types like `c3.large`, `c4.large`, etc.
    - **Automatic Selection:** AWS selects instances based on attributes like **vCPUs** and **memory**.
  - **Allocation Strategy:**
    - **Capacity Optimized:** Ensures capacity is always available.
    - **Lowest Price:** Maximizes cost savings by selecting the lowest-priced instances.
  
- **Direct Spot Instance Launch:**
  - **Advanced Details:**
    - Request a spot instance and set a **max price**.
    - **Request Type:** Choose **one-time** (instance terminates after use) or **persistent** (instance restarts when the spot price is favorable).
    - **Interruption Behavior:** Choose between **hibernate**, **stop**, or **terminate** if the spot price exceeds the max price.

## 2. Reserved Instances

- **Overview:**
  - Purchase a reserved instance for specific instance types like `c5.large`.
  - Terms: **12 or 36 months**.
  - Payment Options: **All upfront**, **Partial upfront**, or **No upfront**.
  - **Flexibility: Convertible Reserved Instances** allow changes to instance type, family, OS, scope, and tenancy.

## 3. Savings Plans

- **Overview:**
  - Commit to a specific amount of spending per hour (e.g., \$10/hour) over **1-3 years**.
  - **Flexibility:**
    - Across **instance types**, **AZs**, and **OS**.
    - Adjust as needed without losing the discount.

## 4. Dedicated Hosts

- **Overview:**
  - Provides a dedicated physical server for compliance requirements or software licenses.
  - **Setup:**
    - Allocate a dedicated host, name it, and choose an **instance family** (e.g., `c5`) and **AZ**.
    - This option is managed through the **License Manager** for better license management.

## 5. Capacity Reservations

- **Overview:**
  - Reserve capacity in a specific AZ for a particular instance type (e.g., `m5.2xlarge`).
  - **Reservation:** Ensure availability regardless of whether you use the reserved instances or not.
  - **Payment:** Pay for the reservation, even if the instances are not launched.



By understanding these methods, you can choose the most cost-effective and suitable EC2 instance launch method for your workload.