Cloud Computing

Assignment No. 2

Name : Unnati Shendge

Class: CSAI(A), Batch: B1 Roll No. 61

Creating account with AWS and launching EC2 instance

**Amazon EC2 Overview**

**Amazon EC2 (Elastic Compute Cloud)** is a web service by Amazon Web Services (AWS) that provides scalable virtual servers in the cloud. It enables users to deploy and manage virtual machines, known as instances, to run applications with flexibility and efficiency. With EC2, you can quickly scale your computing capacity up or down based on your needs, making it ideal for a wide range of use cases, including web hosting, software development, and data processing.

**Key Features of Amazon EC2:**

1. **Scalability**:
   * Easily scale your compute resources up or down according to demand, ensuring optimal performance and cost-efficiency.
2. **Variety of Instance Types**:
   * Choose from a broad selection of instance types, each optimized for different workloads, such as general-purpose, compute-optimized, memory-optimized, and storage-optimized instances.
3. **Flexible Pricing Models**:
   * Pay as you go with On-Demand instances, save with long-term commitments using Reserved instances, or bid for lower prices with Spot instances.
4. **Global Reach**:
   * Deploy instances across multiple regions and availability zones worldwide, ensuring low latency, high availability, and fault tolerance.
5. **Security**:
   * Benefit from robust security features, including secure key pairs, Virtual Private Clouds (VPCs), security groups, and integration with AWS Identity and Access Management (IAM) for granular control over access to your instances.

Amazon EC2 offers the flexibility and power to run anything from simple websites to complex, large-scale applications, making it a fundamental service for businesses of all sizes.

**Launching an EC2 Instance**

To launch an EC2 instance, follow these steps:

1. **Sign In to AWS Management Console**: Navigate to the [AWS Management Console](https://aws.amazon.com/console/).
2. **Open EC2 Dashboard**: Go to the EC2 service under "Compute" in the AWS Management Console.
3. **Launch Instance**:
   * Click on the "Launch Instance" button.
   * Choose an Amazon Machine Image (AMI).
   * Select an Instance Type based on your needs.
   * Configure Instance Details like number of instances, network settings, etc.
   * Add Storage if needed.
   * Add Tags to organize your instances.
   * Configure Security Group settings for firewall rules.
   * Review and launch the instance.
4. **Select or Create a Key Pair**: Choose an existing key pair or create a new one for SSH access to your instance.
5. **Launch**: Click the "Launch" button to start the instance.
6. **Select or Create a Key Pair**: Choose an existing key pair or create a new one for SSH access to your instance.
7. **Launch**: Click the "Launch" button to start the instance.

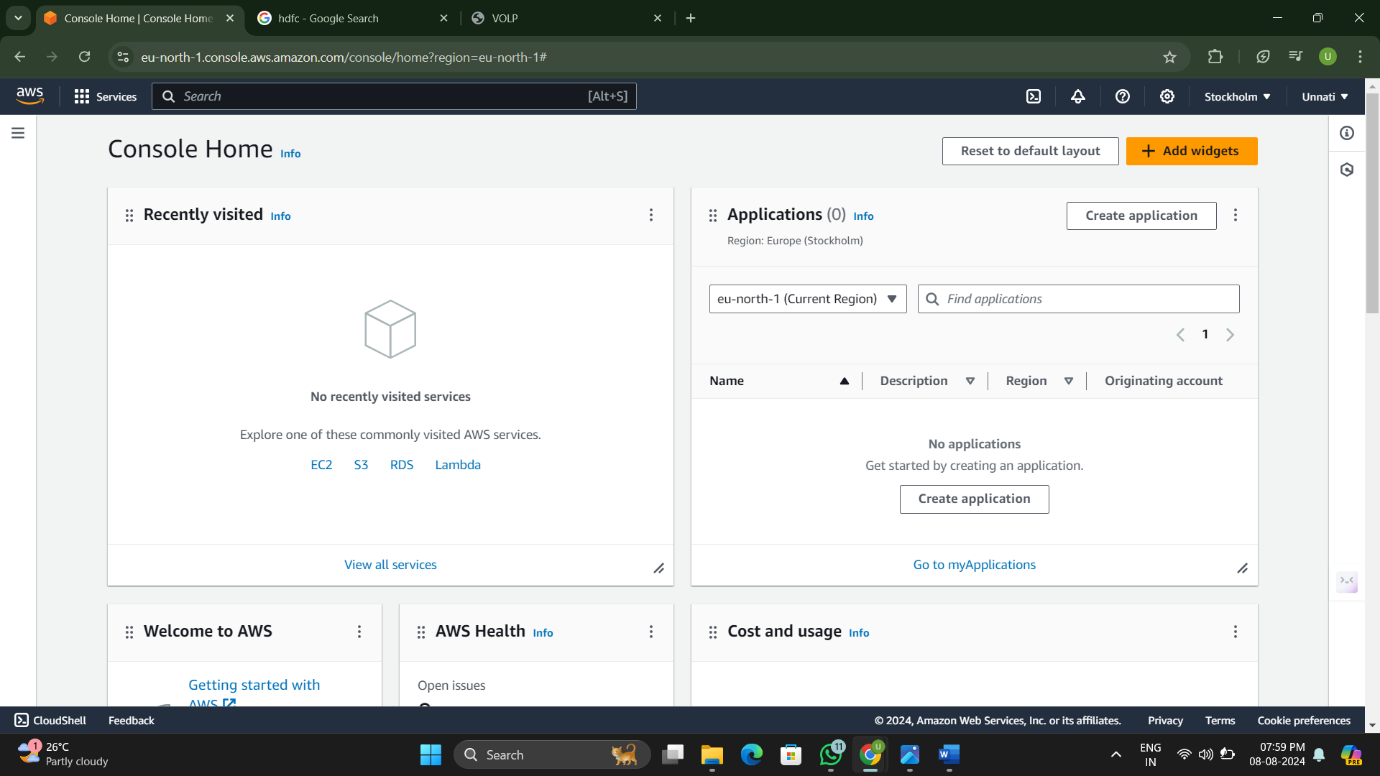
Below is the explanation with Screen Shoots:

**Step 1. Select a region**

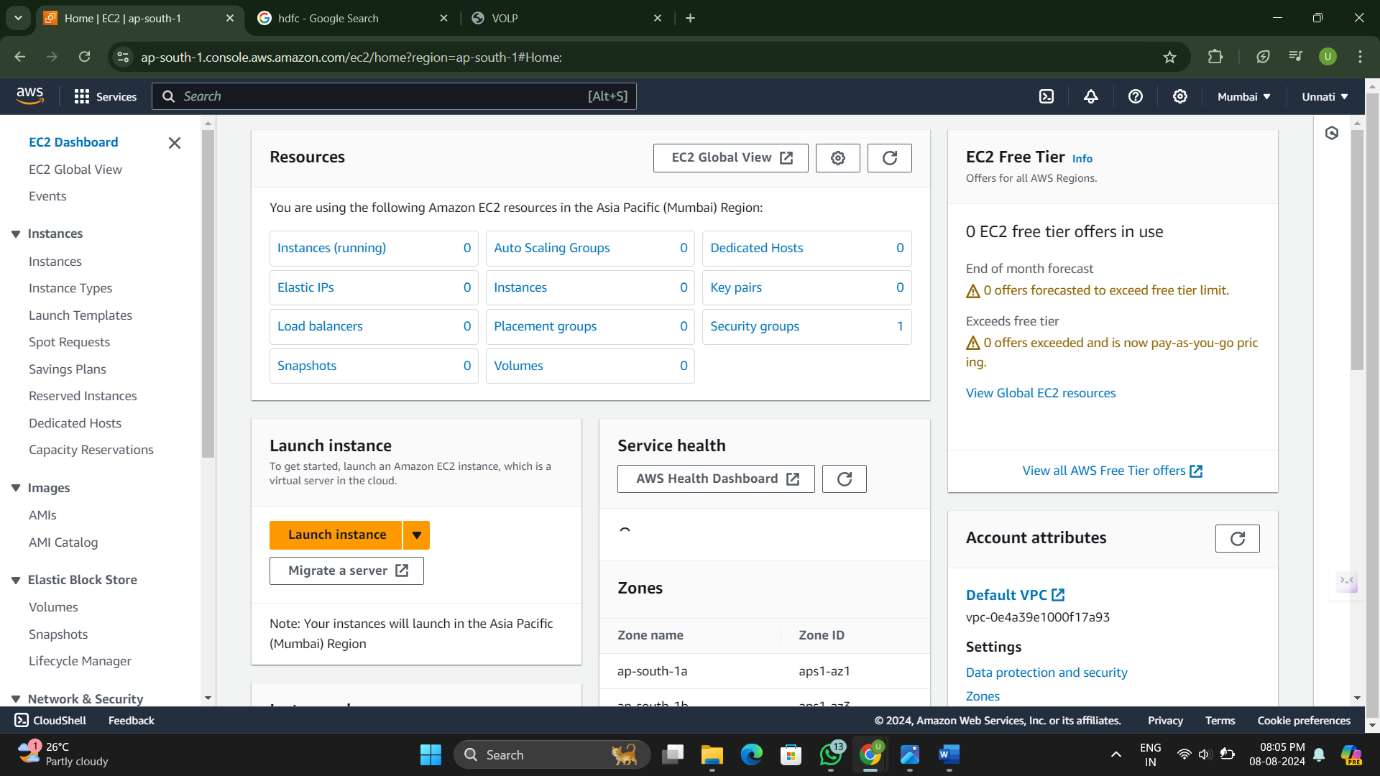
One of the most important steps is to select the desired AWS region. This can be done from the top navigation bar of the AWS Console

**Step 2. Navigate to the EC2 Console**

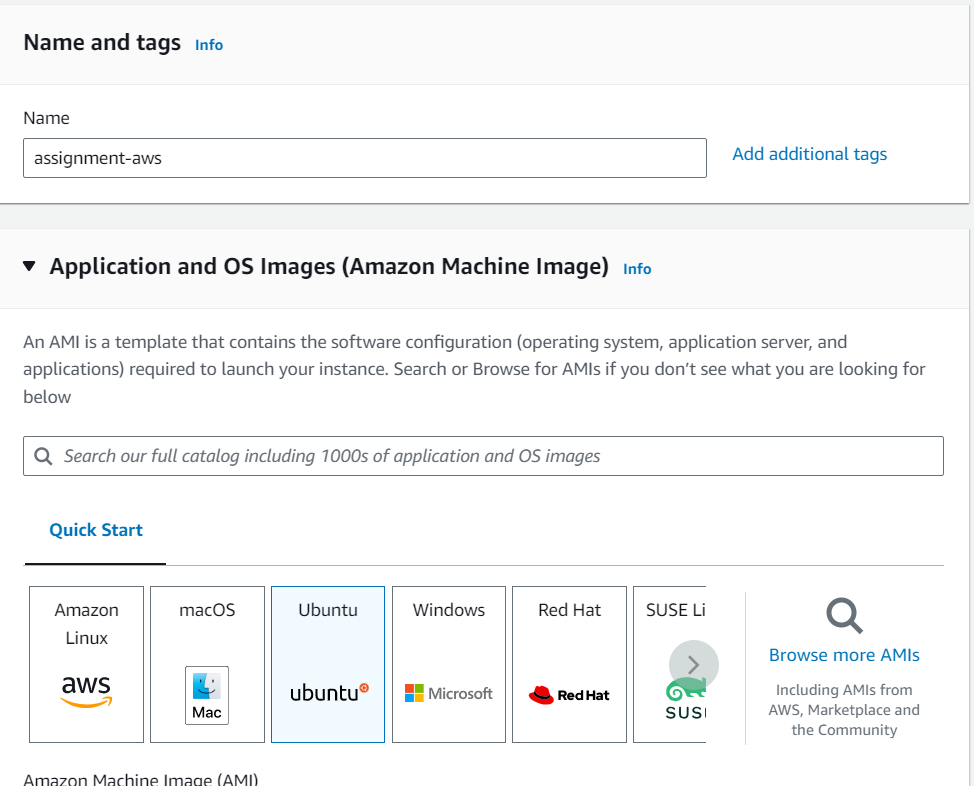
Once you select the desired AWS region, go to the EC2 Console. From the same landing page, search for EC2. Your landing page in the management interface will likely display EC2 among the most commonly used services in the account.



**Step 3: Launch Instance**  
In the EC2 console, click on the “Launch Instance” button to start the instance creation process. This will take you through a series of steps to configure your instance.  
After clicking on the “Launch Instance” button, you will be redirected to a page as shown below.



**Step 4: Provide a Name for the EC2 Instance**  
You need to provide a name for your EC2 instance. Giving your instance a meaningful name can help with easy identification and management of your resources in the future.

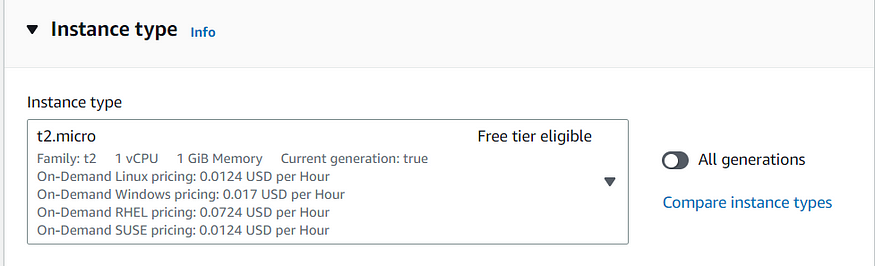


**Step 5: Choose an Amazon Machine Image (AMI)**  
An Amazon Machine Image (AMI) is a pre-configured template that contains the operating system and other software necessary for your instance. In this step, you can choose from a wide variety of AMIs provided by AWS or the AWS Marketplace. Select an AMI that best suits your requirements.  
In this case, you can choose “Amazon Linux” as the operating system for your instance.

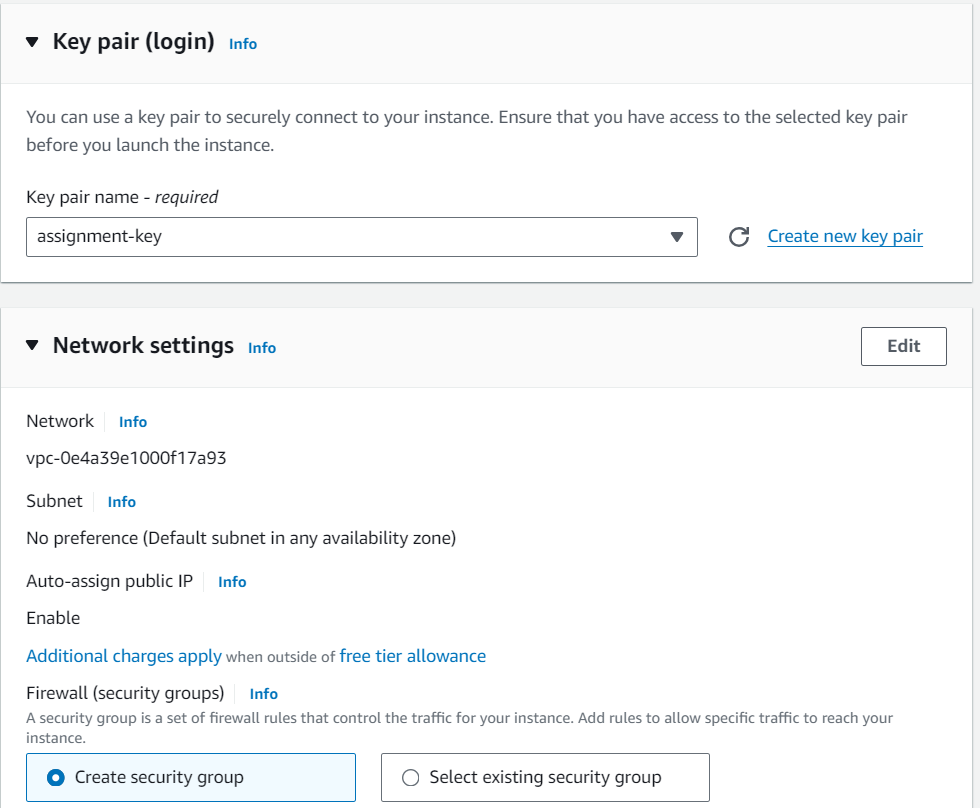
A screenshot of a computer

Description automatically generated

**Step 6: Choose an Instance Type**  
The instance type determines the hardware of the host computer used for your instance. AWS offers a range of Instance Type with varying compute, memory, and storage capabilities. Consider your workload requirements and select the instance type that meets your needs.  
In this case, you can select “t2.micro” as the instance type. The “t2.micro” instance type is eligible for the AWS free tier, which provides limited free usage of certain AWS services.

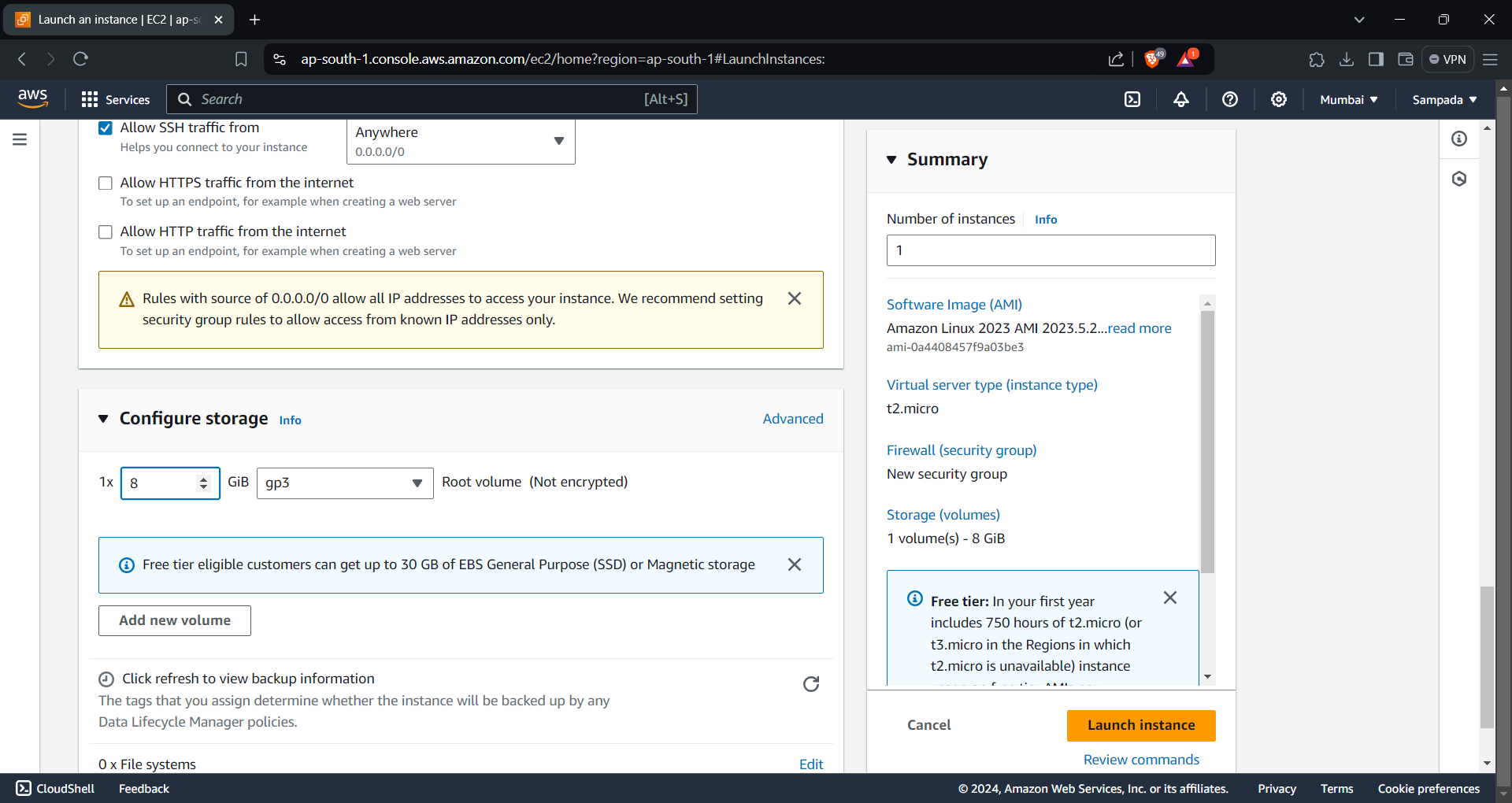


**Step 7: Create a Key Pair**To securely access your EC2 instance from your system, you need to create a key pair. A key pair consists of a public key that AWS stores and a private key that you download to your local machine. The private key is used to authenticate and establish a secure connection with your EC2 instance.

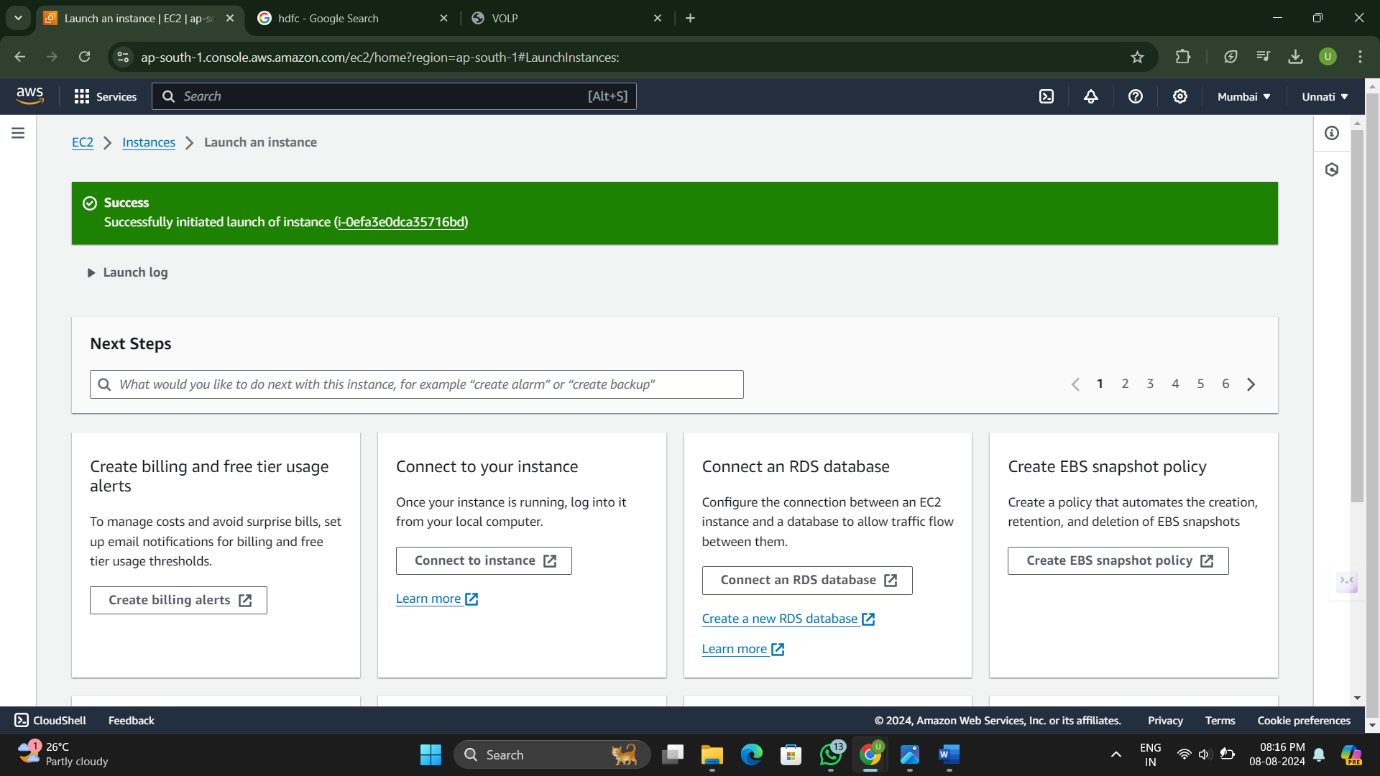


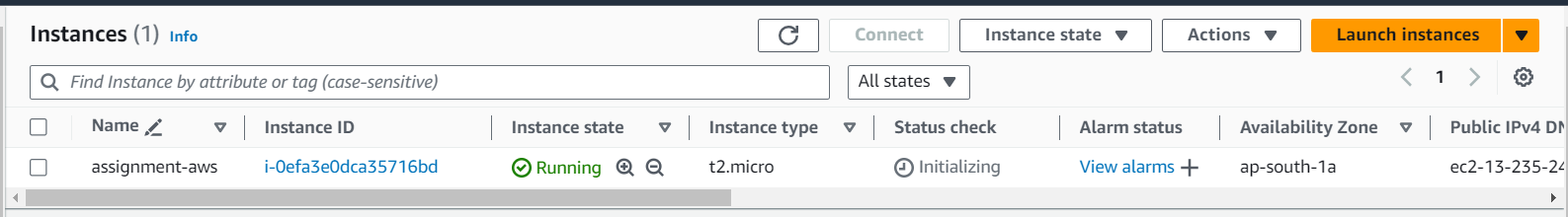
Upon clicking this button, a dialogue box will appear where you can provide a unique name for your key pair. Select the key pair type as “RSA” and choose the private key file format as “.pem”.  
Finally, click on the “Create Key Pair” button to generate your new key pair.

**Step 8: Configure Instance Details, Add Storage, and Configure Security Groups**In this step, you can configure various settings for your EC2 instance. Adjust these settings based on your specific requirements. You have the flexibility to update these details even after launching the instance.



**Step 9: Review and Launch**  
Before launching the instance, review the configuration details to ensure everything is set up correctly. Double-check the instance type, storage options, security groups, and any additional settings you have configured. If everything looks good, click on the “Launch Instance” button.





**Amazon Machine Images (AMIs)**

An **Amazon Machine Image (AMI)** is a crucial element in Amazon EC2, serving as a pre-configured blueprint used to launch virtual servers, known as EC2 instances. An AMI encapsulates everything necessary to set up an EC2 instance, including the operating system, system configurations, application server, and any additional software required to run your applications. This makes AMIs a foundational tool in cloud computing, enabling the rapid and consistent deployment of instances across various environments.

**Key Components of an AMI:**

* **Operating System (OS)**: The core software that manages hardware resources and provides common services for computer programs. The OS included in an AMI can vary from Linux distributions to Windows Server editions.
* **Application Server**: Software that handles requests from clients (e.g., web browsers, mobile apps) and serves responses, typically running applications like Apache, Nginx, or IIS.
* **Installed Applications**: Pre-installed software that can include anything from development tools and databases to security packages, tailored to specific use cases.

**Types of AMIs**

AMIs come in various types, catering to different needs, ranging from general-purpose computing to specialized workloads. Below are the basic categories:

1. **Amazon Linux AMI**:
   * **Description**: The Amazon Linux AMI is a Linux-based OS provided by AWS. It is specifically optimized for performance on Amazon EC2. The Amazon Linux AMI is designed to offer a stable, secure, and high-performance environment that closely integrates with AWS features and services.
   * **Key Features**:
     + Frequent updates with the latest security patches and software packages.
     + Tight integration with AWS, including optimized drivers for EC2.
     + Availability of multiple versions, allowing users to choose the most suitable one for their workload.
     + Includes cloud-init, a tool for initializing cloud instances.
2. **Ubuntu AMI**:
   * **Description**: Ubuntu is one of the most popular Linux distributions worldwide, known for its user-friendliness, vast community support, and regular updates. The Ubuntu AMI is an official Amazon EC2 image maintained by Canonical, the company behind Ubuntu.
   * **Key Features**:
     + Access to a vast repository of open-source software through Ubuntu’s package manager (APT).
     + Regularly updated AMIs with long-term support (LTS) versions, providing stability and security.
     + Pre-configured with common tools and utilities, making it ideal for developers and enterprises.
     + Strong community support and extensive documentation.
3. **Microsoft Windows AMI**:
   * **Description**: Microsoft Windows AMIs are tailored for running Windows-based applications on EC2 instances. These AMIs provide a familiar environment for users and applications that rely on the Windows ecosystem.
   * **Key Features**:
     + Available in multiple versions, including Windows Server editions (e.g., 2016, 2019) and Windows 10.
     + Integration with Microsoft services such as Active Directory, SQL Server, and .NET Framework.
     + Includes Windows Server licenses, so there’s no need to bring your own license (BYOL).
     + Pre-installed with AWS tools like the EC2Config service or EC2Launch, which are used for instance initialization and management.
4. **Custom AMI**:
   * **Description**: Custom AMIs are user-defined images that can be created based on specific requirements. These AMIs can be built by taking an existing AMI and modifying it, or by creating one from scratch using tools like Packer.
   * **Key Features**:
     + Custom configurations: Users can install their own software, configure the operating system, and apply specific settings before saving the image.
     + Consistent deployment: Custom AMIs ensure that every EC2 instance launched from the AMI is identical, which is crucial for maintaining consistency across environments (development, testing, production).
     + Secure and private: Custom AMIs can be shared within an organization or kept private to ensure that sensitive configurations and software are not exposed.
     + Version control: Users can create multiple versions of an AMI, allowing for controlled updates and rollbacks.

**AMI Lifecycle**

* **Creation**: AMIs can be created by taking a snapshot of an existing EC2 instance, using AWS Marketplace, or building from scratch using tools like Packer.
* **Storage**: AMIs are stored in Amazon S3, but users do not interact with S3 directly. Instead, AMIs are managed through the EC2 console, CLI, or API.
* **Sharing**: AMIs can be shared with other AWS accounts or made public for the community.
* **Deletion**: When no longer needed, AMIs can be deregistered, which will also delete the associated snapshots.

**Benefits of Using AMIs**

* **Rapid Deployment**: Launch instances quickly without having to manually install and configure software.
* **Scalability**: AMIs enable scaling by allowing multiple instances to be launched with the same configuration.
* **Customization**: Users can create AMIs tailored to their specific use cases, ensuring consistency across environments.
* **Cost-Efficiency**: By using optimized AMIs, you can ensure that your instances are running efficiently, which can lead to cost savings.

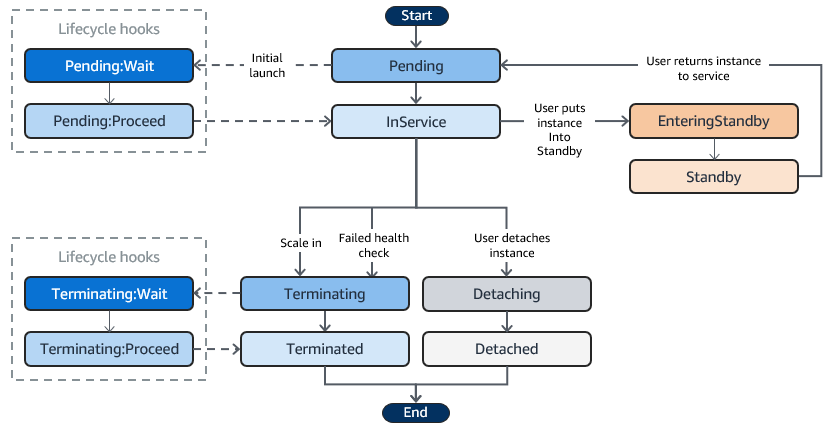
**EC2 Instance Types**

EC2 instances come in various types optimized for different tasks:

* **General Purpose**: Balanced compute, memory, and network resources. Example: t2.micro, t3.medium.
* **Compute Optimized**: High compute power for demanding applications. Example: c5.large.
* **Memory Optimized**: High memory capacity for data-intensive applications. Example: r5.xlarge.
* **Storage Optimized**: High storage throughput and capacity. Example: i3.large.
* **Accelerated Computing**: Instances with GPU support for high-performance tasks. Example: p3.2xlarge.

**Instance Lifecycle**

**Instance Lifecycle Diagram**:



**Start**:

* The instance lifecycle begins with the **Start** state.

**Pending**:

* When an EC2 instance is first launched, it enters the **Pending** state. During this phase, the instance is being initialized, with all the necessary resources being provisioned.
* **Lifecycle Hook - Pending**
  + A lifecycle hook can be applied at this stage, known as Pending:Wait. This hook pauses the instance's initialization, allowing you to perform custom actions, such as configuring the instance or installing software.
* **Pending**
  + After the custom actions are completed, the instance proceeds to the next state through the Pending:Proceed stage.

**InService**:

* Once the initialization is complete, the instance enters the **InService** state, indicating that it is now available to handle traffic. The instance remains in this state during normal operation.

**EnteringStandby & Standby**:

* **EnteringStandby**:
  + Users can manually move an instance to the **EnteringStandby** state, which is typically used for maintenance or scaling operations without terminating the instance.
* **Standby**:
  + After entering standby, the instance moves to the **Standby** state, where it is temporarily removed from service but not terminated. It does not handle any traffic or scale events during this time.
* The instance can be returned to the **InService** state after maintenance or adjustments, where it resumes handling traffic.

**Terminating**:

* Instances may be terminated due to scaling events (e.g., scaling in to reduce the number of instances) or failed health checks.
* **Lifecycle Hook - Terminating**

:

* + During the termination process, a lifecycle hook known as Terminating:Wait can be applied, pausing the termination process. This pause allows you to perform tasks such as saving logs, copying data, or sending notifications.
* **Terminating**

:

* + After completing the necessary tasks, the termination process proceeds through the Terminating:Proceed stage.

**Terminated**:

* The instance is fully shut down and removed from service. In this state, the instance is no longer running, and all associated resources are released.

**Detaching**:

* Users can manually detach an instance from the Auto Scaling group, which moves the instance to the **Detaching** state.
* **Detached**:
  + Once detached, the instance no longer belongs to the Auto Scaling group and will not participate in scaling events, but it remains running and can be managed independently.

**End**:

* The lifecycle concludes at the **End** state after the instance has been terminated or fully detached.

**Lifecycle Description**:

* **Pending**: The instance is being set up.
* **Running**: The instance is up and operational. You can connect and use it.
* **Stopping**: The instance is transitioning to a stopped state. It is no longer running but can be started again.
* **Stopped**: The instance is shut down. You are not billed for instance usage, but storage costs apply.
* **Terminated**: The instance is permanently deleted. Data stored in the instance store is lost, and it cannot be restarted.