NAME: AFIFA TASKEEN

USN:4MH23CA002

TITLE: DEFINITION AND TYPES OF CLOUD SERVICES: SaaS, PaaS, IaaS

SUBMISSION DATE:18 November 2025

INTRODUCTION

Cloud computing is a technology in which, instead of buying and managing our own hardware and software, we use IT resources such as servers, databases, and storage over the internet. This makes it easier for users to access services from anywhere without worrying about maintenance or physical infrastructure. Cloud computing also helps reduce cost, improves scalability, and allows businesses to use powerful tools on demand. Because different users have different needs, cloud computing provides multiple service models that offer various levels of control and convenience. Some users need ready-made software, some need a platform to build applications, and others need full control over virtual machines. These service models help users choose exactly what level of service they require without investing in physical systems. The three main service models in cloud computing are Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS).

OBJECTIVES

1. To understand the concept and meaning of cloud computing.
2. To learn the definitions of SaaS, PaaS, and IaaS.
3. To study how each cloud service model works.
4. To identify real-world examples of each model.
5. To compare the differences between SaaS, PaaS, and IaaS

BACKGROUND

What is Cloud Computing?

Cloud computing is a model that provides easy and on demand access to shared computing resources such as servers, storage, databases, networks, and software over the internet. These resources can be quickly provided and released with very little management effort. Cloud computing removes the need for organizations to buy and maintain physical hardware, making computing more flexible, scalable, and costeffective.

What Are Cloud Service Models?

Cloud service models are the different types of services that cloud providers offer. They explain how much of the work the cloud provider does and how much the user has to manage. Each model gives a different level of control and convenience to the user.

The three cloud service models are:

1. IaaS – infrastructure
2. PaaS – platform for developers
3. SaaS – ready-made software

Why Do We Need Cloud Service Models?

We need cloud service models because:

- Different users need different things.
  Some want full control (IaaS), others want a platform to build apps (PaaS), and some only need ready-made apps (SaaS).
- They save money.
  Users don’t have to buy expensive hardware or software.
- They make work easier.
  No installation, maintenance, or setup needed for many services.
- They give flexibility.
  Users can choose the exact type of service they need.
- They help scale quickly.
  Resources can increase or decrease based on demand.

Infrastructure as a service (IaaS)

Definition

Infrastructure as a Service (IaaS), also called Hardware as a Service (Haas), is a cloud model where users rent computing resources instead of buying physical hardware. It provides virtual servers, storage, networks, and complete infrastructure on demand.

Main Concept

- IaaS works by creating Virtual Machines on top of physical servers using virtualization.
- Users can install their own operating systems and applications inside these VMs.
- Each VM is billed based on its configuration, such as memory size, number of processors, and storage.

Features

- IaaS provides on-demand virtual machines that can be created and deleted easily.
- It supports flexible scaling, allowing resources to increase or decrease based on need.
- Users get full control over the system software and applications they install.
- It follows a pay-as-you-go billing model, so users only pay for what they use.
- IaaS includes automatic load balancing and backup options to improve reliability.

Architecture of IaaS

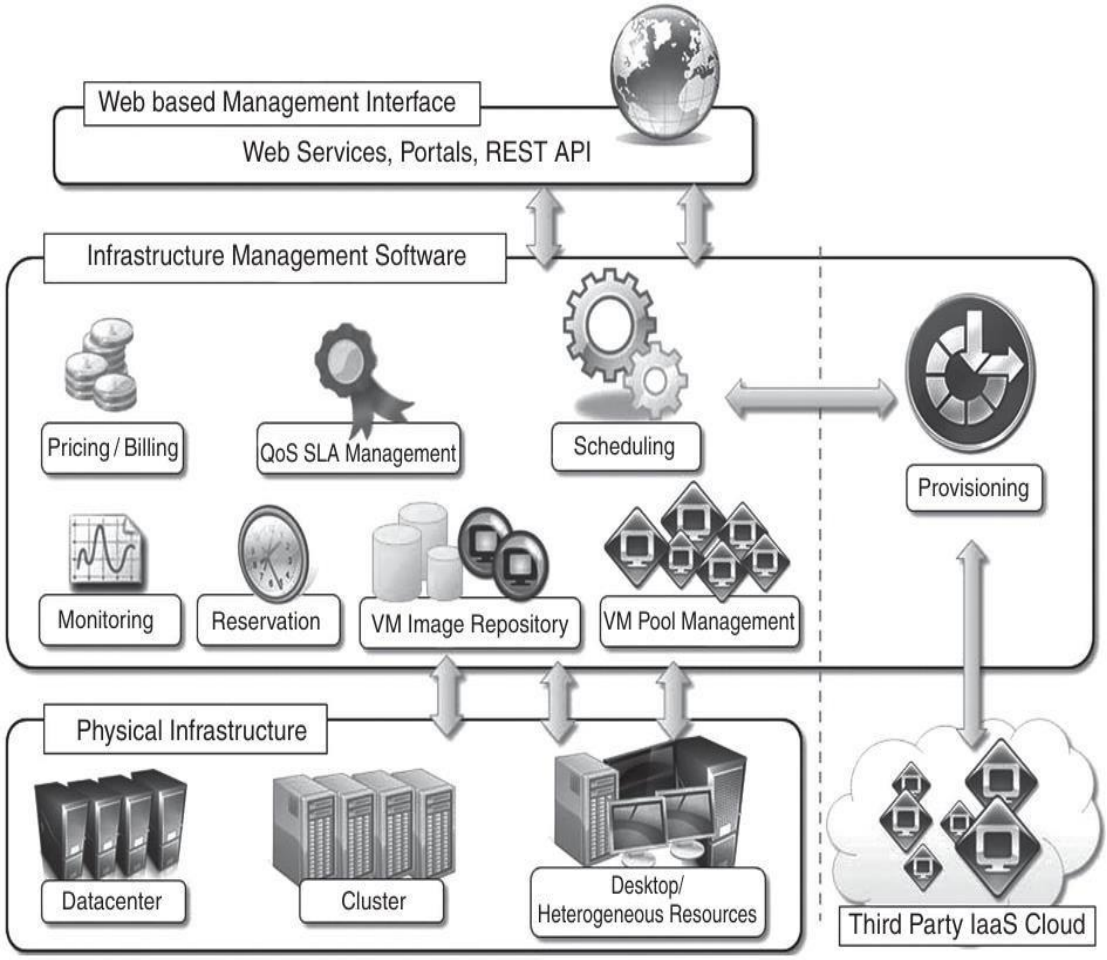


Figure: Architecture of Infrastructure as a Service (IaaS)

1. Physical Infrastructure

- This layer contains physical hardware such as servers, storage devices, and data centers.
- The cloud provider manages all the physical resources and their maintenance.
- Virtualization is used to share the hardware across multiple virtual machines.

2. Infrastructure Management Software

- This layer manages the creation, monitoring, and billing of virtual machines.
- The scheduler allocates hardware resources to VMs.
- The monitoring system tracks performance and ensures smooth operation.
- Billing software calculates the cost based on VM usage.
- A VM image repository stores templates that users can select when creating VMs.

3. Web-Based Management Interface

- This layer provides dashboards, portals, and APIs that users interact with.
- Users can create, configure, and manage virtual machines through this interface.
- It also allows users to check their resource usage and integrate cloud services into applications.

Advantages

For Customers

- Customers save money because they do not need to buy or maintain physical hardware.
- They can scale their resources up or down easily based on usage.
- They have full customization options for their virtual machines.

For Providers

- Providers get better utilization of their hardware.
- They can manage everything centrally and ensure security for customers.

Examples

- Amazon EC2 (AWS)
- Google Compute Engine
- Microsoft Azure Virtual Machines

Platform as a service (PaaS)

Definition

- Platform as a Service (PaaS) is a cloud service model that provides a complete platform for developing, testing, deploying, and managing applications.
- It acts as a middleware layer between the user and the underlying infrastructure.
- Developers can focus mainly on writing application code without worrying about hardware, operating systems, or network settings.

Main Concept

- PaaS provides a ready-made environment that includes tools, programming languages, libraries, and runtime systems needed to build applications.
- The cloud provider manages everything below the application level, including servers, networks, OS, databases, and middleware.
- Developers only manage the application logic, which simplifies the development process.
- PaaS ensures automatic scaling, monitoring, updating, and provisioning of resources.

Features

- PaaS offers a built-in runtime environment for running applications.
- It handles resource allocation such as CPU, memory, and storage automatically.
- It provides elasticity, meaning the platform automatically scales applications based on demand.
- PaaS manages the deployment of applications, including updates and performance tuning.
- It offers user management, security, and access control.
- Quality of Service (QoS) and billing functions are integrated.
- Developers can access APIs, SDKs, and programming libraries for faster development.

Architecture of PaaS

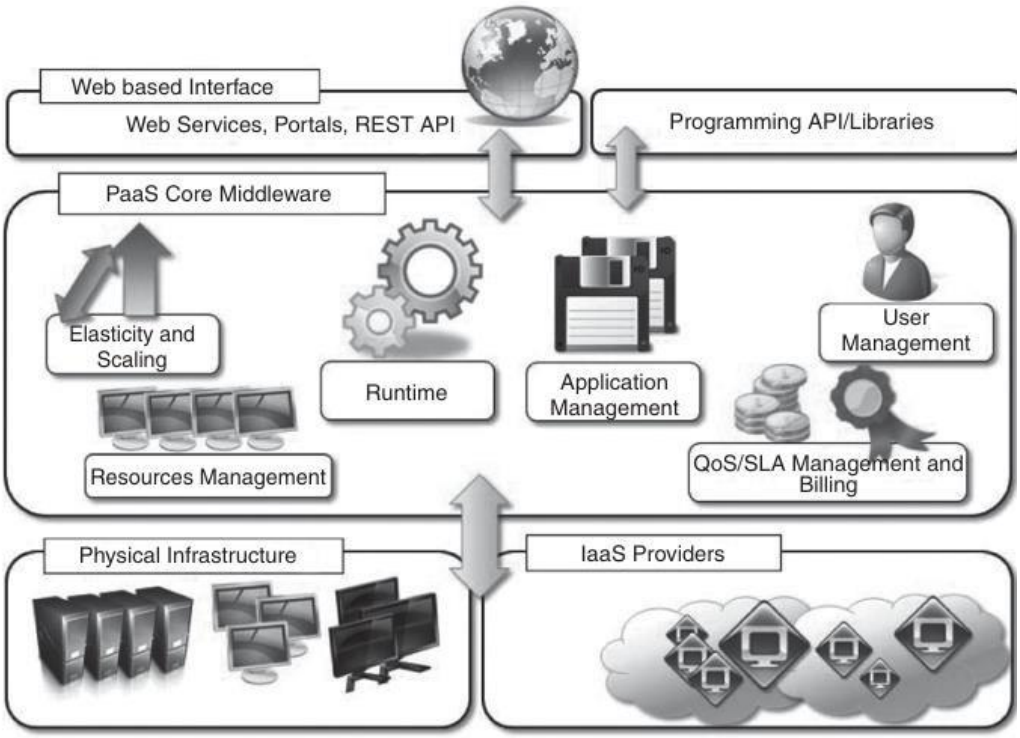


Figure: Platform as a service (PaaS)

The PaaS architecture is divided into the following layers:

1. Physical Infrastructure

- This is the base layer consisting of servers, storage systems, and network resources.
- These physical resources are usually provided and managed by an IaaS provider.
- PaaS services run on top of this infrastructure to provide computing power for applications.

2. PaaS Core Middleware

- This is the main layer that enables all functionalities of the PaaS environment.
- It includes a runtime environment where applications are executed.
- The resource management component allocates CPU, memory, and storage to applications.
- Elasticity and scaling automatically adjust resources to handle changing traffic loads.

- Application management manages deployment, updates, and performance of applications.
- QoS/SLA management and billing ensure service quality and calculate usage cost.

3. Interface Layer

- This layer provides ways for users and developers to interact with the platform.
- The web-based interface allows users to deploy and monitor applications using dashboards or portals.
- Programming APIs and libraries enable developers to write applications using supported languages like Python, Java, .NET, Node.js, etc.

4. Applications and Users

- Applications built on PaaS run using the runtime services provided by the platform.
- Users access these applications through mobile or web interfaces.
- Developers benefit from simplified coding, testing, and deployment.

Advantages

For Developers

- Developers don't need to manage hardware or operating systems.
- Faster development due to built-in tools, libraries, and frameworks.
- Automatic scaling and monitoring help maintain performance.
- Makes team collaboration easier.
- Suitable for rapid prototyping and innovation.

For Organizations

- Reduces cost of setting up and maintaining infrastructure.
- Speeds up time-to-market for applications.
- Simplifies deployment and updates.
- Integrates easily with databases, APIs, and cloud services.

Examples of PaaS

- Google App Engine
- Heroku

- Microsoft Azure App Service
- AWS Elastic Beanstalk

Software as a service (SaaS)

Definition

- Software as a Service (SaaS) is a cloud model where software applications are delivered to users over the internet.
- Users do not install anything on their computer; they simply log in through a web browser to use the application.
- All hardware, servers, databases, updates, and maintenance are handled completely by the service provider.
- Users access the software on a subscription or pay-as-you-go basis.

Main Concept

- SaaS applications are centrally hosted in the provider's data center.
- Multiple users share the same application through a multi-tenant architecture.
- Providers manage everything including updates, security, scaling, and backups.
- Customers only need internet access and login credentials to use the application.
- The same software is provided to many users but can still be lightly customized

Features

- *Web-based access*: Users can access software through a browser from any device.
- *No installation:* No setup or local installation is required on user machines.
- *Automatic updates*: The provider updates the software automatically without disturbing users.
- *Subscription billing*: Users pay monthly or yearly based on usage.
- *Multi-tenancy*: One application serves many users while keeping data separate.
- *Centralized management*: Security, backups, patches, and upgrades are fully handled by provider.

Architecture of SaaS

The SaaS architecture consists of:

1. Application Layer

 This is the actual software users see (email)
It contains UI, business logic, and user workflows.

2. Multi-Tenant Layer

- One software instance serves many customers.
- Data is isolated per customer, but application code is shared.
- This reduces cost and improves efficiency.

3. Data Management Layer

- Stores all customer data securely in databases.
- Handles backups, encryption, and access control.

4. Infrastructure Layer

- Runs on top of virtualized servers, storage, and networks (IaaS).
- Provider manages scaling, performance, and availability.

Advantages

For Users:

- No need for installation or maintenance of software.
- Very low upfront cost because no hardware or licenses are required.
- Accessible from anywhere using a browser or mobile app.
- Automatic maintenance and upgrades save time.
- Easy to collaborate because everyone sees the same updated version.

For Providers:

- Easier to manage and update the application for all users at once.
- Lower operating cost due to shared resources.
- Allows quick feature delivery and system improvement

Examples

- Google Workspace (Docs, Sheets, Gmail)
- Microsoft Office 365
- Salesforce CRM

IMPLEMENTATION

Implementation of IaaS Using AWS EC2

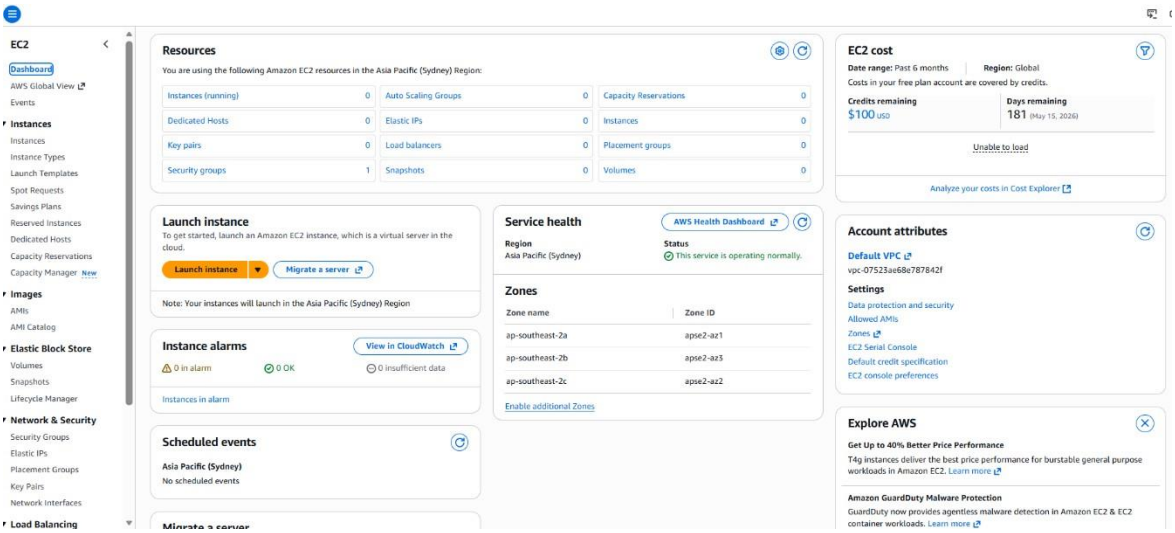
To demonstrate how Infrastructure as a Service (IaaS) works, I created a virtual machine on Amazon Web Services (AWS) using the EC2 service. The steps I followed are explained below in a clear and simple way.

Step 1: Logging in to AWS

- I started by logging in to the AWS Management Console using my account.
- This is needed to access all cloud services'

Step 2: Opening the EC2 Dashboard

- From the AWS services menu, I selected EC2.
- EC2 is the service that provides virtual machines, which is the main part of IaaS.

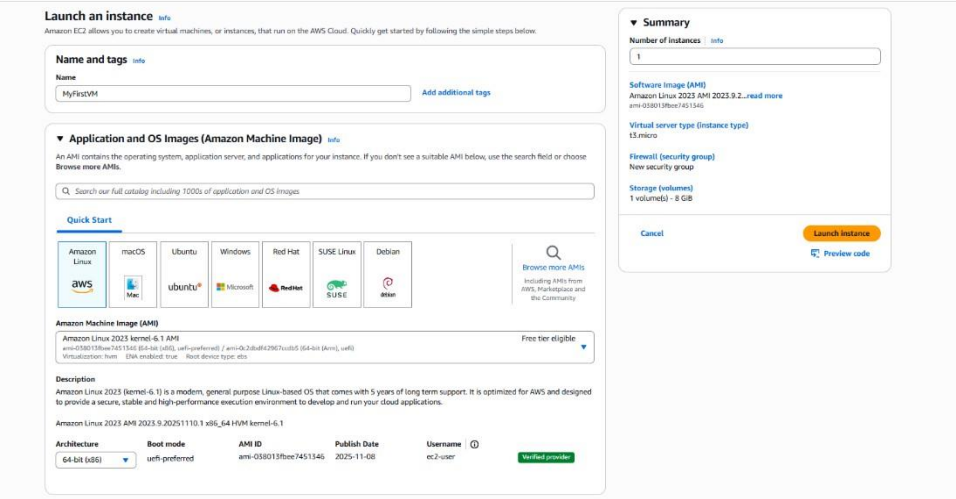


Step 3: Starting the VM Creation

- I clicked on "Launch Instance" to begin creating my virtual machine.
  This button starts the process of setting up a cloud-based server.

Step 4: Naming the VM & Selecting an Operating System

- I gave my instance the name MidFirstVM.
- Then I chose the Amazon Linux 2023 AMI, which is free-tier eligible. The AMI decides the operating system that will run on the VM.

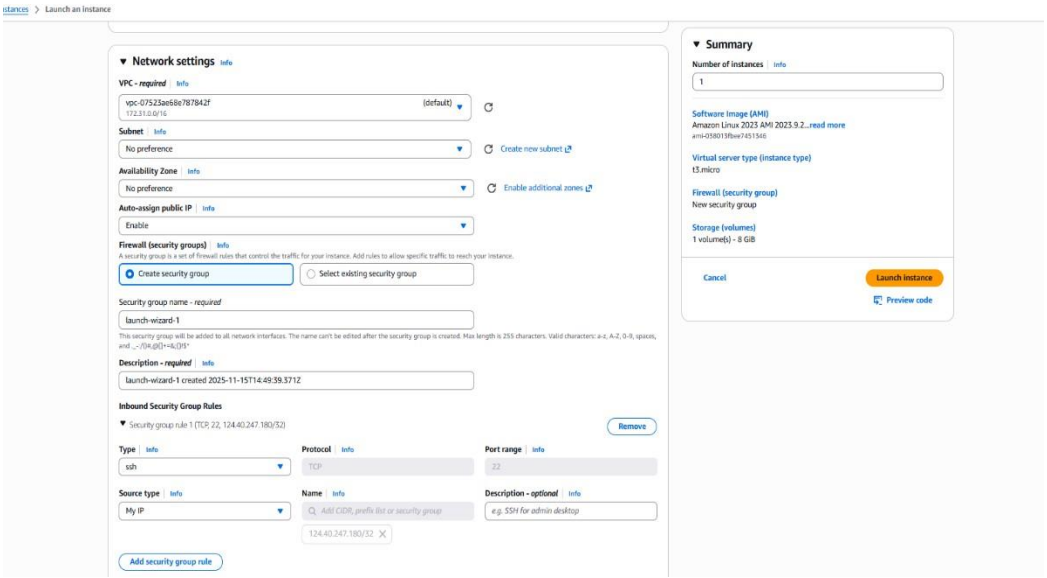


Step 5: Choosing the Instance Type

- I selected the t3. micro instance type.
- This is a lightweight VM and is included in the free tier, which makes it suitable for basic testing.

Step 6: Configuring Network Settings

- I used the default VPC and kept the subnet on “No preference.”
- Then I enabled Auto-assign Public IP so the VM can be accessed online.
  Next, I created a new Security Group and allowed SSH (port 22).
  The security group works like a firewall and controls which traffic is allowed.



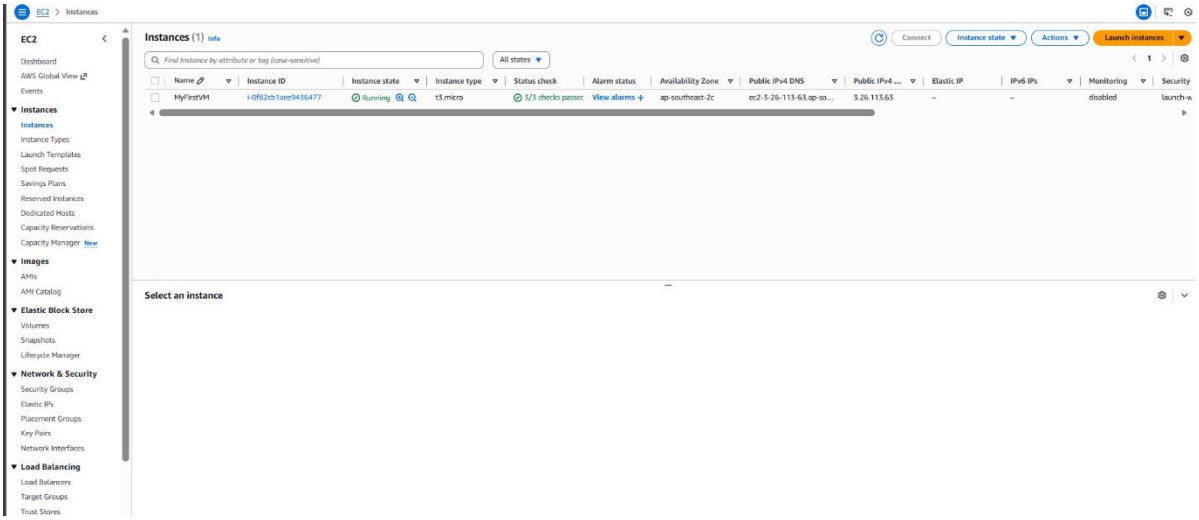
Step 7: Launching the Instance

I clicked Launch Instance to complete the setup.

At this stage, AWS allocates CPU, memory, networking, and storage which is exactly what makes it IaaS.

Step 8: Checking the Instance Status

After a few seconds, the instance appeared on the EC2 dashboard with the status Running. This means the virtual machine is fully created and active.

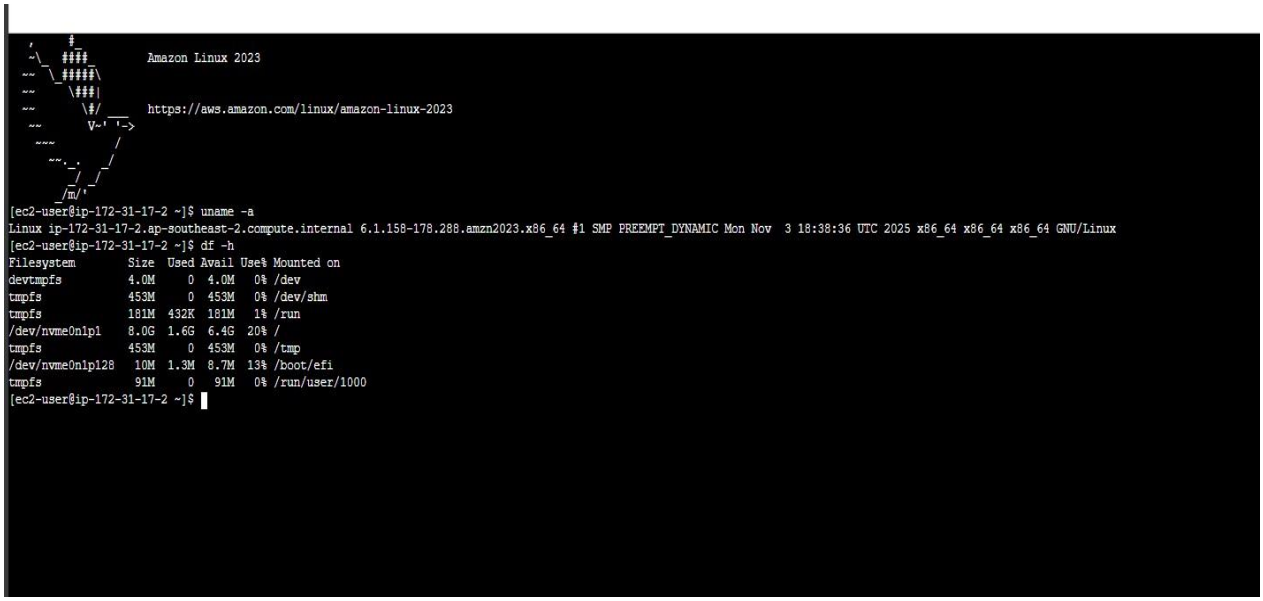


Step 9: Connecting to the VM

- I used EC2 Instance Connect to open a Linux terminal directly in my browser.
- This lets me control the VM just like a real computer using SSH.

Step 10: Testing the VM

- Inside the terminal, I ran simple commands such as:
- uname -a: It tells which Linux version and kernel is running on my EC2 instance
- df -h: checks storage space allocated to my EC2 instance
- This shows that the VM is working properly and ready to use



Conclusion

In this task, I created my own virtual machine using AWS EC2, and it clearly showed why IaaS is so useful. I was able to choose everything myself the operating system, instance size, storage, and network settings just like building my own computer in the cloud. IaaS makes it easy to customize and scale resources whenever needed, without buying or managing physical hardware. It is flexible, fast, and ideal for learning as well as running real-world applications.

Implementation of SaaS using Google Docs

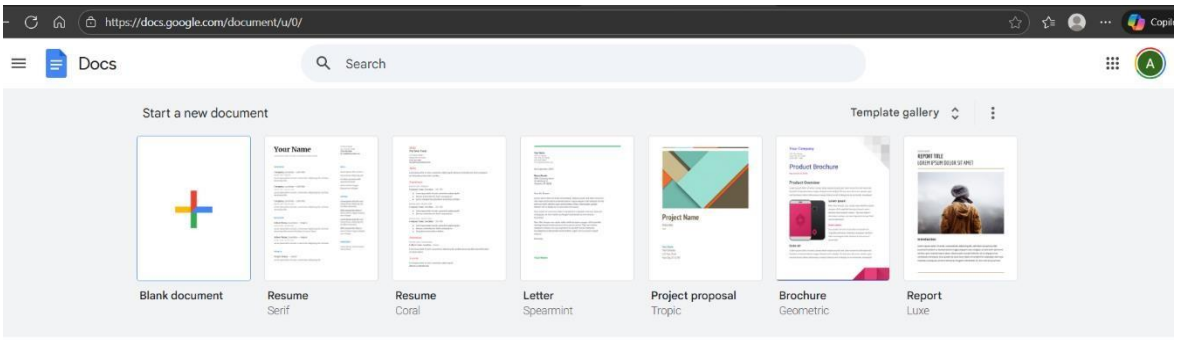
Google Docs is an online document editor that works directly in the browser. You don't need to install anything. Everything is stored automatically in Google Drive, making it a simple example of Software as a Service (SaaS).

Step 1: Open Google Docs

Go to docs.google.com in your web browser.

Step 2: Sign in with Your Google Account

Enter your Gmail ID and login



Step 3: Create a New Document

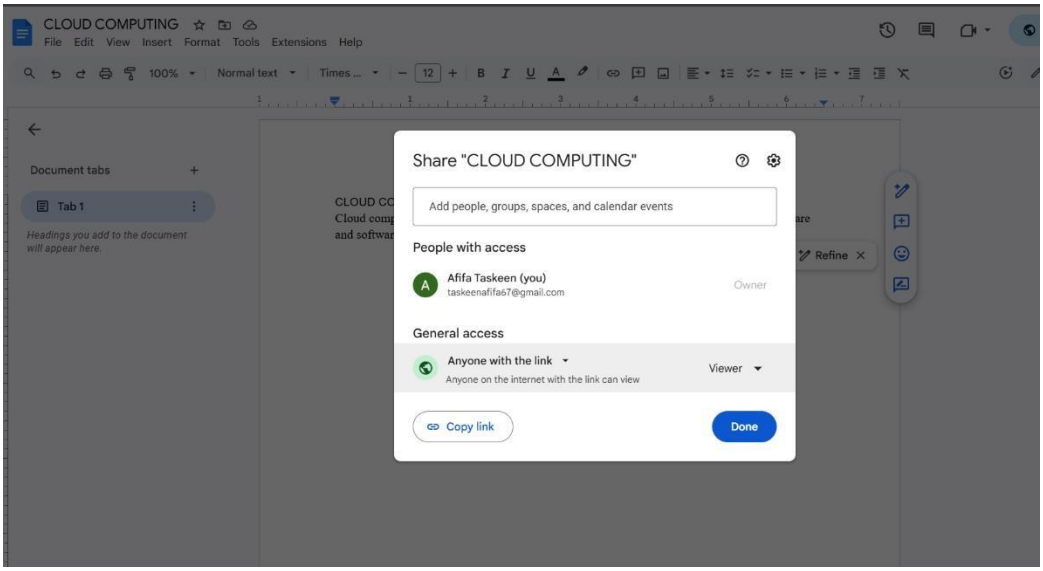
Click on Blank Document to start writing.

Step 4: Type and Edit Your Content

- Start typing your text in the Google Docs editor.
- All changes are saved automatically in the cloud.

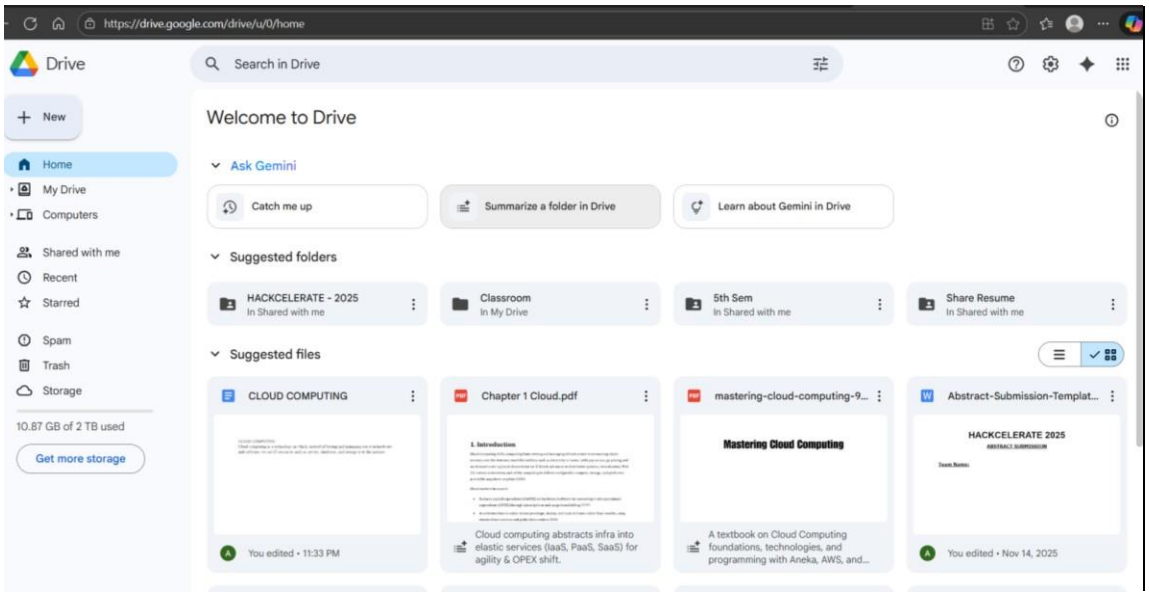
Step 5: Share the Document

- Click the Share button at the top-right corner.
- Choose "Anyone with the link can view/edit" or add an email ID.



Step 6: Access the Document from Any Device

You can open the same document on laptop, phone, or tablet anytime.



Conclusion

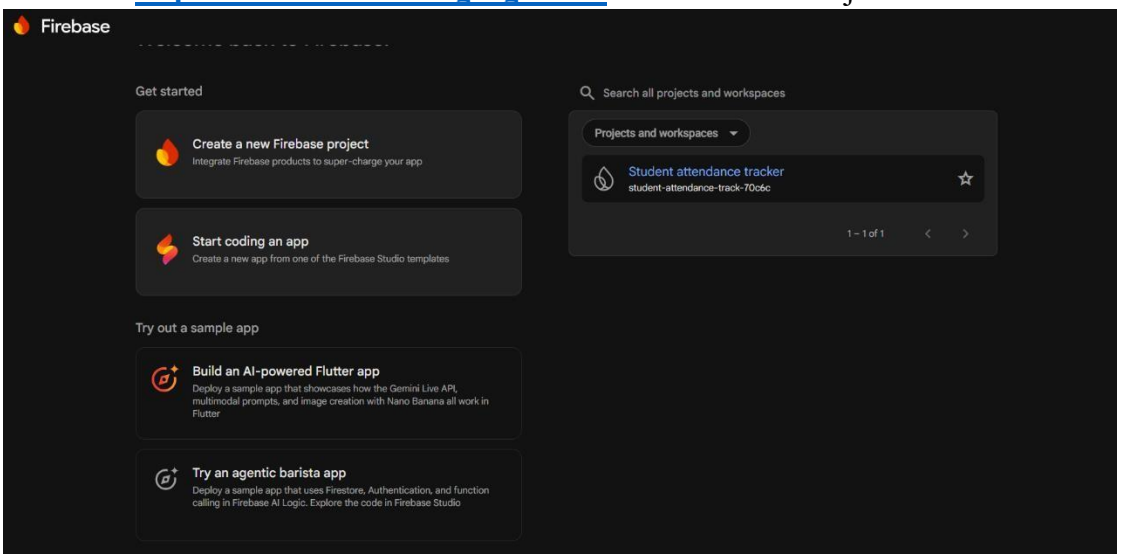
Using Google Docs shows how SaaS works. The software is ready to use, updates automatically, and saves everything online without any setup. It makes working with documents easy and accessible from anywhere.

Implementation of PaaS Using Firebase

Firebase is a cloud-based development platform provided by Google. It offers readymade backend services such as databases, authentication, hosting, and storage. Developers can directly use these services without managing any servers or hardware. This makes Firebase a perfect example of Platform as a Service (PaaS), because it provides the platform and tools needed to build applications easily.

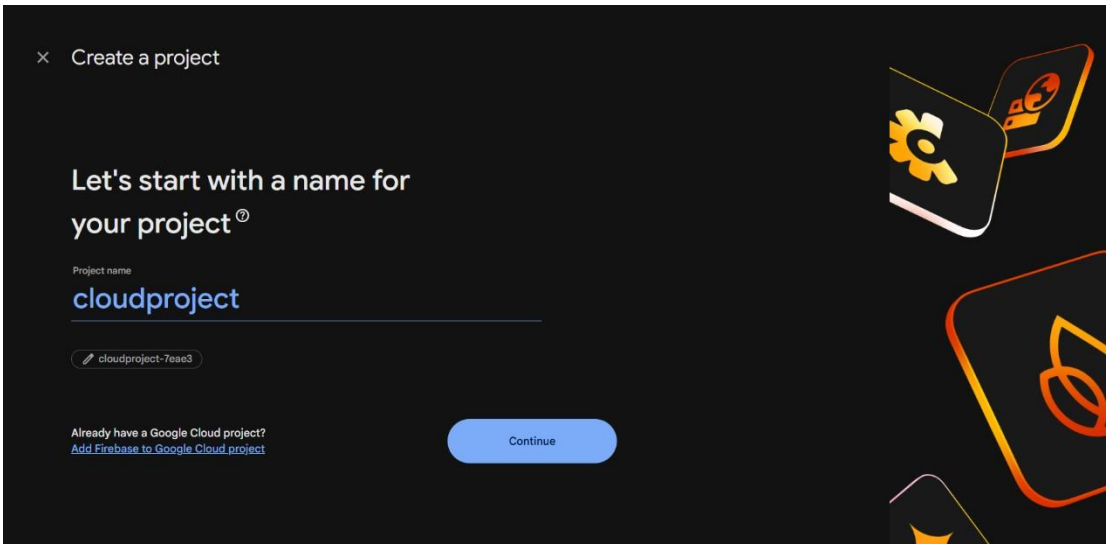
Step 1: Open Firebase Console

Go to **https://console.firebase.google.com** and click Add Project.



Step 2: Create a New Firebase Project

Enter a project name



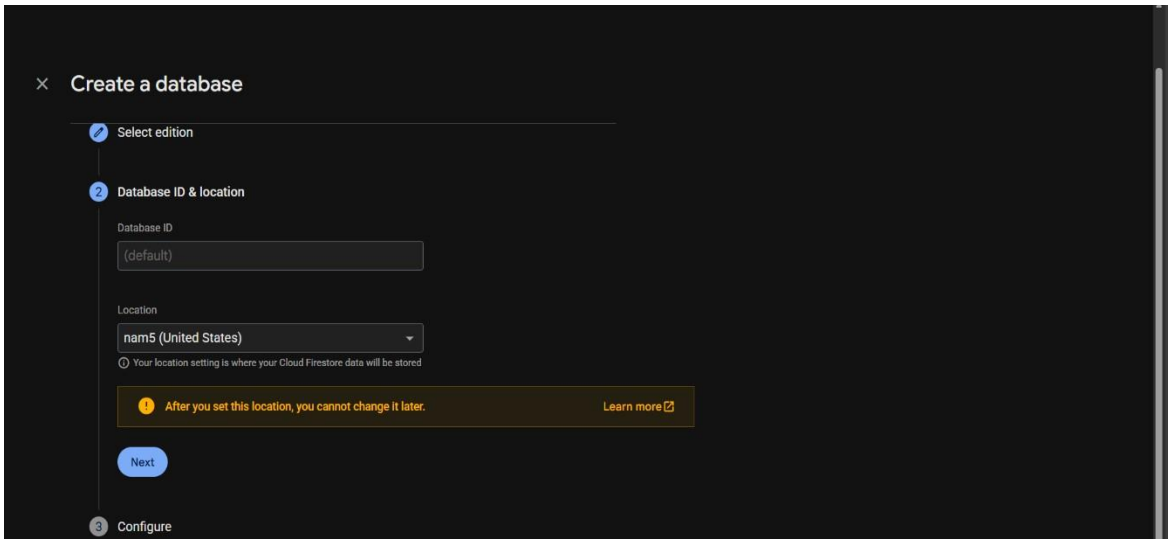
Step 3: Wait for Project Setup

Firebase automatically creates the project and required backend services

Step 4: Open Firebase Database

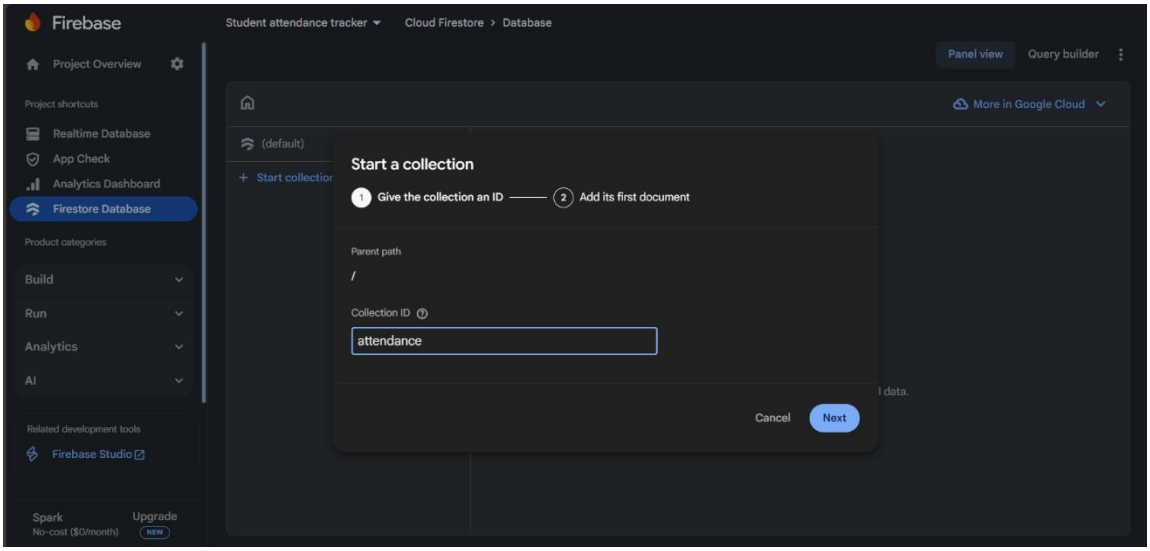
Step 5: Choose Database Mode

Selected test mode for easy demonstration



Step 6: Create a New Collection

Click on start collection and naming the collection id as attendance

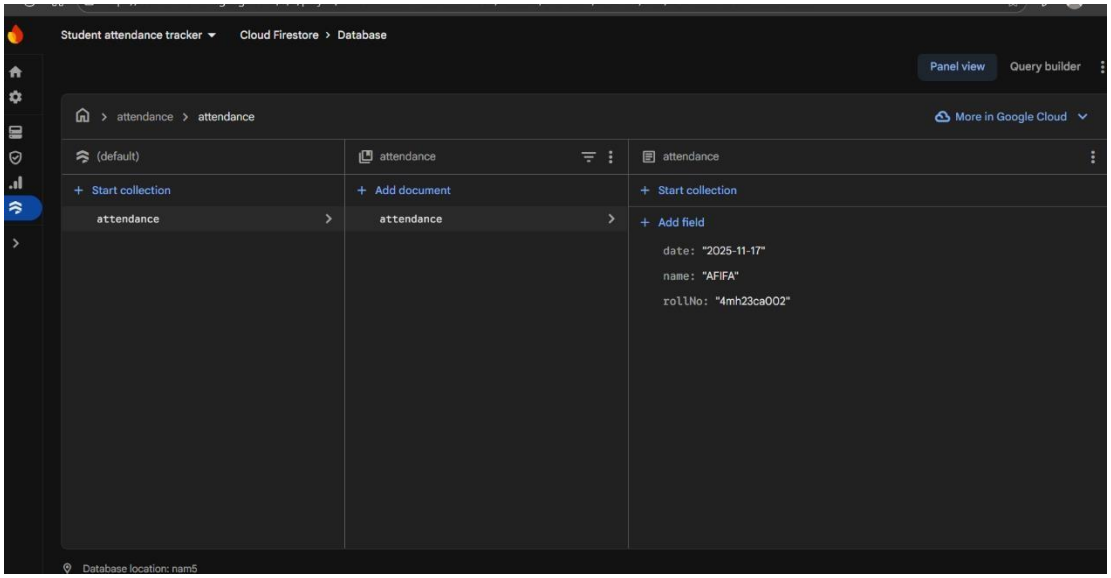


Step 8: Add an Attendance Record

Enter fields such as:

Name, roll No, date, status

Click on Save the record appears



Conclusion

Firebase made it easy to store attendance data without setting up any servers. It gets updated and we can retrieve the data whenever required. The data is stored safely. I only created a project and added data, and the platform handled everything else. This shows how PaaS helps us build apps quickly and easily.

CONCLUSION

This assignment really helped me understand cloud computing in a clear and practical way. By trying out IaaS, PaaS, and SaaS myself, I could see how each model works and why they are useful. With IaaS, I learned how to control and set up my own virtual machine. With PaaS, I saw how easy it is to build application features without worrying about servers. And with SaaS, I understood how simple it is to use software directly through the internet.

Overall, I realised that cloud computing makes technology more flexible, affordable, and easier to work with for everyone whether developers or regular users.

References

1. Introduction to Cloud Computing offered by IBM on Coursera
2. Amazon Web Services Documentation – https://aws.amazon.com
3. Google Firebase Documentation – https://firebase.google.com/docs
4. Google Docs Help Center – https://support.google.com/docs
5. Rajkumar Buyya, “Mastering Cloud Computing,” McGraw Hill Education.

BIBLIOGRAPHY

1. Buya, Rajkumar, Christian Vecchiola, and S. Thamarai Selvi. *Mastering Cloud Computing*. McGraw Hill Education, 2013.
2. National Institute of Standards and Technology (NIST). "The NIST Definition of Cloud Computing." https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-145.pdf
3. Amazon Web Services (AWS). *Amazon EC2 Documentation*. https://docs.aws.amazon.com/ec2/
4. Google Firebase. *Firebase Database Documentation*. https://firebase.google.com/docs/firestore
5. Google Workspace. *Google Docs Help Center*. https://support.google.com/docs
6. IBM Cloud. *Cloud Computing Service Models (IaaS, PaaS, SaaS)* – YouTube Video.
7. Microsoft Azure Documentation. *Cloud Service Models Overview*. https://learn.microsoft.com/azure/
8. Tutorials Point. *Cloud Computing Basics*. https://www.tutorialspoint.com/cloud_computing//