CLOUD COMPUTING AND DEVOPS

Bridging Infrastructure, Automation, and Innovation

A Comprehensive Investigation of Virtualization, Microservices, and Cloud-Based DevOps Ecosystems

Cloud Computing and DevOps

Bridging Infrastructure, Automation, and Innovation

A Comprehensive Investigation of Virtualization, Microservices, and Cloud-Based DevOps Ecosystems

ISBN No: 978-81-998007-4-8

2026 Edition

Dr. Kamal Shah
Principal
St. John College of Engineering and Management, Palghar, Mumbai Maharashtra

Ms. Aishwarya Churi
Assistant Professor
Department of Electronics and Computer science Engineering
St. John College of Engineering and Management, Palghar, Mumbai Maharashtra

Mrs Karishma JamirTamboli
Lecturer
Department of Computer Engineering
St. John College of Polytechnic ,Palghar, Mumbai Maharashtra

Mr Pranjal Save
Technical Expert Department of Computer Engineering
St. John College of Polytechnic ,Palghar, Mumbai Maharashtra

Published By:
Imperial Publications
304 De Elmas Sonawala Cross Road 2,
Goregaon E Mumbai 400063, India
www.imperialpublications.com

ABSTRACT

This book presents a comprehensive and practice-oriented exploration of modern cloud computing, resilient microservices architecture, advanced cloud security practices, data engineering frameworks, and financial governance in cloud environments. Designed to address the rapidly evolving landscape of distributed systems and enterprise cloud adoption, the volume integrates theoretical foundations with hands-on implementation strategies relevant to contemporary industry practices.

Beginning with fundamental concepts, the book establishes a strong foundation in cloud computing principles, including deployment models (Public, Private, Hybrid, and Community Cloud) and service models (SaaS, PaaS, and IaaS). It critically compares cloud computing with other computing paradigms such as grid, distributed, peer-to-peer, and client–server architectures, enabling readers to understand the technological evolution and architectural trade-offs that shaped modern cloud systems.

The second chapter transitions into resilient microservices architecture tailored for Java developers, reflecting the industry's shift from monolithic systems to distributed cloud-native applications. It explores architectural transformation strategies including the Strangler Fig Pattern and Domain- Driven Design, performance engineering in distributed systems, and essential resilience patterns such as Circuit Breaker, Bulkhead, Retry, and Graceful Degradation. The discussion extends to Kubernetes- based container orchestration, automated scaling mechanisms like Horizontal Pod Autoscaling, and integration with managed platforms such as Amazon Elastic Kubernetes Service, providing readers with operational clarity on building fault-tolerant systems.

Advanced cloud security practices form a critical pillar of the book. Moving beyond traditional identity and access management, the text examines Zero Trust Architecture, Attribute-Based Access Control, Kubernetes network security, mutual TLS, automated threat detection systems, and cloud- native incident response frameworks. Real-world tools including AWS GuardDuty, AWS Security Hub, and AWS Lambda are analyzed to demonstrate automated remediation and security orchestration in modern environments.

Recognizing the growing importance of data-driven decision-making, the book dedicates significant attention to data engineering and intelligence. It explores Data Fabric architecture,

modern data lake implementation, cloud-based data pipelines using Amazon S3 and AWS Glue, and real-time business intelligence systems. Practical workflows demonstrate how organizations can transform raw cloud data into actionable insights using executive dashboards and live analytics platforms.

The final section introduces FinOps and cloud governance, emphasizing cost optimization, sustainability, and responsible cloud consumption. Topics include cloud carbon footprint analysis, green cloud computing practices, automated budgeting, anomaly detection, resource rightsizing, and strategic cost control mechanisms that align technical decisions with financial accountability.

By integrating architecture, resilience engineering, cybersecurity, data intelligence, and financial governance into a unified framework, this book equips students, developers, architects, and decision- makers with the knowledge required to design scalable, secure, sustainable, and cost-efficient cloud ecosystems. The content balances conceptual clarity with practical applicability, making it suitable for academic study, professional certification preparation, and industry implementation.

Keywords: Cloud Computing, Microservices Architecture, Kubernetes, Zero Trust Security, Data Engineering, Data Lake, FinOps, Cloud Governance, Business Intelligence, Fault Tolerance, Container Orchestration, Scalability, Sustainability in Cloud.

PREFACE

The evolution of computing from centralized data centers to globally distributed cloud-native ecosystems represents one of the most transformative shifts in modern technology. Organizations today no longer merely host applications; they design scalable, resilient, intelligent, and financially optimized digital ecosystems that must operate continuously in an unpredictable and security- sensitive environment. This book was conceived in response to that transformation.

Over the past decade, cloud computing has moved from being a cost-saving alternative to traditional infrastructure into becoming the foundational backbone of digital innovation. Enterprises now depend on cloud platforms for artificial intelligence, large-scale analytics, microservices-based applications, global collaboration, and real-time customer engagement. However, with this rapid adoption has emerged a complex set of challenges: architectural fragmentation, distributed system failures, cybersecurity threats, cost overruns, and sustainability concerns. Addressing these challenges requires an integrated understanding that spans technology, security, performance engineering, data intelligence, and financial governance.

This book has been structured to reflect that integrated perspective. It begins with fundamental cloud computing concepts to ensure conceptual clarity for readers entering the domain. It then progressively advances toward resilient microservices engineering, acknowledging the shift from monolithic architectures to cloud-native distributed systems. The emphasis on Java-based microservices recognizes the continued relevance of Java in enterprise ecosystems while also preparing developers to leverage containerization and Kubernetes orchestration frameworks.

Security is treated not as an afterthought but as an architectural principle embedded across all layers of cloud design. The discussion of Zero Trust Architecture, dynamic access control models, automated threat detection, and digital forensics reflects the reality that modern systems must assume breach and design for containment and recovery.

Equally important is the growing role of data engineering in cloud environments. Organizations generate massive volumes of structured and unstructured data, and the ability to architect data lakes, implement Data Fabric strategies, and build real-time dashboards is now central to strategic decision- making. By including end-to-end data pipeline demonstrations, this book bridges

theoretical architecture with practical implementation workflows.

The inclusion of FinOps and cloud governance underscores another critical dimension: sustainability and financial accountability. Cloud resources, if unmanaged, can become financially inefficient and environmentally unsustainable. Integrating cost monitoring, resource optimization, and green computing practices ensures that technological progress aligns with economic responsibility and environmental consciousness.

This book is intended for multiple audiences. Undergraduate and postgraduate students will find it suitable as a comprehensive academic text for cloud computing, distributed systems, and data engineering courses. Software developers and architects will benefit from practical design patterns, migration strategies, and resilience engineering techniques. Cloud administrators and DevOps professionals will find actionable guidance on Kubernetes management, scaling, monitoring, and cost optimization. Finally, decision-makers and technology leaders will gain insight into governance frameworks and sustainable cloud strategies.

While every effort has been made to maintain technical accuracy and contemporary relevance, cloud technologies evolve rapidly. Readers are encouraged to treat this book not as a static manual but as a conceptual framework that enables them to adapt to emerging tools, services, and paradigms.

The ultimate goal of this work is to empower readers to build cloud systems that are not only scalable and performant, but also secure, intelligent, resilient, and responsible. In an era where digital infrastructure underpins nearly every aspect of society, designing such systems is not merely a technical challenge—it is a professional responsibility.

-------INDEXING---------

Chapter 1 – Introduction to Cloud

1.1 Introduction to Cloud Computing

- **1.1.1** What is Cloud Computing
  - **1.1.2** Why Cloud Computing
  - **1.1.3** How does cloud computing work
  - **1.1.4** Characteristics of cloud computing
  - **1.1.5** Advantages and Disadvantages of cloud computing

1.2 Cloud Computing Architecture

  - **1.2.1** Introduction to Cloud Architecture
- **1.2.2** Front End and Back End
  - **1.2.3** Components of Cloud Architecture

1.3 Cloud Computing and Other Computing Models

  - **1.3.1** Cloud vs Grid Computing
  - **1.3.2** Cloud vs Client-Server Computing
  - **1.3.3** Cloud vs Peer-to-Peer Architecture
  - **1.3.4** Cloud vs Distributed Computing

1.4 Types of Cloud

  - **1.4.1** Public Cloud
  - **1.4.2** Private Cloud
  - **1.4.3** Hybrid Cloud
  - **1.4.4** Community Cloud
  - **1.4.5** Comparison of Cloud Types

1.5 Service Models of Cloud Computing

  - **1.5.1** Software as a Service (SaaS)
  - **1.5.2** Platform as a Service (PaaS)
  - **1.5.3** Infrastructure as a Service (IaaS)
  - **1.5.4** Difference between IaaS, PaaS and SaaS

Chapter 2 – Resilient Microservices for Java Developers

2.1 Introduction: The Need for Resilience

- **2.1.1** Evolution of Modern Software Systems
- **2.1.2** Changing User Expectations in Digital Platforms
- **2.1.3** Failures in Distributed Cloud Environments
- **2.1.4** Definition and Importance of Resilience
- **2.1.5** Role of Java in Building Resilient Systems
- **2.1.6** Microservices as a Foundation for Resilience

2.2 Monolithic Architecture and Its Limitations

- **2.2.1** Structure of Monolithic Architecture
- **2.2.2** Advantages of Monolithic Systems
- **2.2.3** Scalability Challenges in Monoliths
- **2.2.4** Deployment and Maintenance Difficulties
- **2.2.5** Fault Isolation Issues

2.3 Emergence of Microservices Architecture

- **2.3.1** Concept and Definition of Microservices
- **2.3.2** API Gateway as an Entry Point
- **2.3.3** Independent Service Deployment
- **2.3.4** Data Autonomy and Decentralization

2.4 Migrating Legacy Java Applications

- **2.4.1** Challenges in Legacy System Migration
- **2.4.2** Strangler Fig Pattern
- **2.4.3** Domain-Driven Design (DDD) Concepts
- **2.4.4** Bounded Context and Service Identification

2.5 Performance Engineering

- **2.5.1** Network Latency in Distributed Systems
- **2.5.2** Distributed Monolith Problem
- **2.5.3** Performance Optimization Techniques
- **2.5.4** Monitoring and Observability

2.6 Resilience Patterns and Fault Tolerance

- **2.6.1** Concept of Fault Tolerance
- **2.6.2** Circuit Breaker Pattern
- **2.6.3** Bulkhead Pattern
- **2.6.4** Retry and Timeout Mechanisms
- **2.6.5** Graceful Degradation

2.7 Zero Trust Security

- **2.7.1** Principles of Zero Trust Architecture
- **2.7.2** Role-Based Access Control (RBAC)
- **2.7.3** Network Policies in Kubernetes
- **2.7.4** Mutual TLS (mTLS) Encryption

2.8 Kubernetes and Amazon Elastic Kubernetes Service (EKS)

- **2.8.1** Kubernetes Architecture Overview
- **2.8.2** Control Plane and Worker Nodes
- **2.8.3** Self-Healing and Auto-Recovery Mechanisms
- **2.8.4** Introduction to Amazon EKS

Chapter 3 – Advanced Cloud Security Practices

3.1 Evolving Beyond IAM

- **3.1.1** Challenges of Role-Based Access Control (RBAC)
- **3.1.2** Introduction to Attribute-Based Access Control (ABAC)
- **3.1.3** Leveraging Dynamic Tags in ABAC

3.2 Mastering the Incident Response Lifecycle

- **3.2.1** Phases of Incident Response
- **3.2.2** Hands-On Lab: Credential Leak Simulation
- **3.2.3** Applying Digital Forensics in the Cloud

3.3 Automating Threat Detection & Response

- **3.3.1** Threat Intelligence with AWS GuardDuty
- **3.3.2** Centralized Security Management via AWS Security Hub

- **3.3.3** Automated Remediation Workflows using AWS Lambda

Chapter 4 – Cloud Data Engineering and Intelligence

4.1 Introduction to Data Engineering

- **4.1.1** Importance of Data Engineering in Modern Systems
- **4.1.2** Challenges in Handling Large-Scale Data
- **4.1.3** Cloud-Based Data Processing Approaches

4.2 Data Fabric Architecture

- **4.2.1** Concept of Data Fabric
- **4.2.2** Components of Data Fabric Architecture
- **4.2.3** Data Integration and Data Virtualization

4.3 Modern Data Lake Architecture

- **4.3.1** Introduction to Data Lakes
- **4.3.2** Difference Between Data Warehouse and Data Lake
- **4.3.3** Managing Structured and Unstructured Data

4.4 Building Cloud-Based Data Pipelines

- **4.4.1** Concept of Data Pipelines
- **4.4.2** Data Ingestion and Transformation
- **4.4.3** Pipeline Automation and Scheduling

4.5 Real-Time Data Processing

- **4.5.1** Importance of Real-Time Data Analytics
- **4.5.2** Stream Processing and Event-Driven Architecture

Chapter 5 – FinOps and Cloud Governance

5.1 Introduction to Cloud Financial Management

- **5.1.1** Concept and Principles of FinOps
- **5.1.2** Cloud Cost Management Challenges

5.2 Cloud Cost Optimization Strategies

- **5.2.1** Resource Usage Monitoring and Rightsizing

- **5.2.2** Reserved Instances and Savings Plans
- **5.2.3** Budget Monitoring and Alerts

5.3 Cloud Governance Framework

- **5.3.1** Governance Policies and Compliance
- **5.3.2** Risk Management in Cloud Environments

5.4 Sustainable Cloud Computing

- **5.4.1** Green Cloud Computing Concept
- **5.4.2** Cloud Carbon Footprint and Energy Efficiency

5.5 Future Trends in Cloud Computing

- **5.5.1** AI-Driven Cloud Management
- **5.5.2** Serverless Computing Evolution
- **5.5.3** Edge Computing Integration

Chapter 6 – Case Study

- **6.1** AWS
- **6.2** Azure
- **6.3** Google Cloud
- **6.4** Alibaba

----------LIST OF FIGURES----------

Figure No.	Figure Name	Page No.
Figure 2.1	Structure of a Monolithic Architecture	27
Figure 2.2	Microservices Architecture Model	29
Figure 2.3	Strangler Fig Migration Strategy	31
Figure 2.4	Circuit Breaker State Transition Model	34
Figure 2.5	Zero Trust Implementation in Kubernetes	36
Figure 2.6	Kubernetes Cluster Architecture	37
Figure 2.7	Amazon EKS Architecture Overview	38
Figure 2.8	Horizontal Pod Autoscaler Workflow	40

----------LIST OF TABLES ----------

Table No.	Table Name	Page No.
Table 1.1	Comparison of Cloud Types	16
Table 1.2	Difference between IaaS, PaaS, and SaaS	25
Table 3.1	RBAC vs. ABAC Comparison	44
Table 4.1	Data Warehouse vs. Data Lake	65
Table 5.1	Cloud Cost Management Challenges	87

Chapter 1

INTRODUCTION TO CLOUD COMPUTING

1.1 Introduction to Cloud Computing

1.1.1 What is Cloud Computing

The term cloud refers to a network or the internet. It is a technology that uses remote servers on the internet to store, manage, and access data online rather than local drives. The data can be anything such as files, images, documents, audio, video, and more.

There are the following operations that we can do using cloud computing:

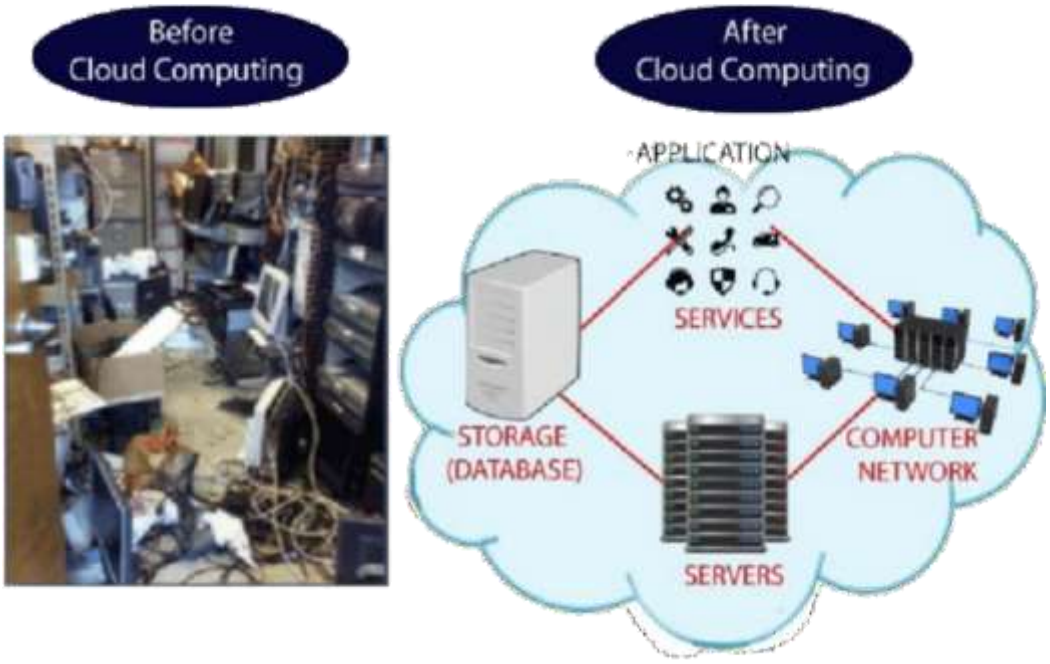
- Developing new applications and services
- Storage, back up, and recovery of data
- Hosting blogs and websites
- Delivery of software on demand
- Analysis of data
- Streaming videos and audios

1.1.2 Why Cloud Computing?

Small as well as large IT companies, follow the traditional methods to provide the IT infrastructure. That means **for any IT company, we need a Server Room that is the basic need of IT companies**.

In that server room, there should be a database server, mail server, networking, firewalls, routers, modem, switches, QPS (Query Per Second means how much queries or load will be handled by the server), configurable system, high net speed, and the maintenance engineers.

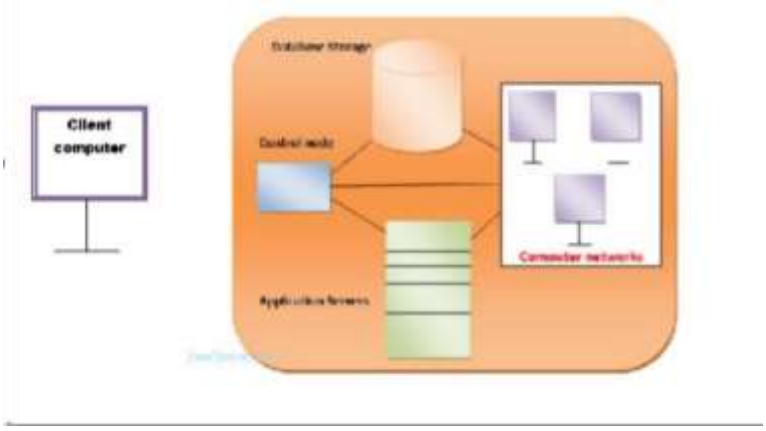
To establish such IT infrastructure, we need to spend lots of money. To overcome all these problems and to reduce the IT infrastructure cost, Cloud Computing comes into existence.



1.1.3 How does cloud computing work

Assume that you are an executive at a very big corporation. Your particular responsibilities include to make sure that all of your employees have the right hardware and software they need to do their jobs. To buy computers for everyone is not enough. You also have to purchase software as well as software licenses and then provide these softwares to your employees as they require. Whenever you hire a new employee, you need to buy more software or make sure your current software license allows another user. It is so stressful that you have to spend lots of money.

But, there may be an alternative for executives like you. So, instead of installing a suite of software for each computer, you just need to load one application. That application will allow the employees to log-in into a Web-based service which hosts all the programs for the user that is required for his/her job. Remote servers owned by another company and that will run everything from e-mail to word processing to complex data analysis programs. It is called cloud computing, and it could change the entire computer industry.



In a cloud computing system, there is a significant workload shift. Local computers have no longer to do all the heavy lifting when it comes to run applications. But cloud computing can handle that much heavy load easily and automatically. Hardware and software demands on the user's side decrease. The only thing the user's computer requires to be able to run is the cloud computing interface software of the system, which can be as simple as a Web browser and the cloud's network takes care of the rest.

1.1.4 Characteristics of Cloud Computing

The characteristics of cloud computing are given below:

1) Agility

The cloud **works in a distributed computing environment**. It shares resources among users and works very fast.

2) High availability and reliability

The availability of servers is high and more reliable because the **chances of infrastructure failure are minimum**.

3) High Scalability

Cloud offers **"on-demand" provisioning of resources on a large scale**, without having engineers for peak loads.

4) Multi-Sharing

With the help of cloud computing, **multiple users and applications can work more efficiently** with cost reductions by sharing common infrastructure.

5) Device and Location Independence

Cloud computing enables the users to access systems using a web browser regardless of their location or what device they use e.g. PC, mobile phone, etc. **As infrastructure is off-site** (typically provided by a third-party) **and accessed via the Internet, users can connect from anywhere**.

6) Maintenance

Maintenance of cloud computing applications is easier, since they **do not need to be installed on each user's computer and can be accessed from different places**. So, it reduces the cost also.

7) Low Cost

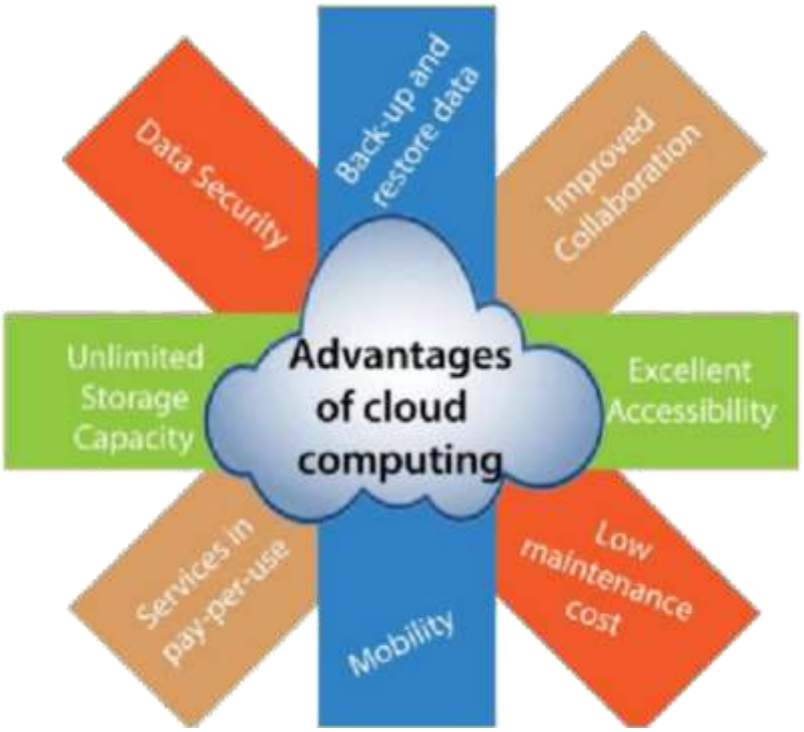
By using cloud computing, the cost will be reduced because to take the services of cloud computing, **IT company need not to set its own infrastructure** and pay-as- per usage of resources.

8) Services in the pay-per-use mode

Application Programming Interfaces **(APIs) are provided to the users so that they can access services on the cloud** by using these APIs **and pay the charges as per the usage of services**.

1.1.5 Advantages and Disadvantages of Cloud Computing

- Advantages of Cloud Computing



1) Back-up and restore data

Once the data is stored in the cloud, it is easier to get back-up and restore that data using the cloud.

2) Improved collaboration

Cloud applications improve collaboration by allowing groups of people to quickly and easily share information in the cloud via shared storage.

3) Excellent accessibility

Cloud allows us to quickly and easily access store information anywhere, anytime in the whole world, using an internet connection. An internet cloud infrastructure increases organization productivity and efficiency by ensuring that our data is always accessible.

4) Low maintenance cost

Cloud computing reduces both hardware and software maintenance costs for organizations.

5) Mobility

Cloud computing allows us to easily access all cloud data via mobile.

6) IServices in the pay-per-use model

Cloud computing offers Application Programming Interfaces (APIs) to the users for access services on the cloud and pays the charges as per the usage of service.

7) Unlimited storage capacity

Cloud offers us a huge amount of storing capacity for storing our important data such as documents, images, audio, video, etc. in one place.

8) Data security

Data security is one of the biggest advantages of cloud computing. Cloud offers many advanced features related to security and ensures that data is securely stored and handled.

Disadvantages of Cloud Computing

1) Internet Connectivity

As you know, in cloud computing, every data (image, audio, video, etc.) is stored on the cloud, and we access these data through the cloud by using the internet connection. If you do not have good internet connectivity, you cannot access these data. However, we have no any other way to access data from the cloud.

2) Vendor lock-in

Vendor lock-in is the biggest disadvantage of cloud computing. Organizations may face problems when transferring their services from one vendor to another. As different vendors provide different platforms, that can cause difficulty moving from one cloud to another.

3) Limited Control

As we know, cloud infrastructure is completely owned, managed, and monitored by the service provider, so the cloud users have less control over the function and execution of services within a cloud infrastructure.

4) Security

Although cloud service providers implement the best security standards to store important information. But, before adopting cloud technology, you should be aware that you will be sending all your organization's sensitive information to a third party, i.e., a cloud computing service provider. While sending the data on the cloud, there may be a chance that your organization's information is hacked by Hackers.

1.2 Cloud Computing Architecture

As we know, cloud computing technology is used by both small and large organizations to store the information in cloud and access it from anywhere at anytime using the internet connection

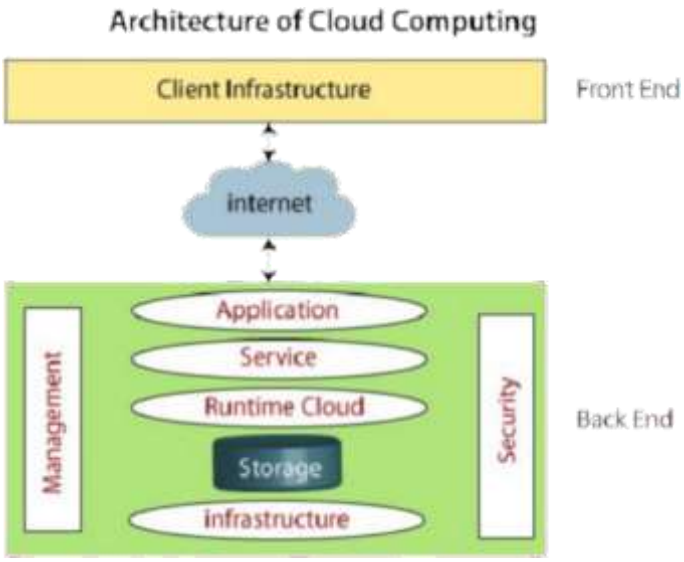
1.2.1 Introduction to Cloud Architecture

Cloud computing architecture is a combination of **service-oriented architecture** and **event-driven architecture**.

Cloud computing architecture is divided into the following two parts -

- Front End
- Back End

The below diagram shows the architecture of cloud computing -



1.2.2 Frond End and Back End

Front End

The front end is used by the client. It contains client-side interfaces and applications that are required to access the cloud computing platforms. The front end includes web servers (including Chrome, Firefox, internet explorer, etc.), thin & fat clients, tablets, and mobile devices.

Back End

The back end is used by the service provider. It manages all the resources that are required to provide cloud computing services. It includes a huge amount of data storage, security mechanism, virtual machines, deploying models, servers, traffic control mechanisms, etc.

1.2.3 Components of Cloud Computing Architecture

There are the following components of cloud computing architecture -

1. Client Infrastructure

Client Infrastructure is a Front end component. It provides GUI (Graphical User Interface) to interact with the cloud.

2. Application

The application may be any software or platform that a client wants to access.

3. Service

A Cloud Services manages that which type of service you access according to the client's requirement.

4. Runtime Cloud

Runtime Cloud provides the **execution and runtime environment** to the virtual machines.

5. Storage

Storage is one of the most important components of cloud computing. It provides a huge amount of storage capacity in the cloud to store and manage data.

6. Infrastructure

It provides services on the **host level**, **application level**, and **network level**. Cloud infrastructure includes hardware and software components such as servers, storage, network devices, virtualization software, and other storage resources that are needed to support the cloud computing model.

7. Management

Management is used to manage components such as application, service, runtime cloud, storage, infrastructure, and other security issues in the backend and establish coordination between them.

8. Security

Security is an in-built back end component of cloud computing. It implements a security mechanism in the back end.

9. Internet

The Internet is medium through which front end and back end can interact and

communicate with each other.

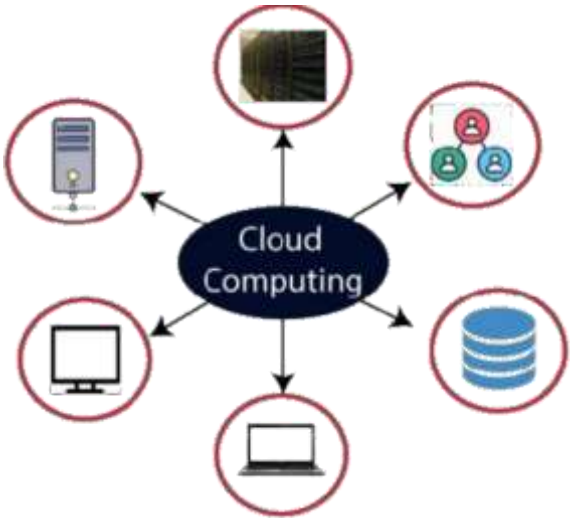
1.3 Cloud Computing and Other Computing Models

1.3.1 Cloud Computing and Grid Computing

- **Cloud Computing**

Cloud computing uses a **client-server** architecture to deliver computing resources such as servers, storage, databases, and software over the cloud (Internet) with pay- as-you-go pricing.

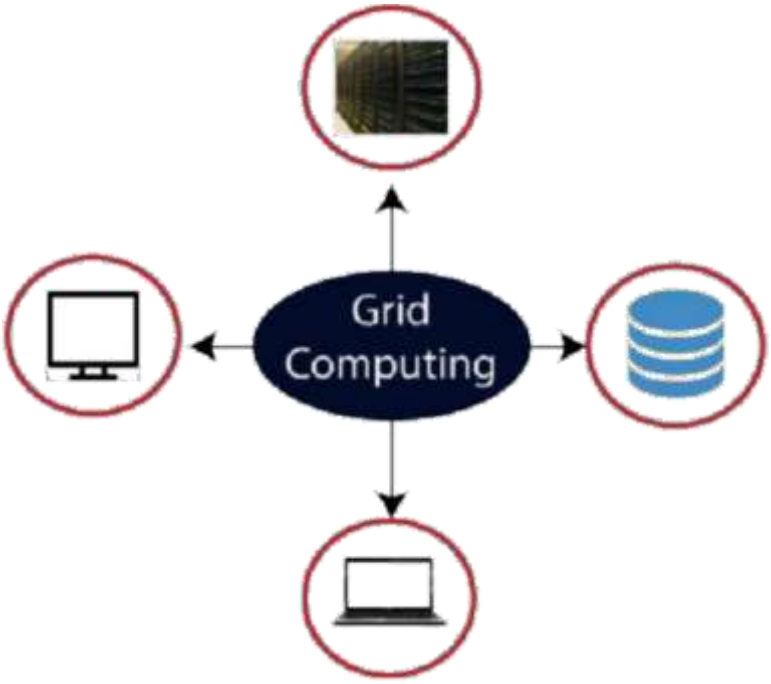
Cloud computing becomes a very popular option for organizations by providing various advantages, including cost-saving, increased productivity, efficiency, performance, data back-ups, disaster recovery, and security.



- **Grid Computing**

Grid computing is also called as "**distributed computing**." It links multiple computing resources (PC's, workstations, servers, and storage elements) together and provides a mechanism to access them.

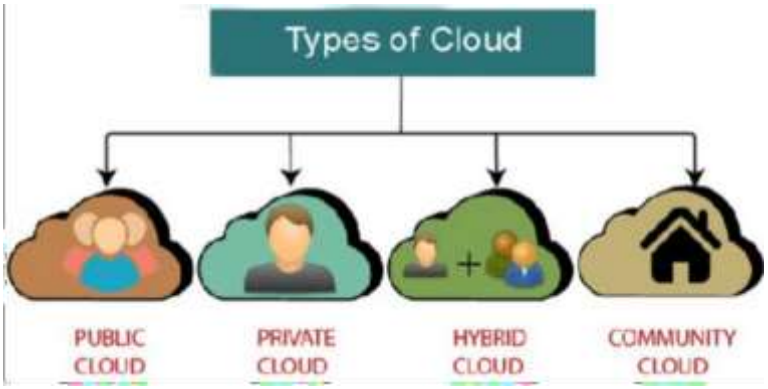
The main advantages of grid computing are that it increases user productivity by providing transparent access to resources, and work can be completed more quickly.



1.3.2 Cloud Computing and Client Server Computing

1.4 Types of Cloud

There are the following 4 types of cloud that you can deploy according to the organization's needs-

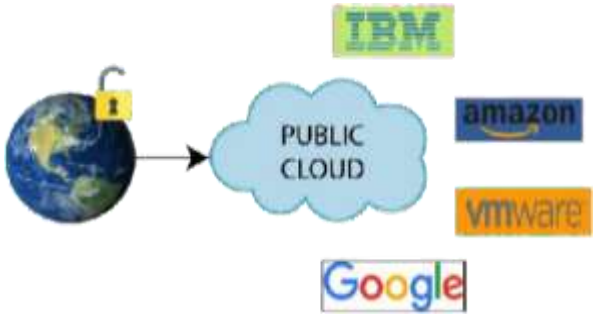


1.4.1 Public Cloud

Public cloud is **open to all** to store and access information via the Internet using the pay-per-usage method.

In public cloud, computing resources are managed and operated by the Cloud Service Provider (CSP).

Example: Amazon elastic compute cloud (EC2), IBM SmartCloud Enterprise, Microsoft, Google App Engine, Windows Azure Services Platform.



Advantages of Public Cloud

There are the following advantages of Public Cloud -

- Public cloud is owned at a lower cost than the private and hybrid cloud.
- Public cloud is maintained by the cloud service provider, so do not need to worry about the maintenance.
- Public cloud is easier to integrate. Hence it offers a better flexibility approach to consumers.
- Public cloud is location independent because its services are delivered through the internet.
- Public cloud is highly scalable as per the requirement of computing resources.
- It is accessible by the general public, so there is no limit to the number of users.

Disadvantages of Public Cloud

- Public Cloud is less secure because resources are shared publicly.
- Performance depends upon the high-speed internet network link to the cloud provider.
- The Client has no control of data.

1.4.2 Private Cloud

Private cloud is also known as an **internal cloud** or **corporate cloud**. It is used by organizations to build and manage their own data centers internally or by the third party. It can be deployed using Opensource tools such as Openstack and Eucalyptus.

Based on the location and management, National Institute of Standards and Technology (NIST) divide private cloud into the following two parts-

- On-premise private cloud
- Outsourced private cloud

Advantages of Private Cloud

There are the following advantages of the Private Cloud -

- Private cloud provides a high level of security and privacy to the users.
- Private cloud offers better performance with improved speed and space capacity.
- It allows the IT team to quickly allocate and deliver on-demand IT resources.
- The organization has full control over the cloud because it is managed by the organization itself. So, there is no need for the organization to depends on anybody.
- It is suitable for organizations that require a separate cloud for their personal use and data security is the first priority.

Disadvantages of Private Cloud

- Skilled people are required to manage and operate cloud services.
- Private cloud is accessible within the organization, so the area of operations is limited.
- Private cloud is not suitable for organizations that have a high user base, and organizations that do not have the prebuilt infrastructure, sufficient manpower to maintain and manage the cloud.

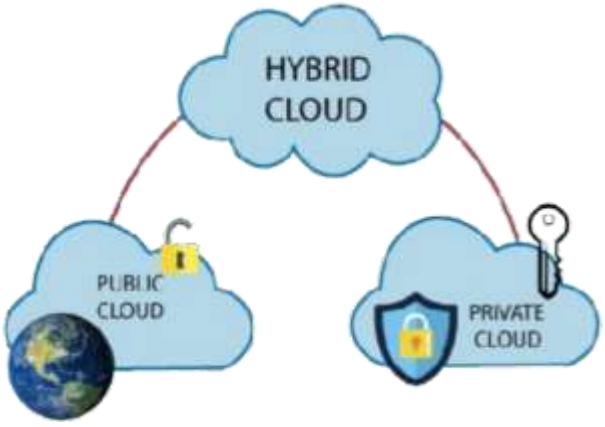
1.4.3 Hybrid Cloud

Hybrid Cloud is a combination of the public cloud and the private cloud. we can say:

Hybrid Cloud = Public Cloud + Private Cloud

Hybrid cloud is partially secure because the services which are running on the public cloud can be accessed by anyone, while the services which are running on a private cloud can be accessed only by the organization's users.

Example: Google Application Suite (Gmail, Google Apps, and Google Drive), Office 365 (MS Office on the Web and One Drive), Amazon Web Services.



Advantages of Hybrid Cloud

There are the following advantages of Hybrid Cloud -

- Hybrid cloud is suitable for organizations that require more security than the public cloud.

- Hybrid cloud helps you to deliver new products and services more quickly.
- Hybrid cloud provides an excellent way to reduce the risk.
- Hybrid cloud offers flexible resources because of the public cloud and secure resources because of the private cloud.

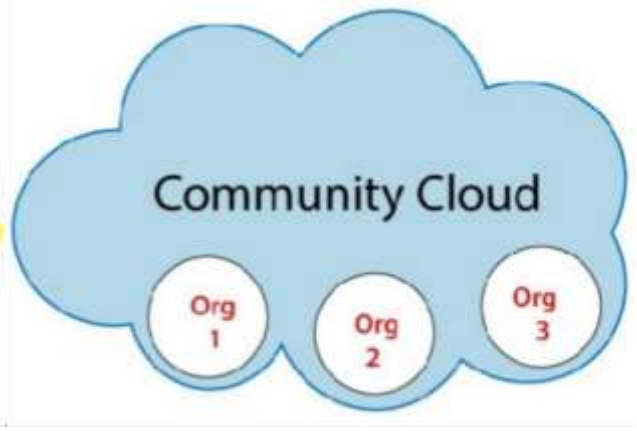
Disadvantages of Hybrid Cloud

- In Hybrid Cloud, security feature is not as good as the private cloud.
- Managing a hybrid cloud is complex because it is difficult to manage more than one type of deployment model.
- In the hybrid cloud, the reliability of the services depends on cloud service providers.

1.4.4 Community Cloud

Community cloud allows systems and services to be accessible by a group of several organizations to share the information between the organization and a specific community. It is owned, managed, and operated by one or more organizations in the community, a third party, or a combination of them.

Example: Health Care community cloud



Advantages of Community Cloud

There are the following advantages of Community Cloud -

- Community cloud is cost-effective because the whole cloud is being shared by several organizations or communities.

- Community cloud is suitable for organizations that want to have a collaborative cloud with more security features than the public cloud.
- It provides better security than the public cloud.
- It provdes collaborative and distributive environment.
- Community cloud allows us to share cloud resources, infrastructure, and other capabilities among various organizations.

Disadvantages of Community Cloud

- Community cloud is not a good choice for every organization.
- Security features are not as good as the private cloud.
- It is not suitable if there is no collaboration.
- The fixed amount of data storage and bandwidth is shared among all community members.

1.4.5 Comparison of Cloud Types

Parameter	Public Cloud	Private Cloud	Hybrid Cloud	Community Cloud
Host	Service provider	Enterprise (Third party)	Enterprise (Third party)	Community (Third party)
Users	General public	Selected users	Selected users	Community members
Access	Internet	Internet, VPN	Internet, VPN	Internet, VPN
Owner	Service provider	Enterprise	Enterprise	Community

1.5 ervice Models of Cloud Computing:

Cloud computing offers the following three type of services:-

1.5.1 Software as a Service (SaaS):

It is also known as **cloud application services**. Mostly, SaaS applications run directly through the web browser means we do not require to download and install these applications.

Characteristics of SaaS

There are the following characteristics of SaaS -

- Managed from a central location
- Hosted on a remote server
- Accessible over the internet
- Users are not responsible for hardware and software updates. Updates are applied automatically.
- The services are purchased on the pay-as-per-use basis

Example: Google Apps, Salesforce Dropbox, Slack, Hubspot, Cisco WebEx.

SaaS is also known as "**On-Demand Software**". It is a software distribution model in which services are hosted by a cloud service provider. These services are available to end-users over the internet so, the end-users do not need to install any software on their devices to access these services.

There are the following services provided by SaaS providers -

Business Services - SaaS Provider provides various business services to start-up the business. The SaaS business services include **ERP** (Enterprise Resource Planning), **CRM** (Customer Relationship Management), **billing**, and **sales**.

Document Management - SaaS document management is a software application offered by a third party (SaaS providers) to create, manage, and track electronic documents.

Social Networks - As we all know, social networking sites are used by the general public, so social networking service providers use SaaS for their convenience and handle the general public's information.

Mail Services - To handle the unpredictable number of users and load on e-mail services, many e-mail providers offering their services using SaaS.

Advantages of SaaS cloud computing layer

1) SaaS is easy to buy

SaaS pricing is based on a monthly fee or annual fee subscription, so it allows organizations to access business functionality at a low cost, which is less than licensed applications.

Unlike traditional software, which is sold as a licensed based with an up-front cost (and often an optional ongoing support fee), SaaS providers are generally pricing the applications using a subscription fee, most commonly a monthly or annually fee.

2. One to Many

SaaS services are offered as a one-to-many model means a single instance of the application is shared by multiple users.

3. Less hardware required for SaaS

The software is hosted remotely, so organizations do not need to invest in additional hardware.

4. Low maintenance required for SaaS

Software as a service removes the need for installation, set-up, and daily maintenance for the organizations. The initial set-up cost for SaaS is typically less than the enterprise software. SaaS vendors are pricing their applications based on some usage parameters, such as a number of users using the

application. So SaaS does easy to monitor and automatic updates.

5. No special software or hardware versions required

All users will have the same version of the software and typically access it through the web browser. SaaS reduces IT support costs by outsourcing hardware and software maintenance and support to the IaaS provider.

6. Multidevice support

SaaS services can be accessed from any device such as desktops, laptops, tablets, phones, and thin clients.

7. API Integration

SaaS services easily integrate with other software or services through standard APIs.

8. No client-side installation

SaaS services are accessed directly from the service provider using the internet connection, so do not need to require any software installation.

Disadvantages of SaaS cloud computing layer

1) Security

Actually, data is stored in the cloud, so security may be an issue for some users. However, cloud computing is not more secure than in-house deployment.

2) Latency issue

Since data and applications are stored in the cloud at a variable distance from the end-user, there is a possibility that there may be greater latency when interacting with the application compared to local deployment. Therefore, the SaaS model is not suitable for applications whose demand response time is in milliseconds.

3) Total Dependency on Internet

Without an internet connection, most SaaS applications are not usable.

4) Switching between SaaS vendors is difficult

Switching SaaS vendors involves the difficult and slow task of transferring the very large data files over the internet and then converting and importing them into another SaaS also.

1.5.2 Platform as a Service (PaaS):

It t is also known as **cloud platform services**. It is quite similar to SaaS, but the difference is that PaaS provides a platform for software creation, but using SaaS, we can access software over the internet without the need of any platform.

There are the following characteristics of PaaS -

- Accessible to various users via the same development application.
- Integrates with web services and databases.
- Builds on virtualization technology, so resources can easily be scaled up or down as per the organization's need.
- Support multiple languages and frameworks.
- Provides an ability to "**Auto-scale**".

Example: Windows Azure, Force.com, Magento Commerce Cloud, OpenShift.

Platform as a Service (PaaS) provides a runtime environment. It allows programmers to easily create, test, run, and deploy web applications. You can purchase these applications from a cloud service provider on a pay-as-per use basis and access them using the Internet connection. In PaaS, back end scalability is managed by the cloud service provider, so end- users do not need to worry about managing the infrastructure.

Example: Google App Engine, Force.com, Joyent, Azure.

1. Programming languages

PaaS providers provide various programming languages for the developers to develop the applications. Some popular programming languages provided by PaaS providers are Java, PHP, Ruby, Perl, and Go.

2. Application frameworks

PaaS providers provide application frameworks to easily understand the application development. Some popular application frameworks provided by PaaS providers are Node.js, Drupal, Joomla, WordPress, Spring, Play, Rack, and Zend.

3. Databases

PaaS providers provide various databases such as ClearDB, PostgreSQL, MongoDB, and Redis to communicate with the applications.

4. Other tools

PaaS providers provide various other tools that are required to develop, test, and deploy the applications.

Advantages of PaaS

There are the following advantages of PaaS -

1) Simplified Development

PaaS allows developers to focus on development and innovation without worrying about infrastructure management.

2) Lower risk

No need for up-front investment in hardware and software. Developers only need a PC and an internet connection to start building applications.

3) Prebuilt business functionality

Some PaaS vendors also provide already defined business functionality so that

users can avoid building everything from very scratch and hence can directly start the projects only.

4) Instant community

PaaS vendors frequently provide online communities where the developer can get the ideas to share experiences and seek advice from others.

5) Scalability

Applications deployed can scale from one to thousands of users without any changes to the applications.

Disadvantages of PaaS cloud computing layer

1) Vendor lock-in

One has to write the applications according to the platform provided by the PaaS vendor, so the migration of an application to another PaaS vendor would be a problem.

2) Data Privacy

Corporate data, whether it can be critical or not, will be private, so if it is not located within the walls of the company, there can be a risk in terms of privacy of data.

3) Integration with the rest of the systems applications

It may happen that some applications are local, and some are in the cloud. So there will be chances of increased complexity when we want to use data which in the cloud with the local data.

1.5.3 Infrastructure as a Service (IaaS):

It is also known as **cloud infrastructure services**. It is responsible for managing applications data, middleware, and runtime environments.

Characteristics of IaaS

- Resources are available as a service
- Services are highly scalable
- Dynamic and flexible
- GUI and API-based access
- Automated administrative tasks

Example: Amazon Web Services (AWS) EC2, Google Compute Engine (GCE), Cisco Metapod.

Iaas is also known as **Hardware as a Service (HaaS)**. It is one of the layers of the cloud computing platform. It allows customers to outsource their IT infrastructures such as servers, networking, processing, storage, virtual machines, and other resources. Customers access these resources on the Internet using a pay-as-per use model.

IaaS is offered in three models: public, private, and hybrid cloud. The private cloud implies that the infrastructure resides at the customer-premise. In the case of public cloud, it is located at the cloud computing platform vendor's data center, and the hybrid cloud is a combination of the two in which the customer selects the best of both public cloud or private cloud.

IaaS provider provides the following services -

1. **Compute:** Computing as a Service includes virtual central processing units and virtual main memory for the Vms that is provisioned to the end- users.
2. **Storage:** IaaS provider provides back-end storage for storing files.
3. **Network:** Network as a Service (NaaS) provides networking components such as routers, switches, and bridges for the Vms.
4. **Load balancers:** It provides load balancing capability at the infrastructure layer.

Advantages of IaaS cloud computing layer

1. Shared infrastructure

IaaS allows multiple users to share the same physical infrastructure.

2. Web access to the resources

Iaas allows IT users to access resources over the internet.

3. Pay-as-per-use model

IaaS providers provide services based on the pay-as-per-use basis. The users are required to pay for what they have used.

4. Focus on the core business

IaaS providers focus on the organization's core business rather than on IT infrastructure.

5. On-demand scalability

On-demand scalability is one of the biggest advantages of IaaS. Using IaaS, users do not worry about to upgrade software and troubleshoot the issues related to hardware components.

Disadvantages of IaaS cloud computing layer

1. Security

Security is one of the biggest issues in IaaS. Most of the IaaS providers are not able to provide 100% security.

2. Maintenance & Upgrade

Although IaaS service providers maintain the software, but they do not upgrade the software for some organizations.

3. Interoperability issues

It is difficult to migrate VM from one IaaS provider to the other, so the customers might face problem related to vendor lock-in.

1.5.4 Difference between IAAS, PAAS and SAAS:

IaaS	Paas	SaaS
It provides a virtual data center to store information and create platforms for app development, testing, and deployment.	It provides virtual platforms and tools to create, test, and deploy apps.	It provides web software and apps to complete business tasks.
It provides access to resources such as virtual machines, virtual storage, etc.	It provides runtime environments and deployment tools for applications.	It provides software as a service to the end-users.
It is used by network architects.	It is used by developers.	It is used by end users.
IaaS provides only Infrastructure.	PaaS provides Infrastructure+Platform.	SaaS provides Infrastructure+Platform +Software.

Chapter 2

RESILIENT MICROSERVICES FOR JAVA DEVELOPERS

2.1 Introduction: The Need for Resilience in Modern Software Systems

The rapid evolution of digital technology has fundamentally transformed the expectations placed on software systems. Applications today are not isolated programs running within a controlled environment; they are distributed platforms serving millions of users across geographical regions, devices, and networks. Whether it is an online banking portal, a ride-sharing application, a streaming service, or an enterprise resource planning system, users demand uninterrupted availability, fast response times, secure transactions, and seamless scalability. Even brief outages can result in significant financial loss and erosion of customer trust.

Historically, enterprise systems were designed under the assumption that failures were rare events. However, in distributed cloud environments, failures are not exceptional; they are normal. Networks become congested, containers crash, servers reboot, databases experience replication delays, and cloud infrastructure occasionally becomes unavailable. In such an environment, building software that merely "works" is insufficient. Systems must be designed to continue functioning despite failures. This property is known as resilience.

Resilience is the capability of a system to withstand disruptions, recover from partial failures, and maintain acceptable service levels under stress. It does not imply that failures will never occur; rather, it assumes that failures are inevitable and prepares the system to handle them gracefully. For Java developers building cloud-native applications, resilience is achieved through architectural design, fault-tolerance mechanisms, security controls, monitoring systems, and automated infrastructure management.

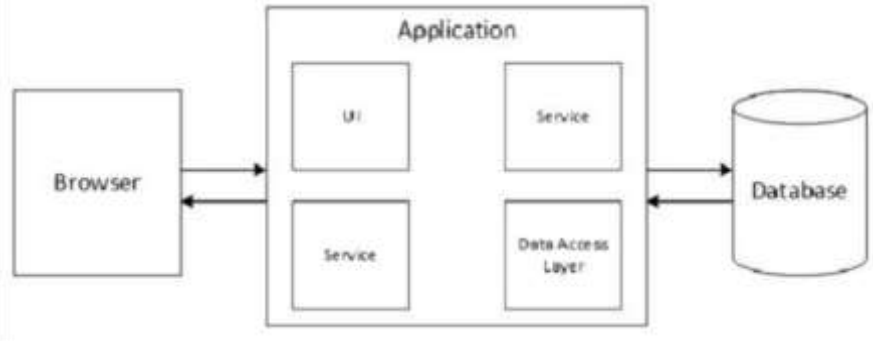
The shift from monolithic systems to microservices architecture has significantly improved scalability and flexibility. However, it has also increased architectural complexity. Microservices introduce distributed communication, network dependencies, and decentralized data management, all of which increase the potential for failure. Therefore, resilience is not optional—it is a foundational requirement of modern microservices systems.

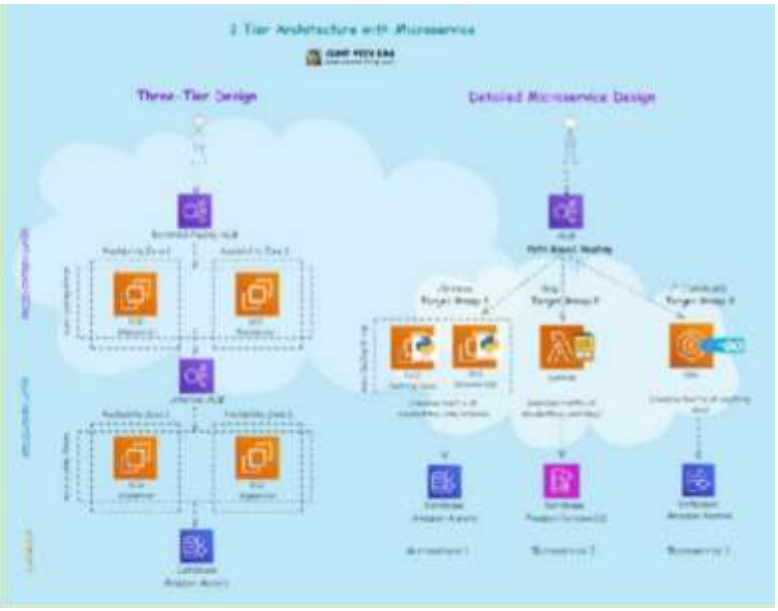
This chapter provides an in-depth exploration of how resilient microservices can be designed and implemented using Java technologies such as Spring Boot, containerization using Docker, orchestration with Kubernetes, deployment on Amazon Elastic Kubernetes Service (EKS), and automated scaling mechanisms. The discussion integrates conceptual understanding with real- world examples to ensure clarity and practical relevance.

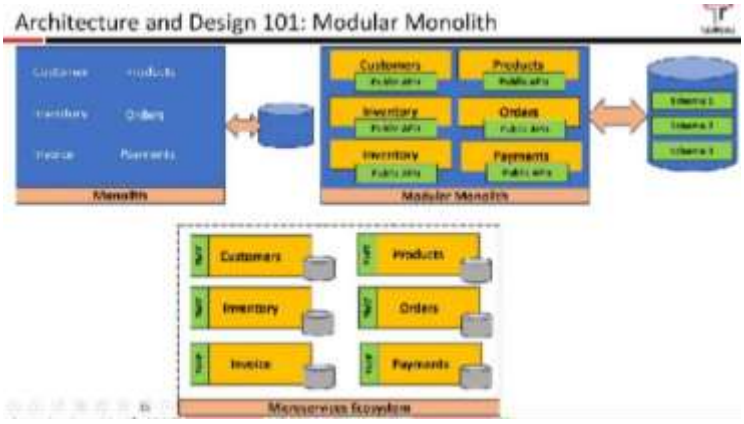
2.2 Understanding Monolithic Architecture and Its Limitations

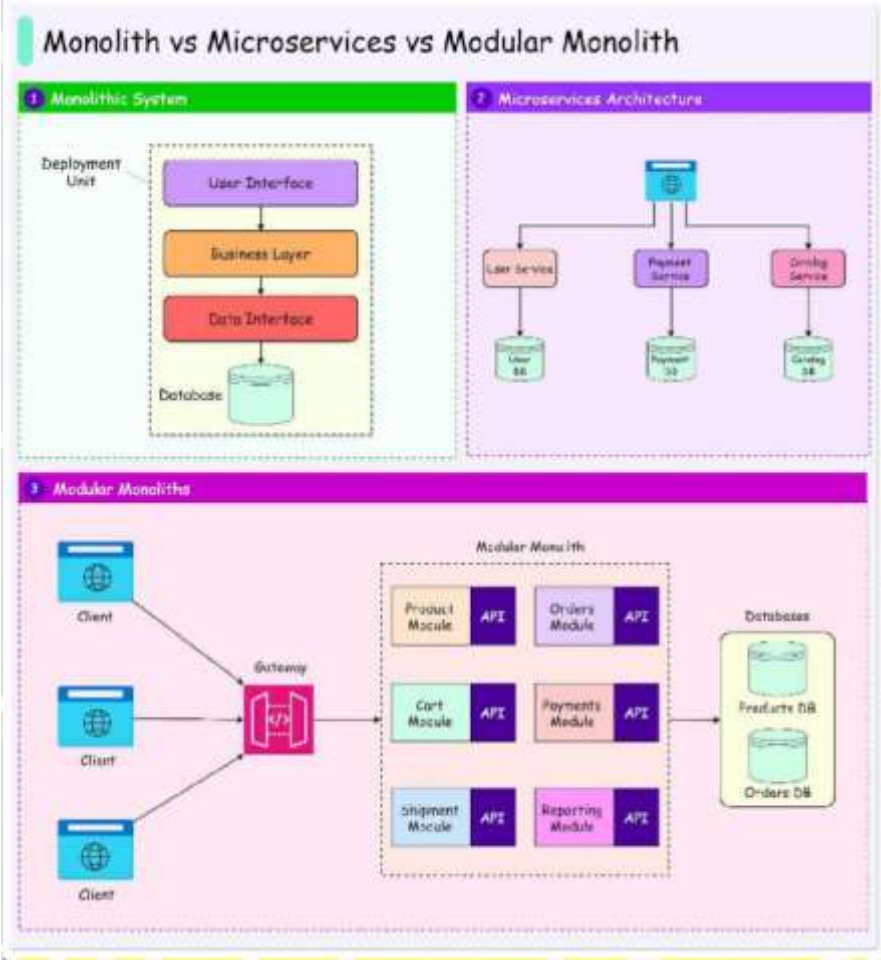
Before exploring microservices, it is important to understand the architectural model that preceded it: the monolith. In a monolithic architecture, all components of an application are developed, compiled, and deployed as a single unit. The presentation layer, business logic layer, and data access layer coexist within the same codebase and share the same runtime environment and database.

Figure 2.1: Structure of a Monolithic Architecture









In the diagram above, the user interface communicates directly with business logic components, which in turn interact with a centralized database. All modules are tightly coupled and deployed together.

At the initial stages of application development, monolithic architecture offers simplicity. Developers can invoke methods within the same process without worrying about network communication. Transaction management is straightforward because all modules share a common database. Debugging can be easier because the system runs as a single application.

However, as the application grows, several limitations become apparent. The codebase becomes large and difficult to understand. A small change in one module may unintentionally affect other modules due to tight coupling. Deployment cycles slow down because the entire application must be rebuilt and redeployed even for minor updates. Testing becomes more complex and time- consuming.

Scalability presents another major challenge. Suppose an online retail system experiences high traffic in its payment processing module during a festive sale. In a monolithic system, scaling requires replicating the entire application, including modules that may not need additional resources. This leads to inefficient resource utilization and increased infrastructure costs.

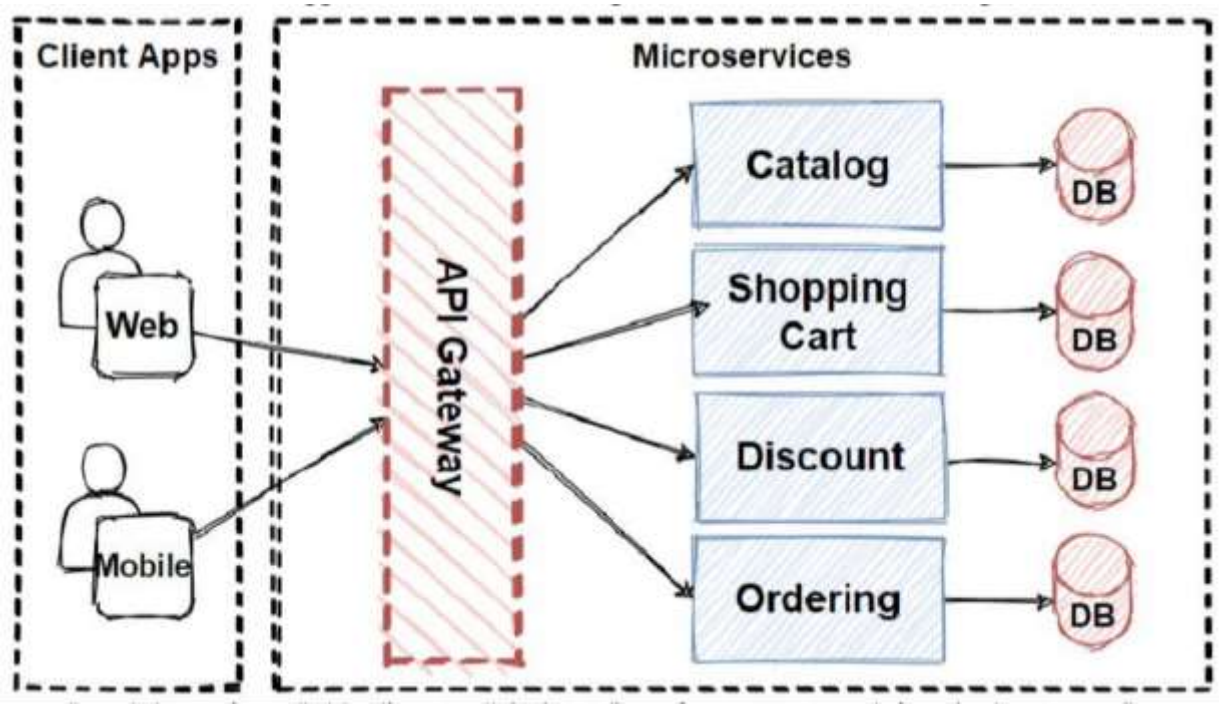
Moreover, fault isolation is limited in monoliths. If a memory leak occurs in one component, it can destabilize the entire application. This lack of isolation increases system fragility.

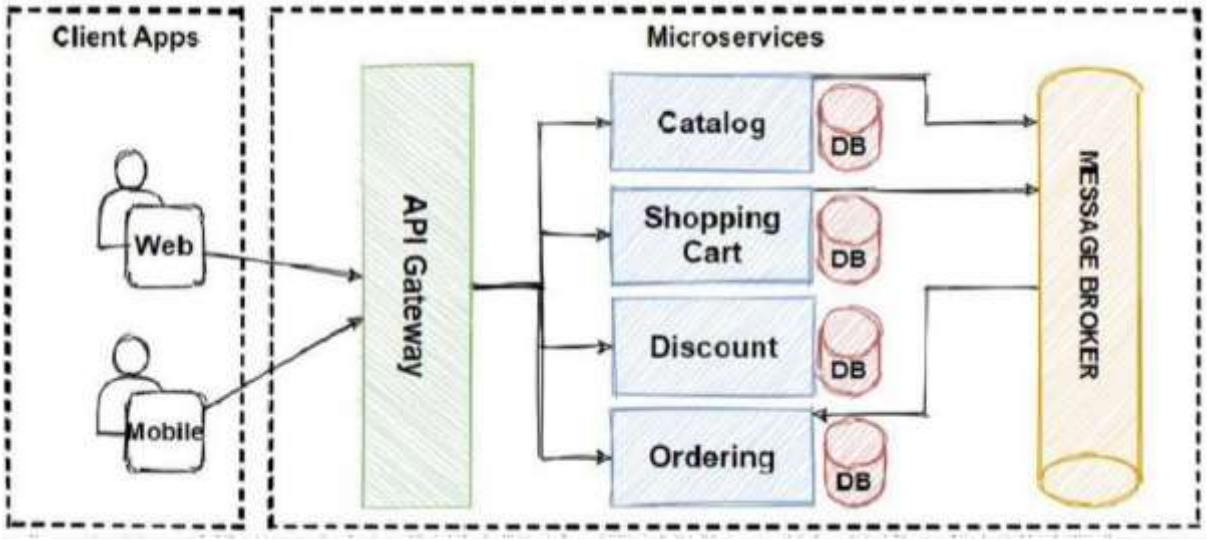
These challenges motivated the shift toward microservices architecture.

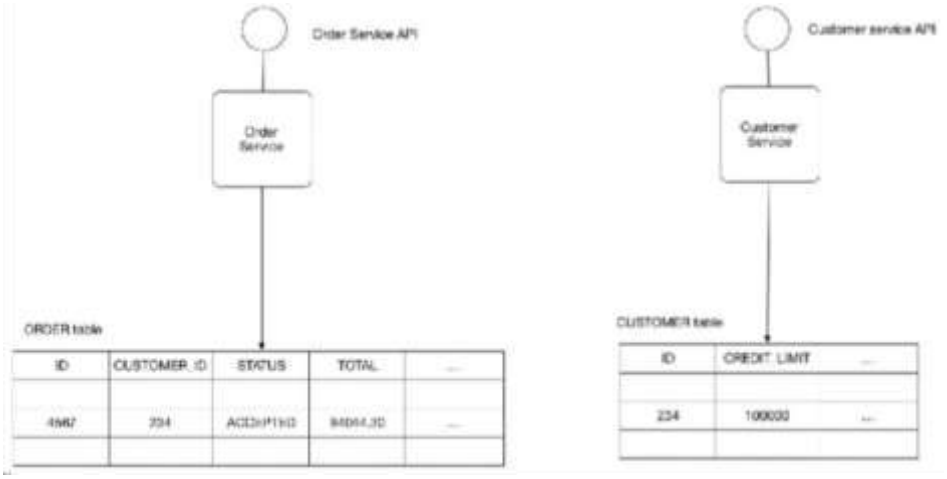
2.3 Emergence of Microservices Architecture

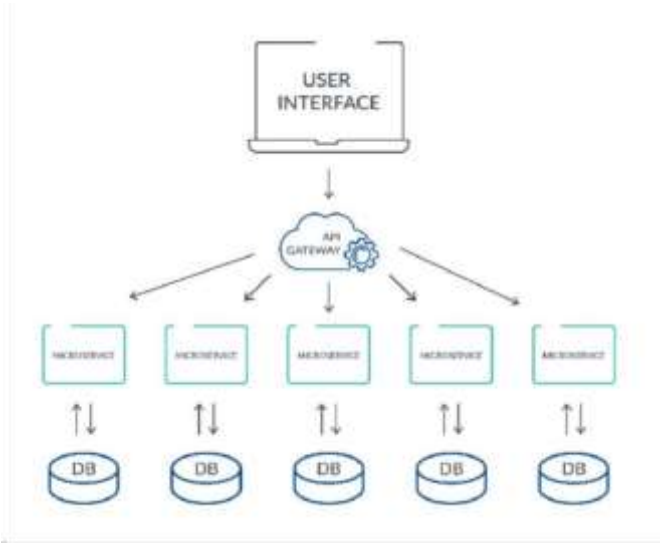
Microservices architecture emerged as a response to the limitations of monolithic systems. Instead of building a single large application, microservices architecture decomposes functionality into smaller, autonomous services. Each service is responsible for a specific business capability and operates independently of others.

Figure 2.2: Microservices Architecture Model









In this model, an API gateway acts as a single entry point for client requests. Behind the gateway, multiple services handle different responsibilities. Each service may have its own database, ensuring data autonomy and reducing coupling.

Consider an online food delivery platform as an example. Instead of a single monolithic

system, the application can be divided into services such as User Management, Restaurant Management, Order Processing, Payment Handling, and Delivery Tracking. Each service can be developed by a separate team and deployed independently. If the Delivery Tracking service requires additional resources due to high demand, it can be scaled without affecting other services.

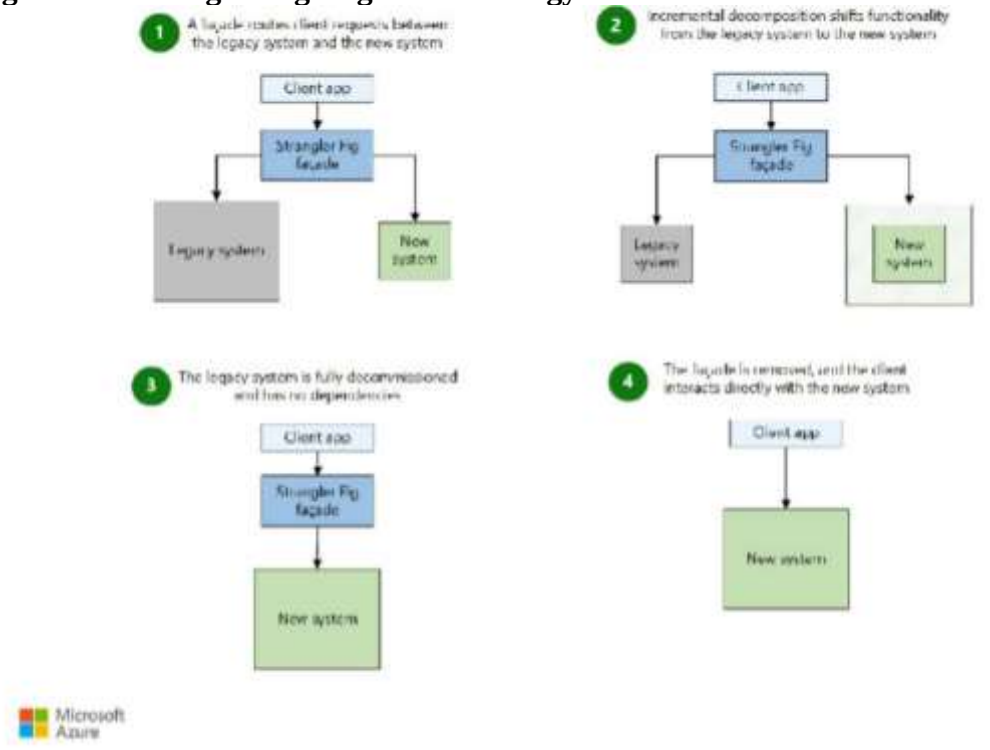
This modular approach improves agility and supports continuous deployment. However, microservices also introduce distributed system challenges. Services communicate over the network, which introduces latency and potential failures. Data consistency becomes more complex when each service maintains its own database. Therefore, resilience mechanisms must be incorporated from the design stage.

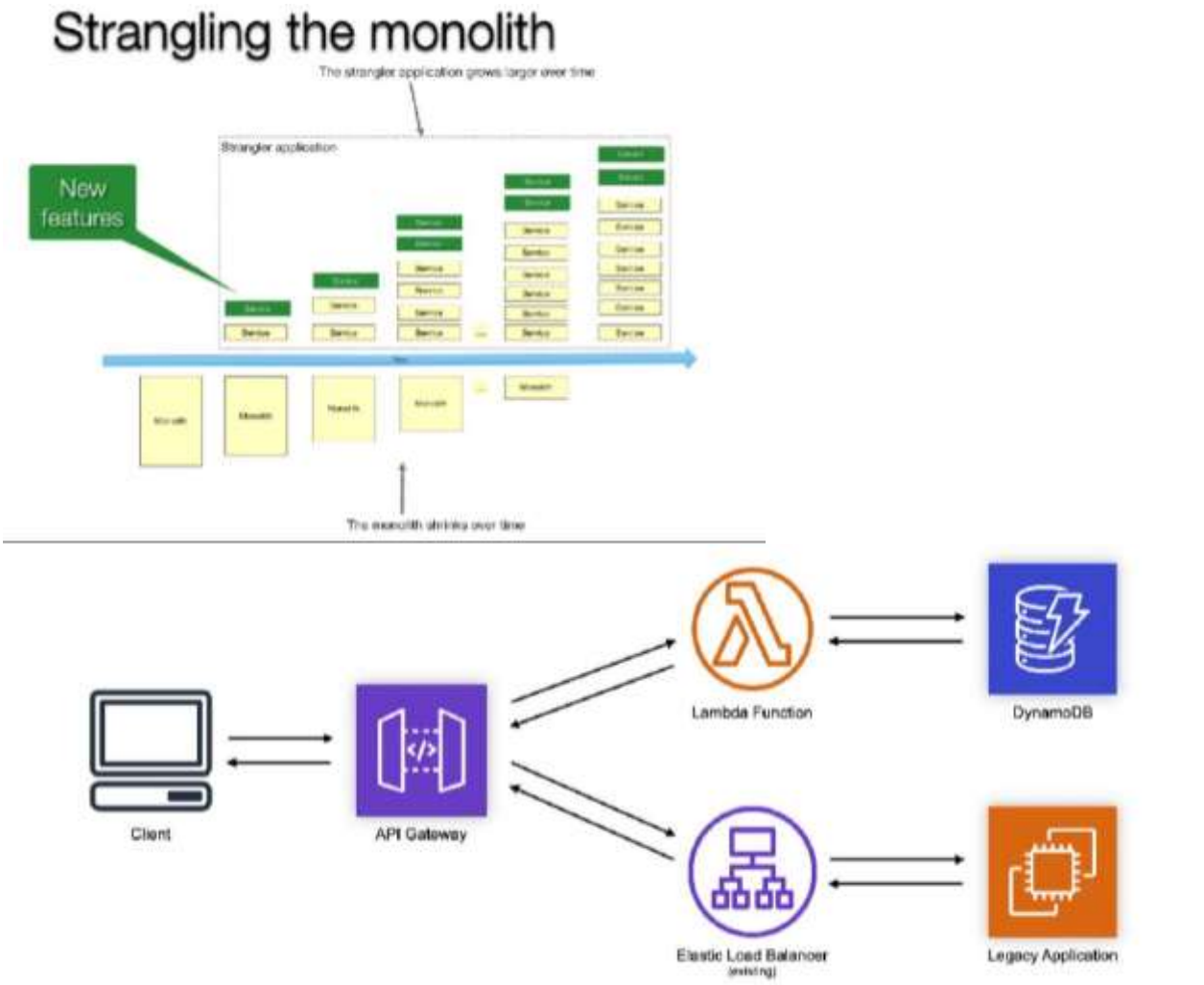
2.4 Migrating Legacy Java Applications to Microservices

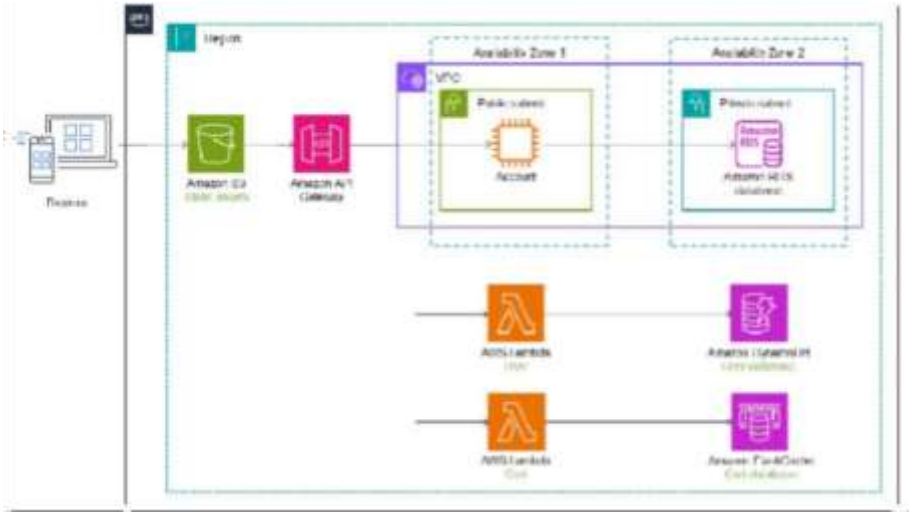
Transitioning from a monolithic system to microservices cannot be accomplished abruptly. Rewriting an entire application is risky and often impractical due to time constraints, cost, and business continuity requirements. Instead, organizations adopt incremental migration strategies.

One widely recognized strategy is the Strangler Fig pattern.

Figure 2.3: Strangler Fig Migration Strategy







The Strangler Fig pattern involves gradually replacing portions of the monolith with new microservices. An API gateway routes certain requests to newly developed services while others continue to be handled by the monolith. Over time, more functionalities are extracted until the monolith becomes obsolete.

For example, in a banking application, the loan processing module may be extracted first because it experiences high traffic. Once stabilized, other modules such as customer notifications or account management can be extracted. This gradual process reduces risk and ensures uninterrupted service.

Domain-Driven Design (DDD) plays an important role during this migration. DDD encourages identifying bounded contexts—logical boundaries within the business domain. Each bounded context can evolve into an independent microservice. This approach ensures that service boundaries align with business requirements rather than technical convenience.

2.5 Performance Engineering in Microservices

While microservices improve scalability and modularity, they introduce performance challenges. In monolithic systems, method calls occur within the same process and memory space. In microservices, service interactions occur over the network, requiring serialization and deserialization of data.

Network latency is unavoidable. If a single user request triggers calls to multiple services, total response time increases. This phenomenon is often referred to as the "distributed monolith" problem when not properly designed.

For example, consider an e-commerce checkout process. The Order Service may call Inventory Service to verify stock availability, Payment Service to process payment, and Notification Service to send confirmation. If each service takes 150 milliseconds to respond, the cumulative delay can significantly impact user experience.

To address these challenges, performance engineering techniques such as caching, asynchronous messaging, and load balancing are employed. Java developers often use Spring Boot Actuator to monitor application metrics and integrate with monitoring tools such as Prometheus. Efficient database indexing and optimized API design also contribute to improved performance.

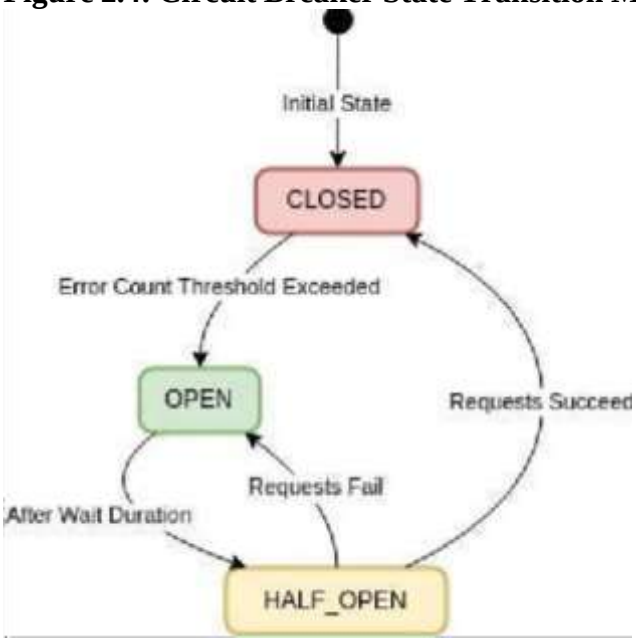
Performance must be continuously monitored and optimized, as microservices operate in dynamic cloud environments where workloads fluctuate unpredictably.

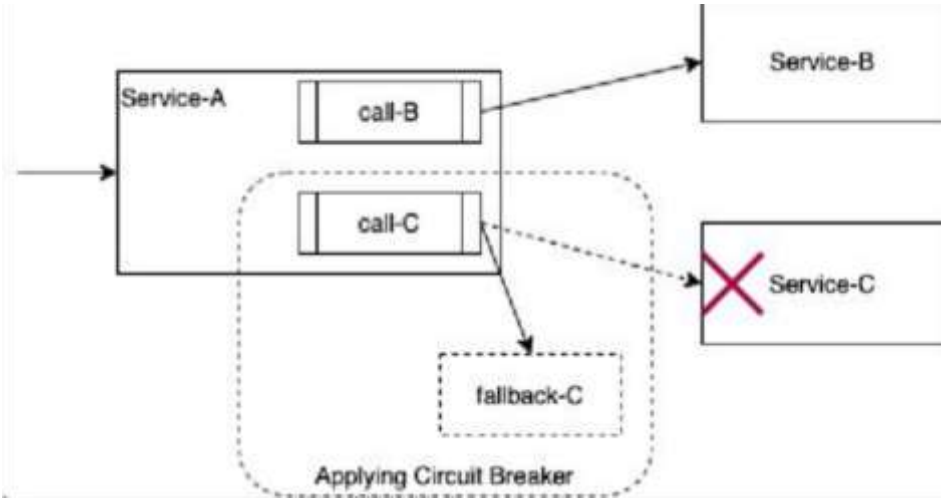
2.6 Resilience Patterns and Fault Tolerance

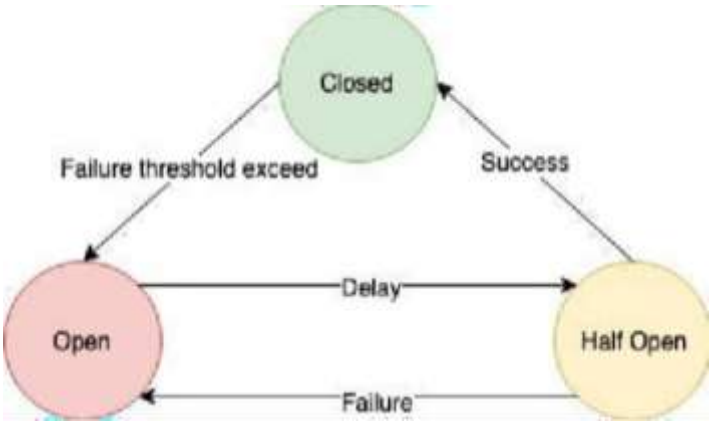
Resilience in microservices requires systematic handling of failures. Several architectural patterns have been developed to manage faults effectively.

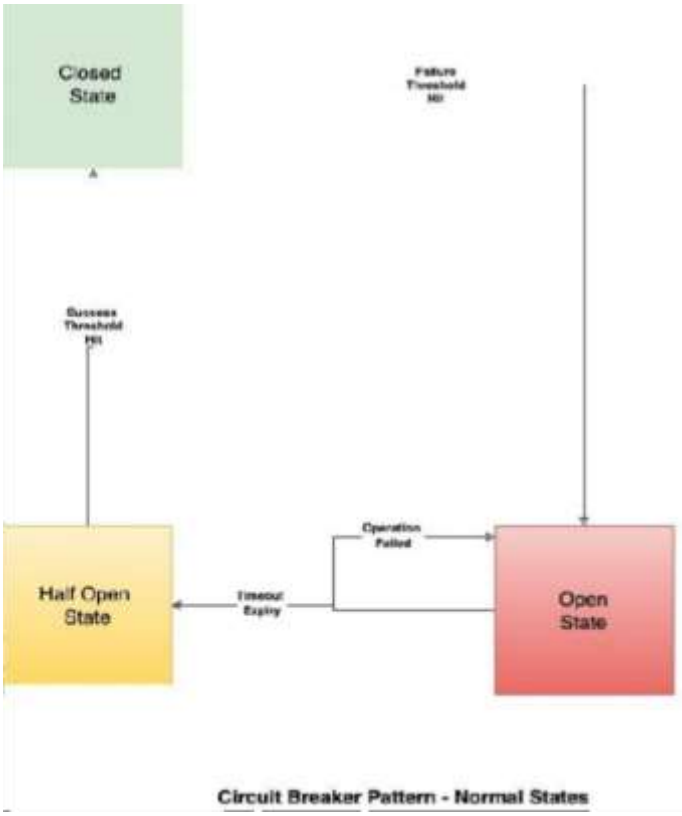
One of the most important patterns is the Circuit Breaker pattern.

Figure 2.4: Circuit Breaker State Transition Model









Circuit Breaker Pattern - Normal States

The Circuit Breaker monitors service calls and detects repeated failures. If failures exceed a predefined threshold, the circuit transitions to an open state, blocking further calls to the failing service. After a waiting period, it transitions to a half-open state to test recovery. This prevents cascading failures and reduces system overload.

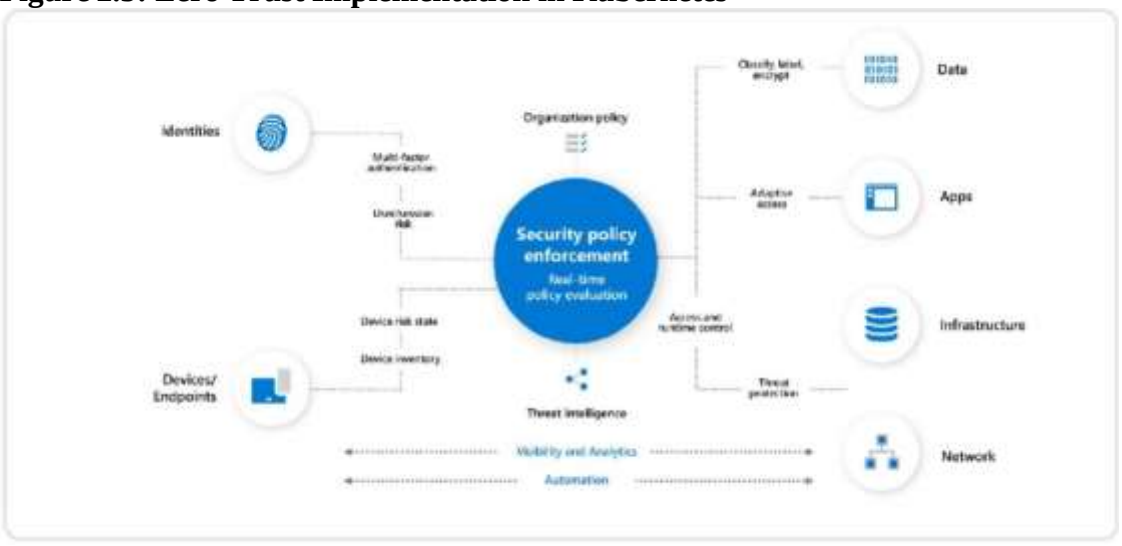
The Bulkhead pattern isolates resources so that failure in one service does not consume all system resources. Retry mechanisms attempt to recover from transient failures, while timeout settings ensure that requests do not wait indefinitely.

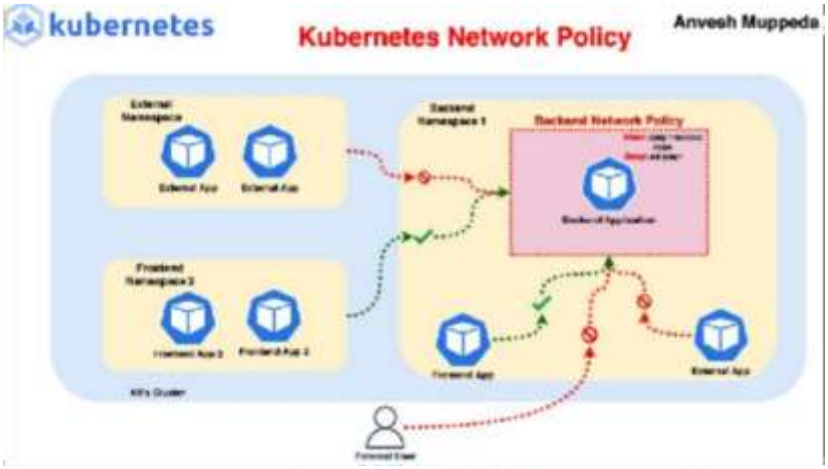
Together, these patterns enable systems to degrade gracefully. For example, if a recommendation service fails in an e-commerce system, the application can still allow users to browse products and complete purchases without recommendations.

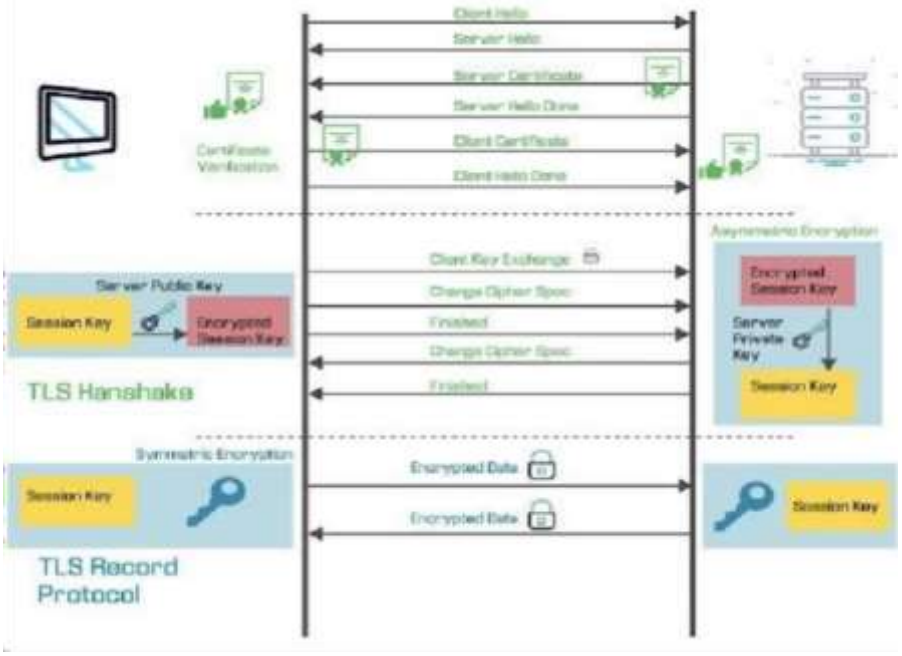
2.7 Zero Trust Security in Cloud-Native Systems

Security in distributed systems must assume that no network segment is inherently secure. Zero Trust architecture operates on the principle of "never trust, always verify."

Figure 2.5: Zero Trust Implementation in Kubernetes







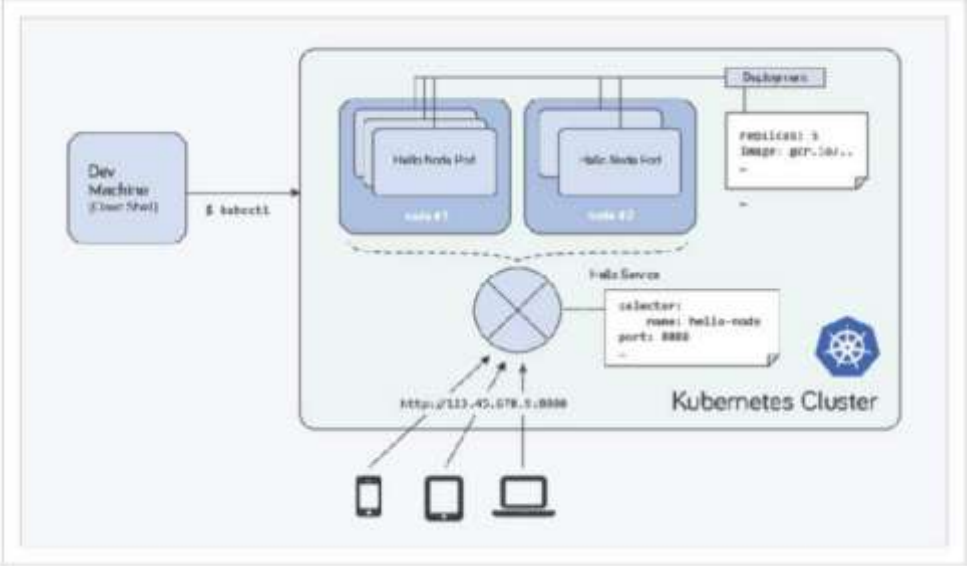
In Kubernetes environments, Zero Trust is implemented through Role-Based Access Control, network policies, and mutual TLS encryption. Even services within the same cluster must authenticate and authorize communication.

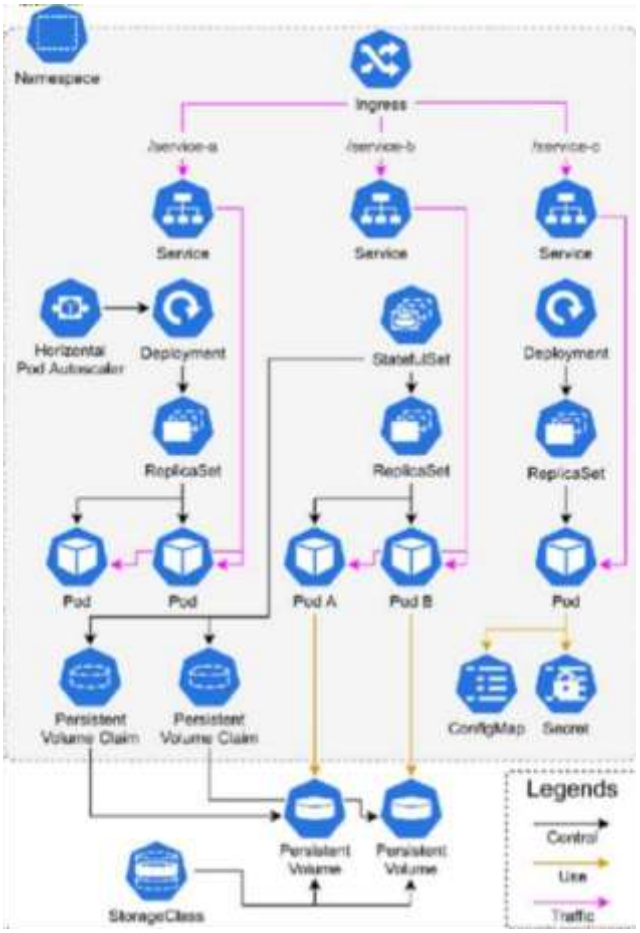
For example, a Payment Service should not be accessible directly by a Delivery Service unless explicitly permitted. By enforcing strict identity verification and encrypted communication, the attack surface is minimized.

2.8 Kubernetes and Amazon EKS

Kubernetes orchestrates containerized applications by managing deployment, scaling, networking, and self-healing.

Figure 2.6: Kubernetes Cluster Architecture

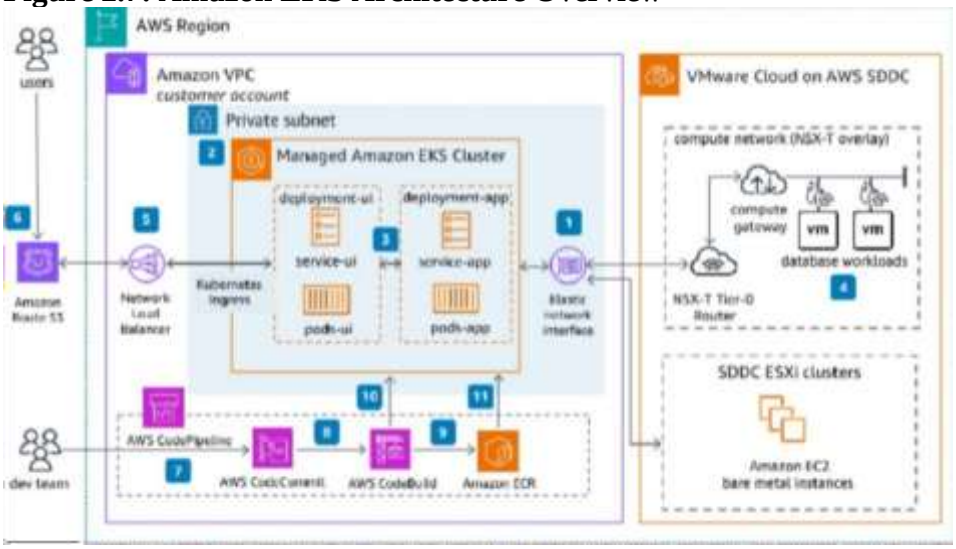


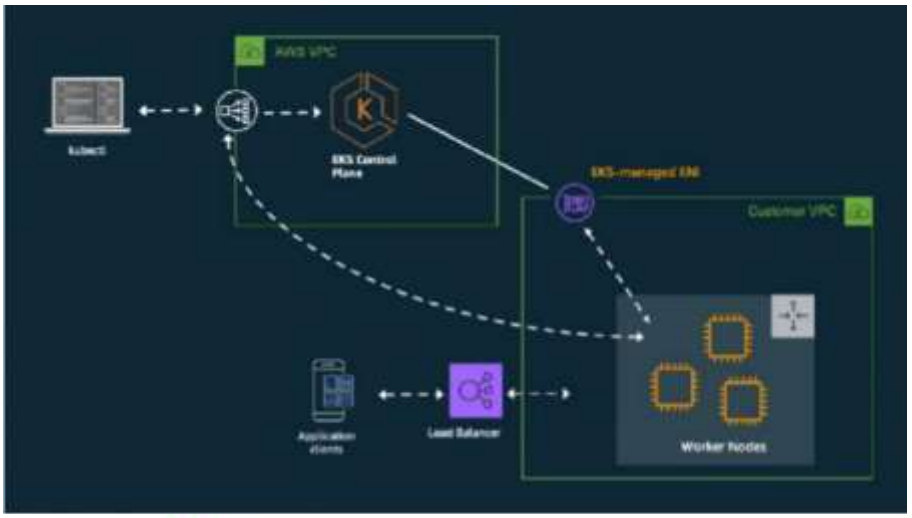


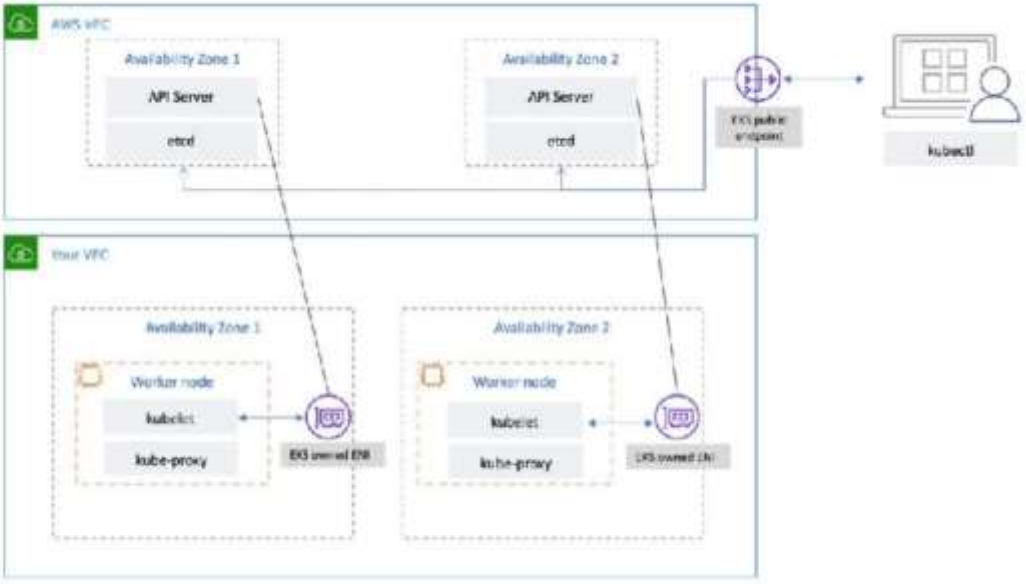
The control plane manages cluster state, while worker nodes run application pods. If a pod crashes, Kubernetes automatically recreates it, enhancing resilience.

Amazon Elastic Kubernetes Service (EKS) provides a managed Kubernetes control plane.

Figure 2.7: Amazon EKS Architecture Overview




AWS VPC
kubectl
EKS Control Plane
EKS-managed ENI
Customer VPC
Application clients
Load Balancer
Worker Nodes


AWS VPC
Availability Zone 1
API Server
etcd
Availability Zone 2
API Server
etcd
kubectl
Your VPC
Availability Zone 1
Availability Zone 2
kubelet
kube-proxy

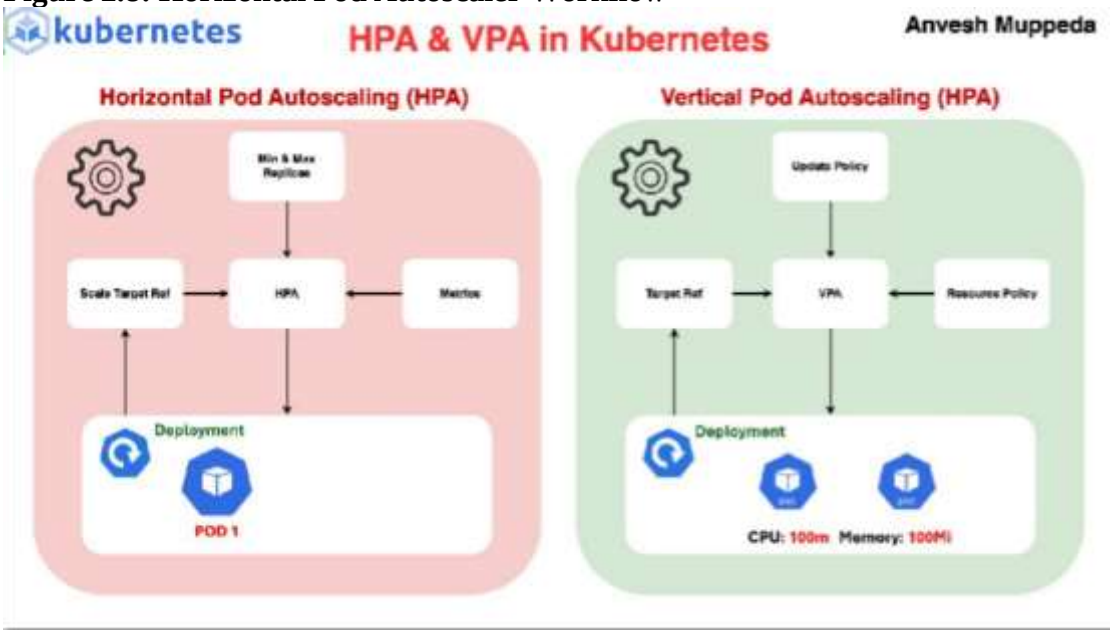

AWS VPC
kubectl
EKS Control Plane
EKS-managed ENI
Customer VPC
Application clients
Load Balancer
Worker Nodes

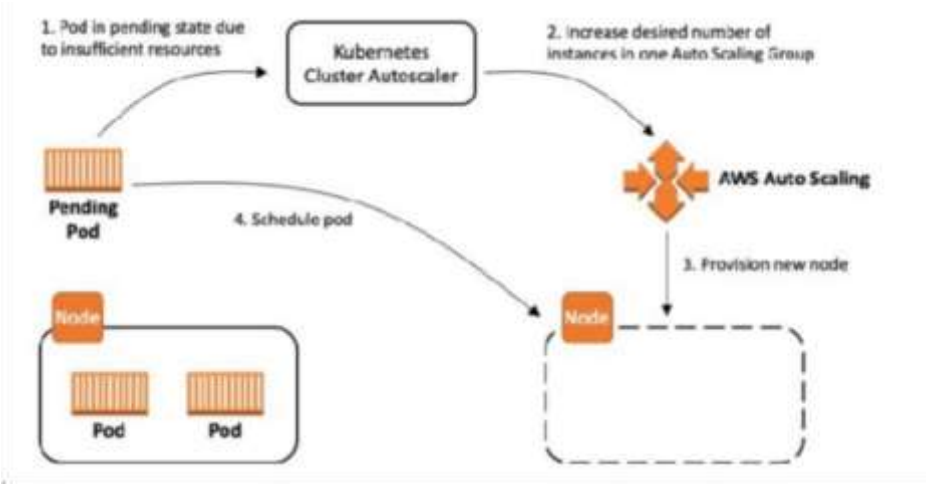
EKS integrates with AWS Identity and Access Management (IAM), supports multi-zone deployment, and simplifies operational management.

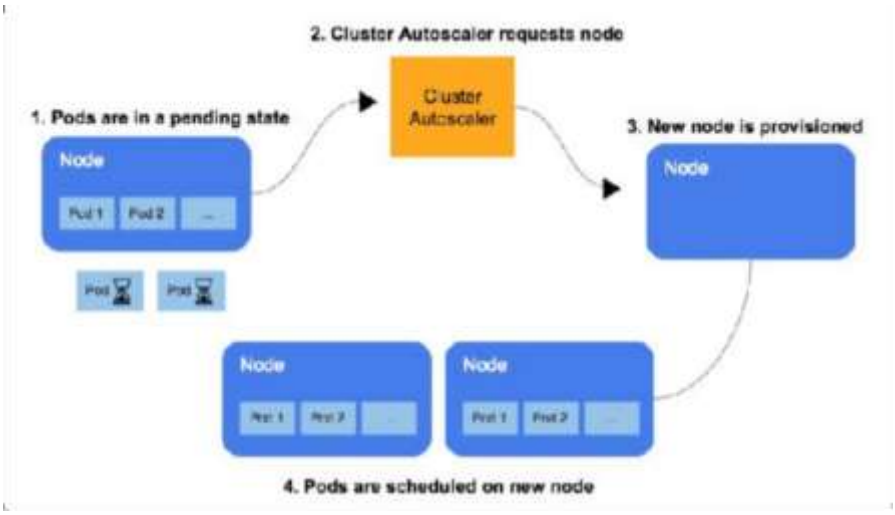
2.9 Automated Scaling and Resource Optimization

Cloud-native systems must dynamically adjust resources based on workload.

Figure 2.8: Horizontal Pod Autoscaler Workflow







The Horizontal Pod Autoscaler increases or decreases the number of pod replicas based on metrics such as CPU usage. The Cluster Autoscaler adjusts the number of worker nodes. Together, these mechanisms ensure performance stability and cost efficiency.

For example, during a festive sale, an e-commerce system may automatically scale from 10 pods to 100 pods within minutes to handle increased traffic.

Chapter 3

ADVANCED CLOUD SECURITY PRACTICES

3.1 Beyond Identity & Access Management (IAM)

Cloud security today is **identity-centric**. IAM controls *who can access what*, but traditional models are not enough for large, dynamic cloud environments. IAM in cloud computing refers to the framework of policies, technologies, and processes that manage and secure digital identities and their access to cloud resources. The primary objective is to ensure that only authorized individuals or systems can access specific resources and that they do so in a secure manner.

Key Components of IAM

1. Authentication

- Authentication is the process of verifying the identity of users, devices, or systems attempting to access cloud resources.

- Multi-factor authentication (MFA) adds an extra layer of security by requiring users to provide multiple forms of identification.

2. Authorization

- Authorization determines the level of access granted to authenticated entities. It involves defining permissions and access controls based on roles, groups, or individual users.

- Role-based access control (RBAC) and attribute-based access control (ABAC) are common authorization models.

3. Identity Lifecycle Management

- IAM involves managing the entire lifecycle of digital identities, from creation to deactivation.

- Provisioning and de-provisioning mechanisms automate the process of granting and revoking access as employees join or leave the organization.

4. Single Sign-On (SSO)

- SSO enables users to log in once and access multiple applications or services without the need to re-authenticate.

- It enhances user experience and reduces the risk associated with password-related vulnerabilities.

5. Audit and Compliance

- IAM systems include audit capabilities to track user activities, changes in permissions, and other relevant events.

- Meeting compliance requirements, such as GDPR or HIPAA, is critical for organizations handling sensitive data.

Benefits of IAM in Cloud Computing

1. Enhanced Security: IAM strengthens security by ensuring that only authorized users have access to specific resources. Multi-layered authentication adds an extra barrier against unauthorized access.

2. Improved Compliance: IAM helps organizations adhere to regulatory requirements by implementing access controls, auditing, and reporting mechanisms.

3. Increased Operational Efficiency: Automated identity management processes streamline onboarding, offboarding, and access requests, reducing administrative overhead.

4. Risk Mitigation: By enforcing least privilege principles and regularly reviewing access permissions, IAM helps mitigate the risk of data breaches and insider threats.

3.1.1 Challenges of Implementing Role-Based Access Control (RBAC)

- Role Explosion (Proliferation): As organizations expand, the number of roles multiplies to meet granular access needs. This quickly becomes overwhelming, defeating RBAC's intended simplicity.
- High Maintenance Overhead: Roles require constant manual updates when employees change positions, get promoted, or leave. This upkeep is time-consuming and error-prone.
- Inflexibility (Role Rigidity): Traditional RBAC is static and struggles to accommodate temporary, context-aware, or dynamic permissions (e.g., location-based or project-specific access).
- Over-Privileged Users: Broad roles often grant more access than necessary, violating the principle of least privilege and creating security gaps.
- Role Conflicts and Ambiguity: Assigning multiple roles to a single user can cause conflicting permissions, while unclear role definitions make it difficult to assign appropriate access.
- Complex Initial Setup: Designing roles requires deep analysis of user responsibilities and access needs. This process is resource-intensive and often demands organizational change.

3.1.2 Attribute-Based Access Control (ABAC)

Attribute-based access control (ABAC) is an authorization paradigm that defines access control policies according to attributes like resource, object, environment, and user attributes. ABAC uses Boolean logic to create access rules containing if-then statements, which define the user, the request, the resource, and the action. For example, if the requester is an accountant, then

allow read-write access to financial data.

ABAC enables organizations to create dynamic, context-aware access control policies using specific attributes according to unique business needs and compliance requirements. Implementation of ABAC was announced as a Priority Objective for implementation by the US Federal Government, and the National Institute of Standards and Technology (NIST) issued a set of standard guidelines that define how to implement ABAC in an enterprise environment.

ABAC systems make access decisions by:

1. Intelligently studying how attributes interact in an environment.
2. Creating rules that determine access for a set of attributes if specific conditions are met.

Here are the four main categories of ABAC security attributes:

Subject/user attributes	Attributes that indicate an individual is attempting to gain access. For example, username, age, ID, job title, organization, job role, department, and security clearance.
Resources/object attributes	Attributes that indicate the resource requested for access.
Action	Attributes that indicate the action the user wants to perform with a resource. For example, view, transfer, read, or delete.
Environmental attributes	Attributes that contextualize the access attempt. For example, time, device, and location.

ABAC systems establish policies that define what combination of various attributes is required to perform a specific action with a certain object or resource. The system uses these policies to grant or deny access.

Here is how the process typically works:

1. An access request is triggered
2. The ABAC tool scans attributes to determine if they match existing policies.
3. If the attributes match a policy, the system grants access to the user.

Related content: Read our guide to ABAC security

RBAC vs ABAC

RBAC and ABAC are access management methods. RBAC grants access to roles, whereas ABAC uses attributes-based policies to grant or deny access.

What is RBAC?

RBAC was formalized by the NIST in 1992 and quickly became the standard for SMBs with over 500 employees. It was implemented in user provisioning systems to streamline the joiners, movers, and leavers (JML) human resources (HR) process. It enables organizations to manage access control by roles instead of the individual user ID or each employee. It involves grouping users and entitlements according to business functions or activities, called roles.

Organizations can now use RBAC to create flat or hierarchical roles and also include inheritance. The key benefit is using roles as an extra level of abstraction. Roles act as a set of entitlements or permissions. It significantly simplifies access management, enabling organizations to assign one role to hundreds of users, a big advantage while scaling up fast.

How ABAC differs from RBAC

ABAC enables organizations to extend existing roles via attributes and policies. It offers the context needed to make intelligent authorization decisions. Instead of granting access only by roles, ABAC systems account for the role, the relevant actions and resources for the job, the location, time, and how the request is made.

ABAC policies are based on individual attributes, consist of natural language, and include context. It eliminates the need for hundreds of overloaded roles, enabling administrators to add, remove, and reorganize attributes without rewriting the policy. It requires significantly fewer roles. As a result, it offers simpler identity management.

Example Policy:

Allow access if Project = "CloudLab" Environment = "Prod" Clearance = "High"

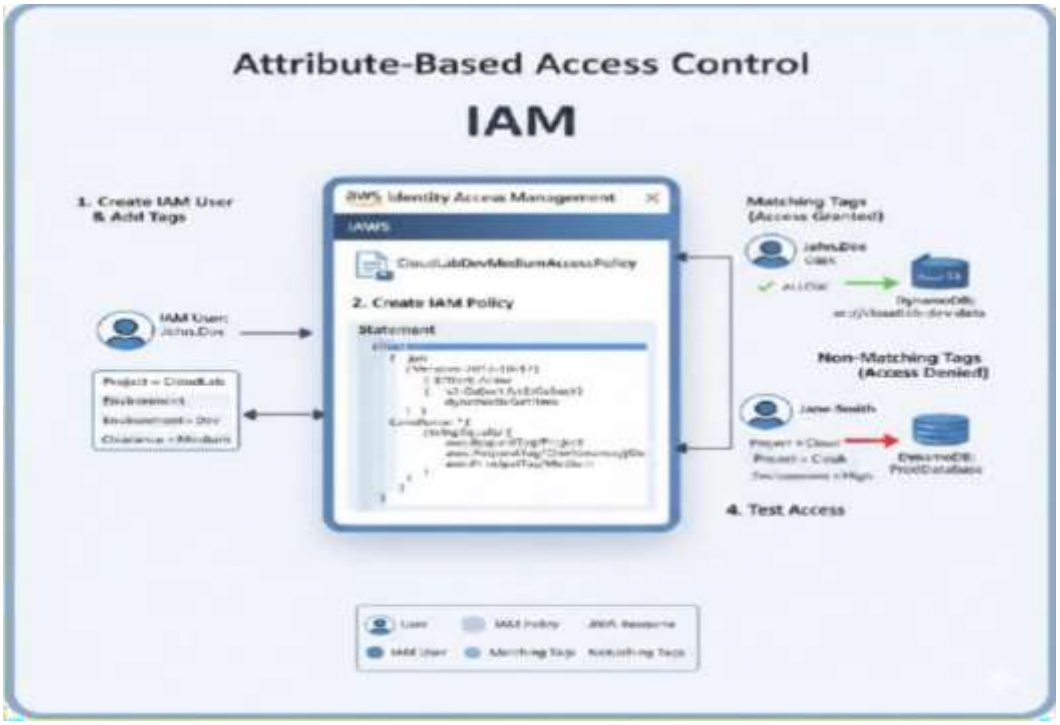
Hands-On Practice 1: Implementing ABAC in AWS

Objective: Configure ABAC using IAM tags.

Steps:

1. Create an IAM user.
2. Add tags:
    - Project = CloudLab
    - Environment = Dev
    - Clearance = Medium

3. Create an IAM policy using conditions:
    - aws:PrincipalTag
4. Attach policy to the user.
5. Test access with matching and non-matching tags.



3.2 The Incident Response Lifecycle

The incident response life cycle is a continuous, four-phase framework—Preparation, Detection & Analysis, Containment/Eradication/Recovery, and Post-Incident Activity— designed to manage cybersecurity incidents. It enables organizations to proactively prepare for, efficiently handle, and learn from security breaches to improve future resilience.

Key Phases of the NIST Incident Response Life Cycle:

- **1. Preparation:** Involves establishing policies, training personnel, and implementing tools to prevent or handle security incidents before they occur.
- **2. Detection and Analysis:** Focuses on identifying, monitoring, and validating security incidents through alerts from intrusion detection systems, logs, and other security measures.
- **3. Containment, Eradication, and Recovery:** Aims to limit the impact of the incident, remove the cause (e.g., malware, unauthorized access), and restore systems to normal operation.
- **4. Post-Incident Activity (Lessons Learned):** Involves analyzing the incident to

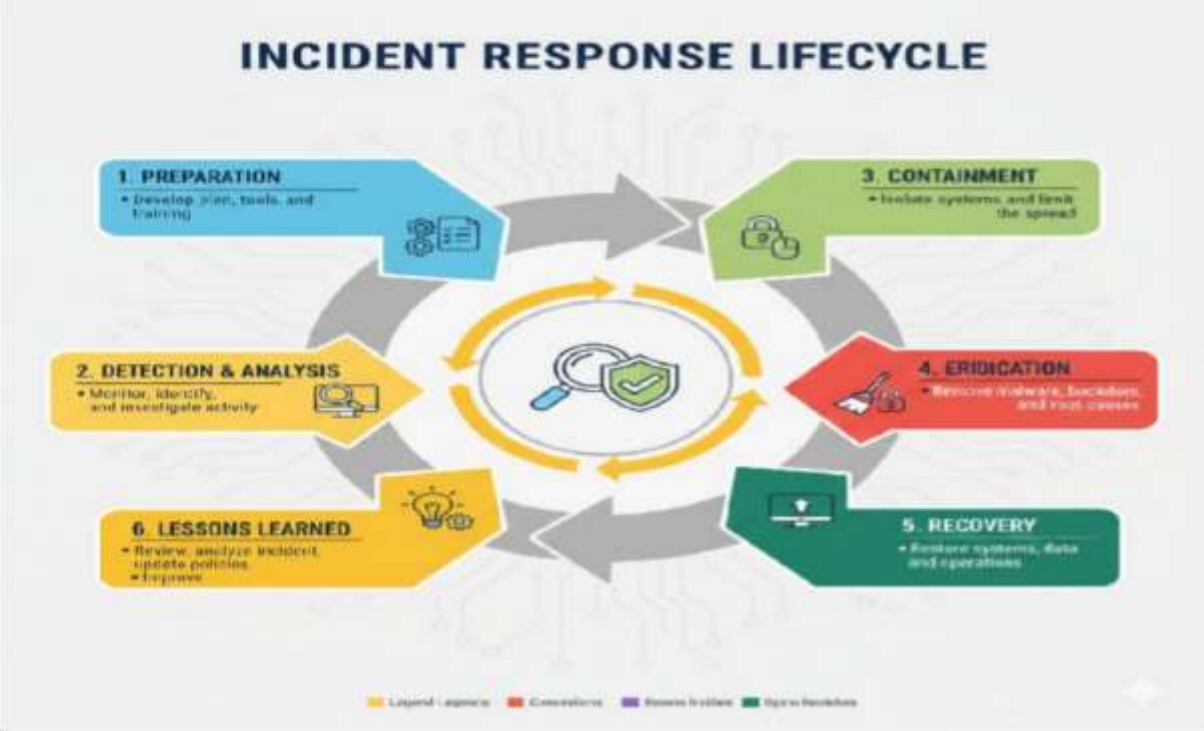
improve future response, documentation, and security measures.

Alternative Frameworks:

- **SANS Institute:** Often cited as a 6-step process: Preparation, Identification, Containment, Eradication, Recovery, and Lessons Learned.
- **Other Variations:** Some models further break down phases into 7 steps, adding specific actions like "re-testing".

The process is cyclical, meaning insights from the post-incident phase directly inform better preparation for future threats.

3.2.1 Incident Response Lifecycle Phases



Key Phases of the NIST Incident Response Life Cycle:

- **1. Preparation:** Involves establishing policies, training personnel, and implementing tools to prevent or handle security incidents before they occur.
- **2. Detection and Analysis:** Focuses on identifying, monitoring, and validating security incidents through alerts from intrusion detection systems, logs, and other security measures.
- **3. Containment, Eradication, and Recovery:** Aims to limit the impact of the incident, remove the cause (e.g., malware, unauthorized access), and restore systems to normal operation.
- **4. Post-Incident Activity (Lessons Learned):** Involves analysing the incident to improve future response, documentation, and security measures.

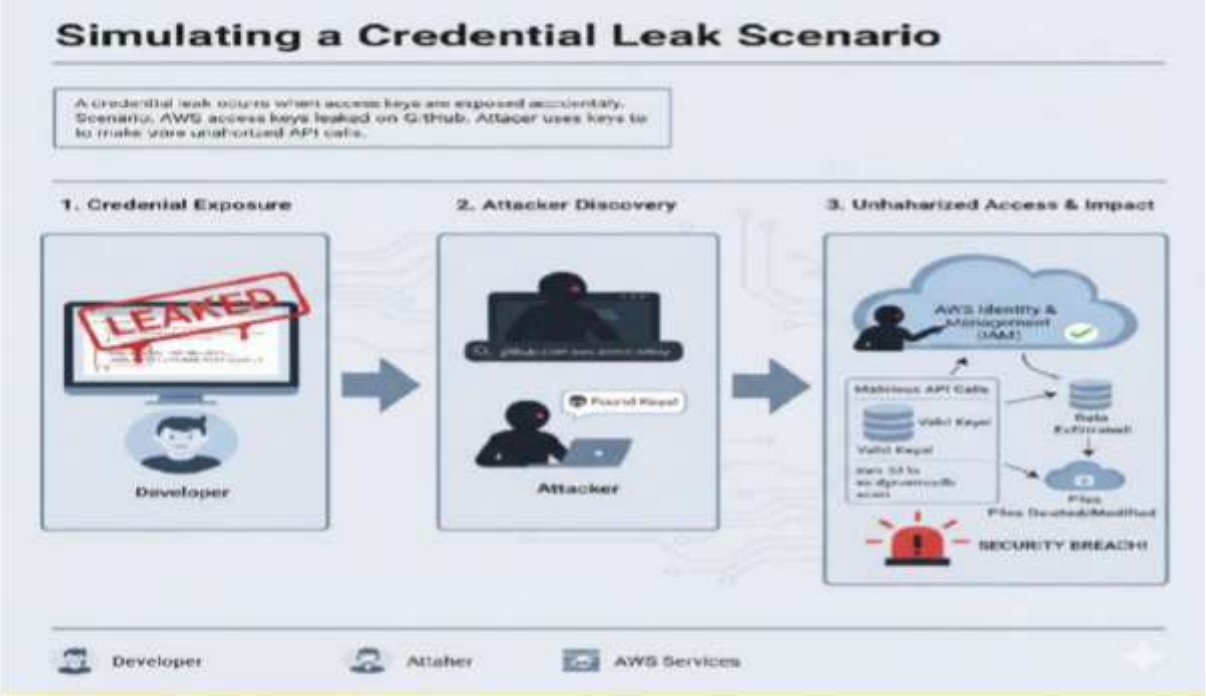
Alternative Frameworks:

- **SANS Institute:** Often cited as a 6-step process: Preparation, Identification, Containment, Eradication, Recovery, and Lessons Learned.
- **Other Variations:** Some models further break down phases into 7 steps, adding specific actions like "re-testing".

The process is cyclical, meaning insights from the post-incident phase directly inform better preparation for future threats.

3.2.2 Lab Exercise: Simulating a Credential Leak Scenario

A **credential leak** occurs when access keys are exposed accidentally.



Here is the detailed breakdown of the scenario shown in the diagram:

1. Credential Exposure (The Root Cause)

The incident almost always begins with a human error during the Preparation or Development phase.

- The Action: A developer hardcodes an access_key_id and secret_access_key into a script for testing.
- The Mistake: They forget to add the file to .gitignore and perform a git push to a public repository.
- The Result: Within seconds, the credentials are live on the internet, accessible to anyone—and anything.

2. Attacker Discovery (The "Bot" Race)

Attackers don't sit around refreshing GitHub pages manually. They use automated scrapers that scan every single commit on GitHub in real-time.

- Scanning: Bots look for specific regex patterns that match AWS key structures (e.g., keys starting with AKIA).
- Validation: Once a key is found, the bot instantly attempts a simple API call (like aws sts get-caller-identity) to see if the key is valid and what permissions it has.

3. Unauthorized Access & Impact

Once the attacker confirms the keys work, they move into the Exploitation phase. Depending on the permissions attached to that user, the impact can be devastating:

- Data Exfiltration: If the keys have S3 permissions, the attacker can download sensitive company data or customer PII (Personally Identifiable Information).
- Resource Hijacking: Attackers often spin up dozens of high-powered EC2 instances for crypto-mining, leaving you with a massive bill.
- Persistence: They might create a new IAM user or add their own SSH keys to your servers so they can get back in even if you delete the original leaked keys.
- Destruction: In "ransomware" scenarios, they may delete your databases (RDS) or backups to force a payment.

How to Respond (Using the IR Lifecycle)

If this happens to you, follow the steps from the Incident Response diagram we discussed:

1. Detection: AWS GuardDuty or GitHub Secret Scanning sends an alert.
2. Containment: Immediately deactivate the access key in the IAM console. Do not just delete the code on GitHub; the history still contains the key!
3. Eradication: Rotate all compromised credentials and check for any "backdoor" accounts the attacker might have created.
4. Recovery: Restore any deleted data from backups and monitor logs (CloudTrail) to ensure the attacker is gone.
5. Lessons Learned: Implement IAM Roles instead of static keys and set up a pre-commit hook to prevent keys from being pushed in the future.

Hands-On Practice 2: Credential Leak Simulation

Objective: Simulate and detect credential misuse.

Steps:

1. Setup: Creating the Target

First, you create an **IAM User** specifically for this lab.

- **Programmatic Access:** This is key. It provides an Access Key ID and Secret Access Key. These are the "strings" that usually get leaked in code.
- **Permissions:** You might give this user ReadOnlyAccess or a custom policy. In a real leak, the damage is limited by whatever permissions this specific user holds.

When you generate the keys, you are creating long-term credentials. Unlike a password that might require MFA, these keys can be used by any machine (the AWS CLI, a script, or a bot) anywhere in the world until they are manually deactivated.

2. Generation: The "Secret" Data

1. When you generate the keys, you are creating long-term credentials. Unlike a password that might require MFA, these keys can be used by any machine (the AWS CLI, a script, or a bot) anywhere in the world until they are manually deactivated.

3. The Attack: Remote Execution

To make the simulation realistic, you use the AWS CLI from a different IP or location (like a personal laptop, a different VPC, or even a CloudShell instance in a different region).

- Command: aws configure — This is where you input the "leaked" keys.
- Context: To AWS, this looks like a valid login, but the metadata (IP address, User Agent) has now changed.

4. Suspicious Actions: Discovery (Reconnaissance)

The attacker doesn't know what they have access to yet. They start running "discovery" commands:

- aws s3 ls: Attempting to see all data storage.
- aws iam list-users: Attempting to see the account structure.
- Why it's suspicious: If a developer in New York suddenly starts listing all S3 buckets from an IP address in a different country at 3:00 AM, it triggers an anomaly.

5. Detection: Observing CloudTrail

AWS CloudTrail is the "security camera" of your AWS account. It records every API call made. When you check the logs, you will see a JSON entry containing:

- who: The IAM user name you created.
- what: The ListBuckets event.
- where: The source IP address of your "attacker" machine.
- when: The exact timestamp of the action.

3.2.3 Digital Forensics in Cloud

Cloud digital forensics focuses on **log analysis and snapshot-based evidence**.

Forensic Evidence Sources

1. VPC Flow Logs (Network Traffic)

VPC Flow Logs are your "network wiretap." They capture information about the IP traffic going to and from network interfaces in your VPC.

- **What they show:** Source IP, Destination IP, Port numbers, Protocol, and whether the traffic was **Accepted** or **Rejected** by Security Groups/ACLs.
- **Forensic Value:** Essential for detecting **Data Exfiltration** (unusual outbound data spikes) or **Lateral Movement** (an attacker jumping from one server to another).

2. CloudTrail Logs (API Activity)

If VPC Flow Logs tell you *where* the data went, CloudTrail tells you *who* moved it. It records every action taken by a user, role, or AWS service.

- **What they show:** The Identity (User/Role), the Source IP, the Time, and the specific API call (e.g., StopInstances, CreateUser, DeleteBucket).
- **Forensic Value:** This is the **Audit Trail**. It proves how an attacker gained entry and what configuration changes they made to stay hidden.

3. EBS Snapshots (Disk Evidence)

EBS Snapshots are point-in-time "photos" of your server's hard drive. In forensics, this is known as **Dead Imaging**.

- **What they show:** The entire file system, including hidden files, deleted files (if not overwritten), system logs, and malware binaries.
- **Forensic Value:** If an EC2 instance is compromised, you take a snapshot immediately. You can then attach that snapshot to a "Forensic Workstation" to analyze the disk without alerting the attacker or accidentally changing the evidence.

Digital Forensics Diagram

Hands-On Practice 3: Cloud Forensics Investigation

Steps:

1. **Enable VPC Flow Logs**

- **Purpose:** VPC Flow Logs capture information about IP traffic going to and from network interfaces in your Virtual Private Cloud (VPC).

- **Steps:**
    - Go to the **Amazon VPC console**.
    - Select your VPC → choose **Flow Logs → Create Flow Log**.
    - Specify:
        - **Filter:** All traffic (to capture accepted and rejected connections).
        - **Destination:** CloudWatch Logs or S3 bucket.
    - Enable logging at the VPC, subnet, or network interface level.
- **Why it matters:** Helps identify suspicious traffic patterns, such as repeated failed connections or unexpected inbound traffic from unknown IPs.

2. Enable CloudTrail Logging

- **Purpose:** AWS CloudTrail records API calls and activities across your AWS environment.
- **Steps:**
    - Go to the **CloudTrail console**.
    - Create a new trail or enable logging on an existing one.
    - Send logs to an S3 bucket for long-term storage and analysis.
    - Optionally, integrate with CloudWatch for real-time monitoring.
- **Why it matters:** Provides visibility into actions taken by users, roles, or services. You can detect unauthorized API calls, privilege escalation, or unusual resource creation/deletion.

3. Create Snapshot of Compromised EC2 EBS Volume

- **Purpose:** Preserves evidence by creating a point-in-time copy of the compromised disk.
- **Steps:**
    - Go to the **EC2 console**.
    - Select the compromised instance → identify its EBS volume.
    - Choose **Create Snapshot**.
    - Store the snapshot securely for forensic analysis.

4. Analyze Logs

Now comes the investigative part. You'll use the logs to answer key forensic questions:

a. Source IP

- Check **VPC Flow Logs** for inbound connections.
- Identify unusual or unauthorized IP addresses.
- Correlate with known malicious IP databases if needed.

b. Time of Attack

- Look at timestamps in **CloudTrail logs** and **Flow Logs**.
- Pinpoint when suspicious activity began (e.g., login attempts, API calls).
- Helps establish a timeline of events.

c. Unauthorized Actions

- **Review CloudTrail logs for:**
  - Failed login attempts.
  - Unauthorized API calls (e.g., DeleteSecurityGroup, StopInstances).
  - Privilege escalation attempts (e.g., attaching policies to IAM roles).
- Compare against normal baseline activity to highlight anomalies**.**

5. Putting It All Together:-

- **Enable logging first** (Flow Logs + CloudTrail) → ensures visibility.
- **Preserve evidence** (snapshot of EBS volume) → prevents tampering.
- **Analyze systematically**:
  - Identify attacker's IP.
  - Establish timeline.
  - Document unauthorized actions.

3.3 Automating Threat Detection

Manual monitoring is not practical in cloud. Automation improves **speed, accuracy, and response time**.

3.3.1 AWS GuardDuty

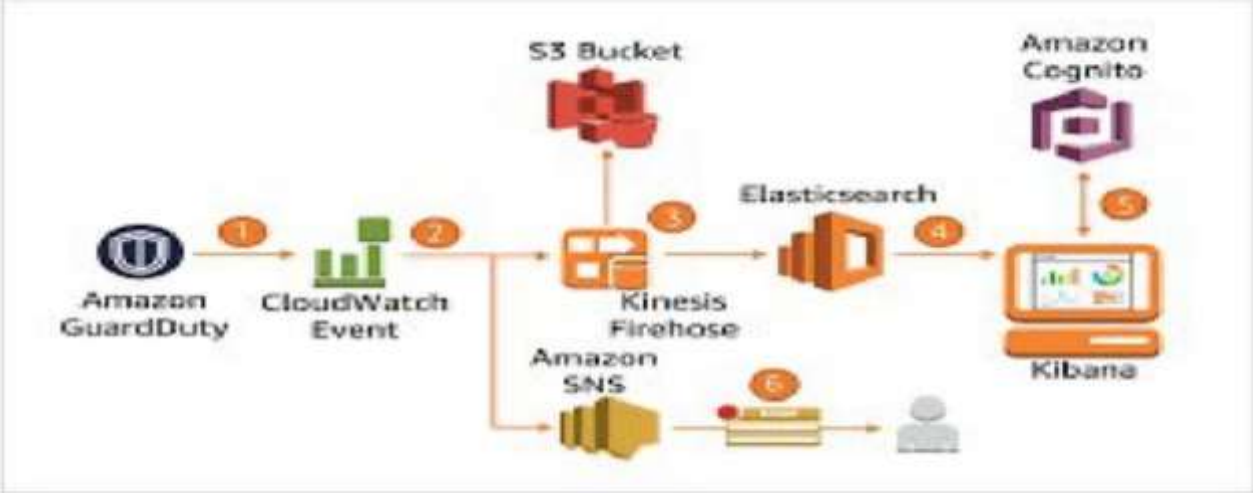
AWS GuardDuty is a managed threat detection service using **Machine Learning (ML)**.

Detects:

- Crypto-mining activity

- Compromised credentials
- Unauthorized API calls
- Malware communication

GuardDuty Diagram



Practice 4: Enabling AWS GuardDuty Steps:

1. Open AWS Console.
2. Enable GuardDuty.
3. Enable all data sources.
4. Generate sample findings.
5. Analyze severity and attack type.

3.3.2 AWS Security Hub

Security Hub centralizes alerts from multiple AWS security services.

Features:

- Single security dashboard
- Compliance checks (CIS, PCI)
- Aggregates GuardDuty findings

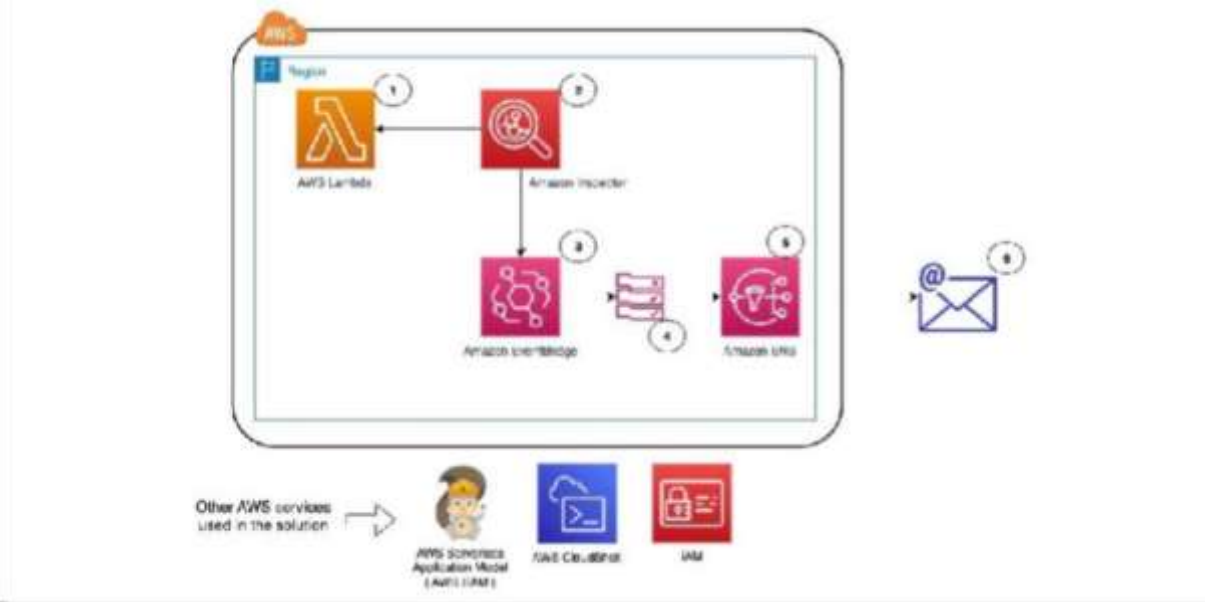
3.3.3 Automated Remediation using AWS Lambda

When a threat is detected:

- Lambda function executes automatically

- Takes corrective action

Automation Diagram



1. Threat Detection (GuardDuty) Amazon GuardDuty continuously scans your AWS environment for signs of malicious or suspicious activity. When it identifies a potential issue, it generates a detailed security finding.

2. Centralized Management (Security Hub) These findings, along with those from other integrated services such as Amazon Inspector, are automatically ingested into AWS Security Hub. Security Hub standardizes them using the AWS Security Finding Format (ASFF), prioritizes them, and makes them available for further action.

3. Event Routing (EventBridge) Security Hub publishes new and updated findings to Amazon EventBridge in near real time. EventBridge rules can be configured to filter findings based on specific criteria—such as severity level, resource type, or finding title—and forward only the relevant ones to a chosen target.

4. Remediation Trigger (AWS Lambda) When a rule matches, EventBridge invokes a designated AWS Lambda function. This function contains the remediation logic, written in Python (using Boto3) or Node.js, and runs under an IAM role with tightly scoped permissions.

5. Automated Response (Corrective Actions) The Lambda function executes the corrective steps defined for the finding. Examples include:

- **Isolating EC2 instances** by moving them into a quarantine security group with no inbound or outbound traffic.
- **Disabling IAM keys** by revoking compromised credentials or restricting the affected principal's permissions.

6. Implementation Steps

- Enable GuardDuty and Security Hub in your AWS account.
- Create a remediation Lambda function with the necessary IAM role and permissions.
- Configure an EventBridge rule to capture the findings you want to remediate automatically.
- Set the Lambda function as the target for that rule.
- Test the setup in a sandbox environment using sample findings or intentionally non-compliant resources.
- Monitor execution logs in CloudWatch to confirm that remediation actions are being applied correctly.

Hands-On Practice 5: Automating Remediation Steps:

1. Enable Security Hub.
2. Integrate GuardDuty findings.
3. Create EventBridge rule.
4. Write Lambda function to:
    - Disable IAM access key
    - Stop EC2 instance
5. Test automated response.

Chapter 4

Data Engineering & Intelligence (The DOMO Integration)

4.1.1 Meaning of Data Engineering Data Engineering

Data engineering is the practice of designing, building, and managing systems that collect, store, and transform raw data into usable information for analysis and decision-making. It serves as the backbone of modern data infrastructure.

Data Engineering is the process of collecting, cleaning, transforming, and storing data so that it can be used for analysis and decision-making. Data Engineers prepare data so that Data Scientists and Analysts can use it easily.

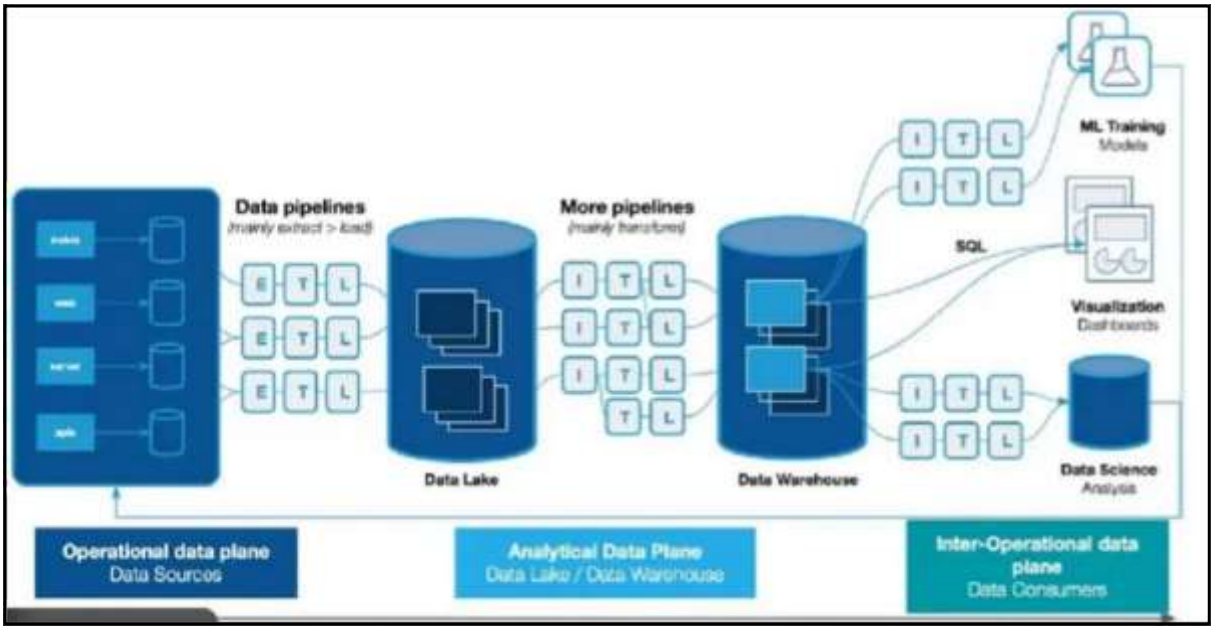
Data Engineering is the process of designing and managing systems that collect, clean, transform, and store data for analysis.

Why Data Engineering is Important?

- Raw data is messy and unorganized.
- Companies collect data from websites, apps, sensors, etc.
- This data must be cleaned and stored properly.
- Without data engineering, analysis is not possible.

• 4.1.2 Importance in Cloud Environment

In a cloud environment, Data Engineering is important because it helps to manage large amounts of data efficiently, securely, and cost-effectively using cloud services.



Key Considerations:

- **Scalability:** The ability to handle increasing amounts of data and workloads efficiently.
- **Integration:** How well the platform integrates with other services and tools.
- **Performance:** The speed and efficiency of data processing and querying.
- **Cost:** Pricing models and cost management features.
- **Ease of Use:** User interface, documentation, and support

• 4.1.3 Role in Business Intelligence

Business Intelligence (BI) means using data to:

- Analyze business performance
- Create reports
- Make smart decisions

Role of Data Engineering in BI

1. Data Collection

Collects data from:

- Sales systems
- Websites
- Customer databases

2. Data Cleaning

Removes:

- Duplicate records
- Missing values
- Incorrect data

3. Data Transformation

- Converts raw data into structured format
- Organizes data for reporting

4. Data Storage

- Stores data in Data Warehouse
- Makes data easily available for BI tools

5. **Supports Reporting & Dashboards**

- Provides accurate data
- Helps managers analyze trends

4.1 Data Fabric Architecture

• 4.2.1 Definition of Data Fabric

Data Fabric is an architecture that **connects and manages data from different sources** (cloud, on-premise, databases, applications) in a unified way. Data Fabric acts like a **bridge** that connects all data and makes it easily available for users.

Data Fabric is a modern data architecture that provides unified access, integration, and management of data across multiple environments such as cloud and on-premise systems.

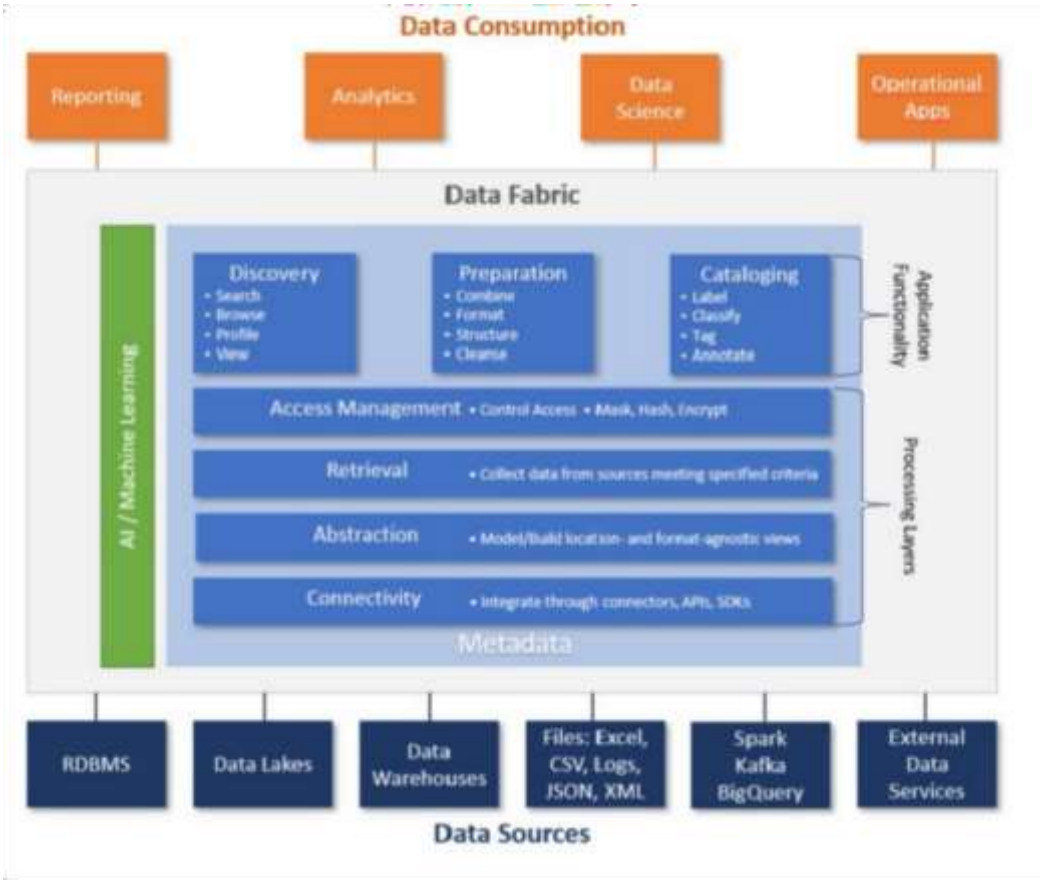


fig: Architecture of data fabric

"A data fabric utilizes continuous analytics over existing, discoverable and inferenced metadata assets to support the design, deployment and utilization of integrated and reusable

data across all environments, including hybrid and multi-cloud plaVorms."

- **4.2.2 Components of Data Fabric**

A data fabric consists of multiple components. These components can be combined in different ways depending on business requirements. Hence, the structure of a data fabric may vary from one organization to another. The major components are explained below:

Key components of a data fabric

- Augmented Data Catalog
- Persistance Layer
- Knowledge Graph
- Insights and Recommendations Engine
- Data Preparation and Data Delivery Layer
- Orchestration and Data Ops

1) Augmented Data Catalog

An augmented data catalog provides access to different types of metadata in a connected manner.
It presents metadata in a graphical and easy-to-understand format.

- It builds relationships between data elements.
- It uses machine learning techniques to connect data assets with business terms.
- It acts as the business semantic layer in the data fabric.

This component helps users easily discover and understand available data.

2) Persistence Layer

The persistence layer is responsible for storing data.
It can store structured and unstructured data based on business needs.

- Supports relational databases (tables).
- Supports non-relational models (NoSQL, documents, etc.).
- Storage type depends on the use case.

This layer ensures flexible and dynamic data storage.

3) Active Metadata

Active metadata is a key feature of a data fabric.

- It collects and analyzes different types of metadata.
- Includes usage-based metadata (how users and systems use data).
- Goes beyond passive metadata (design-time and run-time metadata). It

helps improve data management by continuously monitoring data usage.

4) Knowledge Graph

A knowledge graph represents data in a connected form.

- Uses uniform identifiers.
- Supports flexible schemas.
- Shows relationships between data entities visually.

It makes the data environment searchable and easier to understand.

5) Insights and Recommendations Engine

This component helps in building reliable data pipelines.

- Supports operational use cases.
- Supports analytical use cases.
- Suggests improvements for better performance.

It improves decision-making and data processing efficiency.

6) Data Preparation and Data Delivery Layer

This layer is responsible for moving data from source to target.

- Retrieves data from different sources.
- Delivers data using methods such as:
    - ETL (bulk processing)
    - Messaging systems

- Change Data Capture (CDC)
- Data virtualization
- APIs

It ensures smooth data flow across systems.

7) Orchestration and DataOps

This component manages and coordinates the complete data workflow.

- Controls when pipelines should run.
- Decides how frequently processes execute.
- Monitors and manages generated data.

It ensures smooth end-to-end data operations

• 4.2.3 Benefits of Data Fabric

Data Fabric provides a **unified and intelligent way to manage data** across different systems like cloud, on-premise, and hybrid environments.

1. Provides unified and centralized data access.
2. Improves data integration across multiple environments.
3. Enhances data quality and consistency.
4. Supports real-time and batch processing.
5. Ensures strong security and governance.
6. Offers scalability and cost efficiency.

4.2 Modern Data Lake

• 4.3.1 What is a Data Lake?

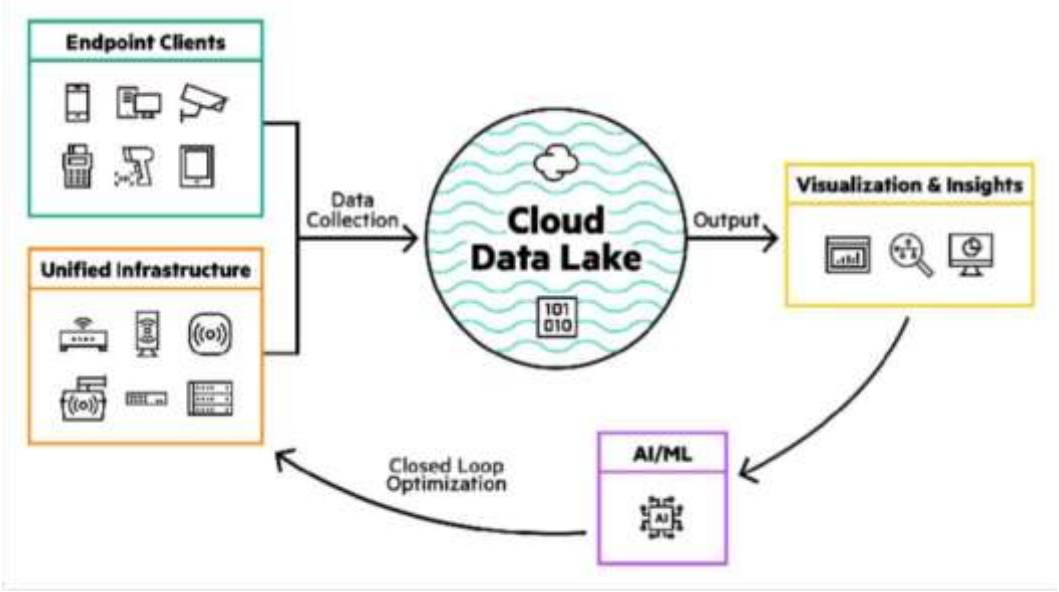
It is a place where all types of data (structured, semi-structured, unstructured) are stored together

A Data Lake is a centralized repository that stores large volumes of structured and unstructured data in its raw format for future processing and analysis.

A **Data Lake** is a storage system that stores **large amounts of raw data** in its original format.

A data lake is a centralized repository that ingests, stores, and allows for processing of large volumes of data in its original form. It can accommodate all types of data, which is then

used to power big data analytics, machine learning, and other forms of intelligent action.



The given diagram represents a **Modern Cloud Data Lake Architecture**. It explains how data is collected from different sources, stored in the cloud, processed, analyzed, and used for decision-making. The complete system works in a continuous improvement cycle.

1) Endpoint Clients

Endpoint clients are the devices and applications that generate data.
Examples include mobile phones, laptops, sensors, CCTV cameras, and point-of-sale systems. These devices continuously produce large volumes of data in different formats.

2) Unified Infrastructure

Unified infrastructure refers to the supporting systems such as servers, storage devices, and network components. It helps in collecting and transferring data from endpoint devices to the cloud environment. This infrastructure ensures smooth communication and data flow.

3) Data Collection

In this stage, data from endpoint clients and infrastructure is gathered and transmitted to the cloud. Data collection may be done using APIs, streaming methods, or ETL processes. The collected data can be structured, semi-structured, or unstructured.

4) Cloud Data Lake

The cloud data lake acts as the central storage system. It stores raw data in its original format.
Key features include:

- Storage of large volumes of data
- Support for all types of data

- Scalability and flexibility
- Cost-effective cloud storage

The data lake serves as a foundation for analytics and advanced processing.

5) Visualization and Insights

After processing the data, it is used for generating reports, dashboards, and analytical insights. Business intelligence tools convert raw data into meaningful information. Managers and decision-makers use these insights to improve business performance.

6) Artificial Intelligence and Machine Learning (AI/ML)

AI and ML models analyze the stored data to identify patterns and trends. These models help in making predictions and automating decisions. Examples include demand forecasting, fraud detection, and customer behavior analysis.

7) Closed Loop Optimization

Closed loop optimization refers to the continuous feedback process. Insights generated from analytics and AI are used to improve system performance. The improved data and results are again stored in the data lake, forming a continuous cycle of enhancement.

• 4.3.2 Difference Between Data Lake and Data Warehouse

Parameter	Data Lake	Data Warehouse
Definition	Storage system that stores raw data in original format	Storage system that stores processed and structured data
Data Type	Structured, Semi-structured, Unstructured	Only Structured data
Data Format	Raw data	Cleaned and processed data
Schema	Schema-on-read (applied after storing data)	Schema-on-write (applied before storing data)
Processing	Data processed when needed	Data processed before storage
Users	Data Scientists, Data Engineers	Business Analysts, Managers
Purpose	Big data analytics, AI/ML	Reporting and Business Intelligence

Storage Cost	Low (uses cheap storage)	Higher (optimized structured storage)
Flexibility	Highly flexible	Less flexible
Performance	Slower for structured queries	Faster for structured queries
Examples of Data	Logs, videos, images, sensor data	Sales reports, financial records

• 4.3.3 Need for Modern Data Lake

Modern data lakes are essential for handling the massive volume, velocity, and variety of data that traditional systems cannot manage, enabling AI/ML workloads, and providing cost-effective, flexible storage. They allow for real-time, unstructured data ingestion, breaking down silos for faster, advanced analytics

1) Handling Large Volume of Data

Organizations generate huge amounts of data from websites, mobile apps, IoT devices, and social media. Modern data lakes can store this large volume of data efficiently.

2) Managing High Velocity of Data

Data is generated continuously and at high speed. Modern data lakes support real-time and streaming data ingestion, allowing immediate storage and processing.

3) Supporting Variety of Data

Data comes in different formats such as structured, semi-structured, and unstructured (images, videos, logs). Modern data lakes can store all types of data in one place.

4) Enabling AI and Machine Learning

AI and ML models require large amounts of raw data for training and prediction. Modern data lakes provide flexible storage suitable for advanced analytics and intelligent systems.

5) Cost-Effective Storage

Modern data lakes use cloud-based storage, which is more economical compared to traditional systems. Organizations can scale storage as per their needs.

6) Breaking Data Silos

Data silos occur when data is stored separately in different departments. Modern data lakes integrate data from multiple sources, providing a unified view.

7) Faster and Advanced Analytics

With centralized and organized storage, organizations can perform faster analytics and generate insights quickly for decision-making.

4.3 Building Data Lake Using AWS

• 4.4.1 Introduction to Amazon Web Services (AWS)

Amazon Web Services (AWS) is a cloud computing platform provided by **Amazon Web Services**. It offers various cloud services such as computing, storage, databases, networking, and analytics.

AWS allows organizations to:

- Store large amounts of data
- Process data efficiently
- Build scalable cloud-based applications
- Pay only for the services used

AWS is widely used for building modern data lakes because it provides reliable and cost-effective cloud infrastructure.

• 4.4.2 Amazon S3

Amazon Simple Storage Service (S3) is a highly durable, scalable, and secure object storage service provided by AWS. It enables users to store and retrieve any amount of data—such as images, backups, and data lakes—via the internet. Data is organized into buckets and objects, offering features like lifecycle policies, encryption, and granular access controls

•Provides secure object storage. •Highly scalable and durable. •Used for backup, websites, big data, etc. •Supports versioning and encryption

Features of Amazon S3:

- Stores unlimited data
- Highly durable and secure
- Supports structured and unstructured data
- Cost-effective storage

In a data lake architecture, Amazon S3 acts as the **central storage layer** where raw data is stored in its original format.

• 4.4.3 AWS Glue

AWS Glue is a serverless data integration service that makes it easy for analytics users to discover, prepare, move, and integrate data from multiple sources. You can use it for analytics, machine learning, and application development. It also includes additional productivity and data ops tooling for authoring, running jobs, and implementing business workflows.

With AWS Glue, you can discover and connect to more than 70 diverse data sources and manage your data in a centralized data catalog. You can visually create, run, and monitor extract, transform, and load (ETL) pipelines to load data into your data lakes. Also, you can immediately search and query cataloged data using Amazon Athena, Amazon EMR, and Amazon Redshift Spectrum.

AWS Glue consolidates major data integration capabilities into a single service. These include data discovery, modern ETL, cleansing, transforming, and centralized cataloging. It's also

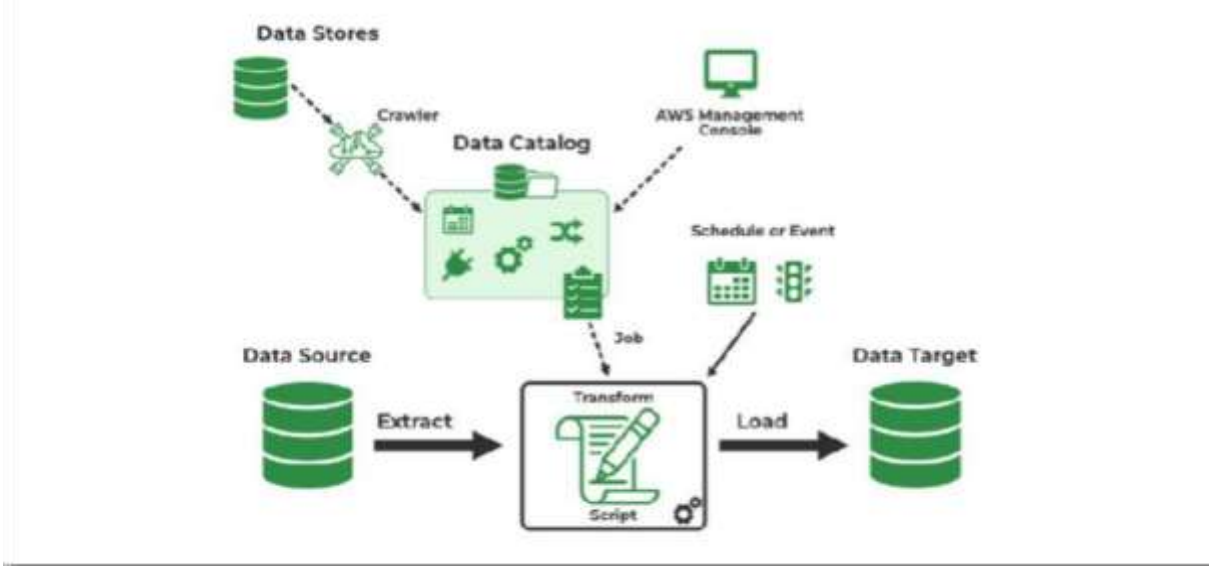
serverless, which means there's no infrastructure to manage. With flexible support for all workloads like ETL, ELT, and streaming in one service, AWS Glue supports users across various workloads and types of users.

Also, AWS Glue makes it easy to integrate data across your architecture. It integrates with AWS analytics services and Amazon S3 data lakes. AWS Glue has integration interfaces and job-authoring tools that are easy to use for all users, from developers to business users, with tailored solutions for varied technical skill sets.

Functions of AWS Glue:

- Extracts data from various sources
- Transforms data into required format
- Loads data into Amazon S3 or other services
- Automatically discovers schema

AWS Glue performs the ETL (Extract, Transform, Load) process in the data lake.



• 4.4.4 Data Lake Architecture Workflow

The workflow of building a data lake using AWS includes the following steps:

1) Data Ingestion

Data is collected from different sources such as applications, databases, IoT devices, etc.

2) Data Storage

The collected raw data is stored in Amazon S3.

3) Data Processing

AWS Glue cleans, transforms, and prepares the data.

4) Data Cataloging

AWS Glue Data Catalog stores metadata and schema information.

5) Data Analysis

Processed data can be analyzed using AWS analytics services.

6) Visualization

Reports and dashboards are created for decision-making.

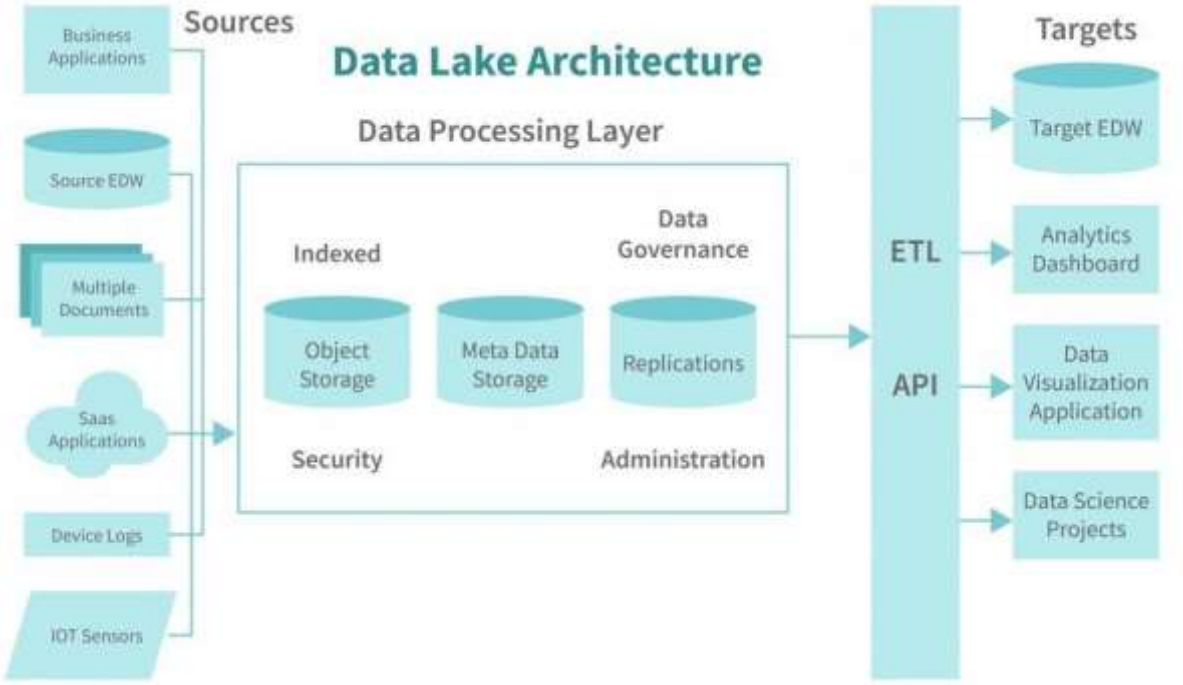


Fig : data lake architecture

4.4 Business Intelligence (BI)

In the modern enterprise, data is often described as the "new oil." However, raw data, much like crude oil, is useless until it is refined. In the context of cloud computing, **Business Intelligence (BI)** serves as the refinery that transforms vast, disorganized data lakes into actionable insights.

4.4.1 Meaning of Business Intelligence

Business Intelligence refers to the procedural and technical infrastructure that collects, stores, and analyzes the data produced by a company's activities. It is an umbrella term that encompasses data mining, process analysis, performance benchmarking, and descriptive

analytics.

In a cloud environment, BI shifts from "hindsight" (what happened?) to "insight" (why did it happen?) and "foresight" (what will happen next?). By leveraging the cloud's elasticity, BI tools can now process structured and unstructured data from disparate sources—social media, IoT sensors, and ERP systems—to provide a 360-degree view of the business.

4.4.2 Importance of BI for Executives

For C-suite executives, BI is not just a reporting tool; it is a **strategic compass**. The transition to cloud-based BI has moved decision-making from "gut feeling" to evidence-based strategy.

- **Accelerated Decision-Making:** Executives no longer have to wait for end-of-month reports. Cloud BI provides "Single Source of Truth" dashboards that reflect current performance.
- **Identifying Market Trends:** By analyzing historical data alongside market variables, executives can identify emerging patterns and pivot strategies before competitors do.
- **Operational Efficiency:** BI highlights bottlenecks in supply chains or internal workflows, allowing for targeted interventions that reduce waste and overhead.
- **Risk Mitigation:** Advanced BI models can predict financial or operational risks, enabling executives to implement safeguards proactively rather than reactively.

4.4.3 Real-Time Analytics in the Cloud

The pinnacle of modern BI is **Real-Time Analytics**. Unlike traditional batch processing— where data is collected and processed in cycles (e.g., every 24 hours)—real-time analytics processes data the moment it enters the system.

Key Components in a Cloud Context:

1. **Streaming Data Pipelines:** Using tools like Apache Kafka or AWS Kinesis, data flows continuously from the source to the analytical engine.
2. **In-Memory Processing:** Cloud providers utilize high-performance RAM to process data instantly, bypassing the delays of traditional disk-based databases.
3. **Instant Visualization:** Dashboards update in milliseconds, allowing a floor manager to see a machine failure or a trader to see a price fluctuation the second it occurs.

Note for the Reader: Real-time analytics in the cloud removes the "latency of thought." When the time between an event occurring and the analysis of that event approaches zero, the business becomes truly "live.

4.5 Integration of Domo for Executive Dashboards

While AWS provides the heavy-duty storage and computing power, **Domo** provides the

sophisticated "last mile" of data delivery: the executive dashboard.

4.5.1 Introduction to Domo

Domo is a cloud-native business management platform. It is unique because it integrates the entire data pipeline—from connection and preparation to visualization and collaboration—into a single interface. It is designed specifically with the non-technical executive in mind, prioritizing ease of use and mobile accessibility.

4.5.2 Features of Domo

Domo stands out in the crowded BI market due to several key features:

- **Magic ETL:** A visual, drag-and-drop tool for data transformation that requires no coding.
- **Beast Mode:** A calculation layer that allows for the creation of complex, custom business metrics on the fly.
- **Buzz:** An integrated collaboration tool that allows teams to discuss specific data "cards" within the platform.
- **Data Governance:** Robust tools to manage who sees what data, ensuring compliance at every level.

4.5.3 Connecting Domo with AWS Data Lake

The integration of Domo with an AWS Data Lake (primarily Amazon S3 and Amazon Athena) creates a scalable, high-performance architecture.

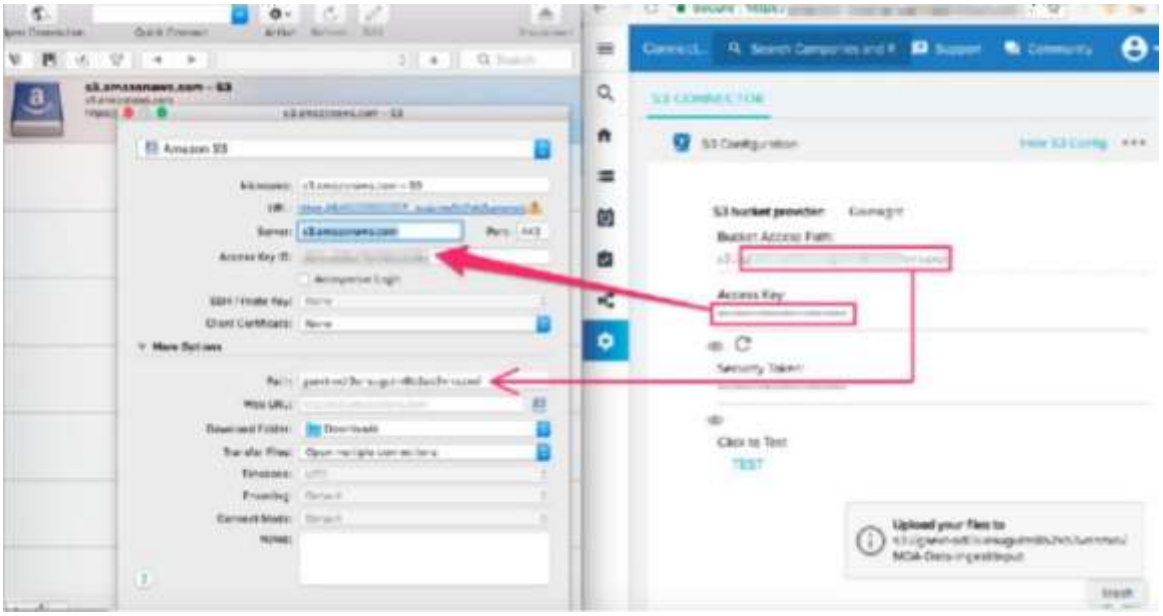
Technical Architecture & Workflow

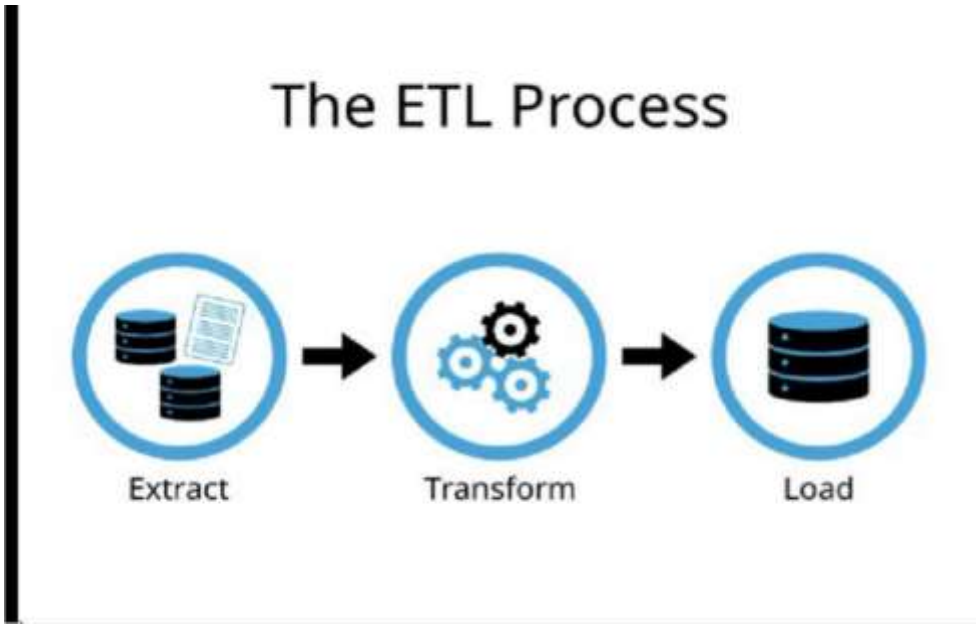
The connection typically follows this architectural path:

1. **Data Ingestion:** Raw data (JSON, CSV, Parquet) is stored in **Amazon S3**.
2. **Schema Definition: AWS Glue** crawls the data to create a metadata catalog.
3. **The Bridge:** Domo connects to AWS using one of two methods:
    - **S3 Connector:** Periodically imports data into Domo's internal high-speed engine (Adrenaline).
    - **Federated Query:** Domo sends a request to **Amazon Athena**, which queries the data directly in S3 without moving it.
4. **Visualization:** The resulting dataset is used to build "Cards" (graphs, maps, or gauges) for the executive dashboard.

Feature	Legacy On-Premise BI	Domo + AWS Cloud BI
Data Refresh	Weekly or Daily	Real-Time / Near Real-Time
Accessibility	VPN/Desktop only	Mobile-First / Browser-Based
Scalability	Limited by Hardware	Elastic (Grows with your data)
Setup Time	Months	Days or Weeks

4.6 Hands-On: End-to-End Data Pipeline Creation





4.6.1 Raw Data Ingestion

Raw data ingestion is the first step in the pipeline. It involves collecting data and storing it in cloud storage.

Upload CSV/JSON to S3

In this step:

- Prepare data files in **CSV or JSON format**
- Login to AWS console
- Open **Amazon S3**
- Create a bucket
- Upload the CSV/JSON file

Amazon S3 stores the raw data in its original format. This acts as the storage layer of the data pipeline.

4.6.2 Data Processing

After uploading data, processing is required to clean and transform it. Processing

includes:

- Removing duplicate records
- Handling missing values
- Converting data types
- Structuring the dataset This

step can be performed using:

- **AWS Glue**
- ETL tools
- Python scripts

The processed data is stored back into S3 or another storage layer for analysis.

4.6.3 Data Integration with DOMO

Data integration connects the processed dataset with a visualization platform like **Domo**.

Connecting S3 Dataset

Steps include:

1. Login to Domo
2. Choose data connector
3. Select Amazon S3 connector
4. Provide bucket details
5. Import dataset

This allows Domo to access data stored in S3.

Scheduling Data Refresh

To keep dashboards updated:

- Set automatic refresh timing (hourly/daily)
- Enable scheduled data updates

This ensures real-time or near real-time data availability.

• 4.7.4 Live Visualization

Live visualization is the final stage of the data pipeline. In this stage, processed data is presented in graphical form using dashboards and reports. Live dashboards automatically update whenever the data source is refreshed. This helps management monitor performance in real time and make quick decisions.

Visualization tools convert complex data into simple and understandable formats such as charts, graphs, and KPI cards.

1) Bar Charts

A bar chart is used to compare values across different categories.

Features:

- Uses rectangular bars to represent data
- Easy to compare multiple categories
- Can be vertical or horizontal

Example:

- Sales by Region

- Revenue by Product Category
- Monthly Customer Growth

Bar charts help in identifying trends and performance differences clearly.

2) Pie Charts

A pie chart represents data in percentage form.

Features:

- Displays proportional distribution
- Divides data into slices
- Total value equals 100%

Example:

- Market share of products
- Percentage of expenses by department
- Customer distribution by location

Pie charts are useful when showing contribution or share of each category.

3) Executive KPI Cards

KPI (Key Performance Indicator) cards display important business metrics in a simple numeric format.

Features:

- Shows single important value
- Easy to read and understand
- Often includes comparison with previous period

Examples of KPI Cards:

- Total Revenue
- Total Sales
- Total Customers
- Monthly Growth Rate

- Profit Percentage

Executive KPI cards provide a quick summary of business performance and help top management in decision-making.

4.7 Advantages of DOMO-Based Data Intelligence

1. Unified Data Integration and Connectors

- **1,000+ Pre-built Connectors:** Domo provides an extensive library of connectors for cloud applications, on-premise databases, spreadsheets, and social media, allowing users to connect data in minutes.
- **Data Silo Elimination:** By acting as a central repository, it breaks down data silos and offers a holistic view of the organization's data landscape.
- **Data Blending:** Data from multiple, disparate sources can be combined into a single, unified view, providing a comprehensive data foundation. www.ardentisys.com +5

2. High-Speed, Real-Time Insights

- **Cloud-Native Architecture:** Being built for the cloud, it eliminates the need for expensive, complex on-premises installations and scales seamlessly.
- **Real-Time Dashboards:** Domo offers real-time data capabilities, ensuring decisions are based on the latest information, allowing for rapid, agile decision-making.
- **Rapid Deployment:** The platform enables quick setup and faster time-to-value compared to traditional BI tools.

3. User Empowerment and Self-Service Analytics

- **No-Code/Low-Code Interface:** Domo features a drag-and-drop interface, specifically "Magic ETL," which allows non-technical business users to clean, join, and transform data without writing complex code.
- **Self-Service Reporting:** Users can create their own reports and visualizations, reducing bottlenecks by minimizing dependency on IT teams.
- **Accessible on Any Device:** With a mobile-first design, users can access dashboards and insights anytime, anywhere.

4. Advanced AI and Predictive Analytics

- **Built-in AI Tools:** Domo embeds AI throughout the platform to help automate reporting, forecast trends, and provide proactive insights.
- **Natural Language Querying (NLQ):** Domo's AI chat allows users to ask questions in plain language and get instant visualizations or answers.

- **Predictive Insights:** Domo allows users to build, train, and deploy machine learning models to forecast future trends.

5. Enhanced Collaboration and Actionable Intelligence

- **Buzz Collaboration Tool:** Domo includes "Buzz," a built-in chat app that allows users to discuss insights directly within the context of the data.
- **Actionable Workflows:** Beyond just viewing data, users can trigger automated workflows, update external systems, and take action immediately based on insights.
- **Customizable Alerts:** Users can set up proactive alerts, notifying them instantly when key metrics change.

6. Security and Governance

- **Granular Permissions:** Domo offers Personalized Data Permissions (PDP), allowing organizations to control exactly what data each user can see, even on shared dashboards.
- **Enterprise-Grade Security:** The platform complies with major security standards, including GDPR, HIPAA, SOC 1/2, and ISO standards.
- **Data Lineage:** Domo allows users to track the source and transformation of data to ensure trust in the insights.

7. Unique "Writeback" Capability

- **Bi-directional Data Flow:** Unlike many BI tools that only read data, Domo allows users to write data back to source systems directly from dashboards, enabling seamless updates to operational data.

Chapter 5

FinOps and Cloud Governance

5.1 Introduction to FinOps

FinOps (Financial Operations) is a cloud financial management practice that helps organizations control and optimize their cloud spending. It brings together finance, operations, and engineering teams to manage cloud costs effectively.

• 5.1.1 Meaning of FinOps

FinOps is a combination of the words **Finance** and **Operations**. FinOps is a cloud financial management practice that enables organizations to control, optimize, and manage cloud expenses through collaboration between finance and technical teams.

FinOps is a practice that helps organizations monitor, manage, and optimize cloud costs while maintaining performance and efficiency.

It encourages collaboration between:

- Finance team
- Engineering team
- Operations team

The main focus of FinOps is to ensure that cloud resources are used efficiently and cost-effectively.

FinOps Lifecycle

The **FinOps Lifecycle** is a continuous framework used to manage and optimize cloud costs effectively. It consists of three major phases: **Inform, Optimize, and Operate**. These phases help organizations understand their cloud spending, reduce unnecessary costs, and maintain efficient cloud operations.

The three pillars of FinOps: Inform, optimize, operate

1. Inform

The **Inform** phase focuses on visibility and transparency of cloud spending. In this stage, organizations collect, analyze, and distribute financial and usage data related to cloud services.

The main objective is to create awareness about how much money is being spent, which services are consuming more resources, and which departments or teams are responsible for the costs.

Key activities in this phase include:

- Tracking cloud usage and expenses
- Generating cost reports and dashboards
- Allocating costs to departments or projects
- Setting budgets and forecasting future expenses

By completing this phase, decision-makers gain a clear understanding of their cloud financial position.

2. Optimize

The **Optimize** phase focuses on improving cost efficiency without affecting performance. After understanding spending patterns in the Inform phase, organizations identify areas where costs can be reduced.

The main objective is to eliminate waste and use resources more efficiently. Key

activities in this phase include:

- Removing unused or idle resources
- Right-sizing virtual machines and storage
- Using reserved or spot instances
- Automating resource scheduling

This phase ensures that organizations achieve maximum value from their cloud investments while minimizing unnecessary expenditure.

3. Operate

The **Operate** phase ensures continuous monitoring and governance of cloud operations. It focuses on maintaining financial control and operational efficiency over time.

The main objective is to sustain optimized cloud usage through proper management practices.

Key activities in this phase include:

- Monitoring budgets and usage regularly
- Enforcing cloud governance policies
- Maintaining performance and reliability
- Continuously reviewing and improving cost strategies

This phase helps organizations maintain discipline in cloud spending and prevents cost overruns.

• 5.1.2 Importance of Cost Management in Cloud

Pay-as-you-go model

Also called on-demand models, pay-as-you-go models charge customers based on cloud usage, typically by the hour (but sometimes by the minute or second).
This model offers users plenty of flexibility and scaling capabilities, and can serve as a practical solution for smaller, less complex operations. However, as organizations grow and require a broader range of cloud resources, pay-as-you-go models can become prohibitively expensive.

Cloud services follow a **pay-as-you-use** model. Without proper cost management, organizations may face high and unexpected expenses.

Importance:

1. Prevents unnecessary spending
2. Optimizes resource utilization
3. Avoids budget overruns
4. Improves financial planning
5. Ensures efficient cloud usage
6. Increases return on investment (ROI)

Proper cost management helps organizations track usage, detect waste, and reduce unused resources.

• 5.1.3 Goals of Cloud Governance

Cloud Governance :

- It is the set of policies or principles that act as the guidance for the adoption use, and management of cloud technology services.
- It is an ongoing process that must sit on top of existing governance models.
- It is a set of rules you create to monitor and amend as necessary in order to control costs, improve efficiency, and eliminate security risks.

Goals of Cloud Governance

Cloud Governance refers to policies and controls that ensure proper use of cloud resources.

Main Goals:

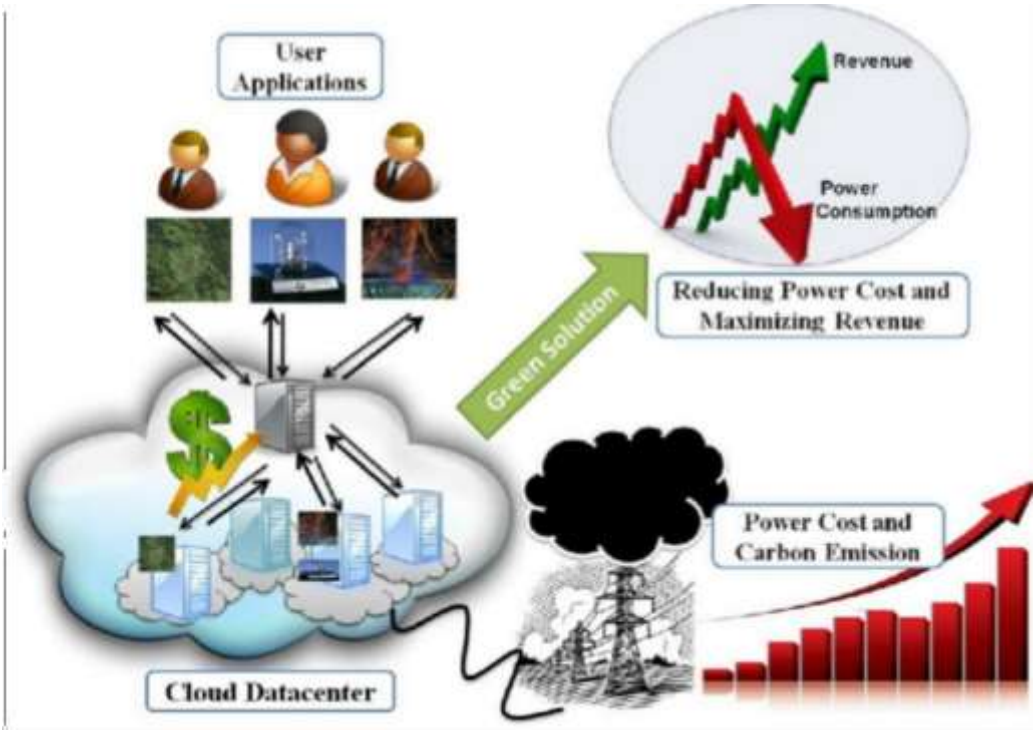
1. **Cost control and optimization:** Ensures that cloud resources are used efficiently to reduce unnecessary expenses and maximize cost savings.
2. **Security and compliance:** Protects cloud data and systems while ensuring adherence to organizational policies and regulatory standards.
3. **Resource standardization:** Maintains uniform configurations and best practices across all cloud resources for consistency and efficiency.
4. **Performance monitoring:** Continuously tracks system performance to ensure reliability, availability, and optimal operation.
5. **Risk management:** Identifies and minimizes potential threats, failures, and security vulnerabilities in the cloud environment.
6. **Accountability and transparency:** Provides clear visibility of cloud usage and responsibilities to promote proper governance and informed decision-making.

Cloud governance ensures that cloud resources are used responsibly and aligned with business objectives.

5.2 Sustainability in Cloud Computing

Sustainability in cloud computing refers to using cloud resources in an environmentally responsible manner. As organizations increasingly depend on cloud services, it becomes important to reduce energy consumption and minimize environmental impact.

Sustainable cloud practices focus on reducing carbon emissions, improving energy efficiency, and promoting green technology.



Above fig shows , Cloud sustainability focuses on running user applications in cloud datacenters in an energy-efficient manner by applying green computing techniques to reduce power consumption, lower carbon emissions and operational costs, while improving performance and maximizing revenue, thereby achieving both environmental protection and business growth.

• 5.2.1 Understanding Cloud Carbon Footprint

Cloud carbon footprint refers to the total amount of carbon dioxide (CO_2) emissions produced due to the use of cloud services.

When we store data, run applications, or process workloads in the cloud, data centers consume electricity. If this electricity is generated using fossil fuels like coal or gas, it produces carbon emissions.

Even though cloud providers are generally more energy-efficient than traditional data centers, cloud usage still contributes to environmental impact.

- More cloud usage means more energy consumption.
- Energy used for servers, cooling systems, and networking equipment produces emissions.
- Efficient cloud usage can reduce carbon footprint.
- Monitoring and optimizing workloads helps in lowering environmental impact.

Organizations should regularly monitor their cloud usage and avoid keeping unused resources

running unnecessarily.

Cloud computing and data centers have become core global infrastructure—the roads and bridges of the internet economy. But they've also become a significant driver of carbon emissions, responsible for an estimated 1.8% of US electricity consumption and the plurality of emissions for many tech companies.

The good news is that cloud giants like Google and Microsoft have become some of the world's largest purchasers of clean power, accelerating the deployment of zero-carbon electricity. The bad news is that not all cloud providers have been as bold, and the true carbon cost of cloud computing is still opaque for most companies.

Watershed helps many of the world's fastest-growing technology companies—like Stripe and Shopify—cut their emissions, including from cloud operations. In this guide, we're sharing what we've learned about calculating emissions from cloud providers and data centers, and about how to drive those emissions to zero.

Data centers and carbon emissions

Technology companies run on data centers: sprawling facilities that house servers that store and process the information that makes up the internet. And it's not just the tech industry: virtually all internet-era companies have growing needs for computing, hosted services, storage, and networking capacity. There are now millions of data centers globally, split across three basic operational structures:

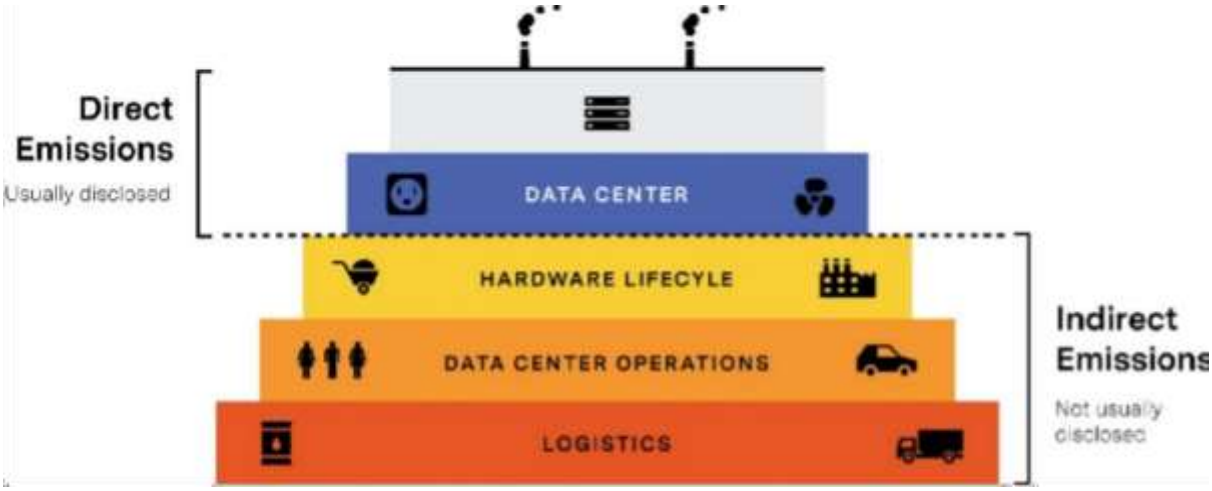
- Some companies build and manage their own facilities
- Some buy their own equipment and co-locate it with a partner like Equinix (referred to as a "colo")
- Some use a cloud service provider like Amazon Web Services (AWS), Google Cloud, or Microsoft Azure to fully manage their cloud

Each of these structures drives significant carbon emissions—both directly (from electricity consumption) and indirectly (e.g., from extracting and shipping the raw materials used to manufacture servers). But the scale of these emissions has long gone underreported. Until a few years ago, very few cloud customers had the means of measuring, understanding, or managing their true contributions.

At Watershed, we've worked with experts to develop a methodology for counting these emissions—down to narrow particulars like instance type, chip brand, cloud provider, and geographic region. (We've drawn on excellent academic work by Arman Shehabi, Jonathan Koomey and others.) We're sharing our high-level approach here.

What causes emissions of data centres?

The first step is to make sure you're capturing all emissions. We think about emissions in two categories: the "direct" emissions from the electricity necessary to operate a data center, and the "indirect" emissions from manufacturing hardware and other operations to deliver cloud services.



The GHG Protocol and cloud

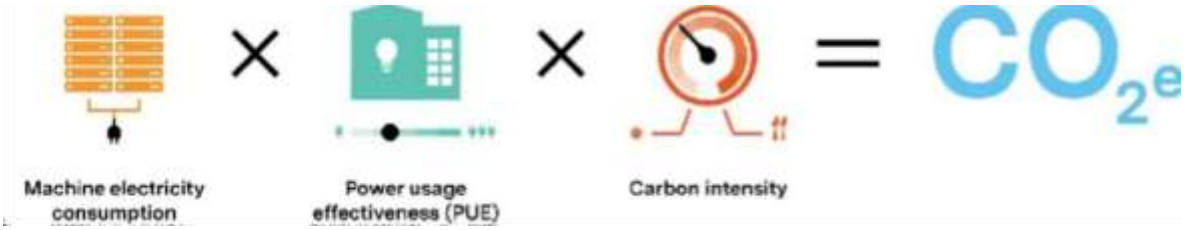
The Greenhouse Gas Protocol—i.e., the equivalent of GAAP for carbon accounting— divides emissions into three "scopes":

- **Scope 1** - Greenhouse gases directly emitted by facilities you own (e.g., burning gas to power your car or office)
- **Scope 2** - Emissions from electricity you purchase
- **Scope 3** - Everything else: carbon emitted upstream (by your suppliers) and downstream (by your customers)

These scopes are relative. If you own a data center, the electricity it consumes is Scope 2, and the emissions from the manufacturing of the underlying hardware are Scope 3. If you're a colo tenant, the electricity used to power your hardware would be Scope 2, but the electricity used to power the broader facility would be Scope 3. If you're using a cloud provider, everything is Scope 3— i.e., by far the largest and most underreported category.

Direct emissions

We've used industry research and the academic literature to develop a formula for data center emissions specifically:



Cloud emissions mainly come from **electricity used by servers**, extra facility energy (measured by **PUE**), and the **carbon intensity** of the power source. Total impact is calculated as **CO_2e emissions**. In addition to electricity, there are **indirect (Scope 3) emissions** from hardware manufacturing, transportation, and data center operations.

How to Reduce Cloud Carbon Footprint (Short)

1. **Use Clean Energy** – Choose renewable-powered data centers or pressure providers to use 100% clean electricity.
2. **Demand Transparency** – Ask cloud providers to clearly report full emissions (including Scope 3).
3. **Aim for Net Zero** – Reduce all direct and indirect emissions and support permanent carbon removal.

5.3 Cost Optimization Strategies

In the cloud, "set it and forget it" is a recipe for fiscal disaster. Because cloud billing is consumption-based, optimization isn't a one-time event—it's a continuous loop of monitoring, analyzing, and refining. This section explores the four primary levers you can pull to slash waste without sacrificing performance.

5.3.1 Identifying Unused or Idle Resources

The most expensive resource is the one you pay for but don't use. "Zombies" in your infrastructure often hide in plain sight.

- **Unattached Storage:** When a Virtual Machine (VM) is deleted, its attached storage (like AWS EBS volumes) often persists, incurring costs indefinitely.
- **Idle Load Balancers:** Often left over from abandoned testing environments or old deployments.
- **Orphaned Snapshots:** Backups of systems that no longer exist.
- **Idle Instances:** Instances running at $0-5\%$ CPU utilization for extended periods.

Pro-Tip: Use native tools like AWS Trusted Advisor, Azure Advisor, or Google Cloud Recommender to generate "Low Utilization" reports automatically.

5.3.2 Automated Resource Scheduling

Most businesses operate on a 40-to-50-hour work week, yet their non-production environments (Dev, Test, QA) often run 168 hours a week.

- **The "Off" Switch:** Automate the shutdown of non-critical resources during evenings

and weekends.

- **The Impact:** Moving from a 24/7 schedule to a 12/5 schedule (12 hours a day, 5 days a week) can result in a **savings of over 60%** on those specific instances.
- **Tools:** Utilize "Instance Schedulers" or serverless functions (like AWS Lambda) to trigger start/stop actions based on time-tags.

5.3.3 Spot Instance Strategy

Spot instances (or Preemptible VMs) allow you to bid on spare cloud capacity at a massive discount—often **70% to 90% off** the On-Demand price.

- **The Catch:** The cloud provider can reclaim the capacity with very short notice (usually 30 seconds to 2 minutes).
- **Best Use Cases:** * Stateless applications.
    - Batch processing and data encoding.
    - CI/CD testing pipelines.
- **Strategy:** Never use Spot instances for your primary database or any stateful service that cannot handle an abrupt shutdown.

5.3.4 Rightsizing Cloud Resources

Rightsizing is the process of matching instance types and sizes to your actual workload performance and capacity requirements.

Condition	Strategy
Over-provisioned	If an instance peaks at 10% CPU, downsize it to a smaller family (e.g., move from a large to a medium).
Wrong Family	If a memory-heavy app is running on a compute-optimized instance, move it to a memory-optimized family to get better value per GB.
Modernization	Upgrade to the latest generation (e.g., moving from m5 to m6) which usually offers better performance at the same or lower price point.

5.4 Hands-On: Budgeting and Cost Monitoring

Budgeting and cost monitoring are important parts of cloud financial management. In cloud computing, services are charged based on usage. Therefore, organizations must regularly track their expenses to avoid unexpected bills. AWS provides built-in tools to create budgets and detect unusual spending patterns.

5.4.1 Creating a Custom AWS Budget

A custom AWS budget helps organizations control their cloud spending by setting financial limits and receiving alerts when spending exceeds the defined amount.

This can be done using **AWS Budgets**, a cost management service provided by **Amazon Web Services**.

Steps to Create a Budget:

1. Login to AWS Management Console.
2. Open Billing and Cost Management Dashboard.
3. Select “Create Budget.”
4. Choose budget type (Cost budget).
5. Enter budget amount and time period (monthly).
6. Configure alerts and notifications.
7. Review and create the budget.

– Setting Monthly Limits

Setting a monthly limit means defining the maximum amount that can be spent in one month.

For example:
If the monthly budget is ₹10,000, AWS tracks usage and compares actual spending with the set limit.

This helps:

- Control unnecessary expenses
- Plan financial resources properly
- Avoid unexpected high bills

– Budget Alerts

Budget alerts notify users when spending reaches a certain percentage of the budget. For example:

- Alert at 50% usage
- Alert at 80% usage
- Alert at 100% usage

Notifications can be sent via:

- Email
- SNS (Simple Notification Service)

Alerts help teams take immediate action before overspending occurs.

5.4.2 Cost Anomaly Detection System

Cost anomaly detection helps identify unusual or unexpected cloud spending patterns.

AWS provides a service called **AWS Cost Anomaly Detection** to automatically detect abnormal spending.

– Monitoring Unusual Spending

The system uses machine learning to analyze historical usage patterns. If there is:

- Sudden increase in cost
- Unexpected resource usage
- Abnormal service charges

The system detects it as an anomaly. Example:
If the average daily cost is ₹500 and suddenly it becomes ₹2000, the system flags it as unusual spending.

– Automated Notifications

When unusual spending is detected:

- Automatic alerts are sent to registered users.

- Notifications can be received via email or SNS.
- Users can immediately investigate the issue.

This helps in:

- Preventing financial loss
- Identifying misconfigured resources
- Detecting security issues

CASE STUDY: CLOUD PLATFORMS (AWS, AZURE, GOOGLE CLOUD, ALIBABA)

1. AMAZON WEB SERVICES (AWS)

Project Goals

- Build a web application capable of scaling smoothly while maintaining speed, reliability, and security.

- Use Infrastructure as Code (Terraform) to provision AWS resources consistently.

- Ensure the platform can handle growth in users, features, and workloads without performance trade-offs.

- Transition the existing environment from GCP to AWS.

- Establish separate production and non-production environments.

- Implement CI/CD pipelines and backups.

- Introduce monitoring and alerting systems.

Challenges

- Finding skilled DevOps team.

- Multi-cloud expertise requirement.

- Downtime due to poor infrastructure practices.

Solution Approach

- AWS setup with best practices.

- Terraform automation.

- CI/CD with Jenkins.

- Monitoring with CloudWatch.

- Migration using Docker and S3.

Client Benefits

- Flexibility and scalability.

- Simplified migration.

- Rapid deployment.

- Enhanced security.

- Cost efficiency.

Technologies Applied

- AWS: EC2, S3, EKS, RDS, IAM, etc.

- Tools: Jenkins, GitHub, Terraform.

2. MICROSOFT AZURE

Introduction

Azure launched in 2010 with strong hybrid cloud capabilities.

Business Model

- Hybrid cloud and enterprise security.

- Customers: Enterprises, SMBs.

- Revenue: Subscription-based.

Customer Journey

- Awareness → Exploration → Adoption → Optimization → Expansion

Infrastructure

- Global data centers.

- IaaS, PaaS, SaaS.

- Security: Azure AD.

Reflection

- Strengths: Hybrid cloud.

- Challenges: Complexity.

3. GOOGLE CLOUD

Introduction

Google Cloud focuses on AI, analytics, and sustainability.

Business Model

- AI/ML tools, BigQuery.

- Pay-as-you-go model.

Customer Journey

- Awareness → Exploration → Adoption → Optimization → Expansion

Infrastructure

- BigQuery, Vertex AI.

- Zero-trust security.

Reflection

- Strengths: AI leadership.

- Challenges: Lower market share.

4. ALIBABA

Introduction

Alibaba is a global e-commerce and cloud company.

Business Model

- B2B focus.

- Revenue from ads, commissions.

Customer Journey

- Awareness → Search → Registration → Purchase → Delivery → Support

Infrastructure

- Alibaba Cloud.

- Alipay, Cainiao logistics.

Reflection Positive:

- Strong ecosystem.

Negative:

- Counterfeit products.

- Data privacy concerns.

REFERENCES

[1] T. Erl, R. Puttaswamy, and C. Khattak, Cloud Computing: Concepts, Technology & Architecture. Upper Saddle River, NJ: Prentice Hall, 2013.

[2] M. J. Kavis, Architecting the Cloud: Design Decisions for Cloud Computing Service Models. Hoboken, NJ: Wiley, 2014.

[3] C. Davis, Cloud Native Patterns: Designing Change-tolerant Software. Shelter Island, NY: Manning, 2019.

[4] G. Hohpe, Cloud Strategy: A Decision-based Approach to Successful Cloud Migration. Independent, 2020.

[5] S. Orban, Ahead in the Cloud: Best Practices for Navigating the Future of Enterprise IT. Seattle, WA: CreateSpace, 2018.

[6] T. Mather, S. Kumaraswamy, and S. Latif, Cloud Security and Privacy. Sebastopol, CA: O'Reilly Media, 2009.

[7] I. Moyse, The Cloud Strategy Playbook. London, UK: Business Expert Press, 2022.

[8] T. Hoff, Explain the Cloud Like I'm 10. Independent, 2017.

[9] J. Garrison and K. Nova, Cloud Native Infrastructure. Sebastopol, CA: O'Reilly Media, 2017.

[10] J. Bond, The Enterprise Cloud: Best Practices for Transforming Legacy IT. Sebastopol, CA: O'Reilly Media, 2015.

[11] B. Wilder, Cloud Architecture Patterns. Sebastopol, CA: O'Reilly Media, 2012.

[12] E. Mansoor, Mastering AWS Cost Optimization. Birmingham, UK: Packt Publishing, 2018.

[13] B. Sosinsky, Cloud Computing Bible. Indianapolis, IN: Wiley, 2011.

[14] J. J. Geewax, Google Cloud Platform in Action. Shelter Island, NY: Manning, 2018.

[15] J. Mulder, Multi-Cloud Strategy for Cloud Engineers. Birmingham, UK: Packt Publishing, 2020.

[16] K. A. Kumari and A. S. Khan, Edge Computing: Fundamentals, Design and Applications. Boca Raton, FL: CRC Press, 2022.

[17] P. Sbarski, Serverless Architectures on AWS. Shelter Island, NY: Manning, 2017.

[18] C. Binnie, Practical Cloud-Native Security. Sebastopol, CA: O'Reilly Media, 2021.

[19] P. van Eijk, Cloud Governance: A Practical Guide. Independent, 2019.

[20] A. Maurya, Running Lean: Iterate from Plan A to a Plan That Works. Sebastopol, CA: O'Reilly Media, 2012.

[21] M. Kavis, High Performance Cloud Auditing. Hoboken, NJ: Wiley, 2021.

[22] J. Baron et al., AWS Certified Solutions Architect Official Study Guide. Indianapolis, IN: Sybex, 2016.

[23] P. De Tender, Azure Strategy and Implementation Guide. Birmingham, UK: Packt Publishing, 2020.

[24] N. B. Ruparelia, Cloud Computing. Cambridge, MA: MIT Press, 2016.

[25] M. L. Abbott and M. T. Fisher, The Art of Scalability. Upper Saddle River, NJ: Addison-Wesley, 2015. ## DevOps & SRE IEEE References

[26] G. Kim, K. Behr, and G. Spafford, The Phoenix Project: A Novel about IT, DevOps, and Helping Your Business Win. Portland, OR: IT Revolution Press, 2013.

[27] G. Kim, J. Humble, P. Debois, and J. Willis, The DevOps Handbook. Portland, OR: IT Revolution Press, 2016.
[28] N. Forsgren, J. Humble, and G. Kim, Accelerate: The Science of Lean Software and DevOps. Portland, OR: IT Revolution Press, 2018.
[29] B. Beyer, C. Jones, J. Petoff, and N. R. Murphy, Site Reliability Engineering: How Google Runs Production Systems. Sebastopol, CA: O'Reilly Media, 2016.
[30] J. Humble and D. Farley, Continuous Delivery. Upper Saddle River, NJ: Addison- Wesley, 2010.
[31] G. Kim, The Unicorn Project. Portland, OR: IT Revolution Press, 2019.
[32] J. Davis and K. Daniels, Effective DevOps. Sebastopol, CA: O'Reilly Media, 2016.
[33] B. Beyer et al., The Site Reliability Workbook. Sebastopol, CA: O'Reilly Media, 2018.
[34] Y. Brikman, Terraform: Up & Running. Sebastopol, CA: O'Reilly Media, 2019.
[35] K. Hightower, B. Burns, and J. Beda, Kubernetes: Up & Running. Sebastopol, CA: O'Reilly Media, 2017.
[36] K. Morris, Infrastructure as Code. Sebastopol, CA: O'Reilly Media, 2016.
[37] S. Newman, Building Microservices. Sebastopol, CA: O'Reilly Media, 2015.
[38] S. Newman, Monolith to Microservices. Sebastopol, CA: O'Reilly Media, 2019.
[39] M. Skelton and M. Pais, Team Topologies. Portland, OR: IT Revolution Press, 2019.
[40] J. Arundel and J. Domingus, Cloud Native DevOps with Kubernetes. Sebastopol, CA: O'Reilly Media, 2019.
[41] M. Hering, DevOps for the Modern Enterprise. Portland, OR: IT Revolution Press, 2018.
[42] D. N. Blank-Edelman, Seeking SRE. Sebastopol, CA: O'Reilly Media, 2018.
[43] L. Campbell and C. Majors, Database Reliability Engineering. Sebastopol, CA: O'Reilly Media, 2017.
[44] C. Majors, L. Fong-Jones, and G. Miranda, Observability Engineering. Sebastopol, CA: O'Reilly Media, 2022.
[45] B. Yuen and A. Matyushentsev, GitOps and Kubernetes. Shelter Island, NY: Manning, 2021.
[46] G. Gruver and T. Mouser, Leading the Transformation. Portland, OR: IT Revolution Press, 2015.
[47] N. Poulton, Docker Deep Dive. Independent, 2020.
[48] L. Hochstein and R. Moser, Ansible: Up and Running. Sebastopol, CA: O'Reilly Media, 2017.
[49] J. Verona, Practical DevOps. Birmingham, UK: Packt Publishing, 2016.
E. M. Goldratt, The Goal: A Process of Ongoing Improvement. Great Barrington, MA: North River Press, 2004

SUBJECT: Cloud Computing and DevOps Author's Profile

1. Dr. Kamal Shah

Dr. Kamal shah is the Principal of St. John College of Engineering and Technology, Palghar. She is a distinguished academician with a Postdoctoral degree in Quantum Computing and a Ph.D. in Information Technology. With an extensive teaching career spanning 27 years, she has imparted knowledge across diverse domains, including Electrical Engineering, Telecommunication Engineering, and Information Technology. Her research interests are centered on Quantum Computing and Machine Learning, fields in which she has made significant contributions. Renowned for her expertise, she is a nationally recognized faculty trainer, specializing in various educational frameworks and advancements in quantum computing technology. Her dedication to academic excellence and innovation has solidified her reputation as a leader in engineering education.

2. Ms. Aishwarya Churi

Ms. Aishwarya Churi is an Assistant Professor in the Department of Computer Science (Data Science) at St. John College of Engineering and Management, Palghar, Maharashtra, with over three years of academic experience since 2022. She has taught a wide range of core subjects including Database Management Systems, Data Warehousing and Mining, Computer Networks, Data Structures, Algorithms, Computer Organization and Architecture, Embedded Systems, VLSI Design, and Digital Electronics. Her academic profile is strengthened by a published patent and several research papers in reputed IEEE and Scopus-indexed journals, focusing on advanced communication systems and emerging technologies. She has also completed professional certifications in Applied Machine Learning, Cryptography, and Artificial Intelligence, demonstrating her commitment to continuous learning. Actively involved in faculty development programs, technical workshops, and student mentoring, she is recognized for her disciplined, student-centric approach, commitment to quality education, and contributions to research and technical textbook authorship.

3. Mrs. Karishma Tamboli

Mrs. Karishma Jamir Tamboli is an Assistant Professor at St. John Polytechnic College, Palghar, Maharashtra, with teaching experience and a strong background in Information Technology and Cloud Computing. She holds a B.E. in Information Technology from Pune University and has practical expertise in modern technologies such as AWS Cloud Computing, Docker, Kubernetes, Jenkins, and DevOps tools. She has completed professional certifications including AWS Certified Cloud Practitioner and AWS re/Start Graduate, demonstrating her commitment to advancing her technical knowledge. She has worked on several cloud-based projects including CI/CD pipeline development and web application deployment using AWS, Jenkins, Docker, and Git, significantly improving deployment efficiency and automation. In addition to her technical work, she has actively participated in national and international technical events, where she received the Best Research Paper Award in an International Paper Presentation for her research contribution. With prior experience as an educator and a passion for mentoring students, she is known for her dedicated teaching approach, technical expertise, and commitment to continuous learning and innovation in cloud and software technologies.

4. Mr. Pranjal Save

Pranjal Save has technical expertise includes programming languages such as C, C++, Java, Python, JavaScript, along with HTML, CSS, and SQL for web and database development. He has practical experience in developing Flutter-based mobile applications, UI/UX projects, and SQL- based backend systems, and continues to explore modern technologies such as cloud computing, backend deployment, and system design. He also has a keen interest in research and academic writing, particularly in emerging areas such as cloud computing, cybersecurity, system design, and modern software architectures, and aims to contribute to the research community through technical publications and research papers. Known for his curiosity and problem-solving mindset, he believes in learning by experimenting with systems, understanding their internal workings, and improving them through practical implementation.

Published by

IMPERIAL PUBLICATIONS