1. Cloud Characteristics

- Cloud computing has several characteristics that distinguish it from traditional computing models. Here are some examples of how these characteristics are exhibited in cloud computing:
    1. **On-demand self-service:**
    2. **Broad network access:**
    3. **Resource pooling:**
    4. **Rapid elasticity:**
    5. **Measured service:**
    6. **Service Models:**
    7. **Deployment Models:**

1. On-demand self-service: Users can provision computing resources, such as server time and network storage, automatically without requiring human interaction with each service provider.

For example, Amazon Web Services (AWS) allows users to provision virtual servers, storage, and databases on-demand through their web console or API.

2. Universal access: Cloud services are available over the network and can be accessed by a variety of devices, including smartphones, tablets, and laptops.

For example, Microsoft Office 365 provides access to Microsoft Office applications and other productivity tools over the internet, allowing users to work from anywhere on any device.

3. Resource pooling: Multiple users can share the same physical resources, such as servers and storage devices, while still maintaining security and privacy.

For example, Google Cloud Platform provides a shared infrastructure for computing, storage, and networking that can be used by multiple customers.

4. Rapid elasticity: Cloud services can be scaled up or down quickly to meet changing demands.

For example, Netflix uses AWS to host its streaming video service, which requires massive amounts of computing and storage resources. During peak usage times, Netflix can quickly scale up its infrastructure to handle the increased demand, and then scale it back down when demand decreases.

5. Measured service: Cloud services are monitored and measured, and users are charged based on their usage.

For example, AWS charges customers based on the amount of computing resources they use, such as the number of virtual servers and the amount of storage and data transfer. This allows customers to pay only for what they use, rather than having to invest in expensive hardware and software upfront.

6. Service Models: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) cater to different needs, offering flexibility in resource management.

7. Deployment Models: Public Cloud, Private Cloud, and Hybrid Cloud options provide varying levels of control, security, and customization for organizations.

- The characteristics of cloud computing include:

1. On-demand self-service: Users can access and provision computing resources, such as storage and processing power, as needed without requiring human interaction with service providers.

2. Broad network access: Cloud services are accessible over the internet through various devices, such as laptops, smartphones, and tablets.

3. Resource pooling: Computing resources are pooled together and shared among multiple users, allowing for efficient utilization and scalability.

4. Rapid elasticity: Cloud services can quickly scale up or down based on the demand, allowing users to easily adjust their resource allocation.

5. Measured service: Cloud usage is monitored and measured, enabling users to pay for only the resources they consume.

6. Service Models: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) cater to different needs, offering flexibility in resource management.

7. Deployment Models: Public Cloud, Private Cloud, and Hybrid Cloud options provide varying levels of control, security, and customization for organizations.

These characteristics enable the flexibility, scalability, and cost-effectiveness of cloud computing.

SUMMARIZED:

1. **On-Demand Self-Service:** Users can easily manage and provision computing resources without human intervention, making resources readily available.
2. **Broad Network (Universal) Access:** Cloud services are accessible over various networks, allowing users to connect from different devices like laptops, smartphones, and tablets.
3. **Resource Pooling:** Computing resources are dynamically assigned and reassigned based on demand, with users having little control over the exact location of these resources.
4. **Rapid Elasticity:** Cloud resources can quickly scale up or down to meet changing demands, providing flexibility and efficiency.
5. **Measured Service:** Usage is automatically optimized, measured, and reported, allowing users to be billed based on actual usage, promoting transparency and cost efficiency.
6. **Service Models:** Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) cater to different needs, offering flexibility in resource management.
7. **Deployment Models:** Public Cloud, Private Cloud, and Hybrid Cloud options provide varying levels of control, security, and customization for organizations.

2. Cloud Delivery Models

- Cloud computing has three main delivery models:
  - *Software as a Service (SaaS)*, *Platform as a Service (PaaS),* and *Infrastructure as a Service (IaaS)*.
- Here are some examples of how these delivery models are used in cloud computing:

1. Software as a Service (SaaS): SaaS provides software applications over the internet, allowing users to access them from anywhere without having to install them on their own devices. Examples of SaaS include Google Docs, which provides a suite of productivity tools such as word processing, spreadsheets, and presentations, and Salesforce, which provides customer relationship management (CRM) software.

2. Platform as a Service (PaaS): PaaS provides a platform for developing and deploying applications, including tools and services for building, testing, and deploying software. Examples of PaaS include Microsoft Azure, which provides a platform for building and deploying web applications, and Google App Engine, which provides a platform for building and deploying scalable web applications.

3. Infrastructure as a Service (IaaS): IaaS provides virtualized computing resources, including servers, storage, and networking, that can be used to build and deploy applications. Examples of IaaS include Amazon Web Services (AWS), which provides virtual servers, storage, and databases that can be used to build and deploy applications, and Microsoft Azure, which provides virtual machines, storage, and networking resources.

- Each of these delivery models provides different levels of abstraction and control over the underlying infrastructure. SaaS provides the highest level of abstraction, allowing users to access software applications without having to worry about the underlying infrastructure. PaaS provides a platform for building and deploying applications, but still abstracts away much of the underlying infrastructure. IaaS provides the lowest level of abstraction, allowing users to have full control over the underlying infrastructure, but also requiring more expertise and management.

There are different cloud delivery models, including:

1. Infrastructure as a Service (IaaS): This model provides virtualized computing resources over the internet. Customers can rent virtual machines, storage, and networks to build their own IT infrastructure.

2. Platform as a Service (PaaS): PaaS offers a platform for developing, testing, and deploying applications. It provides a complete development environment, including operating systems, programming languages, and databases.

3. Software as a Service (SaaS): SaaS delivers software applications over the internet on a subscription basis. Users can access and use the software through a web browser without the need for installation or maintenance.

Examples:

Sure! One **example** of a cloud delivery model is
Amazon EC2 (Elastic Compute Cloud) (IaaS),
which is a public cloud that provides infrastructure as a service (IaaS). With EC2, customers can rent virtual servers in the cloud and have complete control over the operating system and applications running on those servers. This allows customers to easily scale their computing resources up or down based on their needs, without having to invest in and manage physical hardware.

Another example
Google AppEngine (PaaS),
which is also a public cloud but provides an application development platform as a service (PaaS). With AppEngine, developers can build and deploy web applications without having to worry about the underlying infrastructure. Google takes care of managing the servers, storage, and networking, allowing developers to focus solely on writing and deploying their applications.

Another example
SalesForce (SaaS),
of a cloud delivery model, specifically software as a service (SaaS). SalesForce.com provides a suite of business applications, such as customer relationship management (CRM), that are accessed over the internet. Customers can simply subscribe to the service and access the applications through a web browser, without the need to install or maintain any software on their own servers.

These examples demonstrate different cloud delivery models,
where customers can choose the level of control and responsibility they want to have over their IT infrastructure and applications. Public clouds like EC2, AppEngine, and SalesForce.com offer convenience, scalability, and cost savings, making them attractive options for businesses of all sizes.

4. myTrendek application Case study

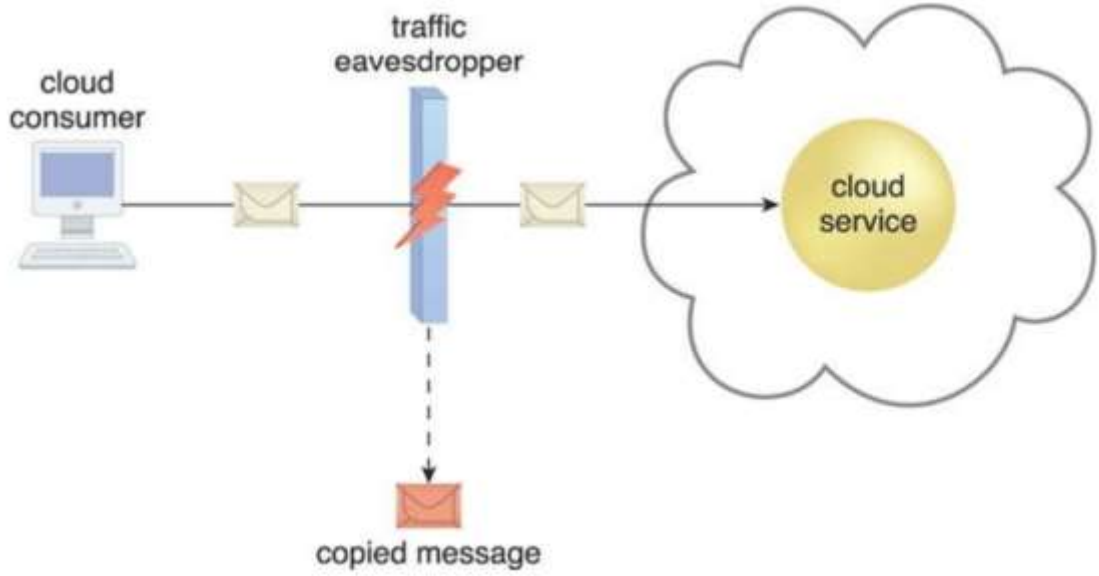
From Concept ref book

5. Types of threats in cloud

i. Traffic Eavesdropping:
ii. Malicious Intermediary:
iii. Denial of Service:
iv. Insufficient Authorization:
v. Virtualization Attack:
vi. Overlapping Trust Boundaries:

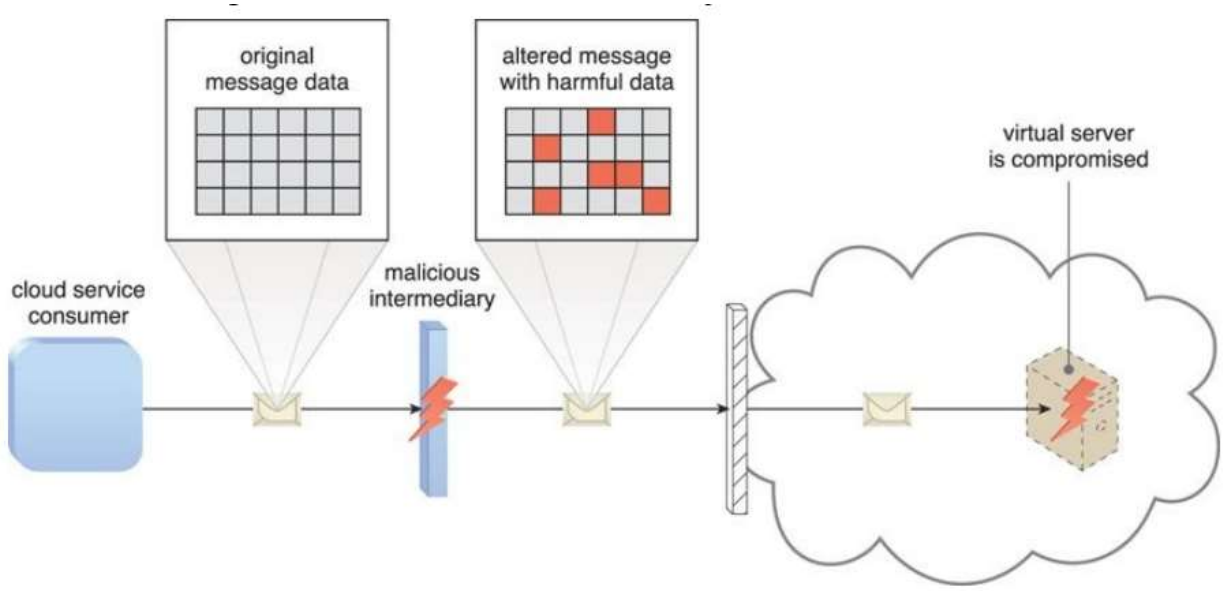
Cloud security threats include various risks and vulnerabilities in cloud computing environments.

i. Traffic Eavesdropping:



i. **Definition**: Traffic eavesdropping: Spying on network traffic to steal secrets.
ii. **Methods**: Attackers eavesdrop using
packet sniffing(*Intercepting and analyzing data packets*),
man-in-the-middle attacks(*Inserting communication to intercept information*) and
network protocol vulnerabilities(*Taking advantage of weaknesses network protocols*).
iii. **Data Privacy**: It can leak sensitive data.
iv. **Data Interception**: It can steal text, images, files, and logins from network traffic.
v. **Wireless Vulnerability**: Wireless networks are easy to eavesdrop on.
vi. **Packet Sniffers**: Attackers use packet sniffers to steal information from network traffic
vii. **MitM(***man-in-the-middle*) **Attacks**: attacker intercepts and alters data between two parties.
viii. **Encryption Importance**: Encryption stops eavesdroppers from reading your data.
ix. **Secure Protocols**: Use HTTPS and VPNs to protect against eavesdropping.

ii. Malicious Intermediary:



i. **Man-in-the-Middle (MitM) Attack**:
They intercept between users and cloud services, and steal data or inject malicious content.

ii. **Eavesdropping**:
Attacker can steal data by eavesdropping on communication between user and cloud.

iii. **Data Manipulation**: Malicious Intermediaries can modify data packets, injecting malicious code or altering data in transit.

iv. **Session Hijacking**: By intercepting authentication tokens or session cookies, attackers can take control of a user's session, gaining unauthorized access.

v. **Phishing**: Malicious Intermediaries may redirect users to fake login pages or cloud service clones to steal login credentials.

vi. **Traffic Redirection:** Malicious intermediaries reroute network traffic through malicious servers, capturing data in transit.

vii. **API Exploitation:** They target vulnerabilities in cloud service APIs, gaining unauthorized access and control over cloud resources

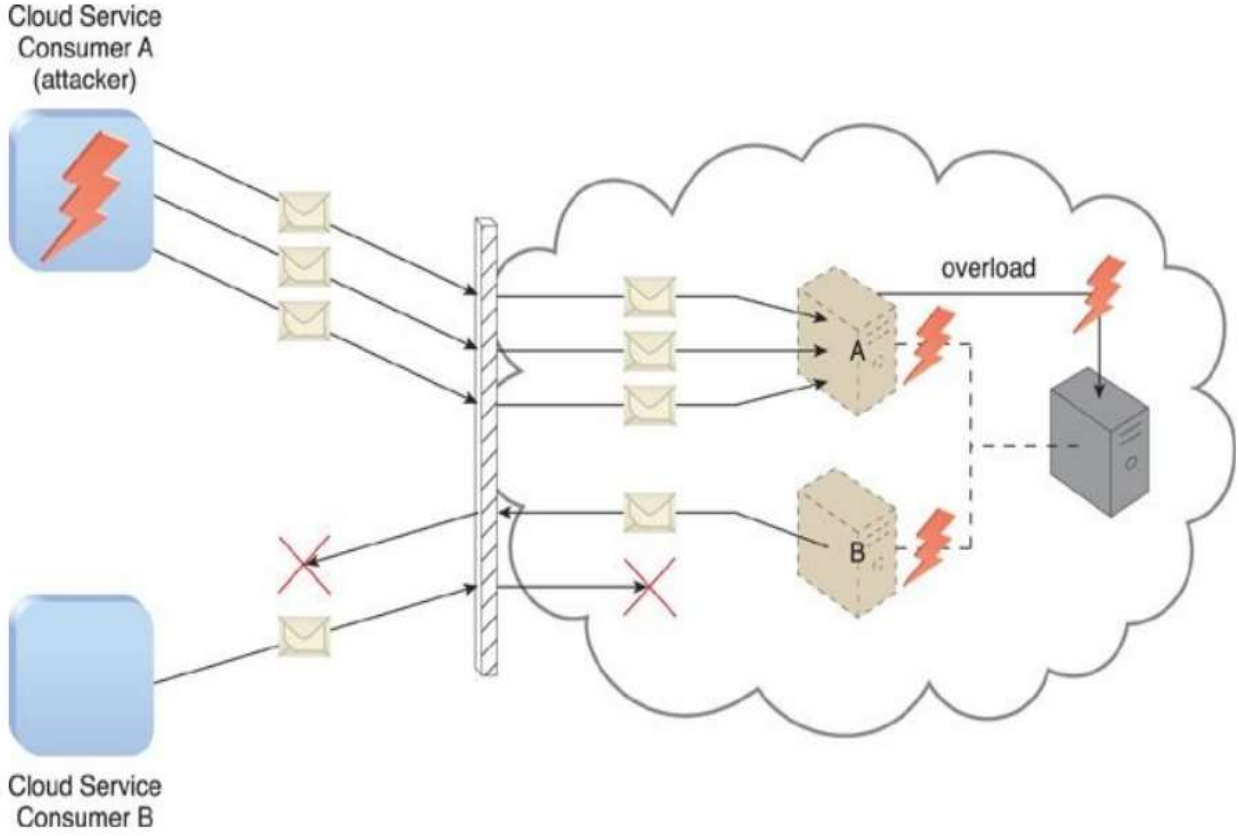
viii. **Denial of Service (DoS)**: Malicious Intermediaries can disrupt cloud services by overloading them with traffic or exploiting vulnerabilities.

ix. **Data Exfiltration**: Attackers can exfiltrate sensitive data, compromising confidentiality and integrity.

x. **Resource Tampering**: Modifying responses from the cloud service, attackers can deliver malicious content or malware to users.

3. Denial of Service attack

iii. Denial of Service:



Definition: A DoS attacks a service or network with a flood of traffic, causing slow or completely unresponsive.
Motivation: Attackers use DoS attacks for various reasons, including financial gain, revenge, competition, or simply to disrupt operations.
Types: There are two of DoS attacks - network-based and application-based
DDoS Attacks: Distributed Denial-of-Service (DDoS) DDoS attacks use many devices to launch DoS attacks. making mitigation more challenging.
Detection and Mitigation: Uses tools and services, such as firewalls, intrusion detection systems (IDS), and content delivery networks (CDNs), are used to detect and mitigate DoS attacks.

Example:

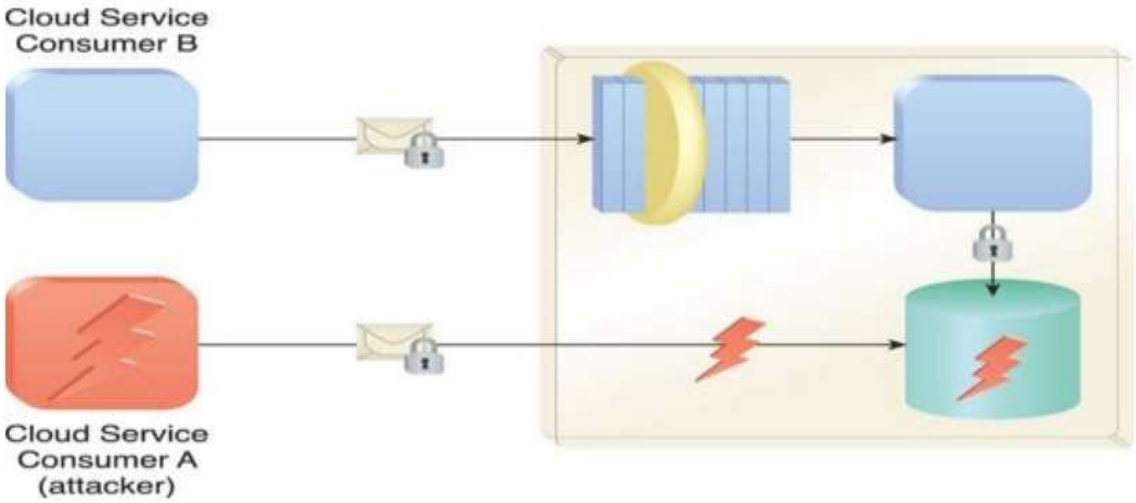
A Denial of Service (DoS) attack is a type of cyber attack that aims to overload IT resources to the point where they cannot function properly. Here is an example of how a DoS attack can be carried out in cloud computing:

1. Suppose a company is using a cloud-based web application to provide services to its customers. An attacker can launch a DoS attack on the web application by sending a large number of requests to the application, overwhelming its capacity to handle incoming traffic. This can cause the application to slow down or crash, making it unavailable to legitimate users.
2. For example, in 2016, a massive DoS attack was launched on the DNS provider Dyn, which caused widespread internet outages across the United States and Europe. The attack was

carried out using a botnet of compromised Internet of Things (IoT) devices, which flooded Dyn's servers with traffic, overwhelming their capacity to handle incoming requests.

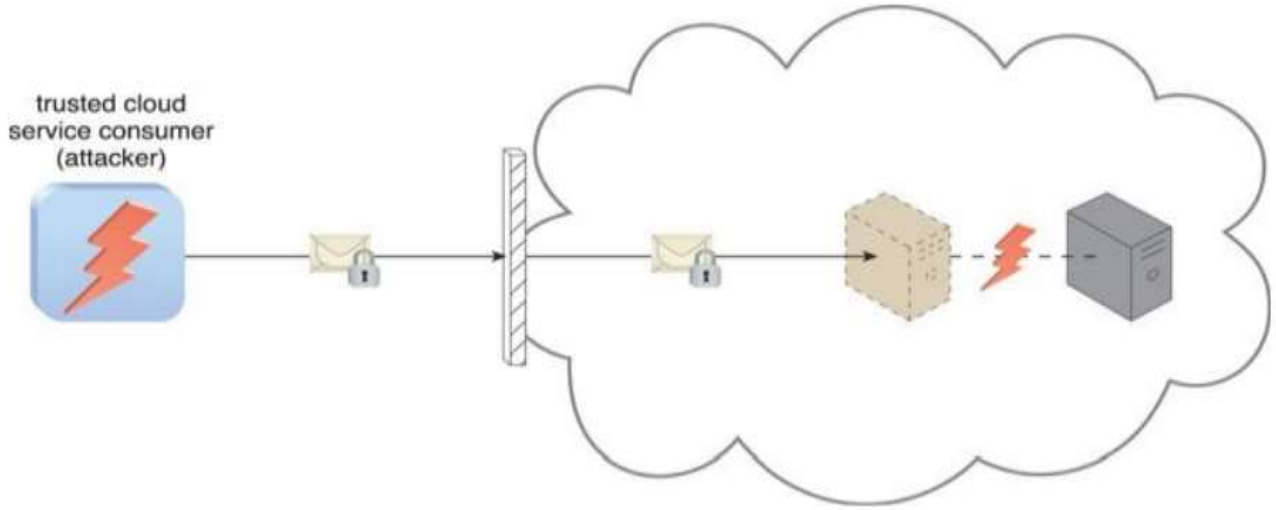
In cloud computing, DoS attacks can also be launched by artificially increasing the workload on cloud services, overloading the network with traffic, or sending multiple cloud service requests designed to consume excessive memory and processing resources. Successful DoS attacks can produce server degradation and/or failure, making cloud services unavailable to legitimate users.

iv. Insufficient Authorization:



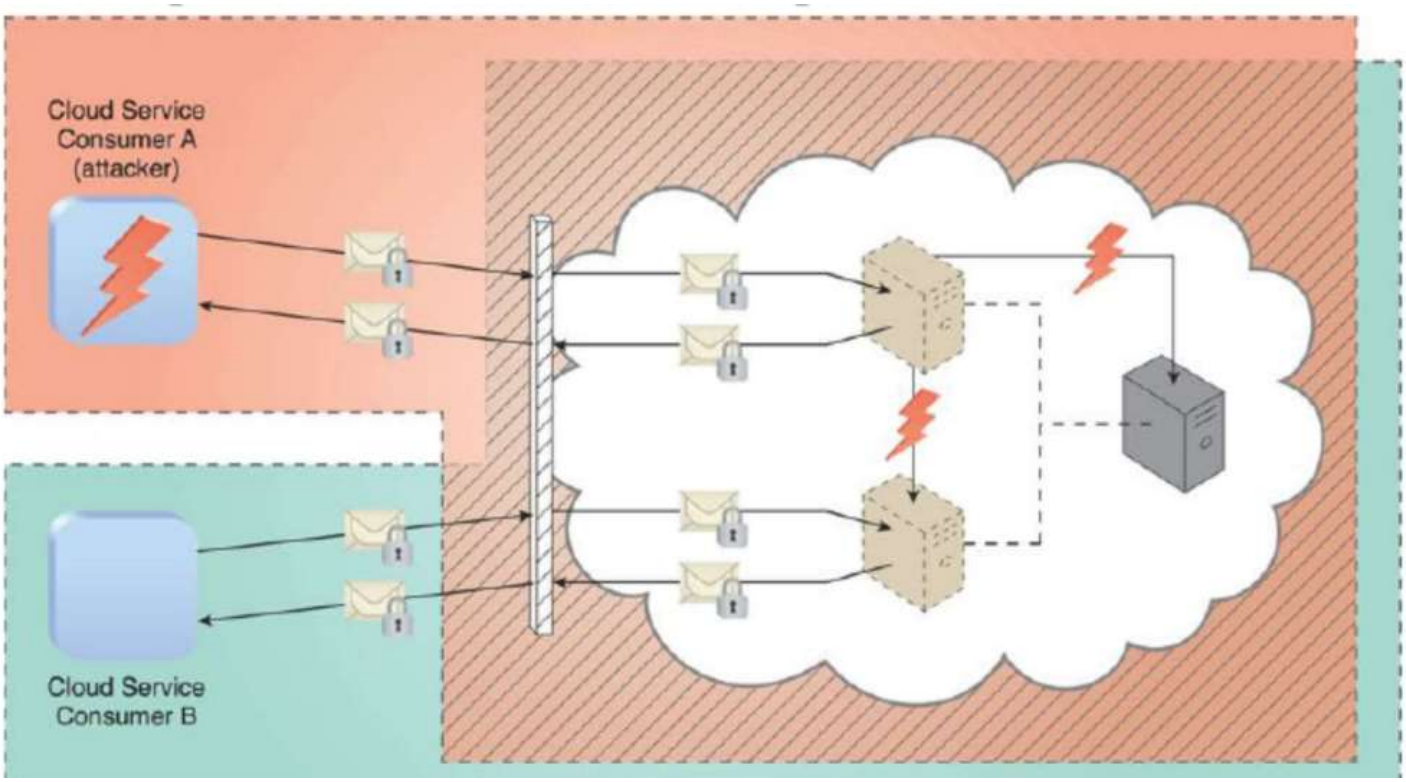
i. **Definition:** Insufficient authorization means not having permission to do something.
ii. **Access Control:** Only authorized people or systems should be able to access data or functionalities.
iii. **Authentication vs. Authorization:** Authentication checks who you are, authorization checks what you can do.
iv. **Common Scenarios:** Insufficient authorization can lead to unauthorized access or unauthorized changes.
v. **Security Risks:** Insufficient authorization is a major security risk. It can lead to data breaches, information leakage, and system vulnerabilities.
vi. **Testing:** Vulnerability assessments testing is helps to identify authorization weaknesses.
vii. **Compliance:** industries and regulations requires strict authorization controls to protect sensitive information.
viii. **Continuous Monitoring:** Regular monitoring and auditing helps to detect and fix authorization issues quickly.

v. Virtualization Attack:



i. **Definition:** Virtualization attacks target virtualized systems and their resources.
ii. **Targets:** Virtualization attacks can target hypervisors, VMs, and containers.
iii. **Attack Vectors:** Attackers can exploit hypervisor vulnerabilities, escape VMs, or attack virtualized networks.
iv. **Goals:** Virtualization attacks aim to steal data, disrupt services, or control the host system.
v. **Isolation:** Strong VM isolation and secure virtualization layer minimize attack surface.
vi. **Importance:** Virtualization security is Importance for cloud and data center security.

vi. Overlapping Trust Boundaries:



i. **Resource Sharing:** Overlapping trust boundaries occur in shared cloud resources.
ii. **Security Domains:** Entities have different security policies, controls, and trust levels.
iii. **Potential Risks:** Overlapping trust boundaries can be risky, as entities may have different security requirements or vulnerabilities.
iv. **Data Segmentation:** Proper segmentation is important to prevent data leakage or unauthorized access across trust boundaries.
v. **Access Control:** Access controls should be enforced to restrict interactions between security domains.
vi. **Encryption:** Data encryption can protect information even if it crosses trust boundaries.
vii. **Audit and Monitoring:** Continuous monitoring is essential to detect and respond to security breaches or policy violations.
viii. **Identity and Access Management (IAM):** Robust IAM systems manage access, across trust boundaries.

6. Amazon Structured Storage

- **Preconfigured EC2 AMIs**
  1. **Description:** Predefined templates with various database management systems (DBMS).
  2. **Instance Creation:** EC2 instances from these AMIs can be paired with EBS volumes for storage.
  3. **Available AMIs:** IBM DB2, Microsoft SQL Server, MySQL, Oracle, PostgreSQL, Sybase, Vertica.
  4. **Pricing:** Hourly pricing based on the EC2 cost model.
  5. **Administration:** Requires users to handle most administrative tasks, including configuration and maintenance.
- **Amazon RDS**
  1. **Description:** Managed relational database service built on EC2 infrastructure.
  2. **Advantages:** Eliminates concerns about storage configuration, high availability, and patching.
  3. **Services:** Provides automatic backups, snapshots, point-in-time recoveries, and replication.
  4. **Engines:** Supports MySQL and Oracle as relational database engines.
  5. **Advanced Features:**
     - **Multi-AZ Deployment:** Ensures failover infrastructure with synchronized copies in different availability zones.
     - **Read Replicas:** Increases performance for read-heavy applications with dedicated database copies.
  6. **Pricing:** Detailed in Table 9.4 for on-demand instances; offers reserved instances for long-term use.
  7. **Management:** AWS handles database management, patching, and offers simplified elastic server management.
  8. **Optimal for:** Applications migrating to AWS infrastructure, especially those based on Oracle and MySQL engines, requiring a scalable solution.
- **SimpleDB Overview**
  1. **Domain Structure:** Uses domains comparable to tables in the relational model.
  2. **Item Representation:** Items are key-value pairs, allowing varied column structures.
  3. **Domain Limits:** Each domain can grow up to 10 GB, with a default limit of 250 domains per user.
  4. **Client Operations:** Users can create, delete, modify, and snapshot domains.
  5. **Data Manipulation:** Supports batch insertion, deletion, and query operations on items and attributes.

6. **Querying Capabilities:** The SELECT clause offers various test operators for querying data.
    - Operators include =, !=, <, >, <=, >=, like, not like, between, is null, is not null, every().
    - Example query: **SELECT * FROM domain_name WHERE every(attribute_name) = 'value'**.
7. **Cross-Domain Queries:** SELECT operator extends queries beyond a single domain for efficient data retrieval.
8. **Scalability and Consistency:** Implements a relaxed constraint model for scalability, resulting in eventually consistent data.
9. **Conditional Operations:** Allows clients to express conditional insertions or deletions.
    - Operations executed only if the specified condition is verified.
10. **Pricing Model:** Charges for data transfer, stored data, and machine usage.

- First 25 SimpleDB instances per month are free; beyond, incurs an hourly charge ($0.140/hour in the U.S. East region).
- Data transfer within AWS network is not charged.

11. **Comparison with S3:** SimpleDB designed for fast access to semi-structured small objects, while S3 is more cost-effective for storing large objects.

7. Architecture of Google AppEngine

Architecture and core concepts

AppEngine is a platform for developing scalable applications accessible through the Web (see Figure 9.2). The platform is logically divided into four major components: infrastructure, the runtime environment, the underlying storage, and the set of scalable services that can be used to develop applications.

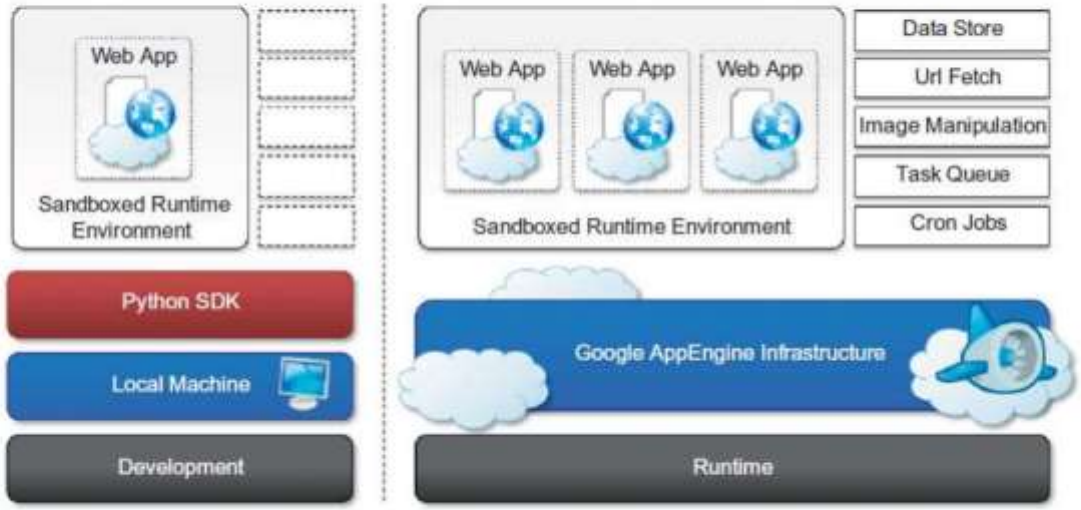


FIGURE 9.2

Google AppEngine platform architecture.

- The architecture of Google AppEngine consists of four major components: infrastructure, runtime environment, underlying storage, and scalable services.

1. **Infrastructure:** App Engine uses Google's datacenters, employing multiple servers to efficiently handle user requests. The infrastructure dynamically allocates resources and redirects requests based on load evaluation[1].

2. **Runtime Environment:** This represents the execution context for hosted applications, starting *when* the request handler begins and terminating upon completion. The *runtime environment* is *sandboxed* and provides scalability for executing applications[1].

3. **Underlying Storage:** App Engine offers a distributed and scalable storage system known as DataStore. It allows efficient management and access to stored data[4].

4. **Scalable Services:** App Engine provides various scalable services such as *compute, storage, networking, application connectivity, and access control*. Developers can leverage these services for building applications that naturally scale on the platform[1][4].

5. **Overall:** Google App Engine leverages Google's infrastructure, providing a scalable and efficient platform for developing and hosting web applications[1].

8. AppFabric of Microsoft Azure

- **Middleware in Azure:** AppFabric is a middleware in Microsoft Azure, facilitating the development, deployment, and management of cloud applications.
- **Core Infrastructure:**
  - Optimized Infrastructure: AppFabric supports scaling out and high availability with sandboxing and multitenancy capabilities.
  - State Management: Manages application data storage and accessibility.
  - Dynamic Address Resolution: Offers dynamic resolution and routing for seamless communication between distributed application components.
- **Communication:**
  - Service Bus: AppFabric includes a service bus supporting various communication protocols and patterns.
  - Communication Models: Facilitates publish-subscribe, full-duplex, point-to-point, peer-to-peer, unicast, multicast, and asynchronous messaging.
- **Authentication and Authorization:**
  - Identity Management: Integrates authentication providers (Active Directory, Windows Live, Google, Facebook) into a unified framework.
  - Access Control: Enables encoding access control rules and defining policies for users and groups, ensuring application security.
- **Data Access:**
  - Caching: Offers caching and distributed caching for improved performance and reduced backend load.
  - SQL Azure: Provides a distributed database middleware for deploying and managing databases in the Azure Cloud.