Unit-5
Service Management

-Dr. Radhika V. Kulkarni

Associate Professor, Dept. of Computer Engineering,

Vishwakarma Institute of Technology, Pune.

DISCLAIMER

This presentation is created as a reference material for the students of TY-Comp. Engg., VIT (AY 2022-23 Sem-2).

It is restricted only for the internal use and any circulation is strictly prohibited.

Syllabus

Unit-V Service Management

[CO5:PO1,PO2,PO3,PO4,PO5,PO9,PO11,PSO2-Strength 3,3,1.3,3,1,2,3]

Service Level Agreements (SLAs), Billing and accounting, Billing in GCP Cloud Security: Introduction to security in the cloud, the shared security model, Encryption options, Authentication and authorization with Cloud IAM, Identify Best Practices for Authorization using Cloud IAM, Introduction to configuration and management tools Ansible, Architecture of DevOps. [4 Hrs]

Service Level Agreements (SLA)

Service Level Agreements (SLA)

- A **service-level agreement** (**SLA**) is defined as an official commitment that prevails between a service provider and a client. Particular aspects of the service – quality, availability, responsibilities – are agreed between the service provider and the service user
- The most common component of SLA is that the services should be provided to the customer as agreed upon in the contract.
- An SLA specifies the details of the service to be provided in terms of metrics agreed upon by all parties, and incentives and penalties for meeting and violating the expectations, respectively.
- SLAs establish customer expectations with regard to the service provider's performance and quality in a number of ways. Some metrics that SLAs may specify include:
  - Availability and uptime - the percentage of the time services will be available.
  - Specific performance benchmarks to which actual performance will be periodically compared.
  - Application response time.
  - The schedule for notification in advance of network changes that may affect users.
  - Help desk response time for various classes of problems.
  - Usage statistics that will be provided.

Service level agreements are also defined at different levels:

1. **Customer-based SLA:** An agreement with an individual customer group, covering all the services they use. E.g., an SLA between a supplier (IT service provider) and the finance department of a large organization for the services such as finance system, payroll system, billing system, procurement/purchase system, etc.

2. **Service-based SLA:** An agreement for all customers using the services being delivered by the service provider. E.g., A mobile service provider offers a routine service to all the customers and offers certain maintenance as a part of an offer with the universal charging.

3. **Multilevel SLA**: The SLA is split into the different levels, each addressing different set of customers for the same services, in the same SLA.

- **Corporate-level SLA**: Covering all the generic service level management (often abbreviated as SLM) issues appropriate to every customer throughout the organization. These issues are likely to be less volatile and so updates (SLA reviews) are less frequently required.

- **Customer-level SLA**: covering all SLM issues relevant to the particular customer group, regardless of the services being used.

- **Service-level SLA**: covering all SLM issue relevant to the specific services, in relation to this specific customer group.

Key Components of a SLA

1. Service Level Parameter : Describes an observable property of a service whose value is measurable.

2. Metrics : These are definitions of values of service properties that are measured from a service-providing system or computed from other metrics and constants. Metrics are the key instrument to describe exactly what SLA parameters mean by specifying how to measure or compute the parameter values.

3. Function : A function specifies how to compute a metric‘s value from the values of other metrics and constants. Functions are central to describing exactly how SLA parameters are computed from resource metrics.

4. Measurement directives: These specify how to measure a metric.

Types of SLAs from the perspective of application hosting

1. Infrastructure SLA :
   - The infrastructure provider manages and offers guarantees on availability of the infrastructure, namely, server machine, power, network connectivity, and so on.
   - Enterprises manage themselves, their applications that are deployed on these server machines.
   - The machines are leased to the customers and are isolated from machines of other customers. In such dedicated hosting environments, a practical example of service-level guarantees offered by infrastructure providers.

2. Application SLA:
   - In the application co-location hosting model, the server capacity is available to the applications based solely on their resource demands.
   - Hence, the service providers are flexible in allocating and de-allocating computing resources among the co-located applications. Therefore, the service

Life Cycle of SLA

- Each SLA goes through a sequence of steps starting from identification of terms and conditions, activation and monitoring of the stated terms and conditions, and eventual termination of contract once the hosting relationship ceases to exist.

- Such a sequence of steps is called SLA life cycle and consists of the following five phases:

1. Contract definition:
   - Generally, service providers define a set of service offerings and corresponding SLAs using standard templates.
   - These service offerings form a catalog. Individual SLAs for enterprises can be derived by customizing these base SLA templates.

2. Publishing and discovery:
   - Service provider advertises these base service offerings through standard publication media, and the customers should be able to locate the service provider by searching the catalog.
   - The customers can search different competitive offerings and shortlist a few that fulfill their requirements for further negotiation.

Phases of SLA Life Cycle (cont..)

3. Negotiation:

- Once the customer has discovered a service provider who can meet their application hosting need, the SLA terms and conditions needs to be mutually agreed upon before signing the agreement for hosting the application.
- For a standard packaged application which is offered as service, this phase could be automated. For customized applications that are hosted on cloud platforms, this phase is manual.
- The service provider needs to analyze the application‘s behavior with respect to scalability and performance before agreeing on the specification of SLA.
- At the end of this phase, the SLA is mutually agreed by both customer and provider and is eventually signed off.

4. Operationalization:

- SLA operation consists of SLA monitoring, SLA accounting, and SLA enforcement.
- SLA monitoring involves measuring parameter values and calculating the metrics defined as a part of SLA and determining the deviations.
- On identifying the deviations, the concerned parties are notified. SLA accounting involves capturing and archiving the SLA adherence for compliance.
- As part of accounting, the application‘s actual performance and the performance guaranteed as a part of SLA is reported.

5. De-commissioning

- SLA decommissioning involves termination of all activities performed under a particular SLA when the hosting relationship between the service provider and the service consumer has ended.
- SLA specifies the terms and conditions of contract termination and specifies situations under which the relationship between a service provider and a service consumer can be considered to be legally ended.

SLA Management in Cloud

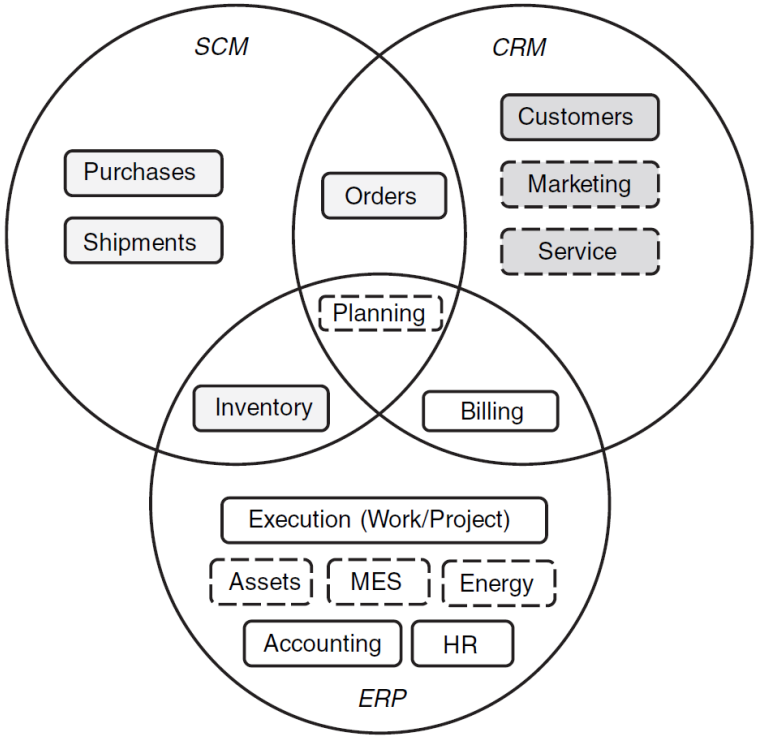
- **SLA management of applications hosted on cloud platforms involves five phases:**
  1. Feasibility
  2. On-boarding
  3. Pre-production
  4. Production
  5. Termination

Billing and Accounting

For more details refer to:
https://services.google.com/fh/files/misc/google-cloud-security-foundations-guide.pdf

Enterprise Application

- A very high level ‘core data model’ for an enterprise:
- Information is maintained in cross-enterprise applications such as
  - Enterprise Resource Planning (ERP):This includes financial accounting, dealing with costing of all work, assets and other inputs, billing and counting incoming revenue, planning all operations, as well as managing employees.
  - Customer Relationship Management (CRM): It focuses on attracting customers, through sales and marketing, and servicing.
  - Supply Chain Management (SCM): It deals with taking sales orders, issuing purchase orders, controlling the inventory of raw and finished goods, and shipping them to customers.



Billing in GCP

- Cloud Billing is a collection of tools that help you track and understand your Google Cloud spending, pay your bill, and optimize your costs.
- It is a mechanism to enhance governance and security by monitoring Google Cloud usage and alerting on unexpected consumption.
- A cloud Billing account defines who pays for a given set of Google Cloud resources.
- To use Google Cloud services, you must have a valid Cloud Billing account, and must link it to your Google Cloud projects.
- Each project is associated with a billing account. Your project's Google Cloud usage is charged to the linked Cloud Billing account.
- You must have a valid Cloud Billing account even if you are in your free trial period or if you only use Google Cloud resources that are covered by the Google Cloud Free Tier.

GCP FinOps Best Practices

- The billing setup within the security foundation is aligned to the best practices defined as below:

Cloud FinOps best practice	Implementation in security foundation
Use a single billing account.	A single billing account is used across all projects in the security foundation.
Devise internal chargeback mechanisms to allocate costs.	Projects are created with metadata that includes information that can be used for internal chargeback.
Apply labels at project creation using automation.	The Cloud Build deployment pipeline applies the metadata at project creation.
Ensure reports can provide data on differing dimensions.	Billing information is exported to BigQuery, which can be used to create multi-dimensional reports.
Define budgets based on the financial budget that you wish to track.	Budgets are defined during project creation through the Cloud Build pipeline.
Define alerts based on budgets.	Alerts are defined during project creation through the Cloud Build pipeline.

Billing Alerts, Exports and Chargeback

- Billing Alerts:
  - Users can set budgets at a project level or organization level, either as a fixed amount to suit steady-state expenditure, or as a percentage of the previous month's spending to suit variable costs.
  - Billing alerts can be applied to different scopes within the organization.
  - Billing alerts are used on a per-project basis with thresholds set at 50%, 75%, 90%, and 95%.

- Billing exports and chargeback:
  - The Cloud console has extensive cost management tools that you can use to view and forecast costs in a variety of formats.
  - The cost management tools are augmented by exporting all billing records to a BigQuery dataset in the Billing project.
  - The export needs to be enabled through the Cloud console. The exported billing records include the project label metadata that's assigned to the project during project creation.
  - As specified, each project has an associated billing code and contact points that can be used for chargeback.
  - A simple SQL query on the exported dataset can be used to provide billing information for each business unit.

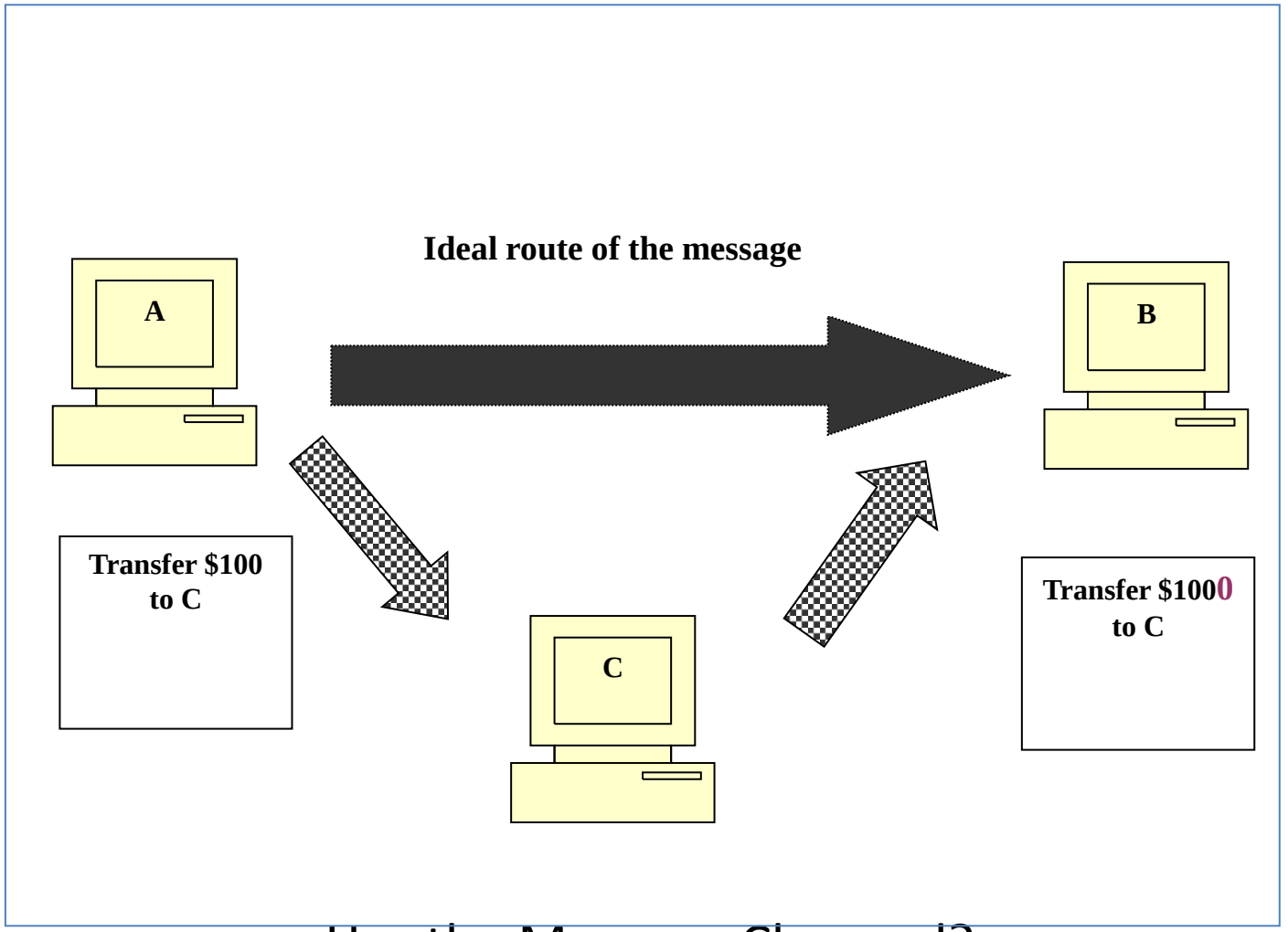
Security in Cloud

CIA

- Each and every security process, layer or software must implement and cover the CIA triad.
  - **C-Confidentiality**: It refers to the process to ensure that information sent between two parties is confidential between them only and not viewed by anyone else.
  - **I-Integrity**: It refers to the process to ensure that the message which is in transit must maintain its integrity i.e., the content of the message must not be changed.
  - **A-Availability**: The systems available for fulfilling requests must be available all the time.

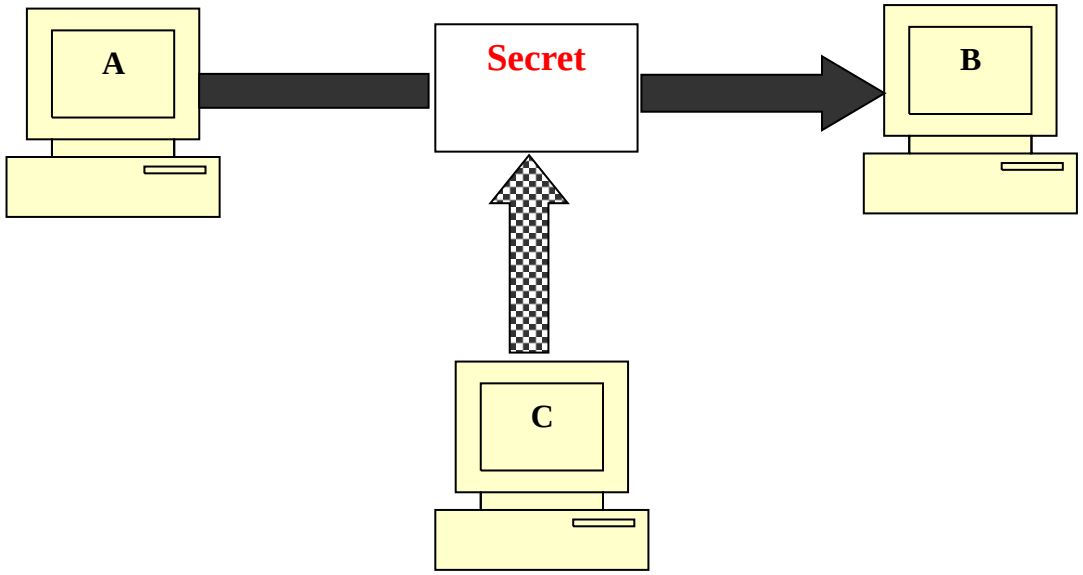
- Along with these, some important parameters are described below:
  - **Authentication**: the process of confirming someone's identity with the supplied parameters like username and password.
  - **Authorization**: the process of granting access to a resource to the confirmed identity based on their permissions.
  - **Non-Repudiation**: a process to make sure that only the intended endpoint has sent the message and later cannot deny it.

Integrity:



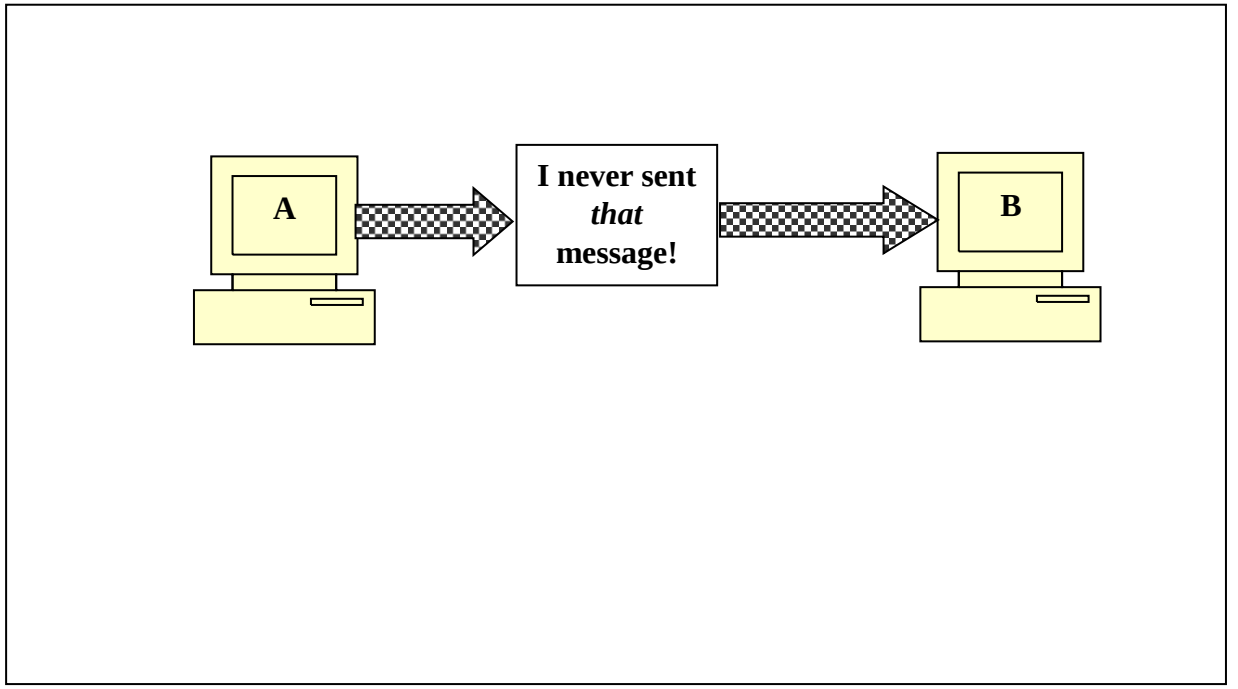
Has the Message Changed?

Confidentiality:



Has someone seen it?

Non-Repudiation:

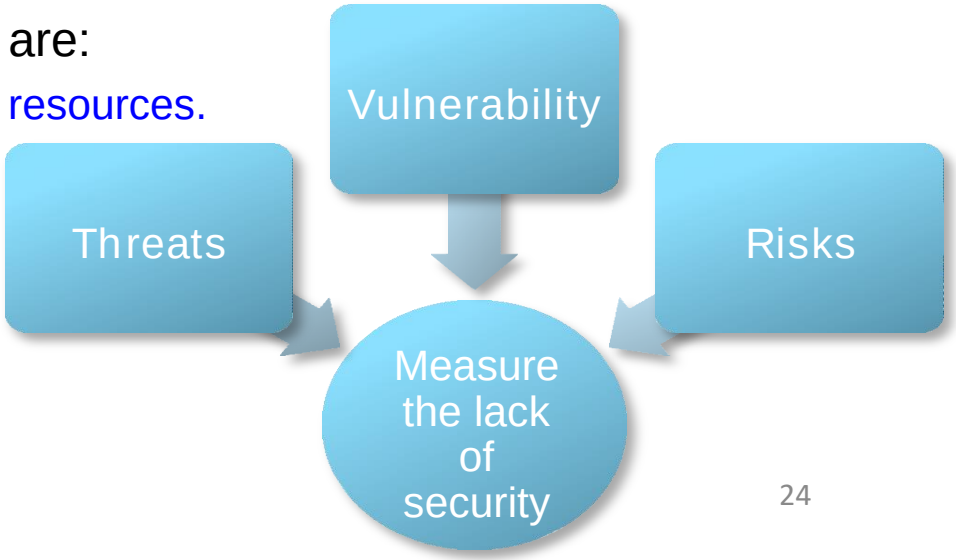


A sends a message and refutes it later

Availability

- Availability is a characteristics of being accessible and usable during a specified time period.
- In typical cloud environment, the availability of cloud services can be a responsibility that is shared by the cloud provider and cloud carrier.

- **Threat**
  - A threat is a potential security violation that can challenge defenses in an attempt to breach privacy and/or cause harm.
  - Both manually and automatically instigated threats are designed to exploit known weaknesses, also referred as vulnerabilities**.**
- **Vulnerability**
  - A vulnerability is a weakness that can be exploited either because it is protected by insufficient security controls, or because existing security controls are overcome by an attack.
  - IT recourse vulnerabilities can have a range have causes, including configuration deficiencies, security policy weaknesses, user errors, hardware or firmware flaws, software bugs and poor security architecture.
- **Risk:**
  - Risk is the possibility of loss or harm arising from performing an activity.
  - Two metrics that can be used to determine risk for an IT resources are:
    1. The probability of a threat occurring to exploit vulnerabilities in the IT resources.
    2. The expectation of loss upon the IT resource being compromised.



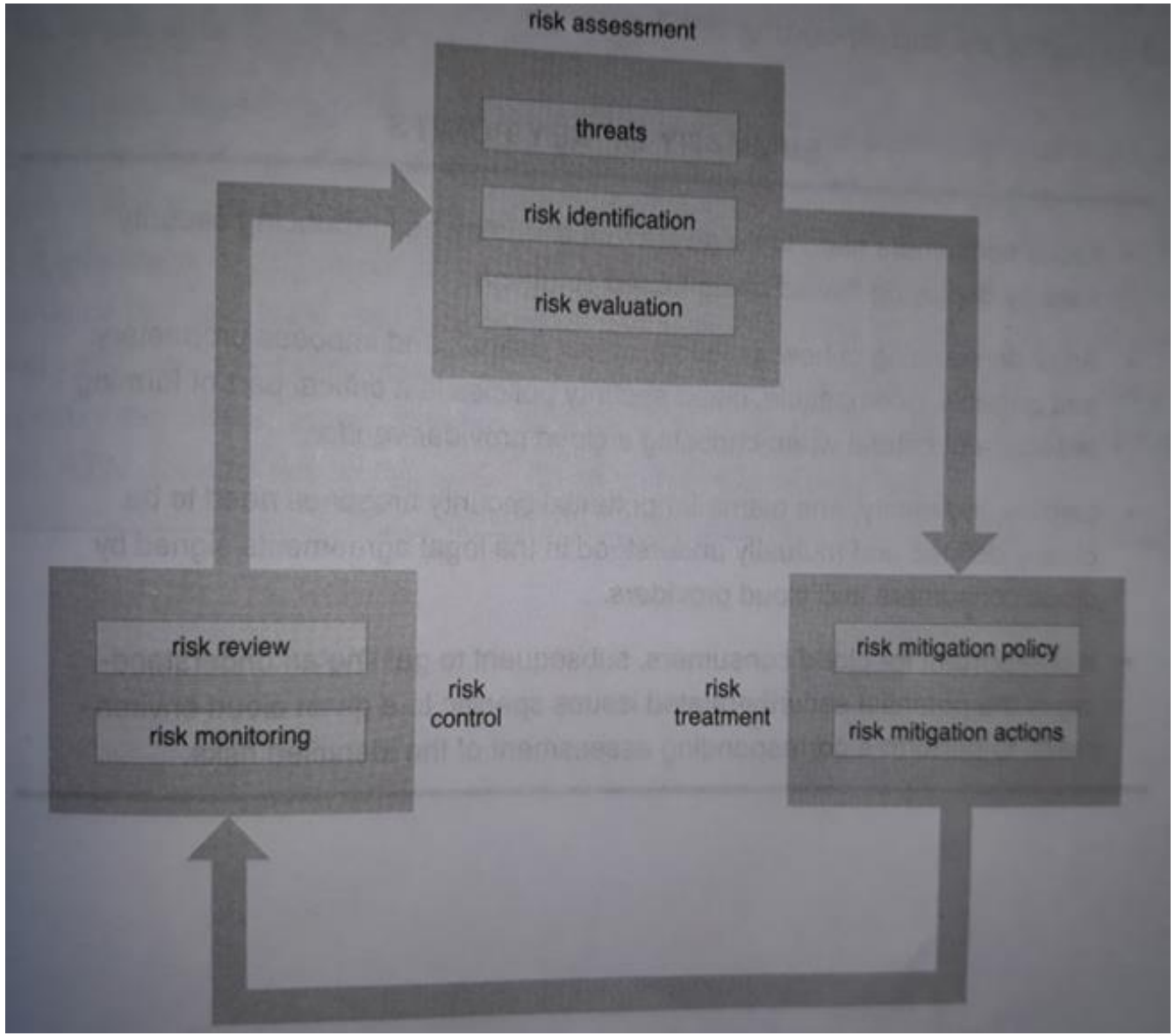
- **Security Policy Disparity**
  - When a cloud consumer places IT resources with a public provider, it may need to accept that its traditional information security approach may not be identical or even similar to that of the cloud provider.
  - This incompatibility needs to be assessed to ensure that any data or other IT assets being relocated to a public cloud are adequately protected.

- **Contracts**
  - Cloud consumers need to carefully examine contracts and SLAs put forth by cloud providers to ensure that security policies, and other relevant guarantees are satisfactory when it comes to asset security.

• Risk Management

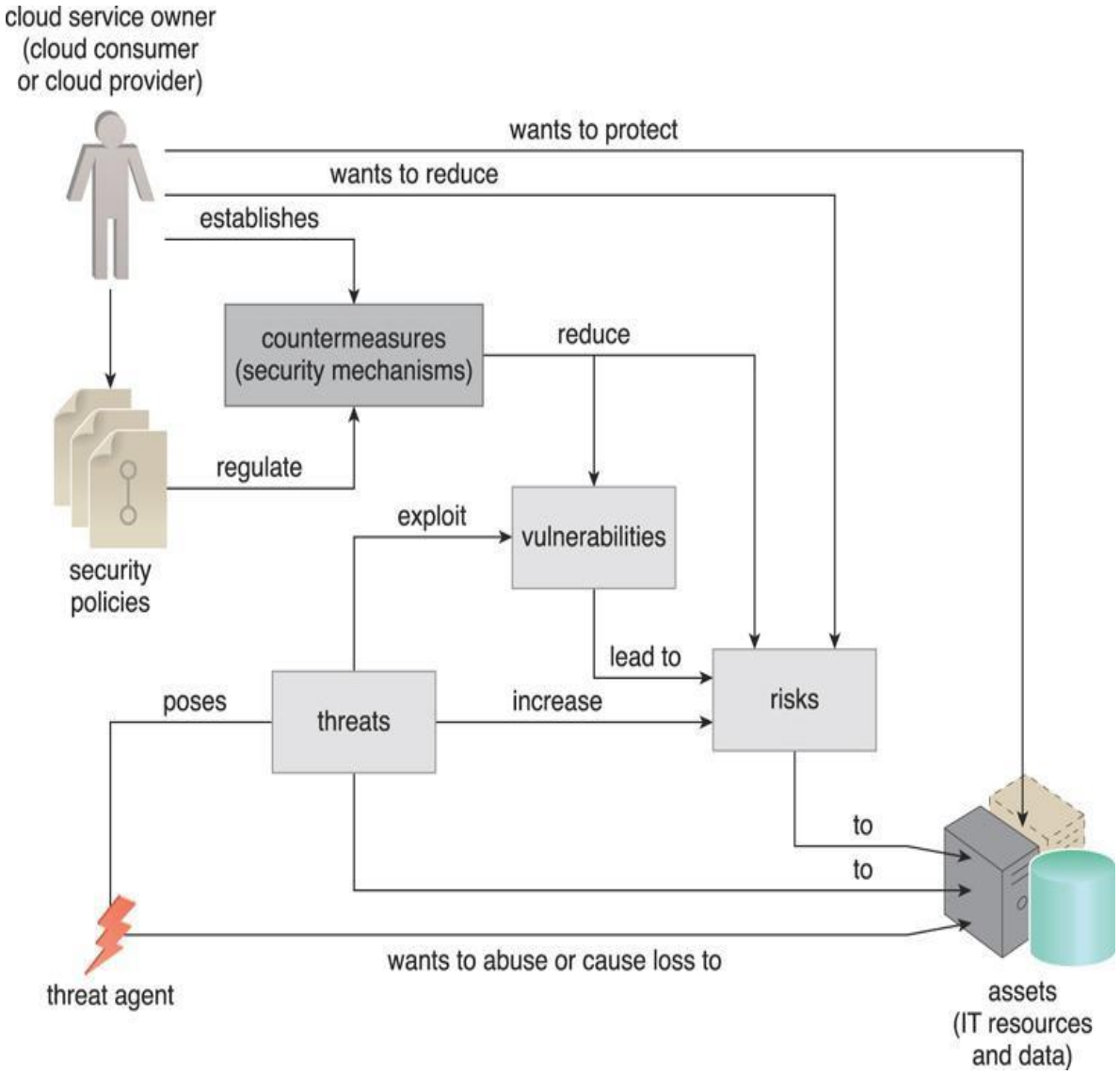
- When accessing the potential impacts and challenges pertaining to cloud adoption, cloud consumers are encouraged to perform a formal risk assessment as part of a risk management strategy.

- A cyclically executed process used to enhance strategic and tactical security, risk management is comprised of a set of coordinated activities for overseeing and controlling risks.



Threat Agents

- A *threat agent* is an entity that poses a threat because it is capable of carrying out an attack.
- Cloud security threats can originate either internally or externally, from humans or software programs.



- **Anonymous Attacker:**
- A non-trusted cloud service consumer without permissions in the cloud .
- It typically exists as an external software program that launches network-level attacks through public networks.

- **Malicious Service Agent** :
- A *malicious service agent* is able to intercept and forward the network traffic that flows within a cloud.
- It typically exists as a service agent (or a program pretending to be a service agent) with compromised or malicious logic.
- It may also exist as an external program able to remotely intercept and potentially corrupt message contents.

Cloud Security Attacks

1) Traffic Eavesdropping:

- *Traffic Eavesdropping* occurs when data being transferred to or within a cloud is passively intercepted by a malicious service agent for illegitimate information gathering purposes.

- The aim of this attack is to directly compromise the confidentiality of the data and possibly the confidentiality of the relationship between the cloud consumer and cloud provider.

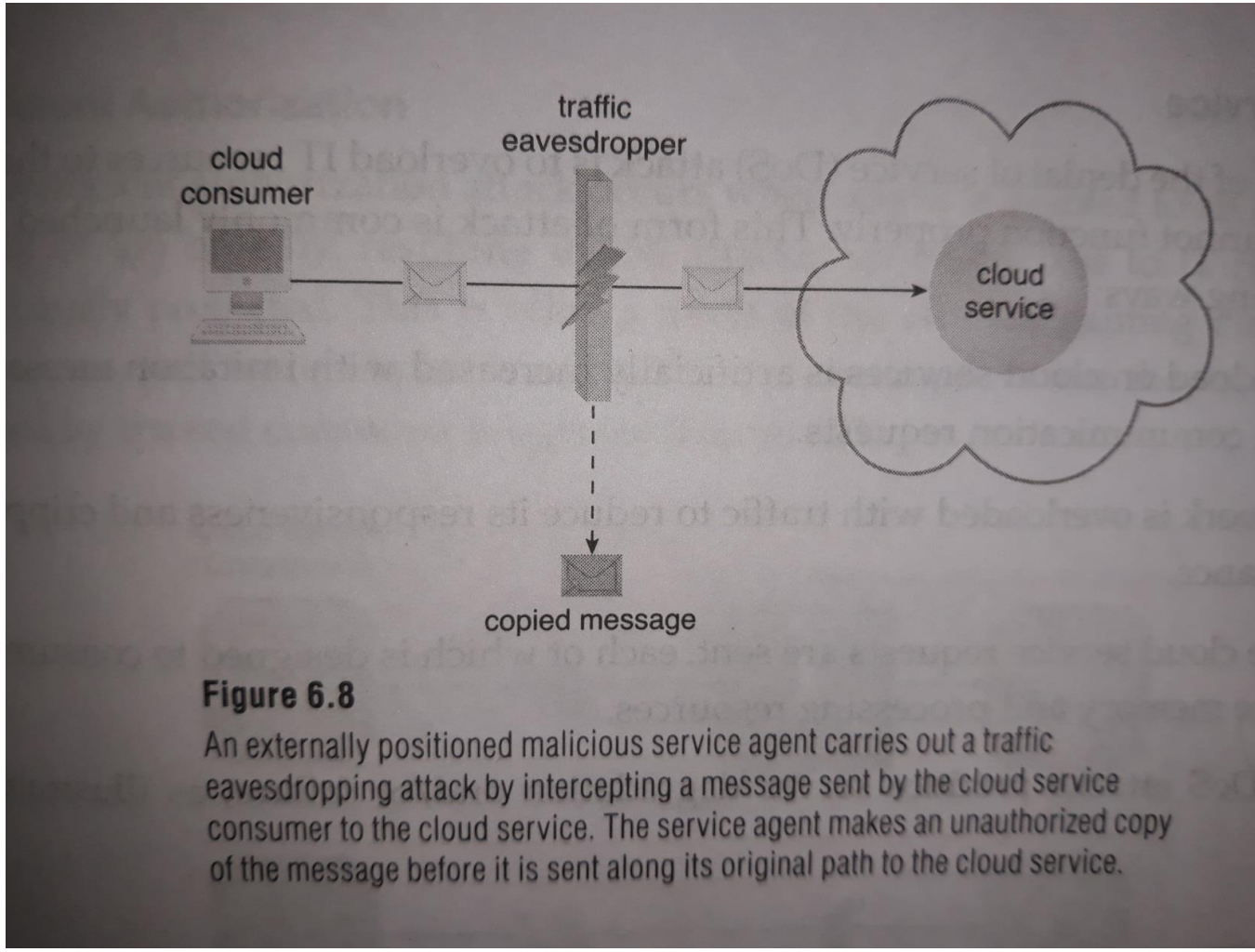


Figure 6.8
An externally positioned malicious service agent carries out a traffic eavesdropping attack by intercepting a message sent by the cloud service consumer to the cloud service. The service agent makes an unauthorized copy of the message before it is sent along its original path to the cloud service.

Cloud Security Attacks (cont..)

2) Malicious Intermediary:

- The malicious intermediary threat arises when messages are intercepted and altered by a malicious service agent, thereby potentially compromising the message's confidentiality and integrity.

- It may also insert harmful data into the message before forwarding it to its destination.

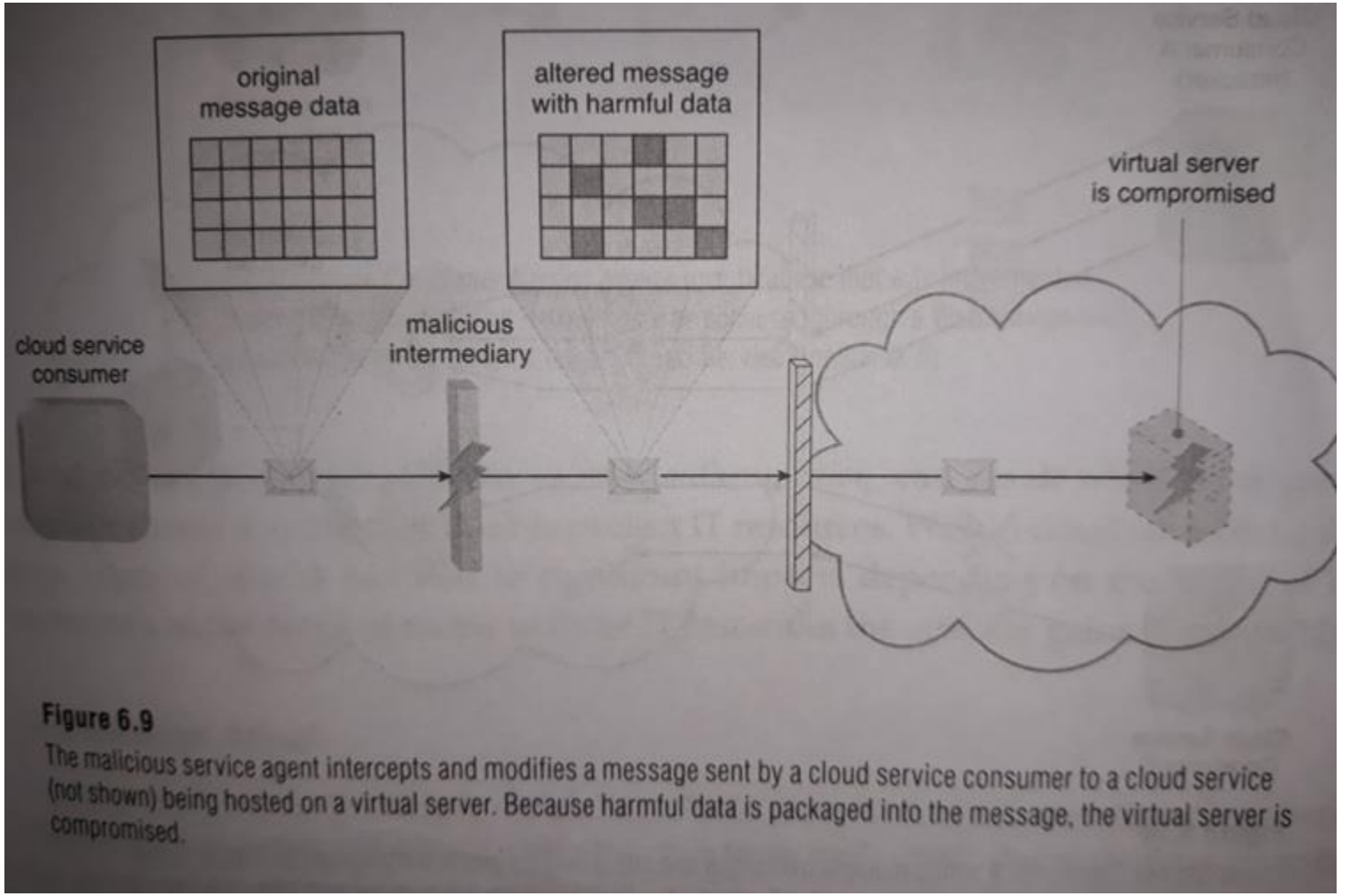


Figure 6.9
The malicious service agent intercepts and modifies a message sent by a cloud service consumer to a cloud service (not shown) being hosted on a virtual server. Because harmful data is packaged into the message, the virtual server is compromised.

Cloud Security Attacks (cont..)

3) Denial of Service (DoS):

- Its objectives is to overload IT resources to the point where they cannot function properly.

- This form of attack is commonly launched in one of the following ways:
  - The workload on cloud services is artificially increased with imitation messages or repeated communication request.
  - The network is overloaded with traffic to reduce its responsiveness and cripple its performance.
  - Multiple cloud service requests are sent, each of which is designed to consume excessive memory and processing resources.

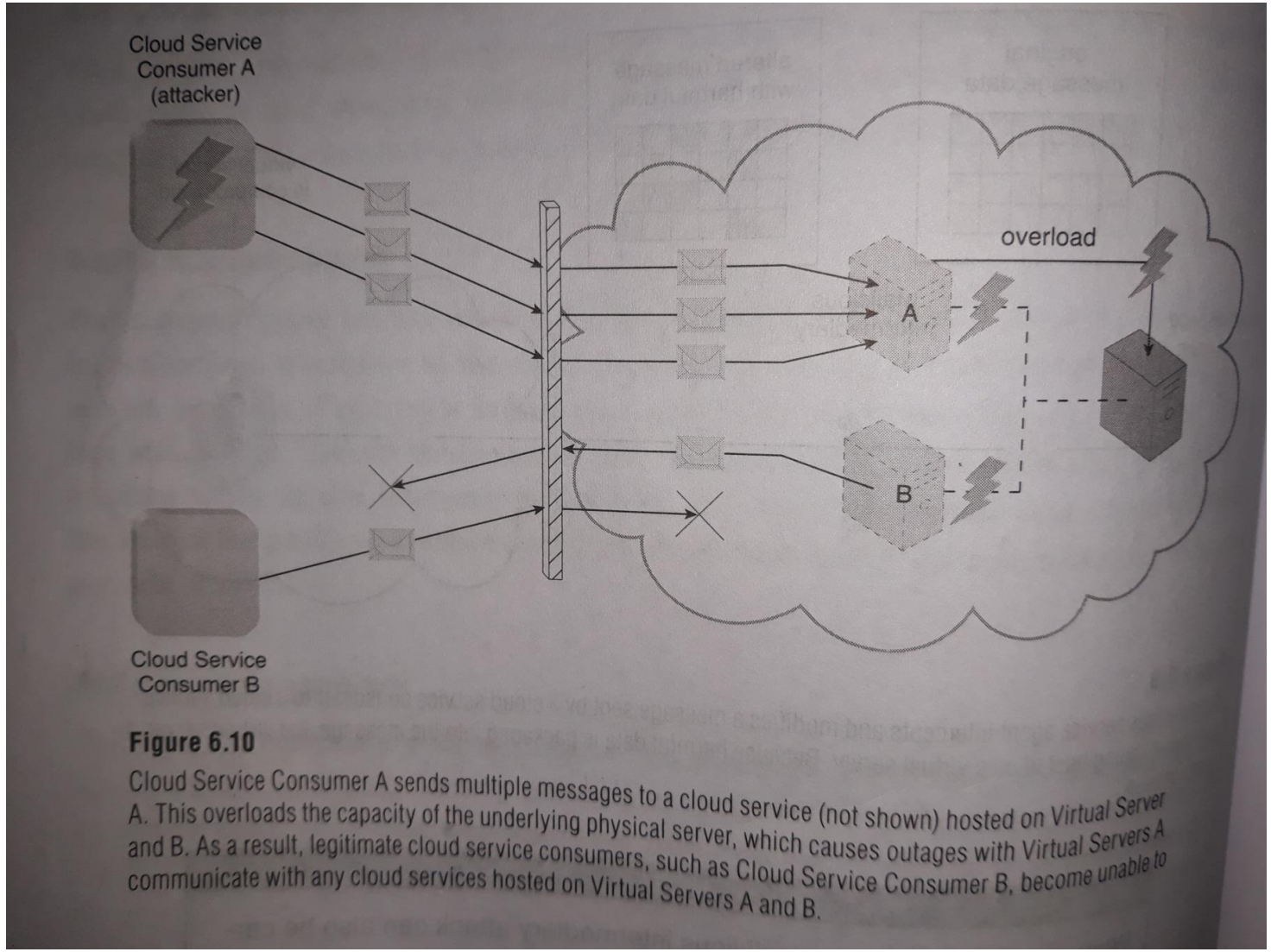
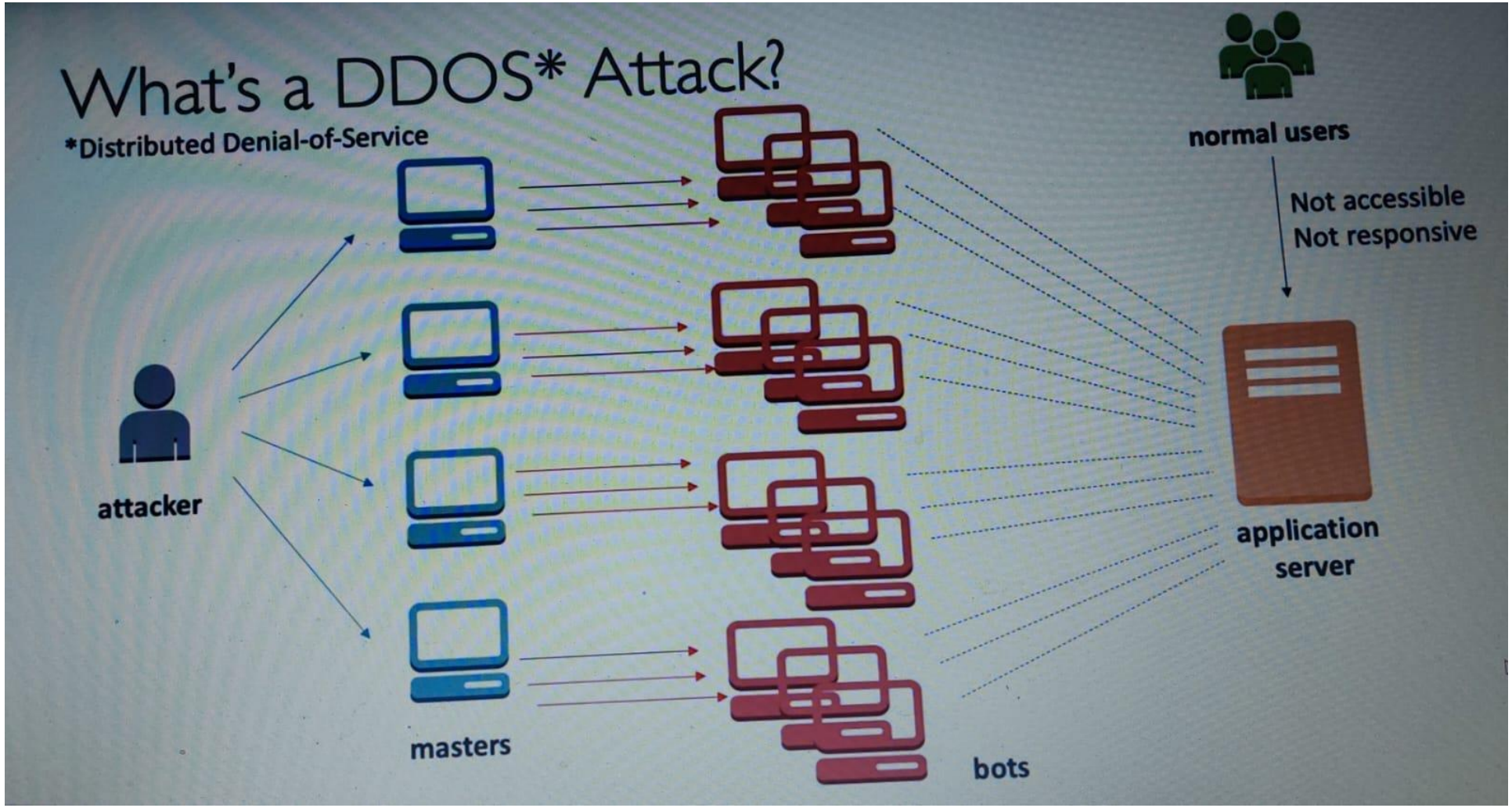


Figure 6.10

Cloud Service Consumer A sends multiple messages to a cloud service (not shown) hosted on Virtual Server A. This overloads the capacity of the underlying physical server, which causes outages with Virtual Servers A and B. As a result, legitimate cloud service consumers, such as Cloud Service Consumer B, become unable to communicate with any cloud services hosted on Virtual Servers A and B.

Cloud Security Attacks (cont..)

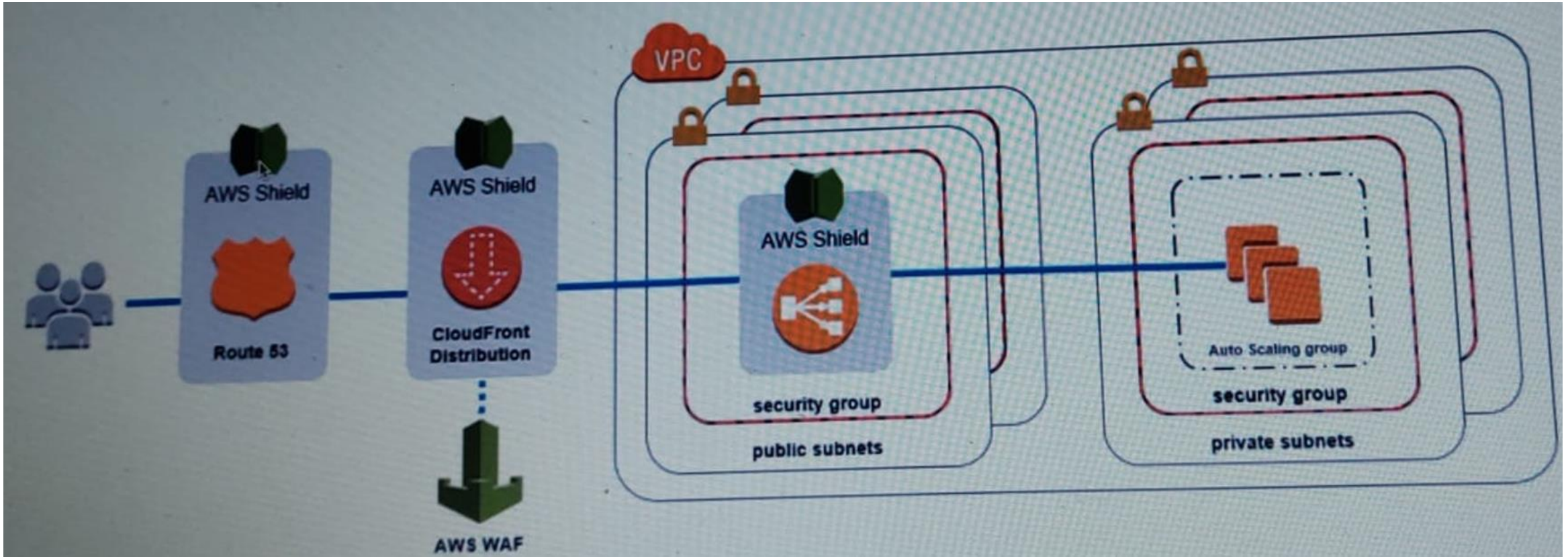
3) Denial of Service (DoS) (cont..)



Cloud Security Attacks (cont..)

3) Denial of Service (DoS) (cont..)

Sample Reference Architecture for DDoS Protection:



Cloud Security Attacks (cont..)

4) Insufficient Authorization:

- This attack occurs when access is granted to an attacker erroneously or too broadly, resulting in the attacker getting access to IT resources that are normally protected.

- This is often a result of the attacker gaining direct access to It resources that were implemented under the assumption that they would only be accessed by trusted consumer program.

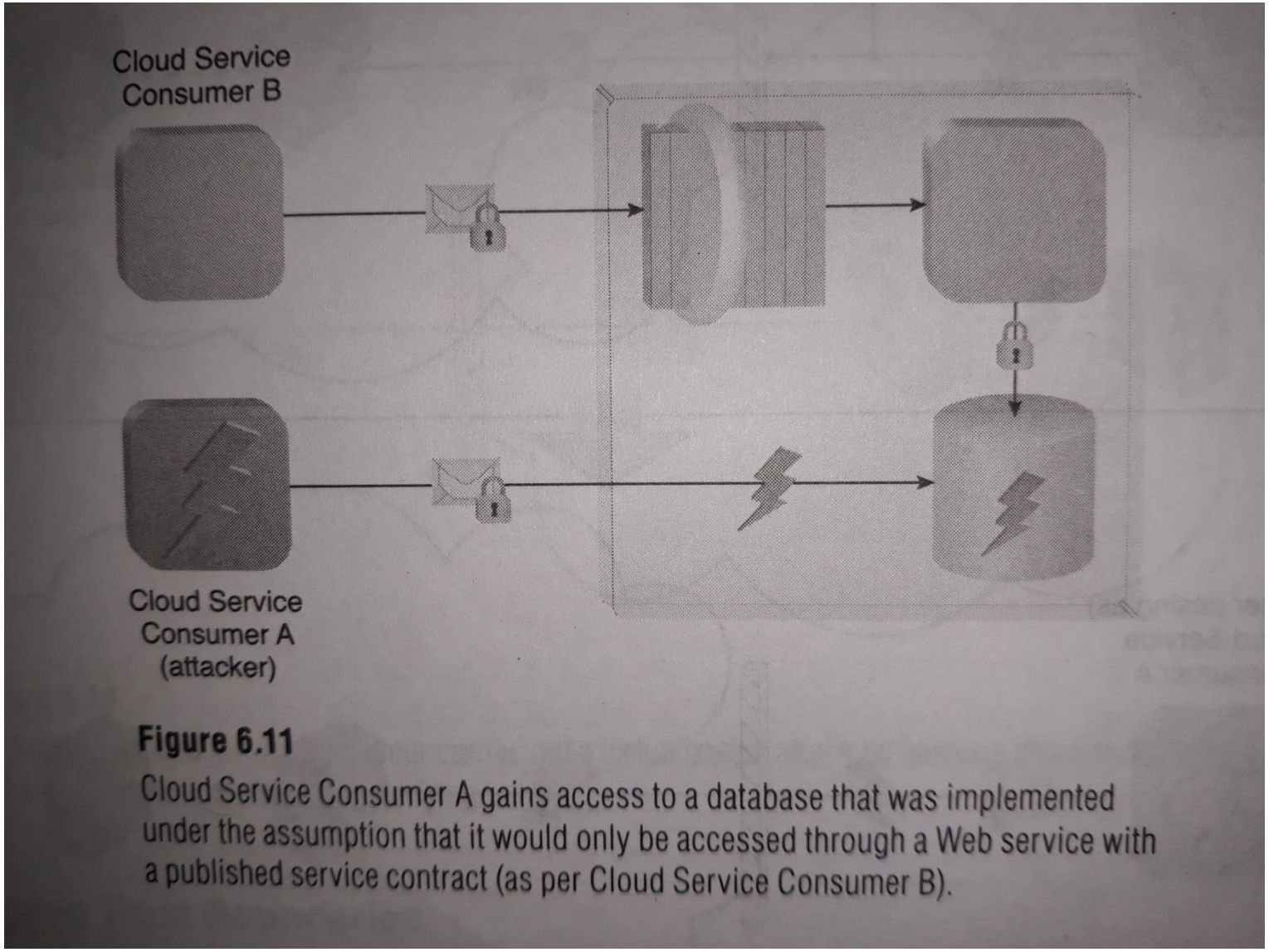


Figure 6.11
Cloud Service Consumer A gains access to a database that was implemented under the assumption that it would only be accessed through a Web service with a published service contract (as per Cloud Service Consumer B).

Cloud Security Threats and Attacks (cont..)

5) Virtualization Attack:

- Virtualization provides multiple cloud consumers with access to IT resources that share underlying hardware but are logically isolated from each other.

- Cloud providers grant cloud consumers administrative access to virtualized IT resources, there is an inherent risk that cloud consumers could abuse this access to attack the underlying physical IT resources.

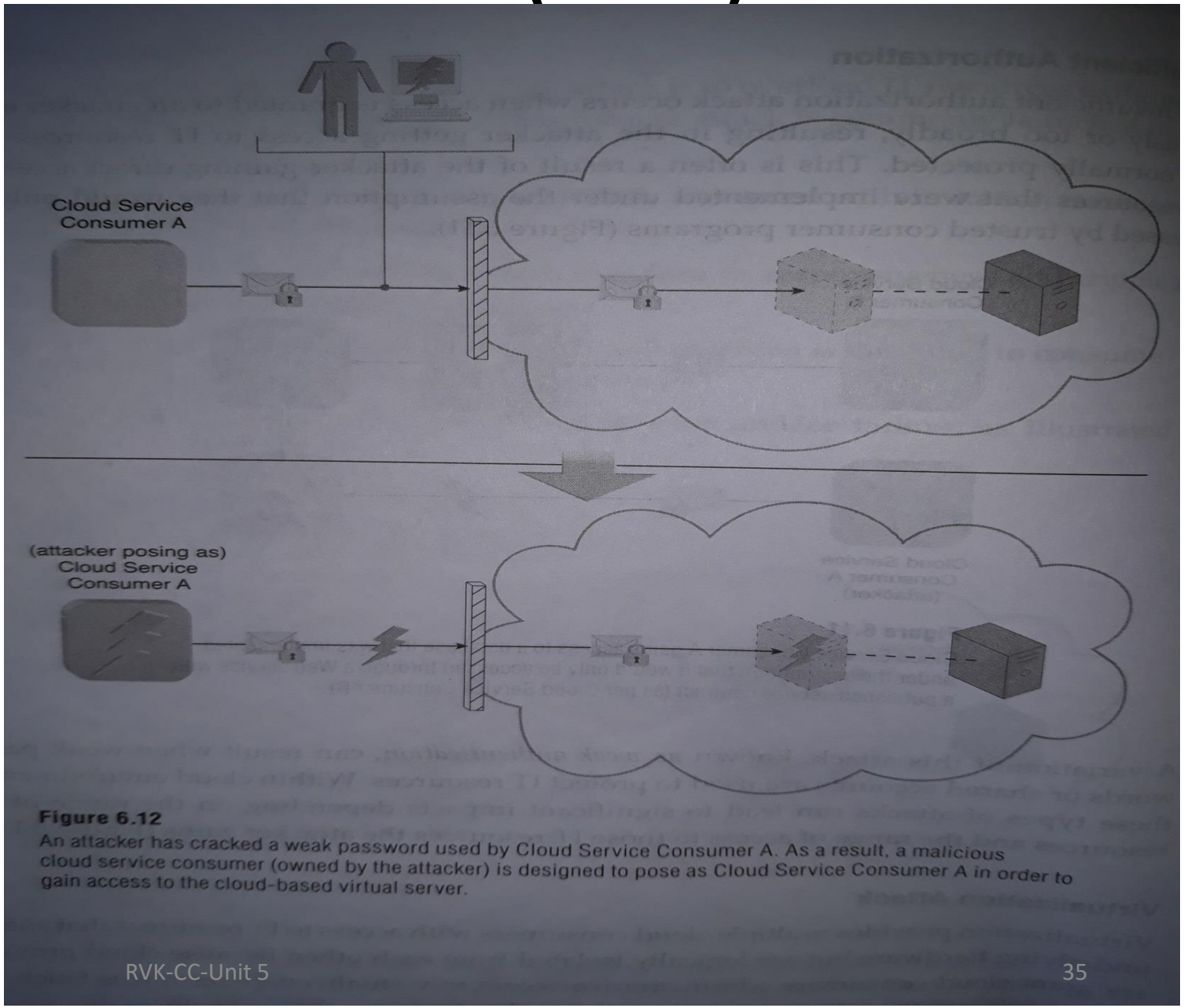


Figure 6.12
An attacker has cracked a weak password used by Cloud Service Consumer A. As a result, a malicious cloud service consumer (owned by the attacker) is designed to pose as Cloud Service Consumer A in order to gain access to the cloud-based virtual server.

Cloud Security Attacks (cont..)

6) Overlapping Trust Boundaries:

- If physical IT resources within a cloud are shared by different cloud service consumers , these cloud service consumers have overlapping trust boundaries .
- Malicious cloud service consumers can target shared IT resources with the intention of compromising cloud consumers or other IT resources that share the same trust boundary.

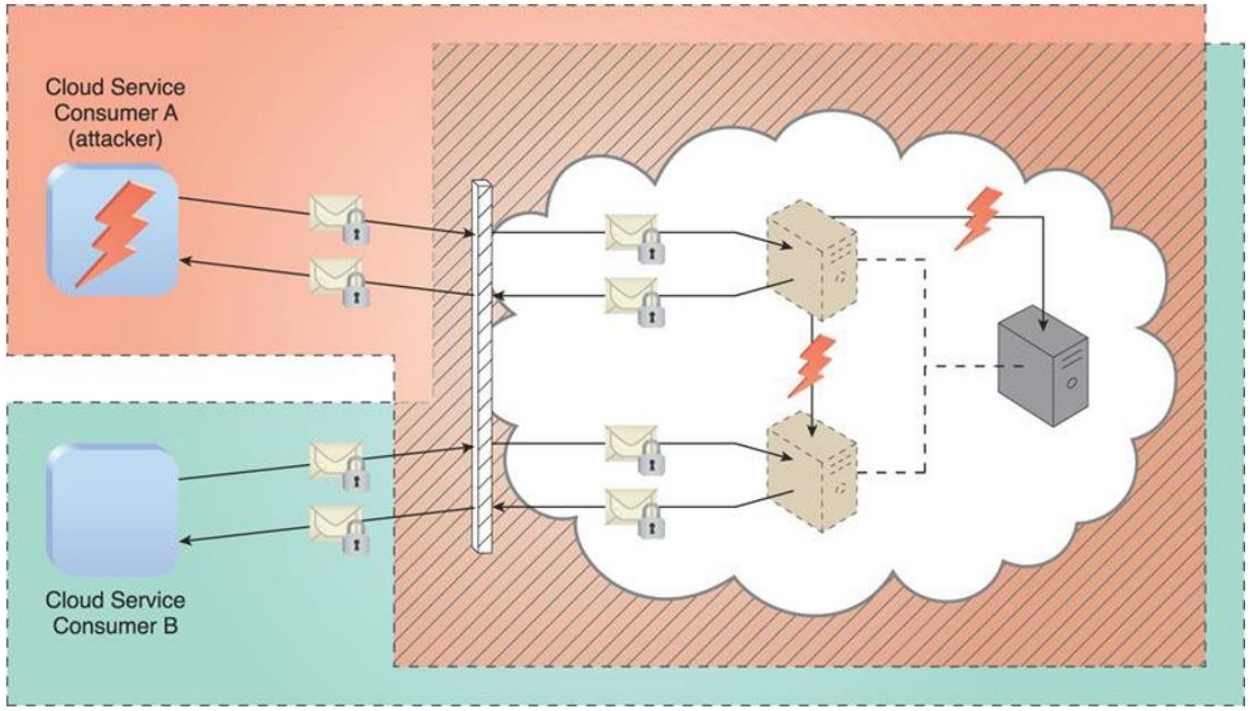


Figure 6.14 Cloud Service Consumer A is trusted by the cloud and therefore gains access to a virtual server, which it then attacks with the intention of attacking the underlying physical server and the virtual server used by Cloud Service Consumer B.

Cloud Security Threats and Attacks (cont..)

7) Flawed Implementations:

- The substandard design, implementation, or configuration of cloud service deployment can have undesirable consequences, beyond runtime exceptions and failures.

- If the cloud provider's software and/or hardware have inherent security flaws or operational weakness, attackers can exploit these vulnerabilities to impair the integrity, confidentiality and/or availability of cloud provider IT resources and cloud consumer IT resources hosted by the cloud provider.

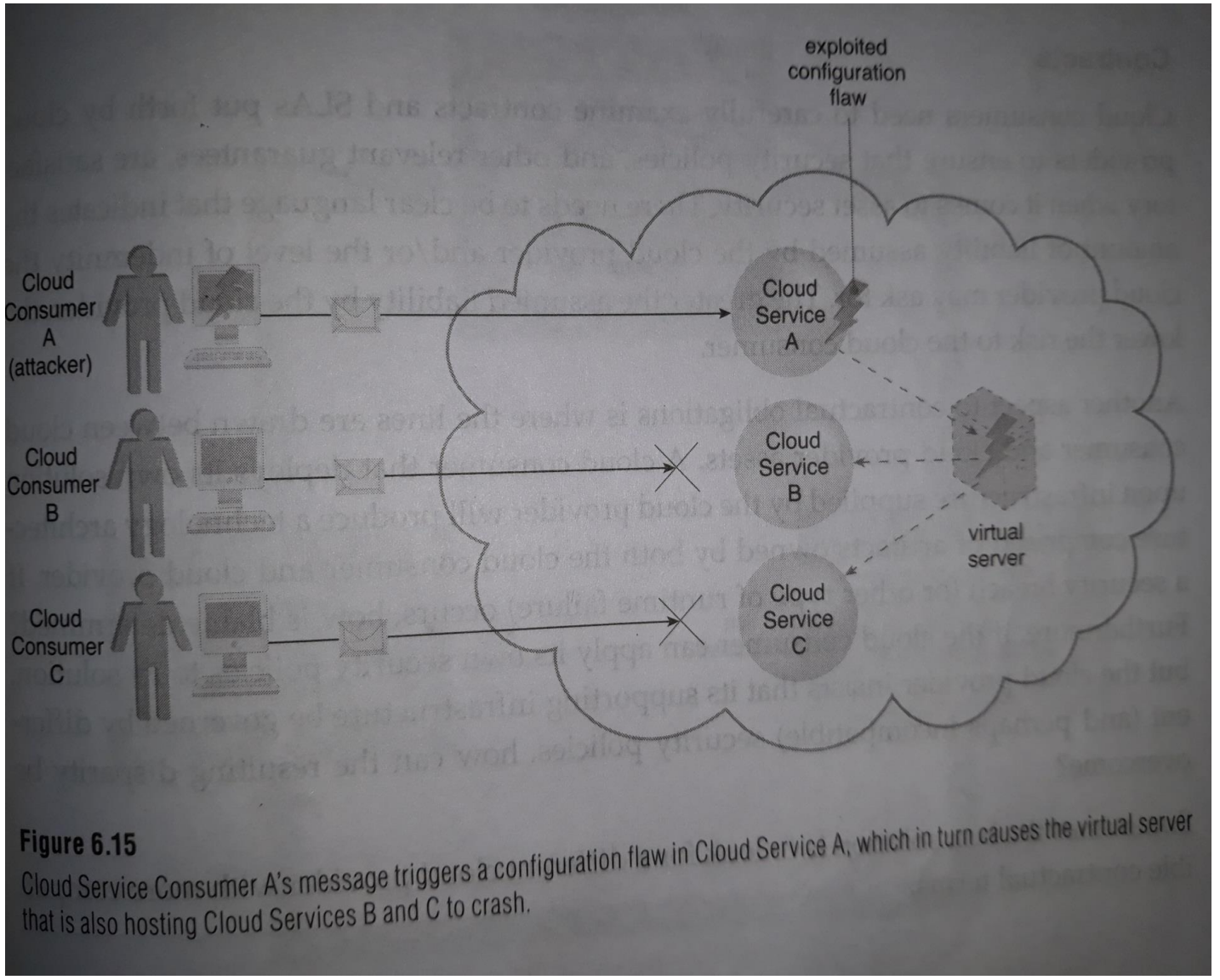


Figure 6.15
Cloud Service Consumer A's message triggers a configuration flaw in Cloud Service A, which in turn causes the virtual server that is also hosting Cloud Services B and C to crash.

Cloud Security Threats and Attacks (cont..)

8) Additional Considerations:

- Liability, indemnity, and blame for potential security breaches need to be clearly defined and mutually understood in the legal agreements signed by cloud consumers and cloud providers.

- It is important for cloud consumers, subsequent to gaining an understanding of the potential security-related issues specific to a given cloud environment, to perform a corresponding assessment of the identified risks.

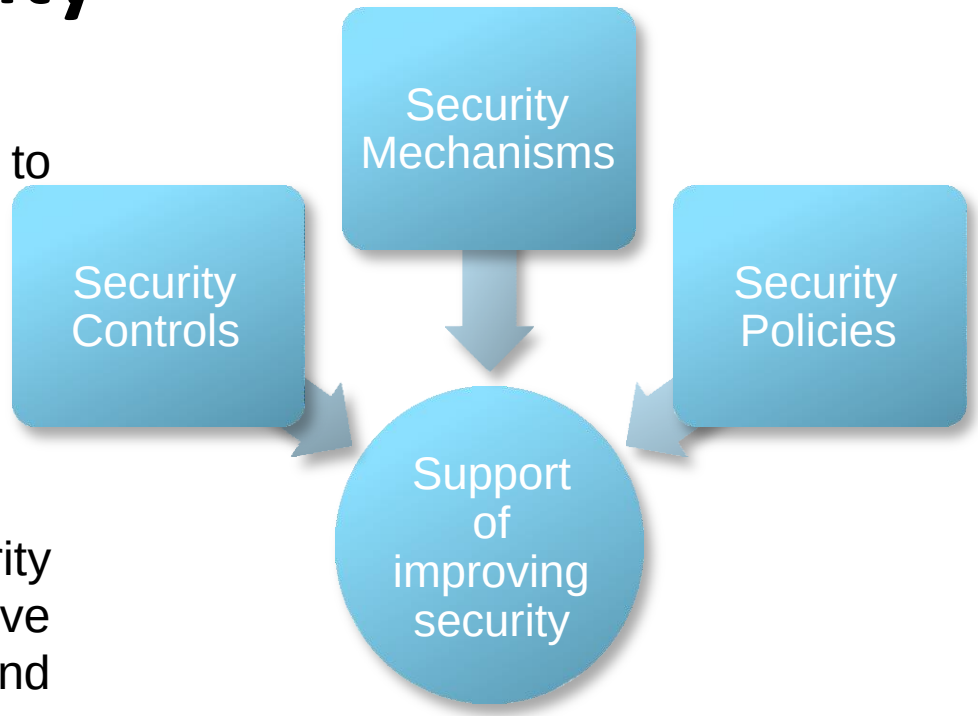
Cloud Security Mechanisms

Improving Security

- **Security Controls:**
  - These are countermeasures used to prevent or respond to security threats and to reduce or avoid risk.
    - Preventive Controls(looking out unauthorized intruders)
    - Detective Controls (Sounding Alarm)
    - Corrective Controls(Recovering)

- **Security Mechanisms:**
  - Countermeasures are typically described in terms of security mechanisms, which are components comprising a defensive framework that protects IT resources, information, and services.

- **Security Policies:**
  - A security policy establishes a set of security rules and regulations. Often, security policies will further define how these rules and regulations are implemented and enforced.



Cloud Security Mechanisms

- Encryption,
- Hashing,
- Digital Signature,
- Public Key Infrastructure (PKI),
- Identity and Access Management (IAM),
- Single Sign-On (SSO),
- Hardened Server Images.

Encryption

- Data by default is coded in a readable format known as plaintext . When transmitted over a network , plaintext is vulnerable to unauthorized and potentially malicious access.

- The encryption mechanism is a digital coding system dedicated to preserving the confidentiality and integrity of data. It is used for encoding plaintext data into a protected and unreadable format.

Encryption and Decryption

- **Types of Encryption:** 1) Symmetric and 2) Asymmetric

1) Symmetric Encryption:-

- Symmetric encryption uses the same key for both encryption and decryption both of which are performed by authorized parties that use the one shared key.
- It is also known as secret key cryptography, messages that are encrypted with a specific key can be decrypted by only that same key.
- Note that symmetrical encryption does not have the characteristics of non-repudiation.

2) Asymmetric Encryption:-

- Asymmetric encryption relies on the use of two different keys namely a private key and a public key.
- With asymmetric encryption (which is also referred to as public key cryptography) , the private key is known only to its owner while the public key is commonly available.
- A document that was encrypted with a private key can only be correctly decrypted with the corresponding public key.

Hashing

- The hashing mechanism is used when a one-way , nonreversible form of data protection is required . Once hashing has been applied to a message, it is locked, and no key is provided for the message to be unlocked. A common application fro this is the storage of passwords.

- Hashing technology can be used to derive a hashing code or message digest from a message , which is often of a fixed length and smaller than the original message.

- The message sender can then utilize the hashing mechanism to attach the message digest to the message. The recipient applies the same hash function to the message to verify that the produced message digest is identical to the one that accompanied the message.

- Any alteration to the original data results in an entirely different message digest and clearly indicates that tampering has occurred.

Hashing (cont..)

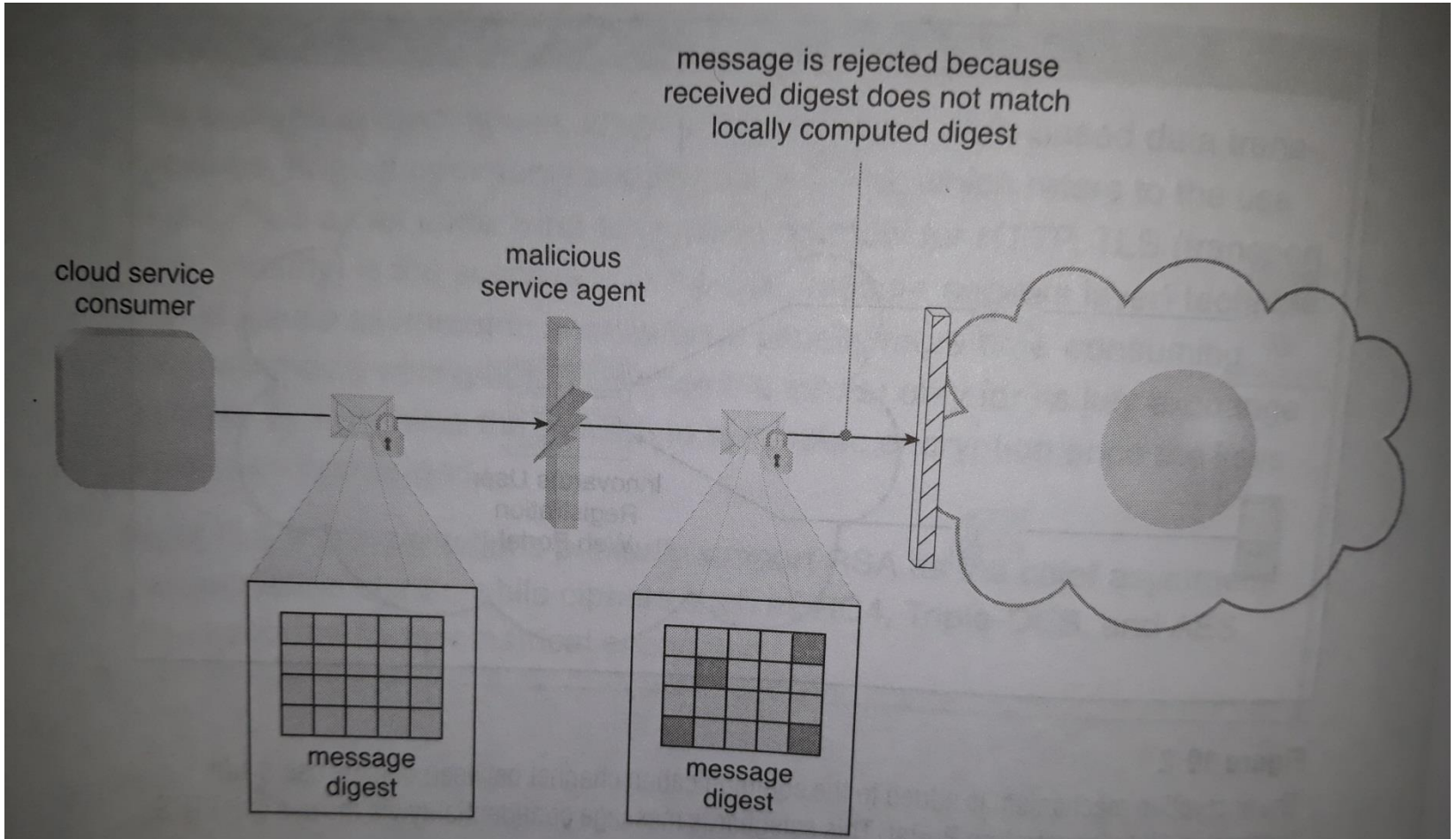


Figure 10.3

A hashing function is applied to protect the integrity of a message that is intercepted and altered by a malicious service agent, before it is forwarded. The firewall can be configured to determine that the message has been altered, thereby enabling it to reject the message before it can proceed to the cloud service.

Digital Signature

- The digital signature mechanism is a means of providing data authenticity and integrity through authentication and non-repudiation.
- A message is assigned a digital signature prior to transmission, which is then rendered invalid if the message experiences any subsequent, unauthorized modifications.
- A digital signature provides evidence that the message received is the same as the one created by its rightful sender.

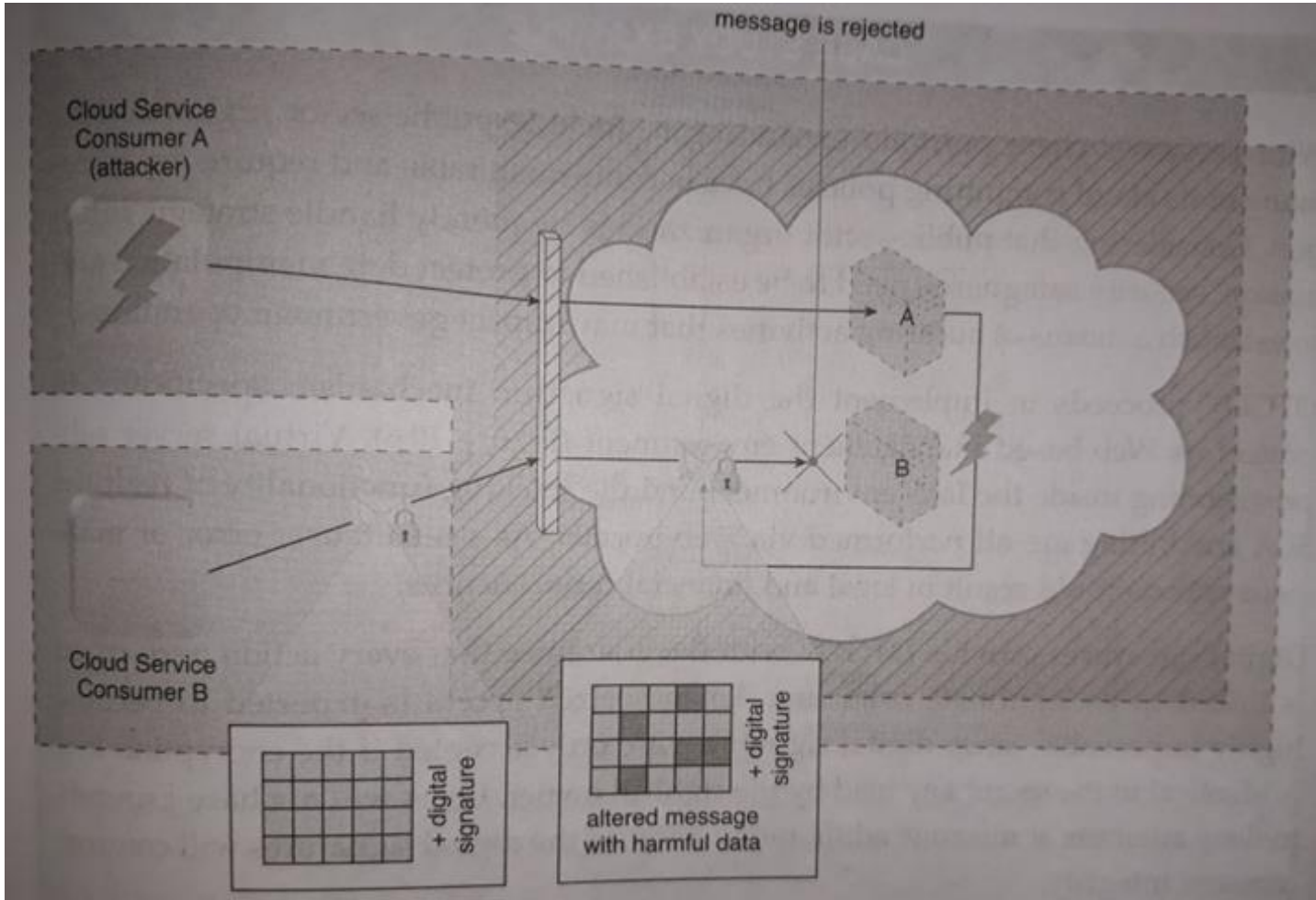


Figure 10.5
Cloud Service Consumer B sends a message that was digitally signed but was altered by trusted attacker Cloud Service Consumer A. Virtual Server B is configured to verify digital signatures before processing incoming messages even if they are within its trust boundary. The message is revealed as illegitimate due to its invalid digital signature, and is therefore rejected by Virtual Server B.

Public Key Infrastructure (PKI)

- A common approach for managing the issuance of asymmetric keys is based on the public key infrastructure (PKI) mechanism, which exists as a system of protocols, data formats, rules and practices that enable large scale systems to securely use public keys with their corresponding key owners (known as public key identification) while enabling the verification of key validity.

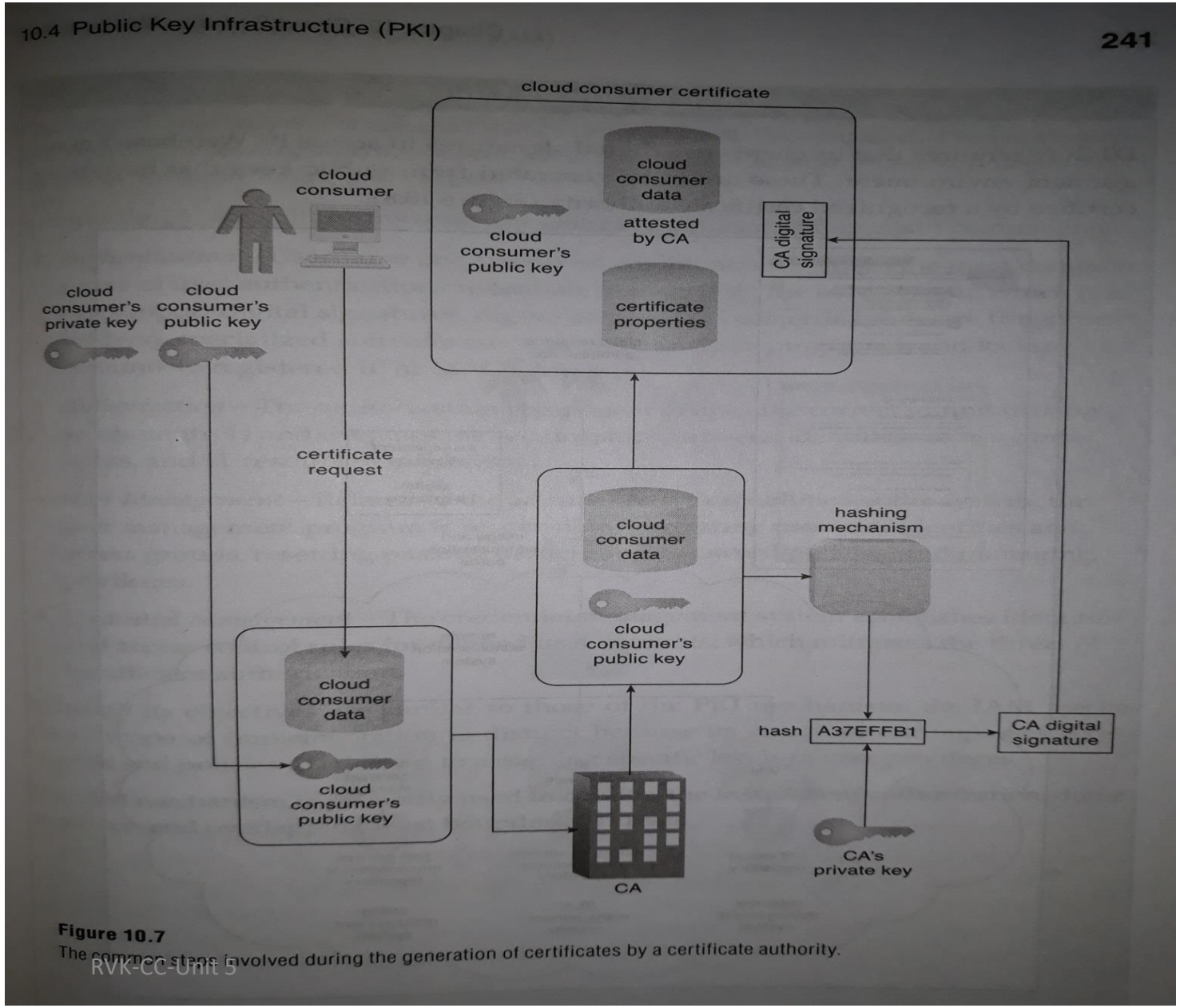


Figure 10.7
The common steps involved during the generation of certificates by a certificate authority.

Best Practices for PKI in Cloud

- The key management server must be hosted within the organization, and whenever the data which is hosted in the cloud needs keys to decrypt the data as a part of end-user request, the key management server provides them. The key used for decryption should never be stored in the cloud VMs and must be in-memory for a few-nano seconds only.
- All the data that is leaving and entering the organizations can be encrypted and decrypted respectively.
- All the VMs that are hosted in the cloud must be encrypted to protect data loss when a VM snapshot is stolen.
- When the data which is encrypted and put in a cloud is no longer needed, the organization must revoke the keys associated with it, so that even if some trail of data remains in the decommissioned VM, it cannot be decrypted.
- The Hardware Security Model (HSM) should be used to store keys for cryptographic operations such as encryption, decryption, etc.
- Use of old and insecure protocols like Data Encryption Standard (DES) must be avoided.

Single Sign – On (SSO)

- Propagating the authentication and authorization information for a cloud service consumer across based IT resources.

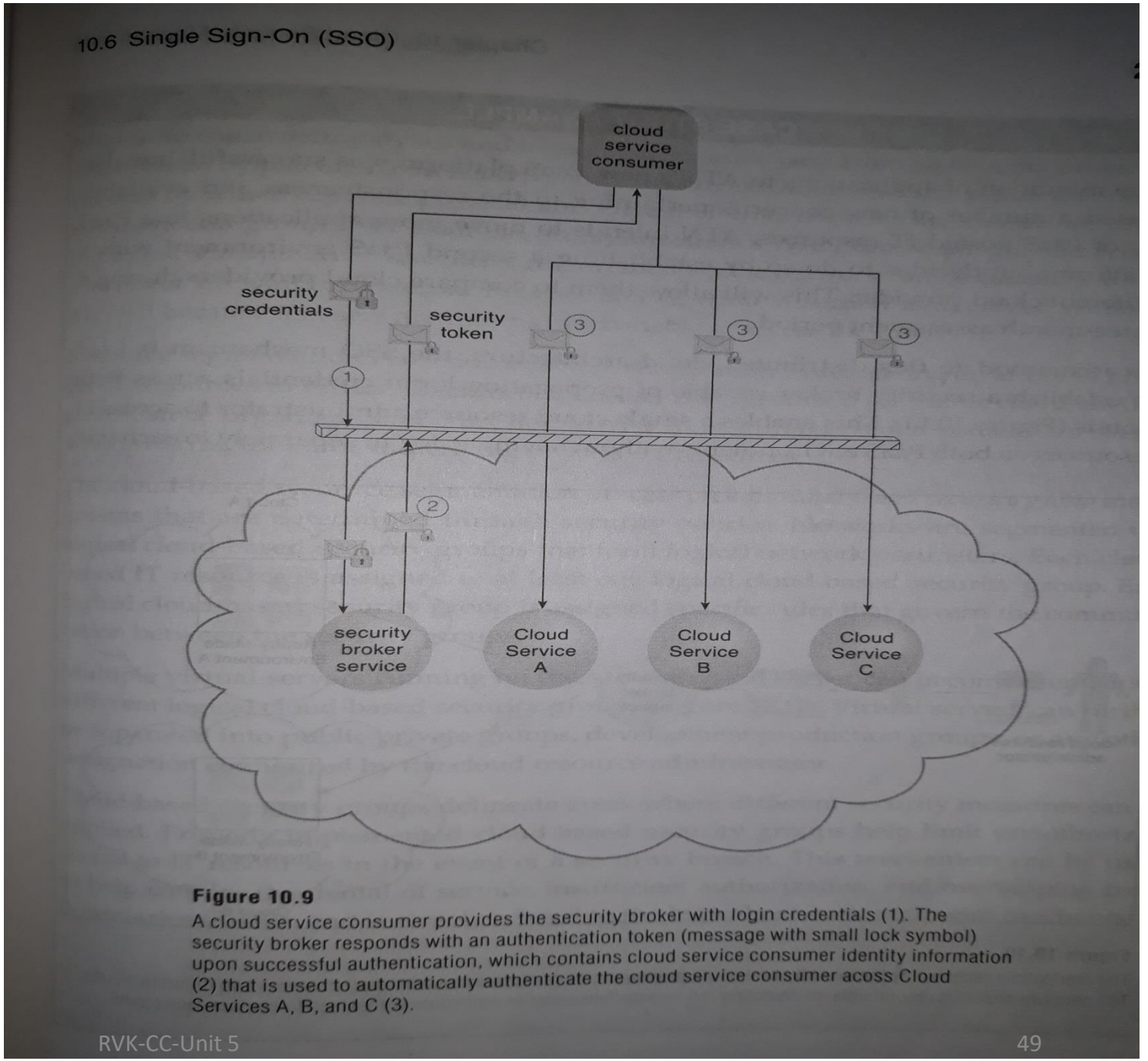


Figure 10.9
A cloud service consumer provides the security broker with login credentials (1). The security broker responds with an authentication token (message with small lock symbol) upon successful authentication, which contains cloud service consumer identity information (2) that is used to automatically authenticate the cloud service consumer acoss Cloud Services A, B, and C (3).

Hardened Virtual Server Images

- A virtual server is created from a template configuration called a virtual server images.
- Hardening is the process of striping unnecessary software from a system to limit potential vulnerabilities that can be exploited by attackers.
- A Hardened Virtual Server Image is a template for virtual service instance creation that has been subjected to hardening process.

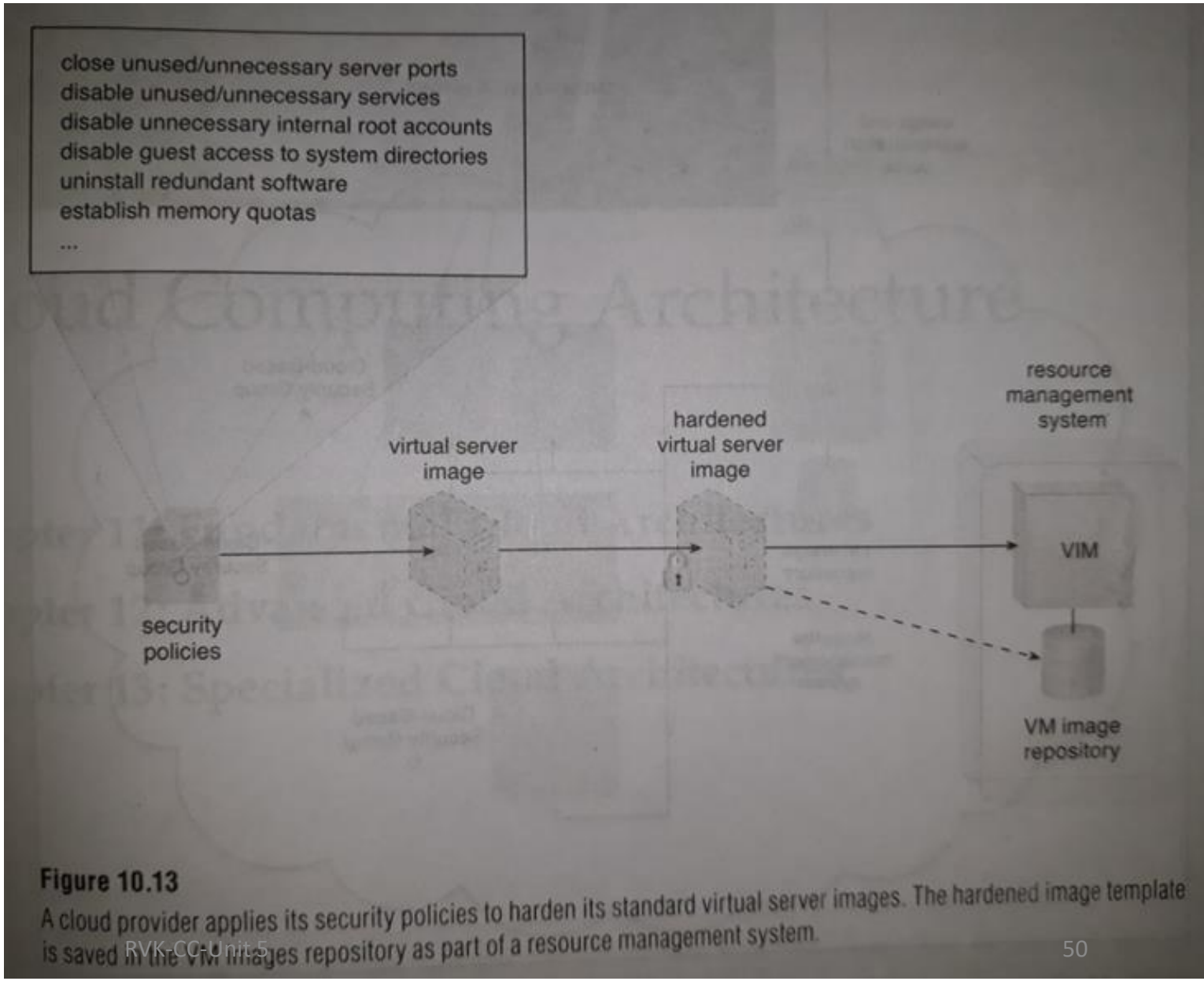
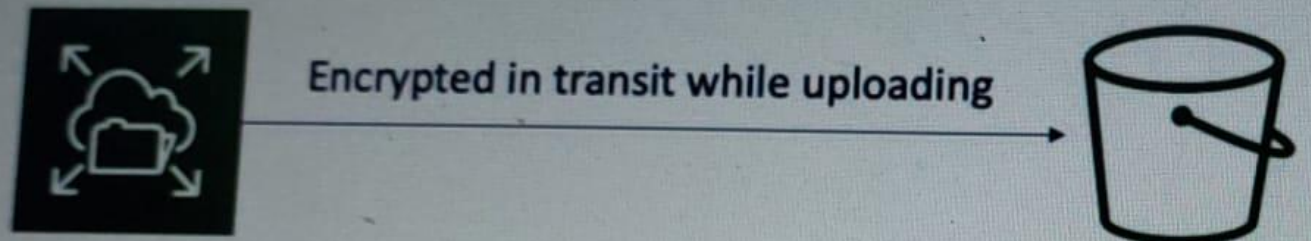


Figure 10.13
A cloud provider applies its security policies to harden its standard virtual server images. The hardened image template is saved in the VM images repository as part of a resource management system.

AWS Key Management Service (AWS KMS)

Data at rest vs. Data in transit



- At rest: data stored or archived on a device
  - On a hard disk, on a RDS instance, in S3 Glacier Deep Archive, etc.
- In transit (in motion): data being moved from one location to another
  - Transfer from on-premises to AWS, EC2 to DynamoDB, etc
  - Means data transferred on the network
- We want to encrypt data in both states to protect it!
- For this we leverage encryption keys

AWS KMS (Key Management Service)

- Anytime you hear "encryption" for an AWS service, it's most likely KMS
- KMS = AWS manages the encryption keys for us
- Encryption Opt-in:
  - EBS volumes: encrypt volumes
  - S3 buckets: Server-side encryption of objects
  - Redshift database: encryption of data
  - RDS database: encryption of data
  - EFS drives: encryption of data
- Encryption Automatically enabled:
  - CloudTrail Logs
  - S3 Glacier
  - Storage Gateway

CloudHSM

- KMS => AWS manages the software for encryption
- CloudHSM => AWS provisions encryption **hardware**
- Dedicated Hardware (HSM = Hardware Security Module)
- You manage your own encryption keys entirely (not AWS)
- HSM device is tamper resistant, FIPS 140-2 Level 3 compliance

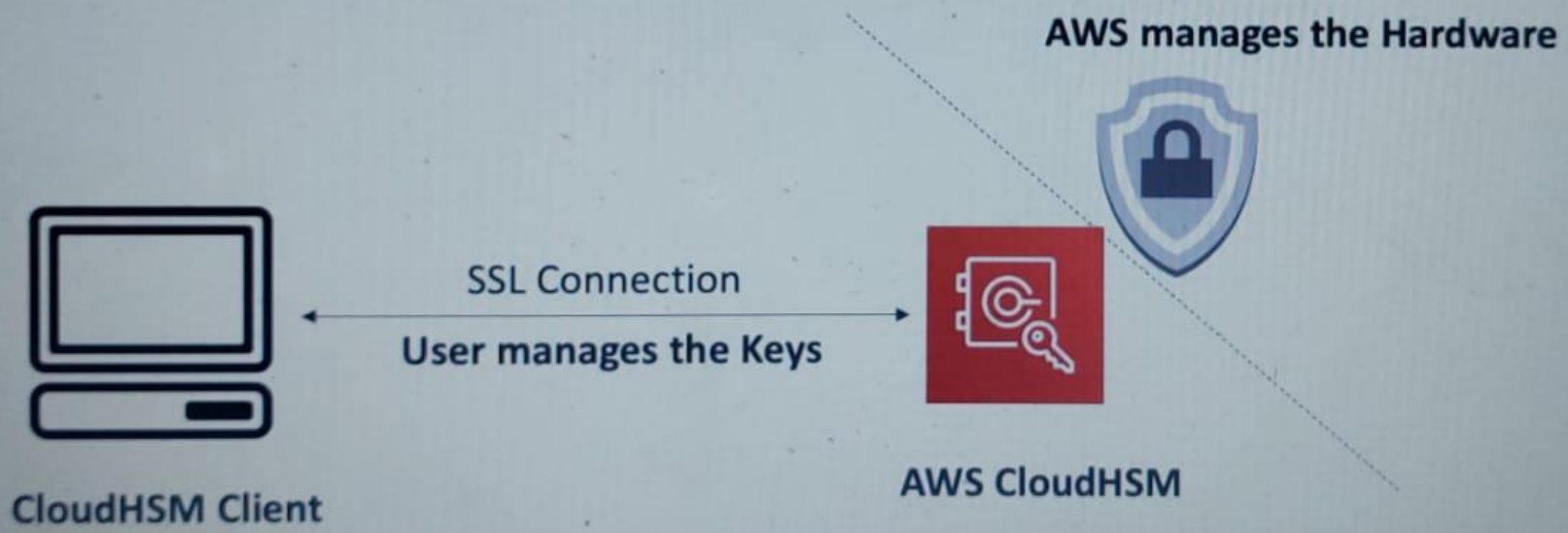
Sample HSM device

CloudHSM

- KMS => AWS manages the software for encryption
- CloudHSM => AWS provisions encryption **hardware**
- Dedicated Hardware (HSM = Hardware Security Module)
- You manage your own encryption keys entirely (not AWS)
- HSM device is tamper resistant, FIPS 140-2 Level 3 compliance

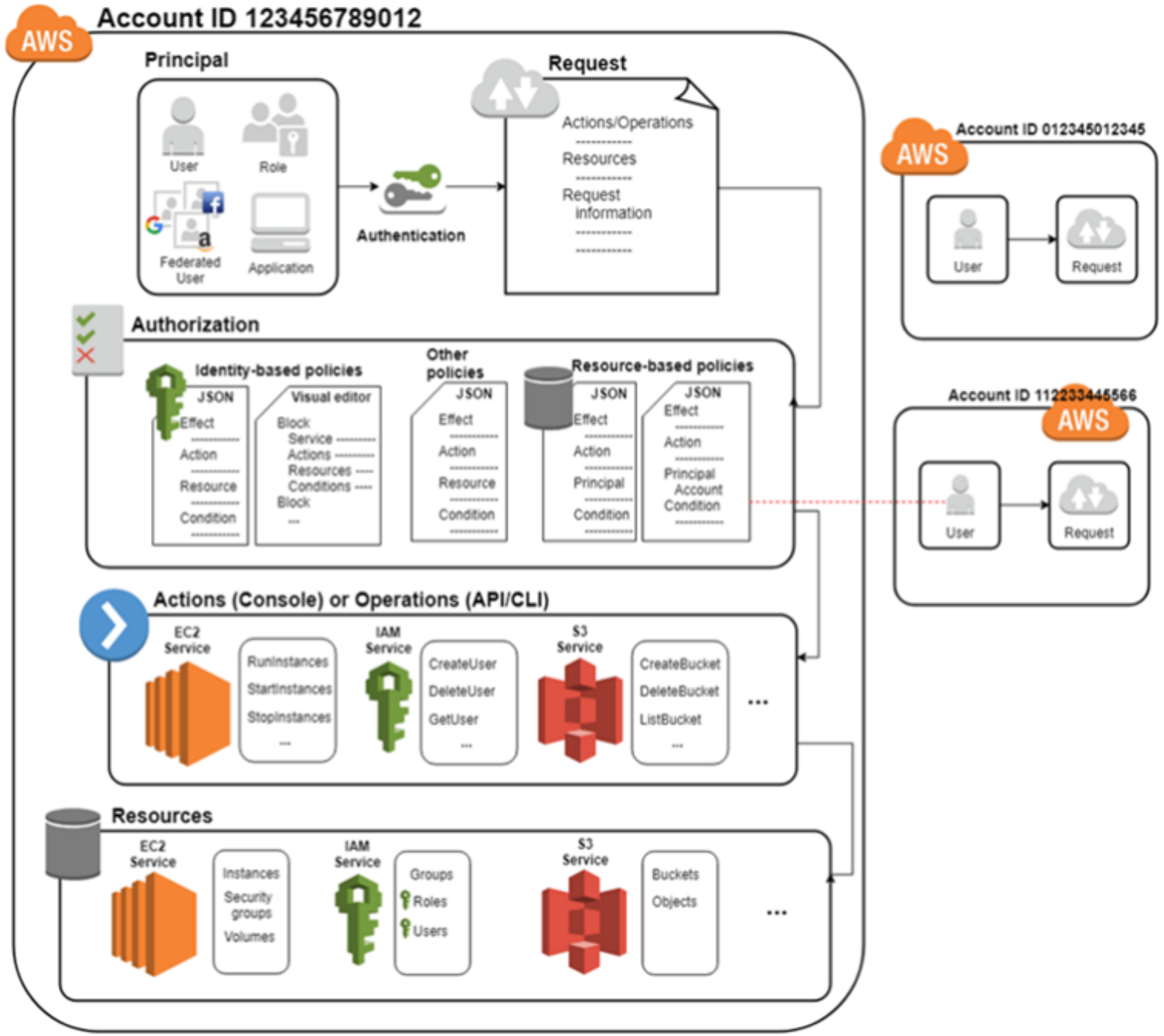
Sample HSM device

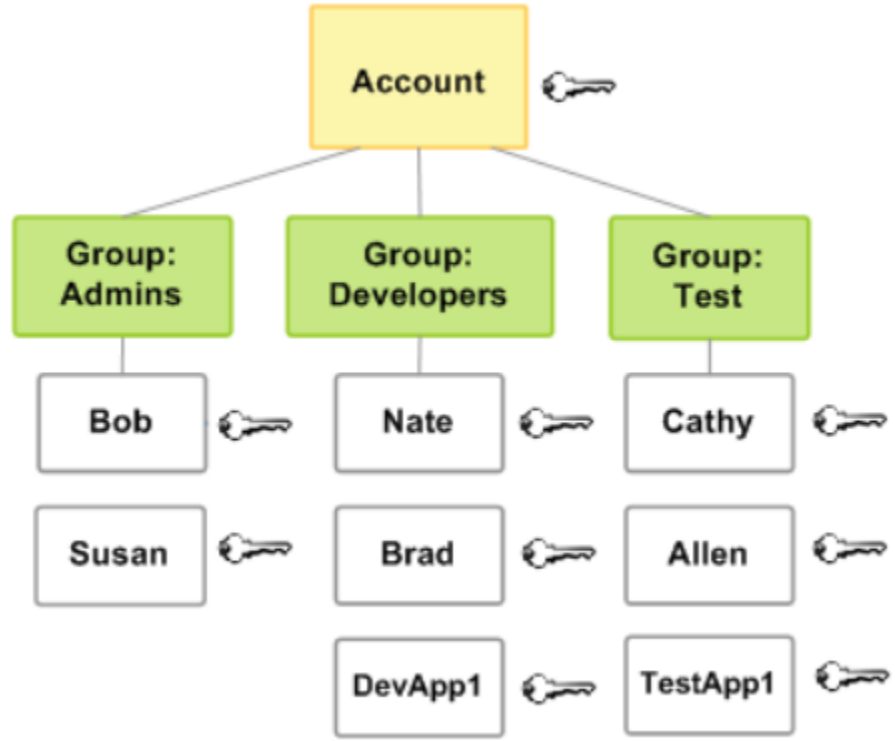
CloudHSM Diagram



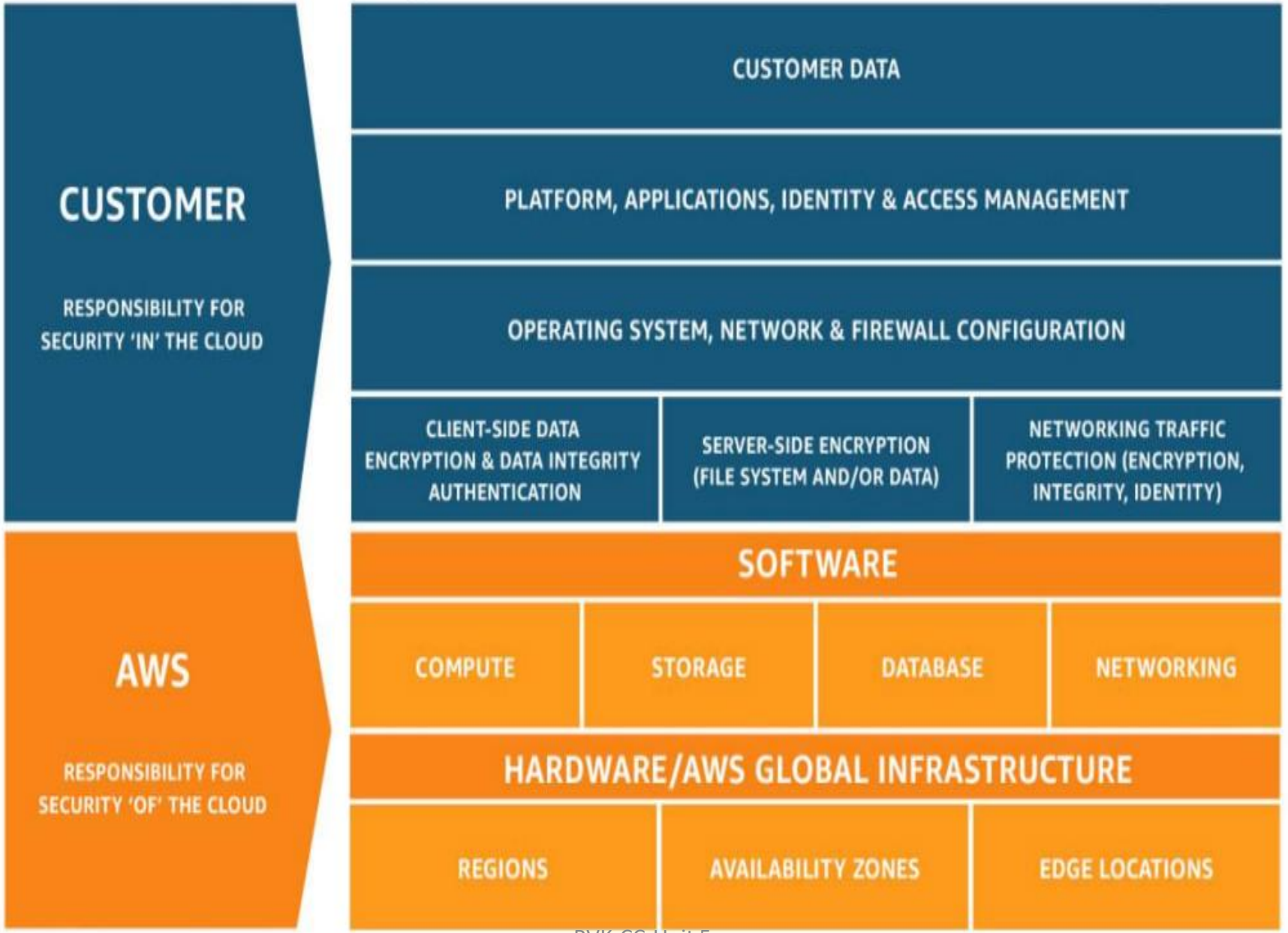
Types of Customer Master Keys: CMK

- Customer Managed CMK:
  - Create, manage and used by the customer, can enable or disable
  - Possibility of rotation policy (new key generated every year, old key preserved)
  - Possibility to bring-your-own-key
- AWS managed CMK:
  - Created, managed and used on the customer's behalf by AWS
  - Used by AWS services (aws/s3, aws/ebs, aws/redshift)
- AWS owned CMK:
  - Collection of CMKs that an AWS service owns and manages to use in multiple accounts
  - AWS can use those to protect resources in your account (but you can't view the keys)
- CloudHSM Keys (custom keystore):
  - Keys generated from your own CloudHSM hardware device
  - Cryptographic operations are performed within the CloudHSM cluster

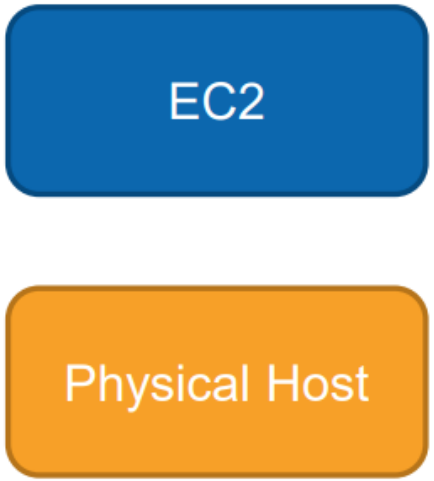

Account ID 123456789012
Principal
User
Role
Federated User
Application
Authentication
Request
Actions/Operations
Resources
Request information
Authorization
Identity-based policies
JSON
Effect
Action
Resource
Condition
Visual editor
Block
Service
Actions
Resources
Conditions
Other policies
Resource-based policies
Principal
Account Condition
Actions (Console) or Operations (API/CLI)
EC2 Service
RunInstances
StartInstances
StopInstances
IAM Service
CreateUser
DeleteUser
GetUser
S3 Service
CreateBucket
DeleteBucket
ListBucket
Resources
Instances
Security groups
Volumes
Groups
Roles
Users
Buckets
Objects
Account ID 012345012345
Request
Account ID 112233445566


Account
Group: Admins
Group: Developers
Group: Test
Bob
Susan
Nate
Brad
DevApp1
Cathy
Allen
TestApp1

AWS Cloud Security



EC2



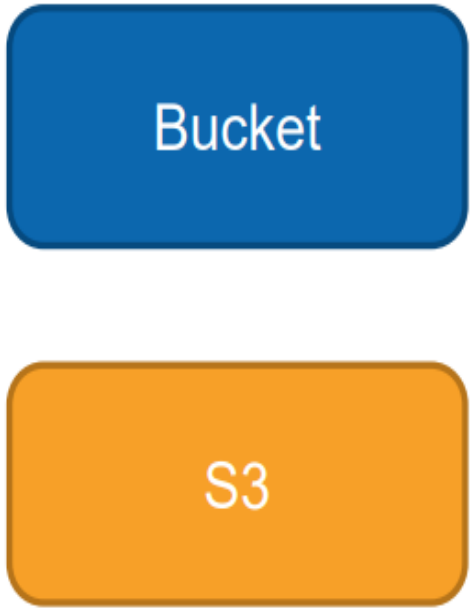
Customer Responsibilities

- Guest OS, Patching
- Firewalls (Security Group, Network ACL)
- Availability, Scalability, Monitoring

AWS Responsibilities

- Physical Host
- Virtualization

S3



Customer Responsibilities

- Storage Class
- Access Controls
- Data Encryption

AWS Responsibilities

- Hardware, Software
- Scalability

Identity and Access Management (IAM)

For more details refer to :
https://docs.aws.amazon.com/pdfs/IAM/latest/UserGuide/iam-ug.pdf

Identity and Access Management (IAM)

- There are many types of security services like Identity and Access Management (IAM), Key Management System (KMS), Cognito, Web Access Firewall (WAF), but IAM is one the most widely used.
- AWS Identity and Access Management (IAM) is a web service for securely controlling access to AWS resources. It enables you to create and control services for user authentication or limit access to a certain set of people who use your AWS resources.
- Elements of IAM Workflow:
  1. Principal: It is an entity that can perform actions on an AWS resource. A user, a role or an application can be a principal.
  2. Authentication: It is the process of confirming the identity of the principal trying to access an AWS product. The principal must provide its credentials or required keys for authentication.
  3. Request: A principal sends a request to AWS specifying the action and which resource should perform it.
  4. Authorization: By default, all resources are denied. IAM authorizes a request only if all parts of the request are allowed by a matching policy. After authenticating and authorizing the request, AWS approves the action.
  5. Actions: These are used to view, create, edit or delete a resource.
  6. Resources: A set of actions can be performed on a resource related to your AWS account.

Identity and Access Management (IAM) (cont..)

- The identity and access management (IAM) mechanism encompasses the components and policies necessary to control and track user identities and access privileges for IT resources , environments and systems.

- Authentication - Username and password combinations remain the most common forms of user authentication credentials managed by the IAM system , which also can support
- digital signatures ,
- digital certificates ,
- biometric hardware (fingerprint readers) ,
- specialized software (such as voice analysis programs)
- locking user accounts to registered IP or MAC addresses.

Identity and Access Management (IAM) (cont..)

1. **Authorization –**
   - The authorization component defines the correct granularity for access controls and oversees the relationships between identities , access control rights and IT resource availability.

2. **Authentication –**
   - Username and password combinations remain the most common forms of user authentication credentials managed by the IAM system , which also can support digital signatures, digital certificates , biometric hardware, specialized software and locking user accounts to registered IP or MAC addresses**.**

3. User Management

- Related to the administrative capabilities of the system, the program is responsible for creating new user identities and access groups , resetting passwords , defining password policies and managing privileges.

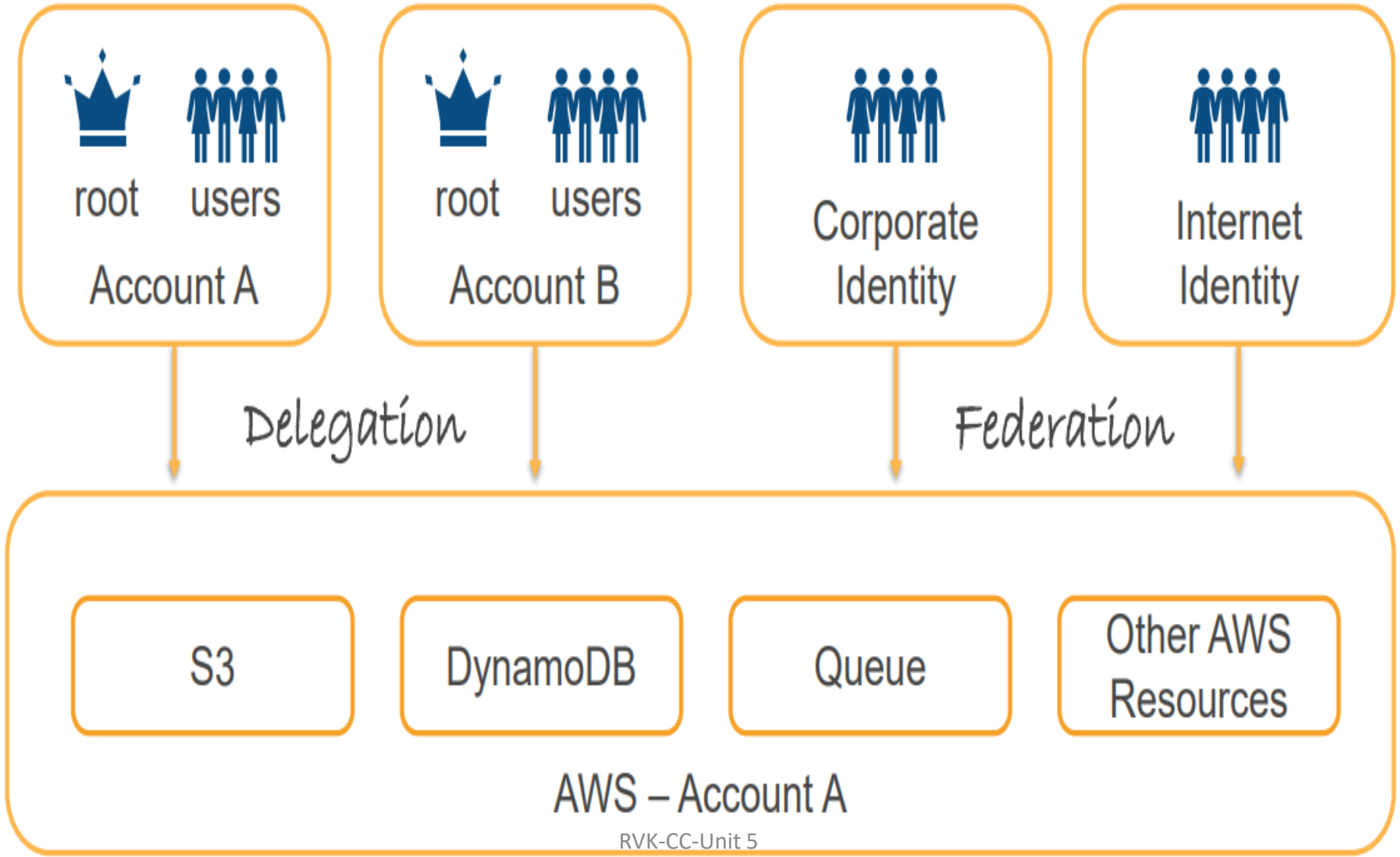
4. Credential Management

- The credential management system establishes identities and access control rules for defined user accounts , which mitigates the threat of insufficient authorization.

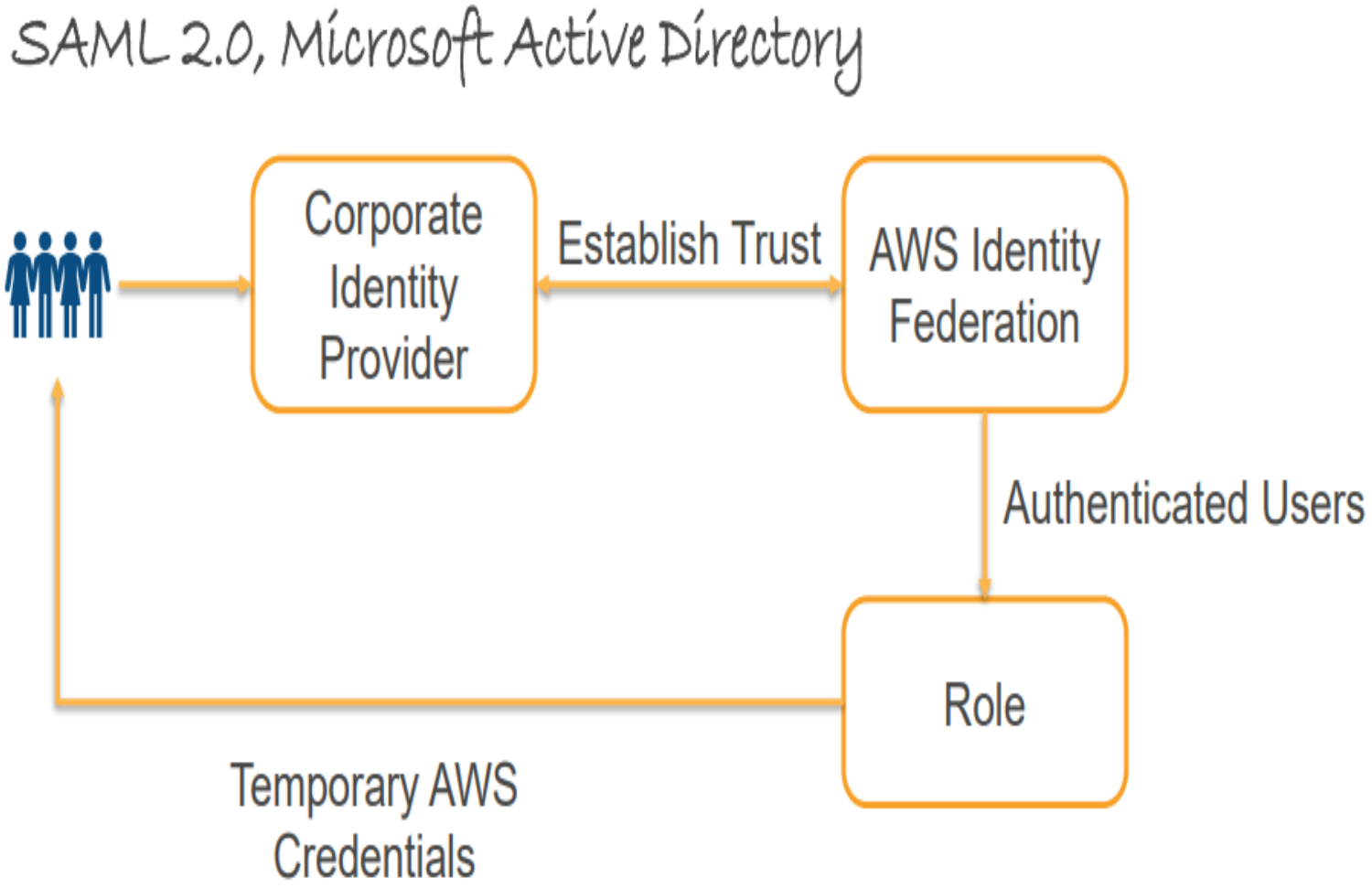
Features of IAM

- Some of the main features of IAM are:
  - Shared access to the AWS account: IAM allows you to create separate usernames and passwords for individual users or resources and delegate access.
  - Granular permissions: Restrictions can be applied to requests. For example, you can allow the user to download information, but deny the user the ability to update information through the policies.
  - Multifactor authentication (MFA): IAM supports MFA, in which users provide their username and password plus a one-time password from their phone—a randomly generated number used as an additional authentication factor.
  - Identity Federation: If the user is already authenticated, such as through a Facebook or Google account, IAM can be made to trust that authentication method and then allow access based on it. This can also be used to allow users to maintain just one password for both on-premises and cloud environment work.
  - Free to use: There is no additional charge for IAM security. There is no additional charge for creating additional users, groups or policies.
  - PCI DSS compliance: The Payment Card Industry Data Security Standard is an information security standard for organizations that handle branded credit cards from the major card schemes. IAM complies with this standard.
  - Password policy: The IAM password policy allows you to reset a password or rotate passwords remotely. You can also set rules, such as how a user should pick a password or how many attempts a user may make to provide a password before being denied access.

Types of Identities

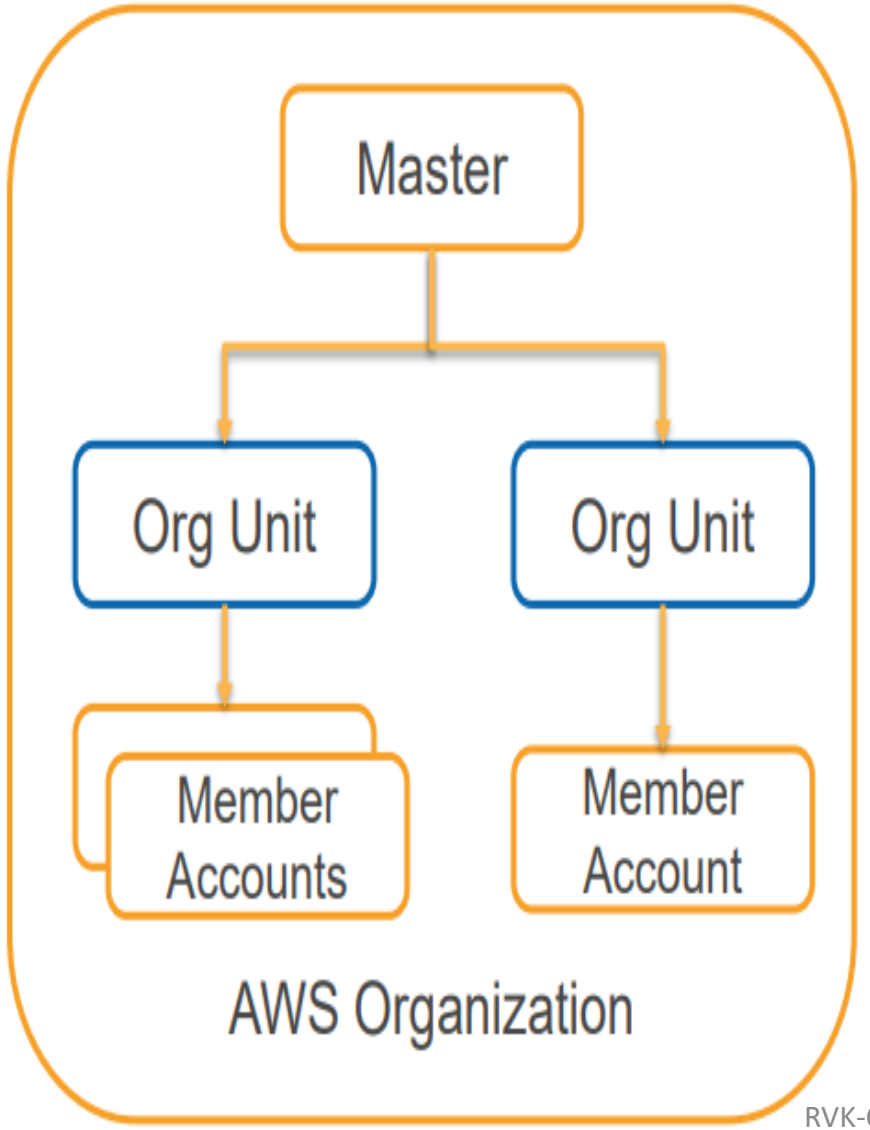


Corporate Identity Federation



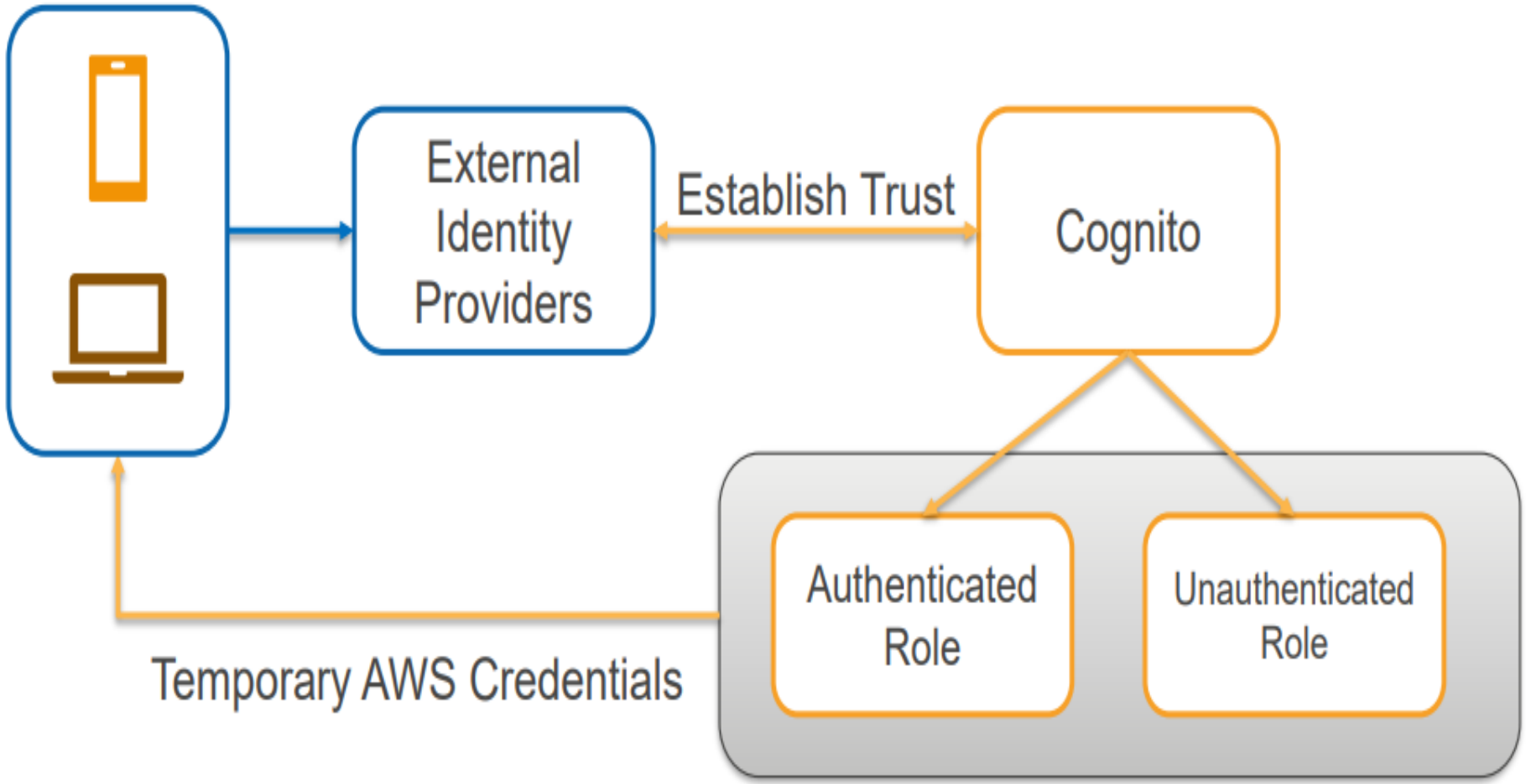
- SAML 2.0(Security Assertion Mark Up Language) to exchange identity and security information between identity provider and application.

AWS Organizations



- Centrally manage costs and billing
- Service Control Policy – control services, resources, regions used by member account
- Share resources across accounts
- AWS Single Sign-on – manage access to employees and accounts
- Centralize identity management and federation

Internet Identity Federation



SAML 2.0, OAuth 2.0, OpenID Connect

Components of IAM

1. Users:

- An IAM user is an identity with an associated credential and permissions attached to it.
- This could be an actual person who is a user, or it could be an application that is a user.
- With IAM, you can securely manage access to AWS services by creating an IAM username for each employee in your organization.
- Each IAM user is associated with only one AWS account.
- By default, a newly created user is not authorized to perform any action in AWS.
- The advantage of having one-to-one user specification is that you can individually assign permissions to each user.

2. Groups

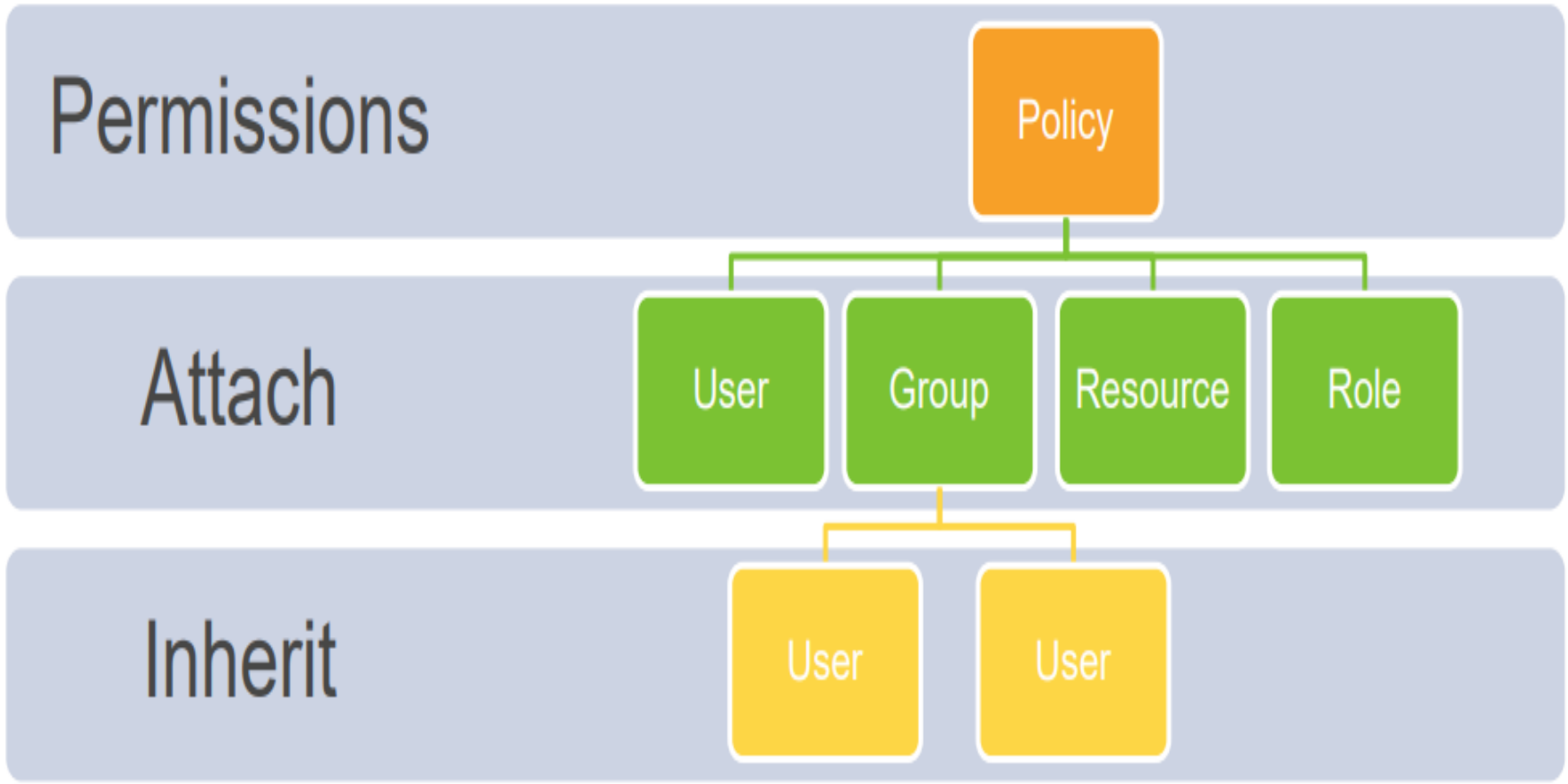
- A collection of IAM users is an IAM group used to specify permissions for multiple users so that any permissions applied to the group are applied to the individual users in that group as well.
- You set permissions for the group, and those permissions are automatically applied to all the users in the group.
- The nested groups are not allowed.
- If you add another user to the group, the new user will automatically inherit all the policies and the permissions already assigned to that group. This lessens the administrative burden.

Components of IAM (cont..)

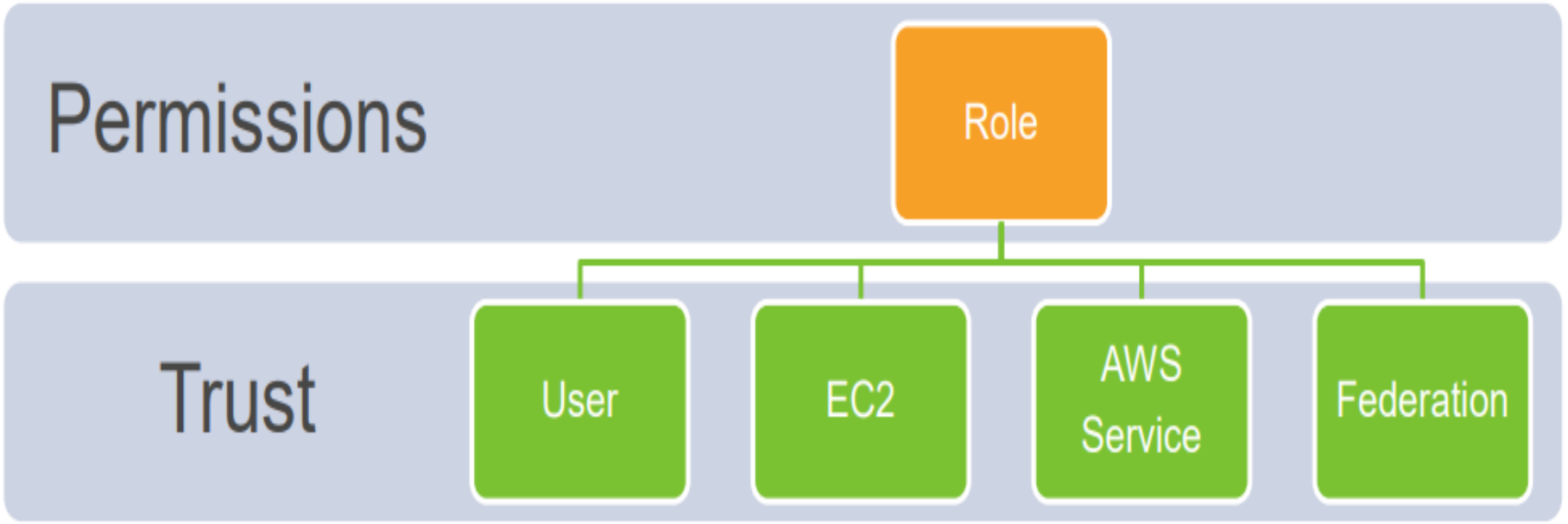
3. Policies:
   - An IAM policy sets permission and controls access to AWS resources. Permissions specify who has access to the resources and what actions they can perform.
   - Policies are stored in AWS as JSON documents.
   - There are **two types of policies**: 1) managed policies (default) and 2) inline policies (AWS-managed or customer-managed).
   - The policy would contain the following information:
     1. Who can access it
     2. What actions that user can take
     3. Which AWS resources that user can access
     4. When they can be accessed

4. Roles:
   - An IAM role is a set of permissions that define what actions are allowed and denied by an entity in the AWS console.
   - It is similar to a user in that it can be accessed by any type of entity (an individual or AWS service).
   - Role permissions are temporary credentials.

Access Management Concepts



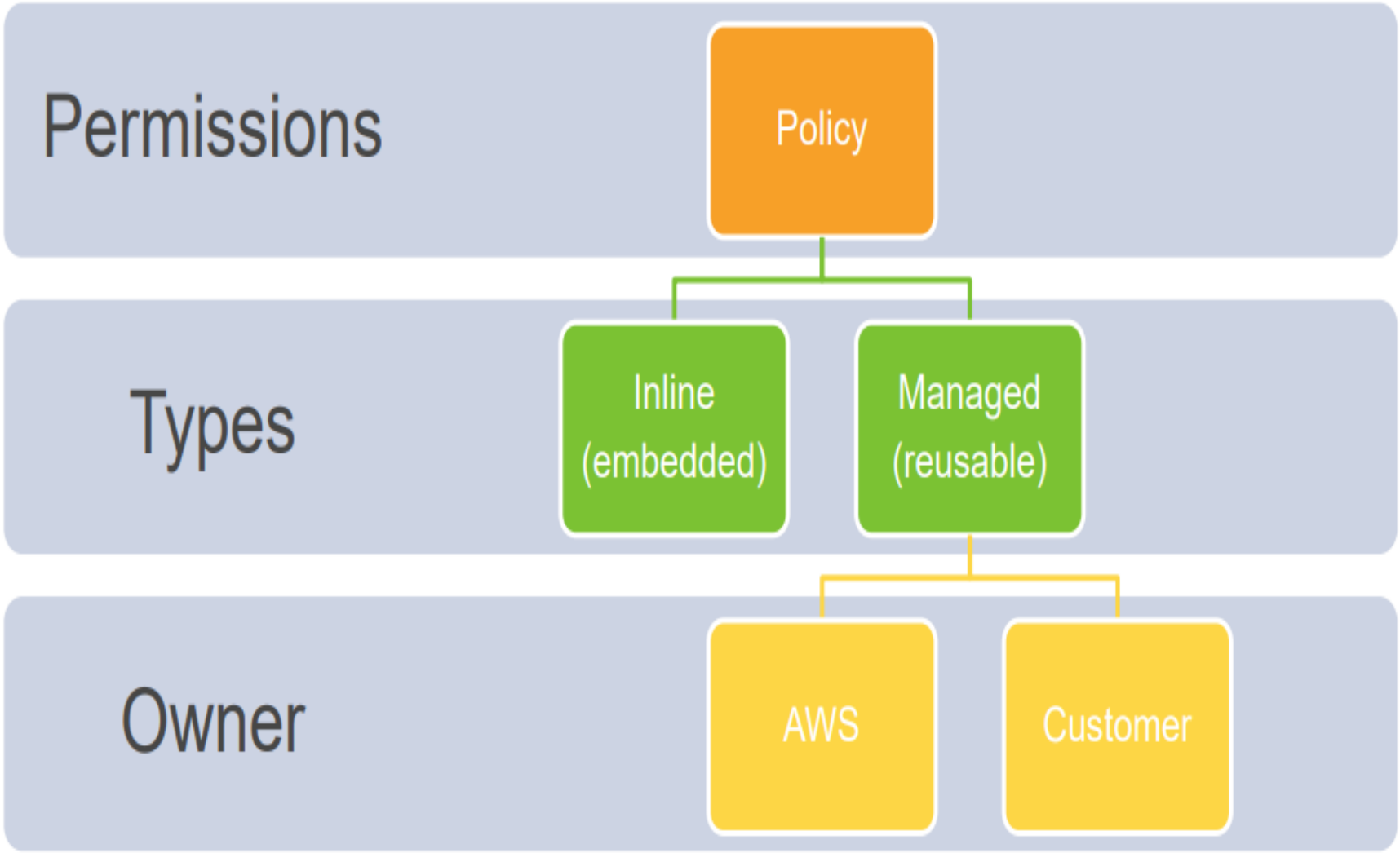
Role



Role has two parts:

1. Who can assume the role (trust relationship) and
2. What access is allowed (permissions)

Policy Types



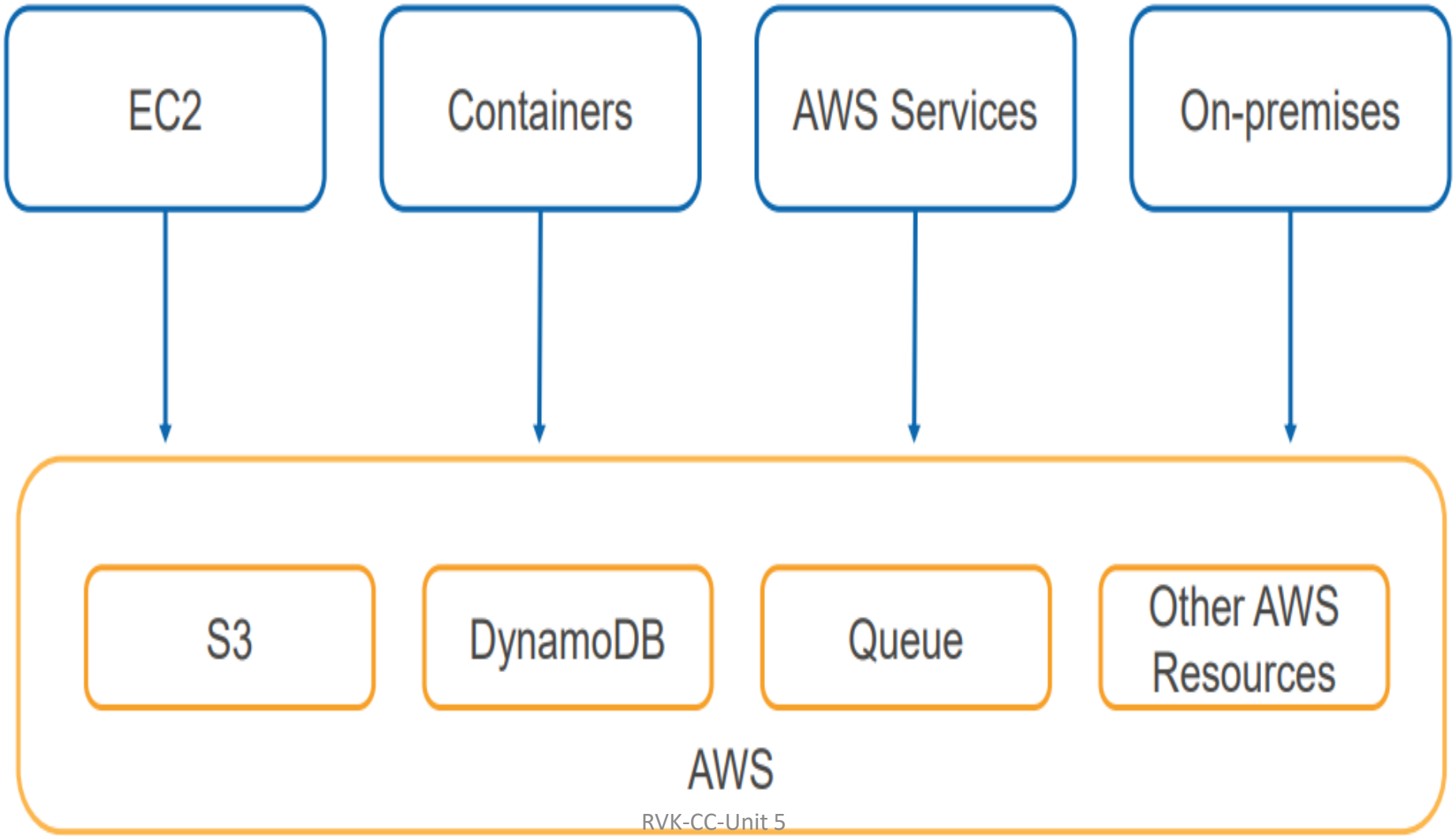
Zero Trust

AWS operates on principle of zero trust

- Authentication – caller needs to prove identity
- Authorization – caller needs permission

- Some services allow anonymous access
    - S3 bucket with Public Access

Types of Applications



IAM User Sign-in Credentials

- No sharing of credentials
- User ID and Password for Management Console Access
- Access Key and Secret Access Key for CLI, Programmatic Access
- Optional Multi-Factor Authentication (MFA)

IAM User Access Management

- No resource access by default
- Identity-based policy - Attach Policy to the User
- Resource-based policy - Attach Policy to the resource (only for supported AWS resources like S3, SQS, Lambda and so forth)
- IAM Roles - Grant temporary access to resources (and user gains privileges assigned to that Role)

Sample Identity-based Policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Resource": "*"
        }
    ]
}
```

Sample Resource-based Policy

```
{"Version": "2012-10-17",
 "Statement": [
    {
        "Effect": "Allow",
        "Action": ["s3:Get*","s3:List*"],
        "Resource": [
            "arn:aws:s3:::bucket_name",
            "arn:aws:s3:::bucket_name/*"],
        "Principal": {
            "AWS": [
            "arn:aws:iam::AWS-account-ID:user/alice",
            "arn:aws:iam::AWS-account-ID:user/bob"]}
    }
    ]
}
```

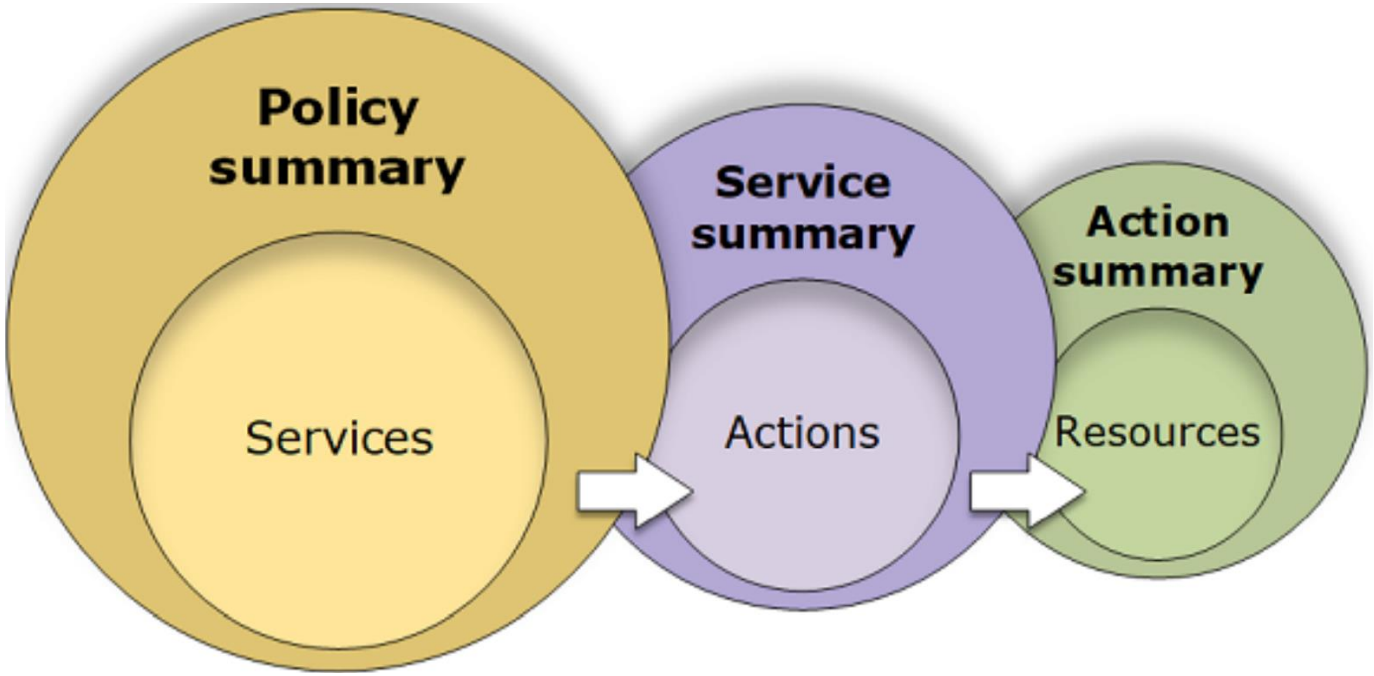
NOTE: Groups are not supported as principals in any policy

Policy Development and Testing

- Policy Visual Editor
- Policy Examples
  https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_examples.html
- Policy Simulator to Test Policy
  https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_testing-policies.html

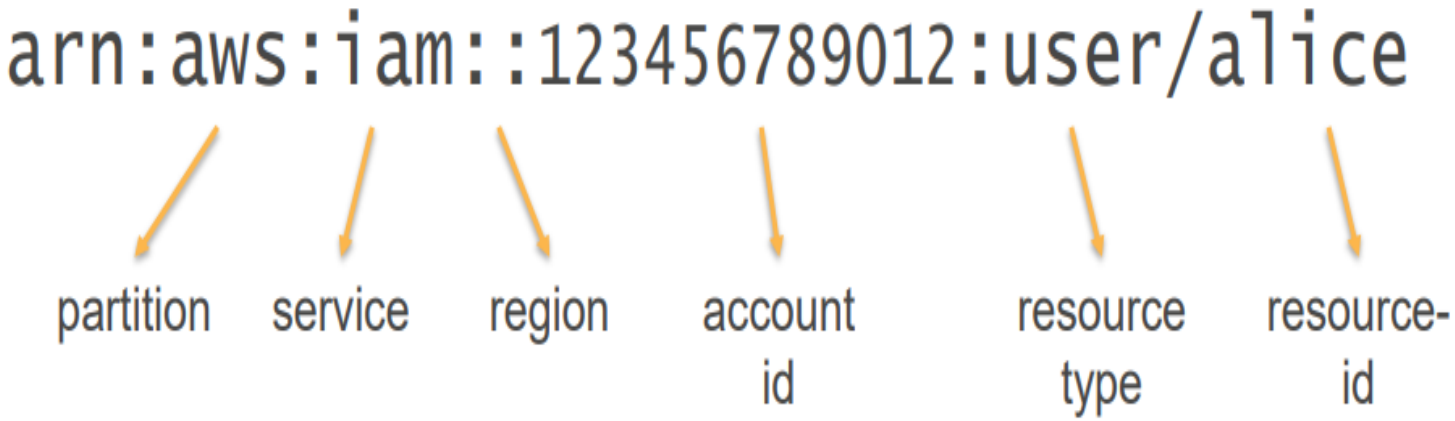
Policy Summary Table

- The IAM console includes policy summary tables that describe the access level, resources, and conditions that are allowed or denied for each service in a policy. Policies are summarized in three tables:
  - The policy summary table: It includes a list of services. Choose a service there to see the service summary.
  - The service summary table: It includes a list of the actions and associated permissions for the chosen service. Choose an action from that table to view the action summary.
  - The action summary table: It includes a list of resources and conditions for the chosen action.



Amazon Resource Name (ARN)

Uniquely identify resource and principal in AWS



Amazon Resource Name (ARN)

Uniquely identify resource and principal in AWS

```
arn:aws:iam::123456789012:user/alice

arn:aws:iam::123456789012:policy/db_admin

arn:aws:s3:::my_bucket

arn:aws:sqs:us-east-2:123456789012:order
```

Policy Document Structure (1/3)

Element	Description
Version	Current version of the policy language. You should always set the version element. Current version is "2012-10-17".
Statement	Permissions are allowed or denied using Statements. A policy document can have one or more statements
Effect	Specifies if a statement allows or denies permission Valid Values: Allow, Deny
Principal	Identity for which the statement applies. Implied when attached to the identity-based policy (for example, a user) Needs to be specified in resource-based policy and IAM Role trust policy Groups are not supported as principals in any policy

Example of Principal

Principal	Example
AWS account, root user	"Principal":{"AWS":"arn:aws:iam::123456789012:root"} "Principal":{"AWS":"123456789012"}
IAM Users	"Principal":{"AWS":"arn:aws:iam::123456789012:user/alice"}
IAM Roles	"Principal":{"AWS":"arn:aws:iam::123456789012:role/cross-acct"}
AWS Services	"Principal":{"Service":"elasticmapreduce.amazonaws.com"}
Anonymous users	"Principal": "*"
Federated Users	"Principal":{"Federated":"www.amazon.com"}
Assumed-role sessions	Use the role session name to uniquely identify a session when the same role is assumed by different principals or for different reasons "Principal":{"AWS":"arn:aws:sts::123456789012:assumed-role/role-name/role-session-name"}

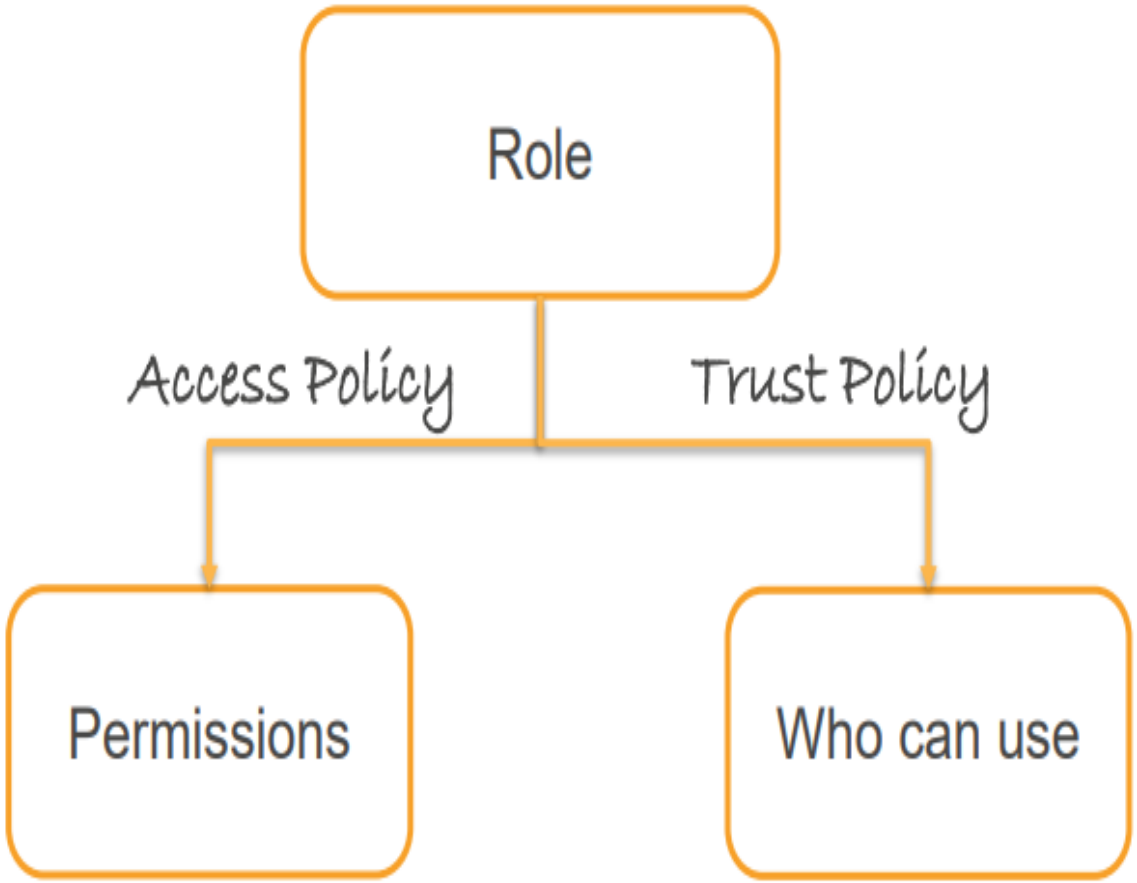
Policy Document Structure (2/3)

Element	Description
Action	Service API actions for which statement applies Example: "Action":"s3:Get*" - All S3 API Calls that have a Get Prefix "Action":"s3:*" - All S3 API Calls
Resource	Resource for which the statement applies Example: "Resource":"arn:aws:s3:::bucket_name" – Actions apply to specified bucket "Resource":"arn:aws:s3:::bucket_name/*" – Actions apply to objects stored in the specified bucket
Conditions	Additional conditions for fine grained access Example: IP Address, Authentication Mechanism

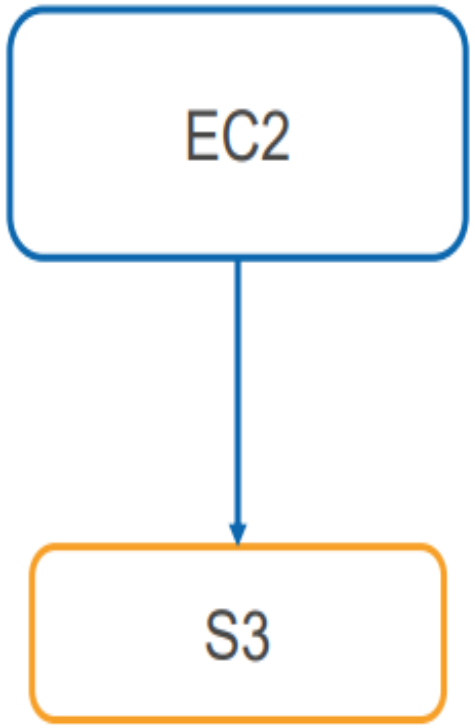
Policy Document Structure (3/3)

Element	Description
NotAction	Matches everything except specified API action Example: "NotAction":"s3:DeleteBucket" - All S3 actions except for deleting a bucket
NotResource	Match everything except the specified resource Example: "NotResource":["arn:aws:s3:::bucket_name", "arn:aws:s3:::bucket_name/*"]
NotPrincipal	Match all principals except for specified principal Example: "Effect":"Deny", "NotPrincipal":["AWS": "123456789012"]

Role Concepts



Application Access to AWS Resources



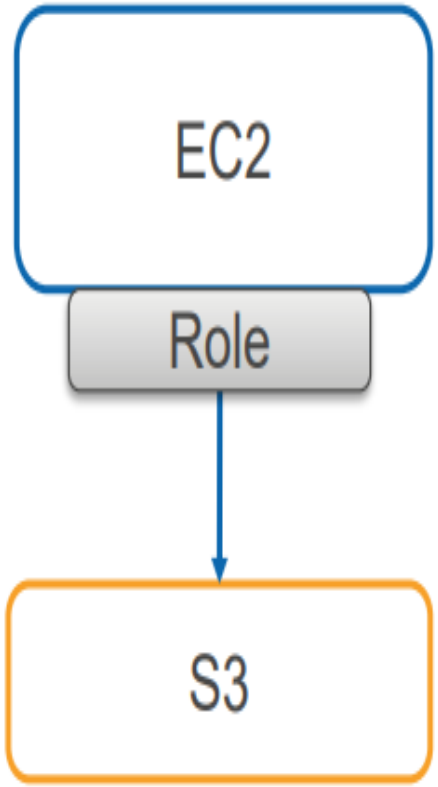
Access Key Credentials (Treat the server as a user):

- Generate Access Key, Secret Access Key
- Store the key in the EC2 instance
- EC2 uses the credentials to talk to services

Issues:

- Access Key Credentials are long-term (several days to months)
- Security risks due to long term credentials (accidental leakage, malicious users)
- Credential rotation is a problem at scale

Application Access to AWS Resources



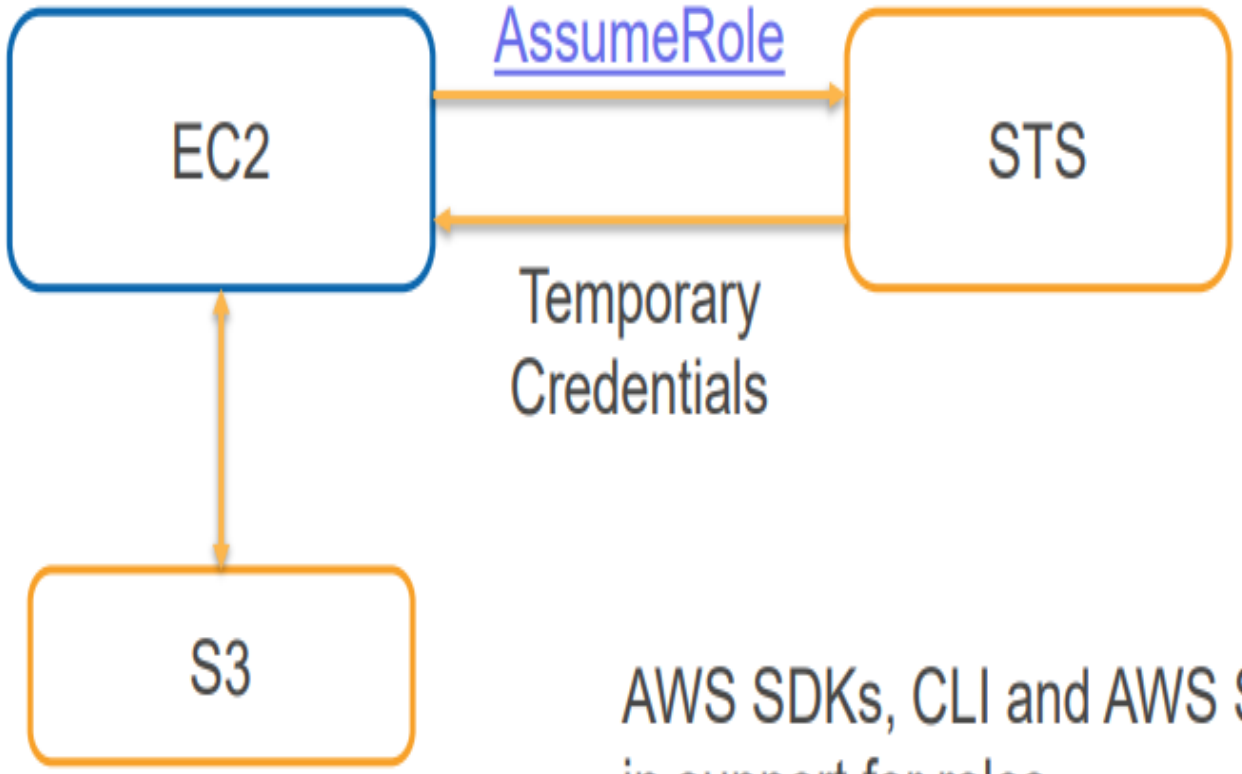
IAM Roles:

- Attach an IAM Role to the instance
- EC2 instance talks to metadata service to get temporary credentials for the role
- EC2 uses the temporary credentials to talk to services

Benefits:

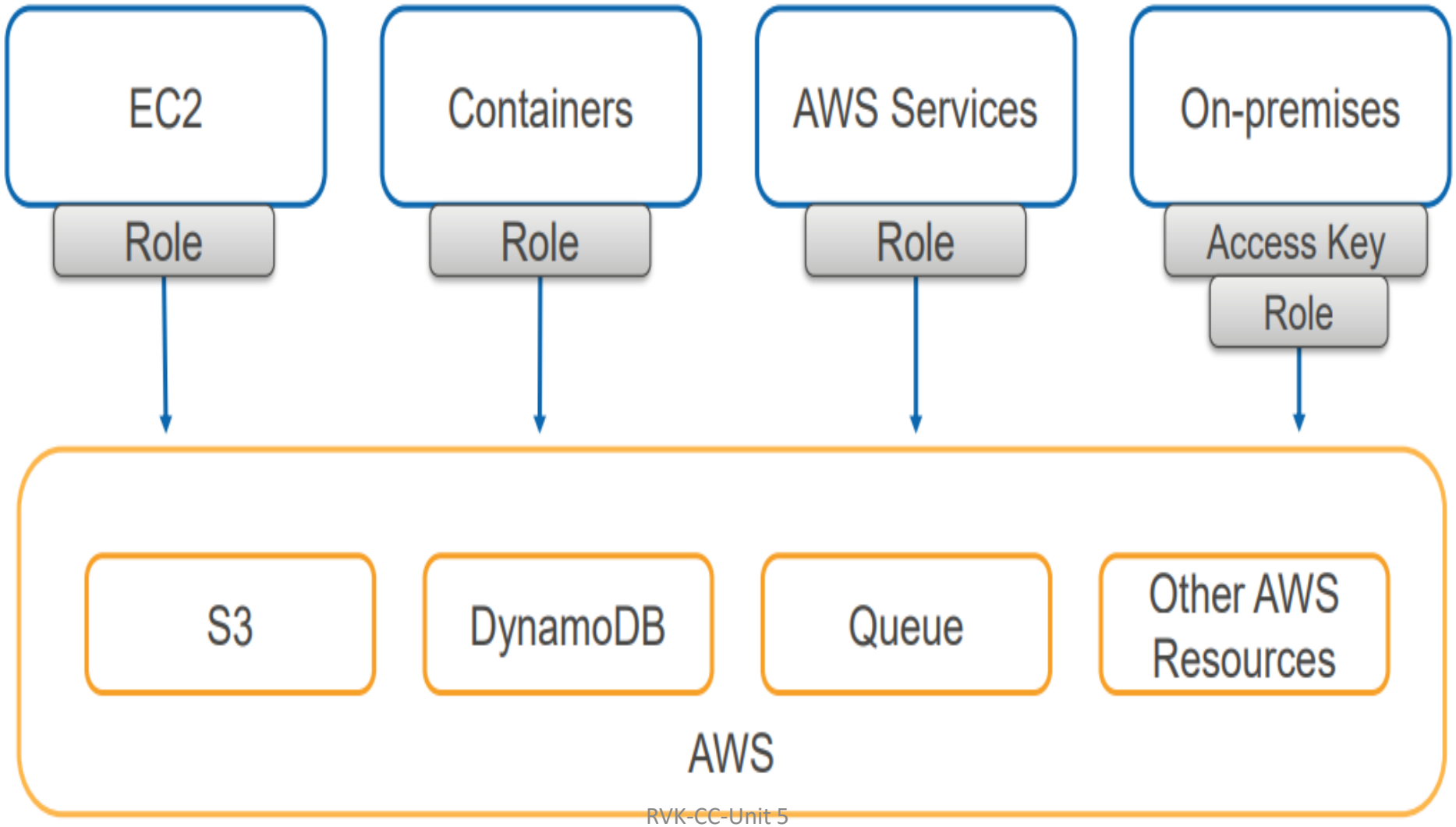
- No need to maintain credentials in the server
- Automatic Credential rotation – Credentials are valid only for a few hours (configurable)
- Reduced impact due to accidental leakage (credential is replaced every few hours)

Security Token Service (STS)

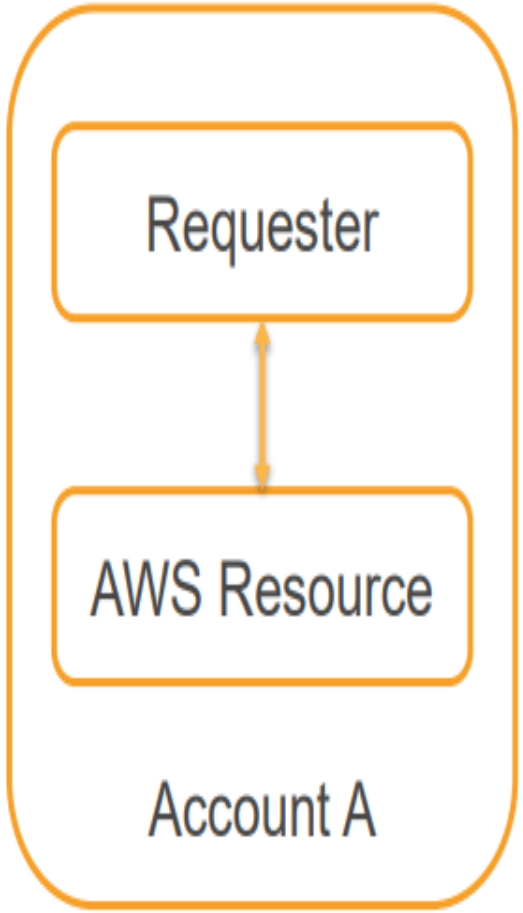


AWS SDKs, CLI and AWS Services have built-in support for roles

Application Access To Resources

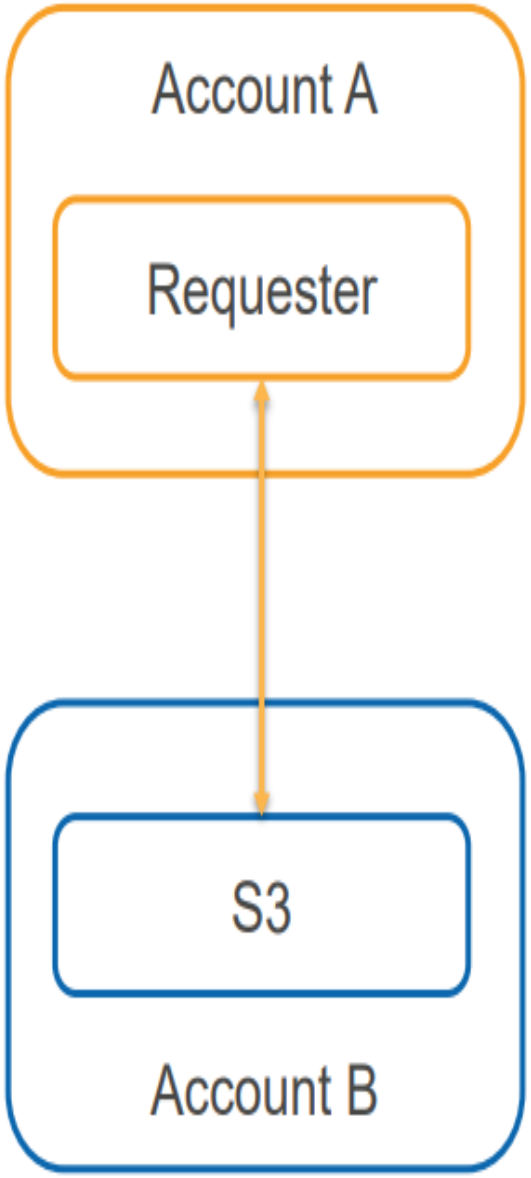


Same Account Access



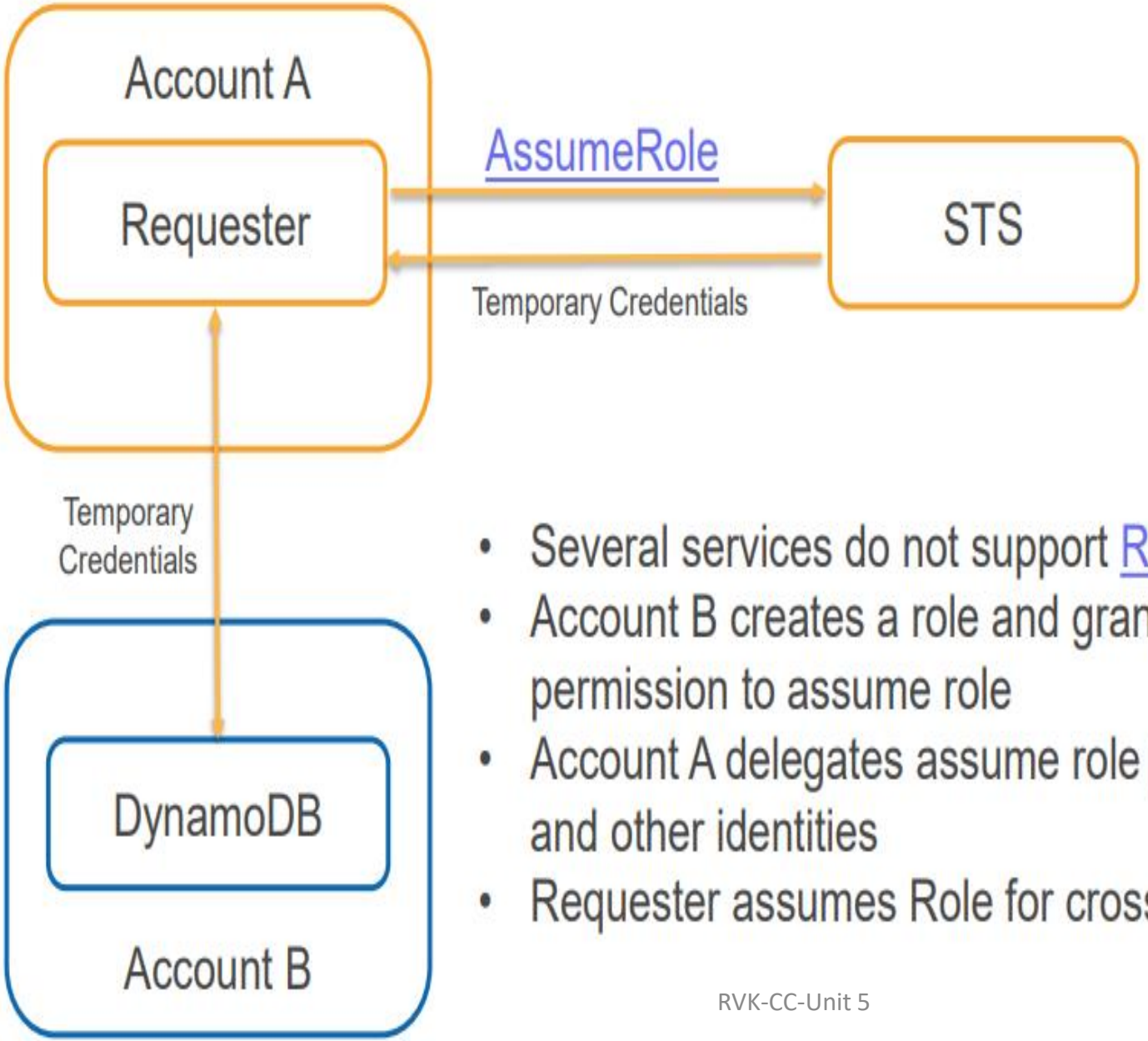
- Identity-based policies – for access to any AWS resource
- Resource-based policies – for access to supported resources
- Roles - for EC2 and Service integration

Cross Account Access using Resource Based Policy



- Resource Based Policies (for supported resources like S3, SQS, SNS, Lambda)
- Identity in Account A needs access to resource in Account B
  - Resource owner (Account B) grants permission to Account A
  - Account A delegate's permission to the other identities in the account (user, role)

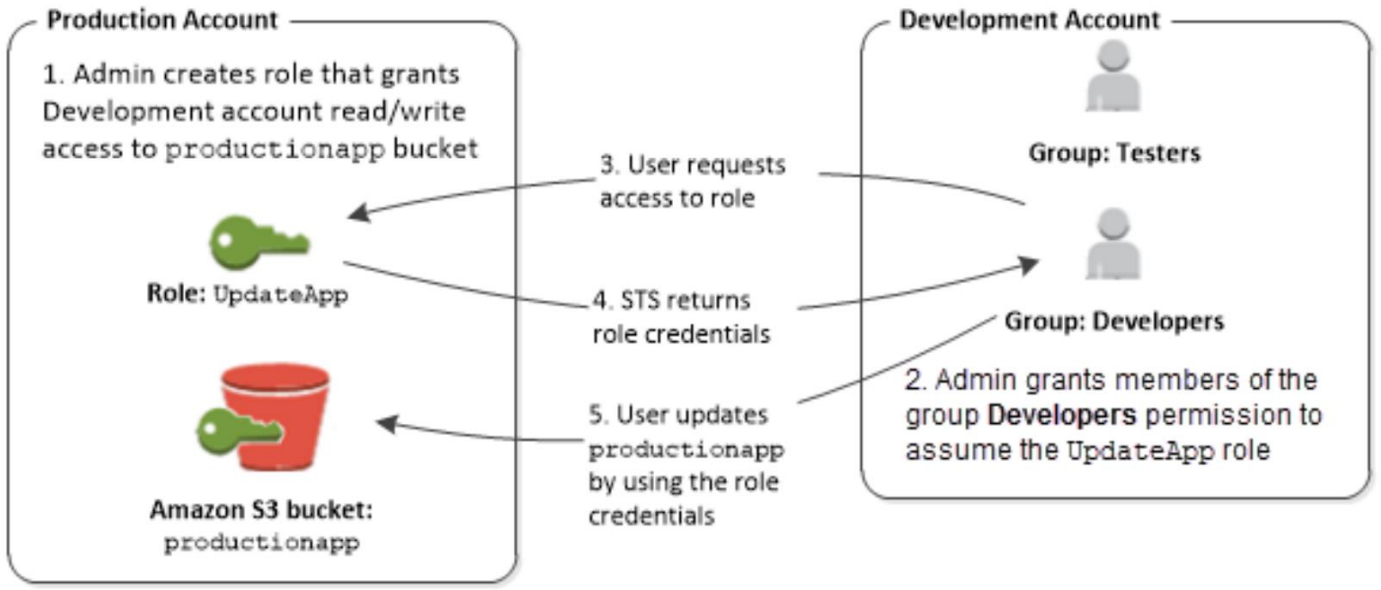
Cross Account Access Using Roles



- Several services do not support Resource Based Policy
- Account B creates a role and grants Account A permission to assume role
- Account A delegates assume role permission to users, and other identities
- Requester assumes Role for cross account access

Example: Scenario using separate development and production accounts

- Imagine that your organization has multiple AWS accounts to isolate a development environment from a production environment.
- Users in the development account might occasionally need to access resources in the production account. For example, you might need cross-account access when you are promoting an update from the development environment to the production environment.
- Although you could create separate identities (and passwords) for users who work in both accounts, managing credentials for multiple accounts makes identity management difficult.
- In the following figure, all users are managed in the development account, but some developers require limited access to the production account. The development account has two groups: Testers and Developers, and each group has its own policy.



Best Practices for Authorization using Cloud IAM

1. Federate to Enterprise and/or Cloud Identity Providers for User Authentication

- There is no need to create a separate island of authentication for the cloud. Authentication via single sign-on (SSO) and identity via Security Assertion Markup Language (SAML) provide users with a consistent identity across clouds, ideally exchanging attribute information as well, providing additional granularity in setting permissions.

2. Use Identity as a Perimeter by Using Accounts to Segregate Business Units

- IaaS/PaaS provider infrastructure is designed to be multitenant to keep one tenant strongly isolated from another. In the event one group experiences an attack or outage, it won't affect other accounts, improving overall resilience.

3. Require Strong Authentication For All Cloud Access

- In no case should a login to a cloud service or access to a cloud-based API be enabled based solely on username/password alone. Stronger authentication should be considered mandatory, and the leading cloud providers offer this capability built in. Ideally, an enterprise privileged access management system would be used for tighter visibility and control of administrative access.

Best Practices for Authorization using Cloud IAM (cont..)

4. Use Granular Permissions and Use IAM Roles

– Other than initial setup, no all-powerful administrative accounts should be used. Fine-grained permissions should be used to reduce the scope of capabilities of a specific user and used to meet separation of duties requirements of auditors and regulators. Further, cloud security can be greatly simplified through the use of standardized IAM roles and their use should be mandatory.

5. Evaluate Permissions Granted

– Continuous monitoring of permissions used by all IAM principles (users, network, storage and so on) at runtime versus the permissions that were initially provisioned should be reported to administrators for review. This is done to proactively trim permissions to reduce the surface area for attack.

6. Expand the Scope of Your IAM Program

– Many other entities other than users in public cloud IAM will be security principals as well (meaning they have rights defined in the cloud provider's IAM system). All VMs, containers, serverless functions, APIs, etc., in cloud-native applications will need an identity and associated permissions defined. Your IAM program must embrace this.

IAM Best Practices

- Identity and Credential Management
- Access Permission Management
- Delegation and Audit

1. Create Individual users

Benefits

- Unique set of credentials
- Individual permissions
- Granular control
- Easy to revoke access

Do

- Create IAM user for yourself
- Create individual users for other

Don't

- Distribute your AWS root credentials
- Use your root account user

2. Configure a strong password policy

Benefits

- Ensures your users and data are protected
- Easy way to enforce passwords complexity requirements
- Increase account resilience against brute force login attempts

Do

- Require password expiration of 90 days
- Require passwords with:
  - ✓ MIN password length of 14
  - ✓ at least one uppercase letter
  - ✓ at least one lowercase letter
  - ✓ at least one symbol
  - ✓ at least one number

3. Rotate security credentials regularly

Benefits

- Reduces the window of potential unauthorized access
- Ensures that data cannot be accessed with old keys which might have been lost or stolen

Do

- Use Access Key Last Used to identify and deactivate credentials that have been unused in 90 or greater days
- Enable credential rotation for IAM users
- Use Credential Report to audit credential rotation.

Enabling credential rotation for IAM users
(Enable access key rotation sample policy)

Access keys

```
{
 "Version":"2012-10-17",
  "Statement": [{
  "Effect": "Allow",
  "Action": [
      "iam:CreateAccessKey",
      "iam:DeleteAccessKey",
      "iam:ListAccessKeys",
      "iam:UpdateAccessKey"],
  "Resource":
   "arn:aws:iam::123456789012:
user/${aws:username}"
}]}
```

Steps to rotate access keys

1. While the first set of credentials is still active, create a second set of credentials, which will also be active by default.
2. Update all applications to use the new credentials.
3. Change the state of the first set of credentials to `Inactive`.
4. Using only the new credentials, confirm that your applications are working well.
5. Delete the first set of credentials.

amazon web services

4. Enable MFA for Privileged users

Benefits

- Provides an extra layer of protection
- Increase security for console and programmatic access

Do

- Enable MFA for your root account
- Virtual, Hardware, or SMS MFA
- Protect sensitive actions with MFA

5. Manage permissions with groups

Benefits

- Reduces the complexity of access management as number of users grow
- Reduces the opportunity for a user to accidently get excessive access
- Easy way to reassign permissions based on change in responsibility
- Easy way to update permissions for multiple users

Do

- Create groups that relate to job functions
- Attach policies to groups
- Use managed policies to logically manage permissions
- Manage group membership to assign permissions

6. Grant least privilege

Benefits

- Minimize chances of accidently performing privileged actions
- Easier to relax than tighten up
- More granular control

Do

- Start with a minimum set of permissions and grant additional permissions as necessary
- Restrict privileged access further with conditions
- Regularly check Access Advisor to restrict access
- Control access to specific resources using resource-based policy

7. Use IAM roles to share access

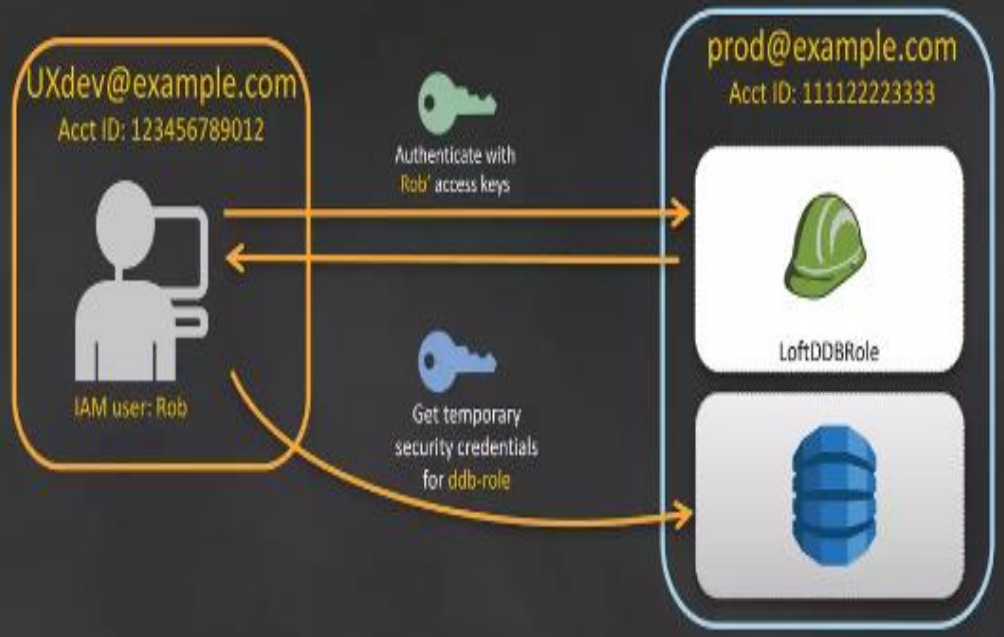
Benefits

- No need to share security credentials
- No need to store long-term credentials
- Control who has access

Do

- Use roles to delegate cross-account access
- Use roles to delegate access within an account
- Use roles to provide access for federated users

Use IAM roles for cross-account access



ddb-role trusts IAM users from the AWS account dev@example.com (123456789012)

```
{ "Statement": [
  {
   "Effect":"Allow",
   "Principal":{"AWS":"123456789012"},
   "Action":"sts:AssumeRole"
}]}
```

Permissions assigned to ddb-role

```
{ "Statement": [
  {   "Action":
    [
      "dynamodb:GetItem",
      "dynamodb:BatchGetItem",
      "dynamodb:DescribeTable",
      "dynamodb:ListTables"
    ],
    "Effect": "Allow",
    "Resource": "*"
}]}
```

Permissions assigned to Rob granting him permission to assume ddb-role in account B

Call AWS APIs using temporary security credentials of ddb-role

```
{ "Statement": [
  {
   "Effect": "Allow",
   "Action": "sts:AssumeRole",
   "Resource":
       "arn:aws:iam::111122223333:role/ddb-role"
}]}
```

8. Use IAM roles for Amazon EC2 instances

Benefits

- Easy to manage access keys on EC2 instances
- Automatic key rotation
- AWS SDKs fully integrated
- AWS CLI fully integrated

Do

- Use roles instead of long term credentials
- Assign least privilege to the application

9. Enable AWS CloudTrail to get logs of API calls

Benefits

- Enables API activity monitoring in your account
- Enables security analysis, resource tracking and compliance auditing

Do

- Ensure AWS CloudTrail is enabled in all regions
- Ensure AWS CloudTrail log file validation is enabled
- Ensure the Amazon S3 bucket of CloudTrail logs is not publicly accessible

10. Reduce or remove use of root

Benefits

- Reduces the risk of accidental changes and unintended disclosure of highly privileged credentials

Do

- Enable MFA for root account user
- If possible, remove root access keys
- Use a strong password for your account
- Use individual users

Configuration and Management Tools

For more details refer to : https://docs.ansible.com/ansible/latest/index.html

Configuration and Management Tool: Ansible

- A configuration management system is made up of several components like servers, storage, networking, and software.
- Its goals are:
  - The maintenance of its several components in known, determined states.
  - The description of the desired state for the system.
  - The automation software, which is responsible for making sure that the target systems and software are maintained in the desired state.
  - The reduction in configuration time

- Ansible is a configuration management platform that automates configuration of systems, deployment of software, and orchestration of more advanced IT tasks such as continuous deployments or zero downtime rolling updates.
- Ansible's main goals are simplicity and ease-of-use.
- It also has a strong focus on security and reliability, featuring a minimum of moving parts, usage of OpenSSH for transport (with other transports and pull modes as alternatives), and a language that is designed around auditability by humans–even those not familiar with the program.

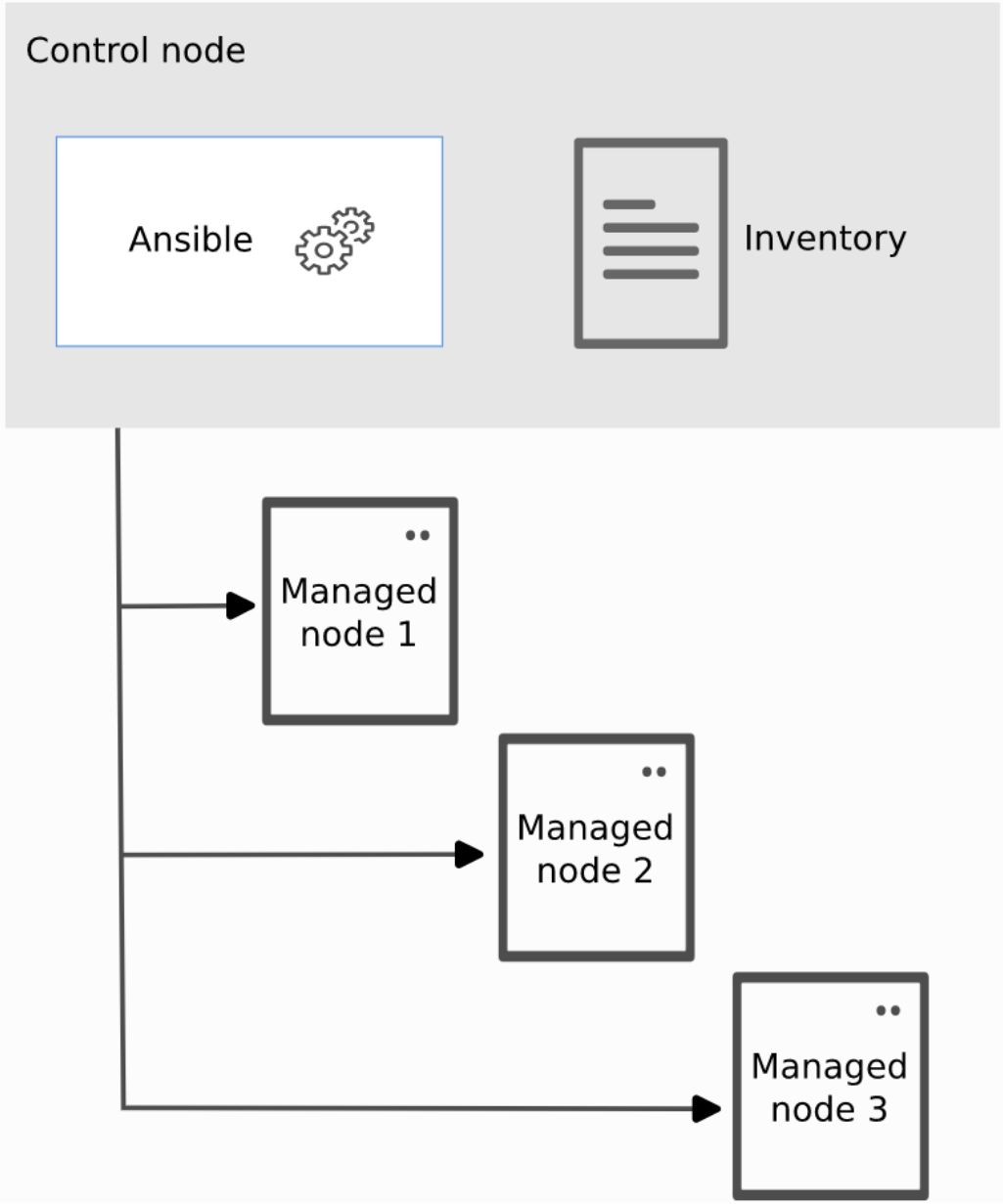
Ansible (cont..)

- It consolidates resources across multiple systems to manage them from a single platform rather than requiring management from one system at a time.
- When you use Ansible to configure these components, difficult manual tasks become repeatable and less vulnerable to error.

- **Benefits of Ansible:**
  - **Simplified Automation**
    - Ansible is a simple-to-use platform, easy to install and configure, with a very fast learning rate. In less than 30 minutes, it's possible to install and configure the system and execute ad hoc commands for servers to solve a specific problem: daylight saving time adjustments, time synchronization, root password change, updating servers, restarting services, and so on.
  - **Low Learning Curve**
    - Ansible is easy to deploy because it uses no agents or additional custom security infrastructure. It also leverages YAML, a simple language to describe your automation job via playbooks. Playbooks push the desired settings on the hosts defined in the inventory and can even be run ad hoc (via the command line, not requiring definitions in files).
  - **Automate Now**
    - From the moment you can ping the hosts through Ansible, you can start automating your environment. Begin with small tasks, following best practices, prioritizing tasks that add value to the business, solve major problems, and gain time and improving productivity.

Components of Ansible

- Ansible automates the management of remote systems and controls their desired state. A basic Ansible environment has three main components:
  - **Control node :** A system (laptops, shared desktops, and servers) on which Ansible is installed. You run Ansible commands such as **ansible** or **ansible-inventory** on a control node.
  - **Managed node:** A remote system, or host, that Ansible controls. These are the target devices (servers, network appliances or any computer) you aim to manage with Ansible.
  - **Inventory:** A list of managed nodes that are logically organized. You create an inventory on the control node to describe host deployments to Ansible. Your inventory can specify information specific to each node, like IP address. It is also used for assigning groups, that both allow for node selection in the Play and bulk variable assignment.



DevOps Architecture

For more details refer to : https://aws.amazon.com/devops/

https://www.ibm.com/cloud/architecture/architectures/devOpsArchitecture/reference-architecture

DevOps

- DevOps is a software development culture, approach and set of tools designed to promote integration and collaboration between traditionally distinct development and operations work and teams.

- It improves efficiency through the automation of repetitive manual tasks in the continuous integration and deployment (CI/CD pipelines) of software iterations.

  DevOps= Automation + Continuous Improvement + Coordination

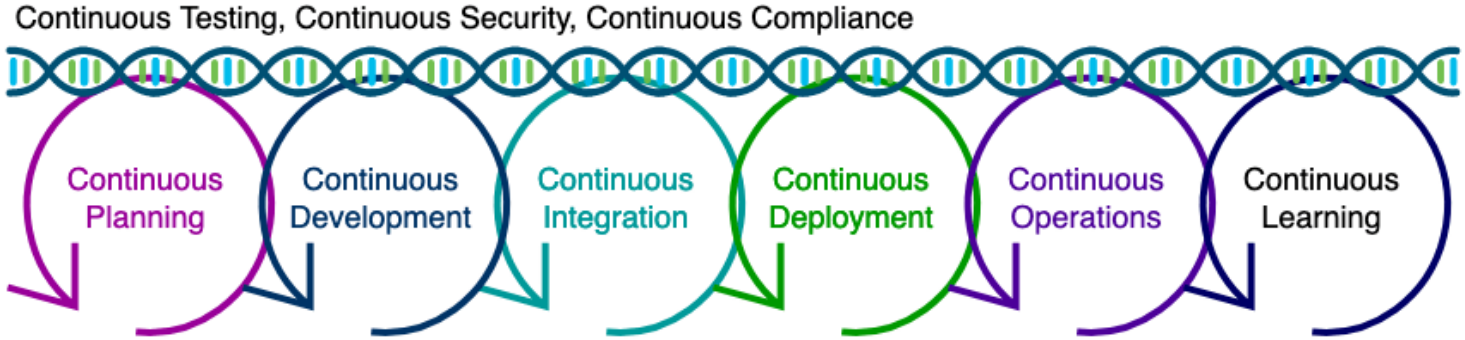
- The advantages of DevOps:
  - Better team collaboration through continuous software delivery and Simple deployment.
  - Improved scalability and effectiveness.
  - In the beginning, mistakes are corrected.
  - Less manual involvement and increased security.
  - Lifecycle of DevOps.

DevOps (cont..)

- DevOps is an approach based on agile and lean principles in which business owners, development, operations, and quality assurance team collaborate to deliver software in a continuous stable manner.
- DevOps is an environment that promotes cross practicality, shared business tasks and belief.
- DevOps is a movement that improves IT service delivery agility.
- DevOps is a culture that promotes better working relationship within the company.
- DevOps is a set of practices that provides rapid, reliable software delivery.

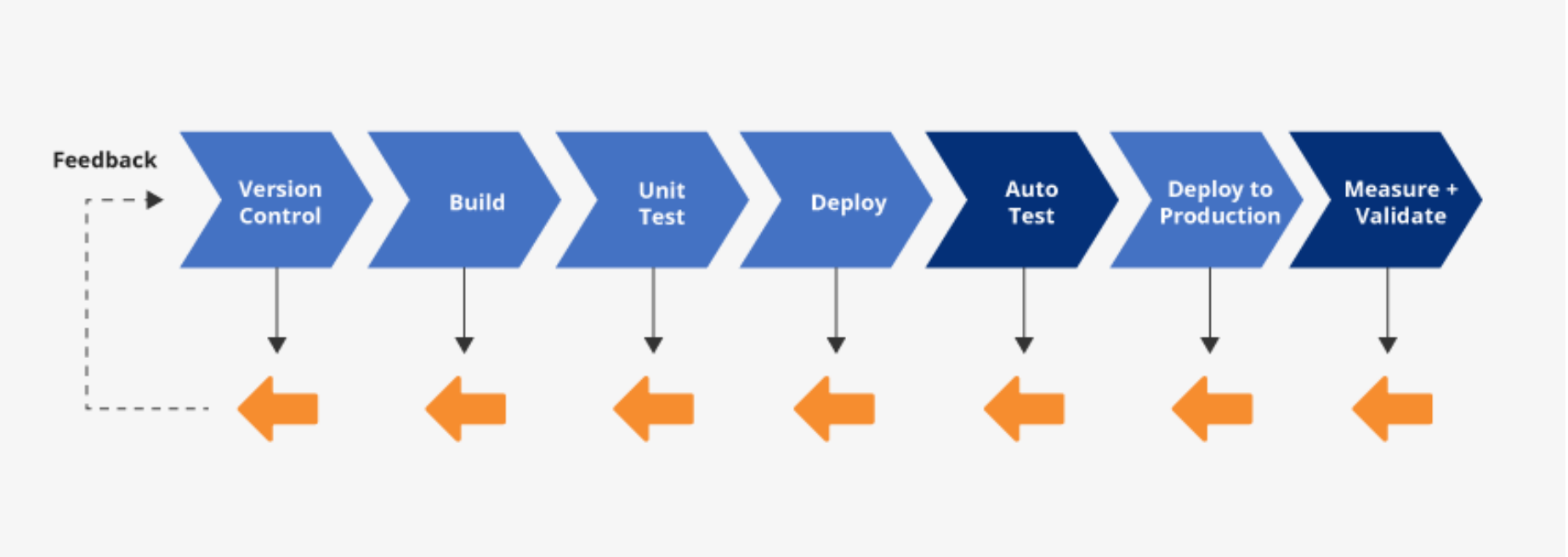
DevOps Essential Practices

- **Continuous Integration:** A software development practice where developers regularly merge their code changes into a central repository, after which automated builds and tests are run.
- **Continuous Delivery:** A software development practice where code changes are automatically built, tested, and prepared for a release to production.
- **Infrastructure as Code:** A practice in which infrastructure is provisioned and managed using code and software development techniques, such as version control, and continuous integration.
- **Monitoring and Logging:** Enables organizations to see how application and infrastructure performance impacts the experience of their product's end user.
- **Communication and Collaboration:** Practices are established to bring the teams closer and by building workflows and distributing the responsibilities for DevOps.
- **Security:** Should be a cross cutting concern. Your continuous integration and continuous delivery (CI/CD) pipelines and related services should be safeguarded and proper access control permissions should be set up.



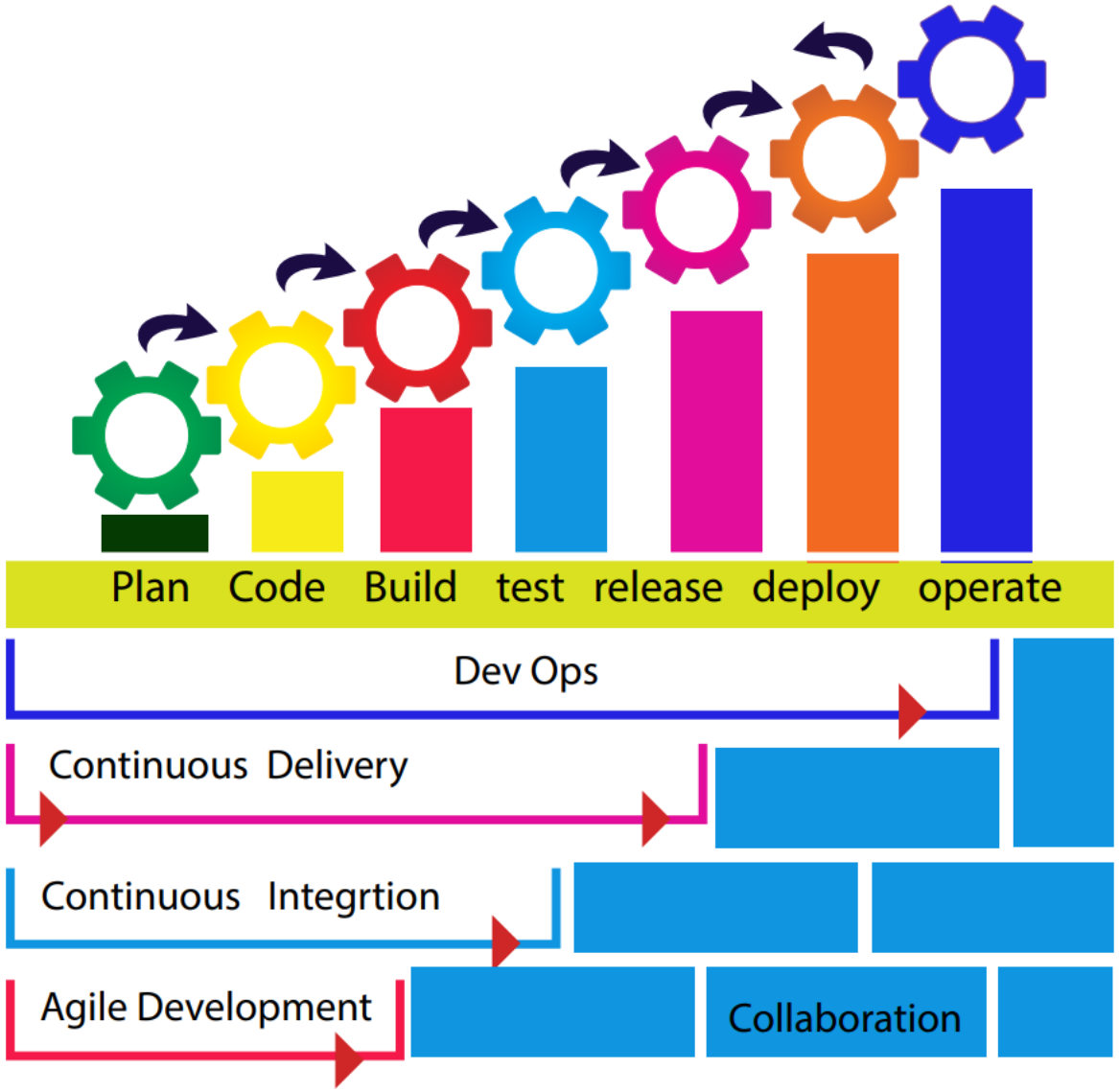
DevOps Pipeline Diagram

- A DevOps pipeline diagram is a collective illustration of the stages involved in a pipeline plus the automated procedures, tools, environments, and technologies that enable operations, non-developers, and developers, all breathing in an environment, to work together through the entire software development lifecycle.
- A DevOps pipeline diagram can also be called a CI/CD flow diagram.



DevOps Process

- Automate Provisioning – Infrastructure as Code
- Automate Builds – Continuous Integration
- Automate Deployments – Defined Deployment Pipeline and Continuous Deployments with appropriate configurations for the environments
- Automate Testing – Continuous Testing, Automated tests after each deployment
- Automate Monitoring – Proper monitors in place sending alerts
- Automate Metrics – Performance Metrics, Logs



Without automation there is no DevOps.

Thank you!