Specialized Cloud Mechanisms

A typical cloud technology architecture contains numerous moving parts to address distinct usage requirements of IT resources and solutions. Each mechanism covered in this chapter fulfills a specific runtime function in support of one or more cloud characteristics.

The following specialized cloud mechanisms are described in this chapter:

- Automated Scaling Listener
- Load Balancer
- SLA Monitor
- Pay-Per-Use Monitor
- Audit Monitor
- Failover System
- Hypervisor
- Resource Cluster
- Multi-Device Broker
- State Management Database

Automated Scaling Listener

The ***automated scaling listener*** mechanism is a service agent that monitors and tracks communications between cloud service consumers and cloud services for dynamic scaling purposes. Automated scaling listeners are deployed within the cloud, typically near the firewall, from where they automatically track workload status information. Workloads can be determined by the volume of cloud consumer-generated requests or via back-end processing demands triggered by certain types of requests. For example, a small amount of incoming data can result in a large amount of processing.

Automated scaling listeners can provide different types of responses to workload fluctuation conditions, such as:

- Automatically scaling IT resources out or in based on parameters previously defined by the cloud consumer (commonly referred to as ***autoscaling).***

Automatic notification of the cloud consumer when workloads exceed current thresholds or fall below allocated resources (Figure 8.1). This way, the cloud consumer can choose to adjust its current IT resource allocation.

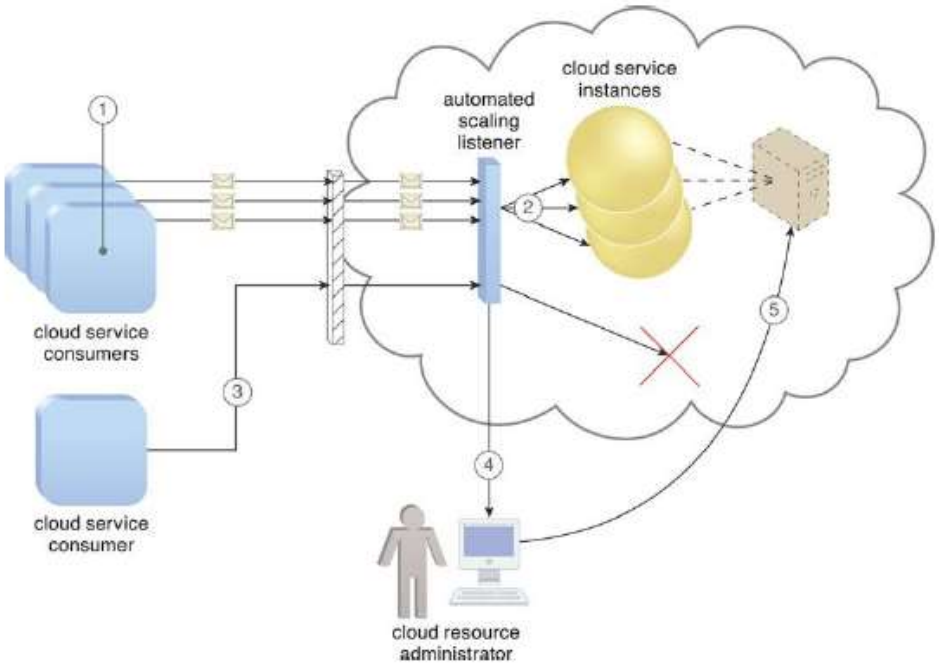


Figure 8.1. Three cloud service consumers attempt to access one cloud service simultaneously (1). The automated scaling listener scales out and initiates the creation of three redundant instances of the service (2). A fourth cloud service consumer attempts to use the cloud service (3). Programmed to allow up to only three instances of the cloud service, the automated scaling listener rejects the fourth attempt and notifies the cloud consumer that the requested workload limit has been exceeded (4). The cloud consumer's cloud resource administrator accesses the remote administration environment to adjust the provisioning setup and increase the redundant instance limit (5).

Load Balancer

A common approach to horizontal scaling is to balance a workload across two or more IT resources to increase performance and capacity beyond what a single IT resource can provide. The *load balancer* mechanism is a runtime agent with logic fundamentally based on this premise.

Beyond simple division of labor algorithms (Figure 8.5), load balancers can perform a range of specialized runtime workload distribution functions that include:

- *Asymmetric Distribution* - larger workloads are issued to IT resources with higher processing capacities *Workload Prioritization* - workloads are scheduled, queued, discarded, and distributed workloads according to their priority levels.
- *Content-Aware Distribution* - requests are distributed to different IT resources as dictated by the request content

A load balancer is programmed or configured with a set of performance and QoS rules and parameters with the general objectives of optimizing IT resource usage, avoiding overloads, and maximizing throughput.

The load balancer mechanisms can exist as a:

- multi-layer network switch
- dedicated hardware appliance

- dedicated software-based system (common in server operating systems)

service agent (usually controlled by cloud management software)

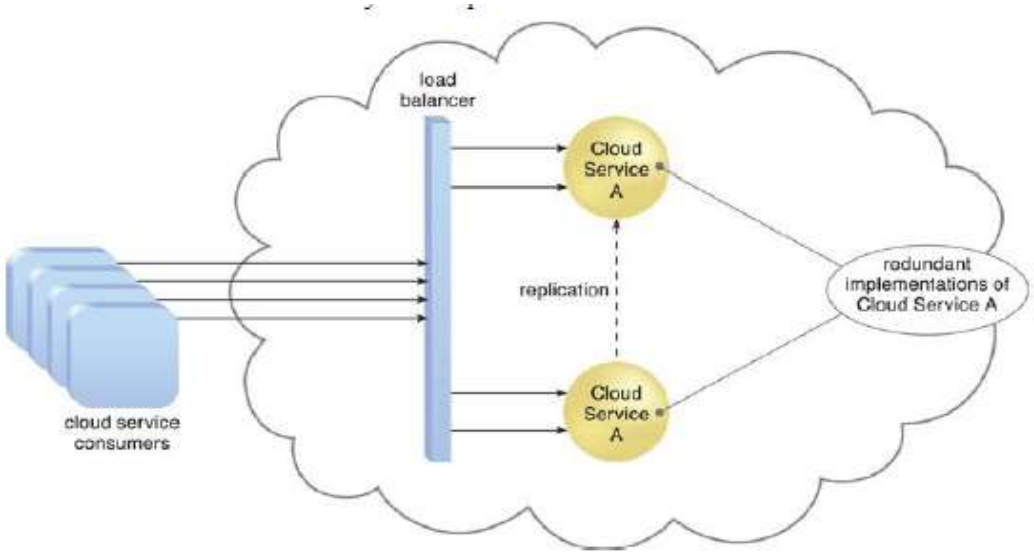


Figure 8.5. A load balancer implemented as a service agent transparently distributes incoming workload request messages across two redundant cloud service implementations, which in turn maximizes performance for the cloud service consumers.

The load balancer is typically located on the communication path between the IT resources generating the workload and the IT resources performing the workload processing. This mechanism can be designed as a transparent agent that remains hidden from the cloud service consumers, or as a proxy component that abstracts the IT resources performing their workload.

SLA Monitor

The SLA monitor mechanism is used to specifically observe the runtime performance of cloud services to ensure that they are fulfilling the contractual QoS requirements that are published in SLAs (Figure 8.7). The data collected by the SLA monitor is processed by an SLA management system to be aggregated into SLA reporting metrics. The system can proactively repair or failover cloud services when exception conditions occur, such as when the SLA monitor reports a cloud service as "down.

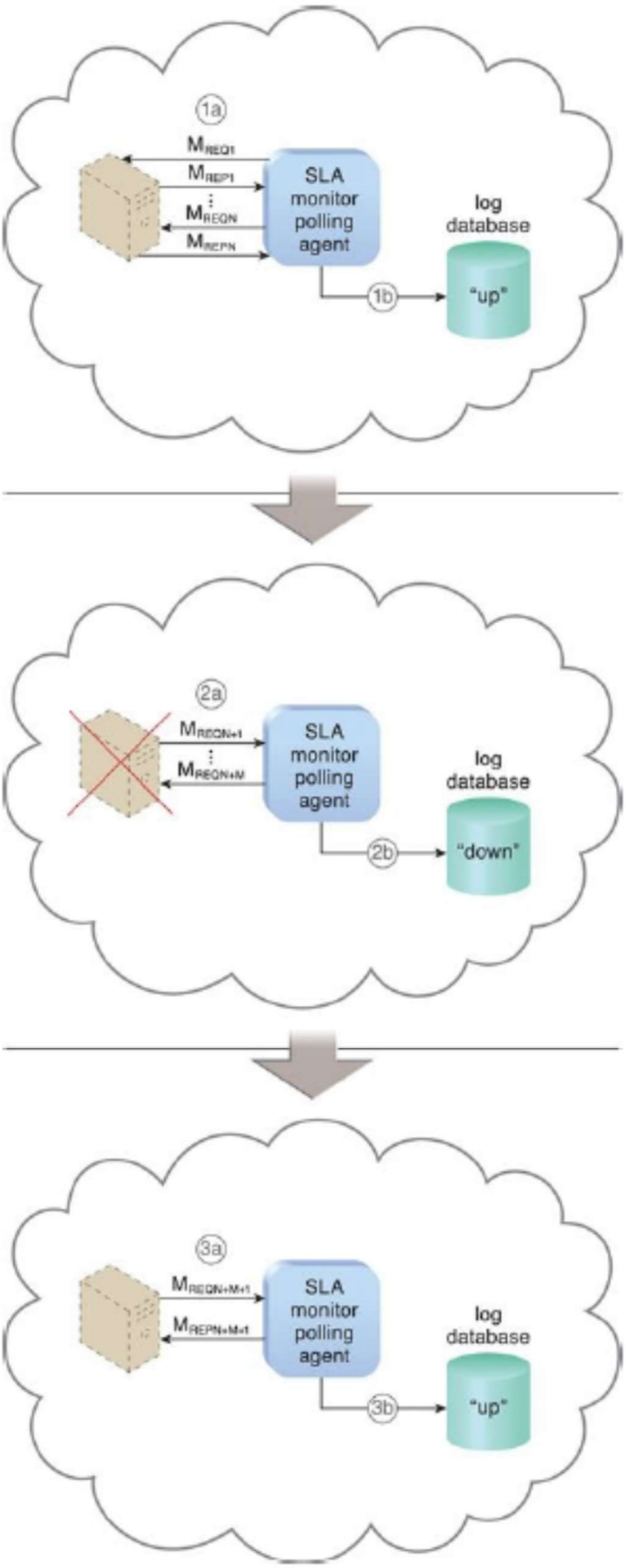

1a
SLA monitor polling agent
log database
1b
"up"
2a
SLA monitor polling agent
log database
2b
"down"
3a
SLA monitor polling agent
log database
3b
"up"

Figure 8.7. The SLA monitor polls the cloud service by sending over polling request messages (M_{REQ1} to M_{REQN}). The monitor receives polling response messages (M_{REP1} to M_{REPN}) that report that the service was "up" at each polling cycle (1a). The SLA monitor stores the "up" time—time period of all polling cycles 1 to N—in the log database (1b).
The SLA monitor polls the cloud service that sends polling request messages (M_{REQN+1} to M_{REQN+M}). Polling response messages are not received (2a). The response messages continue to time out, so the SLA monitor stores the "down" time—time period of all polling cycles N+1 to N+M—in the log database (2b).
The SLA monitor sends a polling request message ($M_{REQN+M+1}$) and receives the polling response message ($M_{REPN+M+1}$) (3a). The SLA monitor stores the "up" time in the log database (3b).

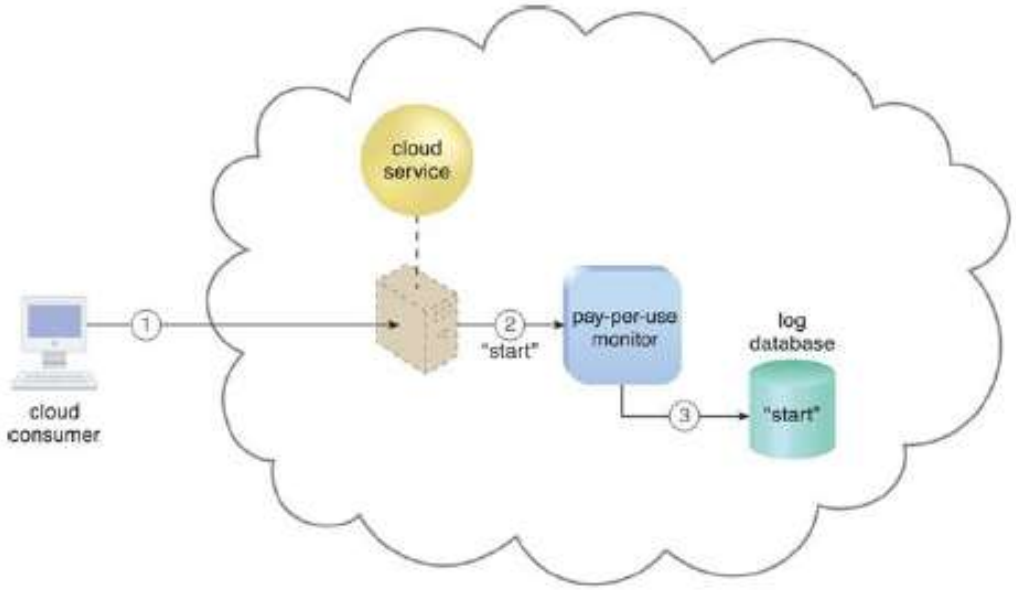
Pay-Per-Use Monitor

The ***pay-per-use monitor*** mechanism measures cloud-based IT resource usage in accordance with predefined pricing parameters and generates usage logs for fee calculations and billing purposes.
Some typical monitoring variables are:

- request/response message quantity
- transmitted data volume
- bandwidth consumption

The data collected by the pay-per-use monitor is processed by a billing management system that calculates the payment fees.

Figure 8.12 shows a pay-per-use monitor implemented as a resource agent used to determine the usage period of a virtual server.



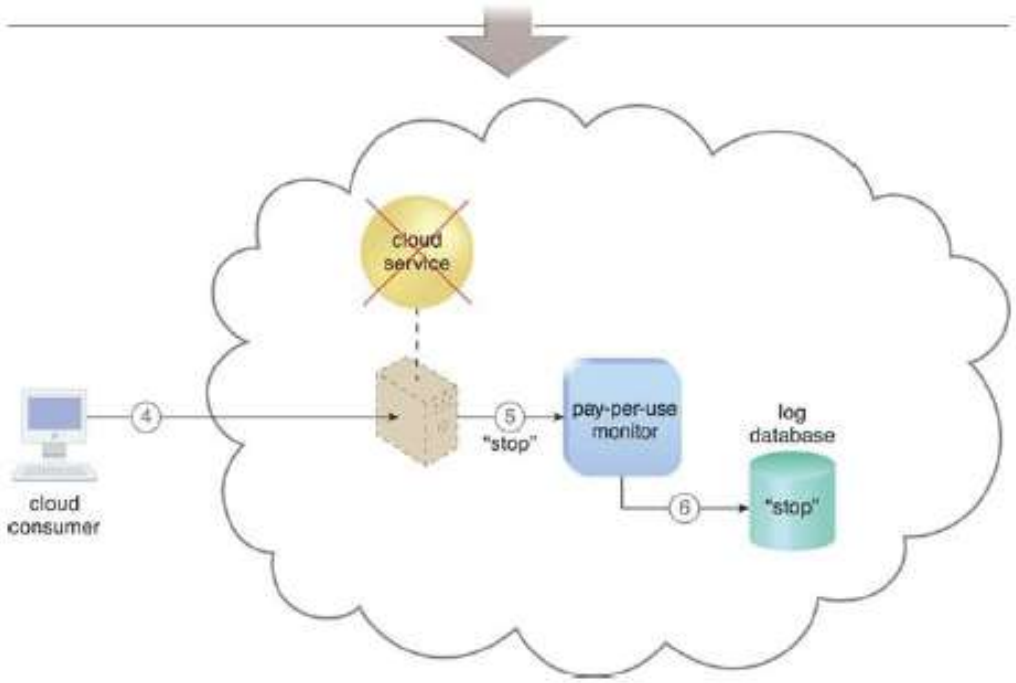


Figure 8.12. A cloud consumer requests the creation of a new instance of a cloud service (1). The IT resource is instantiated and the pay-per-use monitor receives a "start" event notification from the resource software (2). The pay-per-use monitor stores the value timestamp in the log database (3). The cloud consumer later requests that the cloud service instance be stopped (4). The pay-per-use monitor receives a "stop" event notification from the resource software (5) and stores the value timestamp in the log database (6).

Audit Monitor

The ***audit monitor*** mechanism is used to collect audit tracking data for networks and IT resources in support of (or dictated by) regulatory and contractual obligations. Figure 8.15 depicts an audit monitor implemented as a monitoring agent that intercepts "login" requests and stores the requestor's security credentials, as well as both failed and successful login attempts, in a log database for future audit reporting purposes.

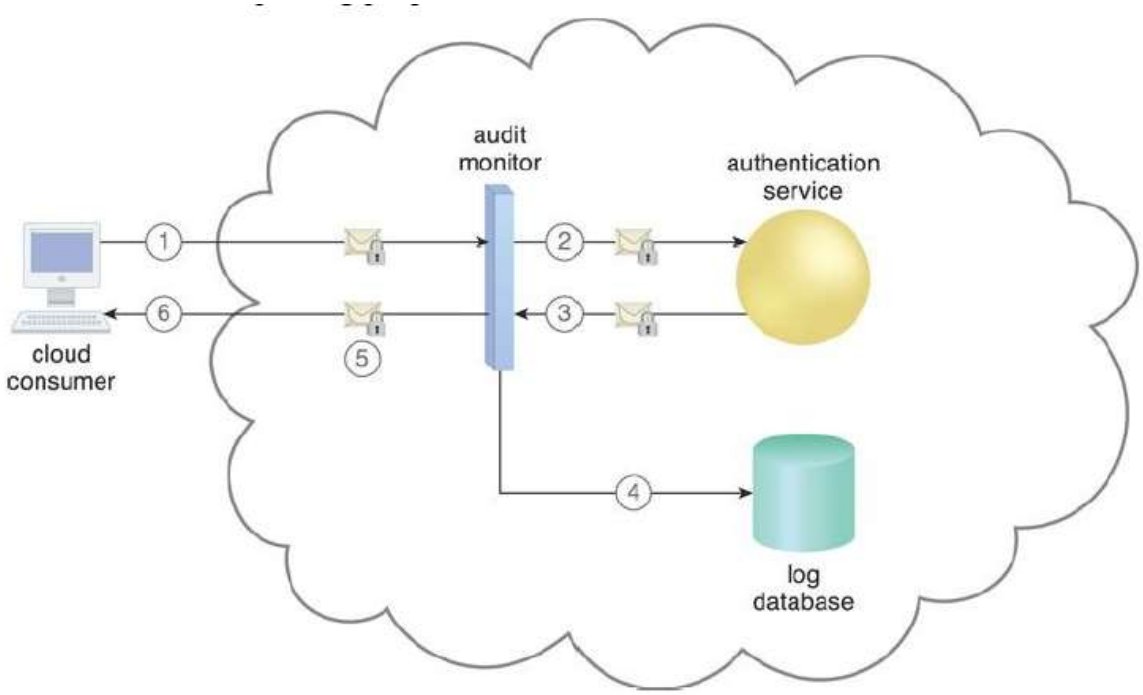


Figure 8.15. A cloud service consumer requests access to a cloud service by sending a login request message with security credentials (1). The audit monitor intercepts the message (2) and forwards it to the authentication service (3). The authentication service processes the security credentials. A response message is generated for the cloud service consumer, in addition to the results from the login attempt (4). The audit monitor intercepts the response message and stores the entire collected login event details in the log database, as per the organization's audit policy requirements (5). Access has been granted, and a response is sent back to the cloud service consumer (6).

Failover System

The ***failover system*** mechanism is used to increase the reliability and availability of IT resources by using established clustering technology to provide redundant implementations. A failover system is configured to automatically switch over to a redundant or standby IT resource instance whenever the currently active IT resource becomes unavailable.

Failover systems are commonly used for mission-critical programs and reusable services that can introduce a single point of failure for multiple applications. A failover system can span more than one geographical region so that each location hosts one or more redundant implementations of the same IT resource.

The resource replication mechanism is sometimes utilized by the failover system to provide redundant IT resource instances, which are actively monitored for the detection of errors and unavailability conditions.

Failover systems come in two basic configurations:

Active-Active

In an active-active configuration, redundant implementations of the IT resource actively serve the workload synchronously (Figure 8.17). Load balancing among active instances is required. When a failure

is detected, the failed instance is removed from the load balancing scheduler (Figure 8.18). Whichever IT resource remains operational when a failure is detected takes over the processing (Figure 8.19).

(Figure 8.19).

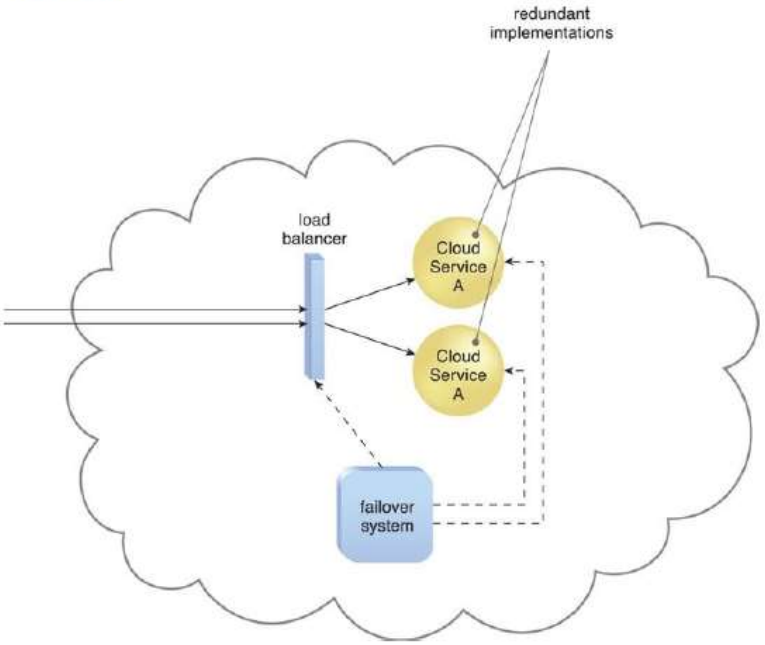


Figure 8.17. The failover system monitors the operational status of Cloud Service A.

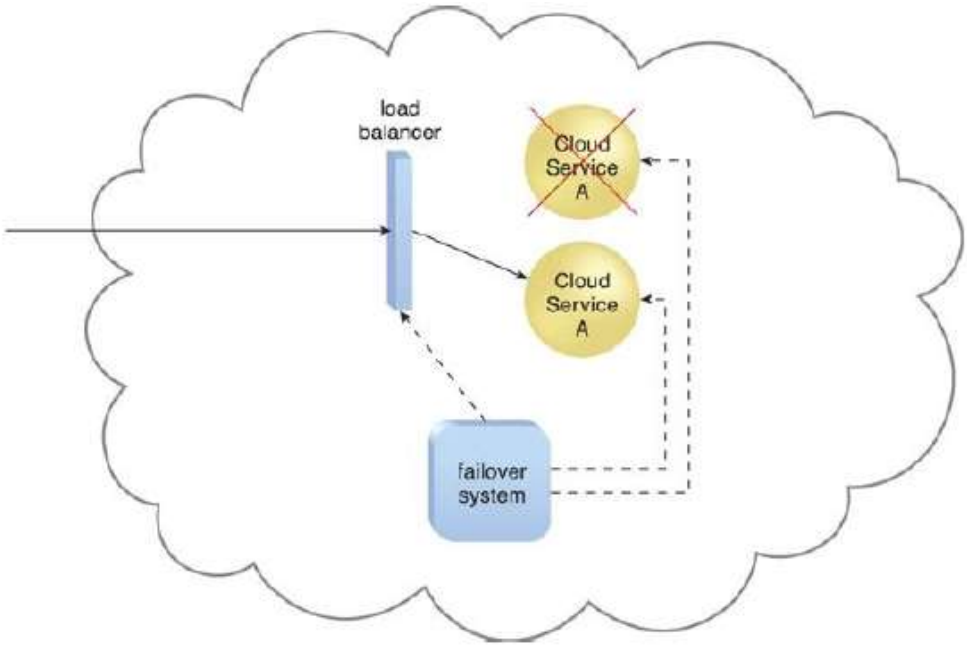


Figure 8.18. When a failure is detected in one Cloud Service A implementation, the failover system commands the load balancer to switch over the workload to the redundant Cloud Service A implementation.

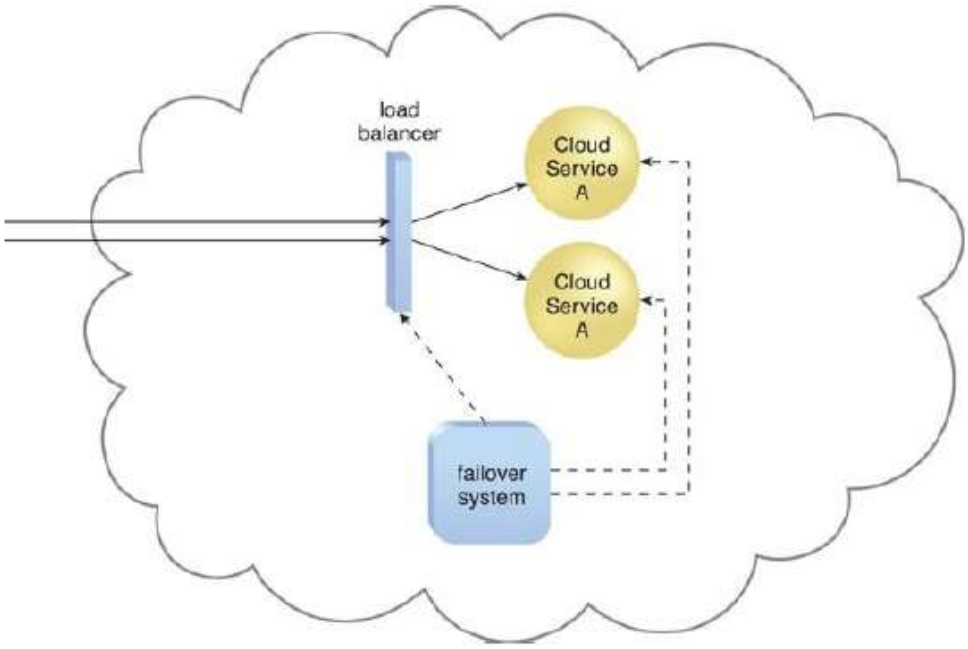


Figure 8.19. The failed Cloud Service A implementation is recovered or replicated into an operational cloud service. The failover system now commands the load balancer to distribute the workload again.

Active-Passive

In an active-passive configuration, a standby or inactive implementation is activated to take over the processing from the IT resource that becomes unavailable, and the corresponding workload is redirected to the instance taking over the operation (Figures 8.20 to 8.22).

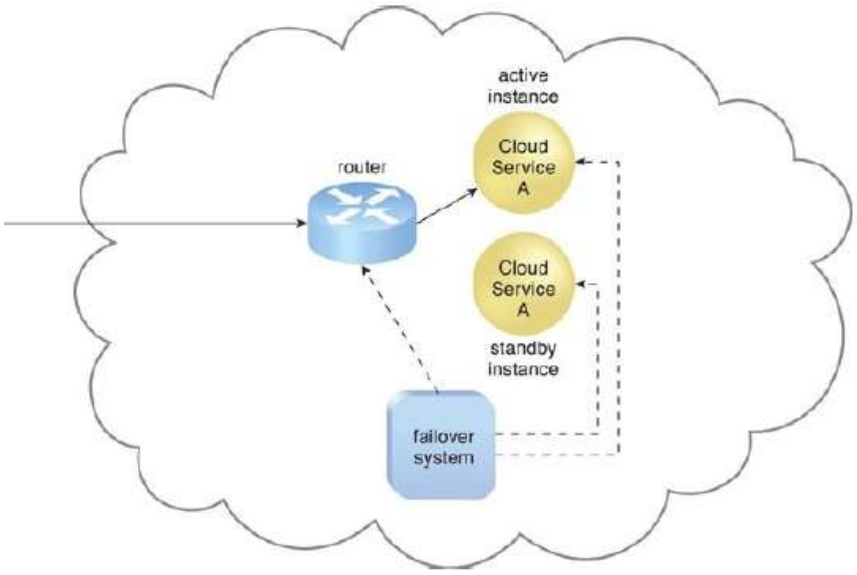


Figure 8.20. The failover system monitors the operational status of Cloud Service A. The Cloud Service A implementation acting as the active instance is receiving cloud service consumer requests.

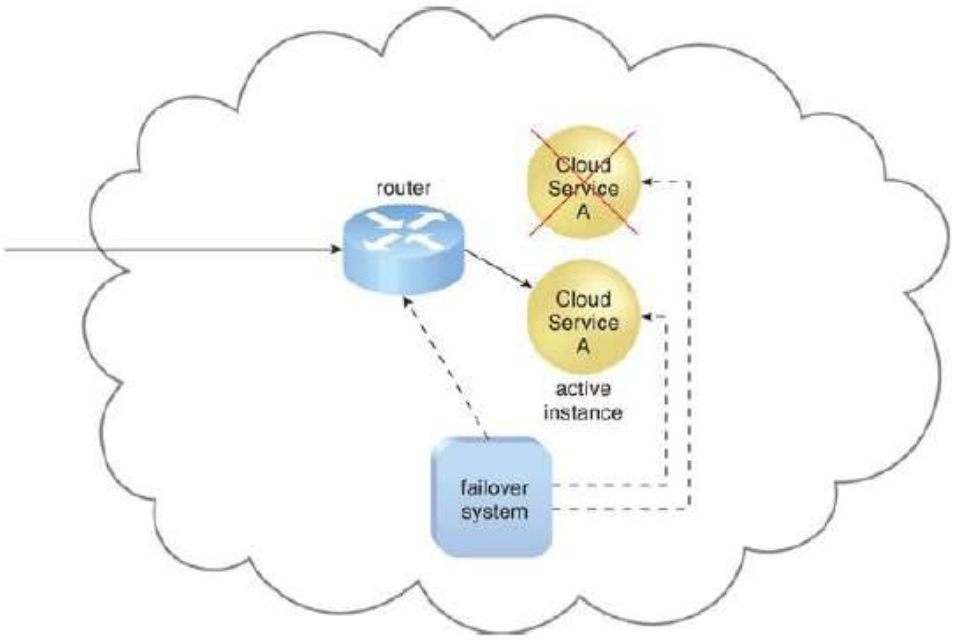


Figure 8.21. The Cloud Service A implementation acting as the active instance encounters a failure that is detected by the failover system, which subsequently activates the inactive Cloud Service A implementation and redirects the workload toward it. The newly invoked Cloud Service A implementation now assumes the role of active instance.

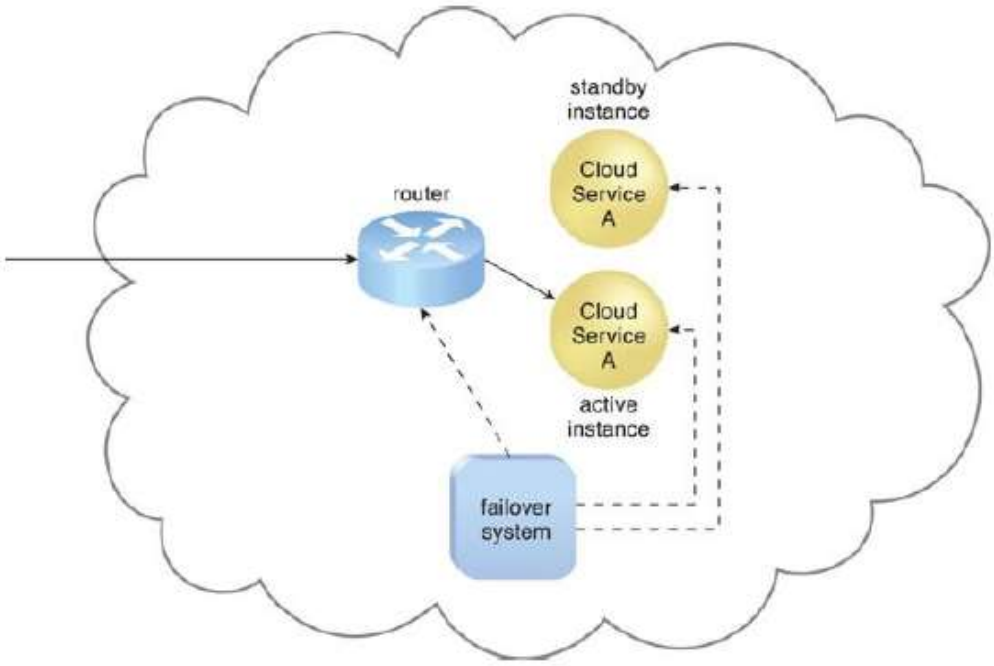


Figure 8.22. The failed Cloud Service A implementation is recovered or replicated an operational cloud service, and is now positioned as the standby instance, while the previously invoked Cloud Service A continues to serve as the active instance.

Some failover systems are designed to redirect workloads to active IT resources that rely on specialized load balancers that detect failure conditions and exclude failed IT resource instances from the workload

distribution. This type of failover system is suitable for IT resources that do not require execution state management and provide stateless processing capabilities. In technology architectures that are typically based on clustering and virtualization technologies, the redundant or standby IT resource implementations are also required to share their state and execution context. A complex task that was executed on a failed IT resource can remain operational in one of its redundant implementations.

Hypervisor

The ***hypervisor*** mechanism is a fundamental part of virtualization infrastructure that is primarily used to generate virtual server instances of a physical server. A hypervisor is generally limited to one physical server and can therefore only create virtual images of that server (Figure 8.27). Similarly, a hypervisor can only assign virtual servers it generates to resource pools that reside on the same underlying physical server. A hypervisor has limited virtual server management features, such as increasing the virtual server's capacity or shutting it down. The VIM provides a range of features for administering multiple hypervisors across physical servers.

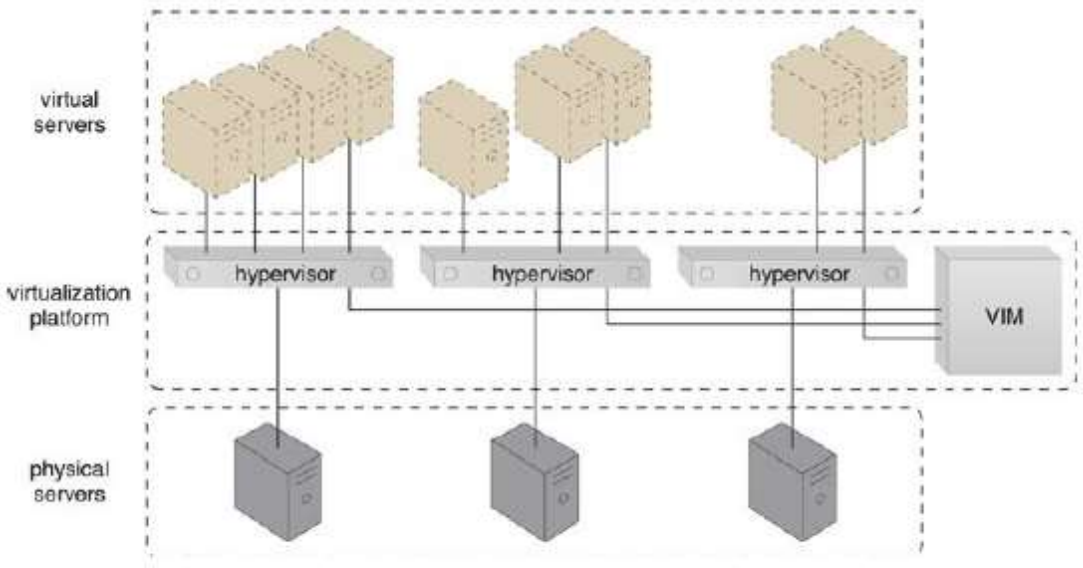


Figure 8.27. Virtual servers are created via individual hypervisor on individual physical servers. All three hypervisors are jointly controlled by the same VIM.

Hypervisor software can be installed directly in bare-metal servers and provides features for controlling, sharing and scheduling the usage of hardware resources, such as processor power, memory, and I/O. These can appear to each virtual server's operating system as dedicated resources.

Resource Cluster

Cloud-based IT resources that are geographically diverse can be logically combined into groups to improve their allocation and use. The ***resource cluster*** mechanism (Figure 8.30) is used to group multiple IT resource instances so that they can be operated as a single IT resource. This increases the combined computing capacity, load balancing, and availability of the clustered IT resources.

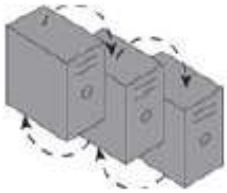


Figure 8.30. The curved dashed lines are used to indicate that IT resources are clustered.

Resource cluster architectures rely on high-speed dedicated network connections, or cluster nodes, between IT resource instances to communicate about workload distribution, task scheduling, data sharing, and system synchronization. A cluster management platform that is running as distributed middleware in all of the cluster nodes is usually responsible for these activities. This platform implements a coordination function that allows distributed IT resources to appear as one IT resource, and also executes IT resources inside the cluster.

Common resource cluster types include:

- ***Server Cluster*** - Physical or virtual servers are clustered to increase performance and availability. Hypervisors running on different physical servers can be configured to share virtual server execution state (such as memory pages and processor register state) in order to establish clustered virtual servers. In such configurations, which usually require physical servers to have access to shared storage, virtual servers are able to live- migrate from one to another. In this process, the virtualization platform suspends the execution of a given virtual server at one physical server and resumes it on another physical server. The process is transparent to the virtual server operating system and can be used to increase scalability by live-migrating a virtual server that is running at an overloaded physical server to another physical server that has suitable capacity.
- ***Database Cluster*** - Designed to improve data availability, this high- availability resource cluster has a synchronization feature that maintains the consistency of data being stored at different storage devices used in the cluster. The redundant capacity is usually based on an active-active or active-passive failover system committed to maintaining the synchronization conditions.
- ***Large Dataset Cluster*** - Data partitioning and distribution is implemented so that the target datasets can be efficiently partitioned without compromising data integrity or computing accuracy. Each cluster node processes workloads without communicating with other nodes as much as in other cluster types.
-

Many resource clusters require cluster nodes to have almost identical computing capacity and characteristics in order to simplify the design of and maintain consistency within the resource cluster architecture. The cluster nodes in high- availability cluster architectures need to access and share common storage IT resources. This can require two layers of communication between the nodes— one for accessing the storage device and another to execute IT resource orchestration (Figure 8.31). Some resource clusters are designed with more loosely coupled IT resources that only require the network layer (Figure 8.32)

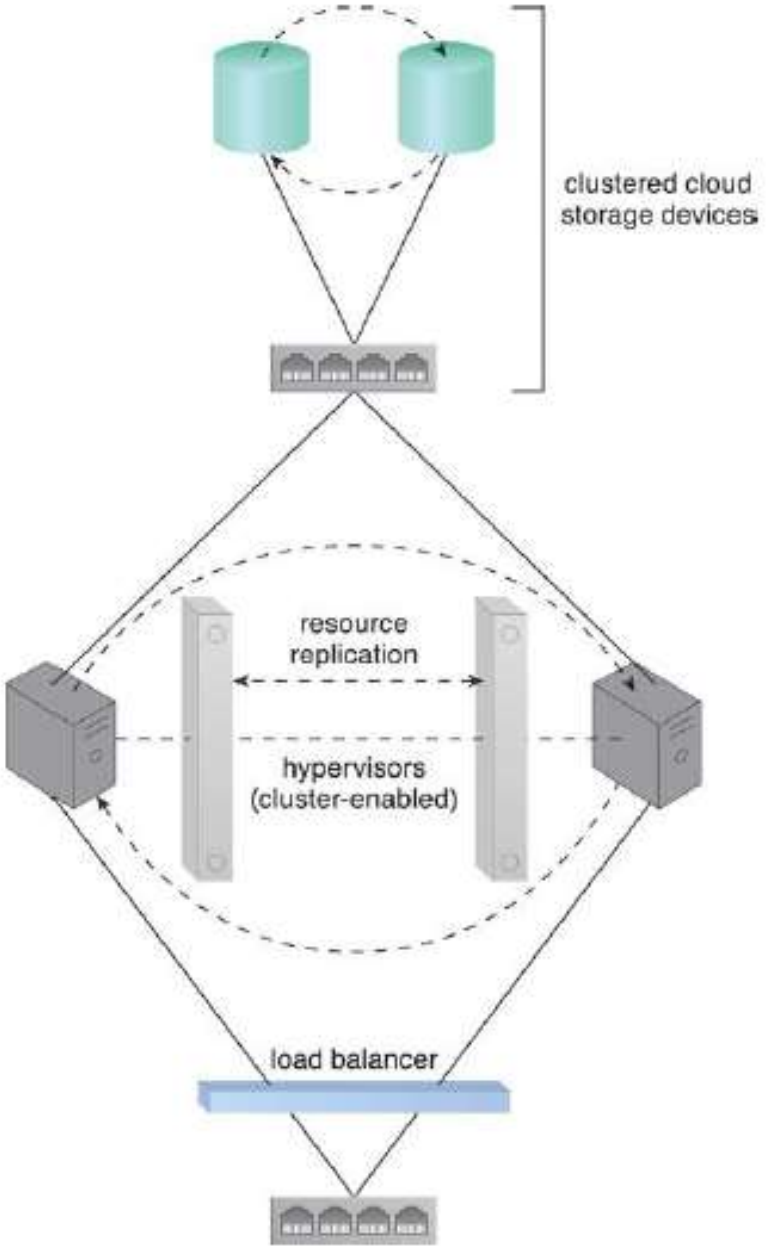


Figure 8.31. Load balancing and resource replication are implemented through a cluster-enabled hypervisor. A dedicated storage area network is used to connect the clustered storage and the clustered servers, which are able to share common cloud storage devices. This simplifies the storage replication process, which is independently carried out at the storage cluster. (See the *Hypervisor Clustering*

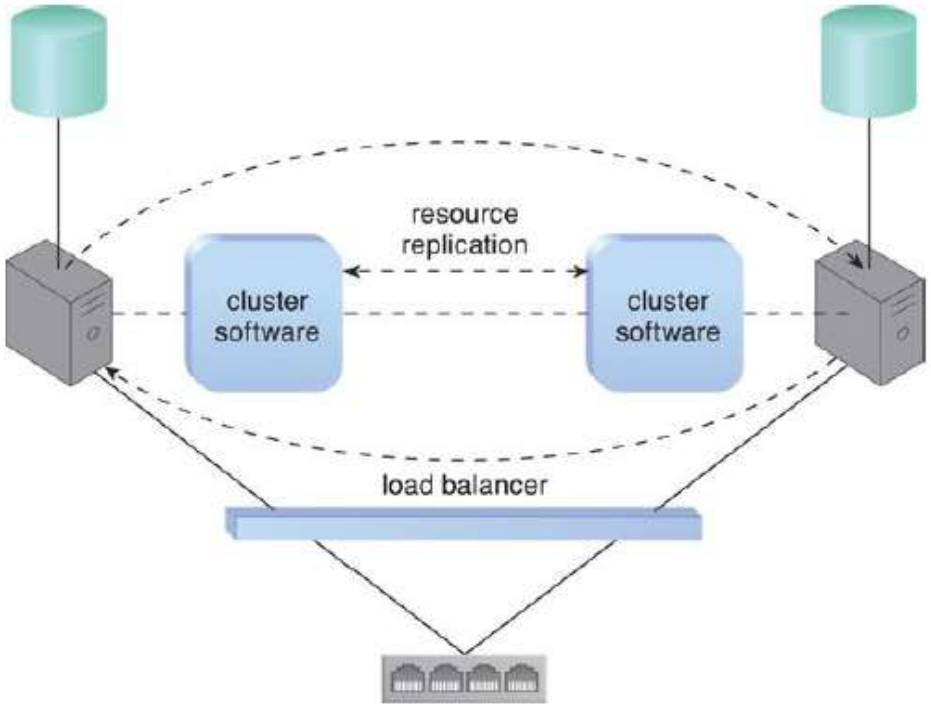


Figure 8.32. A loosely coupled server cluster that incorporates a load balancer. There is no shared storage. Resource replication is used to replicate cloud storage devices through the network by the cluster software.

There are two basic types of resource clusters:

- ***Load Balanced Cluster*** - This resource cluster specializes in distributing workloads among cluster nodes to increase IT resource capacity while preserving the centralization of IT resource management. It usually implements a load balancer mechanism that is either embedded within the cluster management platform or set up as a separate IT resource.
- ***HA Cluster*** - A high-availability cluster maintains system availability in the event of multiple node failures, and has redundant implementations of most or all of the clustered IT resources. It implements a failover system mechanism that monitors failure conditions and automatically redirects the workload away from any failed nodes.

The provisioning of clustered IT resources can be considerably more expensive than the provisioning of individual IT resources that have an equivalent computing capacity.

Multi-Device Broker

An individual cloud service may need to be accessed by a range of cloud service consumers differentiated by their hosting hardware devices and/or communication requirements. To overcome incompatibilities between a cloud service and a disparate cloud service consumer, mapping logic needs to be created to transform (or convert) information that is exchanged at runtime.

The ***multi-device broker*** mechanism is used to facilitate runtime data transformation so as to make a cloud service accessible to a wider range of cloud service consumer programs and devices (Figure 8.35).

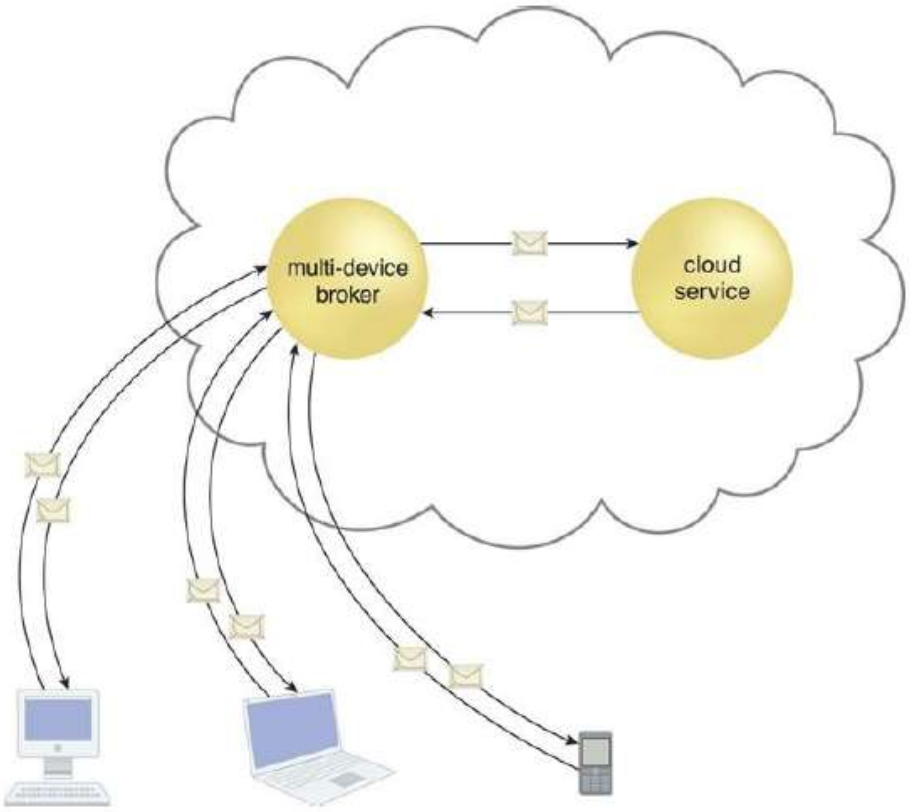


Figure 8.35. A multi-device broker contains the mapping logic necessary to transform data exchanges between a cloud service and different types of cloud service consumer devices. This scenario depicts the multi-device broker as a cloud service with its own API. This mechanism can also be implemented as a service agent that intercepts messages at runtime to perform necessary transformations.

Multi-device brokers commonly exist as gateways or incorporate gateway components, such as:

- ***XML Gateway*** - transmits and validates XML data
- ***Cloud Storage Gateway*** - transforms cloud storage protocols and encodes storage devices to facilitate data transfer and storage
- ***Mobile Device Gateway*** - transforms the communication protocols used

by mobile devices into protocols that are compatible with a cloud service The levels at which transformation logic can be created include:

- transport protocols
- messaging protocols
- storage device protocols
- data schemas/data models

For example, a multi-device broker may contain mapping logic that coverts both transport and messaging protocols for a cloud service consumer accessing a cloud service with a mobile device.

State Management Database

A ***state management database*** is a storage device that is used to temporarily persist state data for software programs. As an alternative to caching state data in memory, software programs can off-load state data to the database in order to reduce the amount of runtime memory they consume (Figures 8.37

and 838). By doing so, the software programs and the surrounding infrastructure are more scalable. State management databases are commonly used by cloud services, especially those involved in long-running runtime activities.

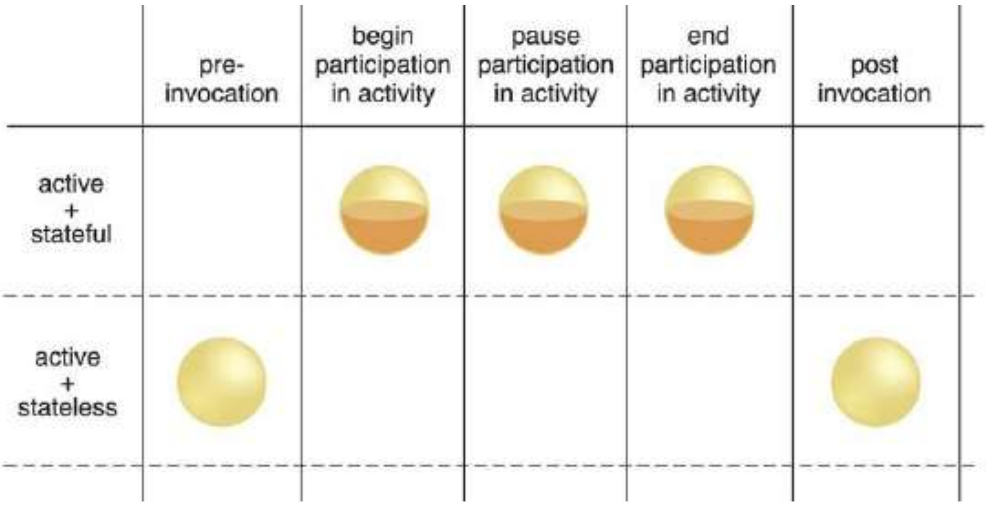


Figure 8.37. During the lifespan of a cloud service instance it may be required to remain stateful and keep state data cached in memory, even when idle.

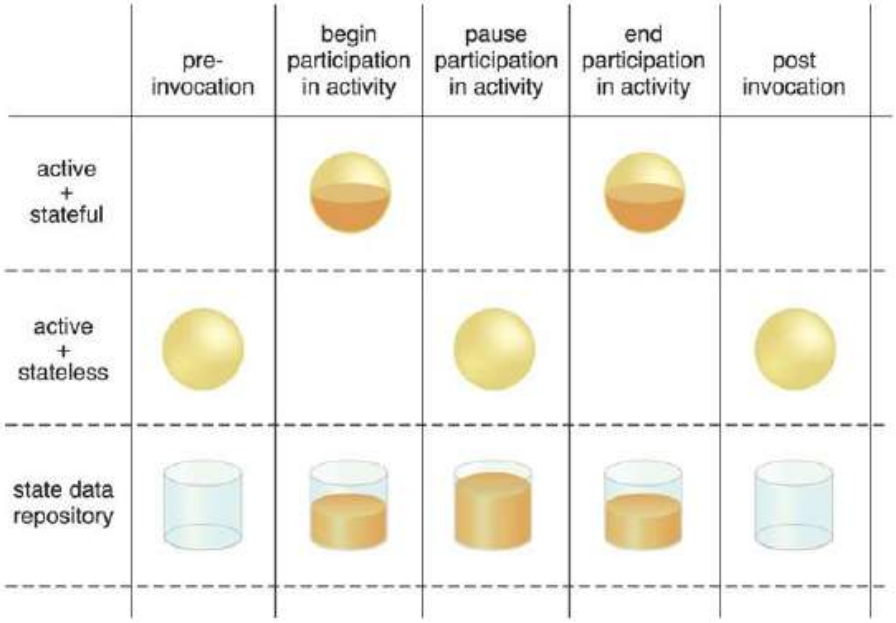


Figure 8.38. By deferring state data to a state repository, the cloud service is able to transition to a stateless condition (or a partially stateless condition), thereby temporarily freeing system resources.

Cloud Management Mechanisms

Cloud-based IT resources need to be set up, configured, maintained, and monitored. The systems covered in this chapter are mechanisms that encompass and enable these types of management tasks. They form key parts of cloud technology architectures by facilitating the control and evolution of the IT resources that form cloud platforms and solutions.

The following management-related mechanisms are described in this chapter:

- Remote Administration System
- Resource Management System
- SLA Management System
- Billing Management System

Remote Administration System

The ***remote administration system*** mechanism provides tools and user-interfaces for external cloud resource administrators to configure and administer cloud-based IT resources.
A remote administration system can establish a portal for access to administration and management features of various underlying systems, including the resource management, SLA management, and billing management systems described in this chapter (Figure 9.2)

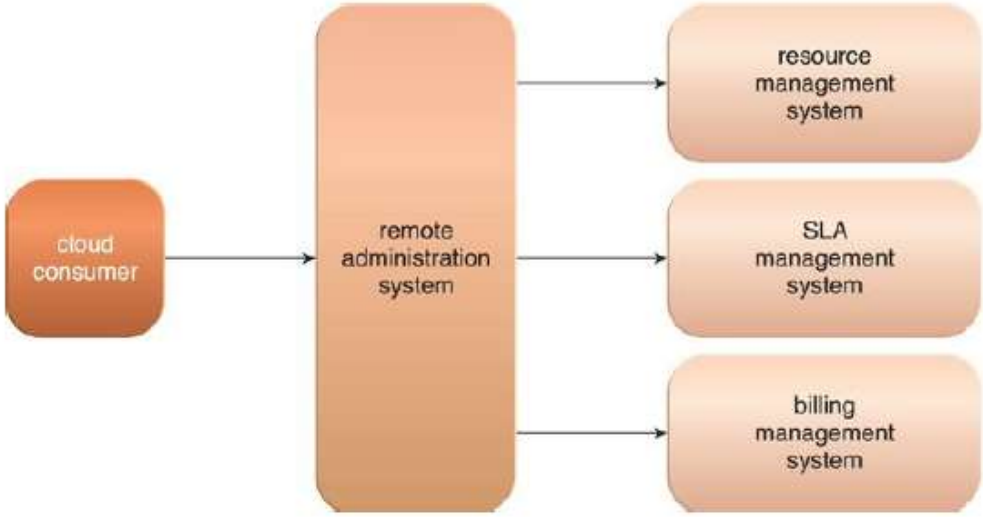


Figure 9.2. The remote administration system abstracts underlying management systems to expose and centralize administration controls to external cloud resource administrators. The system provides a customizable user console, while programmatically interfacing with underlying management systems via their APIs.

The tools and APIs provided by a remote administration system are generally used by the cloud provider to develop and customize online portals that provide cloud consumers with a variety of administrative controls.
The following are the two primary types of portals that are created with the remote administration system:
• ***Usage and Administration Portal*** - A general purpose portal that centralizes management controls to different cloud-based IT resources and can further provide IT resource usage reports.

• ***Self-Service Portal*** - This is essentially a shopping portal that allows cloud consumers to search an up-to-date list of cloud services and IT resources that are available from a cloud provider (usually for lease). The cloud consumer submits its chosen items to the cloud provider for provisioning.

Figure 9.3 illustrates a scenario involving a remote administration system and both usage and administration and self-service portals.

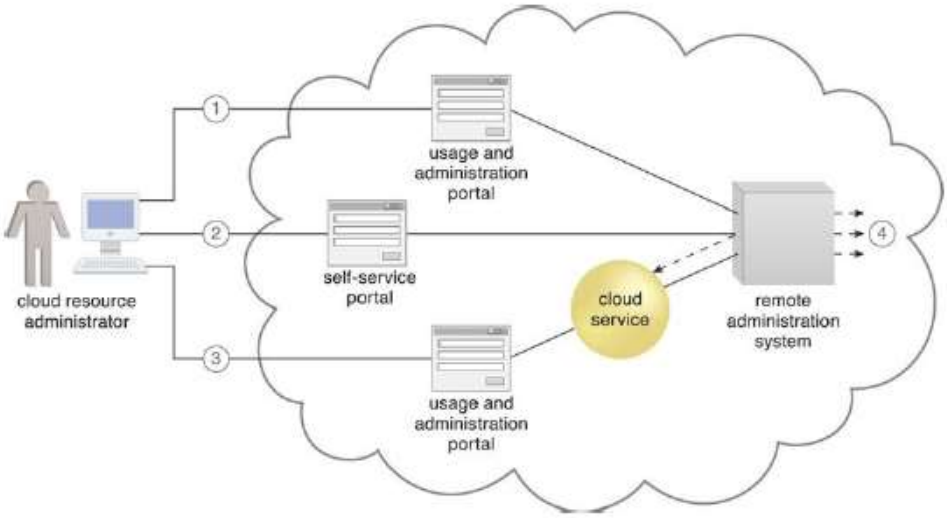


Figure 9.3. A cloud resource administrator uses the usage and administration portal to configure an already leased virtual server (not shown) to prepare it for hosting (1). The cloud resource administrator then uses the self-service portal to select and request the provisioning of a new cloud service (2). The cloud resource administrator then accesses the usage and administration portal again to configure the newly provisioned cloud service that is hosted on the virtual server (3). Throughout these steps, the remote administration system interacts with the necessary management systems to perform the requested actions (4).

Depending on:

- the type of cloud product or cloud delivery model the cloud consumer is leasing or using from the cloud provider,
- the level of access control granted by the cloud provider to the cloud consumer, and
- further depending on which underlying management systems the remote administration system interfaces with, ...tasks that can commonly be performed by cloud consumers via a remote administration console include:
- configuring and setting up cloud services
- provisioning and releasing IT resource for on-demand cloud services
- monitoring cloud service status, usage, and performance
- monitoring QoS and SLA fulfillment
- managing leasing costs and usage fees
- managing user accounts, security credentials, authorization, and access control
- tracking internal and external access to leased services
- planning and assessing IT resource provisioning
- capacity planning

While the user-interface provided by the remote administration system will tend to be proprietary to the cloud provider, there is a preference among cloud consumers to work with remote administration systems that offer standardized APIs. This allows a cloud consumer to invest in the creation of its own front-end with the fore-knowledge that it can reuse this console if it decides to move to another cloud provider that supports the same standardized API.

Resource Management System

The ***resource management system*** mechanism helps coordinate IT resources in response to management actions performed by both cloud consumers and cloud providers (Figure 9.5). Core to this system is the virtual infrastructure manager (VIM) that coordinates the server hardware so that virtual server instances can be created from the most expedient underlying physical server. A VIM is a commercial product that can be used to manage a range of virtual IT resources across multiple physical servers. For example, a VIM can create and manage multiple instances of a hypervisor across different physical servers or allocate a virtual server on one physical server to another (or to a resource pool)

Figure 9.5. A resource management system encompassing a VIM platform and a virtual machine image repository. The VIM may have additional repositories, including one dedicated to storing operational data.

Tasks that are typically automated and implemented through the resource management system include:

- managing virtual IT resource templates that are used to create pre-built instances, such as virtual server images
- allocating and releasing virtual IT resources into the available physical infrastructure in response to the starting, pausing, resuming, and termination of virtual IT resource instances
- coordinating IT resources in relation to the involvement of other mechanisms, such as resource replication, load balancer, and failover system
- enforcing usage and security policies throughout the lifecycle of cloud service instances
- monitoring operational conditions of IT resources

Resource management system functions can be accessed by cloud resource administrators employed by the cloud provider or cloud consumer. Those working on behalf of a cloud provider will often be able to directly access the resource management system's native console. Resource management systems typically expose APIs that allow cloud providers to build remote administration system portals that can be customized to selectively offer resource management controls to external cloud resource administrators acting on behalf of cloud consumer organizations via usage and administration portals.
Both forms of access are depicted in Figure 9.6.

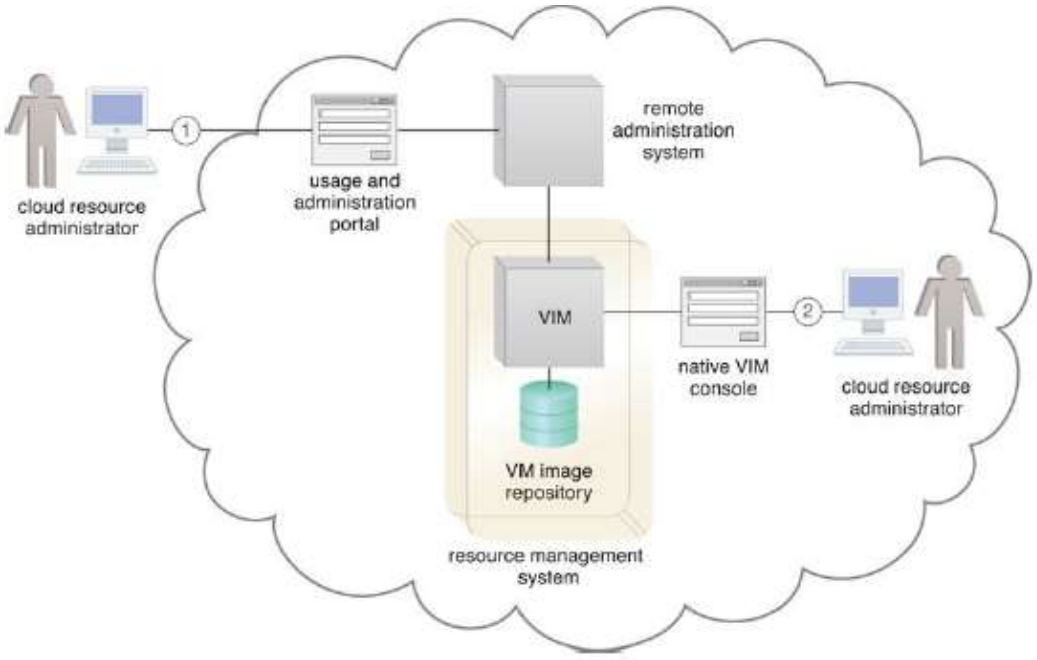


Figure 9.6. The cloud consumer's cloud resource administrator accesses a usage and administration portal externally to administer a leased IT resource (1). The cloud provider's cloud resource administrator uses the native user-interface provided by the VIM to perform internal resource management tasks (2).

SLA management system

The SLA management system mechanism represents a range of commercially available cloud management products that provide features pertaining to the administration, collection, storage, reporting, and runtime notification of SLA data (Figure 9.7)

Figure 9.7. An SLA management system encompassing an SLA manager and QoS measurements repository.

An SLA management system deployment will generally include a repository used to store and retrieve collected SLA data based on pre-defined metrics and reporting parameters. It will further rely on one or more SLA monitor mechanisms to collect the SLA data that can then be made available in near-real time to usage and administration portals to provide on-going feedback regarding active cloud services (Figure 9.8). The metrics monitored for individual cloud services are aligned with the SLA guarantees in corresponding cloud provisioning contracts.

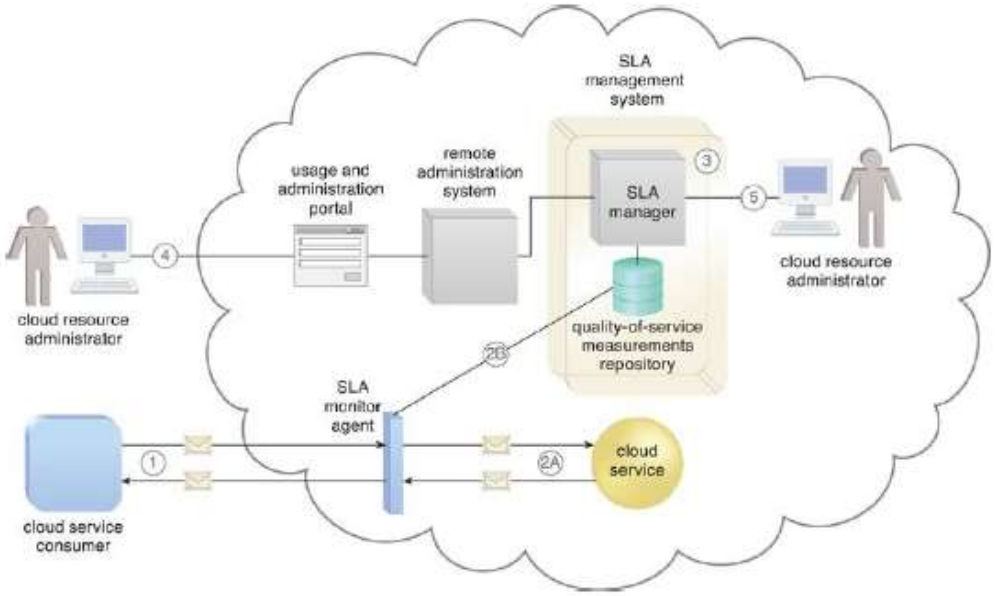


Figure 9.8. A cloud service consumer interacts with a cloud service (1). An SLA monitor intercepts the exchanged messages, evaluates the interaction, and collects relevant runtime data in relation to quality-of-service guarantees defined in the cloud service's SLA (2A). The data collected is stored in a repository (2B) that is part of the SLA management system (3). Queries can be issued and reports can be generated for an external cloud resource administrator via a usage and administration portal (4) or for an internal cloud resource administrator via the SLA management system's native user-interface (5).

Billing Management System

The ***billing management system*** mechanism is dedicated to the collection and processing of usage data as it pertains to cloud provider accounting and cloud consumer billing. Specifically, the billing management system relies on pay-per- use monitors to gather runtime usage data that is stored in a repository that the system components then draw from for billing, reporting, and invoicing purposes (Figures 9.9 and 9.10)

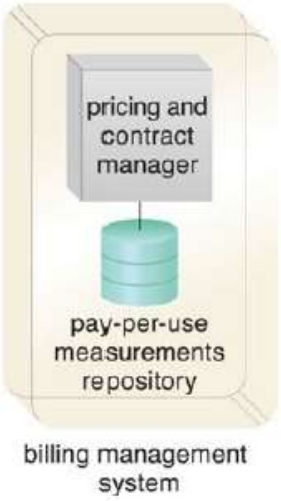


Figure 9.9. A billing management system comprised of a pricing and contract manager and a pay-per-use measurements repository.

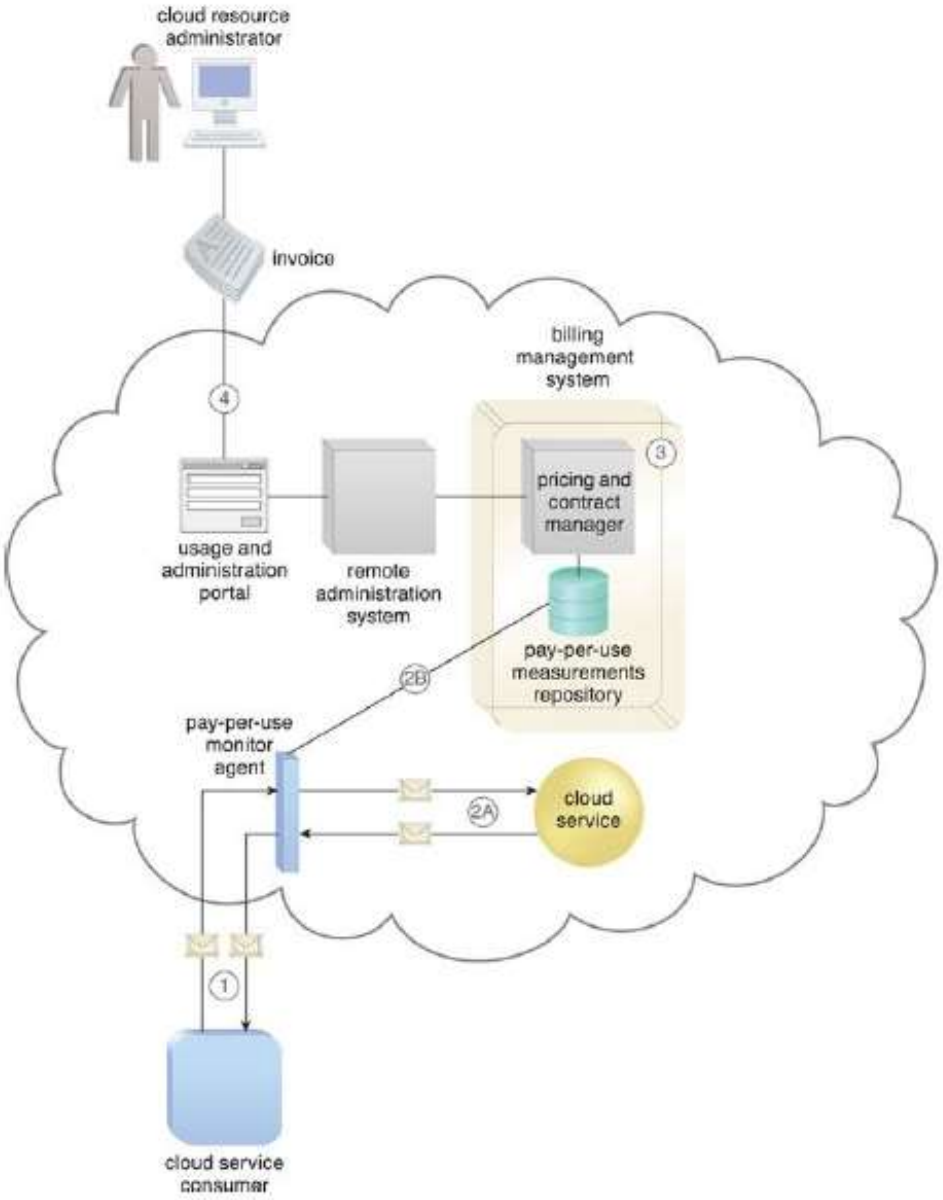


Figure 9.10. A cloud service consumer exchanges messages with a cloud service (1). A pay-per-use monitor keeps track of the usage and collects data relevant to

Cloud Security Mechanisms

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 technology commonly relies on a standardized algorithm called a ***cipher*** to transform original plaintext data into encrypted data, referred to as ***ciphertext.*** Access to ciphertext does not divulge the original plaintext data, apart from some forms of metadata, such as message length and creation date.

When encryption is applied to plaintext data, the data is paired with a string of characters called an ***encryption key,*** a secret message that is established by and shared among authorized parties. The encryption key is used to decrypt the ciphertext back into its original plaintext format.

The encryption mechanism can help counter the traffic eavesdropping, malicious intermediary, insufficient authorization, and overlapping trust boundaries security threats. For example, malicious service agents that attempt traffic eavesdropping are unable to decrypt messages in transit if they do not have the encryption key (Figure 10.1).

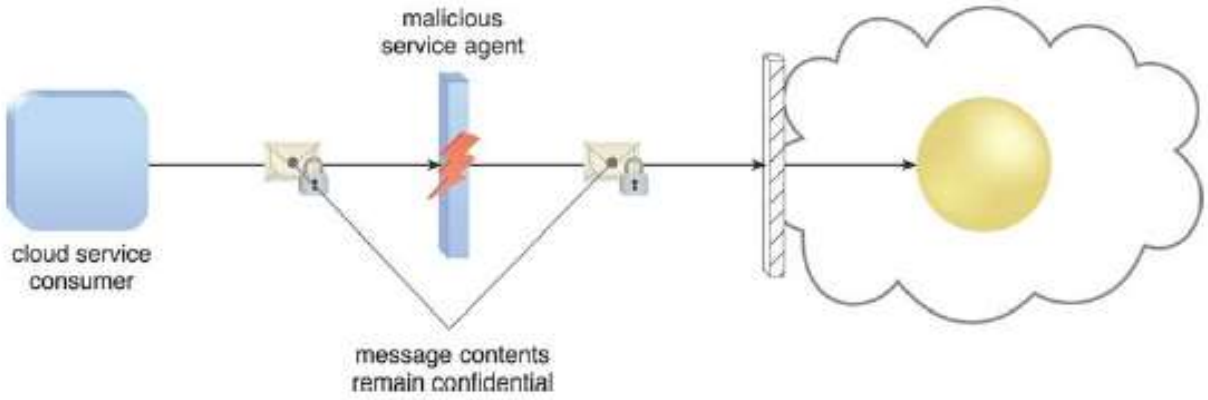


Figure 10.1. A malicious service agent is unable to retrieve data from an encrypted message. The retrieval attempt may furthermore be revealed to the cloud service consumer. (Note the use of the lock symbol to indicate that a

There are two common forms of encryption known as ***symmetric encryption*** and ***asymmetric encryption.***

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. Also known as ***secret key cryptography,*** messages that are encrypted with a specific key can be decrypted by only that same key. Parties that rightfully decrypt the data are provided with evidence that the original encryption was performed by parties that rightfully possess the key. A basic authentication check is always performed, because only authorized parties that own the key can create messages. This maintains and verifies data confidentiality.

Note that symmetrical encryption does not have the characteristic of nonrepudiation, since determining exactly which party performed the message encryption or decryption is not possible if more than one party is in possession of the key.

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. Conversely, a document that was encrypted with a public key can be decrypted only using its private key counterpart. As a result of two different keys being used instead of just the one, asymmetric encryption is almost always computationally slower than symmetric encryption.

The level of security that is achieved is dictated by whether a private key or public key was used to encrypt the plaintext data. As every asymmetrically encrypted message has its own private-public key pair, messages that were encrypted with a private key can be correctly decrypted by any party with the corresponding public key. This method of encryption does not offer any confidentiality protection, even though successful decryption proves that the text was encrypted by the rightful public key owner. Private key encryption therefore offers integrity protection in addition to authenticity and non-repudiation. A message that was encrypted with a public key can only be decrypted by the rightful private key owner, which provides confidentiality protection.

Hashing

The ***hashing*** mechanism is used when a one-way, non-reversible 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 of this mechanism 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.

In addition to its utilization for protecting stored data, the cloud threats that can be mitigated by the hashing mechanism include malicious intermediary and insufficient authorization. An example of the latter is illustrated in Figure 10.3

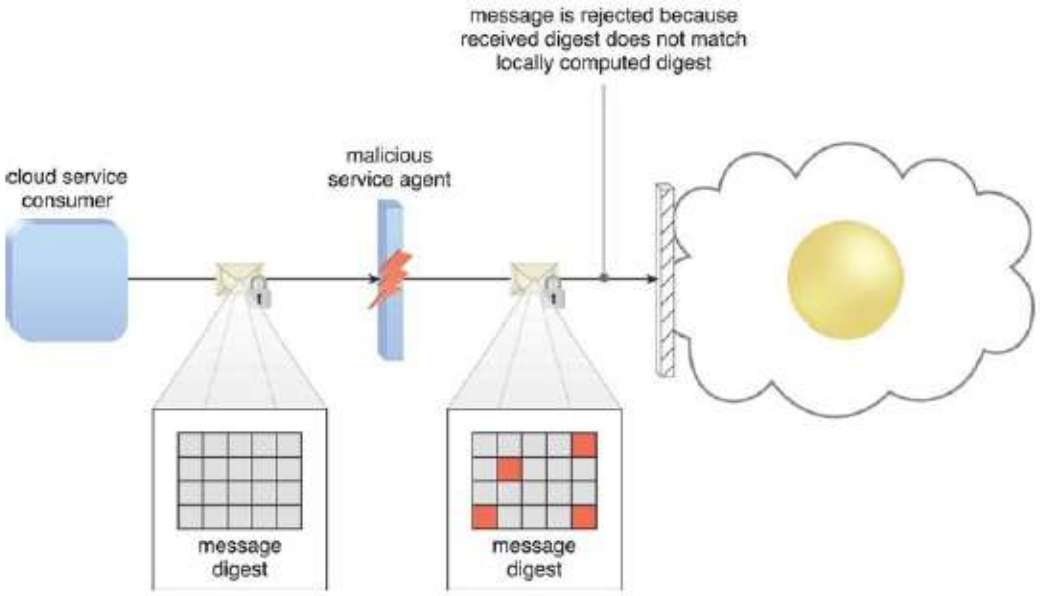


Figure 10.3. A hashing function is applied to protect the integrity of a message that is intercepted and altered by a malicious service agent, before it is forwarded. The firewall can be configured to determine that the message has been altered, thereby enabling it to reject the message before it can 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.

Both hashing and asymmetrical encryption are involved in the creation of a digital signature, which essentially exists as a message digest that was encrypted by a private key and appended to the original message. The recipient verifies the signature validity and uses the corresponding public key to decrypt the digital signature, which produces the message digest. The hashing mechanism can also be applied to the original message to produce this message digest. Identical results from the two different processes indicate that the message maintained its integrity.

The digital signature mechanism helps mitigate the malicious intermediary, insufficient authorization, and overlapping trust boundaries security threats (Figure 10.5).

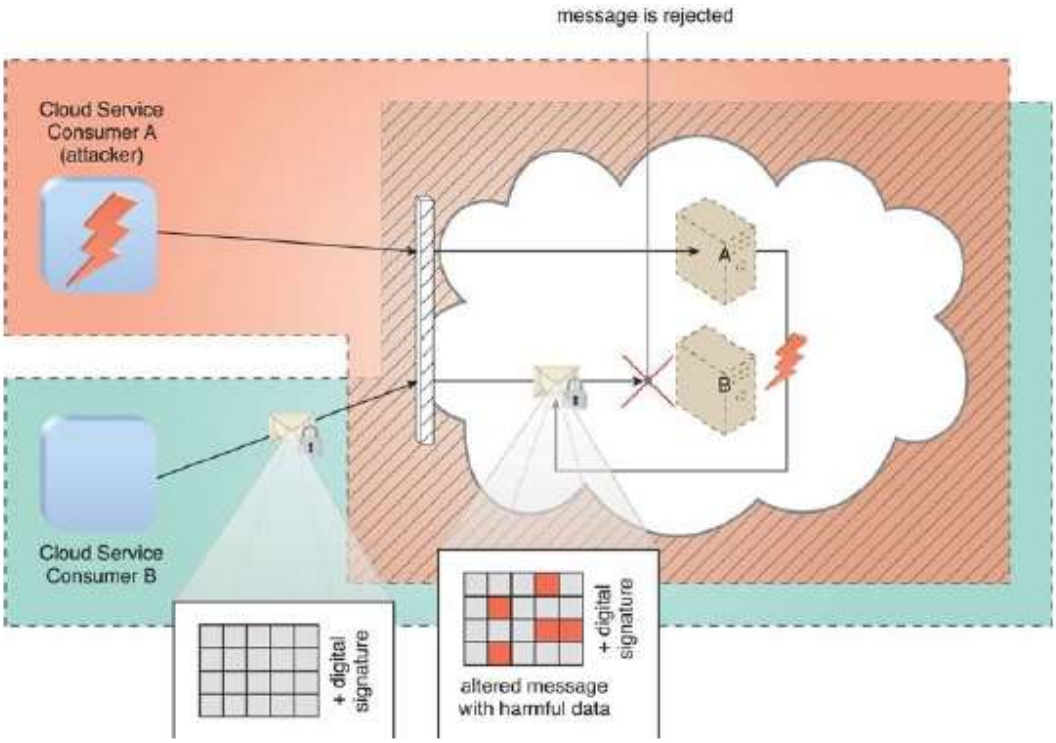


Figure 10.5. Cloud Service Consumer B sends a message that was digitally signed but was altered by trusted attacker Cloud Service Consumer A. Virtual Server B is configured to verify digital signatures before processing incoming messages even if they are within its trust boundary. The message is revealed 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 key cryptography. This system is used to associate public keys with their corresponding key owners (known as ***public key identification)*** while enabling the verification of key validity. PKIs rely on the use of digital certificates, which are digitally signed data structures that bind public keys to certificate owner identities, as well as to related information, such as validity periods. Digital certificates are usually digitally signed by a third-party certificate authority (CA), as illustrated in Figure 10.7

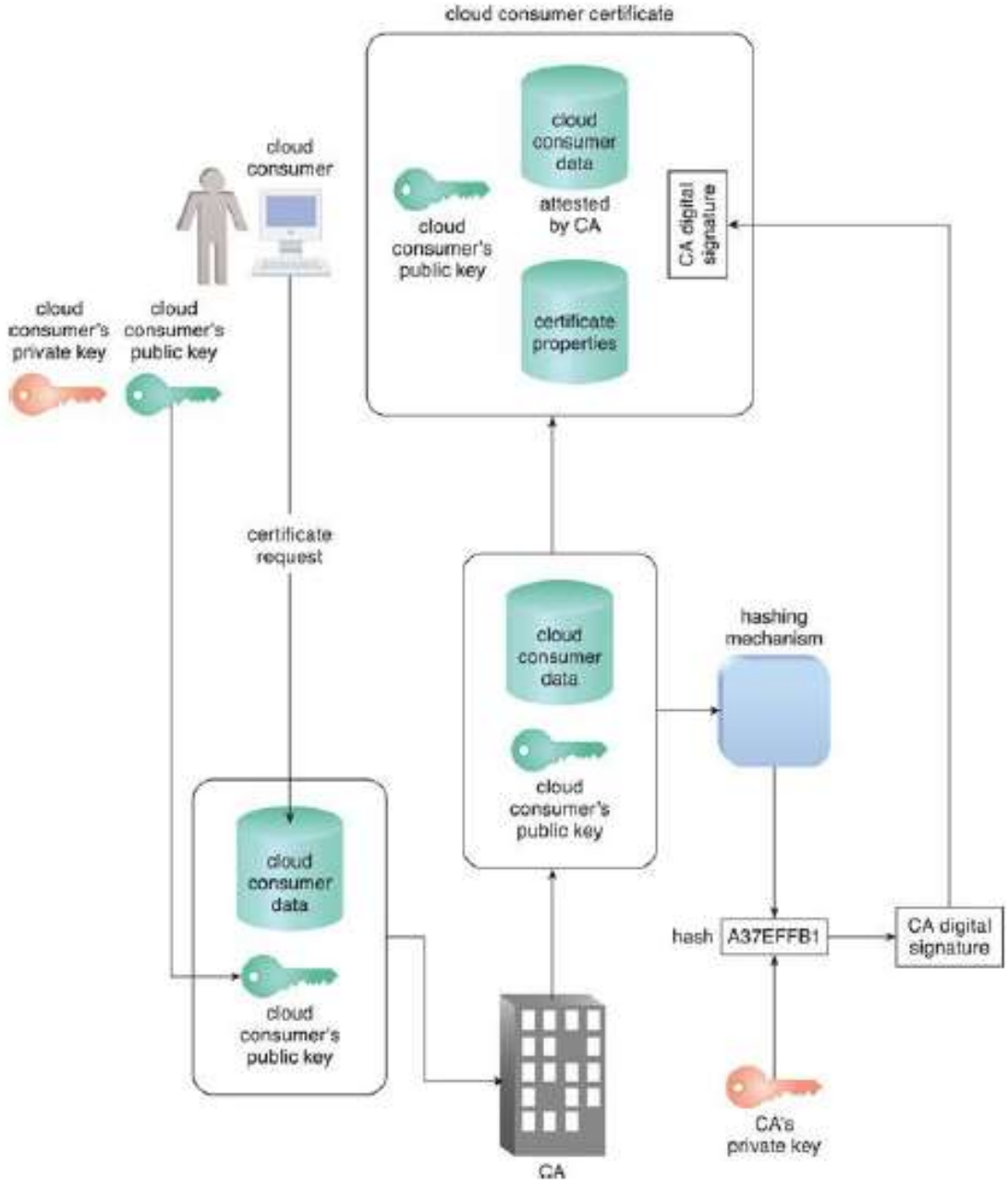


Figure 10.7. The common steps involved during the generation of certificates by a certificate authority.

Other methods of generating digital signatures can be employed, even though the majority of digital certificates are issued by only a handful of trusted CAs like VeriSign and Comodo. Larger organizations,

such as Microsoft, can act as their own CA and issue certificates to their clients and the public, since even individual users can generate certificates as long as they have the appropriate software tools.

Building up an acceptable level of trust for a CA is time-intensive but necessary. Rigorous security measures, substantial infrastructure investments, and stringent operational processes all contribute to establishing the credibility of a CA. The higher its level of trust and reliability, the more esteemed and reputable its certificates. The PKI is a dependable method for implementing asymmetric encryption, managing cloud consumer and cloud provider identity information, and helping to defend against the malicious intermediary and insufficient authorization threats.

The PKI mechanism is primarily used to counter the insufficient authorization threat.

Identity and Access Management (IAM)

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.

Specifically, IAM mechanisms exist as systems comprised of four main components:

- ***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), and locking user accounts to registered IP or MAC addresses.
- ***Authorization*** - The authorization component defines the correct granularity for access controls and oversees the relationships between identities, access control rights, and IT resource availability.
- ***User Management*** - Related to the administrative capabilities of the system, the user management program is responsible for creating new user identities and access groups, resetting passwords, defining password policies, and managing privileges.
- ***Credential Management*** - The credential management system establishes identities and access control rules for defined user accounts, which mitigates the threat of insufficient authorization.

Although its objectives are similar to those of the PKI mechanism, the IAM mechanism's scope of implementation is distinct because its structure encompasses access controls and policies in addition to assigning specific levels of user privileges.

The IAM mechanism is primarily used to counter the insufficient authorization, denial of service, and overlapping trust boundaries threats.

Single Sign-On (SSO)

Propagating the authentication and authorization information for a cloud service consumer across multiple cloud services can be a challenge, especially if numerous cloud services or cloud-based IT resources need to be invoked as part of the same overall runtime activity. The ***single sign-on (SSO)*** mechanism enables one cloud service consumer to be authenticated by a security broker, which establishes a security context that is persisted while the cloud service consumer accesses other cloud services or cloud-based IT resources. Otherwise, the cloud service consumer would need to re-authenticate itself with every subsequent request.

The SSO mechanism essentially enables mutually independent cloud services and IT resources to generate and circulate runtime authentication and authorization credentials. The credentials initially provided by the cloud service consumer remain valid for the duration of a session, while its security context information is shared (Figure 10.9). The SSO mechanism's security broker is especially useful when a cloud service consumer needs to access cloud services residing on different clouds (Figure 10.10)

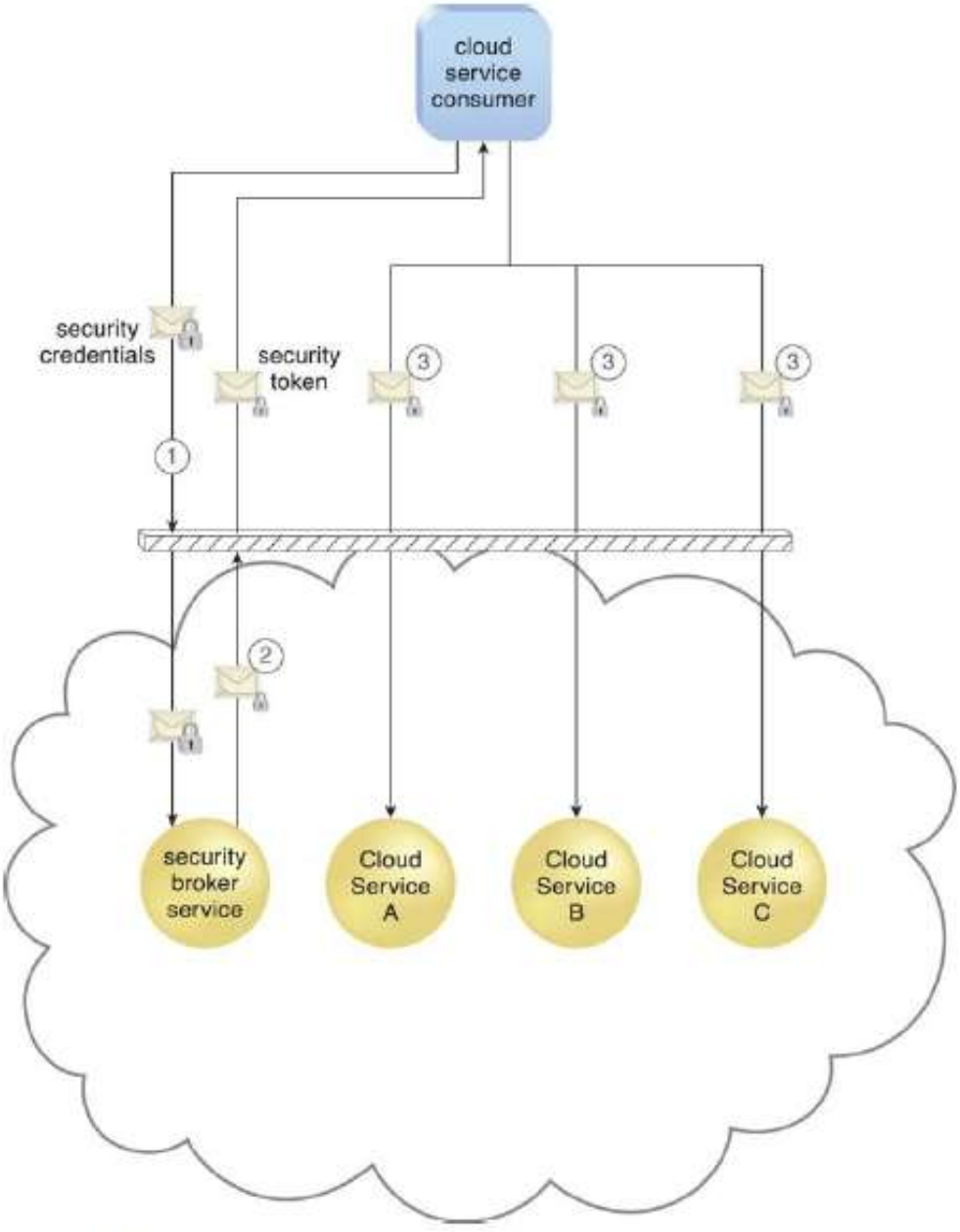


Figure 10.9. A cloud service consumer provides the security broker with login credentials (1). The security broker responds with an authentication token (message with small lock symbol) upon successful authentication, which contains cloud service consumer identity information (2) that is used to automatically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

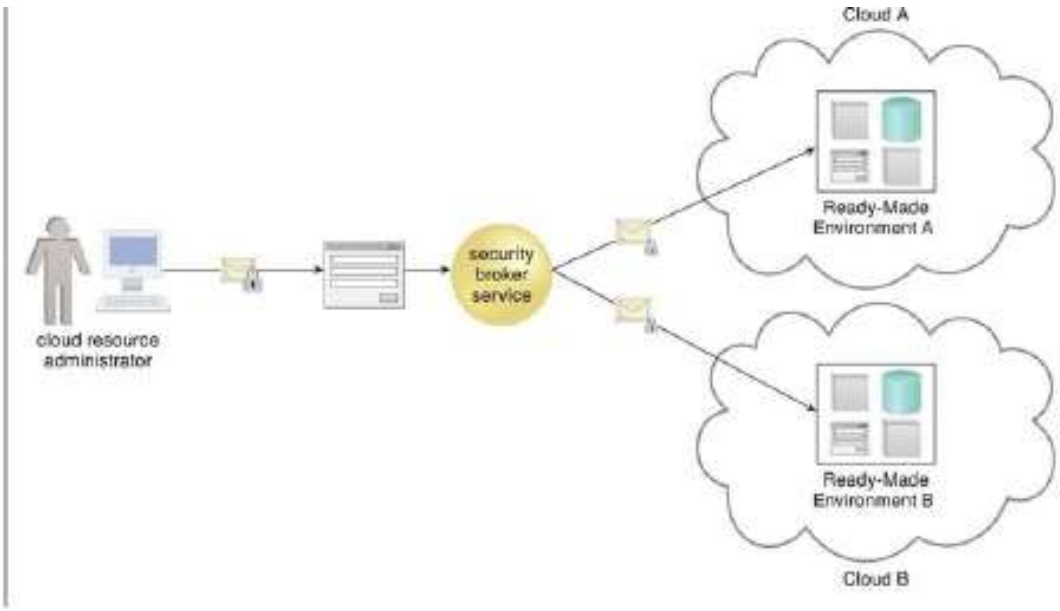


Figure 10.10. The credentials received by the security broker are propagated to ready-made environments across two different clouds. The security broker is responsible for selecting the appropriate security procedure with which to contact each cloud.

Cloud-Based Security Groups

Similar to constructing dykes and levees that separate land from water, data protection is increased by placing barriers between IT resources. Cloud resource segmentation is a process by which separate physical and virtual IT environments are created for different users and groups. For example, an organization's WAN can be partitioned according to individual network security requirements. One network can be established with a resilient firewall for external Internet access, while a second is deployed without a firewall because its users are internal and unable to access the Internet.

Resource segmentation is used to enable virtualization by allocating a variety of physical IT resources to virtual machines. It needs to be optimized for public cloud environments, since organizational trust boundaries from different cloud consumers overlap when sharing the same underlying physical IT resources.

The cloud-based resource segmentation process creates ***cloud-based security group*** mechanisms that are determined through security policies. Networks are segmented into logical cloud-based security groups that form logical network perimeters. Each cloud-based IT resource is assigned to at least one logical cloud-based security group. Each logical cloud-based security group is assigned specific rules that govern the communication between the security groups.

Multiple virtual servers running on the same physical server can become members of different logical cloud-based security groups (Figure 10.11). Virtual servers can further be separated into public-private groups, development- production groups, or any other designation configured by the cloud resource administrator.

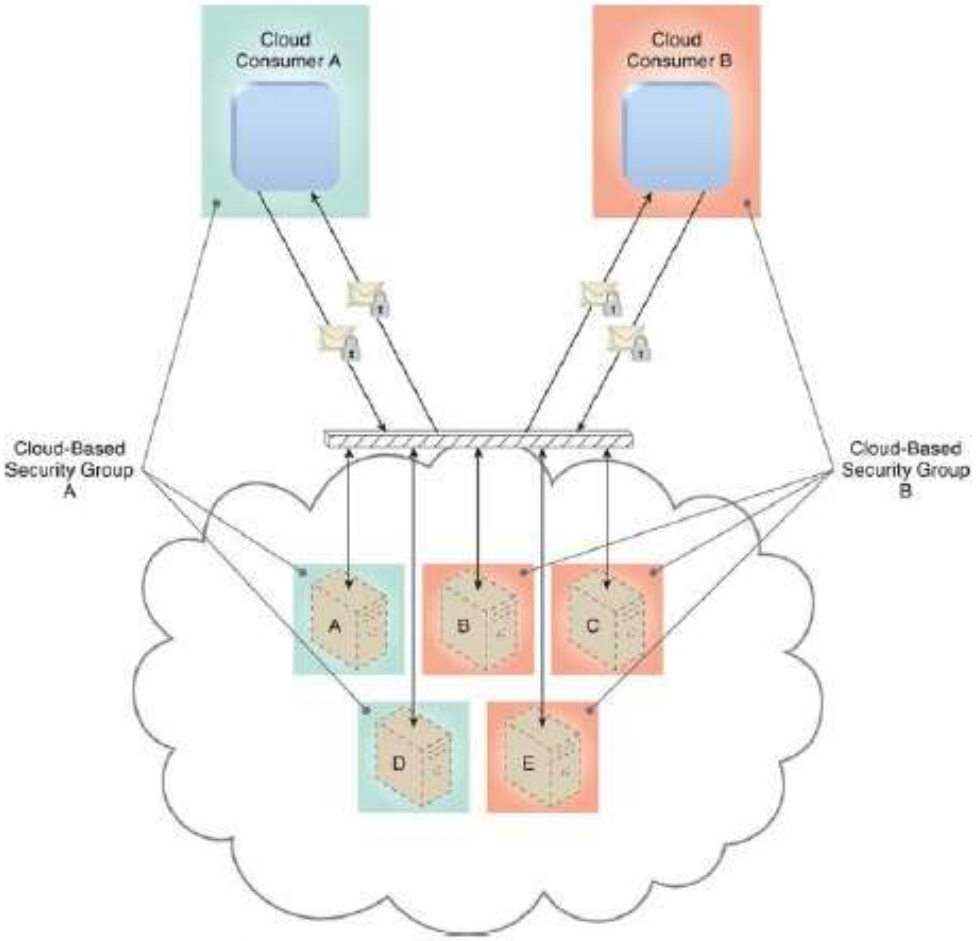


Figure 10.11. Cloud-Based Security Group A encompasses Virtual Servers A and D and is assigned to Cloud Consumer A. Cloud-Based Security Group B is comprised of Virtual Servers B, C, and E and is assigned to Cloud Consumer B. If Cloud Service Consumer A's credentials are compromised, the attacker would only be able to access and damage the virtual servers in Cloud-Based Security Group A, thereby protecting Virtual Servers B, C, and E.

Cloud-based security groups delineate areas where different security measures can be applied. Properly implemented cloud-based security groups help limit unauthorized access to IT resources in the event of a security breach. This mechanism can be used to help counter the denial of service, insufficient authorization, and overlapping trust boundaries threats, and is closely related to the logical network perimeter mechanism

Hardened Virtual Server Images

As previously discussed, a virtual server is created from a template configuration called a virtual server image (or virtual machine image). Hardening is the process of stripping unnecessary software from a system to limit potential vulnerabilities that can be exploited by attackers. Removing redundant programs, closing unnecessary server ports, and disabling unused services, internal root accounts, and guest access are all examples of hardening.

A ***hardened virtual server image*** is a template for virtual service instance creation that has been subjected to a hardening process (Figure 10.13). This generally results in a virtual server template that is significantly more secure than the original standard image.

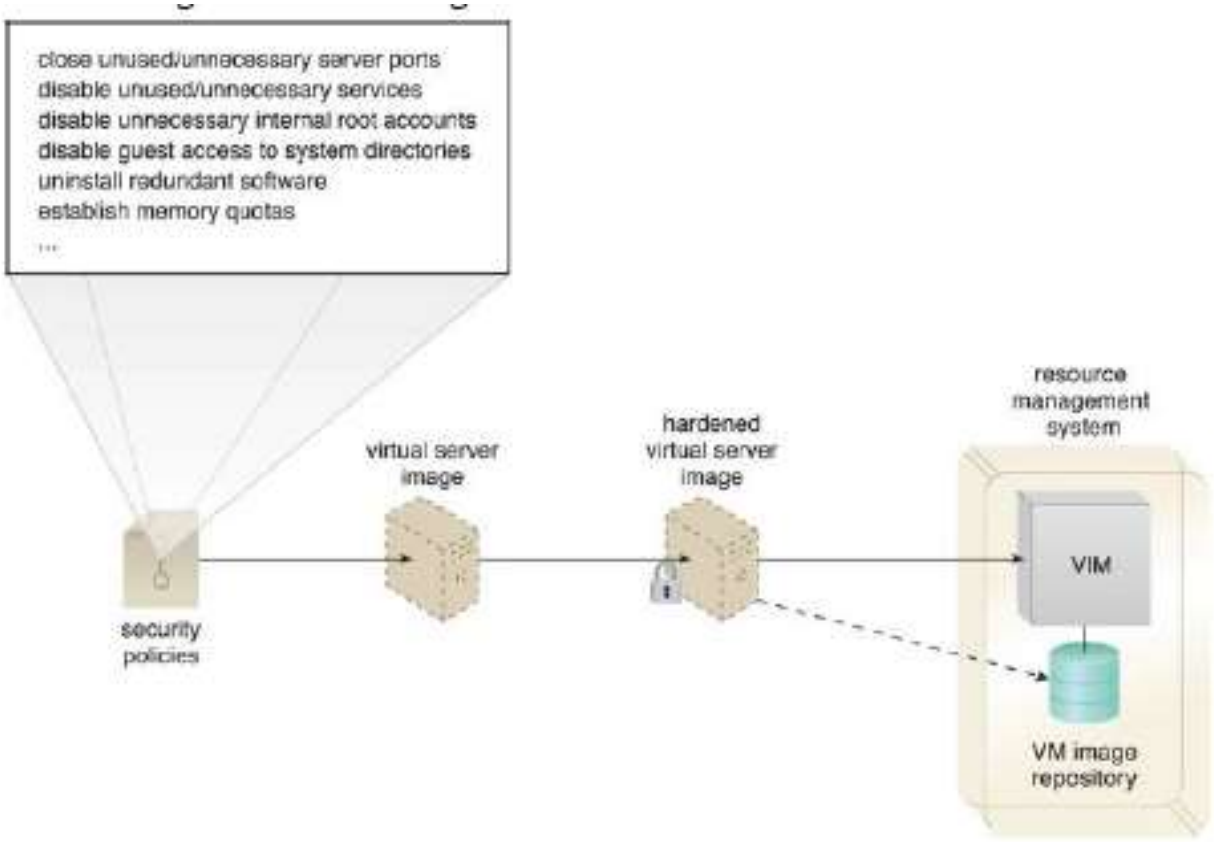


Figure 10.13. A cloud provider applies its security policies to harden its standard virtual server images. The hardened image template is saved in the VM images repository as part of a resource management system.

Hardened virtual server images help counter the denial of service, insufficient authorization, and overlapping trust boundaries threats.