UNIT-3
Cloud Mechanisms Overview

Technology mechanisms represent well-defined IT artifacts that are established within an IT industry and commonly distinct to a certain computing model or platform. The technology-centric nature of cloud computing requires the establishment of a formal level of mechanisms to be able to explore how a given pattern can be applied differently via alternative combinations of mechanism implementations. This not only standardizes proven practices and solutions in a design pattern format, it further adds standardization to pattern application options. It is for this reason that the mechanisms listed on this site have been defined for the cloud computing design patterns catalog.

The following mechanisms are defined:

- Automated Scaling Listener
- Automated Scaling Listener
- SLA Monitor
- Pay-Per-Use Monitor
- Audit Monitor
- Failover System,
- Hypervisor
- Resource Cluster
- Multi-device Broker
- State Management.

Automated Scaling Listener

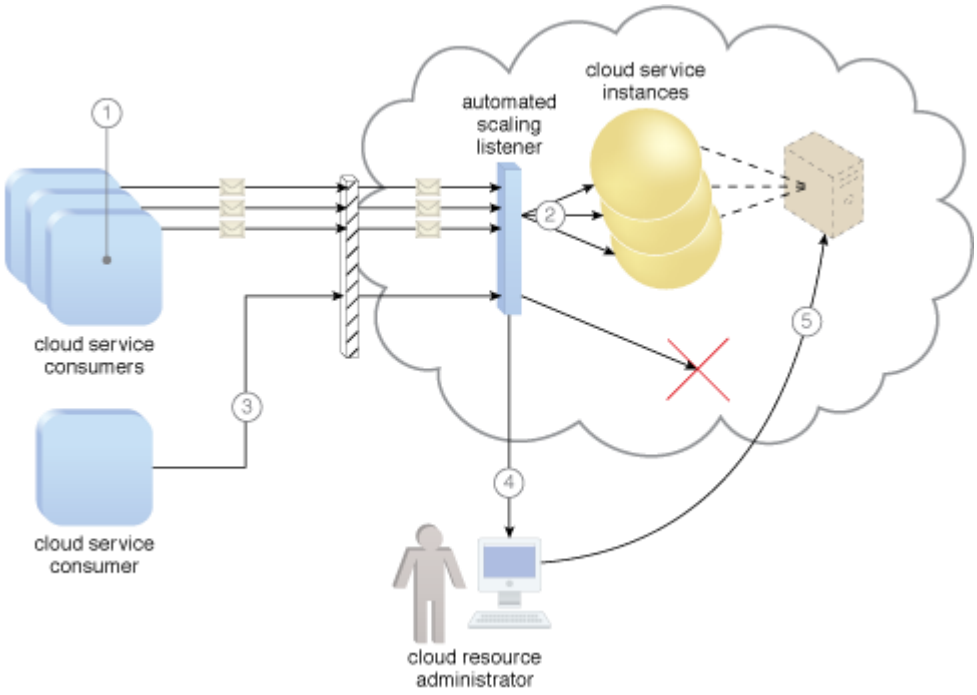
The automated scaling listener mechanism is a service agent that monitors and tracks communications between cloud service consumers and cloud services for dynamic scaling purposes. Automated scaling listeners are deployed within the cloud, typically near the firewall, from where they automatically track 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 auto-scaling).
- Automatic notification of the cloud consumer when workloads exceed current thresholds or fall below allocated resources. This way, the cloud consumer can choose to adjust its current IT resource allocation.

Different cloud provider vendors have different names for service agents that act as automated scaling listeners.



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

The load balancer mechanism is a runtime agent with logic fundamentally based on the premise of employing horizontal scaling to balance a workload across two or more IT resources to increase performance and capacity beyond what a single IT resource can provide. Beyond simple division of labor algorithms (Figure 1), 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

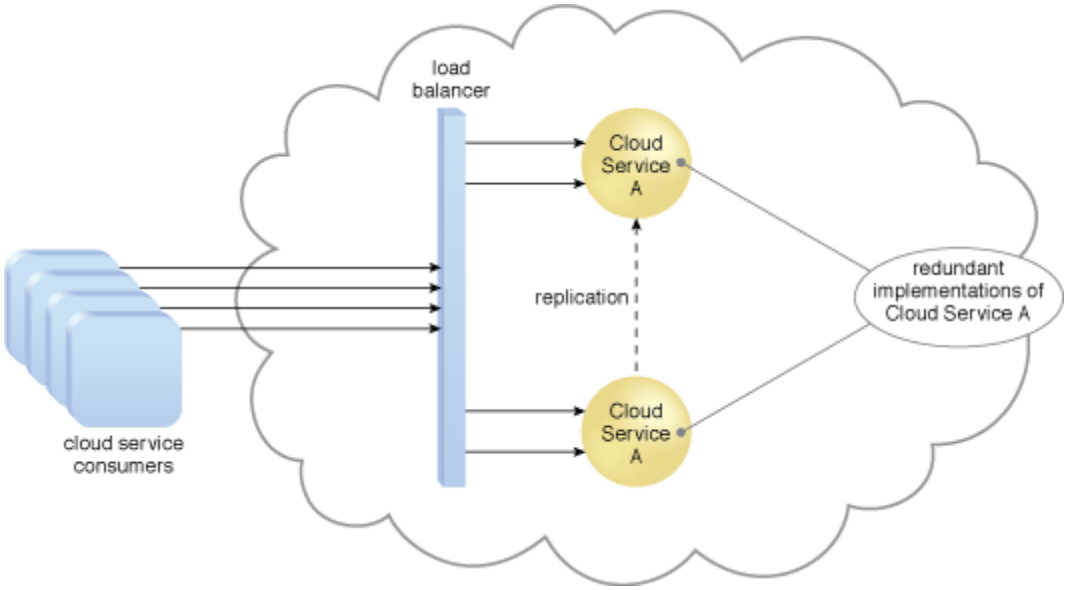


Figure 1 – A load balancer implemented as a service agent transparently distributes incoming workload request messages across two redundant cloud service implementations, which in turn maximizes performance for the clouds service consumers.

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)

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 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.

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-realtime to usage and administration portals to provide ongoing feedback regarding active cloud services (Figure 1). The metrics monitored for individual cloud services are aligned with the SLA guarantees in corresponding cloud provisioning contracts.

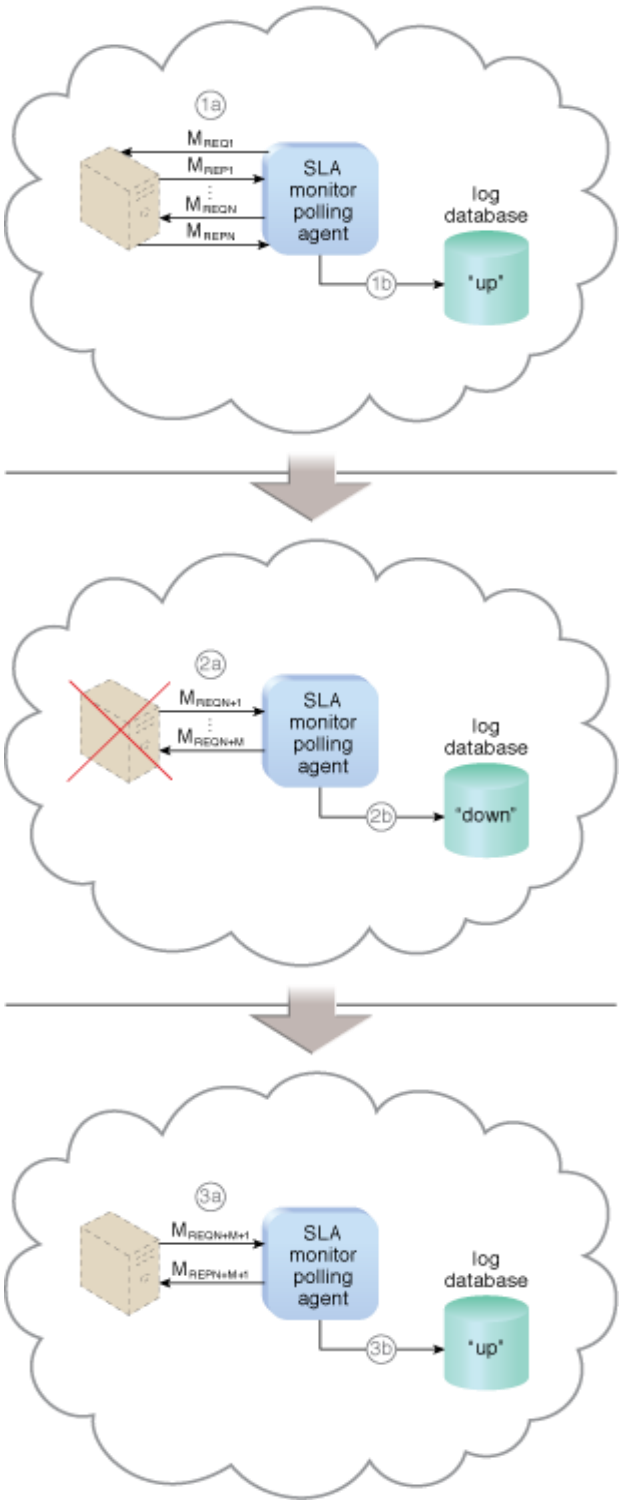


Figure 1 – A cloud service consumer interacts with a cloud service (1). An SLA monitor intercepts the exchanged messages, evaluates the interaction, and collects relevant runtime data in relation to quality-of-service guarantees defined in the cloud service's SLA (2A). The data collected is stored in 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).

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

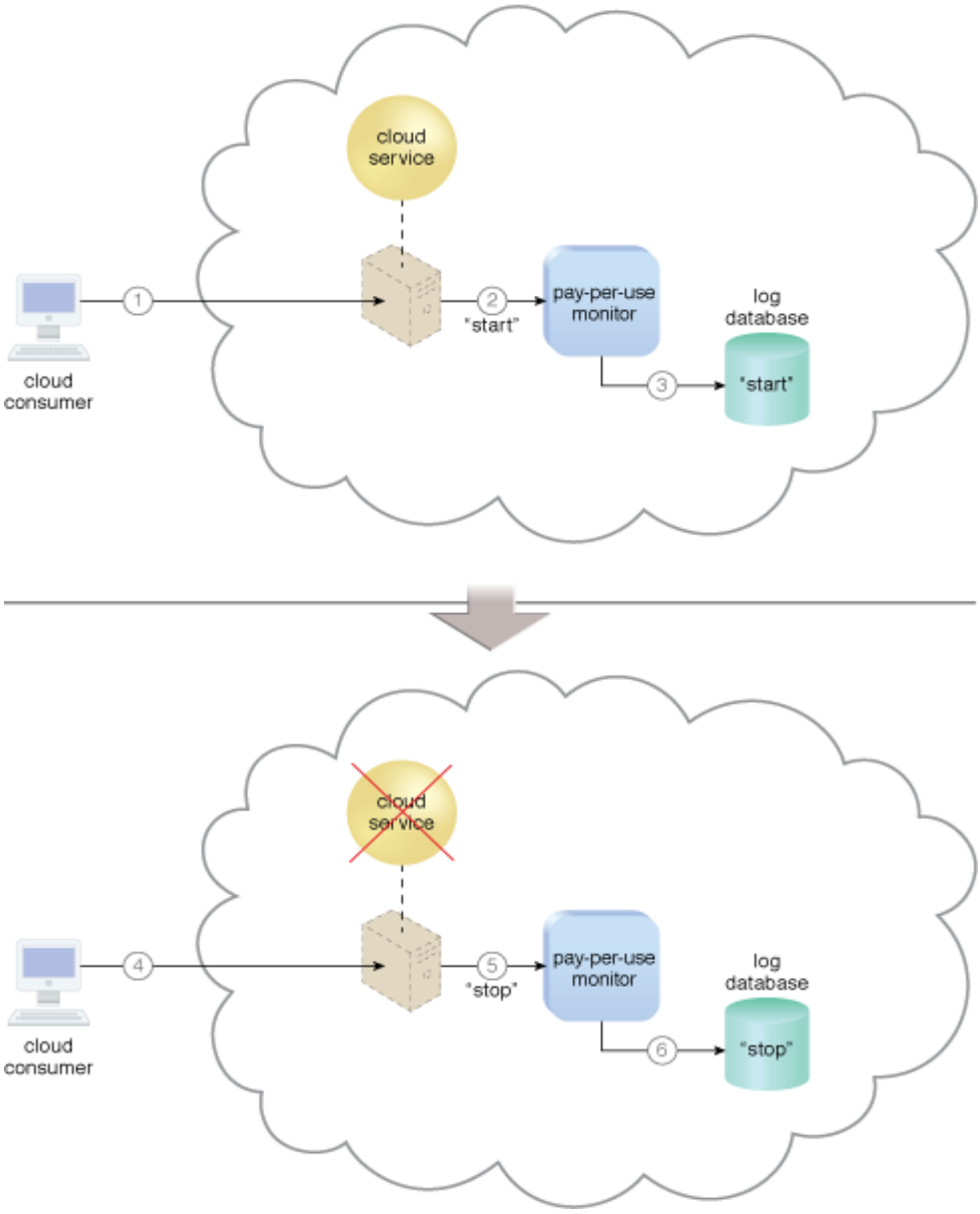


Figure 1 – A cloud consumer requests the creation of a new instance of a cloud service (1). The IT resource is instantiated and they pay-per-use monitor mechanism receives a "start" event notification from the resource software (2). The pay-per-use monitor stores the value timestamp in the log database (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).

Figure 1 illustrates the pay-per-use monitor designed as a monitoring agent that transparently intercepts and analyzes runtime communication with a cloud service.

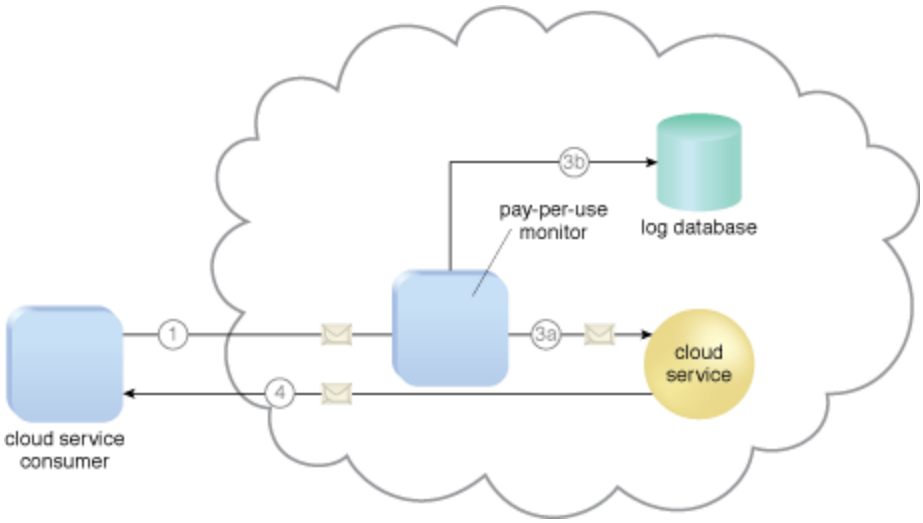
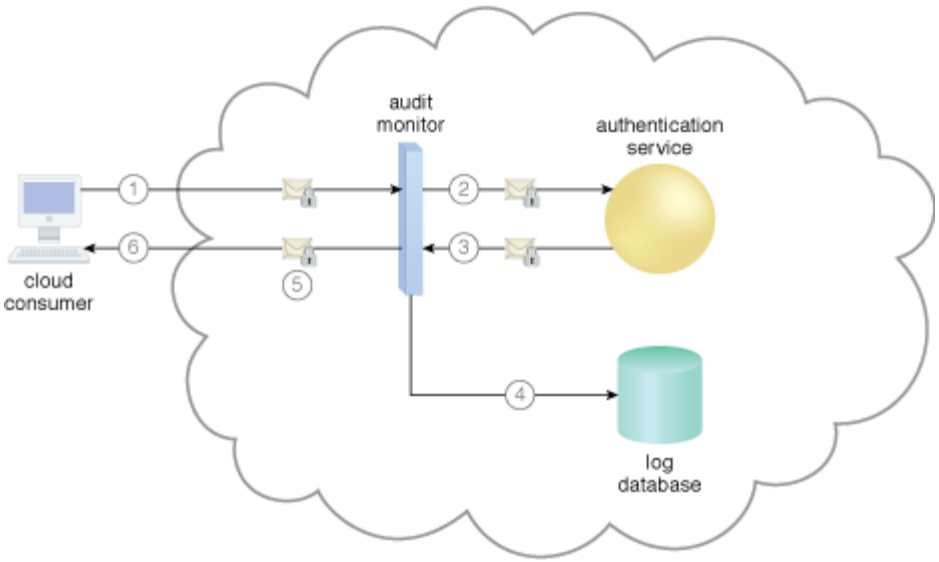


Figure 2 – A cloud service consumer sends a request message to the cloud service (1). The pay-per-use monitor intercepts the message (2), forwards it to the cloud service (3a), and stores the usage information in accordance with its monitoring metrics (3b). The cloud service forwards the response messages back to the cloud service consumer to provide the requested service (4).

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. The figure 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.



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 or for 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.

This mechanism may rely on the resource replication mechanism to supply the 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 1). Load balancing among active instances is required. When a failure is detected, the failed instance is removed from the load balancing scheduler (Figure 2). Whichever IT resource remains operational when a failure is detected takes over the processing (Figure 3).

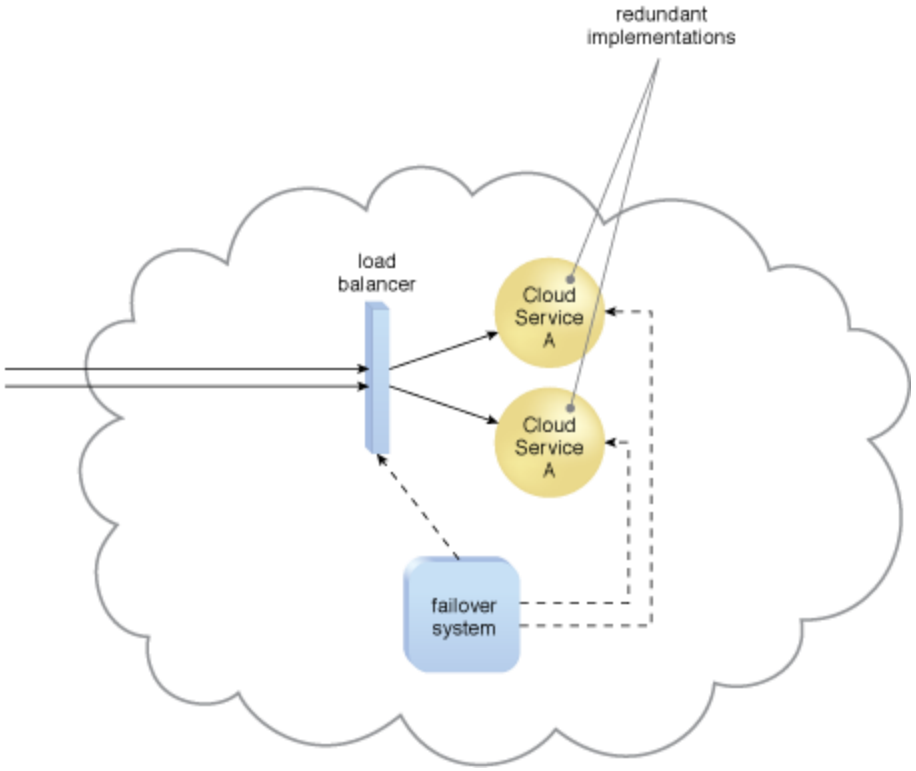


Figure 1 – The failover system monitors the operational status of Cloud Service A.

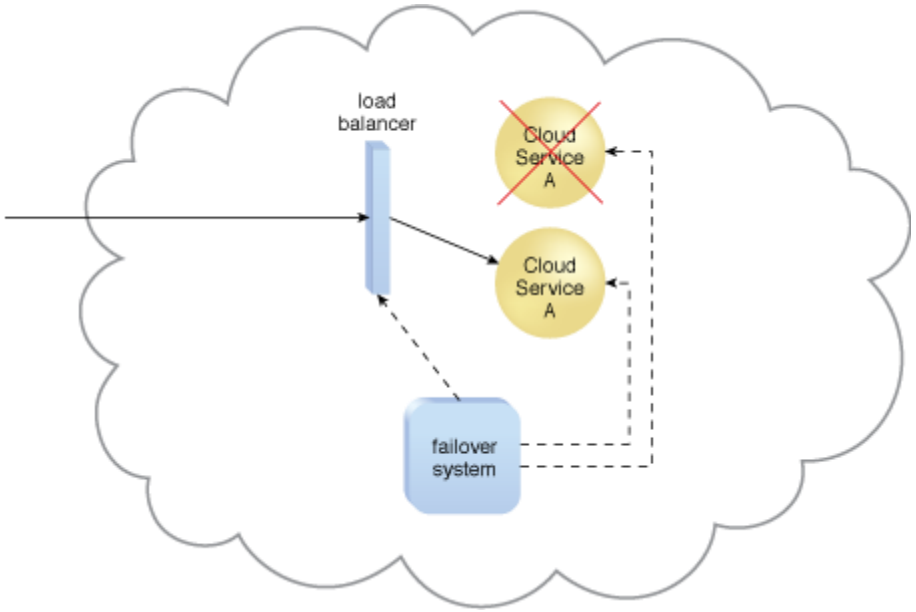


Figure 2 – When a failure is detected in one Cloud Service A implementation, the failover system commands the load balancer to switch over the workload to the redundant Cloud Service A implementation.

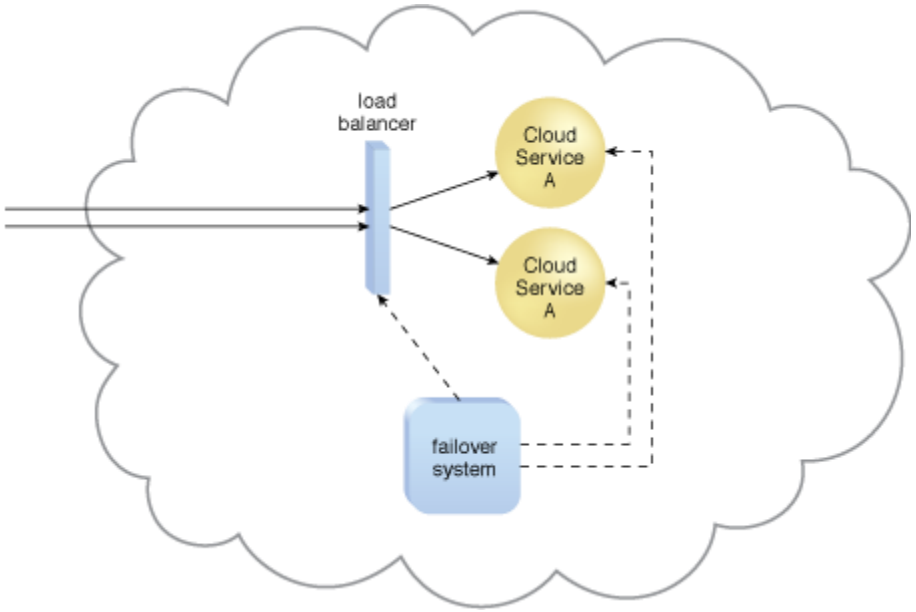


Figure 3 – The failed Cloud Service A implementation is recovered or replicated into an operational cloud service. The failover system now commands the load balancer to distribute the workload again.

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 4 to 5).

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 if its redundant implementations.

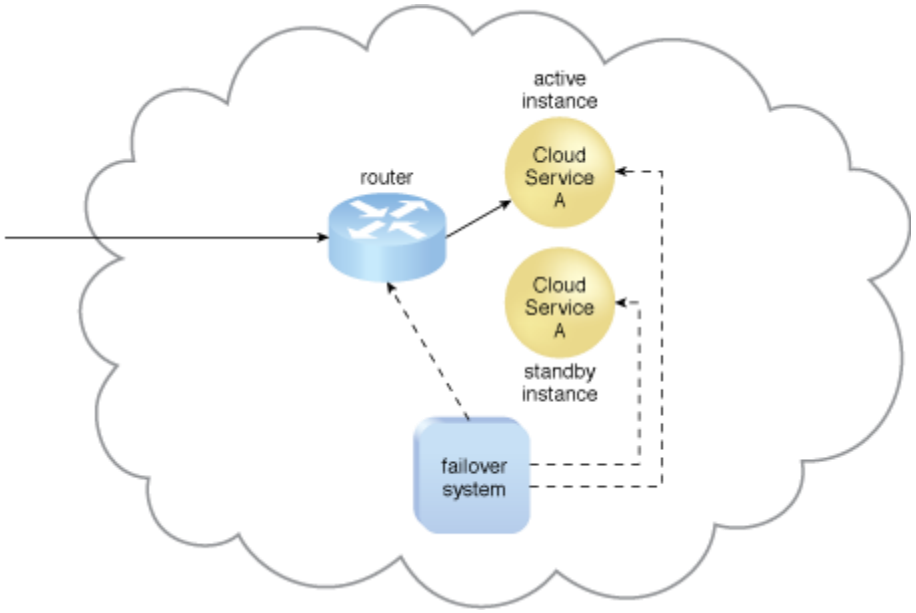


Figure 4 – The failover system monitors the operational status of Cloud Service A. The Cloud Service A implementation acting as the active instance is receiving cloud service consumer requests.

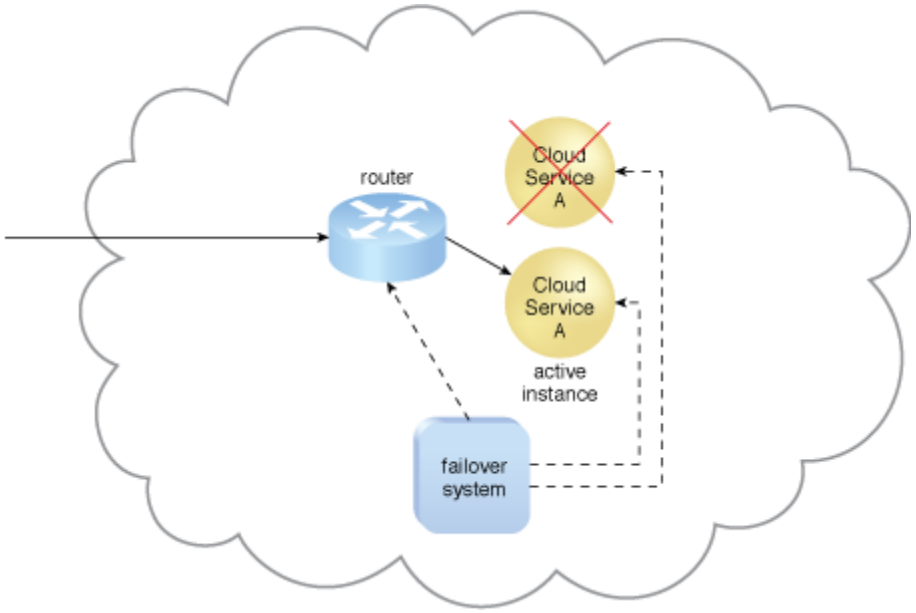


Figure 5 – The Cloud Service A implementation acting as the active instance encounters a failure that is detected by the failover system, which subsequently activates the inactive Cloud Service A implementation and redirects the workload toward it. The newly invoked Cloud Service A implementation now assumes the role of active instance.

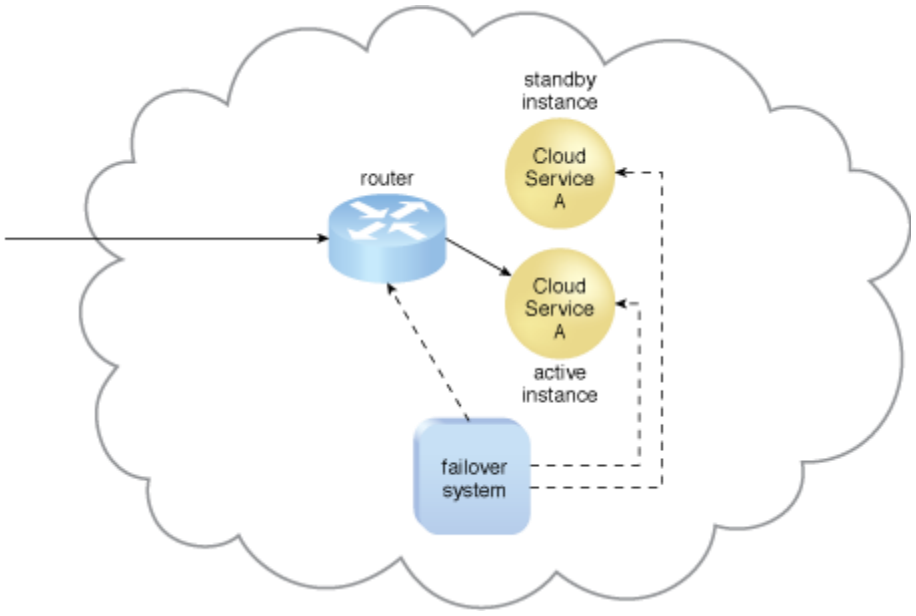


Figure 6 – The failed Cloud Service A implementation is recovered or replicated into an operational cloud service, and is now positioned as the standby instance, while the previously invoked Cloud Service A continues to serve as the active instance.

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 1). Similarly, a hypervisor can only assign the 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.

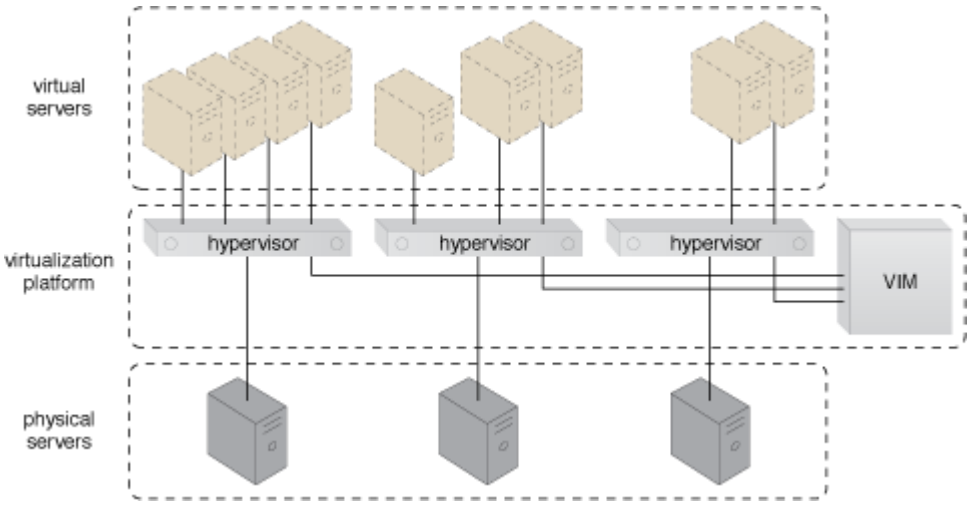


Figure 1 – Virtual servers are created via individual hypervisors on individual physical servers. All three hypervisors are jointly controlled by the same VIM.

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 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.

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 1). Some resource clusters are designed with more loosely coupled IT resources that only require the network layer (Figure 2).

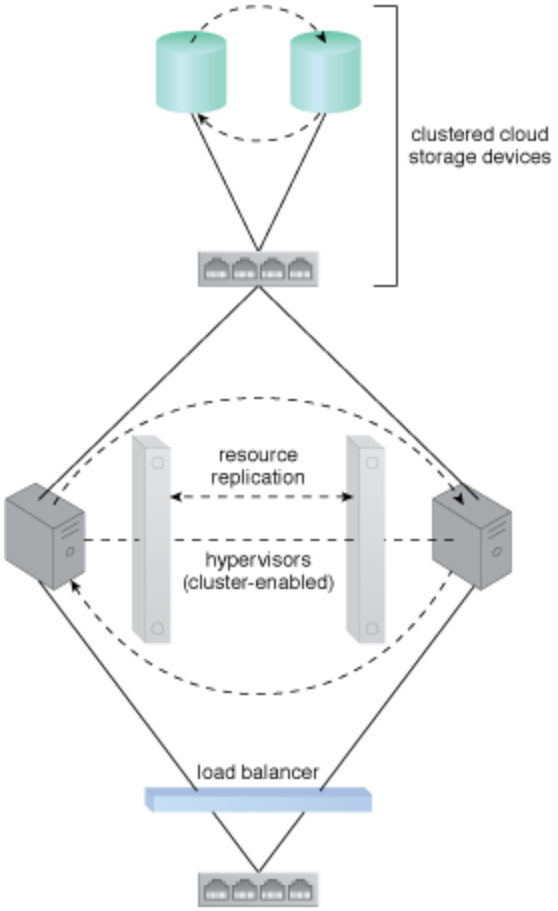


Figure 1 – Load balancing and resource replication are implemented through a cluster enabled hypervisor. A dedicated storage area network is used to connect the clustered storage and the clustered servers, which are able to share common cloud storage devices. This simplifies the storage replication process, which is independently carried out at the storage cluster.

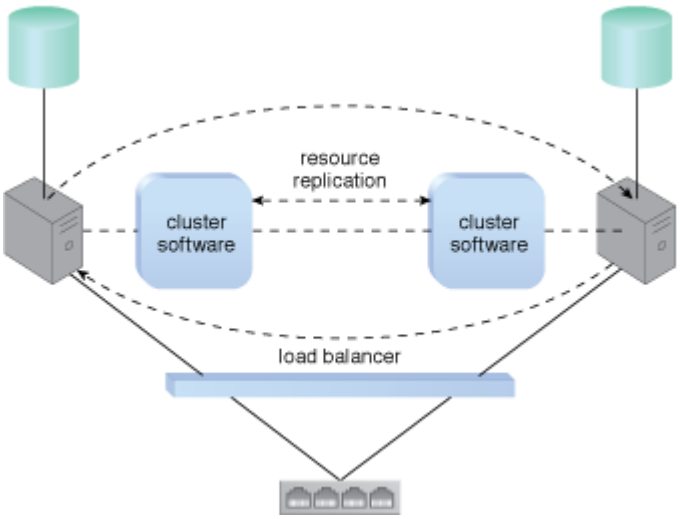


Figure 2 – A loosely coupled server cluster that incorporates a load balancer. There is no shared storage. Resource replication is used to replicate cloud storage devices through the network by the cluster software.

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.
- *High-Availability (HA) Cluster* – A high-availability cluster maintains system availability in the event of multiple node failures, and has redundant implementations of most of 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 different types of cloud service consumers, some of which may be incompatible with the cloud service's published service contract. Disparate cloud service consumers may be differentiated by their hosting hardware devices and/or may have different types of 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 by a wider range of cloud service consumer programs and devices (Figure 1).

Multi-device brokers commonly exist as 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

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.

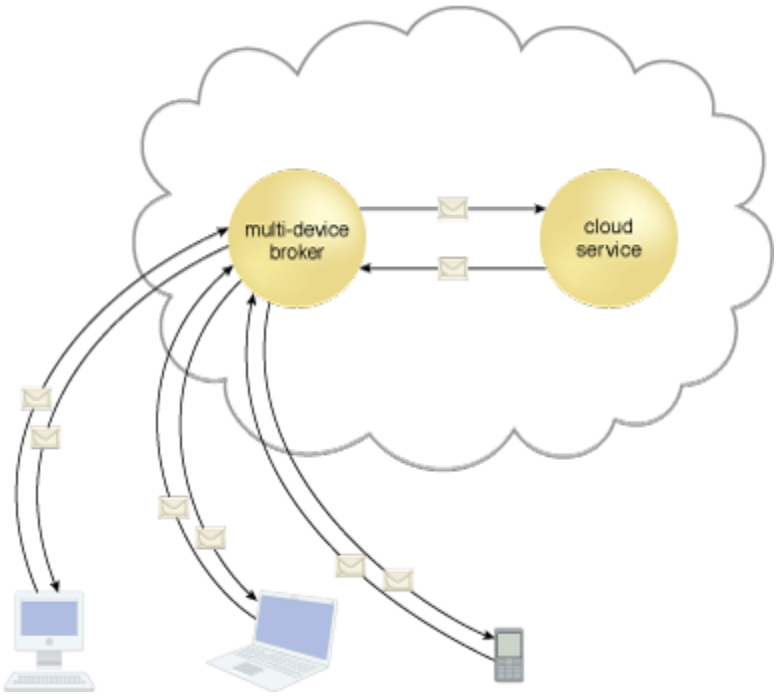


Figure 1 – A multi-device broker contains the mapping logic necessary to transform data exchanges between a cloud service and different types of cloud service consumer devices.

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 offload state data to the database in order to reduce the amount of run-time memory they consume (Figures 1 and 2). 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.

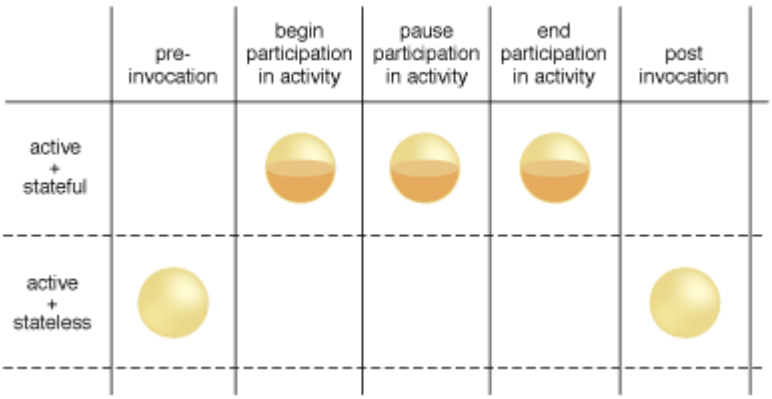


Figure 1 – During the lifespan of a cloud service instance, it may be required to remain stateful and keep state data cached in memory, even when idle.

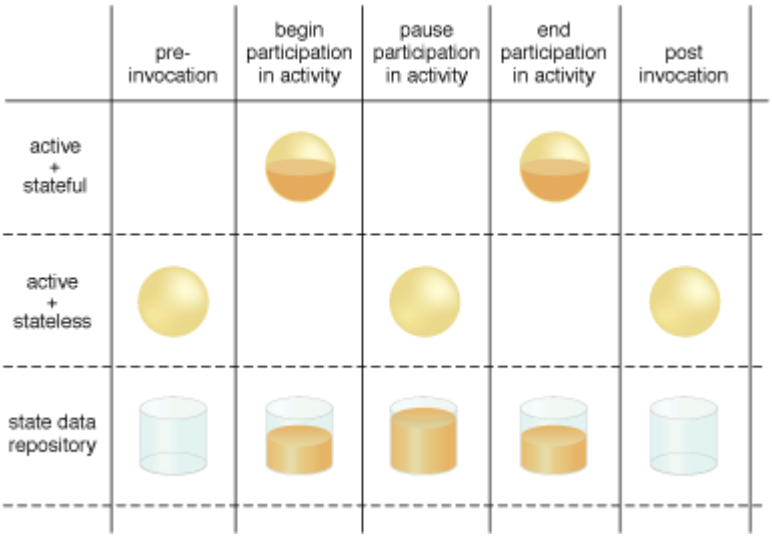


Figure 2 – By deferring state data to a state repository, the cloud service is able to transition to a stateless condition (or a partially stateless condition), thereby temporarily freeing system resources.

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 (Figure 1).

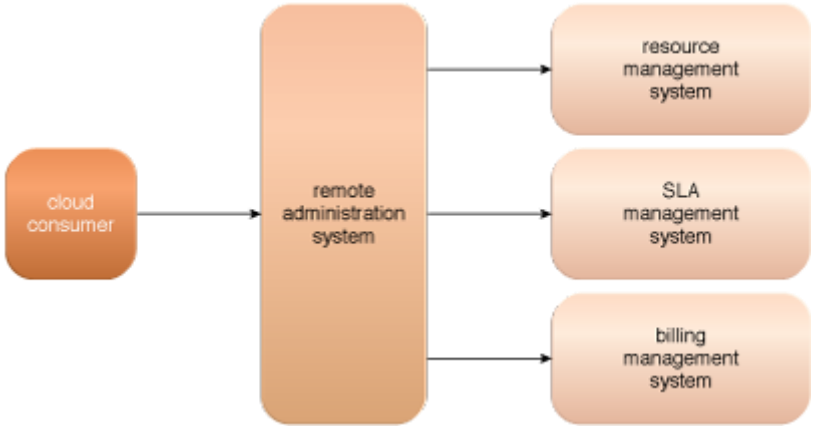


Figure 1 – The remote administration system abstracts underlying management systems to expose and centralize administration controls to external cloud resource administrators. The system provides a customizable user console, while programmatically interfacing with underlying management systems via their 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 centralized 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 2 illustrates a scenario involving a remote administration system and both usage and administration and self-service portals.

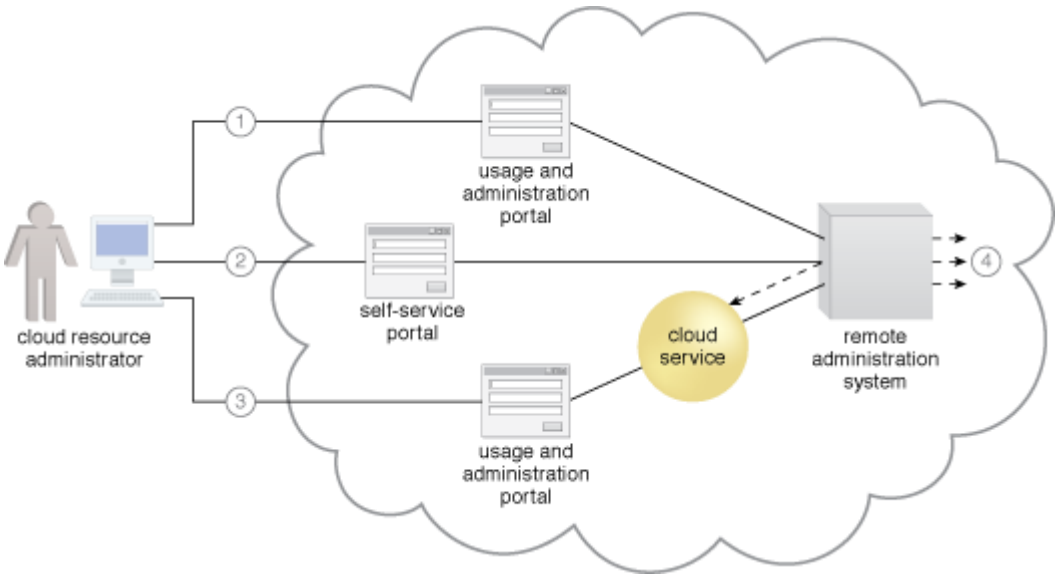


Figure 2 – A cloud resource administrator uses the usage and administration portal to configure an already leased virtual server (not shown) to prepare it for hosting (1). The cloud resource administrator then uses the self-service portal to select and request the provisioning of a new cloud service (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 fulfilment
- 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

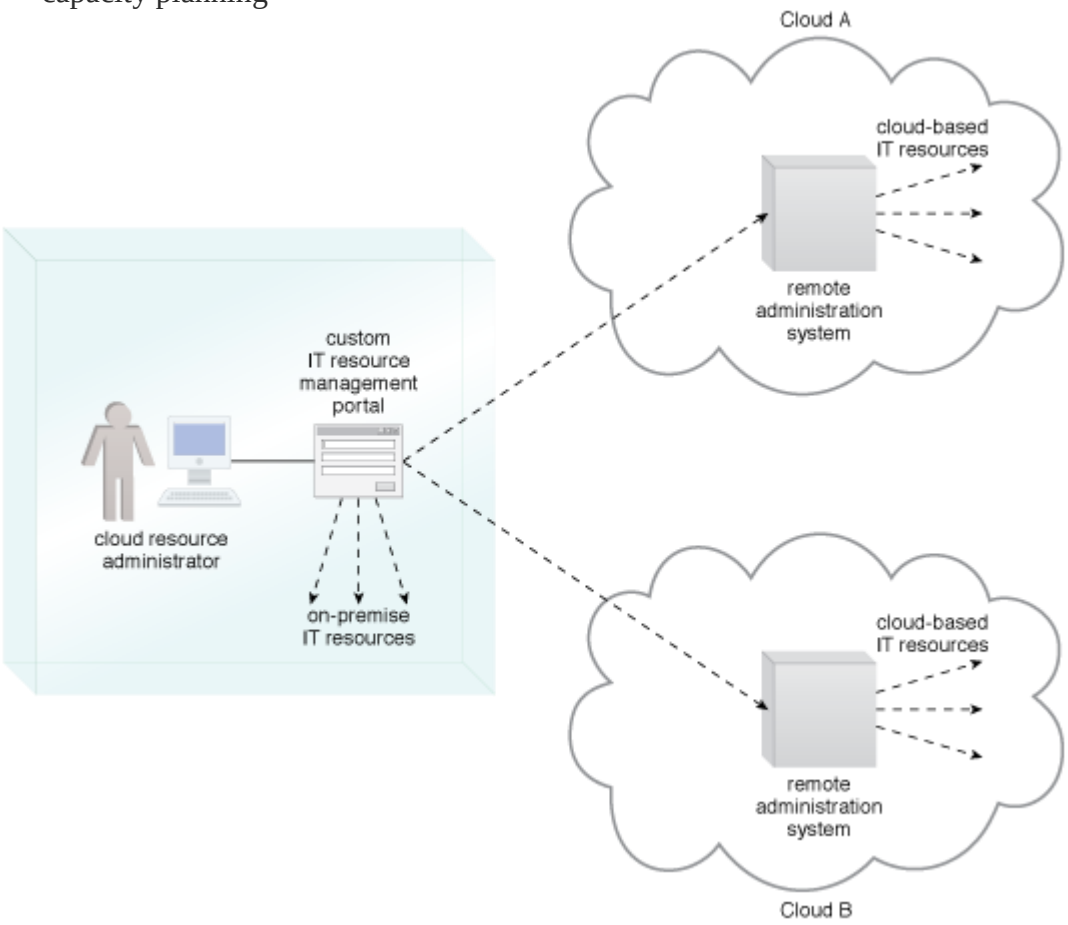


Figure 3 – Standardized APIs published by remote administration systems from different clouds enable a cloud consumer to develop a custom portal that centralizes a single IT resource management portal for both cloud-based and on-premise IT resources.

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 foreknowledge that it can reuse this console if it decides to move to another cloud provider that supports the same standardized API. Additionally, the cloud consumer would be able to further leverage standardized APIs if it is interested in leasing and centrally administering IT resources from multiple cloud providers and/or IT resources residing in cloud and on-premise environments.

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 1). 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).

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.

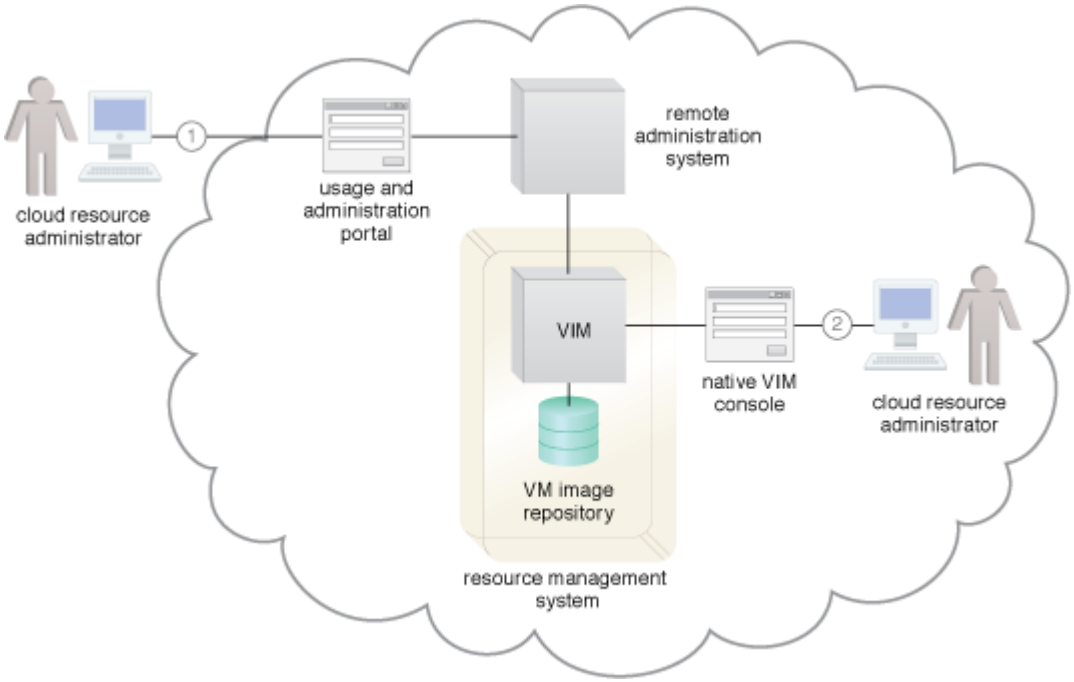


Figure 1 – The cloud consumer's cloud resource administrator accesses a usage and administration portal externally to administer a leased IT resource (1). The cloud provider's cloud resource administrator uses the native user-interface provided by the VIM to perform internal resource management tasks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

SLA Management System

The SLA monitor mechanism is used to specifically observe the runtime performance of cloud services to ensure that they are fulfilling the contractual QoS requirements published in SLAs (Figure 1). The data collected by the SLA monitor is processed by an SLA management system to be aggregated into SLA reporting metrics. This system can proactively repair or failover cloud services when exception conditions occur, such as when the SLA monitor reports a cloud service as "down."

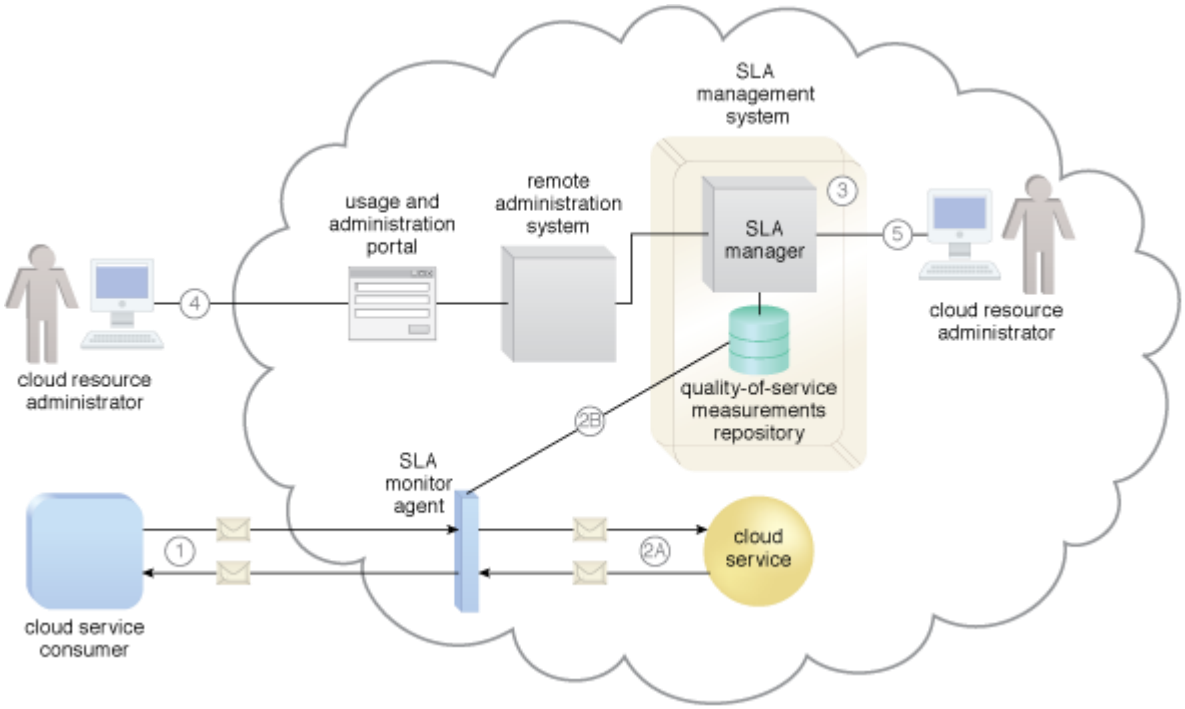


Figure 1 – The SLA monitor polls the cloud service by sending over polling request messages (MREQ1 to MREQN). The monitor receives polling response messages (M to M) that report that the service was "up" at each polling cycle (1a). The SLA monitor stores the "up" time—time period of all polling cycles 1 to N—in the log database (1b). The SLA monitor polls the cloud service that sends polling request messages (M to 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) and receives the polling response message (M) (3a). The SLA monitor stores the "up" time in the log database (3b).

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 (Figure 1).

The billing management system allows for the definition of different pricing policies as well as custom pricing models on a per-cloud consumer and/or per-IT resource basis. Pricing models can vary from the traditional pay-per-use models to flat-rate or pay-per-allocation models, or combinations thereof.

Billing arrangements can be based on pre-usage and post-usage payments. The latter type can include pre-defined limits or can be set up (with the mutual agreement of the cloud consumer) to allow for unlimited usage (and, consequently, no limit on subsequent billing). When limits are established, they are usually in the form of usage quotas. When quotas are exceeded, the billing management system can block further usage requests by cloud consumers.

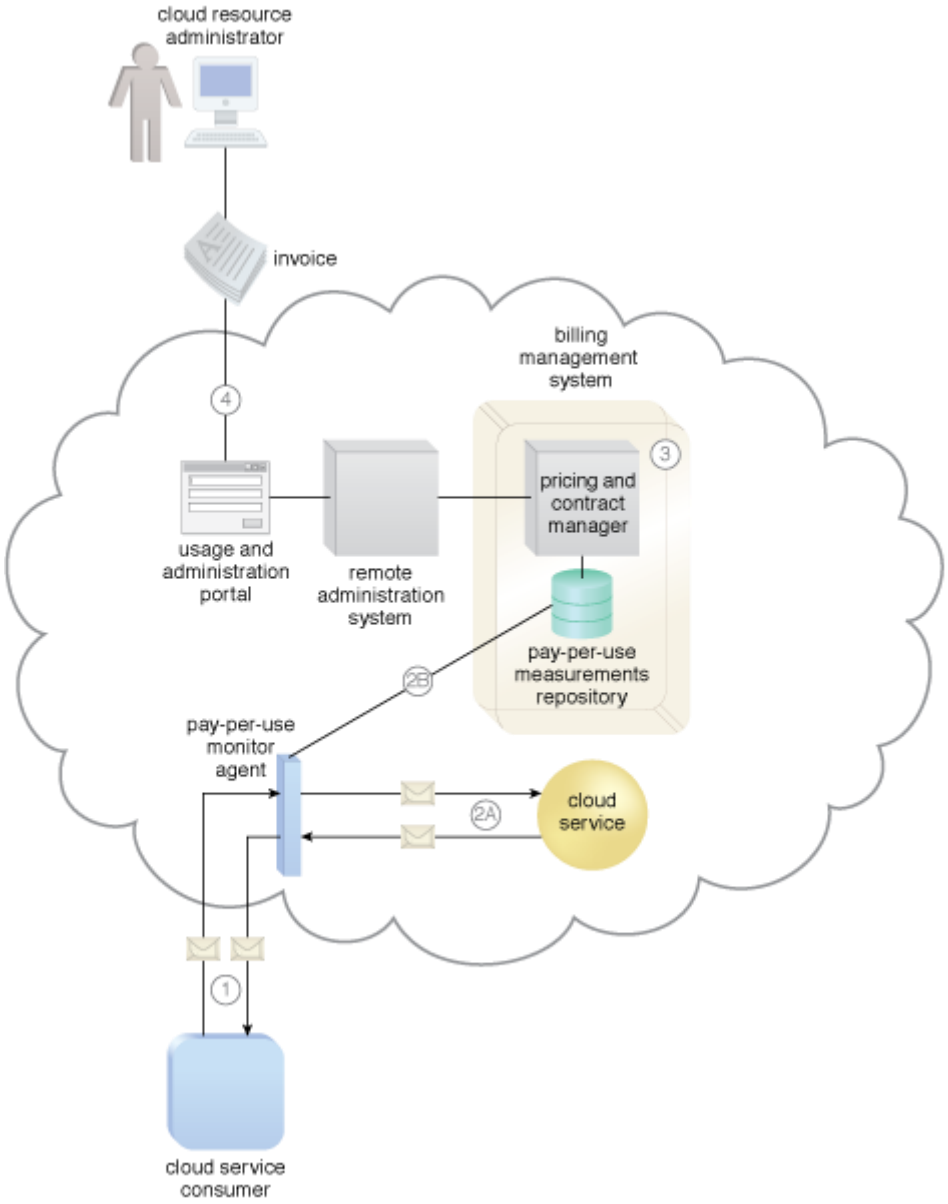


Figure 1 – A cloud service consumer exchanges messages with a cloud service (1). A pay-per-use monitor keeps track of the usage and collects data relevant to billing (2A), which is forwarded to a repository that is part of the billing management system (2B). The system periodically calculates the consolidated cloud service usage fees and generates an invoice for the cloud consumer (3). The invoice may be provided to the cloud consumer through the usage and administration portal (4).