Fundamental Cloud Architectures

This chapter introduces and describes several of the more common foundational cloud architectural models, each exemplifying a common usage and characteristic of contemporary cloud-based environments. The involvement and importance of different combinations of cloud computing mechanisms in relation to these architectures are explored.

1. Workload Distribution Architecture
2. Resource Pooling Architecture
3. Dynamic Scalability Architecture
4. Elastic Resource Capacity Architecture
5. Service Load Balancing Architecture
6. Cloud Bursting Architecture
7. Elastic Disk Provisioning Architecture
8. Redundant Storage Architecture

Workload Distribution Architecture

IT resources can be horizontally scaled via the addition of one or more identical IT resources, and a load balancer that provides runtime logic capable of evenly distributing the workload among the available IT resources (Figure 11.1). The resulting ***workload distribution architecture*** reduces both IT resource over-utilization and under-utilization to an extent dependent upon the sophistication of the load balancing algorithms and runtime logic.

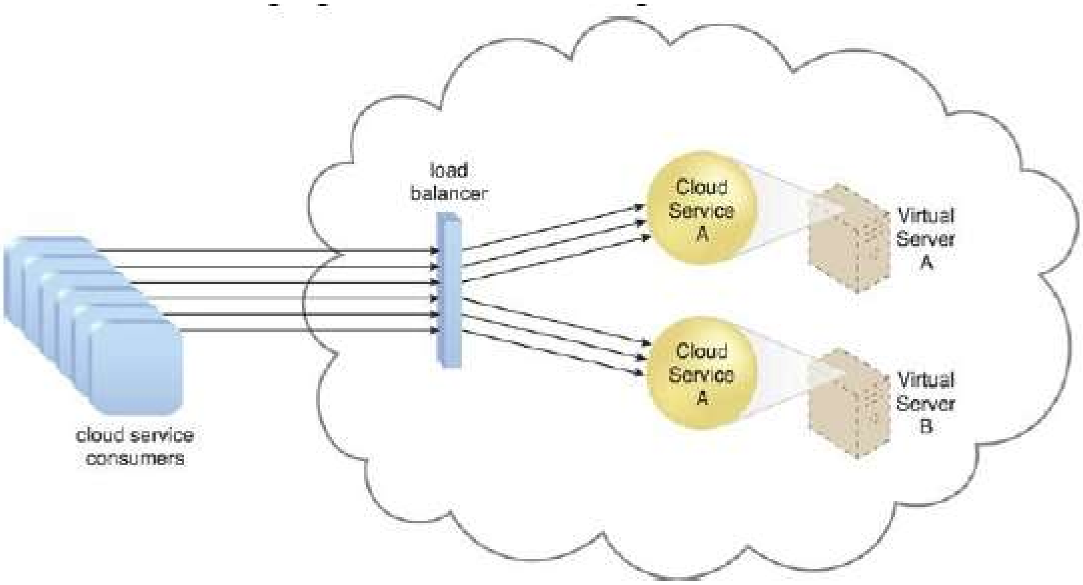


Figure 11.1. A redundant copy of Cloud Service A is implemented on Virtual Server B. The load balancer intercepts cloud service consumer requests and directs them to both Virtual Servers A and B to ensure even workload distribution.

This fundamental architectural model can be applied to any IT resource, with workload distribution commonly carried out in support of distributed virtual servers, cloud storage devices, and cloud services. Load balancing systems applied to specific IT resources usually produce specialized variations of this architecture that incorporate aspects of load balancing, such as:

- the service load balancing architecture explained later in this chapter
- the load balanced virtual server architecture
- the load balanced virtual switches architecture

In addition to the base load balancer mechanism, and the virtual server and cloud storage device mechanisms to which load balancing can be applied, the following mechanisms can also be part of this cloud architecture:

- ***Audit Monitor*** - When distributing runtime workloads, the type and geographical location of the IT resources that process the data can determine whether monitoring is necessary to fulfill legal and regulatory requirements.
- ***Cloud Usage Monitor*** - Various monitors can be involved to carry out runtime workload tracking and data processing.
- ***Hypervisor*** - Workloads between hypervisors and the virtual servers that they host may require distribution.
- ***Logical Network. Perimeter*** - The logical network perimeter isolates cloud consumer network boundaries in relation to how and where workloads are distributed.
- ***Resource Cluster*** - Clustered IT resources in active/active mode are commonly used to support workload balancing between different cluster nodes.
- ***Resource Replication*** - This mechanism can generate new instances of virtualized IT resources in response to runtime workload distribution demands.

Resource Pooling Architecture

A ***resource pooling architecture*** is based on the use of one or more resource pools, in which identical IT resources are grouped and maintained by a system that automatically ensures that they remain synchronized.

Provided here are common examples of resource pools:

- **Physical server pools** are composed of networked servers that have been installed with operating systems and other necessary programs and/or applications and are ready for immediate use.
- **Virtual server pools** are usually configured using one of several available templates chosen by the cloud consumer during provisioning. For example, a cloud consumer can set up a pool of mid-tier Windows servers with 4 GB of RAM or a pool of low-tier Ubuntu servers with 2 GB of RAM.
- **Storage pools, or cloud storage device pools**, consist of file-based or block-based storage structures that contain empty and/or filled cloud storage devices.
- **Network pools** (or interconnect pools) are composed of different preconfigured network connectivity devices. For example, a pool of virtual firewall devices or physical network switches can be created for redundant connectivity, load balancing, or link aggregation.
- **CPU pools** are ready to be allocated to virtual servers, and are typically broken down into individual processing cores.
- **Pools of physical RAM** can be used in newly provisioned physical servers or to vertically scale physical servers.

- **Dedicated pools** can be created for each type of IT resource and individual pools can be grouped into a larger pool, in which case each individual pool becomes a sub-pool (Figure 11.2).
- **Resource pools** can become highly complex, with multiple pools created for specific cloud consumers or applications. A hierarchical structure can be established to form parent, sibling, and nested pools in order to facilitate the organization of diverse resource pooling requirements.
- **Sibling resource pools** are usually drawn from physically grouped IT resources, as opposed to IT resources that are spread out over different data centers. Sibling pools are isolated from one another so that each cloud consumer is only provided access to its respective pool.
- **In the nested pool model**, larger pools are divided into smaller pools that individually group the same type of IT resources together . Nested pools can be used to assign resource pools to different departments or groups in the same cloud consumer organization.

After resources pools have been defined, multiple instances of IT resources from each pool can be created to provide an in-memory pool of "live" IT resources.

In addition to cloud storage devices and virtual servers, which are commonly pooled mechanisms, the following mechanisms can also be part of this cloud architecture:

- ***Audit Monitor*** - This mechanism monitors resource pool usage to ensure compliance with privacy and regulation requirements, especially when pools contain cloud storage devices or data loaded into memory.
- ***Cloud Usage Monitor*** - Various cloud usage monitors are involved in the runtime tracking and synchronization that are required by the pooled IT resources and any underlying management systems.
- ***Hypervisor*** - The hypervisor mechanism is responsible for providing virtual servers with access to resource pools, in addition to hosting the virtual servers and sometimes the resource pools themselves.
- ***Logical Network. Perimeter*** - The logical network perimeter is used to logically organize and isolate resource pools.
- ***Pay-Per-Use Monitor*** - The pay-per-use monitor collects usage and billing information on how individual cloud consumers are allocated and use IT resources from various pools.
- ***Remote Administration System*** - This mechanism is commonly used to interface with backend systems and programs in order to provide resource pool administration features via a front-end portal.
- ***Resource Management System*** - The resource management system mechanism supplies cloud consumers with the tools and permission management options for administering resource pools.
- ***Resource Replication*** - This mechanism is used to generate new instances of IT resources for resource pools.

Dynamic Scalability Architecture

The ***dynamic scalability architecture*** is an architectural model based on a system of predefined scaling conditions that trigger the dynamic allocation of IT resources from resource pools. Dynamic allocation enables variable utilization as dictated by usage demand fluctuations, since unnecessary IT resources are efficiently reclaimed without requiring manual interaction.

The automated scaling listener is configured with workload thresholds that dictate when new IT resources need to be added to the workload processing.

This mechanism can be provided with logic that determines how many additional IT resources can be dynamically provided, based on the terms of a given cloud consumer's provisioning contract.

The following types of dynamic scaling are commonly used:

- ***Dynamic Horizontal Scaling*** - IT resource instances are scaled out and in to handle fluctuating workloads. The automatic scaling listener monitors requests and signals resource replication to initiate IT resource duplication, as per requirements and permissions.
- ***Dynamic Vertical Scaling*** - IT resource instances are scaled up and down when there is a need to adjust the processing capacity of a single IT resource. For example, a virtual server that is being overloaded can have its memory dynamically increased or it may have a processing core added.
- ***Dynamic Relocation*** - The IT resource is relocated to a host with more capacity. For example, a database may need to be moved from a tape-based SAN storage device with 4 GB per second I/O capacity to another disk- based SAN storage device with 8 GB per second I/O capacity.

Figures 11.5 to 11.7 illustrate the process of dynamic horizontal scaling

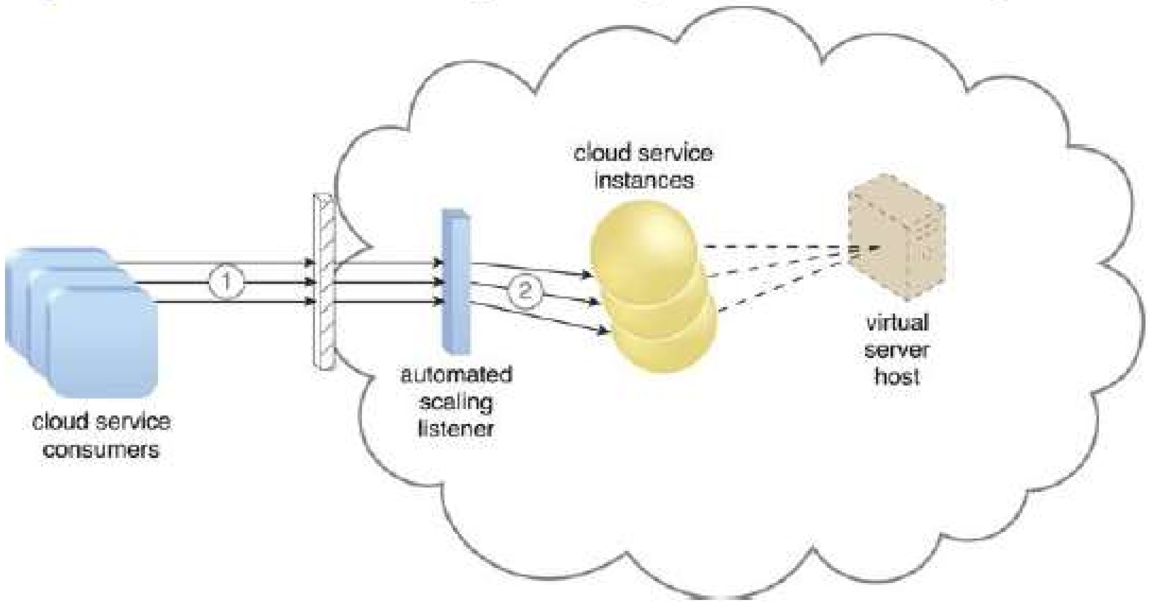


Figure 11.5. Cloud service consumers are sending requests to a cloud service (1). The automated scaling listener monitors the cloud service to determine if predefined capacity thresholds are being exceeded (2).

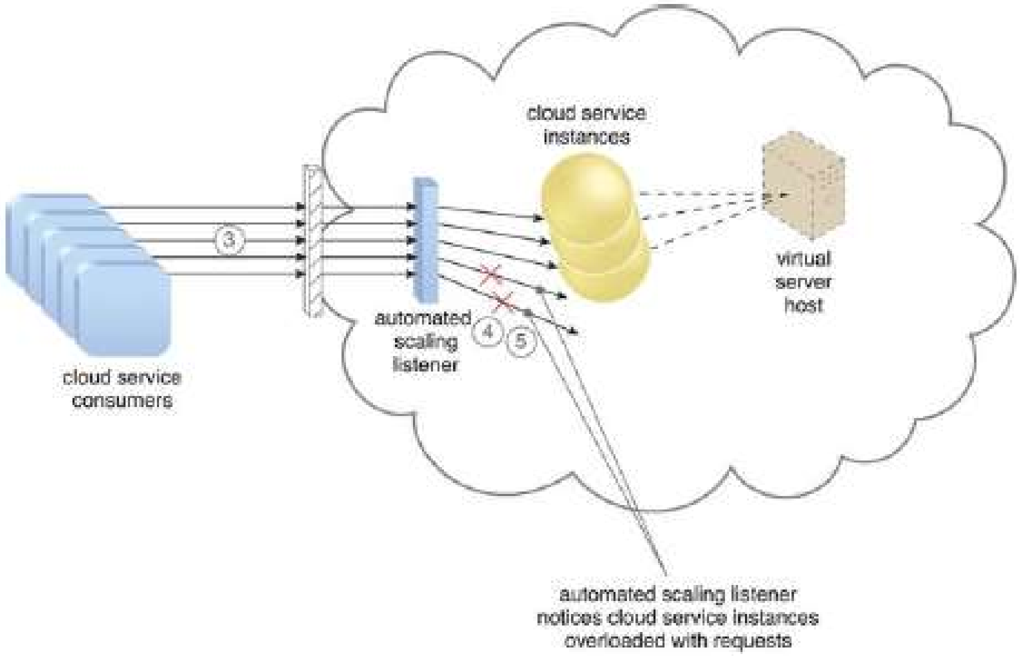


Figure 11.6. The number of requests coming from cloud service consumers increases (3). The workload exceeds the performance thresholds. The automated scaling listener determines the next course of action based on a predefined scaling policy (4). If the cloud service implementation is deemed eligible for additional scaling, the automated scaling listener initiates the scaling process (5).

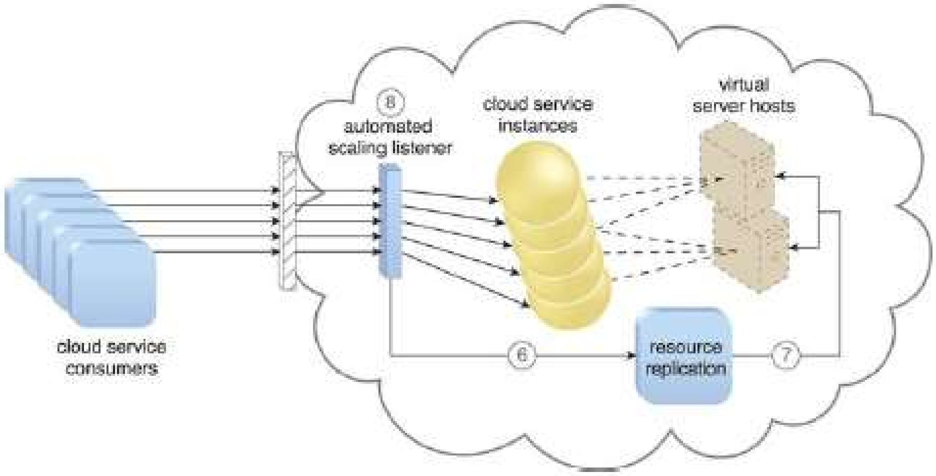


Figure 11.7. The automated scaling listener sends a signal to the resource replication mechanism (6), which creates more instances of the cloud service (7). Now that the increased workload has been accommodated, the automated scaling listener resumes monitoring and detracting and adding IT resources, as required (8).

The dynamic scalability architecture can be applied to a range of IT resources, including virtual servers and cloud storage devices. Besides the core automated scaling listener and resource replication mechanisms, the following mechanisms can also be used in this form of cloud architecture:

- ***Cloud Usage Monitor*** - Specialized cloud usage monitors can track runtime usage in response to dynamic fluctuations caused by this architecture.
- ***Hypervisor*** - The hypervisor is invoked by a dynamic scalability system to create or remove virtual server instances, or to be scaled itself.
- ***Pay-Per-Use Monitor*** - The pay-per-use monitor is engaged to collect usage cost information in response to the scaling of IT resources.

Elastic Resource Capacity Architecture

The ***elastic resource capacity architecture*** is primarily related to the dynamic provisioning of virtual servers, using a system that allocates and reclaims CPUs and RAM in immediate response to the fluctuating processing requirements of hosted IT resources (Figures 11.8 and 11.9)

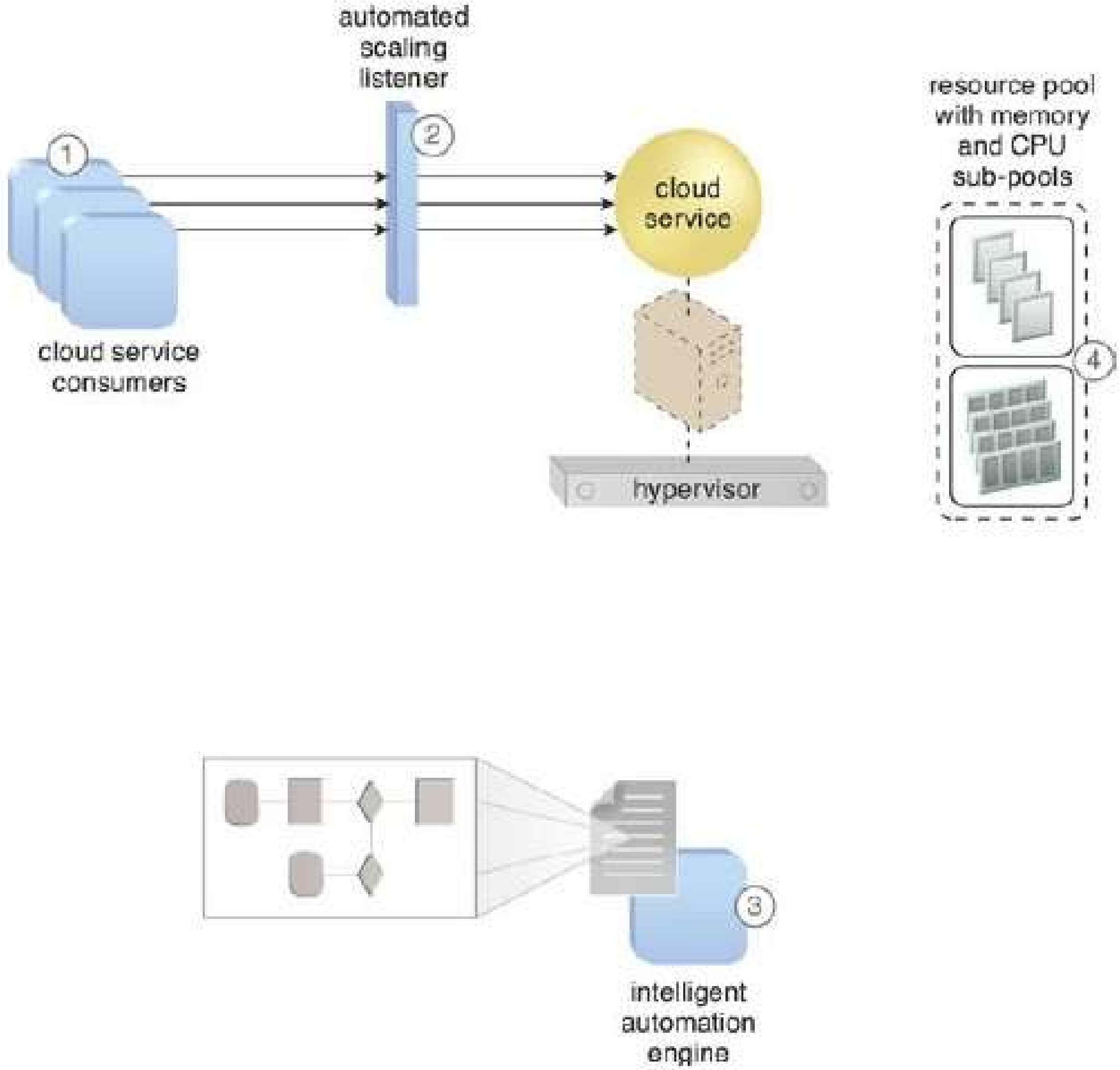

automated scaling listener
2
1
cloud service
resource pool with memory and CPU sub-pools
cloud service consumers
4
hypervisor
3
intelligent automation engine

Figure 11.8. Cloud service consumers are actively sending requests to a cloud service (1), which are monitored by an automated scaling listener (2). An intelligent automation engine script is deployed with workflow logic (3) that is capable of notifying the resource pool using allocation requests (4).

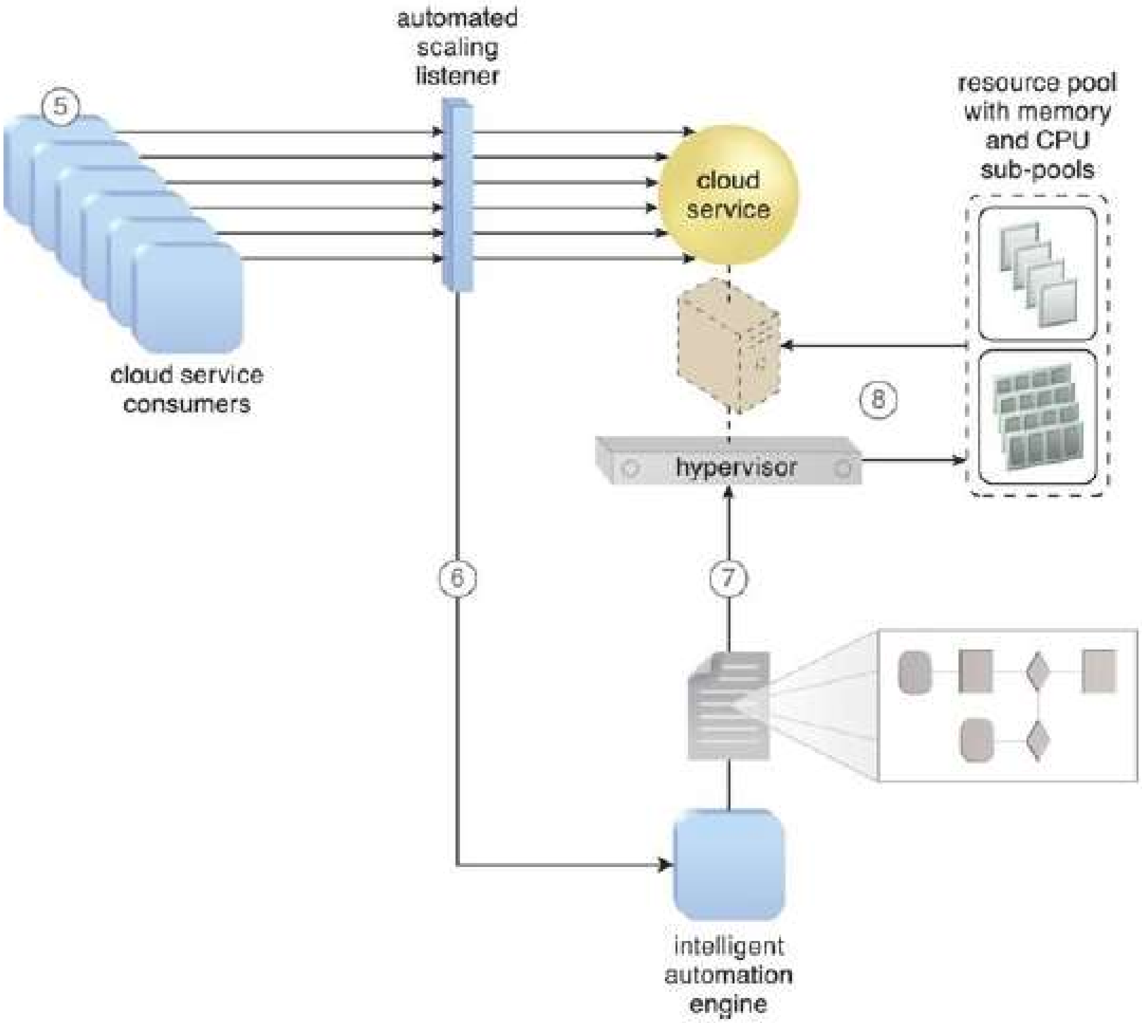


Figure 11.9. Cloud service consumer requests increase (5), causing the automated scaling listener to signal the intelligent automation engine to execute the script (6). The script runs the workflow logic that signals the hypervisor to allocate more IT resources from the resource pools (7). The hypervisor allocates additional CPU and RAM to the virtual server, enabling the increased workload to be handled (8).

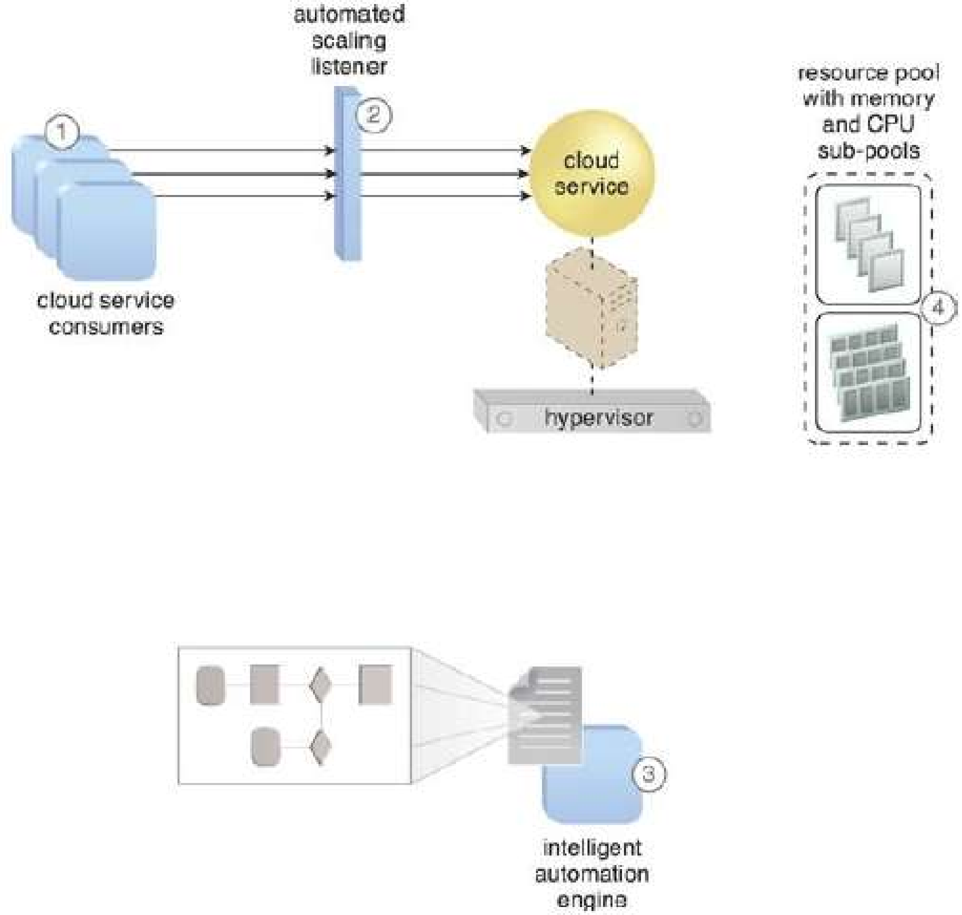

automated scaling listener
1
2
cloud service
cloud service consumers
resource pool with memory and CPU sub-pools
4
hypervisor
3
intelligent automation engine

Figure 11.8. Cloud service consumers are actively sending requests to a cloud service (1), which are monitored by an automated scaling listener (2). An intelligent automation engine script is deployed with workflow logic (3) that is capable of notifying the resource pool using allocation requests (4).

Resource pools are used by scaling technology that interacts with the hypervisor and/or VIM to retrieve and return CPU and RAM resources at runtime. The runtime processing of the virtual server is monitored so that additional processing power can be leveraged from the resource pool via dynamic allocation, before capacity thresholds are met. The virtual server and its hosted applications and IT resources are vertically scaled in response.

This type of cloud architecture can be designed so that the intelligent automation engine script sends its scaling request via the VIM instead of to the hypervisor directly. Virtual servers that participate in elastic resource allocation systems may require rebooting in order for the dynamic resource allocation to take effect.

Some additional mechanisms that can be included in this cloud architecture are the following:

- ***Cloud Usage Monitor*** - Specialized cloud usage monitors collect resource usage information on IT resources before, during, and after scaling, to help define the future processing capacity thresholds of the virtual servers.
- ***Pay-Per-Use Monitor*** - The pay-per-use monitor is responsible for collecting resource usage cost

information as it fluctuates with the elastic provisioning.

- ***Resource Replication*** - Resource replication is used by this architectural model to generate new instances of the scaled IT resources.

Service Load Balancing Architecture

The ***service load balancing architecture*** can be considered a specialized variation of the workload distribution architecture that is geared specifically for scaling cloud service implementations. Redundant deployments of cloud services are created, with a load balancing system added to dynamically distribute workloads.

The duplicate cloud service implementations are organized into a resource pool, while the load balancer is positioned as either an external or built-in component to allow the host servers to balance the workloads themselves.

Depending on the anticipated workload and processing capacity of host server environments, multiple instances of each cloud service implementation can be generated as part of a resource pool that responds to fluctuating request volumes more efficiently.

The load balancer can be positioned either independent of the cloud services and their host servers (Figure 11.10), or built-in as part of the application or server’s environment. In the latter case, a primary server with the load balancing logic can communicate with neighboring servers to balance the workload (Figure 11.11)

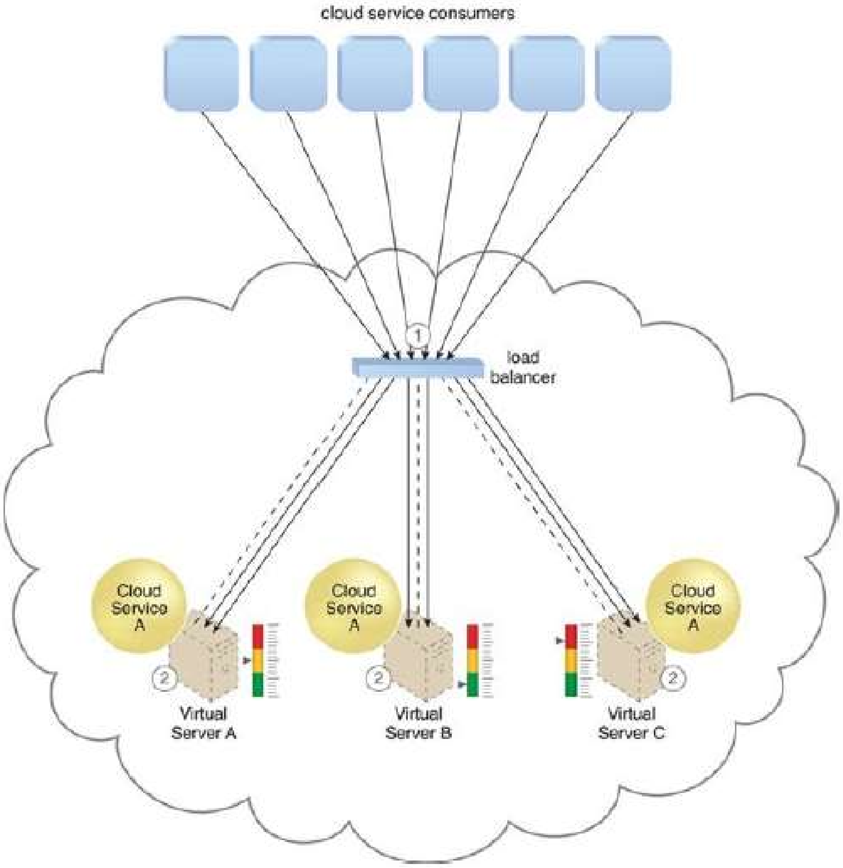


Figure 11.10. The load balancer intercepts messages sent by cloud service consumers (1) and forwards them to the virtual servers so that the workload processing is horizontally scaled (2).

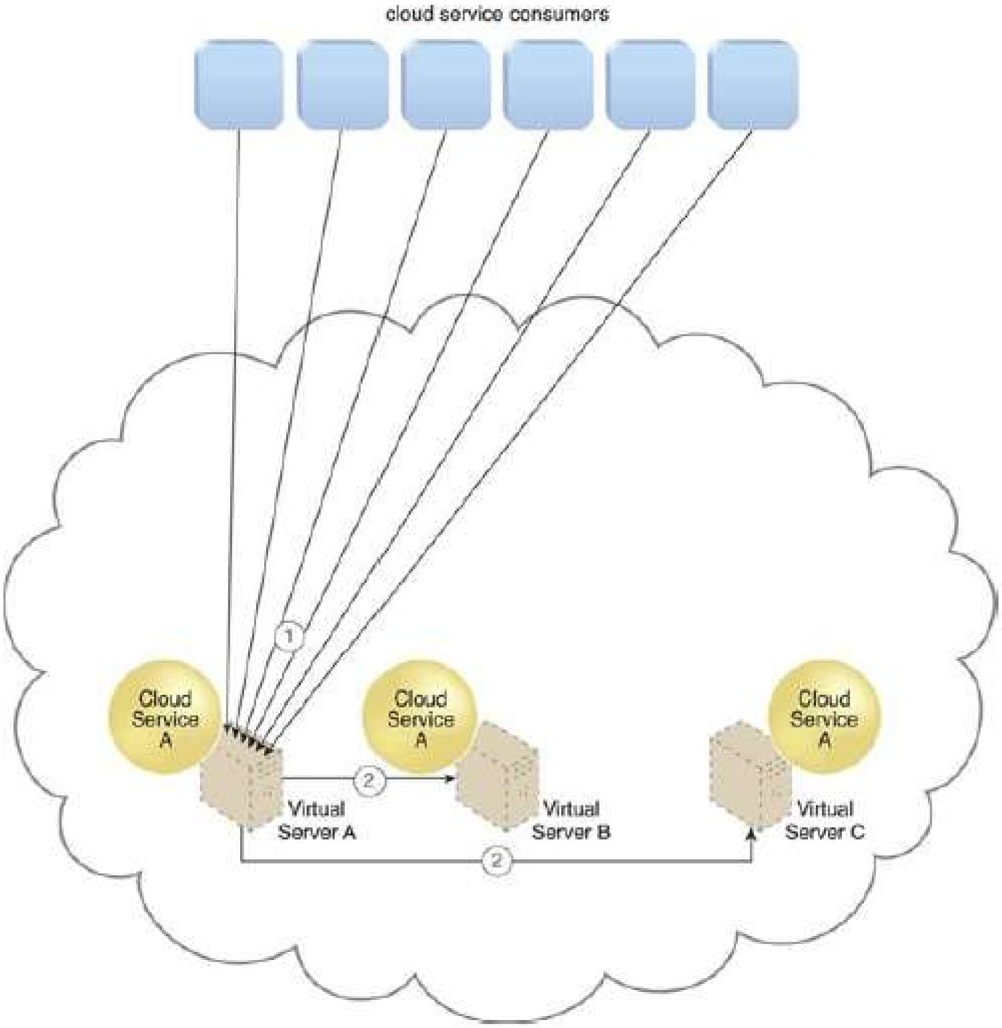


Figure 11.11. Cloud service consumer requests are sent to Cloud Service A on Virtual Server A (1). The cloud service implementation includes built-in load balancing logic that is capable of distributing requests to the neighboring Cloud Service A implementations on Virtual Servers B and C (2).

The service load balancing architecture can involve the following mechanisms in addition to the load balancer:

- ***Cloud Usage Monitor*** - Cloud usage monitors may be involved with monitoring cloud service instances and their respective IT resource consumption levels, as well as various runtime monitoring and usage data collection tasks.
- ***Resource Cluster*** - Active-active cluster groups are incorporated in this architecture to help balance workloads across different members of the cluster.
- ***Resource Replication*** - The resource replication mechanism is utilized to generate cloud service implementations in support of load balancing requirements.

Cloud Bursting Architecture

The ***cloud bursting architecture*** establishes a form of dynamic scaling that scales or “bursts out” on-premise IT resources into a cloud whenever predefined capacity thresholds have been reached. The corresponding cloud-based IT resources are redundantly pre-deployed but remain inactive until cloud bursting occurs. After they are no longer required, the cloud-based IT resources are released and the architecture “bursts in” back to the on-premise environment.

Cloud bursting is a flexible scaling architecture that provides cloud consumers with the option of using cloud-based IT resources only to meet higher usage demands. The foundation of this architectural model is based on the automated scaling listener and resource replication mechanisms.

The automated scaling listener determines when to redirect requests to cloud- based IT resources, and resource replication is used to maintain synchronicity between on-premise and cloud-based IT resources in relation to state information (Figure 11.12**)**

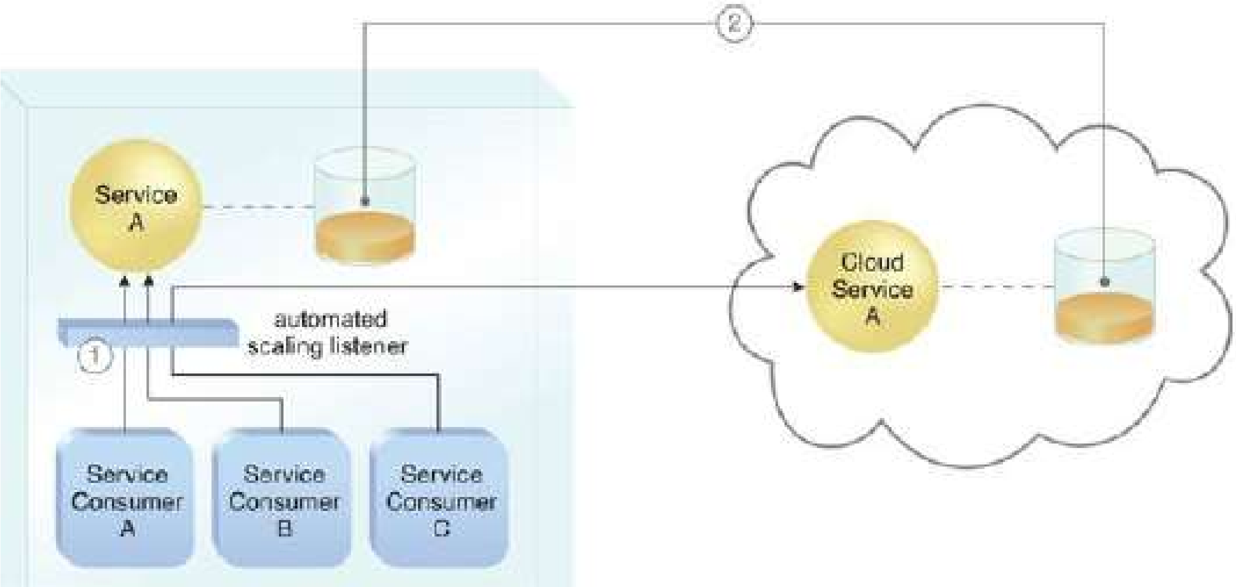


Figure 11.12. An automated scaling listener monitors the usage of on-premise Service A, and redirects Service Consumer C’s request to Service A’s redundant implementation in the cloud (Cloud Service A) once Service A’s usage threshold has been exceeded (1). A resource replication system is used to keep state management databases synchronized (2).

Elastic Disk Provisioning Architecture

Cloud consumers are commonly charged for cloud-based storage space based on fixed-disk storage allocation, meaning the charges are predetermined by disk capacity and not aligned with actual data storage consumption. Figure 11.13 demonstrates this by illustrating a scenario in which a cloud consumer provisions a virtual server with the Windows Server operating system and three 150 GB hard drives. The cloud consumer is billed for using 450 GB of storage space after installing the operating system, even though the operating system only requires 15 GB of storage space.

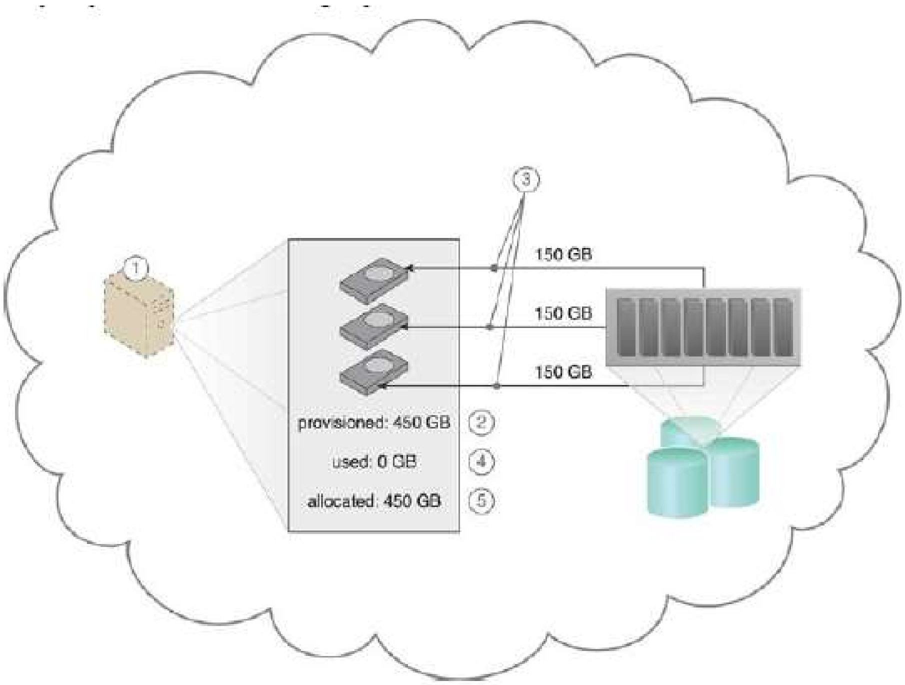


Figure 11.13. The cloud consumer requests a virtual server with three hard disks, each with a capacity of 150 GB (1). The virtual server is provisioned according to the elastic disk provisioning architecture, with a total of 450 GB of disk space (2). The 450 GB is allocated to the virtual server by the cloud provider (3). The cloud consumer has not installed any software yet, meaning the actual used space is currently 0 GB (4). Because the 450 GB are already allocated and reserved for the cloud consumer, it will be charged for 450 GB of disk usage as of the point of allocation (5).

The ***elastic disk provisioning architecture*** establishes a dynamic storage provisioning system that ensures that the cloud consumer is granularly billed for the exact amount of storage that it actually uses. This system uses thin- provisioning technology for the dynamic allocation of storage space, and is further supported by runtime usage monitoring to collect accurate usage data for billing purposes.

The following mechanisms can be included in this architecture in addition to the cloud storage device, virtual server, hypervisor, and pay-per-use monitor:

- ***Cloud Usage Monitor*** - Specialized cloud usage monitors can be used to track and log storage usage fluctuations.
- ***Resource Replication*** - Resource replication is part of an elastic disk provisioning system when conversion of dynamic thin-disk storage into static thick-disk storage is required

Redundant Storage Architecture

Cloud storage devices are occasionally subject to failure and disruptions that are caused by network connectivity issues, controller or general hardware failure, or security breaches. A compromised cloud storage device's reliability can have a ripple effect and cause impact failure across all of the services, applications, and infrastructure components in the cloud that are reliant on its availability.

The ***redundant storage architecture*** introduces a secondary duplicate cloud storage device as part of a failover system that synchronizes its data with the data in the primary cloud storage device. A storage service gateway diverts cloud consumer requests to the secondary device whenever the primary device fails (Figures 11.16 and 11.17)

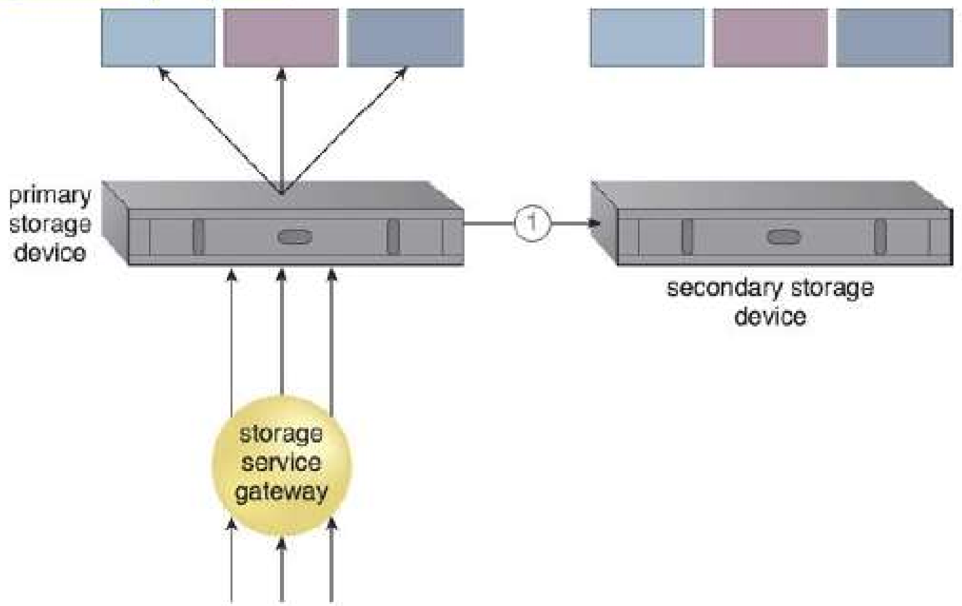


Figure 11.16. The primary cloud storage device is routinely replicated to the secondary cloud storage device (1).

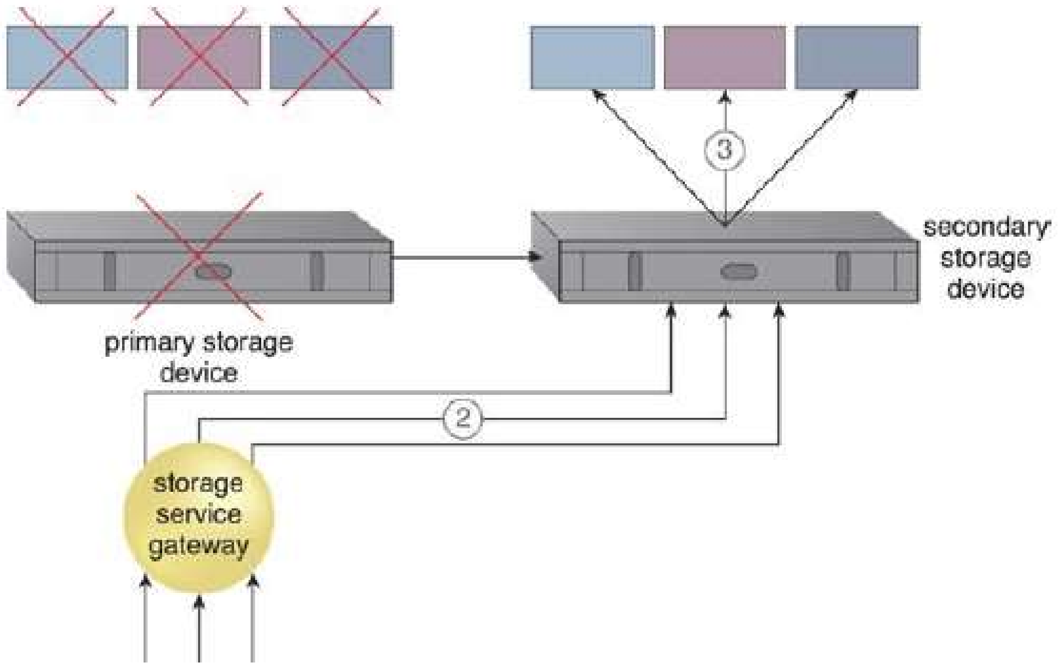


Figure 11.17. The primary storage becomes unavailable and the storage service gateway forwards the cloud consumer requests to the secondary storage device (2). The secondary storage device forwards the requests to the LUNs, allowing cloud consumers to continue to access their data (3).

Advanced Cloud Architectures

- Hypervisor Clustering Architecture
- Load Balanced Virtual Server Instances Architecture
- Non-Disruptive Service Relocation Architecture
- Zero Downtime Architecture
- Cloud Balancing Architecture
- Resource Reservation Architecture
- Dynamic Failure Detection and Recovery Architecture
- Bare-Metal Provisioning Architecture
- Rapid Provisioning Architecture
- Storage Workload Management Architecture

Hypervisor Clustering Architecture

Hypervisors can be responsible for creating and hosting multiple virtual servers. Because of this dependency, any failure conditions that affect a hypervisor can cascade to its virtual servers (Figure 12.1)

cascade to its virtual servers (Figure 12.1).

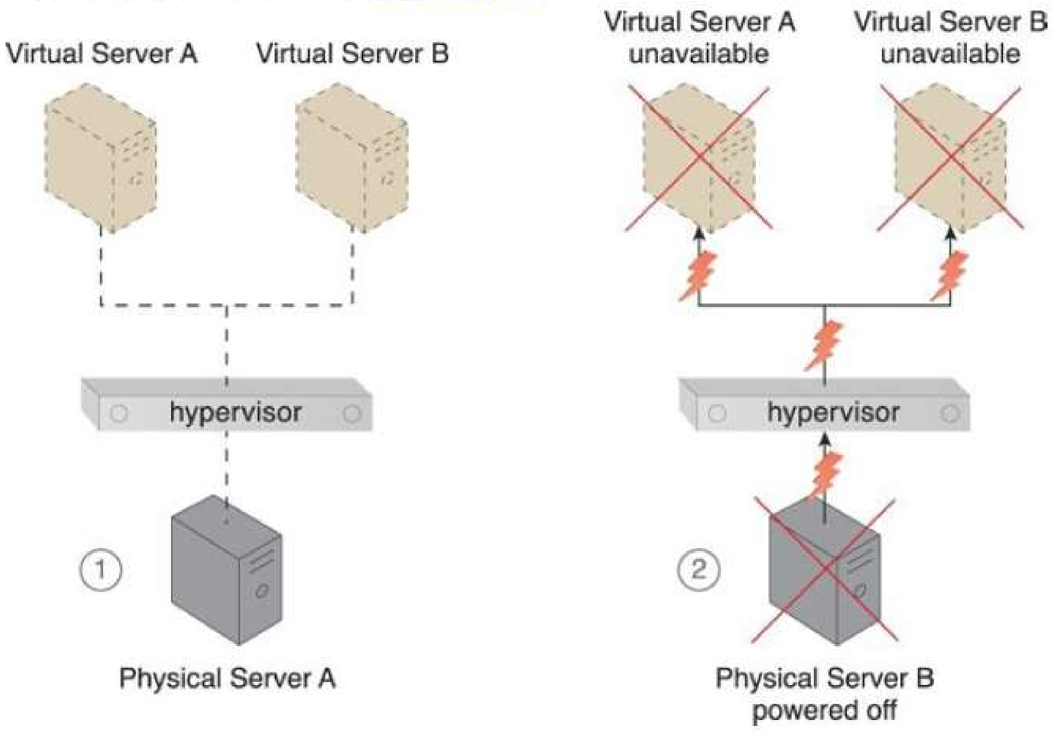

Virtual Server A
Virtual Server B
Virtual Server A unavailable
Virtual Server B unavailable
hypervisor
hypervisor
1
2
Physical Server A
Physical Server B powered off

Figure 12.1. Physical Server A is hosting a hypervisor that hosts Virtual Servers A and B (1). When Physical Server A fails, the hypervisor and two virtual servers consequently fail as well (2).

Heartbeats are system-level messages exchanged between hypervisors, hypervisors and virtual servers, and hypervisors and VIMs.

The ***hypervisor clustering architecture*** establishes a high-availability cluster of hypervisors across multiple physical servers. If a given hypervisor or its underlying physical server becomes unavailable, the hosted virtual servers can be moved to another physical server or hypervisor to maintain runtime operations (Figure 12.2)

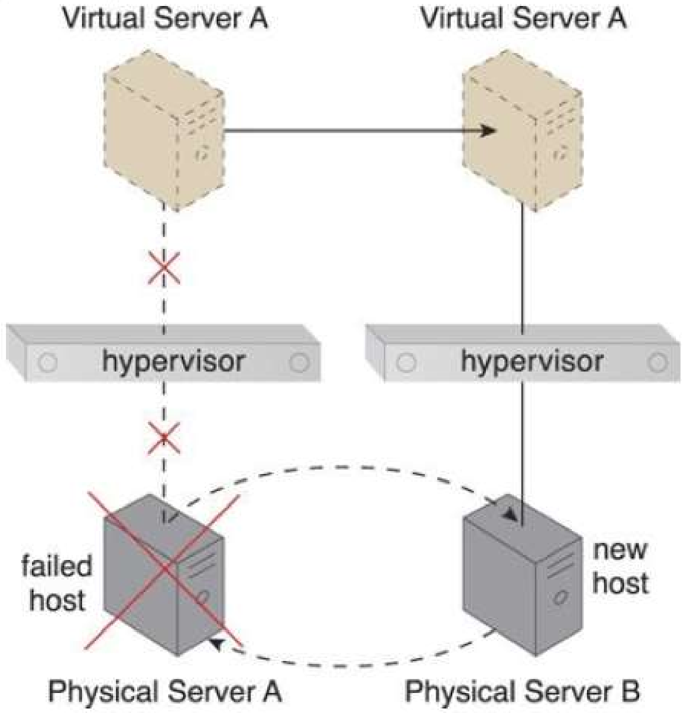


Figure 12.2. Physical Server A becomes unavailable and causes its hypervisor to fail. Virtual Server A is migrated to Physical Server B, which has another hypervisor that is part of the cluster to which Physical Server A belongs.

The hypervisor cluster is controlled via a central VIM, which sends regular heartbeat messages to the hypervisors to confirm that they are up and running. Unacknowledged heartbeat messages cause the VIM to initiate the live VM migration program, in order to dynamically move the affected virtual servers to a new host.

In addition to the hypervisor and resource cluster mechanisms that form the core of this architectural model and the virtual servers that are protected by the clustered environment, the following mechanisms can be incorporated:

- ***Logical Network Perimeter*** - The logical boundaries created by this mechanism ensure that none of the hypervisors of other cloud consumers are accidentally included in a given cluster.
- ***Resource Replication*** - Hypervisors in the same cluster inform one another about their status and availability. Updates on any changes that occur in the cluster, such as the creation or deletion of a virtual switch, need to be replicated to all of the hypervisors via the VIM.

Load Balanced Virtual Server Instances Architecture

Keeping cross-server workloads evenly balanced between physical servers whose operation and management are isolated can be challenging. A physical server can easily end up hosting more virtual servers or receive larger workloads than its neighboring physical servers (Figure 12.7). Both physical server over and under-utilization can increase dramatically over time, leading to on-going performance challenges (for over-utilized servers) and constant waste (for the lost processing potential of under-utilized servers).

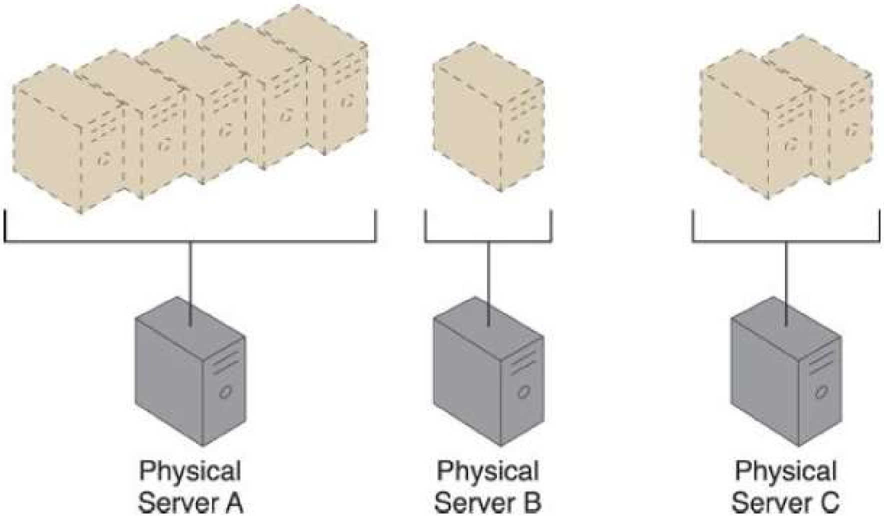


Figure 12.7. Three physical servers have to host different quantities of virtual server instances, leading to both over-utilized and under-utilized servers.

The ***load balanced virtual server instances architecture*** establishes a capacity watchdog system that dynamically calculates virtual server instances and associated workloads, before distributing the processing across available physical server hosts (Figure 12.8)

The capacity watchdog system is comprised of a capacity watchdog cloud usage monitor, the live VM migration program, and a capacity planner. The capacity watchdog monitor tracks physical and virtual server usage and reports any significant fluctuations to the capacity planner, which is responsible for dynamically calculating physical server computing capacities against virtual server capacity requirements. If the capacity planner decides to move a virtual server to another host to distribute the workload, the live VM migration program is signaled to move the virtual server.

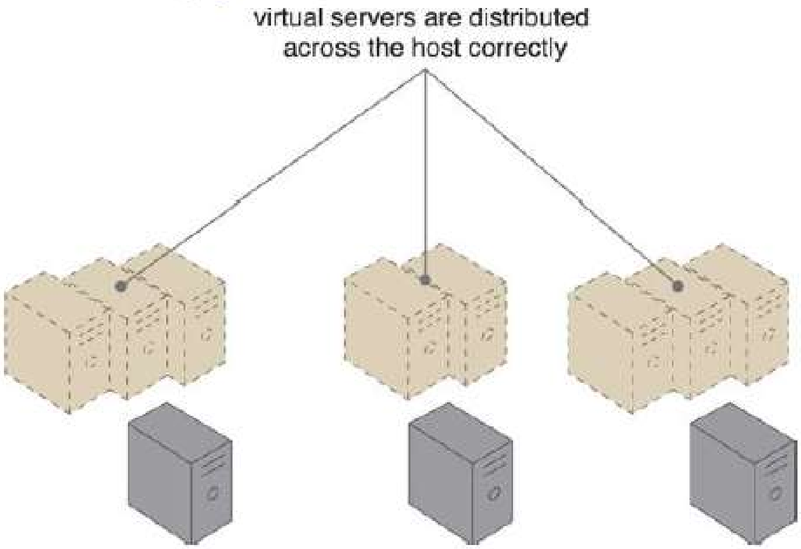


Figure 12.8. The virtual server instances are more evenly distributed across the physical server hosts.

The following mechanisms can be included in this architecture, in addition to the hypervisor, resource clustering, virtual server, and (capacity watchdog) cloud usage monitor:

- ***Automated Scaling Listener*** - The automated scaling listener may be used to initiate the process of load balancing and to dynamically monitor workload coming to the virtual servers via the hypervisors.
- ***Load Balancer*** - The load balancer mechanism is responsible for distributing the workload of the virtual servers between the hypervisors.
- ***Logical Network. Perimeter*** - A logical network perimeter ensures that the destination of a given relocated virtual server is in compliance with SLA and privacy regulations.

Resource Replication - The replication of virtual server instances may be required as part of the load balancing functionality

Non-Disruptive Service Relocation Architecture

A cloud service can become unavailable for a number of reasons, such as:

- runtime usage demands that exceed its processing capacity
- a maintenance update that mandates a temporary outage
- permanent migration to a new physical server host

Cloud service consumer requests are usually rejected if a cloud service becomes unavailable, which can potentially result in exception conditions. Rendering the cloud service temporarily unavailable to cloud consumers is not preferred even if the outage is planned.

The ***non-disruptive service relocation architecture*** establishes a system by which a predefined event triggers the duplication or migration of a cloud service implementation at runtime, thereby avoiding any disruption. Instead of scaling cloud services in or out with redundant implementations, cloud service activity can be temporarily diverted to another hosting environment at runtime by adding a duplicate implementation onto a new host. Similarly, cloud service consumer requests can be temporarily redirected to a duplicate implementation when the original implementation needs to undergo a maintenance outage. The relocation of the cloud service implementation and any cloud service activity can also be permanent to accommodate cloud service migrations to new physical server hosts.

A key aspect of the underlying architecture is that the new cloud service implementation is guaranteed to be successfully receiving and responding to cloud service consumer requests ***before*** the original cloud service implementation is deactivated or removed. A common approach is for live VM migration to move the entire virtual server instance that is hosting the cloud service. The automated scaling listener and/or load balancer mechanisms can be used to trigger a temporary redirection of cloud service consumer requests, in response to scaling and workload distribution requirements. Either mechanism can contact the VIM to initiate the live VM migration process, as shown in Figures 12.12 **.**

Virtual server migration can occur in one of the following two ways, depending on the location of the virtual server's disks and configuration:

- A copy of the virtual server disks is created on the destination host, if the virtual server disks are stored on a local storage device or non-shared remote storage devices attached to the source host. After the copy has been created, both virtual server instances are synchronized and virtual server files are removed from the origin host.

- Copying the virtual server disks is unnecessary if the virtual server's files are stored on a remote storage device that is shared between origin and destination hosts. Ownership of the virtual server is simply transferred from the origin to the destination physical server host, and the virtual server's state is automatically synchronized.

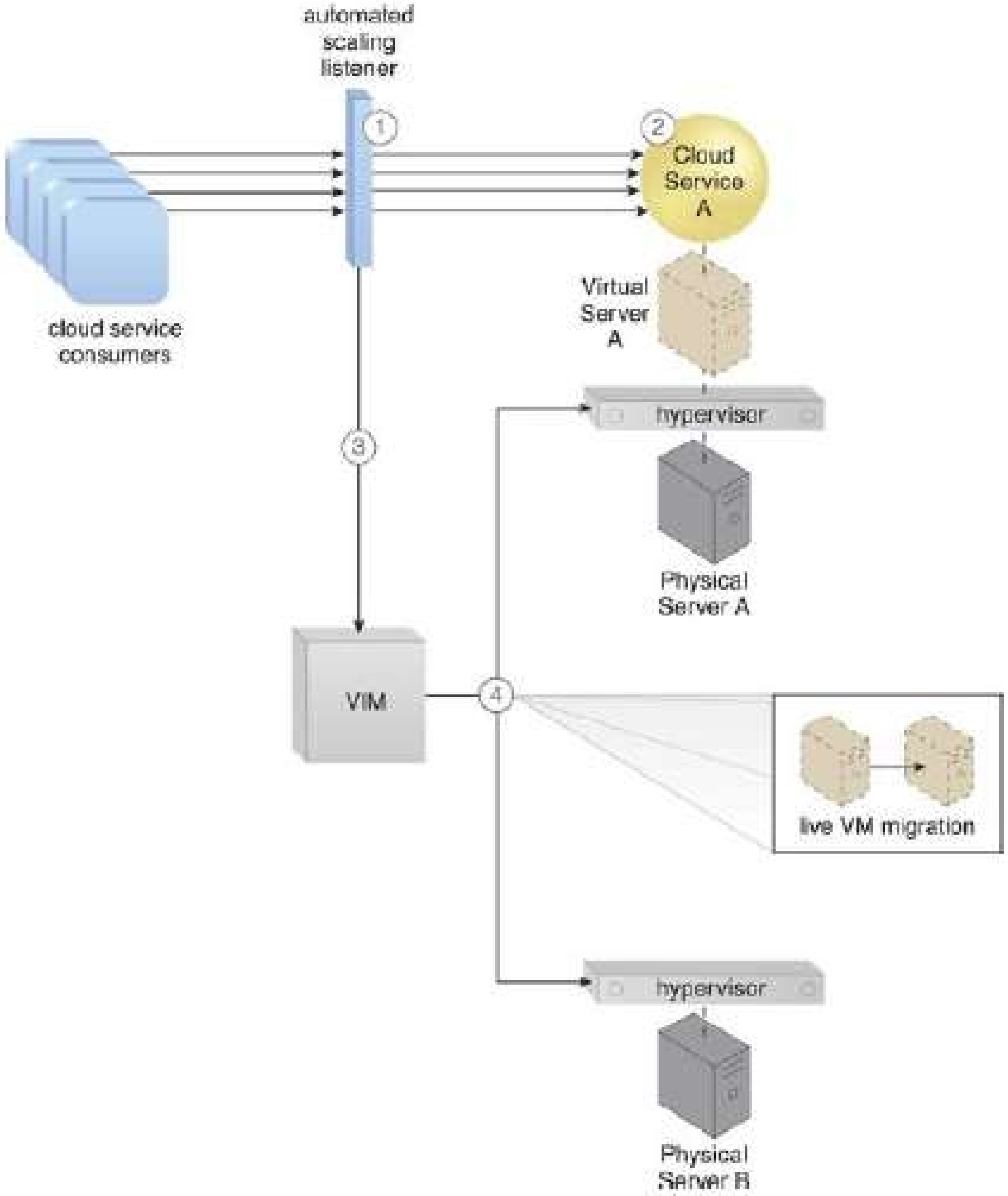


Figure 12.12. The automated scaling listener monitors the workload for a cloud service (1). The cloud service's predefined threshold is reached as the workload increases (2), causing the automated scaling listener to signal the VIM to initiate relocation (3). The VIM uses the live VM migration program to instruct both the origin and destination hypervisors to carry out runtime relocation (4).

This architecture can be supported by the persistent virtual network configurations architecture, so that the defined network configurations of migrated virtual servers are preserved to retain connection with the cloud service consumers.

Besides the automated scaling listener, load balancer, cloud storage device, hypervisor, and virtual server, other mechanisms that can be part of this architecture include the following:

- ***Cloud Usage Monitor*** - Different types of cloud usage monitors can be used to continuously track IT resource usage and system activity.
- ***Pay-Per-Use Monitor*** - The pay-per-use monitor is used to collect data for service usage cost calculations for IT resources at both source and destination locations.
- ***Resource Replication*** - The resource replication mechanism is used to instantiate the shadow copy of the cloud service at its destination.
- ***SLA Management System*** - This management system is responsible for processing SLA data provided by the SLA monitor to obtain cloud service availability assurances, both during and after cloud service duplication or relocation.
- ***SLA Monitor*** - This monitoring mechanism collects the SLA information required by the SLA management system, which may be relevant if availability guarantees rely on this architecture.

Zero Downtime Architecture

A physical server naturally acts as a single point of failure for the virtual servers it hosts. As a result, when the physical server fails or is compromised, the availability of any (or all) hosted virtual servers can be affected. This makes the issuance of zero downtime guarantees by a cloud provider to cloud consumers challenging.

The zero downtime architecture establishes a sophisticated failover system that allows virtual servers to be dynamically moved to different physical server hosts, in the event that their original physical server host fails (figure 12.15).

Multiple physical servers are assembled into a group that is controlled by a fault tolerance system capable of switching activity from one physical server to another, without interruption. The live VM migration component is typically a core part of this form of high availability cloud architecture.

The resulting fault tolerance assures that, in case of physical server failure, hosted virtual servers will be migrated to a secondary physical server. All virtual servers are stored on a shared volume (as per the persistent virtual network configuration architecture) so that other physical server hosts in the same group can access their files.

Besides the failover system, cloud storage device, and virtual server mechanisms, the following mechanisms can be part of this architecture:

- ***Audit Monitor*** - This mechanism may be required to check whether the relocation of virtual servers also relocates hosted data to prohibited locations.
- ***Cloud Usage Monitor*** - Incarnations of this mechanism are used to monitor the actual IT resource usage of cloud consumers to help ensure that virtual server capacities are not exceeded.

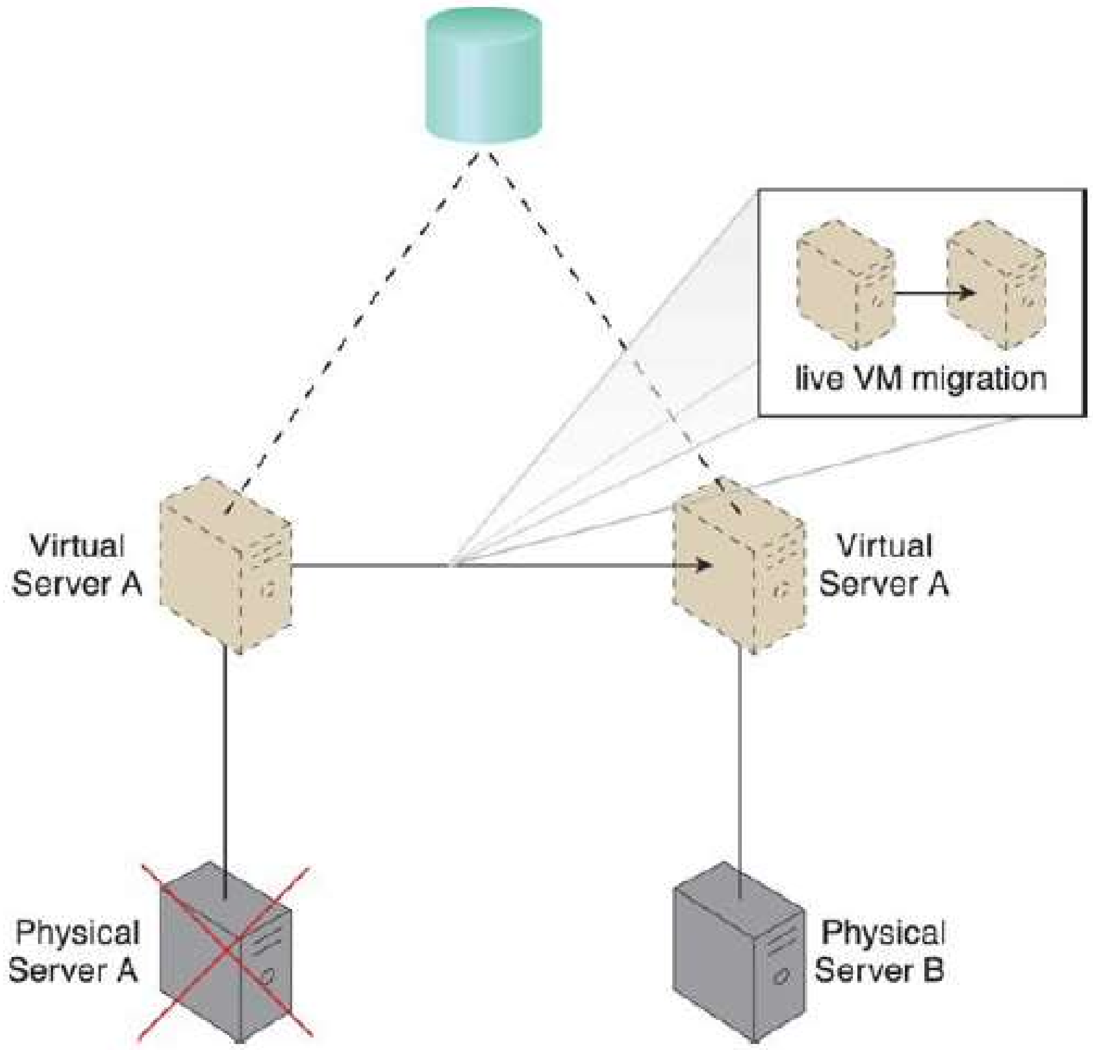


Figure 12.15. Physical Server A fails triggering the live VM migration program to dynamically move Virtual Server A to Physical Server B

- ***Hypervisor*** - The hypervisor of each affected physical server hosts the affected virtual servers.
- ***Logical Network Perimeter*** - Logical network perimeters provide and maintain the isolation that is required to ensure that each cloud consumer remains within its own logical boundary subsequent to virtual server relocation.
- ***Resource Cluster*** - The resource cluster mechanism is applied to create different types of active-active cluster groups that collaboratively improve the availability of virtual server-hosted IT resources.
- ***Resource Replication*** - This mechanism can create the new virtual server and cloud service instances upon primary virtual server failure.

Cloud Balancing Architecture

The ***cloud balancing architecture*** establishes a specialized architectural model in which IT resources can be load-balanced across multiple clouds.

- The cross-cloud balancing of cloud service consumer requests can help:
- improve the performance and scalability of IT resources
- increase the availability and reliability of IT resources
- improve load-balancing and IT resource optimization

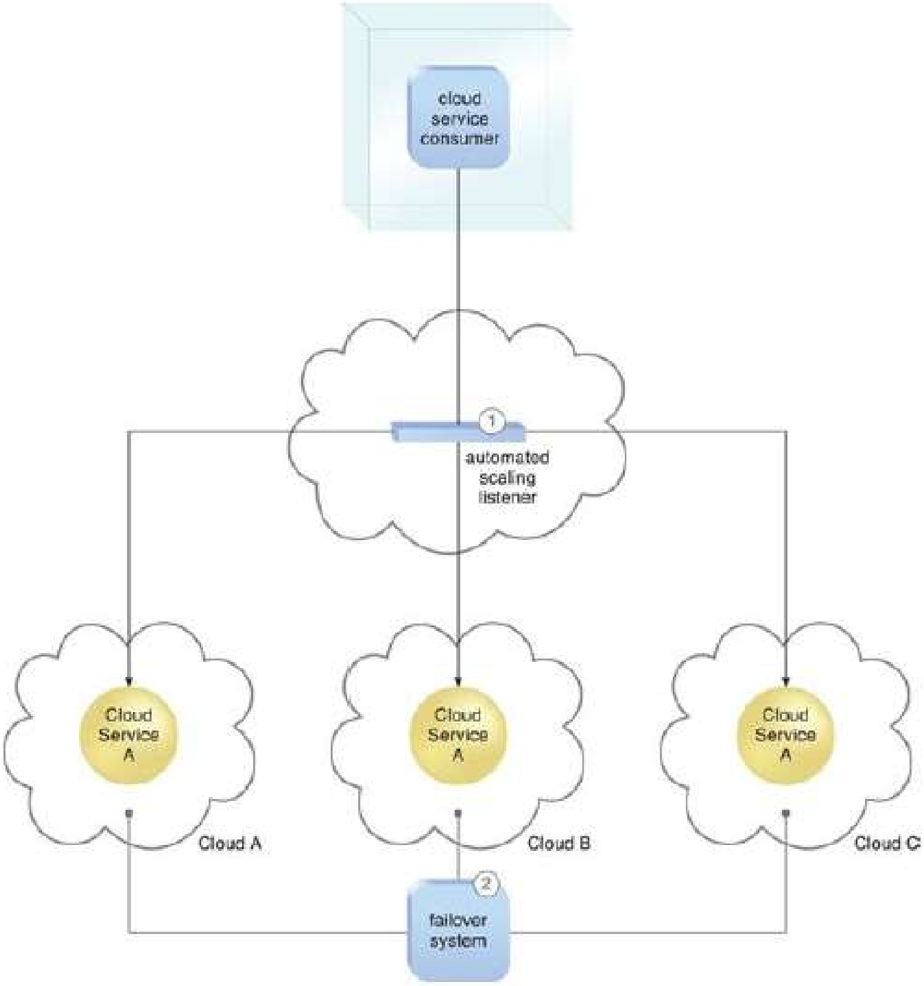
Cloud balancing functionality is primarily based on the combination of the automated scaling listener and failover system mechanisms (Figure 12.16). Many more components (and possibly other mechanisms) can be part of a complete cloud balancing architecture.

As a starting point, the two mechanisms are utilized as follows:

- The automated scaling listener redirects cloud service consumer requests to one of several redundant IT resource implementations, based on current scaling and performance requirements.
- The failover system ensures that redundant IT resources are capable of cross-cloud failover in the event of a failure within an IT resource or its underlying hosting environment. IT resource failures are announced so that the automated scaling listener can avoid inadvertently routing cloud service consumer requests to unavailable or unstable IT resources.

For a cloud balancing architecture to function effectively, the automated scaling listener needs to be aware of all redundant IT resource implementations within the scope of the cloud balanced architecture.

mplete cloud balancing architecture.



12.16. An automated scaling listener controls the cloud balancing process by routing cloud service consumer requests to redundant implementations of Cloud Service A distributed across multiple clouds (1). failover system instills resiliency within this architecture by providing cross-cloud failover (2).

Resource Reservation Architecture

Depending on how IT resources are designed for shared usage and depending on their available levels of capacity, concurrent access can lead to a runtime exception condition called ***resource constraint.*** A resource constraint is a condition that occurs when two or more cloud consumers have been allocated to share an IT resource that does not have the capacity to accommodate the total processing requirements of the cloud consumers. As a result, one or more of the cloud consumers encounter degraded performance or may be rejected altogether. The cloud service itself may go down, resulting in all cloud consumers being rejected.

Other types of runtime conflicts can occur when an IT resource (especially one not specifically designed to accommodate sharing) is concurrently accessed by different cloud service consumers. For example, nested and sibling resource pools introduce the notion of ***resource borrowing,*** whereby one pool can temporarily borrow IT resources from other pools. A runtime conflict can be triggered when the borrowed IT resource is not returned due to prolonged usage by the cloud service consumer that is borrowing it. This can inevitably lead back to the occurrence of resource constraints.

The ***resource reservation architecture*** establishes a system whereby one of the following is set aside exclusively for a given cloud consumer (Figures 12.17)

- single IT resource
- portion of an IT resource
- multiple IT resources

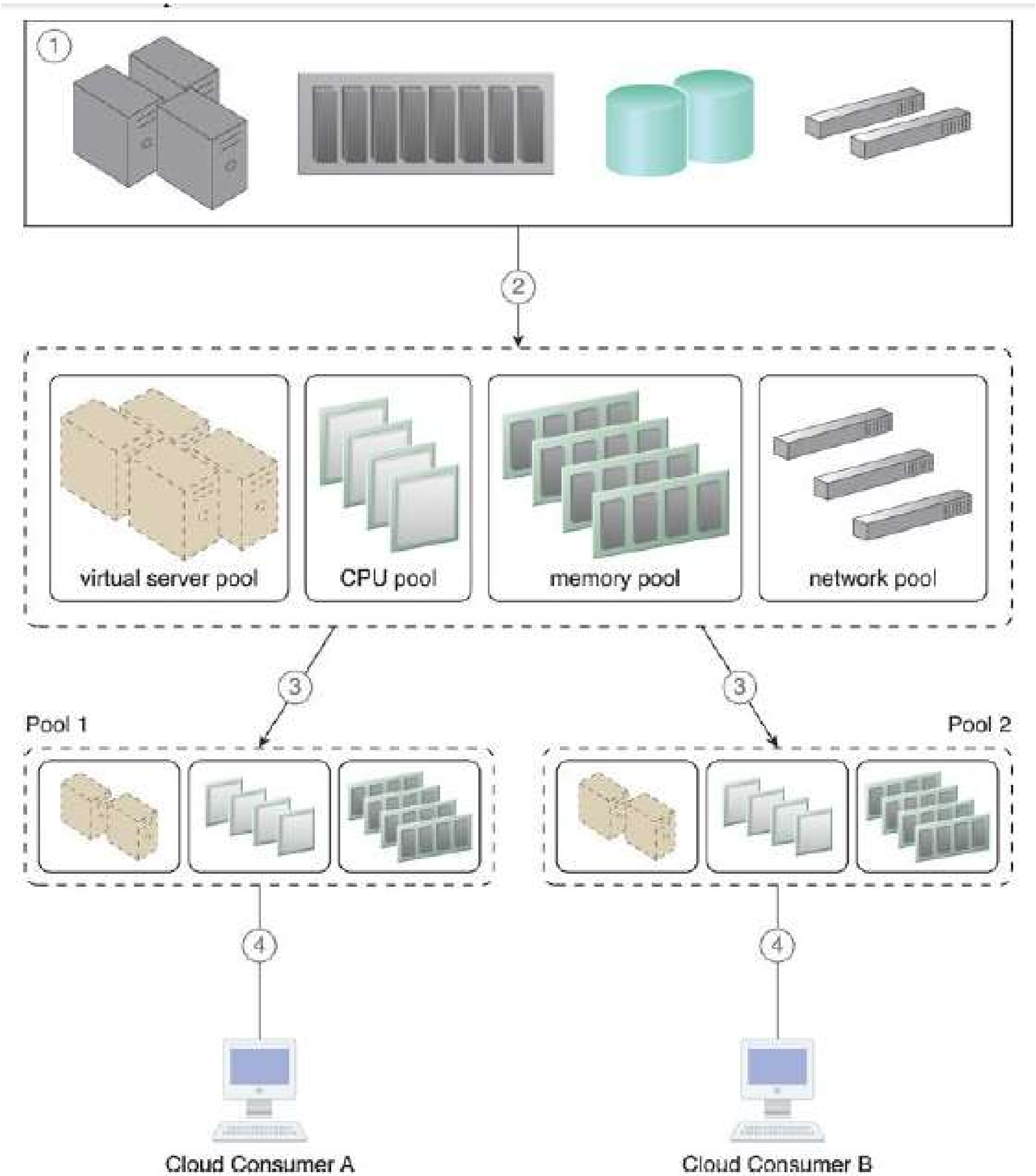


Figure 12.17. A physical resource group is created (1), from which a parent resource pool is created as per the resource pooling architecture (2). Two smaller child pools are created from the parent resource pool, and resource limits are defined using the resource management system (3). Cloud consumers are provided with access to their own exclusive resource pools (4)

This protects cloud consumers from each other by avoiding the aforementioned resource constraint and resource borrowing conditions.

The creation of an IT resource reservation system can require involving the resource management system mechanism, which is used to define the usage thresholds for individual IT resources and resource pools. Reservations lock the amount of IT resources that each pool needs to keep, with the balance of the pool's IT resources still available for sharing and borrowing. The remote administration system mechanism is also used to enable front-end customization, so that cloud consumers have administration controls for the management of their reserved IT resource allocations.

The types of mechanisms that are commonly reserved within this architecture are cloud storage devices and virtual servers. Other mechanisms that may be part of the architecture can include:

- ***Audit Monitor*** - The audit monitor is used to check whether the resource reservation system is complying with cloud consumer auditing, privacy, and other regulatory requirements. For example, it may track the geographical location of reserved IT resources.
- ***Cloud Usage Monitor*** - A cloud usage monitor may oversee the thresholds that trigger the allocation of reserved IT resources.
- ***Hypervisor*** - The hypervisor mechanism may apply reservations for different cloud consumers to ensure that they are correctly allocated to their guaranteed IT resources.
- ***Logical Network. Perimeter*** - This mechanism establishes the boundaries necessary to ensure that reserved IT resources are made exclusively available to cloud consumers.
- ***Resource Replication*** - This component needs to stay informed about each cloud consumer's limits for IT resource consumption, in order to replicate and provision new IT resource instances expediently.

Dynamic Failure Detection and Recovery Architecture

Cloud-based environments can be comprised of vast quantities of IT resources that are simultaneously accessed by numerous cloud consumers. Any of those IT resources can experience failure conditions that require more than manual intervention to resolve. Manually administering and solving IT resource failures is generally inefficient and impractical.

The ***dynamic failure detection and recovery architecture*** establishes a resilient watchdog system to monitor and respond to a wide range of pre-defined failure scenarios (Figures 12.20). This system notifies and escalates the failure conditions that it cannot automatically resolve itself. It relies on a specialized cloud usage monitor called the intelligent watchdog monitor to actively track IT resources and take pre-defined actions in response to predefined events.

defined events.

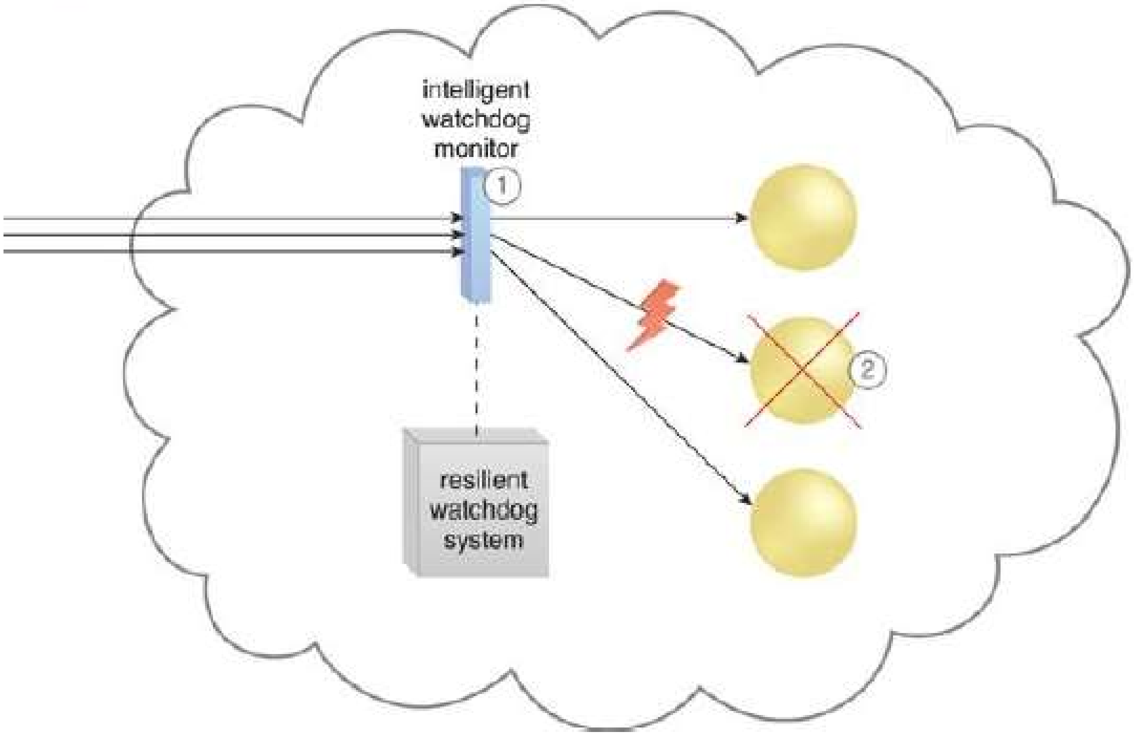


Figure 12.20. The intelligent watchdog monitor keeps track of cloud consumer requests (1) and detects that a cloud service has failed (2).

The resilient watchdog system performs the following five core functions:

- watching
- deciding upon an event
- acting upon an event
- reporting
- escalating

Sequential recovery policies can be defined for each IT resource to determine the steps that the intelligent watchdog monitor needs to take when a failure condition occurs.

Some of the actions the intelligent watchdog monitor commonly takes to escalate an issue include:

- running a batch file
- sending a console message
- sending a text message
- sending an email message
- sending an SNMP trap
- logging a ticket

There are varieties of programs and products that can act as intelligent watchdog monitors. Most can be integrated with standard ticketing and event management systems.

This architectural model can further incorporate the following mechanisms:

- ***Audit Monitor*** - This mechanism is used to track whether data recovery is carried out in compliance with legal or policy requirements.
- ***Failover System*** - The failover system mechanism is usually used during the initial attempts to recover failed IT resources.
- ***SLA Management System*** and ***SLA Monitor*** - Since the functionality achieved by applying this architecture is closely associated with SLA guarantees, the system commonly relies on the information that is managed and processed by these mechanisms.

Bare-Metal Provisioning Architecture

Remotely provisioning servers is common because remote management software is usually native to the operating system of most physical servers. However, access to conventional remote management programs is unavailable for ***bare- metal servers***—physical servers that do not have pre-installed operating systems or any other software.

Most contemporary physical servers provide the option of installing remote management support in the server's ROM. This is offered by some vendors through an expansion card while others have the components already integrated into the chipset. The ***bare-metal provisioning architecture*** establishes a system that utilizes this feature with specialized service agents, which are used to discover and effectively provision entire operating systems remotely.

The remote management software that is integrated with the server's ROM becomes available upon server start-up. A Web-based or proprietary user- interface, like the portal provided by the remote administration system, is usually used to connect to the physical server's native remote management interface.

The IP address of the remote management interface can be configured manually, through the default IP, or alternatively set through the configuration of a DHCP service. IP addresses in IaaS platforms can be forwarded directly to cloud consumers so that they can perform bare-metal operating system installations independently.

Although remote management software is used to enable connections to physical server consoles and deploy operating systems, there are two common concerns about its usage:

- Manual deployment on multiple servers can be vulnerable to inadvertent human and configuration errors.
- Remote management software can be time-intensive and require significant runtime IT resource processing.

The bare-metal provisioning system addresses these issues by using the following components:

- Discovery Agent - A type of monitoring agent that searches and finds available physical servers to be assigned to cloud consumers.
- ***Deployment Agent*** - A management agent that is installed into a physical server's memory, to be positioned as a client for the bare-metal provisioning deployment system.
- ***Discovery Section*** - A software component that scans the network and locates available physical servers with which to connect.

- ***Management Loader*** - The component that connects to the physical server and loads the management options for the cloud consumer.
- ***Deployment Component*** - The component responsible for installing the operating system on the selected physical servers.

The bare-metal provisioning system provides an auto-deployment feature that allows cloud consumers to connect to the deployment software and provision more than one server or operating system at the same time. The central deployment system connects to the servers via their management interfaces, and uses the same protocol to upload and operate as an agent in the physical server's RAM. The bare-metal server then becomes a raw client with a management agent installed, and the deployment software uploads the required setup files to deploy the operating system.

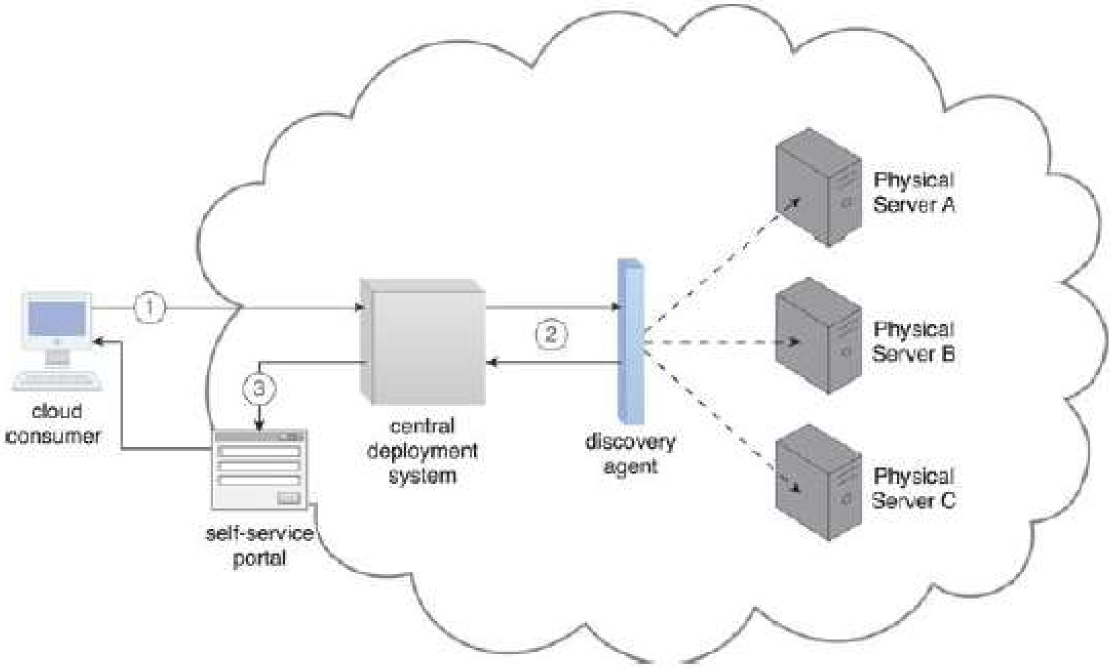


Figure 12.23 The cloud consumer connects to the deployment solution (1) to perform a search using the discovery agent (2). The available physical servers are shown to the cloud consumer (3).

Deployment images, operating system deployment automation, or unattended deployment and post installation configuration scripts can be used via the intelligent automation engine and self-service portal to extend this functionality.

The following additional mechanisms can be part of this architecture:

- ***Cloud Storage Device*** - This mechanism stores operating system templates and installation files, as well as deployment agents and deployment packages for the provisioning system.
- ***Hypervisor*** - The deployment of hypervisors on physical servers as part of the operating system provisioning can be required.
- ***Logical Network. Perimeter*** - Logical network perimeter boundaries help ensure that raw physical servers can only be accessed by authorized cloud consumers.

- ***Resource Replication*** - This mechanism is implemented for the replication of IT resources by deploying a new hypervisor on a physical server to balance the hypervisor workload during or after provisioning.
- ***SLA Management System*** - This management system ensures that the availability of physical bare-metal servers is in accordance with predefined SLA stipulations.

Rapid Provisioning Architecture

A conventional provisioning process can involve a number of tasks that are traditionally completed manually by administrators and technology experts that prepare the requested IT resources as per pre-packaged specifications or custom client requests. In cloud environments, where higher volumes of customers are serviced and where the average customer requests higher volumes of IT resources, manual provisioning processes are inadequate and can even lead to unreasonable risk due to human error and inefficient response times.

For example, a cloud consumer that requests the installation, configuration, and updating of twenty-five Windows servers with several applications requires that half of the applications be identical installations, while the other half be customized. Each operating system deployment can take up to 30 minutes, followed by additional time for security patches and operating system updates that require server rebooting. The applications finally need to be deployed and configured. Using a manual or semi-automated approach requires excessive amounts of time, and introduces a probability of human error that increases with each installation.

The ***rapid provisioning architecture*** establishes a system that automates the provisioning of a wide range of IT resources, either individually or as a collective. The underlying technology architecture for rapid IT resource provisioning can be sophisticated and complex, and relies on a system comprised of an automated provisioning program, rapid provisioning engine, and scripts and templates for on-demand provisioning.

Beyond the components displayed in Figure 12.25, many additional architectural artifacts are available to coordinate and automate the different aspects of IT resource provisioning, such as:

- ***Server* Templates** - Templates of virtual image files that are used to automate the instantiation of new virtual servers.
- **Server Images -** These images are similar to virtual server templates, but are used to provision physical servers.
- ***Application Packages*** - Collections of applications and other software that are packaged for automated deployment.
- ***Application Packager*** - The software used to create application packages.
- ***Custom Scripts*** - Scripts that automate administrative tasks, as part of an intelligent automation engine.
- ***Sequence Manager*** - A program that organizes sequences of automated provisioning tasks.
- ***Sequence Logger*** - A component that logs the execution of automated provisioning task sequences.

- ***Operating System Baseline*** - A configuration template that is applied after the operating system is installed, to quickly prepare it for usage.
- ***Application Configuration Baseline*** - A configuration template with the settings and environmental parameters that are needed to prepare new applications for use.
- ***Deployment Data Store*** - The repository that stores virtual images, templates, scripts, baseline configurations, and other related data.

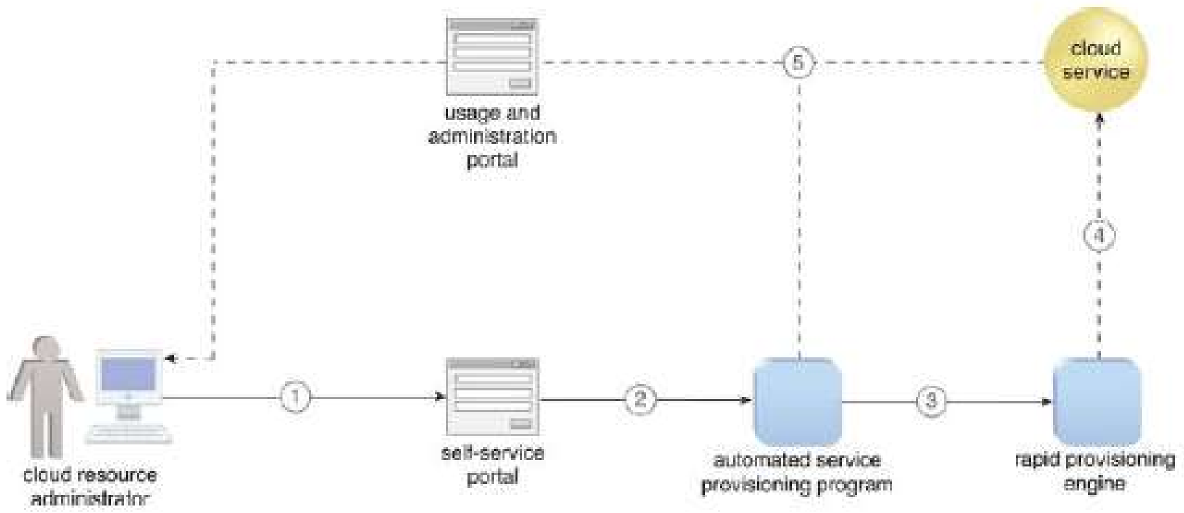

usage and administration portal
5
cloud service
4
1
2
3
cloud resource administrator
self-service portal
automated service provisioning program
rapid provisioning engine

Figure 12.25. A cloud resource administrator requests a new cloud service through the self-service portal (1). The self-service portal passes the request to the automated service provisioning program installed on the virtual server (2), which passes the necessary tasks to be performed to the rapid provisioning engine (3). The rapid provisioning engine announces when the new cloud service is ready (4). The automated service provisioning program finalizes and publishes the cloud service on the usage and administration portal for cloud consumer access (5).

The following step-by-step description helps provide some insight into the inner workings of a rapid provisioning engine, involving a number of the previously listed system components:

1. A cloud consumer requests a new server through the self-service portal.
2. The sequence manager forwards the request to the deployment engine for the preparation of an operating system.
3. The deployment engine uses the virtual server templates for provisioning if the request is for a virtual server. Otherwise, the deployment engine sends the request to provision a physical server.
4. The pre-defined image for the requested type of operating system is used for the provisioning of the operating system, if available. Otherwise, the regular deployment process is executed to install the operating system.
5. The deployment engine informs the sequence manager when the operating system is ready.
6. The sequence manager updates and sends the logs to the sequence logger for storage.
7. The sequence manager requests that the deployment engine apply the operating system baseline to the provisioned operating system.

8. The deployment engine applies the requested operating system baseline.
9. The deployment engine informs the sequence manager that the operating system baseline has been applied.
10. The sequence manager updates and sends the logs of completed steps to the sequence logger for storage.
11. The sequence manager requests that the deployment engine install the applications.
12. The deployment engine deploys the applications on the provisioned server.
13. The deployment engine informs the sequence manager that the applications have been installed.
14. The sequence manager updates and sends the logs of completed steps to the sequence logger for storage.
15. The sequence manager requests that the deployment engine apply the application's configuration baseline.
16. The deployment engine applies the configuration baseline.
17. The deployment engine informs the sequence manager that the configuration baseline has been applied.
18. The sequence manager updates and sends the logs of completed steps to the sequence logger for storage.

The cloud storage device mechanism is used to provide storage for application baseline information, templates, and scripts, while the hypervisor rapidly creates, deploys, and hosts the virtual servers that are either provisioned themselves, or host other provisioned IT resources. The resource replication mechanism is usually used to generate replicated instances of IT resources in response to rapid provisioning requirements.

Storage Workload Management Architecture

Over-utilized cloud storage devices increase the workload on the storage controller and can cause a range of performance challenges. Conversely, cloud storage devices that are under-utilized are wasteful due to lost processing and storage capacity potential (Figure 12.26)

The storage capacity system can keep the hosting storage device in power-saving mode for the periods when the LUNs are being accessed less frequently or only at specific times.

Some other mechanisms that can be included in the storage workload management architecture to accompany the cloud storage device are as follows:

- ***Audit Monitor*** - This monitoring mechanism is used to check for compliance with regulatory, privacy, and security requirements, since the system established by this architecture can physically relocate data.
- ***Automated Scaling Listener*** - The automated scaling listener is used to watch and respond to workload fluctuations.
- ***Cloud Usage Monitor*** - In addition to the capacity workload monitor, specialized cloud usage monitors are used to track LUN movements and collect workload distribution statistics.

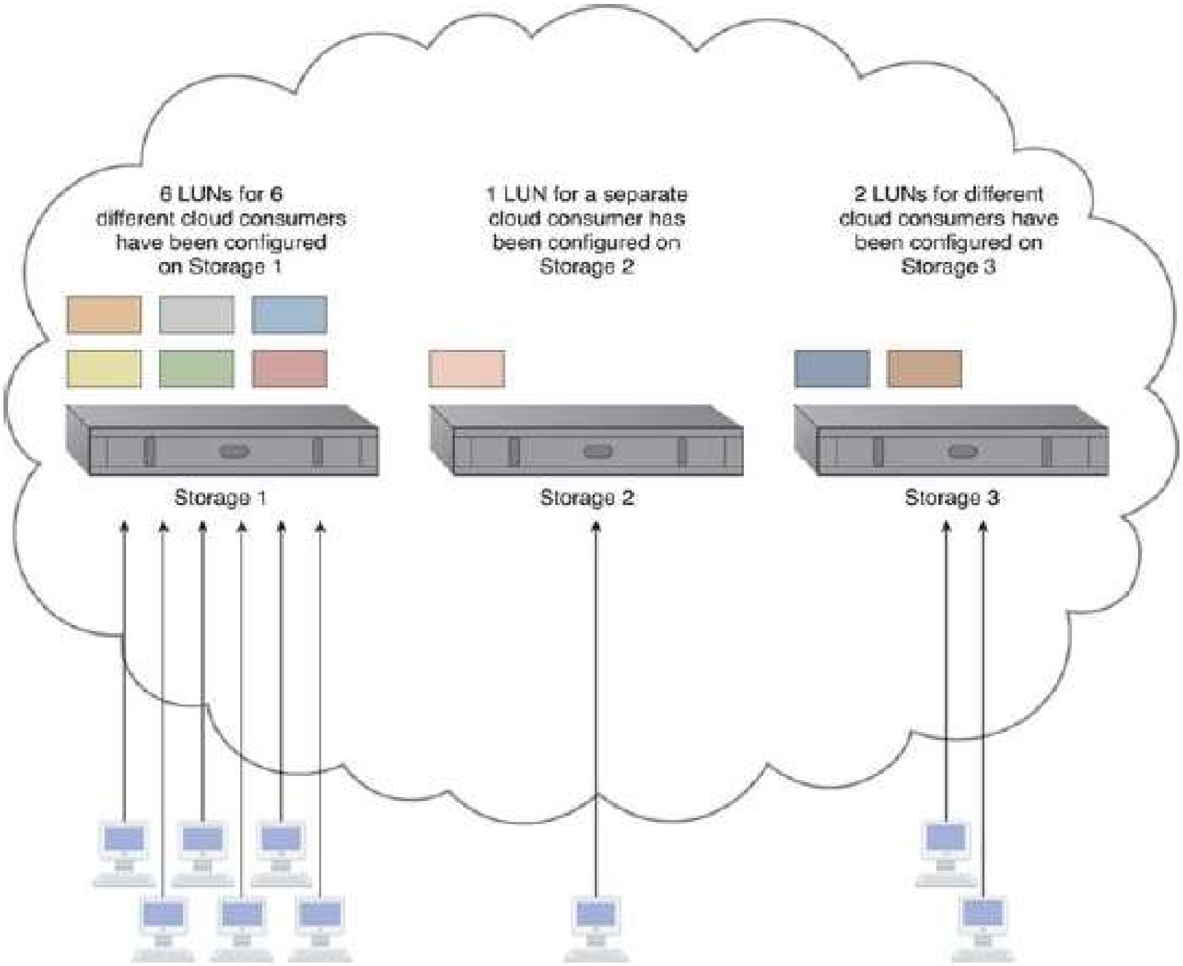

6 LUNs for 6 different cloud consumers have been configured on Storage 1
1 LUN for a separate cloud consumer has been configured on Storage 2
2 LUNs for different cloud consumers have been configured on Storage 3
Storage 1
Storage 2
Storage 3

Figure 12.26. An unbalanced cloud storage architecture has six storage LUNs in Storage 1 for cloud consumers to use, while Storage 2 is hosting one LUN and Storage 3 is hosting two. The majority of the workload ends up with Storage 1, since it is hosting the most LUNs.

- ***Load Balancer*** - This mechanism can be added to horizontally balance workloads across available cloud storage devices.
- ***Logical Network Perimeter*** - Logical network perimeters provide levels of isolation so that cloud consumer data that undergoes relocation remains inaccessible to unauthorized parties.