Cloud Computing

Last updated: 6/1/2023

The information provided by Sam, Cloud Computing ("we", "us", or "our") is for general information purpose only. All information in this template is provided in good faith, however we make no representation or warranty of any kind, express or implied, regarding the accuracy, adequacy, validity, reliability, availability or completeness of any information in the template. UNDER NO CIRCUMSTANCE SHALL WE HAVE ANY LIABILITY TO YOU FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS RESULT OF THE USE OF THE TEMPLATE OR RELIANCE ON ANY INFORMATION PROVIDED IN THE TEMPLATE. YOUR USE OF THE TEMPLATE AND YOUR RELIANCE ON THE DOCUMENT IS SOLELY AT YOUR OWN RISK.

This page is intentionally left blank.

Content

Principles and Paradigms

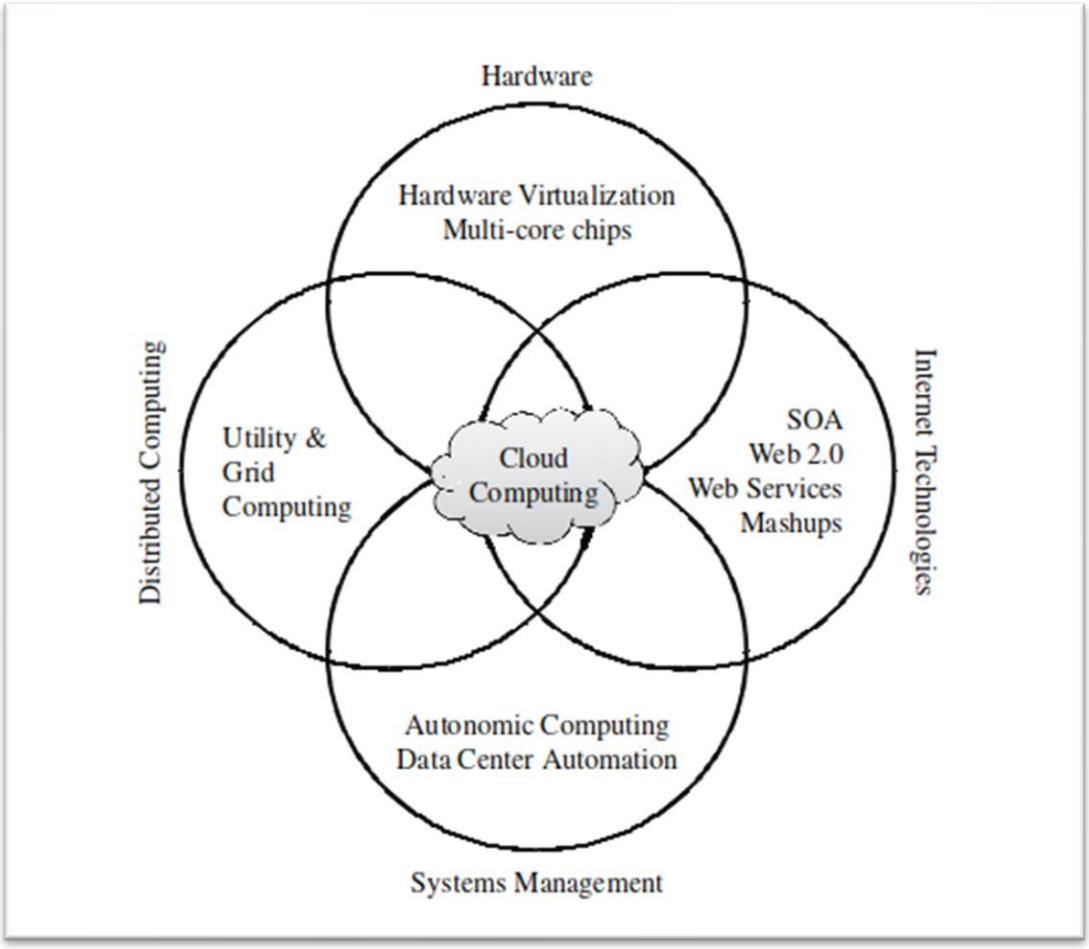
Introduction to Cloud Computing

Cloud Computing in a Nutshell

The concept of virtualization, where the user only sees the result without knowing the underlying processes, has been applied to computing through technologies like cluster, grid, and cloud computing. Cloud computing offers computing, storage, and software as a service, allowing businesses and individuals to access applications from anywhere in the world on demand. Many different definitions of cloud computing have been proposed, but the common characteristics are pay-per-use, elastic capacity, self-service interface, and virtualized resources. Cloud computing has been hailed as a significant switch in the IT world and is seen as the realization of delivering computing as a utility. This article surveys the main technological advancements that contributed to the advent of cloud computing and explains the different R&D efforts in cloud computing, with a focus on architectural aspects and innovative technical features.

Roots of Cloud Computing

Cloud computing has its roots in the convergence of several technological advancements, including hardware, Internet technologies, distributed computing, and systems management. These technologies have matured and been standardized, leading to the emergence of cloud computing. The shift towards utility-supplied computing resources delivered over the Internet is like the switch from in-house generated power to electric power grids.

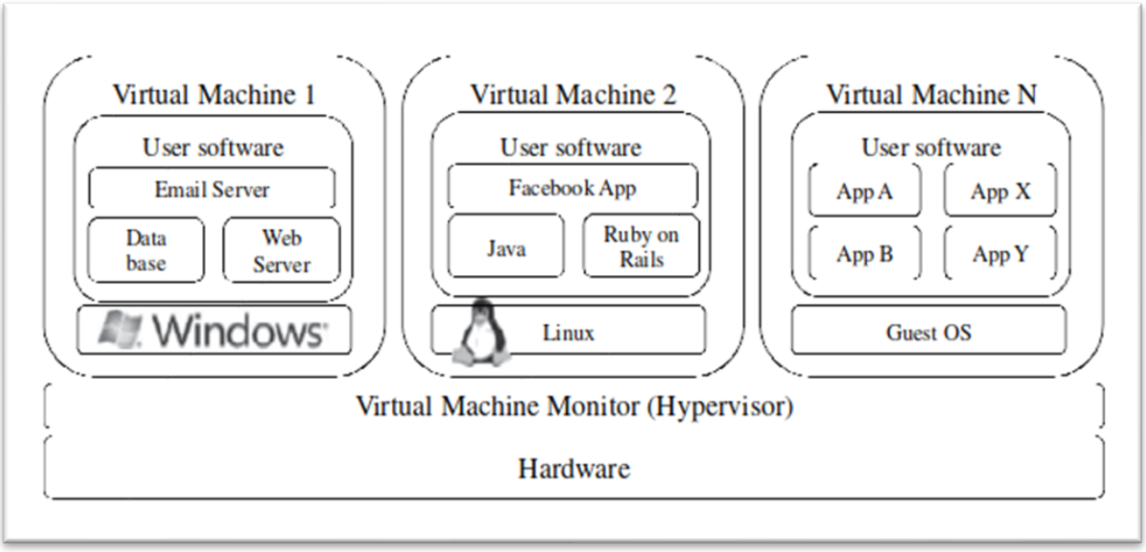


Convergence of various advances leading to the advent of cloud computing

Cloud computing benefits both consumers and providers of IT services by reducing costs and increasing efficiency. Providers achieve better operational costs by building hardware and software infrastructures to provide multiple solutions and serve many users. The potential of delivering computing services with the speed and reliability of local machines is possible due to new paradigms and the advent of increasingly fast fiber-optics networks. This allows providers to offer computing services for a fraction of what it costs for a typical company that generates its own computing power.

The emergence of Web services (WS) open standards has revolutionized the software integration domain. WS has enabled software resources to be packaged as "services," which are well-defined, self-contained modules that provide standard business functionality and are independent of the state or context of other services.

With the maturity of WS, powerful services can be accessed on-demand in a uniform way, enabling services to be composed to perform complex business logic.



A hardware virtualized server hosting three virtual machines

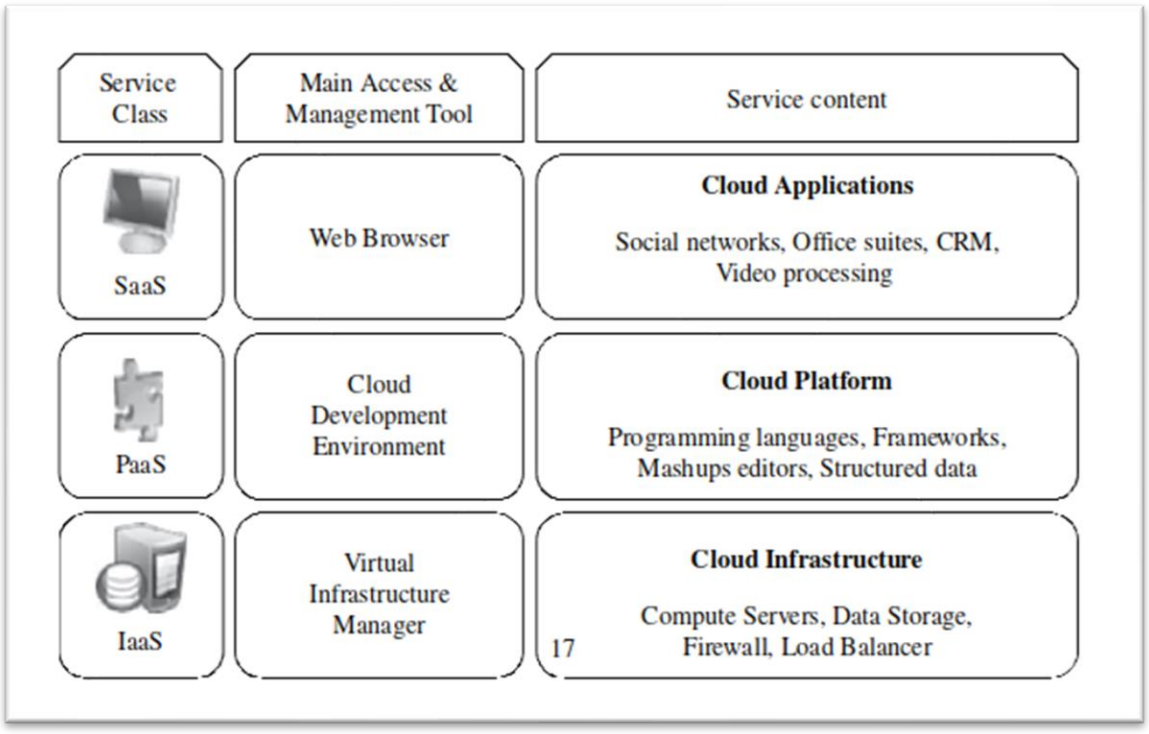
Consumer Web has also embraced the concept of gluing services, especially with the advent of Web 2.0. In the SaaS domain, cloud applications can be built as compositions of other services from the same or different providers. Many building blocks and solutions are now available in public marketplaces. Similarly, the emergence of Grid Computing has allowed aggregation of distributed resources and transparent access to them. The Open Grid Services Architecture (OGSA) has enabled the development of standardized protocols that allow distributed resources to be "discovered, accessed, allocated, monitored, accounted for, and billed for, etc., and in general managed as a single virtual system." The main challenge with grids is to ensure Quality of Service (QoS), especially for time-critical applications, as lack of performance isolation has prevented grid adoption in a variety of scenarios. Additionally, the availability of resources with diverse software configurations and the portability barrier has been a major challenge, and virtualization technology has been identified as the solution to these issues.

The section discusses two topics: virtual appliances and autonomic computing. A virtual appliance is a packaged software application and its environment, making it easier to customize and deploy. Online marketplaces such as VMWare and Amazon allow for the exchange of ready-made virtual appliances. However, interoperability issues can arise due to different hypervisors using different virtual machine (VM) image formats. To address this, the Open Virtualization Format (OVF) was created by several vendors, including VMware and Microsoft, to make packing and distribution of software on VMs more efficient and extensible.

Autonomic computing is a concept aimed at improving systems by decreasing human involvement in their operation. Autonomic systems rely on monitoring probes, an adaptation engine, and effectors to make changes on the system. IBM's Autonomic Computing Initiative has defined the four properties of autonomic systems: self-configuration, self-optimization, self-healing, and self-protection. The concepts of autonomic computing have inspired software technologies for data center automation, which can perform tasks such as managing service levels, data center capacity, proactive disaster recovery, and automation of VM provisioning.

Layers and Types of Clouds

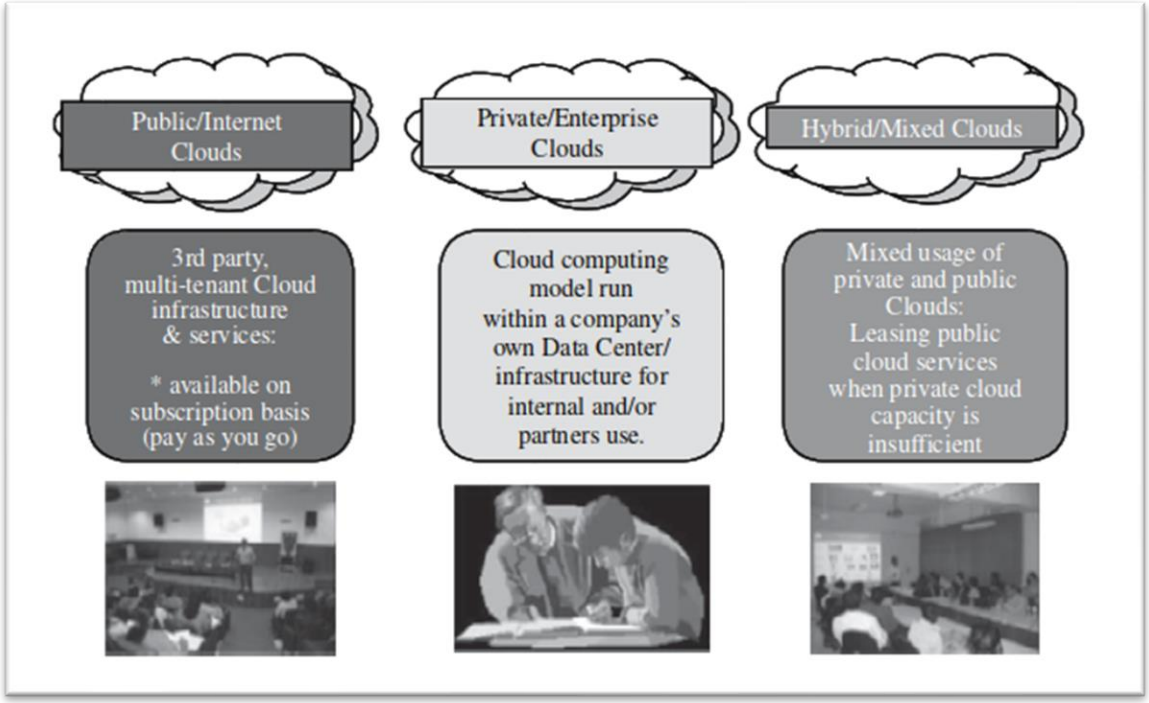
Cloud computing services can be classified into three types: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS).



The cloud computing stack

IaaS provides virtualized resources such as computation, storage, and communication on demand, while PaaS offers an environment for developers to create and deploy applications without needing to know how much computing power or memory they will use. SaaS allows end-users to access applications through web portals, eliminating the need for locally installed programs. Cloud computing can also be classified based on deployment models as public, private, community, or hybrid clouds.

Private clouds involve adding virtualization and cloud-like interfaces to an existing infrastructure, while community clouds are shared by several organizations with shared concerns.



Types of clouds based on deployment models

Hybrid clouds involve supplementing a private cloud with computing capacity from public clouds.

Desired Features of a Cloud

Cloud computing must offer self-service access, per-usage metering and billing, elasticity, and customization to meet the expectations of consumers. Self-service access allows customers to customize, pay for, and use services without the intervention of human operators. Per-usage metering and billing allows users to request and use only the necessary number of resources and pay for them on a short-term basis. Elasticity means providing computing resources in any quantity at any time, allowing for scaling up and down as needed. Customization is crucial, especially in a multi-tenant cloud, where resources must be highly customizable to meet different user needs.

Cloud Infrastructure Management

Managing physical and virtual resources, such as servers, storage, and networks, in a holistic fashion is a challenge faced by IaaS providers when building a cloud infrastructure. The software toolkit responsible for this orchestration is called a virtual infrastructure manager (VIM). VIMs present a set of basic features related to managing the life cycle of VMs, including networking groups of VMs together and setting up virtual disks for VMs. Some VI managers also provide advanced features, such as high availability. The availability of a remote cloud-like interface and the ability to manage many users and their permissions distinguish cloud toolkits from VIMs. Virtualization support, self-service, on-demand resource provisioning, multiple backend hypervisors, storage virtualization, interface to public clouds, and virtual networking are the features that are usually available in VIMs.

This section describes the main features of the most popular virtual infrastructure (VI) managers, with a detailed side-by-side comparison of each tool presented in Table below. The VI managers discussed in this section include Apache VCL, AppLogic, Citrix Essentials, Enomaly ECP, Eucalyptus, and Nimbus.

	License	Installation Platform of Controller	Client UI, API	Backend Hypervisor	Storage Virtualization	Dynamic Resource Allocation	Advance Reservation of Capacity
Apache VCL	Apache v2	Multi-platform (Apache/PHP)	Portal, XML-RPC	VMwarESX,ESXa Server	No	No	Yes
AppLogic	Proprietary	Linux	GUL, CLI	Xen	Global Volume	Yes	No
Citrix Essentials	Proprietary	Windows	GUI, CLI, Portal, XML-RPC	XenServer, Hyper-V	Citrix Storage Link	Yes	No
Enomaly ECP	GPL v3	Linux	Portal, WS	Xen	No	No	No
Eucalyptus	BSD	Linux	EC2 WS, CLI	Xen, KVM	No	Open Nebula	OpenNebul
OpenN Ebula	Apache v2	Linux	XML-RPC, CLI, Java	Xen, KVM	No	Yes	Yes (via Haizea)
OpenP EX	GPL v2	Multi-platform (Java)	Portal, WS	XenServer	No	No	Yes
oVirt	GPL v2	Fedora Linux	Portal	KVM	No	No	No
Platform ISF	Proprietary	Linux	Portal	Hyper-V XenServer, VMware ESX	No	Yes	Yes
Platform VMO	Proprietary	Linux indows	Portal	XenServer	No	Yes	No
VMaWare vSphere	Proprietary	Linux, Windows	CLI, GUI, Portal, WS	VMware ESX, ESXi	VMware vStorage VMFS	VMware DRM	No
VMWare vSphere	Proprietary	Linux, Windows	CLI, GUI, Portal, WS	VMware ESX, ESXi	VMware vCloud partners	No	Yes

Apache VCL provides customized desktop and high-performance computing environments in a cost-effective way with minimal IT staff intervention. It has a self-service web portal and supports the deployment of customized machine images on multiple computers, virtual networks, virtual clusters, and advance reservation of capacity.

AppLogic by 3tera Inc. is a commercial VI manager that manages clusters of virtualized servers and views entire applications as a collection of components. It offers dynamic appliances to add functionalities to applications as pluggable appliances.

Citrix Essentials is a hypervisor-agnostic VI management software that focuses on the management and automation of data centers. It offers several access interfaces, including GUI, CLI, web portal, and XML-RPC interfaces. It provides support for XenServer and Hyper-V hypervisors, Citrix Storage Link storage virtualization, virtual networks, dynamic resource allocation, and data protection with Citrix Consolidated Backup.

Enomaly ECP offers a web-based customer dashboard that allows users to fully control the life cycle of VMs. It allows providers and users to package and exchange applications.

Eucalyptus is an open-source framework that provides an EC2-compatible API and a storage cloud API for storing general user data and VM images. It has a Linux-based controller with administration web portal and supports Xen, KVM, and VMware backends, Amazon EBS-compatible virtual storage devices, and virtual networks.

Nimbus is built on top of the Globus framework, provides an EC2-compatible front-end API, support for Xen, and a backend interface to Amazon EC2. It distinguishes itself from others by providing a Globus Web Services Resource Framework (WSRF) interface and a backend service named Pilot.

Infrastructure as a Service Providers

IaaS providers offer a wide range of features and services, which can include not only virtual servers and storage but also networking, security, database management, and more. Some IaaS providers also offer platform as a service (PaaS) and software as a service (SaaS) option, allowing customers to build and deploy applications in the cloud without worrying about the underlying infrastructure.

One key benefit of IaaS is its scalability, which allows businesses to quickly and easily add or remove resources as needed to meet demand. Many IaaS providers also offer automatic scaling, load balancing, and other features to help customers optimize their resource usage and improve application performance.

	Geographic Presence	Client UI API Language Bindings	Primary Access to Server	Recreation of Capacity	Automated Horizontal Scaling	Runtime Server Resizing/Vertical Scaling
Amazon EC2	US East, Europe	CLI, WS, Portal	SSH (Linux), Remote Desktop (Windows)	Amazon reserved instances	Available with Amazon CloudWatch	No
Flexiscale	UK	Web Console	SSH	No	No	Processors, memory (requires reboot)
GoGrid	-	REST, Java, PHP, Python, Ruby	SSH	No	No	No
Joyent Cloud	US	-	SSH,VirtualMin	No	OpenSolaris	Both hardware (F5 networks) and software (Zeus)
Rackspace Cloud Servers	US	Portal, REST, Python, PHP, Java, C#/. NET	SSH	-	Linux	No

In terms of geographic presence, some IaaS providers have data centers located all around the world, while others have a more limited regional or national footprint. This can be an important consideration for businesses that need to comply with data sovereignty or other regulatory requirements.

Another important feature of IaaS providers is their service-level agreements (SLAs), which specify the level of availability, performance, and other metrics that customers can expect from the service. These SLAs often come with penalties or credits if the provider fails to meet the agreed-upon levels of service.

Platform as a Service Providers

Platform as a Service (PaaS) providers offer users a development and deployment environment where they can create and run their applications without concerning themselves with the low-level details of the platform. PaaS providers offer programming languages, software frameworks, and persistence options, such as Amazon SimpleDB and Google AppEngine datastore, to store user data.

	Target Use	Programming Language, Frameworks	Developer Tools	Programming Models	Automatic Scaling	Backend Infrastructure Providers
Aneka	.Net enterprise applications, HPC	.NET	Standalone SDK	Threads, Task, MapReduce	No	Amazon EC2
AppEngine	Web applications	Python, Java	Eclipse-based IDE	Request-based Web programming	Yes	Own data centers
Force.com	Enterprise applications (esp. CRM)	Apex	Eclipse-based IDE, Web-based wizard	Workflow,Excel-like formula language, Request-based web programming	Unclear	Own data centers
Microsoft Windows Azure	Enterprise and Web applications	.NET	Azure tools for Microsoft Visual Studio	Unrestricted	Yes	Own data centers
Heroku	Web applications	Ruby on Rails	Command-line tools	Request based web programming	Yes	Amazon EC2
Amazon Elastic MapReduce	Data processing	Hive and Pig, Cascading, Java, Ruby, Perl, Python, PHP, R, C++	Karmasphere Studio for Hadoop (Net-Beans-based)	MapReduce	No	Amazon EC2

Additionally, PaaS providers usually support multiple programming languages, with popular languages including Python and Java, .NET languages, and Ruby. PaaS providers offer different types of models to solve different problems. Aneka, App Engine, Microsoft Azure, and Force.com are some examples of PaaS providers that offer different services to developers. For example, Aneka is a .NET-based service-oriented resource management and development platform that supports several programming models to enable execution of legacy HPC applications and MapReduce.

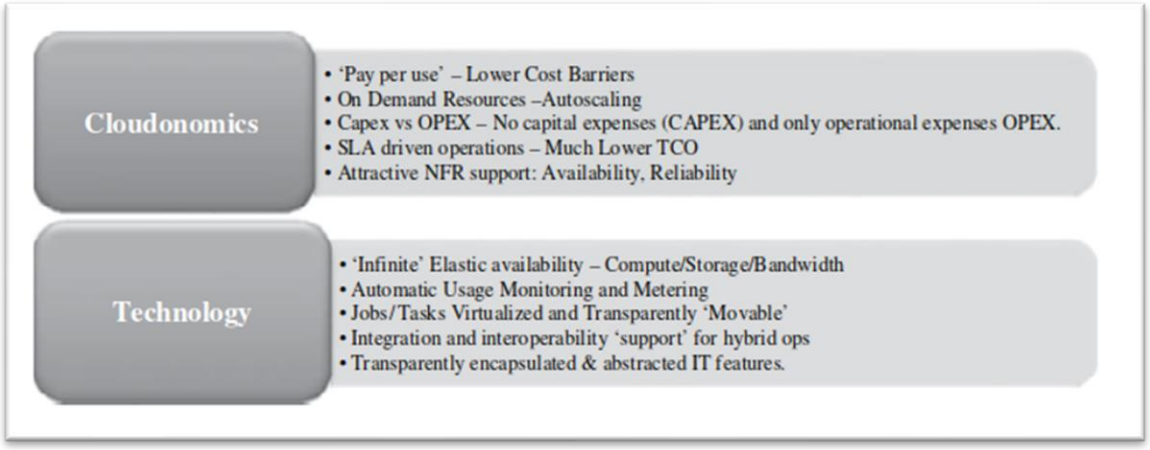
Challenges and Risks

The cloud computing paradigm offers many benefits, but there are also many challenges and risks to be considered. These include security, privacy, data lock-in, standardization, availability, fault tolerance, disaster recovery, resource management, and energy efficiency. Security and privacy are critical issues because the use of third-party services and infrastructures increases the likelihood of attacks. Legal and regulatory issues are also a concern, as different countries have different laws for data management.

Data lock-in is a significant issue because cloud infrastructures and platforms lack standard methods for storing user data and applications, leading to difficulties in interoperability and portability. To address these concerns, standardization efforts, such as the Cloud Computing Interoperability Forum and the Open Virtual Format, are underway. Providers must also ensure availability, performance, and disaster recovery through service level agreements with customers. Resource management is another challenge that providers must tackle, including efficient virtualized resource pool management, dynamic virtual machine mapping policies, and the management of large amounts of data. Energy efficiency is another important issue due to the high electricity consumption of data centers.

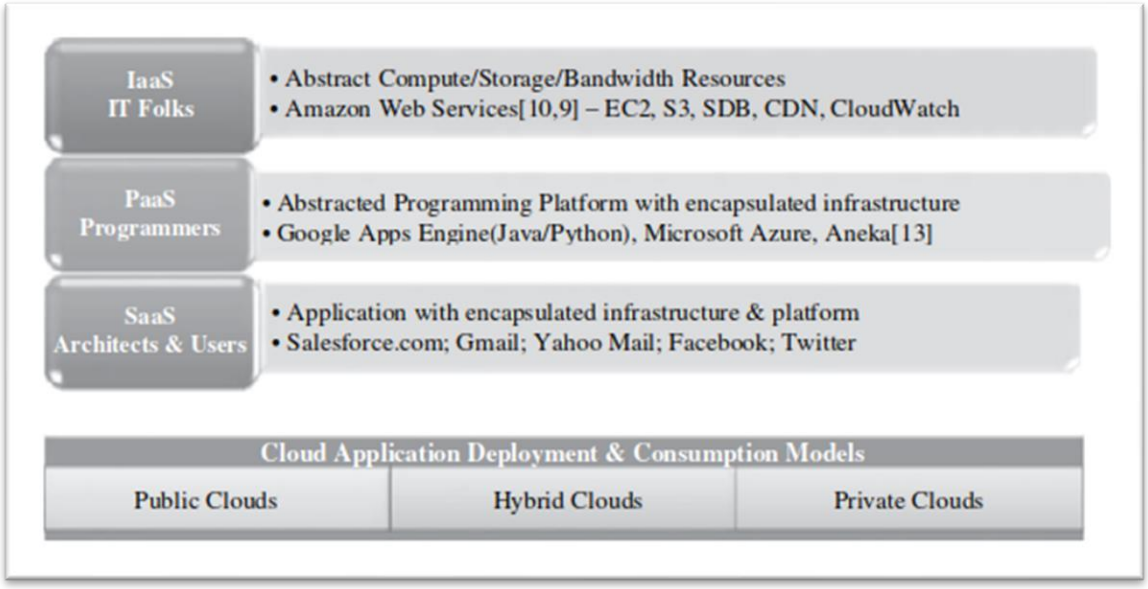
Migrating into a Cloud

Cloud computing has been a disruptive model of IT which raises the expectations of small and medium enterprises. This model is part technology and part business model, and its innovation has been described as a "disruptive techno-commercial model" of IT.



The promise of the cloud computing services

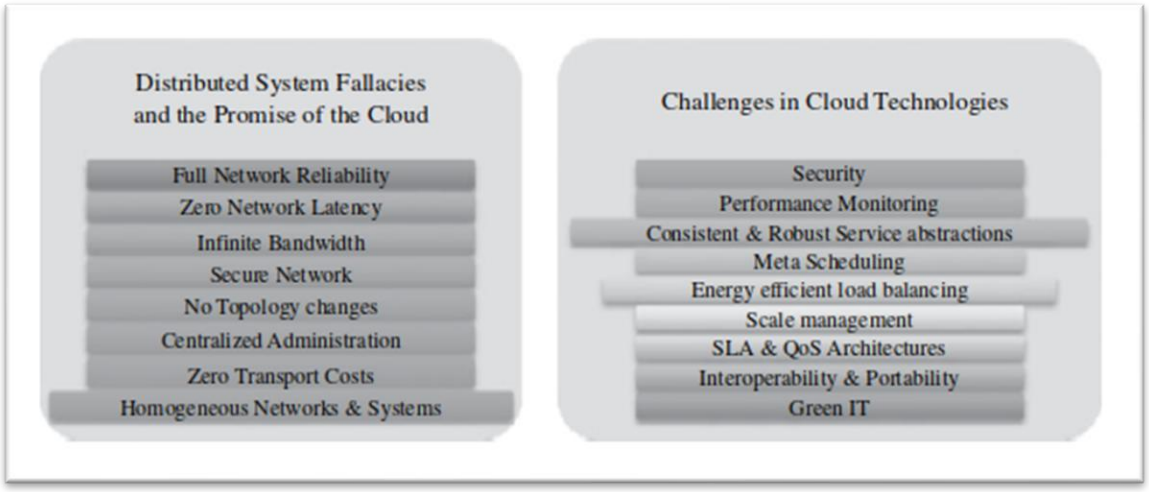
There are several debates on when and how to migrate an application into the cloud, what part of the IT application to migrate, what kind of customers benefit from the migration, and more. The chapter also shares a Seven-Step Model of Migration into the Cloud.



The cloud computing service offering and deployment models

Cloud computing services have been offered on a trial basis, which have demonstrated very attractive pricing models. The initial cloud computing offerings were premature, and the cloud computing service vendors were

grappling with real issues of distributed systems and business models. There have been several efforts in the recent past to define the term "*cloud computing*," but many have not been able to provide a comprehensive one. Most large or small enterprises today are powered by captive data centers. Cloud computing turned attractive because of its ability to pass on the additional demand from the IT setups onto the cloud while paying only for the usage and being unencumbered by the load of operations and management. Small and medium enterprises have yielded substantial and significant economic savings by leveraging cloud computing usage for all additional cyclical IT needs.



Under the hood' challenges of the cloud computing services implementations

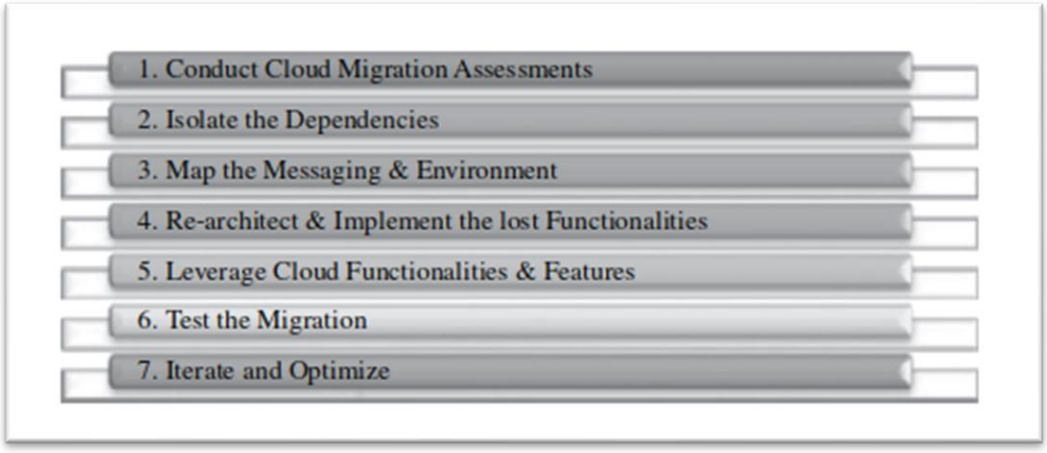
The economics and the associated trade-offs, of leveraging the cloud computing services, now popularly called "*cloudonomics*," for satisfying enterprise's seasonal IT loads has become a topic of deep interest amongst IT managers and technology architects.

Broad Approaches to Migrating into the Cloud

Migrating an enterprise application into the cloud requires a thorough understanding of the economic, business, and technological reasons for migration. Cloud computing offers many economic benefits such as eliminating all capital expenses and only incurring operational expenses, which can reduce the total cost of ownership (TCO) as compared to running a private data center. However, the economic feasibility of migration must be evaluated in terms of short-term and long-term costs. Migration can occur at one of five levels: application, code, design, architecture, and usage. Migration can be driven by economic reasons of cost-cutting for IT capital and operational expenses. Other factors such as licensing, service level agreement (SLA) compliances, and pricing of cloud services can also impact the feasibility of migration. The migration industry thrives on custom and proprietary best practices, which can be specialized at the level of an enterprise application's components, such as migrating application servers or databases. Ultimately, if the costs of using an enterprise application on a cloud are significantly lower than those incurred in a captive data center and the cost of migration does not add to the burden on return on investment (ROI), migration is tenable.

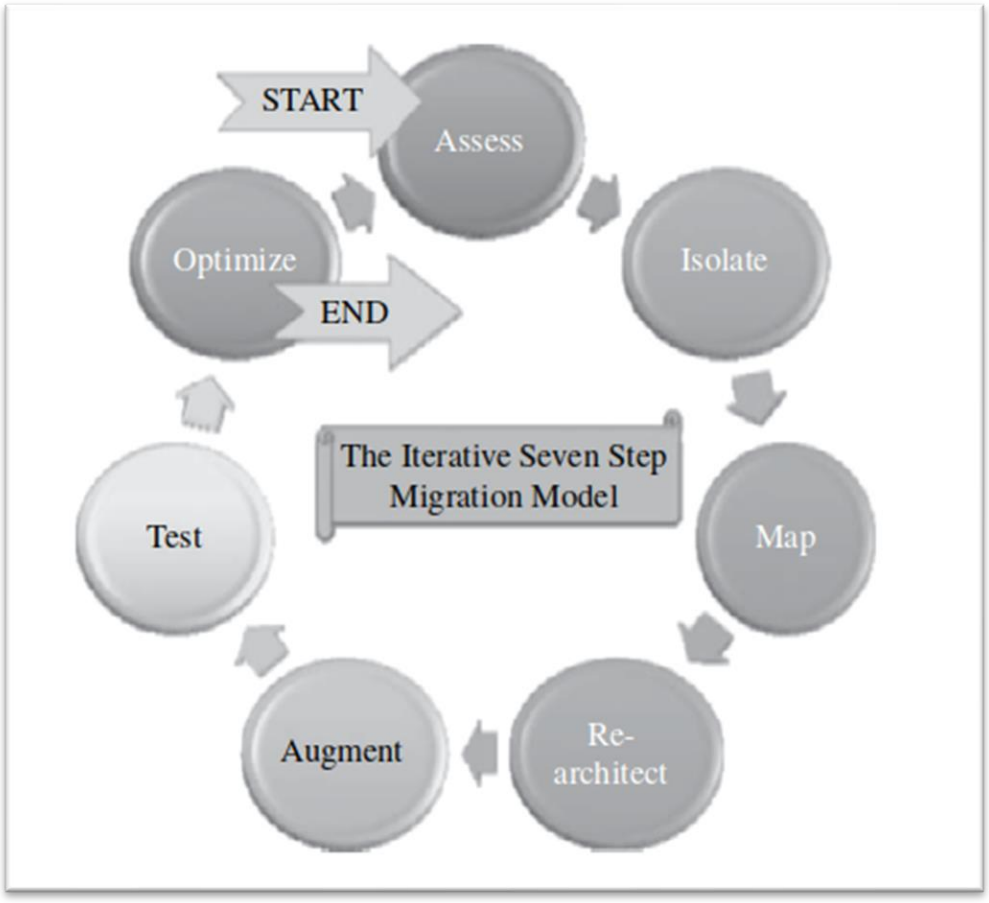
The Seven-Step Model of Migration into a Cloud

Migration into the cloud is usually done in phases, and a structured approach has advantages, as it captures the best practices of various migration projects. However, not many efforts have been made to consolidate this subject, even though it is both a top revenue earner and a long-standing customer pain. The Seven-Step Model of Migration into the Cloud is introduced as part of an effort to understand and leverage cloud computing service offerings in the enterprise context.



The Seven-Step Model of Migration into the Cloud

The seven steps are assessment, isolation, mapping, augmentation, validation, optimization, and migration. Migration assessments are done to understand the issues involved at the application level, code, design, architecture, usage levels, tools being used, test cases, configurations, functionalities, and non-functional requirements. The first process step is at the assessment level, followed by isolating systemic and environmental dependencies of enterprise application components within the captive data center.



The iterative Seven-step Model of Migration into the Cloud

The Seven-Step Model of Migration into the Cloud is more generic, versatile, and comprehensive than the typical approach to migration into Amazon AWS, which involves six phases. Identifying and mitigating migration risks is the biggest challenge for any cloud migration project, and they fall under two categories: general migration risks and security-related migration risks.

Enriching the 'Integration as a Service' Paradigm

Cloud computing has brought a lot of advancements to the IT and business domains by enabling faster realization and proliferation of dynamic, converged, adaptive, on-demand, and online compute infrastructures, which are the key requirements for the future IT. With the shift of most IT offerings such as business services and applications to clouds, cloud movement is picking up fast. As a result, there is a new dimension to the integration scenario as enterprise data and applications are now linked up with cloud applications, and integration middleware is taken to clouds to simplify and streamline enterprise-to-enterprise (E2E), enterprise-to-cloud (E2C), and cloud-to-cloud (C2C) integration. This chapter discusses the impact of the cloud paradigm on integration and how cloud-based middleware can be used to simplify integration goals. IT is tending towards becoming a significant factor and the facilitator of every aspect of human lives, and novel computing paradigms such as cloud, grid, and on-demand computing are evolving relentlessly to be impactful and insightful. The newest buzzword today is ambient intelligence (AmI), where every device will seamlessly and spontaneously coexist to deliver users' needs at the right time and place.

The issues surrounding the integration of Software as a Service (SaaS) applications have led to the investigation of various approaches for their resolution. Private cloud, hybrid, and the more recent community cloud have been identified as potential solutions for the inefficiencies and deficiencies observed in this area. However, as pointed out by experts in the field, there is still a long way to go in addressing these challenges.

Companies such as Boomi have been focusing on this issue and have published well-written white papers elaborating on the issues confronting enterprises seeking to embrace third-party public clouds for hosting their services and applications. One of the main challenges specific to integration is that most SaaS applications are point solutions that service one line of business. Consequently, companies without a method of synchronizing data between multiple lines of businesses are at a serious disadvantage in terms of maintaining accurate data, forecasting, and automating key business processes. Many SaaS providers have responded to the integration challenge by developing APIs. However, accessing and managing data via an API requires a significant amount of coding and maintenance due to frequent API modifications and updates. Furthermore, there is little standardization or consensus on the structure or format of SaaS APIs, which means that the IT department must expend an excess amount of time and resources developing and maintaining a unique method of communication for each SaaS application deployed within the organization.

Data transmission security is another issue that needs to be addressed by the integration solution of choice. While SaaS providers ensure that customer data is secure within the hosted environment, the need to transfer data from on-premise systems or applications behind the firewall with SaaS applications hosted outside of the client's data center poses new challenges. The integration solution must be able to synchronize data bi-directionally from SaaS to on-premise without opening the firewall. Best-of-breed integration providers can offer the ability to do so by utilizing the same security as when a user is manually typing data into a web browser behind the firewall.

To achieve the promised value for businesses and users, the minimum requirement for any relocated application is the interoperability between SaaS applications and on-premise enterprise packages. However, as SaaS applications were not initially designed with interoperability in mind, the integration process has become a more challenging assignment. Other obstructions and barriers come into play when routing messages between on-demand applications and on-premise resources. Message, data, and protocol translations must happen at end-points or at the middleware layer to facilitate sharing and collaboration among participants. As applications and data are diverse, distributed, and decentralized, versatile integration technologies and methods are essential to smooth the integration process.

On the infrastructural front, the arrival of clouds onto the scene has extended the horizon and the boundary of business applications, events, and data. Applications and services are increasingly being readied and relocated to highly scalable and available clouds for business, technical, financial, and green reasons. This immediate impact highlights the need for adaptive integration engines that seamlessly connect enterprise applications with cloud applications. Integration is being stretched further to the level of the expanding internet, which poses a litmus test for system architects and integrators.

The migration of the functionality of a typical enterprise application integration (EAI) hub/enterprise service bus (ESB) into the cloud is the essence of Integration as a Service (IaaS). This approach enables the smooth transport of data between any enterprise and SaaS applications. Users subscribe to IaaS as they would any other SaaS application, and cloud middleware is the logical evolution of traditional middleware solutions, made available as a service.

SaaS integration has become a widely discussed topic in the academic world, and for good reason. Integration technology brings order to the chaos of heterogeneous systems, networks, and services, thereby simplifying operations and increasing efficiency. However, integrating SaaS applications presents unique challenges. Take for instance, the case of a mid-sized paper company that recently became a Salesforce.com CRM customer. While the company's on-premise custom system that uses an Oracle database to track inventory and sales provides significant value, the information within the Salesforce.com system is somewhat redundant with the data stored in the legacy system. This leads to inefficiencies and data quality issues, as information needs to be entered and maintained in two different locations.

To address this problem, data synchronization technology is proposed as the best fit between the Salesforce.com source and the existing Oracle-based legacy system. The technology provides automatic mediation of differences between the two systems, including application semantics, security, interfaces, protocols, and native data formats. This results in complete and compact synchronization of information between the SaaS-delivered and legacy systems, effectively eliminating data quality and integrity issues. However, SaaS applications' dynamic nature, coupled with the sheer amount of information that needs to move between SaaS and on-premise systems, presents further integration challenges. Cloud integration is more complex than local integration, as access to cloud resources is more limited. While embedding integration points in local and custom applications is easy, custom applications must be designed to support integration in the cloud. Enterprises putting their applications in the cloud or subscribers of cloud-based business services are dependent on the vendor to provide integration hooks and APIs.

Cloud resources are virtualized and service-oriented, which means that everything is expressed and exposed as a service. Due to this dynamism, application versioning and infrastructural changes are liable for dynamic changes, which impact the integration model. The tightly coupled integration model fails and falters at the cloud, making it clear that the low-level interfaces ought to follow the Representational State Transfer (REST) route, which is a simple architectural style that subscribes to the standard methods of the Http protocol.

Cloud integration performance is bound to slow down due to the latency aggravation caused by the network distances between elements in the cloud. While clouds support application scalability and resource elasticity, the round-trip latency is an issue that cannot be overlooked.

Integrating SaaS applications requires understanding the special traits and tenets of these applications, as well as the obstacles imposed by clouds, and prescribing proven solutions. The limited access to cloud resources, dynamic resources, and cloud integration performance are some of the challenges that need to be addressed to achieve seamless SaaS integration.

The emergence of cloud computing has given rise to new integration scenarios that require complex and sophisticated solutions. Prior to the cloud model, local systems were interconnected manually, but with the shift to cloud computing, local applications must be connected to the cloud, and cloud applications must be connected to each other, creating a complex integration channel matrix.

Within a Public Cloud

Even though not everything will move to the cloud model all at once, even the simplest scenarios require some form of local/remote integration. Furthermore, certain applications may never leave the building due to regulatory constraints, such as HIPAA, GLBA, and general security issues. This implies that integration must crisscross firewalls somewhere. Three major integration scenarios have been identified: Within a Public Cloud, Homogeneous Clouds, and Heterogeneous Clouds. Within a Public Cloud, two different applications hosted in a cloud must be connected by cloud integration middleware, such as a cloud-based ESB or internet service bus (ISB). These applications can be owned by two different companies or live in a single physical server but run on different virtual machines.

In Homogeneous Clouds, the applications to be integrated are in two geographically separated cloud infrastructures. The integration middleware can be in either cloud 1 or cloud 2, or in a separate cloud. Data and protocol transformation are required, which are handled by the ISB, and the approach is like the enterprise application integration procedure.

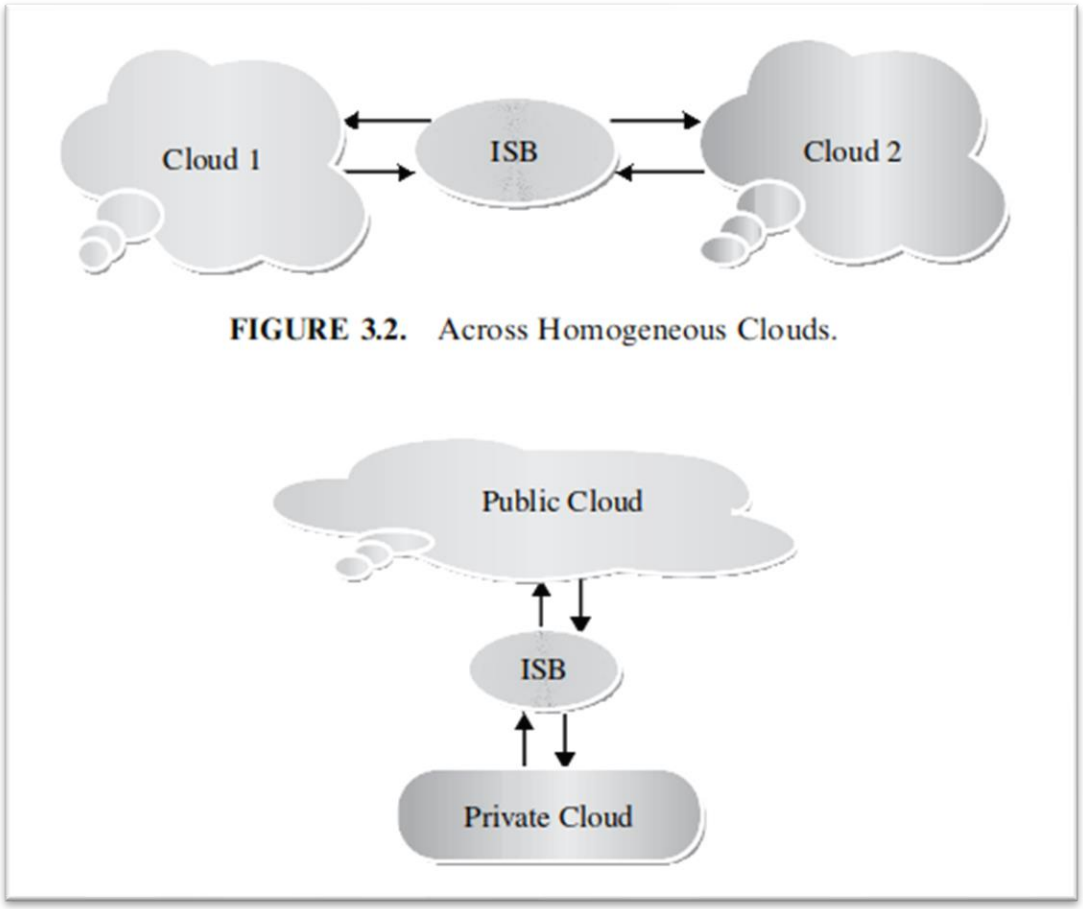


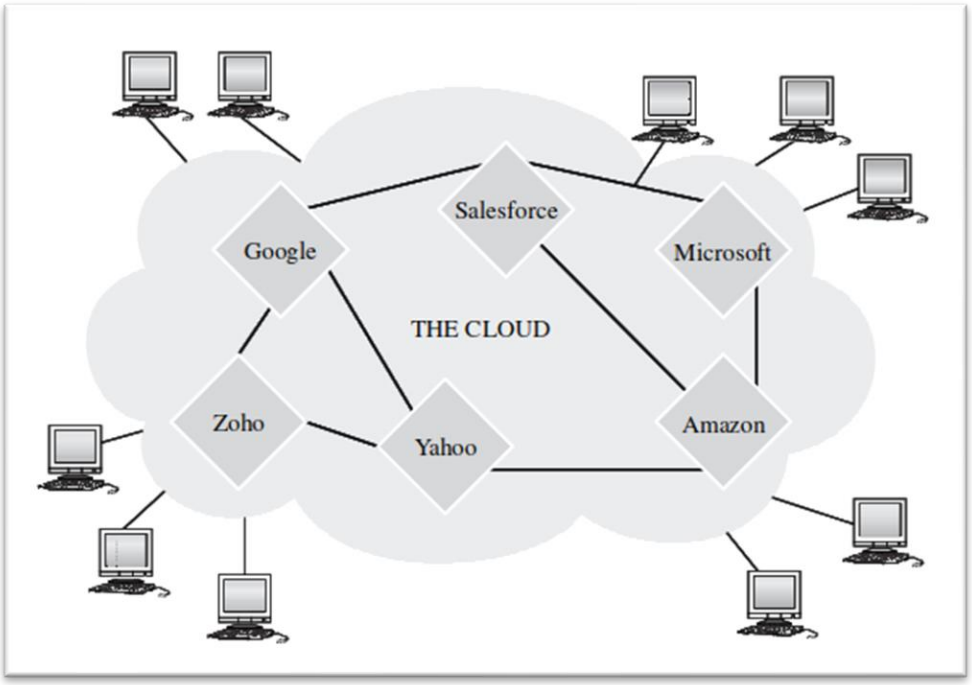
FIGURE 3.2. Across Homogeneous Clouds.

Across Heterogeneous Clouds

In Heterogeneous Clouds, one application is in the public cloud, and the other application is in the private cloud. This is currently the dominating scenario for cloud integration. Businesses subscribe to popular on-demand enterprise packages from established providers such as Salesforce.com, NetSuite, and Ramco Systems' customer relationship management (CRM) and enterprise resource planning (ERP). The first two scenarios will become prevalent once there are several commercial clouds, and cloud services become pervasive. Service integration and composition domains will then become an important and incredible factor for global computing.

Cloud integration has become a popular trend in recent years, allowing businesses to connect their applications and systems with cloud services. There are several methodologies for achieving cloud integration, excluding custom integration through hand-coding. In this regard, there are three main types of cloud integration:

1. Traditional enterprise integration tools can be enhanced with special connectors to access cloud-located applications. This approach is suitable for IT organizations that have already invested in integration suites for their application integration needs. Special drivers, connectors, and adapters are being developed and integrated into existing integration platforms to enable bidirectional connectivity with cloud services. For optimal performance, integration appliances are also available in the market.
2. Traditional enterprise integration tools can be hosted in the cloud. This approach is like the first option, but with the integration software suite hosted on a third-party cloud infrastructure. This eliminates the need for businesses to worry about procuring and managing hardware or installing integration software. It is a good fit for IT organizations that outsource integration projects to IT service providers and systems integrators who have the skills and resources to deliver integrated systems. This approach is particularly useful for cloud-to-cloud (C2C) integration, although it requires a secure VPN tunnel to access on-premise corporate data.
3. Integration-as-a-Service (IaaS) or On-Demand Integration Offerings are SaaS applications designed to deliver integration services securely over the internet. They can integrate cloud applications with on-premise systems and even integrate on-premise systems with other on-premise applications. This approach is a good fit for companies that prioritize ease of use, ease of maintenance, time to deployment, and have a tight budget. It is especially appealing to small and medium-sized enterprises, as well as large enterprises with departmental application deployments. This approach is well-suited for companies that plan to use their SaaS administrator or business analyst as the primary resource for managing and maintaining their integration work.



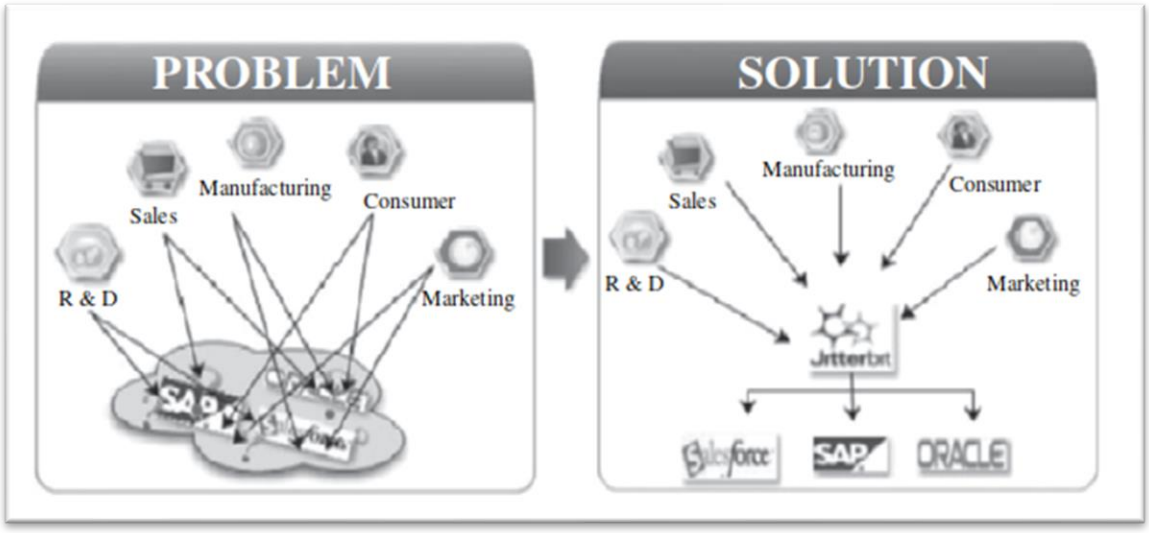
The Smooth and Spontaneous Cloud Interaction via Open Clouds

The integration requirements can be realized using any of the following methods and middleware products:

1. Hosted and extended ESB (Internet service bus / cloud integration bus).
2. Online message queues, brokers, and hubs.
3. Wizard and configuration-based integration platforms (niche integration solutions).
4. Integration service portfolio approach.
5. Appliance-based integration (standalone or hosted).

To ensure that integration platforms and backbones meet business needs, they should possess specific key attributes, including connectivity, semantic mediation, data mediation, data migration, data security, data integrity, and governance. These attributes are critical for a lean data integration lifecycle.

Connectivity refers to the ability of the integration engine to engage with both the source and target systems using available native interfaces. Semantic mediation accounts for the differences between application semantics between two or more systems, while data mediation converts data from a source data format into a destination data format. Data migration involves transferring data between storage types, formats, or systems. Data security ensures that information extracted from the source systems is securely placed into target systems. Data integrity guarantees that data is complete and consistent. Finally, governance is about managing changes to core information resources, including data semantics, structure, and interfaces.



Linkage of On-Premise with Online and On-Demand Applications

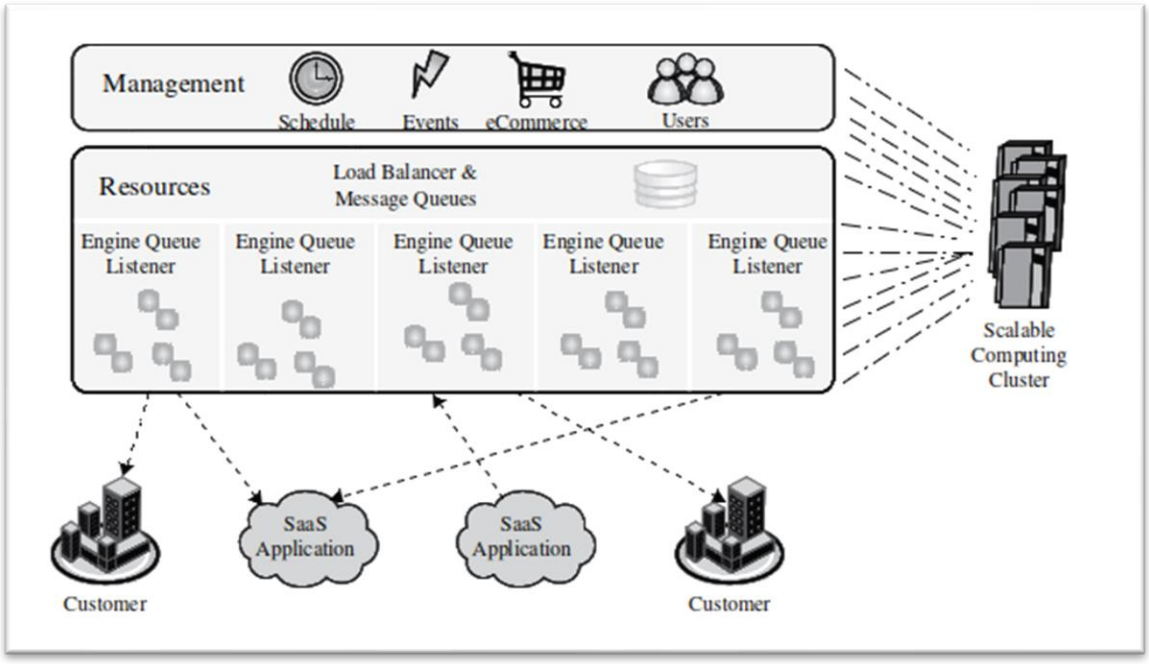
Businesses must carefully and critically analyze these prominent qualities when choosing cloud/SaaS integration providers. With these considerations in mind, businesses can achieve efficient and reliable cloud integration that meets their unique needs.

In the field of software integration, various vendors offer solutions that address the challenges associated with integrating multiple applications in the cloud. These solutions aim to make integration a seamless process, enabling organizations to leverage the benefits of cloud-based applications without the hassle of complex and time-consuming integration processes.

- One such solution is Boomi Atom Sphere, which offers a pure SaaS integration service that connects various applications, including SaaS, PaaS, cloud, and on-premise applications. Atom Sphere provides a centralized platform that allows for the secure building, deployment, and management of simple to complex integration processes using only a web browser. With Atom Sphere, new applications become instantly accessible to the entire community with no additional adapters or upgrades required.
- Another solution is Bungee Connect, which caters to professional developers by offering a cloud development and deployment platform that simplifies the integration of multiple data sources into a single application. Bungee Connect automates the development of rich UI and streamlines the deployment process across multiple web browsers. By leveraging cloud development, Bungee Connect provides added value to organizations committed to building cloud-based applications.
- OpSource Connect is another platform that addresses the challenges of SaaS integration by unifying different SaaS applications in the cloud and legacy applications running behind a corporate firewall. The OpSource Services Bus (OSB) forms the foundation for the platform, enabling applications running on the OpSource On-Demand platform to quickly and easily access web services. This eliminates the need to write code for these business functions, as OpSource has already invested in the upfront development. OpSource Connect also provides a range of features including OpSource Service Connectors, OpSource Connect Certified Integrator Program, OpSource Connect Service Xchange, and OpSource Web Services Enablement Program.
- Snap Logic is yet another data integration platform that can be deployed in both enterprise and cloud landscapes. Snap Logic provides a unique hybrid approach that combines web principles and open-source software with traditional data integration capabilities. The platform is designed to meet the changing requirements imposed by changing data sources, deployment options, and delivery needs. Snap Logic features a Transformation Engine and Repository that can solve even the most complex data integration scenarios. The Snap Logic server is built on a core of connectivity and transformation components, and the Snap Logic designer runs in any web browser, providing an efficient and productive environment for developing transformation logic.

These solutions offer a range of features and capabilities that enable organizations to simplify and streamline the integration of various applications in the cloud. By providing a centralized platform that eliminates the need for complex integration processes, these solutions enable organizations to leverage the benefits of cloud-based applications and services without the hassle of time-consuming integration.

Pervasive Data Cloud is a multi-tenant platform that provides on-demand computing capacity for deploying data-centric applications, including integration solutions. The platform offers Integration as a Service (IaaS) for hosted and on-premises applications and data sources, packaged turnkey integration, connectivity to hundreds of different applications and data sources, and supports every integration scenario.



Pervasive Integrator Connects Different Resources

- Pervasive Data Cloud deploys data-centric applications via several solutions, including the Pervasive Data Synch family of packaged integrations, Pervasive Data Integrator, data migration, consolidation, and conversion, ETL/data warehouse, B2B/EDI integration, application integration (EAI), SaaS/cloud integration, SOA/ESB/web services, data quality/governance, and hubs. The platform provides multi-tenant, multi-application, and multicenter deployment, which is scalable, flexible, easy to access and configure, robust, secure, and affordable.
- On the other hand, Blue Wolf's Integration-as-a-Service (IaaS) solution is the first to offer ongoing support of integration projects guaranteeing successful integration between diverse SaaS solutions, such as salesforce.com, Big Machines, eAutomate, OpenAir, and back-office systems (e.g., Oracle, SAP, Great Plains, SQL Service, and MySQL). The solution, the Integrator, includes proactive monitoring and consulting services to ensure integration success. With remote monitoring of integration jobs via a dashboard included as part of the Integrator solution, Blue Wolf proactively alerts its customers of any issues with integration and helps to solve them quickly.

The Blue Wolf Integrator is designed with user-friendly administration rules that enable the administrator to manage data flow between front and back-office systems with little or no IT support. With a wizard-based approach, the Integrator prompts are presented in simple and non-technical terms. The solution integrates with several platforms, including Salesforce, Big Machines, Oracle, SAP, Microsoft SQL server, MySQL, and supports flat files, such as CSV, XHTML, and many more. Overall, both Pervasive Data Cloud and Blue Wolf's Integration-as-a-Service (IaaS) solution offer robust cloud-based data integration solutions that cater to different needs and use cases.

The emergence of cloud-based data integration platforms has paved the way for real-time data sharing among enterprise information systems and cloud applications. However, the fast-emerging option is to link enterprise and cloud systems via messaging, leading vendors and service organizations to take message-oriented middleware (MoM) to the cloud infrastructure. The messaging middleware as a service (MMaaS) is the grand derivative of the software as a service (SaaS) paradigm, providing integration as a service (IaaS) to hundreds of distributed and enterprise applications. Cloud-based queuing systems such as Online MQ, CloudMQ, and Linxter are accomplishing message-based application and service integration, with messaging being provided to multiple applications using the multi-tenancy property. The integration complexity between different applications and databases that are separated by syntactic, structural, schematic, and semantic deviations can be minimized using data mapping tools and templates. Moreover, scores of adaptors for automating connectivity and integration needs are taking off the ground successfully. The standards-compatible enterprise service bus (ESB) is being taken to the cloud infrastructure to guarantee message enrichment, mediation, content, and context-based message routing. This move will enable loosely or lightly coupled and decoupled cloud services and applications to become a reality, with the maturity and durability of message-centric and cloud-based service bus suites.

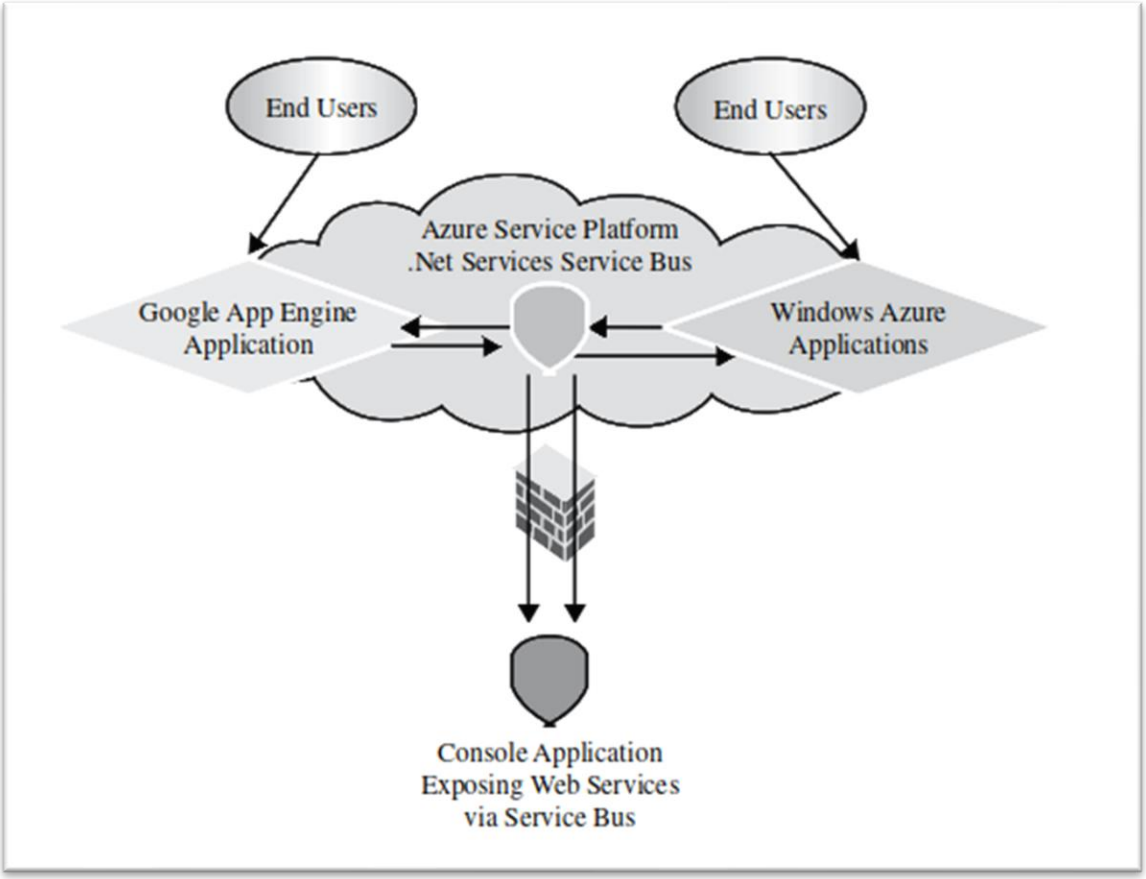
Soon, complex event processing (CEP) engines will be deployed in clouds to capture and capitalize streams of events from diverse sources in different formats and forms, to infer the existing and emerging situation precisely and concisely. Context-aware applications covering all kinds of constituents and participants, enterprise systems, integration middleware, cloud services, and knowledge engines can be built and sustained in a highly interoperable environment, enabling seamless and spontaneous composition and collaboration to create sophisticated services dynamically. Composite applications, services, data, views, and processes will become cloud-centric and hosted to support spatially separated and heterogeneous systems, with frameworks such as service component architecture (SCA) being revitalized to make them fit for cloud environments.

Informatica On-Demand is a suite of SaaS-based data integration solutions developed by Informatica. It enables seamless and secure integration of data between SaaS and on-premise applications over the internet, without requiring the purchase or hosting of any software. This integration service is delivered via an on-demand or as-a-service model, with all relevant features and functions available through a subscription-based service.

- One of the key benefits of leveraging Informatica On-Demand is the rapid development and deployment of integration technology, with zero maintenance required. The technology is also continuously upgraded and enhanced by the vendor, offering users the latest and most up-to-date solutions for their data integration needs.
- Another advantage of Informatica On-Demand is its proven SaaS integration solutions, including its integration with Salesforce.com, which provides users with pre-established connections and metadata understanding. Moreover, the platform's core integration services such as connectivity and semantic mediation are built into the technology, which has proven data transfer and translation capabilities.
- Informatica On-Demand has taken a unique approach by transforming its PowerCenter Data Integration Platform into a true multi-tenant solution, hosted on the cloud. This means that new features and enhancements are immediately available to all customers, without requiring complex software upgrades or additional fees. The platform's multi-tenant architecture also ensures that bandwidth and scalability are shared resources, simplifying the process of meeting varying capacity demands.

Moving on to Microsoft Internet Service Bus (ISB), it is a cloud middleware that provides a common infrastructure for naming, discovering, exposing, securing, and orchestrating web services. ISB is part of the

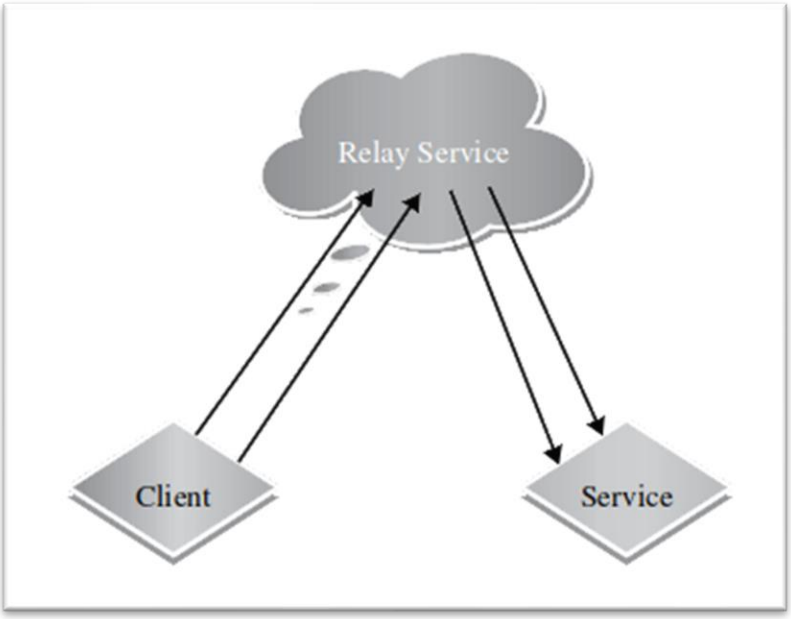
larger Azure cloud operating system developed by Microsoft, which aims to make the development, deployment, and delivery of web and Windows applications on cloud centers easier and more cost-effective.



.NET Service Bus

ISB is composed of three broad areas, including the .NET Service Bus, the .NET Access Control Service, and the .NET Workflow Service. The .NET Service Bus provides a secure and broadly accessible infrastructure for large-scale event distribution, naming, and service publishing. Its connectivity options allow endpoints to be located behind network address translation boundaries or bound to frequently changing, dynamically assigned IP addresses.

- The .NET Access Control Service, on the other hand, provides a hosted, secure, standards-based infrastructure for multiparty, federated authentication, rules-driven, and claims-based authorization. This ranges from simple user name/password-based authentication and authorization to sophisticated WSFederation scenarios.
- The .NET Workflow Service offers a hosted environment for service orchestration based on the familiar Windows Workflow Foundation (WWF) development experience. Its specialized activities for rules-based control flow, service invocation, message processing, and correlation can be executed on demand, on schedule, and at scale inside the .NET Services environment.



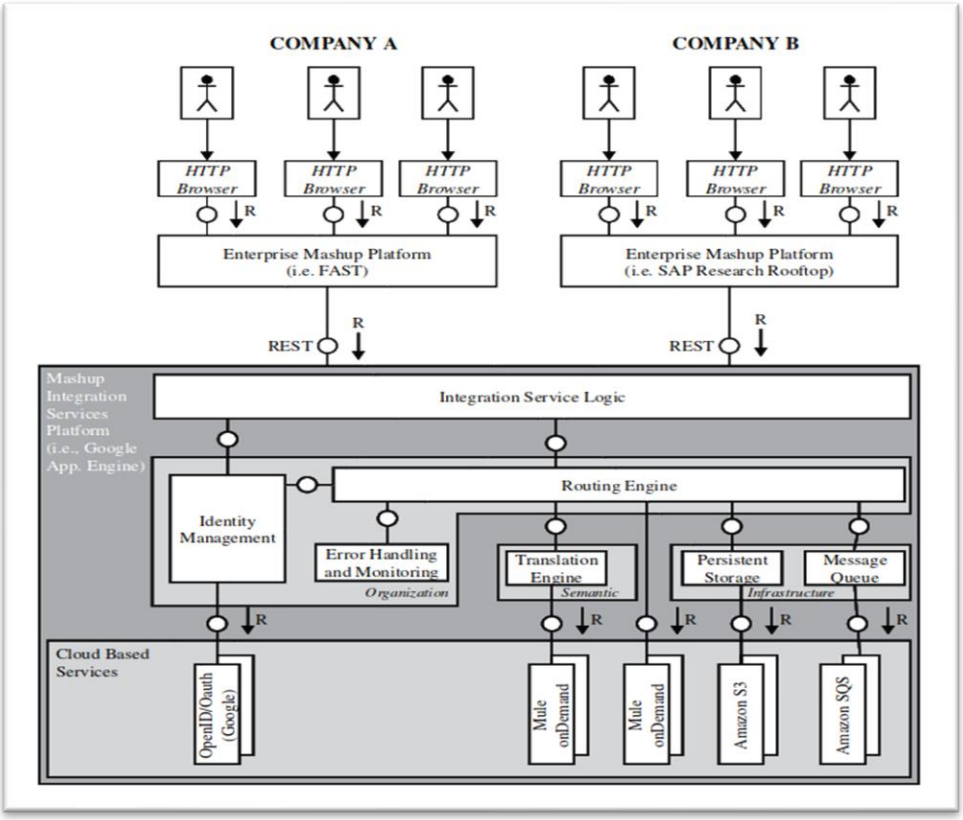
The .NET Relay Service

The key capabilities of the Azure service bus are a federated namespace model, a service registry service, a publish/subscribe event bus, and a relay and connectivity service that provides advanced NAT traversal and pull-mode message delivery capabilities. The relay service helps overcome connectivity challenges by providing a relay service in the cloud that assists connectivity and relays the service calls between clients and the service, thereby addressing scalability, availability, and security issues.

B2Bi systems are an excellent candidate for Infrastructure-as-a-Service (IaaS) as they are traditionally employed to automate business processes between manufacturers and their external trading partners. These systems provide application-to-application (A2A) connectivity, along with the necessary functionality for linking internal and external software. This includes secure data exchange across the corporate firewall, file encryption for safe passage across public networks, large data volume management, batch file transfer, and conversion of disparate file formats. Additionally, B2Bi platforms guarantee data accuracy, integrity, confidentiality, and delivery. The use of a hub-and-spoke centralized architecture further simplifies implementation, provides good control and grip

on system management, and avoids placing an excessive processing burden on the customer side. The hub is installed at the SaaS provider's cloud center to do the heavy lifting, such as reformatting files. A spoke unit, consisting of a small downloadable Java client, is then deployed at each user site to handle basic tasks, such as data transfer. This also eliminates the need for expensive server-based solutions, data mapping, and other tasks at the customer location. Furthermore, enterprises can leverage IaaS to sync up with their partners across the globe, fostering smart and systematic collaboration.

However, there is still a vast need for situational, ad-hoc B2B applications desired by most business end-users. Current solutions focus on automating long-term business relationships and lack intuitive ways to modify or extend them according to ad-hoc or situational needs. Enterprise mashup and lightweight composition approach and tools are promising methods to unleash the huge and untapped potential of empowering end-users to develop or assemble aligned and aware composite services. Enterprise mashups are new-generation web-based application that seem to adequately fulfill the individual and heterogeneous requirements of end-users and foster end-user development. To shorten the traditional and time-consuming development process, this new breed of applications is developed by non-professional programmers, often in a non-formal, iterative, and collaborative way by assembling existing building blocks. Service-Oriented Architecture (SOA) has been presented as a potent solution to an organization's integration dilemmas. However, most Enterprise Service Buses (ESBs) are not designated for cross-organizational collaboration, and problems arise when articulating and aiming for such an extended collaboration. Additionally, end-users usually lack the skill to realize the desired integration scenarios, which leads to high costs for integration projects and unwanted inflexibility.

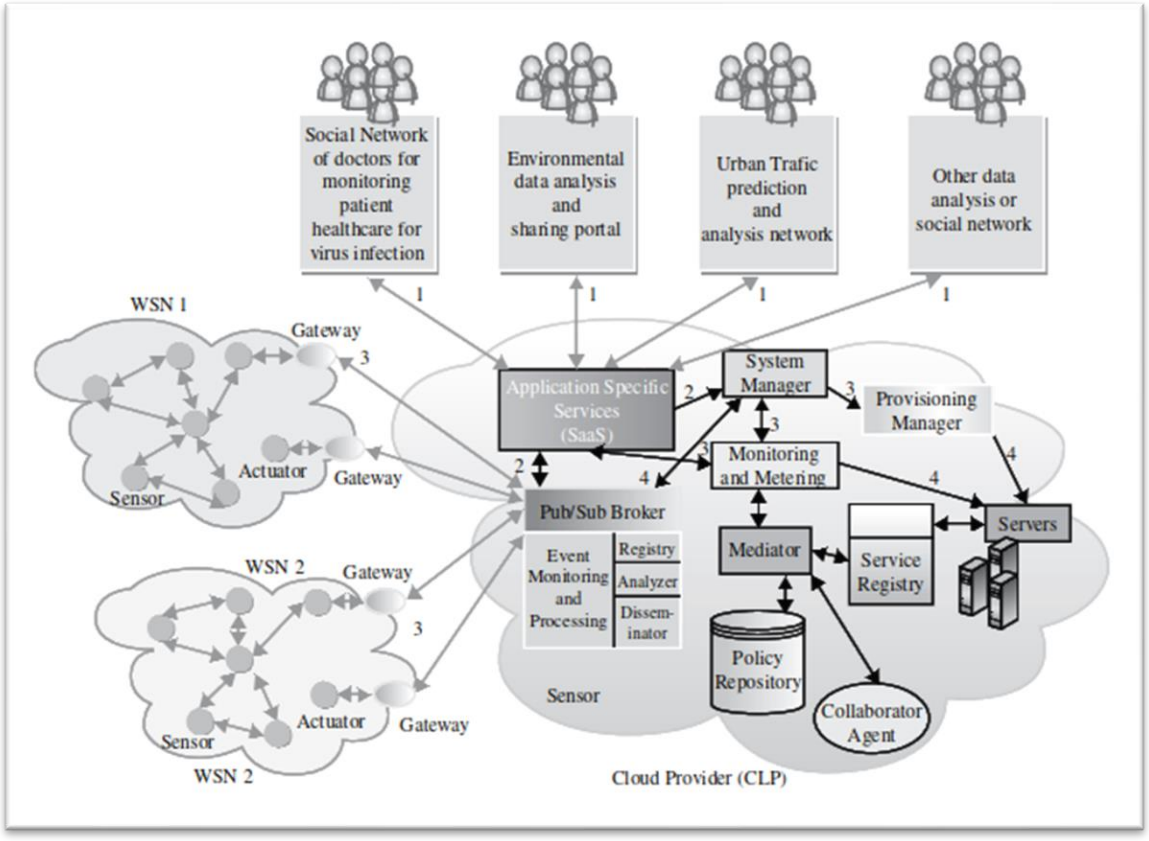


Cloud-based Enterprise Mashup Integration Platform Architecture

One of the challenges in B2B integration is the ownership and responsibility for processes. In many inter-organizational settings, business processes are loosely coupled and/or based on ad-hoc cooperation, making collaboration more complex as more participants become involved. This complexity draws a significant number of differing opinions and viewpoints, making process ownership and responsibility difficult to assign. The Sensor-Cloud Integration framework has emerged as a popular solution to address the challenges in integrating wireless sensor networks (WSNs) with cloud-based community-centric applications. The growing popularity of WSNs has led to their application in diverse fields such as industrial automation, environmental monitoring, transportation business, healthcare, and, more. With the integration of WSNs and cloud-based applications, various innovative and useful solutions can be achieved.

However, the integration of WSNs with cloud-based applications comes with its own set of challenges, such as the need for a robust and resilient framework to support large-scale parallel analysis of sensor data and the need for seamless collaboration between multiple cloud providers. The authors of the Sensor-Cloud Integration framework propose a pub-sub-based model to address these challenges and simplify the integration process.

The framework enables the aggregation, processing, and dissemination of sensor data based on subscriptions, making it possible for researchers to register their interests and access patient data from bio-sensors for large-scale parallel analysis. The framework also facilitates information sharing among researchers to find useful solutions to complex problems, such as tracking the spread of infection during an outbreak of a new virus strain.



The Framework Architecture of Sensor—Cloud Integration

To support the massive computational power and storage requirements of sensor data, the authors suggest a dynamic collaboration with multiple cloud providers. This allows for the efficient allocation of resources and ensures that the framework can scale out quickly to meet the needs of the researchers. The Sensor-Cloud Integration framework is a robust and resilient solution that enables the integration of WSNs with cloud-based community-centric applications. With its content-based pub-sub model and support for seamless collaboration between multiple cloud providers, the framework promises to revolutionize how we approach complex problems in diverse fields.

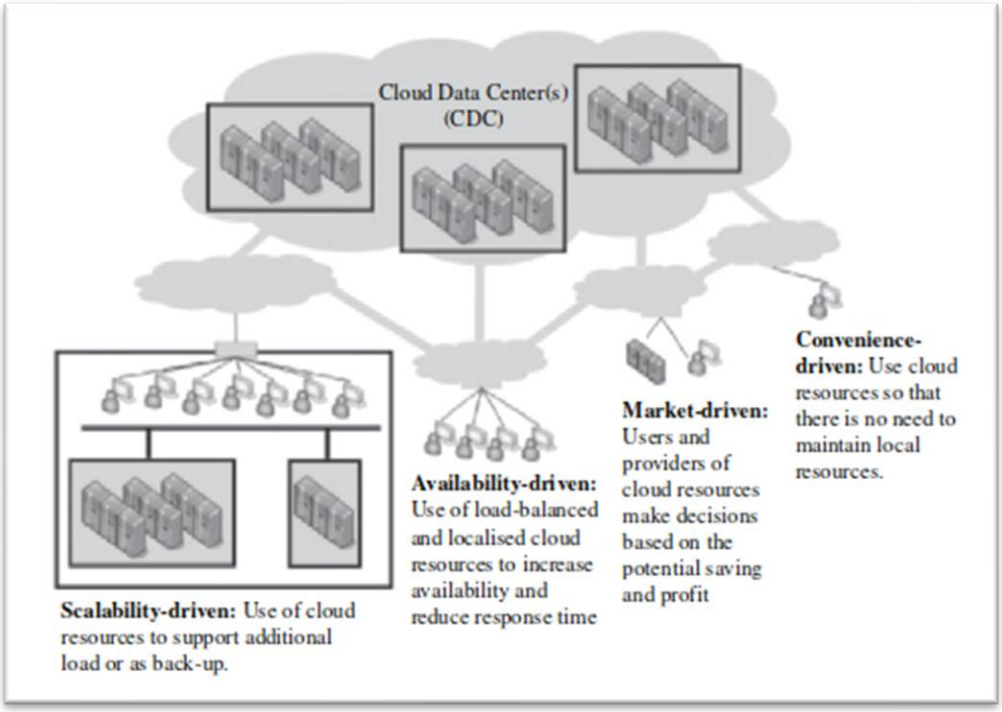
Cast Iron Systems offers pre-configured solutions for today's leading enterprise and on-demand applications. These solutions, built using the Cast Iron product offerings, provide out-of-the-box connectivity to specific applications, along with template integration processes (TIPs) for the most common integration scenarios. For instance, the Cast Iron solution for Salesforce.com comes with built-in AppExchange connectivity and TIPs for customer master, product master, and contact data integration. Cast Iron solutions enable customers to complete application-specific integrations rapidly, using a "configuration, not coding" approach. By utilizing pre-configured templates, rather than starting from scratch with complex software tools and writing copious amounts of code, enterprises can complete critical projects in mere days instead of months.

Moreover, the total cloud infrastructure, comprising prefabricated software modules, can also be produced as an appliance, facilitating the creation of private clouds with ease and speed. In fact, the trend of "appliance as a service" is currently sweeping the cloud service provider (CSP) industry, indicating the importance of such solutions in meeting the evolving needs of modern IT.

The Enterprise Cloud Computing Paradigm

Cloud computing is a constantly evolving field that is marked by the emergence of new vendors, services, and offers. This evolution is primarily driven by the need to satisfy the ever-changing demands of consumers. While cloud computing has primarily been adopted by start-ups or small and medium-sized enterprises (SMEs), widespread enterprise adoption of the cloud computing model is still in its infancy.

Enterprises are still exploring the various usage models where cloud computing can be employed to support their business operations. To pave the way for more widespread adoption of cloud computing, cloud providers will have to meet the stringent requirements of enterprises. This will lead to what is known as enterprise cloud paradigm computing. The enterprise cloud paradigm computing involves aligning a cloud computing model with an organization's business objectives and processes, such as profitability, return on investment, and reduction of operational costs. To fully understand this paradigm, it is important to explore its motivations, objectives, strategies, and methods.



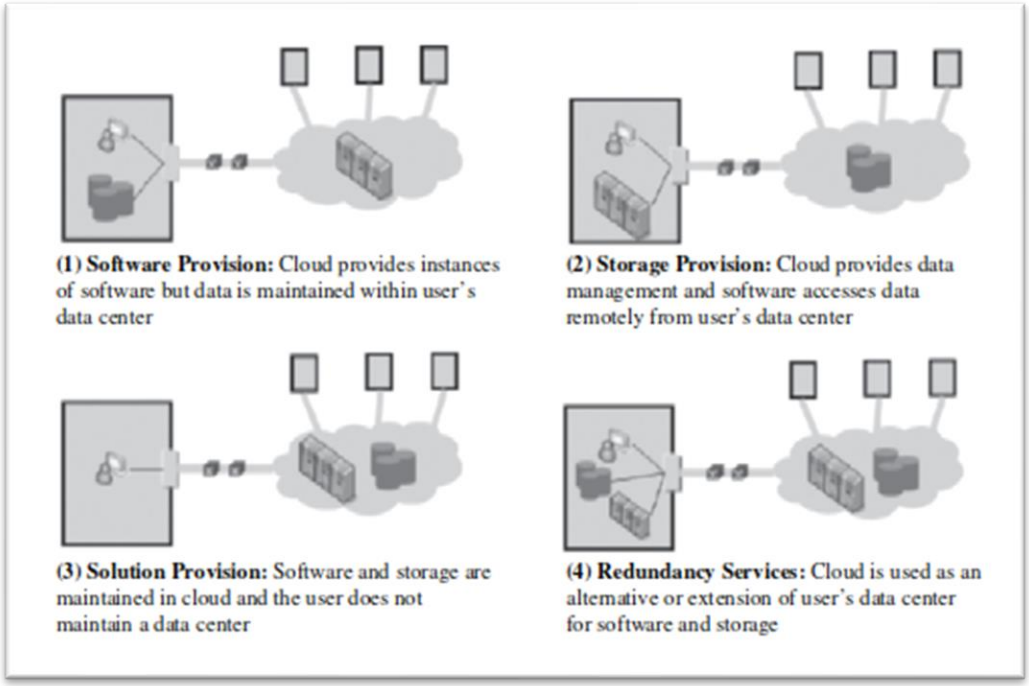
Enterprise cloud adoption strategies using fundamental cloud drivers

One aspect of enterprise cloud paradigm computing is the selection of deployment models and strategies. The National Institute of Standards and Technology (NIST) has identified five essential characteristics of cloud computing:

- On-demand self-service
- Broad network access, resource pooling
- Rapid elasticity
- Measured service

These characteristics are manifested in different ways depending on the deployment model employed.

Some common cloud deployment models include public clouds, private clouds, virtual private clouds, community clouds, and managed clouds. The selection of a deployment model depends on the opportunities to increase earnings and reduce costs, such as capital expenses (CAPEX) and operating expenses (OPEX). Another critical aspect of enterprise cloud paradigm computing is the selection of adoption and consumption strategies. An organization must decide whether an enterprise cloud strategy will increase overall business value and whether the effort and risks associated with transitioning to an enterprise cloud strategy are worth it. They must also consider which areas of business and IT capability should be considered for the enterprise cloud, which cloud offerings are relevant for the organization's purposes, and how transitioning to an enterprise cloud strategy can be piloted and systematically executed.



Enterprise cloud consumption strategies

Issues for Enterprise Applications on the Cloud

Enterprise applications, particularly Enterprise Resource Planning (ERP), are critical to the success of any organization as they provide a comprehensive tool for optimizing business processes with seamless, integrated information flow. ERP solutions are becoming increasingly essential as organizations face growing consumer demands, globalization, and competition. While successful implementation of ERP systems has resulted in a standardized, integrated, and efficient work environment, organizations must constantly improve their business practices and procedures to remain competitive.

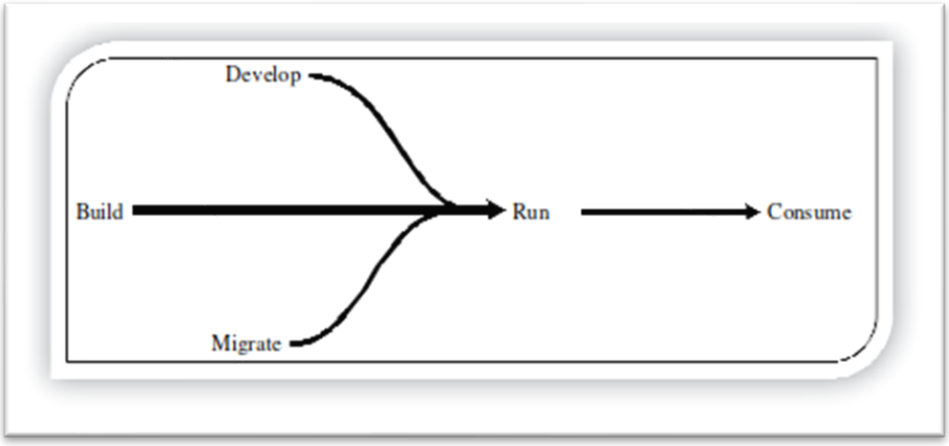
One of the key challenges to ERP implementation on the cloud is infrastructure availability. Adequate IT infrastructure, hardware, and networking are critical to the success of any ERP system. The transition from legacy information systems and business processes to an integrated IT infrastructure requires careful hardware selection, driven by the organization's choice of an ERP software package. The ERP software vendor typically certifies which hardware and hardware configurations must be used to run the ERP system.

Infrastructure-as-a-Service (IaaS) offerings hold promising future scenarios for the implementation of ERP systems. Recent surveys show that nearly all kinds of workloads are seen as suitable to be transferred to IaaS offerings. Companies are just as comfortable deploying production workloads on the cloud as test and development workloads. Technical issues arise when considering the operational characteristics and behaviors of transactional and analytical applications, which underlie the capabilities of ERP. Transactional applications, or OLTP (Online Transaction Processing) applications, manage transaction-oriented applications using relational databases. These applications rely on strong ACID properties (atomicity, consistency, isolation, durability) and are relatively write/update-intensive. OLTP-type ERP components such as sales and distribution, banking and financials, customer relationship management, and supply chain management face major technical and non-technical challenges to deploy in cloud environments. These applications provide mission-critical functions and enterprises have clear security and privacy concerns. Furthermore, classical transactional systems typically use a shared-everything architecture, while cloud platforms mostly consist of shared-nothing commodity hardware. ACID properties are also difficult to guarantee given the concurrent cloud-based data management and storage systems.

Analytical applications, or OLAP (Online Analytical Processing) applications, are used to efficiently answer multidimensional queries for analysis, reporting, and decision support. OLAP applications are relatively read-most or read-only, and ACID guarantees are typically not required. Due to their data-intensive and data-parallel nature, these applications can benefit greatly from the elastic computing and storage available in the cloud. Business Intelligence (BI) and analytical applications are better suited to run in a cloud platform with a shared-nothing architecture and commodity hardware. Opportunities arise in the vision of Analytics as a Service or Agile Analytics. Data sources residing within private or public clouds can be processed using elastic computing resources on-demand, accessible via APIs, web services, SQL, BI, and data mining tools. While transitioning to the cloud can be challenging, particularly in the case of ERP systems, the benefits of cloud computing are hard to ignore. The infrastructure availability and technical issues can be addressed by careful hardware selection, leveraging the benefits of IaaS, and identifying opportunities to decompose complex applications into simpler functional components that can be engineered accordingly. Cloud computing offers unmatched performance and total cost of ownership benefits toward large-scale analytic processing. Leading providers have already offered on-demand tailored business suite solutions as hosted services, making it easier for companies to adopt cloud-based ERP systems.

Transition Challenges

The adoption of cloud technology represents a significant shift in the traditional approach to IT service delivery. As with any major leap, there are inherent risks and challenges to overcome. These challenges can be categorized into five distinct areas: build, develop, migrate, run, and consume.



Five stages of the cloud

For comprehensive ERP transitioning, private, and hybrid cloud models are currently the most relevant. The first challenge facing organizations embarking on this transition is gaining a thorough understanding of the state of their own IT assets. This audit will help determine what can be salvaged from the existing infrastructure and how high in the cloud stack companies should venture. Most companies are likely to stick to Infrastructure as a Service (IaaS), although major development shops may delve into Platform as a Service (PaaS) and Software as a Service (SaaS).

The migration of existing or "legacy" applications to the cloud presents the second challenge. With the expected average lifetime of ERP products being 15 years, companies must address this aspect sooner than later. An application migration is not a straightforward process and poses risks, without always guaranteeing better service delivery. It is essential to guarantee that the migration process can be agnostic of the underlying chosen cloud technology. If automation is possible, the same amount of planning, negotiation, and testing required for risk mitigation as classical software will still be necessary. The concept of decoupling processes is essential for migrating to the cloud. Work needs to be organized using a process-centric model, rather than the standard "silo" approach commonly used in IT. Not all applications will handle such migration without costly overall reengineering, and rebuilding from scratch poses governance, reliability, security/trust, data management, and control/predictability hurdles.

The challenges of cloud operations can be divided into running the enterprise cloud and running applications on the enterprise cloud. Upgrading and updating all IT department components, particularly the human factor, is a significant challenge in the changing IT operations of the day-to-day operation. Once the IT organization has upgraded to provide cloud or is able to tap into cloud resources, maintaining services in the cloud presents further difficulties. Interoperability between in-house infrastructure and service and the Cloud Data Center (CDC) must be maintained.

Most enterprise applications do not require elasticity, which is touted as a killer feature for enterprise applications. Without proper monitoring, troubleshooting, and comprehensive capacity planning, visibility into

the return on investment and the consumption of cloud services become very difficult. Today, there are two major cloud pricing models: allocation-based and usage-based. Finding the right combination of billing and consumption models is a daunting task. Companies must also evaluate hidden costs such as lost IP, risk, migration, delays, and provider overheads. Choosing a new mobile with a carrier plan pales in comparison to finding the ideal cloud service provider. Some providers are proposing a subscription scheme to palliate their limited resources.

Enterprise Cloud Technology and Market Evolution

One of the primary drivers of this evolution is the concern of all stakeholders within the cloud ecosystem regarding vendor lock-in, which is caused using proprietary interfaces, formats, and protocols by cloud vendors. As more and more organizations formulate cloud adoption strategies, the demand for open interoperable standards for cloud management interfaces and protocols, data formats, and other such requirements will become more apparent. This will pressure cloud providers to build their offerings on open interoperable standards to be considered as a viable option by enterprises.

Porter's five forces market model

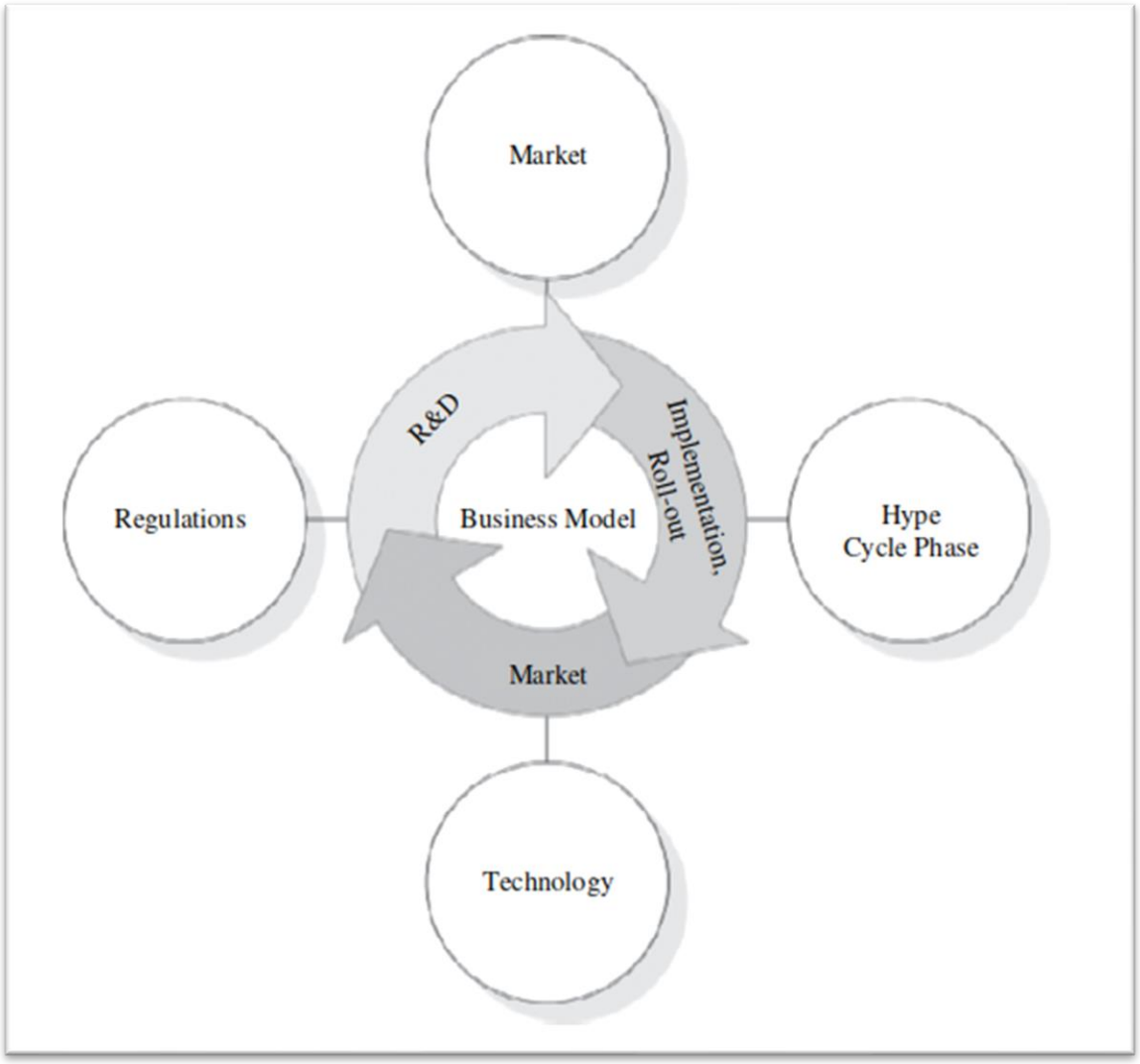
Several initiatives, such as OGF OCCI for computing clouds, SNIA CDMI for storage and data management, and DMTF Virtualization Management, have emerged in this space. However, widespread participation in these initiatives is still lacking, especially among the major cloud vendors such as Amazon, Google, and Microsoft, who are not actively involved in these efforts. Although achieving true interoperability across the board soon seems unlikely, it could lead to the facilitation of advanced scenarios and drive the mainstream adoption of the enterprise cloud computing paradigm. Additionally, standards-based cloud offerings are critical for the evolution and spread of this paradigm because standards drive choice and choice drives the market.

Organizations need assurance that cloud resources and services powering their business operations perform according to business requirements. Service level agreements (SLAs) can prove to be a useful instrument in facilitating enterprises' trust in cloud-based services. Currently, cloud solutions come with primitive or non-existent SLAs. This is bound to change as the cloud market gets more crowded, and providers have to gain competitive differentiation to capture a larger share of the market. This is particularly true for market segments represented by enterprises and large organizations. Enterprises will be interested in choosing offerings with sophisticated SLAs that provide assurances for performance issues and compliance with security or governance standards.

Another important factor is the lack of insights into the performance and health of the resources and services deployed on the cloud. Cloud providers do not currently offer sophisticated monitoring and reporting capabilities that enable customers to comprehend and analyze the operations of these resources and services. However, solutions have started to emerge to address this issue, and it is believed that the situation will improve as the enterprise cloud adoption phenomenon makes it imperative for cloud providers to deliver sophisticated monitoring and reporting capabilities for customers.

Looking at the cloud services stack (IaaS, PaaS, SaaS), the applications space or SaaS has the most growth potential. As forecasted by analyst IDC, applications will account for 38% of the $44.2 billion cloud services market by 2103. Although enterprises have already begun to adopt some SaaS-based solutions, these are primarily edge applications like supplier management, talent management, and performance management, as opposed to core business processes. Integration capabilities will drive mainstream SaaS adoption by enterprises. Moreover, organizations will opt for SaaS applications from multiple service providers to cater to various operational segments of an enterprise, adding an extra dimension of complexity to integration mechanisms.

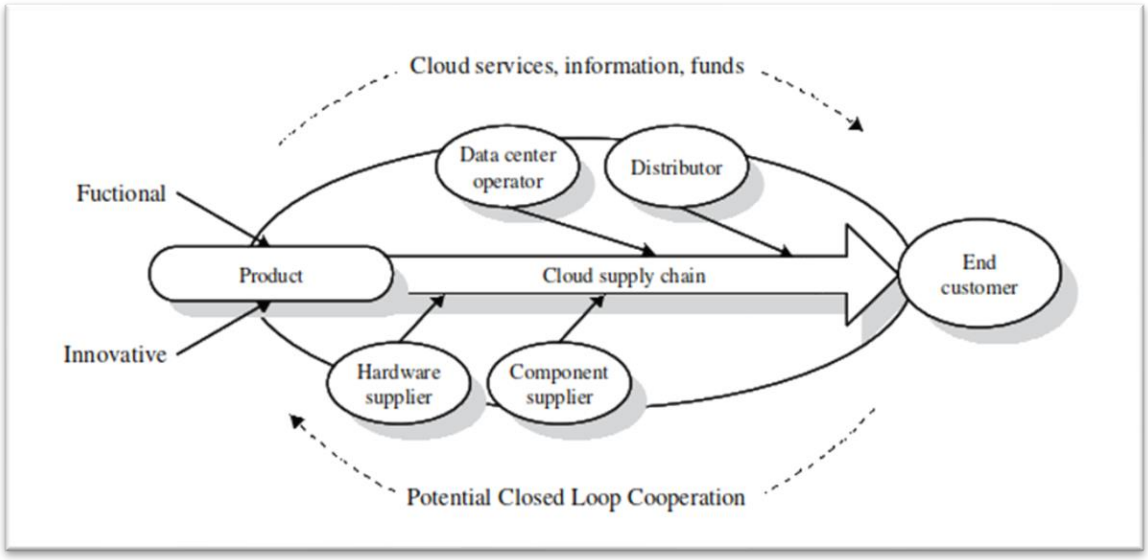
Cloud computing has given rise to alternative data storage technologies based on key-type storage models, such as Amazon Dynamo, Facebook Cassandra, and Google BigTable, as compared to the relational model, which has been the mainstream choice for data storage. This is an emerging trend in the cloud application space and is expected to evolve over time.



Dynamic business models

The Cloud Supply Chain

The business model for enterprise cloud computing can be complex, involving the deployment, security, interconnection, and maintenance of enterprise landscapes and solutions such as ERP. In this context, the concept of a Cloud Supply Chain (C-SC) and Cloud Supply Chain Management (C-SCM) may represent a viable future business model. C-SCM involves the management of a network of interconnected businesses that provide end-to-end product and service packages to customers.



Cloud supply chain (C-SC)

A traditional supply chain typically involves two or more parties linked by a flow of goods, information, and funds. However, a C-SC is defined as two or more parties linked by the provision of cloud services, related information, and funds. The figure illustrates a conceptual representation of the C-SC, showing the flow of products across different organizations such as hardware suppliers, software component suppliers, data center operators, distributors, and the end customer.

The C-SC can be classified into two types of products: functional and innovative. Functional products are characterized by stability, low product variety, low inventory costs, and low obsolescence. They favor competition, which leads to low-profit margins. In contrast, innovative products are characterized by unpredictable demand, short product life cycles, and high uncertainties. Cloud services fulfill the basic needs of customers, favor competition, and show characteristics of innovative products due to their unpredictable demand and short development circles.

	Efficient SC	Responsive SC	Cloud SC
Primary goal	Supply demand at the lowest level of cost	Respond quickly to demand (changes)	Supply demand at the lowest level of costs and respond quickly to demand
Product design strategy	Maximize performance at the minimum product cost	Create modularity to allow postponement of product differentiation	Create modularity to allow individual setting while maximizing the performance of services
Pricing strategy	Lower margins because price is a prime customer driver	Higher margins, because price is not a prime customer driver	Lower margins, as high competition and comparable products
Manufacturing strategy	Lower costs through high utilization	Maintain capacity flexibility to meet unexpected demand	High utilization while flexible reaction on demand
Inventory strategy	Minimize inventory to lower cost	Maintain buffer inventory to meet unexpected demand	Optimize of buffer for unpredicted demand, and best utilization
Lead time strategy	Reduce but not at the expense of costs	Aggressively reduce even if the costs are significant	Strong service-level agreements (SLA) for ad hoc provision
Supplier strategy	Select based on cost and quality	Select based on speed, flexibility, and quantity	Select on complex optimum of speed, cost, and flexibility
Transportation strategy	Greater reliance on low-cost modes	Greater reliance on responsive modes	Implement highly responsive and low-cost modes

The table provides a comparison of traditional supply chain concepts, such as efficient and responsive supply chains, with emerging information and communication technology (ICT) supply chains, including cloud computing and cloud services. The mixed characterization of the C-SC is reflected in the classification of efficient versus responsive supply chains. Functional products are suited to efficient supply chains, while innovative products align with responsive supply chains. The management and restructuring of services, information, and funds for the optimization of the chain are expensive and contribute to supply chain costs.

Concepts, Technology & Architecture

Fundamental Cloud Computing

Our aim is to provide a comprehensive understanding of the fundamental aspects that make up this technology, including its history, definitions, and key business and technology drivers:

- By exploring the roots of cloud computing, tracing its origins to the concept of utility computing, which was first proposed by computer scientist John McCarthy in 1961. The idea of computing in a "cloud" began to gain momentum in the late 1990s with the emergence of various Internet-based services, such as search engines, email services, and social media platforms.
- It wasn't until 2006 that the term "cloud computing" emerged in the commercial arena, with the launch of Amazon's Elastic Compute Cloud (EC2) services and Google Apps. Since then, cloud computing has grown into a mature technology with a wide range of enterprise-oriented services that provide remotely provisioned storage, computing resources, and business functionality.
- The definitions of cloud computing provided by leading industry analysts, including Gartner, Forrester Research, and the National Institute of Standards and Technology (NIST). While these definitions differ in some respects, they all emphasize the idea of delivering IT-enabled capabilities as a service to external customers using Internet technologies.

Capacity planning is the process of estimating and fulfilling the future demands of IT resources, products, and services of an organization.

Capacity refers to the maximum amount of work that an IT resource can deliver in each period. Capacity planning minimizes the discrepancy between the capacity of an IT resource and its demand to achieve predictable efficiency and performance. The three capacity planning strategies are Lead, Lag, and Match. Planning for capacity can be challenging as it requires estimating usage load fluctuations to balance peak usage requirements without unnecessary over-expenditure on infrastructure. Failure to plan for capacity may result in either over-provisioning, leading to inefficiency, or under-provisioning, leading to transaction losses and usage limitations. IT environments require significant investment for infrastructure expansion because the usage potential of an automation solution is limited by the processing power of its underlying infrastructure. Two costs need to be accounted for: the cost of acquiring new infrastructure and the cost of its ongoing ownership. Operational overhead, such as technical personnel, upgrades and patches, utility bills, security and access control measures, and administrative and accounts staff, represents a considerable share of IT budgets, often exceeding upfront investment costs.

The ownership of internal technology infrastructure imposes compound impacts on corporate budgets, potentially inhibiting the business's responsiveness, profitability, and overall evolution.

Organizational agility is the measure of an organization's responsiveness to change. IT enterprises need to respond to business changes by scaling IT resources beyond the scope of what was previously predicted or planned. Insufficient budgeting may restrict capacity planning efforts, leading to infrastructure limitations that prevent the organization from responding to usage fluctuations, even when anticipated. Changing business needs and priorities may require IT resources to be more available and reliable than before. However, runtime exceptions that bring down hosting servers may occur due to a lack of reliability controls within the infrastructure. Upfront investments and infrastructure ownership costs required to enable new or expanded business automation solutions may be prohibitive, leading to decreased ability to meet real-world requirements. Failure to respond to business changes and maintain organizational agility may threaten a business's overall continuity.

The development of technological innovations has been influenced by a range of pre-existing technologies. Clustering technology is a prime example, as it is used to create a group of independent IT resources that work together as a single system. By keeping component devices in synchronization through dedicated, high-speed-communication links, redundancy, and failover features are inherent in the cluster, which reduces system failure rates and increases reliability and availability. Grid computing, on the other hand, provides a platform in which computing resources are organized into one or more logical pools that are collectively coordinated to provide a high-performance distributed grid. This type of computing differs from clustering, as grid systems are much more loosely coupled and distributed. In other words, they involve computing resources that are heterogeneous and geographically dispersed. This aspect of grid computing has influenced various aspects of cloud computing platforms and mechanisms, particularly in relation to common feature sets such as networked access, resource pooling, scalability, and resiliency.

Virtualization, another important technology platform, is used for the creation of virtual instances of IT resources. By allowing physical IT resources to provide multiple virtual images of themselves, their underlying processing capabilities can be shared by multiple users. Virtualization has also inspired many of the core features found in cloud computing mechanisms, such as the ability to overcome performance, reliability, and scalability limitations of traditional virtualization platforms. Technology innovations are often based on pre-existing technologies, such as clustering, grid computing, and virtualization. These technologies have inspired the development of various cloud computing platforms and mechanisms, and they continue to play a significant role in the evolution of the cloud computing landscape.

Understanding Cloud Computing

Cloud computing is a rapidly evolving field that has seen tremendous growth in recent years. This technology has been driven by a variety of business drivers, including capacity planning, cost reduction, and organizational agility. The need for these drivers has led to the formation of cloud-based platforms that can remotely provision scalable and measured IT resources.

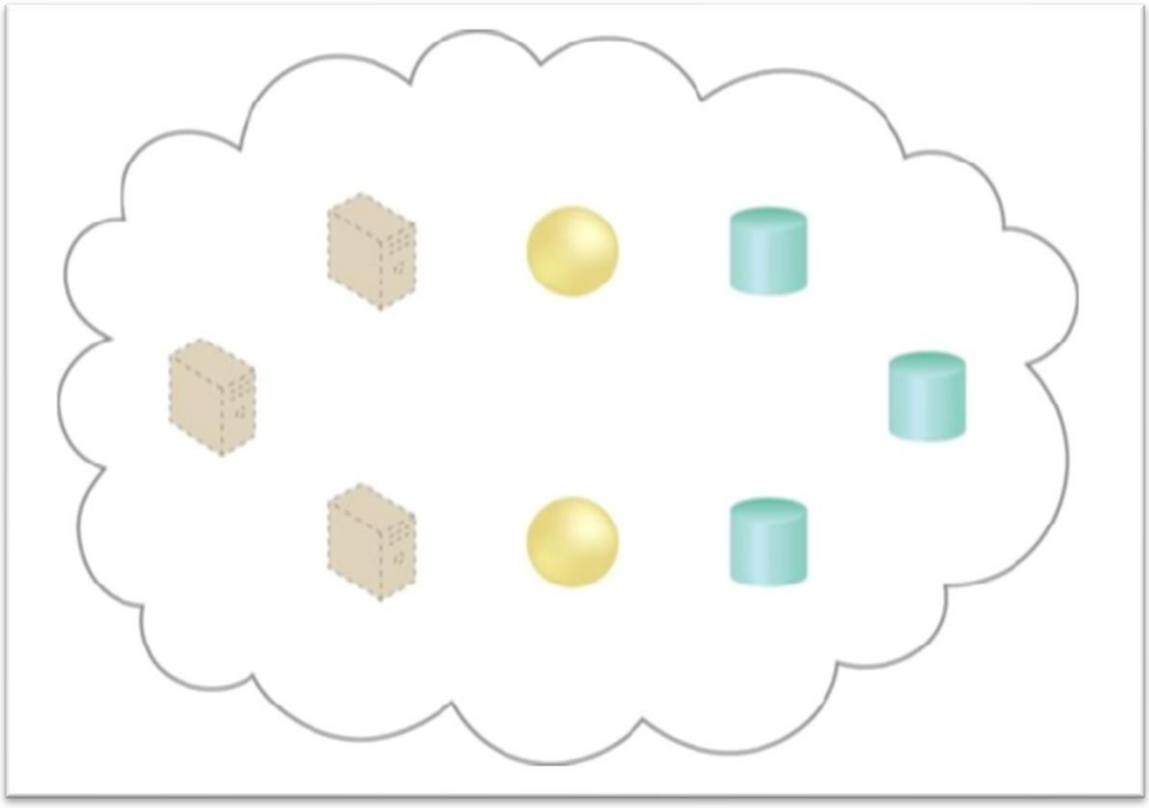
To understand cloud computing, it is essential to identify the primary technology innovations that influenced and inspired key distinguishing features and aspects of cloud computing. These include clustering, grid computing, and traditional forms of virtualization.

Figure

These technologies play a crucial role in the development and implementation of cloud computing. Some of the key cloud-enabling technologies covered in the chapter include broadband networks and internet architecture, data center technology, modern virtualization technology, web technology, multi-tenant technology, and service technology.

Examples of common IT resources

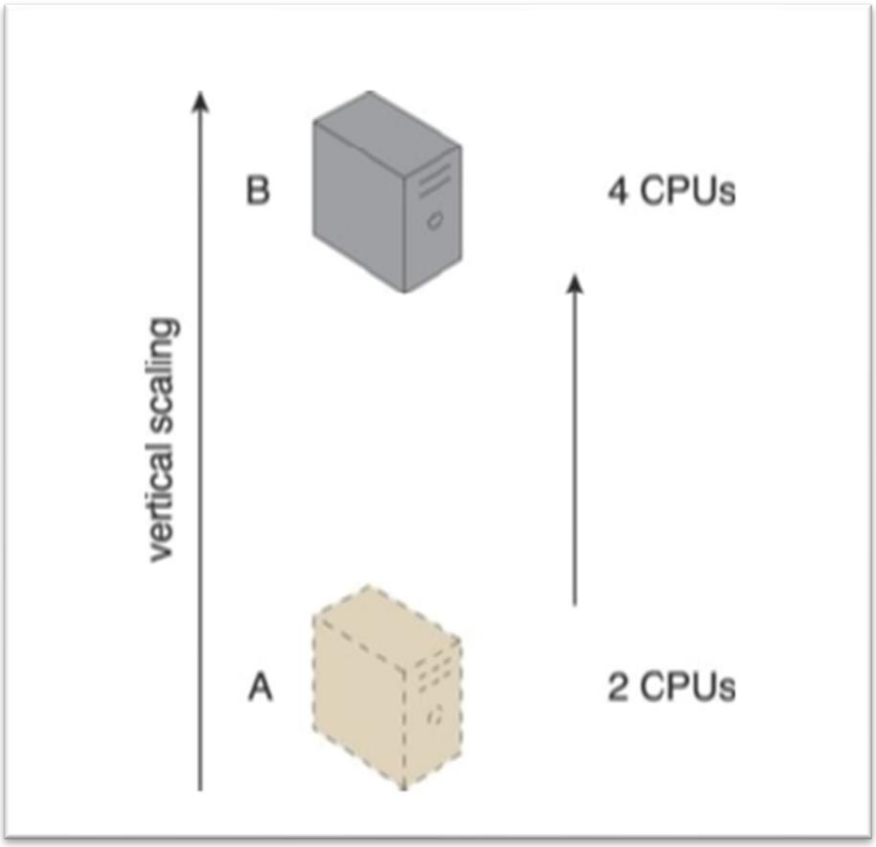
When defining a cloud, it is crucial to understand that it refers to a distinct IT environment designed for the purpose of remotely provisioning scalable and measured IT resources. The cloud symbol originated as a metaphor for the internet, which is, in essence, a network of networks providing remote access to a set of decentralized IT resources.



A cloud is hosting eight IT resources

An IT resource is a physical or virtual IT-related artifact that can be either software-based, such as a virtual server or a custom software program, or hardware-based, such as a physical server or a network device. It is important to distinguish between the cloud and the internet. While the internet provides open access to many web-based IT resources, a cloud is typically privately owned and offers access to IT resources that are metered.

The term "on-premise" is used to describe IT resources that are hosted within a conventional IT enterprise and are not specifically part of a cloud environment. In other words, these resources are located within an organizational boundary that is not cloud-based. On-premise resources cannot be cloud-based, and vice versa. However, on-premise resources can access and interact with cloud-based resources, and they can be moved to the cloud to become cloud-based resources. Both redundant deployments of IT resources and cloud-based environments can exist within an on-premise infrastructure. If the distinction between on-premise and cloud-based IT resources becomes confusing in relation to private clouds, then an alternative qualifier can be used.



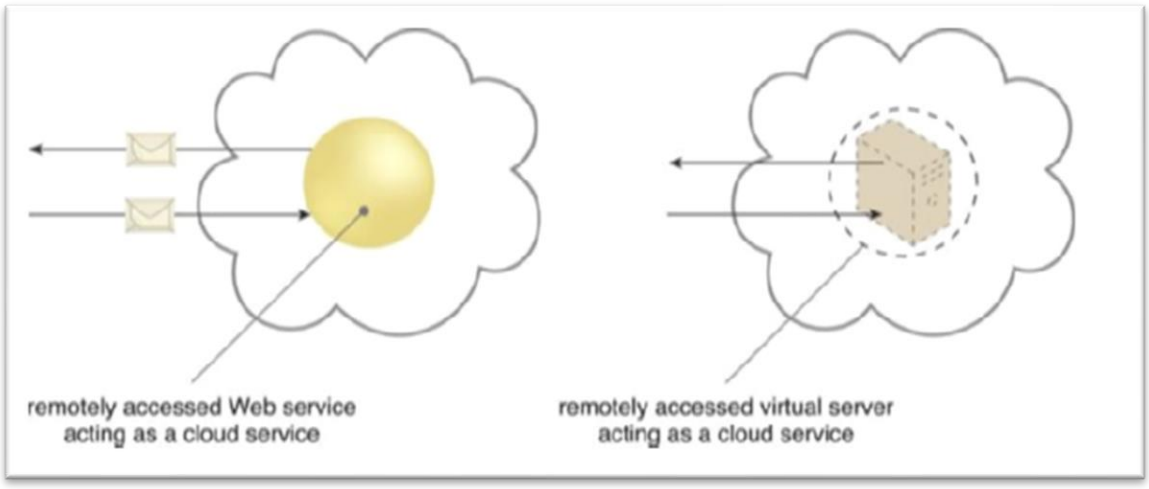
IT resources

In the context of cloud computing, the party that provides cloud-based IT resources is known as the cloud provider, while the party that uses these resources is referred to as the cloud consumer. These roles are typically assumed by organizations in relation to clouds and corresponding cloud provisioning contracts.

Horizontal Scaling	Vertical Scaling
less expensive (through commodity hardware components)	more expensive (specialized servers)
IT resources instantly available	IT resources normally instantly available
resource replication and automated scaling	additional setup is normally needed
additional IT resources needed	no additional IT resources needed
not limited by hardware capacity	limited by maximum hardware capacity

Scaling refers to an IT resource's ability to handle increased or decreased usage demands. There are two types of scaling: horizontal and vertical. Horizontal scaling involves allocating or releasing IT resources of the same

type, while vertical scaling involves replacing an existing IT resource with one of higher or lower capacity. Although horizontal scaling is common in cloud environments, vertical scaling is less so due to the downtime required during the replacement process.



A cloud service with a published technical interface

A cloud service is any IT resource that is made remotely accessible via a cloud. While not all IT resources within a cloud are made available for remote access, a cloud service can exist as a simple Web-based software program with a technical interface invoked via the use of a messaging protocol or as a remote access point for administrative tools or larger environments and other IT resources. The term "service" within the context of cloud computing is quite broad and can refer to a wide range of IT fields falling under the service technology umbrella.

The concept of Cloud Service Consumer refers to the temporary runtime role of a software program that accesses a cloud service. This is demonstrated in the Figure which showcases various types of cloud service consumers, including software programs and services with published service contracts that are capable of remotely accessing cloud services, as well as workstations, laptops, and mobile devices running software that can remotely access other IT resources positioned as cloud services.

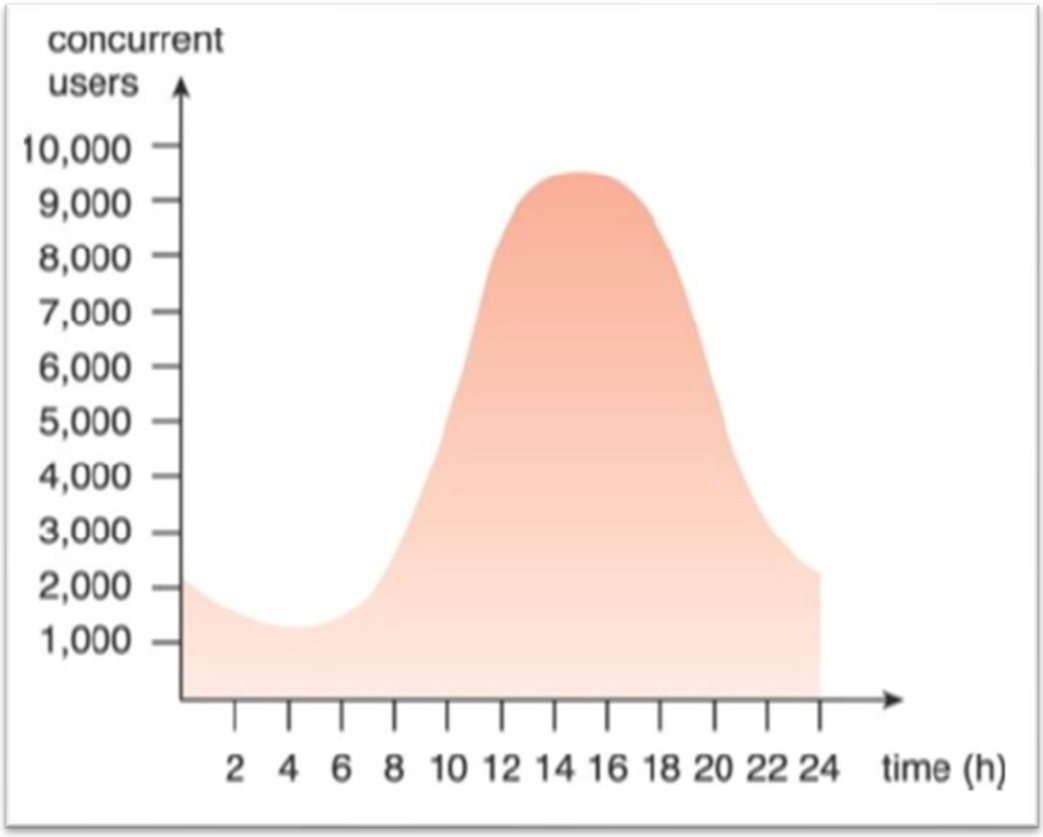
Examples of cloud service consumers

In terms of benefits, cloud computing offers numerous advantages that make it an attractive option for organizations seeking to reduce costs and increase operational efficiency. One of the most significant benefits is the elimination or minimization of upfront financial commitments, such as hardware and software purchases and ownership costs. Cloud providers offer attractively priced leasing packages based on the mass acquisition of

IT resources, which enables organizations to access powerful infrastructure without having to purchase it themselves. Cloud computing also allows for measured usage, which means that measured operational expenditures directly related to business performance can replace anticipated capital expenditures. This feature set is commonly referred to as proportional costs, as it enables organizations to start small and increase IT resource allocation as required. Additionally, the reduction of upfront capital expenses enables the capital to be redirected to core business investments.

The deployment and operation of large-scale data centers by major cloud providers is another opportunity to decrease costs. These data centers are typically located in destinations where real estate, IT professionals, and network bandwidth can be obtained at lower costs, resulting in both capital and operational savings.

Pooled IT resources are made available to and shared by multiple cloud consumers, resulting in increased or even maximum possible utilization. This leads to common measurable benefits to cloud consumers, such as on-demand access to pay-as-you-go computing resources on a short-term basis, the perception of having unlimited computing resources that are available on demand, the ability to add or remove IT resources at a fine-grained level, and abstraction of the infrastructure so applications are not locked into devices or locations and can be easily moved if needed.



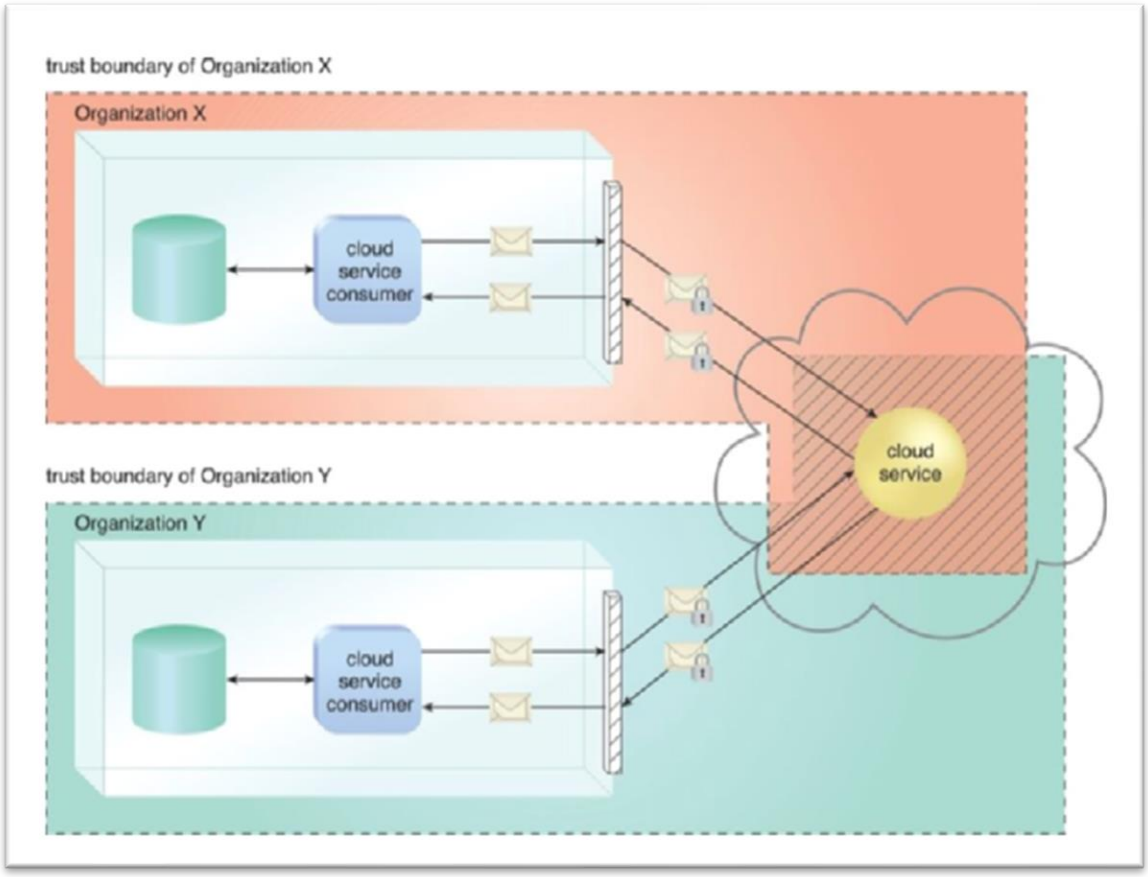
An example of an organization's changing

Despite the ease with which many identify the financial benefits of cloud computing, the actual economics can be complex to calculate and assess. The decision to proceed with a cloud computing adoption strategy will involve much more than a simple comparison between the cost of leasing and the cost of purchasing. For

example, the financial benefits of dynamic scaling and the risk transference of both over-provisioning and under-utilization need to be considered in the decision-making process.

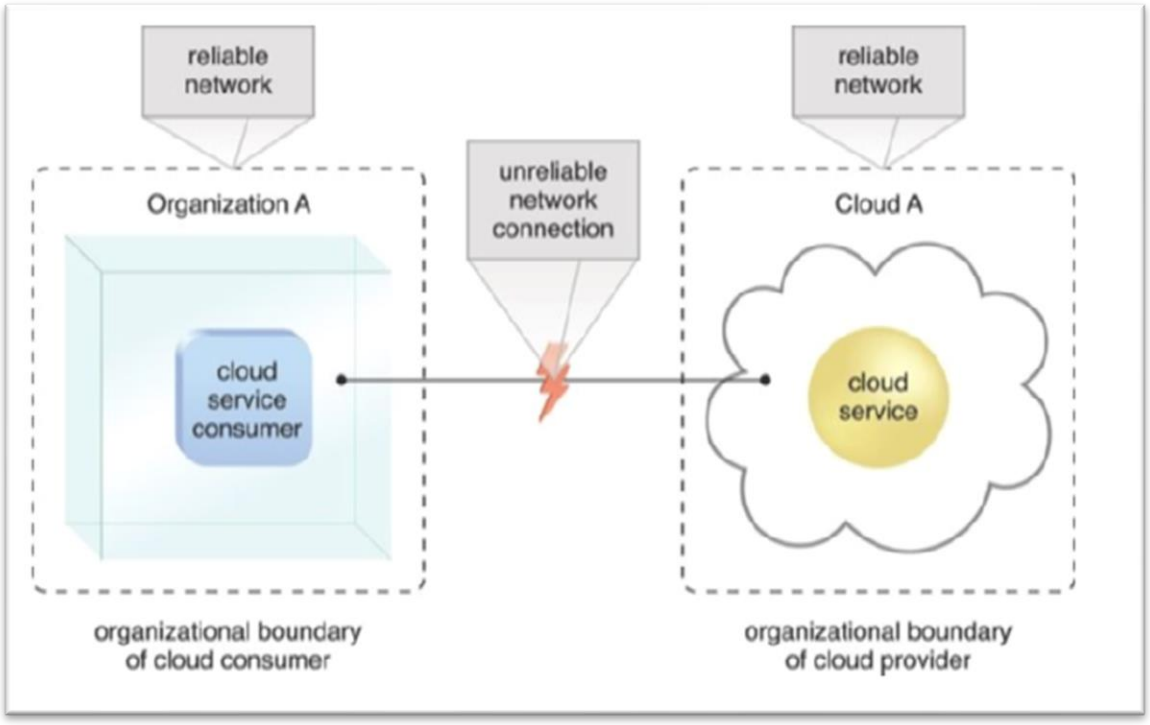
Moving business data to the cloud means that responsibility for data security becomes shared with the cloud provider. The remote usage of IT resources requires an expansion of trust boundaries by the cloud consumer to include the external cloud. However, establishing a security architecture that spans such a trust boundary can be challenging without introducing vulnerabilities. This is especially true unless cloud consumers and cloud providers happen to support the same or compatible security frameworks, which is unlikely with public clouds.

Another consequence of overlapping trust boundaries relates to the cloud provider's privileged access to cloud consumer data. The extent to which the data is secure is now limited to the security controls and policies applied by both the cloud consumer and the cloud provider. Furthermore, as cloud-based IT resources are commonly shared, there can be overlapping trust boundaries between different cloud consumers.



The shaded area

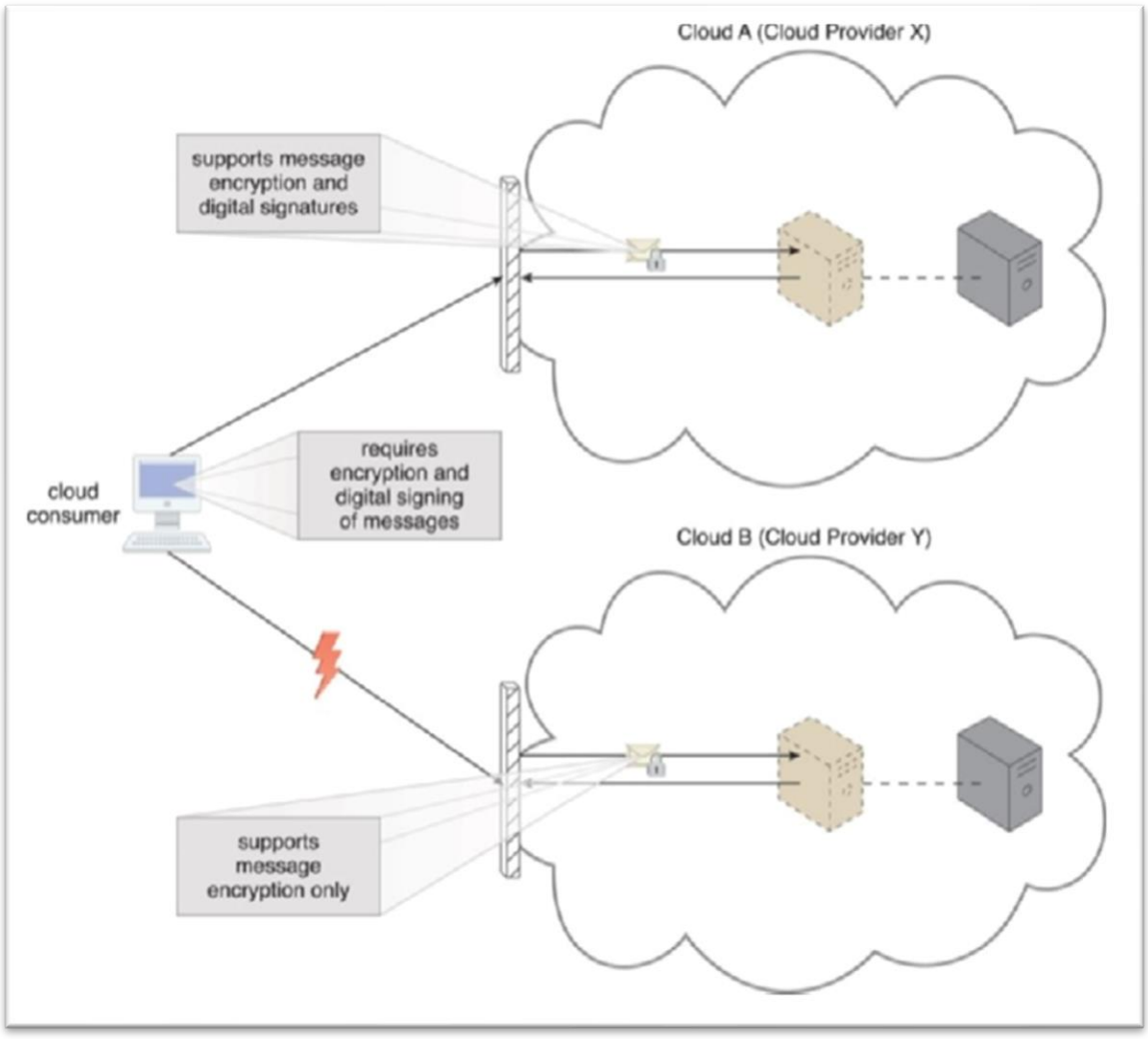
The overlapping of trust boundaries and the increased exposure of data can provide malicious cloud consumers, both human and automated, with greater opportunities to attack IT resources and steal or damage business data. The figure illustrates a scenario whereby two organizations accessing the same cloud service are required to extend their respective trust boundaries to the cloud, resulting in overlapping trust boundaries.



An unreliable network

It can be challenging for the cloud provider to offer security mechanisms that accommodate different security frameworks and diverse trust boundaries, which may lead to potential security breaches.

Therefore, it is essential for cloud consumers to carefully evaluate and select cloud providers that offer robust security features and adhere to stringent security standards.



Cloud A – Clod B Provides X/Y

The cloud computing industry lacks established industry standards, making public clouds proprietary to varying extents. As a result, custom-built solutions with dependencies on these proprietary environments can pose challenges when moving from one cloud provider to another. This challenge is known as portability, which measures the impact of moving IT resources and data between clouds.

- To address multi-regional compliance and legal issues, third-party cloud providers typically establish data centers in affordable or convenient geographical locations. However, cloud consumers may not be aware of the physical location of their IT resources and data when hosted by public clouds. This can pose serious legal concerns for organizations subject to industry or government regulations that specify data privacy and storage policies. For instance, UK laws require personal data belonging to UK citizens to be kept within the United Kingdom.
- Another potential legal issue is the accessibility and disclosure of data. Countries have laws that require certain types of data to be disclosed to certain government agencies or the subject of the data. For instance, a European cloud consumer's data located in the U.S. can be more easily accessed by government agencies due to the U.S. Patriot Act compared to data located in many European Union countries.

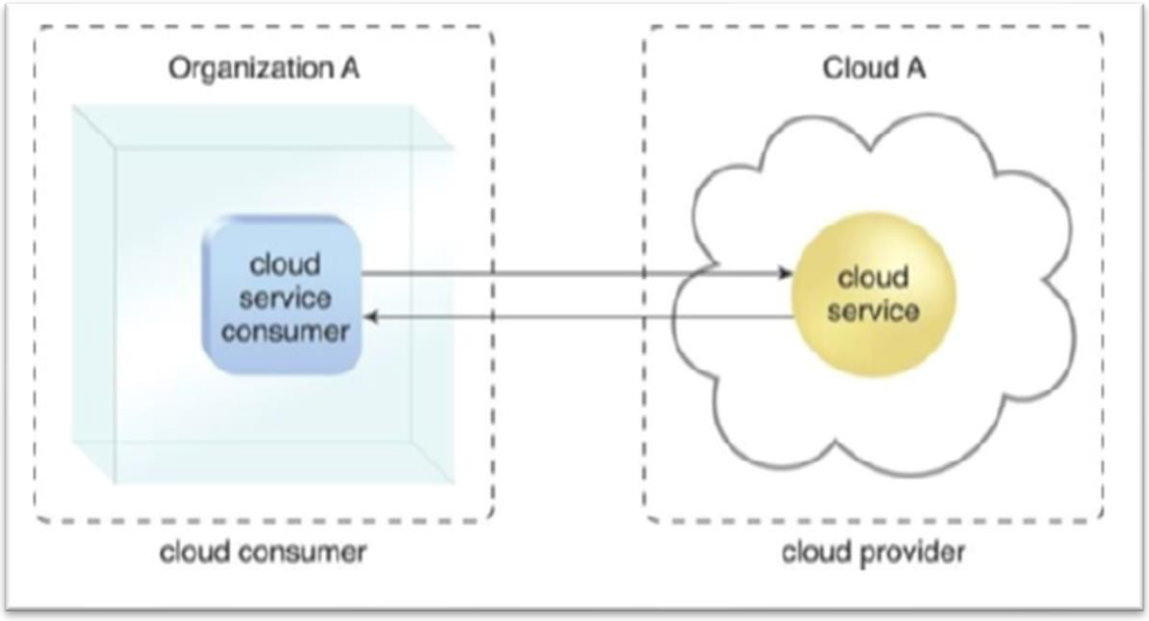
Although most regulatory frameworks acknowledge that cloud consumer organizations are ultimately responsible for the security, integrity, and storage of their own data, even when held by an external cloud provider, it is crucial for cloud consumers to be aware of the legal and compliance issues associated with using public clouds.

Fundamental Concepts and Models

In this section, we will delve into the fundamental models used to categorize and define clouds, as well as their most common service offerings. We will also explore the different organizational roles and their specific characteristics that collectively distinguish a cloud.

The first organizational role we will discuss is the Cloud Provider. This is the organization that provides cloud-based IT resources to its consumers, according to agreed-upon SLA guarantees. The Cloud Provider is responsible for making cloud services available to its consumers and carrying out any required management and administrative duties to ensure the ongoing operation of the overall cloud infrastructure.

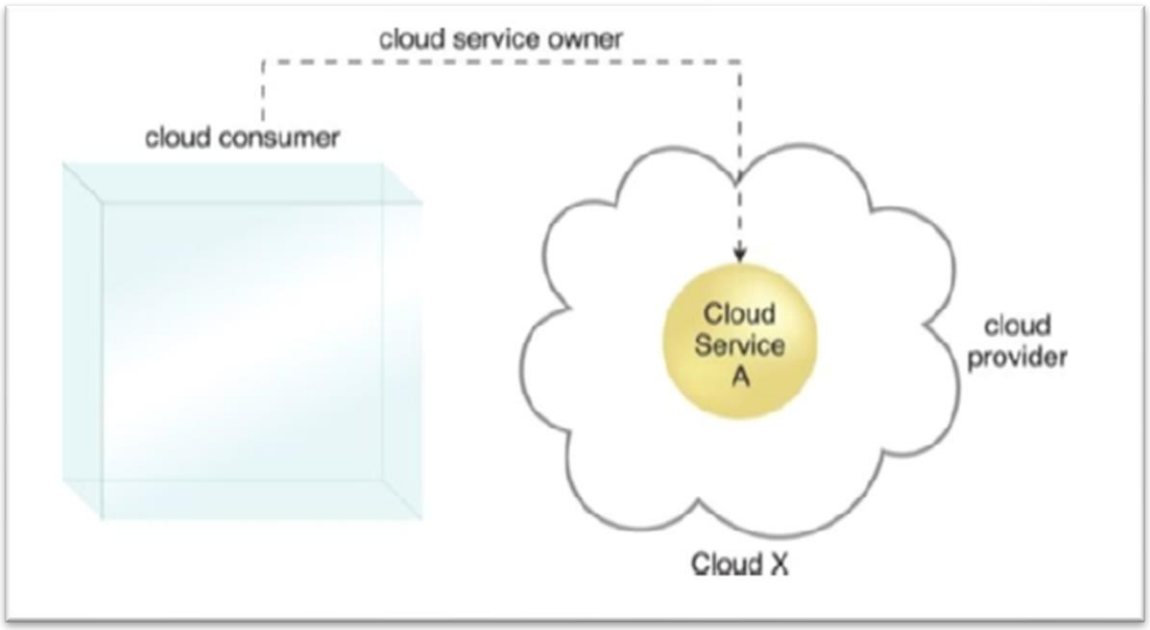
Cloud Providers typically own the IT resources that they make available for lease by Cloud Consumers. However, some Cloud Providers also "resell" IT resources leased from other Cloud Providers.



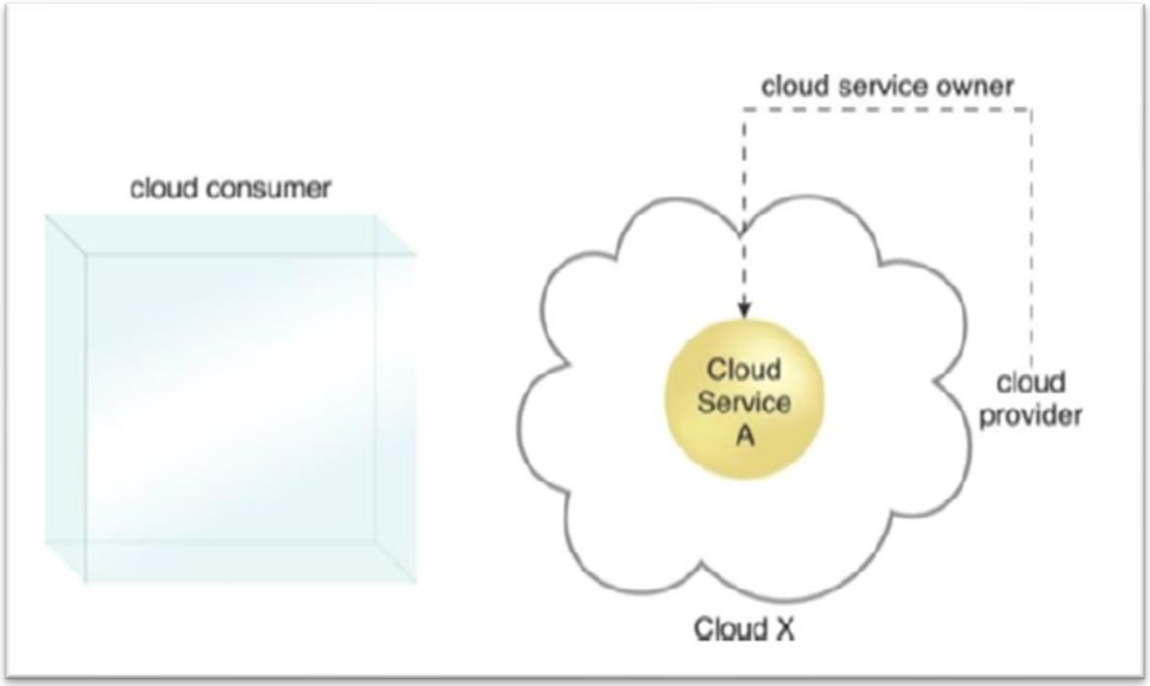
Within Organization A

The Cloud Consumer is the organization or individual that has a formal contract or arrangement with a Cloud Provider to use the IT resources made available by the Cloud Provider.

They access a Cloud Service Consumer to access a Cloud Service. Please note that the figures in this book may not always explicitly label symbols as "Cloud Consumers." Instead, it is generally implied that organizations or individuals remotely accessing cloud-based IT resources are considered Cloud Consumers.

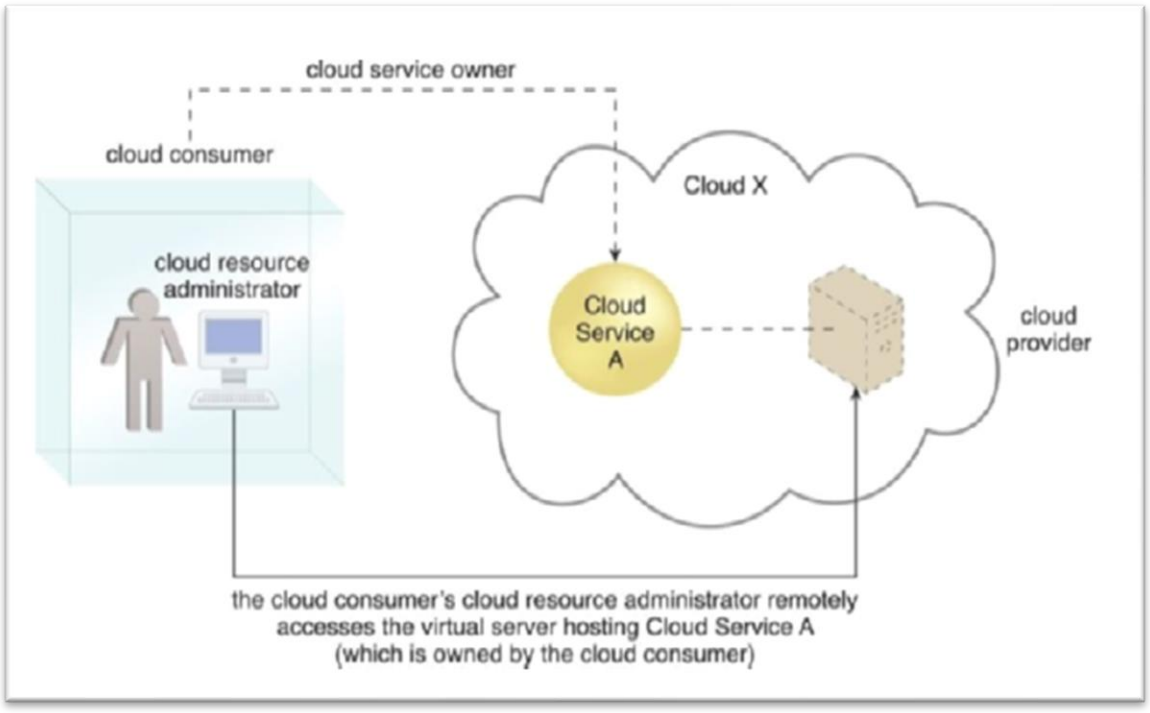


Cloud Consumer

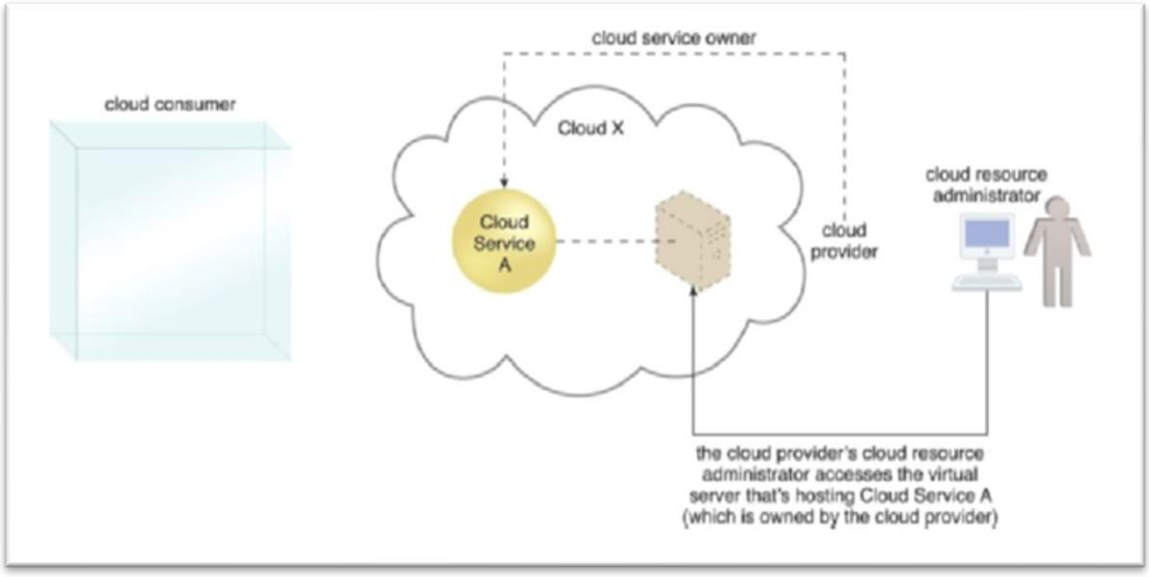


Cloud Provider

The Cloud Service Owner is the person or organization that legally owns a cloud service. It can either be the Cloud Consumer or the Cloud Provider that owns the cloud within which the Cloud Service resides. For example, either the Cloud Consumer or the Cloud Provider of Cloud X could own Cloud Service A.



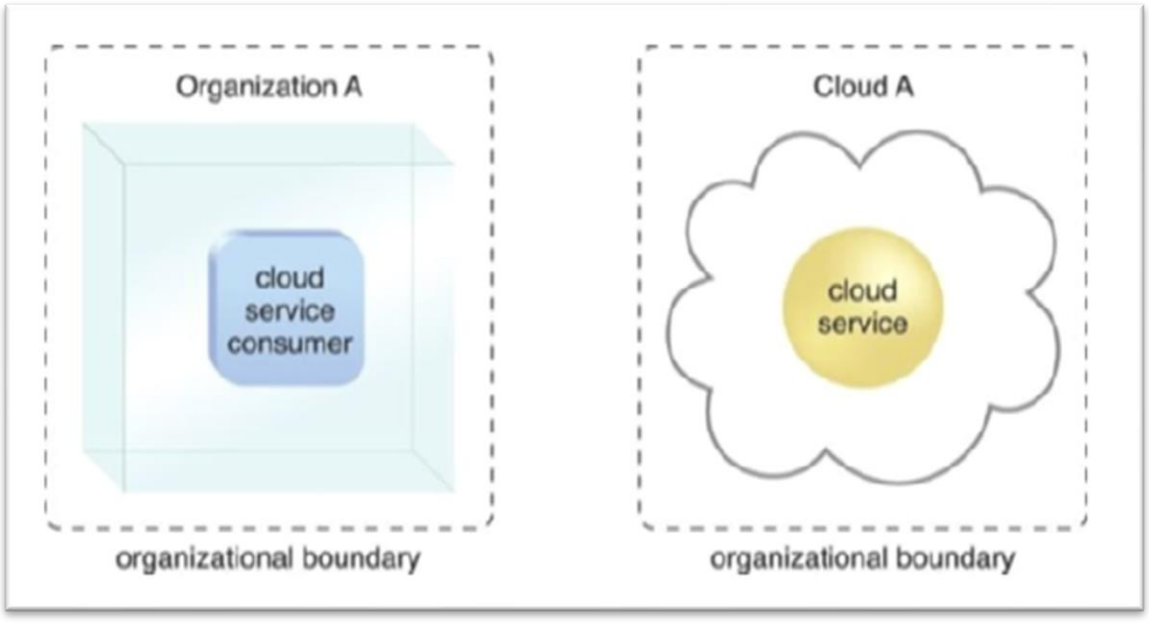
Cloud Administrator



A cloud Resource Administrator

It is important to note that a Cloud Consumer that owns a cloud service hosted by a third-party cloud does not necessarily need to be the user or consumer of the cloud service. Several Cloud Consumer organizations

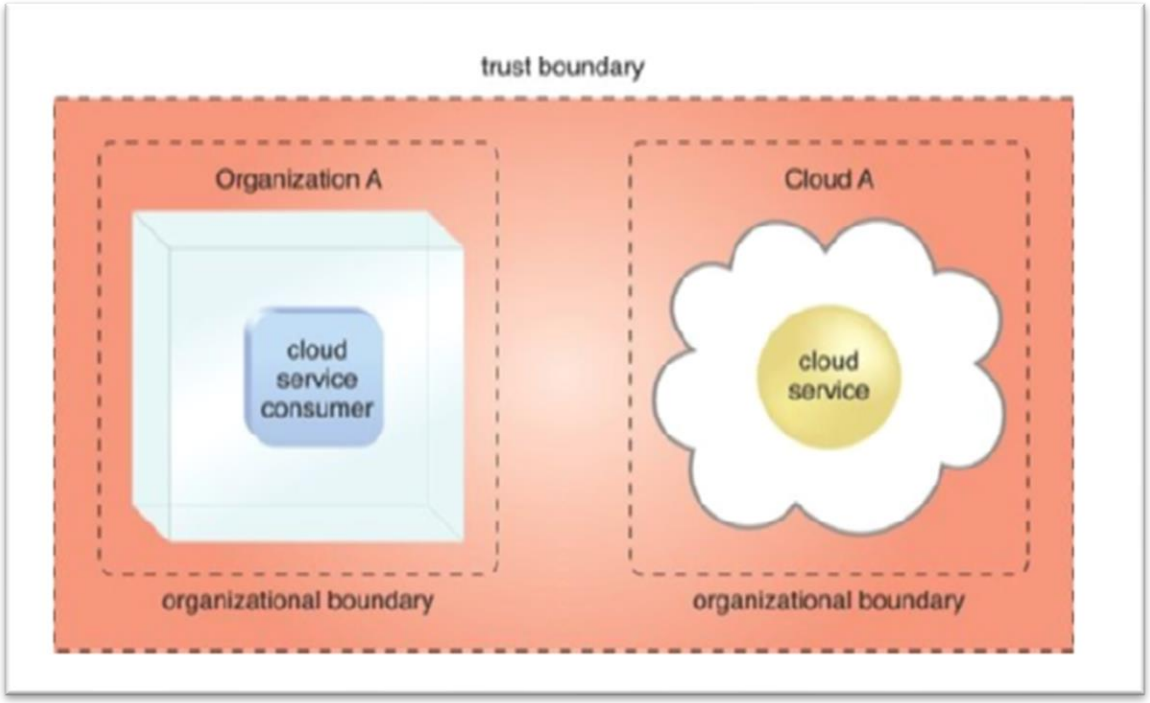
develop and deploy cloud services in clouds owned by other parties for the purpose of making the cloud services available to the public.



Organizational boundaries

The Cloud Resource Administrator is the person or organization responsible for administering a cloud-based IT resource, including cloud services. The Cloud Resource Administrator can belong to the Cloud Consumer or Cloud Provider of the cloud within which the Cloud Service resides.

Alternatively, it can belong to a third-party organization contracted to administer the cloud-based IT resource.



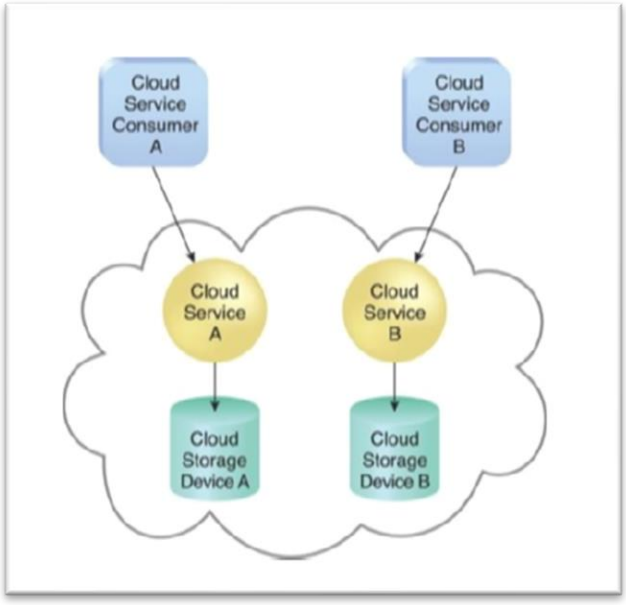
An extended trust boundary

The NIST Cloud Computing Reference Architecture also defines the Cloud Auditor as a third-party, often accredited, that conducts independent assessments of cloud security and compliance. This role is responsible for evaluating the effectiveness of cloud security measures and ensuring that the cloud services comply with regulatory requirements.

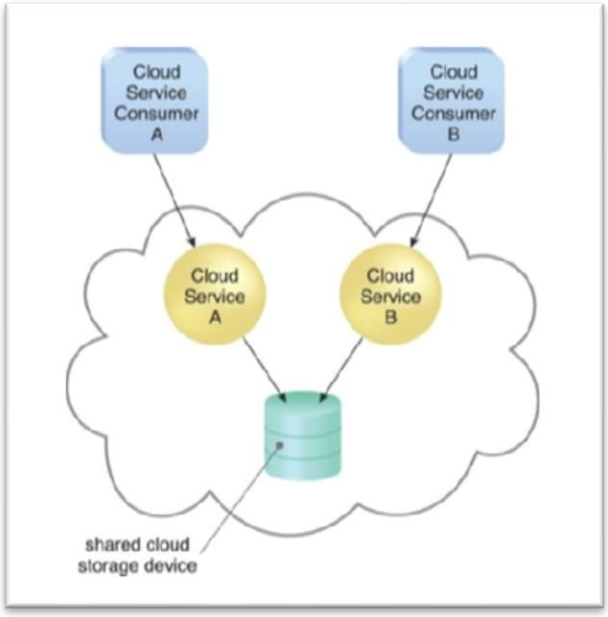
On-demand usage allows cloud consumers to access cloud-based IT resources unilaterally, enabling the freedom to self-provision these resources. Once configured, the usage of these self-provisioned IT resources can be automated, requiring no further human involvement from the cloud consumer or provider. This results in an on-demand usage environment and facilitates service-based and usage-driven features.

Ubiquitous access represents the ability of a cloud service to be widely accessible. To enable this level of access, the cloud service architecture needs to be tailored to the specific needs of different cloud service consumers, supporting a range of devices, transport protocols, interfaces, and security technologies.

Multitenancy and resource pooling are critical characteristics of a cloud environment. Multitenancy enables a software program to serve different consumers (tenants), where each tenant is isolated from the others. Cloud providers pool their IT resources to serve multiple cloud service consumers using multitenancy models that often rely on virtualization technologies. Resource pooling allows cloud providers to pool large-scale IT resources to serve multiple cloud consumers, and different physical and virtual IT resources are dynamically assigned and reassigned based on cloud consumer demand.

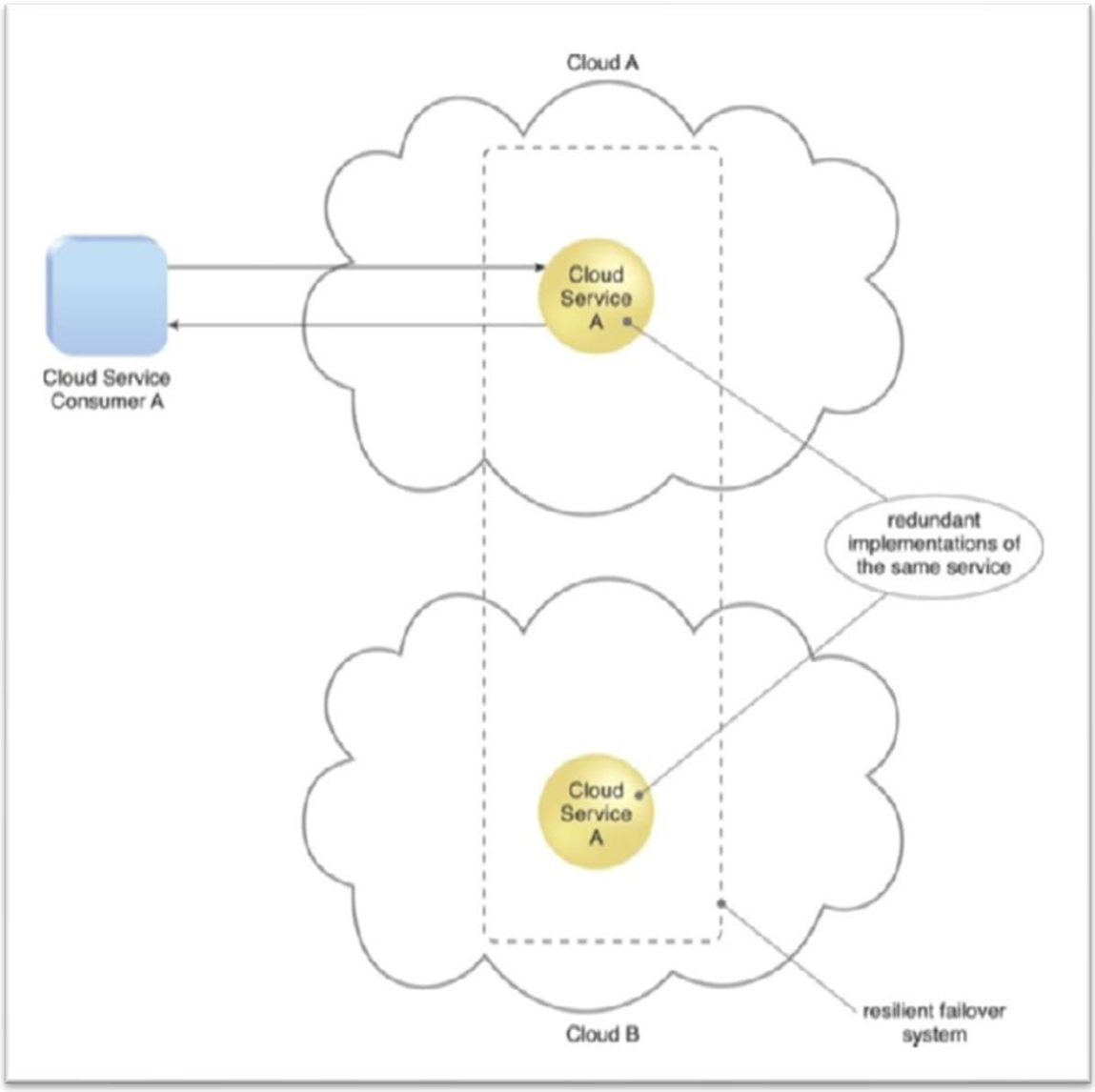


In a single-tenant environment



In a multitenant environment

Elasticity is the automated ability of a cloud to transparently scale IT resources as required in response to runtime conditions or as predetermined by the cloud consumer or provider. Elasticity is closely associated with Reduced Investment and Proportional Costs benefit and is often considered a core justification for cloud computing.

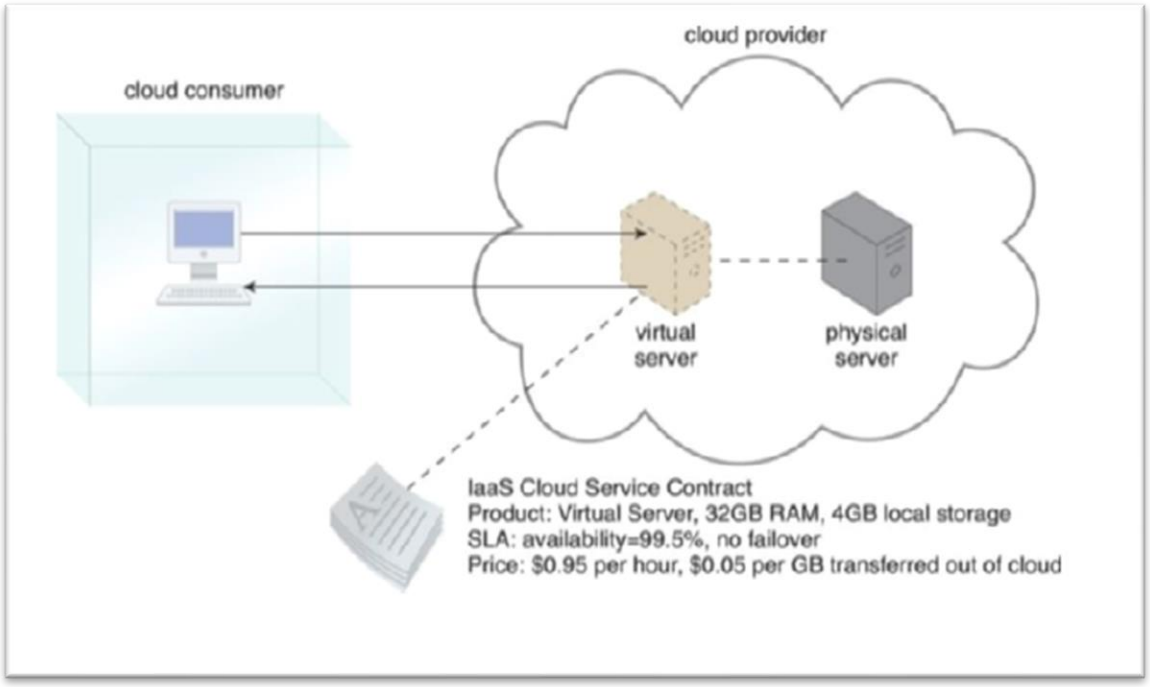


A resilient system in which Cloud B hosts

Measured usage is the ability of a cloud platform to track the usage of its IT resources primarily by cloud consumers. Based on what is measured, the cloud provider can charge a cloud consumer only for the IT resources used and/or for the timeframe during which access to the IT resources was granted. Measured usage is closely related to the on-demand characteristic and is not limited to tracking statistics for billing purposes but also enables cloud consumers to monitor their usage and optimize their resource consumption.

Infrastructure-as-a-Service (IaaS) provides a complete IT environment consisting of IT resources that are infrastructure-centric and can be accessed and managed via cloud service-based interfaces and tools. This

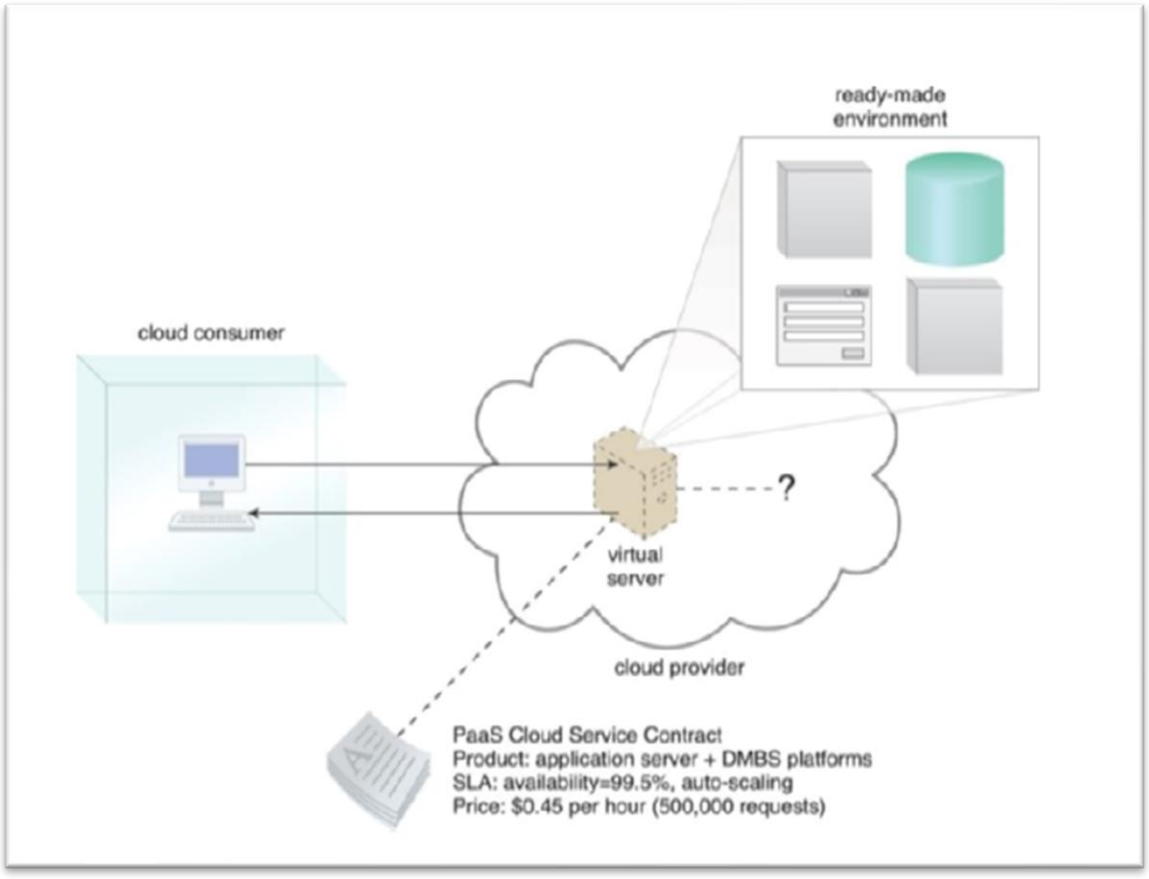
environment comprises hardware, network, connectivity, operating systems, and other raw IT resources that are virtualized and bundled to simplify runtime scaling and infrastructure customization. The main aim of the IaaS environment is to offer cloud consumers a high level of control and responsibility over the configuration and utilization of IT resources, which are usually not pre-configured. Therefore, cloud consumers who require a high level of control over their cloud-based environment prefer IaaS. The IT resources provided by IaaS can vary based on the type and brand of the IaaS products offered by different cloud providers. The central IT resource in a typical IaaS environment is the virtual server that is leased based on the specifications of server hardware requirements, such as processor capacity, memory, and local storage space.



A cloud consumer

Platform-as-a-Service (PaaS) is a pre-defined ready-to-use environment consisting of already deployed and configured IT resources. PaaS relies on a ready-made environment that establishes a set of pre-packaged products and tools used to support the entire delivery lifecycle of custom applications.

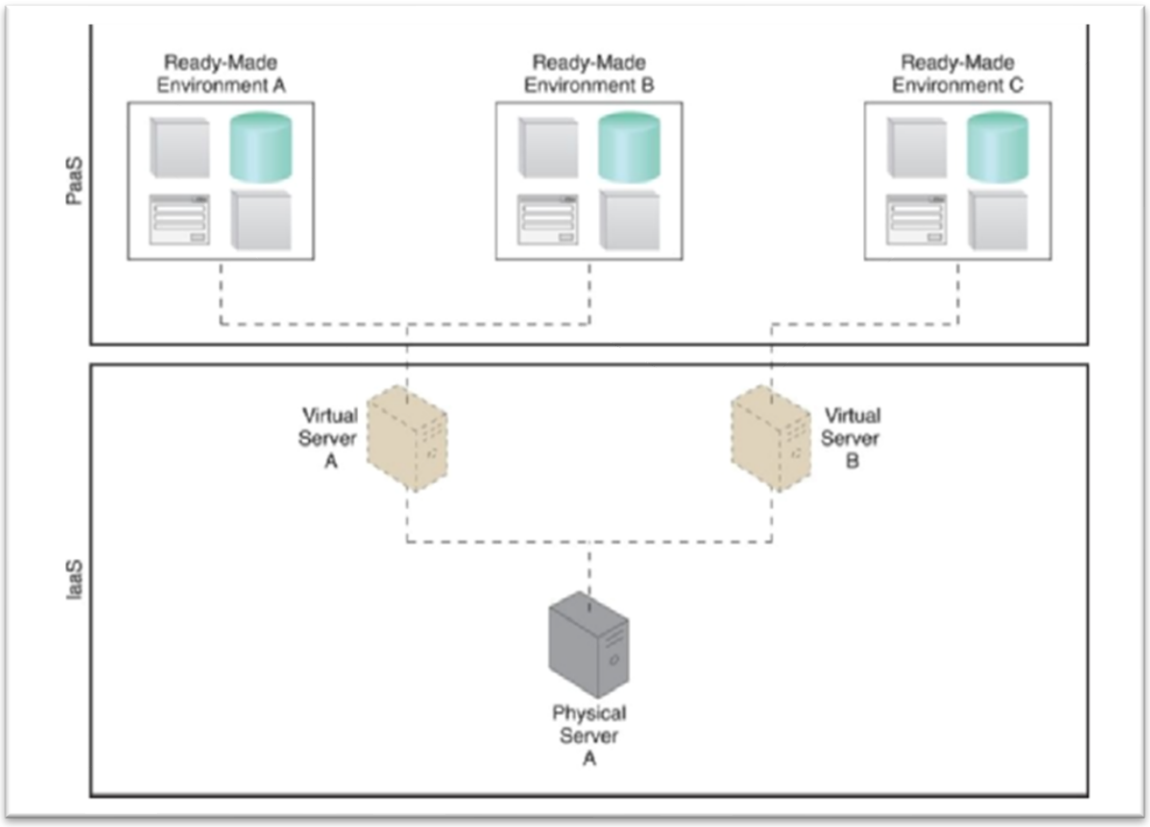
Cloud consumers prefer using and investing in a PaaS environment if they want to extend on-premise environments into the cloud for scalability and economic purposes or if they want to entirely substitute an on-premise environment or become a cloud provider and deploy their own cloud services to be made available to other external cloud consumers.



A cloud consumer

Working within a ready-made platform spares the cloud consumer from the administrative burden of setting up and maintaining the bare infrastructure IT resources provided via the IaaS model.

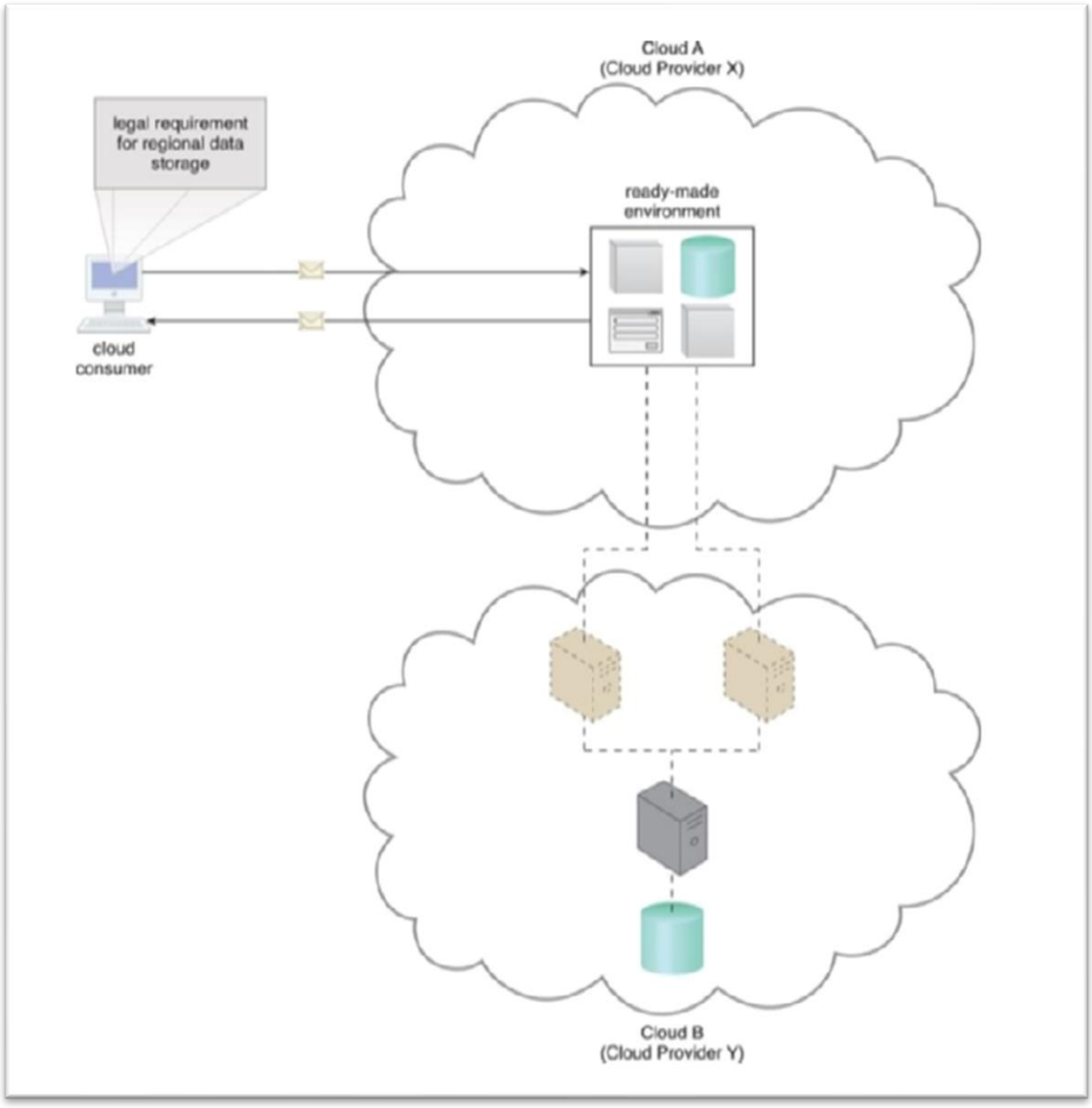
However, the cloud consumer is granted a lower level of control over the underlying IT resources that host and provision the platform. PaaS products are available with different development stacks, such as Google App Engine offering a Java and Python-based environment.



A PaaS environment

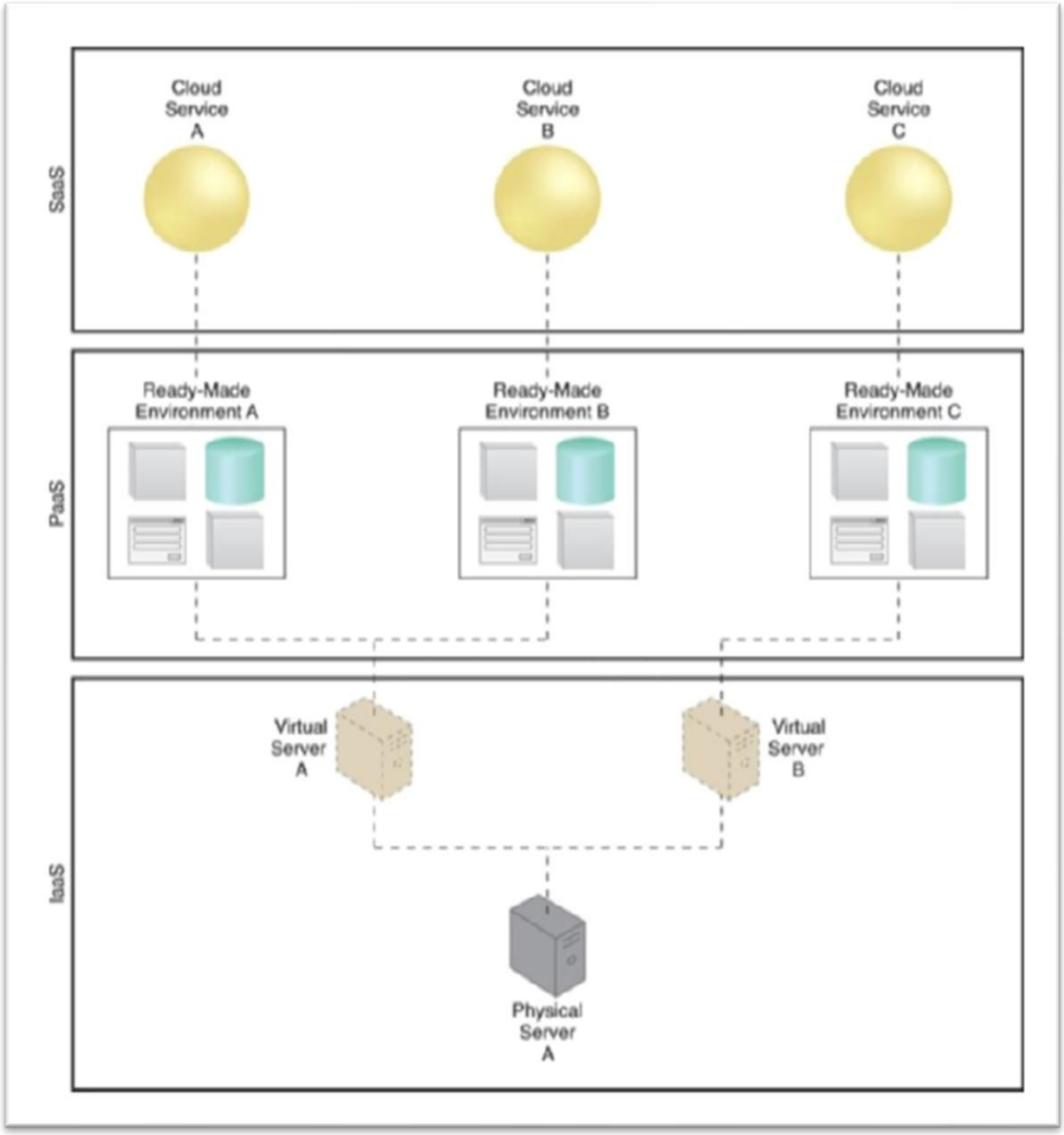
Software-as-a-Service (SaaS) is a software program provided as a shared cloud service and made available as a product or generic utility. The SaaS delivery model is used to make a reusable cloud service widely available to a range of cloud consumers.

A cloud consumer is generally granted very limited administrative control over a SaaS implementation, which is most often provisioned by the cloud provider but can be legally owned by any entity that assumes the cloud implementation's financial risk.



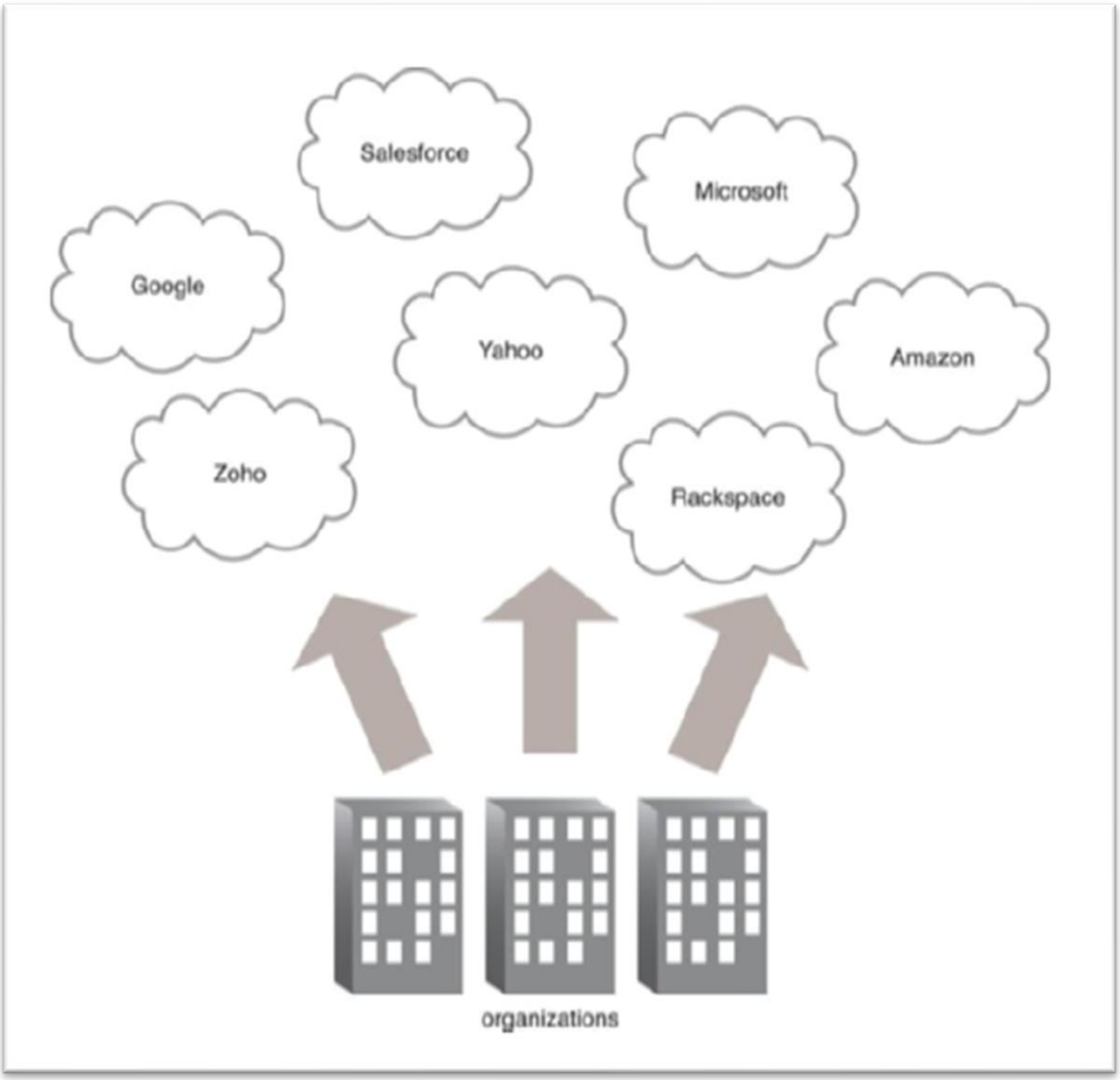
Cloud Providers X and Y

The three cloud delivery models, namely IaaS, PaaS, and SaaS, can be combined to create a layered IT resource architecture.



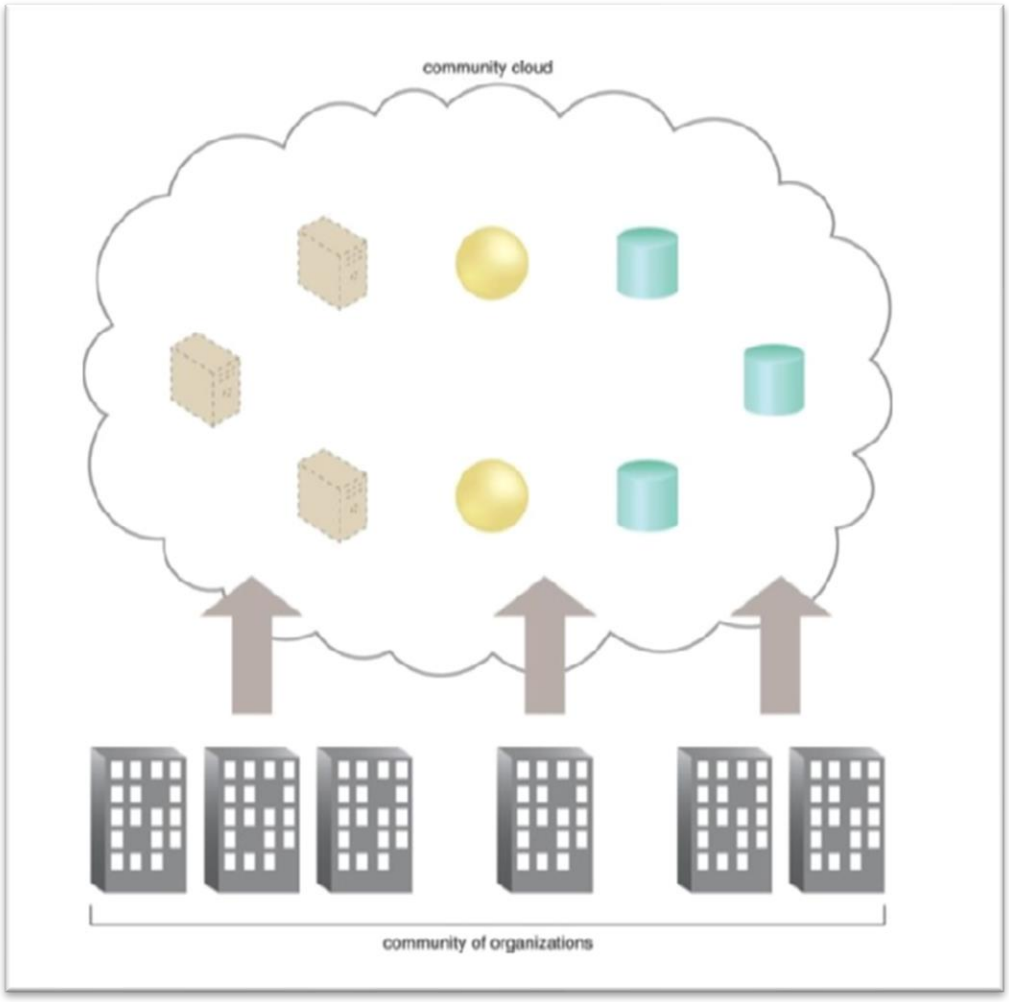
A simple layered view

The PaaS environment can be used to develop and deploy SaaS cloud services, which can be made available as commercial products by the cloud consumer organization. Four common cloud deployment models are Public cloud, Community cloud, Private cloud, and Hybrid cloud. A public cloud is a publicly accessible cloud environment owned by a third-party cloud provider, while a community cloud is a limited access public cloud owned by a specific community of cloud consumers.

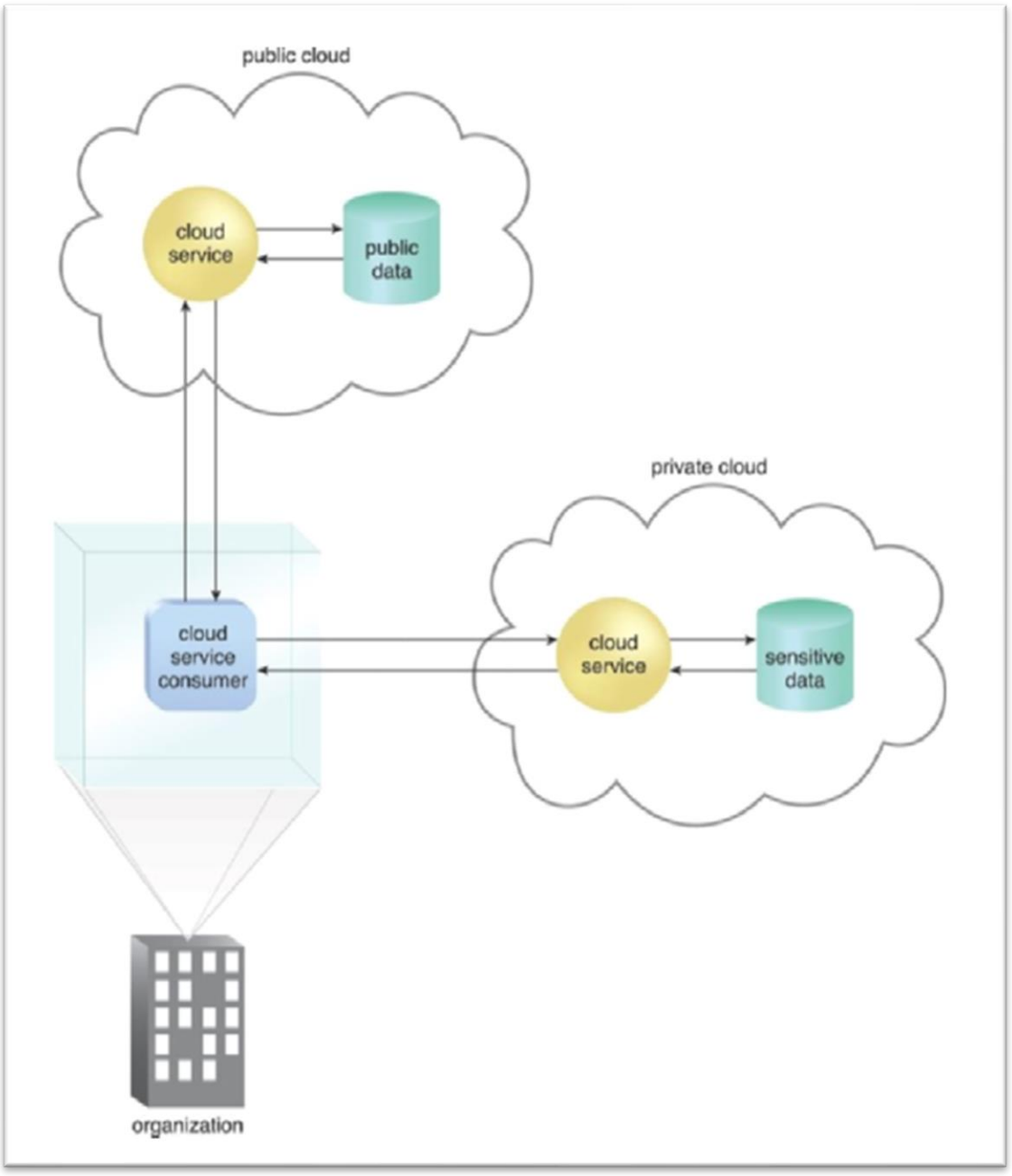


Organizations act as cloud consumers

A private cloud is owned by a single organization and enables centralized access to IT resources by different parts, locations, or departments of the organization. The organization itself assumes both the cloud consumer and cloud provider roles in a private cloud. It is important to use the terms "on-premise" and "cloud-based" correctly within the context of a private cloud.



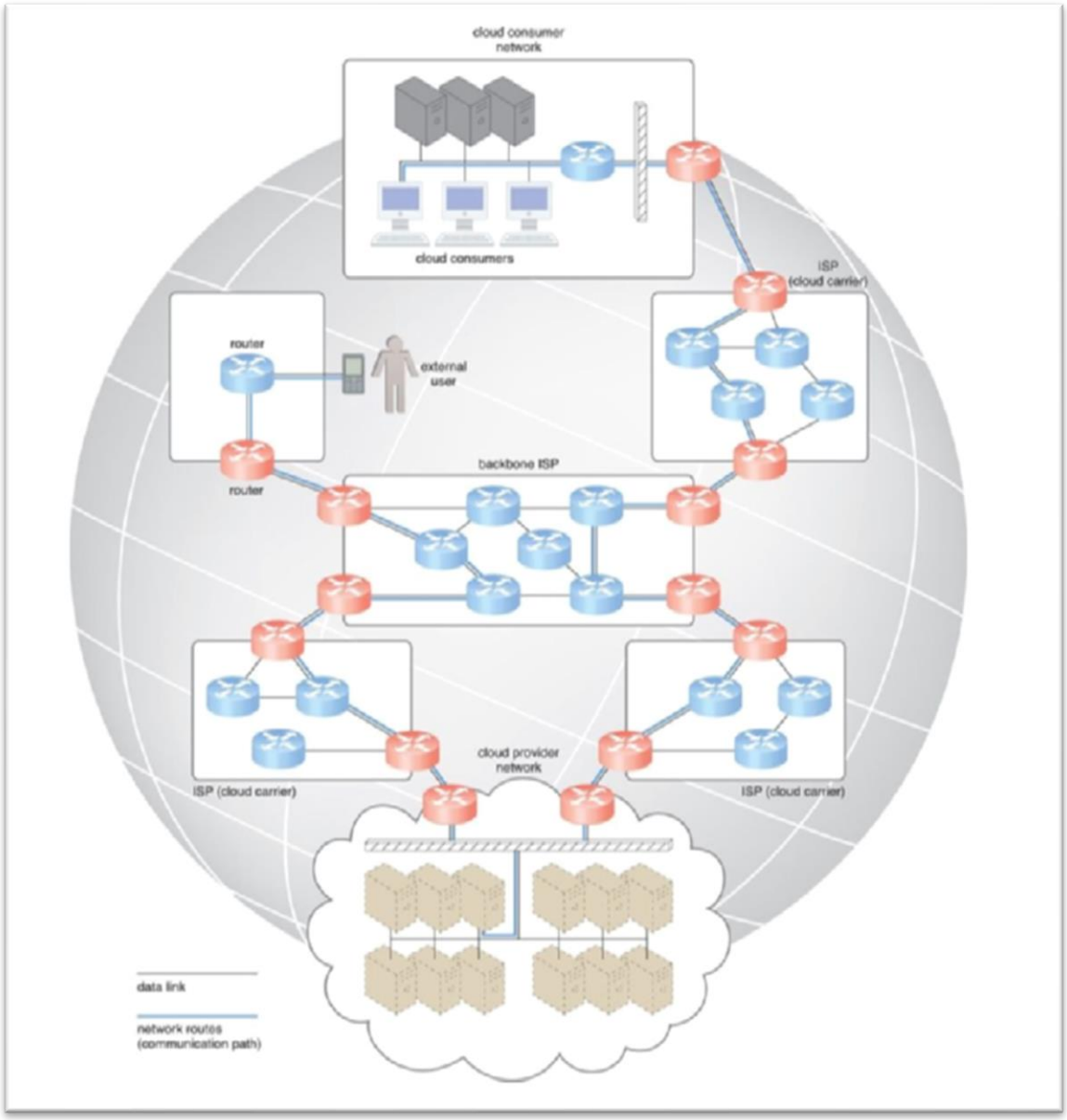
Community of organizations



Hybrid cloud architecture

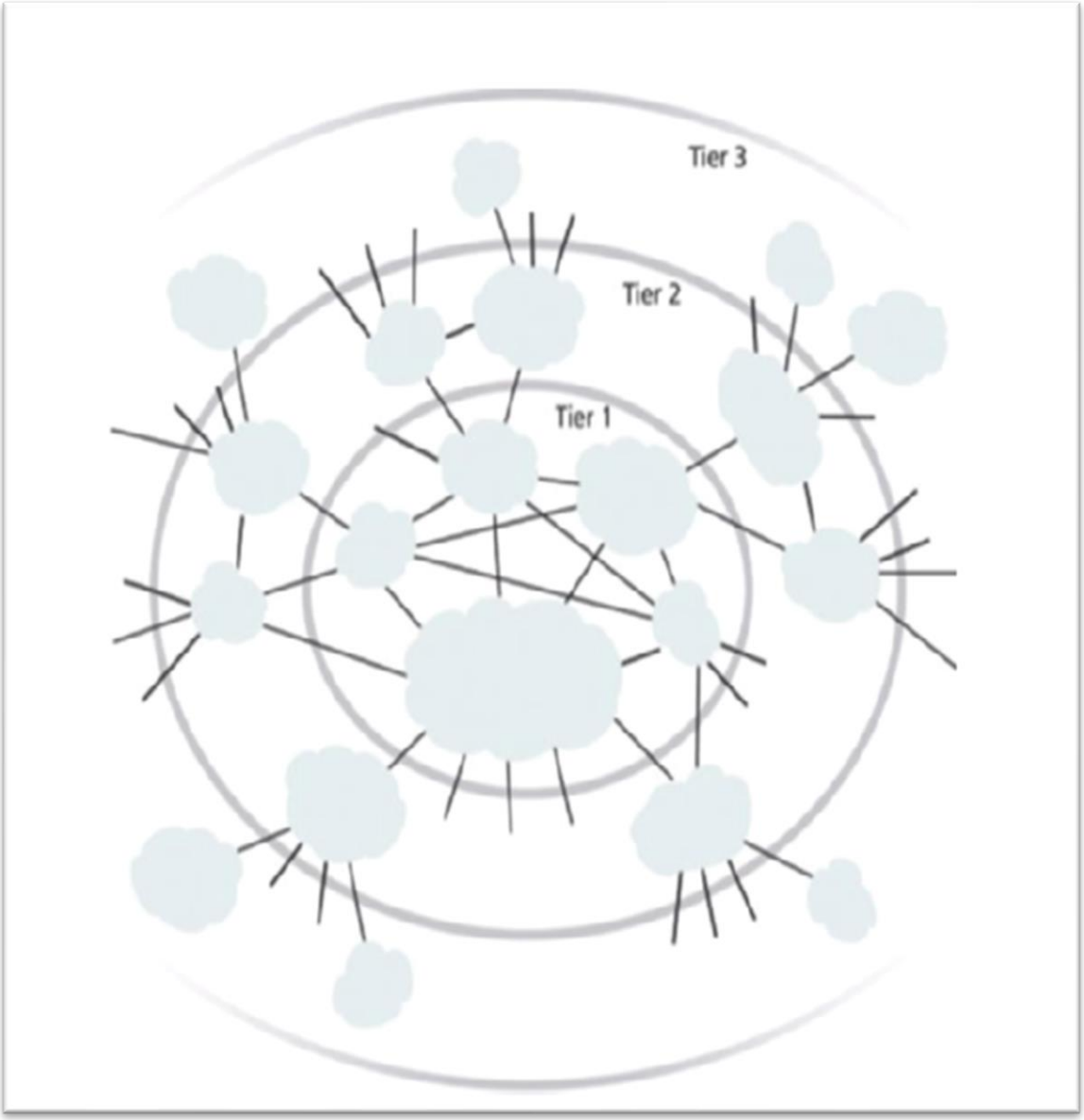
Cloud-Enabling Technology

This section covers the primary technology components that enable the key features and characteristics of contemporary cloud computing. These components include broadband networks and internet architecture, data center technology, virtualization technology, web technology, multi-tenant technology, and service technology. While these technologies existed and matured before the advent of cloud computing, cloud advancements helped further evolve select areas of these technologies.



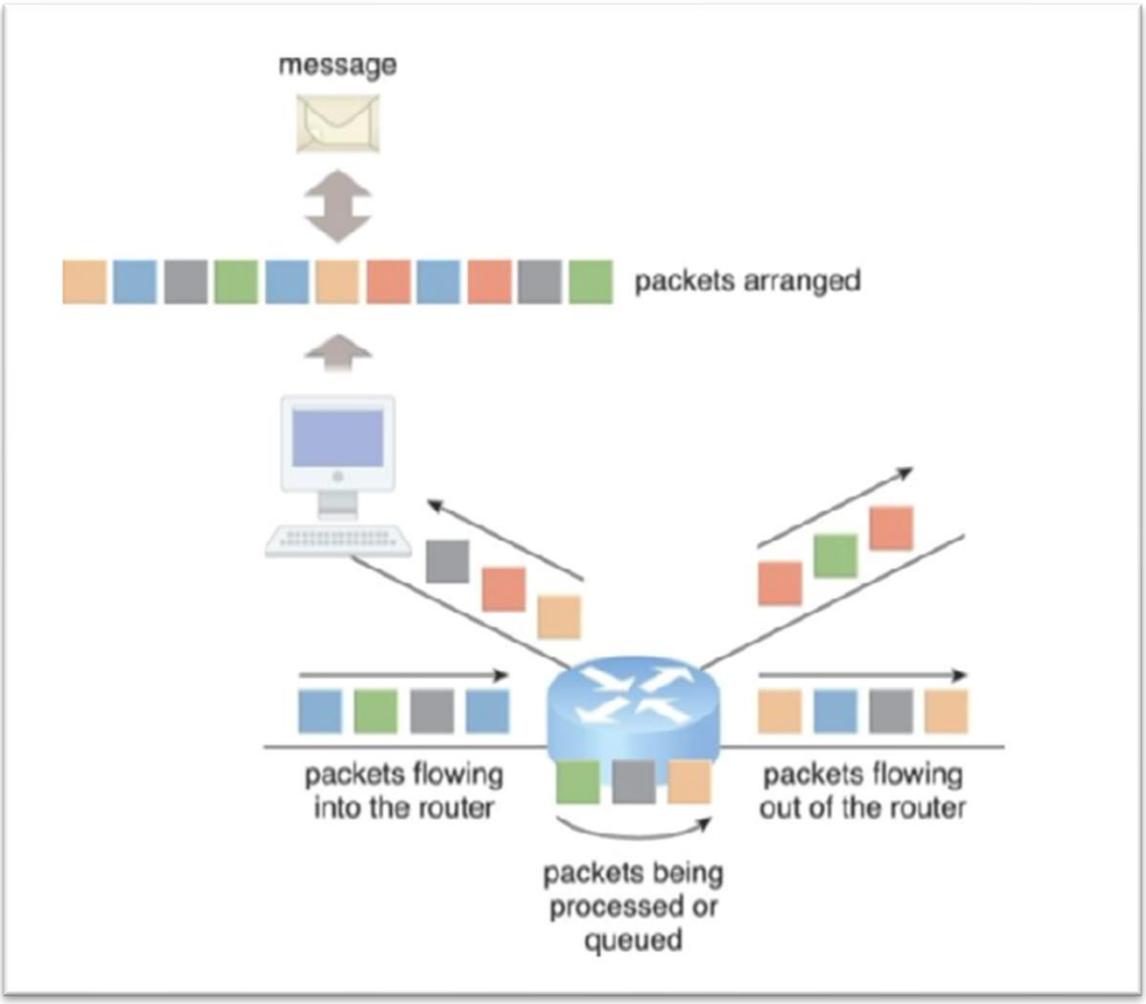
Messages travel over dynamic network routes

Clouds must be connected to a network, and the Internet is the most common option for this. Internet Service Providers (ISPs) establish and deploy the Internet's backbone networks that connect the world's multinational networks.



Abstraction of the inter-networking structure

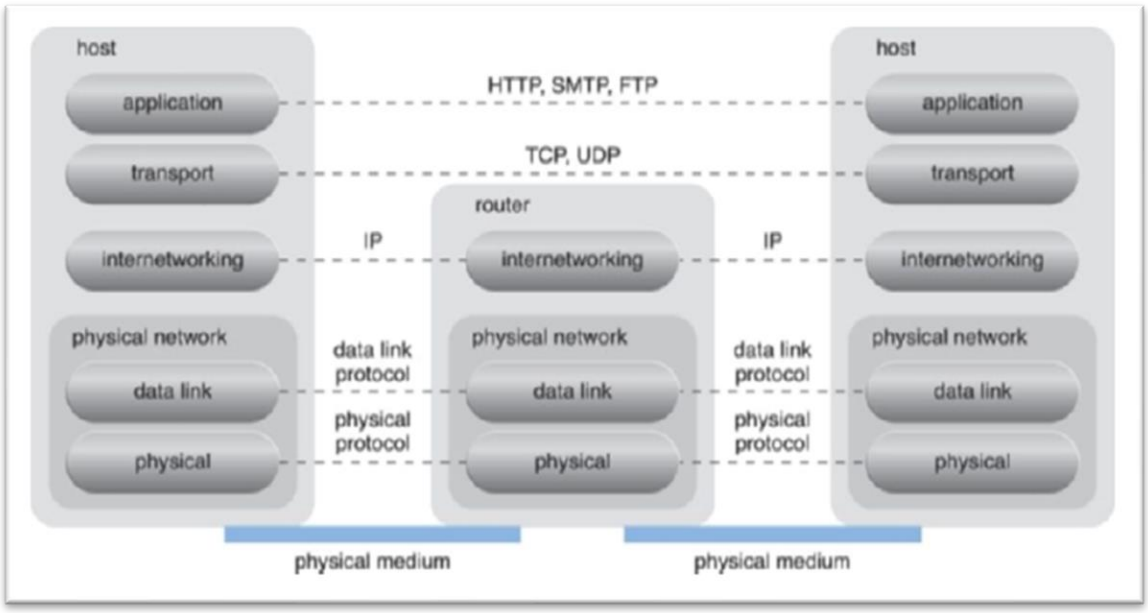
There is no centralized entity that governs the Internet, although bodies like the Internet Corporation for Assigned Names and Numbers (ICANN) supervise and coordinate Internet communications. The Internet's topology has become a dynamic and complex aggregate of ISPs that are highly interconnected via its core protocols.



Packets traveling through the Internet

Worldwide connectivity is enabled through a hierarchical topology composed of Tiers 1, 2, and 3. The core Tier 1 is made of large-scale, international cloud providers that oversee massive interconnected global networks, which are connected to Tier 2's large regional providers.

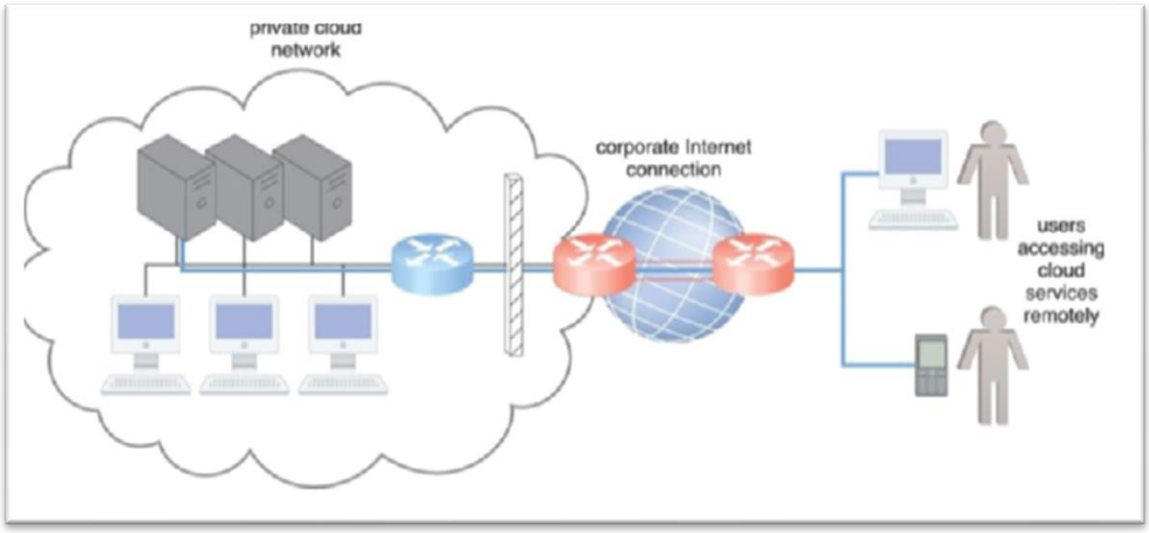
The interconnected ISPs of Tier 2 connect with Tier 1 providers, as well as the local ISPs of Tier 3.



A generic view of the Internet

The communication links and routers of the Internet and ISP networks are IT resources that are distributed among countless traffic generation paths. Connectionless packet switching and router-based inter-connectivity are the two fundamental components used to construct the inter-networking architecture. Connectionless packet switching divides data flows into packets that carry the necessary location information to be processed and routed at every source, intermediary, and destination node. A router is a device that forwards packets between networks and manages network traffic.

Connectivity is a crucial aspect to consider when it comes to traditional on-premise deployment models for enterprise applications and IT solutions, where centralized servers and storage devices are in the organization's own data center, and end-user devices access the data center through the corporate network via TCP/IP over LANs.

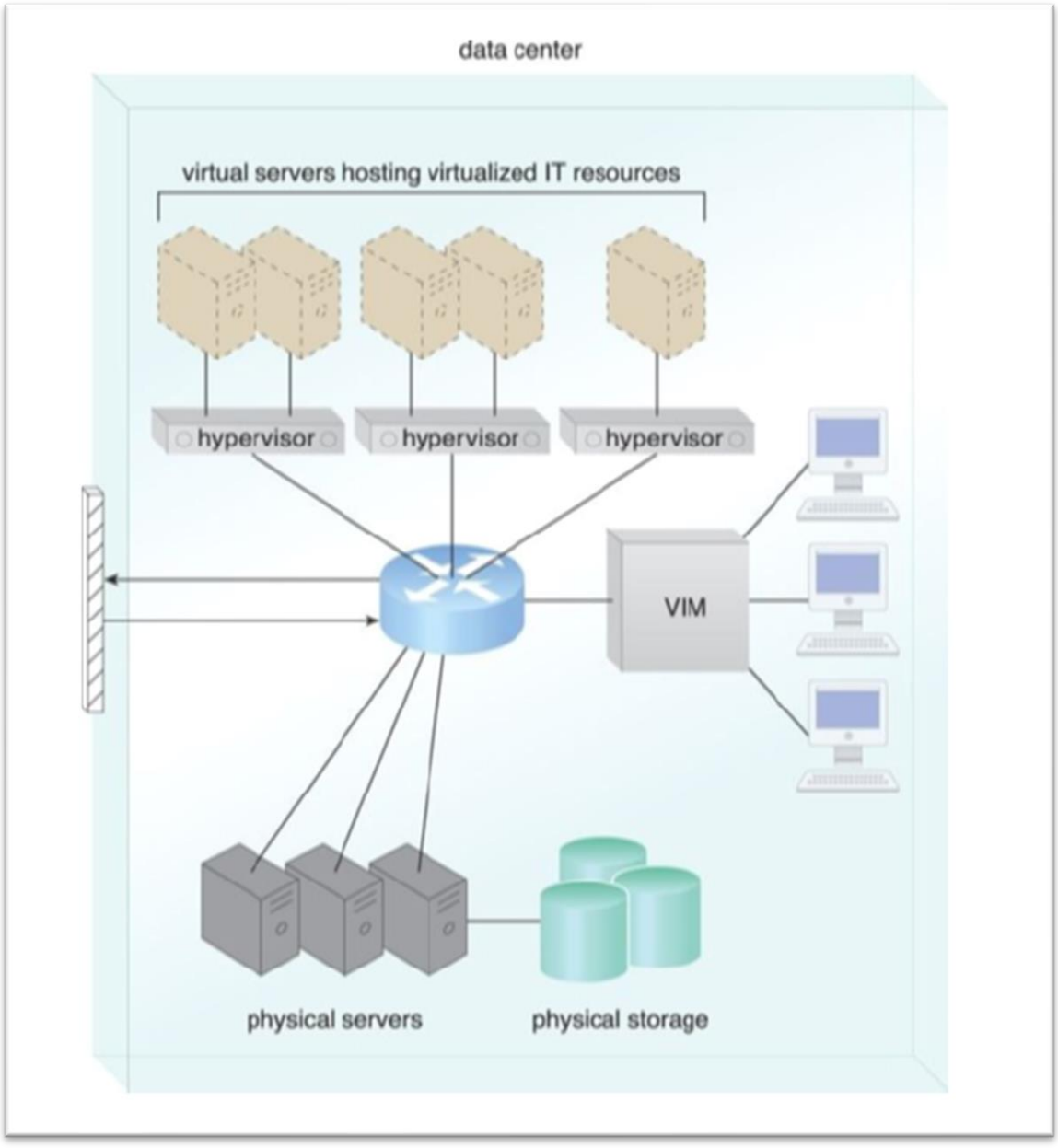


The inter-networking architecture

Although this model is not commonly referred to as a cloud model, it has been implemented numerous times for medium and large on-premise networks. This configuration allows organizations using this deployment model to have complete control over their corporate networks and safeguard them using firewalls and monitoring software.

In contrast, cloud providers can easily configure cloud-based IT resources to be accessible for both external and internal users through an Internet connection. This inter-networking architecture benefits internal users that require ubiquitous access to corporate IT solutions, as well as cloud consumers that need to provide Internet-based services to external users.

Another important aspect to consider is network bandwidth and latency, which are inherent to cloud interconnection. End-to-end bandwidth is determined by the transmission capacity of the shared data links that connect intermediary nodes, while latency is the amount of time it takes for a packet to travel from one data node to another.

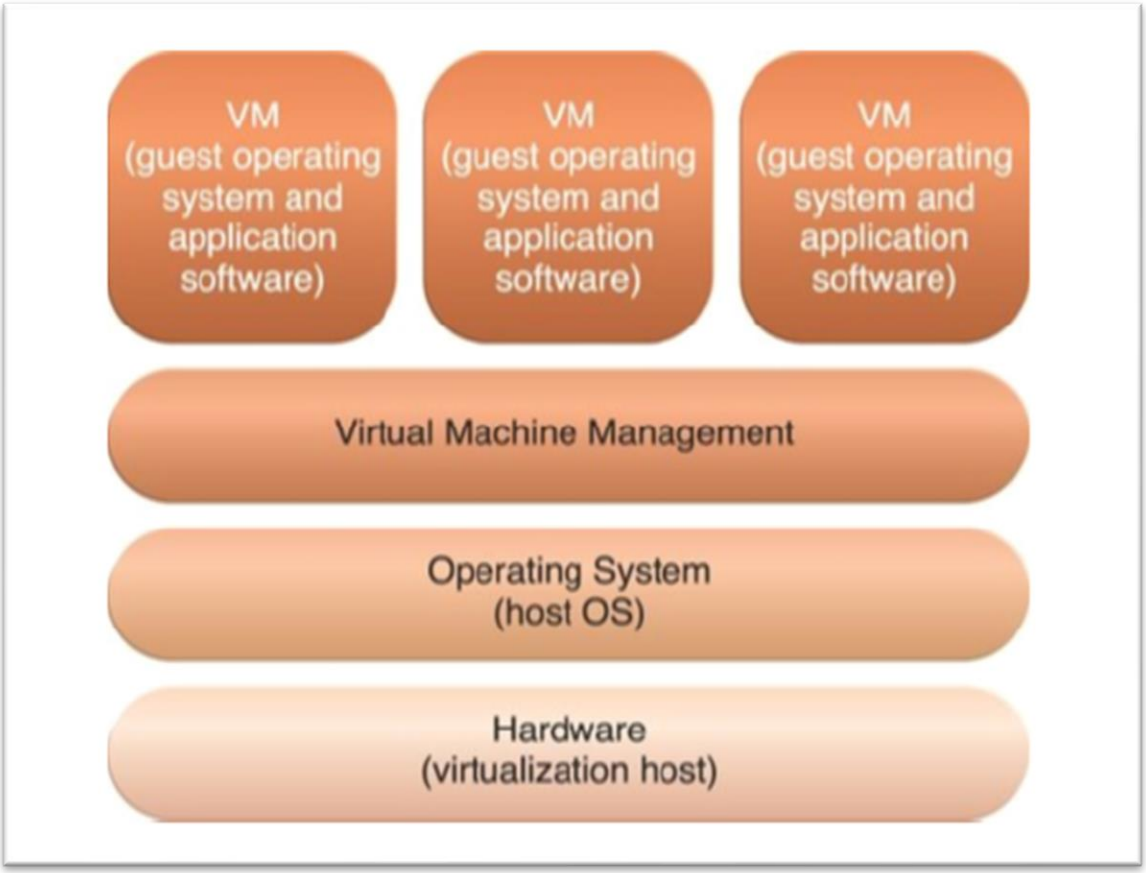


Data center

Latency increases with every intermediary node on the data packet's path and can be impacted by network congestion and traffic conditions in shared nodes. Therefore, it is essential to assess IT solutions against business requirements that are affected by network bandwidth and latency when considering cloud interconnection.

- Data centers are custom-designed facilities equipped with specialized computing, storage, and network equipment. These facilities have functional areas and environmental control stations that regulate heating, ventilation, air conditioning, fire protection, and other subsystems. Data centers have segregated spaces for common rooms and utilities.
- Computing hardware in data centers often involves standardized commodity servers with computing power and storage capacity, as well as different hardware processing architectures. Blade server technologies use rack-embedded physical interconnections, fabrics, shared power supply units, and cooling fans to optimize physical space and power. Storage hardware systems in data centers involve hard disk arrays, I/O caching, hot-swappable hard disks, storage virtualization, and fast data replication mechanisms.
- Data centers also use robotized tape libraries for backup and recovery systems that typically rely on removable media. Networked storage devices usually fall into one of two categories: Storage Area Network (SAN) and Network-Attached Storage (NAS). SAN provides block-level data storage access using industry-standard protocols, such as the Small Computer System Interface (SCSI), while NAS provides file-level data storage access.

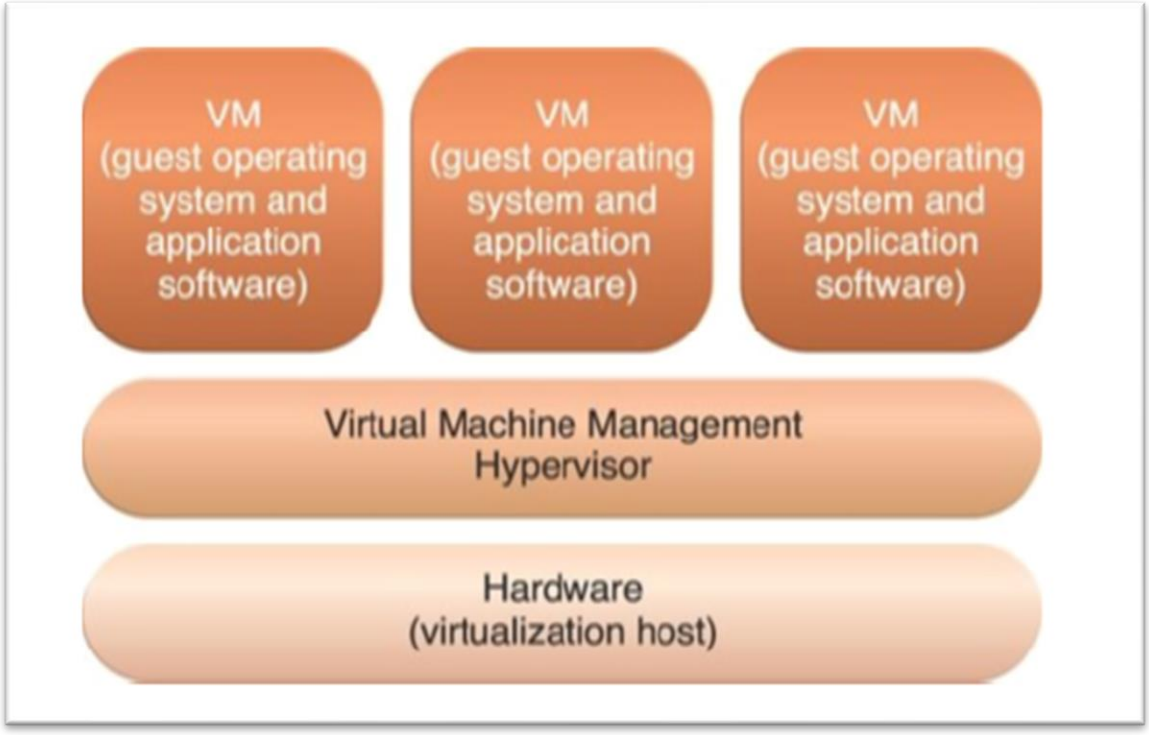
To create a new virtual server through virtualization software, the first step is the allocation of physical IT resources followed by the installation of an operating system. Virtual servers use their own guest operating systems, which are independent of the operating system in which they were created. The guest operating system and application software running on the virtual server are unaware of the virtualization process, meaning that these virtualized IT resources are installed and executed as if they were running on a separate physical server. This uniformity of execution that allows programs to run on physical systems as they would on virtual systems is a vital characteristic of virtualization.



Operating system-based virtualization

Virtualization software runs on a physical server called a host or physical host, whose underlying hardware is made accessible by the virtualization software. The virtualization software functionality encompasses system services that are specifically related to virtual machine management and are not normally found on standard operating systems.

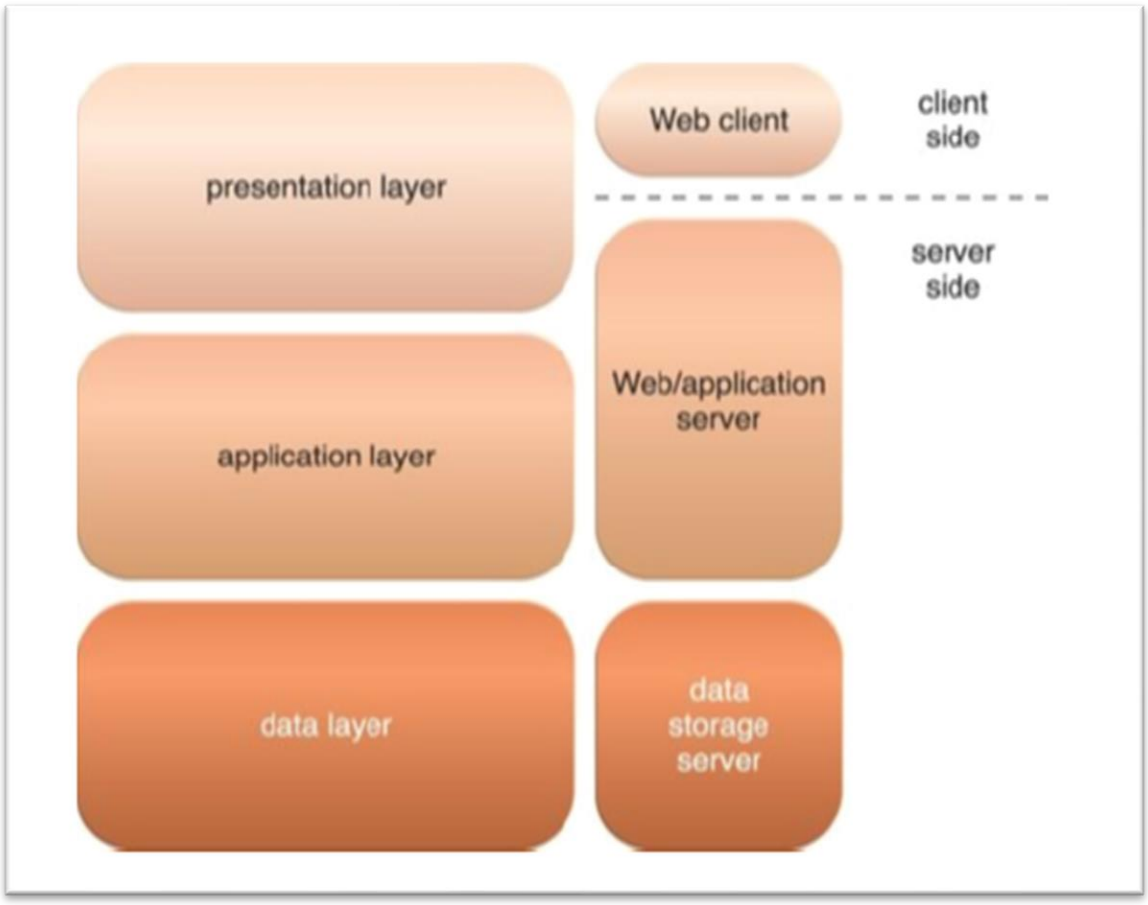
This is why this software is sometimes referred to as a virtual machine manager or a virtual machine monitor (VMM), but is best known as a hypervisor.



Hardware-based virtualization

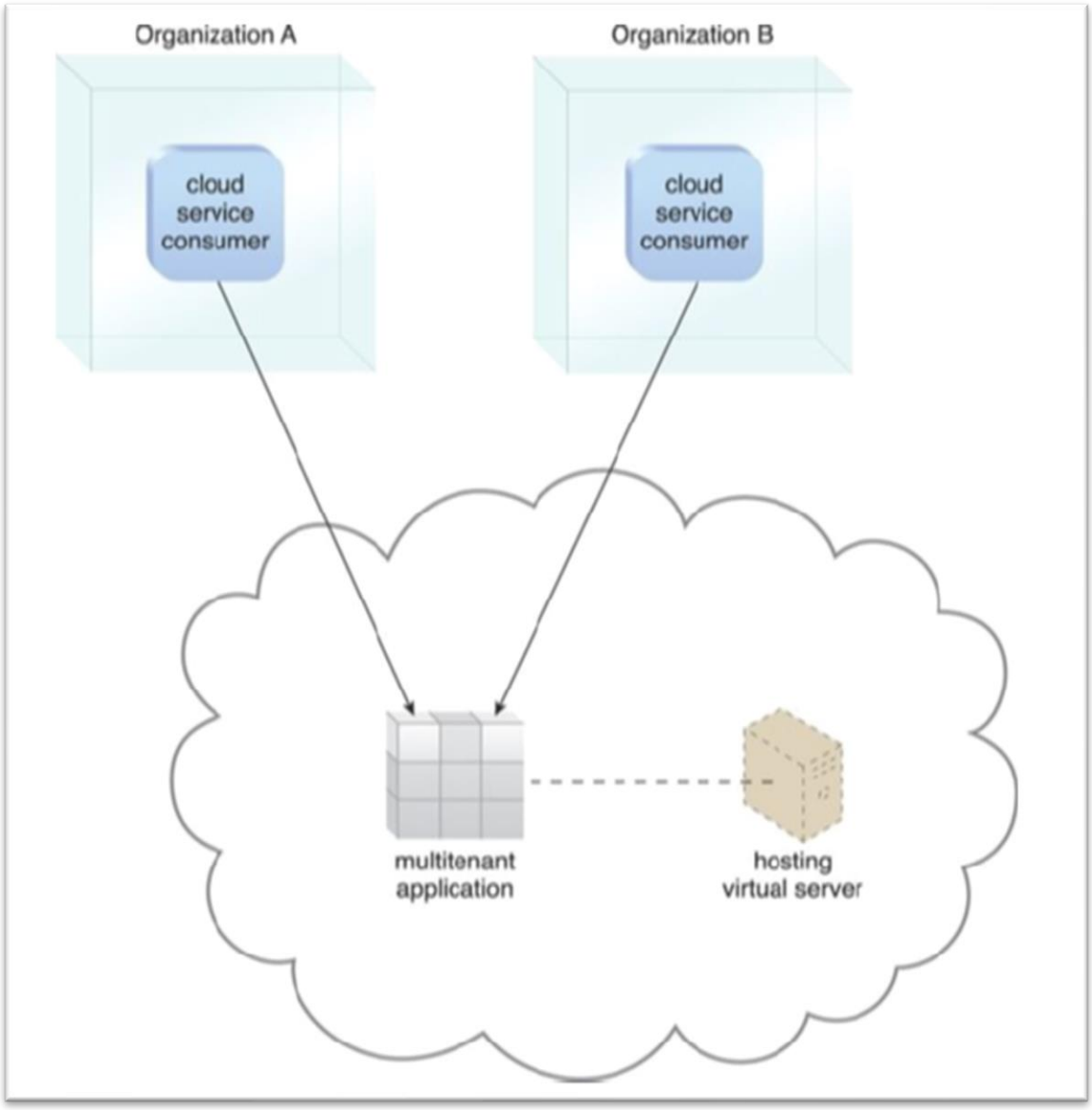
Hardware independence is a key feature of virtualization. Through hardware independence, virtual servers can easily be moved to another virtualization host, automatically resolving multiple hardware-software incompatibility issues. As a result, cloning and manipulating virtual IT resources is much easier than duplicating physical hardware.

The coordination function that is provided by the virtualization software allows multiple virtual servers to be simultaneously created in the same virtualization host, which is called server consolidation.



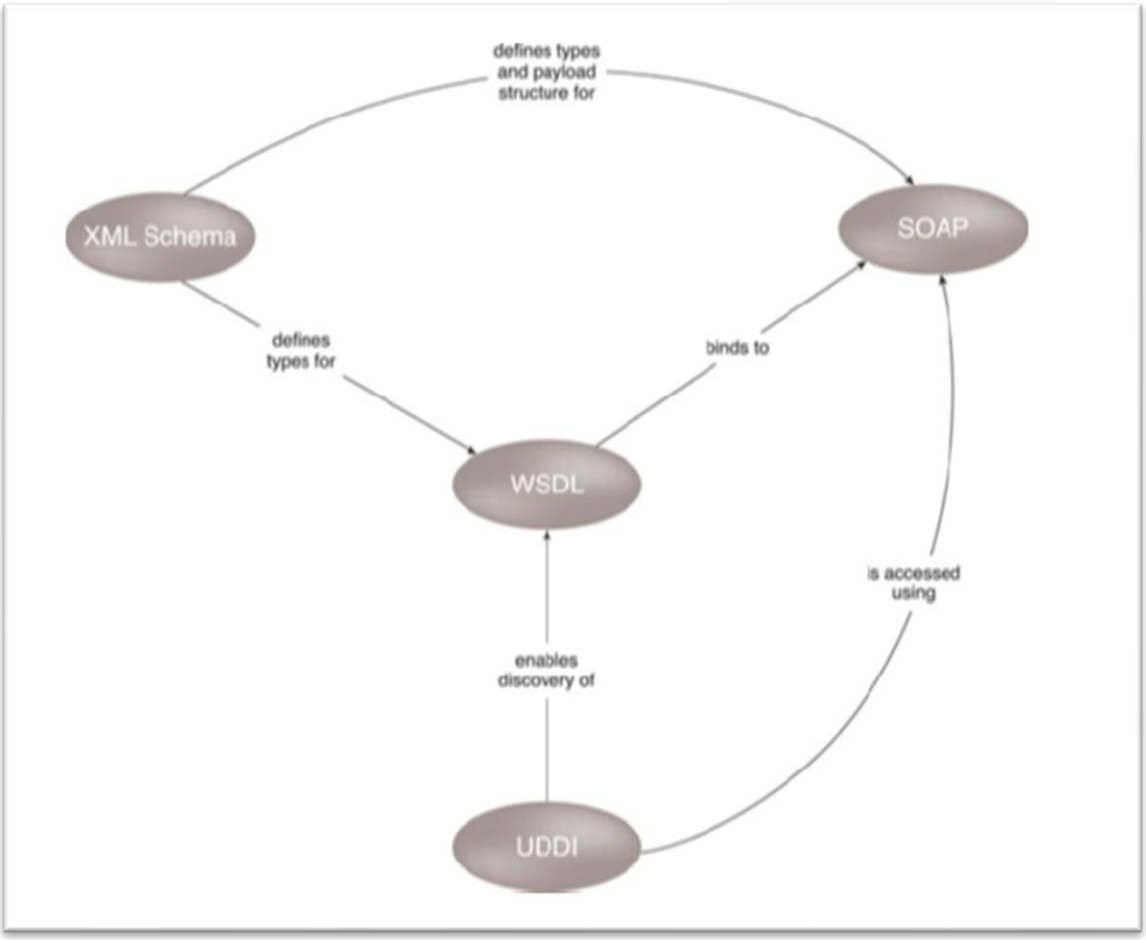
The three basic architectural tiers

Resource replication is another salient feature of virtualization technology, which enables standardized virtual machine images to be created for pre-packaging in virtual disk images in support of instantaneous deployment.



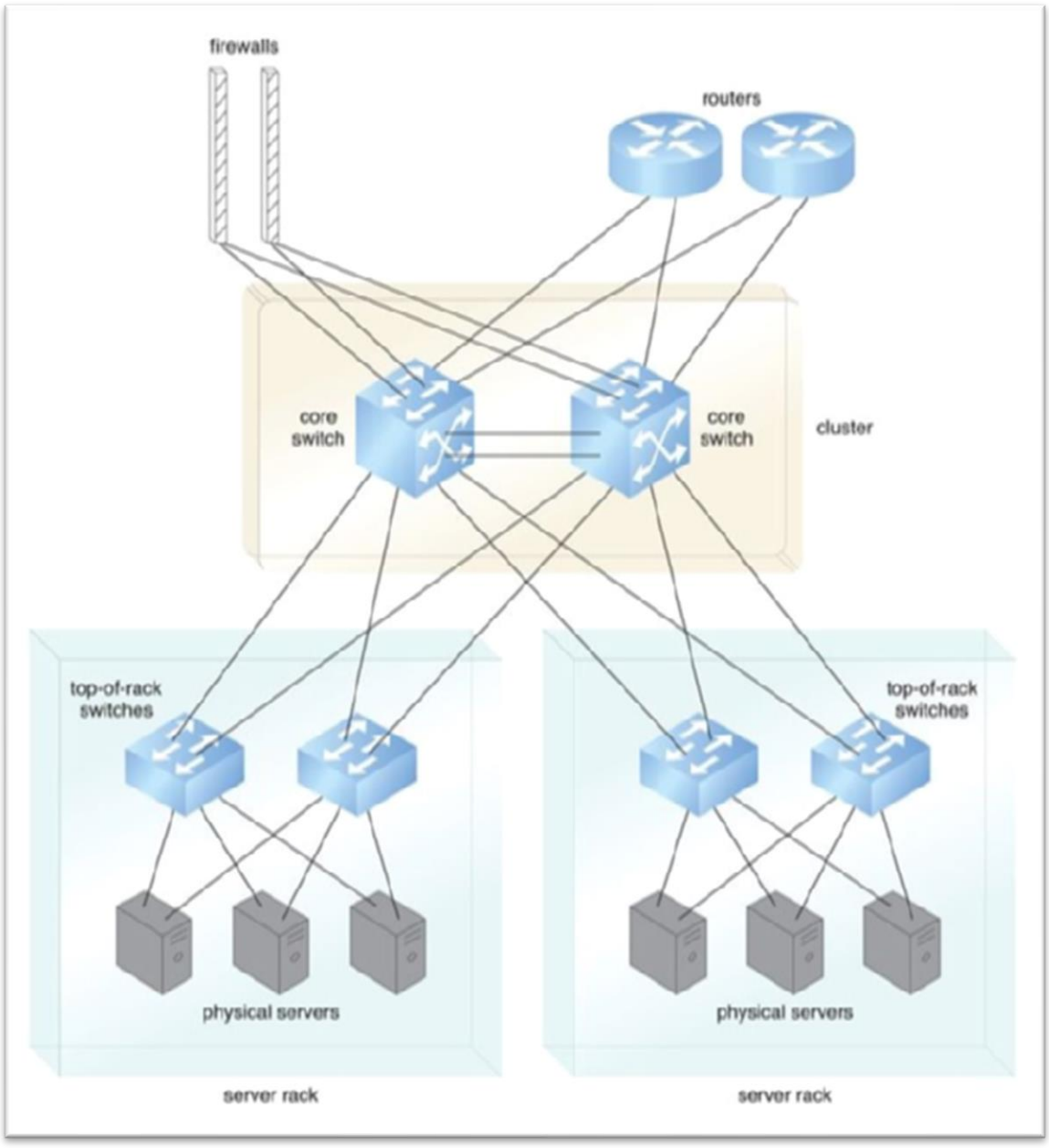
Application that is serving multiple loud

Web technology is used as both the implementation medium and the management interface for cloud services. This is due to the fundamental reliance on internetworking, the universality of web browsers, and the ease of web-based service development. Resources that are accessible through the World Wide Web are known as Web resources, a more general term than IT resources. IT resources, within the context of cloud computing, represent physical or virtual IT-related artifacts that can be software or hardware-based.



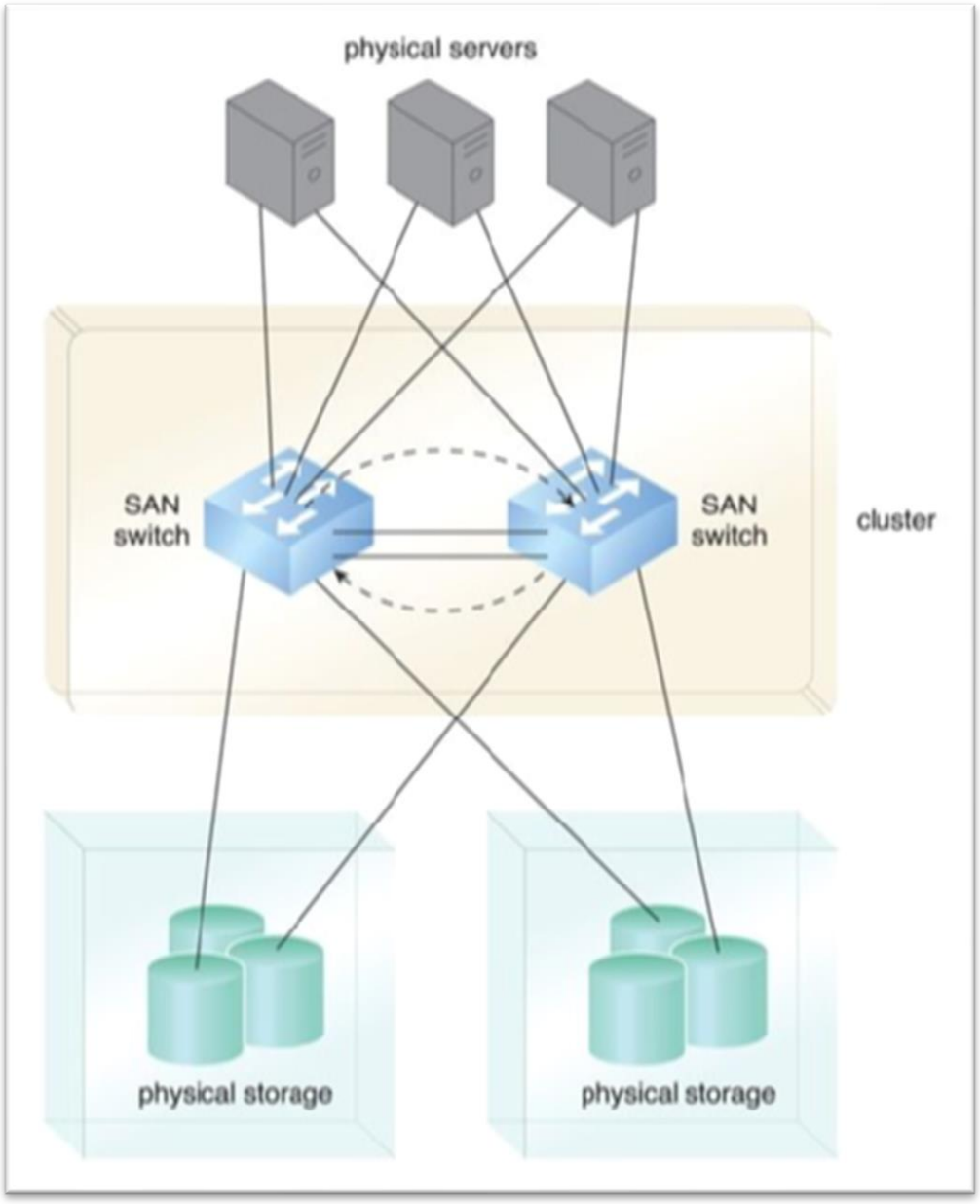
Web service technologies commonly

The Web technology architecture consists of three fundamental elements: Uniform Resource Locator (URL), Hypertext Transfer Protocol (HTTP), and Markup Languages (HTML, XML). Web applications are distributed applications that use web-based technologies and rely on web browsers for the presentation of user interfaces. A common architectural abstraction for web applications is based on the basic three-tier model, which consists of the presentation layer, the application layer, and the data layer. The presentation layer has components on both the client and server side, and Web servers receive client requests and retrieve requested resources directly as static Web pages or by invoking application logic.



Server Network connections

The utilization of web technology in cloud computing is crucial due to its dependence on internetworking, web browser universality, and straightforward web-based service development. In comparison to IT resources, web resources refer to resources or artifacts available via the World Wide Web in a broader sense. In contrast, IT resources are physical or virtual IT-related artifacts that can be software or hardware-based.

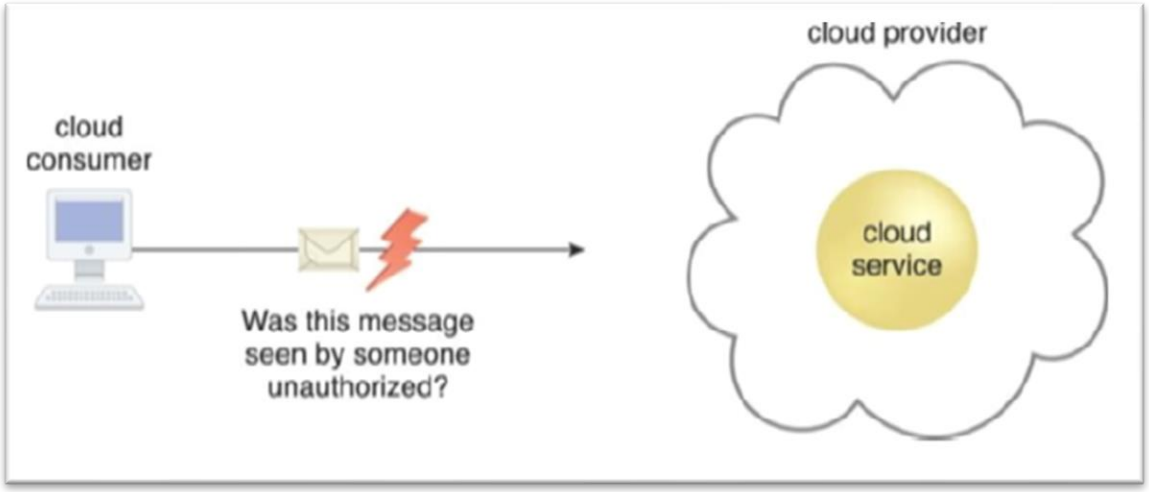


Storage System Network connections

Three fundamental elements make up the technology architecture of the web, which are the Uniform Resource Locator (URL), Hypertext Transfer Protocol (HTTP), and Markup Languages (HTML, XML). Web applications are distributed applications that utilize web-based technologies and rely on web browsers for user-interface presentation.

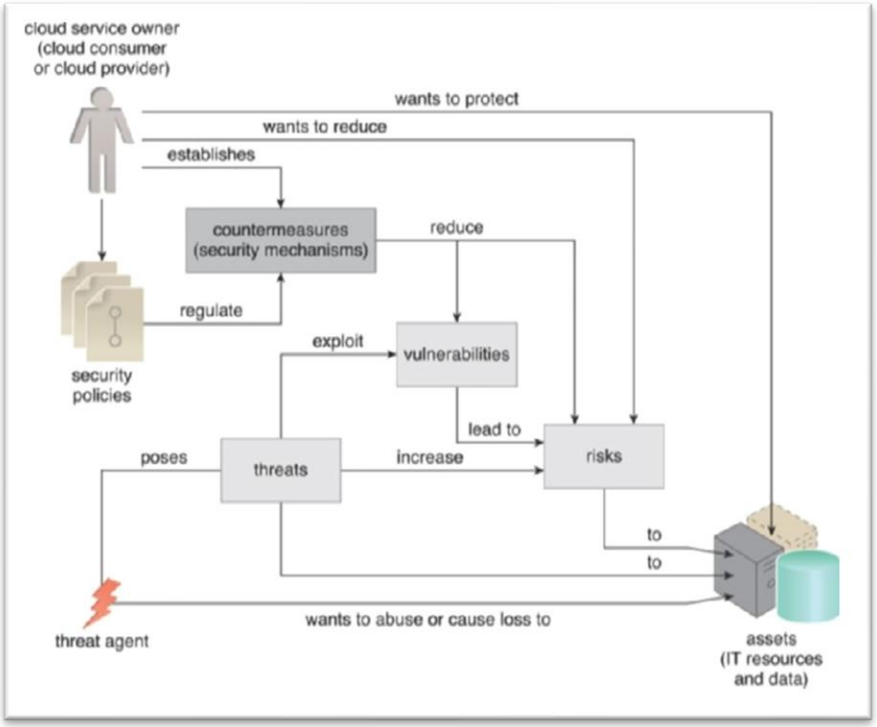
Fundamental Cloud Security

This chapter provides an introduction to basic information security in cloud computing, followed by a definition of common threats and attacks specific to public cloud environments.



The cloud consumer to the cloud service

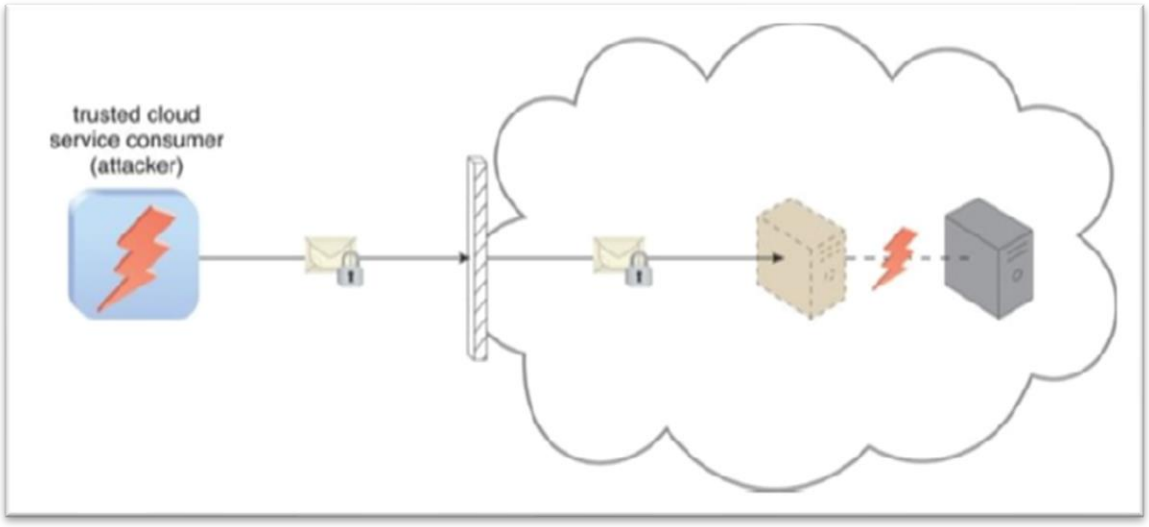
Information security encompasses various techniques, technologies, regulations, and behaviors that work together to protect computer systems and data integrity and access. The goal of IT security measures is to defend against malicious intent and unintentional user error.



How security policies and security

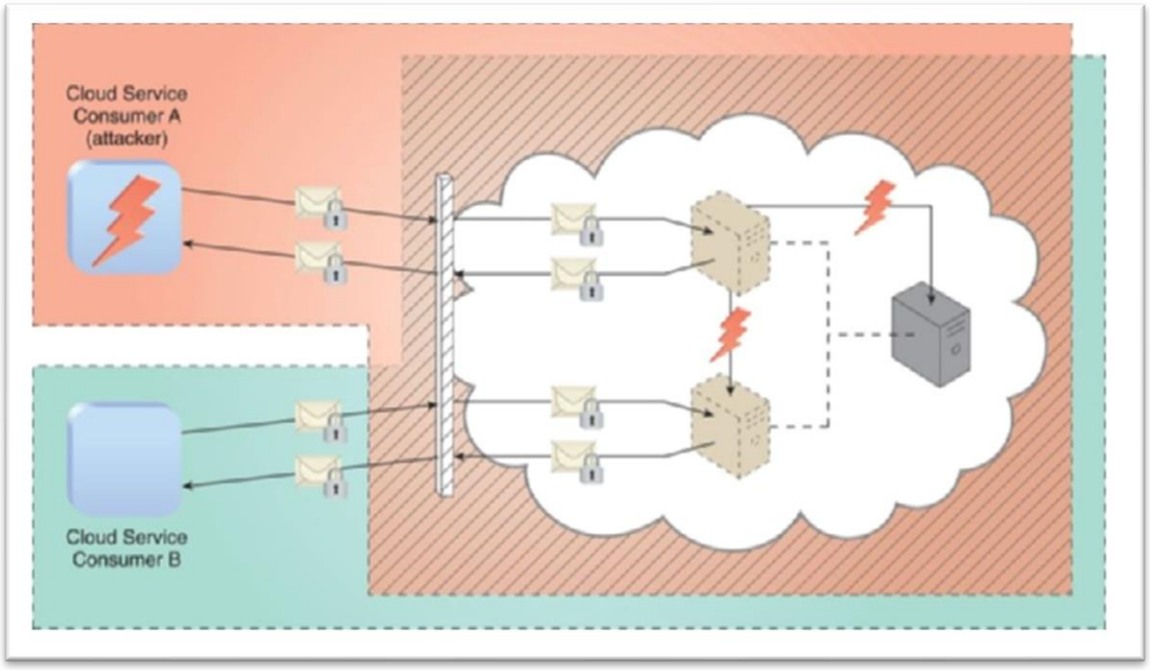
The upcoming sections define fundamental security terms relevant to cloud computing and describe associated concepts. Confidentiality, the first term, refers to something being made accessible only to authorized parties. In cloud environments, this primarily pertains to restricting access to data in transit and storage.

Integrity, the second term, refers to data not having been altered by an unauthorized party. This is an important issue that concerns data integrity in the cloud. Authenticity, the third term, refers to something having been provided by an authorized source. This concept encompasses non-repudiation, which means parties cannot deny or challenge the authentication of an interaction.



An authorized cloud service consumer

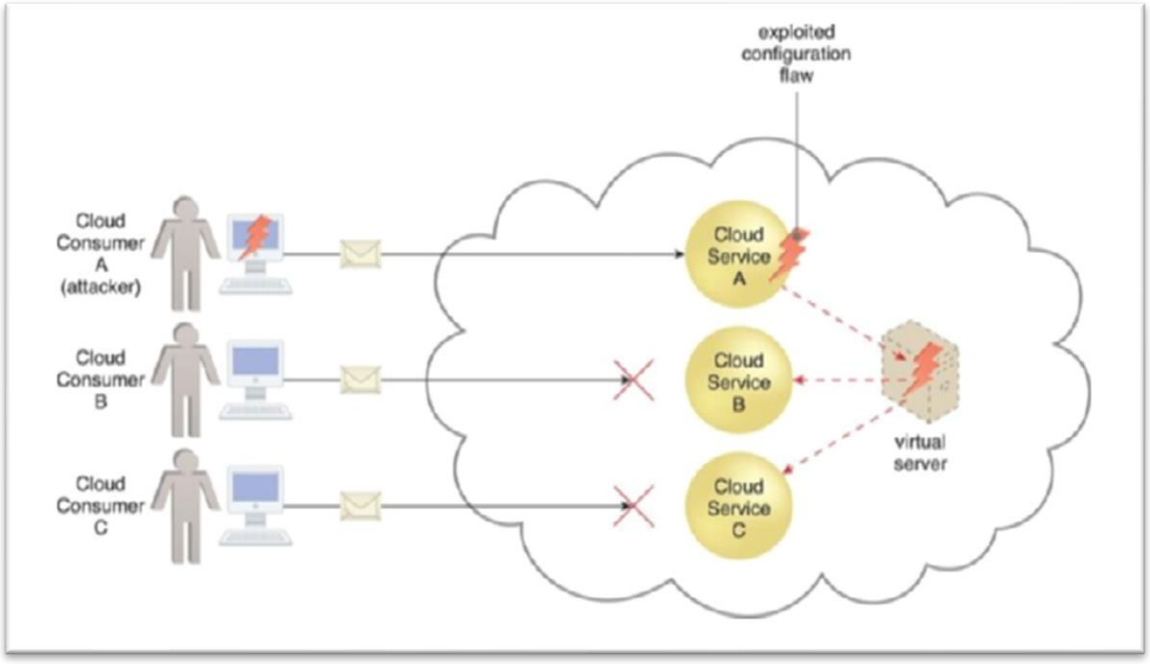
Availability refers to being accessible and usable during a specified period. Threats are potential security violations that can breach privacy and/or cause harm. Vulnerabilities are weaknesses that can be exploited due to insufficient security controls or by an attack. Risk is the possibility of loss or harm arising from an activity, measured according to the threat level and the number of possible or known vulnerabilities.



Cloud Service Consumer A is trusted by the cloud

Security controls are countermeasures used to prevent or respond to security threats and to reduce or avoid risk. Security mechanisms comprise a defensive framework that protects IT resources, information, etc. These mechanisms are typically described in terms of components that work together to counteract security threats and reduce risks. Details on how to use security countermeasures are typically outlined in the security policy, which contains a set of rules and practices specifying how to implement a system, service, or security plan for maximum protection of sensitive and critical IT resources.

Additional Considerations for Cloud Security is a diverse checklist of issues and guidelines for cloud security. One of the issues is Flawed Implementations, which pertains to the undesirable consequences of substandard design, implementation, or configuration of cloud service deployments. If the cloud provider's hardware and/or software have security flaws or operational weaknesses, it could impair the integrity, confidentiality, and/or availability of the cloud provider's IT resources as well as the cloud consumer IT resources hosted by the provider.



Cloud Service Consumer A's message triggers

This could be exploited by attackers. Another issue is Security Policy Disparity, where cloud consumers must understand that their traditional information security approach may not be the same as the cloud provider. The incompatibility needs to be evaluated to ensure that any data or other IT assets being relocated to a public cloud are adequately protected. Furthermore, when leasing infrastructure-based IT resources, the cloud consumer may not be granted sufficient administrative control or influence over the security policies that apply to the IT resources leased from the cloud provider. Contracts are also essential when it comes to asset security.

The on-going risk management process

Cloud consumers must carefully examine contracts and SLAs to ensure that security policies and other relevant guarantees are satisfactory. They must also determine the amount of liability assumed by the cloud provider and the level of indemnity the provider may ask for. Another issue is determining where the lines are drawn between cloud consumer and cloud provider assets. Risk Management is also an essential consideration, and cloud consumers are encouraged to perform a formal risk assessment as part of a risk management strategy.

This is comprised of a set of coordinated activities for overseeing and controlling risks. The main activities are generally defined as risk assessment, risk treatment, and risk control. In the risk assessment stage, the cloud environment is analyzed to identify potential vulnerabilities and shortcomings that threats can exploit. The identified risks are quantified.

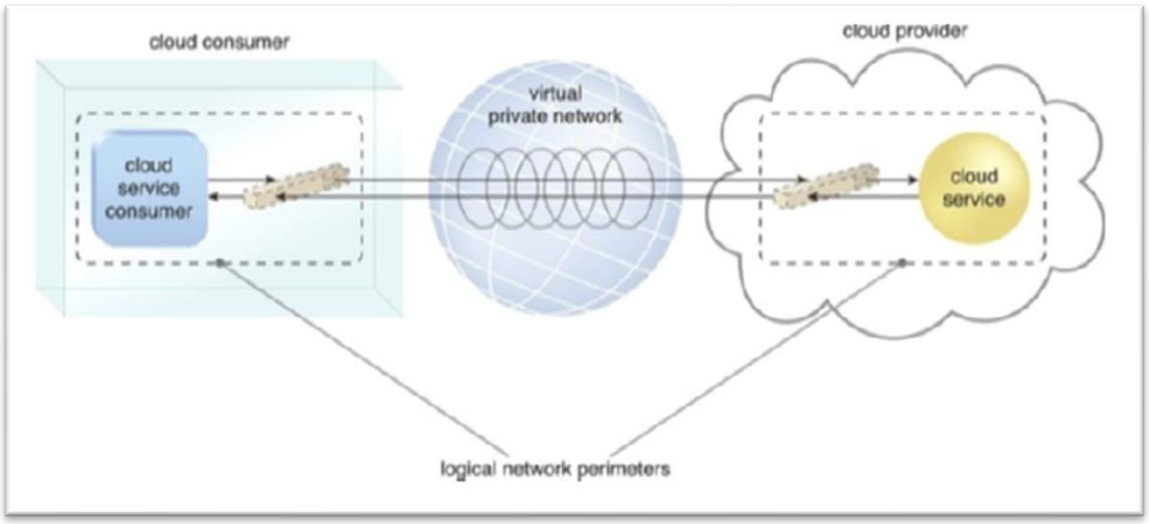
Cloud Computing Mechanism

In the realm of IT, technology mechanisms are specific artifacts that are widely recognized within the industry and unique to a particular computing platform. Due to the technology-focused nature of cloud computing, it is crucial to establish a formal set of mechanisms that serve as the building blocks for exploring cloud technology architectures. However, it is important to note that this collection of mechanisms is not exhaustive, and there are many other possible definitions that can be included. These mechanisms are referred to throughout the architectural models presented: Cloud Computing Architecture.

Cloud Infrastructure Mechanisms

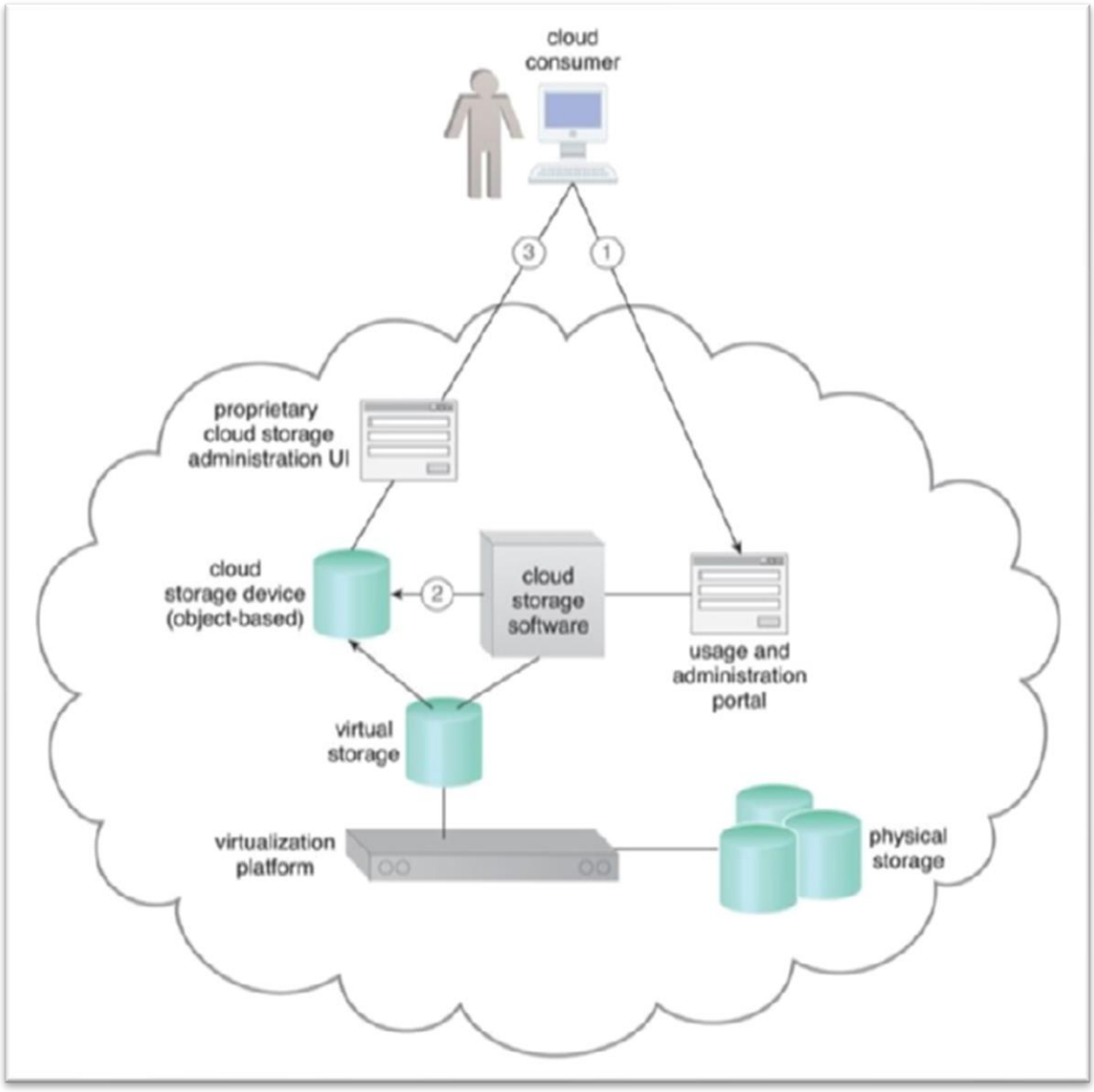
Cloud infrastructure mechanisms are basic components of cloud environments that serve as the foundation of fundamental cloud technology architecture. These mechanisms include Logical Network Perimeter, Virtual Server, Cloud Storage Device, Cloud Usage Monitor, Resource Replication, and Ready-Made Environment. These core components are common to cloud platforms, and each establishes a virtual boundary that can isolate a group of related cloud-based IT resources that may be physically distributed. This article provides an overview of the first three mechanisms.

The Logical Network Perimeter, defined as the isolation of a network environment from the rest of a communications network, can isolate IT resources in a cloud from non-authorized users, non-users, and cloud consumers, and control the bandwidth that is available to isolated IT resources. Typically established through network devices that supply and control connectivity, such as virtual firewalls and virtual networks, this mechanism allows the isolation of the network environment within the data center infrastructure.



Two logical network perimeters surround

Virtual servers, a form of virtualization software that emulates a physical server, are used by cloud providers to share the same physical server with multiple cloud consumers. Each virtual server can host numerous IT resources and cloud-based solutions. Cloud consumers that install or lease virtual servers can customize their environments independently from other cloud consumers that may be using virtual servers hosted by the same underlying physical server.

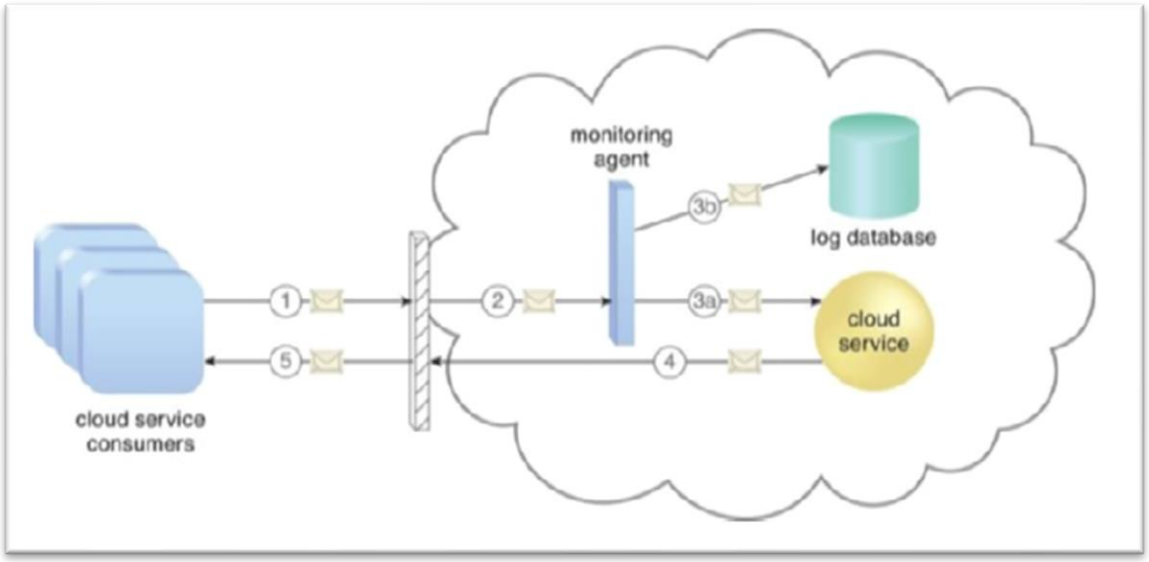


The cloud consumer interacts with the usage and administration portal

Cloud storage devices represent storage devices, such as hard drives, that store data in the cloud. Cloud storage devices can be public, private, or hybrid, depending on the data's sensitivity and the desired level of control. Cloud consumers can access their data from anywhere with an internet connection, and the cloud provider is responsible for managing and maintaining the storage devices.

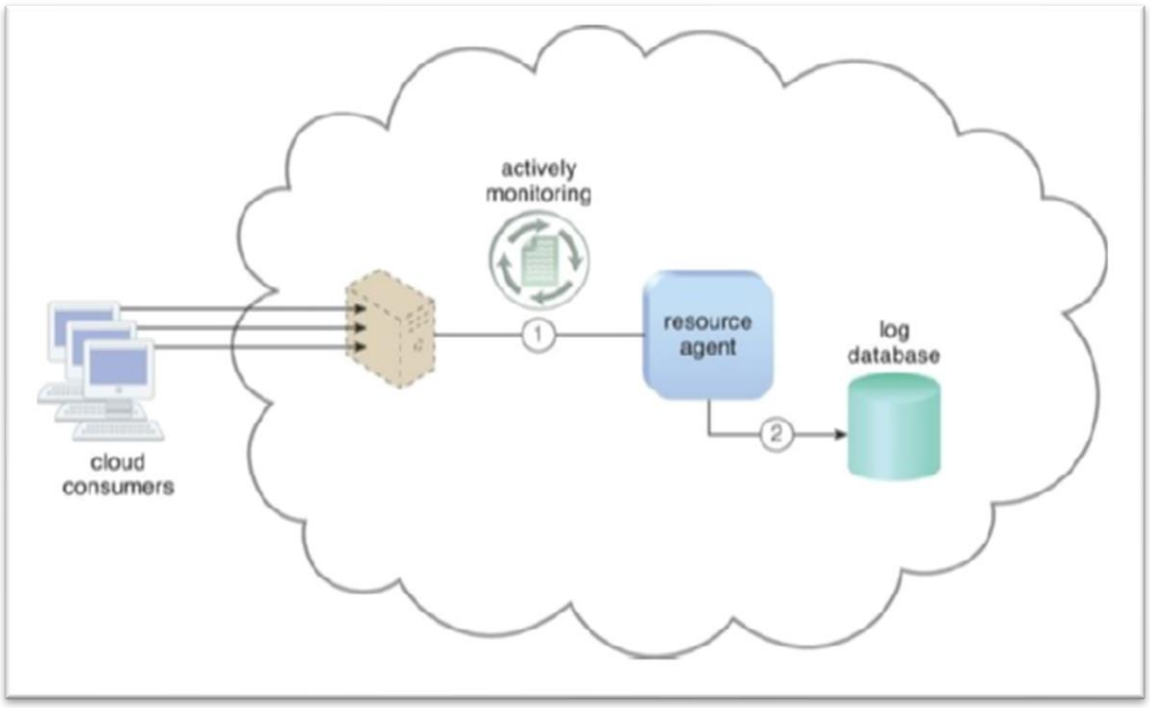
The Cloud Usage Monitor mechanism is a software program responsible for collecting and processing IT resource usage data. Depending on the type of usage metrics they collect and the way usage data needs to be

collected, Cloud Usage Monitors can exist in different formats. This section describes three common agent-based implementation formats. Each can be designed to forward collected usage data to a log database for post-processing and reporting purposes.



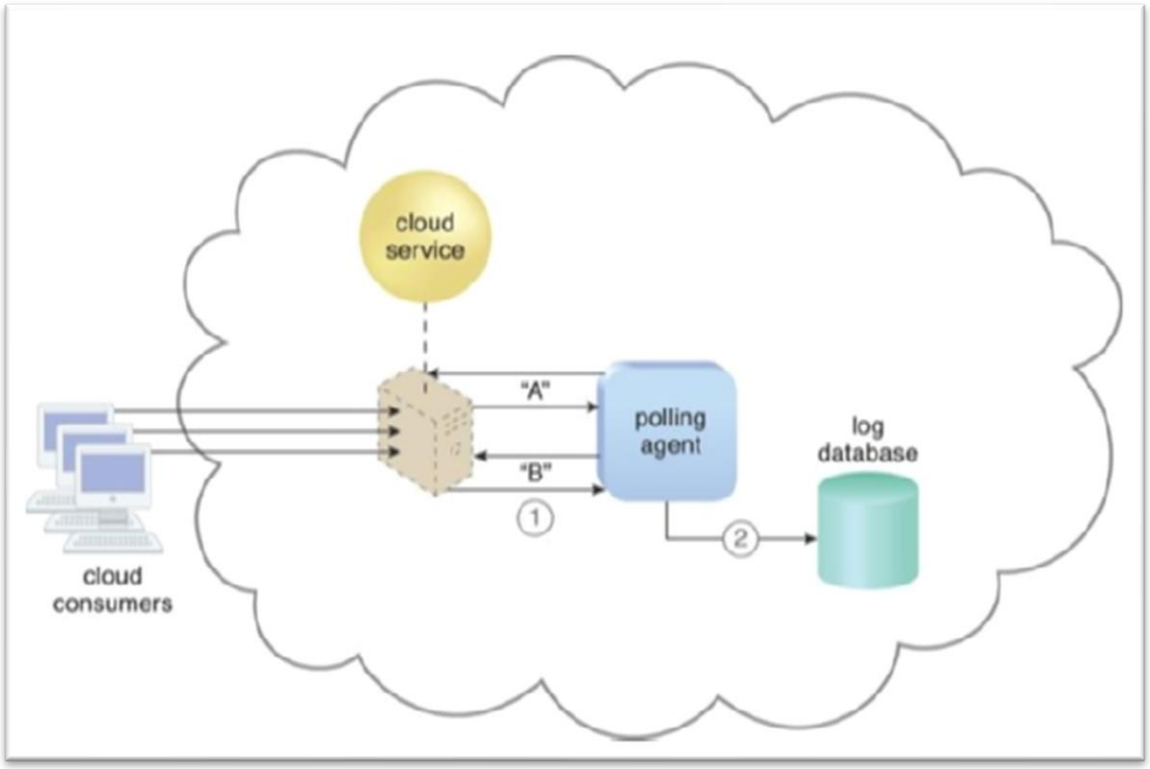
A cloud service consumer

The first format is a Monitoring Agent, which is an event-driven program that exists as a service agent and resides along existing communication paths to transparently monitor and analyze data flows. This type of Cloud Usage Monitor is commonly used to measure network traffic and message metrics.



The resource agent is actively monitoring a virtual server

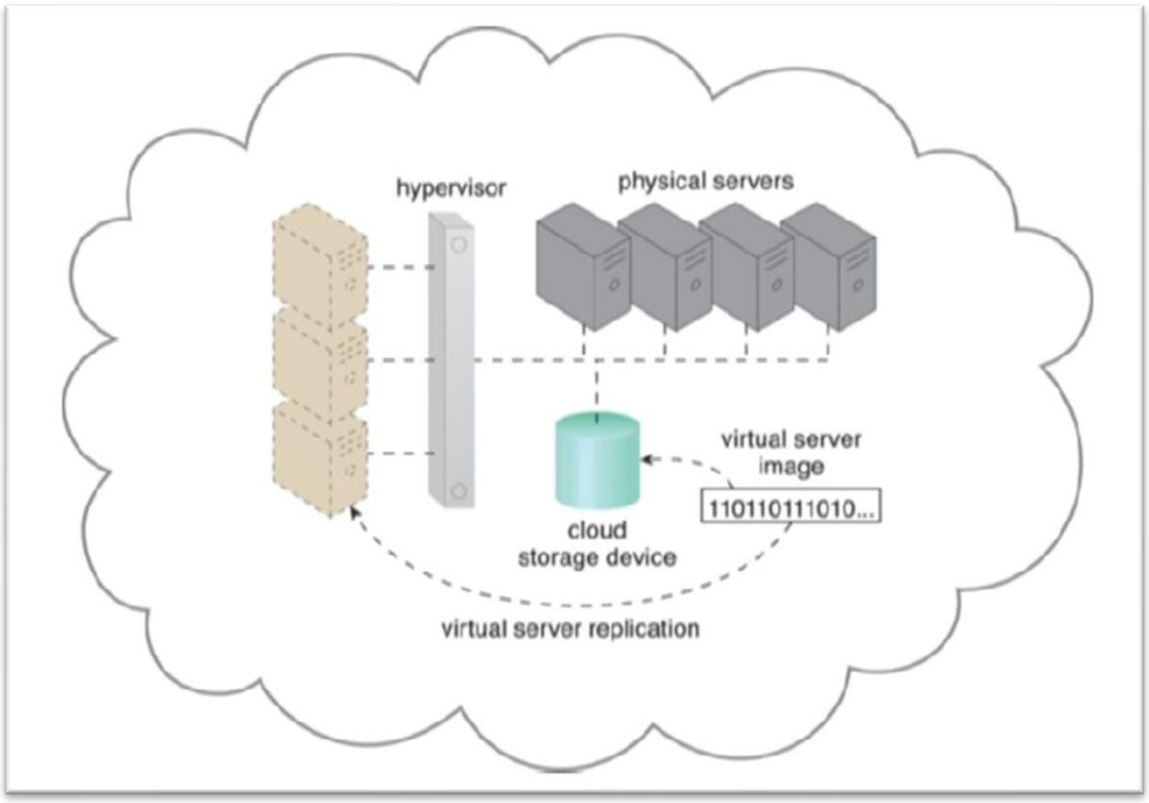
The second format is a Resource Agent, which is a processing module that collects usage data by having event-driven interactions with specialized resource software. This module is used to monitor usage metrics based on pre-defined, observable events at the resource software level, such as initiating, suspending, resuming, and vertical scaling.



A polling agent monitors the status of a cloud service

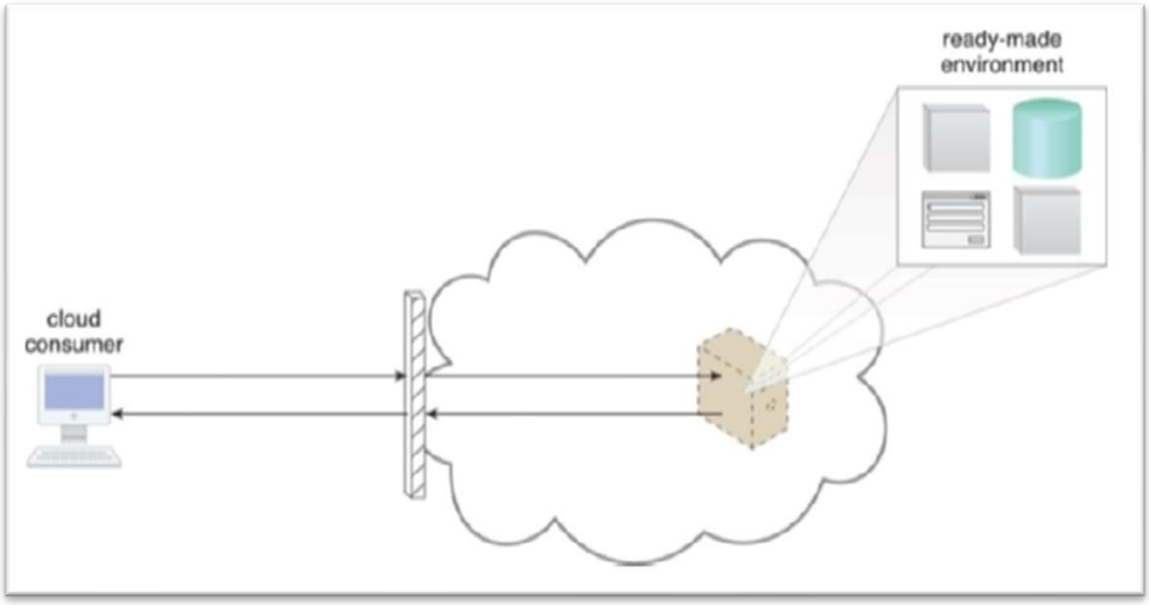
The third format is a Polling Agent, which is a processing module that collects cloud service usage data by polling IT resources. This type of cloud service monitor is commonly used to periodically monitor IT resource status, such as uptime and downtime.

Resource replication is defined as the creation of multiple instances of the same IT resource, typically performed when an IT resource's availability and performance need to be enhanced. Virtualization technology is used to implement the resource replication mechanism to replicate cloud-based IT resources.



The hypervisor replicates several instances

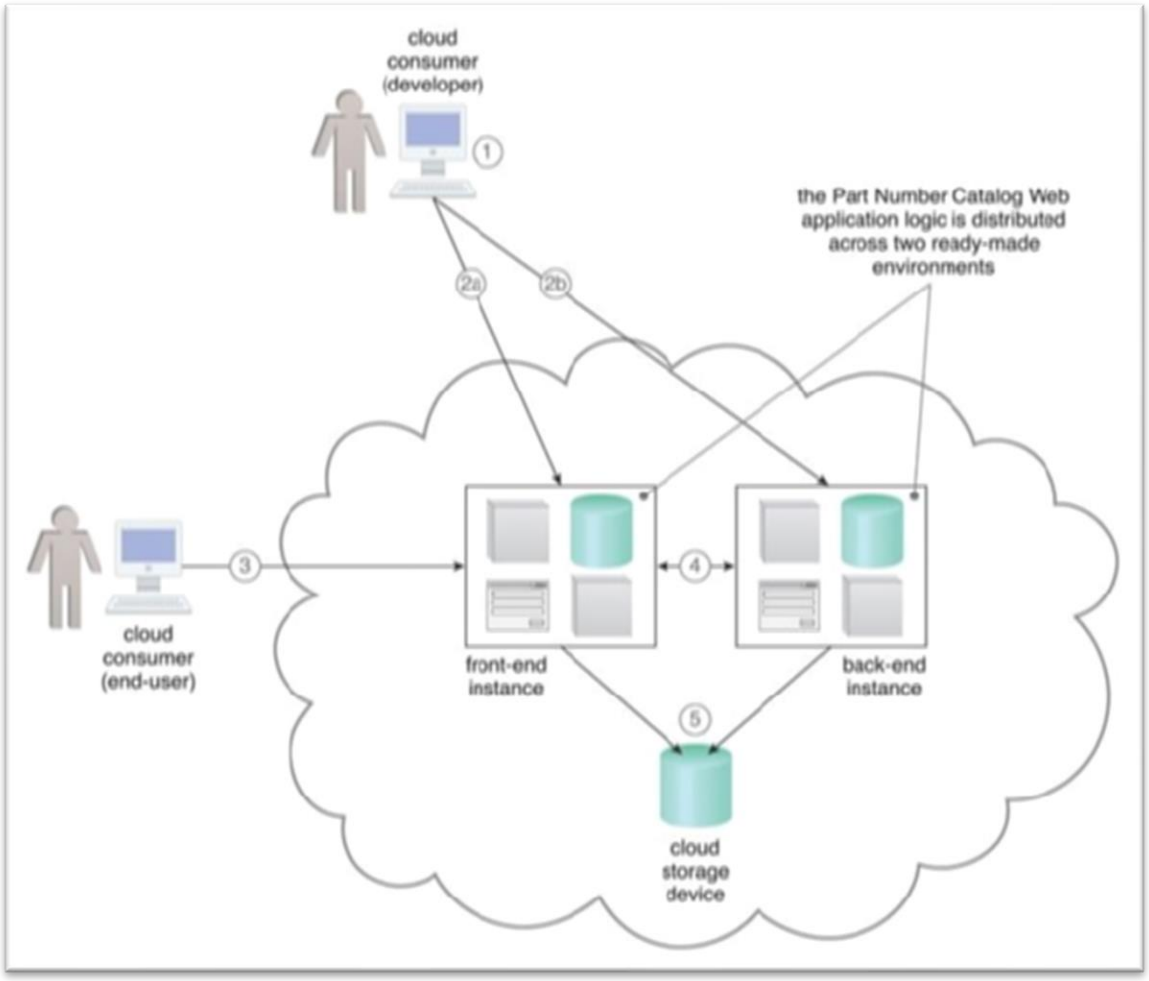
The Ready-Made Environment mechanism is a defining component of the PaaS cloud delivery model that represents a pre-defined, cloud-based platform comprised of a set of already installed IT resources, ready to be used and customized by a cloud consumer. These environments are utilized by cloud consumers to remotely develop and deploy their own services and applications within a cloud. Typical Ready-Made Environments include pre-installed IT resources, such as databases, middleware, development tools, and governance tools.



A cloud consumer accesses a ready-made environment hosted on a virtual server

A Ready-Made Environment is generally equipped with a complete software development kit (SDK) that provides cloud consumers with programmatic access to the development technologies that comprise their preferred programming stacks. Middleware is available for multitenant platforms to support the development and deployment of web applications.

Cloud providers offer runtime execution environments for cloud services that are based on different runtime performance and billing parameters. For example, a front-end instance of a cloud service can be configured to respond to time-sensitive requests more effectively than a back-end instance. The former variation will be billed at a different rate than the latter.



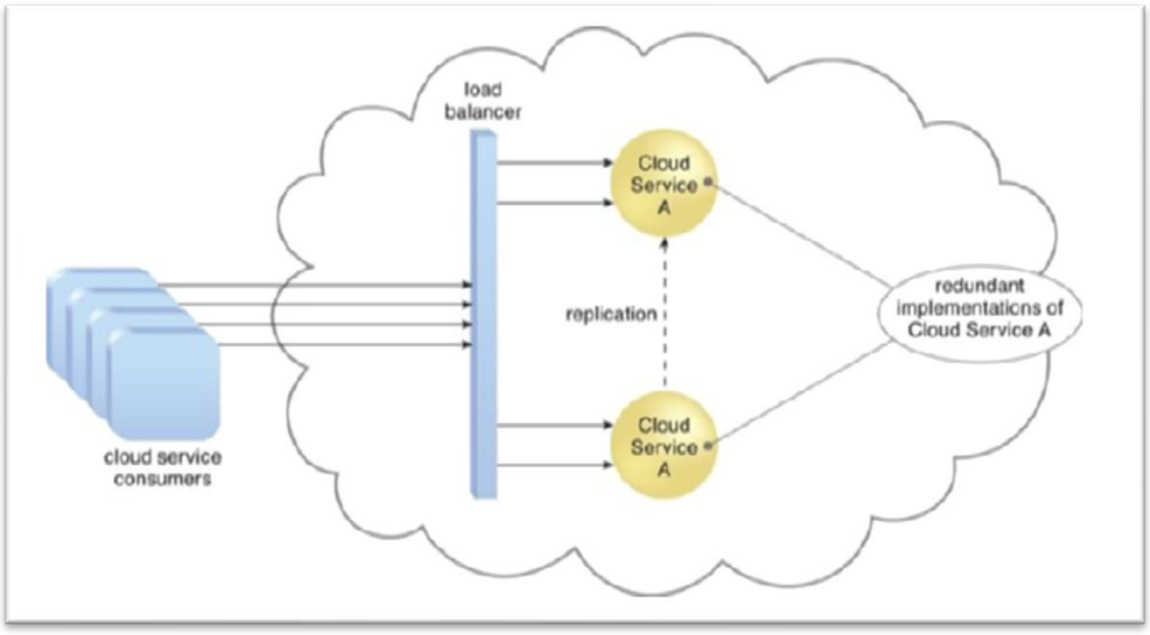
Case Study

As demonstrated in the case study example, a solution can be partitioned into groups of logic that can be designated for both front-end and back-end instance invocation to optimize runtime execution and billing.

Specialized Cloud Mechanisms

This section describes several specialized cloud mechanisms that can be considered extensions to cloud infrastructure. These mechanisms are designed to fulfill specific runtime functions in support of one or more cloud characteristics. The mechanisms discussed in this chapter include Automated Scaling Listener, Load Balancer, SLA Monitor, Pay-Per-Use Monitor Audit Monitor, Failover System, Hypervisor, Resource Cluster, Multi-Device Broker, and State Management Database. These mechanisms can be combined in various ways as part of distinct and custom technology architectures.

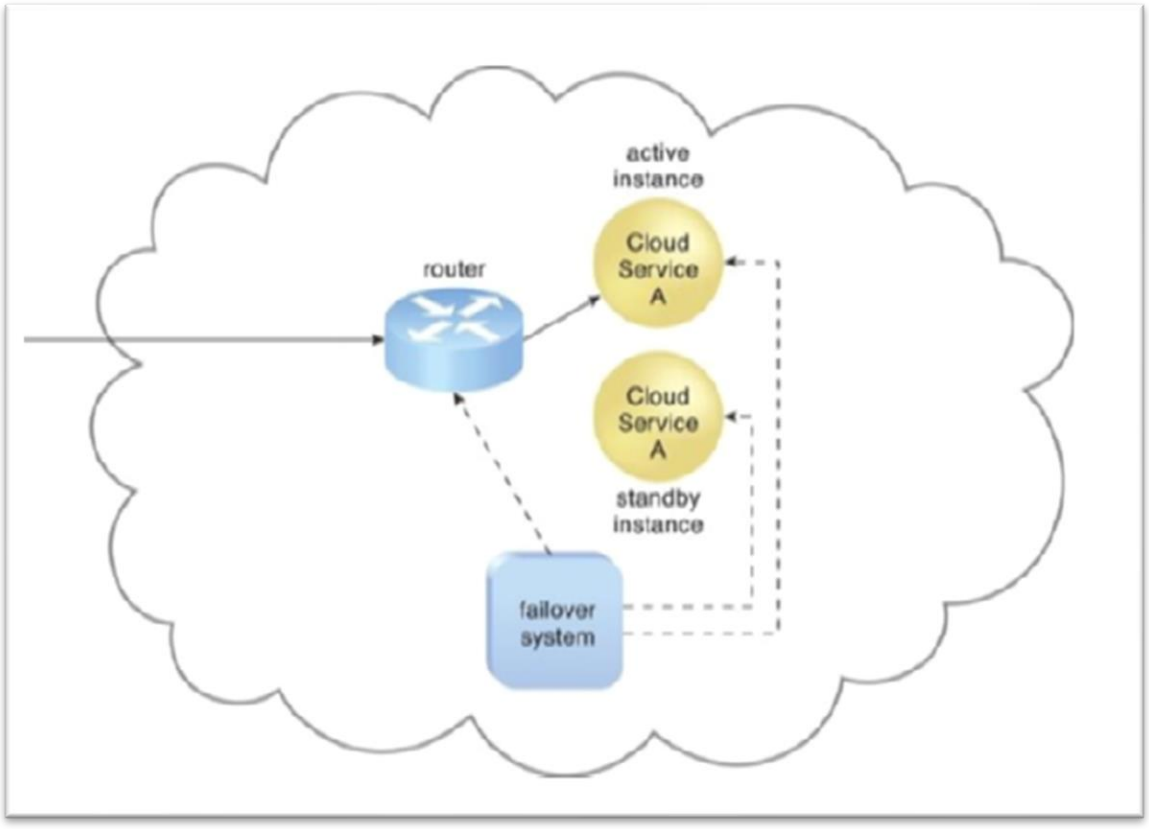
The Automated Scaling Listener is a service agent that monitors and tracks communications between cloud service consumers and cloud services for dynamic scaling purposes.



A load balancer implemented as a service agent

It automatically tracks workload status information and can provide different types of responses to workload fluctuation conditions. It can scale IT resources out or in based on parameters defined by the cloud consumer or notify the cloud consumer when workloads exceed current thresholds or fall below allocated resources.

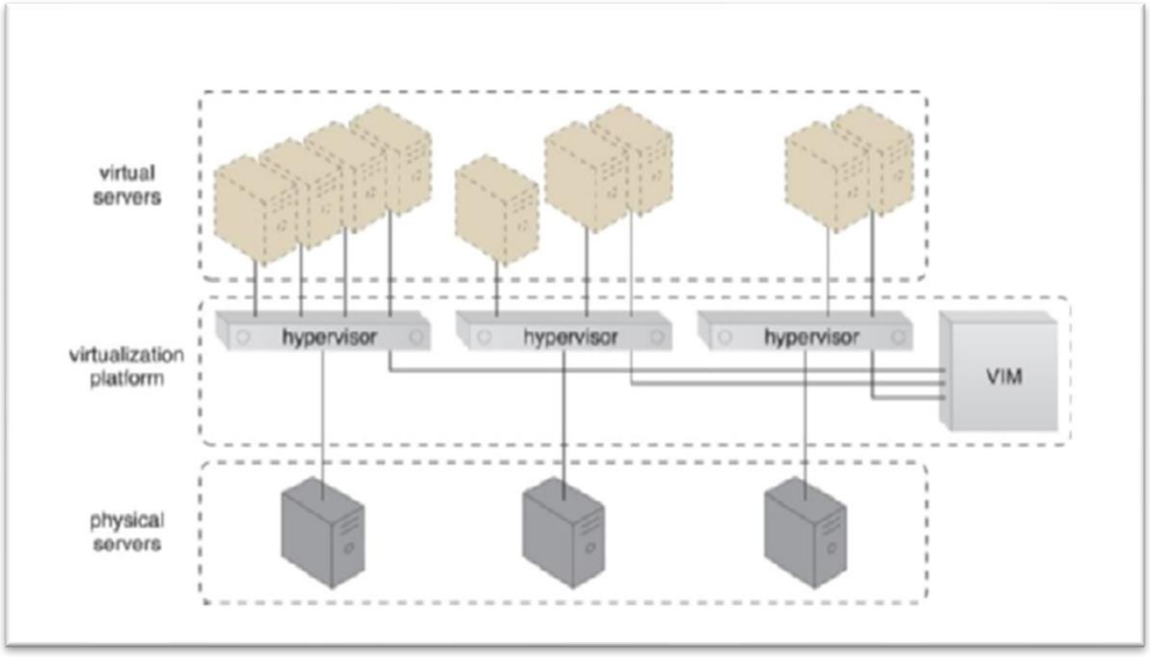
The Load Balancer is a runtime agent designed to balance a workload across two or more IT resources to increase performance and capacity beyond what a single IT resource can provide. It is programmed with a set of performance and QoS rules and parameters to optimize IT resource usage, avoid overloads, and maximize throughput. The Load Balancer mechanism can exist as a multi-layer network switch, dedicated hardware appliance, dedicated software-based system, or service agent.



The failover system monitors

The SLA Monitor mechanism is used to specifically observe the runtime performance of cloud services to ensure that they are fulfilling the contractual QoS requirements that are specified in service-level agreements. The SLA Monitor can be used to identify and resolve issues related to service quality, availability, and responsiveness.

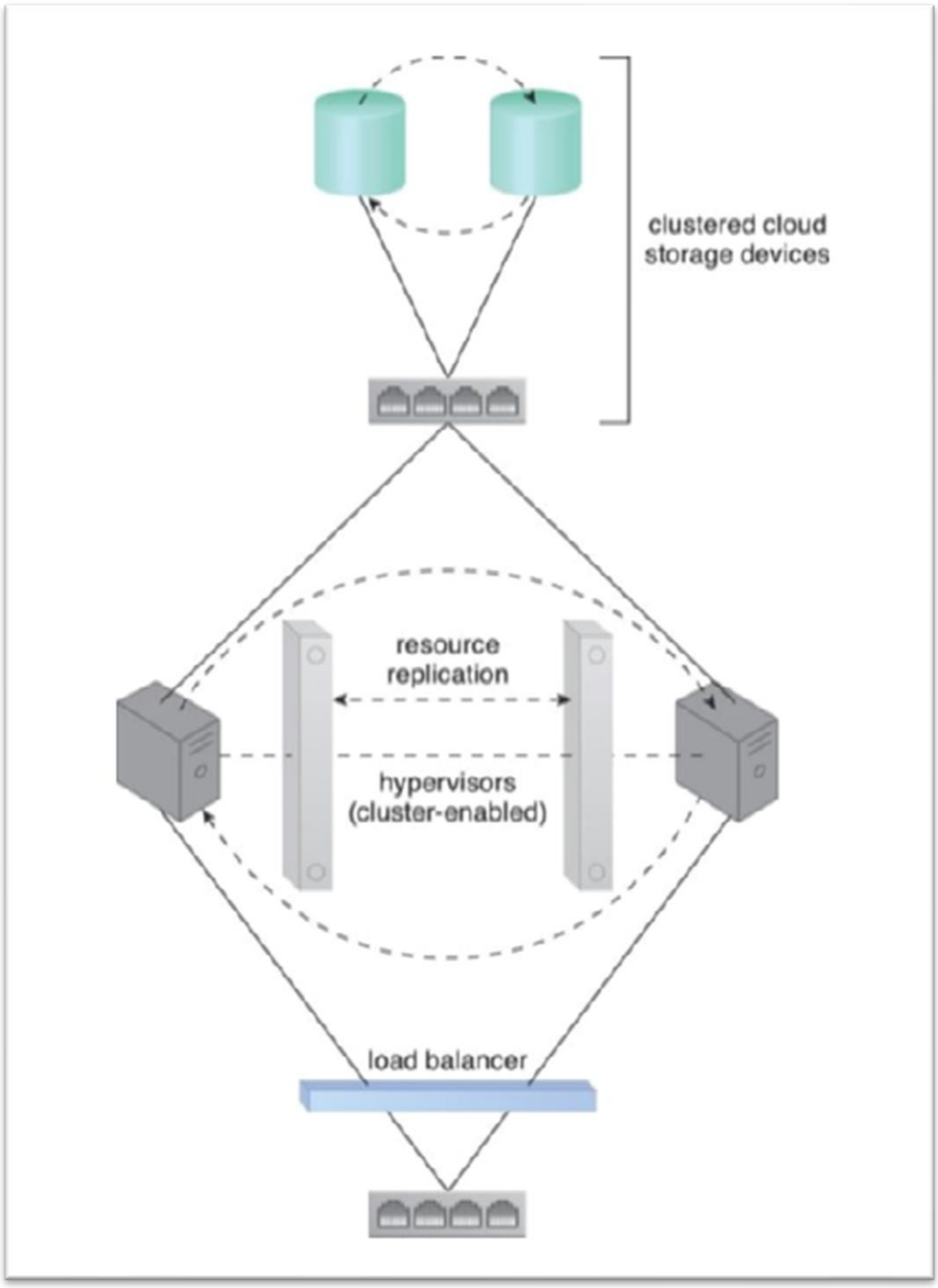
This section describes several specialized cloud mechanisms that can be considered extensions to cloud infrastructure. These mechanisms are designed to fulfill specific runtime functions in support of one or more cloud characteristics. The mechanisms discussed in this chapter include Automated Scaling Listener, Load Balancer, SLA Monitor, Pay-Per-Use Monitor Audit Monitor, Failover System, Hypervisor, Resource Cluster, Multi-Device Broker, and State Management Database. These mechanisms can be combined in various ways as part of distinct and custom technology architectures.



Virtual servers are created via individual hypervisor

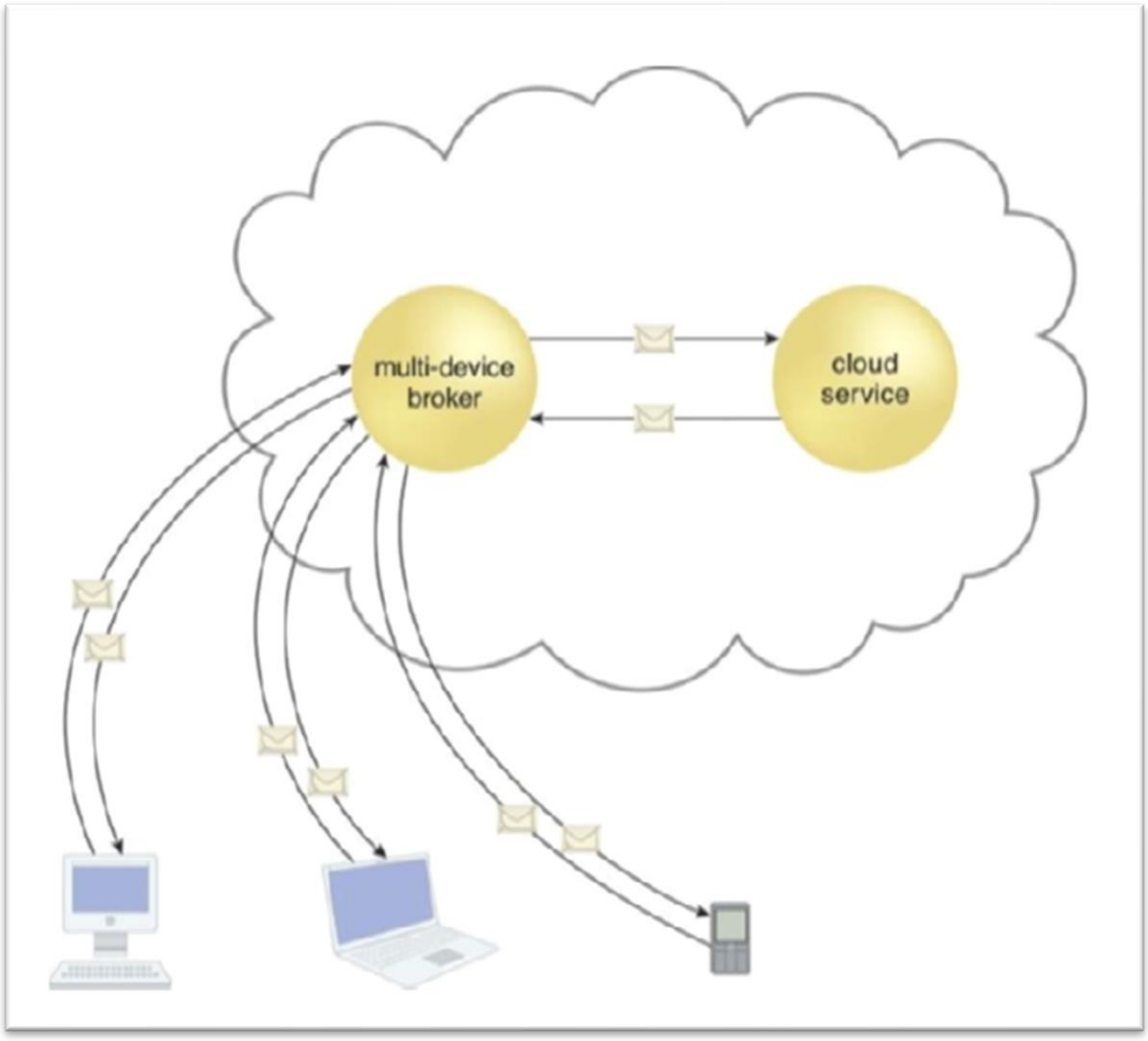
The Automated Scaling Listener is a service agent that monitors and tracks communications between cloud service consumers and cloud services for dynamic scaling purposes. It automatically tracks workload status information and can provide different types of responses to workload fluctuation conditions.

It can scale IT resources out or in based on parameters defined by the cloud consumer or notify the cloud consumer when workloads exceed current thresholds or fall below allocated resources.



Load balancing and resource replication are implemented through a cluster-enabled hypervisor

The Load Balancer is a runtime agent designed to balance a workload across two or more IT resources to increase performance and capacity beyond what a single IT resource can provide. It is programmed with a set of performance and QoS rules and parameters to optimize IT resource usage, avoid overloads, and maximize throughput. The Load Balancer mechanism can exist as a multi-layer network switch, dedicated hardware appliance, dedicated software-based system, or service agent.

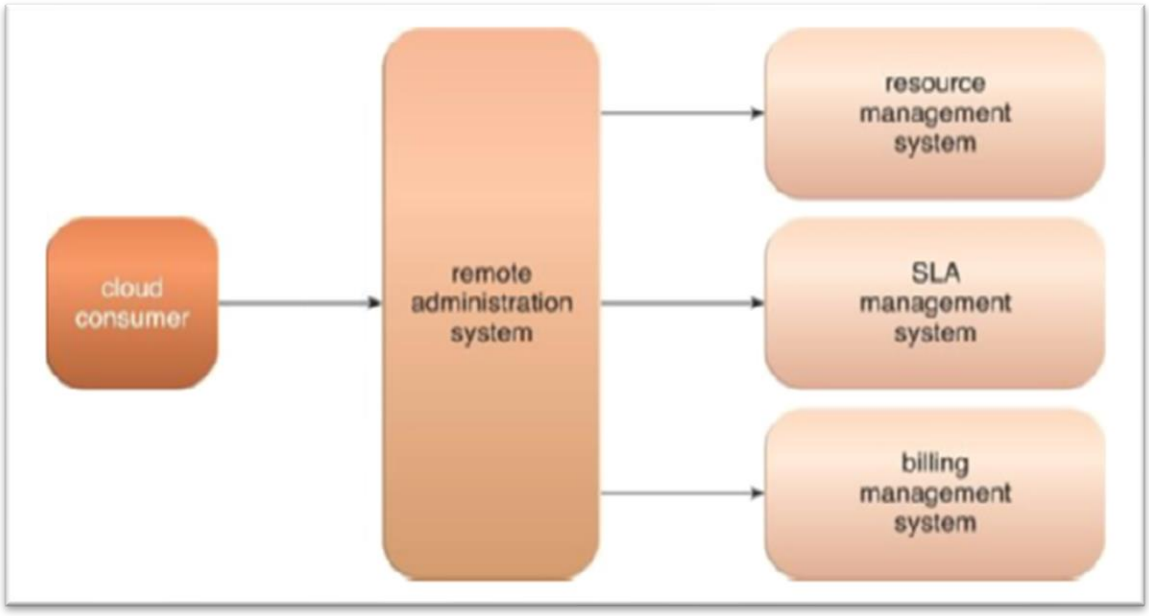


A multi-device broker contains the mapping logic

The SLA Monitor mechanism is used to specifically observe the runtime performance of cloud services to ensure that they are fulfilling the contractual QoS requirements that are specified in service-level agreements. The SLA Monitor can be used to identify and resolve issues related to service quality, availability, and responsiveness.

Cloud Management Mechanisms

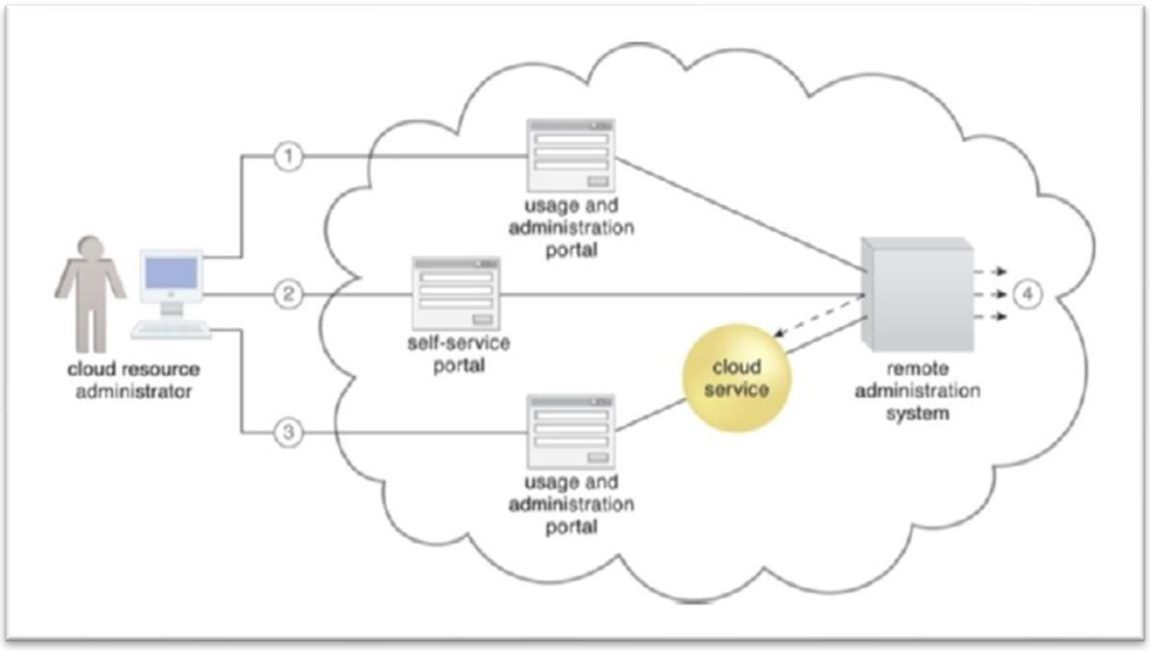
This chapter discusses several cloud-based management mechanisms that aid in the setup, configuration, maintenance, and monitoring of IT resources. These mechanisms form critical components of cloud technology architectures and facilitate the control and evolution of IT resources that make up cloud platforms and solutions. The four management-related mechanisms described in this chapter are the Remote Administration System, Resource Management System, SLA Management System, and Billing Management System. These systems typically offer integrated APIs and can be presented as individual products, custom applications, or combined into various product suites or multifunction applications.



The remote administration system

The Remote Administration System provides tools and user interfaces for external cloud resource administrators to configure and administer cloud-based IT resources. It establishes a portal for access to administration and management features of various underlying systems, including the Resource Management, SLA Management, and Billing Management Systems. The tools and APIs provided by the Remote Administration System are generally used by cloud providers to develop and customize online portals that provide cloud consumers with various administrative controls.

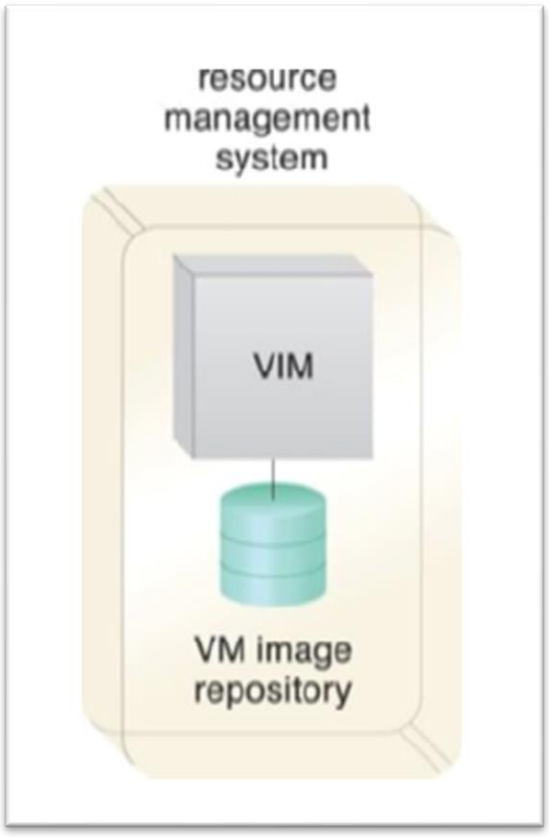
The two primary types of portals created with the Remote Administration System are the Usage and Administration Portal and the Self-Service Portal. The Usage and Administration Portal centralizes management controls to different cloud-based IT resources and can further provide IT resource usage reports. On the other hand, the Self-Service Portal is essentially a shopping portal that allows cloud consumers to search for a list of cloud services and IT resources available for lease from a cloud provider.



A cloud resource administrator

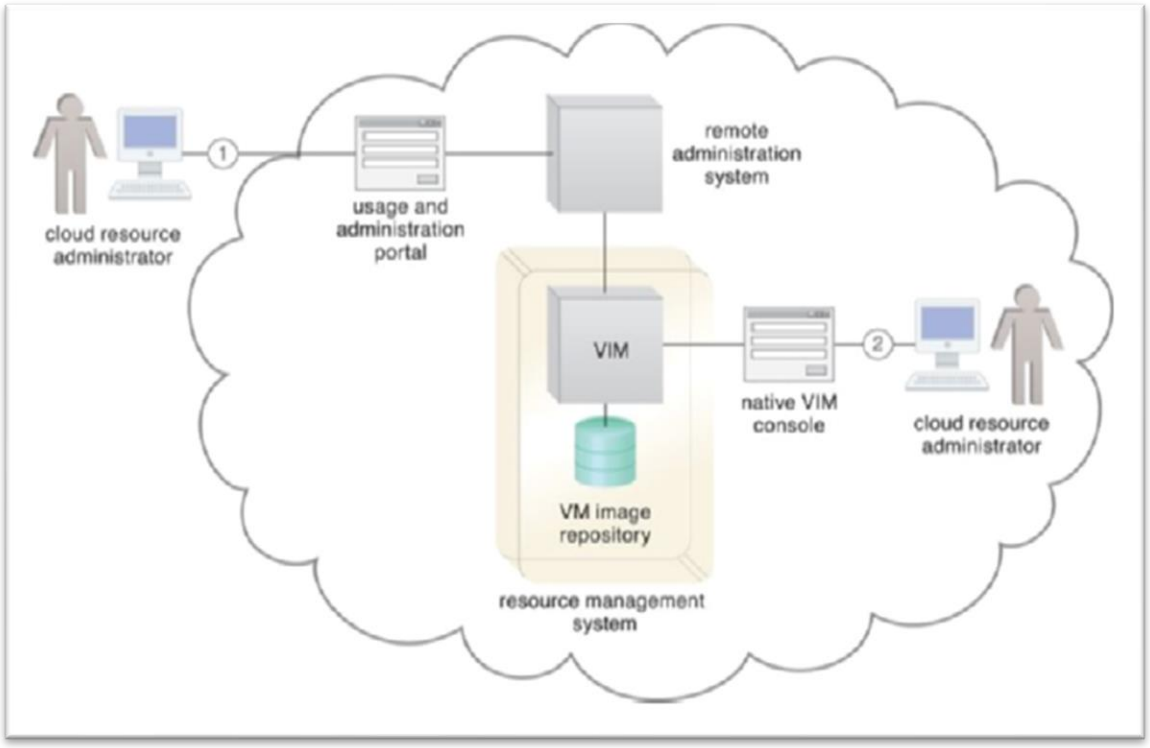
Tasks that can be performed by cloud consumers via a remote administration console include configuring and setting up cloud services, provisioning and releasing IT resources for on-demand cloud services, monitoring cloud service status, usage, and performance, monitoring QoS and SLA fulfillment, managing leasing costs and usage fees, managing user accounts, security credentials, authorization, and access control, tracking internal and external access to leased services, planning and assessing IT resource provisioning, and capacity planning.

While the user-interface provided by the Remote Administration System tends to be proprietary to the cloud provider, cloud consumers prefer working with remote administration systems that offer standardized APIs. This allows a cloud consumer to invest in the creation of its 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 interested in leasing and centrally administering IT resources from multiple cloud providers and/or IT resources residing in cloud and on-premise environments.



A resource management system encompassing

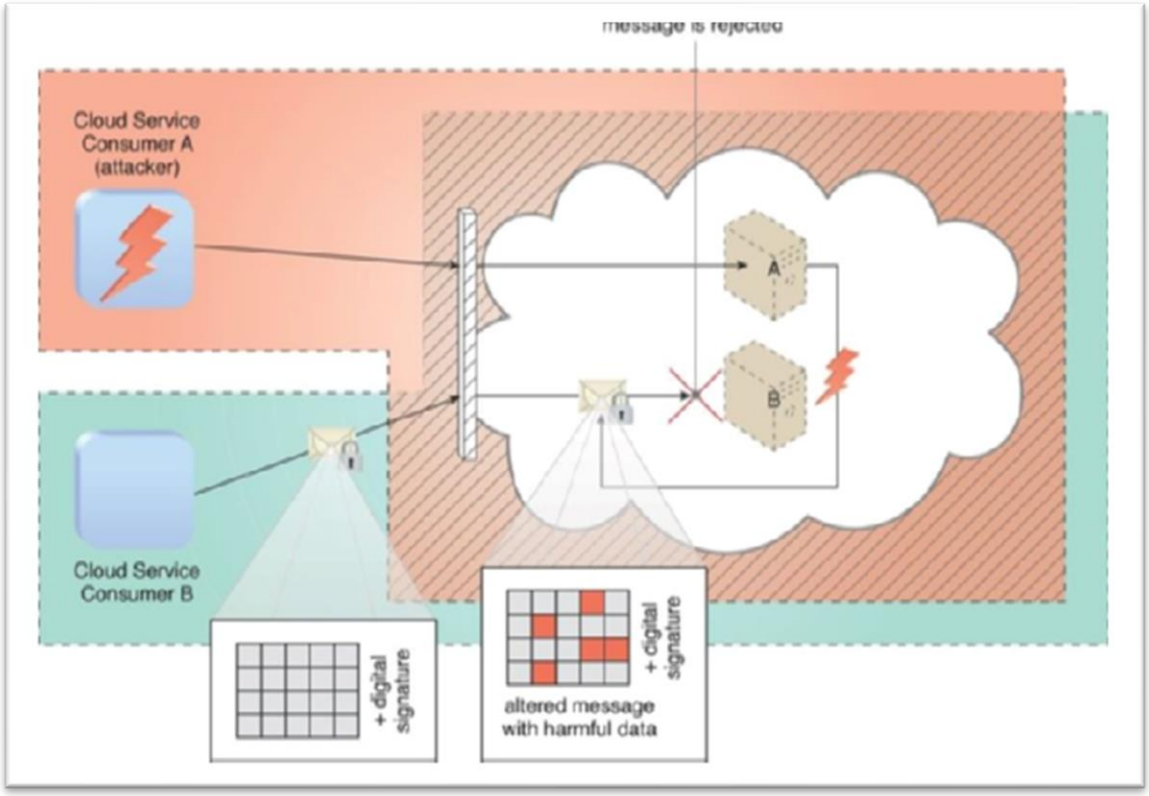
The Resource Management System helps coordinate IT resources in response to cloud consumer requests, such as provisioning new services, upgrading existing services, or reconfiguring existing services. The system enables the automation of resource allocation, resource management, and resource release tasks. This system also provides capabilities such as resource pooling, resource partitioning, resource abstraction, and resource scheduling, which enable efficient and effective use of IT resources.



Cloud resource administration portal

Cloud Security Mechanisms

In this chapter, we will discuss the essential security mechanisms used in cloud computing to mitigate security threats. Encryption is one of these mechanisms, which involves converting plaintext data into an unreadable form through a cipher algorithm. To revert the data to its original form, decryption keys are necessary.



Cloud Service Consumer

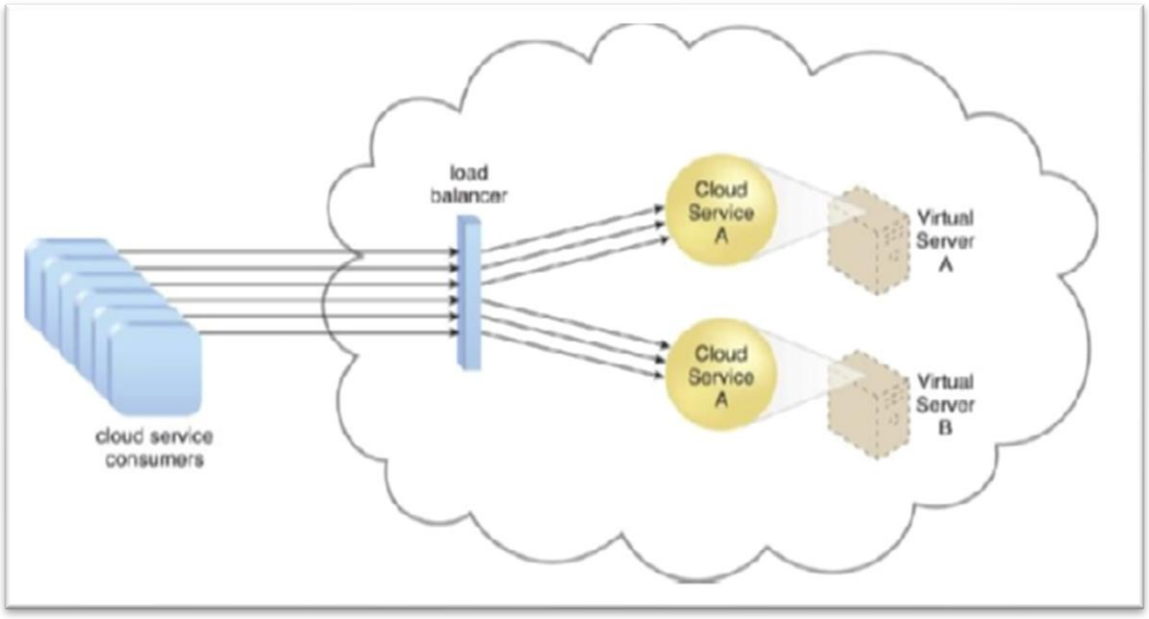
Two types of encryptions exist: symmetric and asymmetric. In symmetric encryption, the same key is utilized for both encryption and decryption, while in asymmetric encryption, different public and private keys are employed. Symmetric encryption ensures confidentiality, but non-repudiation is not guaranteed. On the other hand, asymmetric encryption provides non-repudiation and authenticity, but it does not guarantee confidentiality. Hashing is a mechanism that provides one-way data protection, making it useful when there is no unlocking key available.

Cloud Computing Architecture

Cloud computing architecture is a formalized approach that defines different functional domains within cloud environments by creating distinct solutions composed of interactions, behaviors, and combinations of cloud computing mechanisms and other specialized components. It is important to note that security architectures or models that involve cloud security mechanisms are not included in the upcoming chapters. These topics are discussed separately in a dedicated series on cloud security.

Fundamental Cloud Architectures

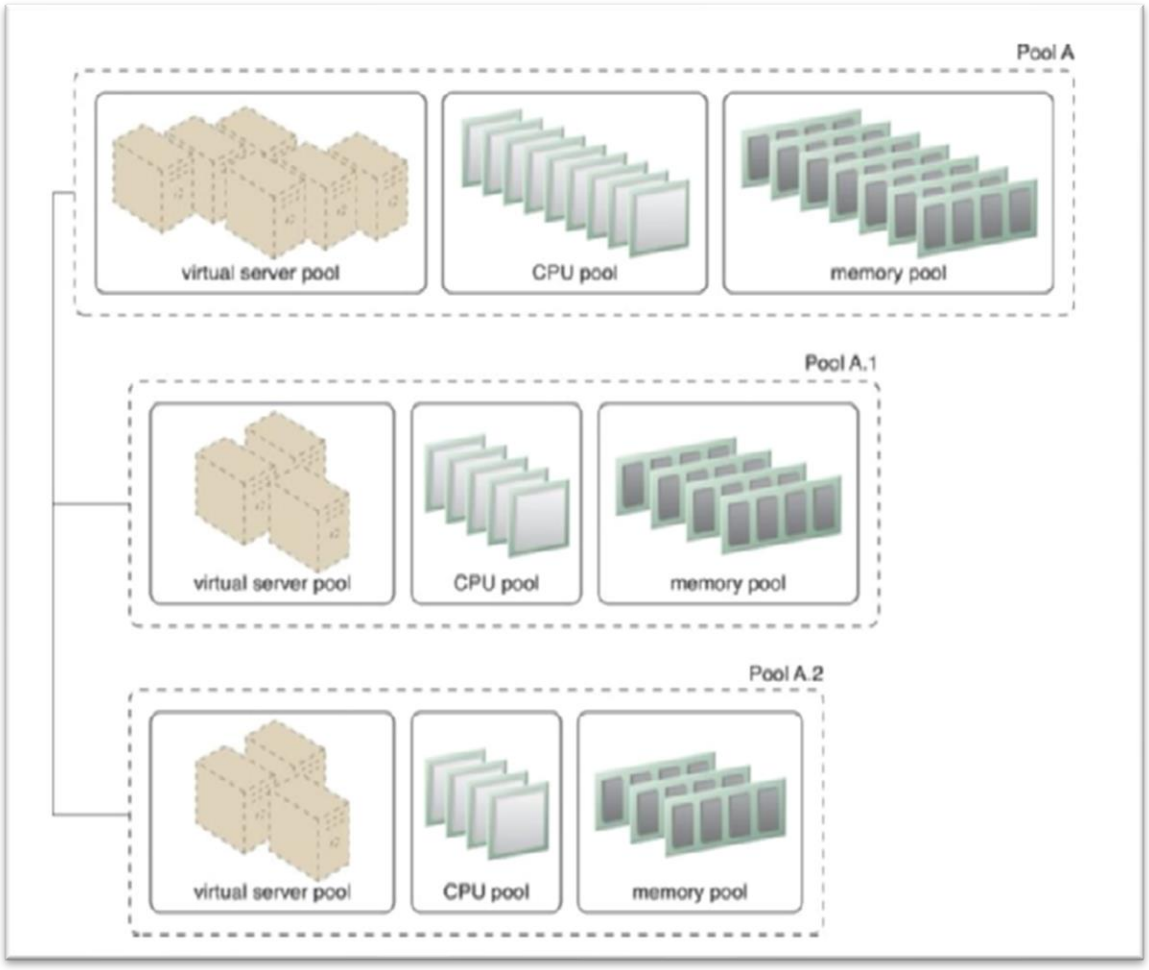
This chapter provides an introduction to and description of various foundational cloud architectural models that exemplify common usage and characteristics of contemporary cloud-based environments. The chapter explores the involvement and importance of different combinations of cloud computing mechanisms in relation to these architectures.



A redundant copy of Cloud Service A is implemented on Virtual Serve

The workload distribution architecture is based on horizontally scaling IT resources 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. This architecture reduces IT resource over-utilization and under-utilization to an extent dependent upon the sophistication of the load-balancing algorithms and runtime logic. 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.

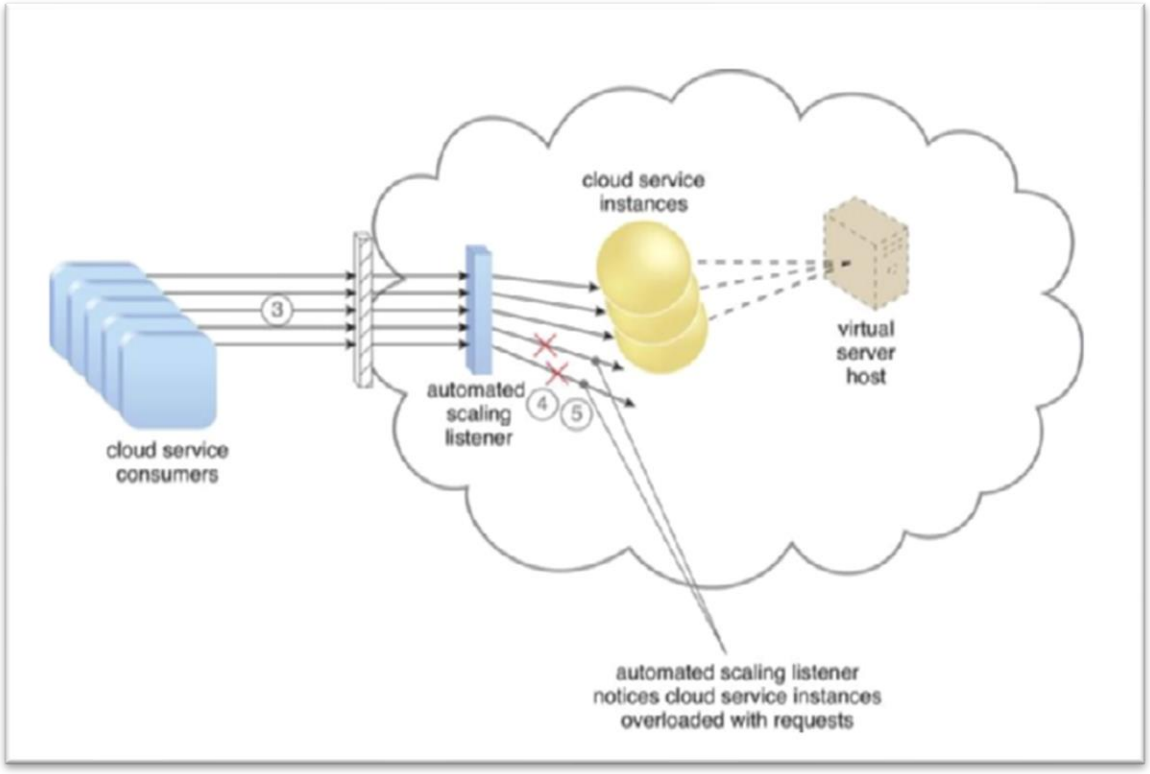
Load-balancing systems applied to specific IT resources usually produce specialized variations of this architecture that incorporate aspects of load balancing, such as the service load-balancing architecture, the load-balanced virtual server architecture, and the load-balanced virtual switches architecture. In addition to the base load balancer mechanism, and the virtual server and cloud storage device mechanisms to which load balancing can be applied, several other mechanisms can also be part of this cloud architecture, including the Audit Monitor, Cloud Usage Monitor, Hypervisor, Logical Network Perimeter, Resource Cluster, and Resource Replication.



Nested Pools

The 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. Common examples of resource pools are physical server pools, virtual server pools, storage pools, network pools, CPU pools, and pools of physical RAM. 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. Resource pools can become highly complex, with multiple sub-pools.

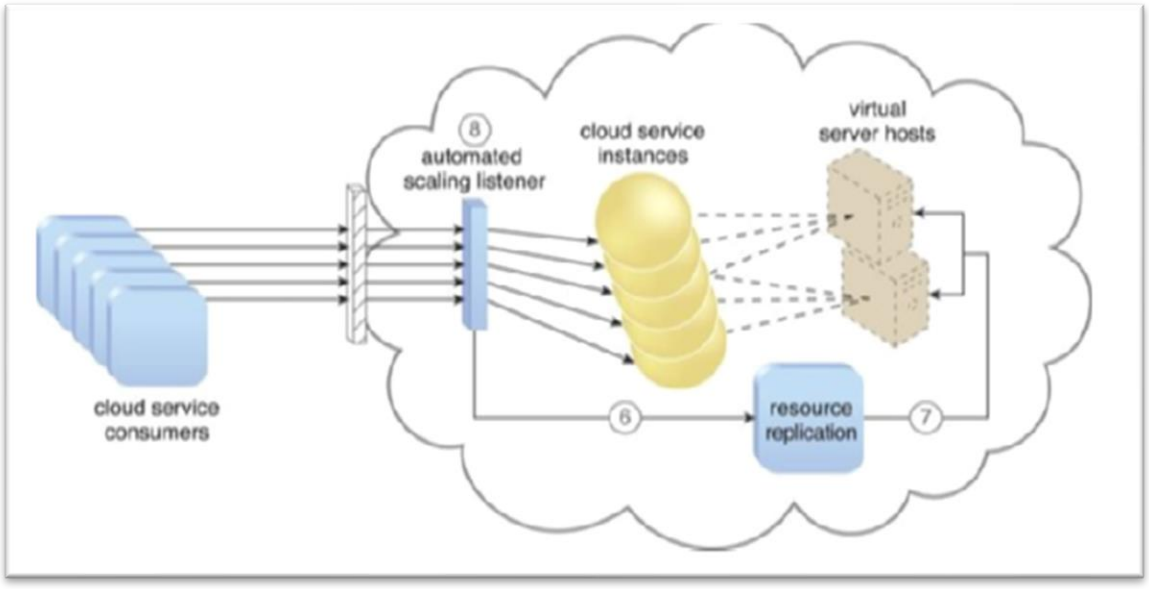
The dynamic scalability architecture is an approach that automates the allocation of IT resources based on predefined scaling conditions. This enables the system to efficiently allocate IT resources as per the usage demand, without the need for manual intervention. An automated scaling listener is configured with workload thresholds that determine when new IT resources should be added to workload processing. This mechanism can be provided with logic that determines how many additional IT resources can be dynamically provided based on a given cloud consumer's provisioning contract.



Cloud service consumers increases

Dynamic scalability can be achieved through three types of scaling: dynamic horizontal scaling, dynamic vertical scaling, and dynamic relocation. Dynamic horizontal scaling involves scaling IT resource instances in and out 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 involves scaling up or down the processing capacity of a single IT resource instance when required. Dynamic relocation involves relocating an IT resource to a host with more capacity.

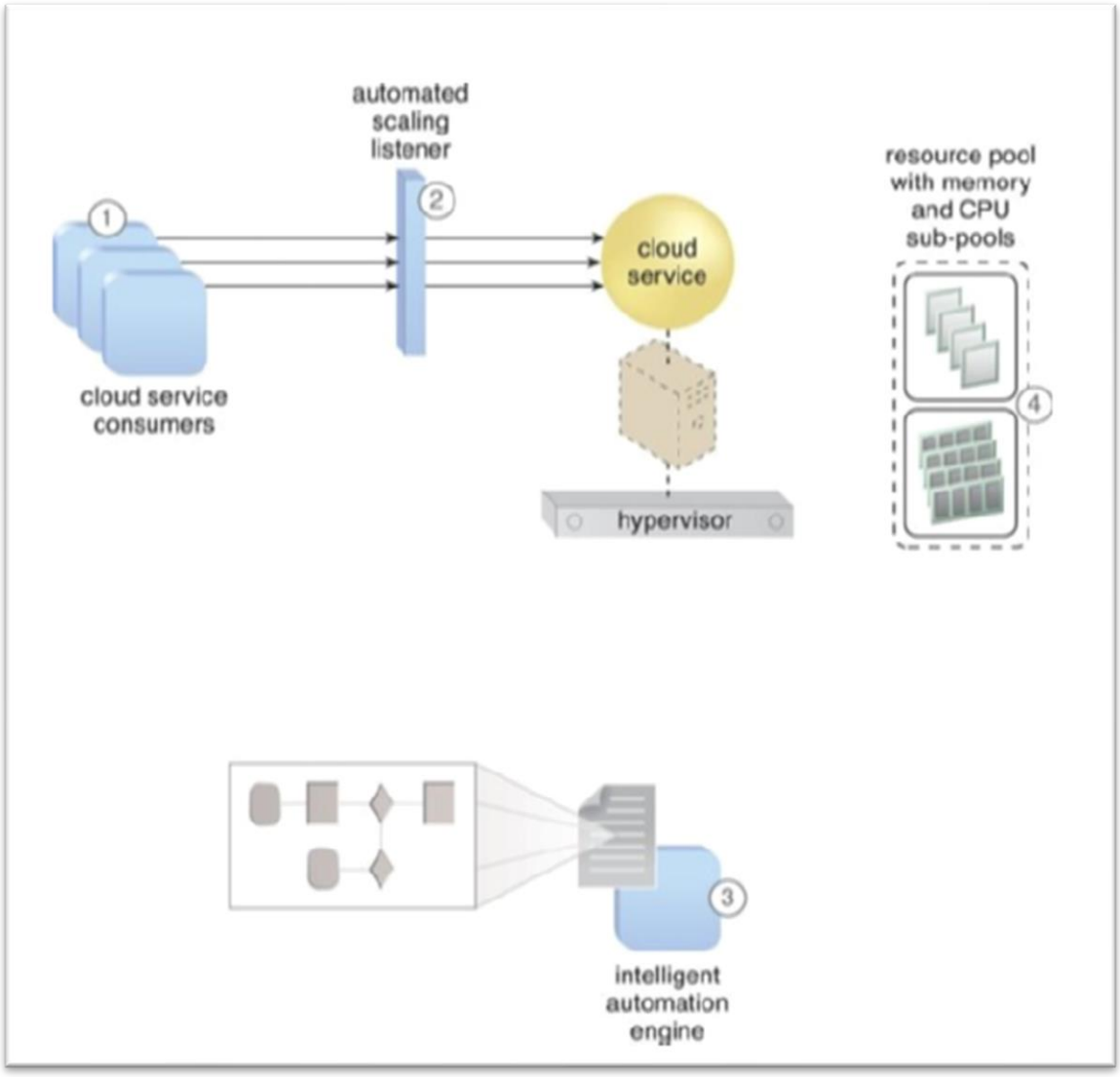


The automated scaling listener

The dynamic scalability architecture can be applied to various IT resources, including virtual servers and cloud storage devices. Specialized cloud usage monitors, hypervisors, and pay-per-use monitors are some of the mechanisms that can be used in this architecture.

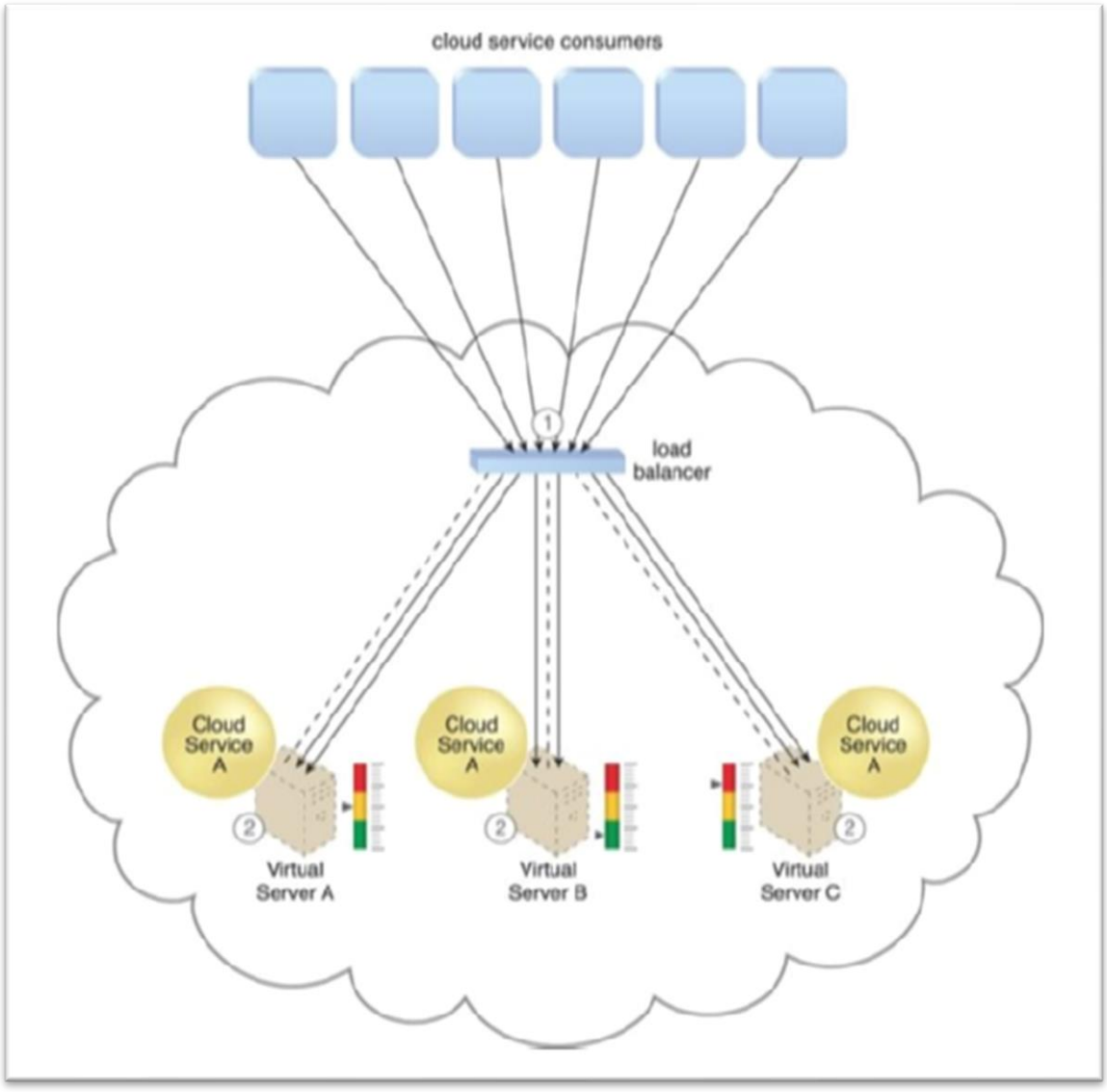
The elastic resource capacity architecture is related to the dynamic provisioning of virtual servers, which allows the allocation and reclamation of CPUs and RAM in immediate response to fluctuating processing requirements of hosted IT resources. Resource pools are used by scaling technology that interacts with the hypervisor and/or VIM to retrieve and return CPU and RAM resources at runtime.

The virtual server is monitored in real time so that additional processing power can be leveraged from the resource pool via dynamic allocation before capacity thresholds are met.



Cloud service consumer

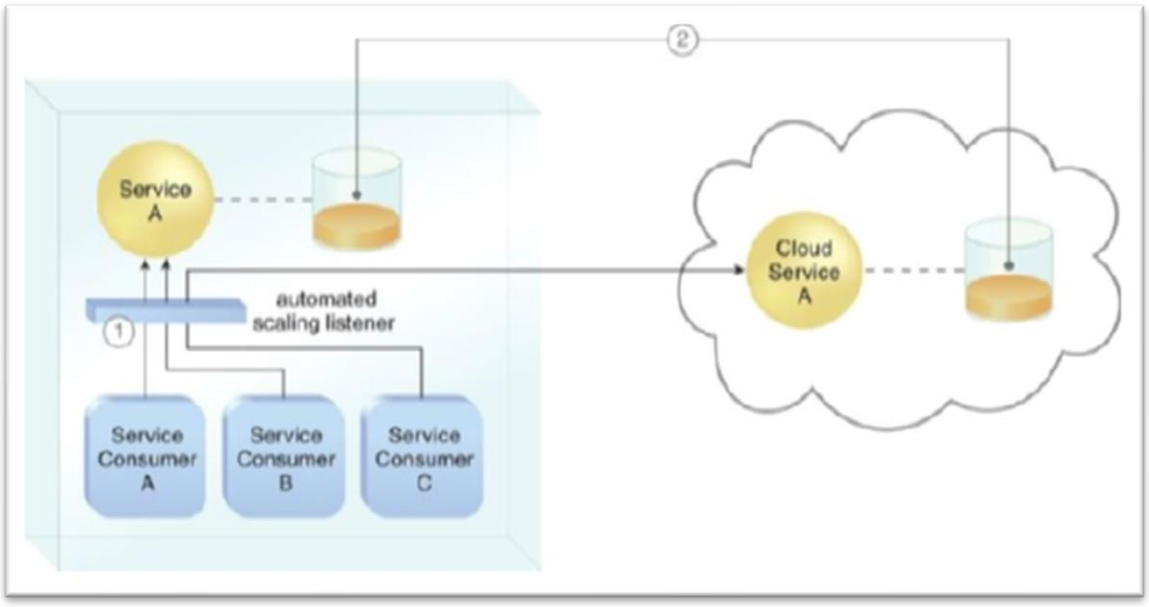
This type of cloud architecture can be designed so that the intelligent automation engine script sends its scaling request via the VIM instead of directly to the hypervisor. Virtual servers that participate in elastic resource allocation systems may require rebooting to take effect. The intelligent automation engine automates administration tasks by executing scripts that contain workflow logic.



Load balancer

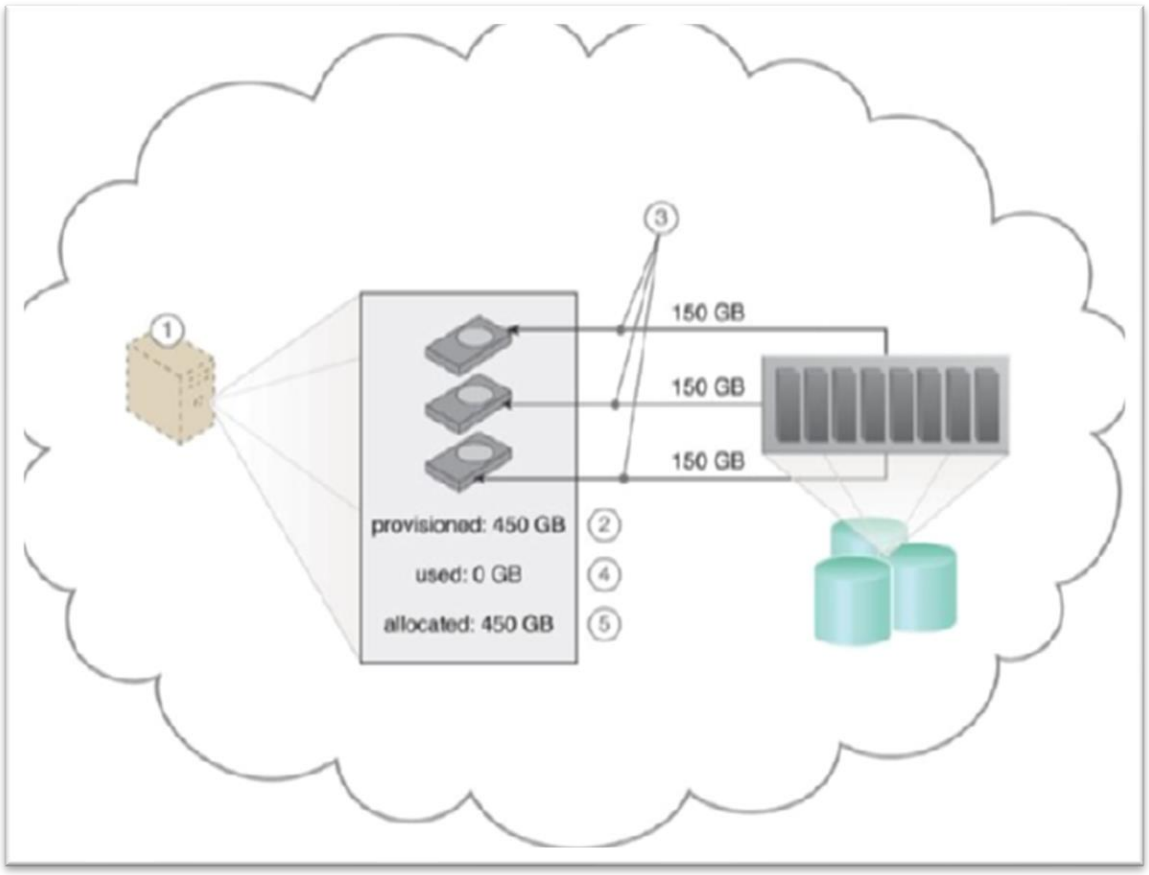
Specialized cloud usage monitors and pay-per-use monitors are some of the additional mechanisms that can be included in this cloud architecture. The former collects resource usage information on IT resources before, during, and after scaling to help define the future processing capacity thresholds of the virtual servers. The latter is responsible for collecting resource usage cost information as it fluctuates with the scaling of IT resources.

The Cloud Bursting Architecture is a scalable architecture that allows the automatic scaling of on-premise IT resources into a cloud when capacity thresholds are reached. The architecture deploys pre-deployed cloud-based IT resources, which remain inactive until required, and releases them when no longer needed. The automated scaling listener is used to determine when to redirect requests to cloud-based IT resources, while resource replication is used to maintain synchronicity between on-premise and cloud-based IT resources in relation to state information.



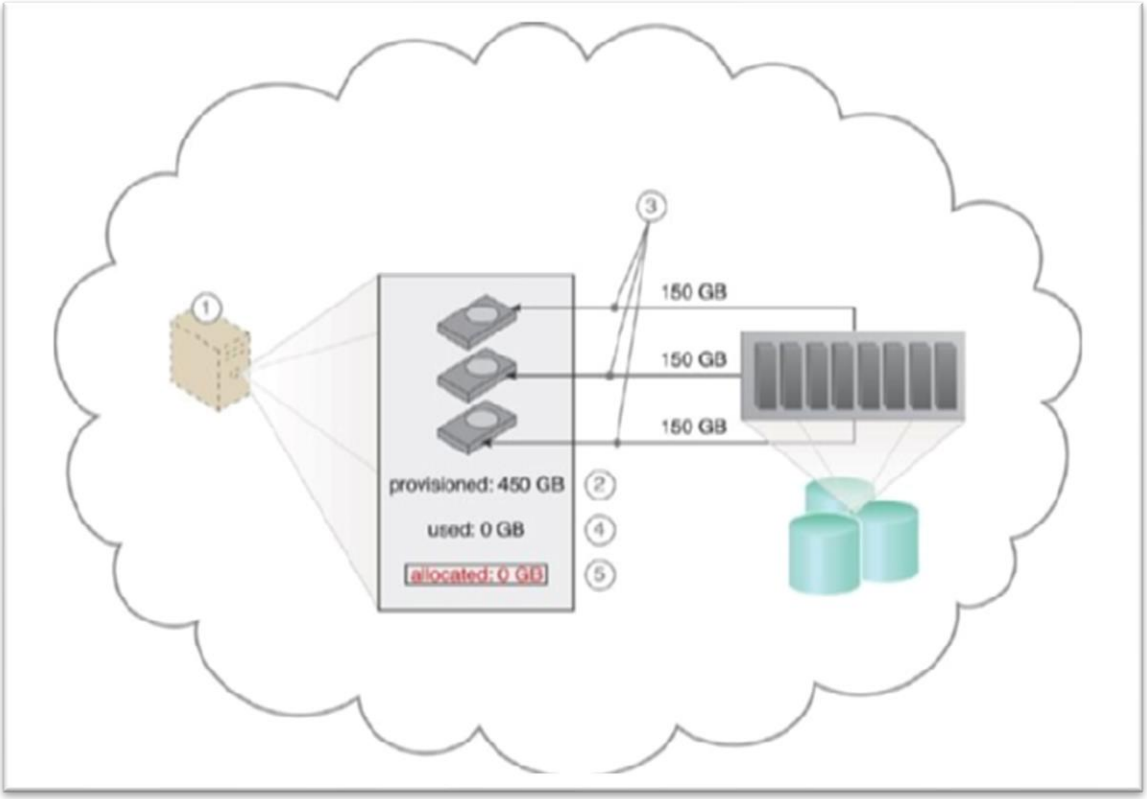
An automated scaling listener monitor

In addition to the automated scaling listener and resource replication, there are other mechanisms that can automate the burst in and out dynamics for this architecture, depending on the type of IT resource being scaled.



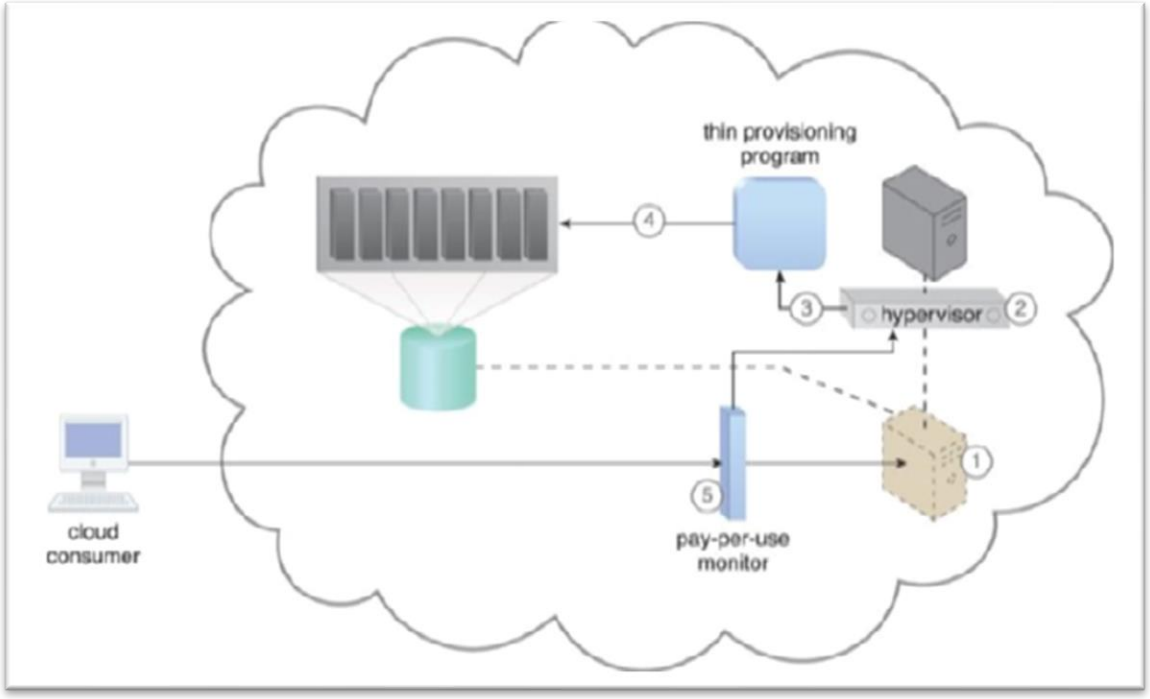
Requests a virtual server

The Elastic Disk Provisioning Architecture establishes a dynamic storage provisioning system that ensures that the cloud consumer is charged only for the actual amount of storage used.



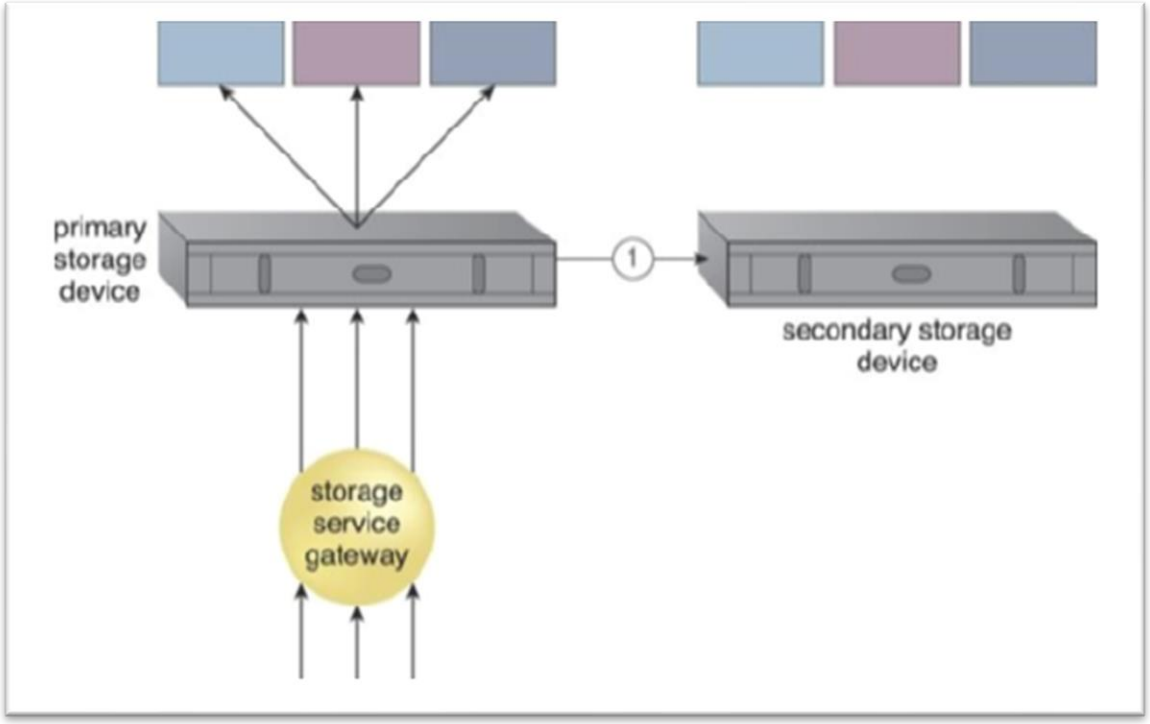
The cloud consumer requests a virtual server

The system uses thin-provisioning technology for the dynamic allocation of storage space and is supported by runtime usage monitoring for accurate usage data for billing purposes.



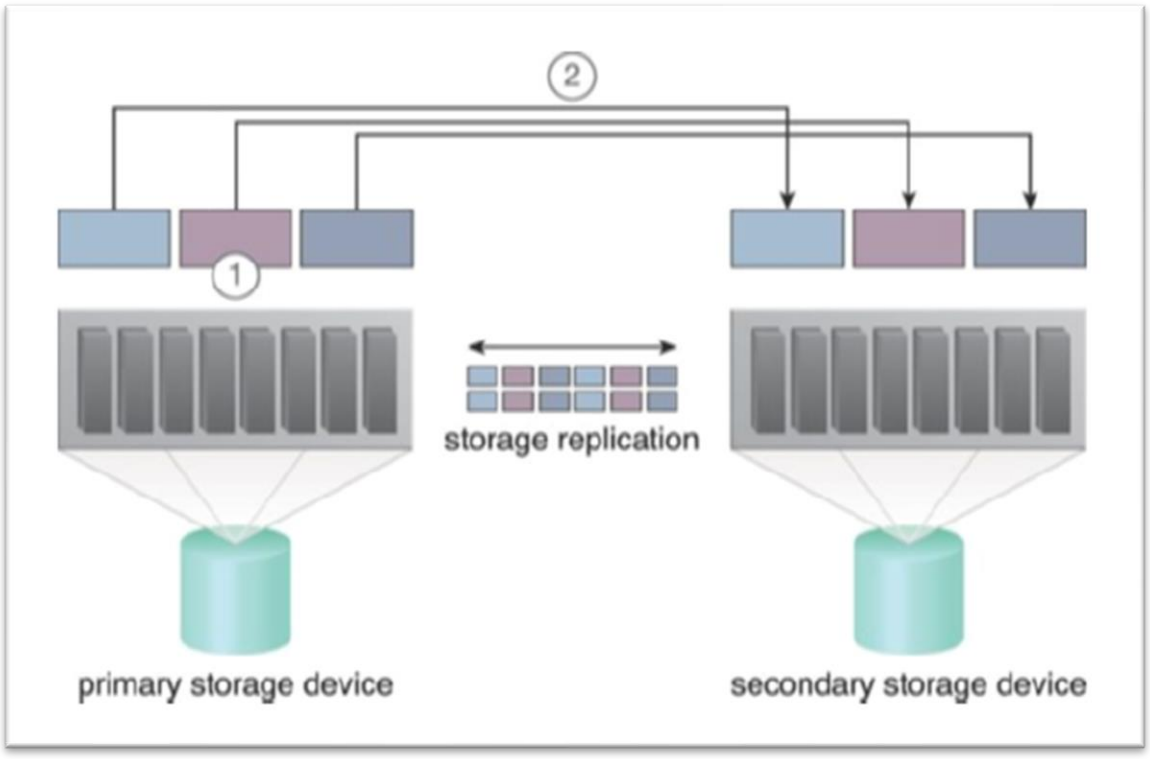
The provisioning of a new virtual server instance begins

Thin-provisioning software is installed on virtual servers that process dynamic storage allocation via the hypervisor, while the pay-per-use monitor tracks and reports granular billing-related disk usage data.



The primary cloud storage device is routinely replicated

Cloud usage monitors can be used to track and log storage usage fluctuations, while resource replication is part of an elastic disk provisioning system when conversion of dynamic thin-disk storage into static thick-disk storage is required.

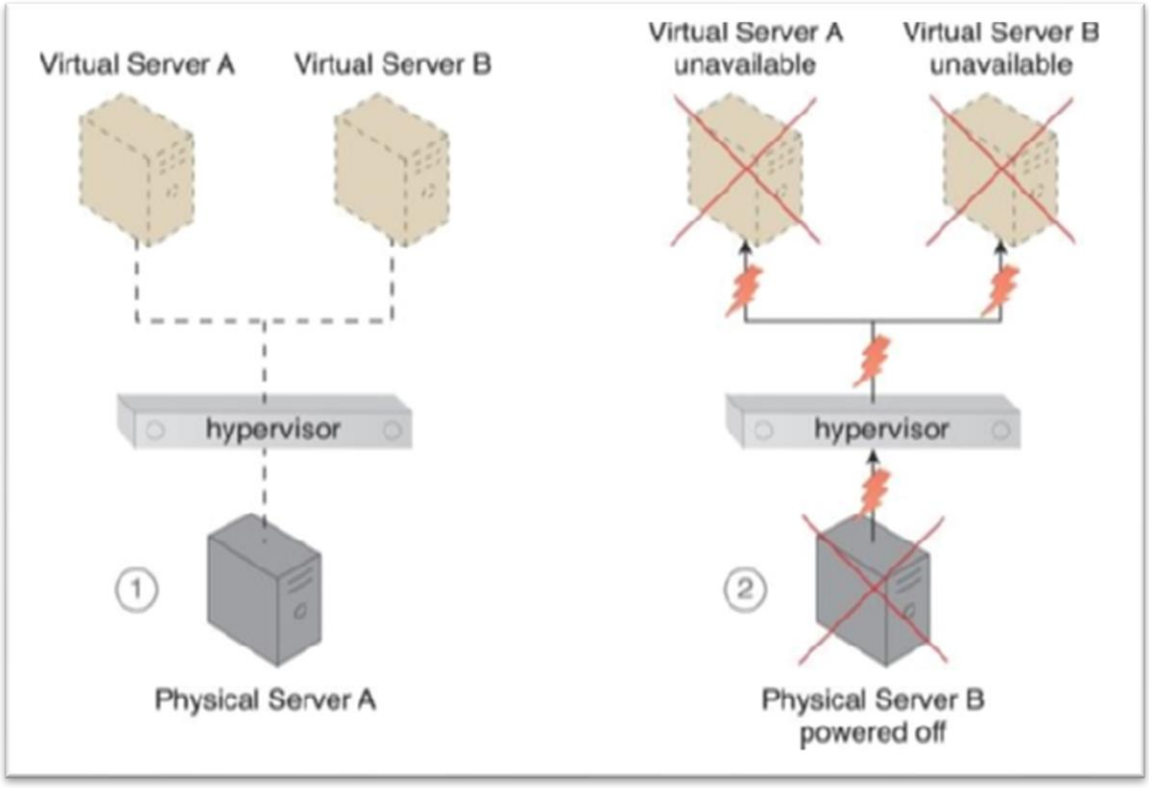


Storage replication is used to keep the redundant storage device synchronized

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 storage device fails. Logical unit numbers (LUNs) represent partitions of physical drives, and the storage service gateway is a component that acts as the external interface to cloud storage services and automatically redirects cloud consumer requests whenever the location of the requested data has changed.

Advanced Cloud Architectures

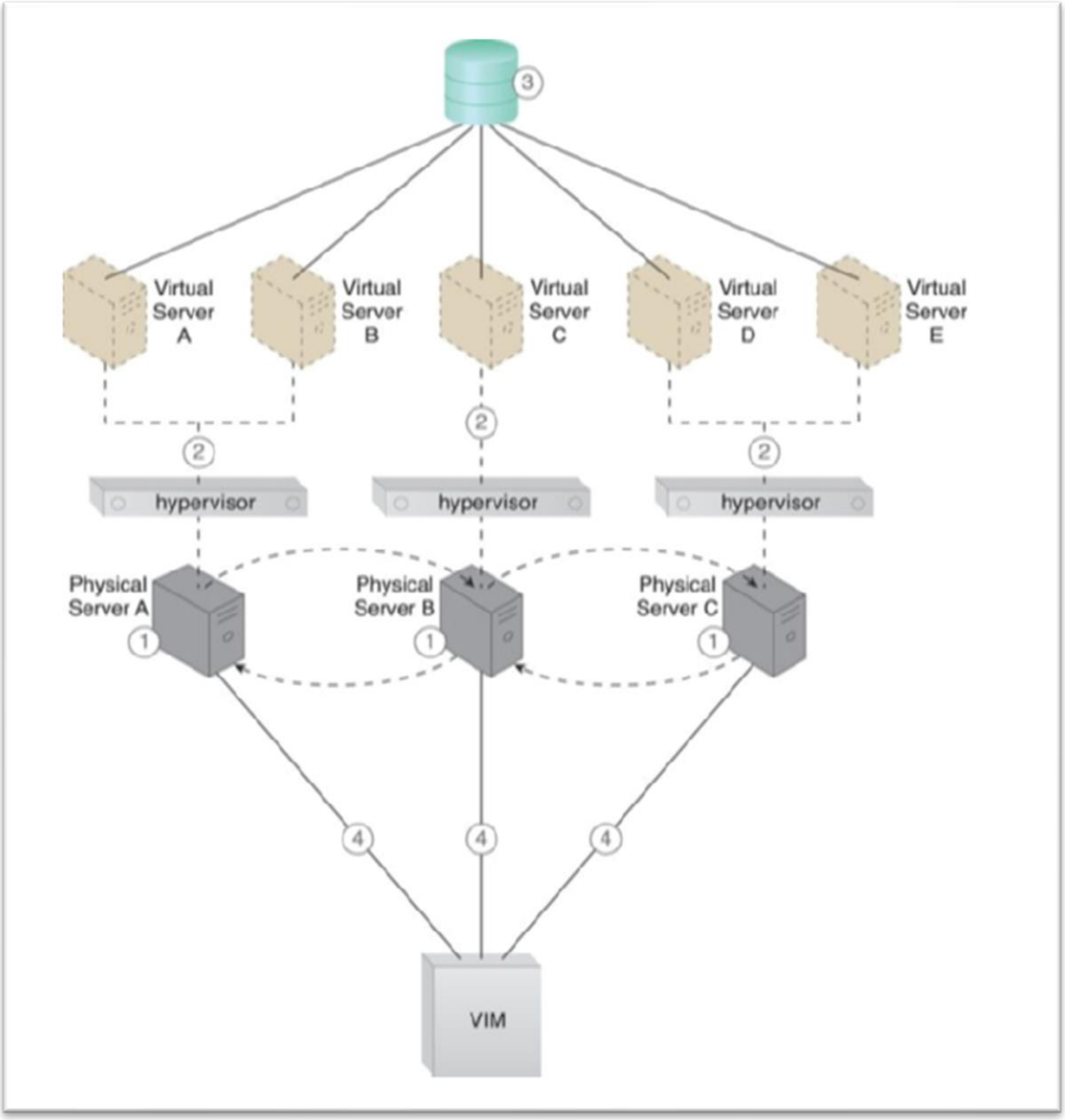
This chapter explores cloud technology architectures that consist of distinct and sophisticated layers, some of which can be built upon more foundational environments established by the architectural models covered. One such model is the Hypervisor Clustering Architecture.



Physical Server A is hosting a hypervisor

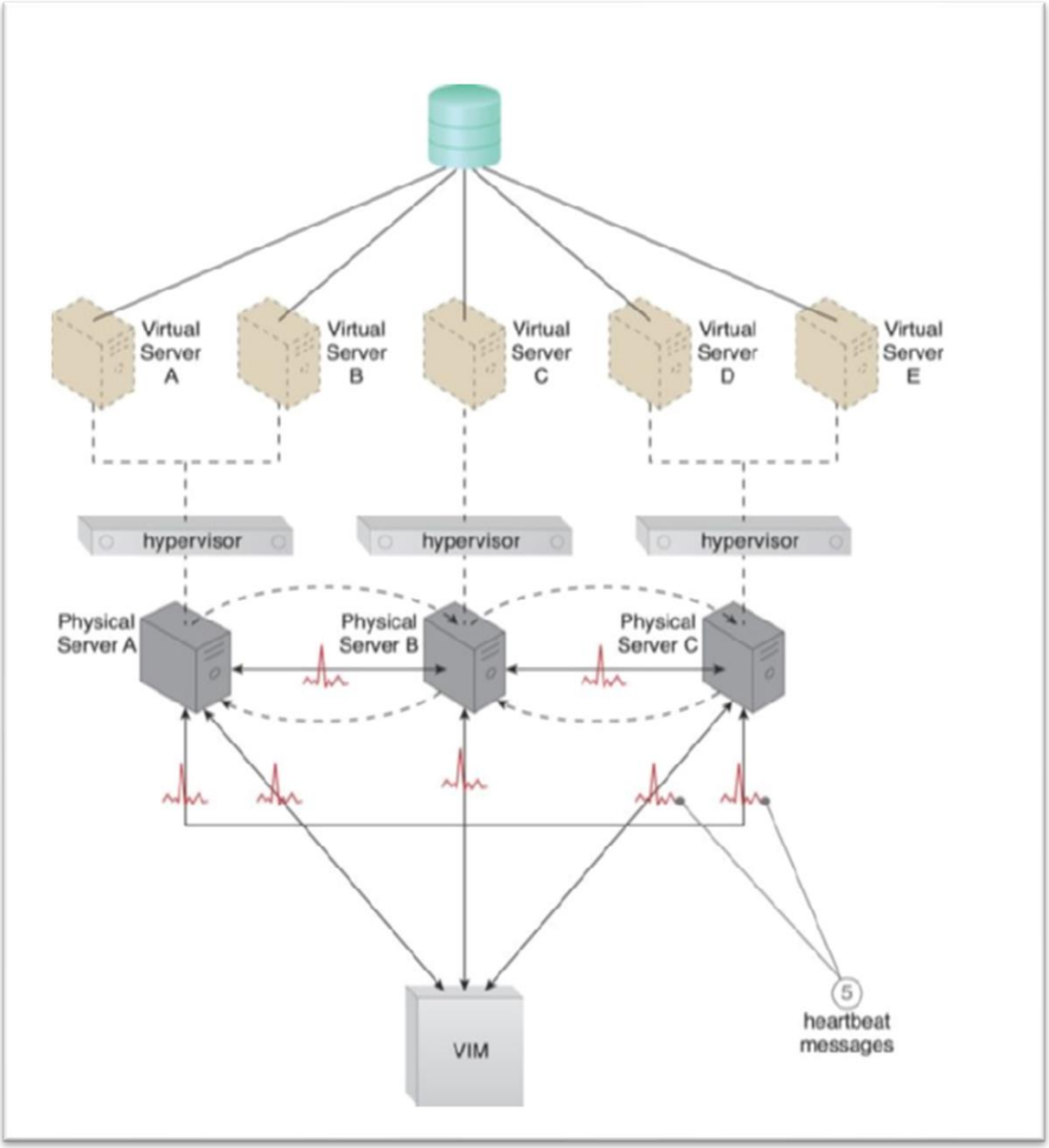
Hypervisors are responsible for creating and hosting multiple virtual servers, and any failure conditions that affect a hypervisor can cascade to its virtual servers. To prevent this, the Hypervisor Clustering Architecture establishes a high-availability cluster of hypervisors across multiple physical servers. In case of failure, hosted virtual servers can be moved to another physical server or hypervisor to maintain runtime operations.

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 to dynamically move the affected virtual servers to a new host.



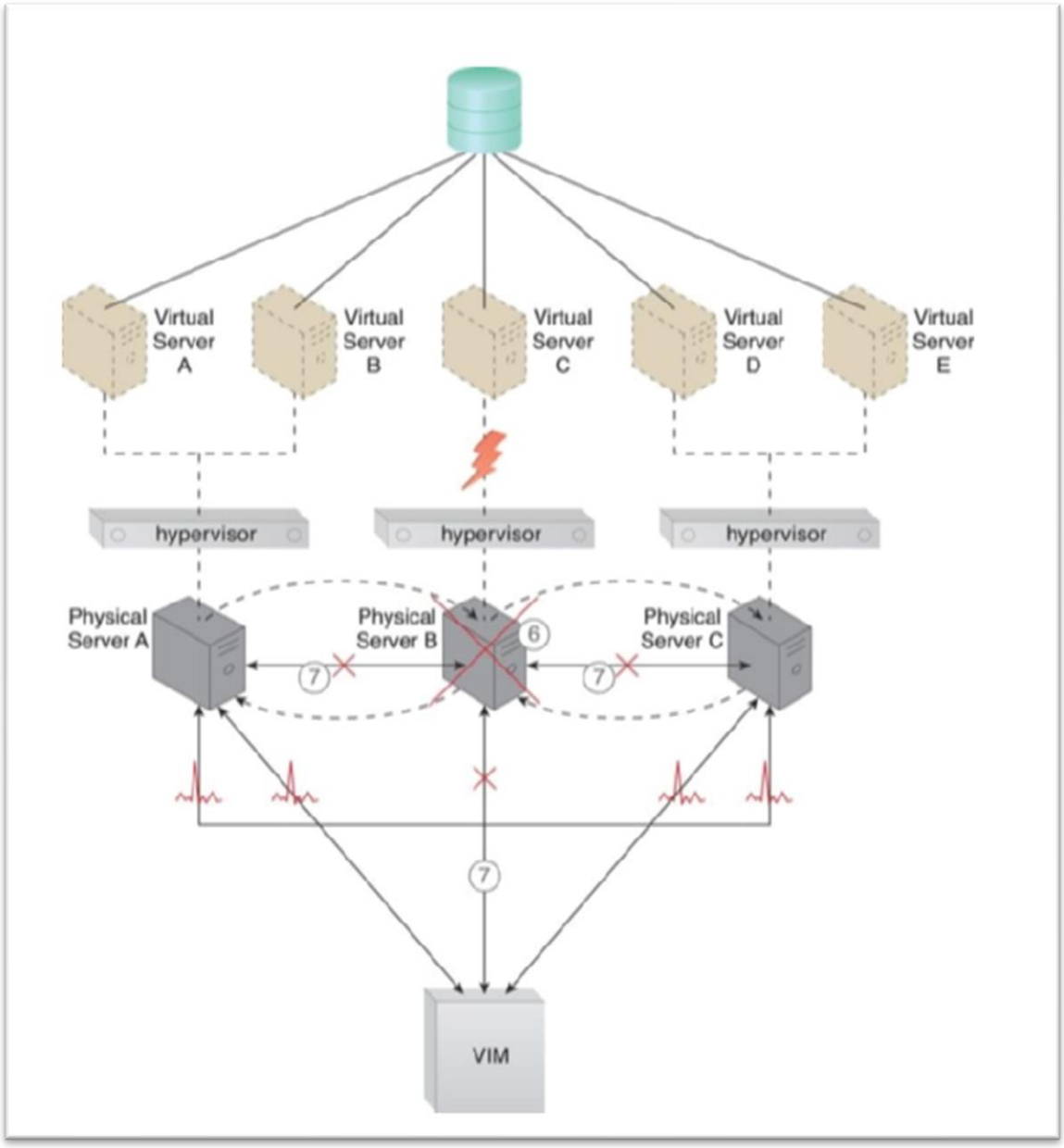
Hypervisors are installed on Physical Servers

Live VM migration is a system that can relocate virtual servers or instances at runtime, and the hypervisor cluster uses a shared cloud storage device to live-migrate virtual servers.



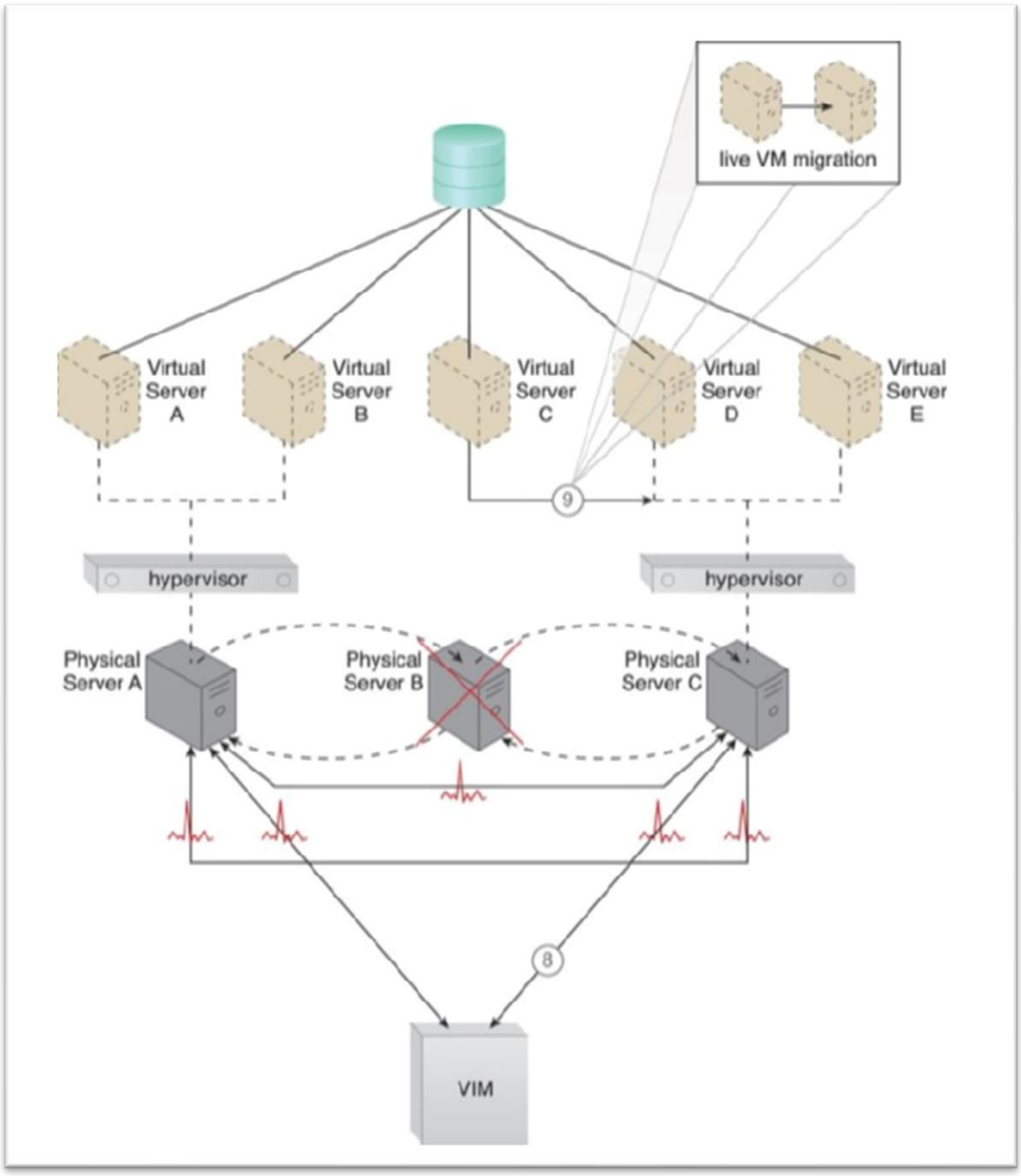
The physical servers exchange heartbeat messages

The Hypervisor Clustering Architecture also includes a logical network perimeter and resource replication to ensure that hypervisors of other cloud consumers are not accidentally included in each cluster and to replicate updates on any changes that occur in the cluster.



Physical Server B fails

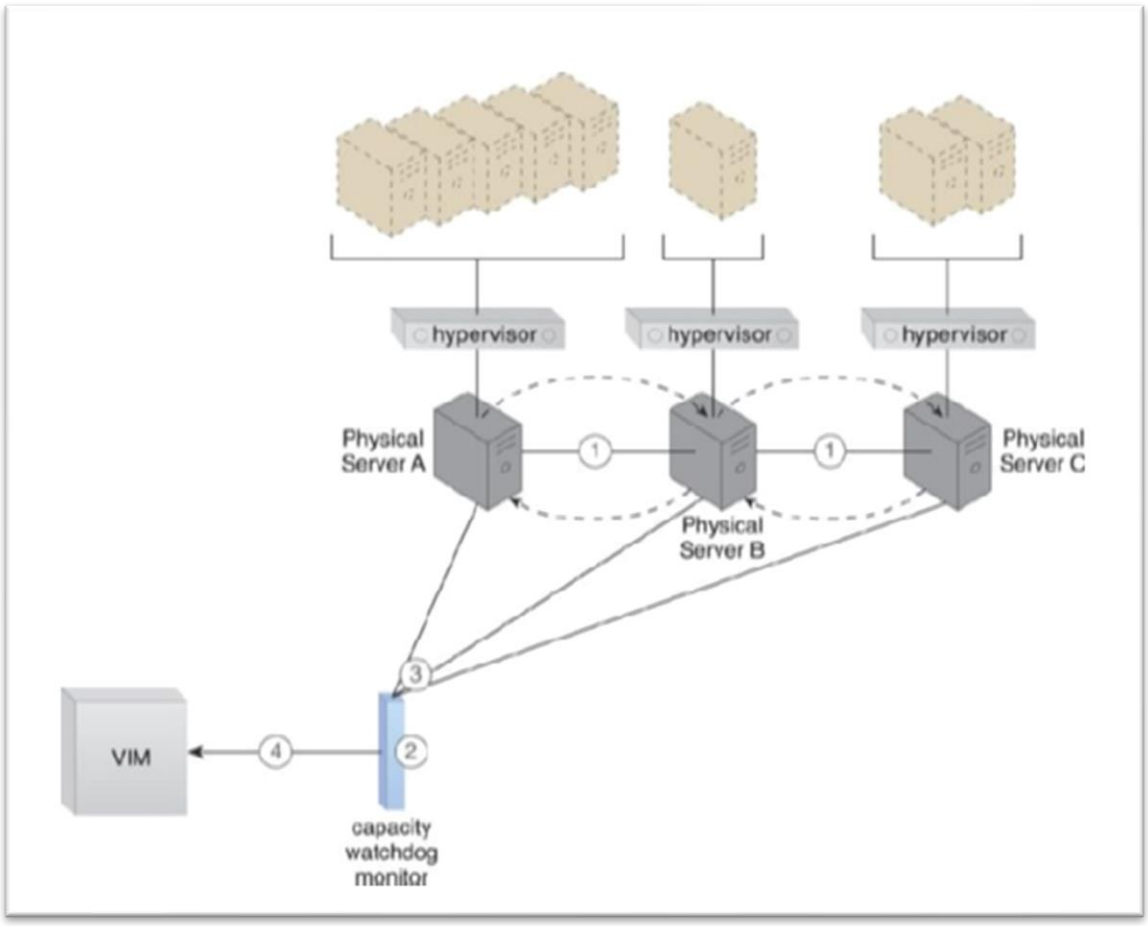
Another architectural model is the Load Balanced Virtual Server Instances Architecture. Keeping cross-server workloads evenly balanced between physical servers can be challenging, as a physical server can easily end up hosting more virtual servers or receive larger workloads than its neighboring physical servers.



The VIM chooses Physical Server

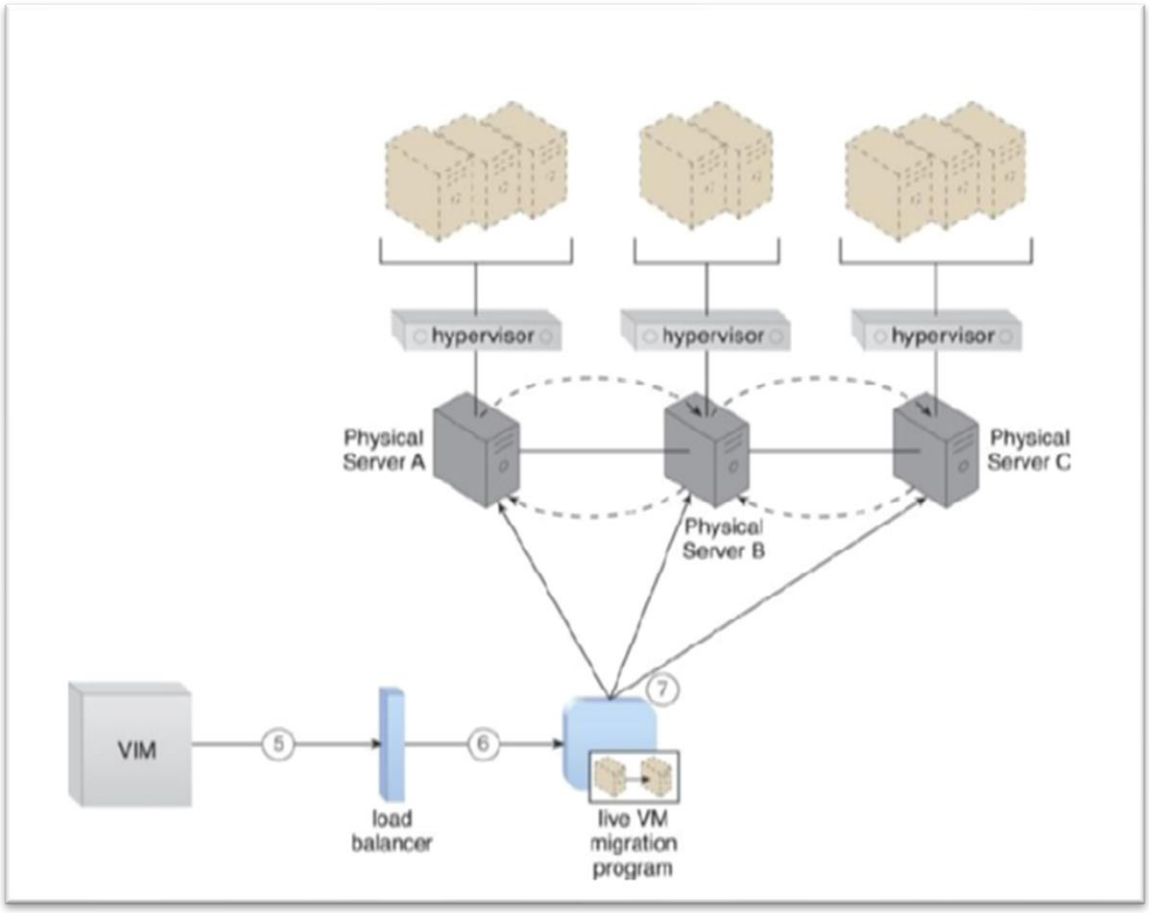
This can result in both physical server over and under-utilization, leading to ongoing performance challenges and constant waste.

The load-balanced virtual server instances architecture incorporates a capacity watchdog system that dynamically calculates virtual server instances and their workloads before distributing processing across available physical server hosts.



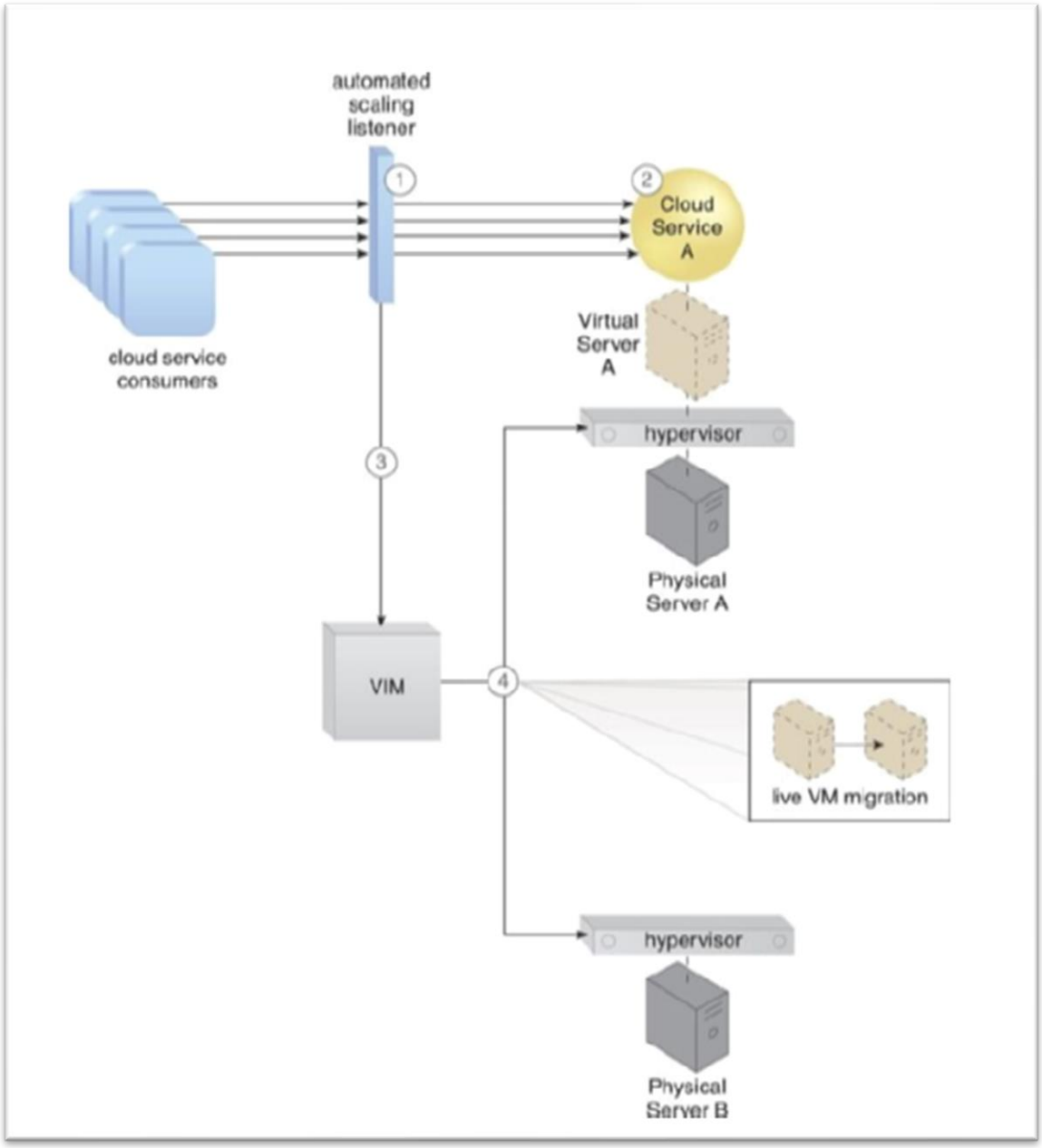
The hypervisor cluster architecture

The capacity watchdog system comprises a capacity watchdog cloud usage monitor, a live VM migration program, and a capacity planner. The capacity watchdog monitor tracks physical and virtual server usage and reports significant fluctuations to the capacity planner, responsible for dynamically calculating physical server computing capacities against virtual server capacity requirements.



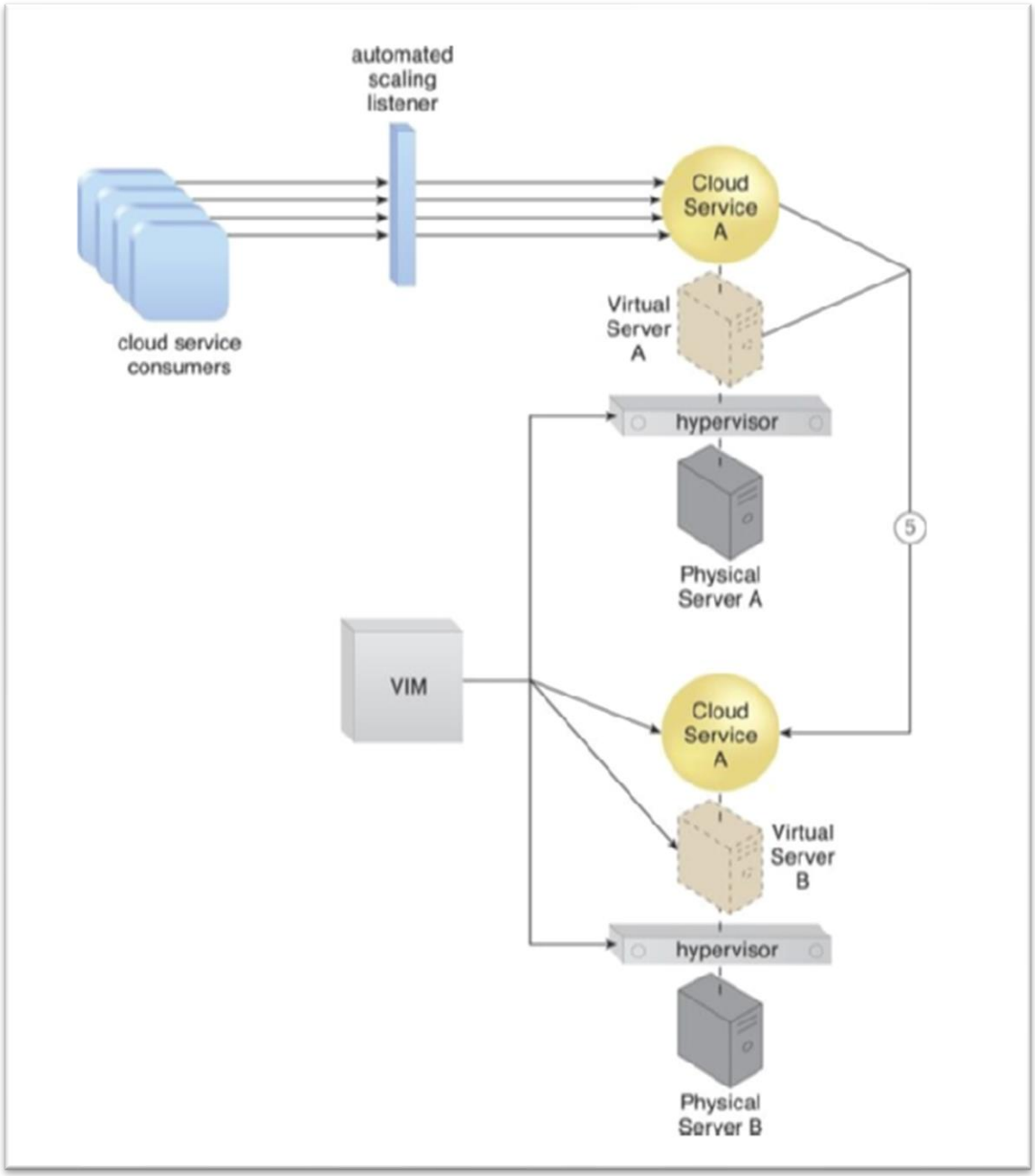
The VIM signals the load balancer

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. In addition to the hypervisor, resource clustering, virtual server, and capacity watchdog cloud usage monitor, several other mechanisms can be included in this architecture, such as an automated scaling listener, load balancer, logical network perimeter, and resource replication.



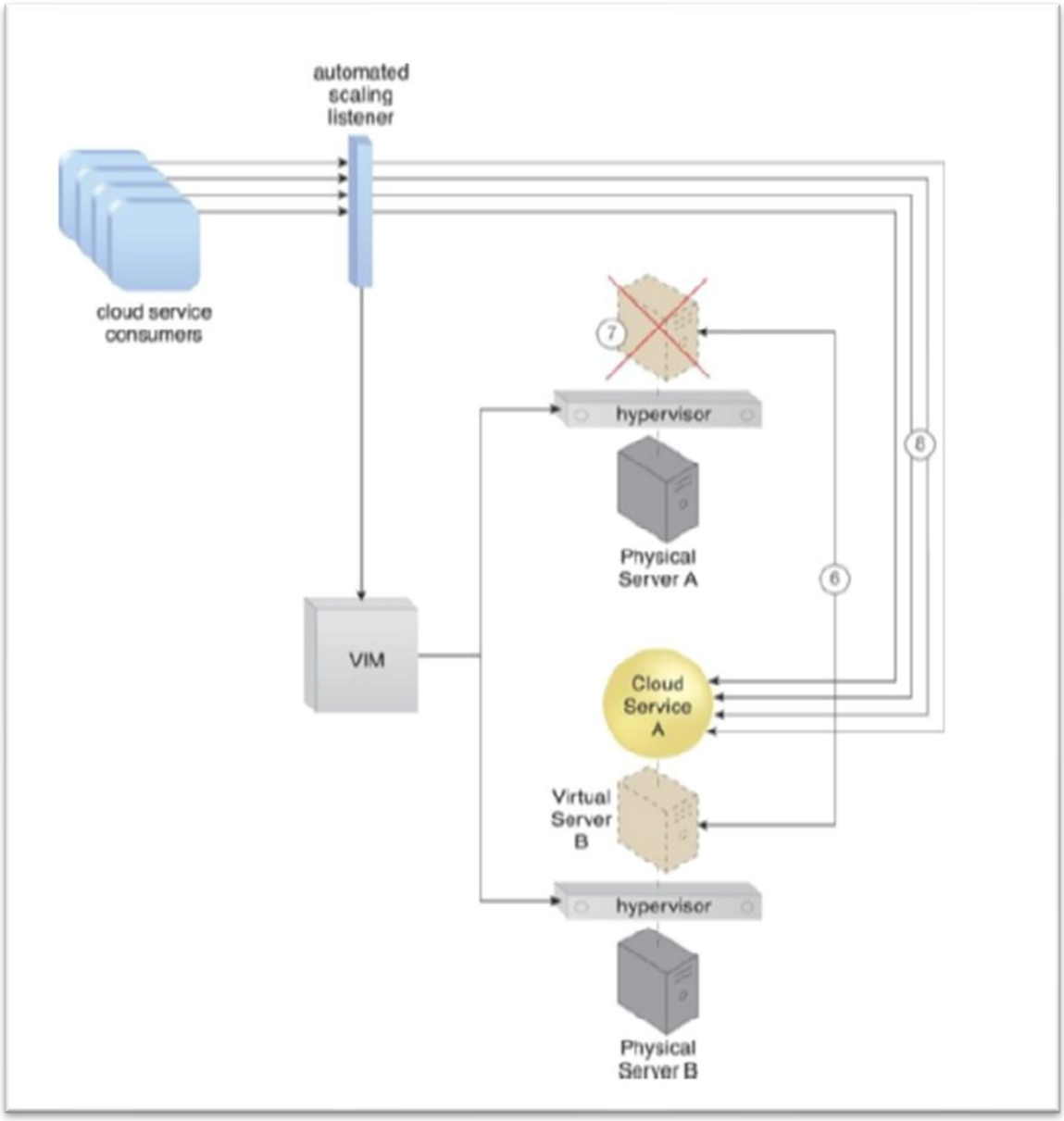
The automated scaling listener

The non-disruptive service relocation architecture aims to avoid any disruption when a cloud service becomes unavailable due to various reasons such as exceeding processing capacity or undergoing maintenance. The architecture establishes a system by which a predefined event triggers the duplication or migration of a cloud service implementation at runtime.



A second copy of the virtual server

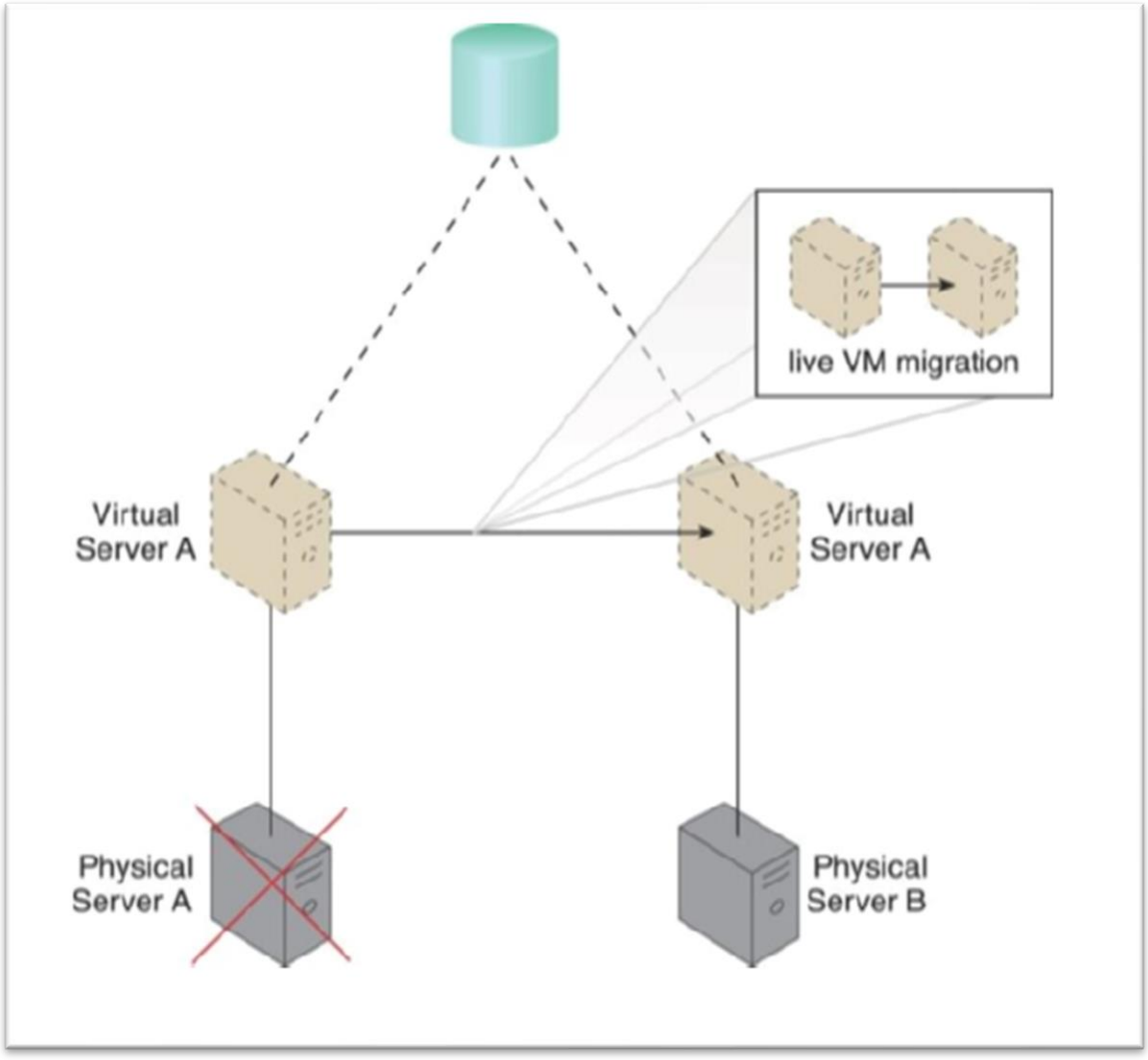
Instead of scaling cloud services in or out with redundant implementations, cloud service activity can be temporarily diverted to another hosting environment at runtime by adding a duplicate implementation to a new host. Virtual server migration can occur in two ways, depending on the location of the virtual server's disks and configuration.



The state of both virtual server instances is synchronized

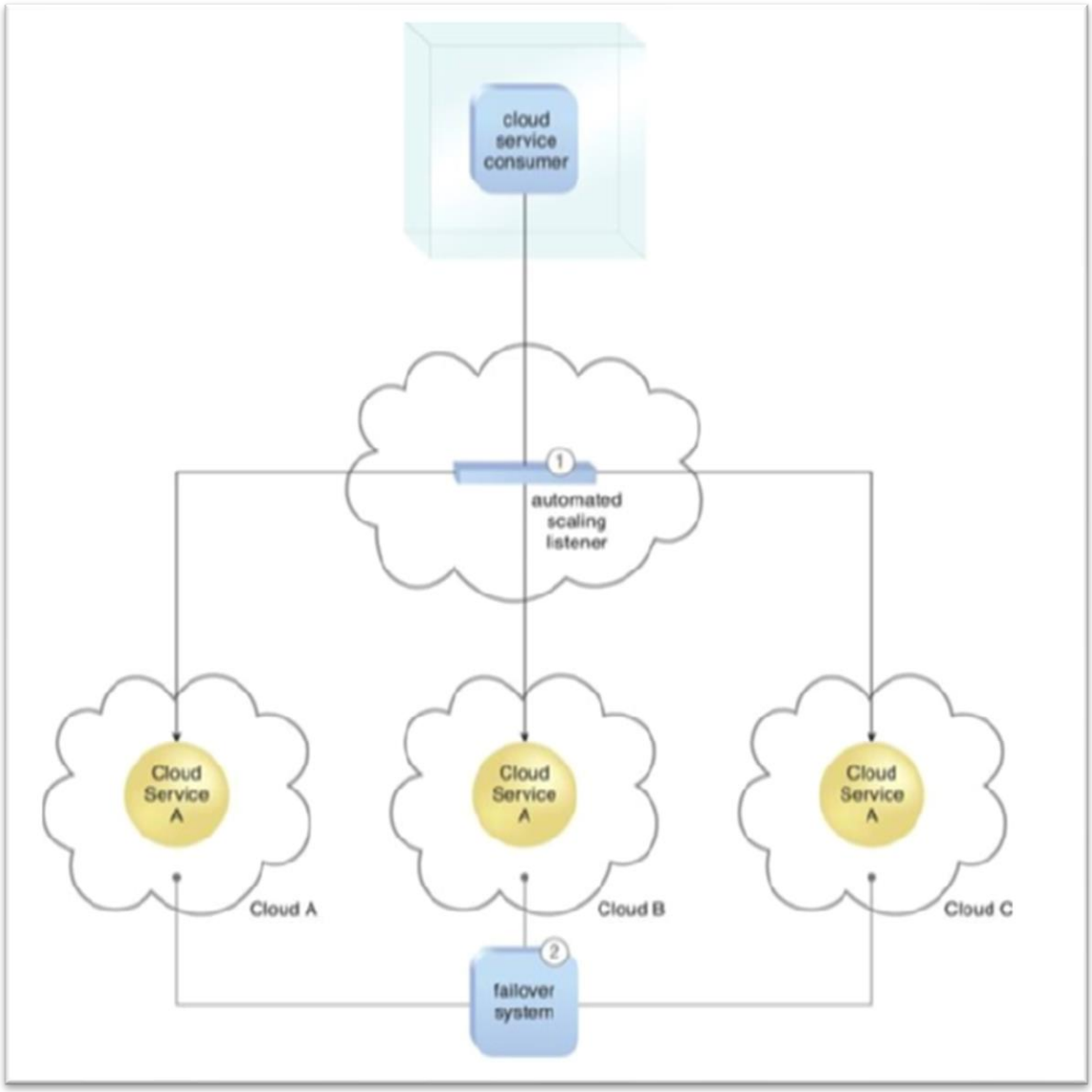
A copy of the virtual server disks is created on the destination host if the virtual server disks are stored on a local storage device or non-shared remote storage devices are attached. The automated scaling listener and/or load balancer mechanisms can be used to trigger a temporary redirection of cloud service consumer requests in response to scaling and workload distribution requirements.

Zero Downtime Architecture is a sophisticated failover system that is capable of dynamically moving virtual servers to different physical server hosts in case of a failure of their original physical server. Physical servers can act as a single point of failure for the virtual servers that they host. As a result, if the physical server fails, the availability of any or all hosted virtual servers can be affected, which makes the issuance of zero downtime guarantees challenging. Multiple physical servers are assembled into a group controlled by a fault tolerance system that can switch activity from one physical server to another without interruption.



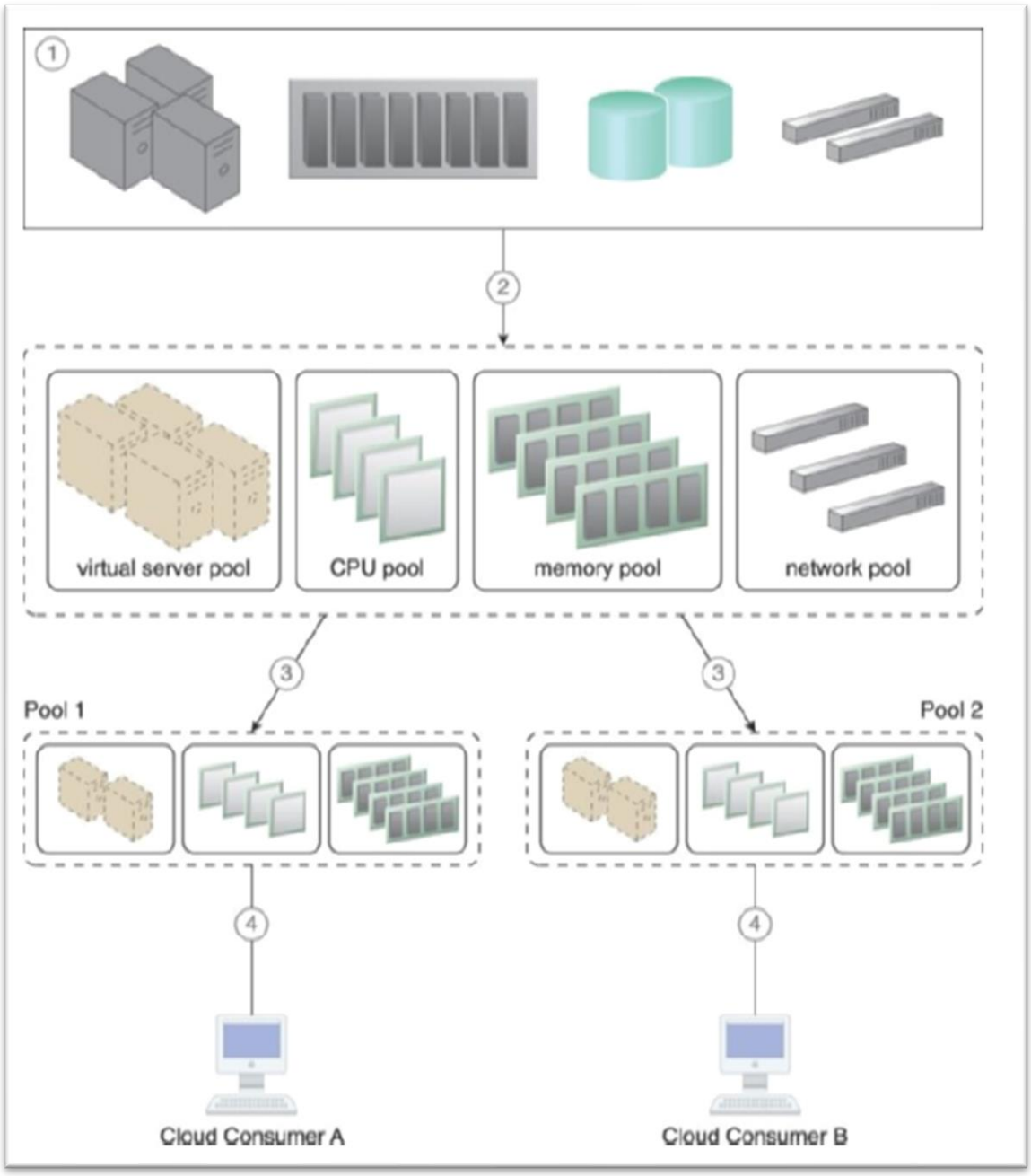
Physical Server fails triggering the live VM migration program

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.



An automated scaling listener control

Besides the failover system, cloud storage device, and virtual server mechanisms, other mechanisms can be part of this architecture, such as the Audit Monitor, the Cloud Usage Monitor, the Hypervisor, the Logical Network Perimeter, the Resource Cluster, and the Resource Replication.



A physical resource group is created

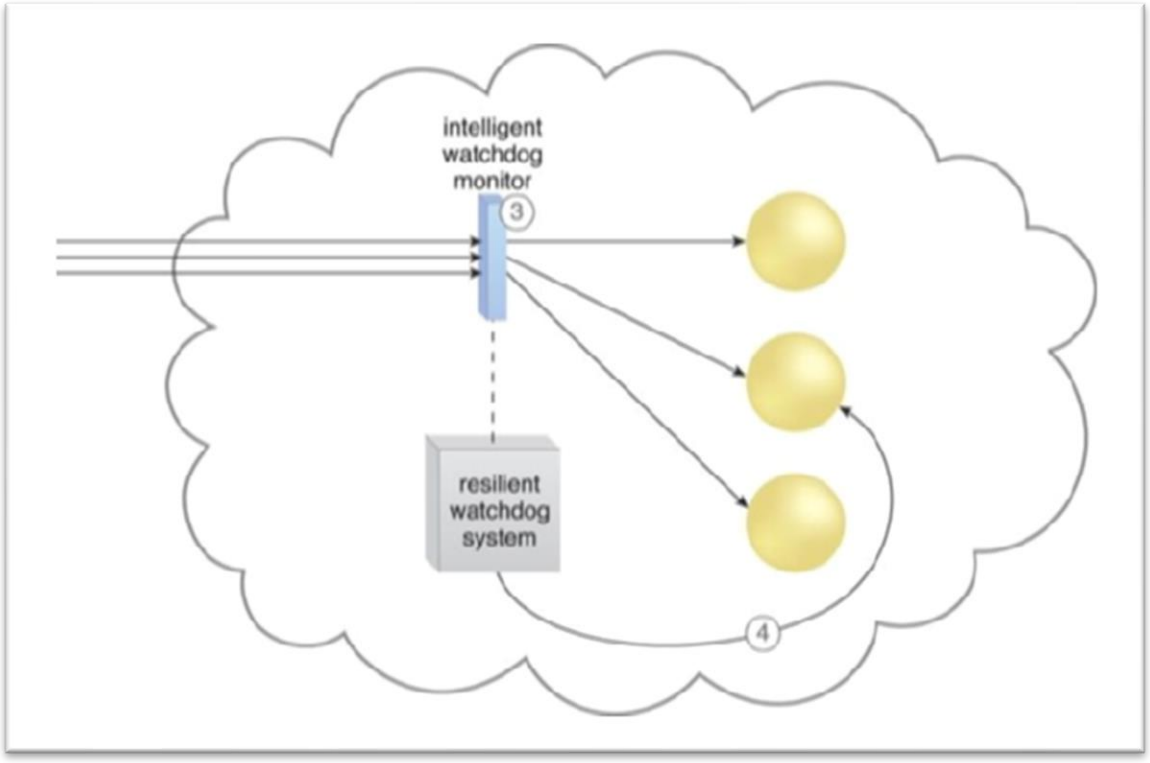
Cloud Balancing Architecture is a specialized architectural model that allows IT resources to be load-balanced across multiple clouds. Cross-cloud balancing of cloud service consumer requests can help improve the performance and scalability of IT resources, increase their availability and reliability, and improve load balancing and resource optimization. The cloud balancing functionality is primarily based on the combination of the automated scaling listener and failover system mechanisms. 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.

The automated scaling listener redirects cloud service consumer requests to one of several redundant IT resource implementations based on current scaling and performance requirements, while 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.

Note that if the manual synchronization of cross-cloud IT resource implementations is not possible, the resource replication mechanism may need to be incorporated to automate the synchronization.

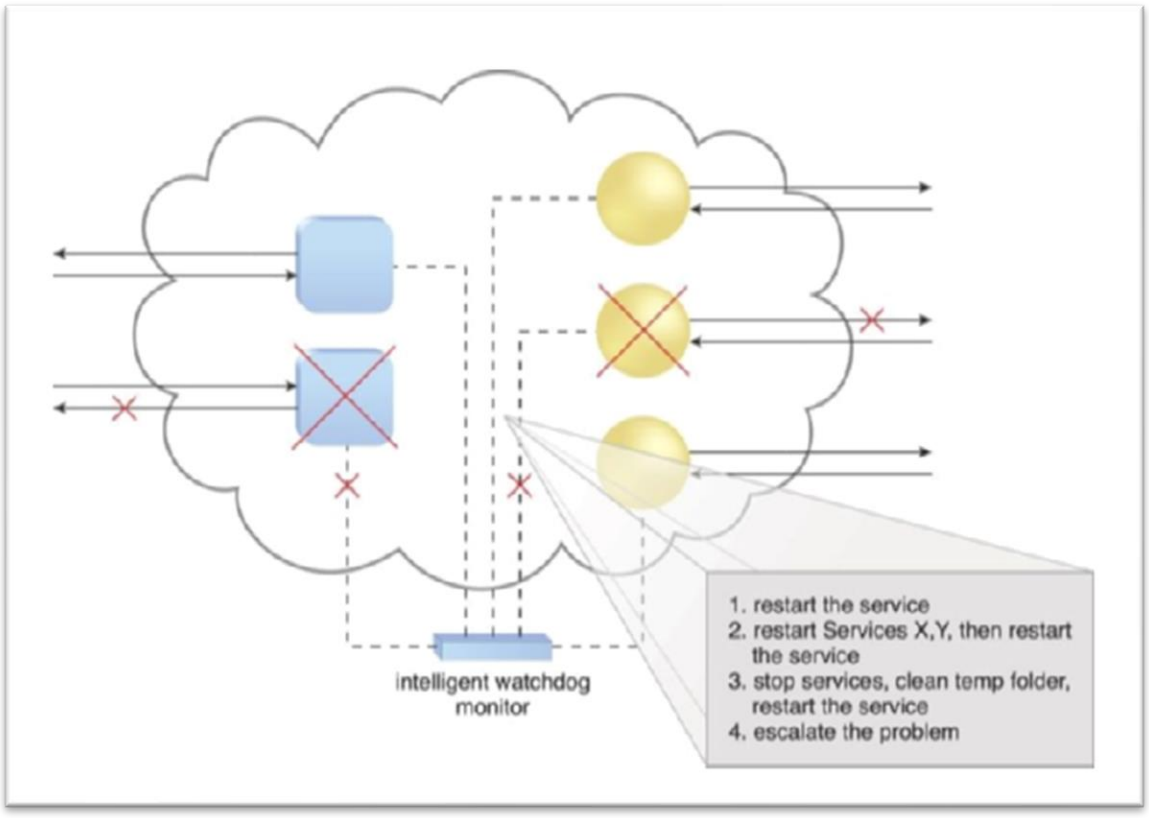
Depending on how IT resources are designed for shared usage and their available levels of capacity, concurrent access can lead to poor performance and data consistency issues. Therefore, it is essential to allocate IT resources on-demand to cloud service consumers, ensuring that they are available and can perform optimally when requested. Resource Reservation Architecture is a mechanism that provides for the allocation of IT resources on-demand to cloud service consumers. This architecture involves the creation of resource pools, each of which is assigned to a specific cloud service consumer. The resource reservation mechanism ensures that the available capacity of each resource pool is continuously monitored and dynamically adjusted to ensure that cloud service consumers receive the resources they require.

Resource Reservation Architecture also provides a mechanism for cloud service consumers to reserve IT resources on-demand. The resource reservation process involves the creation of a reservation request, which includes the resource type, capacity, and duration. Once the reservation request is approved, the corresponding IT resources are allocated to the requesting cloud service consumer. Resource Reservation Architecture can be used to optimize resource utilization, minimize resource contention, and provide predictable performance levels to cloud service consumers.



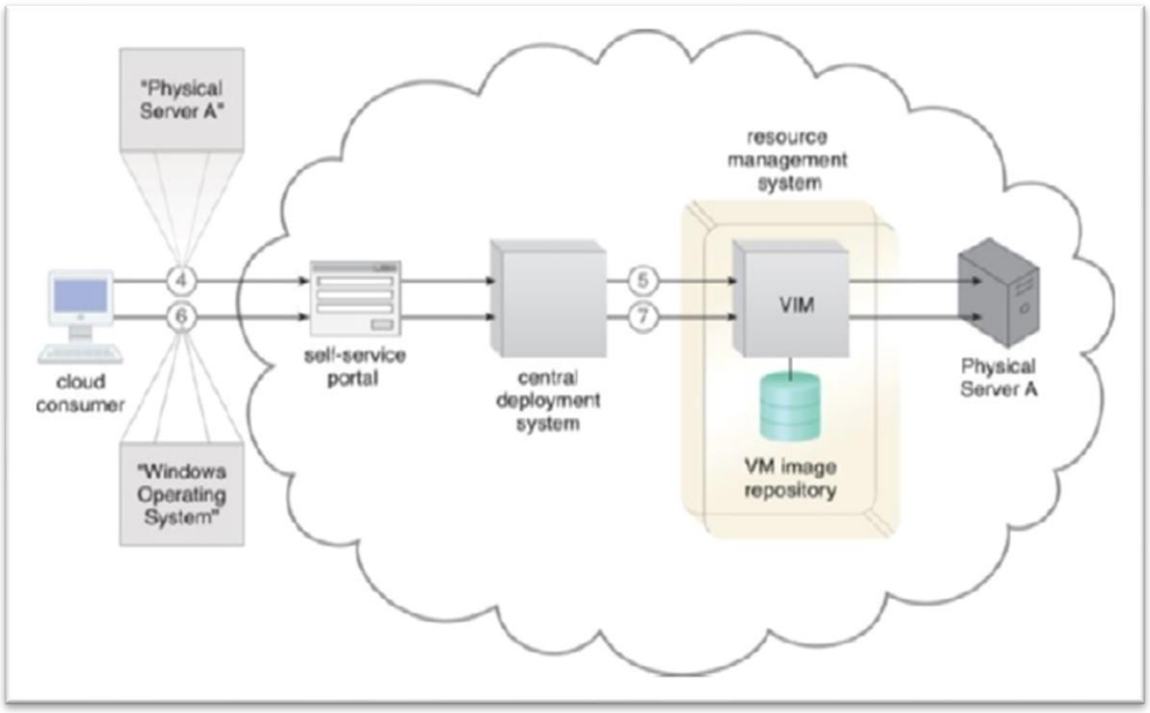
The intelligent watchdog monitor notifies the watchdog system

These IT resources can often experience failure conditions that require more than manual intervention to resolve, which can be inefficient and impractical. The resilient watchdog system monitors and responds to a wide range of pre-defined failure scenarios and escalates the issue if it cannot resolve itself.



The intelligent watchdog monitor refers to its pre-defined policies

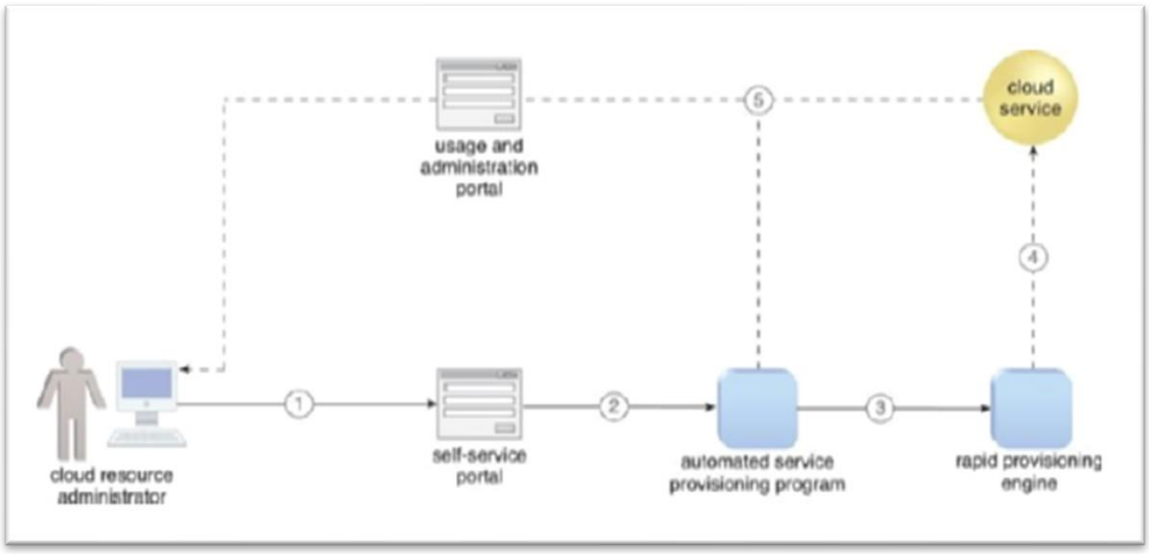
The intelligent watchdog monitor actively tracks IT resources and takes pre-defined actions in response to predefined events. Sequential recovery policies can be defined for each IT resource to determine the steps that the intelligent watchdog monitor needs to take when a failure condition occurs.



Physical server to provision

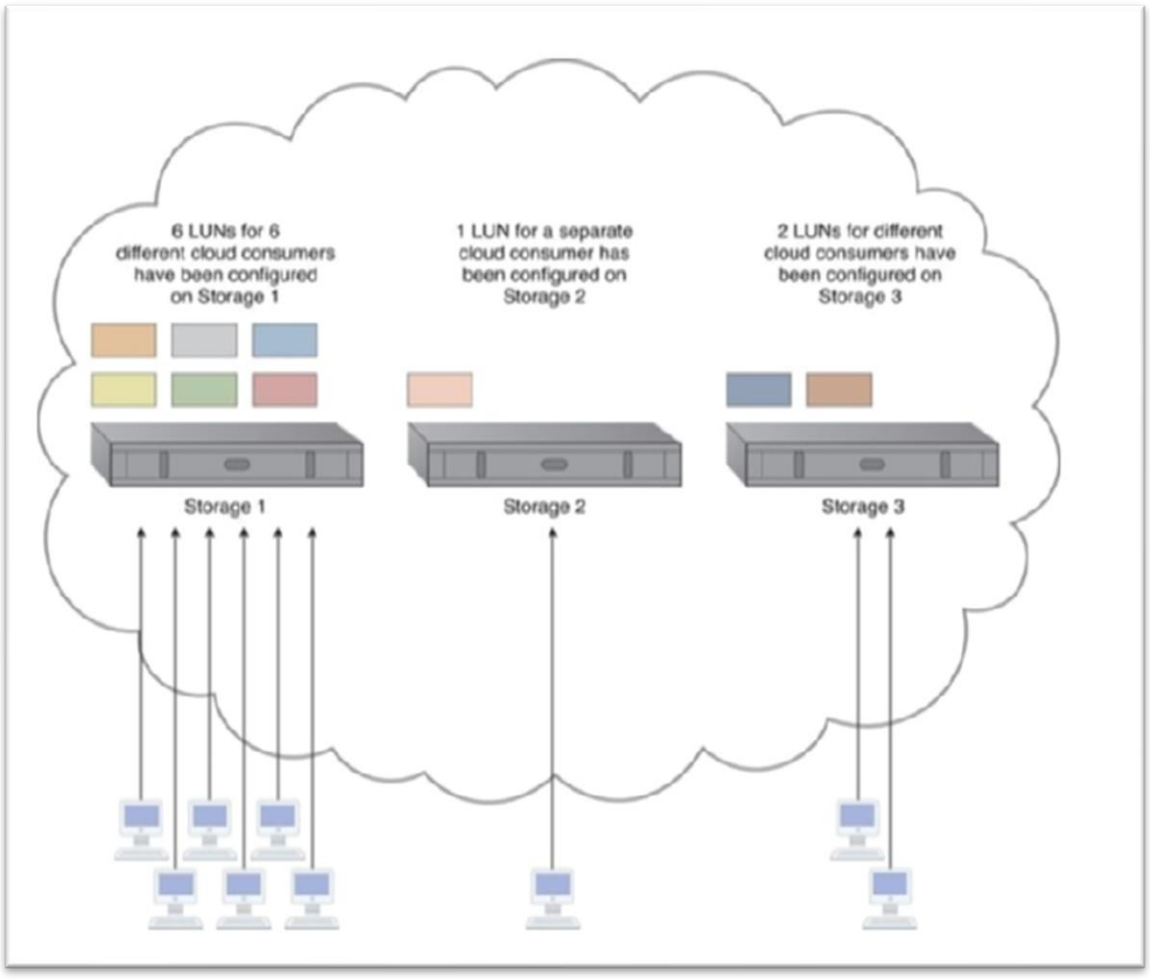
The architectural model can further incorporate Audit Monitor, Failover System, and SLA Management System and SLA Monitor mechanisms. The Bare-Metal Provisioning Architecture establishes a system that utilizes specialized service agents to remotely discover and provision entire operating systems on bare-metal servers that do not have pre-installed operating systems or any other software. This architecture can utilize remote management software that is integrated with the server's ROM and is made available upon server start-up, with a web-based or proprietary user interface to connect to the physical server's native remote management interface. There are concerns regarding its usage, such as being vulnerable to inadvertent human and configuration errors and requiring significant runtime IT resource processing.

The traditional process of provisioning IT resources involves various tasks that are manually performed by administrators and experts. This process can be inadequate in cloud environments, where higher volumes of customers require larger amounts of IT resources. Automating the provisioning process can significantly reduce the amount of time it takes to provision resources and minimize the risk of human error. The Rapid Provisioning Architecture is a system that automates the provisioning of a wide range of IT resources, using an automated provisioning program, rapid provisioning engine, and scripts and templates for on-demand provisioning.



A cloud resource administrator request

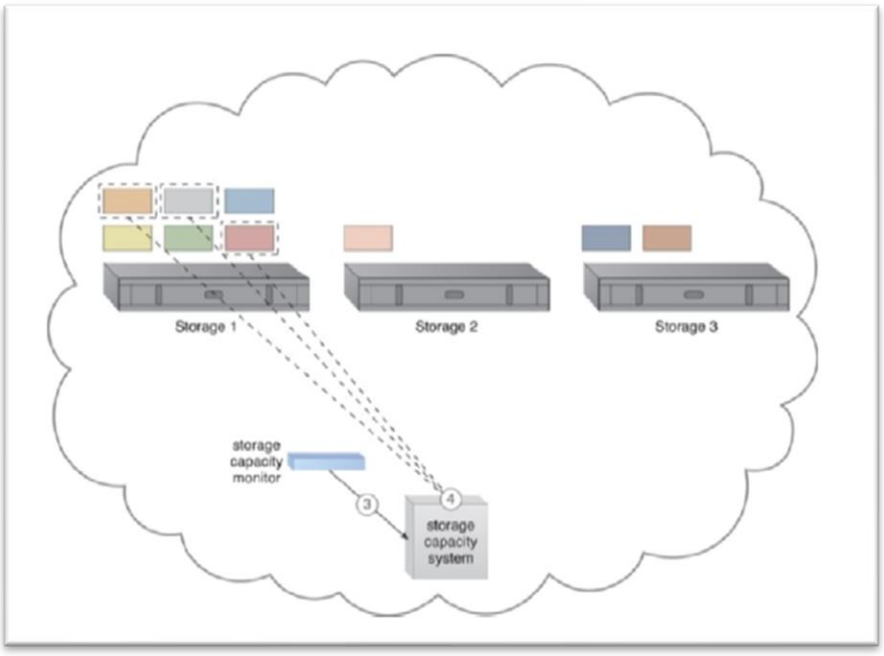
The architecture involves several components, such as server templates, server images, application packages, custom scripts, sequence manager, sequence logger, operating system baseline, application configuration baseline, and deployment data store.



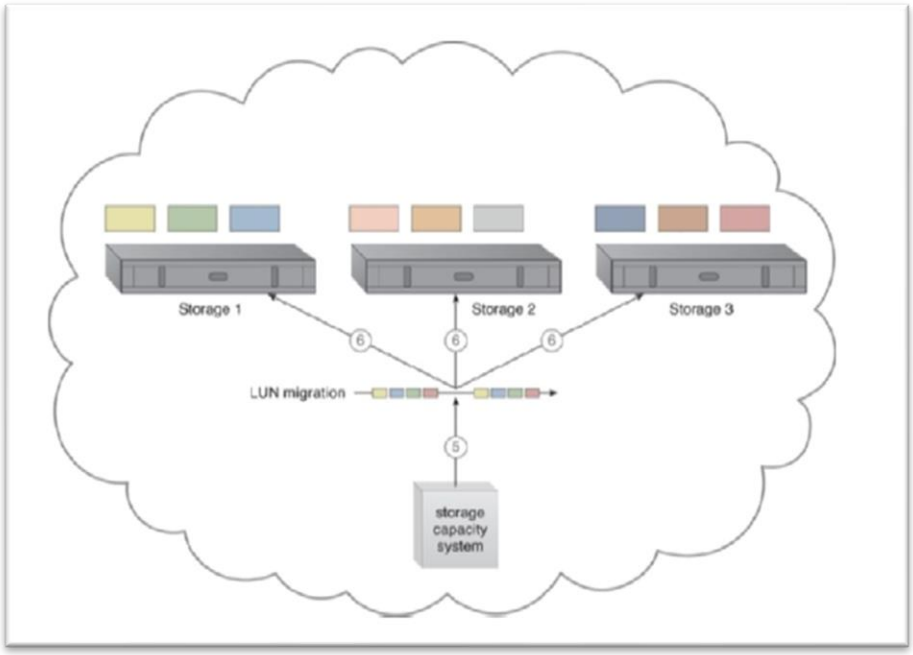
A new cloud service through the self-service portal

These components work together to coordinate and automate different aspects of IT resource provisioning.

To gain insight into the inner workings of the rapid provisioning engine, we describe a step-by-step process that involves several system components. The process begins with a cloud consumer requesting a new server through the self-service portal.



The storage capacity system that Storage 1 is over-utilized

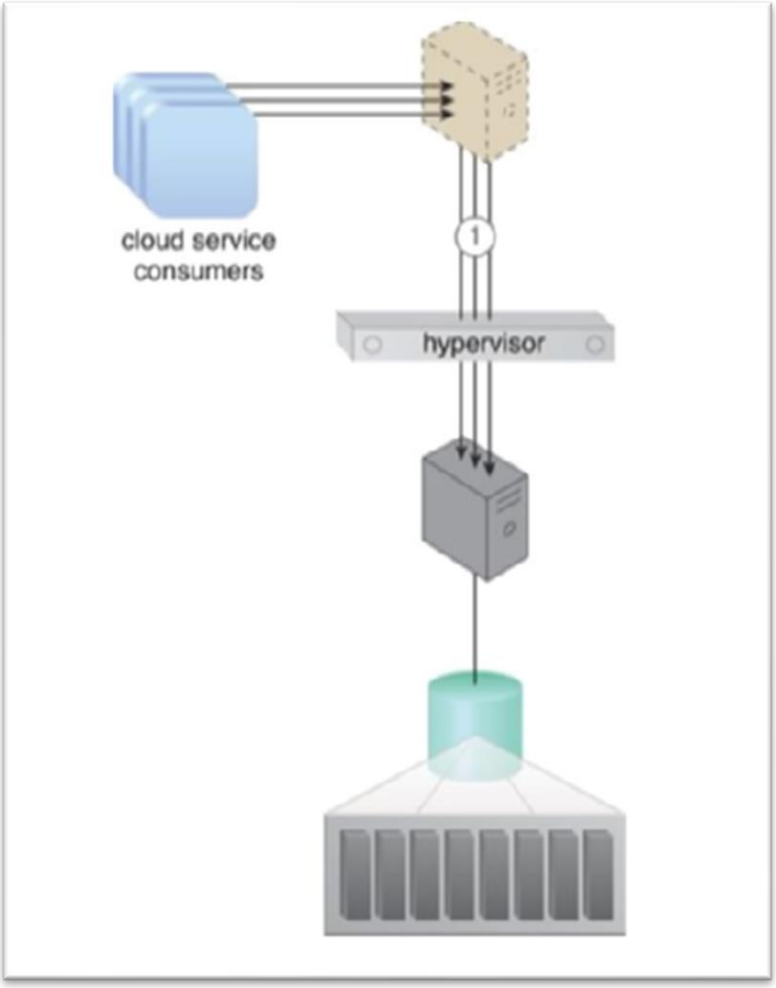


The storage capacity system calls

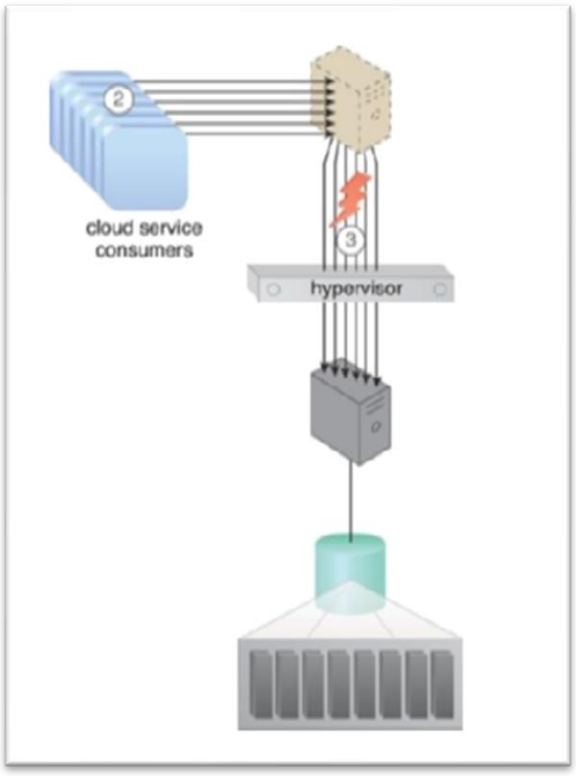
The sequence manager forwards the request to the deployment engine, which prepares the operating system. The deployment engine then uses virtual server templates for provisioning or sends the request to provision a physical server. Once the operating system is ready, the deployment engine informs the sequence manager, which updates and sends logs to the sequence logger for storage. The deployment engine applies the operating system baseline, installs the applications, and informs the sequence manager at each step. Finally, the sequence manager updates and sends logs of completed steps to the sequence logger for storage.

Specialized Cloud Architectures

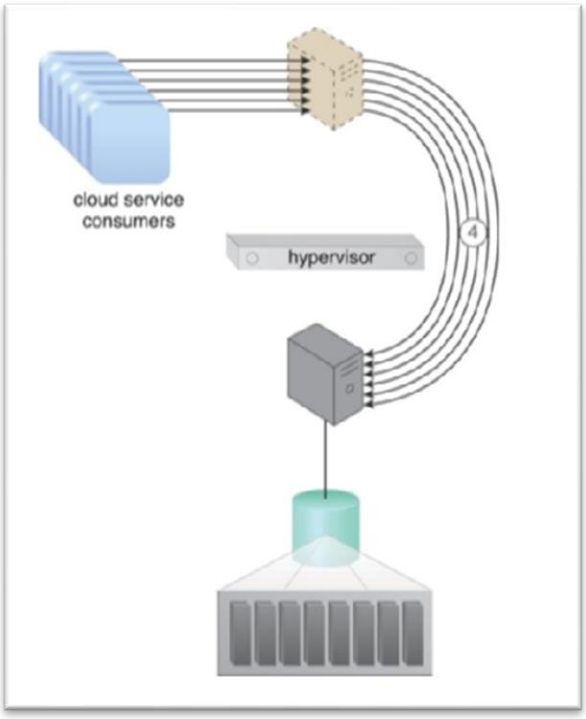
This chapter covers various architectural models that offer creative combinations of mechanisms and specialized components. In this section, we will discuss three architectures, namely Direct I/O Access Architecture, Direct LUN Access Architecture, and Dynamic Data Normalization Architecture.



Access a virtual server

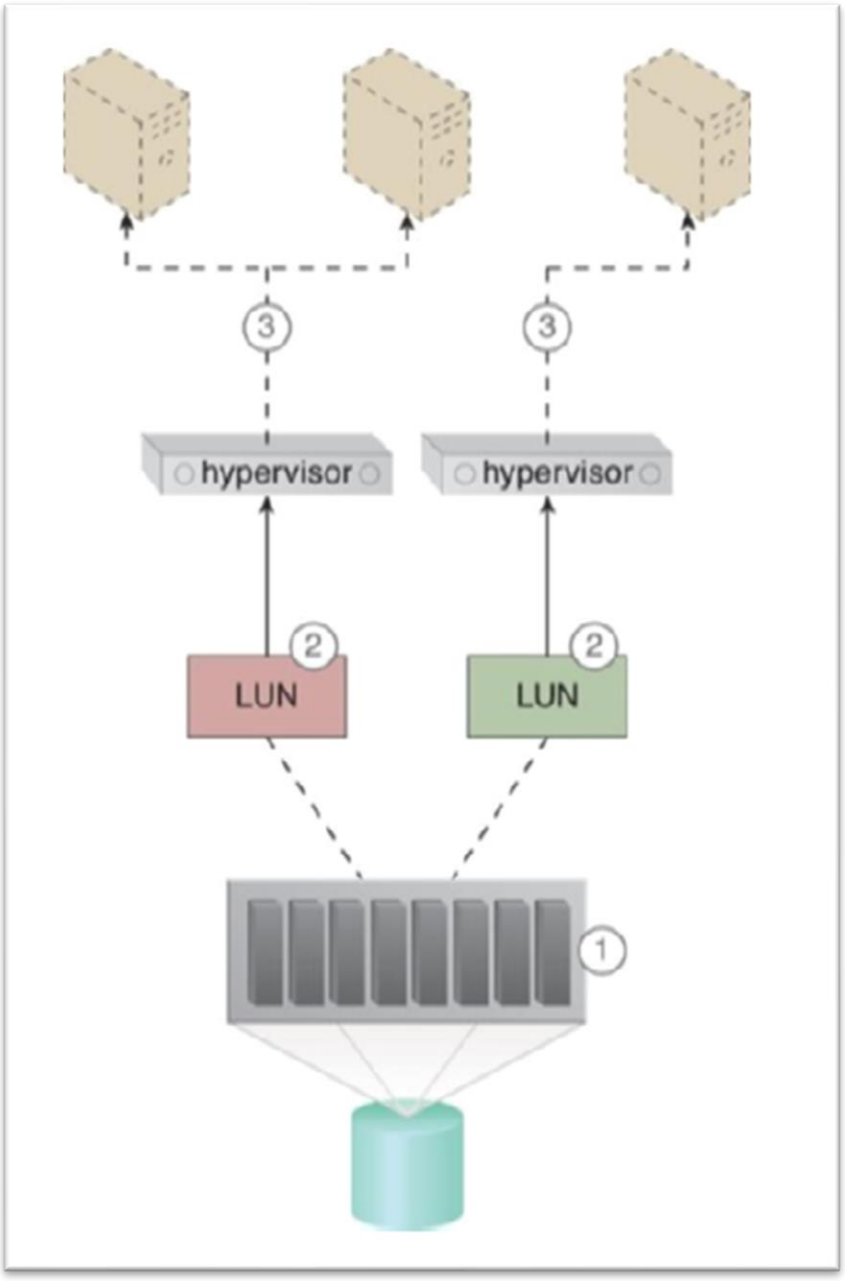


Cloud service consumer requests



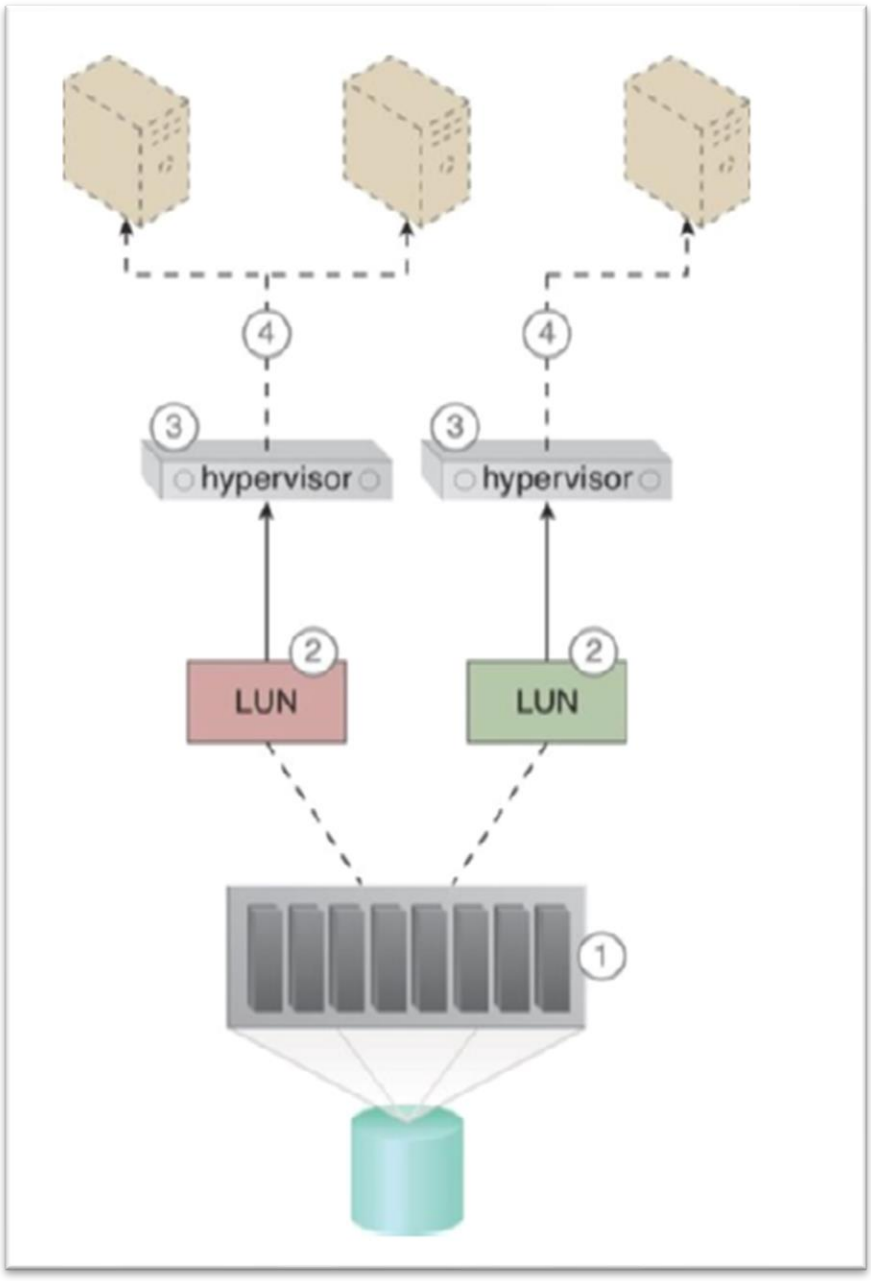
A direct physical link to the physical server

The Direct I/O Access Architecture provides virtual servers with access to physical I/O cards installed on physical servers without the need for hypervisor interaction or emulation. The virtual server can recognize the I/O card as a hardware device with the appropriate drivers installed. Other mechanisms that can be incorporated into this architecture include Cloud Usage Monitor, Logical Network Perimeter, Pay-Per-Use Monitor, and Resource Replication.



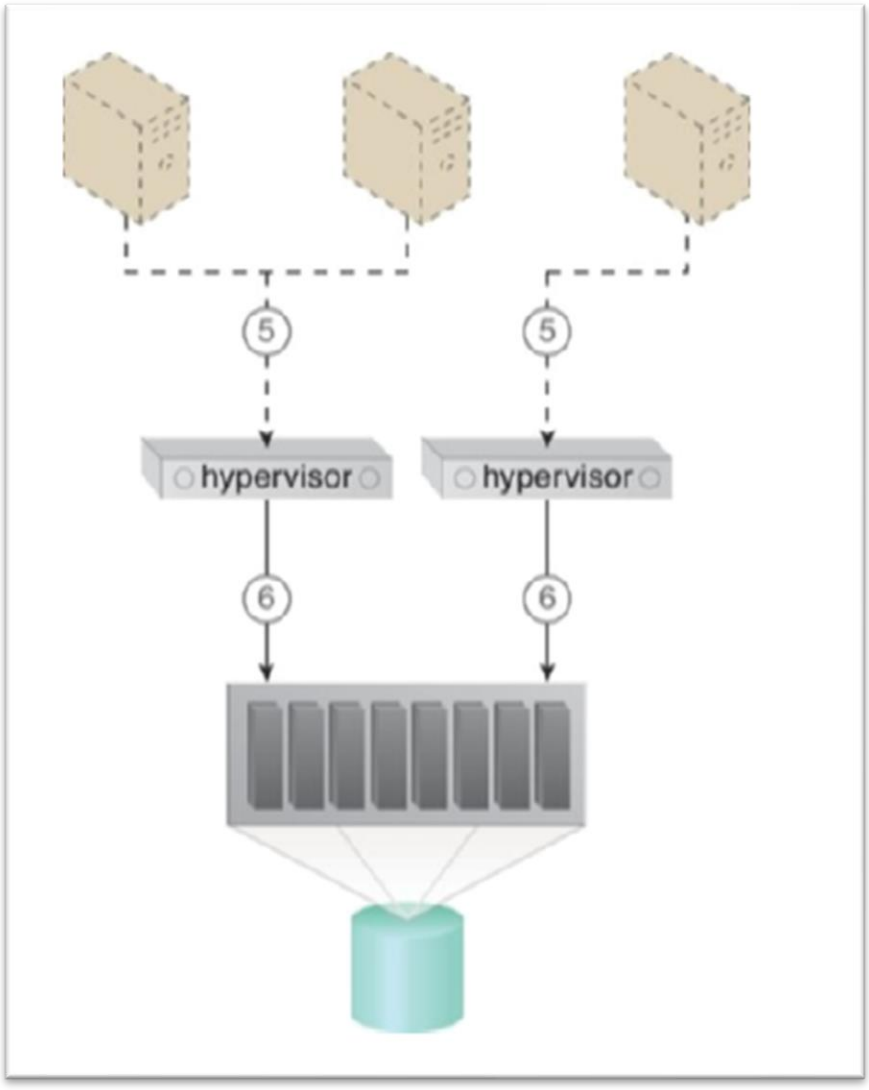
The cloud storage device is installed and configured

The Direct LUN Access Architecture enables virtual servers to directly access RAW block-based storage via a physical HBA card. This architecture is effective for clustered databases that require LUN as a shared volume between two virtual servers.



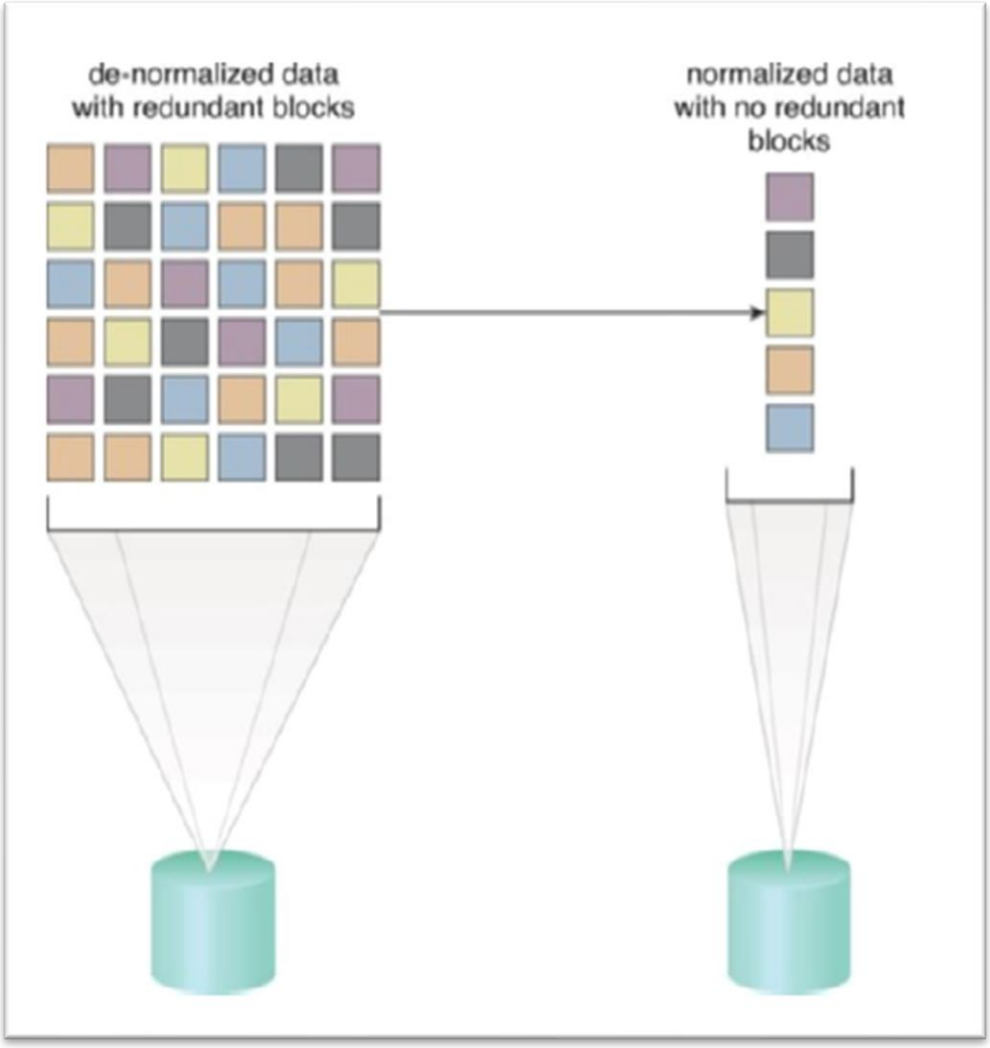
The cloud storage device is installed and configured

The cloud storage device needs to be configured using raw device mapping to make LUNs visible to the virtual servers as a block-based RAW SAN LUN.



The virtual servers' storage commands are received by the hypervisors

Other mechanisms that can be incorporated into this architecture include Cloud Usage Monitor, Pay-Per-Use Monitor, and Resource Replication.

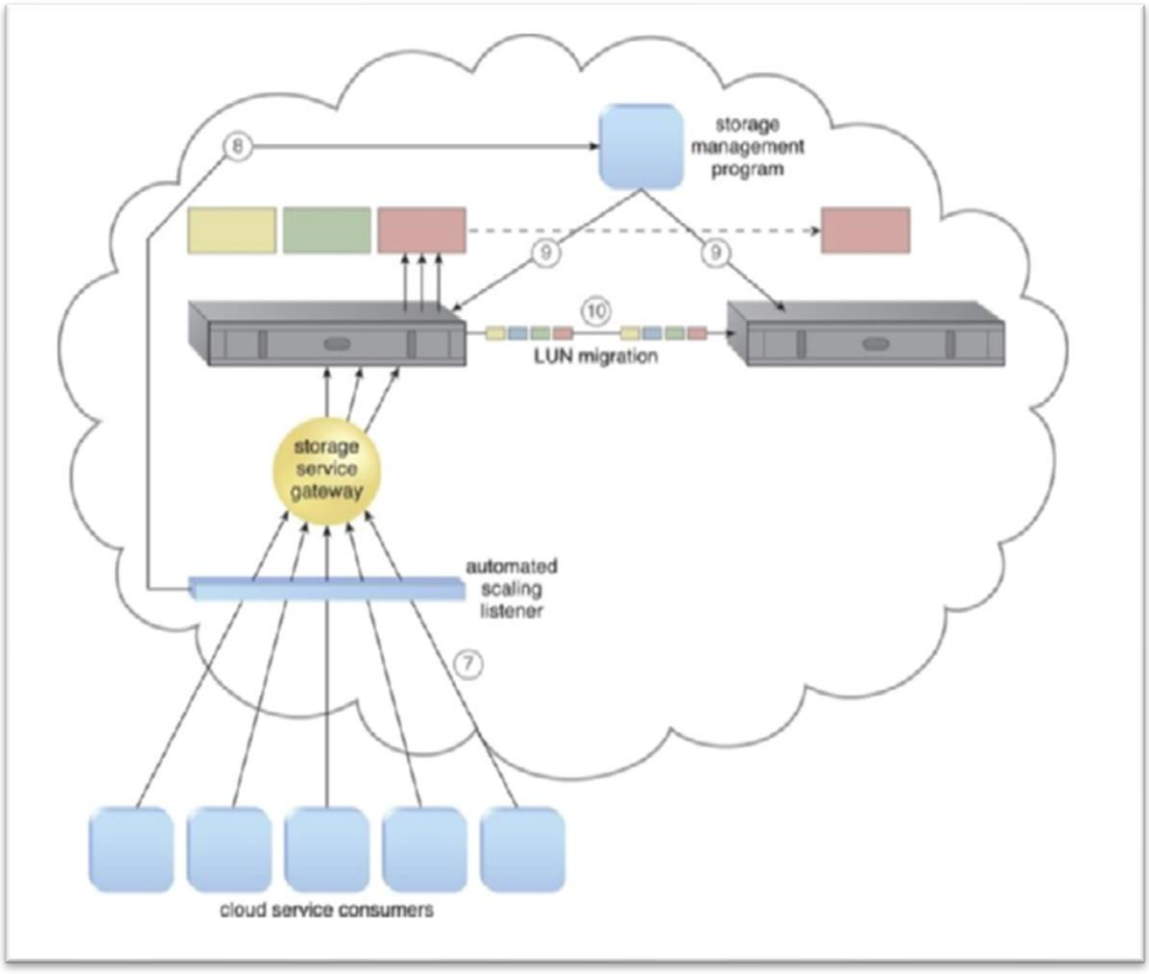


Data sets containing redundant data are unnecessarily bloating storage

The Dynamic Data Normalization Architecture aims to address issues caused by redundant data in cloud-based environments. These issues include the increased time required to store and catalog files, increased required storage and backup space, and increased costs due to increased data volume.

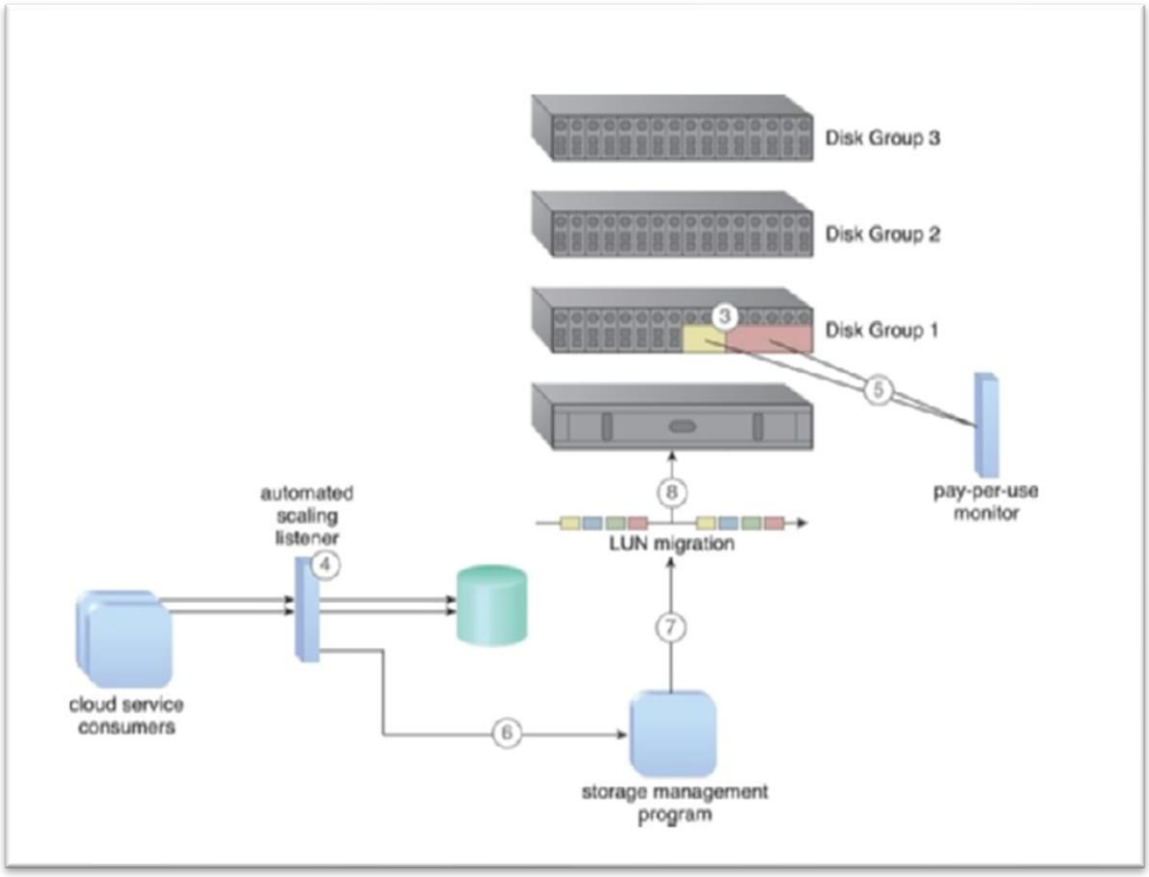
Cloud storage devices may not be able to meet the performance requirements of cloud consumers, and the conventional methods of vertical scaling can be time-consuming and inefficient, as well as wasteful. When requests for access to a LUN increase, the LUN is transferred to a high-performance cloud storage device.

The cross-storage device vertical tiering architecture establishes a system that survives bandwidth and data processing power constraints by vertically scaling between storage devices that have different capacities.



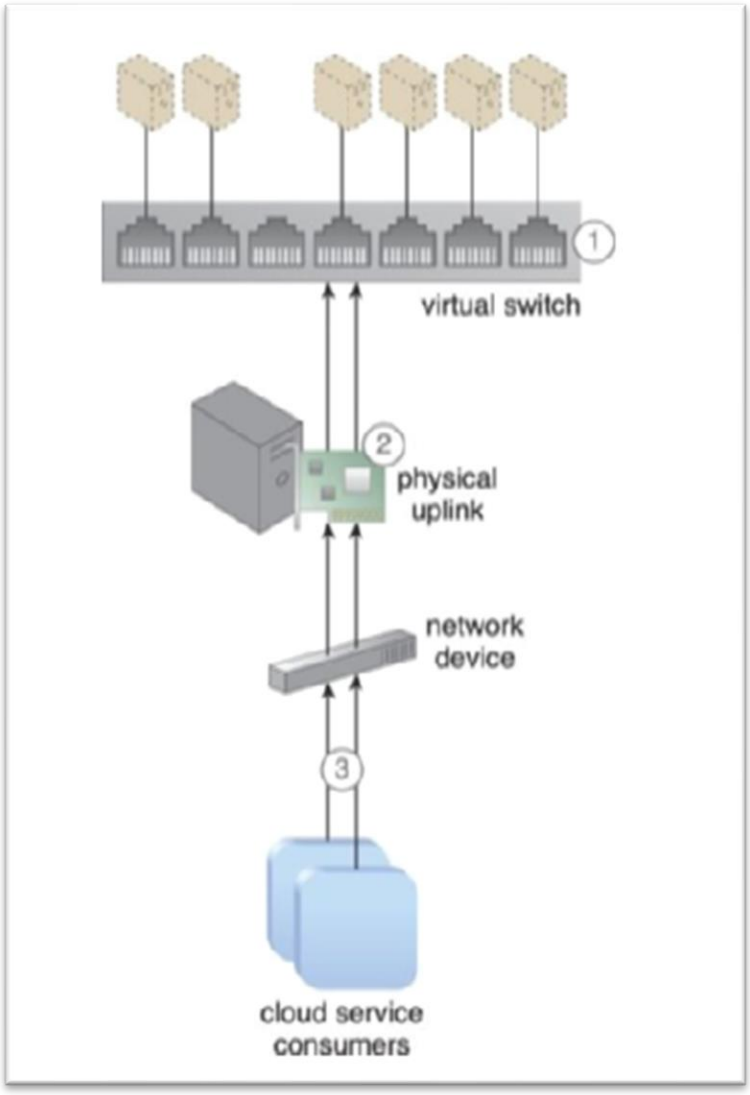
The number of cloud service consumer requests reaches the predefined threshold

The automated scaling listener monitors requests sent to specific LUNs and signals the storage management program to move the LUN to a higher-capacity device when a predefined threshold is reached. The service interruption is prevented because there is no disconnection during the transfer. In addition to the automated scaling listener and cloud storage device, the technology architecture includes mechanisms such as the Audit Monitor, Cloud Usage Monitor, and Pay-Per-Use Monitor.

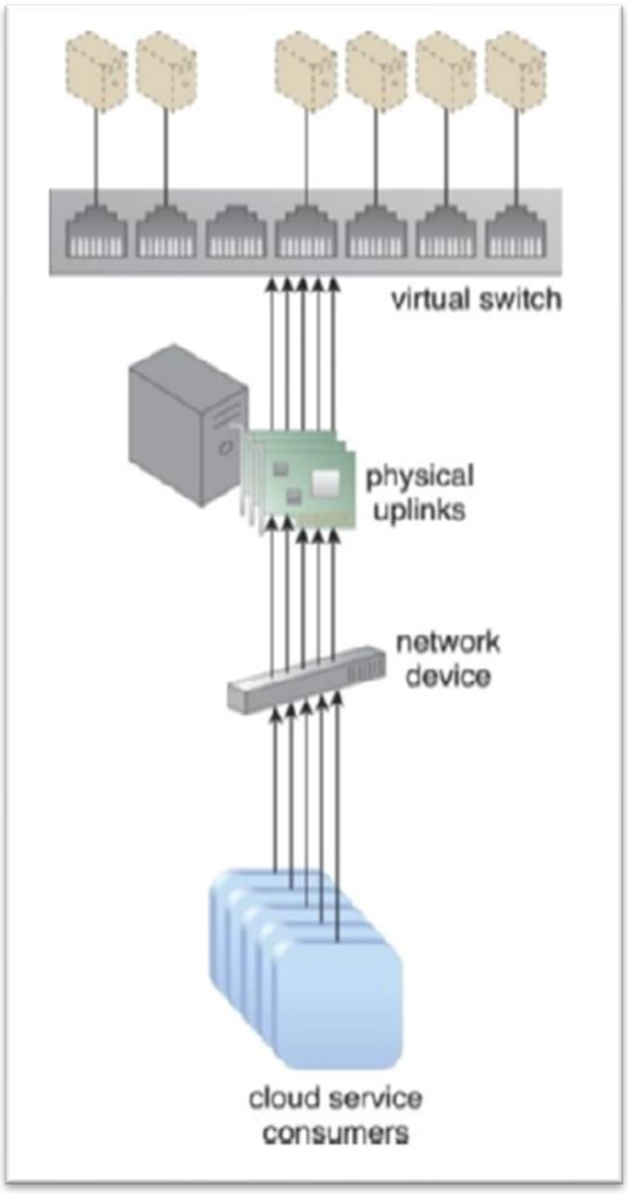


Two LUNs have been created on Disk Group 1

The intra-storage device vertical data tiering architecture establishes a system to support vertical scaling within a single cloud storage device. This system optimizes the availability of different disk types with different capacities. This cloud storage architecture requires the use of a complex storage device that supports different types of hard disks, especially high-performance disks like SATAs, SASs, and SSDs.

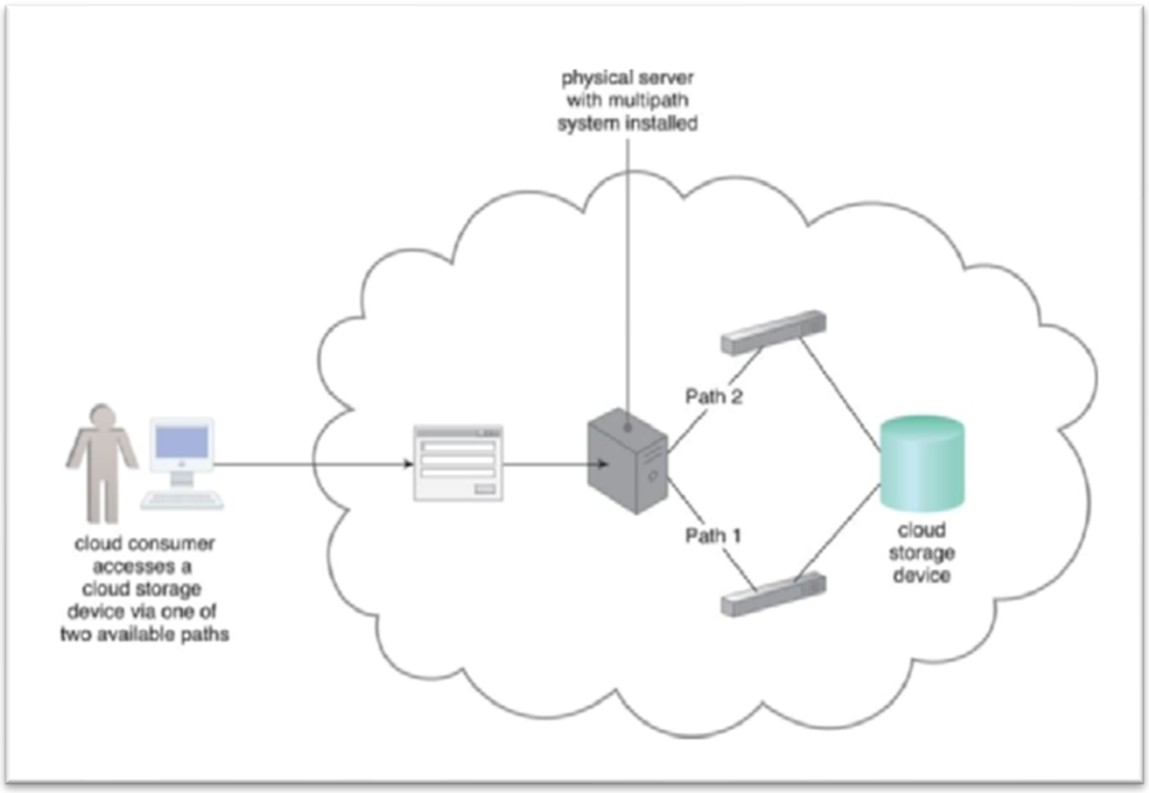


A virtual switch is interconnecting virtual servers



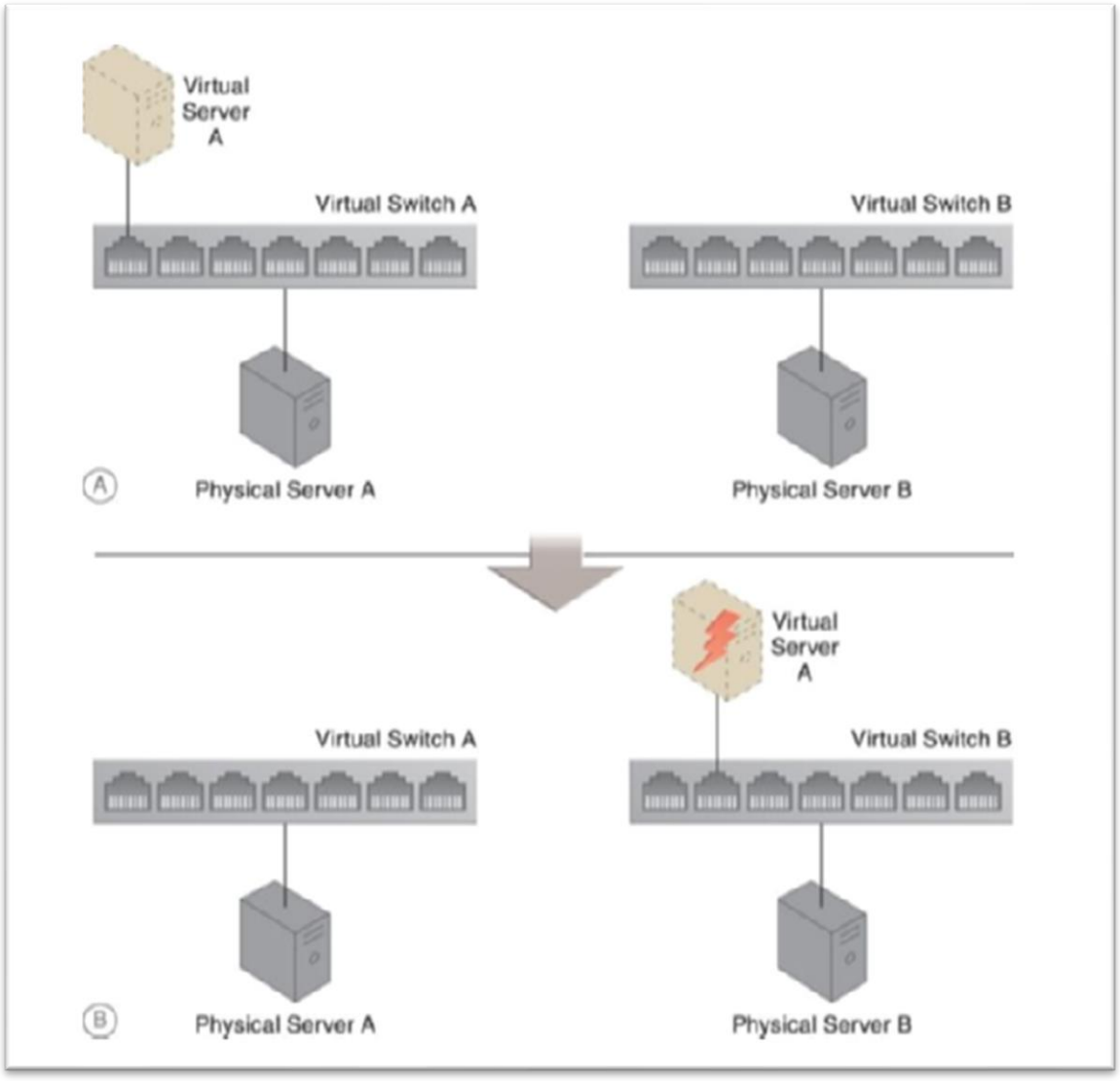
Additional physical uplinks are added to distribute and balance network traffic

The disk types are organized into graded tiers so that LUN migration can vertically scale the device based on the allocation of disk types. The cloud storage architecture can be useful when cloud consumers have distinct data storage requirements that restrict the data's physical location to a single cloud storage device.



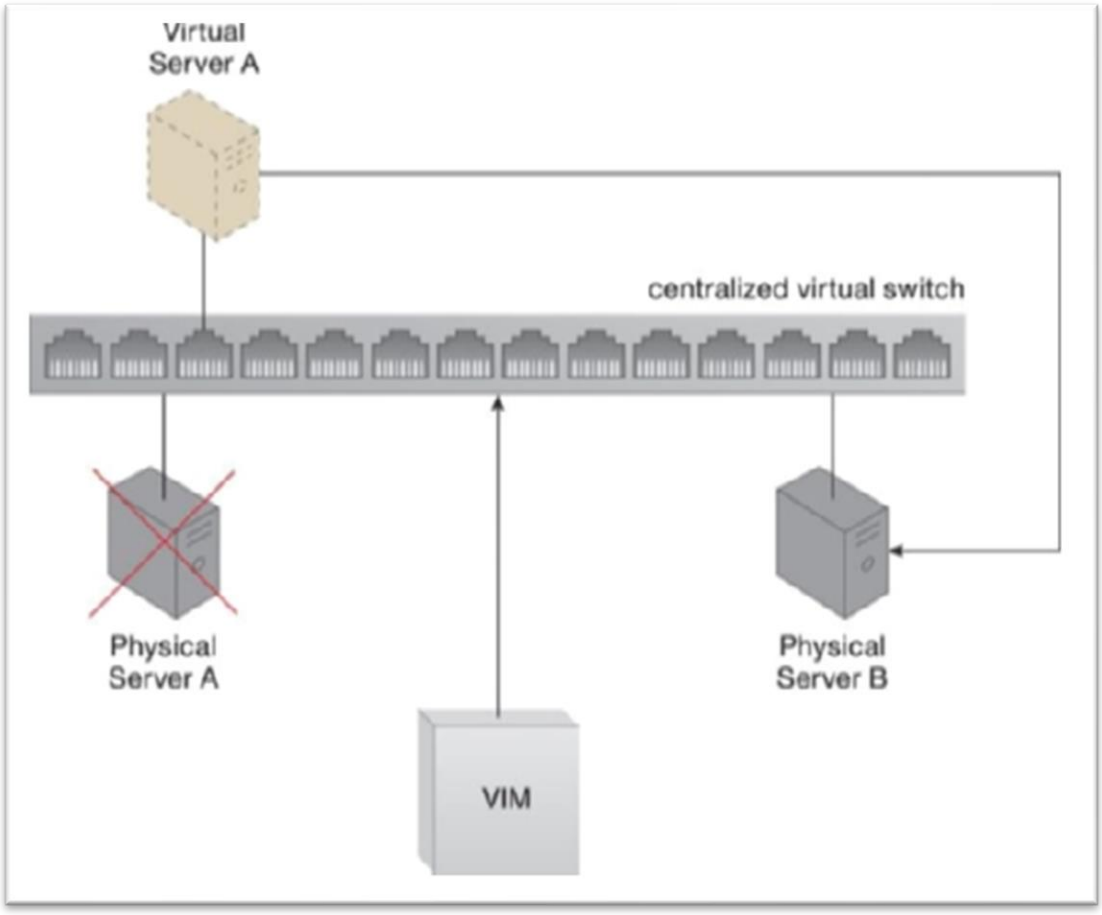
A multi-pathing system is providing alternative paths to a cloud storage device

When virtual servers are created, their network configurations and port assignments are generated on the physical server and the hypervisor that hosts them. However, when a virtual server is moved to another host, it loses network connectivity because the destination hosting environments do not have the required port assignments and network configuration information.



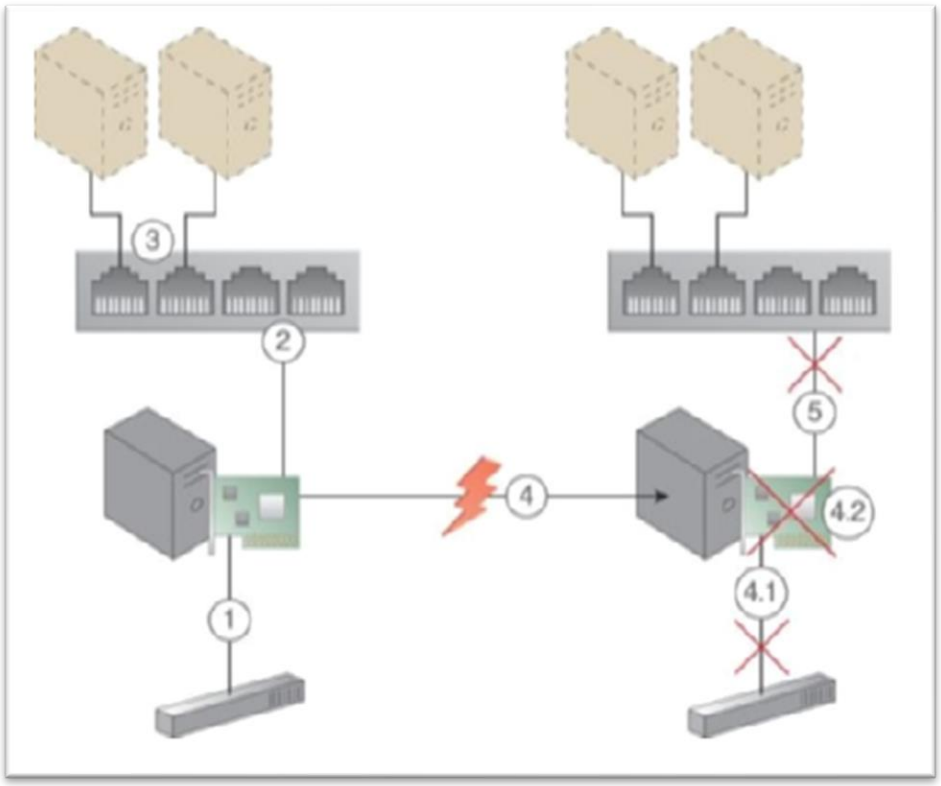
Virtual Server A connected to the network

The persistent virtual network configuration architecture solves this problem by storing the network configuration information in a centralized location and replicating it to physical server hosts. This allows the destination host to access the configuration information when a virtual server is moved from one host to another.



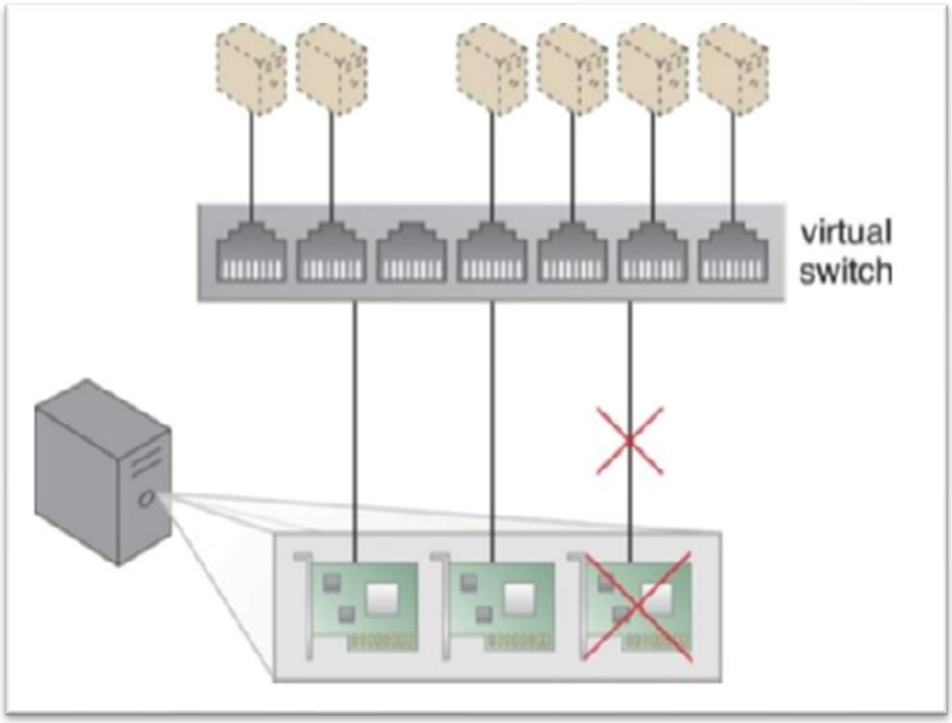
A virtual switch's configuration settings

This architecture includes a centralized virtual switch, VIM, and configuration replication technology. The centralized virtual switch is shared by physical servers and configured via the VIM, which initiates the replication of the configuration settings to the physical servers.



A physical network adapter installed on the host physical server

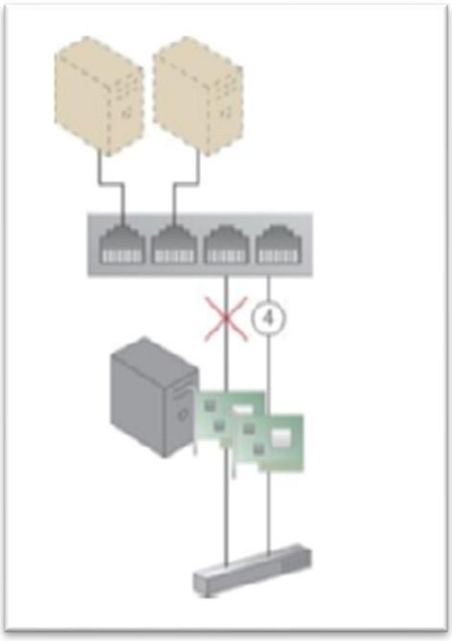
The architecture also supports other mechanisms, such as the hypervisor, which hosts virtual servers that require configuration settings to be replicated across physical hosts; the logical network perimeter, which ensures that access to the virtual server and its IT resources is isolated to the rightful cloud consumer before and after a virtual server is migrated; and resource replication, which is used to replicate the virtual switch configurations and network capacity allocations across the hypervisors via the centralized virtual switch.



Redundant uplinks are installed

Virtual servers are connected to an external network via a virtual switch uplink port. However, if the uplink fails, the virtual server becomes isolated and disconnected from the external network. The architecture for redundant physical connection for virtual servers addresses this issue by establishing one or more redundant uplink connections and positioning them in standby mode.

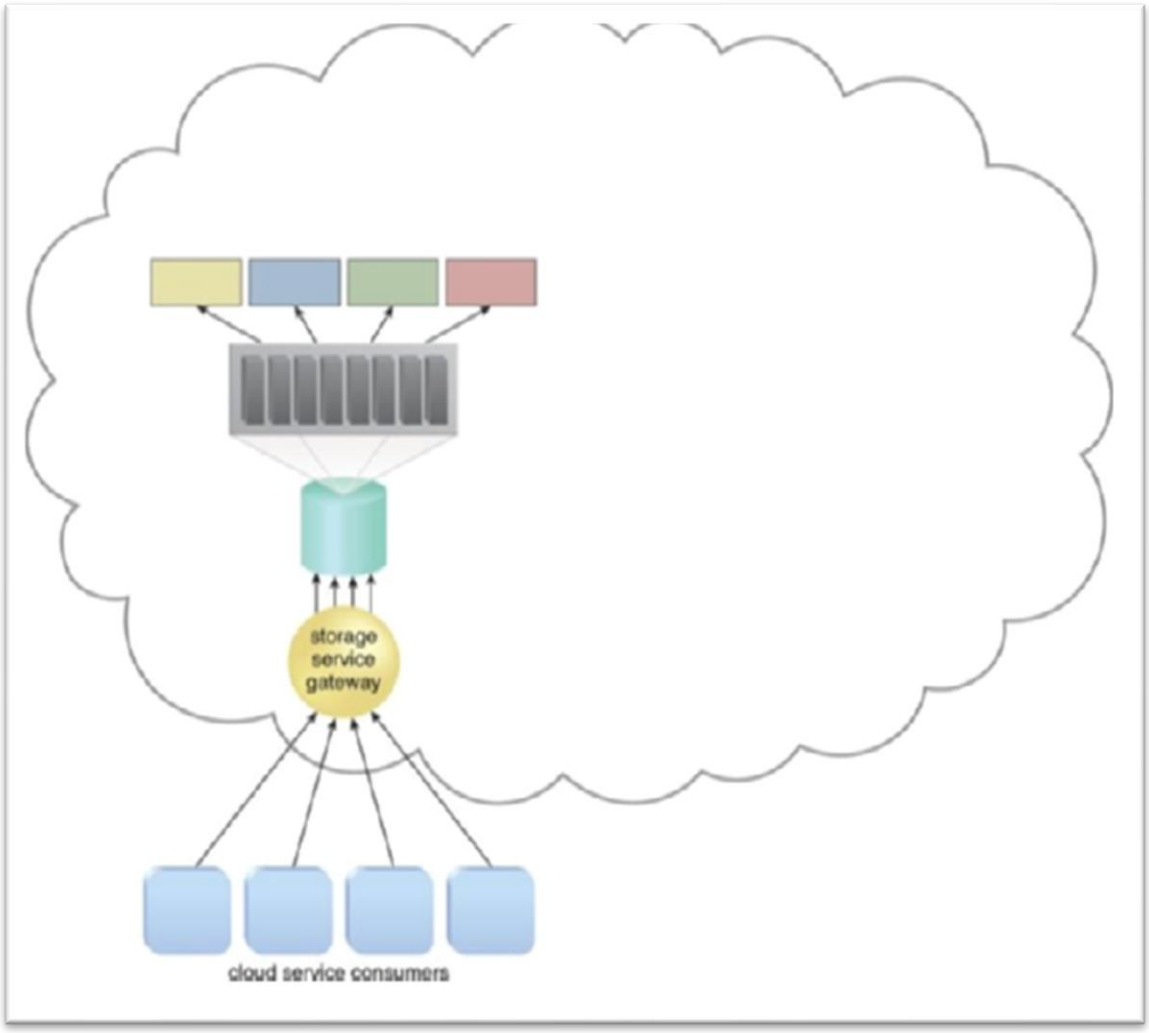
This architecture ensures that a redundant uplink connection is available to connect the active uplink whenever the primary uplink connection becomes unavailable. In the event of a failure, a standby uplink automatically becomes the active uplink, and the virtual servers use the newly active uplink to send packets externally, and transparently to both virtual servers and their users.



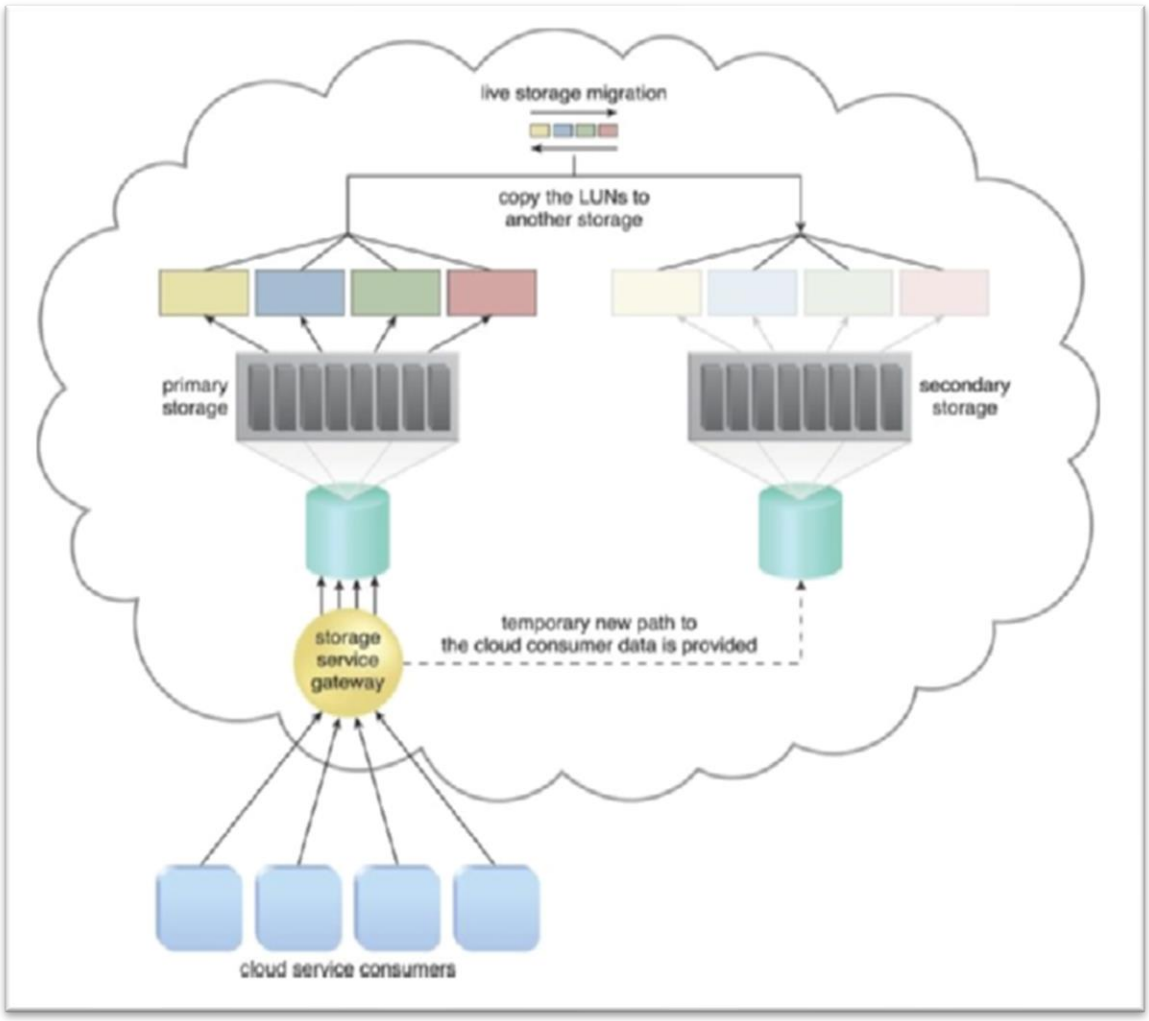
One physical network adapter is designated as the primary adapter

The architecture includes a failover system, which performs the transition of unavailable uplinks to standby uplinks, as well as a hypervisor, which hosts virtual servers and some virtual switches and provides virtual networks and virtual switches with access to the virtual servers. Additionally, the architecture supports the logical network perimeter, which ensures that the virtual switches that are allocated or defined for each cloud consumer remain isolated, and resource replication.

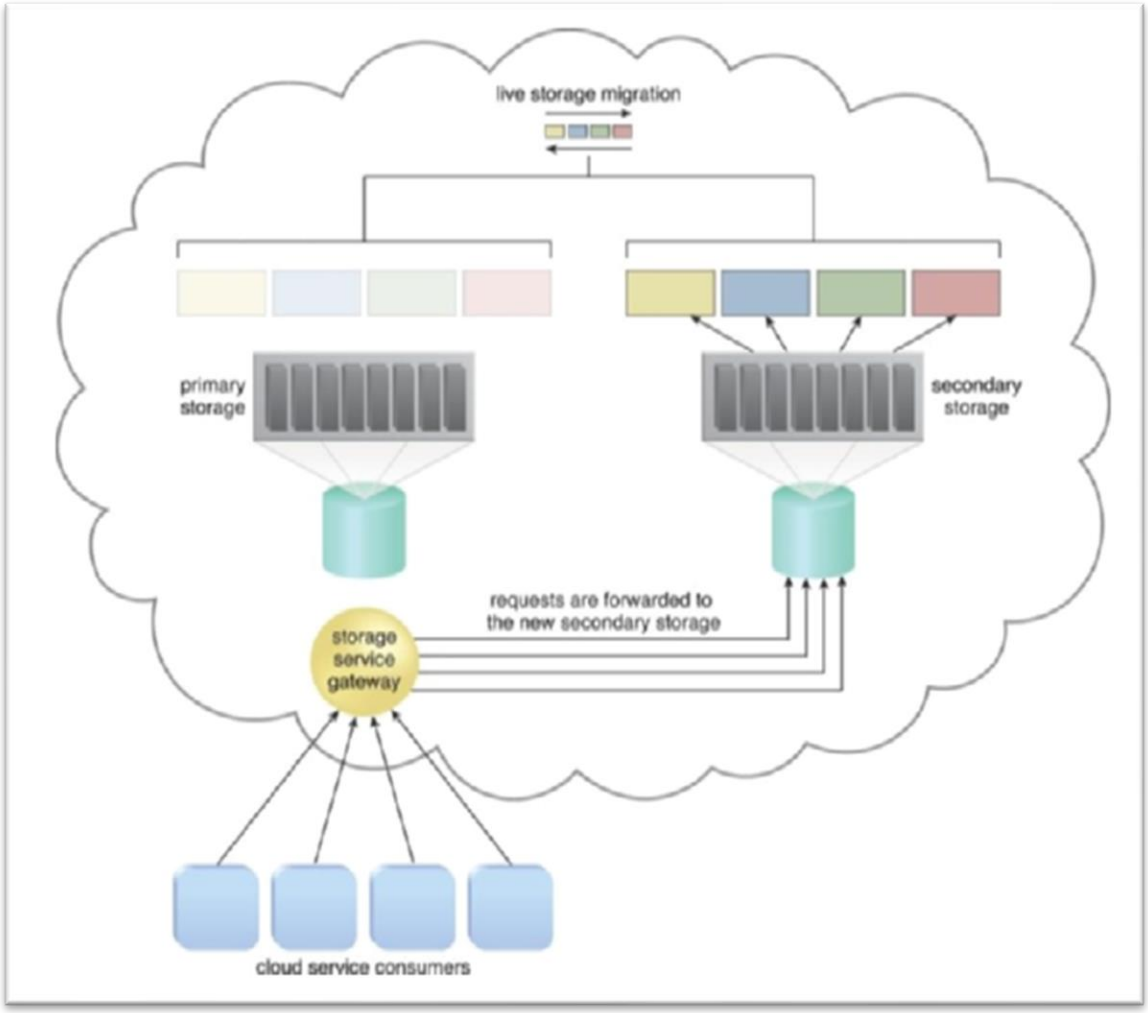
Sometimes cloud storage devices need to be temporarily shut down for maintenance and administrative tasks, causing cloud service consumers and IT resources to lose access to these devices and their stored data. The architecture for storage maintenance window addresses this issue by enabling a pre-scheduled maintenance task to migrate the data to another storage device before the maintenance window begins.



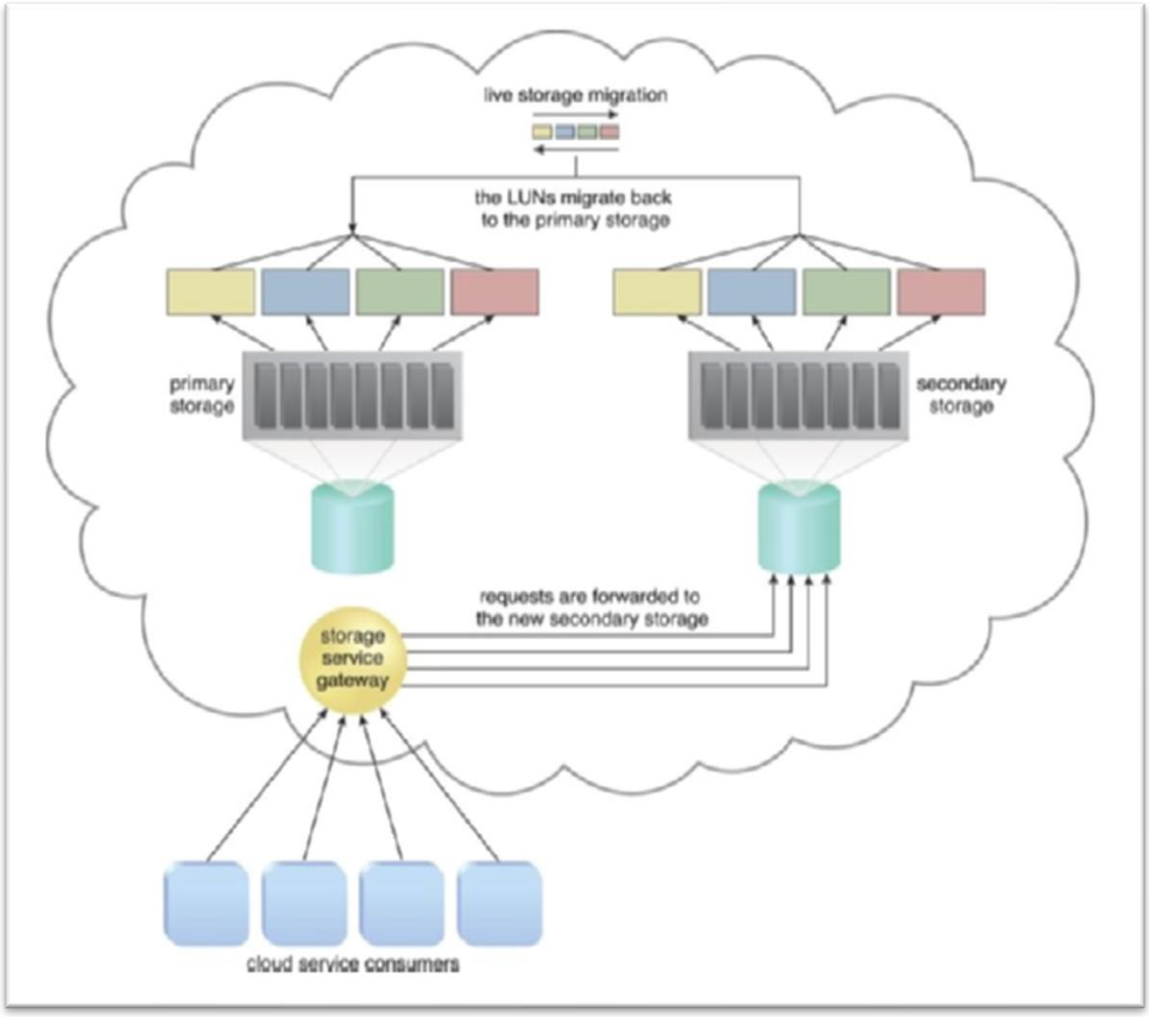
The cloud storage device is scheduled to undergo a maintenance outage



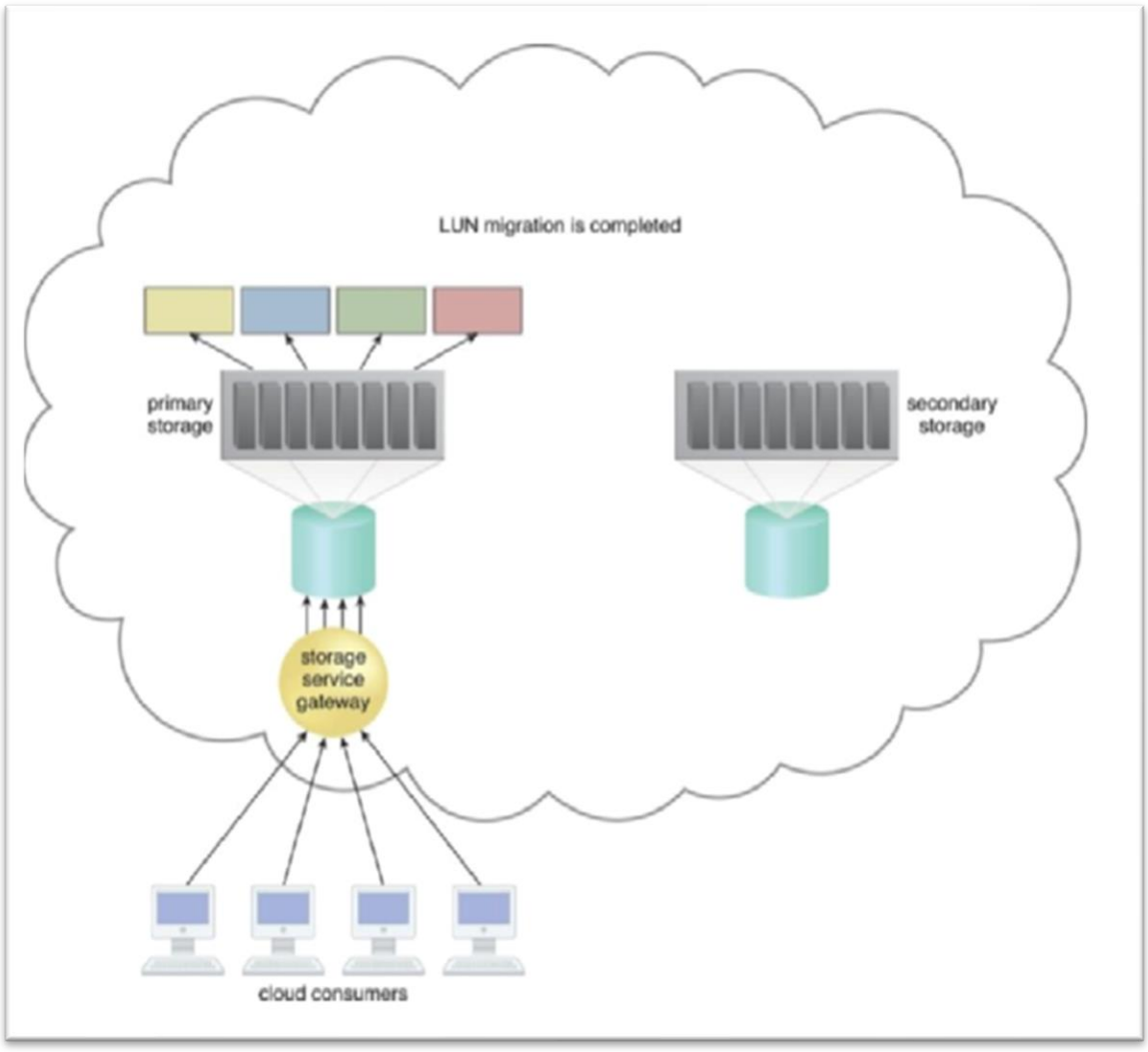
Live storage migration moves the LUNs



The primary storage off for maintenance



The primary storage is brought back online



The data access requests are forwarded back to the primary cloud storage device

The architecture includes a failover system that ensures that the virtual machines can access the data stored on the new device while the maintenance task is being performed on the original device. Additionally, the architecture supports a logical network perimeter that ensures that the virtual machines that are allocated or defined for each cloud consumer remain isolated, and resource replication.

Solutions Architect's Handbook

Attributes of the Solution Architecture

When designing a solution architecture, various attributes need to be considered to ensure the solution's success across multiple projects in an organization. This demands a careful evaluation of the architecture's properties and striking a balance between them. In this chapter, we will provide an in-depth understanding of each attribute and how they coexist in solution design. While there may be more attributes for complex solutions, we will cover the common characteristics that apply to most aspects of solution design. These attributes can be viewed as non-functional requirements (NFRs) that fulfill an essential aspect of design. As a solution architect, it is your responsibility to ensure all attributes satisfy the desired requirements and fulfill customer expectations.

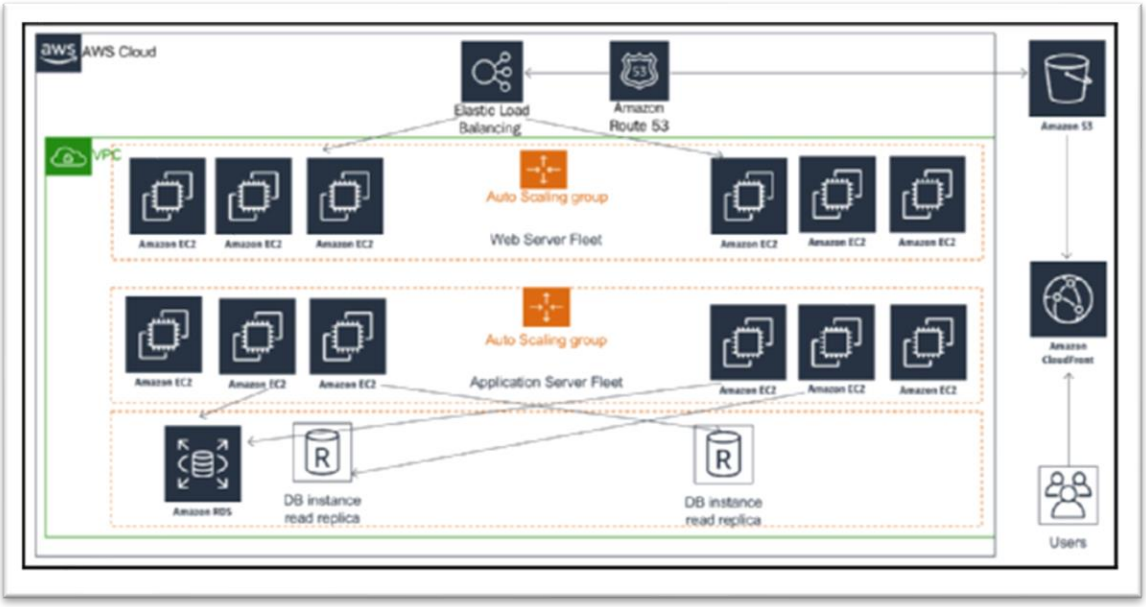
This chapter covers the following topics in detail:

- Scalability and elasticity
- High availability and resiliency
- Fault tolerance and redundancy
- Disaster recovery and business continuity
- Extensibility and reusability
- Usability and accessibility
- Portability and interoperability
- Operational excellence and maintainability
- Security and compliance
- Cost optimization and budgets

By understanding these attributes and their relationships, solution architects can make informed decisions when designing solutions that meet the organization's needs.

Scalability and elasticity

When designing a solution, scalability is always a crucial factor to consider. Enterprises usually plan for scalability, which refers to the ability of a system to handle growing workloads, including multiple layers such as the application server, web app, and database.



Scaling three-tier architecture

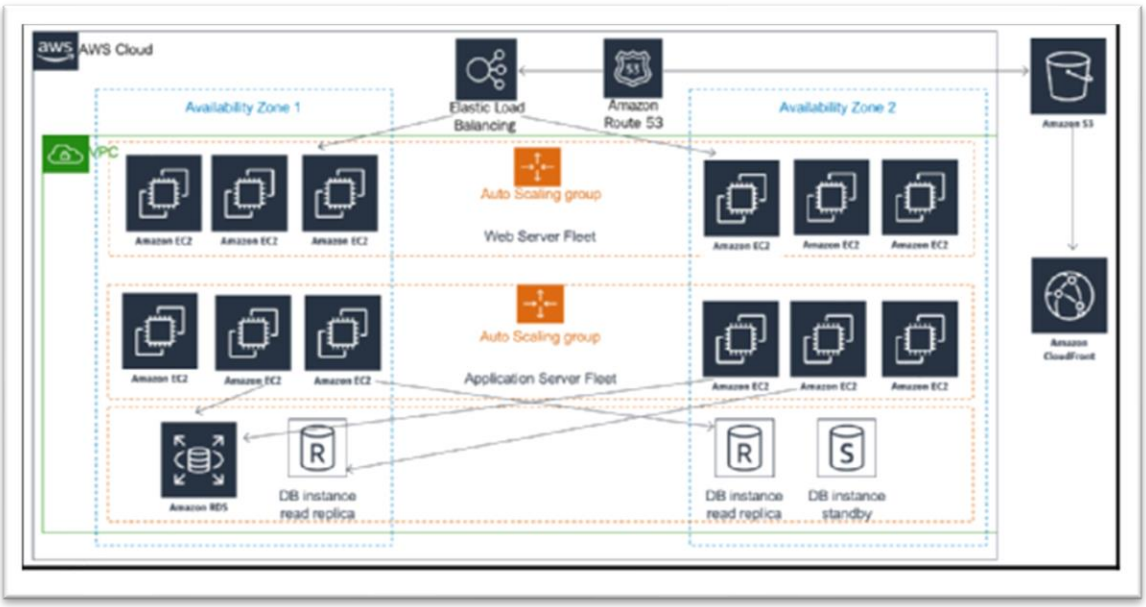
Elasticity is a term replacing scalability, especially for web-based applications, as it includes the ability to shrink the system to save cost. The two traditional modes of scaling are horizontal and vertical scaling. In horizontal scaling, more instances are added to handle increasing workloads, while vertical scaling adds additional compute storage and memory power to the same instance. However, the vertical scaling model may not be cost-effective as the cost increases exponentially. Most businesses have a peak season where the user is most active, and the application must handle additional loads to meet the demand, creating a capacity dilemma in scaling. Elastic workloads are needed to solve this problem.

A modern three-tier architecture with Amazon Web Services can provide elasticity and scalability by using a fleet of web and application servers that auto scale based on CPU and memory utilization. The Content Distribution Network (CDN) can be used to scale static content in the web tier of the architecture.

High availability and resiliency

In the field of information technology, it is crucial for organizations to avoid downtime in their systems. Application downtime can result in a loss of business revenue and trust from users, which highlights the importance of ensuring high availability in solution architecture design. However, the degree of application uptime required varies based on the specific application. For instance, external-facing applications like e-commerce websites and social media platforms need to maintain 100% uptime to prevent negative impacts on user experience. On the other hand, internal applications like HR systems may be able to tolerate some amount of downtime.

When designing a high availability (HA) architecture, it is important to plan the workloads in a physically isolated location within the data center. This way, if an outage occurs in one location, the application replica can operate from another location. The use of load balancers to distribute workloads between multiple availability zones, along with standby instances for the database, can further ensure that the system remains up and running.



High availability and resilience architecture

In addition to high availability, system resiliency is also an essential factor to consider. An application should be able to recover itself without human intervention, which can be achieved through proactive monitoring of workload and the use of self-healing principles. This involves monitoring instance health, such as CPU and memory utilization, and taking proactive actions like spinning up new instances as needed.

By implementing high availability and resiliency, organizations can achieve elasticity and reduce costs by taking out servers when server utilization is low. However, it is equally important to ensure quick recovery in case of a failure to maintain a seamless user experience.

Fault-tolerance and redundancy

In the preceding section, you were introduced to the close relationship between fault tolerance and high availability. High availability refers to an application's ability to remain accessible to users even during an outage, albeit with potentially reduced performance. For instance, if your application requires four servers to handle user traffic, you can distribute two servers across two physically isolated data centers. In the event of an outage in one data center, user traffic can be rerouted to the other data center. However, this setup only provides 50% fault tolerance since you are left with only two servers, which may result in performance issues for users. On the other hand, fault tolerance aims to handle workload capacity during outages without compromising system performance. Achieving full fault tolerance requires a fully redundant architecture, which can be costly due to the increased redundancy. The level of fault tolerance needed depends on the criticality of your application and whether your user base can tolerate degraded performance during the application recovery period.

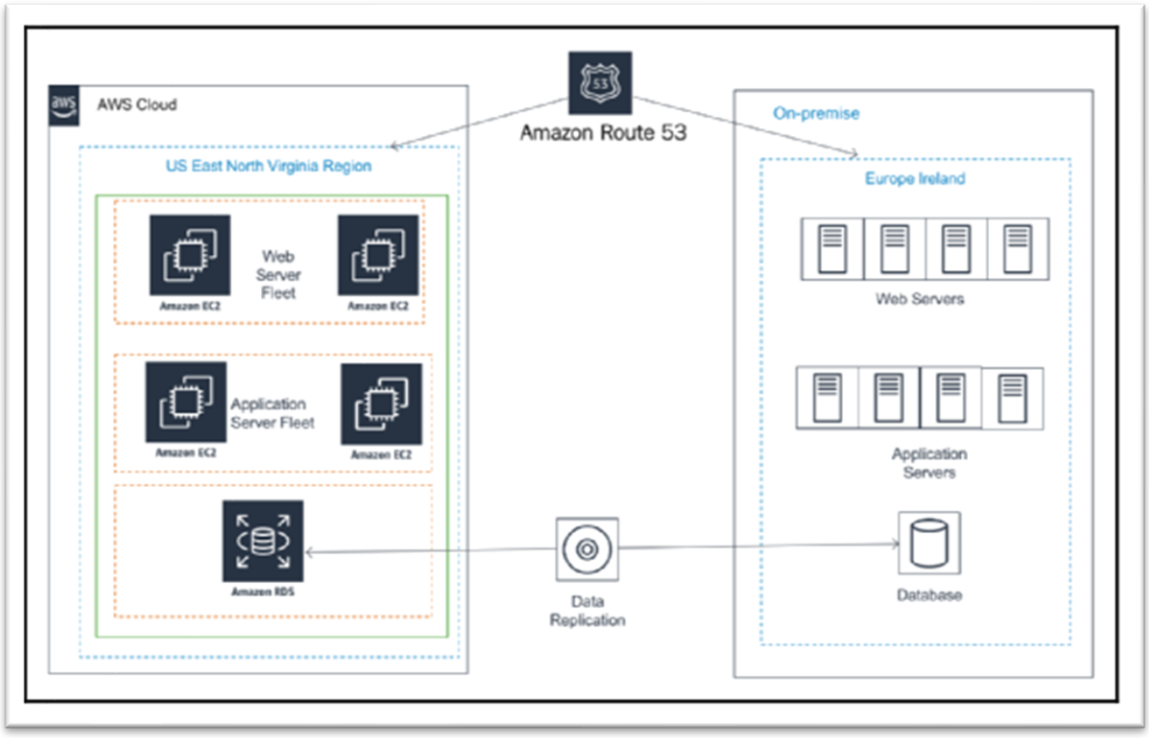
Fault-tolerance architecture

As depicted in the diagram above, the application requires four servers to handle the full workload, distributed across two zones. In both scenarios, 100% high availability is maintained. However, to achieve 100% fault tolerance, full redundancy is necessary, which entails maintaining double the number of servers to ensure that users don't experience any performance issues during an outage in one zone. If the redundancy cost is prohibitive, the solution architect must determine the nature of the application's user base and design for 100% fault tolerance only where necessary.

Disaster recovery and business continuity

In the previous section, you learned about the importance of high availability and fault tolerance for maintaining application uptime. However, in the event of natural disasters or widespread power outages, entire regions may go down, potentially disrupting global business operations. To avoid such scenarios, it is crucial to have a disaster recovery plan in place, which involves preparing IT resources in a different region, possibly in a different country or continent, to ensure business continuity.

When designing a disaster recovery plan, a solution architect must consider the organization's Recovery Time Objective (RTO) and Recovery Point Objective (RPO). RTO refers to the maximum amount of downtime a business can tolerate without significant impact, while RPO indicates the level of data loss the business can withstand. It is essential to understand the criticality of the business and its requirements before deciding on the RTO and RPO levels since they directly impact costs.



Hybrid multi-site disaster recovery architecture

The following architecture diagram depicts a multi-site disaster recovery architecture, with the primary data center located in Ireland, Europe, and the disaster recovery site in Virginia, USA, hosted on AWS public cloud. This design ensures business continuity even if the entire European region or public cloud experiences an outage. The multi-site disaster recovery plan achieves minimal RTO and RPO, resulting in little to no downtime or data loss.

There are several disaster recovery plans available, with varying costs and RTO and RPO levels. These plans include:

- Backup and Store: The least expensive option with maximum RTO and RPO, where server machine images and database snapshots are stored in the disaster recovery site. In case of a disaster, the team will restore the site from a backup.
- Pilot Lite: More expensive than Backup and Store, with less RTO and RPO, this plan involves storing server machine images as a backup, and maintaining a small database server in the disaster recovery site with continuous data sync from the main site. Other critical services, such as Active Directory, may be running in small instances.
- Warm Standby: More expensive than Pilot Lite, with less RTO and RPO, this plan involves running all application servers and the database server on lower capacity instances in the disaster recovery site,

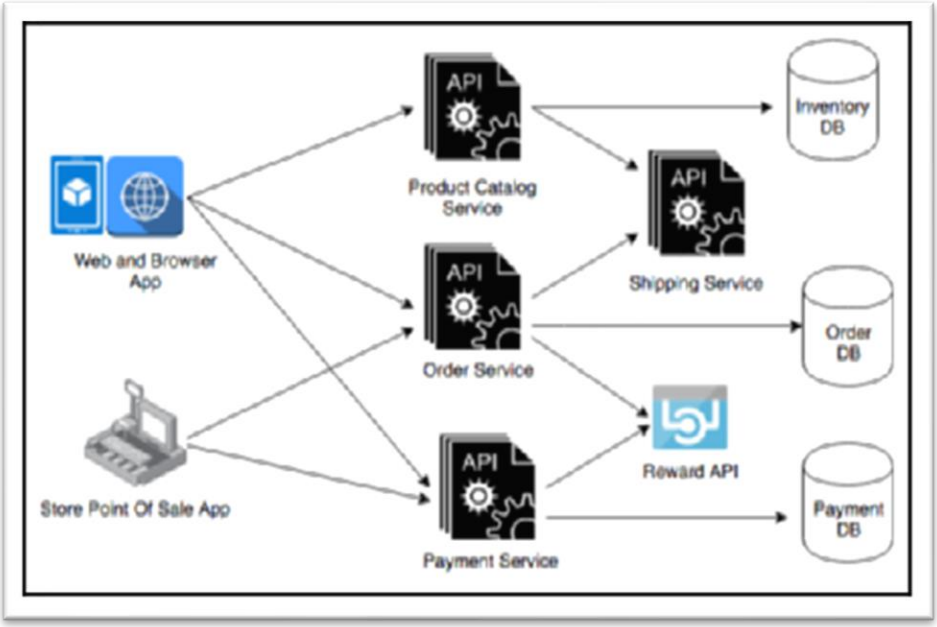
with continuous synchronization with the leading site. In case of a disaster, the team will scale up all servers and databases.

- Multi-site: The most expensive option, with near-zero RTO and RPO, this plan involves maintaining a replica of the leading site in a disaster recovery site with equal capacity actively serving user traffic. In case of a disaster, all traffic will be routed to an alternate location.

While organizations may opt for less expensive disaster recovery options, it is crucial to perform regular testing and ensure failover is working correctly. The team should establish a routine checkpoint in operational excellence to ensure business continuity during a disaster recovery event.

Extensibility and reusability

As businesses grow, they need to continuously add new features and scale their applications to handle an increased user base. A flexible and extendable solution design is necessary for organizations to modularize their application and build a platform with a group of features.



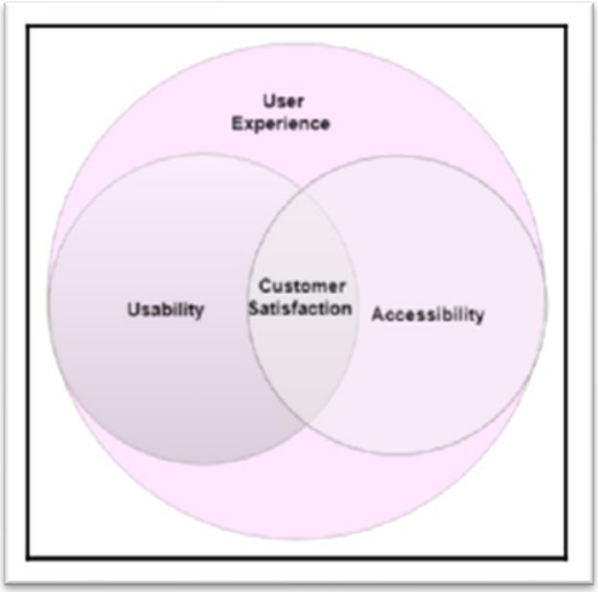
Extensible API-based architecture

To achieve solution extensibility, a loosely coupled architecture is preferred. This can be done by creating a RESTful or queue-based architecture that promotes loosely coupled communication between different modules or across applications. The API-based architecture in an e-commerce application is a good example of how independent services like product catalog, order, payment, and shipping can be utilized by an end-user application in a pick-and-choose manner. In addition, the Reward API allows for third-party API integration, which can help attract new customers and retain existing ones. The use of object-oriented analysis and design (OOAD) concepts such as inheritance and containership can also help create an API framework that is extendable and reusable for adding more features to the same service.

Usability and accessibility

To provide a seamless user experience, it is essential to make your application highly usable. This means that users should be able to navigate your application easily and efficiently without encountering any difficulties. Conducting user research and testing is crucial to determining the usability of your application and ensuring that it satisfies the user experience.

Usability is determined by how quickly users can learn to navigate your application, recover from mistakes, and perform tasks efficiently. If an application is complex and feature-rich, but difficult to use, it will not be effective.



Customer satisfaction with usability and accessibility

When designing an application for a global audience, it is important to consider the technical amenities and physical abilities of your user base. Accessibility design aims to make applications usable by everyone, regardless of their internet connection speed, device, or physical limitations. Solution architects should ensure that applications can be accessed over slow internet connections and are compatible with a diverse set of devices. They may also need to create a separate version of the application to achieve this.

Accessibility design includes features such as voice recognition and voice-based navigation, screen magnifiers, and text-to-speech functionality. Localization is also essential, as it allows the application to be available in specific languages for different regions.

Customer satisfaction is a crucial component of both usability and accessibility, and they both work together. Before starting the solution design process, solution architects should work with product owners to research users by conducting interviews, surveys, and gathering feedback on mock front-end designs. Understanding user limitations and empowering them with supporting features is crucial during application development.

After launching the product, A/B testing can be conducted to understand user reactions to new features. Continuous feedback should also be collected to improve the design of the application. This can be achieved by providing a feedback form or launching customer support.

Portability and interoperability

Interoperability refers to the capability of an application to interact with other applications through a common protocol or format. It is important for an application to seamlessly communicate with upstream systems to consume data and downstream systems to provide data, to achieve an end-to-end functionality, such, as from customer order to delivery. This is applicable across various industries, including e-commerce, healthcare, manufacturing, and telecom, among others.

During the design phase, a solution architect should consider application interoperability by identifying and addressing various system dependencies. This will ensure that an application can communicate with different systems without the need for data mapping, thus reducing costs. To achieve this, industries establish their own standard data exchange formats, which a solution architect should adhere to. Common formats for software design include JSON and XML, which are supported in RESTful API design and microservice architecture.

System portability refers to an application's ability to function across various environments, operating systems, and hardware, without significant changes or modifications. It is essential for achieving higher usability, considering the rapid pace of technology changes. Mobile applications must be compatible with major mobile operating system platforms such as iOS, Android, and Windows.

A solution architect must choose a technology that can achieve the desired portability and interoperability of the application. For example, Java may be preferred for cross-platform applications as it is supported across various operating systems. For mobile applications, a JavaScript-based language like React Native can provide cross-platform mobile app development.

Both interoperability and portability are critical aspects of architecture design, and a solution architect must carefully consider them as per industry requirements and system dependencies to ensure system extensibility and usability. Neglecting these aspects may lead to additional exponential costs in the long run.

Operational excellence and maintainability

Operational excellence is a crucial factor that can set your application apart by providing customers with high-quality, minimal outage services. In addition, it helps increase productivity for engineering and support teams by applying proactive operational excellence. Maintaining an application is also essential, as it helps reduce costs, minimize errors, and gain a competitive edge.

During solution design, a solution architect needs to consider operational readiness. This includes planning for deployment, updates, and long-term operation of the workload. It is essential to incorporate logging, monitoring, and alerting to capture incidents and take prompt action to maintain the best user experience. Automation should be used wherever possible, such as deploying infrastructures or changing the application code, to avoid human error.

Incorporating deployment methods and automation strategies into your design is critical to accelerating the time to market for new changes without affecting existing operations. Operational excellence planning should also include security and compliance elements, as regulatory requirements may change over time, and your application must comply with them to operate.

Maintenance can be proactive or reactive. You can choose to modernize your application to a new platform when a new operating system version is available or monitor system health and wait until the end of the software's life to make changes. In either case, changes should be made incrementally with a rollback strategy.

Setting up a continuous integration and continuous deployment (CI/CD) pipeline can automate the entire process.

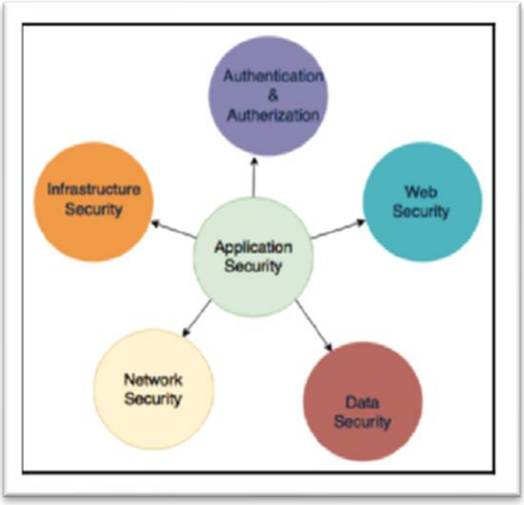
To ensure operational readiness, your architecture design should include appropriate documentation and knowledge-sharing mechanisms. For example, a runbook can be created and maintained to document routine activities, and a playbook can guide your system process through issues. This allows you to act quickly in the event of an incident. Post-incidence root cause analysis can be used to determine why an issue occurred and prevent it from happening again.

Operational excellence and maintenance are ongoing efforts, and every operational event and failure presents an opportunity to learn and improve. Analyzing operational activities and failures, experimenting, and making improvements can lead to better consideration of operational excellence in solution design.

Security and compliance

The design of a solution must prioritize security, as a lack of security can lead to breaches, customer mistrust, and significant business losses. Compliance with industry-standard regulations such as PCI, HIPPA, GDPR, and SOC is essential to ensure the protection of consumer data and provide guidance to organizations. Depending on the industry and region, local legislation must be adhered to through compliance needs. The main security aspects that must be addressed in solution design are authentication and authorization, web security, network security, infrastructure security, and data security.

Authentication involves specifying who can access the system, while authorization applies to user activities once they are inside the system or application. The solution architect must consider appropriate authentication and authorization systems, always beginning with the least privileged user and increasing access as needed based on their role. For corporate internal use, federated organizational systems such as Active Directory, SAML 2.0, or LDAP can be used. For mass user bases like social media websites or gaming apps, OAuth 2.0 and OpenID access are suitable, allowing users to authenticate through other IDs such as Facebook, Google, Amazon, and Twitter.



Security aspects in solution design

Web applications are vulnerable to external attacks and need to be protected against cross-site scripting (XSS), SQL injection, and Distributed Denial of Service (DDoS) attacks. Solution architects should plan to use a Web

Application Firewall (WAF) in combination with a Content Distribution Network (CDN) to prevent and manage these attacks.

Network security must be addressed to prevent unauthorized system access, host vulnerabilities, and port scanning. Solution architects should minimize system exposure by keeping everything behind a corporate firewall and avoiding internet access wherever possible. An Intrusion Detection System (IDS) and an Intrusion Prevention System (IPS) should be implemented in front of network traffic.

Infrastructure security is essential to physically block unauthorized user access to the server. Logical access must be secured by network security, and the appropriate firewall configuration must be implemented. If leasing a data center or using a private cloud, third-party vendors can handle physical security.

Data security is one of the most critical components that need to be secured. Data in transit should be secured with Secure Socket Layer/Transport Layer Security (SSL/TLS) and security certification, while data at rest should be secured using various encryption mechanisms, symmetric or asymmetric. Key management should be used to secure encryption keys with the right key management approach as per application requirements. Automation must be implemented to monitor and detect security breaches. DevSecOps is a trend among organizations that applies best practices to automate security needs and security responses during the software development life cycle, which will be explored in Chapter 12, DevOps and Solution Architecture Framework.

To adhere to local legislation compliance, the solution design must include an audit mechanism. PCI DSS is strictly required for finance, and all activities must be logged and sent to the auditor when required. PII data, such as customer email IDs, phone numbers, and credit card numbers, must be secured by applying encryption and limiting access. In on-premise environments, it is the customer's responsibility to secure the infrastructure and application and get certification for compliance. In public cloud environments such as AWS, this burden is eased.

Cost optimization and budget

In the realm of software development, the success of a solution is contingent upon balancing the limitations of the budget and the expectations of investors. To achieve maximum return on investment (ROI), solution architects must prioritize cost-saving measures in the design process. It is imperative that costs are optimized from the pilot phase through to the solution's implementation and launch. Cost optimization is not a one-time activity, but rather a continuous process that requires ongoing effort.

While cost-saving is a vital consideration, it is not the only constraint that needs to be taken into account. Solution architects must determine whether other components, such as delivery speed and performance, are more critical. A trade-off analysis is required to strike a balance between these constraints. Often, costs increase due to over-provisioning resources and a failure to take procurement costs into account. To avoid excessive under-utilization, solution architects must plan optimal resources carefully. Organizations must also have mechanisms in place to detect ghost resources, such as test environments that are no longer in use, and keep track of their inventory through automated discovery.

When selecting technology, it is essential to evaluate build versus source costs. Third-party tools should be considered when an organization lacks expertise and building costs are high. Ease of learning and complexity of implementation should also be taken into account. From an IT infrastructure perspective, capital expenditure versus operational expenditure must be assessed as maintaining a data center requires significant upfront investment. Multiple options are available, including public cloud, private cloud, multi-cloud, and hybrid approaches.

Cost needs to be continuously monitored and automated alerts set up against budget consumption. It is crucial to divide costs between organizational units and workloads, so responsibilities are shared across all groups. As more historical data is collected, the team needs to optimize operational support and workload to achieve ongoing cost optimization.

Principles of Solution Architecture Design

In the preceding chapter, you gained an understanding of the essential properties of solution architecture. These attributes must be kept in mind while designing a solution. This chapter delves into the principles of solution architecture design, which encompasses the various attributes required for solution design. The chapter outlines important and common design principles, which will be further used to create various design patterns, Solution Architecture Design Patterns.

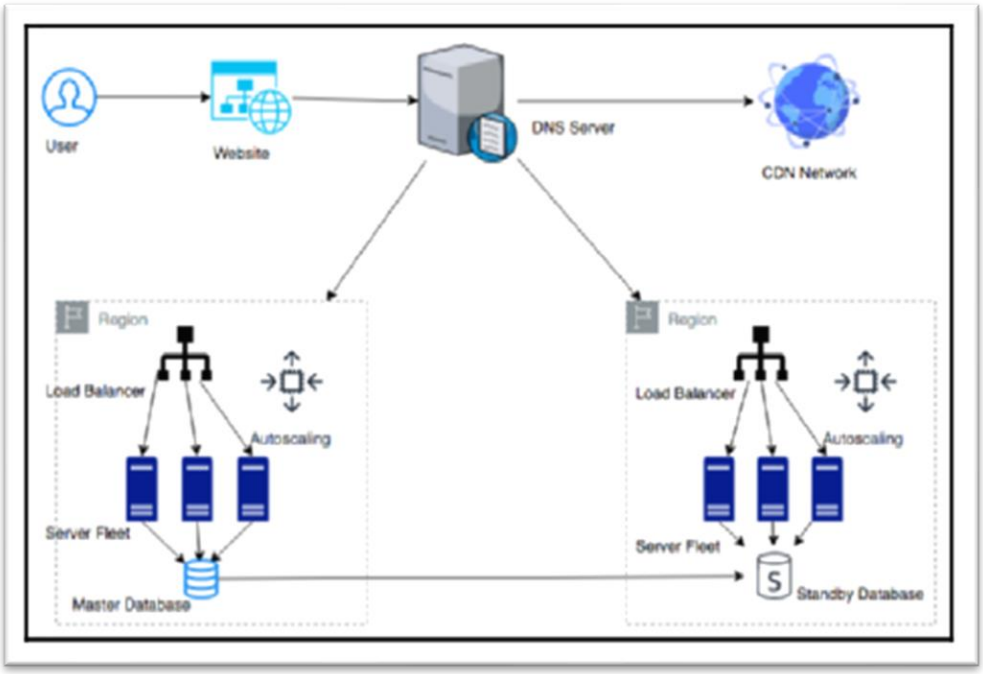
This chapter covers the following topics:

- Scaling Workload
- Building Resilient Architecture
- Designing for Performance
- Using Replaceable Resources
- Implementing Loose Coupling
- Thinking in Terms of Services
- Choosing the Right Storage for the Right Need
- Adopting a Data-Driven Design Approach
- Overcoming Constraints
- Incorporating Security Everywhere
- Automating Everything

The chapter focuses not only on designing scalable, resilient, and high-performance architecture but also on maintaining and securing the architecture. It emphasizes the importance of applying a data-driven and service-oriented approach and addresses various constraints and challenges that may arise. The chapter emphasizes the need for testing and automation to ensure that changes are implemented seamlessly. By following these principles, solution architects can design an optimal solution that meets the organization's requirements.

Building resilient architecture

Resilience is an essential aspect of application architecture that ensures the availability of services to customers and the ability to recover from failures. Achieving resiliency requires implementing best practices across all layers of architecture, including infrastructure, application, database, security, and networking. From a security perspective, Distributed Denial of Service (DDoS)attacks pose a significant threat to application availability. To prevent DDoS attacks, it is essential to keep as much of the application workload as possible in private networks and use a mechanism to identify suspicious traffic. Implementing a content distribution network (CDN) and web application firewall (WAF) rules can help prevent unwanted traffic, and scaling should be a last resort option.



Application architecture resiliency

At the application level, redundancy is critical to achieving resiliency. This can be achieved by duplicating server fleets in different physical locations and using the DNS server to route traffic between different regions. Additionally, using a CDN to distribute and cache static content, load balancers to route traffic to a fleet of servers, auto-scaling to adjust server capacity based on demand, and standby databases to ensure high availability are all important practices.

In the event of component failure, a backup system should be in place to recover and achieve architecture resiliency. To avoid cascading failure, where one component failure can bring down the entire system, timeout mechanisms, traffic rejection, idempotent operations, and circuit-breaking patterns should be implemented. Shallow and deep health checks can also be performed to ensure traffic is routed only to healthy application instances.

Design for performance

The increasing availability of fast internet has led customers to demand high-performance applications that load quickly. Organizations have recognized that application performance has a direct impact on revenue, and slow load times can negatively affect customer engagement. To remain competitive, modern companies must

prioritize high-performance applications. Therefore, performance must be considered at every layer of architecture design, much like resiliency. Teams must monitor performance and continuously work to improve it, as better performance leads to increased user engagement and return on investment. High-performance applications are designed to handle external factors that may cause slowness, such as slow internet connections.

In an ideal environment, automated scaling mechanisms handle the increased workloads without impacting application performance. However, scaling can cause application latency to decrease temporarily. It is therefore important to test for performance under increased load to ensure the desired concurrency and user experience.

Choosing the right server and storage options is crucial for performance. Memory and computing must be appropriately sized to handle the workload, as memory congestion can cause slow performance and even server crashes. Input/output operations per second (IOPS) should be carefully chosen for write-intensive applications to reduce latency and increase disk write speed.

Caching can significantly improve performance at every layer of architecture design. Browser caching, DNS caching, CDN caching, and memory caching are all effective options. Cache engines such as Redis and Memcached can serve frequent queries, and database caching can serve frequent queries from memory. Careful consideration of cache expiration and eviction is also necessary.

Performance is a critical aspect of solution design that directly impacts an organization's profitability. Therefore, the solution architect must think about performance and continuously strive to improve application performance.

Using replaceable resources

Hardware and software updates can cause difficulties in maintaining server configurations. Troubleshooting becomes a tedious task, and it can be challenging to determine which servers can be shut down. To address these issues, it's essential to consider treating servers as replaceable resources. This approach enables organizations to roll out and test new updates faster and with fewer issues.

Creating immutable infrastructure involves replacing both software and hardware during application upgrades. This can be accomplished by making applications stateless and avoiding the hardcoding of server IP or database DNS names. Using virtual machines to create a golden image of the server can make the process more manageable, allowing organizations to dispose of problematic servers and deploy new ones quickly.

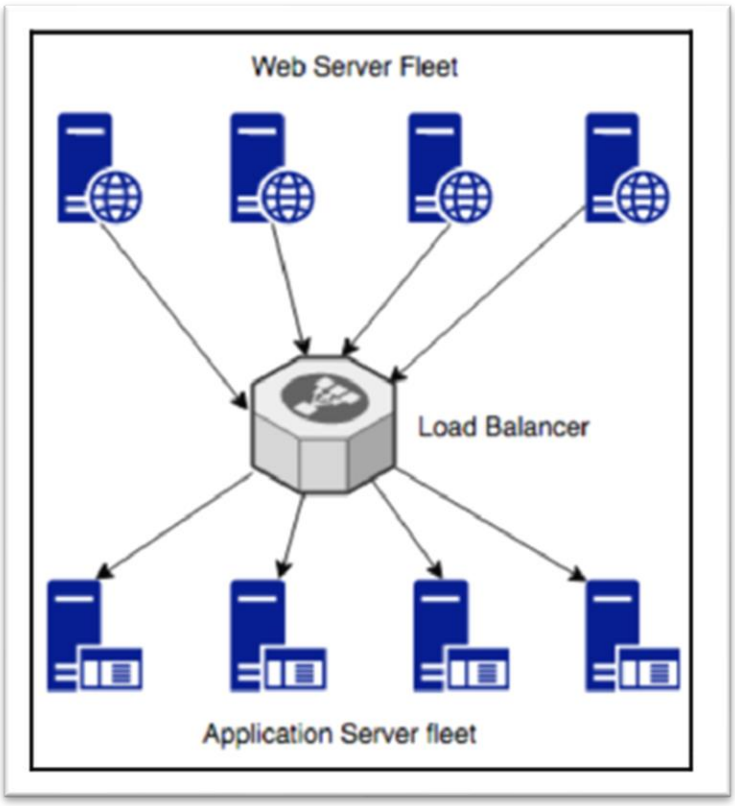
Canary testing is a popular method for applying rolling deployment with immutable infrastructure. It allows organizations to deploy updates safely by routing a small amount of traffic to a new server and gradually increasing it as everything goes well.

To make deployment smoother, solution architects should plan session management, avoid server dependency on hardcoded resources, and set a standard to use various rolling deployment strategies such as A/B testing or Blue/Green deployment. The idea is to treat servers like cattle, not pets, by using replaceable resources to ensure quick recovery and reduced troubleshooting time.

Think loose coupling

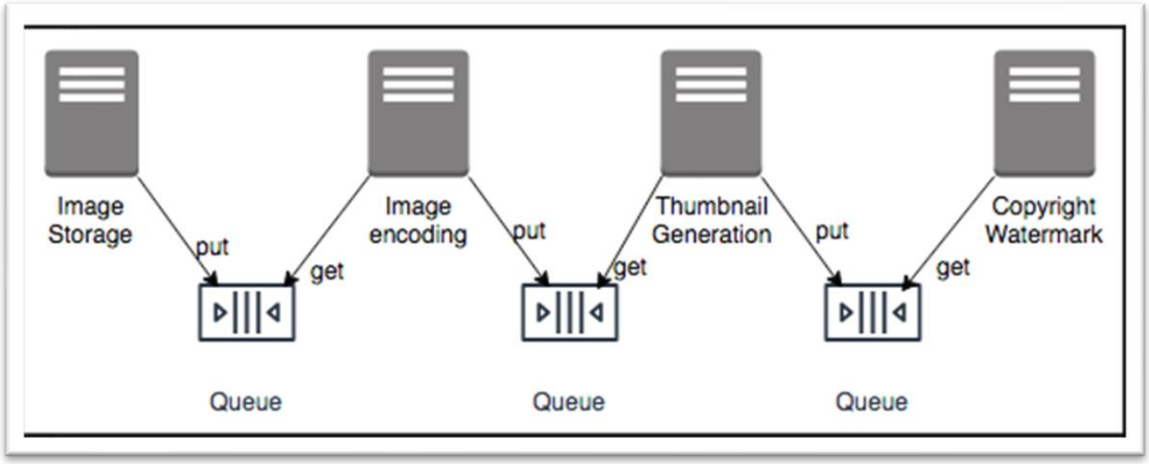
Traditional applications are built with tightly integrated servers, each with specific responsibilities. These applications depend on other servers for complete functionality. In a tightly coupled architecture, web servers and application servers have direct dependencies on each other, as shown in the diagram. If one server goes down, the entire system may fail, making scaling difficult.

To address these challenges, a loosely coupled architecture can be implemented with an intermediate layer such as a load balancer or a queue. A load balancer can automatically direct traffic to healthy servers and replace unhealthy instances gracefully. A queue-based architecture enables the asynchronous linking of systems, allowing independent work and increasing processing power.



Load balancer-based loosely coupled architecture

In complex systems, a loosely coupled architecture can be achieved through a service-oriented architecture (SOA) or microservice architecture. SOA involves independent services communicating with each other over a standard protocol, while microservice architecture enables the decoupling of application components.



Queue-based loosely coupled architecture

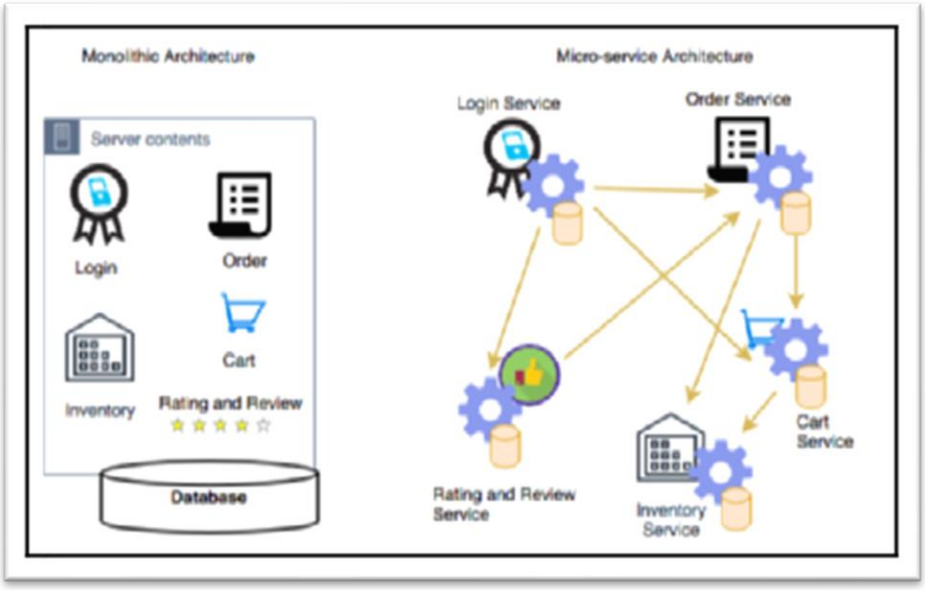
Loose coupling provides benefits such as scalability, high availability, and ease of integration.

Think service not server

In the preceding section, you were introduced to the concept of loose coupling and its significance in achieving scalability and fault tolerance in an architecture. Service-oriented thinking plays a crucial role in achieving a loosely coupled architecture, unlike server-oriented design, which can lead to a tightly coupled architecture and hardware dependency. Service-Oriented Architecture (SOA) helps in creating a solution design that is easy to deploy and maintain.

When it comes to service-oriented thinking, solution architects prefer SOA, which is predominantly based on two types of services - Simple Object Access Protocol (SOAP) and RESTful services. In SOAP-based architecture, messages are formatted in XML and transmitted over the internet using the SOAP protocol that builds on top of the HTTP. In contrast, RESTful architecture formats messages in XML, JSON, or plain text, and sends them over a simple HTTP. RESTful architecture is more popular due to its lightweight nature and simplicity compared to SOAP.

In modern times, microservice architecture is gaining popularity in SOA. Microservices are independently scalable, making it easier to expand or shrink one component of an application without impacting others. The image below shows an example of an e-commerce website that has a monolithic architecture, where all components are built in a single server, tied up with a single database, creating hard dependencies. In contrast, microservice architecture has independent components, each with its framework and database, allowing for individual scalability.



Monolithic to the microservice architecture

To migrate from monolithic to microservice-based architecture, developers can create applications made of small, independent components that form smaller parts to iterate. This modular approach reduces costs, size, and the risk of change. Each component can be created as a service, which reduces the blast radius in case of failure, and enables loose coupling, and scaling.

Solution architects must think of SOA while designing a solution. Services have a smaller surface area of code to maintain, are self-contained, and have no external dependencies. All prerequisites are included in the service, facilitating loose coupling, scaling, and reducing the blast radius in case of failure.

Using the right storage for the right need

The traditional relational database has been used by organizations for decades to store all types of data, even when it's not suitable for that kind of database.

Data Type	Data Example	Storage Type	Storage Example
Transactional, structured schema	User order data, financial transaction	Relational database	Amazon RDS, Oracle, MySQL, PostgreSQL, MariaDB Microsoft SQL Server
Key-value pair, semi-structured, unstructured	User session data, application log, review, comments	NoSQL	Amazon DynamoDB, MongoDB, Apache HBase, Apache Cassandra, Azure Tables
Analytics	Sales data, Supply chain intelligence, Business flow	Data warehouse	IBM Netezza, Amazon Redshift, Teradata, Greenplum, Google BigQuery
In-memory	User home page data, common dashboard	Cache	Redis cache, Amazon ElastiCache, Memcached
Object	Image, video	File-based	SAN, Amazon S3, Azure Blob Storage, Google Storage
Block	Installable software	Block-based	NAS, Amazon EBS, Amazon EFS, Azure Disk Storage
Streaming	IoT sensor data, clickstream data	Temporary storage for streaming data	Apache Kafka, Amazon Kinesis, Spark Streaming, Apache Flink
Archive	Any kind of data	Archive storage	Amazon Glacier, magnetic tape storage, virtual tape library storage
Web storage	Static web contents such as images, videos, HTML pages	CDN	Amazon CloudFront, Akamai CDN, Azure CDN, Google CDN, Cloudflare
Search	Product search, content search	Search index store and query	Amazon Elastic Search, Apache Solr, Apache Lucene
Data catalog	Table metadata, data about data	Meta-data store	AWS Glue, Hive metastore, Informatica data catalog, Collibra data catalog
Monitoring	System log, network log, audit log	Monitor dashboard and alert	Splunk, Amazon CloudWatch, SumoLogic, Loggly

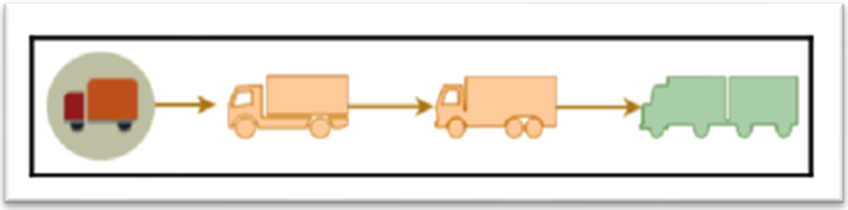
This is like using a Swiss Army knife to build a house—it has multiple tools, but they work only to a limited capacity. When selecting data storage, solution architects must consider factors like data durability, availability, latency, throughput, size, load, integrity, and queries. Different types of data require different storage options, such as NoSQL databases for user session data, data warehouses for semi-structured data, and archive storage for cold data. Data temperature should also be considered, as hot, warm, and cold data have different latency requirements. Choosing the right storage can help improve performance, scalability, and cost savings. It is crucial to choose the right data storage for the right need to save costs and improve performance.

Think data-driven design

Data collection and management are at the core of any software solution. Applications such as e-commerce websites and banking systems collect and store customer and transaction data, making it critical to handle, store, and secure data appropriately. Choosing the right storage and monitoring capabilities helps ensure application performance and uptime, as well as providing insights that can drive business decisions. For example, using cache storage for low-latency applications or content distribution networks for improved page load times. Collecting and analyzing data such as sales and review sentiment data can help to improve the overall customer experience and increase profitability. As a solution architect, it is essential to consider not only application design but also the overall business value proposition.

Overcoming constraints

One popular method for prioritizing customer requirements is MoSCoW, which divides requirements into four categories: Must-have (Mo), Should-have (S), Could-have (Co), and Won't-have (W). It is important to plan a minimum viable product (MVP) for the customer with must-have requirements and gradually add should-have requirements in subsequent iterations. This phased approach helps to efficiently utilize resources and overcome the constraints of time, budget, scope, and resources.



MVP approach to building the solution

The MVP approach also helps to determine customer needs and reduce the waste of resources. The approach involves delivering a working product with essential features that customers can use, and building upon it based on their requirements. This approach also helps to generate revenue early, which can be used to ask for more resources.

When designing applications, it is important to use the most common technique across the organization, which helps to remove everyday challenges. It is also important to ensure that the application is upgradeable and can integrate with different platforms, including legacy systems. The RESTful service model is popular when teams can use any technology for their development.

In conclusion, taking an agile approach and considering constraints as challenges instead of obstacles can help to create a customer-centric product while efficiently utilizing limited resources.

Adding security everywhere

When designing solutions, security is an essential aspect that needs to be considered. Any security gap can have a devastating effect on the business and organization's future. Therefore, it is important to understand your security needs before starting the application design. Security needs to be considered at both the hardware and software levels, including the physical security of the data center, network security, identity, and access management, data security in transit and at rest, and security monitoring.

Application design needs to balance security requirements, such as encryption, with factors like performance and latency. Encryption can impact performance, so it is important to consider when it is necessary. Regulatory compliance is also important, especially for regulated industries like healthcare, finance, or government. Compliance requirements commonly include data protection and auditing. Designing with comprehensive logging and monitoring can fulfill these requirements.

It is important to apply security thinking and regulatory needs while designing solutions. Security automation should also be implemented to reduce and mitigate security incidents.

Automating everything

Automation can prevent human errors that lead to accidents and save costs, increase productivity, and free up human resources. When designing a solution, identify repeatable tasks to automate, such as application testing, IT infrastructure, logging, monitoring, alerting, deployment, and security. Automated testing can speed up deployment and launch while automating infrastructure creates a replica of the environment, avoids configuration errors, and reduces deployment time. Automated monitoring and logs ensure the smooth running and functioning of the application, and deployment automation using continuous integration and deployment allows for frequent launches and agility. Security automation detects suspicious activity and alerts the team to take preventive action. Always consider automation as a critical component when designing an application.

Cloud Migration and Hybrid Cloud Architecture Design

This section discusses the importance of cloud computing as a developing trend for solution architecture. Public cloud computing is a primary destination for hosting applications, making it necessary to understand cloud thinking and not undervalue the preposition and migration method. Cloud computing is the delivery of IT resources over the web on an as-needed basis, and organizations pay for what they use. The chapter will cover topics such as the benefits of cloud-native architecture, creating a cloud migration strategy, steps for cloud migration, creating a hybrid cloud architecture, designing a cloud-native architecture, and popular public cloud choices. The end goal is to provide readers with an understanding of cloud computing and its various strategies and steps, so they can design cloud-native architecture and understand popular public cloud providers.

Benefits of cloud-native architecture

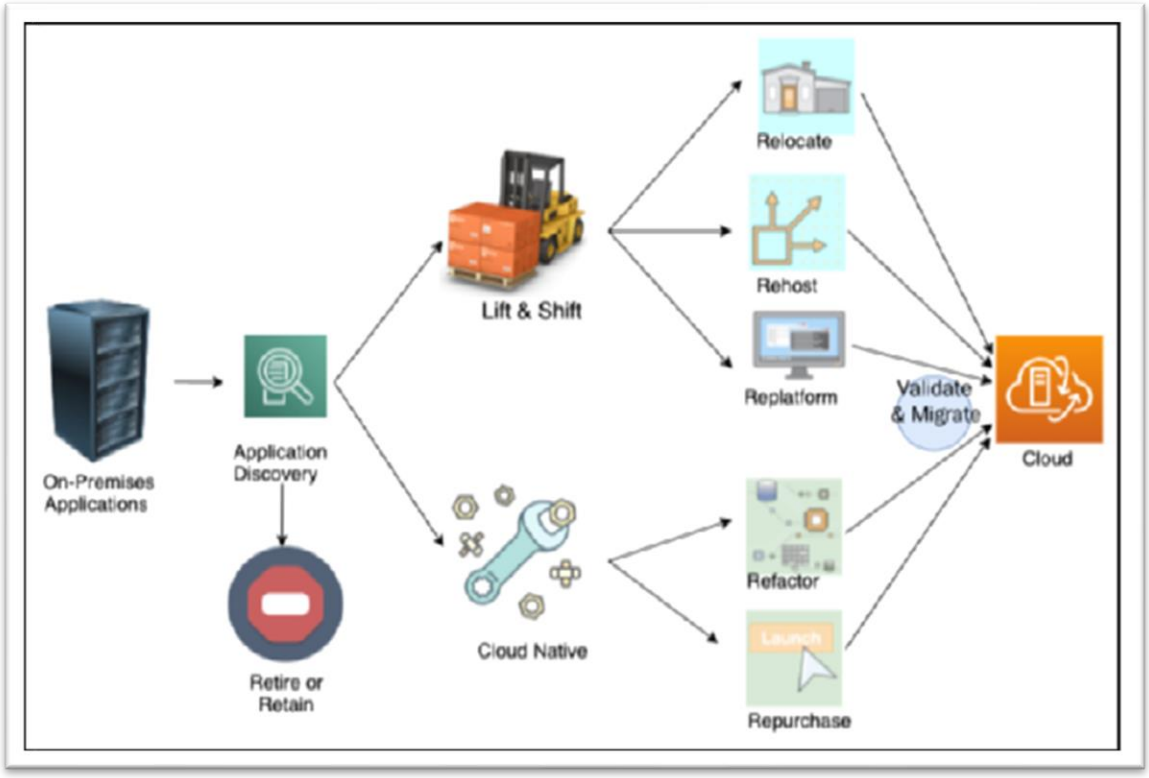
In recent years, technology has advanced rapidly, leading to the emergence of new companies in the cloud world, disrupting traditional organizations. The cloud's pay-as-you-go model reduces risk in experimentation, making it possible for organizations to grow rapidly without upfront costs. The cloud agile approach helps employees develop innovative ideas quickly without waiting for the long infrastructure cycle. With the cloud's elasticity, customers can provision resources per demand instantly, reducing costs and improving the customer experience.

The cloud also provides access to cutting-edge technologies, making it possible for enterprises to get infrastructure worldwide and achieve scalability and elasticity. Different organizations adopt various strategies when it comes to cloud adoption. Common use cases include hosting web applications in the cloud and using them for digital transformation. The cloud helps with data processing and analytics since it's less expensive to collect, store, analyze, and share data.

Building a solution architecture for the cloud requires cloud thinking and understanding the cloud's in-built capabilities. It's important to optimize workloads properly and have a holistic view of components. Cloud-native monitoring and alerting mechanisms are also essential for optimization. Automation is a significant benefit of the cloud, reducing errors and speeding up time-to-market. Security is also important, and cloud-native tools can help with monitoring, alerts, and automation. In the next section, you will learn about various strategies for cloud migration.

Creating a cloud migration strategy

The reasons for migrating to the cloud are crucial in determining migration strategy and prioritizing applications. Cloud migration strategies are often mixed, and multiple tools are used to execute them. The most used strategies are Lift and Shift, Replatform, Refactor, and Retire.



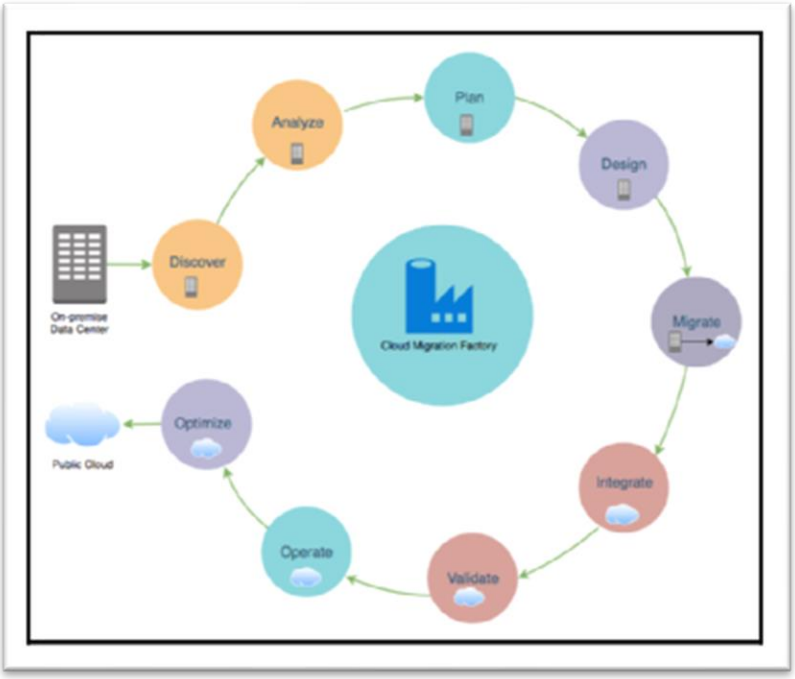
Cloud migration strategy

Lift and Shift involves minimal work but does not take advantage of cloud-native features. Rehost, re-platform, and relocate are often utilized to execute the Lift and Shift approach. Rehost is the quickest and most preferred method for migrating to the cloud. Replatform upgrades the platform as a part of the cloud migration project. Relocation involves moving workloads to the cloud using the accelerated migrations strategy. The reasons for using these strategies are discussed in detail, and the Lift and Shift approach is discussed in further detail, including its three most common migration strategies.

The cloud-native approach requires re-architecting and rewriting an application before migrating to the cloud, which may seem like more upfront work and slower migration to the cloud, but the benefits of a cloud-native application include scalability, security, agility, and cost-efficiency. Refactoring is the process of re-architecting and rewriting the application before migration. Repurchasing involves purchasing a new license or replacing an existing application with another cloud-compatible application. Retaining involves running applications on-premises, while retiring involves decommissioning rarely used or redundant applications. Applications that are not suitable for cloud migration or do not bring any benefit to the cloud can be retained or retired.

Steps for cloud migration

This section explains the importance of setting up a Cloud Center of Excellence (CoE) to standardize the migration process for managing multiple applications in the cloud. The CoE comprises experienced personnel from various IT and business teams that focus on accelerating cloud expertise within the organization. The cloud migration factory defines the migration processes and tools, including the steps involved in cloud migration, as shown in a diagram.



Cloud migration steps

The steps involved in the cloud migration process are Discover, Analyze, Plan, Design, Migrate, Integrate, Validate, Operate, and Optimize. The first step is to assess and prioritize the applications for migration. This is achieved by getting a complete inventory of the IT assets in the environment. The discovery phase of the migration project is essential in identifying all the IT assets, including servers, applications, dependencies, and performance metrics.

During the discovery phase, it is crucial to gather business details about resources to determine migration strategy and create a migration plan. The discovery landscape depends on several factors such as what has already been migrated to the cloud, the estimated duration for the migration project, and the number of phases involved in the migration process.

Automated tools are available to automate the discovery process and provide more detailed information in a variety of formats. The complexity of the discovery process depends on the organization's workload, and it is typically run for at least a couple of weeks to gather more holistic information about the environment. Once the discovery phase is complete, the analysis phase begins, which involves analyzing the gathered information to determine application connectivity and capacity requirements to design and architect the target cloud environment and identify costs.

The process of analyzing server and application dependencies for cloud migration involves examining network connectivity data, port connections, system, and process information on the hosts. Different tools can be used to visualize server dependencies or run queries to list servers with specific attributes. Grouping servers and applications for migration scheduling requires identifying patterns in host configurations such as embedded prefixes, tags, and metadata. To right-size the target environment, performance metrics should be analyzed to revise right-size mapping information or assign a higher priority to under-provisioned servers. Data captured during the discovery process should be analyzed to determine target network details, workload distribution, and the migration phase.

The analysis should be combined with resource availability and business requirements to prioritize the cloud migration workload. Based on the discovery and analysis, an appropriate cloud migration strategy can be determined for each server/application, including choosing a migration strategy, assigning migration priority, and documenting the business driver for migration.

Creating a hybrid cloud architecture

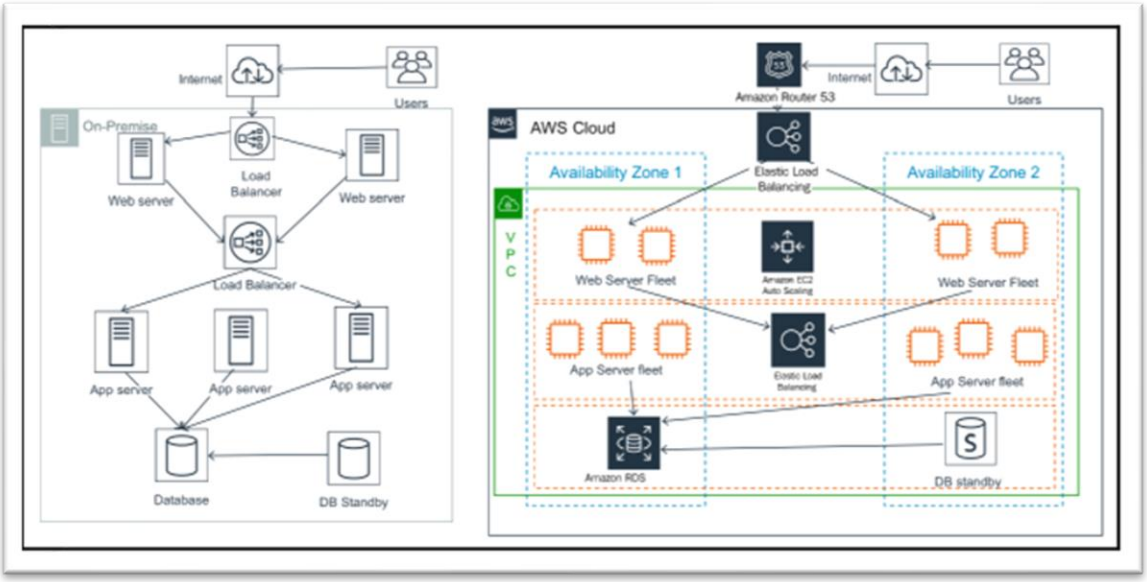
This text outlines the planning phase of a cloud migration project. This phase is preceded by the portfolio discovery phase, during which IT assets are identified and a target destination environment is architected. The main objectives of this phase include selecting a migration strategy, defining success criteria, determining resource sizing, identifying migration patterns, and creating a migration plan.

- Prioritizing applications for migration depend on the organization's strategy and the results of the discovery and analysis phase. Key planning aspects include gathering performance metrics, creating test and user acceptance plans, defining cutover and rollback strategies, and identifying roles and responsibilities.
- It is also important to track KPIs before, during, and after migration. The Scrum methodology can be used to efficiently migrate applications to the cloud by breaking down the process into sprints. These sprints can be structured to include a discover/analyze phase, a plan/design phase, migrate phase, and final cutover.

Designing a cloud-native architecture

The design phase of application migration to the cloud is essential for ensuring that the application meets the required success criteria after migration.

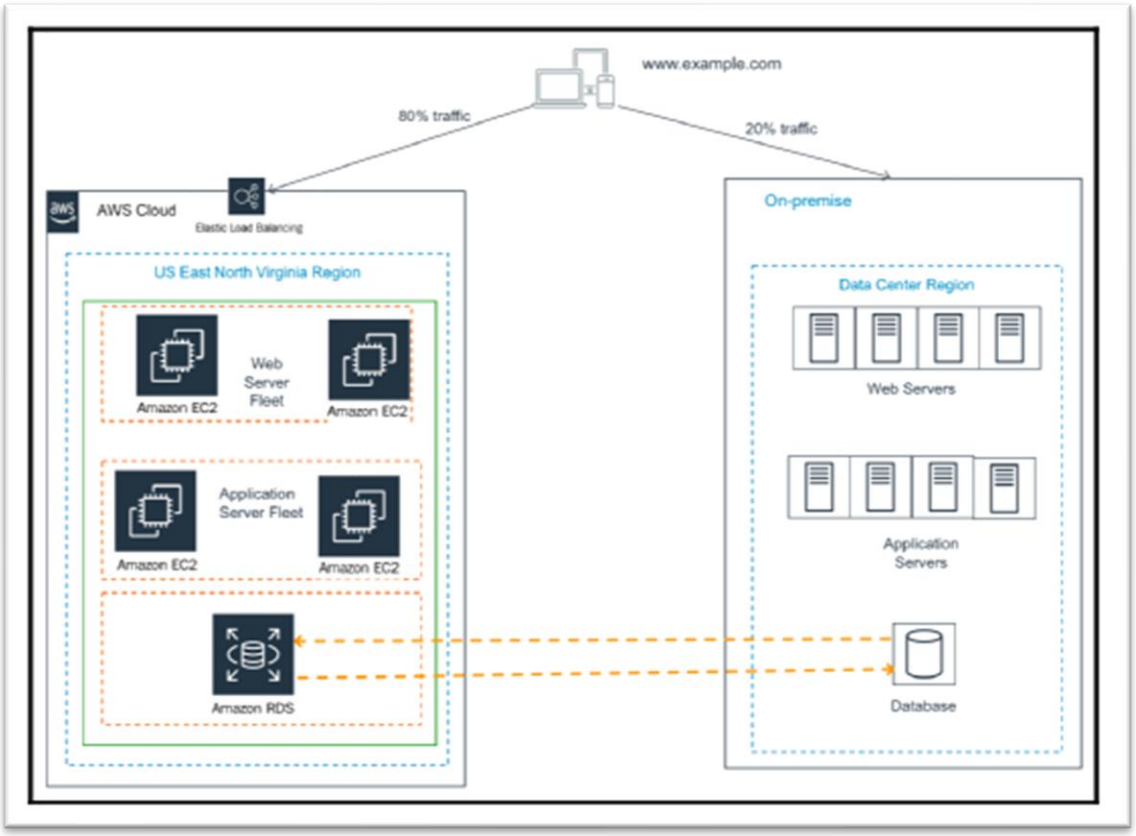
During this phase, opportunities to enhance the application can also be identified. To create and maintain a new architecture for the application, it is necessary to understand the foundational architecture of the organization's on-premises and cloud networks, including user accounts, network configurations, security, monitoring, and governance. The design of the application architecture should aim to benefit from the global cloud infrastructure, improve security, and mitigate risks. The architecture should also be scalable and support growth in users, traffic, or data without affecting performance. Stateless components and distributed processing approaches can be used to handle state information and large amounts of data, respectively.



On-premise to AWS Cloud architecture

Serverless architecture can be used to reduce operational complexity and cost. A migration design from on-premise to AWS cloud can involve rehosting web servers and introducing autoscaling, adding elastic load balancers, migrating application servers using refactor, and changing the platform for the database tier to Amazon RDS. The design phase requires creating a detailed design document for the architecture of each application.

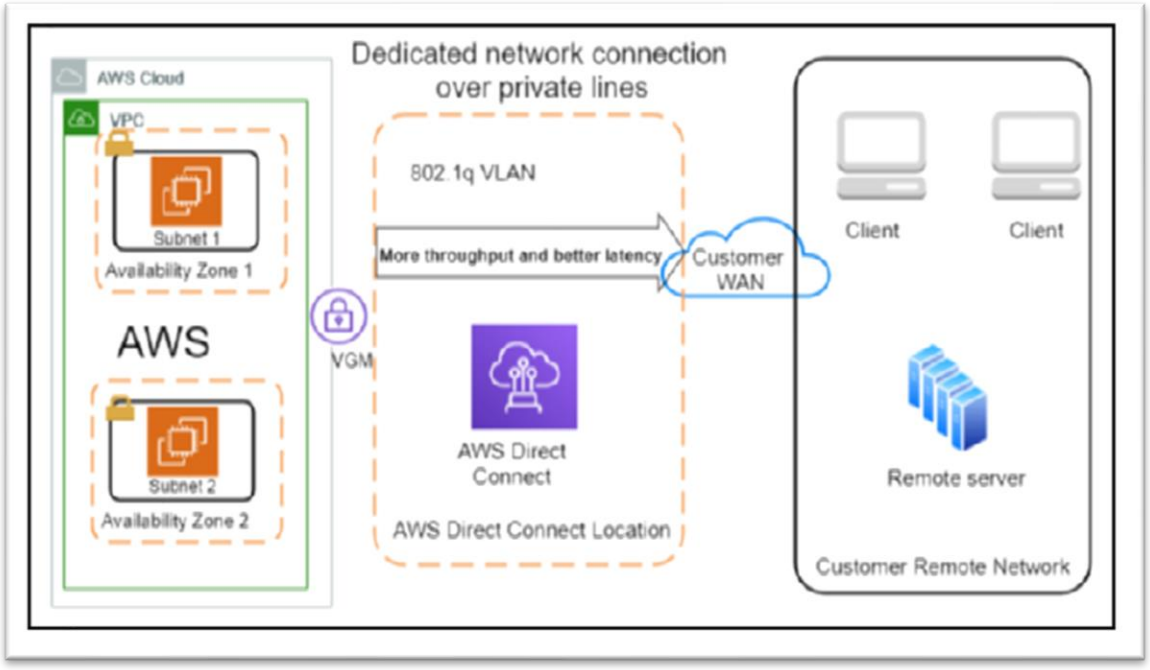
The migration plan should include identifying sprint teams, schedules, prioritized backlog, and application-specific pre-steps.



Live migration cutover using blue-green deployment

Server migration can be achieved through several methods such as the host or OS cloning technique, Disaster Recovery (DR) replication technique, disk copy method, VM copy method, user data copy method, and containerization method. The choice of method will depend on factors such as the OS environment, whether the on-premise servers are running as virtual machines, and whether you can containerize your application. Once the migration is completed, functionality and performance tests need to be performed for connectivity to external applications or any other criteria as needed.

The operation phase of a cloud migration involves identifying process changes and training to support cloud adoption goals. The cloud differs from a traditional data center in that the cloud provider manages the physical data center, allowing companies to provision and de-provision computing resources as needed. The IT operations required in the cloud include server patching, service and application logging, cloud monitoring, event management, cloud security operations, configuration management, cloud asset management, change management, and business continuity with disaster recovery and high availability.

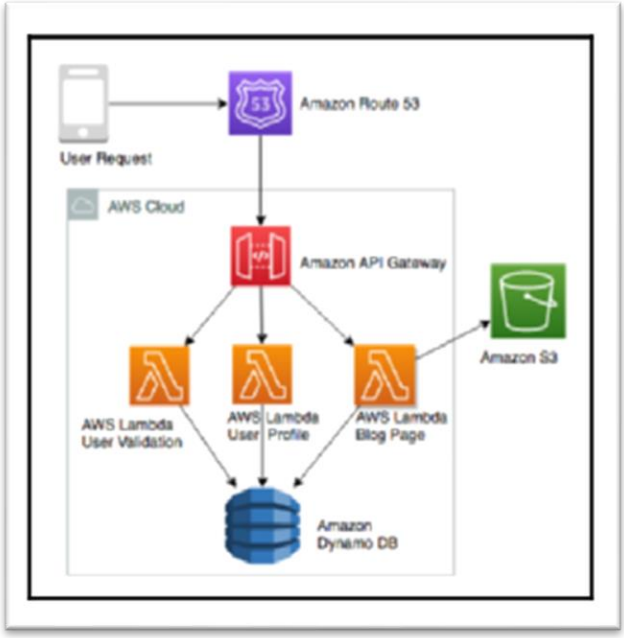


Hybrid cloud architecture (on-premise to cloud connectivity

DevOps methodology promotes collaboration and coordination between developers and operational teams to deliver products or services continuously. Application optimization in the cloud involves optimizing costs, security, reliability, operational excellence, and performance.

To optimize costs, companies need to understand their current cloud environment and the cost of each resource deployed in the cloud. Companies can reduce costs by automating instance deployment, rightsizing resources, using auto-scaling, and adjusting utilization based on price and need.

Cloud-native architecture involves designing applications that are built from the ground up using cloud services and features in the best possible way.



Cloud-native micro-blogging application architecture

This approach is not limited to hosting an application in the cloud but instead focuses on utilizing cloud-native features such as containerization, serverless application design, data lakes, monitoring, and auditing services. The benefits of cloud-native architecture include fast-paced innovation, simplified infrastructure building, and the ability to scale resources on-demand while only paying for what is used.

An example of a cloud-native serverless architecture for a micro-blogging application using AWS services is provided in the text. This architecture enables scalability, high availability, disaster recovery, and cost savings. Furthermore, the architecture provides the ability to replicate and tear down infrastructure easily, thereby enabling the building of experimental systems.

Popular public cloud choices

The major cloud providers at the time of writing are Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP). AWS is one of the oldest and largest cloud providers and offers a wide range of offerings in IaaS, PaaS, and SaaS, including cutting-edge technologies such as machine learning, artificial intelligence, and blockchain. Azure provides similar services as AWS and has a range of offerings in the cloud, including popular Microsoft offerings like Microsoft Office and Microsoft SharePoint. GCP provides cloud offerings in the areas of computing, storage, networking, and machine learning, and has a global network of data centers available as infrastructure as a service. Other cloud vendors, such as Alibaba Cloud, Oracle Cloud, and IBM Cloud, are also available in the market.

However, the major markets are captured by AWS, Azure, and GCP. The choice of which cloud provider to use is up to the customer and can be impacted by factors such as the availability of desired functionality or existing relationships with providers. Some large enterprises choose a multi-cloud strategy to utilize the best providers.

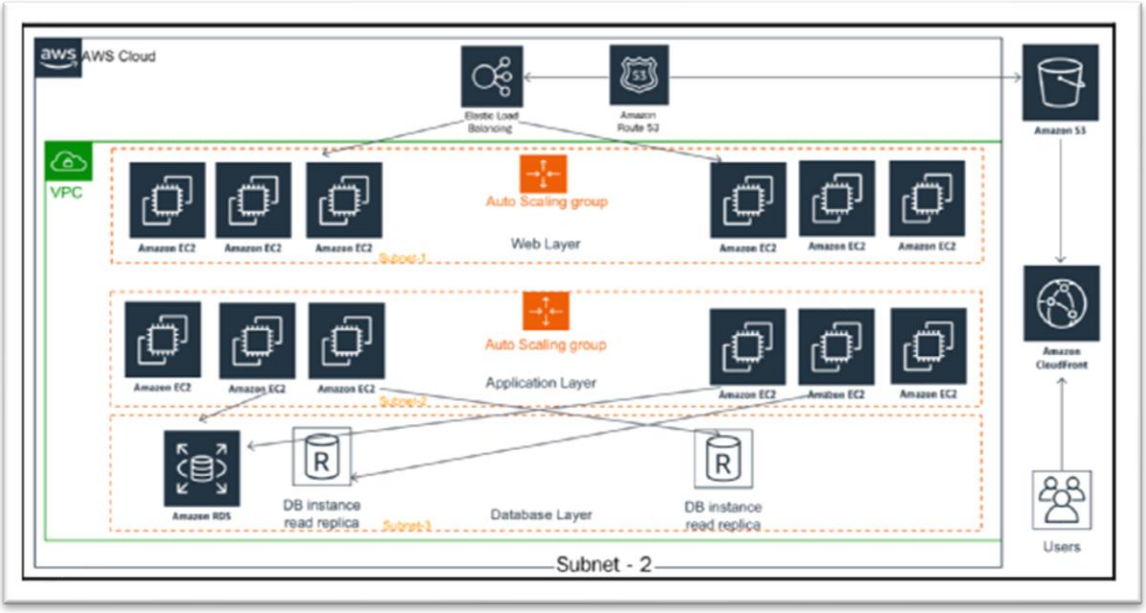
Solution Architecture Design Patterns

A solution architect must select the appropriate design approach based on user requirements and the architecture's constraints, such as cost, performance, scalability, and availability. This chapter provides insight into various solution architecture patterns, reference architectures, and their real-world application.

It covers significant architecture patterns, such as layered, event-driven, microservice, loosely coupled, service-oriented, and RESTful architectures.

Building an n-tier layered architecture

Multitier architecture is a design pattern that requires loosely coupled design principles and scalability and elasticity attributes. The idea is to divide product functions into multiple layers, such as presentation, business, database, and services so that each layer can be implemented and scaled independently.



Three-tier website architecture

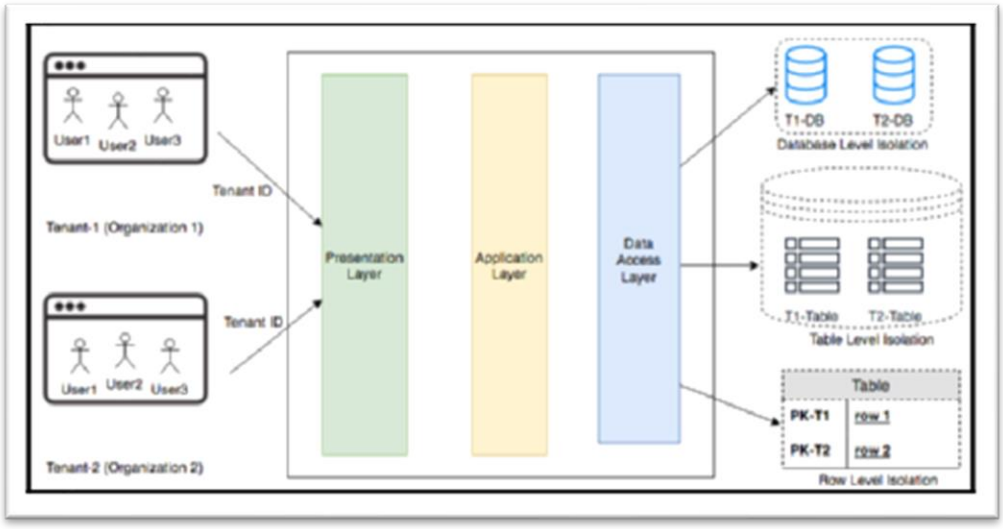
This architecture provides flexibility to add new features in each layer without disturbing the features of other layers. In terms of security, each layer can be kept secure and isolated from others, so if one layer is compromised, the other layers won't be impacted. The most common multitier architecture is three-tier architecture, which consists of the web layer, application layer, and database layer.

The web layer is responsible for the user interface, while the application layer contains the core business logic. The database layer stores all the user and application data. Each layer has its own Auto Scaling configuration, and a network boundary prevents access to other layers. The number of tiers added to the design should be carefully considered, as each layer requires its own fleet of servers and network configurations.

The architect should decide the number of tiers based on application complexity and user requirements.

Creating multi-tenant SaaS-based architecture

This section introduces the concept of multitenancy architecture and its popularity due to the increasing adoption of digital transformation. Multitenancy architecture allows multiple customers (tenants) to share a single instance of software and supporting infrastructure, where each tenant is isolated by their unique configuration, identity, and data. The Software-as-a-Service (SaaS) model is built on multitenancy architecture, allowing SaaS providers to own everything from the hardware to the software, thus offloading an organization's responsibilities to maintain and update the application.



Multi-tenant SaaS architecture

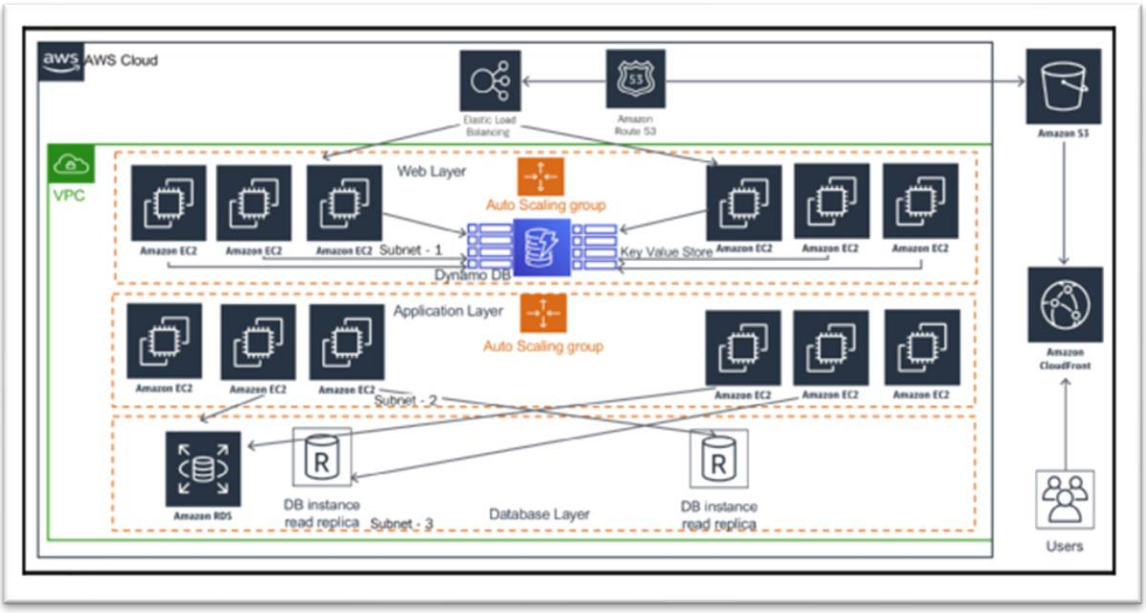
The SaaS model provides each tenant with customizable user interfaces without any code changes, and multiple customers can share a common infrastructure, lowering the cost further. Some popular SaaS providers are Salesforce CRM, the JIRA bug tracking tool, and Amazon QuickSight.

The architecture diagram in the section shows two organizations (tenants) using the same software and infrastructure, with each tenant having data level isolation at the data access layer. This is achieved by providing each tenant with database level isolation, table level isolation, or row level isolation.

For enterprise customers, it is essential to carefully assess whether a SaaS solution is the right fit for them based on their unique features' requirements. They also need to compare the total cost of ownership when taking a build versus buy decision. However, the SaaS model is becoming highly popular as organizations can focus on their business and let the experts handle the IT side of it.

Building stateless and stateful architecture designs

To persist user activity across devices and maintain their state until the transaction is complete, user-session information needs to be stored in a persistent database layer like NoSQL. Traditionally, a stateful architecture stores session information on the server itself, making it difficult to share information between servers and supporting modern microservice architecture.



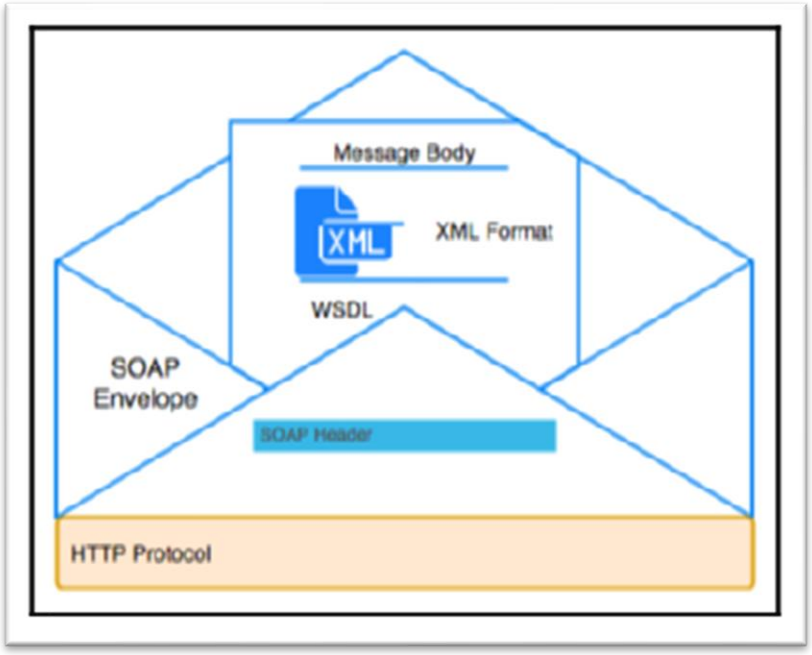
A stateless application architecture

As a stateful application doesn't support horizontal scaling very well and can't be replaced when the application state persists in the server, it's better to focus on the shared session state using a stateless method. A stateless architecture can eliminate the overhead of creating and maintaining user sessions, allowing consistency across application modules and improving performance. Adopting a stateless pattern can be complicated, but it can be achieved by developing applications using a microservice approach with REST design patterns and deploying them in containers.

In the stateless pattern, all user sessions are stored persistently in a NoSQL database, and client-side storage, such as cookies, is used for the session ID. This architecture lets you use the scale-out pattern without losing user state information, and it removes the session timeout issue. However, caution must be exercised to prevent the performance of the data store from becoming a bottleneck.

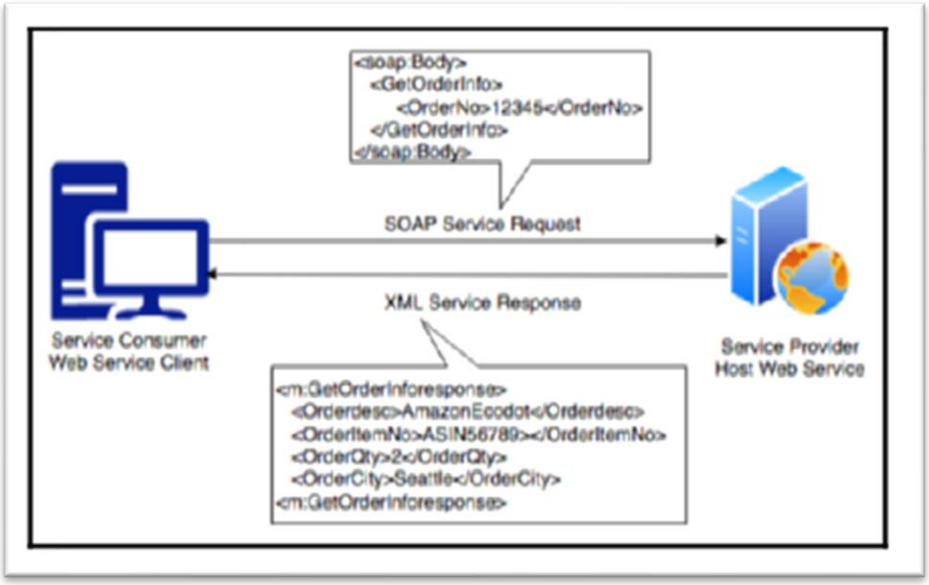
Understanding SOA

In Service Oriented Architecture (SOA), application components communicate with each other via a communication protocol over the network.



SOAP Envelope for web service data exchange

SOA is used to integrate business processes by breaking down monolithic applications into independent services that perform end-to-end functions.



SOAP-based web service

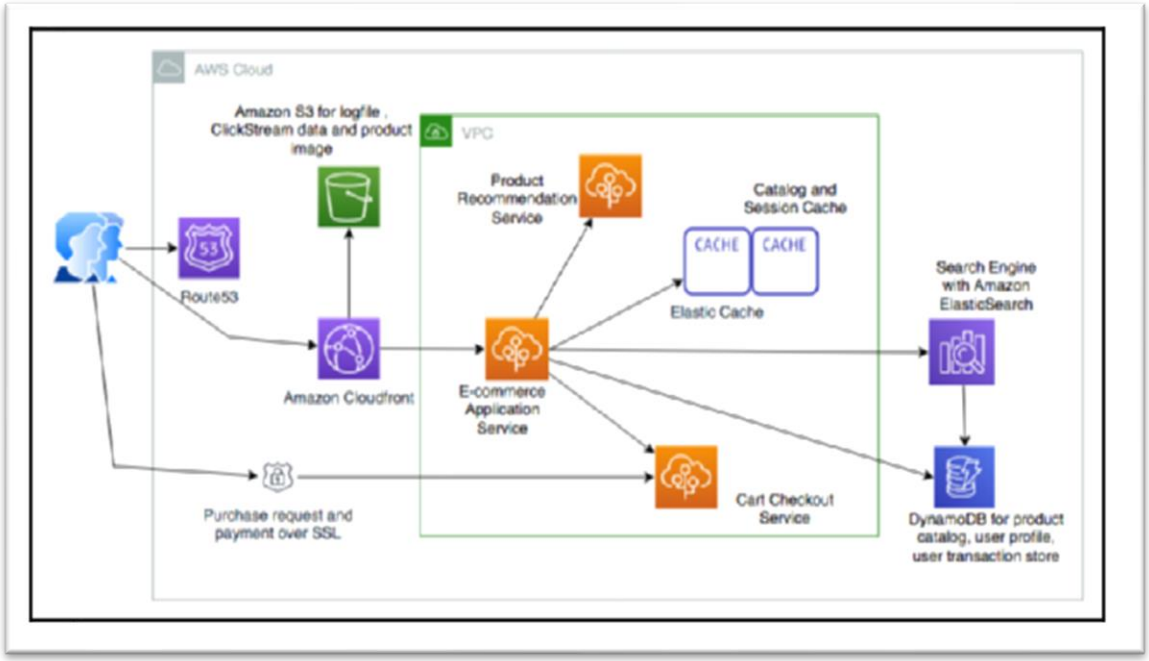
An SOA can also split resources into separate instances of services, enabling a modularized application by function, which helps manage throughput based on individual database needs.

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
        <env:Header>
        <n:orderinfo xmlns:n="http://exampleorder.org/orderinfo">
        <n:priority>1</n:priority>
        <n:expires>2019-06-30T16:00:00-09:00</n:expires>
        </n:orderinfo>
        </env:Header>
        <env:Body>
        <m:order xmlns:m="http://exampleorder.org/orderinfo">
        <m:getorderinfo>
        <m:orderno>12345</m:oderno>
        </m:getorderinfo>
        </m:order>
    </env:Body>
```

SOA can increase complexity and overhead, requiring appropriate governance and the right choice of tools and automation.

Attributes	REST	SOAP
Design	Architectural style with an informal guideline	Predefined rules with a standard protocol
Message Format	JSON, YAML, XML, HTML, plaintext, and CSV	XML
Protocol	HTTP	HTTP, SMTP, and UPD
Session State	Default stateless	Default stateful
Security	HTTPS and SSL	Web Services Security and ACID compliance
Cache	Cached API calls	Cannot cache API calls
Performance	Fast with fewer resources	Needs more bandwidth and compute power

Two ways to implement SOA are Simple Object Access Protocol (SOAP) web service architecture and Representational State Transfer (REST) web service architecture.



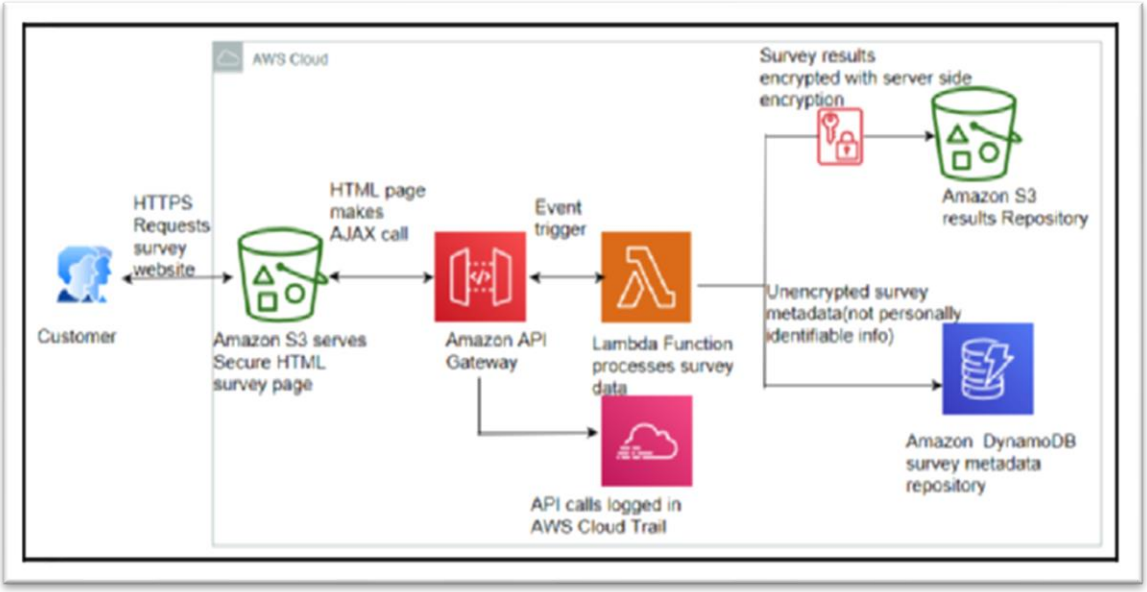
E-commerce website SOA architecture

SOAP is a messaging protocol that exchanges data in XML format while REST is an architecture style that defines the standard for loosely coupled application design using the HTTP protocol for data transmission. REST offers better performance than SOAP, allows multiple messaging formats, and has a lightweight architecture.

Building serverless architecture

Traditionally, to develop an application, one needed a server where the necessary software and operating system could be installed.

During development and deployment, the server had to be maintained and scaled up to meet user demand. This maintenance required a lot of effort and was not directly related to the business problem.



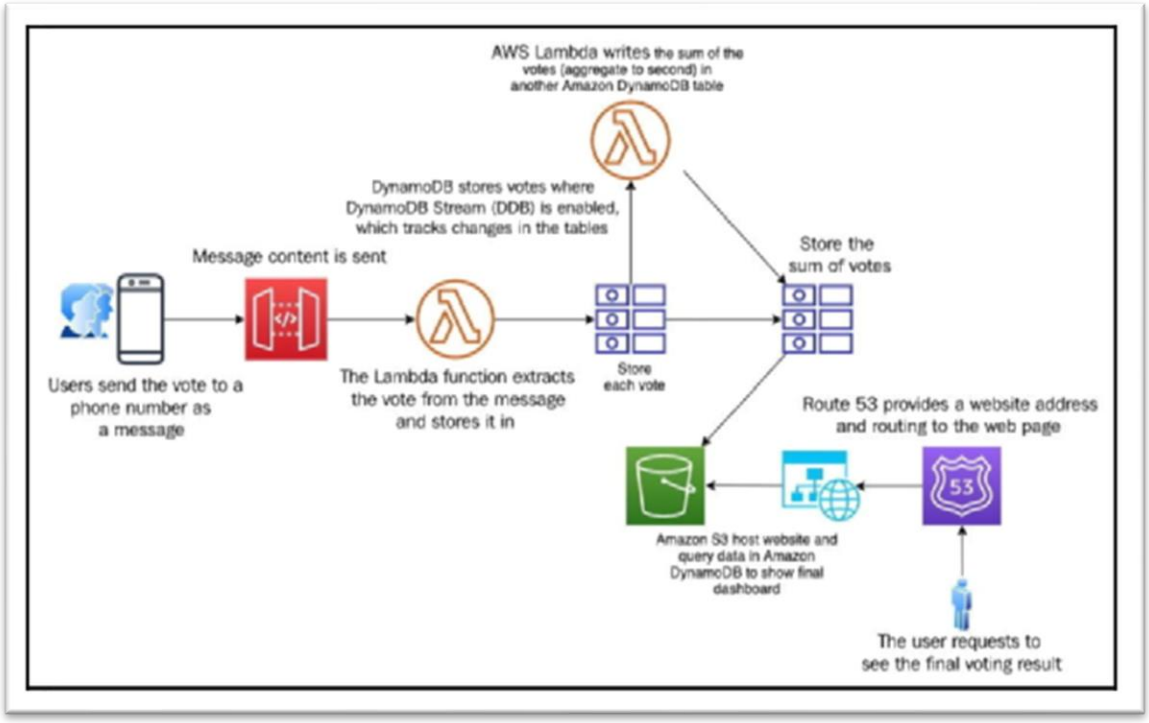
Serverless architecture for a secure survey delivery

Serverless architecture frees developers from infrastructure management and maintenance. AWS Lambda functions are a popular example of Function as a Service (FaaS), a serverless technology provided by Amazon Web Services. Amazon API Gateway provides RESTful endpoints to AWS Lambda functions, while Amazon DynamoDB provides a serverless NoSQL data store, and Amazon S3 provides serverless object data storage.

A reference architecture for a secure survey delivery using serverless architecture is shown in a diagram. The popularity of serverless architecture is on the rise, and reference architectures using serverless services will become increasingly common. The adoption of RESTful-style architectures is also increasing, with the concept of microservices gaining traction.

Creating microservice architecture

The microservices architecture is designed in REST-style web services that can be scaled independently, which allows for easier expansion and shrinkage of the relevant components. This type of system is more fault-tolerant, which means that it can degrade gracefully to avoid cascading failures. Microservices also reduce the surface area of code that needs to be maintained and should be built with no external dependencies, which reduces interdependency between application modules and enables loose coupling.

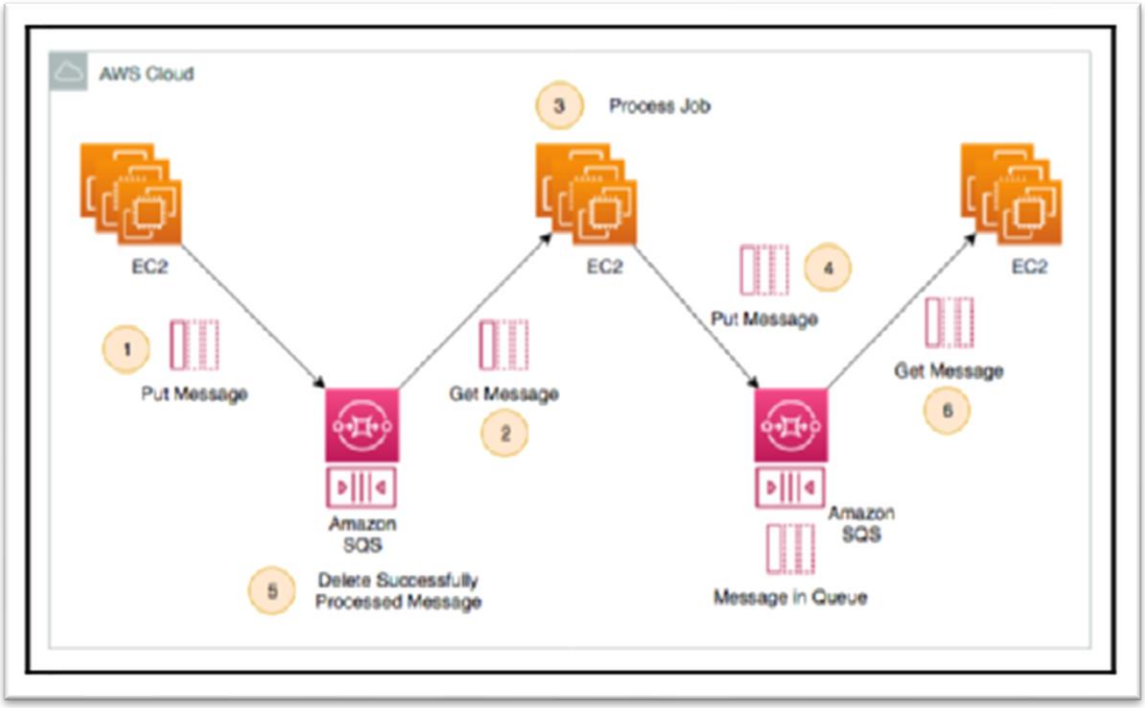


Microservice based real-time voting application architecture

Best practices for designing a microservices architecture include creating a separate data store for each microservice, keeping servers stateless, creating a separate build for each microservice, deploying in a container, using blue-green deployment, and monitoring the environment. An example of a microservice-based architecture is the real-time voting application, which is also serverless. The voting application collects user votes from mobile devices and consolidates them in small microservices that store data in a NoSQL-based Amazon DynamoDB database. The voting data is then aggregated and displayed in a dashboard hosted in Amazon S3. The application uses message queues to achieve accurate loose coupling and prevent application throttling.

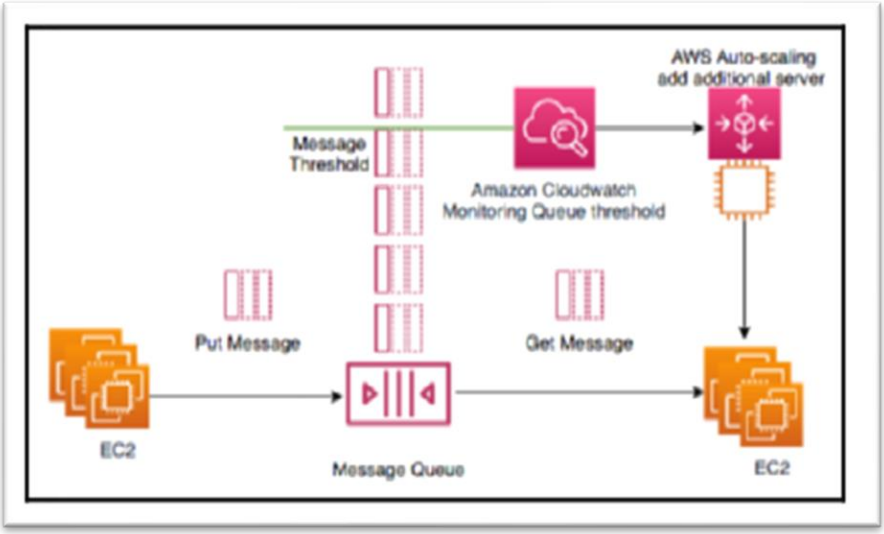
Building queue-based architecture

This section discusses the limitations of RESTful architecture in microservices and introduces a solution to this problem using a queue-based architecture.



Queuing chain pattern architecture

A queue-based architecture allows fully asynchronous communication and provides a loosely coupled architecture by adding message queues between services. The section explains the terminology of a queue-based architecture, including message, queue, producer, consumer, and message broker. The queuing chain pattern is introduced as a typical pattern used in queue-based architecture, which helps to remove a single point of failure and increase the number of servers that can process messages in parallel. The job observer pattern is also discussed, which uses an Auto Scaling group to maintain performance based on the number of messages in the queue.



Job observer pattern architecture

The section concludes by mentioning the limitations of the asynchronous pull method and introducing the concept of event-driven architecture to drive communication between various architecture components.

Creating event-driven architecture

The event-driven architecture enables you to connect a sequence of events to complete a feature flow, such as generating an order invoice and sending an email after completing a payment. Message queues serve as a central point in the event-driven architecture, which can be based on the publisher/subscriber model or the event stream model.

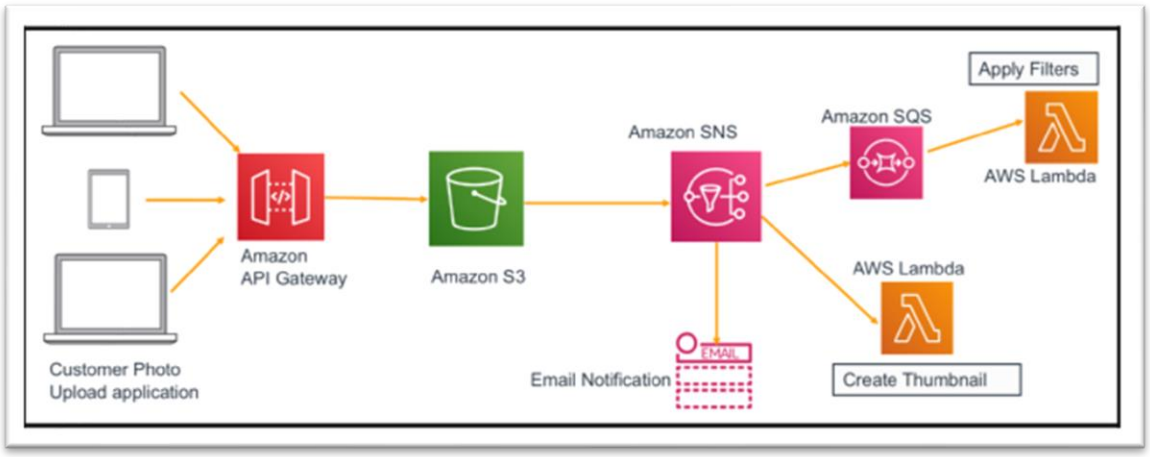
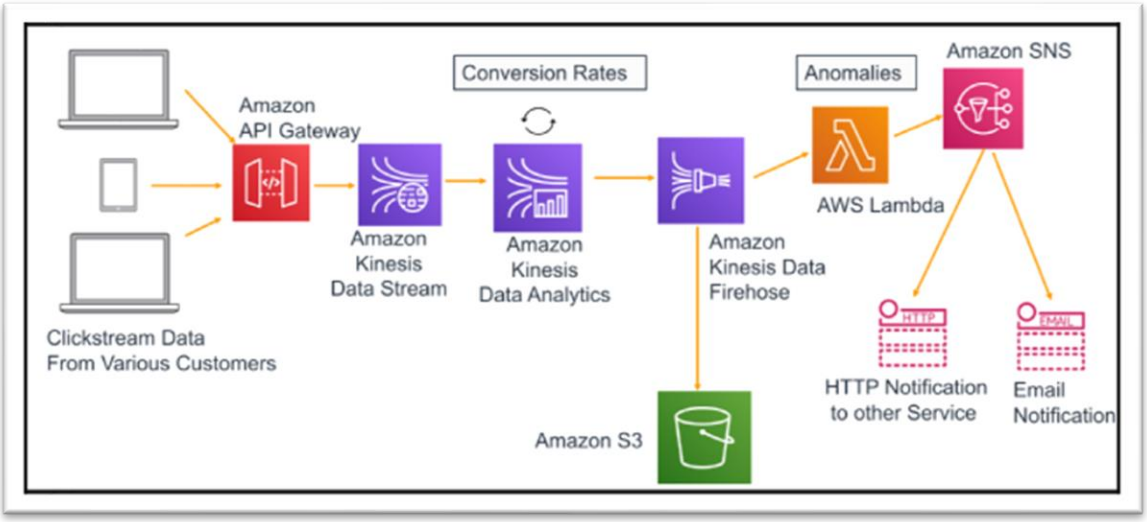


Photo studio application pub/sub-event-driven architecture

In the publisher/subscriber model, when an event is published, all subscribers receive a notification, and each subscriber can take necessary actions accordingly. For instance, a photo studio application enriches a photo with different filters and sends a notification to the user, and the email service, Amazon SQS queue, and AWS Lambda function act as subscribers. In the event stream model, the consumer reads from the continuous flow of events coming from the producer, which is useful for capturing clickstream logs and sending alerts for any detected anomalies.

Principles of Solution Architecture Design

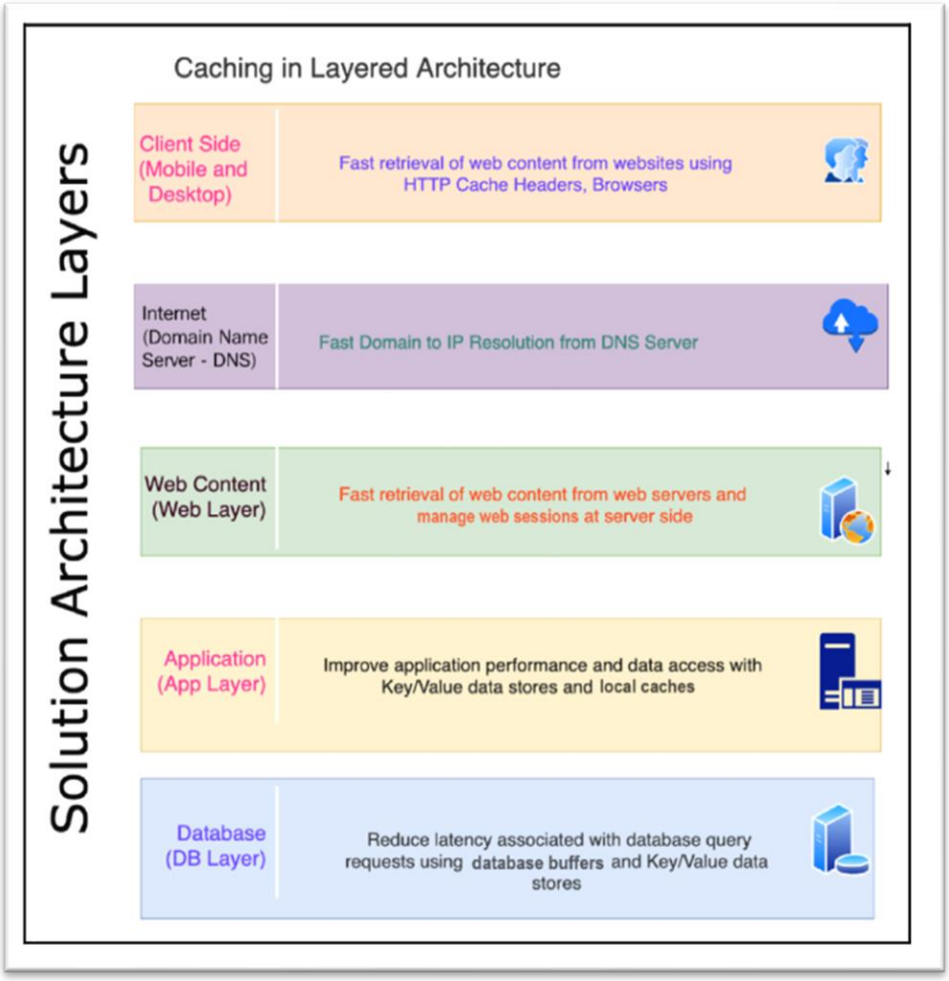


Clickstream analysis event stream architecture

Amazon Kinesis ingests, processes, and stores streaming data, and Kinesis Data Analytics calculates conversion rates for the aggregated data and then sends results to Amazon Kinesis Data Firehose, which stores data files in Amazon S3 storage. A Lambda function reads from the event stream and starts examining the data for anomalies, and when it detects one, it sends a notification to the campaign team via email. Event-driven architecture decouples the producer and consumer, making the architecture extendable for adding a new consumer at any time. Caching is crucial for achieving good application performance and can be implemented in every architecture layer and component.

Building cache-based architecture

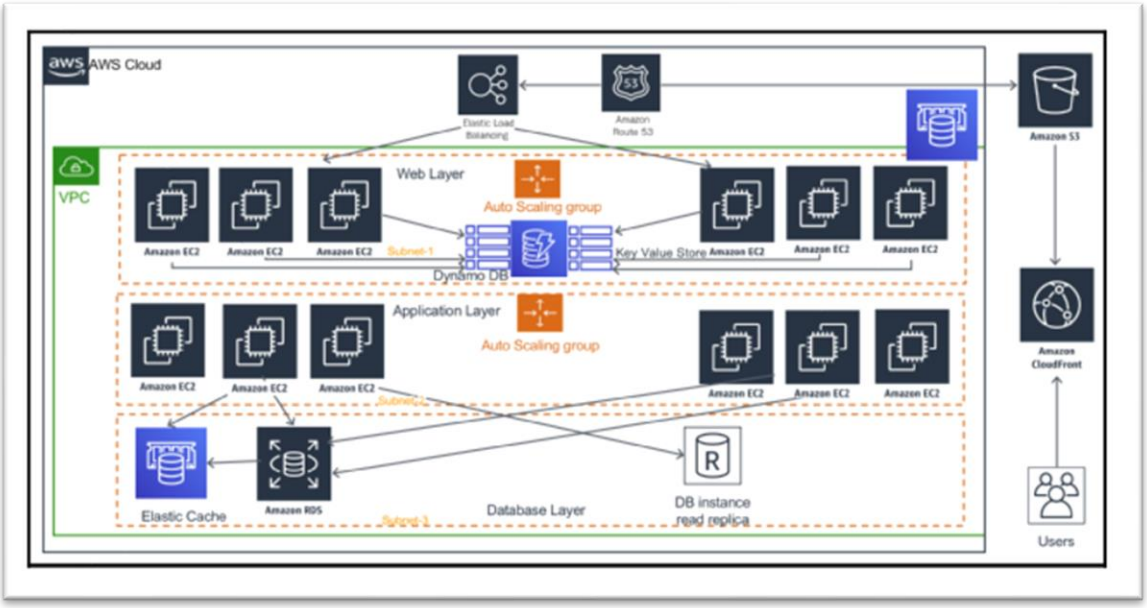
Caching is a technique that temporarily stores data or files in an intermediate location between the requester and permanent storage. This reduces network throughput and increases application speed, leading to lower costs. Caching can be applied at various layers of the architecture, such as the web, application, data, and network layers, to reuse previously retrieved data.



Caching at the architecture layers

Caching is typically done using the server's random-access memory (RAM) or in-memory cache engines. However, a dedicated caching layer that is independent of the application life cycle is recommended for distributed environments to achieve the best performance.

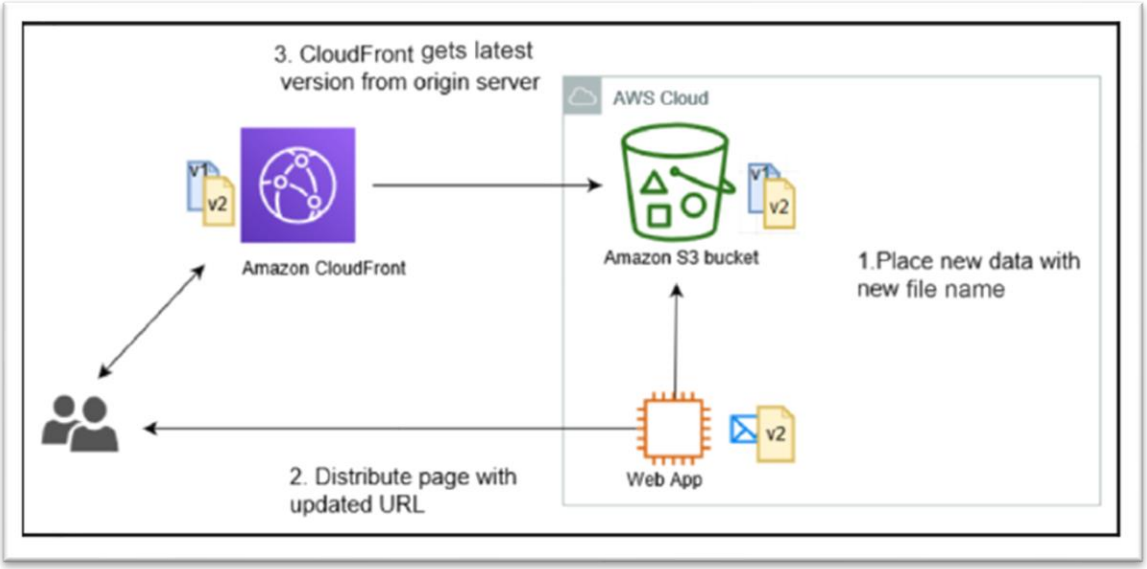
There are different caching mechanisms at each layer of architecture, such as client-side caching, DNS caching, web caching, application caching, and database caching. Redis and Memcached are the most popular caching engines. While Memcached is faster and suitable for low-structured data, Redis is more persistent and can handle complex data structures.



Cache distribution pattern architecture

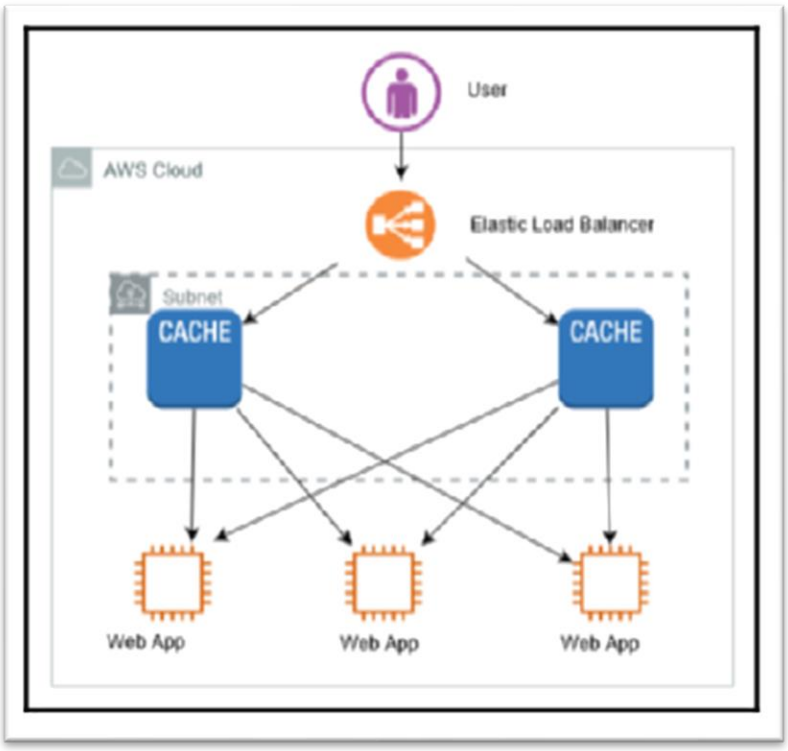
The cache distribution pattern is a way to offload web pages through caching. Caching is applied at the presentation, persistence, and application layers in a three-tier web architecture.

Amazon Route 53, S3, CloudFront, DynamoDB, Elastic Load Balancing, and ElastiCache are some of the caching services provided by Amazon Web Services. In general, static content is cached; however, depending on demand, caching dynamic or unique content might still yield some performance gain.



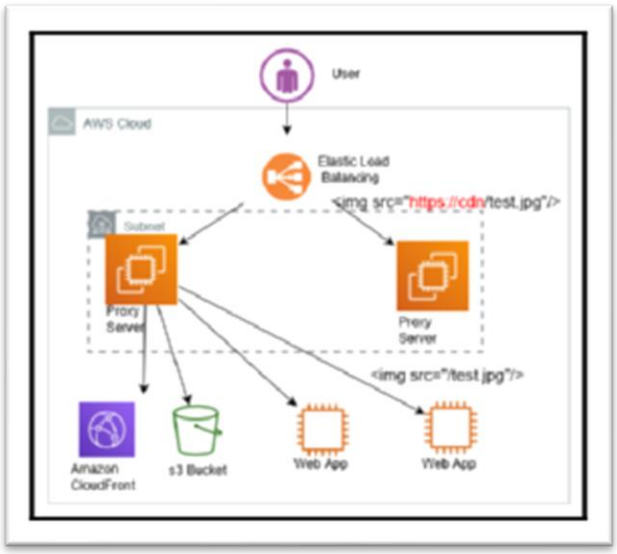
Rename distribution pattern architecture

The Rename Distribution Pattern helps to update the cache as soon as new changes are published, without waiting for the file's TTL to expire, by uploading an updated file with a new filename and updating the web page with the new URL.



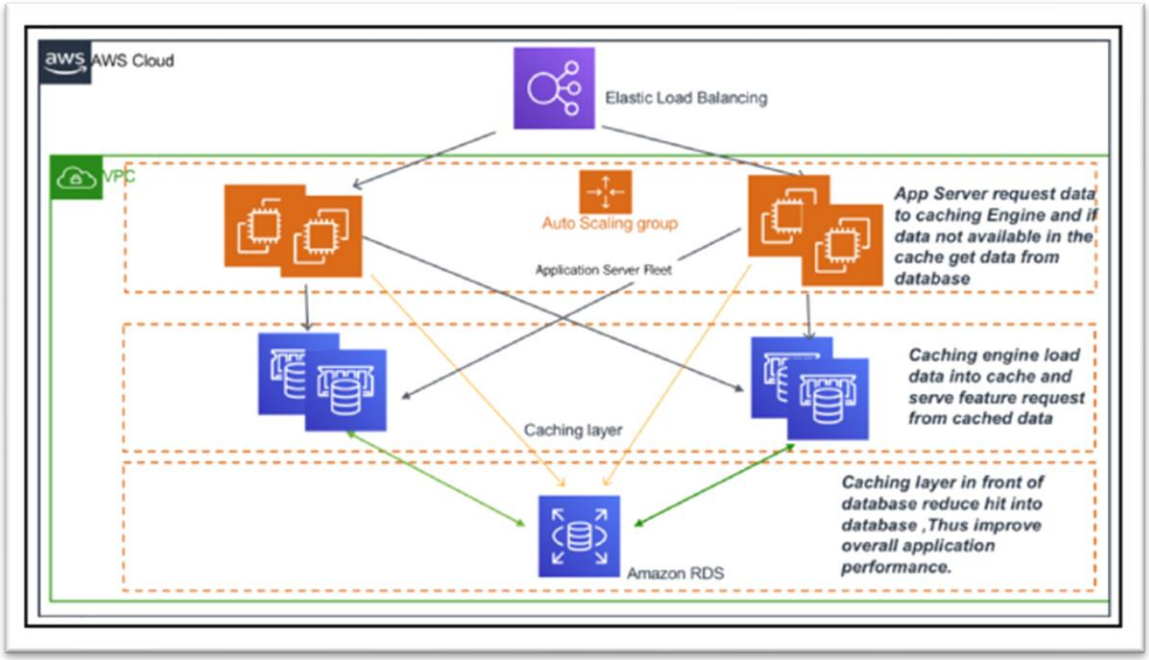
Cache proxy pattern architecture

The Cache Proxy Pattern adds a cache layer to increase application performance, reducing the load of content generation for dynamic content. You can cache information in the cache layer, and you need to maintain multiple copies of the cache to avoid the single point of failure.



Rewrite proxy pattern architecture

The Rewrite Proxy Pattern helps to change the content delivery destination without modifying the actual application, by providing a proxy server in front of the web server fleet.



Application caching pattern architecture

The App Caching Pattern allows you to reduce the load on the database by adding a cache engine layer between the application servers and the database. The article explains two types of caching, Lazy Caching and Write-through, and describes the popular caching engines Redis and Memcached.

Memcached	Redis
Offers multithreading	Single-threaded
Able to use more CPU core for faster processing	Unable to utilize multi-core processor, which results in comparatively slow performance
Supports key-value style data	Supports complex and advance data structures
Lacks data persistence; loses the data stored in cache memory in the event of a crash	Data can persist using built-in read replicas with failover
Easy maintenance	More complexity involved owing to the need to maintain the cluster
Good to cache flat strings such as flat HTML pages, serialized JSON, and more	Good to create a cache for a gaming leader board, a live voting app, and more

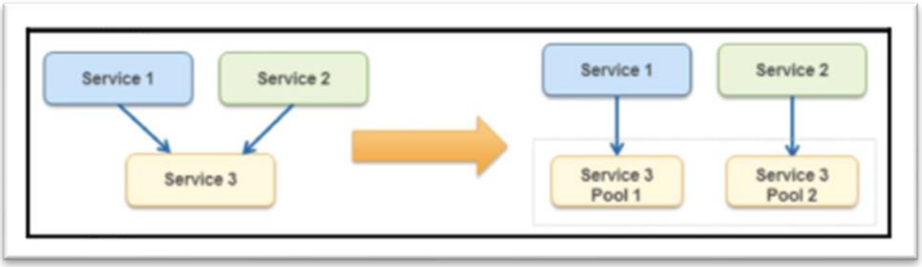
Understanding the circuit breaker pattern

In a distributed system, calls to downstream services may fail or hang, and retries are commonly used. However, remote services can take minutes or hours to recover, which can lead to longer wait times and potentially cause a cascading failure if the retry function consumes all available threads. The circuit breaker pattern is a solution that monitors the health of downstream dependencies and gracefully fails requests until the dependency becomes healthy again.

This can be achieved by implementing a persistence layer to track healthy and unhealthy requests over a defined interval. If a certain percentage of requests are unhealthy, the circuit is marked as open, and all requests throw exceptions for a defined timeout period. After the timeout period, a small percentage of requests try to integrate with the downstream dependency to detect if health has returned. Once a sufficient percentage of requests are healthy again, the circuit closes, and all requests can integrate with the dependency as usual. The implementation involves a state machine to track and share healthy/unhealthy request counts, which can be stored in DynamoDB, Redis/Memcached, or another low-latency persistence store.

Implementing the bulkheads pattern

The bulkhead pattern, originally used in ships, can be applied to large systems to prevent a single failure from causing the entire system to fail.

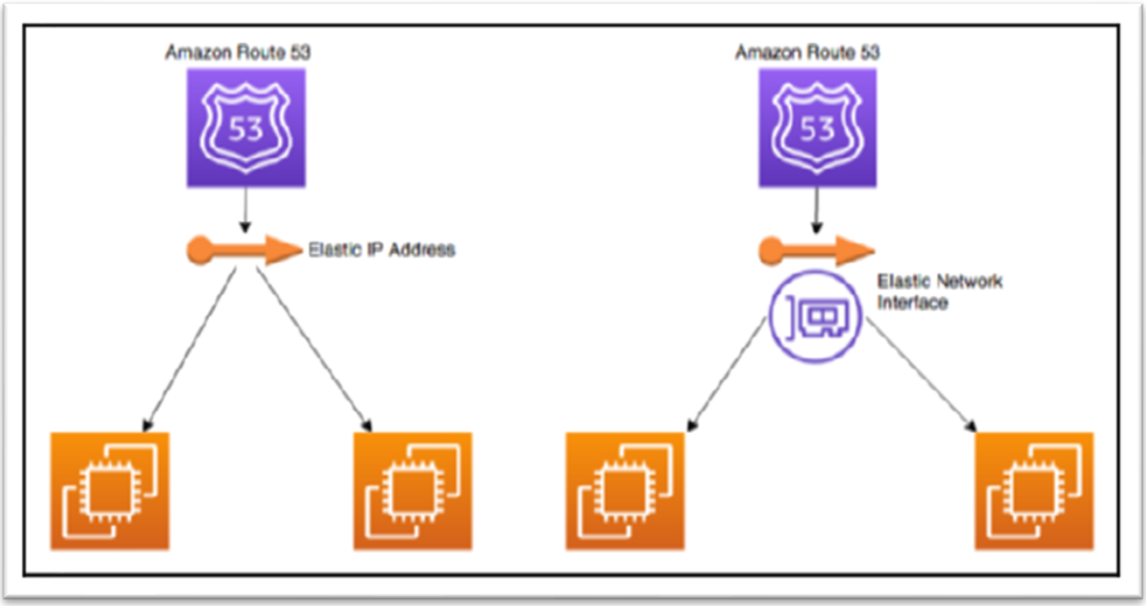


Bulkhead pattern

This pattern involves partitioning the system into separate compartments, each containing services with high dependencies so that if one service fails, it doesn't affect the other services or the entire system. To implement the bulkhead pattern, one should save part of the system, decide on a useful granularity, monitor each service partition's performance, and define a service partition for each business or technical requirement. The pattern can be used to isolate critical consumers from standard consumers and prevent cascading failures. The use of a floating IP can be an effective solution when dealing with legacy application servers that have hardcoded IP addresses or DNS names, as it avoids the need to change the server address when upgrading or modernizing the system.

Creating a floating IP pattern

In monolithic applications, hardcoded parameters based on server DNS name and IP address create challenges when bringing up new servers or performing upgrades.

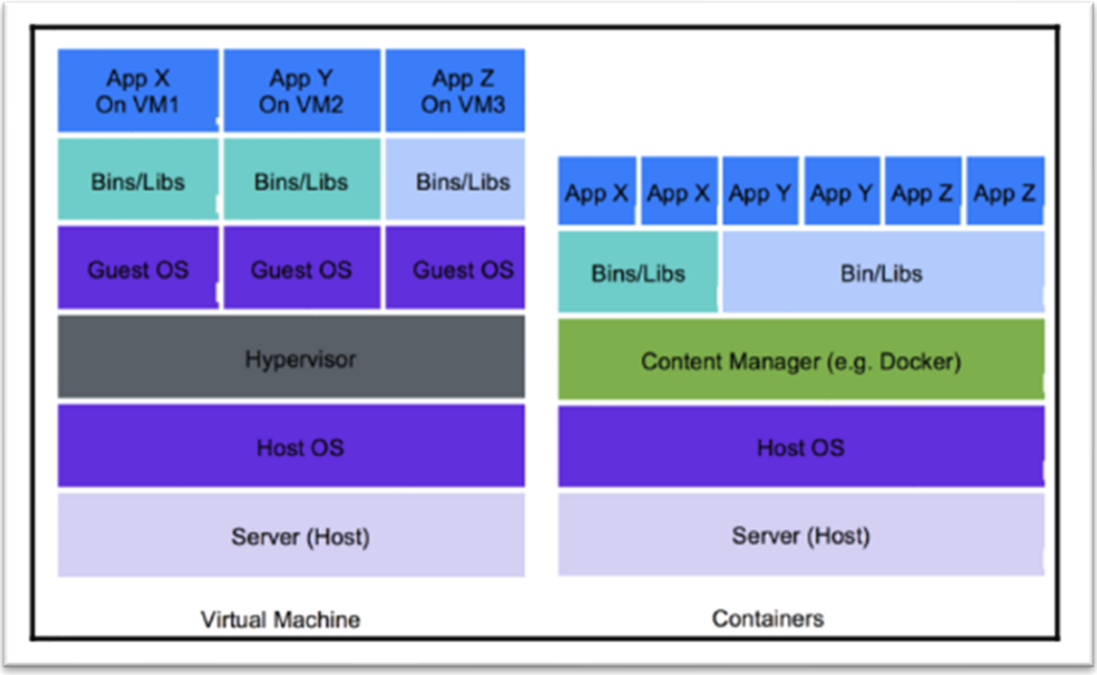


Floating IP and interface pattern

To overcome this issue, moving the network interface from the problematic instance to a new server with the same IP address and DNS name can be an effective solution. This approach can be facilitated through Elastic IP and Elastic Network Interface provided by public cloud services like AWS. Containers can be an alternative to virtual machines for optimizing resource utilization, especially for microservice deployment.

Deploying an application with a container

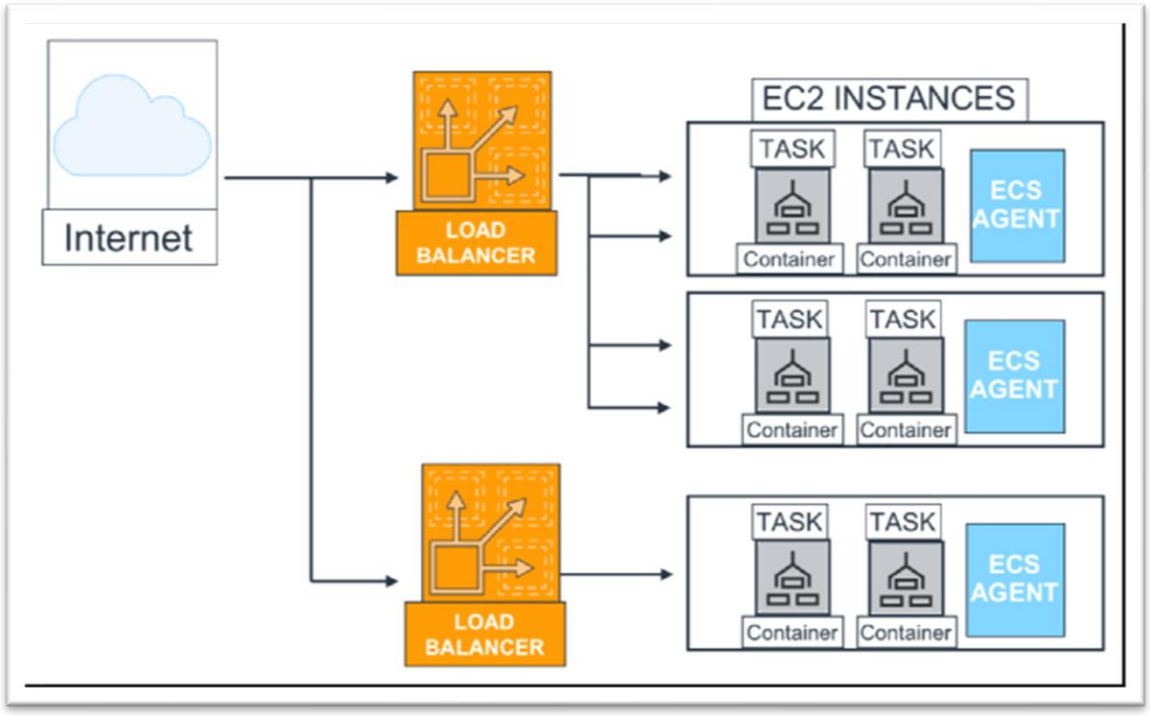
The constant evolution of programming languages and technologies brings new challenges. With different application stacks requiring various hardware and software deployment environments, there is a need to run applications across different platforms and migrate from one platform to another.



Virtual machines and containers for application deployment

Containers have become a popular solution to this problem. Like how shipping containers standardized the transport of freight goods, software containers standardize the transport of applications. Containers create an isolated environment for applications, enabling them to run on a single-host operating system with their file system, storage, RAM, libraries, and their own view of the system. Unlike virtual machines that isolate at the operating system level, containers isolate at the kernel level. The benefits of containers include a portable runtime application environment, faster development and deployment cycles, better resource utilization, the ability to run different application versions, easy management of security aspects, and the ability to package dependencies and applications in a single artifact.

Container deployment is useful for complex applications with multiple microservices as it allows for quicker deployment due to the consistency of the environment. Docker and Kubernetes are examples of container orchestration services that automate container provisioning and handle security, networking, and scaling aspects.

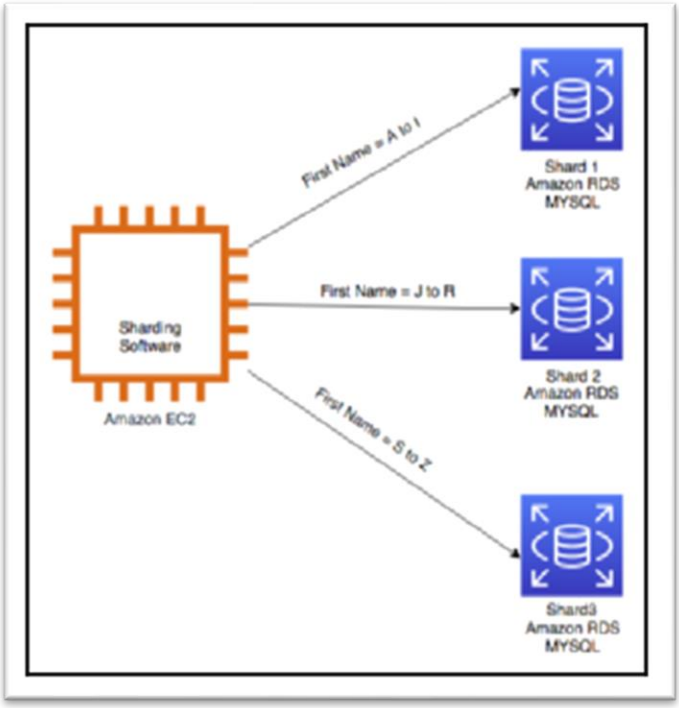


Container deployment architecture

AWS provides Amazon Elastic Container Service (ECS) and Amazon Elastic Kubernetes Service (EKS) to manage containers. As a solution architect, familiarity with all available container options is essential.

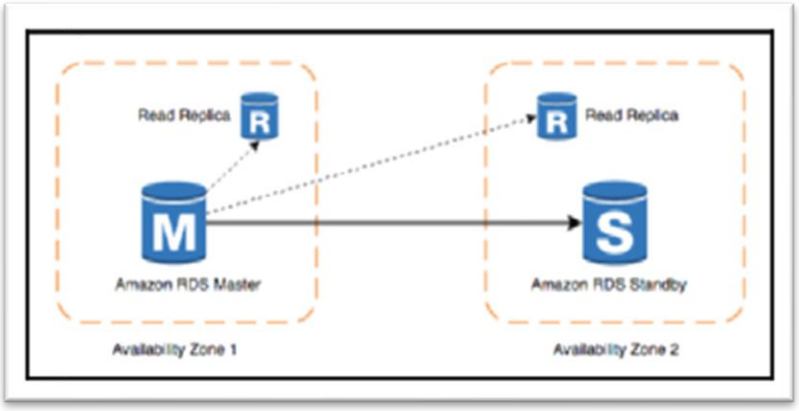
Database handling in application architecture

The importance of data in application development cannot be overstated, and efficiently handling data is crucial to improving application latency and performance. Caching frequently queried data by placing a cache in front of the database can be accomplished by using either a Memcached or Redis cache.



Relational database sharding

Scaling relational databases can be achieved through techniques such as read replicas and sharding. Read replicas can handle read requests while the master database node serves to write and update requests. Sharding allows for multi-master relational databases and distributes writing processes by dividing the database into identical structures.



High-availability database pattern

High database availability can be achieved through a standby replica of the master database instance located in a different availability zone. Backup and archival strategies should also be considered for disaster recovery. Data encryption strategies should be defined for customer-sensitive information. Migrating to a NoSQL database can be considered for greater scalability, management, performance, and reliability. Lastly, analyzing datasets such as clickstream data, application log data, and social media data can provide insight into organizational growth.

Avoiding anti-patterns in solution architecture

Teams often deviate from best practices due to time constraints or resource unavailability. It is essential to give special attention to architectural design anti-patterns, such as those listed below:

- Reactive and manual scaling
- Lack of automation
- Hardcoded IP addresses
- Monolithic application design
- Server-bound application
- Single type of database
- Single point of failure
- No caching of static content
- Insufficient security policy

These anti-patterns can be addressed by implementing best practices, such as:

- Automated scaling
- Identical server configurations
- Load balancers for independent applications and web layers
- SOA for low latency-distributed storage
- The right storage for the right need
- Secondary server for single points of failure
- Use of CDN for caching static content
- Application of the principle of least privilege.

Performance Considerations

Every second of delay in application load time can result in a significant loss of revenue. Therefore, optimizing an application's performance is a crucial aspect of solution design.

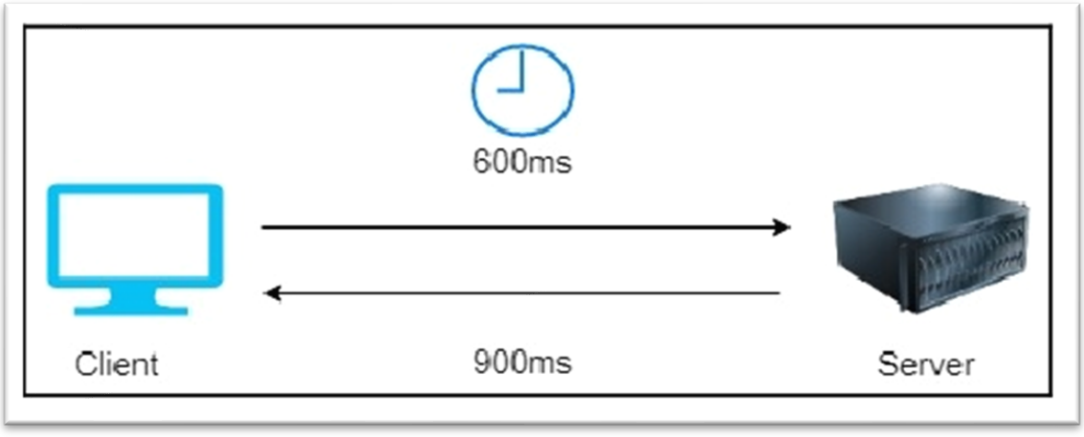
In this chapter, you will learn about the best practices for optimizing an application's performance. This will include design principles for performance optimization, technology selection, and performance monitoring. You will gain an understanding of important performance improvement attributes such as latency, throughput, and concurrency. By the end of the chapter, you will be able to make informed decisions about technology choices to improve performance at various layers of the architecture, including computing, storage, database, and networking.

Design principles for architecture performance

Efficient use of application infrastructure and resources to meet increasing demand and technology evaluation is the primary focus of architectural performance efficiency. Although many large enterprises continue to work on legacy programming languages and technologies due to fear of taking risks, advancements in technology often address critical performance issues and improve application performance. Public cloud providers, such as AWS, Microsoft Azure, and GCP, offer the technology as a service, which allows for the adoption of complex technologies more efficiently and with minimal effort.

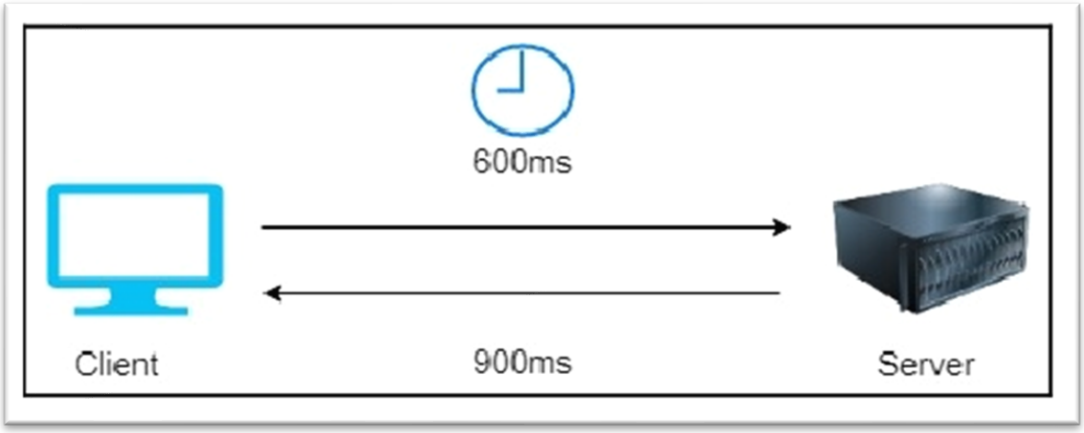
Content distribution networks (CDN) can store heavy image and video data near user locations to reduce network latency and improve performance. With infrastructure as a service (IaaS), it becomes easier to deploy workloads closer to users, which optimizes application performance by reducing latency over the network. As servers virtualize, organizations can become more agile and experiment with applications, applying a high degree of automation to improve performance.

Latency can negatively impact user experience, which is why it's important to aim for reducing response times within the user tolerance limit.



Request response latency in a client-server model

Latency is caused by factors such as network transmission medium, router hops, network propagation, and architecture components. Infrastructure, database, and application-level issues can all cause latency, and there are several strategies to reduce latency, such as optimizing memory and processor usage, using fiber optics lines, partitioning and sharding data, and handling transaction processing with garbage collection and multithreading. Throughput and latency are directly related, so improving latency can lead to higher throughput.

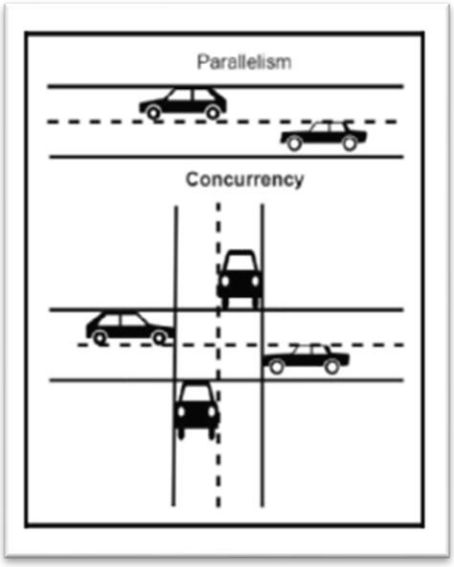


Throughput in a network

Throughput and latency are interrelated, as low latency allows for high throughput. Bandwidth, or the maximum amount of data that can be transferred over the network, plays a crucial role in determining throughput.

Disk throughput is determined by IOPS and the amount of data requested, while disk latency is determined by the time taken to receive a response from the disk. Throughput is also applicable at the CPU, memory, database, and application levels. Efficient use of memory cache and garbage collection handling is important for improving application-level throughput. Concurrency is another factor that can improve application performance, and the article suggests exploring this concept further.

Concurrency is an important factor in application design as it allows applications to perform multiple tasks at the same time. However, it is often confused with parallelism, which is different. In parallelism, an application divides a large task into smaller subtasks and processes them in parallel with a dedicated resource for each subtask. On the other hand, in concurrency, an application processes multiple tasks at the same time by utilizing shared resources among threads. The application can switch from one task to another during processing, which requires managing the critical section of code using locks and semaphores.



Concurrency versus parallelism

Concurrency is essential not only at the application level but also at the network and memory levels. In the case of database concurrency, it is essential to ensure that data is fully committed before another user tries to update it. Caching can significantly improve performance.

To handle concurrency, it is important to understand the differences between concurrency and parallelism. Concurrency enables applications to perform multiple tasks simultaneously by utilizing shared resources among threads, while parallelism processes multiple subtasks in parallel.

At the network level, it is important to handle multiple network connections and process active requests. At the memory level, shared memory concurrency enables concurrent modules to interact with each other. Database concurrency is more complicated, and it is important to ensure that data is fully committed before another user tries to update it. Caching can improve performance, and different types of cache exist in architecture.

While external caching engines and technologies like content distribution networks (CDNs) can be added to improve caching, it is important to understand that every application component and infrastructure has its own cache mechanism. Utilizing the built-in caching mechanism at each layer of the architecture can help reduce latency and improve performance.

Caching occurs at various levels of the architecture, including the server level (where the CPU has its own hardware cache and the disk has a page cache), the database level (where an internal cache and query cache store frequently used data), and the network level (where a DNS cache stores website domain names and corresponding IP addresses).

Design factors such as latency, throughput, concurrency, and caching need to be addressed for architecture performance optimization. The specifics of performance optimization will differ based on the application, and solution architecture needs to direct the effort accordingly. The selection of technology for various architecture levels can help overcome performance challenges.

Technology selection for performance optimization

After finalizing the strategy and starting the solution implementation, the next step is to optimize the application by collecting data through load testing and defining benchmarking. Performance optimization is an ongoing process that requires optimal resource utilization, which should be considered from the beginning of solution design to after the application's launch. Choosing the right resources as per the workload or tweaking the application and infrastructure configuration is essential for optimizing performance. For instance, selecting a NoSQL database to store the session state for an application and storing transactions in the relational database is a great way to offload the production database.

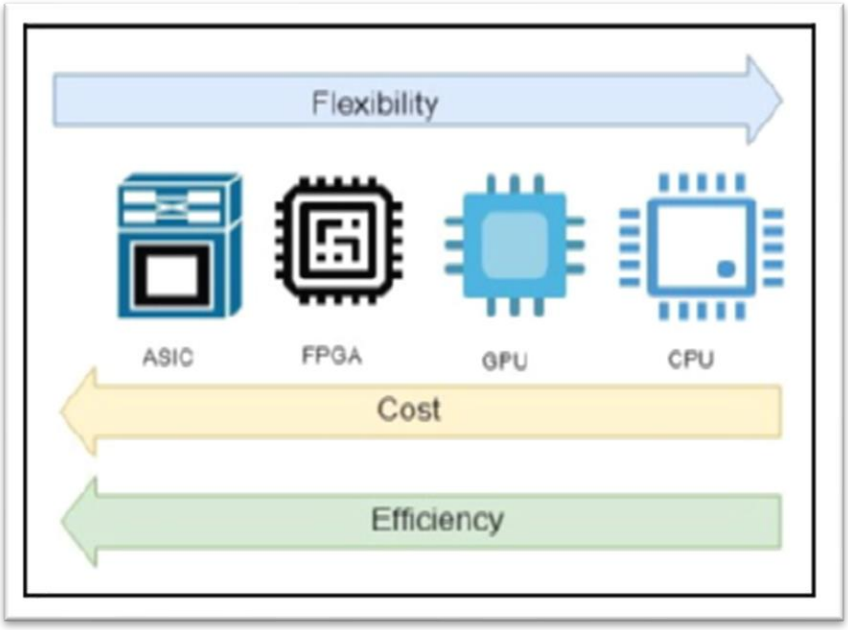
Additionally, loading data from the application database to data warehousing solutions and creating reports from there is a great way to optimize analytics and reporting. When selecting resources, you should consider computing, storage, database, and network types to optimize performance, and you may want to choose a virtual machine, container, or serverless approach depending on your application's workload.

The term "compute" is used instead of "server" to reflect the fact that software deployments are not limited to servers these days. Public cloud providers like AWS offer serverless options such as AWS Lambda, while Microsoft Azure has Azure Functions and Google Cloud Platform (GCP) has Google Cloud Functions. However, organizations still tend to choose servers with virtual machines by default, although containers are becoming increasingly popular due to the need for automation and resource utilization.

Containers are especially popular for microservice application deployment. The best choice of computing depends on the application's use cases, whether it's server instances, containers, or serverless. This section will explore the various computing choices available.

Cloud providers offer virtual servers that can be provisioned through a web console or API call, making infrastructure automation possible. As workloads vary, different processing unit choices are available.

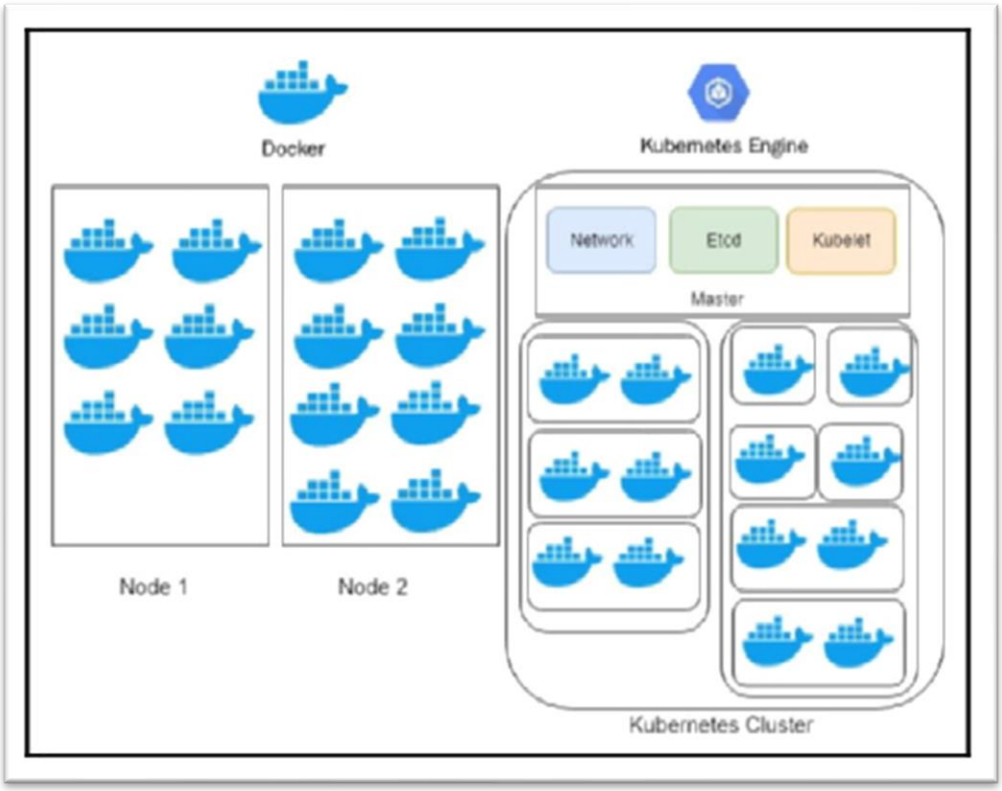
The most popular options are the CPU, GPU, FPGA, and ASIC. CPUs are versatile and cost-effective but may not perform well for parallel processing. GPUs are designed for massively parallel processing, making them suitable for machine learning and other compute-intensive applications, but they consume lots of power. FPGAs are programmable hardware that consumes less power but is less flexible and requires a longer development cycle. ASICs are custom integrated circuit optimizations that provide the most optimal performance but are costly and have the least flexibility to reutilize. CPUs are the cheapest, and ASICs are the costliest.



Comparison between CPUs, GPUs, FPGAs, and ASICs

In addition to CPUs, cloud providers offer instances with GPUs and FPGAs. Other popular computing types include accelerated processing units (APUs) and containers, which optimize the use of resources within virtual machines.

The use of containers has become popular for deploying complex microservices applications due to their ease of automation and efficient resource utilization. Docker is a widely used container technology that allows you to package an application and its dependencies together and deploy it to any operating system platform. Docker containers are portable and can be managed and distributed using a Docker Hub container repository, making overall troubleshooting easier.



Docker and Kubernetes

Kubernetes is a container orchestration system that manages and controls multiple containers in production environments, making applications self-healing, providing horizontal scaling capabilities, and avoiding downtime. Kubernetes and Docker work together to orchestrate software applications, with Docker functioning as an individual piece of the application and Kubernetes taking care of the orchestration. While Kubernetes is complex to learn, public cloud providers offer services to simplify its management. If ultra-low latency is required, bare metal physical machines may be a better choice for application deployment.

Serverless computing has become popular in recent years due to the availability of public cloud offerings from providers such as Amazon, Google, and Microsoft.

With serverless computing, developers can focus on code and application development without worrying about infrastructure management. The concept of Function as a Service (FaaS) is relatively new and available through services such as AWS Lambda, Microsoft Azure Function, and Google Cloud Function. These services allow developers to design event-based architectures or RESTful microservices by adding an API endpoint using Amazon API Gateway and AWS Lambda functions. Serverless computing enables developers to scale their code automatically and independently without paying for idle resources. However, predicting autoscaling costs can be tricky, especially with thousands of features to orchestrate. Organizations can choose their computing

services based on their application's requirements, technology choices, innovation pace, and application nature. For monolithic applications, virtual or bare-metal machines are suitable, while containers are suitable for complex microservices. For simple task scheduling or event-based applications, the function is the obvious choice. Many organizations have built entirely serverless applications, saving costs and achieving high availability without managing any infrastructure.

The performance of an application heavily relies on storage, which is required for various functions such as installation, logging, and file access. Choosing the right storage solution depends on several factors including data format and scalability needs. The choice between block, file, or object storage depends on how data is stored and presented.

Access methods	Block, file, or object
Access patterns	Sequential or random
Access frequency	Online (hot), offline (warm), or archival (cold)
Update frequency	Write once read many (WORM) or dynamic
Access availability	Availability of storage when required
Access durability	Reliability of data store to minimize any data loss
Access throughput	Input/output per second (IOPS) and data read/write per second in MBs.

Each storage format has its own unique features and benefits. The following section will discuss each storage format in more detail to help you choose the optimal storage solution for your application.

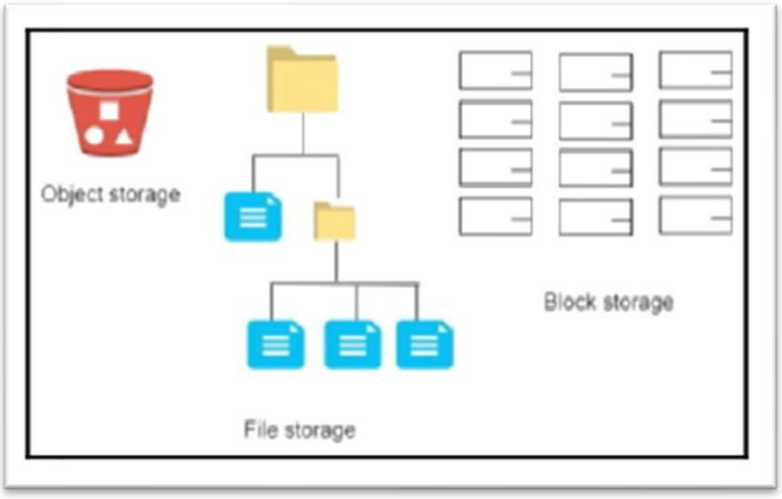
In block storage, data is divided into blocks and stored as chunks, each with a unique ID for faster access. Metadata about files is managed by the server-based operating system. Block storage can be deployed in a storage area network (SAN) for efficient and reliable data storage, making it suitable for database deployment, email servers, application deployment, and virtual machines. SAN storage is sophisticated, high-performance, and expensive, making it ideal for large-scale enterprise applications with low latency requirements.

To configure block-based storage, you need to choose between solid-state drives (SSDs) and hard disk drives (HDDs). HDDs are cheaper but slower, while SSDs are faster, more efficient, and require less power and cooling, but are more expensive. As technology evolves, SSDs will become more affordable and popular.

File storage is a traditional storage method where data is stored as a single piece of information inside folders. Each record has limited metadata, including the filename, creation time, and updated timestamps. On the other hand, NAS storage is a file storage system that is attached to the network and displays where users can store and access their files. NAS storage also manages user privilege, file locking, and other security mechanisms that protect the data.

NAS works well as file-sharing systems and local archives, but it may not be the right solution for storing billions of files due to its limited metadata information and complex folder hierarchy. For that, you need to use object storage, which offers several benefits over file storage.

In data storage, file storage organizes data inside folders with limited metadata. Network area storage (NAS) is attached to the network and manages user privilege, file locking, and security mechanisms. Object storage, on the other hand, bundles data with a unique identifier and customizable metadata using a flat address space, making it easier to locate data and retrieve it faster, making it ideal for high-volume, unstructured data. Cloud data storage, such as Amazon S3, provides unlimited scalable object data storage with high availability and durability. Block storage has low latency and is suitable for single-instance access, while file storage has little latency overhead and is good for multiple instances. Direct-attached storage has limited scalability, while magnetic tape drives are good for archival purposes.



Data storage systems

A redundant array of independent disks (RAID) configuration protects against drive failure and increases disk throughput. Selecting the right storage method depends on the access pattern, and public cloud offerings provide various options.

Choosing the right database is critical for application performance and latency.

When selecting a database, it's important to consider factors such as availability, scalability, data structure, throughput, and durability.

The database technology you choose should be optimized based on the access pattern of your application. Additionally, databases typically offer configuration options for workload optimization, including memory, cache, and storage optimization. It's also essential to consider the operational aspect of database technologies, such as scalability, backup, recovery, and maintenance. In the next section, we'll explore different database technologies that can meet the requirements of various applications.

The concept of Online Transactional Processing (OLTP) is a common method of handling application data using traditional relational databases. OLTP databases are ideal for complex business transactions that require aggregating data and creating complex queries using multiple joins across tables. To optimize your relational database, consider tuning settings such as the database server, operating system-level settings, database engine configuration, and partition. However, scaling can be challenging for relational databases, as they have a limit to the system capacity. Horizontal scaling can be achieved through replica for read scaling and partition for write scaling. OLTP databases do not handle unstructured data efficiently, so NoSQL databases are used in such cases.

Nonrelational databases, also known as NoSQL, are useful for storing unstructured or semistructured data that does not fit well into the rigid structure of a relational database. NoSQL databases provide more flexibility, allowing each record to have a varying number of columns and enabling easy scaling by adding more nodes. NoSQL databases are commonly used for storing user session data and can help make an application stateless to achieve horizontal scaling without sacrificing user experience. However, they do not support complex queries such as joining tables and entities. Some popular NoSQL databases include Cassandra, HBase, MongoDB, and Amazon DynamoDB. While OLTP databases like relational databases can be used for transactional applications, they have limited storage capacity and cannot handle queries for large amounts of data or those that perform aggregations, which are required for data warehouses.

OLTP and NoSQL databases are ideal for application deployment, but they fall short when it comes to analyzing large volumes of structured data. To analyze data quickly, you need a data warehouse platform that uses a columnar format and massive parallel processing (MPP). The columnar format allows you to scan only the required columns, which improves query performance by reducing the amount of data scanned. Massive parallel processing stores data in a distributed manner between child nodes and performs parallel processing, which helps execute queries faster and process a large amount of data quicker.

You can install software such as IBM Netezza or Microsoft SQL server on a virtual machine or use a cloud-native solution like Snowflake or Amazon Redshift to achieve this kind of processing. Data Engineering and Machine Learning provide further insights into data processing and analytics. If you need to store and search a large amount of data, such as logs or documents, you need a data search capability.

To quickly search through large amounts of data for business insights or issue resolution, you will need to utilize search engine technology. Elasticsearch, built on the Apache Lucene library, is a popular search engine platform that can index and analyze data for fast searching.

With its easy-to-use ELK stack, Elasticsearch allows for the automatic discovery of data and has many visualization and analysis tools available. It can be deployed on virtual machines and scaled horizontally to increase capacity by adding new nodes to the cluster. AWS provides the managed service Amazon Elasticsearch Service for cost-effective, cloud-based management of Elasticsearch clusters. Combining OLTP, NoSQL, and OLAP databases can optimize performance for different components of your application. Networking is a crucial component in establishing communication between servers and the outside world, and its role in application performance will be discussed further.

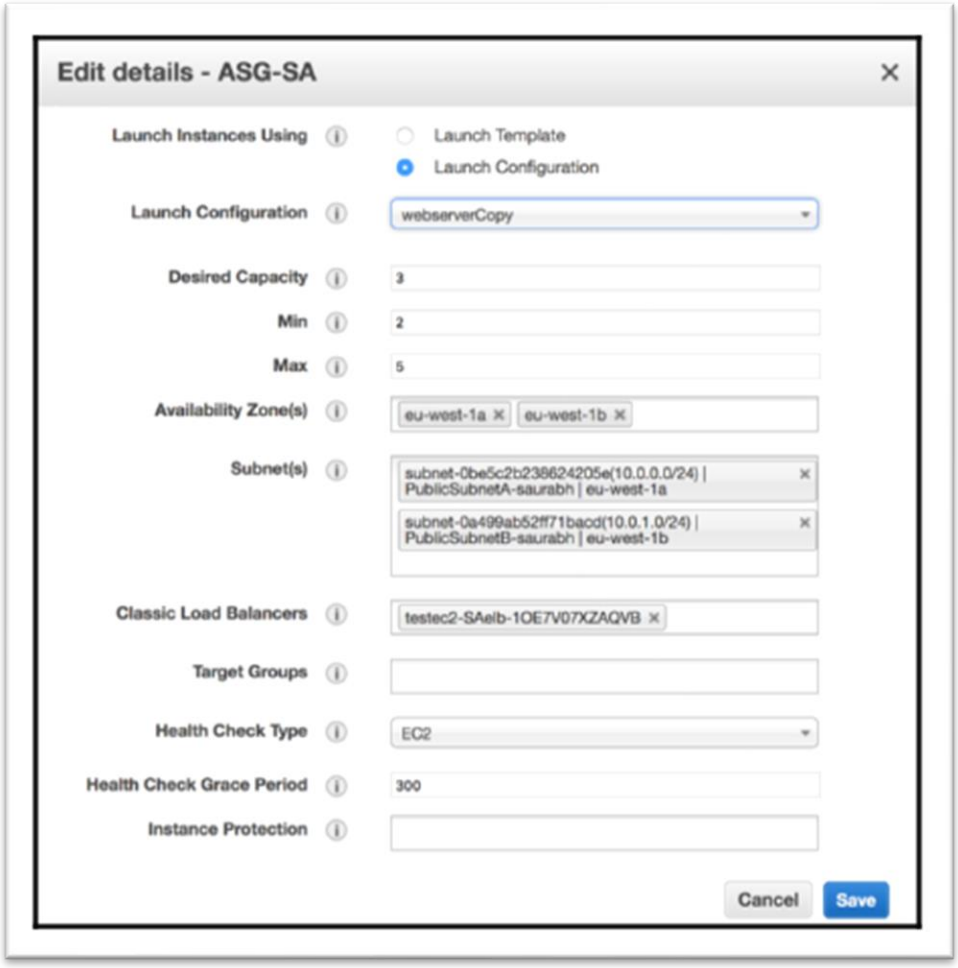
To ensure that an application can respond to user requests in a timely manner, it is important to reduce latency. This can be achieved by simulating the user's location and environment to identify any gaps and designing the server's physical location and caching mechanism accordingly. The choice of networking solution for an application depends on the networking speed, throughput, and network latency requirements. To handle a global user base, it is necessary to have fast connectivity with customers, and edge locations provided by a CDN can help to localize heavy content and reduce overall latency.

Using a CDN is an effective solution for applications that are static-content-heavy and need to deliver large image and video content to end users. Popular CDN solutions include Akamai, Cloudflare, and Amazon CloudFront. If an application is deployed globally, DNS routing strategies can be employed to achieve low latency.

To ensure a quick response from your application, you need to route user requests to the nearest and fastest available server. A DNS router provides the mapping between domain names and IP addresses, and the public cloud-like AWS offers Amazon Route 53, which provides various routing policies to meet your application's needs. The most used routing policies are simple, failover, geolocation, Geo Proximity, latency, and weighted routing. Amazon Route 53 can also detect anomalies in the source and volume of DNS queries, prioritize requests from reliable users, and protect against DDoS attacks. Once traffic passes through the DNS server, it typically reaches a load balancer, which distributes traffic among a cluster of servers.

A load balancer is a device or software that distributes network traffic across multiple servers to improve concurrency, reliability, and application latency.

There are two types of load balancers: layer 4 or network load balancer and layer 7 or application load balancer. The network load balancer routes packets based on information in the packet header, while the application load balancer inspects packets based on the full contents of the packet. Depending on the environment, you can choose a hardware-based or software-based load balancer. Amazon Elastic Load Balancer is a managed virtual load balancer that can be applied at layer 7 or tier 4. The load balancer is used in conjunction with autoscaling to add or remove instances as required, thereby improving overall performance and high availability.



Autoscaling configuration

Autoscaling is a cloud computing feature that enables you to scale up or down your server fleet automatically based on user or resource demand. You can apply autoscaling at every layer of your architecture on public cloud platforms like AWS. Autoscaling enables you to scale your web server fleet based on requests, server memory, CPU utilization, and traffic patterns.

Managing performance monitoring

To proactively understand and reduce performance issues that may impact end-users, it is important to establish a performance baseline and set alarms to notify the team of any threshold breaches. This can be done using a variety of monitoring tools, such as third-party solutions like Splunk or AWS's Amazon CloudWatch.

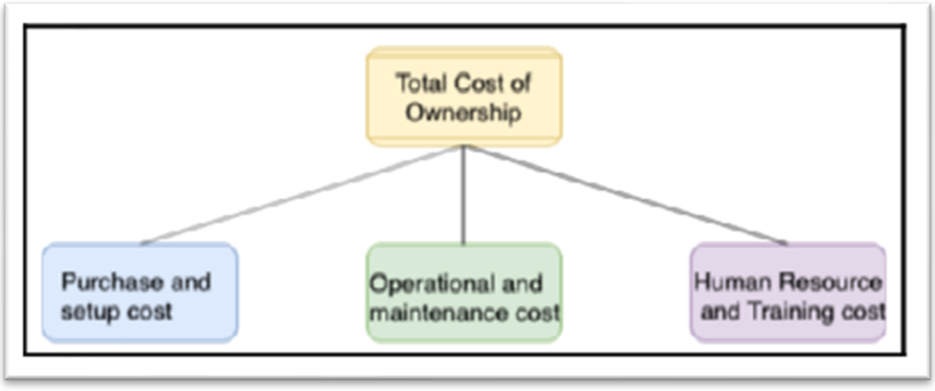
There are two types of monitoring solutions: active and passive. Active monitoring involves simulating user activity to identify performance gaps in advance, while passive monitoring tries to identify unknown patterns in real time. To balance performance needs with cost considerations, architects must determine the trade-offs between durability, consistency, cost, and performance for each application. Improving performance requires collecting and analyzing data, including access patterns and load testing. Consistent performance can be maintained by applying continuous active monitoring in combination with passive monitoring.

Cost Considerations

Cost optimization should not sacrifice customer experience but rather maximize return on investment. Understanding your customer needs is important when planning cost optimization. The chapter covers design principles, techniques, and cost optimization in the public cloud. By the end of the chapter, you will have learned various methods to optimize costs without risking business agility and outcome, as well as monitoring and applying governance for cost control.

Design principles for cost optimization

Cost optimization involves minimizing risk and increasing business value while reducing business costs. Planning for cost optimization involves estimating budgets and forecasting expenditures, implementing a cost-saving plan, and closely monitoring expenditures.



TCO for software

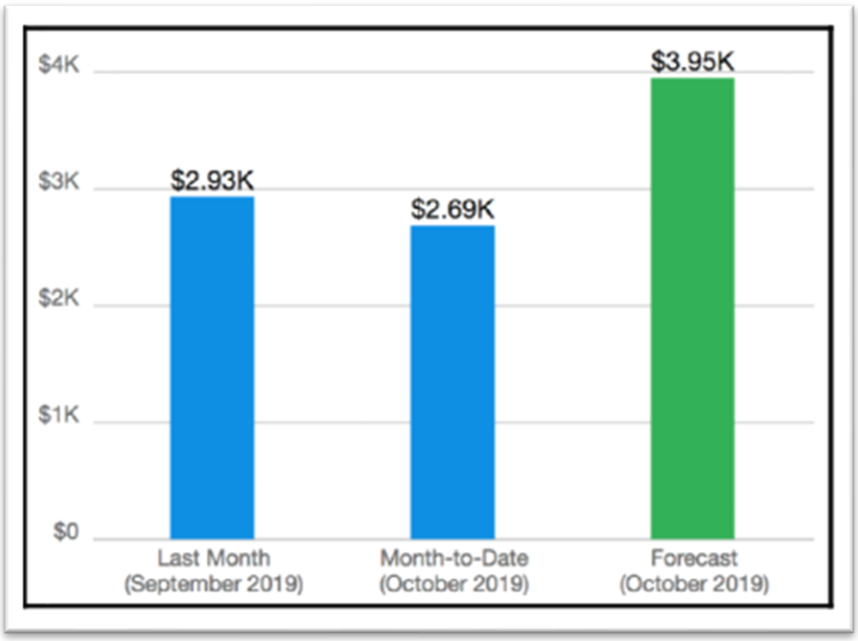
Several design principles can help achieve cost optimization. One of these is calculating the total cost of ownership (TCO), which covers both capital expenditure (CapEx) and operational expenditure (OpEx) involved in the application life cycle. To make more strategic decisions while calculating the return on investment (ROI) in the long run, you should consider all associated costs. For example, when purchasing a refrigerator that runs 24/7, you may opt for a higher-priced energy-saving model, knowing that the total cost over time will be lower due to savings in electricity bills. Similarly, when purchasing software, you should consider the TCO and include costs such as software and hardware acquisition, operational and maintenance costs, and human resource and training costs. When looking for a software solution, you can choose between a subscription-based Software as a Service (SaaS) model, Infrastructure as a Service (IaaS) option, or building the software in-house. In any case, calculating the TCO can help you make a decision that maximizes ROI. Finally, budget and forecast planning can help control TCO and achieve ROI.

The process of planning the budget and forecast is essential for businesses to control their costs and calculate ROI. Organizations create long-term budgets for 1-5 years, which are then broken down into individual project and application levels.

During solution design and development, the team needs to consider the available budget to plan accordingly. While the budget is important for long-term strategic planning, the forecast provides an estimate at a tactical level to decide the business direction. In application development and operation, losing track of the budget can result in cost overruns. The difference between a budget and a forecast is that the forecast helps you take immediate action, while the budget may become unachievable due to changes in the market.

Budget	Forecast
Represents future result and cash flow for business objectives that you want to achieve	Represents revenue and current situation of the business
Plans for the long term, for example 1-5 years	Plans month to month or quarterly
Is adjusted less frequently, maybe once in a year, based on business drivers	Is updated more regularly based on actual business progress
Helps to decide business directions such as organization restructuring based on actual cost versus budgeted cost	Helps to adjust short-term operational costs such as additional staffing
Helps to determine performance by comparing planned cost versus actual cost	Isn't used for performance variation but for streamlining progress

To manage demand and service catalogs, organizations can take either the demand-led or service-led approach. The demand-led approach analyzes historical data to understand factors driving demand, while the service-led approach identifies the most frequently used services and creates a catalog with a granular cost.



Billing and forecast report

Both approaches can result in significant cost savings in the short and long term. However, these transformations require changes in the project planning and approval process and alignment between the business and finance teams to understand the clear relationship between business growth and IT capacity. The cost model should combine offerings from the cloud, on-premises, and off-the-shelf to build the most efficient approach.

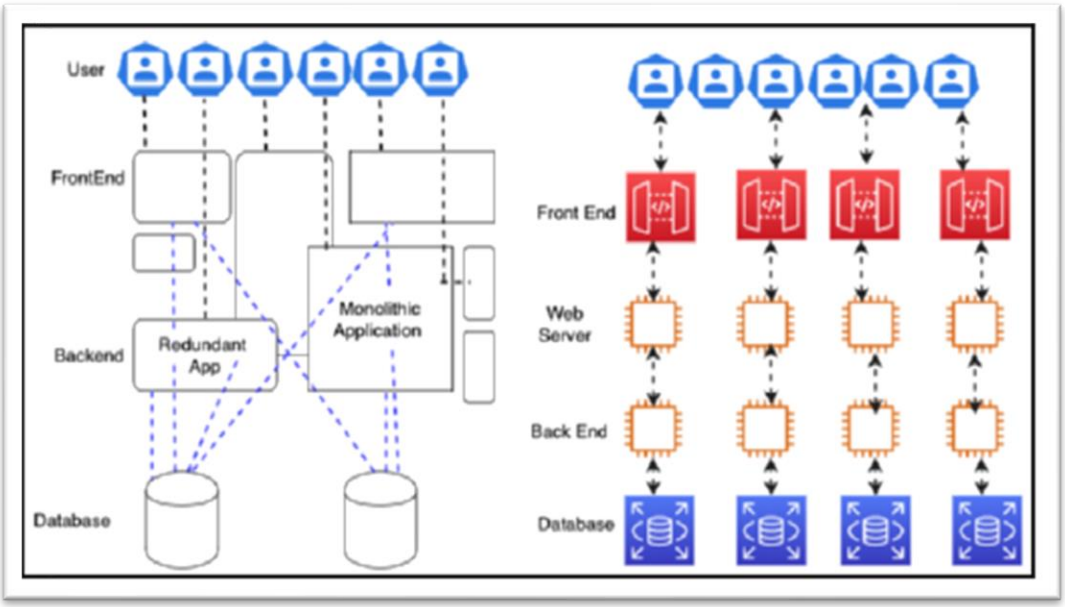
Tracking expenses allows for identifying ROI, rewarding cost-saving efforts, and determining the costs of specific departments or projects.

To hold everyone accountable for cost-saving, organizations can introduce show-back or charge-back mechanisms to share cost responsibility between units. The show-back approach informs each unit of its expenditure without charging the actual amount, while the charge-back mechanism charges each business unit for its IT resource consumption. It is recommended to start with show-back and move to charge-back as the organization matures. Cost tracking creates accountability and helps to understand team operations. To optimize costs, it is important to use the pricing model that suits the workload, establish mechanisms that ensure business objectives are achieved, define a tagging strategy, and apply the check-and-balance approach. Cost optimization should be continuous and never end until the cost of identifying money-saving opportunities is more than the amount of money saved. It is important to ensure that the cost paid for resources is well utilized and avoid underutilized IT resources. Application-level metrics for cost optimization need to be carefully considered, such as introducing archival policies for data storage capacity and checking for appropriate database deployment needs.

Implementing resource management and change control processes during the project life cycle can identify gaps and apply necessary changes for cost-saving. Organizations should continually look for new services and features that might directly reduce costs.

Techniques for cost optimization

Enterprises are investing more in technology to gain a competitive edge and keep up with rapid growth, but with economic instability, cost optimization becomes an essential but challenging task. Organizations often lack a centralized IT architecture, resulting in each business unit trying to build its own set of tools. This lack of overall control results in a lot of duplicate systems and data inconsistency. Improving existing architecture can open doors to bring more opportunities and business to the company, even if it requires a bit more adjustment in the budget. One area where companies can save costs and gain opportunities to bring more revenue is by reducing architectural complexity.



Architectural standardization

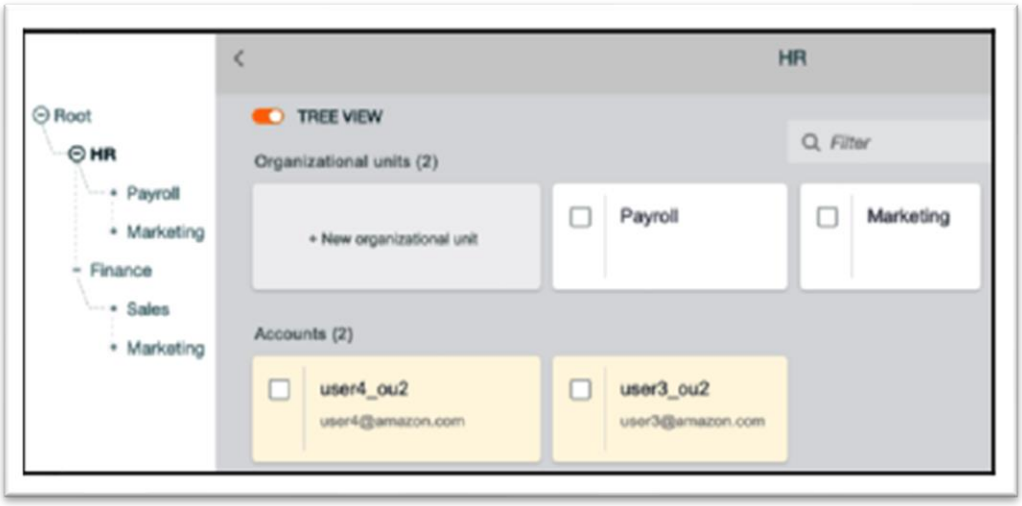
To reduce the complexity of IT architecture, organizations should think of an out-of-the-box solution that fits their business needs and provides an ROI. Customization should be the last approach if no other option is available. Any new application needs to have a more accessible integration mechanism to interact with the existing system using service-oriented architecture (SOA). Taking a modular approach can help to avoid duplication, where each team gets the responsibility of developing a service that every team across the organization can use. With the help of microservices architecture, you can deploy an entire application in a modular way, and if one component is not working, you can rework it without impacting the whole application.

Once a centralized IT architecture is set up, taking a modular approach can help to keep costs down. Empowering the IT architecture team can help align organizational units with the company's vision and support other parallel projects to follow the overall strategy. With the help of the IT architecture team, an architect can advise whether there is any duplicate effort, project, process, or system that is not aligned with the business need. The centralized architecture will reduce complexity and tech debt, bring more stability, and increase quality.

To increase IT efficiency and reduce costs, companies should optimize their use of IT resources. This can be done by tracking software licenses, retiring unused licenses, and negotiating bulk discounts with vendors. Non-compliant or low-value projects should be canceled or deprioritized, and unused applications should be decommissioned. Legacy systems should be modernized, and data should be consolidated. Cloud migration can also help reduce costs, as can the automation of routine tasks. However, cost optimization should be done carefully to avoid adding business risk.

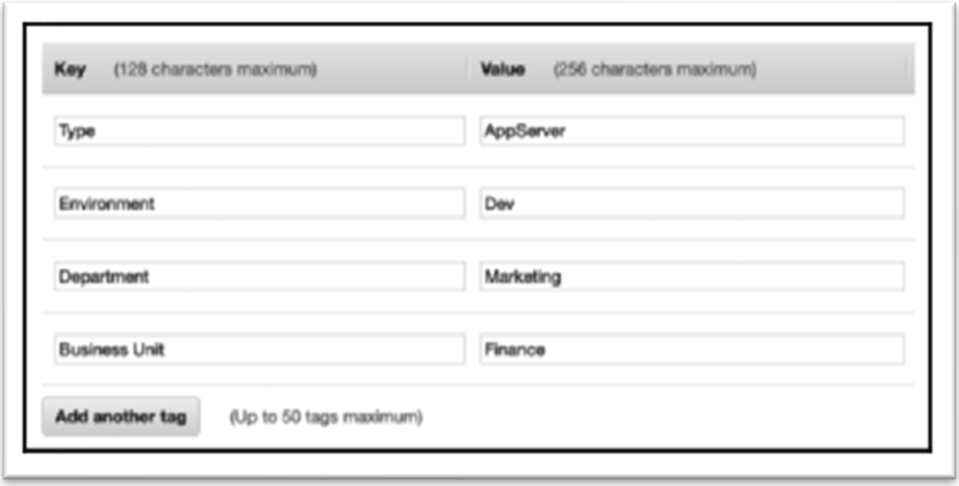
Goals should be set and measured to ensure continuous improvement. At the organizational level, goals should align with business units, while at the team level, they should align with individual systems.

To establish a standard infrastructure, reduce complexity, and eliminate duplicate projects, organizations should develop guidelines for appropriate and efficient system usage and implement governance.



Enterprise account structure for organizational units

This involves setting resource limits across the organization, using a service catalog with infrastructure as code, and considering both resource creation and decommission when defining the process to change them. To drive efficient usage behavior and reduce cost, organizations should determine resource costs to applications, business units, or teams and organize cost structures through resource tagging and account structuring.



Resource tagging for cost visibility

Adopting a charge-back mechanism for each department increases accountability for cost at a more granular level, while account structuring helps to apply high security and compliance standards across the organization. Resource tagging can be used not only to organize costs but also to define a capacity limit, security, compliance, and inventory management.

To create efficient IT architectures, organizations need to collaborate across functional teams and engage all impacted stakeholders in usage and cost discussions. Vendors should provide a cost analysis of any application they own and develop, and teams should be able to translate business, cost, or usage factors from management into system adjustments.

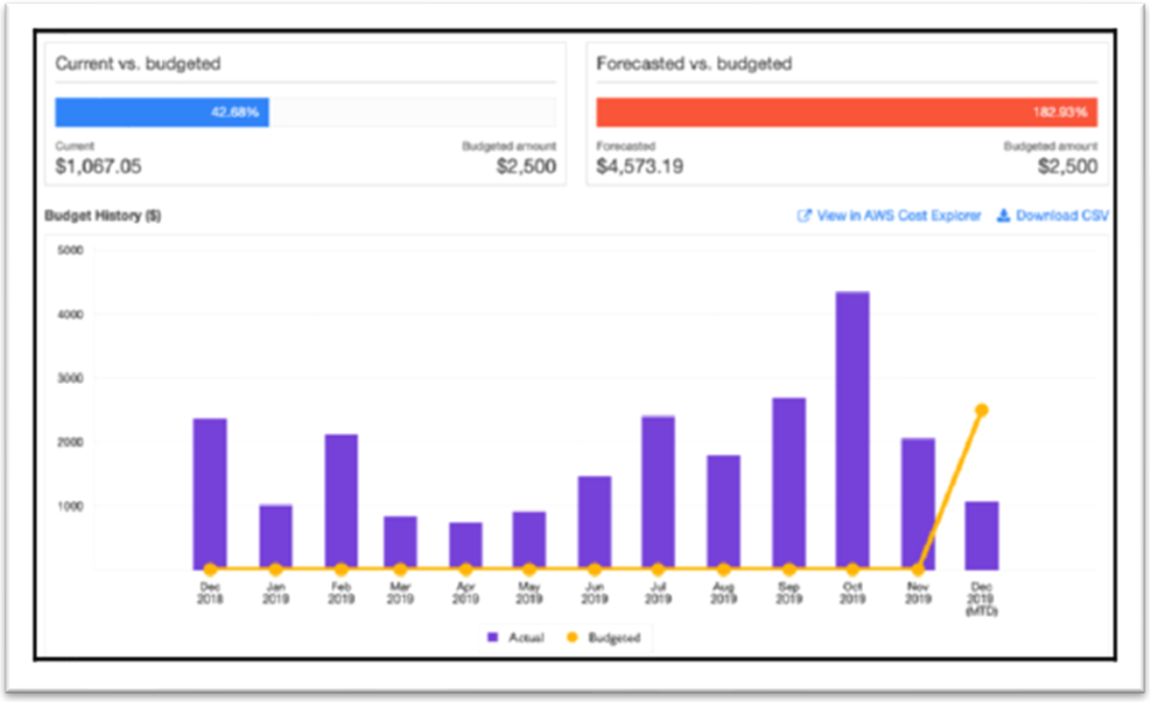
Effective cost tracking is essential for businesses to determine the profitability of their units and products, allocate resources appropriately, and control their expenses.

Resource cost and usage report

To optimize costs, organizations need visibility of their expenditure patterns and must analyze data points to identify cost-saving opportunities. Detailed insight into workload resource utilization is also necessary to forecast future spending accurately.

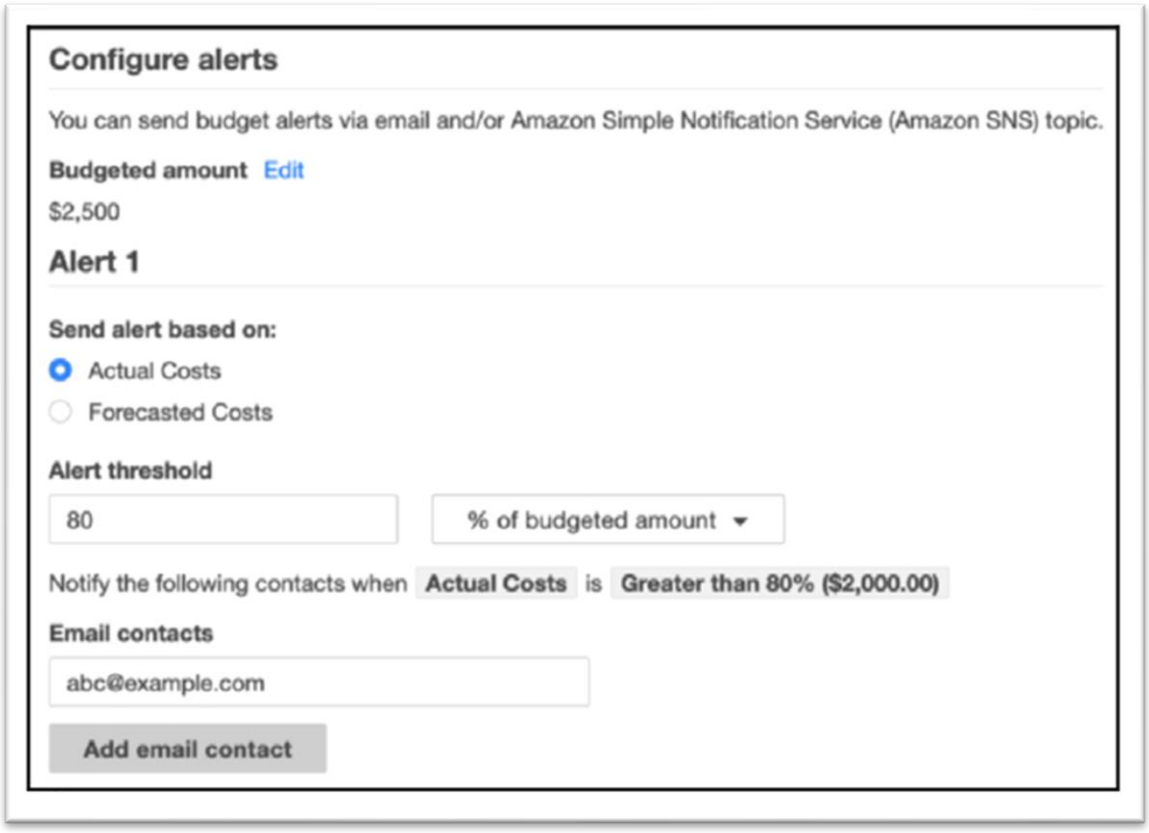
Cost trend and cost forecast report

Visualization reports can help business owners understand cost patterns and take proactive measures to control costs. It is important to be careful when resizing your environment to ensure that you are not impacting the customer experience.



Cost and budget report

Best practices include selecting the correct monitoring cycle and assessing the cost of change against the cost-saving. The right tool that covers all dimensions from cost-saving to system utilization and impact on customer experience due to changes should be used and reports should be utilized to understand business ROI impact due to cost changes.

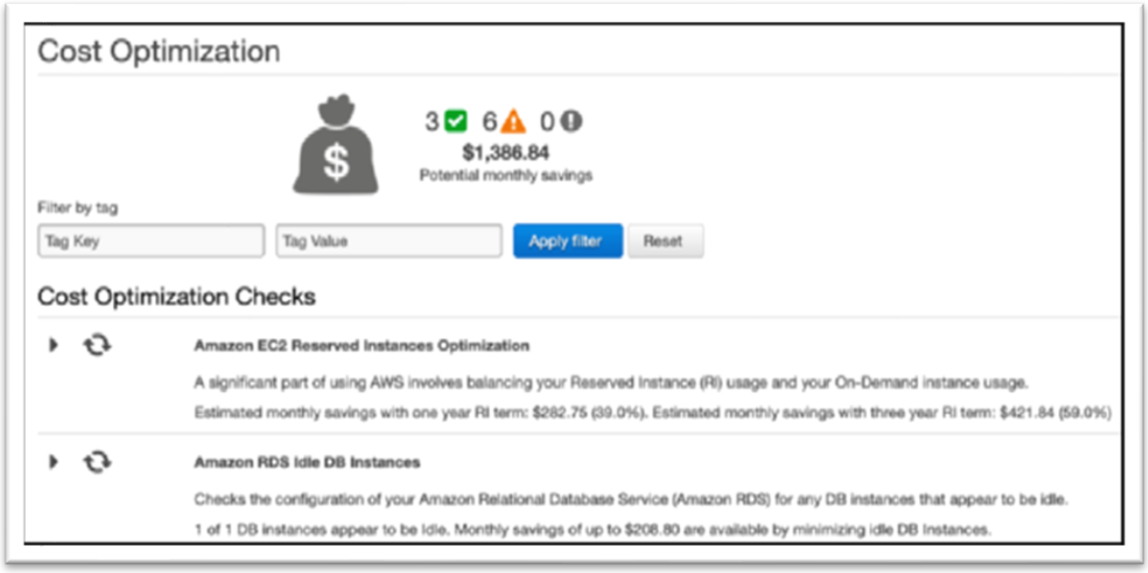


Alert against actual cost

Alerts should be set up when actual costs reach a certain threshold in the budget or forecast, and multiple alerts can be set up to get information when costs reach 50% or 80% of the budget or forecast.

Cost optimization in the public cloud

The public cloud provides cost optimization with the pay-as-you-go model, allowing customers to pay for IT resources as they consume them. The cloud cost model also offers additional tools and functionality, such as cost governance and regularization.



Cost-saving suggestions from AWS Trusted Advisor

AWS Trusted Advisor is an example of a tool that can provide cost-saving recommendations based on resource utilization. The cloud can provide an excellent value proposition for cost-saving, and a hybrid cloud can be established to determine cost structures and potential savings. Public cloud providers are increasingly offering managed services, which reduce infrastructure maintenance costs and overheads for alert and monitoring configurations.

DevOps and Solution Architecture Framework

In traditional environments, the development and IT operations teams work independently and often have conflicting processes and tools, leading to issues and delays in product delivery. DevOps is a methodology that promotes collaboration and coordination between teams to deliver products or services continuously, without compromising on quality, reliability, stability, resilience, or security. This chapter introduces the various components of DevOps, such as continuous integration/continuous delivery (CI/CD), continuous testing, DevSecOps, and implementing a CD strategy. By the end of the chapter, readers will understand the benefits of DevOps, learn about DevOps best practices, and different tools and techniques to implement them.

Introducing DevOps

The DevOps approach involves collaboration between the development and operations teams throughout the software development life cycle, with shared responsibilities and continuous feedback. This approach promotes early detection of defects and faster delivery of software applications, resulting in a shorter time to market, reliable releases, improved code quality, and better maintenance. The DevOps culture requires breaking down barriers between teams, evaluating tools for compatibility and sharing, and developing a range of skills for each function. DevOps is becoming the preferred culture for organizations dealing with cloud or distributed computing.

Understanding the benefits of DevOps

DevOps is a culture and process that helps teams collaborate and automate their development and delivery pipelines. Its goal is to achieve a repeatable, reliable, stable, resilient, and secure CD model.

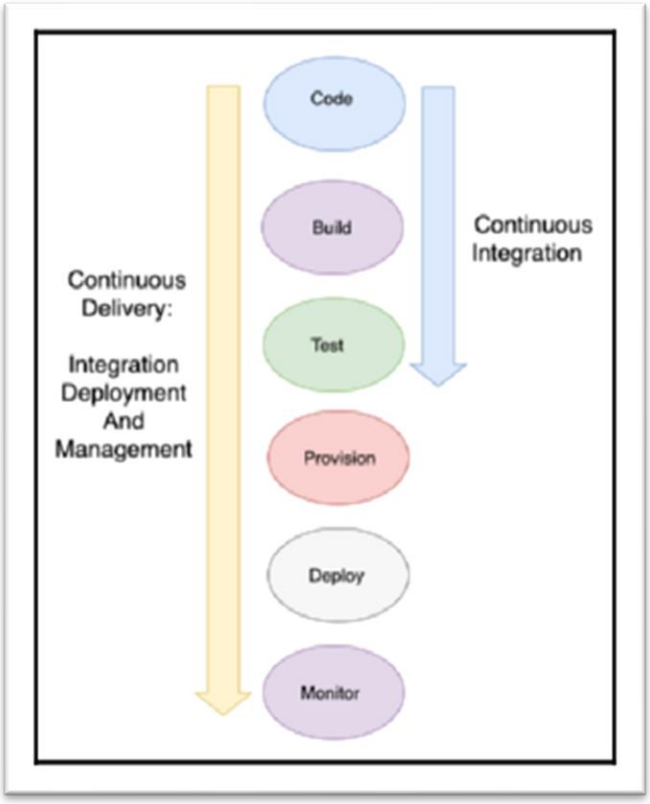
Benefits of DevOps

These benefits help organizations accommodate changing business needs, innovate faster, gain a competitive edge, ensure delivery quality, simplify the process, increase efficiency, and automate security and compliance best practices.

The DevOps model optimizes the productivity of the development team and the reliability of system operations by breaking down the silos between them. Teams take full ownership of the services they deliver, beyond their traditional roles, and develop thinking from a customer point of view to solve any issues.

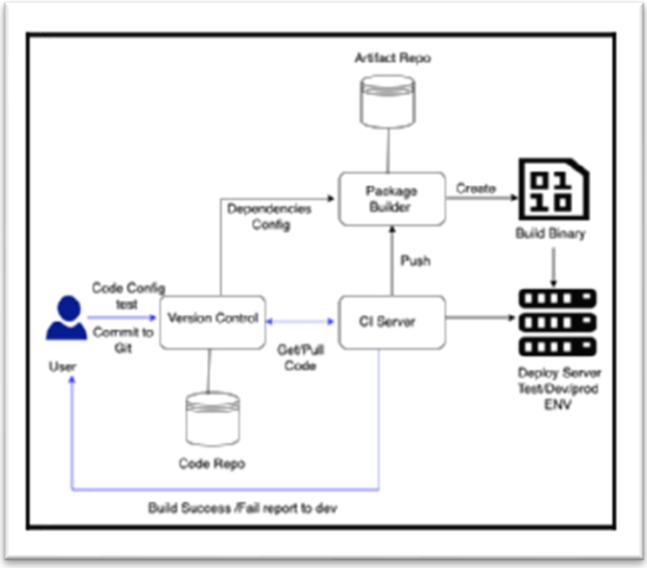
Understanding the components of DevOps

DevOps tools and automation are essential for merging development and system operations. To establish a successful DevOps practice, several critical components need to be considered. These include Continuous Integration/Continuous Delivery (CI/CD), Continuous Monitoring and Improvement, Infrastructure as Code (IaC), and Configuration Management (CM). Automation is a critical best practice for all these elements. It enables efficient, fast, reliable, and repeatable operations through various tools such as scripts and templates.



CI/CD

CI involves frequent code commits to a repository, automated unit, and integration testing, and builds. CD extends CI to deploy builds in production environments after successful automated and manual testing.



CI/CD for DevOps

Automation of the software release process is a crucial aspect of CI/CD. Code repositories maintain different versions of the code accessible to the team for concurrent development. The automation of the entire application flow, from code, commits to deployment and monitoring, is illustrated in a diagram.

IaC scripts automate the provisioning and deployment of infrastructure for testing and production environments. This ensures the quick detection and fixing of bugs and faster launch of feature updates in the production environment. CD ensures that every change is ready to go into production after successful staging and testing. The final deployment to a live production environment is still a business decision, but automation tools automate the deployment process.

Continuous monitoring and improvement are critical to understanding how application and infrastructure performance impacts customers. DevOps teams can track various metrics such as change volume, deployment frequency, lead time, percentage of failed deployments, availability, customer complaint volume, and percentage change in user volume to monitor and improve their practices. Infrastructure as Code (IaC) can automate the task of creating and managing environments, making it easier to complete repetitive tasks and avoid human error. Configuration management (CM) tools such as Chef, Puppet, and Ansible can help standardize resource configurations across an entire infrastructure and automate most system administration tasks.

CM can increase productivity and consistency by allowing teams to deploy the same configuration to hundreds of nodes at the push of a button. CM tools provide their own domain-specific language and set of features for automation. As security becomes a priority, organizations are adopting DevSecOps, which combines development, security, and operations.

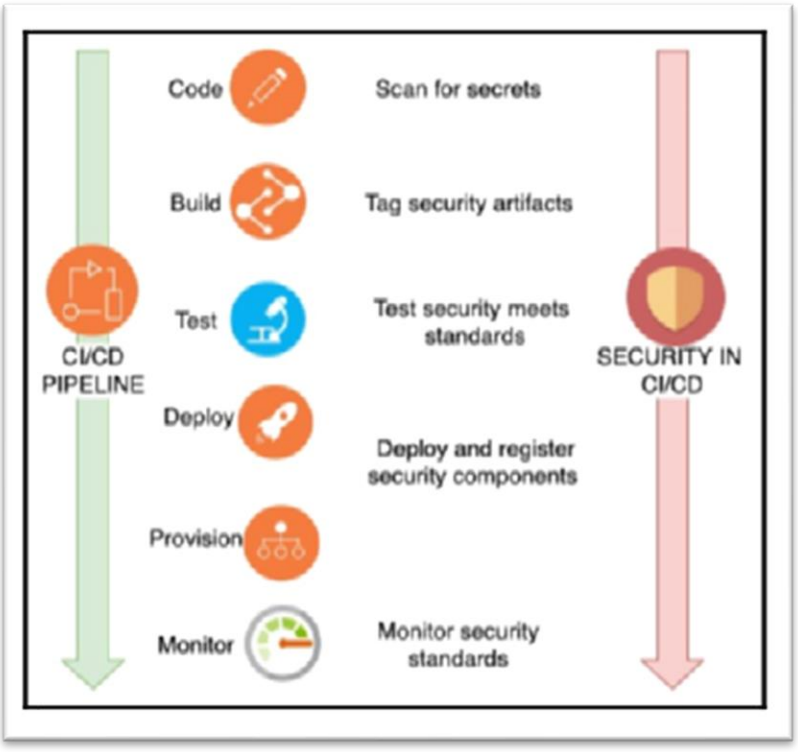
Introducing DevSecOps

DevSecOps is an approach to security that emphasizes automation and scalability. The goal is to integrate security into the software development lifecycle without slowing down the release cycle. DevSecOps is not about auditing code or artifacts, but rather about embedding security into the design and implementing continuous security testing through automation. To achieve a DevSecOps approach, organizations must start with a strong DevOps foundation and ensure that security is everyone's responsibility.

Collaboration between development and security teams is crucial to embed security in the architecture design from inception. Automating continuous security testing and monitoring for drift from the design stage in real-time is also essential. The goal is to keep the pace of innovation while meeting the pace of security automation by implementing automatic incident response remediation to ensure continuous compliance and validation.

Combining DevSecOps and CI/CD

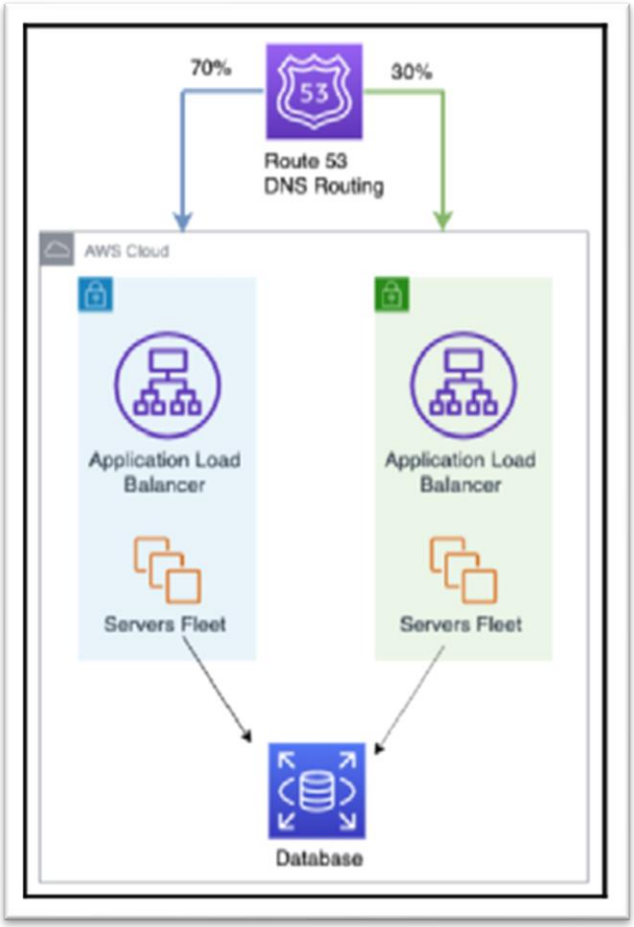
DevSecOps is a methodology that emphasizes security integration and automation throughout the entire CI/CD pipeline to ensure the security of the overall process. This includes managing access and roles assigned to servers, ensuring that build servers are hardened against security glitches, and validating all artifacts and code analysis.



DevSecOps and CI/CD

Implementing a CD strategy

Continuous Deployment (CD) allows for seamless migration from an existing version to a new version of an application. There are five popular CD techniques: In-place deployment, Rolling deployment, Blue-green deployment, Red-black deployment, and Immutable deployment.



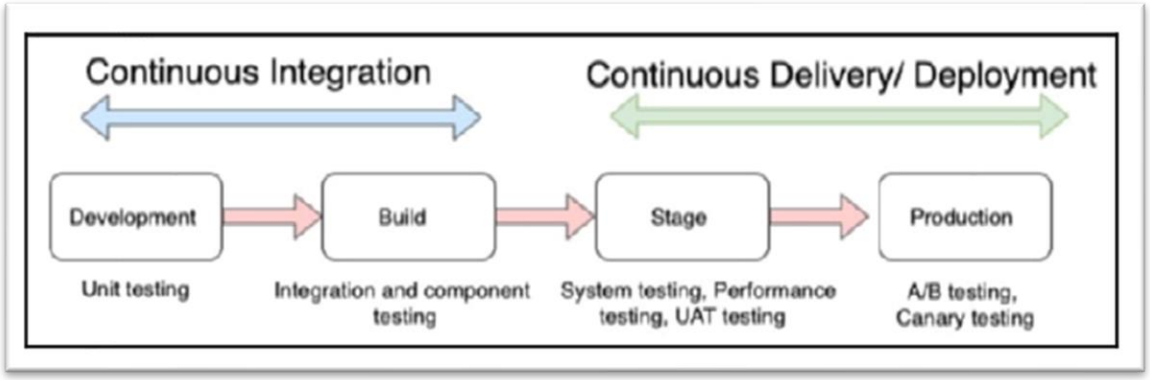
Blue-green deployment DNS gradual cutover

In-place deployment requires downtime and updates the existing fleet of servers. Rolling deployment divides the server fleet and deploys the new and old versions on different subgroups of servers. Blue-green deployment involves two identical environments, blue and green. The blue environment carries live traffic, and the green environment carries the new version of the code. Traffic is gradually shifted to the green environment, which can be monitored for issues before the switchover. Red-black deployment is like blue-green deployment but involves canary testing and a sudden DNS cutover from the old to the new version. Immutable deployment replaces older instances with new server instances.

Each deployment option has its benefits and drawbacks, but they all allow for efficient application migration.

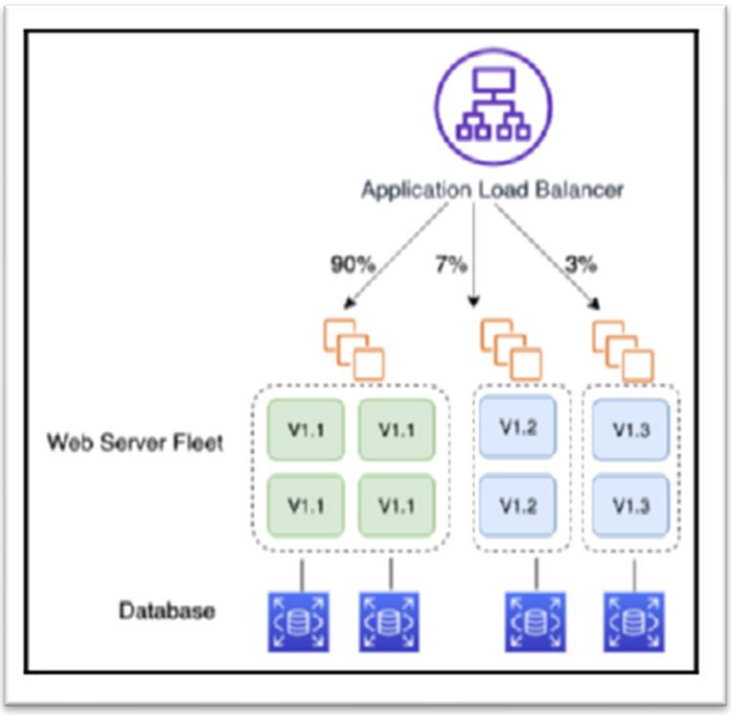
Implementing continuous testing in the CI/CD pipeline

DevOps plays a crucial role in the dynamic business environment where feedback, market trends, and new feature demands keep changing. A robust CI/CD pipeline ensures quick incorporation of feedback and new features and faster delivery of new features to customers.



Continuous testing in CI/CD

Continuous testing is critical to ensuring quality in this pipeline. Unit tests are the most significant component of the testing strategy, with developers conducting them on their machines. Once the code is ready, integration and system testing are performed in separate environments, which can be costly. A staging environment is an exact replica of production, and it is used for end-to-end system testing, performance testing, and User Acceptance Testing (UAT).



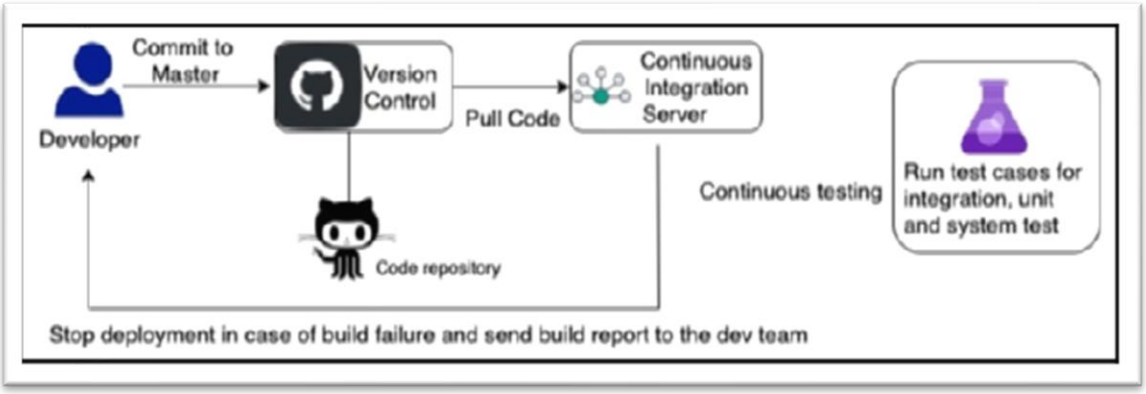
Split users by feature experiment using A/B testing

The production phase requires a testing strategy like A/B testing or canary analysis to ensure new application versions are thoroughly tested before deployment. A/B testing is a methodology that tests different versions of a feature with different user sets to determine the most suitable implementation. Load and performance testing are also critical factors in the testing process. Micro-benchmarking is a way to measure load on instances, where small sub-components or snippets of code are tested to extrapolate general performance data.

Using DevOps tools for CI/CD

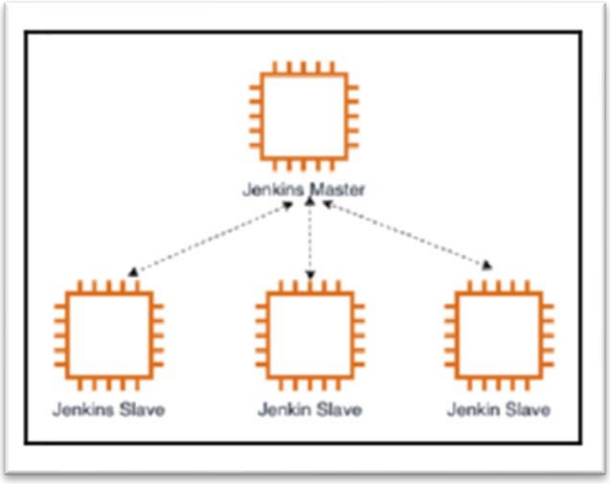
Developers require various tools to build a CI/CD pipeline, including a code editor, a source repository, a build server, a deployment tool, and an overall CI pipeline orchestrator. There are multiple technology options available for DevOps, both in the cloud and on-premises.

For code editing, developers can use a web-based editor such as the ACE editor or AWS Cloud9 IDE, or install a code editor on their local server that connects to application environments. The ACE editor is a web-based editor that provides standard IDE features, supports debugging tools and can handle large files without typing lag. For source code management, developers can set up and manage their own Git server, or use a hosting service like GitHub or Bitbucket. AWS CodeCommit is a managed source control system that offers highly scalable and secure private Git repositories.



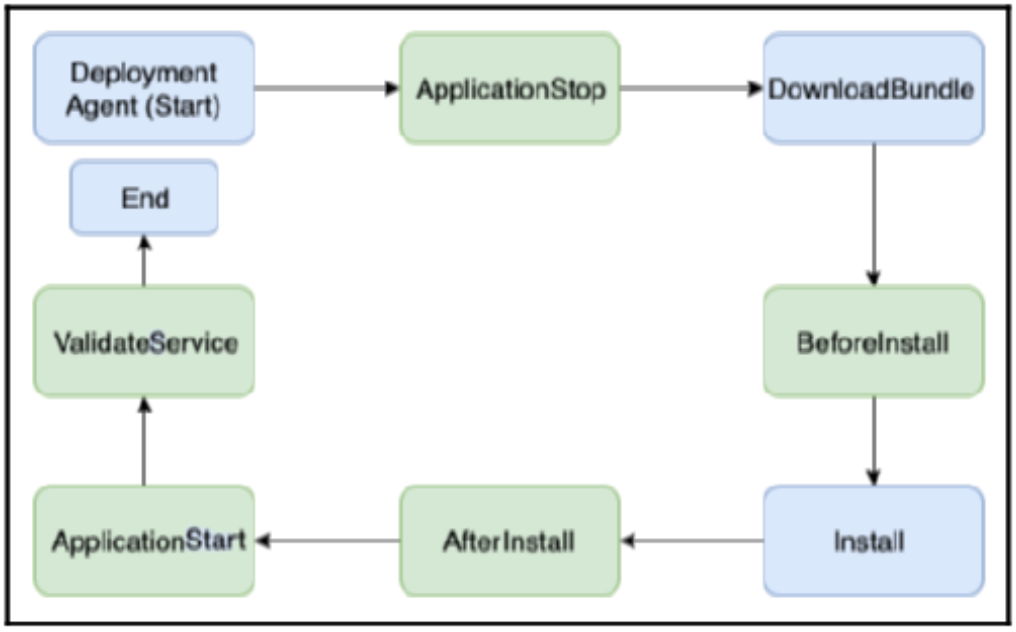
Automation of CI

A CI server or build server is essential for teams working on multiple branches to merge back into the master. CI server hooks provide a way to trigger the build based on the event when code is committed to the repository. Pull requests are a common way to notify and review each other's work before merging into common code branches.



Auto Scaling of Jenkins CI servers

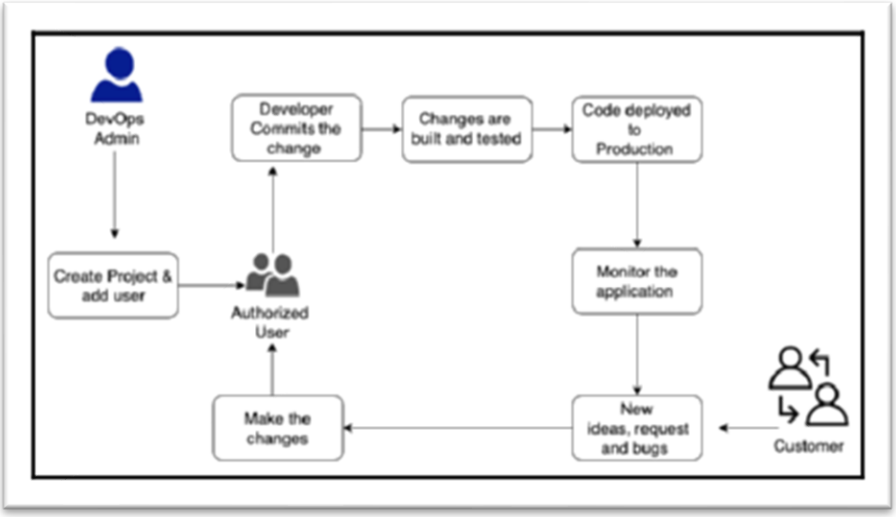
The most popular choice for building the CI server is Jenkins. However, managed code-build services like AWS CodeBuild eliminate the need for server administration and reduce costs significantly with a pay-as-you-go model.



Deployment life cycle event

Implementing DevOps best practices

When building a CI/CD pipeline, it's important to consider creating a project and adding team members to it.



CI/CD workflow best practice

The project dashboard allows you to monitor the code flow through the deployment pipeline, track application activities, trigger alerts, and monitor the build. To design the pipeline, you should consider the number of stages, the type of tests in each stage, the sequence of tests, monitoring and reporting, infrastructure provisioning, and rollback strategy. Automating the CD process will accelerate your process and avoid the need for manual intervention. It's also important to externalize build configurations to tools to keep them consistent between builds and enable better automation. The Twelve-Factor methodology is recommended for applying architecture best practices to each step of application development, regardless of programming language or platform.

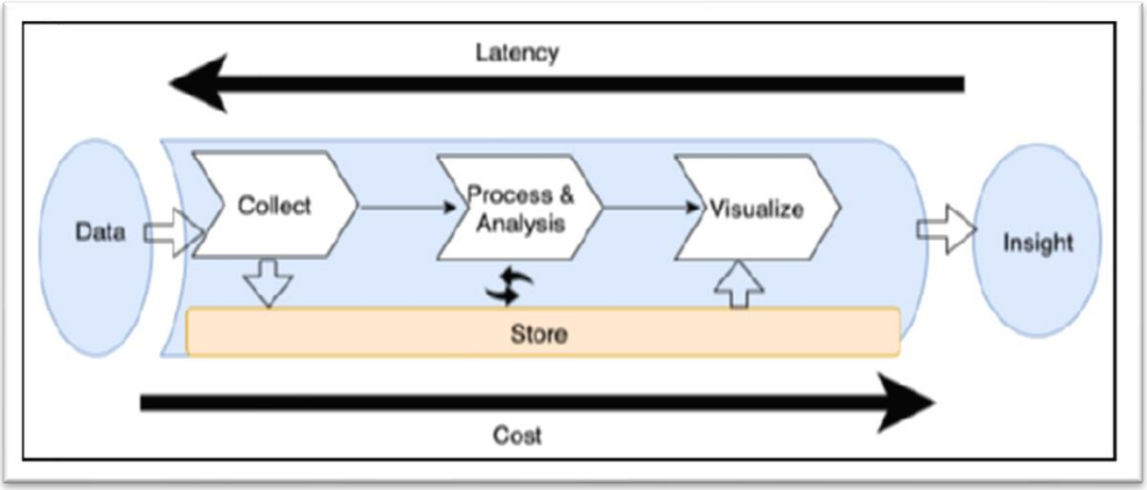
Data Engineering and Machine Learning

Enormous amounts of data are generated at high velocity and quantity, making it challenging to derive insights at a fast pace. To extract business outcomes from this data, continuous innovation is necessary to ingest, store, and process it. With the convergence of various technologies, such as cloud, mobile, and social, advancements in many fields like genomics and life sciences are growing at an ever-increasing rate. To handle unlimited data, modern stream processing systems need to produce continual results with low latency on data with high and variable input rates.

The concept of big data is not just about collecting and analyzing the data but using it to answer questions and create competitive advantages for the organization. This chapter covers topics such as big data architecture, designing for big data, data ingestion, storage, processing and analytics, data visualization, the Internet of Things (IoT), and machine learning (ML) basics. By the end of the chapter, readers will learn about designing big data and analytics architecture, different steps of the big data pipeline, and ML model evaluation techniques.

What is big data architecture?

Computer-generated data can come in various forms, such as semi-structured logs and unstructured binaries, and can be used to generate recommendations for social networking and online gaming. Human-generated data, on the other hand, includes email searches, natural language processing, sentiment analysis, and social graph analysis.

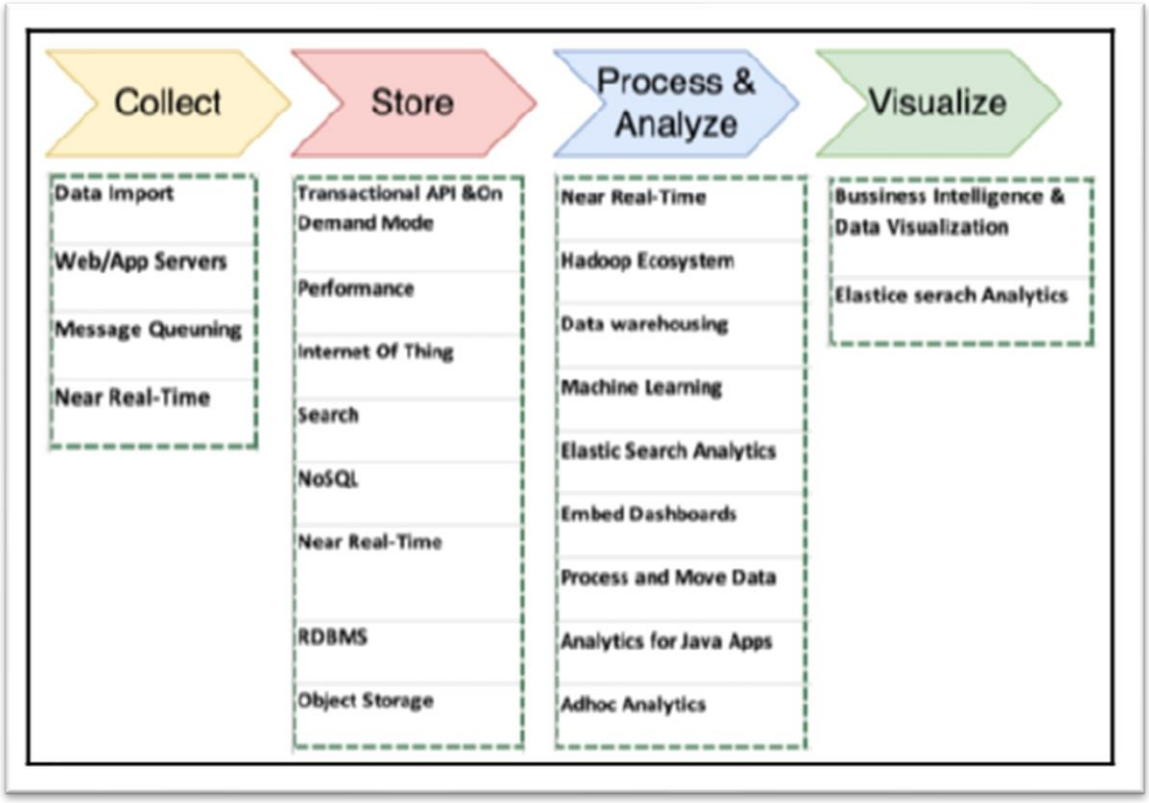


Big data pipeline for data architecture design

To design a data architecture, a standard workflow pipeline needs to be followed, which includes data ingestion, storage, processing, analysis, and visualization. The time-to-answer, which is the latency between the creation of data and when insight can be gained, is determined by the tools used in the pipeline. To balance throughput with cost, it is essential to determine the optimal performance and latency needed to achieve business goals.

Designing big data processing pipelines

In designing big data architectures, it is important to avoid the mistake of relying on a single tool to handle multiple stages of the data pipeline. This approach may seem straightforward but can be vulnerable to breakdowns and is not cost-effective.

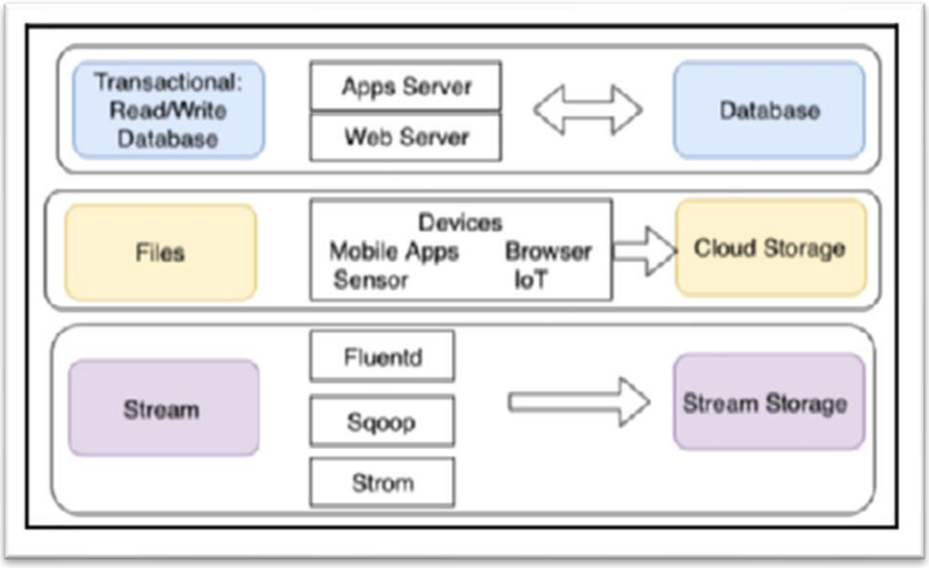


Tools and processes for big data architecture design

Decoupling storage and processing in multiple stages can increase fault tolerance and improve the efficiency of the data pipeline. The structures of the data, maximum acceptable latency, minimum acceptable throughput, and typical access patterns of end users should be considered when determining the right tools for the big data architecture. Understanding how end users will use the data can help determine the breadth and depth of the big data architecture. The following diagram provides an overview of various tools and processes involved in big data architecture design.

Data ingestion

Data ingestion is the process of collecting data from different sources and transferring it for storage. Data sources can be databases, streams, logs, or files. Databases are the most prevalent, and there are many techniques for extracting data from them. Streams are open-ended sequences of time-series data, while logs get generated by applications and operating systems. Files come from self-hosted file systems or third-party data feeds via FTP or APIs. The choice of ingestion solution is determined by the type of data that your environment collects and how it is collected.

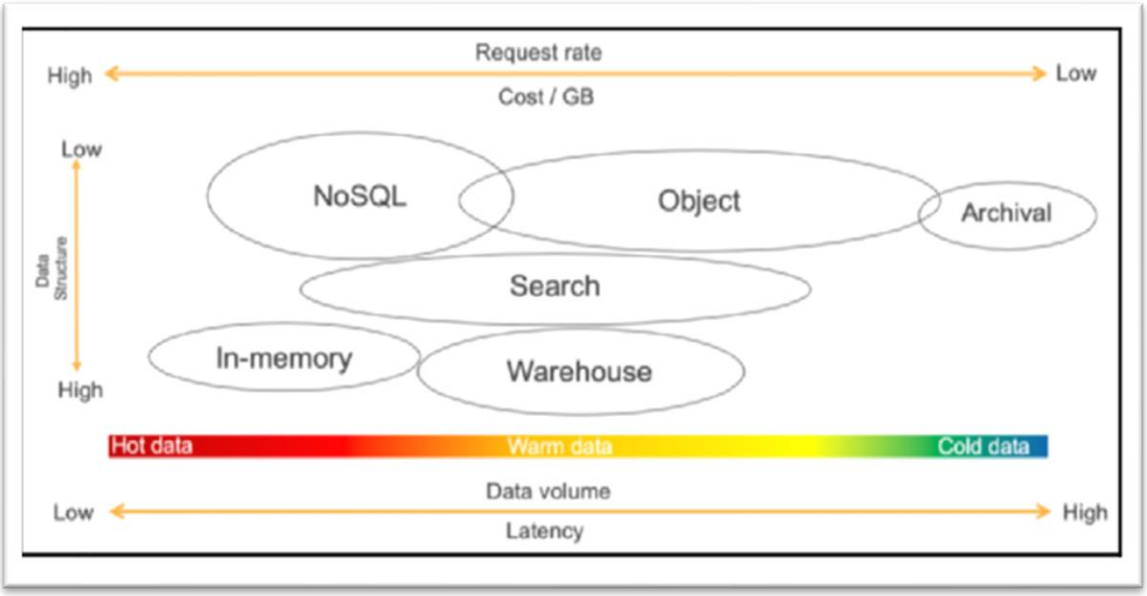


Type of data ingestion

Transactional data should be able to store and retrieve data quickly. For file data, the transfer is one-way, and data is ingested into a single object or file storage for later use. Stream data should be ingested through an appropriate solution such as Apache Kafka or Fluentd. Apache DistCp, Apache Sqoop, Apache Spark Streaming, Apache Kafka, Apache Flume, and Apache Flink are popular open-source tools for data ingestion and transfer. AWS Direct Connect, AWS Storage Gateway, and AWS Data Pipeline are some of the options available to move data to the AWS cloud.

Storing data

When setting up storage for a big data environment, using one solution, such as an RDBMS, for all data storage needs is a common mistake. It is better to use a combination of storage solutions that carefully balance latency with cost. Choosing the right data store depends on factors such as data structure, how quickly new data needs to be available for querying, the size of data ingest, the total volume of data and its growth rate, and the cost of storing and querying data. The type of analytic queries that will run against the data is also important to consider.



Understanding data storage

No single tool can solve all business problems in big data and analytics. In-memory databases like Redis or SAP Hana are appropriate for storing and processing hot data. NoSQL databases are ideal for high-velocity, small-sized records, and for content management to store data catalogs.

Structured data stores have been in use for decades and are a popular choice for data storage. Row-based formats are commonly used in transactional databases such as Oracle, MySQL, SQL Server, and PostgreSQL due to their efficiency in handling frequent data writes from software applications. However, columnar file formats are increasingly being used to enhance data read performance for analytics requirements. In columnar formats, all column values are stored together, which results in better compression and read performance.

Relational databases, such as Oracle, MSSQL, MariaDB, and PostgreSQL, are more suitable for Online Transaction Processing (OLTP) applications. They are good at handling transaction data and complex joint queries between tables. The relational database should adhere to the ACID principles of Atomicity, Consistency, Isolation, and Durability.

Data warehouse databases, such as Amazon Redshift, Netezza, and Teradata, are more suitable for Online Analytical Processing (OLAP) applications. They are designed to execute complex aggregate queries quickly and provide fast aggregation capability over vast volumes of structured data. Modern data warehouses use a columnar base to enhance query performance, and they parallelize queries across multiple nodes to take advantage of massively parallel processing. Data warehouses are central repositories that store current and

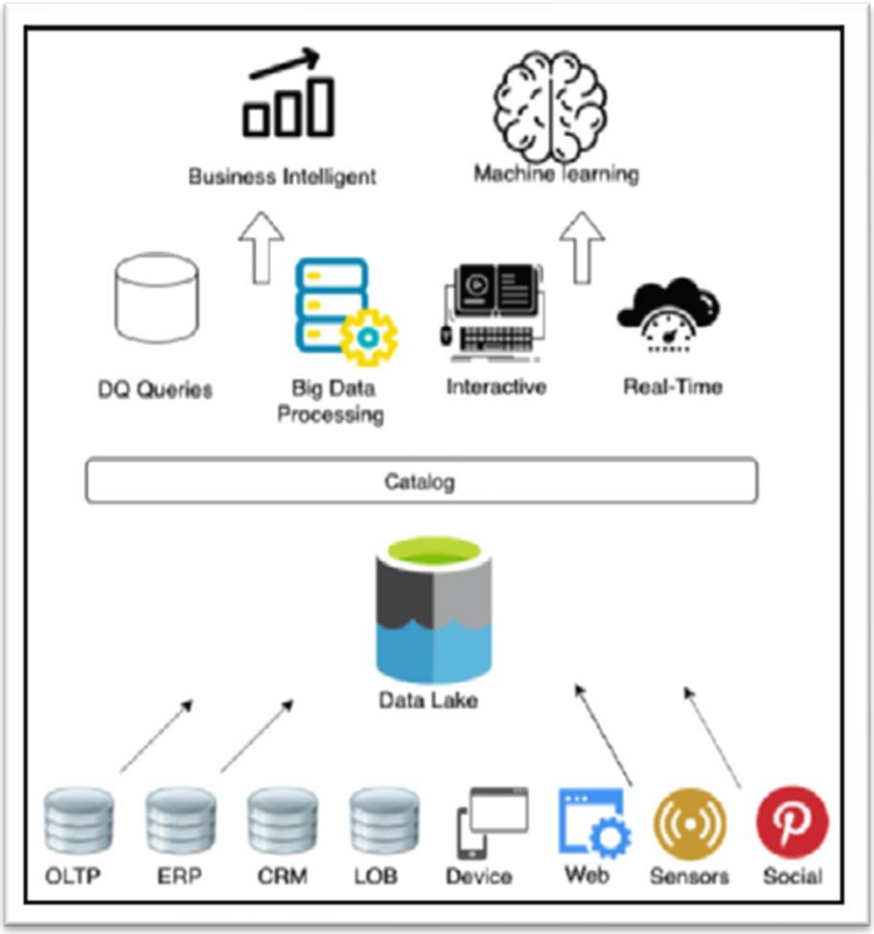
historical data used to help create analytical reports for business data analytics. However, data warehouses cannot be treated as data lakes because they only handle structured relational data, while data lakes work with both structured relational data and unstructured data.

NoSQL databases are used to address scaling and performance challenges that relational databases often encounter. NoSQL databases store data without an explicit structure to link data from different tables and use data models such as columnar, key-value, search, document, and graph.

Properties	SQL Databases	NoSQL Databases
Data model	In SQL databases, the relational model normalizes data into tables containing rows and columns. A schema includes tables, number of columns, relationships between tables, index, and other database elements.	NoSQL databases do not enforce a schema. A partition key is commonly used to retrieve values from column sets. It stores semi-structured data such as JSON, XML, or other documents such as data catalogs and file indexes.
Transaction	SQL-based traditional RDBMS support and are compliant with the transactional data properties of ACID.	To achieve horizontal scaling and data model flexibility, NoSQL databases may trade some ACID properties of traditional RDBMS.
Performance	SQL-based RDBMS were used to optimize storage when the storage was expensive and to minimize the footprint on disk. For traditional RDBMS, performance has mostly relied on the disk. To achieve performance query optimizations, index creation and modifications to the table structure are required.	For NoSQL, performance depends upon the underlying hardware cluster size, network latency, and how the application is calling the database.
Scale	SQL-based RDBMS databases are easiest to scale vertically with high configuration hardware. The additional effort requires relational tables to span across distributed systems such as performing data sharding.	NoSQL databases are designed to scale horizontally by using distributed clusters of low-cost hardware to increase throughput without impacting latency.

These databases are highly distributed and provide scalable performance, high availability, and resilience. NoSQL databases have various types, including columnar, document, graph, and in-memory key-value stores. Elasticsearch is a popular search engine for big data use cases, which works well for warm data that can be queried in an ad hoc fashion across any number of attributes. Hadoop is a perfect choice for unstructured data stores because it is scalable, extensible, and flexible. However, for maximum flexibility and cost-effectiveness, you need to separate computing and storage and scale them both independently, and object storage is more suited to data lakes to store all kinds of data in a cost-effective and performant manner.

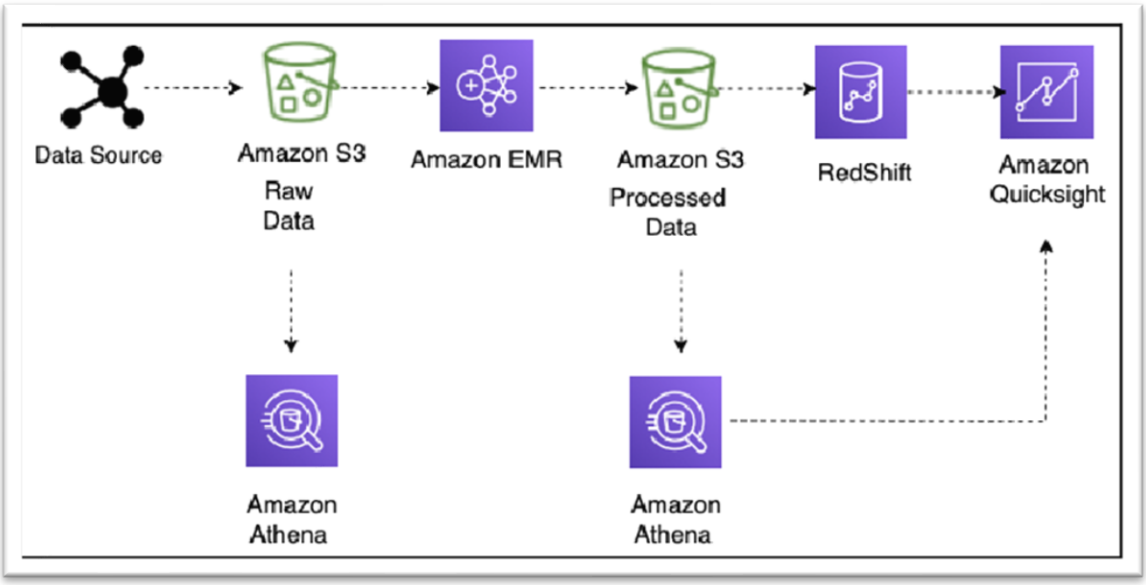
A data lake is a centralized storage repository that can hold both structured and unstructured data. It has become a popular choice for storing and analyzing vast amounts of data in a single location, using open-source file formats to facilitate direct analytics.



Object store for data lake

Processing data and performing analytics

Data analytics is a process of ingesting, transforming, and visualizing data to identify useful insights for business decision-making. Customers are increasingly looking for more insights in less time, and sometimes in real-time. Batch processing involves querying large amounts of cold data that takes hours to provide answers to business questions.



Data lake ETL pipeline for big data processing

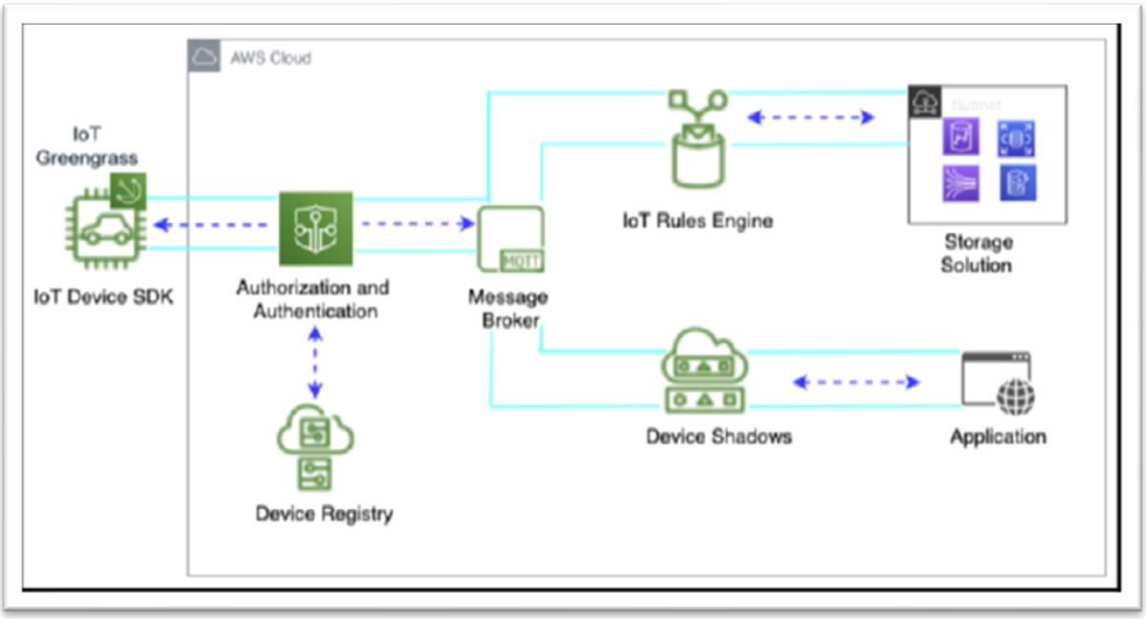
Streaming data processing ingests continuously produced streams of data records and incrementally updates functions in response to each data record. Apache Hadoop and Spark are examples of platforms that support the batch jobs category, while data warehouses support the query engine category. The extract, transform, load (ETL) pipeline is a popular method to process and transform data. Amazon Athena, Elastic MapReduce (EMR), and Redshift are used in conjunction with Amazon S3 to facilitate ad hoc querying, transform and cleanse data, and load it into Redshift. Apache Hadoop, Spark, HUE, Pig, and Hive are some popular data processing technologies. Hive is a data warehouse and query package that runs on top of a Hadoop cluster, while Pig is used to process large amounts of raw data before storing it in a structured format.

Visualizing data

The insights gained from data are crucial for businesses to answer important questions related to revenue, profit, and referrals. However, it can be challenging to obtain relevant information from the enormous amounts of data collected. BI tools can help overcome these challenges, but the cost and time involved in implementing a solution can be significant. There are several data visualization platforms available, including Amazon QuickSight, Kibana, Tableau, Spotfire, Jaspersoft, and PowerBI, each with its own unique features and benefits. As a solution architect, it is essential to be aware of these tools and choose the appropriate one as per your business requirements. With the increase in internet connectivity and the growing network of interconnected devices, there is a need to collect and analyze data from millions of these devices. IoT plays a significant role in this area and can help forecast weather for wind energy and farming, among other things.

Understanding IoT

IoT is a network of physical devices connected to the internet that can perform a range of tasks, such as predictive maintenance and energy efficiency monitoring. However, managing IoT devices and data can be complex. AWS IoT helps remove this complexity by securely connecting any number of devices to a central server, providing infrastructure to scale, and enabling businesses to gain insights from their data.



AWS IoT

The AWS IoT ecosystem includes several components, such as IoT Greengrass, the IoT Device SDK, authentication and authorization mechanisms, an IoT message broker, the IoT Rules Engine, the Device Shadow Service, and the Device Registry. These components work together to enable secure communication between devices and cloud services, as well as data processing and analytics.

While businesses can use business analytics services to analyze past events, they also need to use machine learning (ML) to answer questions about the possibilities of future events.

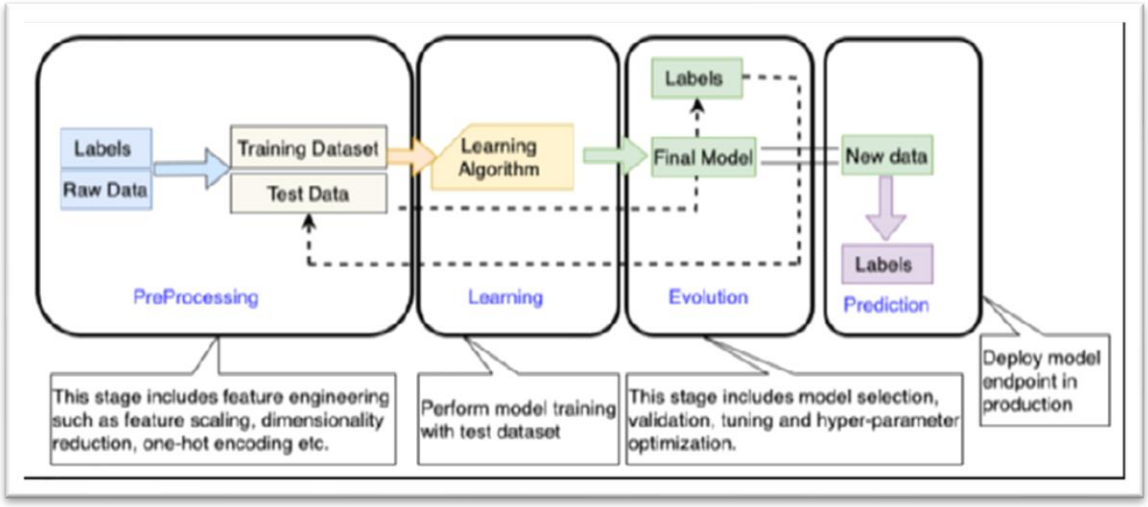
What is ML?

Machine learning (ML) can be used to solve complex problems by discovering trends and patterns in data and using mathematical predictive models based on past factual data. ML can be helpful in situations where creating complex code rules is not feasible when there is a large volume of data that needs human expertise to analyze efficiently when information needs to be adapted and personalized based on individual data, and when there are a lot of tasks with a lot of data available but the information cannot be tracked fast enough to make a rule-based decision.

ML works by training an algorithm on a dataset and using it to predict something based on new data. The quality of the data used for training the model is crucial for getting accurate predictions.

Working with data science and ML

In the field of Machine Learning (ML), working with high-quality data is crucial for the success of an ML model. However, real-world data often comes with various issues like missing values, noise, bias, outliers, etc. Therefore, data preparation, including data cleaning and feature engineering, is essential to get the data ready for ML.



ML workflow

Before starting the data preparation, it's essential to understand the business problem to avoid developing solutions that cannot address the problem. Exploring the data provides necessary information such as data quality, patterns, and potential paths forward once modeling begins.

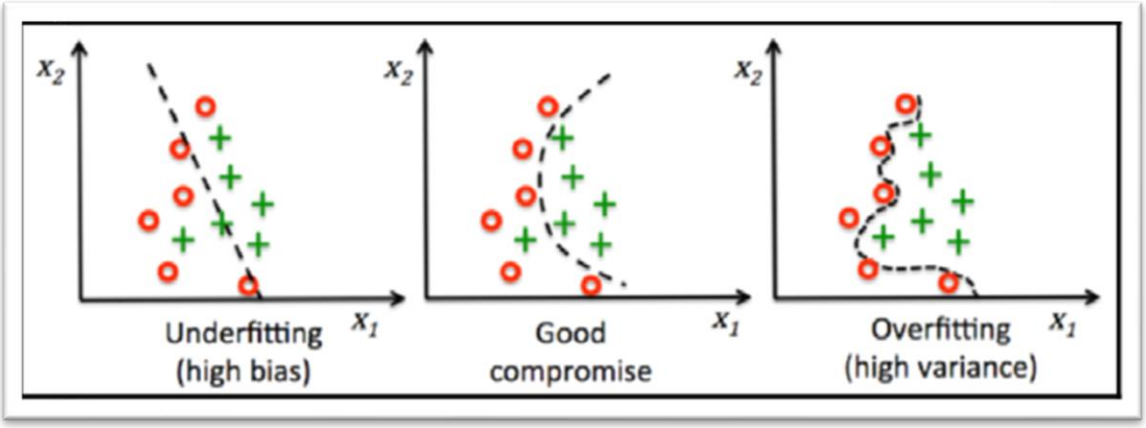
The ML workflow includes three phases: preprocessing, learning, and evaluation. In preprocessing, the data scientist preprocesses the data and divides it into training, validation, and testing datasets. Feature engineering involves finding the right features that can help achieve model accuracy, and the label is the target outcome dependent on feature selection.

- In the learning phase, choose the appropriate ML algorithm and train the ML model on the training dataset. Once trained, evaluate the model's accuracy using a validation dataset and perform model tuning if necessary.
- In the prediction phase, deploy the model to make predictions in real-time or in batches.

It's important to keep in mind that the ML model can face overfitting or underfitting issues based on the data input, which needs to be taken into account to achieve the correct outcome.

Evaluating ML models – overfitting versus underfitting

Overfitting occurs when a machine learning model is too flexible for training data, and memorizes the data, including noise, leading to poor performance on the test set. In contrast, underfitting happens when a model is too simple to capture essential patterns in the training data, leading to a systematic lack of fit in a certain region.



ML model overfitting versus underfitting

The ML model should be tuned to balance these issues and achieve a good fit. The ML workflow is categorized into supervised and unsupervised learning, and the ML algorithm is the core of the workflow.

The correct ML algorithm needs to be chosen based on the data type and the business use case.

Understanding supervised and unsupervised ML

The two main categories of machine learning (ML) are supervised and unsupervised learning. In supervised learning, the algorithm is given a set of training examples with known data and targets, and can then predict the target value for new datasets with the same attributes. In contrast, unsupervised learning is provided with massive amounts of data and must find patterns and relationships between the data to draw inferences. Reinforcement learning is another category where an algorithm is not told what action is correct but is given a reward or penalty after each action in a sequence.

Popular supervised learning algorithms include linear regression, logistic regression, neural networks, k-nearest neighbors, support vector machines, decision trees, and random forests. In contrast, k-means clustering is an example of unsupervised learning, where data is iteratively separated into k clusters by minimizing the sum of distances to the center of the closest cluster.

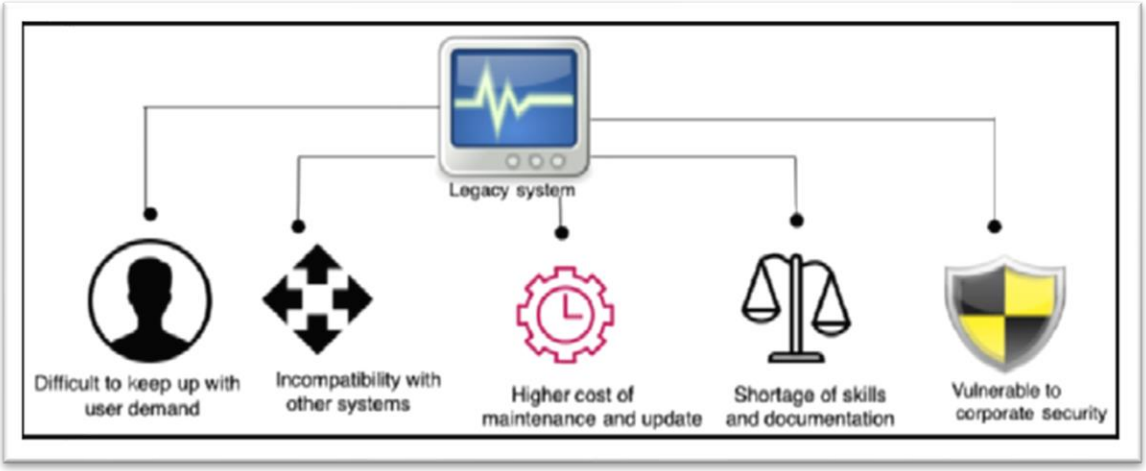
Architecting Legacy Systems

Legacy systems are applications that have been in use for decades without significant changes. These systems become outdated and difficult to maintain as technology evolves. Large enterprises often rely on legacy applications to manage critical business tasks. These systems can be found in industries such as healthcare, finance, transportation, and supply chain management. Maintaining these systems is costly, so it is necessary to consider rearchitecting them. Rearchitecting and modernizing legacy systems can increase agility, and innovation, and optimize costs, and performance.

This chapter will cover the challenges and issues with legacy systems, and techniques for rearchitecting them. Rewriting complex legacy systems may disrupt the business, so it is important to consider refactoring applications or migrating to a more flexible infrastructure. The chapter will cover the challenges of legacy systems, defining a modernization strategy, legacy system modernization techniques, and defining a cloud migration strategy for legacy systems.

Learning the challenges of legacy systems

Legacy applications present significant challenges for organizations as they strive to meet end-user demands and keep pace with the latest technological advancements. While critical applications have been in use for decades, legacy applications hinder innovation and limit an organization's ability to provide new features and benefits to end users. The challenges of legacy systems include difficulty in keeping up with user demand, higher costs of maintenance and updates, shortages of skills and documentation, vulnerability to corporate security issues, and incompatibility with other systems. Organizations that fail to modernize their legacy systems risk losing their competitive edge and potentially going bankrupt, as seen in the cases of Nokia and Kodak.



Challenges with a legacy system

Legacy systems are not very automation-friendly, and maintaining them requires a significant amount of human effort. Furthermore, legacy systems are costly due to proprietary software, license fees, and the difficulty of finding people with the necessary skills to maintain them. Unused code in legacy systems adds an extra layer of unnecessary complexity and technical debt. However, modernizing legacy systems may also be costly due to unknown dependencies and outages, and stakeholders may not immediately see the benefits of modernization, making it difficult to secure finances for legacy modernization.

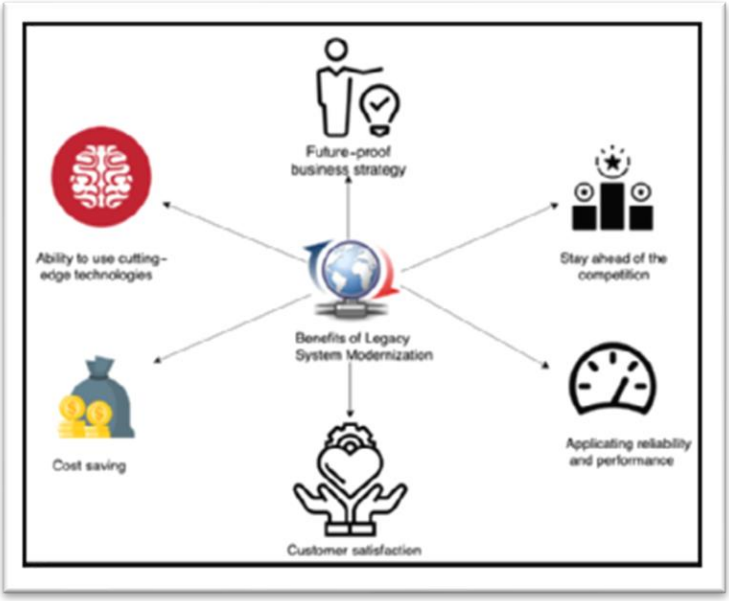
A careful cost-benefit analysis and return on investment assessment must be conducted before deciding to proceed with modernization.

Legacy systems pose multiple challenges to organizations due to skill shortages, inadequate documentation, security vulnerabilities, compliance issues, and system incompatibility. They are hardware-dependent and built on a monolithic architecture that makes it challenging to scale and maintain. As new technologies emerge, organizations risk losing out to competitors if they fail to adopt modern systems. Automation tools and modern architectures make it easier to integrate and accommodate new features, whereas the lack of APIs and standard formats increases complexity and reduces productivity.

Defining a strategy for system modernization

Legacy systems are often left out of enterprise digital strategies, leading to a reactive approach to addressing issues on an as-needed basis.

This approach hinders organizations from executing overall system modernization and benefits. There are two approaches to legacy system modernization, namely the big-bang and the phased approach. The former involves building a new system from scratch and shutting down the old one, while the latter involves upgrading one module at a time and running both old and new systems.



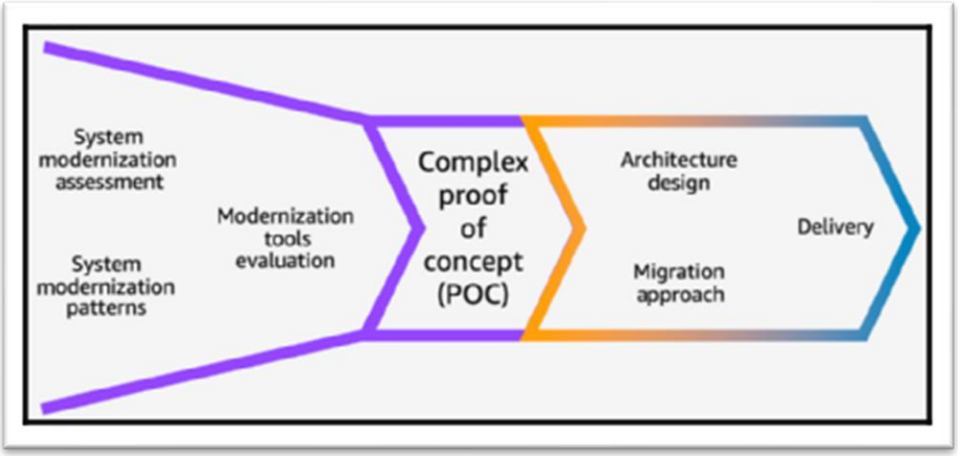
Benefits of legacy system modernization

The benefits of system modernization include better customer satisfaction, future-proof business strategy, staying ahead of the competition, application reliability and performance, ability to use cutting-edge

technology, and cost savings. However, modernizing legacy systems can be complex and require significant effort, so a careful assessment needs to be conducted to determine the appropriate approach.

To assess a legacy application, various techniques can be used to determine its suitability for modernization.

When considering modernizing a legacy system, the first step is to conduct an assessment to understand the overall system better. The technology stack, overall architecture, and code and dependency assessment are the primary areas that solution architects need to focus on when conducting an assessment.



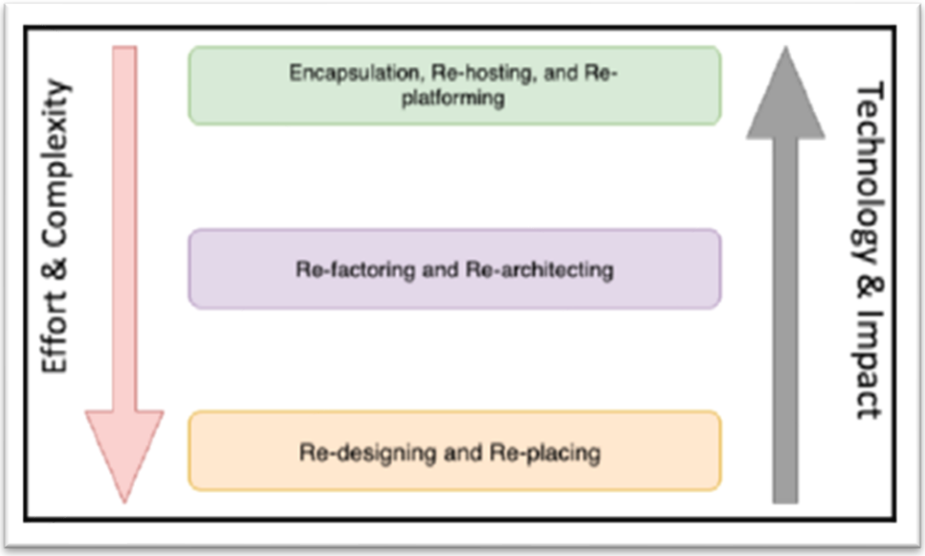
Legacy system modernization approach

After the assessment, the modernization approach should be defined. Various modernization approaches, such as architecture-driven modernization, system re-engineering, and migration and enhancements, are available to choose from.

Documentation and support should be prepared for the long-term sustainability of the new system and graceful migration to it. Keeping track of system dependencies, preparing training content, and updating support runbooks are essential to maintain the new system.

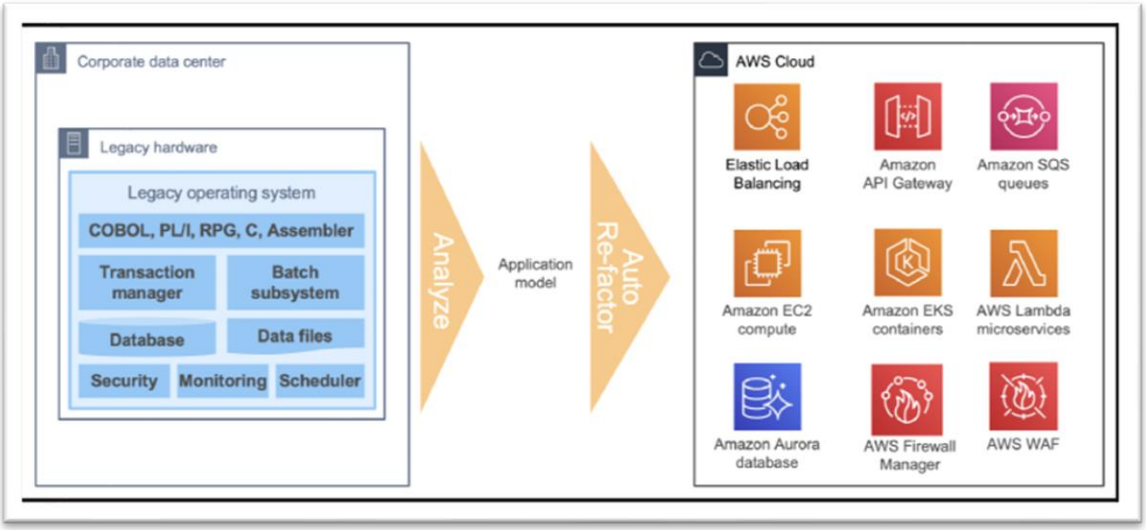
Looking at legacy system modernization techniques

According to the existing application analysis, there are different approaches to upgrading a legacy system. The most straightforward approach is migration and rehosting, where minimal changes are made to the existing system. However, this may not solve the long-term problem or provide benefits.



Legacy system modernization techniques

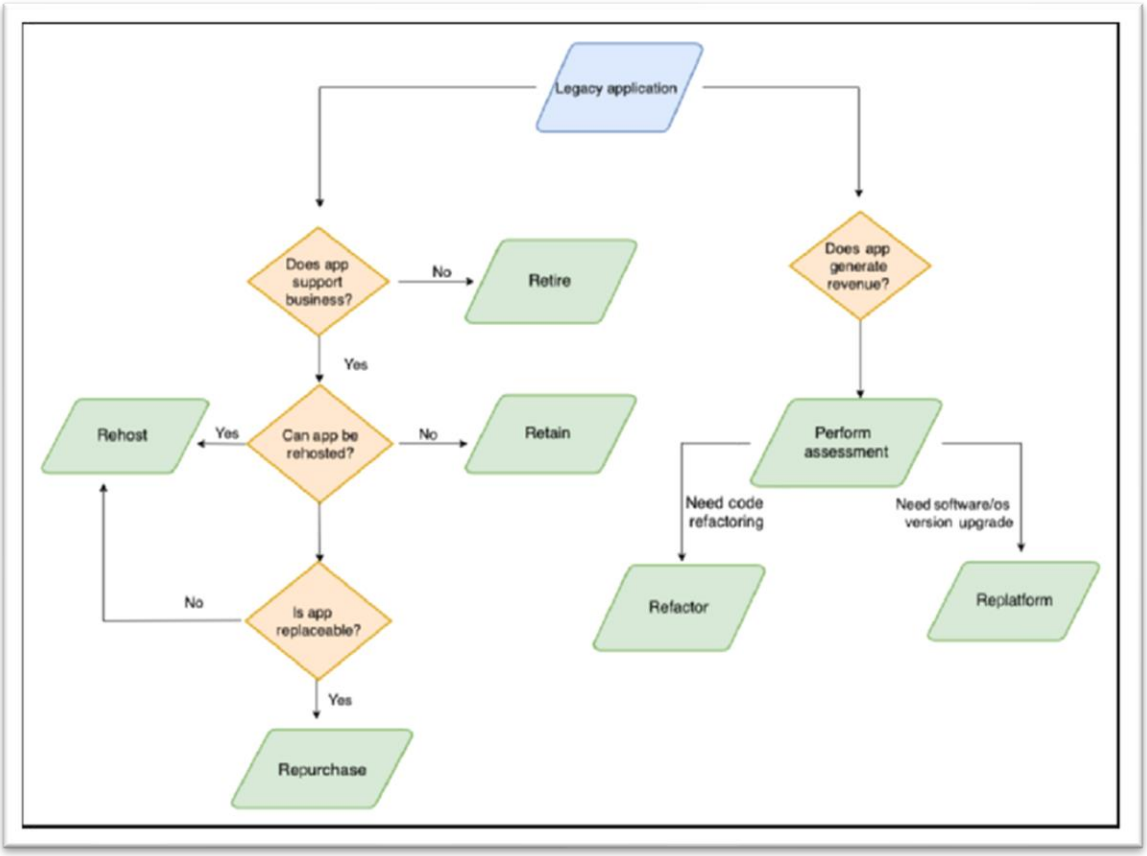
A more complex approach, such as rearchitecting or redesigning the entire application, can be taken if the system no longer meets business needs. Modernization techniques like encapsulation, rehosting, and re-platforming are the simplest approaches that provide partial benefits



Legacy mainframe system modernization to the cloud

Defining a cloud migration strategy for legacy systems

The increasing popularity of cloud technology has led many organizations to consider migrating their legacy applications to the cloud. Cloud migration offers benefits such as scalability, cost-effectiveness, enhanced security, availability, and reliability. Cloud providers like Amazon Web Services (AWS) provide various options to modernize systems, such as using AWS Lambda, Amazon API Gateway, and Amazon DynamoDB to build microservices.



Cloud migration path for legacy system modernization

To determine whether cloud migration is appropriate for a legacy system, a flow diagram is presented. If the system generates revenue and is still heavily used, it may be best to refactor or re-platform it. If cost optimization is the goal, then the lift and shift approach may be the best option. If the legacy application can be replaced, purchasing a cloud-native SaaS version may be feasible. If too many business dependencies are involved, retaining the legacy system on-premises may be necessary. A total cost of ownership (TCO) analysis is recommended, as well as building a proof of concept (POC) for the most complex module to reduce migration risk.