



CLOUD COMPUTING

Introduction to Cloud Computing

K.S.Srinivas

Department of Computer Science and Engineering

CLOUD COMPUTING

Introduction to Cloud Computing

Srinivas K S.

Associate Professor, Department of Computer Science

Introduction to Cloud Computing

Cloud Computing

- Cloud computing is basically delivering computing at the Internet scale.(T2)
- Compute, storage, networking infrastructure as well as development and deployment platforms are made available on-demand within minutes(T2)
- *Cloud computing is a model for enabling **ubiquitous**, convenient, **on-demand** network access to a **shared pool** of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be **rapidly provisioned** and released with minimal management effort or service provider interaction*
- The term is generally used to describe data centers available to many users over the Internet.
- Large clouds, predominant today, often have functions distributed over multiple locations from central servers
- If the connection to the user is relatively close, it may be designated an edge server.
• (https://en.wikipedia.org/wiki/Cloud_computing)

CLOUD COMPUTING

Introduction to Cloud Computing



Features/Characteristics of Cloud Computing(T2)

On demand self-service: The compute, storage or platform resources needed by the user of a cloud platform are self-provisioned or auto provisioned with minimal configuration.

Broad network access: Ubiquitous access to cloud applications from desktops, laptops to mobile devices is critical to the success of a Cloud platform

Resource pooling: Cloud services can support millions of concurrent users; cloud services need to share resources between users and clients in order to reduce costs.

Rapid elasticity: A cloud platform should be able to rapidly increase or decrease computing resources as needed.

Measured service: One of the compelling business use cases for cloud computing is the ability to “pay as you go,” where the consumer pays only for the resources that are actually used by her applications

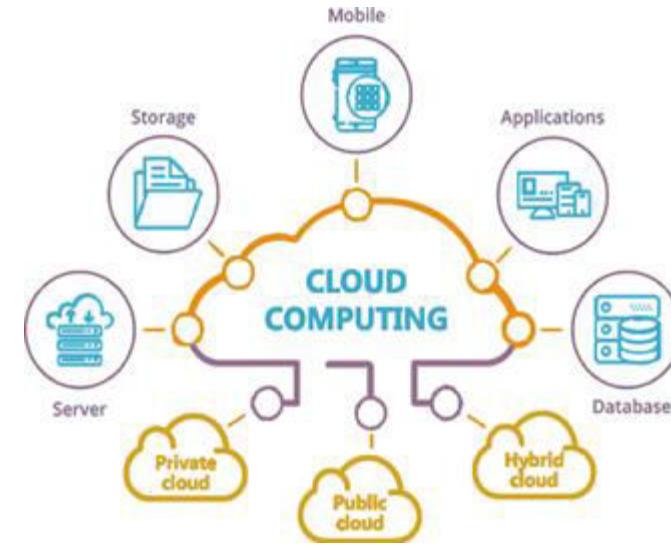
CLOUD COMPUTING

Introduction to Cloud Computing

Cloud computing usage (<https://aws.amazon.com/what-is-cloud-computing/>)

Organizations of every type, size, and industry are using the cloud for a wide variety of use cases, such as data backup, disaster recovery, email, virtual desktops, software development and testing, big data analytics, and customer-facing web applications.

- For example, healthcare companies are using the cloud to develop more personalized treatments for patients.
- Financial services companies are using the cloud to power real-time fraud detection and prevention.
- And video game makers are using the cloud to deliver online games to millions of players around the world.



<https://networkencyclopedia.com/cloud-computing/>

CLOUD COMPUTING

Introduction to Cloud Computing



Cloud Computing Benefits

Agility

Quickly spin up resources as you need them—from infrastructure services, such as compute, storage, and databases, to Internet of Things, machine learning, data lakes and analytics, and much more.

Elasticity

Scale these resources up or down to instantly grow and shrink capacity as your business needs change.

Cost savings

The cloud allows you to trade capital expenses (such as data centers and physical servers) for variable expenses, and only pay for IT as you consume it.

Deploy globally in minutes

With the cloud, you can expand to new geographic regions and deploy globally in minutes

Introduction to Cloud Computing

Genesis of Cloud Computing(T1 1.1)

Over the past 60 years, computing technology has undergone a series of platform and environment changes

Changes in machine architecture, operating system platform, network connectivity, and application workload.

Instead of using a centralized computer to solve computational problems, a parallel and distributed computing system uses multiple computers to solve large-scale problems over the Internet.

The general computing trend is to leverage shared web resources and massive amounts of data over the Internet.

Evolution of Cloud Computing platforms (T1 – 1.1.1.1)

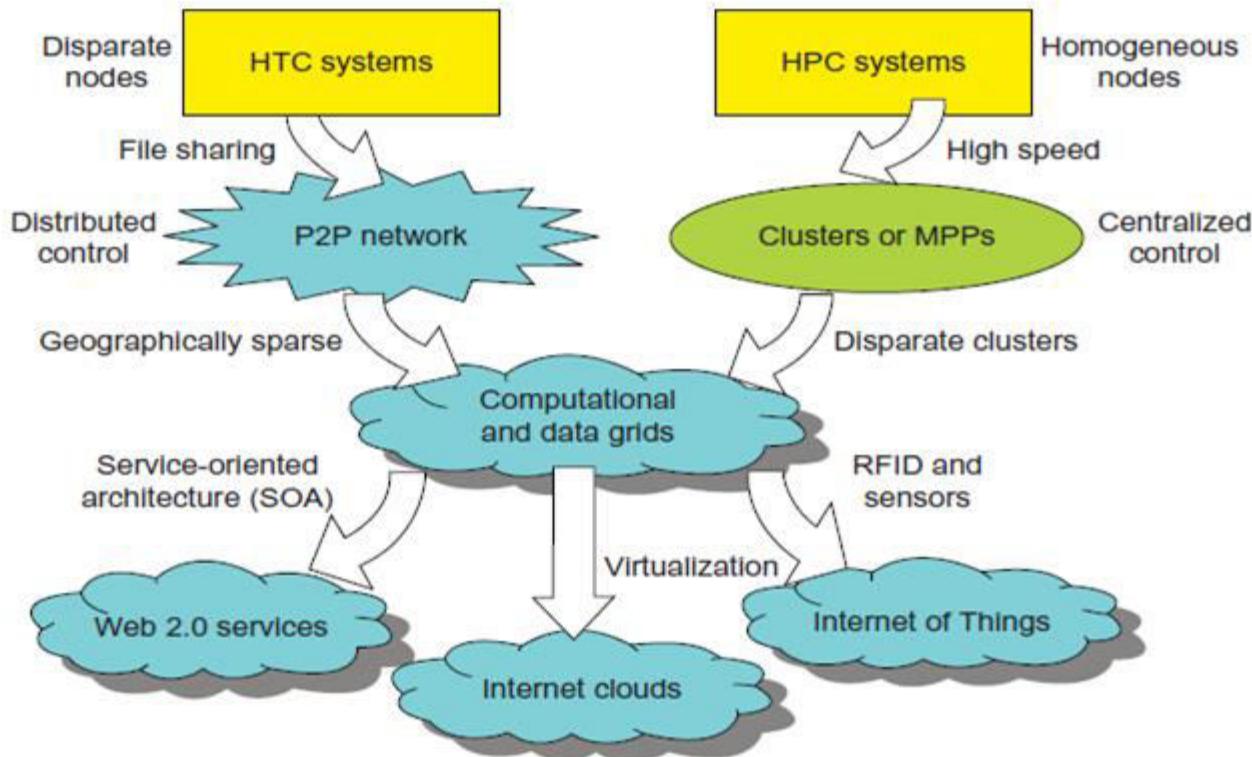


FIGURE 1.1

Evolutionary trend toward parallel, distributed, and cloud computing with clusters, MPPs, P2P networks, grids, clouds, web services, and the Internet of Things.

Introduction to Cloud Computing

Evolution of Cloud Computing (T1 1.1.1.1)

Billions of users use the internet everyday. High performance computing is no longer optimal.

High throughput computing is what is needed using distributed and parallel computing.

Since 1990, the use of both HPC and HTC systems hidden in clusters, grids, or Internet clouds has proliferated.

These systems are employed by both consumers and high-end web-scale computing and information services

The general computing trend is to leverage shared web resources and massive amounts of data over the Internet

Introduction to Cloud Computing

Terms

High Performance Computing (<https://insidehpc.com/hpc-basic-training/what-is-hpc/>)

Most generally refers to the practice of aggregating computing power in a way that delivers much higher performance than one could get out of a typical desktop computer or workstation in order to solve large problems in science, engineering, or business.

A helpful way to help understand what high performance computers are is to think about the what's in them. You have all of the elements you'd find on your desktop — processors, memory, disk, operating system — [just more of them](#)

High performance computers of interest to small and medium-sized businesses today are really *clusters* of computers.

People often refer to the individual *computers* in a cluster as *nodes*

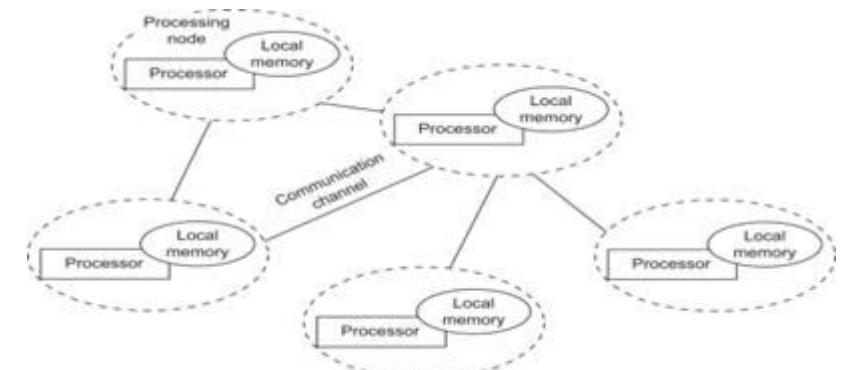
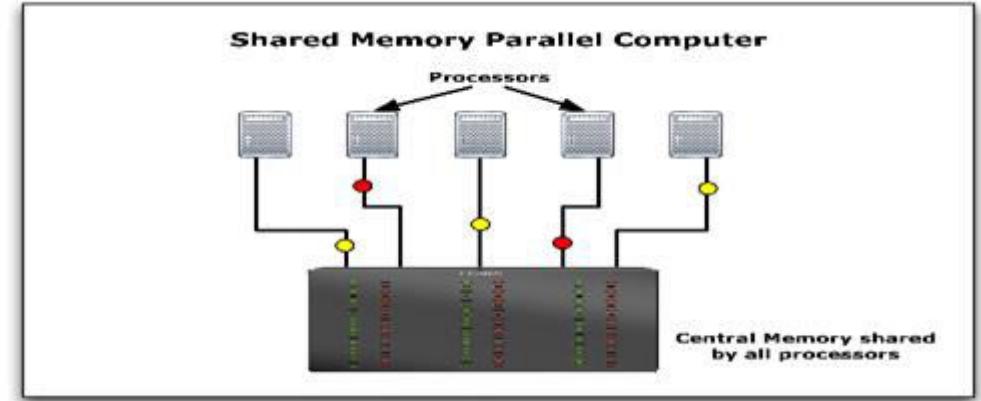
Computing Paradigms distinctions (T1 1.1.1.5)

Distributed computing

A distributed system consists of multiple autonomous computers, each having its own private memory, communicating through a computer network. Information exchange in a distributed system is accomplished through message passing

Parallel Computing

In parallel computing, all processors are either tightly coupled with centralized shared memory or loosely coupled with distributed memory. Inter- processor communication is accomplished through shared memory or via message passing.



CLOUD COMPUTING

Introduction to Cloud Computing

CLOUD COMPUTING

An Internet cloud of resources can be either a centralized or a distributed computing system. The cloud applies parallel or distributed computing, or both. Clouds can be built with physical or virtualized resources over large data centers that are centralized or distributed.

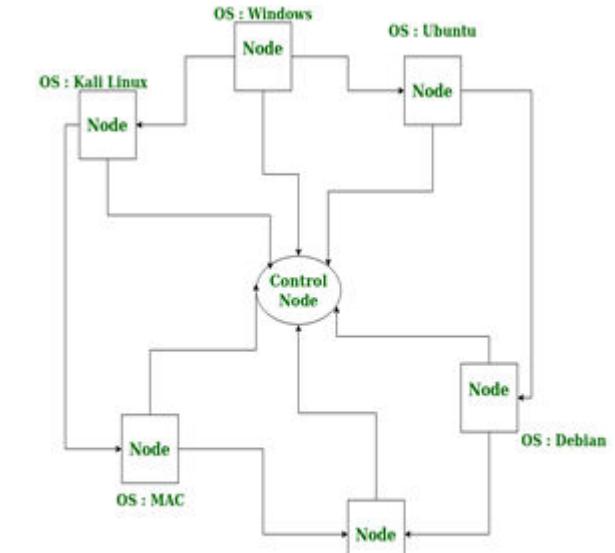
GRID COMPUTING (https://en.wikipedia.org/wiki/Grid_computing)

Grid computing is the use of widely distributed computer resources to reach a common goal.

A computing grid can be thought of as a distributed system with non-interactive workloads that involve many files.

Grid computing is distinguished from conventional high-performance computing systems such as cluster computing in that grid computers have each node set to perform a different task/application.

Grid computers also tend to be more heterogeneous and geographically dispersed (thus not physically coupled) than cluster computers



Topology in Grid Computing

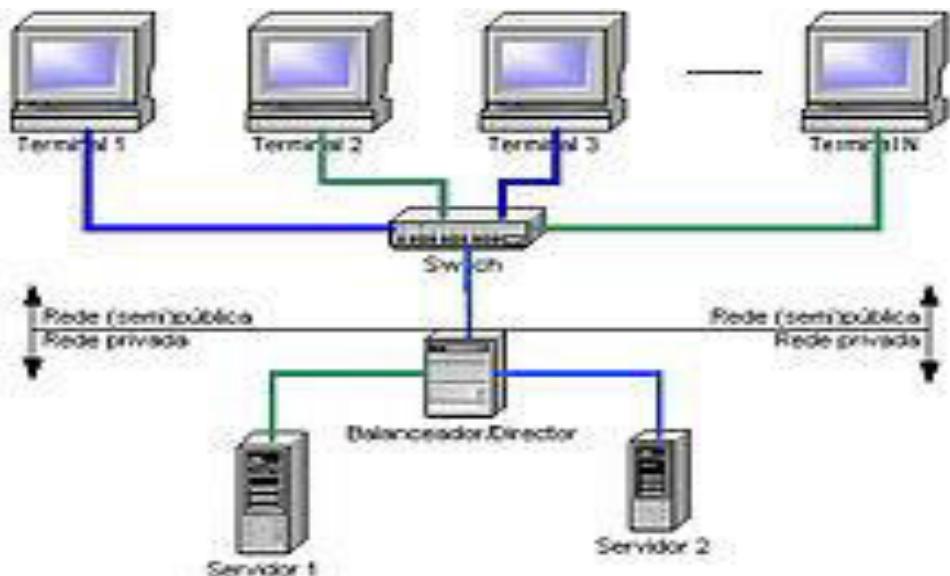
<https://www.geeksforgeeks.org/grid-computing/>

Introduction to Cloud Computing

Grid computing and its salient features and differences with cloud computing are covered in later slides and lectures

CLUSTERS (https://en.wikipedia.org/wiki/Computer_cluster)

A computer cluster is a set of loosely or tightly connected computers that work together so that, in many aspects, they can be viewed as a single system. Unlike grid computers, computer clusters have each node set to perform the same task, controlled and scheduled by software.



Introduction to Cloud Computing

Applications of Cloud Computing (T1 1.1.2.2)

Domain	Specific Applications
Science and engineering	Scientific simulations, genomic analysis, etc. Earthquake prediction, global warming, weather forecasting, etc.
Business, education, services industry, and health care	Telecommunication, content delivery, e-commerce, etc. Banking, stock exchanges, transaction processing, etc. Air traffic control, electric power grids, distance education, etc. Health care, hospital automation, telemedicine, etc.
Internet and web services, and government applications	Internet search, data centers, decision-making systems, etc. Traffic monitoring, worm containment, cyber security, etc. Digital government, online tax return processing, social networking, etc.
Mission-critical applications	Military command and control, intelligent systems, crisis management, etc.

Introduction to Cloud Computing

Design objectives of cloud computing

Efficiency measures the utilization rate of resources in an execution model by exploiting massive parallelism in HPC. For HTC, efficiency is more closely related to job throughput and power efficiency.

Dependability measures the reliability and self-management from the chip to the system and application levels. The purpose is to provide high-throughput service with Quality of Service (QoS) assurance, even under failure conditions.

Adaptation in the programming model measures the ability to support billions of job requests over massive data sets and virtualized cloud resources under various workload and service models.

Flexibility in application deployment measures the ability of distributed systems to run well in both HPC (science and engineering) and HTC (business) applications.

Introduction to Cloud Computing

Compute as a utility (T1 1.1.2.3)

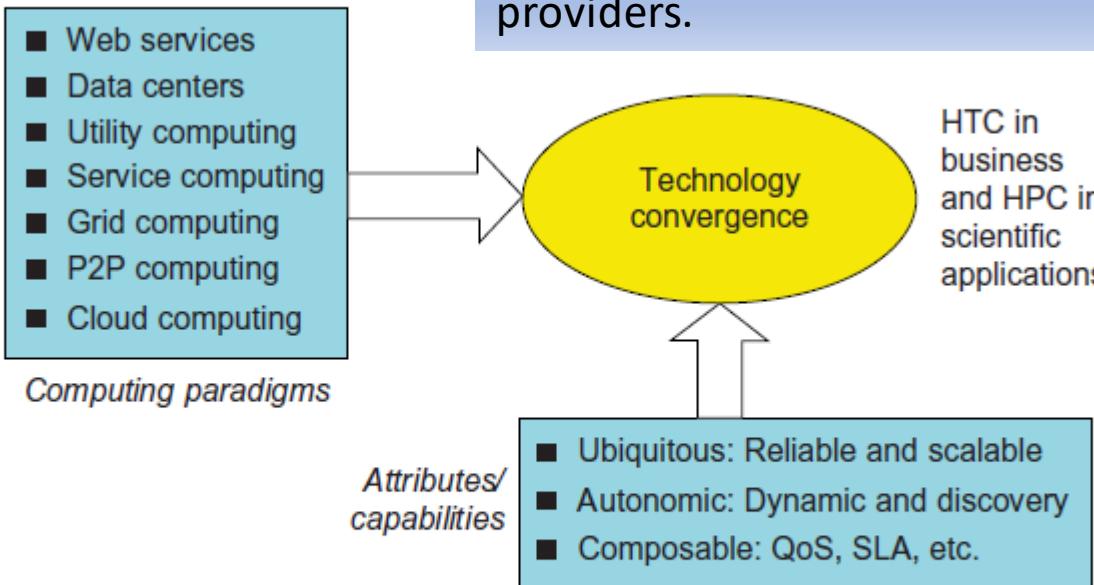


FIGURE 1.2

The vision of computer utilities in modern distributed computing systems.

(Modified from presentation slide by Raj Buyya, 2010)

Introduction to Cloud Computing

Evolution of the Web (A key enabler in cloud computing)

Web 1.0

Design principles

1. Four design essentials of a Web 1.0 site include: Static pages.
2. Content is served from the server's file-system.
3. Pages built using Server Side Includes or Common Gateway Interface (CGI).
4. Frames and Tables used to position and align the elements on a page.

Web 2.0 –

Web 2.0 refers to world wide website which highlight user-generated content, usability and interoperability for end users. Web 2.0 is also called participative social web.

Ajax and JavaScript frameworks extensively used in Web 2.0

Major features of Web 2.0 –

1. Permits users to retrieve and classify the information collectively.
2. Dynamic content that is responsive to user input.
3. Information flows between site owner and site users by means of evaluation & online commenting.
4. Developed APIs to allow self-usage, such as by a software application.

Introduction to Cloud Computing

Web 3.0

1. Semantic Web

Generate, share and connect content through search and analysis based on **the ability to understand the meaning of words**, rather than on keywords or numbers.

2. Artificial Intelligence

Combining this capability with natural language processing, in Web 3.0,

3. 3D Graphics

The three dimensional design is being used extensively in websites and services in Web 3.0. Museum guides, computer games, ecommerce, geospatial contexts, etc. are all examples that use 3D graphics.

4. Ubiquity

Content is accessible by multiple applications, every device is connected to the web, the services can be used everywhere.



THANK YOU

K.S.Srinivas
srinivasks@pes.edu
+91 80 2672 1983 Extn 701



CLOUD COMPUTING

Parallel and Grid Computing

K.S.Srinivas

Department of Computer Science and Engineering

CLOUD COMPUTING

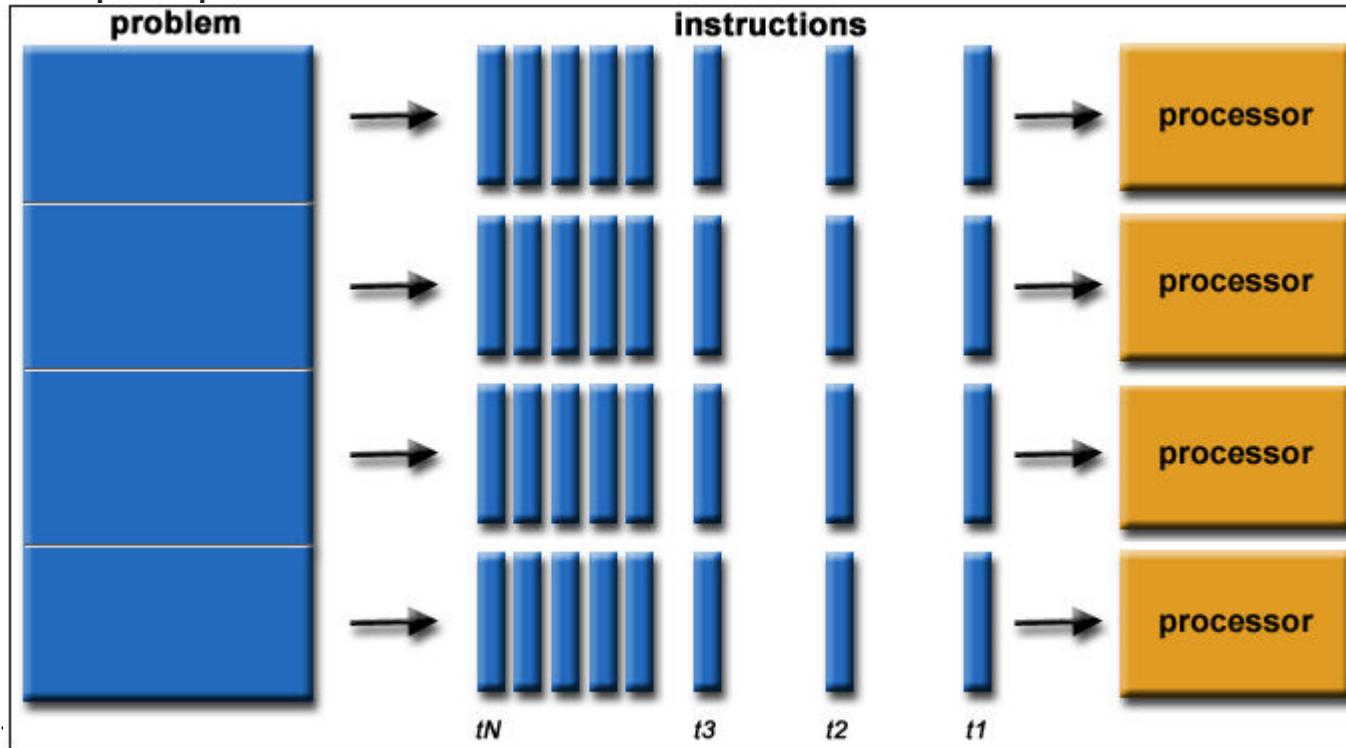
Parallel and Grid Computing

Srinivas K S.

Associate Professor, Department of Computer Science

Parallel Computing (<https://www.omnisci.com/technical-glossary/parallel-computing>)

Parallel computing is a type of computing architecture in which several processors simultaneously execute multiple, smaller calculations broken down from an overall larger, complex problem.



Parallel computing refers to the process of breaking down larger problems into **smaller, independent, often similar parts that can be executed simultaneously by multiple processors** communicating via shared memory, the **results of which are combined upon completion** as part of an overall algorithm.

The primary goal of parallel computing is **to increase available computation power** for faster application processing and problem solving

Computation requests are distributed in small chunks by the application server that are then executed **simultaneously on each server**

There are various degrees of parallelism

- **Bit Level Parallelism**
- **Instruction Level Parallelism**
- **Task Level Parallelism**
- **Data Parallelism**

Bit Level Parallelism (<https://www.tutorialspoint.com/types-of-parallelism-in-processing-execution>)

Bit-level parallelism is a form of parallel computing which is based on **increasing processor word size**. In this type of parallelism, with increasing the word size **reduces the number of instructions** the processor must execute in order to perform an operation on variables whose sizes are greater than the length of the word.

E.g., consider a case where an 8-bit processor must add two 16-bit integers. First the 8 lower-order bits from each integer were must added by processor, then add the 8 higher-order bits, and then two instructions to complete a single operation. A processor with 16-bit would be able to complete the operation with single instruction.

Instruction Level Parallelism

Instruction-level parallelism means the simultaneous execution of multiple instructions from a program. While pipelining is a form of ILP, we must exploit it to achieve parallel execution of the instructions in the instruction stream.

Example

```
for (i=1; i<=100; i= i+1)
    y[i] = y[i] + x[i];
```

This is a parallel loop. Every iteration of the loop can overlap with any other iteration, although within each loop iteration there is little opportunity for overlap.

Data Parallelism

Data Parallelism means concurrent execution of the same task on each multiple computing core.

Let's take an example, summing the contents of an array of size N . For a single-core system, one thread would simply sum the elements $[0] \dots [N - 1]$. For a dual-core system, however, thread A, running on core 0, could sum the elements $[0] \dots [N/2 - 1]$ and while thread B, running on core 1, could sum the elements $[N/2] \dots [N - 1]$. So the Two threads would be running in parallel on separate computing cores.

Task Parallelism

Task Parallelism means concurrent execution of the different task on multiple computing cores.

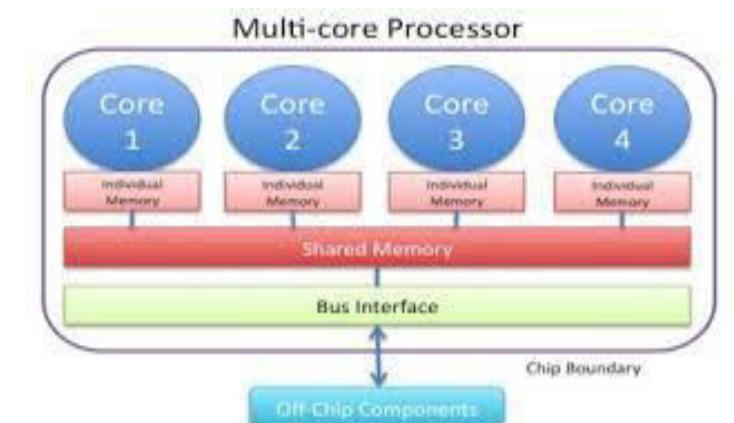
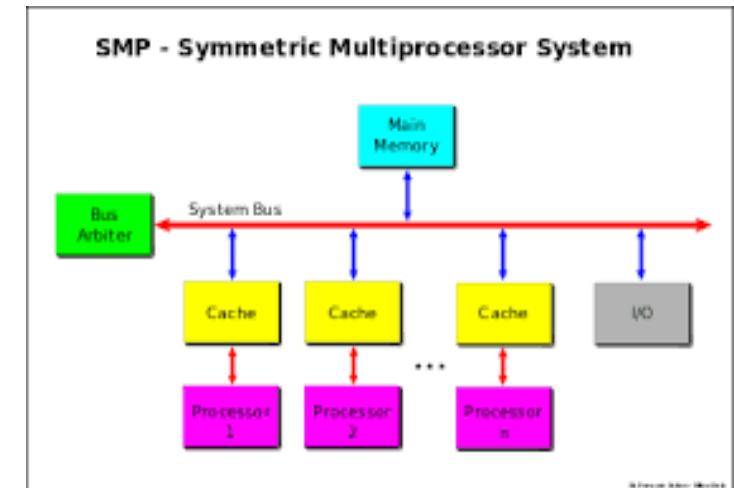
Consider again our example above, an example of task parallelism might involve two threads, each performing a unique statistical operation on the array of elements. Again The threads are operating in parallel on separate computing cores, but each is performing a unique operation.

Parallel Computing Architecture (<https://www.omnisci.com/technical-glossary/parallel-computing>)

Multi-core computing: A multi-core processor is a computer processor integrated circuit with two or more separate processing cores, each of which executes program instructions in parallel.

Symmetric multiprocessing: multiprocessor computer hardware and software architecture in which two or more independent, homogeneous processors are controlled by a **single operating system** instance that treats all processors equally, and is connected to a **single, shared main memory** with full access to all common resources and devices

Each processor has a private cache memory, may be connected using on-chip mesh networks, and can work on any task no matter where the data for that task is located in memory.



Distributed computing: Distributed system components are located on different networked computers that coordinate their actions by communicating via pure HTTP, RPC-like connectors, and message queues.

Significant characteristics of distributed systems include independent failure of components and concurrency of components

Massively parallel computing: refers to the use of numerous computers or computer processors to simultaneously execute a set of computations in parallel.

One approach involves the grouping of several processors in a tightly structured, centralized computer cluster. Another approach is grid computing, in which many widely distributed computers work together and communicate via the Internet to solve a particular problem.

Parallel Computing Software Solutions and Techniques

- **Application checkpointing:** a technique that provides fault tolerance for computing systems by recording all of the application's current variable states, enabling the application to restore and restart from that point in the instance of failure.
- **Automatic parallelization:** refers to the conversion of sequential code into multi-threaded code in order to use multiple processors simultaneously in a shared-memory multiprocessor (SMP) machine.
- **Parallel programming languages:** Parallel programming languages are typically classified as either distributed memory or shared memory. While distributed memory programming languages use message passing to communicate, shared memory programming languages communicate by manipulating shared memory variables.

Difference Between Parallel Computing and Cloud Computing

Cloud computing is a general term that refers to the **delivery of scalable services, such as databases, data storage, networking, servers, and software, over the Internet on an as-needed, pay-as-you-go basis.**

Cloud computing services can be public or private, are fully managed by the provider, and facilitate remote access to data, work, and applications from any device in any place capable of establishing an Internet connection.

The three most common service categories are Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS).

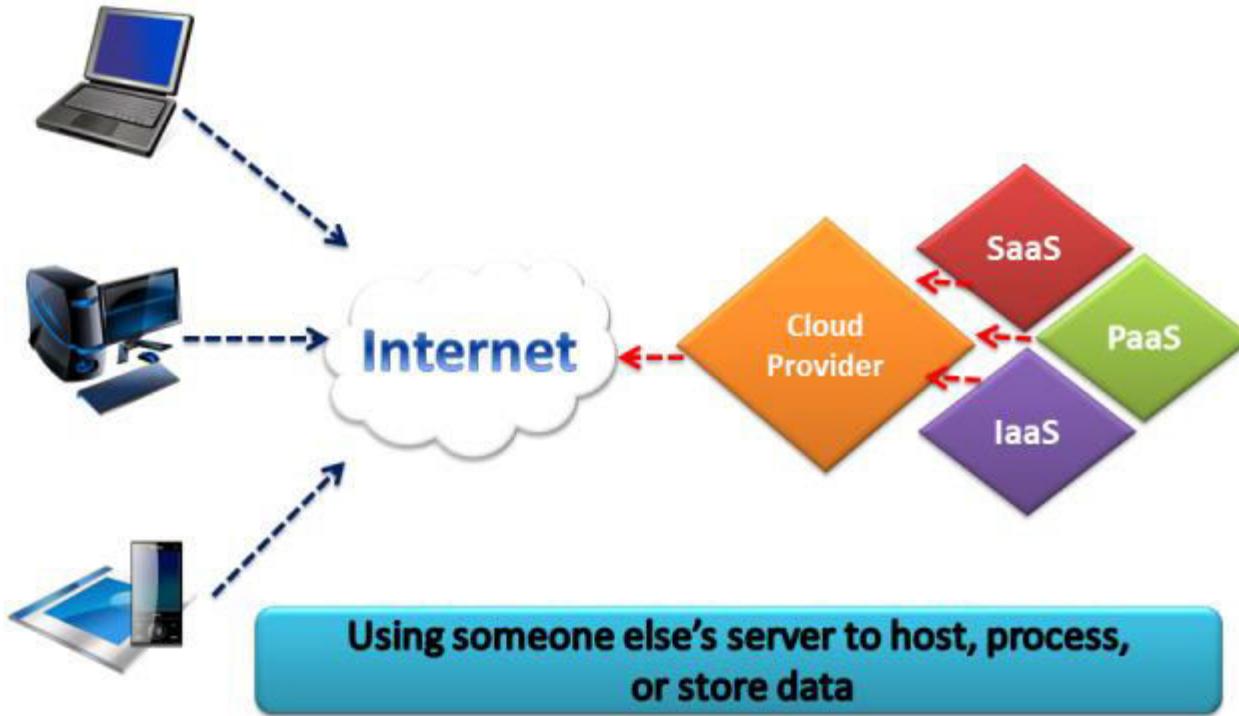
Cloud computing is a paradigm in software development that facilitates broader access to parallel computing via vast, virtual computer clusters, allowing the average user and smaller organizations to leverage parallel processing power and storage options typically reserved for large enterprises.

- See picture in the next slide

CLOUD COMPUTING

Parallel and Grid Computing

Cloud Computing



<https://www.dezyre.com/article/cloud-computing-vs-distributed-computing/94>

CLOUD COMPUTING

Parallel and Grid Computing

Differences between parallel and distributed computing (<https://www.geeksforgeeks.org/difference-between-parallel-computing-and-distributed-computing/>)

S.NO	PARALLEL COMPUTING	DISTRIBUTED COMPUTING
1.	Many operations are performed simultaneously	System components are located at different locations
2.	Single computer is required	Uses multiple computers
3.	Multiple processors perform multiple operations	Multiple computers perform multiple operations
4.	It may have shared or distributed memory	It have only distributed memory
5.	Processors communicate with each other through bus	Computer communicate with each other through message passing.
6.	Improves the system performance	Improves system scalability, fault tolerance and resource sharing capabilities

Grid Computing (T2)

Cloud computing is frequently compared to grid computing.

There are very specific differences between a grid computing infrastructure and cloud computing infrastructure

The vision of grid computing is to enable **computing to be delivered as a utility**

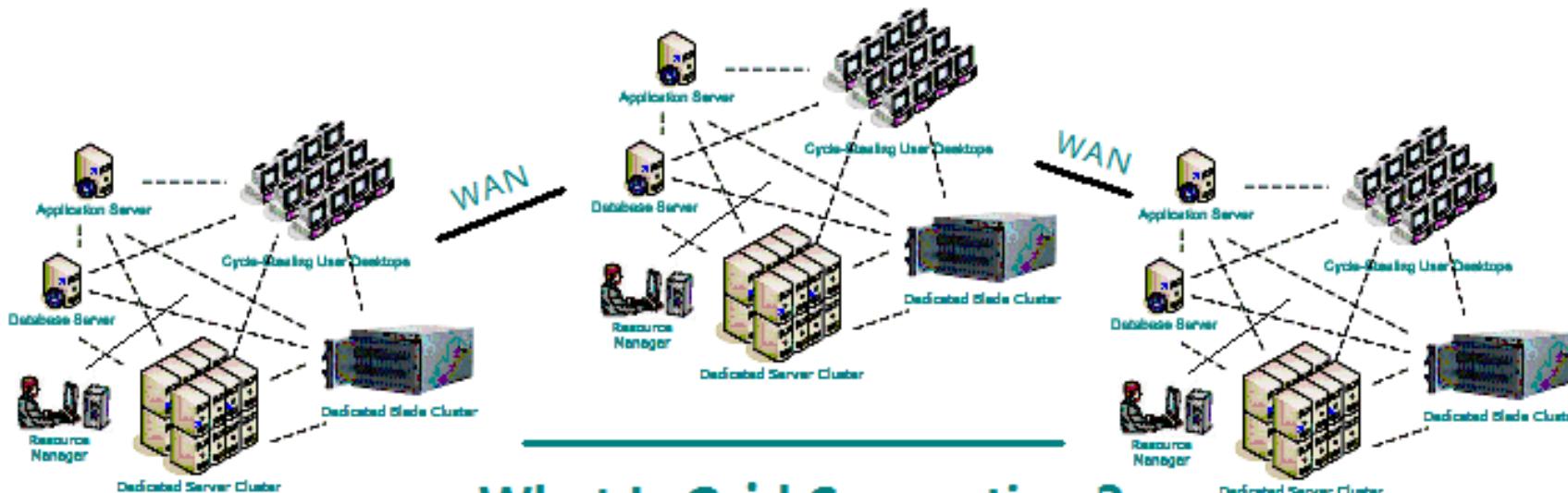
grid computing was meant to be used by individual users who gain access to computing devices **without knowing** where the **resource is located or what hardware it is running**, and so on.

Initial technological focus of grid computing was **limited to enabling shared use of resources with common protocols for access**

A grid is a system that – **Features of a Grid**

1. Co-ordinates resources that are not subject to centralized control
2. Using standard, open, general purpose protocols and interfaces
3. To deliver nontrivial quality of service

Use of a common standard for authentication, authorization, resource discovery and resource access becomes a necessity



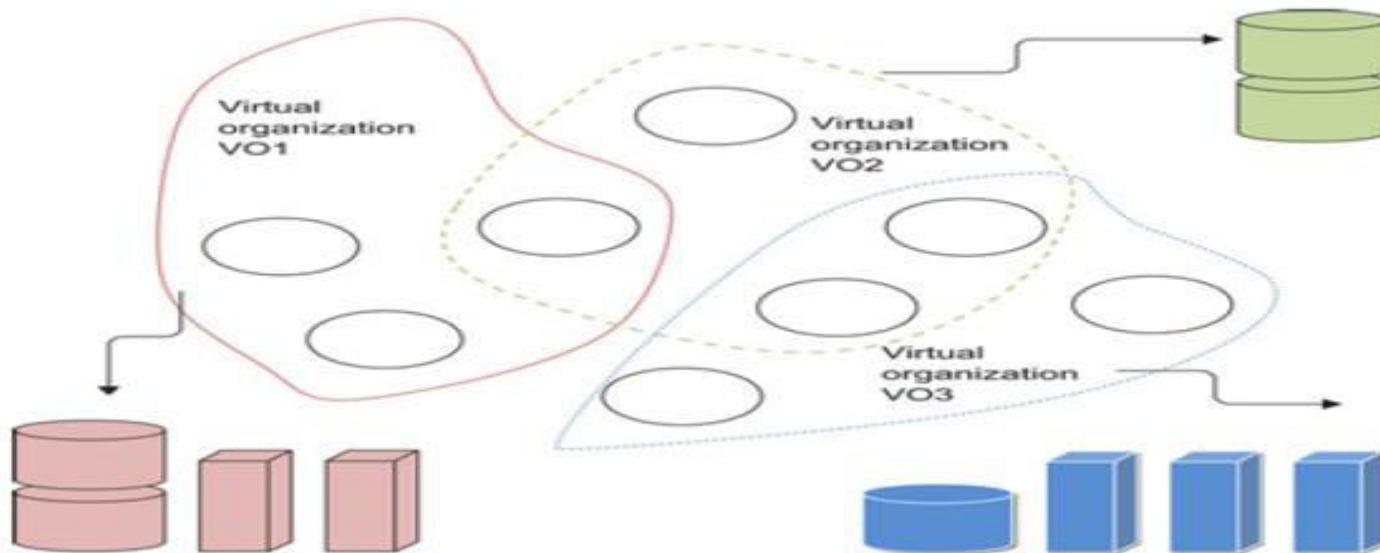
What Is Grid Computing ?

- Particular emphasis was given to **handle heterogeneous infrastructure**
- A **software-only solution** was proposed (**Globus**) and implemented on this heterogeneous infrastructure to enable use of these resources for higher computing needs.
- Establishing **trust and security models** between infrastructure resources pooled from two different administrative domains became even more important.
- Grid computing is defined as "*coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations.*"
- **Resource sharing agreements** needed to be formed among a set of participating parties where sharing meant **DIRECT ACCESS** to the resources.
- Sharing was highly controlled with **resource providers and consumers grouped into virtual organizations** primarily based on sharing conditions.

Virtual Organization:(T2)

- Enables flexible, coordinated, secure resource sharing among participating entities.
- A **virtual organization (VO)** is basically a dynamic collection of individuals or institutions from multiple administrative domains
- A VO forms a **basic unit for enabling access to shared resources** with specific resource-sharing policies applicable for users from a particular VO

Grid Technology enables sharing of resource between parties who may not have any trust earlier



Benefits

- Exploit Underutilized resources
- Resource load Balancing
- Virtualize resources across an enterprise
- Data Grids, Compute Grids
- Enable collaboration for virtual organizations

“A computational grid is a **hardware and software infrastructure** that provides dependable, consistent, pervasive, and inexpensive access to **high-end computational capabilities.**”

“The Grid: Blueprint for a New Computing Infrastructure”, Kesselman & Foster

Example : Science Grid (US Department of Energy)

A **data grid** is a grid computing system that deals with data — the **controlled sharing and management of large amounts of distributed data.**

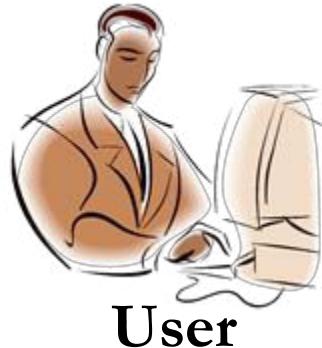
Data Grid is the storage component of a grid environment. Scientific and engineering applications require access to large amounts of data, and often this data is widely distributed. A data grid provides seamless access to the local or remote data required to complete compute intensive calculations.

Example :

Biomedical informatics Research Network (BIRN),
the Southern California earthquake Center (SCEC).

A typical view of Grid environment

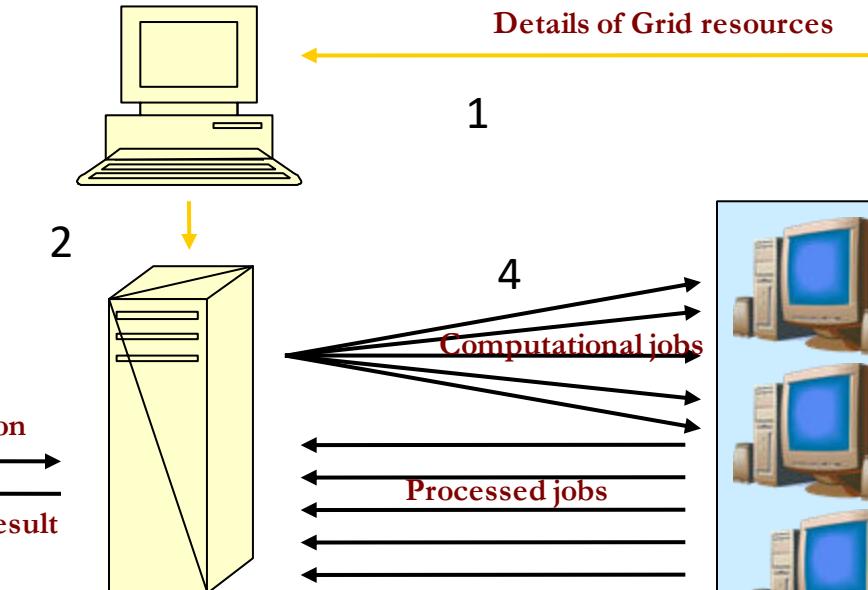
Grid Information Service
system collects the details of the available Grid resources and passes the information to the resource broker.



User

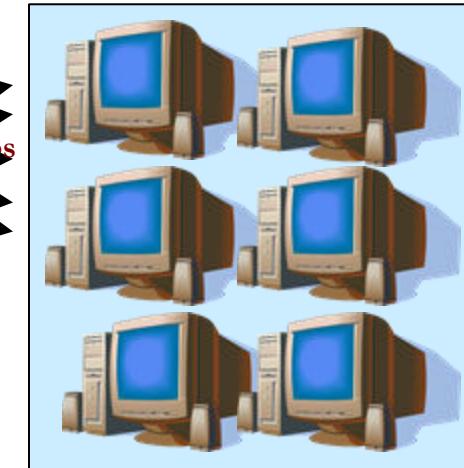
A **User** sends computation or data intensive application to Global Grids in order to speed up the execution of the application.

Grid Information Service



Resource Broker

A **Resource Broker** distribute the jobs in an application to the Grid resources based on user's QoS requirements and details of available Grid resources for further executions.



Grid Resources

Grid Resources (Cluster, PC, Supercomputer, database, instruments, etc.) in the Global Grid execute the user jobs.

Layered Grid Architecture

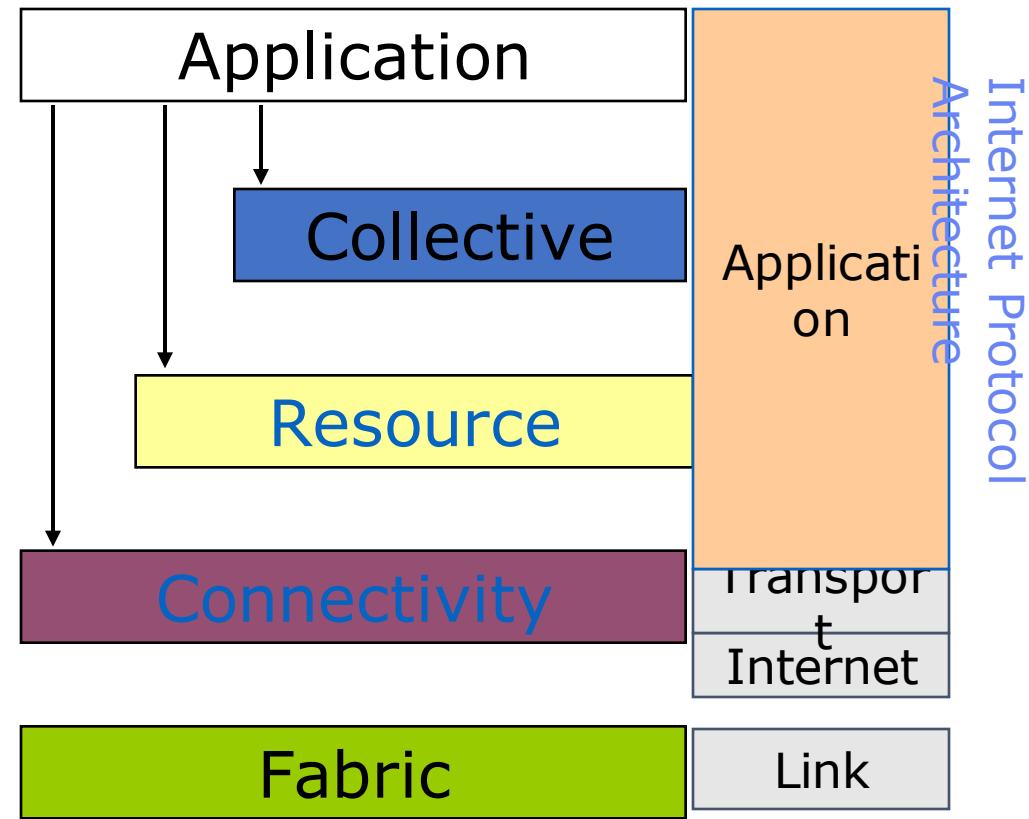
(By Analogy to Internet Architecture)

“Coordinating multiple resources”: ubiquitous infrastructure services, app-specific distributed services

“Sharing single resources”: negotiating access, controlling use

“Talking to things”: communication (Internet protocols) & security

“Controlling things locally”: Access to, & control of, resources



App	Discipline-Specific Data Grid Application
Collective (App)	Coherency control, replica selection, task management, virtual data catalog , virtual data code catalog, ...
Collective (Generic)	Replica catalog, replica management , co-allocation, certificate authorities, metadata catalogs,
Resource	Access to data , access to computers , access to network performance data, ...
Connect	Communication, service discovery (DNS) , authentication, authorization, delegation
Fabric	Storage systems, clusters, networks, network caches, ...

CLOUD COMPUTING

Parallel and Grid Computing – Grid Middleware

Grids are typically managed by grid ware -

a special type of middleware that enable sharing and manage grid components based on user requirements and resource attributes (e.g., capacity, performance)

Software that connects other software components or applications to provide the following functions:

- Run applications on suitable available resources
 - Brokering, Scheduling
- Provide uniform, high-level access to resources
 - Semantic interfaces
 - Web Services, Service Oriented Architectures
- Address inter-domain issues of security, policy, etc.
 - Federated Identities
- Provide application-level status
- monitoring and control

Middleware

- Globus –chicago Univ
- Condor – Wisconsin Univ – High throughput computing
- Legion – Virginia Univ – virtual workspaces- collaborative computing
- IBP – Internet back pane – Tennessee Univ – logistical networking
- NetSolve – solving scientific problems in heterogeneous env – high throughput & data intensive

CLOUD COMPUTING

Parallel and Grid Computing – Some Major Grid Projects

Name	URL/Sponsor	Focus
EuroGrid, Grid Interoperability (GRIP)	eurogrid.org European Union	Create tech for remote access to super comp resources & simulation codes; in GRIP, integrate with Globus Toolkit™
Fusion Collaboratory	fusiongrid.org DOE Off. Science	Create a national computational collaborative for fusion research
Globus Project™	globus.org DARPA, DOE, NSF, NASA, Msoft	Research on Grid technologies ; development and support of Globus Toolkit™; application and deployment
GridLab	gridlab.org European Union	Grid technologies and applications
GridPP	gridpp.ac.uk U.K. eScience	Create & apply an operational grid within the U.K. for particle physics research
Grid Research Integration Dev. & Support Center	grids-center.org NSF	Integration, deployment, support of the NSF Middleware Infrastructure for research & education

CLOUD COMPUTING

Parallel and Grid Computing

Cloud Computing	Grid Computing
Cloud Computing follows client-server computing architecture.	Grid computing follows a distributed computing architecture.
Scalability is high.	Scalability is normal.
Cloud Computing is more flexible than grid computing.	Grid Computing is less flexible than cloud computing.
Cloud operates as a centralized management system.	Grid operates as a decentralized management system.
In cloud computing, cloud servers are owned by infrastructure providers.	In Grid computing, grids are owned and managed by the organization.
Cloud computing uses services like IaaS, PaaS, and SaaS.	Grid computing uses systems like distributed computing, distributed information, and distributed pervasive.
Cloud Computing is Service-oriented.	Grid Computing is Application-oriented.
It is accessible through standard web protocols.	It is accessible through grid middleware.

High-Throughput Computing

- Uses the grid to schedule large numbers of loosely coupled or independent tasks, with the goal of putting **unused processor cycles to work**.

On-Demand Computing

- Uses grid capabilities to meet **short-term requirements for resources** that are not locally accessible.
- Models **real-time computing demands**.



THANK YOU

K.S.Srinivas
srinivasks@pes.edu
+91 80 2672 1983 Extn 701



CLOUD COMPUTING

CLOUD COMPUTING MODELS

K.S.Srinivas

Department of Computer Science and Engineering

CLOUD COMPUTING

Cloud Computing Models and Business Case

Srinivas K S.

Associate Professor, Department of Computer Science

CLOUD COMPUTING

Cloud Computing Models

"

Cloud computing is a model for enabling ubiquitous, convenient, on demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction

-

NIST (National Institute of Standards

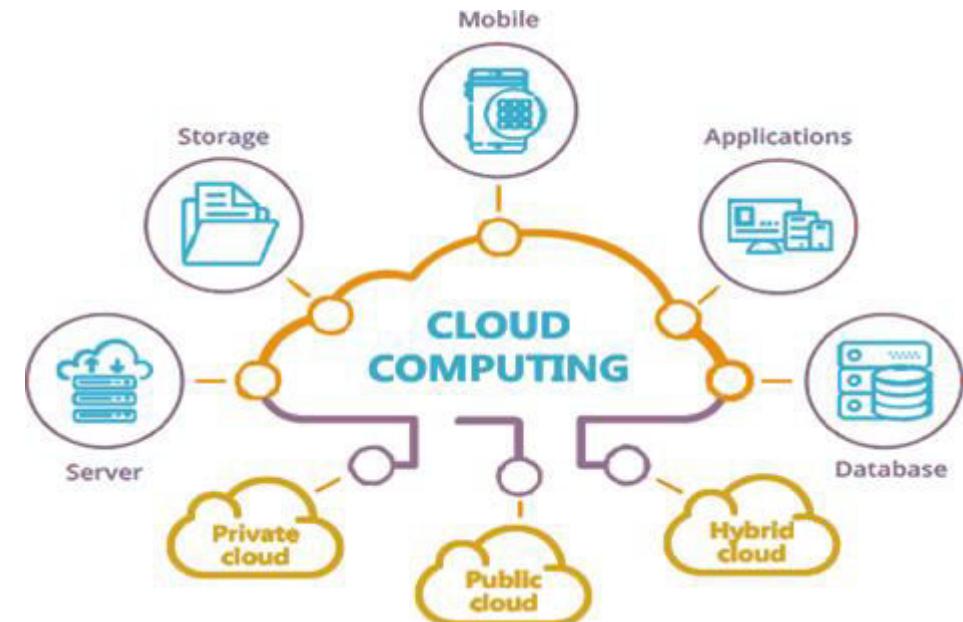
Key Terms

Ubiquitous

On Demand

Shared Pool of configurable resources

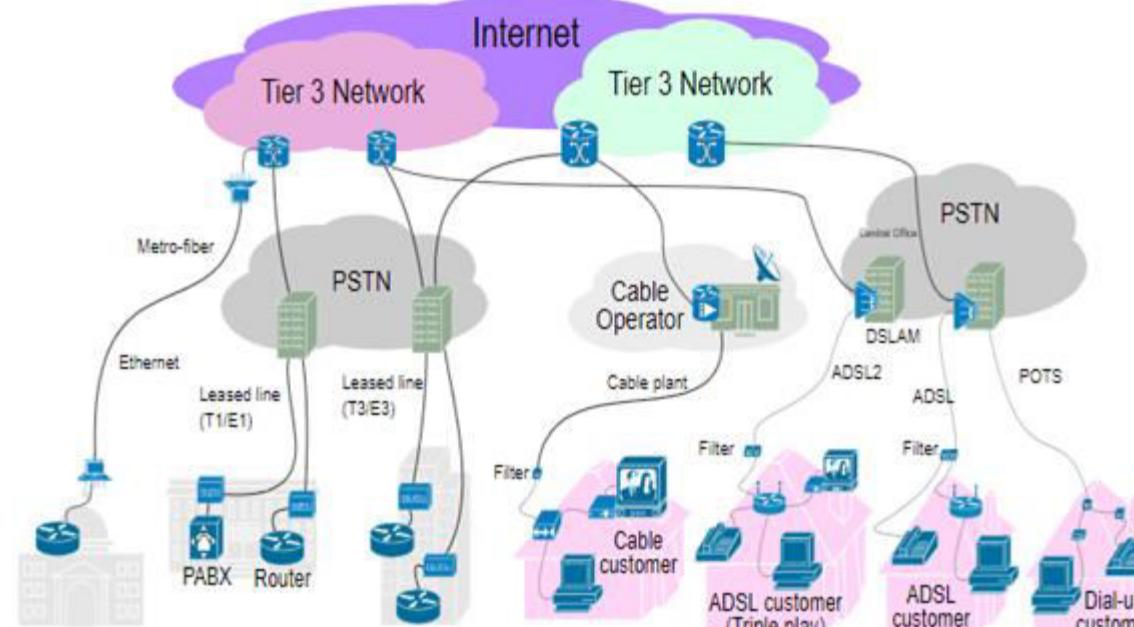
Rapidly Provisioned



Technologies that enable cloud computing

1. Broadband networks and internet architecture
 1. All clouds must be connected to a network
 2. Internet's largest backbone networks, established and deployed by ISPs, are interconnected by core routers
2. Data center technology
3. Virtualization technology
4. Web technology
5. Multitenant technology

Connectionless pack switching
Router based Interconnectivity



Standardization and Modularity

Data centers are built upon standardized commodity hardware and designed with modular architecture.

Web Technologies used in Cloud Computing

- Uniform resource locator (URL)
- Hypertext transfer protocol (HTTP)
- Markup languages (HTML, XML)



Cloud Service Models/Classification (T2)

- Infrastructure as a service
- Platform as a service
- Software as a service
- IAAS has lesser automation than SAAS
- IAAS has higher flexibility than SAAS

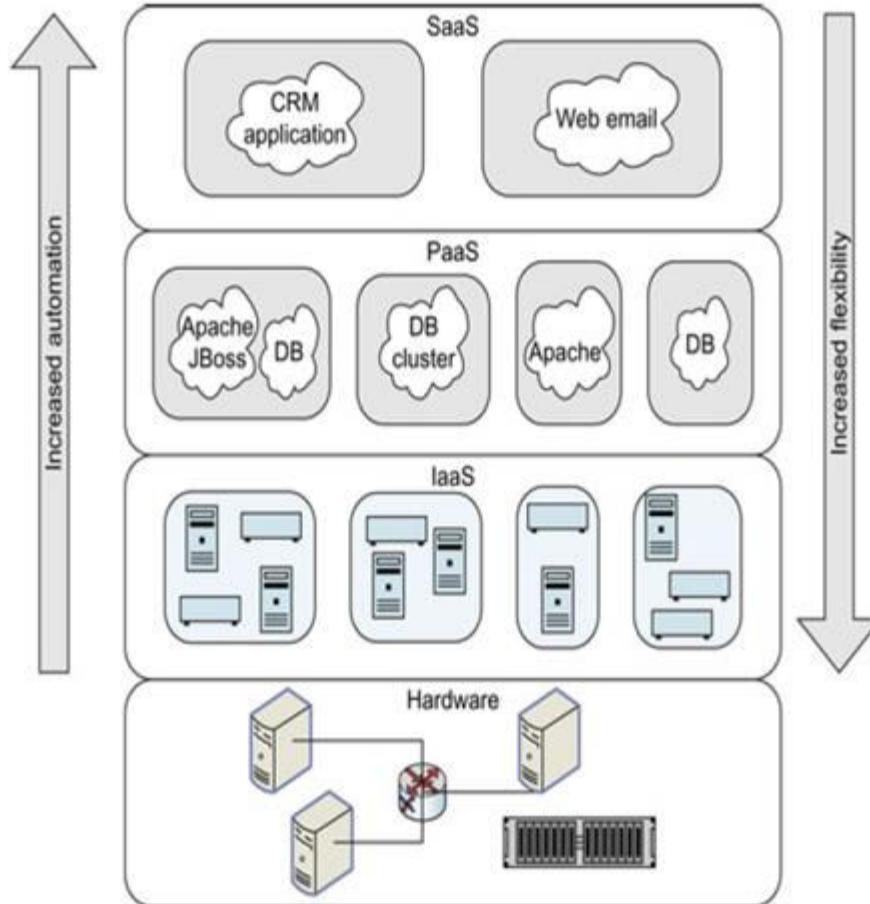


Figure 1.4 Cloud service models

Cloud Computing Models

In the IaaS service model, the physical hardware (servers, disks, and networks) is abstracted into virtual servers and virtual storage.

These **virtual resources can be allocated on demand** by the cloud users, and configured into virtual systems on **which any desired software can be installed**. As a result, this architecture has the greatest flexibility, but also the least application automation from the user's viewpoint.

Above this is the PaaS abstraction, which **provides a platform built on top of the abstracted hardware** that can be used by developers to create cloud applications

PaaS will have commands available that will allow them to **allocate middleware servers** (e.g., a database of a certain size), configure and load data into the middleware

On PaaS you can **develop an application that runs on top of the middleware**

CLOUD COMPUTING

Cloud Computing Models



Above this is the SaaS abstraction, **which provides the complete application** (or solution) as a service, enabling consumers to use the cloud **without worrying about all the complexities of hardware, OS or even application installation**

SaaS architecture has the **least flexibility and most automation** for the user.

Infrastructure as a Service

The IaaS model is about providing compute and storage resources as a service

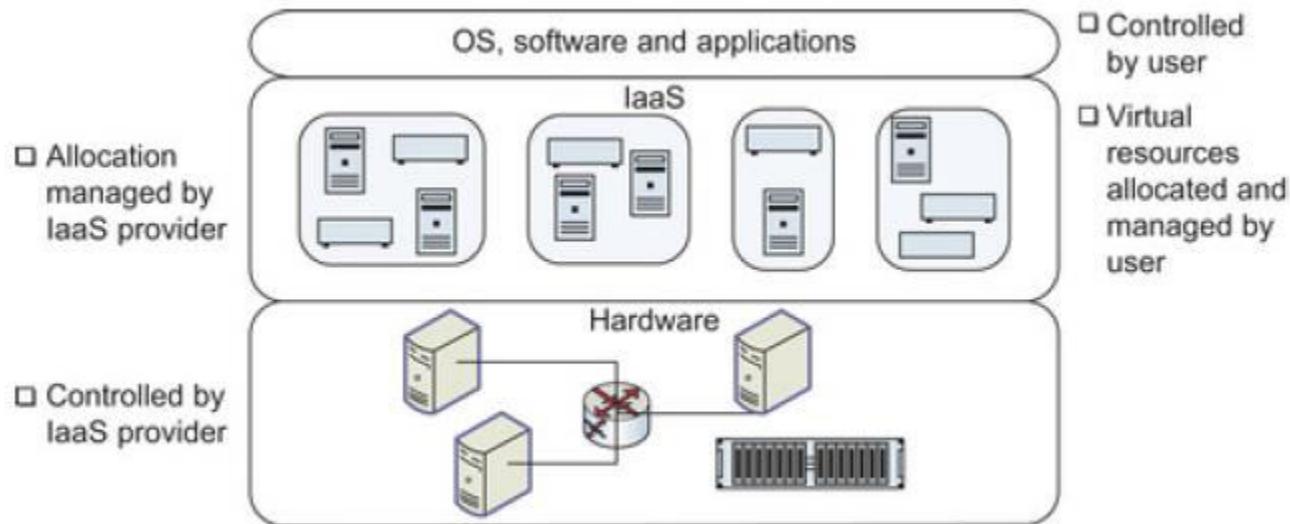
The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, deployed applications, and possibly limited control of select networking components (e.g., host firewalls). NIST Definitions

The user of IaaS has single ownership of the hardware infrastructure allotted to him (may be a virtual machine)

The IaaS provider has control over the actual hardware and the cloud user can request allocation of virtual resources

IaaS is well suited for users who want complete control over the software stack that they run

- Amazon Web Services.
- Microsoft Azure.
- Google Cloud.
- IBM Cloud.
- Oracle Cloud.



Platform as a Service (T2)

The PaaS model is to provide a **system stack or platform for application deployment** as a service

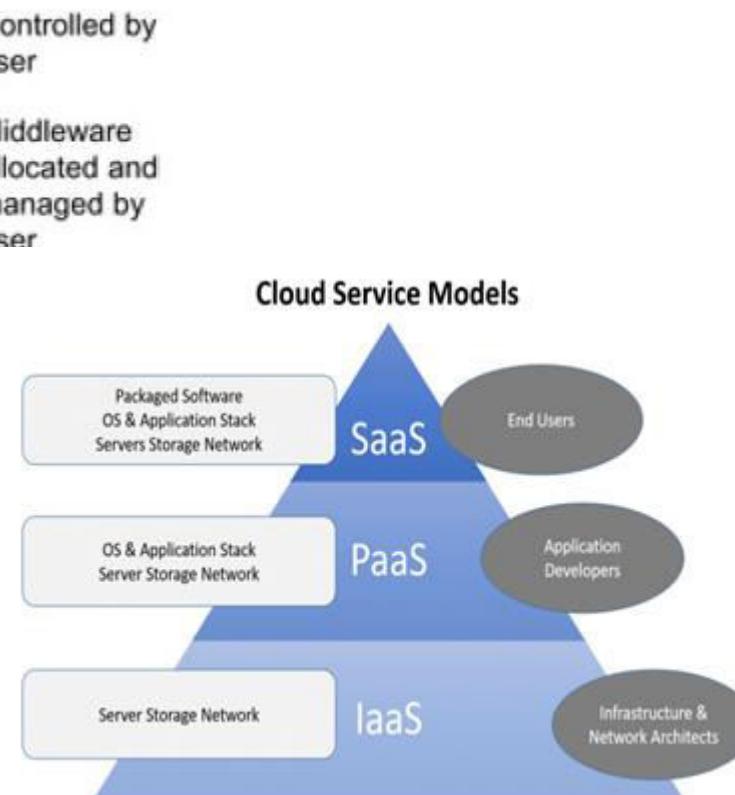
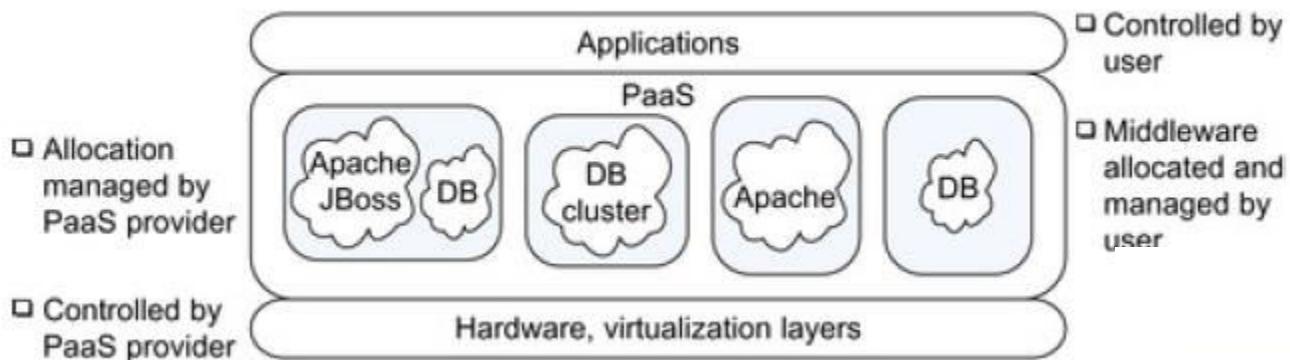
The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly application hosting environment configurations

The hardware, as well as any **mapping of hardware to virtual resources, such as virtual servers, is controlled by the PaaS provider**

The PaaS provider supports selected **middleware, such as a database, web application server**

The cloud user can configure and build on top of this middleware, such as define a new database table in a database

PaaS platforms are well suited to those cloud users who find that the middleware they are using matches the middleware provided by one of the PaaS vendors



Cloud Computing Models

Software as a Service (T2)

SaaS is about providing the complete application as a service

The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through a thin client interface such as a web browser (e.g., web-based email). The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings.

Users who log into the SaaS service can both use the application as well as configure the application for their use

When configuration settings are changed, the SaaS infrastructure performs any management tasks needed (such as allocation of additional storage) to support the changed configuration

SaaS is almost a no programming Model

Small amount of scripting and programming is usually available for advanced users if these are not available as configuration settings

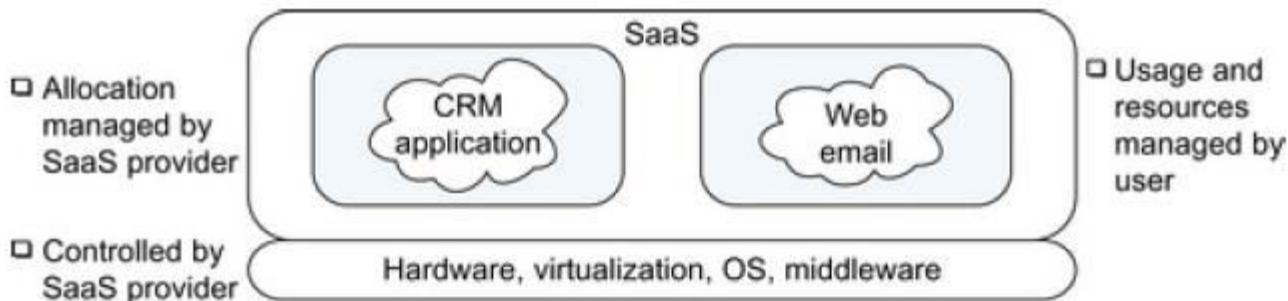


Figure 1.7 SaaS cloud model.



THANK YOU

K.S.Srinivas
srinivasks@pes.edu
+91 80 2672 1983 Extn 701



CLOUD COMPUTING

TECHNOLOGIES CHALLENGES AND BUSINESS CASE

K.S.Srinivas

Department of Computer Science and Engineering

CLOUD COMPUTING

Technology Challenges and Business Case

Srinivas K S.

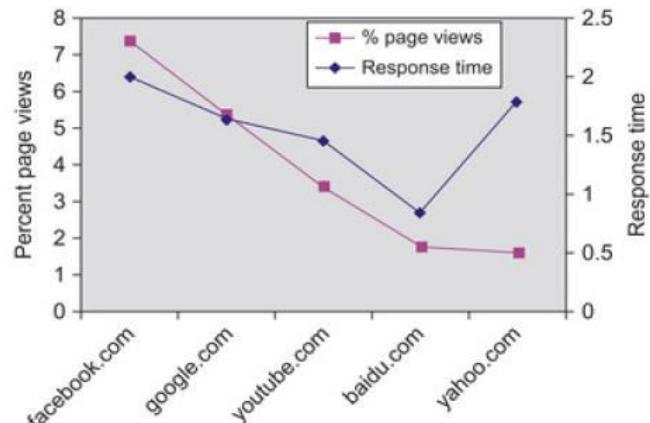
Associate Professor, Department of Computer Science

Cloud Computing has technology challenges

- on account of scale and spread
- Shared by Many users
- Much larger than traditional computing environments
- Many different types of computing environments

These challenges all the 3 computing models.

As traffic increases the environment must still maintain a acceptable response time



Elasticity

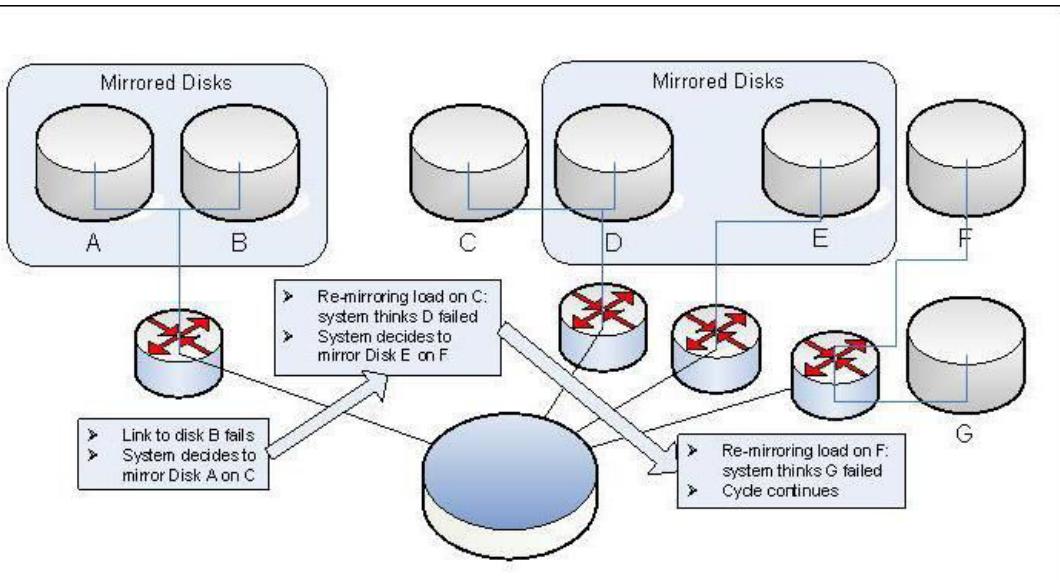
How to scale up and down quickly?

Resource allocation and workload placement algorithms are required.

Performance Unpredictability

How do we ensure reliability especially when resources are shared

Reliability



Compliance

Is cloud provider complying with privacy rules?

Health care industry requires health care compliance administrator

Sarbanes Oxley 2002

Framed in 2002 after Enron, WorldCom financial accounting scandals

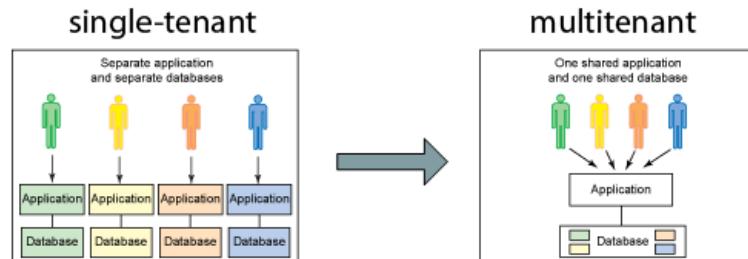
Example: Audit controls on transactions

India SEBI Clause 49 (Corporate Governance Practices)

Multi tenancy

Sharing of same database by multiple users

Share without compromising security



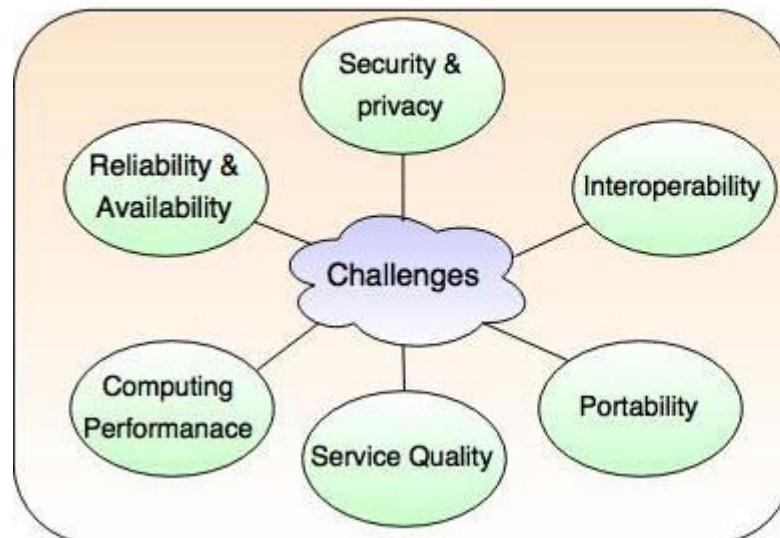
CLOUD COMPUTING

Technology Challenges and Business Case

To support such high transaction rates with good response time, it must be **possible to scale both compute and storage resources very rapidly**. Scalability of both compute power and storage is therefore a major challenge for all three cloud models.

Techniques such **multi-tenancy**, or fine-grained sharing of resources, are needed for supporting such large numbers of users(**Scalability**)

Following diagram shows the major challenges in cloud computing.



Security and Privacy (<https://www.tutorialride.com/cloud-computing/challenges-in-cloud-computing.htm>)

- Security and privacy are the main challenge in cloud computing.
- These challenges can be reduced by using security applications, encrypted file systems, data loss software.

Interoperability

- The application on one platform should be able to incorporate services from the other platform. This is known as Interoperability.
- It is becoming possible through web services, but to develop such web services is complex.

Portability

- The applications running on one cloud platform can be moved to new cloud platform and it should operate correctly without making any changes in design, coding.
- The portability is not possible, because each of the cloud providers uses different standard languages for their platform.

Computing Performance

- High network bandwidth is needed for data intensive applications on cloud, this results in high cost.
- In cloud computing, low bandwidth does not meet the desired computing performance.

Reliability and Availability

Most of the businesses are dependent on services provided by third-party, hence it is mandatory for the cloud systems to be reliable and robust.

In large-scale environments, hardware failures and software bugs can be expected to occur relatively frequently.

The problem is complicated by the fact that failures can trigger other failures, leading to an avalanche of failures that can lead to significant outages.

Availability

Service is important and many businesses run mission critical applications on the cloud

Availability is often achieved by using redundancy at the infrastructure , middleware or at the application level

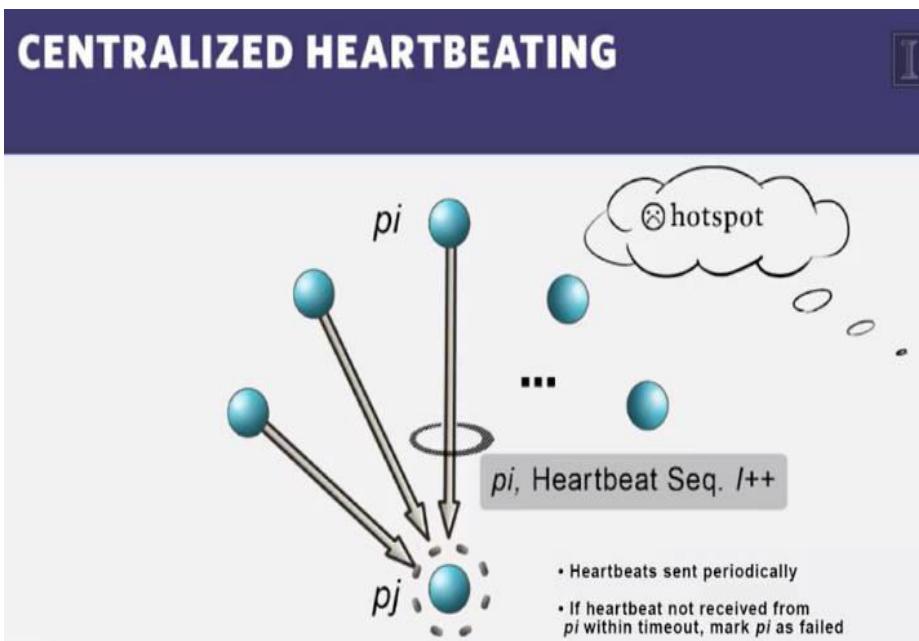
There are 2 techniques that cloud models use for ensuring high availability

1. The first technique is **failure detection**, where the cloud infrastructure detects failed application instances, and avoids routing requests to such instances
2. The second technique is **application recovery**, where failed instances of application are restarted.

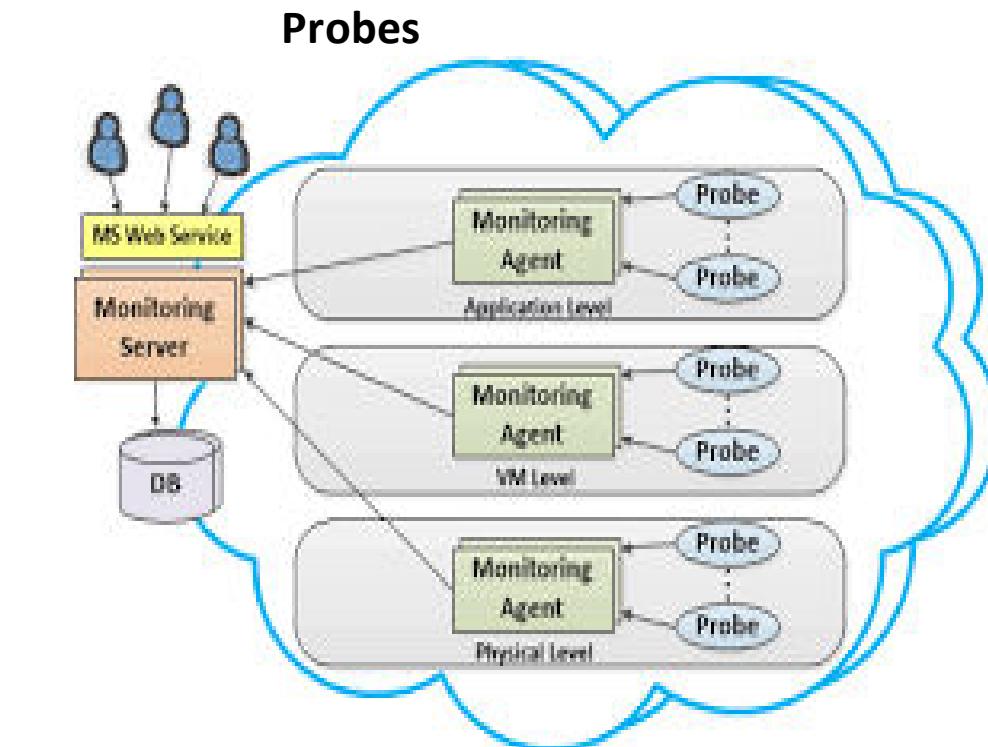
Failure Detection (T2)

Failure Monitoring

Heart Beats



<https://medium.com/@taka.atsushi/coursera-cloud-computing-concepts-part-1-538df331ff41>



https://www.researchgate.net/figure/Abstract-Architecture-View-monitoring-metrics-from-the-Cloud-element-they-reside-on-and-fig1_271428939

There are two techniques of **failure monitoring** [T2].

The first method is **heartbeats**, where each application instance periodically sends a signal (called a heartbeat) to a monitoring service in the cloud. If the monitoring service does not receive a specified number of consecutive heartbeats, it may declare the application instance as failed.

The second is the method of **probes**. Here, the monitoring service periodically sends a probe, which is a lightweight service request, to the application instance. If the instance does not respond to a specified number of probes, it may be considered failed.

There is a trade-off between speed and accuracy of detecting failures. To detect failures rapidly, it may be desirable to set a low value for the number of missed heartbeats or probes.

Redirection: After identifying failed instances, it is necessary to avoid routing new requests to these instances. A common mechanism used for this in HTTP-based protocols is HTTP-redirection.

Application Recovery

In addition to directing new requests to a server that is up, it is necessary to recover old requests

An application independent method of doing this is **checkpoint/restart**. Here, the cloud infrastructure periodically saves the state of the application

If the application is determined to have failed, the most recent checkpoint can be activated, and the application can resume from that state.

Checkpointing is also present in certain middleware such as Dockers.

Global and local snapshots are taken from which recovery is performed. There are off course challenges associated with Checkpointing.

Potential benefits of Cloud Computing

- Operational
 - Speed to Market
 - Scalability
 - Agility
 - Highly Automated
 - Flexibility
- Financial
 - Pay per use
 - Lower Capital Investment
 - Reduced financial risk
 - Reduced Cost
 - Highly Automated

<http://www.saasblogs.com/2008/12/01/demystifying-the-cloud-where-do-saas-paas-and-other-acronyms-fit-in/>

Inhibitors to adopting cloud computing

- Security
- Compliance
- Interoperability and Vendor Lock-in

Deployment Models

Private Clouds

- Built for a single enterprise

Community Clouds

- Infrastructure shared by a community

Public Cloud

- Owned by a service provider, both public and commercial are serviced

Hybrid Cloud

- Mix of the above

CLOUD COMPUTING

Technology Challenges and Business Case

Comparison of public vs private cloud

Table 1.1 Hypothetical Cost of Public Vs Private Cloud

(in USD)	Private Cloud			Public Cloud		
	Year 1	Year 2	Year 3	Year 1	Year 2	Year 3
Hardware	70,000	40,000	20,000			
Setup Costs	30,000			5,000		
Software (Licensing)	200,000	400,000	700,000			
Labor costs	200,000	200,000	200,000			
Service costs				300,000	600,000	1,000,000
WAN costs				15,000	30,000	56,000
Cost for year	500,000	640,000	920,000	305,000	600,000	1,000,000
Total	2,060,000			2,006,000		

CLOUD COMPUTING

Technology Challenges and Business Case



CLOUD COMPUTING

Technology Challenges and Business Case



CLOUD COMPUTING

Technology Challenges and Business Case





THANK YOU

K.S.Srinivas
srinivasks@pes.edu
+91 80 2672 1983 Extn 701



CLOUD COMPUTING

Public and Private Cloud

K.S.Srinivas

Department of Computer Science and Engineering

CLOUD COMPUTING

Public and Private Clouds

Srinivas K S.

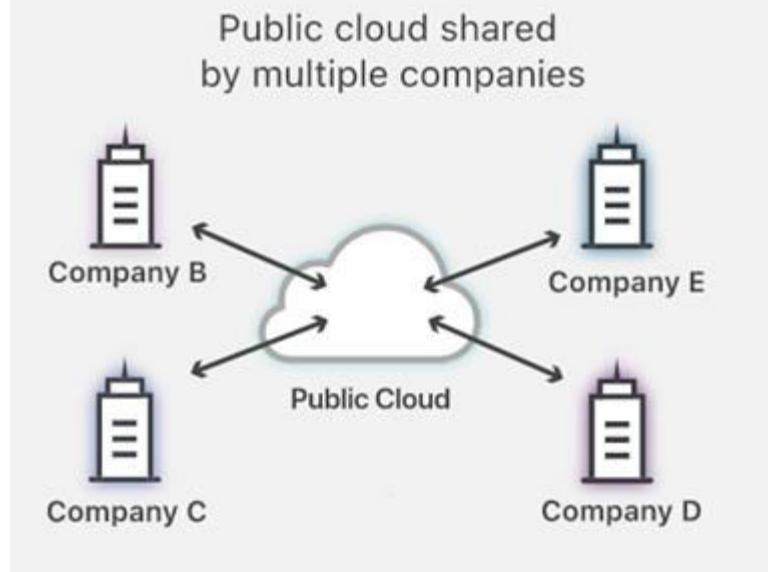
Associate Professor, Department of Computer Science

CLOUD COMPUTING

Public Cloud

So, what is the ‘public cloud’?

The infrastructure for the public cloud is owned by the cloud vendor. The cloud user pays the cloud vendor for using the infrastructure. Like all cloud services, a public cloud service runs on remote servers that a provider manages. Customers of that provider access those services over the Internet. In a public cloud, a pool of virtual resources is automatically provisioned and allocated among multiple clients through a self-service interface.



But what makes the public cloud, ‘public’?

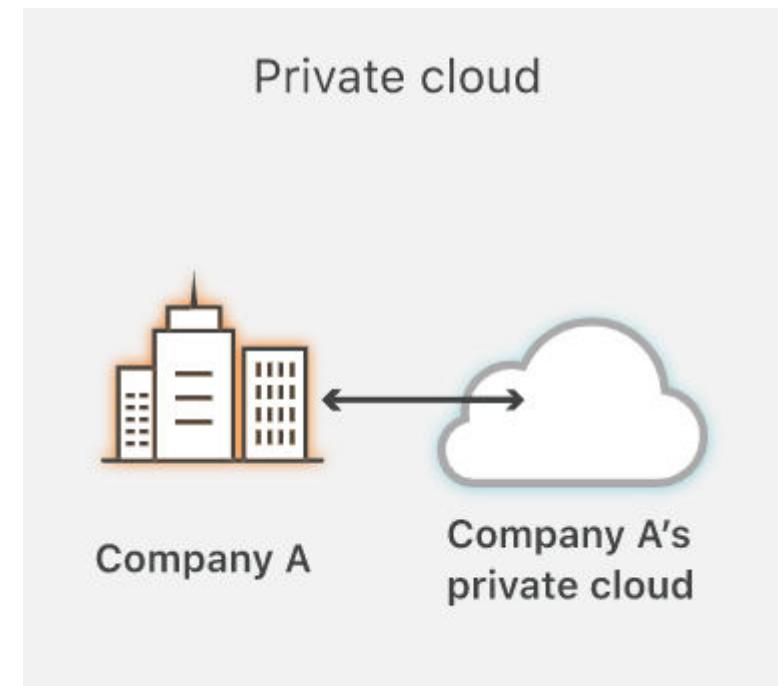
1. **Resource Allocation** - Tenants outside the provider's firewall share cloud services and virtual resources that come from the provider's set of infrastructure, platforms, and software.
2. **Usage Agreements** - While resources are distributed on an as-needed basis, a pay-as-you-go model isn't a necessary component. Some customers use public clouds at no cost (Example: Massachusetts Open Cloud).
3. **Management** - At a minimum, the provider maintains the hardware underneath the cloud, supports the network, and manages the virtualization software.

CLOUD COMPUTING

Private Cloud

What about private cloud?

The private cloud model utilizes the in-house infrastructure to host the different cloud services. Private cloud is a computing model that offers a proprietary environment dedicated to a single business entity. It is also referred to as an internal or a corporate cloud. A private cloud strategy may be comprised of hardware hosted locally at a facility owned by a business, or it may be hosted by a cloud service provider. Virtual private clouds are typically paid for on a rolling basis, but provisioned hardware and storage configurations maintain the benefits of a secure, exclusive network.



CLOUD COMPUTING

Internal vs Hosted Private Cloud

Internal Private Cloud

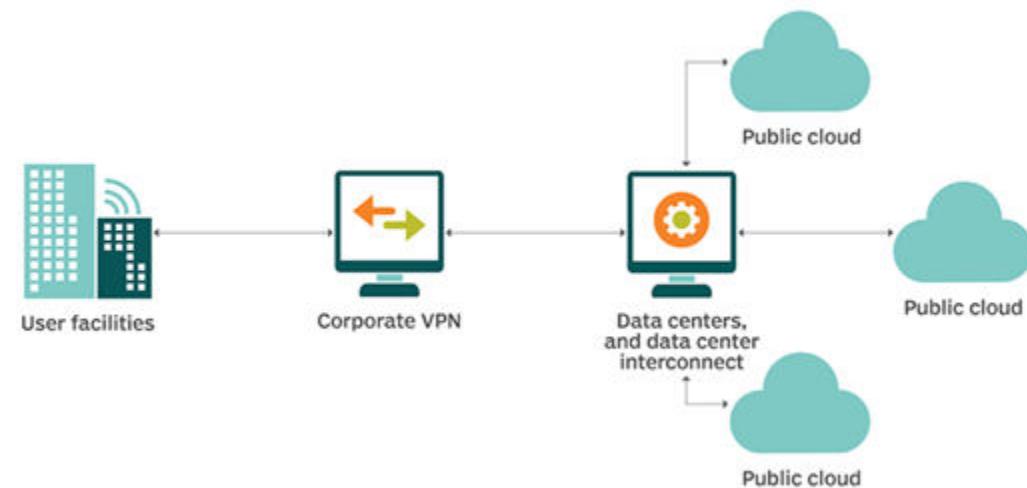


Hosted Private Cloud



Can we have the best of both worlds?

A hybrid cloud is a cloud computing environment that uses a mix of on-premises, private cloud and third-party, public cloud services with orchestration between these platforms. This typically involves a connection from an on-premises data center to a public cloud. A hybrid cloud model allows enterprises to deploy workloads in private IT environments or public clouds and move between them as computing needs and costs change. This gives a business greater flexibility and more data deployment options. A hybrid cloud workload includes the network, hosting and web service features of an application.



Advantages of Public Cloud

What motivates organizations to use public cloud?

1. **Low Cost** - Public cloud has a lower cost than private, or hybrid cloud, as it shares the same resources with a large number of consumers. You are essentially outsourcing these costs to a third party who can handle them more efficiently.
2. **Less server management** - If an organization uses a public cloud, internal teams don't have to spend time managing servers – as they do for legacy on-premises data centers or for internal private clouds.
3. **Time Saving** - Since the cloud service provider is responsible for the management and maintenance of data centers, the client can save the time required to establish connectivity, deploy new products, release product updates, configure servers etc.
4. **Analytics** - Public cloud services can perform analytics on high volumes and accommodate a variety of data types to present business insights.
5. **Virtually unlimited scalability** - Cloud capacity and resources rapidly expand to meet user demands and traffic spikes. Public cloud users also achieve greater redundancy and high availability due to the providers' various, logically separated cloud locations.

Advantages of Private Cloud

What motivates organizations to use private cloud?

1. **More Control** - Private clouds have more control over their resources and hardware than public clouds because it is only accessed by a single organization.
2. **Security and compliance** - For businesses operating in heavily regulated industries, compliance is paramount. Private cloud infrastructure gives organizations the ability to comply with strict regulations because sensitive data is held on hardware that cannot be accessed by anyone else.
3. **Customization** - Private clouds are fully configurable by the organizations using the solution. A fully private cloud is constructed by an on-site cloud architect, which means stakeholders can specify the exact environment needed to run proprietary applications. Hosted private clouds offer the same advantages but require no on-site setup.

Disadvantages of Public Clouds

1. **Security** - Verification of the security of data arises as a concern in public clouds, since the data is not being stored by the enterprise. Cloud service providers have attempted to address this problem by acquiring third-party certification.
2. **Compliance** - Many companies might question if the cloud provider is complying with the security rules relating to data. Cloud service providers have attempted to address these issues through certification as well.
3. **Interoperability and vendor lock-in** - once a particular public cloud has been chosen, it would not be easy to migrate away, since the software and operating procedures would all have been tailored for that particular cloud.

Disadvantages of Private Clouds

Disadvantages of Private Clouds

1. **Cost** - With exclusivity comes increased cost. If you plan to build your own private cloud, you face a large capital outlay.
2. **Under-utilisation** - With a private cloud, the cost of capacity underutilization is a cost to you, not to your provider. Therefore managing and maximising utilisation becomes your concern.
3. **Platform scaling** - Large upward changes in your requirements are likely to require scaling of the physical infrastructure. This is fine but may take longer than simply scaling a virtual machine within existing capacity.

Public Cloud vs Private Cloud

Private cloud vs. public cloud

Think of public cloud computing as being like a laundromat. Typically, a laundromat has enough machines for everyone to do the laundry they need, even though it's shared by multiple strangers.

A private cloud is like a laundromat that belongs to just one person, and only that person has access to it. In this way, the owner can run as many loads of laundry as they need and be assured that no one else has access to their laundry.

Private clouds and public clouds both use cloud technologies like virtualization and share characteristics such as scalability and broad access. The main difference between them is that a public cloud can be accessed by multiple customers of the cloud vendor, while a private cloud is only accessible to one organization.

CLOUD COMPUTING

Public Cloud vs Private Cloud

What factors influence whether an organization use a private cloud or public cloud?

1. **Infrastructure** - The private cloud model utilizes the in-house infrastructure to host the different cloud services. The cloud user here typically owns the infrastructure. The infrastructure for the public cloud on the other hand, is owned by the cloud vendor. The cloud user pays the cloud vendor for using the infrastructure. On the positive side, the public cloud is much more amenable to provide elasticity and scaling-on-demand since the resources are shared among multiple users
2. **Network bandwidth constraints and cost** - Disruptions in the connectivity between the client and the cloud service will affect the availability of cloud-hosted applications. On a low bandwidth network, the user experience for an interactive application may also get affected. If the storage is intended to be used for a longer term, then it may be more cost-effective to buy storage and compute and use it as a private cloud. Thus, it can be seen that one of the factors dictating the use of a private cloud or a public cloud for storage is how long the resources are needed.
3. **Control and Security** - Some businesses may prefer to use a private cloud, especially if they have extremely high security standards. Using a private cloud eliminates intercompany multitenancy (there will still be multitenancy among internal teams) and gives a business more control over the cloud security measures that are put in place.

CLOUD COMPUTING

Public Cloud vs Private Cloud

Public Cloud	Private Cloud	Hybrid Cloud
No maintenance costs	Dedicated, secure	Policy-driven deployment
High scalability, flexibility	Regulation compliant	High scalability, flexibility
Reduced complexity	Customizable	Minimal security risks
Flexible pricing	High scalability	Workload diversity supports high reliability
Agile for innovation	Efficient	Improved security
Potential for high TCO	Expensive with high TCO	Potential for high TCO
Decreased security and availability	Minimal mobile access	Compatibility and integration
Minimal control	Limiting infrastructure	Added complexity

Benefits

Drawbacks



THANK YOU

K.S.Srinivas
srinivasks@pes.edu
+91 80 2672 1983 Extn 701



CLOUD COMPUTING

Distributed System Models & Business Drivers

K.S.Srinivas

Department of Computer Science and Engineering

CLOUD COMPUTING

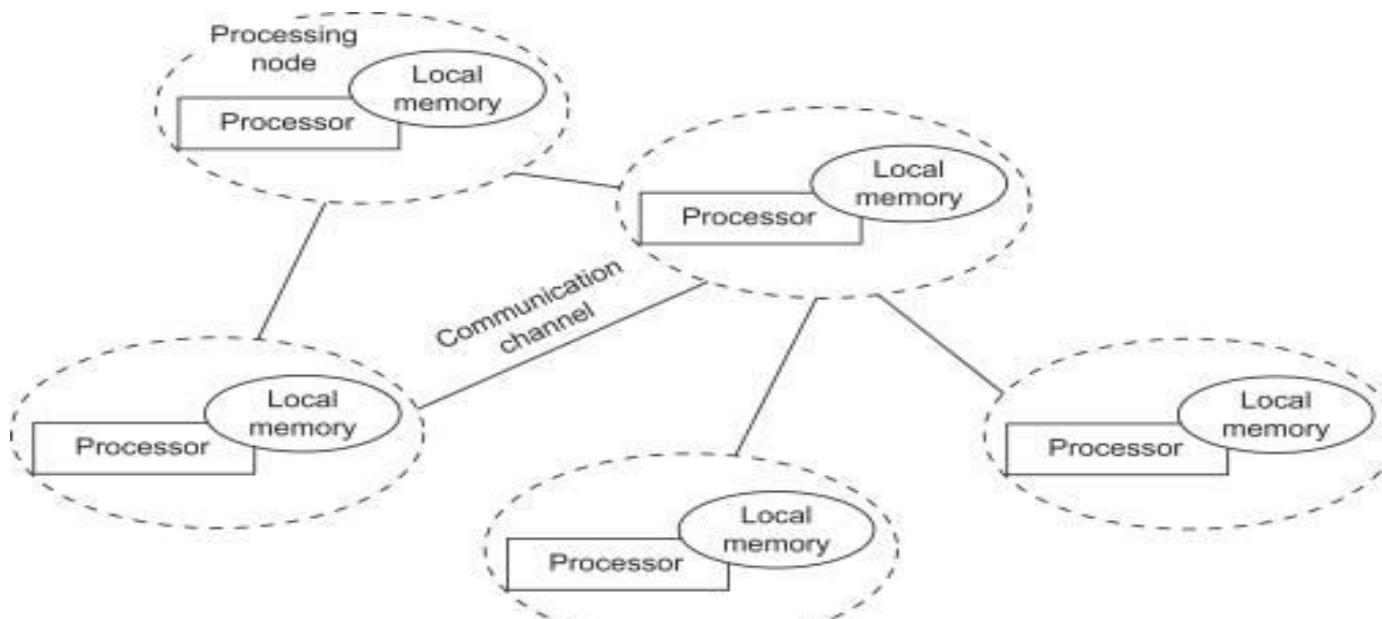
Distributed System Models

Srinivas K S.

Associate Professor, Department of Computer Science

Distributed computing

A distributed system consists of multiple autonomous computers, each having its own private memory, communicating through a computer network. Information exchange in a distributed system is accomplished through message passing. Distributed and cloud computing systems are built over a large number of autonomous computer nodes. These node machines are interconnected by SANs, LANs, or WANs in a hierarchical manner.



Distributed system can be classified as three different types of models:

1. Architectural Models

- a. How are responsibilities distributed between system components and how are these components placed?
- b. Ex: Cluster Architecture, P2P, Client-Server Model

2. Interaction Models

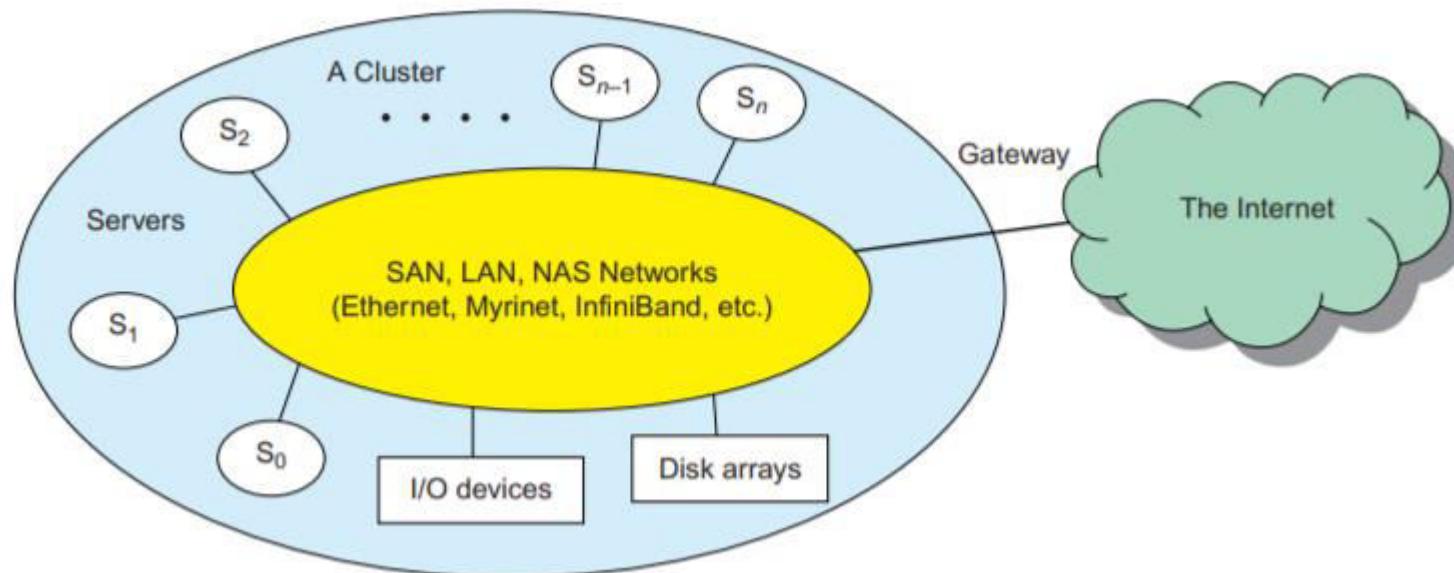
- a. How do we handle time? Are there time limits on process execution, message delivery, and clock drifts?
- b. Ex: Synchronous distributed systems, Asynchronous distributed systems

3. Fault Models

- a. What kind of faults can occur and what are their effects?
- b. Ex: Omission faults, Arbitrary faults, Timing faults

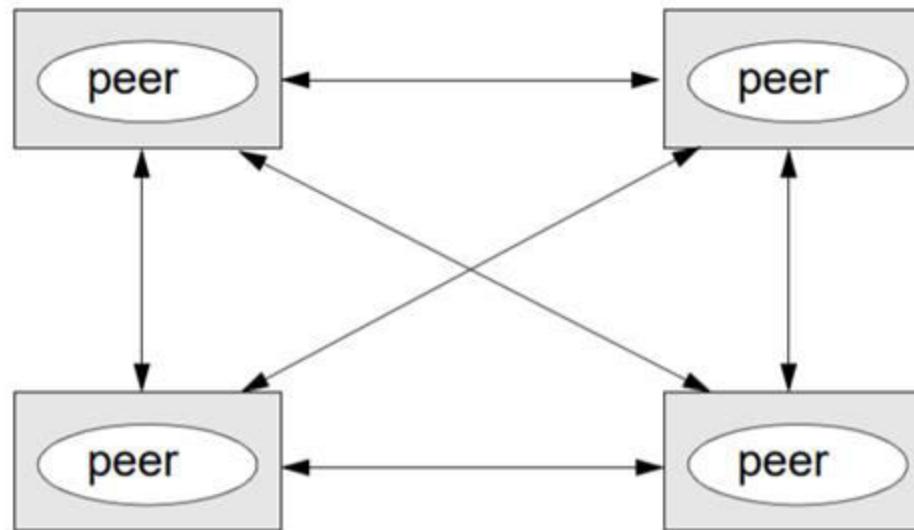
What is Cluster Architecture?

Through hierarchical construction using a SAN, LAN, or WAN, one can build scalable clusters with an increasing number of nodes. The cluster is connected to the Internet via a virtual private network (VPN) gateway. The gateway IP address locates the cluster. The system image of a computer is decided by the way the OS manages the shared cluster resources. Most clusters have loosely coupled node computers. All resources of a server node are managed by their own OS. Thus, most clusters have multiple system images as a result of having many autonomous nodes under different OS control.



What does a Peer-to-Peer System mean?

In a P2P network, every node (peer) acts as both a client and server. Peers act autonomously to join or leave the network. No central coordination or central database is needed. No peer machine has a global view of the entire P2P system. The system is self-organizing with distributed control. This implies that no master-slave relationship exists among the peers. . In other words, no peer machine has a global view of the entire P2P system. Processing and communication loads for access to objects are distributed across many computers and access links. This is the most general and flexible model.

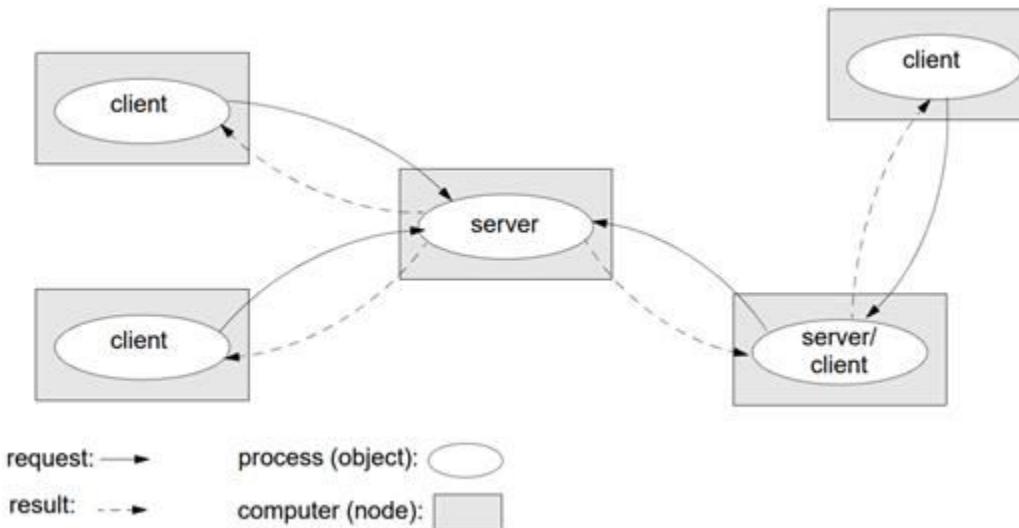


Architectural Models - Client-Server Model

What is the Client-Server model?

The system is structured as a set of processes, called servers, that offer services to the users, called clients.

The client-server model is usually based on a simple request/reply protocol, implemented with send/receive primitives or using remote procedure calls (RPC). The client asks the server for a service, the server does the work and returns a result or an error code if the required work could not be done.



Here are some other architecture models that you should know about:

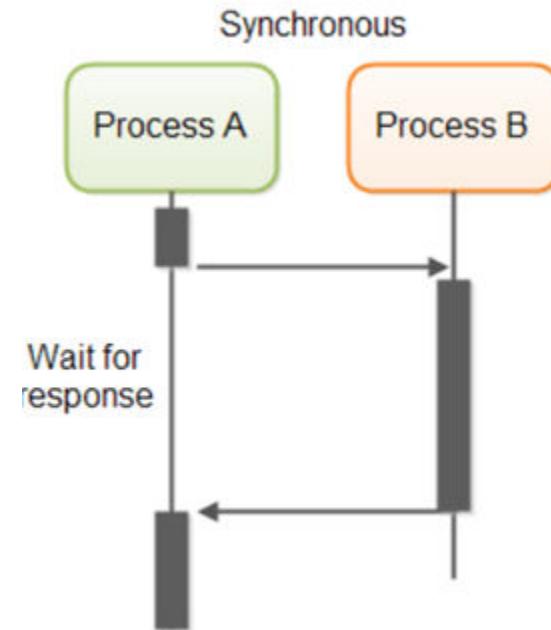
1. **Three-tier** - Architectures that move the client intelligence to a middle tier so that stateless clients can be used. This simplifies application deployment. Most web applications are three-tier.
2. **n-tier** - Architectures that refer typically to web applications which further forward their requests to other enterprise services. This type of application is the one most responsible for the success of application servers.
3. **Grid Computing Infrastructures** - A computing grid offers an infrastructure that couples computers, software/middleware, special instruments, and people and sensors together. The grid is often constructed across LAN, WAN, or Internet backbone networks at a regional, national, or global scale. Enterprises or organizations present grids as integrated computing resources. They can also be viewed as virtual platforms to support virtual organizations.

What are some features of a Synchronous Distributed System?

1. Lower and upper bounds on execution time of processes can be set
2. Transmitted messages are received within a known bounded time.
3. Drift rates between local clocks have a known bound.

What are the consequences of having a Synchronous Distributed System?

1. There is a notion of global physical time
2. Predictable in terms of timing, only such systems can be used for hard real-time applications.
3. It is possible and safe to use timeouts in order to detect failures of a process or communication link.



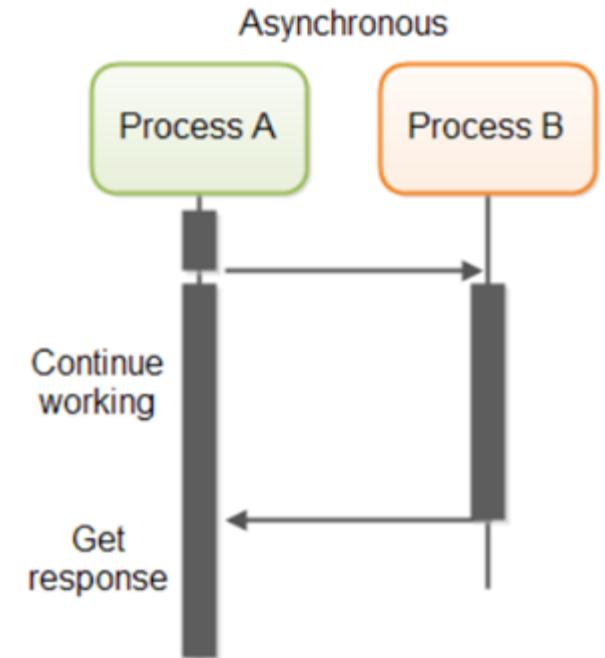
NOTE: Clock drift refers to several related phenomena where a clock does not run at exactly the same rate as a reference clock. That is, after some time the clock "drifts apart" or gradually desynchronizes from the other clock.

What are some features of an Asynchronous Distributed System?

1. No bound on process execution time (nothing can be assumed about speed, load, reliability of computers)
2. No bound on message transmission delays
3. No bounds on drift rates between local clocks.

What are the consequences of having an Asynchronous Distributed System?

1. There is no global physical time, reasoning can be only in terms of logical time.
2. Unpredictable in terms of timing.
3. Cannot use timeouts to diagnose issues



NOTE: Clock drift refers to several related phenomena where a clock does not run at exactly the same rate as a reference clock. That is, after some time the clock "drifts apart" or gradually desynchronizes from the other clock.

What is the use of a Fault Model?

1. Faults can occur both in processes and communication channels. The reason can be both software and hardware.
2. Fault models are needed in order to build systems with predictable behaviour in case of faults (systems which are fault tolerant).
3. A fault tolerant system will function according to the predictions, only as long as the real faults behave as defined by the “fault model”.

Omission and Arbitrary Failures

<i>Class of failure</i>	<i>Affects</i>	<i>Description</i>
Fail-stop	Process	Process halts and remains halted. Other processes may detect this state.
Crash	Process	Process halts and remains halted. Other processes may not be able to detect this state.
Omission	Channel	A message inserted in an outgoing message buffer never arrives at the other end's incoming message buffer.
Send-omission	Process	A process completes a <i>send</i> , but the message is not put in its outgoing message buffer.
Receive-omission	Process	A message is put in a process's incoming message buffer, but that process does not receive it.
Arbitrary (Byzantine)	Process or channel	Process/channel exhibits arbitrary behaviour: it may send/transmit arbitrary messages at arbitrary times, commit omissions; a process may stop or take an incorrect step.

Timing failures

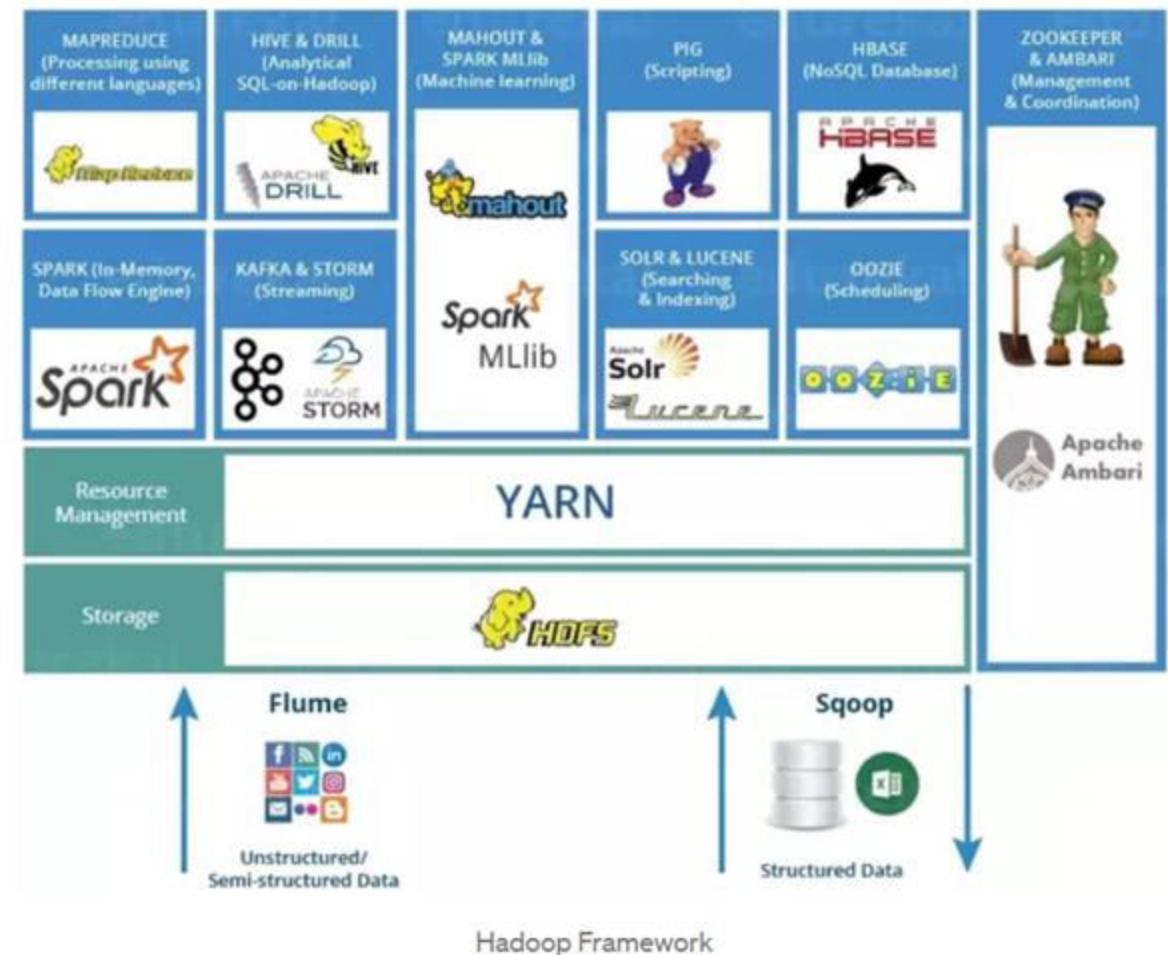
<i>Class of Failure</i>	<i>Affects</i>	<i>Description</i>
Clock	Process	Process's local clock exceeds the bounds on its rate of drift from real time.
Performance	Process	Process exceeds the bounds on the interval between two steps.
Performance	Channel	A message's transmission takes longer than the stated bound.

- Applicable in synchronous systems with set time limits
- Not applicable in asynchronous systems since no time limits can be guaranteed.

What is Hadoop?

Hadoop is an open-source, a Java-based programming framework that continues the processing of large data sets in a distributed computing environment. It based on the Google File System or GFS.

Hadoop runs few applications on distributed systems with thousands of nodes involving petabytes of information. It has a distributed file system, called Hadoop Distributed File System or HDFS, which enables fast data transfer among the nodes.



References for Distributed System Models

- [Distributed Architecture](#)
- [Distributed Computing Architectures - Wikipedia](#)
- ["Explain Distributed system models with diagram"](#)
- [Make your existing solution tastier with serverless salt: distributed system](#)
- [System Models for Distributed and Cloud Computing - UNF](#)
- [Fundamental Distributed System Models - RIT](#)
- [MODELS OF DISTRIBUTED SYSTEMS - Linköping University](#)
- [Distributed System Models](#)
- [A Brief Summary of Apache Hadoop: A Solution of Big Data Problem and Hint comes from Google](#)

CLOUD COMPUTING

Business Drivers for Cloud Computing

Srinivas K S.

Associate Professor, Department of Computer Science

CLOUD COMPUTING

Business Drivers for Cloud Computing



"If someone asks me what cloud computing is, I try not to get bogged down with definitions. I tell them that, simply put, cloud computing is a better way to run your business." - Marc Benioff, Founder, Salesforce.

Why should a business move to the cloud?

1. **Cost** - Using a cloud platform, a business does not need a very high upfront capital investment in hardware. Moreover, it is hard to estimate the amount of resources an organisation needs in its nascent stages, using a cloud platform eliminates the risk of buying more than what is needed. Often, the cost of replatforming to the cloud is lower than the license renewal of their legacy applications leading to a healthy ROI after a short period of time. Cloud computing also has the potential to reduce IT costs through automated management.
1. **Assurance** - Owning a complex infrastructure means that the organization is responsible for its maintenance and availability. Cloud provides high availability and eliminates need for an IT house in every company, which requires highly skilled administrators. Data will be more secure in the cloud and the user will attain better uptime because its solutions are maintained by providers that have built their businesses around these competencies.

3. **Agility and Innovation** - This refers to the ability of enterprise IT departments to respond quickly to requests for new services. Cloud computing, by increasing manageability, increases the speed at which applications can be deployed on public and private clouds. Cloud technologies are also easier to enhance and swap out to accommodate changing business needs.

3. **Flexibility and Scalability** - Scalability refers to the ease with which the size of the infrastructure can be increased to accommodate increased workload. If you are managing your own equipment, this means increased cost as you install new infrastructure and more costs to maintain that equipment. With cloud-based computing, you can simply purchase more storage or computing power to meet your needs and, best of all, have instant access to that expanded capability.

5. **Business Growth** - Business growth is one of the top benefits organisations realise as a result of cloud adoption, with 52% of enterprises reporting increased growth (as per 2015 Cloud Enterprise Report).
5. **Efficiency** - Efficiency is about removing unnecessary steps to streamline processes in order to increase productivity or deliver on customer requirements faster. As a result, increasing efficiency also supports business growth, by increasing worker productivity, and experience initiatives, by fulfilling customer needs faster.
5. **Experience** - Finally, an important requirement for a business is improving customer experience. Two of the most common ways organisations are achieving this goal with cloud technology are by introducing new channels of engagement and improving workplace productivity.

Downsides of Cloud Computing

What is inhibiting all business from migrating to the cloud?

1. Disadvantages of Public Clouds

- a. **Security** - Verification of the security of data arises as a concern in public clouds, since the data is not being stored by the enterprise. Cloud service providers have attempted to address this problem by acquiring third-party certification.
- b. **Compliance** - Many companies might question if the cloud provider is complying with the security rules relating to data. Cloud service providers have attempted to address these issues through certification as well.
- c. **Interoperability and vendor lock-in** - once a particular public cloud has been chosen, it would not be easy to migrate away, since the software and operating procedures would all have been tailored for that particular cloud.

Downsides of Cloud Computing

What is inhibiting all business from migrating to the cloud?

2. Disadvantages of Private Clouds

- a. **Cost** - With exclusivity comes increased cost. If you plan to build your own private cloud, you face a large capital outlay.
- b. **Under-utilisation** - With a private cloud, the cost of capacity underutilization is a cost to you, not to your provider. Therefore managing and maximizing utilisation becomes your concern.
- c. **Platform scaling** - Large upward changes in your requirements are likely to require scaling of the physical infrastructure. This is fine but may take longer than simply scaling a virtual machine within existing capacity.



THANK YOU

K.S.Srinivas
srinivasks@pes.edu
+91 80 2672 1983 Extn 701



CLOUD COMPUTING

Cloud Architecture

K.S.Srinivas

Department of Computer Science and Engineering

CLOUD COMPUTING

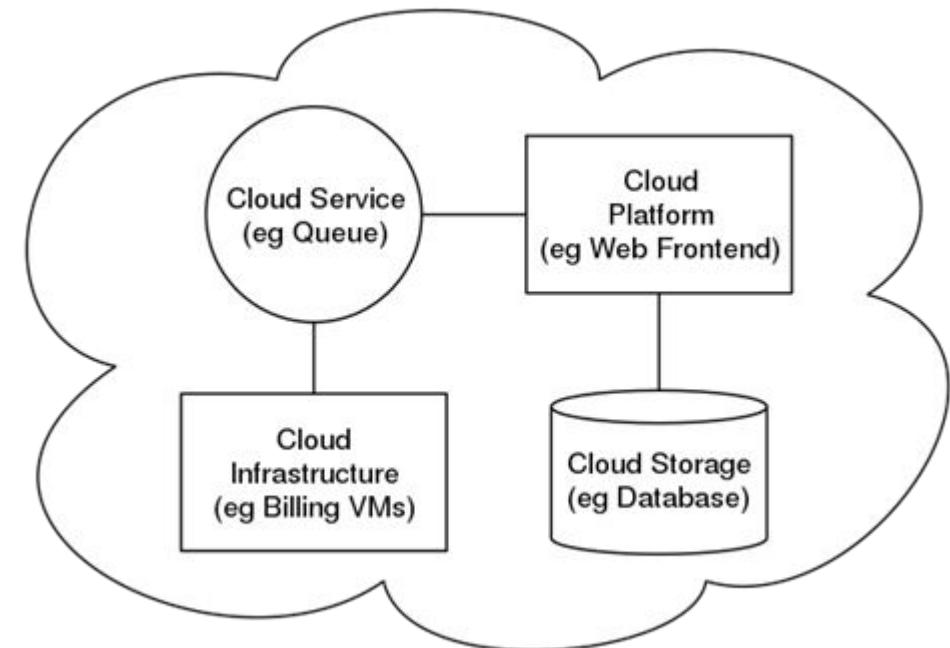
Cloud Architecture

Srinivas K S.

Associate Professor, Department of Computer Science

What is cloud architecture?

Cloud Architecture refers to the various components engineered to leverage the power of cloud resources to solve business problems. Cloud architecture defines the components as well as the relationships between them. These components typically consist of a front end platform (fat client, thin client, mobile), back end platforms (servers, storage), a cloud based delivery, and a network (Internet, Intranet, Intercloud). Combined, these components make up cloud computing architecture.

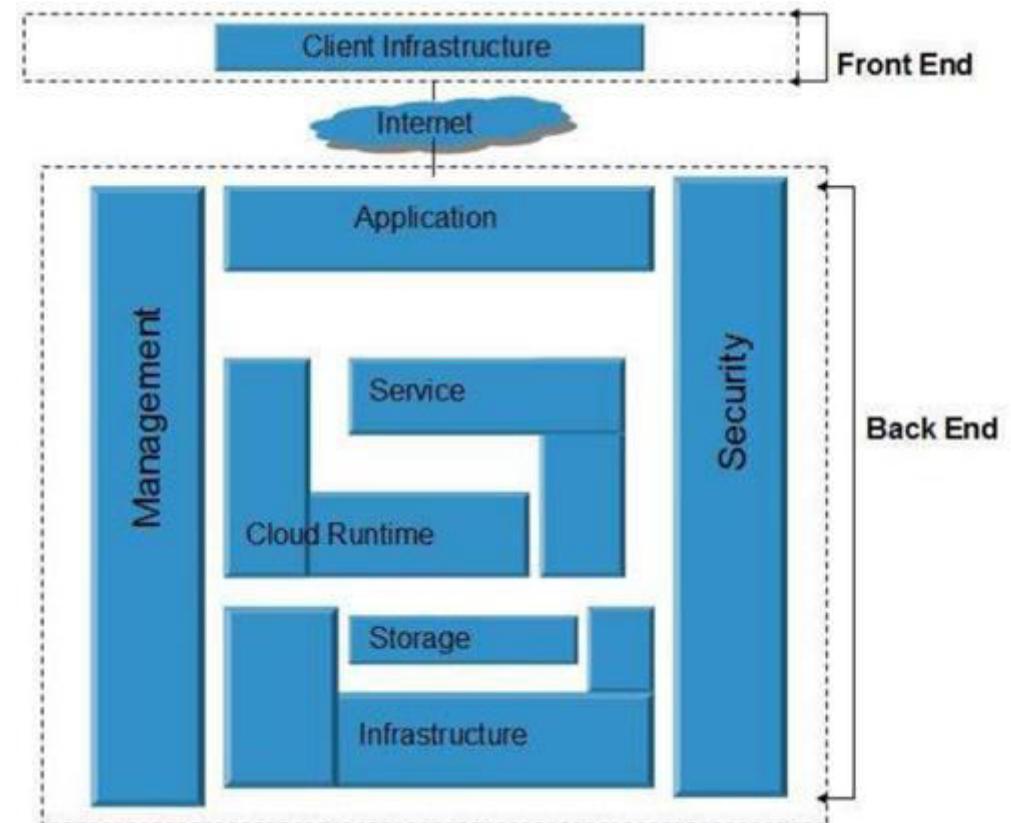


What is a front-end?

The front end is used by the client. It contains client-side interfaces and applications that are required to access the cloud computing platforms. The front end includes web servers (including Chrome, Firefox, internet explorer, etc.), thin & fat clients, tablets, and mobile devices.

What is a back-end?

The back end is used by the service provider. It manages all the resources that are required to provide cloud computing services. It includes a huge amount of data storage, security mechanism, virtual machines, deploying models, servers, traffic control mechanisms, etc.



Cloud Architecture - Components

What are some components of Cloud Architecture?

1. **Client Infrastructure** - Infrastructure of the GUI to interact with the cloud.
2. **Application** - It can either be a software or a platform
3. **Service** - Cloud can provide services at various levels. Ex: SaaS, PaaS, IaaS, DaaS
4. **Storage** - Storage is one of the most important components of cloud computing. It provides a huge amount of storage capacity in the cloud to store and manage data.
5. **Management** - Components like application, service, etc in the backend need to be managed and coordination between them needs to be established.
6. **Security** - It implements security management to the cloud server with virtual firewalls which results in preventing data loss.
7. **Internet**
8. **Hypervisor** - to implement Virtualization (more on this in Unit-2)

Note: DaaS - Data as a Service

CLOUD COMPUTING

Cloud Platform Design Goals



Scalability, virtualization, efficiency, and reliability are four major design goals of a cloud computing platform.

Cloud management receives the user request, finds the correct resources, and then calls the provisioning services which invoke the resources in the cloud. The cloud management software needs to support both physical and virtual machines.

The platform needs to establish a very large-scale HPC infrastructure. The hardware and software systems are combined to make it easy and efficient to operate. System scalability can benefit from cluster architecture. If one service takes a lot of processing power, storage capacity, or network traffic, it is simple to add more servers and bandwidth. System reliability can benefit from this architecture. Data can be put into multiple locations.

Enabling Technologies for Clouds

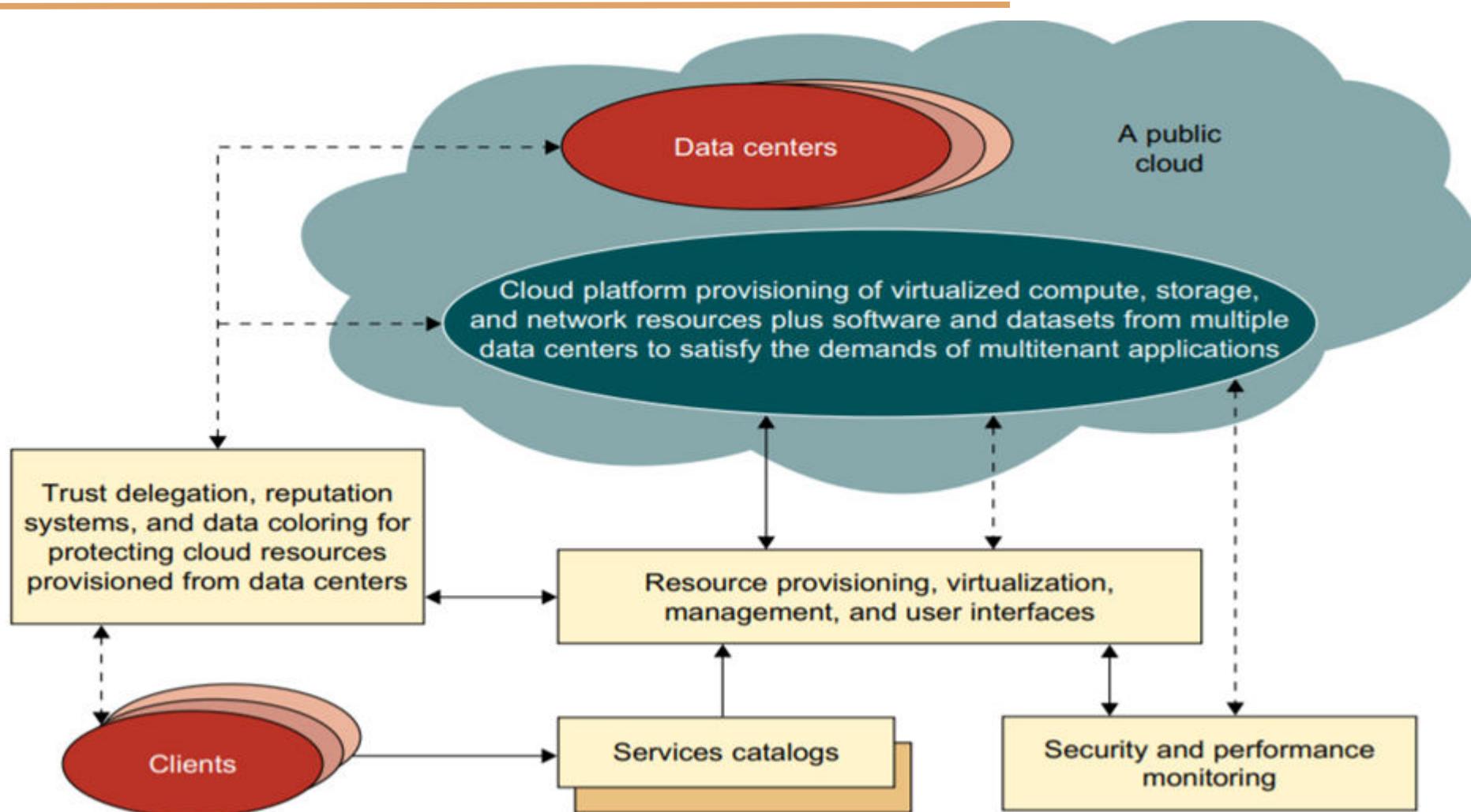
The key driving forces behind cloud computing are the ubiquity of broadband and wireless networking, falling storage costs, and progressive improvements in Internet computing software.

Table 4.3 Cloud-Enabling Technologies in Hardware, Software, and Networking

Technology	Requirements and Benefits
Fast platform deployment	Fast, efficient, and flexible deployment of cloud resources to provide dynamic computing environment to users
Virtual clusters on demand	Virtualized cluster of VMs provisioned to satisfy user demand and virtual cluster reconfigured as workload changes
Multitenant techniques	SaaS for distributing software to a large number of users for their simultaneous use and resource sharing if so desired
Massive data processing	Internet search and web services which often require massive data processing, especially to support personalized services
Web-scale communication	Support for e-commerce, distance education, telemedicine, social networking, digital government, and digital entertainment applications
Distributed storage	Large-scale storage of personal records and public archive information which demands distributed storage over the clouds
Licensing and billing services	License management and billing services which greatly benefit all types of cloud services in utility computing

CLOUD COMPUTING

Generic Cloud Architecture Example



A security-aware cloud platform built with a virtual cluster of VMs, storage, and networking resources over the data-center servers operated by providers.

Generic Cloud Architecture Example

- The figure shows a security-aware cloud architecture. The Internet cloud is envisioned as a massive cluster of servers. These servers are provisioned on demand to perform collective web services or distributed applications using data-center resources. The cloud platform is formed dynamically by provisioning or deprovisioning servers, software, and database resources. Servers in the cloud can be physical machines or VMs. User interfaces are applied to request services. The provisioning tool carves out the cloud system to deliver the requested service.
- In addition to building the server cluster, the cloud platform demands distributed storage and concomitant services. The cloud computing resources are built into the data centers, which are typically owned and operated by a third-party provider. Consumers do not need to know the underlying technologies.
- The cloud demands a high degree of trust of massive amounts of data retrieved from large data centers. We need to build a framework to process large-scale data stored in the storage system. This demands a distributed file system over the database system. Other cloud resources are added into a cloud platform, including storage area networks (SANs), database systems, firewalls, and security devices.
- The software infrastructure of a cloud platform must handle all resource management and do most of the maintenance automatically. Software must detect the status of each node server joining and leaving, and perform relevant tasks accordingly.

Layered Cloud Architecture

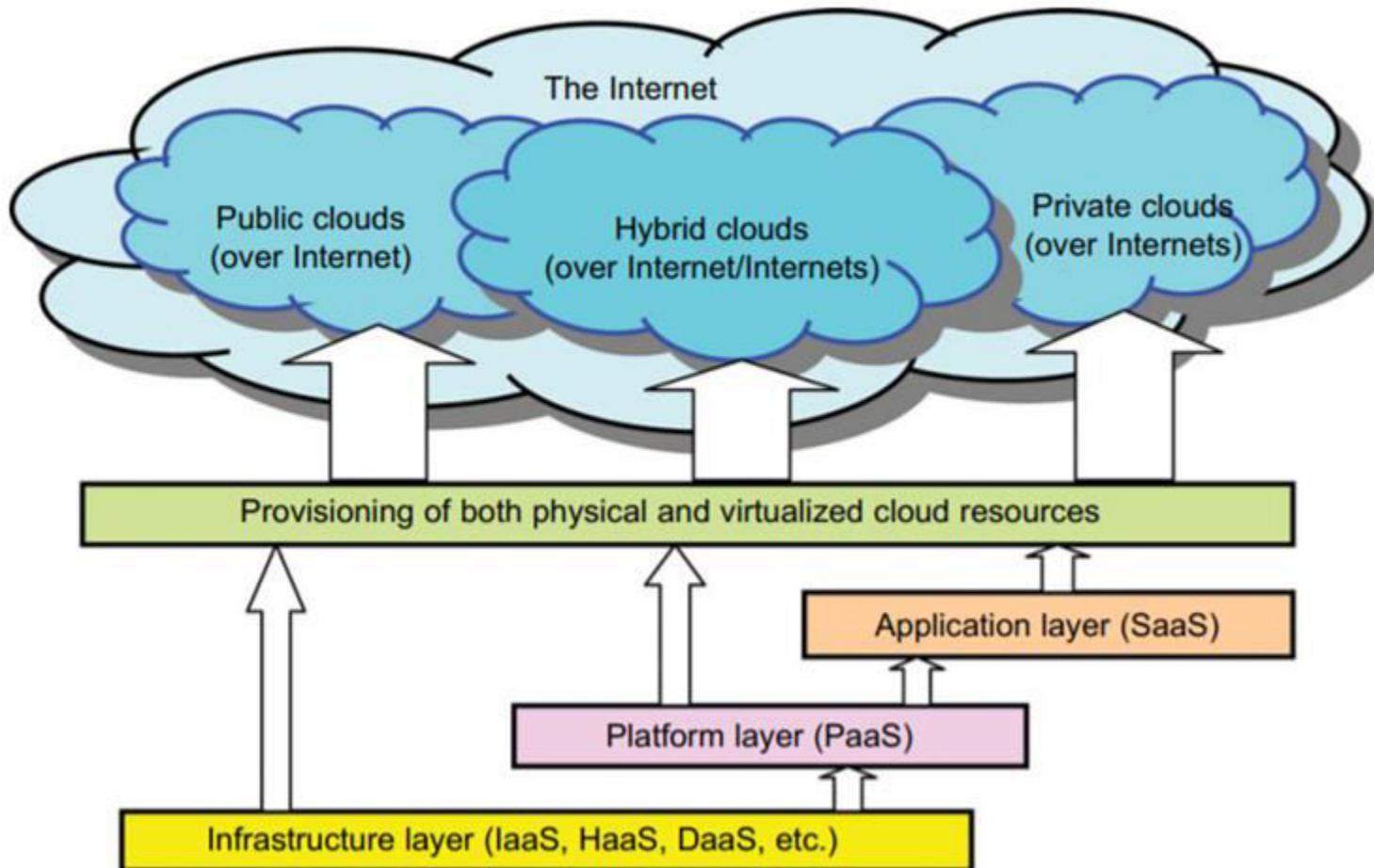
The architecture of a cloud is developed at three layers: infrastructure, platform, and application. These three development layers are implemented with virtualization and standardization of hardware and software resources provisioned in the cloud. The services to public, private, and hybrid clouds are conveyed to users through networking support over the Internet and intranets involved.

The infrastructure layer is built with virtualized compute, storage, and network resources. The abstraction of these hardware resources is meant to provide the flexibility demanded by users.

The platform layer is for general-purpose and repeated usage of the collection of software resources. This layer provides users with an environment to develop their applications, to test operation flows, and to monitor execution results and performance.

The application layer is formed with a collection of all needed software modules for SaaS applications. Service applications in this layer include daily office management work, such as information retrieval, document processing, and calendar and authentication services.

-

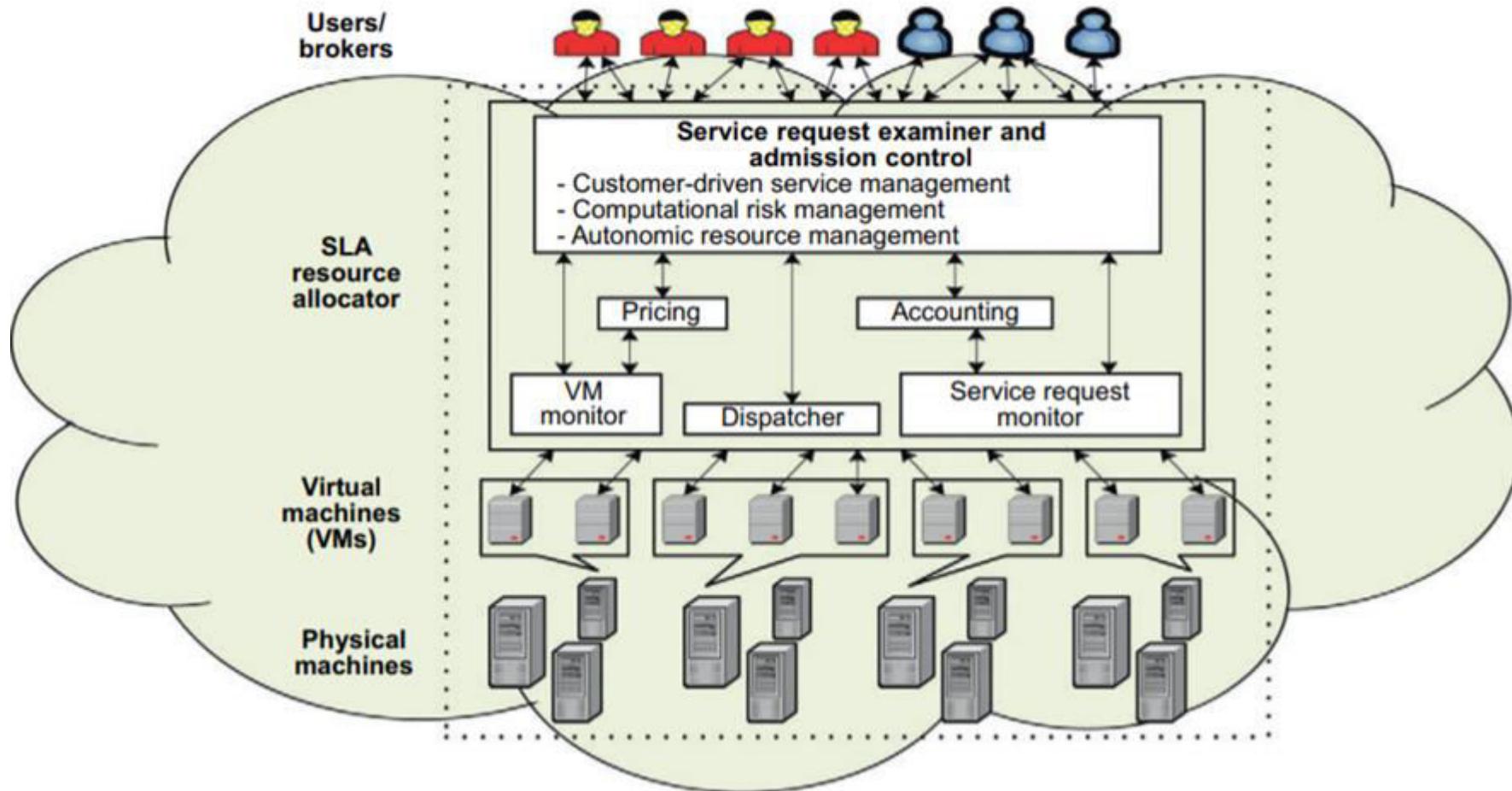


Market-Oriented Cloud Architecture

As consumers rely on cloud providers to meet more of their computing needs, they will require a specific level of QoS to be maintained by their providers, in order to meet their objectives and sustain their operations. To achieve this, the providers cannot deploy traditional system-centric resource management architecture. Instead, market-oriented resource management is necessary to regulate the supply and demand of cloud resources to achieve market equilibrium between supply and demand.

The designer needs to provide feedback on economic incentives for both consumers and providers. The purpose is to promote QoS-based resource allocation mechanisms. In addition, clients can benefit from the potential cost reduction of providers.

The request examiner ensures that there is no overloading of resources whereby many service requests cannot be fulfilled successfully due to limited resources. The Pricing mechanism decides how service requests are charged. The VM Monitor mechanism keeps track of the availability of VMs and their resource entitlements. The Dispatcher mechanism starts the execution of accepted service requests on allocated VMs. The Service Request Monitor mechanism keeps track of the execution progress of service requests. Multiple VMs can be started and stopped on demand on a single physical machine to meet accepted service requests, hence providing maximum flexibility.





THANK YOU

K.S.Srinivas
srinivasks@pes.edu
+91 80 2672 1983 Extn 701



CLOUD COMPUTING

Infrastructure as a Service (IaaS)

K.S.Srinivas

Department of Computer Science and Engineering

CLOUD COMPUTING

Infrastructure as a Service (IaaS)

Srinivas K S.

Associate Professor, Department of Computer Science

CLOUD COMPUTING

Cloud Service Models

Cloud computing delivers infrastructure, platform, and software (application) as services, which are made available as subscription-based services in a pay-as-you-go model to consumers. The services provided over the cloud can be generally categorized into three different service models:

- Infrastructure as a service
- Platform as a service
- Software as a service

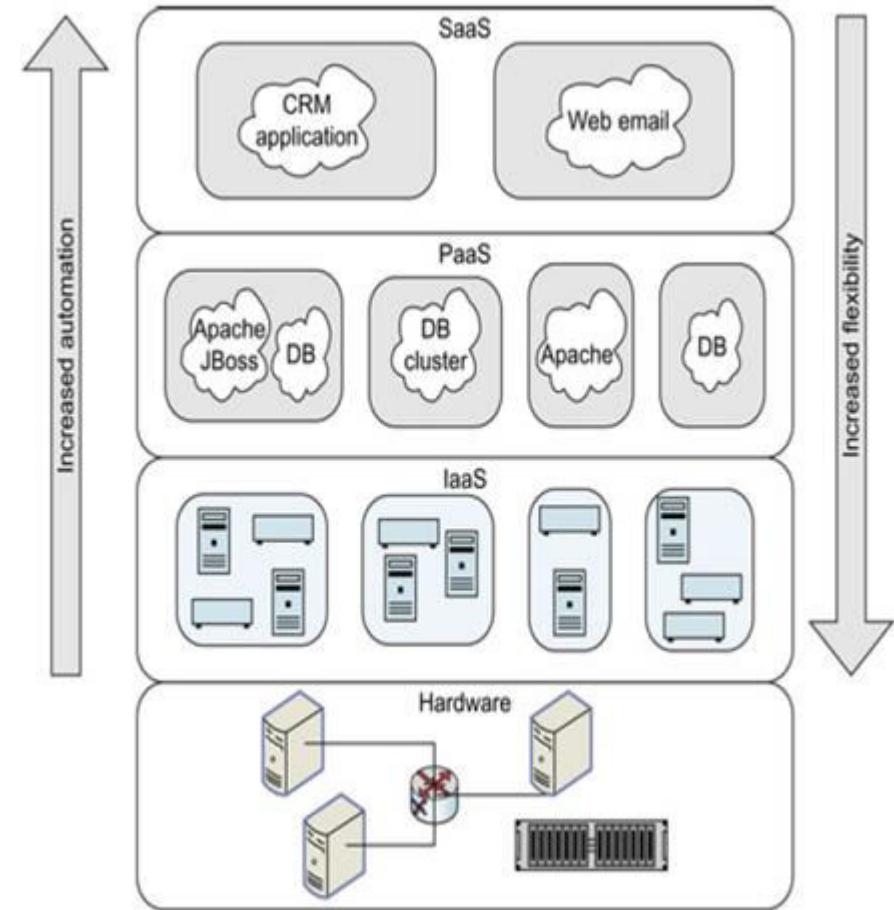


Figure 1.4 Cloud service models

The IaaS model is about providing compute and storage resources as a service.

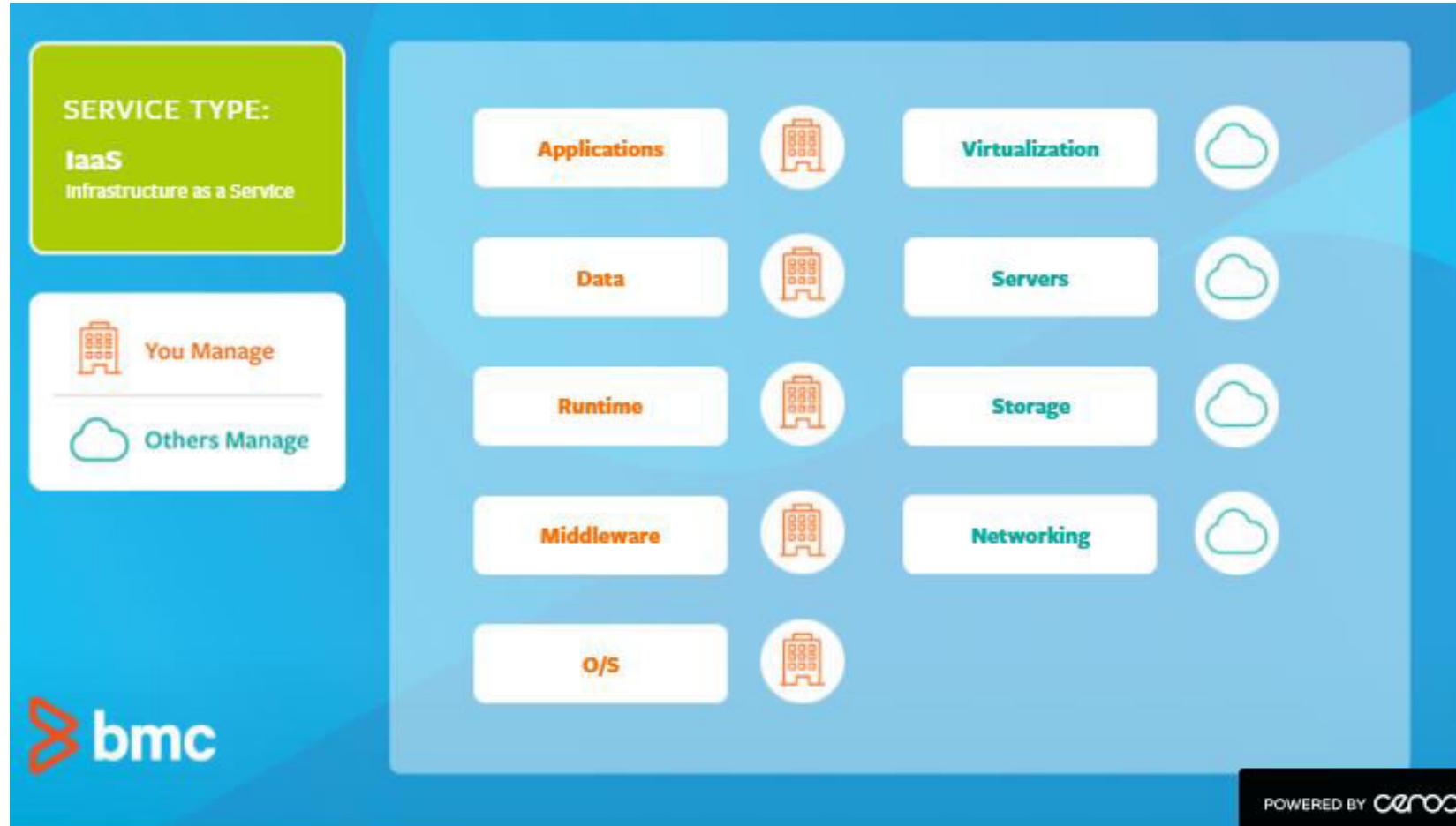
The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, deployed applications, and possibly limited control of select networking components (e.g., host firewalls). NIST Definitions

The user of IaaS has single ownership of the hardware infrastructure allotted to him. The IaaS provider has control over the actual hardware and the cloud user can request allocation of virtual resources.

IaaS enables end users to scale and shrink resources on an as-needed basis, reducing the need for high, up-front capital expenditures or unnecessary “owned” infrastructure, especially in the case of “spiky” workloads.

CLOUD COMPUTING

Infrastructure as a Service (IaaS)



CLOUD COMPUTING

IaaS platform and architecture

<https://www.ibm.com/cloud/learn/iaas#toc-virtual-pr-BS7->

Hwbs

IaaS is made up of a collection of physical and virtualized resources that provide consumers with the basic building blocks needed to run applications and workloads in the cloud.

Physical data centers: IaaS providers will manage large data centers, typically around the world, that contain the physical machines required to power the various layers of abstraction on top of them and that are made available to end users over the web. In most IaaS models, end users do not interact directly with the physical infrastructure, but it is provided as a service to them.

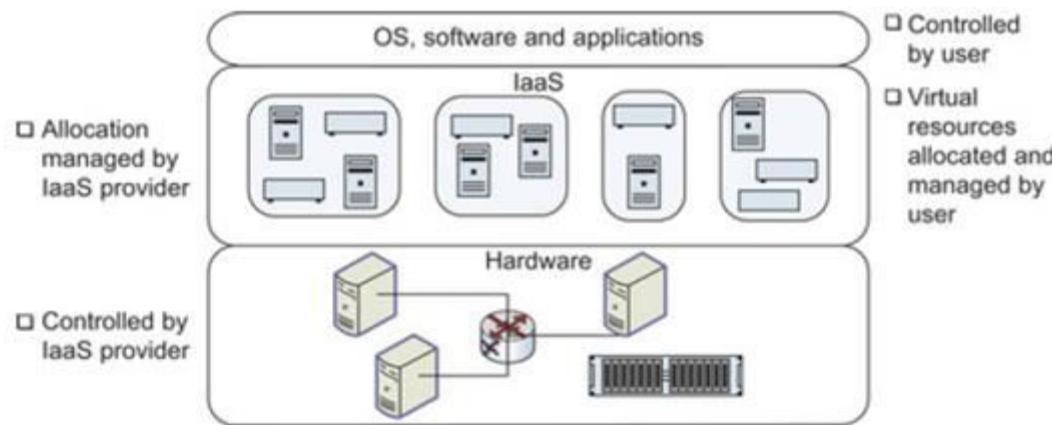
Compute: Providers manage the hypervisors and end users can then programmatically provision virtual “instances” with desired amounts of compute and memory (and sometimes storage). Most providers offer both CPUs and GPUs for different types of workloads. Cloud compute also typically comes paired with supporting services like auto scaling and load balancing that provide the scale and performance characteristics that make cloud desirable in the first place.



IaaS platform and architecture

Network: Networking in the cloud is a form of Software Defined Networking in which traditional networking hardware, such as routers and switches, are made available programmatically, typically through APIs.

Storage: The three primary types of cloud storage are block storage, file storage, and object storage. Block and file storage are common in traditional data centers but can often struggle with scale, performance and distributed characteristics of cloud. Thus, of the three, object storage has thus become the most common mode of storage in the cloud given that it is highly distributed (and thus resilient), it leverages commodity hardware, data can be accessed easily over HTTP, and scale is not only essentially limitless but performance scales linearly as the cluster grows.



- **Flexible:** The most flexible cloud computing model.
- **Control:** Clients retain complete control of their infrastructure
- **Pay-as-you-Go:** Unlike traditional IT, IaaS does not require any upfront, capital expenditures, and end users are only billed for what they use.
- **Speed:** With IaaS, users can provision small or vast amounts of resources in a matter of minutes, testing new ideas quickly or scaling proven ones even quicker.
- **Availability:** Through things like multizone regions, the availability and resiliency of cloud applications can exceed traditional approaches.

- **Scale:** With seemingly limitless capacity and the ability to scale resources either automatically or with some supervision, it's simple to go from one instance of an application or workload to many.
- **Latency and performance:** Given the broad geographic footprint of most IaaS providers, it's easy to put apps and services closer to your users, reducing latency and improving performance.

CLOUD COMPUTING

When to Use IaaS

<https://www.bmc.com/blogs/saas-vs-paas-vs-iaas-whats-the-difference-and-how-to-choose/#ref3>



- **Startups and small companies** may prefer IaaS to avoid spending time and money on purchasing and creating hardware and software.
- **Larger companies** may prefer to retain complete control over their applications and infrastructure, but they want to purchase only what they actually consume or need.
- **Companies experiencing rapid growth** like the scalability of IaaS, and they can change out specific hardware and software easily as their needs evolve.
- **Website hosting** :Running websites using IaaS can be less expensive than traditional web hosting.

CLOUD COMPUTING

When to Use IaaS

<https://azure.microsoft.com/en-in/overview/what-is-iaas/>



- **Storage, backup and recovery :**Organisations avoid the capital outlay for storage and complexity of storage management, which typically requires a skilled staff to manage data. IaaS is useful for handling unpredictable demand and steadily growing storage needs. It can also simplify planning and management of backup and recovery systems.
- **High-performance computing.** High-performance computing (HPC) on supercomputers, computer grids or computer clusters helps solve complex problems involving millions of variables or calculations.
- **Big data analysis.** Mining data sets to locate or tease out hidden patterns requires a huge amount of processing power, which IaaS economically provides.
- Anytime you are unsure of a new application's demands, IaaS offers plenty of flexibility and scalability.

- **Security:** While the customer is in control of the apps, data, middleware, and the OS platform, security threats can still be sourced from the host or other virtual machines (VMs). Insider threat or system vulnerabilities may expose data communication between the host infrastructure and VMs to unauthorized entities.
- **Multi-tenant security:** Since the hardware resources are dynamically allocated across users as made available, the vendor is required to ensure that other customers cannot access data deposited to storage assets by previous customers. Similarly, customers must rely on the vendor to ensure that VMs are adequately isolated within the multi tenant cloud architecture.
- **Internal resources and training:** Additional resources and training may be required for the workforce to learn how to effectively manage the infrastructure. Due to inadequate control into the infrastructure however, monitoring and management of the resources may be difficult without adequate training and resources available in house.

CLOUD COMPUTING

IaaS Providers

- Amazon Web Services : Amazon EC2.
- Microsoft Azure : Azure Virtual Machines.
- Google Cloud : Google compute engine.
- IBM Cloud : IBM Cloud Private.
- Digital Ocean : Digital Ocean Droplets.

CLOUD COMPUTING

Demo: IaaS on AWS (Compute as a service)

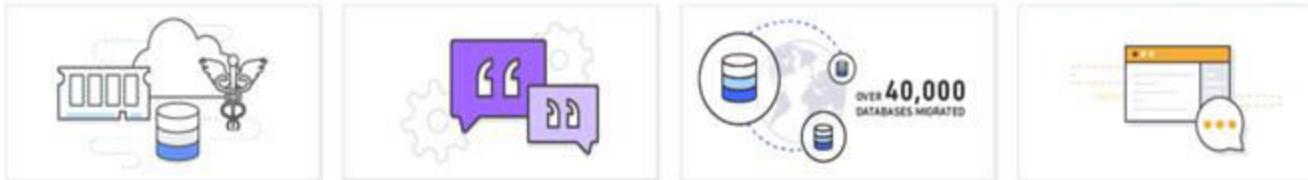
tutorial

<https://www.datacamp.com/community/tutorials/aws-ec2-beginner-tutorial>



- Step 1: Create an AWS Account and Sign into AWS.

A screenshot of the AWS homepage. At the top, there's a navigation bar with links like "Menu", "Contact Sales", "Products", "Pricing", "Getting Started", "More", "English", "My Account", and a prominent yellow "Sign In to the Console" button, which is highlighted with a red box. Below the navigation bar, there's a large banner for "AWS re:Invent" with the text "November 27 - December 1 | Las Vegas, NV" and "Register now while tickets are still available". To the right of the banner, there's a callout box with "Manage Your Resources" and "Sign In to the Console" buttons, along with information about the "AWS Console Mobile App" and a link to "Download the Mobile App".



ELASTICACHE FOR REDIS HIPAA ELIGIBLE
Power secure healthcare apps with sub-millisecond latency

AMAZON LEX
Building better chatbots

AWS DATABASE MIGRATION SERVICE
Easily migrate and convert databases

CONTACT SALES
Get help from our sales experts

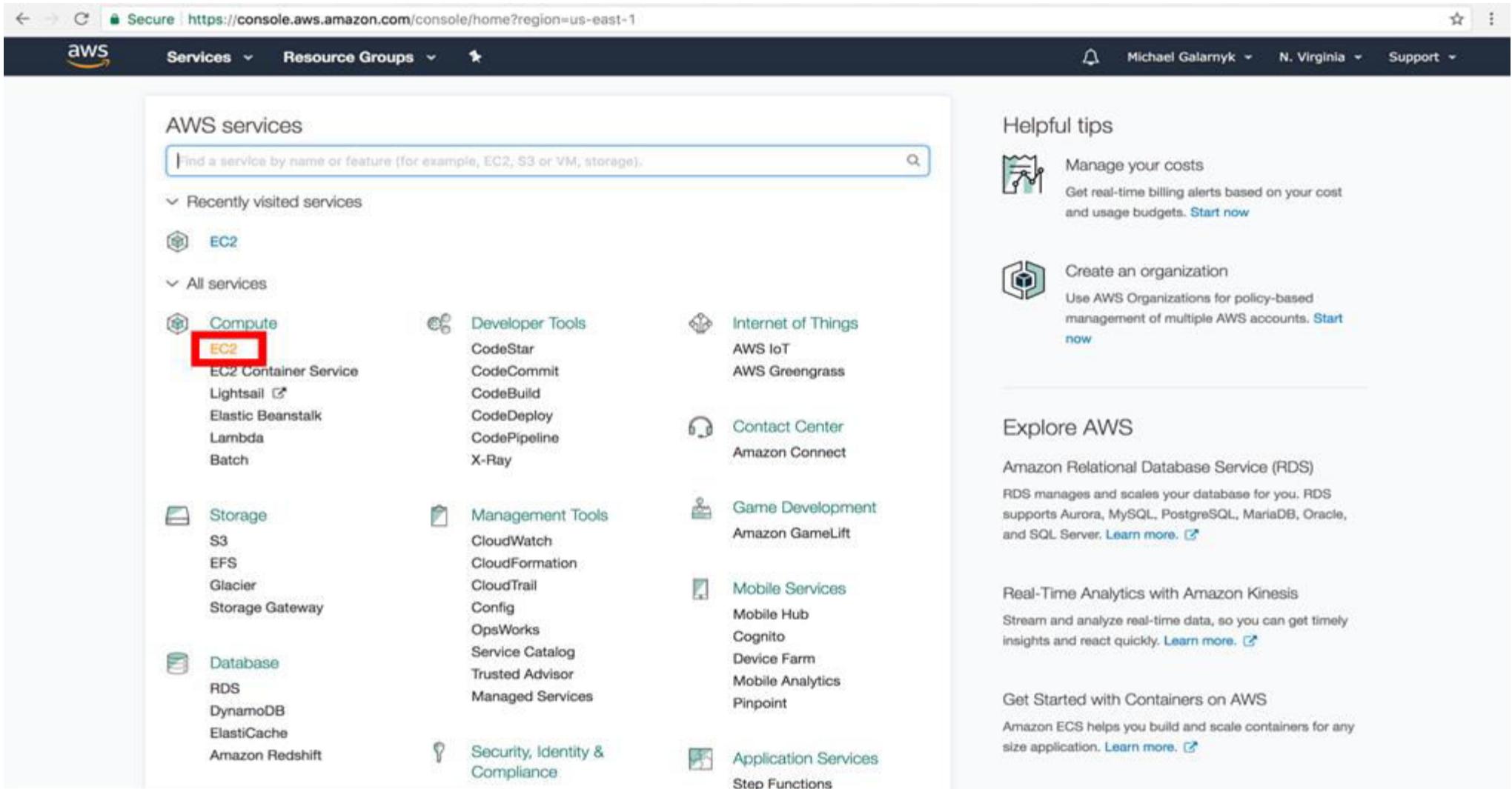
Explore Our Products



CLOUD COMPUTING

Demo: IaaS on AWS (Compute as a service)

- Step 2: On the EC2 Dashboard, click on EC2..



The screenshot shows the AWS Cloud Services Dashboard. At the top, there's a navigation bar with the AWS logo, a 'Services' dropdown, 'Resource Groups' dropdown, and user information for Michael Galarnyk (N. Virginia) and Support. Below the navigation is a search bar with placeholder text 'Find a service by name or feature (for example, EC2, S3 or VM, storage)' and a magnifying glass icon.

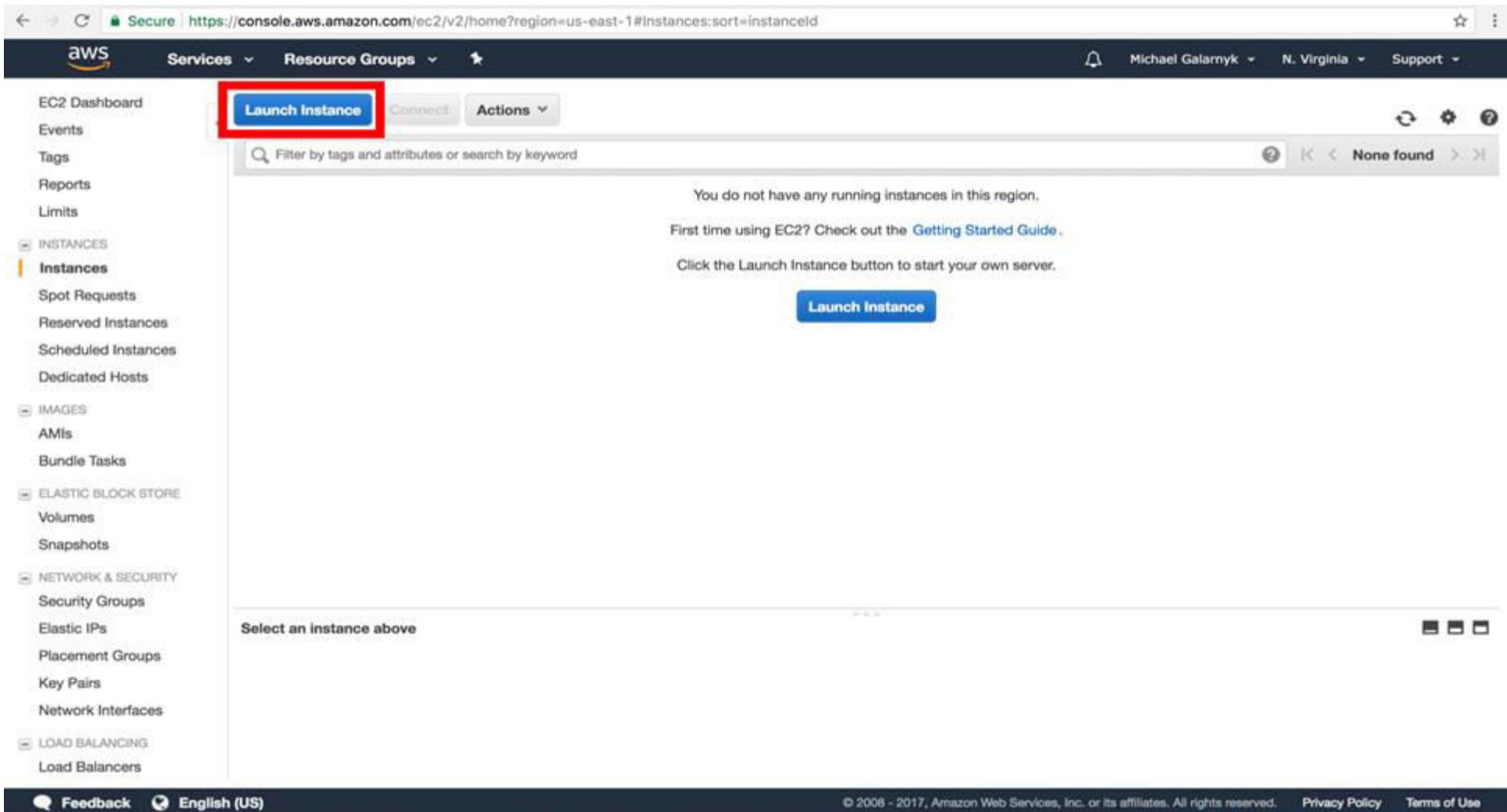
The main area is titled 'AWS services' and contains a grid of service icons and names. The 'Compute' section is expanded, showing 'EC2' which is highlighted with a red box. Other services listed under Compute include EC2 Container Service, Lightsail, Elastic Beanstalk, Lambda, and Batch. The other sections shown are Storage (S3, EFS, Glacier, Storage Gateway), Database (RDS, DynamoDB, ElastiCache, Amazon Redshift), and various Management Tools, Mobile Services, Security, Identity & Compliance, and Application Services.

To the right of the service grid, there's a 'Helpful tips' section with two items: 'Manage your costs' (with a graph icon) and 'Create an organization' (with a hexagon icon). Below that is an 'Explore AWS' section with three items: 'Amazon Relational Database Service (RDS)', 'Real-Time Analytics with Amazon Kinesis', and 'Get Started with Containers on AWS'.

CLOUD COMPUTING

Demo: IaaS on AWS (Compute as a service)

- Step 3: On the Amazon EC2 console, click on Launch Instance.

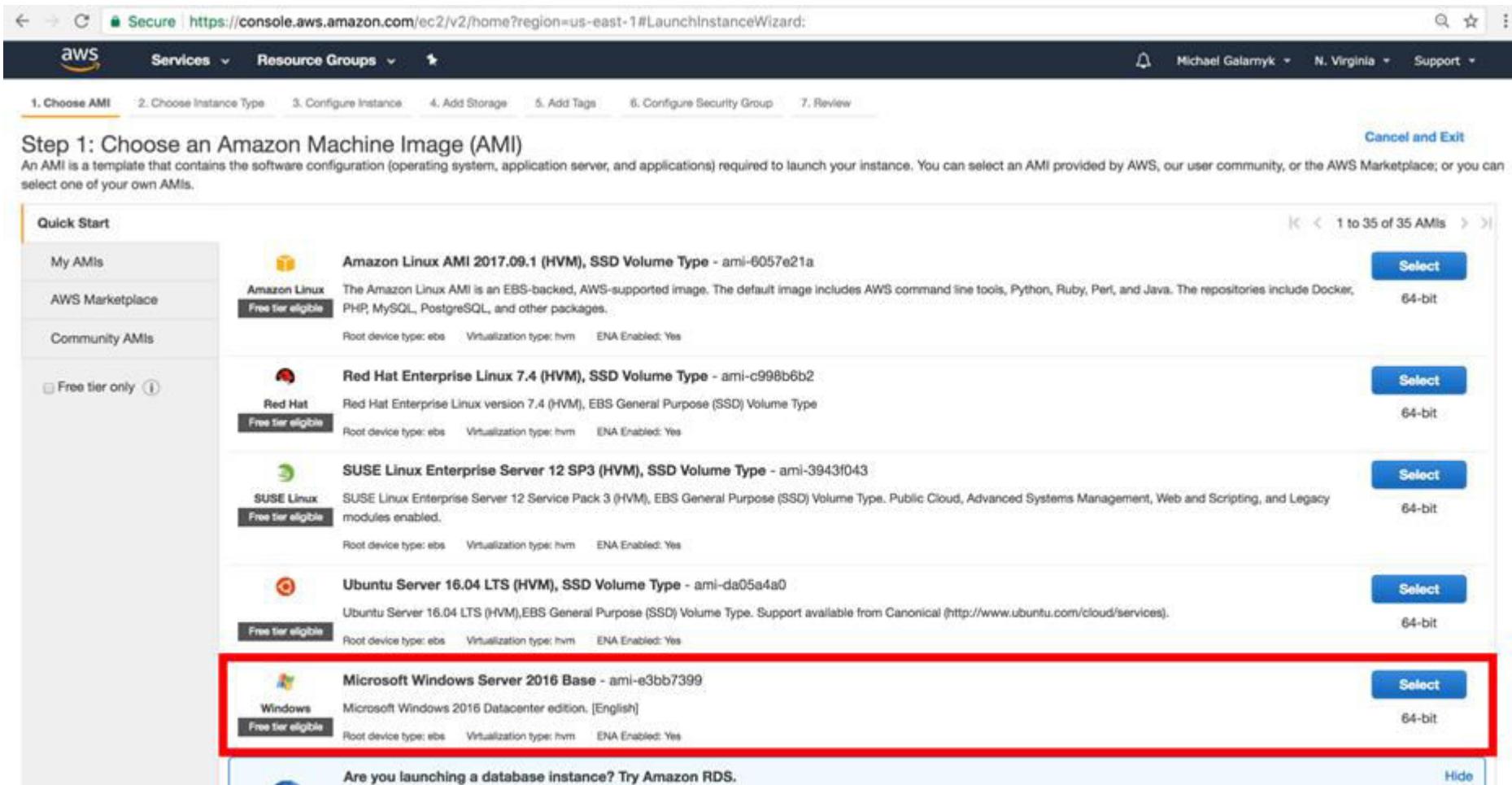


The screenshot shows the AWS EC2 Instances page. The left sidebar menu is visible, with 'Instances' selected. The main content area displays a message: 'You do not have any running instances in this region.' Below it, there's a note: 'First time using EC2? Check out the [Getting Started Guide](#).'. A prominent blue 'Launch Instance' button is centered in the middle of the page. The browser address bar shows the URL: <https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#instances;sort=instanceId>.

CLOUD COMPUTING

Demo: IaaS on AWS (Compute as a service)

- Step 4: Select the required operating system image.



The screenshot shows the AWS Launch Instance Wizard Step 1: Choose AMI. The page title is "Step 1: Choose an Amazon Machine Image (AMI)". It explains that an AMI is a template containing software configuration required to launch an instance. The user can select from AWS-provided, Marketplace, or community AMIs, or their own. A sidebar on the left lists "Quick Start" options: My AMIs, AWS Marketplace, Community AMIs, and a "Free tier only" checkbox. The main content area displays a list of available AMIs:

Image	Name	Description	Select	Architecture
	Amazon Linux AMI 2017.09.1 (HVM), SSD Volume Type - ami-6057e21a	The Amazon Linux AMI is an EBS-backed, AWS-supported image. The default image includes AWS command line tools, Python, Ruby, Perl, and Java. The repositories include Docker, PHP, MySQL, PostgreSQL, and other packages.	<button>Select</button>	64-bit
	Red Hat Enterprise Linux 7.4 (HVM), SSD Volume Type - ami-c998b6b2	Red Hat Enterprise Linux version 7.4 (HVM), EBS General Purpose (SSD) Volume Type	<button>Select</button>	64-bit
	SUSE Linux Enterprise Server 12 SP3 (HVM), SSD Volume Type - ami-3943f043	SUSE Linux Enterprise Server 12 Service Pack 3 (HVM), EBS General Purpose (SSD) Volume Type. Public Cloud, Advanced Systems Management, Web and Scripting, and Legacy modules enabled.	<button>Select</button>	64-bit
	Ubuntu Server 16.04 LTS (HVM), SSD Volume Type - ami-da05a4a0	Ubuntu Server 16.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).	<button>Select</button>	64-bit
	Microsoft Windows Server 2016 Base - ami-e3bb7399	Microsoft Windows 2016 Datacenter edition. [English]	<button>Select</button>	64-bit

A red box highlights the Microsoft Windows Server 2016 Base AMI. At the bottom of the page, there is a note: "Are you launching a database instance? Try Amazon RDS." with a "Hide" button.

Demo: IaaS on AWS (Compute as a service)

- Step 5: Select the required hardware configuration

Secure | https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#LaunchInstanceWizard:

aws Services Resource Groups

Michael Galarnyk N. Virginia Support

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your computing needs.

Filter by: All instance types Current generation Show/Hide Columns

Currently selected: t2.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only)

Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
General purpose	t2.micro <small>Free tier eligible</small>	1	1	EBS only	-	Low to Moderate	Yes
General purpose	t2.small	1	2	EBS only	-	Low to Moderate	Yes
General purpose	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
General purpose	t2.large	2	8	EBS only	-	Low to Moderate	Yes
General purpose	t2.xlarge	4	16	EBS only	-	Moderate	Yes
General purpose	t2.2xlarge	8	32	EBS only	-	Moderate	Yes
General purpose	m4.large	2	8	EBS only	Yes	Moderate	Yes
General purpose	m4.xlarge	4	16	EBS only	Yes	High	Yes

Cancel Previous Review and Launch Next: Configure Instance Details

CLOUD COMPUTING

Demo: IaaS on AWS (Compute as a service)

- Step 6: click on "Review and Launch"

Secure | https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#LaunchInstanceWizard:

AWS Services Resource Groups Michael Galarnyk N. Virginia Support

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your computing needs.

Filter by: All instance types Current generation Show/Hide Columns

Currently selected: t2.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only)

Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
General purpose	t2.micro <small>Free tier eligible</small>	1	1	EBS only	-	Low to Moderate	Yes
General purpose	t2.small	1	2	EBS only	-	Low to Moderate	Yes
General purpose	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
General purpose	t2.large	2	8	EBS only	-	Low to Moderate	Yes
General purpose	t2.xlarge	4	16	EBS only	-	Moderate	Yes
General purpose	t2.2xlarge	8	32	EBS only	-	Moderate	Yes
General purpose	m4.large	2	8	EBS only	Yes	Moderate	Yes
General purpose	m4.xlarge	4	16	EBS only	Yes	High	Yes

Cancel Previous **Review and Launch** Next: Configure Instance Details

CLOUD COMPUTING

Demo: IaaS on AWS (Compute as a service)

- Step 7: Click on Launch.

Secure | https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#LaunchInstanceWizard:

aws Services Resource Groups

Michael Galamyk N. Virginia Support

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 7: Review Instance Launch

Please review your instance launch details. You can go back to edit changes for each section. Click **Launch** to assign a key pair to your instance and complete the launch process.

⚠ Improve your instances' security. Your security group, launch-wizard-4, is open to the world.
Your instances may be accessible from any IP address. We recommend that you update your security group rules to allow access from known IP addresses only.
You can also open additional ports in your security group to facilitate access to the application or service you're running, e.g., HTTP (80) for web servers. [Edit security groups](#).

AMI Details [Edit AMI](#)

 Microsoft Windows Server 2016 Base - ami-e3bb7399
Free tier eligible Microsoft Windows 2016 Datacenter edition. [English]
Root Device Type: ebs Virtualization type: hvm

If you plan to use this AMI for an application that benefits from Microsoft License Mobility, fill out the [License Mobility Form](#). Don't show me this again

Instance Type [Edit instance type](#)

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
t2.micro	Variable	1	1	EBS only	-	Low to Moderate

Security Groups [Edit security groups](#)

Security group name: launch-wizard-4
Description: launch-wizard-4 created 2017-11-09T12:13:45.100-08:00

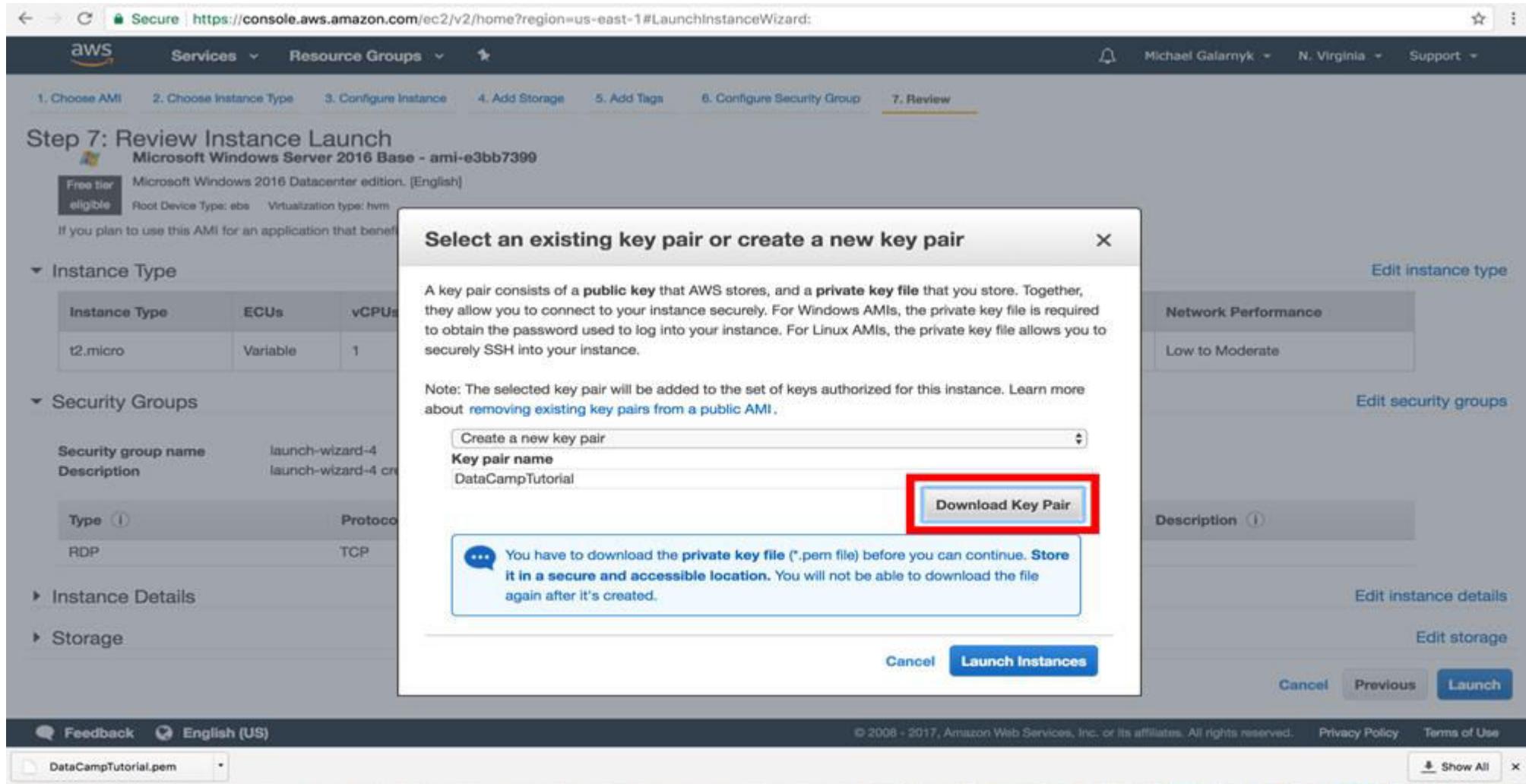
Cancel Previous **Launch**

Feedback English (US) © 2006 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

CLOUD COMPUTING

Demo: IaaS on AWS (Compute as a service)

- Step 8: 7. Select "Create a new key pair". Click on "Download Key Pair". This will download the key. Keep it somewhere safe.

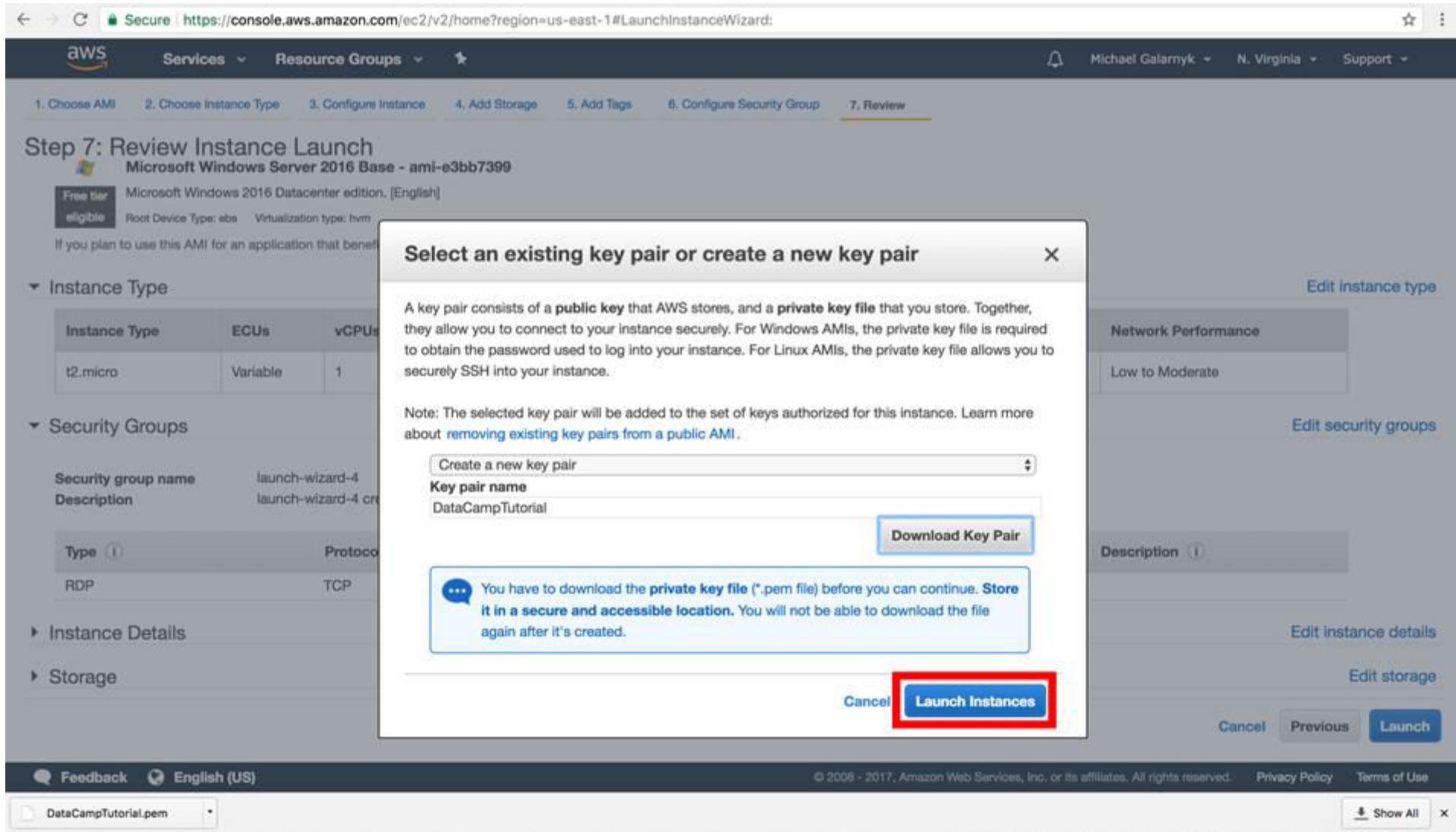


The screenshot shows the AWS Cloud9 interface for launching an instance. The main window displays the 'Step 7: Review Instance Launch' configuration. A modal dialog box titled 'Select an existing key pair or create a new key pair' is open. Inside the dialog, there is a note about key pairs and a form to either 'Create a new key pair' or select an existing one ('Key pair name: DataCampTutorial'). A prominent blue button labeled 'Download Key Pair' is highlighted with a red box. Below the button, a message box contains the instruction: 'You have to download the private key file (*.pem file) before you can continue. Store it in a secure and accessible location. You will not be able to download the file again after it's created.' At the bottom of the dialog are 'Cancel' and 'Launch Instances' buttons.

CLOUD COMPUTING

Demo: IaaS on AWS (Compute as a service)

- Step 9: Next, click on "Launch Instances".



Secure | https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#LaunchInstanceWizard:

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 7: Review Instance Launch

Microsoft Windows Server 2016 Base - ami-e3bb7399

Free tier eligible Microsoft Windows 2016 Datacenter edition, [English]

Root Device Type: ebs Virtualization type: hvm

If you plan to use this AMI for an application that benefits from a low-latency connection to Amazon RDS, choose a t2 instance type.

Instance Type

Instance Type	ECUs	vCPUs
t2.micro	Variable	1

Security Groups

Security group name	Description
launch-wizard-4	launch-wizard-4 created

Type: RDP Protocol: TCP

Instance Details

Storage

Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about [removing existing key pairs from a public AMI](#).

Create a new key pair

Key pair name: DataCampTutorial

Download Key Pair

You have to download the **private key file (*.pem file)** before you can continue. Store it in a secure and accessible location. You will not be able to download the file again after it's created.

Cancel Launch Instances

Cancel Previous Launch

Feedback English (US)

DataCampTutorial.pem

Show All

CLOUD COMPUTING

Demo: IaaS on AWS (Compute as a service)



- Step 10: The instance is now launched. Go back to the Amazon EC2 console.

Secure | https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#LaunchInstanceWizard:

aws Services Resource Groups

Michael Galarmy N. Virginia Support

Launch Status

Your instances are now launching
The following instance launches have been initiated: [i-0029f691bf6a7c52f](#) View launch log

Get notified of estimated charges
Create billing alerts to get an email notification when estimated charges on your AWS bill exceed an amount you define (for example, if you exceed the free usage tier).

How to connect to your instances

Your instances are launching, and it may take a few minutes until they are in the running state, when they will be ready for you to use. Usage hours on your new instances will start immediately and continue to accrue until you stop or terminate your instances.

Click [View Instances](#) to monitor your instances' status. Once your instances are in the running state, you can [connect](#) to them from the Instances screen. [Find out](#) how to connect to your instances.

Here are some helpful resources to get you started

- [Amazon EC2: User Guide](#)
- [How to connect to your Windows instance](#)
- [Learn about AWS Free Usage Tier](#)
- [Amazon EC2: Microsoft Windows Guide](#)
- [Amazon EC2: Discussion Forum](#)

While your instances are launching you can also

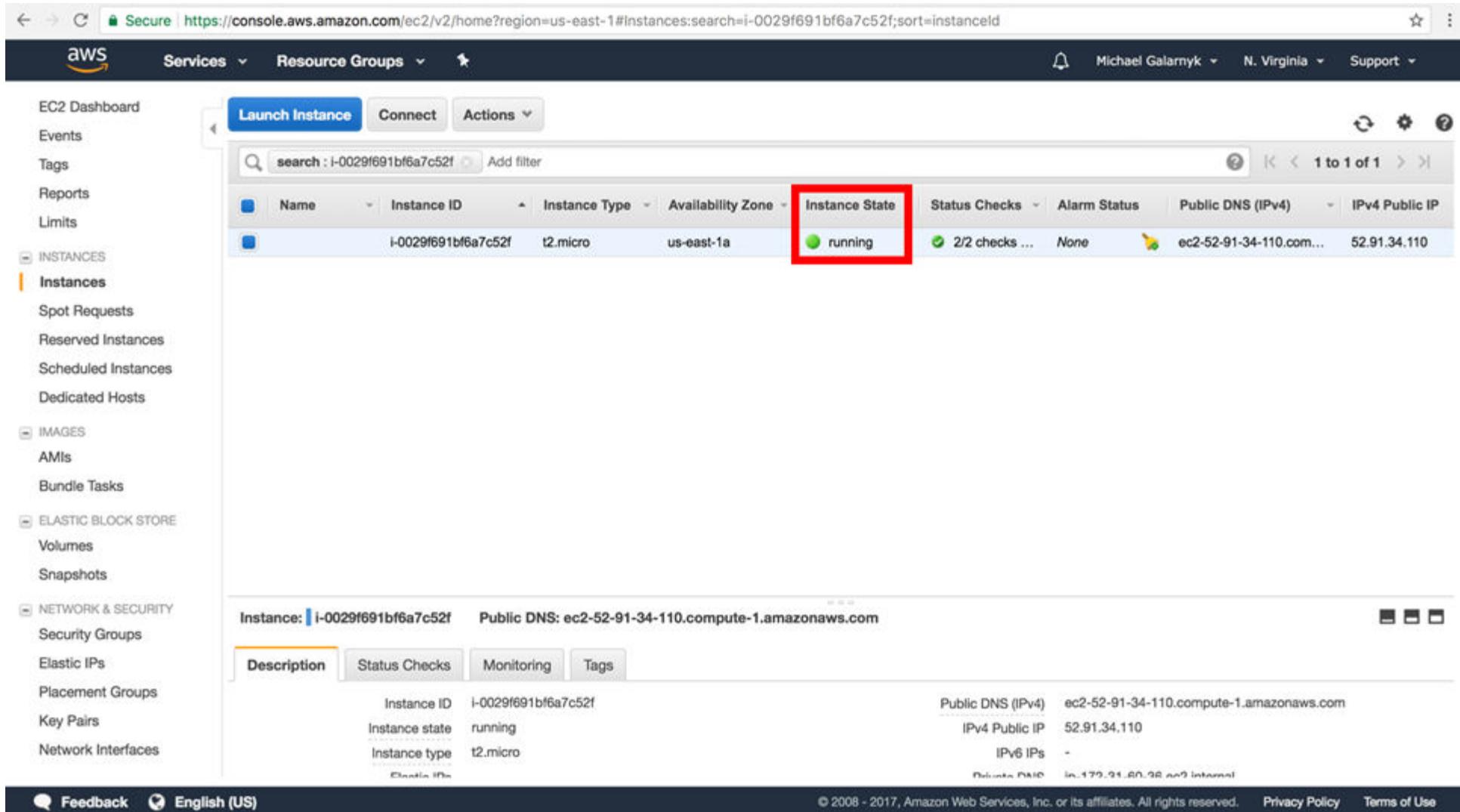
- [Create status check alarms](#) to be notified when these instances fail status checks. (Additional charges may apply)
- [Create and attach additional EBS volumes](#) (Additional charges may apply)

Feedback English (US) DataCampTutorial.pem Show All

© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Demo: IaaS on AWS (Compute as a service)

- Step 11: Observe the instance is up and running



The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with navigation links for EC2 Dashboard, Events, Tags, Reports, Limits, INSTANCES (with 'Instances' selected), Spot Requests, Reserved Instances, Scheduled Instances, Dedicated Hosts, IMAGES (with 'AMIs' selected), and ELASTIC BLOCK STORE (with 'Volumes' selected). The main content area has tabs for Launch Instance, Connect, and Actions. Below that is a search bar with 'search : i-0029f691bf6a7c52f'. The main table has columns: Name, Instance ID, Instance Type, Availability Zone, Instance State, Status Checks, Alarm Status, Public DNS (IPv4), and IPv4 Public IP. A single row is shown for an instance with Instance ID 'i-0029f691bf6a7c52f', Instance Type 't2.micro', Availability Zone 'us-east-1a', and Instance State 'running' (highlighted with a red box). At the bottom, there's a detailed view for the selected instance, showing fields like Instance ID, Instance state, Instance type, Public DNS (IPv4), IPv4 Public IP, IPv6 IPs, and Private IP.

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP
	i-0029f691bf6a7c52f	t2.micro	us-east-1a	running	2/2 checks ...	None	ec2-52-91-34-110.compute-1.amazonaws.com...	52.91.34.110

Instance: i-0029f691bf6a7c52f Public DNS: ec2-52-91-34-110.compute-1.amazonaws.com

Description	Status Checks	Monitoring	Tags
Instance ID: i-0029f691bf6a7c52f	Instance state: running	Instance type: t2.micro	Public DNS (IPv4): ec2-52-91-34-110.compute-1.amazonaws.com
Instance state: running	IPv4 Public IP: 52.91.34.110	IPv6 IPs: -	Private IP: 172.31.80.38 (not internal)
Instance type: t2.micro			

Feedback English (US) © 2006 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use



THANK YOU

K.S.Srinivas
srinivasks@pes.edu
+91 80 2672 1983 Extn 701



CLOUD COMPUTING

REST and Web Services

K.S.Srinivas

Department of Computer Science and Engineering

CLOUD COMPUTING

REST and Web Services

Srinivas K S.

Associate Professor, Department of Computer Science

CLOUD COMPUTING

Service oriented architecture (SOA)

<https://www.ibm.com/cloud/learn/soa>,

<https://www.geeksforgeeks.org/service-oriented-architecture/>



SOA, or service-oriented architecture, defines a way to make software components reusable via service interfaces. These interfaces utilize common communication standards in such a way that they can be rapidly incorporated into new applications without having to perform deep integration each time.

Each service in an SOA embodies the code and data integrations required to execute a complete, discrete business function. The service interfaces provide loose coupling, meaning they can be called with little or no knowledge of how the integration is implemented underneath. The services are exposed using standard network protocols—such as SOAP (simple object access protocol)/HTTP or JSON/HTTP—to send requests to read or change data. The services are published in a way that enables developers to quickly find them and reuse them to assemble new applications.

two major service-oriented architecture styles :

1. **REST** (REpresentational State Transfer)
2. **WS** (Web Services)

REST, or REpresentational State Transfer, is an architectural style for providing standards between computer systems on the web, making it easier for systems to communicate with each other. REST-compliant systems, often called RESTful systems, are characterized by how they are stateless and separate the concerns of client and server. It is a software architecture style for distributed systems, particularly distributed hypermedia systems, such as the World Wide Web. It was developed in parallel with the HTTP/1.1

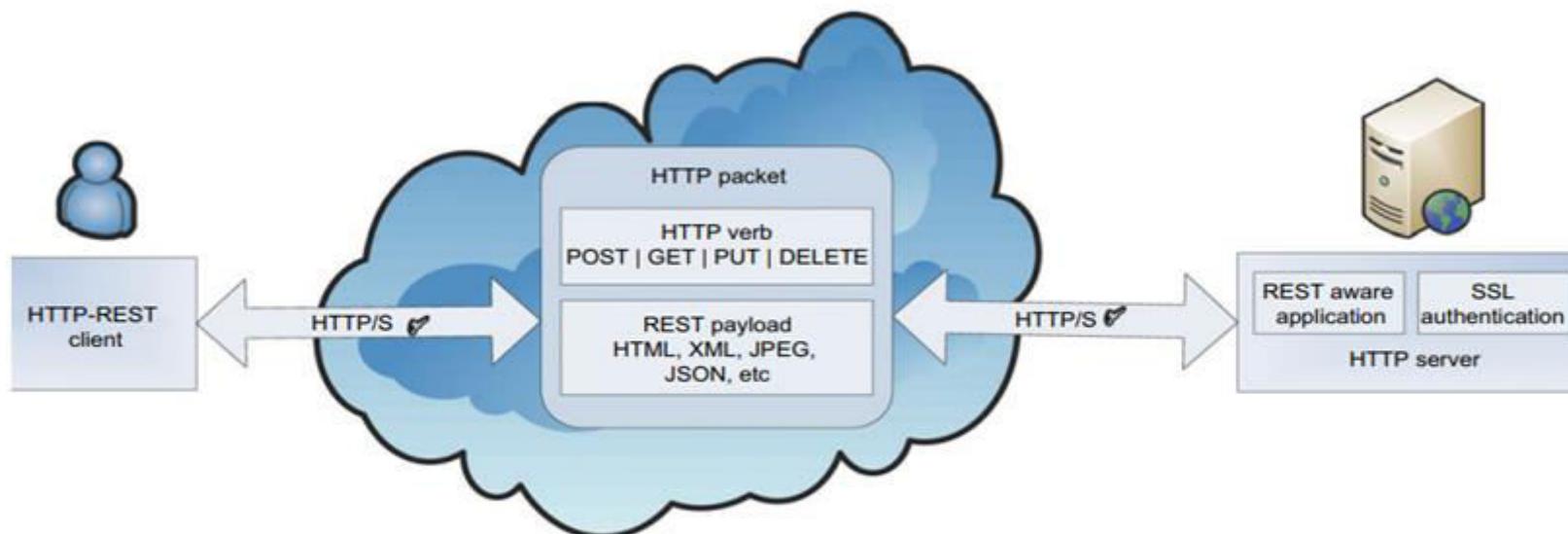


FIGURE 5.1

A simple REST interaction between user and server in HTTP specification.

The REST architectural style is based on four principles:

1. Resource Identification through URIs.
1. Uniform, Constrained Interface.
1. Self-Descriptive Message.
1. Stateless Interactions.

Resource : The key abstraction of information in REST is a resource. Any information that can be named can be a resource, such as a document or image or a temporal service.

The RESTful web service exposes a set of resources which identify targets of interaction with its clients. Each particular resource is identified by a unique name, or more precisely, a Uniform Resource Identifier (URI) providing a global addressing space for resources involved in an interaction between components as well as facilitating service discovery. The URIs can be bookmarked or exchanged via hyperlinks, providing more readability.

URI – Uniform Resource Identifier

- The name of the object on the web
- Identifies a resource by
 - Name
 - Location
 - Or both

URL – Uniform Resource Locator

- Subset of an URI
- Specifies where to find a resource – the location
- How to retrieve the resource

Interaction with RESTful web services is done via the HTTP standard, client/server cacheable protocol.

Resources are manipulated using a fixed set of four CRUD (create, read, update, delete) operations:

- GET — retrieve a specific resource or a collection of resources
- POST — Create or partial update of a resource
- PUT — Create or update resource by replacement.
- DELETE — remove a resource

Operations characteristics:

1. **Safe** : An operation is safe if does not modify resources.
2. **Idempotent**: An idempotent operation is an operation, action, or request that can be applied multiple times without changing the result, i.e. the state of the system, beyond the initial application. Making multiple identical requests has the same effect as making a single request.

Ref - <https://stackoverflow.com/a/31797951>

Operation	Safe	Idempotent	When to use
GET	Yes	Yes	Mostly for retrieving resources. Can call multiple times.
PUT	No	Yes	Modifies a resource but no additional impact if called multiple times
POST	No	No	Modifies resources, multiple calls will cause additional effect if called with same parameter
DELETE	No	Yes	Removing a resource.

A REST message includes enough information to describe how to process the message. This enables intermediaries to do more with the message without parsing the message contents.

In REST, resources are decoupled from their representation so that their content can be accessed in a variety of standard formats (e.g., HTML, XML, MIME, plain text, PDF, JPEG, JSON, etc.) i.e the resources can have **multiple representations**. RESTful systems empowers client to ask for data in a form they understand.

Example:

```
GET /articles/23 HTTP/1.1
Accept: text/html, application/xhtml
```

```
HTTP/1.1 200 (OK)
Content-Type: text/html
```

Client request

Server Response

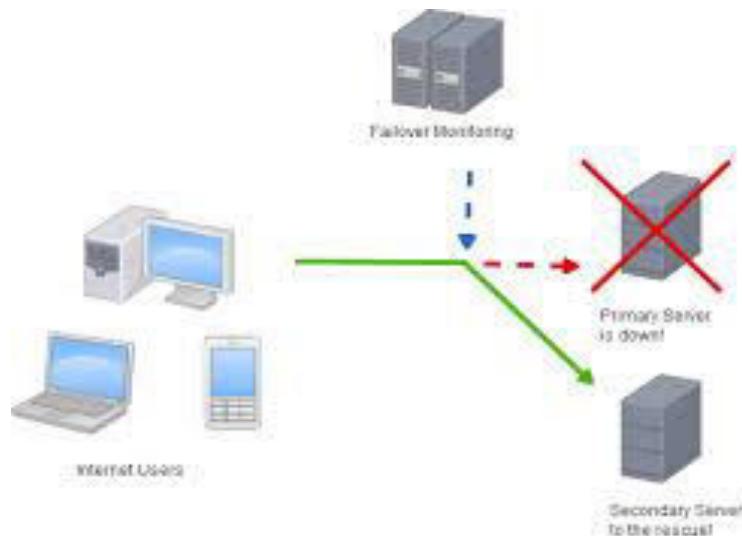
Metadata about the resource is available and can be used for various purposes, such as cache control, transmission error detection, authentication or authorization, and access control.

Stateless Interactions: Systems that follow the REST paradigm are stateless, meaning that the server does not need to know anything about what state the client is in and vice versa. In this way, both the server and the client can understand any message received, even without seeing previous messages. Each client request is treated independently.

Statelessness benefits :

1. Clients isolated against changes on server
2. Promotes redundancy - unlocks performance:
 - a. don't really need to know which server client was talking to
 - b. No synchronization overhead

No state saved on server, so even if server fails, the client connects to another server and continue



1. Rest is not a standard :

It is a design and architectural style for large scale distributed systems

1. Rest is not same as http:

Http can also be used in non-RESTful way. Example : SOAP services that use http to transport data.

Web Service : a software system designed to support interoperable machine-to-machine interaction over a network.

The term “web service” is often referred to a self-contained, self-describing, modular application designed to be used and accessible by other software applications across the web. Once a web service is deployed, other applications and other web services can discover and invoke the deployed service. Other systems interact with the web service in a manner prescribed by its description.

A web service is one of the most common instances of an SOA implementation.

The technologies that make up the core of today's web services are:

1. Simple Object Access Protocol (SOAP)
2. Web Services Description Language (WSDL)
3. Universal Description, Discovery, and Integration (UDDI)

Simple Object Access Protocol (SOAP)

SOAP provides a standard packaging structure for transmission of XML documents over various Internet protocols, such as SMTP, HTTP, and FTP. By having such a standard message format, heterogeneous middleware systems can achieve interoperability i.e different systems can communicate and share services regardless of their underlying implementation.

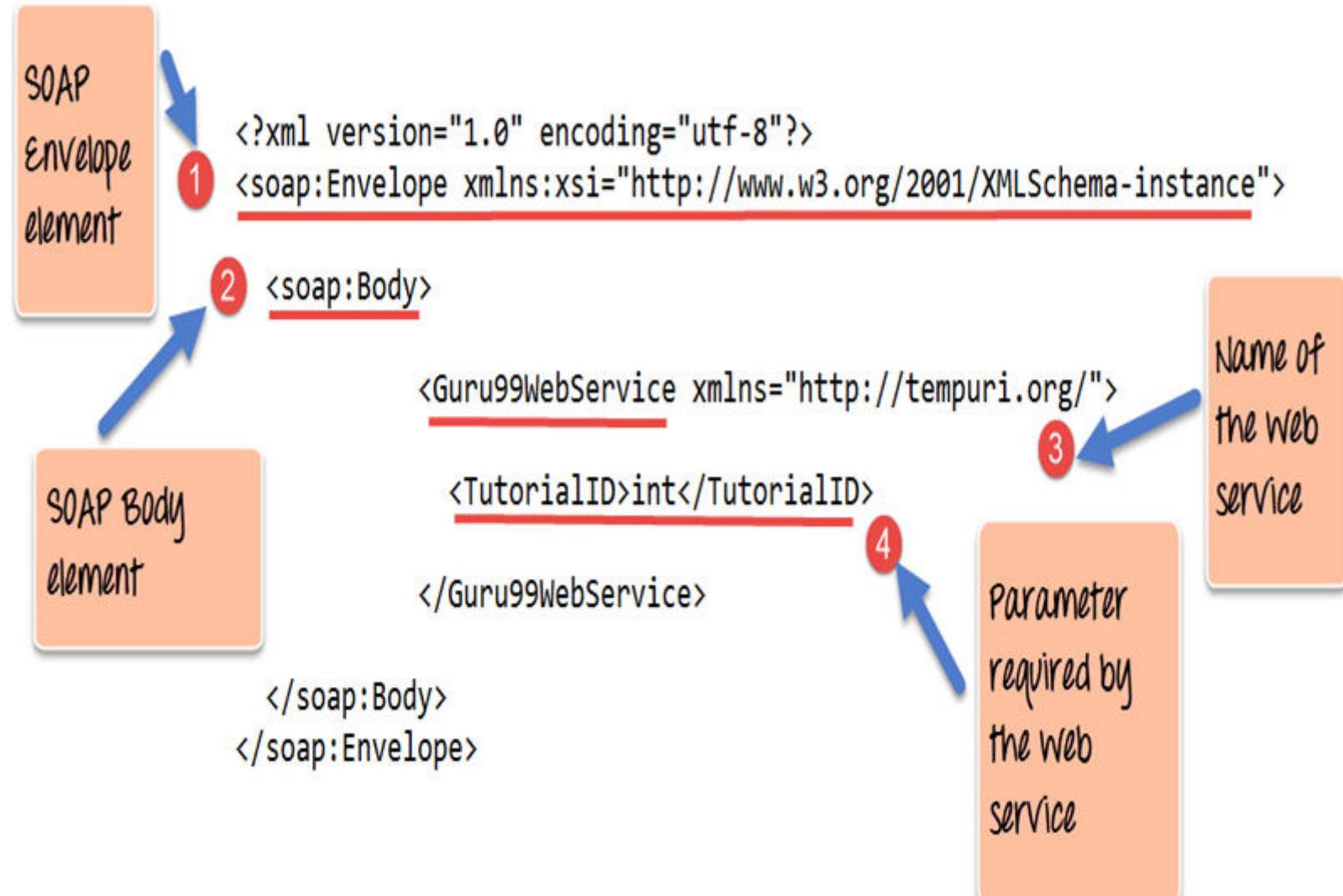
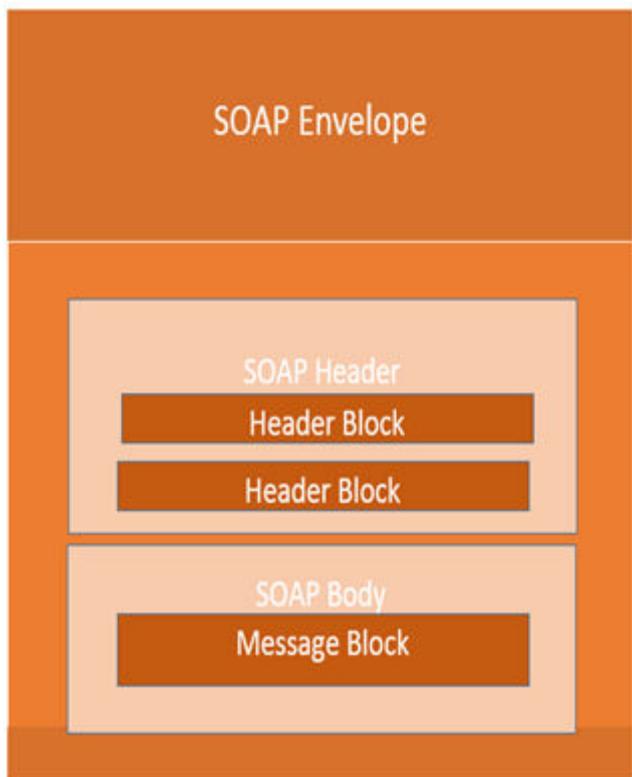
A SOAP message consists of a root element called *envelope*, The envelope element is the mandatory element in the SOAP message and is used to encapsulate all of the data in the SOAP message. It contains :

1. A *Header* element that contains header information – The header element can contain information such as authentication credentials which can be used by the calling application. It can be extended by intermediaries with additional application-level elements such as routing information ,transaction management, message parsing instructions.
2. a *body* element that carries the payload of the message.

The content of the payload will be marshaled by the sender's SOAP engine and unmarshaled at the receiver side, based on the XML schema that describes the structure of the SOAP message.

Ref - <https://www.guru99.com/soap-simple-object-access-protocol.html>

SOAP Message Structure :



Web Services Description Language (WSDL)

WSDL describes the interface, a set of operations supported by a web service in a standard format. It standardizes the representation of input and output parameters of its operations as well as the service's protocol binding, the way in which the messages will be transferred on the wire. Using WSDL enables disparate clients to automatically understand how to interact with a web service.

Universal Description, Discovery, and Integration (UDDI)

UDDI provides a global registry for advertising and discovery of web services. Users can find web services by searching for names, identifiers, categories, or the specification implemented by the web service.

CLOUD COMPUTING

Web Services in action

(T2)

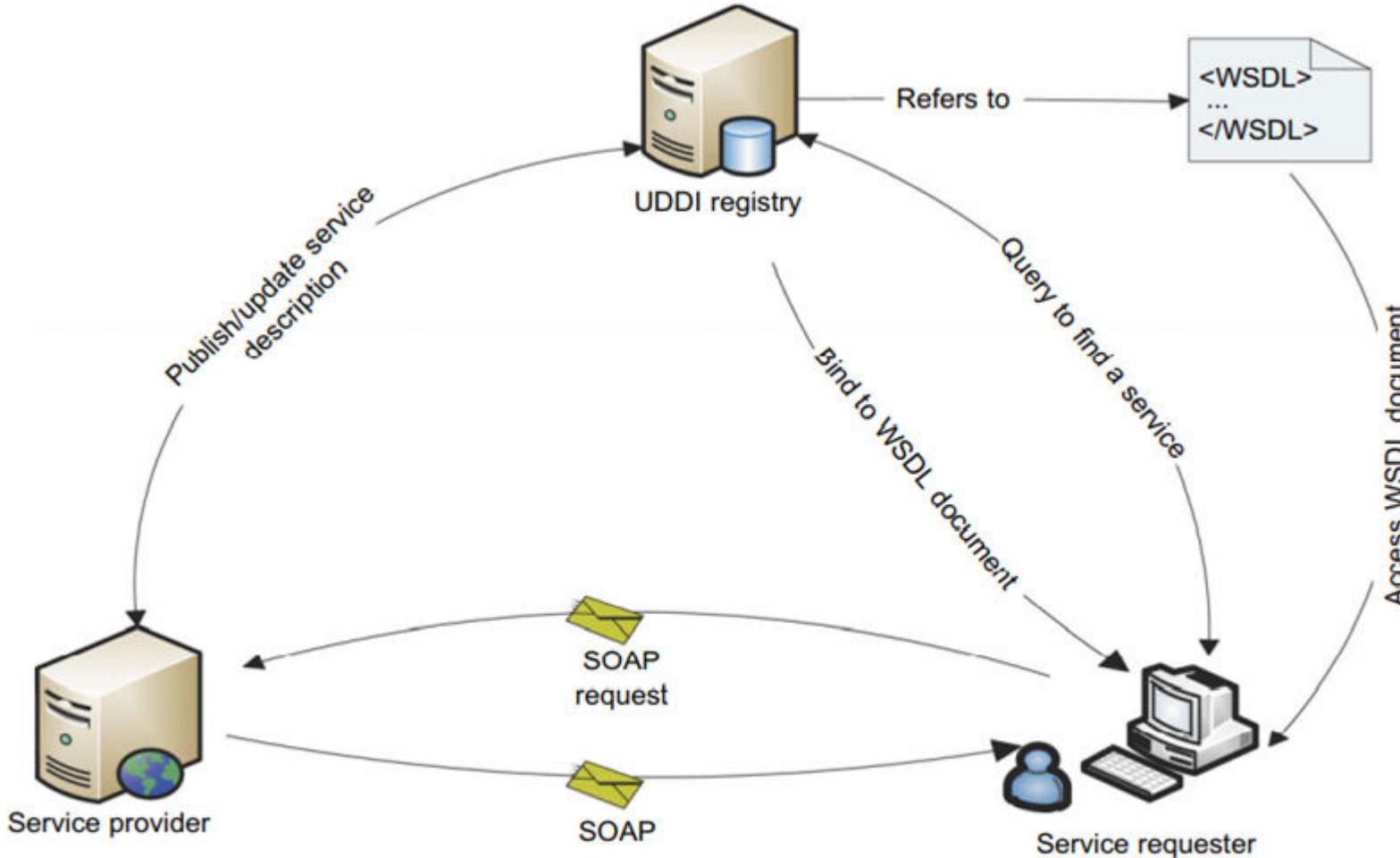


FIGURE 5.2

A simple web service interaction among provider, user, and the UDDI registry.

Refer to Figure 5.2(previous slide):

Step 1 : The service provider publishes its location and description to the UDDI registry.

Step 2: The service requester queries the UDDI registry using names, identifiers, or the specification. The UDDI registry finds the corresponding service and returns the WSDL document of the service.

Step 3: The service requester reads the WSDL document and forms a SOAP message binding to all constraints in the WSDL document.

Step 4: This SOAP message is sent to the web service as the body of an HTTP/SMTP/FTP request.

Step 5: The web service unpacks the SOAP request and converts it into a command that the application can understand and executes the command.

Step 6: The web service packages the response into another SOAP message, which it sends back to the client program in response to its HTTP/SMTP/FTP request

A web service protocol stack is a protocol stack (a stack of computer networking protocols) that is used to define, locate, implement, and make Web services interact with each other. A Web service protocol stack typically stacks four protocols:

1. **Transport Protocol:** responsible for transporting messages between network applications and includes protocols such as HTTP, SMTP, FTP, as well as the more recent Blocks Extensible Exchange Protocol (BEEP).
2. **Messaging Protocol:** responsible for encoding messages in a common XML format so that they can be understood at either end of a network connection. Currently, this area includes such protocols as XML-RPC, WS-Addressing, and SOAP.
3. **Description Protocol:** used for describing the public interface to a specific Web service. The WSDL interface format is typically used for this purpose.
4. **Discovery Protocol:** centralizes services into a common registry so that network Web services can publish their location and description, and makes it easy to discover what services are available on the network. Universal Description Discovery and Integration (UDDI) was intended for this purpose, but it has not been widely adopted.



THANK YOU

K.S.Srinivas
srinivasks@pes.edu
+91 80 2672 1983 Extn 701



CLOUD COMPUTING

Platform as a Service(PaaS) Programming Model

K.S.Srinivas

Department of Computer Science and Engineering

CLOUD COMPUTING

Platform as a Service(PaaS) Programming Model

Srinivas K S.

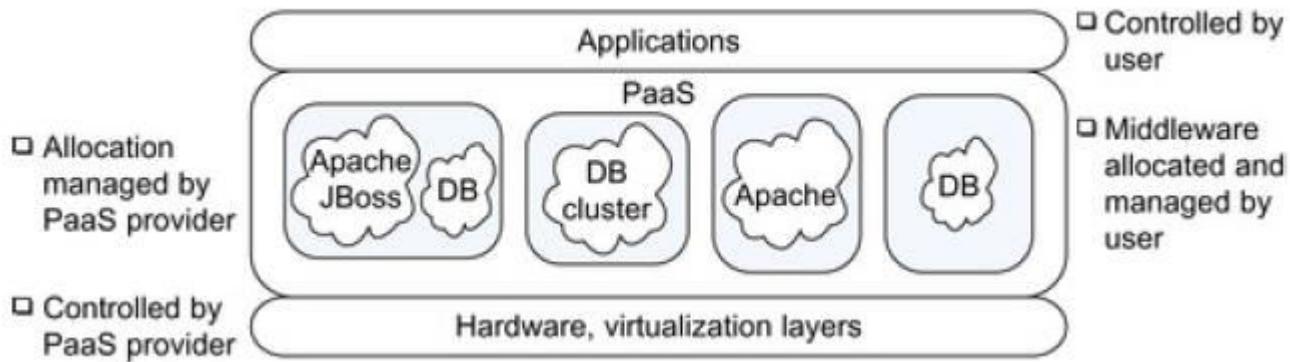
Associate Professor, Department of Computer Science

Platform as a Service(PaaS) Programming Model

<https://azure.microsoft.com/en-in/overview/what-is-paas/>

Platform as a service (PaaS) is a complete development and deployment environment in the cloud, with resources that enable you to deliver everything from simple cloud-based apps to sophisticated, cloud-enabled enterprise applications. You purchase the resources you need from a cloud service provider on a pay-as-you-go basis and access them over a secure Internet connection.

PaaS abstraction **provides a platform built on top of the abstracted hardware** that can be used by developers to create cloud applications. This platform is delivered via the web, giving developers the freedom to concentrate on building the software without having to worry about operating systems, software updates, storage, or infrastructure.



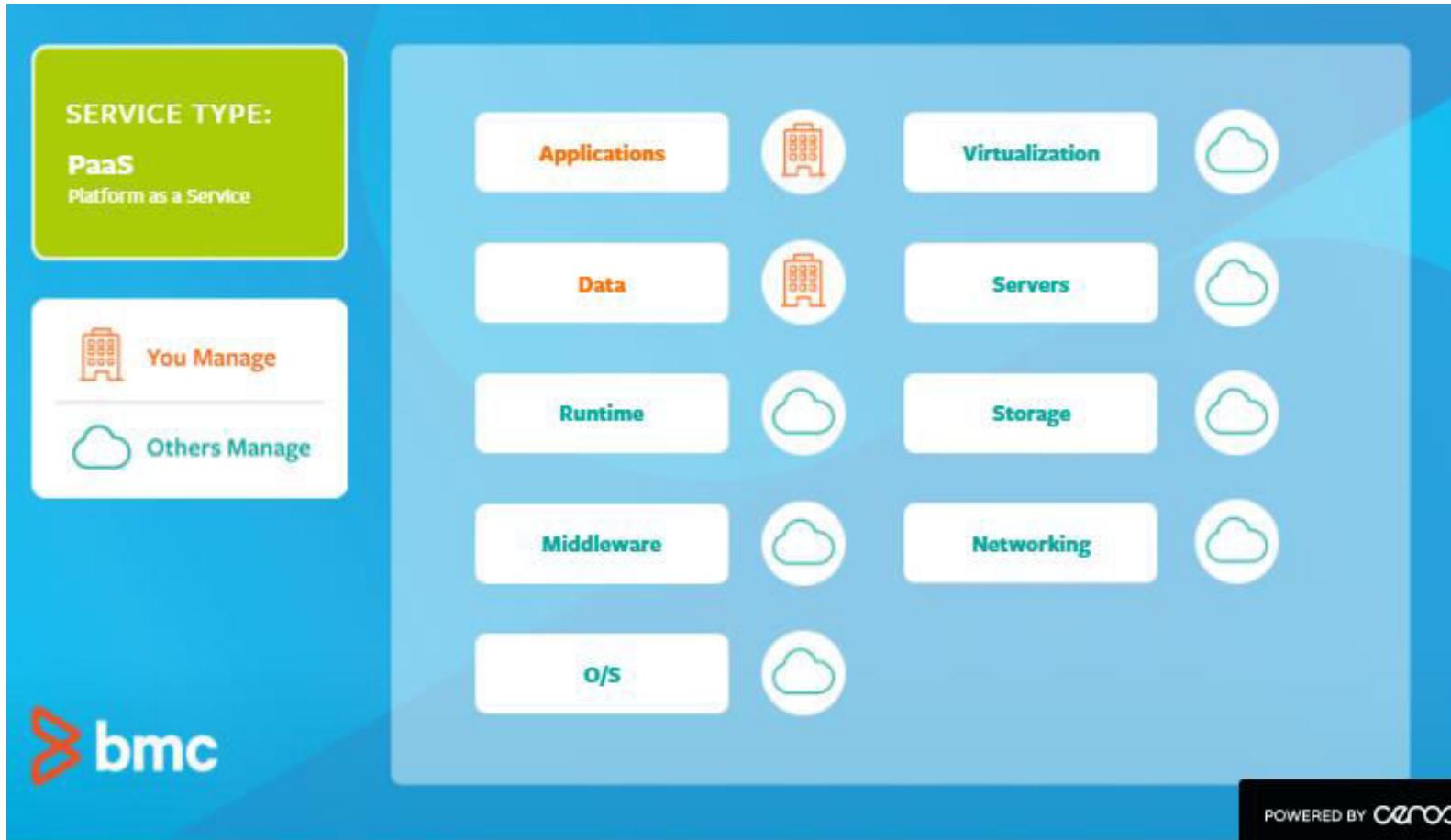
Platform as a Service(PaaS) Programming Model

Like IaaS, PaaS includes infrastructure—servers, storage and networking—but also middleware, development tools, business intelligence (BI) services, database management systems and more. PaaS is designed to support the complete web application lifecycle: building, testing, deploying, managing and updating.

The hardware, as well as any **mapping of hardware to virtual resources, such as virtual servers, is controlled by the PaaS provider**. The cloud user can configure and build on top of this middleware, such as define a new database table in a database

CLOUD COMPUTING

Platform as a Service(PaaS) Programming Model



Ref - <https://www.bmc.com/blogs/saas-vs-paas-vs-iaas-whats-the-difference-and-how-to-choose/#ref3>

1. **Faster time to market:** With PaaS, there's no need to purchase and install the hardware and software you'll use to build and maintain your application development platform and no need for development teams to wait while you do this. You simply tap into the cloud service provider's PaaS resources and begin developing immediately.

1. **Faster, easier, less-risky adoption of a wider range of resources:** PaaS platforms typically include access to a greater variety of choices up and down the application development stack—operating systems, middleware, and databases, and tools such as code libraries and app components—than you can affordably or practically maintain on-premises. It also lets you test new operating systems, languages, and tools without risk—that is, without having to invest in the infrastructure required to run them.

1. **Develop for multiple platforms—including mobile—more easily.** Some service providers give you development options for multiple platforms, such as computers, mobile devices and browsers making cross-platform apps quicker and easier to develop.

4. **Easy, cost-effective scalability:** If an application developed and hosted on-premises starts getting more traffic, you'll need to purchase more computing, storage, and even network hardware to meet the demand, which you may not be able to do quickly enough and can be wasteful (since you typically purchase more than you need). With PaaS, you can scale on-demand by purchasing just the amount of additional capacity you need.
4. **Support geographically distributed development teams:** Because the development environment is accessed over the Internet, development teams can work together on projects even when team members are in remote locations.
4. **Efficiently manage the application lifecycle :** PaaS provides all of the capabilities that you need to support the complete web application lifecycle: building, testing, deploying, managing and updating within the same integrated environment.
4. **Lower costs:** Because there's no infrastructure to build, your upfront costs are lower. Costs are also lower and more predictable because most PaaS providers charge customers based on usage.

1. **Development framework:** PaaS provides a framework that developers can build upon to develop or customise cloud-based applications. PaaS lets developers create applications using built-in software components. Cloud features such as scalability, high-availability and multi-tenant capability are included, reducing the amount of coding that developers must do.
1. **Analytics or business intelligence:** Tools provided as a service with PaaS allow organisations to analyse and mine their data, finding insights and patterns and predicting outcomes to improve forecasting, product design decisions, investment returns and other business decisions.
1. PaaS can **streamline workflows when multiple developers** are working on the same development project. If other vendors must be included, PaaS can provide great speed and flexibility to the entire process. PaaS is particularly beneficial if you need to create customized applications.

4. **API development and management:** You can use PaaS to develop, run, manage, and secure application programming interfaces (APIs) and microservices.

4. **Internet of Things (IoT):** PaaS can support the broad range of application environments, programming languages, and tools used for IoT deployments.

1. **Operational limitation(Lack of control):** Customized cloud operations with management automation workflows may not apply to PaaS solutions, as the platform tends to limit operational capabilities for end users. Although this is intended to reduce the operational burden on end users, the loss of operational control may affect how PaaS solutions are managed, provisioned, and operated.
1. **Vendor lock-in:** Business and technical requirements that drive decisions for a specific PaaS solution may not apply in the future. If the vendor has not provisioned convenient migration policies, switching to alternative PaaS options may not be possible without affecting the business.
1. **Runtime issues:** In addition to limitations associated with specific apps and services, PaaS solutions may not be optimized for the language and frameworks of your choice. Specific framework versions may not be available or perform optimally with the PaaS service. Customers may not be able to develop custom dependencies with the platform.

PaaS Limitations & Concerns

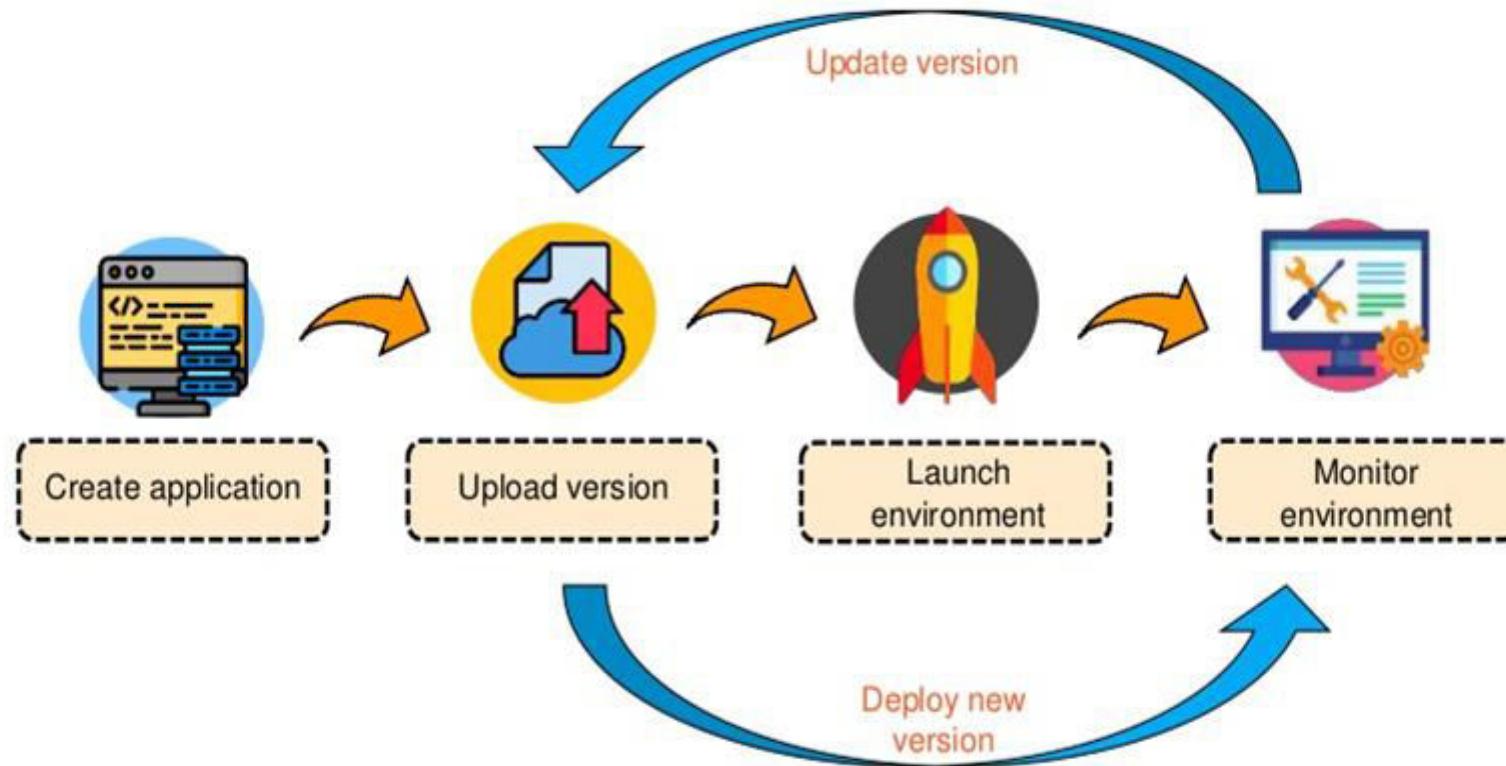
4. **Data security:** Organizations can run their own apps and services using PaaS solutions, but the data residing in third-party, vendor-controlled cloud servers poses security risks and concerns. Your security options may be limited as customers may not be able to deploy services with specific hosting policies.
4. **Integrations:** The complexity of connecting the data stored within an onsite data center or off-premise cloud is increased, which may affect which apps and services can be adopted with the PaaS offering. Particularly when not every component of a legacy IT system is built for the cloud, integration with existing services and infrastructure may be a challenge.
4. **Customization of legacy systems:** PaaS may not be a plug-and-play solution for existing legacy apps and services. Instead, several customizations and configuration changes may be necessary for legacy systems to work with the PaaS service. The resulting customization can result in a complex IT system that may limit the value of the PaaS investment altogether.

- Amazon Web Services : Elastic Beanstalk.
- Microsoft Azure : Azure DevOps.
- Google Cloud : Google App Engine.

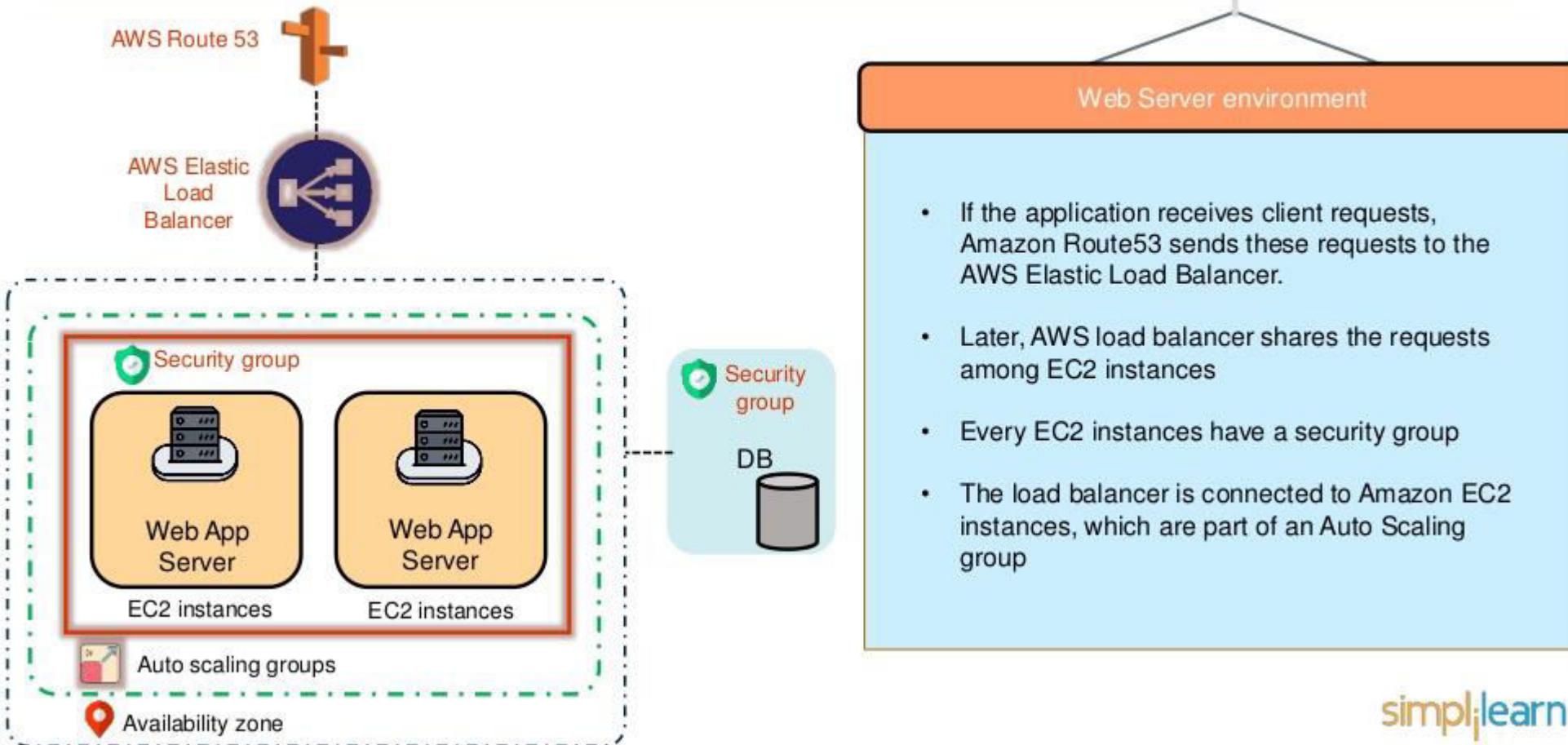
Table 4.2 Five Public Cloud Offerings of PaaS [10,18]

Cloud Name	Languages and Developer Tools	Programming Models Supported by Provider	Target Applications and Storage Option
Google App Engine	Python, Java, and Eclipse-based IDE	MapReduce, web programming on demand	Web applications and BigTable storage
Salesforce.com's Force.com	Apex, Eclipse-based IDE, web-based Wizard	Workflow, Excel-like formula, Web programming on demand	Business applications such as CRM
Microsoft Azure	.NET, Azure tools for MS Visual Studio	Unrestricted model	Enterprise and web applications
Amazon Elastic MapReduce	Hive, Pig, Cascading, Java, Ruby, Perl, Python, PHP, R, C++	MapReduce	Data processing and e-commerce
Aneka	.NET, stand-alone SDK	Threads, task, MapReduce	.NET enterprise applications, HPC

How does Elastic Beanstalk in AWS work?

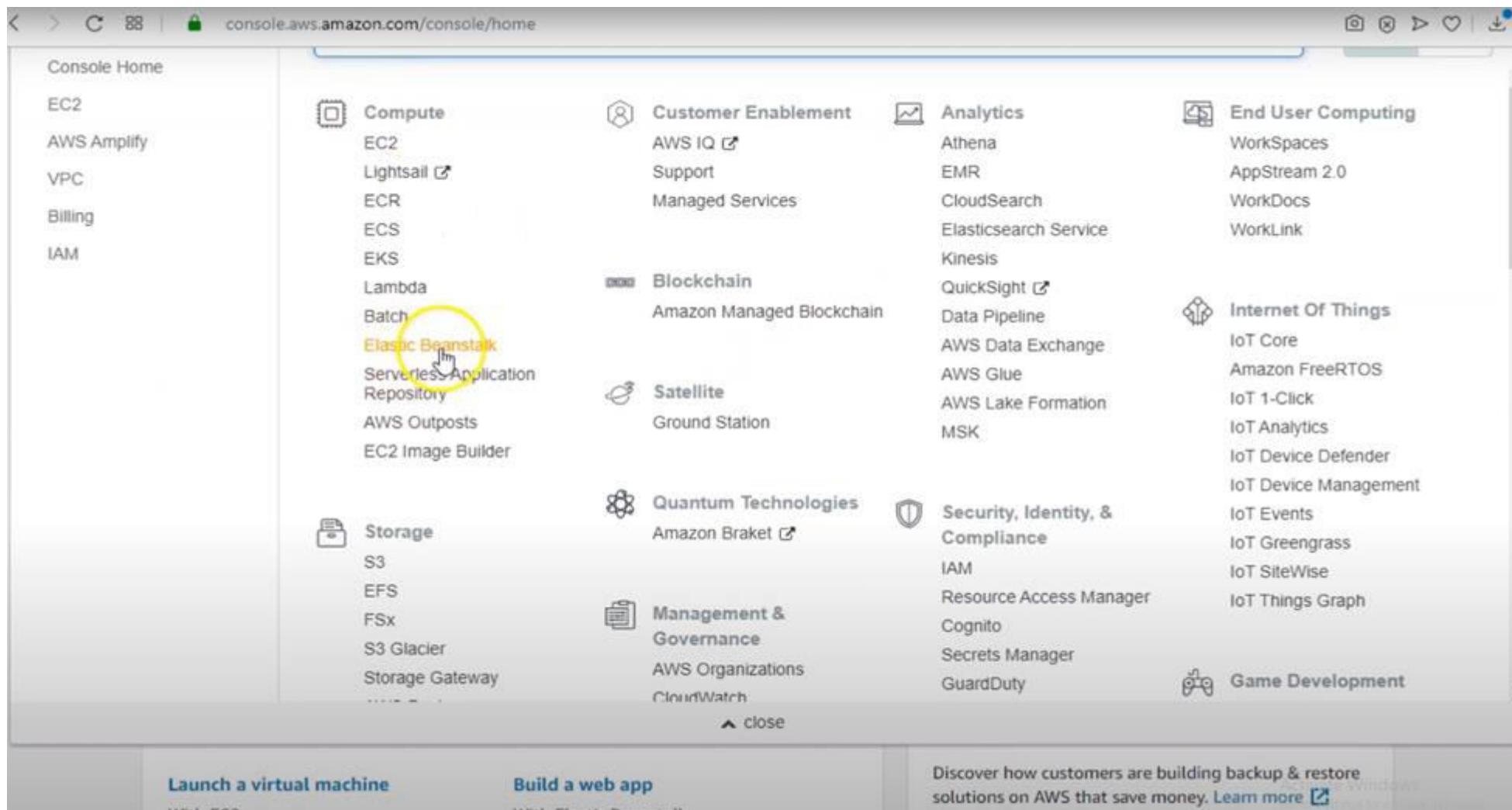


Architecture of AWS Elastic Beanstalk



PaaS Demo : AWS Elastic Beanstalk

Step 1: Log into AWS, go to services and select Elastic Beanstalk



The screenshot shows the AWS Management Console home page. On the left, there is a sidebar with links to EC2, AWS Amplify, VPC, Billing, and IAM. The main area displays various AWS services in a grid. The 'Compute' section includes EC2, Lightsail, ECR, ECS, EKS, Lambda, Batch, and Elastic Beanstalk. The 'Storage' section includes S3, EFS, FSx, S3 Glacier, and Storage Gateway. Other sections like Customer Enablement, Analytics, End User Computing, Blockchain, Satellite, Quantum Technologies, Management & Governance, Security, Identity, & Compliance, and Game Development also contain several services each. At the bottom of the page, there are three promotional cards: 'Launch a virtual machine with EC2', 'Build a web app with Elastic Beanstalk', and 'Discover how customers are building backup & restore solutions on AWS that save money. Learn more'.

PaaS Demo : AWS Elastic Beanstalk

Step 2: Click on Get started and then create new application

Welcome to AWS Elastic Beanstalk

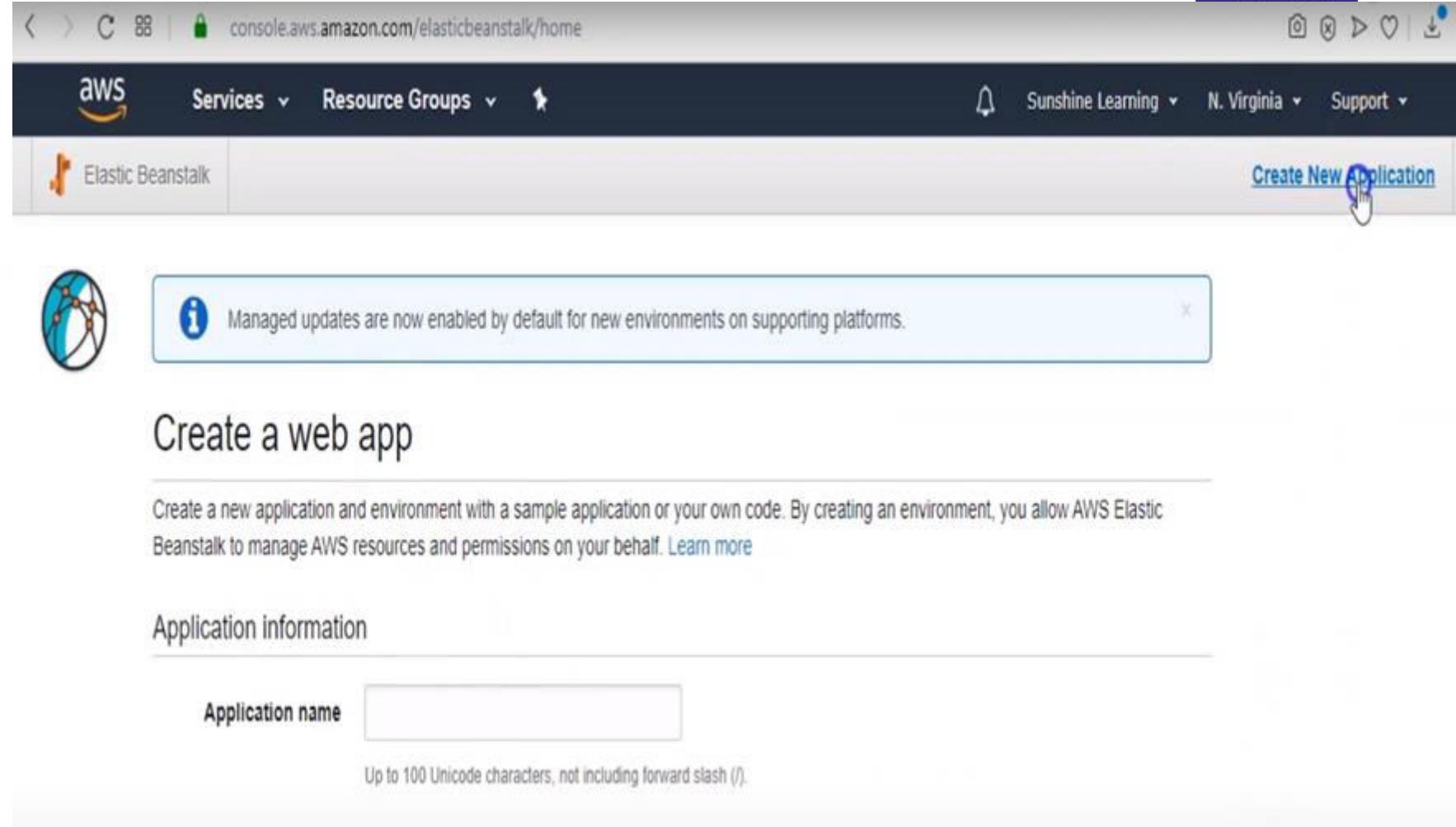
With Elastic Beanstalk, you can **deploy**, **monitor**, and **scale** an application quickly and easily. Let us do the heavy lifting so you can focus on your business.

To deploy your **existing web application**, create an [application source bundle](#) and then [create a new application](#). If you're using **Git** and would prefer to use it with our command line tool, please see [Getting Started with the EB CLI](#).

To deploy a **sample application**, click **Get started**, choose a name, select a platform and click **Create app**.

By launching the sample application, you allow AWS Elastic Beanstalk to administer AWS resources and necessary permissions on your behalf. [Learn more](#)

Get started



console.aws.amazon.com/elasticbeanstalk/home

aws Services Resource Groups Sunshine Learning N. Virginia Support

Elastic Beanstalk Create New Application

Managed updates are now enabled by default for new environments on supporting platforms.

Create a web app

Create a new application and environment with a sample application or your own code. By creating an environment, you allow AWS Elastic Beanstalk to manage AWS resources and permissions on your behalf. [Learn more](#)

Application information

Application name

Up to 100 Unicode characters, not including forward slash (/).

PaaS Demo : AWS Elastic Beanstalk

Step 3: Enter application name, description and click on create

Create New Application

Application Name XYZ
Maximum length of 100 characters, not including forward slash (/).

Description Demo App
Maximum length of 200 characters.

Tags

Apply up to 50 tags. You can use tags to group and filter your resources. A tag is a key-value pair. The key must be unique within the resource and is case-sensitive.
[Learn more](#)

Key (127 characters maximum)	Value (255 characters maximum)
<input type="text"/>	<input type="text"/>

50 remaining

CLOUD COMPUTING

PaaS Demo : AWS Elastic Beanstalk



Step 4: Observer we don't have an environment for application, click on create one now.

A screenshot of the AWS Elastic Beanstalk console. The URL in the browser is "console.aws.amazon.com/elasticbeanstalk/home". The top navigation bar shows "aws", "Services", "Resource Groups", a notification bell, "Sunshine Learning", "N. Virginia", and "Support". Below the navigation is a secondary menu with "Elastic Beanstalk" selected, "XYZ", and "Create New Application". The main content area is titled "All Applications > XYZ". On the left, there's a sidebar with "Environments" (which is highlighted in orange), "Application versions", and "Saved configurations". The main content area displays a message: "No environments currently exist for this application. [Create one now.](#)". A yellow circle highlights the "Create one now." link.

All Applications > XYZ

Actions ▾

Environments

Application versions

Saved configurations

No environments currently exist for this application. [Create one now.](#)

PaaS Demo : AWS Elastic Beanstalk

Step 5: Enter the domain name for the application

The screenshot shows the 'Create a web server environment' wizard on the AWS Elastic Beanstalk console. The URL in the browser is `console.aws.amazon.com/elasticbeanstalk/home`. The page title is 'Create a web server environment'. A sub-instruction says: 'Launch an environment with a sample application or your own code. By creating an environment, you allow AWS Elastic Beanstalk to manage AWS resources and permissions on your behalf. [Learn more](#)'.

Environment information

Choose the name, subdomain, and description for your environment. These cannot be changed later.

Application name XYZ

Environment name Xyz-env

Domain xyzabc.us-east-1.elasticbeanstalk.com **Check availability**

xyzabc.us-east-1.elasticbeanstalk.com **is available.**

Description

Base configuration

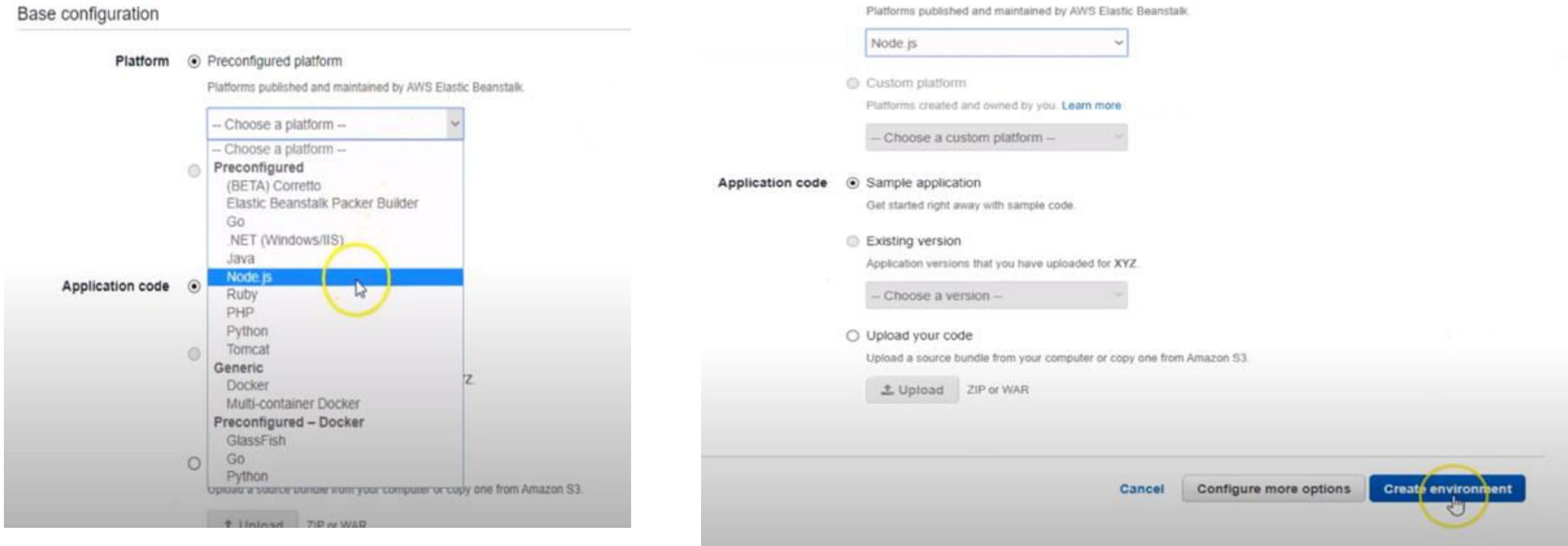
Platform Preconfigured platform
Platforms published and maintained by AWS Elastic Beanstalk

Choose a platform

Custom platform

PaaS Demo : AWS Elastic Beanstalk

Step 6: Enter the platform(Node.js in this case) and click on create environment



The screenshot shows the 'Base configuration' step of the AWS Elastic Beanstalk environment creation wizard. It is divided into two main sections: 'Platform' and 'Application code'.

Platform: A dropdown menu titled 'Platforms published and maintained by AWS Elastic Beanstalk' contains the following options:

- Choose a platform --
- Choose a platform --
- Preconfigured**
 - (BETA) Corretto
 - Elastic Beanstalk Packer Builder
 - Go
 - NET (Windows/IIS)
 - Java
 - Node.js** (highlighted with a yellow circle)
 - Ruby
 - PHP
 - Python
 - Tomcat
- Generic**
 - Docker
 - Multi-container Docker
- Preconfigured – Docker**
 - GlassFish
 - Go
 - Python

Application code: A dropdown menu titled 'Platforms published and maintained by AWS Elastic Beanstalk' contains the following options:

- Node.js
- Custom platform
 - Choose a custom platform --

Application code (radio buttons):

- Sample application**
Get started right away with sample code.
- Existing version**
Application versions that you have uploaded for XYZ.
 - Choose a version --
- Upload your code**
Upload a source bundle from your computer or copy one from Amazon S3.
 - Upload ZIP or WAR**

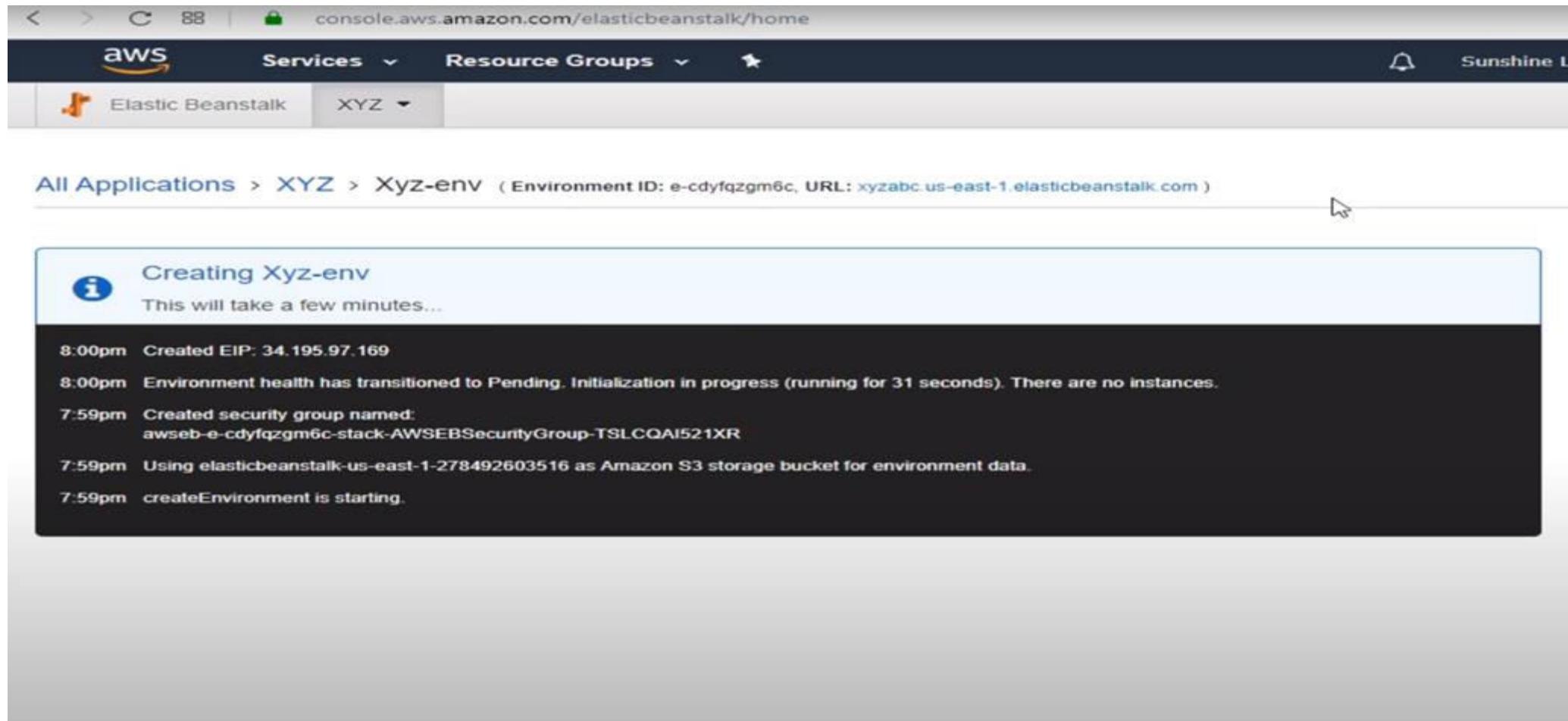
Buttons at the bottom:

- Cancel
- Configure more options
- Create environment** (highlighted with a yellow circle)

CLOUD COMPUTING

PaaS Demo : AWS Elastic Beanstalk

Step 7: Elastic Beanstalk will create the environment and we can observe the logs on the dashboard.
Click on the application name(XYZ in this case).



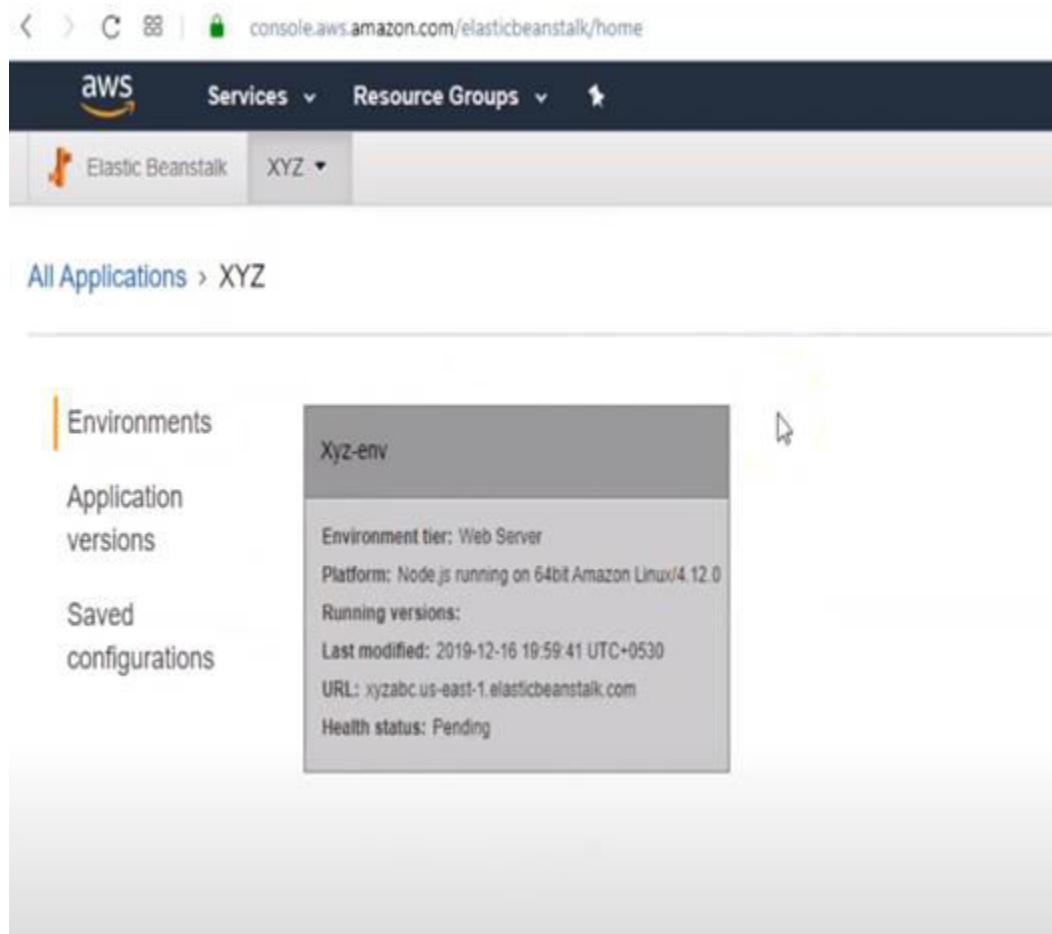
The screenshot shows the AWS Elastic Beanstalk console interface. At the top, the URL is `console.aws.amazon.com/elasticbeanstalk/home`. The navigation bar includes the AWS logo, Services dropdown, Resource Groups dropdown, and a bell icon for notifications. Below the navigation bar, there are two tabs: "Elastic Beanstalk" and "XYZ". The "XYZ" tab is selected, indicated by a dropdown arrow next to its name.

The main content area displays the progress of creating an environment named "XYZ-env". A message at the top says "Creating Xyz-env" with an information icon and the note "This will take a few minutes...". Below this, a log window shows the following events:

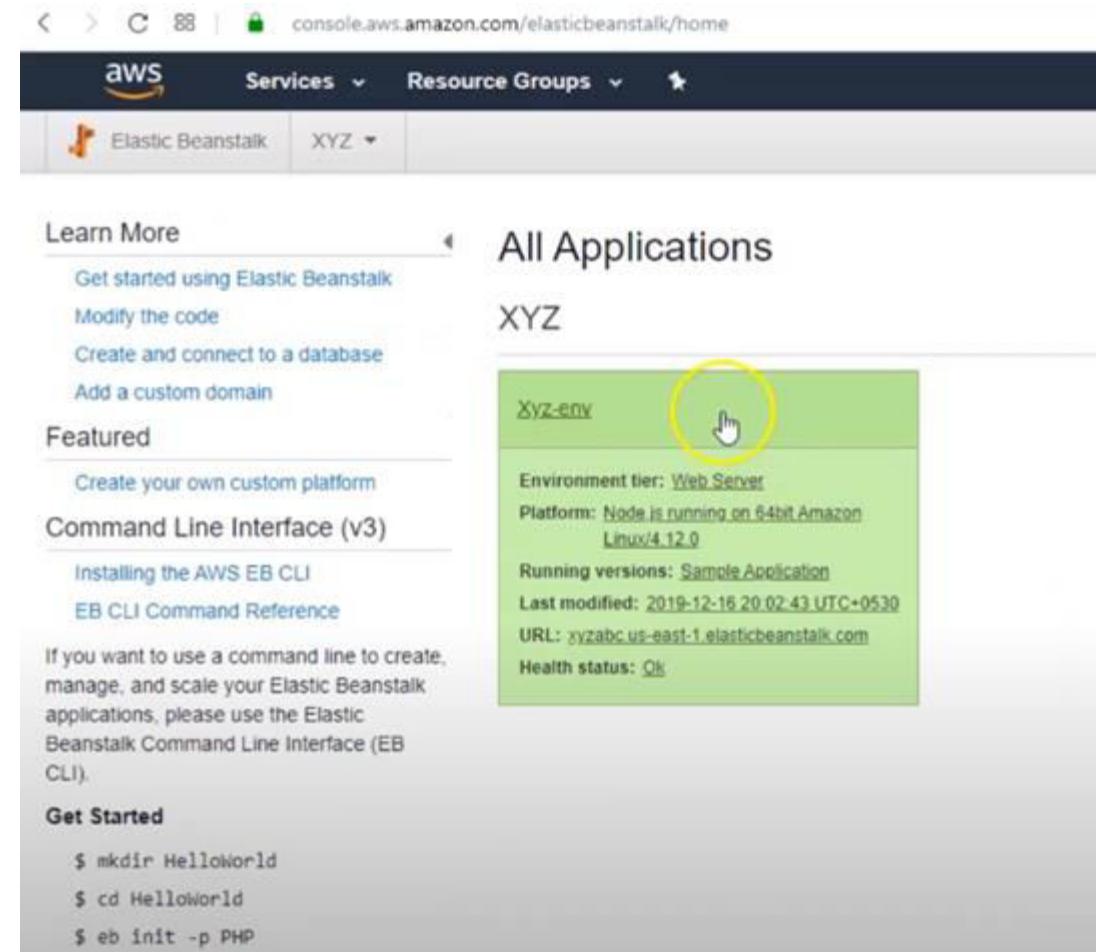
- 8:00pm Created EIP: 34.195.97.169
- 8:00pm Environment health has transitioned to Pending. Initialization in progress (running for 31 seconds). There are no instances.
- 7:59pm Created security group named: awseb-e-cdyfqzgm6c-stack-AWSEBSecurityGroup-TSLCQAI521XR
- 7:59pm Using elasticbeanstalk-us-east-1-278492603516 as Amazon S3 storage bucket for environment data.
- 7:59pm createEnvironment is starting.

PaaS Demo : AWS Elastic Beanstalk

Step 8: Observe the colour of the box depicting the environment. Grey colour indicates the build is in progress. The green colour indicates the environment is up and running. Once the environment is ready click on it.



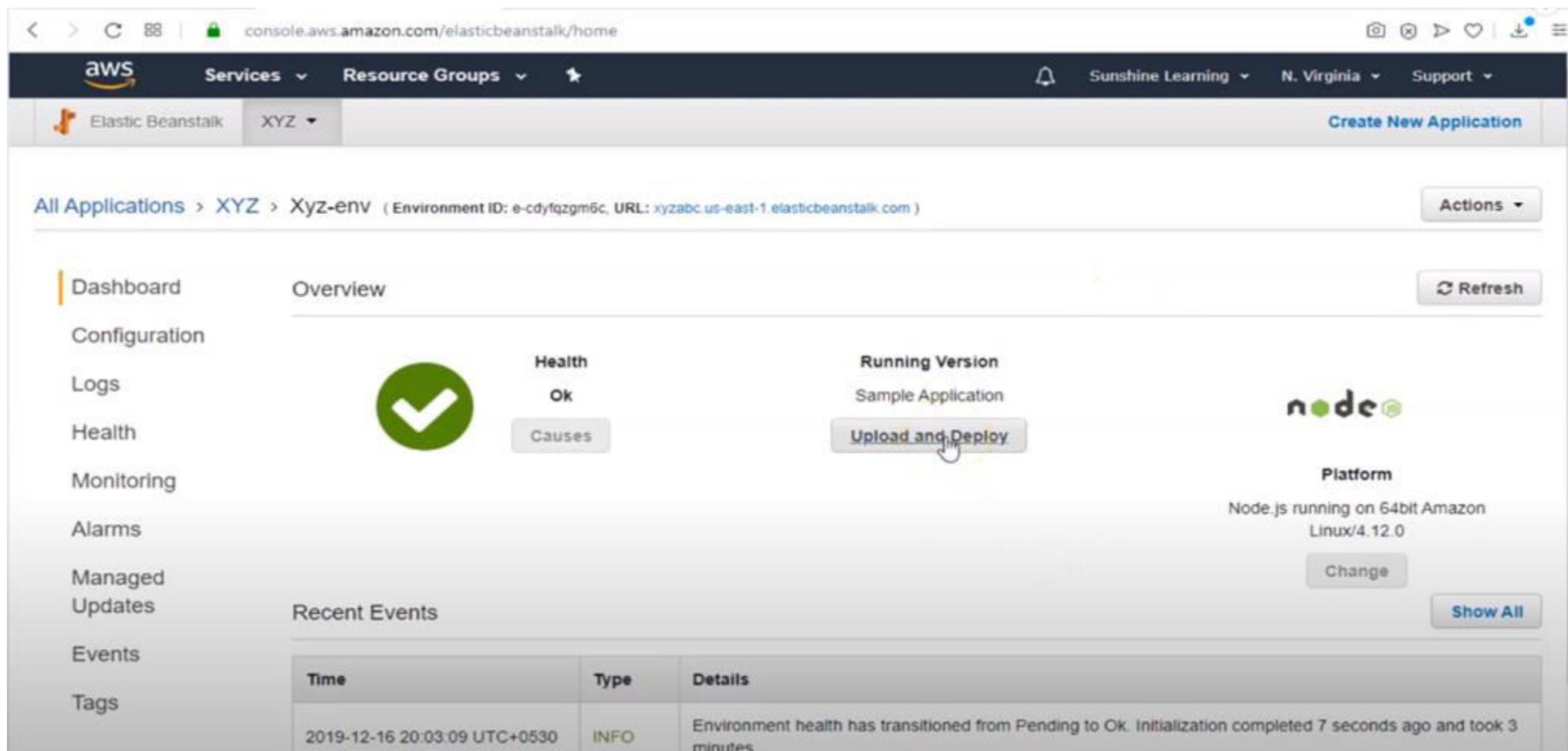
The screenshot shows the AWS Elastic Beanstalk console. The URL in the address bar is `console.aws.amazon.com/elasticbeanstalk/home`. The navigation bar includes the AWS logo, Services dropdown, and Resource Groups dropdown. Under the Services dropdown, 'Elastic Beanstalk' is selected. The current view is for the 'XYZ' resource group. On the left sidebar, there are links for 'Environments', 'Application versions', and 'Saved configurations'. The main content area displays the details for the 'Xyz-env' environment. It shows the environment tier as 'Web Server', the platform as 'Node.js running on 64bit Amazon Linux/4.12.0', and the last modified date as '2019-12-16 19:59:41 UTC+0530'. The URL listed is `xyzabc.us-east-1.elasticbeanstalk.com`. The 'Health status' is listed as 'Pending'.



The screenshot shows the AWS Elastic Beanstalk console. The URL in the address bar is `console.aws.amazon.com/elasticbeanstalk/home`. The navigation bar includes the AWS logo, Services dropdown, and Resource Groups dropdown. Under the Services dropdown, 'Elastic Beanstalk' is selected. The current view is for the 'XYZ' resource group. On the left sidebar, there are links for 'Learn More' (Get started using Elastic Beanstalk, Modify the code, Create and connect to a database, Add a custom domain) and 'Featured' (Create your own custom platform, Command Line Interface (v3), Installing the AWS EB CLI, EB CLI Command Reference). The main content area displays the details for the 'Xyz-env' environment. It shows the environment tier as 'Web Server', the platform as 'Node.js running on 64bit Amazon Linux/4.12.0', and the last modified date as '2019-12-16 20:02:43 UTC+0530'. The URL listed is `xyzabc.us-east-1.elasticbeanstalk.com`. The 'Health status' is listed as 'Ok'. A yellow circle highlights the environment name 'Xyz-env'.

PaaS Demo : AWS Elastic Beanstalk

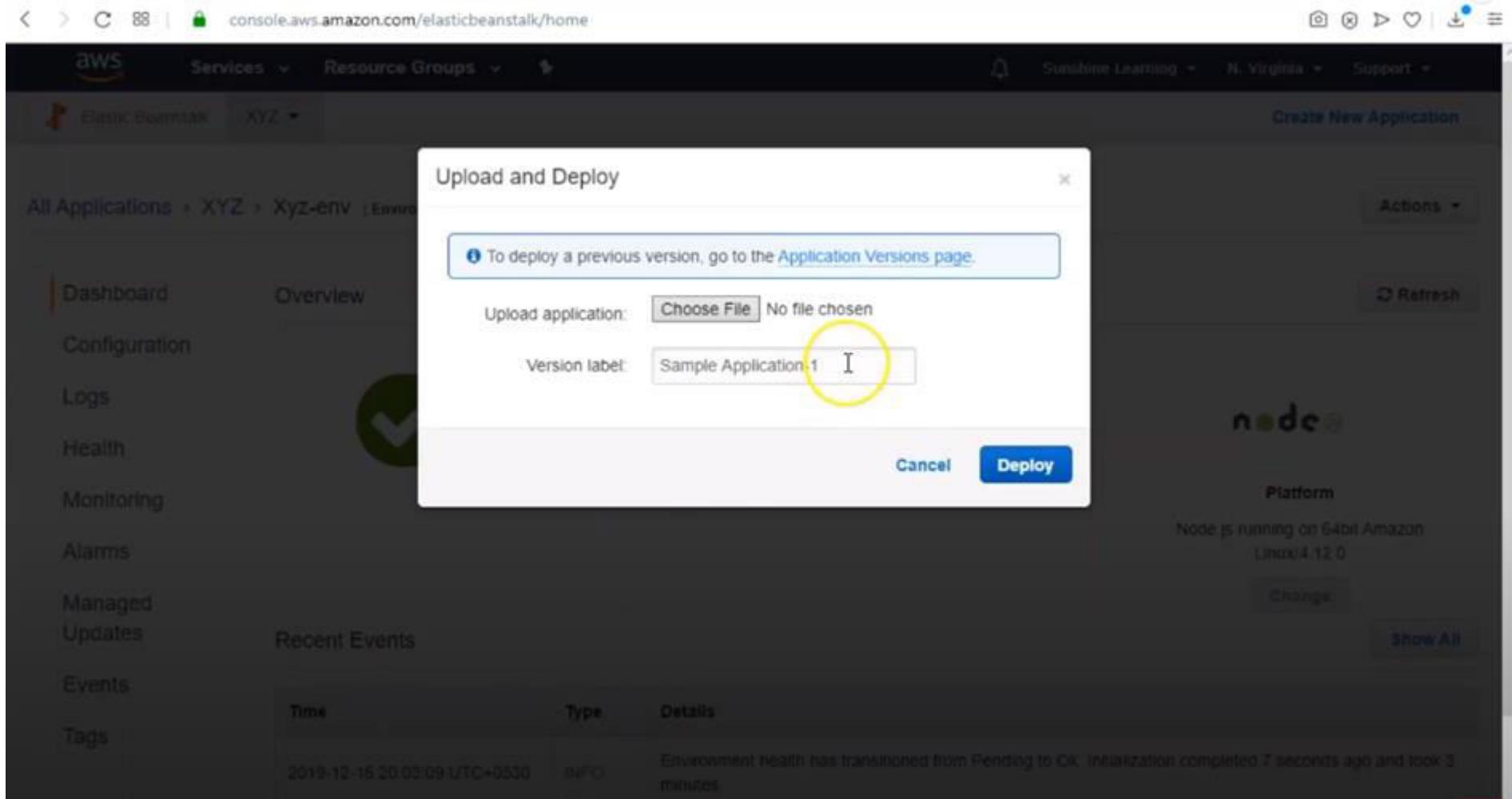
Step 9: We can see a dashboard giving an overview of the environment. Observe the health status and the selected platform. Click on Upload and Deploy



The screenshot shows the AWS Elastic Beanstalk Overview page for the 'XYZ' application environment 'Xyz-env'. The left sidebar lists navigation options: Dashboard (selected), Overview, Configuration, Logs, Health (with a green checkmark icon), Monitoring, Alarms, Managed Updates, Events, and Tags. The main content area displays the 'Overview' tab. It includes sections for 'Health' (status: Ok, causes: none), 'Running Version' (Sample Application), and 'Platform' (Node.js running on 64bit Amazon Linux/4.12.0). A prominent 'Upload and Deploy' button is highlighted with a mouse cursor. Below these, the 'Recent Events' section shows a single entry: '2019-12-16 20:03:09 UTC+0530' (INFO) - 'Environment health has transitioned from Pending to Ok. Initialization completed 7 seconds ago and took 3 minutes.'

PaaS Demo : AWS Elastic Beanstalk

Step 10: Upload your application file and deploy.



The screenshot shows the AWS Elastic Beanstalk console. A modal window titled "Upload and Deploy" is open. It contains a message: "To deploy a previous version, go to the [Application Versions page](#)". Below this is a "Upload application:" section with a "Choose File" button and a message "No file chosen". Underneath is a "Version label:" section with an input field containing "Sample Application 1" and a dropdown arrow. At the bottom of the modal are "Cancel" and "Deploy" buttons. The background shows the main Elastic Beanstalk dashboard for the "XYZ" environment, which includes sections for Dashboard, Configuration, Logs, Health, Monitoring, Alarms, Managed Updates, Recent Events, and Tags. The "Recent Events" table has one entry: "Environment health has transitioned from Pending to OK: Initialization completed 7 seconds ago and took 3 minutes".



THANK YOU

K.S.Srinivas
srinivasks@pes.edu
+91 80 2672 1983 Extn 701



CLOUD COMPUTING

COMMUNICATION USING MESSAGE QUEUES

K.S.Srinivas

Department of Computer Science and Engineering

CLOUD COMPUTING

Communication using Message Queues

Srinivas K S.

Associate Professor, Department of Computer Science

“Inter-service communication is essential to consider when building micro-services.”

Monolithic Application

- Runs on a single process, with multiple components.
- Components communicate(pass data) with one another though language-level function calls.
- All components run within the same process — intra-process communication.

Micro-service Architecture

- Runs on multiple processes/services, usually across multiple servers.
- Each service instance is a process.
- Services interact using inter-process communication protocols.
- Protocols such as HTTP, AMQP, or a binary protocol like TCP.

Interaction styles

(<https://livebook.manning.com/book/microservices-patterns/chapter-3/15>)

Interaction styles can be categorised in two dimensions

First dimension

1. One-to-one : Each client request is processed by exactly one service
2. One-to-many : Each request is processed by multiple services

Second dimension

1. Synchronous : The client expects a timely response from the service and might even block while it waits.
2. Asynchronous : The client doesn't block, and the response, if any, isn't necessarily sent immediately.

Table 3.1 The various interaction styles can be characterized in two dimensions: one-to-one vs one-to-many and synchronous vs asynchronous.

	one-to-one	one-to-many
Synchronous	Request/response	—
Asynchronous	Asynchronous request/response One-way notifications	Publish/subscribe Publish/async responses

The following are the different types of one-to-one interactions

1. *Request/response*— A service client makes a request to a service and waits for a response. The client expects the response to arrive in a timely fashion. It might even block while waiting. This is an interaction style that generally results in services being tightly coupled.

3. *Asynchronous request/response*— A service client sends a request to a service, which replies asynchronously. The client doesn't block while waiting, because the service might not send the response for a long time.

4. *One-way notifications*— A service client sends a request to a service, but no reply is expected or sent.

One-to-many interaction

The following are the different types of one-to-many interactions

- *Publish/subscribe*— A client publishes a notification message, which is consumed by zero or more interested services.
- *Publish/async responses*— A client publishes a request message and then waits for a certain amount of time for responses from interested services.

Asynchronous vs Synchronous messaging

Advantages of Asynchronous messaging

- Reduced coupling: The message sender does not need to know about the consumer.
- Multiple subscribers: Using a pub/sub model, multiple consumers can subscribe to receive events.
- Failure isolation: If the consumer fails, the sender can still send messages. The messages will be picked up when the consumer recovers. Asynchronous messaging can handle intermittent downtime. Synchronous APIs, on the other hand, require the downstream service to be available or the operation fails.
- Load leveling: A queue can act as a buffer to level the workload, so that receivers can process messages at their own rate.

Asynchronous vs Synchronous messaging

Disadvantages of Asynchronous messaging

- Coupling with the messaging infrastructure: Using a particular messaging infrastructure may cause tight coupling with that infrastructure. It will be difficult to switch to another messaging infrastructure later.
- Latency: End-to-end latency for an operation may become high if the message queues fill up.
- Complexity: Handling asynchronous messaging is not a trivial task. For example, handling of duplicated messages, or correlating request and response messages using a separate response queue.
- Throughput: If message queues are used, each message requires at least one queue operation and one dequeue operation. Moreover, queue semantics generally require some kind of locking inside the messaging infrastructure.

There are two main paradigms for asynchronous communication :

1. Message Queues
2. Event streams (Pub/Sub)

A message queue is a form of asynchronous service-to-service communication used in serverless and microservices architectures.

Messages are stored on the queue until they are processed and deleted.

Each message is processed only once, by a single consumer.

Message queues can be used to decouple heavyweight processing, to buffer or batch work, and to smooth spiky workloads.

Examples - Apache ActiveMQ, RabbitMQ, Apache Kafka



Message Queues (contd.)

A message queue provides a lightweight buffer which temporarily stores messages, and endpoints that allow software components to connect to the queue in order to send and receive messages.

The messages are usually small, and can be things like requests, replies, error messages, or just plain information.

To send a message, a component called a producer adds a message to the queue. The message is stored on the queue until another component called a consumer retrieves the message and does something with it.

Many producers and consumers can use the queue, but each message is processed only once, by a single consumer. For this reason, this messaging pattern is often called one-to-one, or point-to-point, communications.

When a message needs to be processed by more than one consumer, message queues can be combined with Pub/Sub messaging in a fanout design pattern.

CLOUD COMPUTING

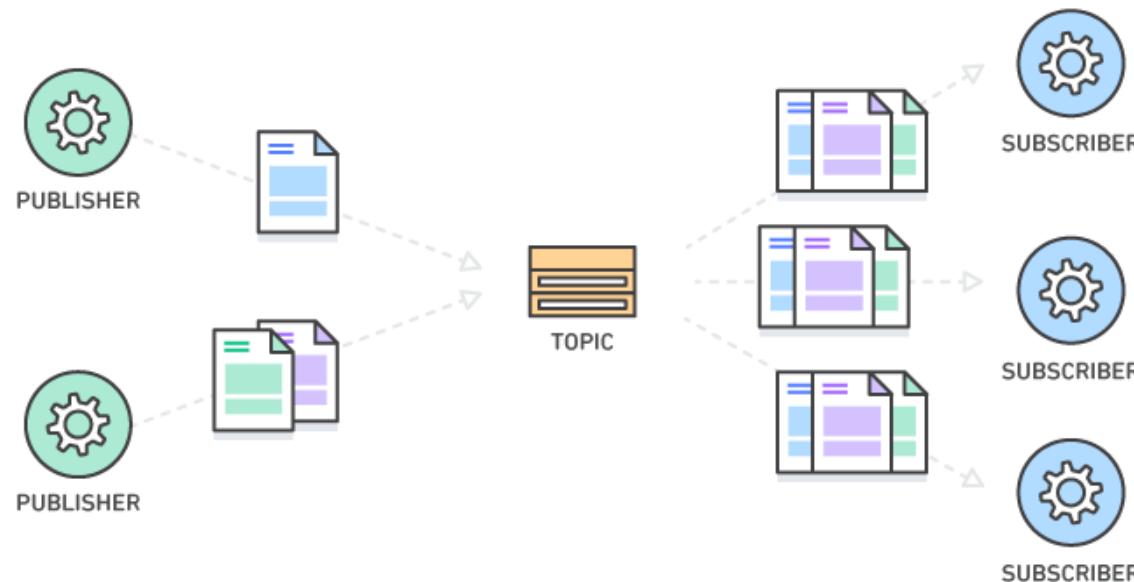
Publish-Subscribe

<https://aws.amazon.com/pub-sub-messaging/>

Publish/subscribe messaging is another asynchronous service-to-service communication mechanism

In a pub/sub model, any message published to a topic is immediately received by all of the subscribers to the topic.

Pub/sub messaging can be used to enable **event-driven architectures**, or to decouple applications in order to increase performance, reliability and scalability.



Four core concepts make up the pub/sub model:

1. **Topic** – An intermediary channel that maintains a list of subscribers to relay messages to that are received from publishers
2. **Message** – Serialized messages sent to a topic by a publisher which has no knowledge of the subscribers
3. **Publisher** – The application that publishes a message to a topic
4. **Subscriber** – An application that registers itself with the desired topic in order to receive the appropriate messages

Advantages

1. Loosing coupling

Publishers are never aware of the existence of subscribers so that both systems can operate independently of each other. This methodology removes service dependencies that are present in traditional coupling.

For example, a client generally cannot send a message to a server if the server process is not running. With pub/sub, the client is no longer concerned whether or not processes are running on the server.

2. Scalability

Pub/sub messaging can scale to volumes beyond the capability of a single traditional data centre.

This level of scalability is primarily due to parallel operations, message caching, tree-based routing, and multiple other features built into the pub/sub model.

Benefits of Pub-Sub model

1. Eliminate Polling:

Message topics allow instantaneous, push-based delivery, eliminating the need for message consumers to periodically check or “poll” for new information and updates.

This promotes faster response time and reduces the delivery latency that can be particularly problematic in systems where delays cannot be tolerated.

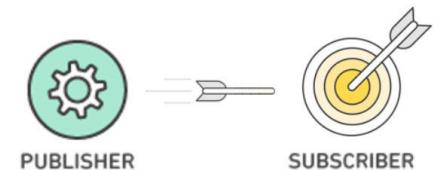


2. Dynamic Targeting

Instead of maintaining a roster of peers that an application can send messages to, a publisher will simply post messages to a topic.

Then, any interested party will subscribe its endpoint to the topic, and start receiving these messages.

Subscribers can change, upgrade, multiply or disappear and the system dynamically adjusts.



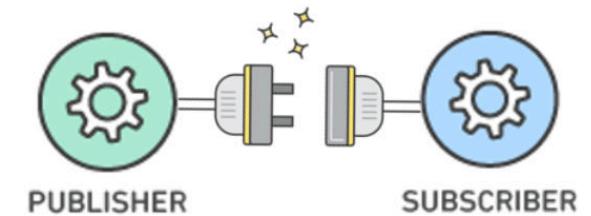
Publish-Subscribe (contd.)

Benefits of Pub-Sub model (contd.)

3. Decouple and Scale Independently:

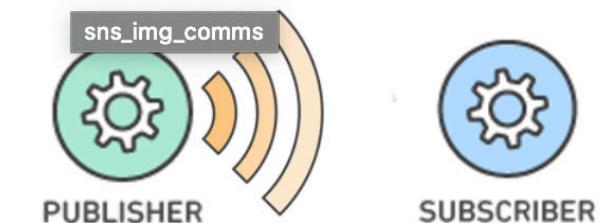
Publishers and subscribers are decoupled and work independently from each other, which allows you to develop and scale them independently.

Adding or changing functionality won't send ripple effects across the system, because Pub/Sub allows you to flex how everything uses everything else.



4. Simplify Communication

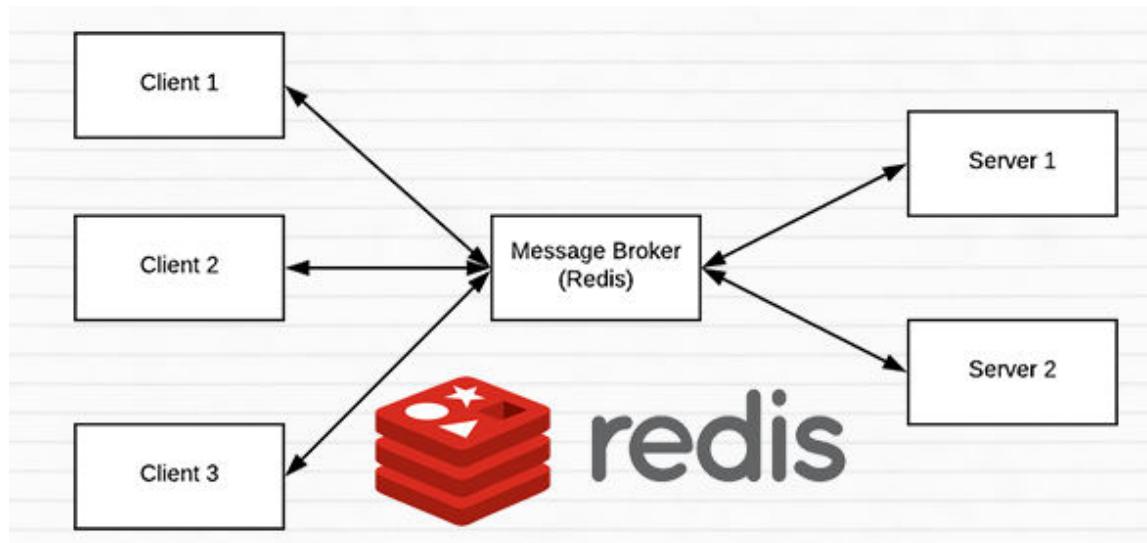
Communications and integration code is some of the hardest code to write. The Publish Subscribe model reduces complexity by removing all the point-to-point connections with a single connection to a message topic, which will manage subscriptions to decide what messages should be delivered to which endpoints. Fewer callbacks results in looser coupling and code that's easier to maintain and extend.



Redis, which stands for **Remote Dictionary Server**, is a fast, open-source, in-memory key-value data store for use as a database, cache, **message broker, and queue**.

Why use Redis?

- All Redis data resides in-memory, in contrast to databases that store data on disk or SSDs. By eliminating the need to access disks, in-memory data stores such as Redis avoid seek time delays and can access data in microseconds.





THANK YOU

K.S.Srinivas
srinivasks@pes.edu
+91 80 2672 1983 Extn 701



CLOUD COMPUTING

COMMUNICATION USING MESSAGE QUEUES

K.S.Srinivas

Department of Computer Science and Engineering

CLOUD COMPUTING

Communication using Message Queues

Srinivas K S.

Associate Professor, Department of Computer Science

“Inter-service communication is essential to consider when building micro-services.”

Monolithic Application

- Runs on a single process, with multiple components.
- Components communicate (pass data) with one another through language-level function calls.
- All components run within the same process — intra-process communication.

Micro-service Architecture

- Runs on multiple processes/services, usually across multiple servers.
- Each service instance is a process.
- Services interact using inter-process communication protocols.
- Protocols such as HTTP, AMQP, or a binary protocol like TCP.

Interaction styles can be categorised in two dimensions

First dimension

1. One-to-one : Each client request is processed by exactly one service
2. One-to-many : Each request is processed by multiple services

Second dimension

1. Synchronous : The client expects a timely response from the service and might even block while it waits.
2. Asynchronous : The client doesn't block, and the response, if any, isn't necessarily sent immediately.

Table 3.1 The various interaction styles can be characterized in two dimensions: one-to-one vs one-to-many and synchronous vs asynchronous.

	one-to-one	one-to-many
Synchronous	Request/response	—
Asynchronous	Asynchronous request/response One-way notifications	Publish/subscribe Publish/async responses

One-to-one interaction

The following are the different types of one-to-one interactions

1. *Request/response*— A service client makes a request to a service and waits for a response. The client expects the response to arrive in a timely fashion. It might even block while waiting. This is an interaction style that generally results in services being tightly coupled.

3. *Asynchronous request/response*— A service client sends a request to a service, which replies asynchronously. The client doesn't block while waiting, because the service might not send the response for a long time.

4. *One-way notifications*— A service client sends a request to a service, but no reply is expected or sent.

One-to-many interaction

The following are the different types of one-to-many interactions

- *Publish/subscribe*— A client publishes a notification message, which is consumed by zero or more interested services.
- *Publish/async responses*— A client publishes a request message and then waits for a certain amount of time for responses from interested services.

Asynchronous vs Synchronous messaging

Advantages of Asynchronous messaging

- Reduced coupling: The message sender does not need to know about the consumer.
- Multiple subscribers: Using a pub/sub model, multiple consumers can subscribe to receive events.
- Failure isolation: If the consumer fails, the sender can still send messages. The messages will be picked up when the consumer recovers. Asynchronous messaging can handle intermittent downtime. Synchronous APIs, on the other hand, require the downstream service to be available or the operation fails.
- Load leveling: A queue can act as a buffer to level the workload, so that receivers can process messages at their own rate.

Asynchronous vs Synchronous messaging

Disadvantages of Asynchronous messaging

- Coupling with the messaging infrastructure: Using a particular messaging infrastructure may cause tight coupling with that infrastructure. It will be difficult to switch to another messaging infrastructure later.
- Latency: End-to-end latency for an operation may become high if the message queues fill up.
- Complexity: Handling asynchronous messaging is not a trivial task. For example, handling of duplicated messages, or correlating request and response messages using a separate response queue.
- Throughput: If message queues are used, each message requires at least one queue operation and one dequeue operation. Moreover, queue semantics generally require some kind of locking inside the messaging infrastructure.

There are two main paradigms for asynchronous communication :

1. Message Queues
2. Event streams (Pub/Sub)

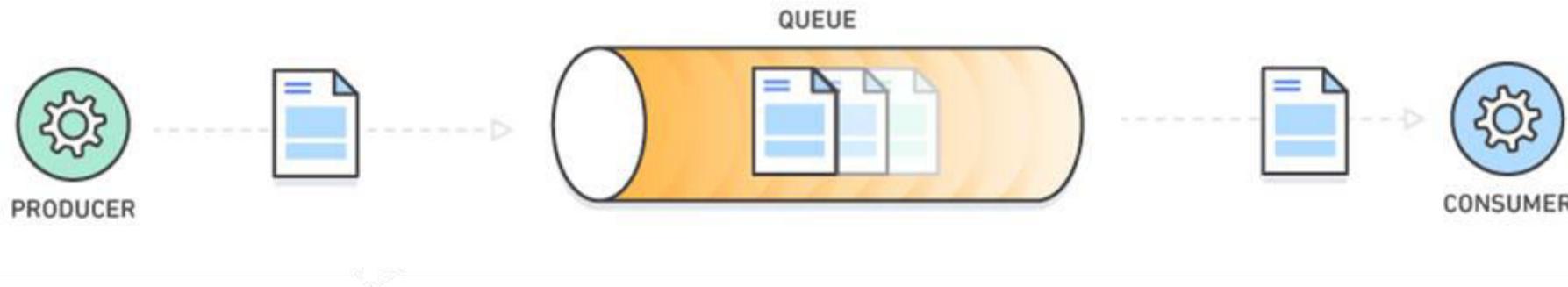
A message queue is a form of asynchronous service-to-service communication used in serverless and microservices architectures.

Messages are stored on the queue until they are processed and deleted.

Each message is processed only once, by a single consumer.

Message queues can be used to decouple heavyweight processing, to buffer or batch work, and to smooth spiky workloads.

Examples - Apache ActiveMQ, RabbitMQ, Apache Kafka



A message queue provides a lightweight buffer which temporarily stores messages, and endpoints that allow software components to connect to the queue in order to send and receive messages.

The messages are usually small, and can be things like requests, replies, error messages, or just plain information.

To send a message, a component called a producer adds a message to the queue. The message is stored on the queue until another component called a consumer retrieves the message and does something with it.

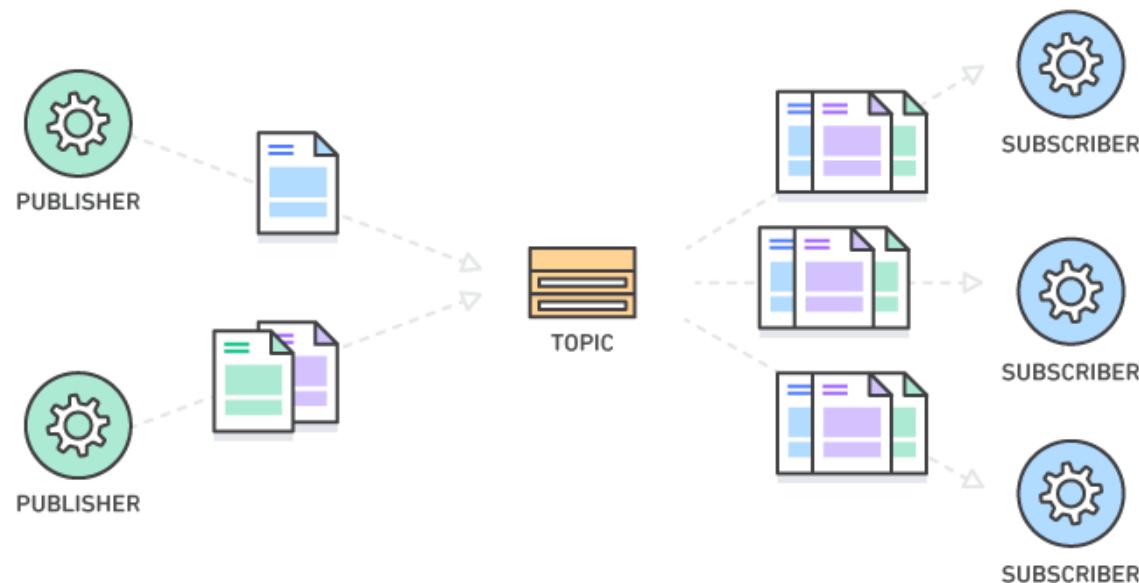
Many producers and consumers can use the queue, but each message is processed only once, by a single consumer. For this reason, this messaging pattern is often called one-to-one, or point-to-point, communications.

When a message needs to be processed by more than one consumer, message queues can be combined with Pub/Sub messaging in a fanout design pattern.

Publish/subscribe messaging is another asynchronous service-to-service communication mechanism

In a pub/sub model, any message published to a topic is immediately received by all of the subscribers to the topic.

Pub/sub messaging can be used to enable event-driven architectures, or to decouple applications in order to increase performance, reliability and scalability.



Publish-Subscribe (contd.)

Four core concepts make up the pub/sub model:

1. **Topic** – An intermediary channel that maintains a list of subscribers to relay messages to that are received from publishers
2. **Message** – Serialized messages sent to a topic by a publisher which has no knowledge of the subscribers
3. **Publisher** – The application that publishes a message to a topic
4. **Subscriber** – An application that registers itself with the desired topic in order to receive the appropriate messages

Advantages

1. Loosing coupling

Publishers are never aware of the existence of subscribers so that both systems can operate independently of each other. This methodology removes service dependencies that are present in traditional coupling.

For example, a client generally cannot send a message to a server if the server process is not running. With pub/sub, the client is no longer concerned whether or not processes are running on the server.

2. Scalability

Pub/sub messaging can scale to volumes beyond the capability of a single traditional data centre.

This level of scalability is primarily due to parallel operations, message caching, tree-based routing, and multiple other features built into the pub/sub model.

Publish-Subscribe (contd.)

Benefits of Pub-Sub model

1. Eliminate Polling:

Message topics allow instantaneous, push-based delivery, eliminating the need for message consumers to periodically check or “poll” for new information and updates.

This promotes faster response time and reduces the delivery latency that can be particularly problematic in systems where delays cannot be tolerated.

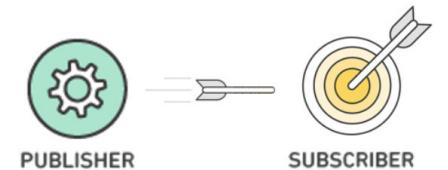


2. Dynamic Targeting

Instead of maintaining a roster of peers that an application can send messages to, a publisher will simply post messages to a topic.

Then, any interested party will subscribe its endpoint to the topic, and start receiving these messages.

Subscribers can change, upgrade, multiply or disappear and the system dynamically adjusts.



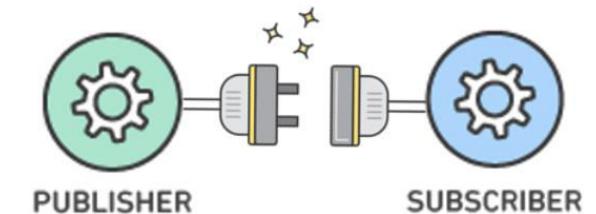
Publish-Subscribe (contd.)

Benefits of Pub-Sub model (contd.)

3. Decouple and Scale Independently:

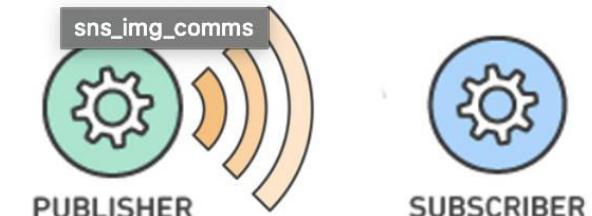
Publishers and subscribers are decoupled and work independently from each other, which allows you to develop and scale them independently.

Adding or changing functionality won't send ripple effects across the system, because Pub/Sub allows you to flex how everything uses everything else.



4. Simplify Communication

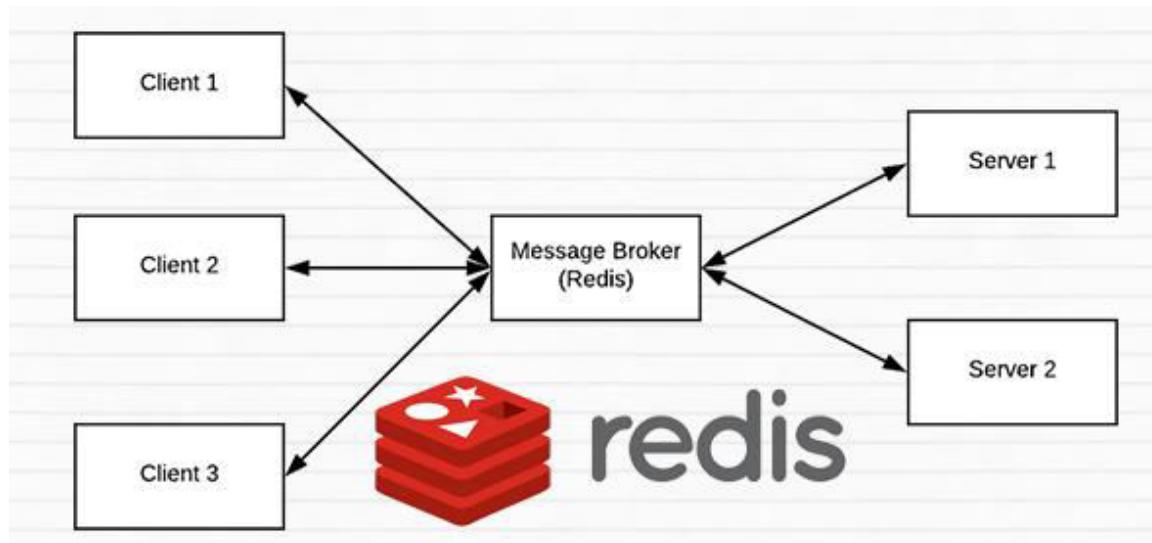
Communications and integration code is some of the hardest code to write. The Publish Subscribe model reduces complexity by removing all the point-to-point connections with a single connection to a message topic, which will manage subscriptions to decide what messages should be delivered to which endpoints. Fewer callbacks results in looser coupling and code that's easier to maintain and extend.



Redis, which stands for **Remote Dictionary Server**, is a fast, open-source, in-memory key-value data store for use as a database, cache, **message broker, and queue**.

Why use Redis?

- All Redis data resides in-memory, in contrast to databases that store data on disk or SSDs. By eliminating the need to access disks, in-memory data stores such as Redis avoid seek time delays and can access data in microseconds.





THANK YOU

K.S.Srinivas
srinivasks@pes.edu
+91 80 2672 1983 Extn 701



CLOUD COMPUTING

SaaS Programming Model

K.S.Srinivas

Department of Computer Science and Engineering

CLOUD COMPUTING

SaaS Programming Model

Srinivas K S.

Associate Professor, Department of Computer Science

CLOUD COMPUTING

Software as a Service



What is SaaS?

- Software as a service (or SaaS) is a way of delivering applications over the Internet—as a service. Instead of installing and maintaining software, you simply access it via the Internet, freeing yourself from complex software and hardware management.
- SaaS applications are sometimes called Web-based software, on-demand software, or hosted software. Whatever the name, SaaS applications run on a SaaS provider's servers. The provider manages access to the application, including security, availability, and performance.

The Payoff

SaaS customers have no hardware or software to buy, install, maintain, or update. Access to applications is easy: You just need an Internet connection.

Nearly every software that runs in your browser and is targeted towards the end user can be categorised as a Software as a Service product.

Popular SaaS Applications

1. Office apps - Google Docs, Sheets, Office 365
2. Email client - Gmail, Outlook
3. File Storage - DropBox, OneDrive
4. Social Media - Facebook, Snapchat
5. Customer Relationship Management - Salesforce
6. Customer support - Zendesk



Characteristics

1. Multi-tenant Architecture

A multi-tenant architecture, in which all users and applications share a single, common infrastructure and code base that is centrally maintained.

Because SaaS vendor clients are all on the same infrastructure and code base, vendors can innovate more quickly and save the valuable development time previously spent on maintaining numerous versions of outdated code.

2. Easy Customisation

The ability for each user to easily customise applications to fit their business processes without affecting the common infrastructure.

Because of the way SaaS is architected, these customisations are unique to each company or user and are always preserved through upgrades.

That means SaaS providers can make upgrades more often, with less customer risk and much lower adoption cost.

3. Better Access

Improved access to data from any networked device while making it easier to manage privileges, monitor data use, and ensure everyone sees the same information at the same time.

4. SaaS Harnesses the Consumer Web

Anyone familiar with Amazon.com or Gmail will be familiar with the Web interface of typical SaaS applications.

With the SaaS model, you can customise with point-and-click ease, making the weeks or months it takes to update traditional business software seem hopelessly old fashioned.

5. SaaS Trends

Organisations are now developing SaaS integration platforms (or SIPs) for building additional SaaS applications.

The consulting firm Saugatuck Technology calls this the “third wave” in software adoption: when SaaS moves beyond standalone software functionality to become a platform for mission-critical applications.

Why use SaaS?

1. **Flexible payments:** Rather than purchasing software to install, or additional hardware to support it, customers subscribe to a SaaS offering. Generally, they pay for this service on a monthly basis using a pay-as-you-go model.
2. **Scalable usage:** Cloud services like SaaS offer high vertical scalability, which gives customers the option to access more, or fewer, services or features on-demand.
3. **Automatic updates:** Rather than purchasing new software, customers can rely on a SaaS provider to automatically perform updates. This further reduces the burden on in-house IT staff.
4. **Accessibility and persistence:** Since SaaS applications are delivered over the Internet, users can access them from any Internet-enabled device and location.

Disadvantages of SaaS

1. Security

Actually, data is stored in the cloud, so security may be an issue for some users. However, cloud computing is not more secure than in-house deployment.

2. Latency issue

Since data and applications are stored in the cloud at a variable distance from the end-user, there is a possibility that there may be greater latency when interacting with the application compared to local deployment. Therefore, the SaaS model is not suitable for applications whose demand response time is in milliseconds.

3. Total Dependency on Internet

Without an internet connection, most SaaS applications are not usable.

4. Switching between SaaS vendors is difficult

Switching SaaS vendors involves the difficult and slow task of transferring the very large data files over the internet and then converting and importing them into another SaaS also.

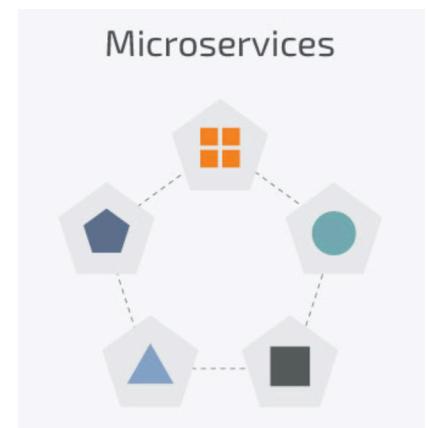
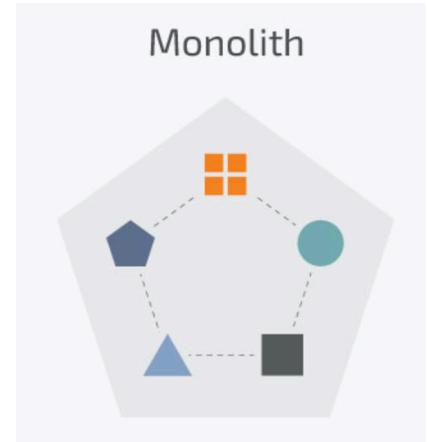


Monolithic Architecture

It is considered to be a traditional way of building applications. A monolithic application is built as a single and indivisible unit. Usually, such a solution comprises a client-side user interface, a server side-application, and a database.

Microservices Architecture

While a monolithic application is a single unified unit, a microservices architecture breaks it down into a collection of smaller independent units. These units carry out every application process as a separate service. So all the services have their own logic and the database as well as perform the specific functions.



Strengths of Monolithic Architecture

Strengths of Monolithic Architecture

1. Development is quite simple.
2. Testing is very simple - Just launch the application and start end-to-end testing. We can also do test automation using Selenium without any difficulty.
3. Deploying the monolithic application is straightforward - Just copy the packaged application to the server.
4. Scalability is simple - We only need to have a new instance of the monolithic application and ask the load balancer to distribute load to the new instance as well. However, as the monolithic application grows in size, scalability becomes a serious issue.

Monolithic architecture worked successfully for many decades. In fact, many of the most successful and largest applications were initially developed and deployed as a monolith.

But serious issues like flexibility, reliability and scalability have led to the emergence of microservices architecture.

“ The microservice architectural style is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API.

These services are built around business capabilities and independently deployable by fully automated deployment machinery.

There is a bare minimum of centralized management of these services, which may be written in different programming languages and use different data storage technologies. ”

- Martin Fowler

Benefits of Microservices Architecture

Benefits

1. Flexibility:

Microservices architecture is quite flexible. Different microservices can be developed in different technologies. Since a microservice is smaller, the code base is quite less, so it's not that difficult to upgrade the technology stack versions. Also, we can incrementally adopt a newer technology without much difficulty.

2. Reliability:

Microservices architecture can be very reliable. If one feature goes down, the entire application doesn't go down. We can fix the issue in the corresponding microservice and immediately deploy it.

3. Development speed:

Development is pretty fast in microservices architecture. Since the volume of code is much less for a microservice, it's not difficult for new team members to understand and modify the code. They become productive right from the start. Code quality is maintained well. The IDE is much faster. A microservice takes much less time to start up. All these factors considerably increase developers' productivity.

Benefits of Microservices Architecture (contd.)

3. Building complex applications:

With microservice architecture, it's easy to build complex applications. If the features of the application are analyzed properly, we can break it down into independent components which can be deployed independently. Then, even the independent components can be further broken down into small independent tasks which can be deployed independently as a microservice. Deciding the boundaries of a microservice can be quite challenging. It's actually an evolutionary process, but once we decide on a microservice, it's easy to develop, as there are no limitation in technologies.

4. Scalability:

Scalability is a major advantage in microservice architecture. Each microservice can be scaled individually. Since individual microservices are much smaller in size, caching becomes very effective.

5. Continuous deployment:

Continuous deployment becomes easier. In order to update one component, we have to redeploy only that particular microservice.

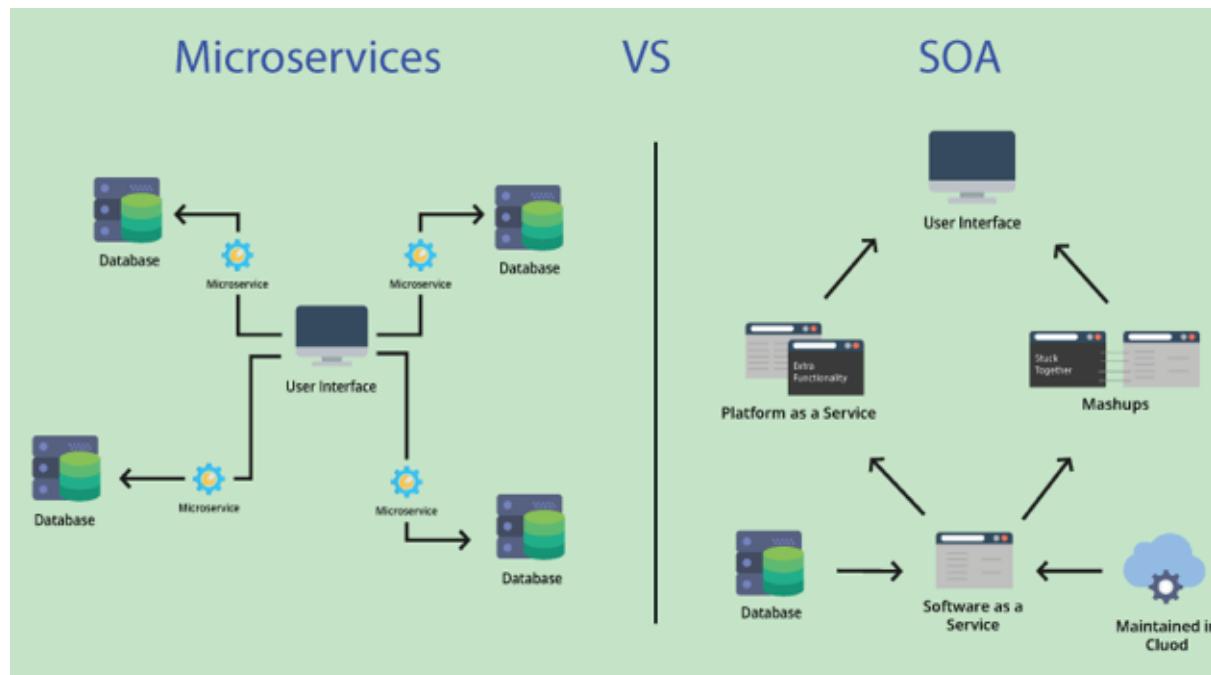
CLOUD COMPUTING

Comparison with SOA

What is SOA?

Service-oriented architecture was largely created as a response to traditional, monolithic approaches to building applications. SOA breaks up the components required for applications into separate service modules that communicate with one another to meet specific business objectives.

Microservice architecture is generally considered an evolution of SOA as its services are more fine-grained, and function independently of each other



CLOUD COMPUTING

Comparison with SOA

SOA	Microservice Architecture
Follows “ share-as-much-as-possible ” architecture approach	Follows “ share-as-little-as-possible ” architecture approach
Importance is on business functionality reuse	Importance is on the concept of “ bounded context ” or Single Responsibility
They have common governance and standards	They focus on people, collaboration and freedom of other options
Uses Enterprise Service bus (ESB) for communication	Simple messaging system
They support multiple message protocols	They use lightweight protocols such as HTTP/REST etc.
Multi-threaded with more overheads to handle I/O	Single-threaded usually with the use of Event Loop features for non-locking I/O handling
Maximizes application service reusability	Focuses on decoupling
Traditional Relational Databases are more often used	Modern Relational Databases are more often used
A systematic change requires modifying the monolith	A systematic change is to create a new service
DevOps / Continuous Delivery is becoming popular, but not yet mainstream	Strong focus on DevOps / Continuous Delivery



THANK YOU

K.S.Srinivas
srinivasks@pes.edu
+91 80 2672 1983 Extn 701



CLOUD COMPUTING

SaaS Programming Model

K.S.Srinivas

Department of Computer Science and Engineering

CLOUD COMPUTING

SaaS Programming Model

Srinivas K S.

Associate Professor, Department of Computer Science

CLOUD COMPUTING

Software as a Service

What is SaaS?

- Software as a service (or SaaS) is a way of delivering applications over the Internet—as a service. Instead of installing and maintaining software, you simply access it via the Internet, freeing yourself from complex software and hardware management.
- SaaS applications are sometimes called Web-based software, on-demand software, or hosted software. Whatever the name, SaaS applications run on a SaaS provider's servers. The provider manages access to the application, including security, availability, and performance.

The Payoff

SaaS customers have no hardware or software to buy, install, maintain, or update. Access to applications is easy: You just need an Internet connection.

CLOUD COMPUTING

SaaS Examples

Nearly every software that runs in your browser and is targeted towards the end user can be categorised as a Software as a Service product.

Popular SaaS Applications

1. Office apps - Google Docs, Sheets, Office 365
2. Email client - Gmail, Outlook
3. File Storage - DropBox, OneDrive
4. Social Media - Facebook, Snapchat
5. Customer Relationship Management - Salesforce
6. Customer support - Zendesk



Characteristics

1. Multi-tenant Architecture

A multi-tenant architecture, in which all users and applications share a single, common infrastructure and code base that is centrally maintained.

Because SaaS vendor clients are all on the same infrastructure and code base, vendors can innovate more quickly and save the valuable development time previously spent on maintaining numerous versions of outdated code.

2. Easy Customisation

The ability for each user to easily customise applications to fit their business processes without affecting the common infrastructure.

Because of the way SaaS is architected, these customisations are unique to each company or user and are always preserved through upgrades.

That means SaaS providers can make upgrades more often, with less customer risk and much lower adoption cost.

3. Better Access

Improved access to data from any networked device while making it easier to manage privileges, monitor data use, and ensure everyone sees the same information at the same time.

4. SaaS Harnesses the Consumer Web

Anyone familiar with Amazon.com or Gmail will be familiar with the Web interface of typical SaaS applications.

With the SaaS model, you can customise with point-and-click ease, making the weeks or months it takes to update traditional business software seem hopelessly old fashioned.

5. SaaS Trends

Organisations are now developing SaaS integration platforms (or SIPs) for building additional SaaS applications.

The consulting firm Saugatuck Technology calls this the “third wave” in software adoption: when SaaS moves beyond standalone software functionality to become a platform for mission-critical applications.

Why use Saas?

1. **Flexible payments:** Rather than purchasing software to install, or additional hardware to support it, customers subscribe to a SaaS offering. Generally, they pay for this service on a monthly basis using a [pay-as-you-go model](#).
2. **Scalable usage:** Cloud services like SaaS offer high vertical scalability, which gives customers the option to access more, or fewer, services or features on-demand.
3. **Automatic updates:** Rather than purchasing new software, customers can rely on a SaaS provider to automatically perform updates. This further reduces the burden on in-house IT staff.
4. **Accessibility and persistence:** Since SaaS applications are delivered over the Internet, users can access them from any Internet-enabled device and location.

Disadvantages of SaaS

1. Security

Actually, data is stored in the cloud, so security may be an issue for some users. However, cloud computing is not more secure than in-house deployment.

2. Latency issue

Since data and applications are stored in the cloud at a variable distance from the end-user, there is a possibility that there may be greater latency when interacting with the application compared to local deployment. Therefore, the SaaS model is not suitable for applications whose demand response time is in milliseconds.

3. Total Dependency on Internet

Without an internet connection, most SaaS applications are not usable.

4. Switching between SaaS vendors is difficult

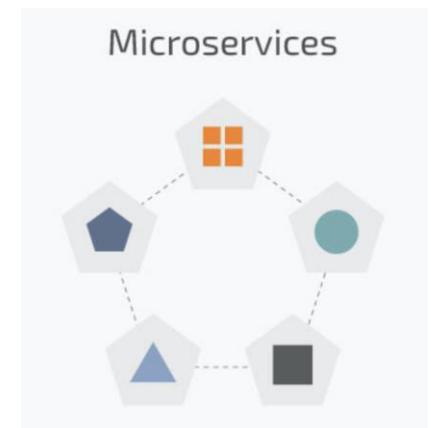
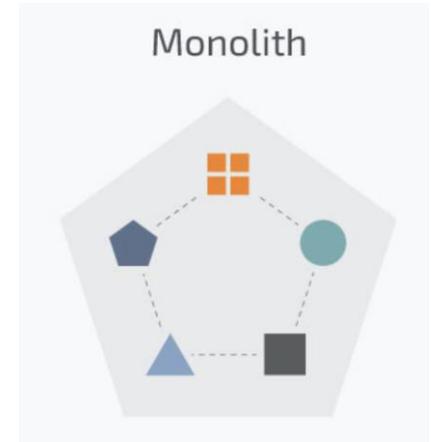
Switching SaaS vendors involves the difficult and slow task of transferring the very large data files over the internet and then converting and importing them into another SaaS also.

Monolithic Architecture

It is considered to be a traditional way of building applications. A monolithic application is built as a single and indivisible unit. Usually, such a solution comprises a client-side user interface, a server side-application, and a database.

Microservices Architecture

While a monolithic application is a single unified unit, a microservices architecture breaks it down into a collection of smaller independent units. These units carry out every application process as a separate service. So all the services have their own logic and the database as well as perform the specific functions.



Strengths of Monolithic Architecture

Strengths of Monolithic Architecture

1. Development is quite simple.
2. Testing is very simple - Just launch the application and start end-to-end testing. We can also do test automation using Selenium without any difficulty.
3. Deploying the monolithic application is straightforward - Just copy the packaged application to the server.
4. Scalability is simple - We only need to have a new instance of the monolithic application and ask the load balancer to distribute load to the new instance as well. However, as the monolithic application grows in size, scalability becomes a serious issue.

Monolithic architecture worked successfully for many decades. In fact, many of the most successful and largest applications were initially developed and deployed as a monolith.

But serious issues like flexibility, reliability and scalability have led to the emergence of microservices architecture.

CLOUD COMPUTING

Microservices Architecture



“ The microservice architectural style is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API.

These services are built around business capabilities and independently deployable by fully automated deployment machinery.

There is a bare minimum of centralized management of these services, which may be written in different programming languages and use different data storage technologies. ”

- Martin Fowler

Benefits of Microservices Architecture

Benefits

1. Flexibility:

Microservices architecture is quite flexible. Different microservices can be developed in different technologies. Since a microservice is smaller, the code base is quite less, so it's not that difficult to upgrade the technology stack versions. Also, we can incrementally adopt a newer technology without much difficulty.

2. Reliability:

Microservices architecture can be very reliable. If one feature goes down, the entire application doesn't go down. We can fix the issue in the corresponding microservice and immediately deploy it.

3. Development speed:

Development is pretty fast in microservices architecture. Since the volume of code is much less for a microservice, it's not difficult for new team members to understand and modify the code. They become productive right from the start. Code quality is maintained well. The IDE is much faster. A microservice takes much less time to start up. All these factors considerably increase developers' productivity.

Benefits of Microservices Architecture (contd.)

3. Building complex applications:

With microservice architecture, it's easy to build complex applications. If the features of the application are analyzed properly, we can break it down into independent components which can be deployed independently. Then, even the independent components can be further broken down into small independent tasks which can be deployed independently as a microservice. Deciding the boundaries of a microservice can be quite challenging. It's actually an evolutionary process, but once we decide on a microservice, it's easy to develop, as there are no limitation in technologies.

4. Scalability:

Scalability is a major advantage in microservice architecture. Each microservice can be scaled individually. Since individual microservices are much smaller in size, caching becomes very effective.

5. Continuous deployment:

Continuous deployment becomes easier. In order to update one component, we have to redeploy only that particular microservice.

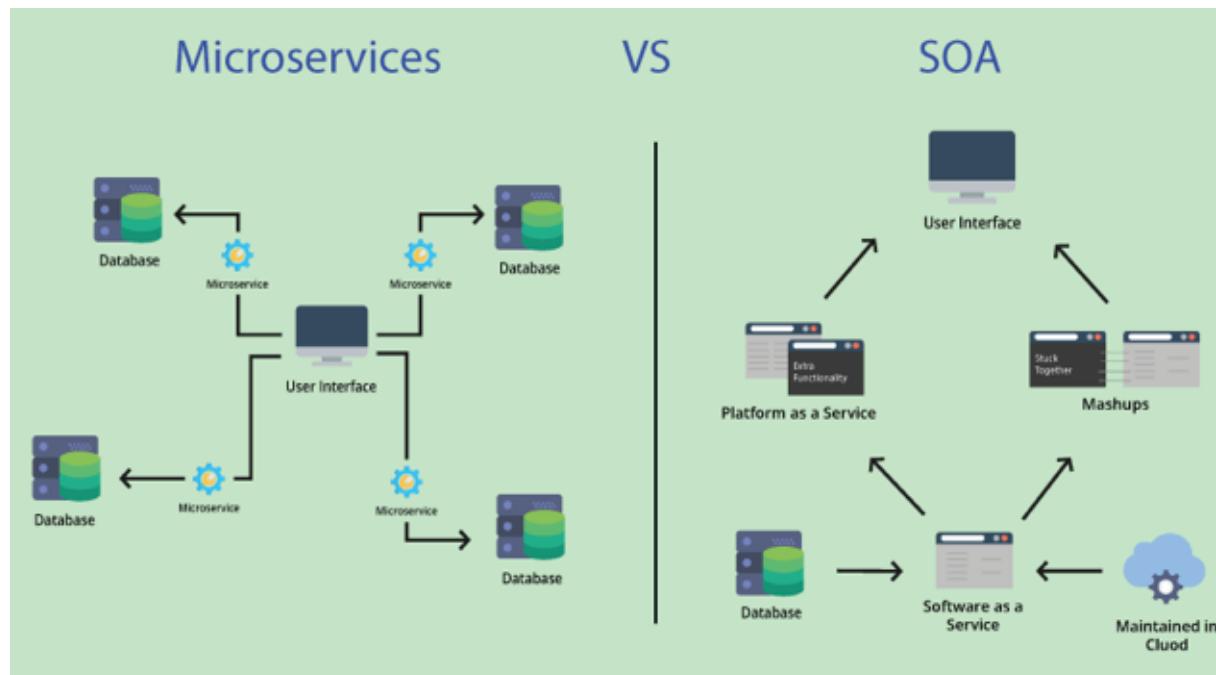
CLOUD COMPUTING

Comparison with SOA

What is SOA?

Service-oriented architecture was largely created as a response to traditional, monolithic approaches to building applications. SOA breaks up the components required for applications into separate service modules that communicate with one another to meet specific business objectives.

Microservice architecture is generally considered an evolution of SOA as its services are more fine-grained, and function independently of each other



CLOUD COMPUTING

Comparison with SOA

SOA	Microservice Architecture
Follows “ share-as-much-as-possible ” architecture approach	Follows “ share-as-little-as-possible ” architecture approach
Importance is on business functionality reuse	Importance is on the concept of “ bounded context ” or Single Responsibility
They have common governance and standards	They focus on people, collaboration and freedom of other options
Uses Enterprise Service bus (ESB) for communication	Simple messaging system
They support multiple message protocols	They use lightweight protocols such as HTTP/REST etc.
Multi-threaded with more overheads to handle I/O	Single-threaded usually with the use of Event Loop features for non-locking I/O handling
Maximizes application service reusability	Focuses on decoupling
Traditional Relational Databases are more often used	Modern Relational Databases are more often used
A systematic change requires modifying the monolith	A systematic change is to create a new service
DevOps / Continuous Delivery is becoming popular, but not yet mainstream	Strong focus on DevOps / Continuous Delivery



THANK YOU

K.S.Srinivas
srinivasks@pes.edu
+91 80 2672 1983 Extn 701



CLOUD COMPUTING

Challenges of migrating monolithic applications to microservices

K.S.Srinivas

Department of Computer Science and Engineering

CLOUD COMPUTING

Challenges of migrating monolithic applications to microservices

Srinivas K S.

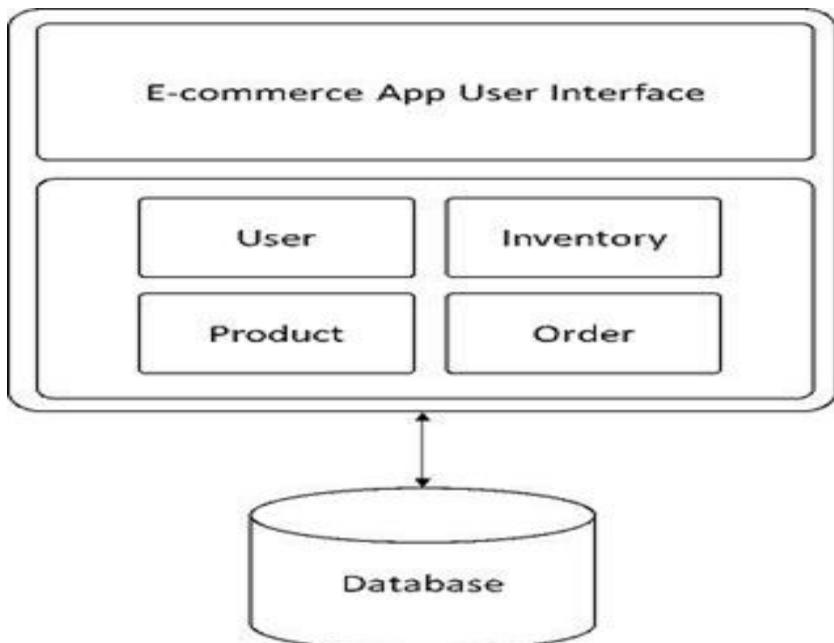
Associate Professor, Department of Computer Science

CLOUD COMPUTING

Migration is not easy

Microservices architecture promises to solve the shortcomings of monolithic applications, so many enterprises are interested in migrating their applications to be microservices. This migration is a worthwhile journey, but not an easy one.

Most enterprises have a large application in place for their business use cases. Take the example of an e-commerce application used by an online retailer. The diagram below shows its monolithic architecture.

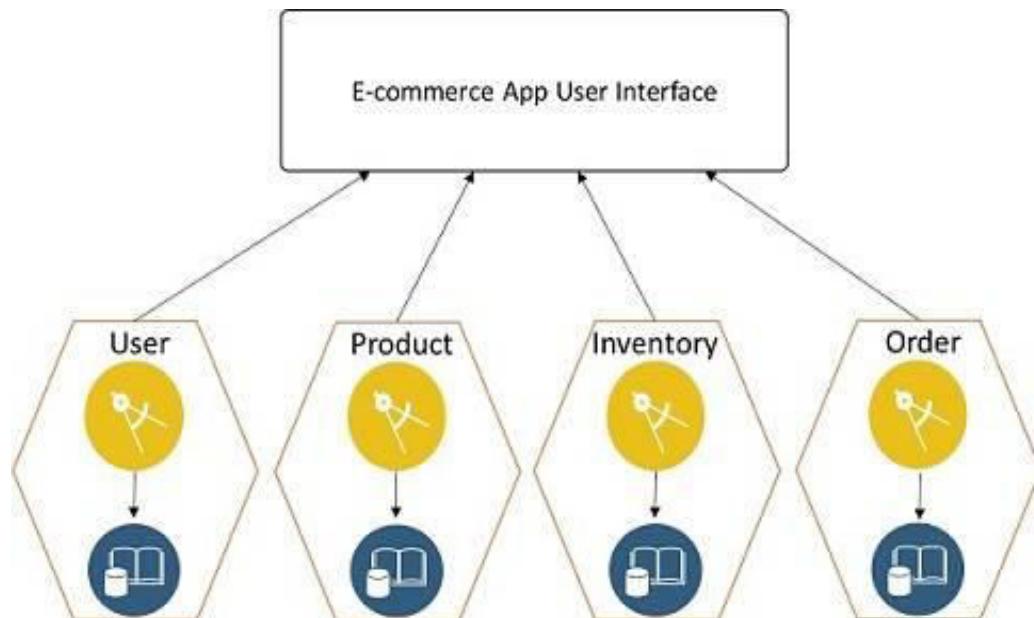


CLOUD COMPUTING

Migration is not easy

Monolithic architecture seems like a good idea to quickly get the application started, it becomes a problem over time. As the business and user base grows, customers expect newer user experiences, and integration requirements increase, the monolithic approach becomes a bottleneck to growth.

In contrast, Figure below shows how the same e-commerce application can be broken down into a microservices architecture.



CLOUD COMPUTING

Migration is not easy

microservices

[https://www.datacenters.com/news/challenges-in-transitioning-to-](https://www.datacenters.com/news/challenges-in-transitioning-to-microservices)

The idea seems simple but is sometimes difficult to perceive. The architecture team should consider the following challenges:

1. Service decomposition : Possible candidates that could represent a microservice.
2. Persistence : Monolithic database decomposition
3. Tackling transaction boundaries
4. Performance
5. Testing
6. Inter-service communication



In transitioning an existing monolith to a microservices, you would typically need to decompose the existing application into granular microservices. This is a challenging task as we need to identify possible independent components in a monolithic application.

Although breaking down a monolithic application can seem like an uphill task to start, with certain guidelines even this mammoth task is possible:

1. Stop adding more things to the monolithic app. Fix what is broken and accept only small changes.
1. Find out the so-called seams in your monolithic app. Identify components that are more loosely coupled than others and use them as a starting point.
1. Identify low hanging fruits, such as the components for which business units want to add more advanced features.

CLOUD COMPUTING

Service Decomposition

During this transitioning process, you would typically need to decompose the monolith to building more and more granular microservices to suit the business needs.

Once this is accomplished, there would be more moving parts in the application. As a result, this would lead to operational and infrastructural overheads, i.e., configuration management, security, provisioning, integration, deployment, monitoring, etc.

One of the ways to reduce these complexities is in using containerization. In using containerization, provisioning, configuration, and deployment of microservices would be simplified.

Persistence : Monolithic database decomposition

A monolithic application typically contains a single data store. In contrast, a microservices-based application contains multiple databases, typically one for each service. One of the key challenges in transitioning to microservices-based architecture is in understanding and handling decentralized data management.

Splitting the data model of a monolithic application to fit the autonomous data models of a microservices-based application is challenging. Breaking a monolith data model into separate autonomous data models that are local to each microservice is a daunting task, the following challenges need to be considered:

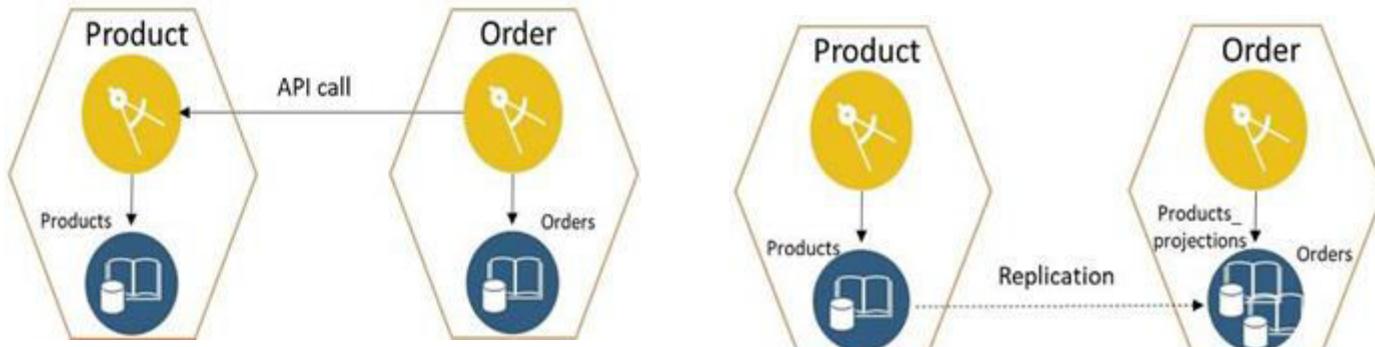
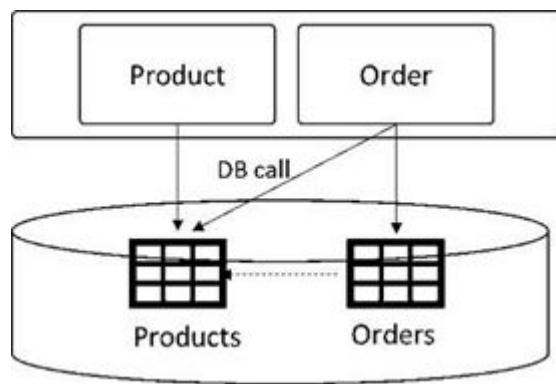
- A. Reference tables
- B. Shared mutable data
- C. Shared table

Persistence : Monolithic database decomposition

A. Reference tables :

The most common type of pattern that is seen within monolithic applications is a reference table. In this pattern, the function or module accesses a table that belongs to another function or module. The joins between a module's own table and another table are used to retrieve the required. Using the previous example of an e-commerce application, the Order module uses a reference to the Products table to retrieve product information. But in a microservices architecture, the Products table and Order module become separate microservices. Here are two options for you to consider to segregate the database objects :

1. Data as an API
2. Projection/Replication of data

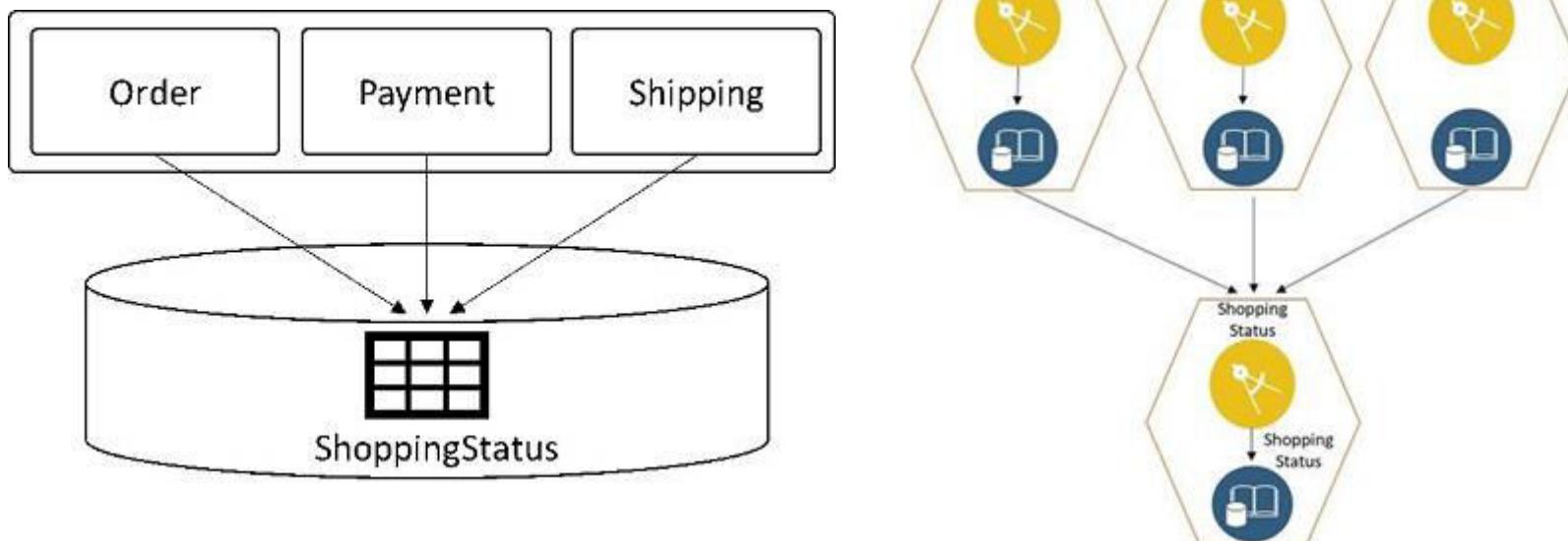


Persistence : Monolithic database decomposition

B. Shared mutable data :

In monolithic applications, there is a common pattern that is known as shared mutable state. This pattern represents certain domain realities that are modeled in a database and accessed by different functionalities or modules of the application. This is mainly done for convenience. For example, in the e-commerce application, the Order, Payment, and Shipping functionalities use the same ShoppingStatus table to maintain the customer's order status throughout the shopping journey.

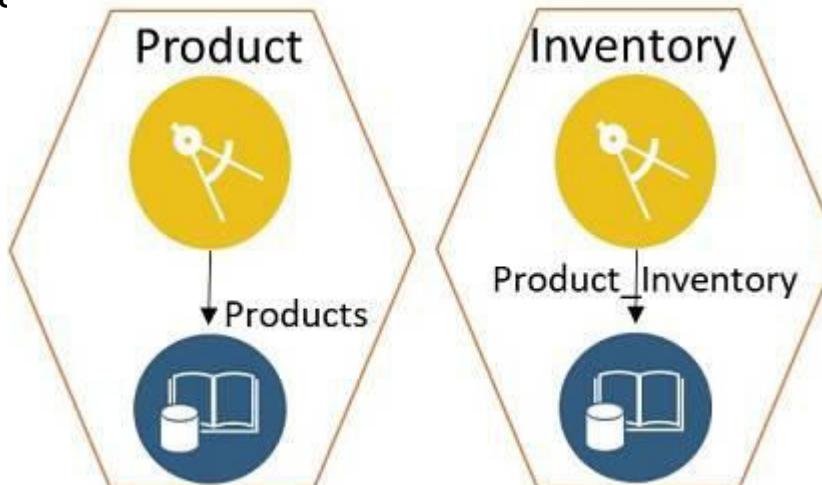
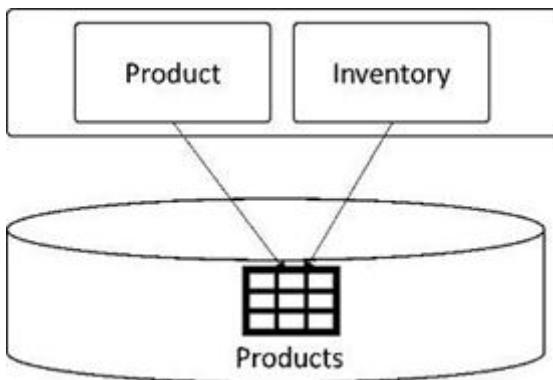
While moving to microservice architecture they should eventually be modeled as a separate microservice.



C. Shared table:

The shared table pattern is very similar to the shared mutable state pattern and is the result of erroneous domain modeling. In this scenario, a database table is modeled with attributes that are needed by two or more functionalities or modules. Again, this is done mostly for convenience reasons. To explain this, Figure below shows a scenario where the Products table is modeled to cater to the Product and Inventory functionalities.

While moving to a microservices architecture, Product and Inventory are modeled as separate services. The Products table is split into individual entities that belong to the specific bounded context of the separate Product and Inventory microservices, as shown in Figure.

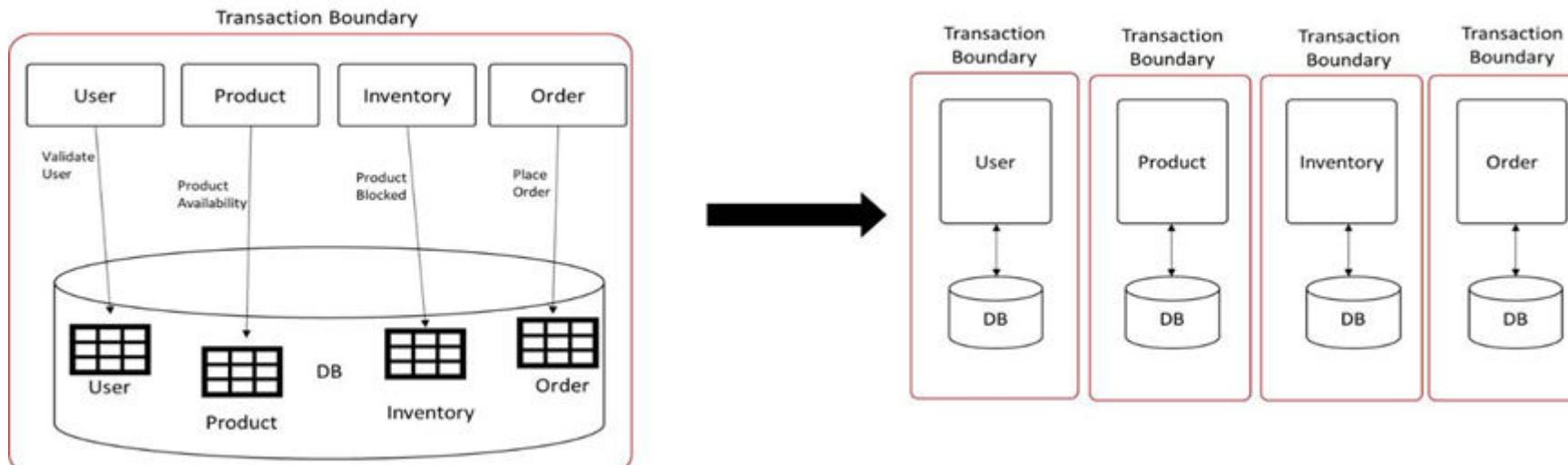


Tackling transaction boundaries

With a monolithic application that has a single database, you could guarantee the ACID (Atomicity, Consistency, Isolation, Durability) properties and ability to perform transactions by keeping locks on rows.

Things changed drastically when enterprises moved from service-oriented architecture to microservices architecture. They now must deal with database per service eventually leading to distributed transaction trouble. Following are the ideal ways to deal with distributed transactions :

- A. Two-phase commit (2PC)
- B. Compensating transactions (Saga transactions)

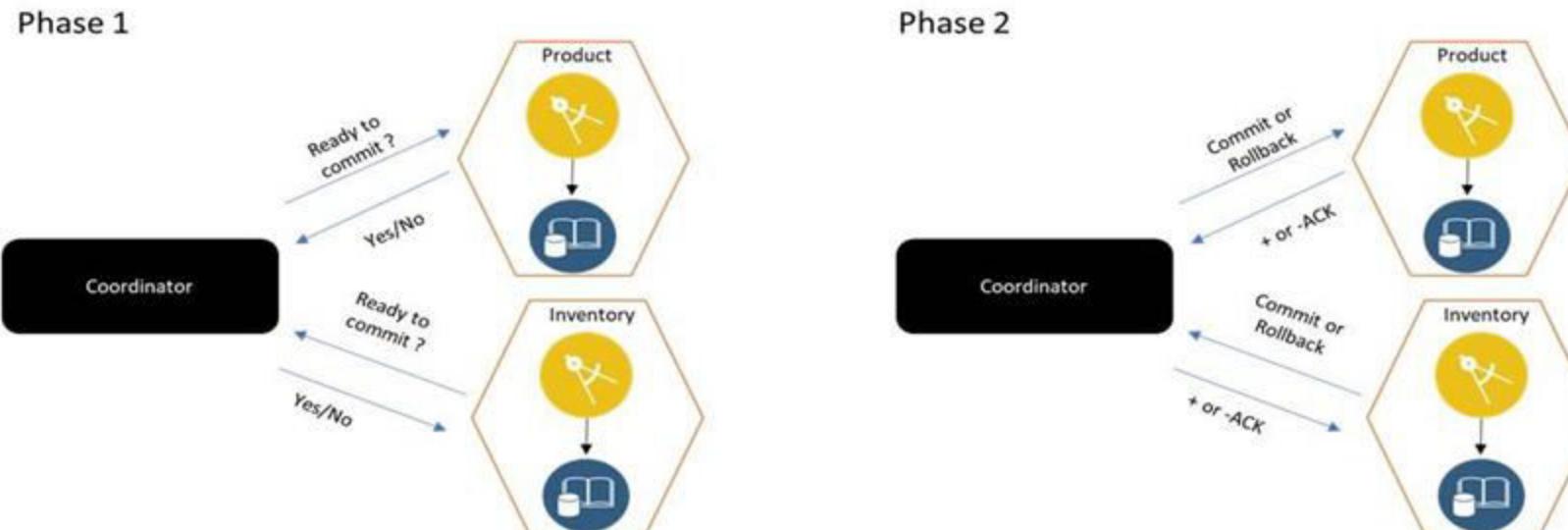


A. Two-phase commit (2PC):

In a two-phase commit, you have a controlling node that houses most of the logic, and a few participating nodes on which the actions are performed. It works in two phases:

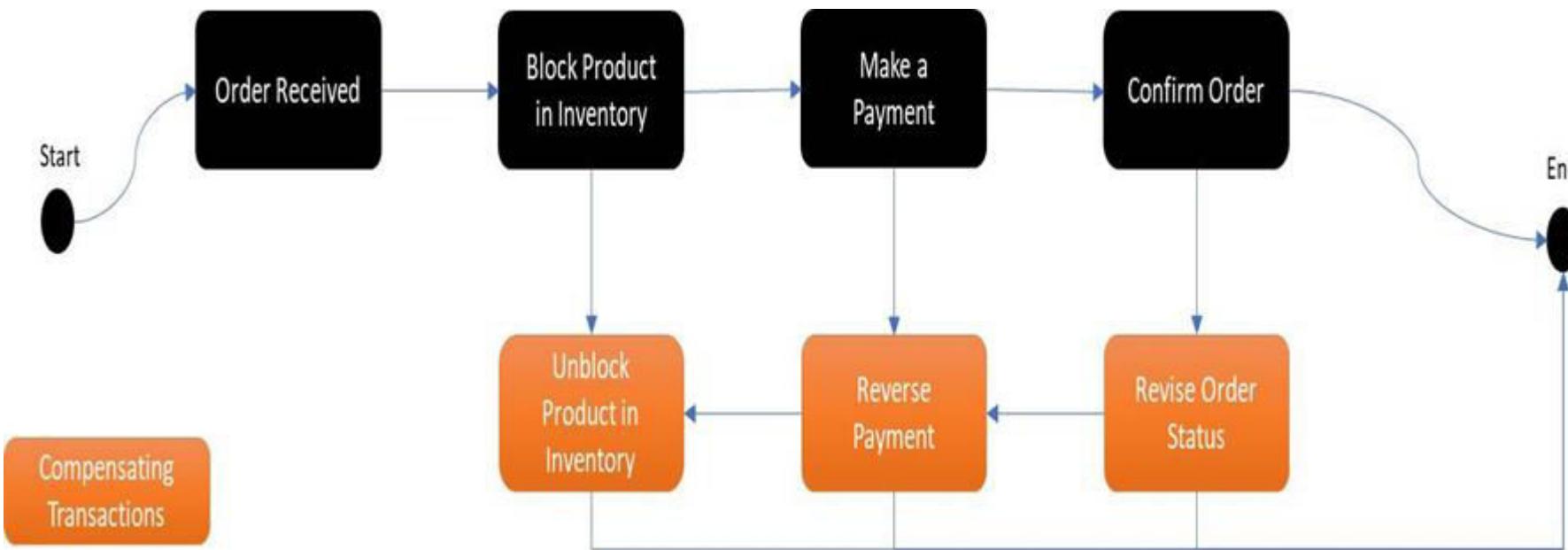
Prepare phase (Phase 1): The controlling node asks all of the participating nodes if they are ready to commit. The participating nodes respond with yes or no.

Commit phase (Phase 2): If all of the nodes replied in the affirmative, then the controlling node asks them to commit. Even if one node replies in the negative, the controlling node asks them to roll back.



B. Compensating transactions (Saga transactions):

A saga is a sequence of local transactions. Each service in a saga performs its own transaction and publishes an event. The other services listen to that event and perform the next local transaction. If one transaction fails for some reason, then the saga also executes compensating transactions to undo the impact of the preceding transactions.



The increase in resource usage may cause a microservices-based application to execute slower.

You can overcome this challenge by :

1. Introducing additional servers.
1. You can log performance data and detect the problems. You should take advantage of logging to store performance data in a repository so that you can analyze the data at a later point of time.
1. You can implement throttling, handle service timeouts, implement dedicated thread pools, implement circuit breakers and take advantage of asynchronous programming to boost the performance of microservice-based applications.

Testing microservices can become challenging, particularly the integration tests. To write an effective integration test case, the QA engineer should have good knowledge of each of the services that are a part of the solution.

Another reason why testing a microservices-based application is difficult is because microservices-based applications are generally asynchronous. The best approach to solving these testing challenges is adopting various testing methodologies and tools and leveraging continuous integration capabilities through automation and standard agile methodologies.

Inter-service communication

In a distributed system, the components should be able to communicate with each other and in microservices-architecture, it is no exception. In a monolithic application, the components invoke one another through method or function calls. In contrast, a microservices-based application is distributed in nature with each service running isolated from another service. Hence, it is imperative that services in a microservice-based application communicate using inter-process communication (also known as IPC) mechanisms. However, inter-service communication in a microservices-based application poses a lot of challenges.

Since microservices are distributed in nature, remote invocation of these services is a challenge and you should understand the necessary patterns to overcome the challenges involved. Since services in a microservice-based application communicate with one another using IPC mechanisms you should consider issues like, how the services will interact, how to handle failures, etc.



THANK YOU

K.S.Srinivas
srinivasks@pes.edu
+91 80 2672 1983 Extn 701



CLOUD COMPUTING

REST and Web Services

K.S.Srinivas

Department of Computer Science and Engineering

CLOUD COMPUTING

REST and Web Services

Srinivas K S.

Associate Professor, Department of Computer Science

Service oriented architecture (SOA)

SOA, or **service-oriented architecture**, defines a way to make software components reusable via service interfaces. These interfaces utilize common communication standards in such a way that they can be rapidly incorporated into new applications without having to perform deep integration each time.

Each service in an SOA embodies the code and data integrations required to execute a complete, discrete business function. The service interfaces provide loose coupling, meaning they can be called with little or no knowledge of how the integration is implemented underneath. The services are exposed using standard network protocols—such as SOAP (simple object access protocol)/HTTP or JSON (Java Script Object Notation)/HTTP—to send requests to read or change data. The services are published in a way that enables developers to quickly find them and reuse them to assemble new applications.

two major service-oriented architecture styles :

1. **REST** (REpresentational State Transfer)

2. **WS** (Web Services)

<https://www.ibm.com/cloud/learn/soa>,

<https://www.geeksforgeeks.org/service-oriented-architecture/>

CLOUD COMPUTING

SOA (Cont.)

Service-oriented architecture (SOA) and development is a paradigm where software components are created with concise interfaces, and each component performs a discrete set of related functions.

With its well-defined interface and contract for usage, each component, provides a service to other software components.

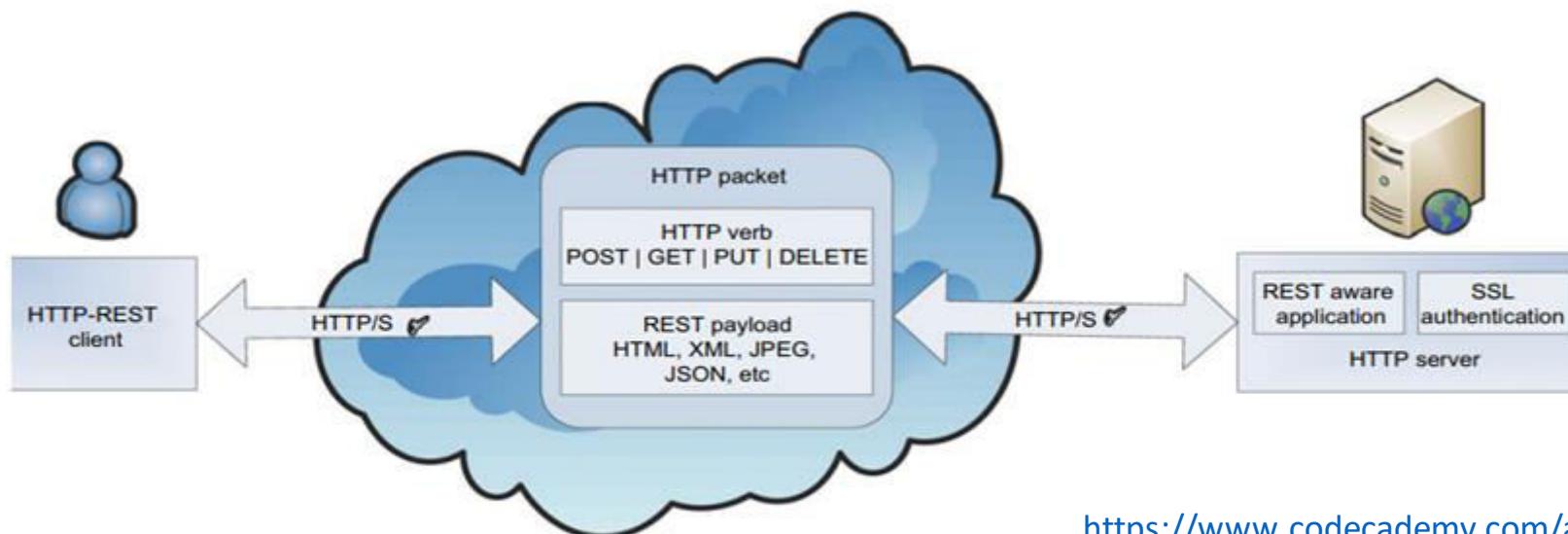
This is analogous to an accountant who provides a service to a business, even though that service consists of many related functions—bookkeeping, tax filing, investment management, and so on.

There are no technology requirements or restrictions with SOA. You can build a service in any language with standards such as CORBA, remote procedure calls (RPC), or XML. the technology choices are SOAP and the Web Service Definition Language (WSDL); both XML-based.

WSDL describes the interface (the "contract"), while SOAP describes the data that is transferred. Because of the platform-neutral nature of XML, SOAP, and WSDL, Java is a popular choice for web-service implementation due to its OS-neutrality.

REST(REpresentational State Transfer)

REST, or REpresentational State Transfer, is an architectural style for providing standards between computer systems on the web, making it easier for systems to communicate with each other. REST-compliant systems, often called RESTful systems, are characterized by how they are stateless and separate the concerns of client and server. It is a software architecture style for distributed systems, particularly distributed hypermedia systems, such as the World Wide Web. It was developed in parallel with the HTTP/1.1



<https://www.codecademy.com/articles/what-is-rest>

FIGURE 5.1

A simple REST interaction between user and server in HTTP specification.

- ❑ **SOAP** stands for Simple Object Access Protocol whereas **REST** stands for Representational State Transfer.
- ❑ **SOAP** is a protocol whereas **REST** is an architectural pattern.
- ❑ **SOAP** only works with XML formats whereas **REST** work with plain text, XML, HTML and JSON.
- ❑ **SOAP** cannot make use of **REST** whereas **REST** can make use of **SOAP**

The REST architectural style is based on four principles:

- Resource Identification through URIs.
- Uniform, Constrained Interface.
- Self-Descriptive Message.
- Stateless Interactions.

Resource : The key abstraction of information in REST is a resource. Any information that can be named can be a resource, such as a document or image or a temporal service.

The RESTful web service exposes a set of resources which identify targets of interaction with its clients. Each particular resource is identified by a unique name, or more precisely, a Uniform Resource Identifier (URI) providing a global addressing space for resources involved in an interaction between components as well as facilitating service discovery. The URIs can be bookmarked or exchanged via hyperlinks, providing more readability.

URI – Uniform Resource Identifier

- The name of the object on the web
- Identifies a resource by
 - Name
 - Location
 - Or both

URL – Uniform Resource Locator

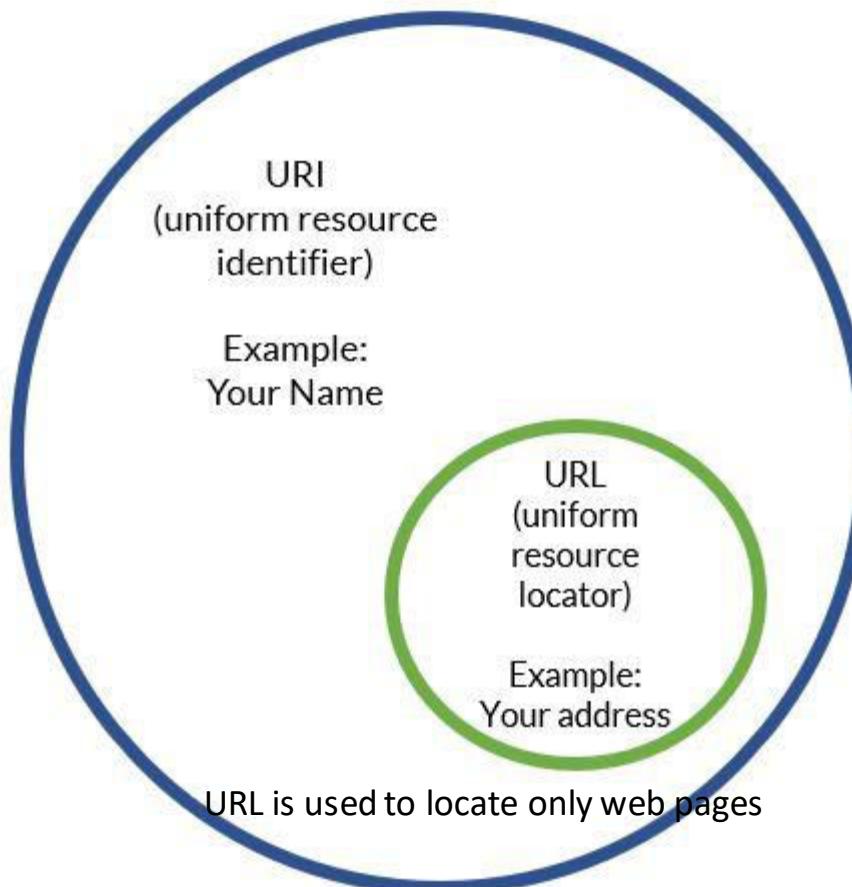
- Subset of an URI
- Specifies where to find a resource – the location
- How to retrieve the resource

CLOUD COMPUTING

REST (Cont.)

- A **URI** is an **identifier** of a specific resource like a page, book or a document.
- A **URL** is a special type of identifier that also tells us how to access it, such as HTTPs, FTP, etc
- Your name could be a URI because it identifies you, but it couldn't be a URL because it doesn't help anyone find your location.
- On the other hand, your address is both a URI and a URL because it both identifies you *and* it provides a location for you.

URI used in HTML, XML and other files XSLT (Extensible Stylesheet Language Transformations) and more



Interaction with RESTful web services is done via the HTTP standard, client/server cacheable protocol.

Resources are manipulated using a fixed set of four CRUD (create, read, update, delete) operations:

- GET — retrieve a specific resource or a collection of resources
- POST — Create or partial update of a resource
- PUT — Create or update resource by replacement.
- DELETE — remove a resource

Operations characteristics:

1. **Safe** : An operation is safe if does not modify resources.
2. **Idempotent**: An idempotent operation is an operation, action, or request that can be applied multiple times without changing the result, i.e. the state of the system, beyond the initial application. Making multiple identical requests has the same effect as making a single request.

Operation	Safe	Idempotent	When to use
GET	Yes	Yes	Mostly for retrieving resources. Can call multiple times.
PUT	No	Yes	Modifies a resource but no additional impact if called multiple times
POST	No	No	Modifies resources, multiple calls will cause additional effect if called with same parameter
DELETE	No	Yes	Removing a resource.

REST Principles - Self-Descriptive Message

A REST message includes enough information to describe how to process the message. This enables intermediaries to do more with the message without parsing the message contents.

In REST, resources are decoupled from their representation so that their content can be accessed in a variety of standard formats (e.g., HTML, XML, MIME, plain text, PDF, JPEG, JSON, etc.) i.e the resources can have **multiple representations**. RESTful systems empowers client to ask for data in a form they understand.

Example:

```
GET /articles/23 HTTP/1.1
Accept: text/html, application/xhtml
```

Client request

```
HTTP/1.1 200 (OK)
Content-Type: text/html
```

Server Response

Metadata about the resource is available and can be used for various purposes, such as cache control, transmission error detection, authentication or authorization, and access control.

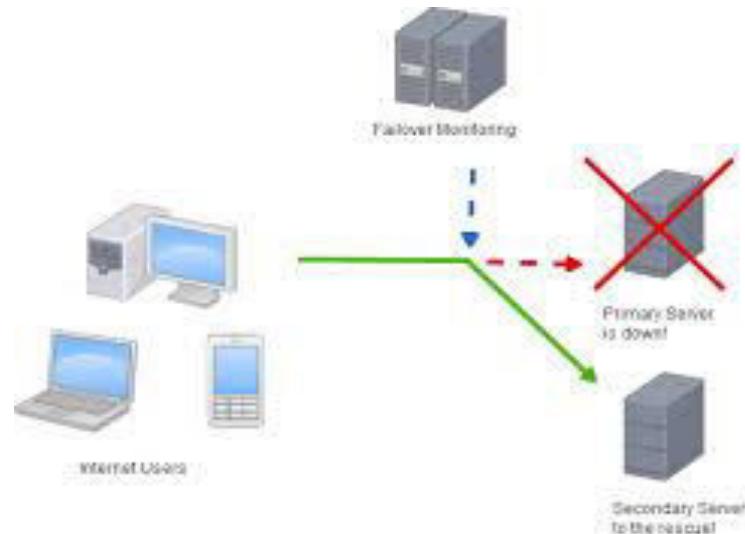
REST Principles - Stateless Interactions

Stateless Interactions: Systems that follow the REST paradigm are stateless, meaning that the server does not need to know anything about what state the client is in and vice versa. In this way, both the server and the client can understand any message received, even without seeing previous messages. Each client request is treated independently.

Statelessness benefits :

1. Clients isolated against changes on server
2. Promotes redundancy - unlocks performance:
 - a. don't really need to know which server client was talking to
 - b. No synchronization overhead

No state saved on server, so even if server fails, the client connects to another server and continue



1. Rest is not a standard :

It is a design and architectural style for large scale distributed systems

1. Rest is not same as http:

Http can also be used in non-RESTful way. Example : SOAP services that use http to transport data.

Web Service : a software system designed to support interoperable machine-to-machine interaction over a network.

The term “web service” is often referred to a self-contained, self-describing, modular application designed to be used and accessible by other software applications across the web. Once a web service is deployed, other applications and other web services can discover and invoke the deployed service. Other systems interact with the web service in a manner prescribed by its description.

A web service is one of the most common instances of an SOA implementation.

The technologies that make up the core of today's web services are:

1. Simple Object Access Protocol (SOAP)
2. Web Services Description Language (WSDL)
3. Universal Description, Discovery, and Integration (UDDI)

Simple Object Access Protocol (SOAP)

SOAP provides a standard packaging structure for transmission of XML documents over various Internet protocols, such as SMTP, HTTP, and FTP. By having such a standard message format, heterogeneous middleware systems can achieve interoperability i.e different systems can communicate and share services regardless of their underlying implementation.

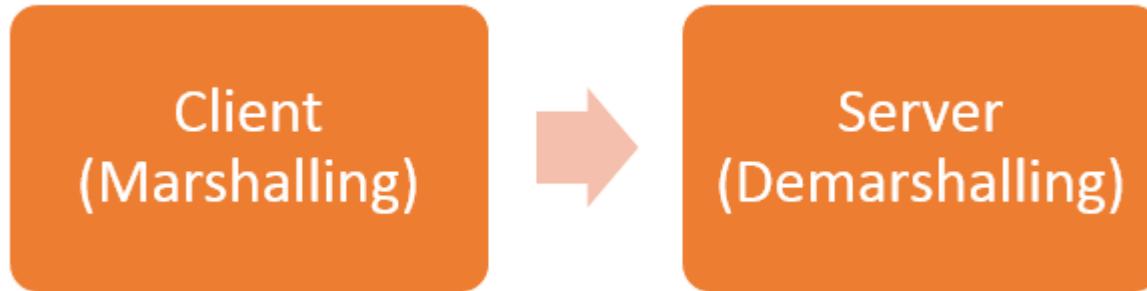
A SOAP message consists of a root element called *envelope*, The envelope element is the mandatory element in the SOAP message and is used to encapsulate all of the data in the SOAP message. It contains :

1. A *Header* element that contains header information – The header element can contain information such as authentication credentials which can be used by the calling application. It can be extended by intermediaries with additional application-level elements such as routing information ,transaction management, message parsing instructions.
2. a *body* element that carries the payload of the message.

The content of the payload will be marshaled by the sender's SOAP engine and unmarshaled at the receiver side, based on the XML schema that describes the structure of the SOAP message.

Ref - <https://www.guru99.com/soap-simple-object-access-protocol.html>

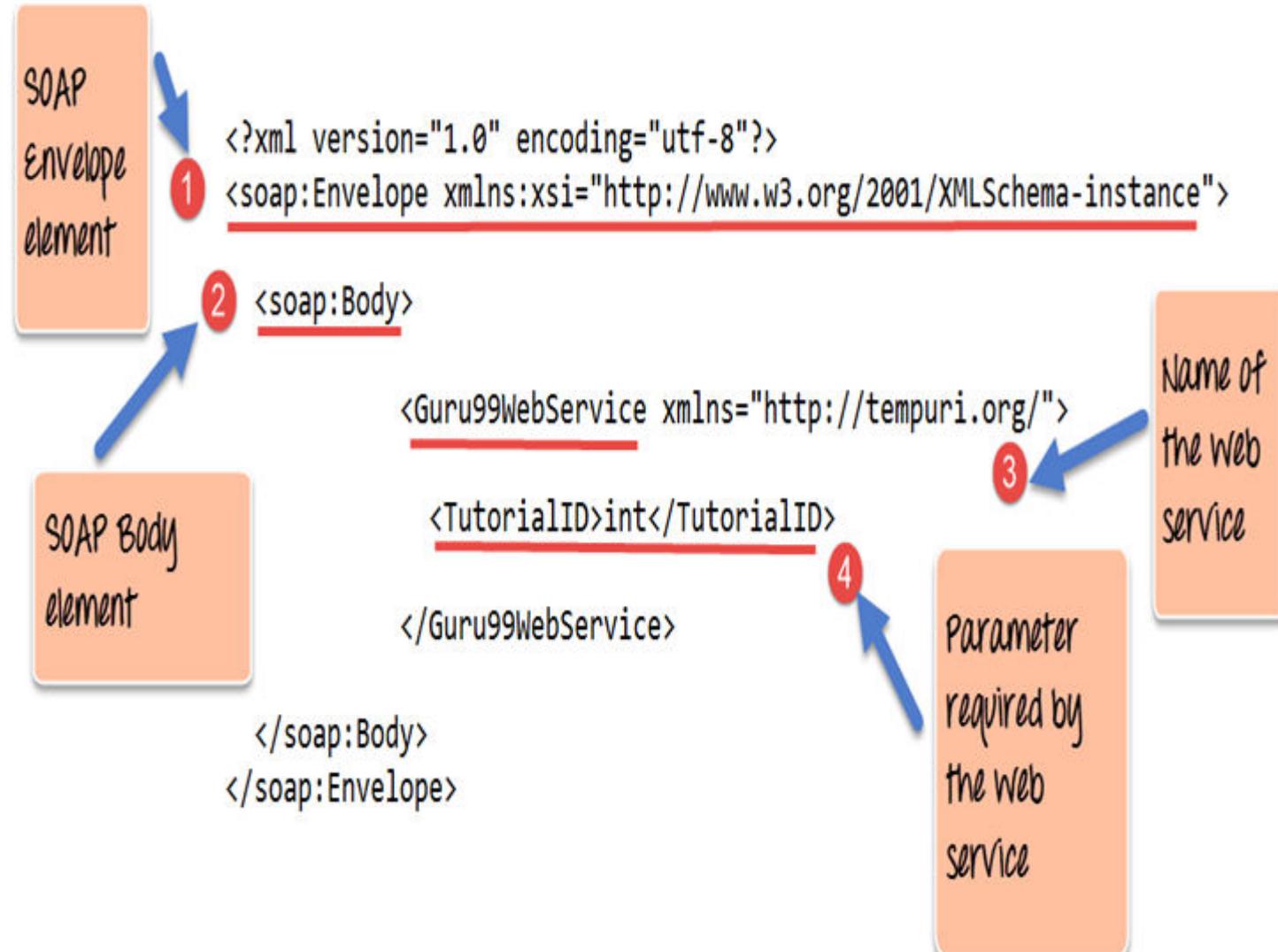
SOAP is an XML-based protocol for accessing web services over HTTP.



The client would format the information regarding the procedure call and any arguments into a SOAP message and sends it to the server as part of an HTTP request. This process of encapsulating the data into a SOAP message was known as **Marshalling**.

The server would then unwrap the message sent by the client, see what the client requested for and then send the appropriate response back to the client as a SOAP message. The practice of unwrapping a request sent by the client is known as **Demarshalling**.

SOAP Message Structure :



Web Services Description Language (WSDL)

WSDL describes the interface, a set of operations supported by a web service in a standard format. It standardizes the representation of input and output parameters of its operations as well as the service's protocol binding, the way in which the messages will be transferred on the wire. Using WSDL enables disparate clients to automatically understand how to interact with a web service.

Universal Description, Discovery, and Integration (UDDI)

UDDI provides a global registry for advertising and discovery of web services. Users can find web services by searching for names, identifiers, categories, or the specification implemented by the web service.

Universal Description, Discovery and Integration (UDDI) is a platform independent framework functioning like a directory that provides a mechanism to locate and register web services on the internet.

UDDI is an XML-based registry for business internet services. A provider can explicitly register a service with a *Web Services Registry* such as UDDI or publish additional documents intended to facilitate discovery of documents. The service users or consumers can search web services manually or automatically.

Web Services Description Language (WSDL) is a markup language that describes the web service.

https://www.tutorialspoint.com/uddi/uddi_quick_guide.htm

CLOUD COMPUTING

Web Services in action

(T2)

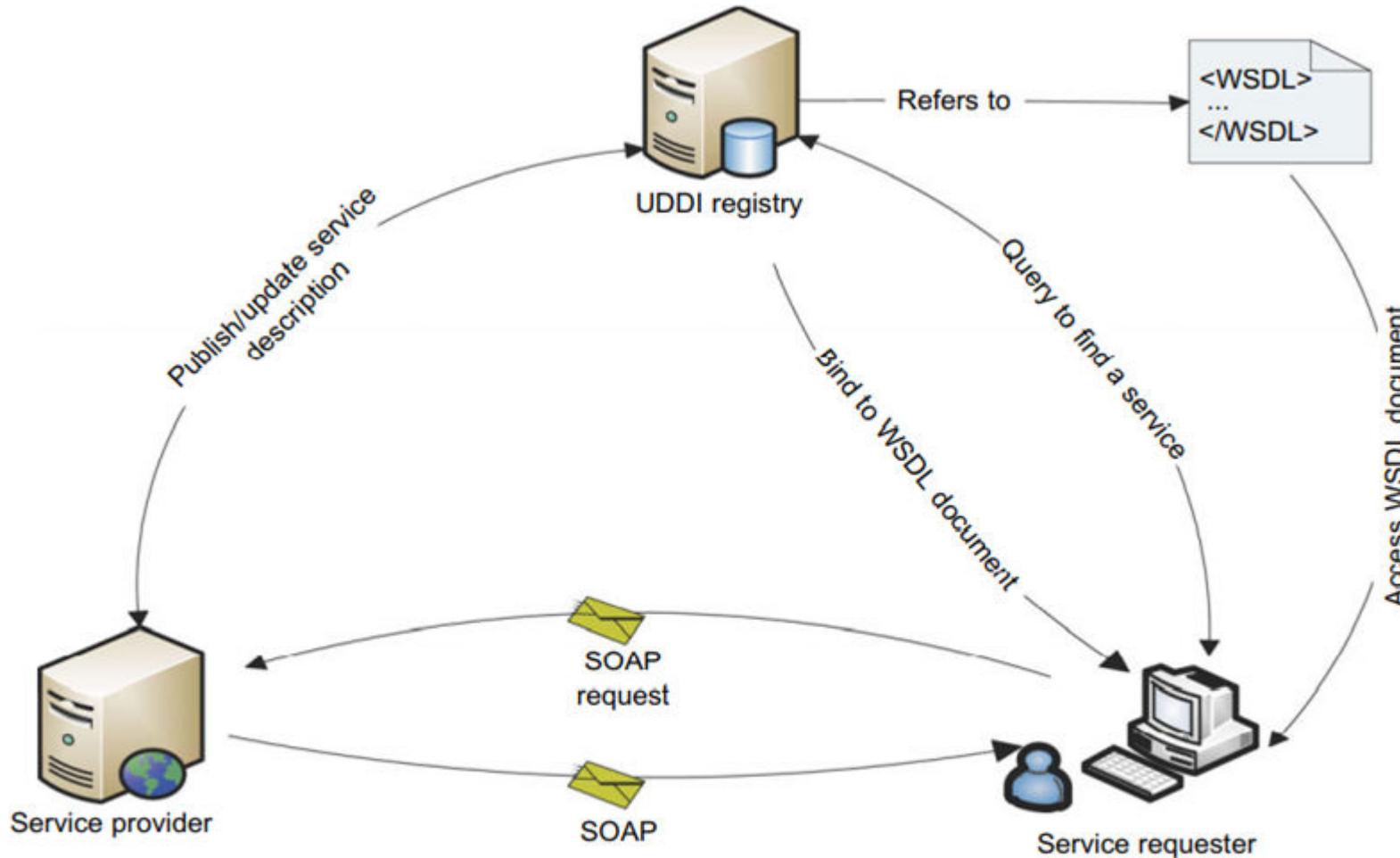


FIGURE 5.2

A simple web service interaction among provider, user, and the UDDI registry.

Refer to Figure 5.2(previous slide):

Step 1 : The service provider publishes its location and description to the UDDI registry.

Step 2: The service requester queries the UDDI registry using names, identifiers, or the specification. The UDDI registry finds the corresponding service and returns the WSDL document of the service.

Step 3: The service requester reads the WSDL document and forms a SOAP message binding to all constraints in the WSDL document.

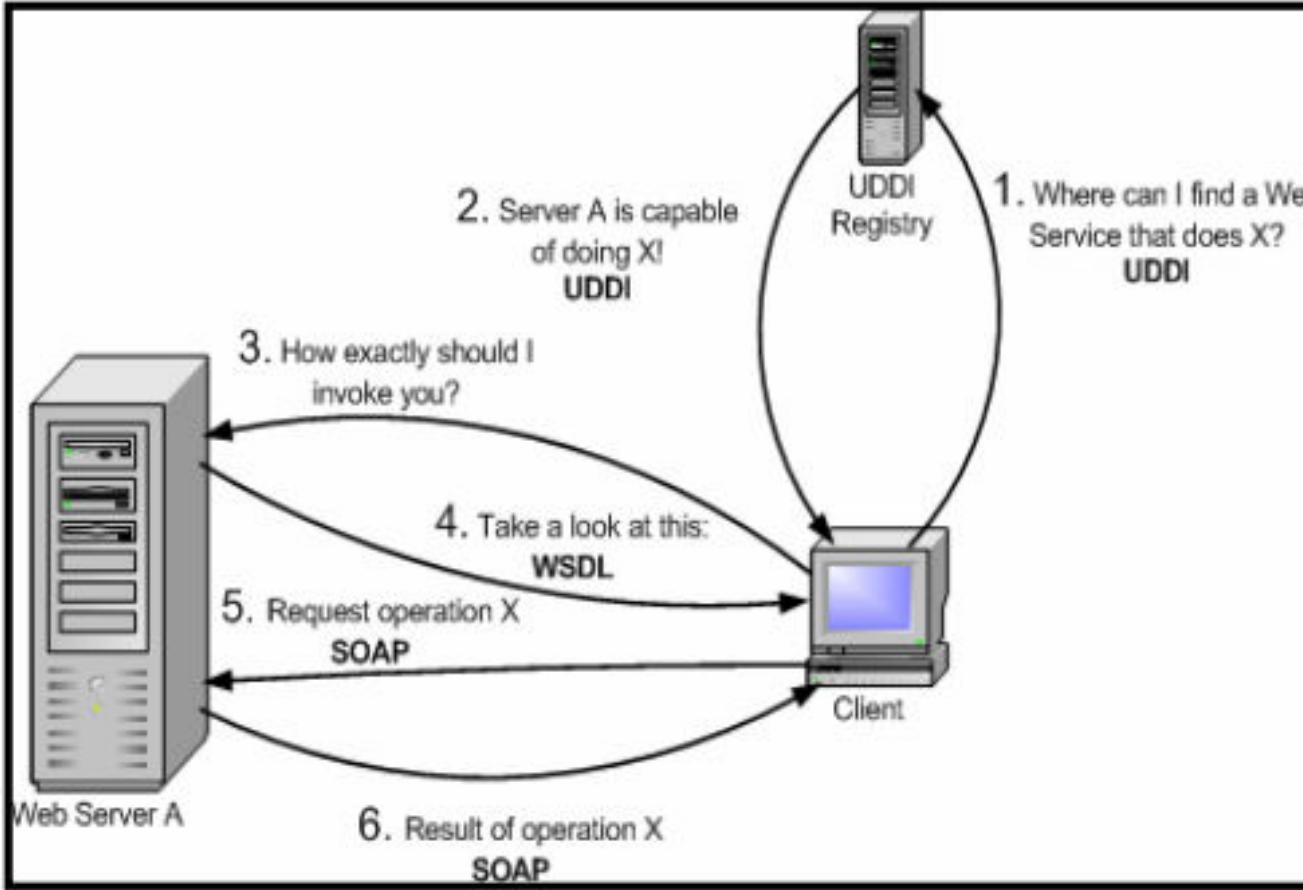
Step 4: This SOAP message is sent to the web service as the body of an HTTP/SMTP/FTP request.

Step 5: The web service unpacks the SOAP request and converts it into a command that the application can understand and executes the command.

Step 6: The web service packages the response into another SOAP message, which it sends back to the client program in response to its HTTP/SMTP/FTP request

Step 7: The client program unpacks the SOAP message to obtain the results

Another look at Web Services discovery and invocation process



Web services protocol stack

A web service protocol stack is a protocol stack (a stack of computer networking protocols) that is used to define, locate, implement, and make Web services interact with each other. A Web service protocol stack typically stacks four protocols:

1. **Transport Protocol:** responsible for transporting messages between network applications and includes protocols such as HTTP, SMTP, FTP, as well as the more recent Blocks Extensible Exchange Protocol (BEEP).
2. **Messaging Protocol:** responsible for encoding messages in a common XML format so that they can be understood at either end of a network connection. Currently, this area includes such protocols as XML-RPC, WS-Addressing, and SOAP.
3. **Description Protocol:** used for describing the public interface to a specific Web service. The WSDL interface format is typically used for this purpose.
4. **Discovery Protocol:** centralizes services into a common registry so that network Web services can publish their location and description, and makes it easy to discover what services are available on the network. Universal Description Discovery and Integration (UDDI) was intended for this purpose, but it has not been widely adopted.

CLOUD COMPUTING

REST vs RESTful

Basis terms

Restless Web Service

Restful Web Service

Architecture

It is not based on the principles of REST.

It is based on the principles of REST architecture and is integrable with other computer systems on the network.

REST principles

It does not uses REST principles.

It uses REST principles.

Communication

It uses SOAP services.

It uses REST services.

Data Formats

It supports only XML format.

It supports JSON, HTML, etc format.

Functions

These services uses service interface to show business logic.

These services uses URL to show business logic.

Usability and Flexibility

It is less usable and flexible for the users.

It is more usable and flexible for the users.

Security Aspects

More secure as it designs it's own security layer.

Less secure as it uses the security layers to communication protocols.

Bandwidth

It uses a small bandwidth.

It uses a large bandwidth.



THANK YOU

K.S.Srinivas
srinivasks@pes.edu
+91 80 2672 1983 Extn 701