Introduction to Cloud Computing

Giuliano Taffoni - I.N.A.F.

"Cloud Computing concept and architecture"

2024 @ Università di Trieste

Overview

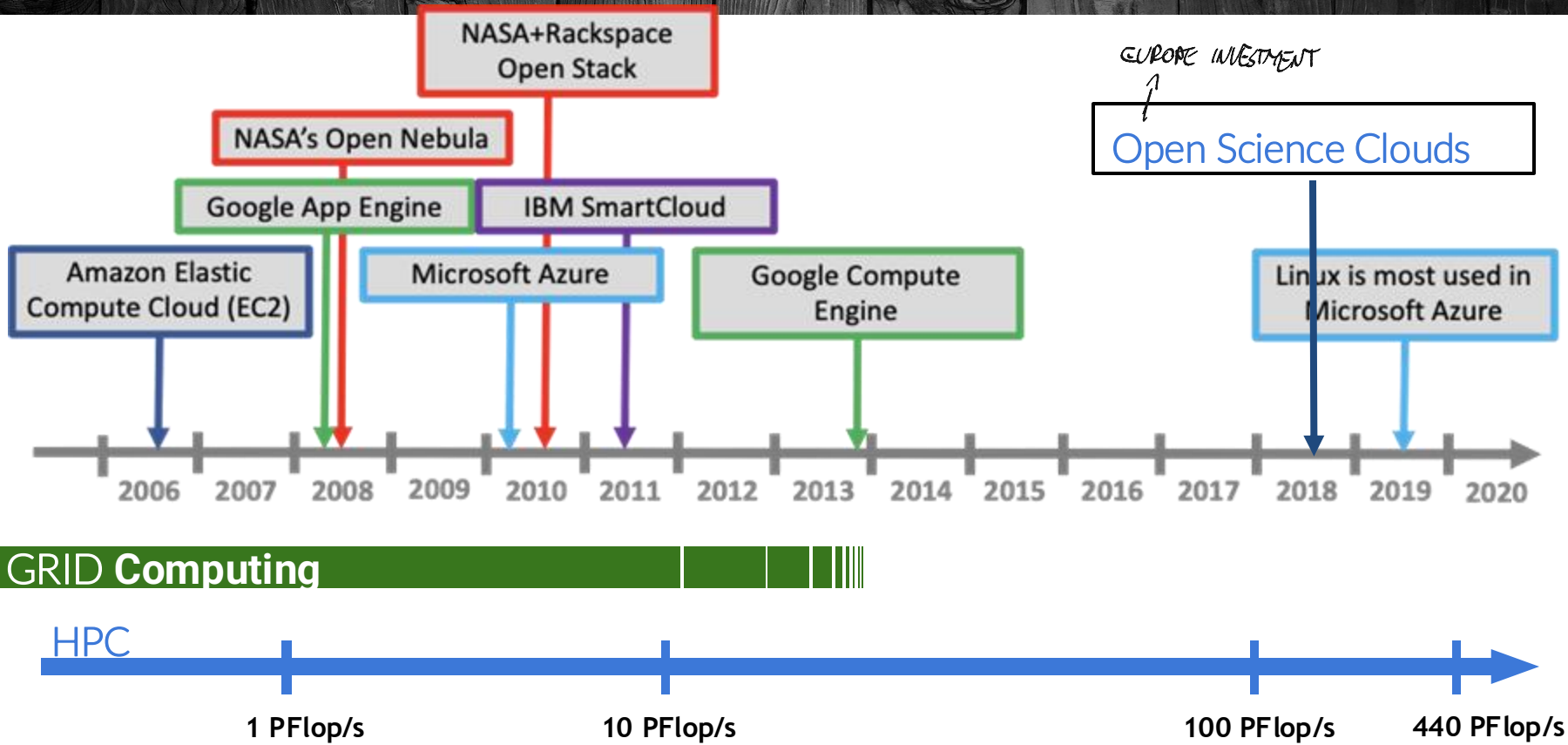
- What is Cloud Computing?
- Main concept underlying the cloud idea
- Main advantages and disadvantages
- Cloud architecture

Cloud Computing

There are several definitions of Cloud Computing:

- The concrete example of Utility Computing.
- The natural evolution of Grid Computing.
- A Distributed System

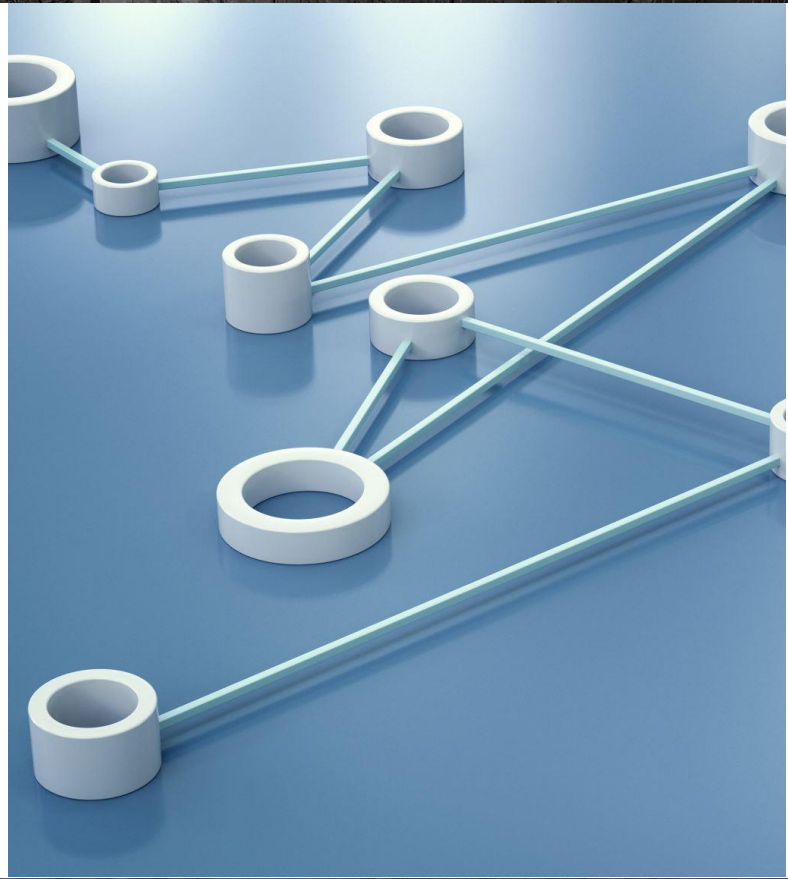
The Era of Computing



NIST Definition

Cloud computing is a model for enabling 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. “

“This cloud model promotes availability and is composed of five essential characteristics, three service models, and four deployment models.”



Cloud Computing primer

Cloud computing is the on-demand availability of computer system resources, especially data storage (“cloud storage”) and computing power, without direct active management by the user.

Clouds may be limited to a single organization (private/enterprise cloud), or be available to many organizations (public cloud), maybe a mixture of the two (hybrid cloud);

Cloud implements a pay-as-you-go model based on the concept of infinite resources availability;

Cloud is tightly coupled to Virtualization and Containerization, Microservicing and Composability

Cloud Base Concepts

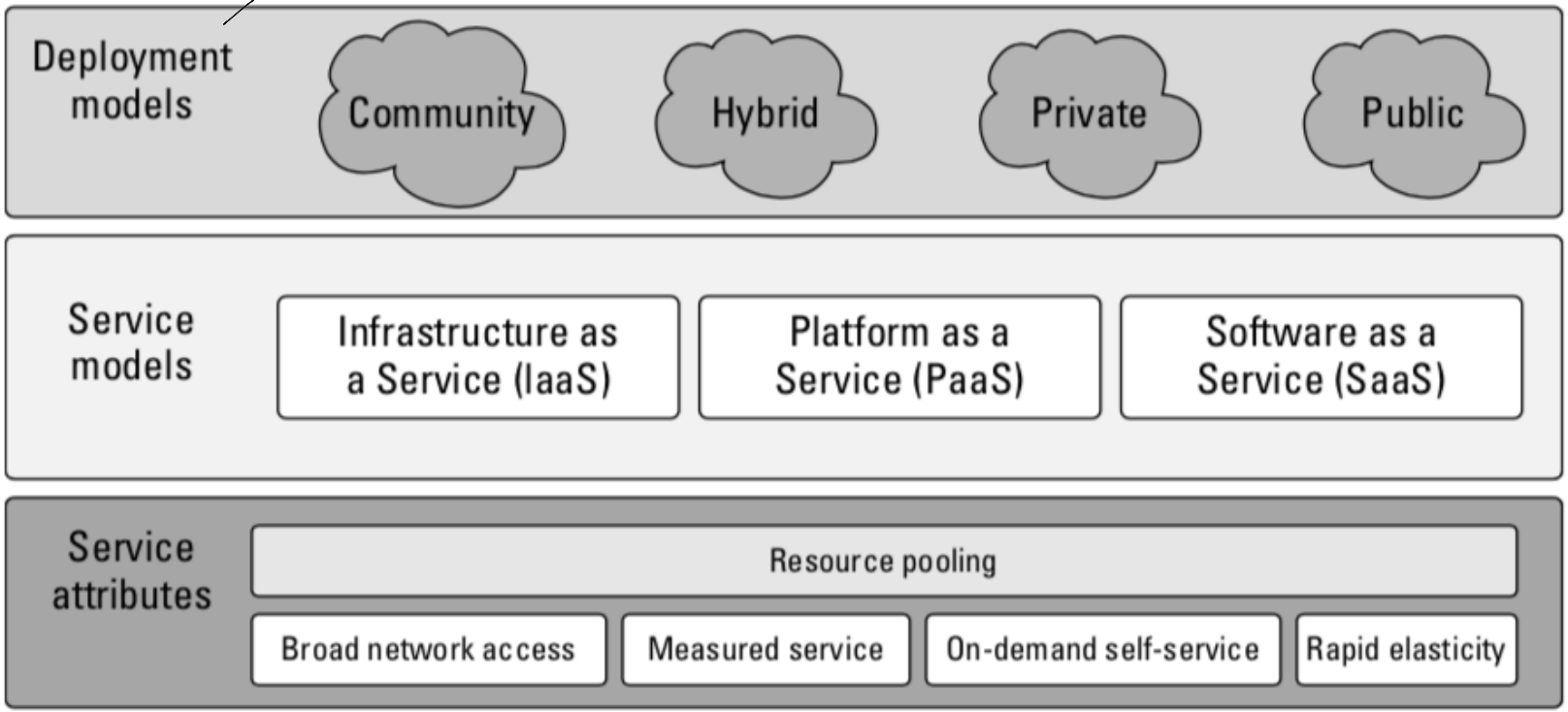
Abstraction: Cloud computing abstracts the details of system implementation from users and developers. Applications run on physical systems that aren't specified, data is stored in locations that are unknown, administration of systems is outsourced to others, and access by users is ubiquitous.

YOU DON'T KNOW WHERE THE INFRASTRUCTURE AND DATA ARE.
MAYBE THEY ARE NOT EVEN IN THE SAME BUILDING.

Virtualization: Cloud computing virtualizes systems by pooling and sharing resources. Systems and storage can be provisioned as needed from a centralized infrastructure, costs are assessed on a metered basis, multi-tenancy is enabled, and resources are scalable with agility.

NIST model view

HOW A CLOUD CAN BE DEPLOYED BASED ON YOUR NEEDS



Service Attributes: on-demand

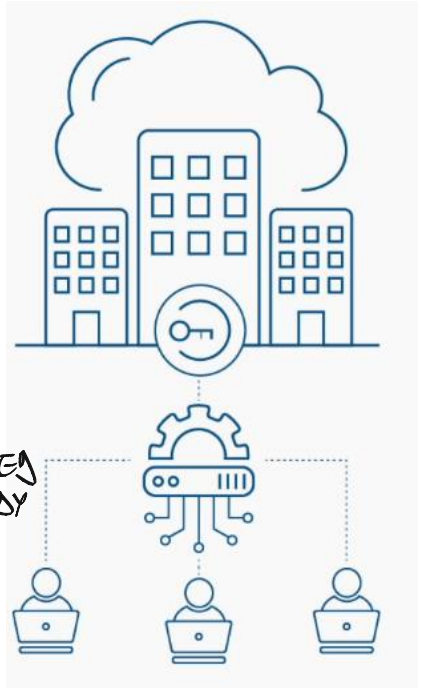
On-demand computing is a business computing model in which computing resources are made available to the user
on an "as needed" basis.

SOMETIME THE CLOUD IS EMPTY AND SOME TIMES IS CROWD. IT CAN BE UNPREDICTABLE.

Rather than all at once, **on-demand computing** allows cloud hosting companies to provide their clients with access to computing resources as they become necessary.

THE CLOUD HAVE TO BE USED AT EVERY TIME AND BE READY

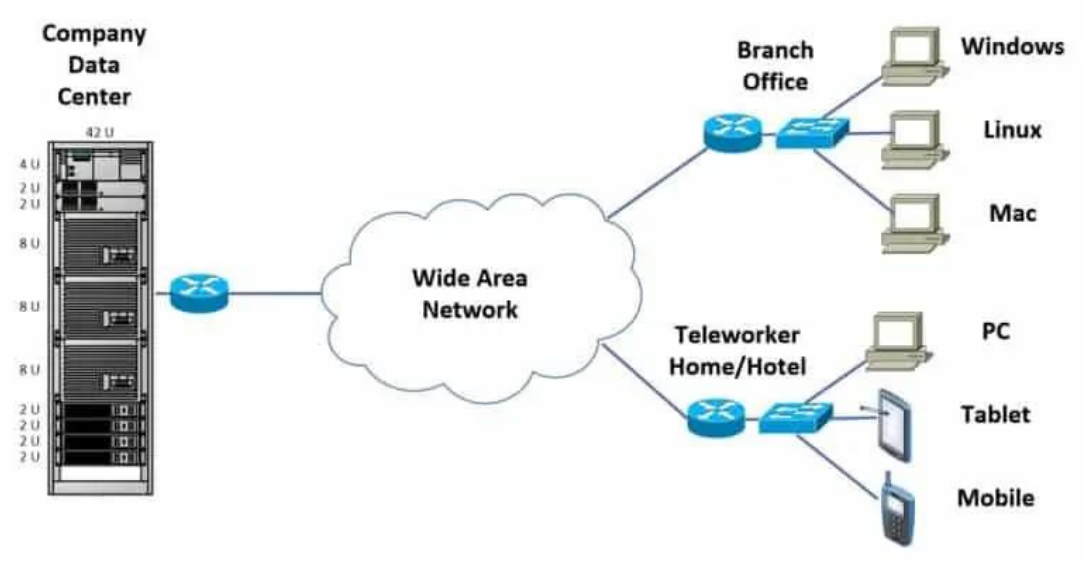
The on-demand computing model overcomes the common challenge that enterprises encountered of not being able to meet **unpredictable, fluctuating computing demands efficiently**.



Service Attributes: Broad Network Access

Capabilities are **available over the network** and accessible through standard mechanisms

Broad Network Access promotes use by heterogeneous thin or thick client platforms (e.g. laptop, Desktop, mobile devices etc.)

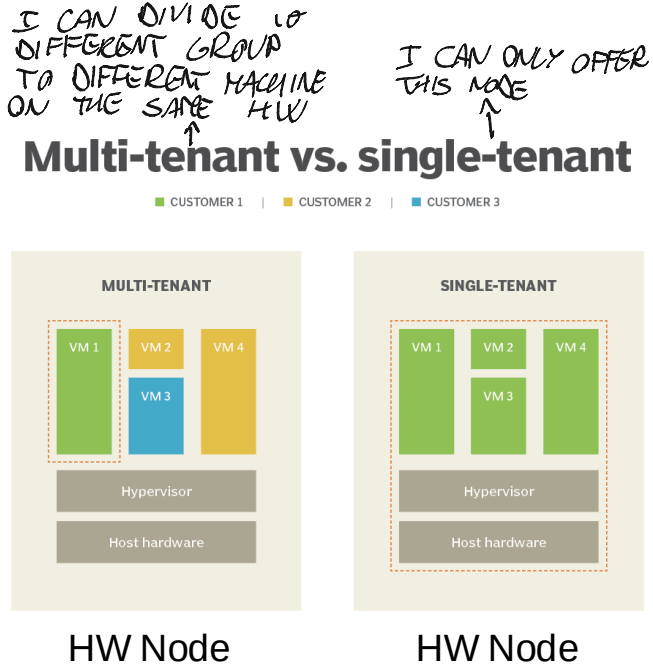


SA: Resource Pooling -- Multi-tenancy

Computing resources are storage, processing, memory, network bandwidth and virtual machines

Provider's **computing resources are pooled** to serve multiple consumers, allocated and deallocated as needed. **Tenants are Isolated.**

Location independence: there is no control over the exact location of the resources. This has major implications performance, scalability, security.



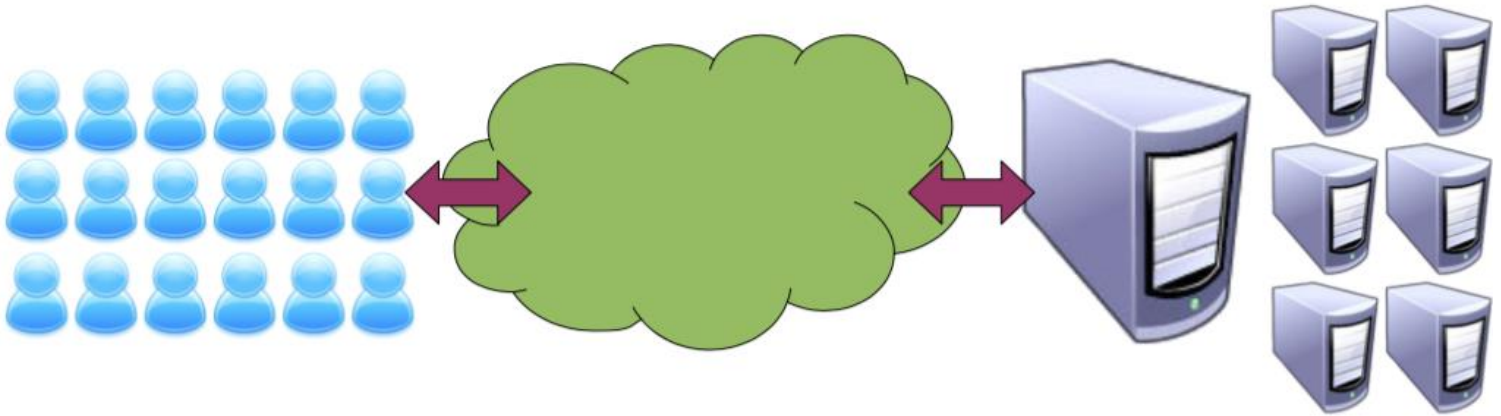
THE MAIN POINT REMAINS THE SAME: THE CLIENT DOESN'T NEED TO KNOW HOW IT WORKS. HE NEED, ASK, PAY AND USE.

Service Attribute: Rapid Elasticity

Capabilities can be rapidly and elastically provisioned

We can add resources by scale up or scale out systems.

Virtually unlimited resources



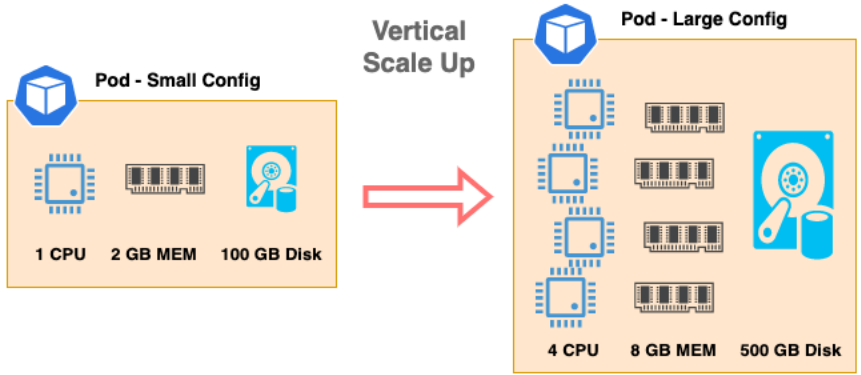
Vertical and horizontal scaling

What does it mean scale-up or scale-out a system?

Vertical and horizontal scaling

Scaling up (or vertical scaling) is adding more resources—like CPU, memory, and disk—to increase more compute power and storage capacity.

CONS: THE SYSTEM HAVE TO BE RESTARTED

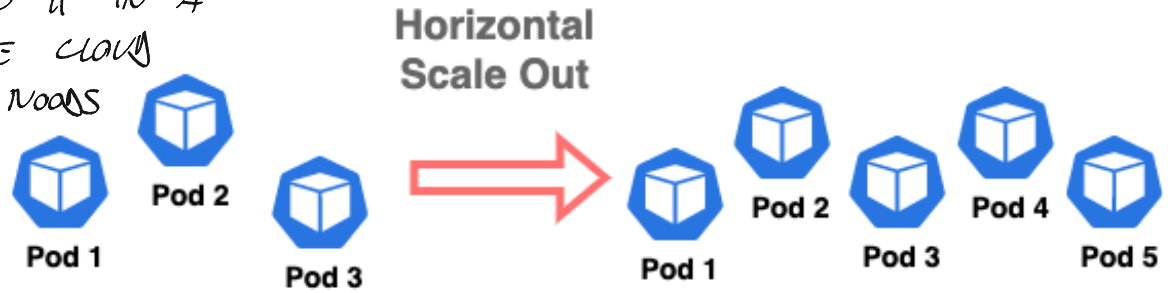


Vertical and horizontal scaling

Scaling out (or horizontal scaling) keep single resource and increase the number of resources

"It delivers long-term scalability. Scaling back is easy. You can utilize commodity servers."

IS COMMO DOING IT IN A HIGH PERFORMICE CLOUD JUST ADD MORE NODES

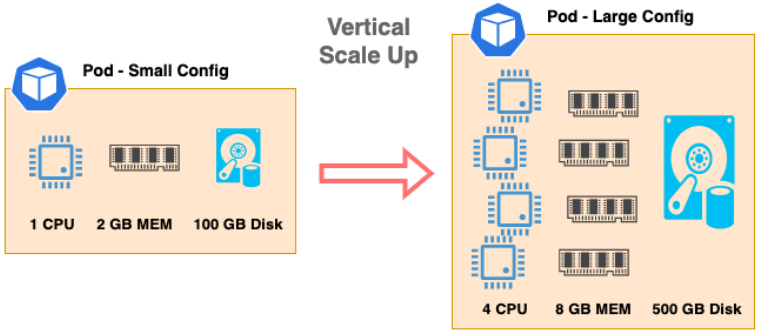


Vertical and horizontal scaling

What are the Scaling limits?

What are the Scaling Bottlenecks?

- HW PROBLEM :
  - MEMORY SIZE
  - FILE SYSTEM
  - CORE
  - MOTHERBOAR
- ELECTRICITY
- TEMPERATURE



Service Attribute: Measured service

Metering capability of service/resource abstractions in terms of storage, processing, bandwidth, active user accounts etc.

Remember the utility computing and pay as you go model.
(more on this later when we discuss deployment models)

Cloud additional attributes

Lower costs

Ease of utilization

Quality of Service (QoS)

Reliability

Outsourced IT management

Simplified maintenance and upgrade

Low Barrier to Entry

Questioning the Cloud approach

Do I need Cloud Computing if my organization is large enough to support IT solutions?

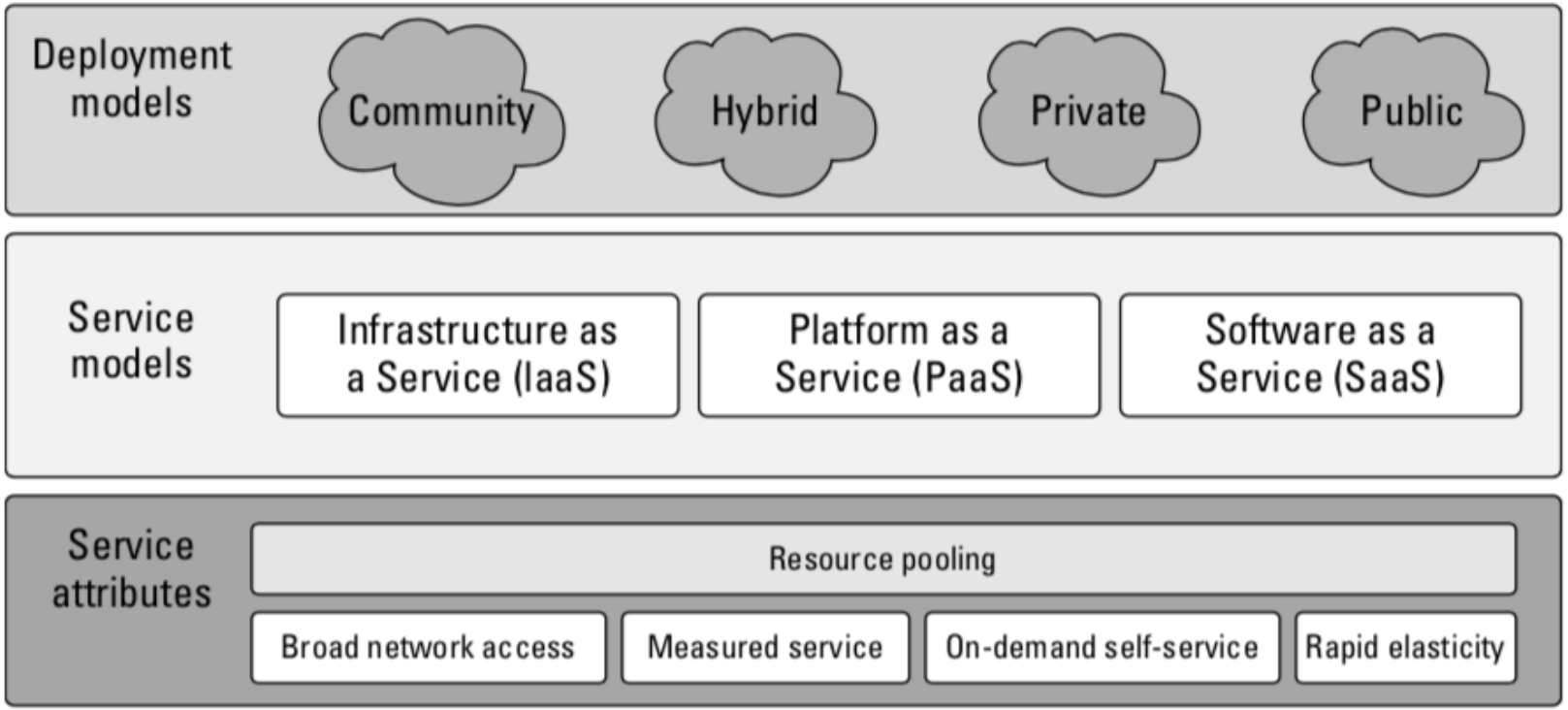
May I customize my application as much as I want?

Intrinsic WLAN latency is introduced (everything works on LAN, LAN fault means no access to cloud)

Cloud Computing is a stateless system, you need additional tools as service brokers, transaction manager, and middleware

Privacy and Security

NIST model view



Cloud Service Models

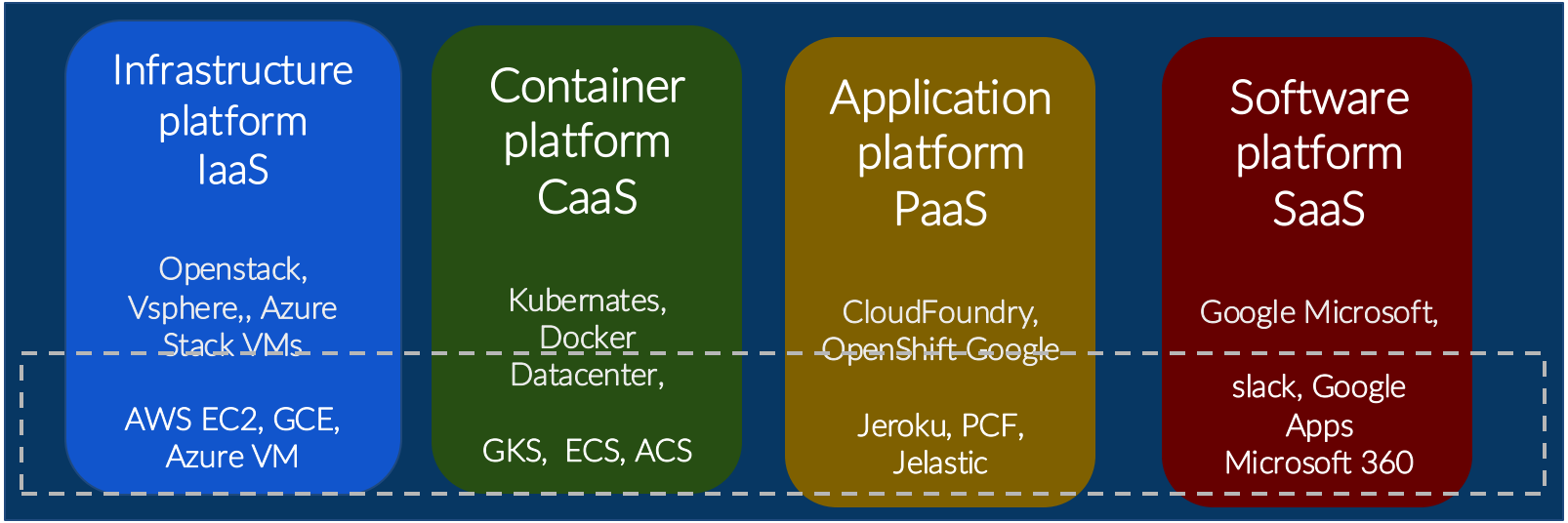
XaaS, or "<*Something*> as a Service"

Infrastructure as a Service: IaaS provides virtual machines, virtual storage, virtual infrastructure, and other hardware assets as resources that clients can provision.

Platform as a Service: PaaS provides virtual machines, operating systems, applications, services, development frameworks, transactions, and control structures. I NEED A PYTHON ENVIRONMENT (DATABASE, FRAMEWORK)

Software as a Service: SaaS is a complete operating environment with applications, management, and the user interface. GOOGLE DOCS

Cloud Service models



Cloud Deployment Models

Public cloud: The public cloud infrastructure is available for public use alternatively for a large industry group and is owned by an organization selling cloud services.

Commercial cloud: The private cloud infrastructure is operated for the exclusive use of an organization. The cloud may be managed by that organization or a third party. Private clouds may be either on- or off-premises.

Hybrid cloud: A hybrid cloud combines multiple clouds (commercial, community of public) where those clouds retain their unique identities, but are bound together as a unit.

Community cloud: A community cloud is one where the cloud has been organized to serve a common function or purpose.

Cloud Architecture

The advances in system virtualization is mixed with standard networking protocols already used in Internet.

The cloud creates a system where resources can be pooled and partitioned as needed.

Cloud architecture can couple software running on virtualized hardware in multiple locations to provide an on- demand service to user-facing hardware and software

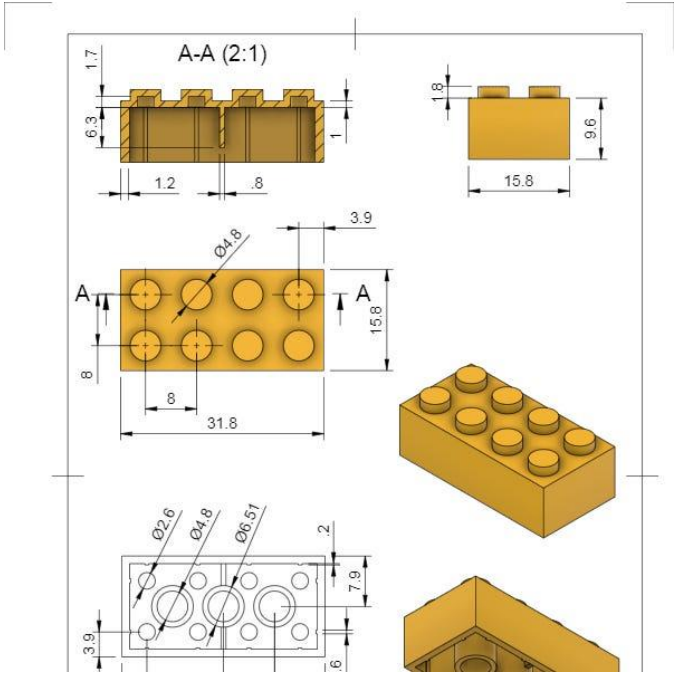
Virtualized resources are easy to optimize and modify

How Can I build a Cloud?

Composability

ALL IS A SMALL UNITS THA CAN BE STACK AND WORK WITH EACH OTHER

- Infrastructures
- Platforms
- Virtual Appliance
- Communication protocols
- Clients
- New way to build applications



Composability

Each component used during your development process is pluggable and can be replaced, scaled, and consistently improved to help you meet your business needs. When establishing composable architecture, each component of your development stack and microservices should be able to communicate with one another — regardless of differences in language or code.

A composable system uses components to assemble services that can be tailored for a specific purpose using standard part.

Modular: It is a self-contained and independent unit that is cooperative, reusable, and replaceable.
Stateless: A transaction is executed without regard to other transactions or requests.

Micro-Service

EVERY TIME YOU WANT TO CHANGE SOMETHING YOU HAVE TO REWRITE EVERYTHING

Monolithic Application: everything is integrated

Micro-service

Independent and self-contained units that perform a given task.
That sound great!


THE DEPENDENCIES WILL KILL YOU

Micro-service

VERY GOOD WHEN EXECUTE IN CONTAINER THAT ISOLATE THE TASK

Isolate tasks - a task can range from just a simple function (i.e. serve a file to download) to complex computer programs (i.e. classify images using a neural network).

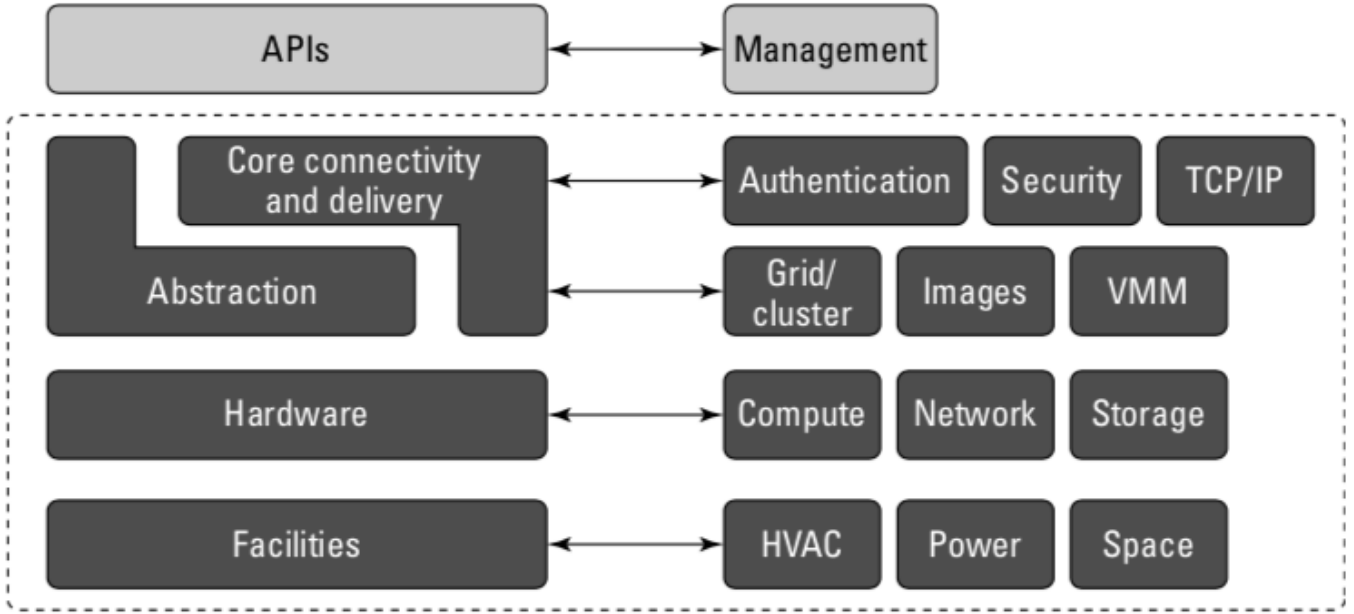
Standard Access -- Microservices are interacted with using a well-defined interface, usually a REST API over HTTP, but a Secure Shell protocol (SSH) is a perfectly viable interface as well.

Stateless operations

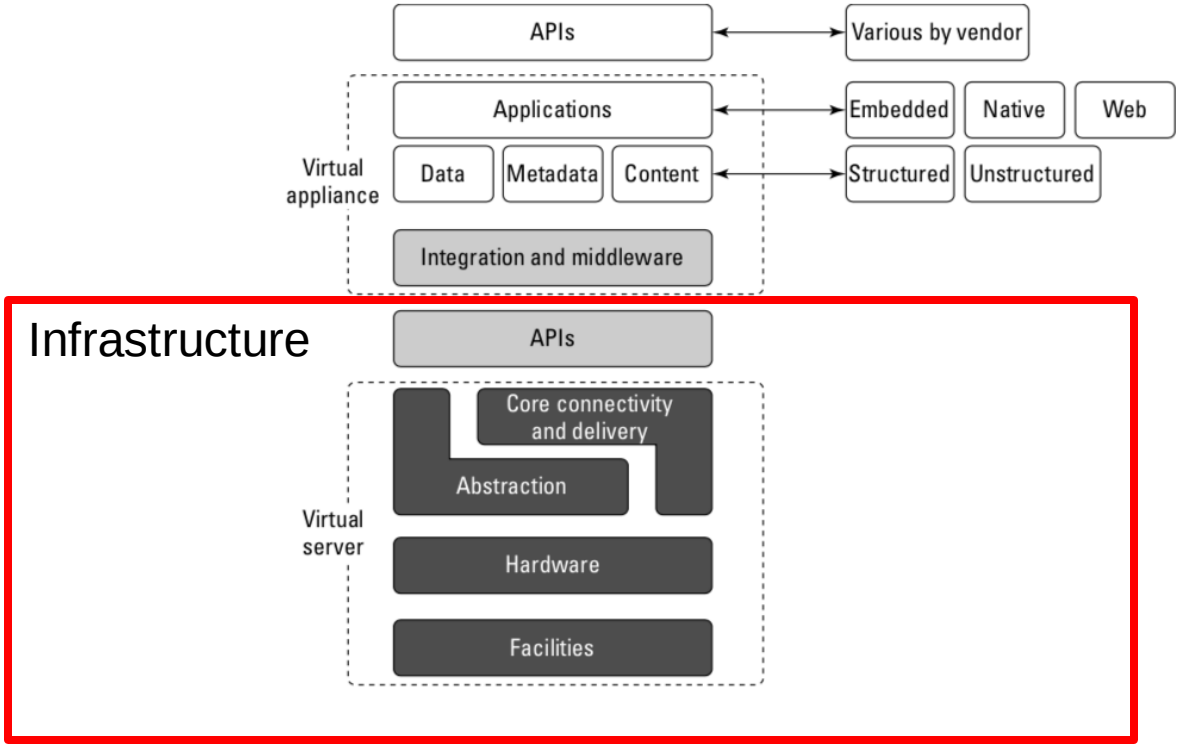
Microservices are heavily used in the so called "devops" space, and widely used for the underlying building blocks of modern Clouds.

Cloud Infrastructure

Based on VM or container technology



Cloud: Platform

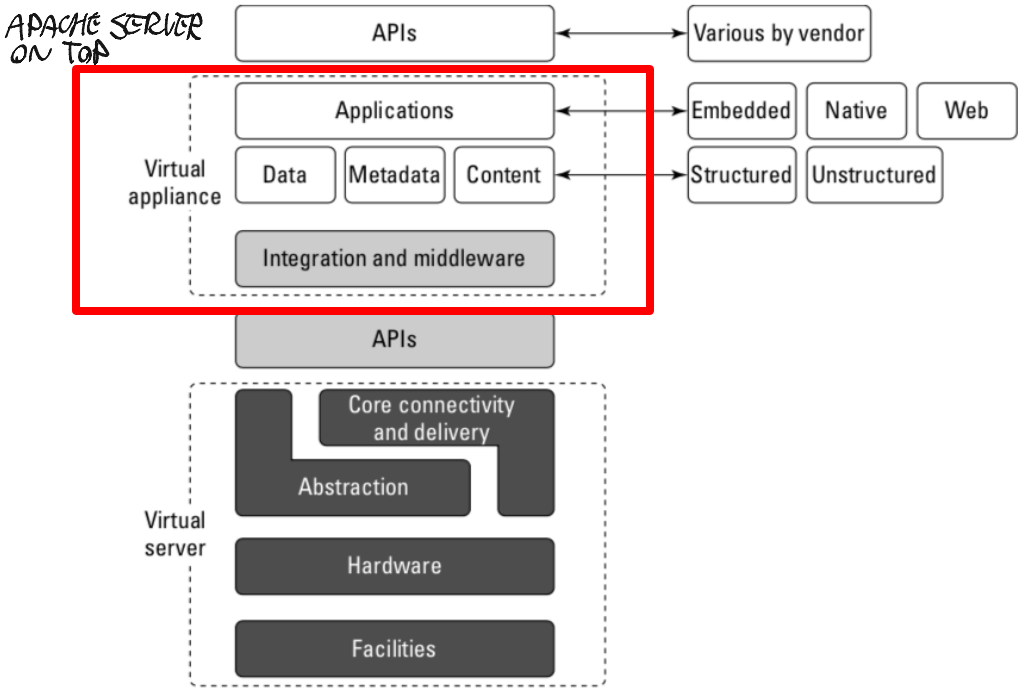


Virtual Appliance

Applications such as a Web server or database server that can run on a virtual machine image are referred to as virtual appliances.

Virtual appliances are software installed on virtual servers.

Virtual appliances are basis for assembling more complex services, the appliance being one of your standardized components (see IaaS lecture for example of VA)



Communication Protocols

Cloud computing arises from services available over the Internet communicating using the standard Internet protocol suite underpinned by the HTTP and HTTPS transfer protocols.

RCP → XML-RPC

SOAP
Web Service Description Language

REST
Atom publishing protocol

5 minutes of REST

"Representational State Transfer is intended to evoke an image of how a well-designed Web application behaves: a network of web pages (a virtual state-machine), where the user progresses through an application by selecting links (state transitions), resulting in the next page (representing the next state of the application) being transferred to the user and rendered for their use."

ONE WEY TO DESIGN WEB APPLICATION
NOT A PROTOCOL, JUST A DESIGN

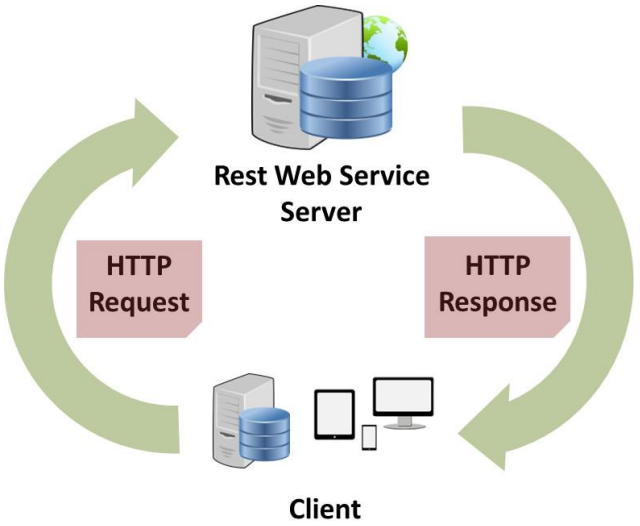
Roy Fielding

REST is...

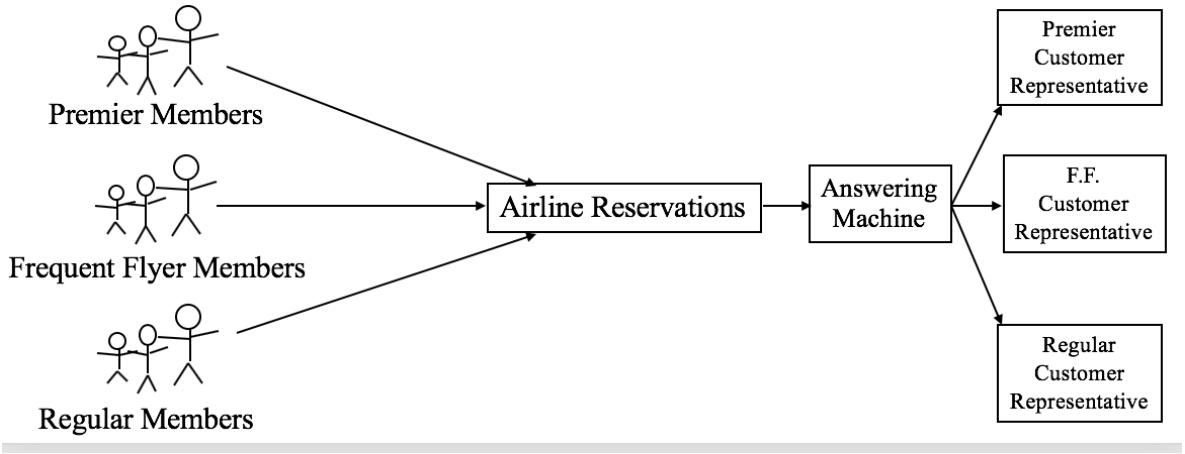
...is a design pattern.

It is a certain approach to creating Web Services.

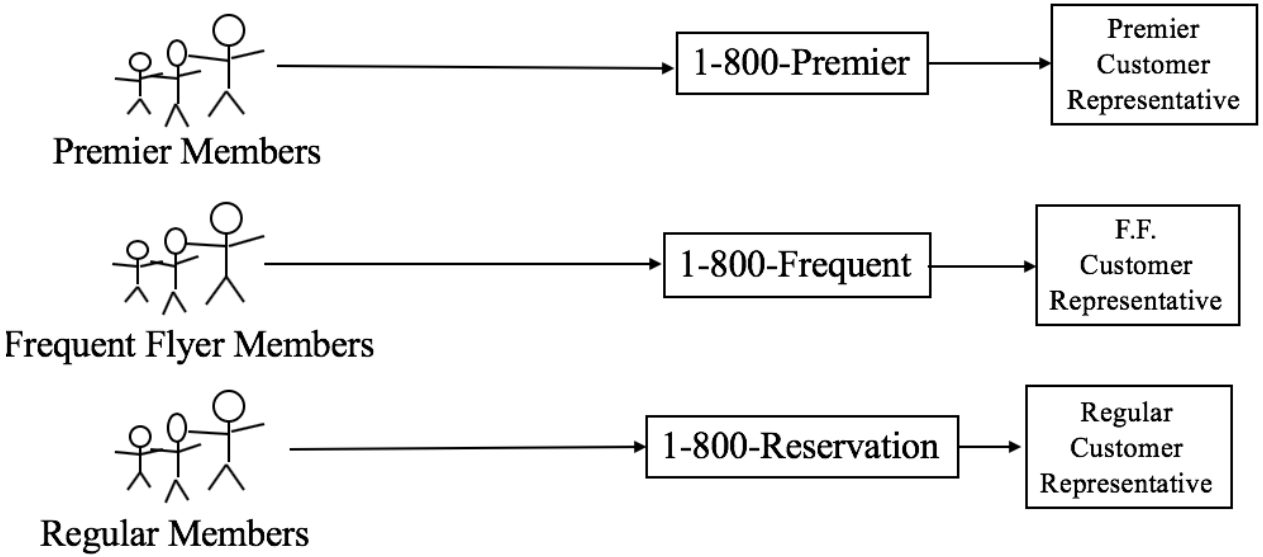
To understand the REST design pattern, let's look at an example



Airline Reservation Service



Airline Reservation Service



Advantages of REST approach

- The different URLs are discoverable by search engines and UDDI registries.
- It's easy to understand what each service does simply by examining the URL, *i.e.,* it exploits the Principle of Least Surprise.
- There is no need to introduce rules. Priorities are elevated to the level of a URL. "What you see is what you get."
- It's easy to implement high priority - simply assign a fast machine at the premier member URL.
- There is no bottleneck. There is no central point of failure.
- Consistent with Axiom 0.

Tim Berners-Lee Web Design, Axiom 0

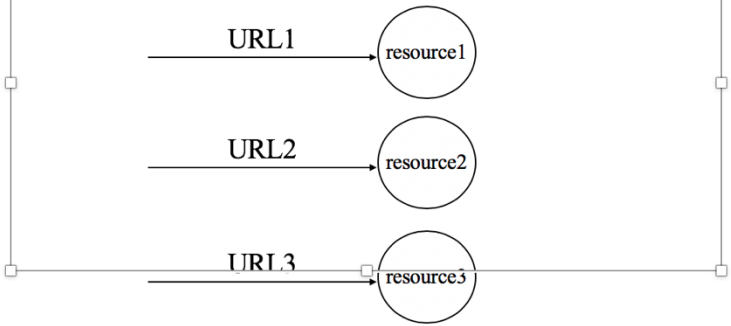
This approach is based upon the incorrect assumption that a URL is "expensive" and that their use must be rationed.

It violates Tim Berners-Lee Web Design, Axiom 0

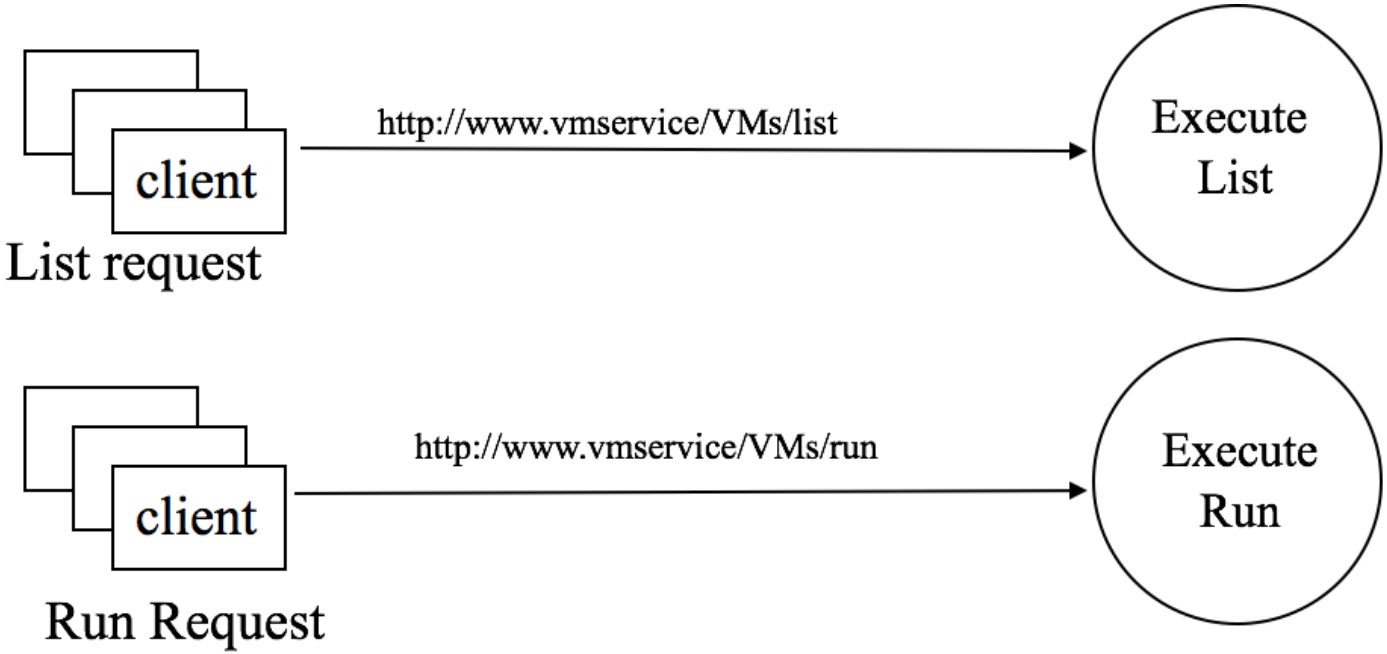
Web Design, Axiom 0

(Tim Berners-Lee, director of W3C)

Axiom 0: all resources on the Web must be uniquely identified with a URI.



Virtual Machine service



REST design pattern

Resources: Every distinguishable entity is a resource. A resource may be a Web site, an HTML page, an XML document, a Web service, a physical device, etc.

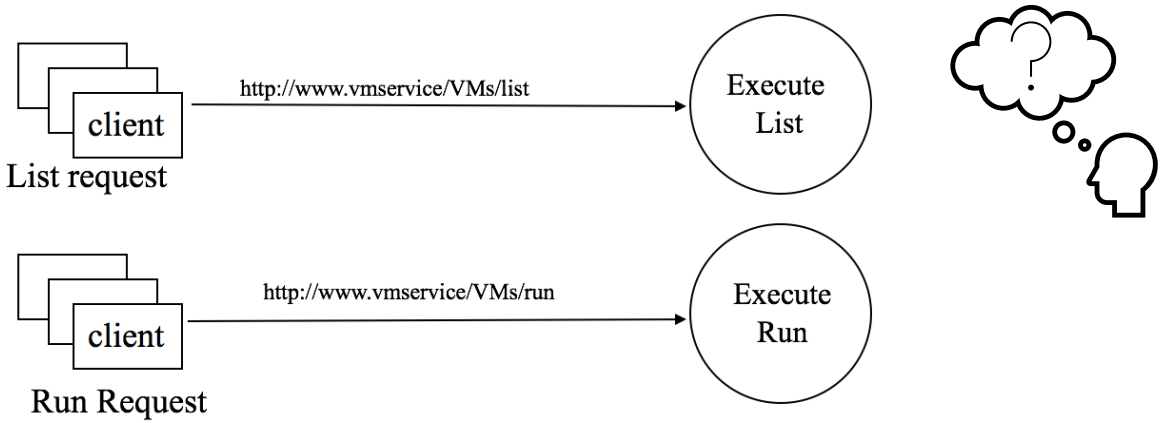
URLs Identify Resources: Every resource is uniquely identified by a URL. This is Tim Berners-Lee Web Design, Axiom 0.

REST Implementation

1) Identify all the conceptual entities that we wish to expose as services.
2) Create a URL to each resource.
3) Categorize our resources according to whether clients can just receive a representation of the resource (using an HTTP GET), or whether clients can modify (add to) the resource using HTTP POST, PUT, and/or DELETE).
4) All resources accessible via HTTP GET should be side-effect free. That is, the resource should just return a representation of the resource. Invoking the resource should not result in modifying the resource.
5) Put hyperlinks within resource representations to enable clients to drill down for more information, and/or to obtain related information.
6) Design to reveal data gradually. Don't reveal everything in a single response document. Provide hyperlinks to obtain more details.

REST Implementation

7) Specify the format of response data using a schema (DTD, W3C Schema, RelaxNG, or Schematron). For those services that require a POST or PUT to it, also provide a schema to specify the format of the response.
8) Describe how our services are to be invoked using either a WSDL document, or simply an HTML document.



5 min of REST…in summary

Client-Server: a pull-based interaction style (Client request data from servers as and when needed).
Stateless: each request from client to server must contain all the information necessary to understand the request, cannot take advantage of any stored context on the server.
Cache: to improve network efficiency, responses must be capable of being labelled as cacheable or non-cacheable.
Uniform interface: all resources are accessed with a generic interface (e.g., HTTP GET, POST, PUT, DELETE).
Named resources - the system is comprised of resources which are named using a URL.
Interconnected resource representations - the representations of the resources are interconnected using URLs, thereby enabling a client to progress from one state to another.

Key concepts to know…

1) Definition of Cloud Computing
2) Cloud Service Models
3) Cloud deployment Models
4) Cloud Main Attributes