Cloud Computing

Definitions (I)

We have redefined Cloud Computing to include everything that we already do. I do not understand what we would do differently other then change the working of some of our ads.

Larry Ellison
Oracle CEO

It's stupidity. It's worse than stupidity: it's a marketing hype campaign. Somebody is saying this is inevitable – and whenever you hear somebody saying that, it's very likely to be a set of businesses campaigning to make it true.

Richard Stallman
GNU & FSF father

Cloud computing is the use of computing resources (hardware and software) that are delivered as a service over a network (typically the Internet).

Wikipedia

Definitions (III)

National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

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.

http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

Definitions (III)

Cloud computing refers to both the applications delivered as services over the Internet and the hardware and systems software in the data centers that provide those services.
The services themselves have long been referred to as Software as a Service (SaaS), so we use that term.
The datacenter hardware and software is what we will call a Cloud.
When a Cloud is made available in a pay-as-you-go manner to the public [...] the service being sold is Utility Computing.

http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf

Properties and Characteristics

Scalability
Elasticity

- Dynamic provision
- Multi-tenant design

Manageability
Interoperability

- Autonomic Control
- System monitoring
- Billing System

Availability
Reliability

- Fault tolerance
- System resilience
- Security

Accessibility
Portability

- Uniform access
- Thin client

Performance
Optimization

- Parallel processing
- Load balancing
- Job scheduling

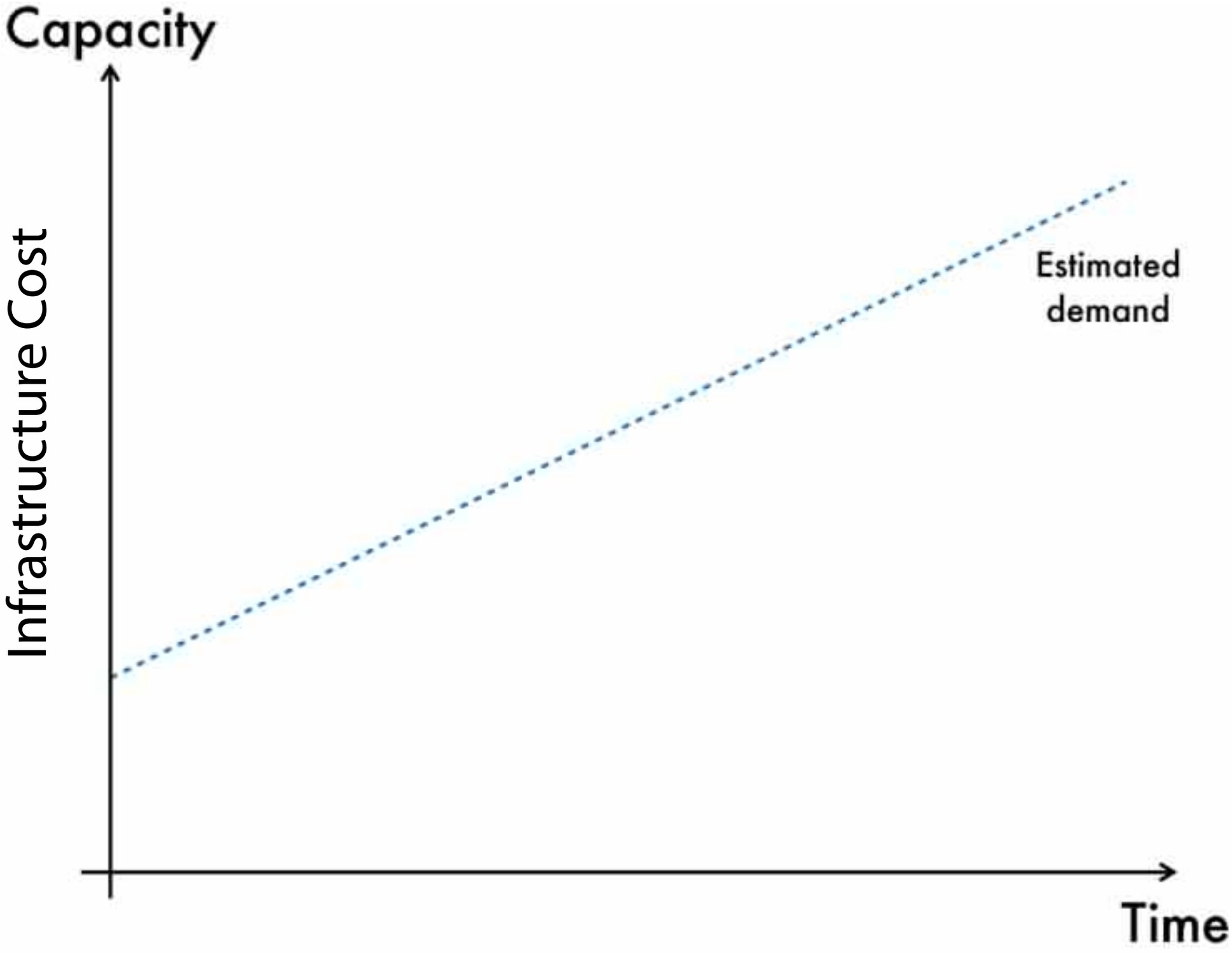
Scalability and Elasticity

- Scalability
  - A desirable property of a system, a network, or a process, which indicates its ability to either handle growing amounts of work in a graceful manner or to be readily enlarged.

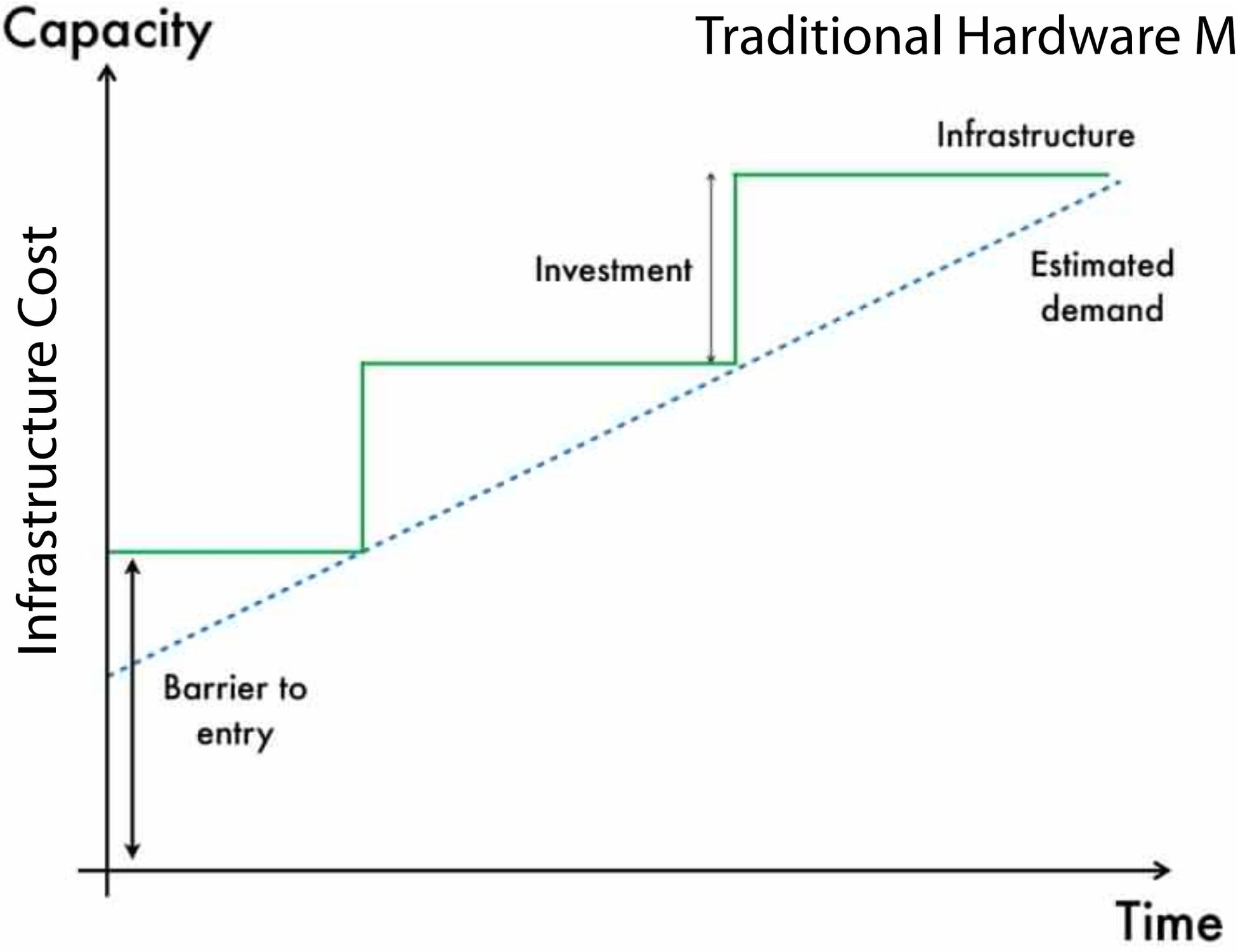
- Elasticity
  - The ability to apply a quantifiable methodology that allows for the basis of an adaptive introspection with in a real time infrastructure.

- How to achieve?
  - Dynamic provisioning
  - Multi-tenant design

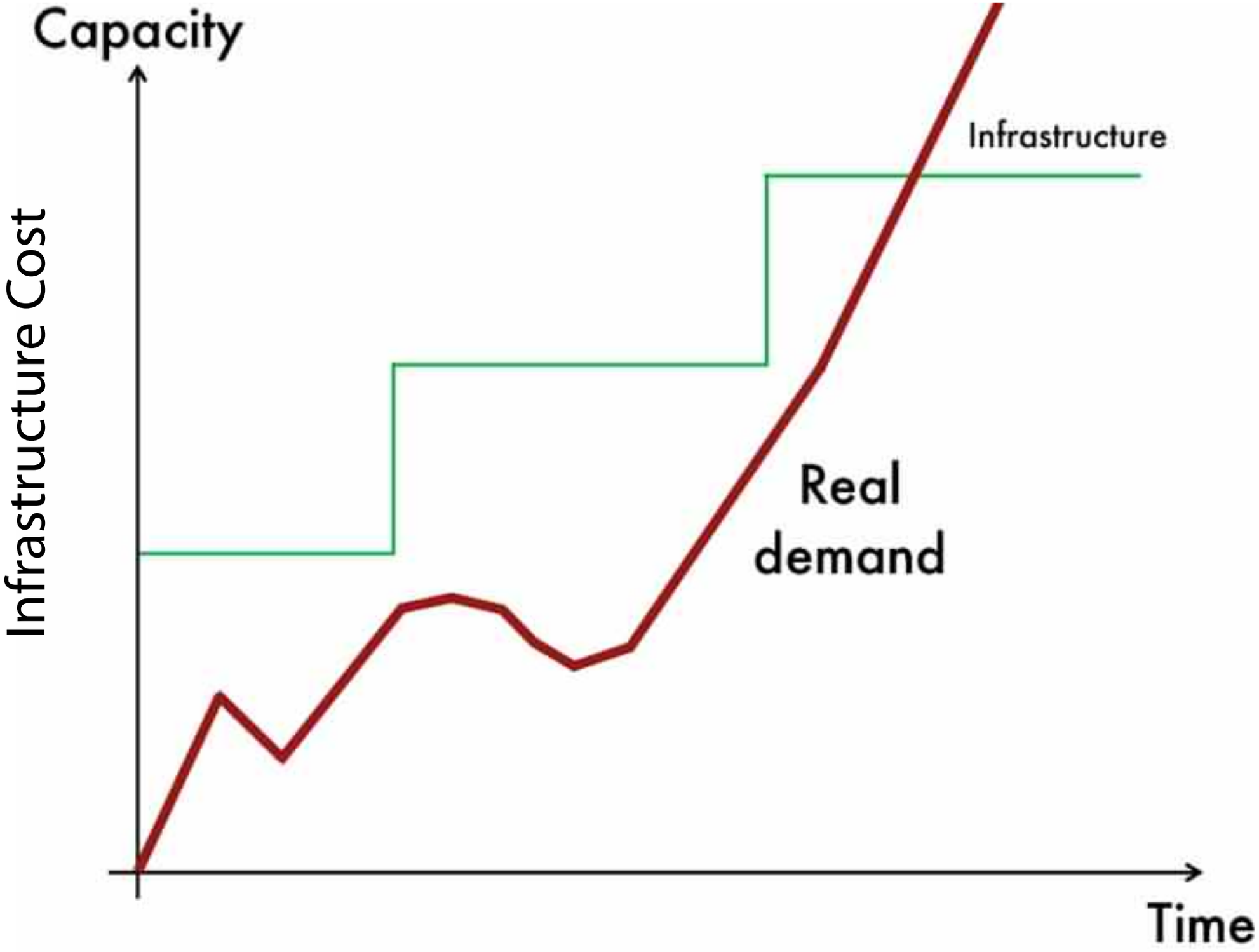
Elasticity (I)



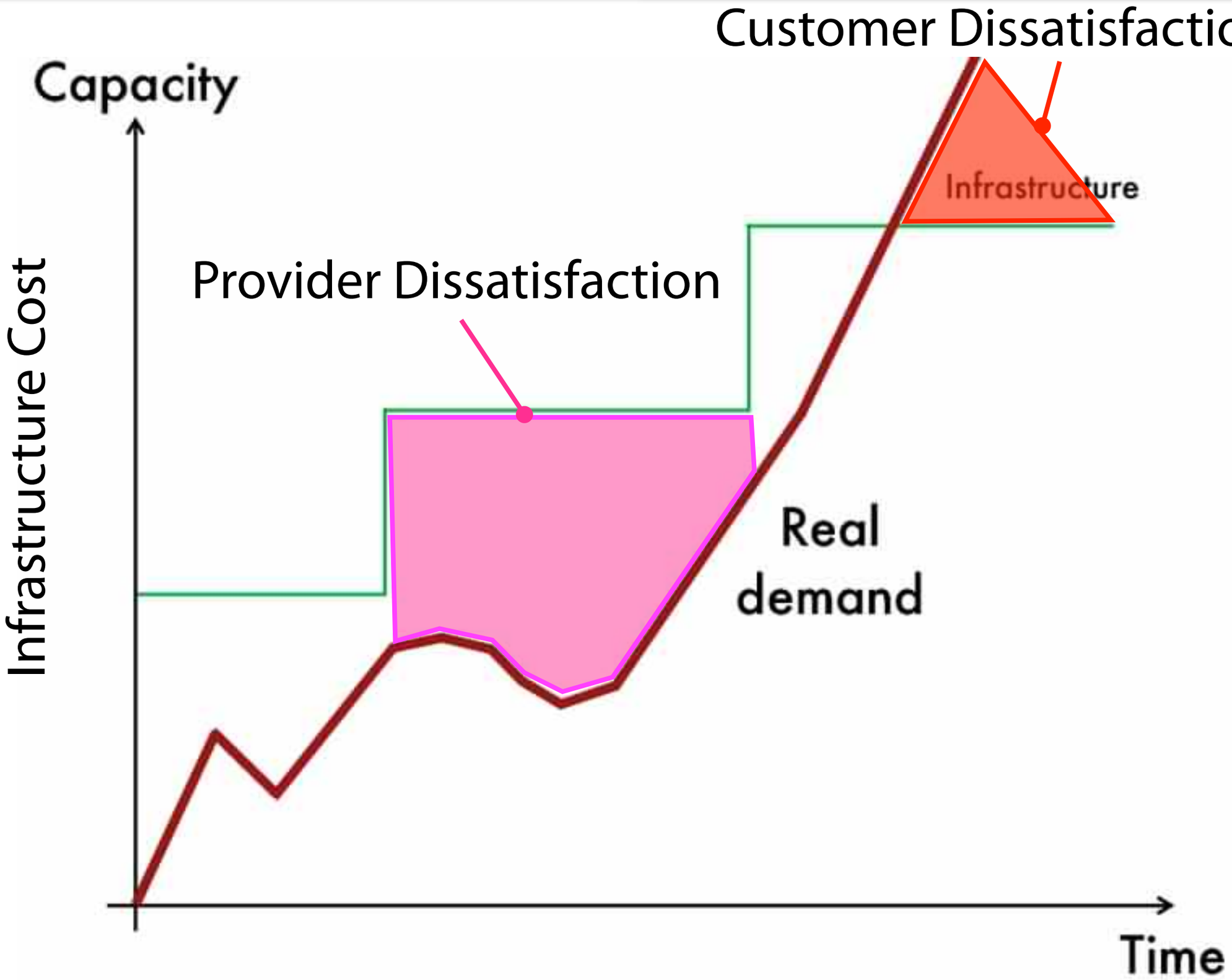
Elasticity (II)



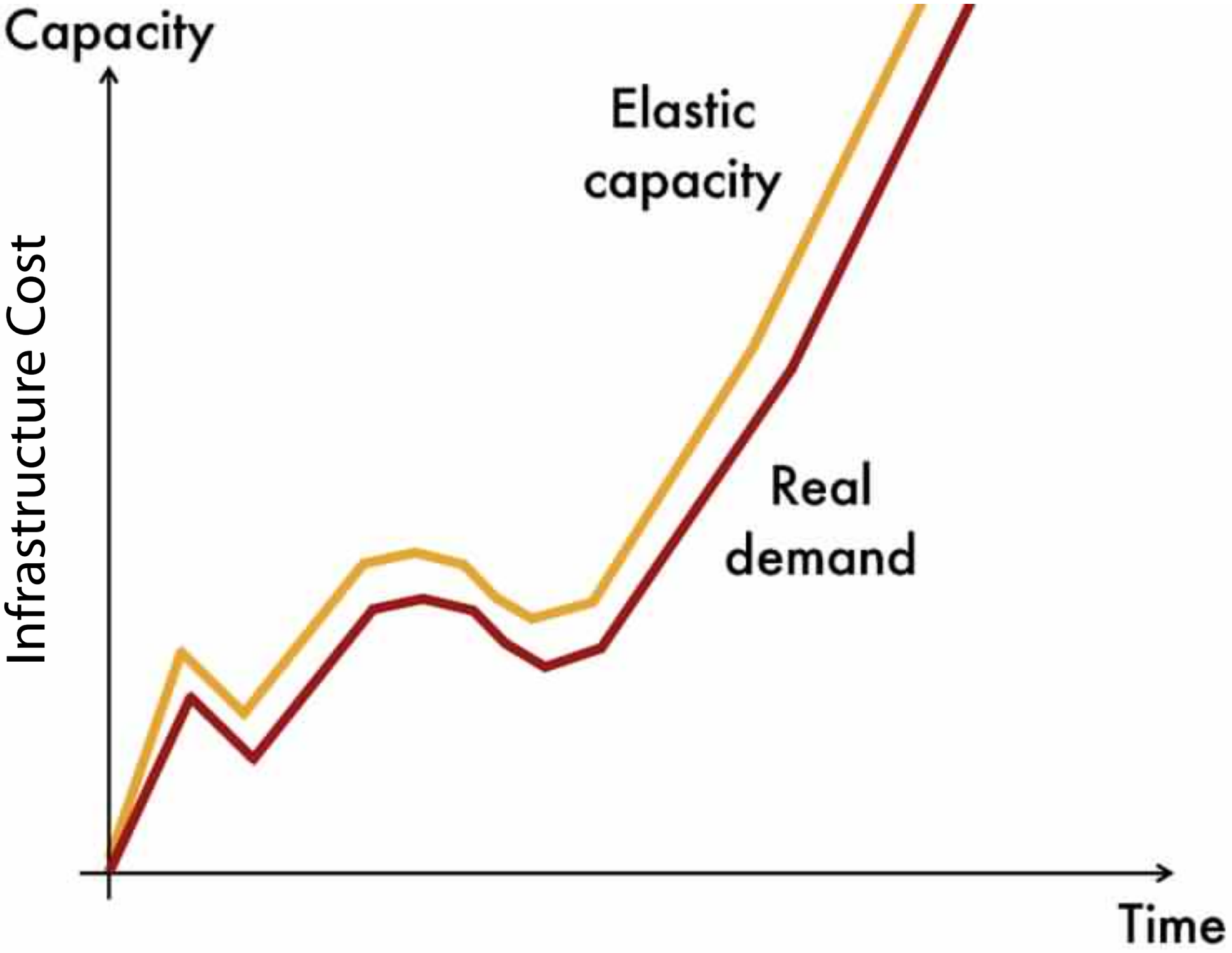
Elasticity (III)



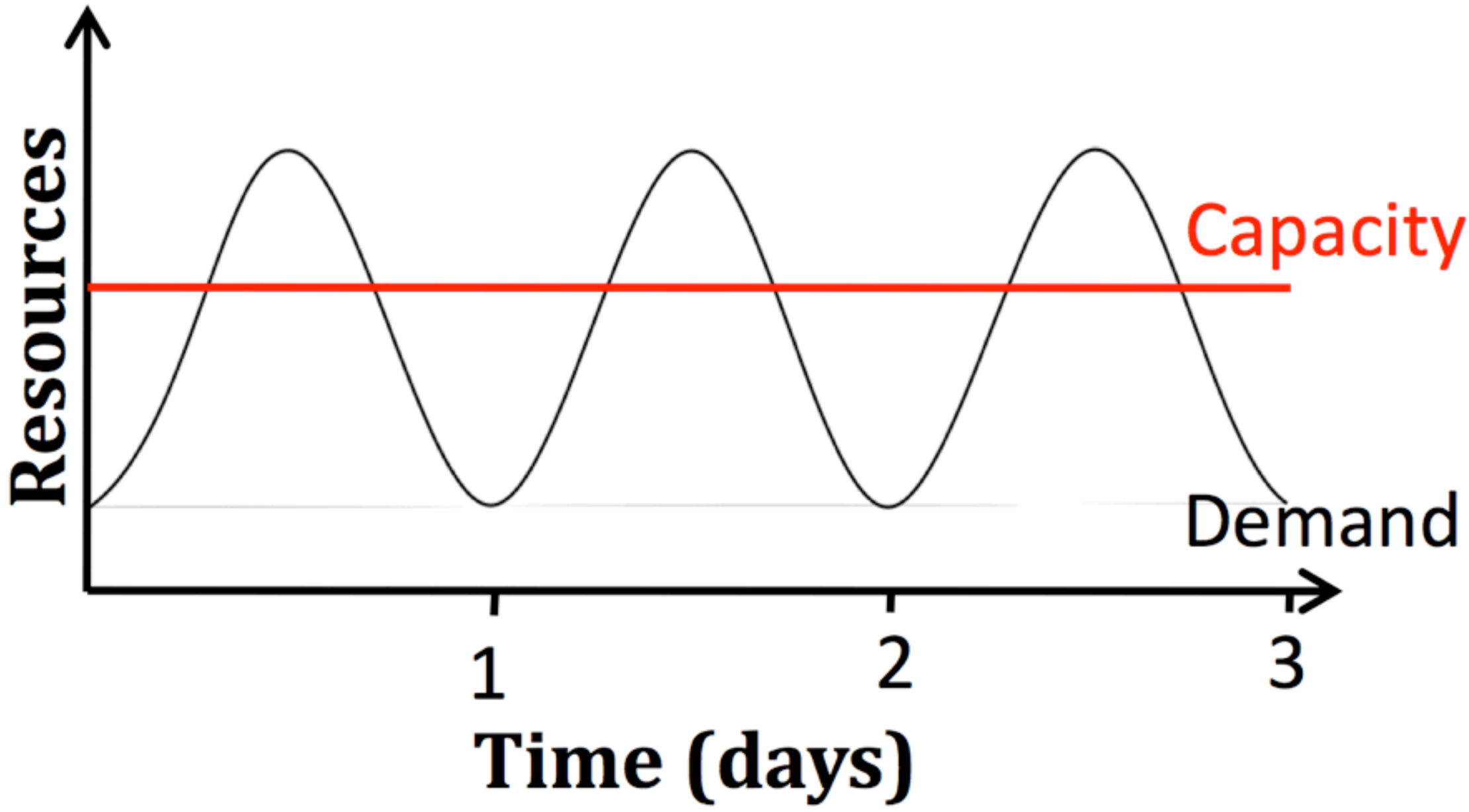
Elasticity (III)



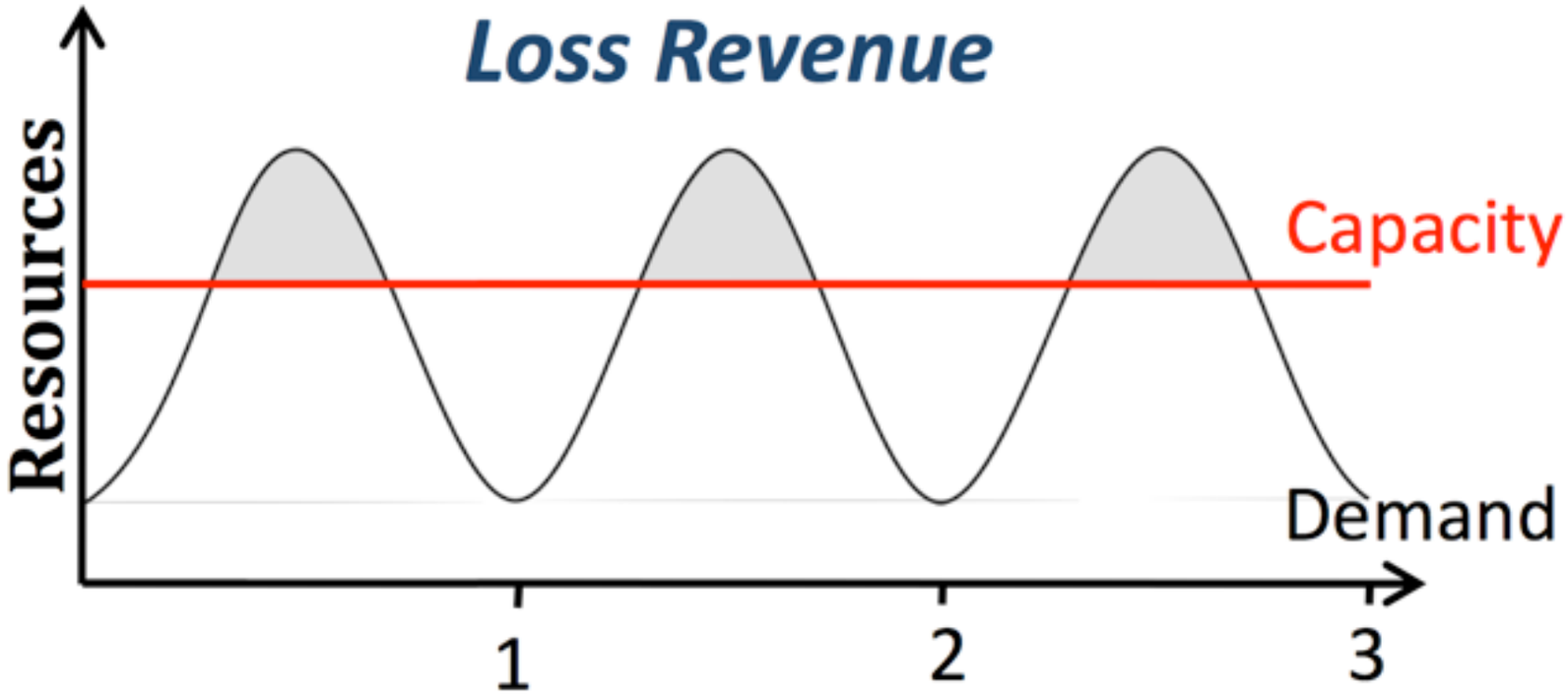
Elasticity (IV)

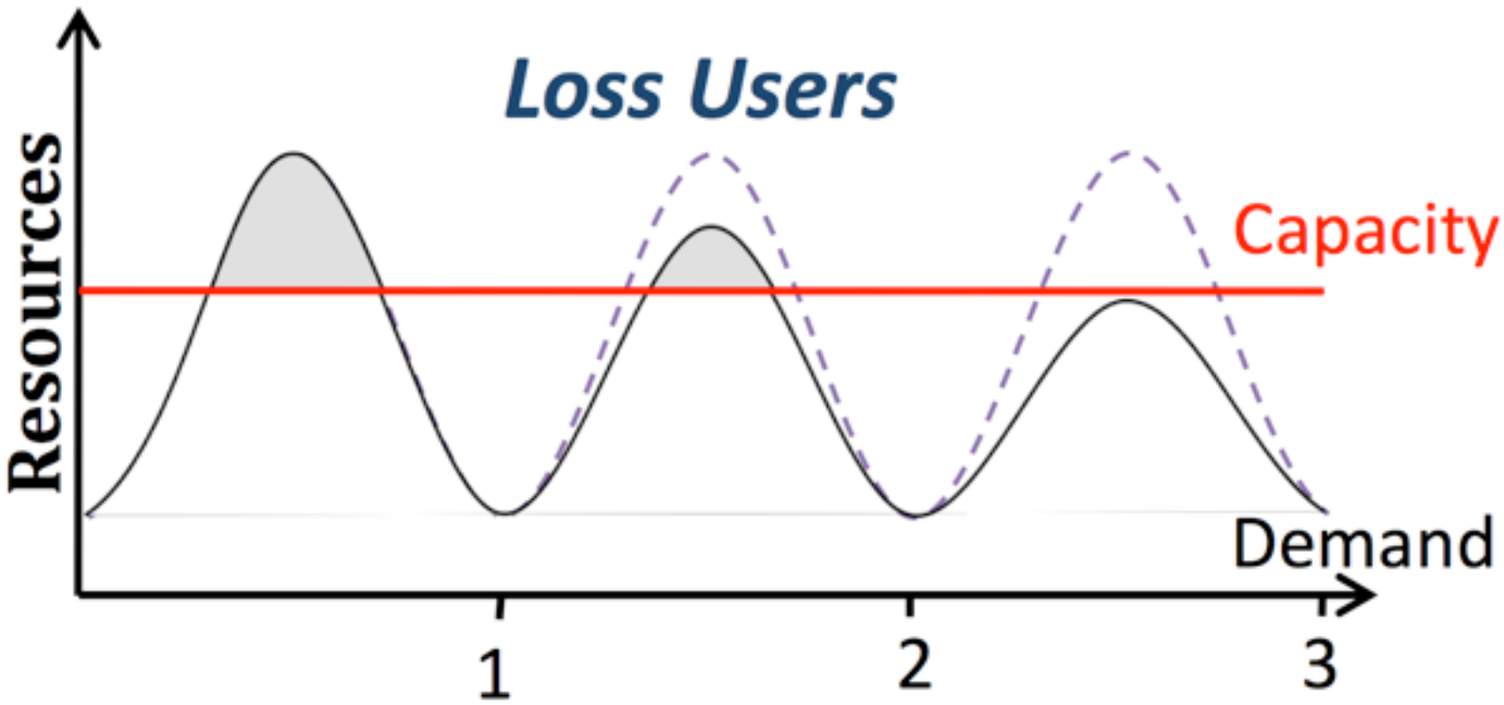


Traditional Provisioning

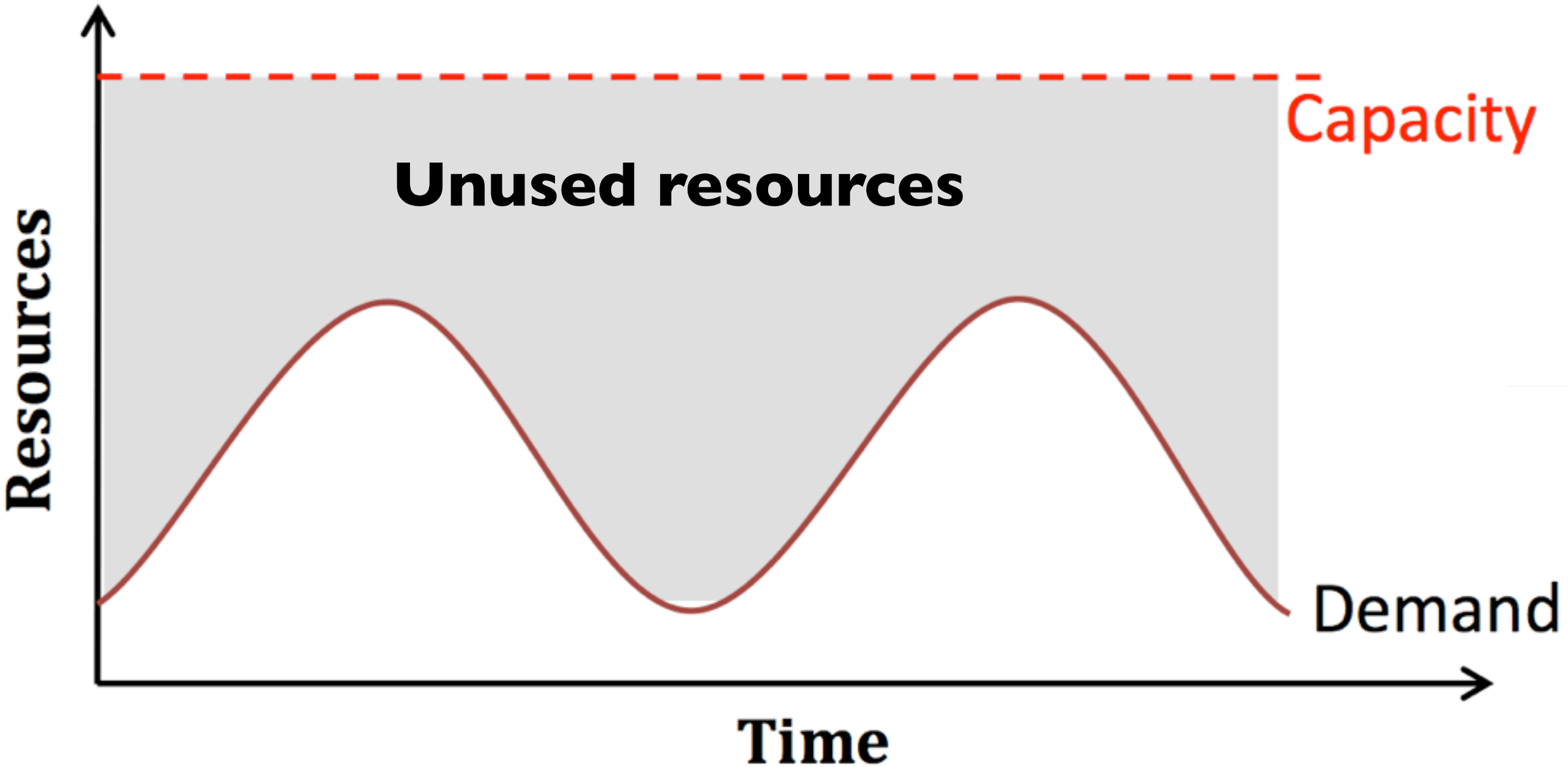


Under-provisioning



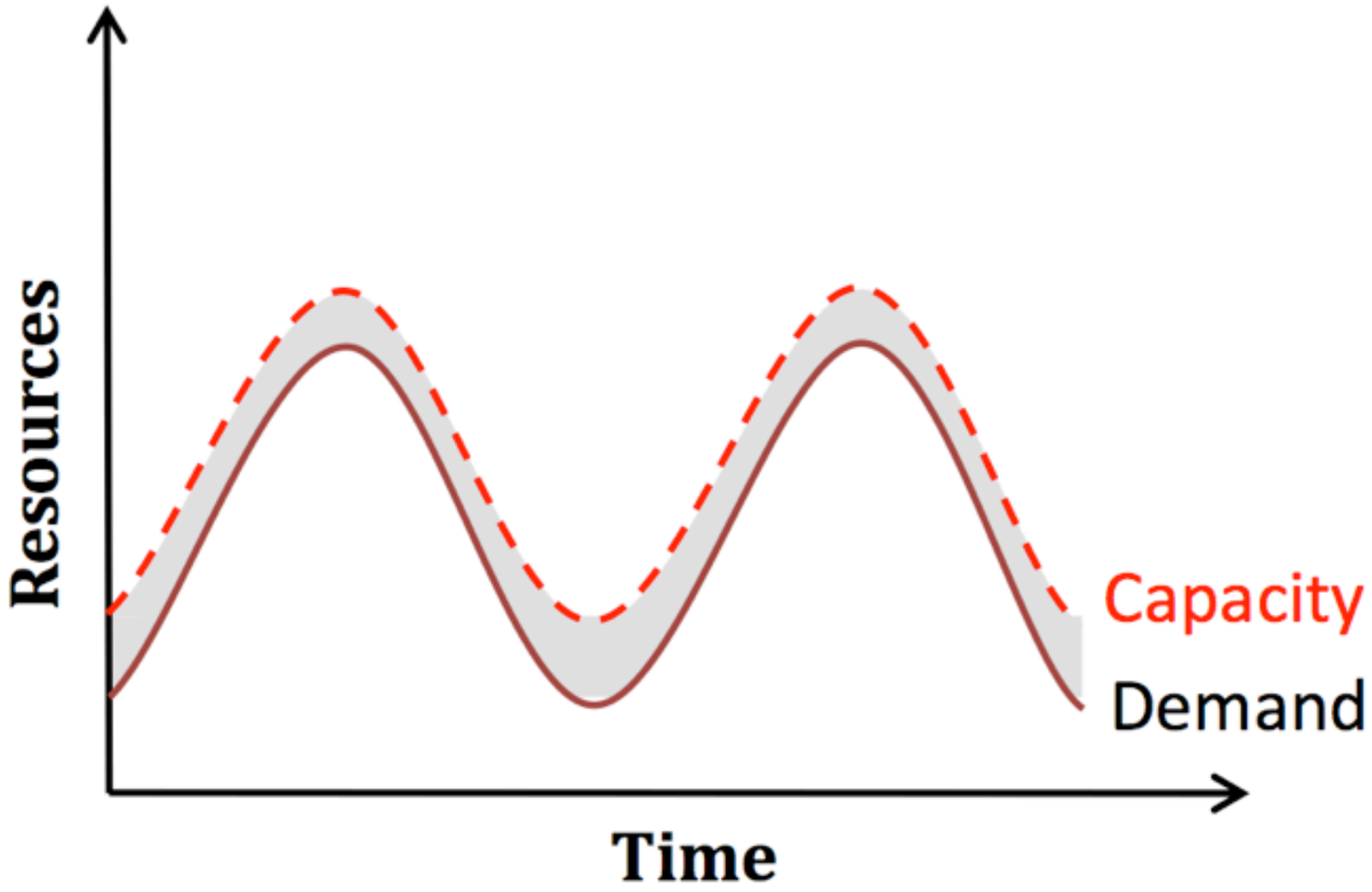


Over-provisioning



Dynamic Provisioning

- Cloud resources should be provisioned dynamically
  - Meet seasonal demand variations
  - Meet demand variations between different industries
  - Meet burst demand for some extraordinary events



Multi-tenant Design

- Multi-tenant refers to a principle in software architecture where a single instance of the software runs on a server, serving multiple client organizations.
  - Multi-tenancy is contrasted with a multi-instance architecture where separate software instances (or hardware systems) are set up for different client organizations
  - With a multi-tenant architecture, a software application is designed to virtually partition its data and configuration thus each client organization works with a customized virtual application instance.
- Client requirements:
  - Multi-tenant applications are typically required to provide a high degree of customization to support each target organization's needs.
  - Multi-tenant applications are expected to provide adequate levels of security and robustness.

Availability and Reliability

- Availability
  - The degree to which a system, subsystem, or equipment is in a specified operable and committable state at the start of a mission, when the mission is called for at an unknown time.
- Reliability
  - The ability of a system or component to perform its required functions under stated conditions for a specified period of time.

- How to achieve?
  - Fault tolerance
  - System resilience
  - Security

Fault Tolerance

- Fault-tolerance is the property that enables a system to continue operating properly in the event of the failure of some of its components.
- If its operating quality decreases at all, the decrease is proportional to the severity of the failure, as compared to a naively-designed system in which even a small failure can cause total breakdown.

- Four basic characteristics :
  - No single point of failure
  - Fault detection and isolation to the failing component
  - Fault containment to prevent propagation of the failure
  - Availability of reversion modes

Single Point of Failure

- A part of a system which, if it fails, will stop the entire system from working.

- The assessment of a potentially single location of failure identifies the critical components of a complex system that would provoke a total systems failure in case of malfunction.

- Countermeasure: preventing single point of failure
  - If a system experiences a failure, it must continue to operate without interruption during the repair process.

Fault Detection and Isolation

- A subfield of control engineering which concerns itself with monitoring a system, identifying when a fault has occurred and pinpoint the type of fault and its location.

- Countermeasure: isolate failing component
  - When a failure occurs, the system must be able to isolate the failure to the offending component.

Fault Containment

- Some failure mechanisms can cause a system to fail by propagating the failure to the rest of the system.

- Mechanisms that isolate a rogue transmitter or failing component to protect the system are required.

- Countermeasure: availability of reversion modes
  - System should be able to maintain some check points which can be used in managing the state changes.

System Resiliency

- Resilience is the ability to provide and maintain an acceptable level of service in the face of faults and challenges to normal operation.

- Resiliency pertains to the system's ability to return to its original state after encountering trouble. In other words, if a risk event knocks a system offline, a highly resilient system will return back to work and function as planned as soon as possible.

- Disaster recovery is the process, policies and procedures related to preparing for recovery or continuation of technology infrastructure critical to an organization after a natural or human-induced disaster.

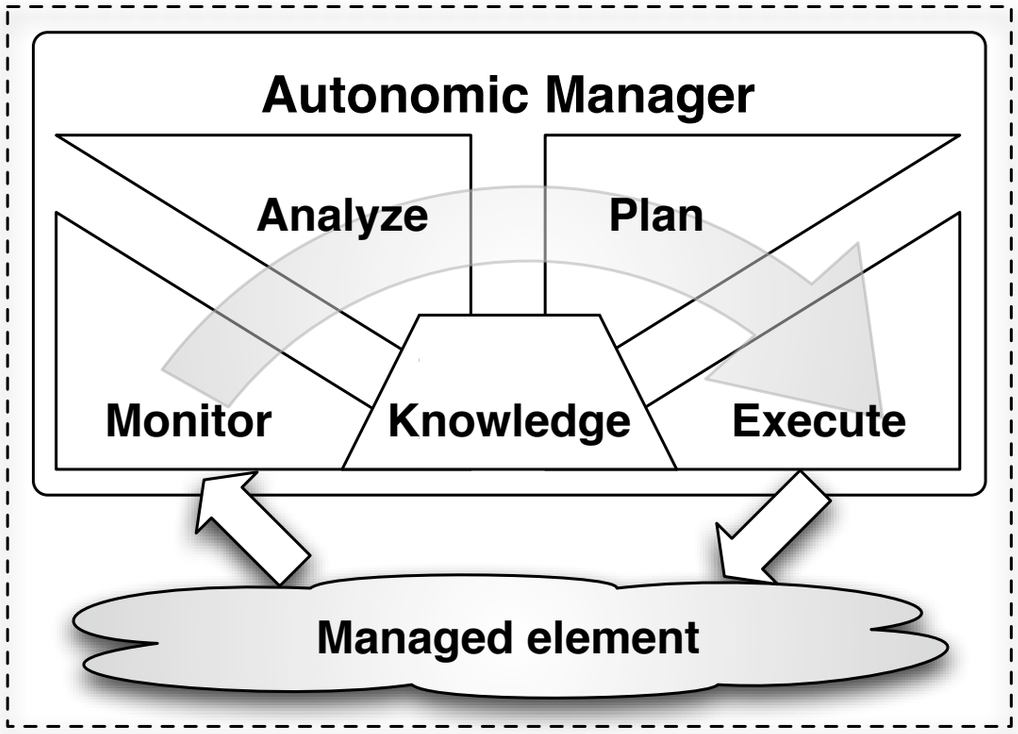
Security

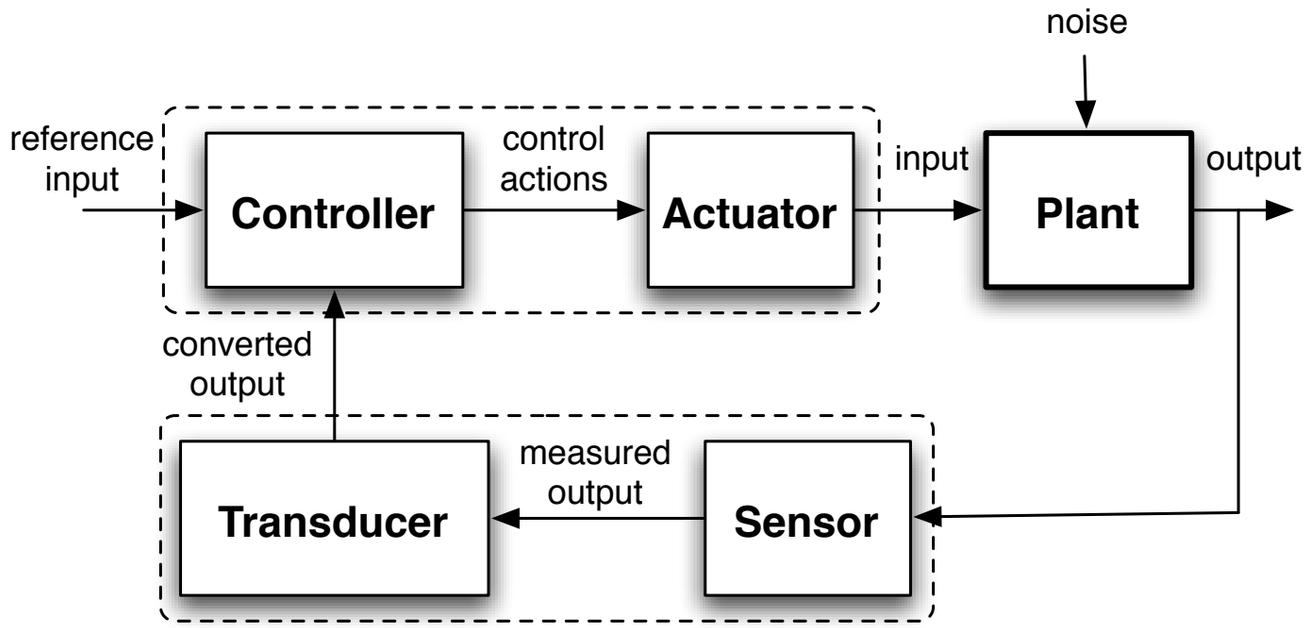
- Cloud security is an evolving sub-domain of computer security, network security, and, more broadly, information security.
- It refers to a broad set of policies, technologies, and controls deployed to protect data, applications, and the associated infrastructure of cloud computing.
- Important security and privacy issues:
  - Data Protection
    - To be considered protected, data from one customer must be properly segregated from that of another.
  - Identity Management
    - Every enterprise will have its own identity management system to control access to information and computing resources.
  - Application Security
    - Cloud providers should ensure that applications available as a service via the cloud are secure.
  - Privacy
    - Providers ensure that all critical data are masked and that only authorized users have access to data in its entirety.

Manageability and Interoperability

- Manageability
  - Enterprise-wide administration of cloud computing systems.
  - Systems manageability is strongly influenced by network management initiatives in telecommunications.
- Interoperability
  - Interoperability is a property of a product or system, whose interfaces are completely understood, to work with other products or systems, present or future, without any restricted access or implementation.
- How to achieve?
  - Autonomic Control
  - System Monitoring
  - Billing System

Autonomic Control





- Autonomic Computing
  - Its ultimate aim is to develop computer systems capable of self-management, to overcome the rapidly growing complexity of computing systems management, and to reduce the barrier that complexity poses to further growth.

Self-* Properties

- Autonomic System Functional Areas (CHOP)
  - Self-Configuration
    - Automatic configuration of components.
  - Self-Healing
    - Automatic discovery, and correction of faults.
  - Self-Optimization
    - Automatic monitoring and control of resources to ensure the optimal functioning with respect to the defined requirements.
  - Self-Protection
    - Proactive identification and protection from arbitrary attacks.

System Monitoring

- A System Monitor in systems engineering is a process within a distributed system for collecting and storing state data.

- What should be monitored in the Cloud ?
  - Physical and virtual hardware state
  - Resource performance metrics
  - Network access patterns
  - System logs
  - etc...

Billing System

- Billing System in Cloud
  - Users pay as many as they used.
  - Cloud provider must first determine the list of service usage price.
  - Cloud provider have to record the resource or service usage of each user, and then charge users by these records.

- How can cloud provider know users' usage ?
  - Get those information by means of monitoring system.
  - Automatically calculate the total amount of money which user should pay.
  - Automatically request money from use's banking account.

Performance and Optimization

- Performance
  - Is characterized by the amount of useful work accomplished by a computer system compared to the time and resources used.
  - Depending on the context, good computer performance may involve one or more of the following:
    - Short response time for a given piece of work
    - High throughput (rate of processing work)
    - Low utilization of computing resource(s)
    - High availability of the computing system or application
    - Fast (or highly compact) data compression and decompression
    - High bandwidth / short data transmission time
- Optimization
  - Improvement of performance
- How to achieve?
  - Parallel processing
  - Load balancing
  - Job scheduling

Parallel Processing

Parallel processing is a form of computation in which many calculations are carried out simultaneously, operating on the principle that large problems can often be divided into smaller ones, which are then solved concurrently.

- Parallelism in different levels:
  - Bit level parallelism
  - Instruction level parallelism
  - Data level parallelism
  - Task level parallelism
- Hardware approaches
  - Multi-core computer
  - Symmetric multi-processor
  - General purpose graphic processing unit
  - Vector processor
  - Distributed computing
    - Cluster computing
    - Grid computing
- Software approaches
  - Parallel programming language
  - Automatic parallelization

Load Balancing

Load balancing is a technique to distribute workload evenly across two or more computers, network links, CPUs, hard drives, or other resources, in order to get optimal resource utilization, maximize throughput, minimize response time, and avoid overload.

- Why should be load balanced ?
  - Improve resource utilization
  - Improve system performance
  - Improve energy efficiency

Job Scheduling

- A job scheduler is a software application that is in charge of unattended background executions, commonly known for historical reasons as batch processing.

- Synonyms are batch system, Distributed Resource Management System (DRMS), and Distributed Resource Manager (DRM).

- Various schemes are used to decide which particular job to run. Parameters that might be considered include:
  - Job priority
  - Compute resource availability
  - License key if job is using licensed software
  - Execution time allocated to user
  - Number of simultaneous jobs allowed for a user
  - Estimated execution time
  - Elapsed execution time
  - Availability of peripheral devices
  - Occurrence of prescribed events

Accessibility and Portability

- Accessibility
  - Accessibility is a general term used to describe the degree to which a product, device, service, or environment is accessible by as many people as possible.
- Portability
  - Service portability is the ability to access services using any devices, anywhere, continuously with mobility support and dynamic adaptation to resource variations.
- How to achieve?
  - Uniform access
  - Thin client

Uniform access and Thin client

- How do users access cloud services?
  - Cloud providers should provide their cloud services by means of widespread accessing media. In other word, users from different operating systems or other accessing platforms should be able to directly be served.
  - Nowadays, web browser technique is one of the most widespread platform in almost any intelligent electronic devices. Cloud services take this into concern, and delivery their services with web-based interface through the Internet.
- What is thin client?
  - Thin client is a computer or a computer program which depends heavily on some other computer to fulfill its traditional computational roles. This stands in contrast to the traditional fat client, a computer designed to take on these roles by itself.
- Characteristics :
  - Cheap client hardware
    - While the cloud providers handle several client sessions at once, the clients can be made out of much cheaper hardware.
  - Diversity of end devices
    - End user can access cloud service via plenty of various electronic devices, which include mobile phones and smart TV.
  - Client simplicity
    - Client local system do not need complete operational functionalities.

Cloud Benefits

Markets & Enterprises

Users

Benefits from Clouds

- For the market and enterprises
  - Reduce initial investment
  - Reduce capital expenditure
  - Improve industrial specialization
  - Improve resource utilization

- For the end user and individuals
  - Reduce local computing power
  - Reduce local storage power
  - Variety of thin client devices in daily life

Initial Investment

- Traditional process of enterprises to initiate business:
  - Survey and analysis the industry and market
  - Estimate the quantity of supply and demand
  - Purchase and deploy IT infrastructure
  - Install and test the software system
  - Design and develop enterprise specific business service
  - Announce the business service to clients

- Some drawbacks :
  - The survey, analysis and estimation may not 100% correct
  - Infrastructure deployment is time consuming
  - Enterprises should take the risk of wrong investment

Initial Investment

- Initiate business with Cloud Computing services:
  - Survey and analysis the industry and market
  - Choose one cloud provider for enterprise deployment
  - Design and develop business service upon cloud environment
  - Announce the business service to clients

- Some benefits:
  - Enterprise do not need to own the infrastructure
  - Enterprise can develop and deploy business service in short time
  - Enterprise can reduce the business loss of wrong investment

Reduce Capital Expenditure

- Traditional capital expenditure of enterprises:
  - Each enterprise should establish its own IT department
  - IT department should handle the listing jobs
    - Manage and administrate hardware and software
    - Apply regular data backup and check point process
    - Purchase new infrastructure and eliminate outdated one
    - Always standby for any unexpected IT problems

- Some drawbacks :
  - Enterprise pays for IT investment which is not its business focus
  - Enterprise should take the risk of hardware/software malfunction
  - Replacing and updating infrastructure is time consuming and risky

Reduce Capital Expenditure

- Capital expenditure with Cloud Computing service:
  - Enterprise can almost dismiss its IT department
  - The jobs of IT department can be achieved by cloud provider
    - Dynamically update and upgrade hardware or software
    - Dynamically provision and deploy infrastructure for enterprise
    - Automatically backup data and check consistency
    - Self-recover from disaster or system malfunction

- Some benefits :
  - Enterprise can shift effort to its business focus
  - Enterprise can reconfigure its IT services in short time
  - Enterprise pays to cloud provider as many as the service used

Improve Industrial Specialization

- Traditional industry and market:
  - Each enterprise has to own its IT department
  - IT resources are managed directly by the enterprises
  - IT complexity should be addressed and managed with care by the enterprise itself

- Some drawbacks :
  - IT department is not the business focus of enterprises
  - Most enterprise do not maintain correctly their IT resources
  - Enterprises must optimize their IT resources usage

Improve Industrial Specialization

- Outsourcing to Cloud providers
  - Cloud providers centrally maintain IT infrastructure for enterprises
  - Cloud providers are business-focused on providing reliable IT resources
  - Cloud providers employ experts for management and administration
  - Enterprises only rent and pay the services they ne

- Some benefits :
  - IT service performance is optimized by experts
  - Enterprises can focus on their business
  - IT resource waste is reduced

Improve Resource Utilization

- Traditional resource utilization
  - Enterprises rarely take care about IT resource utilization
  - IT resources are not well managed by the enterprises
  - IT resources usually for peak demand

- Some drawbacks :
  - Power and spaces utilization wasted
  - IT resources can not be shared across enterprises

Improve Resource Utilization

- Outsourcing to Cloud providers
  - Cloud providers centrally maintain IT infrastructure for enterprises
  - Cloud providers build performance optimized hardware
  - Cloud providers build consolidated cooling systems
  - Cloud providers take care of legal policy issues

- Some benefits :
  - IT resources can be shared among enterprises
  - IT infrastructure performance can be optimized
  - Large-scale integrated optimization can be applie

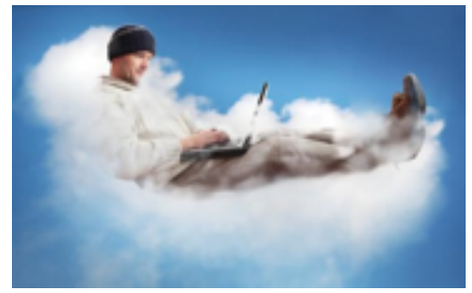
Reduce Local Computing Power

- Traditional local computing power requirements:
  - Need to buy your own personal computer
  - Buy powerful processor if you need intensive computing
  - Buy a lot of memory to meet application requirements
  - Install plenty of applications
  - Manage (security) upgrades

- Some drawbacks :
  - Hard to replicate the same system environment
  - Need to regularly update or upgrade software and hardware
  - Need to reinstall application if you reinstall the OS

Reduce Local Computing Power

- Using Cloud Computing services:
  - Can utilize remote computing power in the Cloud
  - Need a basic computer to connect to Internet
  - Applications automatically managed



- Some benefits:
  - Access personal computer anywhere through network
  - Dynamically request more resources on demand
  - Application must not be manually upgraded/managed/reinstalled

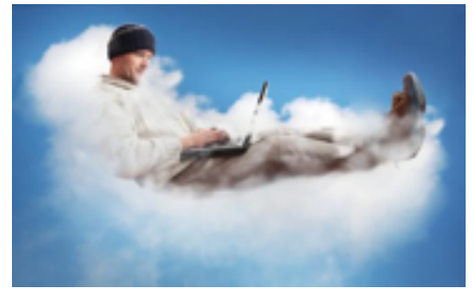
Reduce Local Storage Power

- Traditional local storage power requirement:
  - User code and data files stored in local devices
  - Manual backup regularly preventing hardware failu
  - Physical / power / heating requirements

- Some drawbacks :
  - Storage space may not be enough
  - Storage space may be too much
  - Data consistency between computers is hard
  - Need to sacrifice storage for backups

Reduce Local Storage Power

- Using Cloud Computing services:
  - User code and data files stored in the Cloud
  - Cloud provider guarantees the data availability



- Some benefits:
  - Dynamic allocation of storage on demand
  - Seamless access to data through the network
  - No need to care about data consistency
  - No need to care about data losses and backups
  - No need to care about space/power

Variety of End Devices

- Traditional computing resources:
  - Connection to the Internet through personal computers
  - Only PCs can deliver reasonable computing power
  - Small devices have hardware and power limitations

- Some drawbacks :
  - Computing power is not portable
  - Small devices can only perform small works

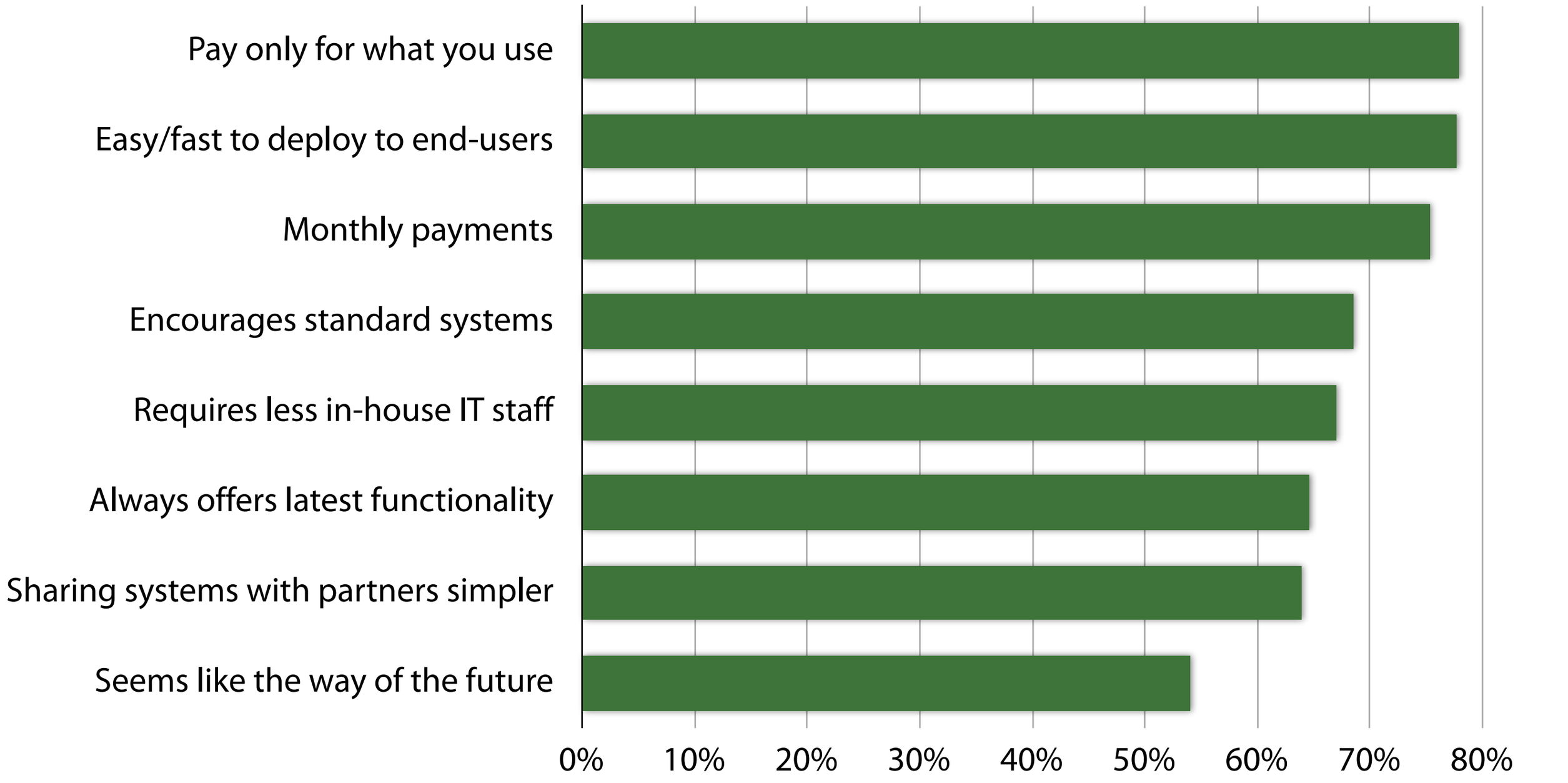
Variety of End Devices

- Devices integrated with Cloud Computing:
  - Devices connect to the Internet through wireless networks
  - Devices access Cloud services through Web interfaces
  - Devices outsource computing jobs to the cloud

- Some benefits:
  - Users can easily access Cloud services through small devices
  - User can access almost unlimited computing power
  - Small devices can be managed through Clouds (install/upgrade apps)

Cloud Benefits

Question: Rate the benefits commonly ascribed to the cloud model

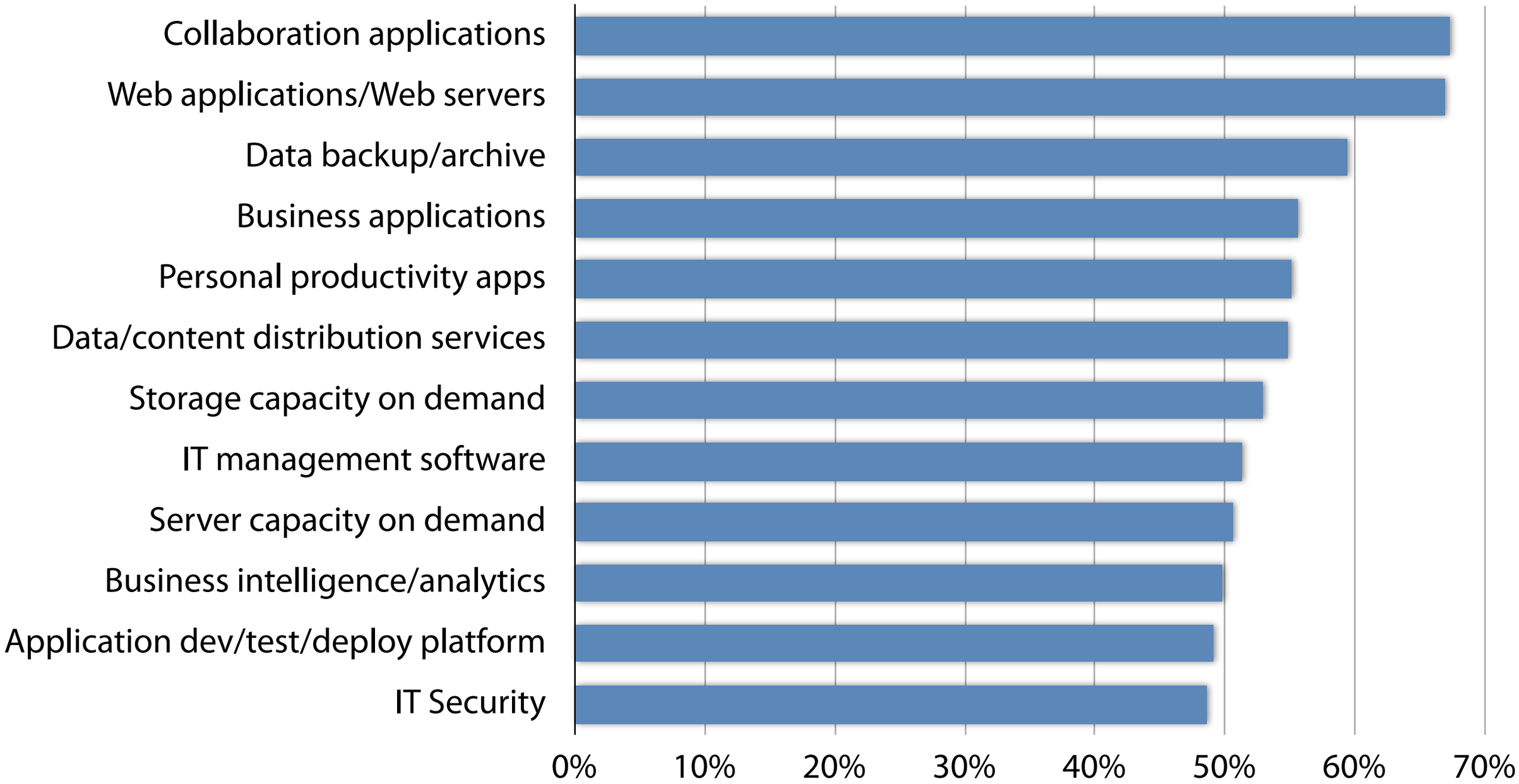


Scale: 1 = not at all concerned, 5 = very concerned, % responding 3,4,5 (263 interviews)

Source: IDC's QuickLook Survey, IDC's Enterprise Panel 3Q09

Cloud Adoptions

Question: Rate your likelihood to pursue the cloud model for the following

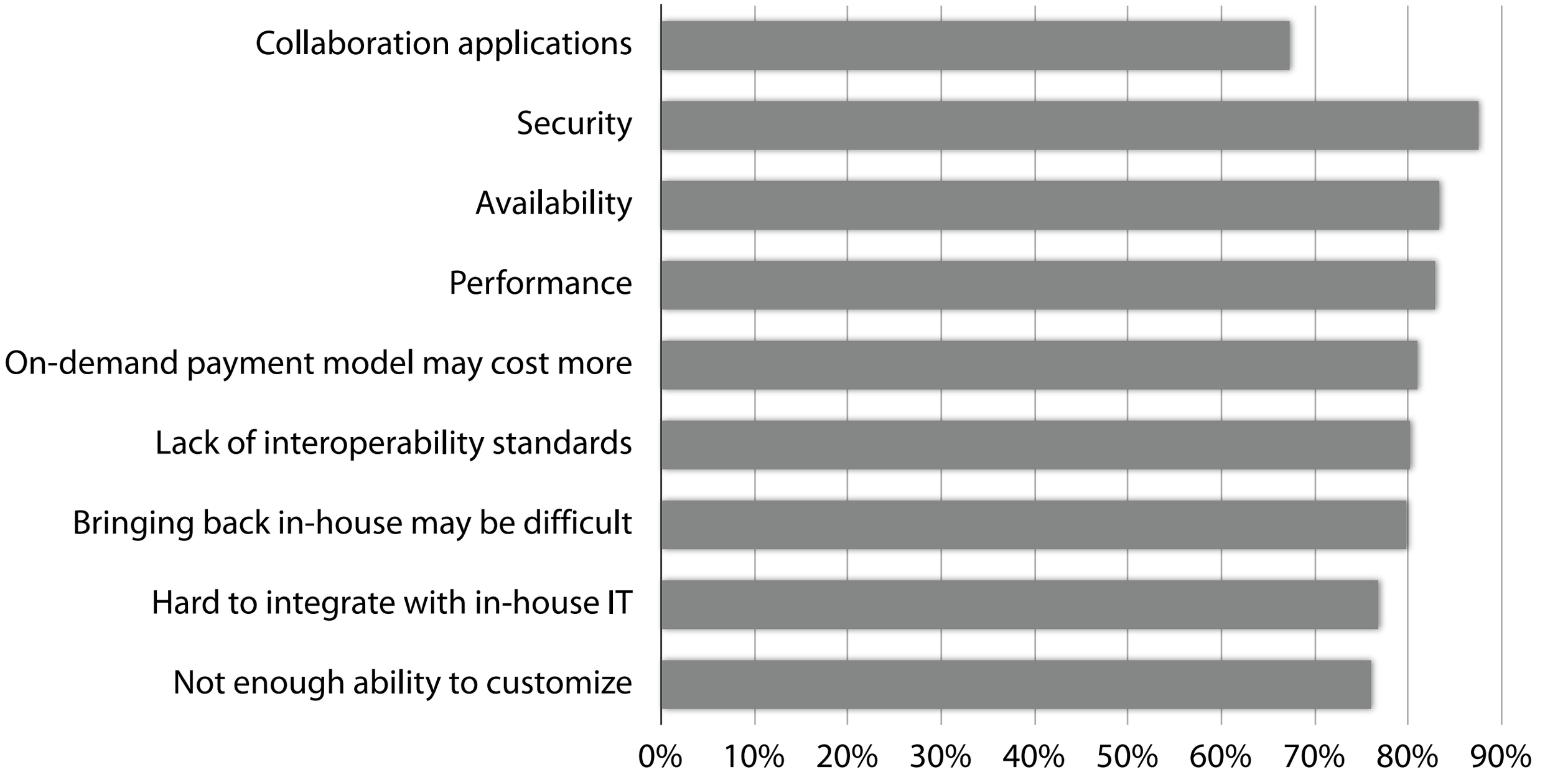


Scale: 1 = not at all concerned, 5 = very concerned, % responding 3,4,5 (263 interviews)

Source: IDC's QuickLook Survey, IDC's Enterprise Panel 3Q09

Cloud Challenges

Question: Rate the challenges/issues of the cloud model



Scale: 1 = not at all concerned, 5 = very concerned, % responding 3,4,5 (263 interviews)

Source: IDC's QuickLook Survey, IDC's Enterprise Panel 3Q09

Service Models for Housing

- You look for a place to live in a new city

- What do you do?

- Build a new house?
  - You can fully control everything you like to have in your house
  - It is a hard work...

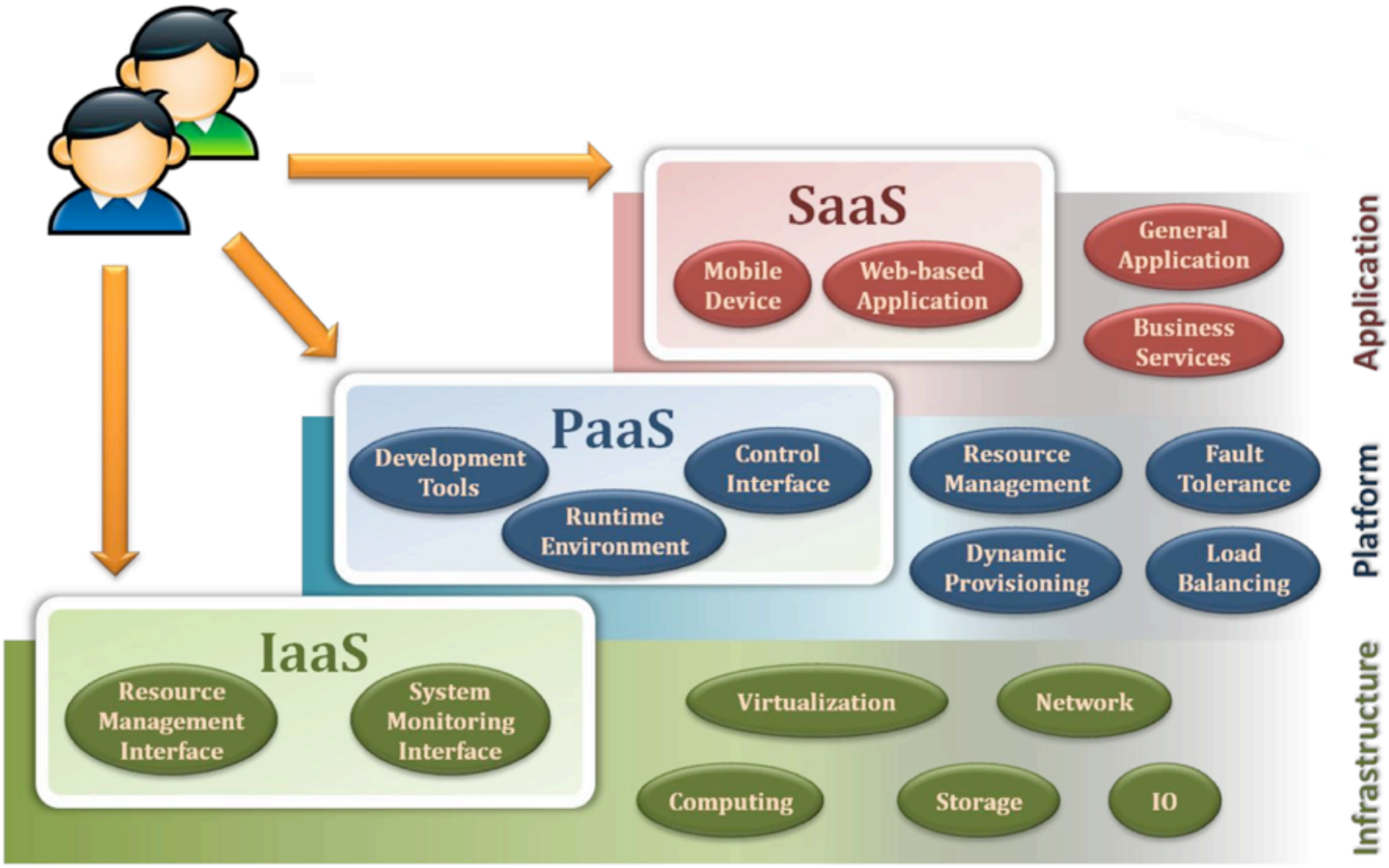
- Buy an empty house?
  - You can fully control only some parts of your house
  - Can not change original infrastructure

- Live in a hotel?
  - Great if you only want to enjoy your life
  - House just for living

Service Models for Clouds

- You want an IT department!
- You can rent some infrastructure and build up your IT system with these resources, under your full control
  - Build a new house
  - Technically, use **Infrastructure as a Service** (IaaS)
- You develop your IT applications through one Cloud platform, not caring about low level resource management
  - Buy an empty house
  - Technically, use **Platform as a Service** (PaaS)
- You directly use some existing IT solutions, provided by the Cloud and ignoring any further deteal
  - Live in a hotel
  - Technically, use **Software as a Service** (SaaS)

Service Models Summary



Infrastructure as a Service (I)

- 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 .

- Examples: Amazon EC2, Eucalyptus, OpenNebula

Infrastructure as a Service (II)

- Enabling technology: **virtualization**
  - An abstraction of logical resources away from underlying physical resources.
- Supported **properties**:
  - Manageability and interoperability
  - Availability and reliability
  - Scalability and elasticity
- Provided service:
  - **Virtual Machine** – As an IaaS provider, we should be able to provide the basic virtual machine operations, such as creation, suspension, resumption and termination, ... and be able to monitor some system states of each virtual machine, such as CPU loading, memory utilization, IO loading and internal network loading, ...
  - **Virtual Storage** – As an IaaS provider, we should be able to provide the basic virtual storage operations, such as space allocation, space release, data writing and data reading, ... and be able to monitor some storage states of each virtual storage, such as virtual space utilization, data duplication and storage device access bandwidth, ...
  - **Virtual Network** – As an IaaS provider, we should be able to provide the basic virtual network operations, such as IP address allocation, domain name register, connection establishment and bandwidth provision, ... and be able to monitor some network states of each virtual network, such as virtual network bandwidth, network connectivity and network load balancing, ...

Platform as a Service (I)

- 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.

- Examples: Microsoft Windows Azure, Google App Engine, Hadoop, Amazon Elastic MapReduce

Platform as a Service (II)

- Enabling technology: **runtime environment**
  - Refers to collection of software services available. Usually implemented by a collection of program libraries.
- Supported **properties**:
  - Manageability and interoperability
  - Performance and optimization
  - Availability and reliability
  - Scalability and elasticity
- Provided service:
  - **Programming IDE** – Users make use of programming IDE to develop their services among PaaS. This IDE should integrate the full functionalities which supported from the underling runtime environment. This IDE should also provide some development tools, such as profiler, debugger and testing environment.
  - **System Control Interface**
    - Police-Based Control
      - Typically described as a principle or rule to guide decisions and achieve rational outcome(s)
      - Make the decision according to some requirements
    - Workflow Control
      - Describe the flow of installation and configuration of resources
      - Workflow processing daemon delivers speedy and efficient construction and management of cloud resources

Software as a Service (I)

- 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).

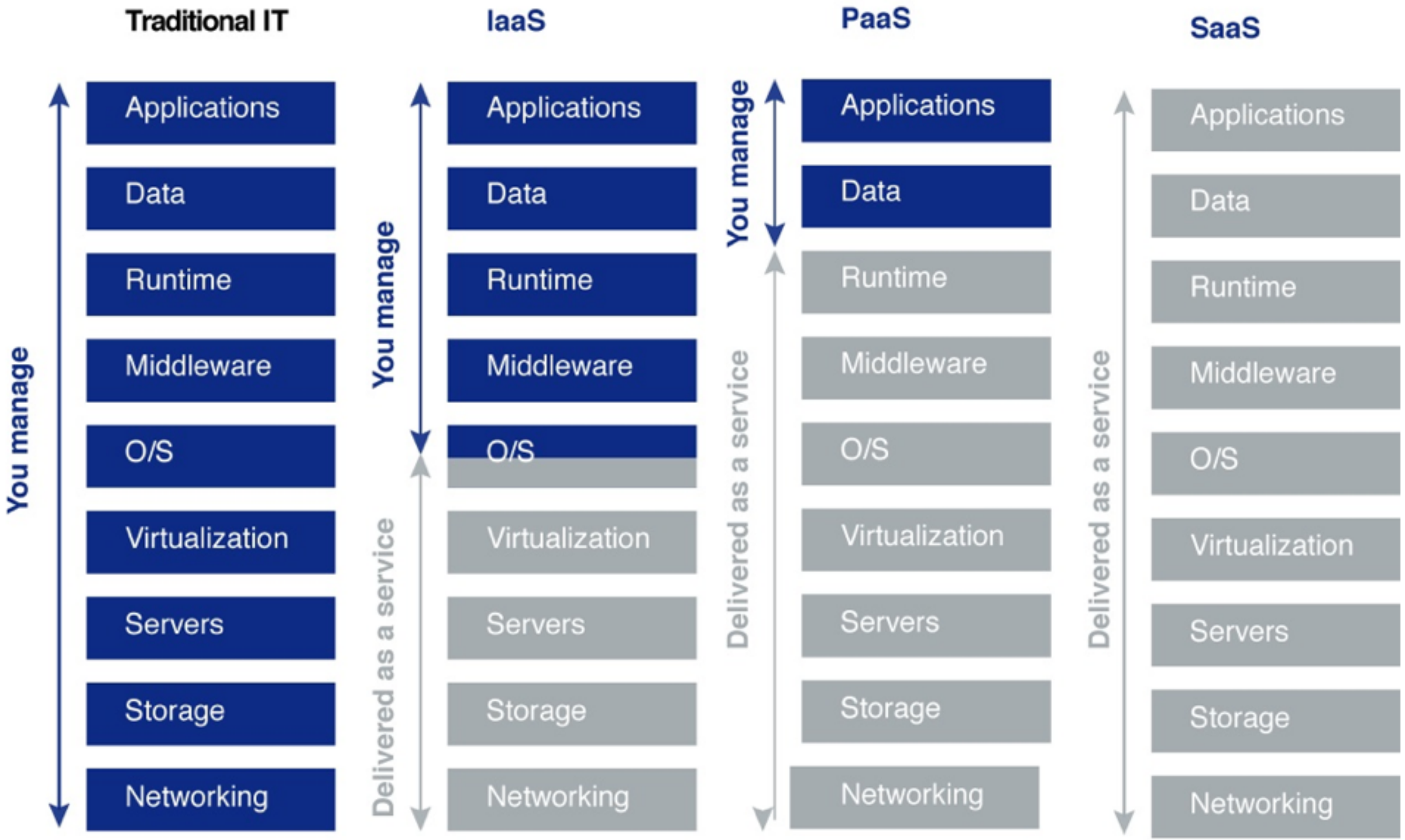
- The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings.

- Examples: Gmail, DropBox, Google, iCloud

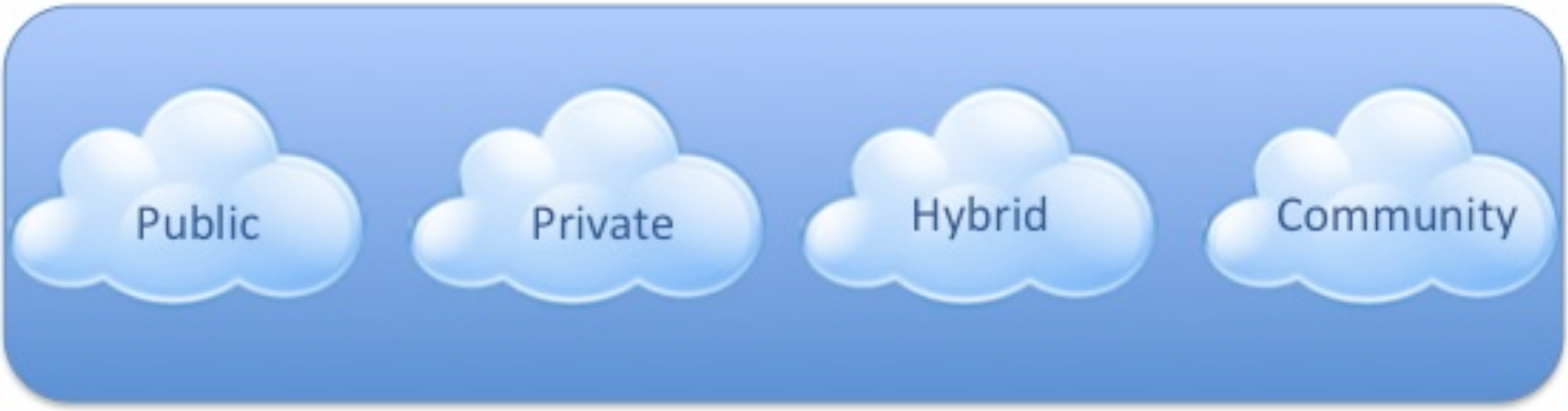
Software as a Service (II)

- Enabling technology: **web service**
  - Refers to a method of communication between two electronic devices over the World Wide Web.
- Supported **properties**:
  - Accessibility and portability
- Provided service:
  - **Web-based Applications**
    - General Applications – Applications which are designed for general propose, such as office suit, multimedia and instant message, ...
    - Business Applications – Application which are designed for business propose, such as ERP, CRM and market trading system, ...
    - Scientific Applications – Application which are designed for scientific propose, such as aerospace simulation and biochemistry simulation, ...
    - Government Applications – Applications which are designed for government propose, such as national medical system and public transportation system service, ...
  - **Web Portal**
    - Apart from the standard search engine feature, web portals offer other services such as e-mail, news, stock prices, information, databases and entertainment.
    - Portals provide a way for enterprises to provide a consistent look and feel with access control and procedures for multiple applications and databases, which otherwise would have been different entities altogether.

Service Models Summary

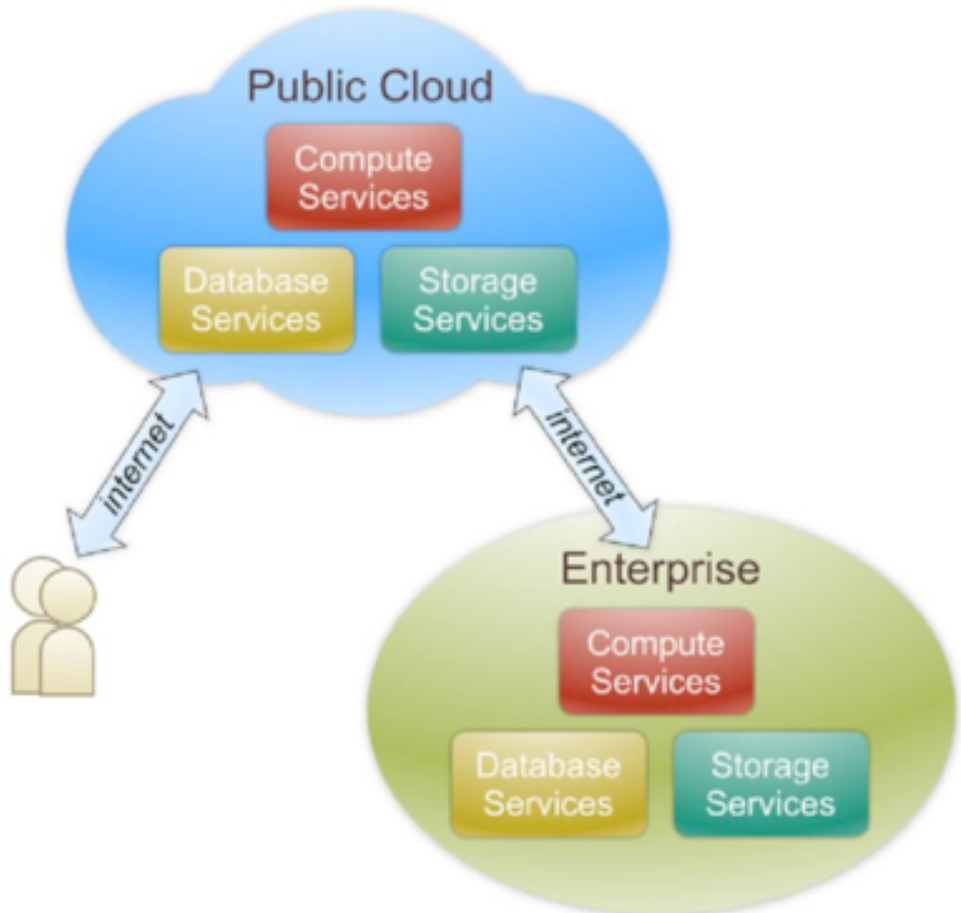


Deployment Models



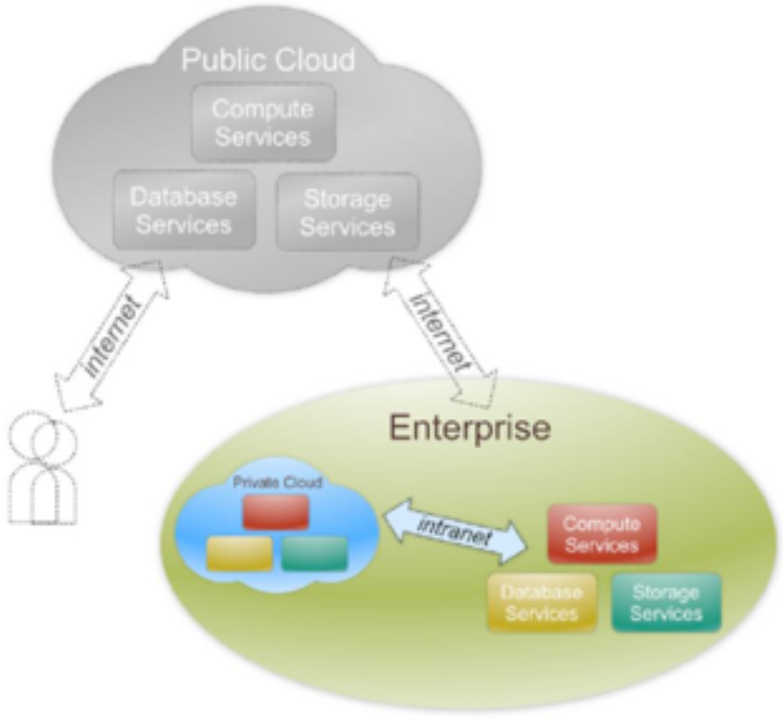
Public Cloud

- The cloud infrastructure is made available to the general public or a large industry group and is owned by an organization selling cloud services.
- Also known as external cloud or multi-tenant cloud, this model essentially represents a cloud environment that is openly accessible.



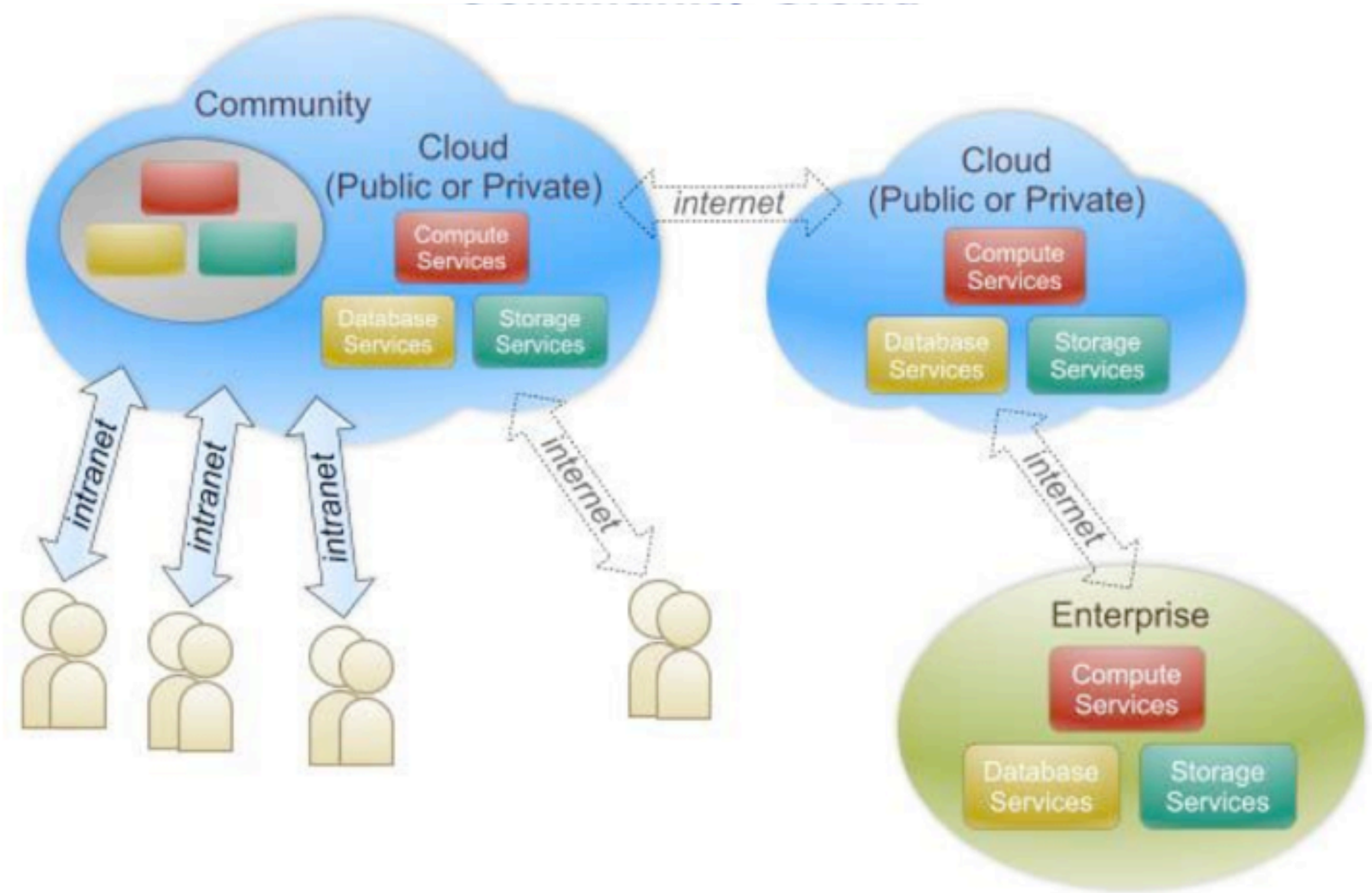
Private Cloud

- The cloud infrastructure is operated solely for an organization. It may be managed by the organization or a third party and may exist on premise or off premise.
- Also referred to as internal cloud or on-premise cloud, a private cloud intentionally limits access to its resources to service consumers that belong to the same organization that owns the cloud.



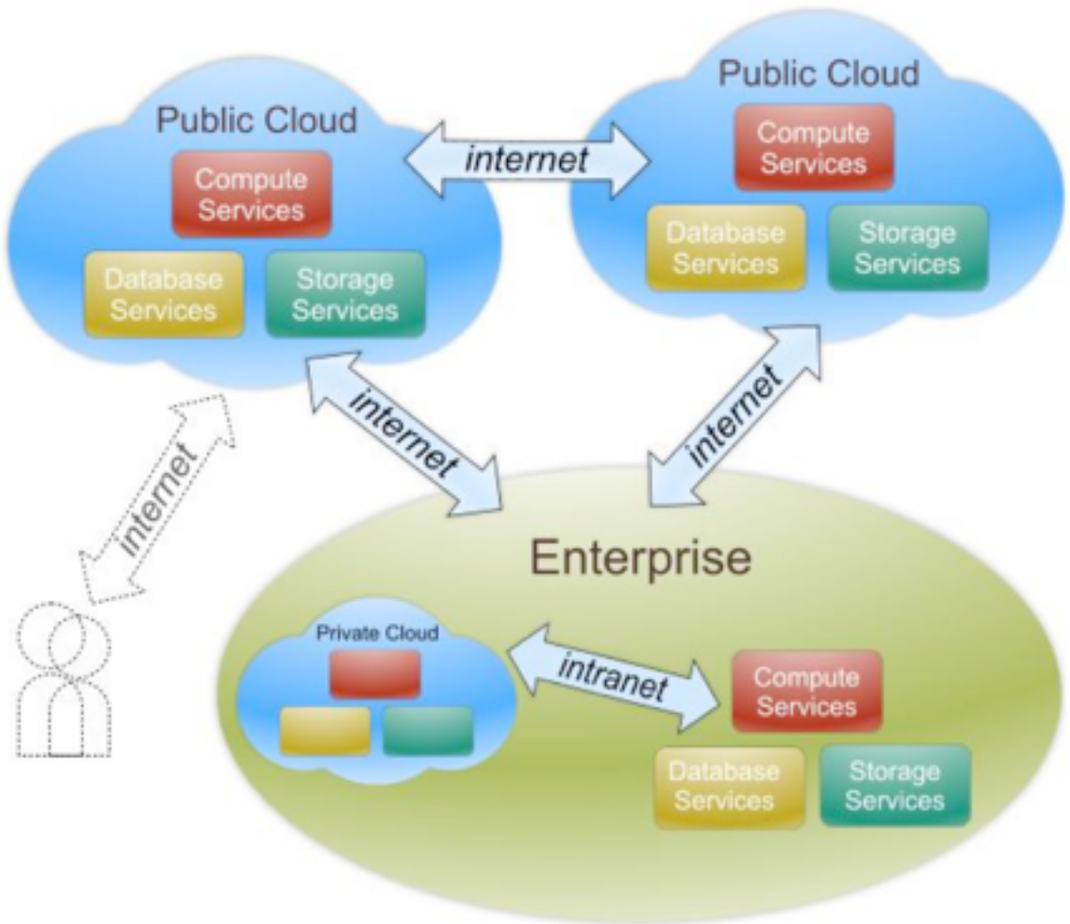
Community Cloud

- The cloud infrastructure is shared by several organizations and supports a specific community that has shared concerns (e.g., mission, security requirements, policy, and compliance considerations).



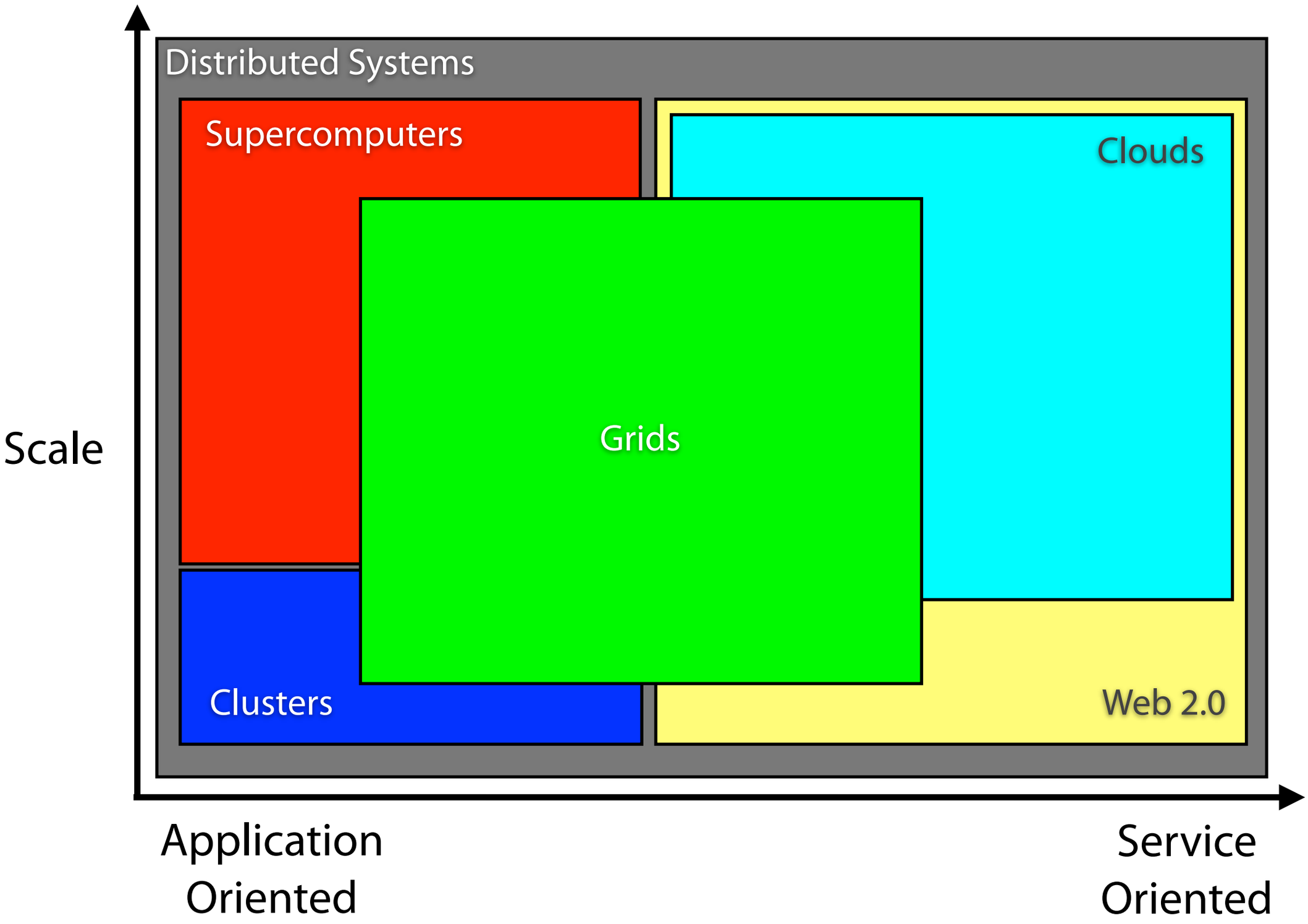
Hybrid Cloud

- The cloud infrastructure is a composition of two or more clouds (private, community, or public) that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load-balancing between clouds).



Grids versus Clouds

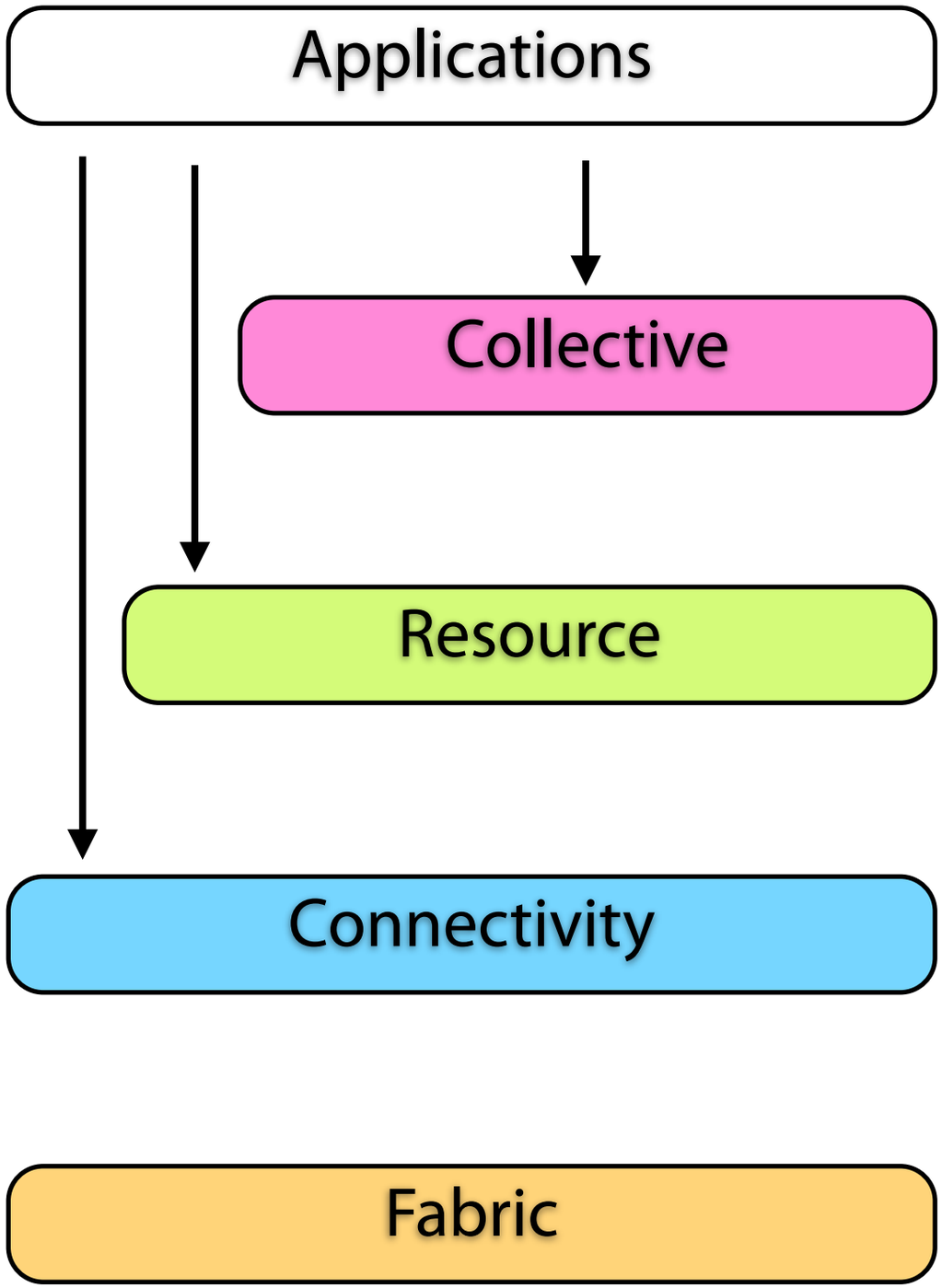
Grids and Clouds Overview

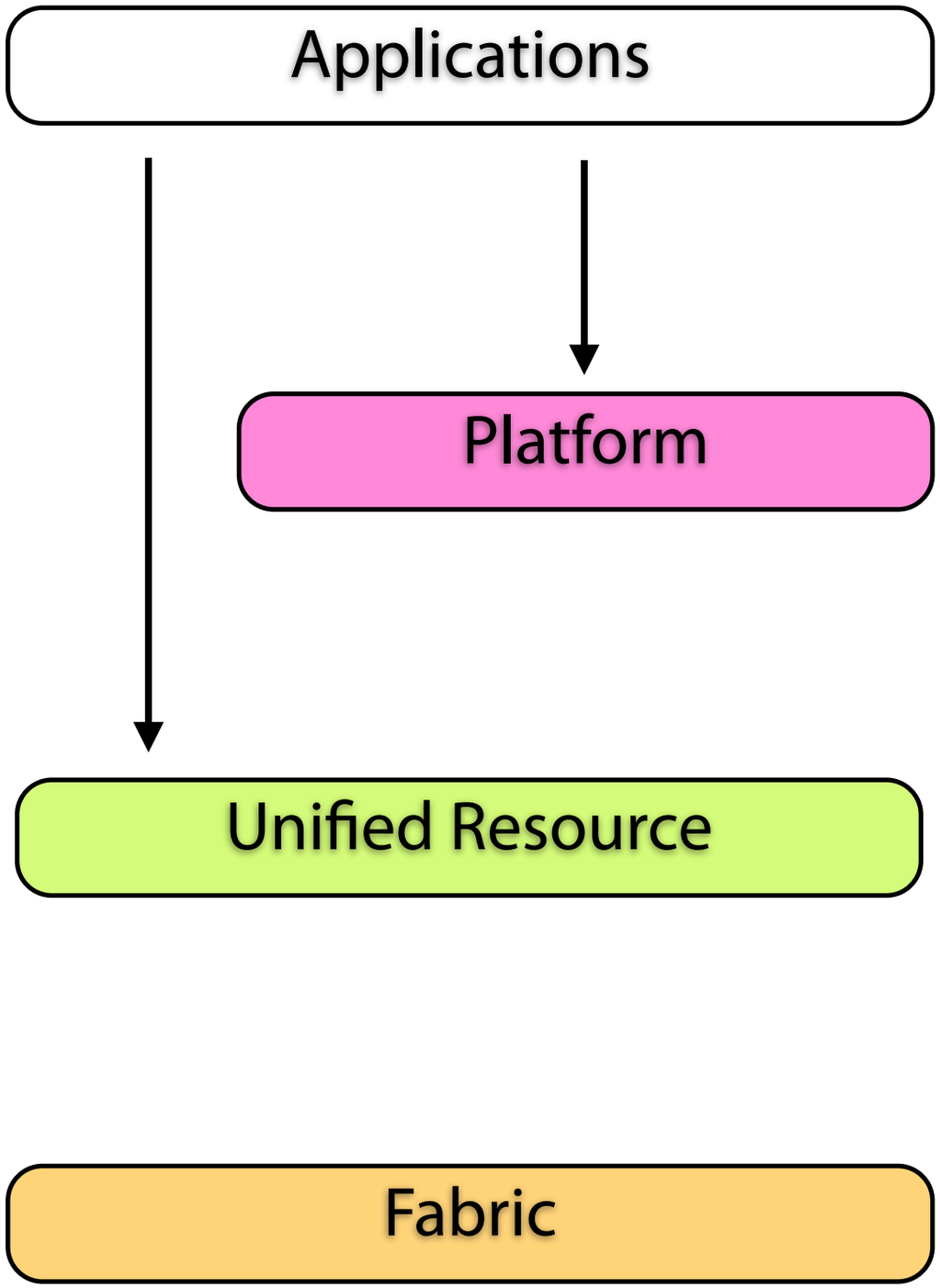


Business Model

- **Classical Computing**
  - One-time payment for unlimited use
  - Capital expenditure
  - Subject to de-pricing
- **Grid Computing**:
  - Project-oriented service units (i.e. CPU hours)
  - Prevent capital expenditure
  - Shared operational expenditure
- **Cloud Computing**
  - Pay-per-use
  - Economy of scale
  - Amazon charges instance-hours, GB/month (disk), TB/month (network)

Architecture





Compute and Data Models

- **Grid Computing**:

- Batch-scheduled compute model (LRM + GRAM)
- Dedicated resources governed by queue scheduling systems
- No native support for interactive applications
- Space-sharing allocations
- Data-to-code

- **Cloud Computing**

- Time sharing
- All users share everything
- Elasticity
- Pay-per-use
- Code-to-data (but multicores can limit)

Programming Model

- **Grid Computing**:
- Large-scale scientific computations
- Heterogenous resources
- Different administrative domains
- Fast and efficient codes
- Complex coding (focus on non-functional logic)
- Example: MPI

- **Cloud Computing**
- Large-scale data processing
- Homogeneous resources
- Same administrative domain
- Data crunching code
- Simple coding (focus on business logic)
- Example: MapReduce

Application Model

- **Grid Computing**:

- High Performance Computing
  - Tightly coupled parallel jobs
- High Throughput Computing
  - Loosely coupled parallel jobs
- Small Number of Large Batch Jobs

- **Cloud Computing**

- Still at infancy
- Independent Jobs
- Large Number of Small Interactive Jobs

Security Model

- **Grid Computing**:

- Heterogenous Resources
- Multiple Administrative Domains
- Large Interoperability
- Based on GSI, PKI, SSL/TLS, SSO, Delegation

- **Cloud Computing**

- Homogeneous Resources
- Single Administrative Domain
- Limited Interoperability
- Based on Web forms over SSL, emails, credit card

Typical Security Concerns

1. **Privileged user access**: sensitive data processed outside the enterprise needs the assurance that they are only accessible and propagated to privileged users
2. **Regulatory compliance**: a customer needs to verify if a Cloud provider has external audits and security certifications and if their infrastructure complies with some regulatory security requirements
3. **Data location**: since a customer will not know where her data will be stored, it is important that the Cloud provider commit to storing and processing data in specific jurisdictions and to obey local privacy requirements on behalf of the customer
4. **Data segregation**: one needs to ensure that one customer's data is fully segregated from another customer's data;
5. **Recovery**: it is important that the Cloud provider has an efficient replication and recovery mechanism to restore data if a disaster occurs;
6. **Investigative support**: Cloud services are especially difficult to investigate, if this is important for a customer, then such support needs to be ensured with a contractual commitment;
7. **Long-term viability**: your data should be viable even the Cloud provider is acquired by another company.

Reading Assignments

- NIST (National Institute of Standards and Technology), *NIST Cloud Computing Reference Architecture*, http://csrc.nist.gov/groups/SNS/cloud-computing/

- M. Armbrust et al., *Above the Clouds: A Berkeley View of Cloud Computing*, Technical Report No. UCB/EECS-2009-28, University of California at Berkeley, 2009.

- I. Foster et al., *Cloud Computing and Grid Computing 360-Degree Compared*, Grid Computing Environments Workshop, 2008. GCE '08 , pp.1-10, 12-16 Nov. 2008

- R. Buyya et al., *Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility*, FGCS 25(6), pp. 599-616, Jun 2009

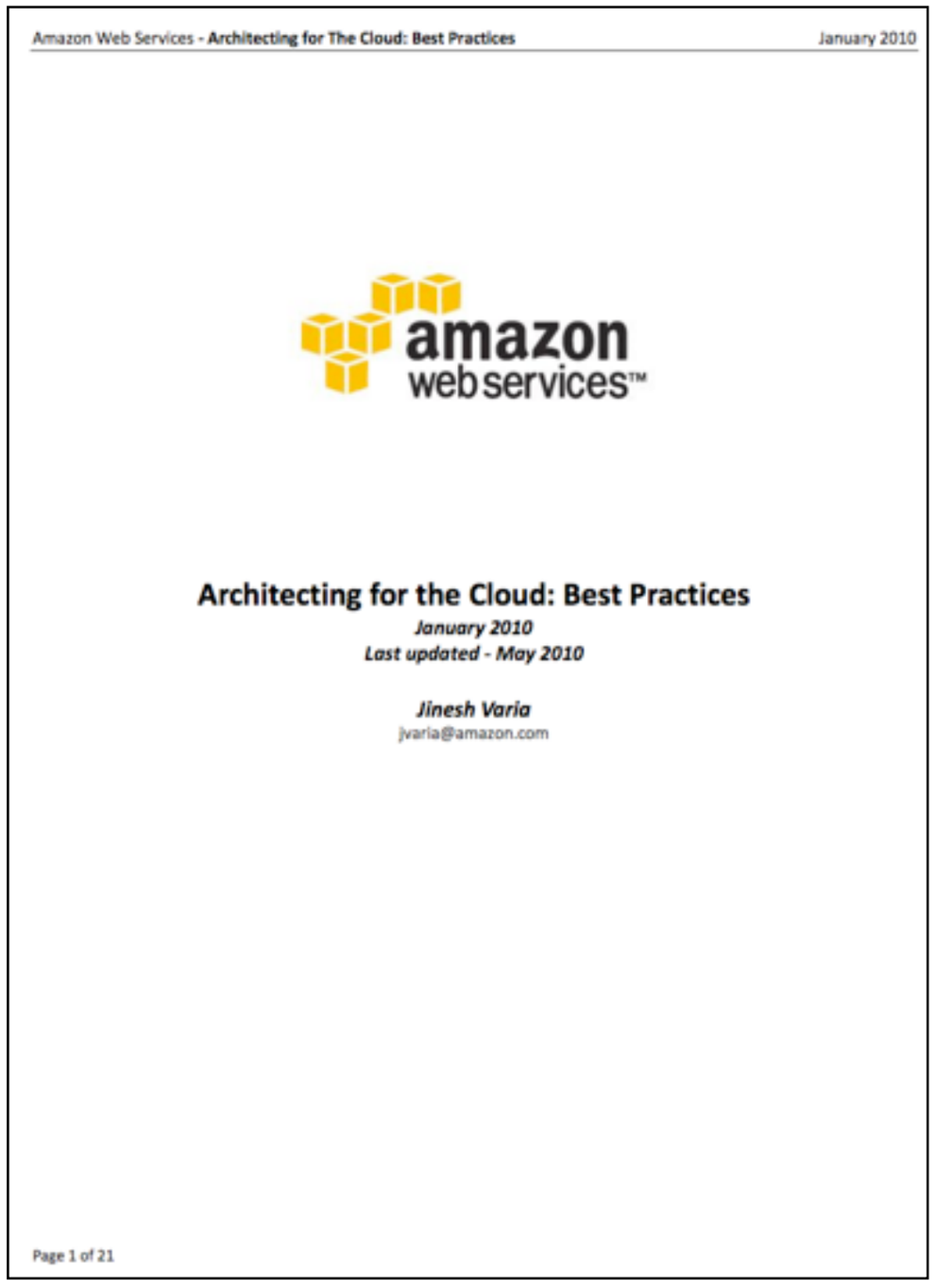
Design Patterns for the Cloud

based on

Amazon Web Services

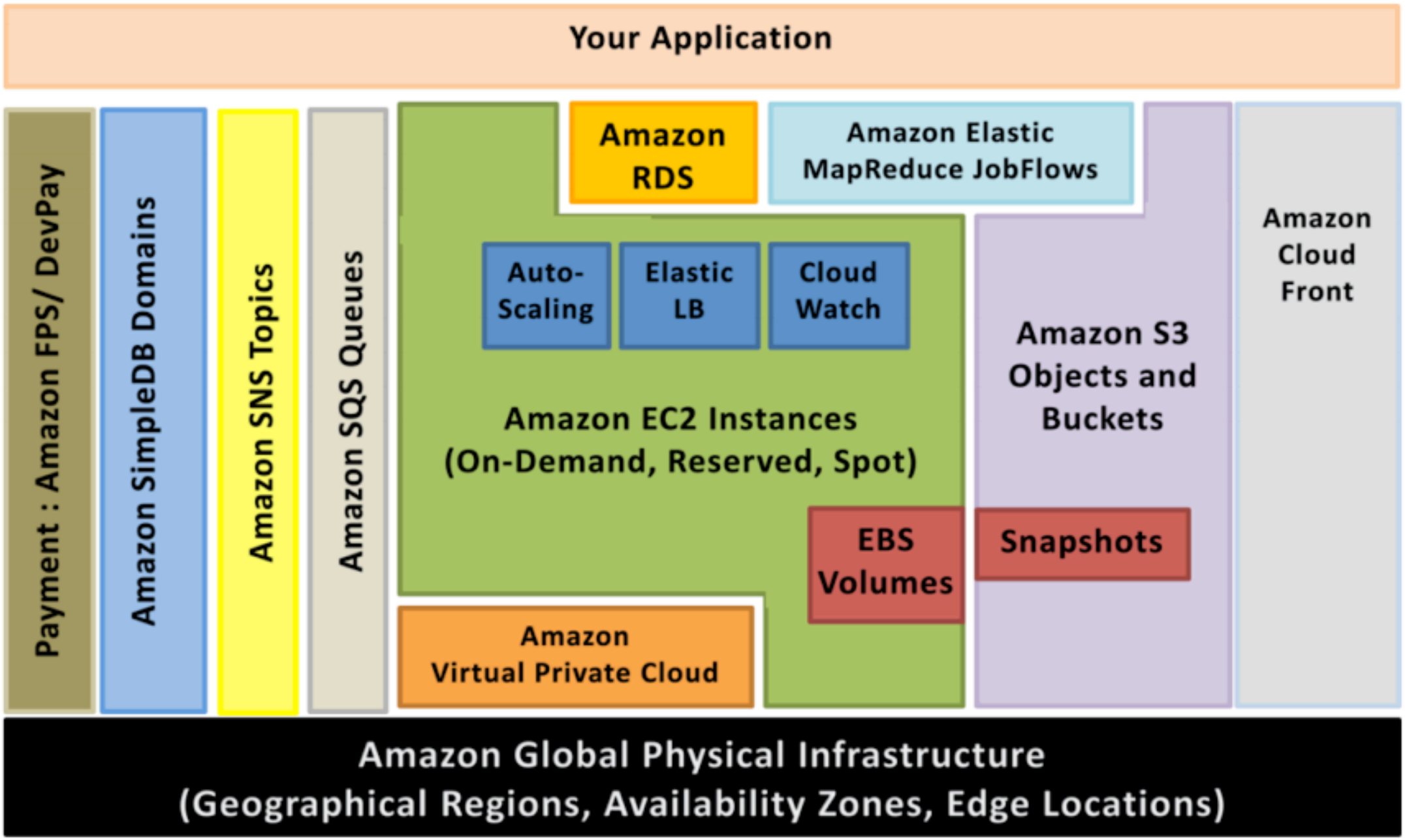
Architecting for the Cloud: Best Practices

Jinesh Varia



http://media.amazonwebservices.com/AWS_Cloud_Best_Practices.pdf

Amazon Web Services



Amazon Web Services

amazon web services

Database

DynamoDB
Predictable and Scalable NoSQL Data Store

ElastiCache
In-Memory Cache

RDS
Managed Relational Database

Redshift
Managed Petabyte-Scale Data Warehouse

Storage and Content Delivery

S3
Scalable Storage in the Cloud

EBS
Networked Attached Block Device

CloudFront
Global Content Delivery Network

Glacier
Archive Storage in the Cloud

Storage Gateway
Integrates On-Premises IT with Cloud Storage

Import Export
Ship Large Datasets

Cross-Service

Support
Phone & email fast-response 24X7 Support

Marketplace
Bull and Sell Software and Apps

Management Console
UI to manage AWS services

SDKs, IDE kits and CLIs
Develop , integrate and manage services

Compute & Networking

EC2
Virtual Servers in the Cloud

VPC
Virtual Secure Network

ELB
Load balancing Service

Auto Scaling
Automatically scale up and down

Elastic MapReduce
Managed Hadoop Framework

Direct Connect
Dedicated Network Connection to AWS

Route 53
Scalable Domain Name System

Deployment & Management

CloudFormation
Templated AWS Resource Creation

CloudWatch
Resource and Application Monitoring

Data Pipeline
Orchestration for Data-Driven Workflows

Elastic Beanstalk
AWS Application Container

IAM
Secure AWS Access Control

OpsWorks
DevOps Application Management Service

CloudHSM
Hardware-based key storage for compliance

App Services

CloudSearch
Managed Search Service

Elastic Transcoder
Easy-to-use Scalable Media Transcoding

SES
Email Sending Service

SNS
Push Notification Service

SQS
Message Queue Service

SWF
Workflow Service for Coordinating App Components

AWS Global Physical Infrastructure
(Geographical Regions, Availability Zones, Edge Locations)

Scalable Architectures

A **scalable architecture** is critical to take advantage of a **scalable infrastructure**

The cloud is designed to provide conceptually **infinite scalability**.

Characteristics of Truly Scalable Service

- Increasing **resources** results in a **proportional** increase in **performance**
- A scalable service is capable of **handling heterogeneity**
- A scalable service is **operationally efficient**
- A scalable service is **resilient**
- A scalable service becomes more **cost effective** when it grows

1. Design for Failure

- "Everything fails, all then time" - Werner Vogels, Amazon's CTO
- Avoid single points of failure
- Assume everything fails, and design backwards
- Goal: Applications should continue to function even if the underlying physical hardware fails or is removed or replicated
- The following strategies can help in event of failure:

1. Have a coherent backup and restore strategy for your data and automate it
2. Build process threads that resume on reboot
3. Allow the state of the system to re-sync by reloading messages from queues
4. Keep pre-configured and pre-optimized virtual images to support (2) and (3) on launch/boot
5. Avoid in-memory sessions or stateful user context, move that to data stores.

1. AWS Tactics

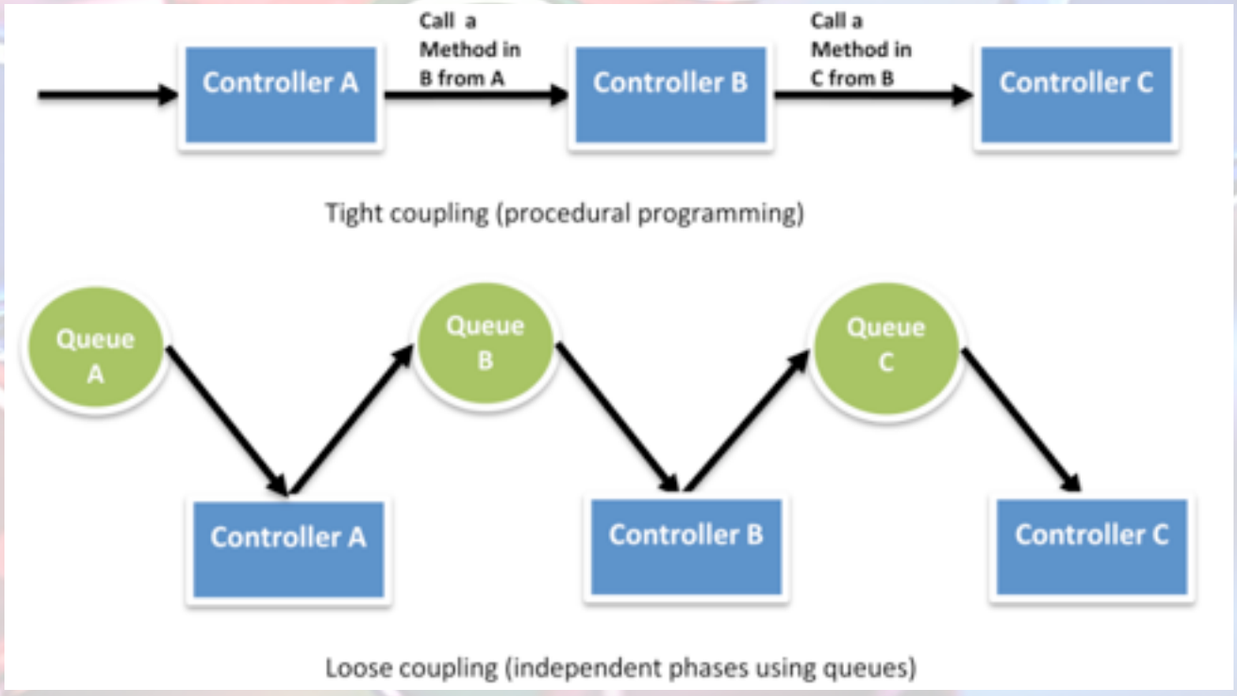
1. **Elastic IP** is a static IP that is dynamically re-mappable. You can quickly remap and failover to another set of servers so that your traffic is routed to the new servers.
2. **Availability Zones** are conceptually like logical datacenters. By deploying your architecture to multiple availability zones, you can ensure highly availability.
3. Maintain an **Amazon Machine Image** so that you can restore and clone environments very easily in a different Availability Zone.
4. Utilize **Amazon CloudWatch** (or various real-time open source monitoring tools) to get more visibility and take appropriate actions in case of hardware failure or performance degradation.
5. Setup an **Auto scaling group** to maintain a fixed fleet size so that it replaces unhealthy Amazon EC2 instances by new ones.
6. Utilize **Amazon EBS** and set up cron jobs so that incremental snapshots are automatically uploaded to **Amazon S3** and data is persisted independent of your instances.
7. Utilize **Amazon RDS** and set the retention period for backups, so that it can perform automated backups.

2. Design Loosely Coupled Systems

- The cloud reinforces the SOA design principle that **the more loosely coupled the components of the system, the bigger and better it scales**.
- Build components that **do not have tight dependencies** on each other.
- Build **asynchronous systems** and scaling horizontally become very important in the context of the cloud.
- Build systems to **scale out** by adding more instances of same component

2. AWS Tactics

1. Use **Amazon SQS** as buffers between components
2. Design every component such that it expose a **service interface** and is **responsible for its own scalability** in all appropriate dimensions and **interacts** with other components **asynchronously**
3. Bundle the logical construct of a component into an **Amazon Machine Image** so that it can be deployed more often
4. Make your applications **as stateless as possible**. Store session state outside of component (in Amazon SimpleDB, if appropriate)



3. Implement Elasticity

- Elasticity can be implemented in three ways:
  1. **Proactive Cyclic Scaling**: Periodic scaling that occurs at fixed interval (daily, weekly, monthly, quarterly)
  2. **Proactive Event-based Scaling**: Scaling just when you are expecting a big surge of traffic requests due to a scheduled business event (new product launch, marketing campaigns)
  3. **Auto-scaling based on demand**. By using a monitoring service, your system can send triggers to take appropriate actions so that it scales up or down based on metrics (utilization of the servers or network i/o, for instance)
- To implement "Elasticity", one has to first **automate the deployment process** and **streamline the configuration and build process**. This will ensure that the system can scale without any human intervention.

3. Design your AMI

- The cloud allows you to automate your deployment process.
- Take the time to create an automated deployment process early on during the migration process and not wait till the end.
- Creating an automated and repeatable deployment process will help reduce errors and facilitate an efficient and scalable update process.
- To automate the deployment process:
  - Create a **library of "recipes"** – small frequently-used scripts (for installation and configuration)
  - Manage the configuration and deployment process using **agents bundled inside an AMI**
  - **Bootstrap your instances**

3. AMI Design Approaches

Web Server

App Server

MVC

Your code

Libraries

Packages

DB

Framework

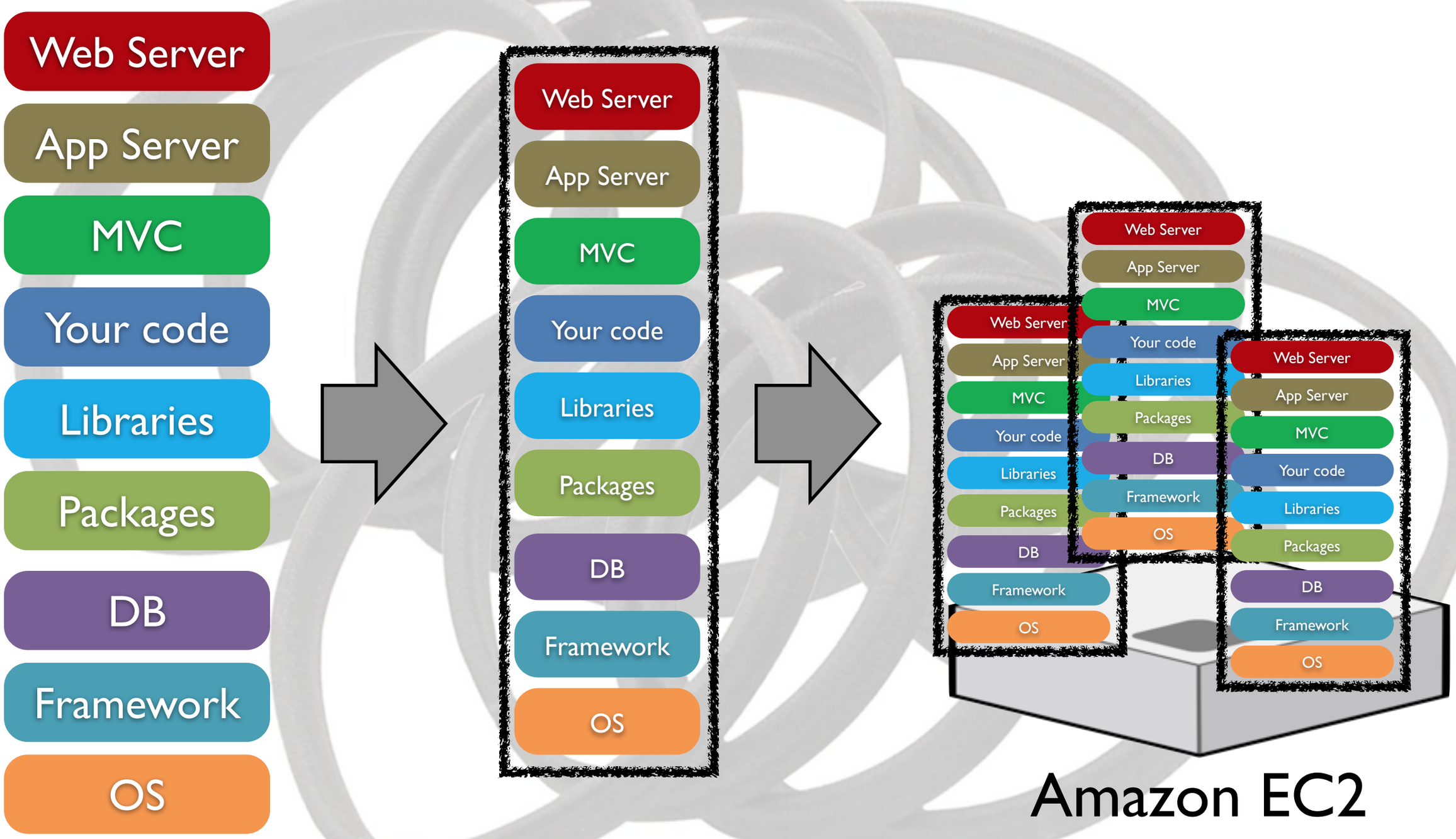
OS

1. **Inventory of static AMIs**

2. **Golden AMIs with fetch on boot**

3. **AMIs with Just Enough OS and agent**

3. Inventory of static AMIS



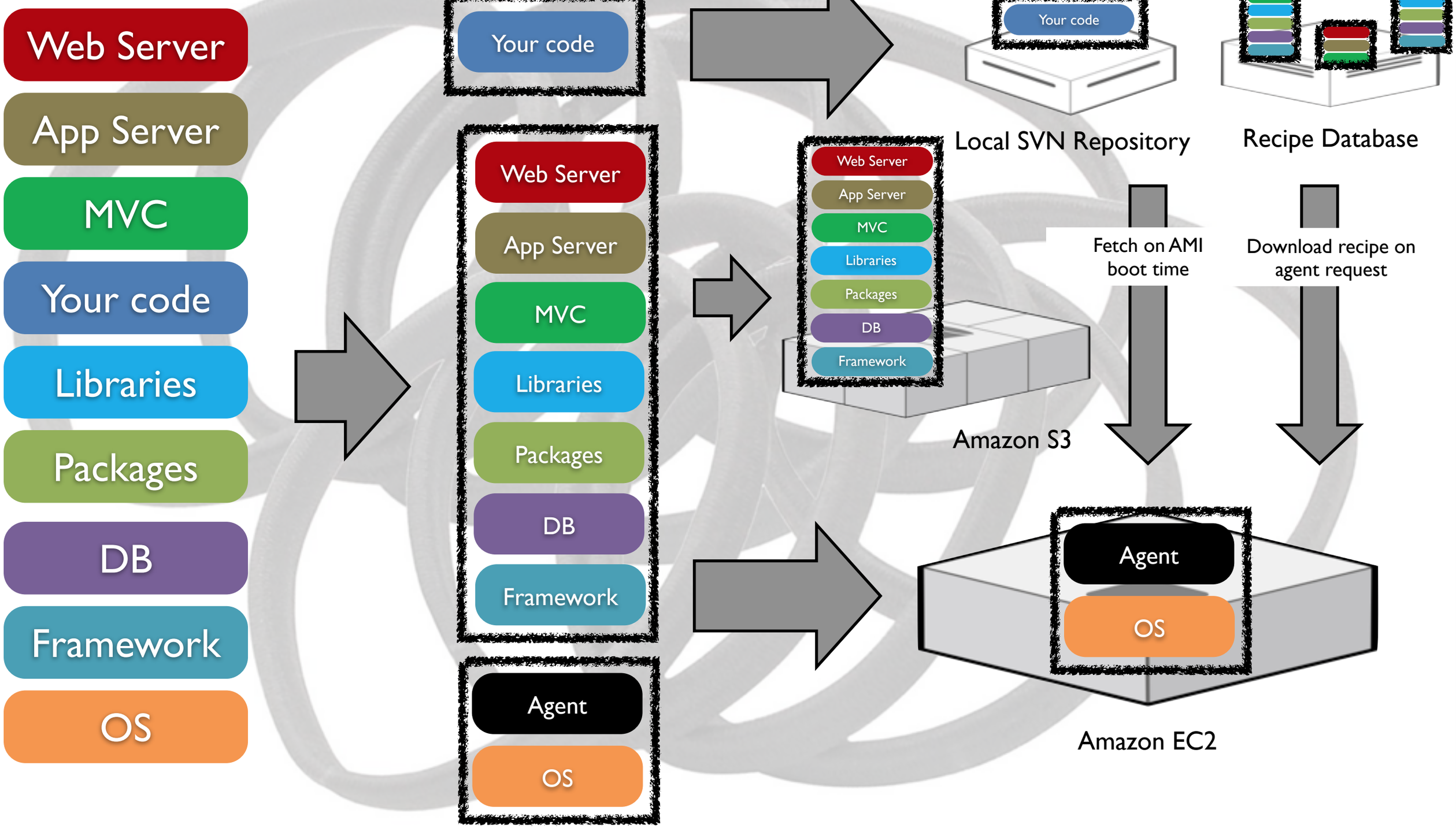
3. Golden AMIs with fetch on boot

Web Server
App Server
MVC
Your code
Libraries
Packages
DB
Framework
OS

Your code
MVC
Libraries
Packages
DB
Web Server
App Server
Framework
OS

Your code
Local SVN Repository
MVC
Libraries
Packages
DB
Amazon S3
Fetch on AMI boot time
Web Server
App Server
Framework
OS
Web Server
App Server
Framework
OS
Web Server
App Server
Framework
OS
Amazon EC2

3. AMI with JeOS and agent



3. AWS Tactics

1. Define **Auto-scaling groups** for different clusters
2. **Monitor your system metrics** (CPU, Memory, Disk I/O, Network I/O) using Amazon CloudWatch and take appropriate actions (launching new AMIs dynamically using the Auto-scaling service) or send notifications.
3. **Store and retrieve machine configuration information dynamically**: Utilize Amazon SimpleDB to fetch config data during boot-time of an instance (eg. database connection strings). SimpleDB may also be used to store information about an instance such as its IP address, machine name and role.
4. Design a build process such that it **dumps the latest builds to a bucket** in Amazon S3; **download the latest version** of an application from during system startup.
5. Invest in **building resource management tools** (Automated scripts, pre-configured images) or Use smart open source configuration management tools.
6. Bundle **Just Enough Operating System** (JeOS) and your software dependencies into an Amazon Machine Image so that it is easier to manage and maintain. Pass configuration files or parameters at launch time and retrieve user data and instance metadata after launch.
7. Reduce bundling and launch time by **booting from Amazon EBS volumes** and attaching multiple Amazon EBS volumes to an instance. **Create snapshots** of common volumes and **share snapshots** among accounts wherever appropriate.
8. Application components should **not assume health or location of hardware** it is running on.

4. Think Parallel

- **Serial** and **Sequential** is now history
- The cloud is designed to handle massively parallel operations when it comes to accessing (retrieving and storing) data: **leverage request parallelization**
- **Multi-threading** your requests by using multiple concurrent threads
- The processes of a cloud application should be made **thread-safe** through a **share-nothing** philosophy
- **Distribute** the incoming requests across **multiple asynchronous** web servers using **load balancer**

5. Leverage Storage Options

- In the cloud, you are **paying for bandwidth** in and out of the cloud
- Transfer and the cost can add up very quickly.
- Keep **dynamic data** closer to the compute element
- Keep **static data** closer to the end-user
- If a large quantity of data that needs to be processed resides outside of the cloud, use **Sneakernet :-)**
- If the data is static and not going to change often (for example, images, video, audio, PDFs, JS, CSS files), it is advisable to take advantage of a **content delivery service** so that the static data is cached at an edge location closer to the end-user (requester) thereby lowering the access latency.

5. AWS Tactics

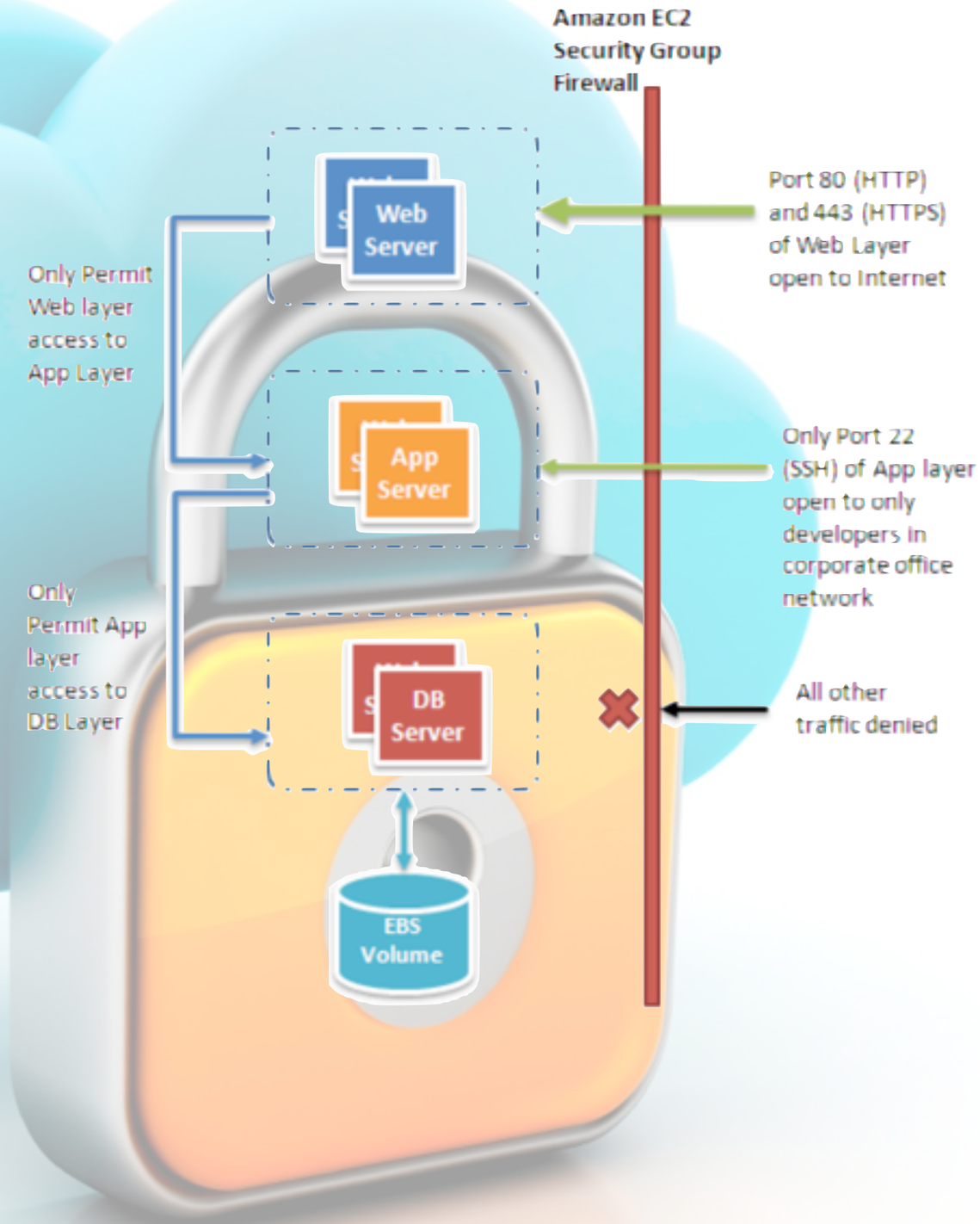
	Amazon S3 + CF	Amazon EC2 Ephemeral Store	Amazon EBS	Amazon SimpleDB	Amazon RDS
Ideal for	Storing Large write-once, read-many types of objects, Static Content Distribution	Storing non-persistent transient updates	Off-instance persistent storage for any kind of data,	Querying light-weight attribute data	Storing and querying structured Relational and referential Data
Ideal examples	Media files, audio, video, images, Backups, archives, versioning	Config Data, scratch files, TempDB	Clusters, boot data, Log or data of commercial RDBMS like Oracle, DB2	Querying, Mapping, tagging, click-stream logs, metadata, shared-state management, indexing	Complex transactional systems, inventory management and order fulfillment systems
Not recommended for	Querying, Searching	Storing Database logs or backups, customer data		Relational (joins) query	
Not recommended examples	Database, File Systems	Sensitive data	Content Distribution	OLTP, DW cube rollups	Simple lookups

6. Security

- In the cloud, **security** should be implemented **in every layer** of the cloud application architecture
- **Physical security** is typically handled by your service provider
- **Network** and **application-level security** is your responsibility
- Protect your data **in transit**
- Protect your data **at rest**
- Protect your **AWS credentials**
- Manage multiple Users and their permissions with **IAM**

6. AWS Tactics

- Every Amazon EC2 instance is protected by one or more **security groups**
- **Named sets of rules** that specify which ingress (i.e., incoming) network traffic should be delivered to your instance.
- You can specify TCP and UDP ports, ICMP types and codes, and source addresses.
- Security groups give you basic **firewall-like protection** for running instances.



Web Application Hosting Example

Traditional Architecture

Exterior Firewall
Hardware or Software Solution to open standard ports (80, 443)

Web Load Balancer
Hardware or Software solution to distribute traffic over web servers

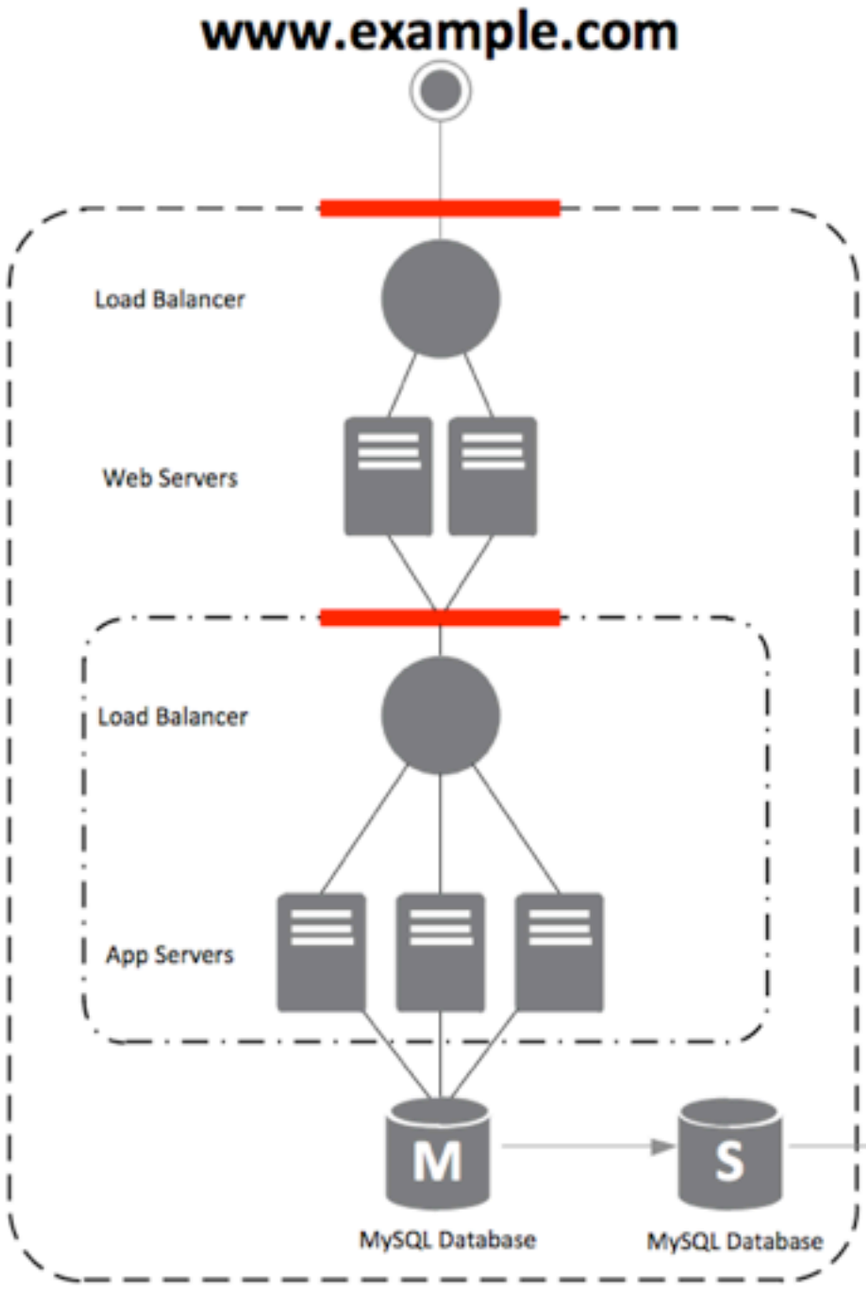
Web Tier
Fleet of machines handling HTTP requests

Backend Firewall limits access to application tied from web tier

App Load Balancer
Hardware or Software solution to spread traffic over app servers

App Server Tier
Fleet of machines handling Application specific workloads
Caching server machines can be implemented at this layer

Data Tier
Database Server machines with master and local running separately, Network storage for static objects



taken from: http://media.amazonwebservices.com/AWS_Web_Hosting_Best_Practices.pdf

Web Application Hosting Example

Amazon WS Architecture

Route 53
Provides DNS services to simplify domain management and zone APEX support (http://example.com)

Elastic Load Balancer
ELB to spread traffic to Web Server Autoscaling groups

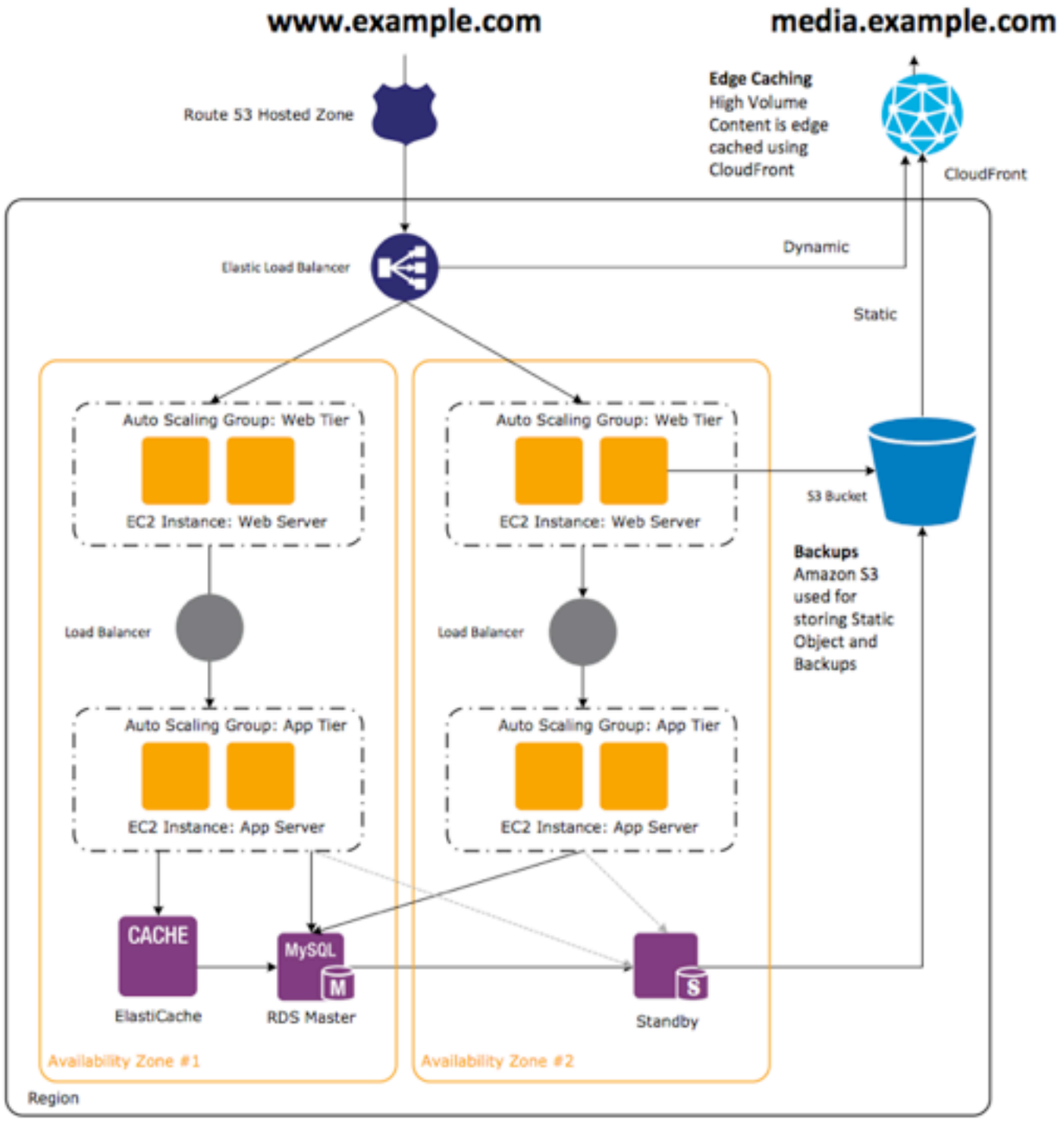
Exterior Firewall moved to every Web Server instance via Security Groups
Auto Scaling Web Tier
Group of EC2 instances handling HTTP requests.
Backend Firewall moved to every back-end instance

App Server Load Balancer
Software LB (e.g. HAProxy) on EC2 instances to spread traffic over app server cluster

Auto-scaling App Tier
Group of EC2 instances running the actual app. Instances belong to Auto Scaling group

ElastiCache
Provides caching services for app, removing load from database tier

DB Tier
MySQL RDS DB creates a highly available architecture with multi-AZ deployments. Read-only replicas can also be used to scale read intensive applications



taken from: http://media.amazonwebservices.com/AWS_Web_Hosting_Best_Practices.pdf