



UNIVERSITÀ DEGLI STUDI DI NAPOLI  
**FEDERICO II**

# La Virtualizzazione nel Dispiegamento di Sistemi Complessi

Prof. Sergio Di Martino  
Dott. Luigi Libero Lucio Starace

# La virtualizzazione

- ▶ La virtualizzazione è una tecnica per aumentare l'efficacia e l'efficienza di Centri Elaborazione Dati complessi
- ▶ Tecnica risalente agli anni '60, ma ritornata in auge una decina di anni fa, grazie all'evoluzione tecnologica delle CPU

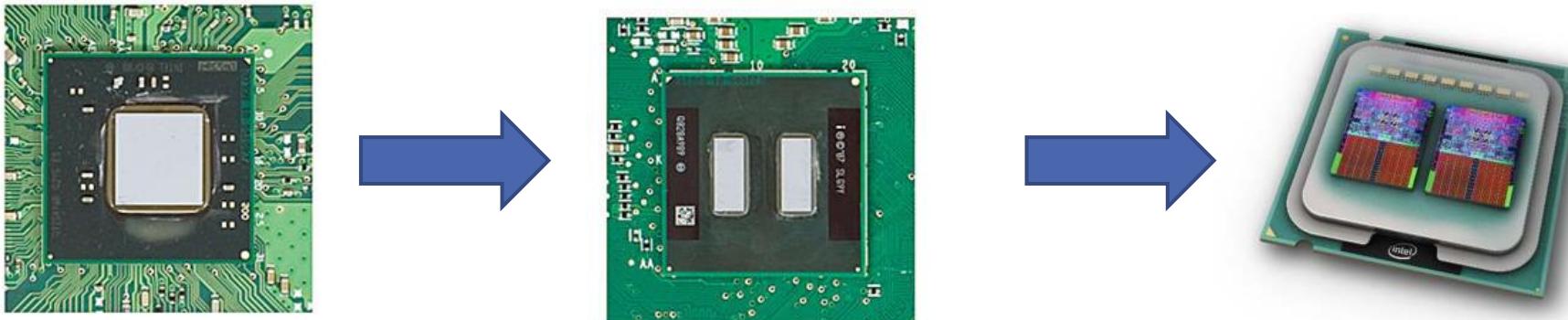
# L'evoluzione delle CPU

# Evoluzione delle CPU

- ▶ I microprocessori hanno da sempre avuto un rapidissimo incremento nelle prestazioni e una diminuzione dei costi.
- ▶ La strategia per migliorare le prestazioni è storicamente stata quella di aumentare la "velocità" di un processore.
  - ▶ 1978: Intel 8086, 4MHz
  - ▶ 1985: Intel 386, 25 MHz (~ 6x)
  - ▶ 1995: Intel Pentium Pro, 200 MHz (~ 8x)
  - ▶ 2005: Intel Pentium P4 3,8 GHz (3800 MHz) (~ 19x)
- ▶ Questa strategia ha subito una battuta d'arresto nel 2005
  - ▶ 2020: Intel i9-10980XE 3,0 GHz

# Le CPU multicore

- ▶ Raggiunti i limiti fisici del silicio, la soluzione è stata aumentare il numero di unità di processo (multi core), allo scopo di aumentare la potenza di calcolo.
  - ▶ 2020: Intel i9-10980XE 3,0 GHz → 18 unita di calcolo
  - ▶ 2020: AMD Ryzen™ Threadripper™ 3970X 3,7 GHz → 32 unita di calcolo



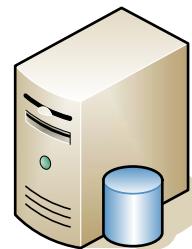
# Problemi dei Multi-core

- ▶ Tradizionalmente i programmi sono stati scritti per essere eseguiti su un computer con una singola CPU.
- ▶ La stragrande maggioranza delle applicazioni sono costituite da programmi sequenziali.
- ▶ Come sfruttare più unità in parallelo?

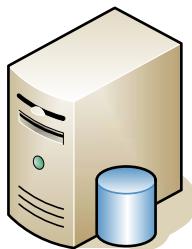
# Infrastruttura tradizionale di un CED

# Infrastruttura Tradizionale di CED

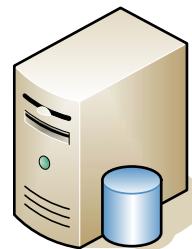
- ▶ Singoli server che offrono singoli servizi e/o applicazioni (*one workload, one box*)
- ▶ Nuovo Servizio e/o Applicazione → Nuovo Server, con proprio Storage



**Server Application**



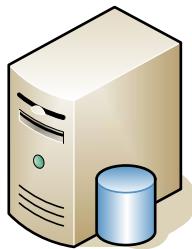
**Server di Dominio**



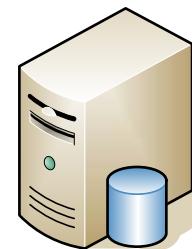
**Server Database**



**Server di Posta**



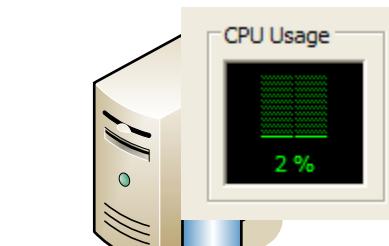
**Server Fax**



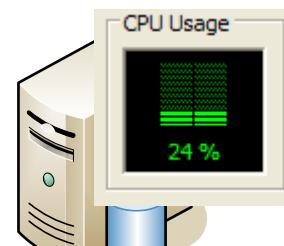
**Server Terminal**

# Infrastruttura Tradizionale di CED

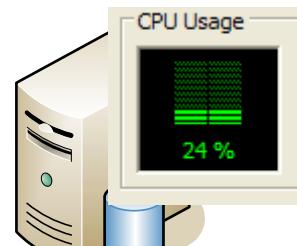
- ▶ Tutti i server, consumano (tanta) corrente, emettono calore, occupano spazio, vanno gestiti fisicamente...



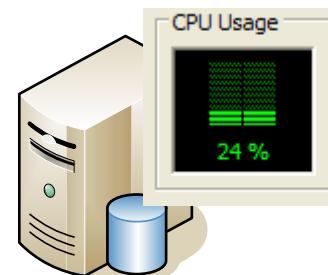
**Server Application**



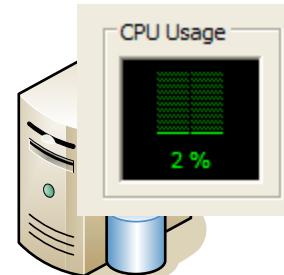
**Server di Dominio**



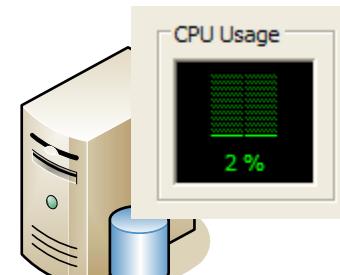
**Server Database**



**Server di Posta**



**Server Fax**



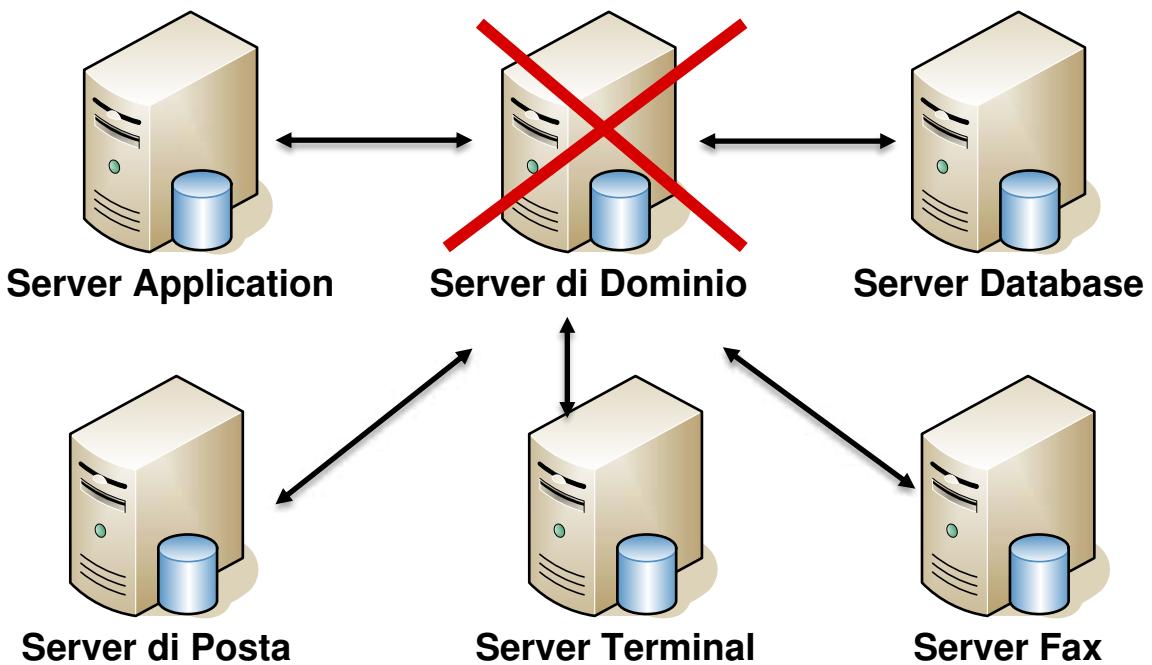
**Server Terminal**

# Infrastruttura Tradizionale di CED: Conseguenze

- ▶ Sottoutilizzo delle risorse
  - ▶ La maggior parte dei server sfrutta in media tra il 5% ed il 15% della propria capacità computazionale.
  - ▶ In caso di servizi critici, necessità di ridondanza (computer di riserva normalmente non usati).
- ▶ Aumento dei costi di installazione e gestione
  - ▶ Rilascio aggiornamenti su scala nazionale molto costosi
  - ▶ Back-up da effettuare su ogni macchina
  - ▶ ...

# Infrastruttura Tradizionale di CED : Conseguenze

- ▶ I servizi sono intrinsecamente interconnessi.
  - ▶ Es: Il server di dominio, che gestisce le autenticazioni degli utenti, viene interrogato da tutti gli altri servizi/server
  - ▶ Cosa succede se si rompe un server?



# Fail di un server nello scenario tradizionale

- ▶ In caso di FAIL di un server si possono bloccare i servizi di altri Server in comunicazione con esso.
- ▶ Ciò comporta:
  - ▶ Interruzione dell'operatività collegata a quel servizio
  - ▶ Nei casi più gravi interruzione dell'intera operatività aziendale
  - ▶ Aumento dei costi (costi di fermo e di ripristino)
  - ▶ Tempi di fermo non sempre prevedibili

# Le Macchine virtuali

Concetti Generali

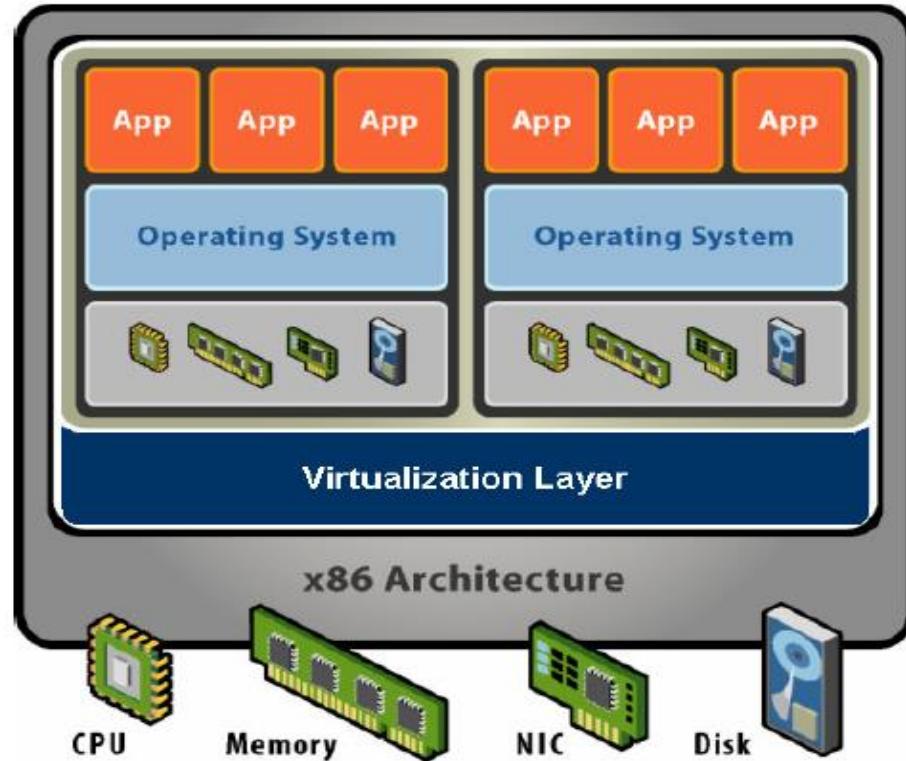
# Un Server Fisico

- ▶ Un Server fisico consiste di:
  - ▶ Una serie di risorse hardware
  - ▶ Un sistema operativo
  - ▶ Uno o più software applicativi
- ▶ **Idea:** se la CPU del Server ha più unità di calcolo parallele, possiamo "simulare" l'esistenza di più computer, indipendenti dal "vero" hw sottostante?



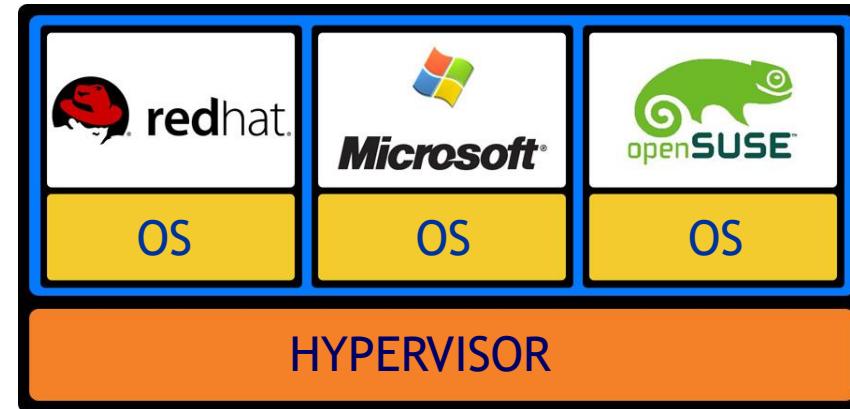
# Una Macchina Virtuale

- ▶ La virtualizzazione consiste nella creazione di una versione virtuale di una risorsa normalmente fornita fisicamente e appartenente a un sistema.
  - ▶ Migliore sfruttamento delle risorse e delle unità di calcolo



# Come si ottiene la Virtualizzazione?

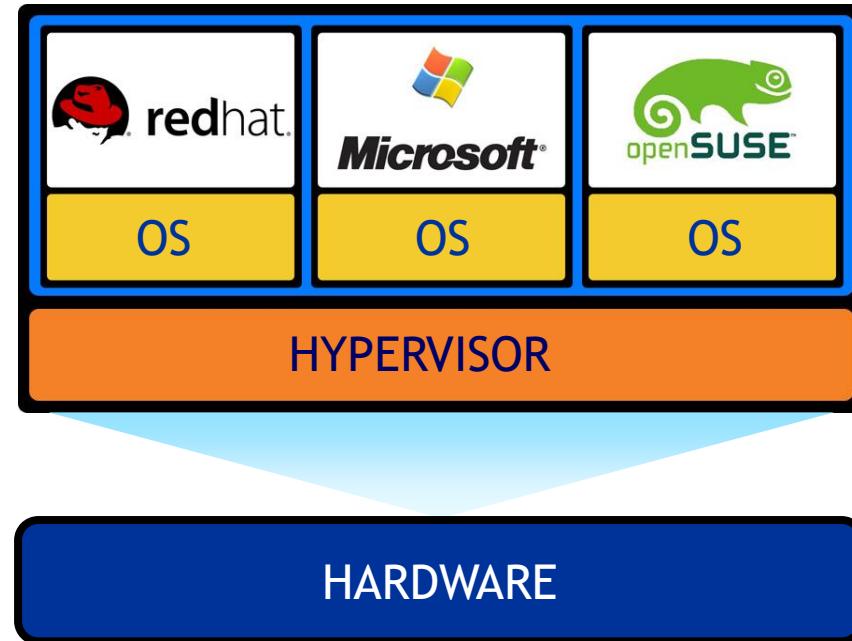
- ▶ Si introduce un nuovo strato che parte prima del sistema operativo, chiamato Hypervisor.
- ▶ L'Hypervisor simula l'esistenza di più computer "virtuali" su uno fisico



HARDWARE

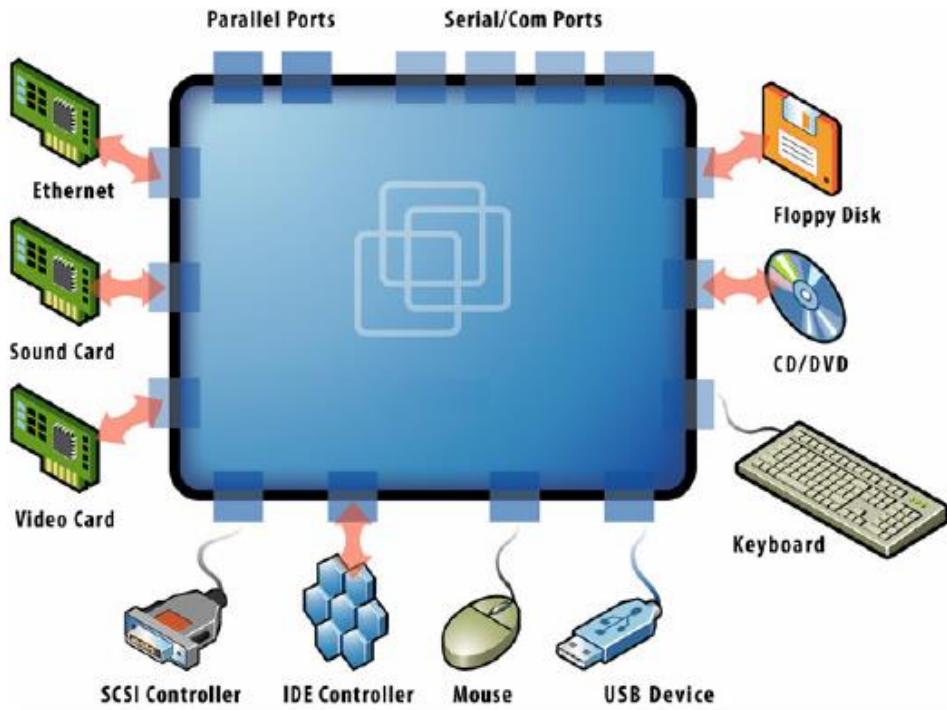
# Come si ottiene la Virtualizzazione?

- ▶ L'Hypervisor offre le risorse fisiche ai computer virtualizzati, e si occupa di gestirle in modo concorrente a favore delle macchine virtuali.
- ▶ In ogni computer virtualizzato, si installa un normalissimo sistema operativo, che non ha modo di sapere se sta funzionando su un computer "reale" o virtuale.



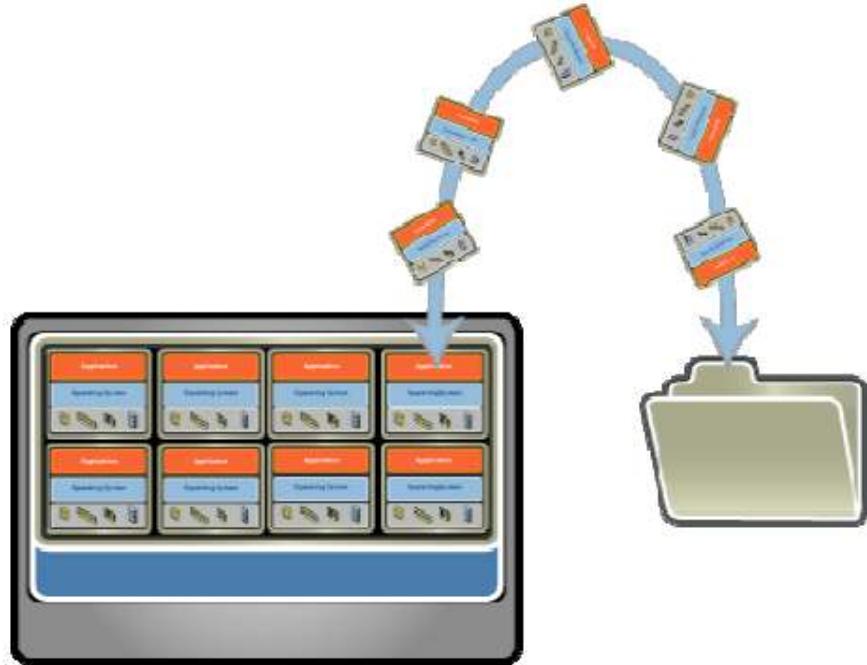
# Indipendenza dall'Hardware

- ▶ L'Hypervisor simula una configurazione standard di un server, nascondendo il vero hardware fisico
- ▶ Conseguenza: Le Virtual Machines possono funzionare su server fisici diversi senza bisogno di essere configurate



# Memorizzazione di una VM

- ▶ Una virtual machine è **un file**, contenente:
  - ▶ Sistema Operativo, Applicativi, Dati
  - ▶ Rappresentazione della Memoria del Server
- ▶ Un file memorizza lo stato di un server virtuale e ne permette la copia su un numero arbitrario di server fisici
- ▶ Esistono strumenti commerciali che trasformano il contenuto di un server fisico in una VM



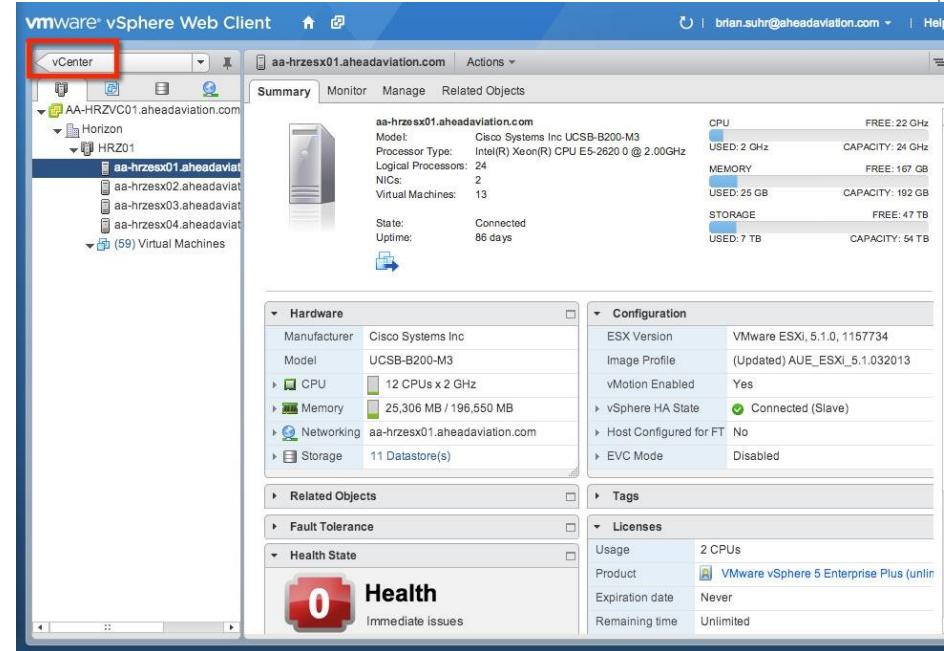
# Isolamento di una VM

- ▶ Ogni VM è totalmente isolata dalle altre VM in esecuzione sullo stesso server fisico
- ▶ In caso di attacco o di crash di una VM, le altre non sono intaccate



# Gestione di VM

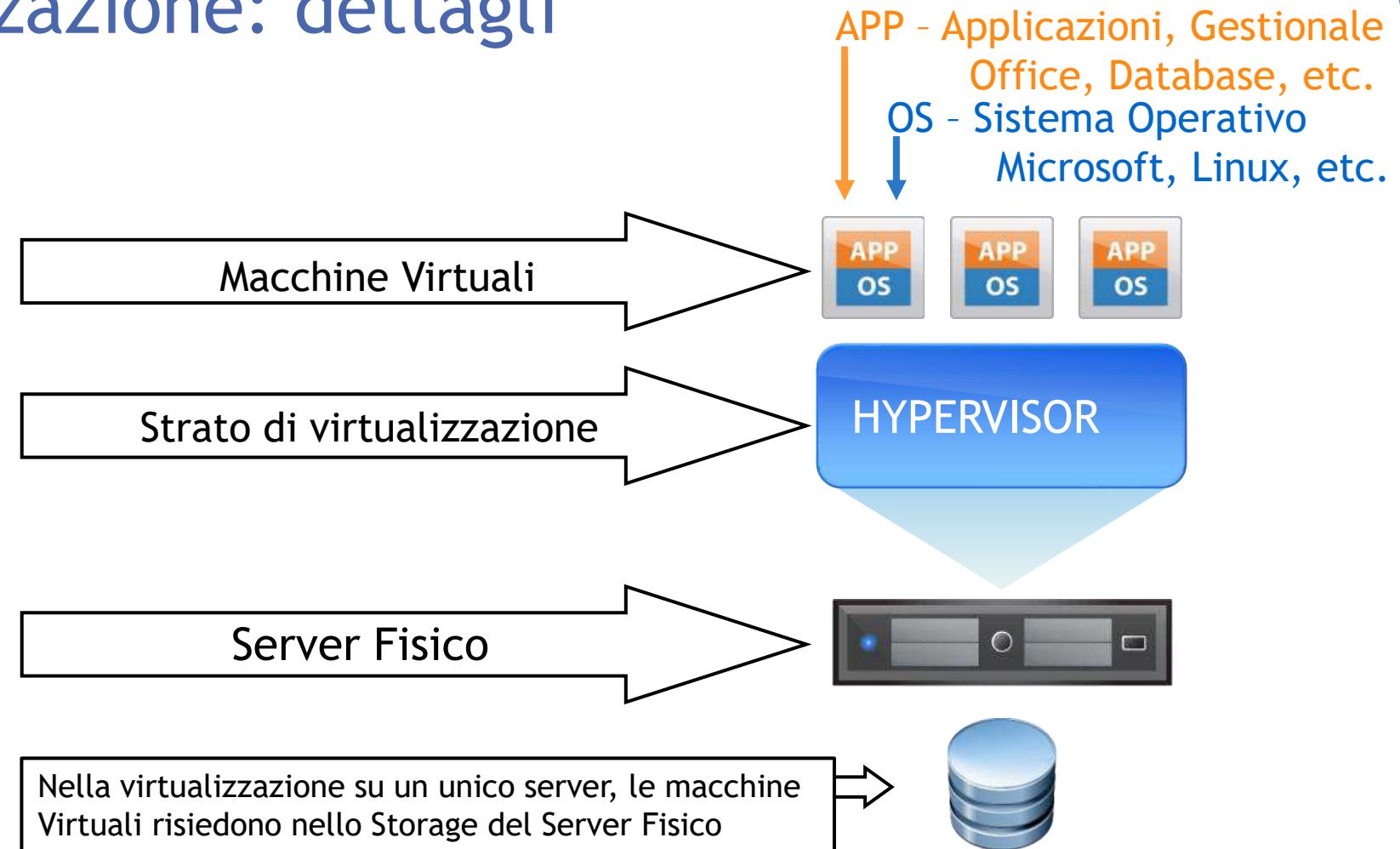
- ▶ Tutte le VM di un CED possono essere gestite attraverso un'unica finestra.
- ▶ La gestione dell'intero CED risulta centralizzata, con notevole semplificazione della gestione



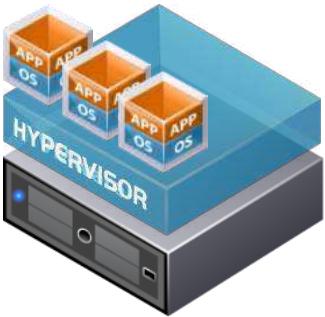
# Le Macchine virtuali

Dettagli Operativi

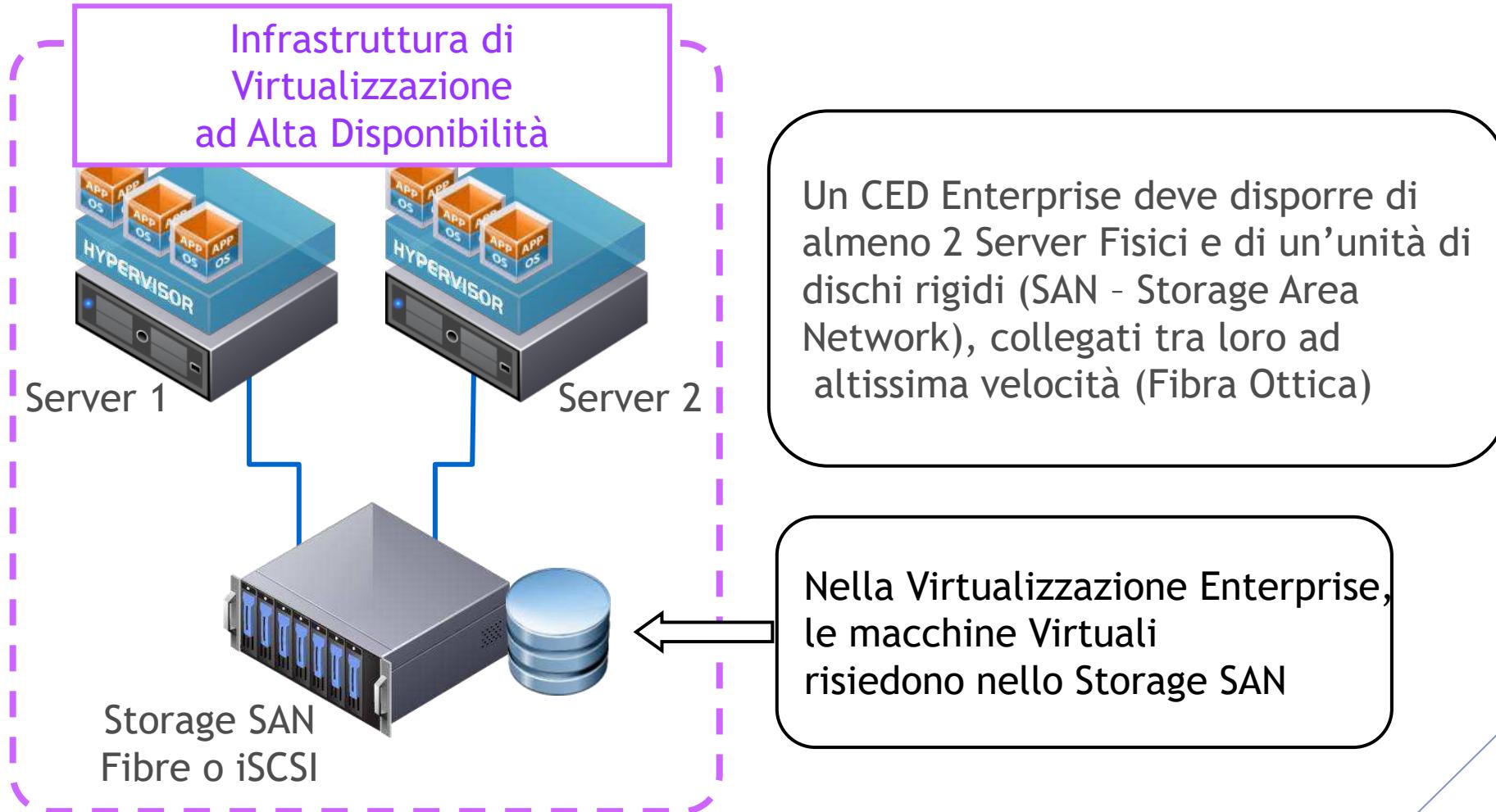
# Virtualizzazione: dettagli



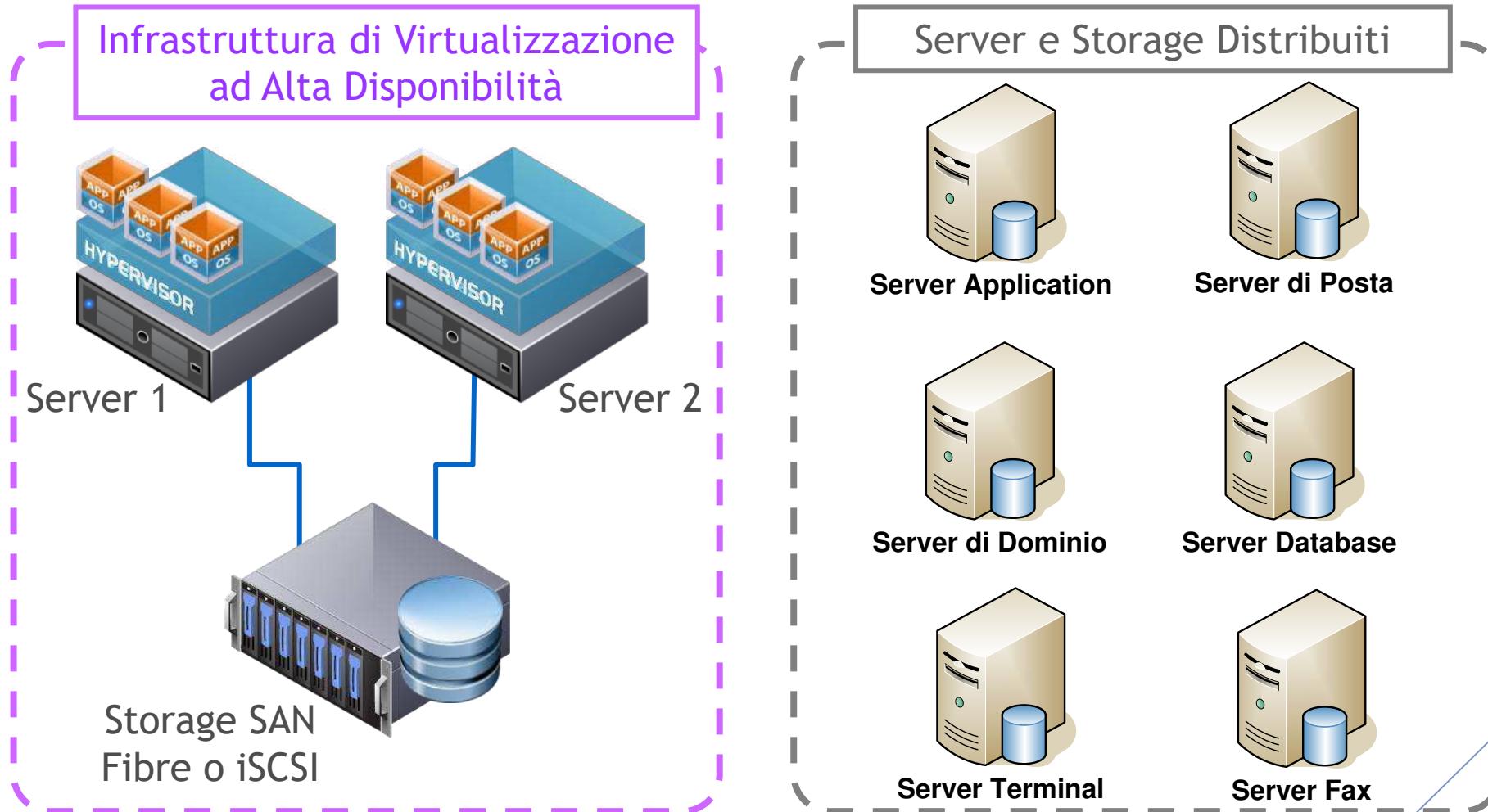
# Virtualizzazione Enterprise ad Alta Disponibilità



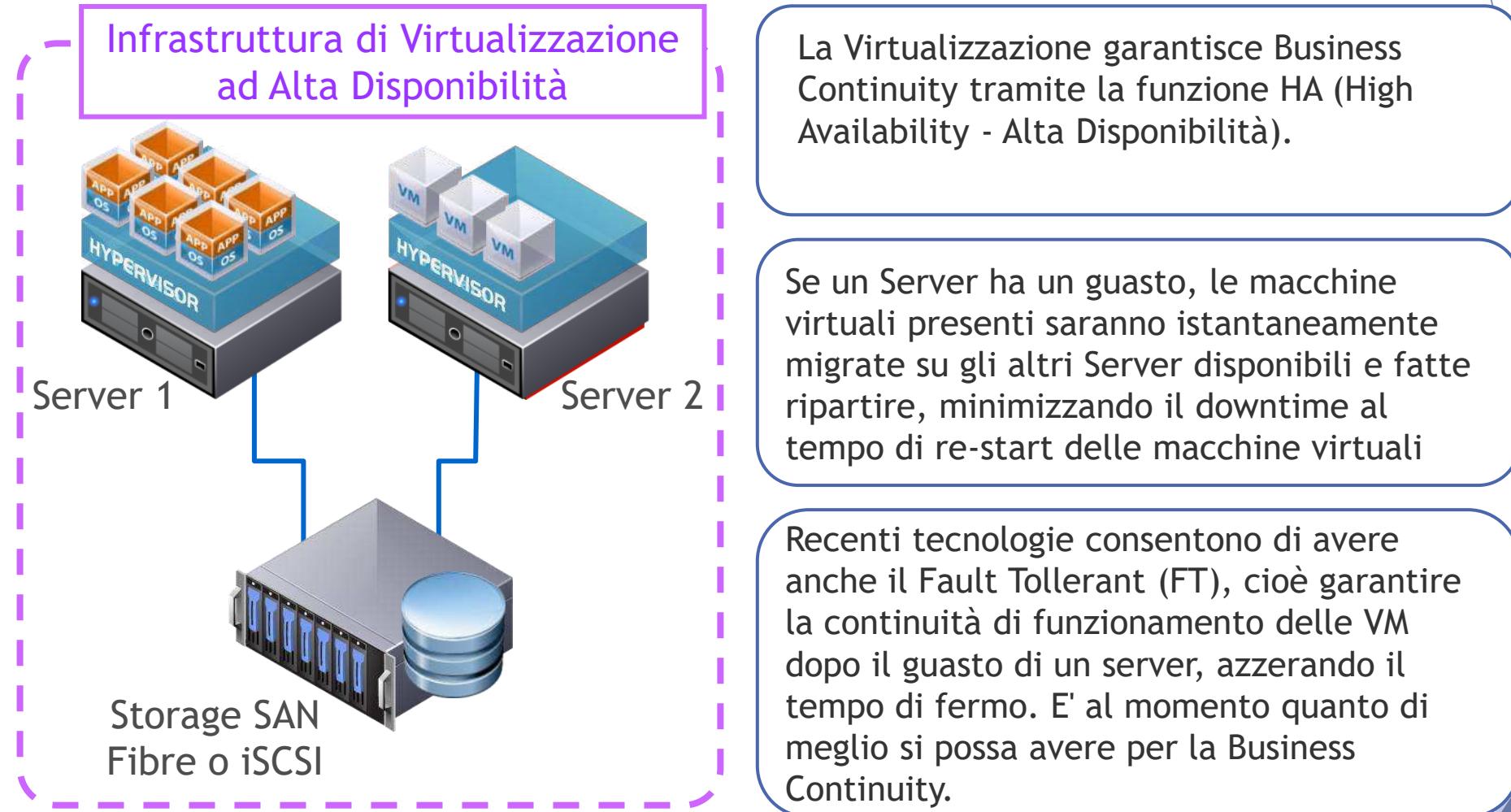
# Virtualizzazione Enterprise ad Alta Disponibilità



# Migrazione da Fisico a Virtuale

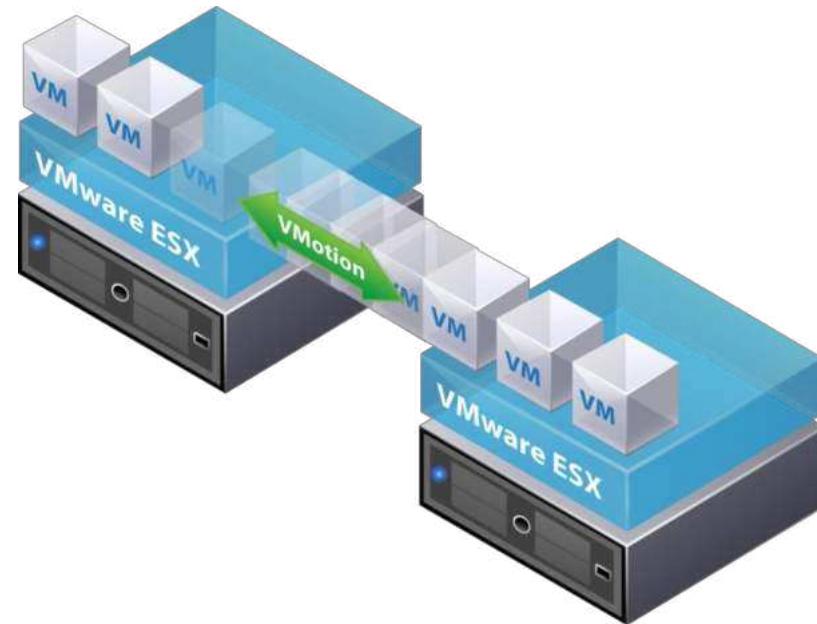


# Come Funziona un sistema ad Alta Disponibilità



# Virtualizzazione: La Migrazione Live delle VM

- ▶ La migrazione "a caldo" delle VM fra i vari Server Fisici è una delle tante funzionalità del sistema di virtualizzazione.
- ▶ Consente operazioni di manutenzione o di bilanciamento del carico mantenendo l'operatività dei server virtuali.

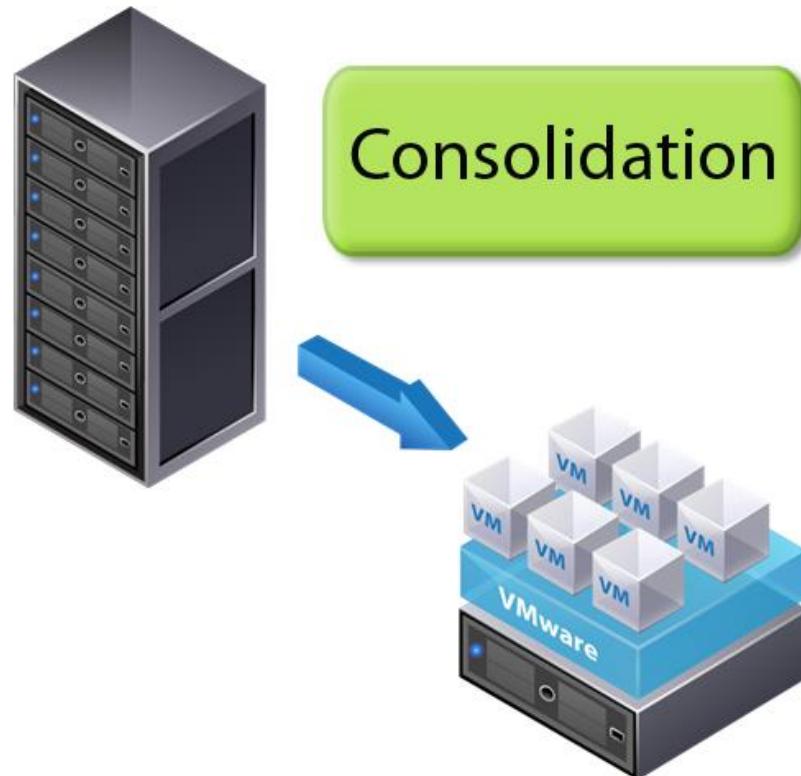


# Le Macchine virtuali

Benefici

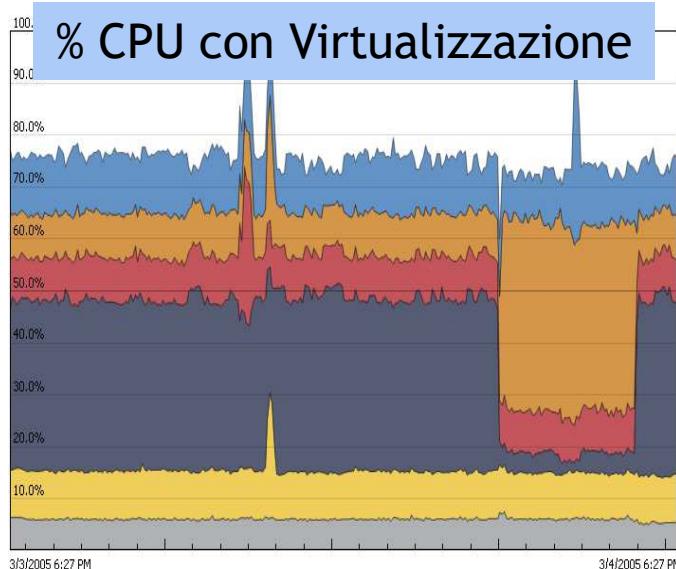
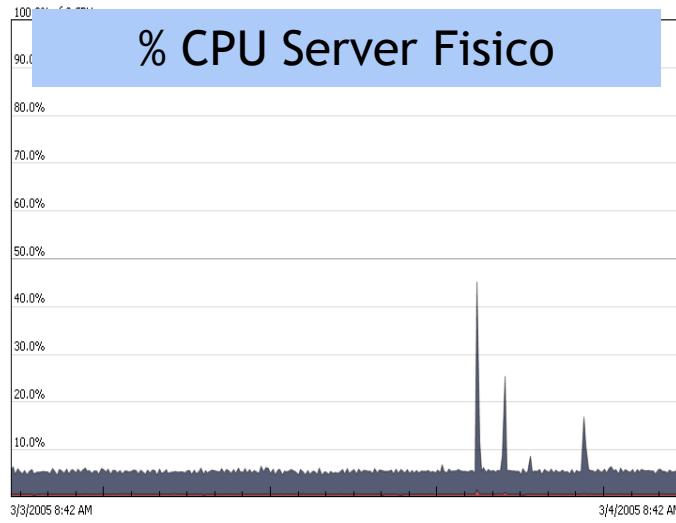
# Consolidamento di un CED

- ▶ Con l'introduzione della virtualizzazione si parla di "**consolidamento**" di un CED.
- ▶ Sfruttando al meglio le risorse di computazione parallela offerte dai nuovi processori, è possibile ridurre significativamente il numero di server necessari per erogare i servizi informatici richiesti da un ente



# Benefici introdotti dalla Virtualizzazione: Ottimizzazione

- ▶ Ottimizzazione delle risorse HW - La virtualizzazione permette di eseguire distinti sistemi operativi contemporaneamente in una singola macchina
- ▶ Ottimizzazione della occupazione dello spazio disco - ad una macchina virtuale viene assegnato lo spazio disco strettamente necessario.
  - ▶ Il resto del volume resta disponibile per nuove macchine virtuali



# Benefici introdotti dalla Virtualizzazione: Semplificazione

- ▶ Backup e disaster recovery
  - ▶ L'intero server (SO, applicazioni, dati, dispositivi e stato) non è altro che un file, più facile creare copie di backup per il disaster recovery
- ▶ Indipendenza dall'hardware
- ▶ Rilasci semplificati
  - ▶ Possibilità di clonare “LIVE” le macchine virtuali per test e aggiornamenti, senza compromettere l'integrità di quella originale.

# Benefici introdotti dalla Virtualizzazione: Costi

- ▶ Riduzione dei costi di IT
  - ▶ Riduzione del numero dei server necessari, con conseguente risparmio sui costi energetici, di condizionamento e dimensione degli spazi fisici richiesti dal CED.
- ▶ Riduzione dei costi di personale IT
  - ▶ Gestione centralizzata: Incremento di produttività del reparto IT, dovuto alla gestione centralizzata delle macchine attraverso un'unica interfaccia di amministrazione

# La Virtualizzazione Al Ministero di Giustizia

# Virtualizzazione e Giustizia

- ▶ Al momento la maggior parte dei CED del Ministero di Giustizia sfrutta i vantaggi della virtualizzazione
- ▶ Il Ministero ha una Enterprise Licence Agreement con VMWare, leader di mercato per la fornitura di soluzioni virtuali a livello mondiale
- ▶ I contratti con le aziende fornitrici prevedono il rilascio degli applicativi per Giustizia principalmente sotto forma di Macchine Virtuali

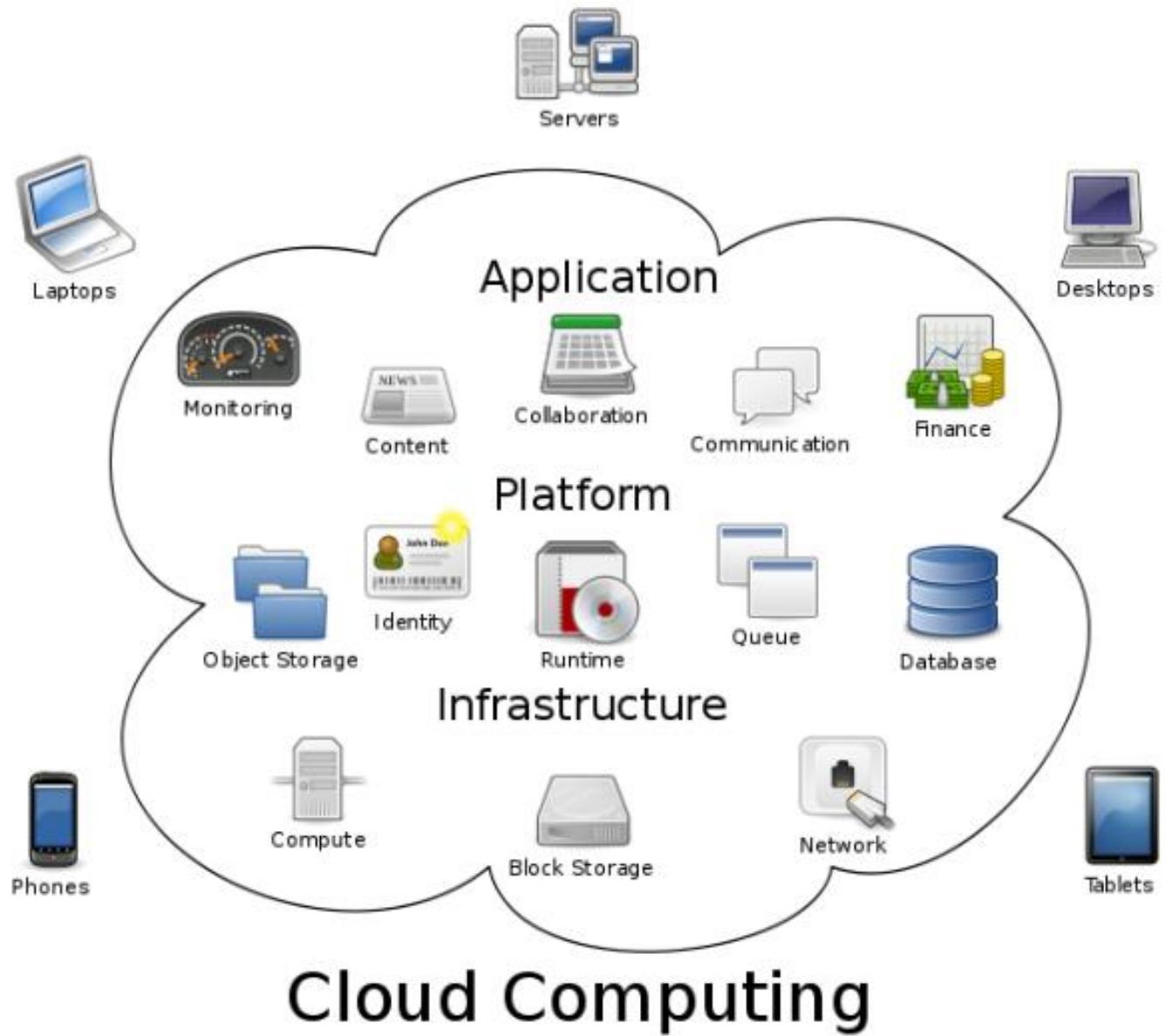
# Esempio del CISIA di Napoli

- ▶ Centro Elaborazione Dati per Civile (interdistrettuale) e Penale.
- ▶ Circa 800 macchine virtuali in esecuzione in parallelo, con bilanciamento del carico.
- ▶ Circa 30 server fisici.
- ▶ 2 Storage Area Network, con circa 500 TB di spazio.
- ▶ Gestione di 800 macchine da 2 console di amministrazione.

# Cloud Computing

# Cloud Computing

- ▶ IT resources provided as a service
  - ▶ Compute, storage, databases, queues
- ▶ Clouds leverage economies of scale of commodity hardware
  - ▶ Cheap storage, high bandwidth networks & multicore processors
  - ▶ Geographically distributed data centers
- ▶ Offerings from Microsoft, Amazon, Google, ...



# Cloud Scalability & Elasticity

# Scalability & Elasticity

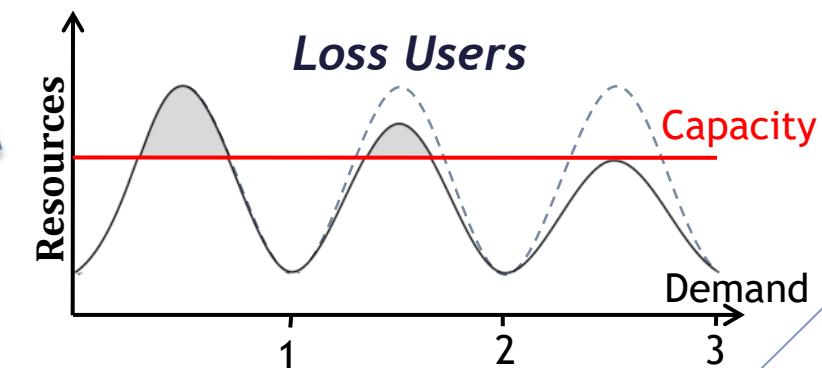
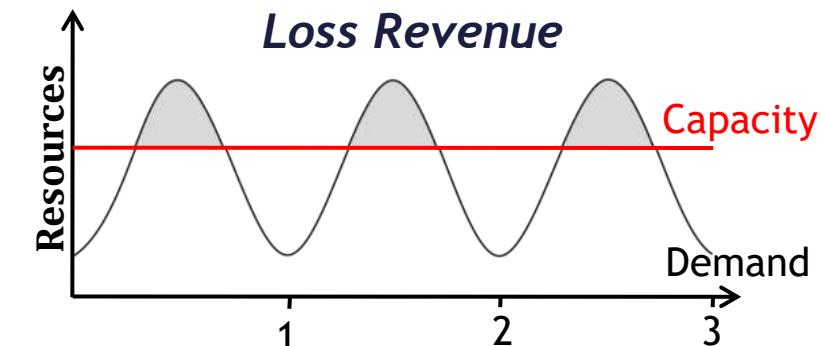
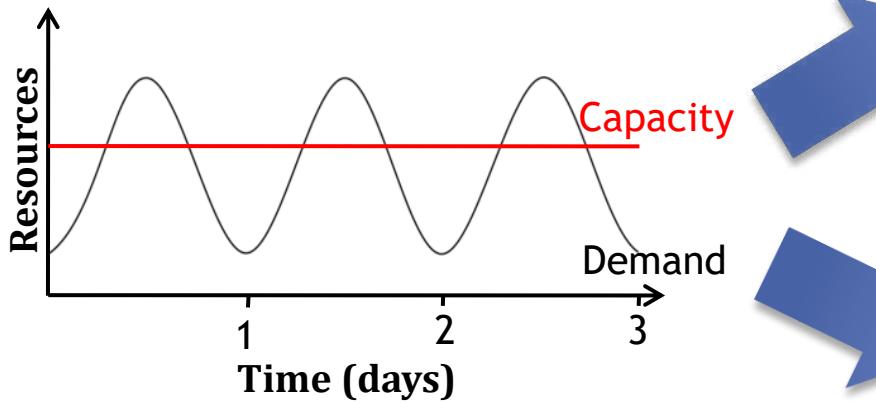
- ▶ What is scalability ?
  - ▶ A desirable property of a system, a network, or a process, which indicates its ability to either handle growing amounts of work in a graceful manner or to be readily enlarged.
- ▶ What is elasticity ?
  - ▶ The ability to grow or shrink infrastructure resources dynamically as needed to adapt to workload changes in an autonomic manner, maximizing the use of resources..
- ▶ But how to achieve these properties ?
  - ▶ Dynamic provisioning
  - ▶ Multi-tenant design

# Dynamic Provisioning

- ▶ Dynamic Provisioning is a simplified way to explain a complex networked server computing environment where server computing instances are provisioned or deployed from a administrative console or client application by the server administrator, network administrator, or any other enabled user.

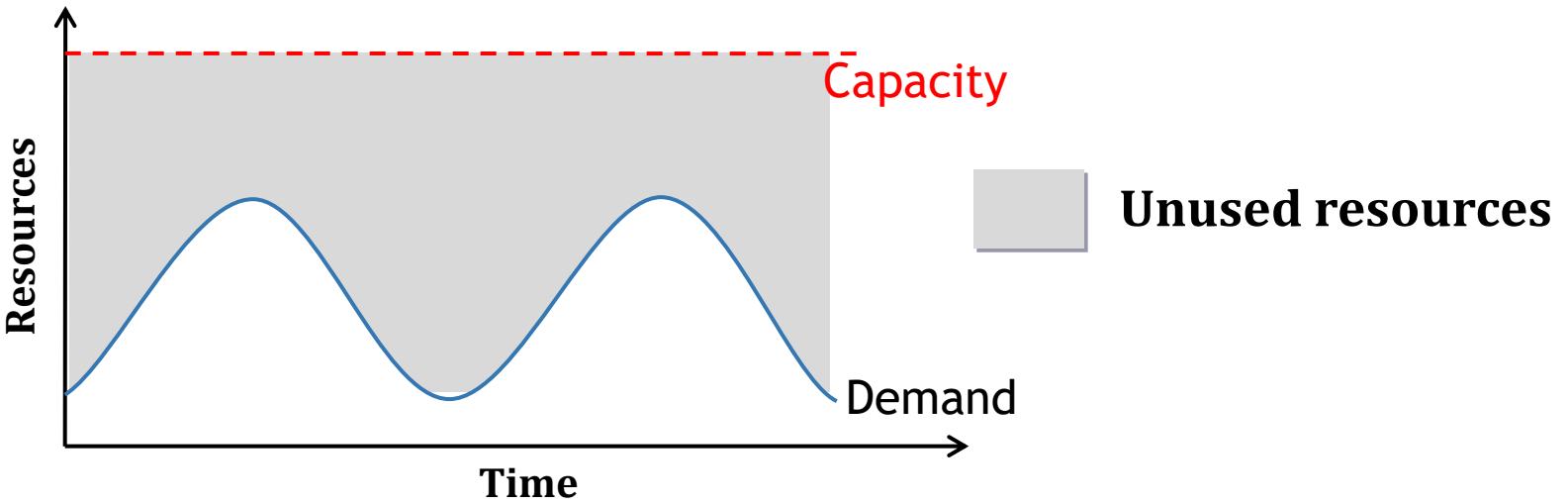
# Dynamic Provisioning

- ▶ In traditional computing model, two common problems :
  - ▶ Underestimate system utilization which result in under provision



# Dynamic Provisioning

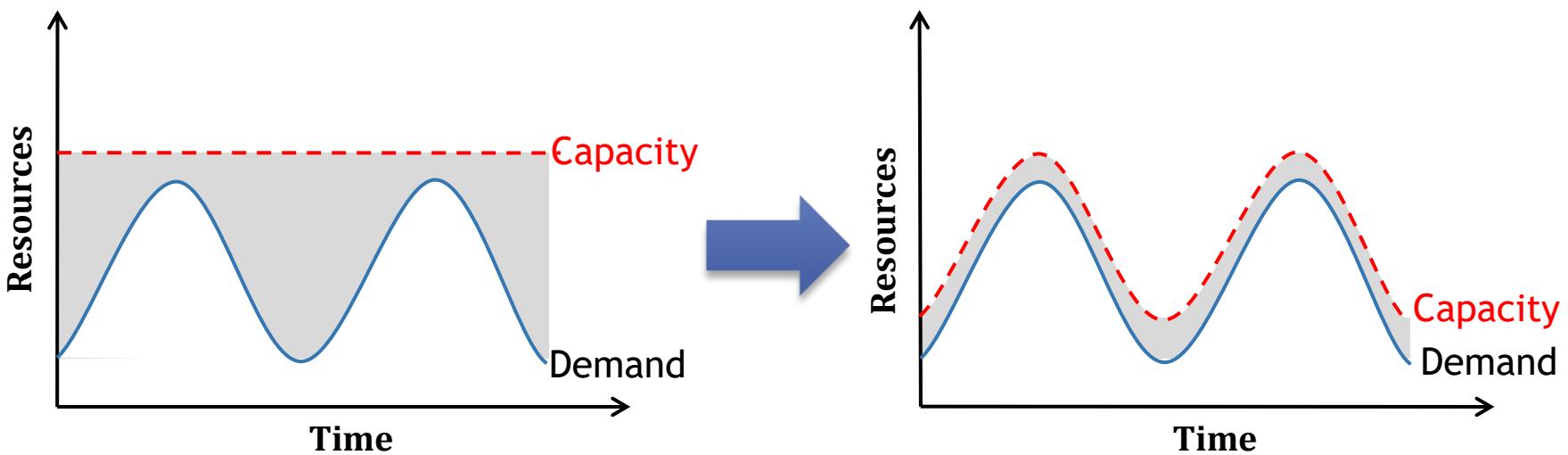
- ▶ Overestimate system utilization which result in low utilization



- ▶ How to solve this problem ??
- ▶ Dynamically provision resources

# 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.
  - ▶ 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 oriented requirements :
  - ▶ Customization
    - ▶ Multi-tenant applications are typically required to provide a high degree of customization to support each target organization's needs.
  - ▶ Quality of service
    - ▶ Multi-tenant applications are expected to provide adequate levels of security and robustness.

# Availability & Reliability

- ▶ What is 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.
  - ▶ Cloud system usually require high availability
    - ▶ Ex. “Five Nines” system would statistically provide 99.999% availability
- ▶ What is reliability ?
  - ▶ The ability of a system or component to perform its required functions under stated conditions for a specified period of time.
- ▶ But how to achieve these properties ?
  - ▶ Fault tolerance system
  - ▶ Require system resilience
  - ▶ Reliable system 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

# System Resilience

- ▶ 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.
- ▶ Some risk events
  - ▶ If power is lost at a plant for two days, can our system recover ?
  - ▶ If a key service is lost because a database corruption, can the business recover ?

# System Security

- ▶ Security issue in Cloud Computing :
  - ▶ 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.

# Manageability & Interoperability

# Manageability & Interoperability

- ▶ What is manageability ?
  - ▶ Enterprise-wide administration of cloud computing systems. Systems manageability is strongly influenced by network management initiatives in telecommunications.
- ▶ What is 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.
- ▶ But how to achieve these properties ?
  - ▶ System control automation
  - ▶ System state monitoring

# Control Automation

- ▶ What is 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.
- ▶ Architectural framework :
  - ▶ Composed by Autonomic Components (AC) which will interact with each other.
  - ▶ An AC can be modeled in terms of two main control loops (local and global) with sensors (for self-monitoring), effectors (for self-adjustment), knowledge and planer/adapter for exploiting policies based on self- and environment awareness.

# System Monitoring

- ▶ What is system monitor ?
  - ▶ 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
- ▶ Anything more ?
  - ▶ Billing system

# Billing System

- ▶ 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. And automatically request money from use's banking account.

# Accessibility & Portability



**Anyone !  
Anytime !  
Anywhere !**



# Accessibility & Portability

- ▶ What is 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.
- ▶ What is service portability ?
  - ▶ Service portability is the ability to access services using any devices, anywhere, continuously with mobility support and dynamic adaptation to resource variations.
- ▶ But how to achieve these properties ?
  - ▶ Uniform access
  - ▶ Thin client

# Uniform Access

- ▶ How do users access cloud services ?
  - ▶ Cloud provider should provide their cloud service 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 service take this into concern, and delivery their services with web-based interface through the Internet.





# Thin Client

- ▶ 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.

# Benefits of Cloud Computing

# Reduce 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

# Reduce Initial Investment

- ▶ Initiate business with Cloud Computing services :
  - ▶ Survey and analysis the industry and market
  - ▶ Chose 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 Initial Investment

- ▶ What does cloud computing achieve ?

	Traditional	With Cloud Computing
<b>Investment Risk</b>	<i>Enterprise takes the risk</i>	<i>Cloud reduces the risk</i>
<b>Infrastructure</b>	<i>Enterprise owns the infrastructure</i>	<i>Cloud provider owns the infrastructure</i>
<b>Time duration</b>	<i>Long deployment time</i>	<i>Fast to business ready</i>

# 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

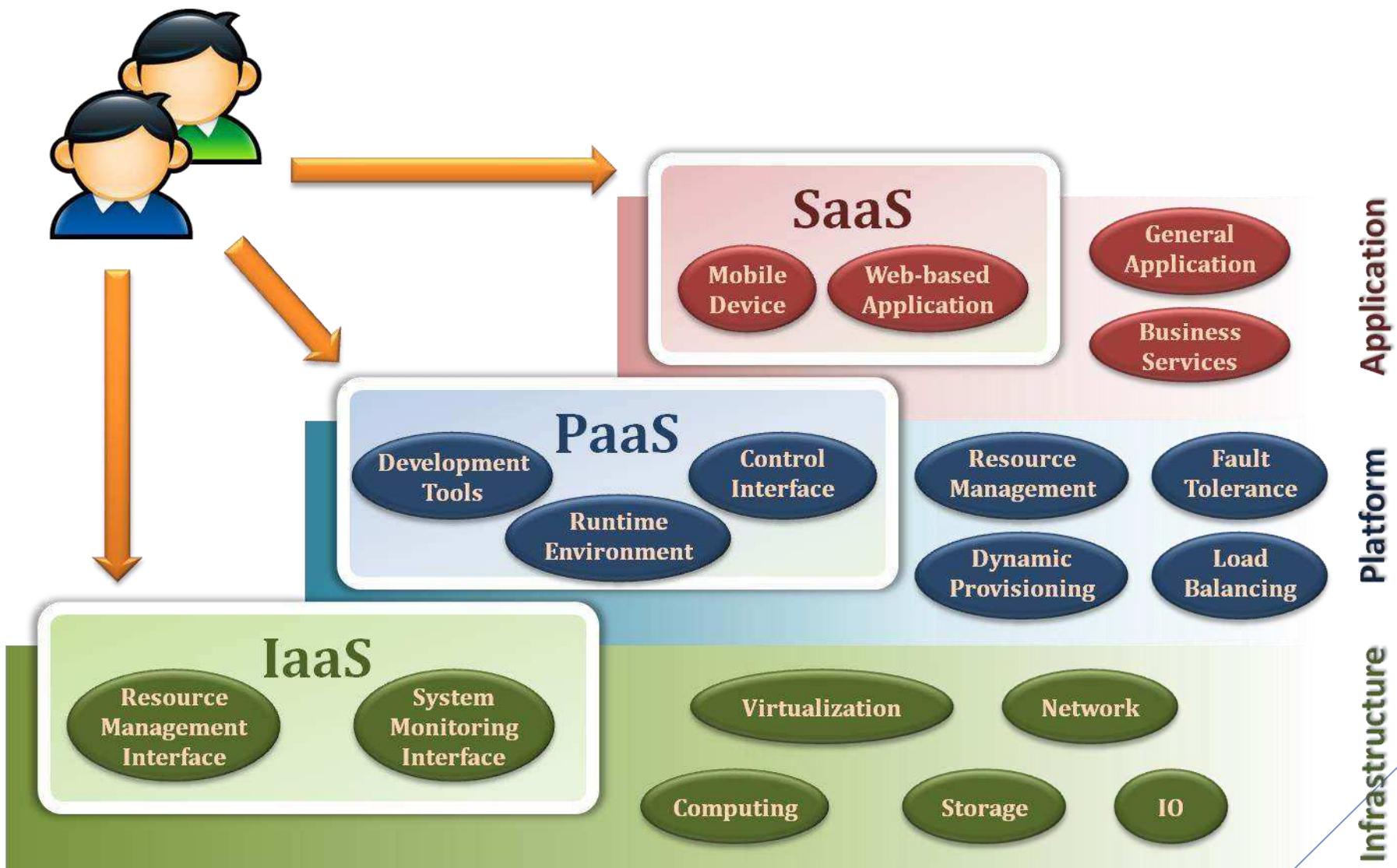
# Reduce Capital Expenditure

- ▶ What dose cloud computing achieve ?

	Traditional	With Cloud Computing
<b>Business focus</b>	<i>Need to own its IT department</i>	<i>Cloud provider takes care everything</i>
<b>Payment</b>	<i>Pay for all investment and human resource</i>	<i>Enterprise pays as the service used</i>
<b>Time duration</b>	<i>Long establish time</i>	<i>Fast to business ready</i>

# Service Models

# Service Model Overview

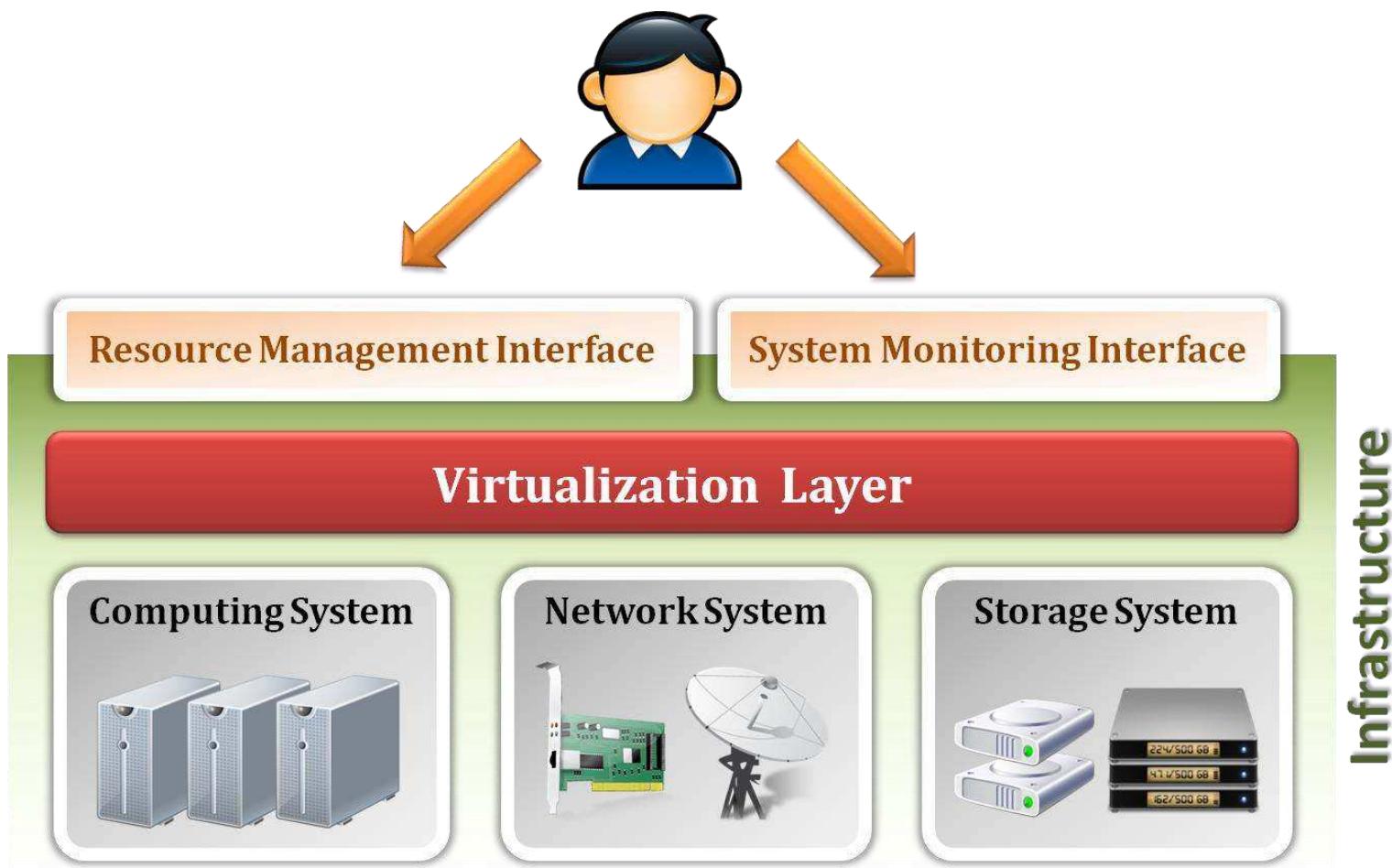


# Infrastructure as a Service - IaaS

- ▶ 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
  - ▶ ... etc

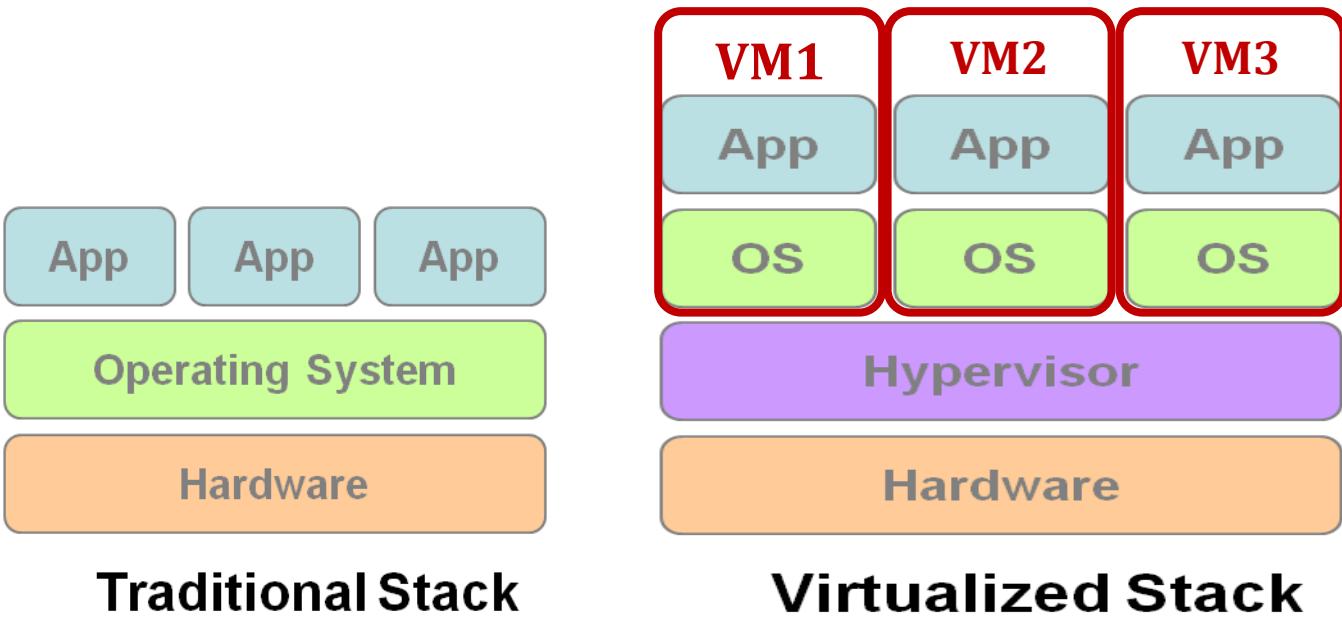
# Infrastructure as a Service

- ▶ System architecture :



# Infrastructure as a Service

- ▶ Enabling technique - *Virtualization*
  - ▶ Virtualization is an abstraction of logical resources away from underlying physical resources.
    - ▶ Virtualization technique shift OS onto hypervisor.
    - ▶ Multiple OS share the physical hardware and provide different services.
    - ▶ Improve utilization, availability, security and convenience.

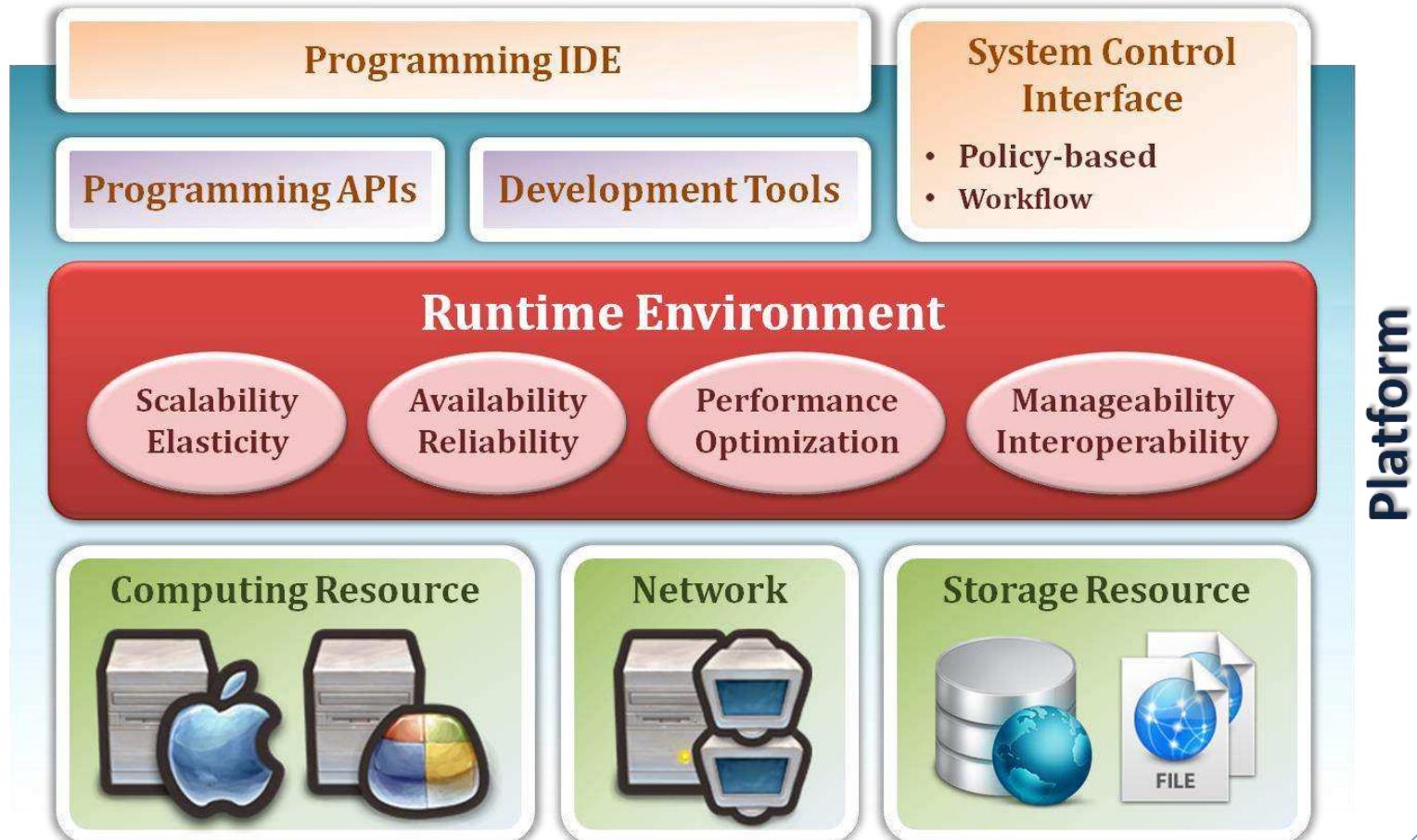


# Platform as a Service - PaaS

- ▶ 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
  - ▶ ... etc

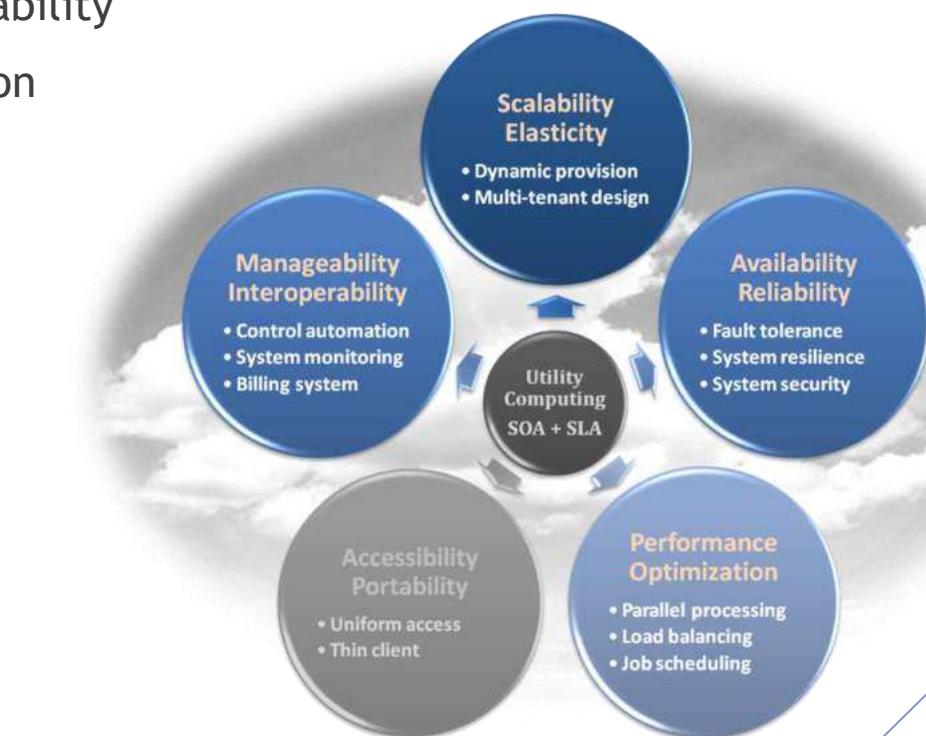
# Platform as a Service

## ► System architecture :



# Platform as a Service

- ▶ Enabling technique - **Runtime Environment Design**
  - ▶ Runtime environment refers to collection of software services available.  
Usually implemented by a collection of program libraries.
- ▶ Common properties in Runtime Environment :
  - ▶ Manageability and Interoperability
  - ▶ Performance and Optimization
  - ▶ Availability and Reliability
  - ▶ Scalability and Elasticity



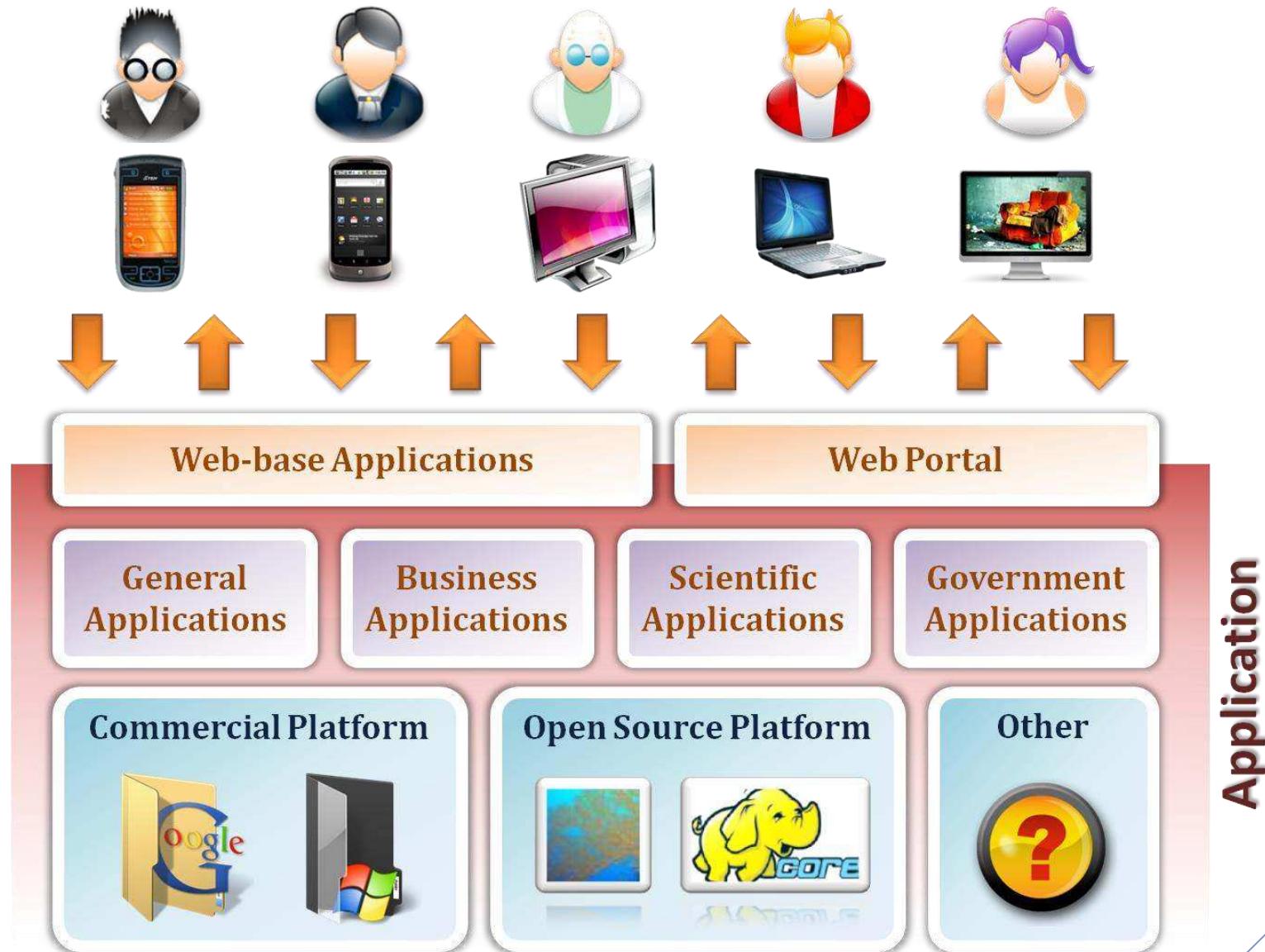
# Platform as a Service

- ▶ Provide service - **Programming IDE**
  - ▶ Users make use of programming IDE to develop their service 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.
  - ▶ The programming APIs supported from runtime environment may be various between different cloud providers, but there are still some common operating functions.
    - ▶ Computation, storage and communication resource operation

# Software as a Service - SaaS

- ▶ 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 :
  - ▶ Google Apps (e.g., Gmail, Google Docs, Google sites, ...etc)
  - ▶ SalesForce.com
  - ▶ EyeOS
  - ▶ ... etc

# Software as a Service



# Software as a Service

- ▶ Provide service - Web-based Applications
  - ▶ Conventional applications should translate their access interface onto web-based platform.
  - ▶ Applications in different domains
    - ▶ General Applications - Applications which are designed for general propose, such as office suit, multimedia and instant message, ...etc.
    - ▶ Business Applications - Application which are designed for business propose, such as ERP, CRM and market trading system, ...etc.
    - ▶ Scientific Applications - Application which are designed for scientific propose, such as aerospace simulation and biochemistry simulation, ...etc.
    - ▶ Government Applications - Applications which are designed for government propose, such as national medical system and public transportation system service, ...etc.

# Deployment models

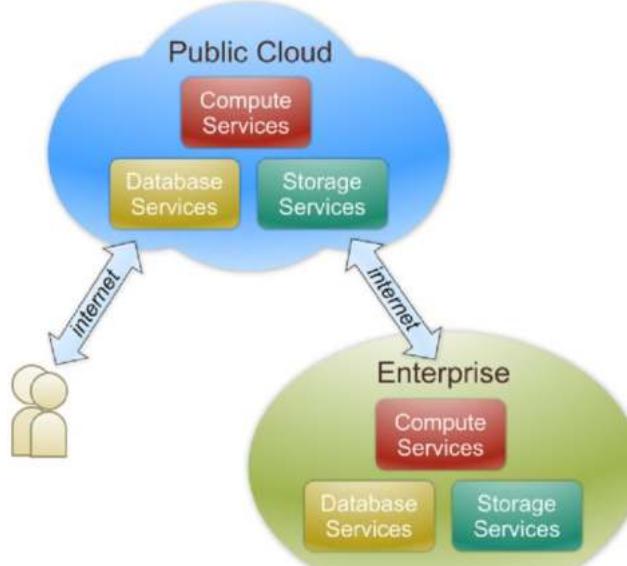
How to deploy a cloud system ?

# Deployment Model

- ▶ There are four primary cloud deployment models :
  - ▶ Public Cloud
  - ▶ Private Cloud
  - ▶ Community Cloud
  - ▶ Hybrid Cloud
- ▶ Each can exhibit the previously discussed characteristics; their differences lie primarily in the scope and access of published cloud services, as they are made available to service consumers.

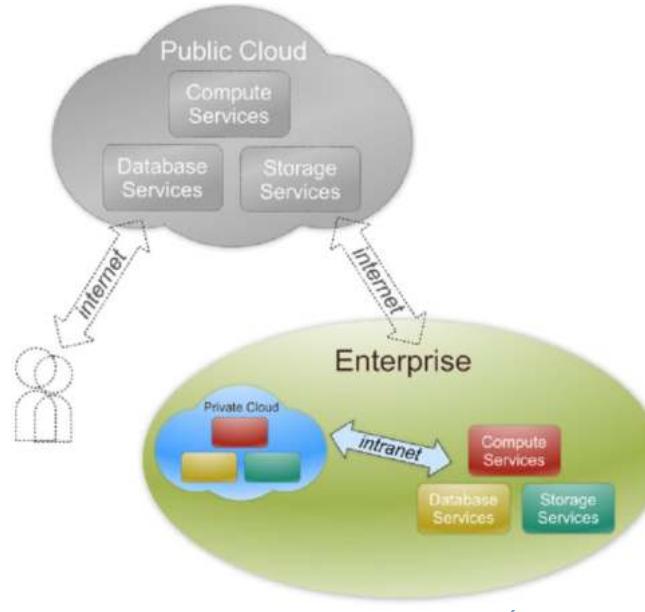
# Public Cloud

- ▶ 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.
  - ▶ Basic characteristics :
    - ▶ Homogeneous infrastructure
    - ▶ Common policies
    - ▶ Shared resources and multi-tenant
    - ▶ Leased or rented infrastructure
    - ▶ Economies of scale



# 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.
  - ▶ Basic characteristics :
    - ▶ Heterogeneous infrastructure
    - ▶ Customized and tailored policies
    - ▶ Dedicated resources
    - ▶ In-house infrastructure
    - ▶ End-to-end control



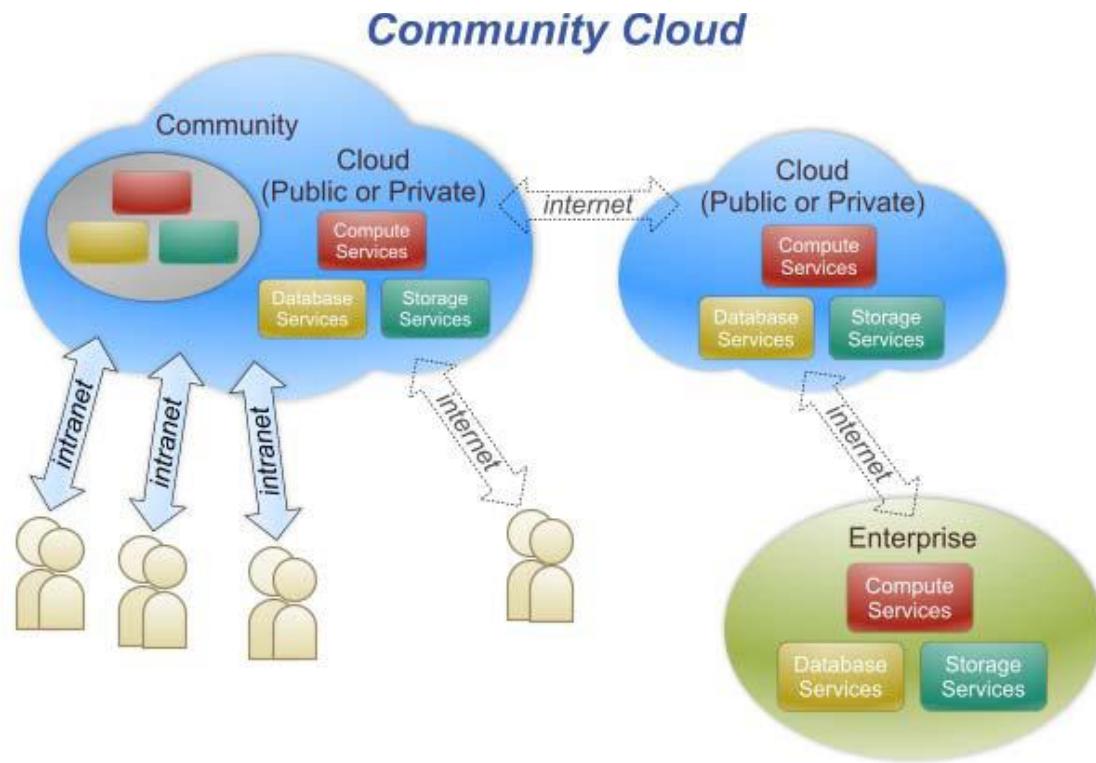
# Public vs. Private

- ▶ Comparison :

	Public Cloud	Private Cloud
<b><i>Infrastructure</i></b>	<i>Homogeneous</i>	<i>Heterogeneous</i>
<b><i>Policy Model</i></b>	<i>Common defined</i>	<i>Customized &amp; Tailored</i>
<b><i>Resource Model</i></b>	<i>Shared &amp; Multi-tenant</i>	<i>Dedicated</i>
<b><i>Cost Model</i></b>	<i>Operational expenditure</i>	<i>Capital expenditure</i>
<b><i>Economy Model</i></b>	<i>Large economy of scale</i>	<i>End-to-end control</i>

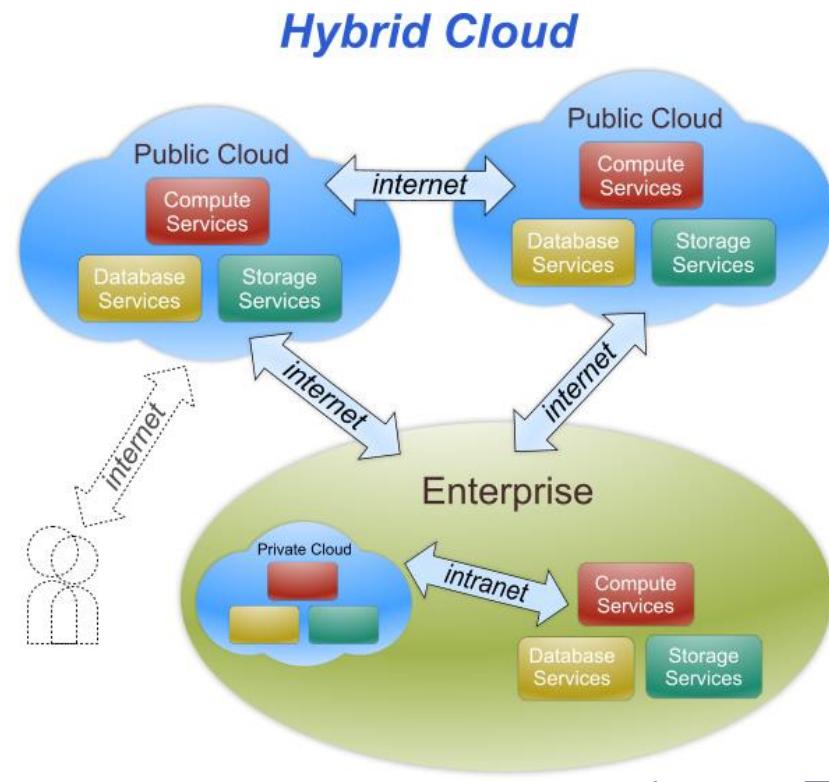
# Community Cloud

- ▶ 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).

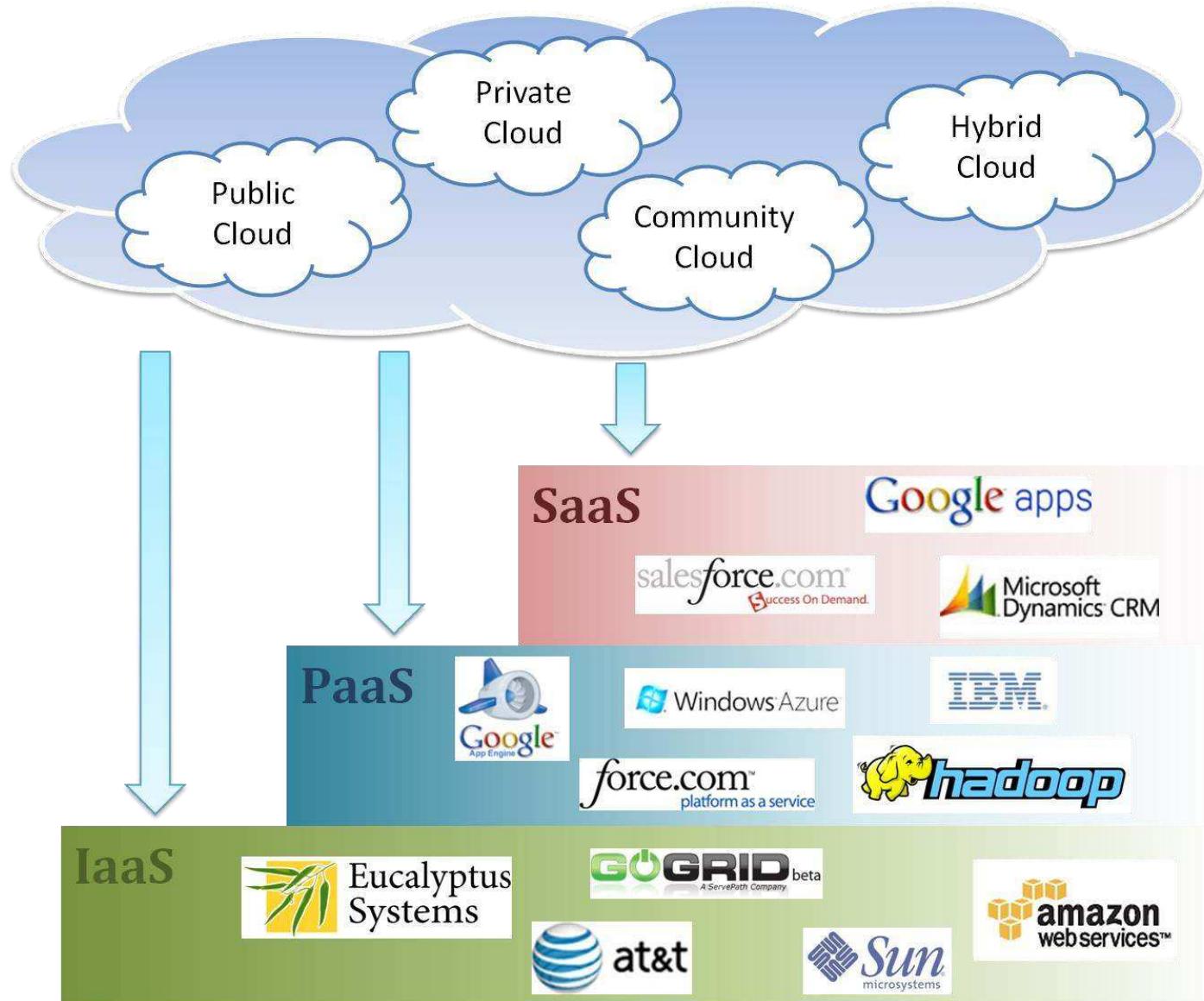


# Hybrid Cloud

- ▶ 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).



# Cloud Ecosystem

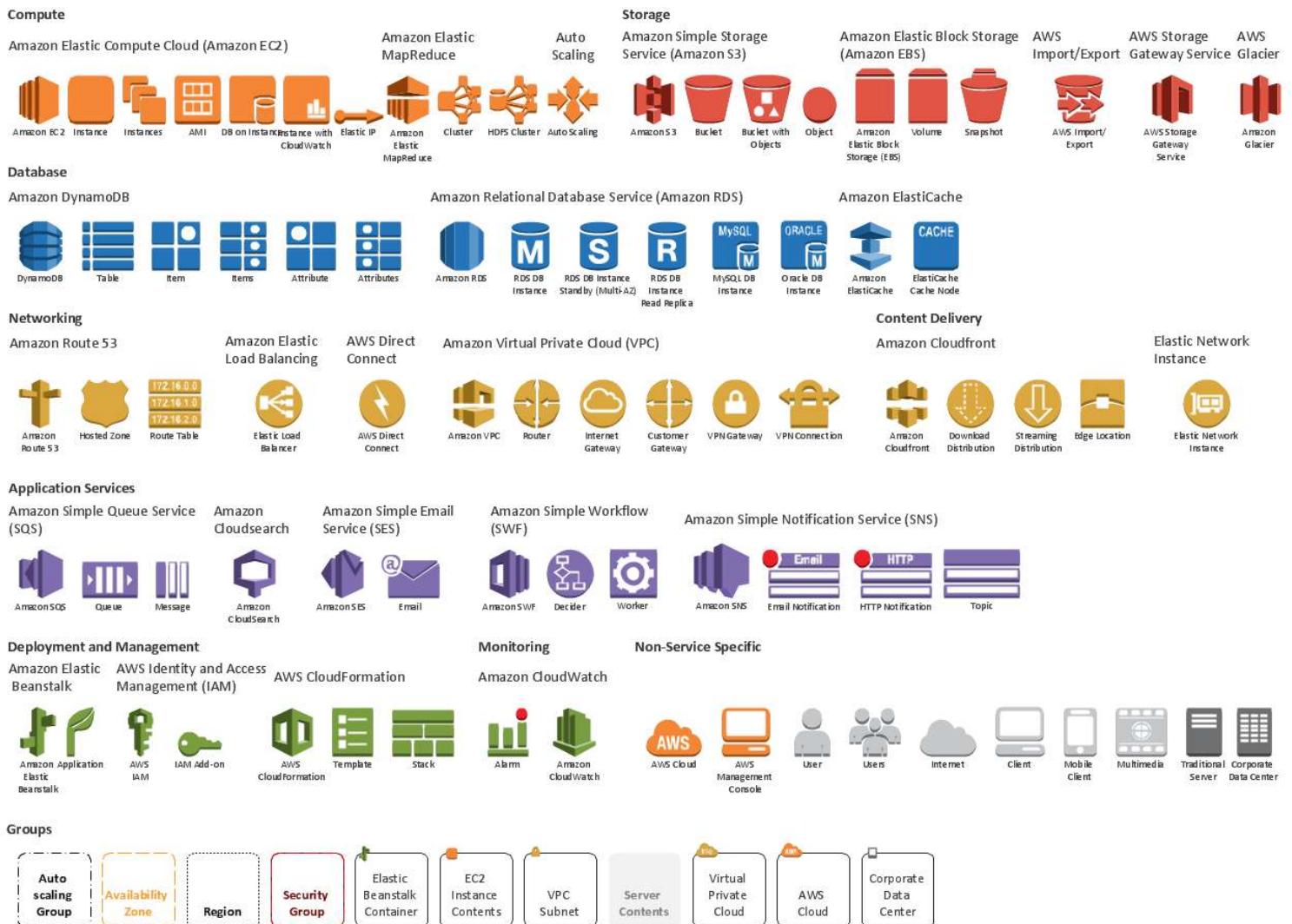


# Cloud Providers

# AWS

- ▶ Elastic Compute Cloud - EC2 (IaaS)
- ▶ Simple Storage Service - S3 (IaaS)
- ▶ Elastic Block Storage - EBS (IaaS)
- ▶ SimpleDB (SDB) (PaaS)
- ▶ Simple Queue Service - SQS (PaaS)
- ▶ CloudFront (S3 based Content Delivery Network - PaaS)
- ▶ Consistent AWS Web Services API

# AWS Ecosystem



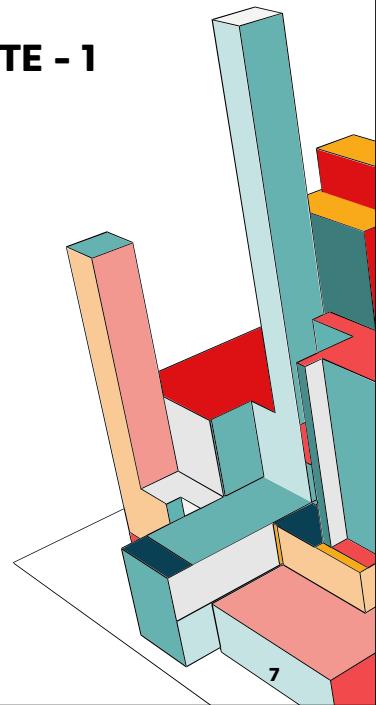
# What does Azure platform offer to developers?



## PROGETTAZIONE DI UN INTERFACCIA UTENTE - 1

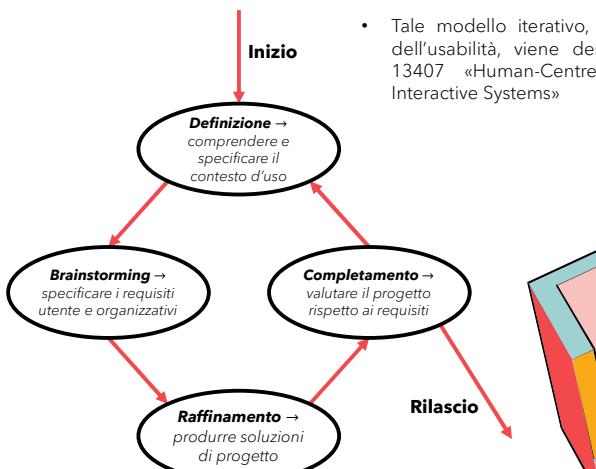
- Per riuscire ad avviare la progettazione di un interfaccia utente è fondamentale conoscere:
  - L'idea
  - Gli utenti
  - Come fare una verifica dell'usabilità
- In particolare, la verifica dell'usabilità deve essere ripetuta per ogni fase della progettazione, non solo quando il prodotto è finito.
- Ogni fase ha il seguente schema generale:**
  - Si tengono riunioni con il team di lavoro
  - Al termine di queste si realizzano gli output
  - Si discute e si valuta sulla qualità degli output prodotti (verifica)
  - Si itera fino a quando non si raggiunge la soglia di accettabilità concordata
  - Si passa alla fase successiva
- Le fasi di progettazione di un interfaccia utente sono le seguenti:**
  - DEFINIZIONE** (dell'idea e degli utenti) → documento di specifica dei requisiti
  - BRAINSTORMING** → prototipazione visuale di bassa qualità (es. mockup su carta)
  - RAFFINAMENTO** → prototipazione visuale di alta qualità (es. mockup funzionanti e testabili)
  - COMPLETAMENTO** → sistemi completi che possono entrare in fase di Alpha / Beta testing (verifica dell'usabilità sul campo)

12/03/2022



7

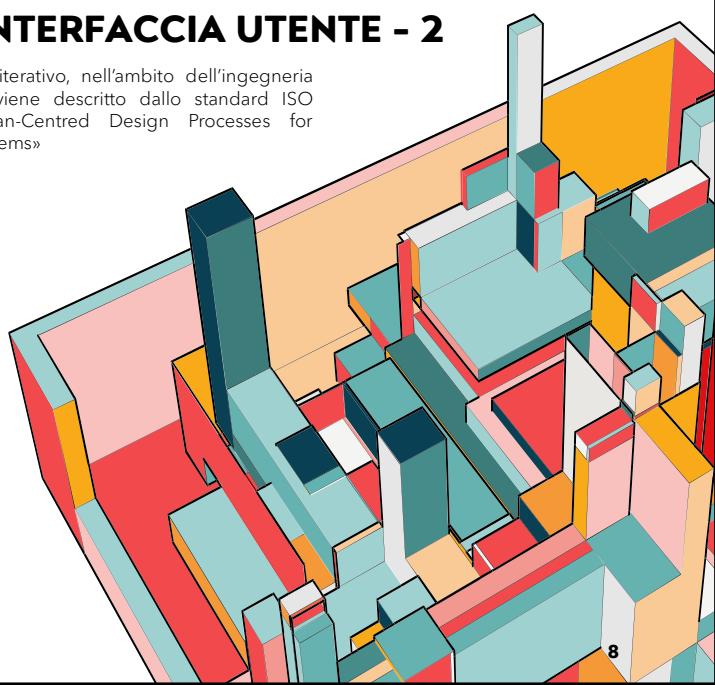
## PROGETTAZIONE DI UN INTERFACCIA UTENTE - 2



- Tale modello iterativo, nell'ambito dell'ingegneria dell'usabilità, viene descritto dallo standard ISO 13407 «Human-Centred Design Processes for Interactive Systems»

- In particolare, anche la fase di Rilascio può essere iterativa (es. Alpha Test, Beta Test, sviluppo di nuovi aggiornamenti, ...)

12/03/2022

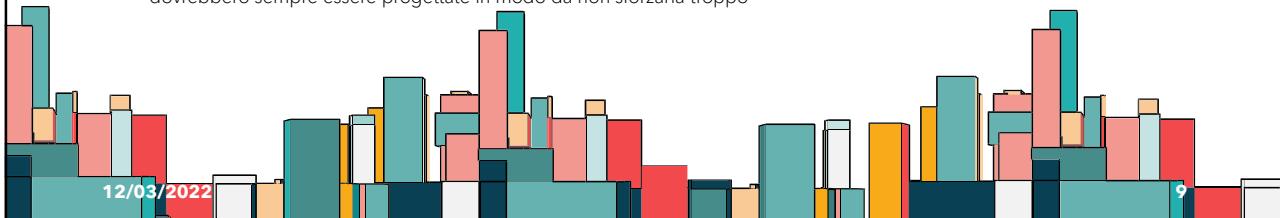


8

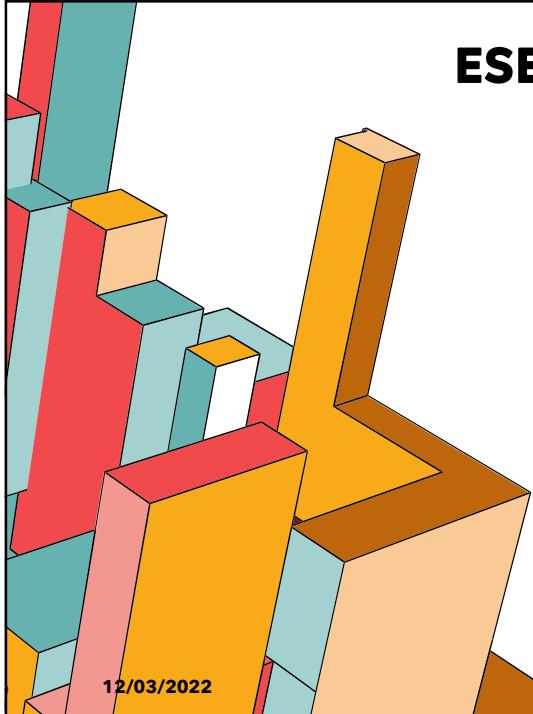
## PROGETTAZIONE DI UN'INTERFACCIA UTENTE - 3

- **Le 8 regole d'oro di Ben Shneiderman:**

1. **STRIVE FOR CONSISTENCY:** è consigliato utilizzare icone, colori e menù familiari. In questo modo gli utenti già sanno come utilizzare l'interfaccia senza neanche averla mai aperta
2. **ENABLE FREQUENT USERS TO USE SHORTCUTS:** la presenza di scorciatoie (shortcuts) permetterà agli utenti più esperti e frequenti di velocizzare il modo in cui svolgono le loro attività quotidiane
3. **OFFER INFORMATIVE FEEDBACK:** per ogni azione o componente dell'interfaccia, dovrebbe sempre esserci un feedback «human-readable» visibile per un certo lasso di tempo (es. mostrare la percentuale di caricamento)
4. **DESIGN DIALOGUE TO YIELD CLOSURE:** è consigliato inserire sempre un feedback di chiusura quando un utente termina un'operazione (es. schermata di ringraziamento dopo un acquisto)
5. **OFFER SIMPLE ERROR HANDLING:** un sistema dovrebbe sempre essere progettato a prova di stupido, ma nel caso in cui compaia in errore è consigliato fornire all'utente istruzioni semplici e spiegate passo per passo per risolverlo
6. **PERMIT EASY REVERSAL OF ACTION:** introdurre la possibilità di permettere agli utenti di ripristinare facilmente le loro azioni allevia la loro ansia di imbattersi in situazioni mai viste prima
7. **SUPPORT INTERNAL LOCUS OF CONTROL:** le interfacce utente devono essere progettate dal punto di vista dell'utente, quindi facendo in modo che abbiano sempre loro il controllo e che il sistema si comporti secondo le loro aspettative
8. **REDUCE SHORT TERM MEMORY LOAD:** la memoria a breve termine delle persone, nella maggior parte dei casi, impedisce loro di memorizzare codici, istruzioni, ecc. nel passaggio da una schermata all'altra. Per questo motivo, le interfacce dovrebbero sempre essere progettate in modo da non sforzarla troppo



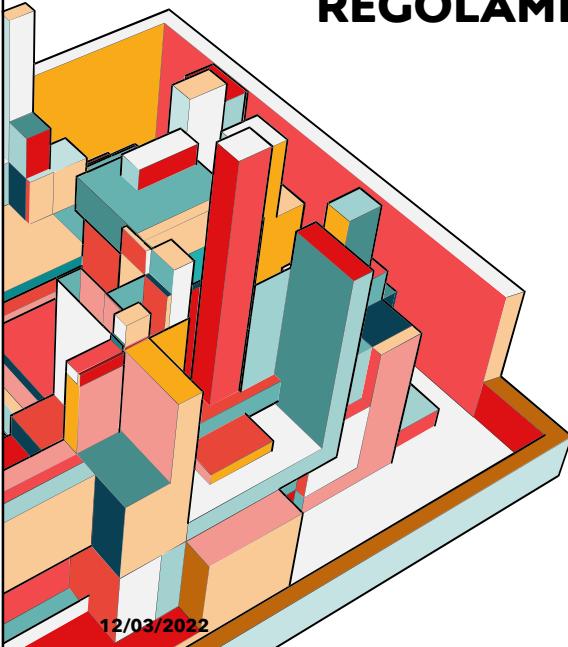
## ESEMPIO DI PROGETTAZIONE



• Il processo iterativo user-centred di un sito web semplice si può sviluppare in **7 macrofasi principali, con 5 prototipi principali**, ciascuno dei quali ha tecniche realizzative e finalità specifiche:

1. **Definizione dei requisiti** → documento dei requisiti
2. **Avviamento del progetto** → piano di qualità
3. **Web design** → prototipo di navigazione (fornire un'enorme ridondanza per accedere alle stesse sezioni favorisce l'utilizzo del sito a tutti i tipi di utente)
4. **Visual design** → prototipo di comunicazione
5. **Sviluppo** → prototipo funzionale
6. **Redazione dei contenuti** → prototipo editoriale
7. **Pubblicazione e gestione del sito** → sistema on-line

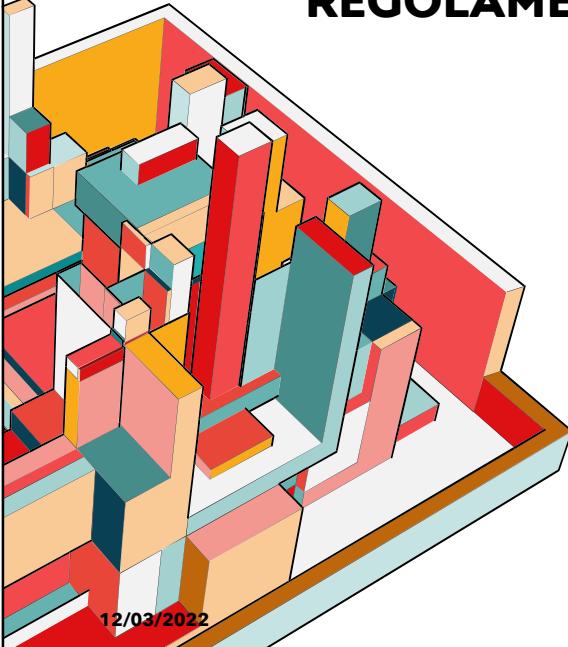
## REGOLAMENTARE LA CREATIVITÀ - 1



- La progettazione di un sistema è una fase estremamente creativa.
- Esistono diversi processi che ci permettono di passare da una descrizione testuale di bisogni e vincoli (documento dei requisiti) all'invenzione del sistema che rispetti tali bisogni e vincoli (design concept). Gli approcci possibili sono:
  - **MIMESI:** è un processo che modifica delle caratteristiche di un prodotto già esistente per generare un nuovo prodotto (es. da calcolatrice fisica ad app della calcolatrice)
  - **IBRIDAZIONE:** è un processo che unifica due prodotti già esistenti per crearne uno nuovo (es. mouse con pulsanti e laser per le presentazioni oppure Maps che integra sia le mappe sia i punti di interesse)
  - **METAFORA:** è un processo nel quale si prende un prodotto appartenente ad un certo campo semantico (donatore) e si trasporta la sua struttura in un campo semantico completamente diverso (ricevente)
    - L'esempio più tipico è quello del trasporto dei concetti di finestra, desktop e bottone al campo semantico delle interfacce di un computer

11

## REGOLAMENTARE LA CREATIVITÀ - 2

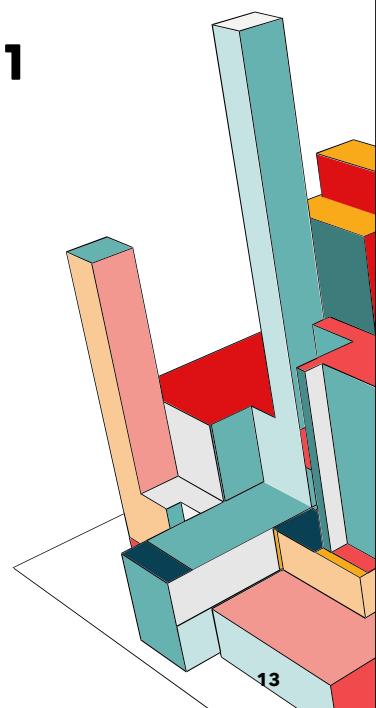
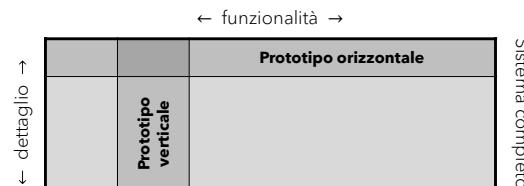


- Gli approcci possibili sono:
  - **VARIAZIONE (approccio evolutivo):** è un processo che permette di evolvere un prodotto già esistente aggiungendo, rimuovendo o semplicemente spostando una componente dalla sua struttura creando, così, una variante
  - **COMPOSIZIONE:** è un processo in cui si prende spunto dall'esperienza sviluppata in altri progetti al fine di svilupparla ulteriormente adattandola a nuovi contesti. Infatti, il termine «design pattern» indica una soluzione generale ad un problema di progettazione che si ripropone in tanti ambiti e situazioni diversi

12

# COME VALUTARE L'USABILITÀ - 1

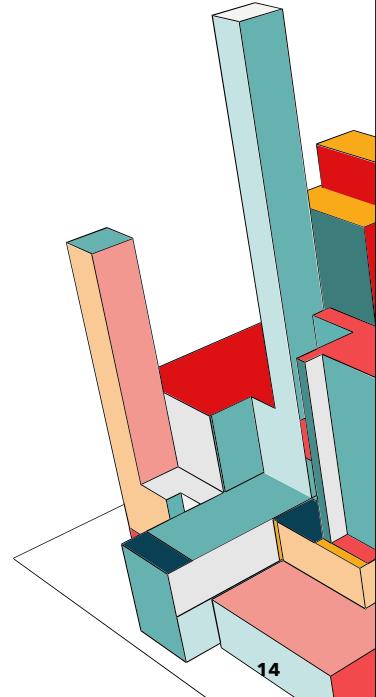
- È possibile misurare l'usabilità di un sistema che non è stato ancora realizzato, basandoci sulla disponibilità di prototipi
- Un **prototipo** è il progetto astratto di un sistema che realizza una o più funzionalità del **prodotto finito** e possono avere diversi gradi di dettaglio



12/03/2022

# COME VALUTARE L'USABILITÀ - VALUTAZIONI EURISTICHE

- **Valutazioni euristiche (ispezioni)**
  - Sono eseguite da esperti di usabilità con l'aiuto di regole formali, senza alcun coinvolgimento dei clienti
  - **Euristiche di Nielsen** (molti simili alle 8 regole d'oro di Ben Shneiderman)
    1. Visibilità dello stato del sistema (feedback)
    2. Corrispondenza tra mondo reale e sistema (comprendibile)
    3. Libertà e controllo da parte degli utenti (recupero dagli errori)
    4. Consistenza e standard (conoscenze pregresse)
    5. Prevenzione degli errori (progettazione solida)
    6. Riconoscere piuttosto che ricordare (evitare di sforzare la memoria a breve termine)
    7. Flessibilità ed efficienza d'uso (scorciatoie)
    8. Design minimalista ed estetico
    9. Aiutare gli utenti a riconoscere, diagnosticare e correggere gli errori
    10. Guida e documentazione (necessaria per aiutare l'utente)
  - Sono poco costose, tuttavia restituiscono risultati soggettivi, basati sull'esperienza, il punto di vista e le conoscenze del valutatore
  - Impiegando più valutatori, che analizzino in momenti separati il sistema, è possibile aumentare l'affidabilità delle risposte restituite

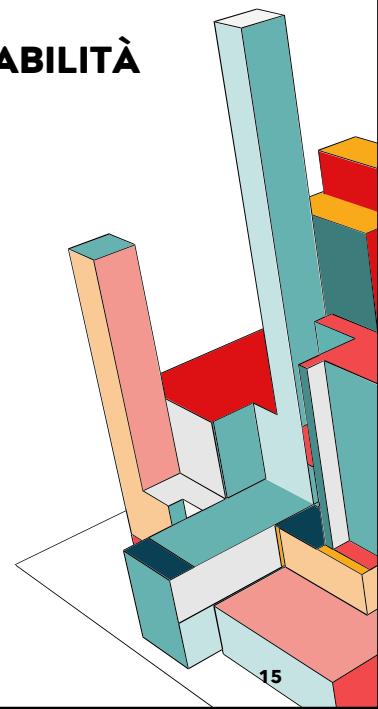


12/03/2022

## COME VALUTARE L'USABILITÀ - TEST DI USABILITÀ

- Sono eseguiti da un campione di utenti (rappresentativo del target) in un ambiente controllato (usability lab) chiedendo loro di eseguire compiti tipici dell'utilizzo del sistema
- Oltre agli utenti che provano il sistema, il test coinvolge anche
  - Clienti e stakeholders
  - Facilitatore: gestisce la regia della prova
  - Uno o più osservatori: annotano i comportamenti ritenuti «significativi» degli utenti (devono conoscere il sistema alla perfezione)
- Ha lo scopo di identificare indicazioni concrete per migliorare il sistema, per rimuovere le difficoltà attraverso la tecnica «think aloud» (l'utente esprime ad alta voce tutti i pensieri che balzano in mente mentre usa il sistema)
- In particolare, è consigliato registrare audio / video degli utenti che provano il sistema, in modo da rivedere in seguito tutto ciò che è accaduto durante la prova
  - Espressioni facciali
  - Tocchi sul prototipo
- Gli utenti scelti per l'esecuzione del test di usabilità possono essere scelti in base al
  - livello di conoscenza del dominio applicativo
  - livello di familiarità con la tecnologia utilizzata

12/03/2022

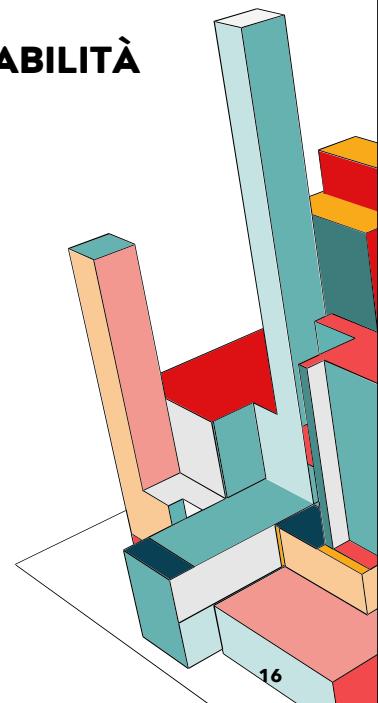


15

## COME VALUTARE L'USABILITÀ - TEST DI USABILITÀ

- Si possono classificare rispetto al momento in cui sono eseguiti:
  - **Formativi:** sono eseguiti durante il ciclo iterativo di progettazione, per sottoporre i diversi prototipi prodotti a prove d'uso con gli utenti. Lo scopo è quello di individuare il maggior numero di problemi e difetti
  - **Sommativi:** sono eseguiti al termine del processo di progettazione. Sono test più completi che non hanno lo scopo di dare nuove indicazioni ai progettisti, così come valutare il sistema, i suoi pregi, difetti e caratteristiche
- Si possono classificare rispetto alle attività condotte dagli utenti:
  - **Test di compito:**
    - Gli utenti svolgono singoli compiti corrispondenti a singole funzionalità del sistema
    - Possono essere eseguiti anche quando il sistema non è completamente pronto
    - È importante non dare all'utente troppe istruzioni, altrimenti i problemi di usabilità non emergeranno
  - **Test di scenario:**
    - Viene indicato agli utenti un obiettivo da raggiungere attraverso una serie di compiti elementari, senza essere indicati esplicitamente
    - Permette agli utenti di seguire una strategia soggettiva, sulla base delle preferenze e abitudini personali

12/03/2022



16

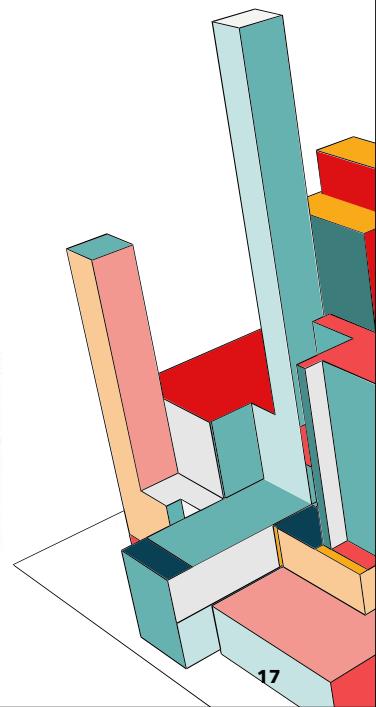
## COME VALUTARE L'USABILITÀ - MISURE

- Tempo impiegato da ogni utente per l'esecuzione di ciascun compito
- Tasso di successo, cioè la percentuale di compiti che ogni utente riesce a portare a termine
  - Il calcolo del tasso di successo può essere realizzato mediante l'uso di una tabella in cui si indica se un utente ha completato con successo (S), ha fallito (F) oppure ha fallito in modo parziale (P) l'esecuzione di un compito
  - Infine, il tasso di successo può essere calcolato come il rapporto tra il numero di successi (o successi parziali) sul numero di compiti eseguiti in totale

	Compito 1	Compito 2	Compito 3	Compito 4	Compito 5	Compito 6
Utente 1	F	F	S	F	F	S
Utente 2	F	F	P	F	P	F
Utente 3	S	F	S	S	P	S
Utente 4	S	F	S	F	P	S

$$\text{Tasso di successo} = \frac{(9 * 1 + (4 * 0,5))}{24} = 46\%$$

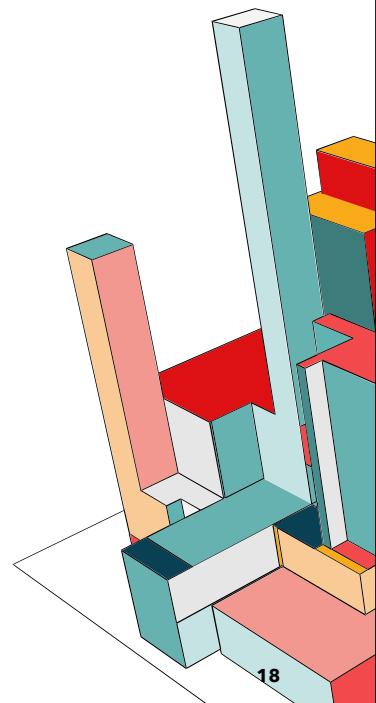
12/03/2022



## COME VALUTARE L'USABILITÀ - 2

- **Un test di usabilità viene condotto in quattro fasi successive**
  1. PIANIFICAZIONE (a inizio progetto)
  2. PREPARAZIONE
  3. ESECUZIONE
  4. ANALISI E PROPOSTE
- **L'output finale di un test di usabilità è il rapporto di valutazione**
  - È un documento che descrive in modo accurato i risultati dei test effettuati, e fornisce le prove del fatto che essi siano stati eseguiti adottando metodi adeguati (es. utenti rappresentativi del target, numero di test adeguato, strumentazioni e metodologie adeguati, ...)
  - Uno schema generale per la stesura del rapporto di valutazione è il seguente
    - Identificazione del documento (nomi autori, data e versione) e sommario
    - Prodotto valutato (breve descrizione del prodotto)
    - Obiettivi della valutazione
    - Metodologia utilizzata (es. numero utenti, compiti assegnati, euristiche, ...)
    - Sintesi delle misure (tabelle) e delle interviste agli utenti
    - Analisi dei risultati
    - Raccomandazioni (descrizione testuale dei miglioramenti da eseguire a seguito dell'esecuzione del test) e allegati

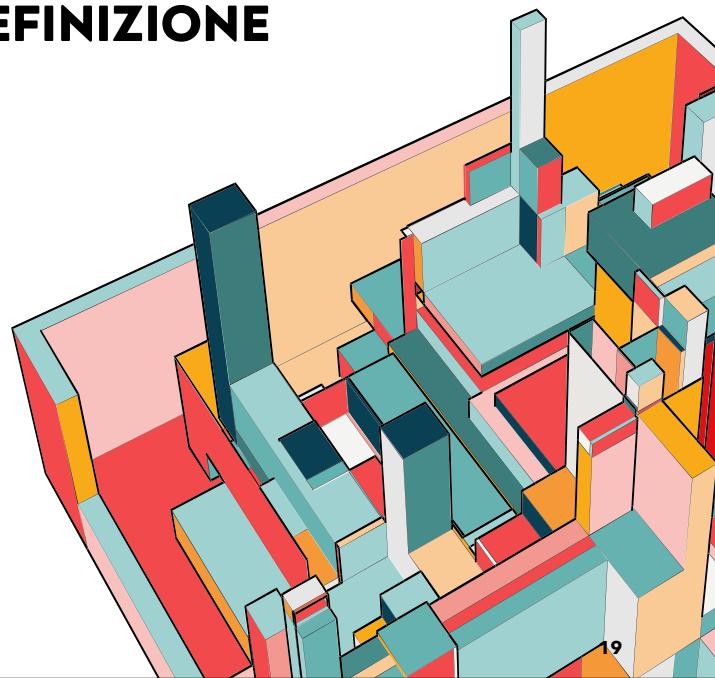
12/03/2022



## STATECHARTS - DEFINIZIONE

- Sono diagrammi utilizzati per descrivere il comportamento di un singolo oggetto in diversi casi d'uso
- **NON sono diagrammi utilizzati per descrivere comportamenti che coinvolgono più oggetti** che collaborano tra loro in diversi casi d'uso (activity diagrams)
- Nella programmazione ad oggetti, si disegna uno statechart per ogni singola classe al fine di mostrare il comportamento del ciclo di vita di un singolo oggetto
- È spesso utilizzato per descrivere comportamenti complessi, come le interfacce grafiche, in modo tale da facilitarne la comprensione

12/03/2022

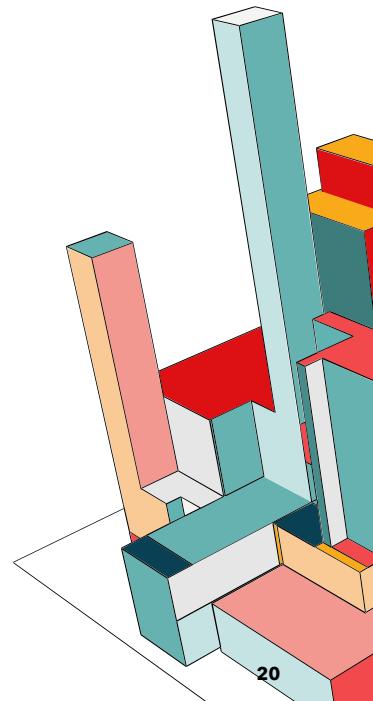


19

## STATECHARTS - TERMINOLOGIA

- **STATO / VERTICE:**
  - Rappresentano delle situazioni in cui valgono determinate condizioni statiche (il sistema è in attesa di un evento) o dinamiche (il sistema sta eseguendo un task).
  - In particolare, si devono rappresentare solo gli stati / comportamenti che coinvolgono direttamente l'oggetto che si sta studiando e non anche altri comportamenti.
  - Essenzialmente, stati diversi implicano un modo diverso di reagire agli eventi
- **TRANSIZIONE:**
  - È una freccia che interconnecta due stati. Indica il passaggio da uno stato all'altro e può essere etichettato con un'etichetta del tipo **«Trigger [Guard] / Activity»**
    - **Trigger:** è l'evento che scatena il cambiamento di stato. La mancanza di trigger è rara in uno statechart, ma più comune negli activity diagram, ed indica che la transizione deve essere eseguita immediatamente (spontanea)
    - **Guard:** è una condizione booleana che deve essere assolutamente vera, se presente, affinché la transizione si possa avviare. Se tale condizione è assente, allora la transizione viene sempre eseguita quando accade il Trigger
    - **Activity:** è un comportamento, espresso in linguaggio naturale o codice, che viene eseguito durante la transizione. La mancanza di un Activity, indica che non viene fatto nulla durante la transizione (solo il cambiamento di stato)
  - Quando accade un evento in uno stato, è possibile seguire solo una singola transizione. È, quindi, fondamentale che se si hanno molteplici transizioni con lo stesso Trigger, la Guard sia mutuamente esclusiva
  - È anche possibile avere una **self-transition**: sono transizioni che riportano allo stesso stato di partenza

12/03/2022

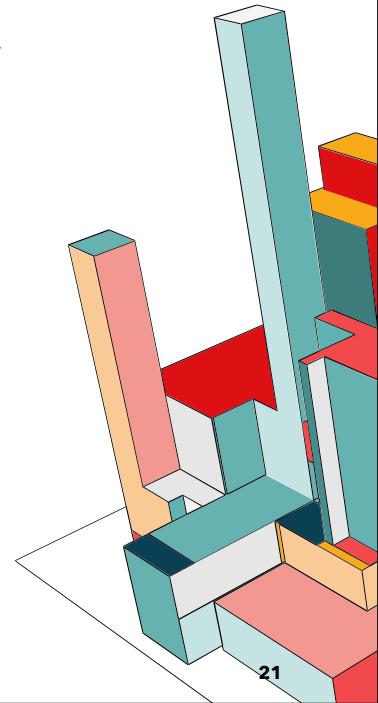


20

## STATECHARTS - TERMINOLOGIA

- **REGIONE:** è una sezione del diagramma contenente stati e transizioni. Una regione può contenere al più uno pseudo-stato iniziale ed al più uno stato finale.
- **PSEUDO-STATO INIZIALE:** uno statechart inizia sempre con lo pseudo-stato iniziale. Esso non è un vero e proprio stato, ma ha una freccia che punta al primo stato nel diagramma
- **STATO FINALE:** è lo stato che indica il completamento del ciclo di vita della macchina che si sta rappresentando (nella programmazione a oggetti, segue l'eliminazione dell'istanza dell'oggetto).
- **ATTIVITÀ INTERNE:**
  - Gli stati possono anche contenere una lista di attività interne. Un'attività interna ha la stessa sintassi delle etichette delle transizioni, quindi il Trigger indica quando viene eseguita l'Activity e la Guard è una condizione booleana. Inoltre, esistono alcuni Trigger riservati:
    - **Entry:** specifica l'Activity da eseguire quando si entra nello stato
    - **Do:** specifica l'Activity da eseguire mentre si è nello stato (si avvia dopo aver terminato tutte le Activity della transizione precedente). In particolare, una do-activity può essere interrotta da una transizione, a differenza di altri tipi di attività interne che non possono
    - **Exit:** specifica l'Activity da eseguire quando si esce dallo stato
  - Sono molto simili alle self-transitions

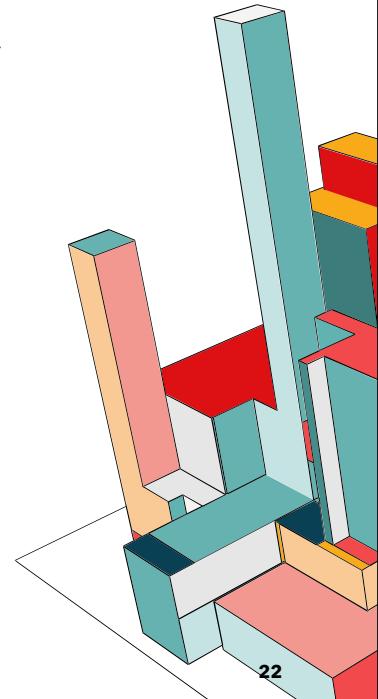
12/03/2022



## STATECHARTS - TERMINOLOGIA

- **SHALLOW HISTORY PSEUDOSTATE:**
  - È uno pseudo-stato che rappresenta lo stato attivo più di recente in uno stato composto, ma non uno dei suoi sotto-stati
  - Ce ne può essere solo uno negli stati composti, ed al più uno per ogni regione
  - Se questo pseudo-stato ha una transizione, allora si entra nello stato puntato nel caso in cui non vi sia una configurazione recente
- **DEEP HISTORY PSEUDOSTATE:** è uno pseudo-stato simile a quello precedente, ma questo indica che ripristina la configurazione più recente dell'intera regione (anche sotto-stati di sotto-stati)

12/03/2022



# STATECHARTS - TERMINOLOGIA

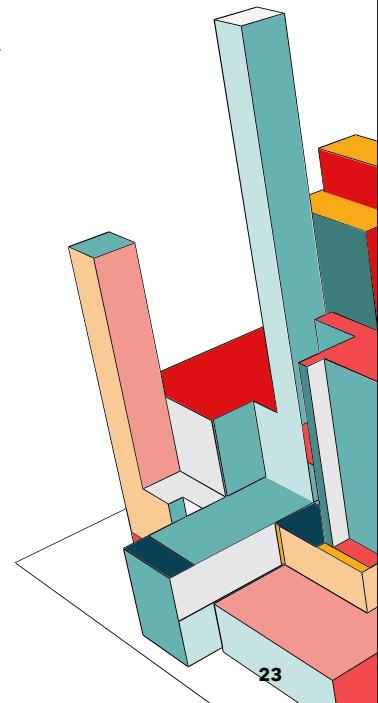
## • STATI COMPOSTI (o super-stato)

- Sono stati che definiscono al loro interno una nuova sotto-regione
- Sono utilizzati per raggruppare transizioni e attività interne comuni tra più stati. Infatti, non utilizzando un super-stato si dovrebbero realizzare transizioni e attività interne identiche per ogni sotto-stato
- In particolare, è anche possibile costruire uno stato composto contenente più sotto-regioni parallele
  - Rappresentano comportamenti che accadono nello stesso periodo di tempo
  - Si esce da uno stato composto con più regioni parallele solo quando tutte le regioni hanno raggiunto lo stato finale

## • FORK / JOIN PSEUDOSTATE:

- **Fork:** permette di dividere una singola transizione in più transizioni che entrano in stati su regioni parallele
- **Join:** permette di unire più transizioni in arrivo da stati su più regioni parallele in unica transizione

12/03/2022



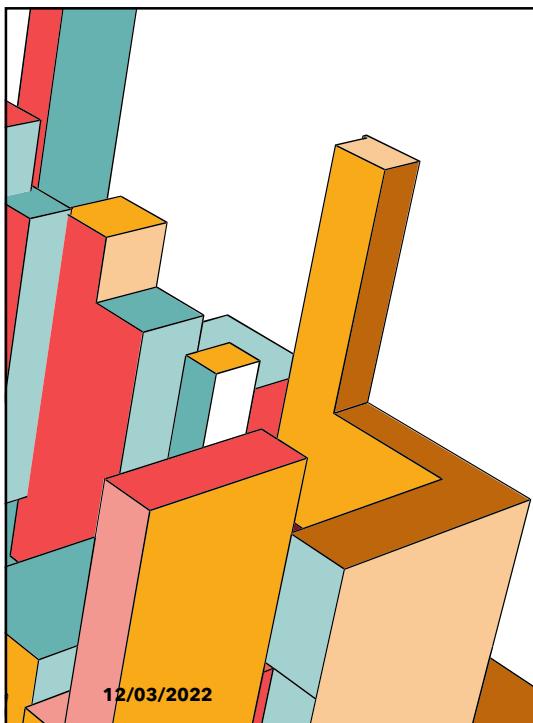
# STATECHARTS - ESEMPI

- Si descriva con uno statechart il comportamento di una generica finestra (es. minimizzata, massimizzata, modalità «in finestra», ...) in un ambiente desktop basato su finestre (es. Windows)



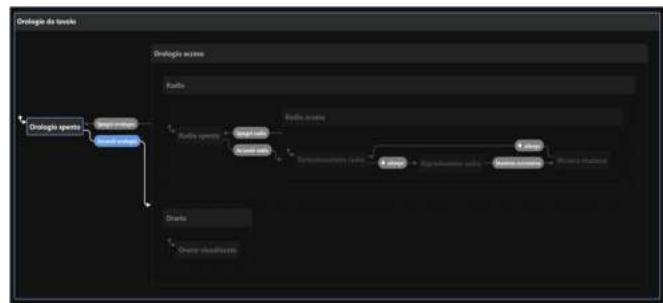
12/03/2022

24



## STATECHARTS - ESEMPI

- Un orologio da tavolo, una volta acceso, mostra l'orario corrente sul proprio display LCD e, se l'utente preme un apposito pulsante, può anche sintonizzarsi su stazioni radio e riprodurne le trasmissioni dalle casse integrate. Tramite un pulsante "next station" è possibile passare alla stazione radio successiva, che verrà riprodotta dopo una breve fase di ricerca e sintonizzazione.



12/03/2022

25

## STATECHART PER INTERFACCE GRAFICHE

- La progettazione di un prodotto software è tipicamente un processo informale e creativo che applica notazioni ed euristiche formali per esprimere soluzioni ai problemi
- Finora abbiamo visto cosa sono gli statechart nel dettaglio, senza dare però alcun consiglio su come applicare tali notazioni ai problemi di progettazione che possono insorgere durante il processo software
  - Le seguenti euristiche non sono state pensate come un insieme di passi da seguire in modo rigido; per questo motivo, possono essere applicate in qualsiasi ordine.
- Il concetto di base è che durante la progettazione si dovrebbero identificare gli stati di uno statechart partendo dal livello più astratto fino al livello più dettagliato
  - Tuttavia, bisogna sempre tener conto del comportamento degli stati più dettagliati e, quindi, un approccio completamente top-down è impossibile da applicare

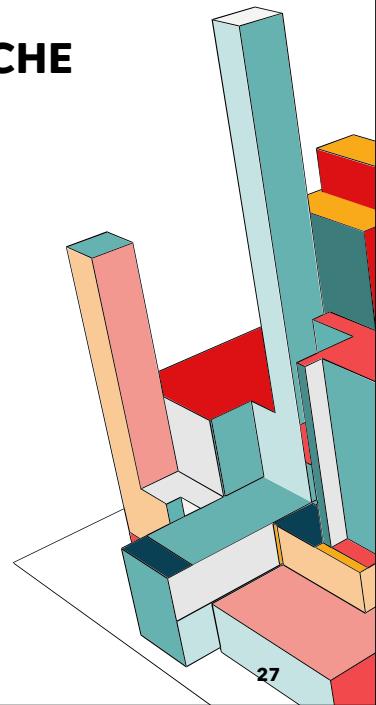
12/03/2022

26

## STATECHART PER INTERFACCE GRAFICHE

- Identificare i comportamenti più generici dell'interfaccia grafica (high-level states)
  - Progettare una schermata dell'interfaccia per volta, tenendo sempre conto dell'interazione tra due o più schermate
    - Una schermata è lo spazio sul dispositivo usato dall'utente per navigare nell'applicazione attraverso la realizzazione di eventi (es. click, swipe, ...) sulle affordance (oggetti della schermata)
  - Gli stati high-level di uno statechart corrispondono alle schermate (finestre) e popup (sottofinestre) di un'applicazione
    - Supponiamo che un'interfaccia sia fatta di 5 schermate → avremo 5 stati high-level. Ognuno di questi stati, poi, è composto da sottoregioni e altri stati che corrispondono ai popup che possono comparire in quella schermata a fronte di un evento.
    - In particolare, se un popup è specifico di una schermata, allora esso deve far parte di una sua sottoregione, altrimenti può anche essere rappresentato come stato high-level accessibile da più stati

12/03/2022



## STATECHART PER INTERFACCE GRAFICHE

- Considerare i comportamenti identificati allo step precedente in maggior dettaglio
  - Si può iniziare stilando una lista di tutte le affordance presenti sulla schermata (es. pulsanti, testi, ...) e di tutte le shortcuts che è possibile utilizzare (es. CTRL+C, ...)
- Capire se i comportamenti e affordance di un oggetto dell'interfaccia sono fissi o variabili
  - Quando si clicca un bottone, viene eseguita sempre la stessa azione, oppure l'azione dipende dal contesto attuale dell'applicazione?
  - Gli oggetti dell'interfaccia sono sempre attivi, o vi sono casi in cui sono disabilitati? I colori degli oggetti sono sempre gli stessi o cambiano?

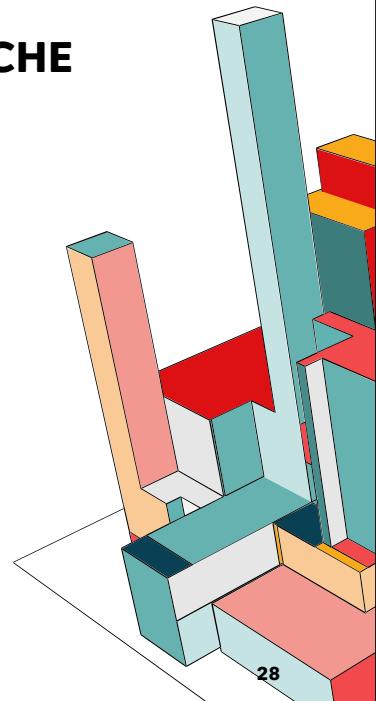
Tutte queste domande ci permettono di identificare

  - Nuovi stati appartenenti alle sottoregioni di uno stato high-level
  - Nuove transizioni tra gli stati (specificando trigger, guard e actions)

Alcuni esempi di domande sono

  - Quali eventi consentono l'entrata o l'uscita in o da una schermata?
    - Cosa accade se si chiude l'applicazione ma non si sono salvate le modifiche?
    - È possibile entrare in una schermata in tutte le circostanze o ci sono condizioni in cui ciò non è consentito (es. pagina vuota, non si hanno i privilegi, ...)

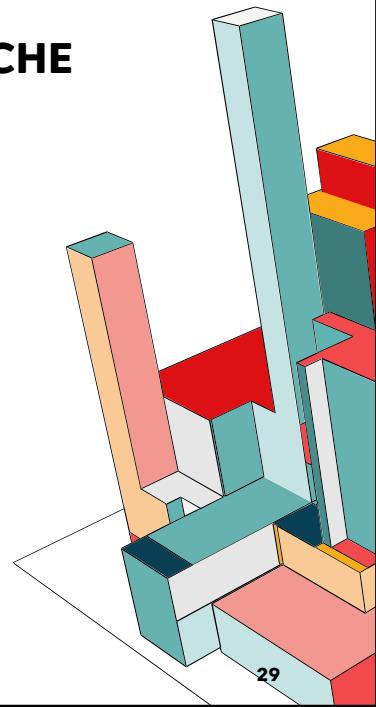
12/03/2022



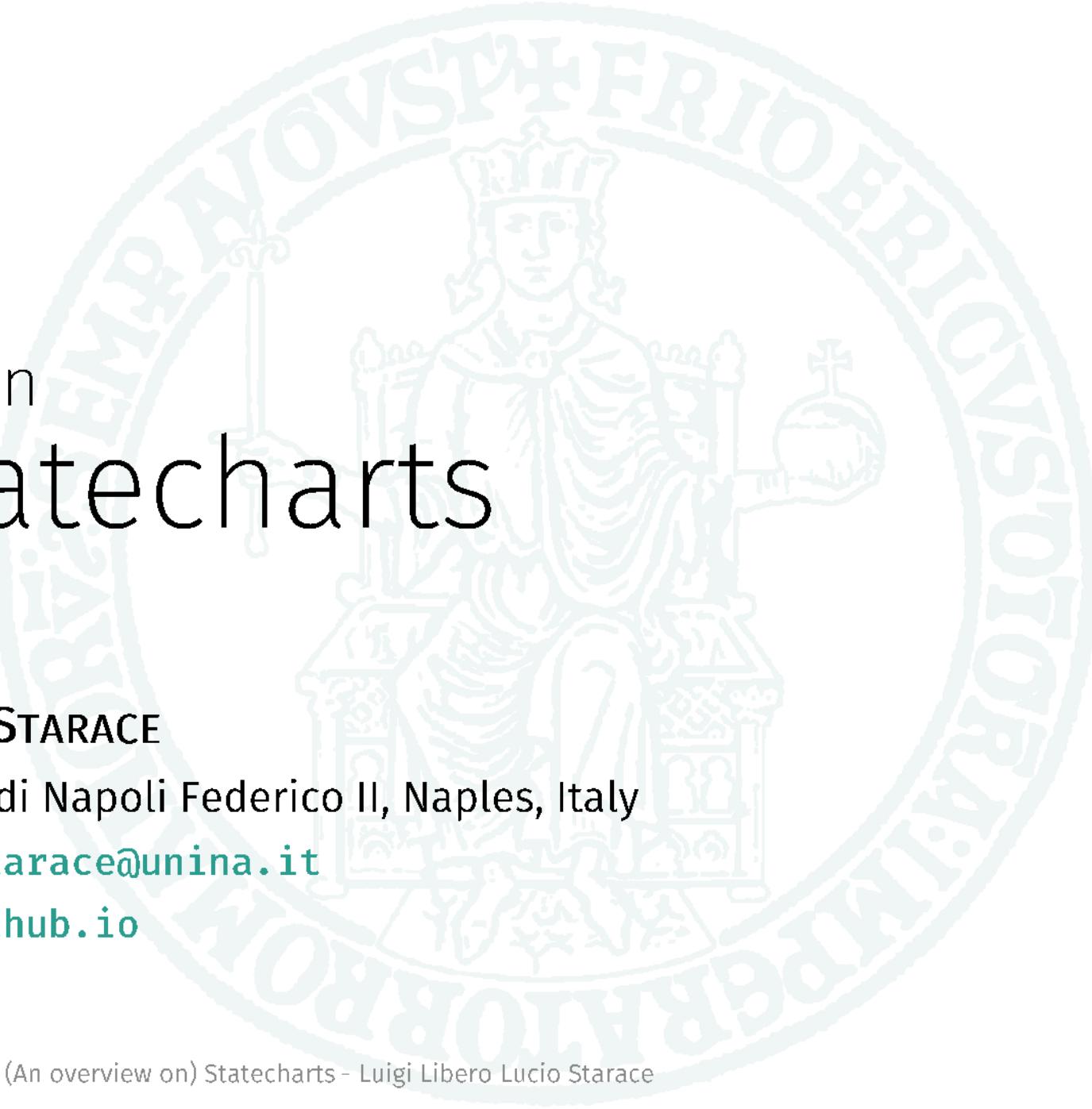
## STATECHART PER INTERFACCE GRAFICHE

Alcuni esempi di domande sono

- La modalità di un sistema è uno stato in cui può entrare il software e dove gli effetti delle azioni degli utenti possono variare in base alla modalità
    - Ad ogni modalità si associa uno stato dello statechart, e da esso le transizioni possono partire per cause diverse oppure passare in altri stati rispetto ad altre modalità del sistema (es. sola lettura, lettura-scrittura, ...)
    - La modalità del sistema può variare solo all'entrata di una schermata o anche durante la sua visualizzazione?
  - Vi sono elementi il cui aspetto varia (es. viene disabilitato, diviene invisibile, cambia colore / posizione, ...) oppure il cui comportamento varia (es. esegue azioni diverse in contesti diversi, ...) a seguito di un'interazione con l'utente?
4. Trasformare tutte le risposte ottenute in uno statechart completo
- Si utilizza un approccio top-down: prima si realizzano gli stati corrispondenti agli eventi e poi si descrivono le azioni, cioè prima si costruisce lo scheletro dello statechart e, successivamente, si aggiungono eventi e azioni



12/03/2022



# An overview on UML Statecharts

Luigi Libero Lucio STARACE

Università degli Studi di Napoli Federico II, Naples, Italy

[luigiliberolucio.starace@unina.it](mailto:luigiliberolucio.starace@unina.it)

<https://luistar.github.io>

# Context and Motivations

Modelling behaviours of (reactive) systems

# UML Statecharts

- Also known as **UML (Behavioural) State Machines**
- Extension of Harel's Statecharts [1]
- Widely-used to model dynamic aspects of **systems** (especially **reactive** ones)

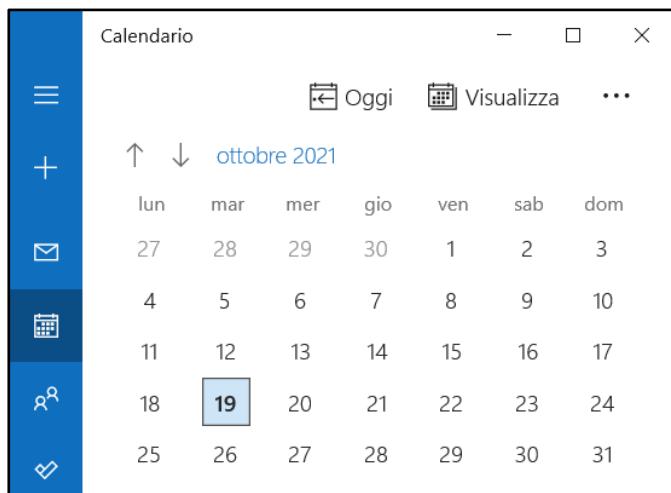


[1] Harel, D. (1987). Statecharts: A visual formalism for complex systems. *Science of computer programming*, 8(3), 231-274.

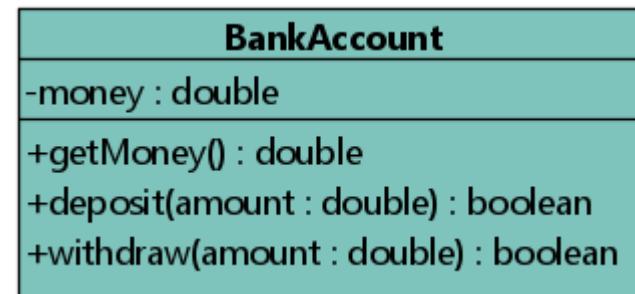
# Reactive Systems

Systems that **react** to (external or internal) events

Anything comes to mind?



Software with a GUI



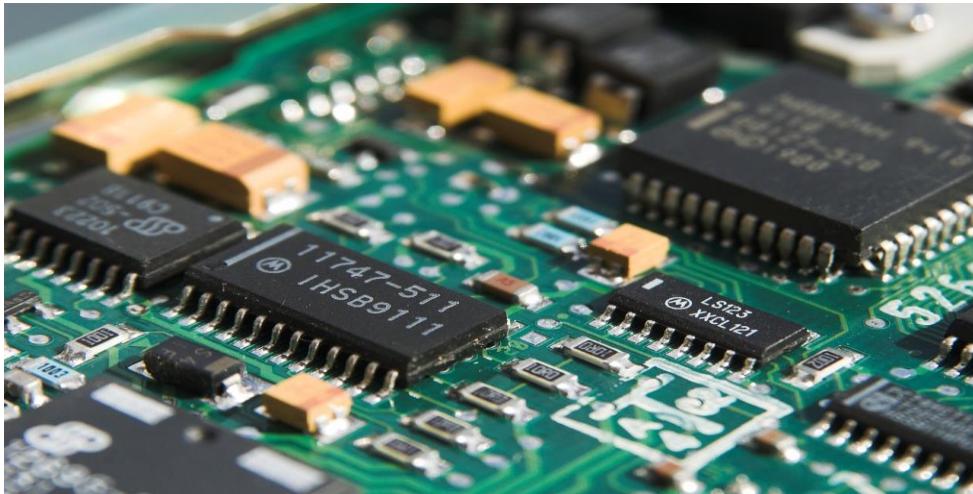
Objects in Object Oriented  
programming



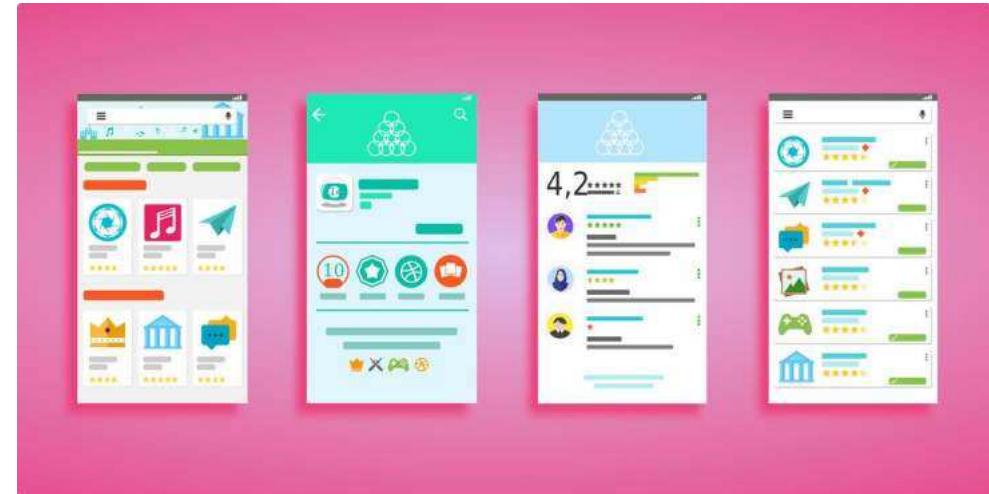
The ABS controller on the  
Ferrari F458

# Statecharts in the real world

Statecharts are largely used in the industry, and not only for modelling!



Embedded software is often automatically generated from Statecharts



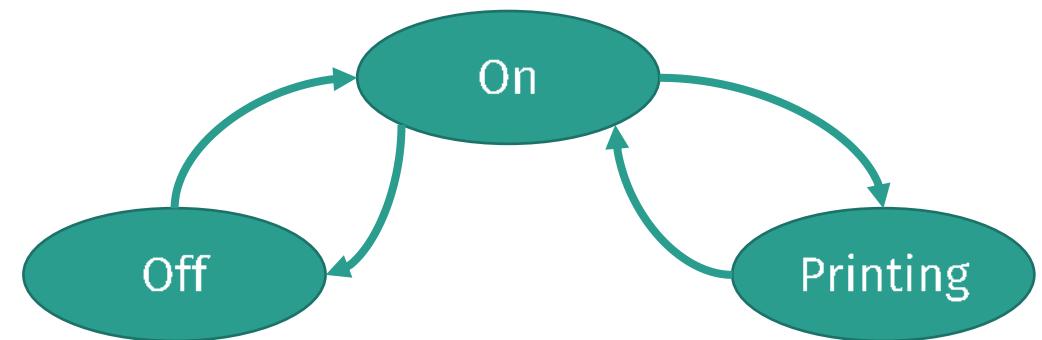
The logic behind modern User Interfaces can be managed through Statecharts

# Modelling with States and Transitions

States represent situations in which some invariant condition holds.

- **Static conditions:** system is waiting for something to happen.
- **Dynamic conditions:** system is performing a specific task.

Transitions represent possible state changes



# UML Statecharts

Syntax and Semantics

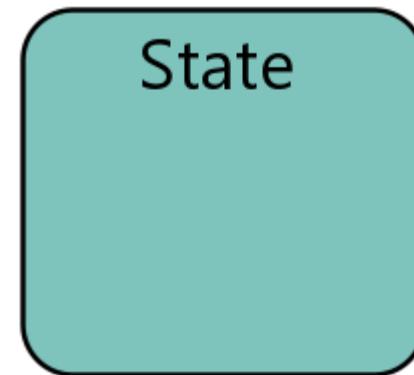
# Regions, vertices and transitions

- A UML Statechart contains a top-level **region**
- A region contains **vertices** and **transitions**
- **Vertices** represent states
- **Transitions** are depicted as directed edges between two vertices
- Several kinds of vertices exist, with different semantics

# Simple states

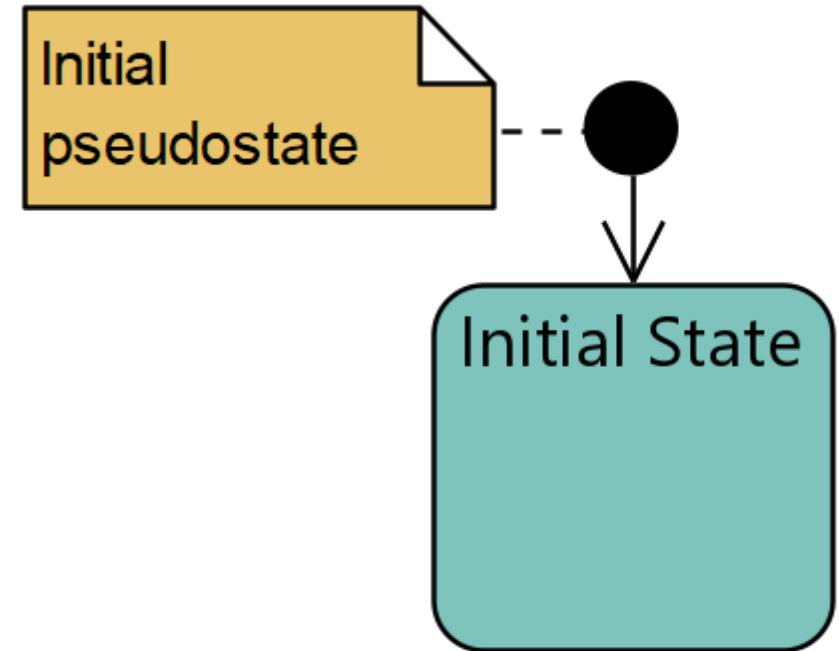
Represent unstructured system states

- Depicted as a rectangle with rounded edges
- A **name compartment** holds the (optional) name of the state, as a string.



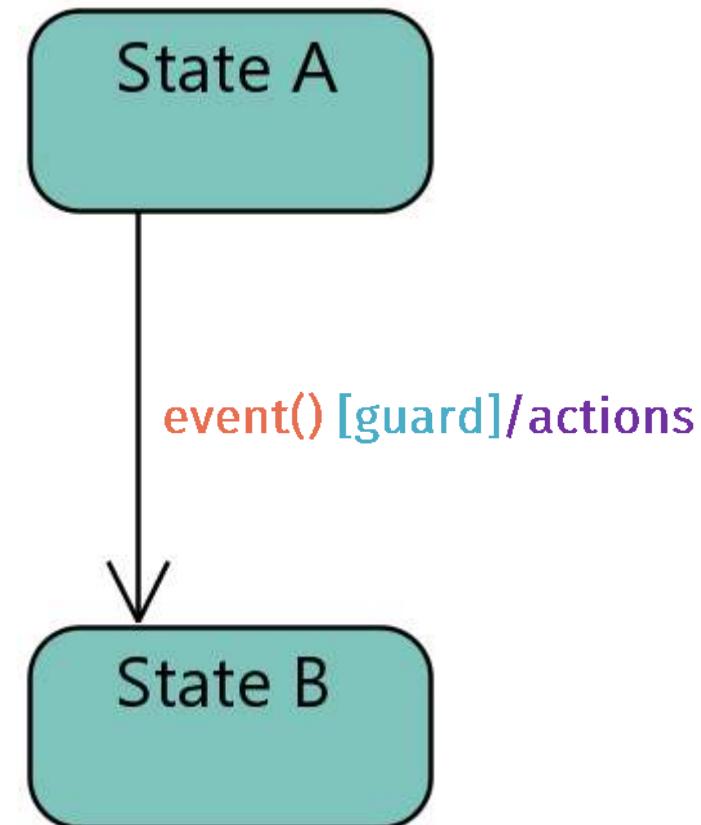
# Initial Pseudostates and Final States

- **Initial pseudostates** are used to mark the default (initial) state
- A region can contain at most one initial pseudostate.
- **Final states** model a situation in which the computation is completed (i.e., the system won't process any additional events)



# Transitions: Syntax

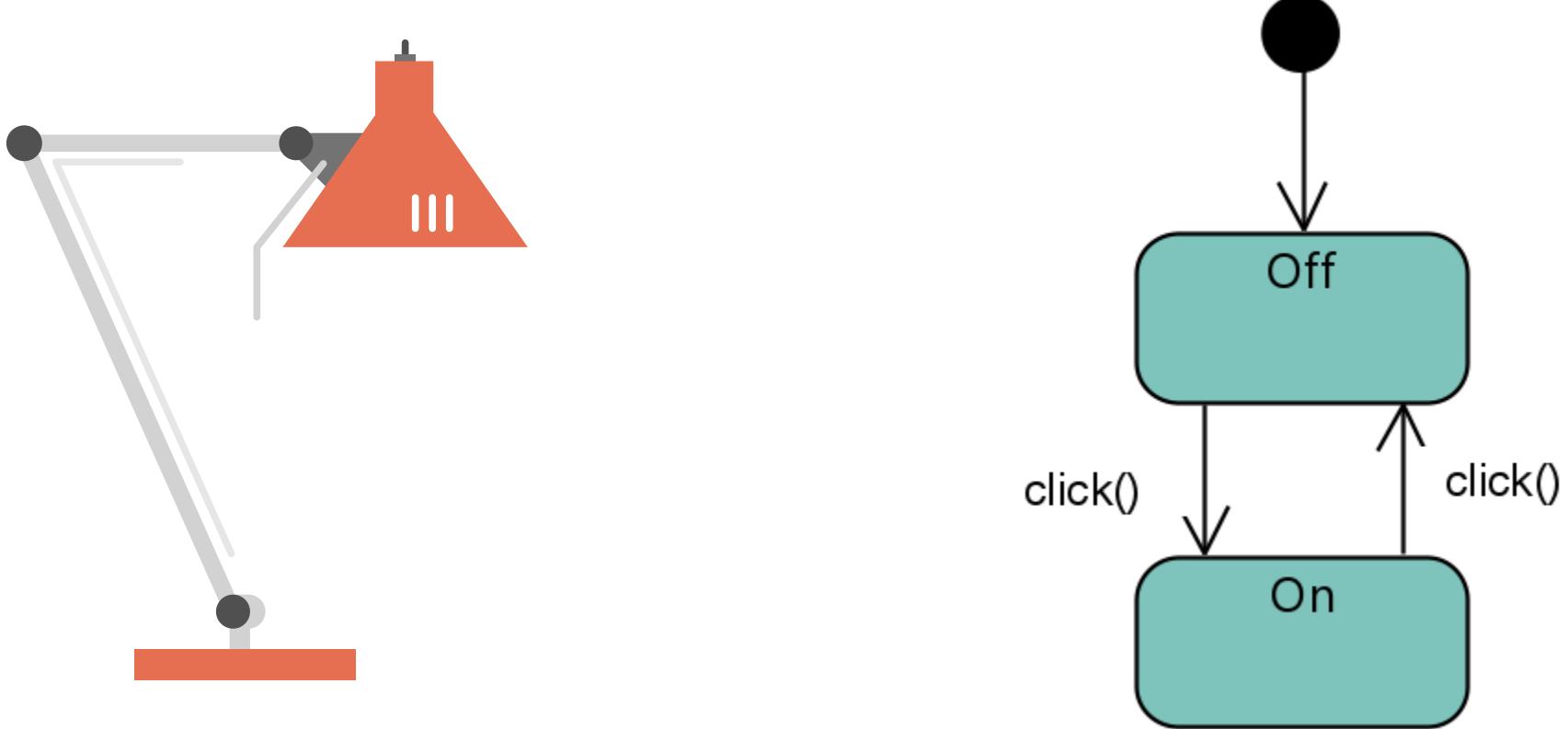
- Transitions indicate state changes
- Can be decorated with a label of the form:  
**triggers [guard] /actions**
  - **triggers** is a list of events that may induce a state change
  - **guard** is a Boolean condition
  - **actions** is a list of operations to execute when the transition fires.
- All the above parts of the label are optional
- Self-transitions are possible



# Transitions: Semantics

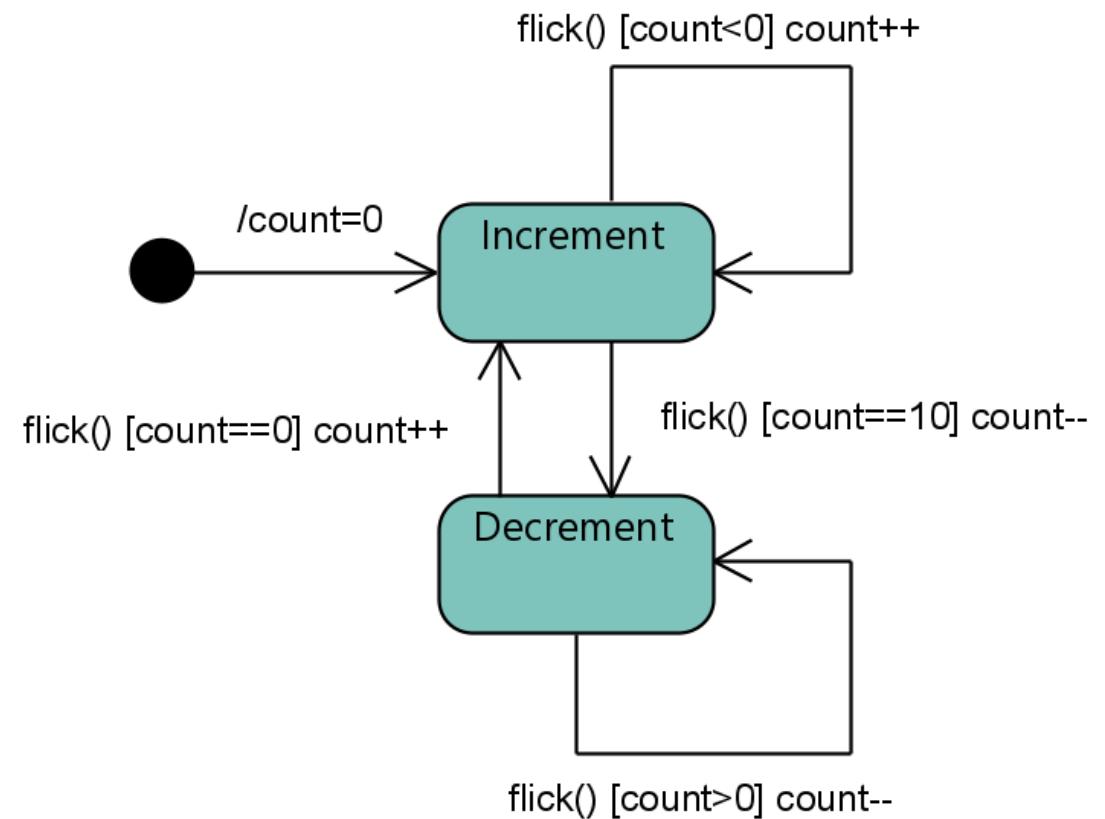
- For a transition to be **fireable**:
  - Events matching all of the triggers should be fired;
  - The condition in the guard must evaluate to TRUE.
- A **spontaneous** transition is one with no **triggers** and no **guard**.
- After a transition fires, its associated list of actions is executed.
- If multiple transitions are fireable, only one of them actually fires (nondeterministically determined).

# Example: a lamp with a single button



# Example: a simple counter (Java)

```
public class Counter {  
    private int count = 0;  
    private String mode = "increment";  
    public void flick() {  
  
        if(mode.equals("increment"))  
            count++;  
        else  
            count--;  
  
        if(count==10)  
            mode = "decrement";  
        else if (count==0)  
            mode = "increment";  
  
    }  
}
```



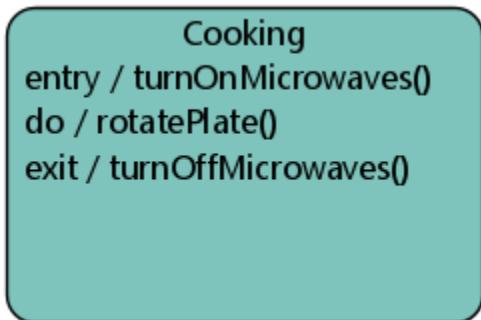
# States: internal activities

States can (optionally) contain a list of **internal activities**.

Each activity is characterized by a **label** indicating **when** the activity is to be invoked.

Reserved labels:

- **entry** / activity performed upon entry
- **do** / performed as long as the system is in the state (after entry activities completed)
- **exit** / activity performed upon exit



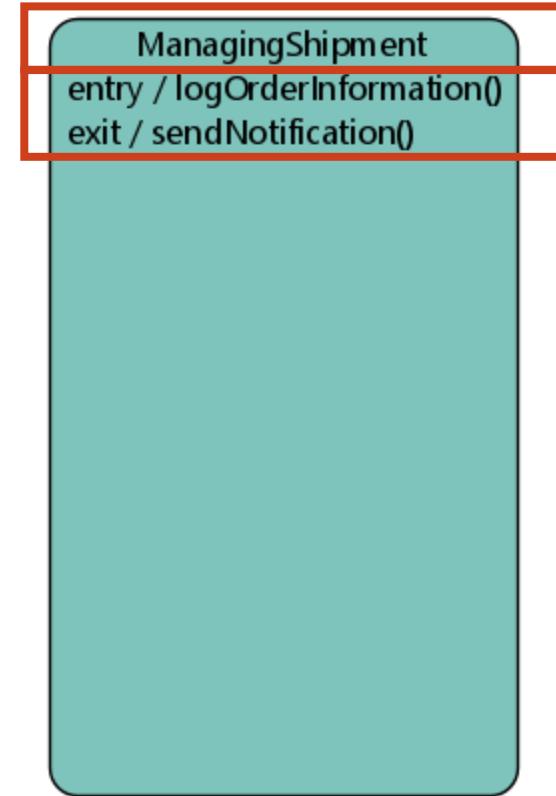
# Composite States

A state can contain:

- name compartment
- internal activities compartment
- One (or more) inner regions!

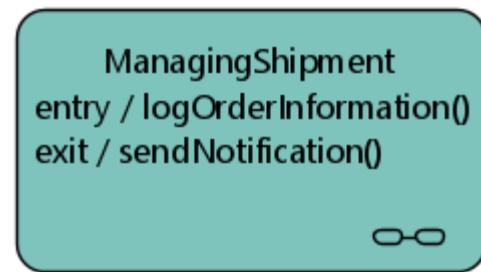
A state with inner regions is a  
**composite state**

- States in a inner region are  
called **substates**



# Composite States

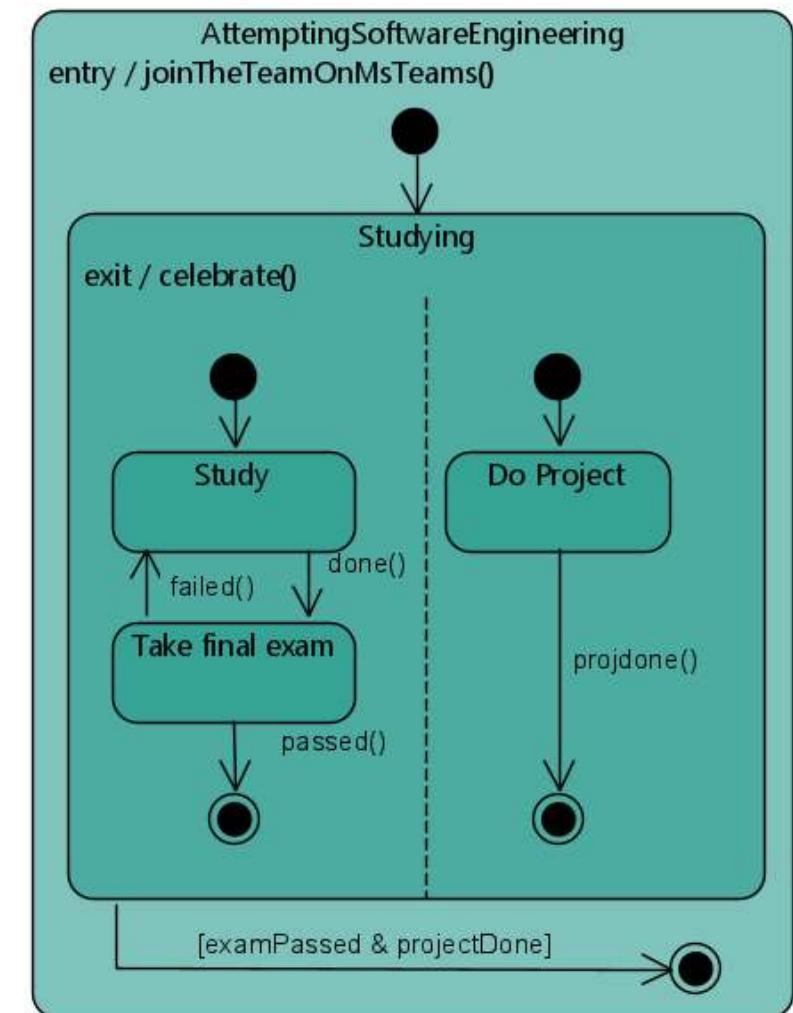
- Allow modellers to define a **hierarchical structure**
- The inner region details the behaviour of the state it belongs to
- Provide a elegant and concise way to model complex behaviours (and hide complexity when not necessary)



The ManagingShipment composite state, with the inner region hidden

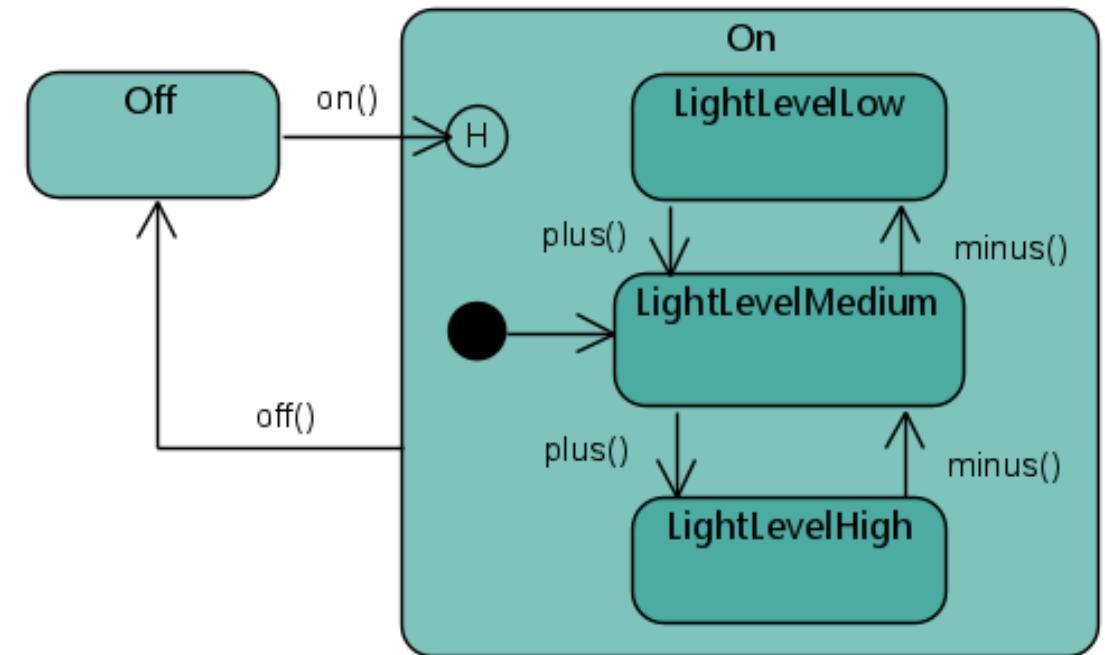
# Composite States: parallel regions

- Composite states can contain multiple regions, representing behaviours that may occur in parallel
- When exiting from a composite state, all of its regions are terminated



# Shallow History Pseudostates

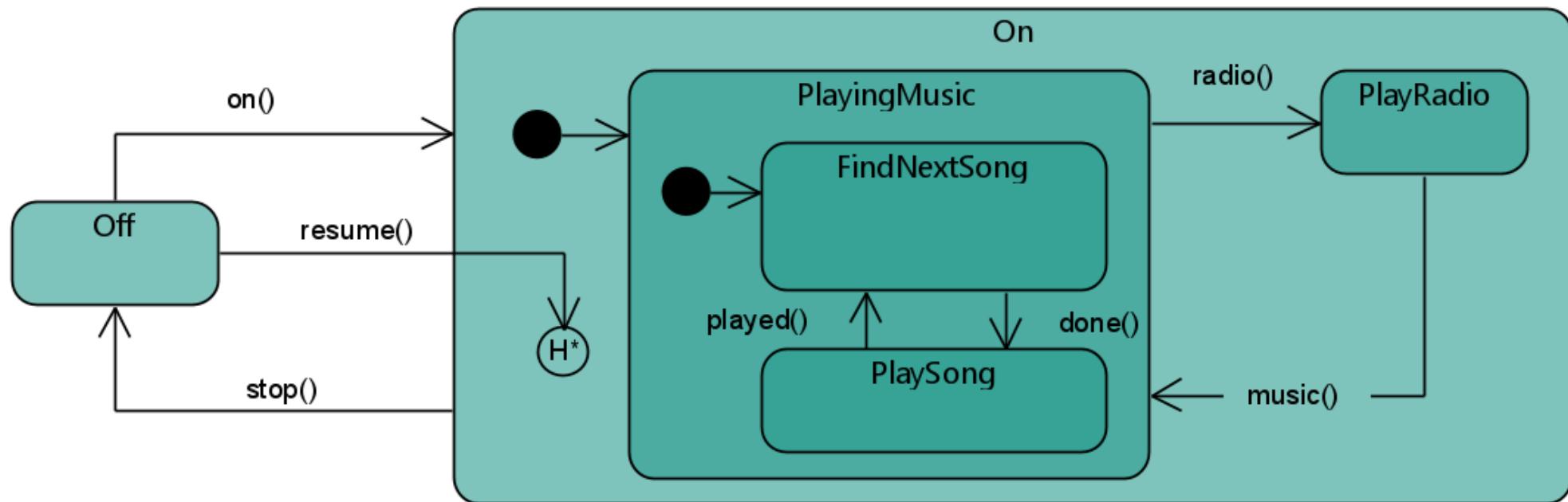
- Depicted as a  $\textcircled{H}$
- Represents the most recently active state of a composite state, but not substates of that substate!
- Only in composite states, and only one per region



Statechart for a lamp with three different light levels

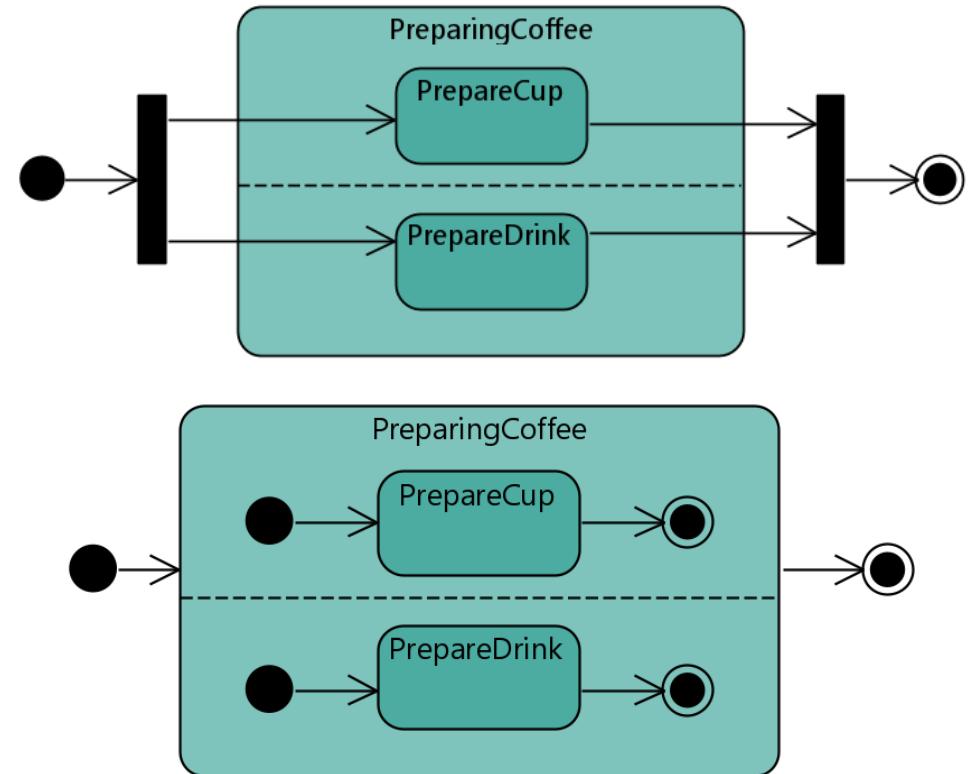
# Deep History Pseudostates

- Depicted as a  $H^*$
- Same as shallow history ones, but restore the entire region configuration (substates of the substates included!)



# Fork and Join Pseudostates

- **Forks** split incoming transitions into multiple transitions entering vertices in orthogonal regions
- **Joins** merge transitions exiting vertices in orthogonal regions into a single transition



Semantically equivalent statecharts.  
Top one uses fork and join pseudonodes

# Statecharts: Tips and Tricks

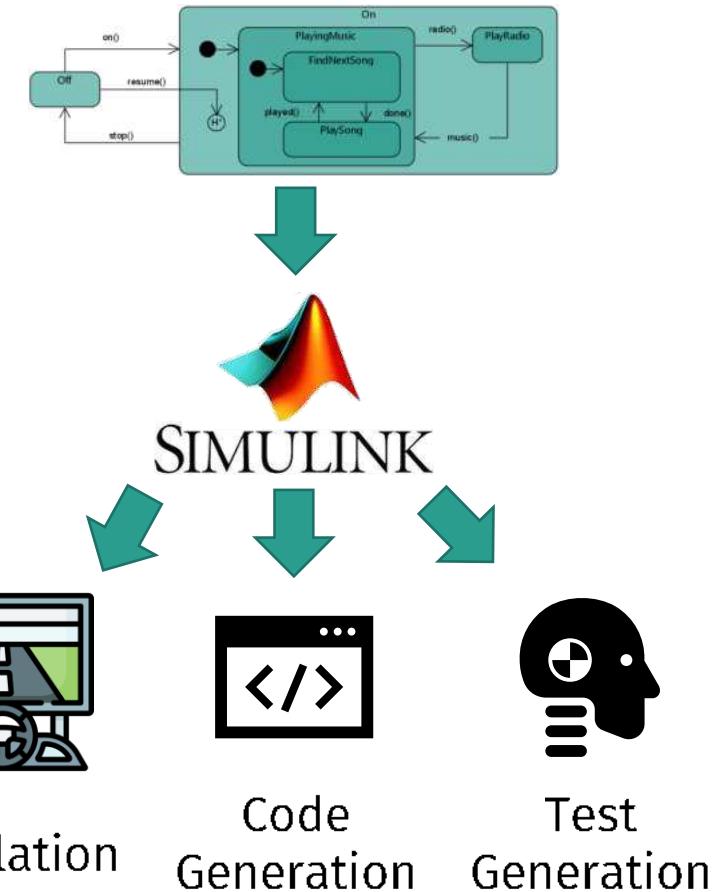
- Each state should typically have at least one transition entering it, and one exiting transition.
- Diagrams are typically read from top-left to bottom-right, so place initial/final pseudostates accordingly!
- If multiple states have a common entry and/or exit condition, consider using composite states.
- Make sure not to model non-determinism, unless it's what you *really* need!
- **At most one state can be active in a region at any given time!**

# Statecharts in the wild

Practical applications of Statecharts (other than modelling!)

# Model-driven Development

- Next step in the increasing abstraction trend
- De-facto standard in many embedded software domains (e.g.: automotive)
- Thanks to tools such as **Simulink**, it is possible to simulate Statechart models, to automatically generate code and tests, and much more (e.g.: formal methods!)



# Model-driven Development

## Pros

- In some domains, typically more cost-effective, faster and leads to higher quality
- Models understandable by domain experts
- Models are documentation!
- Less technology dependant
- Less personnel dependant

## Cons

- Tools are expensive
- Not flexible enough for some applications
- Code generation typically supported for a limited number of platforms

# Managing UI States with Statecharts

- Statecharts can also be used to «guide» GUI logic!
- Statecharts are easier to understand (than code!)
- Behaviour is **decoupled** from GUI components
  - Separate the **WHEN** (encoded in the Statechart) from the **WHAT** (what should happen, encoded in the UI component)
- Statecharts **scale** well as complexity grows
- Studies [2] have shown lower defect counts for statechart-based GUI controllers.

[2] Ian Horrocks, *Constructing the UI in Statecharts*

# Example: Statechart-based UI with XState

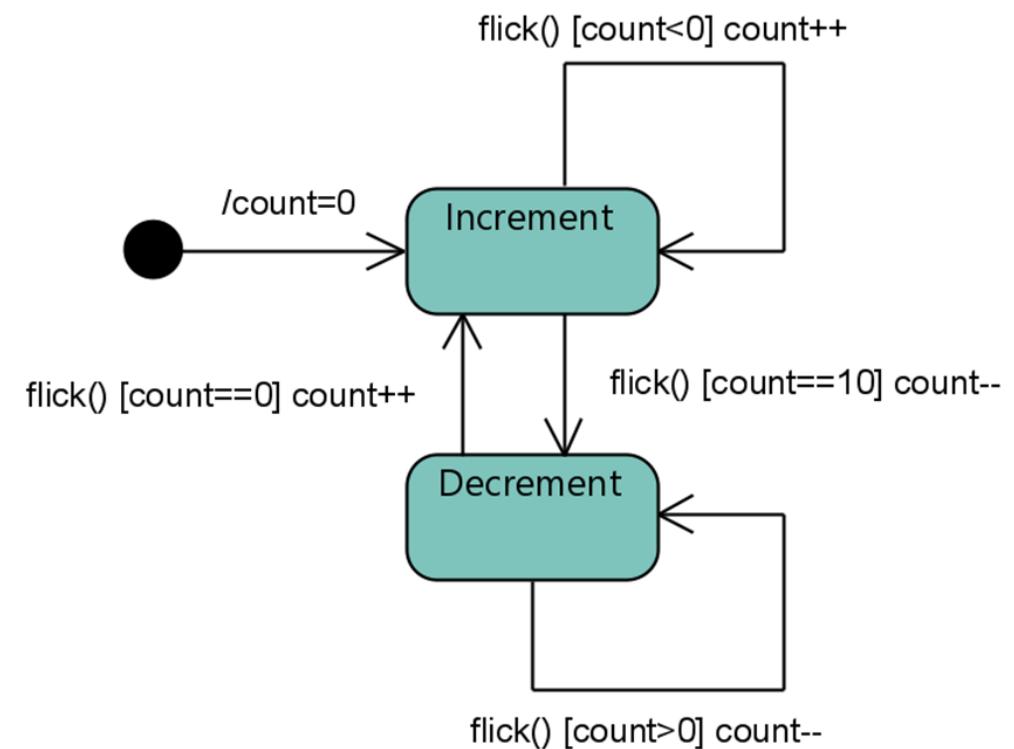
- XState is an open-source Javascript library to create, interpret, and execute statechart models
- Can be integrated with many Javascript UI libraries such as React, Vue, Svelte
- Great at managing UI State through Statecharts
- Also supports testing!
- Available at: <https://xstate.js.org/>



# Example: Statechart-based UI with XState

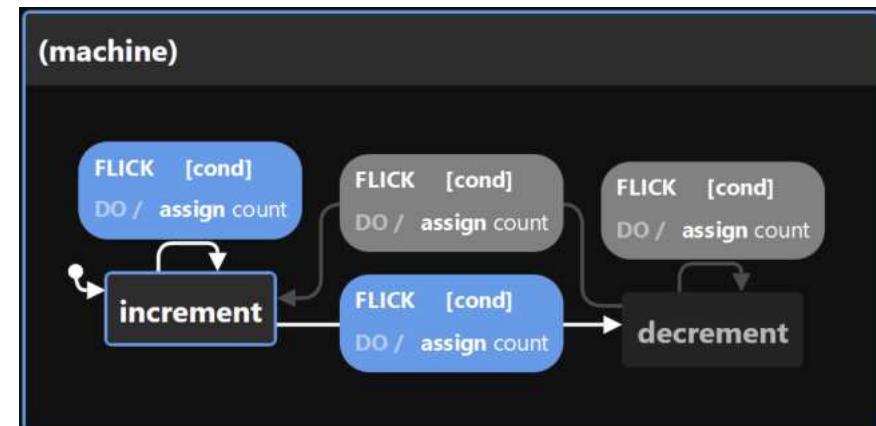
Remember our Counter example?

```
public class Counter {  
    private int count = 0;  
    private String mode = "increment";  
    public void flick() {  
        if(count>10)  
            mode = "decrement";  
        else if (count<0)  
            mode = "increment";  
  
        if(mode.equals("increment"))  
            count++;  
        else  
            count--;  
    }  
}
```

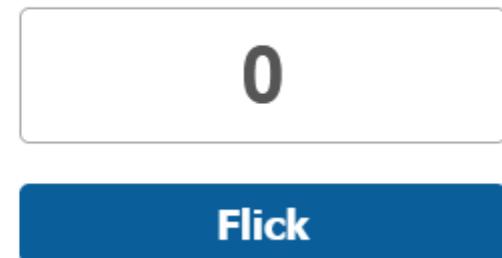


# Example: Statechart-based UI with XState

- Let's implement a simple UI for it, and let's do it the Statechart way, with Xstate and React
- Code Sandbox available [here](#):



↑  
**The Counter App**



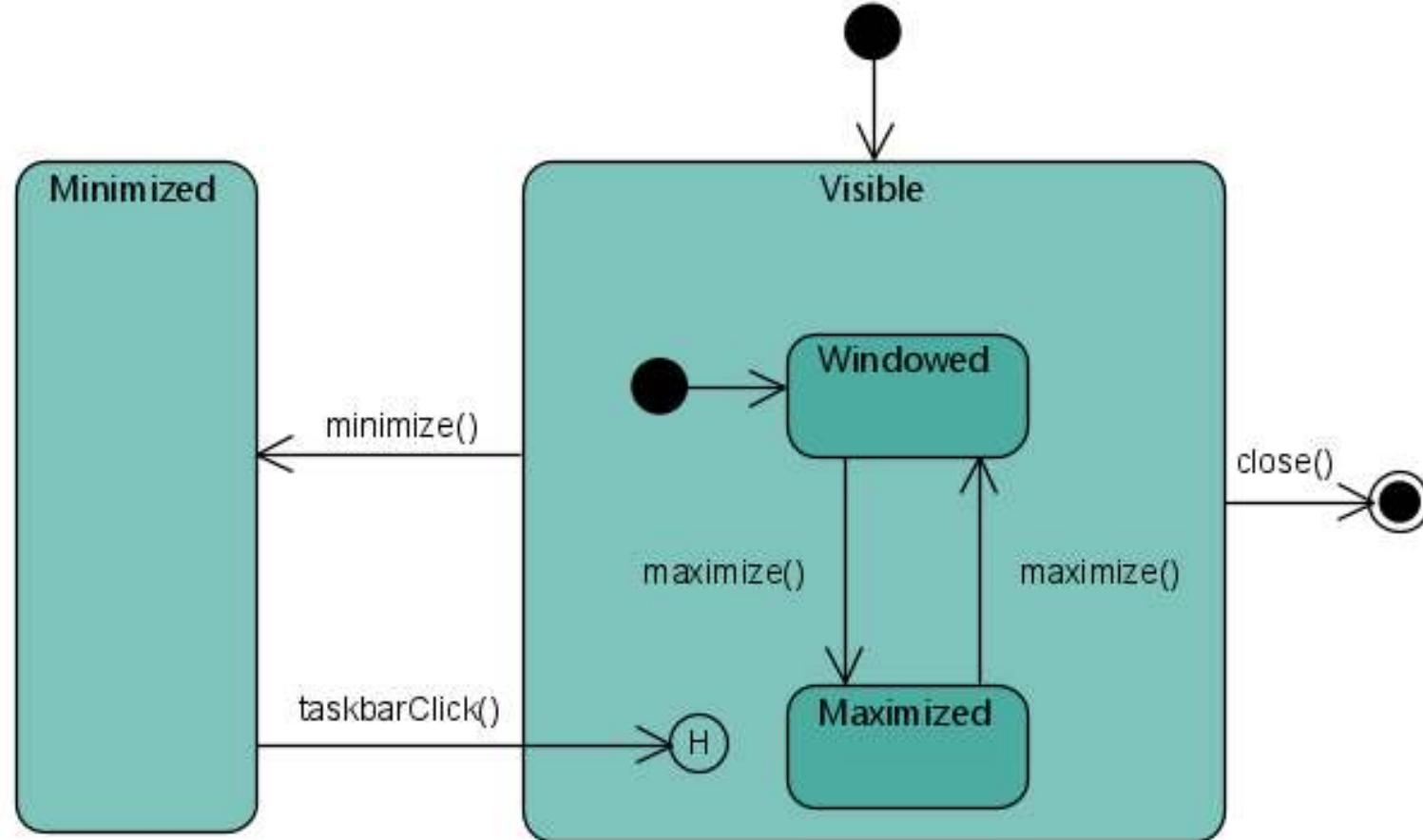
# Practice time

Brave and bold volunteers, come forward!

# Exercise #1

Si descriva con uno Statechart il comportamento di una generica finestra (e.g.: minimizzata, massimizzata, modalità finestra, etc.) in un ambiente desktop basato su finestre (come quello di Microsoft Windows).

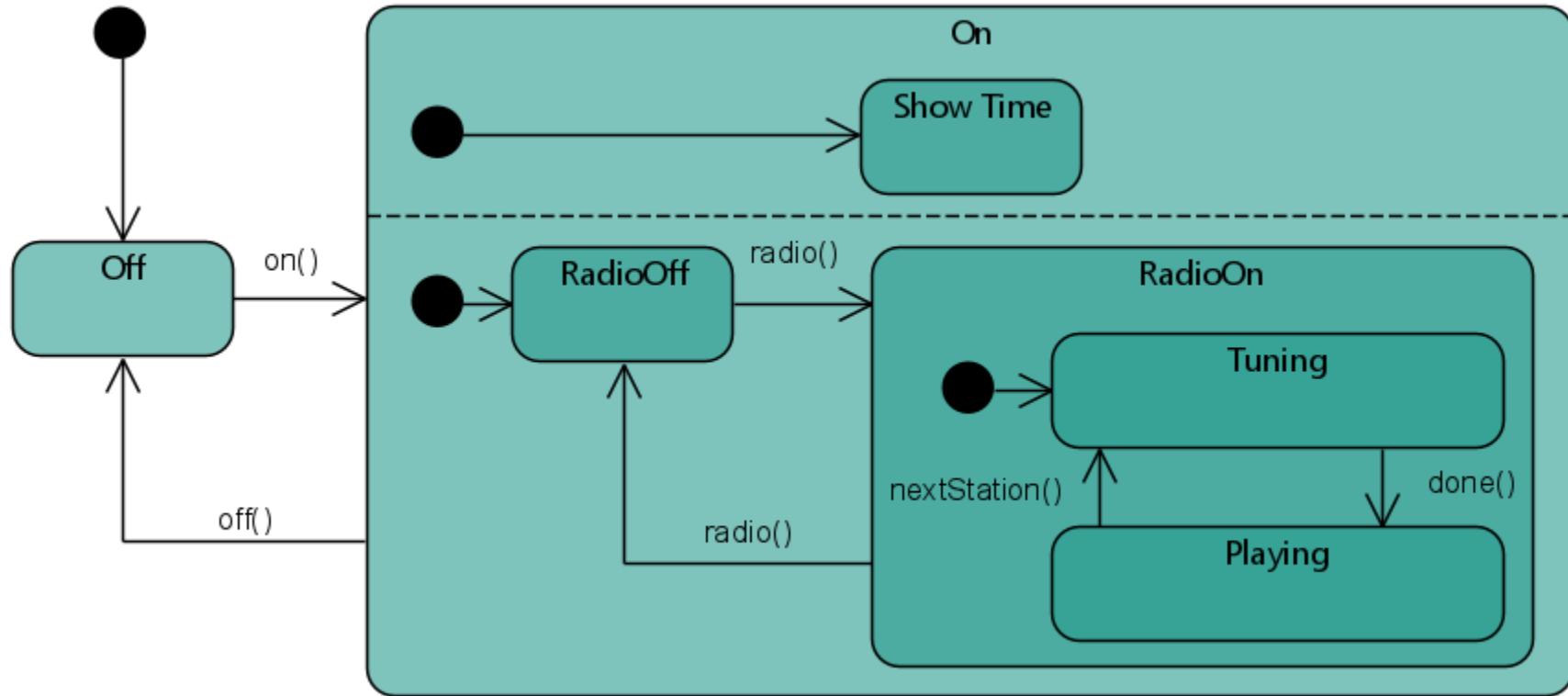
# Exercise #1 – Proposed solution



## Exercise #2

Un orologio da tavolo, una volta acceso, mostra l'orario corrente sul proprio display LCD e, se l'utente preme un apposito pulsante, può anche sintonizzarsi su stazioni radio e riprodurne le trasmissioni dalle casse integrate. Tramite un pulsante «next station» è possibile passare alla stazione radio successiva, che verrà riprodotta dopo una breve fase di ricerca e sintonizzazione.

# Exercise #2 – Proposed solution



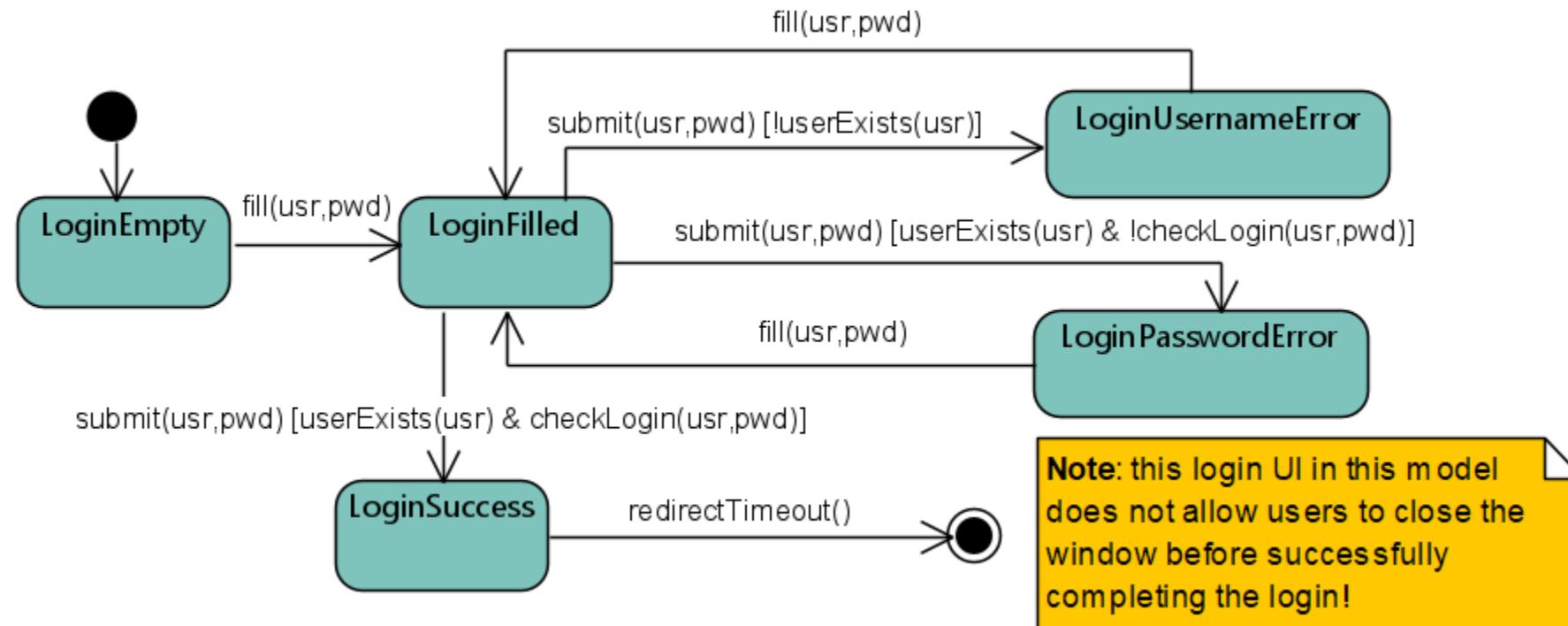
## Exercise #2 – Follow up

Come si potrebbe modificare lo statechart precedente per fare in modo che, all'accensione, l'orologio da tavolo riprenda con la riproduzione della radio se la radio era attiva nel momento dello spegnimento?

# Exercise #3

La schermata di login di un'applicazione permette agli utenti di inserire le proprie credenziali ed accedere. Se il nome utente inserito non è tra quelli presenti nel sistema, viene mostrato un warning dedicato. Altrimenti, se il nome è presente ma la password errata, viene mostrato un diverso warning e viene abilitato un pulsante per accedere alla funzionalità di reset password. Se le credenziali sono corrette, si accede al sistema.

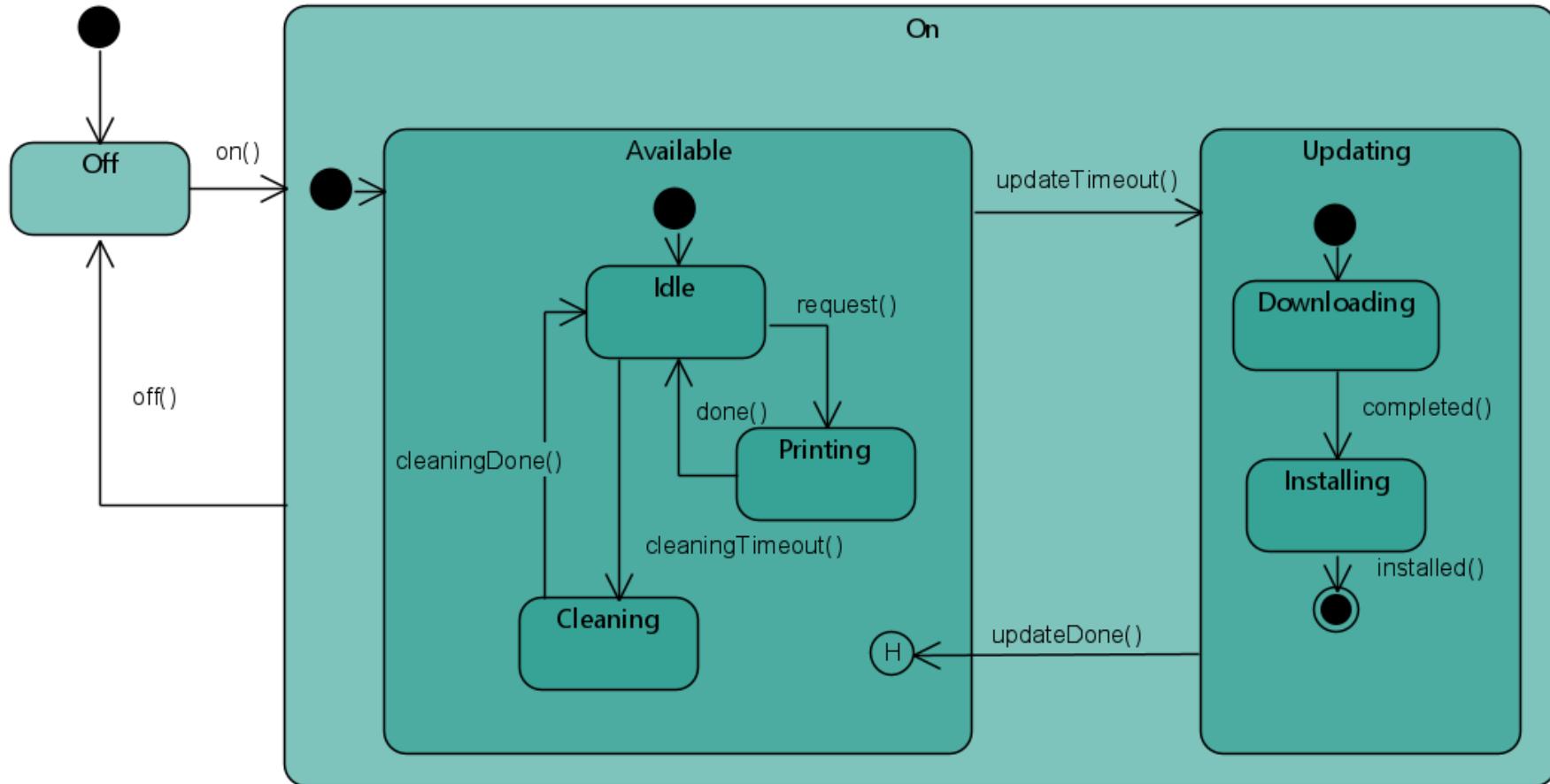
# Exercise #3 – Proposed solution



## Exercise #4

Una stampante, previa accensione, resta in attesa di ricevere via rete documenti da stampare. In presenza di richieste, la stampante procede alla stampa. Quando è accesa e non è in fase di stampa, la stampante, una volta al giorno, effettua la pulizia delle testine. Inoltre, sempre con cadenza giornaliera, la stampante scarica e installa aggiornamenti dalla casa madre. In questo caso, la stampante interrompe qualsiasi attività in corso per effettuare l'aggiornamento, e le riprende ad aggiornamento effettuato.

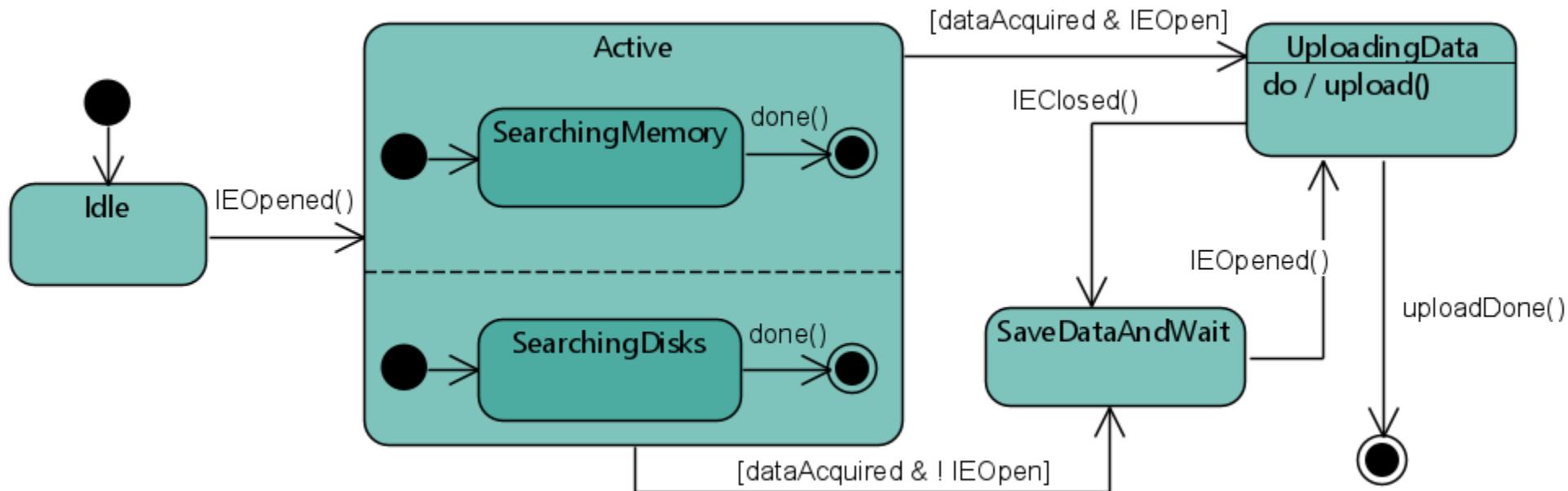
# Exercise #4 – Proposed solution



# Exercise #5

Un malware, una volta installato su un PC, rimane latente fino a quando l'utente non apre Internet Explorer. A quel punto, il malware si attiva e, in parallelo, ricerca informazioni sensibili nei dischi rigidi e nella memoria del PC.Terminate queste attività, il malware sfrutta una vulnerabilità di Internet Explorer per inviare le informazioni raccolte a un server remoto. Se Internet Explorer viene chiuso prima dell'invio delle informazioni, il malware salva le informazioni trovate e riprova ad inviarle al successivo avvio di Internet Explorer.

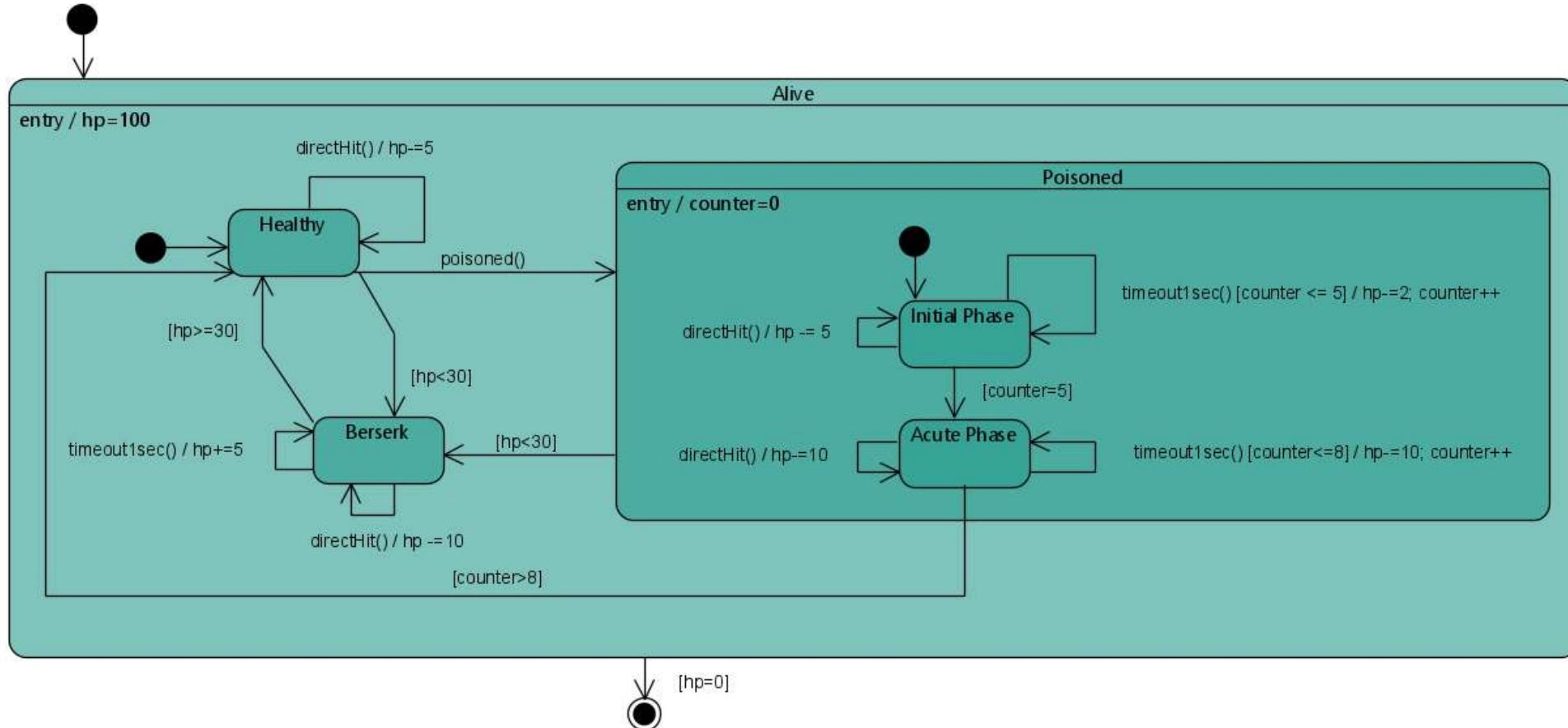
# Exercise #5 – Proposed solution



# Exercise #6 – Video Game Player

In un videogame, il giocatore inizia la partita con una salute pari a 100 HP. Durante la partita, il giocatore può subire attacchi diretti, che diminuiscono la salute residua di 5 HP. Inoltre, il giocatore può essere avvelenato. L'avvelenamento prevede una fase iniziale che dura 5 secondi, in cui il giocatore subisce 2 HP di danno al secondo, e una fase acuta, che dura 3 secondi e durante la quale il giocatore subisce 10 HP di danno al secondo. Mentre il giocatore è avvelenato in fase acuta, i danni inflitti da attacchi diretti raddoppiano. Quando la salute scende al di sotto della soglia critica di 30 HP, il giocatore passa in modalità “berserk”. Quando è in questa modalità, il giocatore è immune all'avvelenamento e si cura di 5 HP al secondo, ma subisce danni doppi dai colpi diretti. Quando i punti salute scendono a zero, il giocatore muore e la partita termina.

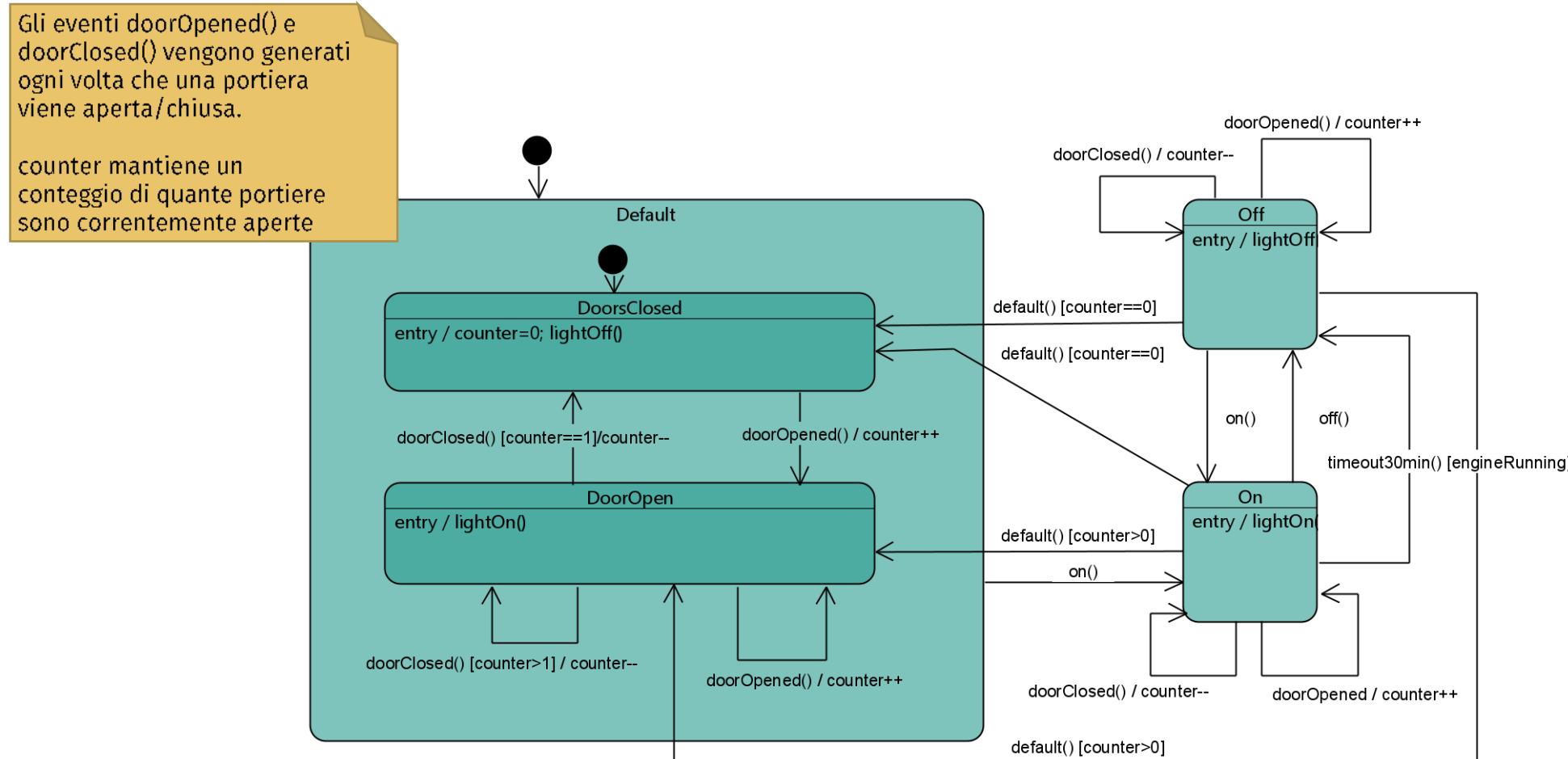
# Exercise #6 – Proposed solution



## Exercise #7

Le luci di cortesia di un'auto hanno un interruttore che può assumere tre posizioni: ON, OFF, e DEFAULT. Quando l'interruttore è in posizione ON, le luci di cortesia sono sempre accese. Al contrario, quando è in posizione OFF, le luci di cortesia sono sempre spente. Quando l'interruttore è in posizione DEFAULT, le luci si accendono soltanto quando una delle portiere è aperta, e restano spente altrimenti. Inoltre, quando il motore è spento e l'interruttore è in posizione ON, le luci si spengono in ogni caso dopo 30 minuti per evitare di consumare la batteria, e l'interruttore si sposta su OFF.

# Exercise #7 – Proposed solution



# Exercise #7 – Follow up

The proposed solution is quite complex. Is it possible to express the same behaviours with a simpler statechart?

**Hint:** try introducing some composite states!

# References and further readings

- OMG UML Specification (2.5)  
<https://www.omg.org/spec/UML/2.5/PDF/>
- Ivar Jacobson, James Rumbaugh and Grady Booch. "The unified modeling language reference manual."

# **AN INTRODUCTION TO CLOUD COMPUTING AND AMAZON WEB SERVICES**

---

Luigi Libero Lucio Starace

*luigiliberolucio.starace@unina.it*

November 19, 2027

Università degli Studi di Napoli “Federico II”, Naples, Italy

## CLOUD COMPUTING

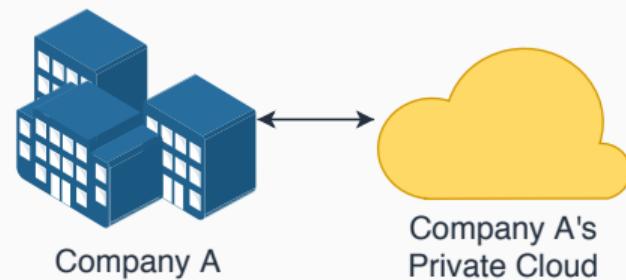
---

Cloud computing is the on-demand delivery of computing resources through a cloud services platform via the internet with pay-as-you-go pricing.

Cloud computing is the **on-demand delivery** of computing resources through a cloud services platform via the internet with pay-as-you-go pricing.

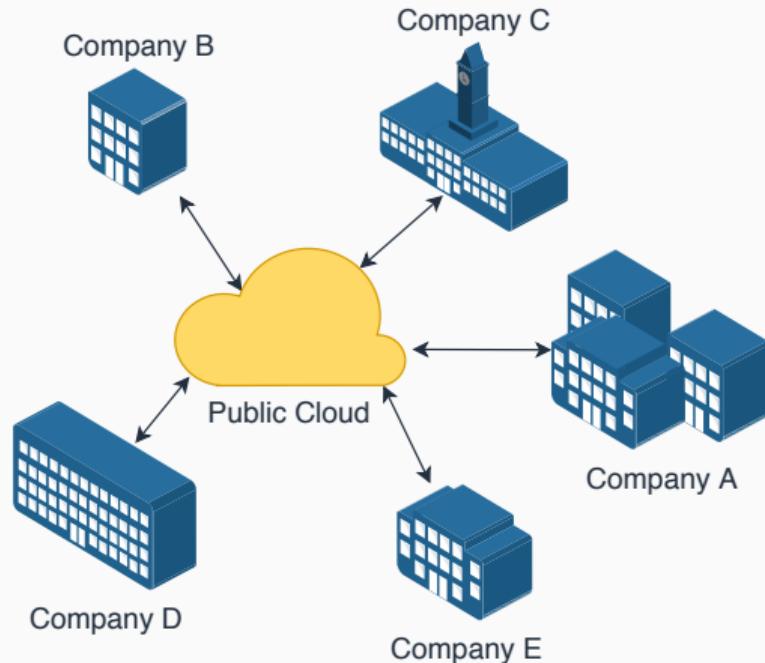
Cloud computing is the **on-demand delivery** of computing resources through a cloud services platform via the internet with **pay-as-you-go** pricing.

## ■ Private Cloud



# PRIVATE AND PUBLIC CLOUD

- Private Cloud
- Public Cloud



## THE BIGWIGS IN PUBLIC CLOUD

- Google



Google Cloud

## THE BIGWIGS IN PUBLIC CLOUD

- Google
- IBM



## THE BIGWIGS IN PUBLIC CLOUD

- Google
- IBM
- Microsoft



## THE BIGWIGS IN PUBLIC CLOUD

- Google
- IBM
- Microsoft
- Alibaba



## THE BIGWIGS IN PUBLIC CLOUD

- Google
- IBM
- Microsoft
- Alibaba
- Amazon



According to [Fle19]:

- 91% of the surveyed companies uses public cloud services

According to [Fle19]:

- 91% of the surveyed companies uses public cloud services
- 84% of these enterprises have a multi-cloud strategy

According to [Fle19]:

- 91% of the surveyed companies uses public cloud services
- 84% of these enterprises have a multi-cloud strategy
  - they buy cloud services from different providers;

According to [Fle19]:

- 91% of the surveyed companies uses public cloud services
- 84% of these enterprises have a multi-cloud strategy
  - they buy cloud services from different providers;
  - some of them also combine public and private clouds (**hybrid cloud** approach).

## PUBLIC CLOUD ADOPTION

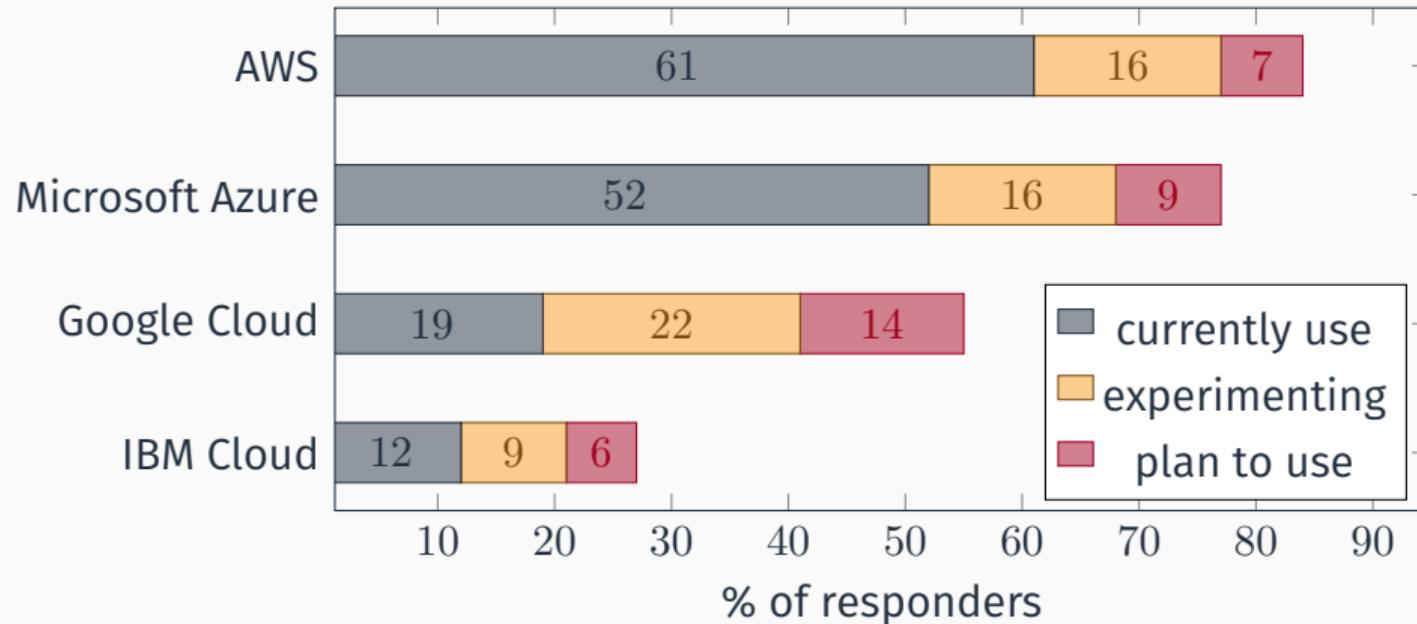


Figure 1: Public cloud adoption [Fle19]

## WHY MIGHT ONE USE CLOUD COMPUTING?

Suppose you have a great business idea.

What steps do you take to start making money?

## STARTING A BUSINESS: THE TRADITIONAL APPROACH

1. Estimate supply and demand;

## STARTING A BUSINESS: THE TRADITIONAL APPROACH

1. Estimate supply and demand;
2. Estimate infrastructural needs;

## STARTING A BUSINESS: THE TRADITIONAL APPROACH

1. Estimate supply and demand;
2. Estimate infrastructural needs;
3. Purchase and deploy infrastructure;

## STARTING A BUSINESS: THE TRADITIONAL APPROACH

1. Estimate supply and demand;
2. Estimate infrastructural needs;
3. Purchase and deploy infrastructure;
4. Install and test your system;

## STARTING A BUSINESS: THE TRADITIONAL APPROACH

1. Estimate supply and demand;
2. Estimate infrastructural needs;
3. Purchase and deploy infrastructure;
4. Install and test your system;
5. Offer your services to clients.

## STARTING A BUSINESS: ISSUES WITH THE TRADITIONAL APPROACH

- Infrastructure is **very expensive**:

## STARTING A BUSINESS: ISSUES WITH THE TRADITIONAL APPROACH

- Infrastructure is **very expensive**:
  - Hardware costs; 

## STARTING A BUSINESS: ISSUES WITH THE TRADITIONAL APPROACH

- Infrastructure is **very expensive**:
  - Hardware costs; 
  - Real estate costs;

## STARTING A BUSINESS: ISSUES WITH THE TRADITIONAL APPROACH

- Infrastructure is **very expensive**:

- Hardware costs; 
- Real estate costs;
- Cooling system; Power bill;

## STARTING A BUSINESS: ISSUES WITH THE TRADITIONAL APPROACH

- Infrastructure is **very expensive**:

- Hardware costs; 
- Real estate costs;
- Cooling system; Power bill;
- A few systems engineers;

## STARTING A BUSINESS: ISSUES WITH THE TRADITIONAL APPROACH

- Infrastructure is **very expensive**:

- Hardware costs; 
- Real estate costs;
- Cooling system; Power bill;
- A few systems engineers;
- Redundancy for high reliability.

## STARTING A BUSINESS: ISSUES WITH THE TRADITIONAL APPROACH

- Infrastructure is **very expensive**:
  - Hardware costs; 
  - Real estate costs;
  - Cooling system; Power bill;
  - A few systems engineers;
  - Redundancy for high reliability.
- It takes time to deploy the infrastructure;

## STARTING A BUSINESS: ISSUES WITH THE TRADITIONAL APPROACH

- Infrastructure is **very expensive**:
  - Hardware costs; 
  - Real estate costs;
  - Cooling system; Power bill;
  - A few systems engineers;
  - Redundancy for high reliability.
- It takes time to deploy the infrastructure;
- **What if the estimations were wrong?**

## STARTING A BUSINESS: USING CLOUD COMPUTING

1. Choose one or more cloud services providers;

## STARTING A BUSINESS: USING CLOUD COMPUTING

1. Choose one or more cloud services providers;
2. Deploy your systems on the cloud;

## STARTING A BUSINESS: USING CLOUD COMPUTING

1. Choose one or more cloud services providers;
2. Deploy your systems on the cloud;
3. Offer your services to clients;

## STARTING A BUSINESS: USING CLOUD COMPUTING

1. Choose one or more cloud services providers;
2. Deploy your systems on the cloud;
3. Offer your services to clients;
4. Pay for what you use.

## WHY MIGHT ONE USE CLOUD COMPUTING?

### Traditional process

- ✗ High investment risk;

### With Cloud Computing

## WHY MIGHT ONE USE CLOUD COMPUTING?

### Traditional process

- ✗ High investment risk;

### With Cloud Computing

- ✓ Reduced risk;

## WHY MIGHT ONE USE CLOUD COMPUTING?

### Traditional process

- ✗ High investment risk;
- ✗ Long time-to-market;

### With Cloud Computing

- ✓ Reduced risk;

## WHY MIGHT ONE USE CLOUD COMPUTING?

### Traditional process

- ✗ High investment risk;
- ✗ Long time-to-market;

### With Cloud Computing

- ✓ Reduced risk;
- ✓ Shorter time-to-market;

## WHY MIGHT ONE USE CLOUD COMPUTING?

### Traditional process

- ✗ High investment risk;
- ✗ Long time-to-market;
- ✓ Manages own data;

### With Cloud Computing

- ✓ Reduced risk;
- ✓ Shorter time-to-market;

## WHY MIGHT ONE USE CLOUD COMPUTING?

### Traditional process

- ✗ High investment risk;
- ✗ Long time-to-market;
- ✓ Manages own data;

### With Cloud Computing

- ✓ Reduced risk;
- ✓ Shorter time-to-market;
- ✗ Trust the vendor?

## WHY MIGHT ONE USE CLOUD COMPUTING?

### Traditional process

- ✗ High investment risk;
- ✗ Long time-to-market;
- ✓ Manages own data;
- ✓ Completely in control;

### With Cloud Computing

- ✓ Reduced risk;
- ✓ Shorter time-to-market;
- ✗ Trust the vendor?

## WHY MIGHT ONE USE CLOUD COMPUTING?

### Traditional process

- ✗ High investment risk;
- ✗ Long time-to-market;
- ✓ Manages own data;
- ✓ Completely in control;

### With Cloud Computing

- ✓ Reduced risk;
- ✓ Shorter time-to-market;
- ✗ Trust the vendor?
- ✗ Dependant from a specific vendor?

The traditional approach lacks **Elasticity**.

## ■ What is Elasticity?

The ability to grow or shrink infrastructure resources dynamically as needed to adapt to workload changes, possibly in an autonomic manner.

Public Cloud providers manage to offer services at very low prices, thanks to:

- Economies of scale.
- Reduced Hardware costs.
- **Huge** data centers.

# Containerized data centers

<https://www.ibm.com/us-en/marketplace/prefabricated-modular-data-center>



# Google data center in Hamina, Finland

<https://www.google.com/about/datacenters/locations/hamina/>



# Microsoft's underwater data center

<https://news.microsoft.com/innovation-stories/project-natick-underwater-datacenter/>



- Software as a Service (SaaS)

- **Software as a Service (SaaS)**

The service vendor provides the user with a completed product that is run and managed by the service provider.

- **Software as a Service (SaaS)**

The service vendor provides the user with a completed product that is run and managed by the service provider.

- **Platform as a Service (PaaS)**

- **Software as a Service (SaaS)**

The service vendor provides the user with a completed product that is run and managed by the service provider.

- **Platform as a Service (PaaS)**

The service vendor provides the user with a set of API which can be used to build, test and deploy applications.

- **Software as a Service (SaaS)**

The service vendor provides the user with a completed product that is run and managed by the service provider.

- **Platform as a Service (PaaS)**

The service vendor provides the user with a set of API which can be used to build, test and deploy applications.

- **Infrastructure as a Service (IaaS)**

- **Software as a Service (SaaS)**

The service vendor provides the user with a completed product that is run and managed by the service provider.

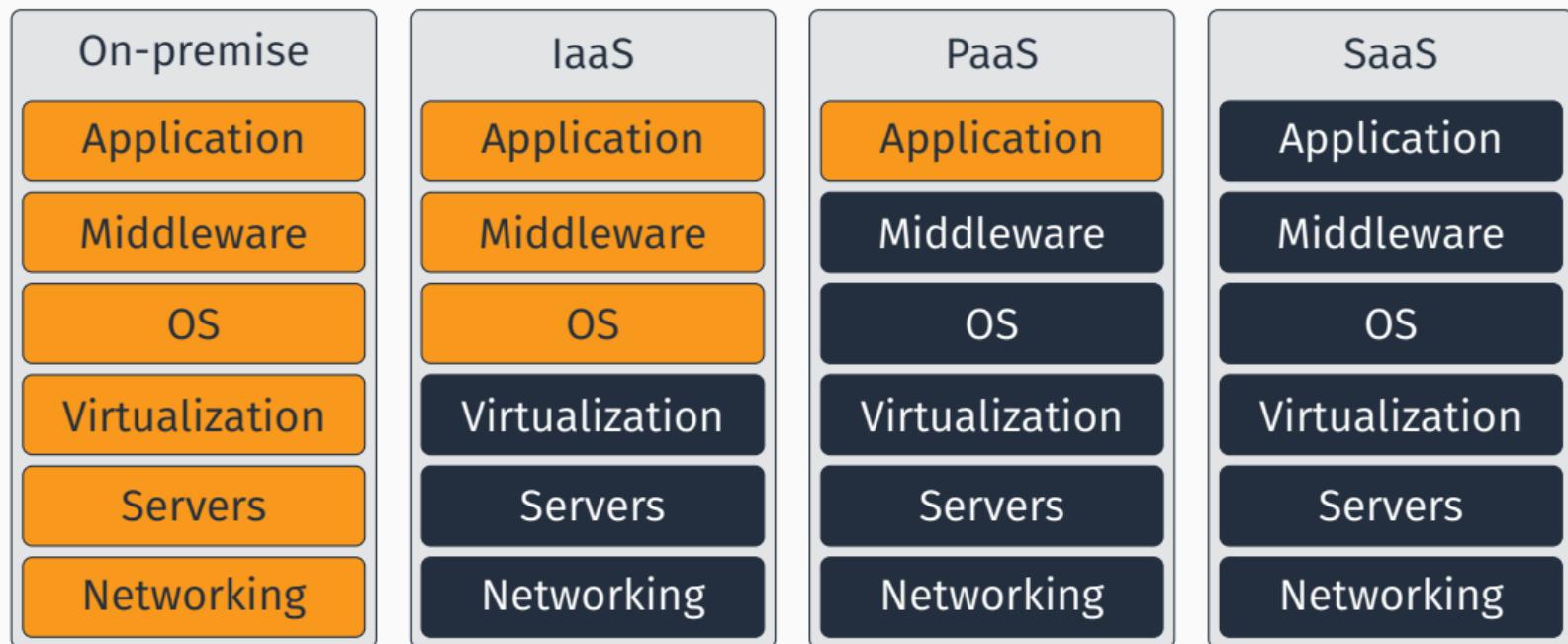
- **Platform as a Service (PaaS)**

The service vendor provides the user with a set of API which can be used to build, test and deploy applications.

- **Infrastructure as a Service (IaaS)**

The service vendor provides users access to computing resources such as servers, storage and networking.

# SERVICE MODELS: A VISUAL COMPARISON



User manages



Someone else manages

## AWS INFRASTRUCTURE

---

# AWS INFRASTRUCTURE

- AWS is present in 22 geographical regions.



# AWS INFRASTRUCTURE

- AWS is present in 22 geographical regions.
- Each region consists of multiple (3–6) availability zones.



# AWS INFRASTRUCTURE

- AWS is present in 22 geographical regions.
- Each region consists of multiple (3–6) availability zones.
- An availability zone can be multiple data centers, with up to hundreds of thousands of servers.



- AWS is present in 22 geographical regions.
- Each region consists of multiple (3–6) availability zones.
- An availability zone can be multiple data centers, with up to hundreds of thousands of servers.
- 216 points of presence for effective caching and content delivery.



## CORE AWS SERVICES

---



Amazon Web Services is a collection of cloud-based services.



Amazon Web Services is a collection of cloud-based services. **A very big one.**



Amazon Web Services is a collection of cloud-based services. **A VERY big one.**

# Periodic Table of Amazon Web Services

WPS Office

Periodic Table of Amazon Web Services																		
Amazon A		Amazon S3		Amazon Sg		Amazon storage services		Amazon Sg		Amazon storage		Amazon Sg		Amazon storage		Amazon Sg		
Amazon CloudSearch	Amazon GS ground station	Amazon Analytics	Amazon Application Integration	Amazon CompRules	Amazon Customer Engagement	Amazon Database	Amazon DevTools	Amazon IoT	Amazon Machine Learning	Amazon Media Services	Amazon Migration & Transfer	Amazon Mobile	Amazon Network & Content Delivery	Amazon Robotics	Amazon Satellite	Amazon Storage	Amazon Security, Identity & Compliance	
Amazon MR	Amazon Gf glue	Amazon AR & VR	Amazon AWS Cost Management	Amazon Blockchain	Amazon Business Applications	Amazon Database	Amazon Developer Tools	Amazon End User Computing	Amazon Game Tech	Amazon Internet of Things	Amazon Machine Learning	Amazon Migration & Transfer	Amazon Mobile	Amazon Network & Content Delivery	Amazon Robotics	Amazon Satellite	Amazon Storage	Amazon Security, Identity & Compliance
Amazon Es	Amazon Lf lake formation	Amazon Ce cost explorer	Amazon Ax price for business	Amazon Cs AWS Lambda	Amazon L lambda	Amazon Co connect	Amazon Cb codebuild	Amazon Au metrics	Amazon Ds relational database service	Amazon I iot core	Amazon Dd IoT device connector	Amazon Ac auto scaling	Amazon Ms managed services	Amazon Mc managed cloud services	Amazon Kv media media services	Amazon Ad application discovery service	Amazon Am amplify	Amazon V virtual private cloud
Amazon Kines	Amazon Sf step functions	Amazon Bu budgets	Amazon Ch chime	Amazon Ks step functions	Amazon O insights	Amazon Pi pinpoint	Amazon Cc codecommit	Amazon D dynamoDB	Amazon Rs redshift	Amazon Os IoT device free tier	Amazon Dm IoT device management	Amazon Cf cloudformation	Amazon Ow appworks	Amazon Mv managed workflow	Amazon Dm database migration service	Amazon Po pipeline	Amazon Fx fix for faster	Amazon 53 route 53
Amazon Ka	Amazon Mg machine learning	Amazon Cu cost & usage report	Amazon Wm workmail	Amazon Ls lightail	Amazon Sa media application controller	Amazon E simple email service	Amazon Cd codedeploy	Amazon Do documentdb	Amazon Ti timestream	Amazon Gg greengrass	Amazon Ev IoT events	Amazon Ct cloudtrail	Amazon Ph personal health information	Amazon Sm managed migration services	Amazon Df device farm	Amazon Pl video processing	Amazon Fw file sync for serverless	Amazon Gi s3 glacier
Amazon R	Amazon N simple notification service	Amazon Ri simple queue service	Amazon C elastic cloud compute	Amazon Ba batch	Amazon Ls elastic load balancing	Amazon Cs codestar	Amazon Cp codepipeline	Amazon Ec elastic cache	Amazon Ms mobile analytics service	Amazon Lc IoT device connect	Amazon Sw IoT siteadvisor	Amazon Co config	Amazon Sc service catalog	Amazon Mp managed migration	Amazon Sn game family	Amazon Ga global acceleration	Amazon Cm cloud map	Amazon Ba backup
Amazon Qs	Amazon Q quicksight	Amazon Bc Amazon business intelligence	Amazon Au AWS Lambda	Amazon Eb Amazon business intelligence	Amazon Vm Amazon managed machine learning	Amazon Co connect	Amazon Clj command line interface	Amazon Ne Neptune	Amazon Gl game lift	Amazon An IoT analytics	Amazon Tg IoT thing graph	Amazon Ct control tower	Amazon Sm systems manager	Amazon Ms managed migration	Amazon SF transfer for drift	Amazon Tg transit gateway	Amazon Dc direct connect	Amazon Sn snow family
Amazon Dp	Amazon Ap appsync	Amazon Su samplers	Amazon Fx forge	Amazon Rm rolesmaker	Amazon Cg cloudy	Amazon Sd fonts & styles	Amazon Oi Amazon ledger	Amazon Ly lumberyard	Amazon Bu IoT button	Amazon Dc IoT partner device catalog	Amazon Cm IoT developer kit	Amazon Ta hybrid database	Amazon Mt managed migration	Amazon Wd workdocs	Amazon Wo worklink	Amazon Ws workspaces	Amazon As synchronize 2.0	
Amazon Sm	Amazon Co comprehend	Amazon Ei elastic inference	Amazon Fc forecast	Amazon Lx lex	Amazon Pe personalize	Amazon Po poly	Amazon Rk rekognition	Amazon Gt image analysis	Amazon Tx texttract	Amazon Tr translate	Amazon Dr deep learning	Amazon Di deeplore	Amazon If inferentia	Amazon Mx open source on aws	Amazon Tf tensor flow	Amazon Sm	Amazon As synchronize 2.0	
Amazon Im	Amazon Cd identity access management	Amazon Co cognito	Amazon Gd guarduty	Amazon I inspector	Amazon M machine	Amazon Ar artifact	Amazon Cm codeartifact manager	Amazon Hs cloudhsm	Amazon Ds directory service	Amazon Fm familial manager	Amazon Km key management service	Amazon O organization	Amazon Sm secrets manager	Amazon Sh security hub	Amazon S shield	Amazon Ss single sign-on	Amazon Wf web application firewall	

## CORE AWS SERVICES

---

COMPUTING AND STORAGE

# AMAZON ELASTIC COMPUTE CLOUD (EC2)

- (Virtual) Servers on demand



---

EC2 pricing [web](#)

Azure: Virtual Machines [web](#)

Google Cloud: Compute Engine [web](#)

# AMAZON ELASTIC COMPUTE CLOUD (EC2)

- (Virtual) Servers on demand
- Different types of instances to suit computing needs



---

EC2 pricing [↗ web](#)

Azure: Virtual Machines [↗ web](#)

Google Cloud: Compute Engine [↗ web](#)

# AMAZON ELASTIC COMPUTE CLOUD (EC2)

- (Virtual) Servers on demand
- Different types of instances to suit computing needs
- Per-second (or per-hour) billing



---

EC2 pricing [↗ web](#)

Azure: Virtual Machines [↗ web](#)

Google Cloud: Compute Engine [↗ web](#)

# AMAZON ELASTIC COMPUTE CLOUD (EC2)

- (Virtual) Servers on demand
- Different types of instances to suit computing needs
- Per-second (or per-hour) billing
- Data transfer **not** included!



---

EC2 pricing [↗ web](#)

Azure: Virtual Machines [↗ web](#)

Google Cloud: Compute Engine [↗ web](#)

# AMAZON ELASTIC COMPUTE CLOUD (EC2)

- (Virtual) Servers on demand
- Different types of instances to suit computing needs
- Per-second (or per-hour) billing
- Data transfer **not** included!
- Persistent storage **not** included!



---

EC2 pricing [↗ web](#)

Azure: Virtual Machines [↗ web](#)

Google Cloud: Compute Engine [↗ web](#)

# AMAZON ELASTIC COMPUTE CLOUD (EC2)

- (Virtual) Servers on demand
- Different types of instances to suit computing needs
- Per-second (or per-hour) billing
- Data transfer **not** included!
- Persistent storage **not** included!
- Elasticity **not** included!



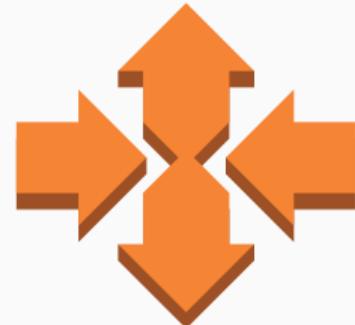
---

EC2 pricing [↗ web](#)

Azure: Virtual Machines [↗ web](#)

Google Cloud: Compute Engine [↗ web](#)

- *Scaling is the ability to increase or decrease the compute capacity of your application*

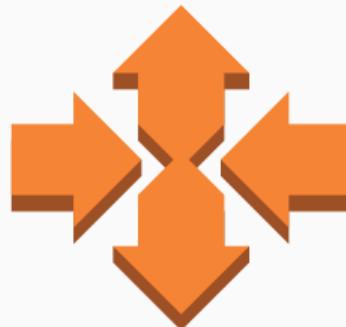


---

Azure: Virtual Machine Scale Sets [\[web\]](#)  
Google Cloud: Load Balancing [\[web\]](#)

# AMAZON EC2 AUTO SCALING

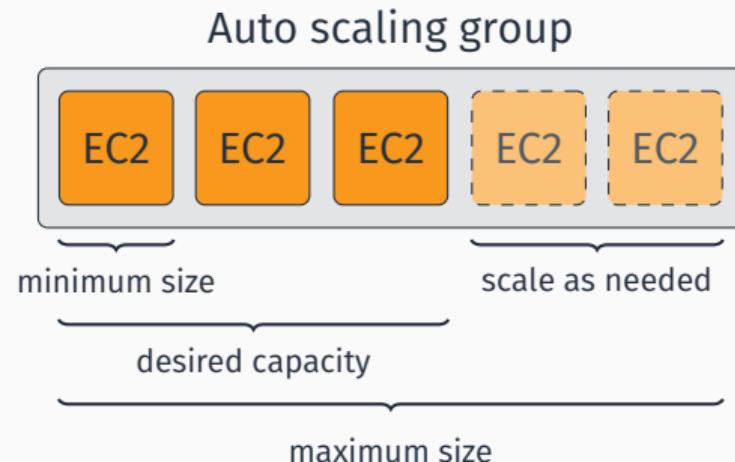
- *Scaling is the ability to increase or decrease the compute capacity of your application*
- Scale your application manually, on a scheduled basis or on demand



---

Azure: Virtual Machine Scale Sets  web  
Google Cloud: Load Balancing  web

## AMAZON EC2 AUTO SCALING: DETAILS



## AMAZON ELASTIC LOAD BALANCING (ELB)

- Distributes incoming traffic across multiple EC2 instances



---

Azure: Load Balancer  web

## AMAZON ELASTIC LOAD BALANCING (ELB)

- Distributes incoming traffic across multiple EC2 instances
- Pay-per-use billing



---

Azure: Load Balancer  web

# AMAZON ELASTIC LOAD BALANCING (ELB)

- Distributes incoming traffic across multiple EC2 instances
- Pay-per-use billing
  - Execution time



---

Azure: Load Balancer  web

# AMAZON ELASTIC LOAD BALANCING (ELB)

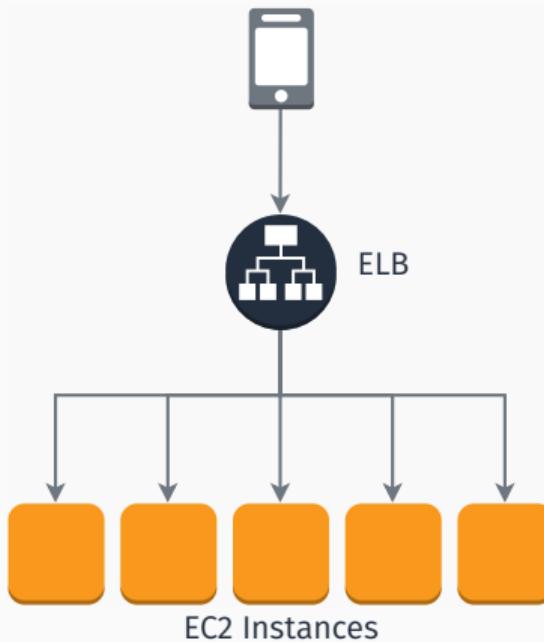
- Distributes incoming traffic across multiple EC2 instances
- Pay-per-use billing
  - Execution time
  - Number of requests / traffic



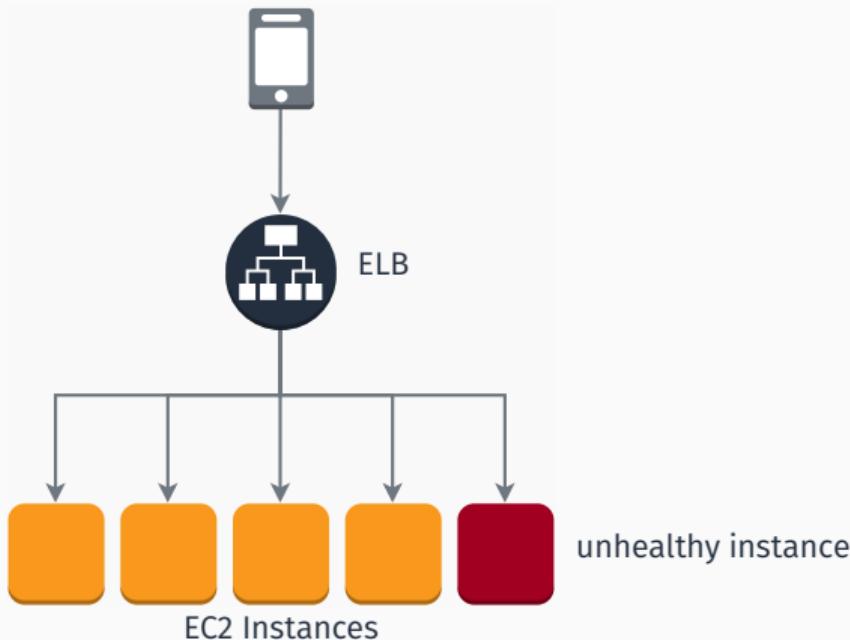
---

Azure: Load Balancer  web

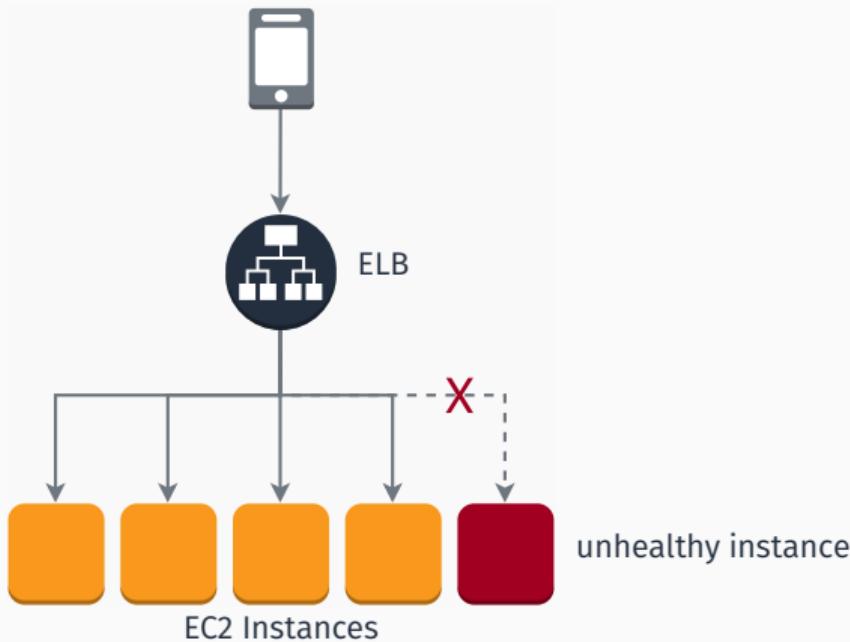
# AMAZON ELASTIC LOAD BALANCING (ELB)



# AMAZON ELASTIC LOAD BALANCING (ELB)



# AMAZON ELASTIC LOAD BALANCING (ELB)



## CLOUD STORAGE PRODUCTS

- Elastic Block Storage (EBS)
  - Persistent local storage for EC2 instances.

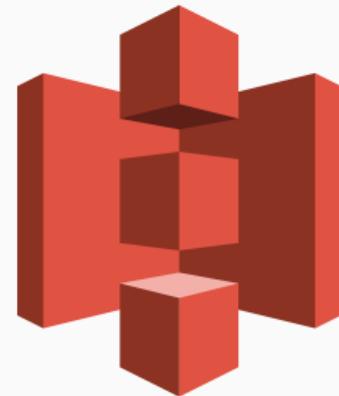


## CLOUD STORAGE PRODUCTS

- Elastic Block Storage (EBS)
  - Persistent local storage for EC2 instances.
- Elastic File System (EFS)
  - File system interface to share data between EC2 instances.

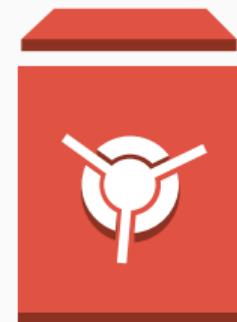


- Elastic Block Storage (EBS)
  - Persistent local storage for EC2 instances.
- Elastic File System (EFS)
  - File system interface to share data between EC2 instances.
- Simple Storage Service (S3)



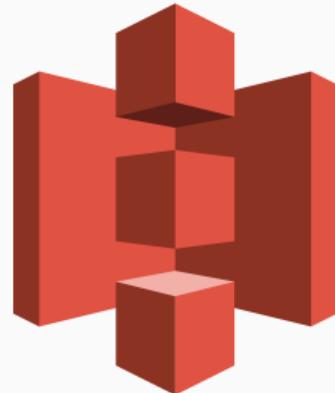
## CLOUD STORAGE PRODUCTS

- Elastic Block Storage (EBS)
  - Persistent local storage for EC2 instances.
- Elastic File System (EFS)
  - File system interface to share data between EC2 instances.
- Simple Storage Service (S3)
- Glacier
  - Durable and cheap long-term storage.



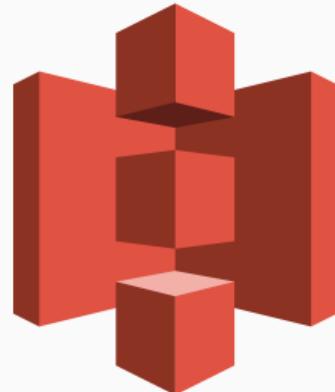
## AMAZON SIMPLE STORAGE SERVICE (S3)

- *store and retrieve any amount of data from anywhere*



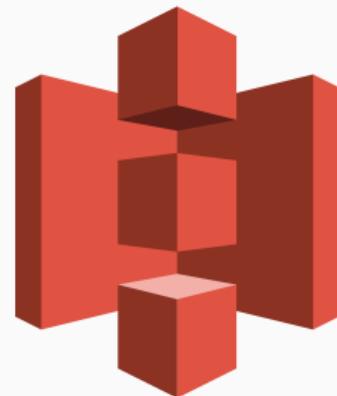
## AMAZON SIMPLE STORAGE SERVICE (S3)

- *store and retrieve any amount of data from anywhere*
- 99.99999999% durability (nine nines!)



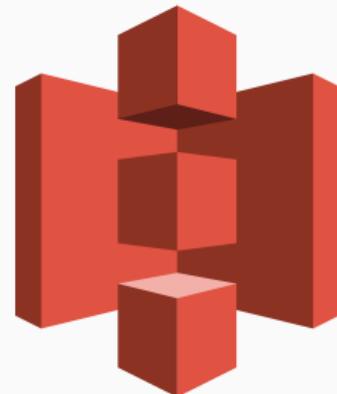
## AMAZON SIMPLE STORAGE SERVICE (S3)

- *store and retrieve any amount of data from anywhere*
- 99.99999999% durability (nine nines!)
- Data is distributed across a *minimum* of three availability zones



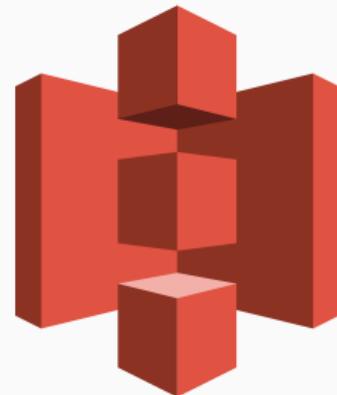
## AMAZON SIMPLE STORAGE SERVICE (S3)

- *store and retrieve any amount of data from anywhere*
- 99.99999999% durability (nine nines!)
- Data is distributed across a *minimum* of three availability zones
- A logical unit of storage is a *bucket*



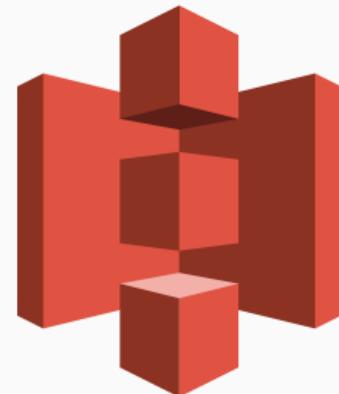
## AMAZON SIMPLE STORAGE SERVICE (S3)

- *store and retrieve any amount of data from anywhere*
- 99.99999999% durability (nine nines!)
- Data is distributed across a *minimum* of three availability zones
- A logical unit of storage is a *bucket*
- Multiple storage classes



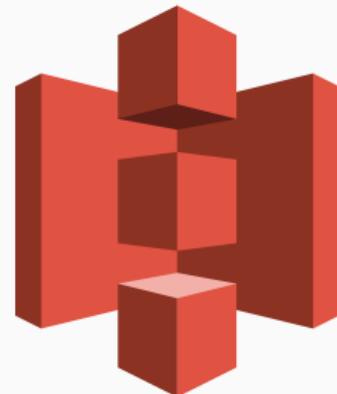
## AMAZON SIMPLE STORAGE SERVICE (S3)

- *store and retrieve any amount of data from anywhere*
- 99.99999999% durability (nine nines!)
- Data is distributed across a *minimum* of three availability zones
- A logical unit of storage is a *bucket*
- Multiple storage classes
  - Standard



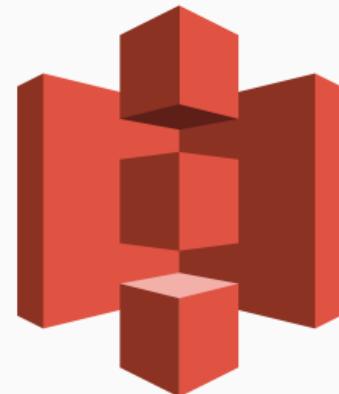
## AMAZON SIMPLE STORAGE SERVICE (S3)

- *store and retrieve any amount of data from anywhere*
- 99.99999999% durability (nine nines!)
- Data is distributed across a *minimum* of three availability zones
- A logical unit of storage is a *bucket*
- Multiple storage classes
  - Standard
  - Infrequent Access



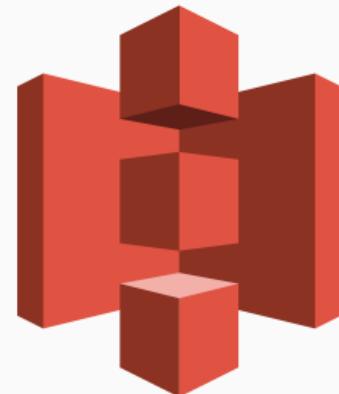
## AMAZON SIMPLE STORAGE SERVICE (S3)

- *store and retrieve any amount of data from anywhere*
- 99.99999999% durability (nine nines!)
- Data is distributed across a *minimum* of three availability zones
- A logical unit of storage is a *bucket*
- Multiple storage classes
  - Standard
  - Infrequent Access
  - One zone-Infrequent Access



## AMAZON SIMPLE STORAGE SERVICE (S3)

- *store and retrieve any amount of data from anywhere*
- 99.99999999% durability (nine nines!)
- Data is distributed across a *minimum* of three availability zones
- A logical unit of storage is a *bucket*
- Multiple storage classes
  - Standard
  - Infrequent Access
  - One zone-Infrequent Access
  - Amazon Glacier



## AMAZON SIMPLE STORAGE SERVICE (S3) - MORE

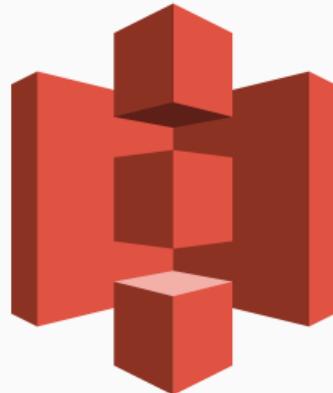
Pricing:

Storage class	Storage (per month)	Retrieval (per 1K req.)
Standard	\$0.022 per GB	\$0.0004
Infrequent access	\$0.0125 per GB	\$0.001
IA single zone	\$0.01 per GB	\$0.001
Glacier	\$0.004 per GB	\$0.0004

Table 1: S3 pricing (Ireland)

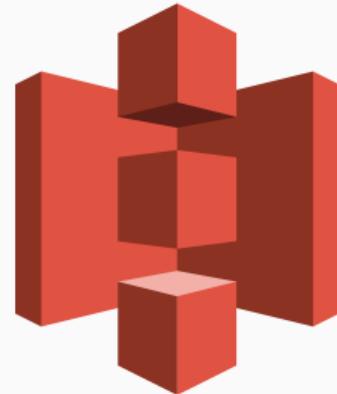
## AMAZON SIMPLE STORAGE SERVICE (S3) - MORE

- Well-integrated with other services



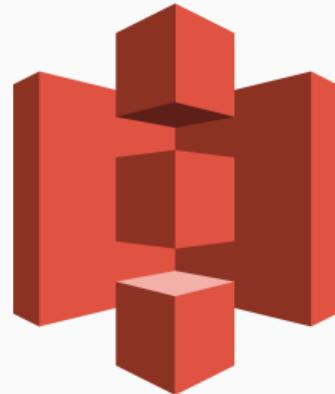
## AMAZON SIMPLE STORAGE SERVICE (S3) - MORE

- Well-integrated with other services
  - Machine Learning



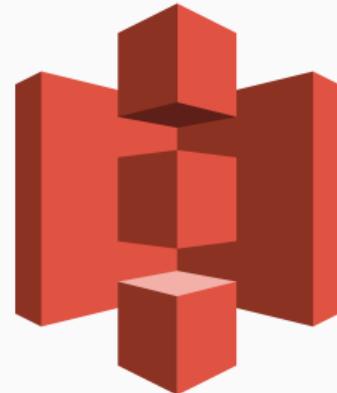
## AMAZON SIMPLE STORAGE SERVICE (S3) - MORE

- Well-integrated with other services
  - Machine Learning
  - Big Data Analysis



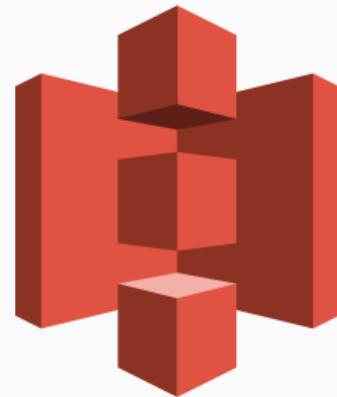
## AMAZON SIMPLE STORAGE SERVICE (S3) - MORE

- Well-integrated with other services
  - Machine Learning
  - Big Data Analysis
- REST API

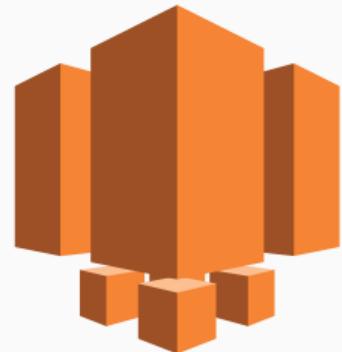


## AMAZON SIMPLE STORAGE SERVICE (S3) - MORE

- Well-integrated with other services
  - Machine Learning
  - Big Data Analysis
- REST API
- Can be used to host static websites



- A lightweight, simplified offer

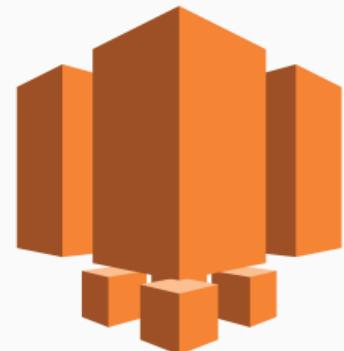


---

Websites: [EC2](#) [Lightsail](#)

## AMAZON LIGHTSAIL

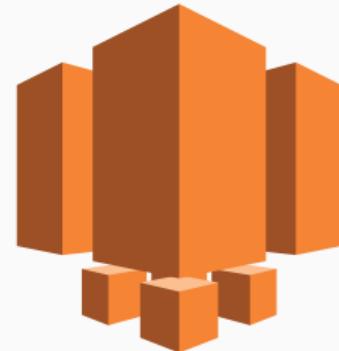
- A lightweight, simplified offer
- Bundles computing, storage, and networking capacity



---

Websites: [EC2](#) [Lightsail](#)

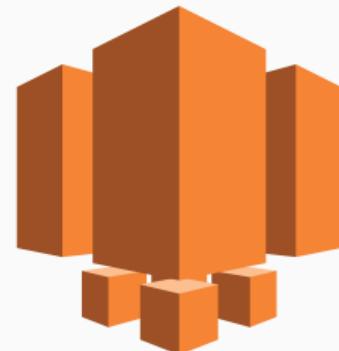
- A lightweight, simplified offer
- Bundles computing, storage, and networking capacity
- Preconfigured instances for



---

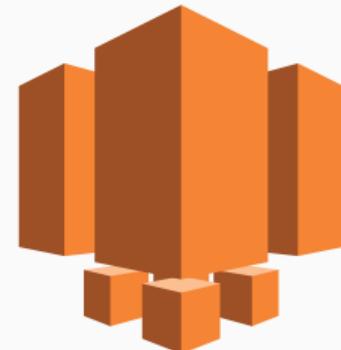
Websites: [EC2](#) [Lightsail](#)

- A lightweight, simplified offer
- Bundles computing, storage, and networking capacity
- Preconfigured instances for
  - Debian, Windows Server, ...



Websites: [EC2](#) [Lightsail](#)

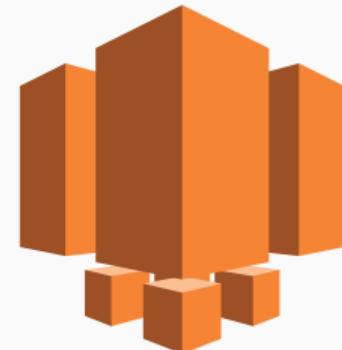
- A lightweight, simplified offer
- Bundles computing, storage, and networking capacity
- Preconfigured instances for
  - Debian, Windows Server, ...
  - Wordpress, Magento, Redmine, ...



---

Websites: [EC2](#) [Lightsail](#)

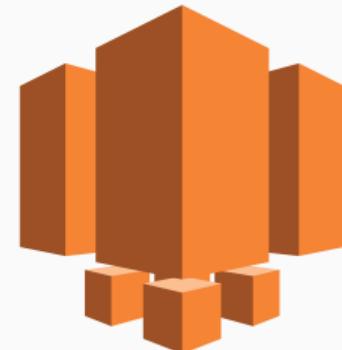
- A lightweight, simplified offer
- Bundles computing, storage, and networking capacity
- Preconfigured instances for
  - Debian, Windows Server, ...
  - Wordpress, Magento, Redmine, ...
  - LAMP stack, Nginx, ...



---

Websites: [EC2](#) [Lightsail](#)

- A lightweight, simplified offer
- Bundles computing, storage, and networking capacity
- Preconfigured instances for
  - Debian, Windows Server, ...
  - Wordpress, Magento, Redmine, ...
  - LAMP stack, Nginx, ...
- Low and **predictable** monthly costs



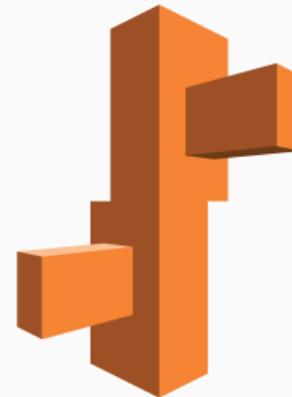
---

Websites: [EC2](#) [Lightsail](#)

## AMAZON ELASTIC BEANSTALK

---

- “Easy to begin, impossible to outgrow”



## AMAZON ELASTIC BEANSTALK

---

- “*Easy to begin, impossible to outgrow*”
- Easy-to-use service to deploy web apps



## AMAZON ELASTIC BEANSTALK

- “*Easy to begin, impossible to outgrow*”
- Easy-to-use service to deploy web apps
- Supports Apache, Nginx, IIS and more



- “Easy to begin, impossible to outgrow”
- Easy-to-use service to deploy web apps
- Supports Apache, Nginx, IIS and more
- Supports Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker



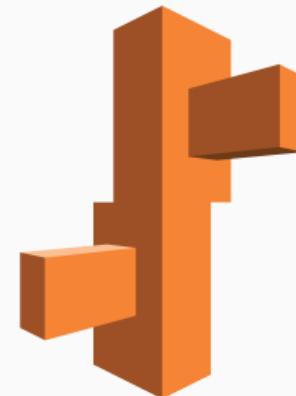
## AMAZON ELASTIC BEANSTALK

- “Easy to begin, impossible to outgrow”
- Easy-to-use service to deploy web apps
- Supports Apache, Nginx, IIS and more
- Supports Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker
- Manages auto-scaling, load balancing, health monitoring



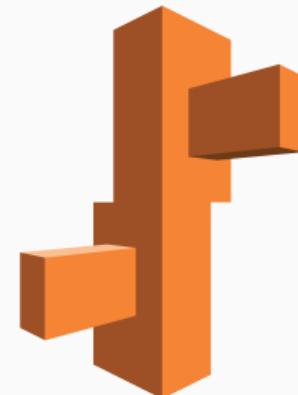
## AMAZON ELASTIC BEANSTALK

- “Easy to begin, impossible to outgrow”
- Easy-to-use service to deploy web apps
- Supports Apache, Nginx, IIS and more
- Supports Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker
- Manages auto-scaling, load balancing, health monitoring
- Customizable



## AMAZON ELASTIC BEANSTALK

- “Easy to begin, impossible to outgrow”
- Easy-to-use service to deploy web apps
- Supports Apache, Nginx, IIS and more
- Supports Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker
- Manages auto-scaling, load balancing, health monitoring
- Customizable
- Free of charge. Pay only for the AWS resources you use.



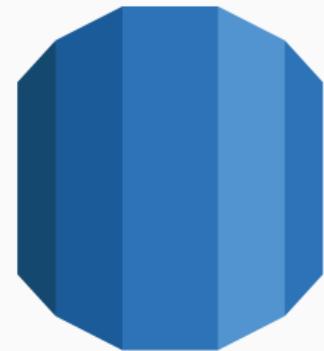
## CORE AWS SERVICES

---

### DATABASE SERVICES

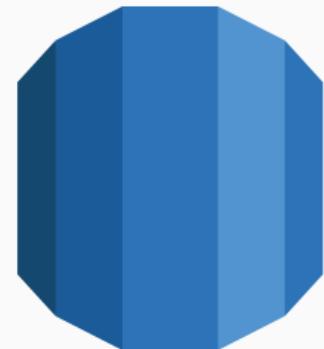
## RELATIONAL DATABASE SERVICE (RDS)

- Set up, operate a relational database in the cloud.



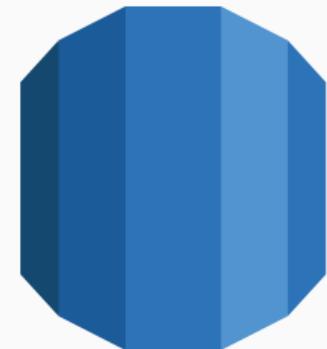
## RELATIONAL DATABASE SERVICE (RDS)

- Set up, operate a relational database in the cloud.
- Takes care of backups, patching.



## RELATIONAL DATABASE SERVICE (RDS)

- Set up, operate a relational database in the cloud.
- Takes care of backups, patching.
- Supports:



## RELATIONAL DATABASE SERVICE (RDS)

- Set up, operate a relational database in the cloud.
- Takes care of backups, patching.
- Supports:
  - MySQL, PostgreSQL, MariaDB



## RELATIONAL DATABASE SERVICE (RDS)

- Set up, operate a relational database in the cloud.
- Takes care of backups, patching.
- Supports:
  - MySQL, PostgreSQL, MariaDB
  - Oracle, MS SQL Server



## RELATIONAL DATABASE SERVICE (RDS)

- Set up, operate a relational database in the cloud.
- Takes care of backups, patching.
- Supports:
  - MySQL, PostgreSQL, MariaDB
  - Oracle, MS SQL Server
  - Amazon Aurora



## NON RELATIONAL DATABASE SERVICES

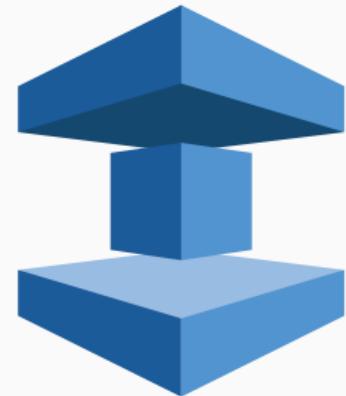
- DynamoDB

- *Fast and flexible NoSQL database service for any scale.*



# NON RELATIONAL DATABASE SERVICES

- DynamoDB
  - *Fast and flexible NoSQL database service for any scale.*
- ElastiCache
  - In memory data store.
  - Supports memcached, Redis



# NON RELATIONAL DATABASE SERVICES

- DynamoDB
  - *Fast and flexible NoSQL database service for any scale.*
- ElastiCache
  - In memory data store.
  - Supports memcached, Redis
- Neptune
  - Graph database service
  - Supports RDF, SPARQL, ...



## CORE AWS SERVICES

---

## DEVELOPER TOOLS

- CodeCommit



- CodeCommit
  - Managed, scalable, private *git* server



## ■ CodeCommit

- Managed, scalable, private *git* server
- Pricing based on active users (5 free, 1\$ for each additional user)



- CodeStar



- CodeStar
  - Wrapper around developer tools to simplify setup



- CodeStar

- Wrapper around developer tools to simplify setup
- Templates



- CodeStar

- Wrapper around developer tools to simplify setup
- Templates
- Team Management



### ■ CodeStar

- Wrapper around developer tools to simplify setup
- Templates
- Team Management
- Central Project Dashboard



### ■ CodeStar

- Wrapper around developer tools to simplify setup
- Templates
- Team Management
- Central Project Dashboard
- Free of charge



- CodeStar

- Wrapper around developer tools to simplify setup
- Templates
- Team Management
- Central Project Dashboard
- Free of charge

- Cloud9



- CodeStar

- Wrapper around developer tools to simplify setup
- Templates
- Team Management
- Central Project Dashboard
- Free of charge

- Cloud9

- Cloud-based full-fledged IDE



- CodeStar

- Wrapper around developer tools to simplify setup
- Templates
- Team Management
- Central Project Dashboard
- Free of charge

- Cloud9

- Cloud-based full-fledged IDE
- Runs in a web browser



### ■ CodeStar

- Wrapper around developer tools to simplify setup
- Templates
- Team Management
- Central Project Dashboard
- Free of charge

### ■ Cloud9

- Cloud-based full-fledged IDE
- Runs in a web browser
- Collaborative editing and chat



### ■ CodeStar

- Wrapper around developer tools to simplify setup
- Templates
- Team Management
- Central Project Dashboard
- Free of charge

### ■ Cloud9

- Cloud-based full-fledged IDE
- Runs in a web browser
- Collaborative editing and chat
- Greatly-integrated with AWS



### ■ CodeStar

- Wrapper around developer tools to simplify setup
- Templates
- Team Management
- Central Project Dashboard
- Free of charge

### ■ Cloud9

- Cloud-based full-fledged IDE
- Runs in a web browser
- Collaborative editing and chat
- Greatly-integrated with AWS
- Free of charge



## CORE AWS SERVICES

---

MACHINE LEARNING: APPLICATION SERVICES

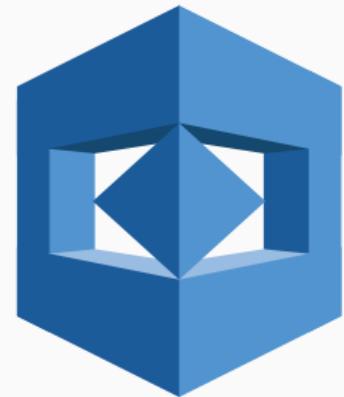
## MACHINE LEARNING: APPLICATION SERVICES

- Comprehend (for NLP) [!\[\]\(6131f092b405127b373957824d1d4fcb\_img.jpg\) website](#)



## MACHINE LEARNING: APPLICATION SERVICES

- Comprehend (for NLP) [website](#)
- Rekognition (Visual Analysis) [website](#)



## MACHINE LEARNING: APPLICATION SERVICES

- Comprehend (for NLP) [!\[\]\(b2dc5b514007e41ec1df5aee56b42de4\_img.jpg\) website](#)
- Rekognition (Visual Analysis) [!\[\]\(f8b2914c3dbe56db4c3079560ca59187\_img.jpg\) website](#)
- Translate

- Comprehend (for NLP) [!\[\]\(69b0ac0b0c2ed00f70ae8303ca5bdaee\_img.jpg\) website](#)
- Rekognition (Visual Analysis) [!\[\]\(1c3265e4f8f915df04c9dff79fb42f15\_img.jpg\) website](#)
- Translate
- Polly (text-to-speech)

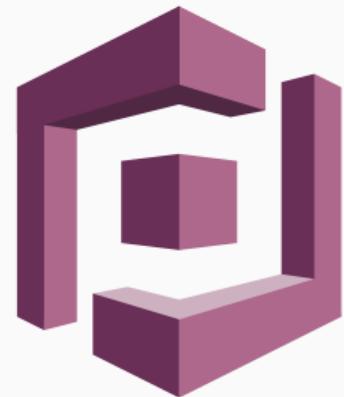
- Comprehend (for NLP) [!\[\]\(3ae2332930f6ac8d5075a4e055334d3e\_img.jpg\) website](#)
- Rekognition (Visual Analysis) [!\[\]\(8b83ed5d0ab109bf17eaf90356a48400\_img.jpg\) website](#)
- Translate
- Polly (text-to-speech)
- Transcribe (speech-to-text)

## CORE AWS SERVICES

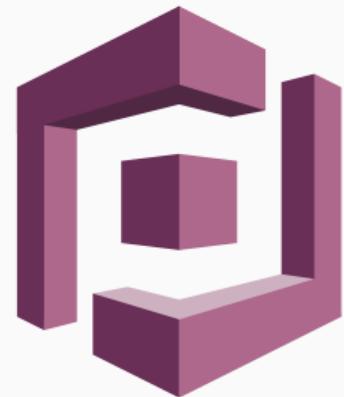
---

## MISCELLANEA

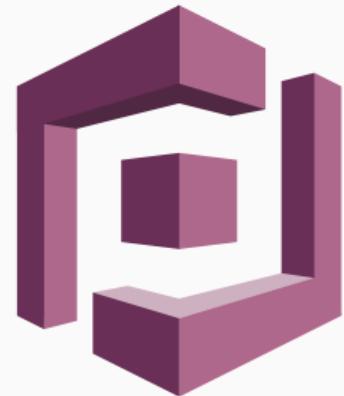
- Cognito



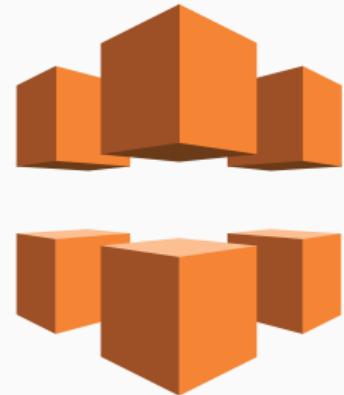
- Cognito
  - Sign-up and authentication



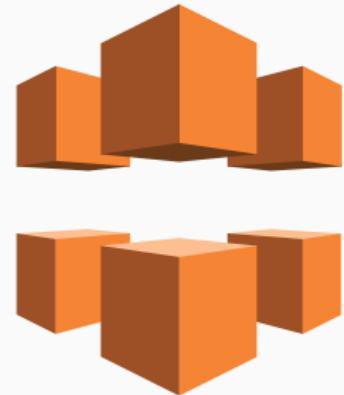
- Cognito
  - Sign-up and authentication
  - Federated identities



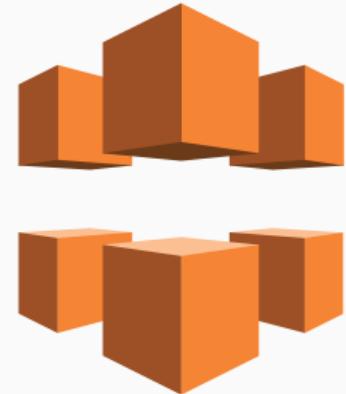
- Cognito
  - Sign-up and authentication
  - Federated identities
- CloudFront



- Cognito
  - Sign-up and authentication
  - Federated identities
- CloudFront
  - Content Delivery Network



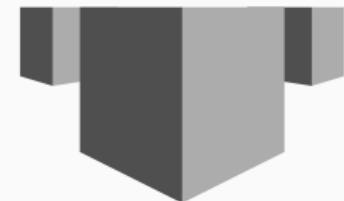
- Cognito
  - Sign-up and authentication
  - Federated identities
- CloudFront
  - Content Delivery Network
  - 116 Points of Presence in 56 cities across 24 countries



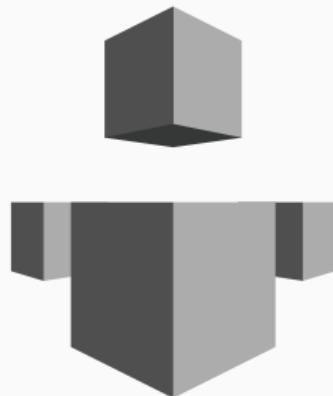
- Cognito
  - Sign-up and authentication
  - Federated identities



- CloudFront
  - Content Delivery Network
  - 116 Points of Presence in 56 cities across 24 countries
- Mechanical Turk



- ???

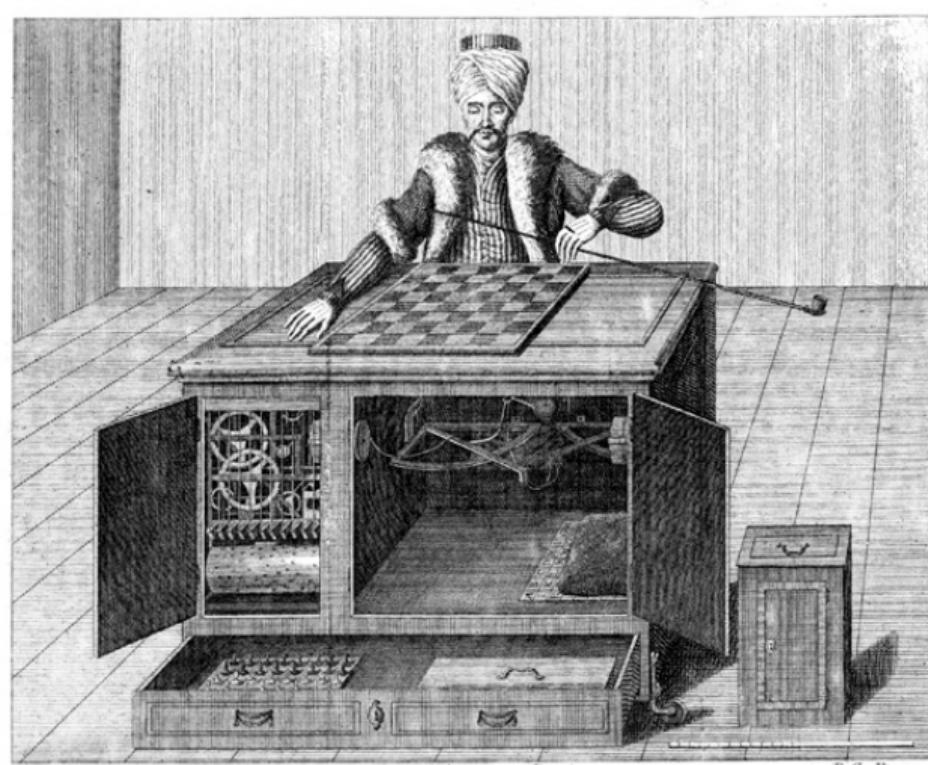


## THE TURK

*The Turk* was a chess-playing automaton built in 1770.

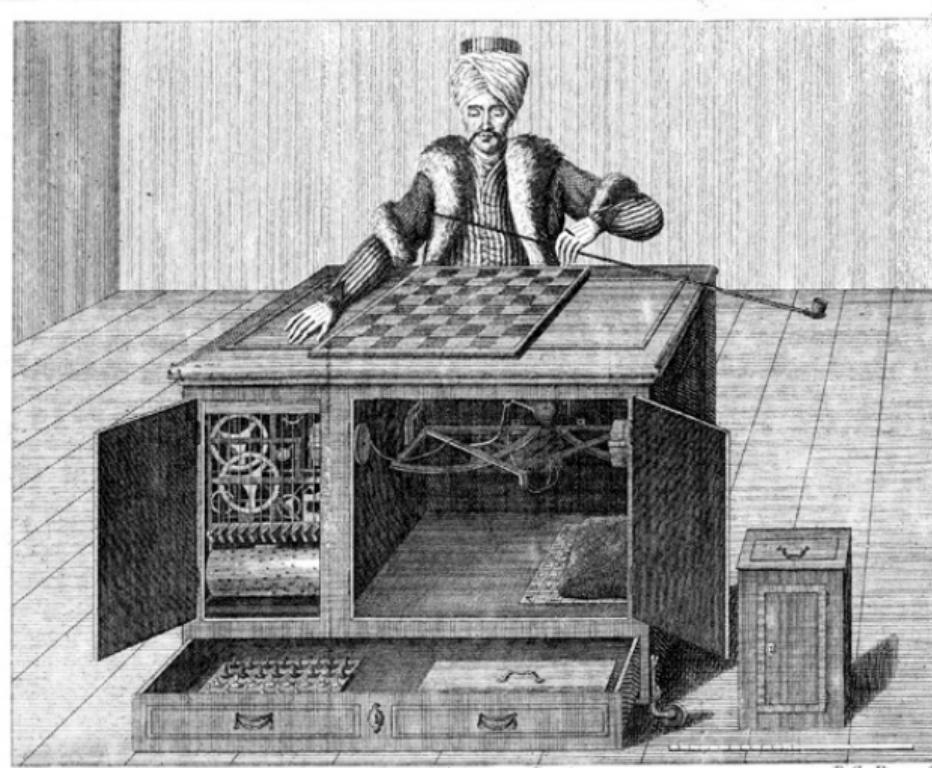
# THE TURK

*The Turk* was a chess-playing automaton built in 1770.

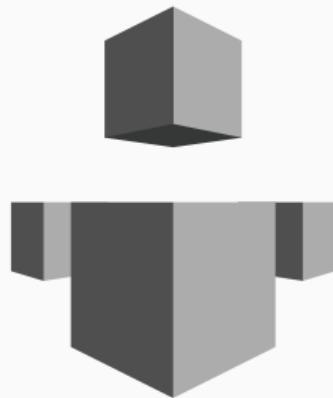


# THE TURK

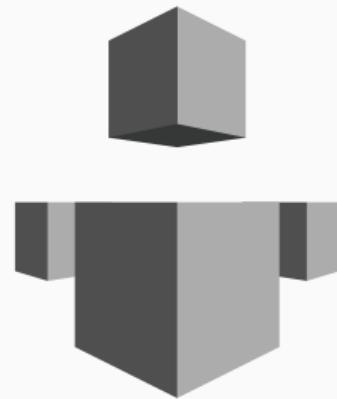
*The Turk* was a chess-playing automaton built in 1770. Obviously it was a fraud.



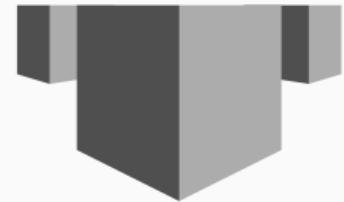
- ???



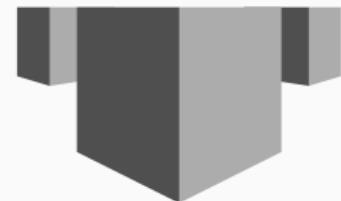
- Human Intelligence through an API



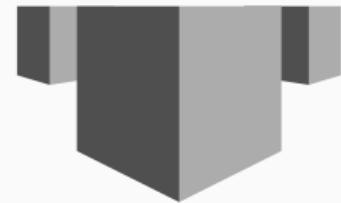
- Human Intelligence through an API
- Create HIT (Human Intelligence Task)



- Human Intelligence through an API
- Create HIT (Human Intelligence Task)
- Elastic, on-demand workforce



- Human Intelligence through an API
- Create HIT (Human Intelligence Task)
- Elastic, on-demand workforce
- Available 24/7





# Practice time!

## SCENARIO

---

- You just had a million dollar idea.

## SCENARIO

---

- You just had a million dollar idea.
- Your web application is finished. It looks great and works like a charm.

## SCENARIO

---

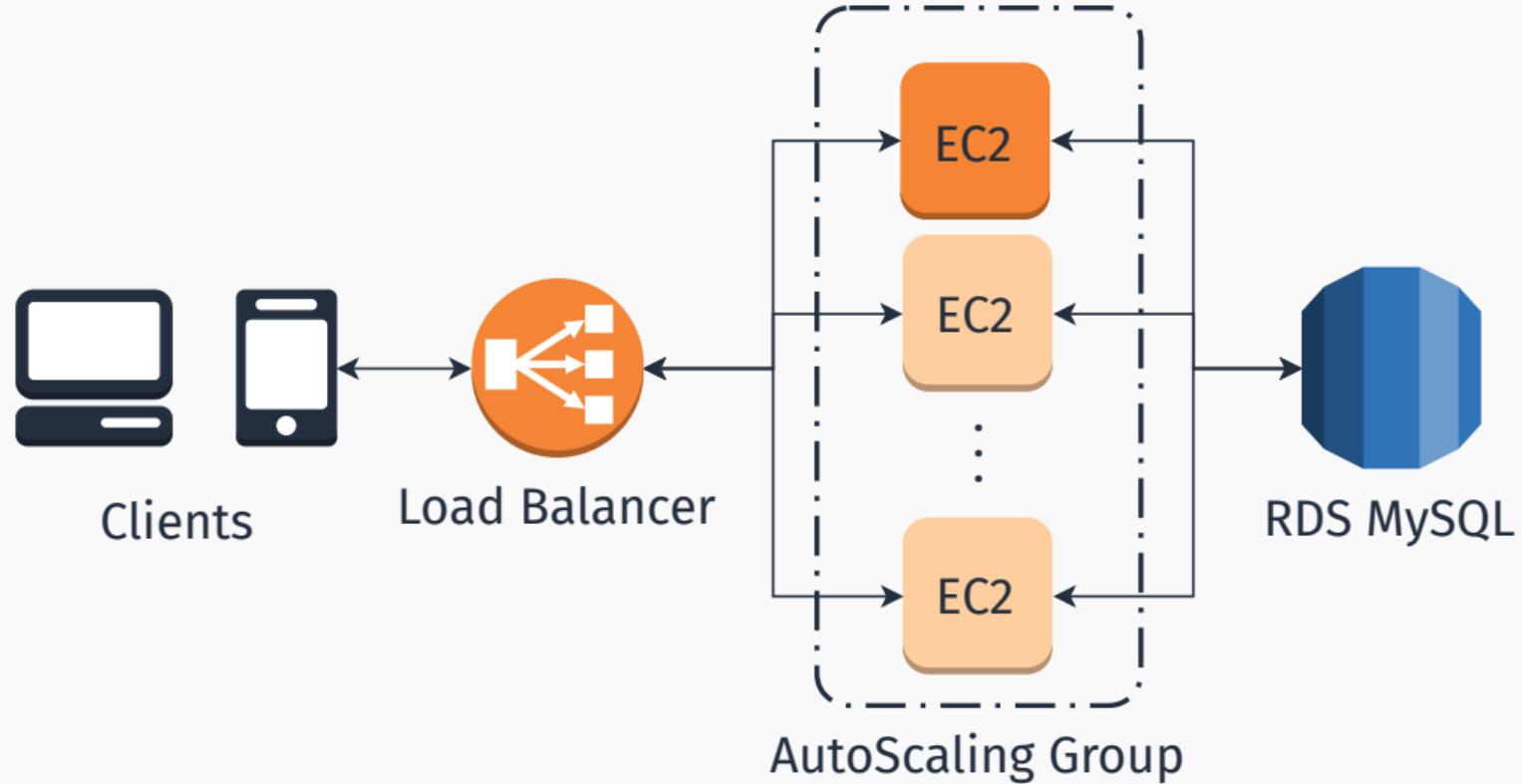
- You just had a million dollar idea.
- Your web application is finished. It looks great and works like a charm.
- You're ready to start earning some dough!

The web app is built with a classic LAMP stack:

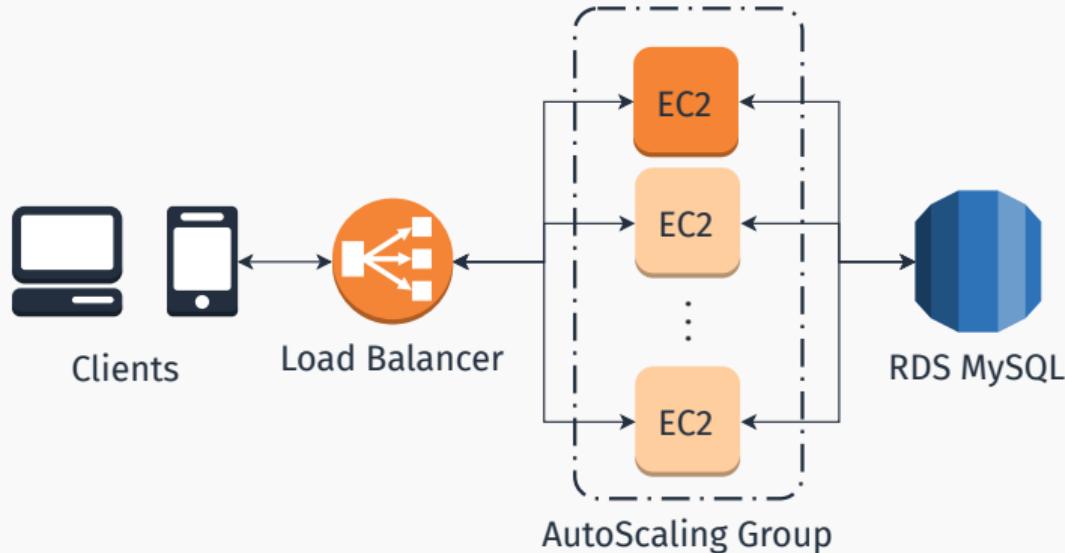
- Linux
- Apache web server
- MySQL relational database
- PHP

**How WOULD YOU DO IT?**

## PROPOSED ARCHITECTURE



## PROPOSED ARCHITECTURE



- Is this *really* scalable?

## TAKE HOME MESSAGES

---

## TAKE HOME MESSAGES

---

- Cloud computing and service models

## TAKE HOME MESSAGES

---

- Cloud computing and service models
- Core AWS services:

## TAKE HOME MESSAGES

---

- Cloud computing and service models
- Core AWS services:
  - Computing: EC2, AutoScaling groups, Lightsail

## TAKE HOME MESSAGES

---

- Cloud computing and service models
- Core AWS services:
  - Computing: EC2, AutoScaling groups, Lightsail
  - Storage: S3, EBS/EFS

## TAKE HOME MESSAGES

---

- Cloud computing and service models
- Core AWS services:
  - Computing: EC2, AutoScaling groups, Lightsail
  - Storage: S3, EBS/EFS
  - Database: RDS, DynamoDB

## TAKE HOME MESSAGES

---

- Cloud computing and service models
- Core AWS services:
  - Computing: EC2, AutoScaling groups, Lightsail
  - Storage: S3, EBS/EFS
  - Database: RDS, DynamoDB
  - Developer tools: CodeCommit, CodeStar, Cloud9

## TAKE HOME MESSAGES

---

- Cloud computing and service models
- Core AWS services:
  - Computing: EC2, AutoScaling groups, Lightsail
  - Storage: S3, EBS/EFS
  - Database: RDS, DynamoDB
  - Developer tools: CodeCommit, CodeStar, Cloud9
  - Machine Learning - Application Services: Comprehend, Rekognition;

## TAKE HOME MESSAGES

---

- Cloud computing and service models
- Core AWS services:
  - Computing: EC2, AutoScaling groups, Lightsail
  - Storage: S3, EBS/EFS
  - Database: RDS, DynamoDB
  - Developer tools: CodeCommit, CodeStar, Cloud9
  - Machine Learning - Application Services: Comprehend, Rekognition;
  - Others: Cognito, MechanicalTurk

## TAKE HOME MESSAGES

---

- Cloud computing and service models
- Core AWS services:
  - Computing: EC2, AutoScaling groups, Lightsail
  - Storage: S3, EBS/EFS
  - Database: RDS, DynamoDB
  - Developer tools: CodeCommit, CodeStar, Cloud9
  - Machine Learning - Application Services: Comprehend, Rekognition;
  - Others: Cognito, MechanicalTurk
- A cloud architecture for a classic web application on AWS

**ANY QUESTIONS?**

## REFERENCES I

- [Fle19] Flexera. *Cloud Computing Trends: 2019 State of the Cloud Survey*. Feb. 27, 2019. URL:  
<https://www.flexera.com/blog/cloud/2019/02/cloud-computing-trends-2019-state-of-the-cloud-survey/>  
(visited on 03/21/2020).

# An Introduction to Containerization for Software Engineers

**Luigi Libero Lucio Starace**

<https://luistar.github.io>

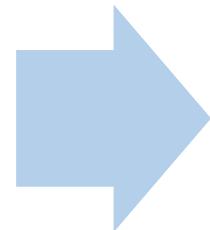
[luigiliberolucio.starace@unina.it](mailto:luigiliberolucio.starace@unina.it)

# Trends in the Software Industry

- The software industry has changed

Before:

- Monolithic software
- Long development cycles
- Single target environment
- Slowly scale up



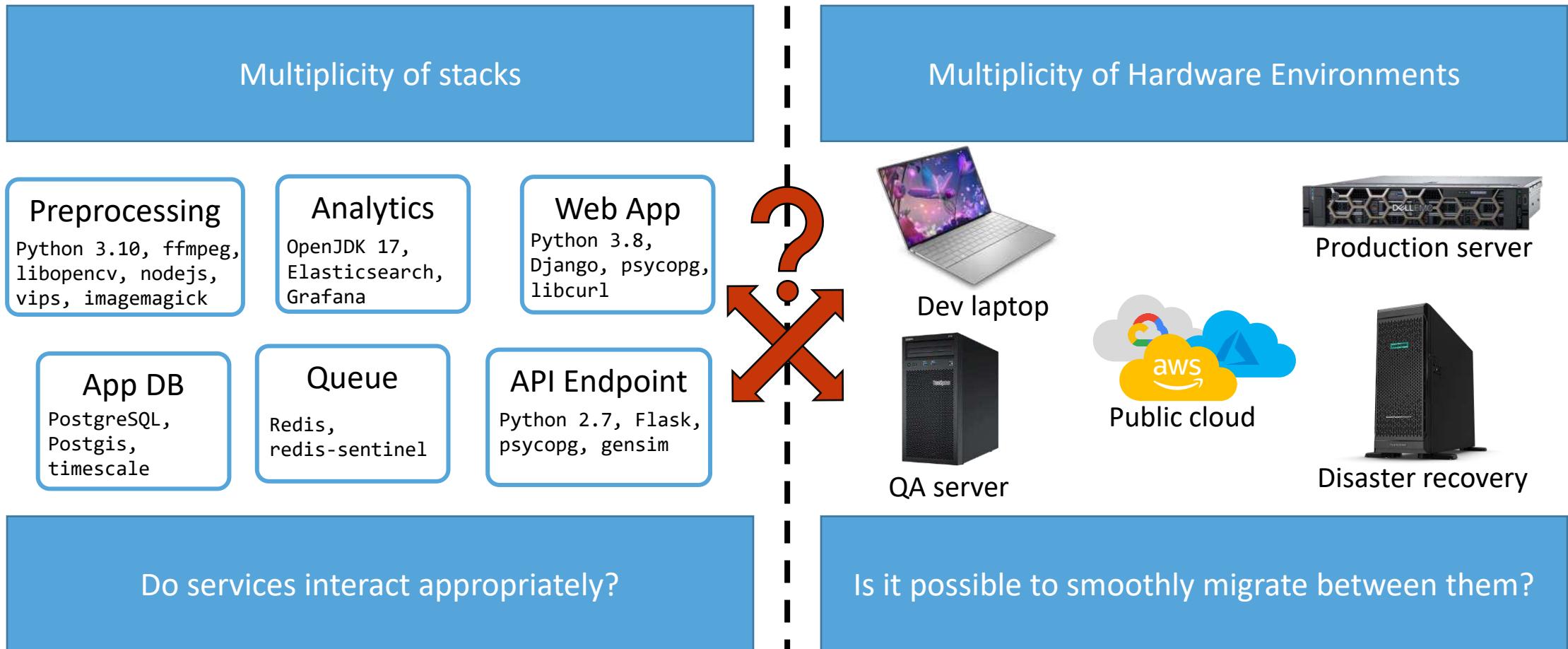
Now:

- Decoupled services
- Fast, iterative improvements
- Multiple target environments
- Must quickly scale up

# Deployments are becoming more complex

- Each independent service/components uses many stacks
  - Languages
  - Frameworks
  - Databases
- Many different targets
  - Development environments
  - Pre-production, QA, staging...
  - Production: On premises, public cloud, hybrid solutions

# The Challenge



# The «Matrix from Hell»

		Environments				
		Dev Laptop	Production Server	Distaster Recovery	Public Cloud	QA Server
Stacks	Web App	?	?	?	?	?
	API Endpoint	?	?	?	?	?
	Analytics	?	?	?	?	?
	App DB	?	?	?	?	?
	Queue	?	?	?	?	?
	Preprocessing	?	?	?	?	?

# Cargo Transportation before 1960s

Multiplicity of goods



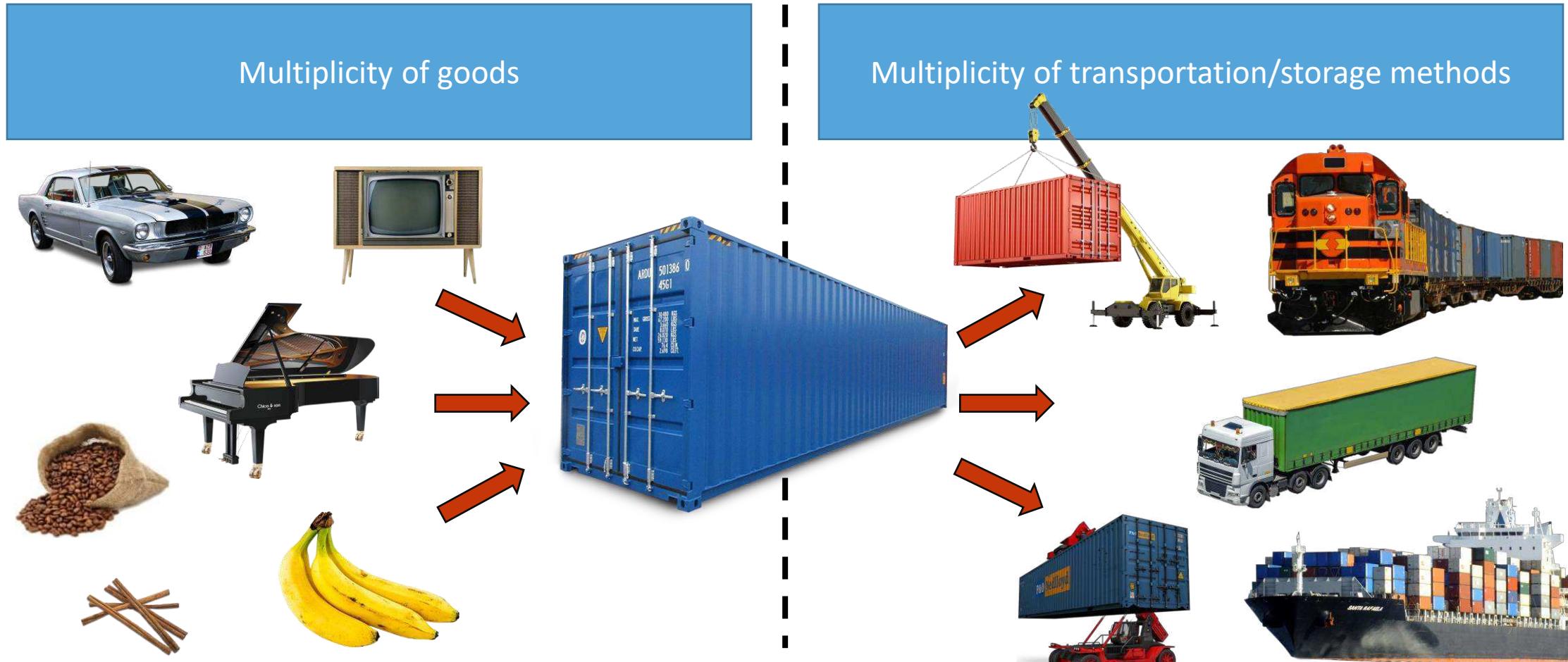
Multiplicity of transportation/storage methods



Do goods interact with each others?

Is it possible to smoothly change transport mode?

# Solution: Containers

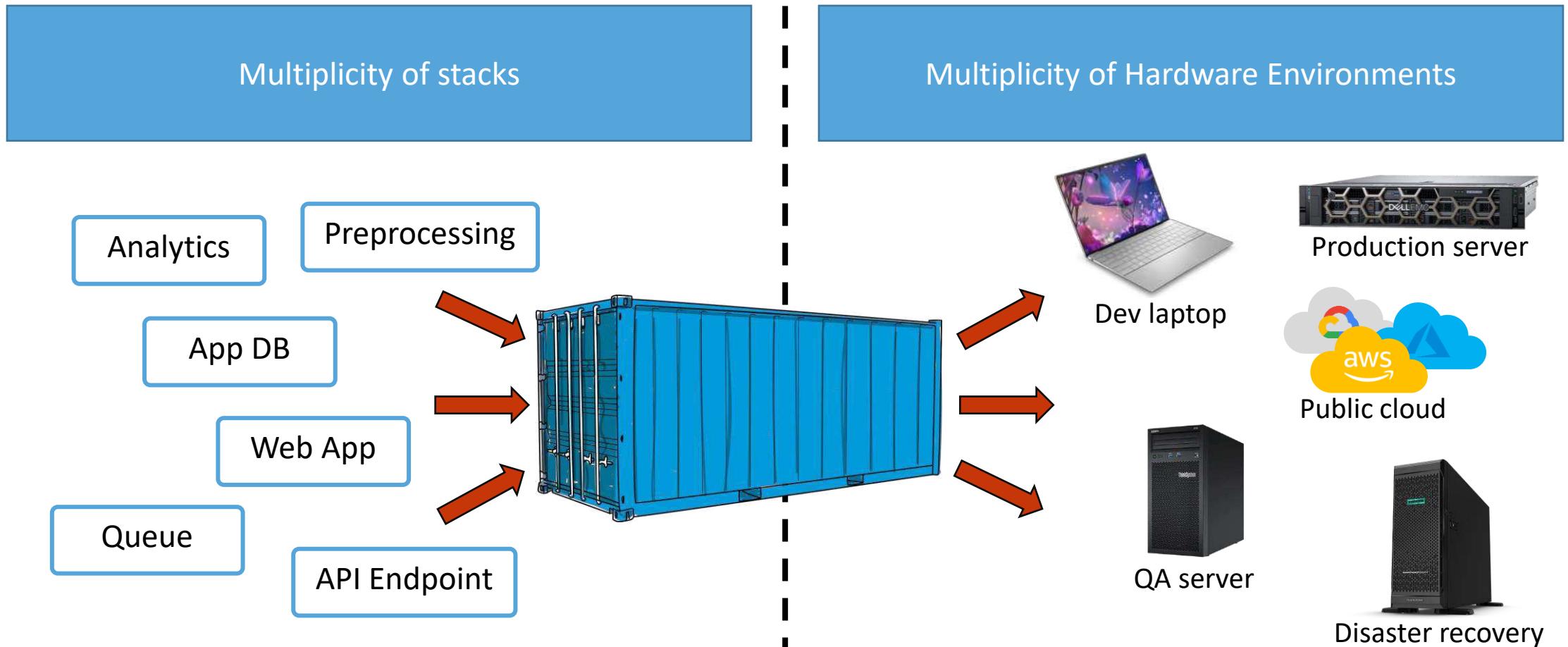


# Containers

- **Standardized** (all have the same size)
- Can be loaded with virtually any good
- Prepared by the people in charge of shipping
  - Make sure that no unwanted interactions happen **inside**
- Sealed until final delivery
- During transport, all containers are the same
  - Easy to load, unload, stack, etc..



# Containers for Code



# Why should we bother?

## Developers

- Only need to care about what's **inside** the container
- Simplify setup of dev env.
- No worries about library/dependencies conflicts
- **Build once, run anywhere\***

\*anywhere with the same architecture and a modern Linux kernel

## Operations

- Only need to care about what's **outside** the container
- Every container can be managed the same way
- Simplify lifecycle management
- **Configure once, run anything\*\***

\*\*anything built based on the same architecture and kernel

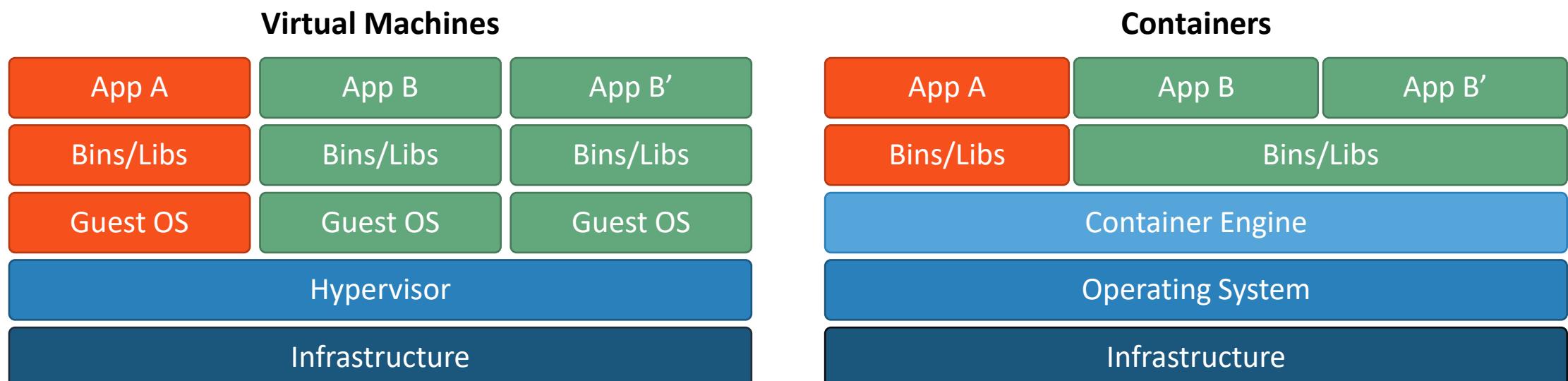
# The «Matrix from Hell», solved

		Environments				
		Dev Laptop	Production Server	Distaster Recovery	Public Cloud	QA Server
Stacks	Web App					
	API Endpoint					
	Analytics					
	App DB					
	Queue					
	Preprocessing					

# Virtual Machines vs Containers

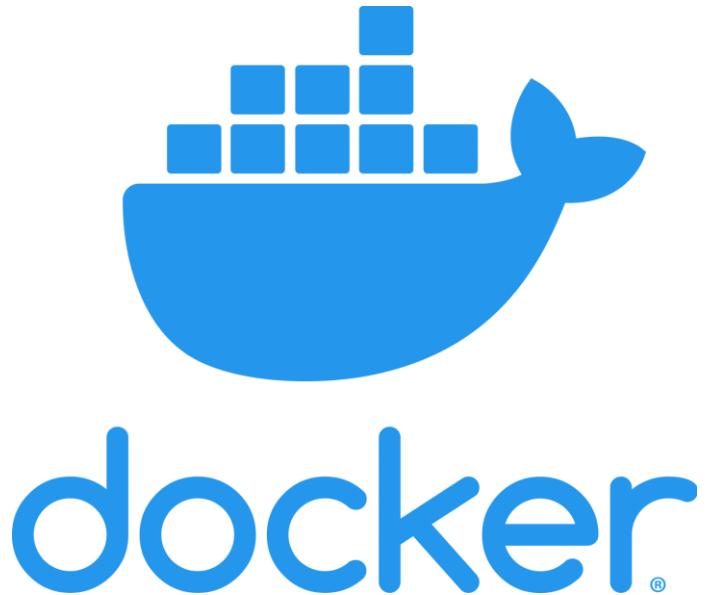
Don't virtual machines solve the same problems as containers?

- Yes, but they're **not as lightweight**
- Containers allow for **significantly faster** deployment, restart, etc...

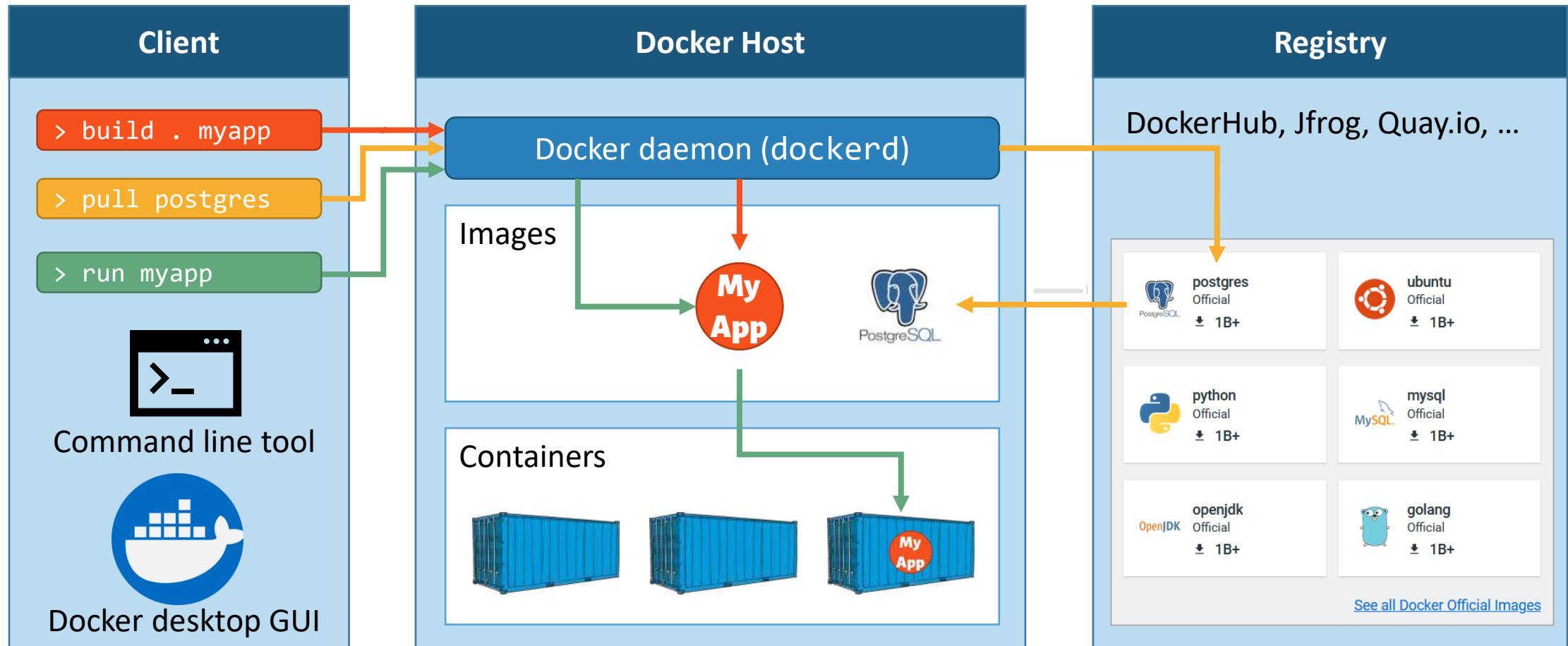


# Docker: The Container Engine

- <https://www.docker.com/>
- Project started in 2013
- Used by more than **13 million devs**
- More than **9 million «dockerized» applications**
- **De facto standard** for containerizing software
- Alternatives exist:
  - [LXD](#), [BuildKit](#), [Buildah](#), [Podman](#), ...

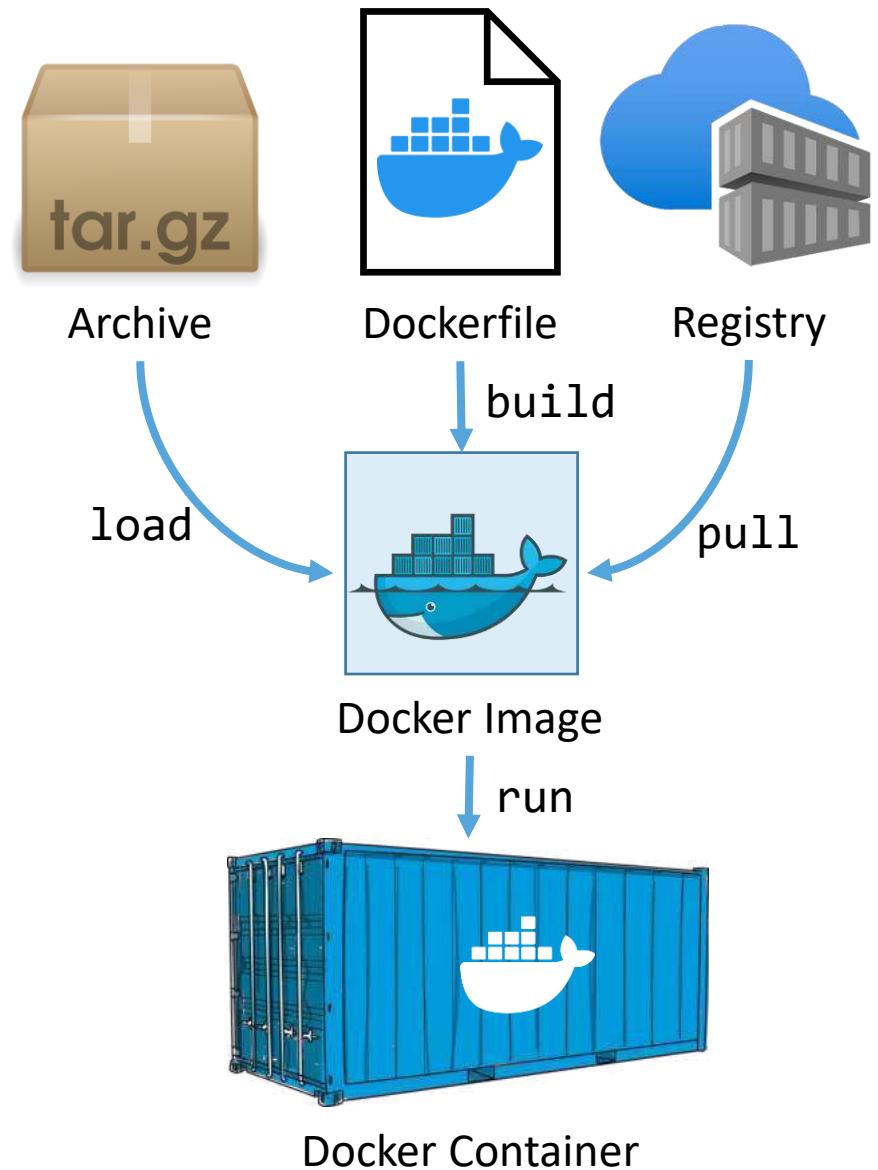


# Docker Architecture



# Docker Images

- Portable, read-only templates
- Contain all the instruction to create a container
- Can be **loaded** from a tar archive file
- Can be **downloaded** from a registry
- Can be built by **extending** an **existing image** with a list of instruction specified in a text file (**Dockerfile**)



# Getting to know Docker

Hands on session with Docker basics

# Running our first container: pulling the image

```
$> docker pull ubuntu:20.04
20.04: Pulling from library/ubuntu
675920708c8b: Pull complete
Digest: sha256:35ab2bf57814e9ff49e365efd5a5935b6915eede5c7f8581e9e1b85e0eecbe16
Status: Downloaded newer image for ubuntu:20.04
docker.io/library/ubuntu:20.04
```

```
$> docker image list
REPOSITORY          TAG           IMAGE ID        CREATED       SIZE
ubuntu              20.04         a0ce5a295b63   3 weeks ago   72.8MB
```

# Running our first container

```
$> docker run -it --name my-first-container ubuntu:20.04
root@f2ce5afe0cba:/# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root
run  sbin  srv  sys  tmp  usr  var
root@f2ce5afe0cba:/# apt update -qq && apt install -y cowsay fortune
root@f2ce5afe0cba:/# /usr/games/fortune | /usr/games/cowsay
-----
/ Never look up when dragons fly \
\ overhead.                         /
-----
 \  ^__^
  \  (oo)\_____
    (__)\       )\/\
        ||----w |
        ||          |
root@f2ce5afe0cba:/# exit
```

# Running our first container: start and attach

```
$> docker container list --all
CONTAINER ID        IMAGE               COMMAND            CREATED           STATUS            NAMES
f2ce5afe0cba        ubuntu:20.04      "bash"           11 mins ago     Exited (0)       my-first-container

$> docker start my-first-container

$> docker container list --all
CONTAINER ID        IMAGE               COMMAND            CREATED           STATUS            NAMES
f2ce5afe0cba        ubuntu:20.04      "bash"           11 mins ago     Up 10 secs      my-first-container

$> docker attach my-first-container
root@f2ce5afe0cba:/# /usr/games/fortune
Never laugh at live dragons.
-- Bilbo Baggins [J.R.R. Tolkien, "The Hobbit"]
```

# Running our first container: transfer files

```
$> echo "Hello" > file.txt  
  
$> docker cp ./file.txt my-first-container:/home/file.txt  
  
$> docker attach my-first-container  
root@b43ea0a68502:/# ls /home/  
file.txt  
root@b43ea0a68502:/# cat /home/file.txt  
"Hello"  
root@b43ea0a68502:/# echo "Hello UniNA!" > /home/file.txt  
root@b43ea0a68502:/# read escape sequence  
  
$> docker cp my-first-container:/home/file.txt ./file.txt  
  
$> type file.txt  
Hello UniNA!
```

# Running our first container: detach and kill

- To detach from the interactive terminal, press the hotkeys **CTRL+P** followed by **CTRL+Q**

```
$> docker attach my-first-container
root@f2ce5afe0cba:/# /usr/games/fortune
Never laugh at live dragons.
        -- Bilbo Baggins [J.R.R. Tolkien, "The Hobbit"]
root@f2ce5afe0cba:/# read escape sequence

$> docker container list --all
CONTAINER ID        IMAGE               COMMAND      CREATED       STATUS        NAMES
f2ce5afe0cba        ubuntu:20.04      "bash"       11 mins ago   Up 59 secs   my-first-container

$> docker kill my-first-container
my-first-container
```

# Running our first container: exec and rm

```
$> docker start my-first-container

$> docker exec -ti my-first-container bash -c /usr/games/fortune
You never hesitate to tackle the most difficult problems.

$> docker container list --all
CONTAINER ID        IMAGE               COMMAND      CREATED       STATUS        NAMES
f2ce5afe0cba        ubuntu:20.04      "bash"       11 mins ago   Up  59 secs   my-first-container

$> docker kill my-first-container

$> docker rm my-first-container

$> docker container list --all
CONTAINER ID        IMAGE               COMMAND      CREATED       STATUS        NAMES
```

# Building our own first Image: Dockerfile

- A Dockerfile is a set of commands to assemble an Image
- Start **FROM** a base image
- **RUN** commands, **COPY** files, **EXPOSE** ports, set **ENVIRONMENT** vars, ...
- [Dockerfile reference](#)

```
# Start from the ubuntu:20.04 base image
FROM ubuntu:20.04
# Update the list of packages and install fortune and cowsay
RUN apt update -qq && apt install -y -q fortune cowsay
# Copy file.txt from the Dockerfile dir. to /home/file.txt in the Container
COPY ./file.txt /home/file.txt
# Default entrypoint for executing containers
CMD bash
```

# Building our own first Image

```
$> cd ubuntu-fortune-cowsay  
  
$> dir /b  
Dockerfile  
file.txt  
  
$> docker build -t "ubuntu-fortune-cowsay" .  
  
[+] Building 25.8s (8/8) FINISHED  
=> [internal] load build definition from Dockerfile          0.0s  
=> [internal] load metadata for docker.io/library/ubuntu:20.04 0.0s  
=> CACHED [1/3] FROM docker.io/library/ubuntu:20.04          0.0s  
=> [2/3] RUN apt update -qq && apt install -y -q fortune cowsay 24.9s  
=> [3/3] COPY ./file.txt /home/file.txt                      0.1s  
=> => writing image ha256:6e05a97a366b87c98a2[...]26678046c 0.0s  
=> => naming to docker.io/library/ubuntu-fortune-cowsay        0.0s
```

# Building our own first Image

```
$> docker image list --all
REPOSITORY          TAG      IMAGE ID      CREATED        SIZE
ubuntu-fortune-cowsay  latest   6e05a97a366b  23 seconds ago  159MB
ubuntu              20.04    a0ce5a295b63  3 weeks ago   72.8MB
$> docker run -it --name my-ubuntu-container ubuntu-fortune-cowsay
root@a87b47206c9a:/# /usr/games/fortune | usr/games/cowsay
_____
< You look tired. >
-----
      \  ^__^
       \  (oo)\_____
         (__)\       )\/\
             ||----w |
             ||     |

root@a87b47206c9a:/# cat /home/file.txt
Hello UniNA!
```

# Communication over the web for distributed systems with REST APIs

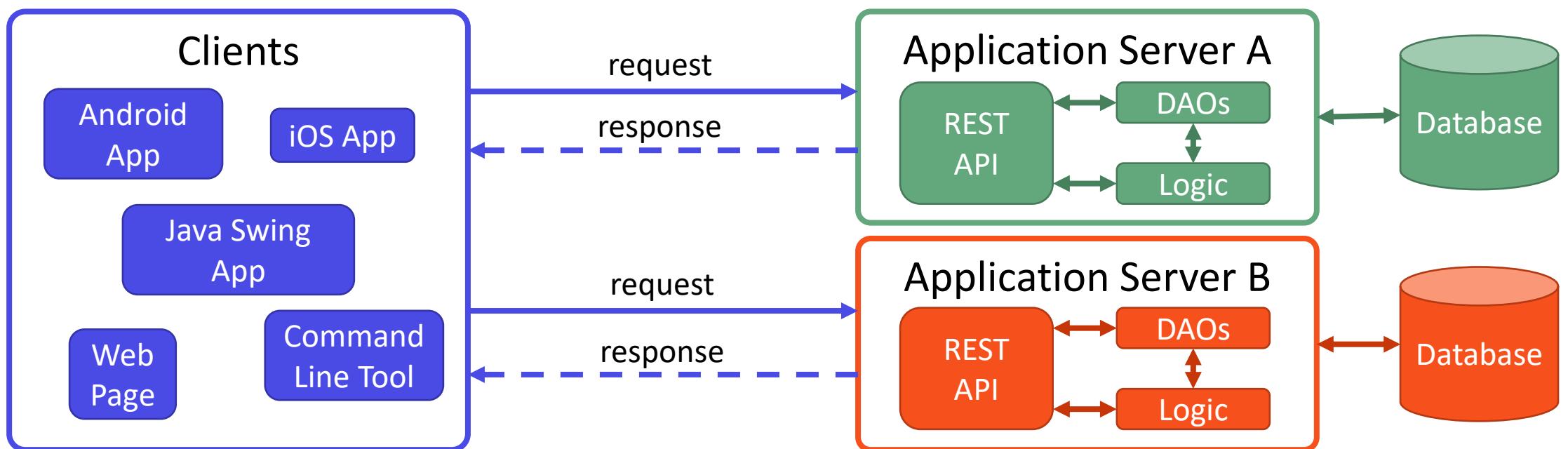
**Luigi Libero Lucio Starace**

<https://luistar.github.io>

[luigiliberolucio.starace@unina.it](mailto:luigiliberolucio.starace@unina.it)

# Architectural Background

- Clients and servers on different machines, communicate over WWW



# Communication mechanisms

How can we handle communication between clients and applications?

- Manually define a protocol over TCP, open sockets, etc...
  - Not very cost-effective, not very interoperable
- CORBA, Java RMI, SOAP, ...
  - Dedicated protocols exist(ed), re-inventing an alternative to the web
- Just use web standards!

# REST

- REST is a set of principles and guidelines that define how web standards should be used
- Based on HTTP and URIs
- Provides a common and consistent interface based on «proper» use of HTTP

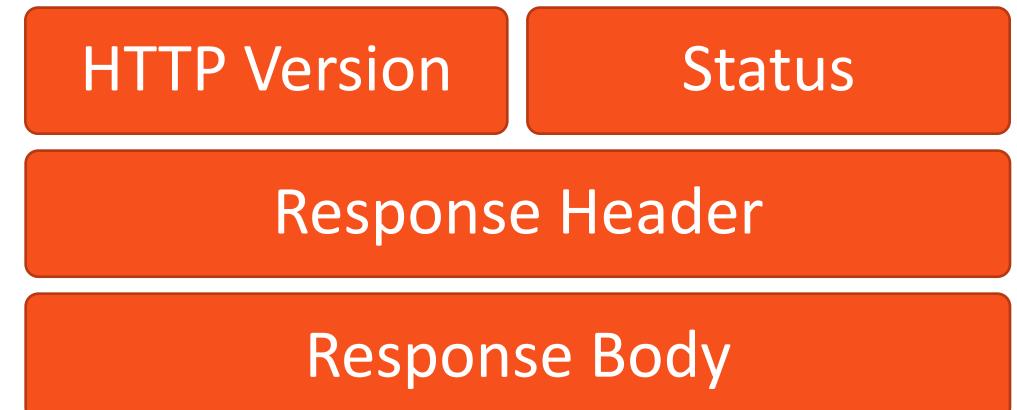
# HTTP

- HyperText Transfer Protocol
- Two types of messages: Request and Response

## Request



## Response



# HTTP Requests

- HTTP can request different kinds of operations (using **Verbs**)
- The resource on which to operate is identified by the URI
- Based on idea of CRUD (Create, Retrieve, Update, Delete)

Verb	Operation Description
PUT	Here is some new data. Save it and <b>CREATE</b> a new resource
GET	<b>RETRIEVE</b> information about the resource
POST	Here is <b>UPDATED</b> info about the resource
DELETE	<b>DELETE</b> this resource

# HTTP Responses

- Status codes give information about the outcome of the request

Status Code	Meaning
<b>200</b>	Request was successfully handled
<b>201</b>	A resource was successfully created
<b>400</b>	Cannot understand the request
<b>401</b>	Authentication failed, or not authorized.
<b>404</b>	Resource not found
<b>405</b>	Method not supported by resource
<b>500</b>	Application error

# HTTP: Data interexchange formats

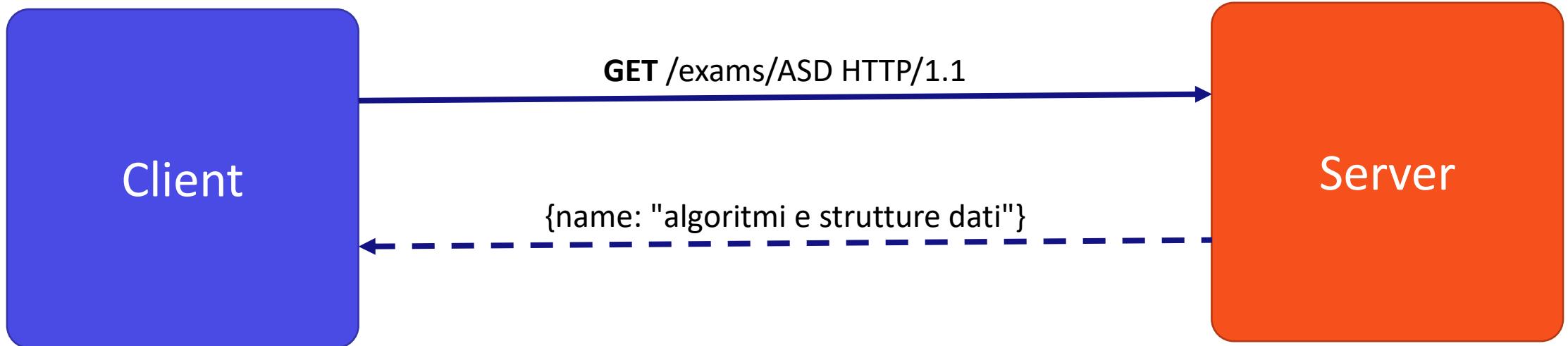
- Widely used formats include JSON and XML

```
{  
  "exams": [  
    {"name": "ASD",  
     "grade": 30  
    }, {  
      "name": "INGSW",  
      "grade": 30  
    }  
  ]  
}
```

```
<?xml version="1.0" encoding="UTF-8" ?>  
<root>  
  <exams>  
    <exam>  
      <name>ASD</name>  
      <grade>30</grade>  
    </exam>  
    <exam>  
      <name>INGSW</name>  
      <grade>30</grade>  
    </exam>  
  </exams>  
</root>
```

# Representational State Transfer (REST)

- **Application State** (on the Client)
- **Resource State** (on the Server)
- Transferred using appropriate representations (e.g.: JSON, XML, ...)



# REST Fundamentals

- A REST API allows to interact with **resources**
- All requests are associated to a unique URI
- «A resource is anything that's important enough to be referenced as a thing in itself.»<sup>1</sup>
- Resource typically (but not necessarily) correspond to persistent domain objects
- HTTP verbs should be used to retrieve or manipulate resources

[1] Richardson, Leonard, and Sam Ruby. *RESTful web services*. "O'Reilly Media, Inc.", 2008.

# REST: Examples

We want to write an app that manages a list of Exams a student took

HTTP verb/URI	Meaning
<b>GET /exams</b>	Retrieve a list of all saved exams
<b>GET /exams/&lt;ID&gt;</b>	Retrieve only the exam whose ID is <ID>
<b>POST /exam</b>	Save a new Exam. Data of the exam to save are in the request body
<b>PUT /exam/&lt;ID&gt;</b>	Replace (or create) the exam whose ID is <ID> using the data in the request body
<b>DELETE /exam/&lt;ID&gt;</b>	Delete the exam having ID <ID>

# A RESTful service with Spring Boot

- Practical Demo!

# API Authentication/Authorization

- REST APIs allow users to manipulate resources
- In most cases, we don't want that everyone is able to do so
- **Authentication:** We want that only legit users can access the resources
- **Authorization:** We may also want that some users can access only certain resources (e.g.: an employee shouldn't be able to update its own salary)

# How to secure REST APIs

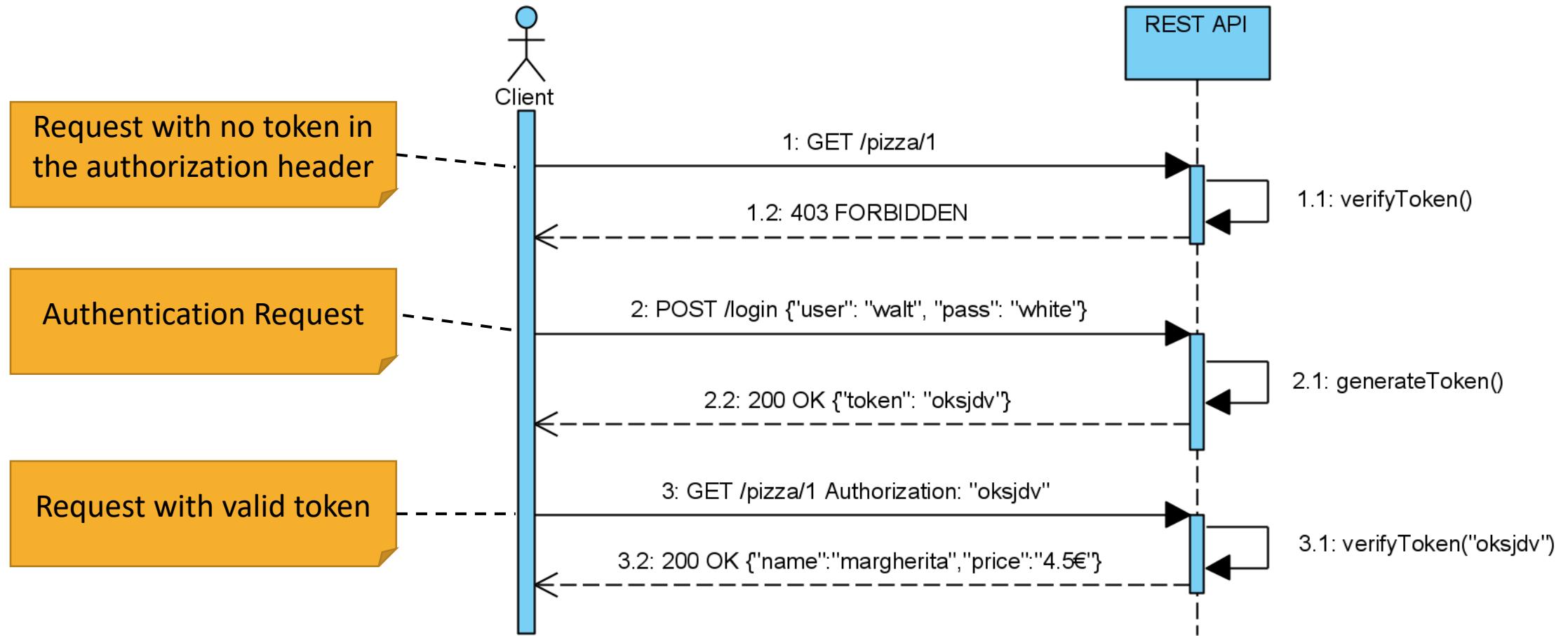
A widely-used authentication scheme is based on Tokens



Idea:

1. Clients send a request with username and password to the API
2. API validates username and password, and generates a token
3. The token (a string) is returned to the client
4. Client must pass the token back to the API at every subsequent request that requires authentication (in the Authorization Header)
5. API verifies the token before responding

# Token-based Authentication Scheme



# JSON Web Token (JWT)

- JWT is a widely-adopted open standard ([RFC 7519](#))
- Allows to securely share claims between two parties
- A JWT token is a string consisting of three parts, separated by “.”
- Structure: **Header.Payload.Signature**

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpX  
VCJ9 . eyJuYW1lIjoibHVpZ2kiLCJyb2x  
lIjoiaWRtaW4iLCJleHAiOjE2NzAzOTg  
0MzJ9 . fopBYrax8wcB7rnPjCc0Mc62IT  
21JdvyOdyixMwMZAQ

# JSON Web Token

eyJhbGciOiJIUzI  
1NiIsInR5cCI6Ik  
pXVCJ9.eyJuYW1l  
IjoibHVpZ2kiLCJ  
yb2x1IjoiYWRTaw  
4iLCJleHAiOjE2N  
zAzOTg0MzJ9.fop  
BYrax8wcB7rnPjC  
c0Mc62IT21Jdvy0  
dyixMwMZAQ

Base64Url Encoding

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```

Base64Url Encoding

```
{  
  "name": "luigi",  
  "role": "admin",  
  "exp": 1670398432  
}
```

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  secret_key  
)
```

# Deploying a Web Service in the Cloud

Luigi Libero Lucio Starace

[luigiliberolucio.starace@unina.it](mailto:luigiliberolucio.starace@unina.it)

Università degli Studi di Napoli Federico II, Naples, Italy

November 19, 2021

# Goals of this demonstration

- Download and run a simple Node.js web service locally
  - Requirements: Node.js (recommended v14.16.1 or above)
- Deploy the web service in the Cloud using AWS (on an AWS EC2 instance)

# Download the web service

Clone the repo: <https://github.com/luistar/cloud-ws-demo.git>

```
>> git clone https://github.com/luistar/cloud-ws-demo.git
Cloning into 'cloud-ws-demo'...
remote: Enumerating objects: 11, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 11 (delta 2), reused 7 (delta 1), pack-reused 0
Unpacking objects: 100% (11/11), done.
```

Then move to the newly-created directory

```
>> cd cloud-ws-demo
```

# Check out the app.js file (if you want)

Two get endpoints: /greet and /fortune

```
const express = require('express')
const app = express()
const port = 3000

app.get('/greet', (req, res) => {
  res.send('Hello World!')
})

app.get('/fortune', (req, res) => {
  // [...] computes a random fortune message (omitted for brevity)
  res.send(fortune)
})
```

# Install dependencies and run the web service

## Install dependencies

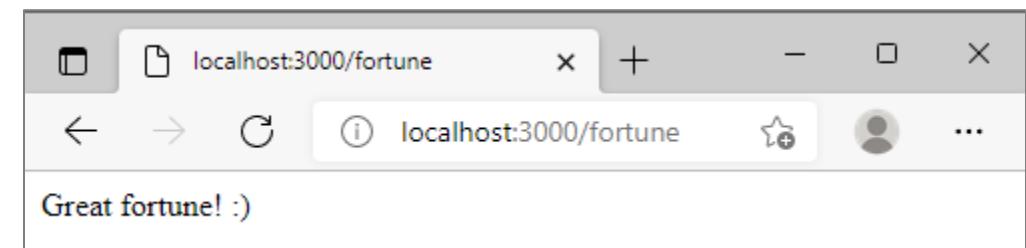
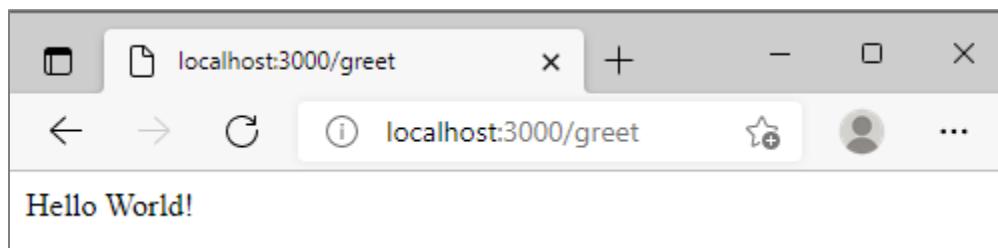
```
>> npm install
```

## Run the web service

```
>> node app.js
```

```
Web service listening at http://localhost:3000
```

## And try it with your web browser!



# Let's move to the Cloud

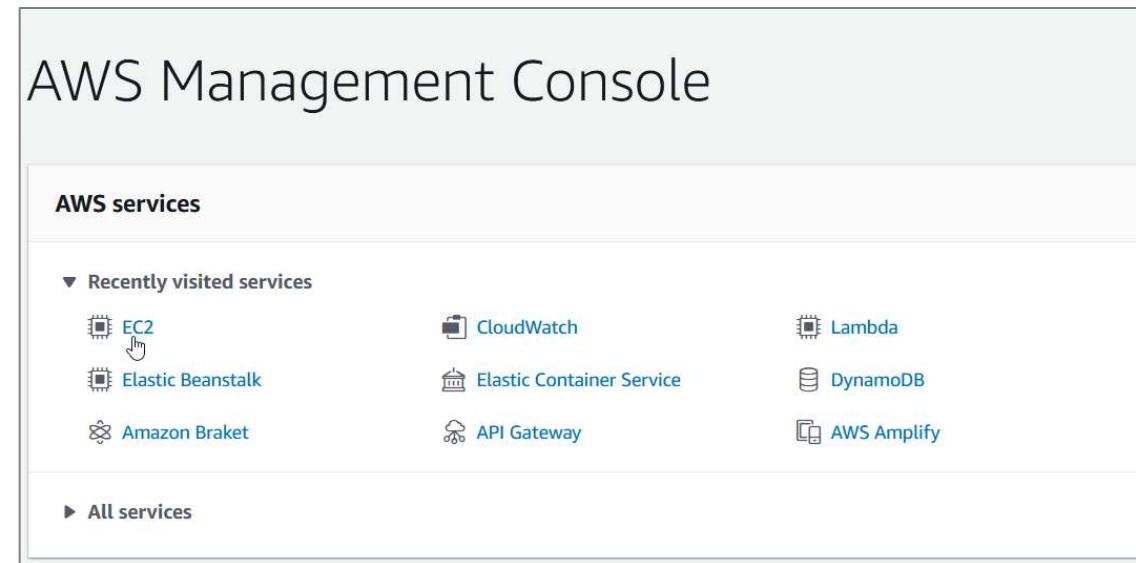
We'll use AWS for this tutorial

# Requirements

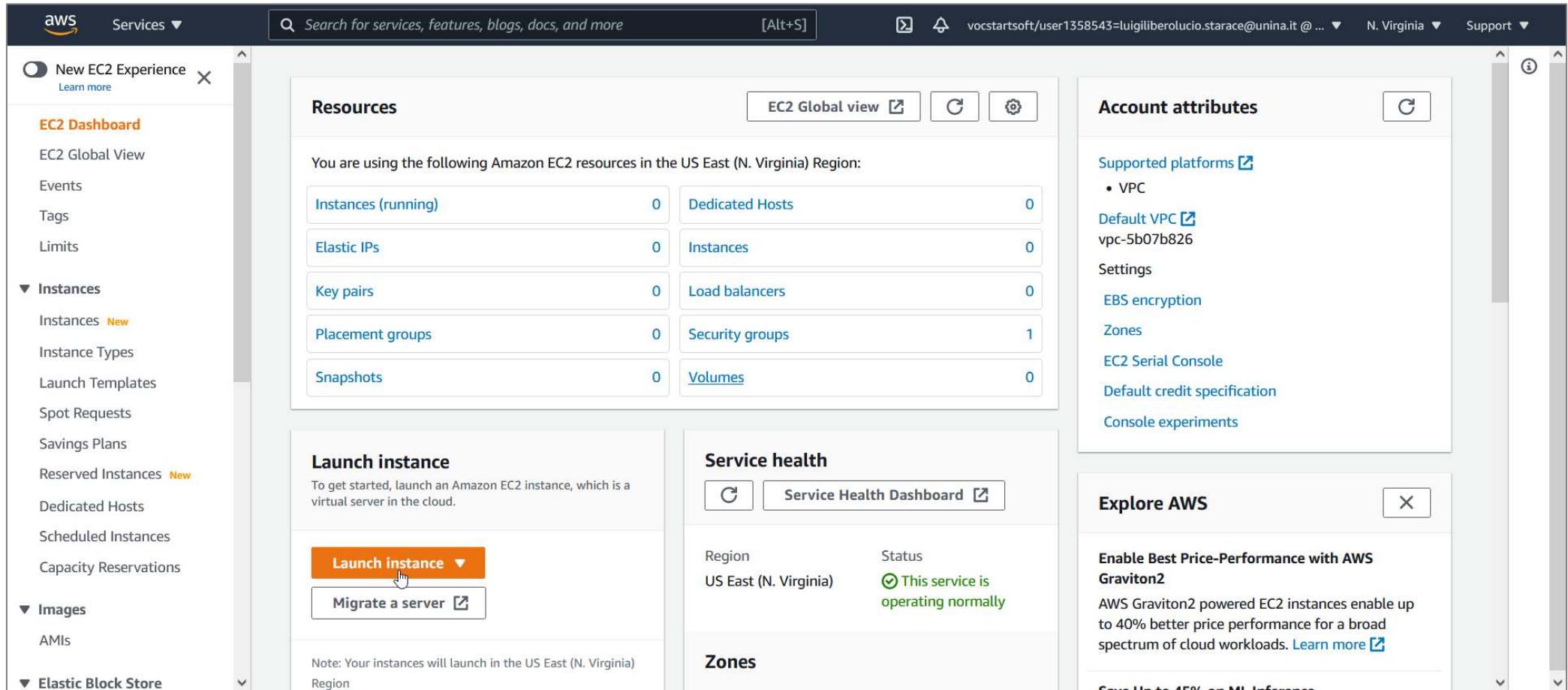
- To replicate this demo you'll need an AWS account. If you're eligible for the free trial, it'll cost you nothing!
- If you prefer a different provider (such as Azure) the procedure is basically the same (so a little of google-fu will surely suffice!)

# EC2 deployment

- We'll start from the most basic deployment: running the web service on a virtual server in the Cloud.
- Let's start by selecting the EC2 service from the AWS web console



# Click on the «Launch instance» button



# Select the Ubuntu Server 20.04 LTS Machine Image

The screenshot shows the AWS CloudFormation console with the following details:

- Step 1: Choose an Amazon Machine Image (AMI)**
- Description:** Volume Type. Amazon EC2 AMI Tools preinstalled; Apache 2.2, MySQL 5.5, PHP 5.3, and Ruby 1.8.7 available.
- Root device type:** ebs    **Virtualization type:** hvm    **ENA Enabled:** Yes
- AMI Options:**
  - Ubuntu Server 20.04 LTS (HVM), SSD Volume Type -** ami-083654bd07b5da81d (64-bit x86) / ami-04fe9398b2a27a600 (64-bit Arm)
    - Select** button
    - 64-bit (x86)
    - 64-bit (Arm)
  - Ubuntu Server 18.04 LTS (HVM), SSD Volume Type -** ami-0279c3b3186e54acd (64-bit x86) / ami-0528007a60177dd84 (64-bit Arm)
    - Select** button

# Select the t2.micro instance type

The screenshot shows the AWS EC2 instance creation wizard at Step 2: Choose an Instance Type. The 't2.micro' instance type is selected, highlighted with a blue border and a green 'Free tier eligible' badge. The table lists various t2 instance types based on memory size: nano (0.5 GiB), micro (1 GiB, currently selected), small (2 GiB), medium (4 GiB), large (8 GiB), xlarge (16 GiB), and 2xlarge (32 GiB). All instances use EBS storage and have low to moderate network performance and IPv6 support.

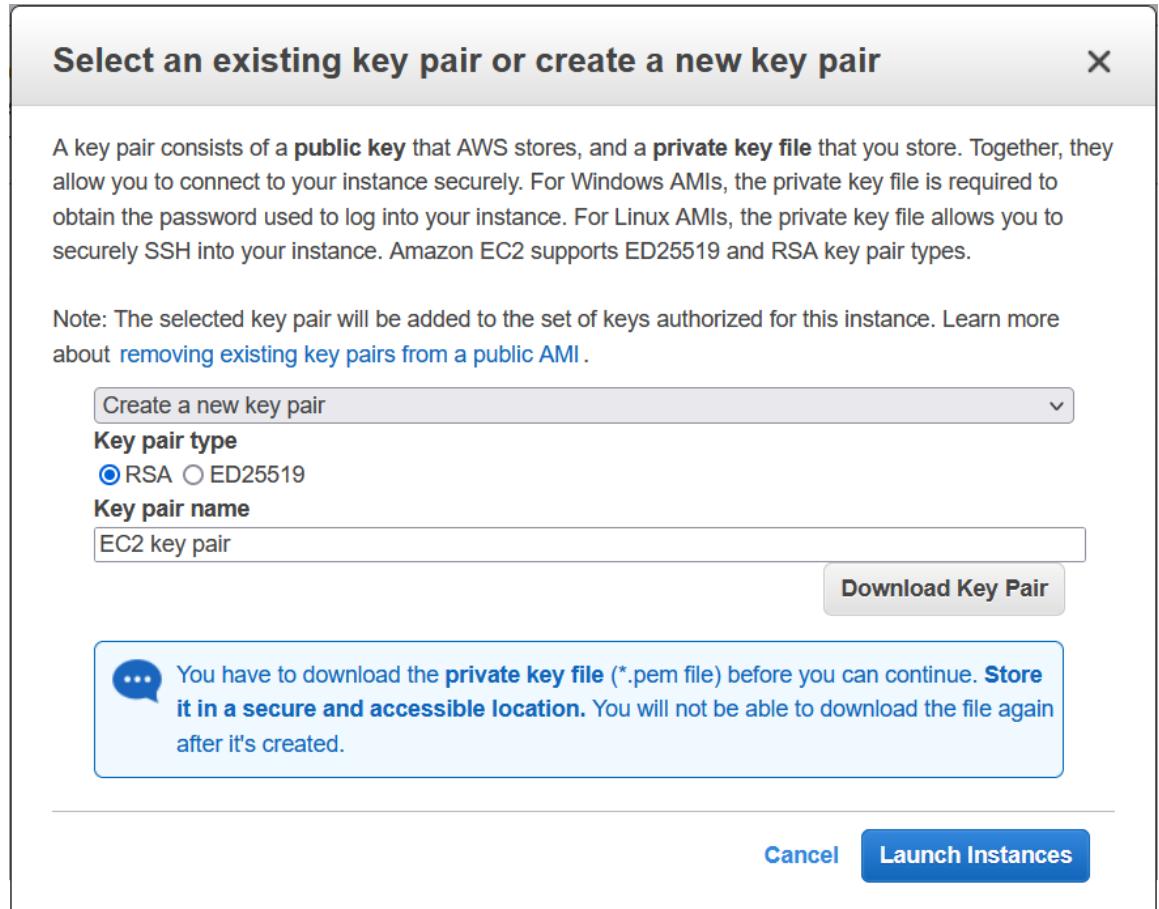
	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
<input type="checkbox"/>	t2	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
<input checked="" type="checkbox"/>	t2	t2.micro <small>Free tier eligible</small>	1	1	EBS only	-	Low to Moderate	Yes
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

Currently selected: t2.micro (- ECUs, 1 vCPUs, 2.5 GHz, -, 1 GiB memory, EBS only)

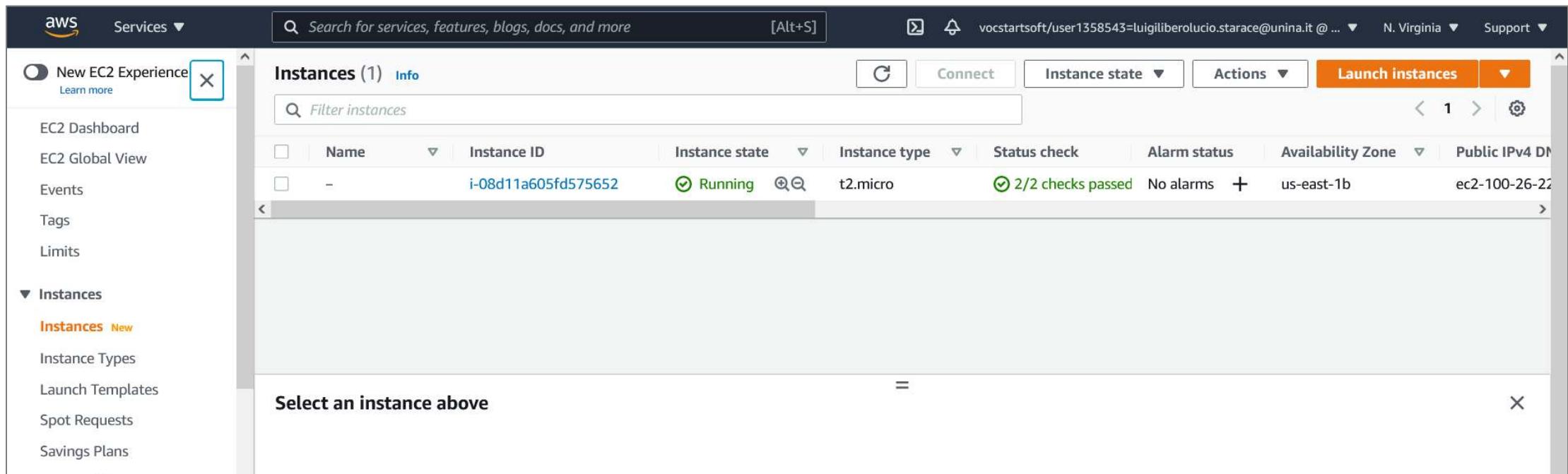
Cancel Previous Review and Launch Next: Configure Instance Details

# Generate a new key pair to connect to the instance

- Make sure to download the .pem file



# Wait for the instance to start



The screenshot shows the AWS EC2 Instances page. The top navigation bar includes the AWS logo, a search bar, and account information. On the left, a sidebar lists various EC2-related options like EC2 Dashboard, Global View, Events, Tags, Limits, and a expanded 'Instances' section with sub-options: Instances (New), Instance Types, Launch Templates, Spot Requests, and Savings Plans. The main content area displays a table titled 'Instances (1)'. The table has columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IPv4 DNS. A single row is shown for an instance with the following details: Name is empty, Instance ID is i-08d11a605fd575652, Instance state is Running (indicated by a green checkmark), Instance type is t2.micro, Status check is 2/2 checks passed (green checkmark), Alarm status is No alarms, Availability Zone is us-east-1b, and Public IPv4 DNS is ec2-100-26-22. Below the table, a message says 'Select an instance above'.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
-	i-08d11a605fd575652	Running	t2.micro	2/2 checks passed	No alarms	us-east-1b	ec2-100-26-22

# Click on the instance id to know more

## Note the public IPv4 DNS name!

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with options like New EC2 Experience, EC2 Dashboard, EC2 Global View, Events, Tags, Limits, Instances (selected), Instances (New), Instance Types, and Launch Templates. The main area displays the Instance summary for instance **i-08d11a605fd575652**, which was updated less than a minute ago. The instance is currently **Running**. The summary table includes the following details:

Value	Description
i-08d11a605fd575652	Instance ID
-	IPv6 address
ip-172-31-16-240.ec2.internal	Private IPv4 DNS
100.26.22.241   <a href="#">open address</a>	Public IPv4 address
Running	Instance state
t2.micro	Instance type
172.31.16.240	Private IPv4 addresses
ec2-100-26-22-241.compute-1.amazonaws.com   <a href="#">open address</a>	Public IPv4 DNS
-	Elastic IP addresses

# SSH into your new (virtual) server

## SSH into your instance

```
>> ssh -i /path/to/key.pem ubuntu@<the-ipv4-dns-name-we-noted>
```

If everything worked, you should be in!

```
ubuntu@ip-172-31-16-240:~$
```

**Notice:** you might be getting an «**unprotected private key file**» error. In that case, make sure to correctly assign permissions to the pem file, so that only you can see it. Use chmod if on a unix derivative, or see [this](#) if you're on Windows.

# Install Node.js on your virtual server

Update and then install nodejs and npm

```
ubuntu@ip-172-31-16-240:~$ sudo apt update  
ubuntu@ip-172-31-16-240:~$ sudo apt install nodejs npm
```

# Then clone the web service

Clone and install dependencies

```
ubuntu@ip-172-31-16-240:~$ git clone https://github.com/luistar/cloud-ws-demo.git  
ubuntu@ip-172-31-16-240:~$ cd cloud-ws-demo  
ubuntu@ip-172-31-16-240:~/cloud-ws-demo$ npm install
```

And then you're ready to start our Web Service

```
ubuntu@ip-172-31-16-240:~/cloud-ws-demo$ node app.js  
Web service listening at http://localhost:3000
```

# We're almost there!

Our web service is up and running in our virtual server, but we still cannot reach it from the internet!



# Allow incoming traffic

- Select the security group in the security tab in your instance summary page

The screenshot shows the AWS Management Console with the EC2 service selected. In the top left, there's a modal titled "New EC2 Experience" with a close button. The main content area shows the "Instances" section of the EC2 dashboard. On the right, the "Instance summary for i-08d11a605fd575652" is displayed, with the "Security" tab selected. The summary includes fields for Instance ID (i-08d11a605fd575652), IPv6 address (empty), Private IPv4 DNS (ip-172-31-16-240.ec2.internal), VPC ID (vpc-5b07b826), and Subnet ID (subnet-820b39cf). Below this, the "Security details" section shows the IAM Role (empty) and the Security groups, which is where a red arrow points to the "sg-081860d1accea4b99 (launch-wizard-1)" entry.

EC2 > Instances > i-08d11a605fd575652

Instance summary for i-08d11a605fd575652 Info

Updated less than a minute ago

Instance ID i-08d11a605fd575652

IPv6 address -

Private IPv4 DNS ip-172-31-16-240.ec2.internal

VPC ID vpc-5b07b826

Subnet ID subnet-820b39cf

Details Security Networking Storage Status checks

▼ Security details

IAM Role -

Security groups sg-081860d1accea4b99 (launch-wizard-1)

aws Services ▾

New EC2 Experience X

Learn more

EC2 Dashboard

EC2 Global View

Events

Tags

Limits

▼ Instances

Instances New

Instance Types

Launch Templates

Spot Requests

Savings Plans

Reserved Instances New

Dedicated Hosts

Scheduled Instances

Capacity Reservations

▼ Images

AMIs

▼ Elastic Block Store

Volumes New

Snapshots

Lifecycle Manager New

▼ Network & Security

Security Groups

# Add a new inbound traffic rule

Type: Custom TCP

Port range: 3000

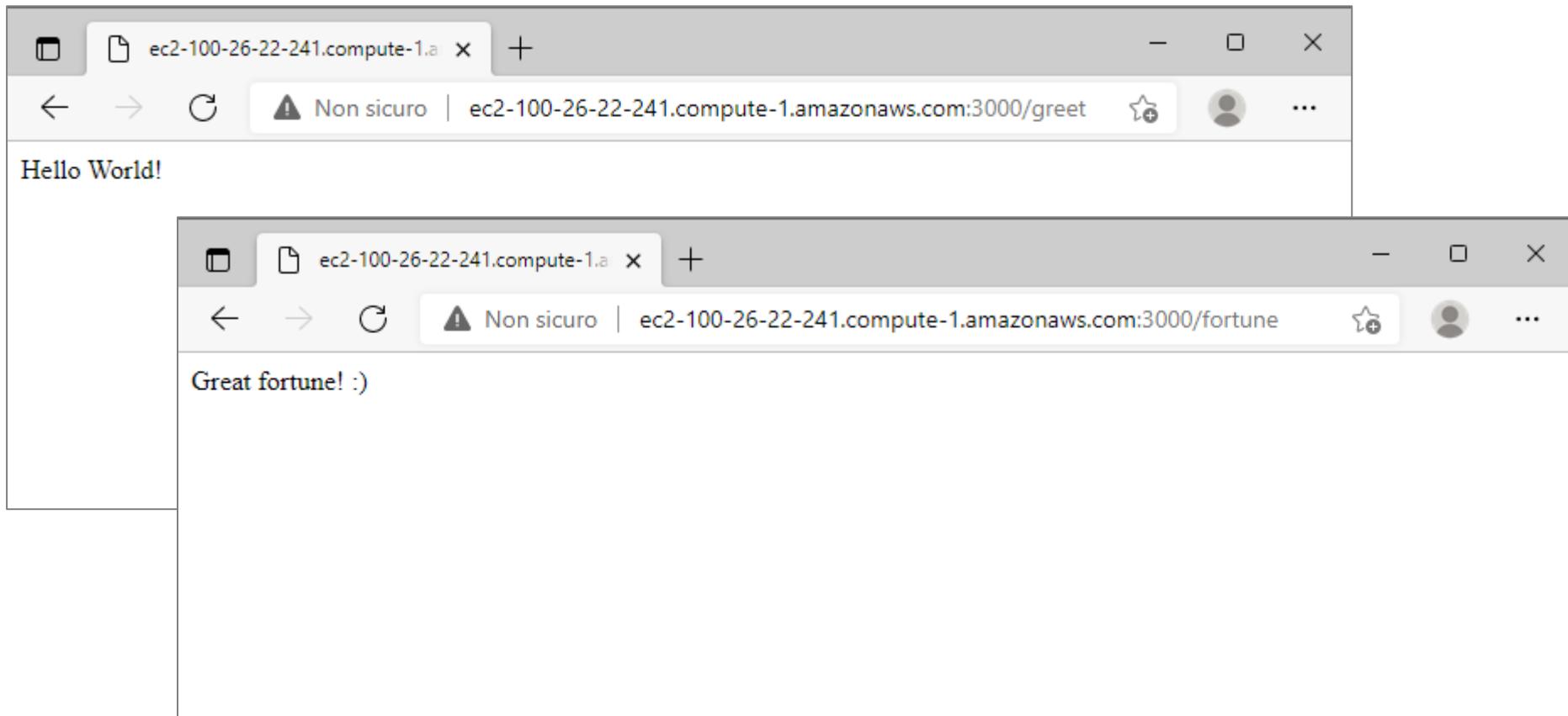
Source: Anywhere IPv4

The screenshot shows the 'Edit inbound rules' section of the AWS EC2 Security Groups interface. It displays a single inbound rule named 'Inbound rule 1'. The rule details are as follows:

- Security group rule ID:** sgr-0c5783850cdb9e099
- Type:** Custom TCP
- Protocol:** TCP
- Port range:** 3000
- Source type:** Anywhere-IPv4
- Source:** 0.0.0.0/0
- Description - optional:** Allow users to reach our web service

At the bottom left, there is a 'Delete' button and a 'Add rule' button.

# Guess what? Now it works!



# Creating an image from our instance

Updating the repositories, installing Node.js and npm, downloading and installing our web service was **fun**, sure.

Can we skip all that next time? Yes, we can create an image!

The screenshot shows the AWS EC2 Instances page. A green success banner at the top states: "Successfully created ami-0111ce16133f6fa5f from instance i-08d11a605fd575652." The main table displays one instance: "i-08d11a605fd575652" (Name), "Running" (Instance state), "t2.micro" (Instance type), "2/2 checks passed" (Status check), "No alarms" (Alarm status), "us-east-1b" (Availability Zone), and "ec2-100-26-22-241" (Public IPv4 DNS). On the right side of the instance row, there is a "Actions" dropdown menu with options: "Create image" (highlighted in blue), "Create template from instance", and "Launch more like this". The left sidebar shows the "Instances" section with sub-links: "Instances", "Instance Types", "Launch Templates", "Spot Requests", "Savings Plans", and "Reserved Instances". The top navigation bar includes the AWS logo, a search bar, and account information.

# Terminate our instance

- Remember that we're paying as long as that instance is up! Terminate it when it's not needed anymore!

The screenshot shows the AWS EC2 Instances page. A success message at the top states: "Successfully created ami-0111ce16133f6fa5f from instance i-08d11a605fd575652." The main table displays one instance:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
-	i-08d11a605fd575652	Running	t2.micro	2/2 checks passed	No alarms	us-east-1b

The "Actions" column for this instance contains a dropdown menu with the following options: Stop instance, Start instance, Reboot instance, Hibernate instance, and Terminate instance. The "Terminate instance" option is highlighted with a red box.

# Start the instance from the image we saved

This time we can select our image!

The screenshot shows the AWS Step Functions interface for creating a new state machine. The top navigation bar includes the AWS logo, 'Services ▾', a search bar ('Search for services, [Alt+S]'), and user information ('vocstartsoft/user1358543=luigiliberolucio.stara'). Below the navigation is a horizontal progress bar with seven steps: 1. Choose AMI (highlighted in yellow), 2. Choose Instance Type, 3. Configure Instance, 4. Add Storage, 5. Add Tags, 6. Configure Security Group, and 7. Review.

**Step 1: Choose an Amazon Machine Image (AMI)**

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Search for an AMI by entering a search term e.g. "Windows"

Search by Systems Manager parameter

Quick Start

My AMIs

AWS Marketplace

Community AMIs

Ownership

Owned by me

Shared with me

Architecture

32-bit (x86)

**Web Service Demo - ami-0111ce16133f6fa5**

Ubuntu instance with nodejs, ready to run our web service

Root device type: ebs Virtualization type: hvm Owner: 040720104196 ENA

Enabled: Yes

Select

64-bit (x86)

# Running the image we saved

- This time, when starting the EC2 instance, make sure to use the same security group we already set up! (otherwise, you will need to add the inbound rule to port 3000 again!)

# Suggested next steps

- i. Check out **AWS Elastic IP Address** if you need a static IP that does not change when you stop and restart an instance
- ii. Try adding an **auto-scaling group!**
- iii. After doing (i) and (ii), try deploying our web service using **AWS ElasticBeanstalk!**

# GETTING TO KNOW AMAZON WEB SERVICES

Introduction and core services

---

Luigi Libero Lucio Starace

[luigiliberolucio.starace@unina.it](mailto:luigiliberolucio.starace@unina.it)

March 23, 2020

University of Naples, Federico II

## 1 A little bit of context

# OUTLINE

---

1 A little bit of context

2 AWS Infrastructure

# OUTLINE

---

1 A little bit of context

2 AWS Infrastructure

3 Core AWS services

# OUTLINE

- 1 A little bit of context
- 2 AWS Infrastructure
- 3 Core AWS services
- 4 Take Home Messages

## A LITTLE BIT OF CONTEXT

---

Cloud computing is the on-demand delivery of computing resources through a cloud services platform via the internet with pay-as-you-go pricing.

Cloud computing is the **on-demand delivery** of computing resources through a cloud services platform via the internet with pay-as-you-go pricing.

Cloud computing is the **on-demand delivery** of computing resources through a cloud services platform via the internet with **pay-as-you-go** pricing.

- Software as a Service (SaaS)

- **Software as a Service (SaaS)**

The service vendor provides the user with a completed product that is run and managed by the service provider.

- **Software as a Service (SaaS)**

The service vendor provides the user with a completed product that is run and managed by the service provider.

- **Platform as a Service (PaaS)**

- **Software as a Service (SaaS)**

The service vendor provides the user with a completed product that is run and managed by the service provider.

- **Platform as a Service (PaaS)**

The service vendor provides the user with a set of API which can be used to build, test and deploy applications.

- **Software as a Service (SaaS)**

The service vendor provides the user with a completed product that is run and managed by the service provider.

- **Platform as a Service (PaaS)**

The service vendor provides the user with a set of API which can be used to build, test and deploy applications.

- **Infrastructure as a Service (IaaS)**

- **Software as a Service (SaaS)**

The service vendor provides the user with a completed product that is run and managed by the service provider.

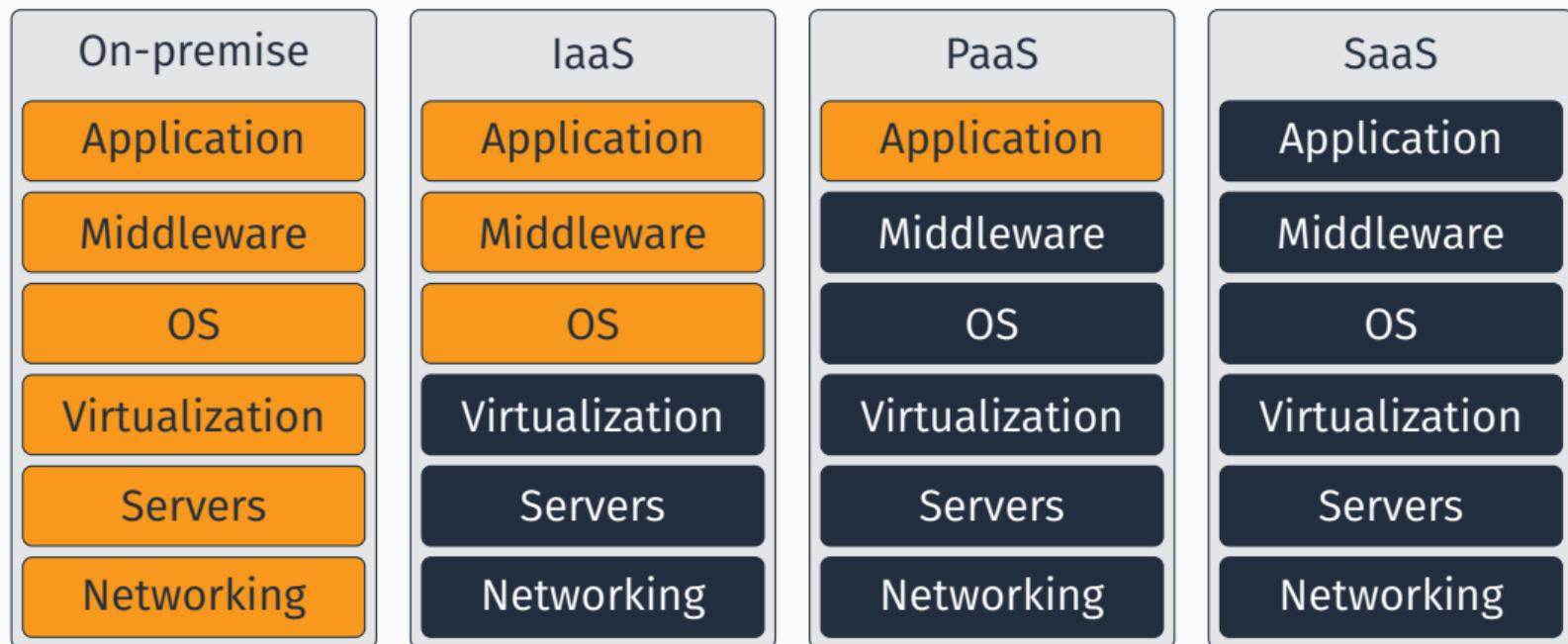
- **Platform as a Service (PaaS)**

The service vendor provides the user with a set of API which can be used to build, test and deploy applications.

- **Infrastructure as a Service (IaaS)**

The service vendor provides users access to computing resources such as servers, storage and networking.

# SERVICE MODELS: A VISUAL COMPARISON



User manages



Someone else manages

- Google



# Google Cloud

## THE BIGWIGS

- Google
- IBM



IBM Cloud

## THE BIGWIGS

---

- Google
- IBM
- Microsoft



## THE BIGWIGS

---

- Google
- IBM
- Microsoft
- Alibaba



## THE BIGWIGS

- Google
- IBM
- Microsoft
- Alibaba
- Amazon



According to [Fle19]:

- 91% of the surveyed companies uses public cloud services

According to [Fle19]:

- 91% of the surveyed companies uses public cloud services
- 84% of these enterprises have a multi-cloud strategy

According to [Fle19]:

- 91% of the surveyed companies uses public cloud services
- 84% of these enterprises have a multi-cloud strategy
  - they buy cloud services from different providers;

According to [Fle19]:

- 91% of the surveyed companies uses public cloud services
- 84% of these enterprises have a multi-cloud strategy
  - they buy cloud services from different providers;
  - they combine public and private clouds (hybrid cloud approach).

## PUBLIC CLOUD ADOPTION

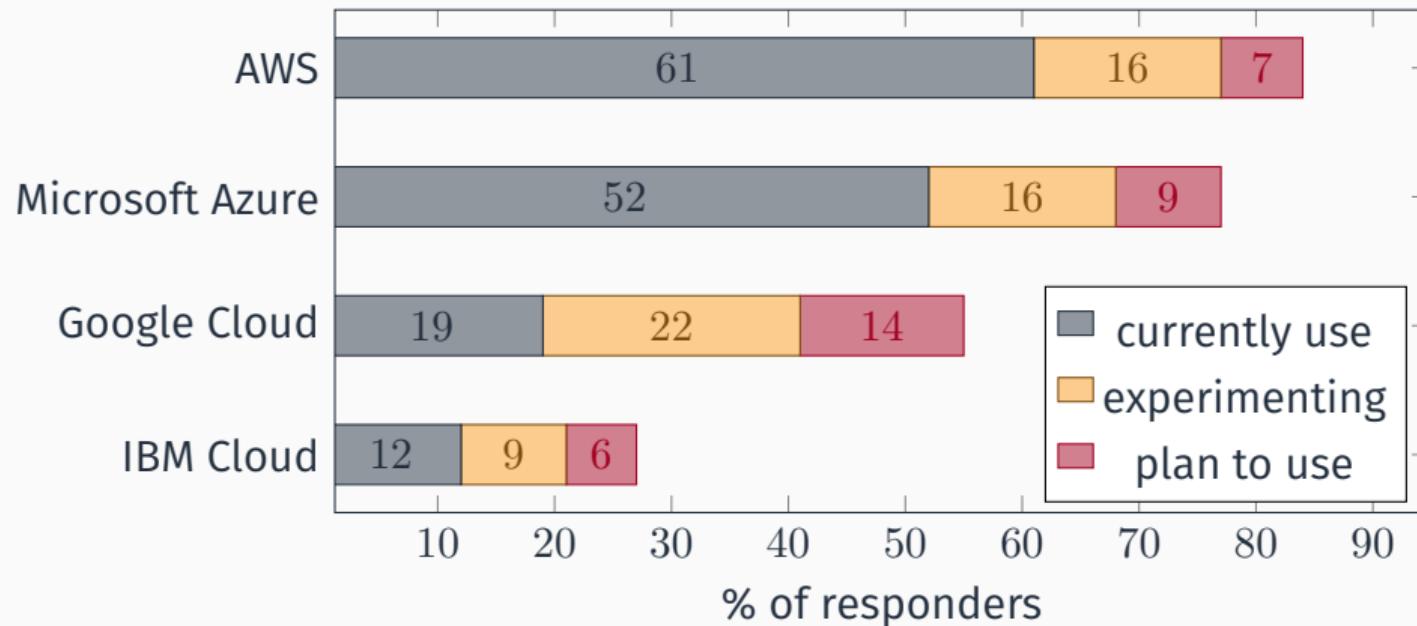


Figure 2: Public cloud adoption [Fle19]

## AWS INFRASTRUCTURE

---

- AWS is present in 22 geographical regions.



# AWS INFRASTRUCTURE

- AWS is present in 22 geographical regions.
- Each region consists of multiple (3–6) availability zones.



- AWS is present in 22 geographical regions.
- Each region consists of multiple (3–6) availability zones.
- An availability zone can be multiple data centers, with up to hundreds of thousands of servers.



- AWS is present in 22 geographical regions.
- Each region consists of multiple (3–6) availability zones.
- An availability zone can be multiple data centers, with up to hundreds of thousands of servers.
- 216 points of presence for effective caching and content delivery.



## CORE AWS SERVICES

---



Amazon Web Services is a collection of cloud-based services.



Amazon Web Services is a collection of cloud-based services. **A very big one.**



Amazon Web Services is a collection of cloud-based services. **A VERY big one.**

# Periodic Table of Amazon Web Services

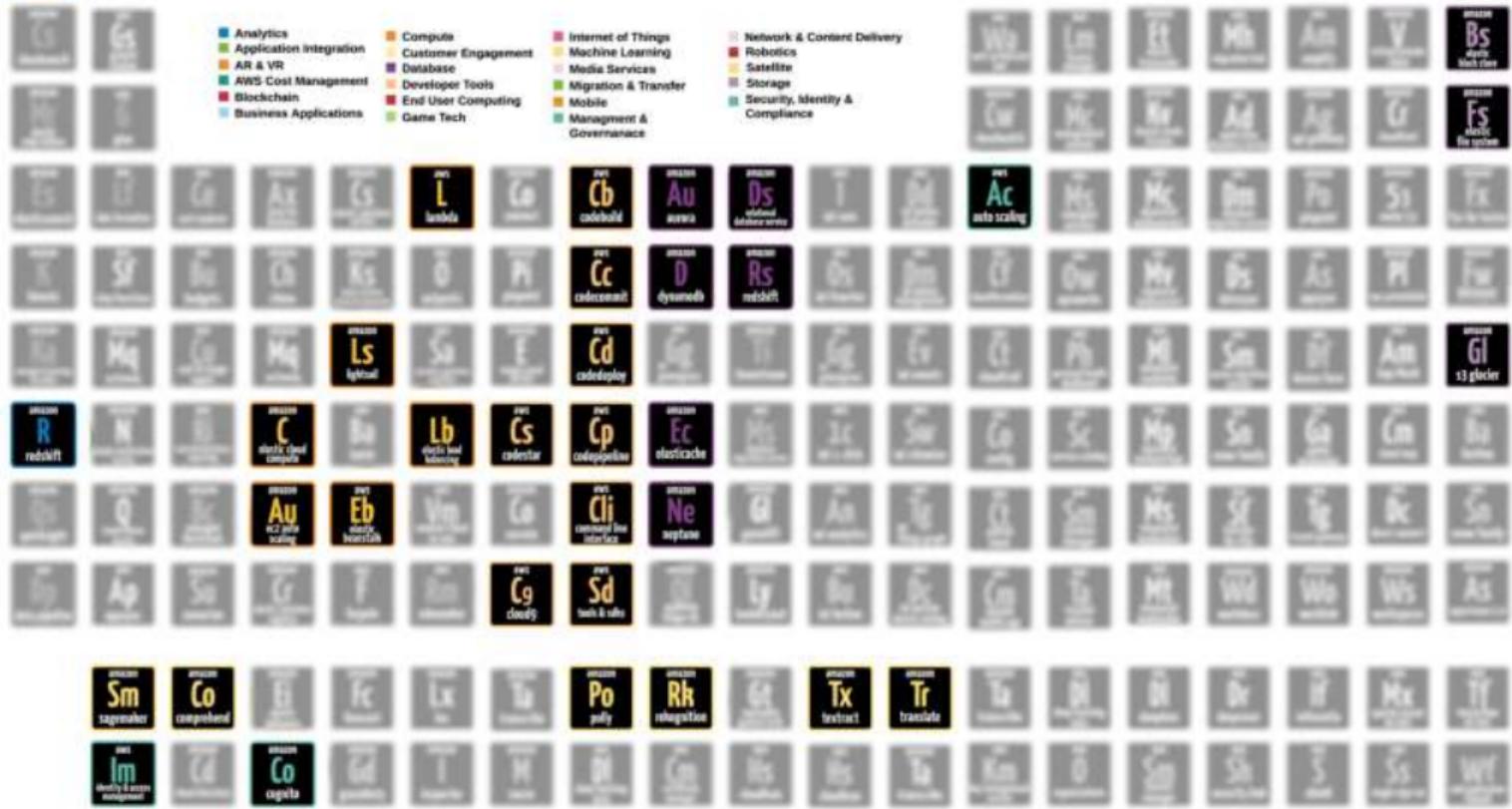
by [@jasonclegg](#)

<b>A</b> athena	<b>S3</b> simple storage service
<b>Gs</b> general service	<b>Sg</b> storage gateway
<b>Mr</b> machine learning	<b>V</b> virtual private cloud
<b>G</b> glacier	<b>Ws</b> workspaces
<b>Analytics</b>	<b>Amplify</b>
<b>Application Integration</b>	<b>Amplify</b>
<b>AR &amp; VR</b>	<b>Amplify</b>
<b>AWS Cost Management</b>	<b>Amplify</b>
<b>Blockchain</b>	<b>Cr</b> cloudfront
<b>Business Applications</b>	<b>Fs</b> elastic file system
<b>Computer</b>	<b>Es</b> elastic search
<b>Customer Engagement</b>	<b>Et</b> elastic transcoder
<b>Database</b>	<b>Mh</b> migration hub
<b>Developer Tools</b>	<b>Ad</b> application discovery service
<b>End User Computing</b>	<b>Ag</b> api gateway
<b>Mobile</b>	<b>Dm</b> database migration service
<b>Game Tech</b>	<b>Po</b> pinpoint
<b>Internet of Things</b>	<b>53</b> route 53
<b>Machine Learning</b>	<b>Fx</b> fsx for hpc
<b>Media Services</b>	<b>Pl</b> sns for lambda
<b>Migration &amp; Transfer</b>	<b>Kv</b> amazon media streams
<b>Mobile</b>	<b>Mc</b> managed cloud services
<b>Management &amp; Governance</b>	<b>Ms</b> managed serverless
<b>Network &amp; Content Delivery</b>	<b>Mc</b> managed cloud connect
<b>Robotics</b>	<b>Dm</b> database migration service
<b>Satellite</b>	<b>Po</b> pinpoint
<b>Storage</b>	<b>53</b> route 53
<b>Security, Identity &amp; Compliance</b>	<b>Pl</b> sns for lambda
<b>Amazon</b>	<b>Fw</b> fsx for windows file server
<b>Amazon</b>	<b>As</b> appsync
<b>Amazon</b>	<b>Df</b> device farm
<b>Amazon</b>	<b>Gl</b> glacier
<b>Amazon</b>	<b>Sn</b> snow family
<b>Amazon</b>	<b>Cm</b> cloud map
<b>Amazon</b>	<b>Ba</b> batch
<b>Amazon</b>	<b>Qs</b> quicksight
<b>Amazon</b>	<b>Ri</b> redshift insights
<b>Amazon</b>	<b>N</b> simple notification service
<b>Amazon</b>	<b>Ri</b> redshift insights reporting
<b>Amazon</b>	<b>C</b> lambda
<b>Amazon</b>	<b>Ba</b> batch
<b>Amazon</b>	<b>Lb</b> elastic load balancer
<b>Amazon</b>	<b>Cs</b> cloudsearch
<b>Amazon</b>	<b>Cp</b> codepipeline
<b>Amazon</b>	<b>Ec</b> elastic cache
<b>Amazon</b>	<b>Ms</b> managed migration service
<b>Amazon</b>	<b>Ic</b> iot click
<b>Amazon</b>	<b>Sw</b> iot surveillance
<b>Amazon</b>	<b>Co</b> cataloging
<b>Amazon</b>	<b>Sc</b> service catalog
<b>Amazon</b>	<b>Mp</b> managed migration package
<b>Amazon</b>	<b>Sn</b> snow family
<b>Amazon</b>	<b>Ga</b> global acceleration
<b>Amazon</b>	<b>Cm</b> cloud map
<b>Amazon</b>	<b>Ba</b> batch
<b>Amazon</b>	<b>Q</b> simple queue service
<b>Amazon</b>	<b>Bc</b> managed blockchain
<b>Amazon</b>	<b>Au</b> elastic auto scaling
<b>Amazon</b>	<b>Eb</b> elastic beanstalk
<b>Amazon</b>	<b>Vm</b> amazon cloud9 on aws
<b>Amazon</b>	<b>Co</b> connect
<b>Amazon</b>	<b>Cli</b> command line interface
<b>Amazon</b>	<b>Ne</b> neptune
<b>Amazon</b>	<b>Gl</b> game lift
<b>Amazon</b>	<b>An</b> iot analytics
<b>Amazon</b>	<b>Tg</b> iot things graph
<b>Amazon</b>	<b>Ct</b> control tower
<b>Amazon</b>	<b>Sm</b> systems manager
<b>Amazon</b>	<b>Ms</b> managed migration service
<b>Amazon</b>	<b>Tg</b> transit gateway
<b>Amazon</b>	<b>Dc</b> direct connect
<b>Amazon</b>	<b>Sn</b> snow family
<b>Amazon</b>	<b>Dp</b> data pipeline
<b>Amazon</b>	<b>Ap</b> appsync
<b>Amazon</b>	<b>Su</b> surveillance
<b>Amazon</b>	<b>Cr</b> elastic container registry
<b>Amazon</b>	<b>F</b> forgette
<b>Amazon</b>	<b>Rm</b> recommender
<b>Amazon</b>	<b>Cg</b> cloud9
<b>Amazon</b>	<b>Sd</b> media & sales
<b>Amazon</b>	<b>OI</b> amazon iot things
<b>Amazon</b>	<b>Ly</b> lambda yard
<b>Amazon</b>	<b>Bu</b> iot button
<b>Amazon</b>	<b>Dc</b> iot partner device catalog
<b>Amazon</b>	<b>Cm</b> connected mobile app
<b>Amazon</b>	<b>Ta</b> iot tap device
<b>Amazon</b>	<b>Mt</b> managed migration tool
<b>Amazon</b>	<b>Wd</b> workdocs
<b>Amazon</b>	<b>Wo</b> worklink
<b>Amazon</b>	<b>Ws</b> workspaces
<b>Amazon</b>	<b>As</b> appsync 2.0
<b>Amazon</b>	<b>Sm</b> sage maker
<b>Amazon</b>	<b>Co</b> comprehend
<b>Amazon</b>	<b>Ei</b> elastic inference
<b>Amazon</b>	<b>Fc</b> forecast
<b>Amazon</b>	<b>Lx</b> lex
<b>Amazon</b>	<b>Pe</b> personalize
<b>Amazon</b>	<b>Po</b> poly
<b>Amazon</b>	<b>Rh</b> re荐荐
<b>Amazon</b>	<b>Gt</b> recommendation tool
<b>Amazon</b>	<b>Tx</b> translate
<b>Amazon</b>	<b>Tr</b> translate
<b>Amazon</b>	<b>Ta</b> transcribe
<b>Amazon</b>	<b>DI</b> deep learning infer
<b>Amazon</b>	<b>DI</b> deepspeech
<b>Amazon</b>	<b>Dr</b> deepracer
<b>Amazon</b>	<b>If</b> inferentia on aws
<b>Amazon</b>	<b>Mx</b> machine learning on aws
<b>Amazon</b>	<b>Tf</b> transformer on aws
<b>Amazon</b>	<b>Wf</b> web application firewall
<b>Amazon</b>	<b>Im</b> identity & access management
<b>Amazon</b>	<b>Cd</b> cloud directory
<b>Amazon</b>	<b>Co</b> cognito
<b>Amazon</b>	<b>Gd</b> guardduty
<b>Amazon</b>	<b>I</b> inspector
<b>Amazon</b>	<b>M</b> macie
<b>Amazon</b>	<b>Ar</b> artifact
<b>Amazon</b>	<b>Cm</b> certificate manager
<b>Amazon</b>	<b>Hs</b> cloudhsm
<b>Amazon</b>	<b>Ds</b> directory service
<b>Amazon</b>	<b>Fm</b> flexible machine learning
<b>Amazon</b>	<b>Km</b> key management service
<b>Amazon</b>	<b>O</b> organizations
<b>Amazon</b>	<b>Sm</b> serverless managed service
<b>Amazon</b>	<b>Sh</b> security hub
<b>Amazon</b>	<b>Ss</b> single sign-on
<b>Amazon</b>	<b>Wf</b> web application firewall



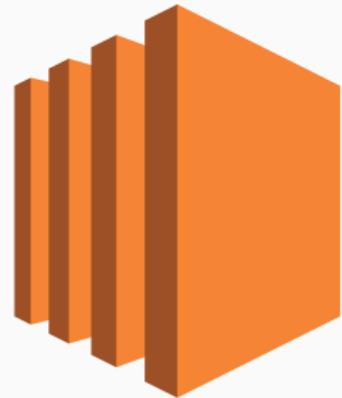
# Periodic Table of Amazon Web Services

My First Computer



# AMAZON ELASTIC COMPUTE CLOUD (EC2)

- (Virtual) Servers on demand



---

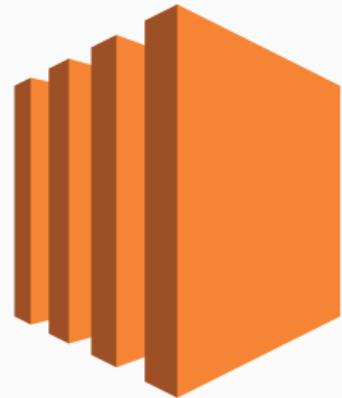
EC2 pricing [↗ web](#)

Azure: Virtual Machines [↗ web](#)

Google Cloud: Compute Engine [↗ web](#)

# AMAZON ELASTIC COMPUTE CLOUD (EC2)

- (Virtual) Servers on demand
- Different types of instances to suit computing needs



---

EC2 pricing [↗ web](#)

Azure: Virtual Machines [↗ web](#)

Google Cloud: Compute Engine [↗ web](#)

# AMAZON ELASTIC COMPUTE CLOUD (EC2)

- (Virtual) Servers on demand
- Different types of instances to suit computing needs
- Per-second (or per-hour) billing



---

EC2 pricing [↗ web](#)

Azure: Virtual Machines [↗ web](#)

Google Cloud: Compute Engine [↗ web](#)

# AMAZON ELASTIC COMPUTE CLOUD (EC2)

- (Virtual) Servers on demand
- Different types of instances to suit computing needs
- Per-second (or per-hour) billing
- Data transfer **not** included!



---

EC2 pricing [↗ web](#)

Azure: Virtual Machines [↗ web](#)

Google Cloud: Compute Engine [↗ web](#)

# AMAZON ELASTIC COMPUTE CLOUD (EC2)

- (Virtual) Servers on demand
- Different types of instances to suit computing needs
- Per-second (or per-hour) billing
- Data transfer **not** included!
- Persistent storage **not** included!



---

EC2 pricing [↗ web](#)

Azure: Virtual Machines [↗ web](#)

Google Cloud: Compute Engine [↗ web](#)

# AMAZON ELASTIC COMPUTE CLOUD (EC2)

- (Virtual) Servers on demand
- Different types of instances to suit computing needs
- Per-second (or per-hour) billing
- Data transfer **not** included!
- Persistent storage **not** included!
- Scaling **not** included!



---

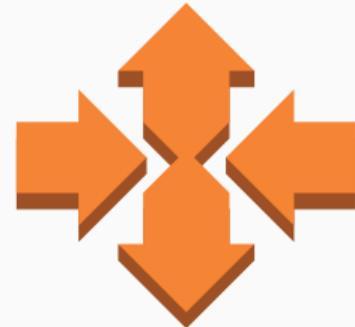
EC2 pricing [↗ web](#)

Azure: Virtual Machines [↗ web](#)

Google Cloud: Compute Engine [↗ web](#)

# AMAZON EC2 AUTO SCALING

- *Scaling is the ability to increase or decrease the compute capacity of your application*

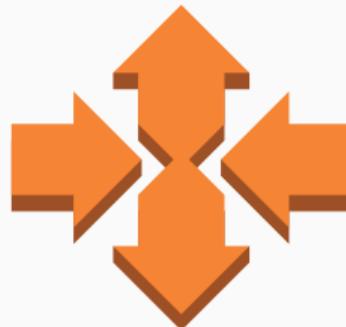


---

Azure: Virtual Machine Scale Sets [\[web\]](#)  
Google Cloud: Load Balancing [\[web\]](#)

# AMAZON EC2 AUTO SCALING

- *Scaling is the ability to increase or decrease the compute capacity of your application*
- Scale your application manually, on a scheduled basis or on demand

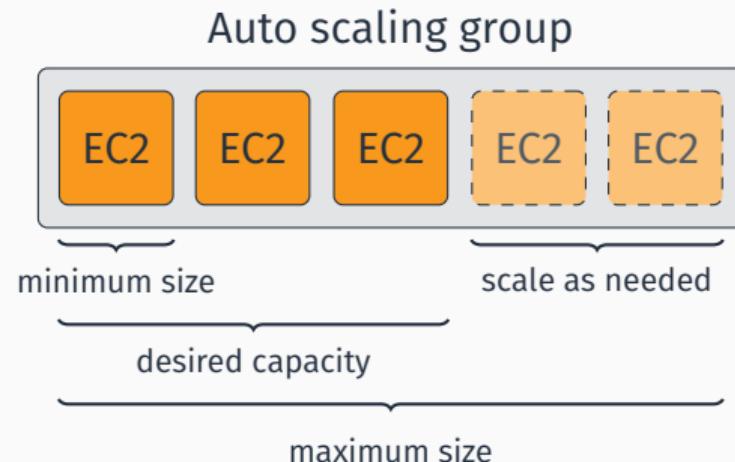


---

Azure: Virtual Machine Scale Sets  web

Google Cloud: Load Balancing  web

## AMAZON EC2 AUTO SCALING: DETAILS



## AMAZON ELASTIC LOAD BALANCING (ELB)

- Distributes incoming traffic across multiple EC2 instances



---

Azure: Load Balancer  web

# AMAZON ELASTIC LOAD BALANCING (ELB)

- Distributes incoming traffic across multiple EC2 instances
- Pay-per-use billing



---

Azure: Load Balancer  web

# AMAZON ELASTIC LOAD BALANCING (ELB)

- Distributes incoming traffic across multiple EC2 instances
- Pay-per-use billing
  - Execution time



---

Azure: Load Balancer  web

# AMAZON ELASTIC LOAD BALANCING (ELB)

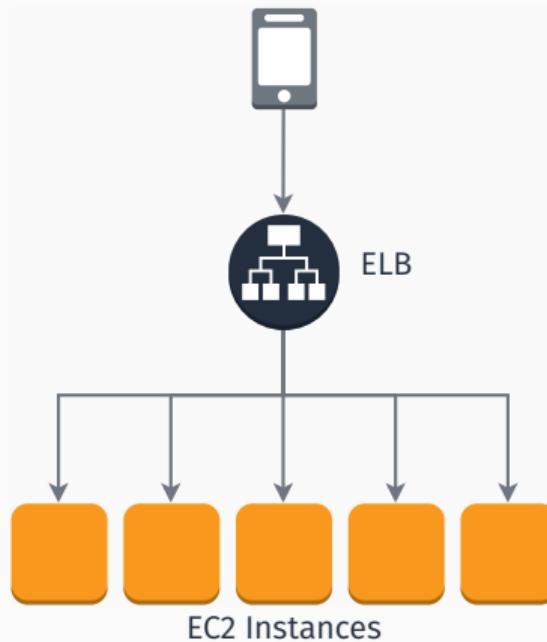
- Distributes incoming traffic across multiple EC2 instances
- Pay-per-use billing
  - Execution time
  - Number of requests / traffic



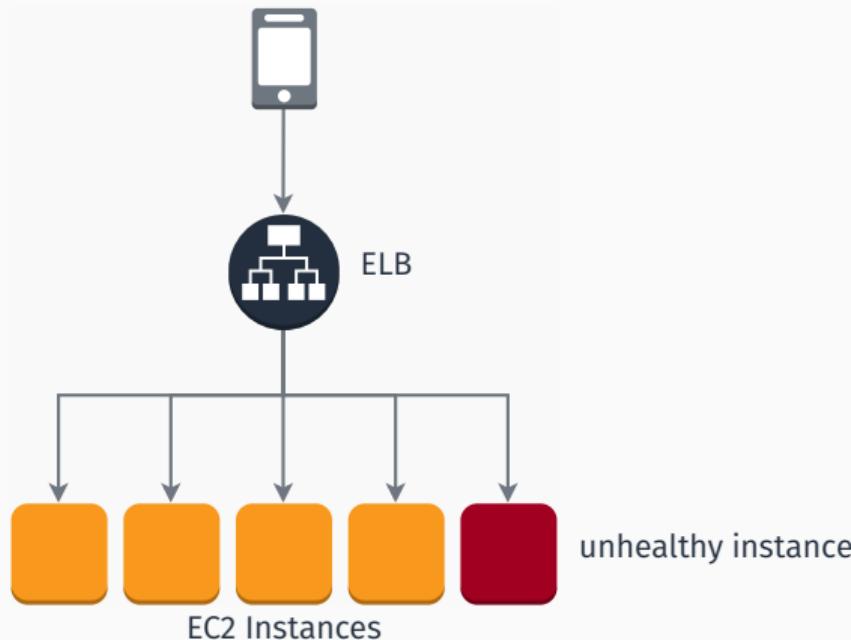
---

Azure: Load Balancer  web

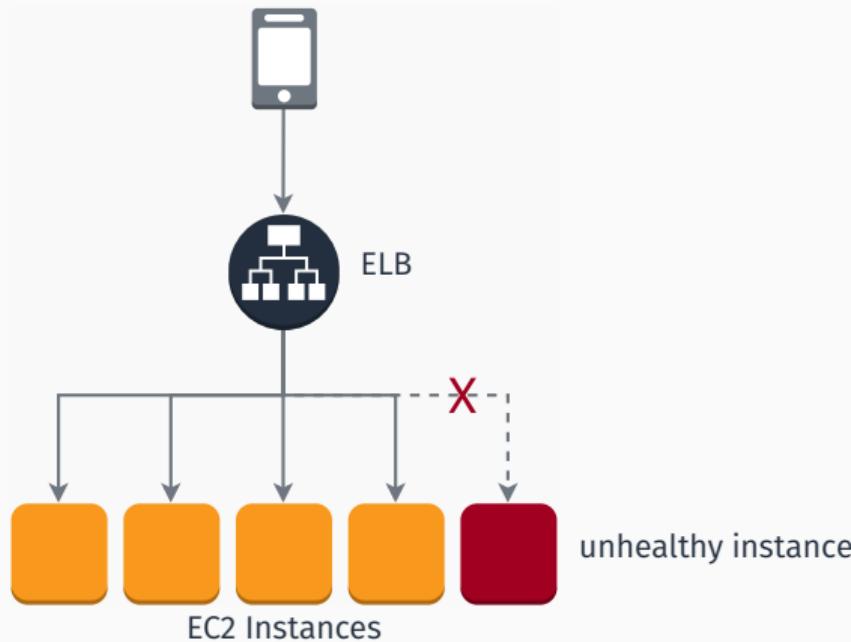
# AMAZON ELASTIC LOAD BALANCING (ELB)



# AMAZON ELASTIC LOAD BALANCING (ELB)



# AMAZON ELASTIC LOAD BALANCING (ELB)



## CLOUD STORAGE PRODUCTS

- Elastic Block Storage (EBS)
  - Persistent local storage for EC2 instances.



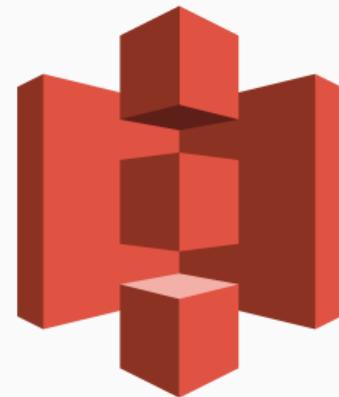
## CLOUD STORAGE PRODUCTS

- Elastic Block Storage (EBS)
  - Persistent local storage for EC2 instances.
- Elastic File System (EFS)
  - File system interface to share data between EC2 instances.



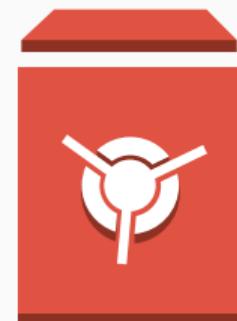
## CLOUD STORAGE PRODUCTS

- Elastic Block Storage (EBS)
  - Persistent local storage for EC2 instances.
- Elastic File System (EFS)
  - File system interface to share data between EC2 instances.
- Simple Storage Service (S3)



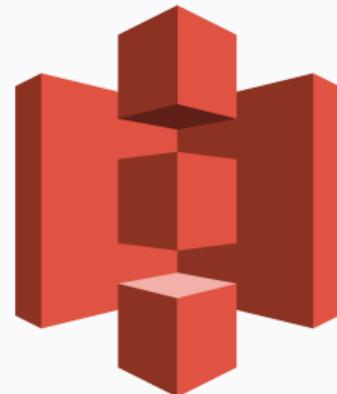
## CLOUD STORAGE PRODUCTS

- Elastic Block Storage (EBS)
  - Persistent local storage for EC2 instances.
- Elastic File System (EFS)
  - File system interface to share data between EC2 instances.
- Simple Storage Service (S3)
- Glacier
  - Durable and cheap long-term storage.



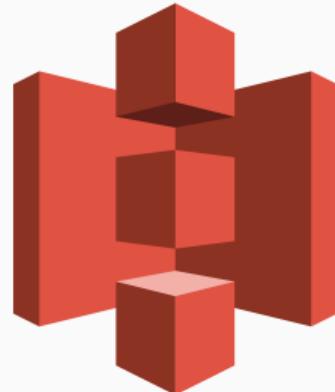
## AMAZON SIMPLE STORAGE SERVICE (S3)

- *store and retrieve any amount of data from anywhere*



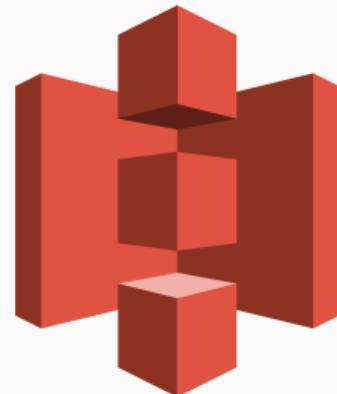
## AMAZON SIMPLE STORAGE SERVICE (S3)

- *store and retrieve any amount of data from anywhere*
- 99.99999999% durability (nine nines!)



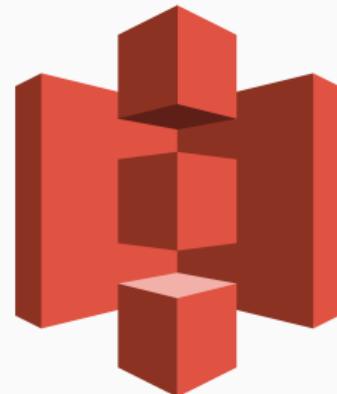
## AMAZON SIMPLE STORAGE SERVICE (S3)

- *store and retrieve any amount of data from anywhere*
- 99.99999999% durability (nine nines!)
- Data is distributed across a *minimum* of three availability zones



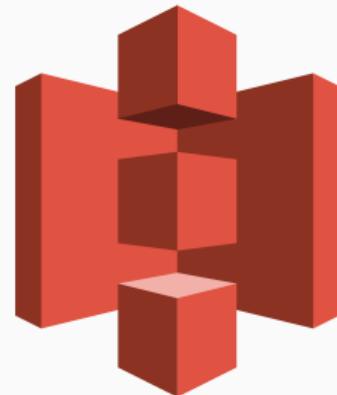
## AMAZON SIMPLE STORAGE SERVICE (S3)

- *store and retrieve any amount of data from anywhere*
- 99.99999999% durability (nine nines!)
- Data is distributed across a *minimum* of three availability zones
- A logical unit of storage is a *bucket*



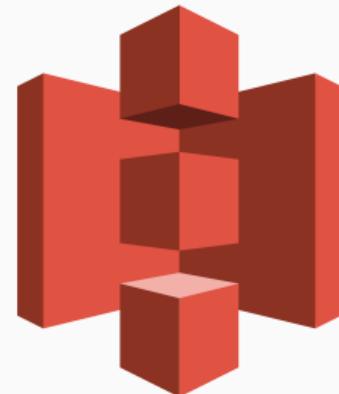
## AMAZON SIMPLE STORAGE SERVICE (S3)

- *store and retrieve any amount of data from anywhere*
- 99.99999999% durability (nine nines!)
- Data is distributed across a *minimum* of three availability zones
- A logical unit of storage is a *bucket*
- Multiple storage classes



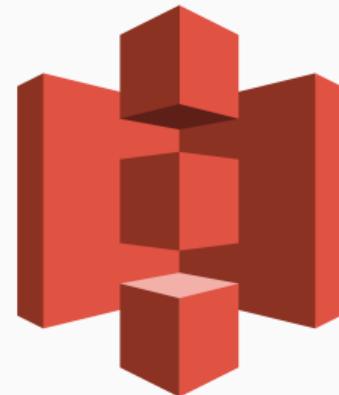
## AMAZON SIMPLE STORAGE SERVICE (S3)

- *store and retrieve any amount of data from anywhere*
- 99.99999999% durability (nine nines!)
- Data is distributed across a *minimum* of three availability zones
- A logical unit of storage is a *bucket*
- Multiple storage classes
  - Standard



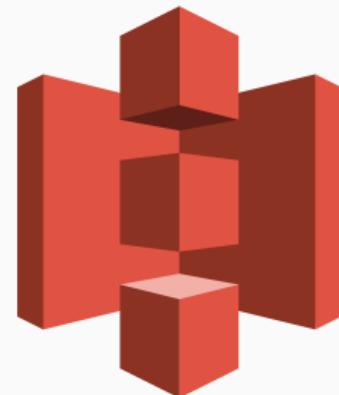
## AMAZON SIMPLE STORAGE SERVICE (S3)

- *store and retrieve any amount of data from anywhere*
- 99.99999999% durability (nine nines!)
- Data is distributed across a *minimum* of three availability zones
- A logical unit of storage is a *bucket*
- Multiple storage classes
  - Standard
  - Infrequent Access



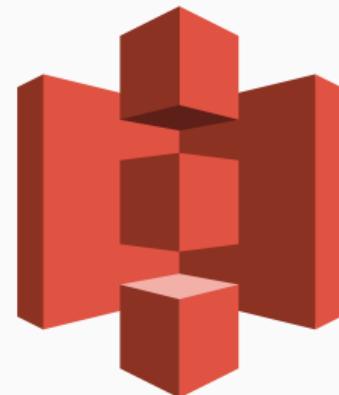
## AMAZON SIMPLE STORAGE SERVICE (S3)

- *store and retrieve any amount of data from anywhere*
- 99.99999999% durability (nine nines!)
- Data is distributed across a *minimum* of three availability zones
- A logical unit of storage is a *bucket*
- Multiple storage classes
  - Standard
  - Infrequent Access
  - One zone-Infrequent Access



## AMAZON SIMPLE STORAGE SERVICE (S3)

- *store and retrieve any amount of data from anywhere*
- 99.99999999% durability (nine nines!)
- Data is distributed across a *minimum* of three availability zones
- A logical unit of storage is a *bucket*
- Multiple storage classes
  - Standard
  - Infrequent Access
  - One zone-Infrequent Access
  - Amazon Glacier



## AMAZON SIMPLE STORAGE SERVICE (S3) - MORE

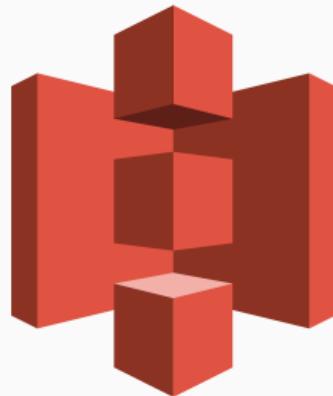
Pricing:

Storage class	Storage (per month)	Retrieval (per 1K req.)
Standard	\$0.022 per GB	\$0.0004
Infrequent access	\$0.0125 per GB	\$0.001
IA single zone	\$0.01 per GB	\$0.001
Glacier	\$0.004 per GB	\$0.0004

Table 1: S3 pricing (Ireland)

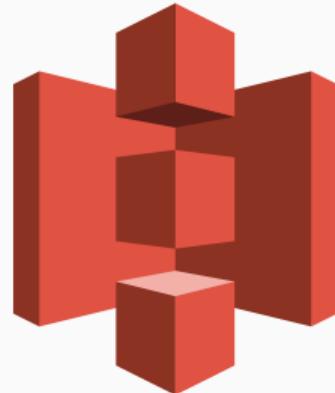
## AMAZON SIMPLE STORAGE SERVICE (S3) - MORE

- Well-integrated with other services



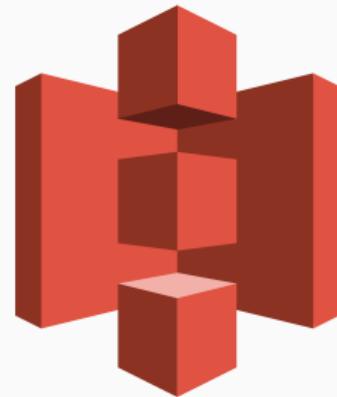
## AMAZON SIMPLE STORAGE SERVICE (S3) - MORE

- Well-integrated with other services
  - Machine Learning



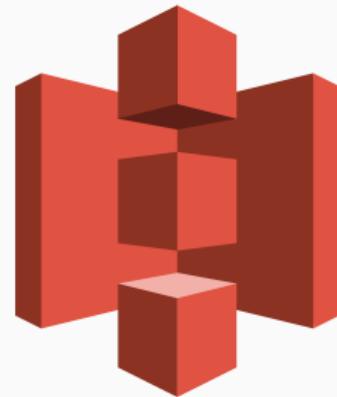
## AMAZON SIMPLE STORAGE SERVICE (S3) - MORE

- Well-integrated with other services
  - Machine Learning
  - Big Data Analysis



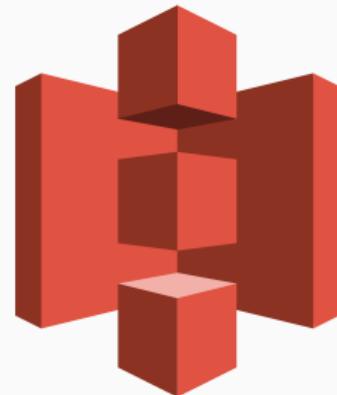
## AMAZON SIMPLE STORAGE SERVICE (S3) - MORE

- Well-integrated with other services
  - Machine Learning
  - Big Data Analysis
- REST API

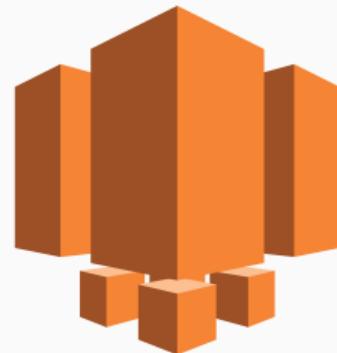


## AMAZON SIMPLE STORAGE SERVICE (S3) - MORE

- Well-integrated with other services
  - Machine Learning
  - Big Data Analysis
- REST API
- Can be used to host static websites



- A lightweight, simplified offer

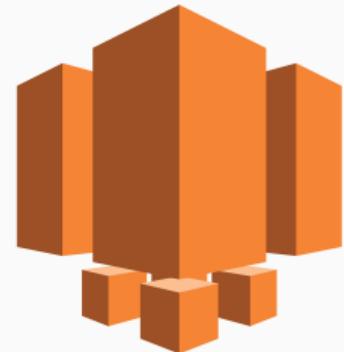


---

Websites: [EC2](#) [Lightsail](#)

## AMAZON LIGHTSAIL

- A lightweight, simplified offer
- Bundles computing, storage, and networking capacity

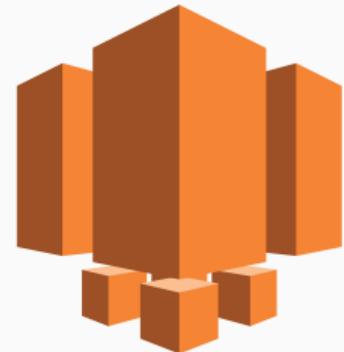


---

Websites: [EC2](#) [Lightsail](#)

## AMAZON LIGHTSAIL

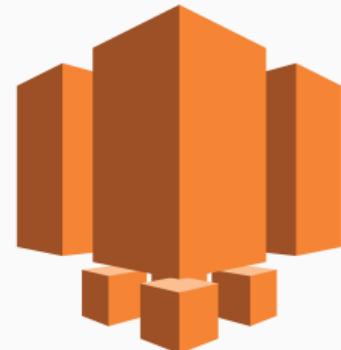
- A lightweight, simplified offer
- Bundles computing, storage, and networking capacity
- Preconfigured instances for



---

Websites: [EC2](#) [Lightsail](#)

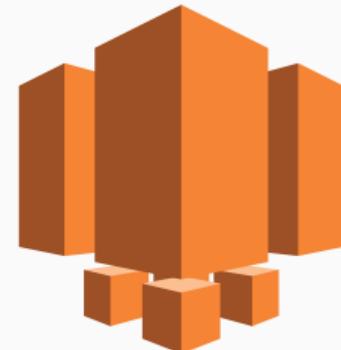
- A lightweight, simplified offer
- Bundles computing, storage, and networking capacity
- Preconfigured instances for
  - Debian, Windows Server, ...



---

Websites: [EC2](#) [Lightsail](#)

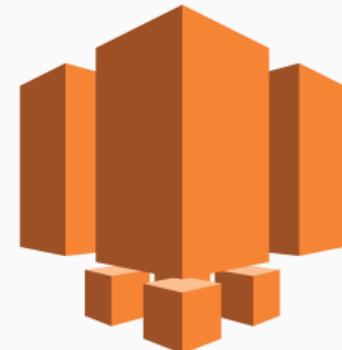
- A lightweight, simplified offer
- Bundles computing, storage, and networking capacity
- Preconfigured instances for
  - Debian, Windows Server, ...
  - Wordpress, Magento, Redmine, ...



---

Websites: [EC2](#) [Lightsail](#)

- A lightweight, simplified offer
- Bundles computing, storage, and networking capacity
- Preconfigured instances for
  - Debian, Windows Server, ...
  - Wordpress, Magento, Redmine, ...
  - LAMP stack, Nginx, ...



---

Websites: [EC2](#) [Lightsail](#)

- A lightweight, simplified offer
- Bundles computing, storage, and networking capacity
- Preconfigured instances for
  - Debian, Windows Server, ...
  - Wordpress, Magento, Redmine, ...
  - LAMP stack, Nginx, ...
- Low and **predictable** monthly costs



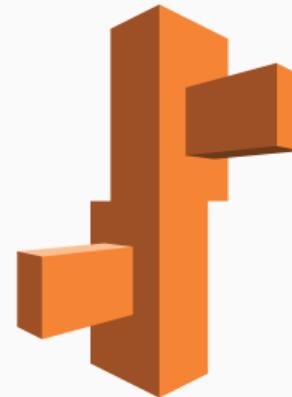
---

Websites: [EC2](#) [Lightsail](#)

## AMAZON ELASTIC BEANSTALK

---

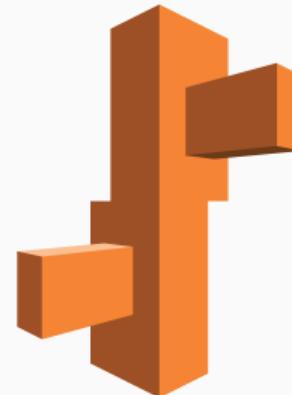
- “Easy to begin, impossible to outgrow”



## AMAZON ELASTIC BEANSTALK

---

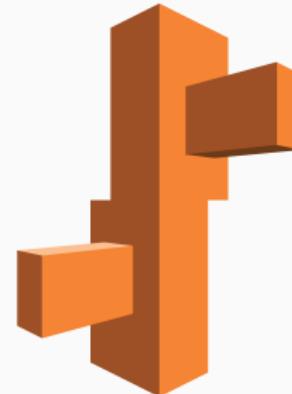
- “*Easy to begin, impossible to outgrow*”
- Easy-to-use service to deploy web apps



## AMAZON ELASTIC BEANSTALK

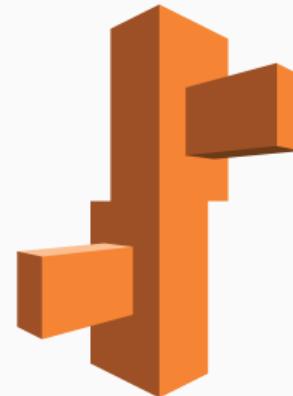
---

- “*Easy to begin, impossible to outgrow*”
- Easy-to-use service to deploy web apps
- Supports Apache, Nginx, IIS and more



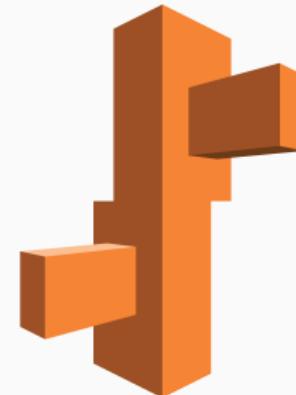
## AMAZON ELASTIC BEANSTALK

- “*Easy to begin, impossible to outgrow*”
- Easy-to-use service to deploy web apps
- Supports Apache, Nginx, IIS and more
- Supports Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker



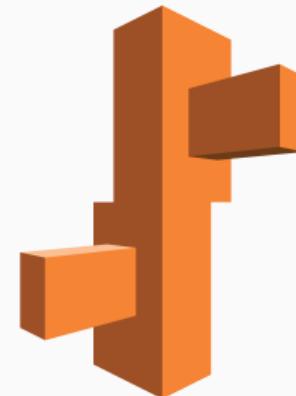
## AMAZON ELASTIC BEANSTALK

- “Easy to begin, impossible to outgrow”
- Easy-to-use service to deploy web apps
- Supports Apache, Nginx, IIS and more
- Supports Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker
- Manages auto-scaling, load balancing, health monitoring



## AMAZON ELASTIC BEANSTALK

- “Easy to begin, impossible to outgrow”
- Easy-to-use service to deploy web apps
- Supports Apache, Nginx, IIS and more
- Supports Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker
- Manages auto-scaling, load balancing, health monitoring
- Customizable



## AMAZON ELASTIC BEANSTALK

- “Easy to begin, impossible to outgrow”
- Easy-to-use service to deploy web apps
- Supports Apache, Nginx, IIS and more
- Supports Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker
- Manages auto-scaling, load balancing, health monitoring
- Customizable
- Free of charge. Pay only for the AWS resources you use.



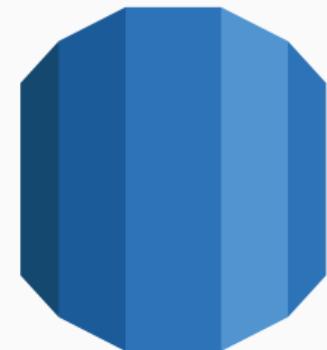
## RELATIONAL DATABASE SERVICE (RDS)

- Set up, operate a relational database in the cloud.



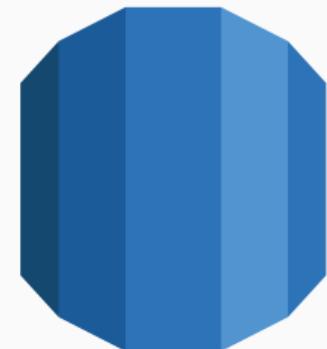
## RELATIONAL DATABASE SERVICE (RDS)

- Set up, operate a relational database in the cloud.
- Takes care of backups, patching.



## RELATIONAL DATABASE SERVICE (RDS)

- Set up, operate a relational database in the cloud.
- Takes care of backups, patching.
- Supports:



## RELATIONAL DATABASE SERVICE (RDS)

- Set up, operate a relational database in the cloud.
- Takes care of backups, patching.
- Supports:
  - MySQL, PostgreSQL, MariaDB



## RELATIONAL DATABASE SERVICE (RDS)

- Set up, operate a relational database in the cloud.
- Takes care of backups, patching.
- Supports:
  - MySQL, PostgreSQL, MariaDB
  - Oracle, MS SQL Server



## RELATIONAL DATABASE SERVICE (RDS)

- Set up, operate a relational database in the cloud.
- Takes care of backups, patching.
- Supports:
  - MySQL, PostgreSQL, MariaDB
  - Oracle, MS SQL Server
  - Amazon Aurora





# Practice time!

## SCENARIO

---

- You just had a million dollar idea.

## SCENARIO

---

- You just had a million dollar idea.
- Your web application is finished. It looks great and works like a charm.

## SCENARIO

---

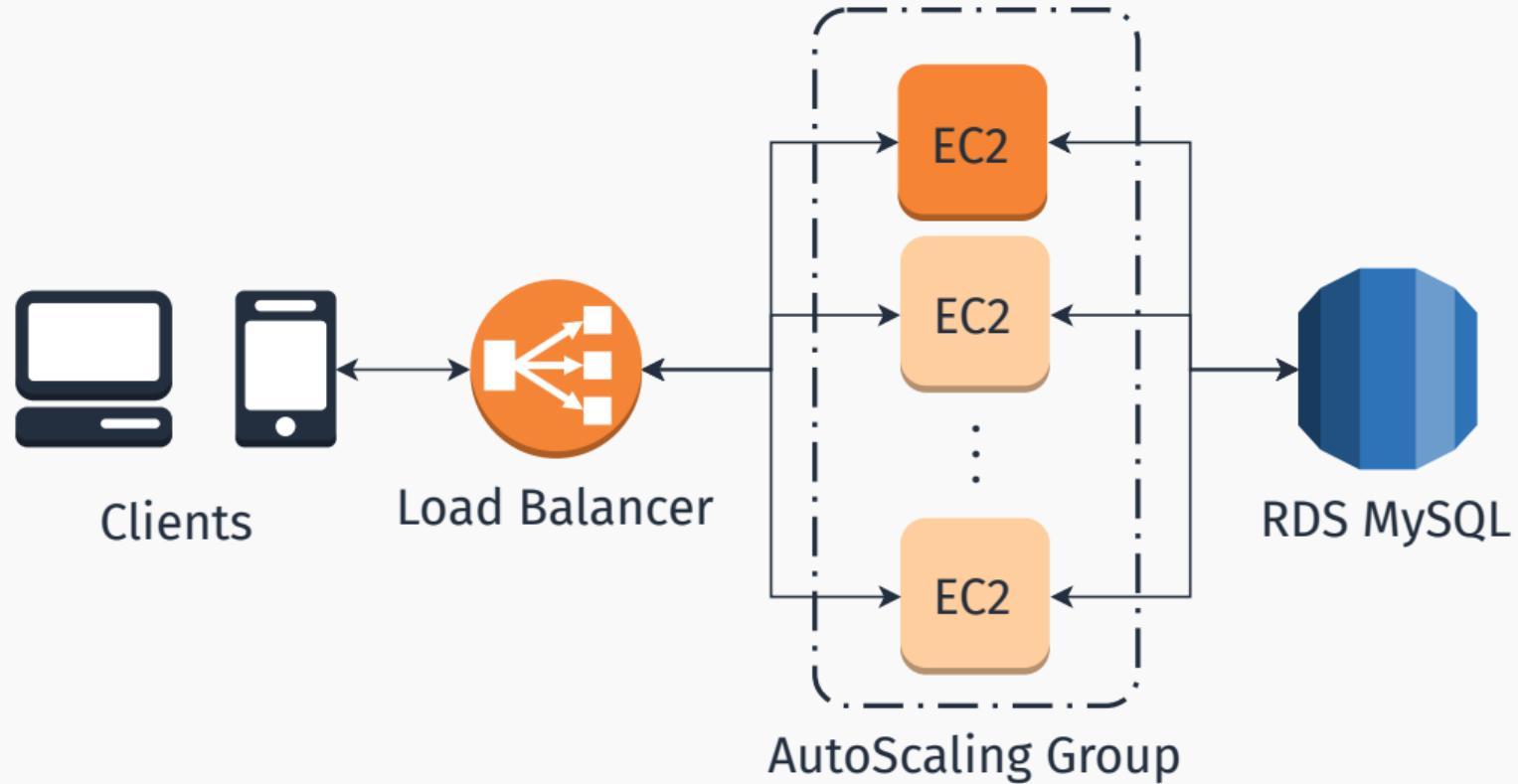
- You just had a million dollar idea.
- Your web application is finished. It looks great and works like a charm.
- You're ready to start earning some dough!

The web app is built with a classic LAMP stack:

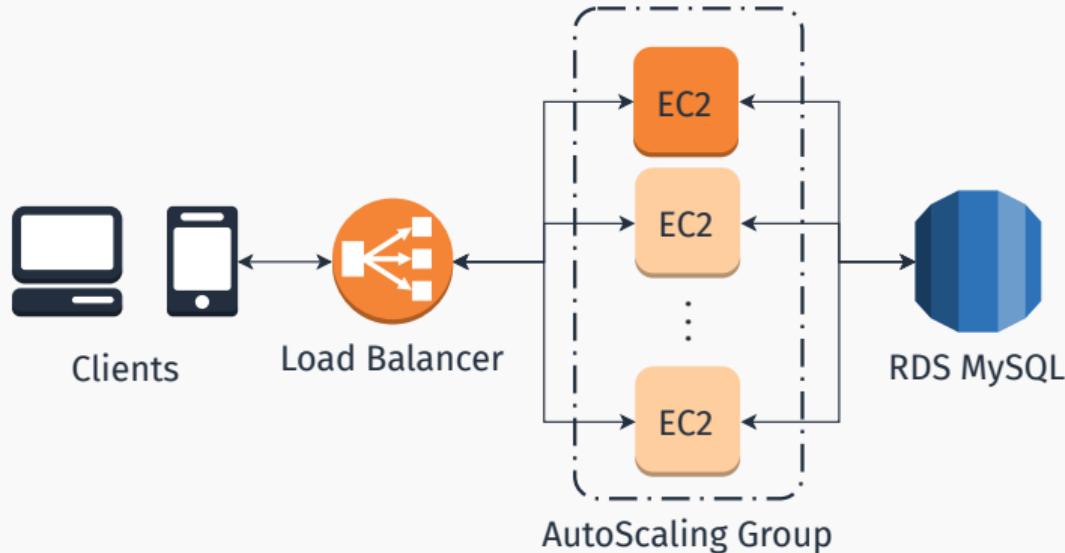
- Linux
- Apache web server
- MySQL relational database
- PHP

**How WOULD YOU DO IT?**

## PROPOSED ARCHITECTURE



## PROPOSED ARCHITECTURE



- Is this *really* scalable?

## TAKE HOME MESSAGES

---

## TAKE HOME MESSAGES

---

- Cloud computing and service models

## TAKE HOME MESSAGES

---

- Cloud computing and service models
- Core AWS services:

## TAKE HOME MESSAGES

---

- Cloud computing and service models
- Core AWS services:
  - EC2, AutoScaling groups

## TAKE HOME MESSAGES

---

- Cloud computing and service models
- Core AWS services:
  - EC2, AutoScaling groups
  - S3, EBS/EFS

## TAKE HOME MESSAGES

---

- Cloud computing and service models
- Core AWS services:
  - EC2, AutoScaling groups
  - S3, EBS/EFS
  - RDS

## TAKE HOME MESSAGES

---

- Cloud computing and service models
- Core AWS services:
  - EC2, AutoScaling groups
  - S3, EBS/EFS
  - RDS
- A cloud architecture for a classic web application on AWS

**SEE YOU TOMORROW!**

## REFERENCES I

- [Fle19] Flexera. *Cloud Computing Trends: 2019 State of the Cloud Survey*. Feb. 27, 2019. URL:  
<https://www.flexera.com/blog/cloud/2019/02/cloud-computing-trends-2019-state-of-the-cloud-survey/> (visited on 03/21/2020).

# Unit Testing with JUnit



*Get more out of your unit tests with Hamcrest and Mockito!*

Luigi Libero Lucio STARACE  
[luigiliberolucio.starace@unina.it](mailto:luigiliberolucio.starace@unina.it)

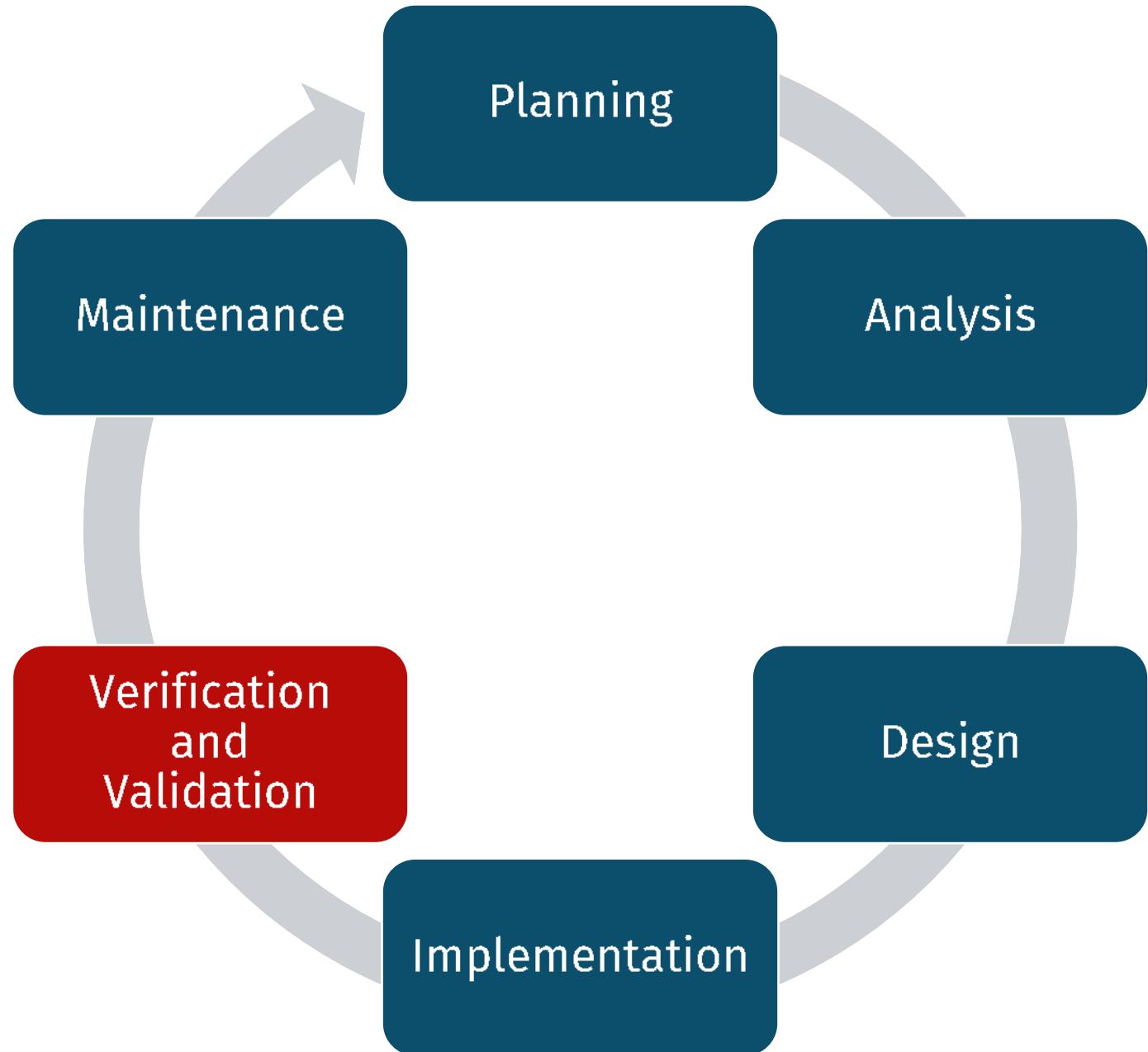
April 28, 2021  
University of Naples, Federico II

# What's the plan

1. Unit testing
2. JUnit 5
3. Assertion Frameworks
4. Testing in Isolation

# Unit Testing

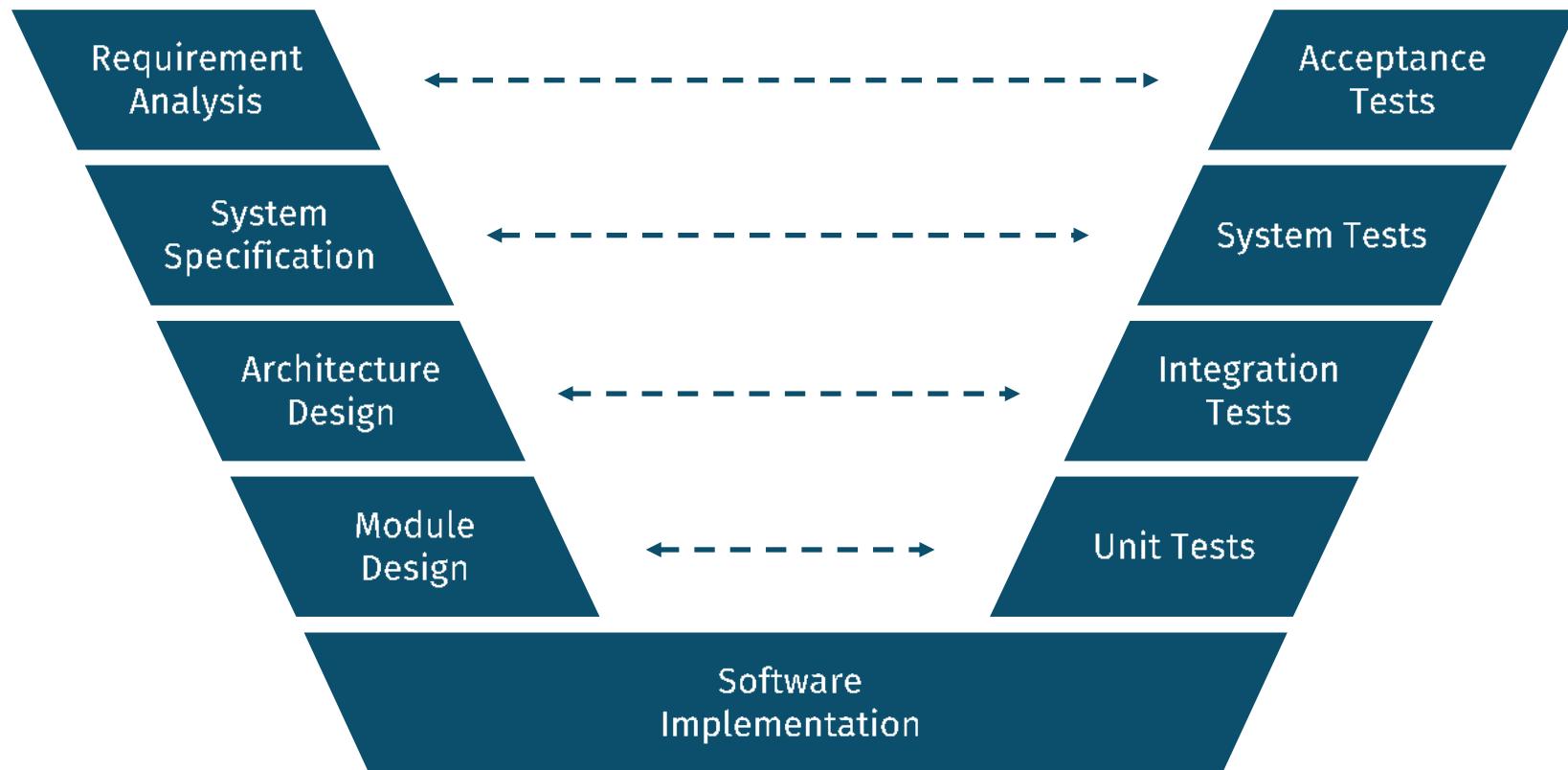
# The Software Life-cycle



# Software Verification

- How can we increase the confidence that our software works as intended?
- Static verification:
  - No code execution involved;
  - Code reviews, inspections with checklists, static analysis;
- Dynamic verification:
  - Testing!

# The V-Model

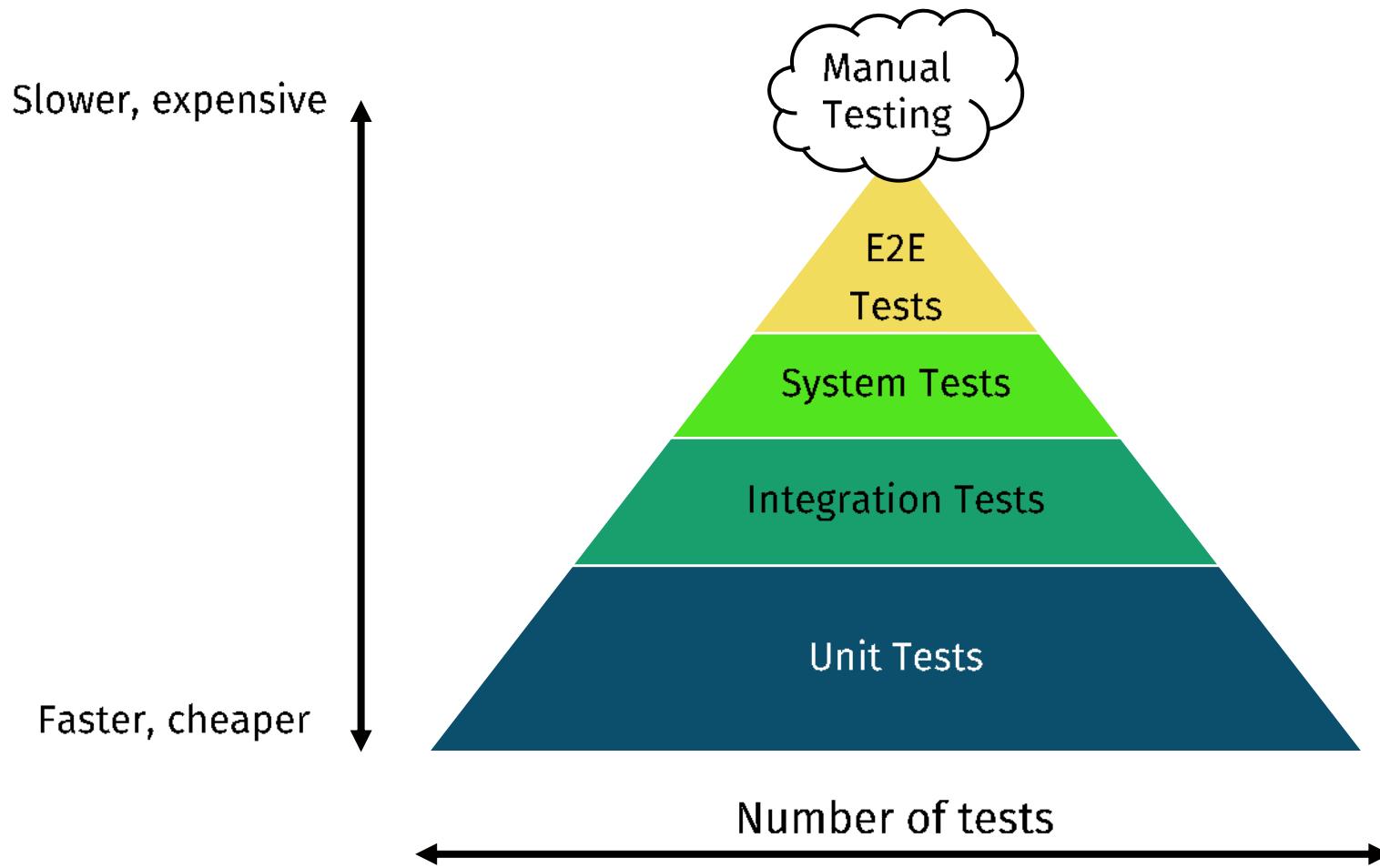


# Software Testing

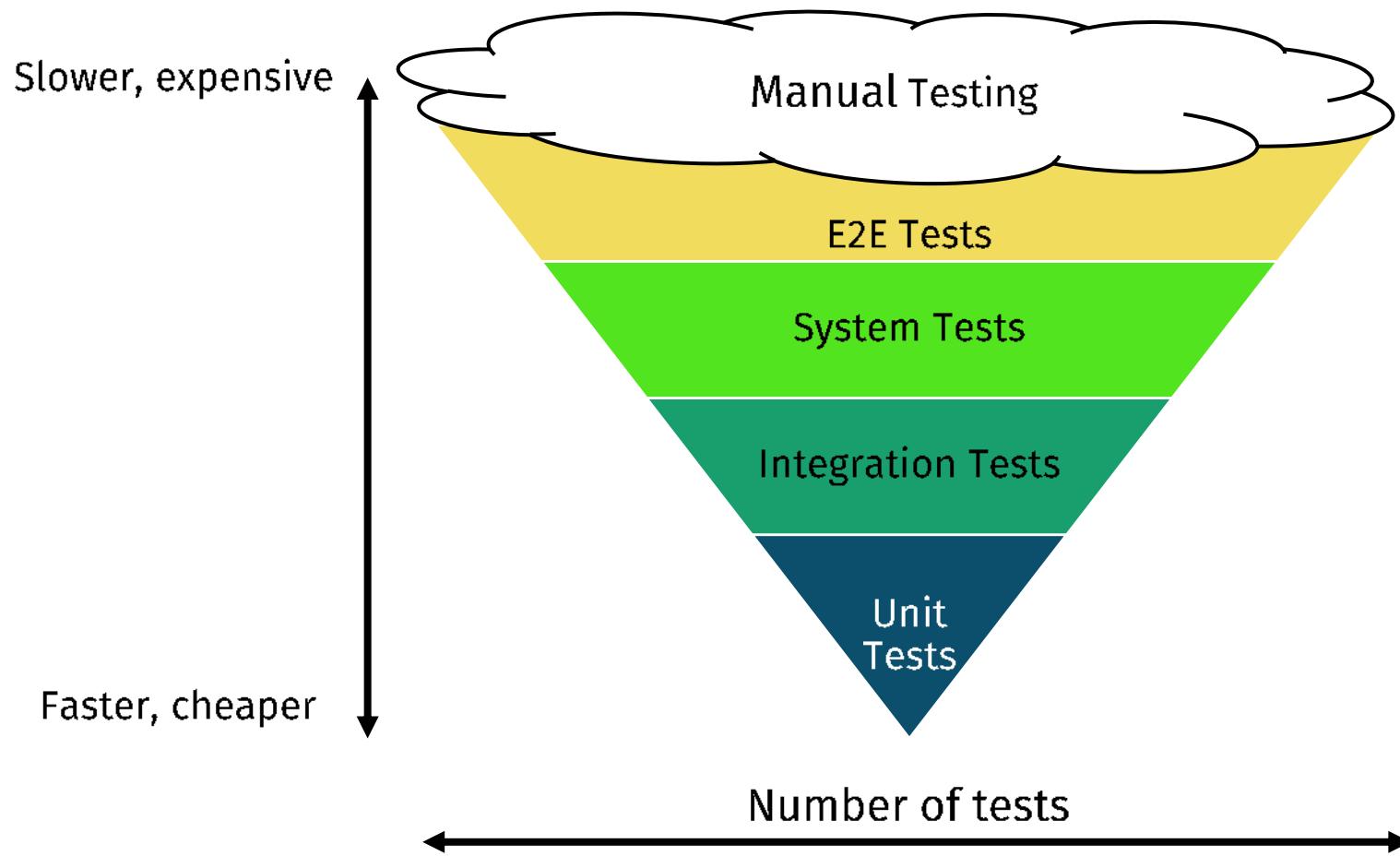
Can be performed at different levels:

- **Unit testing**
  - Tests a single *unit* (e.g. a class, or a method)
- **Integration testing**
  - Checking that different units work together as intended
- **System testing**
  - Targets the system as a whole, against its specification
- **Acceptance – End-to-End (E2E) testing**
  - Is the product acceptable for delivery?

# The Testing Pyramid



# The Testing Ice-cream Cone Anti-pattern



# What's Unit Testing?

A method by which individual *units* of source code are tested to determine whether they work as intended.

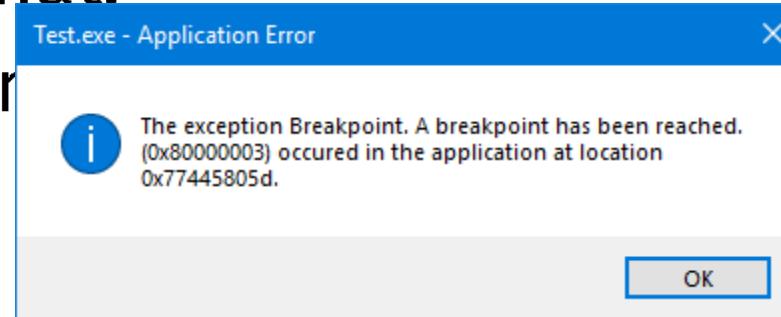
- So, what's a source code unit?
  - In object-oriented design, usually it's a class;
  - In procedural and functional approaches, it might be a single function;
  - The developer team decides what makes sense to be a unit.
- Unit tests can be organized in:
  - **Test cases**, which verify a certain behaviour of a single unit;
  - **Test suites**, i.e. group of test cases insisting on related behaviours.

# Reasons to Unit Test

- Early error detection;
- Supports maintenance (helps avoiding regressions);
- Improves design;
- Helps determining specifications;
- Product documentation.

# Excuses **not** to Unit Test

- I *never* make mistakes!
- Ain't got time for that!
- Management doesn't care



# Properties of a good Unit Test

- It is consistent, repeatable;
- It's fast;
- It's ***clean!***

“

*What makes a clean test? Three things. Readability, readability, and readability. Readability is perhaps even more important in unit tests than it is in production code.*

*What makes tests readable? The same thing that makes all code readable: clarity, simplicity, and density of expression. In a test you want to say a lot with as few expressions as possible.*

”

— Robert Martin, Clean Code



# JUnit

- Open-source framework for writing and running tests in Java;
- Originally written by *Erich Gamma* and *Kent Back*;
- One of many frameworks in the xUnit family:
  - nUnit, RUnit, PHPUnit, CppUnit, and more.
- Used in 30.7% of the top 10,000 Java projects on GitHub<sup>1</sup>.

[1] <https://blog.overops.com/githubs-10000-most-popular-java-projects-here-are-the-top-libraries-they-use/>

# On JUnit

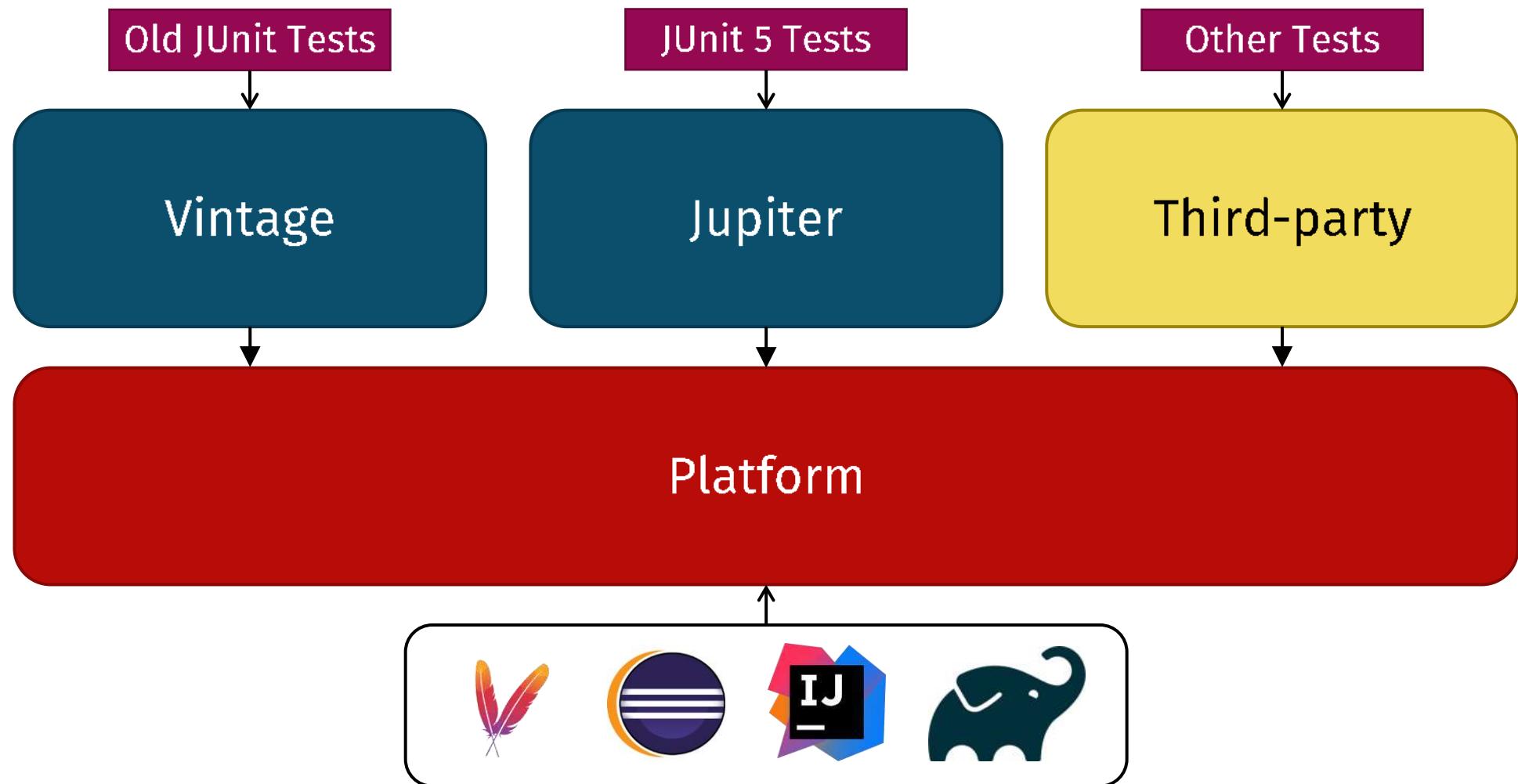
“

*Never in the field of software development have so many owed so much to so few lines of code.*

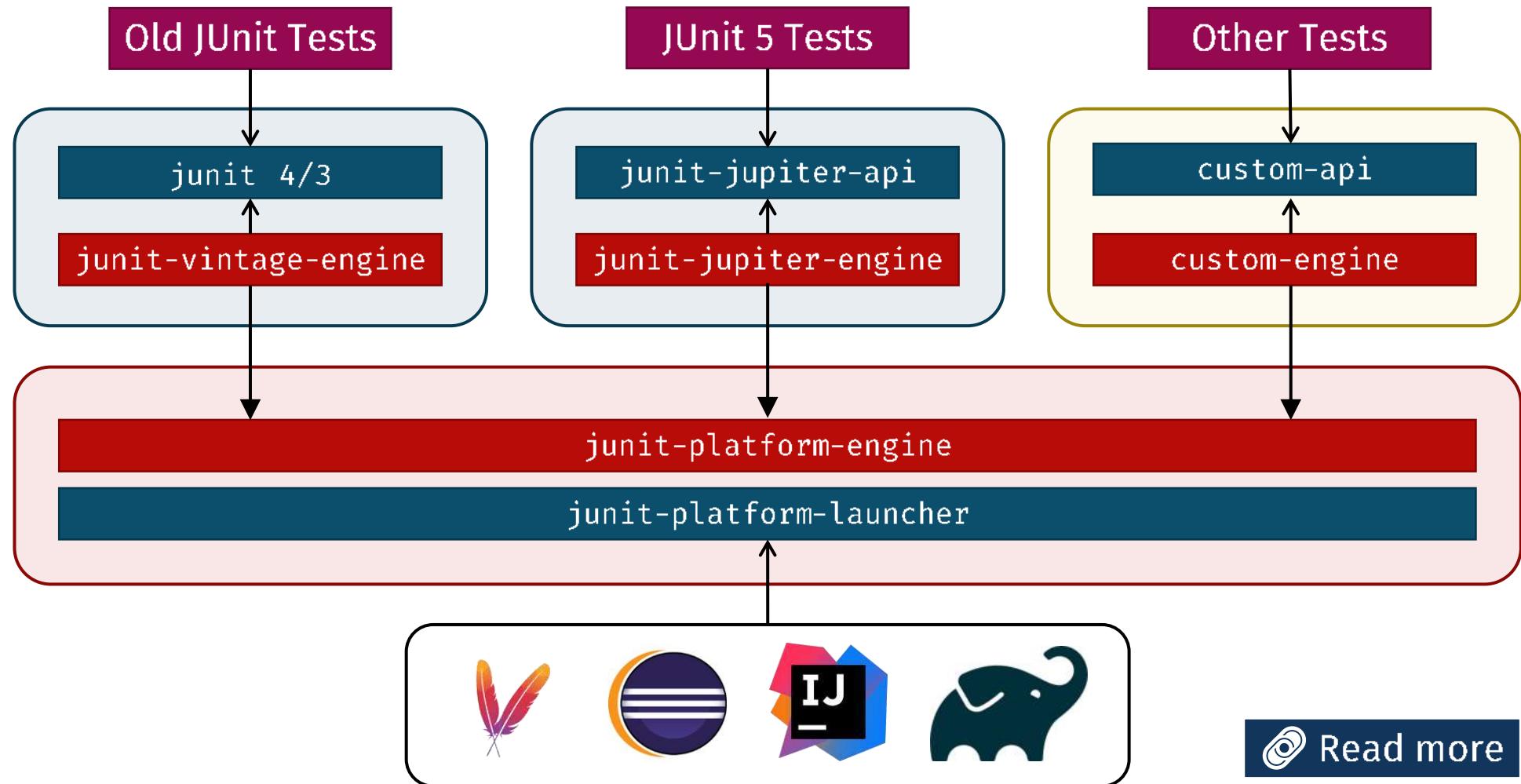
”

— Martin Fowler

# The JUnit 5 Architecture



# The JUnit 5 Architecture



# Installing JUnit 5

- Out-of-the-box support in many IDEs:
  - Eclipse, IntelliJ IDEA, NetBeans, VS Code
- Through build-tools like Maven, Ant, Gradle;
- Manually downloading the required artifacts.

# JUnit 5 – Arrange, Act, Assert

Test Annotation

```
import org.junit.jupiter.api.Test;  
import static org.junit.jupiter.api.Assertions.*;
```

Arrange

```
class GenericTest {  
    @Test  
    void shouldPerformOperationAsExpected() {
```

Act

```
        ClassUnderTest cut = new ClassUnderTest();  
        cut.initialize(list, of, parameters);
```

Assert

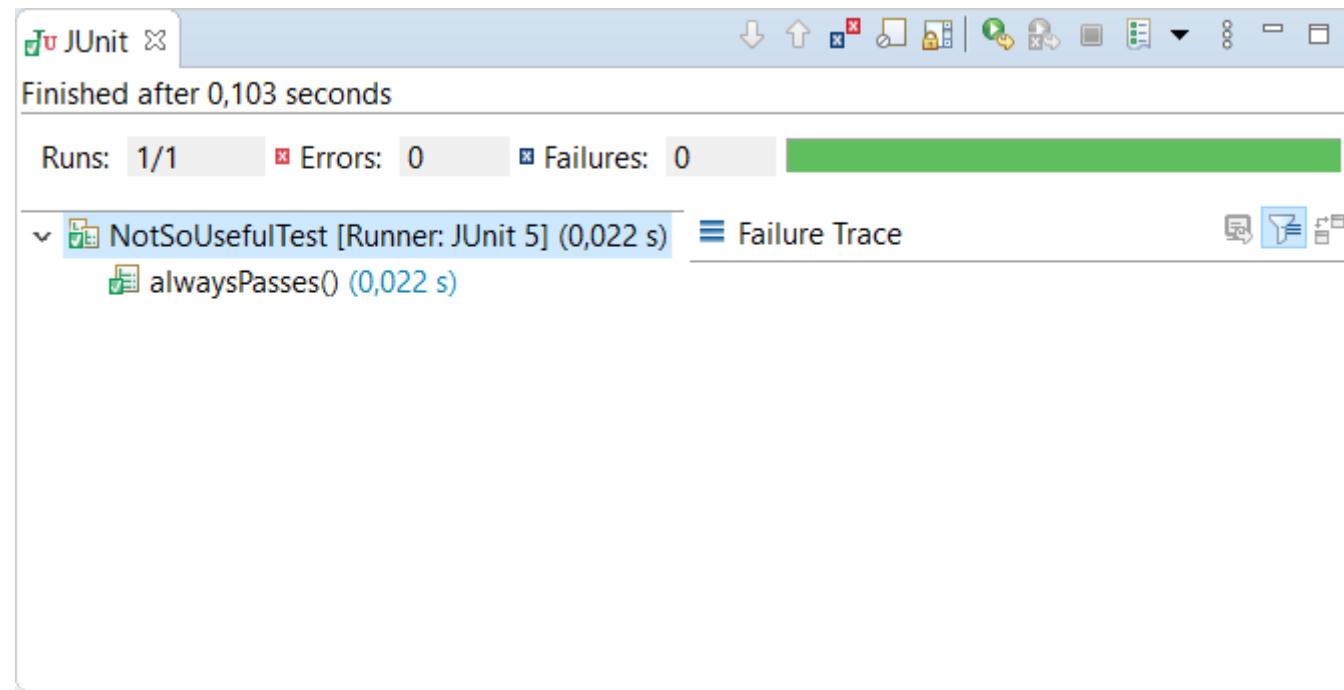
```
        cut.performOperation();
```

```
        assertTrue(cut.someCondition());  
        assertFalse(cut.someOtherCondition());
```

```
    }
```

# How do we run the tests? (1 of 3)

- In Eclipse, right click on the Test class, then *Run as > JUnit test*



# How do we run the tests? (2 of 3)

- JUnit test can be ran also from build tools
  - E.g., for Maven, check out the official `maven-surefire-plugin`
  - Useful in CI/CD;

# How do we run the tests? (2 of 3)

```
[INFO] --- maven-surefire-plugin:2.22.2:test (default-test) @ test ---
[INFO]
[INFO] -----
[INFO] T E S T S
[INFO] -----
[INFO] Running test.NotSoUsefulTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.016 s - in test.NotSoUsefulTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time:  2.386 s
[INFO] Finished at: 2020-04-11T18:46:42+02:00
[INFO] -----
```

# How do we run the tests? (3 of 3)

- The `ConsoleLauncher` is a command-line application to run tests.
- You can download it as an executable jar with all dependencies included.

# How do we run the tests? (3 of 3)

```
$> java -jar junit-platform-console-standalone-1.6.2.jar -cp build/classes/java/test --scan-classpath
└JUnit Jupiter
  └NotSoUsefulTest
    └alwaysPasses() ✓

Test run finished after 24 ms
[      0 containers found      ]
[      0 containers skipped     ]
[      0 containers started     ]
[      0 containers aborted     ]
[      0 containers successful  ]
[      0 containers failed      ]
[      1 tests found           ]
[      0 tests skipped          ]
[      1 tests started          ]
[      0 tests aborted          ]
[      1 tests successful        ]
[      0 tests failed           ]
```

# JUnit 5 – Lifecycle Annotations

- `@Test`
  - Used to mark test methods;
- `@BeforeAll` / `@AfterAll`
  - Denotes methods that should be executed before / after all test methods in the test class;
- `@BeforeEach` / `@AfterEach`
  - Denotes methods that should be executed before / after each test methods;

# JUnit 5 – Test Instance Lifecycle

- By default, JUnit creates a new instance of each test class before running each test method.
- To execute all test cases on the same test instance, you can annotate the test class with:

```
@TestInstance(Lifecycle.PER_CLASS)
```

# JUnit 5 – Assertions

- Assertions are utility methods we can use to declaratively assert conditions in tests;
- They're defined as static methods of the Assertions class;
- Failed assertions usually throw AssertionFailedError.

# JUnit 5 – Assertion examples

## assertTrue() and assertFalse()

```
int number = 0; boolean condition = true; String message = "ABCDEF";

// passes
assertTrue(condition);

// fails, with custom error message
assertFalse(number < 1, "Number shouldn't be less than 1.");

// fails, with error message supplier
assertTrue(message.contains("Z"),() ->
    String.format("Message (%s) should contain \"Z\"",message)
);
```

# JUnit 5 – Assertion examples

## Assert[Not]Equals() and assert[Not]Same()

```
Sheep dolly = new Sheep("Dolly");
Sheep otherDolly = dolly.clone();

assertEquals(dolly, otherDolly);
assertNotSame(dolly, otherDolly);
```

# JUnit 5 – Assertion examples

- assertNull() and assertNotNull()
- assertArrayEquals()
- assertIterableEquals()
- assertTimeout() and assertTimeoutPreemptively()
- assertThrows() and assertDoesNotThrow()

```
assertTimeoutPreemptively(Duration.ofMillis(300), () -> { object.doStuff(); });

assertThrows(RuntimeException.class, ()-> {
    throw new RuntimeException();
});
```

# JUnit 5 – Assertion examples

## AssertAll()

```
Person grandPa = getRandomGrandPa();

assertAll(
    () -> assertNotNull(grandPa.getName()      , "Name should not be null"),
    () -> assertTrue   (grandPa.getAge() > 50, "Should be older than 50"),
    () -> assertEquals ("M" , grandPa.getSex() , "Should be male")
);

// Same as this? No! AssertAll always executes all the assertions!
assertNotNull(grandPa.getName()      , "Name should not be null");
assertTrue   (grandPa.getAge() > 50, "Should be older than 50");
assertEquals ("M" , grandPa.getSex() , "Should be male");
```

# JUnit 5 – Test Naming

Test classes and methods can declare custom names with the `@DisplayName` annotation:

- Can contain spaces;
- Special characters;
- And even emojis (some men just want to watch the world burn).

```
@DisplayName("A special test case")
class DisplayNameDemo {

    @Test
    @DisplayName("Custom name")
    void testWithCustomName() {}

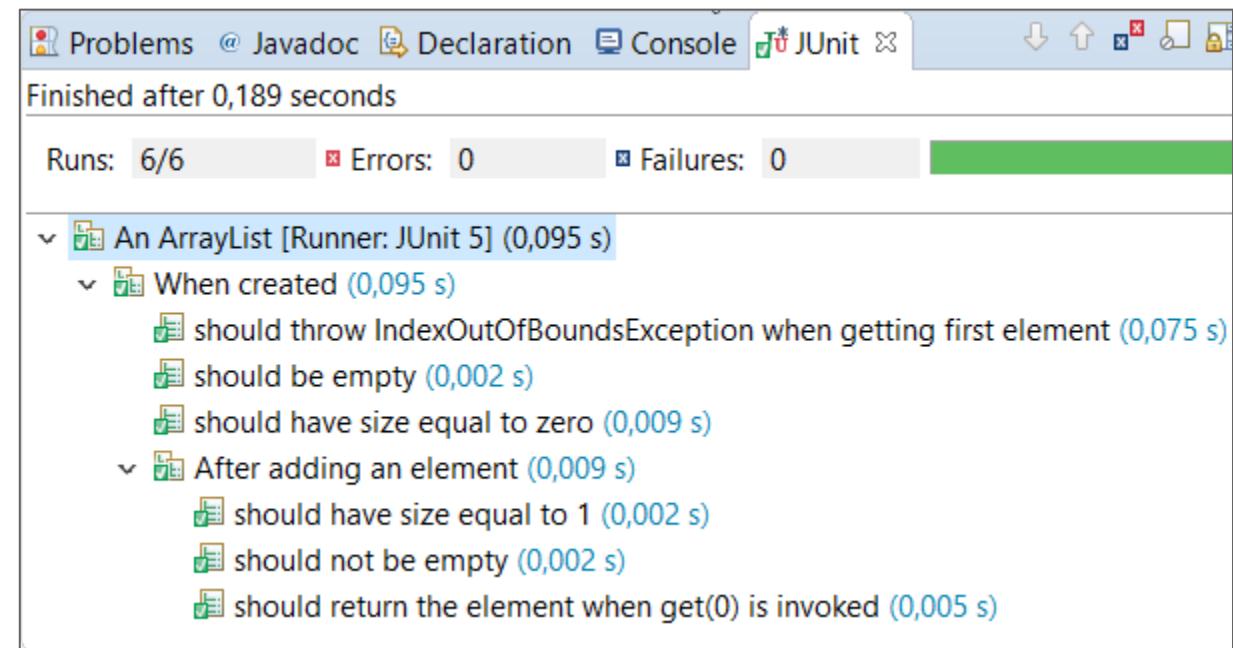
    @Test
    @DisplayName("JUnit")
    void testWithSpecialName() {}

    @Test
    @DisplayName("普遍存在")
    void testWithEmojiName()
}
```

# JUnit 5 – Nested Test Classes

With `@Nested` test classes, you can express relationship among different group of tests and better organize your test code.

```
@DisplayName("An ArrayList")
class NamingExample {
    @Nested
    @DisplayName("When created")
    public class TestEmptyArrayList {
        @Test
        @DisplayName("should be empty")
        void shouldBeEmpty() {}
    }
}
```



# JUnit 5 – Parametrized Tests

Sometimes, test cases differ only for their inputs and expected results.

How can we make our code **DRYer**?

```
@Test  
void shouldSumTwoPositiveNumbers(){  
    int a = 4, b= 6, expected = 10;  
    int actual = Calculator.sum(a, b);  
    assertEquals(actual, expected);  
}  
  
@Test  
void shouldSumTwoNegativeNumbers(){  
    int a = -3, b= -5, expected = -8;  
    int actual = Calculator.sum(a, b);  
    assertEquals(actual, expected);  
}  
  
//and so on...
```

# JUnit 5 – Parametrized Tests

This is more compact, but...

- Not very readable!
- Stops at the first failed assertion!
- It could be hard to find which assertion failed!

```
@Test  
void shouldSumCorrectly(){  
    int[][] inputs = {{4,6,10},{-3,-5,-8}};  
    for(int[] input : inputs) {  
        int a = input[0], b = input[1],  
            expected = input[2];  
        int actual = Calculator.sum(a, b);  
        assertEquals(actual, expected);  
    }  
}
```

# JUnit 5 – Parametrized Tests

This evaluates all assertions,  
but...

- Still not very readable!
- Still, It could be hard to find which assertion failed!

```
@Test  
void shouldSumCorrectlyV2(){  
    int[][] inputs = {{4,6,10},{-3,-5,-8}};  
    List<Executable> assertions =  
        new ArrayList<Executable>();  
    for(int[] input : inputs) {  
        int a = input[0], b = input[1],  
            expected = input[2];  
        int actual = Calculator.sum(a, b);  
        assertions.add(() ->  
            assertEquals(actual, expected));  
    }  
    assertAll(assertions);  
}
```

# JUnit 5 – Parametrized Tests

Parametrized tests make running a test multiple times with different inputs possible.

A parametrized test method

- Is annotated with `@ParametrizedTest`
- Takes at least one argument
- Is provided with at least one data source

JUnit will run the parametrized method with each of the inputs provided by the data sources.

# JUnit 5 – Parametrized Test Data Sources

Data sources in Junit 5 include:

- `@NullSource`: provides one single null argument. Cannot be used with primitive parameters;
- `@EmptySource`: provides a single empty argument for a single parameter of type String, Collections, or primitive array.
- `@ValueSource`: lets you specify an array of literal values to be used as arguments parameterized test invocation;
- `@MethodSource`: lets you specify a method that provides a `Stream<Arguments>` to be used as inputs.

# JUnit 5 – Parametrized Test Examples

```
@ParameterizedTest
@EmptySource
@ValueSource(strings = {"racecar", "AnNa", "KAYAK", "123454321", "Madam, I'm Adam"})
void shouldReturnTrueForPalindromeStrings(String string) {
    assertTrue(PalindromeChecker.isPalindrome(string));
}

@ParameterizedTest(name = "Test {index} ==> '{0}' is NOT palindrome")
@NullSource
@ValueSource(strings = {"jUnit", "abc", "Martin Fowler", "123ABC", " xy "})
void shouldReturnFalseForNonPalindromeStrings(String string) {
    assertFalse(PalindromeChecker.isPalindrome(string));
}
```

# JUnit 5 – Parametrized Test Examples

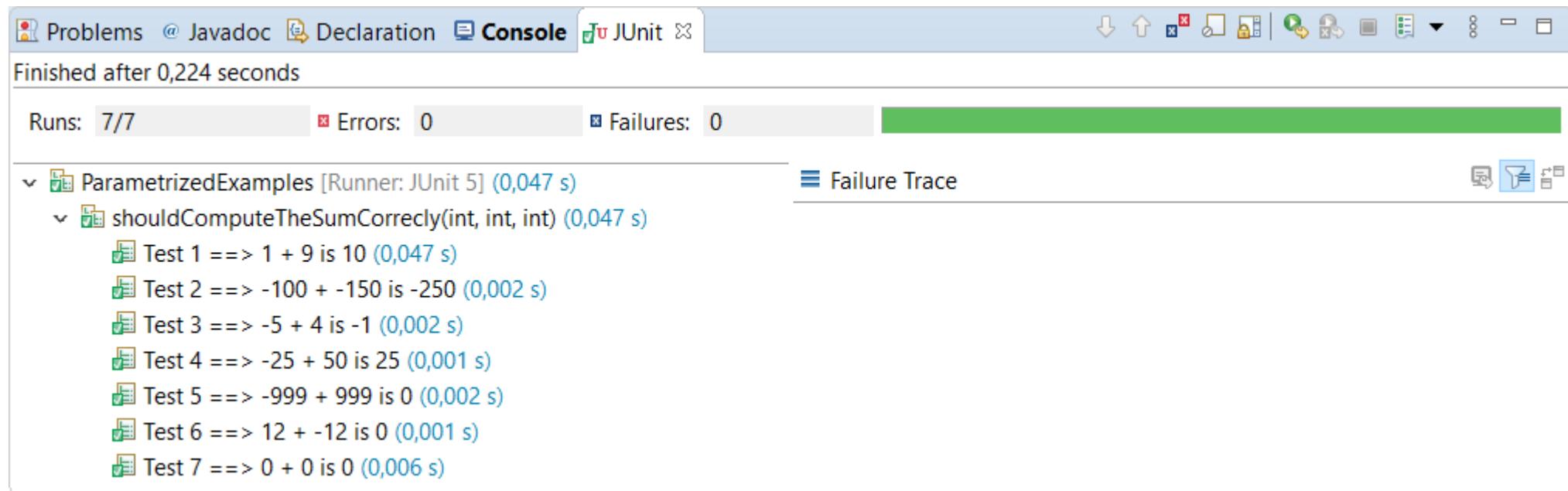
The screenshot shows the Eclipse IDE interface with the JUnit view selected. The status bar at the top indicates "Finished after 0,185 seconds". Below it, summary statistics are displayed: "Runs: 12/12", "Errors: 0", and "Failures: 0". A green progress bar is partially filled. The main content area displays a tree view of test results under "ParametrizedExamples [Runner: JUnit 5] (0,060 s)". The tree includes two test methods: "shouldReturnTrueForPalindromeStrings(String)" and "shouldReturnFalseForNonPalindromeStrings(String)". The first method has six data points: [1] (0,046 s), [2] racecar (0,002 s), [3] AnNa (0,002 s), [4] KAYAK (0,002 s), [5] 123454321 (0,002 s), and [6] Madam, I'm Adam (0,003 s). All data points are marked with a green checkmark icon. The second method has six test cases: Test 1 ==> 'null' is NOT palindrome (0,003 s), Test 2 ==> 'jUnit' is NOT palindrome (0,001 s), Test 3 ==> 'abc' is NOT palindrome (0,001 s), Test 4 ==> 'Martin Fowler' is NOT palindrome (0,002 s), Test 5 ==> '123ABC' is NOT palindrome (0,001 s), and Test 6 ==> ' xy ' is NOT palindrome (0,005 s). All test cases are marked with a green checkmark icon. To the right of the tree view, there is a "Failure Trace" section which is currently empty.

# JUnit 5 – Parametrized Test Examples

```
@ParameterizedTest(name = "Test {index} ==> {0} + {1} is {2}")
@MethodSource("addInputProvider")
void shouldComputeTheSumCorrecly(int a, int b, int expected) {
    int actual = Calculator.sum(a, b);
    assertEquals(expected, actual);
}

static Stream<Arguments> addInputProvider() {
    return Stream.of(
        Arguments.of(1,9,10),
        // more arguments..
        Arguments.of(0,0,0)
    );
}
```

# JUnit 5 – Parametrized Test Examples



# JUnit 5 – Conditional Test Execution

If we don't want a test method to run, we can disable it.  
Disabled test methods will be skipped by the JUnit runner.

A test method can be disabled *unconditionally*

```
@Test  
@Disabled  
void disabledTest() {}  
  
@Test  
@Disabled("Disabled until F23 is implemented")  
void disabledTest2() {}
```

# JUnit 5 – Conditional Test Execution

If we don't want a test method to run, we can disable it.  
Disabled test methods will be skipped by the JUnit runner.

A test method can be disabled *based on conditions*

```
@Test  
@DisabledOnOs(OS.WINDOWS)  
void testDisabledOnWindows() {}  
  
@Test  
@EnabledOnOs({OS.WINDOWS, OS.MAC})  
void testEnabledOnWindowsAndMac() {}
```

# JUnit 5 – Conditional Test Execution

If we don't want a test method to run, we can disable it.  
Disabled test methods will be skipped by the JUnit runner.

A test method can be disabled *based on conditions*

```
@Test  
@EnabledForJreRange(min = JAVA_8)  
void fromJava8toCurrentJavaFeatureNumber() {}  
  
@Test  
@DisabledOnJre(JAVA_9)  
void notOnJava9() {}
```

# JUnit 5 – Assumptions

- Assumptions are utility methods we can use to conditionally execute tests (or parts of tests);
- They're defined as static methods of the Assumptions class;
- If an assumption fails, the current test (or part thereof) is skipped.

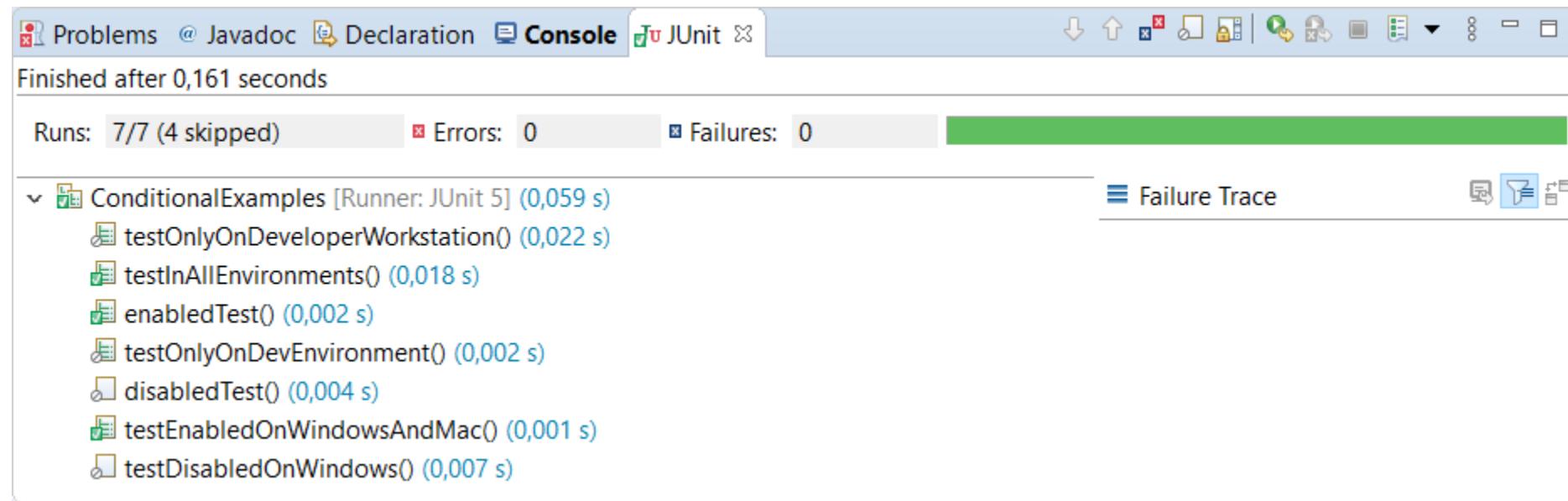
```
@Test
void testOnlyOnDeveloperWorkstation() {
    assumeTrue("DEV".equals(System.getenv("ENV")),
        "Aborting test: not on dev environment");
    assertSomething(/*...*/); // only in dev environment
}
```

# JUnit 5 – Assumptions

- Assumptions are utility methods we can use to conditionally execute tests (or parts of tests);
- They're defined as static methods of the Assumptions class;
- If an assumption fails, the current test (or part thereof) is skipped.

```
@Test
void testInAllEnvironments() {
    assumingThat("PROD".equals(System.getenv("ENV"))), () -> {
        assertSomething(/*...*/); // only in PROD environment
    });
    assertSomething(/*...*/); // always
}
```

# JUnit 5 – Conditional Test Execution



# JUnit 5 – Extension Model

The behaviour of test classes and test methods can be furtherly extended by registering Extensions.

Extensions are related to certain phases in the lifecycle of a test, and their methods are called by the JUnit engine when the corresponding phase is reached.

Common extension points are

- Test Instance Post-processing (e.g. to initialize parameters)
- Parameter resolution (e.g. to add parameter sources)
- Lifecycle callbacks (e.g. run a custom method after each test)

# JUnit 5 – Extension Model

```
@ExtendWith(DatabaseExtension.class)
class AnIntegrationTest {

    @InitDatabase          // custom annotation
    DatabaseDAO database; // automatically initialized

    // test methods..
}
```

# Assertion Frameworks

# Assertion Frameworks

Assertion frameworks allow developers to write better assertions, and thus better tests, without re-inventing the wheel!

- More readable
- More flexible
- More informative error messages

Notable examples include: Hamcrest, Google Truth, AssertJ

**What's wrong with standard JUnit assertions?**

# Let's try our hand with an assertion

Suppose you need to test the `getDivisors()` method:

```
class MathUtils {  
  
    // Returns a list of divisors of number  
    static List<Integer> getDivisors(int number){  
        List<Integer> list = new ArrayList<Integer>();  
        for(int i = 1; i <= number; i++) {  
            if(number % i == 0) {  
                list.add(i);  
            }  
        }  
        return list;  
    }  
}
```

# Let's try our hand with an assertion (v. 1)

```
@Test  
void testDivisorsOfEight() {  
    List<Integer> divisors = MathUtils.getDivisors(8);  
    List<Integer> expected = Arrays.asList(1,2,4,8);  
    assertEquals(expected, divisors);  
}
```

Quite brittle! We're over-specifying

- The test depends on the implementation, not on the public interface
- We don't really need the divisors to be in ascending order
- A small change to the implementation could break the test
  - E.g. adding 1 and number to the list before the loop, since both are sure to be divisors!

# Let's try our hand with an assertion (v. 1)

```
@Test  
void testDivisorsOfEight() {  
    List<Integer> divisors = MathUtils.getDivisors(8);  
    List<Integer> expected = Arrays.asList(1,2,4,8);  
    assertEquals(expected, divisors);  
}
```

Error message might not be very informative!

- With two `List<Integer>` you should get something like:  
`expected: <[1,2,4,8]> but was: <[2,4,8]>`  
That's not too bad, but what if there were 1000 elements?
- With two `List<Object>` you would have gotten something on the lines of `expected: <java.lang.ArrayList@6adbdf5> but was <java.lang.ArrayList@5ajrd98>`  
which is not very helpful!

# Let's try our hand with an assertion (v. 2)

```
@Test  
void testDivisorsOfEight() {  
    List<Integer> divisors = MathUtils.getDivisors(8);  
    List<Integer> expected = Arrays.asList(1,2,4,8);  
    assertTrue(divisors.containsAll(expected));  
}
```

Too loose! We're under-specifying

- The test would pass when `divisors` contains more numbers!

# Let's try our hand with an assertion (v. 3)

```
@Test  
void testDivisorsOfEight() {  
    List<Integer> divisors = MathUtils.getDivisors(8);  
    List<Integer> expected = Arrays.asList(1,2,4,8);  
    assertAll(  
        () -> assertTrue(expected.containsAll(divisors)),  
        () -> assertTrue(divisors.containsAll(expected))  
    );  
}
```

Solved the over/under-specification issues, but..

- Made it a little harder to read (think if we couldn't use `containsAll()`)
- The error message we get is something like: `Multiple Failures (1 failure)  
expected: <true> but was: <false>`  
which is, again, not very helpful!

# What's wrong with JUnit Assertions?

- Can lead to hard-to-read tests
- Can lead to over/under-specification
  - Resist the temptation of `assertEquals()` when it's not what we *really* need
  - Do not settle with loose specifications for the sake of convenience
- Error messages not always immediate to understand
  - There exist ways to achieve better error messages in standard JUnit assertions (e.g. the `Supplier<String>` parameter), but why should we re-invent the wheel?

# HAMCREST

# MATCHERS

# Getting to know Hamcrest

Hamcrest allows developers to declaratively define

**Matchers that can be combined to create flexible expressions of intent.**

- Not only for testing (UI Validation, data filtering, ...)
- Matcher objects represent conditions, and they can be combined declaratively to create natural-language like (fluent) assertions

# Testing with Hamcrest

The entry point for Hamcrest in our tests is the method

```
<T> void assertThat(T actual, Matcher<? super T> matcher)
```

# Testing with Hamcrest

The entry point for Hamcrest in our tests is the method

```
<T> void assertThat(T actual, Matcher<? super T> matcher)
```

```
// these express the same assertion
assertEquals(expected, actual);
assertThat(actual, equalTo(expected));
assertThat(actual, is(expected));
assertThat(actual, is(equalTo(expected)));

// and so do these                                // and these
assertTrue(condition);                          assertNotNull(actual);
assertThat(condition, is(true));                assertThat(actual, is(not(nullValue())));
```

# Hamcrest – Core Matchers

- `equalTo(o)` given an Object, returns a Matcher that matches with any object which is equal to o (according to `o.equals()`);
- `not(m)` returns a Matcher that complements the given Matcher m;
- `is(k)` if given an object, returns `equalTo(k)`. If given a Matcher, returns the matcher without changing its behaviour;

```
assertThat("foo", is(not(equalTo("bar"))));
```

# Hamcrest – String Matchers

Take a String `s` as input and return a suitable Matcher

- `containsString(s)` and `containsStringIgnoringCase(s)`
- `startsWith(s)` and `startsWithIgnoringCase(s)`
- `endsWith(s)` and `endsWithIgnoringCase(s)`

```
assertThat("Hamcrest", startsWith("Ham")); // passes
assertThat("Hamcrest", startsWith("ham")); // fails
assertThat("Hamcrest", containsStringIgnoringCase("CRE")); //passes
assertThat("Hamcrest", endsWithIgnoringCase("REST")); //passes
```

# Hamcrest – Comparable Matchers

Take a Comparable `c` as input and return a suitable Matcher

- `greaterThan(c)` and `greaterThanOrEqualTo(c)`
- `lessThan(c)` and `lessThanOrEqualTo(c)`
- `closeTo(c,delta)` with delta being the admissible error

```
assertThat(10, is(greaterThanOrEqualTo(10)));
assertThat(10, is(LessThan(100)));
double a = 1.001, b = 1.002, delta = 0.002;
assertThat(a, is(closeTo(b, delta)));
```

# Hamcrest – Logical Matchers

- `allOf(m1, ...)` returns a matcher that matches the examined object only if it matches all the given matchers.
- `anyOf(m1, ...)` returns a matcher that matches the examined object only if it matches one of the given matchers.
- `both(m1).and(m2)` and `either(m1).or(m2)` build matchers that match only if both (resp. either) `m1` and (resp. or) `m2` match.

```
assertThat("Hamcrest", allOf(startsWith("Ham"), endsWith("crest")));
assertThat(123, anyOf(isLessThan(50)), isGreaterThan(120)));
assertThat(123, either(isLessThan(50)).or(isGreaterThan(130))));
```

# Hamcrest – Object Matchers

- hasProperty(s) matches with all objects with an «s» property
- hasProperty(s, m) matches with all objects whose «s» field matches the given matcher

```
Car car = new Car("Stelvio", 280, "Gasoline"); //model, hp, fuel
assertThat(car, hasProperty("fuel"));
assertThat(car, hasProperty("hp", is(greaterThan(200))));
```

# Hamcrest – Collection Matchers: Iterables

- `hasItem(m)` matches with Iterables that contain an item matching with `m`
- `contains(k1,...)` matches only if each `n`-th element of the examined Iterable matches with the corresponding `n`-th matcher/object
- `containsInAnyOrder(k1,...)` matches only if each element of the examined Iterator matches with exactly one of the matchers/objects (same length required).

```
List<String> s = Arrays.asList("Anna", "Bob", "Carl");
assertThat(s, hasItem(both(startsWith("A")).and(endsWith("a"))));
assertThat(s, contains(startsWith("A"), startsWith("B"), startsWith("C")));
assertThat(s, containsInAnyOrder("Bob", "Anna", "Carl"));
```

# Hamcrest – Collection Matchers: Maps

- `hasKey(m)` matches with Maps that contain a key matching with m
- `hasValue(m)` matches with Maps containing a value matching with m
- `hasEntry(m1,m2)` matches with Maps that contain a key matching with m1 whose value matches with m2.

```
Map<String, Integer> map = new HashMap<String, Integer>();  
map.put("Anna", 1); map.put("Bob", 3); map.put("Carl", 3);  
assertThat(map, hasKey(containsString("nn")));  
assertThat(map, hasValue(is(greaterThan(2))));  
assertThat(map, hasEntry(startsWith("A"), is(LessThan(2))));
```

# Hamcrest – There's much more!

- Check out [the docs](#) if you want to know more!
- For a starter, you can take a look at the [Matchers class](#).

# Let's get back at our example

```
@Test
void testDivisors() {
    List<Integer> divisors = MathUtils.getDivisors(8);
    List<Integer> expected = Arrays.asList(1,2,4,8);
    assertAll( // v3
        () -> assertTrue(expected.containsAll(divisors)),
        () -> assertTrue(divisors.containsAll(expected))
    );
    // with Hamcrest
    assertThat(divisors, containsInAnyOrder(1,2,4,8));
}
```

And here's what an error message looks like with Hamcrest:

```
Expected: iterable with items [<1>, <2>, <4>, <8>] in any order
but: no item matches: <1> in [<2>, <4>, <8>]
```

# Another example

We need to check that in a `List<Car>` there is at least one Car such that its fuel is "Electric" or "Hydrogen".

# Another example – The classic way

We need to check that in a `List<Car>` there is at least one Car such that its fuel is "Electric" or "Hydrogen".

```
boolean found = false;
for(Car c: Car.getCars()) {
    if(c.getFuel().equals("Electric") || c.getFuel().equals("Hydrogen")) {
        found = true;
        break;
    }
}
assertTrue(found);

> Error message:
Expected: <true> but was: <false>
```

# Another example – The Hamcrest way

We need to check that in a `List<Car>` there is at least one Car such that its fuel is "Electric" or "Hydrogen".

```
assertThat(Car.getCars(), hasItem(  
    hasProperty("fuel", either(is("Electric")).or(is("Hydrogen"))))  
));
```

> Error message:

```
Expected: a collection containing  
    hasProperty("fuel", (is "Electric" or is "Hydrogen"))  
but: mismatches were: [ property 'fuel' was "Hidrogen",  
    property 'fuel' was "Gas", property 'fuel' was "Gas"]
```

# Writing our own Matcher

Suppose we need to check that a `List<Student>` contains at least one `Student` whose name is not a palindrome.

Yes, we *really* do need that! 😳

```
assertThat(list, hasItem(hasProperty("firstName", is(not(palindrome())))));
```

Hamcrest features a lot of built-in Matchers, but none to match palindrome strings, unfortunately.

Not too bad, we'll write it ourselves!

# Writing our own Matcher

To create a new matcher, one needs to implement the Matcher interface:

```
public interface Matcher<T> {  
    boolean matches(Object actual);  
    void describeMismatch(Object actual, Description mismatchDescription);  
}
```

Often, anyway, it is more practical to extend one of the abstract classes BaseMatcher<T> or TypeSafeMatcher<T>, which implement Matcher<T>.

# Writing our own Matcher

```
public class IsPalindrome extends TypeSafeMatcher<String>{  
  
    @Override  
    public void describeTo(Description description) {  
        description.appendText("a palindrome string");  
    }  
  
    @Override  
    protected boolean matchesSafely(String item) {  
        StringBuilder sb = new StringBuilder(item).reverse();  
        return sb.toString().equalsIgnoreCase(item);  
    }  
  
    public static IsPalindrome palindrome() {  
        return new IsPalindrome();  
    }  
}
```

Generates a description of the object we match

matchesSafely is called by TypeSafeMatcher.matches()

Static method to access our matcher, so we can use it as the built-in ones

# Using our own matcher

```
import static org.hamcrest.Matchers.*;  
import static it.unina.computerscience.softeng.junitdemo.examples.IsPalindrome.*;  
  
// ...  
  
assertThat(list, hasItem(hasProperty("firstName", is(not(palindrome())))));  
  
> Error message:  
Expected: a collection containing  
    hasProperty("firstName", is not a palindrome string)  
but: mismatches were: [ property 'firstName' was "Anna",  
    property 'firstName' was "Bob", property 'firstName' was "Otto"]
```

# Other Java Assertion frameworks

Other well-known Java assertion frameworks include:

- AssertJ ([web](#))
- Google Truth ([web](#))

They're less general than Hamcrest, and are based on method chaining.

```
// AssertJ
assertThat(gamma.getName()).startsWith("Er").endsWith("ich");
assertThat(theGangOfFour).contains(gamma, helm).doesNotContain(fowler);

// Truth
assertThat(theGangOfFour).containsExactly(gamma, helm, vliss, johnson).inOrder();
```

# AssertJ / Truth vs Hamcrest

## AssertJ / Truth

- Based on method chaining
  - No LISP-like parentheses hell
  - IDE auto-complete
- Generally can provide better error messages

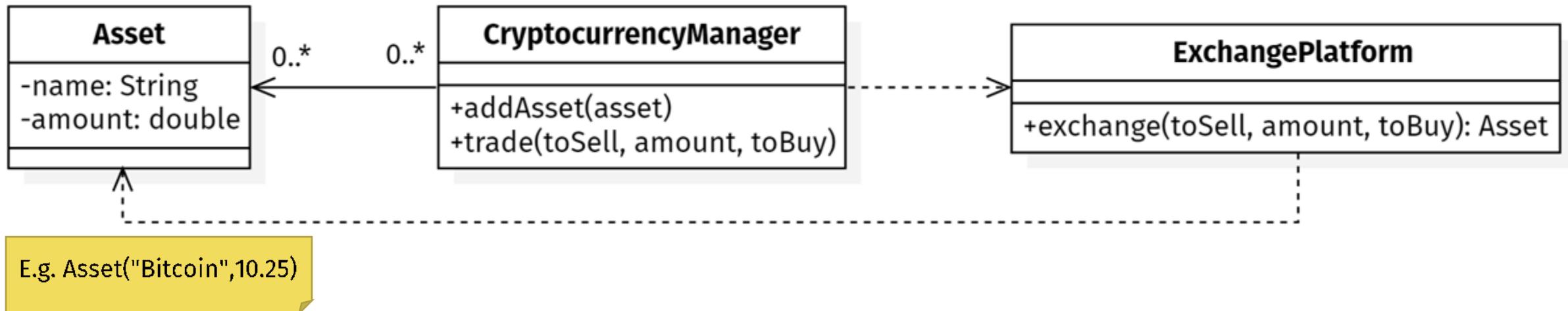
## Hamcrest

- More general
  - Applications outside of testing
  - Can be used to set expectations when mocking

# Testing in Isolation

# Dependencies and Testing

A class may depend on other helper classes (e.g. DAOs, APIs)



Suppose we need to test the `trade()` method.

# Dependencies and Testing

```
@Test  
void testTrading() {  
    // arrange  
    CryptocurrencyManager c = new CryptocurrencyManager();  
    c.addAsset(new Asset("BTC",10.0));  
    c.addAsset(new Asset("EUR",10000.0));  
    // act (1 BTC = 7000.0 EUR)  
    c.trade("EUR", 7000.0, "BTC"); ←  
    // assert  
    assertTrue(c.getAsset("EUR").get().getAmount() == 3000.0);  
    assertTrue(c.getAsset("BTC").get().getAmount() == 11.0);  
}
```

We're using the real ExchangeAPI, trading **real** Euros!

The exchange rate might change tomorrow!

See anything wrong with this?

# Dependencies and Testing

Using real production objects as helpers in our unit tests has a few drawbacks and may not always be possible:

- When a test fails, we don't know if the bug is in the unit we're testing or in its dependencies;
- The introduction of a bug in a highly-used helper object can cause a ripple of failing tests all across the system;
- Sometimes, we cannot afford to use real production objects! Think of a `BankTransactionsDAO`, or a `MissileLauncher`!

How can we alleviate the above issues and still test our units?

# Test Doubles

Test Doubles are replacements for production objects that are typically used in tests.

Test Doubles can be classified as follows:

- **Fakes:** objects that have working implementations, but are not suitable for production because of limitations;
- **Stubs:** replace a real component and return pre-defined answers;
- **Mocks:** more advanced, configurable test stubs that can also record the indirect outputs of the unit under test.

# (Lack of) Inversion of Control

Suppose our CryptocurrencyManager was like this:

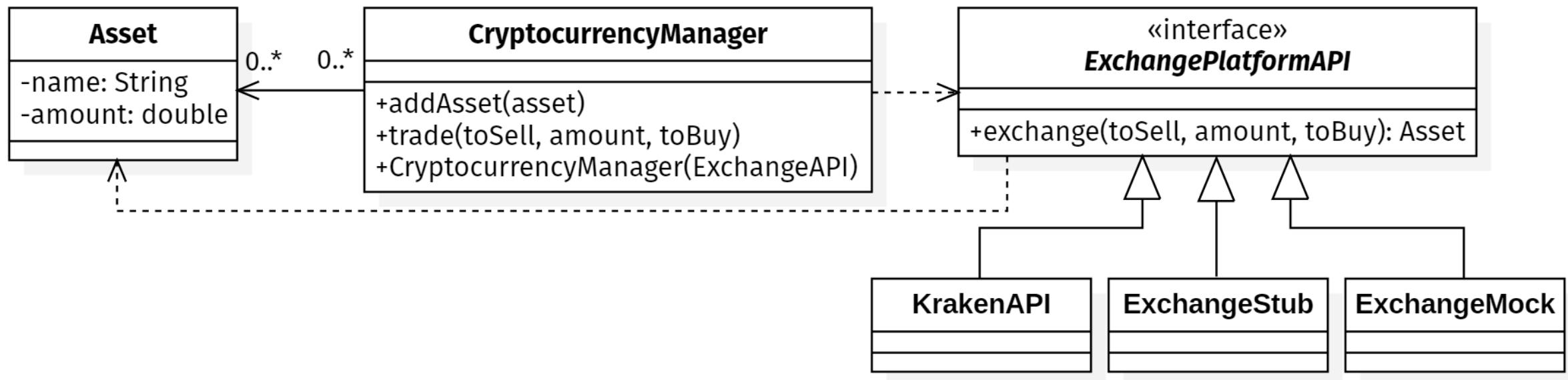
```
public class CryptocurrencyManager {  
    private ExchangePlatform exchangeAPI = new ExchangePlatform();  
    CryptocurrencyManager() {/*...*/}  
    public void trade(String toSell, double amountToSell, String toBuy) {/*...*/}  
}
```

That's really BAD design!

- What if we want to support more than one exchange platform?
- There's no way we can use a test double in place of the real deal!

# Inversion of Control

- Client classes should be able to provide dependencies
  - Through constructors, public setters, builders, method parameters...
- Inversion of control is fundamental to make units testable



# Dependencies and Testing

```
@Test  
void testTrading() {  
    // arrange  
    CryptocurrencyManager c = new CryptocurrencyManager(new ExchangeStub());  
    c.addAsset(new Asset("BTC",10.0));  
    c.addAsset(new Asset("EUR",10000.0));  
    // act (1 BTC = 7000.0 EUR)  
    c.trade("EUR", 7000.0, "BTC");  
    // assert  
    assertTrue(c.getAsset("EUR").get().getAmount() == 3000.0);  
    assertTrue(c.getAsset("BTC").get().getAmount() == 11.0);  
}
```



We're not trading  
**real** Euros  
anymore!



The exchange rate  
won't change if the  
stub doesn't!

# Our very simple Exchange Platform Stub

```
class ExchangeStub implements ExchangeAPI {  
    @Override  
    public Asset exchange(String toSell, double amount, String toBuy) {  
        return new Asset("BTC",1.0);  
    }  
}
```

Don't be fooled by this tiny example! Writing mocks and stubs can be very tedious!

Think of multiple methods, with many conditions each!

- E.g.: when amount < 1000, then return X, when >2000 ...
- And you might not be able to re-use it in the next test!

# Mocking frameworks

Writing stubs and mocks is boring and takes time.

Mocking frameworks allow us to easily create mocks and stubs to use in our tests.

Popular frameworks include: [Mockito](#), [EasyMock](#), [JMockit](#).

# Introducing Mockito

Mockito is a well-known mocking framework.  
Top 10 Java library across all libraries<sup>1</sup>.

Plus, it tastes really good!

[1] <https://blog.overops.com/githubs-10000-most-popular-java-projects-here-are-the-top-libraries-they-use/>

# Mockito – Creating Mocks

Mocks can be created with the static method

```
<T> T mock(Class<T> classToMock)
```

By default, each method of the mock will return the default value for the corresponding type.

```
List mockedList = mock(ArrayList.class); // with classes
Map mockedMap    = mock(Map.class);      // and with interfaces too

mockedMap.get("Foo"); // returns null
mockedList.get(2);   // returns null
mockedList.isEmpty(); // returns false
mockedList.size();   // returns 0
```

# Mockito – Configuring Mocks

We can configure the behaviour of our mock as follows:

```
List mockedList = mock(ArrayList.class); // with classes
Map mockedMap = mock(Map.class);           // and with interfaces too

// configuration
when(mockedMap.get("Foo")).thenReturn("Bar");
doReturn("Bar").when(mockedList).get(2); // alternative way
when(mockedList.size()).thenReturn(99);

mockedMap.get("Foo"); // returns "Bar"
mockedList.get(2);   // returns "Bar"
mockedList.isEmpty(); // returns false as before
mockedList.size();   // returns 99
```

# Mockito – Argument matchers

Often, we won't need to be so specific about input arguments.  
In fact, being more generic could help us write less configuration code and more flexible tests.

```
// configuration: order matters!
when(mockedList.get(anyInt())).thenReturn("Mockito!");
when(mockedList.get(lt(0))).thenThrow(IllegalArgumentException.class);
when(mockedList.get(-25)).thenReturn("MOCKITO!");

mockedList.get(1988); // returns "Mockito!"
mockedList.get(-25); // returns "MOCKITO!"
mockedList.get(-2); // throws IllegalArgumentException
```

# Mockito – Configuring Mocks

When we invoke a method on mock with a given list of parameters, Mockito tries to find the latest configuration rule that «matches», and applies that.

If no configuration rule is found, it defaults to default values.

Therefore, configuration order matters, and more general rules should be declared before the more specific ones.

# Mockito – Argument matchers

Mockito features a lot of built-in argument matchers, that can often also be combined in a Hamcrest-like fashion.

E.g.: `any()`, `anyString()`, `anyCollection()`, `leq()`, `isNull()`, ...

We won't examine them in detail today, but if you're interested you can check out the [ArgumentMatchers](#) and the [AdditionalMatchers](#) classes from the docs.

It's also possible to use Hamcrest matchers as Mockito argument matchers with the [MockitoHamcrest](#) class!

# Mockito – Arg. matchers with Hamcrest

```
import static org.mockito.Mockito.*;
import static org.mockito.hamcrest.MockitoHamcrest.argThat;
import static org.hamcrest.Matchers.*;
import static it.unina.computerscience.softeng.junitdemo.examples.IsPalindrome.*;

// configuration - argThat is a static method from the MockitoHamcrest class
when(mockedList.add(argThat(
    is(palindrome()) // Hamcrest Matcher
))).thenReturn(true);

mockedList.add("Abe"); // returns false
mockedList.add("Bob"); // returns true
mockedList.add("Otto"); // returns true
```

# Mockito – Configuring Answers

When mocking with Mockito, you're not limited to pre-determined return values.

- With the generic interface [Answer](#), you can customize the behaviour of your mocks by specifying an action to be executed to compute the desired output.

# Mockito – Configuring Answers

```
List<Integer> myMock = mock(ArrayList.class);

when(myMock.get(anyInt())).thenAnswer(
    new Answer() {
        public Object answer(InvocationOnMock invocation) {
            return ((Integer)invocation.getArgument(0)) + 1;
        }
    });
}

//same as the above
when(myMock.get(anyInt())).thenAnswer(
    invocation -> ((Integer)invocation.getArgument(0)) + 1
);

myMock.get(42); // returns 43
myMock.get(99); // returns 100
```

# Mockito – Verifying behaviour

Usually, in the assert phase of our tests, we focus on checking that the final state of the Class Under Test is the one we expect. This is testing by side-effects, and often it is not enough!

```
// assert
assertTrue(c.getAsset("EUR").get().getAmount() == 3000.0);
assertTrue(c.getAsset("BTC").get().getAmount() == 11.0);
// hopefully, a call was made to ExchangeAPI.exchange() with the right params!
```

# Mockito – Verifying behaviour

After you're done with your mock, you can further inspect it to check if and how it's been used!

Out-of-the-box, with no configuration required, a Mockito mock keeps track of every method call it receives.

You can verify a mock's past behaviour with the `verify()` method and its variants.

# Mockito – Verifying behaviour: examples

```
// check that myMock.size() was called exactly once
verify(myMock).size();
// check that myMock.get(19) was called exactly once
verify(myMock).get(19);
// check that myMock.get was called exactly three times with any int arg.
verify(myMock, times(3)).get(anyInt()); // with Mockito matcher
// check that myMock.get was called no more than three times with any even arg.
verify(myMock, atMost(3)).get(argThat(is(even()))); // with Hamcrest matcher
// check that myMock.get(0) was called at least two time
verify(myMock, atLeast(2)).get(0);
// check that no interaction occurred with mockedStack
verifyNoInteractions(mockedStack);
// check that myMock.addAll was never called
verify(myMock, times(0)).addAll(anyCollection());
```

# Mockito – Verifying behaviour: examples

```
// check that method calls happened in a precise order
InOrder inOrder = inOrder(myMock);
inOrder.verify(myMock).get(19);
inOrder.verify(myMock).get(42);
inOrder.verify(myMock).get(99);

// check that no interaction happened after myMock.get(99)
verifyNoMoreInteractions(myMock);
```

# Mockito – Verifying behaviour: errors

Here's what a verify( ) error message looks like:

> Error message:

```
org.mockito.exceptionsverification.NoInteractionsWanted:  
No interactions wanted here:  
-> at CryptocurrencyManagerTest.java:153  
But found this interaction on mock 'arrayList':  
-> CryptocurrencyManagerTest.java:138
```

# Getting back at our example

```
@Test
void testTrading() {
    // create and configure mock
    ExchangePlatformAPI api = mock(ExchangePlatformAPI.class);
    when(api.exchange("EUR", 7000.0, "BTC")).thenReturn(new Asset("BTC",1.0));
    // arrange
    CryptocurrencyManager c = new CryptocurrencyManager(api);
    c.addAsset(new Asset("BTC",10.0)); c.addAsset(new Asset("EUR",10000.0));
    // act
    c.trade("EUR", 7000.0, "BTC");
    //assert
    assertTrue(c.getAsset("EUR").get().getAmount() == 3000.0);
    assertTrue(c.getAsset("BTC").get().getAmount() == 11.0);
    verify(api).exchange("EUR",7000.0,"BTC"); // API were actually called
}
```

# Mockito and JUnit 5

There's an official [mockito-junit-jupiter](#) extension for JUnit 5.

```
@ExtendWith(MockitoExtension.class) // register the extension
class CryptocurrencyManagerTest {

    @Mock ExchangePlatformAPI api; // parameter initialization

    @Test
    void testTrading() {/*...*/}

    @Test
    void testWithMockParam(@Mock List<Integer> list) {/*...*/} //param. resolution

}
```

# Mockito – Limitations

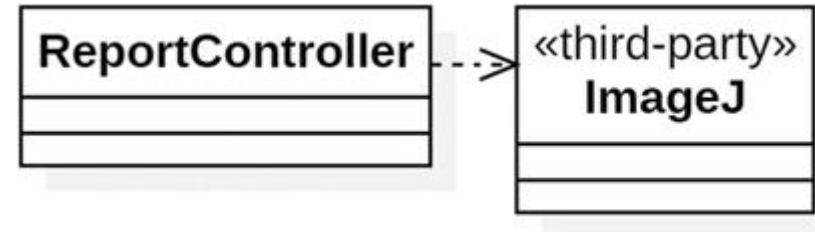
With Mockito you cannot mock:

- constructors;
- static methods;
- equals(), hashCode();

If you need to, you can either consider refactoring, or checkout more "invasive" tools like [PowerMock](#), which uses custom classloaders and bytecode manipulation.

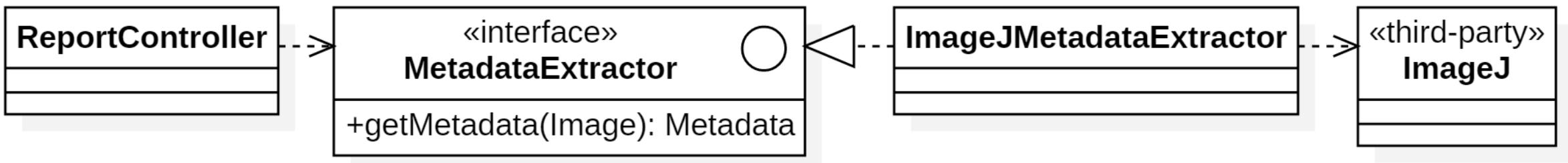
# Rules to mock by

1. Mock behaviour, not data!
  - Just instantiate your POJOs, use builders if you need to.
2. Do not mock types you don't own (e.g. third-party libraries)
  - It might cover up bugs!
  - You should put up an adapter layer between your units and 3<sup>rd</sup>-party libraries, and mock the adapter



# Rules to mock by

1. Mock behaviour, not data!
  - Just instantiate your POJOs, use builders if you need to.
2. Do not mock types you don't own (e.g. third-party libraries)
  - It might cover up bugs!
  - You should put up an adapter layer between your units and 3<sup>rd</sup>-party libraries, and mock the adapter



# References

- <https://martinfowler.com/bliki/UnitTest.html>
- <https://martinfowler.com/bliki/TestDouble.html>
- <http://xunitpatterns.com/Test%20Double.html>
- <https://martinfowler.com/articles/mocksArentStubs.html>
- <https://junit.org/junit5/docs/current/user-guide/>
- <http://hamcrest.org/JavaHamcrest/>
- <https://site.mockito.org/>