Lesson 1.1: Technologies for virtualization

Claudio Ardagna – Università degli Studi di Milano

Cloud and Distributed Computing

Introduction

- Virtual comes from the Latin word virtus
  - It means "virtue", "capability", "potential"
- Different meanings
  - Something that could happen
  - In computer science it is used to indicate what is not real
- Simulated, emulated and virtual are NOT synonyms

Definition

- Virtualization
  - Activity aiming to create replacements (virtual resources) for real resources, that have the same functionalities and external interfaces of their counterpart, but different attributes (dimension, performance, cost)
  - A mechanism through which virtual version of resources, usually provided physically, are being created
  - A technique used to recreate - through software - an environment that looks like a hardware to the host operating system

Emulation vs Simulation vs Virtualization

- They are related word but NOT the same thing

- Emulation: we execute a system like it is another system
  - It means executing OS, API, functions on a machine which they have not been developed for
  - System A gets inputs from System B, System A produces outputs of System B

Emulation vs Simulation vs Virtualization

- Why do we use emulation?
  - Executing an OS on a not-compatible hardware platform (e.g., Microsoft OS on Mac hardware platform)
  - Executing an application on a not-compatible device (e.g., Windows application on Mac, arcade game systems)
  - Reading data written on a memorization device through a device that we no longer have or that no longer works

Emulation vs Simulation vs Virtualization

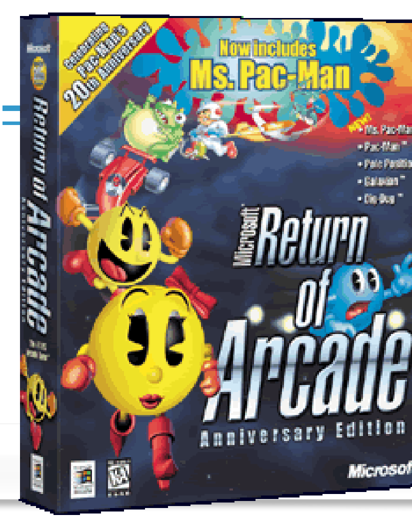
- Emulation (hardware): a computer implemented for executing programs defined for another architecture

- Emulator creates a dump of the software and just emulates the hardware
  - Aiming to replicate the system functioning

Emulation vs Simulation vs Virtualization

- Simulation
  - An application allowing to execute old programs, defined for different platforms, on modern machines
  - Replicates the behavior of a system
  - It’s like a software emulation
  - Has the same goal of an emulator but rewriting the routines of the program to simulate
  - For example «Microsoft Return of Arcade» produced by Microsoft for PC in the second half of the 90s

- Simulation vs emulation
  - Simulation is fast but less precise
  - Emulation is precise but slow



Emulation vs Simulation vs Virtualization

- High level emulation
  - An intermediate between emulation and simulation
  - They recreate the functionalities of an emulated system using similar or equivalent functions in the emulating system (host)
  - Execution time faster than hardware emulation, but less accuracy
  - Nintendo 64 UltraHLE translates CPU functions and graphic system in equivalent functions of host machine CPUs and graphic cards

Emulation vs Simulation vs Virtualization

- Virtualization: the technique for using resources and devices in a functional way, without considering their physical layout

- Including the division of a single physical computer in many virtual machines with a dedicated hardware
  - Virtual machine is software container with software-based CPU, RAM, hard disk and network connection
  - Transparent: an OS or an application do not distinguish between a virtual and a physical machine

Emulation vs Simulation vs Virtualization: Summary

	Main Characteristics
Emulation	Emulates the behavior of the real system, executes unmodified code, accurate and flexible, expensive
Simulation	Approximates the behavior of the real system, requires rewriting software, cheap and flexible, accuracy decrease
Virtualization	Virtualizes the exact behavior of the real system, cross-platform, accurate, flexible, expensive

Virtualization

- Compatible with Intel x86 machines
- Each machine has a full and dedicated environment (encapsulation)
- Each machine is isolated from each other just like physical separation (isolation)
- Independent from underlying hardware (hardware independence)
- Created using existing hardware (partitioning)

A little bit of history

- IBM was the first to develop virtualization on mainframes to execute processes and applications in a concurrent way
  - IBM S/360 Model 67: the first virtualized system (1964)

- A fundamental paper in the sector (Goldberg and Popek) is from 1974
  - Formal requirements for virtualizable third generation architectures

- 1980-1990: client-server applications and distributed computing limit the application of the virtualization

- In the last 20 years the underutilization problem of the 60s comes back
  - Many physical servers, high costs, fault and obsolescence issues
  - In 1998 VMware is born, the first virtualization solution for x86 systems

A little bit of history

- There exist many types of virtualization
  - The Java virtual machine executing Java code
  - Volumes exported from a SAN (Storage Area Network)
  - System resources from the programs point of view
  - The most complex network topologies

- Virtualization strategies evolve from the idea of executing a system on another one, to tools to maximize resources usage, to mechanisms to realize models for offering IT resources as services (next lesson)
  - IDC in 2015 shows how client virtualization has become mature
  - Gartner in May 2016 states that server virtualization has reached its pike, more than 75% of x86 server workload is virtualized

A little bit of history

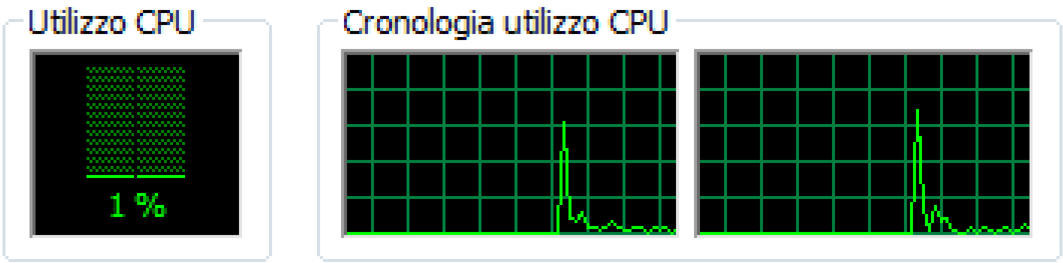
- One of the most developed sector in the IT world
  - Virtualization market
    - Data center 8.06 bln $ by 2022 (marketsandmarkets), 3.75 bln $ in 2017
    - Desktop 13.45 bln $ by 2022 (marketsandmarkets), 7.83 bln $ in 2017
  - Market of 160 bln $ for Cloud computing by 2022, 130 bln$ in 2017
    - https://www.statista.com/statistics/510350/worldwide-public-cloud-computing/
  - All the big players have a strategy (IBM, Red Hat, SuSE, Microsoft, Apple, ...)
    - Virtualization adoption is 76% in 2016
  - AMD and Intel have CPUs with support for virtualization

Reason for virtualization: issues

- Too many servers, small workloads (<40% usage)
- Old hardware does not work
- Infrastructural requirements are always increasing (many independent servers)
- Small flexibility in shared environments

Reason for virtualization: issues

- Underutilization of hardware



- High costs and needs
  - Maintenance, Leases, Networking, Floor space, Cooling, Power, Disaster Recovery
- Heterogeneous environments
  - Linux, Microsoft, IBM, Apple, SUN

Reason for virtualization: pros

- Consolidation and hardware costs decreasing
  - It's possible to use virtualization to access resources and manage them efficiently for reducing operation and management costs, while keeping the necessary computational power
  - It's possible to use virtualization to let a single server work like more virtual servers
- Workload optimization
  - Virtualization enables dynamically answering to applications' needs
  - It's possible to use virtualization to increase resource usage, enabling dynamic sharing of resource pools

Reason for virtualization: pros

- Flexibility and IT responsiveness
  - Virtualization allows having a single consolidated view of each resource of the network, which is easy to access and location-independent
  - Virtualization allows reducing the management of the environment, providing emulation to support compatibility and increased interoperability
- Multiple execution environments
  - Chosen by the user basing on his needs
- Simplified management
  - Single vision of all resources
  - Centralized control of the environment

Reason for virtualization: other pros

- Better performance
- Transparency
- Heterogeneity
- Portability
- Interoperability
- Green IT

Reason for virtualization: scenarios

- Server consolidation: many network services offered by distinct servers are migrated into a single server
- Testing: it is possible to duplicate a test server into a production one (or vice versa)
- Training: it is possible to provide a complete study environment, which is hardware-independent

Virtualization management

- Analysis and planning
- Adaptation and post-adaptation period
- Virtualized infrastructure maintenance

Analysis and planning

- Compatibility and support of existing hardware
- License analysis
  - Some software restrict the number of instances
  - Other ones (e.g., Windows Server 2003 Datacenter Edition) do not have restrictions
- Migration and deployment planning
- Staff Training
- ROI evaluation

Adaptation and post-adaptation

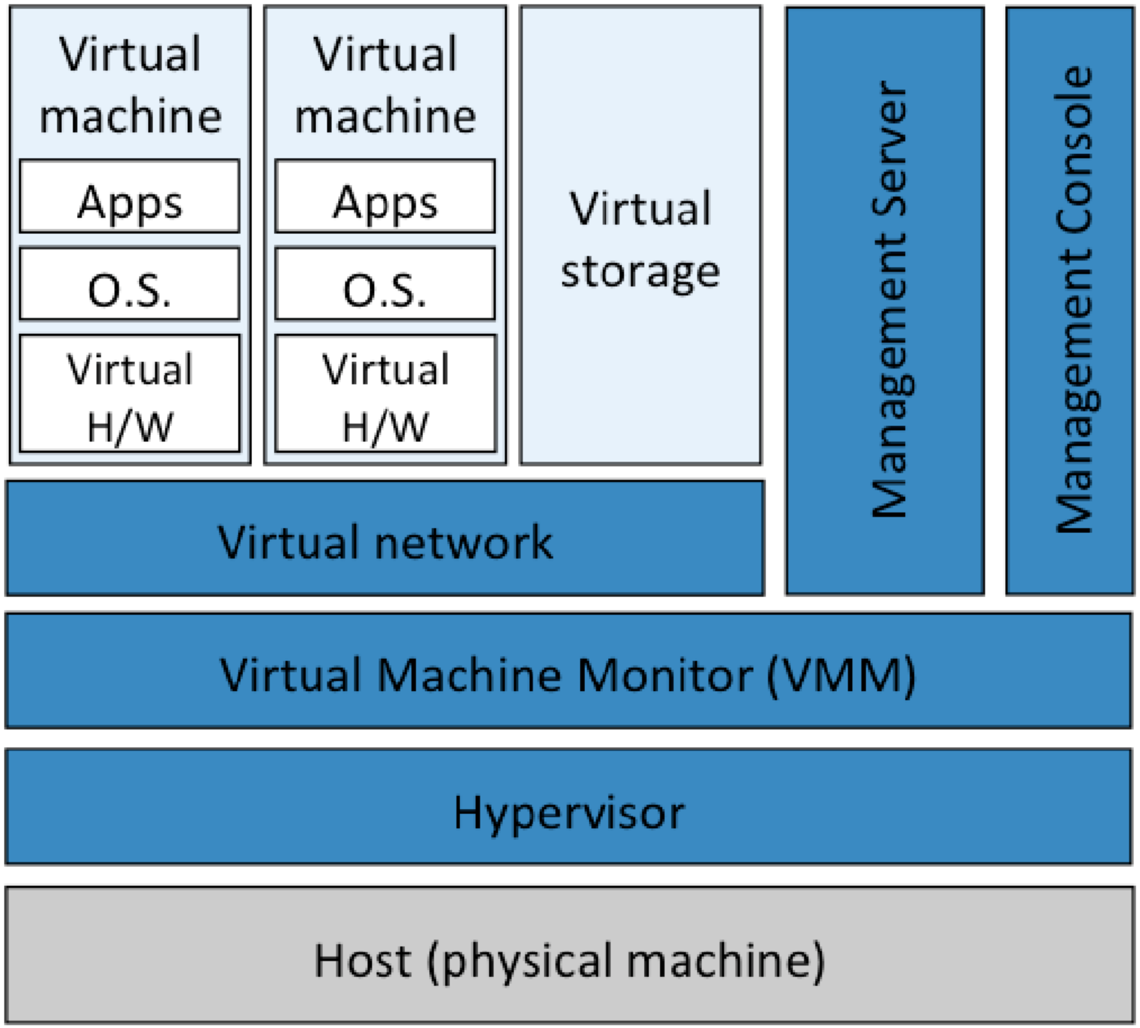
- Moving to a virtual machine/network is not hassle free

- Need to evaluate
  - Reliability: a single physical machine introduces the need of disaster recovery solutions
  - Performance in a real industrial environment
  - Efficiency of the implemented solution
  - Security: multitenancy requires control of encapsulation and message security

Maintenance

- Scalability: hardware does not grow with infrastructure growing
- Security of virtual machine and virtualization platform
- Responsibility: roles and privileges
- Evaluation of the virtualization market

Virtualization components



Terminology

- Hypervisor: the system component realizing virtualization
  - Allows multiple OS running on the same computer
  - Mediates between virtual machine and physical devices
  - Mediates hardware requests down to the physical level
  - Implements the virtual machine monitor providing virtualized hardware to virtual machines
- Virtual Machine Monitor (VMM): the application component realizing virtualization
  - A part of the hypervisor
  - Keeps track of what happens inside virtual machines
  - Redirects to physical resources
  - Supports resource sharing between users
  - Guarantees virtualization transparency to users

Terminology

- Guest OS: the virtualized system
  - Virtual machine synonym
  - Encapsulated system made up of OS and applications
  - Uses the hardware abstraction provided by VMM

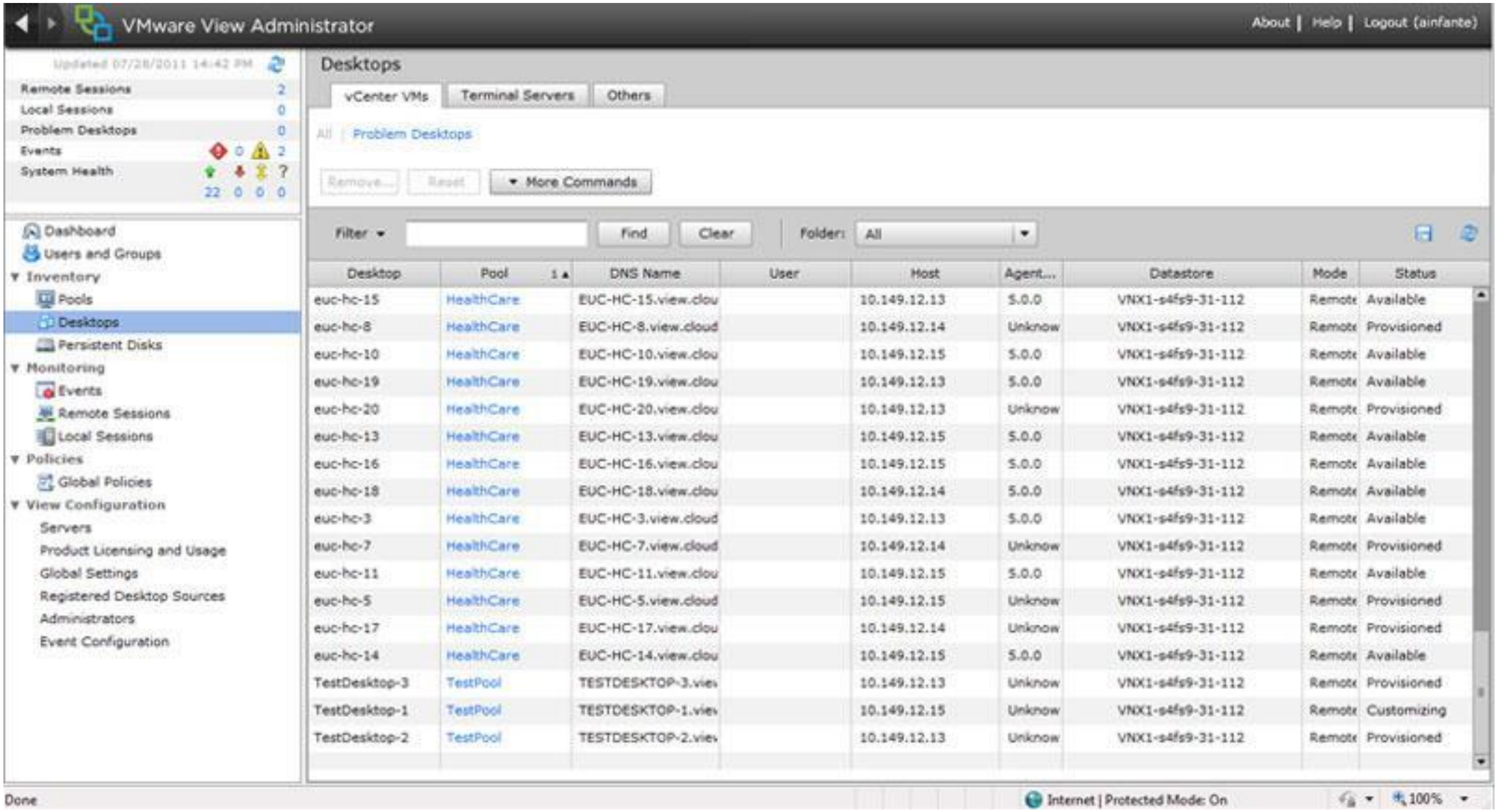
- Host OS: the real system
  - Physical machine (and OS) hosting the virtualized system
  - Manages physical hardware
  - Can install the hypervisor

Terminology

- Management Server: virtualization platform
  - Made up of set of components for managing virtual machines, consolidating servers, allocating resources, migrations, high availability

- Management Console: provides access to the virtualization management interface
  - Allows adding, updating, deleting, configuring virtual machines
  - Standalone client standalone or web interface

Terminology



Terminology

- Network components: enables developing virtual networks
  - Network devices (switch, router...) completely controlled through software
  - Simulated protocols and network stack in order to replicate physical ones

- Virtualized storage: provides abstract components of physical storage
  - Accessed through the network or with direct connection
  - Data are logically partitioned (they belong to the same storage)

Virtualization means…

- The system executed in the virtual environment must behave exactly as it was executed on an equivalent physical machine
- The virtualization environment must have full over the virtual resources
- A statistically relevant percentage of instructions must be executed without involving virtualization
- This last property is not mandatory, yet it guarantees the efficiency of the virtual machine

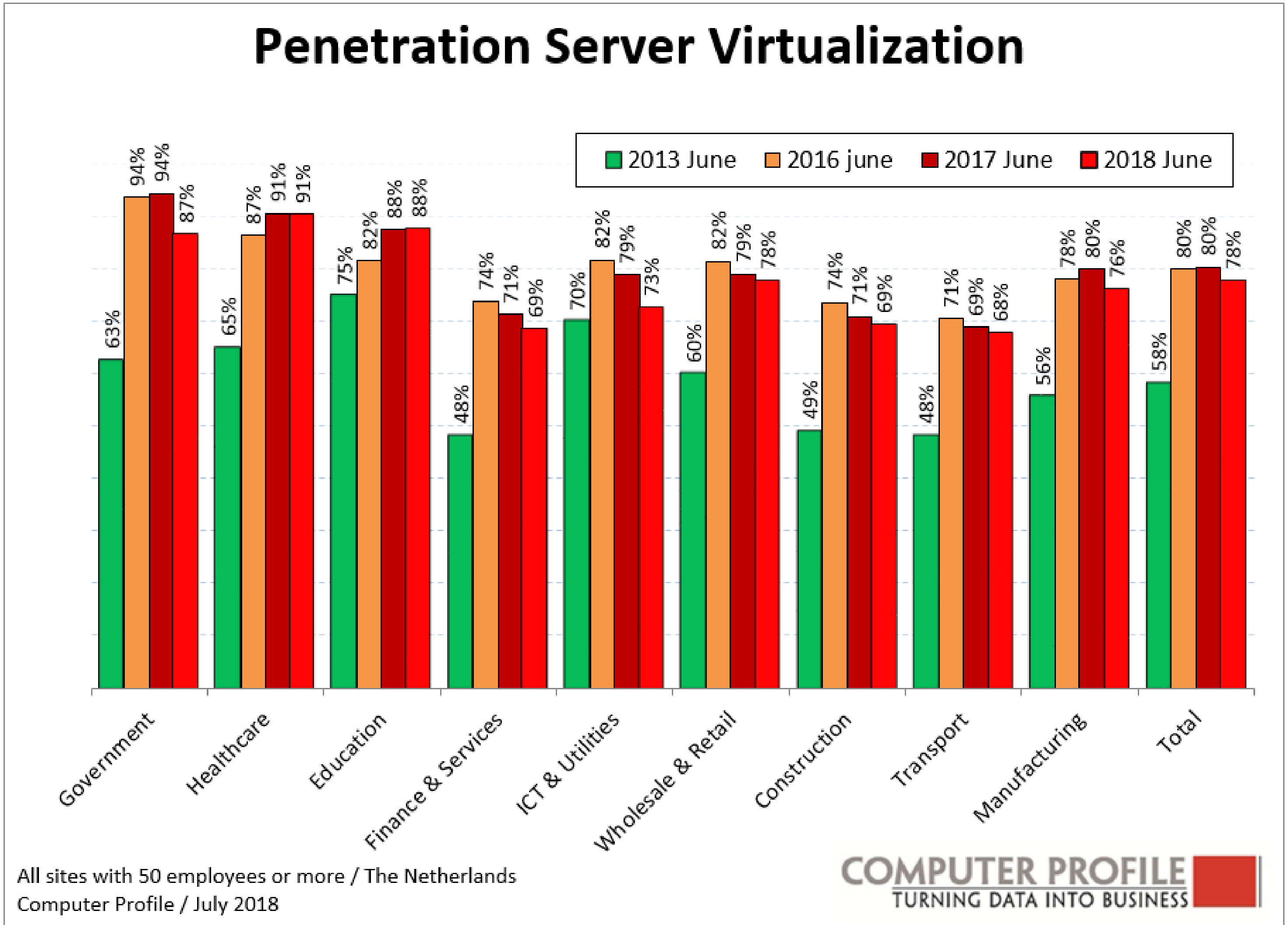
Where are we?

- Many commercial products
  - VMware, Microsoft, Sun, ...
  - Open source: Xen,..
- Good hardware support
  - Well fitted for 64 bits multi-core processors
  - Intel VT (Virtualization Technology) provides native hardware support to the al Virtual Machine Monitor
- Virtualization is technologically mature

Most widespread hypervisors

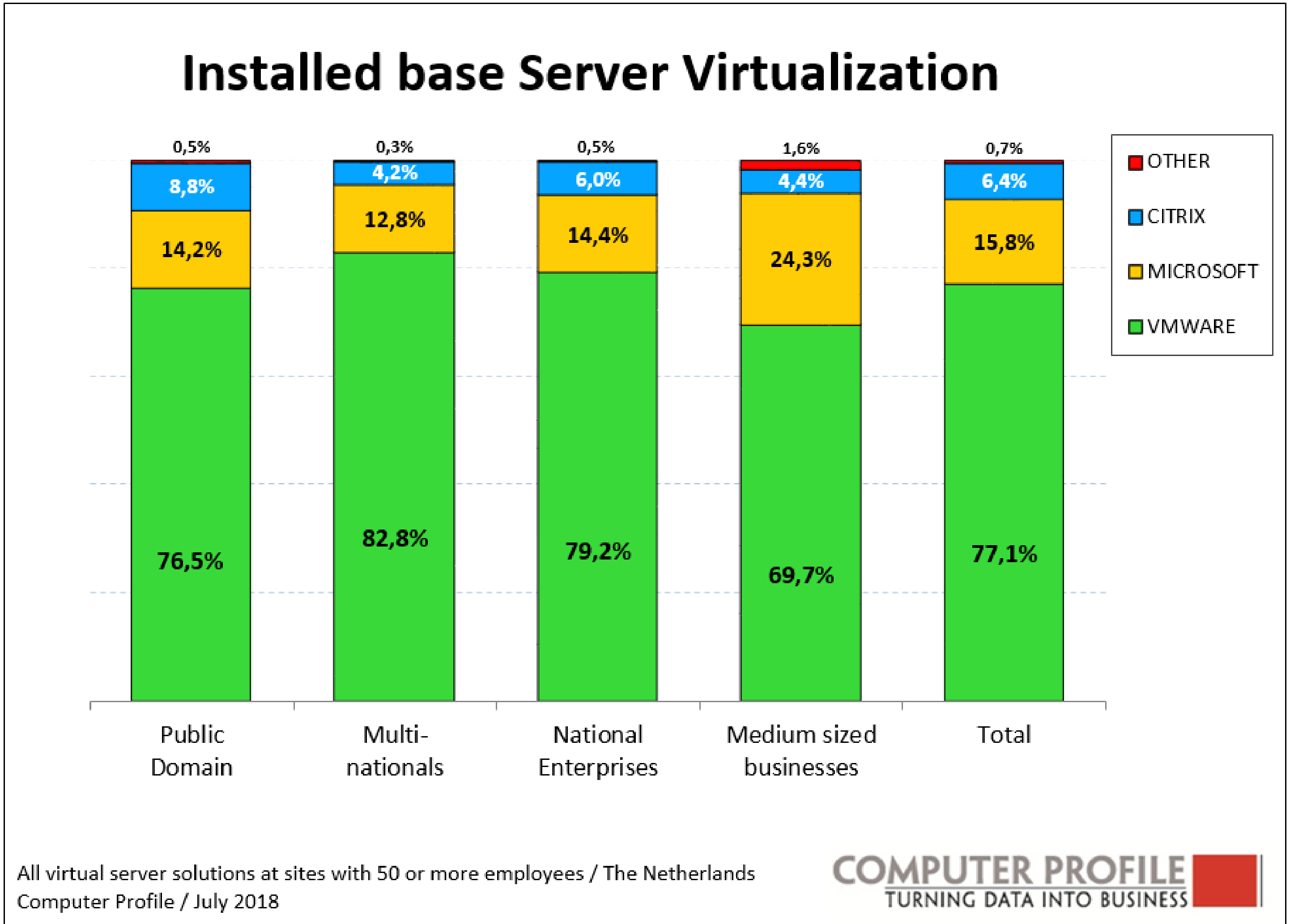
- VMware ESXi
- Xen
- Microsoft Hyper-V
- KVM
- Oracle VM VirtualBox

Some data

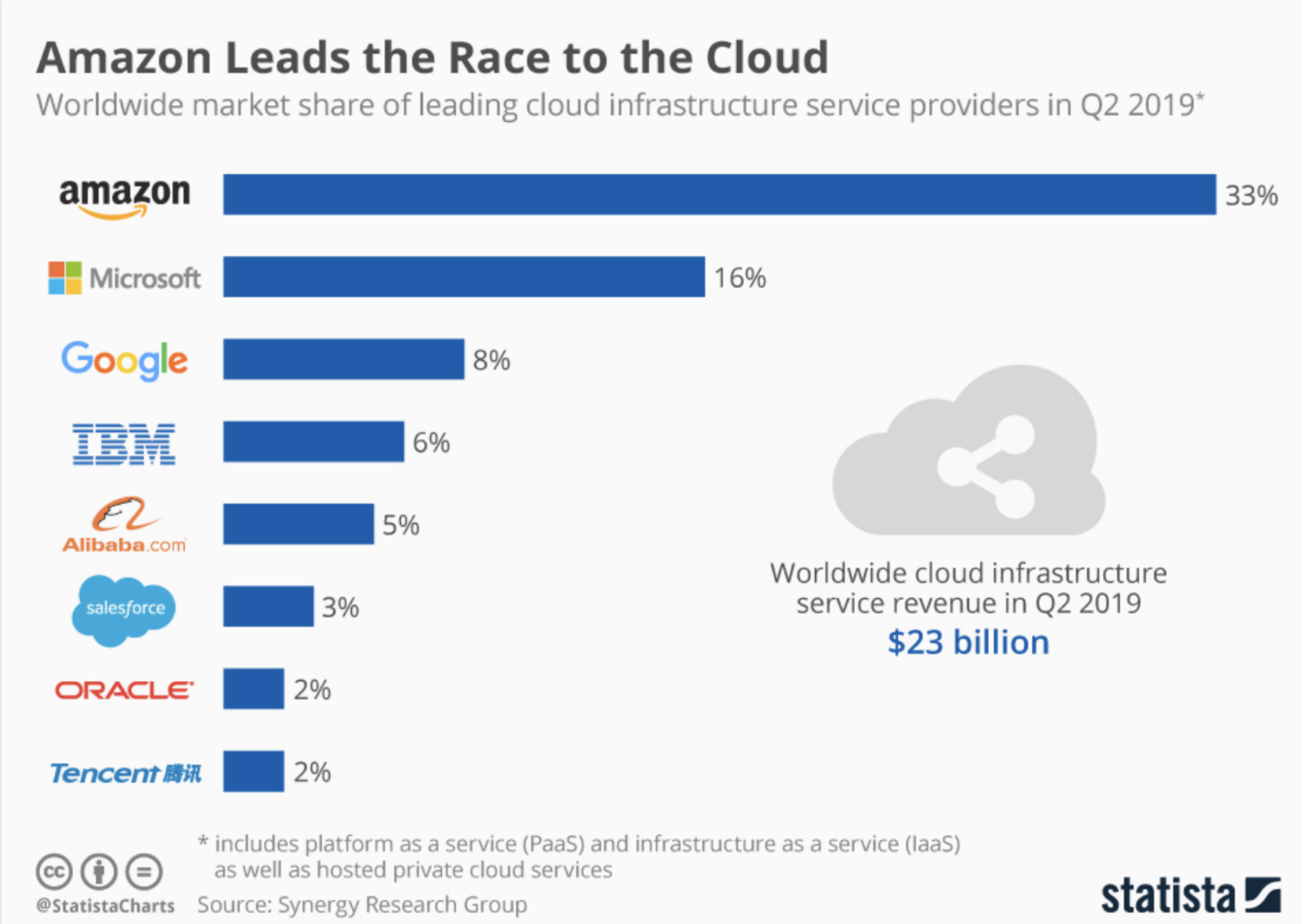


Penetration Server Virtualization
2013 June
2016 june
2017 June
2018 June
Government 63% 94% 94% 87%
Healthcare 65% 87% 91% 91%
Education 75% 82% 88% 88%
Finance & Services 48% 74% 71% 69%
ICT & Utilities 70% 82% 79% 73%
Wholesale & Retail 60% 82% 79% 78%
Construction 49% 74% 71% 69%
Transport 48% 71% 69% 68%
Manufacturing 56% 78% 80% 76%
Total 58% 80% 80% 78%
All sites with 50 employees or more / The Netherlands
Computer Profile / July 2018
COMPUTER PROFILE
TURNING DATA INTO BUSINESS

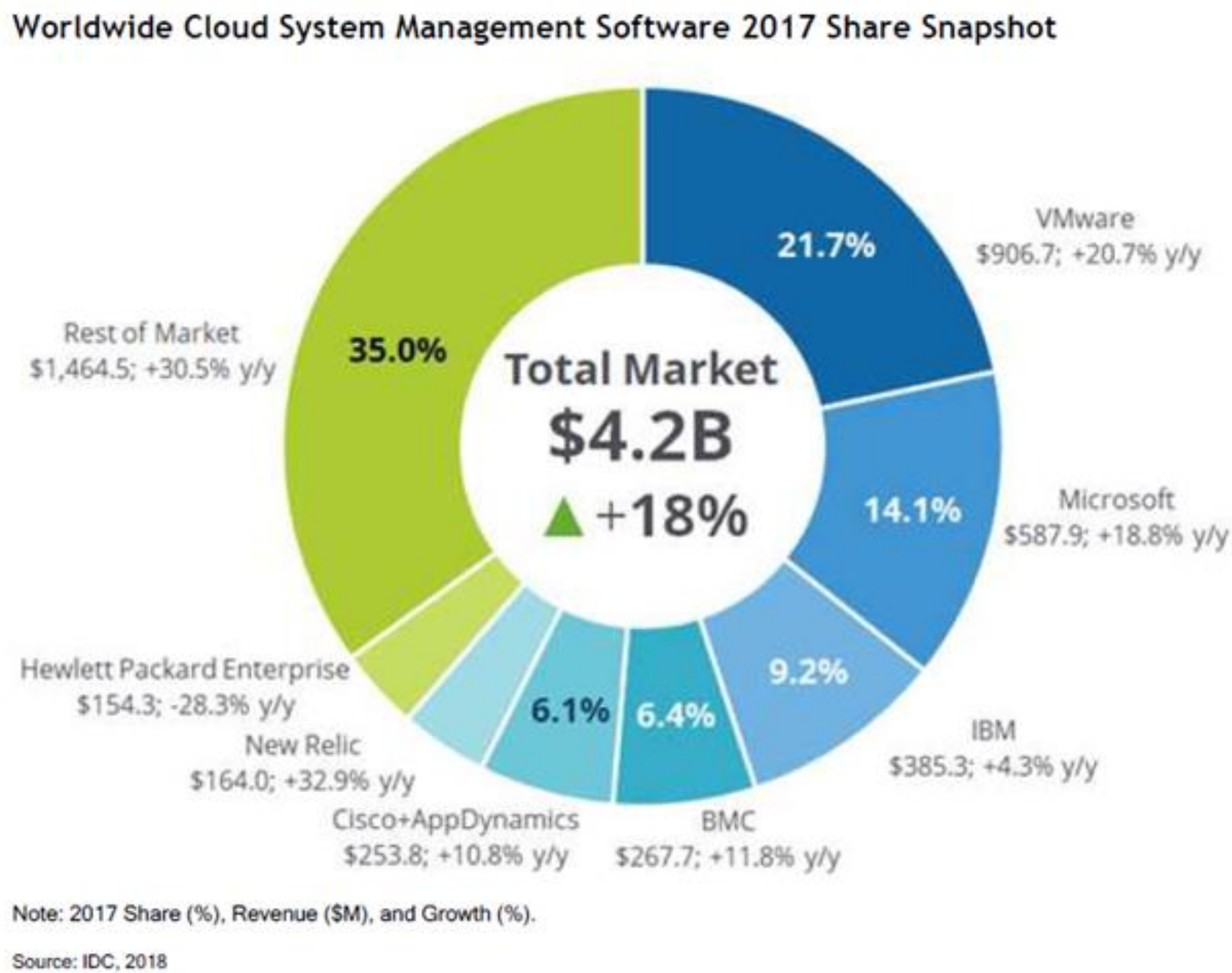
Some data


Installed base Server Virtualization
0,5%
8,8%
14,2%
76,5%
0,3%
4,2%
12,8%
82,8%
0,5%
6,0%
14,4%
79,2%
1,6%
4,4%
24,3%
69,7%
0,7%
6,4%
15,8%
77,1%
OTHER
CITRIX
MICROSOFT
VMWARE
Public Domain
Multi-nationals
National Enterprises
Medium sized businesses
Total
All virtual server solutions at sites with 50 or more employees / The Netherlands
Computer Profile / July 2018
COMPUTER PROFILE
TURNING DATA INTO BUSINESS

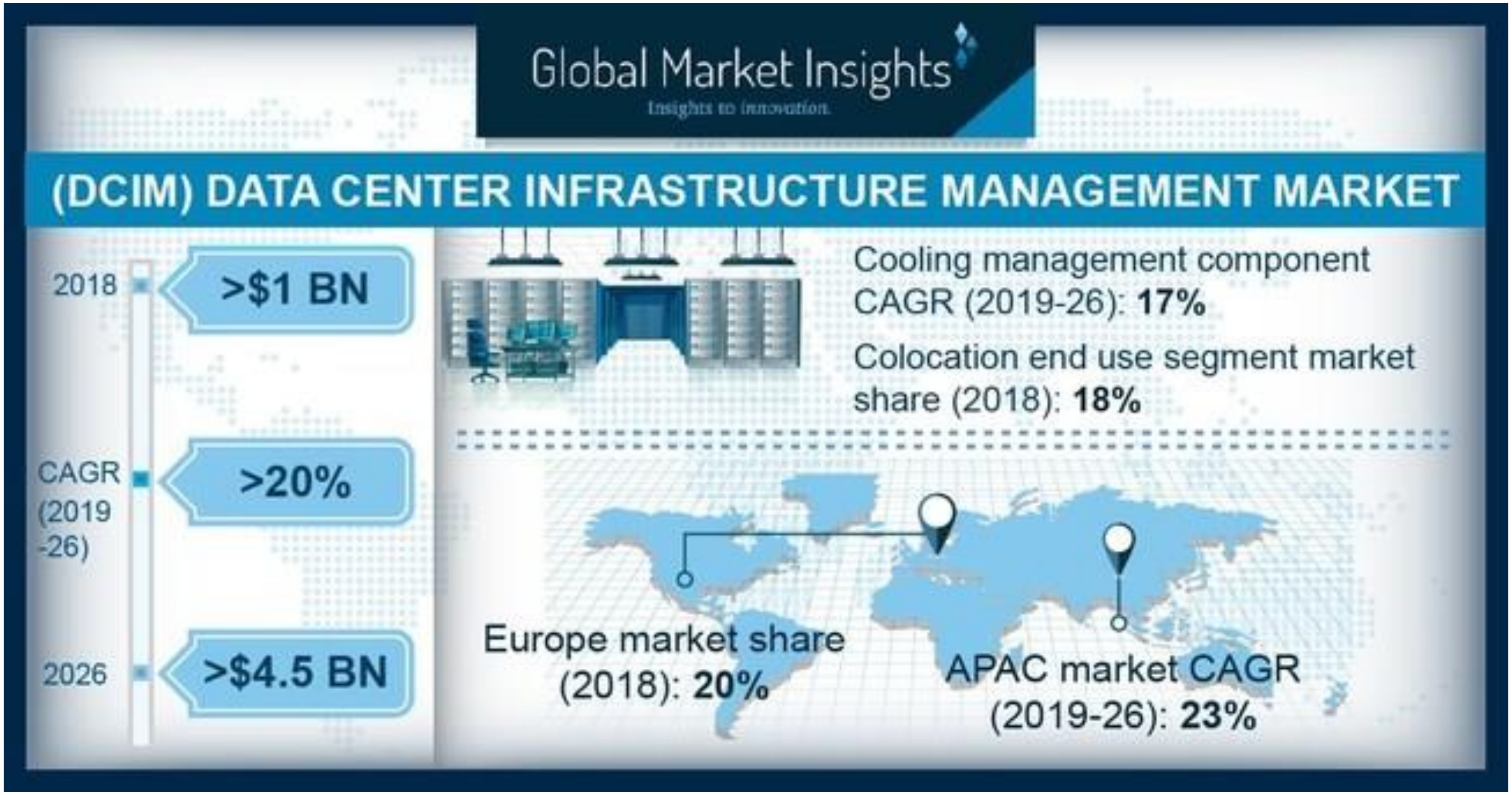
Some data


Amazon Leads the Race to the Cloud
Worldwide market share of leading cloud infrastructure service providers in Q2 2019*
amazon
33%
Microsoft
16%
Google
8%
IBM
6%
Alibaba.com
5%
salesforce
3%
ORACLE
2%
Tencent 腾讯
2%
Worldwide cloud infrastructure service revenue in Q2 2019
$23 billion
* includes platform as a service (PaaS) and infrastructure as a service (IaaS) as well as hosted private cloud services
@StatistaCharts
Source: Synergy Research Group
statista

Some data

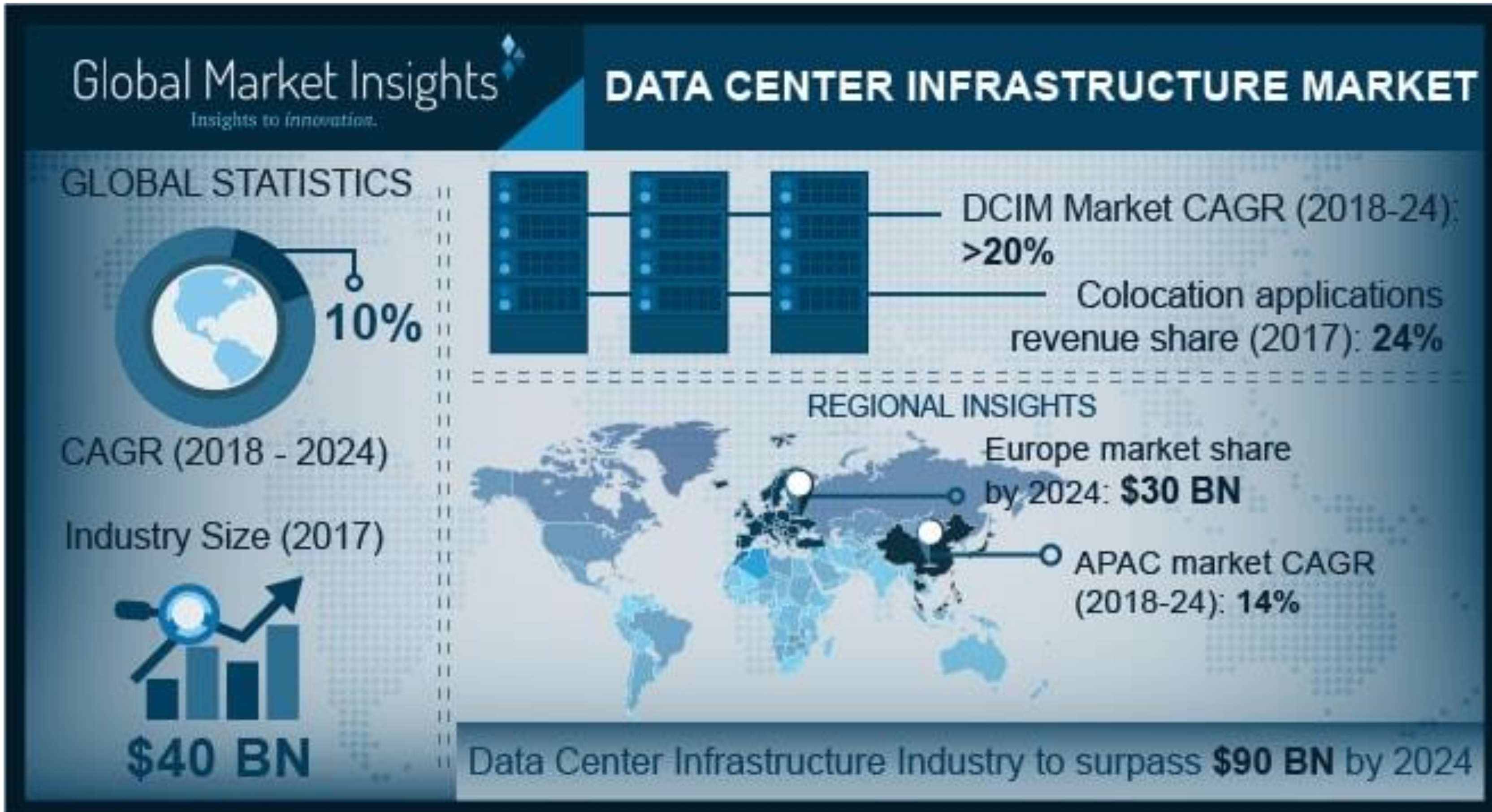

Worldwide Cloud System Management Software 2017 Share Snapshot
VMware
$906.7; +20.7% y/y
21.7%
Microsoft
$587.9; +18.8% y/y
14.1%
IBM
$385.3; +4.3% y/y
9.2%
BMC
$267.7; +11.8% y/y
6.4%
Cisco+AppDynamics
$253.8; +10.8% y/y
6.1%
New Relic
$164.0; +32.9% y/y
Hewlett Packard Enterprise
$154.3; -28.3% y/y
Rest of Market
$1,464.5; +30.5% y/y
35.0%
Total Market
$4.2B
+18%
Note: 2017 Share (%), Revenue ($M), and Growth (%).
Source: IDC, 2018

Some data



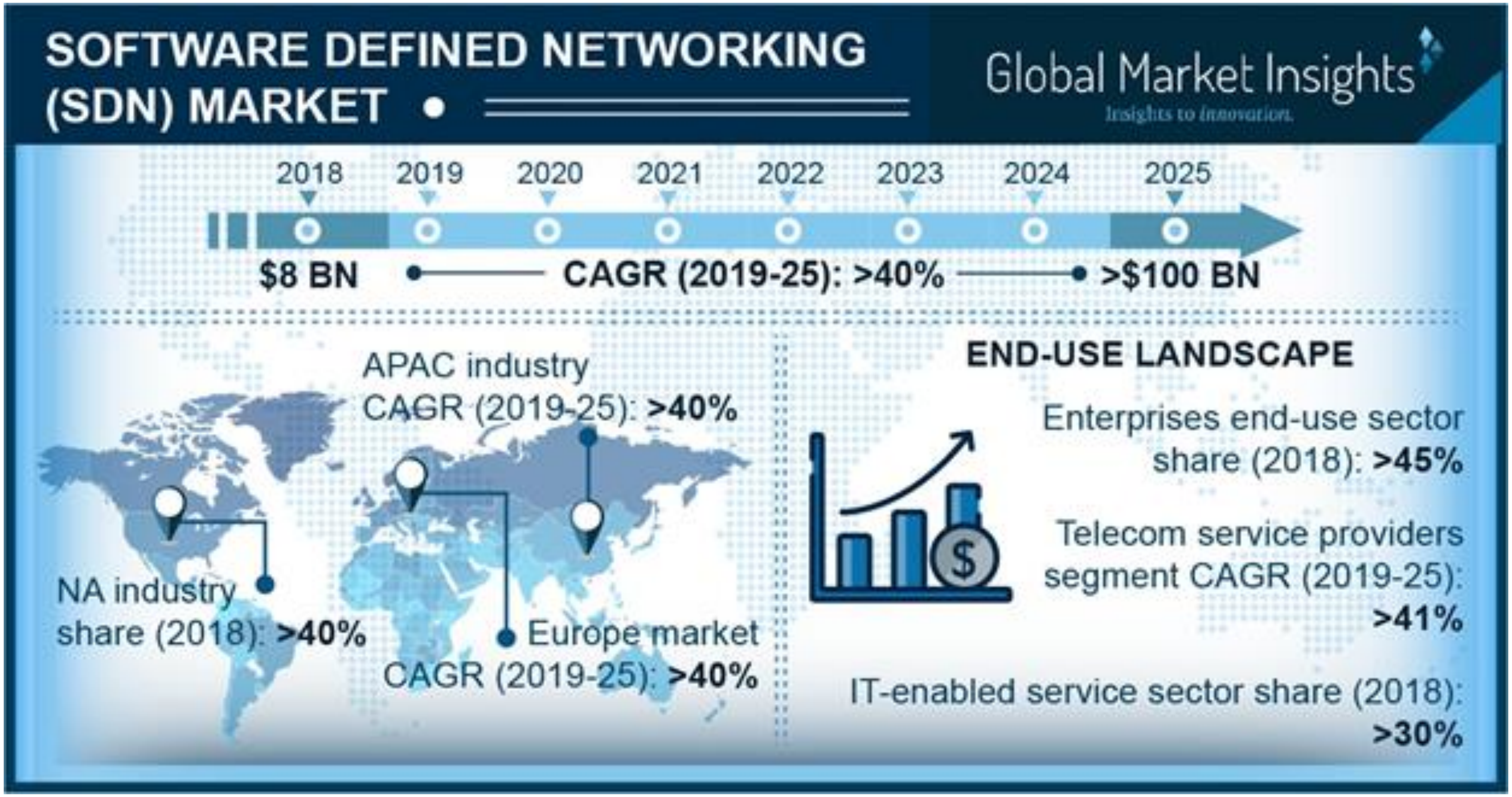
Global Market Insights
Insights to innovation.
(DCIM) DATA CENTER INFRASTRUCTURE MANAGEMENT MARKET
2018
>$1 BN
CAGR (2019 -26)
>20%
2026
>$4.5 BN
Cooling management component CAGR (2019-26): 17%
Colocation end use segment market share (2018): 18%
Europe market share (2018): 20%
APAC market CAGR (2019-26): 23%

Some data



Global Market Insights
Insights to innovation.
DATA CENTER INFRASTRUCTURE MARKET
GLOBAL STATISTICS
10%
CAGR (2018 - 2024)
Industry Size (2017)
$40 BN
DCIM Market CAGR (2018-24): >20%
Colocation applications revenue share (2017): 24%
REGIONAL INSIGHTS
Europe market share by 2024: $30 BN
APAC market CAGR (2018-24): 14%
Data Center Infrastructure Industry to surpass $90 BN by 2024

Some data: more recent



Source: Global Market Inside (2019)

Client and Server virtualization

Virtual desktop

- An area with growing interest
  - Enterprise desktop with centralized security and management
  - They encapsulate the OS providing “virtual hardware”
  - At the basis of architectures for on demand laboratories

- Servers host virtual desktop machines
  - A VMware server can manage more than 1000 virtual machines
  - Virtual desktops used at CS dept. (Crema) as a test environment for teaching networks and security (each student has its own virtual network that can be managed remotely)
    - Machines eventually available also for external students
  - Virtual desktops used also in student labs and classrooms (each student can download locally its own profile)

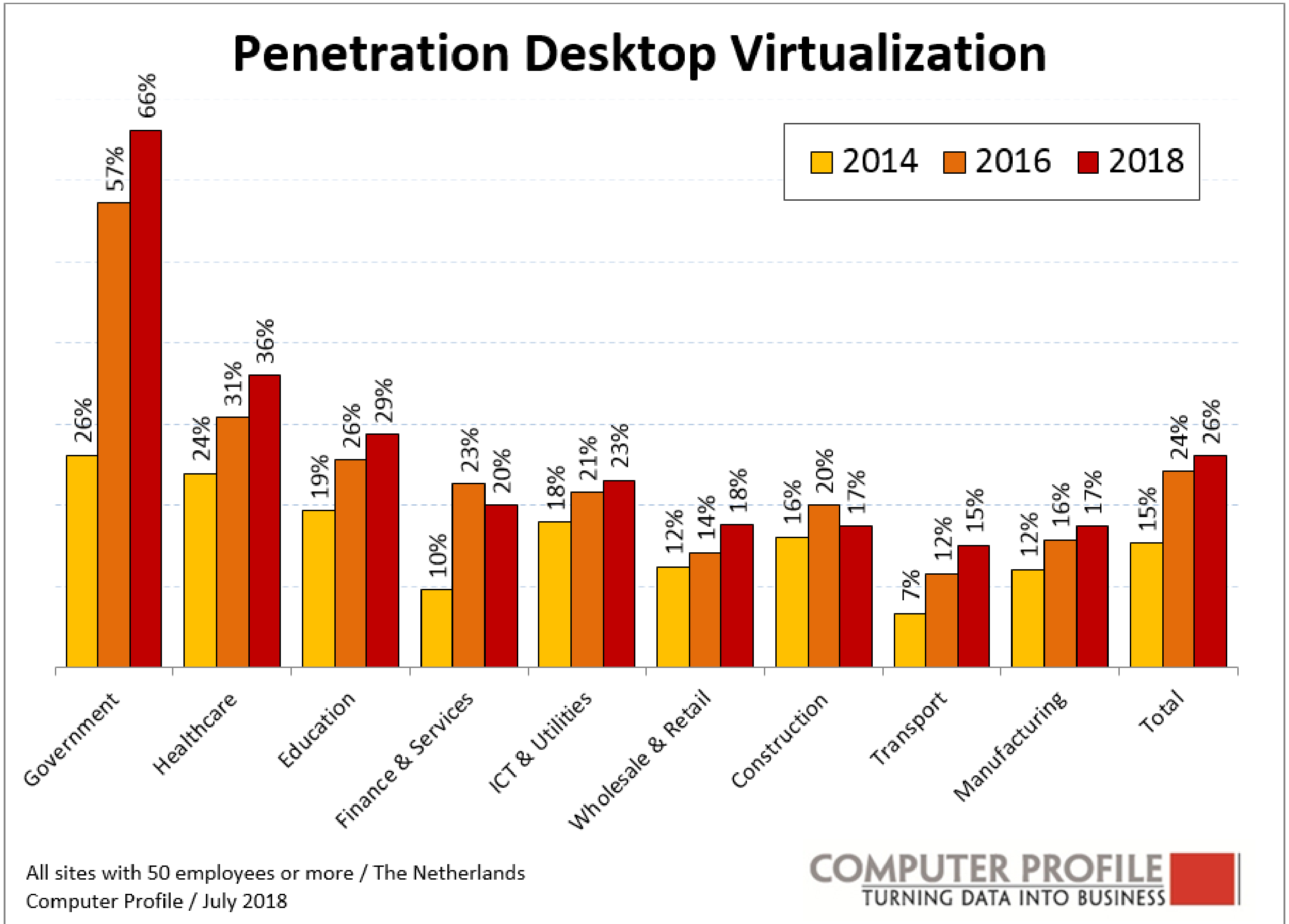
Virtual desktop

- Make available a complete desktop environment working on a datacenter server
- Users connect remotely to the desktop environment from any PC or thin-client device by using a remote virtualization protocol (e.g., RDP for Microsoft)
- Useful for remote and temporary workers, for testing and developing
- The virtual machine executes an unmodified instance of the OS guaranteeing compatibility with all resources
- Virtually everyone can work remotely
- The management system (vSphere in the VMware suite) should guarantee load balance, high availability, scalability and performance

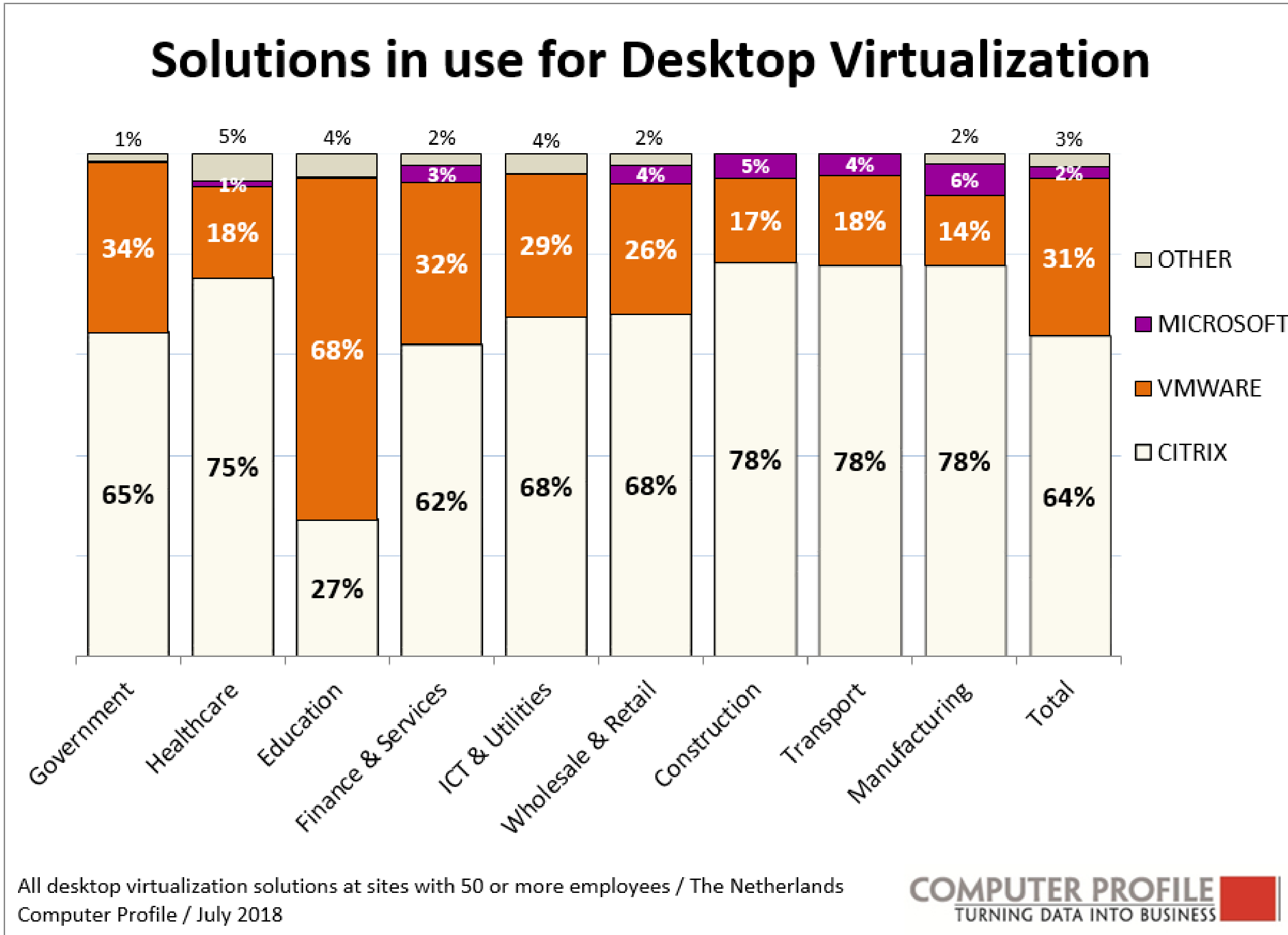
Virtual desktop

- Pros
  - Centralized management and security
  - Business Continuity
  - Isolation and standard way of managing PCs
  - Decreases the need to buy new hardware
  - Decreases the time of adding a new image <10 min
  - Centralized administration of all desktops, eventually located anywhere in the world (vCenter in the VMware suite)

Virtual Desktop Penetration



Virtual Desktop Penetration



Virtual desktop

- Single remote desktop
- Shared desktops
- Virtual desktop machines
- Blade physical desktops

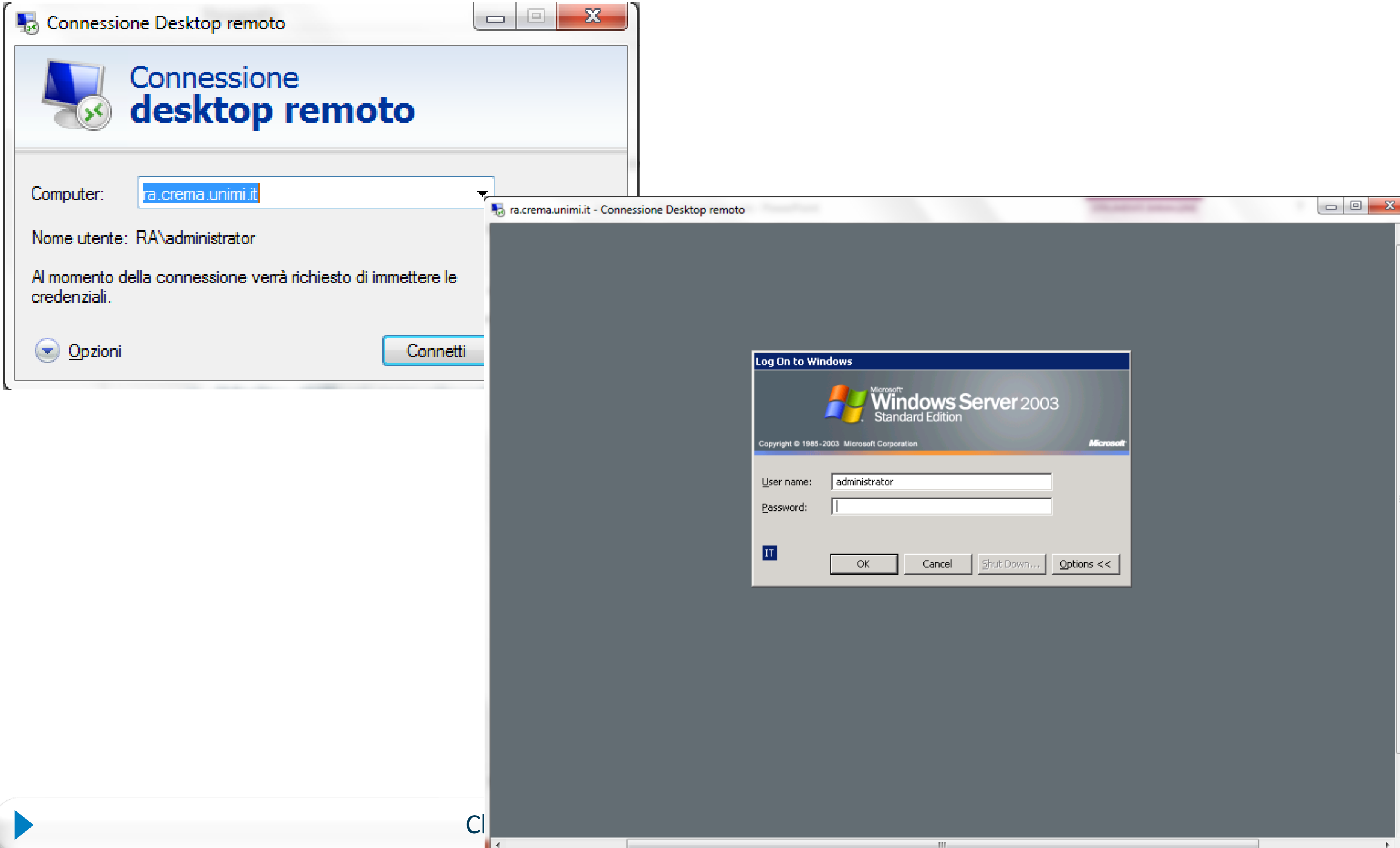
Virtual desktop

- Single remote desktop

  - Remote PC management (e.g., pcAnywhere, WebEx, VNC and Windows Remote Desktop Protocol, TeamViewer)

  - Widely used to virtualize the desktop of a server which we don’t have physical access to

Virtual desktop



Virtual desktop

- Shared desktops
  - Based upon a server hosting desktop users and applications
  - The client can be a standard PC, a notebook, a thin client
  - Desktop sharing is widely used because all the computing power is located on a server and only the monitor, keyboard and mouse are connected to the network
  - This system allows a centralized management of desktops and their applications, simplifying licensing and making easier to solve problems, because user applications are located on the server and not on several machines
  - Not rare to find farms of terminal servers hosting hundreds or even thousands of user desktops

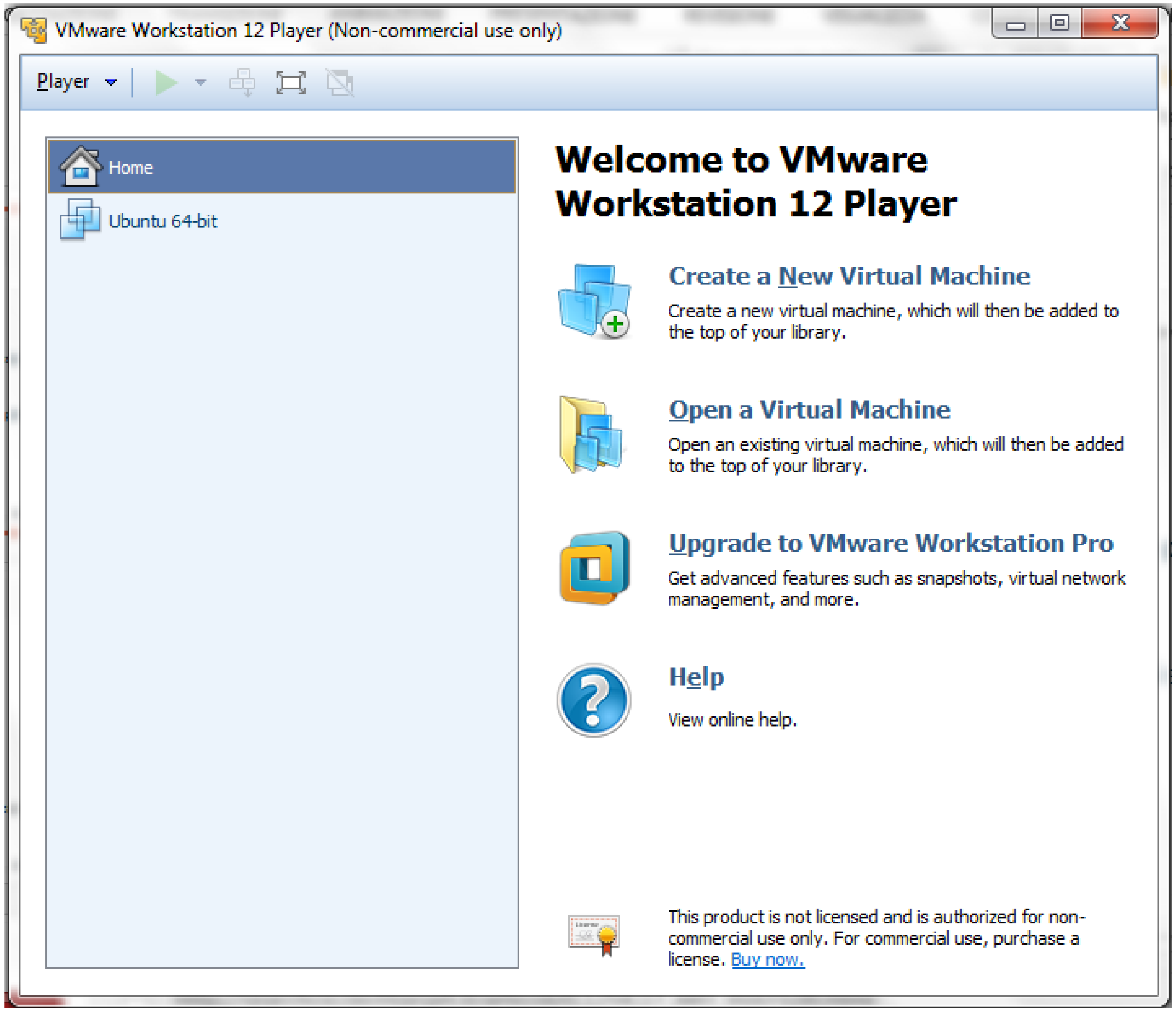
Virtual desktop

- Blade physical desktops
  - Users have their own PC, but the physical hardware is a “blade PC” located in the datacenter
  - Main pros/cons
    - Each user has his own PC, instead of sharing resources with other users on the server
    - Terminal servers hosting shared desktops can be influenced by eventual server faults/problems
    - Blades require more maintenance because there could be 100 blade PCs instead of just one server

Virtual desktop

- Virtual desktop machines
  - The opposite of a shared desktop
  - A single client - PC or notebook - can host multiple desktops
  - Multiple desktops can use different OSes

Virtual desktop

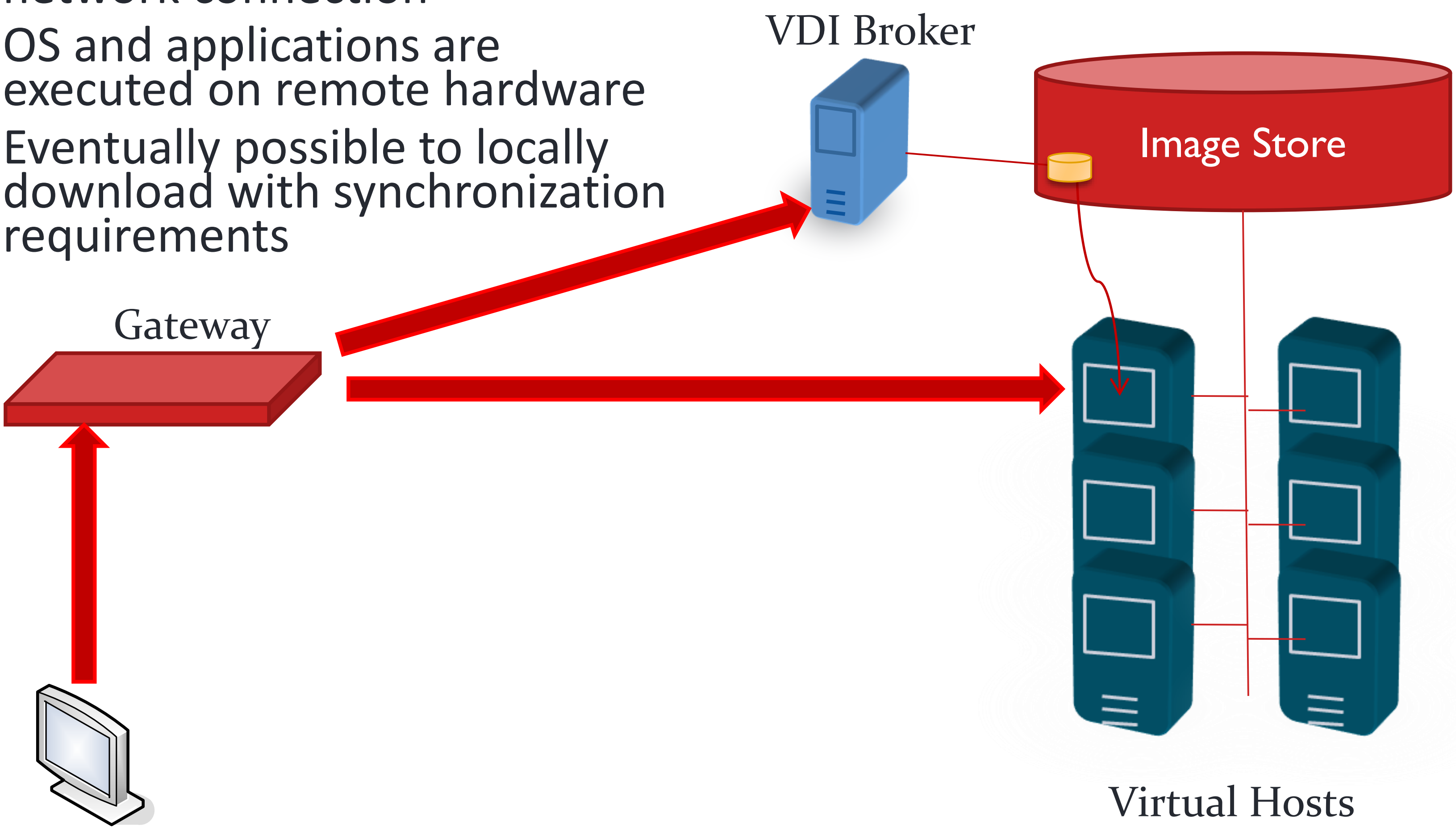


Virtual Desktop Infrastructure

- It executes desktop OSes in server rooms
- Server Virtualization or Blade Servers
- A “broker” connects users with virtual desktops
- Centralized management
- Dedicated images to users or a pool of standard images

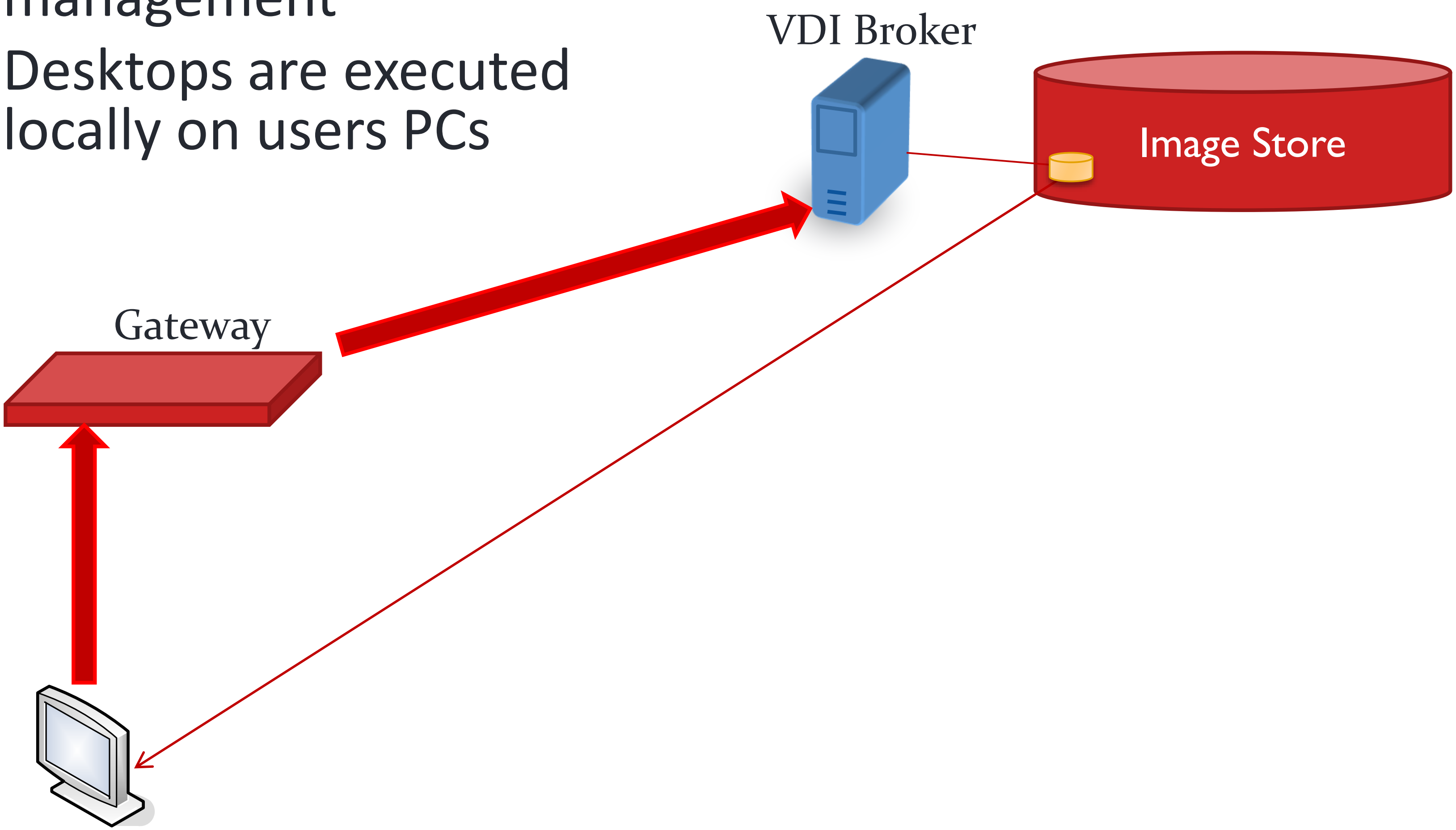
VDI Central Hosting

- It requires a continuous network connection
- OS and applications are executed on remote hardware
- Eventually possible to locally download with synchronization requirements



VDI Local Hosting

- Centralized image management
- Desktops are executed locally on users PCs

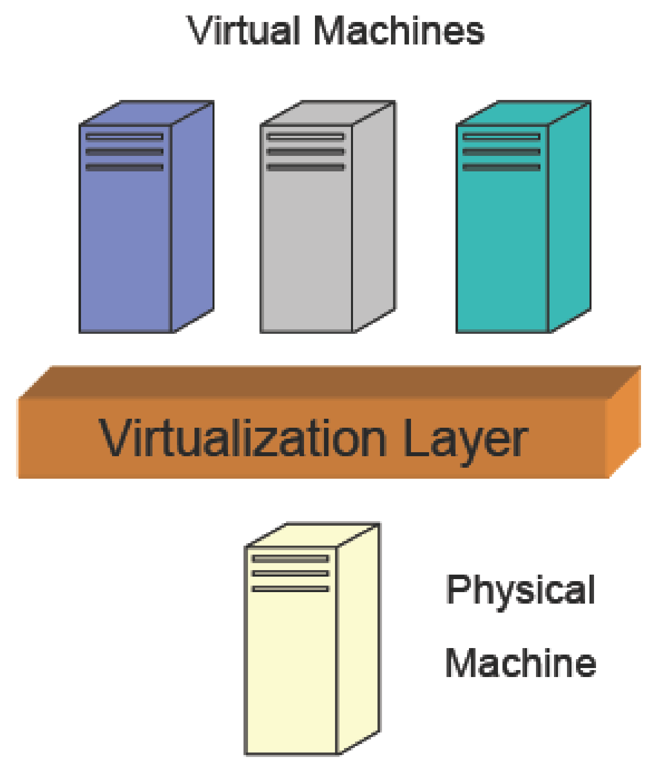


Server virtualization

- It encapsulates the OS and presenting a “virtual hardware”
- It executes several OSes on a single hardware platform
- Consolidation of underused servers

Server virtualization

- Decreases the total cost of ownership (TCO)
  - System usage increases (servers actually has less than 10%of usage)
  - Reduces hardware (25% of TCO)
  - Space, electricity, cooling (50% of datacenter operating costs)
- Increase server usage
- Simplifies management
  - Dynamic provisioning
  - Workload management/isolation
  - Virtual machine migration
  - Reconfiguration
- Better security
- Improves IT financial investment



Server virtualization

- Creation of multiple instances of logical servers on a single physical hardware
- All drivers are virtualized, same virtual hardware independent from physical hardware
- Each virtual machine is completely independent from each other and is not aware of being virtualized

Server virtualization

- Efficient hardware utilization
- Efficient staff
- Matching between needs and resources in the long term
- Fast server provision
- Better redundancy
- Hardware maintenance without application unavailability
- System images are simplified
- Disaster Recovery

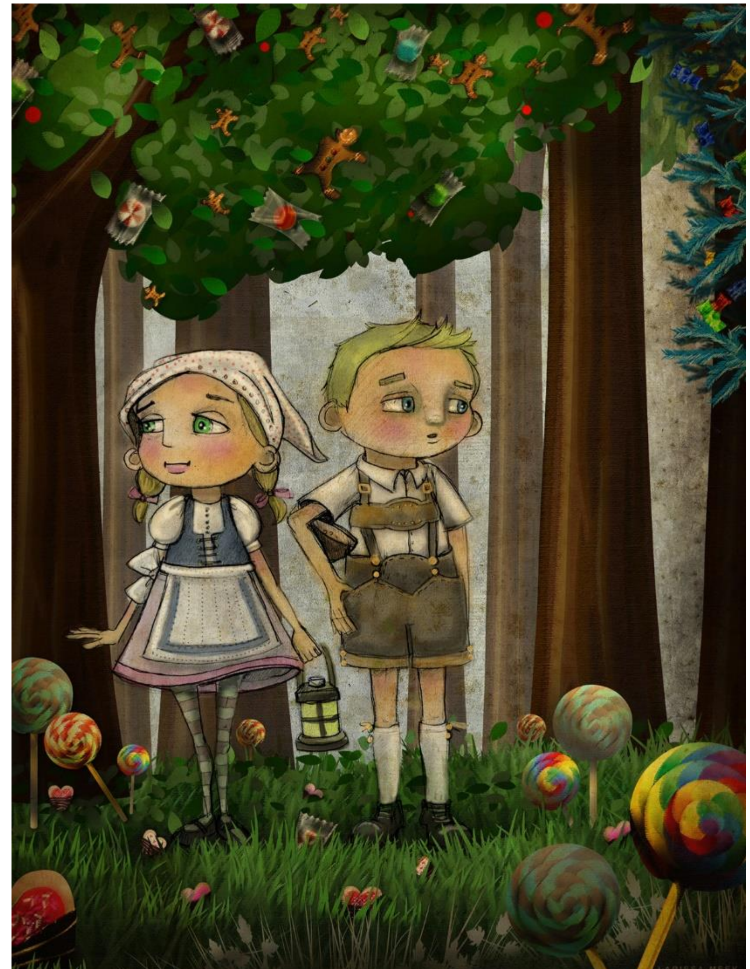
Lesson 2.1: Introduction to Cloud

Claudio Ardagna – Università degli Studi di Milano

Cloud and Distributed Computing

A handful of definitions

- Distributed system
- Parallel system
- Cluster computing
- Meta-computing
- Grid computing
- Peer-to-peer computing
- Global computing
- Internet computing
- Network computing
- Cloud computing

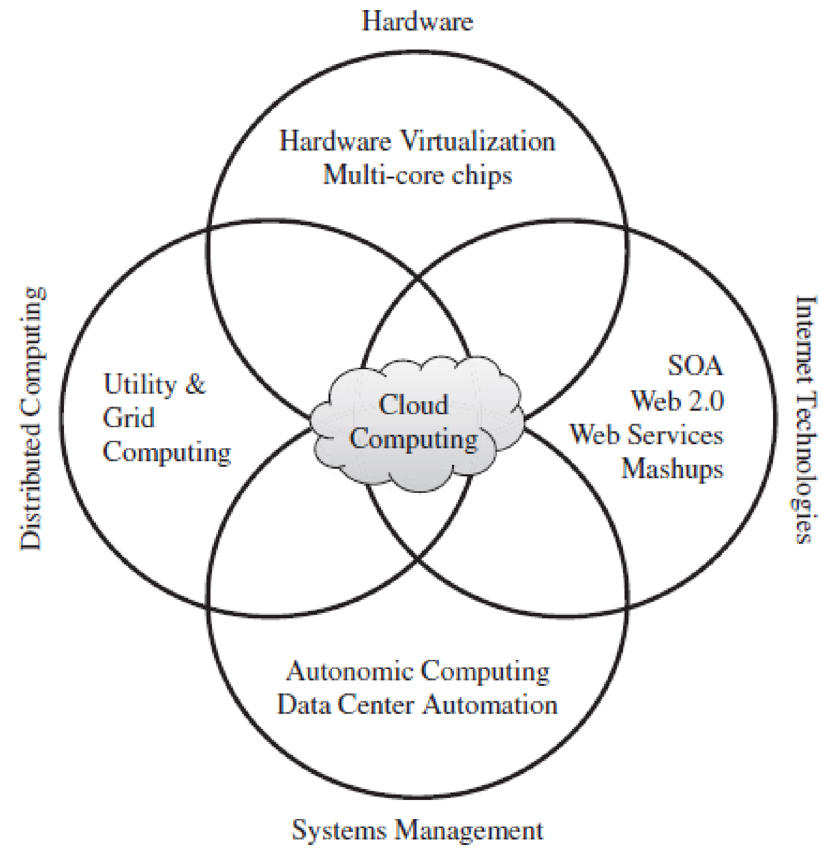


A handful of definitions

- Distributed system
- Parallel system
- Cluster computing
- Meta-computing
- Grid computing
- Peer-to-peer computing
- Global computing
- Internet computing
- Network computing
- Cloud computing

From mainframes to cloud

- Roots of cloud computing
  - Hardware: virtualization, multi-core
  - Internet technologies: Web service, SOA, Web 2.0
  - Distributed systems: cluster, grid
  - System management: autonomic computing, data center automation



Rajkumar Buyya, Andrzej M. Goscinski, Cloud Computing: Principles and Paradigms

Distributed systems

- N autonomous processors (sites): n administrators, n operating systems, n data/control flows
- A single network
- User view: a single system (virtual)
  - «A distributed system is a collection of independent computers that appear to the users of the system as a single computer» Distributed Operating Systems, A. Tanenbaum, Prentice Hall, 1994
- Developer view: client-server

SOA, Web Service, Web 2.0

- Open standard for software integration
- Web services compose applications executed on different messagging platforms
  - Information exchanged between different applications
  - Internal applications now distributed to the outside
- Standardized Web service software stack
  - Search, selection, and composition mechanisms
  - Messagging and packaging
  - Security, QoS
  - Based on HTTP and XML

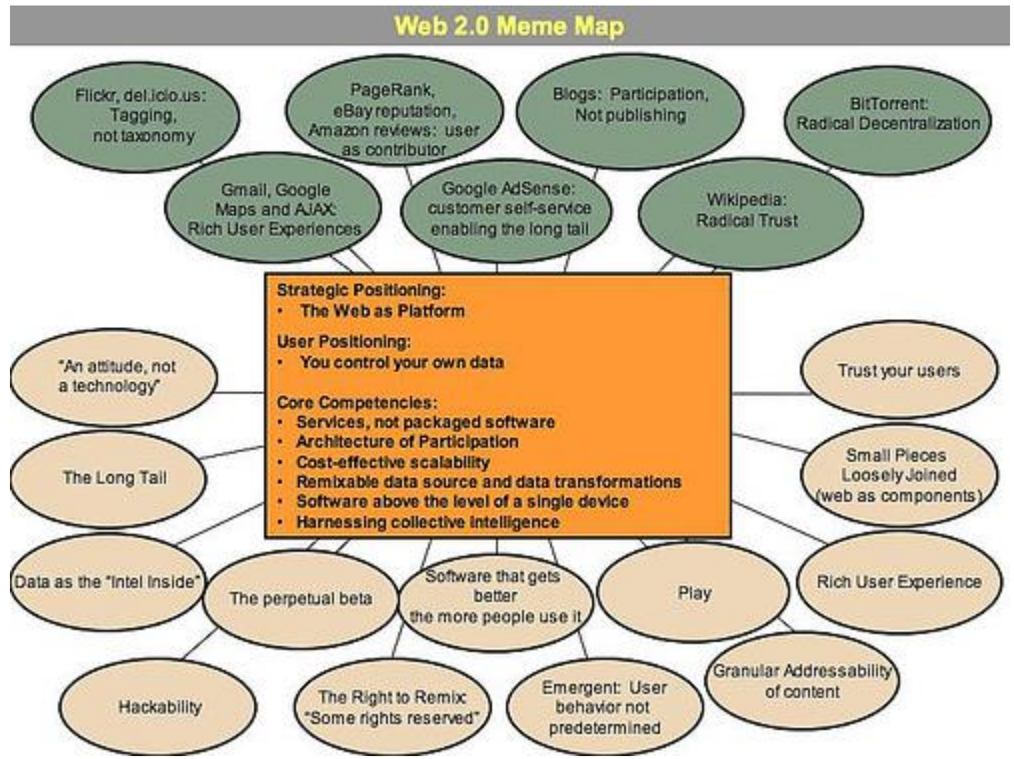
SOA, Web Service, Web 2.0

- Service-Oriented Architecture (SOA)
  - Based on the delivery approach proper of Web services
  - Implement the concepts of distributed system and computing, providing a loosely-coupled, standard, and protocol-independent system
  - Provide software resources as a service with public interface

- Enterprise applications in the SOA environment
  - Collection of services creating complex business logics
  - Used also for consumers and not only for enterprises

SOA, Web Service, Web 2.0

- Web 2.0
  - Made popular by Tim O'Reilly and Dale Dougherty at O'Reilly Media Web 2.0 Conference (end of 2004)
  - Term introduced by Darcy DiNucci in 1999
- The user becomes a content creator
- Include dynamic web, blog, forum, social network, web service…



http://www.oreilly.com/pub/a/web2/archive/what-is-web-20.html

SOA, Web Service, Web 2.0

- Web 2.0 is based on service composition (Web Mashup)
  - Examples are web sites for travel booking including information from hotels and car rental agencies
- Portions of services integrated using standard protocols, such as SOAP and REST
- Cloud applications are developed as service compositions at different layers

Parallel Systems

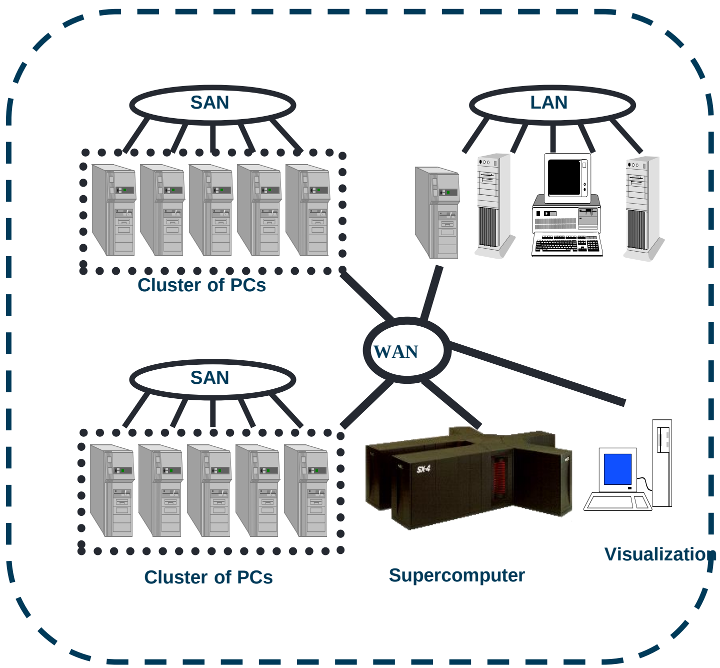
- Parallel systems
  - 1 PC, n nodes: one admin, one operating system
  - Memory: distributed vs shared
  - Developer view: one machine that executes parallel code
  - Different development processes (message passing, distributed shared memory, data parallelism…)

Cluster and network computing

- Cluster computing
  - PC are interconnected through a high-performance network
    - They form a parallel machine
  - Main approaches
    - Dedicated network dedicata (Myrinet, SCI, Infiniband, Fiber Channel...)
    - General-purpose network (fast LAN)
- Network computing
  - Extend cluster computing to WAN
  - A set of distributed PCs and server on MAN/WAN executing parallel code
    - Internet computing (SETI@HOME), P2P, Grid computing...

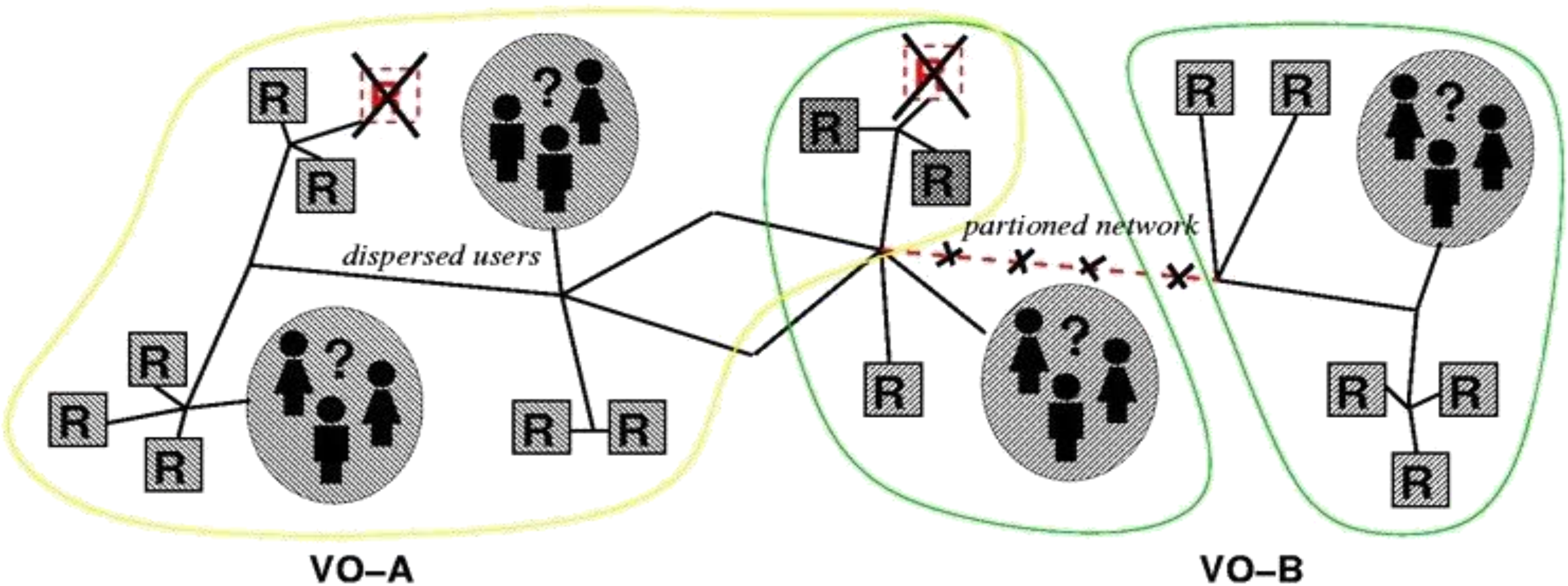
Meta computing (beginning of ’90)

- Meta computer = set of distributed resources able to corrabolatively execute code
- A virtual machine executed on a distributed system



Grid computing

- «Coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organisations» (I. Foster)



Grid computing

- Permit aggregation of distributed resources and transparent access (e.g., TeraGrid, EGEE)
  - Share storage and compute to the aim of executing complex scientific applications (e.g., climate modeling)
- Rely on Web Service standard protocols
- Distributed resources accessed, allocated, monitored, and managed as a single virtualized system
  - On demand delivery of compute services
- Problems
  - No isolation and QoS
  - Heterogeneous software configurations (OS, libreries, compiler…)
  - Require environments with ad hoc configurations
  - Portability
- Possible approach: Virtualization

cloud
computing
Cloud
resources
software
systems
services
infrastructure
like
large
servers
time
open
data
business
type
small
shared
standards
technologies
together
used
SMB

Utility Computing

- IT revolution or return to the origin?
  - From in-house access to computing resources and services, to remote access using the Internet
  - Similar to what happened for electric power distribution
- Utility computing defined as: «on demand delivery of infrastructure, applications, and business processes in a security-rich, shared, scalable, and standard-based computer environment over the Internet for a fee»
- In utility computing
  - Users define requirements in terms of QoS and price willing to pay
  - Service providers define utility in terms of profits

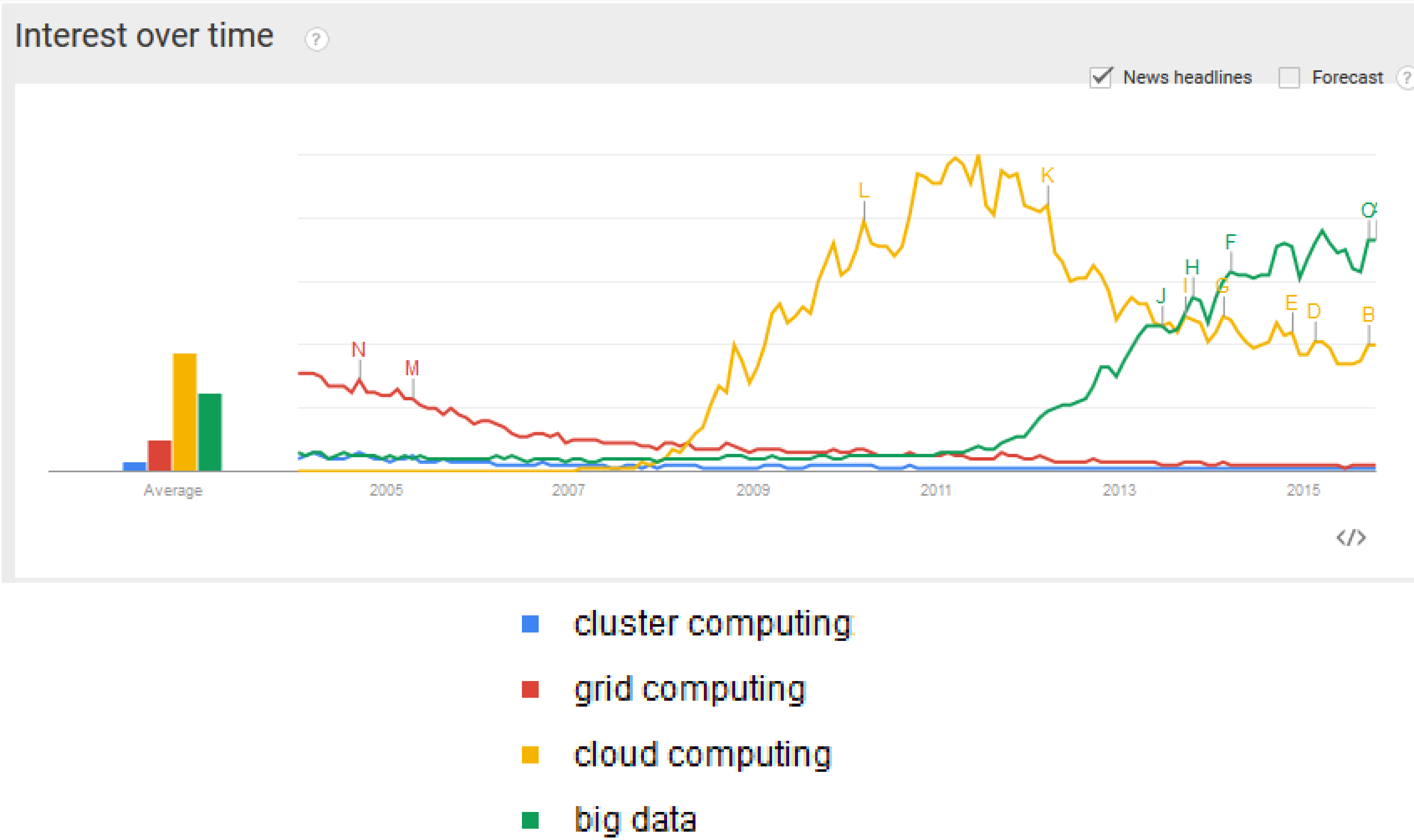
Autonomic Computing

- Autonomic systems with self-management
- Provide adaptation mechanisms
- Reduce user involvement
- Data center automation
  - Application SLA management
  - Management of data center capacity
  - Proactive disaster recovery
  - Automatic provisioning of virtual machines

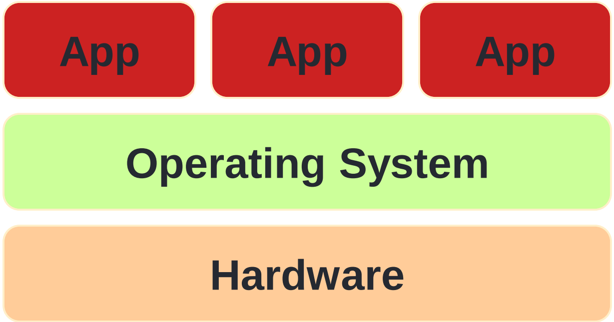
What is a cloud?

- Based on the concepts of utility computing, autonomic computing, distributed systems
- IT as a service
  - Storage, data processing and additional IT services distributed by external providers
  - Applications as "computing utilities", no need to know the computing infrastructure beneath
  - Pay as you go
- Which infrastructure?
  - Basic cloud structure is transparent to users
    - Hardware independent
  - In general, cluster of servers with open source operating systems

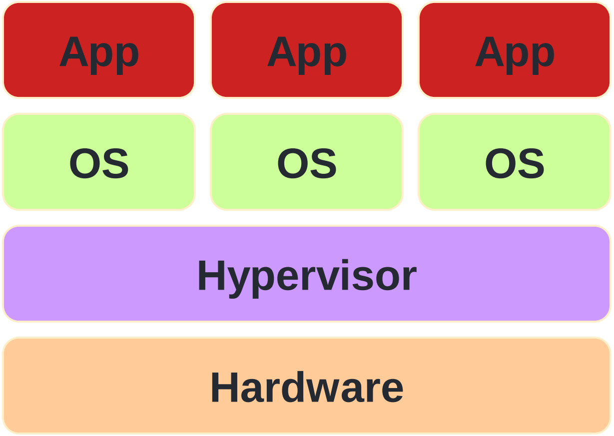
Cloud trend


Interest over time
News headlines
Forecast
N
M
L
K
J
I
H
G
F
E
D
B
Average
2005
2007
2009
2011
2013
2015
cluster computing
grid computing
cloud computing
big data

Key technology: virtualization

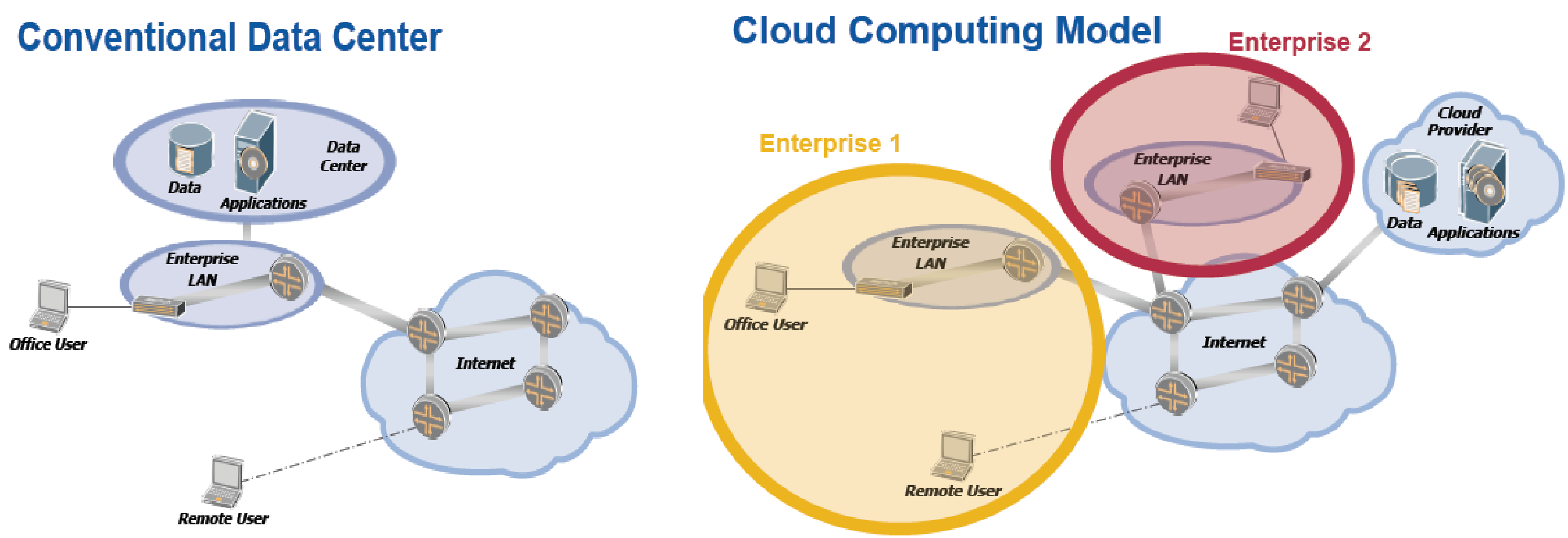


Traditional stack

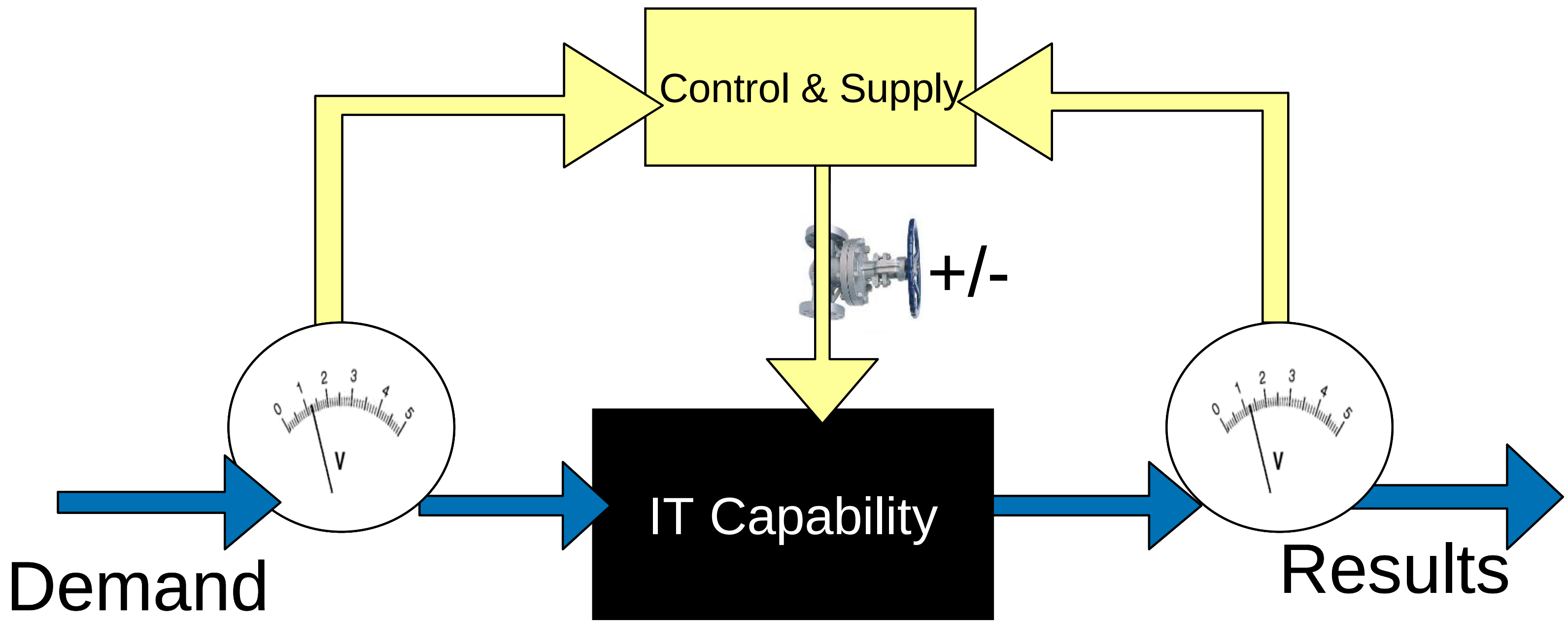


Virtualized stack

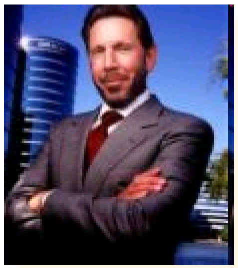
From private data-center to the cloud



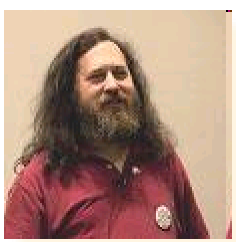
IT Capability = Commodity



Is it a good idea?



The interesting thing about Cloud Computing is that we've redefined Cloud Computing to include everything that we already do. . . . I don't understand what we would do differently in the light of Cloud Computing other than change the wording of some of our ads.
Larry Ellison, Wall Street Journal, 26/9/2008



It's stupidity. It's worse than stupidity: it's a marketing hype campaign. Somebody is saying this is inevitable — and whenever you hear somebody saying that, it's very likely to be a set of businesses campaigning to make it true.
Richard Stallman, The Guardian, 29/9/2008

Cloud Computing in a nutshell

- Similar to electric power distribution
  - We use electricity without knowing how the generation infrastructure and distribution network are implemented
- Concept extended to IT
  - Functionalities released hiding internal functioning
  - Computers integrate distributed components providing processing, storage, data, software resources
- Cloud computing
  - On-demand access to resources
  - Pay-as-you-go paradigm
  - Infrastructure is seen as a «cloud», where resources are made available to users and enterprises
  - Provide computing, storage, software «as a service»

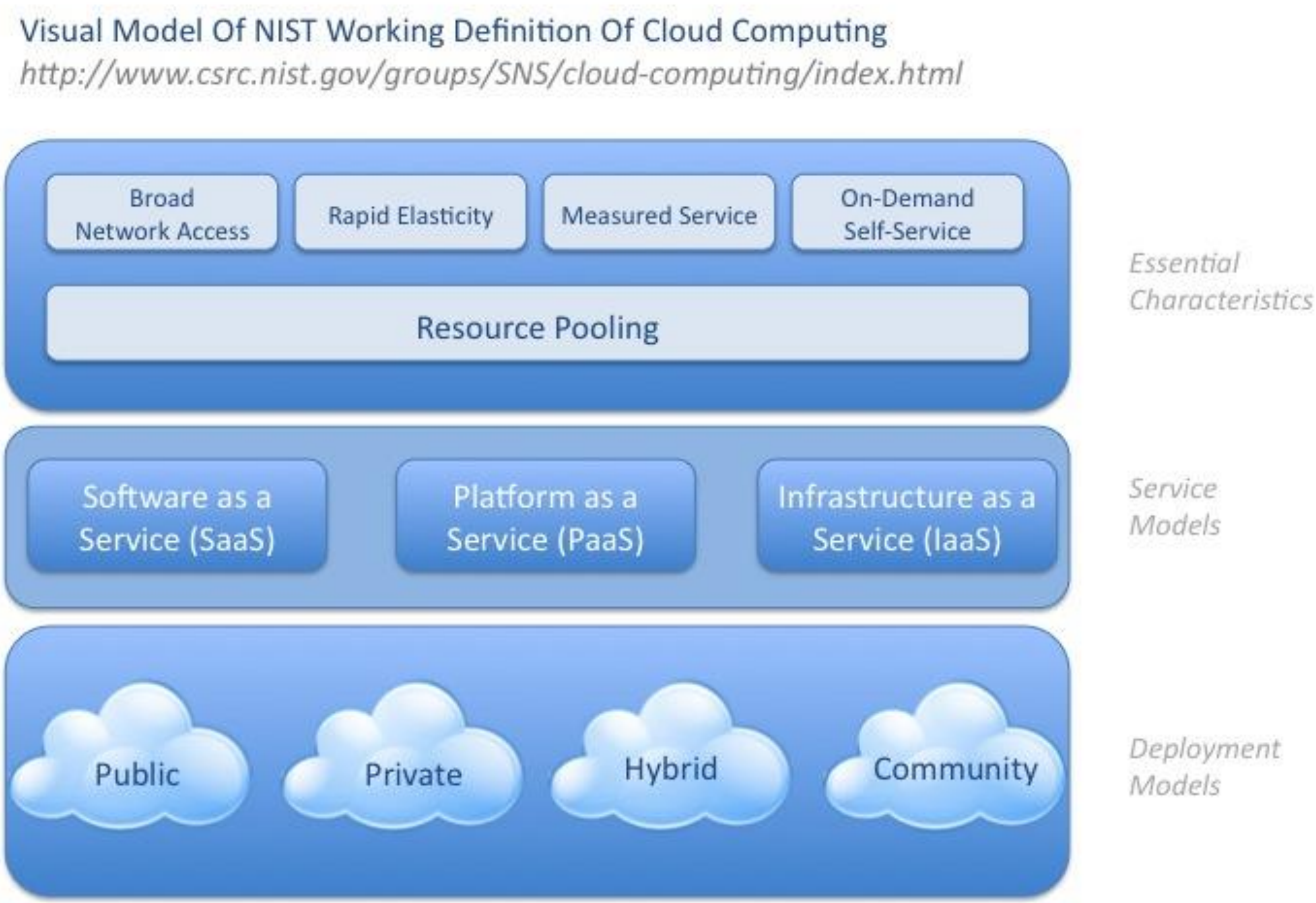
Cloud Computing: Definitions

- Many definitions of cloud computing
  - R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility, Future Generation Computer Systems, 25:599-616, 2009.
  - L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, A break in the clouds: Towards a cloud definition, SIGCOMM Computer Communications Review, 39:50-55, 2009.
  - McKinsey & Co., Clearing the Air on Cloud Computing, Technical Report, 2009.
  - M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, and R. Katz, Above the Clouds: A Berkeley View of Cloud Computing, UC Berkeley Reliable Adaptive Distributed Systems Laboratory White Paper, 2009
  - P. Mell and T. Grance, The NIST Definition of Cloud Computing, National Institute of Standards and Technology, Information Technology Laboratory, Technical Report Version 15, 2009

Cloud Computing: NIST definition

- National Institute of Standards and Technology (NIST) Special Publication 800-145, The NIST Definition of Cloud Computing, Peter Mell and Timothy Grance
- «Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models.

Cloud Computing: NIST definition



Cloud Computing: Main characteristics

- *On-demand self-service*
  - A client can procure resources, server time and network storage on demand and interacting with service providers in an automatic way
- *Broad network access*
  - Resources available through the net
  - Access by means of standard protocols
  - Support for all device types and platforms (e.g., mobile phone, tablet, laptop, workstation)

Cloud Computing: Main characteristics

- *Rapid elasticity*
  - Ability to scale resources in an elastic way based on real needs
  - Scale out, scale in, scale down
  - A client has the impression of having infinite resources, though it is not true
  - No waste of resources typical of on-premise systems
- *Measured service*
  - Cloud manages and optimizes resources dynamically, using metering functionalities (pay per use)
  - Resource usage is monitored, controlled, and logged providing transparency to cloud providers and customers

Cloud Computing: Main characteristics

- *Resource pooling*
  - Resources are divided to serve multiple clients using a multi-tenant model
  - Physical and virtual resources are dynamically (re-)assigned on demand
  - Location independence: client does not have control on the resource location, though it could require a specific location at different granularities (country, state…)
  - Resources include storage, processing, memory, and network bandwidth

Cloud Computing: Additional Characteristics

- *Lower costs*
  - Greater and more efficient use of resources
- *Ease of utilization*
  - No need of hardware and software licenses
- *Quality of Service (QoS)*
  - Address expectations in the contract with the provider
- *Reliability*
  - Provide scalability, load balancing, failover
  - More reliable infrastructure than the ones under direct control
- *Outsourced IT management*
  - User manages the business, someone else the computing infrastructure

Cloud Computing: Multi-Tenancy

- A single software instance is shared by different enterprises/clients (tenants)
- Fundamental aspect of cloud computing
- Users data are virtually isolated, physical isolation not implemented

Cloud Computing: Multi-Tenancy

- Pros
  - Reduced costs for the cloud provider
  - Dynamic access to shared resources
- Cons
  - Users might access data of other users
    - No physical separation
  - Data backup and restore are more difficult

Cloud Computing: Service Models

- *Infrastructure as a Service (IaaS)*
- *Platform as a Service (PaaS)*
- *Software as a Service (SaaS)*

Cloud Computing: IaaS

- Users manage the whole processing (CPU), memory, storage, network, and additional computing resources
  - Amazon, Google, Nuvola Italiana
- Users can install and execute generic code including operating systems and applications
- Users neither manage nor control the cloud infrastructure, while it controls operating systems, storage, and installed applications
  - No need to control hw with all problems due to aging, fault…
- Users has a limited control of network components (e.g., host firewall)

Cloud Computing: IaaS

- Provide virtualized resources on demand
- Provide different servers with different operating systems and an ad hoc stack software
- Amazon provides virtual machines with different combinations of operating systems
  - EC2 Service
  - It is similar to having a physical server
  - Users can start and stop a VM, install a software, connect virtual disks

Cloud Computing: PaaS

- Users install and create applications developed using programming languages, libraries, services, and tools supported by the provider
- Users keep control on installed platform
  - E.g., LAMP (Linux, Apache, MySQL), OwnCloud
- Users do not manage or control the infrastructure (including network, operating system, server, storage)
  - They install neither DB/Apache, nor keep the memory under control
- Users keep control of installed applications and environment configurations for application hosting

Cloud Computing: PaaS

- Abstraction layer making the cloud programmable
- Platform installed on the infrastructure
  - Platform offers an environment where developers install and create their applications
  - No need of knowing the infrastructure beneath
  - Multiple programming models and specialized services (e.g., authentication, payment) offered as building blocks
- GoogleAppEngine provides an environment for development and hosting of web applications
  - Supports different languages such as Python and Java
  - Building blocks: mail service, instant messaging service (XMPP), and many others

Cloud Computing: SaaS

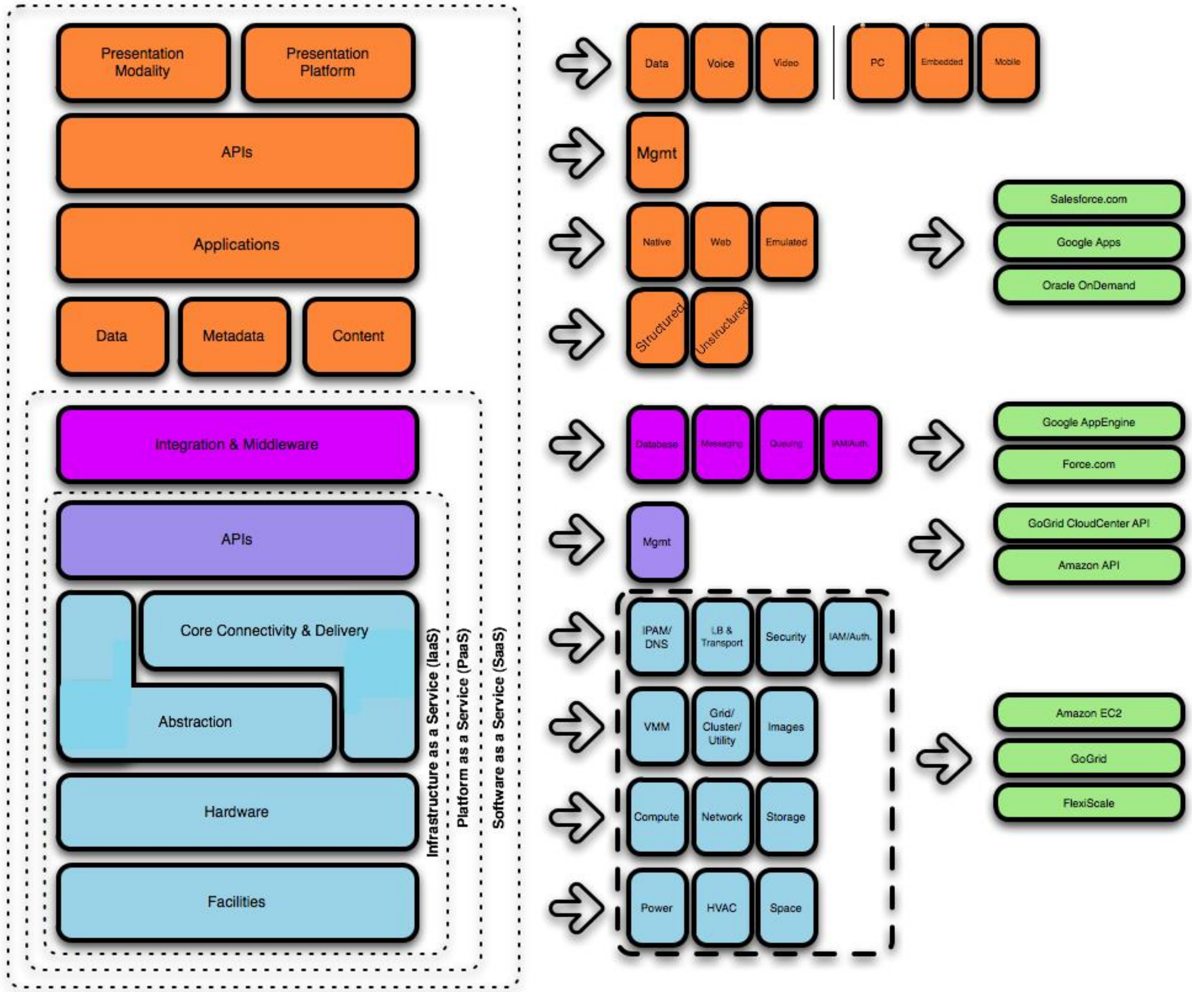
- Users use the applications of a cloud provider executed on the cloud infrastructure
  - For instance, Gmail, Googledocs, Dropbox, Office365, and many other…
- Applications accessible by means of different client devices
  - For instance, web browser (e.g., web-based email) or program interface
- Users do not manage or control the cloud infrastructure (including network, operating system, server, storage), and application-specific functionalities
- Users can only control a limited number of user-specific configurations
  - They only use tools that are useful for their business

Cloud Computing: SaaS

- Applications are deployed at the top of the cloud stack
- Services are accessible using a browser
- Paradigm shift: from software installed locally to software installed remotely
- Reduce the effort of the users in the application management, and simplify development and testing for providers
- Salesforce.com
  - Online CRM, on demand access and configuration of applications

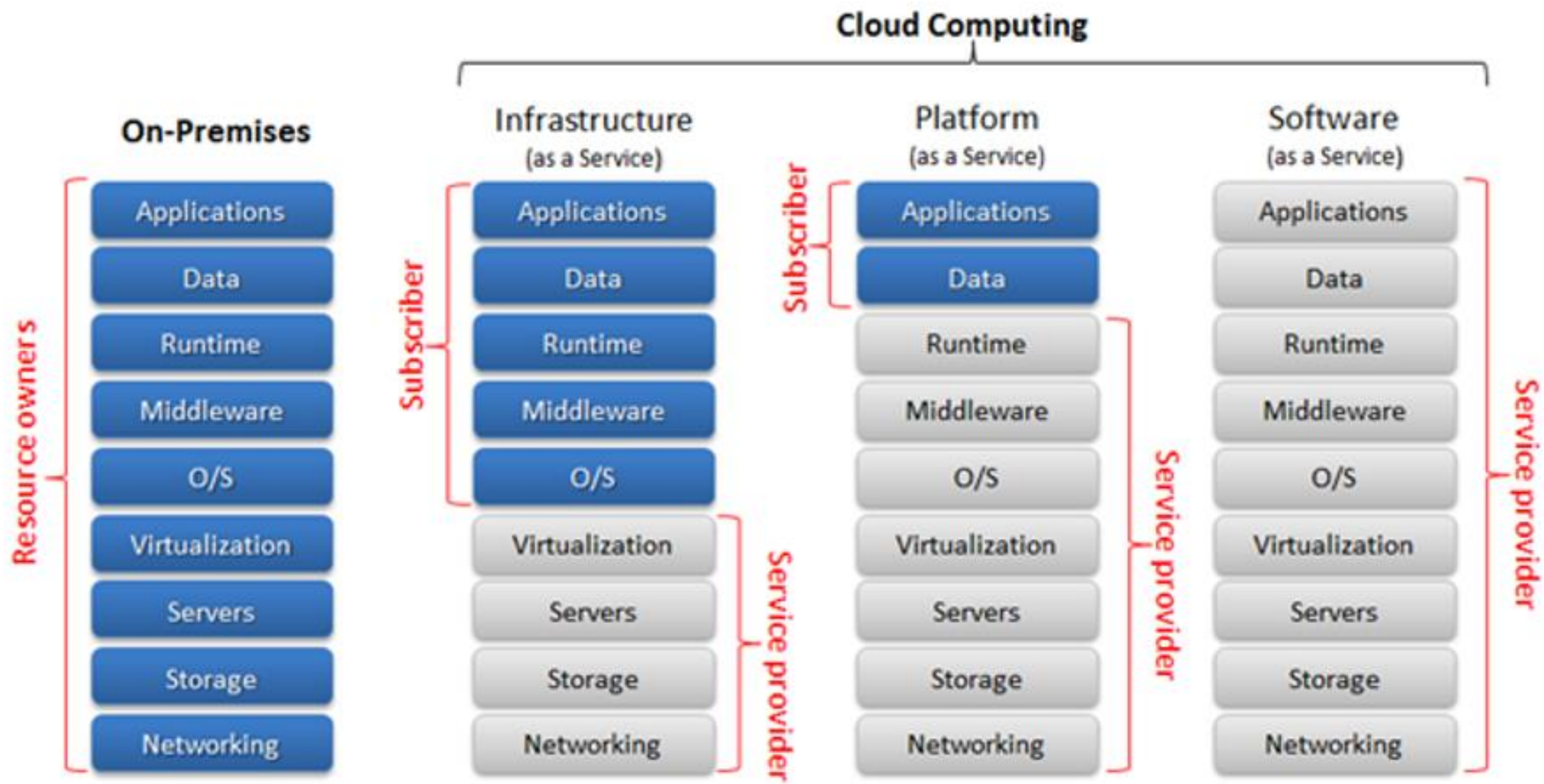
Cloud Computing: Stack Overview

- Cloud Reference Model
- The lower portion includes hardware and infrastructure (including network)
- Each model inherits the ability of the models beneath



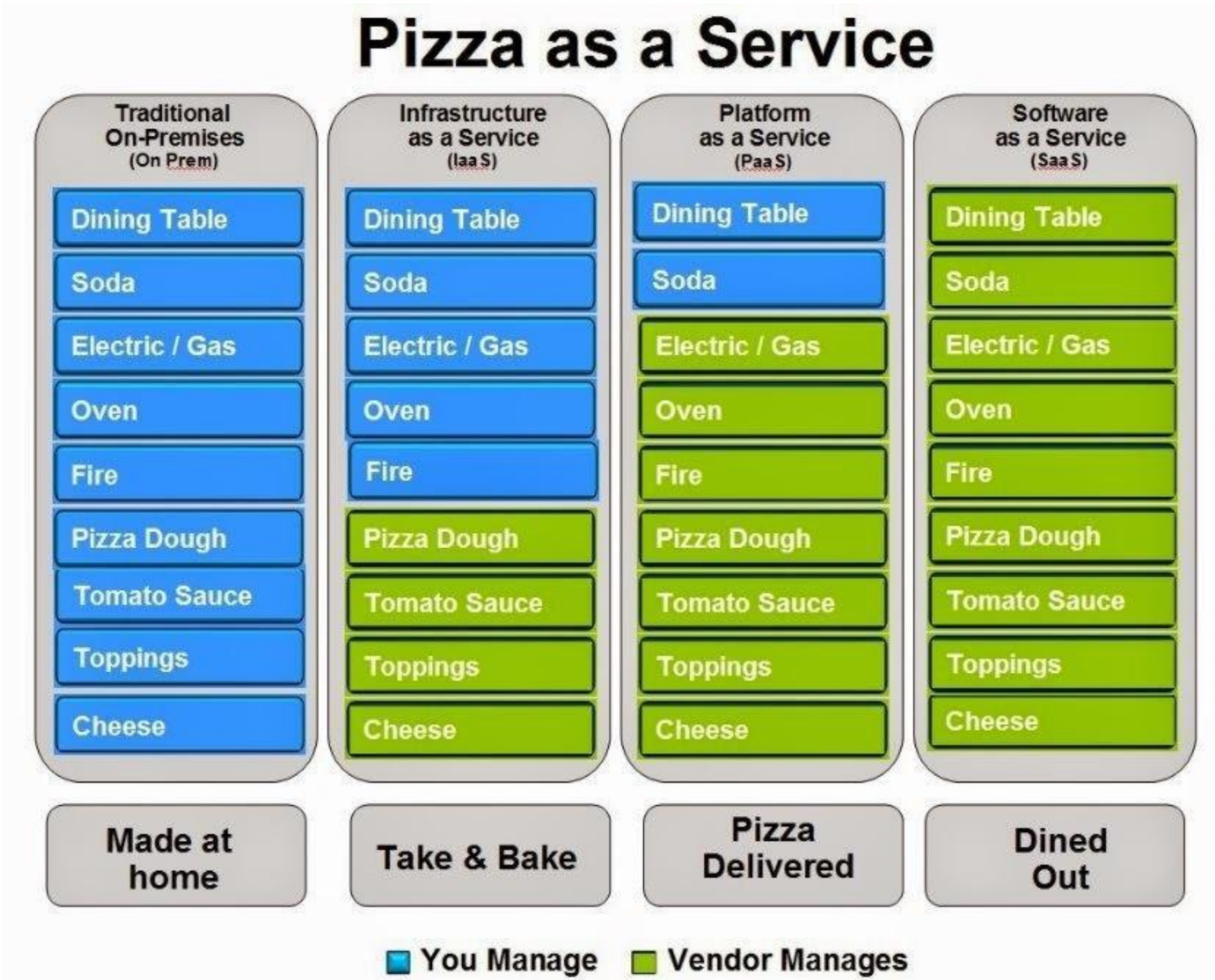
https://www.rationalsurvivability.com/blog/2009/03/update-on-the-cloud-ontologytaxonomy-model/

Cloud Computing: Delivery Models



http://blogs.technet.com/b/yungchou/archive/2010/12/17/cloud-computing-concepts-for-it-pros-2-3.aspx

Cloud Computing: Real Life



https://www.linkedin.com/pulse/20140730172610-9679881-pizza-as-a-service

Cloud Computing: Taxonomy

- http://cloudtaxonomy.opencrowd.com/static/cloudtaxonomy/pdf/cloud_taxonomy_arch.pdf

INFRASTRUCTURE SERVICES

STORAGE
Amazon S3
Amazon SimpleDB
Microsoft SSDS
Rackspace Mosso CloudFS
Google BigTable
Amazon EBS
HP Cloud Object Store
Internap Cloud Storage
AT&T Synaptic Storage
Softlayer Cloud Storage
Zetta

COMPUTE
Amazon EC2
Serve Path GoGrid
Rackspace Mosso Cloud
Joyent Accelerations
Agathon Group
Flexiscale
Elastichosts
Hosting.com CloudNine
Enki
Terramark
ITRiCTY
Savvis Cloud Compute
HP Helion
IBM Smart Cloud

SERVICE MANAGEMENT
Rightscale
Scalr
CohesiveFT
Kaavo
CloudStatus
Ylastic
enStatus
New Relic
Bitnami

CLOUD SOFTWARE

DATA
Apache Accumulo
AsterData
Cassandra
Clustrix
CouchDB
DBShards
InfiniDB
MongoDB
NuoDB
Redis
Riak

MAP REDUCE
Actian Analytics
Cloudera
GridGain
Hadoop
Hortonworks
JethroData
MapR
Pig
SpliceMachine
Storm

CLOUD MANAGEMENT
3Terra App Logic
VMWare Ops
Eucalyptus
Chef
Puppet
Hyperic
Enomaly
OpenStack
CloudStack

PLATFORM SERVICES

GENERAL PURPOSE
Force.com
Agile Apps Live
Rollbase
Google App Engine
Engine Yard
ElasticHosts
Caspio
Qrimp
MS Azure Services Platform
CogHead
Bungee Labs Connect
Jelastic
OpenShift

BUSINESS INTELLIGENCE
Micro Strategy
Cloud9 Analytics
K2 Analytics
Logi Analytics
Panorama
PivotLink
Indicee
Good Data
Kognito Cloud
bime

INTEGRATION
Amazon SQS
MuleSource Mule OnDemand
Boomi
OpSource Connect
Cast Iron
Microsoft BizTalk Services
gnip
SnapLogic SaaS Solution Packs
Appian Anywhere

BIG DATA AS A SERVICE
Actian Cloud Edition
Altiscale
Amazon Kinesis
BigML
Datameer
Mortar Data
Qubole
RMS (one)
Wolfram Data Framework

SOFTWARE SERVICES

BILLING
Aria Systems
OpSource
Redi2
Zoura

CRM
NetSuite
Parature
Responsys
Rightnow
Salesforce.com
LiveOps

PROJECT MANAGEMENT
Aconex
Assembla
Huddle
smartsheet
trackerSuite.Net

SALES
Xactly
StreetSmarts
Salesforce.com

DESKTOP PRODUCTIVITY
Zoho
Google Apps
Parallels
ClusterSeven

FINANCIALS
Concur
Xero
Workday
Beam4d
Intuit
Expensify

HUMAN RESOURCES
Taleo
Workday
iCIMS
Successfactors
Saba
StaffRoster

CONTENT MANAGEMENT
Clickability
SpringCM
CrownPoint
Netdocuments
Tie Kinetix

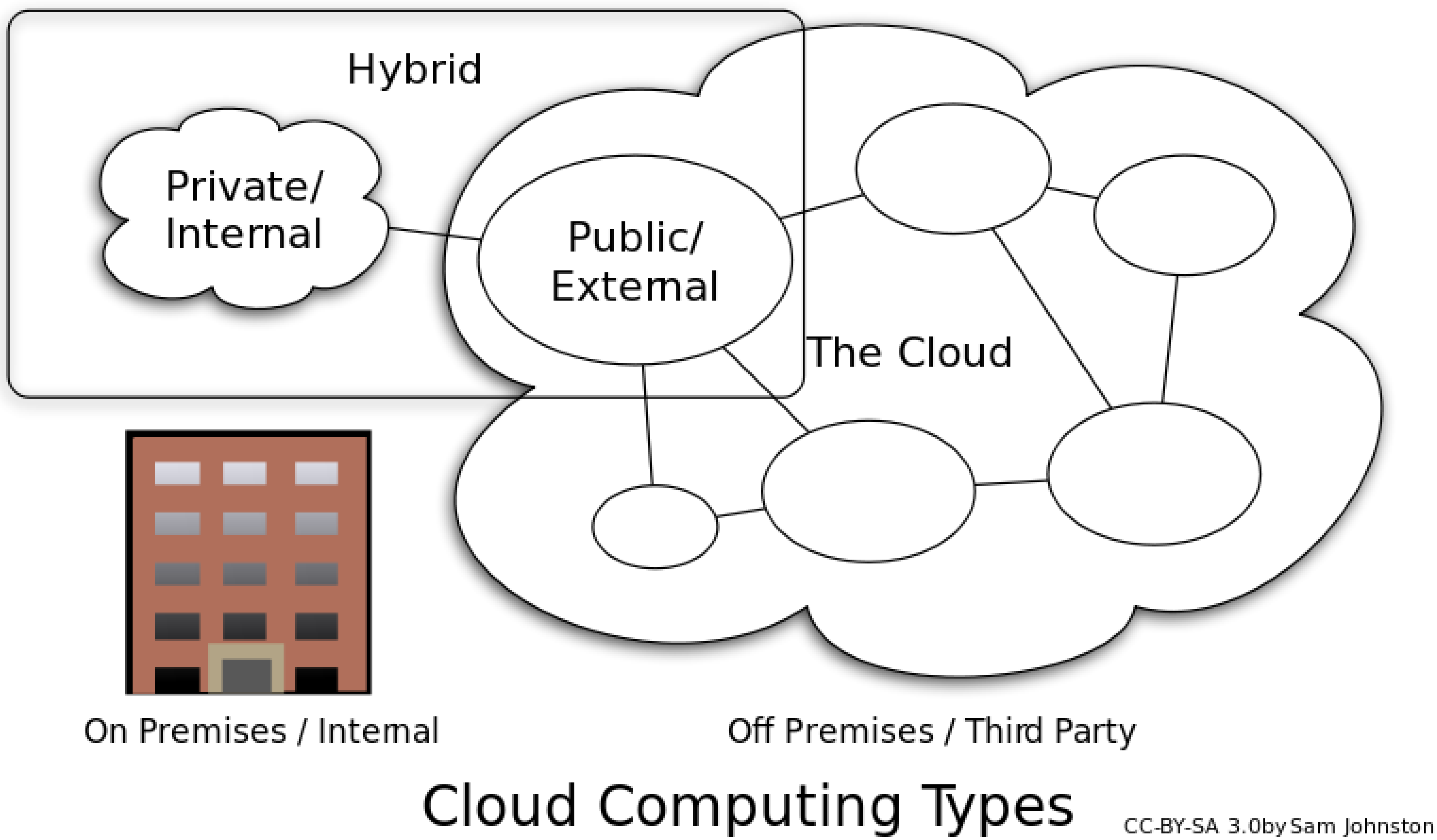
DOCUMENT MANAGEMENT
NetDocuments
Questys
DocLanding
Aconex
Xythos
KnowledgeTreeLive
SpringCM

SECURITY
Ping Identity
OpenID/OAuth
Symplified
Alert Logic
AT&T Secure Email Gateway
Qualys

Cloud Computing: Deployment Model

- Private cloud
- Community cloud
- Public cloud
- Hybrid cloud

Cloud Computing: Deployment Model



Cloud Computing: Private Cloud

- Cloud infrastructure provided for an exclusive use of a single organization including multiple users (tenants)
  - For instance, UNIcloud

- Owned and managed by a single organization, third party, or a combination of the two

- Can be on premise or off premise

Cloud Computing: Community Cloud

- Cloud infrastructure provided for an exclusive use of a community of users
  - Users include organizations that have common goals (e.g., mission, security requirements, policies, and compliance requirements)
- Owned and managed by one or more organizations, third parties, or a combination of the two
- Can be on premise or off premise

Cloud Computing: Public Cloud

- Cloud infrastructure provided to the public
- Owned and managed by a single business, academic, government organization, or a combination
- On premise for cloud providers, off premise for users

Cloud Computing: Hybrid Cloud

- Cloud infrastructure is a combination of two or more cloud infrastructure (private, community, or public)
- Each cloud infrastructure remains separated, but are composed using standard or proprietary technologies
- Allow data and application portability
  - Cloud bursting for load balancing between clouds: "**Cloud bursting** is an application deployment model in which an application runs in a private **cloud** or data center and **bursts** into a public **cloud** when the demand for computing capacity spikes."

Examples

Offer

Huge datacenter
Scalable software infrastructure
System knowledge

- Ingredients: CPU, bandwidth, storage
- Illusion of infinite resources
- No need to least buy/use

Cloud Computing

Cloudware - Cloud Computing Without Compromise

MOSSO the hosting cloud

"Why do it yourself if you can pay someone to do it for you?"

Amazon Elastic Compute Cloud (Amazon EC2) - *Beta*

VERIO An NTT Communications Company

Joyent

b-hive

GOGRID beta A ServePath Company

Esempio: Amazon Web Services

Products | Solutions | Resources

Infrastructure Services

- Amazon Elastic Compute Cloud (Amazon EC2)
- Amazon SimpleDB
- Amazon Simple Storage Service (Amazon S3)
- Amazon CloudFront
- Amazon Simple Queue Service (Amazon SQS)
- AWS Premium Support

Payments & Billing Services

- Amazon Flexible Payments Service (Amazon FPS)
- Amazon DevPay

On-Demand Workforce

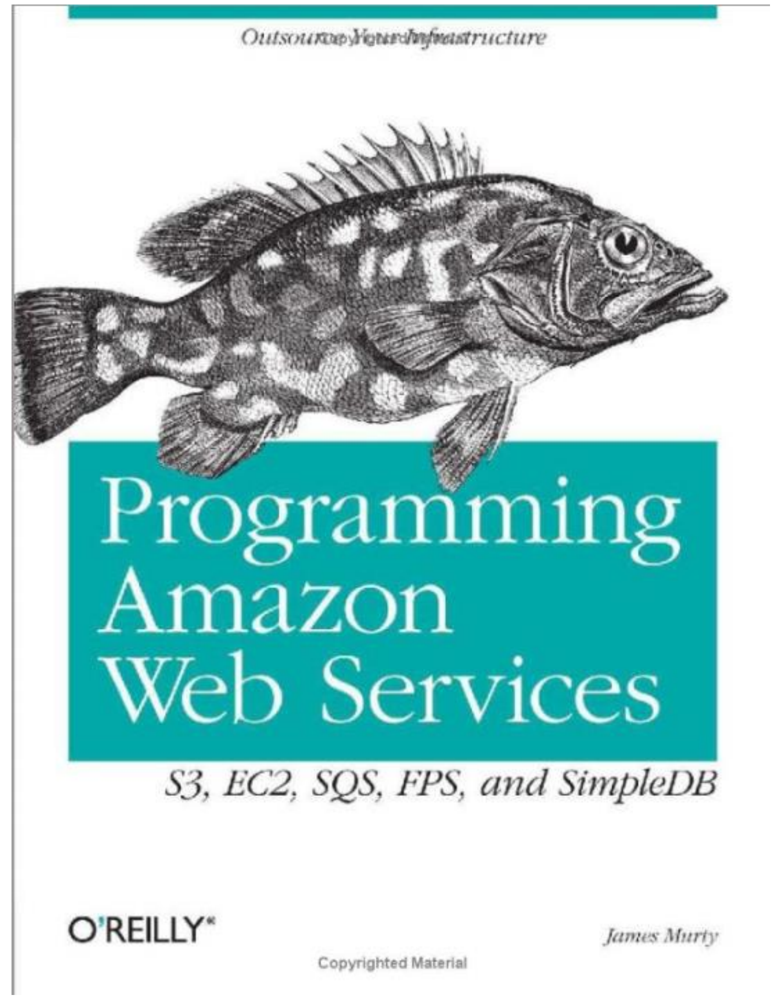
- Amazon Mechanical Turk

Alexa Web Services

- Alexa Web Information Service
- Alexa Top Sites
- Alexa Site Thumbnail

Amazon Fulfillment & Associates

- Amazon Fulfillment Web Service (Amazon FWS)
- Amazon Associates Web Service



Example: Amazon Web Services

- Elastic Compute Cloud (EC2)
  - Hourly rent of virtual machine
  - Charge = VM instance/hour
  - Additional charges for bandwidth from/to instances

- Simple Storage Service (S3)
  - Object storage
  - Charge = GB/month
  - Additional charges for bandwidth from/to storage

Amazon EC2

- Amazon EC2 instances
  - https://aws.amazon.com/it/ec2/instance-types/

A1 T3 T3a T2 M6g M5 M5a M5n M4

M5 instances are the latest generation of General Purpose Instances powered by Intel Xeon® Platinum 8175 processors. This family provides a balance of compute, memory, and network resources, and is a good choice for many applications.

Features:

- Up to 3.1 GHz Intel Xeon® Platinum 8175 processors with new Intel Advanced Vector Extension (AVX-512) instruction set
- New larger instance size, m5.24xlarge, offering 96 vCPUs and 384 GiB of memory
- Up to 25 Gbps network bandwidth using Enhanced Networking
- Requires HVM AMIs that include drivers for ENA and NVMe
- Powered by the AWS Nitro System, a combination of dedicated hardware and lightweight hypervisor
- Instance storage offered via EBS or NVMe SSDs that are physically attached to the host server
- With M5d instances, local NVMe-based SSDs are physically connected to the host server and provide block coupled to the lifetime of the M5 instance
- New 8xlarge and 16xlarge sizes now available.

Instance Size	vCPU	Memory (GiB)	Instance Storage (GiB)	Network Bandwidth (Gbps)	EBS Bandwidth (Mbps)
m5.large	2	8	EBS-Only	Up to 10	Up to 4,750
m5.xlarge	4	16	EBS-Only	Up to 10	Up to 4,750
m5.2xlarge	8	32	EBS-Only	Up to 10	Up to 4,750
m5.4xlarge	16	64	EBS-Only	Up to 10	4,750
m5.8xlarge	32	128	EBS Only	10	6,800
m5.12xlarge	48	192	EBS-Only	10	9,500
m5.16xlarge	64	256	EBS Only	20	13,600
m5.24xlarge	96	384	EBS-Only	25	19,000
m5.metal	96*	384	EBS-Only	25	19,000
m5d.large	2	8	1 x 75 NVMe SSD	Up to 10	Up to 4,750
m5d.xlarge	4	16	1 x 150 NVMe SSD	Up to 10	Up to 4,750
m5d.2xlarge	8	32	1 x 300 NVMe SSD	Up to 10	Up to 4,750
m5d.4xlarge	16	64	2 x 300 NVMe SSD	Up to 10	4,750

Amazon EC2

- Pricing
  - https://aws.amazon.com/it/ec2/pricing/

Linux | RHEL | SLES | Windows | Windows with SQL Standard | Windows with SQL Web | Windows with SQL Enterprise | Linux with SQL Standard | Linux with SQL Web | Linux with SQL Enterprise

Region: US East (Ohio)

	vCPU	ECU	Memory (GiB)	Instance Storage (GB)	Linux/UNIX Usage
General Purpose - Current Generation					
a1.medium	1	N/A	2 GiB	EBS Only	$0.0255 per Hour
a1.large	2	N/A	4 GiB	EBS Only	$0.051 per Hour
a1.xlarge	4	N/A	8 GiB	EBS Only	$0.102 per Hour
a1.2xlarge	8	N/A	16 GiB	EBS Only	$0.204 per Hour
a1.4xlarge	16	N/A	32 GiB	EBS Only	$0.408 per Hour
a1.metal	16	N/A	32 GiB	EBS Only	$0.408 per Hour
t3.nano	2	Variable	0.5 GiB	EBS Only	$0.0052 per Hour
t3.micro	2	Variable	1 GiB	EBS Only	$0.0104 per Hour

Region: US East (Ohio)

Data Transfer IN To Amazon EC2 From Internet	
All data transfer in	$0.00 per GB
Data Transfer OUT From Amazon EC2 To Internet	
Up to 1 GB / Month	$0.00 per GB
Next 9.999 TB / Month	$0.09 per GB
Next 40 TB / Month	$0.085 per GB
Next 100 TB / Month	$0.07 per GB
Greater than 150 TB / Month	$0.05 per GB

Cloud computing models

- Infrastructure as a Service (IaaS)
  - Machine cycles for client applications
  - Examples: Amazon EC2, GoGrid, AppNexus
- Platform as a Service (PaaS)
  - APIs for application development
  - Examples: Google App Engine, CloudFoudry, Heroku
- Software as a Service (SaaS)
  - Execution of turn-key applications on behalf of the client
  - Examples: Gmail, GoogleDocs, Dropbox
- Others: DaaS, NaaS, IPaaS, SOA-aaS…

SaaS

Software as a Service

PaaS

Platform as a Service

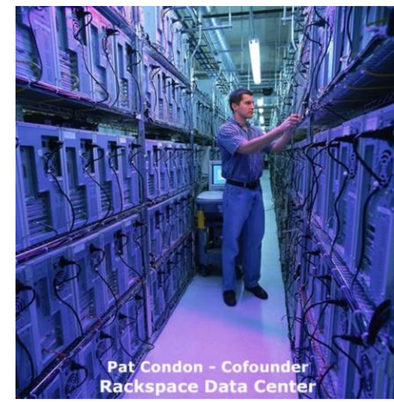
IaaS

Infrastructure as a Service

IaaS
Infrastructure as a Service

Supply of devices for virtual compute

- Access to administration functions
  - Mix of operating systems
  - Access control
  - Perimeter control
  - Routing
  - Load balancing



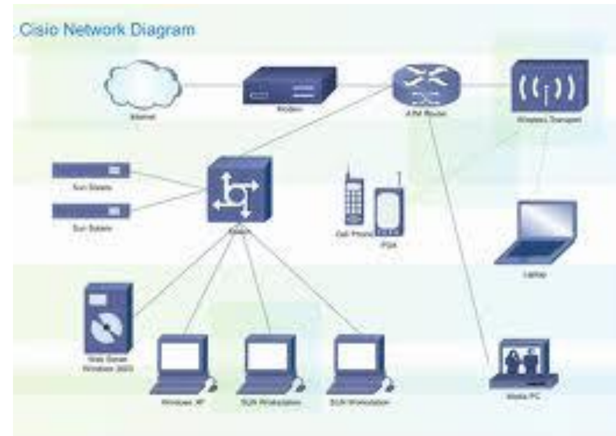
IaaS

Virtual networks

- Virtual networks are (all or in part) simulated on physical servers
- Protocol integrity not guaranteed
- Interface between physical and virtual networks is not standard

IaaS

WHAT YOU SEE...



ISN'T WHAT YOU GET...

Advantages

- Pay per use
- Modular scalability
- Security
- Reliability
- APIs

IaaS

Examples

- Problem: recurrently execute a batch job without owning a proper machine for execution
  - Solution: use a virtual machine on Amazon EC2
- Problem: activate a temporary Web site (a few days)
  - Solution: use a virtual web server on FlexiScale
- Problem: provide employee with a remote storage without sufficient storage in the enterprise
  - Solution: Amazon S3

IaaS

PaaS
Platform as a Service

Supply of virtual “horizontal” services

- Services deployed on demand
- No a priori estimation of amount of requests, procurement, ...
- No management overhead

PaaS

Main services

- Libraries, tools and platforms for web development
- Computing platform
- Mainly for developers

PaaS

Advantages

- Pay per use
- Modular scalability
- Security
- Reliability
- APIs

PaaS

Example

- Problem: develop a web application or a cloud service without installing the whole software stack including libreries and tools
  - Solution: usare Amazon Elastic Beanstalk, Google App Engine, Microsoft Azure…

PaaS

SaaS
Software as a Service

Supply of application software

- Mainly for PMI
- No management of hardware/software
- Access via browser

SaaS

Advantages

- Pay per use
- Modular scalability
- Security
- Reliability
- APIs

SaaS

Example

- Problem: CRM (Customer relationship management) management is too expensive
  - Solution: use a cloud version of the software, such as Salesforce.com

- Problem: the mail server is slow and unreliable
  - Solution: use a mail service on cloud, such as Hosted Exchange

SaaS

SaaS
Software as a Service

PaaS
Platform as a Service

IaaS
Infrastructure as a Service

Common characteristics

SaaS

PaaS

IaaS

- Remotely hosted: data and services are on a remote infrastructure
- Ubiquitous: data and services available everywhere
- Commodified: supply model is similar to utilities – electricity, gas

Other advantages

SaaS

PaaS

IaaS

- Low costs of maintenance
- Management of peaks of traffic/requests
- Fast application roll-out

More Refined Categorization

- Storage-as-a-service
- Database-as-a-service
- Information-as-a-service
- Process-as-a-service
- Application-as-a-service
- Platform-as-a-service
- Integration-as-a-service
- Security-as-a-service
- Management-as-a-service
- Governance-as-a-service
- Testing-as-a-service
- Infrastructure-as-a-service

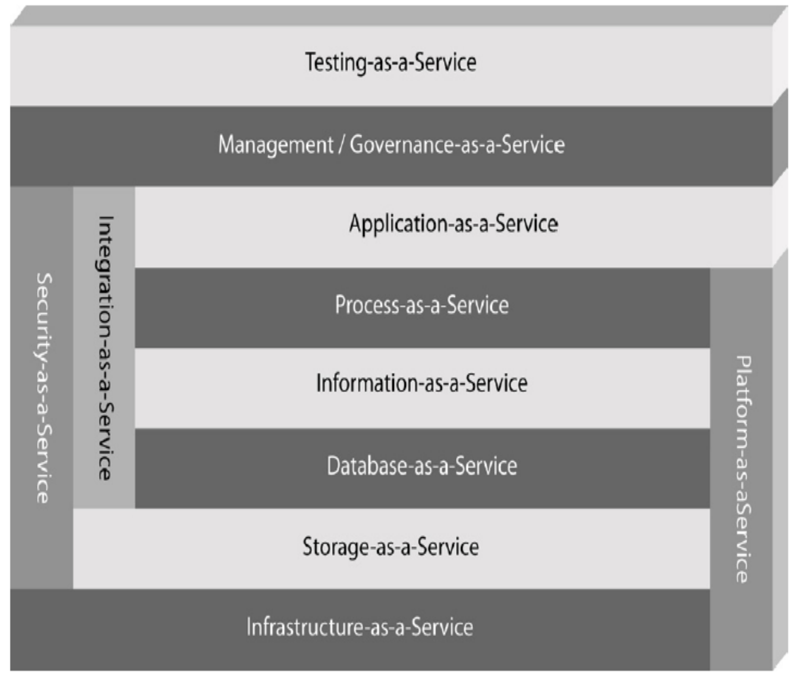


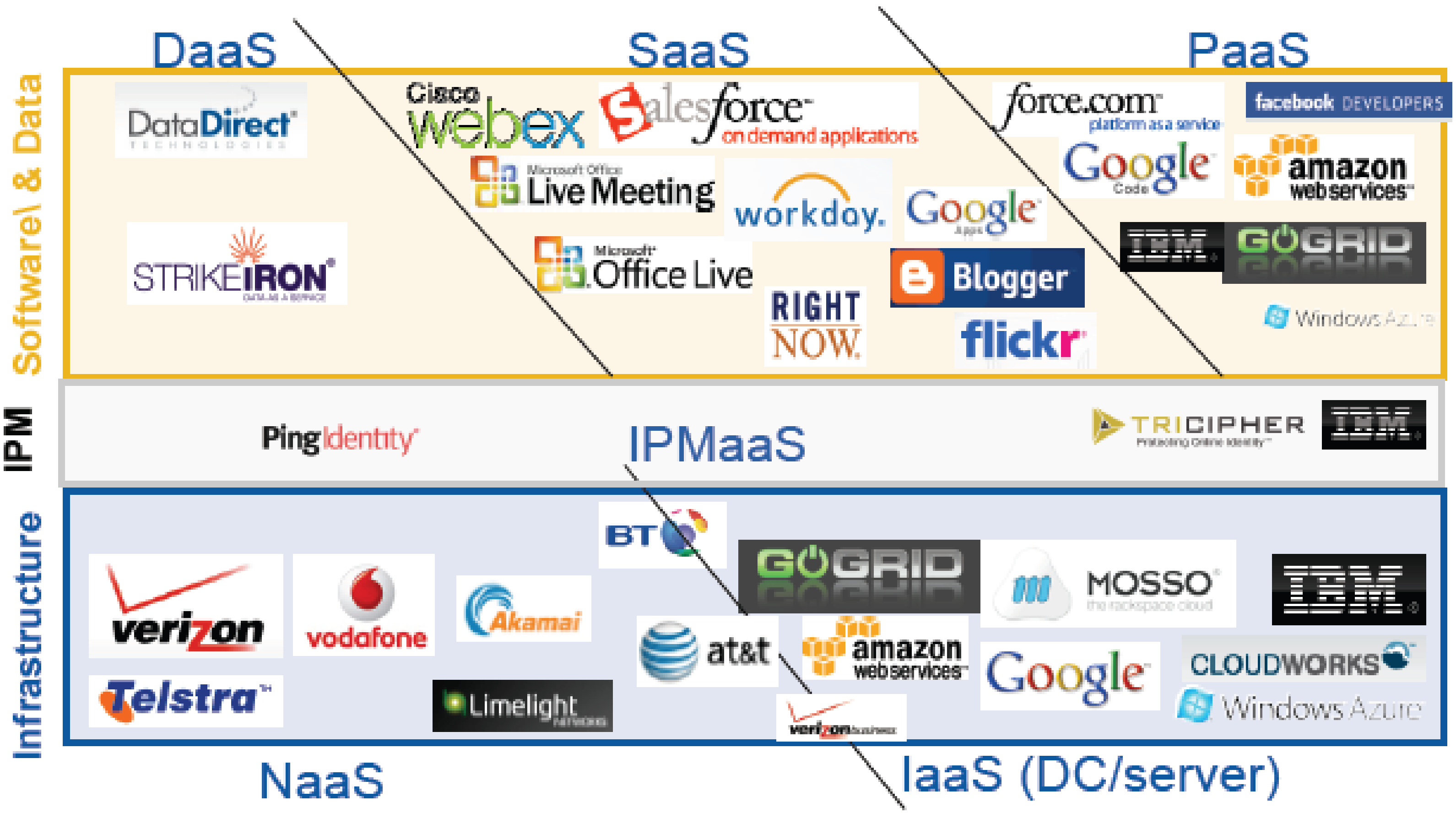
Figure 1: The patterns or categories of cloud computing providers allow you to use a discrete set of services within your architecture.

InfoWorld Cloud Computing Deep Dive

Everything as a Service

- Utility computing = Infrastructure as a Service (IaaS)
  - Why buy machines when you can rent cycles?
  - Examples: Amazon's EC2, Rackspace
- Platform as a Service (PaaS)
  - Give me nice API and take care of the maintenance, upgrades, …
  - Example: Google App Engine
- Software as a Service (SaaS)
  - Just run it for me!
  - Example: Gmail, Salesforce

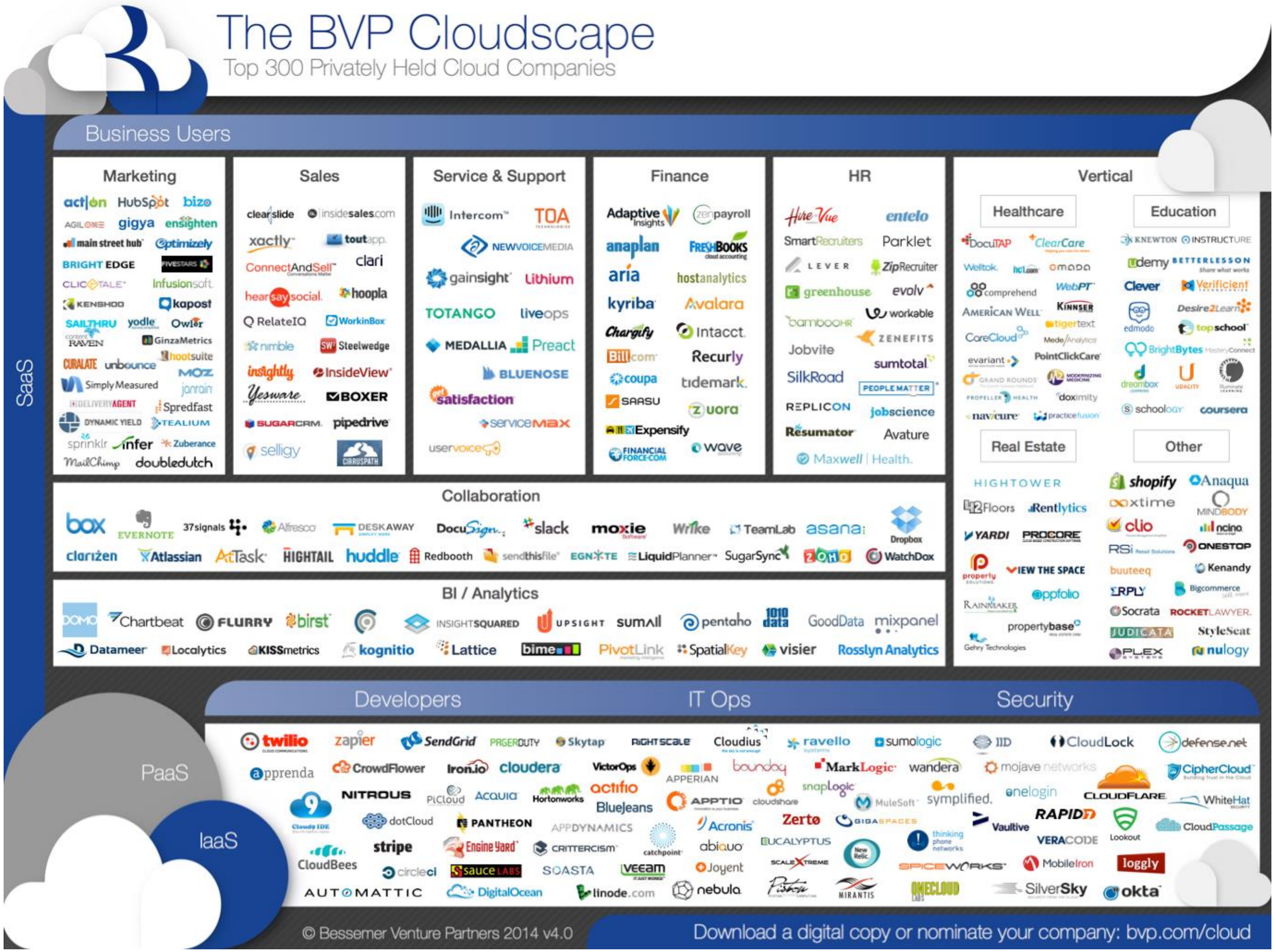
Overview


DaaS
SaaS
PaaS
Software\ & Data
DataDirect
STRIKEIRON
Cisco webex
salesforce on demand applications
Live Meeting
workday.
Office Live
RIGHT NOW.
Google
Blogger
flickr
force.com platform as a service
facebook DEVELOPERS
Google Code
amazon webservices
IBM
GOGRID
Windows Azure
IPM
PingIdentity
IPMaaS
TRICIPHER
IBM
Infrastructure
verizon
vodafone
Akamai
Telstra
Limelight
BT
at&t
GOGRID
amazon webservices
verizon business
MOSSO the rackspace cloud
IBM
Google
CLOUDWORKS
Windows Azure
NaaS
IaaS (DC/server)

Overview

- http://www.ashwinirath.com/cloud/

Overview


The BVP Cloudscape
Top 300 Privately Held Cloud Companies
Business Users
SaaS
Marketing
Sales
Service & Support
Finance
HR
Vertical
Healthcare
Education
Real Estate
Other
Collaboration
BI / Analytics
Developers
IT Ops
Security
PaaS
IaaS
act-on
HubSpot
gigya
ensighten
Optimizely
BRIGHTEDGE
Infusionsoft
kapost
yodle
hootsuite
unbounce
MOZ
Simply Measured
Spredfast
TEALIUM
sprinklr
infer
Zuberance
MailChimp
doubledutch
clearslide
insidesales.com
xactly
toutapp
ConnectAndSell
clari
hearsaysocial
hoopla
RelateIQ
nimble
Steelwedge
insightly
InsideView
Yesware
BOXER
SUGARCRM
pipedrive
selligy
Intercom
TOA
NEWVOICEMEDIA
gainsight
Lithium
TOTANGO
liveops
MEDALLIA
Preact
BLUENOSE
Satisfaction
servicemax
uservoice
Adaptive Insights
zenpayroll
anaplan
FRESHBOOKS
aria
hostanalytics
kyriba
Avalara
Chargify
Intacct
Bill.com
Recurly
coupa
tidemark
SAASU
zuora
Expensify
FINANCIALFORCE.COM
wave
Hire Vue
entelo
SmartRecruiters
Parklet
LEVER
ZipRecruiter
greenhouse
evolv
bambooHR
workable
ZENEFITS
Jobvite
sumtotal
SilkRoad
PEOPLEMATTER
REPLICON
jobscience
Resumator
Avature
Maxwell Health
DocuTAP
ClearCare
KNEWTON
INSTRUCTURE
udemy
BETTERLESSON
comprehend
Clever
Verificient
AMERICAN WELL
Kinnser
Desire2Learn
CoreCloud
edmodo
topschool
evariant
PointClickCare
BrightBytes
doximity
schoology
coursera
naviHealth
practicefusion
HIGHTOWER
shopify
Anaqua
Floors
Rentlytics
xtime
MINDBODY
YARDI
PROCORE
clio
ncino
RSI
ONESTOP
VIEW THE SPACE
Kenandy
appfolio
ERPLY
Bigcommerce
RAINMAKER
propertybase
Socrata
ROCKETLAWYER
JUDICATA
StyleSeat
PLEX
nulogy
box
EVERNOTE
37signals
Alfresco
DESKAWAY
DocuSign
slack
moxie
Wrike
TeamLab
asana
Dropbox
clarizen
Atlassian
AtTask
HIGHTAIL
huddle
Redbooth
sendthisfile
EGNYTE
LiquidPlanner
SugarSync
zoho
WatchDox
DOMO
Chartbeat
FLURRY
birst
INSIGHTSQUARED
UPSIGHT
sumAll
pentaho
1010data
GoodData
mixpanel
Datameer
Localytics
KISSmetrics
kognitio
Lattice
bime
PivotLink
SpatialKey
visier
Rosslyn Analytics
twilio
zapier
SendGrid
PAGERDUTY
Skytap
RIGHTSCALE
Cloudius
ravello
sumologic
CloudLock
defense.net
apprenda
CrowdFlower
Iron.io
cloudera
VictorOps
APPERIAN
boundary
MarkLogic
wandera
mojave networks
CipherCloud
NITROUS
PiCloud
ACQUIA
Hortonworks
actifio
BlueJeans
APPTIO
cloudshare
snapLogic
MuleSoft
symplified
onelogin
CLOUDFLARE
WhiteHat
dotCloud
PANTHEON
APPDYNAMICS
Acronis
Zerto
GIGASPACES
Vaultive
RAPID7
CloudPassage
stripe
Engine Yard
CRITTERCISM
catchpoint
abiquo
EUCALYPTUS
VERACODE
Lookout
CloudBees
circleci
sauce LABS
SOASTA
veeam
Joyent
SCALEXTREME
SPICEWORKS
MobileIron
loggly
AUTOMATTIC
DigitalOcean
linode.com
nebula
MIRANTIS
ONECLOUD
SilverSky
okta
© Bessemer Venture Partners 2014 v4.0
Download a digital copy or nominate your company: bvp.com/cloud

The promise of cloud computing

- Full network reliability
- Zero network latency
- Infinite bandwidth
- Secure network
- No topology change
- Centralized administration
- Zero transport cost
- Homogeneous network and system

Open challenges

- Security
- Performance monitoring
- Consistent and robust service abstraction
- Meta scheduling
- Energy-efficient load balancing
- Scale management
- SLA & QoS architectures
- Interoperability and portability
- Green IT

Programmable cloud

- Developers see cloud as a set of semi-finished services that can be used to develop applications and processes

- Application engine can be inside or outside cloud borders

- Cloud provider provides an API for different languages and programming environments

Example: Google App Engine

- Google App Engine manages HTTP(S) requests only
  - RPC style: request in, processing, response out

- Application-level configuration: tend to zero

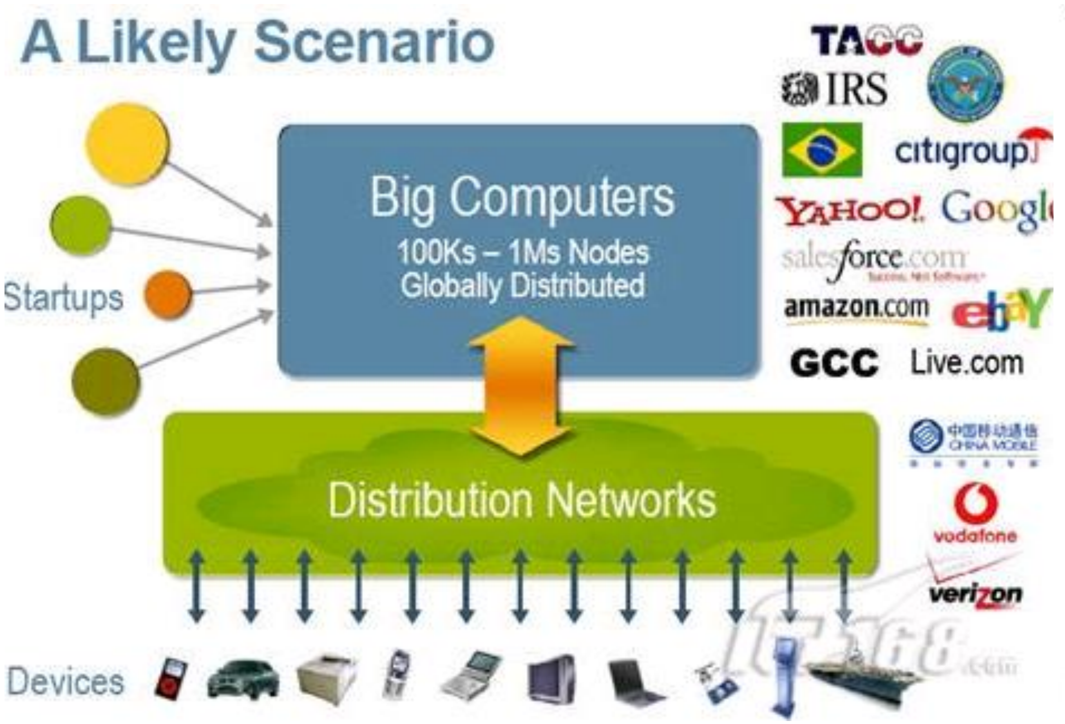
- High scalability
  - No fixed limits to number of applications, requests/sec, storage size
  - Simple APIs

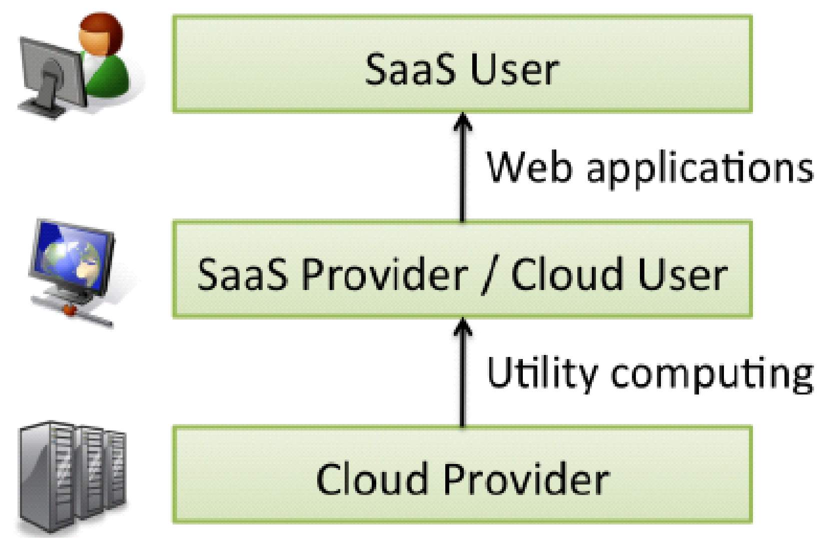
Cloud as a supercomputer

- Task with
  - Fixed frequency (block resources is useless)
  - Huge amount of data
  - Need to be resilient to faults and failures

- Examples
  - DNA sequencing
  - Analysis of revisions/editing on Wikipedia
  - Analysis of Web pages using a crawler (Google)
  - Image analysis

Multi-level distribution





Conclusions

- History of distributed systems

- Cloud computing
  - Service model
  - Deployment model

Lesson 2.2: Migration and Cloudonomics

Claudio Ardagna – Università degli Studi di Milano

Cloud and Distributed Computing

Is the cloud good for your business?

- The analysis of the characteristics and advantages of the cloud with respect to on-premise systems can tell whether the cloud is good for our business

- Before taking a decision
  - Where is headed our business?
  - What are we trying to achieve?
  - Which are the goals of our business?
  - The replies to these questions give an **end goal**
    - Without end goals and priorities it is difficult to understand whether the cloud is good or not

End goals

Strategic

Financial

Customer

End goals

Strategic

Financial

Customer

Key drivers for improvement

- Know where the drawbacks are and additional business functionalities are needed
- Identify the challenges to be faces
- Define the means to address the challenges

Performance assessment

- These activities provide a performance assessment of a business
- This assessment provides a clear idea on the fact that the cloud could help a specific business
- Upon identifying goals, targets, and key results, we need to understand whether the cloud can support them

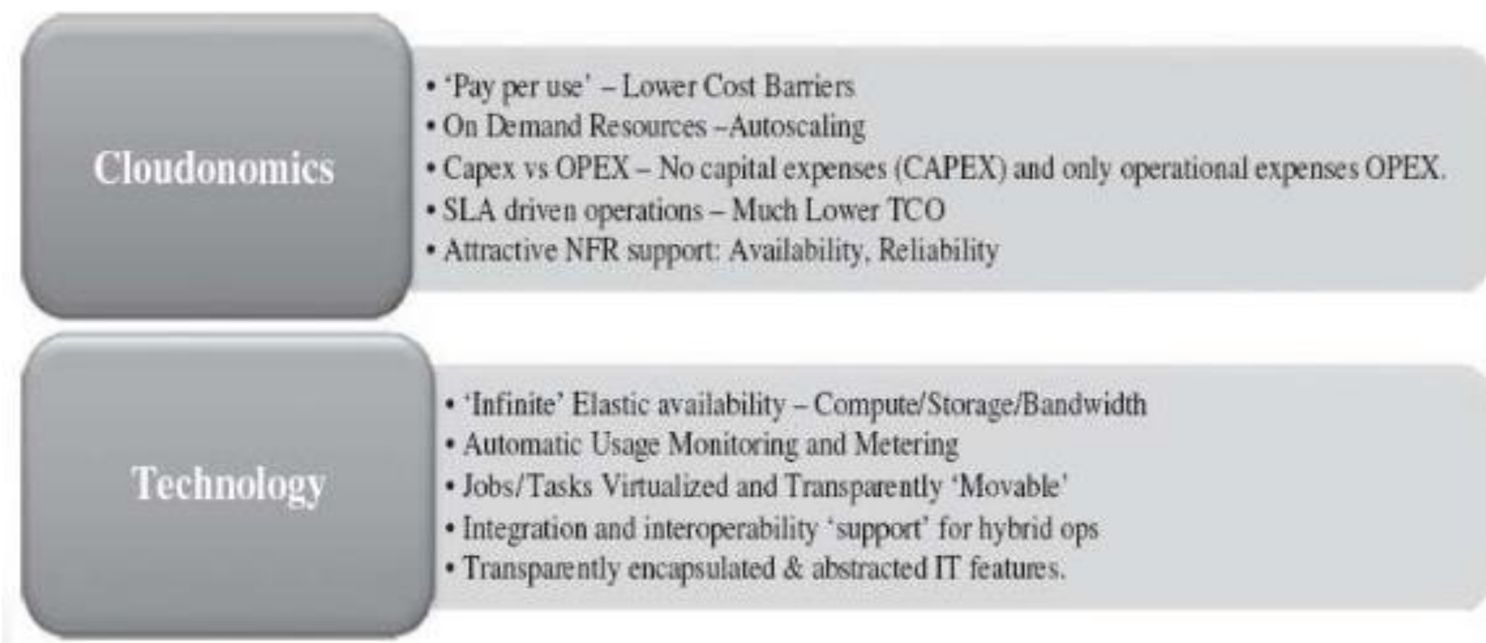
Cloud Migration

Key questions

- When and how we should migrate an application to the cloud?
- Which part or component of an IT application can/must be migrated to the cloud and which ones not?
- Which kind of customers will benefit from IT migration to the cloud? Which benefits for business?

The promise of cloud computing

- Reduced complexity of systems and their management
- Easy and uniform cloud abstractions
- Cloudonomics: cost savings and economic aspects introduced by the cloud and related trade-offs
  - E.g., seasonal IT load



The promise of cloud computing

- Full network reliability
- Zero network latency
- Infinite bandwidth
- Secure network
- No topology change
- Centralized administration
- Zero transport cost
- Homogeneous network and system

The promise of cloud computing

- Security
- Performance monitoring
- Consistent and robust service abstraction
- Meta scheduling
- Energy-efficient load balancing
- Scale management
- SLA & QoS architectures
- Interoperability and portability
- Green IT

Why migrate

- On-demand resourcing
- Scalability
- Economy of scale
- Flexibility and elasticity
- Growth
- Utility based metering
- Shared infrastructure
- High availability
- Security

On-demand resourcing

On premise

- Adding additional resources, computation, storage, network, requires a long purchasing process
  - Contacting the supplier
  - Obtaining a price
  - Ordering the hardware
  - Installing, configuring, cabling the hardware in a data center
- Process requiring weeks, even though days/hours can be too much in some cases
- Risk of losing customers

Cloud

- An allocation process almost immediate
- Can allocate where and when is needed
  - If I have a CPU usage spike I can power on a new server in seconds
- On-premises issues are solved with an almost immediate access to resources that are being selected and configuring by choosing through a series of options

Scalability

On premise

- Not supported
- Scalability in on-premises systems requires significant financial resources and space in the data center

Cloud

- Scalability in the cloud offers scaling up and scaling out resources depending on requirements and requests of services and applications
  - Scaling up and scaling down allow modifying the power of an instance
  - Scaling in and scaling out add or remove the number of instances
- Scalability is possible because the cloud supports the concept of on-demand resourcing

Economy of scale

On premise

- Traditional hosting costs are much higher for unit of resource

Cloud

- Resource sharing among tenants and the huge amount of resources provided by a public cloud allow offering computing, storage and network at very low prices
- The more you buy, the cheaper it becomes
- Cloud resources are much more economical than the same ones on on-premise infrastructure

Flexibility and elasticity

On premise

- Difficult to manage spikes
- Resources must be planned in advance to be able to handle spikes
- If planning is wrong then problems with customers and loss of reputation

Cloud

- Cloud computing offers huge flexibility and elasticity
- It's possible to choose how many resources without establishing the needed power in advance
- The infrastructure can adapt, like an accordion

Growth

On premise

- In traditional systems the business growth could require
  - Buying a new office
  - Hiring new workers
  - Waiting for months

Cloud

- Almost immediate support for every growth profile
- Limits to the grow are reduced compared to classical environments

Utility based metering

On premise

- Servers are kept on 24/7/365
- Electricity and cooling costs, wear out...

Cloud

- Pay-per-use
- Pay for what I use and nothing more
- Turning off servers when they are not needed

Shared infrastructure

On premise

- Not supported
- Each user has its own dedicated hardware

Cloud

- Hosts are virtualized
- Different tenants share the same resources
  - Reducing hardware, cooling, space…
- It's possible to have dedicated hosts or instances
  - Dedicated instances, same host, the instance is executed on a dedicated hardware (specific core)
  - Dedicated host, all dedicated to a tenant
    - Allows deciding where instances should be executed, managing licensing…

High availability

On premise

- Supporting on-premises high availability requires higher costs and competence often not affordable to medium sized companies
- Often medium sized companies do not have replicated sites and do not have advanced disaster recovery programs

Cloud

- Native support for replicating resources and services
- Replicas among several zones and geographical regions
- Important to understand which part of the resilience depend on the vendor and which part on the user

Security

On premise

- Mostly security
- Almost no assurance, compliance, certification

Cloud

- Provides infrastructures often already certificated and compliant to standard
- For example: PCI DSS, ISO, HIPAA, SOX

Migration and cloudonomics

Migration and cloudonomics

- Cloudonomics
  - Economic rationale for using cloud technologies
  - Important to increment company ROI

- Dilemma for IT manager, SW architect, decision-maker
  - At which costs I migrate to the cloud?
  - The cloud can satisfy company strategies?
  - Which is the Total Cost of Ownership with respect to private data center solutions?

Migration

- 5 level of migration
  - application
  - code
  - design
  - architecture
  - usage
- Migration levels are applied to the different IaaS, PaaS, SaaS levels
  - specific use cases only for IaaS and PaaS
  - only one for SaaS (using applications in the cloud)
- $P \rightarrow P'_C + P'_l \rightarrow P'_{OFC} + P'_l$
  - P = application
  - P'_C = application after cloud migration (hybrid cloud)
  - P'_l = part of the application executed locally
  - P'_{OFC} = part of the application optimized for the cloud

Migration: 7-steps model

- A structured approach migration-oriented
  - Cloud Migration Assessment
  - Isolate the dependencies
  - Map the messaging & the environment
  - Re-architect and implement the lost functionalities
  - Leverage cloud functionalities & features
  - Test the migration
  - Iterate and optimize

Cloud Economics 101

- The cloud is compensated though
  - Economies of scales on physical resources
  - Virtualization = better usage of physical resources = low costs
  - Update are automated
  - Application roll-out is faster = chance of revenue

SaaS

PaaS

IaaS

Cloudonomics

- Depending on cutting costs in terms of
  - IT capital expense (CapEx): costs for providing a service (e.g., purchasing a printer)
  - IT operational expense (OpEx): costs for allowing the working of the service (e.g., toner, papers, electricity)
  - With the cloud CapEx are moved toward OpEx, reducing risks by moving them to cloud provider
- Long term benefits
  - Seasonal offload and highly variable (opportunistic migration)
  - Total offload (full migration to the cloud)
- The migration is profitable if medium costs are lower in the cloud and migration costs do not impact on profits

Cloudonomics

- Other cloudonomics factors
  - Licenses
  - SLA compliance
  - Costs for cloud service
    - Elastic storage
    - Elastic compute
    - Elastic bandwidth

The Law of Cloudonomics

- Joe Wienman di AT&T Global Services defines the 10 laws of Cloudonomics
  1. Utility services cost less even though they cost more
  2. On-demand trumps forecasting
  3. The peak of the sum is never greater than the sum of the peaks
  4. Aggregate demand is smoother than individual
  5. Average unit costs are reduced by distributing fixed costs over more units of output
  6. Superiority in numbers is the most important factor in the result of a combat (Clausewitz)
  7. Space-time is a continuum (Einstein/Minkowski)
  8. Dispersion is the inverse square of latency
  9. Don’t put all your eggs in one basket
  10. An object at rest tends to stay at rest (Newton)

The Law of Cloudonomics

- Law 1: Utility services cost less even though they cost more
  - The cost for a time unit is higher
  - On demand access to utility reduces the total costs
- Law 2: On-demand trumps forecasting
  - Capability of allocation and de-allocazione almost immediate
  - Forecasts are often wrong, being able of reacting immediately allows a huge profit
- Law 3: The peak of the sum is never greater than the sum of the peaks
  - Companies install resources to handle spikes
  - The amount of resources is the sum of the spikes
  - The cloud installs less resources (resource reallocation)

The Law of Cloudonomics

- Law 4: Aggregate demand is smoother than individual
  - The aggregation of requests of different customers tends to reduce variations
  - Cloud obtains the best efficiency
- Law 5: Average unit costs are reduced by distributing fixed costs over more units of output
  - Fixed costs are better distributed
  - Cost per unit is decreased in several contexts: storage, bandwidth…
- Law 6: Superiority in numbers is the most important factor in the result of a combat (Clausewitz)
  - Typical military strategy
  - Numerical superiority can win battles
  - A DoS attack is harder in the cloud

The Law of Cloudonomics

- Law 7: Space-time is a continuum (Einstein/Minkowski)
  - Advantage comes from a faster decision-making that can fastly react to variations in the environment
  - Cloud scalability allows a faster decision-making
- Law 8: Dispersion is the inverse square of latency
  - Reducing latency is fundamental for many applications
  - Reducing latency of an half requires 4 times more computational nodes
  - Easier in the cloud

The Law of Cloudonomics

- Law 9: Don’t put all your eggs in one basket
  - Better reliability
  - Replica on distributed data centers
- Law 10: An object at rest tends to stay at rest (Newton)
  - Company data centers are installed in company sites
  - Cloud sites where they are installed where it is more advantageous
    - E.g., closer to a backbone network with low cost access to energy, cooling...

Costi del Cloud Computing

- Cloud supports scalability and elasticity
  - Result: you pay only what you consume

- But how much does the cloud really cost?
  - How much does X cost in X years when there is a more or less constant growth of usage?
  - If I expect to grow Y% every year, how much does it cost if I instead grow Z% every year?
  - If I launch a new product with traffic spikes, how much does it cost?

- We need to forecast cost trends (models of costs)
  - Modelling servers, storage, database, data transfer, support costs, elasticity and growth patterns

Cloud computing costs: growth patterns

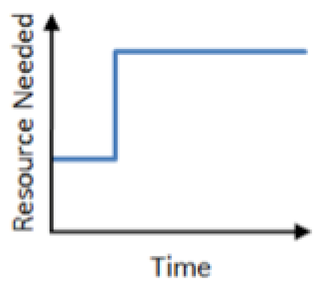
- They influence cloud costs
- Three types
  - Constant Growth
  - Seasonal Growth
  - Lifecycle Growth
- Patterns are not mutually exclusive, they can happen at the same time

Cloud computing costs: growth patterns

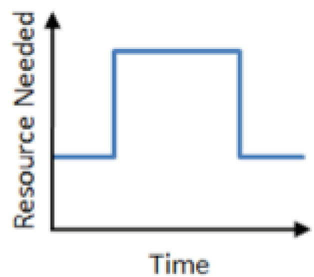
- Constant Growth
  - The number of users grows every month
  - The number of servers grows proportionally to handle requests
  - Corner case: fixed number of users, constant grow of storage utilization
- Seasonal Growth
  - Both growths and shrinkages are expected during the year
  - For example, web applications providing services satisfying seasonal customers (Christmas tickets…)
- Lifecycle Growth
  - Companies observing temporary growths during the launch of new products and commercial activities
  - Higher spikes lasting only few weeks/months and then they stabilize to lower rates

Growth patterns: permanent vs temporary

- Permanent Pattern
  - The pattern lasts: when applied the number of resources to use changes
  - For example, storage unit initially using 100 GB per month and then incremented of 5 GB each month
- Temporary Patterns
  - The pattern has a time duration: at the end of the window the resource usage comes back to the original value
  - For example, 20 web servers used to support customers each month, they double when there are sales



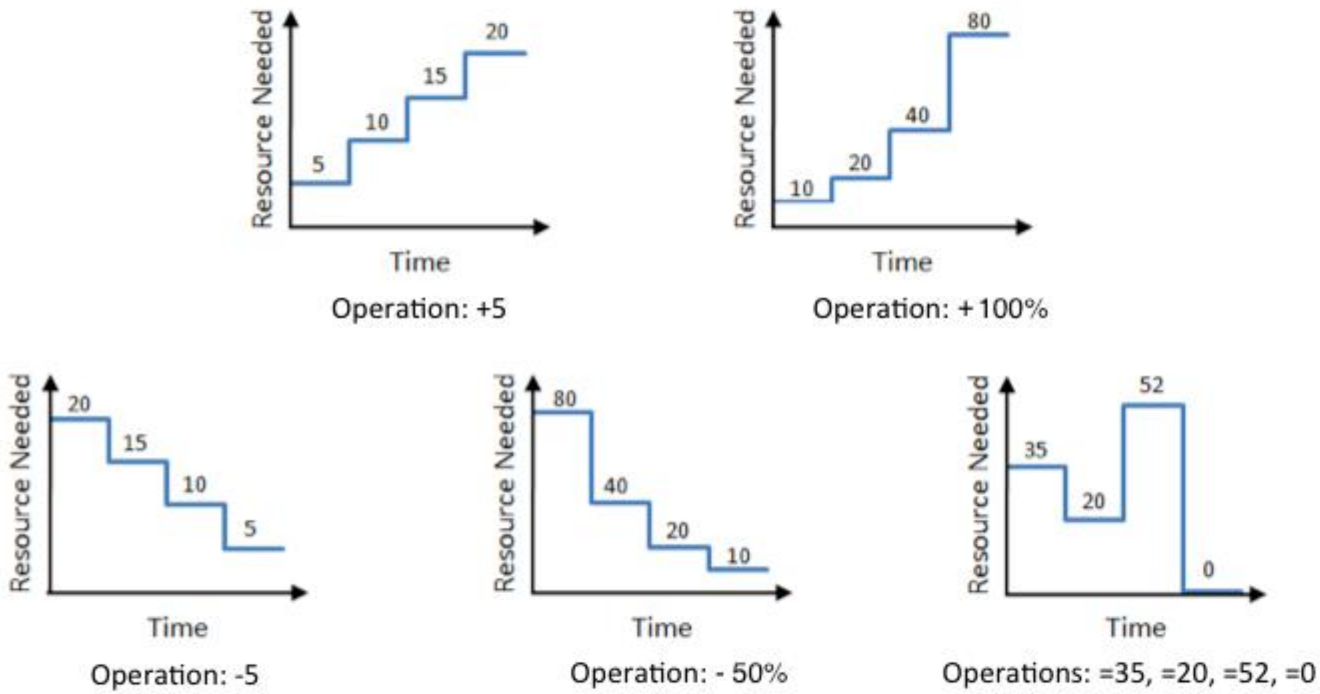
Permanent Pattern



Temporary Pattern

Growth patterns: operators

- Different growth patterns
  - Add (+), Subtract (-), Increase by (%), Decrease by (%), e Set to (=)

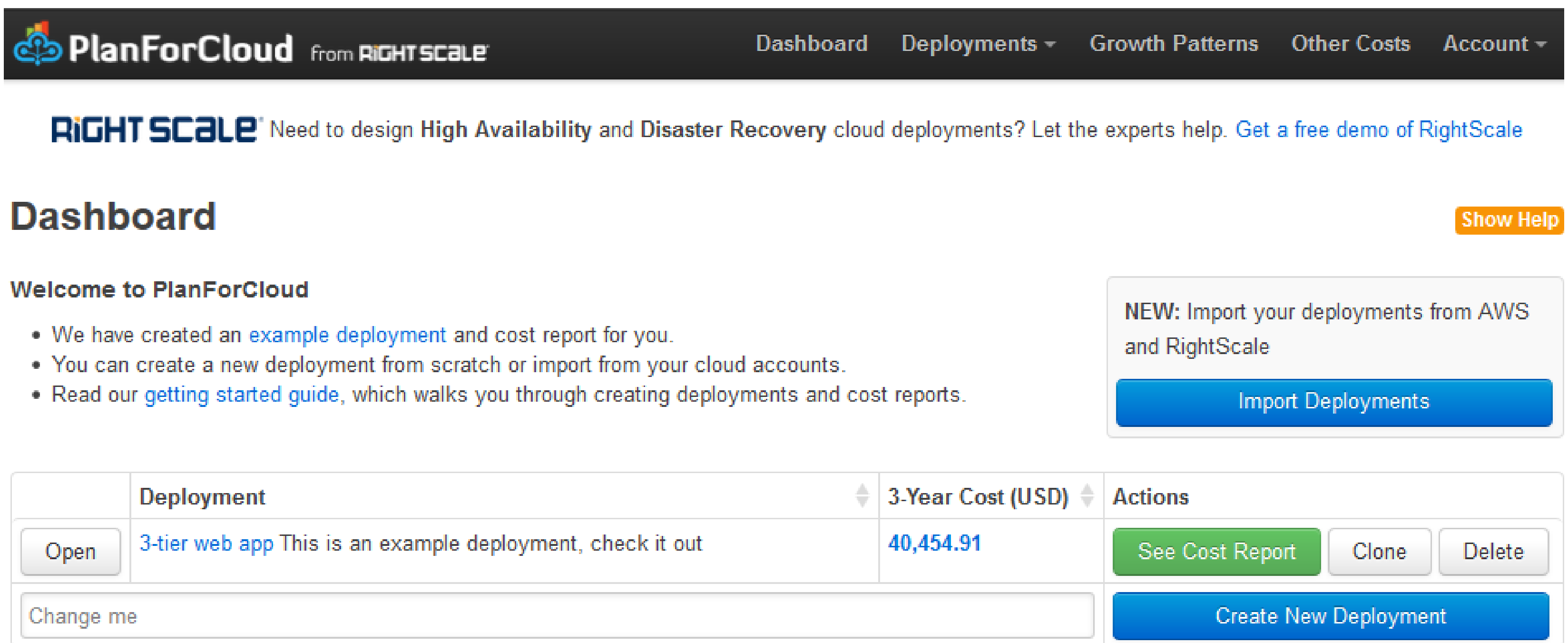


PlanforCloud: Forecasting the Cost of Your Growth

- 5 steps
  - Modelling the required cloud resources
  - Generating a report of costs
  - Creating a new pattern
  - Applying the new pattern
  - Generating a new report of costs

PlanforCloud: Forecasting the Cost of Your Growth

- Modelling the required cloud resources
  - Can be defined from scratch or imported from AWS or Rightscale deployment



PlanforCloud: Forecasting the Cost of Your Growth

- Modelling the required cloud resources
  - Adding resources

Deployment - 3-tier web app This is an example deployment, check it out

Show Help

Servers | Storage | Databases | Data Transfer | Support Plans | Other Costs

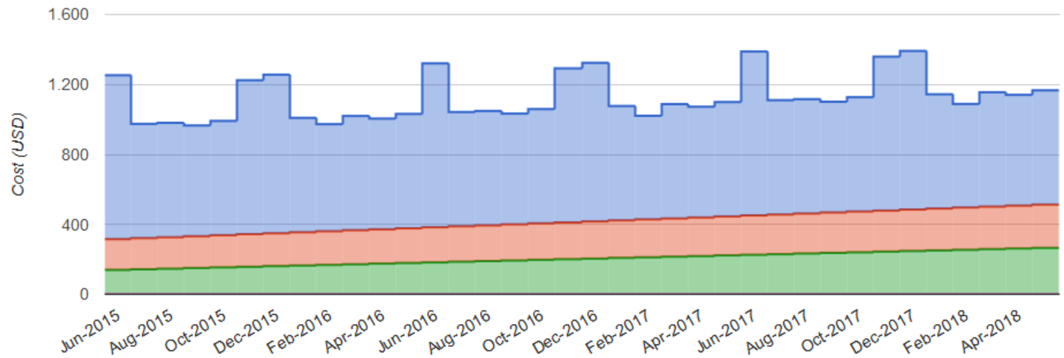
Add Server

See 3-Year Cost Report

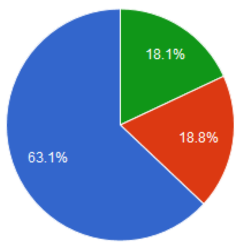
Name	Cloud	Server Type	Usage	Quantity	
Base web server	Rackspace USA	1GB server Linux - On-Demand	24hours/day	1 0 Patterns	
DR server	AWS US-West (Northern California)	m1.small Linux - Reserved 1-Year Light-Utilization	24hours/day	1 0 Patterns	
Hosting www site	Google US	n1-standard-1 Linux - On-Demand	24hours/day	1 1 Patterns	
Load balancer - HAProxy	Rackspace USA	1GB server Linux - On-Demand	24hours/day	1 0 Patterns	
Peak web server	Rackspace USA	512 server Linux - On-Demand	12hours/day	2 1 Patterns	

PlanforCloud: Forecasting the Cost of Your Growth

- Generating a report of costs



Total Cost Breakdown (USD)



Key

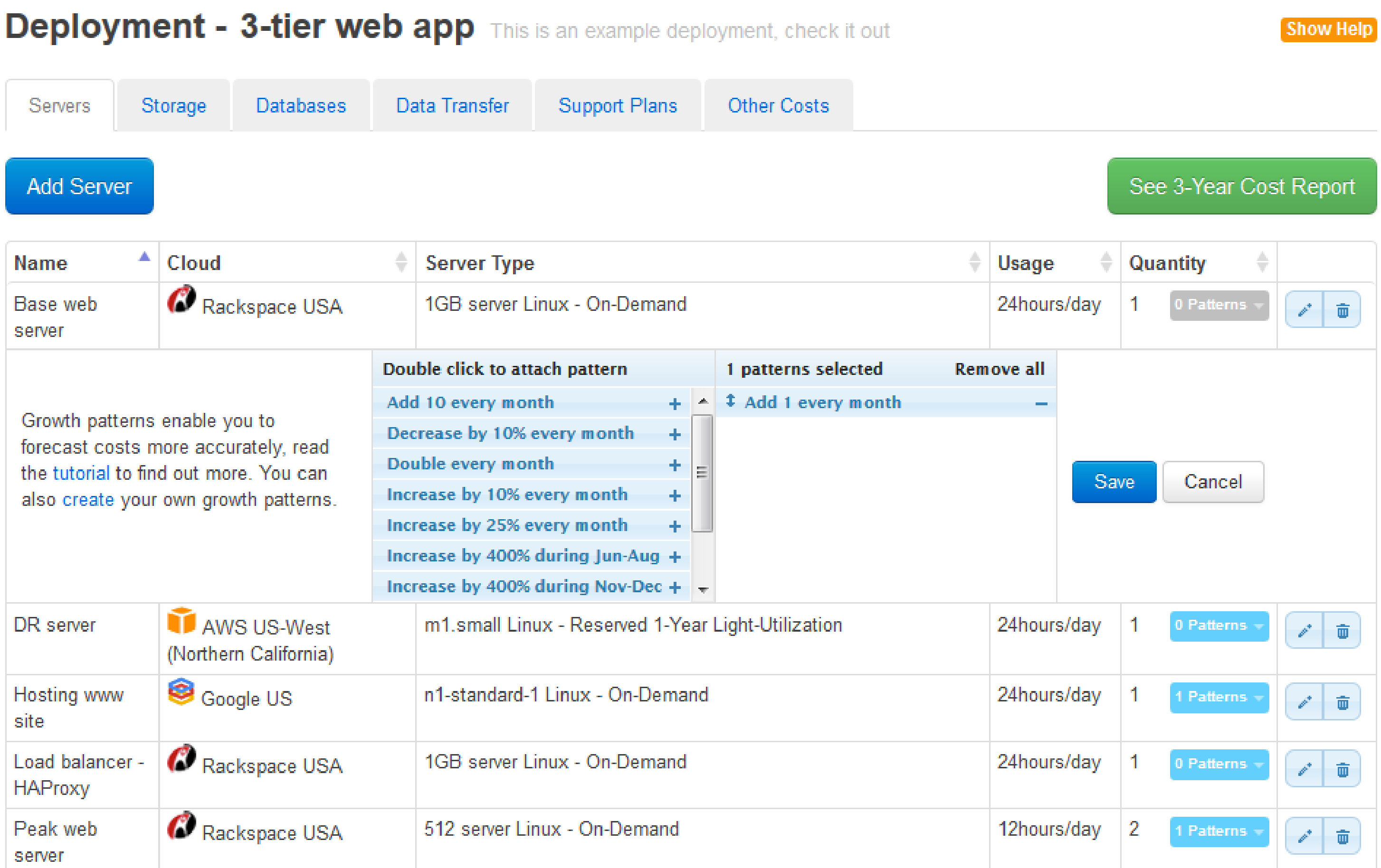
- **Server & DB Running:** The cost of 'instance hours' for servers and databases
- **Storage:** The cost of storage volumes and objects
- **Data Transfer:** The cost of data being transferred in and out of the deployment
- **Storage I/O:** The cost of read and write requests to storage
- **DB Transactions:** The cost of running transactions on the databases
- **Support:** The cost of provider support plans
- **Other costs:** Any additional costs that are included in the deployment

Yearly Deployment Costs (USD)

Year	Server & DB	Storage	Data Transfer	Storage I/O	DB Transactions	Support	Other Costs	Total
Year 1	8,519.39	2,244.72	1,922.40	0.00	0.00	0.00	0.00	12,686.51
Year 2	8,498.84	2,538.48	2,440.80	0.00	0.00	0.00	0.00	13,478.12
Year 3	8,498.84	2,832.24	2,959.20	0.00	0.00	0.00	0.00	14,290.28
Totals	25,517.07	7,615.44	7,322.40	0.00	0.00	0.00	0.00	**40,454.91**

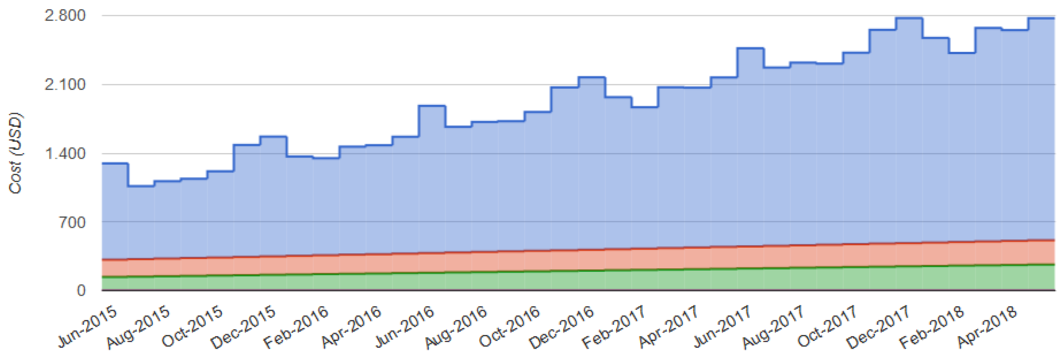
PlanforCloud: Forecasting the Cost of Your Growth

- Creating a new pattern

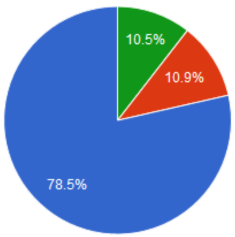


PlanforCloud: Forecasting the Cost of Your Growth

- Generate a new cost report



Total Cost Breakdown (USD)



Key

- **Server & DB Running:** The cost of 'instance hours' for servers and databases
- **Storage:** The cost of storage volumes and objects
- **Data Transfer:** The cost of data being transferred in and out of the deployment
- **Storage I/O:** The cost of read and write requests to storage
- **DB Transactions:** The cost of running transactions on the databases
- **Support:** The cost of provider support plans
- **Other costs:** Any additional costs that are included in the deployment

Yearly Deployment Costs (USD)

Year	Server & DB	Storage	Data Transfer	Storage I/O	DB Transactions	Support	Other Costs	Total
Year 1	11,943.71	2,244.72	1,922.40	0.00	0.00	0.00	0.00	16,110.83
Year 2	18,217.40	2,538.48	2,440.80	0.00	0.00	0.00	0.00	23,196.68
Year 3	24,524.60	2,832.24	2,959.20	0.00	0.00	0.00	0.00	30,316.04
Totals	54,685.71	7,615.44	7,322.40	0.00	0.00	0.00	0.00	**69,623.55**

Total Cost of Ownership (TCO)

- Utilization cost estimation of a product for its life cycle
- Important for deciding whether to migrate to the cloud or not
- Some providers supply TCO comparisons between clouds and traditional IT infrastructures
  - AWS https://awstcocalculator.com/

Service-Level Agreement (SLA)

- Establish agreements between the cloud provider and users
  - Uptime (availability) of the service
  - Response time or latency
  - Component reliability
  - Responsibility of each party
  - Warranties

Licenses

- Based upon EULA (End User License Agreement) for traditional software
- Establish whether the software is
  - Property of the user
  - Can be installed on one or more machines
  - Allows one or more connections
  - Has to follow vendor regulations
- In the cloud
  - Software license is associated with the user account
  - Subscription or usage model
  - Licensing modes are continuously evolving

Costs and licenses

Cloud costs: IaaS

- Based upon the pay-per-use concept (usage model)
- I buy computing, storage, network resources and I pay for what I use
  - Amount of traffic generated
  - Number of CPU cycles used
  - Amount of storage used

Amazon EC2

- EC2 instances
  - https://aws.amazon.com/en/ec2/instance-types/

A1 T3 T3a T2 M6g M5 M5a M5n M4

M5 instances are the latest generation of General Purpose Instances powered by Intel Xeon® Platinum 8175 processors. This family provides a balance of compute, memory, and network resources, and is a good choice for many applications.

Features:

- Up to 3.1 GHz Intel Xeon® Platinum 8175 processors with new Intel Advanced Vector Extension (AVX-512) instruction set
- New larger instance size, m5.24xlarge, offering 96 vCPUs and 384 GiB of memory
- Up to 25 Gbps network bandwidth using Enhanced Networking
- Requires HVM AMIs that include drivers for ENA and NVMe
- Powered by the AWS Nitro System, a combination of dedicated hardware and lightweight hypervisor
- Instance storage offered via EBS or NVMe SSDs that are physically attached to the host server
- With M5d instances, local NVMe-based SSDs are physically connected to the host server and provide block coupled to the lifetime of the M5 instance
- New 8xlarge and 16xlarge sizes now available.

Instance Size	vCPU	Memory (GiB)	Instance Storage (GiB)	Network Bandwidth (Gbps)	EBS Bandwidth (Mbps)
m5.large	2	8	EBS-Only	Up to 10	Up to 4,750
m5.xlarge	4	16	EBS-Only	Up to 10	Up to 4,750
m5.2xlarge	8	32	EBS-Only	Up to 10	Up to 4,750
m5.4xlarge	16	64	EBS-Only	Up to 10	4,750
m5.8xlarge	32	128	EBS Only	10	6,800
m5.12xlarge	48	192	EBS-Only	10	9,500
m5.16xlarge	64	256	EBS Only	20	13,600
m5.24xlarge	96	384	EBS-Only	25	19,000
m5.metal	96*	384	EBS-Only	25	19,000
m5d.large	2	8	1 x 75 NVMe SSD	Up to 10	Up to 4,750
m5d.xlarge	4	16	1 x 150 NVMe SSD	Up to 10	Up to 4,750
m5d.2xlarge	8	32	1 x 300 NVMe SSD	Up to 10	Up to 4,750
m5d.4xlarge	16	64	2 x 300 NVMe SSD	Up to 10	4,750

Amazon EC2

- Pricing
  - https://aws.
    amazon.com/e
    ec2/pricing/

Linux | RHEL | SLES | Windows | Windows with SQL Standard | Windows with SQL Web
Windows with SQL Enterprise | Linux with SQL Standard | Linux with SQL Web | Linux with SQL Enterprise

Region: US East (Ohio)

	vCPU	ECU	Memory (GiB)	Instance Storage (GB)	Linux/UNIX Usage
General Purpose - Current Generation					
a1.medium	1	N/A	2 GiB	EBS Only	$0.0255 per Hour
a1.large	2	N/A	4 GiB	EBS Only	$0.051 per Hour
a1.xlarge	4	N/A	8 GiB	EBS Only	$0.102 per Hour
a1.2xlarge	8	N/A	16 GiB	EBS Only	$0.204 per Hour
a1.4xlarge	16	N/A	32 GiB	EBS Only	$0.408 per Hour
a1.metal	16	N/A	32 GiB	EBS Only	$0.408 per Hour
t3.nano	2	Variable	0.5 GiB	EBS Only	$0.0052 per Hour
t3.micro	2	Variable	1 GiB	EBS Only	$0.0104 per Hour

Region: US East (Ohio)

Data Transfer IN To Amazon EC2 From Internet	
All data transfer in	$0.00 per GB
Data Transfer OUT From Amazon EC2 To Internet	
Up to 1 GB / Month	$0.00 per GB
Next 9.999 TB / Month	$0.09 per GB
Next 40 TB / Month	$0.085 per GB
Next 100 TB / Month	$0.07 per GB
Greater than 150 TB / Month	$0.05 per GB

Amazon S3

- Pricing
  - https://aws.amazon.com/en/s3/pricing/

Data Transfer IN To Amazon S3 From Internet	
All data transfer in	$0.00 per GB
Data Transfer OUT From Amazon S3 To Internet	
Up to 1 GB / Month	$0.00 per GB
Next 9.999 TB / Month	$0.09 per GB
Next 40 TB / Month	$0.085 per GB
Next 100 TB / Month	$0.07 per GB
Greater than 150 TB / Month	$0.05 per GB

	PUT, COPY, POST, LIST requests (per 1,000 requests)	GET, SELECT, and all other requests (per 1,000 requests)	Lifecycle Transition requests (per 1,000 requests)	Data Retrieval requests (per 1,000 requests)	Data retrievals (per GB)
S3 Standard	$0.0054	$0.00043	$0.00	$0.00	$0.00
S3 Intelligent - Tiering	$0.0054	$0.00043	$0.01	$0.00	$0.00
S3 Standard - Infrequent Access*	$0.01	$0.001	$0.01	$0.00	$0.01
S3 One Zone - Infrequent Access*	$0.01	$0.001	$0.01	$0.00	$0.01

Cloud costs: SaaS

- Based mostly upon the subscription model
- I pay a monthly/annual fee and I use the service
- There often exist two versions: a free one and a not-free one

Office365 (SaaS)

Best value: up to 6 users on PC or Mac

Office 365 Home

$99.99 / year

★★★★☆

Buy now

Or buy at $9.99 per month

Try free for 1 month >

Share with your family, up to 6 people, across all their devices.

Office apps included

Word, Excel, PowerPoint, Outlook, Publisher (PC only), Access (PC only)

Services included

OneDrive, Skype

Office 365 Personal

$69.99 / year

★★★★☆

Buy now

Or buy at $6.99 per month

For one person, across all your devices.

Office apps included

Word, Excel, PowerPoint, Outlook, Publisher (PC only), Access (PC only)

Services included

OneDrive, Skype

Office Home & Student 2019

$149.99

★★★★☆

Buy now

One-time purchase

For one person, installed on 1 PC or Mac.

Office apps included

Word, Excel, PowerPoint

Services included

(not included)

Office365 (SaaS)

	BUSINESS TEAM STARTING AT **$4.99**/MO Per User, Billed Annually MULTIPLE USERS	BUSINESS **$7.99**/MO Billed Annually SINGLE USER*	STANDARD **$3.99**/MO Billed Annually SINGLE USER*
Number of private diagrams	**UNLIMITED**	**UNLIMITED**	**200**
Image export	✔	✔	✔
Visio import	✔	✔	✔
UML & wireframe shapes	✔	✔	
Google Drive integration	✔		
Private sharing	✔		
Commenting Tool	✔		

Cloud costs: PaaS

- Sometimes based upon a mix of usage and subscription model
  - E.g., OpenShift https://www.openshift.com/pricing/plan-comparison.html

	FREE PLAN	BRONZE PLAN	SILVER PLAN
BASE PRICE	Free	Free	$20/month
APPLICATION IDLING	24 hours	Never	Never
INCLUDED GEARS	3 small gears	3 small gears	3 small gears
MAX GEARS	3	16	16+
SCALING	Yes (3 min / 3 max)	Yes (3 min / 16 max)	Yes (3 min / 16 max)
GEAR SIZES	small	small ($0.02/hour) small.highcpu ($0.025/hour) medium ($0.05/hour) large ($0.10/hour)	small ($0.02/hour) small.highcpu ($0.025/hour) medium ($0.05/hour) large ($0.10/hour)
STORAGE	1GB per gear	1GB per gear; $1.00/month per additional GB	6GB per gear; $1.00/month per additional GB
SSL	Shared	For custom domains	For custom domains
TEAMS	Not included	Up to 15	Up to 15

OpenShift (PaaS)

- Based mostly upon the concept of subscription model
- Requires a monthly/annual fee to use the service
- There often exist two versions: a free one and a not-free one

Comparison among heterogeneous cloud providers

Total Cost of Ownership (TCO)

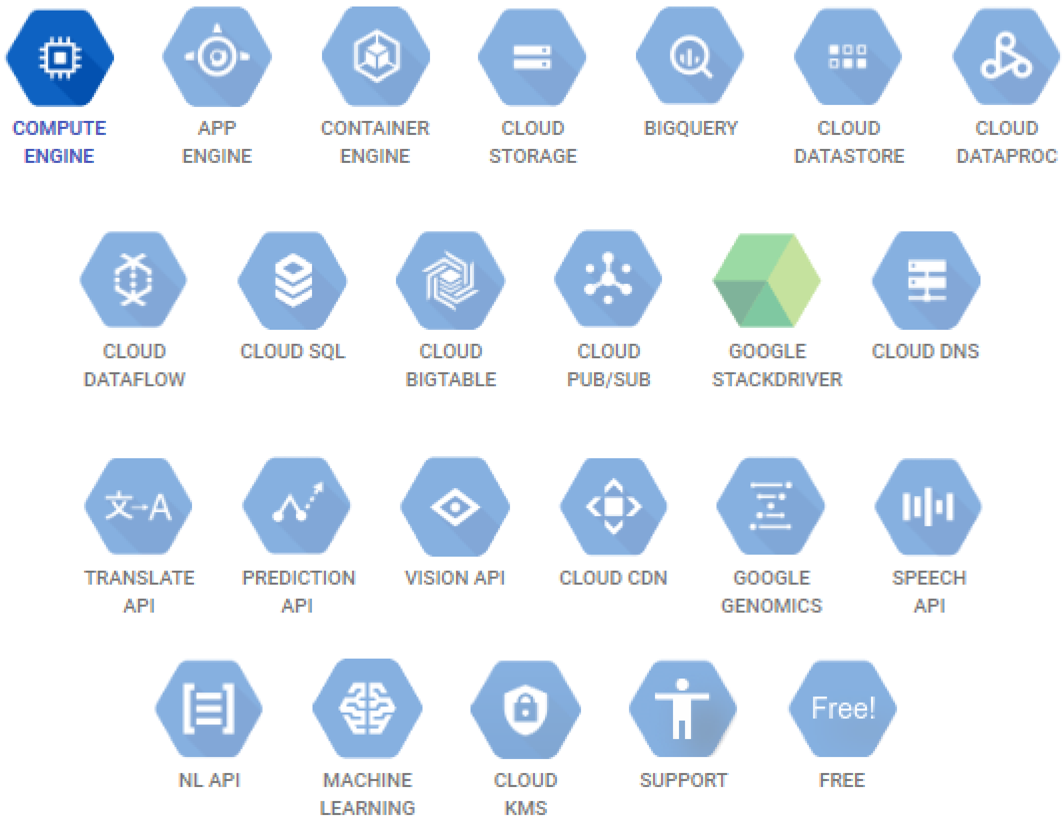
- Total Cost of Ownership (TCO) developed by Gartner in 1987 and used to compute all the costs of an IT equipment life cycle, its purchase, installation, management and disposal
- How to choose whether to migrate to the cloud and to which provider
  - Need to estimate usage costs of product for its life cycle
  - By comparing several offerings and deployment modes
- Many providers supply comparisons between TCO in the cloud and in traditional IT infrastructure
  - Try to google "Cloud Cost Calculator" you will find many solutions to compute your cloud migration costs
  - Many of these solutions compare also the costs of different providers

Total Cost of Ownership (TCO)

- Different cloud providers with different calculators
  - Google Cloud Platform Pricing Calculator
  - Microsoft Azure Cloud Calculator
  - AWS Cloud Cost Calculator
  - Rackspace Cloud Calculator

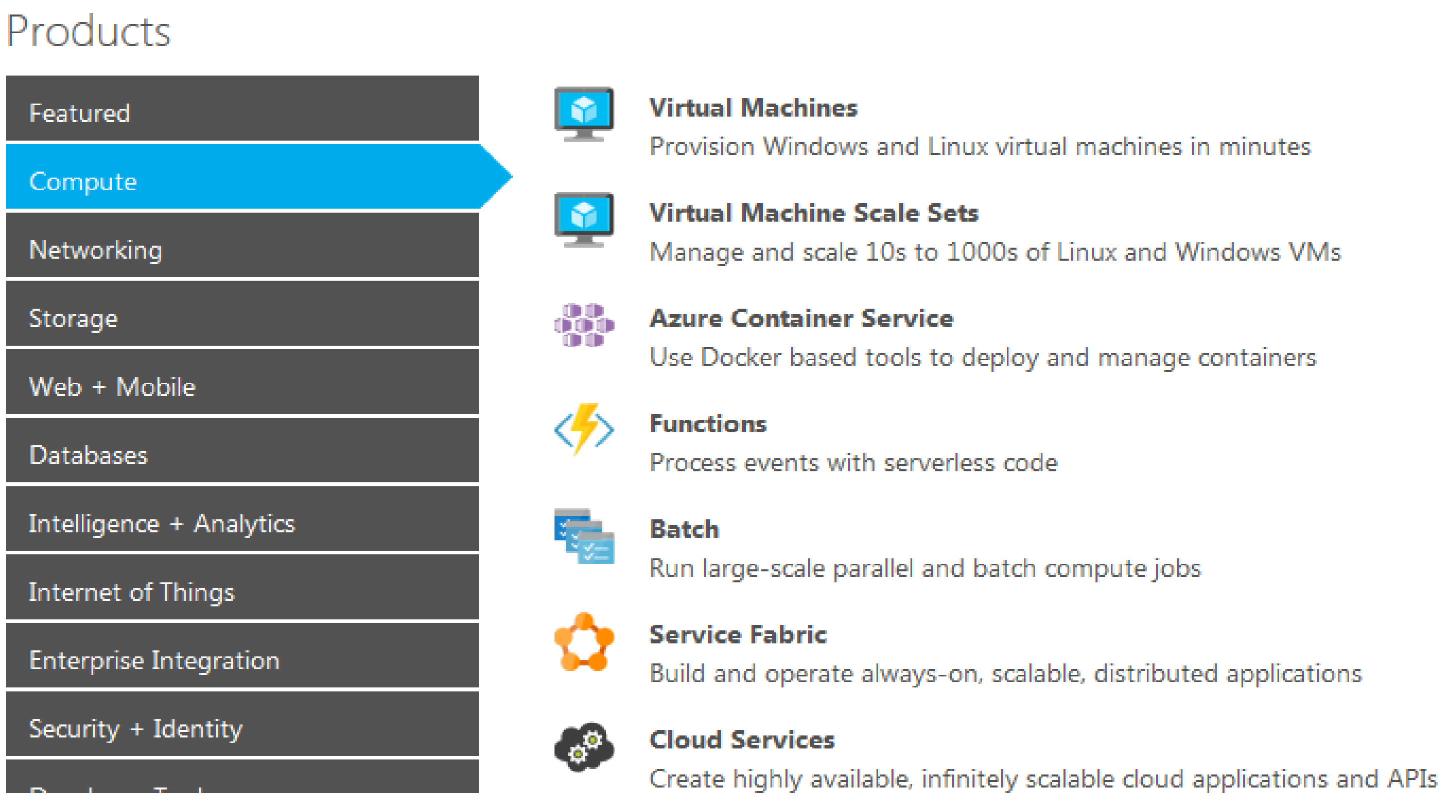
Total Cost of Ownership (TCO)

- Google Cloud Platform Pricing Calculator
  - https://cloud.google.com/products/calculator/



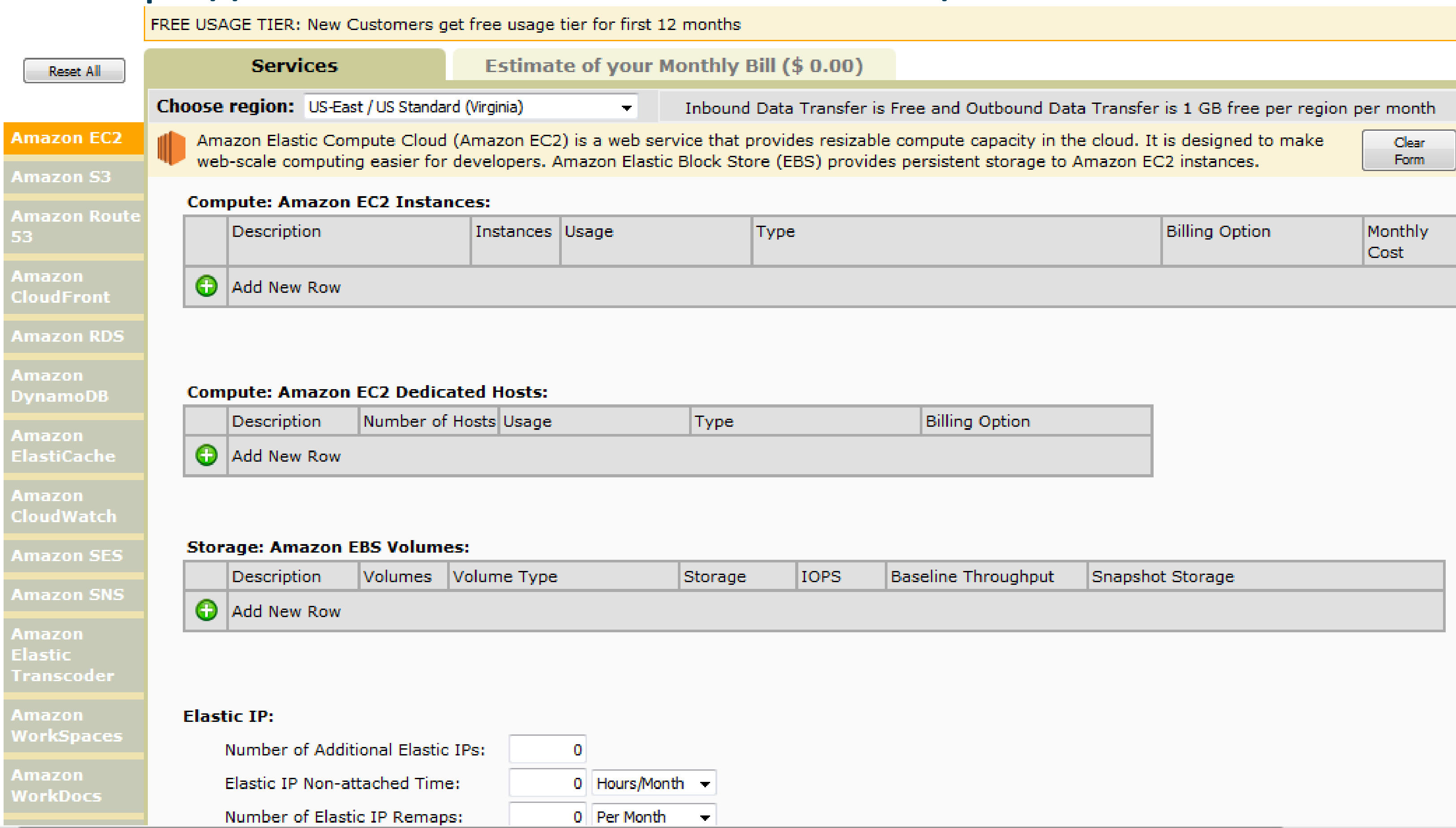
Total Cost of Ownership (TCO)

- Microsoft Azure Cloud Calculator
  - https://azure.microsoft.com/en-us/pricing/calculator/



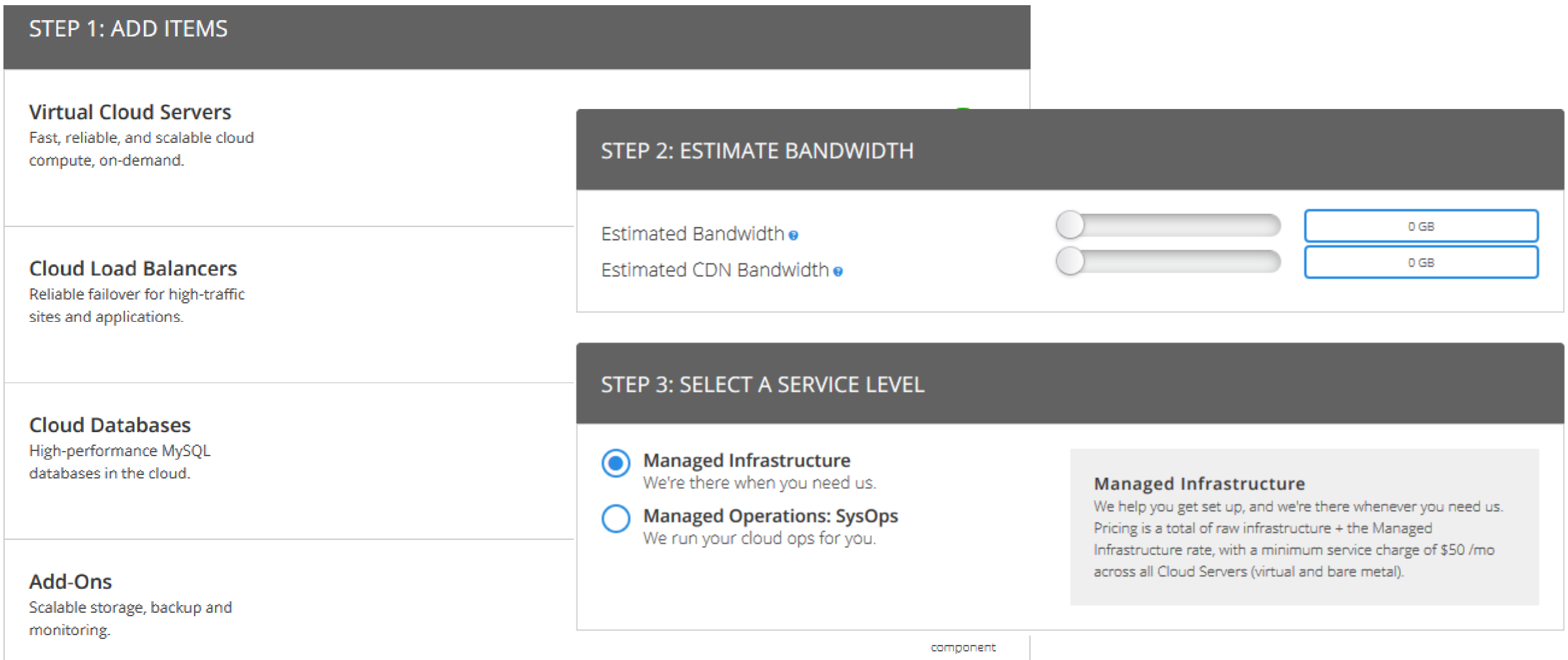
Total Cost of Ownership (TCO)

- AWS Cloud Cost Calculator
  - https://calculator.s3.amazonaws.com/index.html



Total Cost of Ownership (TCO)

- Rackspace Cloud Calculator
  - https://www.rackspace.com/calculator



Total Cost of Ownership (TCO)

- Rackspace provides an environment to manage multiple clouds
  - Including AWS, Azure, Openstack

- Rackspace has recently proposed a comparison work among AWS, Azure and Google costs

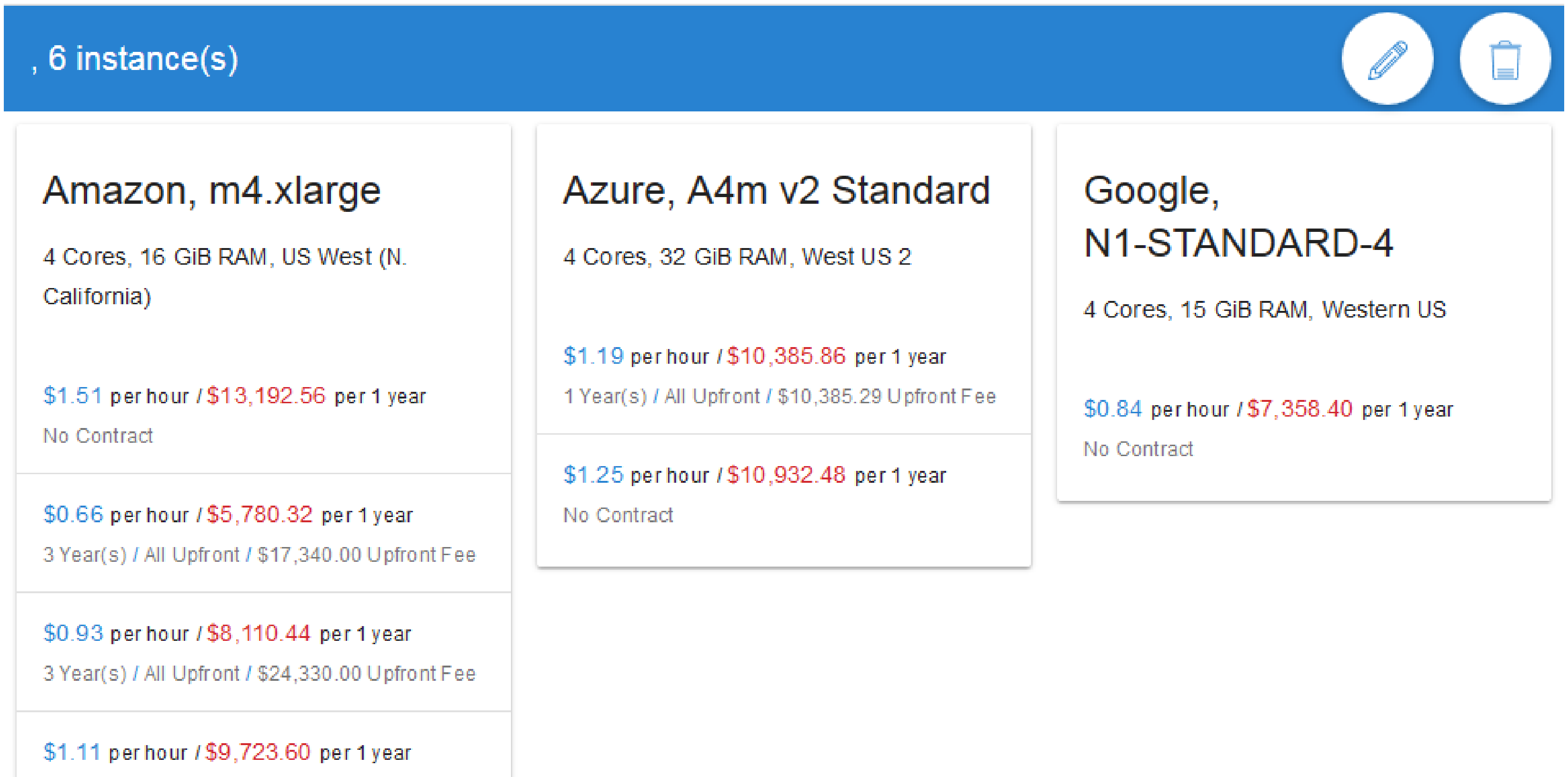
- AWS vs Azure vs Google Cloud Pricing: Compute Instances
  - http://www.rightscale.com/blog/cloud-cost-analysis/aws-vs-azure-vs-google-cloud-pricing-compute-instances

Comparison among cloud offerings

- Public Cloud Cost Comparison Calculator
- Compares the costs of several cloud providers
  - An independent third party
  - Allows having an objective evaluation

Comparison among cloud offerings

- Unigma calculator
  - https://calculator.unigma.com/



Comparison among cloud offerings

- Cloud cost calculator
  - https://www.scalyr.com/cloud/

Restrict region, provider, and lease (Default: North America, any lease)

Restrict server size (specify a range of CPU, RAM, or storage size)

Amortization period and financial assumptions (Default: 24 hours/day for 1 month)

Describe servers using ◉ absolute units ("8 GB") ○ units per dollar per month ("$1.26 GB/$" -- reflects amortized cost)

Provider	Server Type	Cores	RAM	Disk	SSD	$/Month	Lease Type	Upfront	Hourly	Location
Atlantic.Net	256MB	1	256 MB	9.766 GB		$3.652	hourly		$0.005	Eastern-USA
Atlantic.Net	512MB	1	512 MB	19.531 GB		$4.967	hourly		$0.007	Eastern-USA
Digital Ocean	512MB	1	512 MB		20 GB	$5.000	hourly		$0.007	New York
Digital Ocean	512MB	1	512 MB		20 GB	$5.000	hourly		$0.007	San Francisco
Atlantic.Net	1GB	1	1 GB	78.125 GB		$9.935	hourly		$0.014	Eastern-USA
Digital Ocean	1GB	1	1 GB		30 GB	$10.00	hourly		$0.015	New York
Digital Ocean	1GB	1	1 GB		30 GB	$10.00	hourly		$0.015	San Francisco
Google	f1-micro	0.182	614.4 MB			$13.88	hourly		$0.019	US Central
Amazon	t1.micro	0.182	615 MB			$14.61	on demand		$0.020	us-east-1 (Virginia)
Amazon	t1.micro	0.182	615 MB			$14.61	on demand		$0.020	us-west-2 (Oregon)
Azure	Extra small VM	0.167	768 MB	20 GB		$14.61	hourly		$0.020	Virginia

Comparison among cloud offerings

- Cloudorado
  - Cloud Computing Comparison Engine
  - https://www.cloudorado.com/

- Not only it compares costs, but also non functional aspects: certification, supported standard, security …

Service-Level Agreement (SLA)

- They establish agreements between the cloud provider and users
  - Uptime (availability) of the servizio
  - Response time or latency
  - Component availability
  - Responsibility of each party
  - Warranties

- They collect a set of metrics useful to evaluate different cloud providers at the moment of migrating to the cloud

Lesson 2.3: IaaS – OpenStack (READ-ONLY)

Claudio Ardagna – Università degli Studi di Milano

Cloud and Distributed Computing

Cloud Computing: IaaS

- A user manages the entire processing (CPU), memory, storage, network, and additional computational resources
  - Amazon, Google, Nuvola Italiana
- A user can install and execute generic code, including operating systems and applications
- A user does not manage or control the cloud infrastructure, while she controls operating systems, storage, and installed applications
  - No need to check hardware and manage failures, faults, obsolescence…
- A user has a limited control of network components (e.g., host firewall)

Cloud Computing: IaaS

- Provide on demand virtualized resources
- Provide different server with different operating systems and an ad hoc software stack
- Amazon provides virtual machines with different operating systems
  - EC2 Service
  - It is similar to a physical server
  - Users can start and stop a VM, install software, connect virtual disks

OpenStack

OpenStack – History

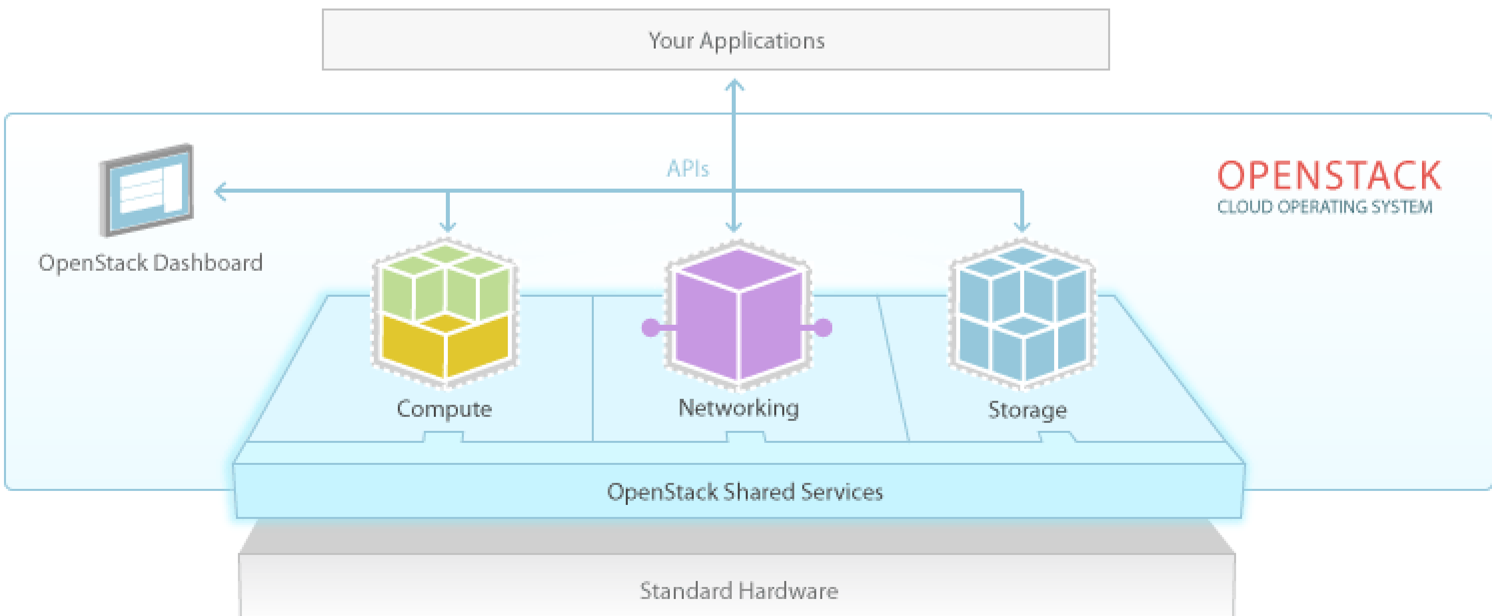
- July 2010: Rackspace and NASA start an initiative called OpenStack
  - Aim to foster the adoption of a cloud computing solution for service offer by enterprises
- October 2010: first version - OpenStack *Austin*
  - Integrate NASA Nebula platform and Rackspace Cloud Files
- 2011: Ubuntu Linux developers adopt Openstack towards the release of the second version *Bexar*
  - Complete support from the third version *Cactus*
    - Available also for Debian release
- 2012: Debian 7.0 includes OpenStack *Essex*, RedHat distributes OpenStack based on *Essex* release

OpenStack – History

- 2013: RedHat provides commercial support to OpenStack Grizzly
- July 2013: NASA leaves the project
  - Due to lack of technical progresses and other factors
  - Concentrate on the use of public clouds
- August 2013: Avaya decides to use OpenStack to create an end-to-end virtual network infrastructure
- May 2014: HP releases HP Helion based on OpenStack IceHouse
- Last release: Train

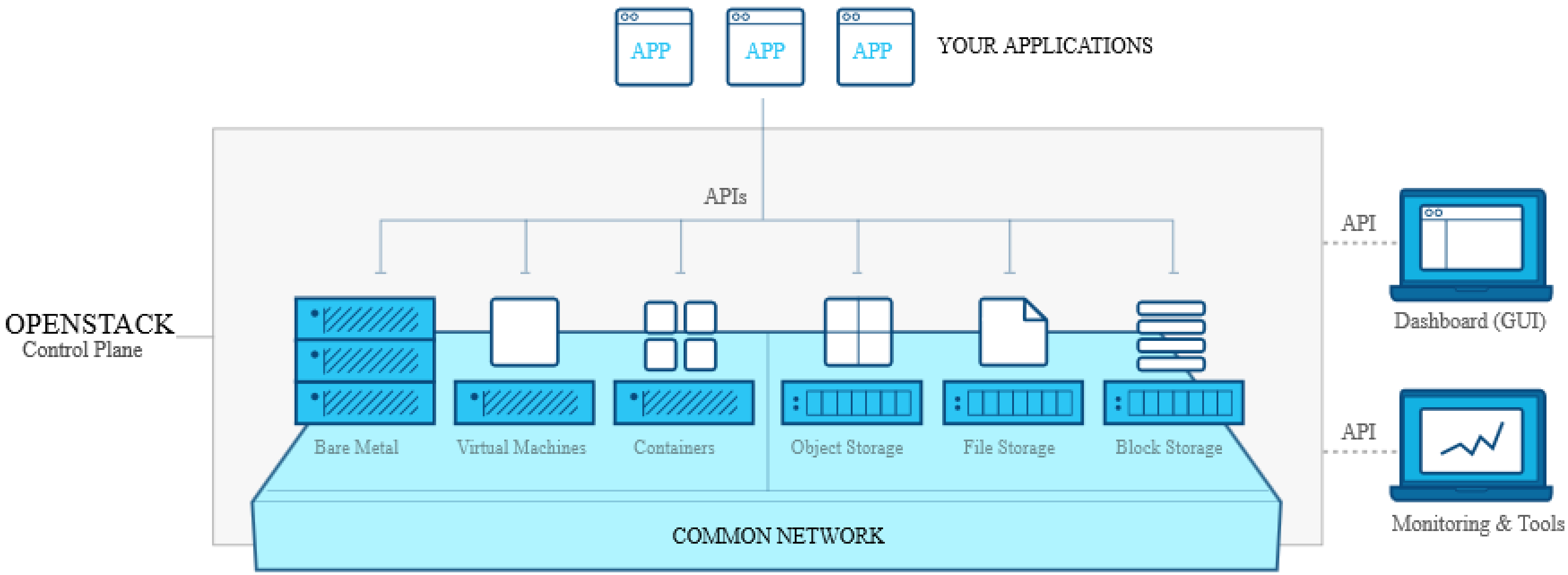
OpenStack

- Open source software for private and public cloud creation
- Cloud operating system that controls a pool of resources: compute, storage, networking
  - Administrator manages resources using the dashboard
  - Users access resources through a web interface
- https://docs.openstack.org/train/admin/
- https://releases.openstack.org/train/index.html
- https://www.openstack.org/software/start/



OpenStack

- Open source software for private and public cloud creation
- Cloud operating system that controls a pool of resources: compute, storage, networking
  - Administrator manages resources using the dashboard
  - Users access resources through a web interface
- https://docs.openstack.org/train/admin/
- https://releases.openstack.org/train/index.html
- https://www.openstack.org/software/start/



OpenStack

- Provide users with a trustworthy and configurable cloud solution
- Simple to implement, provide scalability and extensibility, provide many advanced functionalities
- Interconnected services implement different components of the cloud infrastructure
  - Can be accessed through APIs or dashboard

OpenStack Dashboard

- Implement a GUI to access, retrieve, and automatize cloud resources
- Permit to deliver third-party services like billing and monitoring
- Extensible web application to control compute, storage and networking resources
- Provide an overview on the cloud dimension and status
  - Permit to create users and projects, assign users to projects and limit the use of resources

OpenStack Compute

- OpenStack supports providers in the release of computing resources on demand
  - Resources accessed through APIs or web interface
- Developed for horizontal scaling
- Provide a flexible architecture
  - Open source
  - Can be integrated with third parties and legacy systems
- Manage and automate resource distribution and support virtualization techniques
  - Xen, KVM

OpenStack Compute

- Functionalities
  - Manage virtualized resources
  - Support for LAN (DHCP, IPv6)
  - Authentication and API usage
  - Synchronous and distributed architecture
  - VM image management (also live)
  - Floating IP
  - Security group
  - RBAC
  - Projects and quota
  - …

OpenStack Storage

- Support both Object Storage and Block Storage
- Object Storage provides storage with optimized costs and ability to scale out
  - Storage platform fully distributed, can be accessed through APIs, can be integrated with applications or used for storage
  - It is not a traditional file system, it stores data in objects
  - OpenStack manages replica and supports horizontal scalability
- Block Storage permits to connect block devices to compute instances to achieve greater performance and intengrates with enterprise storage platforms
  - Manage creation, mounting and unmounting of block storage
  - File divided in fixed data blocks
  - Integrated with OpenStack compute and dashboard

OpenStack Storage

- Functionalities
  - Based on common hardware
  - Scalability support
  - Unlimited storage
  - Replica support
  - No central DB
  - Can be integrated with compute
  - Support S3 APIs
  - Snapshot and backup per block volume

OpenStack Networking

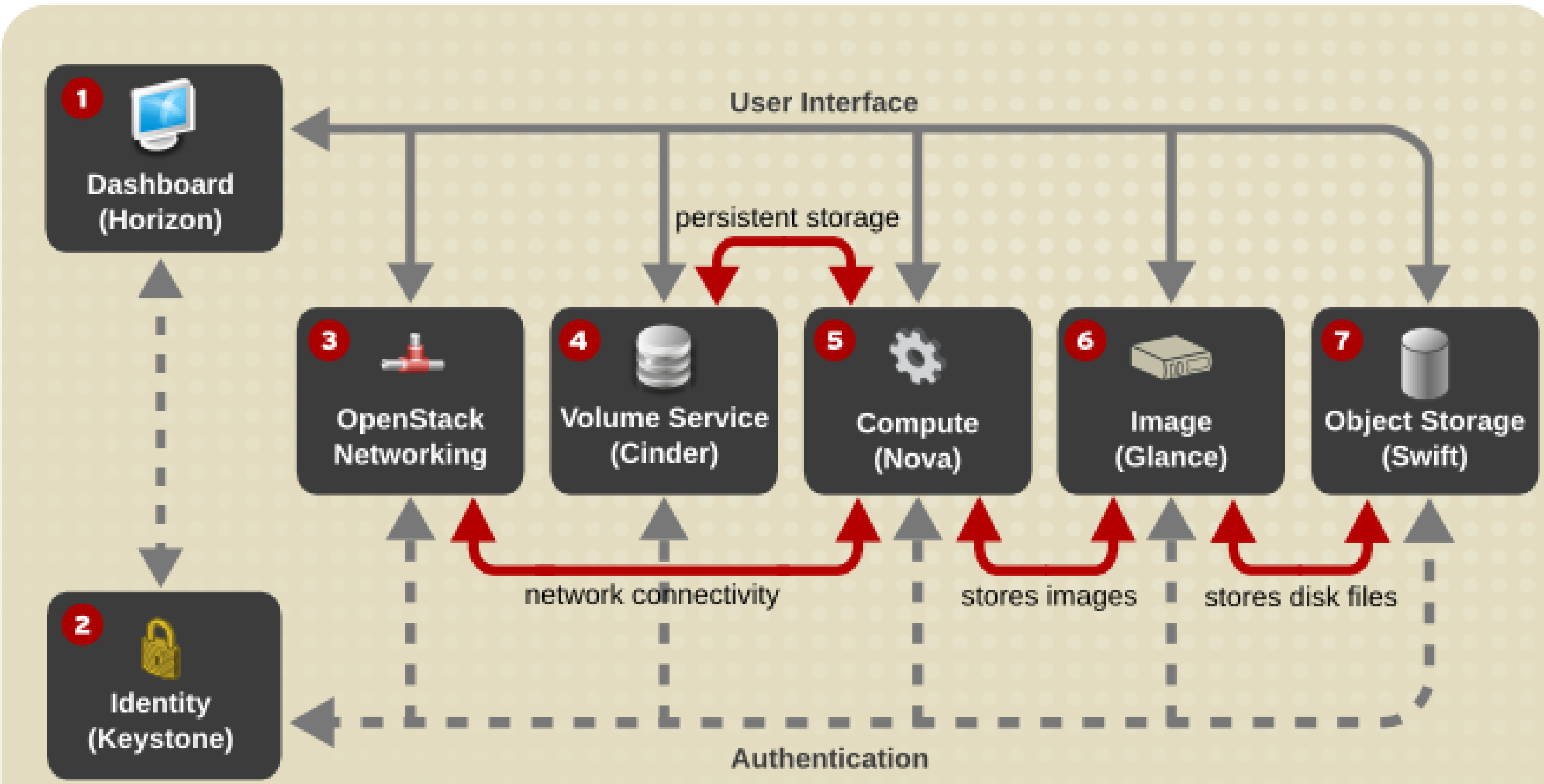
- Datacenter network contains an ever increasing number of servers, network equipments, storage systems, and security appliances
  - These devices are divided in VMs and virtual networks
  - IP address, routing configurations, security rules grew exponentially
  - Traditional approaches to network management do not provide automatic and scalable support
  - Users require more control and flexibility
- OpenStack Networking is scalable, based on APIs and "pluggable"
  - Manage network and IP address
  - Ensure no bottleneck or limitations in a cloud deployment
  - Users create their own network, control traffic and connect servers and devices

OpenStack Shared Service (excerpt)

- Identity Service
- Image Service
- Telemetry Service
- Orchestration Service
- Database Service

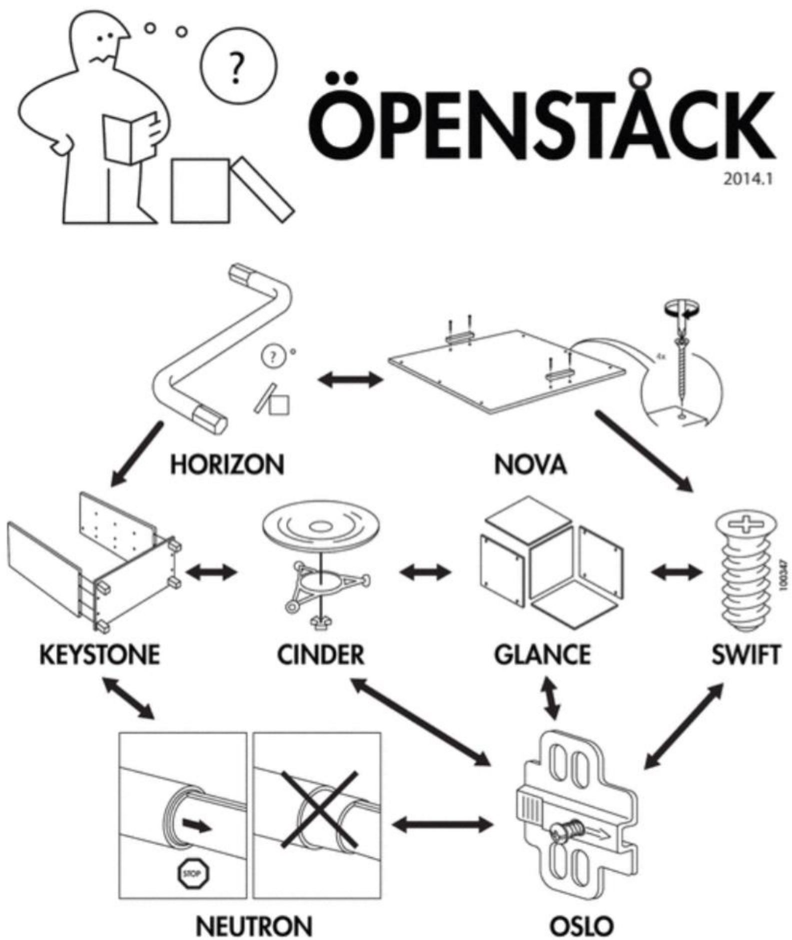
OpenStack Services

- Complete list https://www.openstack.org/software/project-navigator



https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux_OpenStack_Platform/2/html/Getting_Started_Guide/ch01.html

OpenStack Services



How OpenStack is made

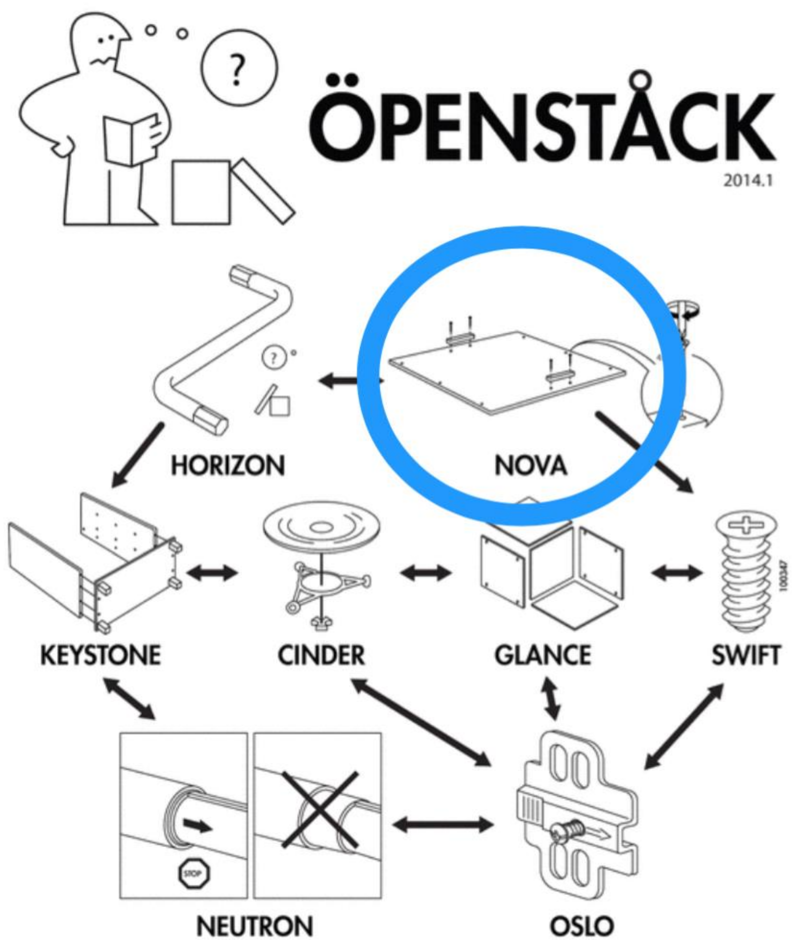
Mapping services/projects

- Dashboard -> Horizon
- Compute -> Nova
- Networking -> Neutron
- Object storage -> Swift
- Block storage -> Cinder
- Identity -> Keystone
- Image -> Glance
- Telemetry -> Ceilometer
- Orchestration -> Heat
- Database -> Trove

Horizon – Dashboard

- Modular web application based on Django (web framework based on python language)
- Dashboard can be accessed by clients and APIs of any OpenStack services
- Admin endpoints provide admin functionalities through the dashboard
- Permit to execute all operations for the management of the infrastructure and account
- A few clicks to deploy/undeploy a VM, create volumes, manage security

Compute – Nova



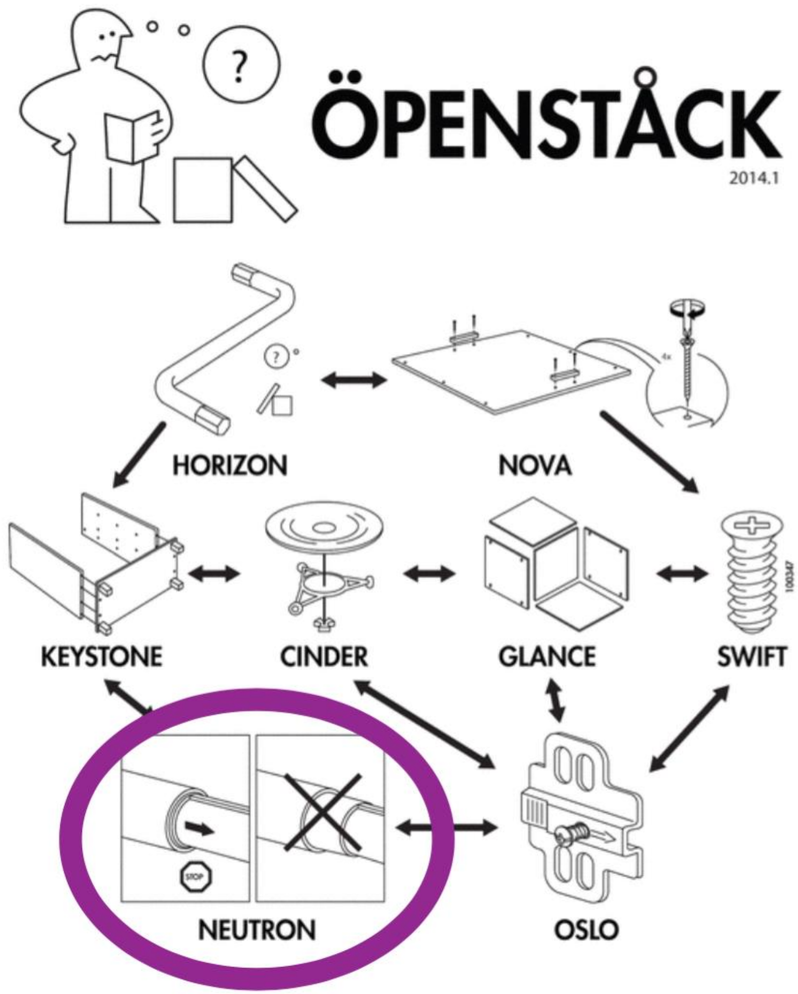
Compute

- KVM
- VMWare
- Hyper-V
- XenServer
- PowerKVM

Compute – Nova

- Fundamental service of OpenStack infrastructure
- Manage the entire lifecycle of the VM instances in the virtualized environment
  - Creation, coordination, deletion of VMs
- Interact with the identity service for authentication, with the image service to access disk and server images, and dashboard to provide admin and user interface
- Horizontal scalability on standard hardware and retrieve images needed for instance creation
- Nova-network demon used for network management

Networking – Neutron



Network

- OpenvSwitch (ML2)
  - VLAN
  - VxLAN
  - Flat Network
  - Provider Networks
- 3rd parties
  - NSX
  - OpenContrail/Contrail
  - & others

Networking – Neutron

- Networking service (neutron-server)
- Permit to create and implement in the virtual network devices and interface managed by other OpenStack services
- Plug-in: full flexibility, permit to manage and interconnext different network devices and software
- Interact with compute service to provide a network infrastructure and support network access to the instances

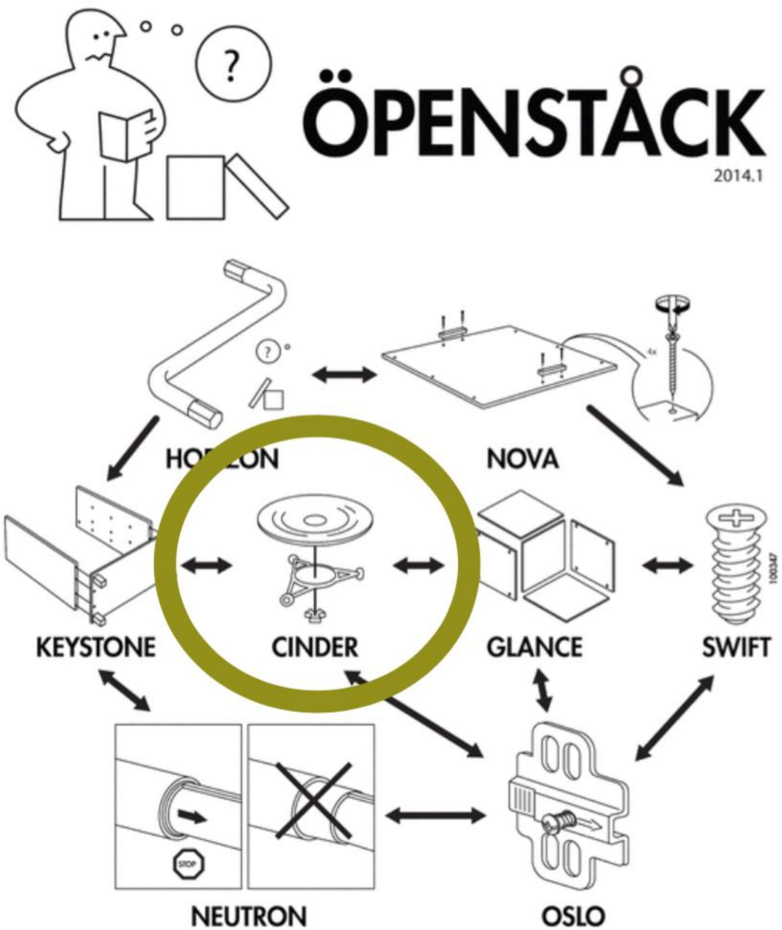
Networking – Neutron

- Main operations: port plug and unplug, net and subnet creation, IP addressing
- Plug-in and agents differ on the basis of the provider and technologies used to implement the cloud environment
- Evolution of nova-network
  - More functionalities
  - More difficult to manage
- Nova-network and Neutron are two different implementations of the Networking-as-a-Service paradigm for OpenStack
- Neutron manages a great number of plug-ins supporting complex configurations

Storage concept

- On-instance / ephemeral
  - Provide space from zero
  - Associated with VM
  - Can be accessed when the VM is active
  - Support flavor-based restrictions
- Block Storage
- Object Storage

Block storage (cinder)



Storage

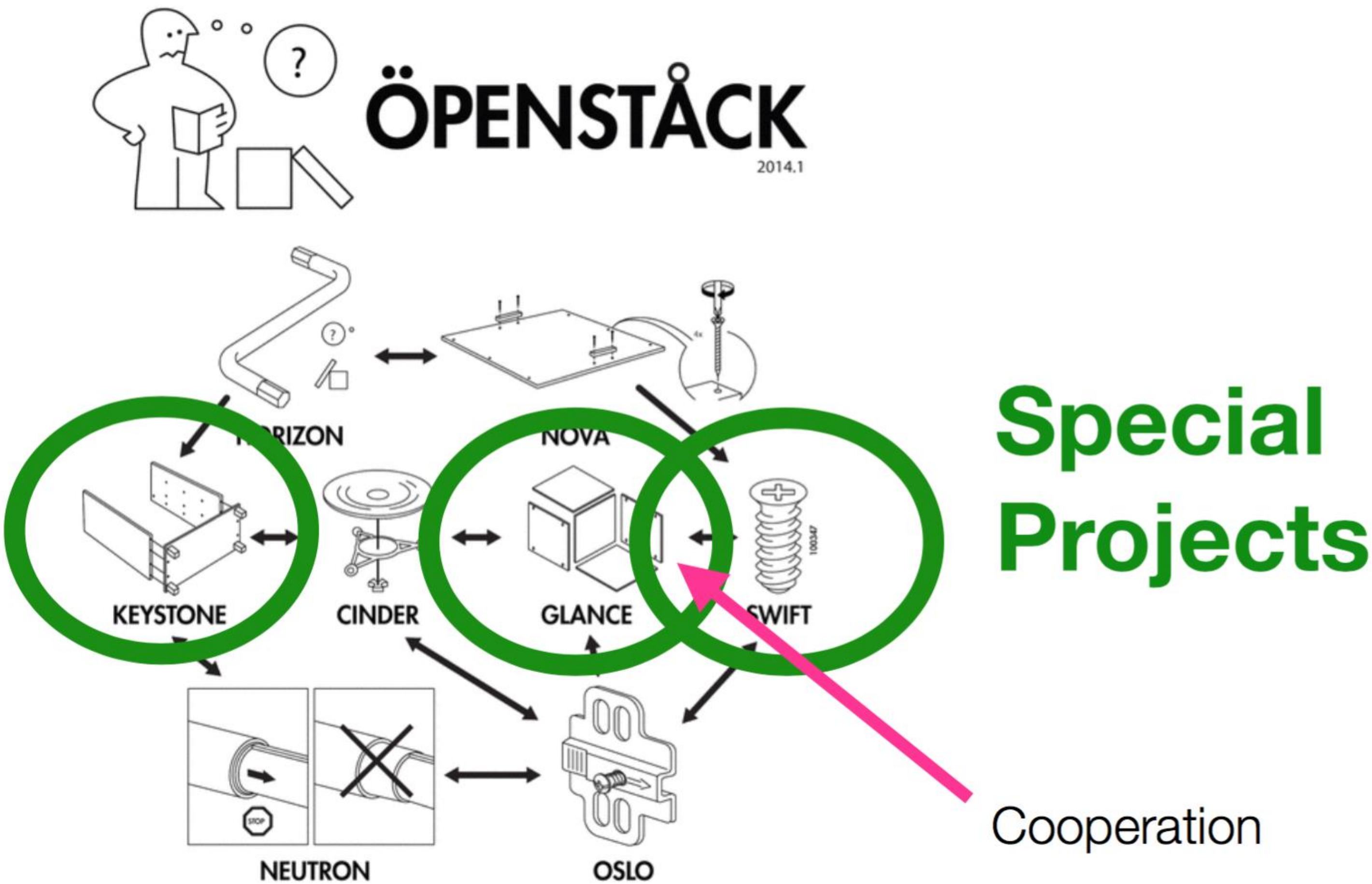
- Software Defined
  - CEPH
  - Gluster
  - HPE LeftHand
  - IBM GPFS
- Traditional
  - NetApp
  - HPE 3PAR & XP
  - Dell StorageCenter
  - EMC VMAX
  - Hitachi
  - Nexenta
  - … many others

Full list:
https://github.com/openstack/cinder/

Block storage (cinder)

- Add persistent storage to a VM
- Data maintained until the VM is canceled
- Provide an infrastructure to manage volumes and volume snapshots
- Interact with OpenStack Compute
- Can be accessed by VMs
- Mounted through OpenStack Block Storage controlled protocol (e.g., iSCSI)
- Storage dimension depends on the needs

Some additional services



Object storage (swift)

- Multi-tenant object storage system
  - Used to store data and images of VMs
- Data maintained until the VM is canceled
- Can be accessed everywhere
- Manage big amount of data and can scale using RESTful HTTP API
- Provide abstractions to store methods, non storage itself
- Can be executed independently from Compute Nova

Identity – Keystone

- Two main functions
  - Track users and corresponding privileges
  - Provide a list of services available with API endpoint
- At service installation time, for identity management, any services in the OpenStack infrastructure is registered
  - At this point, the identity service knows active services and where they are located in the network
- Keystone generates authorization tokens for users
  - Using keystone APIs, a user submits its credentials and retrieves the authentication token
  - Maintain a table with users and privileges

Identity – Keystone

This image refers to OpenStack Kilo

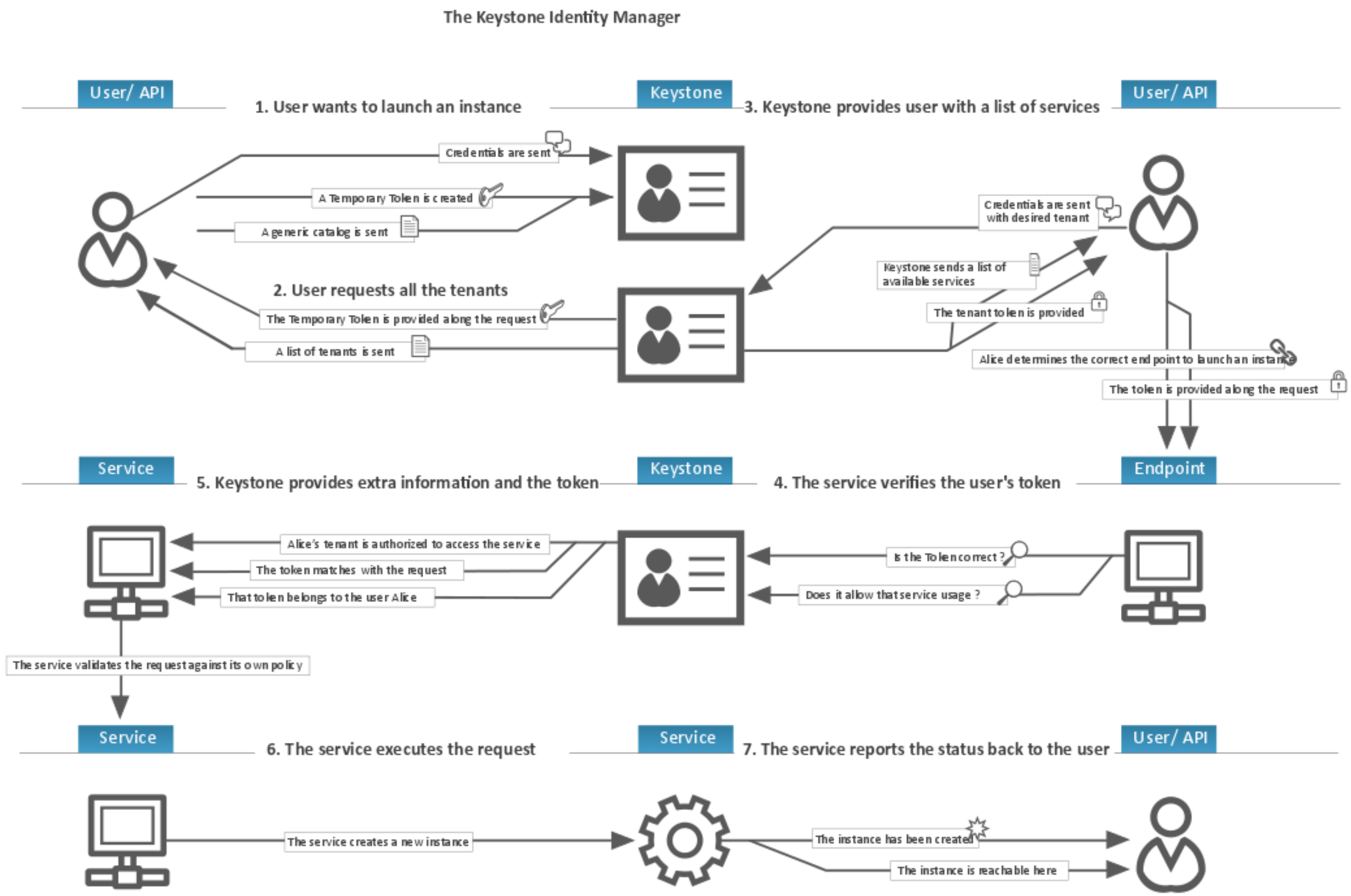


Image – Glance

- Important for IaaS
- Accept requests for disk images or server images, VMs, or metadata corresponding to images of end users or OpenStack Compute components
- Support storage of disk images and server images on different types of repositories including OpenStack Object Storage
- Support caching, replication to provide consistency and availability, auditing

Telemetry – Ceilometer

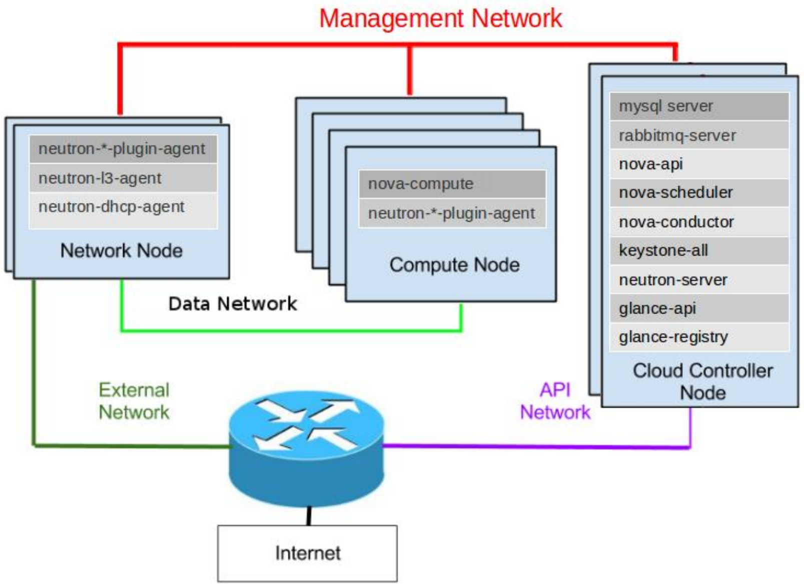
- Collect metering data from OpenStack services
  - CPU, network usage
  - Useful to calculate the bill (pay-as-you-go)
- Collect events and metering data monitoring notifications sent by services
- Publish data through data store and message queue
- Raise alarms when collected data violates predefined rules
- Contain: compute-agent, central-agent, notification-agent, collector, alarm evaluator, alarm notifier, API server

Orchestration – Heat

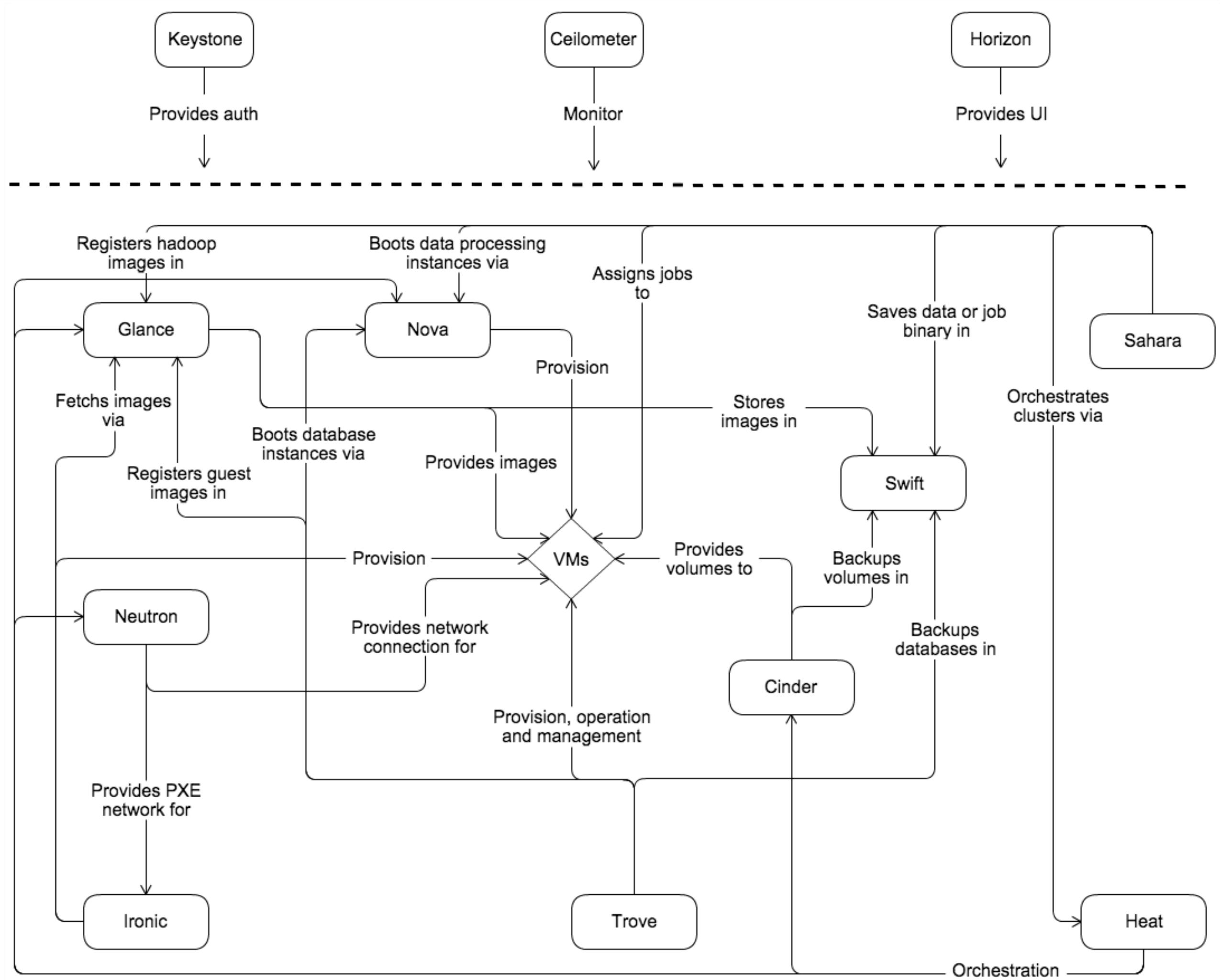
- Provide a template-based orchestration
  - Permit to describe and automatize the infrastructure deployment
  - Execution of calls to OpenStack API to generate executable cloud applications
- The template language permits to
  - Specify configurations of compute, storage and network
  - Specify post-deployment activities to automate the provisioning of infrastructure, services, and applications
  - Create many types of OpenStack resources: instances, floating IPs, volumes, security groups and users
- Heat permits developers to directly integrate the orchestration module of by means of custom plug-in
  - Provide high availability and nested stack

The network

- Four networks
  - Management network
    - Dedicated to internal communications between processes
    - Exchange of information between OpenStack and system (MySQL, KVM) services
    - Isolated and secure, only for accesses from services composing the infrastructure
  - Data network
    - Communications at ISO/OSI level 3
    - Isolated and secure
    - Mapped on a physical network available on Neutron or Nova-Network
  - External network
    - Provide OpenStack services to external users
    - Access to instances of the external network
  - API network
    - Dedicated to direct messages to the public APIs of the services

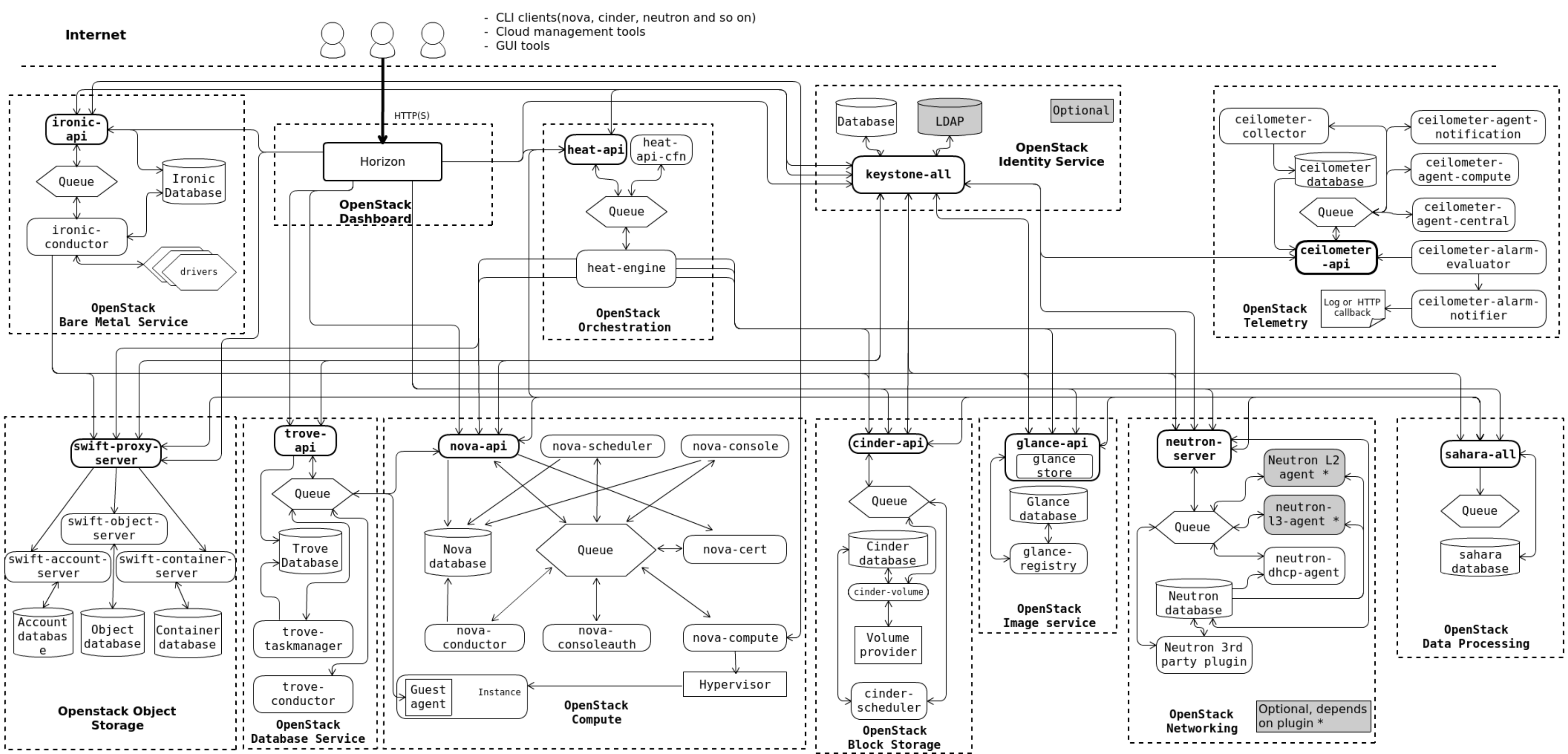


Conceptual Architecture



Logical Architecture

- API REST-based communication
- Service interaction



Logical Architecture

- All services authenticate using the Identity service
- Services interact through public APIs
  - Unique exception for privileged admin commands
- OpenStack services composed by different processes
  - All services have an API process waiting for requests, pre-processing requests and distributing them to the entity responsible
- Communications between processes happens with an Advanced Message Queuing Protocol (AMQP) message broker; service status stored in a database
  - Different options: RabbitMQ, Qpid, MySQL, MariaDB, and SQLite
- Access to OpenStack through a web interface (implemented from dashboard service), client from command line, and API request with browser plug-in or curl

Conclusions

- Main characteristics of OpenStack IaaS
- Overview of OpenStack components
- OpenStack architecture

Lesson 2.4: OpenStack – Lab.

Claudio Ardagna– Università degli Studi di Milano

Cloud and Distributed Computing

Introduction to Computer Networks (READ-ONLY)

Introduction

- Last centuries characterized by different revolutions
  - 18°: mechanical systems, industrial revolution
  - 19°: steam engines
  - 20°: IT
    - Collection and storage
    - Analysis
    - Distribution

Introduction

- 20th century
  - Phone system worldwide
  - Radio and TV
  - Computer and internet
  - Satellite communications
  - Mobile network
- 21st century
  - Smartphone and app
  - Web applications
  - Cloud computing
  - Big data
  - IoT

Computer Networks

- Mainframe - terminals
  - Computational power in a single machine
  - Access through a terminal
- Computer networks
  - Autonomous and interconnected processors
- Internet
  - Network of networks
  - Distributed and decentralized topology

Computer Networks

- Used by enterprises for
  - Resource sharing
  - Reliability
  - Cost reduction
  - Scalability
  - People communication

- Used by users to
  - Access remote infomation
  - Communicate with other users
  - Entertainment and social networking

Computer Networks

- Communication system depends by
  - The nature of the application
  - The number of machines involved
  - The physical distance

- Let us consider two machines
  - Same room: point-to-point communication
  - Different locations: public lines (PSTN - Public Switched Telephone Network)

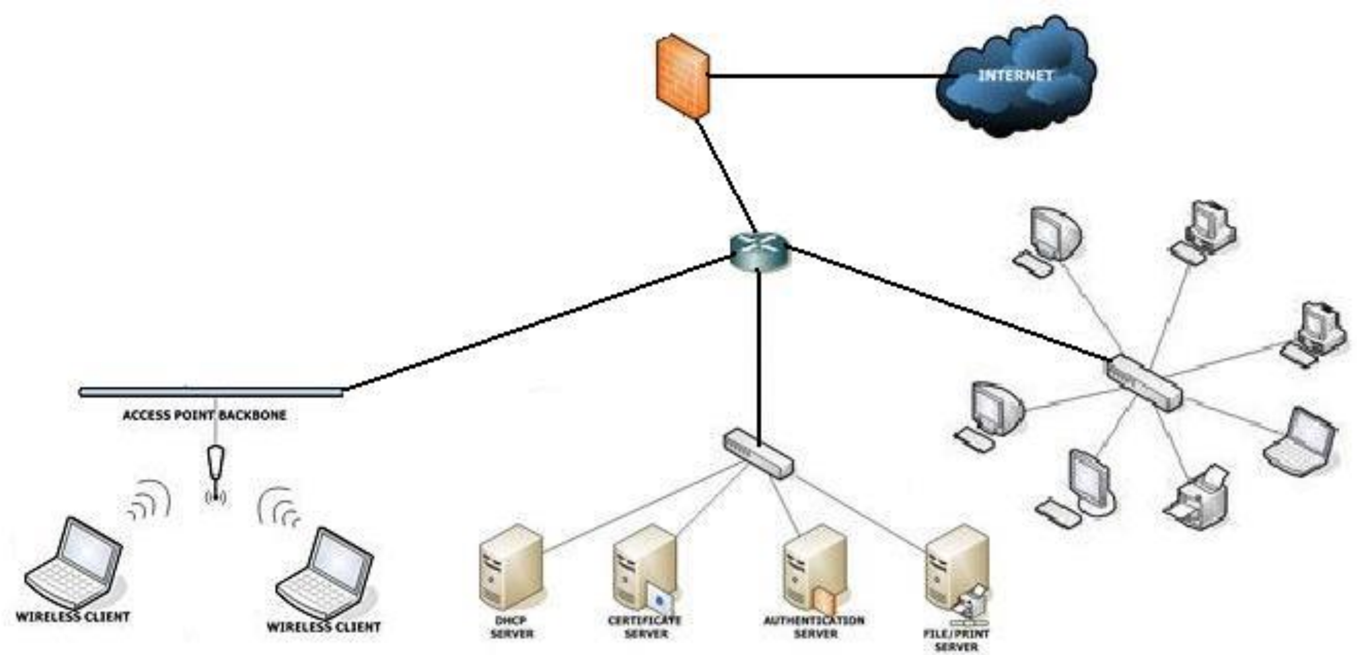
Computer Networks – Dimensional Scale

- Local Area Network (LAN)
- Metropolitan Area Network (MAN)
- Wide Area Network (WAN)

Distance between processors	Environment	Network type
10m	Room	LAN
100m	Building	LAN
1km	Campus	LAN
10km	City	MAN
100km	Country	WAN
1000km	Continent	WAN
10.000km	Planet	Internet (WAN)

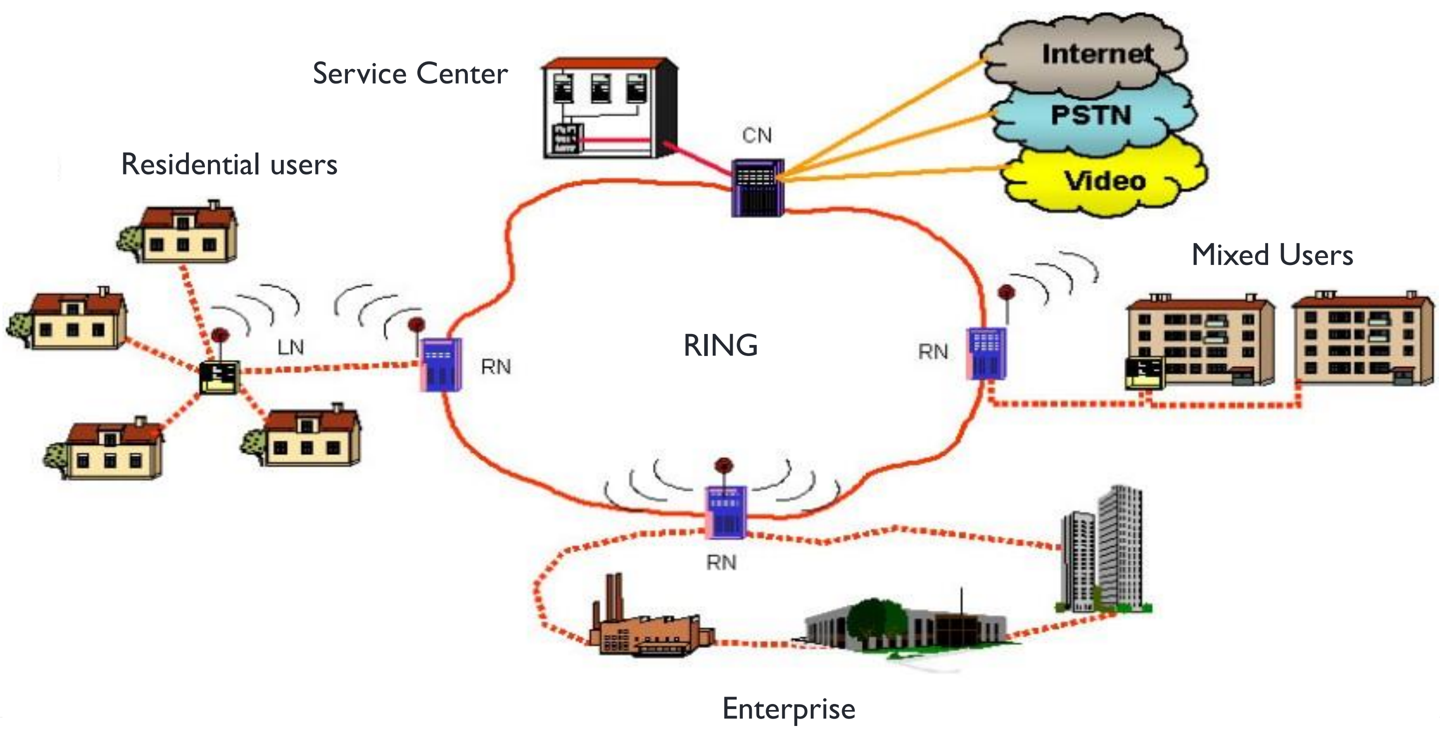
Local Area Network (LAN)

- Managed by single organizarion
- Cover some km
- Do not reside on public property (bulding, campus)
- Connect PC or workstation



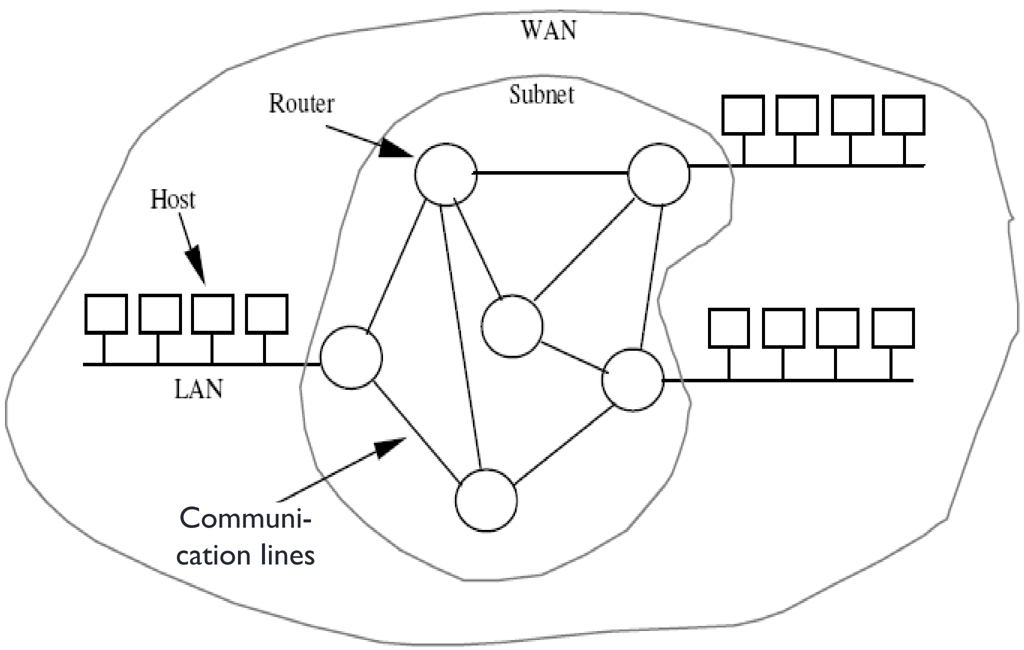
Metropolitan Area Network (MAN)

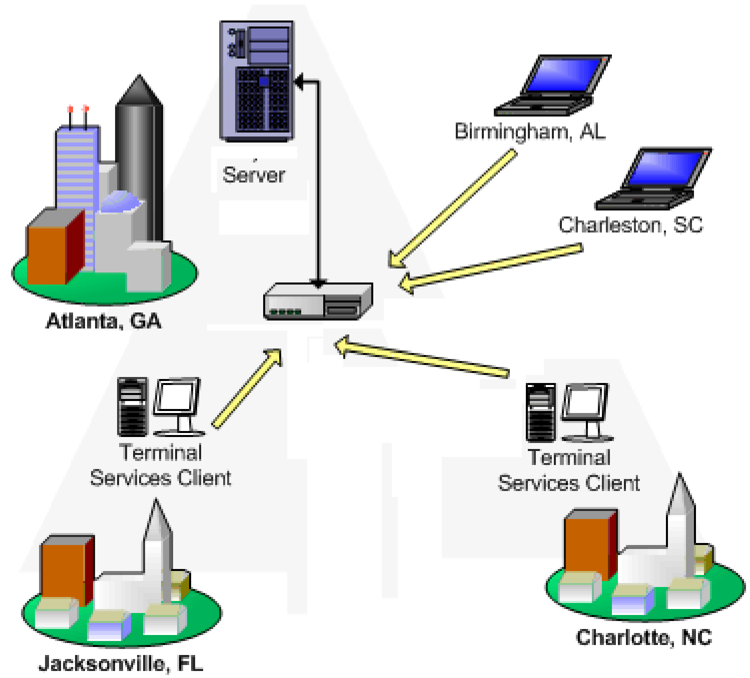
- Cover a metropolitan area
- Public line (pay per use)
- Larger than LAN



Wide Area Network (WAN)

- Cover entire nations, continents, planet
- Composed of
  - PCs, Server
  - Communication subnet (router and communication lines)





Internet

- Network of networks (LAN, MAN, WAN)
- Distributed network, similar to a WAN…
- …but different
  - Connect heterogeneous networks
  - Need of special components (gateway)

Standard ISO-OSI

- Mid 70s the IT industry realized the advantages of open systems
- International Standard Organization (ISO) defined the standard regulating the global structure of a complete subsystem of communication
  - Known as ISO reference model for the interconnection of open systems (OSI - Open System Interconnection)

Standard ISO-OSI

- A complex communication system
  - An unstructured implementation based on a «single program» is not simple to test and modify
- For this reason ISO adopted a layered model
- The communication system can be divided in layers, each one executing a specific predefined function

Standard ISO-OSI

- ISO/OSI layers can be classified in two categories
  - Functions dependent on the network (**Media layers**) (Physical, Data link, Network)
  - Function focused on the application (**Host layers**) (Transport, Session, Presentation, Application)

ISO/OSI
Application
Presentation
Session
Transport
Network
Datalink
Physical

Standard ISO-OSI

- The function of each layer is defined as a set of rules and agreements used to communicate with the corresponding remote layer (**protocol**)

- Each layer provides services to the upper layer and uses the services by the lower layer

Protocol

- The communication between entities requires cooperation, that is, collaboration to achieve a common goal
  - Communications are regolated using protocols

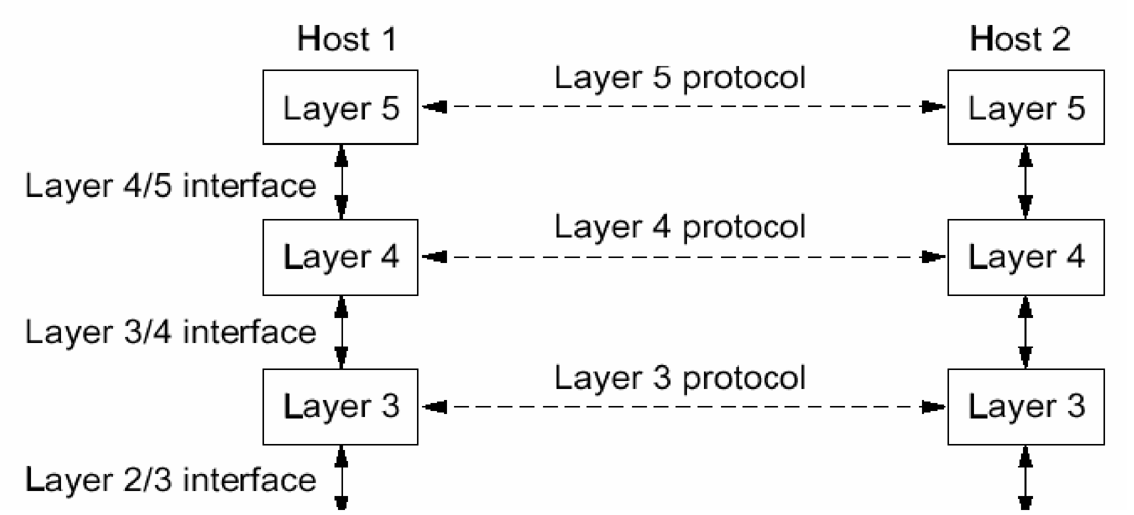
- **Protocol**: set of rules and agreements followed byentities, located on different nodes, that want to communicate to carry out a common work
  - These rules have the goal of ensure an efficient and reliable cooperation for node communication, the execution of services that consider the characteristics of a typical distributed system (limited bandwidth, latency, communication errors, …)

Protocol

- **Syntax**: set and structure of the commands and responses, message format
- **Semantics**: meaning of the commands, actions, responses to be used during message exchange
- **Timing**: definition of the possible timeline for the sending of commands and messages, as well as responses

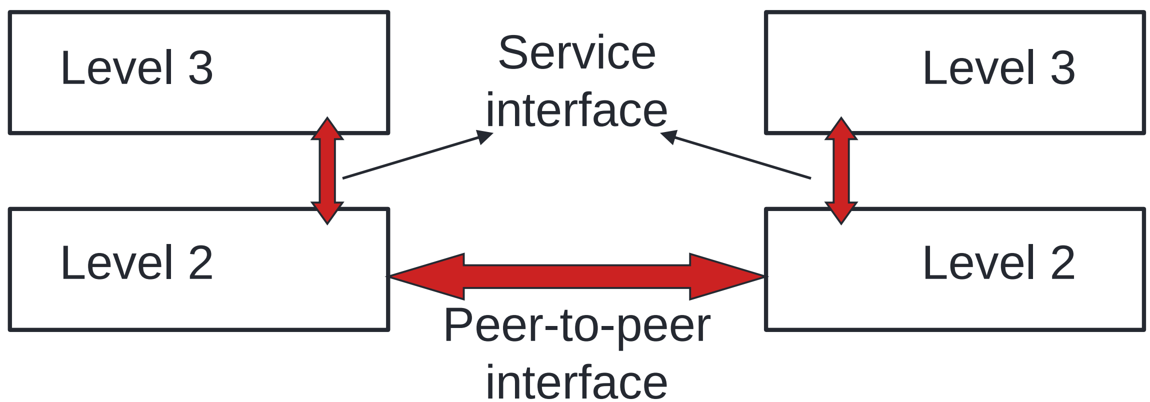
Protocol

- Every protocol ha an «internal» interface towards upper and lower layers, and an «external» interface towards the corresponding layer of another node



Protocol

- **Service interface** (“internal”): operation and services offered to upper layer
- **Peer-to-peer interface** (“external”): messages exchanged with the corresponding layer (*peer*) on the other node

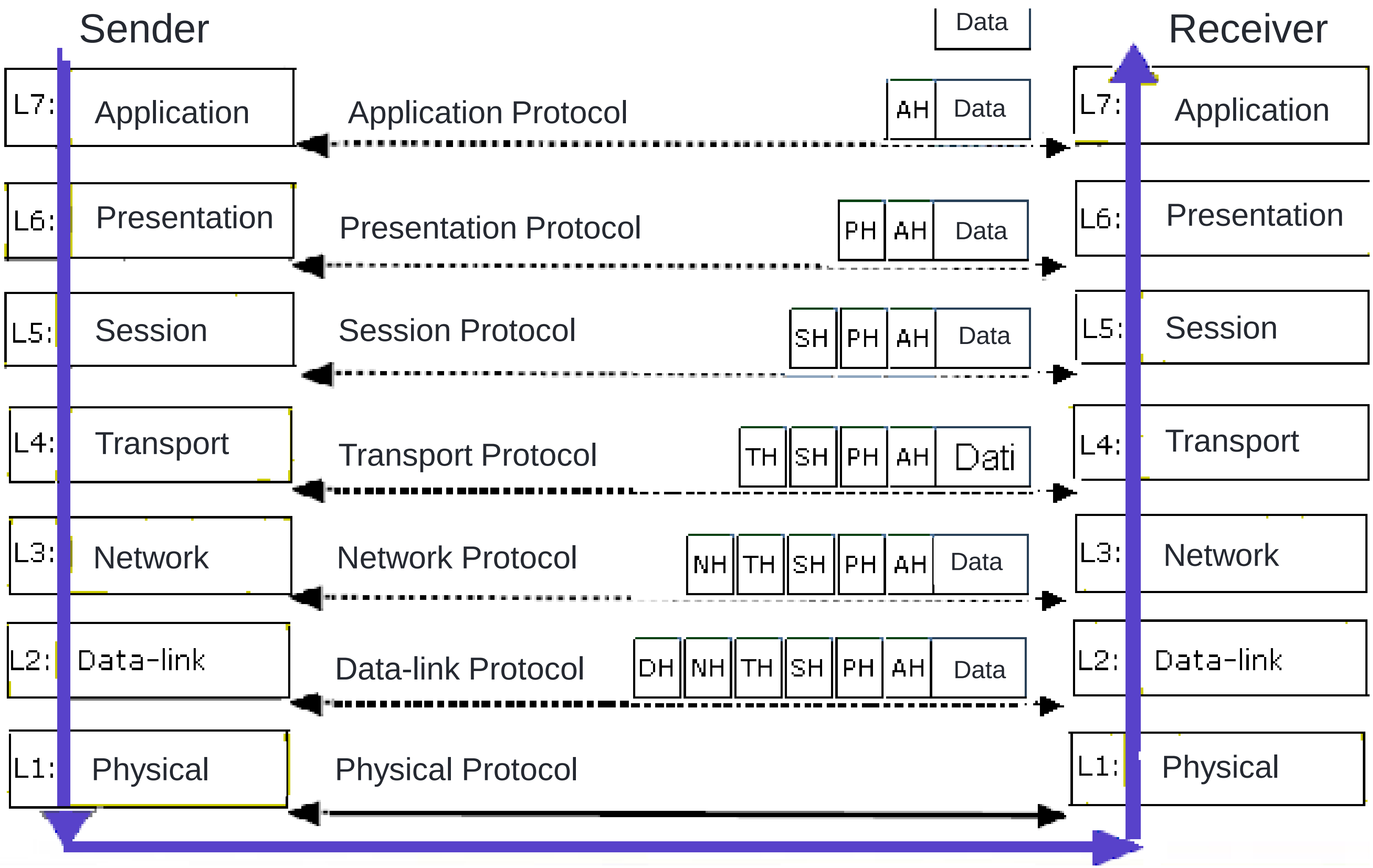


Standard ISO-OSI

- Define a model of the structure of a communication system

- Specific standards can be defined for each layer
  - Not necessary to have one and only one standard for each layer
  - A set of standards can be associated with each layer, each one providing differentfunctionalities

Standard ISO-OSI



Standard ISO-OSI: Application

- Provide an interface towards the user, with a set of distributed services on the network
- Access to services happens by means of a call to primitives similar to system calls for the local system
- The behavior of the communication subsystem is transparent
- Protocols: HTTP, SMTP, FTP, SNMP, Telnet, DNS

Standard ISO-OSI: Transport

- Interface between higher and lower layers

- Provide a system for message transportation independent from the type of network at the lower layers

- Integrity control (reliable transmission)

- Ordering of received packets

Standard ISO-OSI: Transport

- Transport
  - No errors
  - Sequence
  - No loss
  - No duplicates
  - Quality of service
- Protocols: TCP, UDP

Standard ISO-OSI: Network

- It is responsible of opening and ending a connection
- Include functionalities like: routing on the network, addressing and flow control
- When internet is considered, this layer is responsible to put together different networks (internetworking)
- Protocols: IP, ICMP, IPsec, ARP, RIP, OSPF

Simplified OSI

- To make it more suitable to a real network scenario
- At the end of the 80s, OSI trend was towards layers reduction due to the widespread diffusion of the Internet

Interface Reduction

- ISO/OSI model has 7 layers, and 6 interfaces between layers
- Each interface has a set of precise responsibilities
- High modularity but
  - Do not take the evolution of network applications into account
  - Trend towards reduction of the interfaces due to the success of Internet protocols (TCP/IP)

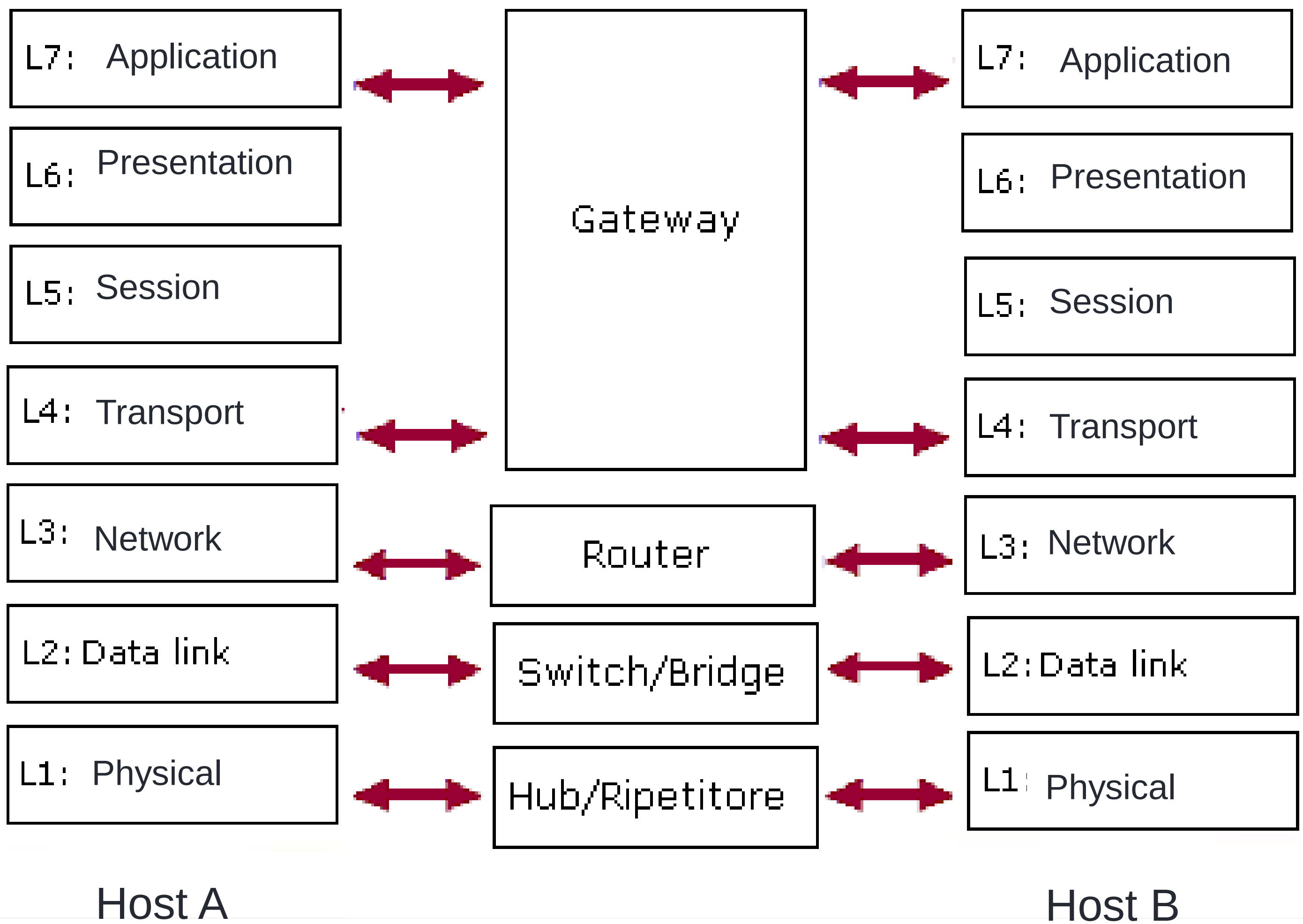
OSI vs TCP/IP model

ISO OSI	TCP/IP
Application	Application
Presentation	
Session	
Transport	Trasport
Network	Network
Datalink	Datalink
Physical	Physical

Simplified OSI (TCP/IP model)

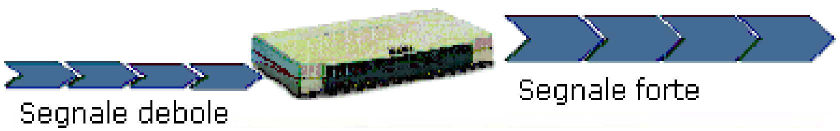
- Application layer: real applications
- Transport layer: add functionalities like reliability and fault tolerance
- Network layer: basic communication between heterogeneous networks

Interconnection Devices and ISO/OSI

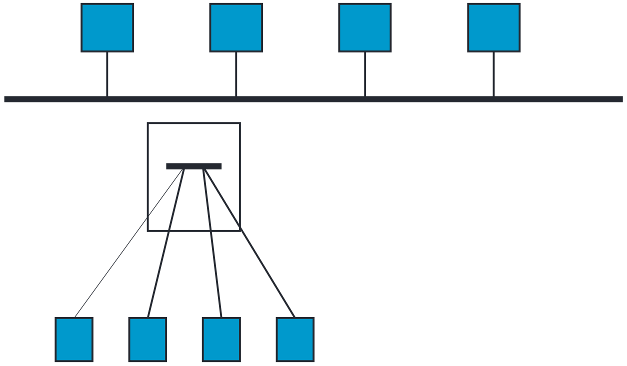


Interconnection Devices

- **Repeater**: device at physical layer, which restores data and collision signal
  - Digital amplifier
  - Extend the lenght of the network
  - Permits the traffic to traverse the LAN segments

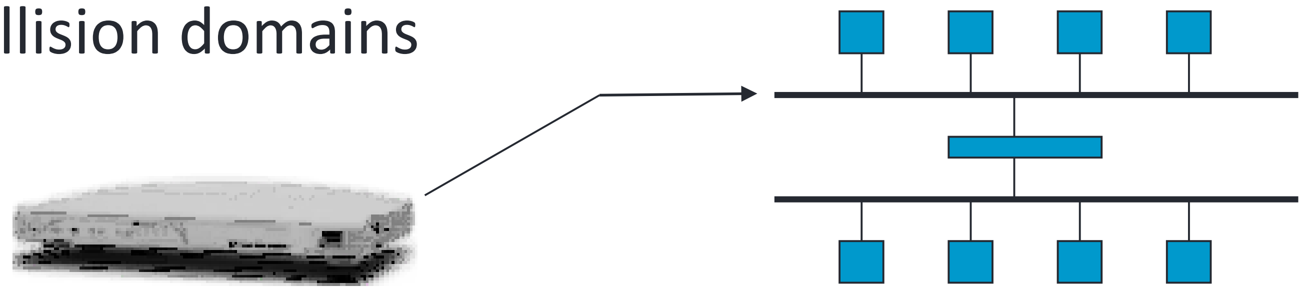


- **Hub**: multi-port repeater at physical layer, with fault detector

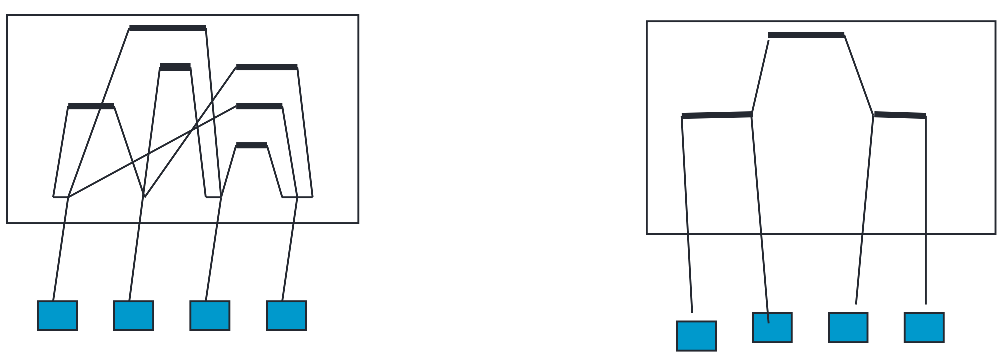


Interconnection Devices

- **Bridge**: device at data link layer, connecting two or more collision domains



- **Switch**: multi-port bridge with parallel active path
  - Smart hub: read the address of the destination and use this information to forward the frame
  - Avoid collisions thanks to independent paths



Switch

- A switch selects a path or line and send every frame to its destination
  - A switch is simpler than a router

Router

- Identify the next node of the network to which a packet should be sent in the path towards final destination
- Use information of layer 3 network protocols in the packets to route them from one LAN to another
  - It means the routed must know all protocols at network layer that can be used by the corresponding networks
  - Mainly used in TCP/IP network: use IP addresses for routing
- Communicate among them to identify the best path to increase velocity and reduce network traffic (like a global navigator)

Gateway

- Used to interconnect networks with different protocols
- Work at network and upper layers of the ISO/OSI
- To communicate with a host residing on another network, we need to configure a router towards that network
- If not exist, a gateway is used (default IP router)
- If no gateway exists, only communications in local network are possible
- Gateway receives data from a network with a specific protocol stack, remove the protocol stack and reconstruct the message using the network protocol of the destination network

IP, TCP, UDP (READ ONLY)

Network Layer: IP

- Protocol for the delivery of packets from a host sender to a host receiver
  - Unique identifier for each host (IP address)
  - Logic communication between hosts
- But
  - *No connection*: every packet is treated independently from the others
  - *Not reliable*: the delivery is not guaranteed (packets can be lost, duplicated, delayed or delivered in the wrong order)
  - *Delivery with commitment*: attempt to deliver every packet (unreliability due to network congestion or node/router failures)

IP Protocol

- IP protocol provides a connectionless and unreliable datagram service
- Unreliable means that there are no guarantees that an IP packet reaches its destination (best effort service)
- Connectionless means that IP protocol does not maintain information on the state of the forwarded packets
- Every packet is treated independently from each other
  - IP datagrams can be delivered out of sequence

Header IP

<table>
<tr><td>4bit version</td><td>4b head. length</td><td>8bit type of service (TOS)</td><td colspan="2">16bit total length (in bytes)</td></tr>
<tr><td colspan="3">16bit identification</td><td>3bit flags</td><td>13bit fragment offset</td></tr>
<tr><td colspan="2">8bit time to live (TTL)</td><td>8bit protocol</td><td colspan="2">16bit header checksum</td></tr>
<tr><td colspan="5">32bit source IP address</td></tr>
<tr><td colspan="5">32bit destination IP address</td></tr>
<tr><td colspan="5">options, data</td></tr>
</table>

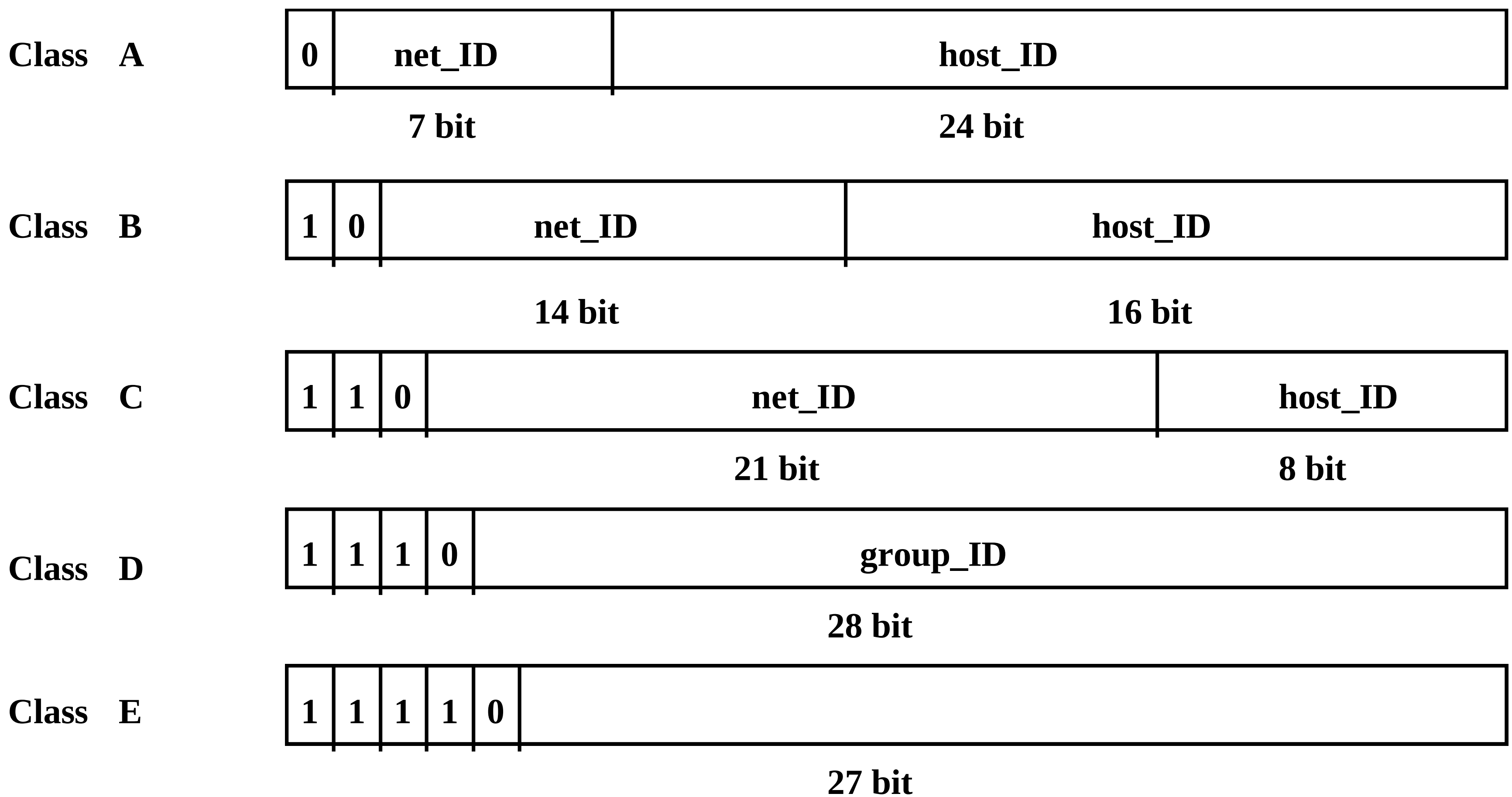
Internet

- Internet or network of networks have these properties
  1. PC on local networks (*subnets*) can communicate among them
  2. Data link layer of the subnets can be heterogeneou (e.g., Ethernet, Token Ring)
  3. Can include an unlimited number of hosts, within the limits granted by the maximum number of hosts that can be connected to each subnet

IP Address

- Every network interface has a unique IP address of fixed length (4 bytes = 32 bits)
- IP addresses are a finite resource (IPv4 and IPv6 addresses)
- Five classes of IP addresses: A, B, C, D and E
  - D for broadcast communications, E is not used
- IP addresses use prefix
  - The network prefix (netid) of an IP address tells the network to which the interface connects

Classes of IP Addresses



Classes of IP Addresses

- Different classes differs by the prefix, the different distribution of the net_ID of the local network and host_ID of the network card
- Every addressing is given on the basis of the number of machines that connect to the local network
  - Address of Class C: first 24 bits are fixed (21 represents the network) and 8 bits are free, permitting to identify at most 256 maccines (with the same net_ID)
- Addresses assignedusing pointed decimal notation (e.g., 196.20.44.2). Every number identify the content of one byte of the IP address
- The decimal valueof the first byte can be used to identify the class
  - Up to 127 is class A, from 128 to 191 is class B, from 192 to 219 is class C and so on

Subnet Mask

- The division in classes of the IP addresses provides three standard models of partitioning the IP address (32 bit) between net_id and host_id
- Not always practice and useful
- We can obtain a different partition between host_id and net_id associating a *subnet mask*, defined as the binary number that put in *AND with the IP address provides the real net_id*

Subnet Mask

- All three classes of IP address are associated with default masks

Class	Default Subnet Mask
A	255.0.0.0
B	255.255.0.0
C	255.255.255.0

- Applying the subnet masks we obtain a prefix equivalent to the net_id

TCP/IP

- Transmission Control Protocol/Internet Protocol (TCP/IP) is a network software supporting applications in the commmunication by means of a protocol that is routable, the same used in Internet
- TCP/IP defines a fixed header and three special headers
  - A simple one, best effort (User-Datagram, UDP)
  - A complex one for a reliable flow service (TCP)
  - One for control messages (Internet Control Message Protocol – ICMP)

TCP

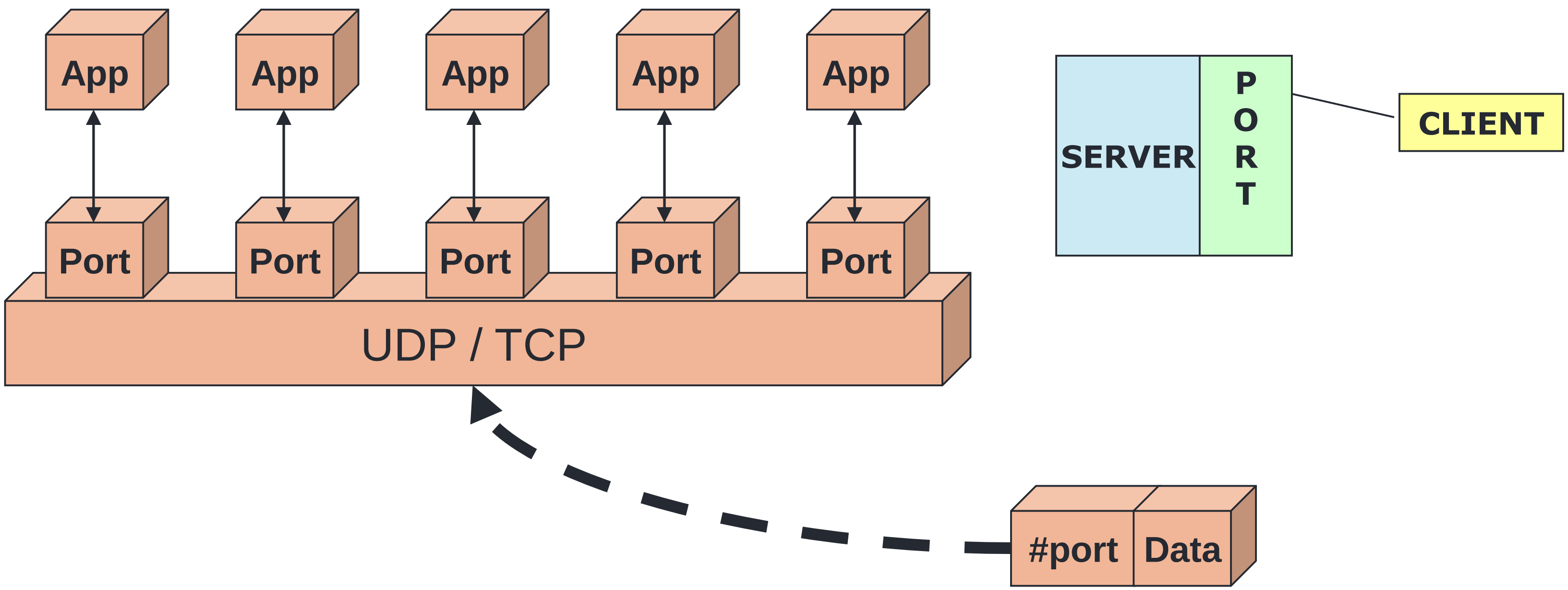
- TCP is a connection-oriented protocol
- It must guarantee two main conditions
  1. The destination is reachable
  2. All packets sent by the sender reaches their destination
- To this aim, TCP protocol needs of additional information with respect to the ones contained in the IP header
  - Additional header for each IP packet to be sent

TCP

<table>
<tr><td colspan="8">16bit source port number</td><td>16bit destination port number</td></tr>
<tr><td colspan="9">32bit sequence number</td></tr>
<tr><td colspan="9">32bit acknowledgment number</td></tr>
<tr><td>4b head. length</td><td>6bit (reserved)</td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td>16bit window size</td></tr>
<tr><td colspan="8">16bit TCP checksum</td><td>16bit urgent pointer</td></tr>
<tr><td colspan="9">options, data</td></tr>
</table>

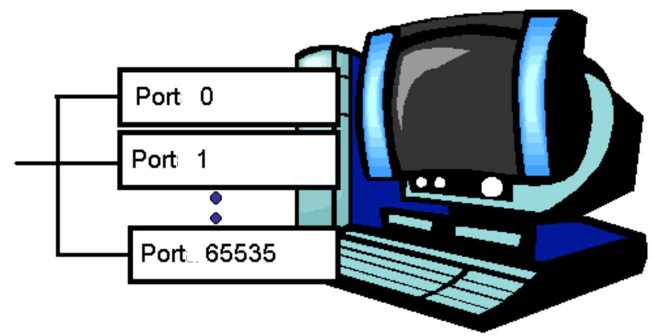
TCP Ports

- TCP and UDP use ports to map data in input with a particular process active on a PC
- Each socket is bound to a port number such that the TCP layer can identify the destination application to which data must be delivered



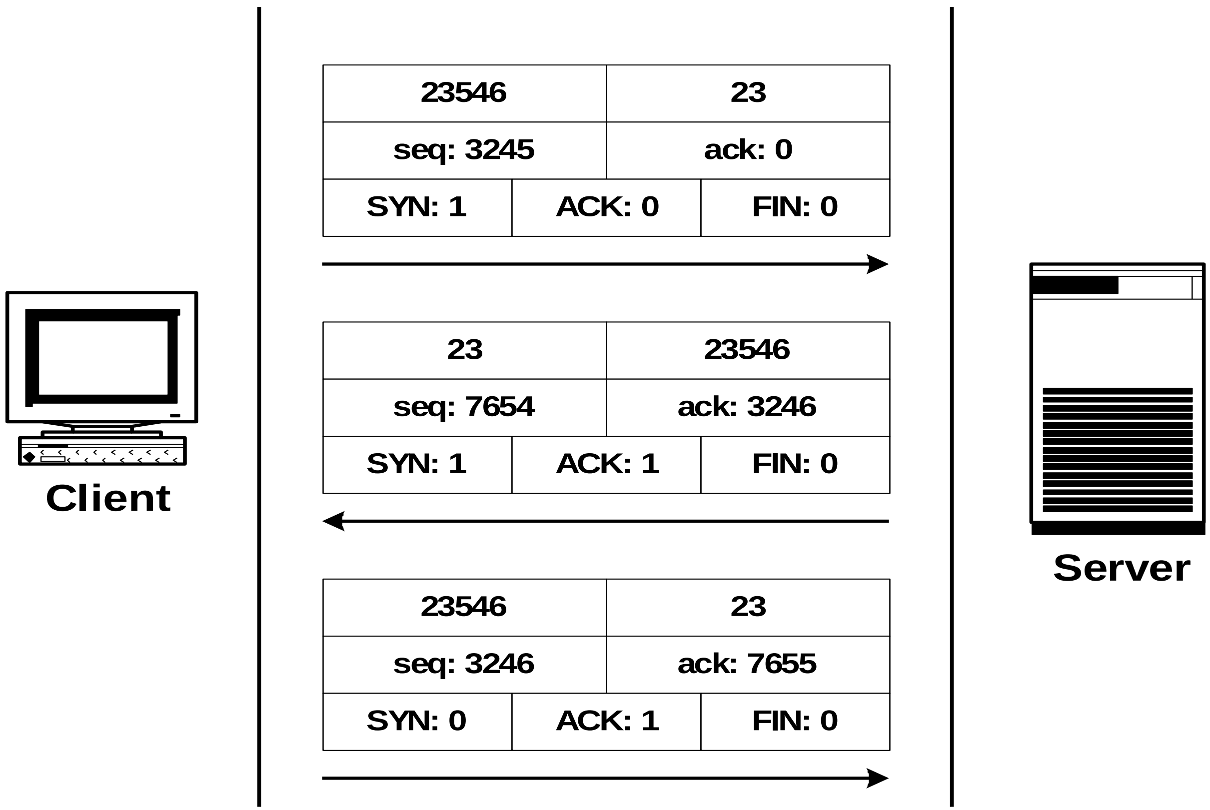
Well-Known Ports

- Ports are represented as positive integers (16 bit)
- Represent a point of connection betweem physical and application layers; represent a communication channel
- Some ports are reserved for well-known services
  - ftp -> 21/tcp
  - telnet -> 23/tcp
  - smtp -> 25/tcp
  - http -> 80/tcp
  - login -> 513/tcp
- Processes and services at user level use port numbers >=1024



Threeway handshake

- Used by TCP to establish a connection between two hosts



UDP

- A protocol working at the same layer as TCP but not oriented to the connection
- There is no connection phase
- Packets are sent without knowing whether the application at the destination is ready to receive them
- Simplified header

Header UDP

- UDP adds only source and destination ports to IP packets to support application-layer communication

16bit source port number	16bit destination port number
16bit UDP length	16bit UDP checksum
data	

TCP vs UDP

- TCP
  - Connection oriented
  - Reliable transport
  - Flow management
  - Congestion control
- UDP
  - Unreliable
  - Does not guarantee flow management nor congestion control

Transport Layer: Which Protocol to Use

- Data loss
  - Some apps (e.g., audio) can tolerate some loss
  - Other apps (e.g., file transfer, telnet) require reliable connection
- Timing
  - Some apps (e.g., VOIP, interactive game) require low delay
- Bandwidth
  - Some apps (e.g., multimedia) require a minimum amount ofbandwidth to be efficient
  - Some apps (“elastic apps”) use the available bandwidth

Transport Layer: Which Protocol to Use

Application	Data loss	Bandwidth	Time Sensitive
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
Web documents	no loss	elastic	no
real-time audio/video	loss-tolerant	audio: 5kbps-1Mbps video:10kbps-5Mbps	yes, 100's msec
stored audio/video	loss-tolerant	same as above	yes, few secs
interactive games	loss-tolerant	few kbps up	yes, 100's msec
instant messaging	no loss	elastic	yes and no

Transport Layer: Which Protocol to Use

Application	Application layer protocol	Underlying transport protocol
e-mail	SMTP [RFC 2821]	TCP
remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	proprietary (e.g. RealNetworks)	TCP or UDP
Internet telephony	proprietary (e.g., Dialpad)	typically UDP

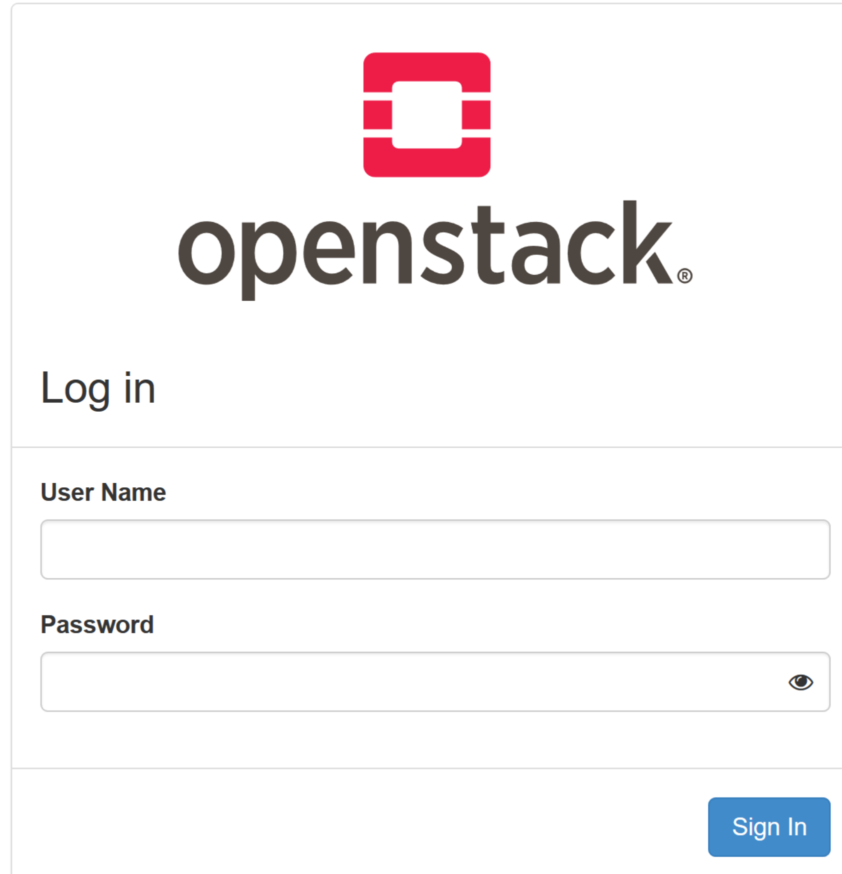
Open Stack Lab

Goals

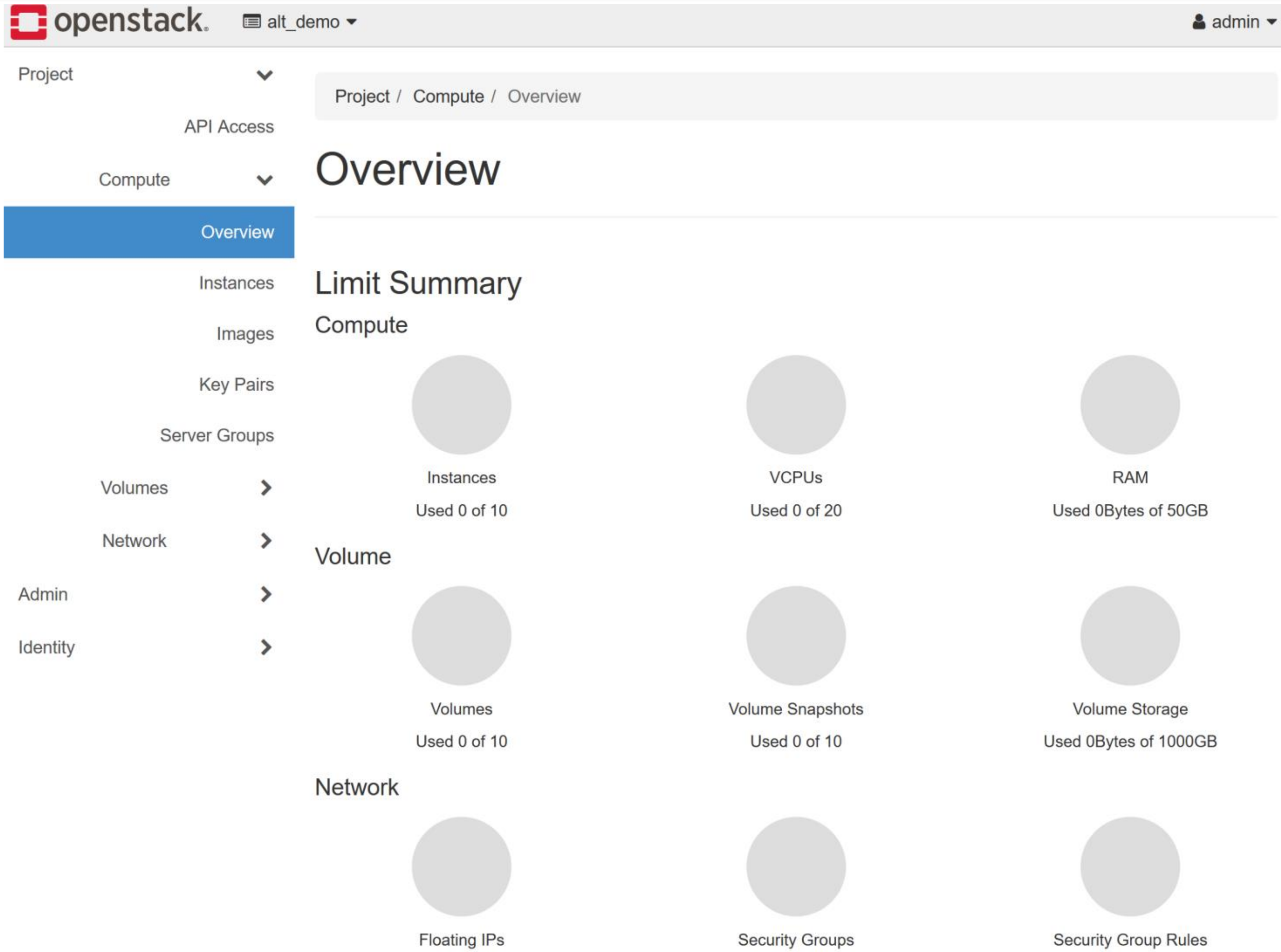
- Access a tenancy on OpenStack
- Create a virtual machine
- Link a virtual machine to a virtual network reachable from the external network
- Install a Web server on a virtual machine
- Access index.html

OpenStack: Login

- URL
- Username
- Password



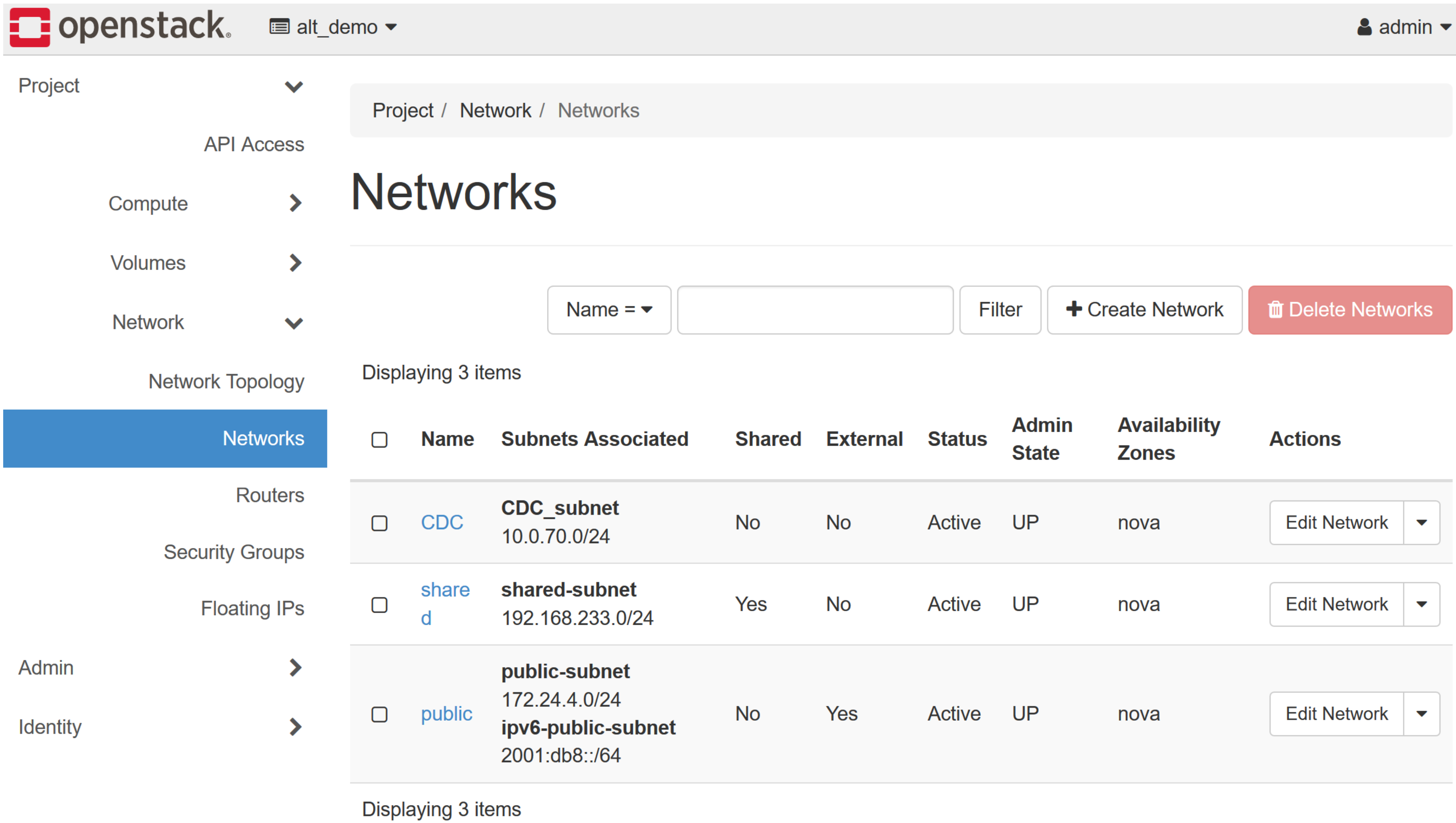
OpenStack: Dashboard



Create a virtual network: Subnet

- Tab network -> networks -> create network
  - Add network name
    - Admin status UP
    - We deployed the switch that will connect the virtual machines
  - Subnet menu: create the subnet to be associated to the network
    - Add name
    - Network address: for example 10.0.70.0/24
    - IPV4
    - Gateway 10.0.70.1
  - Subnet detail
    - Enable dhcp
    - Name server DNS 8.8.8.8

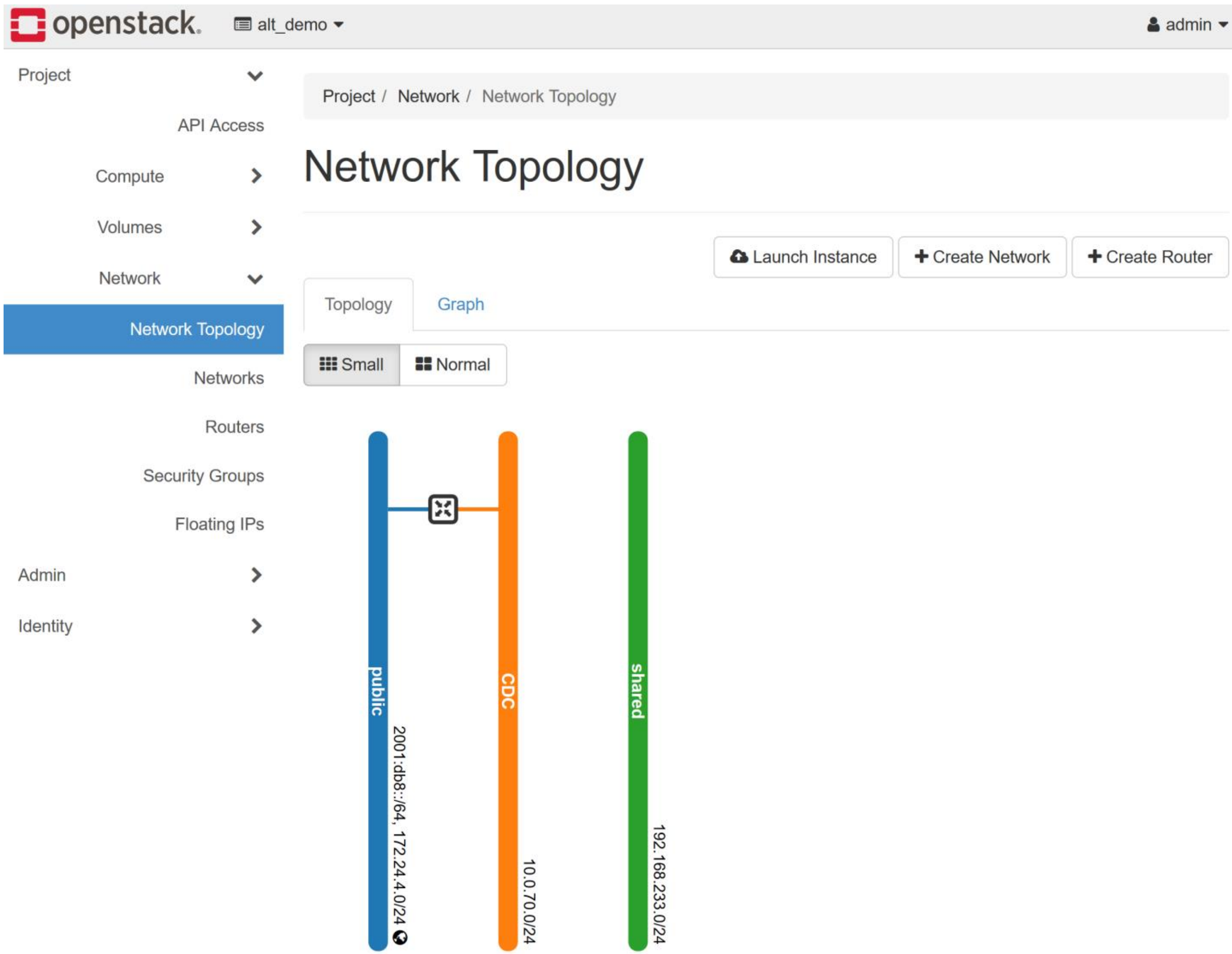
Create a virtual network: Subnet



Create a virtual network: Router

- Create router
  - Add name
  - Add external network
- Left mouse on router, interface, add interface (menu «Network Topology»)
  - Select the new subnet (our network)
  - Click submit
- Back to network topology
  - On router select view router details
  - Click on set gateway
  - Select public-subnet (external network)
- Click on router details and check whether fixed ip is the one of the default gateway
- Back on network topology (FINE)

Create a virtual network: Router



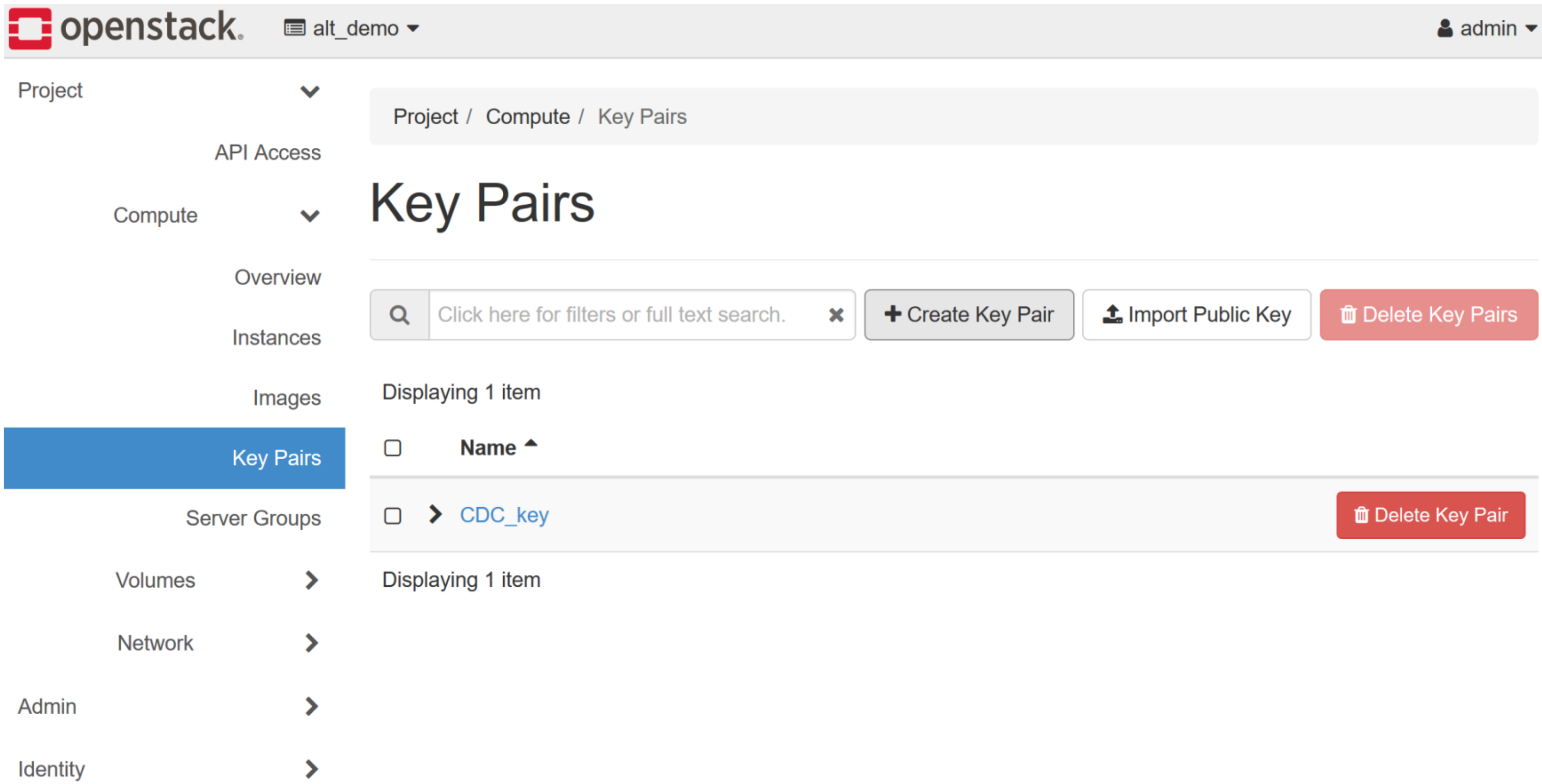
Create a virtual network: VM

- Goal: create a VM reachable from the outside using a secure connection (SSH)

- Create a SSH key used by the VM to comunicate
  - Menu compute -> key pairs
    - Create key pairs
    - Add key nameInserire nome della chiave

- We have a ready-to-be-used security key

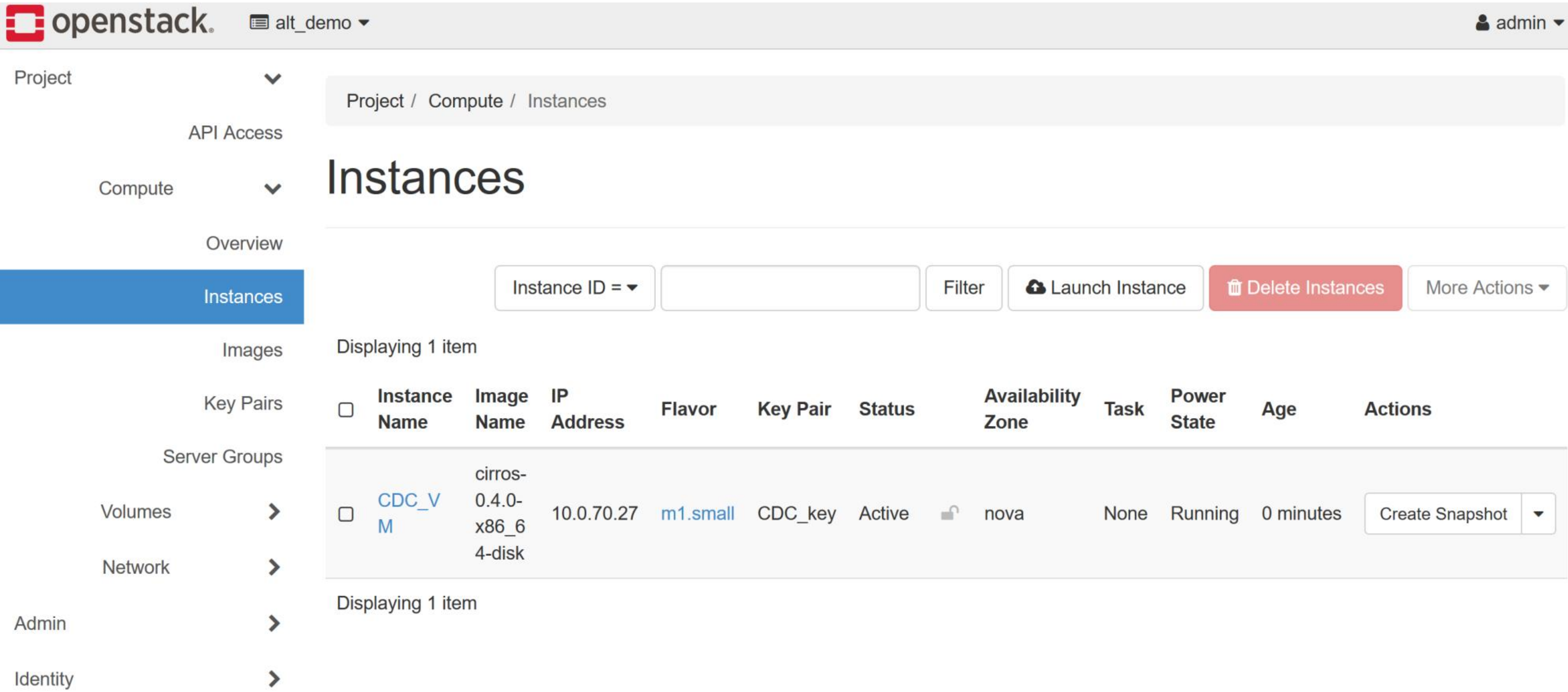
Create a virtual network: VM



Create a virtual network: VM

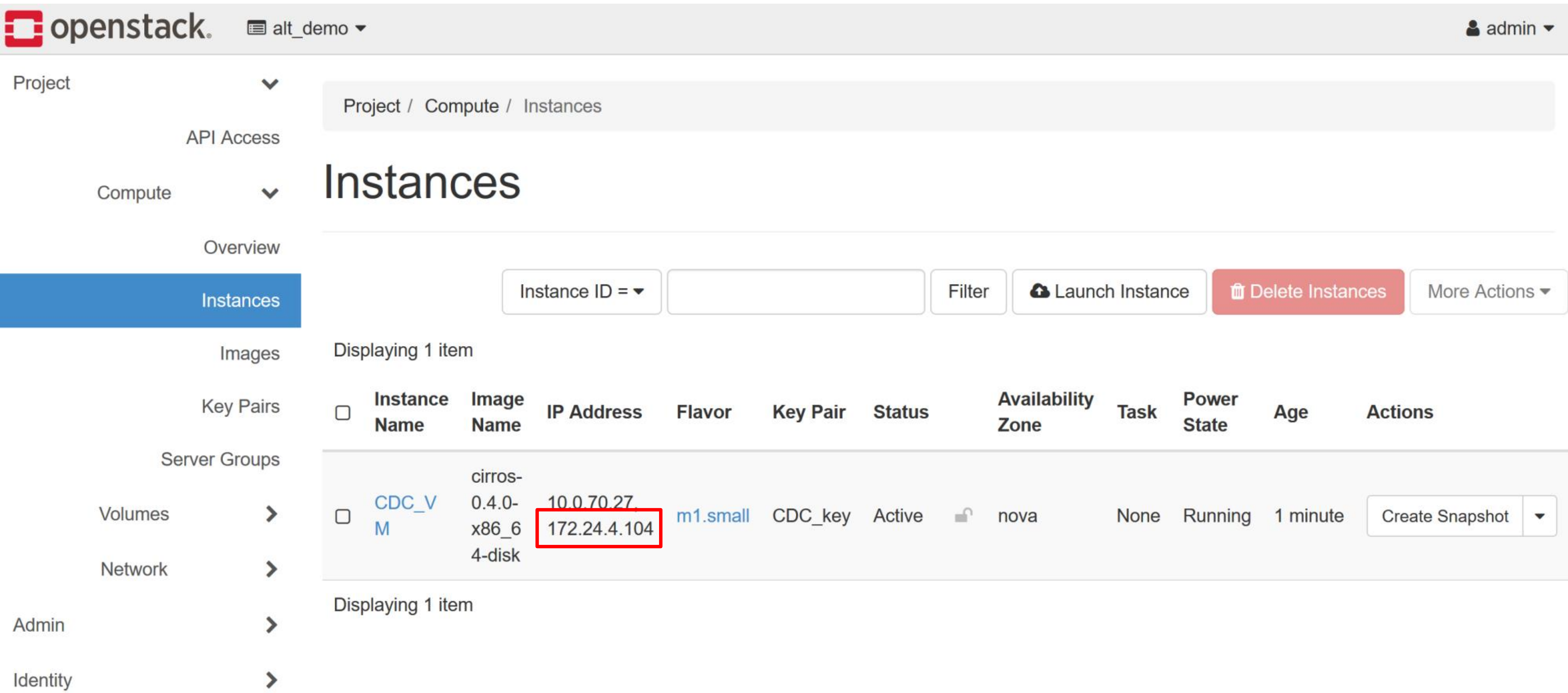
- Menu instances to create a VM instance
  - Launch instance
  - Menu details
    - Add name
    - Boot from image or boot image source
    - Select Cirros image
    - Select m1.small
  - Menu Key Pair
    - Select the created key pair
  - Menu networks to add the created network
    - Select our network
    - Launch

Create a virtual network: VM



Create a virtual network: VM

- Select associate floating IP to the new machine
  - Floating IP makes the VM visible to the external network



Create a virtual network: Privileges

- Modify the default security group to permit access from the external network and enable SSH

- Tab network -> security groups
  - Manage default security group
  - Add a rule
    - Rule predefined tcp -> SSH, HTTP

Create a virtual network: Permessi

openstack. alt_demo ▾ admin ▾

- Project
  - API Access
  - Compute
  - Volumes
  - Network
    - Network Topology
    - Networks
    - Routers
    - Security Groups
    - Floating IPs
- Admin
- Identity

Project / Network / Security Groups / Manage Security Group Rul...

Manage Security Group Rules: default (700c372d-ff36-492a-a90a-3e6ca42e1b41)

+ Add Rule | Delete Rules

Displaying 5 items

Direction	Ether Type	IP Protocol	Port Range	Remote IP Prefix	Remote Security Group	Description	Actions
Egress	IPv4	Any	Any	0.0.0.0/0	-	-	Delete Rule
Egress	IPv6	Any	Any	::/0	-	-	Delete Rule
Ingress	IPv4	Any	Any	-	default	-	Delete Rule
Ingress	IPv4	TCP	22 (SSH)	0.0.0.0/0	-	-	Delete Rule
Ingress	IPv6	Any	Any	-	default	-	Delete Rule

Displaying 5 items

Create a virtual network: Connect to VM

- Use SSH PUTTY client and its tool PUTTYgen for key generation
- Generate a key for communication
  - Start puttygen
  - Open file .pem downloaded when created the key
  - Load the key
    - Save private key
- Access using putty
- Session folder
  - Add session name
  - SSH destination: floating ip used for the VM created in Open Stack
- Connection folder->ssh->auth
  - Add private key file for authentication
- Execute putty
  - Insert default username for cirrus: cirros
  - Insert default password for cirrus: gocubsgo

Create a virtual network: Install apache2 (ubuntu)

- sudo apt install apache2
  - If not working edit /etc/resolv.conf adding 8.8.8.8 as DNS
  - If not working «sudo apt-get update»
  - sudo nano /etc/resolv.conf
    - Add «nameserver 8.8.8.8»

- sudo service apache2 start
  - If not working «unable to resolve host» modify /etc/hosts
  - sudo nano /etc/hosts and add
    - 127.0.0.1 machine_name

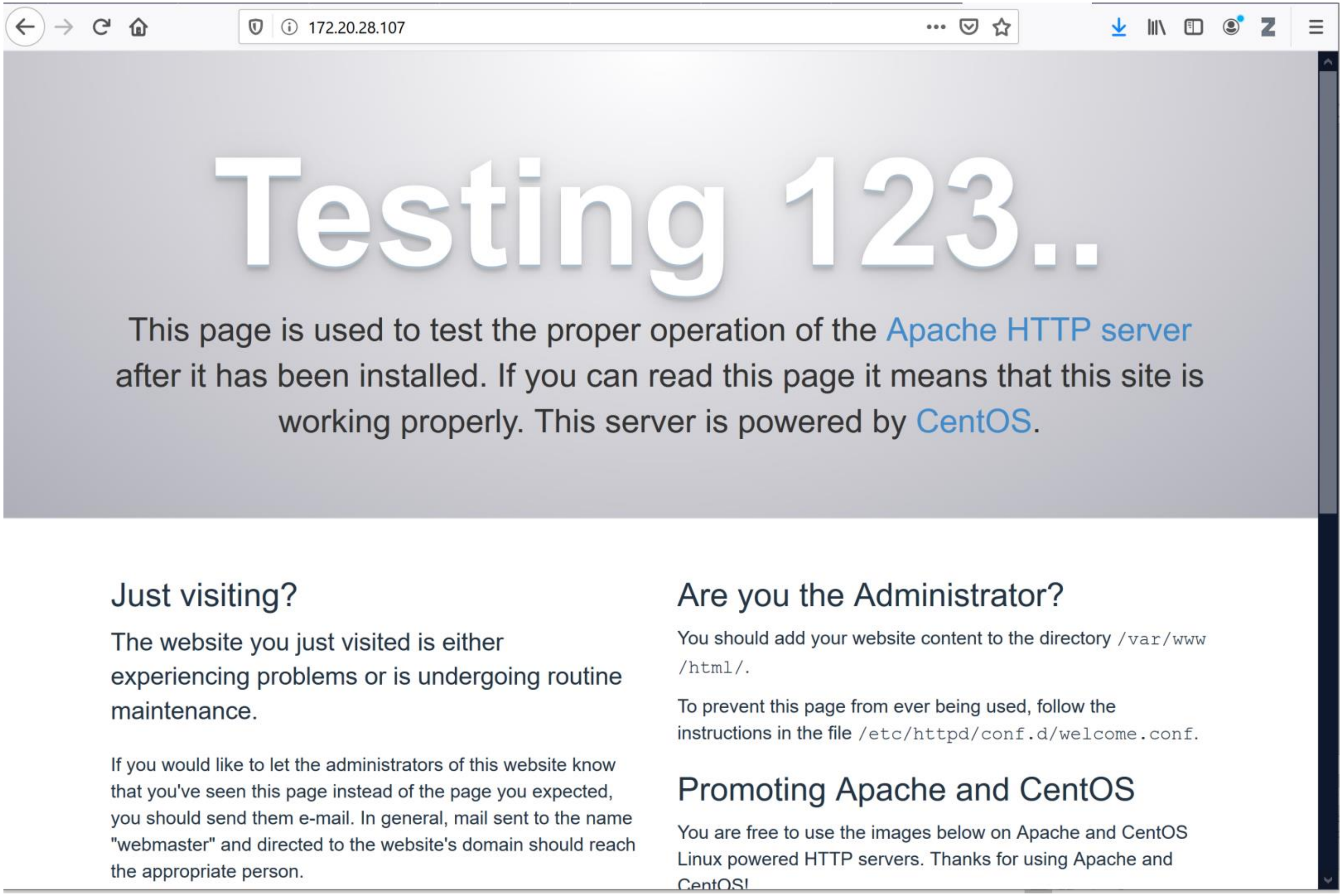
- Access http://floating_ip/index.html

- sudo service apache2 start

Create a virtual network: Install apache (centOS)

- Log in with username centos
- sudo yum –y update
- sudo yum install httpd
- sudo service httpd start
- Access http://floating_ip/index.html

Create a virtual network: Install apache (centOS)



Conclusions

- We configured a tenancy creating a network and a VM that can be accessed from the external network
- We created a secure connection with putty and managed a remote VM
- We installed HTTPD and accessed its index.html

Lesson 3.1: Big Data PaaS

Claudio Ardagna – Università degli Studi di Milano

Cloud and Distributed Computing

Scenario

- A huge amount of data are generated and collected every minute (sensors)
  - 1.7 million billion bytes of data, over 6 megabytes for each human for minute (2016)
  - 2.5 quintillion bytes of data created each day
    - IDC predicts that *by* 2025, the total *amount* of digital *data* created worldwide will rise to 163 zettabytes (1billion terabytes = 10^{21} bytes)
  - The trend is rapidly accelerating with the growth of the Internet of Things (IoT), 200 billions of connected devices by 2020

- Low latency access to huge distributed data sources has become a value proposition

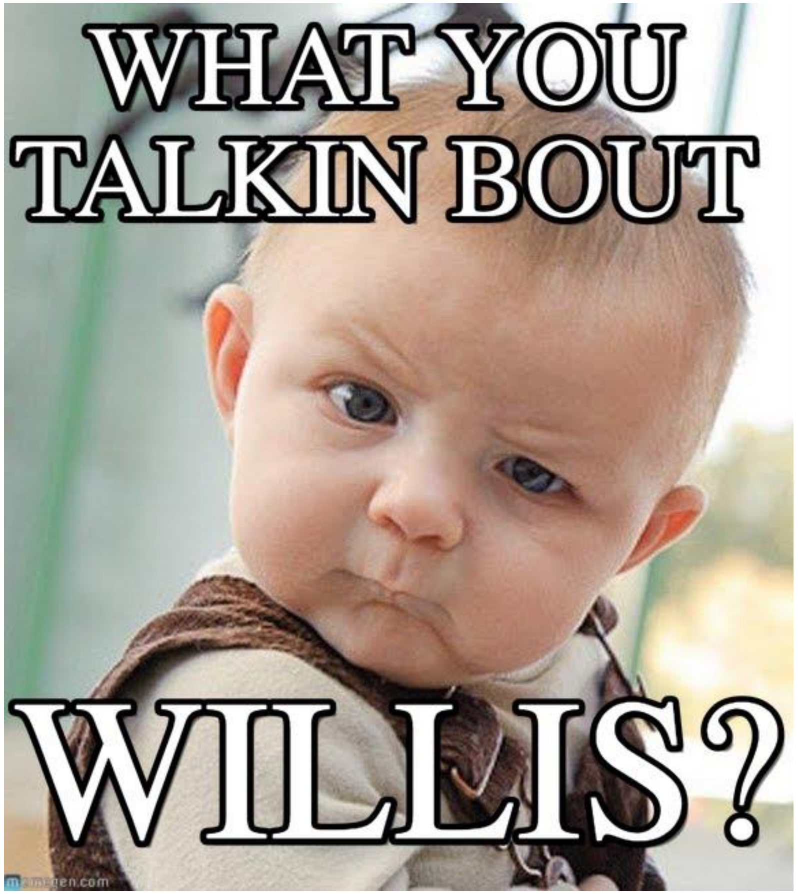
- Business intelligence applications require proper big data analysis and management functionalities

Quintillion?!?

- 2.5 Quintillionbyte
- 2,500,000 Terabytes
- 2,500,000,000,000,000,000 Bytes

Quintillion?!?

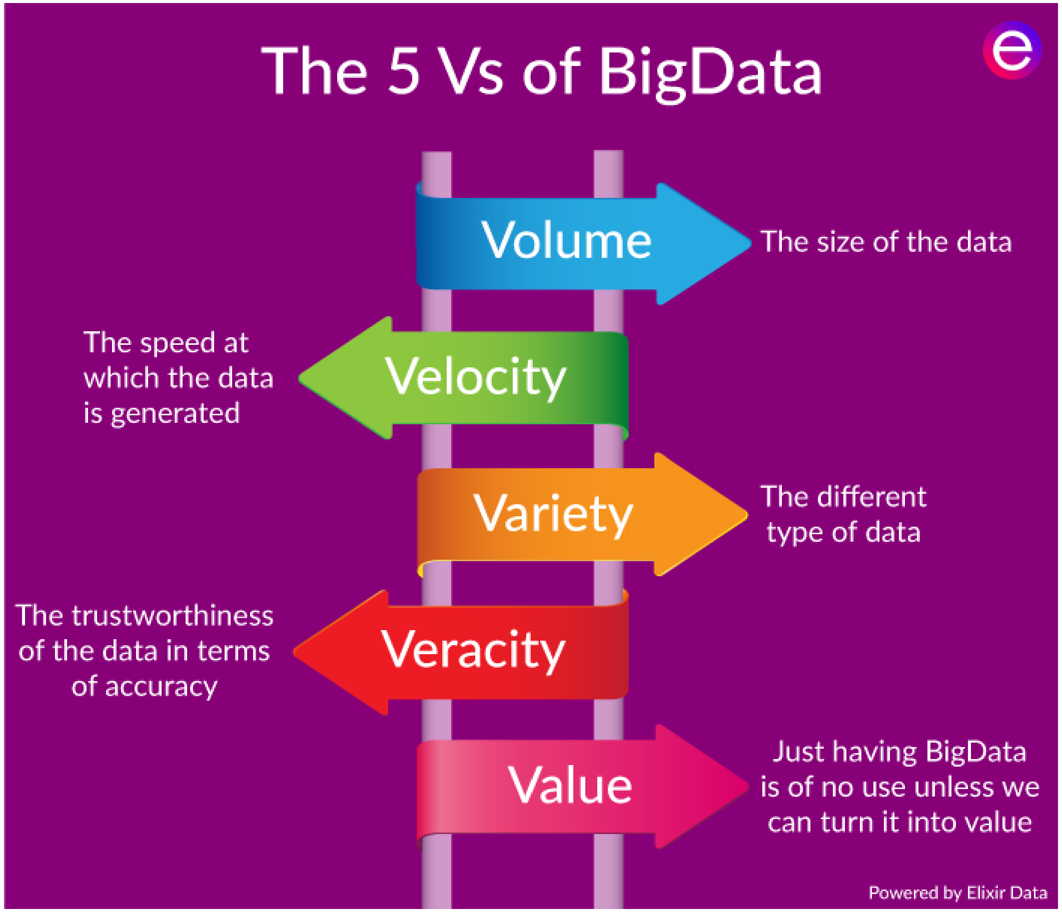
- 2.5 Quintillionbyte
- 2,500,000 Terabytes
- 2,500,000,000,000,000,000 Bytes



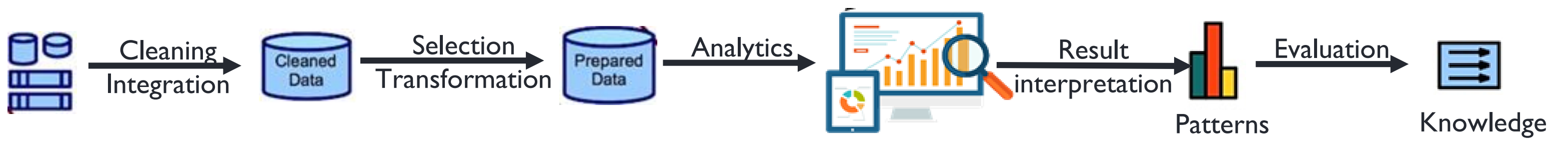
Quintillion?!?

https://www.forbes.com/sites/nicolemartin1/2019/08/07/how-much-data-is-collected-every-minute-of-the-day/

The Five Vs


The 5 Vs of BigData
Volume
The size of the data
Velocity
The speed at which the data is generated
Variety
The different type of data
Veracity
The trustworthiness of the data in terms of accuracy
Value
Just having BigData is of no use unless we can turn it into value
Powered by Elixir Data

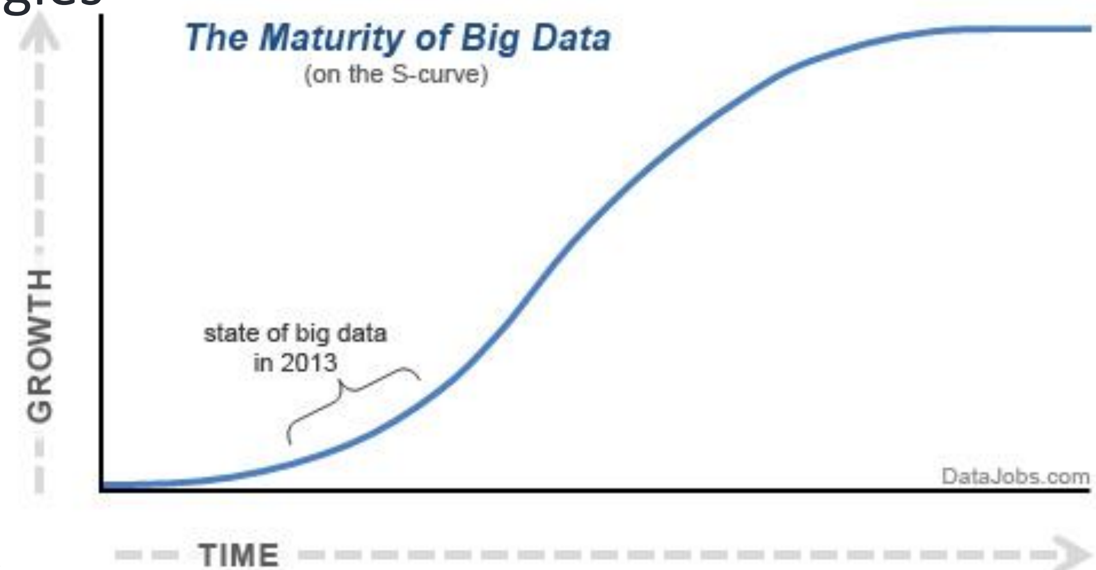
The data analytics pipeline and areas



Data representation	Specify **how data are represented**: NoSQL, Graph-based, Relational, Extended relational, Markup based, Hybrid
Data preparation	Specify **how to prepare data for analitycs**: anonymize, reduce dimensions, hash
Data analytics	Specify the **expected outcome**: descriptive, prescriptive, predictive
Data processing	Specify **how data will be routed and parallelized**, and **how the analytics will be computed**: parallel batch, stream, hybrid
Data display and reporting	Specify the **display and reporting of the results**: scalar, multi-dimensional

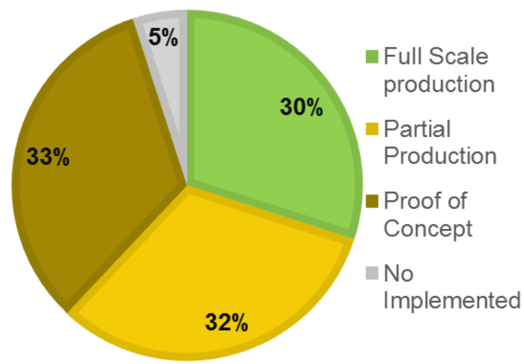
Status on Big Data implementation

- Big data technologies have grown tremendously in past few years
- Industry-wide adoption for big data has been phenomenal which led to the increase in demand but shortage in supply of talented professionals in this field
- Jump on Big Data bandwagon behavior has created more semi-skilled people and few who has in-depth command on these technologies



Big Data Implementation Challenges

- Only 30% businesses has Big Data insights fully integrated into their operations
- Many businesses (38%) are struggling even with proofs of concept
- Key challenges associated with the development and management of Big Data initiatives:



BIG DATA IMPLEMENTATION STATUS 2016

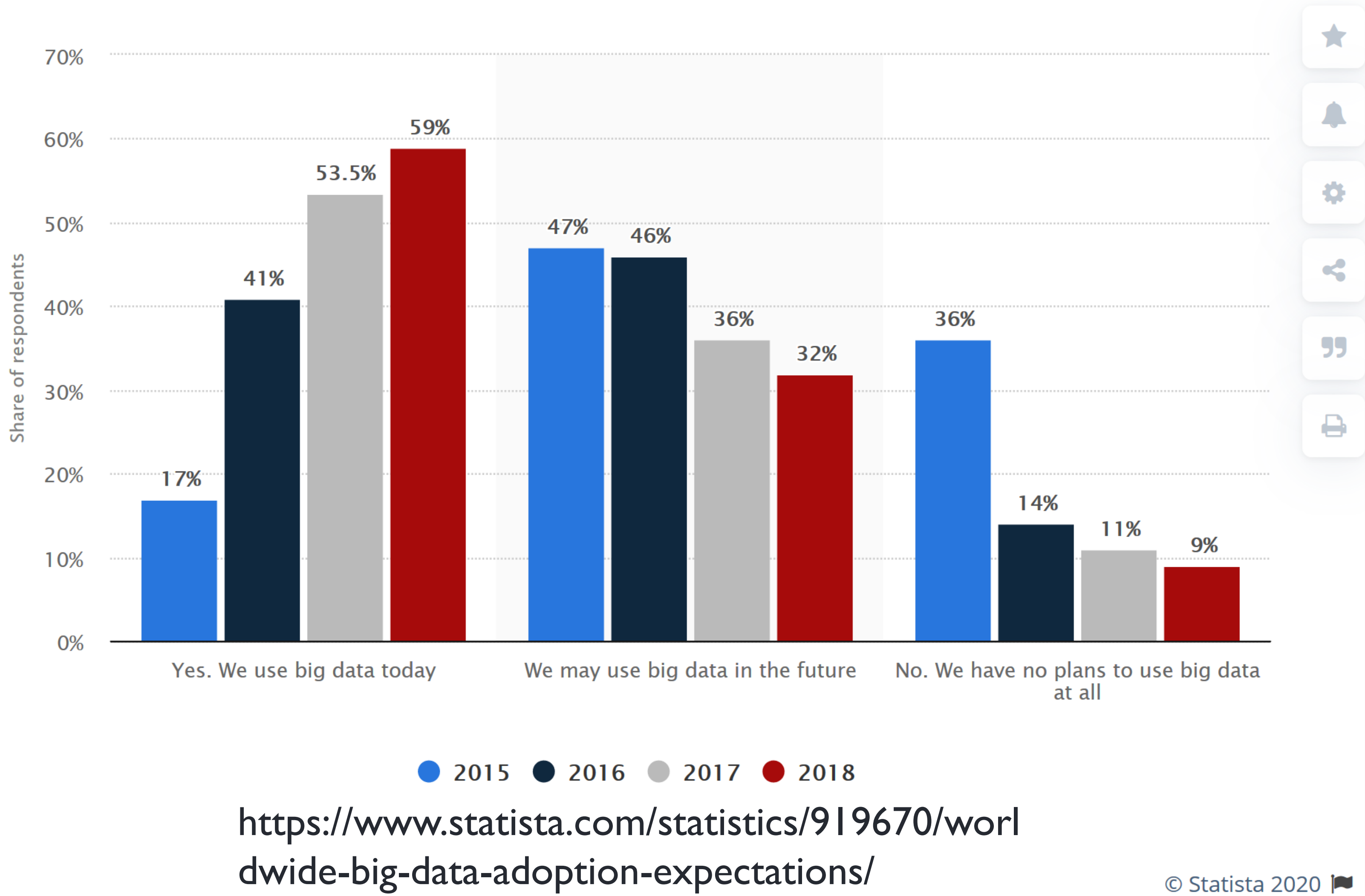
Lack of skills & clarity on Big Data technology

Lack of general Architecture & Lack of standard processes

Ineffective governance models

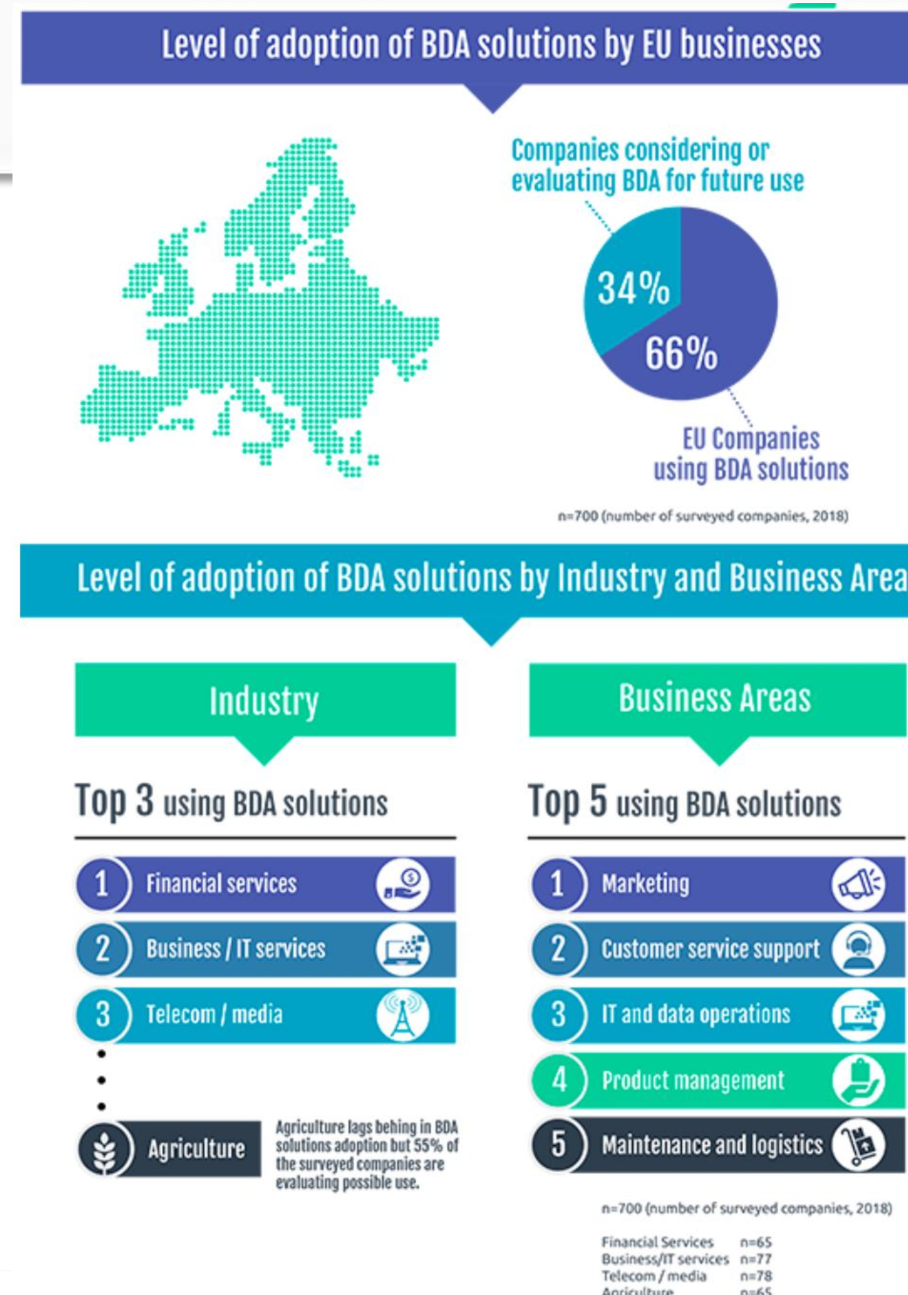
NewVentage Partner (NVP) BigData Executive Survey2016: represented 44 Fortune 1000 and leading firms

Big Data Adoption



https://www.statista.com/statistics/919670/worldwide-big-data-adoption-expectations/

Big Data Adoption



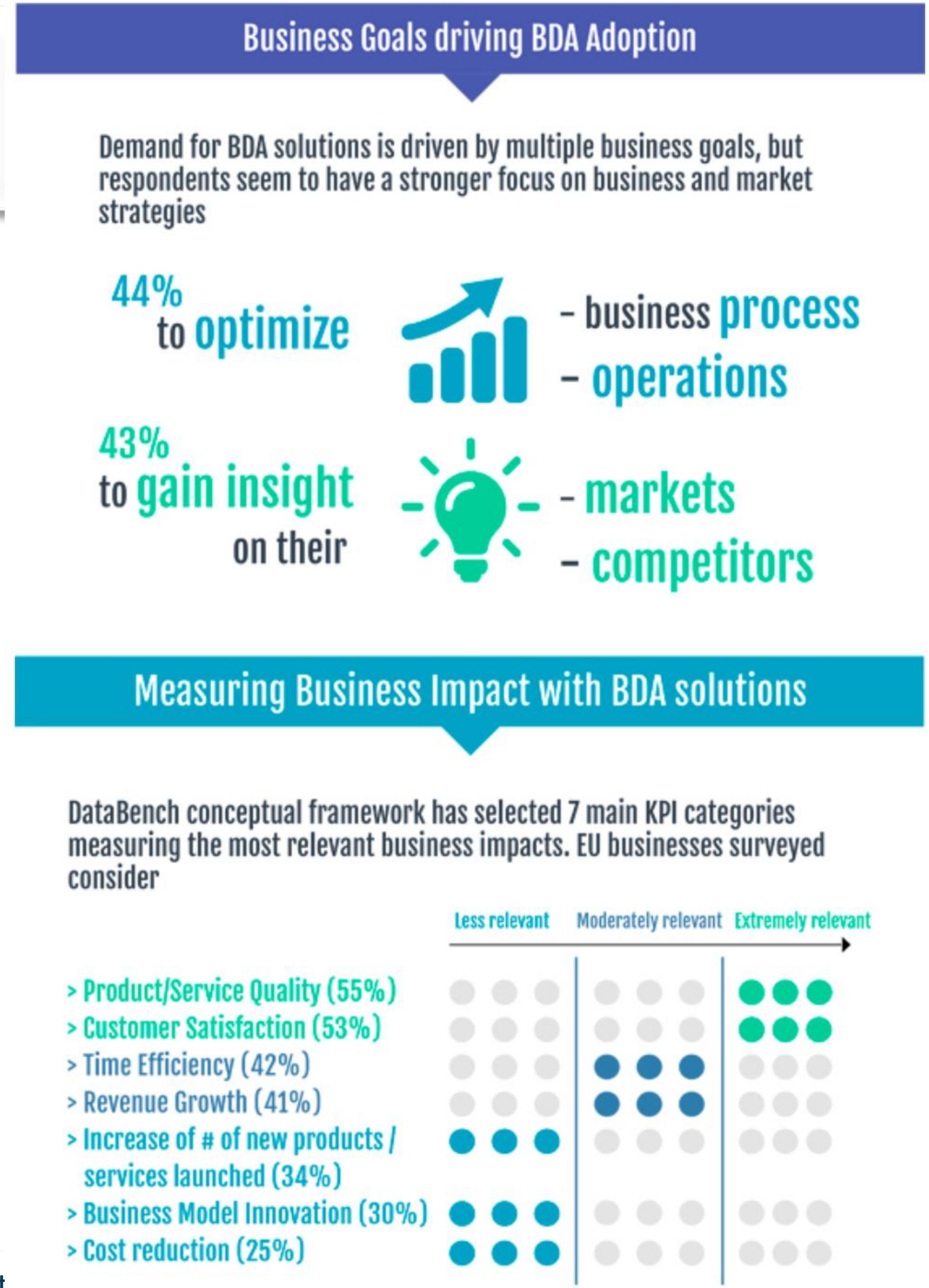
DataBench infographic based on a survey on 700 European companies

https://www.big-data-value.eu/databench-infographic-based-on-a-survey-on-700-european-companies/

Big Data Adoption

DataBench infographic based on a survey on 700 European companies

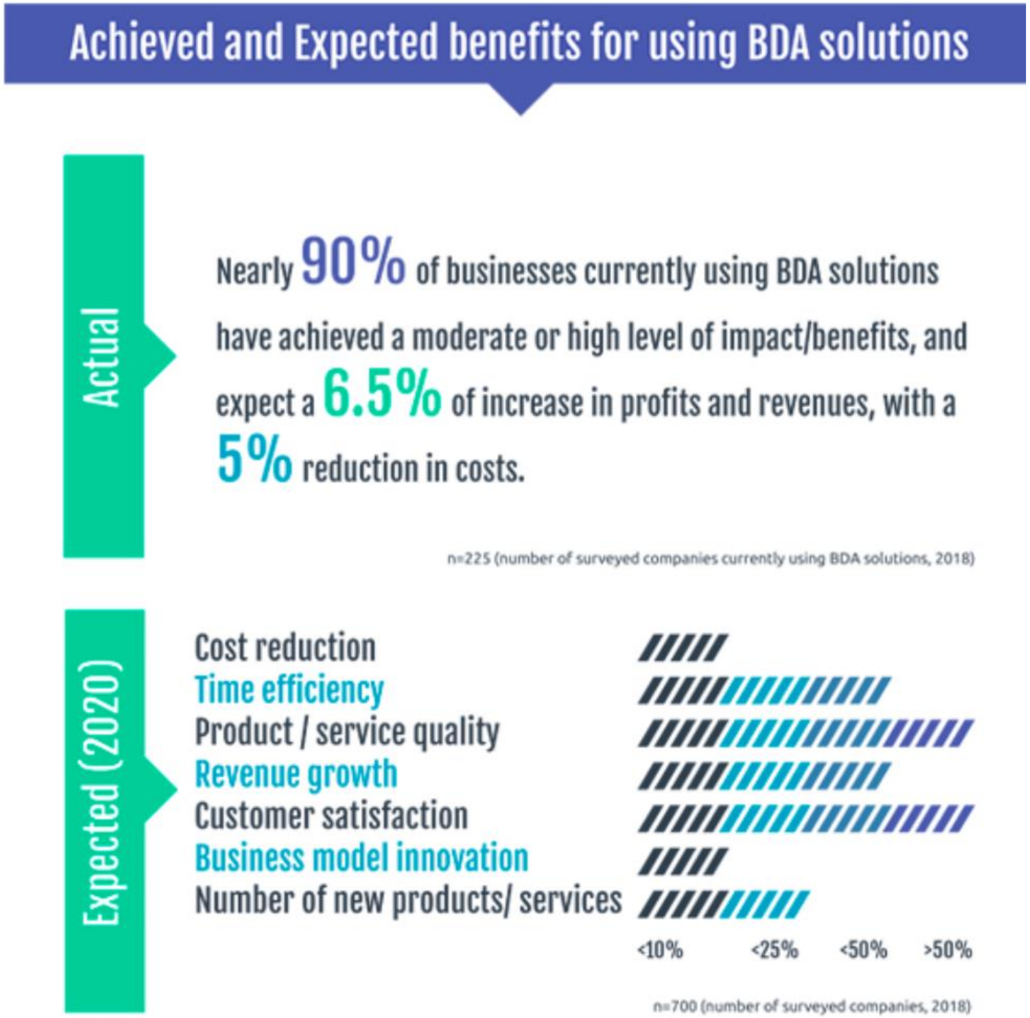
https://www.big-data-value.eu/databench-infographic-based-on-a-survey-on-700-european-companies/



Big Data Adoption

DataBench infographic based on a survey on 700 European companies

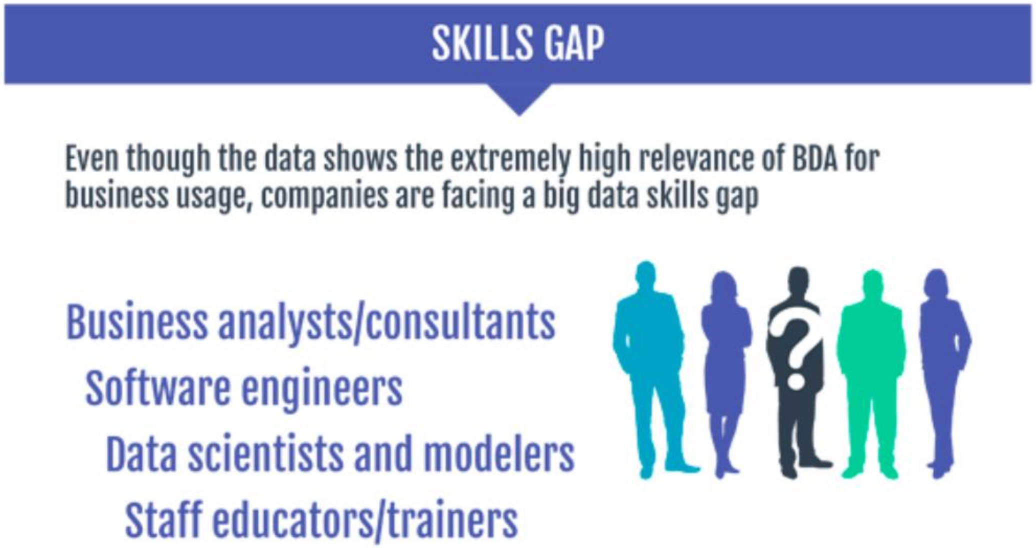
https://www.big-data-value.eu/databench-infographic-based-on-a-survey-on-700-european-companies/



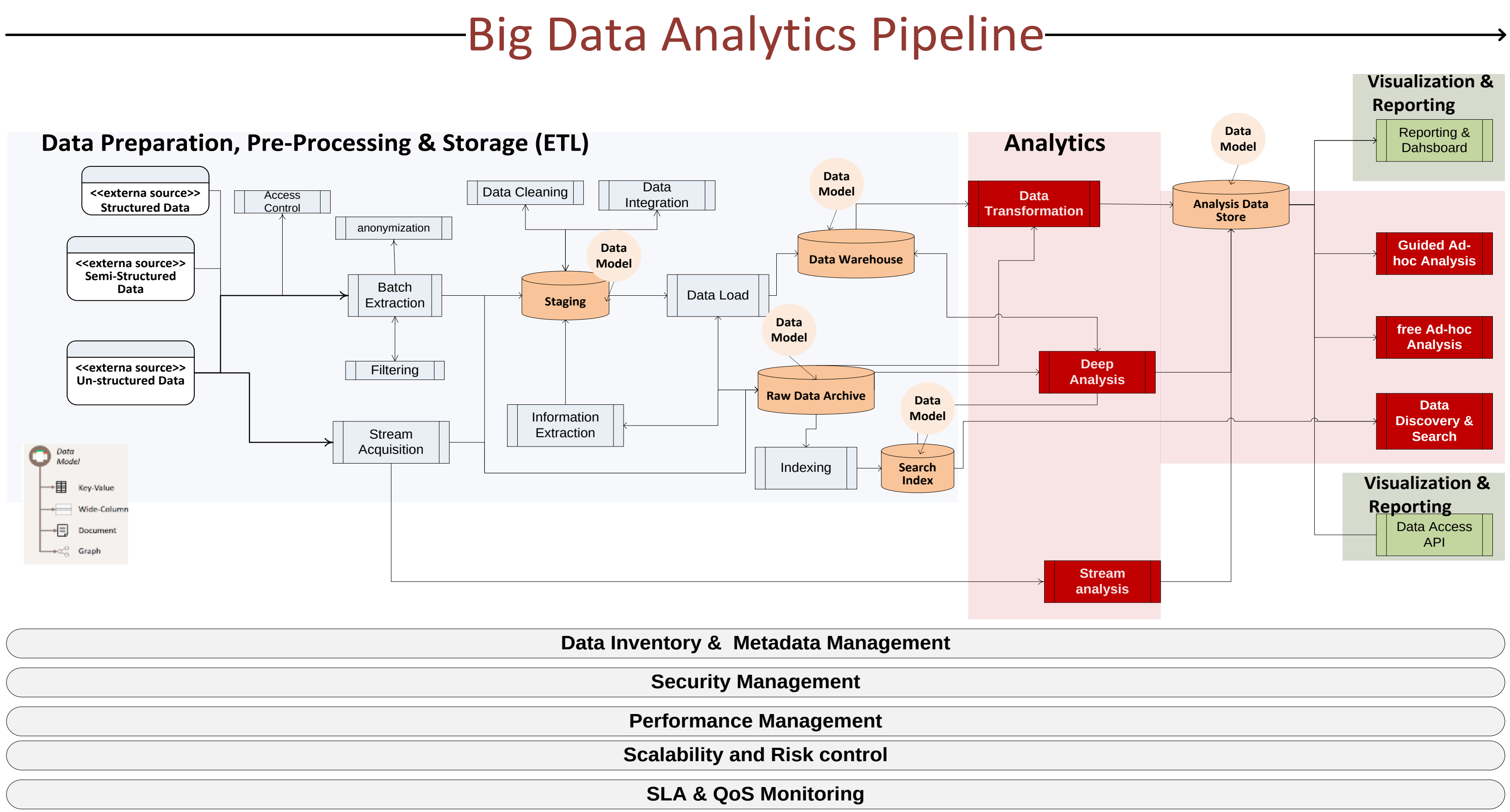
Big Data Adoption

DataBench infographic based on a survey on 700 European companies

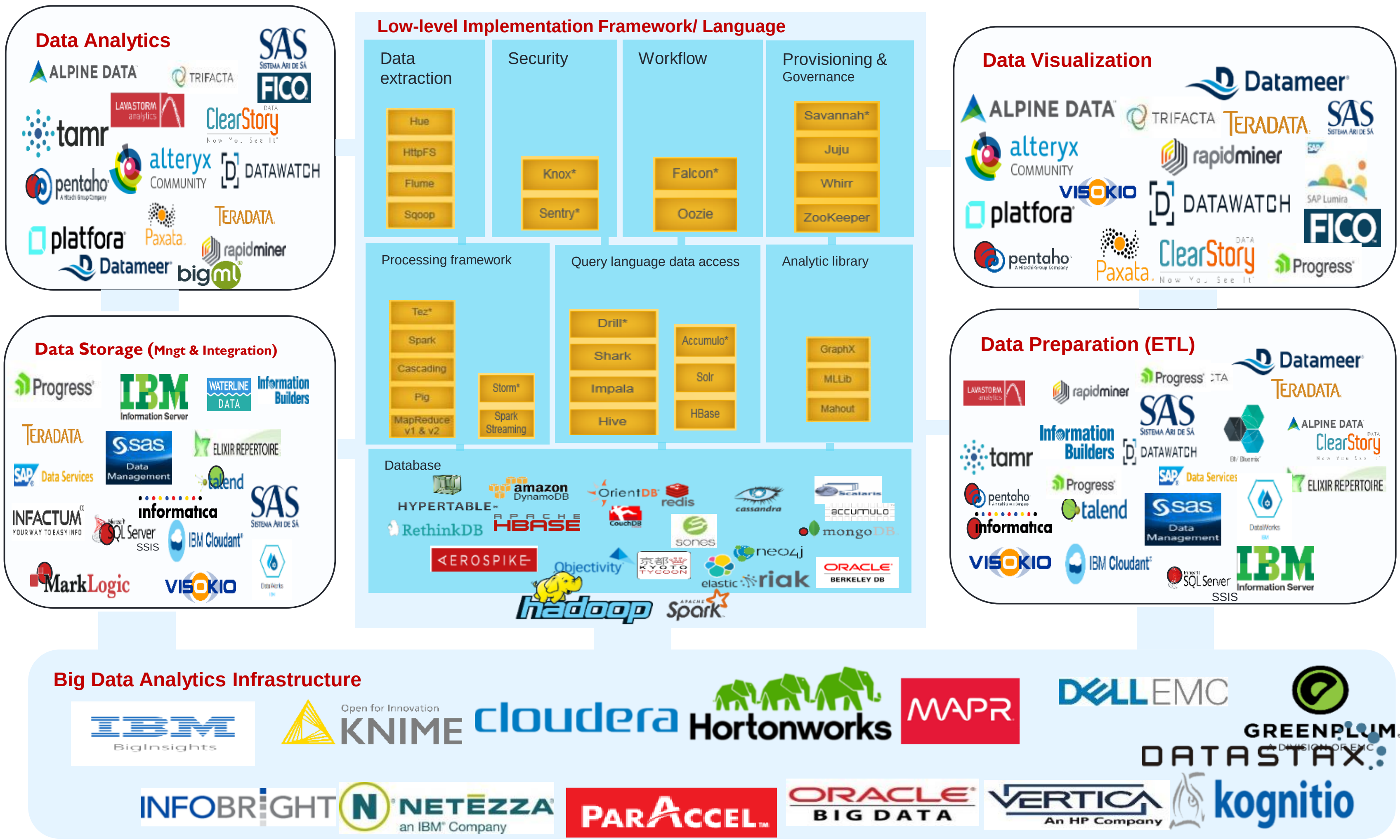
https://www.big-data-value.eu/databench-infographic-based-on-a-survey-on-700-european-companies/



High Level View of Big Data Analytics Pipeline Building Blocks



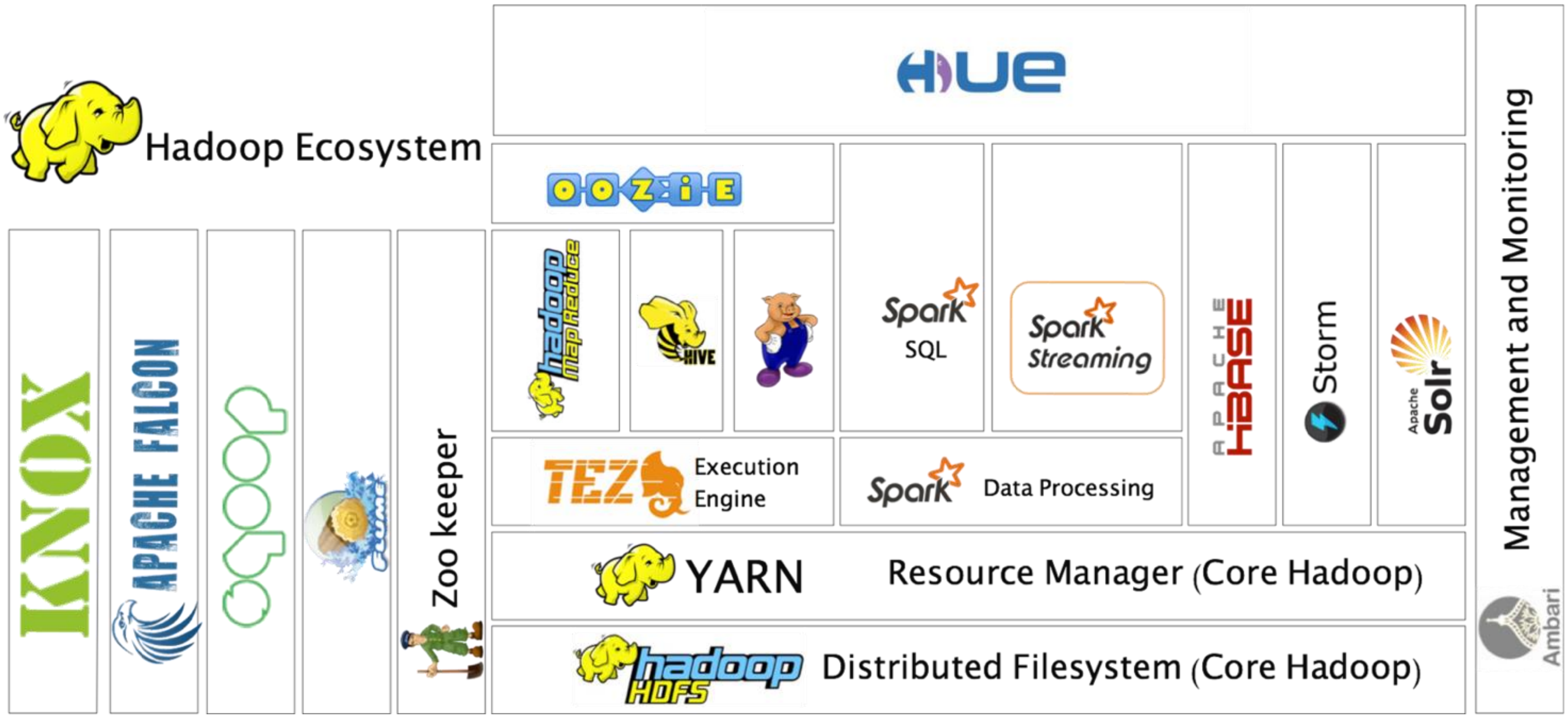
Big Data Technology Landmark



Big Data Implementation Frameworks and Languages

- Big Data technology has been made available to the community through many open source Big Data processing frameworks, storage, and implementation languages
- Companies who are interested in implementing Big Data can exploit the open access to these technologies to build their internal big data platform without depending on a particular Big Data vendor
  - Data Processing Framework: Hadoop MapReduce, Spark, Storm, Tez, Pig
  - Query Language and Data Access: Impala, Hive, Shark, Drill, Solr, Accumulo
  - Analytics and Machine Learning Algorithms: GraphX, MLLib, Mahout
  - Data Extraction and Aggregation Service: Flume, Sqoop, HttpFS
  - Security Framework: Sentry, Knox
  - Workflow Engine: Oozie, Kepler, Falcon
  - Provisioning Service: Juju, ZooKeeper, Whirr
  - Storage: Neo4J, Cassandra, redis, Hbase, RethinkDB

Apache Hadoop Stack Example (Hortonworks)



Big Data Analytics Infrastructure Platforms

- There are number of commercial analytics platforms that are built on top of Big Data open source technologies (e.g., Apache Hadoop, Spark, Storm, etc)
- Main purpose of these platforms is to provide reliable analytics platforms on top of Apache Hadoop, or Spark. Cloudera, Hortonworks, MapR, and IBM insights are the leaders in this segment
- Summary of their main features
  - Provide enterprise-ready Hadoop distributions
  - Management of Hadoop clusters
  - Performance analytics
  - Security and SLA monitoring
  - Support for integrated marketing solutions

Data Storage & Preparation Technology

- There are number of commercial tools that enable the management and governance of big data storage
- Most of these technologies provide as well a data preparation functionalities which refer to in Big Data community as ETL (Extract, Transform, and Load) tools
- Data Storage and Preparation tools offers usually the following features
  - Data inventory
  - Metadata management
  - Data quality
  - Data integration
  - Data security
  - Fault-tolerance
  - ETL
  - SLA monitoring

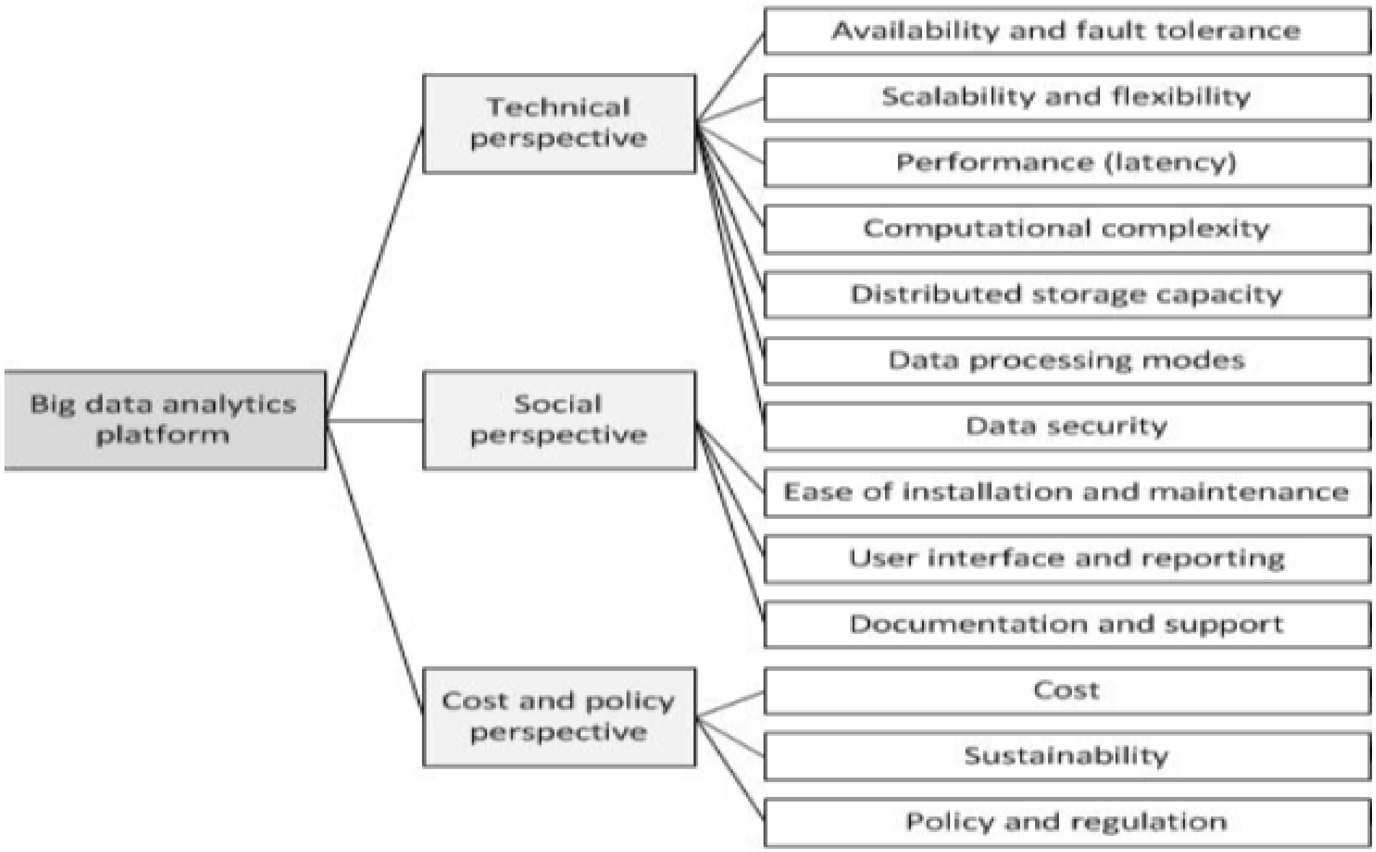
Data Analytics & Visualization Technology

- This category of technologies provides an integrated environment for applying common analytics techniques on data such as: machine learning, data mining, text mining, predictive analytics and business analytics
- At the same time, they offer different visualization and reporting techniques
- Summary of their main features
  - Graphical interface to design the analytics model
  - Store the analysis result to different databases
  - Provide data training and validation environment

Selecting Big Data Technology

- Selecting the appropriate Big Data technology that suits your business requirements requires to take in consideration the following key aspects
  - Recognize that there is no single ‘Big Data’ technology
  - Big Data has many different use cases
  - Big Data skills deficits
  - Make sure that your planning is long-term
  - Consider agile, flexible Big Data platform
  - Balance between bottom-up (tech-led) and top-down (business-led) planning

Criterion for Selecting Big Data Technology



Technical Perspective

- There are many factors that drive a decision toward the selection of a Big Data technology stack
  - Handling huge data volumes, variety, velocity
    - Flexibility, availability, fault tolerance, scalability
  - Security, Performance
  - Distributed Storage
  - Easy data integration
  - Support for integrated marketing solutions
  - Support for advanced analytics
  - Expertise available
  - Support procedural/algorithmic queries
  - Processing mode: Real-time analysis
  - Minimize administrative overhead and costs
  - Uptime and load without disruption

Technical Perspective

- Different Big Data analytics use cases will have different factors in the technology stack necessary for their support

Use cases \ Factors	Handling huge volume	Minimize data modeling	Easy data integration	Support for Integrated Marketing Solutions	Support BI tools	Support stats tools	Expertise available	Support algorithmic queries	Real-time support	Minimize administration	Uptime/load without disruption
Advanced web analytics	◆	◆	●	■	●	◆	◆	◆	■	■	■
Customer personalization	●	■	◆	◆	●	■	■	◆	■	◆	●
Advanced analytics modeling	◆	■	◆	■	■	■	■	◆	■	●	◆
Email targeting	■	■	◆	◆	●	◆	●	●	■	■	■
Site personalization	●	■	●	◆	■	◆	◆	■	◆	■	◆
Loyalty program analytics	■	■	◆	●	◆	◆	●	●	■	■	◆
Merchandising analytics	●	●	●	■	■	●	●	◆	●	■	■
Enterprise dashboarding	■	●	◆	■	◆	●	●	■	■	◆	◆
Social media analytics	◆	●	●	■	■	◆	◆	◆	■	■	■
Call avoidance and operations	●	●	◆	■	●	◆	◆	●	■	■	■

◆ High importance

● Medium importance

■ Low importance

Social Perspective

- Ease of installation and maintenance
  - Command line interface or graphical user interface, skills and knowledge needed for the deployment of a new solution
- User interface and reporting
  - Usability and complexity of features
  - Collaborative environment
- Documentation and support
  - Simple description of each tool feature, technical and customer support

Cost and Policy perspective

- Cost
  - Open source vs Commercial
- Sustainability of the solution
  - The cost associated with the skills maintenance, configuration, and adjustments to the level of agility in development
  - How much data will the organization need to manage and process today and in the future
- Policy and regulation (related to the deployment of the selected solution)
  - Privacy and data protection policies (e.g, GDPR)
  - Law conflicts
  - Restrictions of the use

Select and Manage Your Big Data Technology?

Big Data-as-a-Service

What is Big Data as a Service (BDaaS)?

- Organizations undergoing digital transformation need to understand Big Data
- The costs of setting up big data infrastructure have been prohibitive for mid-sized organizations, or those without strong technical expertise
- Offerings by cloud and SaaS/PaaS vendors are democratizing big data
  - You do not need a team of big data experts to set up infrastructure—they'll manage it for you, at affordable rates

What is Big Data as a Service (BDaaS)?

- BDaas is any service that involves managing or running big data on the cloud

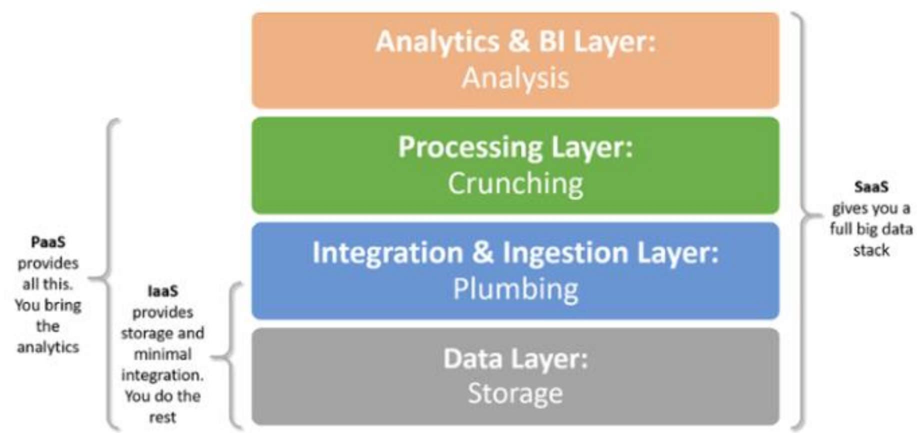
- It is not just about storage and cost
  - BDaaS solutions offer in-built solutions for artificial intelligence and analytics
  - You can accomplish some pretty impressive results without having to have a huge team of data analysts, scientists and architects around you

Why Big Data as a Service (BDaaS)?

- Advantages of BDaas
  - It makes many of the aspects that managing a big data infrastructure yourself so much easier
  - One of the biggest advantages is that it makes managing large quantities of data possible for medium-sized businesses
  - With BDaaS solutions that run in the cloud, companies do not need to stump up cash up front, and operational expenses on hardware can be kept to a minimum
  - With cloud computing, your infrastructure requirements are fixed at a monthly or annual cost

The different models of BDaaS

- Three different BDaaS models
  - Big Data Infrastructure as a Service (IaaS) – Basic data services from a cloud service provider.
  - Big Data Platform as a Service (PaaS) – Offerings of an all-round Big Data stack like those provided by Amazon S3, EMR or RedShift. This excludes ETL and BI.
  - Big Data Software as a Service (SaaS) – A complete Big Data stack within a single tool



https://hub.packtpub.com/big-data-as-a-service-bdaas-solutions-comparing-iaas-paas-and-saas/

How does the Big Data IaaS Model work?

- "Buy the engine and build the car around it, the IaaS model may be for you"
  - "The integration and workflow are on you"

- Example: Amazon AWS IaaS architecture, which combines S3 and EC2.
  - S3 acts as a data lake that can store infinite amounts of structured as well as unstructured data.
  - EC2 acts a compute layer that can be used to implement a data service of your choice and connects to the S3 data.

How does the Big Data IaaS Model work?

- Data layer
  - **Hadoop** – The Hadoop ecosystem can be run on an EC2 instance giving you complete control
  - **NoSQL Databases** – These include MongoDB or Cassandra
  - **Relational Databases** – These include PostgreSQL or MySQL
- Compute layer
  - **Self-built ETL scripts** that run on EC2 instances
  - **Commercial ETL tools** that can run on Amazon's infrastructure and use S3
  - **Open source processing tools** that run on AWS instances, like Kafka

How does the Big Data PaaS Model work?

- A standard Hadoop cloud-based Big Data Infrastructure on Amazon contains
  - Data Ingestion – Logs file data from any data source
  - Amazon S3 Data Storage Layer
    - Amazon EMR – A scalable set of instances that run Map/Reduce against the S3 data.
    - Amazon RDS – A hosted MySQL database that stores the results from Map/Reduce computations.
  - Analytics and Visualization – Using an in-house BI tool.
- A similar set up can be replicated using Microsoft’s Azure HDInsight. The data ingestion can be made easier with Azure Data Factory’s copy data tool
  - Azure offers several storage options like Data lake storage and Blob storage that you can use to store results from the computations
- Other offers: Google AppEngine-MapReduce, Heroku Treasure Data Hadoop add-on, IBM Apache Hadoop in SmartCloud

How does the Big Data PaaS Model work?

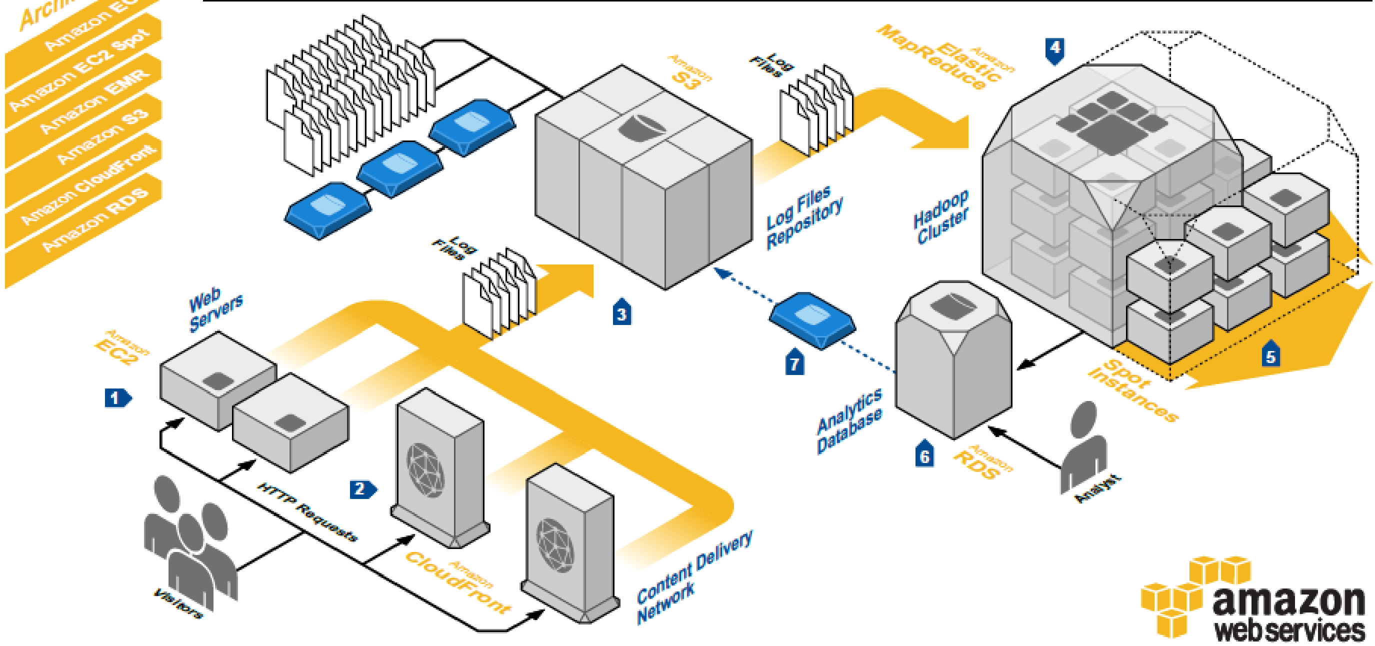
WEB LOG ANALYSIS

Amazon Web Services provides services and infrastructure to build reliable, fault-tolerant, and highly available web applications in the cloud. In production environments, these applications can generate huge amounts of log information.

This data can be an important source of knowledge for any company that is operating web applications. Analyzing logs can reveal information such as traffic patterns, user behavior, marketing profiles, etc.

However, as the web application grows and the number of visitors increases, storing and analyzing web logs becomes increasingly challenging.

This diagram shows how to use Amazon Web Services to build a scalable and reliable large-scale log analytics platform. The core component of this architecture is Amazon Elastic MapReduce, a web service that enables analysts to process large amounts of data easily and cost-effectively using a Hadoop hosted framework.



System Overview

1 The web front-end servers are running on **Amazon Elastic Compute Cloud** (Amazon EC2) instances.

2 **Amazon CloudFront** is a content delivery network that uses low latency and high data transfer speeds to distribute static files to customers. This service also generates valuable log information.

3 Log files are periodically uploaded to **Amazon Simple Storage Service** (Amazon S3), a highly available and reliable data store. Data is sent in parallel from multiple web servers or edge locations.

4 An **Amazon Elastic MapReduce** cluster processes the data set. **Amazon Elastic MapReduce** utilizes a hosted Hadoop framework, which processes the data in a parallel job flow.

5 When **Amazon EC2** has unused capacity, it offers EC2 instances at a reduced cost, called the *Spot Price*. This price fluctuates based on availability and demand. If your workload is flexible in terms of time of completion or required capacity, you can dynamically extend the capacity of your cluster using **Spot Instances** and significantly reduce the cost of running your job flows.

6 Data processing results are pushed back to a relational database using tools like *Apache Hive*. The database can be an **Amazon Relational Database Service** (Amazon RDS) instance. **Amazon RDS** makes it easy to set up, operate, and scale a relational database in the cloud.

7 Like many services, **Amazon RDS** instances are priced on a pay-as-you-go model. After analysis, the database can be backed-up into **Amazon S3** as a database snapshot, and then terminated. The database can then be recreated from the snapshot whenever needed.

How does the Big Data SaaS model work?

- A fully hosted Big Data stack complete that includes everything from data storage to data visualization contains the following:
  - **Data Layer** – Data needs to be pulled into a basic SQL database. An automated data warehouse does this efficiently
  - **Integration Layer** – Pulls the data from the SQL database into a flexible modeling layer
  - **Processing Layer** – Prepares the data based on the custom business requirements and logic provided by the user
  - **Analytics and BI Layer** – Fully featured BI abilities which include visualizations, dashboards, and charts, etc.
- Azure Synapse Analytics and AWS Redshift are the popular SaaS options that offer a complete data warehouse solution in the cloud
  - Their stack integrates all the four layers and is designed to be highly scalable
  - Google BigQuery is another contender for generating meaningful insights at an unmatched price-performance

Comparing IaaS, PaaS, SaaS

- IaaS model compared to SaaS and PaaS:
  - IaaS is “hard core”, more complex and often more expensive than other options
  - Suitable for organizations with very complex data pipelines, or those moving existing infrastructure to the cloud
  - Although IaaS is more difficult than other hosted models, it can be vastly superior to an on-premise data infrastructure
  - Lower upfront hardware costs
  - Amazon, Azure and other cloud vendors provide a scalable, performant foundation compared to your own data center
  - Most importantly, forget about maintaining the data storage layer
    - Goodbye expensive storage apps; hello Amazon S3 and Azure Blob Storage

Comparing IaaS, PaaS, SaaS

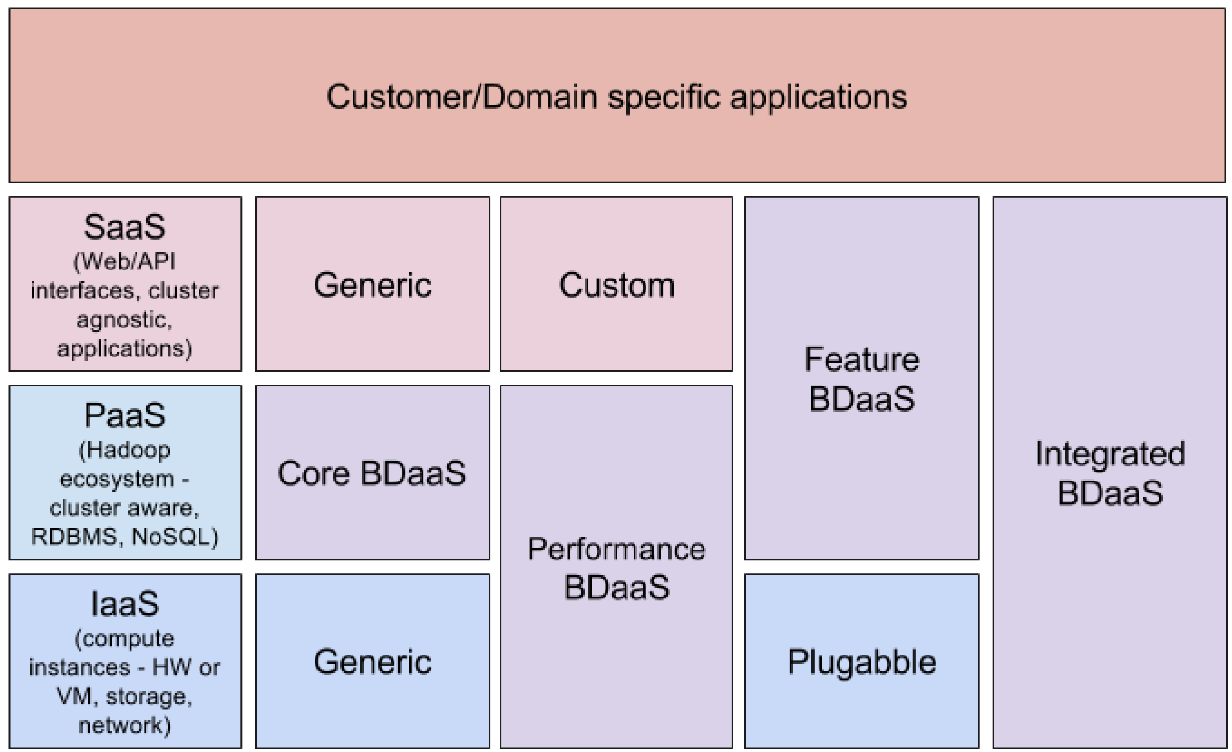
- PaaS model compared to IaaS and SaaS
  - PaaS is the middle ground—you can offload most of the work to your cloud vendor, filling in any needed gaps
  - You can still build custom data ingestion flows, and Bring Your Own BI This requires a higher level of expertise compared to SaaS options

SaaS model compared to IaaS and PaaS

  - Without complex organizational dependencies or data processes, there is little-to-no downside for smaller organizations or green field applications
  - Go from data to insights quickly, at low cost
  - Switch to a more customized implementation via PaaS or IaaS model when you need power or custom processes

Choosing the right BDaaS provider

- Core BDaaS
- Performance BDaaS
- Feature BDaaS
- Integrated BDaaS



Core BDaaS

- Core BDaaS uses a minimal platform like Hadoop with YARN and HDFS and other services like Hive
  - This service has gained popularity among companies which use this for any irregular workloads or as part of their larger infrastructure
  - They might not be as performance intensive as the other two categories.
- A prime example would be Elastic MapReduce (EMR) provided by AWS
  - This integrates freely with NoSQL store, S3 Storage, DynamoDB and similar services
  - Given its generic nature, EMR allows a company to combine it with other services which can result in simple data pipelines to a complete infrastructure

Performance BDaaS

- One path of vertical integration for BDaaS is downwards to include an optimized infrastructure
  - It allows to do away with some overheads of virtualization and specifically build hardware servers and networks that cater to Hadoop’s performance needs
- Performance BDaaS assists businesses that are already employing a cluster-computing framework like Hadoop to further optimize their infrastructure as well as the cluster performance
  - Performance BDaaS is a good fit for companies that are rapidly expanding and do not wish to be burdened by having to build a data architecture and a SaaS layer

Performance BDaaS

- The benefit of outsourcing the infrastructure and platform is that companies can focus on specific processes that add value instead of concentrating on complicated Big Data related infrastructure
  - For instance, there are many third-party solutions built on top of Amazon or Azure stack that let you outsource your infrastructure and platform requirements to them.

Feature BDaaS

- The other path of integration for BDaaS is upwards to include features beyond the common Hadoop ecosystem offerings.
- If your business is in need of additional features that may not be within the scope of Hadoop, Feature BDaaS may be the way forward
  - It focuses on productivity as well as abstraction
  - It is designed to enable users to be up and using Big Data quickly and efficiently

Feature BDaaS

- Feature BDaaS combines both PaaS and SaaS layers
  - This includes web/API interfaces, and database adapters that offer a layer of abstraction from the underlying details
  - Businesses do not have to spend resources and manpower setting up the cloud infrastructure
  - Instead, they can rely on third-party vendors like Qubole and Altiscale that are designed to set it up and running on AWS, Azure or cloud vendor of choice quickly and efficiently

Integrated BDaaS

- Lastly, another option is a fully vertically integrated BDaaS that combines the performance and feature benefits of the previous two BDaaS
- This is an appealing approach since it could result in the perfect BDaaS, which is productive and supports business users and experts, and provides maximum performance.
- Both feature and performance BDaaS are at early stages and the integrated BDaaS could in practice turn out to be a squaring the circle problem

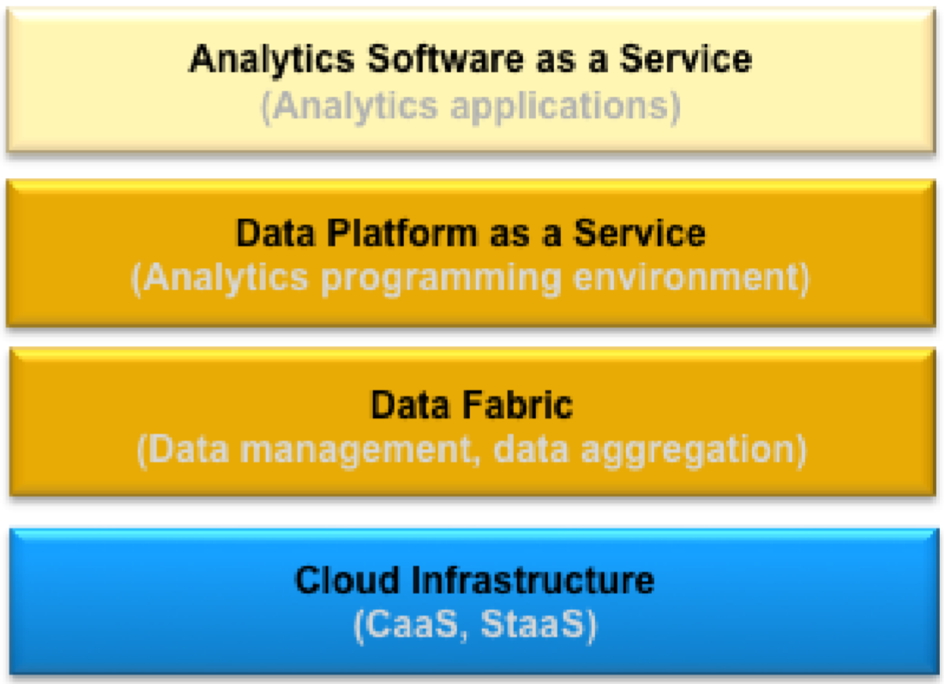
Additional Aspects

- **Low or Zero Startup Costs** – A number of BDaaS providers offer a free trial period. Therefore, theoretically, you can start seeing results before you even commit a dollar
- **Scalable** – Growth in scale is in the very nature of a Big Data project. The solution should be easy and affordable to scale, especially in terms of storage and processing resources.
- **Industry Footprint** – It is a good idea to choose a BDaaS provider that already has an experience in your industry. This is doubly important if you are also using them for consultancy and project planning requirement

Additional Aspects

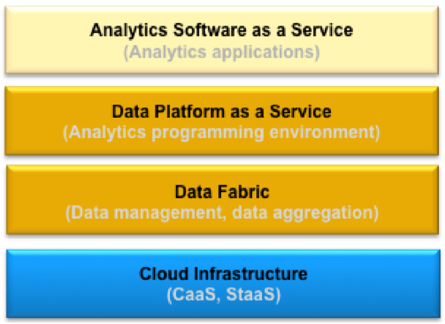
- **Real-Time Analysis and Feedback** – The most successful Big Data projects today are those that can provide almost immediate analysis and feedback. This helps businesses to take remedial action instantly instead of working off of historical data
- **Managed or Self-Service** – Most BDaaS providers today provide a mix of both managed as well as self-service models based on the company's needs. It is common to find a host of technical staff working in the background to provide the client with services as needed

Big Data as-a-Service: Another View



Big Data-as-a-Service: Cloud Infrastructure

- Big Data-as-a-Service usually leverages cloud components
  - Compute and storage nodes
- Data produced by applications deployed on a cloud infrastructure
- Moving data is costly
  - Having data already available in the service provider’s infrastructure increase performance and service offering

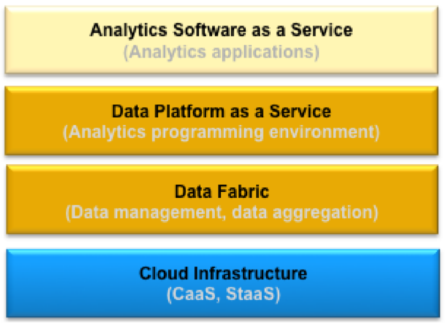


Big Data-as-a-Service: Data Fabric

- Service providers can offer data fabric services

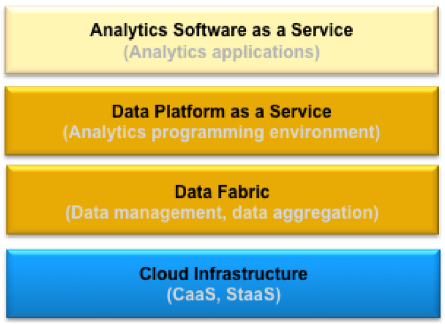
- Data management
  - Platform-as-a-Service
  - Database-as-a-Service

- Data aggregation and exposure
  - Data-as-a-Service: aggregating and managing datasets, controlled access to data
    - E.g. Google’s Public Data service



Big Data-as-a-Service: Data Platform-as-a-Service

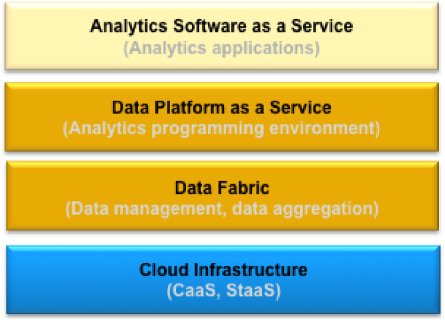
- Service provider not only puts a data management infrastructure in line, but also the execution environment for data processing applications and scripts
- Users can upload both their data and analytics jobs
  - Platform take care of scale out/scale down appropriate clusters of data and processing nodes
  - Users=data scientists/programmers able to manage private dedicated analytics environments, as well as to write analytics jobs



Big Data-as-a-Service: Analytics SaaS

- Users is familiar with an analytics platform at a higher abstraction layer
  - Execute scripts and queries that data scientists or programmers developed for them
  - Generate reports, visualizations, dashboards

- Analytics SaaS has many vertical-specific solutions



Conclusions

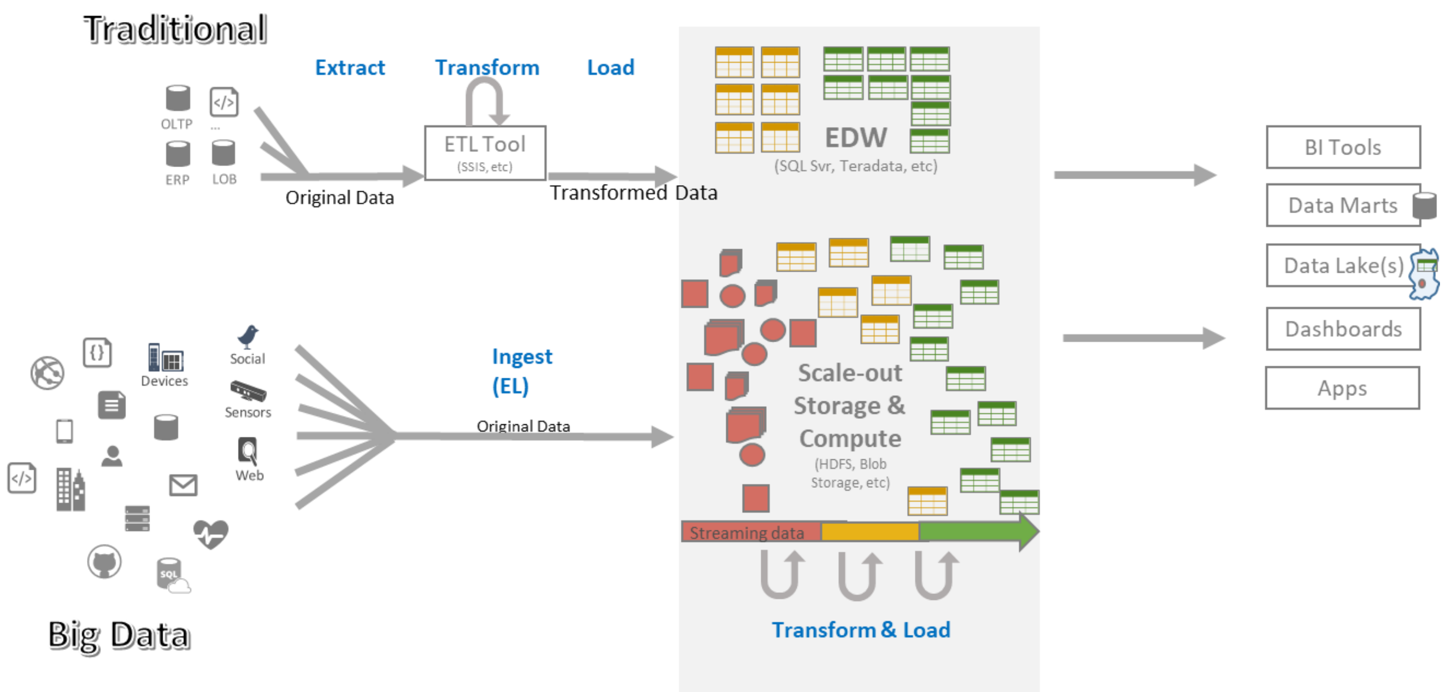
- Complexity of Big Data pipelines
- Big Data adoption
- Big Data Platform-as-a-Service
  - Core BDaaS, Performance BDaaS, Feature BDaaS, Integrated BDaaS
  - Cloud Infrastructure, Data Fabric, Data Platform-as-a-Service, Analytics SaaS

Lesson 3.2: AZURE HD Insight (READ ONLY)

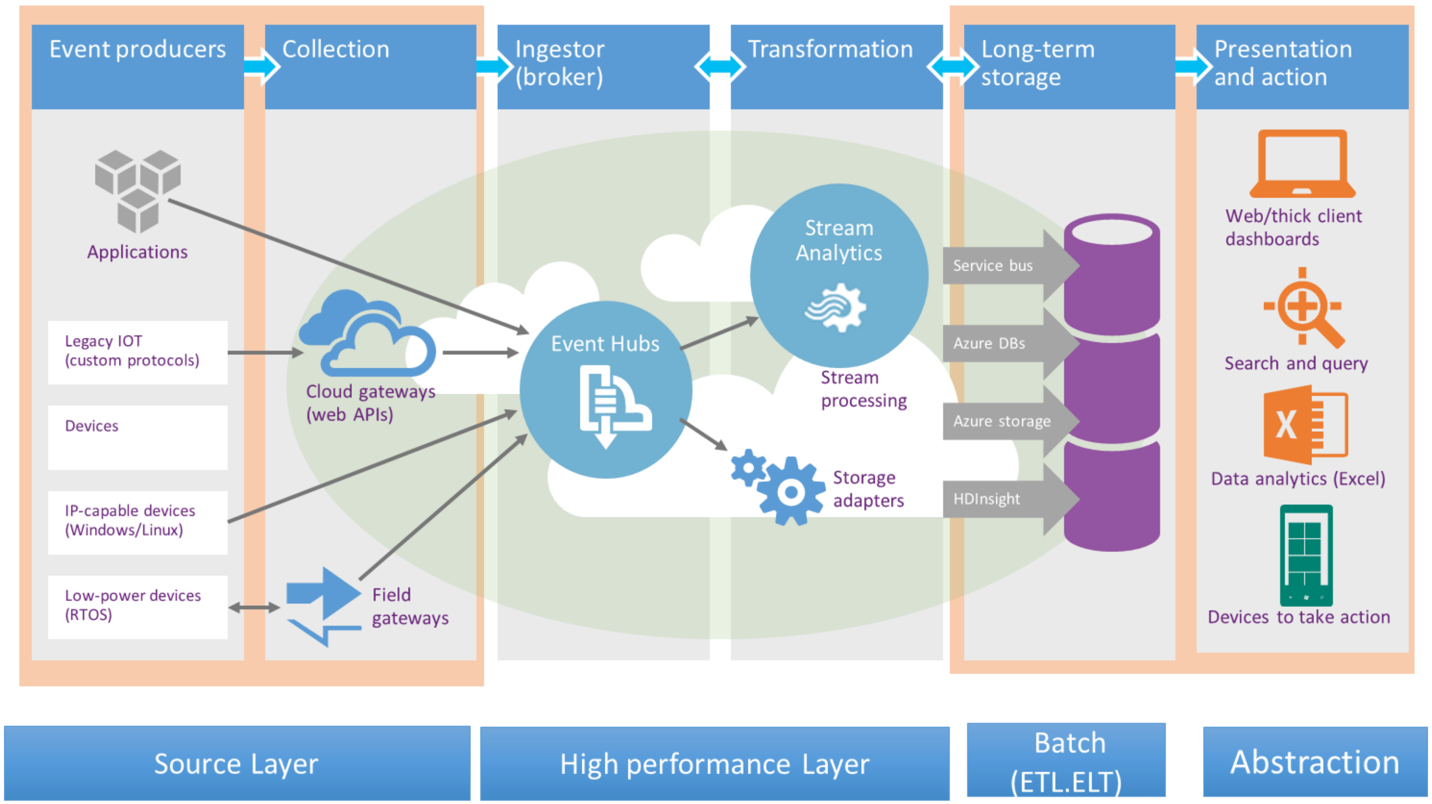
Claudio Ardagna – Università degli Studi di Milano

Cloud and Distributed Computing

Evolving Approaches to Analytics



Canonical Architecture



Introducing Hadoop

- Apache Hadoop is for big data - Open Source for reliable, scalable, distributed computing.
- It is a set of open source projects that transform commodity hardware into a service that can:
  - Store petabytes of data reliably
  - Allow huge distributed computations
- Key attributes:
  - Hadoop common – utilities to support modules
  - HDFS (Hadoop Distributed File System) – high throughput
  - YARN – job scheduling and cluster RM
  - MapReduce – YARN-based for parallel processing
  - Spark – compute engine
  - Pig – data-flow language & execution framework
  - Oozie – workflow scheduler
  - Ambari – provisioning, managing and monitoring clusters
  - Sqoop – bulk data transfer between Hadoop & Relational DB
  - Batch processing centric – using a “Map-Reduce” processing paradigm

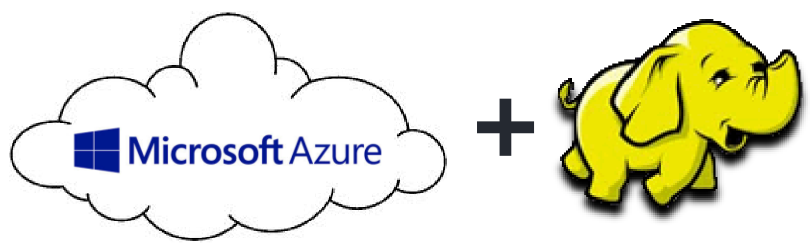
Introducing Hadoop

- Comparison to Traditional RDBMS

	TRADITIONAL RDBMS	HADOOP
Data Size	Gigabytes (Terabytes)	Petabytes (even Exabytes)
Access	Interactive and Batch	Batch
Updates	Read / Write many times	Write once, Read many times
Structure	Static Schema	Dynamic Schema
Integrity	High (ACID)	Low
Scaling	Nonlinear	Linear
DBA Ratio	1:40	1:3000

Introducing Azure HDInsight

- Azure HDInsight is Microsoft's Hadoop-based service that enables big data solutions in the cloud
- A cloud implementation on Microsoft Azure of the rapidly expanding Apache Hadoop technology stack
- Hortonworks Data Platform that is the go-to solution for big data analysis



About Microsoft Azure HDInsight

Cloud + Hadoop

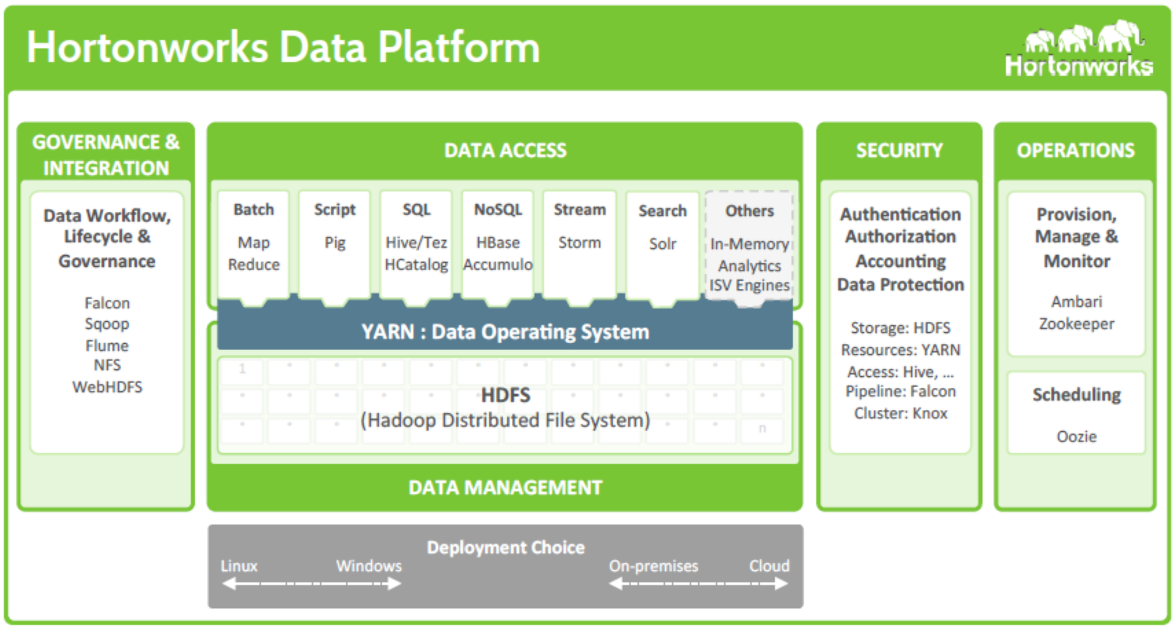
- Key attributes:
  - integrates with Microsoft BI & scripting tools :
  - Power BI,
  - Excel
  - SSAS and SSRS
  - PowerShell
  - implementations of Apache Spark, HBase, Storm, Pig, Hive, Sqoop, Oozie, Ambari, and so on.

About Microsoft Azure HDInsight

Cloud + Hadoop

- Microsoft's managed Hadoop as a Service
- 100% open source Apache Hadoop
- Built on the latest releases across Hadoop (2.4)
  - YARN
  - Stinger Phase 2 (Faster queries)
- Up and running in minutes with no hardware to deploy
- Access Data with Pig and Hive
- Utilize familiar BI tools for analysis including Microsoft Excel

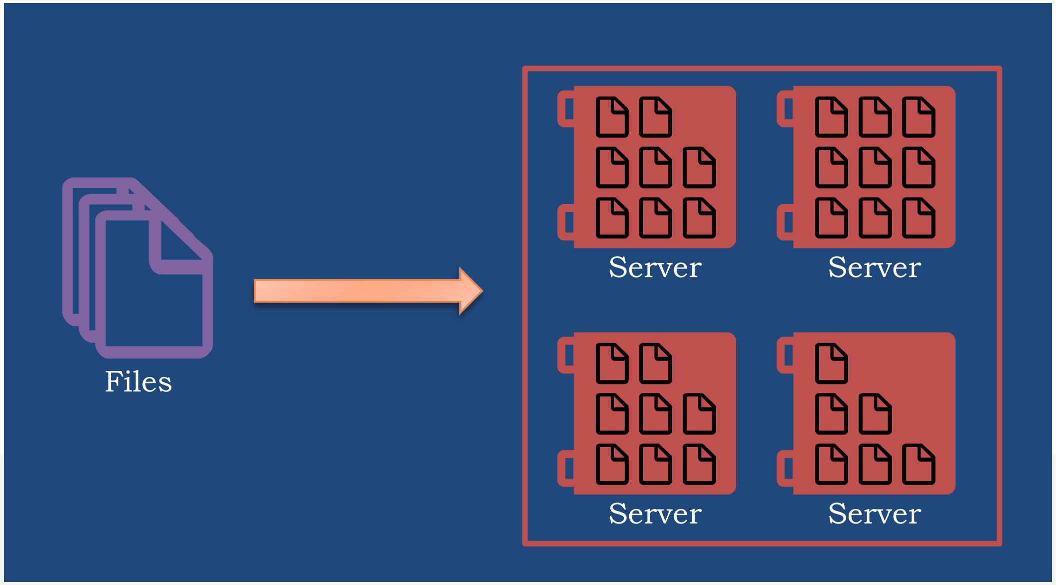
Hortonworks Data Platform On Azure



HDP is the only completely open Hadoop data platform available. All solutions in HDP are developed as projects through the ***Apache Software Foundation (ASF). There are NO proprietary extensions in HDP.***

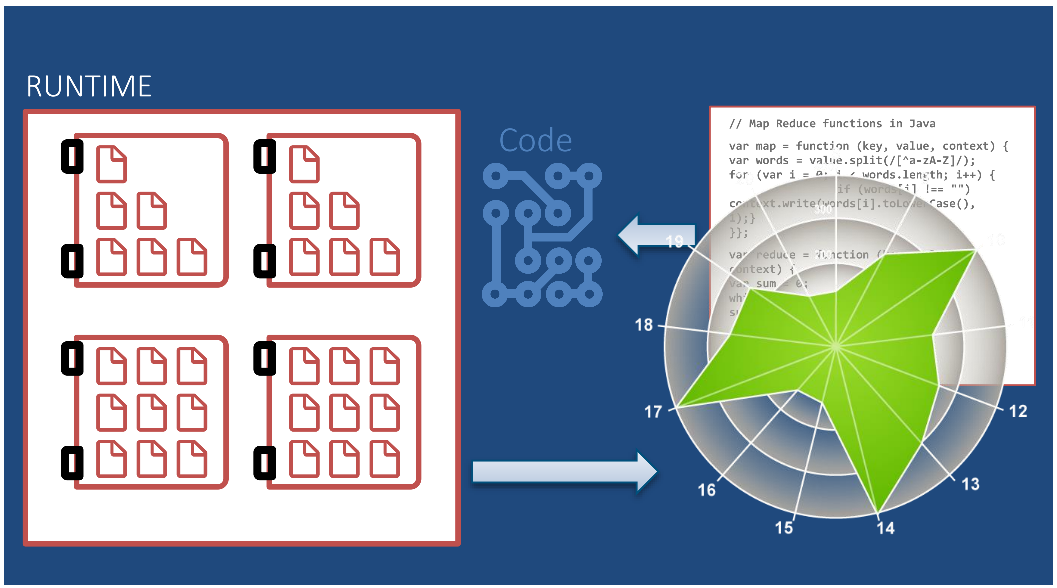
Hadoop

- How it Works: 1 – Data Storage

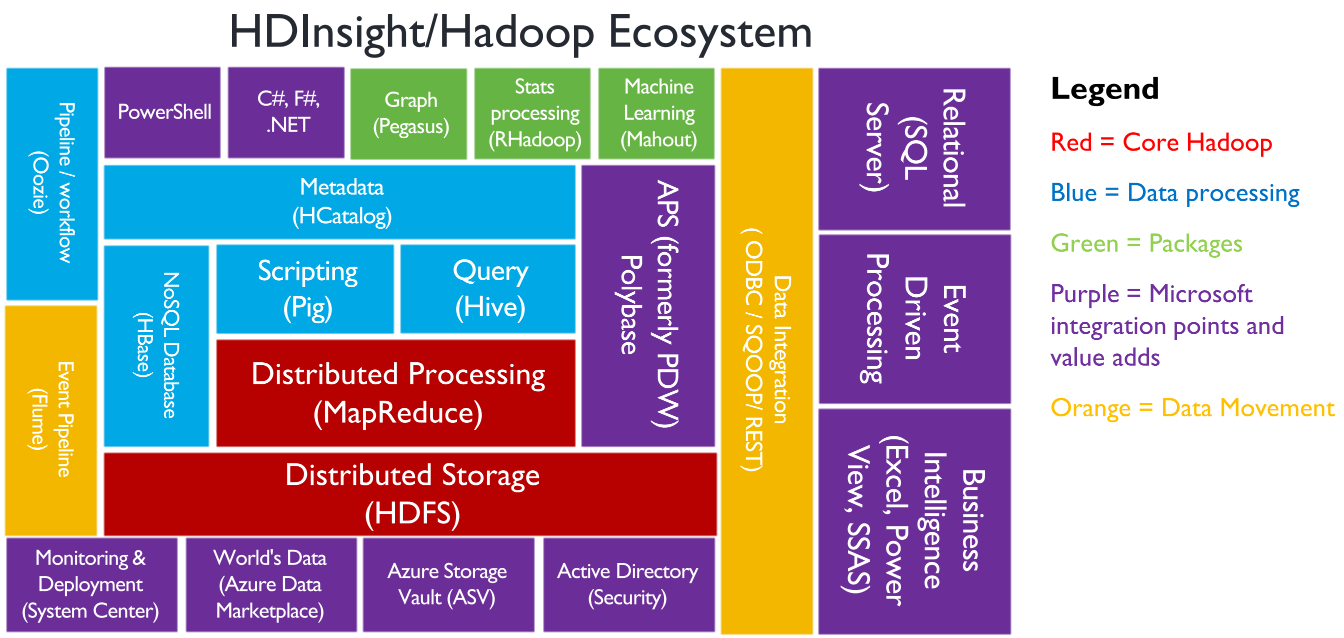


Hadoop

- How it Works: 2 – Take the Processing to the Data



Introducing the zoo:

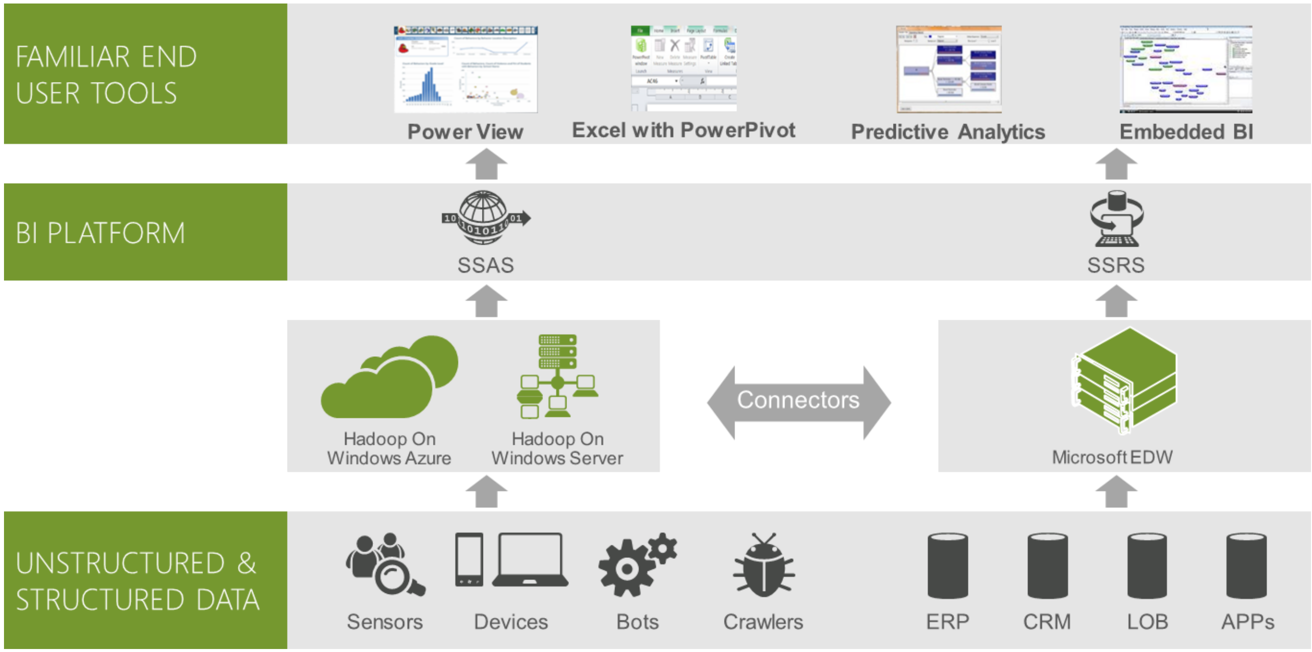


Programming HDInsight

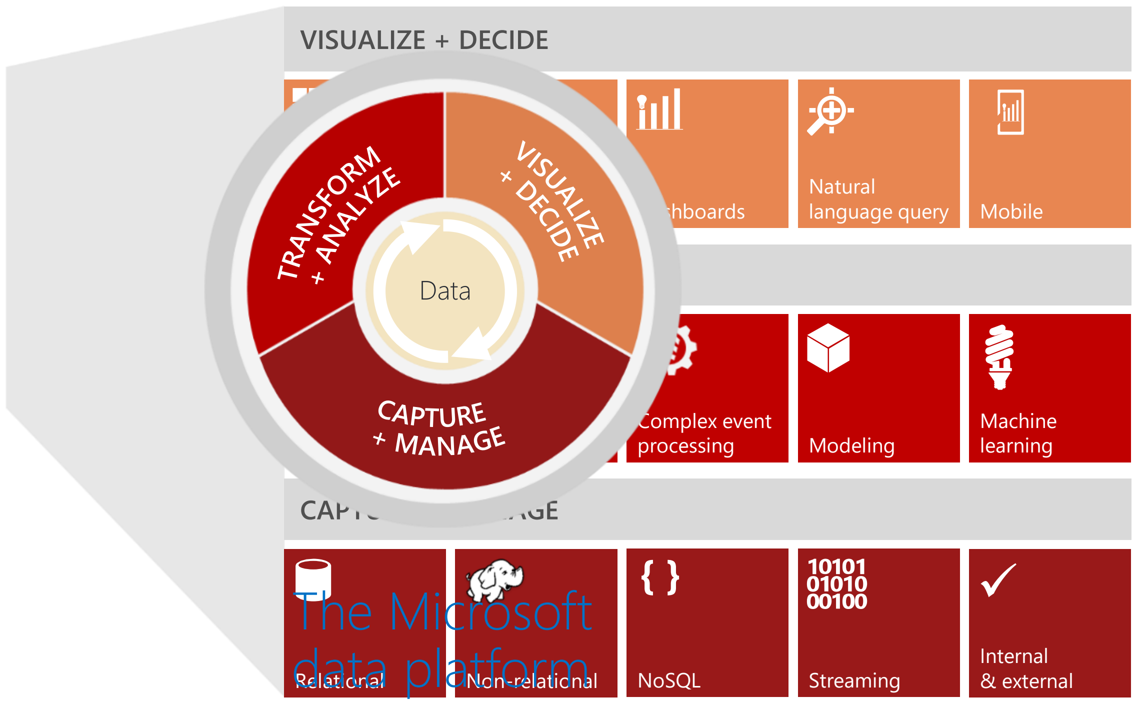
- Since HDInsight is a service-based implementation, you get immediate access to the tools you need to program against HDInsight/Hadoop

Existing Ecosystem	• Hive, Pig, Sqoop, Mahout, Cascading, Scalding, Scoobi, Pegasus, etc.
.NET	• C#, F# Map/Reduce, LINQ to Hive, .Net Management Clients, etc.
JavaScript	• JavaScript Map/Reduce, Browser-hosted Console, Node.js management clients
DevOps/IT Pros:	• PowerShell, Cross-Platform CLI Tools

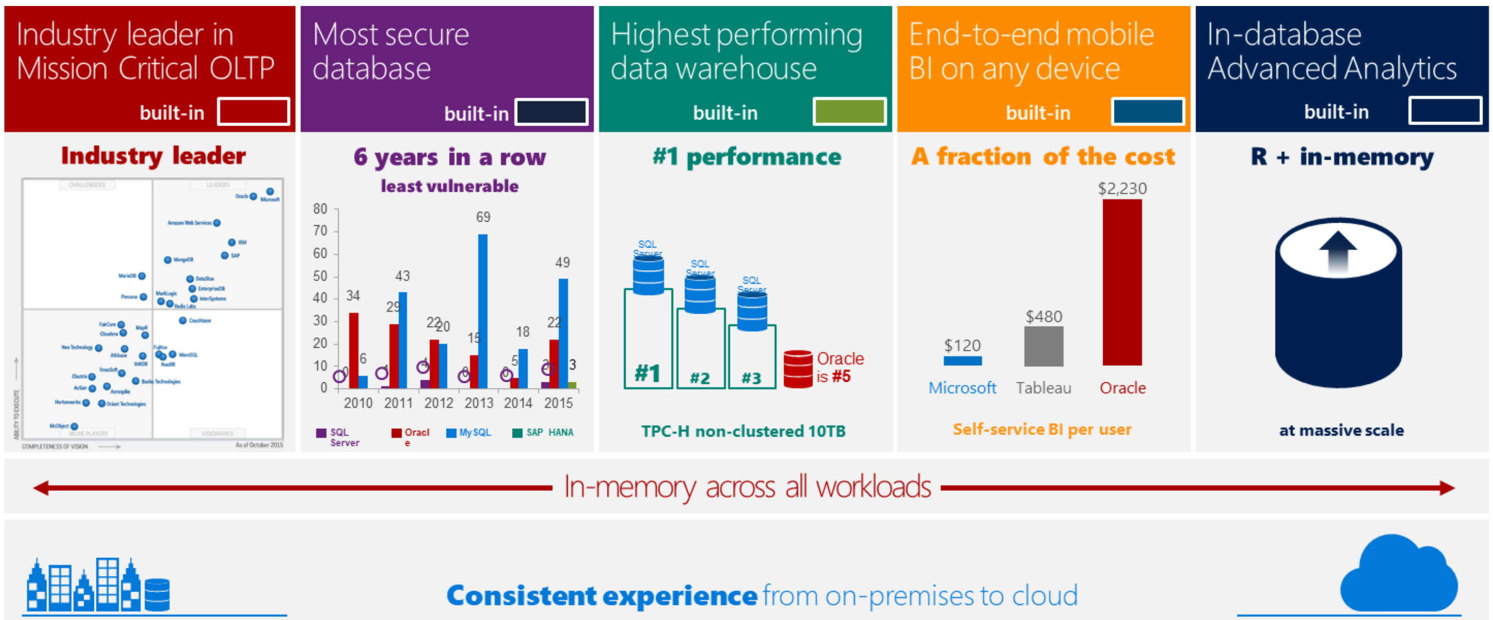
Microsoft Big Data Solution



The Microsoft data platform



SQL Server 2016: Everything built-in



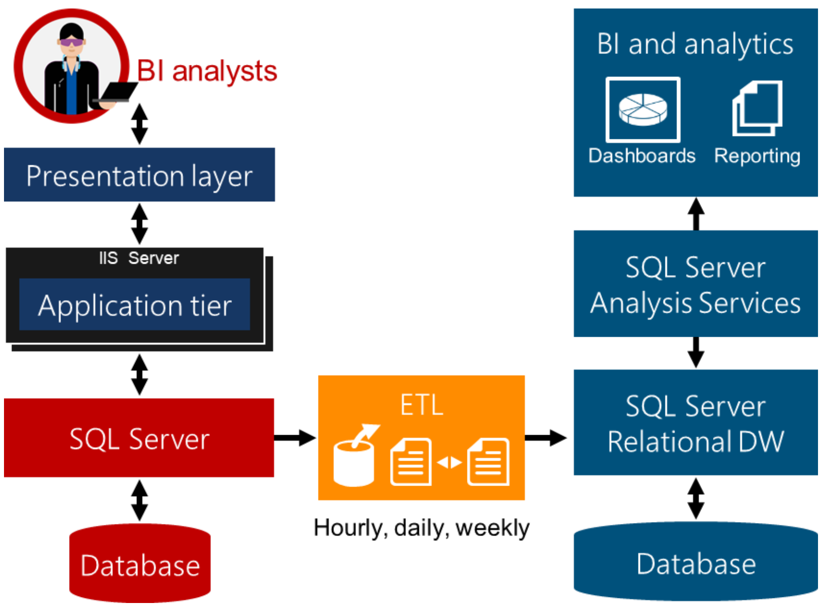
The above graphics were published by Gartner, Inc. as part of a larger research document and should be evaluated in the context of the entire document. The Gartner document is available upon request from Microsoft. Gartner does not endorse any vendor, product or service depicted in its research publications, and does not advise technology users to select only those vendors with the highest ratings or other designation. Gartner research publications consist of the opinions of Gartner's research organization and should not be construed as statements of fact. Gartner disclaims all warranties, expressed or implied, with respect to this research, including any warranties of merchantability or fitness for a particular purpose.

National Institute of Standards and Technology Comprehensive Vulnerability Database update 10/2015

TPC-H non-clustered results as of 04/06/15, 5/04/15, 4/15/14 and 11/25/13, respectively. http://www.tpc.org/tpch/results/tpch_perf_results.asp?resulttype=noncluster

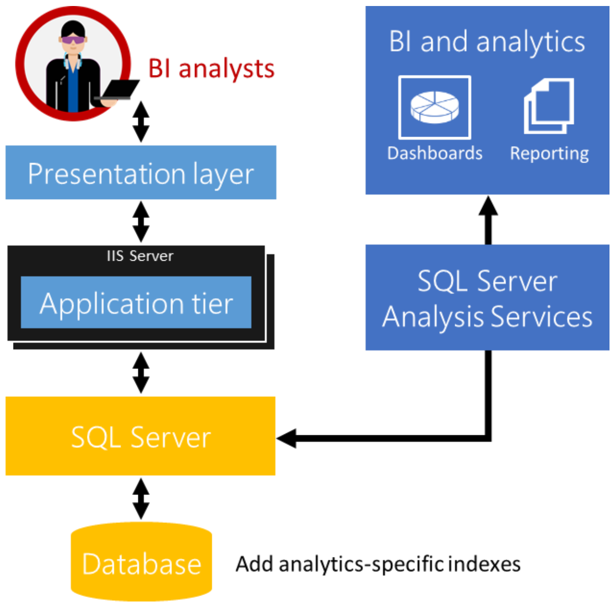
Traditional operational/analytics architecture

- Key issues
  - Complex implementation
  - Requires two servers (capital expenditures and operational expenditures)
  - Data latency in analytics
  - High demand;
    requires real-time analytics

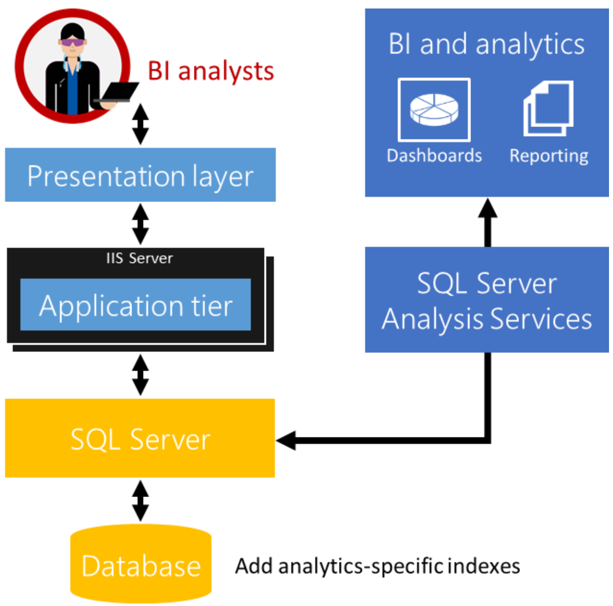


Minimizing data latency for analytics

- Challenges
  - Analytics queries are resource intensive and can cause blocking
  - Minimizing impact on operational workloads
  - Sub-optimal execution of analytics on relational schema
- Benefits
  - No data latency
  - No ETL
  - No separate data warehouse



Minimizing data latency for analytics



Key points

Create an updateable NCCI for analytics queries

Drop all other indexes that were created for analytics

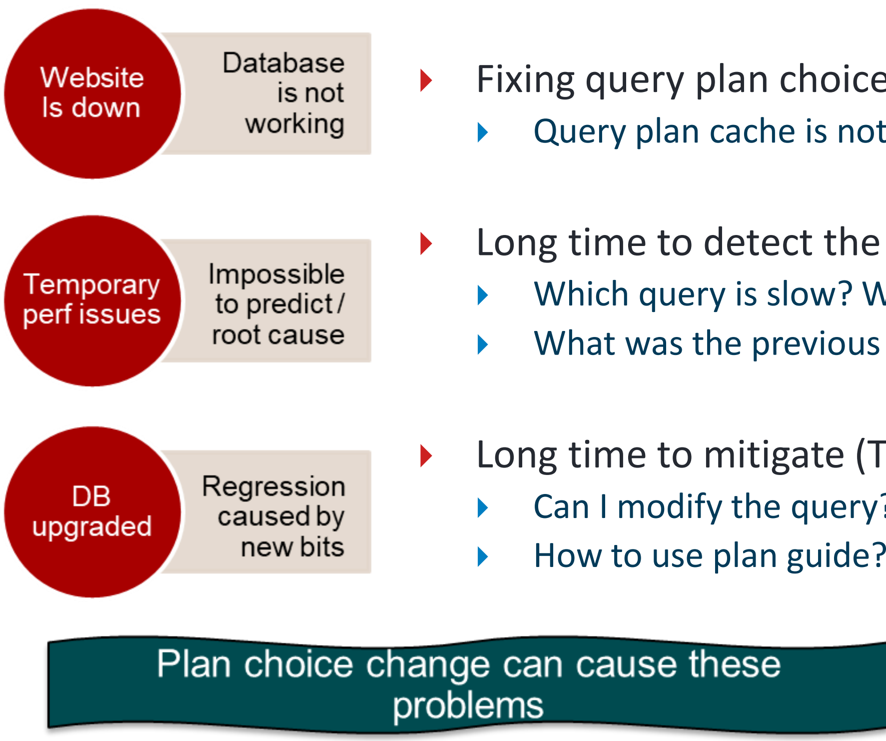
No application changes

Columnstore index is maintained just like any other index

Query optimizer will choose columnstore index where needed

Achieved using columnstore indexes

Problems with query performance



- Fixing query plan choice regressions is difficult
  - Query plan cache is not well-suited for performance troubleshooting
- Long time to detect the issue (TTD)
  - Which query is slow? Why is it slow?
  - What was the previous plan?
- Long time to mitigate (TTM)
  - Can I modify the query?
  - How to use plan guide?

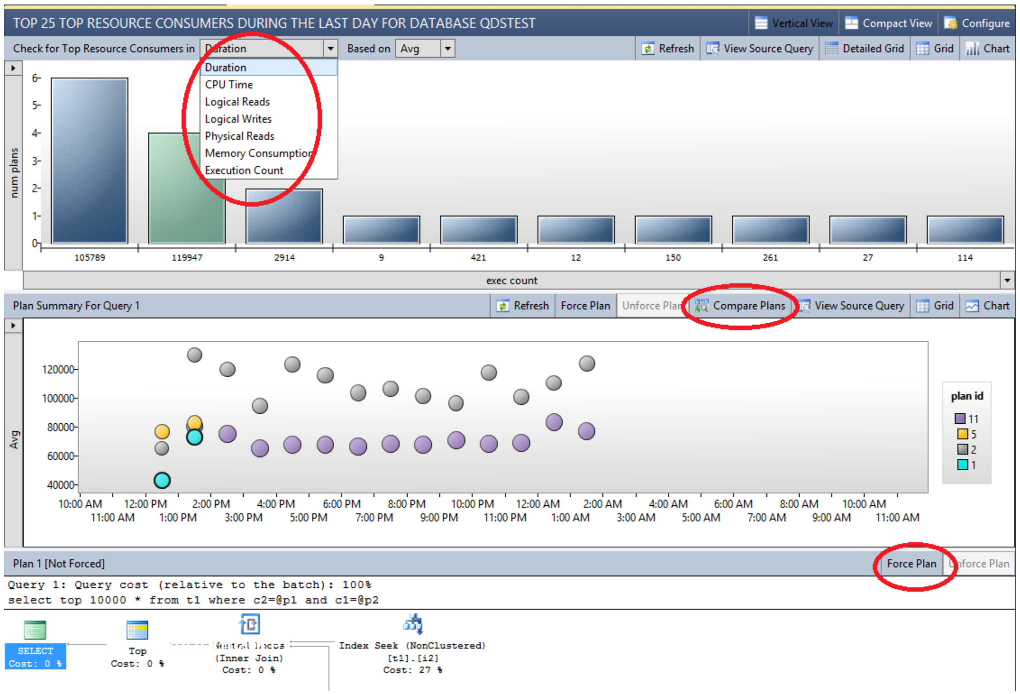
The solution: Query Store

- Dedicated store for query workload performance data
  - Captures the history of plans for each query
  - Captures the performance of each plan over time
  - Persists the data to disk (works across restarts, upgrades, and recompiles)

- Significantly reduces TTD/TTM
  - Find regressions and other issues in seconds
  - Allows you to force previous plans from history

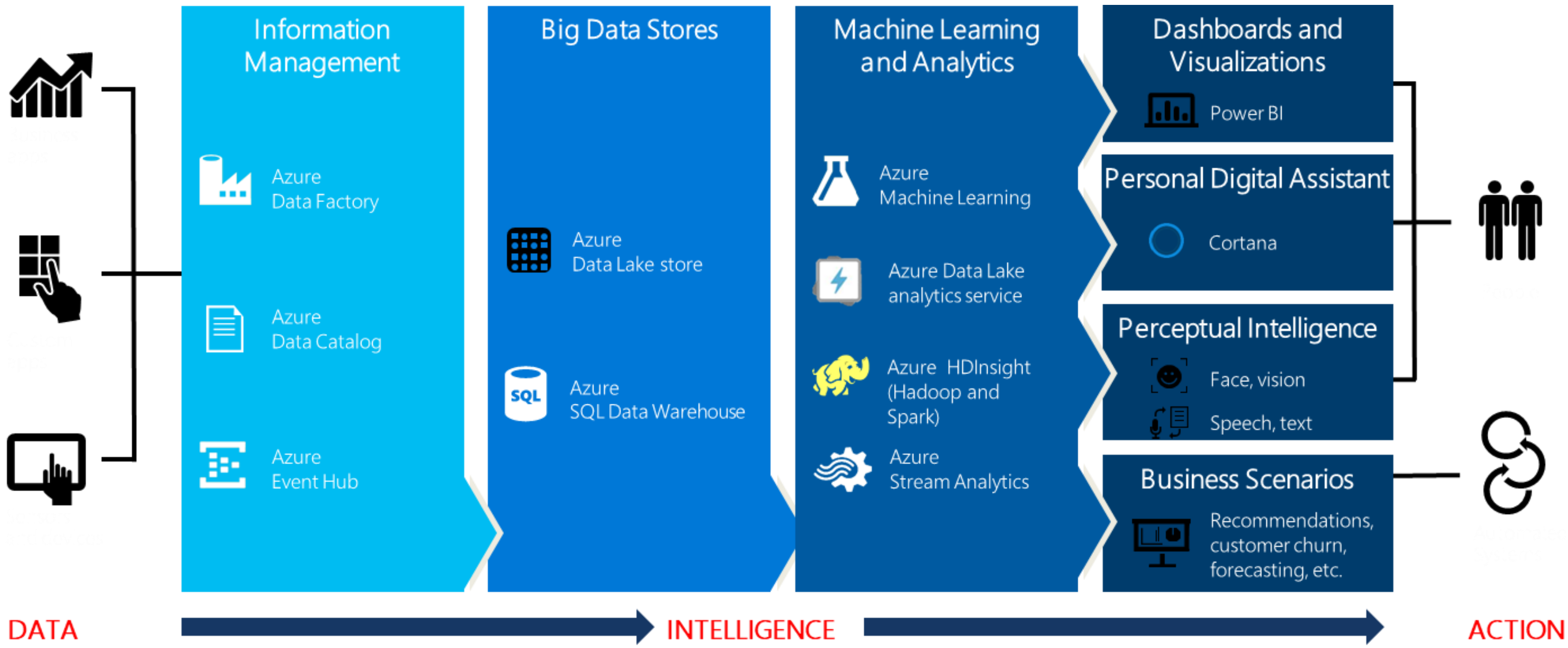
- DBA is now in control

Monitoring performance by using the Query Store



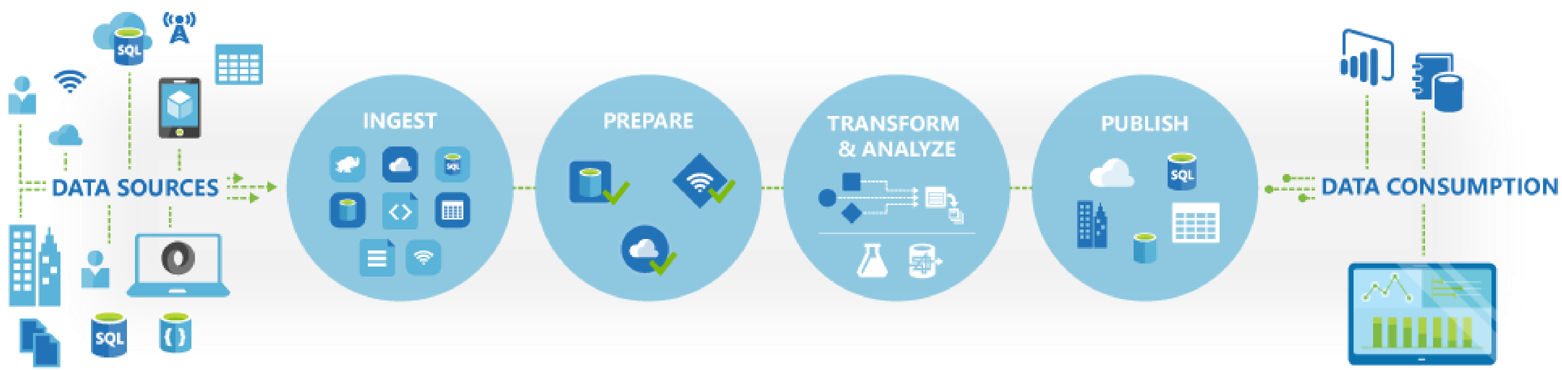
The Query Store feature provides DBAs with insight on query plan choice and performance

Cortana Analytics Suite
Transform data into intelligent action



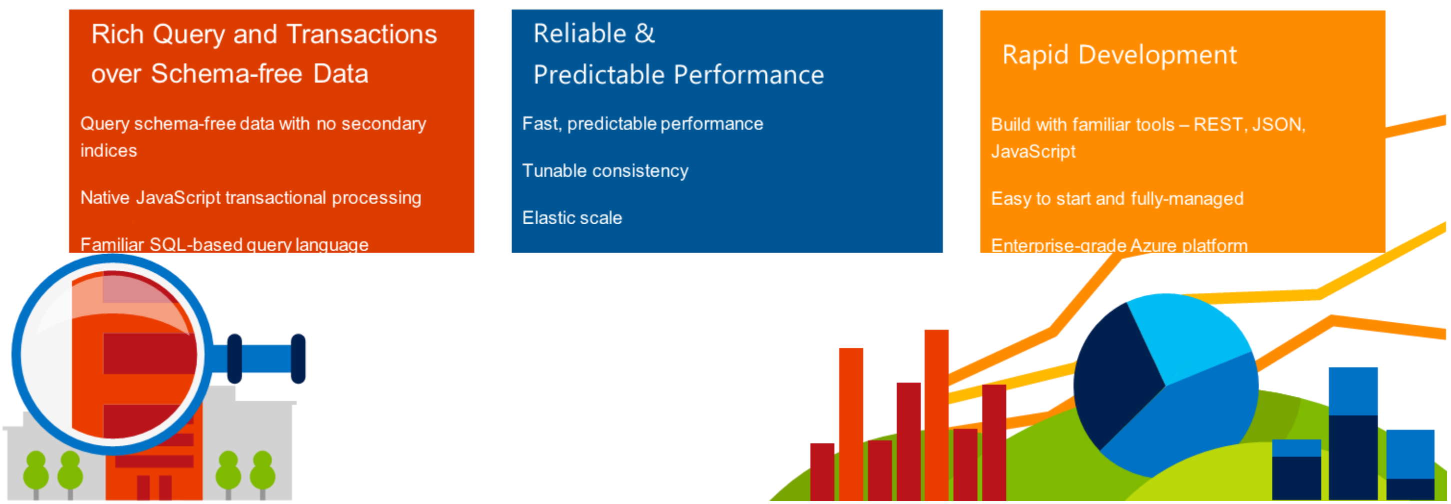
Azure Data Factory

- A managed cloud service for building & operating data pipelines
- Part of the Cortana Analytics Suite

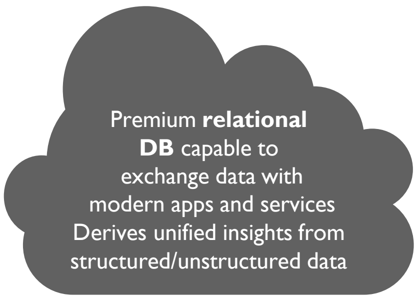


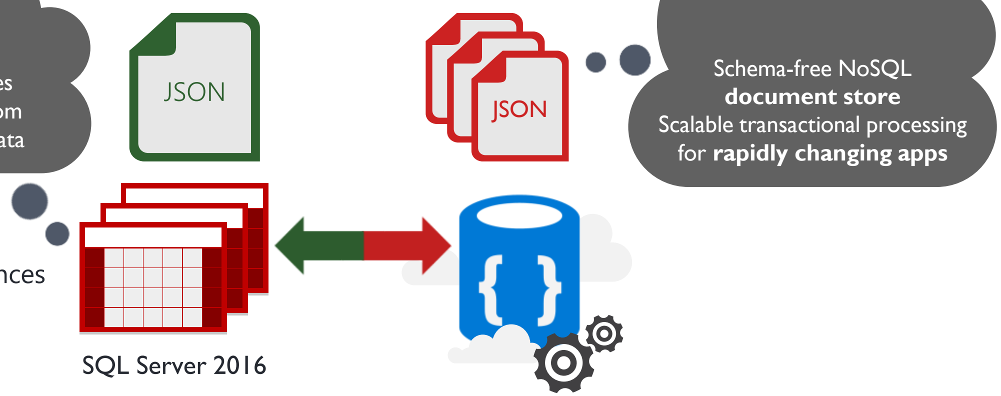
DocumentDB – Schema Free DB

- A NoSQL document database-as-a-service, fully managed by Microsoft Azure.
- For cloud-designed apps when query over schema-free data; reliable and predictable performance; and rapid development are key. First of its kind database service to offer native support for JavaScript, SQL query and transactions over schema-free JSON documents.
- Perfect for cloud architects and developers who need an enterprise-ready NoSQL document database.



SQL Server and Azure DocumentDB





Capability Greatly enhances developer productivity

Benefits Added native JSON support in the core database engine supports schema-free data. Tackle more diverse data types right in SQL Server, support in DocumentDB

TechNet Virtual Lab
“Exploring SQL Server 2016 support for JSON data” - http://go.microsoft.com/?linkid=9898458

What's next?

- Moving from on premises platforms to Big Data-as-a-Service
- Different solutions at different layers of a cloud environment
- Next step: Big Data Analytics-as-a-Service

Lesson 4.1: Microservices

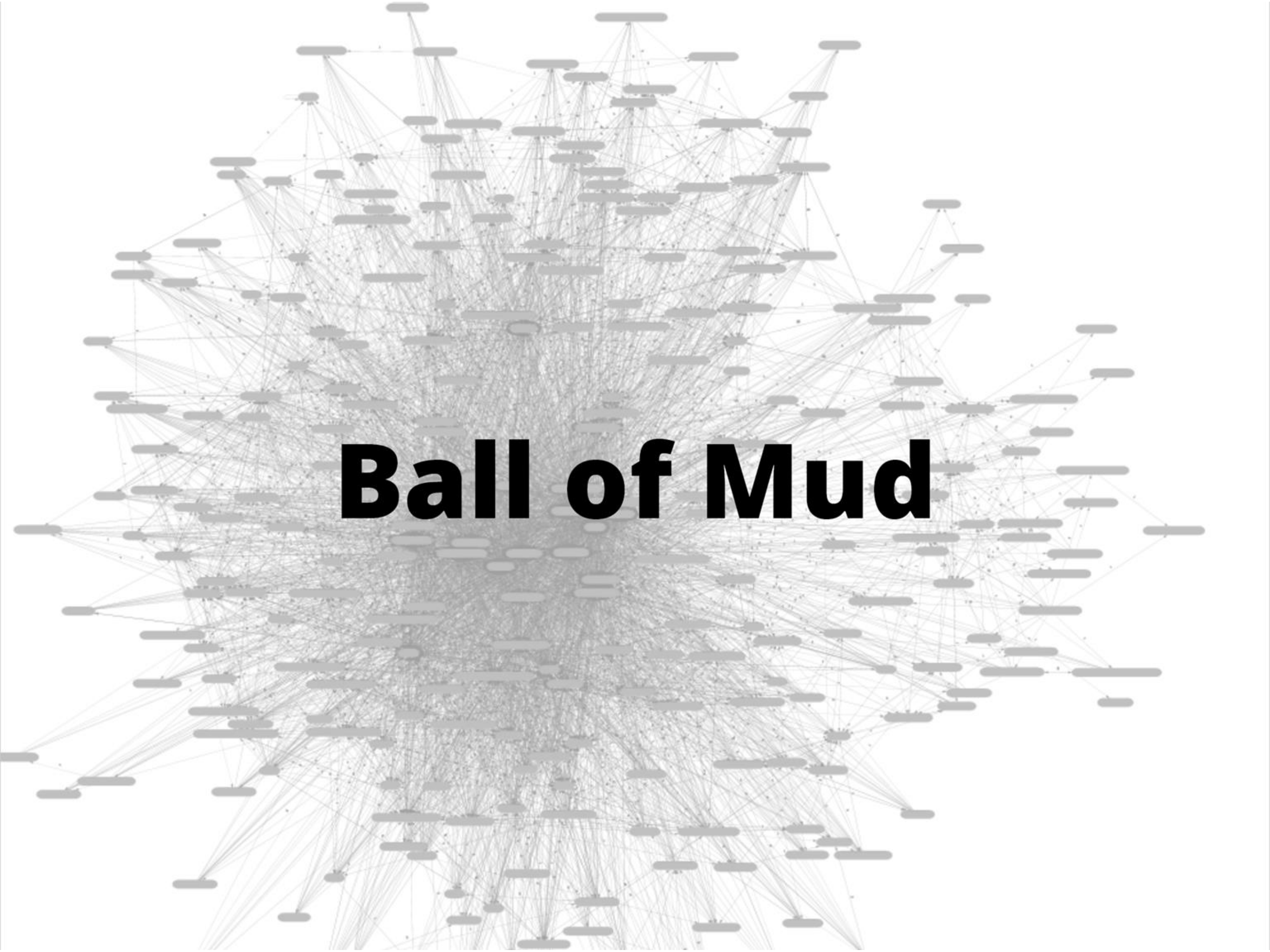
Ernesto Damiani, Claudio Ardagna – Università degli Studi di Milano

Cloud and Distributed Computing

Issues

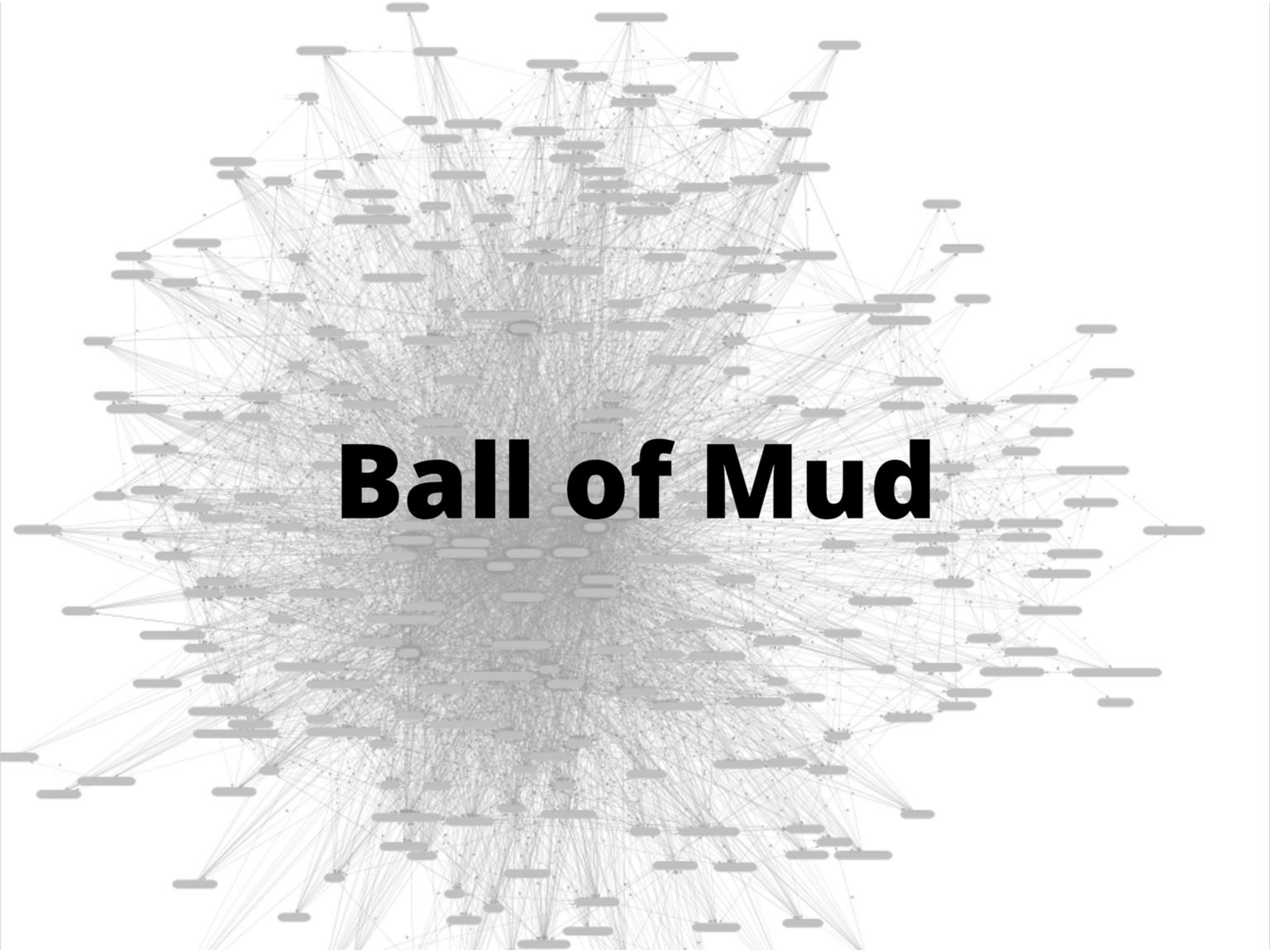
- Reference environment
  - Web systems
  - Jointly developed by several development teams
  - Traffic requiring horizontal scaling (e.g. load balancing between different copies of the system)

- Fact
  - These systems are often realized as monolithic or layered systems


Ball of Mud

Observed problems

- "expediency over design"
- Brian Foot & Joseph Yoder


Ball of Mud


FH16 XL
VOLVO

Monolithic software

- Monolithic software
  - One build and deployment unit
  - One code base
  - One technology stack (Linux, JVM, Tomcat, Libraries)

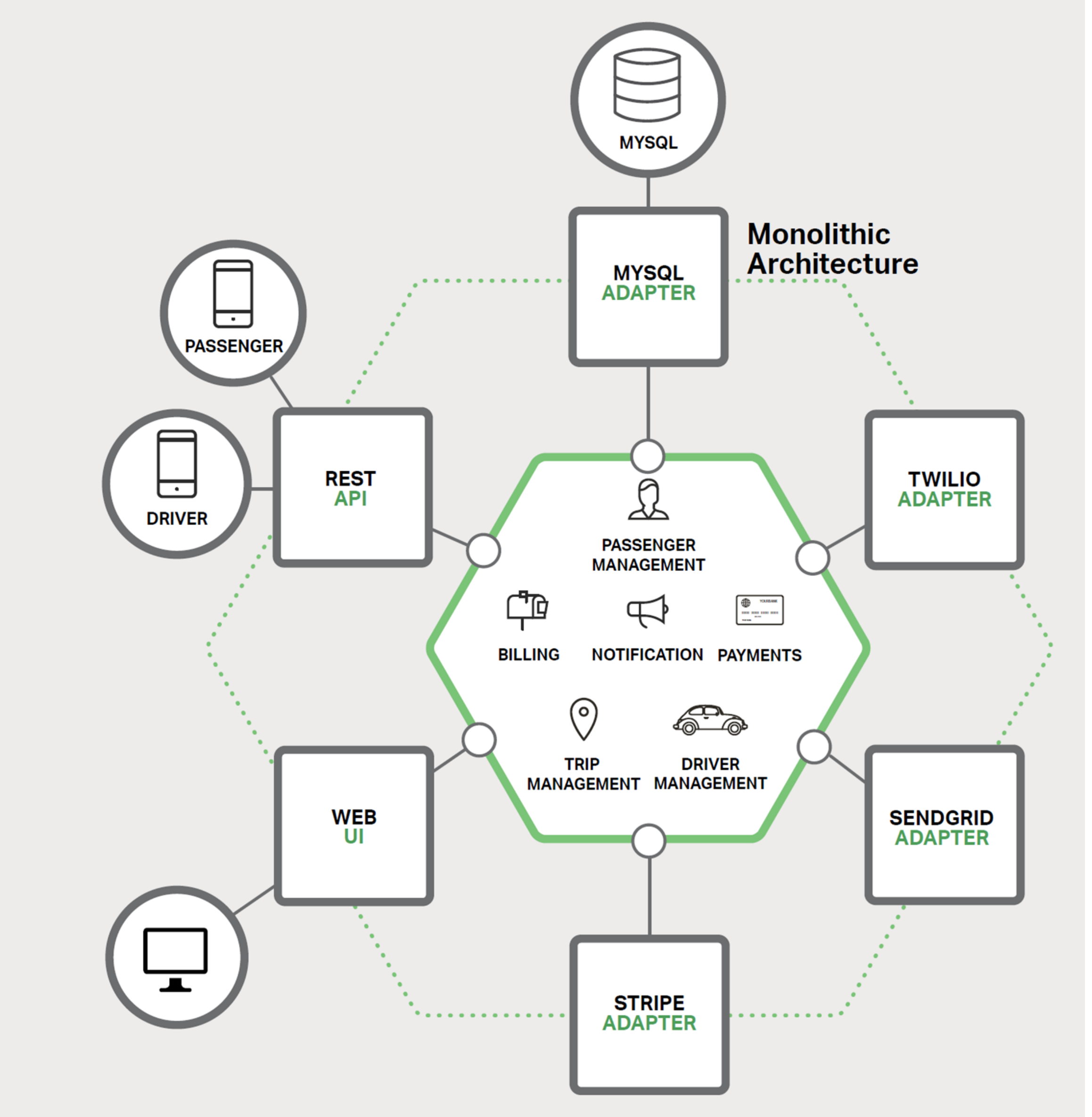
- Pros
  - Simple mental model for the developers
    - A single access unit for development, compilation and deployment
  - Simple scaling model for the operations
    - Multiple copies are executed behind a load balancer

Monolithic application: Example

- Application for calling a taxi, a rival of Uber and Hailo
  - A new project to develop manually or using code generators within platforms like Rails, Spring Boot, Play, or Maven

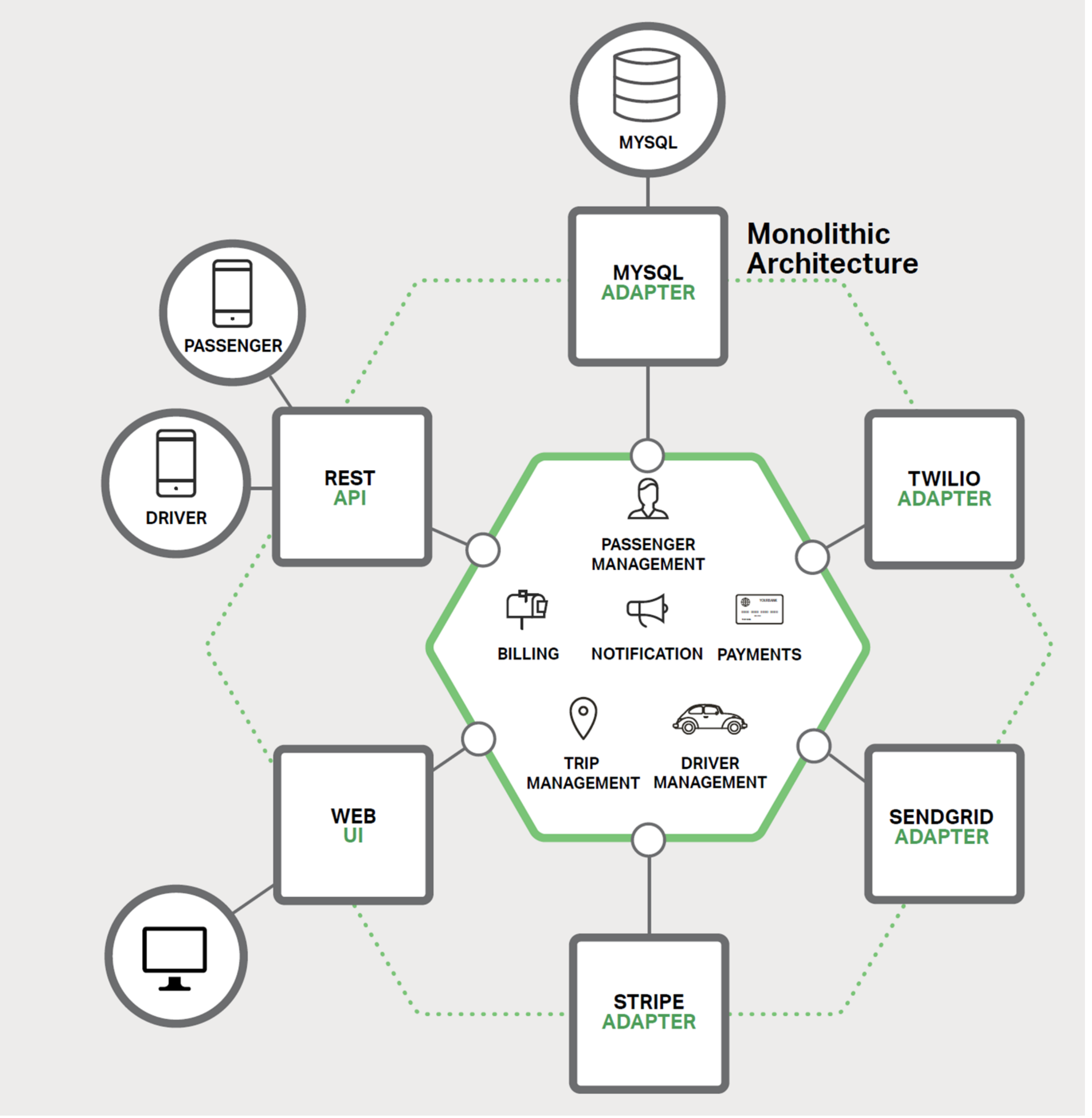
Monolithic application: Example

- The core of the application is the business logic
  - Made up of modules defining services, domain object and events.
- All around there are the adapters interfacing with the external world
  - Database access component
  - Messaging component producing/consuming messages
  - Web component exposing APIs or implementing a UI



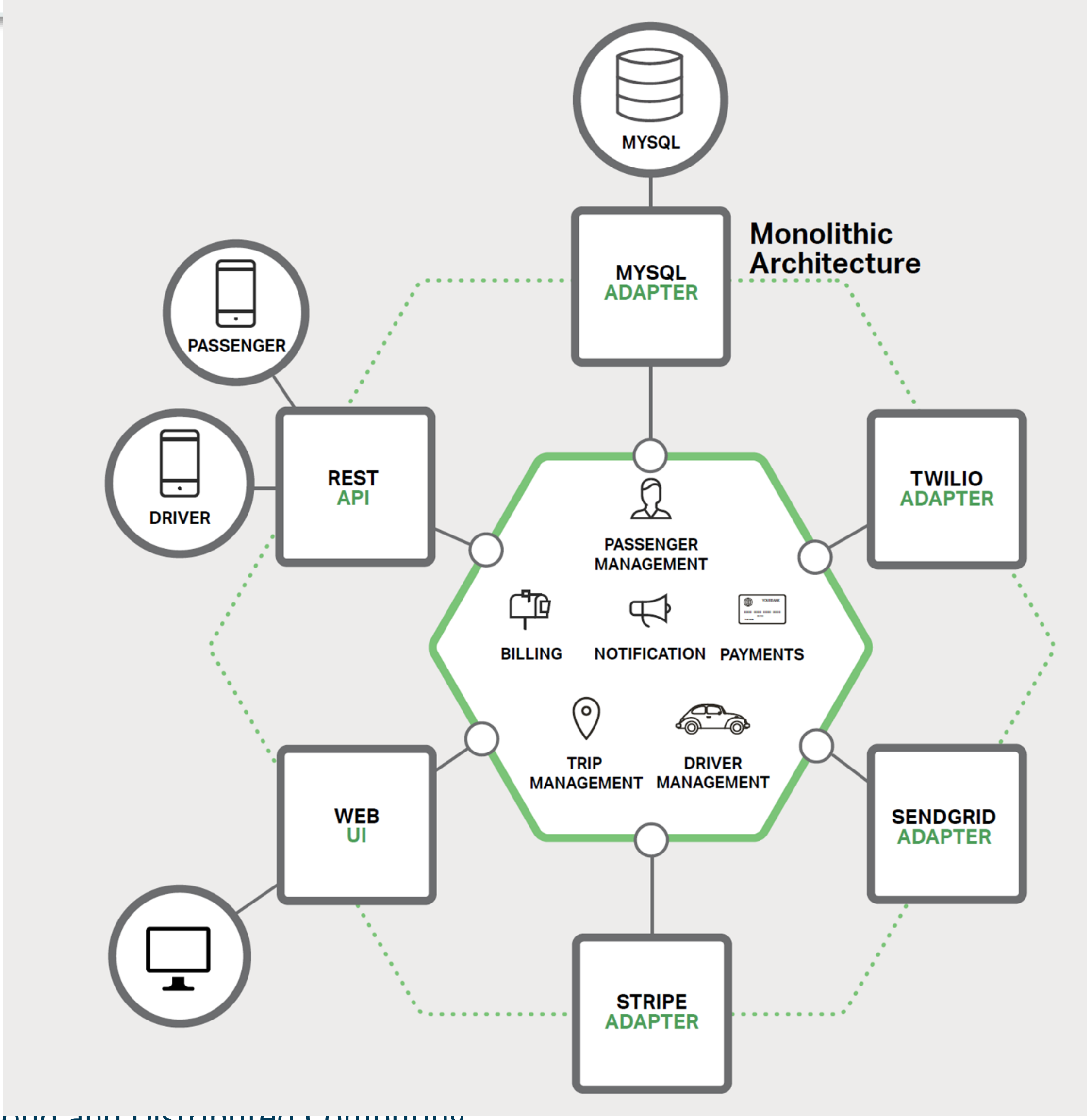
Monolithic application: Example

- Even if the architecture is modular…
- … the application is packaged and installed as monolith
  - Java applications packaged as WAR files and installed on application servers like Tomcat or Jetty



Monolithic application: Example

- Easy to develop
  - IDE and other tools are products for building a single application
- Easy to test
  - End-to-end testing launching the application and testing the UI with testing packages like Selenium
- Easy to deploy
  - It's enough to copy the application package within the server
- Scaling is supported by executing multiple copies behind a load balancer
- Works well in the first stages of the project

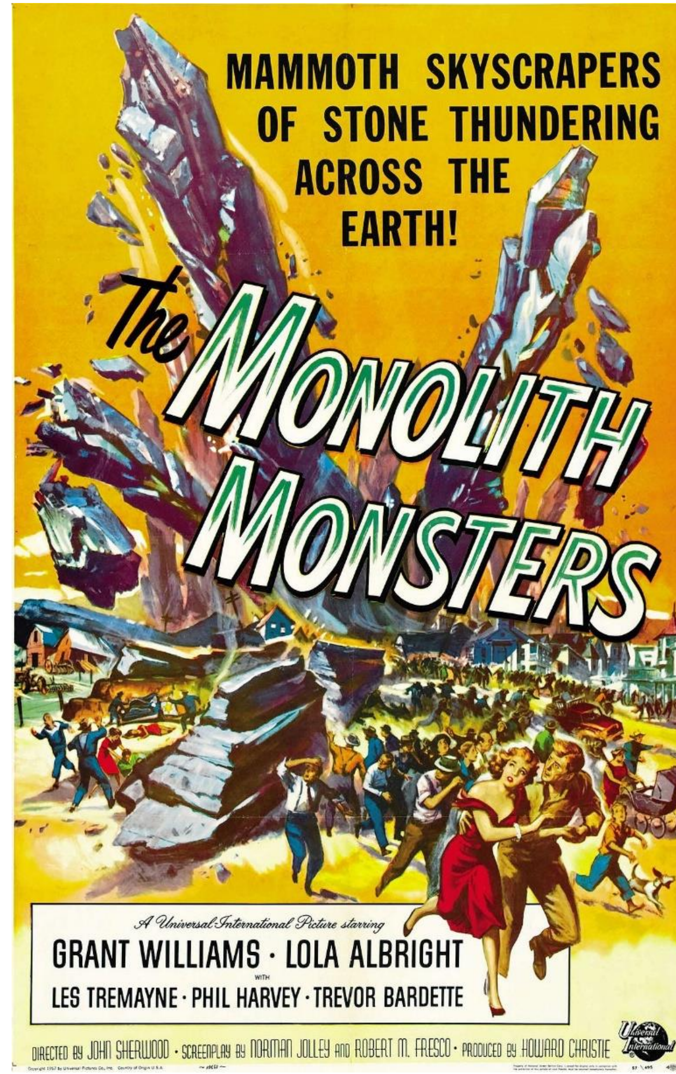


Issues with Monolithic Software

- The code base is huge and scares developers
- Development tools become overloaded
  - Refactoring requires minutes
  - Building requires hours
  - Testing with continuous integration requires days
- Limited scalability
  - Executing a copy of the system is resource-intense
  - Does not scale with the volume of data
  - Out-of-the-box
  - Limited deployment frequency
  - Re-deployment implies stopping the system
  - Re-deployment will fail and increase the perceived risk

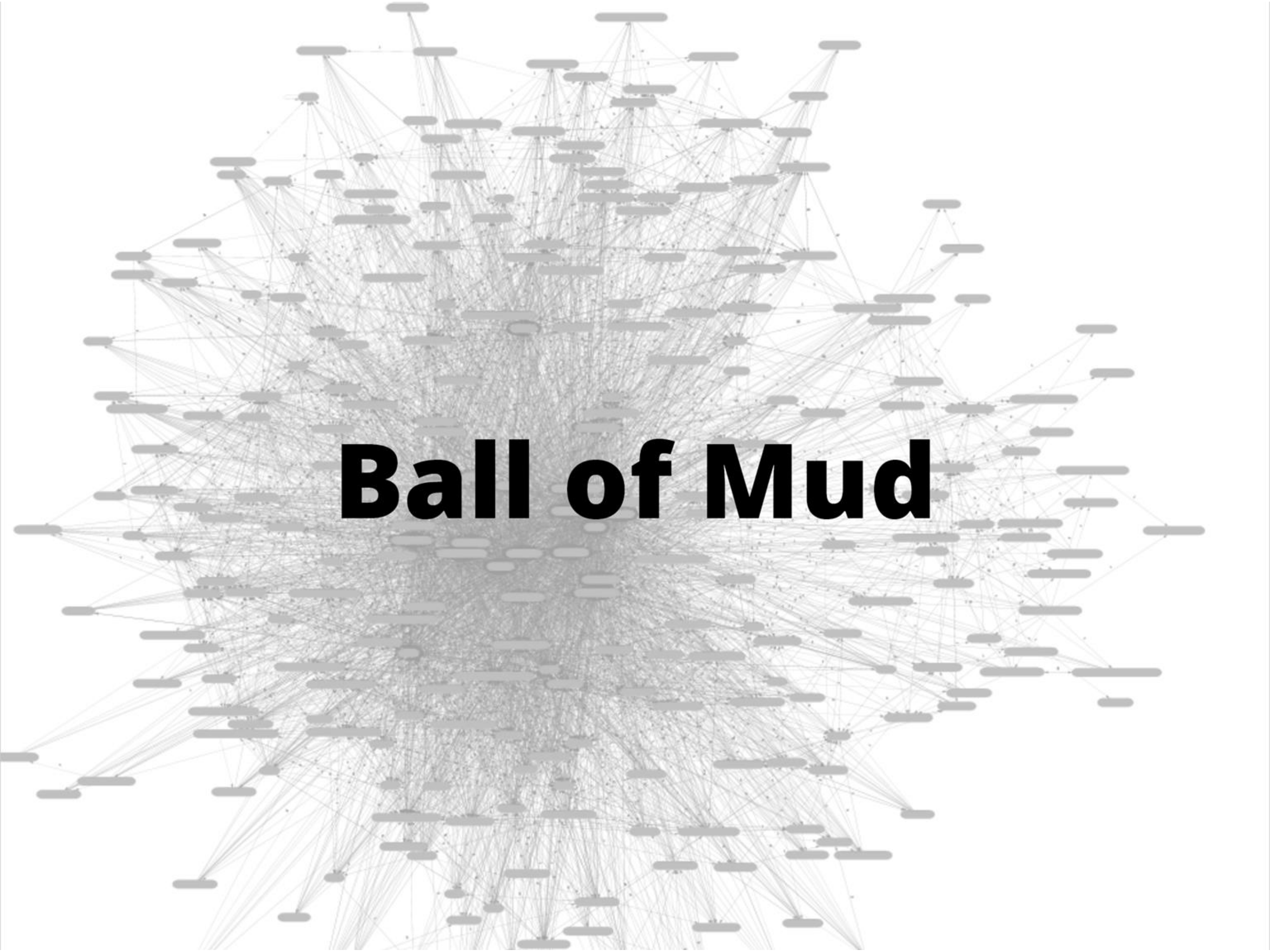
Issues with Monolithic Software

- The biggest problem is that successful applications grow over time and become huge
- The development team implements new use cases, which means adding code
- After few years the application becomes a monolith of monstrous dimensions



Issues with Monolithic Software

- Monolith monster = world of pain in the development organization
- Every attempt of agile development and release will fail
- The application will become so big that it will be impossible to manage and understand
  - Fixing bugs and implementing new features become difficult and require a lot of time
  - Downward spiral… a code difficult to understand increases the chance of uncorrect updates
- The result is a monstrous, incomprehensible…


Ball of Mud


DOH!

Issues with Monolithic Software

- Other cons
  - Inefficient development
    - High startup time
    - If the restart happens frequently, most of day is spent waiting
  - Complex applications are an obstacle to continuous deployment
    - State of art requires pushing updates to production more times in a day
    - With monolithic applications you need to re-deploy from scratch to update just a small part of the application
  - Monolithic applications are difficult to scale if modules require conflicting resources

Issues with Monolithic Software

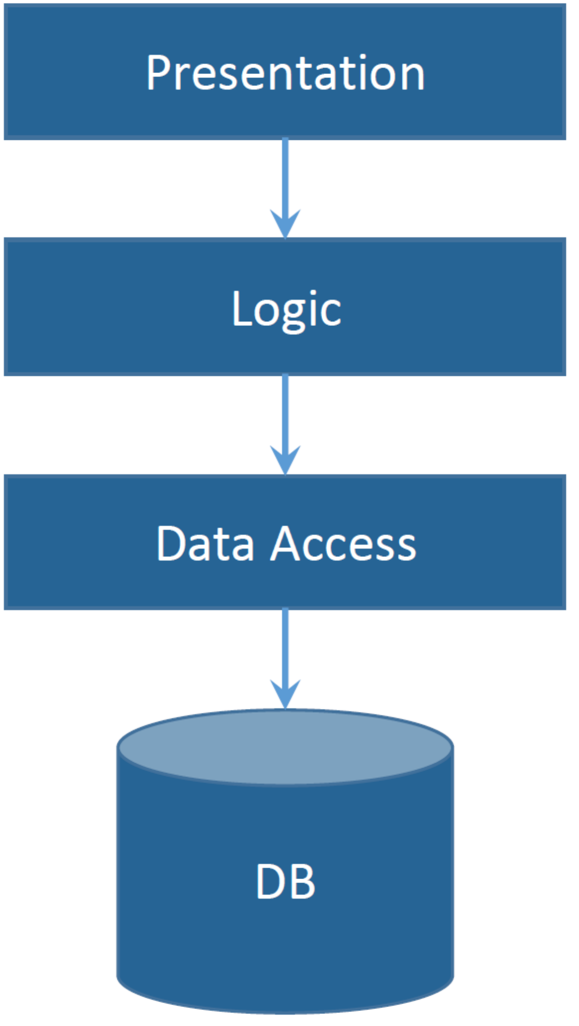
- Reduced reliability because all the modules are executed in the same process
  - A bug in a module, for example memory loss, may cause a failure in the whole process
- Finally, monolithic applications make it difficult to adopt new frameworks and languages
  - For example, consider a code with 2 million of LoC written with the framework XYZ. It is very expensive and slow to rewrite the whole application with a different framework, even if that framework is better

Issues with Monolithic Software

To summarize: you have a successful business-critical application that has grown into a monstrous monolith that very few, if any, developers understand. It is written using obsolete, unproductive technology that makes hiring talented developers difficult. The application is difficult to scale and is unreliable. As a result, agile development and delivery of applications are impossible.

Layered system

- The monolithic system is decomposed in layers
  - Presentation, logic, data access
  - At most a set of technologies for each layer
  - Presentation: Linux, JVM, Tomcat, Libs, EJB client, JavaScript
  - Logic: Linux, JVM, EJB container, Libs
  - Data Access: Linux, JVM, EJB JPA, EJB container, Libs



- Pros
  - Simple mental model, simple dependencies
  - Simple deployment and scaling model

Issues of layered systems

- Big code base (one for each layer)
- … with the same impact on development, build, and deployment
- Better scaling, but still limited
- Limited staff growth: to simplify, one team for layer works well
  - Developers become specialized in a specific layer
  - The communication among teams is influenced by different level of competences

Growth of the systems beyond limits

- Applications and teams have to grow beyond the limits imposed by monolithic and layered systems
  - Often done in an uncontrolled way
- Big companies have layered systems interacting in an undocumented way
- These systems often fail in an unexpected way
- How can a company grow while maintaining an architecture and an IT vision?
  - Big successful companies (Amazon, Netflix) trace the route toward the microservice architecture

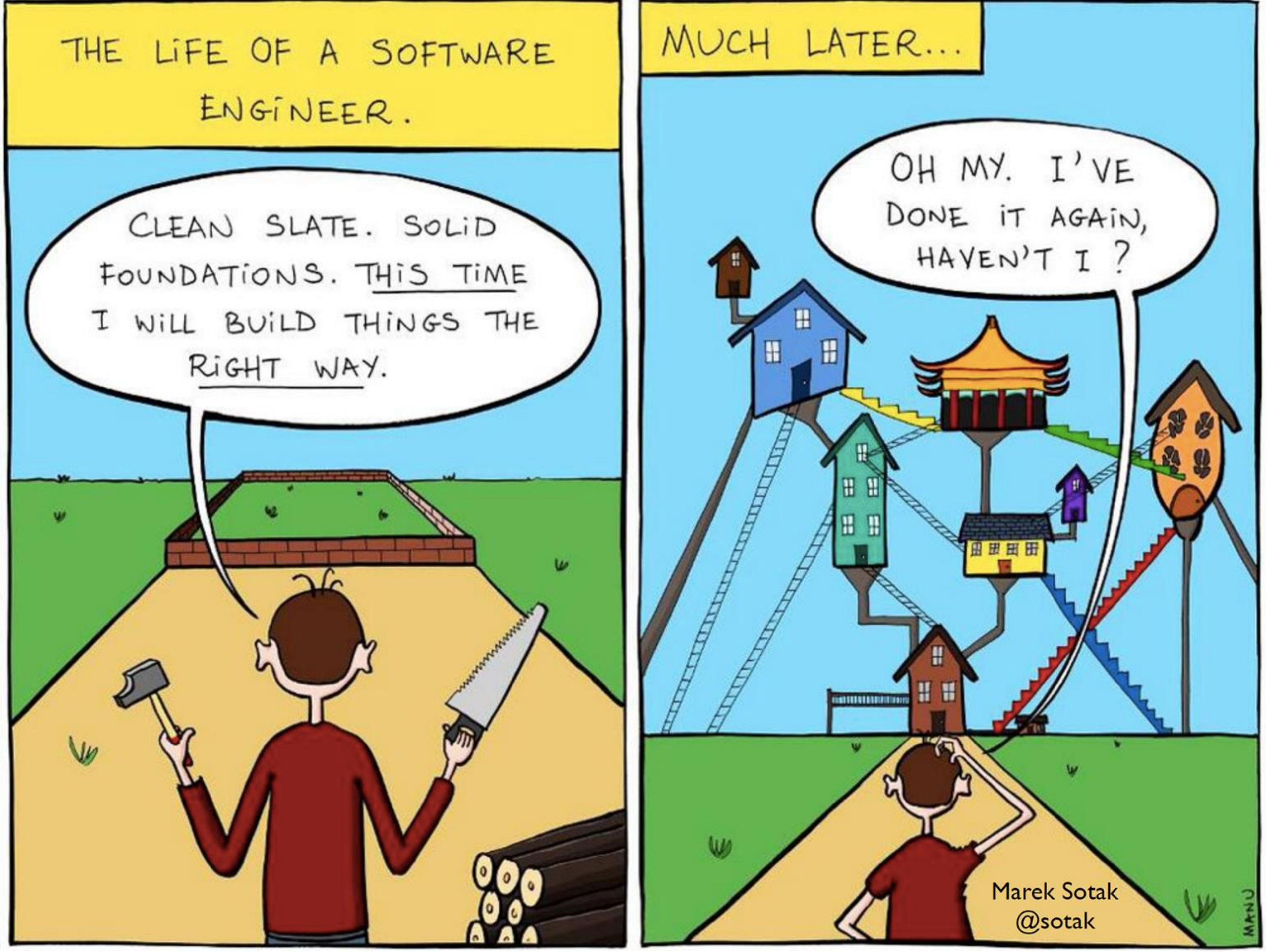
So what? Microservices


Modena
MOTORSPORT

History

- 2011: term coined in a software architecture conference close to Venice
- May 2012: microservice identified as the most proper term
- March 2012: “Microservices: Java, the Unix Way” at 33rd degree by James Lewis
- September 2012: “µService Architecture“ at Baruco (Barcelona Ruby Conference 2012) by Fred George
- Adrian Cockroft in the meantime becomes the pioneer of this style at Netflix, calling it “fine grained SOA”

http://martinfowler.com/articles/microservices.html#footnote-etymology


THE LIFE OF A SOFTWARE ENGINEER.
CLEAN SLATE. SOLID FOUNDATIONS. THIS TIME I WILL BUILD THINGS THE RIGHT WAY.
MUCH LATER...
OH MY. I'VE DONE IT AGAIN, HAVEN'T I ?
Marek Sotak
@sotak
MANU

Principle

- At a logical level, microservice architecture means

  functional system decomposition into manageable and independently deployable components

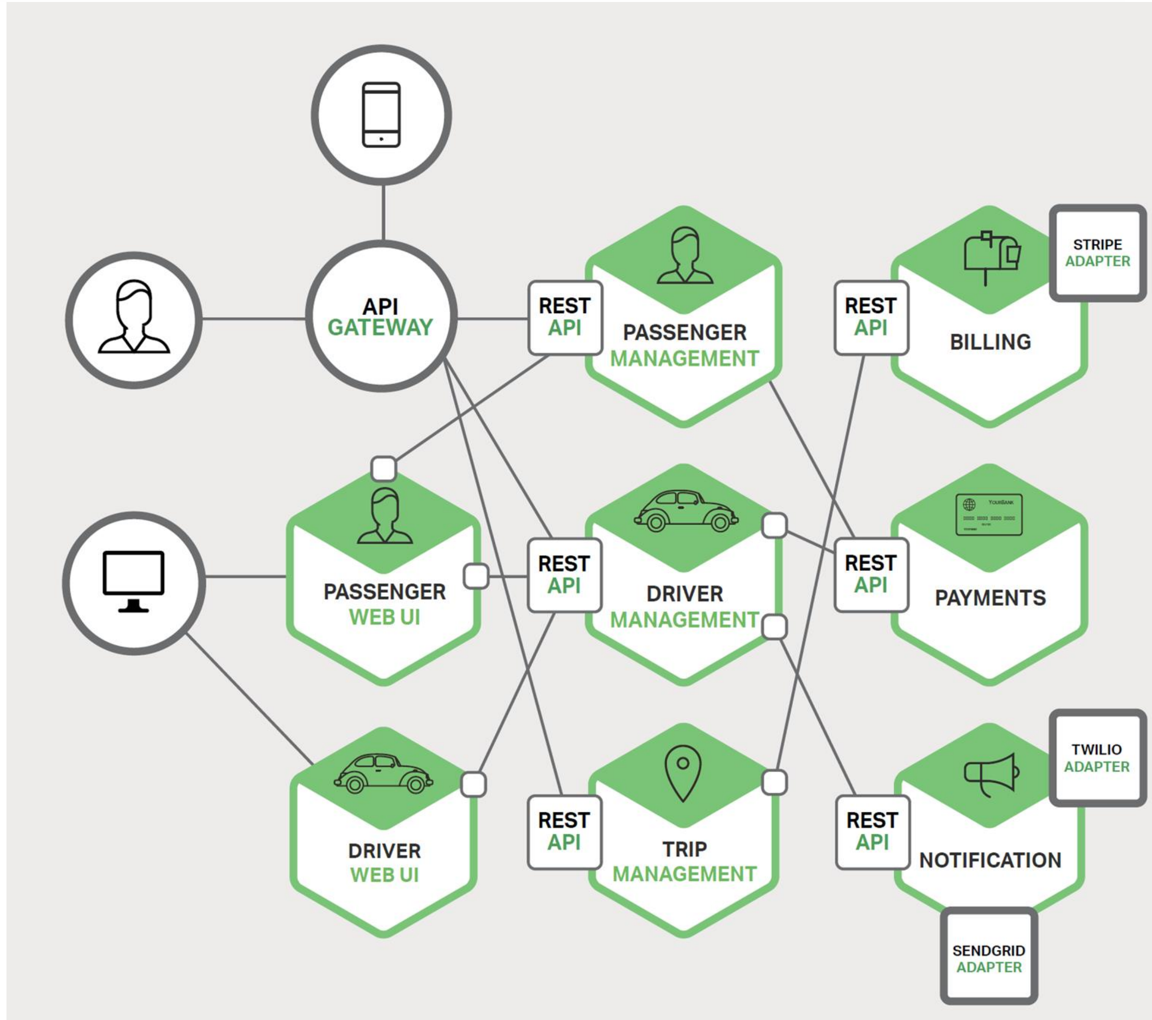
- The term “micro” is referred to the dimension
  - A microservice has to be manageable by a single development team
- Functional system decomposition means vertical decomposition (as opposed to the horizontal approach of a layered system)
- Independent deployment means no shared state or inter-process communication (often through HTTP interfaces REST-like)

Principle

- The idea is to divide the application into small interconnected services
- Each service typically implements a set of features or functionalities like order management, customer management, etc.
- Each microservice is a mini-application with its own hexagonal architecture consisting of business logic and several adaptors
  - Some microservices expose an API that can be used by other microservices, or by client applications
  - Other microservices implements a web interface
- At run time, each instance is, often, a virtual machine or a Docker container

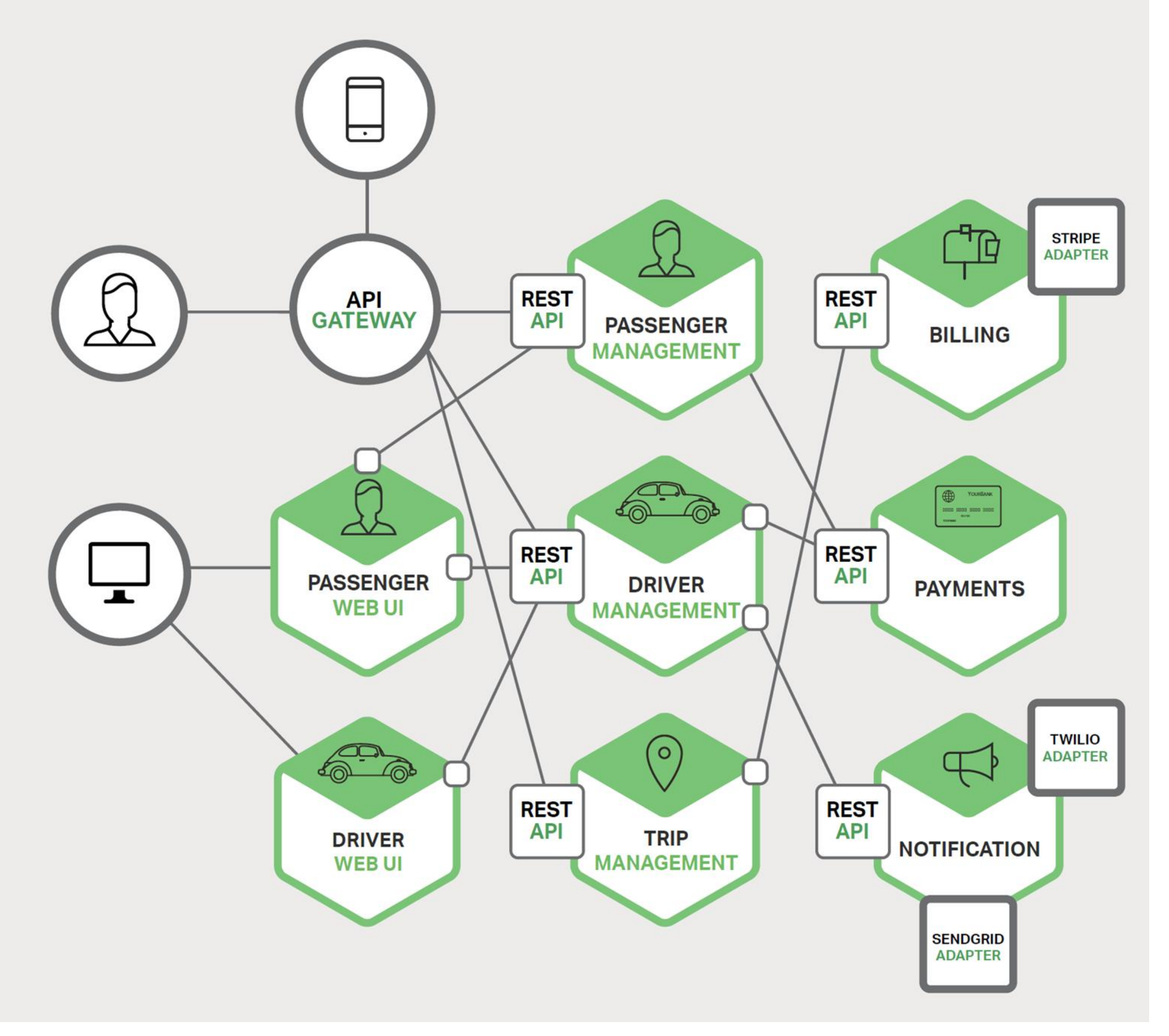
Microservice architecture: Example

- Each functional area is implemented with its own microservice
- The web application is divided into a set of simpler web applications
- It simplifies the deployment by distinguishing operations for specific users, or specialized use cases



Microservice architecture: Example

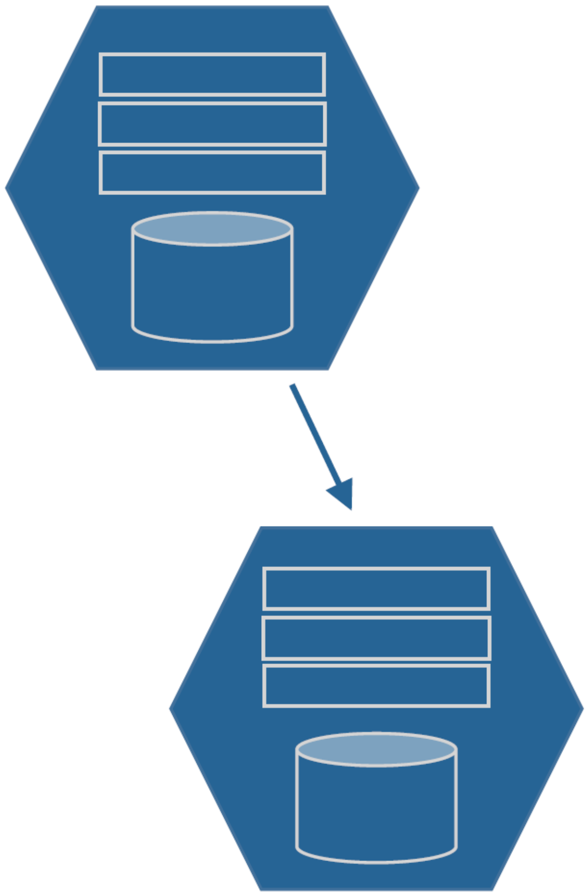
- Each backend service exposes an API and several services use the API of other services
- Services can use asynchronous communication based on messages
- Communications are mediated by a known broker like an API Gateway
  - API Gateway is responsible for activities such as load balancing, caching, access control, API metering, and monitoring



More in detail

- Each microservice is functionally complete with
  - Resource representation
  - Data management
- Each microservice manages a resource
  - Client
  - Shop Item
  - Cart
  - Checkout

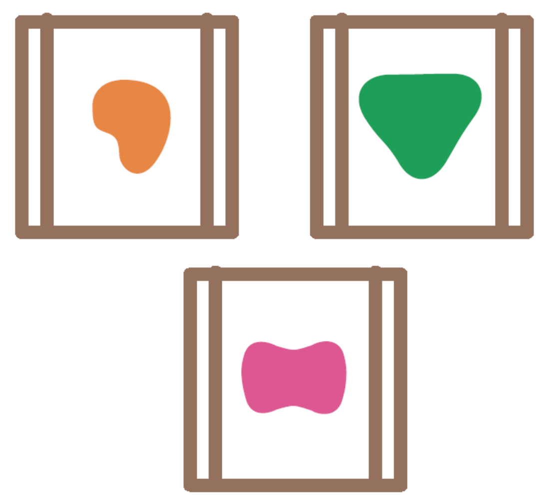
Microservices are called *fun-sized services*, because they are *“still fun to develop and deploy”*



Easy and fun

- A monolithic application has all the functionalities in a single process…

- A microservice architecture has each element/functionality in a separated service…



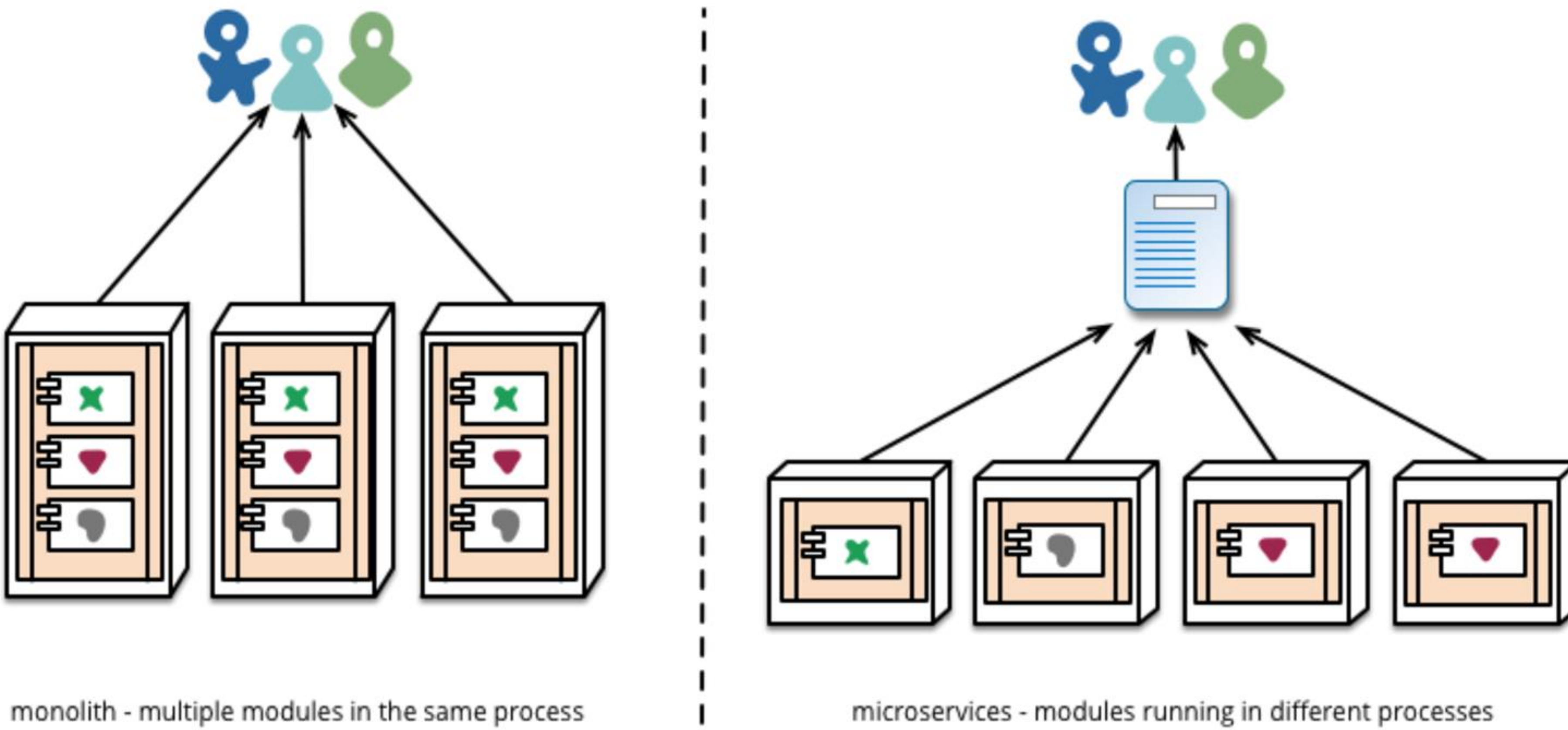
Independent deployment is fundamental

- Allows separation and independent evolution
  - Code base
  - Technological stack
  - Scaling
  - Functionalities

Independent code base

- Each service has its own software repository
  - The code is maintainable for the developers
    - *It fits into their brain*
  - Tools work faster – building, testing, refactoring the code take seconds
  - Service startup takes few seconds
  - There are no accidental cross-dependencies between different code bases

Independent Process



Transforming a monolith into microservices

Introduction

- Transforming a monolithic application into a set of microservices is a sort of an application modernization
  - That’s what developers have been doing for decades

- A strategy to not use “Big Bang rewrite”
  - Building a microservice application from scratch
  - Even though it can sound interesting, it’s extremely risky and may fail with high probability

Big Bang Rewrite

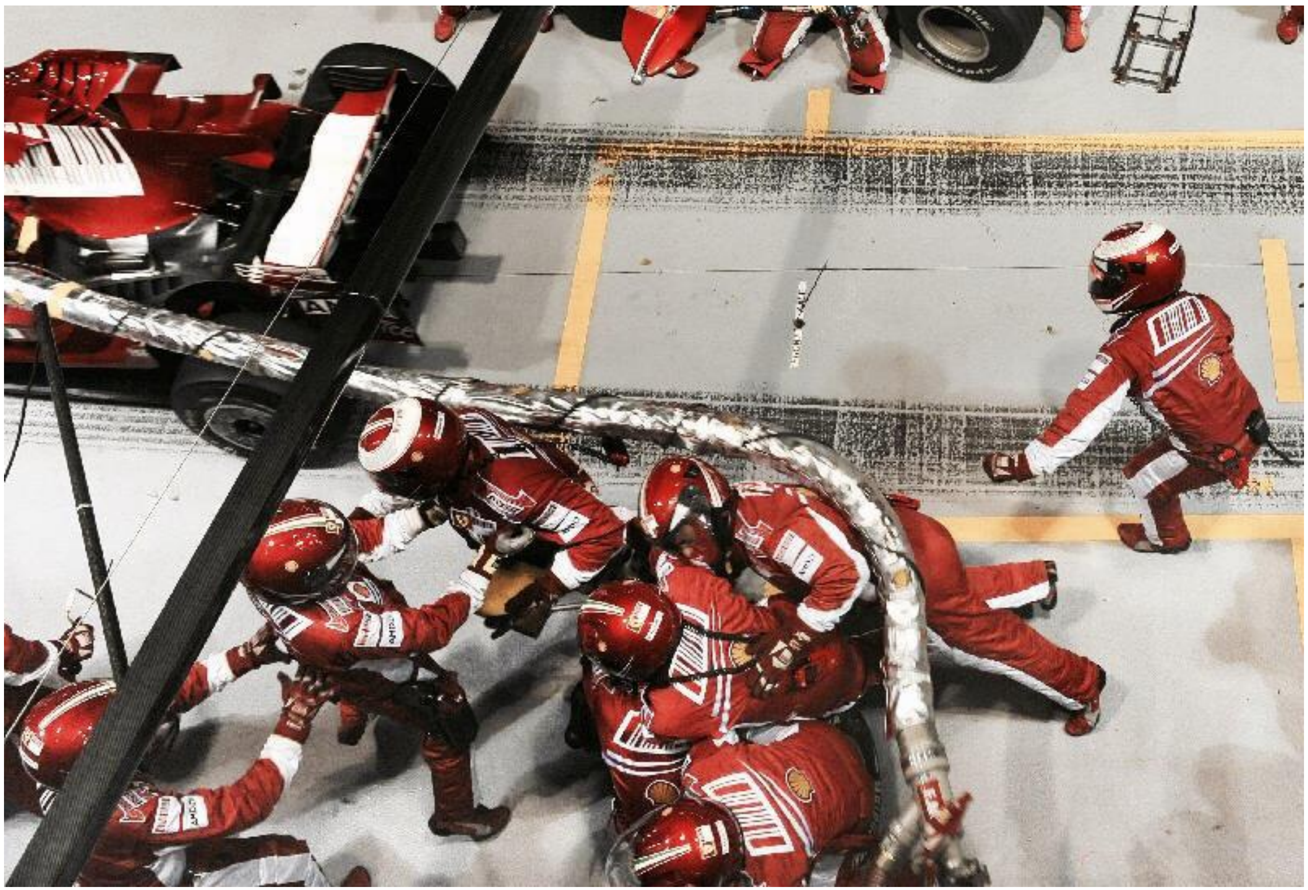
- “The only thing a Big Bang rewrite guarantees is a Big Bang!” cit. Martin Fowler

Refactoring

- Incremental refactoring of the monolithic application
- Gradual adding of new functionalities and extension of existing functionalities as microservices
  - Modifying the monolithic application in a complementary way
  - Executing microservices and modified monolith in tandem
  - The monolithic application becomes smaller until it disappears or becomes itself another microservice

Refactoring

- Still risky, but less than a Big Bang rewrite.



Main Strategies

- Strategy #1 – Stop Digging
- Strategy #2 – Split Frontend and Backend
- Strategy #3 – Extract Services

Strategy #1 – Stop Digging

Introduction

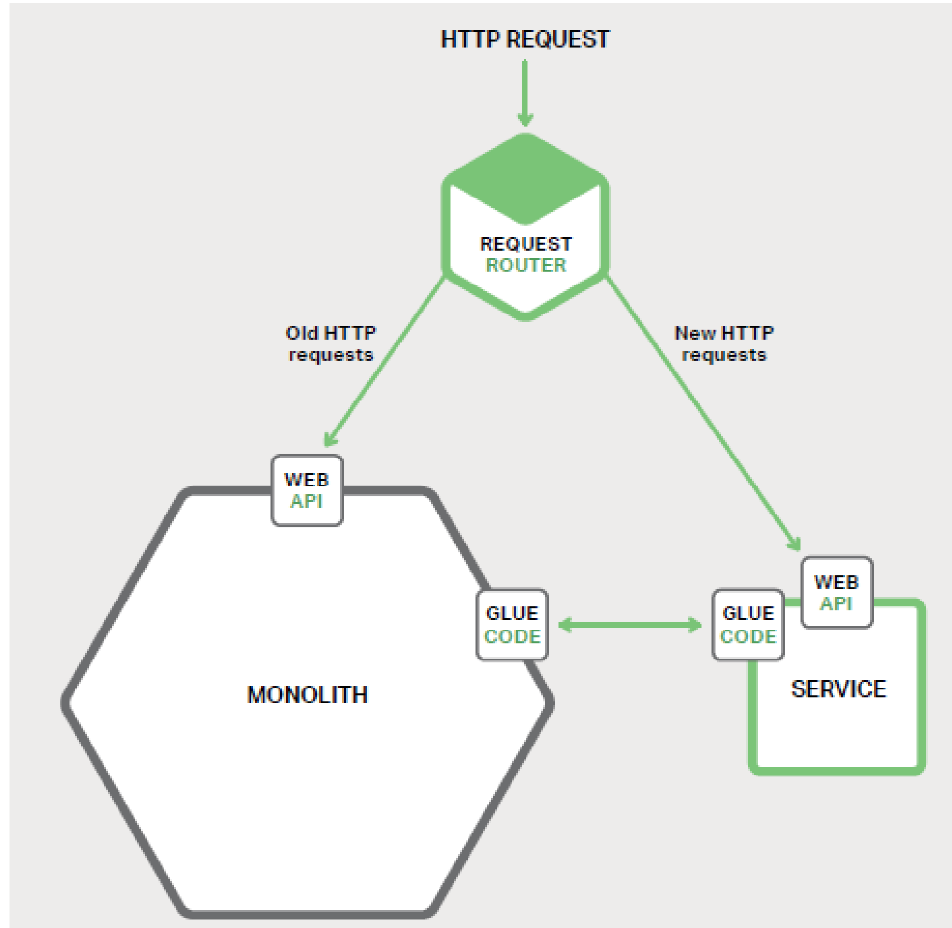
If You're in a Hole, Stop Digging

Soft migration

- When a migration has become unmanageable, you need to stop the enlargement of the monolith
- New functionalities should not add new code to the monolith
- The idea is to put the new code in standalone microservices

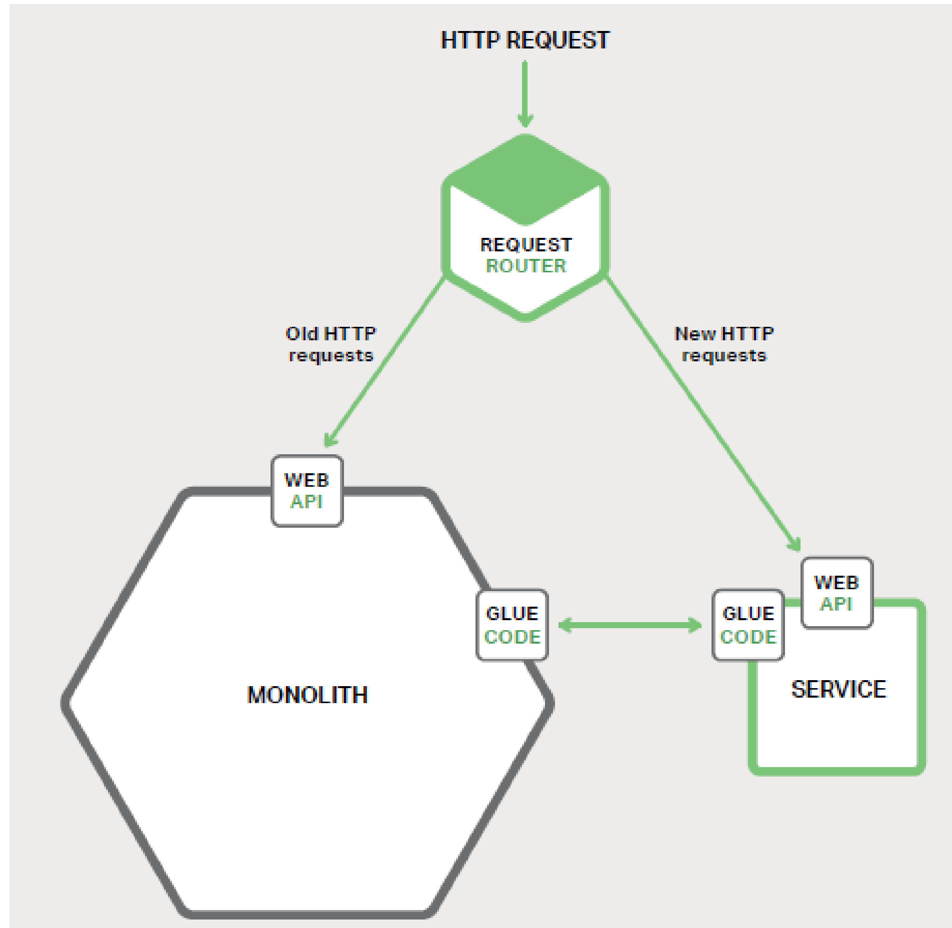
Soft migration

- The router is similar to the API gateway
- Legacy requests go to the monolith
- Requests to new functionalities go to the microservice



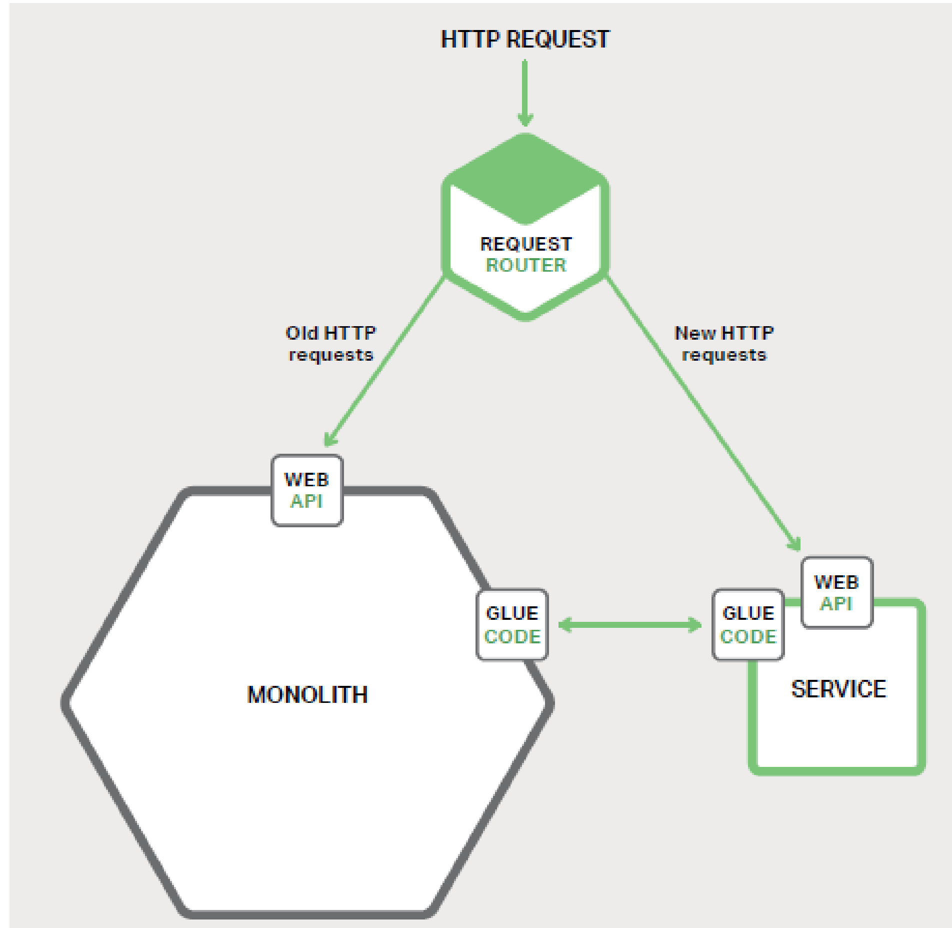
Glue Code

- Glue code integrates the service with the monolith
- The service often needs to access data managed by the monolith
- Glue code is responsible for data integration



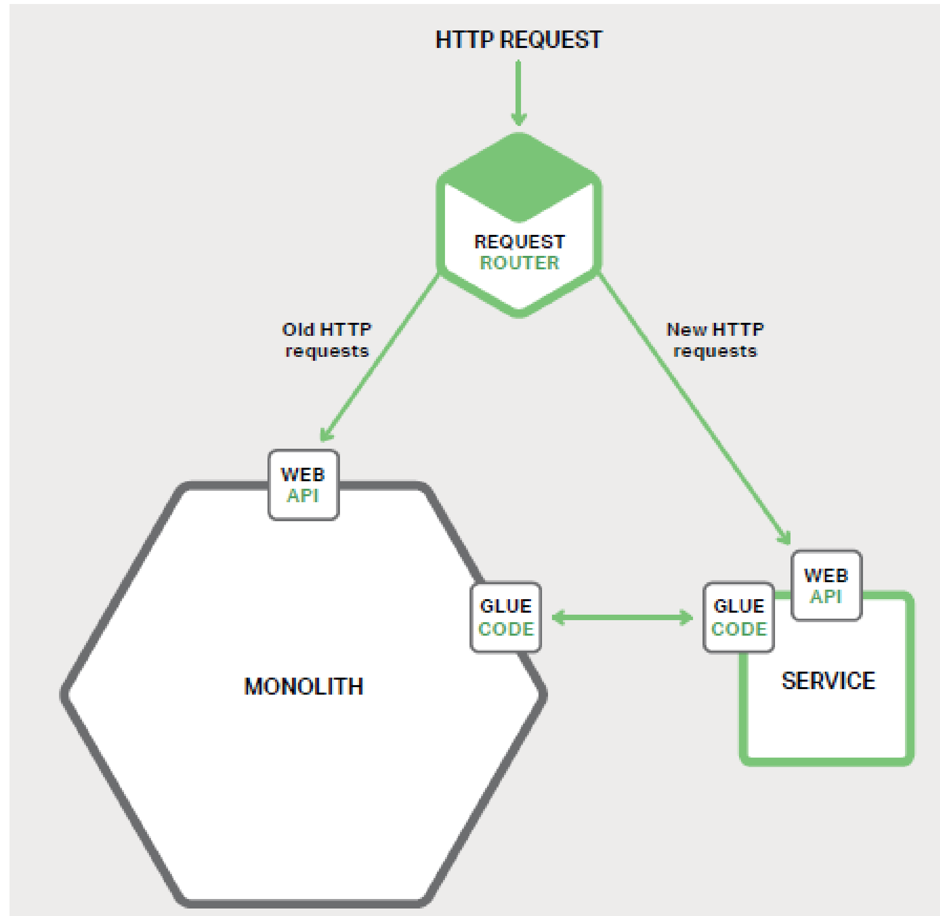
Glue Code

- Three strategies for accessing monolith's data
  - Invoking a remote API offered by the monolith
  - Accessing directly to the monolith database
  - Keeping a copy of the data, which is synchronized with the monolith database



Glue Code

- The glue code is also called *anti-corruption layer*
- It avoids the service with its own domain model to be polluted by concepts of the domain model of the monolith
- The glue code translates between two different models



Strategy #1: Pros and cons

- Pros
  - Implementing new functionalities as a light service avoiding making the monolith even more unmanageable
  - The service can be developed, deployed, and scaled independently from the monolith
  - Advantages coming from a microservice architecture for each new created service
- Cons
  - Does not solve the monolith issue
  - To solve it you need to destroy the monolith (next strategies)

Strategy #2 – Splitting Frontend and Backend

Layer separation

- Separating the presentation layer from business logic and data access layer
  - Presentation layer – Components handling HTTP requests and implementing a (REST) API or a HTML-based web UI
    - Often it is a big code base
  - Business logic layer – Core application components implementing business rules
  - Data-access layer – Components providing access to infrastructure components like database and message broker

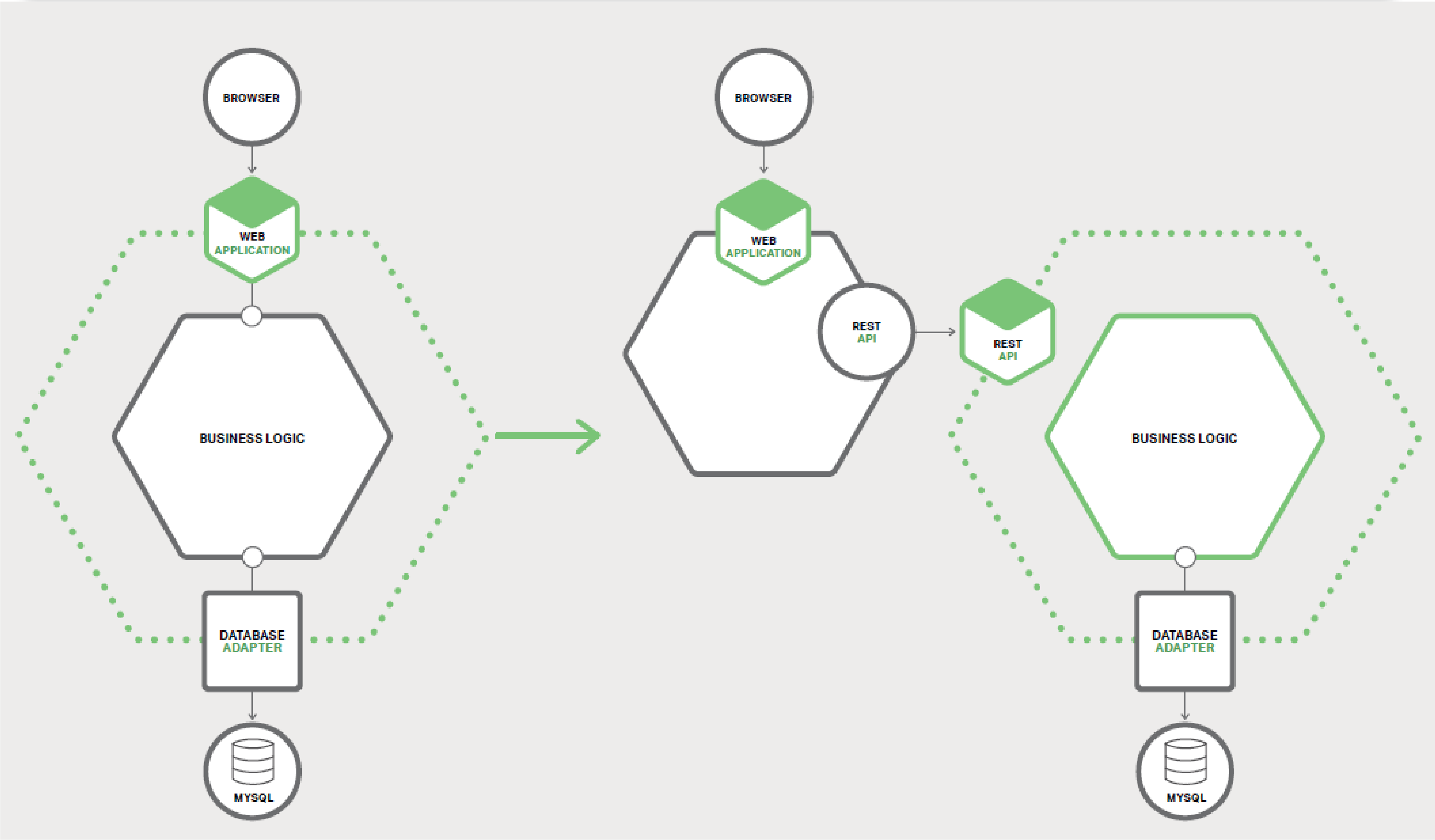
How to

- There is often a clear separation between presentation logic, on one side, and business and data access logic, on the other side

- The business tier has a coarse-grained API consisting of one or more façade, encapsulating components with the business logic

- This API is the natural path to follow to divide the monolith into two smaller applications
  - An application contains the presentation layer
  - The other one the business and the data-access logic

How to



BROWSER
WEB
APPLICATION
BUSINESS LOGIC
DATABASE
ADAPTER
MYSQL
BROWSER
WEB
APPLICATION
REST
API
REST
API
BUSINESS LOGIC
DATABASE
ADAPTER
MYSQL

Strategy #2: Pros and cons

- Pros
  - It allows independent development, deployment, and scaling of two applications
    - Presentation-layer: developers can rapidly iterate the user interface and execute integration testing
  - A remote API is exposed and can be called by the developed microservices

- Cons
  - Partial solution
  - One or both the applications will become an unmanageable monolith

- There is the need of a third strategy to eliminate the monolith/s

Strategy #3 – Extract Services

Shrinking the Monolith

- Transforming the modules of the monolith into standalone microservices
- Once enough modules have been converted, the monolith will no longer be an issue
- The monolith disappears and becomes so small to be considered just another service

Shrinking the Monolith

Phase 1: Prioritizing Which Modules to Convert into Services

- A monolithic complex application is composed of dozens or hundreds modules, all of them can be eventually converted

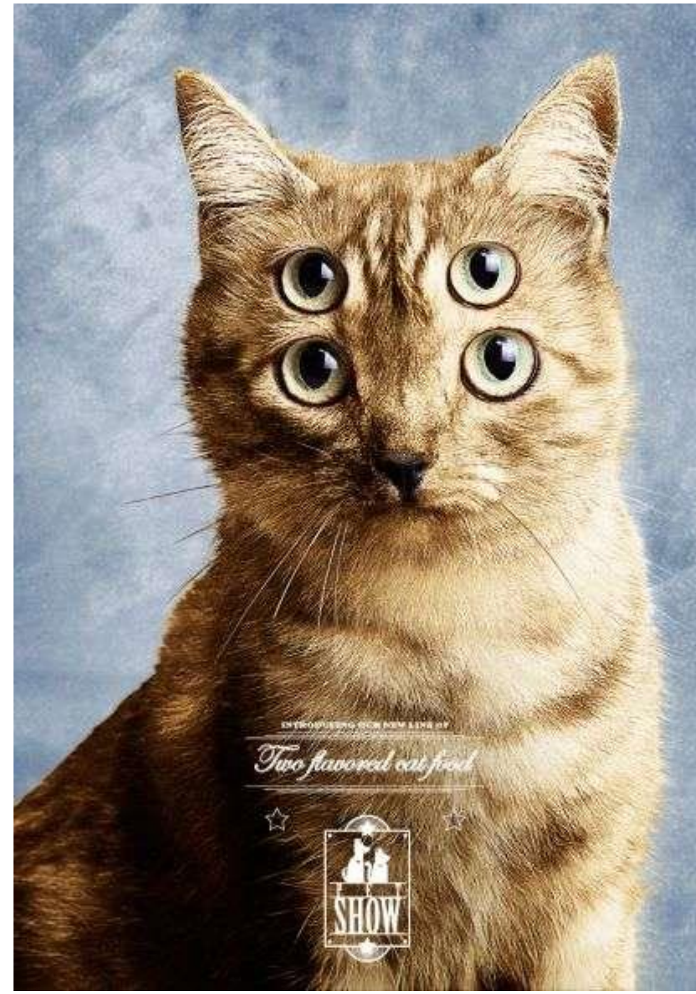
- How to choose which module to convert?
  - Starting with a few simple-to-extract modules (build experience)
  - Then, going on with modules giving more pros

Phase 1: Prioritizing Which Modules to Convert into Services

- Convert a module to a service is time consuming
- Prioritize modules according to benefits
  - Extract modules that change frequently
  - The converted modules support faster development processes
    - Development and deployment are independent from the monolith

Phase 1: Prioritizing Which Modules to Convert into Services

- Ordering first modules having requirements in terms of resources that are significantly different from the rest of the monolith
  - For example, converting a module having an in-memory database into a service
  - Once done, the service can be deployed on a host, being bare metal server, VM or cloud instances, with more memory
- Ordering first modules implementing computationally expensive algorithms
  - The service can be deployed on a host with many CPUs
- The conversion of modules with specific requirements in terms of resources into services makes the application easier and less expensive to scale



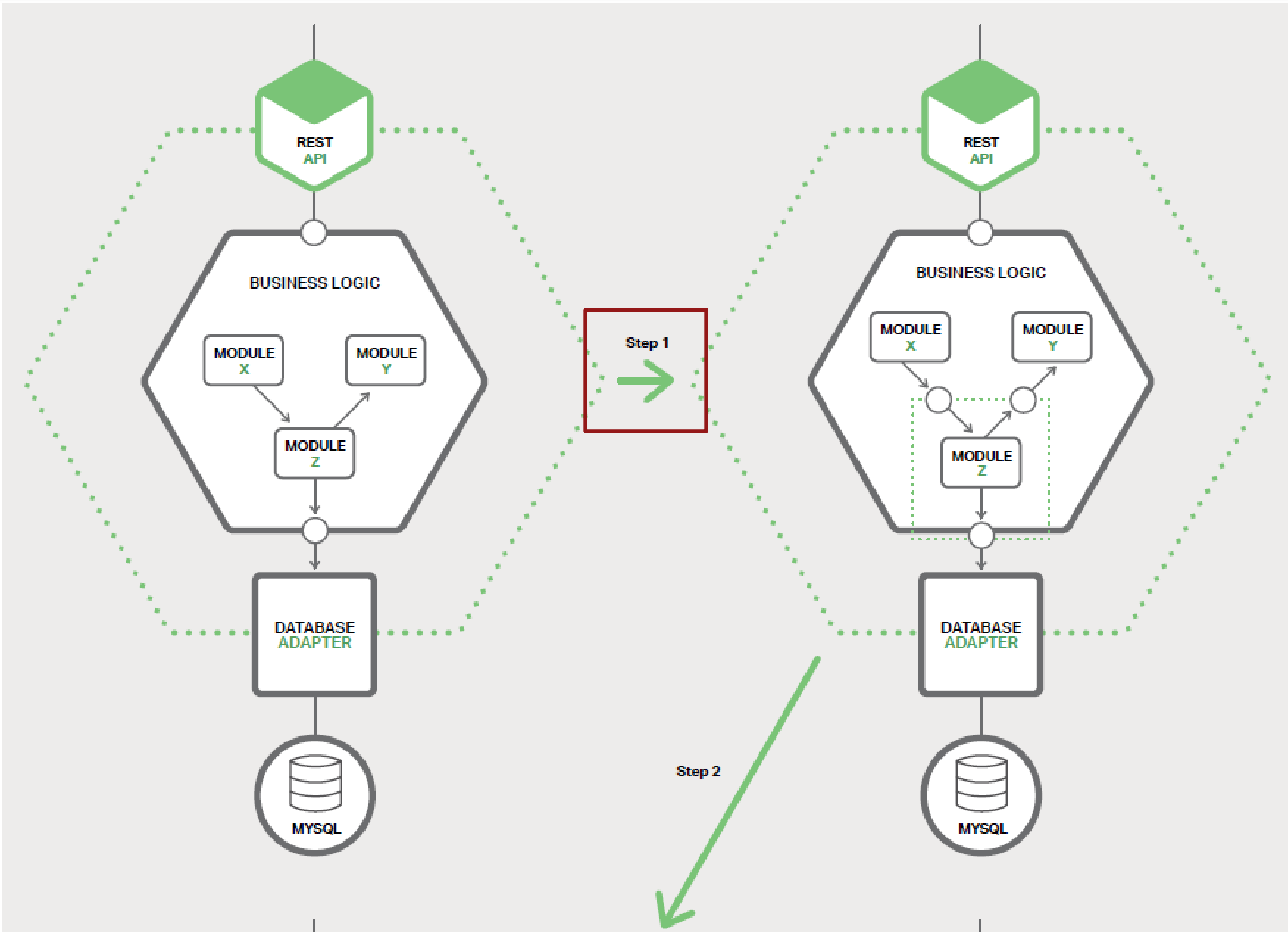
Phase 1: Prioritizing Which Modules to Convert into Services

- Searching for modules having clear and recognizable interactions with the external world
- It is easy and less expensive to modify a model having simple boundaries into a service
  - For example, a module communicating with the rest of the application with asynchronous messages
  - It is relatively less expensive and easy to transform these modules into microservices

Phase 2: Extract a Module

- Defining a clear interface between the module and the monolith
  - Bidirectional API, since the monolith will need data managed by the service and vice versa
- It is difficult to implement an API because of the many dependencies and interaction patterns between the module and the application
  - Often much code has to be modified to break the dependencies among the classes

Phase 2: Extract a Module (Step 1)



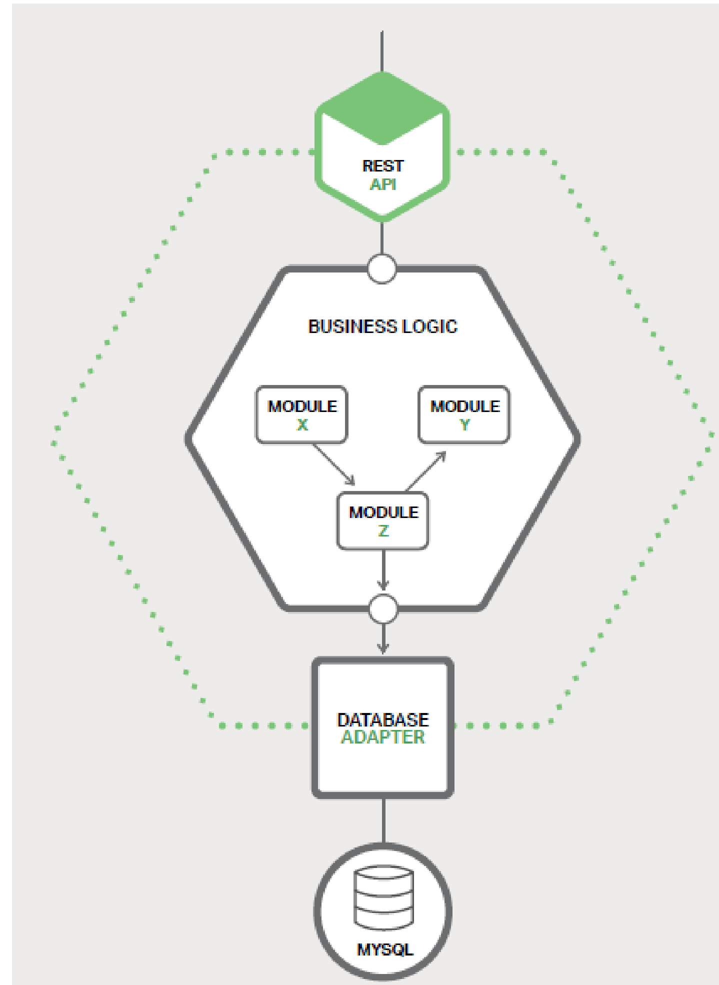
Phase 2: Extract a Module (Step 1)

- After having implemented the external interface, the module becomes a service

- Need of writing the code allowing the monolith and the service to communicate, through an API using an inter-process communication (IPC) mechanism

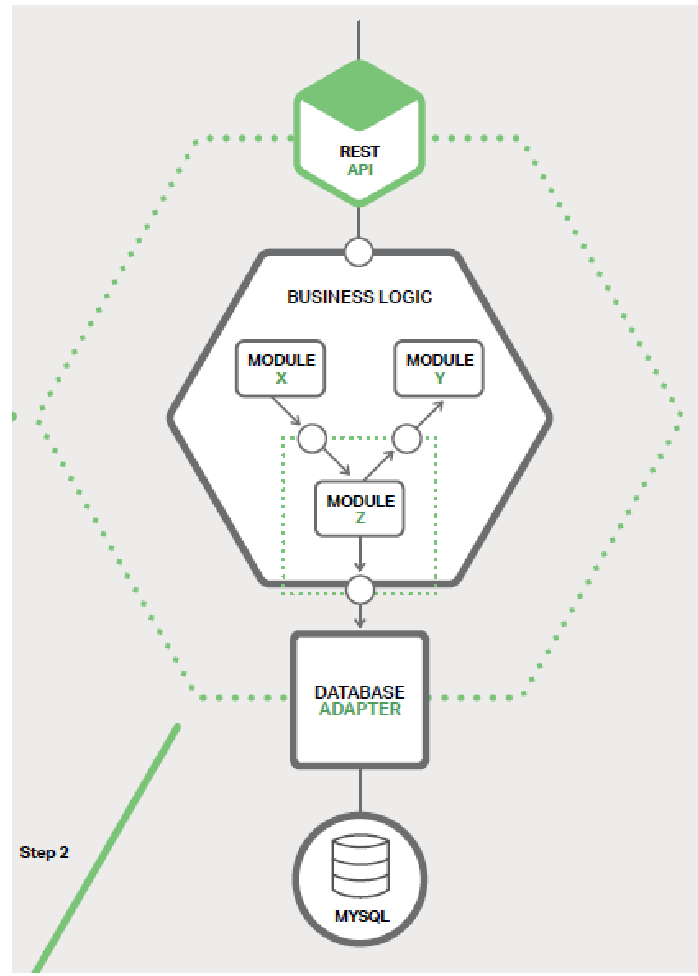
Phase 2: Extract a Module (Step 1)

- Module Z is the candidate to extract
- Its components are being used by Module X
- It uses Module Y

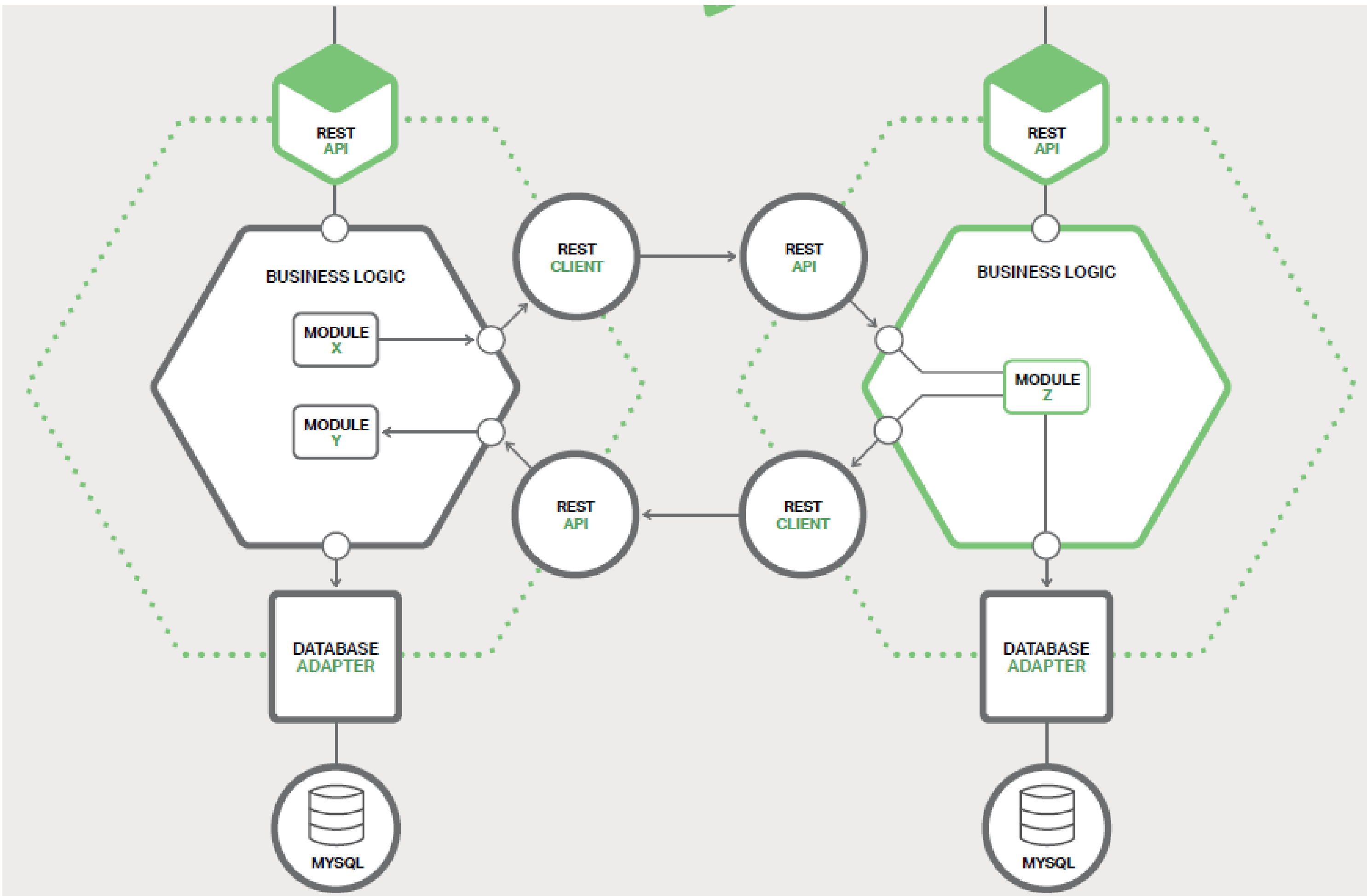


Phase 2: Extract a Module (Step 1)

- The first step defines a pair of coarse-grained APIs
- The first interface (inbound) is used by Module X to invoke Module Z
- The second interface (outbound) is used by Module Z to invoke Module Y



Phase 2: Extract a Module (Step 2)



Phase 2: Extract a Module (Step 2)

- Modifying the module in a standalone service
- Inbound and outbound interfaces are implemented by a code using an IPC mechanism
- Building the service combining Module Z with a Microservice Chassis framework
  - It manages cross-cutting concerns like service discovery

Phase 2: Extract a Module (Step 2)

- Once the module has been extracted, there is one more service to be developed, deployed, and scaled independently from the monolith and other services

- It is possible to rewrite the service from scratch
  - The API code integrating the service with the monolith becomes an anti-corruption layer transforming the two domain models

- Each time a service is being extracted, we take a step more towards microservices: *shrinking the monolith*

References

- Books:
  - Continuous Delivery - Jez Humble, Dave Farley
  - Working Effectively with Legacy Code - Michael Feathers
  - Domain Driven Design - Eric Evans
  - Your Brain at Work - David Rock
  - Refactoring Databases - Scott W Ambler & Pramod Sadalage
  - Building Microservices - Sam Newman
  - Microservices: From Design to Deployment - NGINX

- Articles/Blogs:
  - Ball of Mud: http://www.laputan.org/mud/
  - Demming - http://leanandkanban.wordpress.com/2011/07/15/demings-14-points/
  - Coding Horror: http://www.codinghorror.com/blog/2007/11/the-big-ball-of-mud-and-other-architecturaldisasters.html
  - http://devlicio.us/blogs/casey/archive/2009/05/14/commercial-suicide-integration-at-the-database-level.aspx
  - Evolutionary Architecture and Emergent Design: http://www.ibm.com/developerworks/java/library/j-eaed1/index.html
  - Microservices: http://www.infoq.com/presentations/Micro-Services and http://yobriefca.se/blog/2013/04/29/micro-service-architecture/ and http://davidmorgantini.blogspot.co.uk/2013/08/micro-services-what-are-microservices.html
  - http://martinfowler.com/articles/microservices.html
  - http://highscalability.com/blog/2014/4/8/microservices-not-a-free-lunch.html

Lesson 5.1: Big Data Analytics-as-a-Service and Microservices

Claudio Ardagna, Ernesto Damiani – Università degli Studi di Milano

Cloud and Distributed Computing

Project

HORIZON 2020

TOREADOR

Project ID: 688797
Funded under:
H2020-EU.2.1.1. - INDUSTRIAL LEADERSHIP - Leadership in enabling and industrial technologies - Information and Communication Technologies (ICT)

TrustwOrthy model-awaRE Analytics Data platfORm

From 2016-01-01 **to** 2018-12-31, ongoing project

Project details

Total cost:	**Topic(s):**
EUR 6 311 218,75	ICT-16-2015 - Big data - research
EU contribution:	**Call for proposal:**
EUR 6 311 218,75	H2020-ICT-2015 See other projects for this call
Coordinated in:	**Funding scheme:**
Italy	RIA - Research and Innovation action

Consortium

Project Coordinator
PI: Prof. Ernesto Damiani

TOREADOR Concept

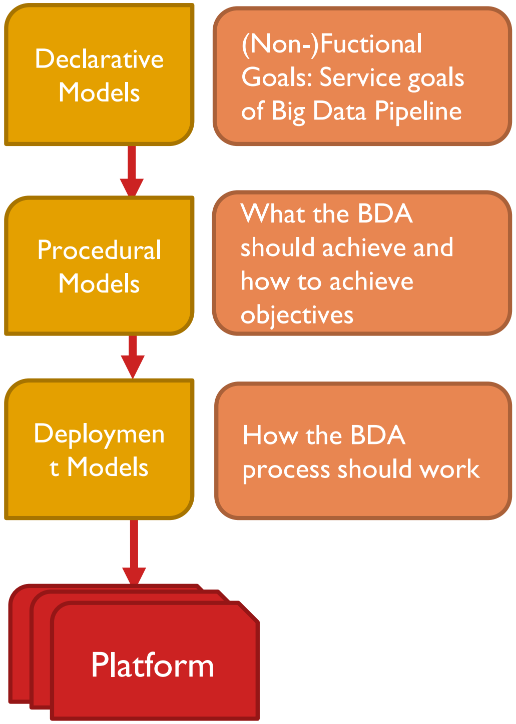
- Many SME lack the IT expertise and budget to fully exploit Big Data Analytics (BDA)
- To overcome this hurdle, TOREADOR takes a Model-based BDA-as-a-service (MBDAaaS) approach
- TOREADOR open, suitable for-standardisation models will support substantial automation and commoditization of Big Data Analytics
- Once TOREADOR MBDAaaS will become widespread, price competition on Big Data services will ensue, driving costs of Big Data analytics well within reach of EU organizations

TOREADOR Objectives

- Automation and commoditization of Big Data analytics
  - MBDAaaS provides models of the entire Big Data process and of its artefacts
  - Specification of a fully declarative framework and a model set supporting Big Data analytics
  - Enabling it to be easily tailored to domain-specific customer requirements
- SLA approaches to guarantee contractual quality, performance, and security of BDA
- Design and development of automatic deployment of TOREADOR analytic solutions

TOREADOR Overview

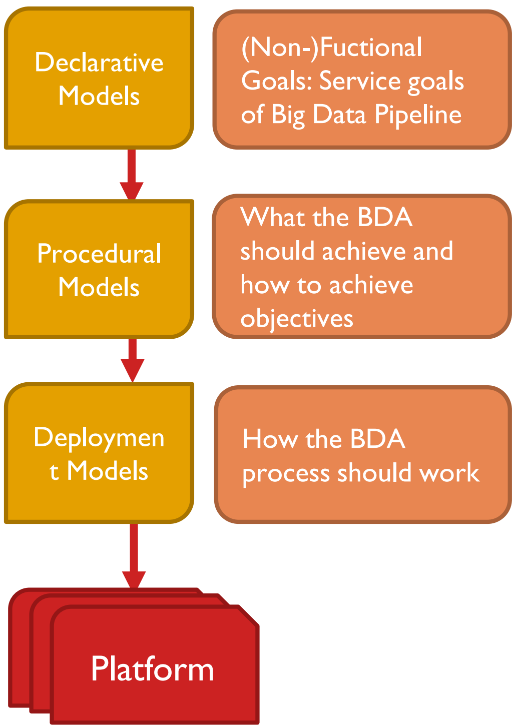
- Model-driven approach
- Abstract the typical procedural models (e.g., data pipeline) implemented in big data frameworks
- Develop model transformations to translate modelling decisions into actual provisioning



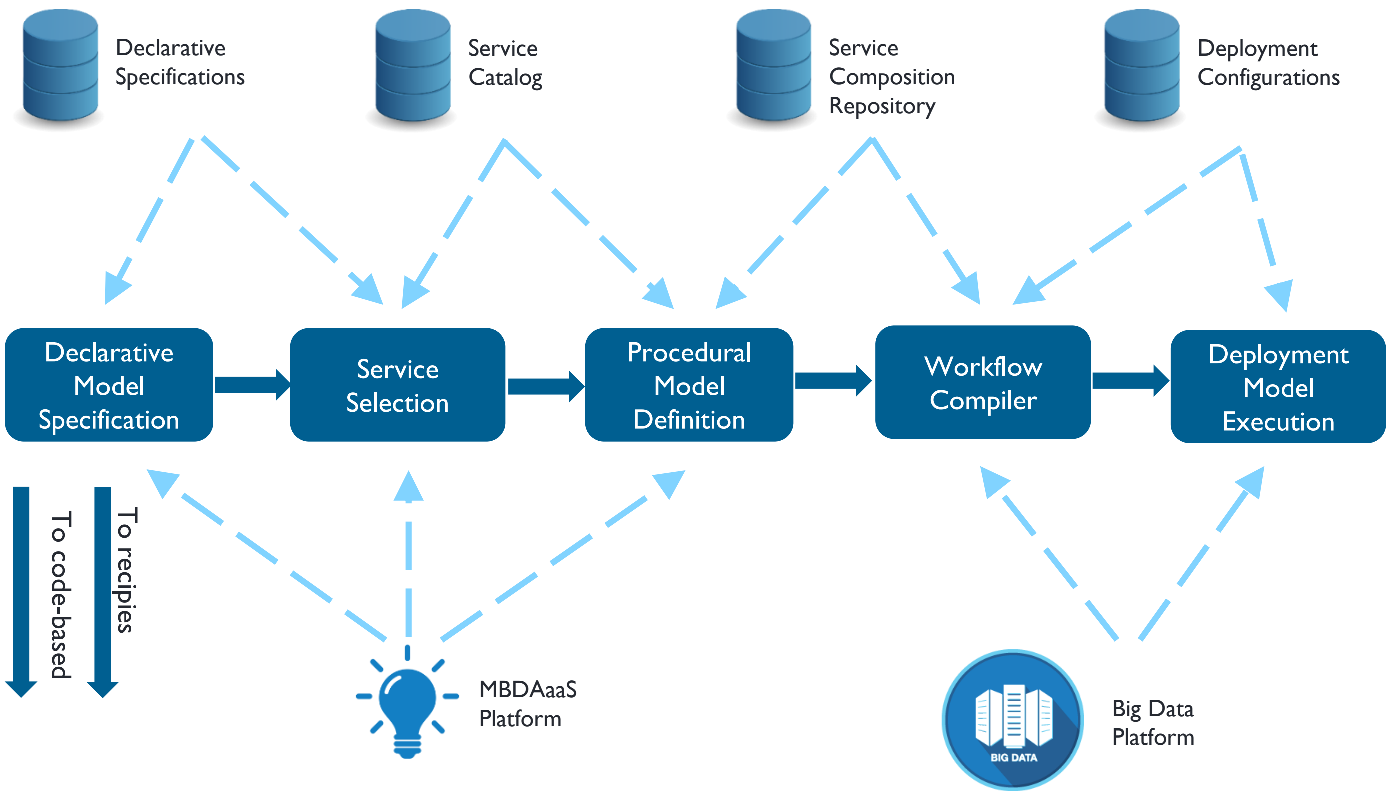
TOREADOR Overview

The Model-driven approach supports

- Usability and productivity
  - Customers lacking Big Data expertise in managing big data analytics deploying a full big data pipeline
  - Fast roll-out: efficient link between R&D and production
- Accountability and reproducibility
  - Multiple solutions can be compared
  - Clear specification of the services
  - Reuse and modularity
- Verifiability
  - Assess preconditions
  - Check consistency with requirements
- Technology neutrality
  - Multiple platforms are supported

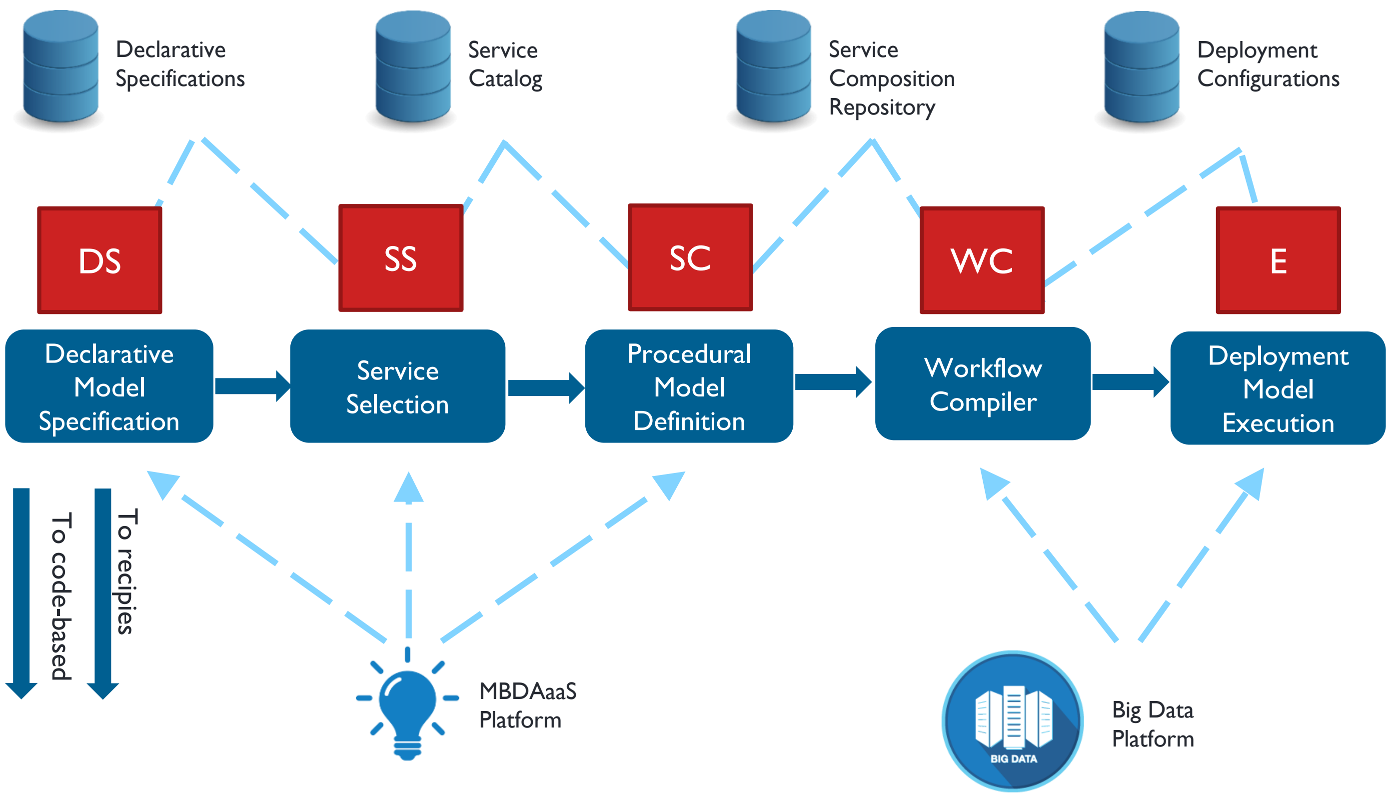


Overview of the Methodology



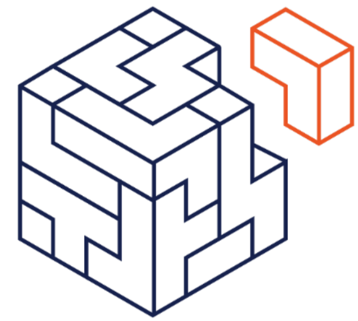
[SERVICE-BASED] C.A. Ardagna, V. Bellandi, M. Bezzi and P. Ceravolo, E. Damiani, C. Hebert, "Model-based Big Data Analytics-as-a-Service: Take Big Data to the Next Level," in *IEEE Transactions on Services Computing (TSC)*, 2018
[CODE-BASED] B. Di Martino, S. D'Angelo, A. Esposito, S. Maisto, S. Nacchia, "A Compiler for Agnostic Programming and Deployment of Big Data Analytics on Multiple Platforms", in IEEE Transactions on Parallel and Distributed Systems (TPDS), 2019 (accepted for publication)

Overview of the Methodology

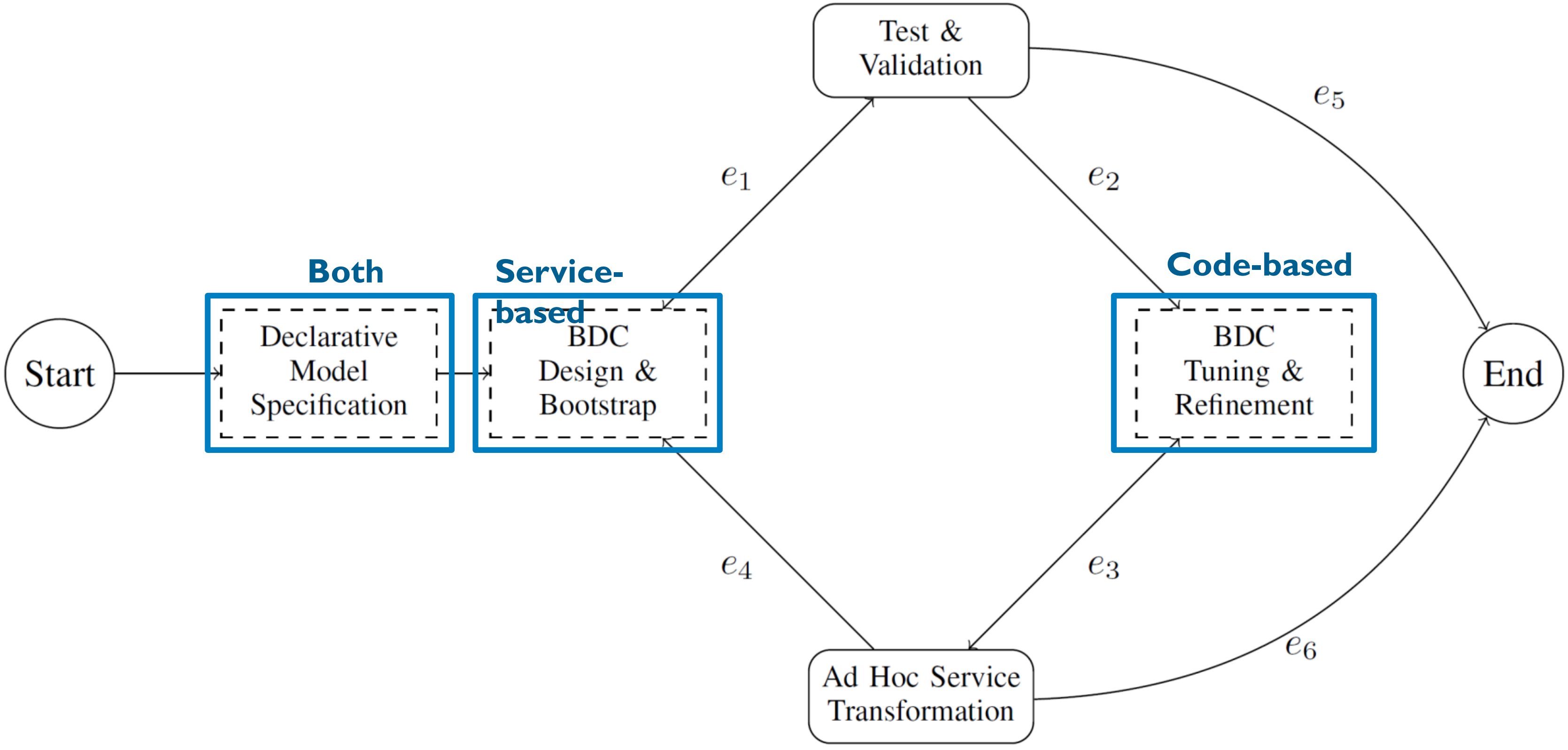


Two Methodology Lines

- Service-based
  - No coding, for basic users
  - Analytics services are provided by the target TOREADOR platform
  - Big Data campaign built by composing existing services
  - Based on model transformations
- Code-based
  - Advanced users
  - Analytics algorithms are developed by the users
  - Parallel computations are configured by the users
- Both driven by the same declarative model
  - Code once, deploy everywhere
- Support for batch and stream
- Hybridization and «docker landing» supported

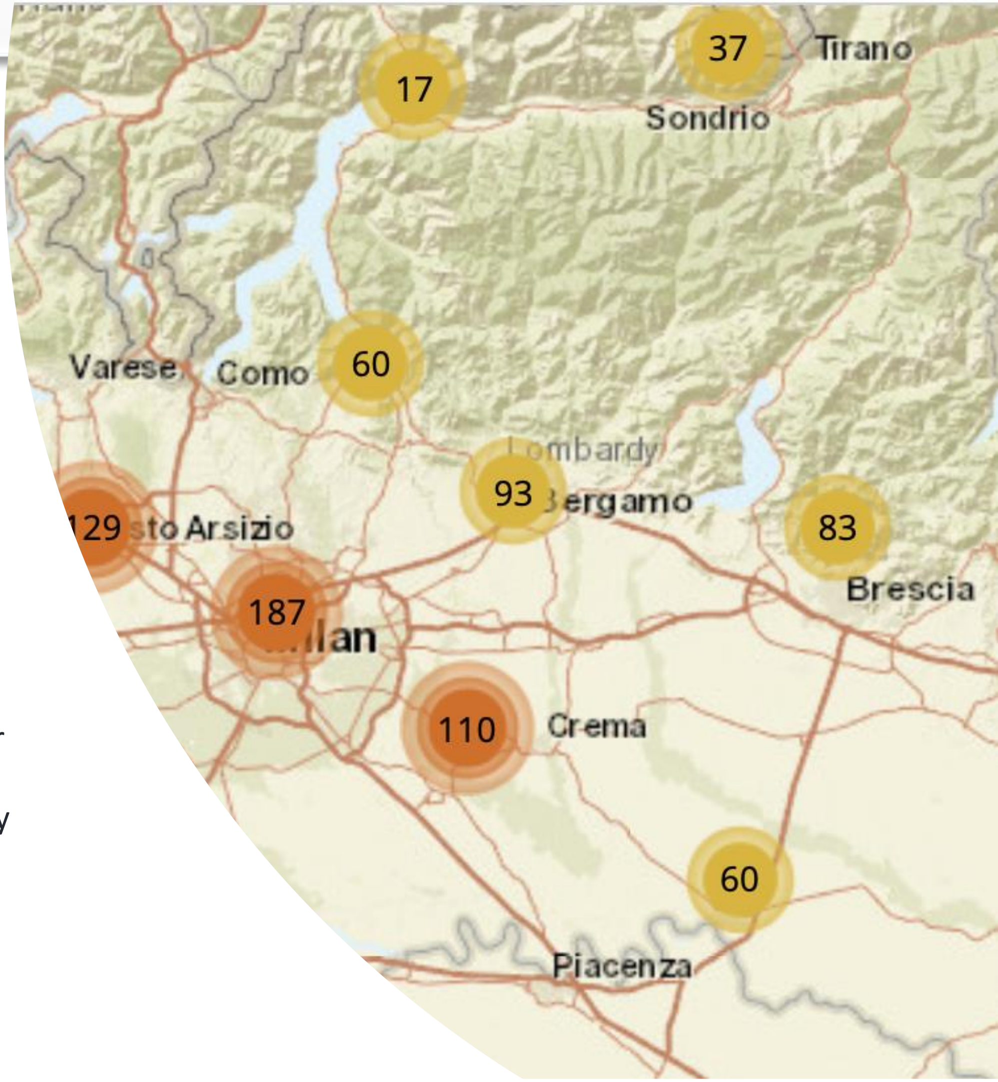


BDC Development LifeCycle Methodology



Reference Scenario

- Our reference scenario is an infrastructure for pollution monitoring managed by Lombardia Informatica, an agency of Lombardy region in Italy.
- A network of sensors acquire pollution data everyday.
  - **sensors**, containing information of a specific acquiring sensor such as ID, pollutant type, unit of measure
  - **data acquisition stations**, managing a set of sensors and containing information regarding their position (e.g. longitude/latitude)
  - **pollution values**, containing the values acquired by sensors, the timestamp, and the validation status. Each value is validated by a human operator that manually labels it as **valid** or **invalid**.



Reference Scenario

- The goal is to deploy a DSS using our methodology to reduce the time required for validating data
- The DSS must
  - predict the labels of acquired data in real time
  - alert the operator when anomalous values are observed

6276	22/01/2019 09:00:00 AM	1230,9	✓
6276	22/01/2019 10:00:00 AM	1167,9	✓
6328	04/02/2019 09:00:00 AM	1017,6	✓
30162	16/01/2019 09:00:00 AM	1005,4	✓
6328	15/01/2019 09:00:00 PM	982,6	✓
6276	21/01/2019 09:00:00 AM	949,4	✓
6356	11/01/2019 10:00:00 AM	934,3	✓
10025	01/01/2019 01:00:00 AM	-9999	✗
11041	01/01/2019 03:00:00 AM	-9999	✗
11041	01/01/2019 12:00:00 AM	-9999	✗
10023	01/01/2019 04:00:00 AM	-9999	✗
10023	01/01/2019 03:00:00 AM	-9999	✗
10025	01/01/2019 04:00:00 AM	-9999	✗
11041	01/01/2019 04:00:00 AM	-9999	✗

Declarative Model Definition

Declarative Models: vocabulary

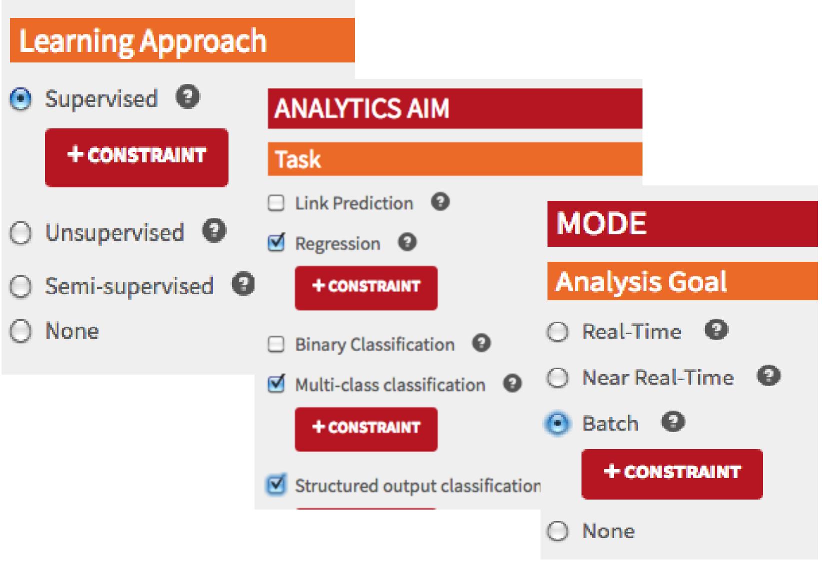
- Declarative model offers a vocabulary for an computation independent description of BDA

- Organized in 5 areas
  - Representation (Data Mode, Data Type, Management, Partitioning)
  - Preparation (Data Reduction, Expansion, Cleaning, Anonymization)
  - Analytics (Analytics Model, Task, Learning Approach, Expected Quality)
  - Processing (Analysis Goal, Interaction, Performances)
  - Visualization and Reporting (Goal, Interaction, Data Dimensionality)

- Each specification can be structured in three levels:
  - Goal: Indicator – Objective – Constraint
  - Feature: Type – Sub Type – Sub Sub Type

Declarative Models

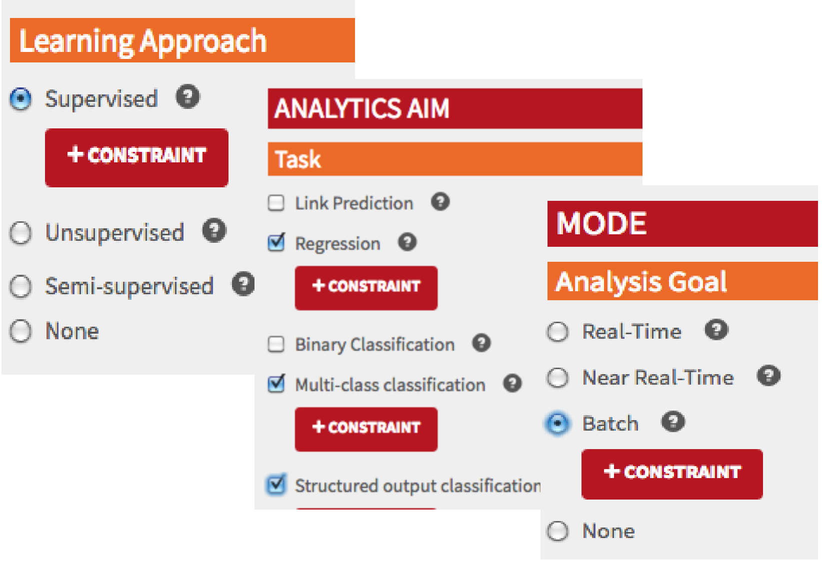
- A web-based GUI for specifying the requirements of a BDA



  - No coding, for basic users
  - Analytics services are provided by the target TOREADOR platform
  - Big Data campaign built by composing existing services
  - Based on model transformations

Declarative Models

- A web-based GUI for specifying the requirements of a BDA
  - Data_Preparation.Data_Source_Model.Data_Model.Document_Oriented
  - Data_Analytics.Analytics_Aim.Task.Crisp_Clustering



Declarative Model for Reference Scenario

- The solution require to processing stages training step and a prediction step
- Our DM includes two requirement specifications
  - DataPreparation.DataTransformation.Filtering;
  - DataAnalitycs.LearningApproach.Supervised;
  - DataAnalitycs.LearningStep.Training;
  - DataAnalitycs.AnalyticsAim.Regression;
  - DataProcessing.AnalyticsGoal.Batch.

DS1

  - DataAnalitycs.LearningApproach.Supervised;
  - DataAnalitycs.LearningStep.Prediction;
  - DataAnalitycs.AnalyticsAim.Regression;
  - DataProcessing.AnalyticsGoal.Streaming.

DS2

Checking Consistency of Declarative Models

- Interference Declarations
  - Boolean Interference: $P \rightarrow \neg Q$
  - Intensity of an Interference: $DP \cap DQ$

- Interference Enforcement
  - The interference enforcement process is modeled as a function that takes as input an interference and produce as output a rule $r \in R$ that, applied to the specification with lower priority, resolves the interference

Interference Declaration

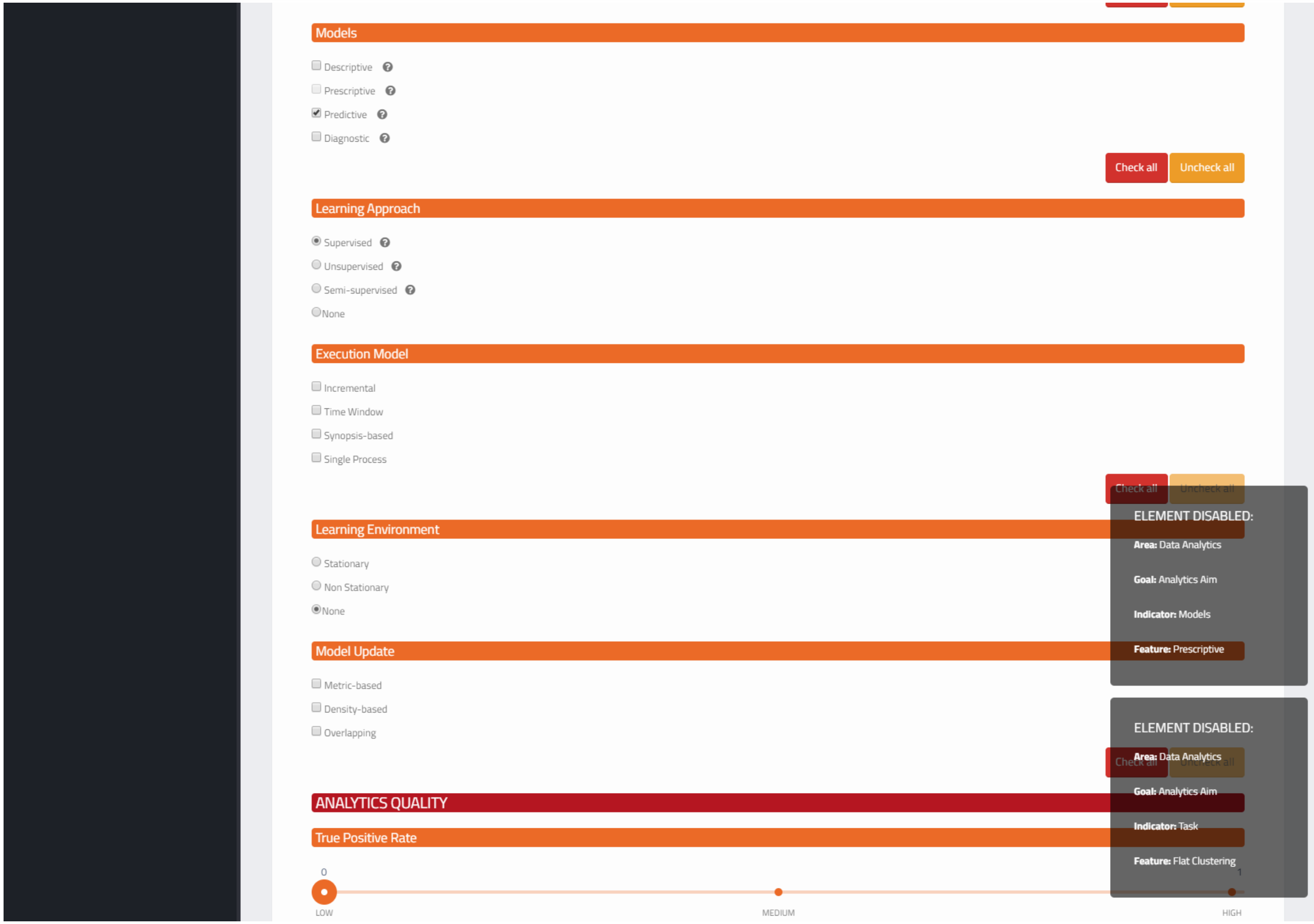
- A few examples

  - Data_Preparation.Anonymization. Technique.k-anonymity
  →¬ Data_Analitycs.Analitycs_Quality. False_Positive_Rate.low

  - Data_Preparation.Anonymization. Technique.hashing
  →¬ Data_Analitycs.Analitycs_Aim. Task.Crisp_Clustering.algorithm=k-mean

  - Data_Representation.Storage_Property. Coherence_Model.Strong_Consistency
  →¬ Data_Representation.Storage_Property. Partitioning

Consistency Check

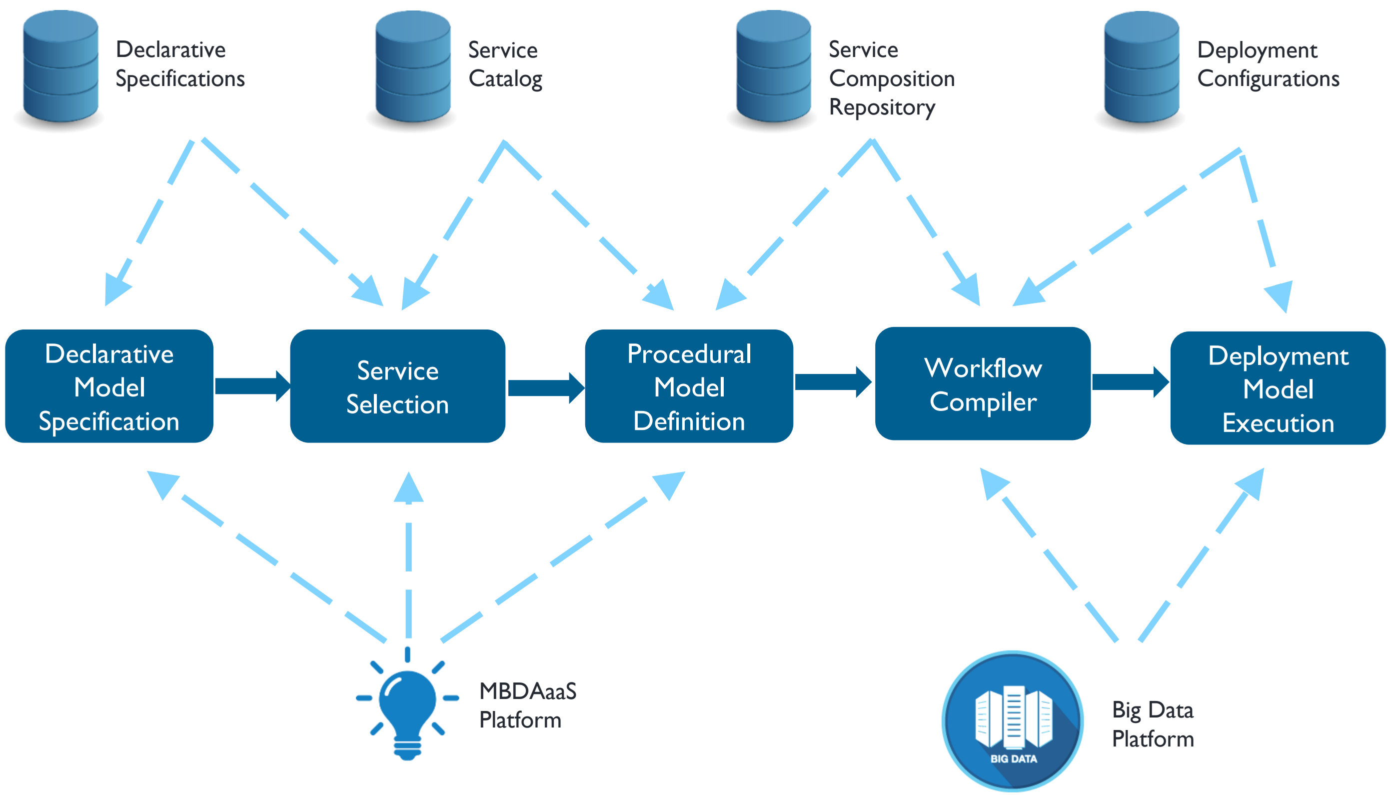


Service-Based Line

Methodology: Building Blocks

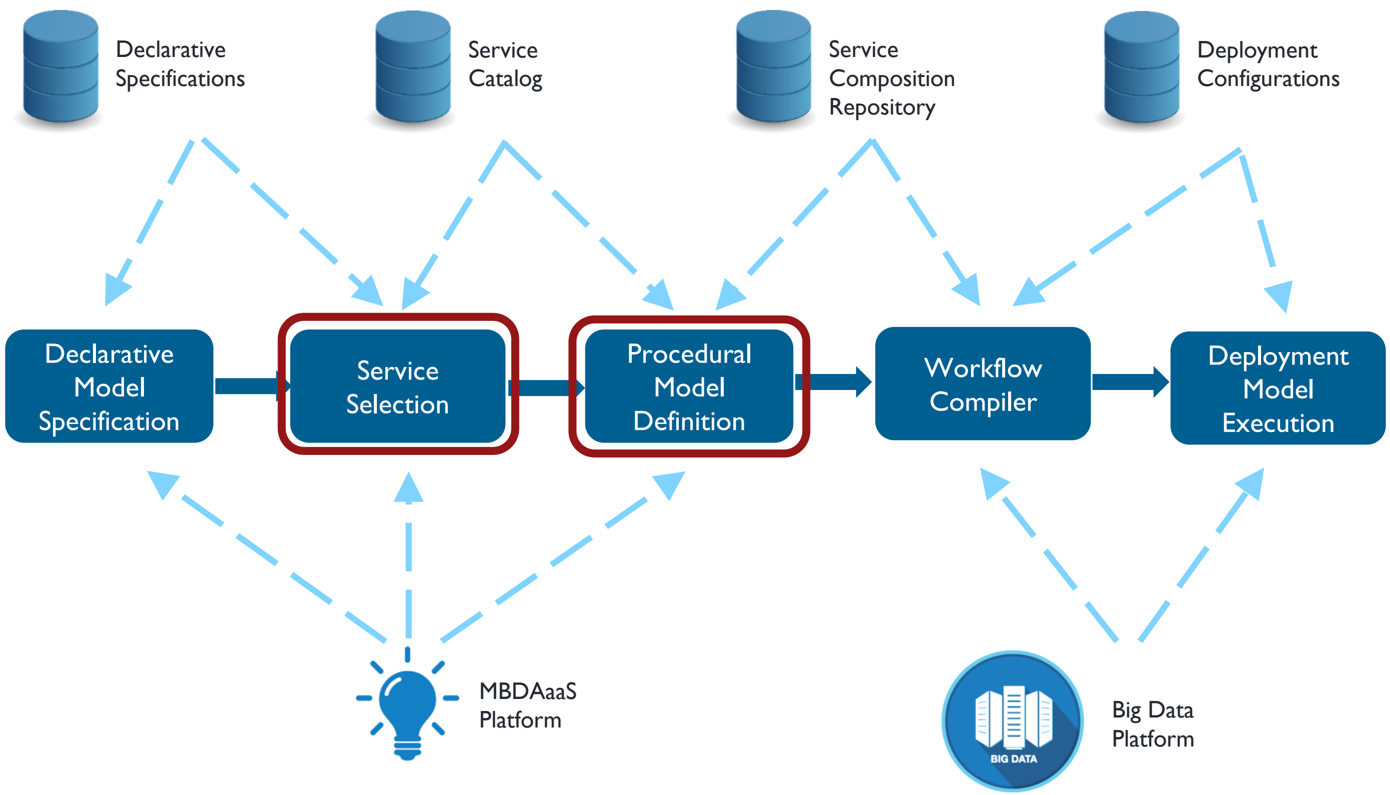
- Declarative Specifications allow customers to define declarative models shaping a BDA and retrieve a set of compatible services
- Service Catalog specifies the set of abstract services (e.g., algorithms, mechanisms, or components) that are available to Big Data customers and consultants for building their BDA
- Service Composition Repository permits to specify the procedural model defining how services can be composed to carry out the Big Data analytics
  - Support specification of an abstract Big Data service composition
- Deployment Configurations define the platform-dependent version of a procedural model, as a workflow that is ready to be executed on the target Big Data platform

Overview of the Methodology



Procedural Model Definition

Overview of the Methodology

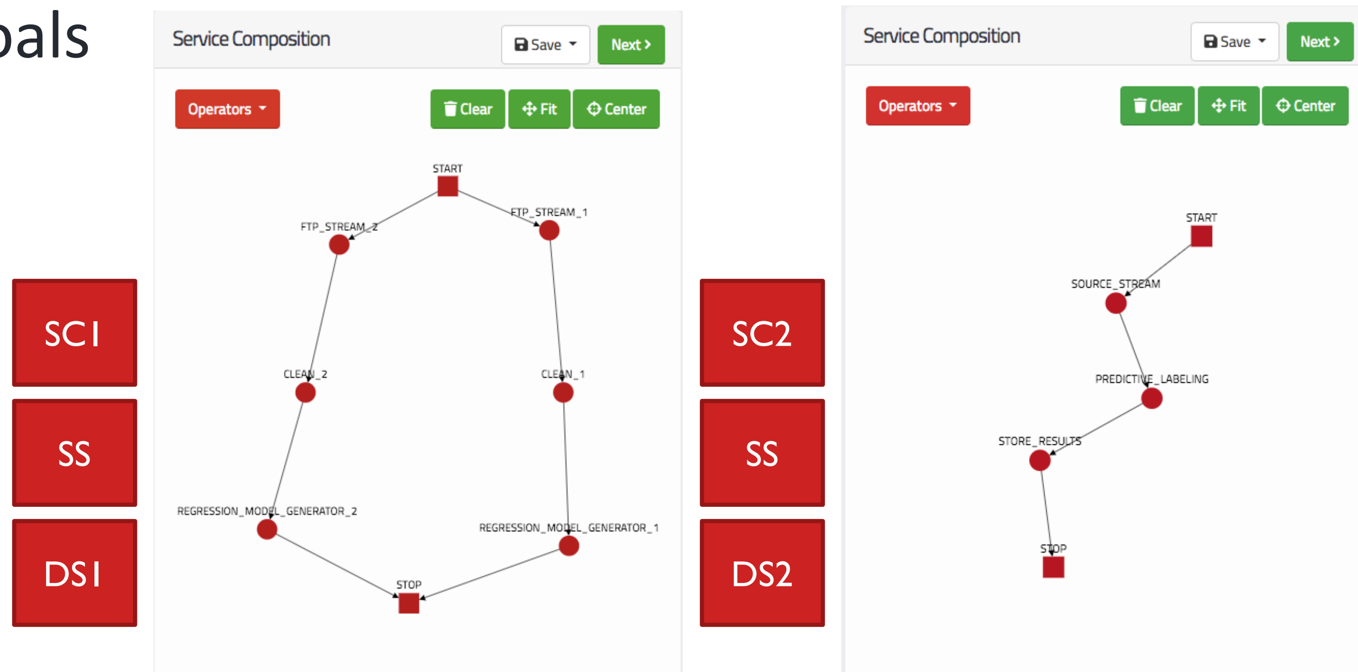


Procedural Models

- Platform-independent models that formally and unambiguously describe how analytics should be configured and executed
  - They are generated following goals and constraints specified in the declarative models
  - They provide a workflow in the form of a service orchestration
    - Sequence
    - Choice
    - If-then
    - Do-While
    - Split-Join

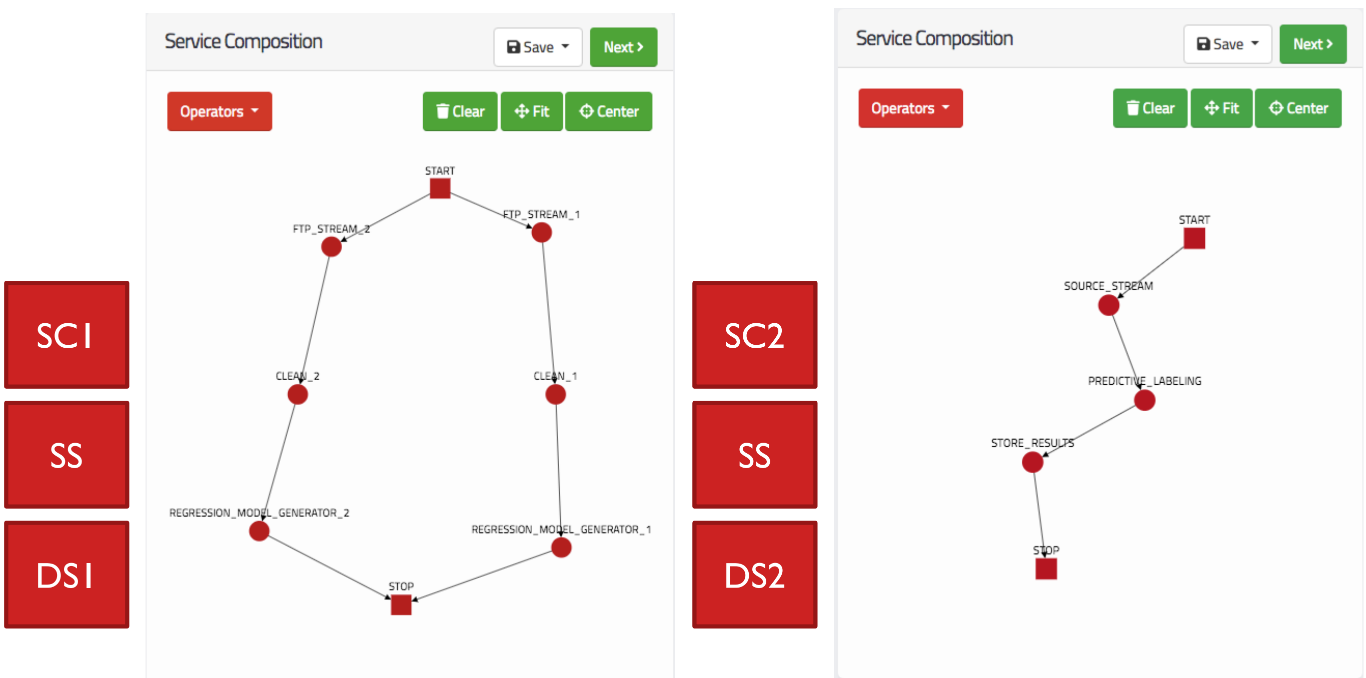
Procedural Model

- TOREADOR (**SS**) will return a set of services consistent with **DS1** and **DS2**
- The user can **compose** these services to address the scenario goals



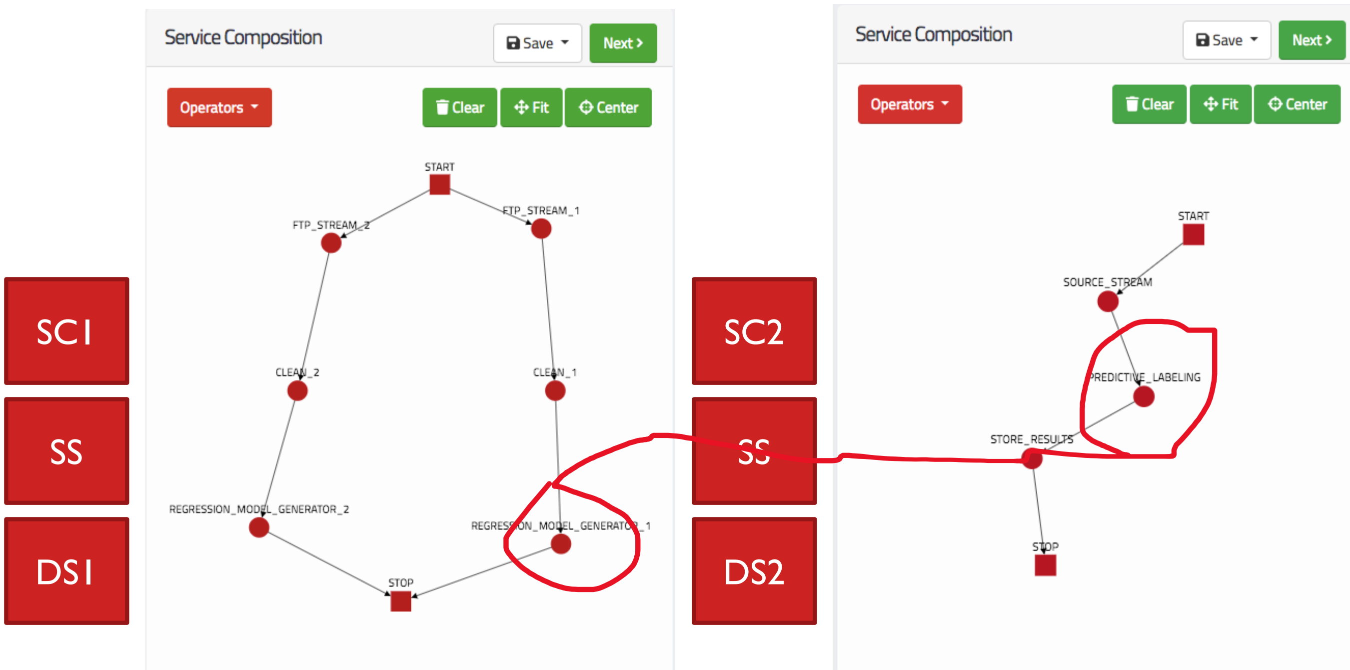
Procedural Model

- The two compositions must be connected: the output of **SC1** is a pre-requisite for **SC2**



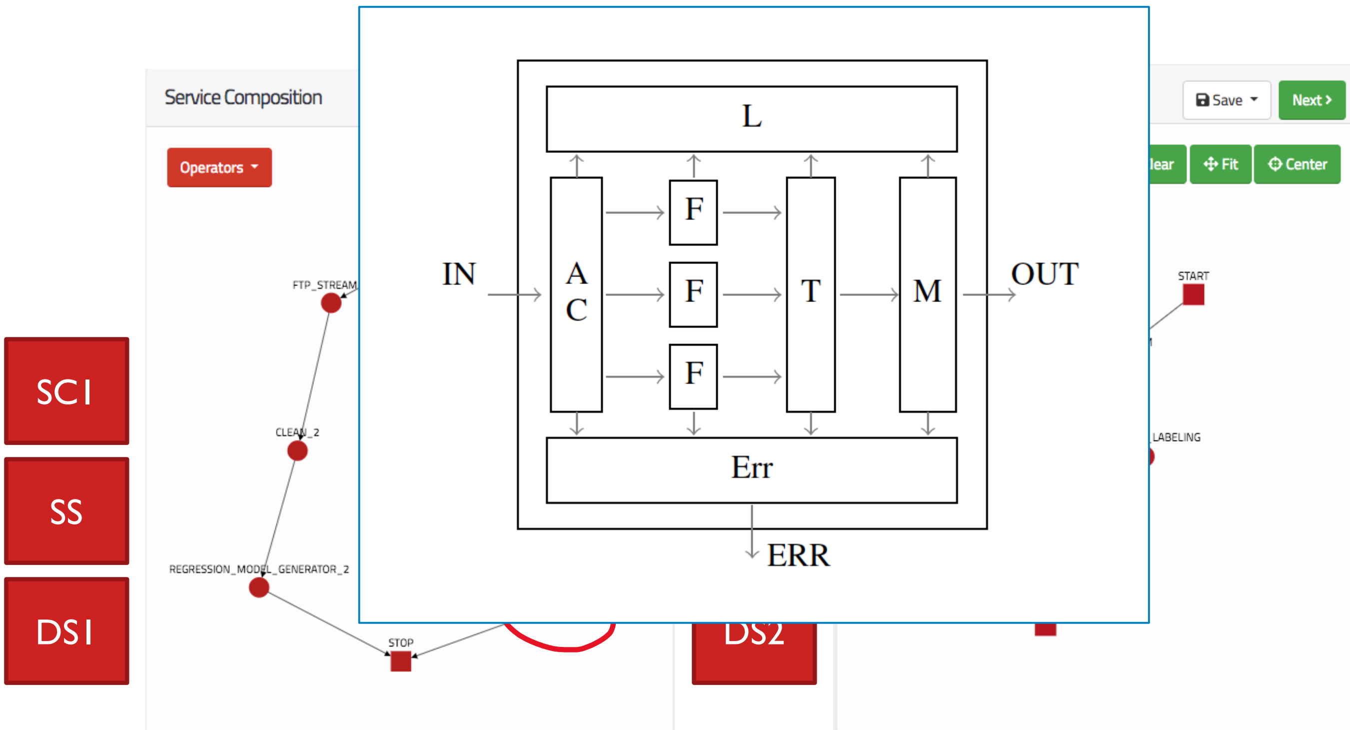
Procedural Model

- The two compositions must be connected: the output of **SC1** is a pre-requisite for **SC2**



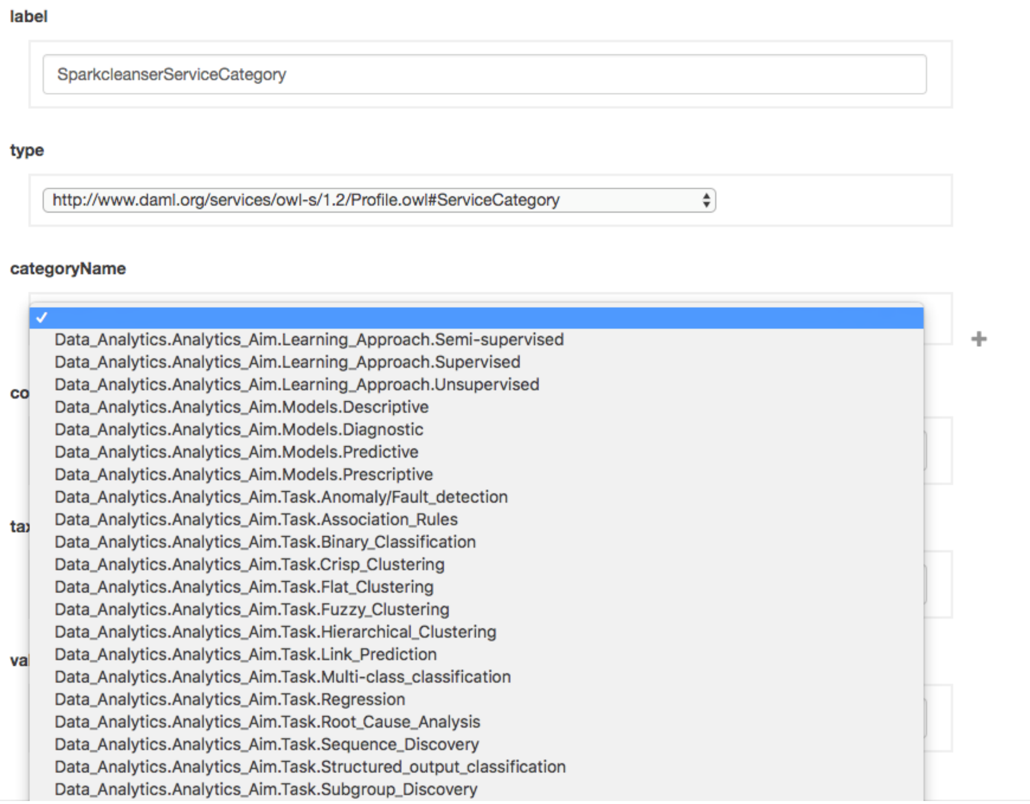
Procedural Model

- The two compositions must be connected as the out put of **SC1** is a pre requirement for **SC2**



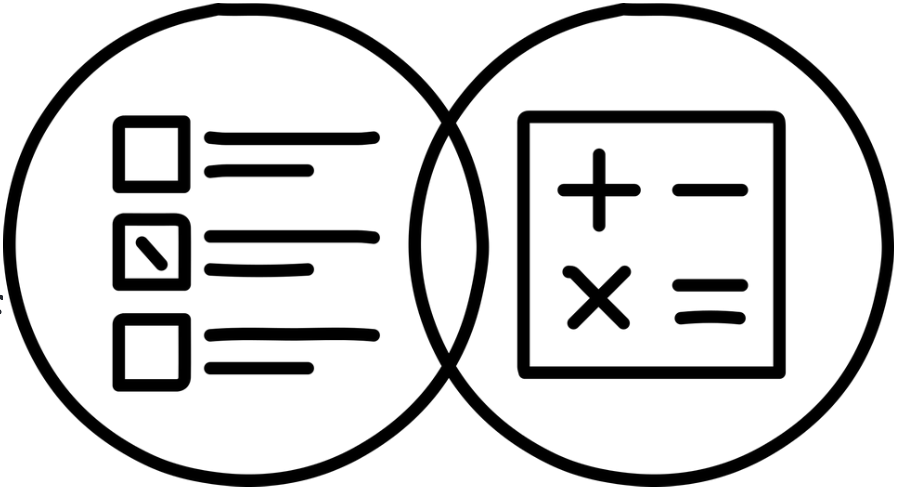
Annotations on procedural models

- Services and Computational Models at Procedural level are annotated and indexed



Service Selection

1. Receives a list of declarative models
2. Extract areas and categories from declarative models
3. Identify services compatible with the extracted areas and categories
4. For each area return a list of compatible services



We support Boolean formulas of specifications in a Disjunctive Normal Form

Service Selection

The aim is to identify specifications that are effective in discriminating services

Data_Representation.Data_Source_Property.Data_Managment.Data_Stream

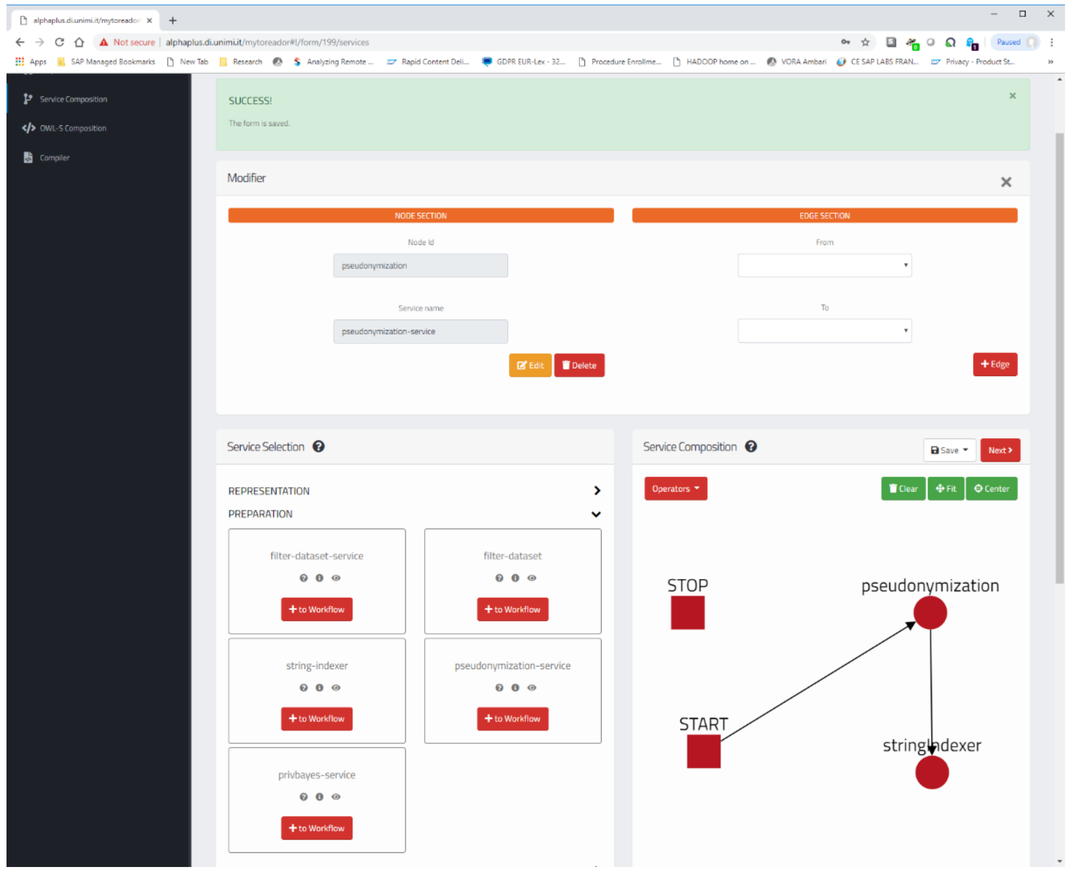
AND

Data_Analytics.Analytics_Aims.Task.Crisp_Clustering

The only services consistent with these two constraints are micro clustering algorithms

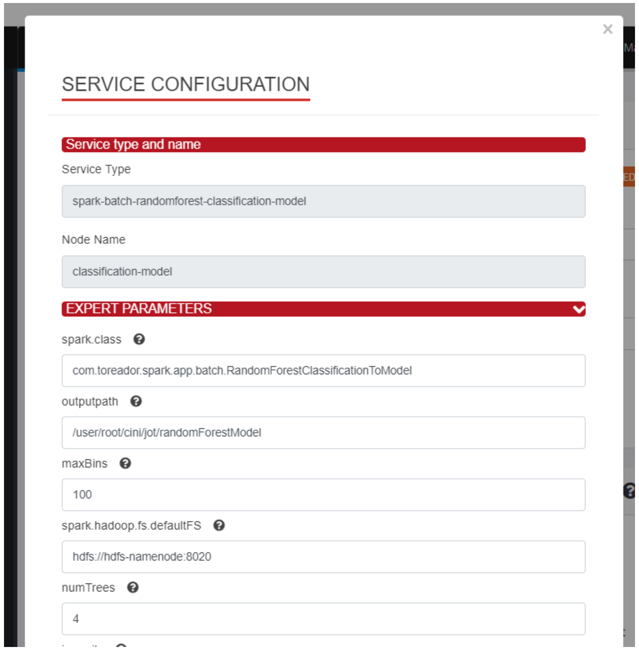
Service Composition

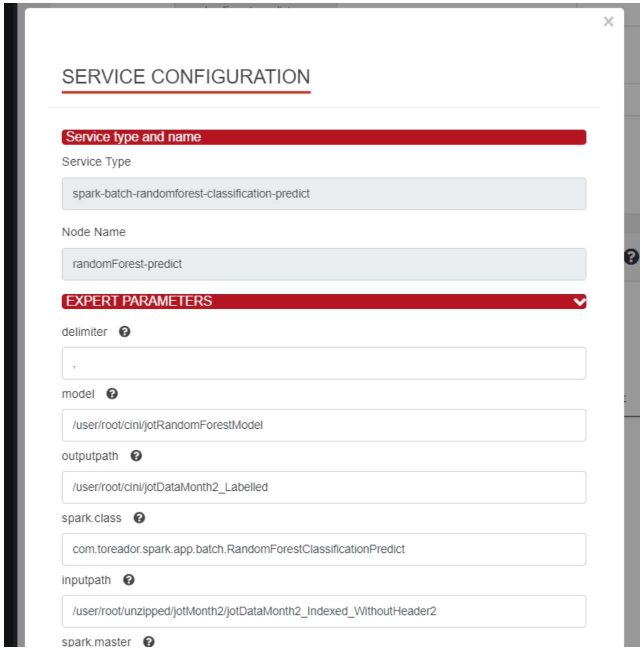
- User creates the flow based on the list of returned services



Service Composition

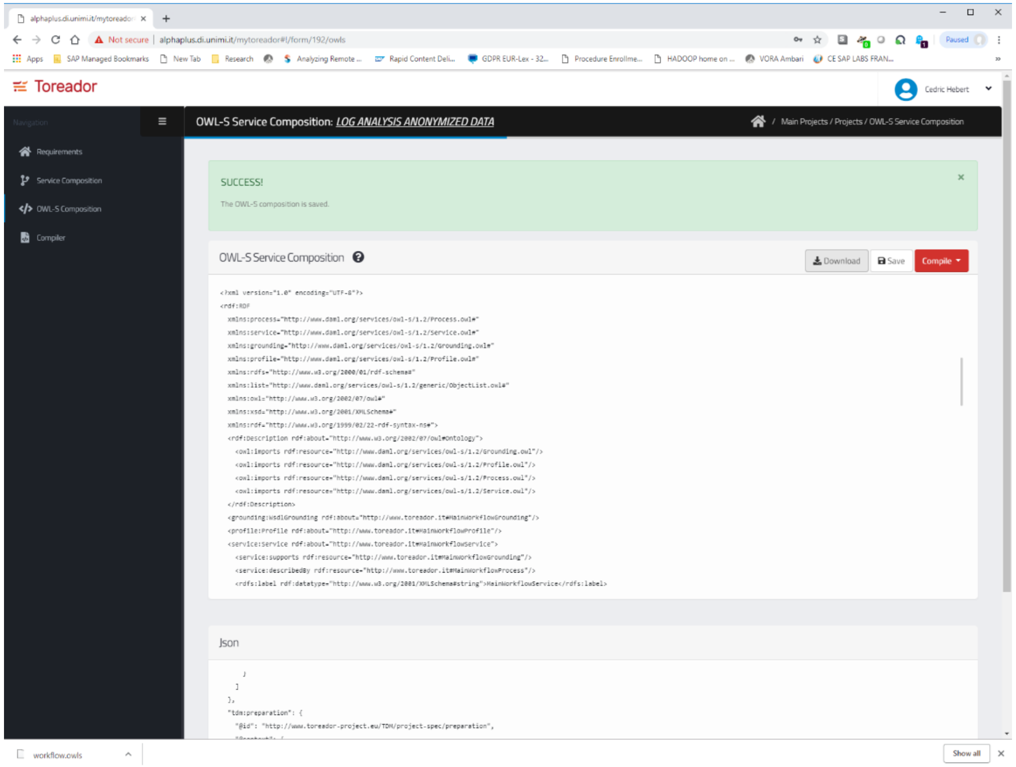
- User creates the flow based on the list of returned services
- Services enriched with ad hoc parameters





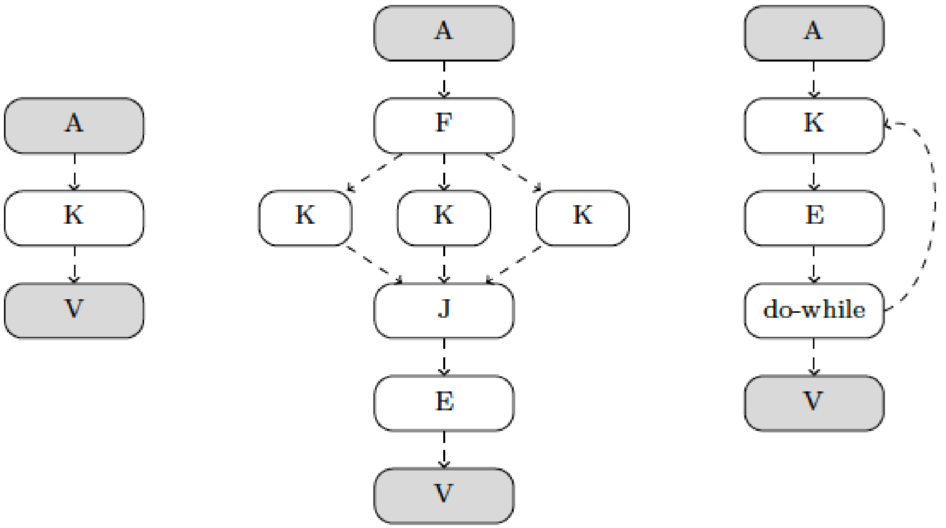
Service Composition

- User creates the flow based on the list of returned services
- Services enriched with ad hoc parameters
- The flow is submitted to the service which translates it into service composition



Service Composition

- All internals are made explicits
  - Clear specification of the services
  - Reuse and modularity



(a) (b) (c)

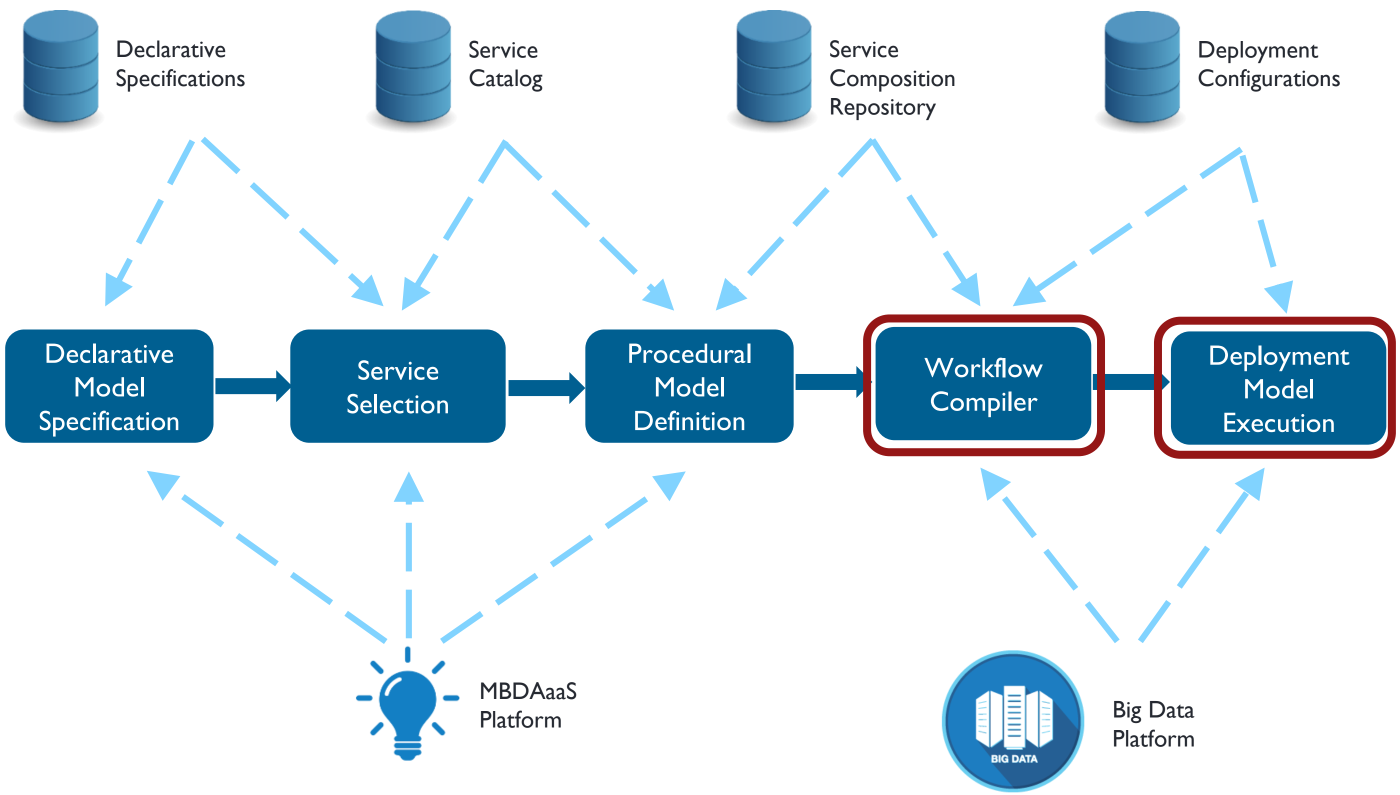
Managing Multiple Declarations

- Some tasks require as a precondition the execution of other specific tasks which may or may not be deployed on the same platform
  - E.g. to run a classifier (scorer/regression) we need to have available a classification model generated by a training/test task

- TOREADOR made available two solutions
  - Data as connectors
    - E.g. the model is the data connector
  - Saving a workflow as a new service in the catalog
    - E.g. the service for the generation of the classification model

Deployment Model Definition

Overview of the Methodology



Deployment Model

- The TOREADOR **compiler**
  - Takes **SC1** and **SC2** to produce two executable workflows
  - Supports different engines and languages: **Spring Cloud DataFlow** and **Oozie**
  - Can be extended to any engines simply defining the **proper driver**

WC1

SC1

SS

DS1

```
spark−filter−sensorsTest : filter
−−expr=”sensorsDF#SensorId === 5958” −− i n p u t P a
t h = ” / u s e r / r o o t / s e n s o r s / j o i n e d . c s v ”
−−outputPath=”/user/root/sensors test.csv” &&
spark−assemblerTest : spark−assembler
−−features=”Data,Quote”−−inputPath=”/user/root/sen
sors test.csv”
−−outputPath=”/user/root/sensors/sensors test
assembled.csv” &&
spark−gbt−predict :
batch−gradientboostedtree−classification−predict
−−inputPath =/ user / root / sensors / sensors
−−outputPath =/ user / root / sensors / sensors −− m o
d e l = / u s e r / r o o t / s e n s o r s / m o d e l
```

WC2
1-n

SC2

SS

DS2

```
spark−filter−sensorsTest : filter
−−expr=”sensorsDF#SensorId === 5958” −− i n p u t P a
t h = ” / u s e r / r o o t / s e n s o r s / j o i n e d . c s v ”
−−outputPath=”/user/root/sensors test.csv” &&
spark−assemblerTest : spark−assembler
−−features=”Data,Quote”−−inputPath=”/user/root/sen
sors test.csv”
−−outputPath=”/user/root/sensors/sensors test
assembled.csv” &&
spark−gbt−predict :
batch−gradientboostedtree−classification−predict
−−inputPath =/ user / root / sensors / sensors
−−outputPath =/ user / root / sensors / sensors −− m o
d e l = / u s e r / r o o t / s e n s o r s / m o d e l
```

Workflow compiler

- It consists of two main sub-processes
  - Structure generation: the compiler parses the procedural model and identifies the process operators (sequence, alternative, parallel, loop) composing it
  - Service configuration: for each service in the procedural model the corresponding one is identified and inserted in the deployment model
- Support transformations to any orchestration engine available as a service
  - Available for Oozie and Spring Cloud DataFlow

Deployment Model

- Workflow compiler takes as input
  - the OWL-S service composition
  - information on the target platform (e.g., installed services/algorithms),
- It produces as output an executable workflow

- For example an Oozie workflow
  - XML file of the workflow
  - job.properties
  - System variables

Translating the Composition Structure

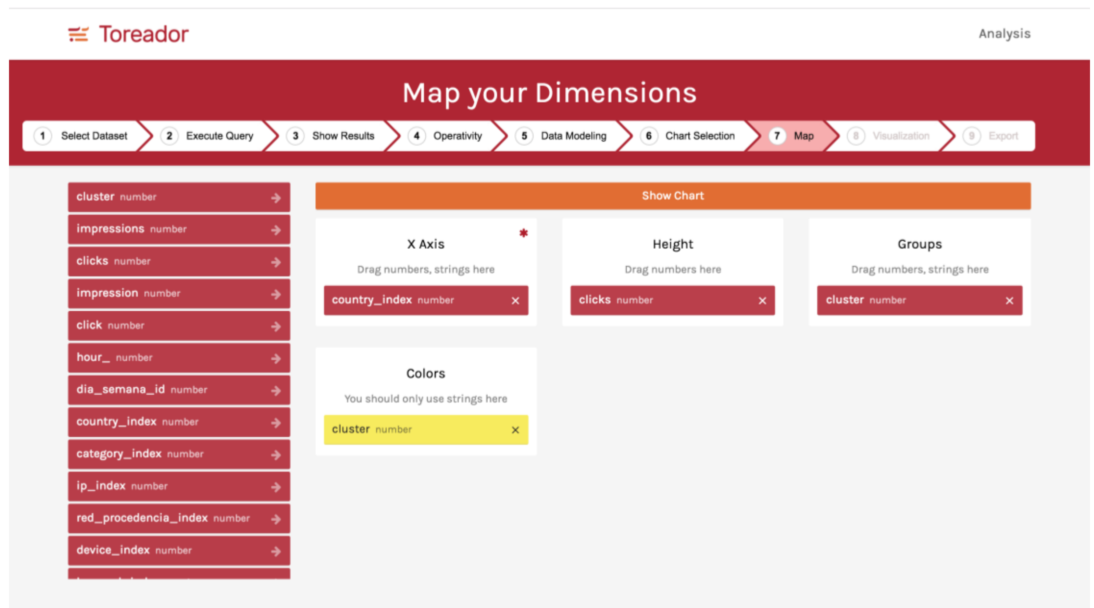
- Deployment models:
  - specify how procedural models are instantiated and configured on a target platform
  - drive analytics execution in real scenario
  - are platform-dependent

- Workflow compiler transforms the procedural model in a deployment model that can be directly executed on the target platform.

- This transformation is based on a compiler that takes as input
  - the OWL-S service composition
  - information on the target platform (e.g., installed services/algorithms),
- and produces as output a technology-dependent workflow

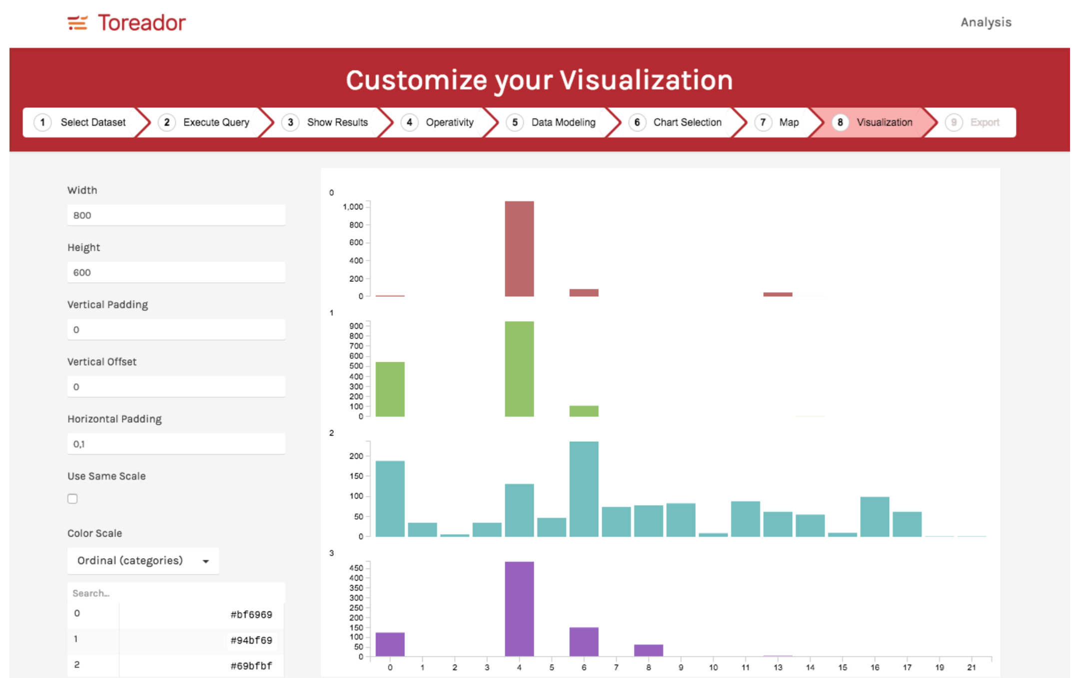
Deployment Model

- The execution **E2** will produce results



Deployment Model

- The execution **E2** will produce results



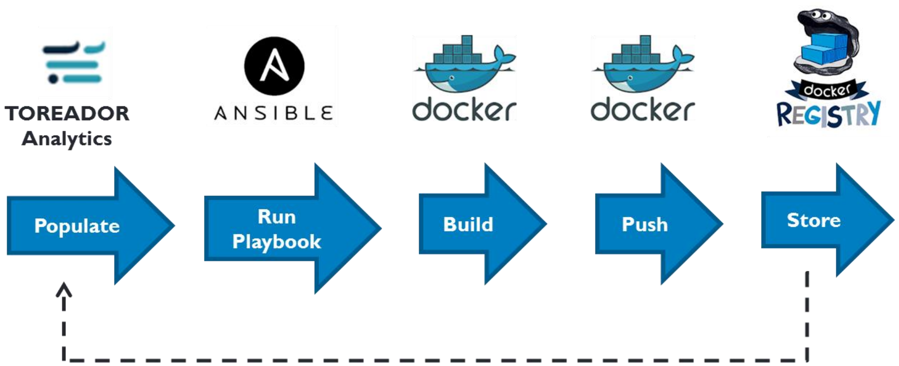
Experimental Results: Performance

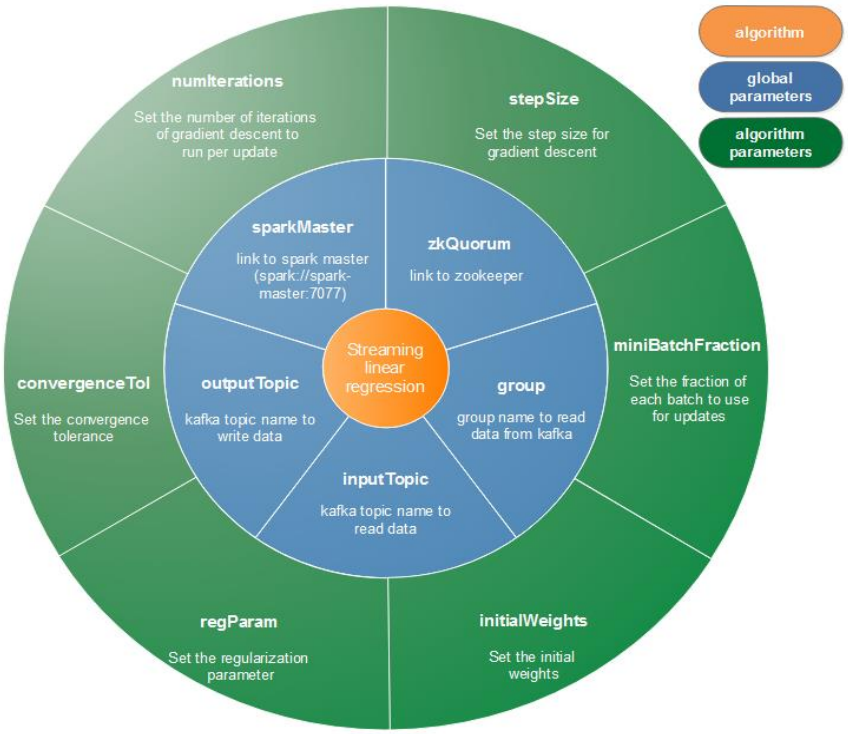
- Overhead
  - Time needed to generate the OWL-S version of our procedural model (OWL-S Generation) and the executable model (Compiler Execution).
  - Compiler Execution measures the time needed to produce the XML for Oozie workflow engine (Training Phase) and the DSL for Spring Cloud Data Flow (Prediction Phase)
- TOREADOR vs development cost
  - Development cost estimated using the Constructive Cost Model (COCOMO) methodology
  - Development cost as a function of the program size and a set of “cost drivers” that include subjective assessment of the products, hardware, personnel, and project attributes

Project Phase	OWL-S Generation	Compiler Execution	Total Costs
Training	$4.5s$	$8.7s$	$13.2s$
Prediction	$1.3s$	$3.5s$	6.8
Global	$5.8s$	$12.2s$	20s

Project Phase	Generation Time Cost	SLOC	Duration, (in months)
Training	$13.2s$	857	3.55
Prediction	$6.8s$	427	2.72
Global	$20s$	1284	4.19

Analytics Deployment Approach





Conclusions

- **A new development life cycle for Big Data**, conciliating exploration and refinement, fast deployment and controlled execution

- A methodology based on an **iterative sequence** of two phases
  - **Design and bootstrap** based on the Model-based Big Data as-a-Service (MBDAaaS) paradigm
  - **Tuning and refinement** based on a model-driven, code-based approach

Key Takeaways

- Batch and stream computations
  - Our methodology guide the user in selecting consistent set of services for both batch and stream computations
- Multiple platforms
  - Our methodology implements a smart compiler supporting the deployment of interconnected computations residing on different platforms
- End-to-end verifiability
  - Our methodology provides an end-to-end procedure for checking the consistency of model specifications
- Model reuse and refinement
  - Our methodology supports model reuse and refinement
  - Declarative, procedural and deployment models can be stored in templates to replicate or extend designed computations