Middleware for cloud computing

Agenda

- Cloud computing
- Programming for the Cloud
- Cloud computing middleware
- Middleware as a service: SOA vs SaaS

What is the Cloud?

- **Cloud** is a kind of **computing commodity** over the Internet
  - *Cloud platform*: a collection of integrated hardware, software, and network infrastructure
  - *Renting* to customers virtual hardware, software or networking services
- The platforms (*clouds*) hide the complexity and the details of the underlying physical infrastructure by providing a simple graphical interface and/or some API (Applications Programming Interface)

Cloud

- A cloud platform provides **services** that are always on, and on demand can be exploited anywhere, anytime
- Pay for use and elastic (= as needed)
  - scale up and down in capacity and functionalities
- The services are available to generic users, enterprises, corporations, or businesses markets
  - Thus, Cloud Computing can be considered a step on from Service Oriented Computing

Top ten cloud service providers 2024

#	Cloud Service Provider	Regions	Availability Zones
1	Amazon Web Services (AWS)	33	105
2	Microsoft Azure	64	126
3	Google Cloud Platform (GCP)	40	121
4	Alibaba Cloud	30	89
5	Oracle Cloud	48	58
6	IBM Cloud	10	30
7	Tencent Cloud	21	65
8	OVHcloud	17	37
9	DigitalOcean	9	15
10	Linode (Akamai)	20	20

https://dgtlinfra.com/top-cloud-service-providers/

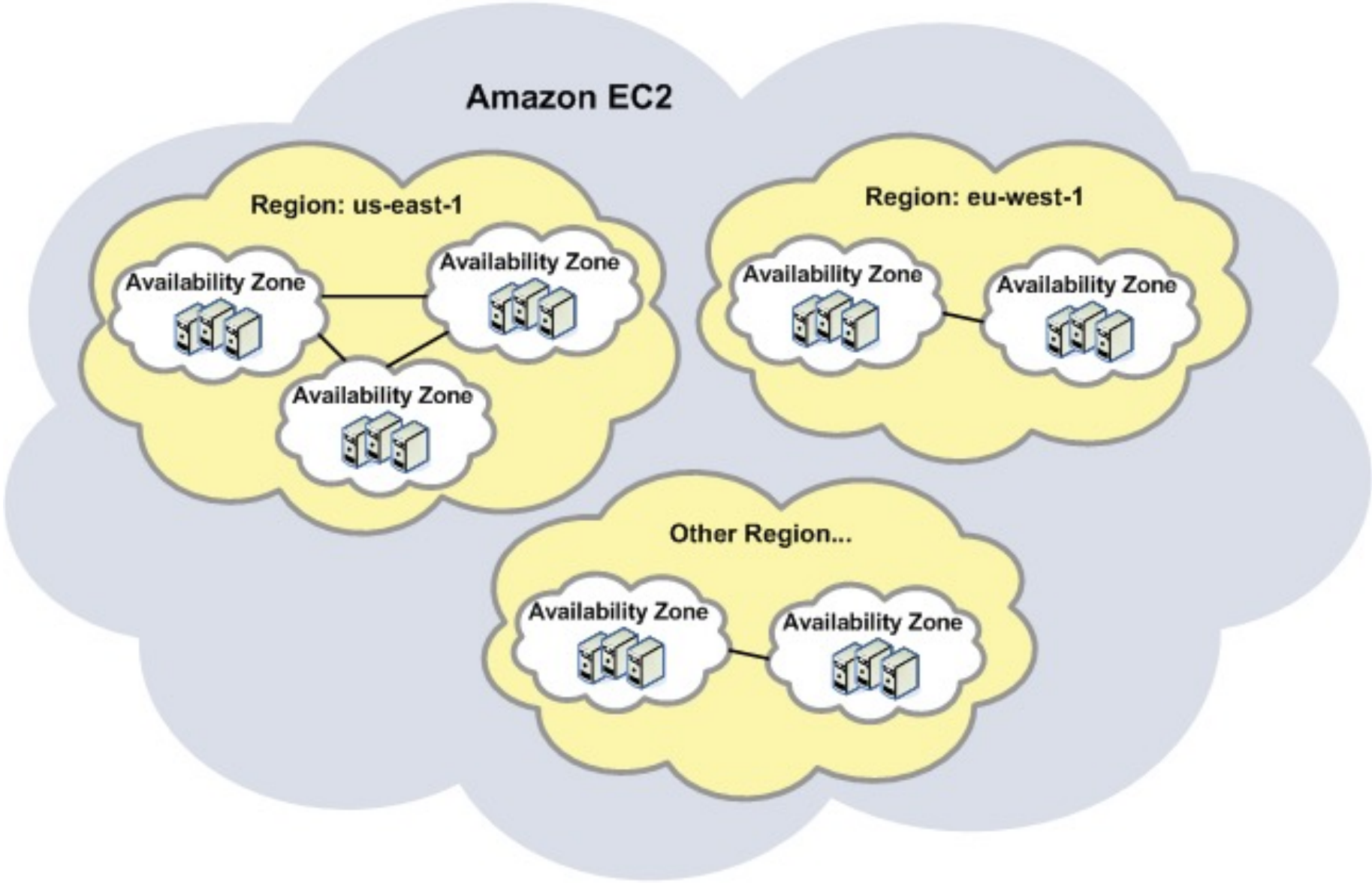
Example: AWS

- Amazon Web Services (AWS) is the largest cloud service provider. From its data centers, AWS provides over 200 services including computing, storage, and database.
- AWS currently has 33 regions and 105 availability zones in operation

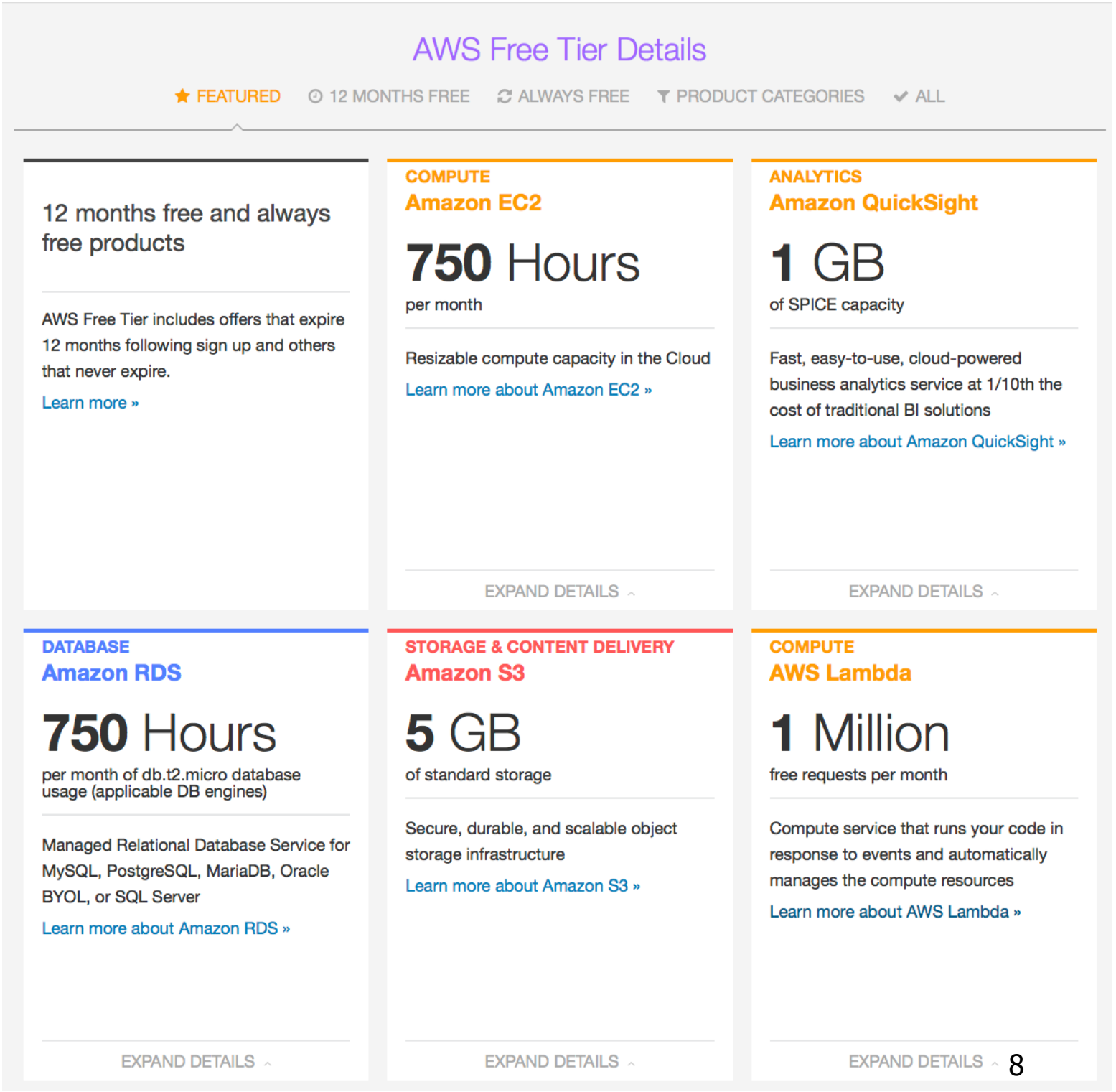
A region is a geographic location where a cloud provider has established at least a data center. Each region is typically isolated from others in terms of network connectivity to ensure high fault tolerance. When you choose a region, you are selecting where your data and services will be hosted, which can impact latency, data residency compliance, and costs.

An availability zone (AZ) is a physically isolated location within a region. Each AZ has independent power, cooling, and networking, so one AZ going down (e.g., due to a power outage) does not affect the others in the same region. Each region usually has multiple AZs to enable redundancy and fault tolerance. When deploying applications, best practices involve spreading resources across multiple AZs within the same region to improve resilience against failures.

Amazon EC2


Amazon EC2
Region: us-east-1
Availability Zone
Availability Zone
Availability Zone
Region: eu-west-1
Availability Zone
Availability Zone
Other Region...
Availability Zone
Availability Zone

AWS free tier

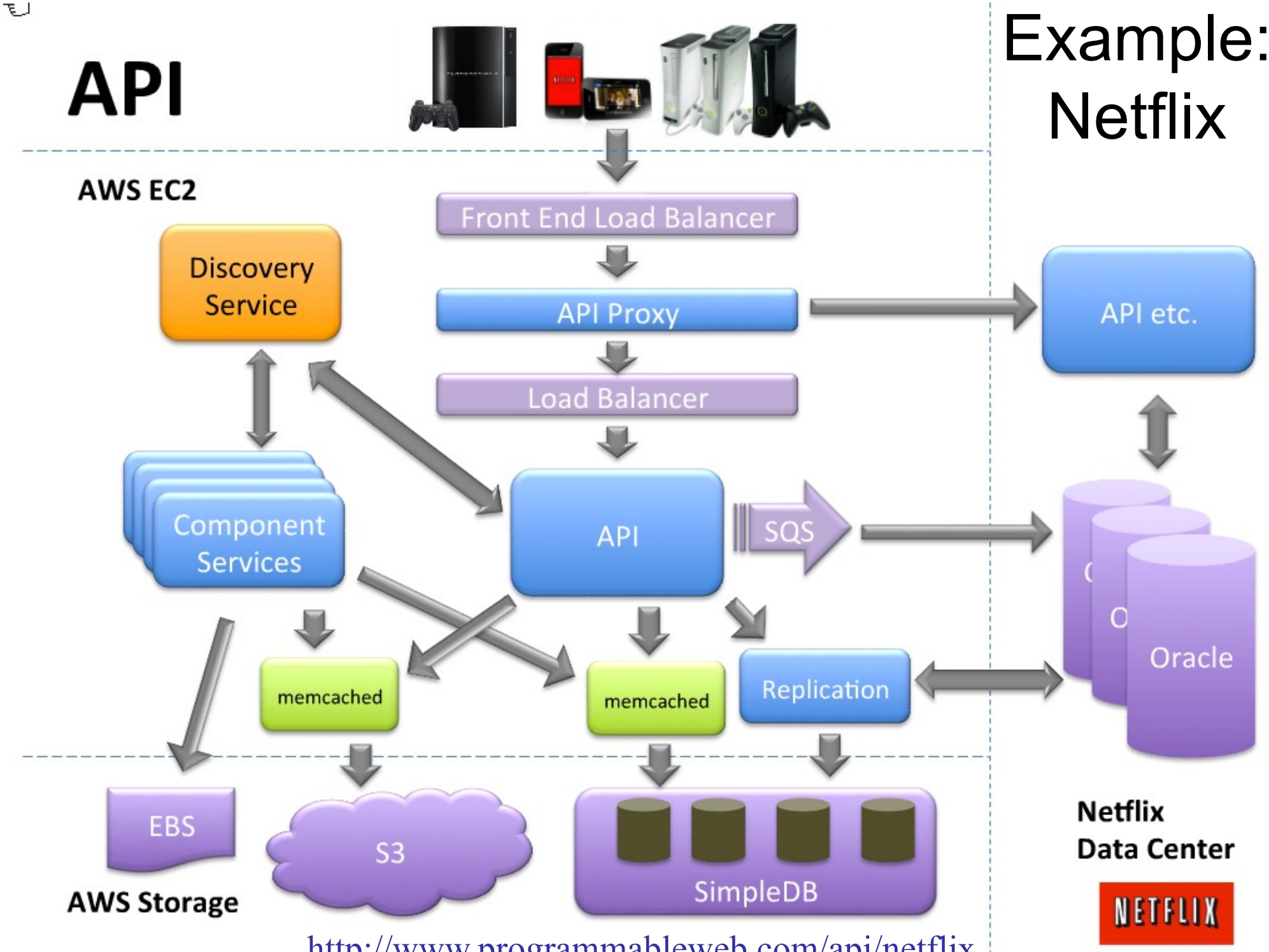


Amazon Elastic Cloud Computing EC2

- Provider of cloud infrastructure
- Data centers worldwide
- Many types of Virtual Machines
  - e.g. Linux, Windows, FreeBSD Unix
- Free tier for one year
- Per-hour charging (NB: complex pricing model, y2021)
  - Medium 0.0049$/h
  - Large 0.0098$/h
  - Xlarge 0.0197$/h

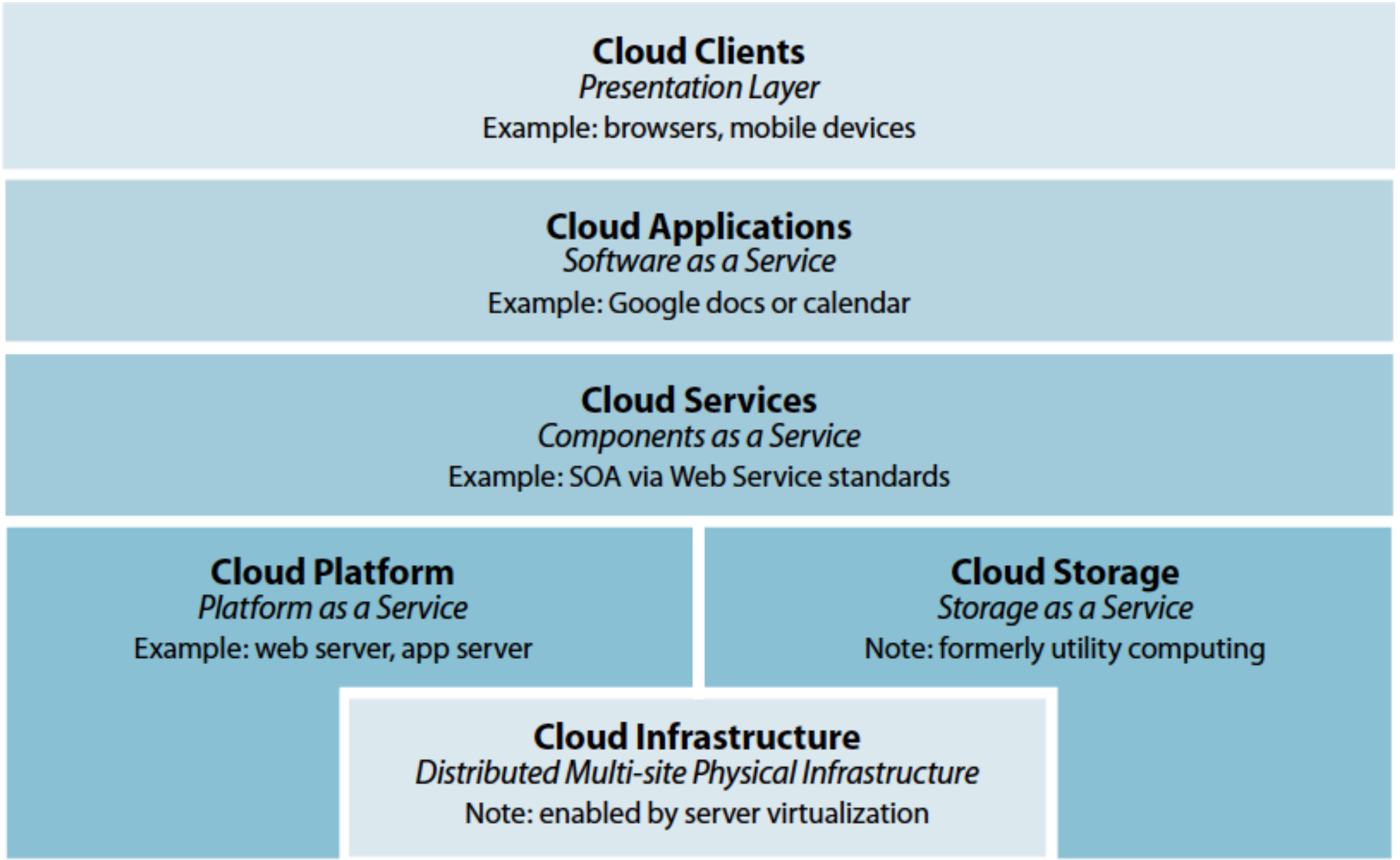
http://aws.amazon.com/ec2/pricing/

Example: Netflix



http://www.programmableweb.com/api/netflix

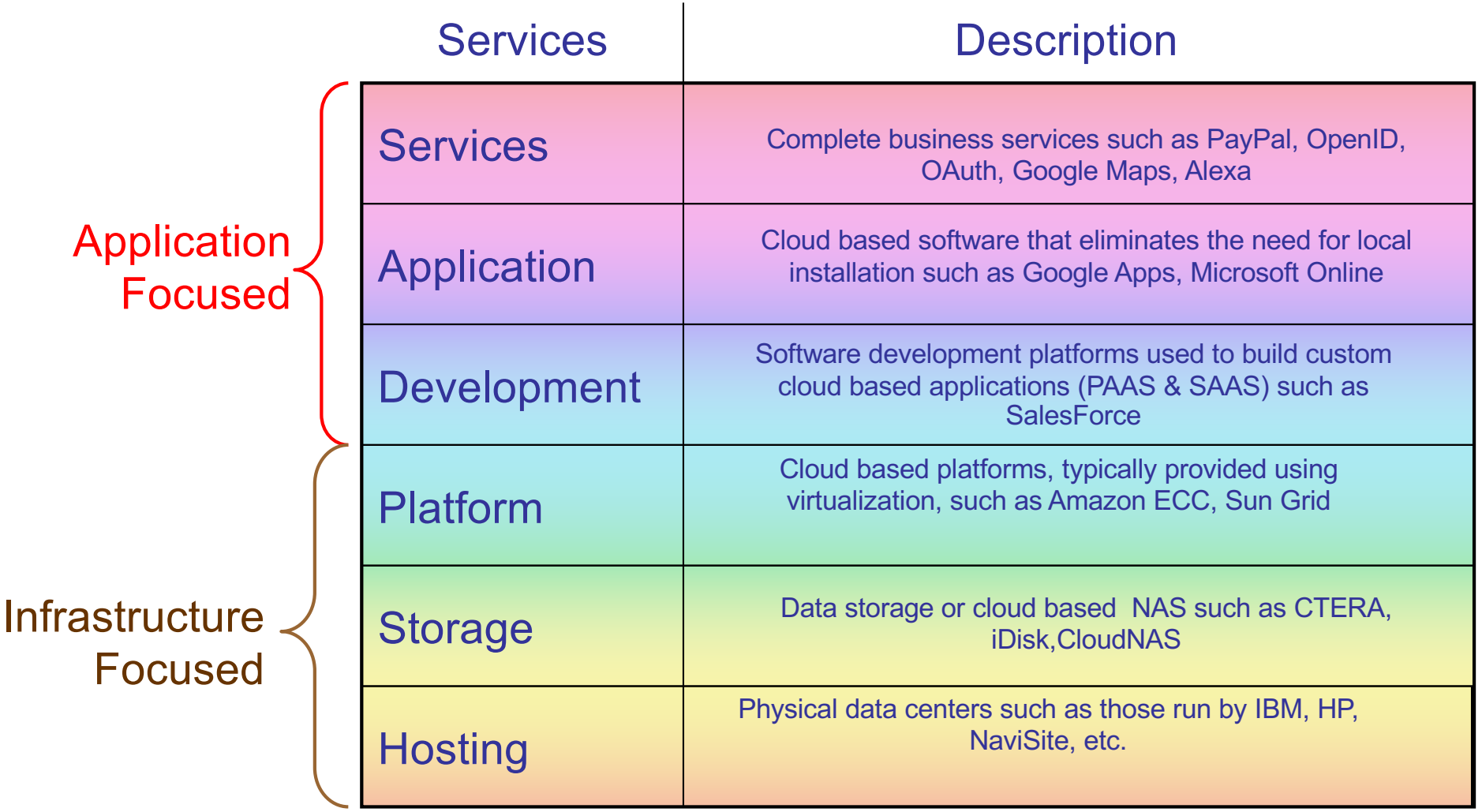
A cloud reference stack



The cloud as a programmable platform

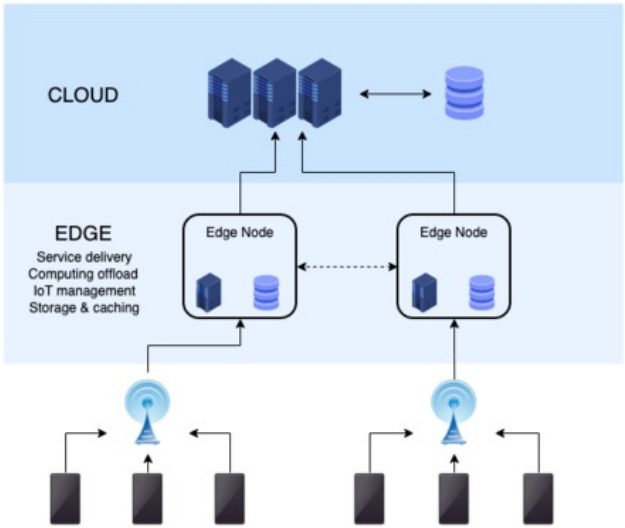
- Programming the cloud requires specific approaches and (programming for the cloud is different from programming for a traditional OS, the Web, or any smarthpones)
- Any cloud platform offers some API so that the programmer has not to care of the underlying infrastructure
- This is a generic idea called «serverless»
- Amazon Lambda is a very popular serverless API, introduced in 2014, and is Amazon-specific
- Most companies try to avoid Amazon vendor lockin using Kubernetes, that is open source and portable , but complex to manage

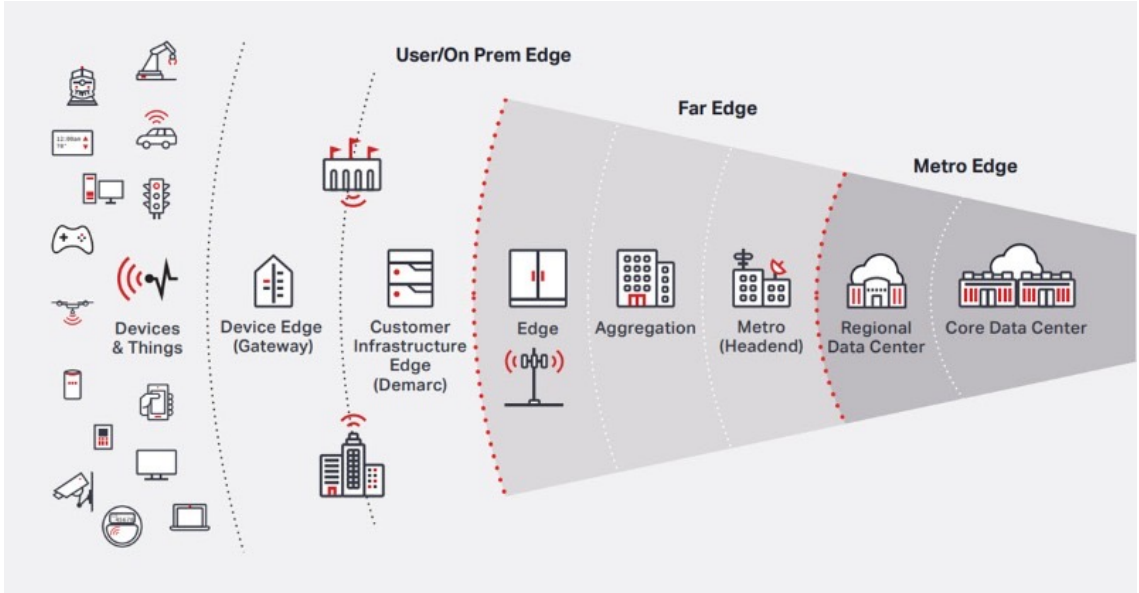
Cloud Computing Service Layers



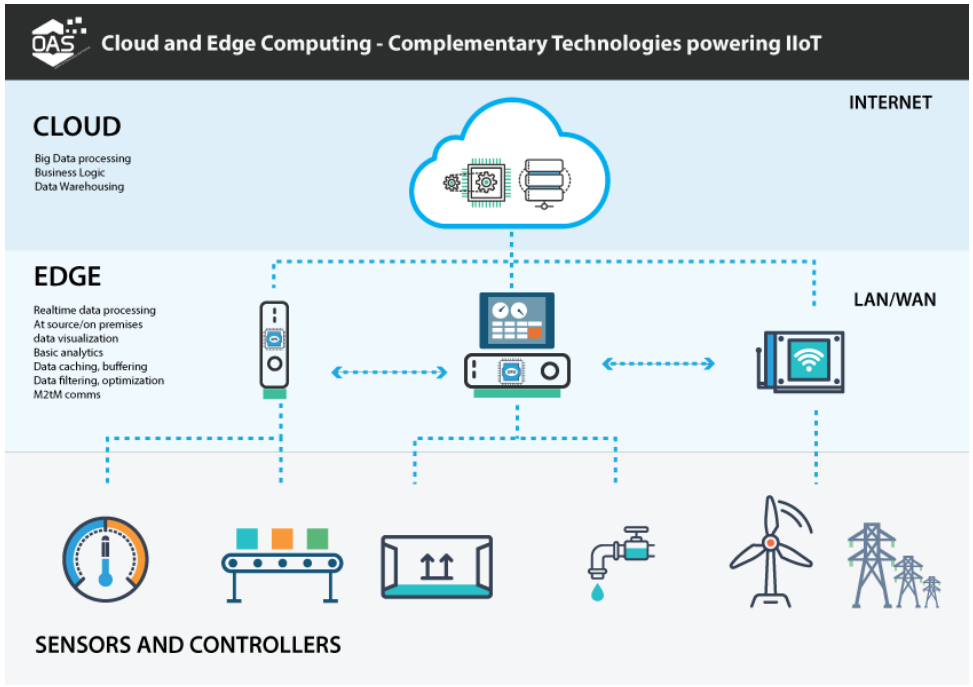
Edge computing

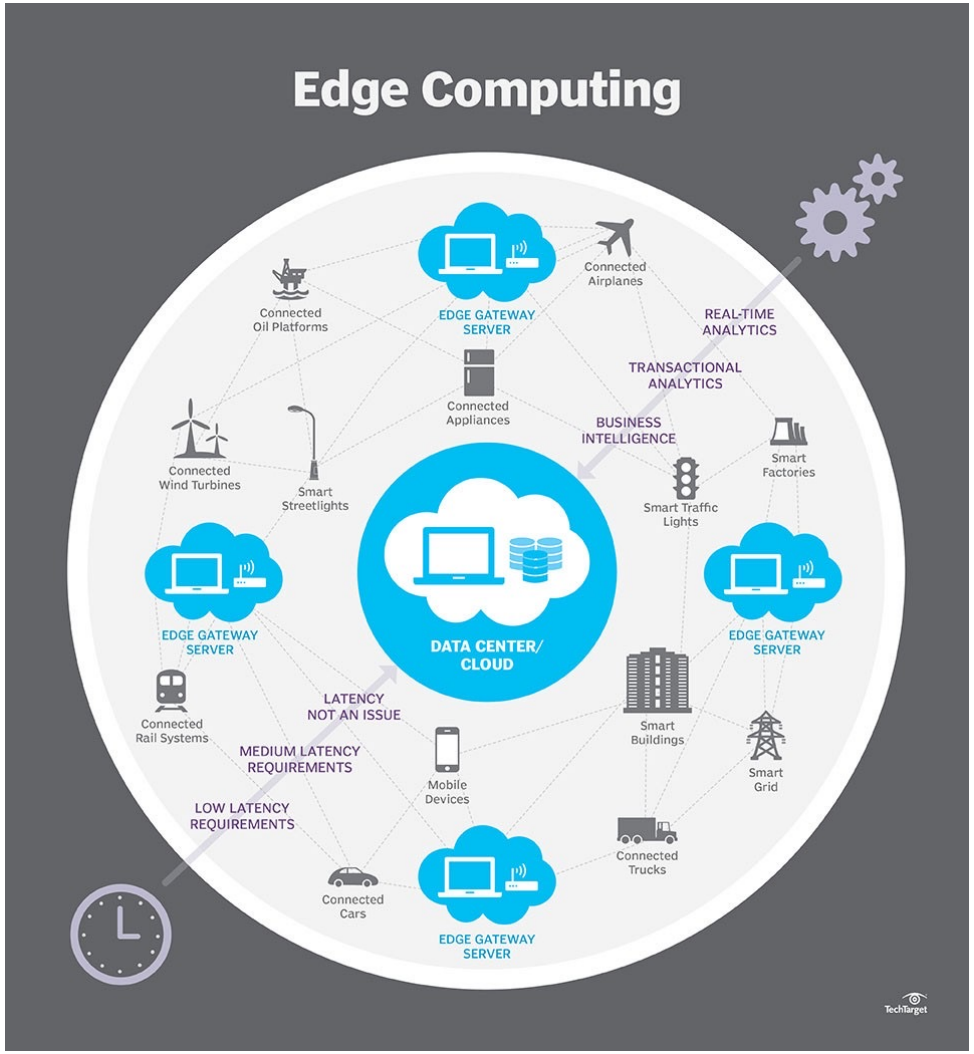
- Edge computing is a cloud-based paradigm that brings computation and data storage closer to the location where it is needed, to improve response times and save bandwidth.
- The origins of edge computing lie in content delivery networks to serve web and video content from edge servers that were deployed close to users
- Edge computing enables IoT architectures



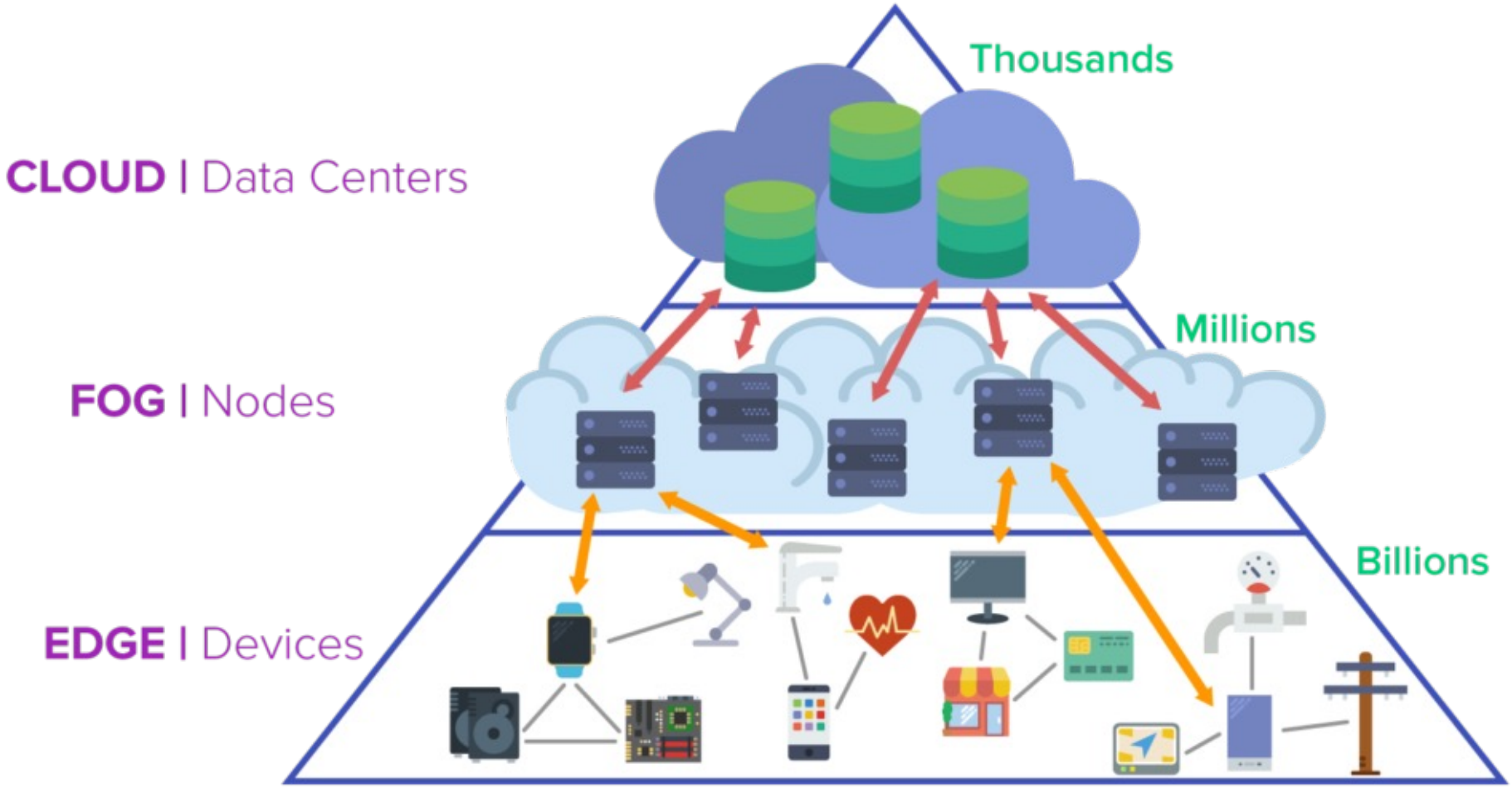


The edge of the internet





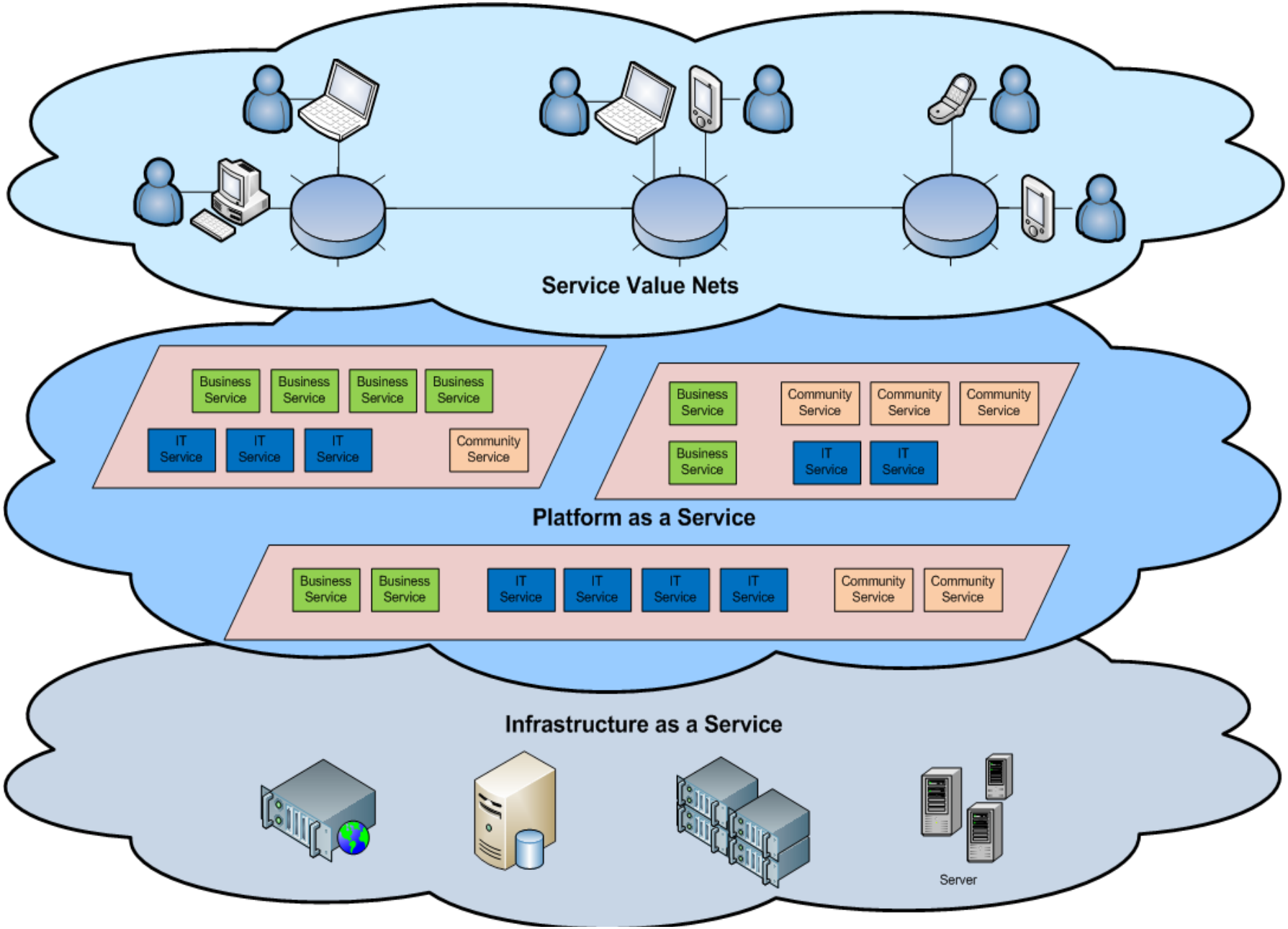
https://medium.com/hackernoon/edge-computing-a-beginners-guide-8976b6886481


CLOUD | Data Centers
Thousands
FOG | Nodes
Millions
EDGE | Devices
Billions

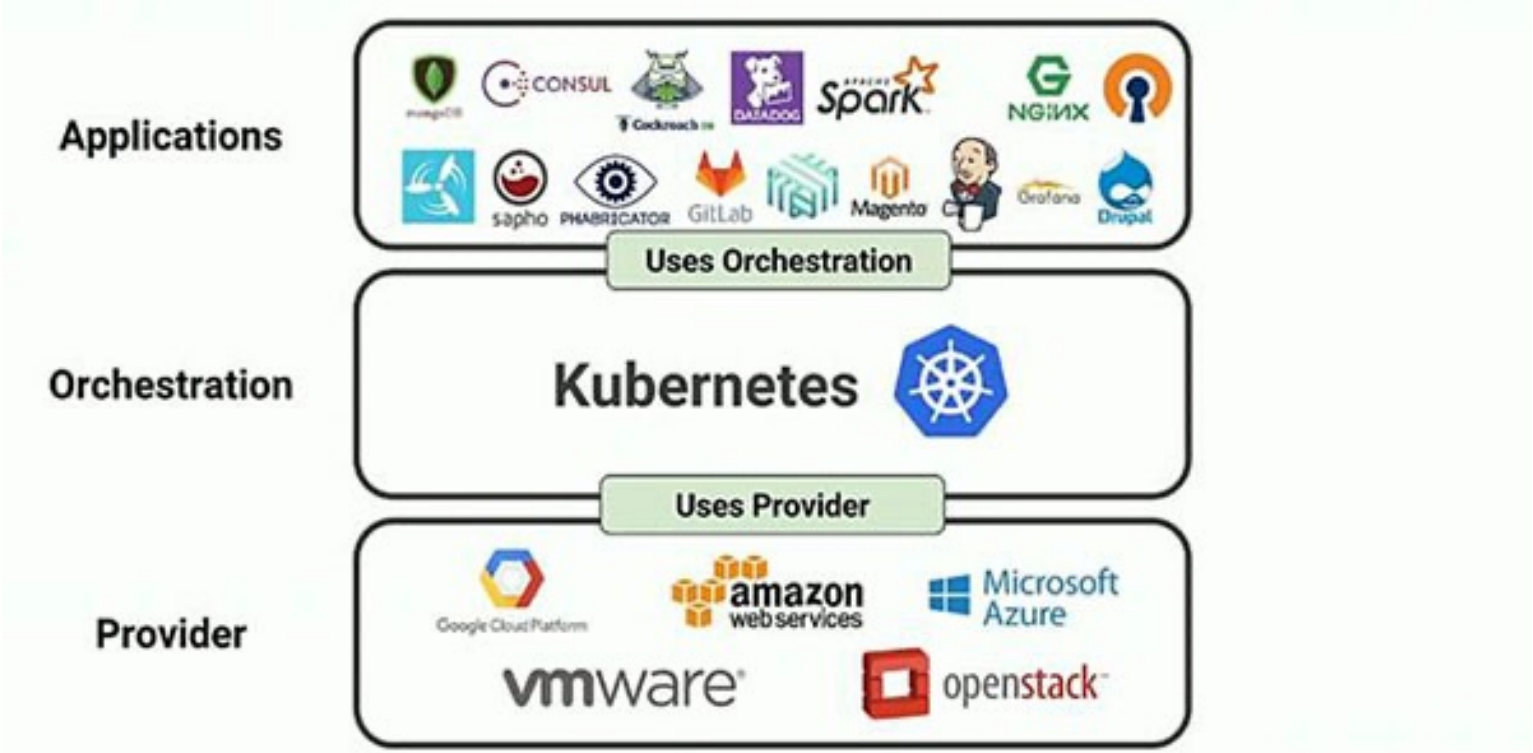
Features in-the-cloud

- Cloud Computing is a generic term used to refer to Internet-based deployment of services
- A number of features define data, applications, services, and infrastructures *in-the-cloud*:
  - **Remotely hosted**: Services or data are hosted remotely
  - **Ubiquitous**: Services or data are available from anywhere
  - **Commodified**: The result is a utility computing model similar to that of traditional utilities, like gas and electricity - you pay for what you use

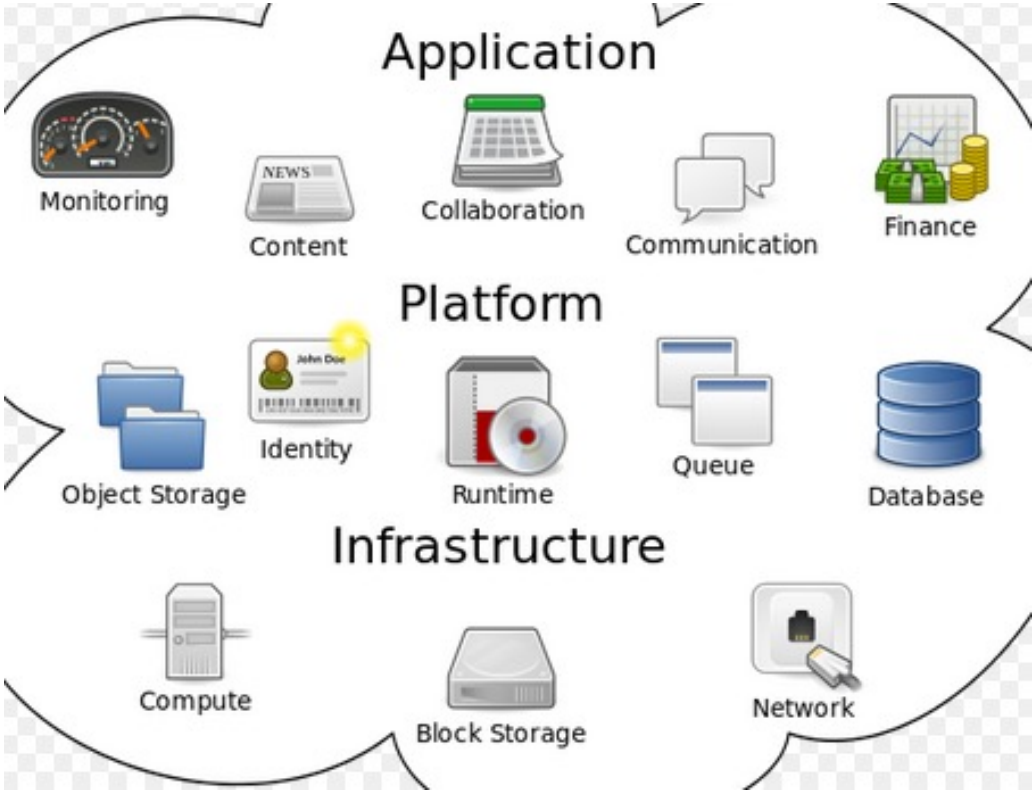
Cloud Architecture

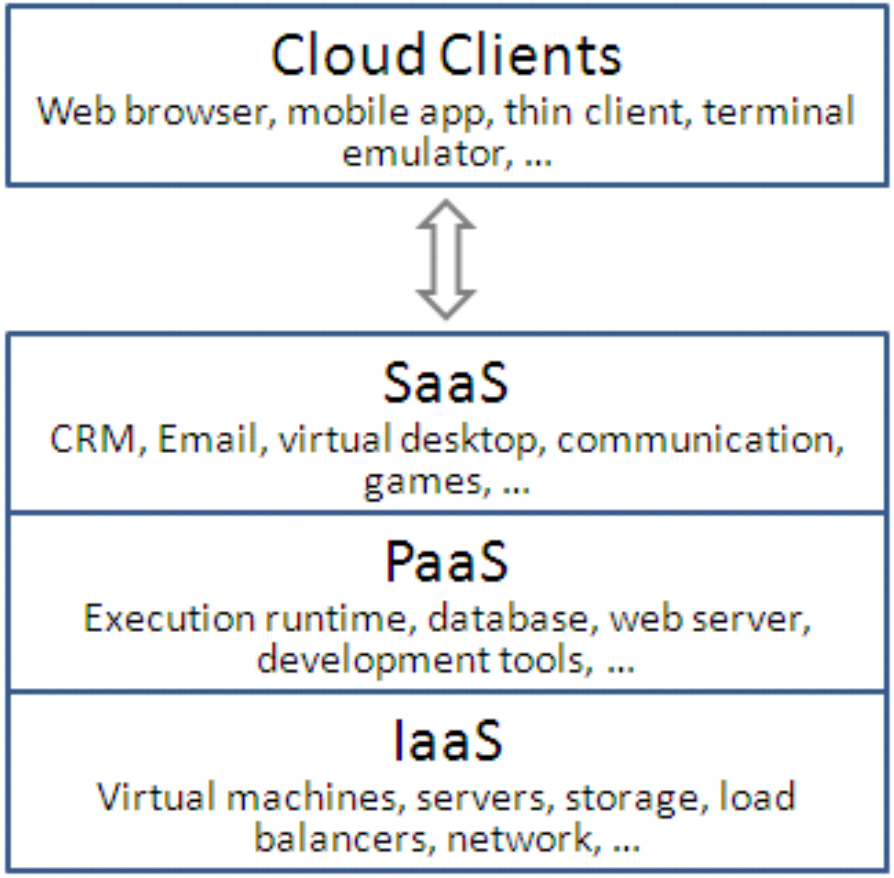


Services in the cloud: examples

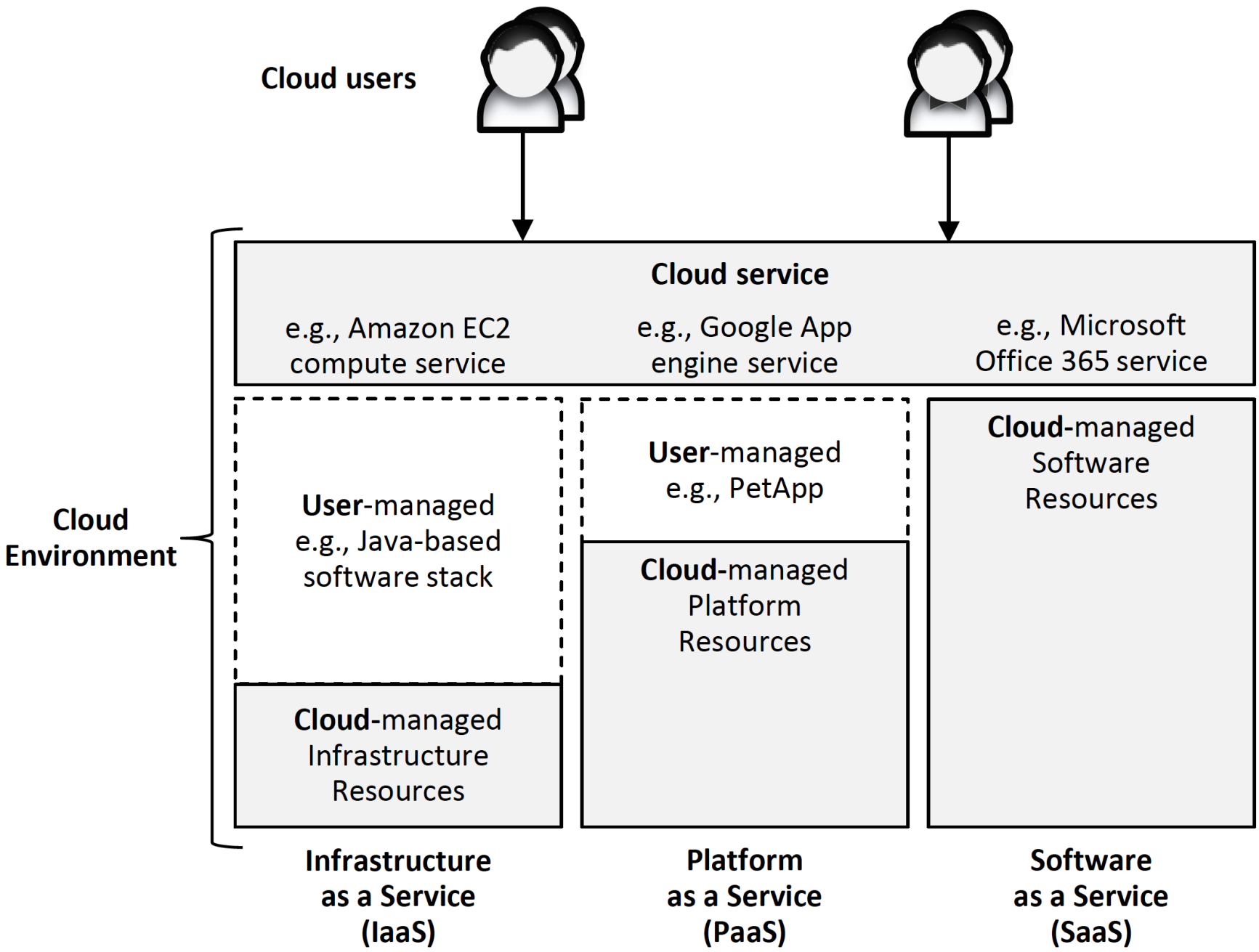


IaaS, PaaS, SaaS

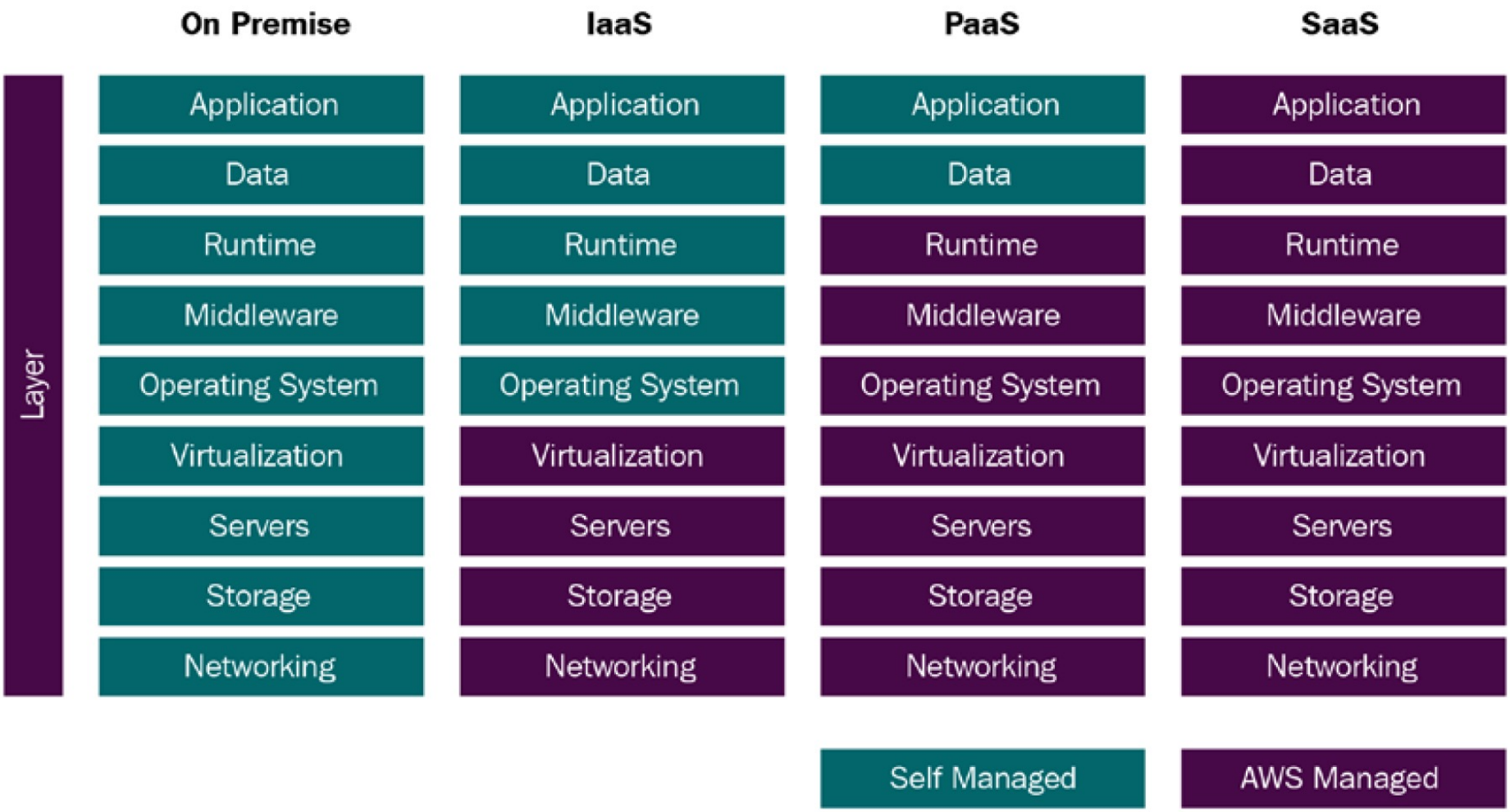




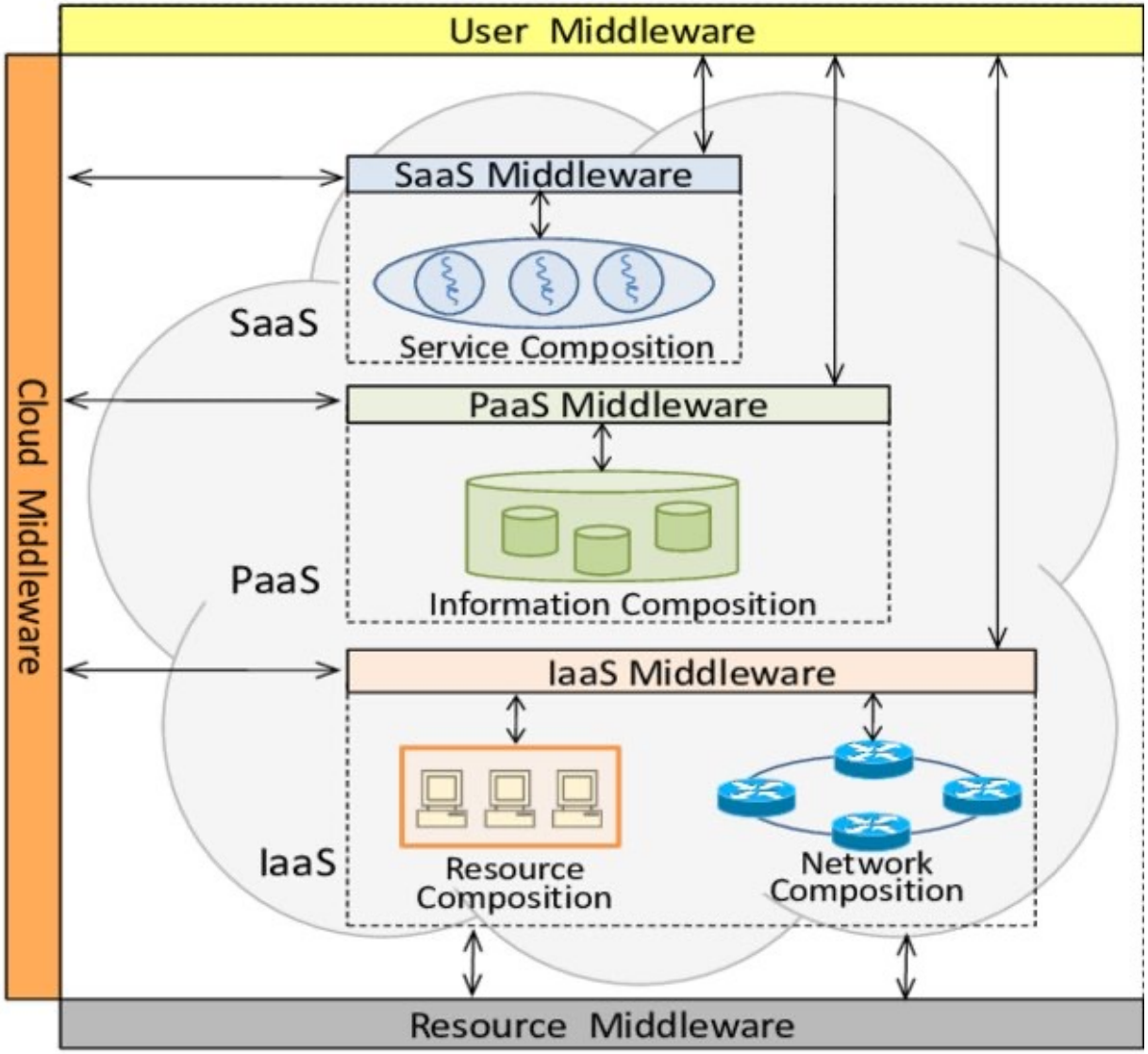
SaaS, PaaS, IaaS



Automated vs self-operated

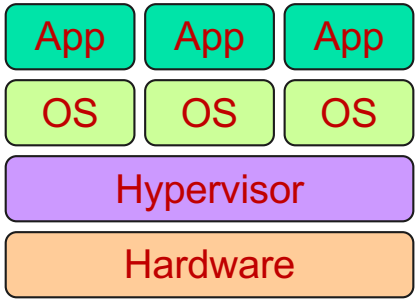


Middleware in the cloud



Virtualization

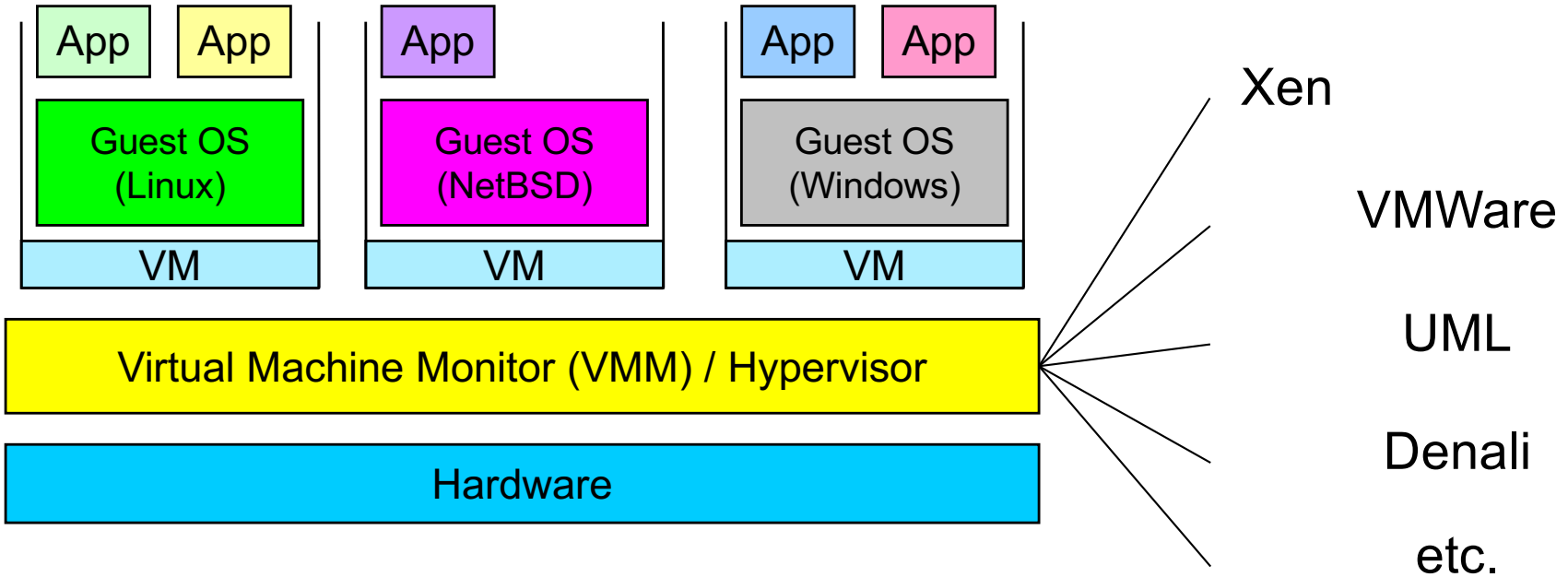
- Virtual workspaces:
  - An abstraction of an execution environment that can be made dynamically available to authorized clients by using well-defined protocols,
  - Resource quota (e.g. CPU, memory share),
  - Software configuration (e.g. O/S, provided services).
- Implement on Virtual Machines (VMs):
  - Abstraction of a physical host machine,
  - Hypervisor intercepts and emulates instructions from VMs, and allows management of VMs,
  - VMWare, Xen, etc.
- Provide infrastructure API:
  - Plug-ins to hardware/support structures



Virtualized Stack

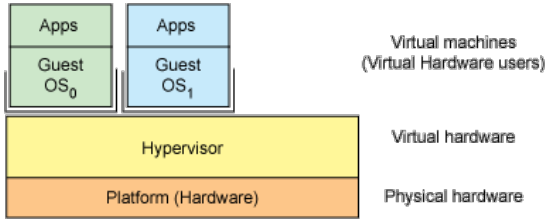
Virtual Machines

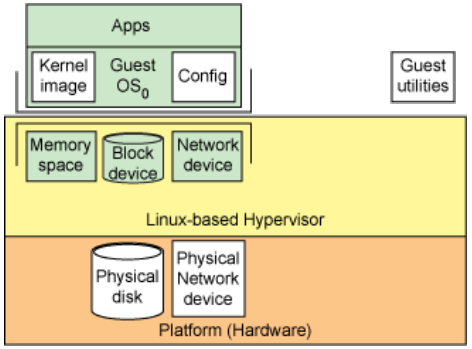
- VM technology allows multiple virtual machines to run on a single physical machine.



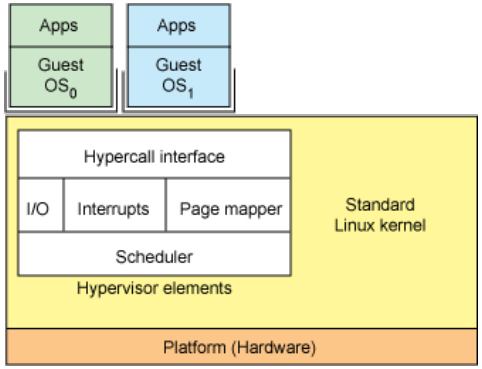
Performance: Para-virtualization (e.g. Xen) is very close to raw physical performance!

Hypervisor





Minimal mapping



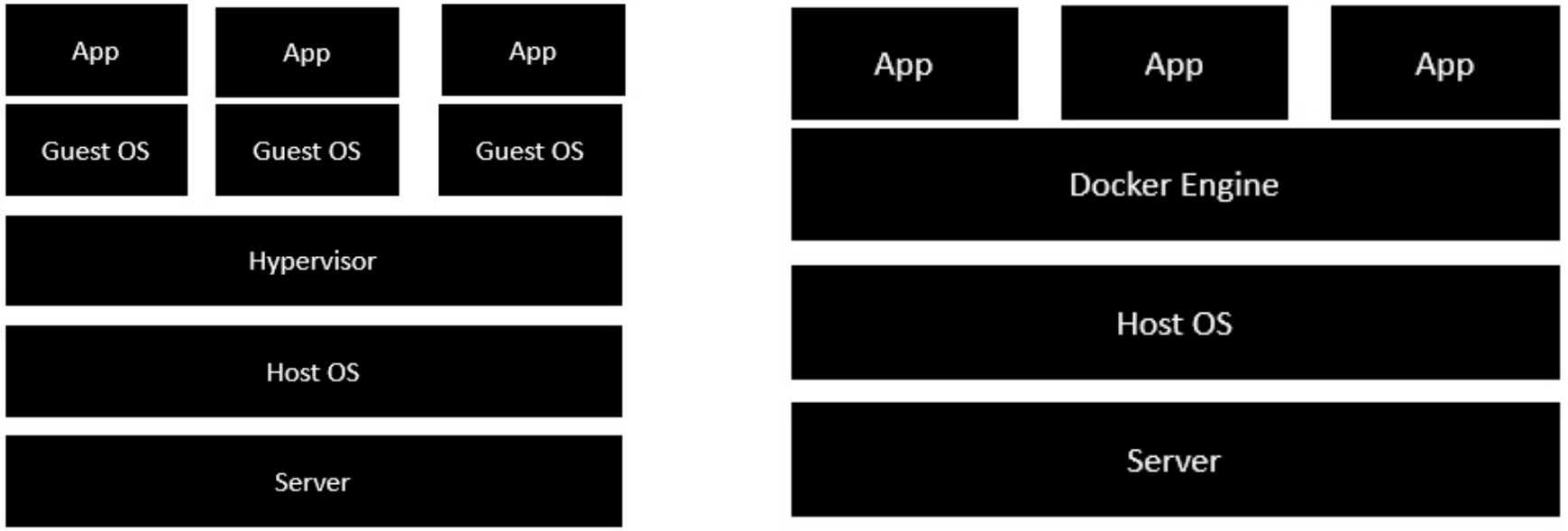
Simplified hypervisor architecture

Source: www.ibm.com/developerworks/linux/library/l-hypervisor/

Docker

Docker containers wrap a software in a complete filesystem that contains everything needed to run: code, runtime, system tools, libraries – anything that can be installed on a server.

This guarantees that the software will always run the same, regardless of its environment.

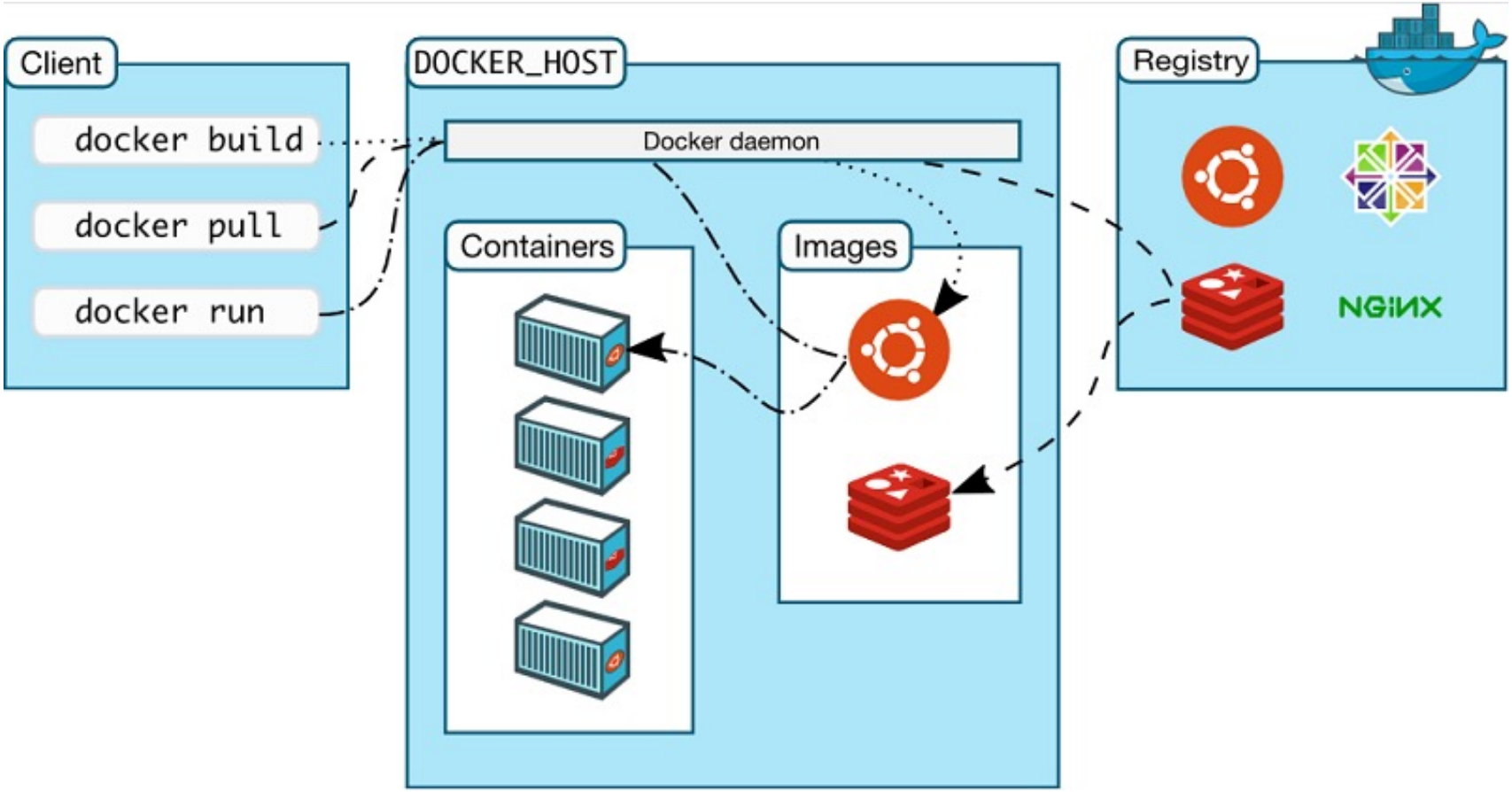


medium.freecodecamp.org/a-beginner-friendly-introduction-to-containers-vms-and-docker-79a9e3e119b

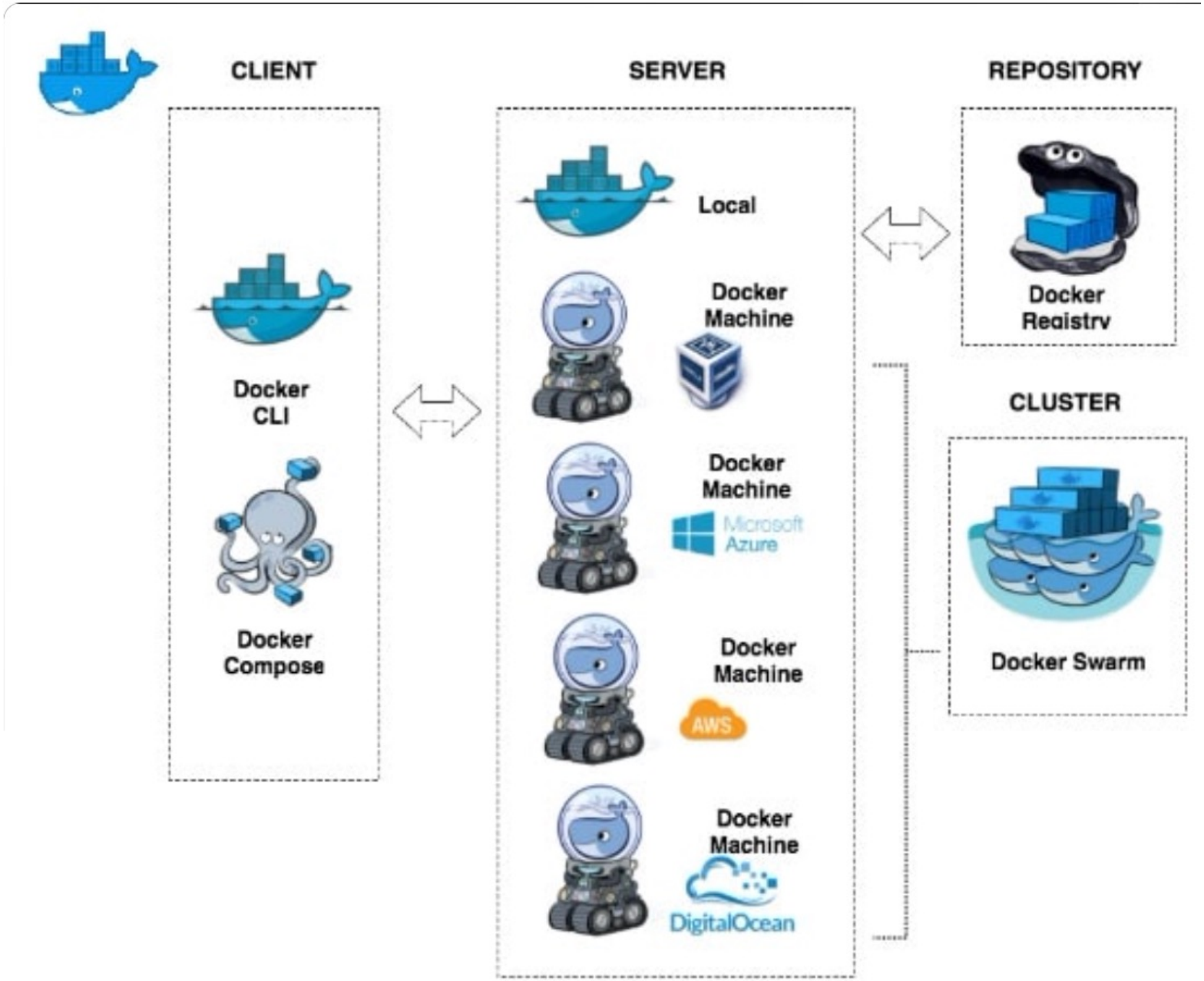
Docker

- Docker can package an application together with all its libraries, configuration files, and dependencies
- Docker is used by both developers and system administrators, making it an essential tool for DevOps teams
- Every Docker container needs a Dockerfile
  - Operating system supporting the container: What is the operating system associated with the container?
  - Environmental variables used: For example, most deployments require a list of variables. Also, is this a production or test deployment? What department is this deployment for? What department should be billed?
  - Locations of files used: For example, where are the data files located? Where are the executable files?
  - Network ports used: For example, what ports are open? Which port is used for HTTP traffic?

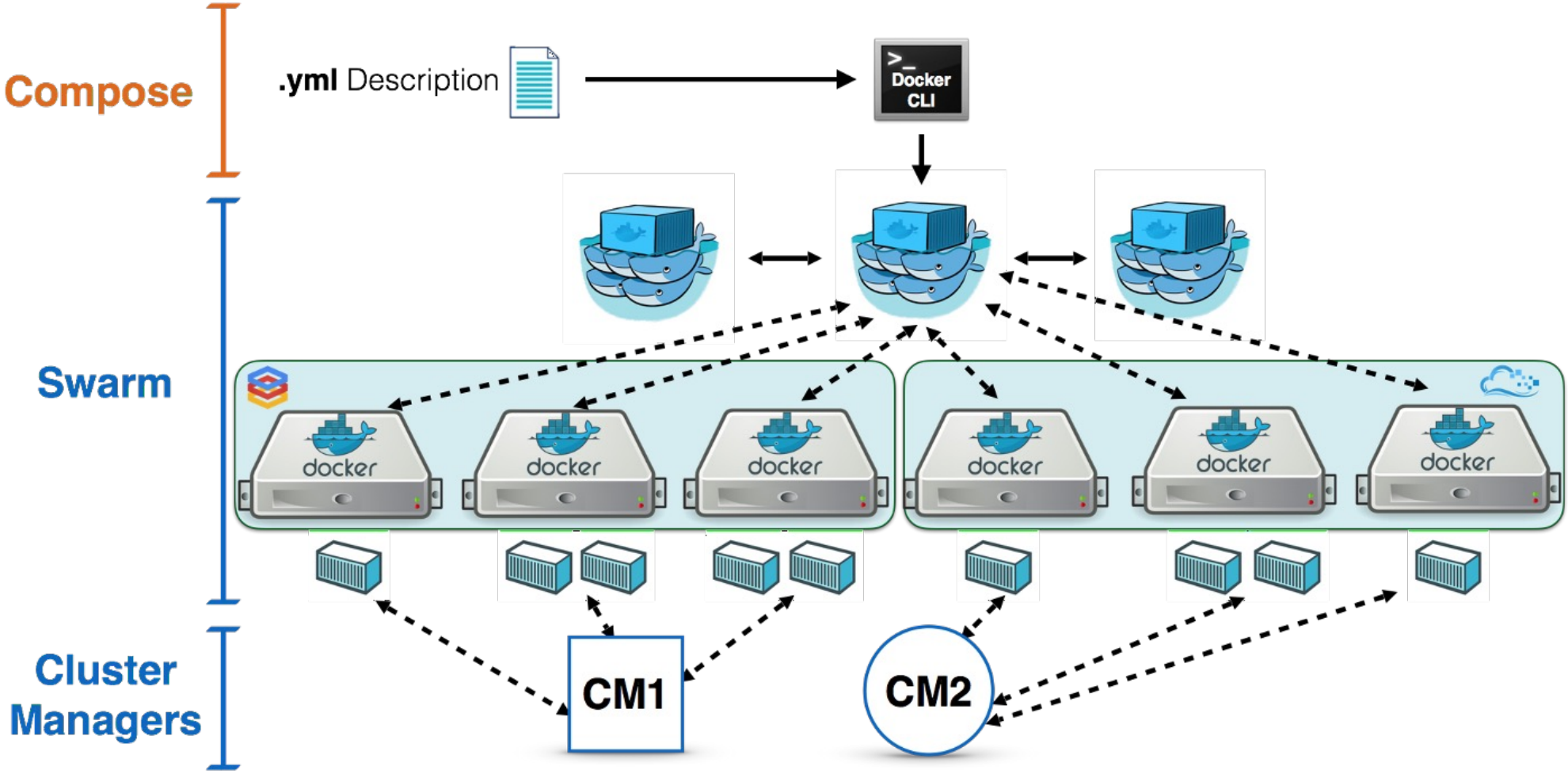
Docker


Client
docker build
docker pull
docker run
DOCKER_HOST
Docker daemon
Containers
Images
Registry
NGINX

Docker compose



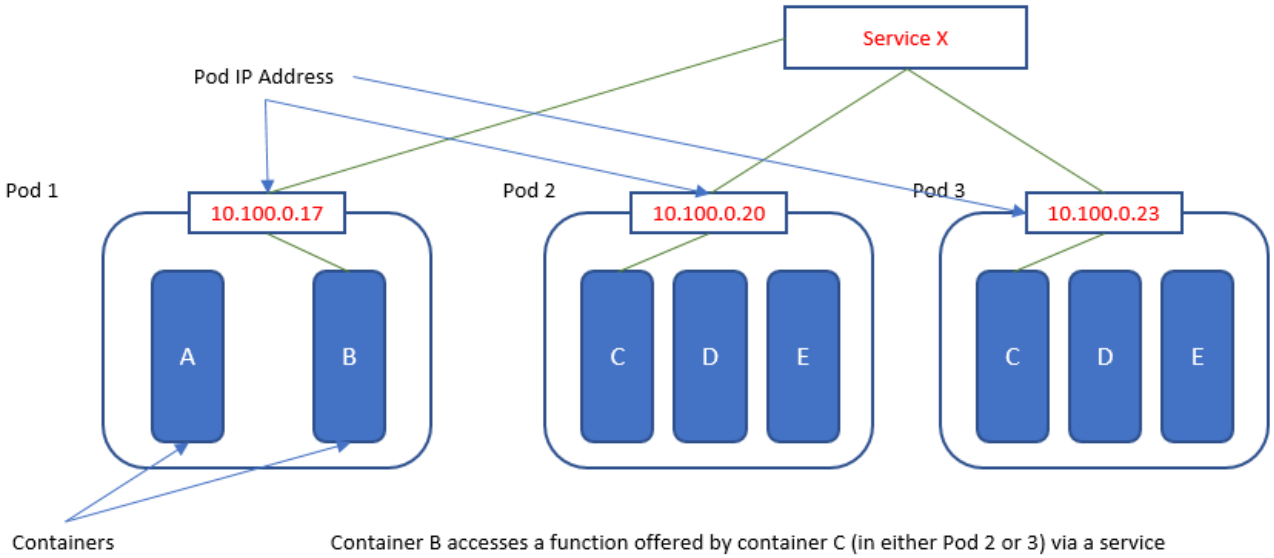
Docker swarm



Docker Swarm is a clustering and scheduling tool for Docker containers.
With Swarm, IT administrators and developers can establish and manage a cluster of Docker nodes as a single virtual system

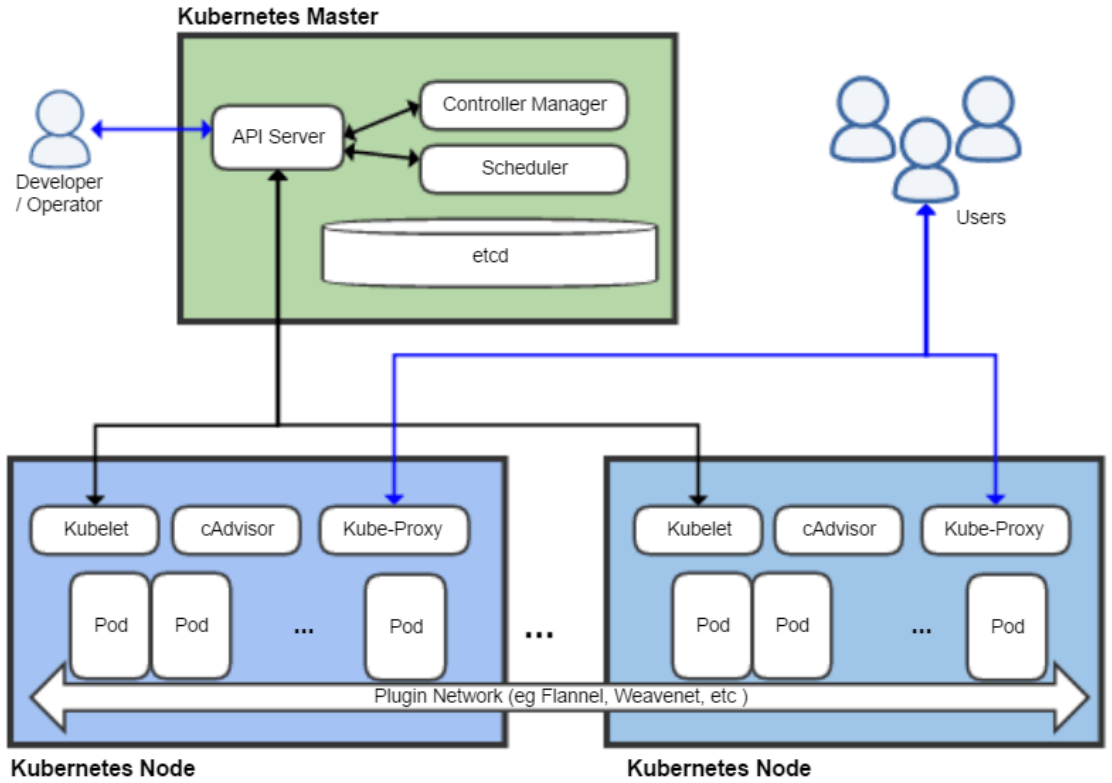
https://medium.com/tech-tajawal/create-cluster-using-docker-swarm-94d7b2a10c43

Kubernetes



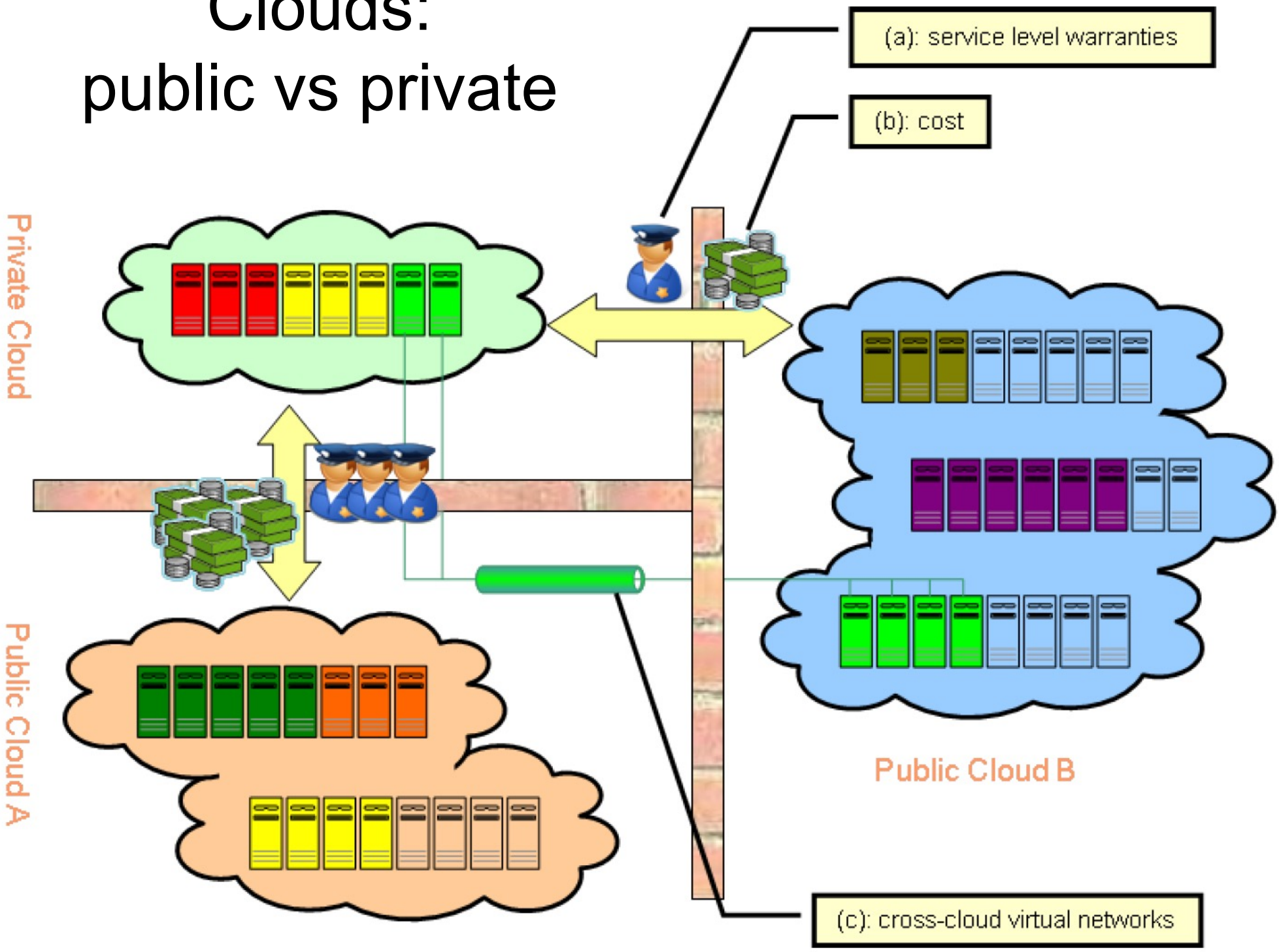
- Kubernetes is an open-source container-orchestration system for automating application deployment, scaling, and management.
- It aims to provide a "platform for automating deployment, scaling, and operations of application containers across clusters of hosts".
- It works with a range of container tools, including Docker.
- Many cloud services offer a Kubernetes-based platform or infrastructure as a service (PaaS or IaaS) on which Kubernetes can be deployed as a platform-providing service.

Kubernetes



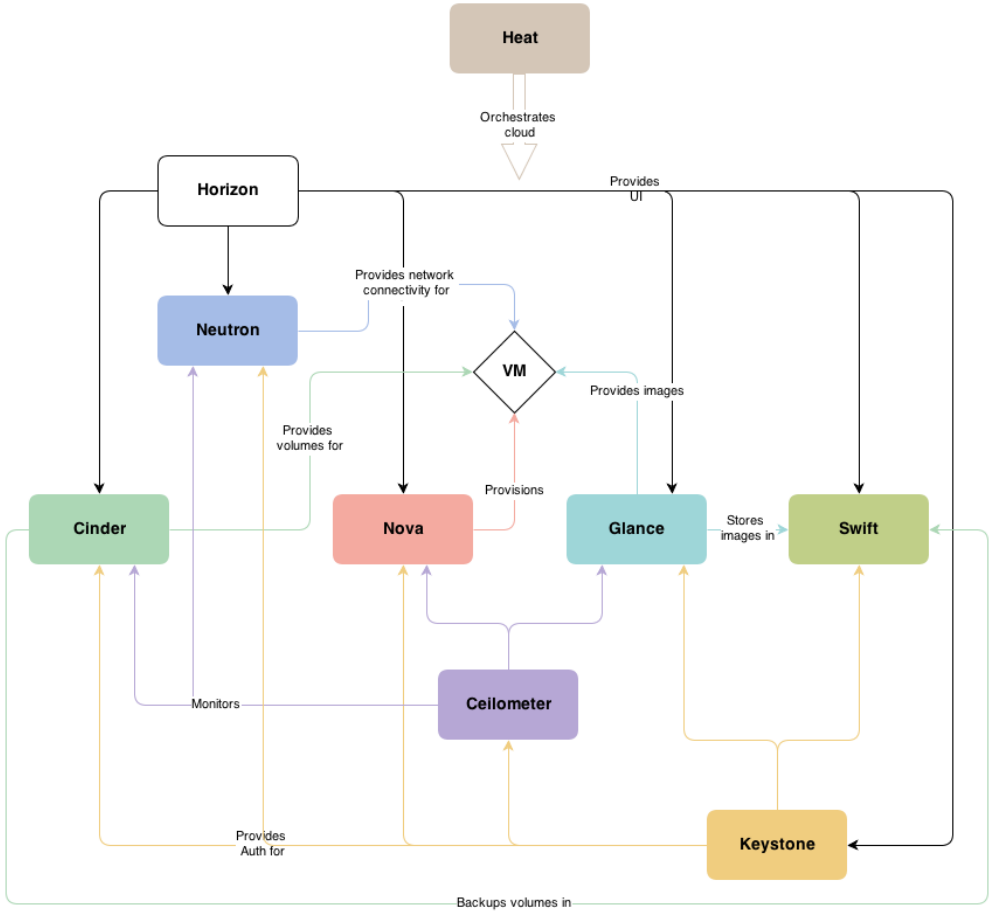
A Kubernetes master is a grouping of three processes operating in a node in the Kubernetes cluster. This node is designated as the master node

Clouds: public vs private



OpenStack

- Open source cloud computing software (used for public or private clouds)
- Consists in multiple services:
  - **Keystone:** identity services (authentication, authorization, accounting)
  - **Cinder:** management of block storage volumes
  - **Nova:** management and provision of virtual resources (VM instances)
  - **Glance:** management of VM images
  - **Swift:** management of object storage
  - **Neutron:** management of network resources (IPs, routing, connectivity)
  - **Horizon:** GUI dashboard for end users
  - **Heat:** orchestration of virtualized environments (important for providing elasticity)
  - **Ceilometer:** monitoring of virtual resources

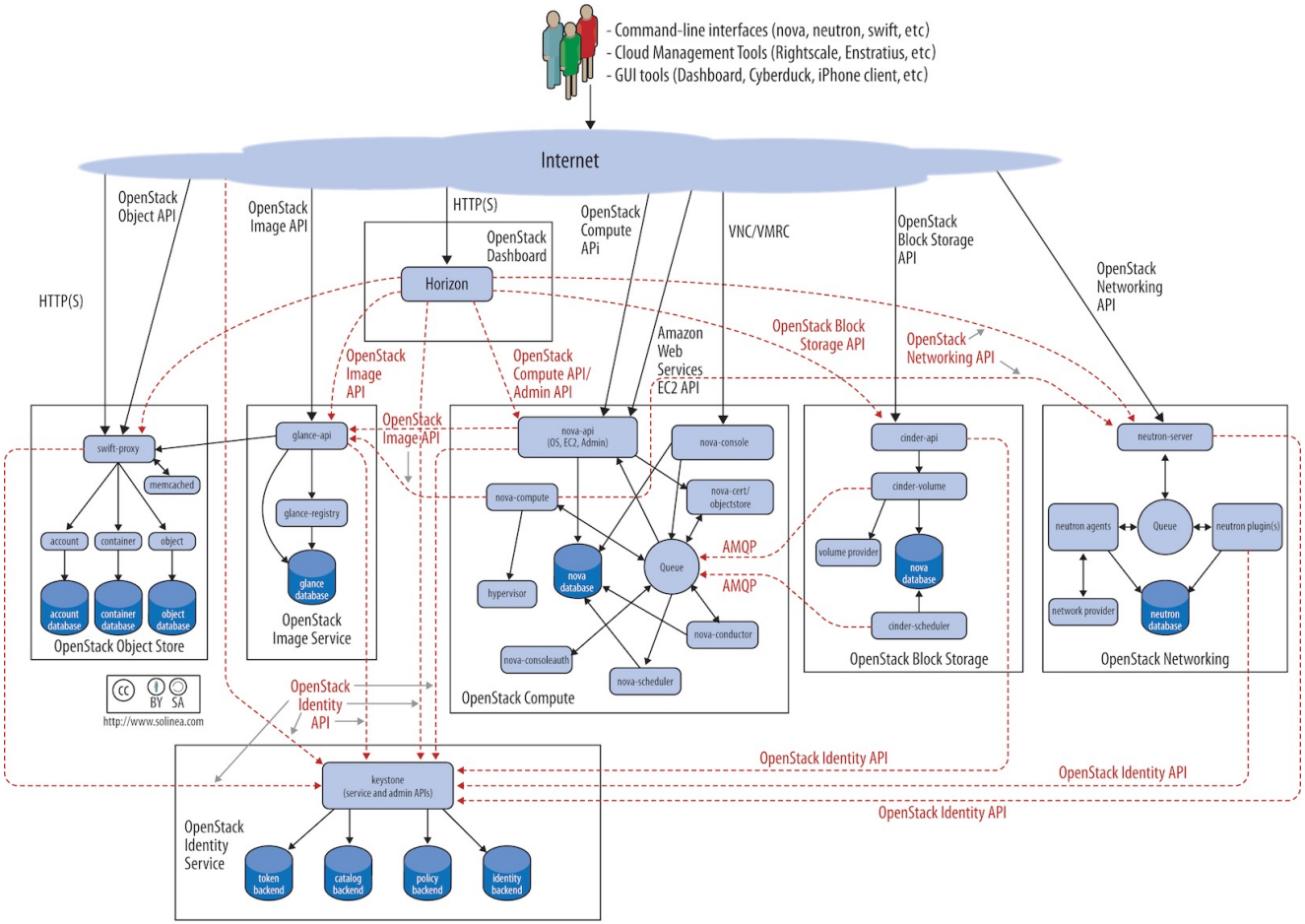


OpenStack

Opebstack is deployed as infrastructure-as-a-service (IaaS), whereby virtual servers and other resources are made available to customers.

The platform consists of components that control diverse, multi-vendor hardware pools of processing, storage, and networking resources throughout a data center.

Users either manage it through a web-based dashboard, through command-line tools, or through RESTful web services.



Kubernetes vs OpenStack

- **Kubernetes:** Kubernetes is an open source *orchestration* system for Docker containers. It handles scheduling onto nodes in a computer cluster and actively manages workloads to ensure that their state matches the users declared intentions.
- **OpenStack:** *Open source software for building private and public clouds*. A cloud operating system that controls large pools of compute, storage, and networking resources throughout a datacenter, all managed through a dashboard that gives administrators control while empowering their users to provision resources through a web interface.
- Kubernetes and OpenStack are respectively classified as **"Container"** and **"Open Source Cloud"** .

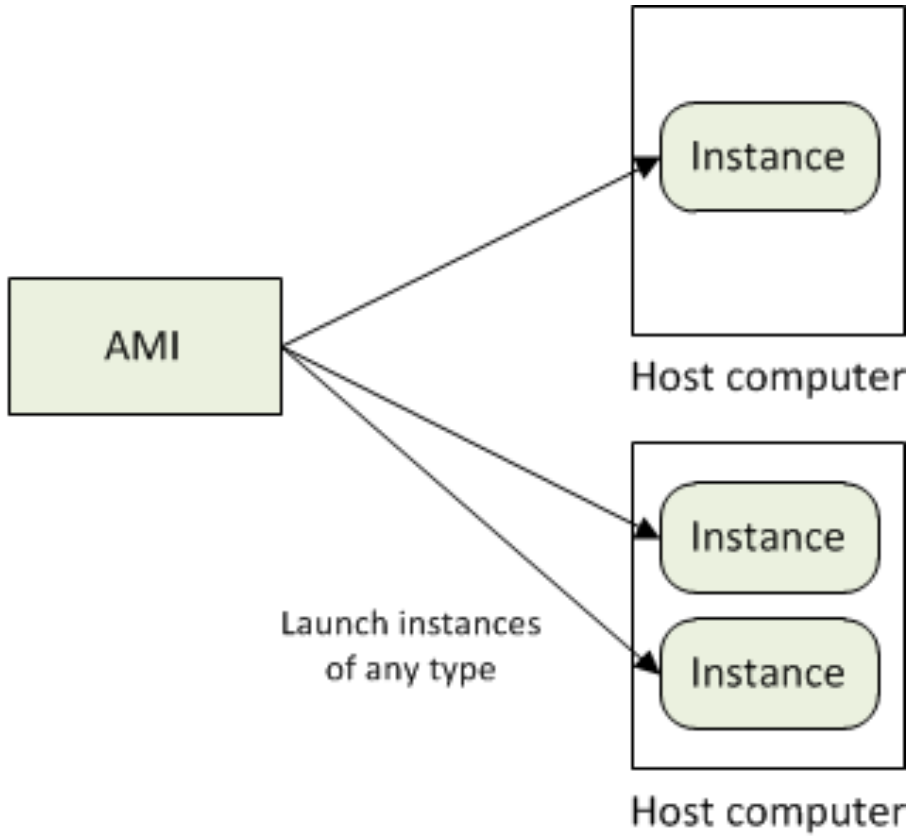
- **"Leading docker container management solution"** is the primary reason why developers consider Kubernetes over the competitors, whereas **"Private cloud"** was stated as the key factor in picking OpenStack.

Main types of clouds: IaaS

- Infrastructure as a service: Resource Clouds, provide resources as services to the user
- Pay per use based on SLA
- Different resources may be provided :
  - AMI (Amazon Machine Image)
  - Data & Storage Clouds examples:
    Amazon S3 (Simple Storage Service), SQL Azure.
  - Compute Clouds examples:
    Amazon EC2 (Elastic Cloud Computing), Zimory, Elastichosts.

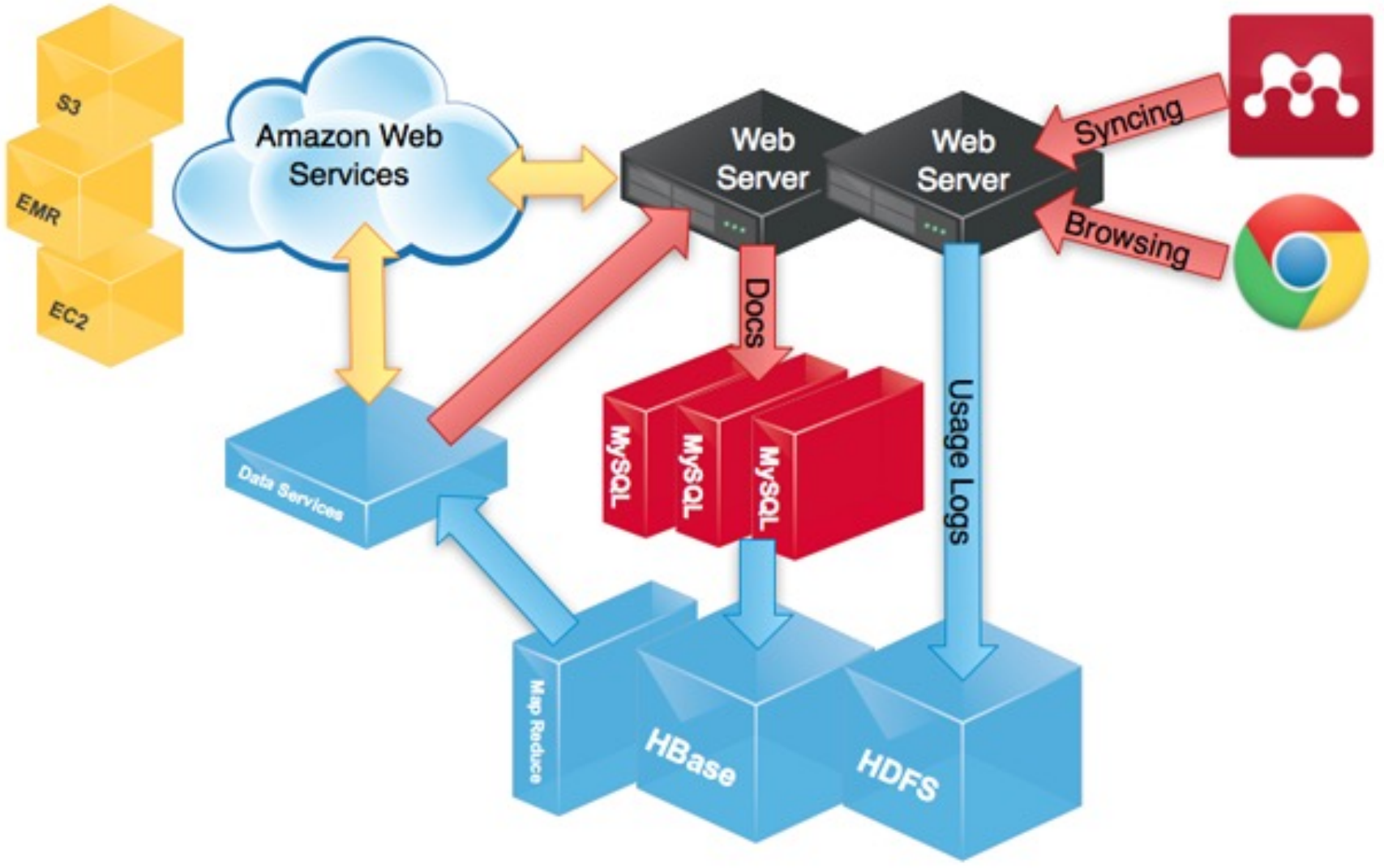
Amazon Machine Images

- An *Amazon Machine Image (AMI)* is a template that contains a software configuration (for example, an operating system, an application server, and applications)
- From an AMI, you launch *instances*, which are copies of the AMI running as virtual servers in the cloud
- You can launch multiple instances of an AMI



http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html

Mendeley (Elsevier) case study

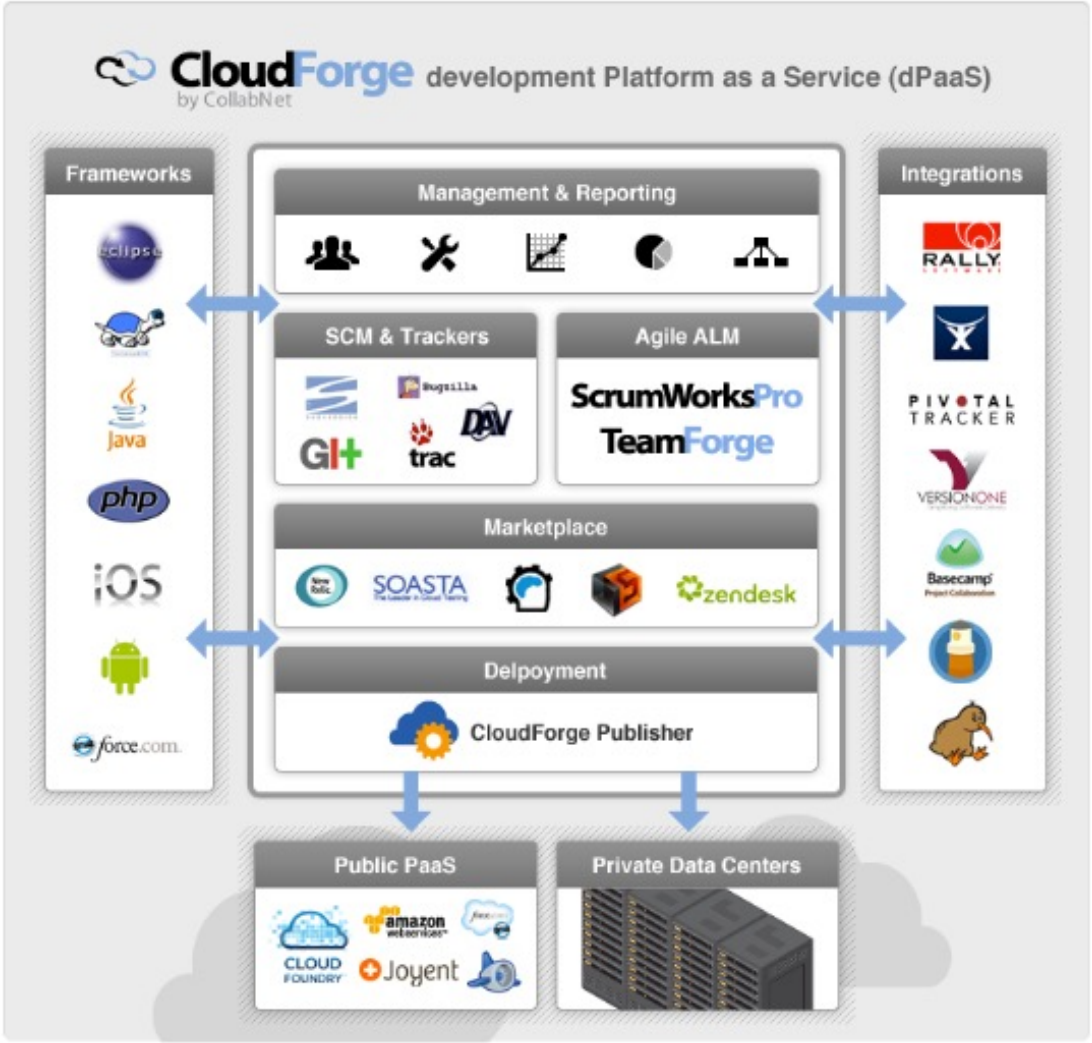


Main types of clouds: PaaS

- Platform as a Service provides computational resources via a platform upon which applications and services can be developed and hosted.
- PaaS makes use of dedicated APIs to control the behaviour of a server hosting engine which executes and replicates the execution according to user requests (e.g. access rate).
- As each provider exposes its own API, applications developed for one specific cloud provider cannot be moved to another cloud host
- Examples: Force.com, CloudForge, Google App Engine, Heroku (SalesForce), CloudBees, OpenShift, Windows Azure (Platform)
- Pattern: MapReduce

PaaS example: CloudForge (terminated)

CloudForge offered application development tools and services, such as Git hosting, Subversion (SVN) hosting, issue trackers and Application Lifecycle Management



https://www.cloudforge.com/teamforge-comparison-matrix

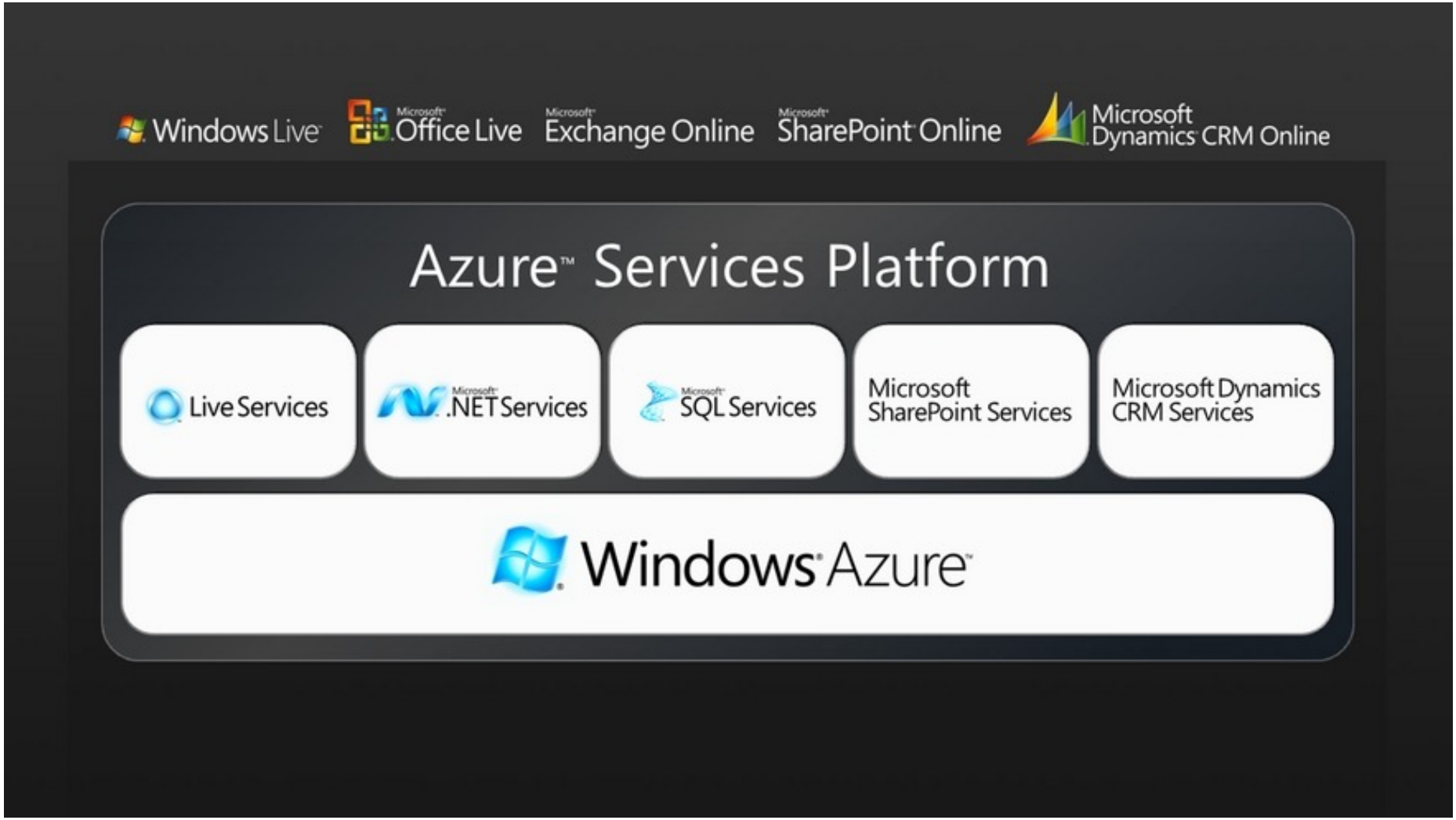
Deployment

- "Deployment-as-a-service" increasingly common
  - monthly pay-as-you-go curated environment (Heroku)
  - hourly pay-as-you-go cloud computing (EC2)
  - hybrid: overflow from fixed capacity to elastic capacity
  - Remember administration costs when comparing!
- Good framework can help at deployment time
  - Separate abstractions for different types of state: session state, asset server, caching, database
  - ORM – natural fit for social computing, and abstracts away from SQL (vs Web 1.0 PHP, e.g.)
  - REST – make your app RESTful from start, so that "SOA"-ifying it is trivial
- Scaling structured storage: open challenge

Microsoft Azure

- Windows Azure
  - Compute: Virtualized computing based on Windows
  - ServerStorage: Durable, scalable, & available storage
  - Management: Automated management of the service
- SQL AzureDatabase:
  - Relational processing for structured/unstructured data
- .Net Services
  - Service Bus: General purpose application bus
  - Access Control: Rule-driven
- Azure provides a complete cloud computing stack. The administration interface is simple. It allows to allocate a server or database capacity, hook in the service bus, and configure an application

MS Azure services platform

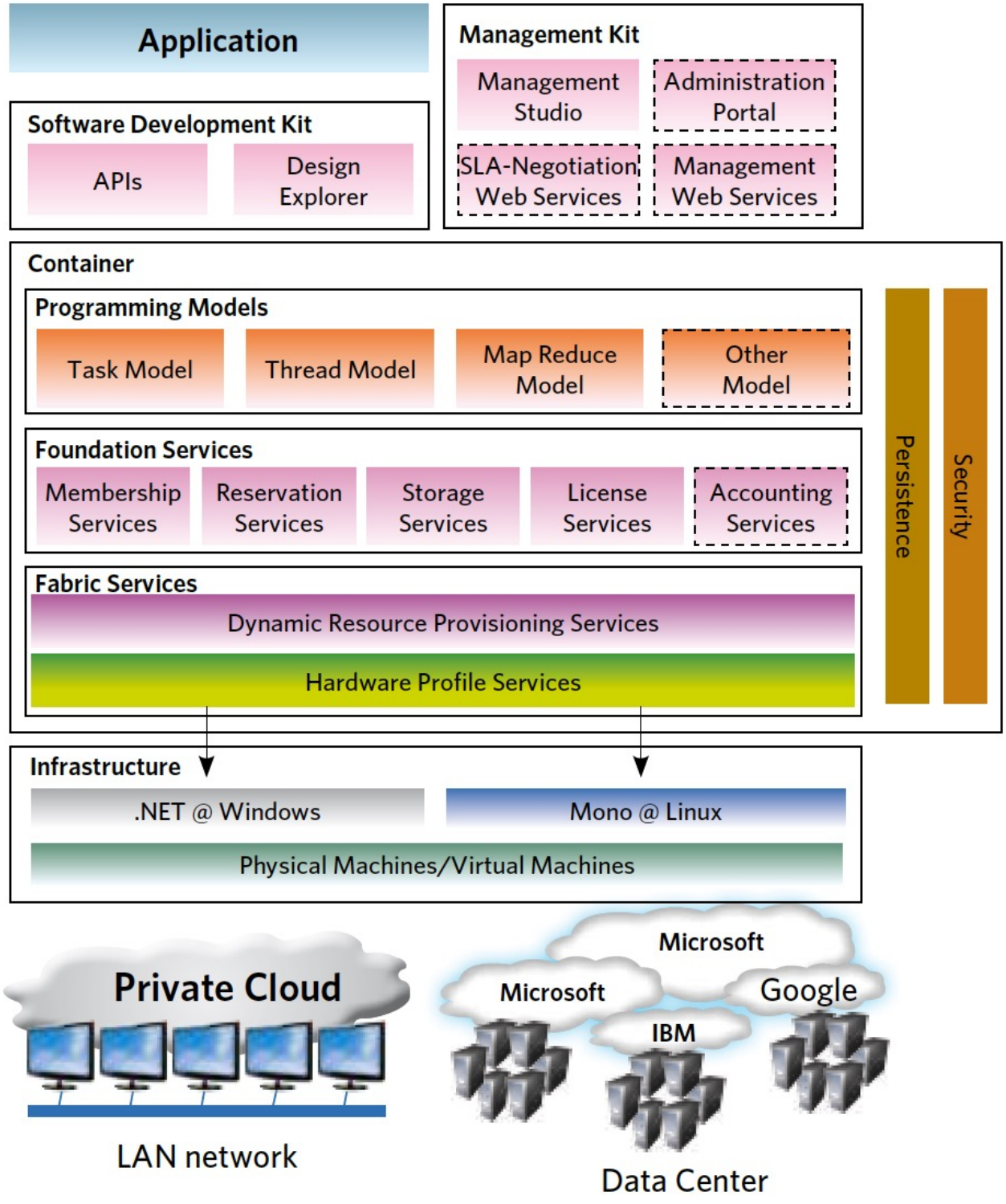


Aneka

platform and framework for developing distributed applications on the Cloud.
It harnesses the spare CPU cycles of a heterogeneous network of desktop PCs and servers or datacenters on demand

http://www.manjrasoft.com/aneka_architecture.html

Ranjan, Decentralized Overlay for Federation of Enterprise Clouds, *Handbook of Research on Scalable Computing Technologies*, 2009, IGI Global



Data-as-a-Service

Enterprise

Single users

Common needs:

- **Huge quantities of data**
- **Syncronization, collaboration**
- **Privacy and security**

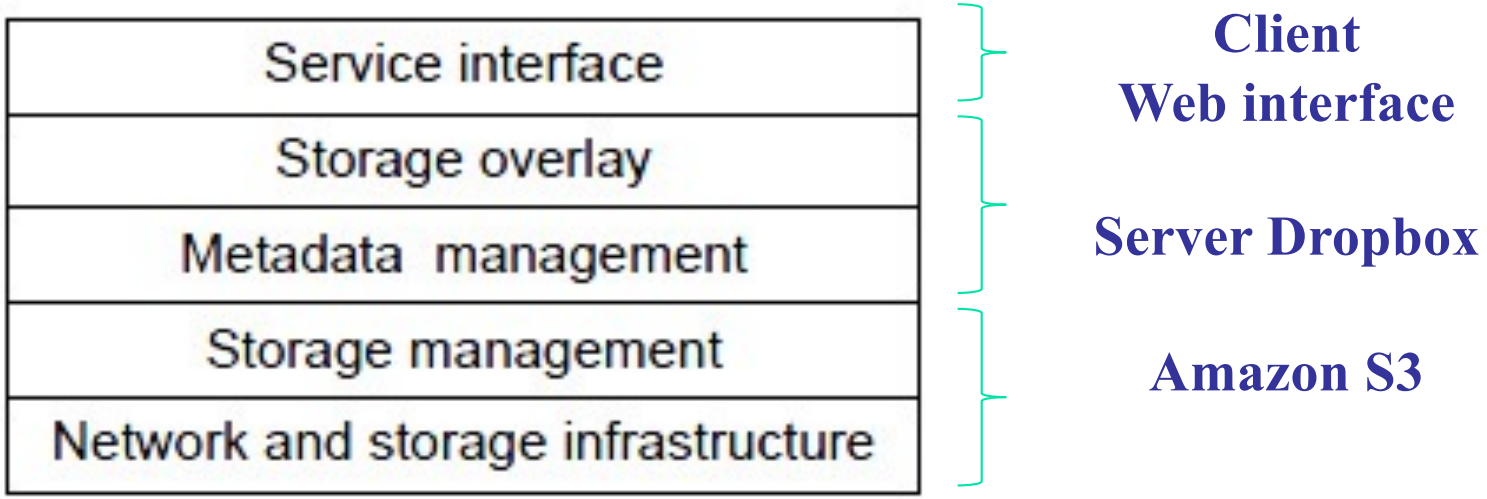
- Need flexibility
- No big initial investments
- Pay for use

- Each user owns everal devices
- Social services and sharing
- Usability, user-friendliness
- Enjoy free services

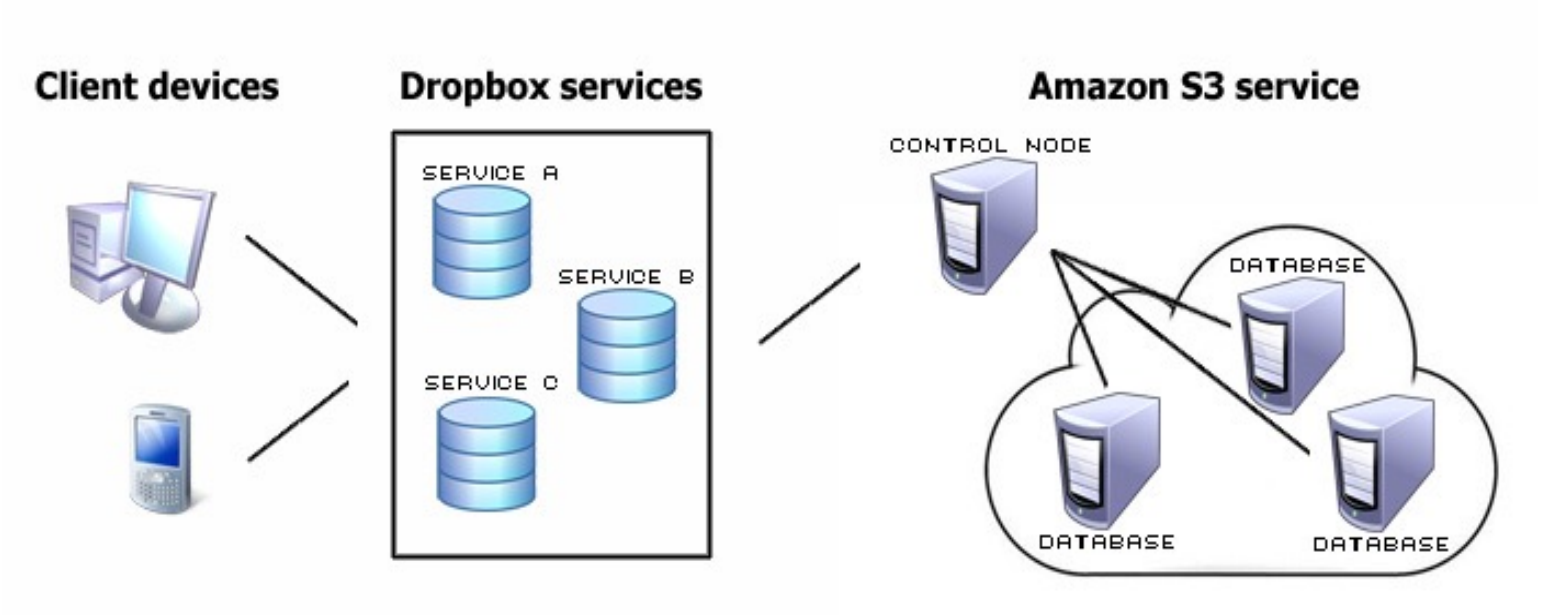
Storage utility

PRODUCT FEATURES	Dropbox	Google Drive	MS Skydrive	Apple iCloud
Free storage (GB)	2 up to 18	5	7 up to 25	5
Monthly price plans (GB)	50 for 10 $ 100 for 20 $ more customizable	25 for 2.49 $ 100 for 4.99 $ 200 for 9.99 $ 400 for 19.99 $	20 for 0.83 $ 50 for 2.08 $ 100 for 4.17 $	20 for 1.67 $ 40 for 3.33 $ 100 for 8.33 $
Max file size (GB)	0.3 (web) 2 (client)	10	2	0.025 (web) 2 (client)
Desktop apps	PC, Mac, Linux	PC, Mac	PC, Mac	Mac
Mobile apps	Android, iOS, Blackberry, Symbian	Andrian, iOS	Windows Phone, iOS	iOS

Dropbox architecture



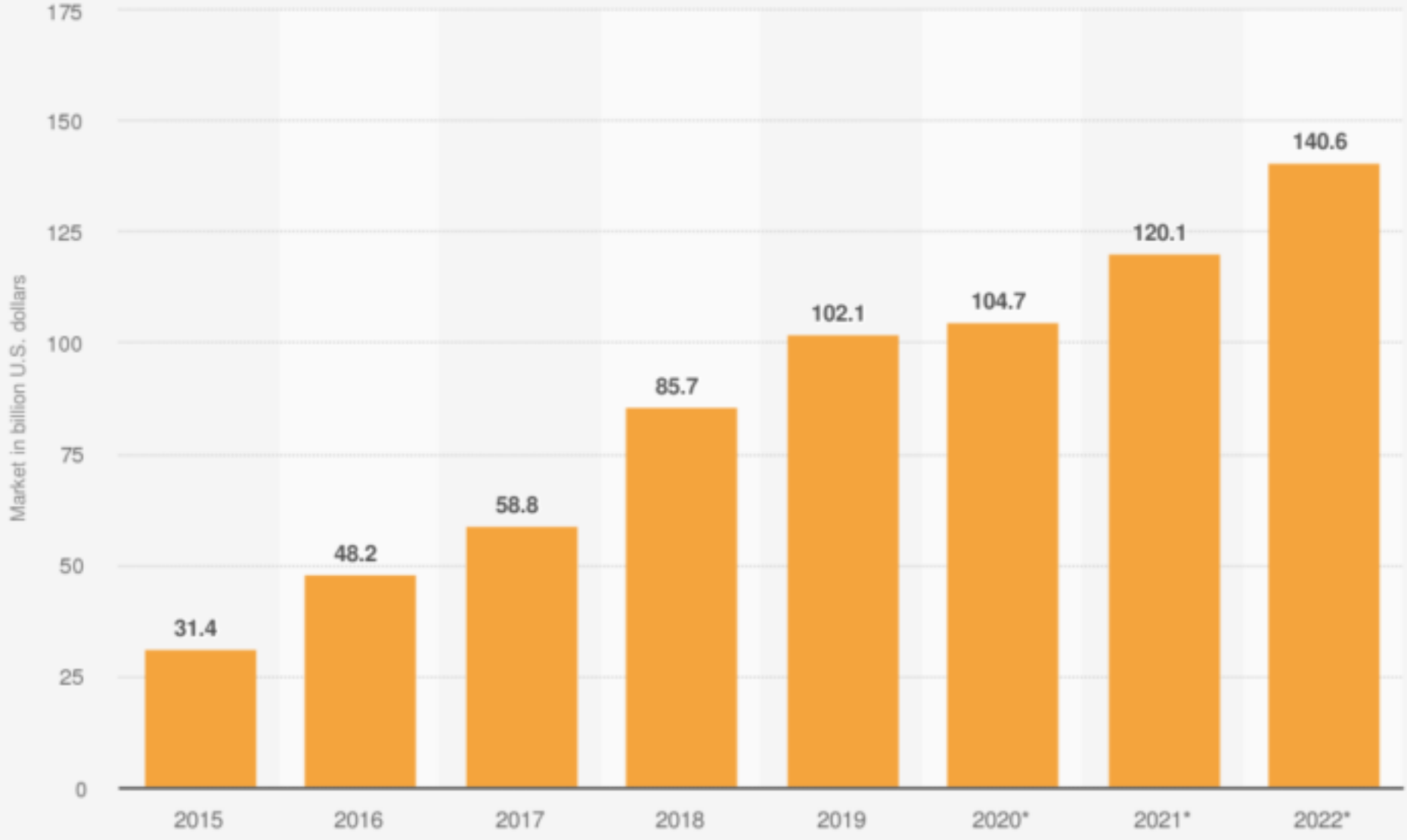
FONTE: [1]



Software-as-a-Service (SaaS)

- 1990: Web 0.9 (Tim Berners-Lee)
- 1995: Web 1.0 (some dynamic content, Netscape)
- 1997: Services (Google, e-commerce...)
- 1999: LoudCloud (first infrastructure for SaaS)
- 2000: Web 2.0 (rich UI's, social computing)
- 2001: Autonomic computing (IBM)
- 2004: SaaS & SOA (Google Maps, Amazon S3...)
- 2006: Amazon Web Services
- 2008: Cloud Computing (pay as you go)
- 2011: Microservices
- 2014: serverless - AWS lambda

Public cloud application services/software as a service (SaaS) market revenues worldwide from 2015 to 2022 (in billion U.S. dollars)



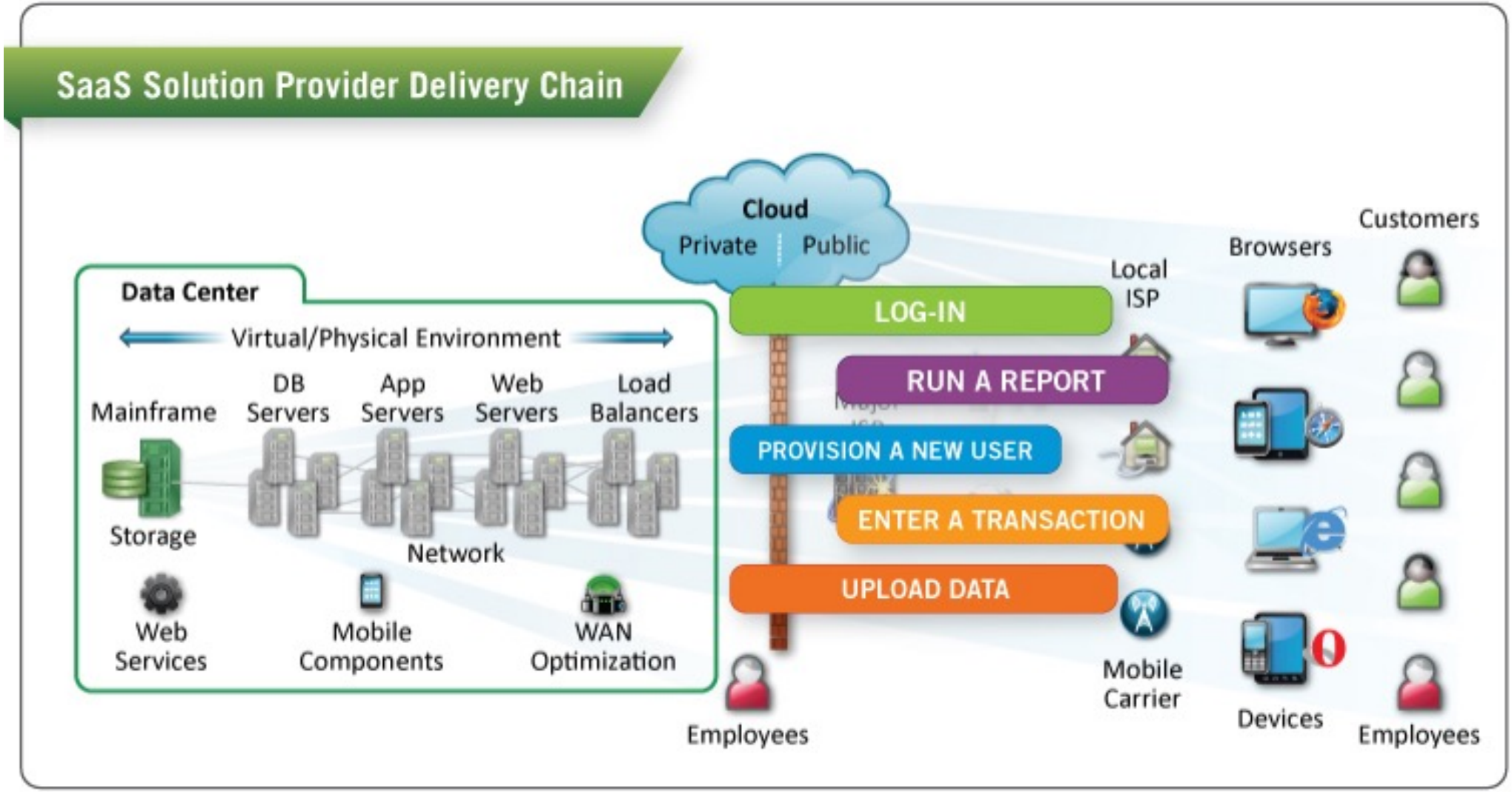
Source
Gartner

Additional Information:
Worldwide; 2015 to 2020

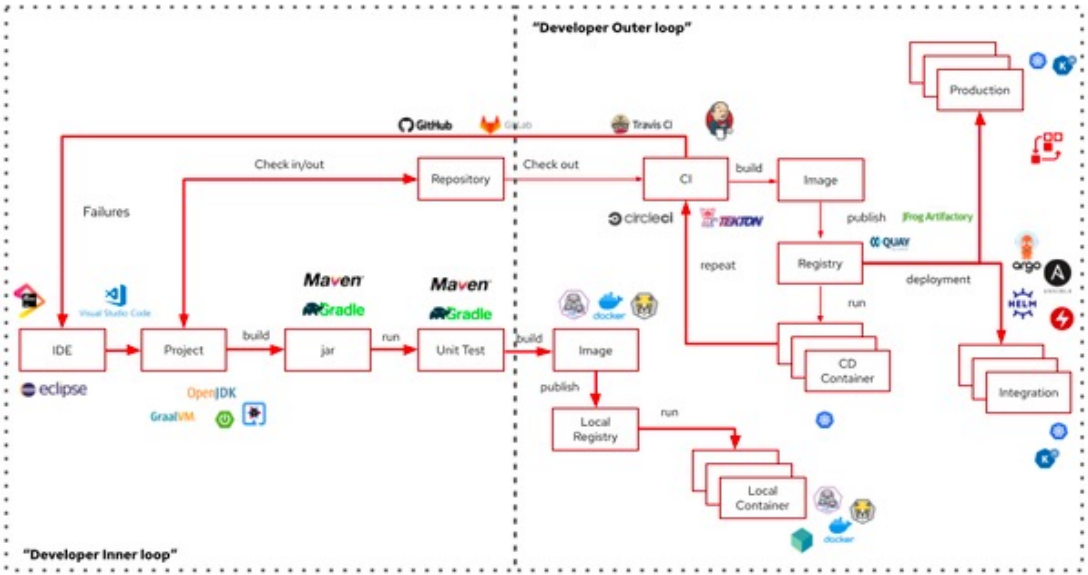
Software-as-a-Service (SaaS)

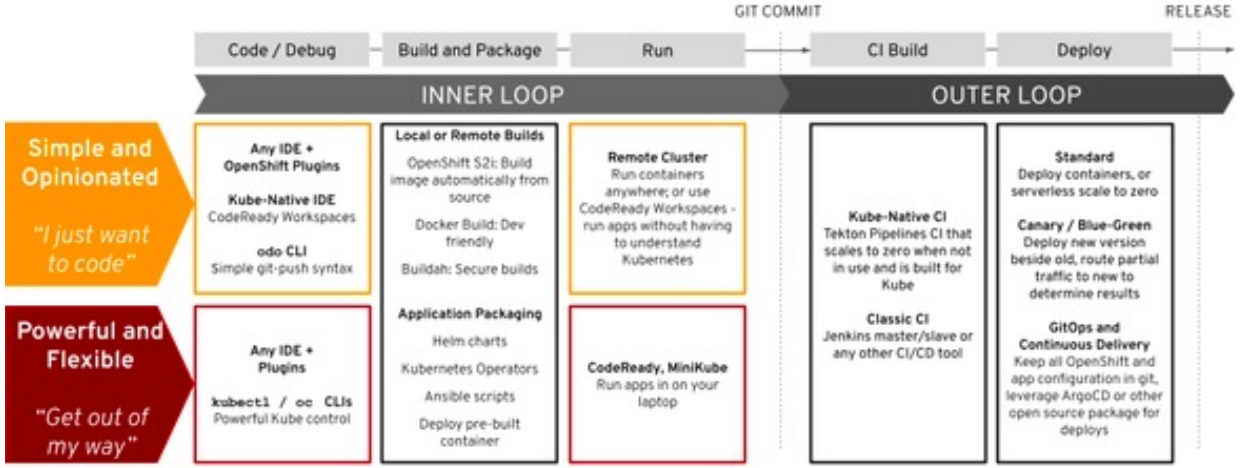
- Definition of SaaS: Internet-based deployment (for access and management) of commercially available software
- Service managed from “central” locations enabling customers to access applications remotely via web
- Delivery: one-to-many model (single instance, multi-tenant architecture) including architecture, pricing, partnering, and management characteristics
- Centralized updating, which obviates the need for end-users to download patches and upgrades
- Frequent integration into a larger network of communicating software—either as part of a *mashup* or a plugin to a platform as a *service*

SaaS delivery chain

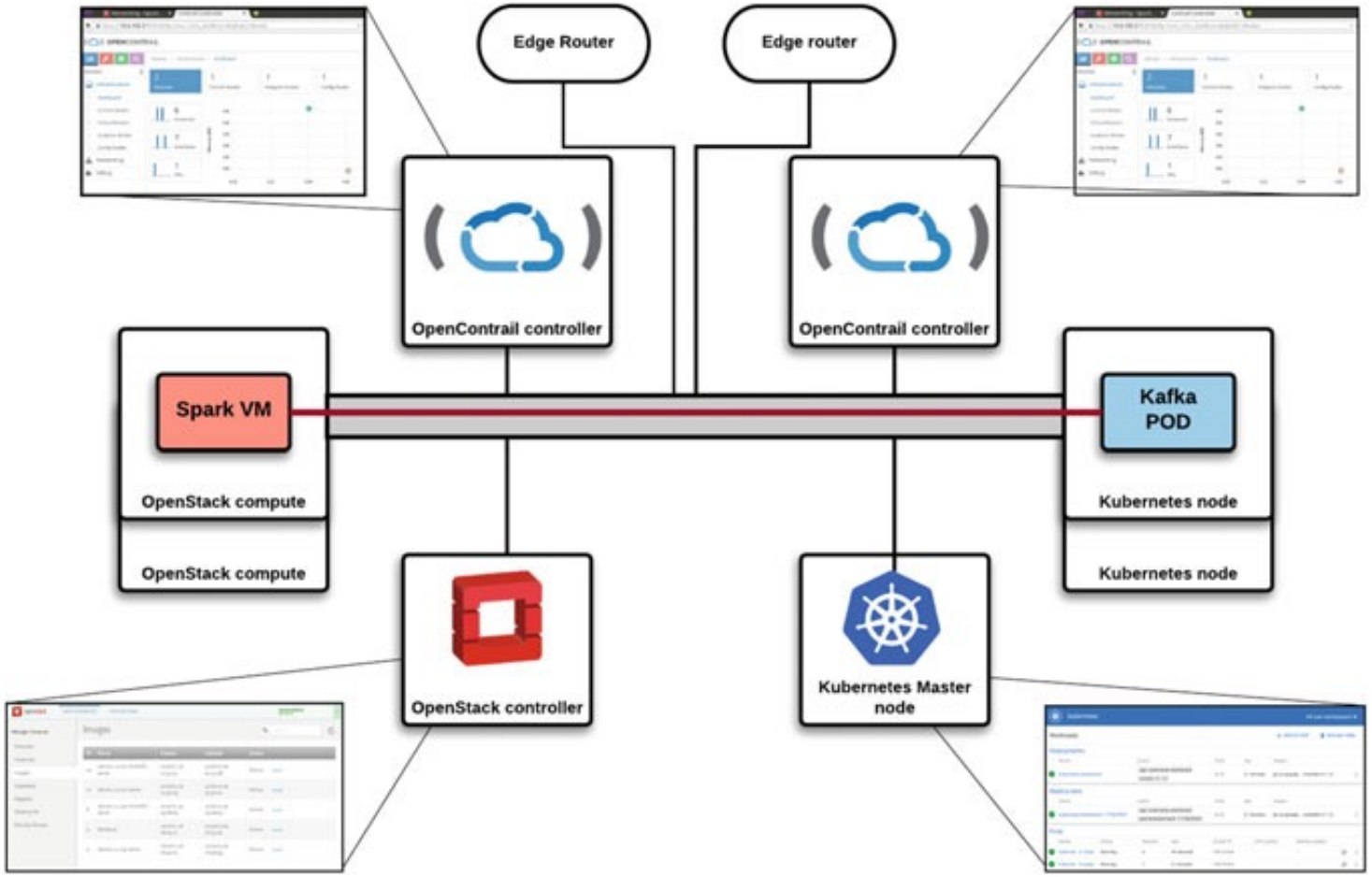


Source: Compuware


"Developer Outer loop"
GitHub
Travis CI
Check in/out
Repository
Check out
CI
build
Image
Production
Failures
circleci
TEKTON
publish
JFrog Artifactory
QUAY
repeat
Registry
deployment
argo
HELM
Maven
Gradle
Maven
Gradle
IDE
Project
build
jar
run
Unit Test
build
Image
run
CD Container
Integration
eclipse
OpenJDK
GraalVM
publish
Local Registry
run
Local Container
"Developer Inner loop"

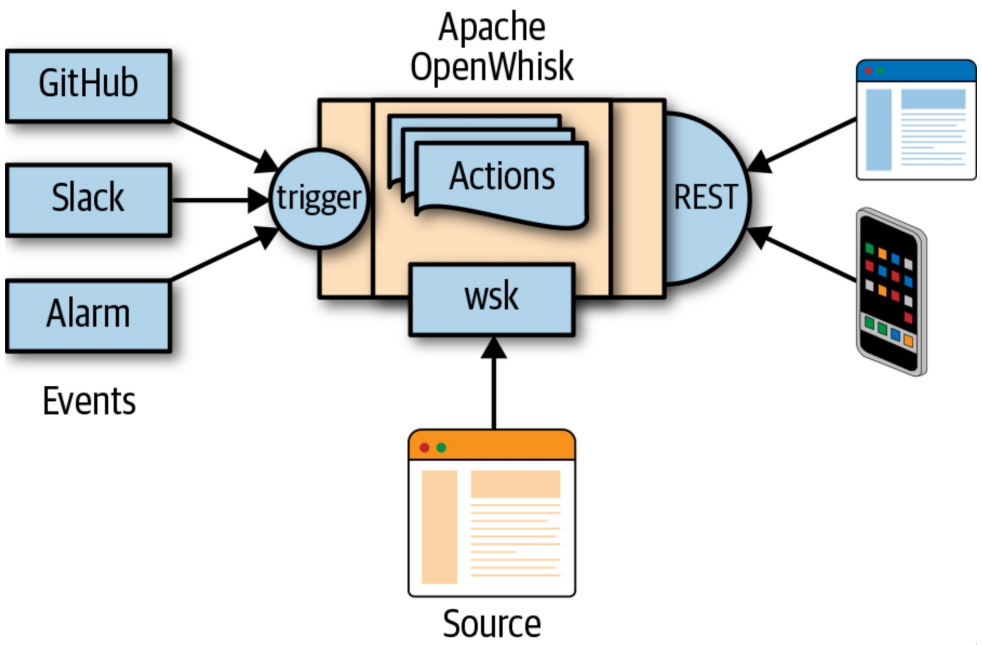

GIT COMMIT
RELEASE
Code / Debug
Build and Package
Run
CI Build
Deploy
INNER LOOP
OUTER LOOP
Simple and Opinionated
"I just want to code"
Any IDE + OpenShift Plugins
Kube-Native IDE
CodeReady Workspaces
odo CLI
Simple git-push syntax
Local or Remote Builds
OpenShift S2i: Build image automatically from source
Docker Build: Dev friendly
Buildah: Secure builds
Application Packaging
Helm charts
Kubernetes Operators
Ansible scripts
Deploy pre-built container
Remote Cluster
Run containers anywhere; or use CodeReady Workspaces - run apps without having to understand Kubernetes
Kube-Native CI
Tekton Pipelines CI that scales to zero when not in use and is built for Kube
Classic CI
Jenkins master/slave or any other CI/CD tool
Standard
Deploy containers, or serverless scale to zero
Canary / Blue-Green
Deploy new version beside old, route partial traffic to new to determine results
GitOps and Continuous Delivery
Keep all OpenShift and app configuration in git, leverage ArgoCD or other open source package for deploys
Powerful and Flexible
"Get out of my way"
Any IDE + Plugins
kubectl / oc CLIs
Powerful Kube control
CodeReady, MiniKube
Run apps in on your laptop

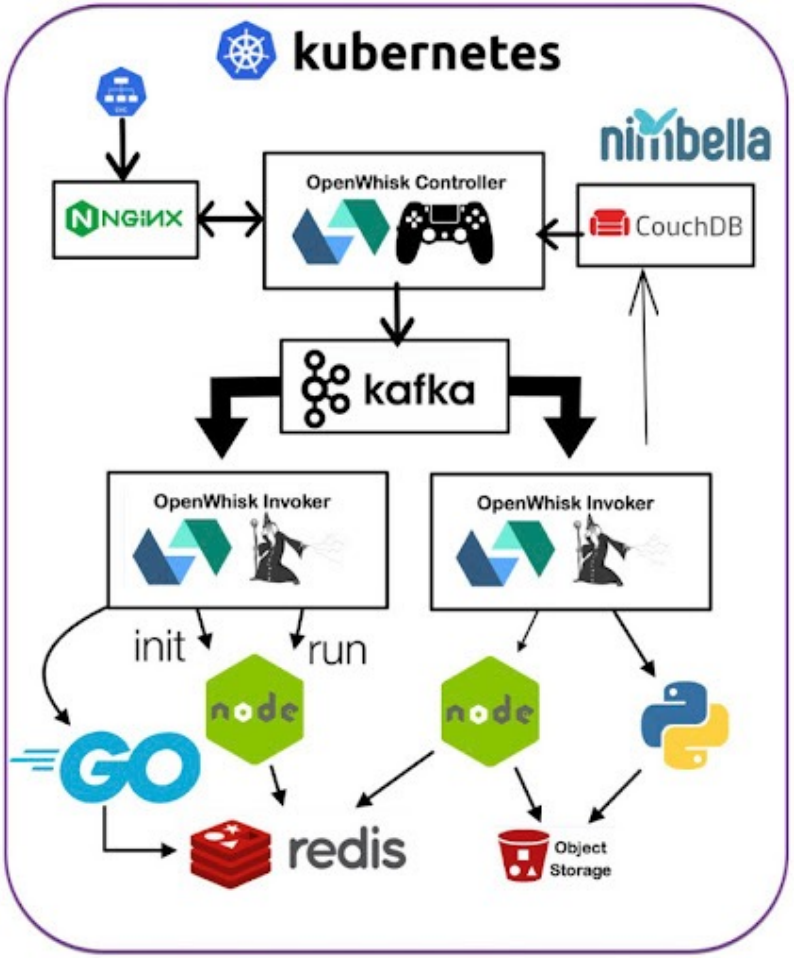
Some cloud sw stack



Raj, *Software defined cloud centers*, Springer 2018

Serverless





Apache OpenWhisk is an open source, distributed Serverless platform that executes functions (fx) in response to events at any scale. OpenWhisk manages the infrastructure, servers and scaling using Docker containers.

OpenWhisk supports a programming model in which developers write functional logic (called Actions), in any supported programming language, that can be dynamically scheduled and run in response to associated events (via Triggers) from external sources (Feeds) or from HTTP requests.

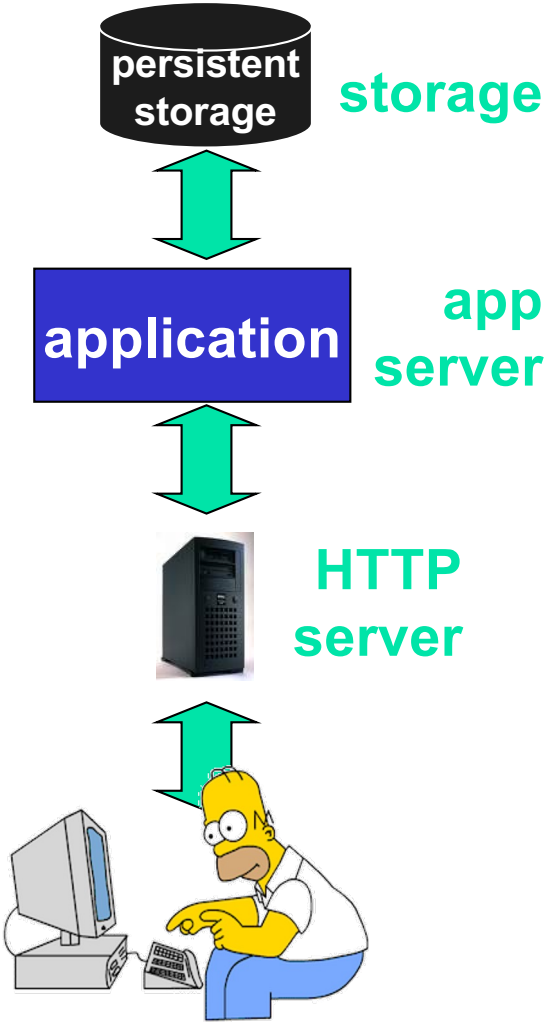
SaaS vs SOA

- SOA is an architecture for structuring a system as a set of decoupled services, whereas SaaS is hosting a set of software services over the Web
  - SaaS focuses on Software Hosted As A Service
  - SOA focuses on Software Designed As A Service.
- SaaS may be considered as a business model in which a consumer is involved; SOA is a design model with no assumption on the user
- all SaaS implementations follow the SOA concept. SaaS relies upon the Web, whereas SOA does not restrict its use on the web only
- SaaS means using software as a service over the Web using some protocol, which is used to communicate between the client side application and the server side software service.
- Traditionally, SaaS services use usually RESTful interactions, instead web services use RPC (Remote Procedure Call)

SaaS 3-tiers architecture

- *Common gateway interface* (cgi): allows a Web server to run a program
  - Server maps some URI's to application names
  - When the app is run, it gets the complete HTTP request including headers
- "Arguments" embedded in URL with "&" syntax or sent as request body (with POST)

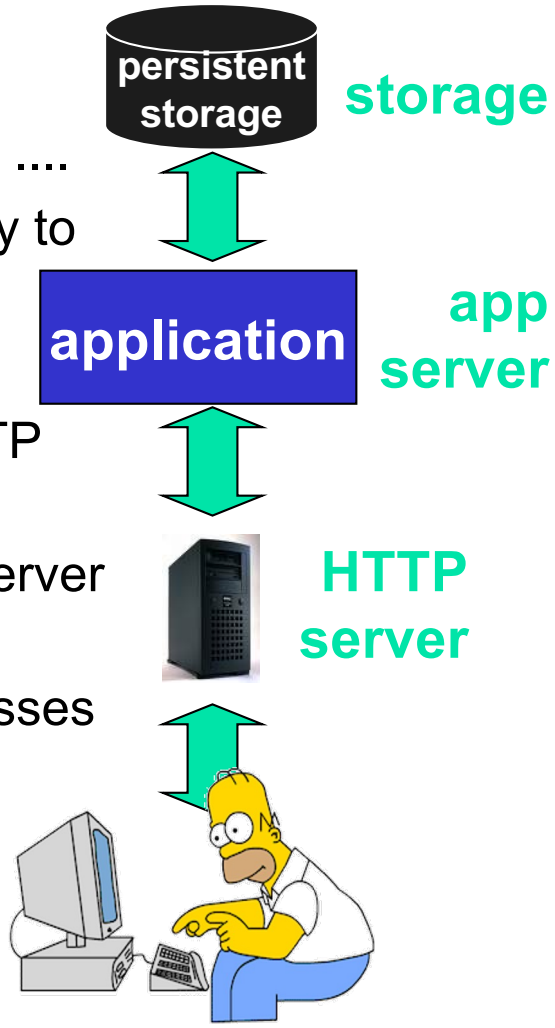
  http://www.foo.com/search?term=white%20rabbit&show=10&page=1
- App generates the entire response
  - content (HTML? an image? some javascript?)
  - HTTP headers & response code
- Plug-in *modules* for Web servers allow long-running CGI programs & link to language interpreters



• **Various *frameworks* have evolved to capture this structure**

SaaS 3-tiers deployment

- **HTTP server** ("web server")
  - "fat" (e.g. Apache): support virtual hosts, plugins for multiple languages, URL rewriting, reverse proxying, ...
  - "thin" (*nginx, thin,* Tomcat, ...): bare-bones machinery to support *one* language/framework; no frills
- **Application server**
  1. separate server process, front-ended by a "thin" HTTP server
  2. **or** linked to an Apache worker via FastCGI or web server plug-in: mod_perl, mod_php, mod_rails, ...
  - Apache can spawn/quiesce/reap independent processes
- **Persistent storage**
  - Common RDBMS (MySQL, PostgreSQL, etc.)
  - communicate w/app via proprietary or standardized database "connector" (ODBC, JDBC, ...)
- Hence **LAMP**: Linux, Apache, MySQL, PHP/Perl/Python



Discussion

SOA vs Clouds

Horizontal (SOA) vs. vertical (Clouds) services

1. SOAs focus mainly on business
   - each service may represent one aspect of the business
2. Clouds layered according to software stacks
   - the lower services support the upper ones to deliver applications

SOA vs Clouds

1. SOA concerns the *application architecture*
   - Components designed for specific roles in a SOA application
   - Design starts with a business problem and then abstract out the services
   - Services can be re-used by other applications in the future
2. Clouds are for the *delivery of IT infrastructures*
   - Services based on their roles in a software stack, therefore mostly well defined
   - The cloud services are domain- or problem- independent
   - The cloud services can be easily re-used by any application

SOA vs Clouds

1. SOA is mainly for enterprise computing
   - Cloud computing is internet-based services
2. SOA provides services of computing
   - Cloud computing provides computing of services
3. SOA applies the service principle to application software
   - Cloud applies the service principle at all levels: infrastructure, platform, and application level
4. Cloud is in many ways beyond SOA: implement Cloud, also get to keep SOA

Common issues

- Network dependence
  - Same structural weakness when the network is not performing or is unavailable
- Producer/ consumer model
- Share concepts of service orientation
- Services can be shared by multiple applications/users
- Demand minimum coupling among different parts of the system
  - any change on one part of the system has limited impact on the overall system
- SOA and clouds complement each other

Summary

- SOA define the Web as a powerful application and legacy integrator
- The standards to support SOA based on WebServices - such as XML, WSDL, and SOAP - are stable
- However, technologies to support quality attributes of SOA-WS, such as security, transaction efficiency, and availability are still evolving
- Clouds foster novel ecosystems: they require changes to the development, deployment, and operational processes (everything-as-a-service)
- Novel interesting research problems

Self test

- What is a hybrid cloud architecture?
- In which ways are clouds scalable?
- Which are the main components and connectors of clouds?
- Compare a SOA with a cloud architectural stack
- Which architectural issues and patterns are typical of cloud computing systems?

Readings

- Erl & Cope, *Cloud computing design patterns*, Prentice Hall 2015
- Bergmayr, *An Architecture Style for Cloud Application Modeling*, PhD th, TUWien 2016
- Ibryam, *Kubernetes Patterns: Reusable elements for cloud native applications*, OReilly, 2019

Relevant sites

- cloudcomputingpatterns.org
- docs.microsoft.com/en-us/azure/architecture/patterns/
- aws.typepad.com/
- developers.google.com/appengine/
- www.windowsazure.com
- www.armandofox.com/geek/teaching/
- radlab.cs.berkeley.edu

Journals and conferences

- IEEE Transactions on Cloud Computing
- Journal of Cloud Computing: Advances, Systems and Applications
- IEEE/ACM Int. Conf. on Utility and Cloud Computing
- IEEE Int. Conf. on Cloud Computing
- ACM Symp. on Cloud Computing
- ACM Cloud and Autonomic computing Conf.

Questions?