CLOUD AND SCIENTIFIC COMPUTING

Distributed Systems

4. Sem BSc Informatics

IMC FH Krems

OUTLINE

- Cloud computing
  - Introduction
  - Virtualization
  - Public/private/hybrid clouds
  - How to develop containerized applications
    - Docker
    - Kubernetes
- Scientific computing
  - Massive parallel processors (MPP)
  - Computing grids
  - Programming paradigms for parallel distributed computing
    - MPI
    - MapReduce
    - Apache Hadoop

OUTLINE

- **Cloud computing**
  - Introduction
  - Virtualization
  - Public/private/hybrid clouds
  - How to develop containerized applications
    - Docker
    - Kubernetes
- Scientific computing
  - Massive parallel processors (MPP)
  - Computing grids
  - Programming paradigms for parallel distributed computing
    - MPI
    - MapReduce
    - Apache Hadoop

CLOUD COMPUTING

- Distributed computing paradigm where data centers provide access to a pool of **virtualized** resources
  - CPU
  - Memory
  - I/O
- Hardware details are transparent to the user.
- Different types of workloads:
  - Batch-style backend jobs.
  - Interactive applications.
- Big, complex, scalable data centers using commodity hardware (x86, low-cost terabyte disks, Ethernet networks)

CLOUD COMPUTING

Requirements for internet clouds:

- Rapid provisioning of resources (virtual environments).
- Self-recovering.
- Highly scalable.
- Security and separation between tenants.
- Real-time monitoring
  - Resource allocation.
  - Rebalancing.
- Enabling technology: **virtualization.**

CLOUD COMPUTING

Objectives for computing clouds:

- Shift from desktop to **data centers**
  - Newest trend → Cloud to Edge
- Service **provisioning** and **pricing**: consumers and end-users sign SLAs. Pricing based on usage.
- **Scalability** and **performance**: must scale with number of users and maintain performance
- Data **privacy**
- **Interoperability** and **openness**: use of standards between multiple providers
- Q: Advantages of cloud computing for software development?

CLOUD COMPUTING

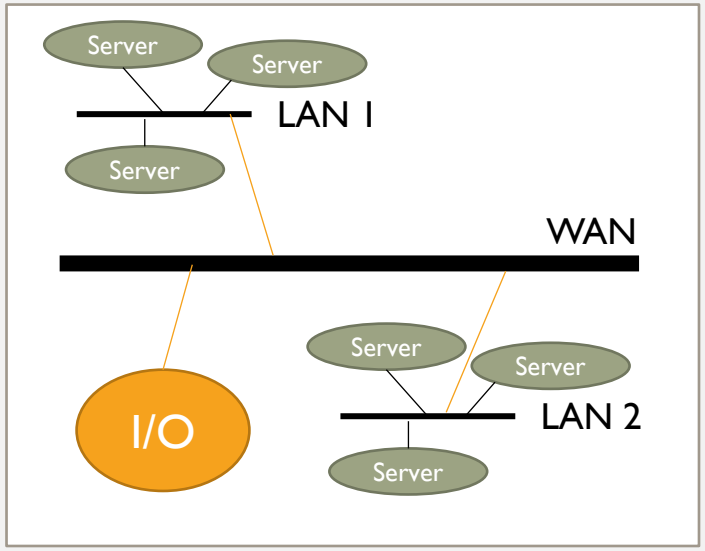
Different cloud-computing models:

- **Infrastructure as a Service** (IaaS): provides computing resources (servers, networks, storage, virtual data centers). User requests and releases resources, which are managed by cloud provider.
- **Platform as a Service** (PaaS): provides runtime and deployment environments for user-built applications (middleware, databases, development-tools)
- **Software as a Service** (SaaS): applications hosted in the cloud (business applications, CRM, HR, etc). Hosting and maintenance are oursourced.
- **Function as a Service** (FaaS): ”serverless” architectures. User can deploy applications without building/maintaining infrastructure

CLOUD COMPUTING

How are clouds built?

- Commodity hardware (x86 servers, low-cost storage, switches).
- Join computing nodes into *clusters*.
  - Hierarchical construction.
  - SAN, LAN, WAN → increasing number of nodes.
- **Virtualization:** servers in data centers are VMs.



VIRTUALIZATION

- Clouds need large amounts of computing resources.
- Need to aggregate resources in a virtualized manner.
- Need to provision resources quickly and dynamically.
- Advantages
  - Maximize resource utilization.
  - Application flexibility.
  - Software manageability.
  - Security (isolation).

VIRTUALIZATION

Lowest level

Highest level

Levels of virtualization

- **Instruction set architecture (ISA):** a given instruction set is emulated by the instruction set of the host machine (binary translation). Example: IA32-EL.
- **Hardware abstraction level (HAL):** performed on top of the hardware by generating a virtual hardware environment for a virtual machine. Example: WMWare ESXi.
- **Operating System level:** using containers/VMs on the same physical servers. Containers share a common kernel but behave as separate servers with own file systems, users and settings. Examples: Virtualbox, Docker.
- **Library support level:** virtualized interface calls (API) are hooked and translated between different architectures. Example: WINE.
- **User-application level:** high-level language VMs like JVM and the .NET CLR.

VIRTUALIZATION

Hardware abstraction and **Operating System** are the most important virtualization levels for cloud computing.

- Hosted virtual machines provided on top of a host OS (most popular approach).
- VMs are easy to migrate.
- VMs are isolated from one another (security).
- Provisioning of full VMs can be costly and storage can be an issue
  - Each VM creates its image from scratch
  - VMs must be provisioned quickly
  - Storage: abundant repeated content
- Using **containers** both problems can be addressed
  - Containers are more efficient to start
  - Containers are more storage efficient

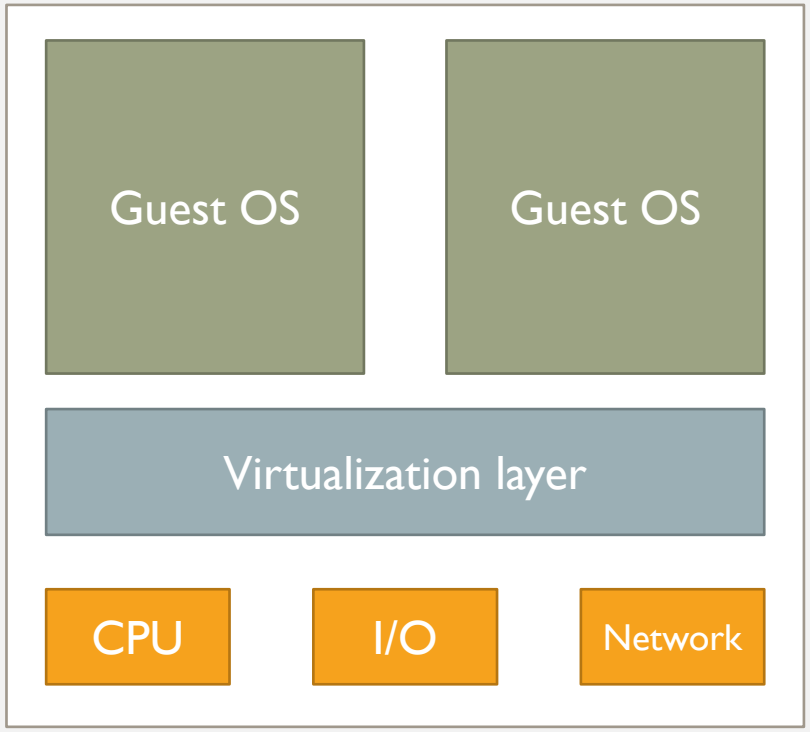
VIRTUALIZATION

What is a container?

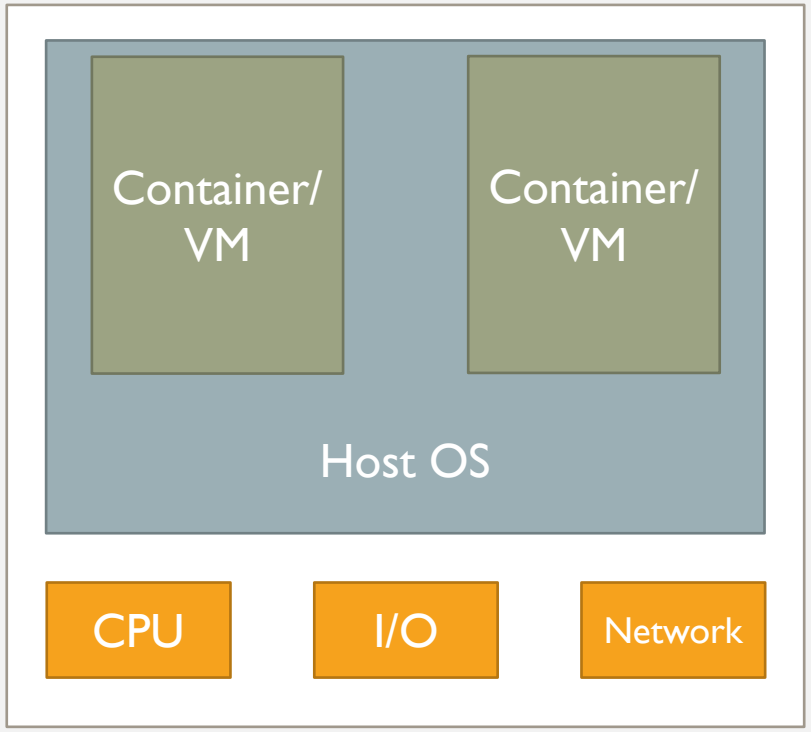
- Isolated VM within a single OS kernel.
- Virtualization layer *inside* the OS to partition physical resources.
- Container looks like a real server from an outside observer‘s point of view.
  - Own processes, files, settings, networking interfaces, user accounts, etc.
- Virtualization layer *inside* the OS to partition physical resources.
- IaaS → based upon VMs.
- PaaS → frequently based upon containers.

VIRTUALIZATION

Hardware virtualization (bare-metal - ESXi)



OS level virtualization (Virtualbox, Docker)



CLOUD COMPUTING

Types of clouds:

- **Public** clouds.
- **Private** clouds.
- **Hybrid** clouds.

PUBLIC CLOUDS

- Accesible over the Internet.
- Distributed geographically
  - Fault tolerance.
  - Response latency reduction.
  - Other (legal) reasons.
- Subscription-based services.
  - Price-per-use basis.
- Google Cloud, Amazon Web Services (AWS), Microsoft Azure.
- Proprietary API for accessing the provided services.

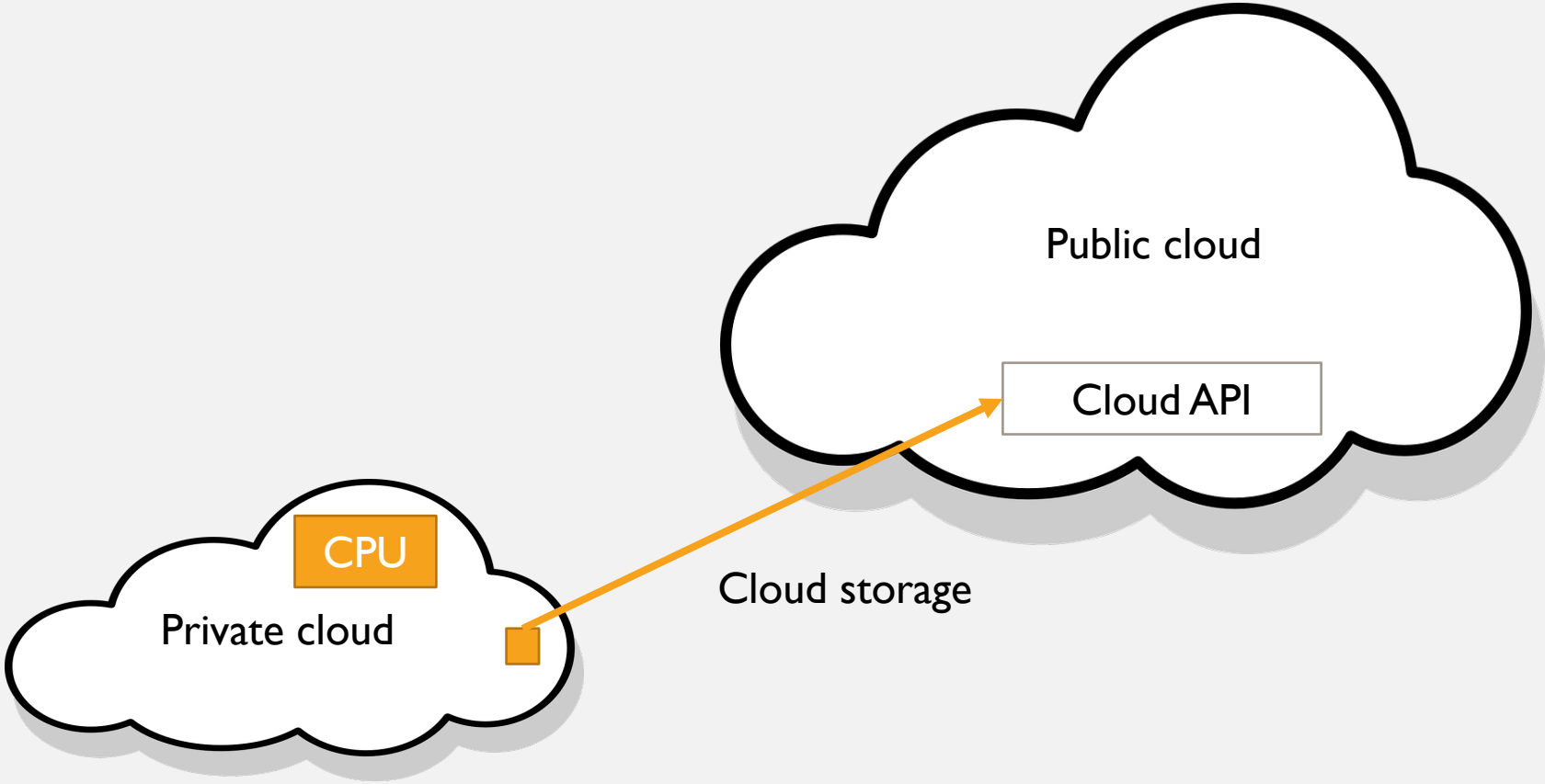
Google Cloud

Azure

PRIVATE CLOUDS

- Owned by an organization inside their own intranet.
- Managed by the organization.
- Limited access.
- Private infrastructure for providing flexibility and run service workloads.
- More customizable.
- Data protection.
- More control.

HYBRID CLOUDS



CLOUD COMPUTING

Who are the users of cloud computing services?

- **Individual users:** mostly use services provided by application providers (Saas).
- **Organizations:** typically PaaS and IaaS users, interact directly with cloud platform providers and cloud infrastructure providers.

CLOUD COMPUTING

Main players in the cloud computing industry:

- **Google:** offers PaaS and SaaS services for application development with **Google Cloud**
- **Amazon:** offers IaaS and PaaS services in the context of **Amazon Web Services (AWS)**
- **Microsoft:** offers IaaS, PaaS and SaaS services provided by **Azure**.

Google Cloud

CLOUD COMPUTING

Google Cloud

Google Cloud – cloud.google.com

- PaaS and SaaS provider.
- Combines Google services to build scalable/HA applications using common languages and frameworks like Java, Ruby, Python, C#, Node.js
  - Google File System (GFS): storage for large amounts of data.
  - MapReduce: distributed computing framework (batch).
  - BigTable: storage service for structured data (NoSQL).
  - BigQuery: serverless data warehouse (analytics).
  - Chubby: distributed application lock services.
- Pay per use model.

CLOUD COMPUTING

Google Cloud – cloud.google.com

- Service free within quota for Google account holders (virtually everyone).
- Cloud SDK for local development and interaction with the cloud services.
- Examples of Google Cloud applications.
  - Google search.
  - Gmail.
  - Drive.
  - Maps.

Google Cloud

CLOUD COMPUTING

Amazon Web Services (AWS) – aws.amazon.com

- Provides IaaS and PaaS services
- Main building blocks:
  - Elastic Compute Cloud (EC2): virtualized computing platform.
  - Simple Storage Service (S3): object based large scale storage system.
  - Relational Database Service (RDS): relational database in the cloud (PostgreSQL, MariaDB, MySQL, Oracle, MS-SQL Server).
  - Lambda: serverless computing (FaaS, PaaS)
  - Simple Queue Service (SQS): message-oriented middleware for loosely coupled distributed applications

CLOUD COMPUTING

Amazon Web Services (AWS) – aws.amazon.com

- Elastic Load Balancing (ELB): distribute incoming traffic between e.g. EC2 instances, avoiding non-operating nodes and evenly distributing load.
- CloudWatch: integrated application performance monitoring (APM)
- … and many more

CLOUD COMPUTING

Microsoft Azure – azure.microsoft.com

- SaaS, PaaS and IaaS services.
- Mostly .NET and Windows OS workloads, but other frameworks/OSes (Linux) possible.
- SDK and Azure platform can also be run locally.
- Azure Virtual Machines: VM provisioning (Windows, Linux).
- Azure Functions: serverless computing.
- App Service: web application development and deployment
  - Common frameworks and programming languages (not only .NET family).
  - Migration service for older .NET applications.

CLOUD COMPUTING

Microsoft Azure – azure.microsoft.com

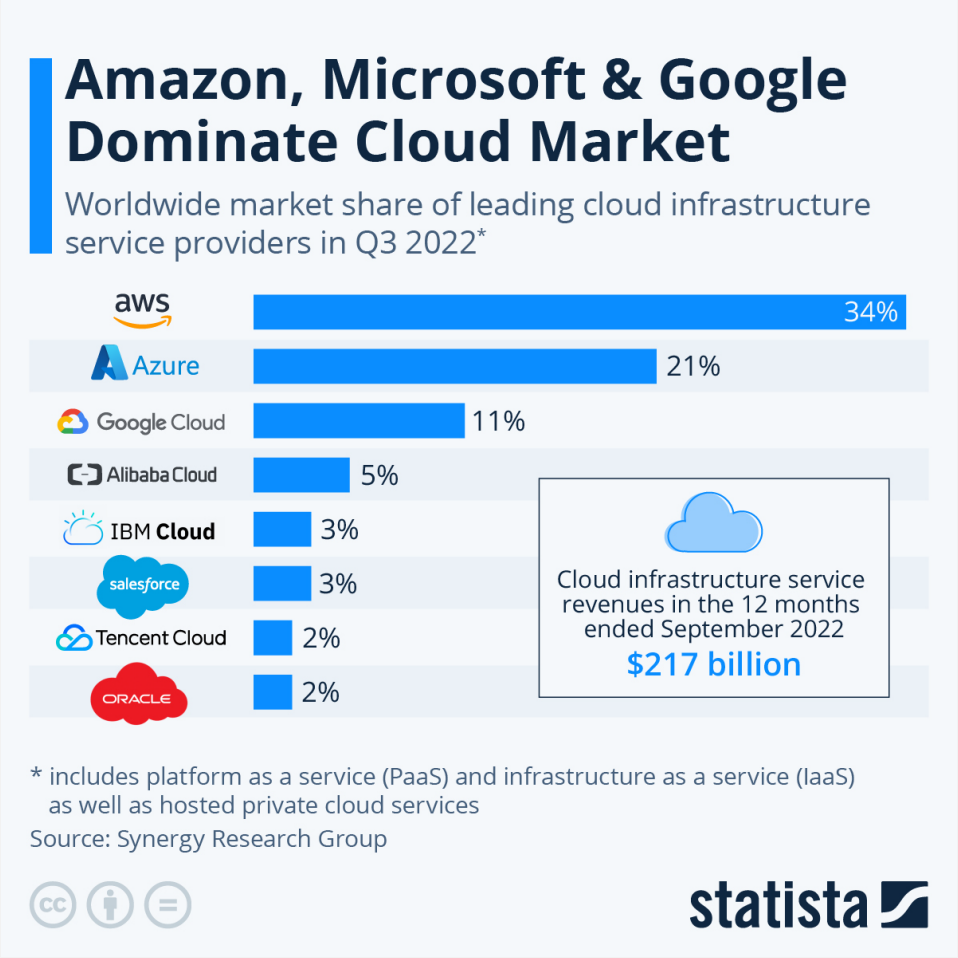
- Active Directory: SSO and authentication service.
- Spring Cloud: support for Spring Boot applications.
- Container Instances: deploying containers directly to the cloud.
- **Cognitive and AI services**
  - Language models
  - Open AI APIs

CLOUD COMPUTING

European Clouds

- Azure, AWS and Google Cloud offer server locations all over the world (including EU).
- Current status (GDPR)
  - Transfer of personal data to datacenters owned by US companies.
  - Privacy Shield was struck down by court (European Court of Justice, 2020) after allegations presented by Austrian activist Max Schrems.
  - Full compliance with standard contractual clauses.
  - Proposal from EU-Commission: EU-US Data Privacy Framework.
- FugaCloud: IaaS.
- Hetzner: IaaS.

CLOUD COMPUTING



CLOUD COMPUTING

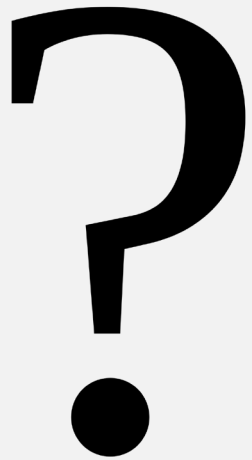
Cloud platform	IaaS	PaaS	FaaS	SaaS
Google Cloud	Compute Engine, Storage	App Engine, Kubernetes Engine, BigTable, BigQuery, Cloud SQL	-	Search, Gmail, Drive, Docs, Maps, Earth
AWS	EC2, S3	SQS, RDS, ELB	Lambda	-
Azure	Virtual Machines	App Service, SQL	Functions	Office

CLOUD COMPUTING

How to build applications for the cloud?

1. Develop/deploy using proprietary PaaS frameworks.
2. Develop/deploy using VMs.
3. Develop/deploy using containers.

What option would you use?



CLOUD COMPUTING

1. **Develop/deploy using proprietary PaaS frameworks**

Pros:

- More productivity and efficiency.
- No need for hardware maintenance.
- No need for development platform maintenance.

Cons:

- Vendor dependency.

CLOUD COMPUTING

2. **Develop/deploy using VMs**

Pros:

- Vendor independency.
- Switch to other cloud providers possible.
- Computing resources flexibility → different VMs for different workloads.
- No hardware maintenance.

Cons:

- Provisioning VMs not efficient.
- Software setup maintenance (OS, runtimes, frameworks, dependencies).
- HA/Fault tolerance/scalability has to be solved manually.

CLOUD COMPUTING

3. **Develop/deploy using containers**

Pros:

- Vendor independency.
- Switch to other cloud providers possible.
- Computing resources flexibility.
- No hardware maintenance.
- Can be provisioned quickly.
- Use container images.
- HA/Fault/scalability tolerance using existing middleware solutions (i.e. Kubernetes).

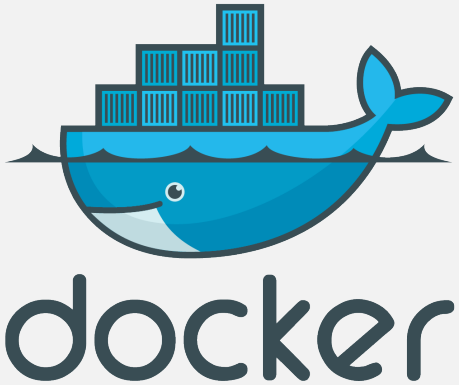
Cons:

- Container image maintenance.

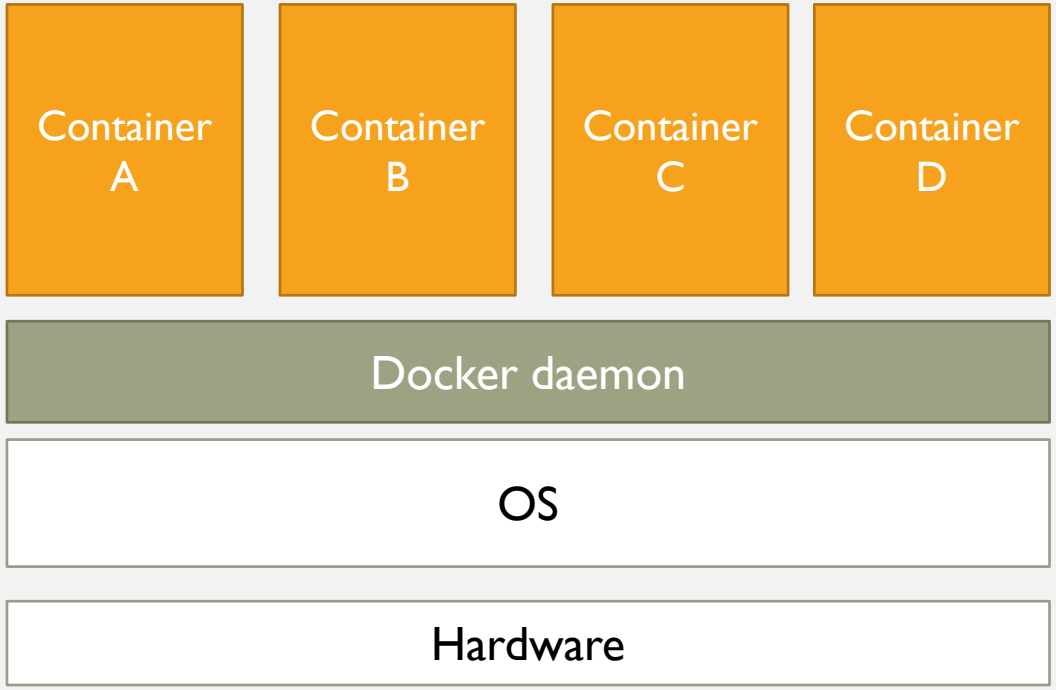
CLOUD COMPUTING

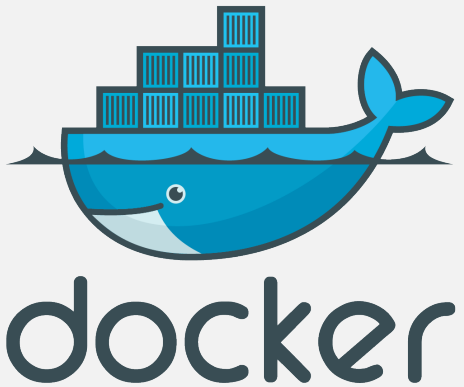
Docker

- Most established container technology.
- OS-level virtualization.
- Containers are isolated from one another and expose a given application functionality.
- Application runs in a standard environment which can be deployed on Linux, Windows and Mac hosts.
- Services to build and host container images (Docker Hub).



CLOUD COMPUTING

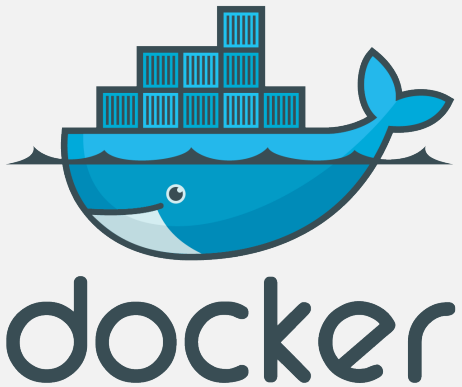




CLOUD COMPUTING

Docker

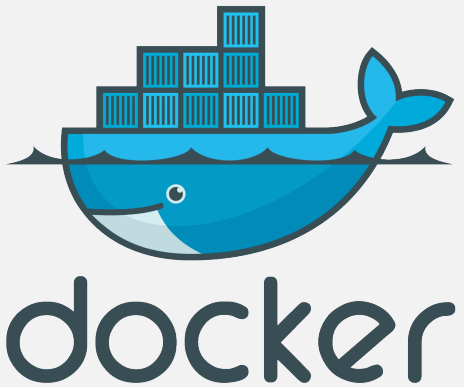
- Docker daemon (dockerd): process that manages containers and communicates with user.
  - Docker (client): communicates with dockerd via command line interface.
  - Start/stop/build containers.
- Image: template to build containers (instances).
  - Smaller than containers.
- Repositories: contain versionized images to be pulled and built.
- Port bindings: mapping ports in the host to ports in the container.
  - Example: "map port 80 in the host to port 8080 in the container".
- Dockerfiles (Exercises).



CLOUD COMPUTING

Docker

- Compose: tool for defining multi-container deployments.
- Deployments can be defined using YAML files.
- Different deployments for development/staging/production.

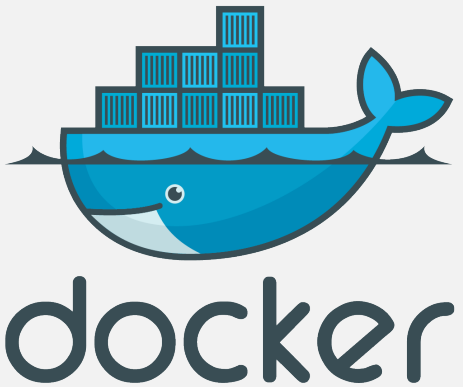


- Exercises

CLOUD COMPUTING

Docker

- What do we need to do before going into production?



CLOUD COMPUTING

Kubernetes (k8s)

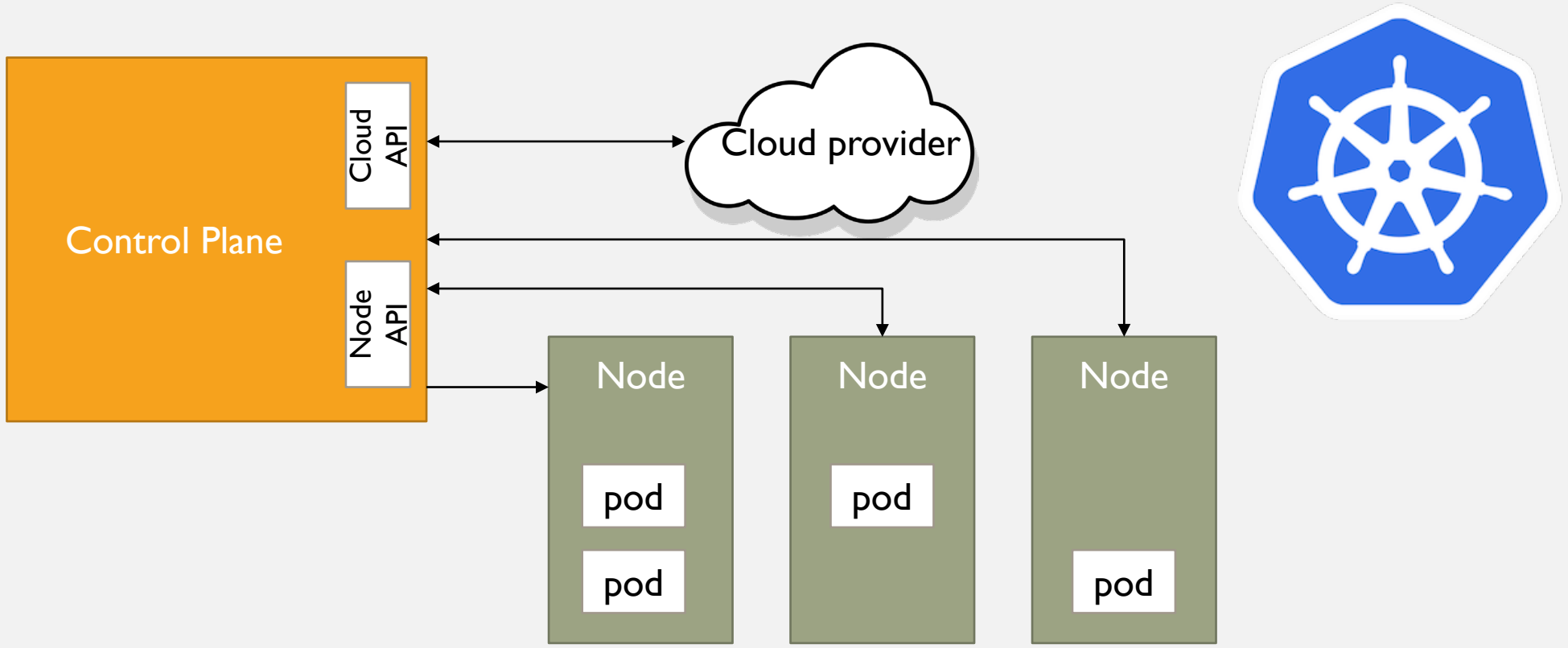
- Widely established container orchestration framework.
- Automatic management, deployment, scaling and fault tolerance for containerized applications.
- Application can be split between different containers.
  - One container per service.
- Logical unit → **pod**.
- Pods are scheduled on nodes.
- Groups of pods → services.
- Container runtime (i.e. docker) needed.

CLOUD COMPUTING

Kubernetes (k8s)

- Nodes form a *cluster*.
- Each *pod* has a unique IP inside the cluster.
- Control plane (apiserver) listens for remote connections on a secure port (typically 443 - HTTPS).
  - Communicates with worker nodes (kubelet).
  - Schedules replicas.
  - Communicates with cloud provider API.

CLOUD COMPUTING



CLOUD COMPUTING

Kubernetes (k8s)

Usage scenarios:

- Handle load peaks: Kubernetes can scale automatically after reaching a predefined measure (e.g. CPU load).
- Fault tolerance: Kubernetes can automatically restart a service after noticing non-responsiveness (health-checks).
- High availability: service update applications with zero downtime (rolling update).

CLOUD COMPUTING