

# A-Z

## API

Application Program Interfaces are methods for programs to exchange data without the involvement of humans. APIs often involve a schematic description of how the data looks like and where it needs to be sent. Today many APIs are based on [HTTP](#).

## Container

Containers are discussed in [lecture 4](#) and provide a way to package an application with its runtime requirement in a self-contained manner. Containers also provide limited security isolation.

## Container Orchestrator

A container orchestrator is responsible for managing containers across multiple virtual or physical servers. When a server fails or a container crashes the orchestrator is responsible for scheduling the container on a different server. Some orchestrators also manage integration with the cloud. Orchestrators are discussed in [lecture 4](#)

## CDN

A content delivery network replicates content across the globe to bring the content closer to the end user. CDNs are discussed in [lecture 3](#).

## DBaaS

Databases as a Service are discussed in [lecture 3](#) providing managed databases to a cloud customer without needing to manage a server, or the database system.

## Docker

[Docker](#) is one of the earliest modern containerization systems. Docker was largely responsible for promoting the immutable infrastructure concept in containers.

Docker also contains the Docker Swarm orchestrator, a very simple system to manage containers across multiple machines.

Docker is discussed in [lecture 4](#) and [exercise 3](#).

## FaaS

Functions as a Service are a concept discussed in [lecture 3](#) that allow a developer to run program code without needing to manage a server or a runtime environment.

## Grafana

Grafana is an open source graphing dashboard. It is discussed in [exercise 5](#).

## HTTP

The Hypertext Transfer Protocol is the protocol that powers the world wide web allowing for the easy up- and download of data. With HTTP each file (resource) has a unique URL on a server and can be linked.

## IaaS

Infrastructure as a Service is discussed in [lecture 2](#) and provides virtual machines and related services to cloud customers.

## IOMMU

The IOMMU is the memory management unit that virtualizes the Direct Memory Access (DMA) requests from I/O components and separates them from each other to prevent attacks over Thunderbolt/USB.

## Istio

[Istio](#) is a service mesh used in conjunction with Kubernetes to facilitate microservices. Istio is discussed in [lecture 5](#)

## Kubernetes

[Kubernetes](#) is an advanced, and thus complex container orchestrator. It not only manages containers, it also contains a large number of cloud integrations for storage, networking, load balancers, autoscaling, and more. Kubernetes is discussed in [lecture 4](#)

## Load balancer

Load balancers provide either network or application level traffic distribution across multiple servers. They are discussed in [lecture 2](#) and [lecture 3](#)

## Microservices

Microservices are a concept discussed in [lecture 5](#) for creating multiple small applications working together across the network.

## MMU

The Memory Management Unit is the component of the CPU that translates virtual memory addresses to real addresses. It is used to separate applications from each other.

## PaaS

Platform as a Service is a collection of services discussed in [lecture 3](#) that give a developer the ability to deploy an application without needing to manage IaaS servers.

## Prometheus

Prometheus is an open source, cloud native metrics collection system (time series database). It is discussed in [lecture 5](#) and [exercise 4](#).

## Rack

A closet with standardized mounts for servers.

## Router

A network device that forwards layer 3 (IP) packets between separate networks.

## Switch

A network device that forwards Ethernet frames (packets) between devices. It does not perform layer 3 (IP) routing.