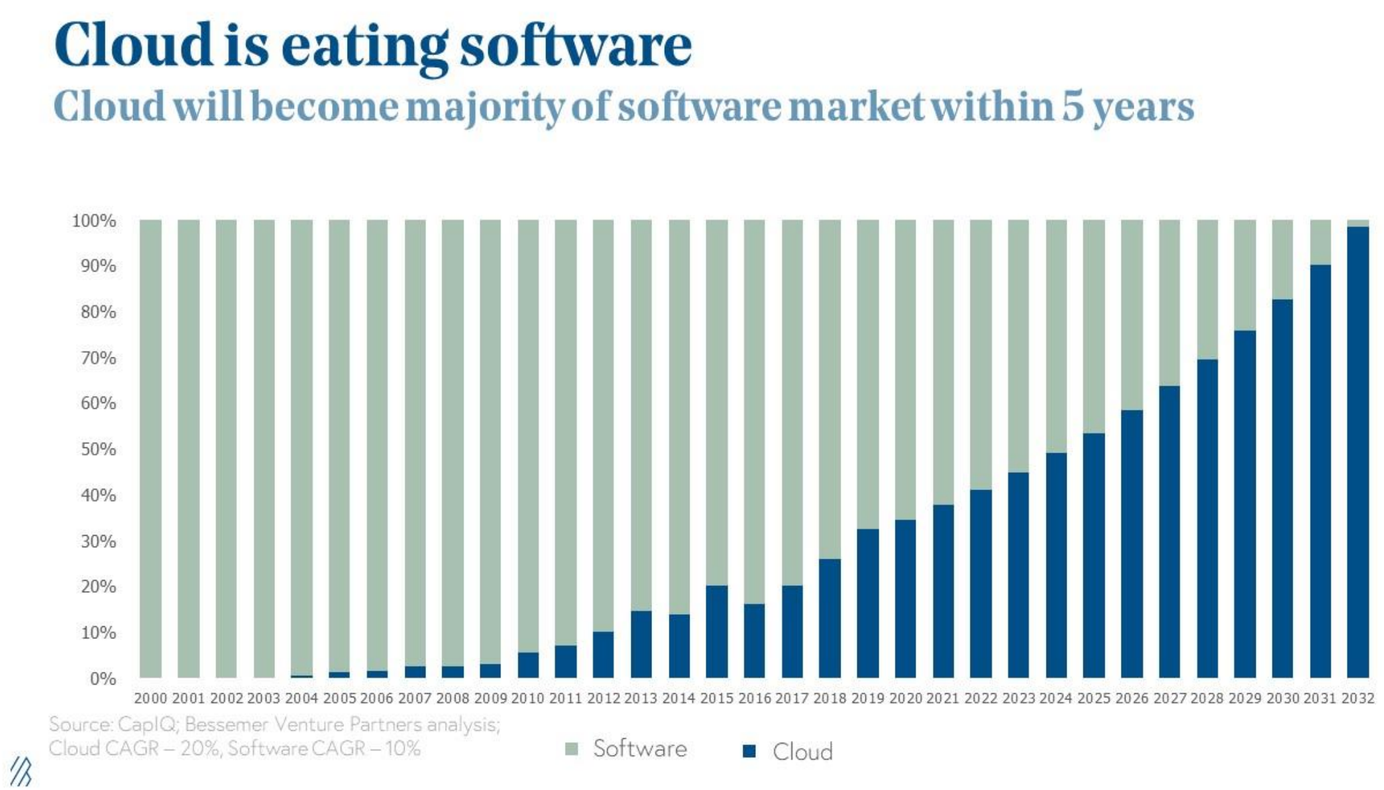
Programming of Distributed Systems

Topic VII – Cloud Computing

Dr.-Ing. Dipl.-Inf. Erik Schaffernicht

Why talk about Cloud Computing?



https://www.bvp.com/atlas/state-of-the-cloud-2020

Definition according NIST

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.

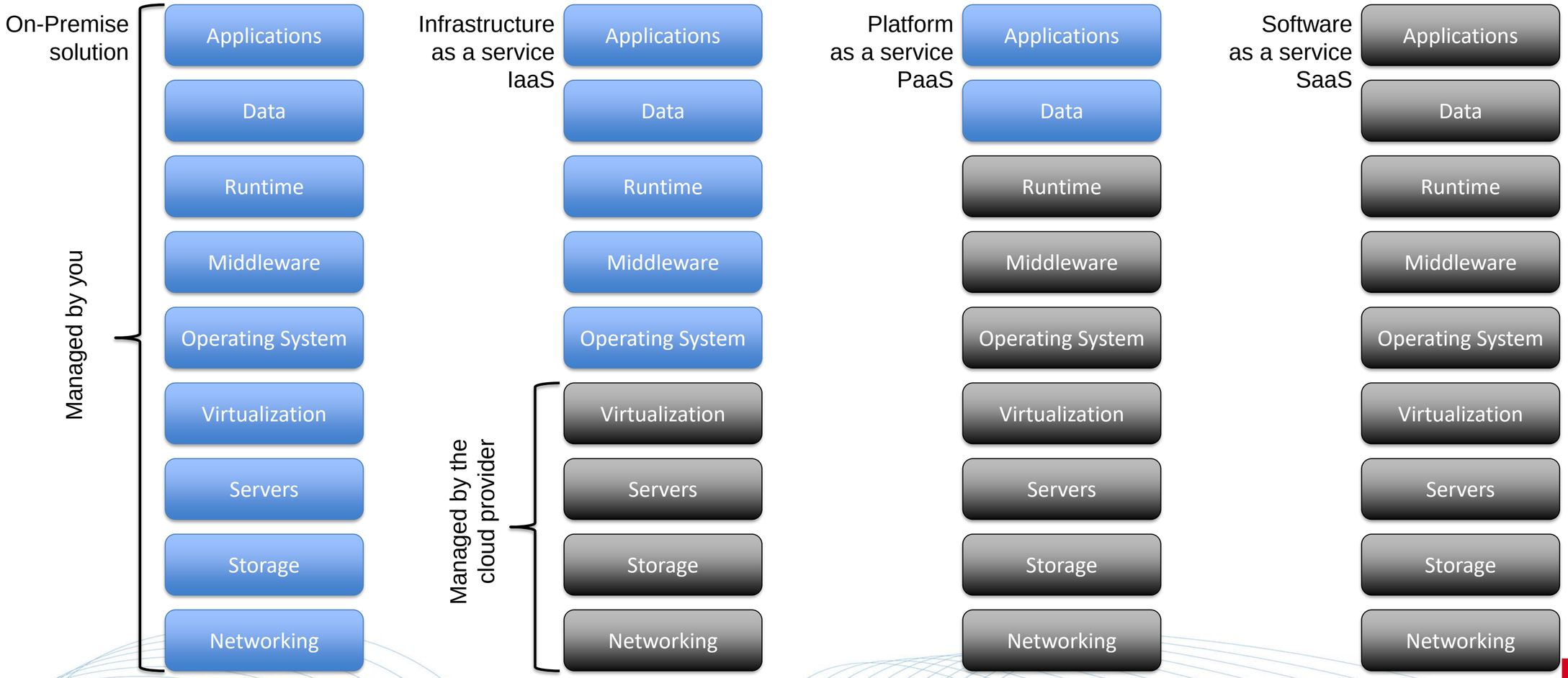
https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf

Essential Characteristics & Deployment

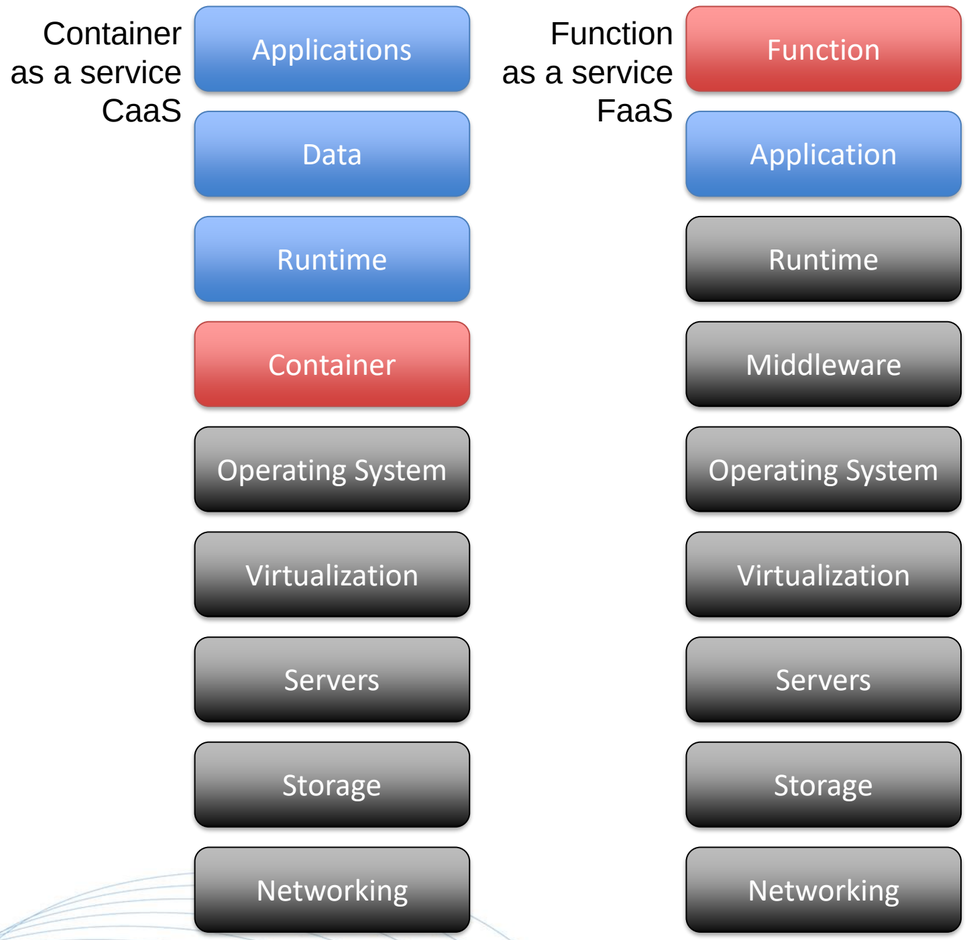
- On-demand self-service
- Broad network access
- Resource pooling
- Rapid elasticity
- Measured service

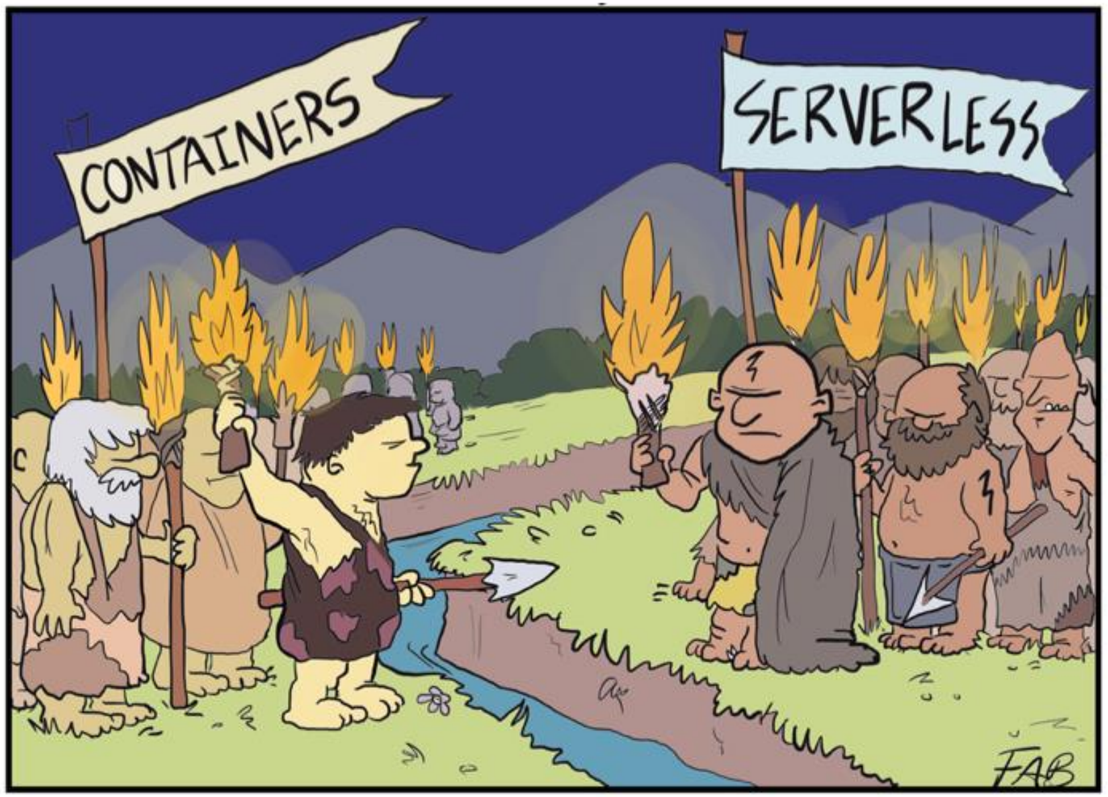
- Private Cloud
- Public Cloud
- Hybrid Cloud

Classic Service Models



More recent Service Models

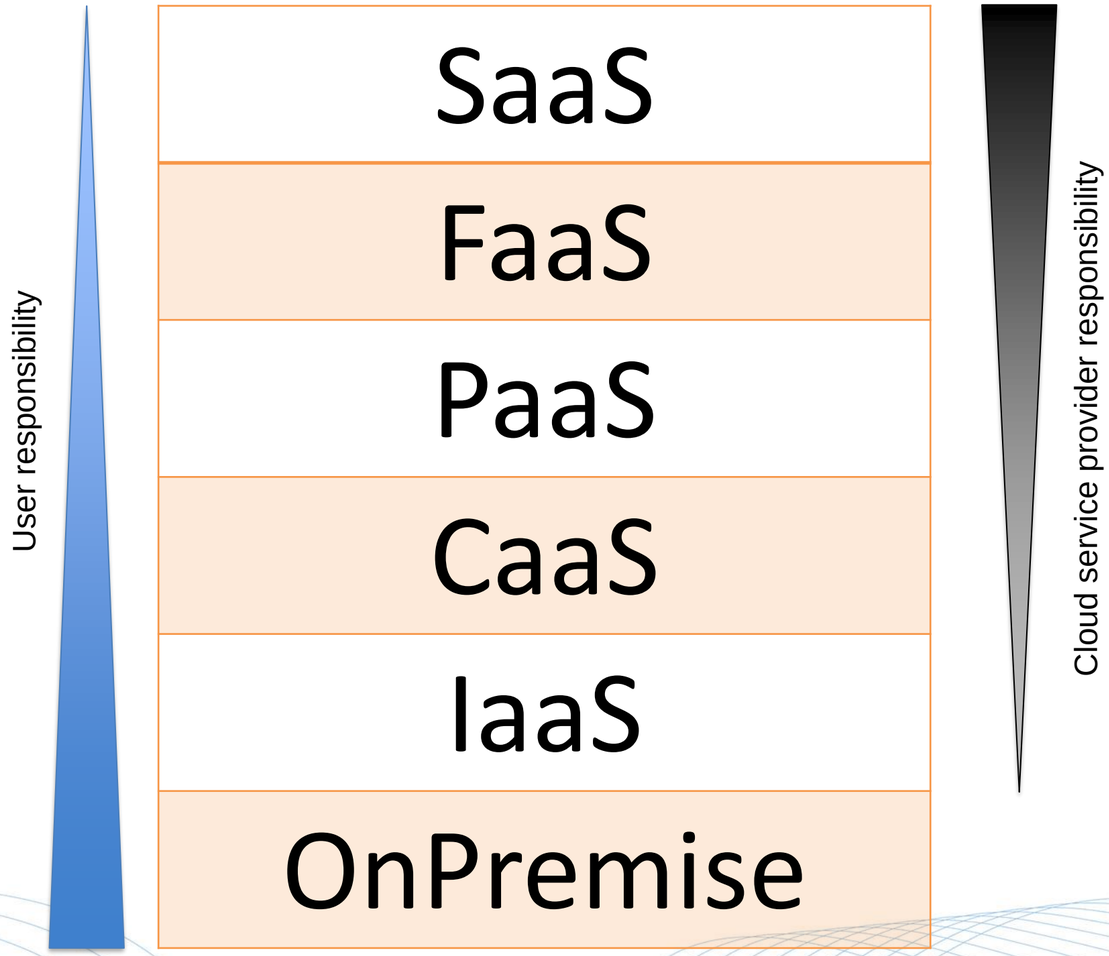




The two tribes regarded each other suspiciously in the glow of their blazing production environments.

https://faasandfurious.com/46

Hierarchy of Service Models

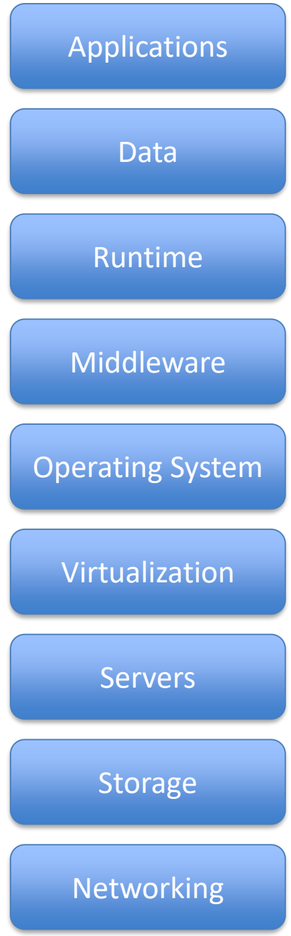


Service Model: On Premise

Traditional approach: Organization owns and manages its own IT infrastructure

Pros: total control of the infrastructure

Cons: Capacity Management, Hardware Management, Networking Management, Storage Management, Security



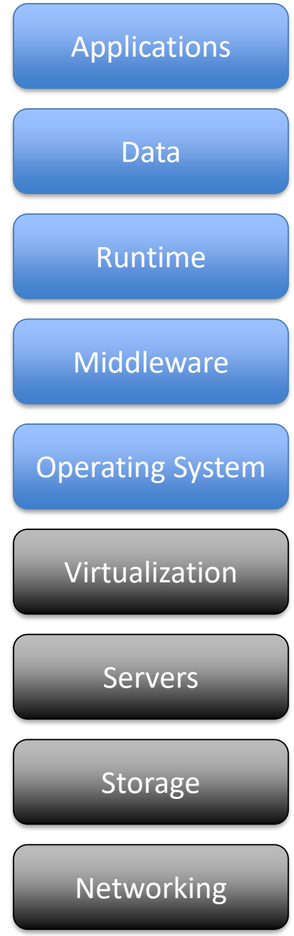
Service Model: Infrastructure-as-a-Service

Service as a basic commodity: Rent resources managed by the cloud provider

Pros: no hardware necessary, elastic, pay-per-use

Cons: Loss of control over hardware, possible higher costs in the long run, legal issues (data management), application redesign

Example: Azure, Stackscale, AWS, VMware



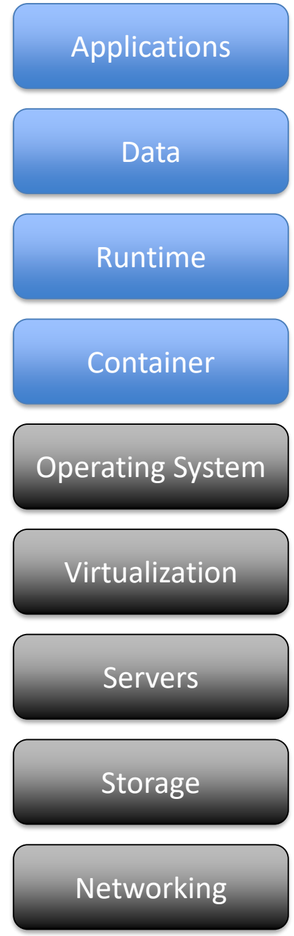
Service Model: Container-as-a-Service

Service as a basic commodity: Applications and dependencies packed as an image

Pros: multiple containers per machine, avoids vendor lock-in (i.e. easy to multi-cloud), declarative deployment

Cons: requires a container orchestrator + IaaS issues

Example: Docker Engine, Google Container Engine, OpenShift



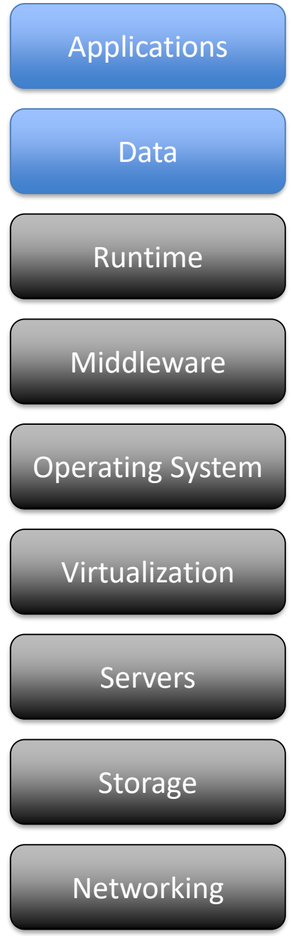
Service Model: Platform-as-a-Service

“Rent the operating system”: Customer uploads and controls the application

Pros: infrastructure managed by provider

Cons: limited to the deployment environment
customer in charge of load balancing & networking

Example: Heroku, AWS Elastic Beanstalk, Google App Engine



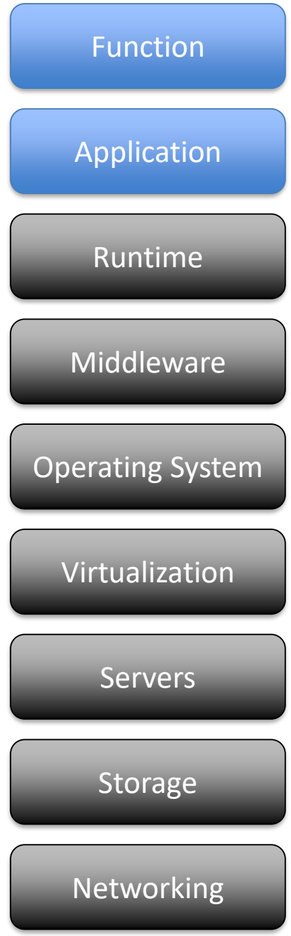
Service Model: Function-as-a-Service

Serverless: “Stateless Cloud RPCs”

Pros: autoscaling, no costs for idle time (compared to PaaS)

Cons: performance (latency for infrequently used functions)
vendor lock-in, monitoring and debugging

Example: AWS Lambda, Google Cloud Functions, Apache OpenWhisk, Oracle Cloud Fn, Cloudflare Workers



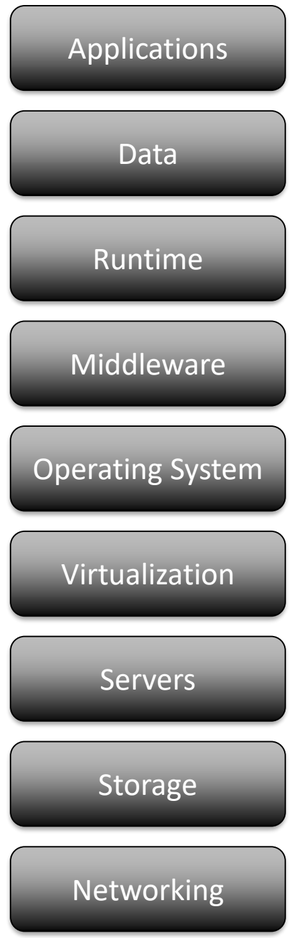
Service Model: Software-as-a-Service

Existing application that is provided on a cloud infrastructure: Customer access via thin clients/browser

Pros: no deployment or configuration decisions

Cons: limited control

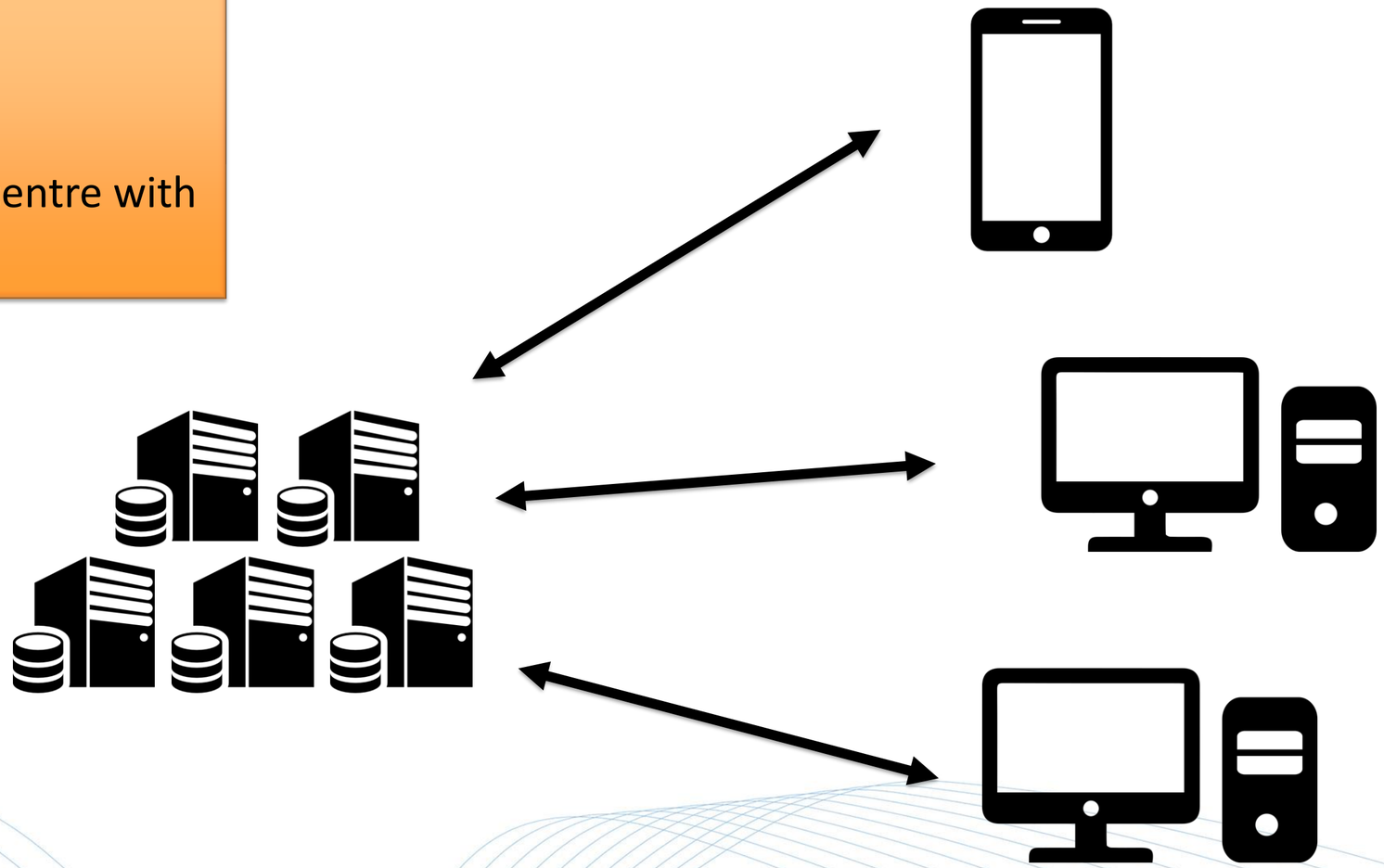
Example: Trello, Slack, Gmail, Office365, …



Traditional Data Center Approach

Issues:

- Network latency
- Bandwidth issues
- Overwhelmed data centre with unsuitable hardware



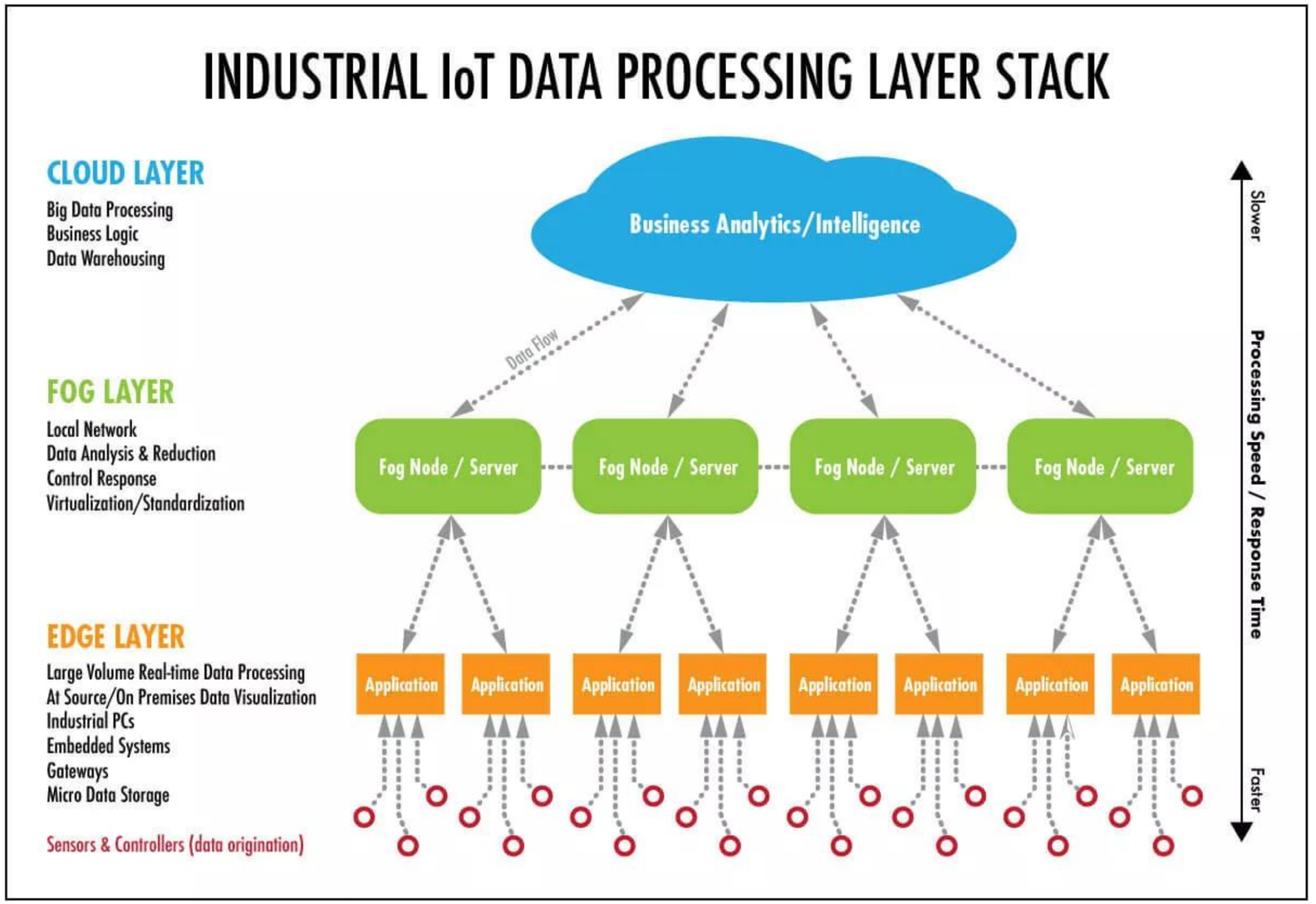
Edge Computing

Edge Nodes – Mini data centres:

- Small and modular
- Possibly mobile
- Non-traditional networking
- Mixture of hardware devices

Fog Computing

- Represents the integration of cloud and edge
- Distributed resources and services along a continuum from Cloud to IoT



Cloud and Fog