Managed Services

- Do you want to continue **running applications in the cloud, the same way you run them in your data center**?
- OR are **there OTHER approaches**?
- You should **understand some terminology** used with cloud services:
- **IaaS** (Infrastructure as a Service) PaaS (Platform as a Service)
- **FaaS** (Function as a Service)
- **CaaS** (Container as a Service).
- **Serverless**.
- Let's get on a quick **journey** to understand these!

IAAS (Infrastructure as a Service)

*Use **only infrastructure** from cloud provider.

Example: Using VM to deploy your applications or databases.

You are responsible for:

*Application Code and Runtime

* Configuring load balancing.

* Auto scaling OS upgrades and patches.

* Availability.

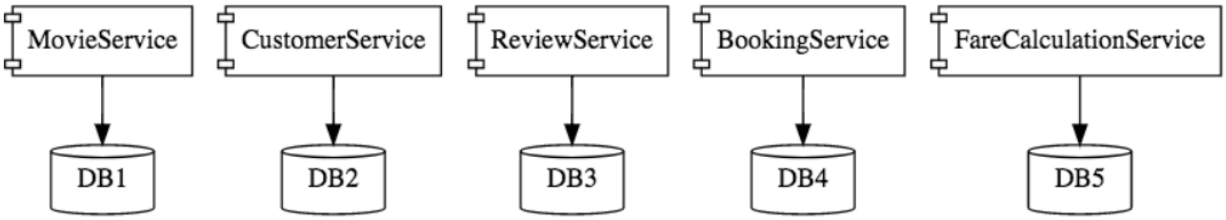
* etc.. (and a lot of things!)

PAAS (Platform as a Service)

- Use a platform provided by cloud.
- **Cloud provider** is responsible for:
- OS (incl. upgrades and patches)
- Application Runtime.
- Auto scaling, Availability & Load balancing etc..
- You are responsible for:
- Configuration (of Application and Services)
- Application code (if needed).
- Varieties:
- **CAAS (Container as a Service):** Containers instead of Apps.
- **FAAS (Function as a Service):** Functions instead of Apps

Databases - Relational & NoSQL (Amazon RDS, Google Cloud SQL, Azure SQL Database etc), Queues, AI, ML, Operations etc!

Microservices



- Enterprises are heading towards microservices architectures.

- Build small focused microservices.

- **Flexibility to innovate** and build applications in different programming languages (Go, Java, Python, JavaScript, etc).

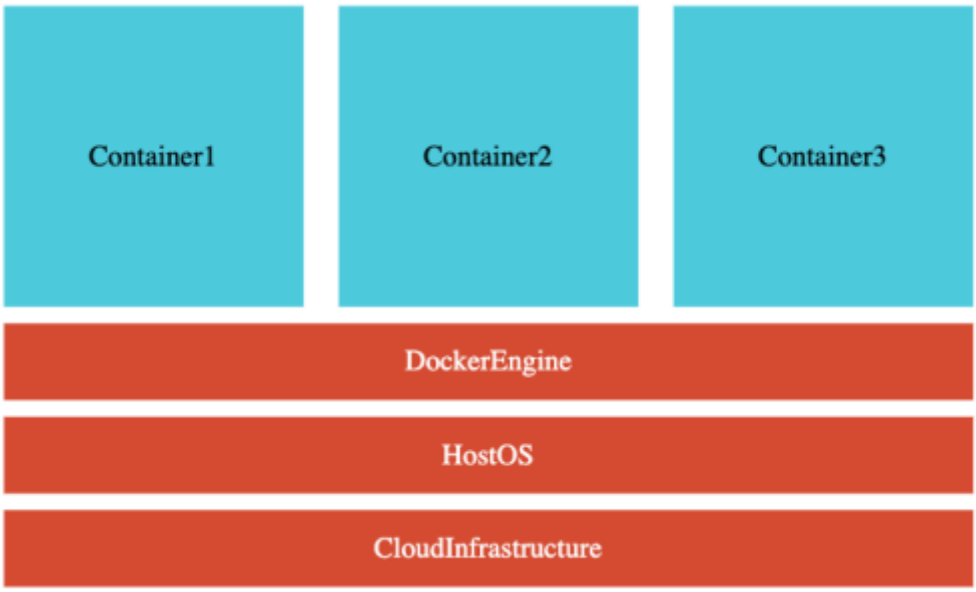
- BUT **deployments become complex!**

- How can we have **one way of deploying** Go, Java, Python or JavaScript .. microservices?

- Enter containers!

==========

Containers – Docker



*Create Docker images for each microservice.

- Docker image **has all needs of a microservice:**
- Application Runtime (JDK or Python or NodeJS).
- Application code and Dependencies.
- Runs the **same way on** any infrastructure:
- Your local machine.
- Corporate data center.

- Cloud.

- Advantages

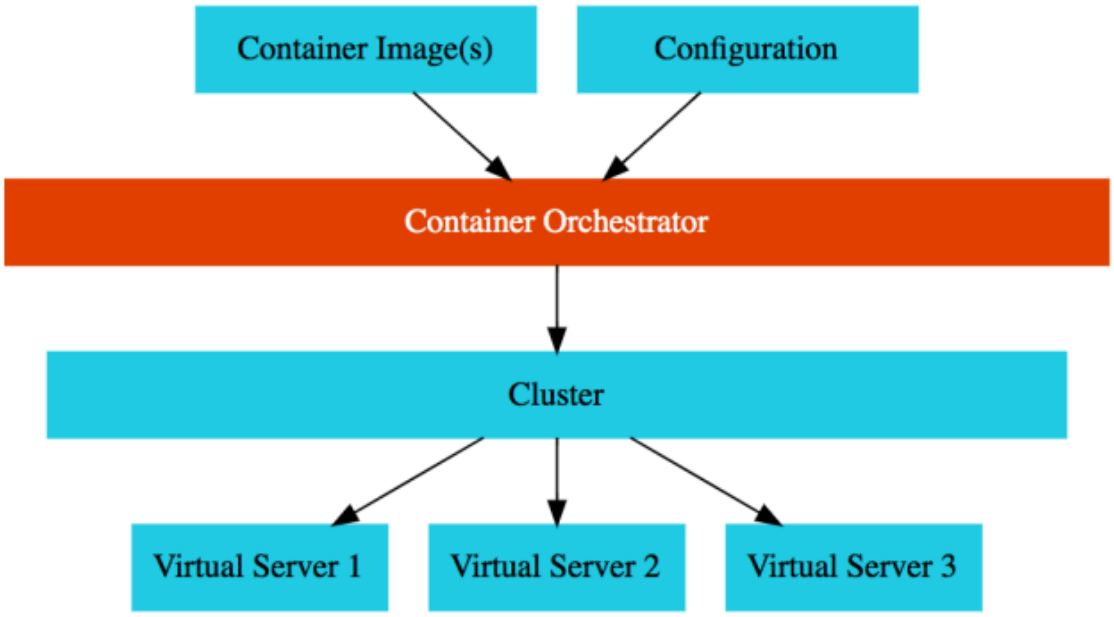
- Docker containers are **light weight**.

- Compared to Virtual Machines as they do not have a Guest OS.

- Docker provides for containers.

- Docker is **cloud neutral.**

Container Orchestration



* **Requirement** : I want 10 instances of Microservice A container, 15 instances of Microservice B container and ....

Typical Features:

- **Auto Scaling** - Scale containers based on demand.
- **Service Discovery** - Help microservices find one another.
- **Load Balancer** - Distribute load among multiple instances of a microservice.
- **Self Healing** - Do health checks and replace failing instances.
- **Zero Downtime Deployments** - Release new versions without downtime.

Serverless

* What do we think about when we develop an application?
- Where to deploy? What kind of server? What OS?
- How do we take care of scaling and availability of the application?
- **What if you don't need to worry about servers and focus on your code?**
- Enter **Serverless**.
- Remember: **Serverless does NOT mean "No Servers".**
- **Serverless for me:**
- You **don't worry** about infrastructure (ZERO visibility into infrastructure).
- Flexible scaling and automated high availability.
- Most Important: **Pay for use.**
- Ideally ZERO REQUESTS => ZERO COST.
- **You focus on code** and the cloud managed service takes care of all that is needed to scale your code to serve millions of requests!
- And you pay for requests and NOT servers!

Serverless - My Perspective!

- Serverless - Important Features:
- 1: Zero worry about infrastructure, scaling and availability
- 2: Zero invocations => Zero Cost (Can you scale down to ZERO instances?)
- 3: Pay for invocations and NOT for instances (or nodes or servers)
    - Serverless **Level 1**: Features (1 + 2)
    - Serverless **Level 2**: Features (1 + 2 + 3).
    - When I refer to Serverless, I'm referring to Level 2.
- HOWEVER cloud providers include managed services at Level 1 and Level 2:
- **Level 1: Google App Engine** (Google Calls it "App Engine is a fully managed, serverless platform"), **AWS Fargate** (AWS calls it "serverless compute engine for containers").
    - Scale down to ZERO instances when there is no load, BUT you pay for number (and type) of instances running!
- **Level 2: Google Functions, AWS Lambda, Azure Functions** etc.
    - You pay for invocations.