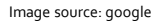# Microservices

Docker: Containers

Kubernetes: Orchestrators

Contents

- What are Microservices?
- Why Using Microservices?
- Patterns / Monolithic vs Microservices Architecture
- Microservices are Perfect!?
- Important keys about microservices
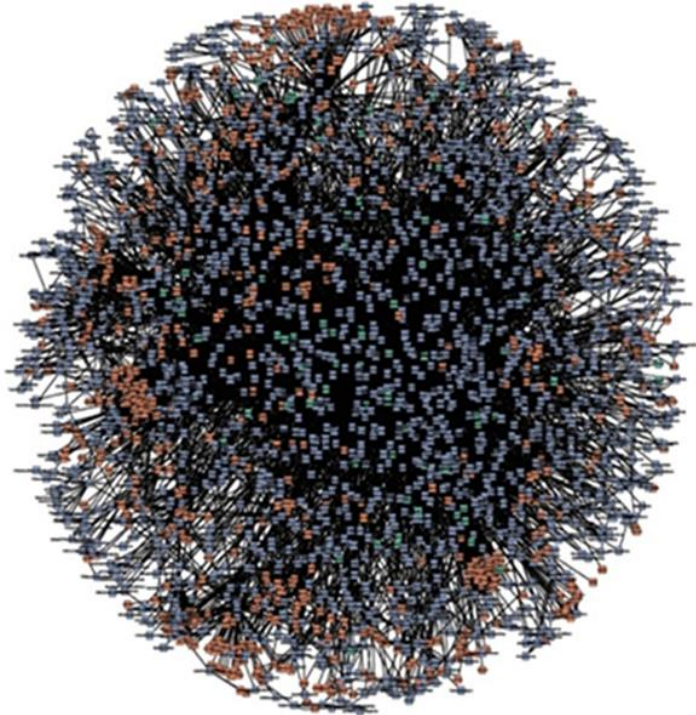- Docker: Container for Microservices
  - Where it came from?
  - What is Docker?
  - Why Docker?
  - Are there limitations to using Docker?
- Kubernetes : Orchestrator for Microservices
  - Where it came from?
  - What is Kubernetes?
  - Kubernetes Features

Image source: google

What are Microservices?

Amazon

Netflix

What are Microservices?

James Lewis and Martin Fowler

- the microservice architectural style is an approach to
- developing a single application* as a **suite of small services**,
- each **running in its own process** and communicating with lightweight mechanisms, often an HTTP resource API.
- These services are built around business capabilities and
- **independently deployable** by fully automated deployment machinery.
- There is a **bare minimum of centralized management**** of these services, which may be written in different programming languages and use different data storage technologies.
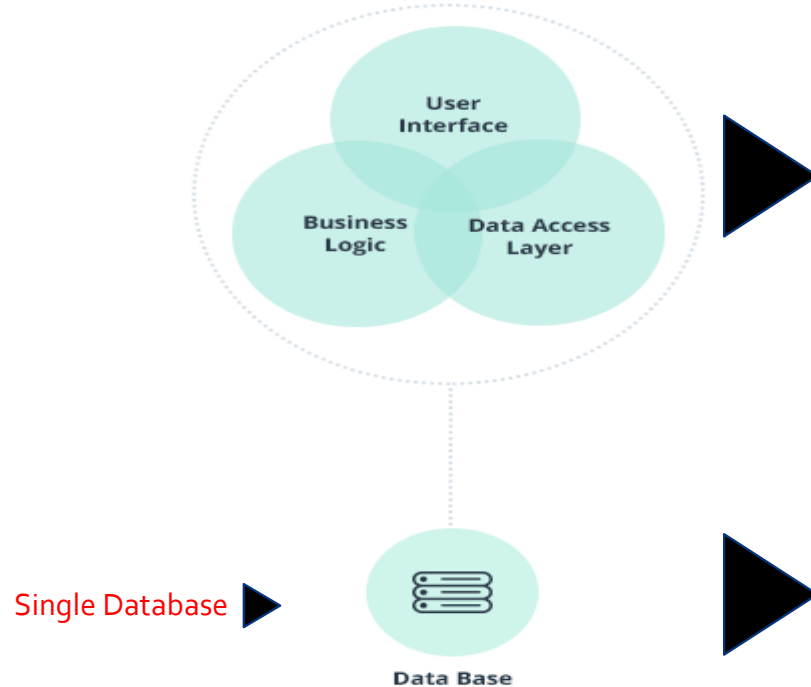
* Single Resposibilty Principle(SRP)

** Loosly coupled

Why using Microservices?

Traditional / monolithic architecture could not meet to requirements of the new World.
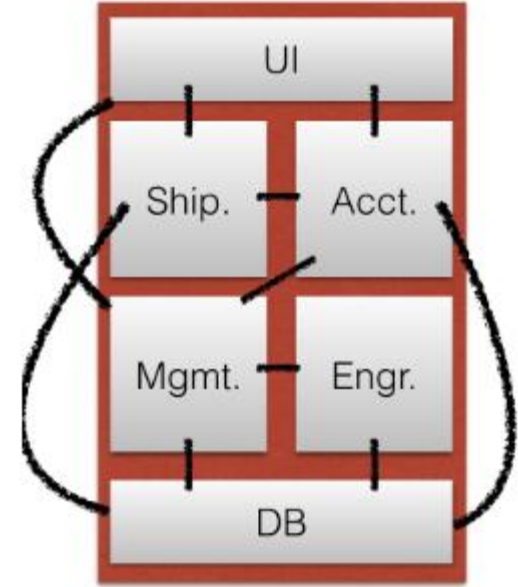
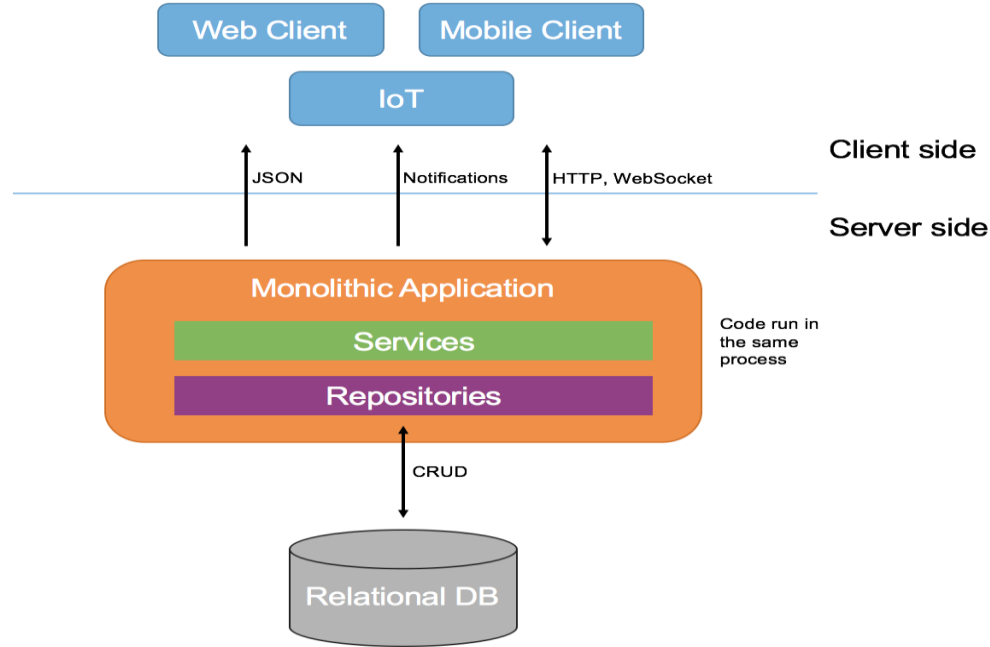# Traditional approach:Monolithic Architecture

User
Interface

Business
Logic

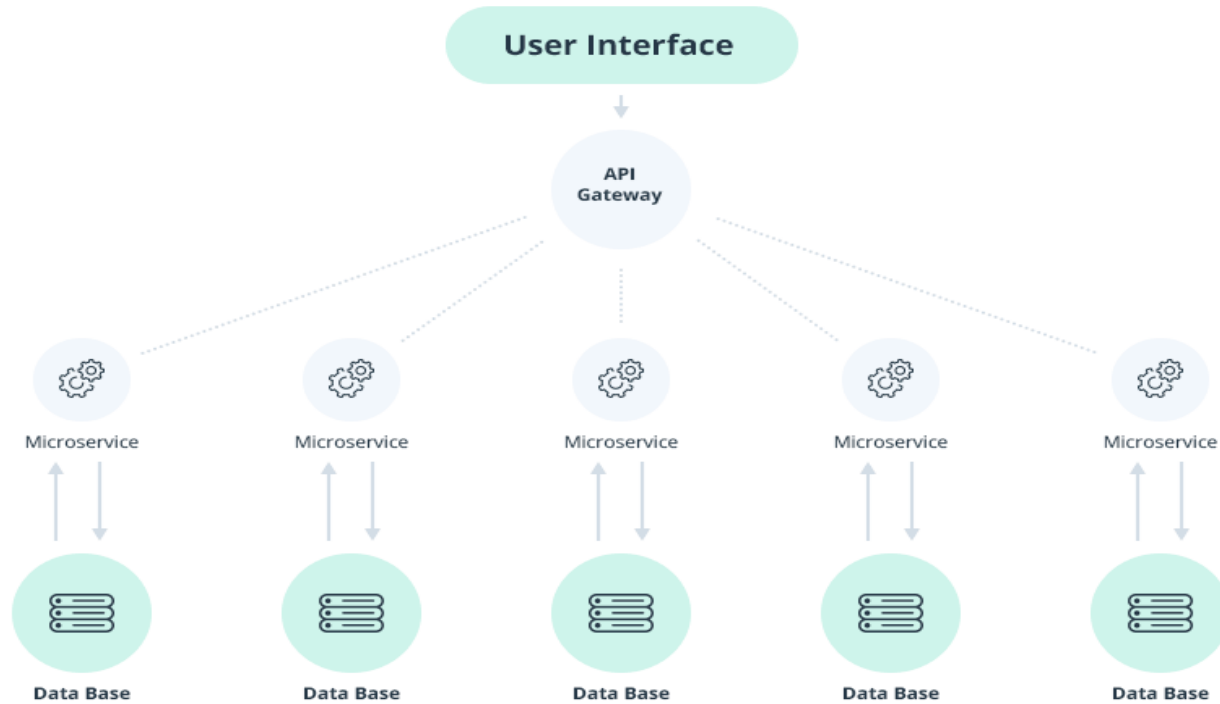Data Access
Layer

Single Database

Data Base

Monolithic tiers

- Presentation: JS / IOS / Android

- Application: Java / Python

- Database: MongoDB, PostgreSQL, Oracle

Monolithic Architecture Example

Image Source: https://learn.g2crowd.com/microservices

That is why we need to new approach.

New approach is Microservices

Flexibility...

Reliability…

Development(Test, Maintenance) Speed…

Building Complex Application...

Scalability…

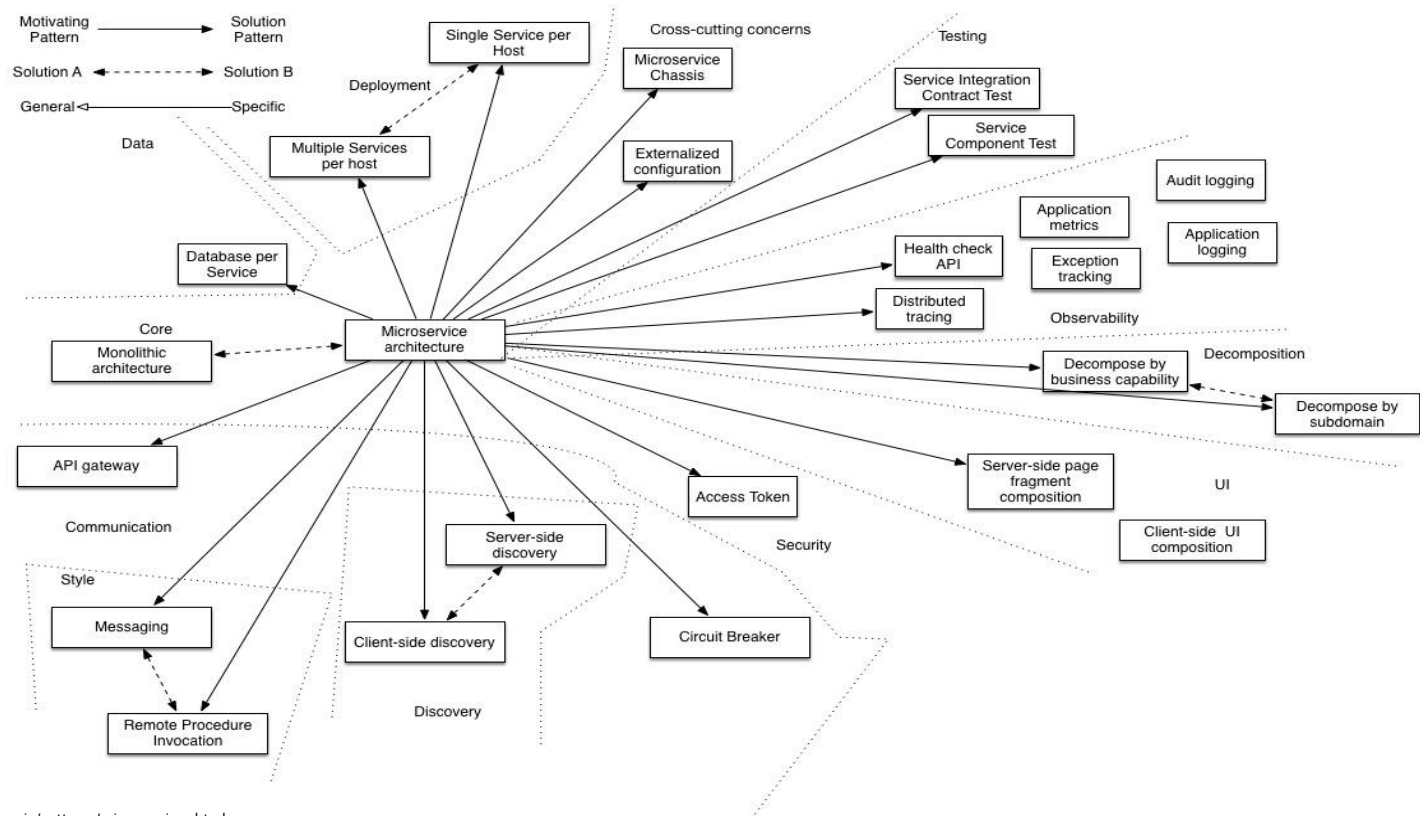Continuous Deployment…

For new World Microservices are good solution, but unfortunately have drawbacks at the same time.

Increased Memory Consumption…

Distributed System...

Microservices with Distributed System

Source:https://microservices.io/patterns/microservices.html

Important keys about microservices…

- Data-base
  // MongoDB, Apache Cassandra, Redis

- Data Synchronization
  // Apache Kafka, Akka

- Testing
  // Hoverfly, Pact, VCR

- Logging
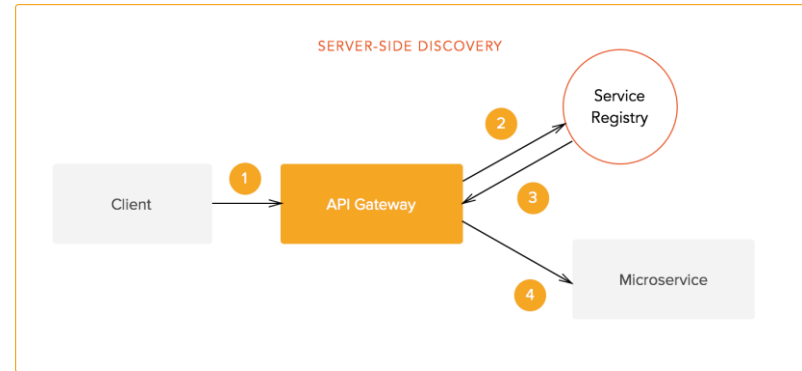 // Istio, Loggly

- Security / Autherization and Authentication
  // Kerberos, Open ID

- Continuous Delivery
  // Jenkins, Asgard, Aminator

- Containers
  // Docker**, Rocker, Rkt

- Orchestrating
  // Kubernetes**, Mesos, Docker Swarm

- Distributed Services

  ➢ Service Registry and Discovery
     // Eureka, Consul, Zookeeper

- Distributed Services

  ➢ Circuit Breaker: *Failure Managment*

    - Closed

    - Open

    - Half-Open

    // Hsytrix

- Distributed Services

  ➢ Gateway
    // Zuul, Netty, Feign

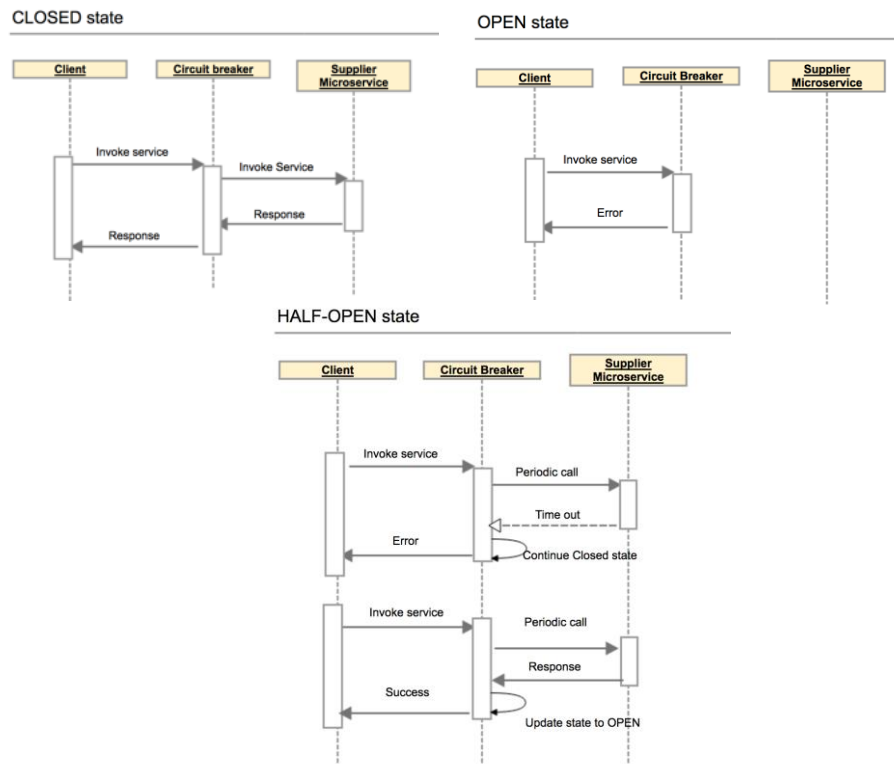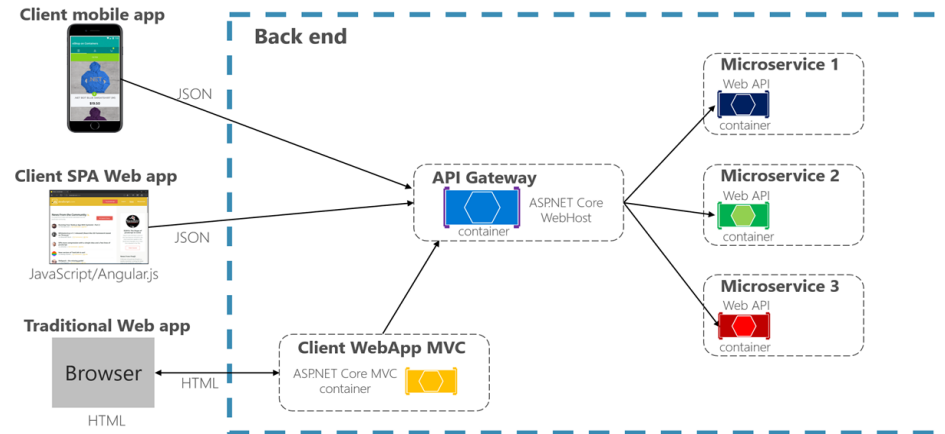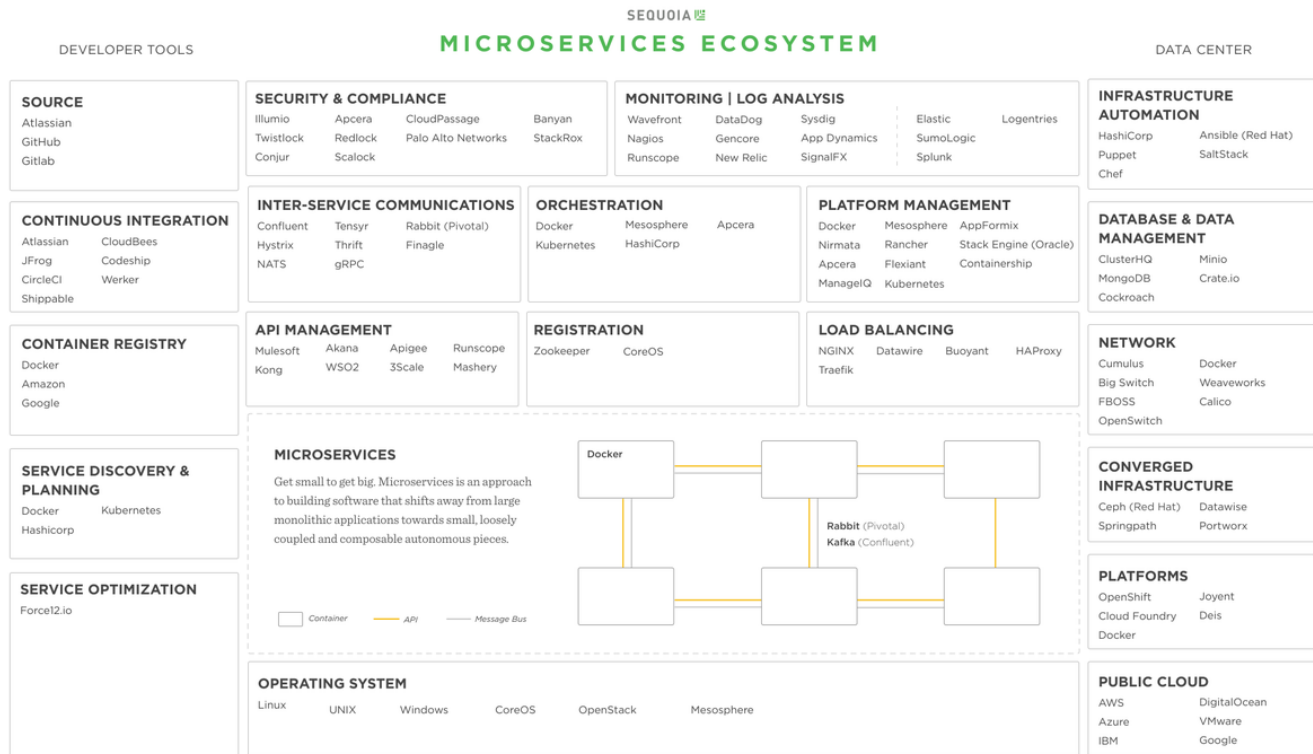# Microservices Ecosystem

SEQUOIA

MICROSERVICES ECOSYSTEM

DEVELOPER TOOLS

DATA CENTER

SOURCE
Atlassian
GitHub
Gitlab

SECURITY & COMPLIANCE
Illumio	Apcera	CloudPassage	Banyan
Twistlock	Redlock	Palo Alto Networks	StackRox
Conjur	Scalock		

MONITORING | LOG ANALYSIS
Wavefront	DataDog	Sysdig	Elastic	Logentries
Nagios	Gencore	App Dynamics	SumoLogic	
Runscope	New Relic	SignalFX	Splunk	

INFRASTRUCTURE AUTOMATION
HashiCorp	Ansible (Red Hat)
Puppet	SaltStack
Chef	

CONTINUOUS INTEGRATION
Atlassian	CloudBees
JFrog	Codeship
CircleCI	Werker
Shippable	

INTER-SERVICE COMMUNICATIONS
Confluent	Tensyr	Rabbit (Pivotal)
Hystrix	Thrift	Finagle
NATS	gRPC	

ORCHESTRATION
Docker	Mesosphere	Apcera
Kubernetes	HashiCorp	

PLATFORM MANAGEMENT
Docker	Mesosphere	AppFormix
Nirmata	Rancher	Stack Engine (Oracle)
Apcera	Flexiant	Containership
ManageIQ	Kubernetes	

DATABASE & DATA MANAGEMENT
ClusterHQ	Minio
MongoDB	Crate.io
Cockroach	

CONTAINER REGISTRY
Docker
Amazon
Google

API MANAGEMENT
Mulesoft	Akana	Apigee	Runscope
Kong	WSO2	3Scale	Mashery

REGISTRATION
Zookeeper	CoreOS

LOAD BALANCING
NGINX	Datawire	Buoyant	HAProxy
Traefik			

NETWORK
Cumulus	Docker
Big Switch	Weaveworks
FBOSS	Calico
OpenSwitch	

SERVICE DISCOVERY & PLANNING
Docker	Kubernetes
Hashicorp	

MICROSERVICES

Get small to get big. Microservices is an approach to building software that shifts away from large monolithic applications towards small, loosely coupled and composable autonomous pieces.

Docker

Rabbit (Pivotal)
Kafka (Confluent)

Container — API — Message Bus

CONVERGED INFRASTRUCTURE
Ceph (Red Hat)	Datawise
Springpath	Portworx

SERVICE OPTIMIZATION
Force12.io

PLATFORMS
OpenShift	Joyent
Cloud Foundry	Deis
Docker	

OPERATING SYSTEM
Linux	UNIX	Windows	CoreOS	OpenStack	Mesosphere

PUBLIC CLOUD
AWS	DigitalOcean
Azure	VMware
IBM	Google

Includes both companies and open source projects

Version 1.2 | 01.18.16

Architecture evolution for some tech companies

- eBay

  - Monolithic Perl -> Monolithic C++ -> Java -> Microservices

- Amazon

  - Monolithic C++ -> Java / Scala-> Microservices

- Twitter

  - Monolithic Rails -> JS / Rails / Scala -> Microservices

Docker : Container For Microservices...

Where it came from?

- The name Docker comes from British colloquialism meaning dock worker

- First version released 2013

- Written in Go/Golang

What is Docker?

- « a software platform that allows you to build, test, and deploy applications quickly. Docker packages software into standardized units called containers** that have everything the software needs to run including libraries, system tools, code, and runtime.»

Amazon

- A lightweight way to virtualize applications
- Shortly write your code in any language and run anywhere - WORA
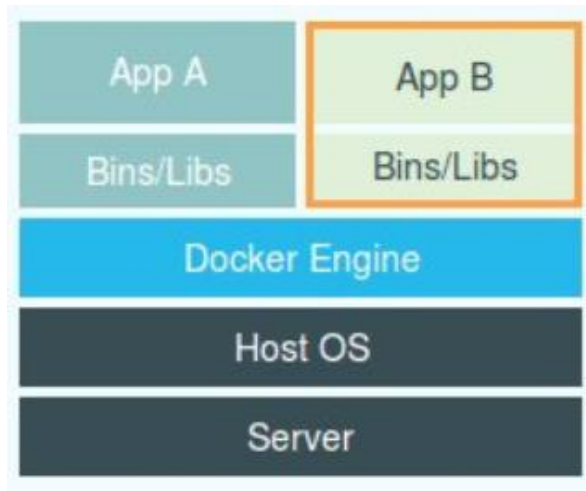
Is Docker a Virtual Machine?

- Virtual but not machine, it is a container what virtualized/containerized apps
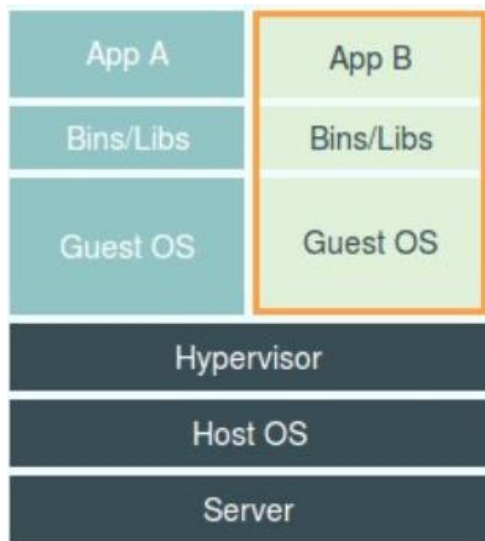
Virtual Machines
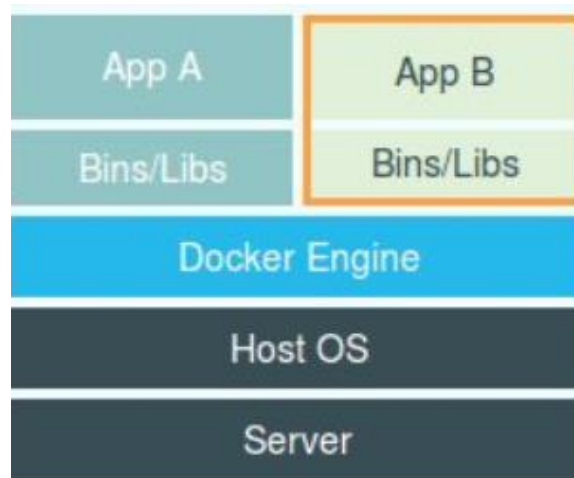
Docker / Container

Docker : Container For Microservices…

Why Docker?

- Efficiency

- Velocity(to market, deployment)…

- Portability…

- Sharability…

300% +
Faster Time to Market

1300% +
Developer Productivity
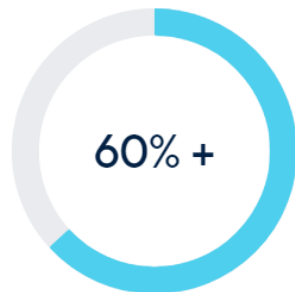
60% +
Deployment Velocity
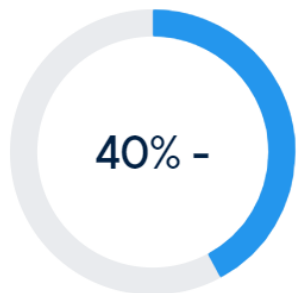
40% -
IT Infrastructure Reduction
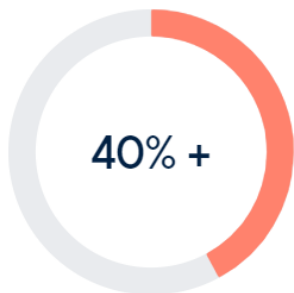
40% +
IT Operational Efficiency
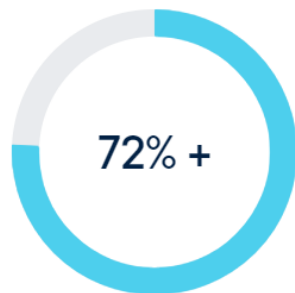
72% +
Faster Issue Resolution

Any limitations to using Docker?

- Docker, by itself is very good when it manages single container

Docker Difficulties

-    But when you start using more and more containers and containerized app get struggling with very difficulties.

Docker Difficulties

Containers are great but not enough at...

- Managment
- Orchestrating
- Scalling up
- Redundancy
- Scheduling
- Service Discovery
- Resiliency
- Configuration

That is why we need to Orchestrators.

Kubernetes: Orchestrator for Microservices/Containers...

Where it came from?

- Kubernetes word from Greek 'Helmsman'
  < the person who steers a ship

- Also the root of the words 'governor' and 'cybernetic'

kubernetes

Kubernetes: History

- Born in Google and Informed by Google's experiences and internal systems(borg)

- Donated to Cloud Native Computing Foundation(CNCF) in 2014 (open source)

- Written in Go/Golang

- %100 Open Source

- https://github.com/kubernetes/kubernetes

- Often shortened to K8s

Source: wiki, pluralsight

What is Kubernetes?

Kubernetes is

- An orchestrator for microservice apps run on containers

- A container scheduler

- Managment containerized cluster/applications and automating deployment tool

- Supports multiple cloud and multiple container runtimes
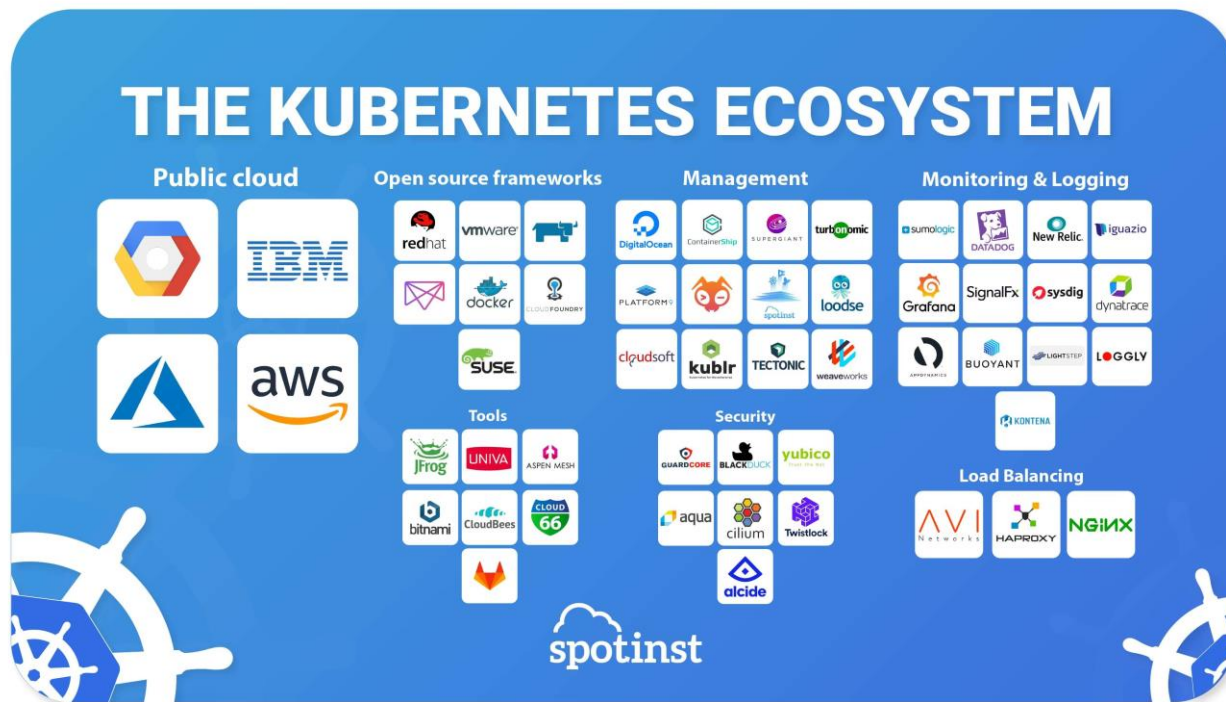
Kubernetes

Think a team without a coach / manager
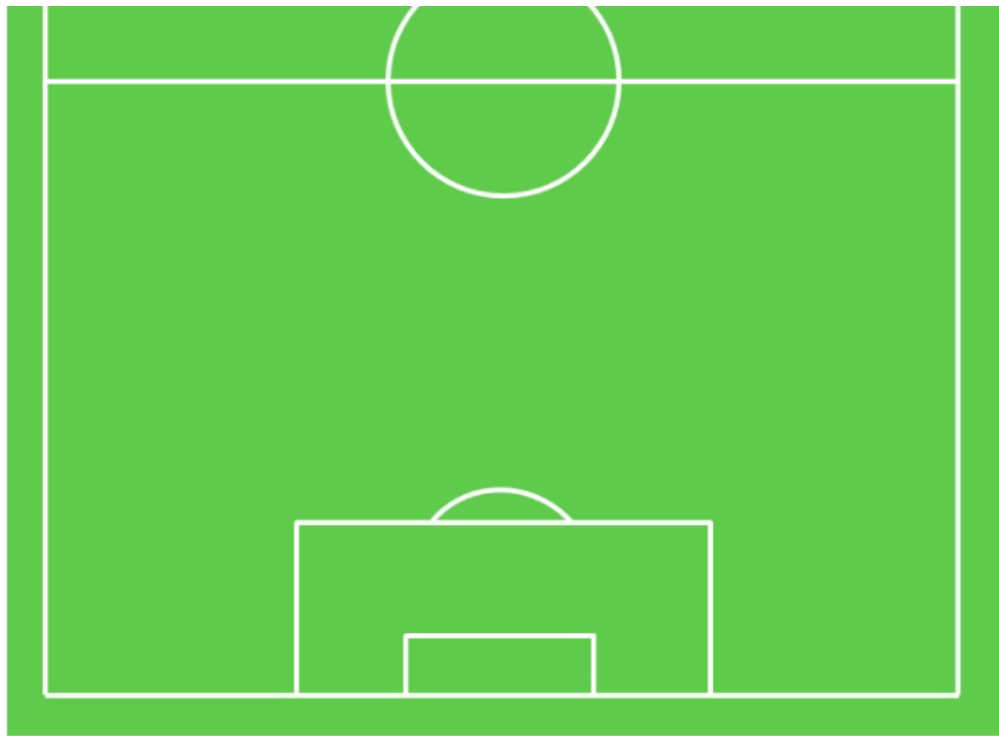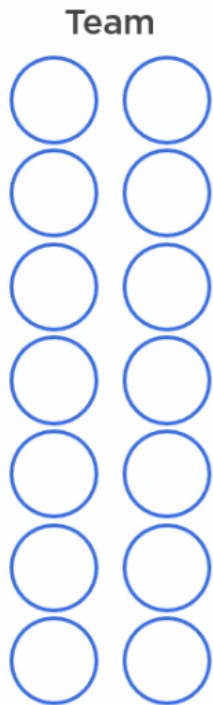
Team

How are they organize?
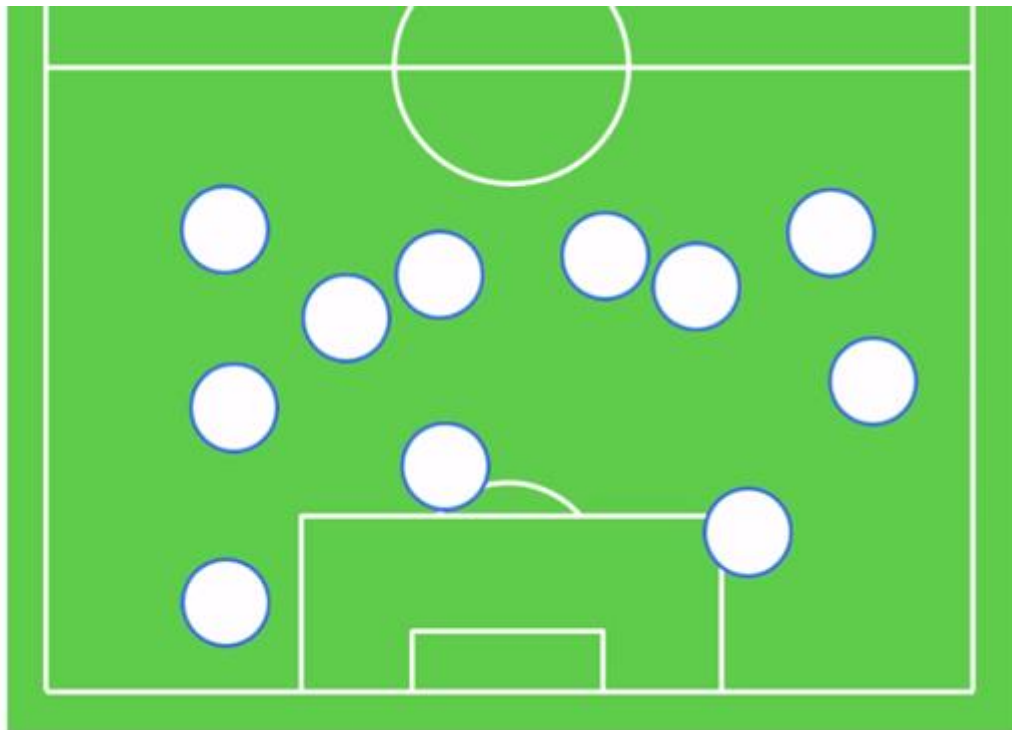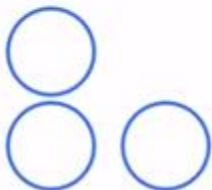
But if they have…
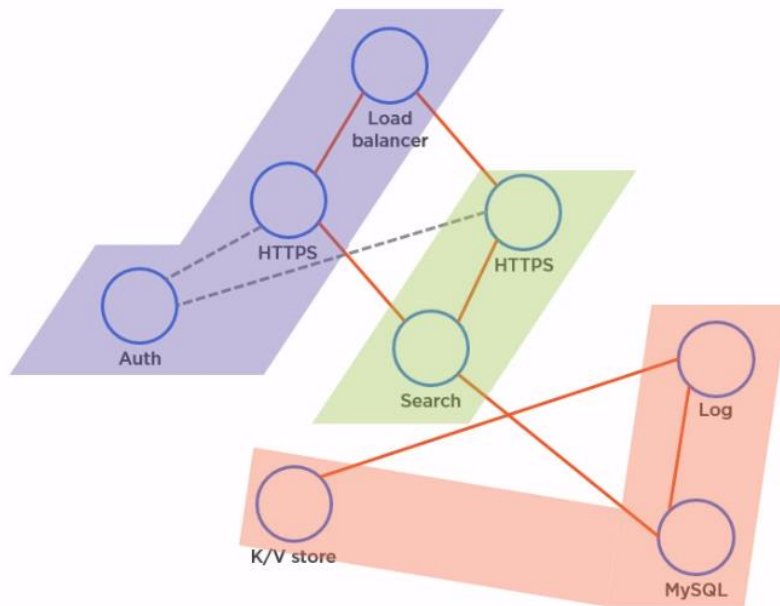
Kubernetes do exactly the same thing like manager.
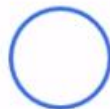
Managed, Organized, Orchestrated

Kubernetes Features…

Service Discovery and Load Balancing...

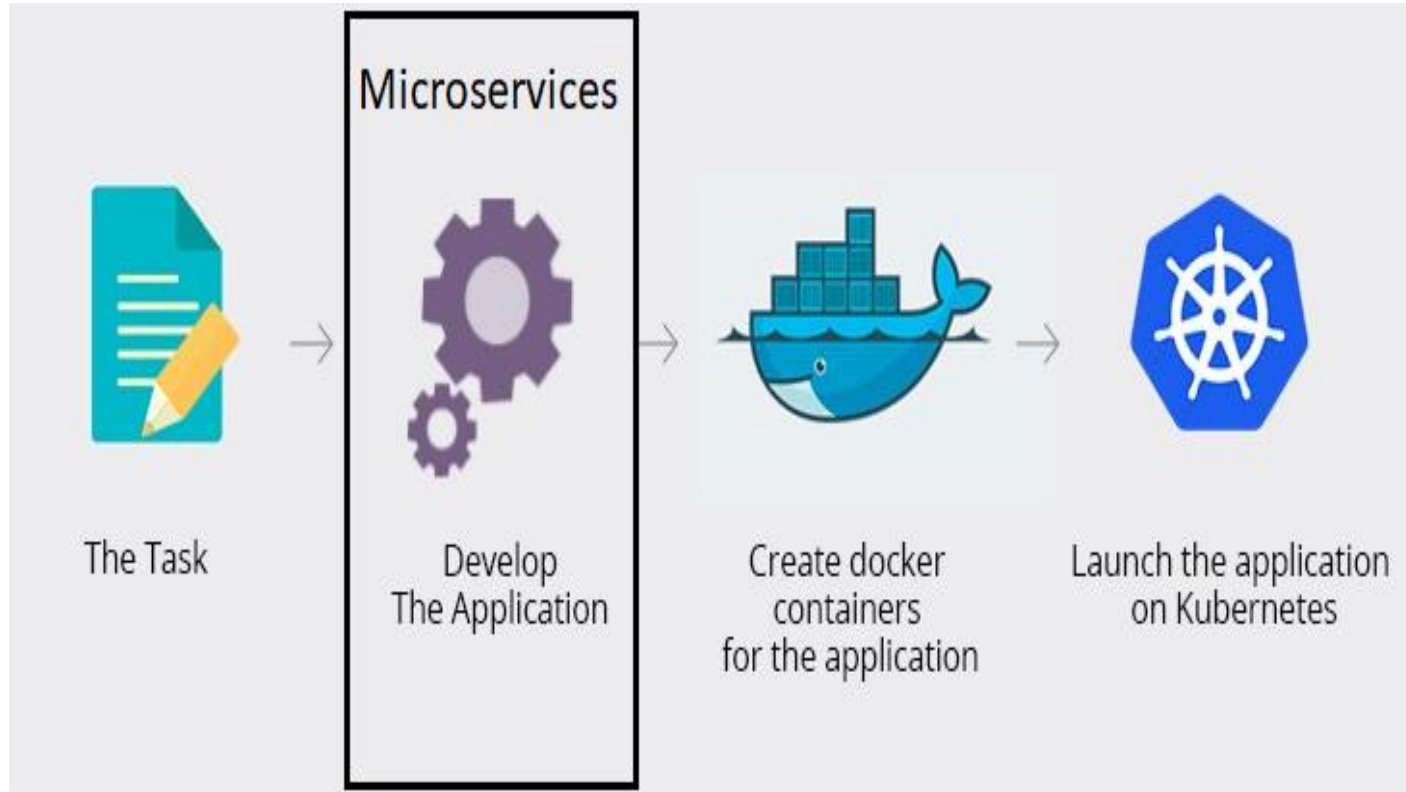Storage Orchestration…
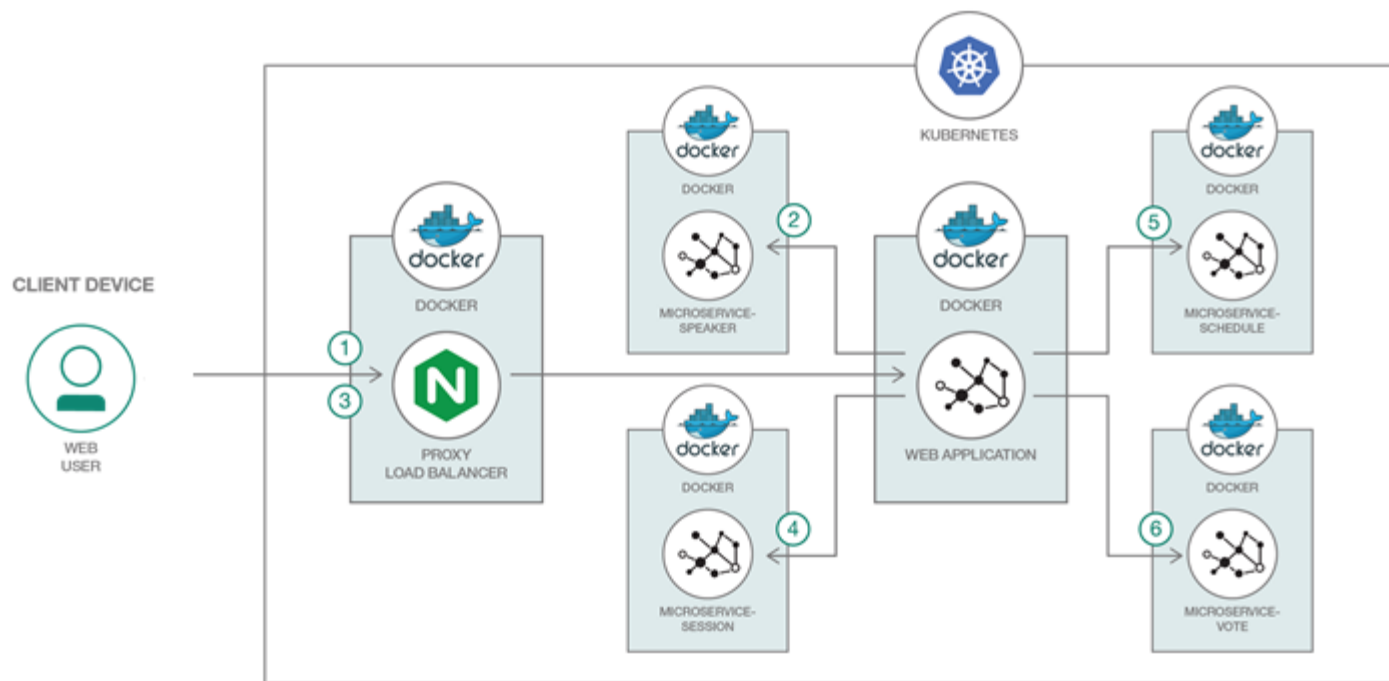
Automated rollouts and rollbacks…

Batch Execution...

Cost Savings…

Microservices > Docker > Kubernetes Flow

The Task → Microservices: Develop The Application → Create docker containers for the application → Launch the application on Kubernetes
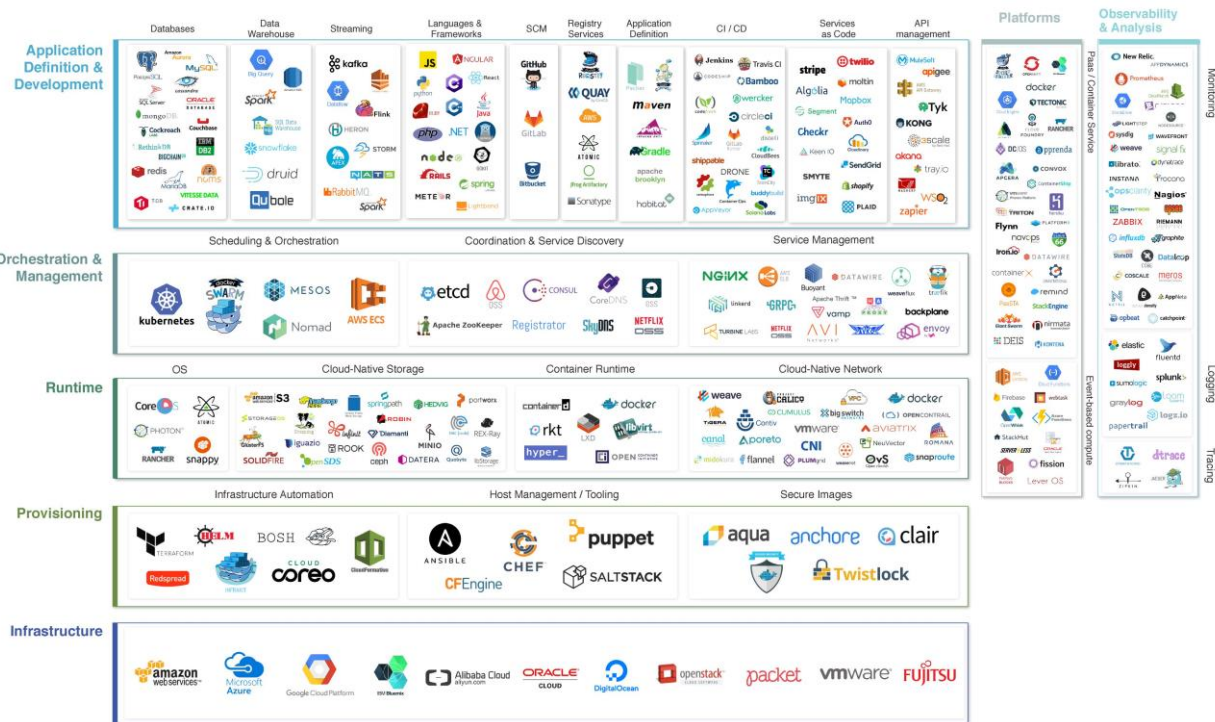
Microservices > Docker > Kubernetes Flow Example

Cloud Native Landscape v0.9.4

Any questions?