Fundamentals of Cloud Native Architecture: Accelerate you Infra-DevOps skills

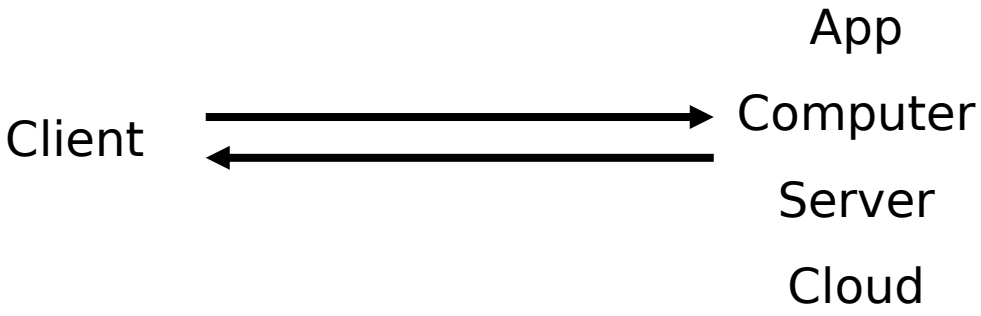
Setyo Legowo
Tech Consultant Engineer at **99.co**

Kubernetes and Cloud Native Bandung
November 16th, 2019

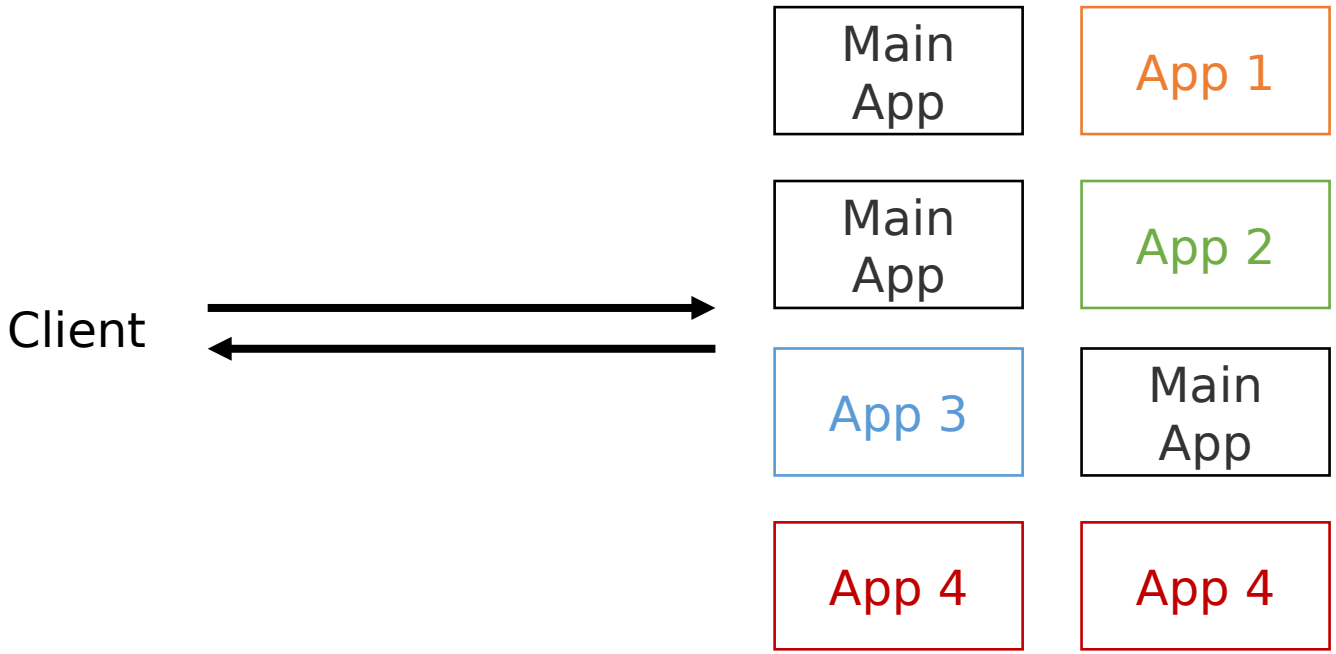
Presented by 99.co

Start from ...

- Growing and Rapid Changes Requirement
- Classic vs modern best practice
- Interoperability: work on many infrastructure
- Taking advantage of cloud computing/infrastructure



Basic cloud computing



NOT SO SIMPLE BECAUSE NOT SIMPLE

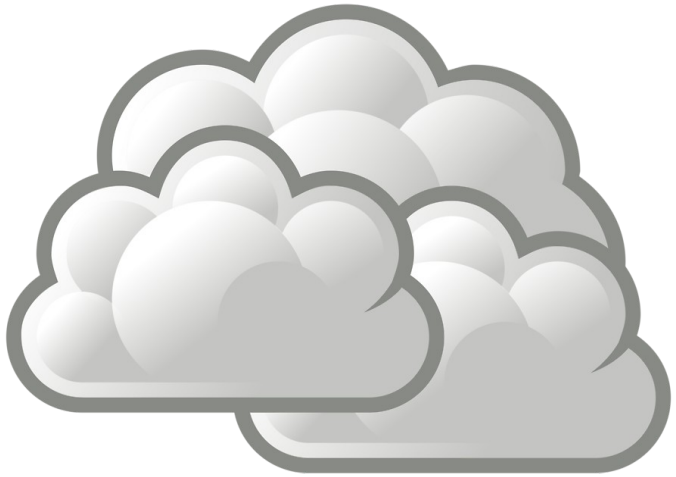
Design Planning

Step after requirement gathering

- Foundation of design:
  - Abstraction
  - Modularity
  - Control Hierarchy
  - Structural Partitioning
  - Information Hiding
  - ...
- In short, your app needs to run on cloud basis

Main Focus of Cloud Based App

- Efficiency or cost reduction
- Data security
- Scalability or flexibility
- Mobility
- Disaster recovery
- Control
- Competitive Edge



https://www.globaldots.com/blog/cloud-computing-benefits

Choosing “Cloud”

- Public vs Private vs Hybrid Cloud
- “Vendor” dependent or open source

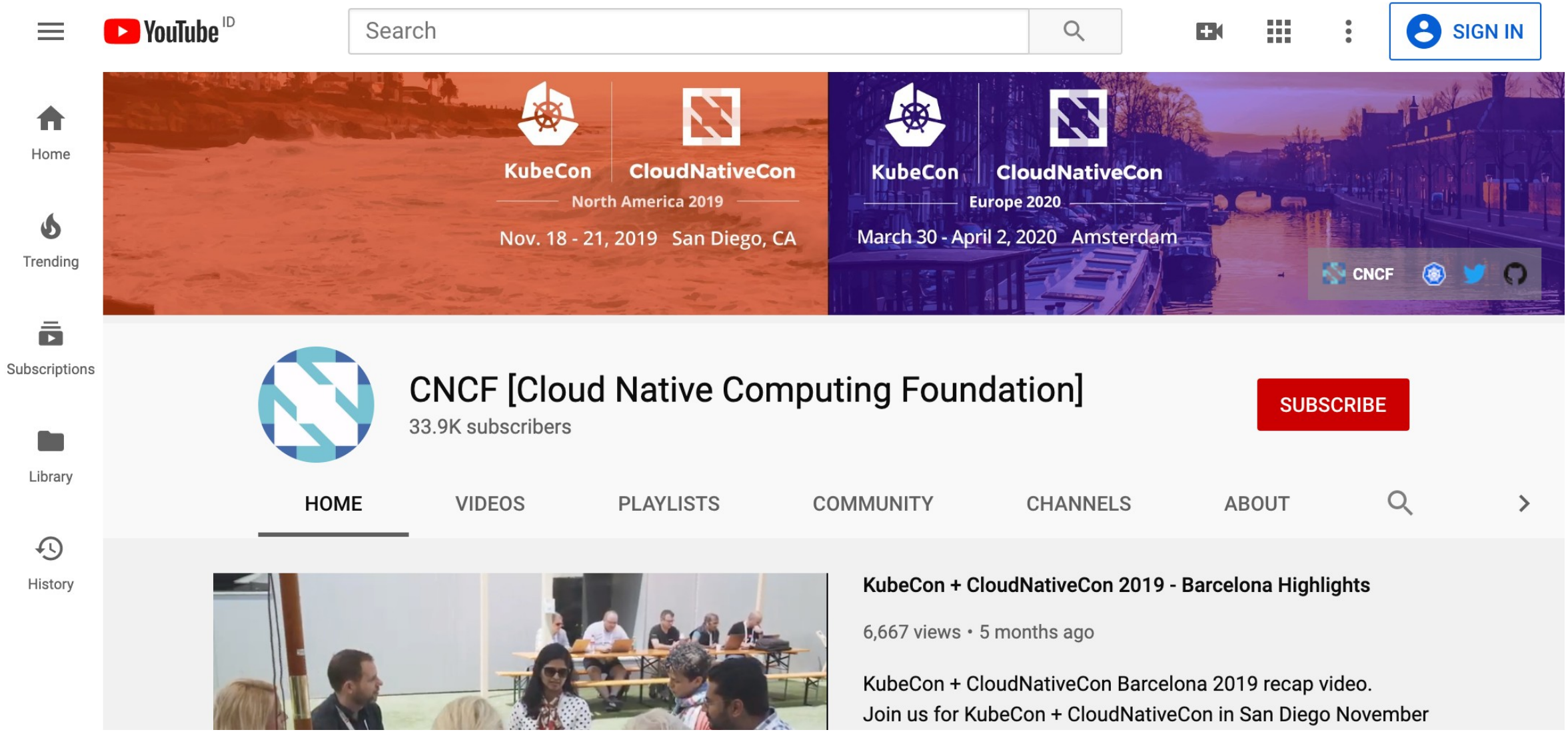
- Moving to public cloud is not always good choice but often

Cloud Native?

Principles of Cloud Native Architecture

- Application is always changing
- Application is designed with automation
- Application is designed for smarter with “state”
- Application that compatible with managed services
- Practically designed with good intuition of security

https://cloud.google.com/blog/products/application-development/5-principles-for-cloud-native-architecture-what-it-is-and-how-to-master-it



Source: https://www.youtube.com/channel/UCvqbFHwN-nwalWPjPUKpvTA screenshot at Nov 15th, 2019

Why CNCF?

- Their projects are under a foundation
- Modern best practices
- Project documentations are complete and easy to understand
- Give all benefits of Cloud Native

Kubernetes

Orchestration

Prometheus

Monitoring

Envoy

Network Proxy

CoreDNS

Service Discovery

containerd

Container Runtime

Fluentd

Logging

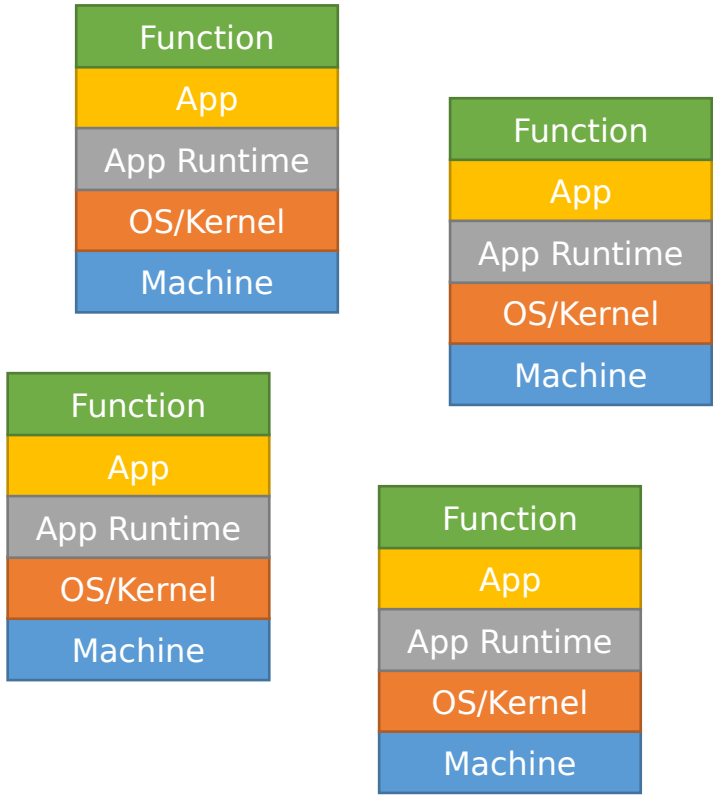
Jaeger

Distributed Tracing

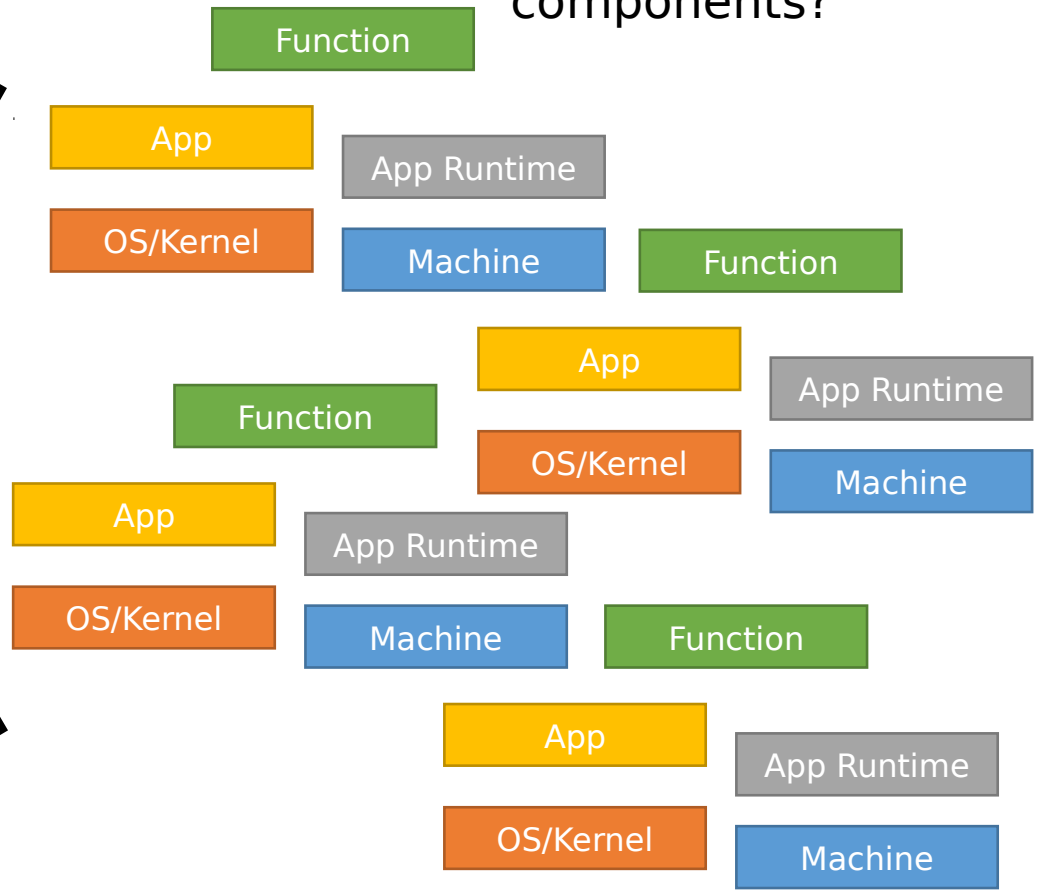
Vitess

Storage

Old way application shipment

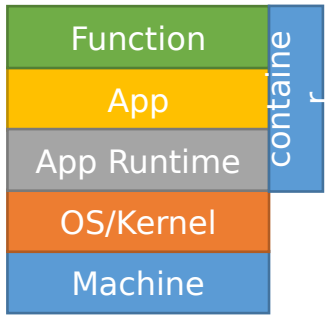


How to manage deployment of those components?



Containerization

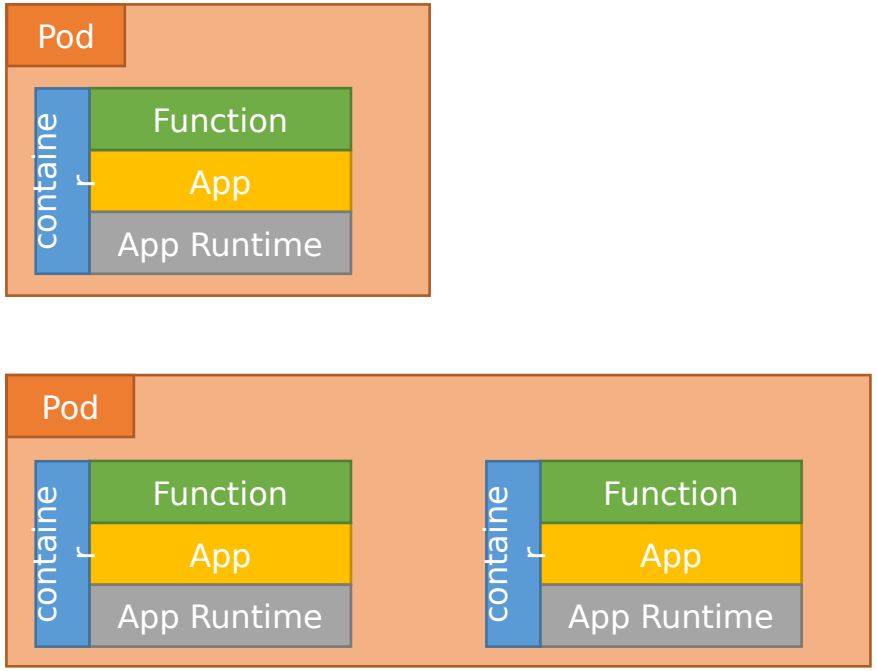
- Simplify packaging for deployment
- Virtualizing with shared kernel



Kubernetes: Container Orchestration

- Automatic configuration, coordination, and management of containers
- Previous name: Borg
- Building blocks:
  - Container
  - Pod
  - Replica Set
  - Service
  - Node
  - Master

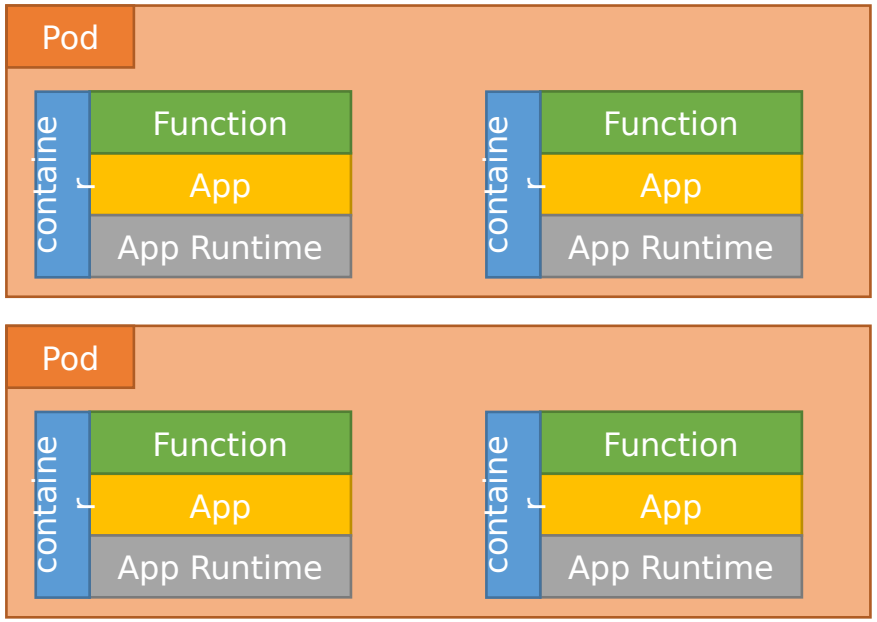
Kubernetes: Pod



- Abstraction of group of containers
- Basic unit for scheduling

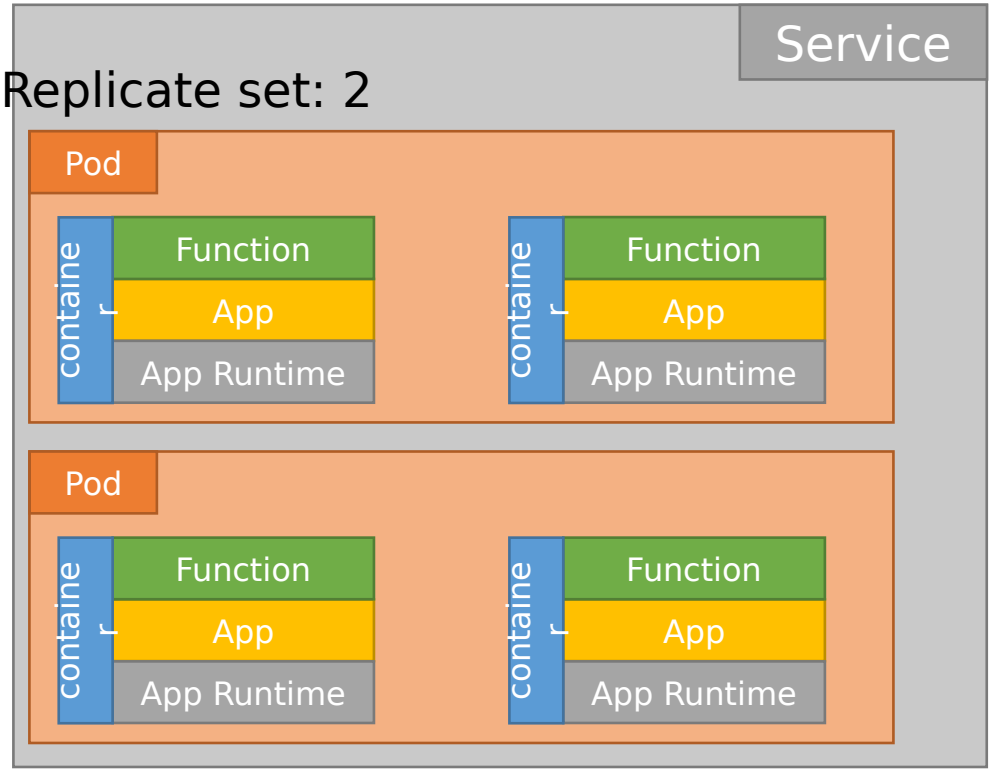
Kubernetes: Replica Set

Replicate set: 2



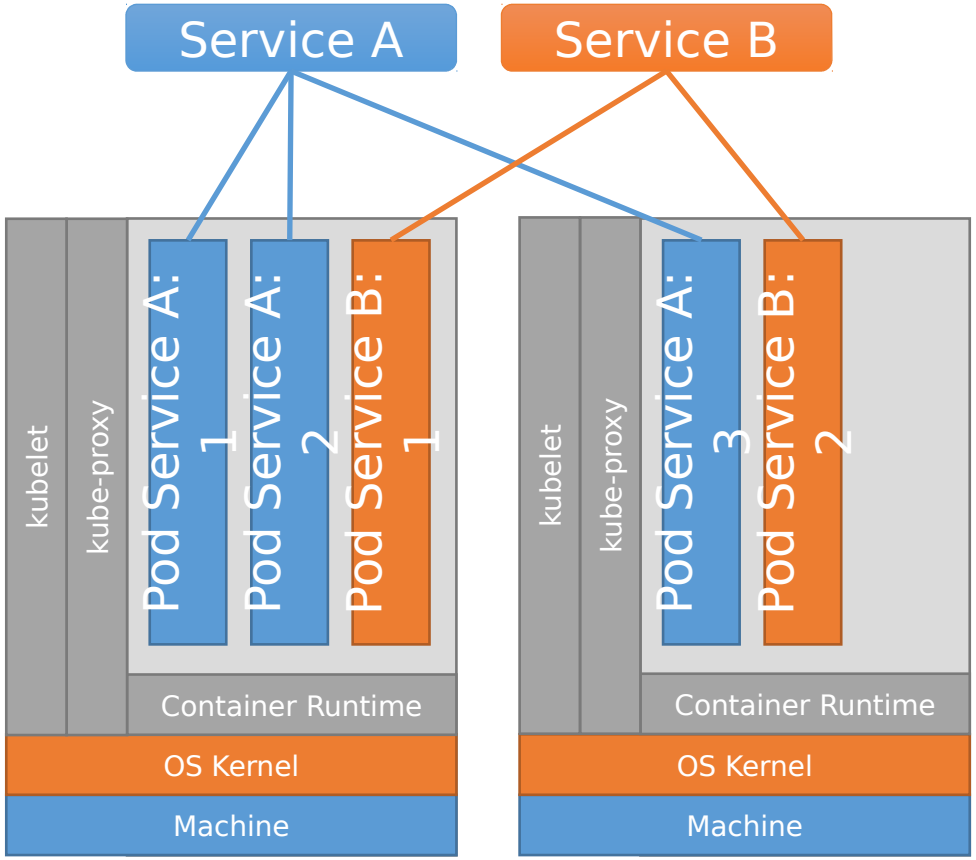
- Total pod replica for a service
- Use for maintain availability

Kubernetes: Service



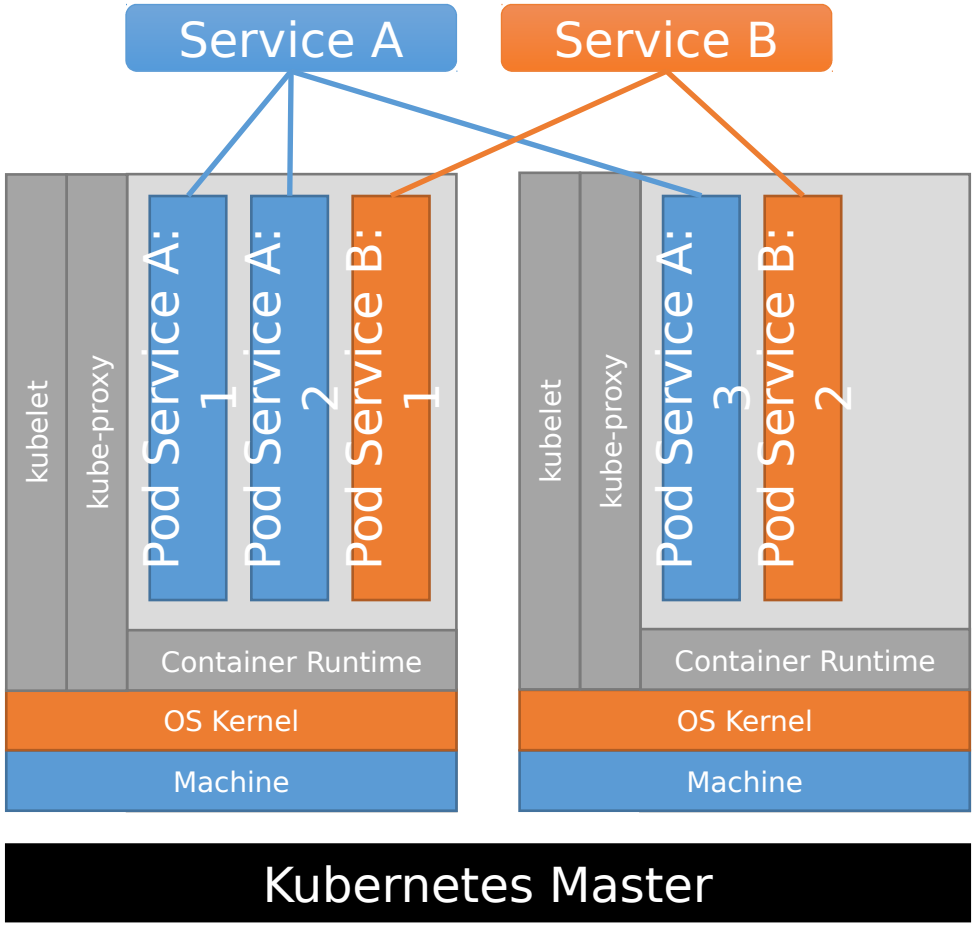
- Set of pods that work together
- Natively applied:
  - Load balancer
  - Service discovery
- Automated rollout and rollback

Kubernetes: Node



- Worker machine of Kubernetes
- Previously known as **minion**.
- Components inside a node at least:
  - container runtime
  - kubelet
  - kube-proxy
- Additional:
  - fluentd

Kubernetes: Master



- api-server that works like controller
- Components inside master:
  - kube-controller-manager
  - cloud-controller-manager
  - kube-apiserver
  - etcd
  - kube-scheduler

Impact to Software Design

- Introducing maintained microservices
  - Strong loosely couple
  - Designed for HA
  - Elastic up and down, reduce expenditure
- Use network massively. API everywhere.
- Forcing services to run stateless, state is saved in centralized/clustered storage
- Forcing to containerized everything
  - Most modern tools support containerization
- Testing and debugging become harder

Monitoring

Statistic: performance, resource usage, cost and security.

Prometheus

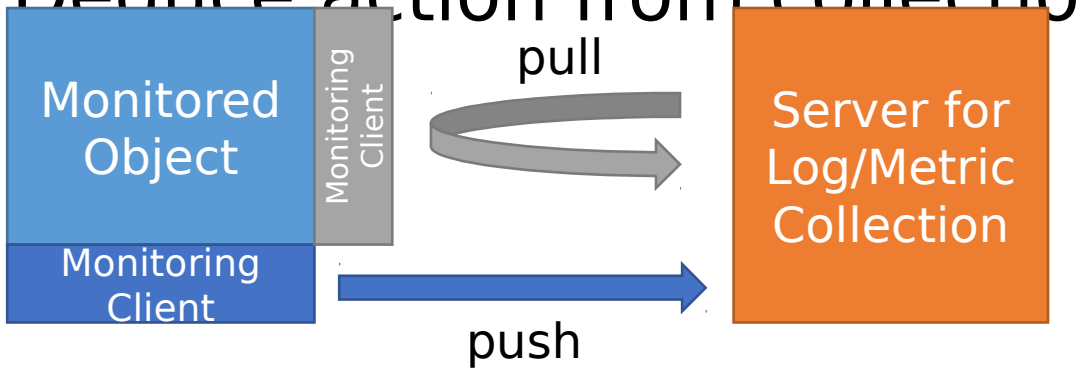
Monitoring

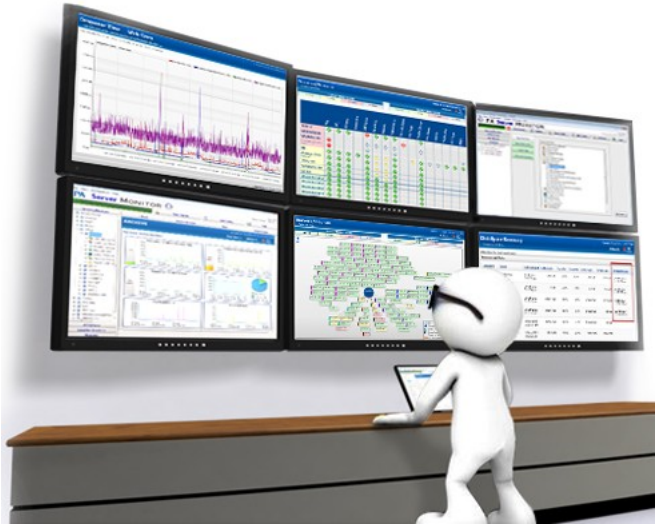
Grafana

- Designed to complement with monitoring usage
  - Time Series
  - Multidimensional
  - Support push and pull metrics
  - Support alert: PagerDuty, email, etc
- Compatible with Grafana for visualizing metrics
- How about monitoring for security?

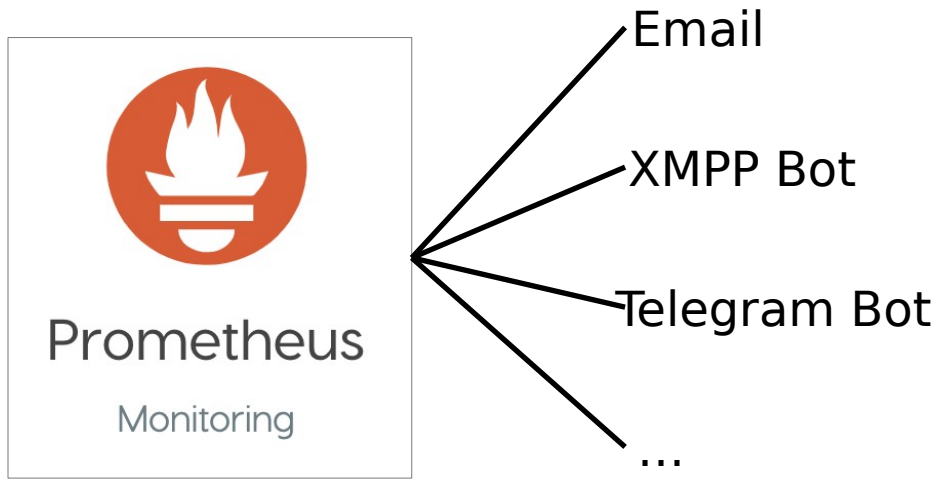
Log/Metric Collection

- Monitoring always about collecting information. In this case log or metrics.
- Heavily use time series record
- Deduce action from collection





Monitoring



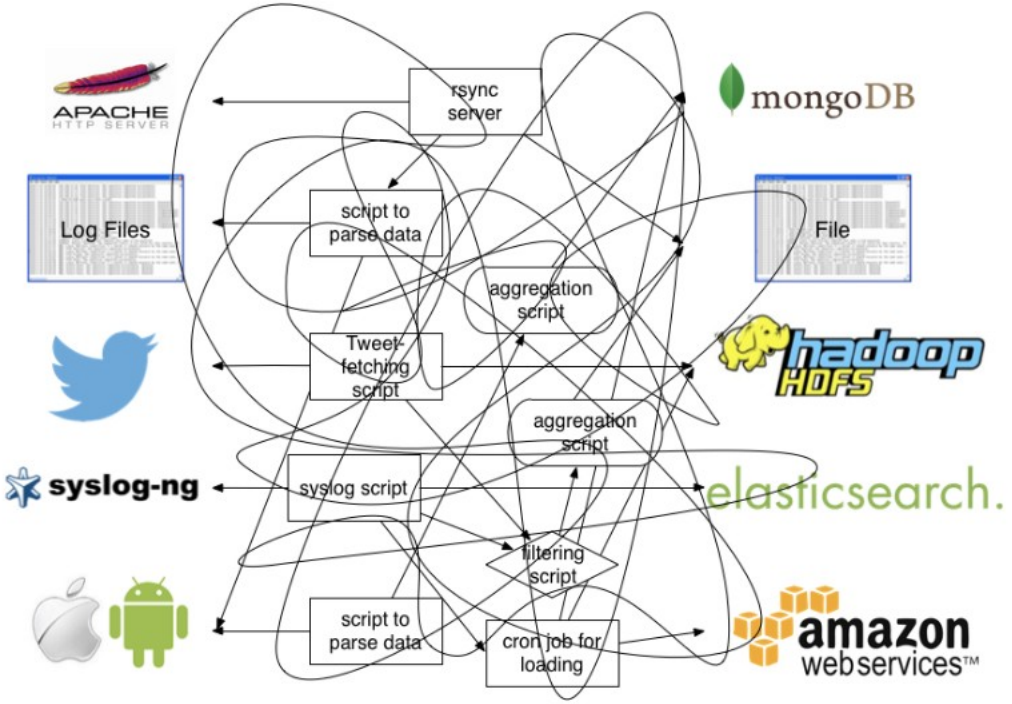
- Mainly used for monitoring performance and resource usage
- Deduce from collection:
  - Scaling strategy
  - Cost reduction
- Integration for automation: webhook alert

Logging

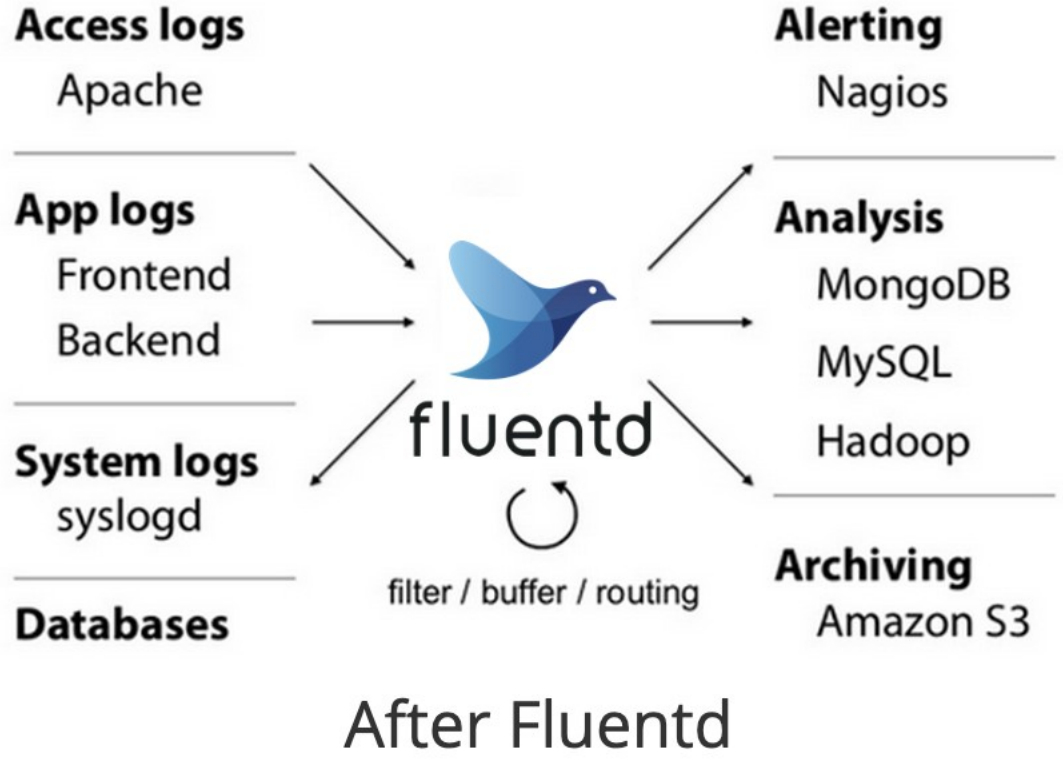
Record activity of running software

- Another terms: journal, records, footprints
- A log item usually represented as string.
- Use strategy to separate log like categorization, levelling, context, etc.
- FluentD: Log transformer and collector
- The most complete compatible integration

Logging



Before Fluentd

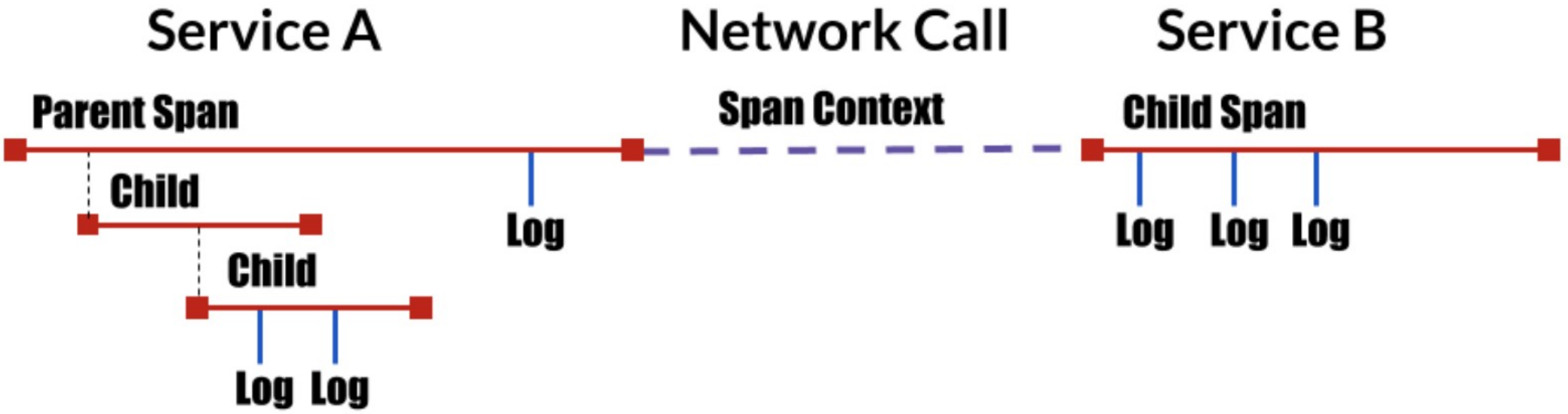


After Fluentd

Debugging and Tracing

- Exploiting logging strategy
- Tracing in distributed services is hard
  - Classic stack-trace is not enough
  - Complex architecture make debugging harder
- Need standardize how to trace in distributed services: OpenTracing
- Competitor: OpenCensus

OpenTracing: Concepts



Source: https://opentracing.io/docs/overview/

Summary so far

- Kubernetes mostly provide all Cloud-Native application benefits: rapid changes, deployment strategies, elastic scaling, intuitive security
- Compatible with microservices design
- Monitoring, logging and distributed tracing provides brilliant insight how “well” is the software design, especially when reaches medium scales.

More...

- Routing: Envoy (network proxy), CoreDNS (service discovery), Linkerd (service mesh)
- Security: Open Policy Agent (policy), Notary (signing)
- Storage: etcd, TiKV, Rook
- Distributed tracing: Jaeger
- Messaging: NATS
- Serverless: CloudEvents

- 99.co is leading property tech company
- 99.co helps the 99% make the best property decisions

  by empowering home-seekers, home-owners and agents with best technology, data and design.

- We are using public cloud IaaS: AWS and GCP

Culture

i. Relentless

/:/ showing no abatement of severity, intensity, strength, or pace

ii. Resourceful

/:/ capable of devising ways and means

iii. Respect

/:/ to hold in esteem and honor

Engineering Team

- 2 countries, 3 offices
  - SG: Ayer Rajah Crescent
  - ID: Jakarta (REA/rumah123) and Bandung (urbanindo)
- Bandung office
  - 30 people: 2 PM, 1 SM, 2 UI/UX, 3 QAs, 22 Engineers
  - 7 of them still in school
- We value skills, grade/school is additional

We are Hiring

- Infra & DevOps Engineer
- Product Manager
- Software Engineer (Back End)
- Senior Software Engineer (Fullstack)
- iOS Engineer
- Android Engineer
- UI/UX Designer
- Data Analyst