# How Kubernetes helps Devops

Sreenivas Makam
Partner Engineer @Google Cloud
@Kubernetes SriLanka meetup, 20th September 2018

# Agenda

- Definitions - Devops, Cloud native, Microservices
- Why Containers and Kubernetes?
- Kubernetes for Devops
  - Abstractions
  - Design patterns
  - Integrations
  - Extensions
- Use cases

# DevOps

From Wikipedia, the free encyclopedia

**DevOps** (a clipped compound of "development" and "operations") is a software engineering culture and practice that aims at unifying software development (Dev) and software operation

**Joshua Bernstein**
@quityourjoshing

Follow

Replying to @kelseyhightower @beevek @copyconstruct

DevOps is operational model where by software development principals are applied to operations

11:31 PM - 25 Apr 2017

**Cindy Sridharan**
@copyconstruct

Follow

So after all, turns our SRE is just what one calls a "DevOps engineer"
How do I know? A Google SRE tells me so.
😇 docs.google.com/presentation/d …

## SRE

Ultimately responsible for the reliability of `google.com`

Less than 50% time spent on operations,

More than 50% on engineering reliability and automation

**SRE = DevOps Engineer**

Cloud Native is structuring **teams**, **culture** and **technology** to utilize **automation** and **architectures** to manage **complexity** and unlock **velocity**.

Joe Beda, Heptio

*The term "Microservice Architecture" has sprung up over the last few years to describe particular way of designing software applications as suites of independently deployable services. While there is no precise definition of this architectural style, there are certain common characteristics around organization around business capability, automated deployment, intelligence in the endpoints, and decentralized control of languages and data.*

- Martin Fowler

**Kelsey Hightower**

@kelseyhightower

Following

I'm convinced the majority of people managing infrastructure just want a PaaS. The only requirement: it has to be built by them.

4:08 PM - 11 Apr 2017

# Goals of Devops

- Infrastructure as code
- Immutable infrastructure
- Faster release cycles
- Automated CI/CD pipeline
- Provide visibility into production application

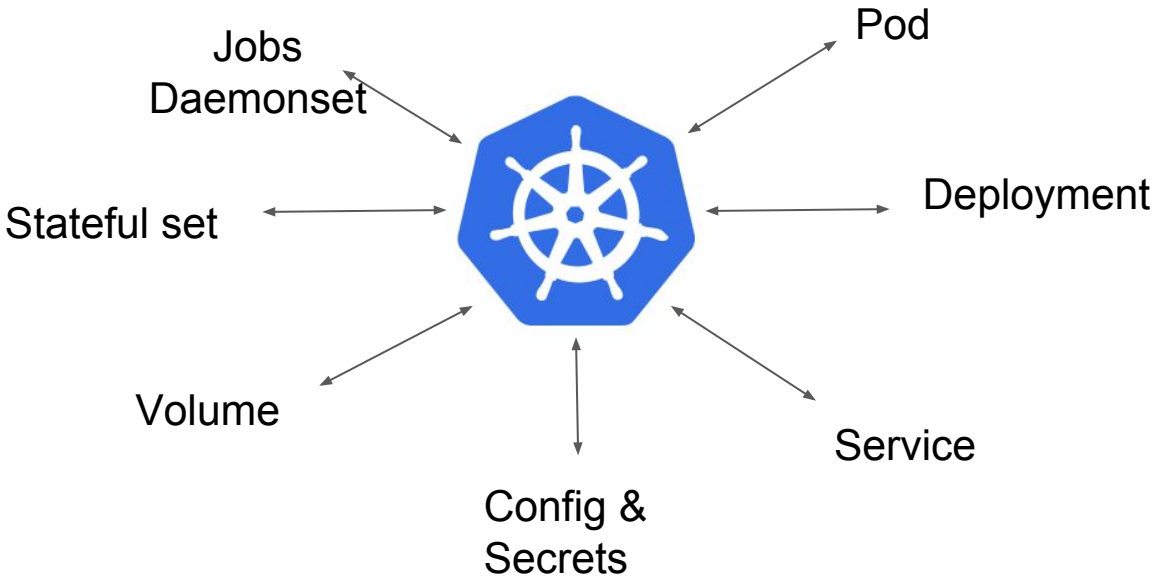
# Why Containers for Devops

- Self contained
- Portability
- Decoupling from machine
- Image immutability
- Faster development
- Faster deployment

# Why Kubernetes for Devops

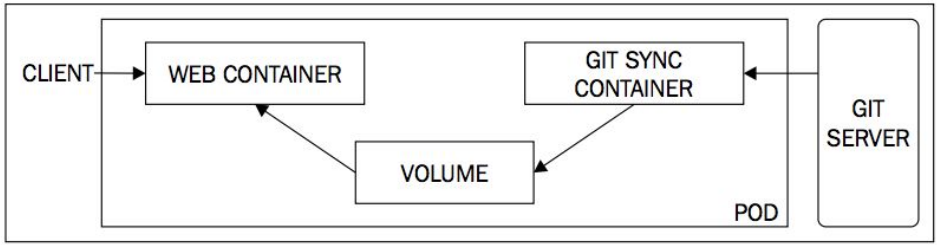
- Decoupling from infra
- Autoscaling
- Autohealing
- Automated rollout and rollbacks
- Abstractions that are cloud native and microservices friendly
- Extensible
- Open-source
- Integrates well with other Devops tools
- Supports imperative and declarative configuration

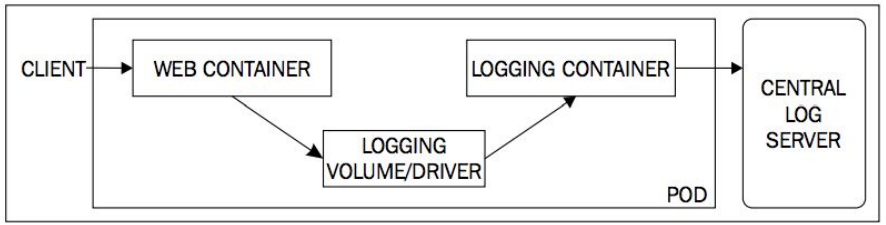
# Kubernetes abstractions



# Pod patterns - Sidecar

Sidecar containers extend and enhance the “main” container

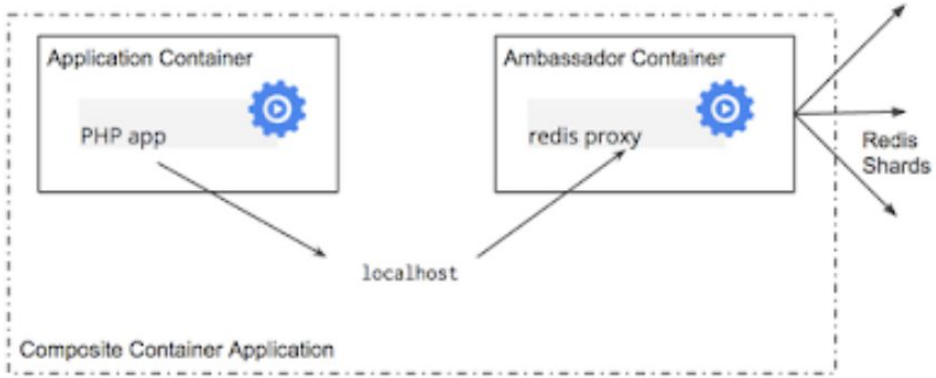



**Other examples:**
Istio envoy proxy
Monitoring
Database config

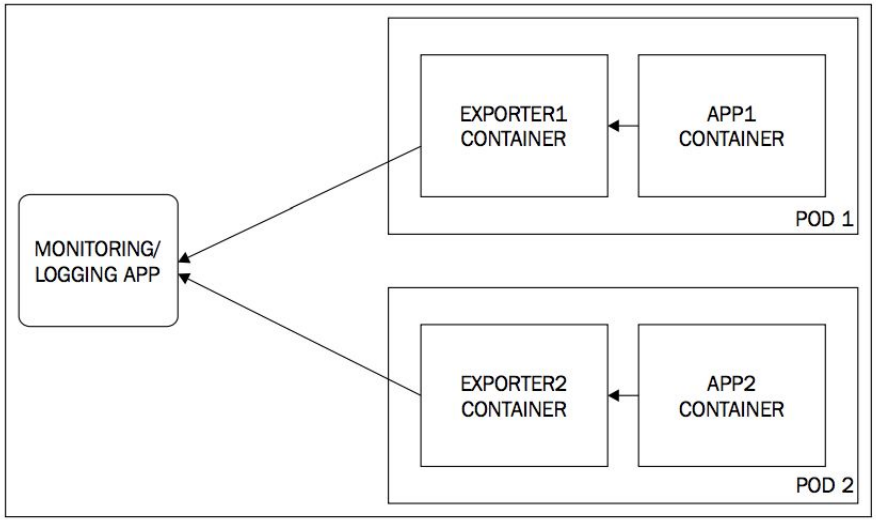
# Pod patterns - Ambassador

Ambassador containers proxies a local connection to the world and hides the complexity to access the service.



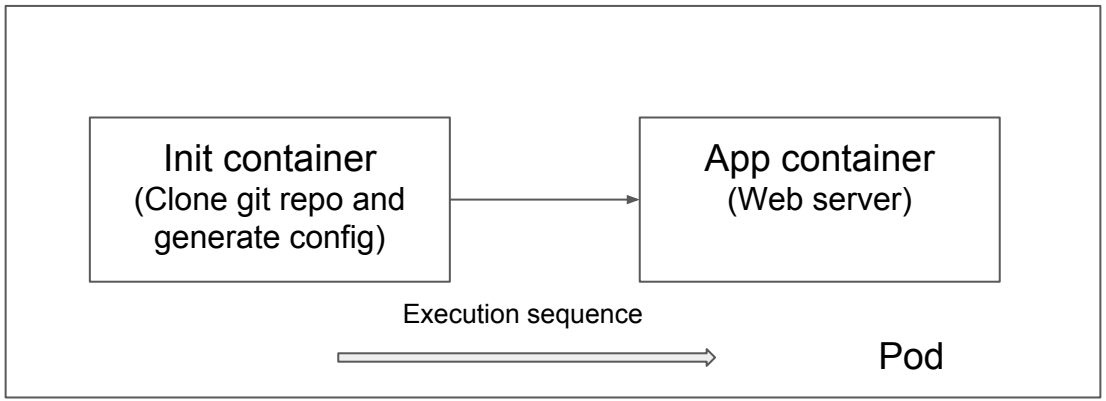
# Pod patterns - Adapter

Adapter containers standardize and normalize output.

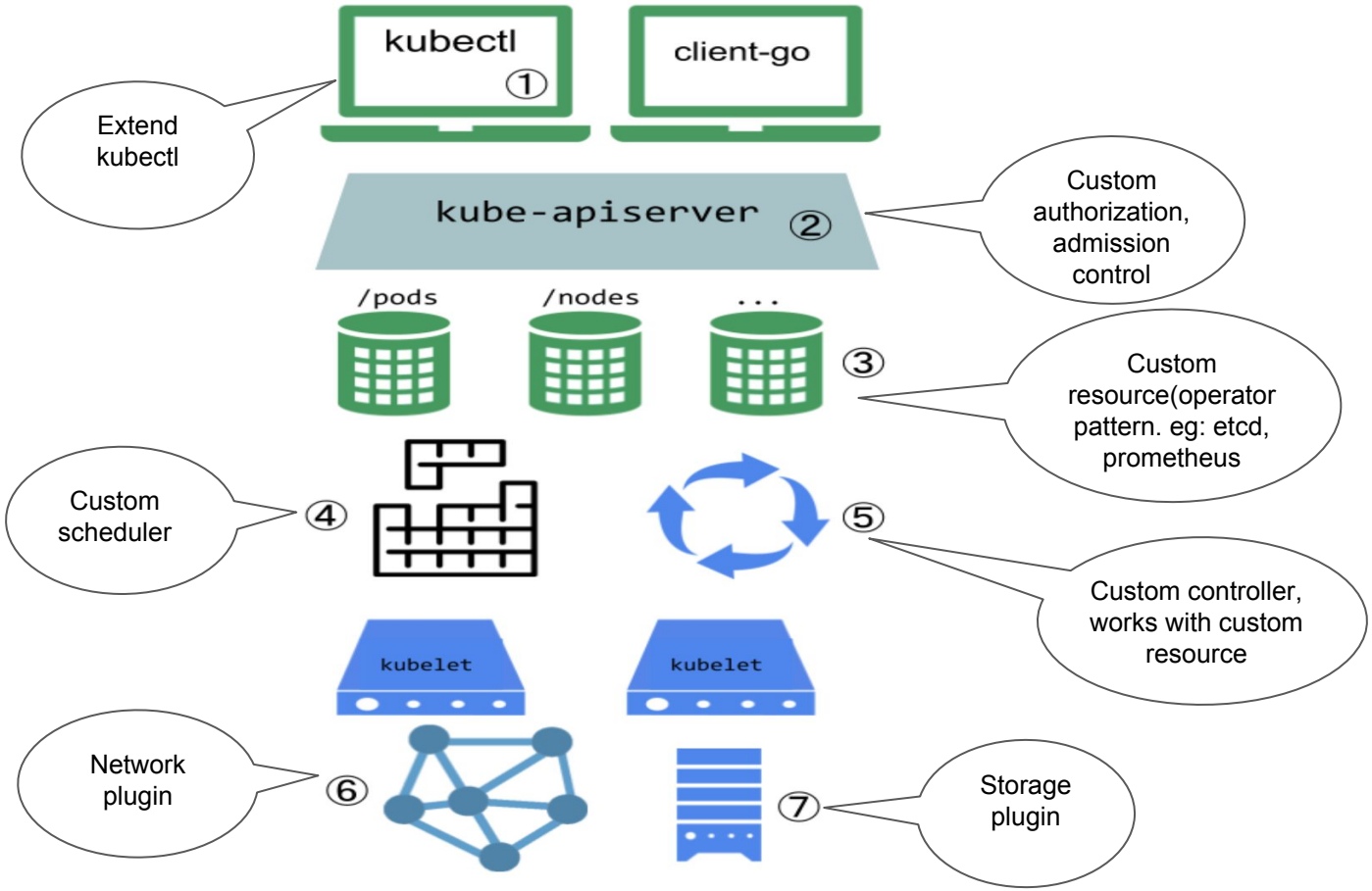


# Pod patterns - Init containers

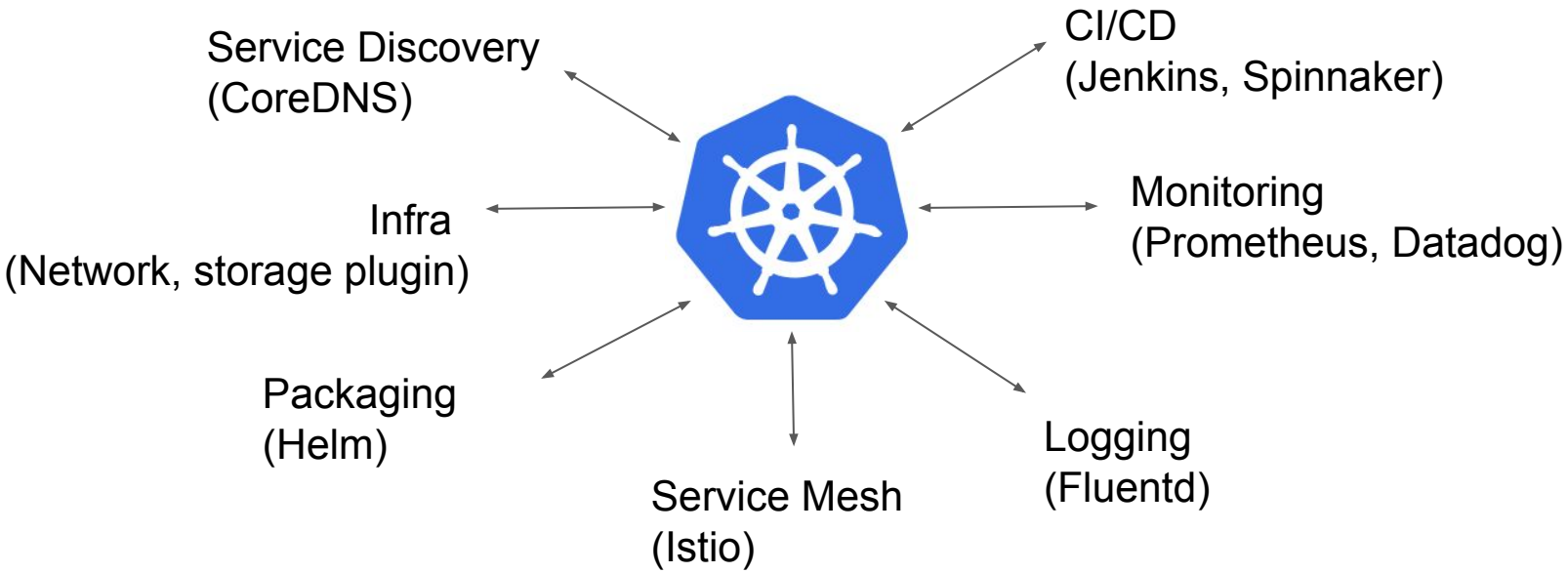
Specialized containers that runs to completion before application containers in a pod can get started. This enforces sequence.



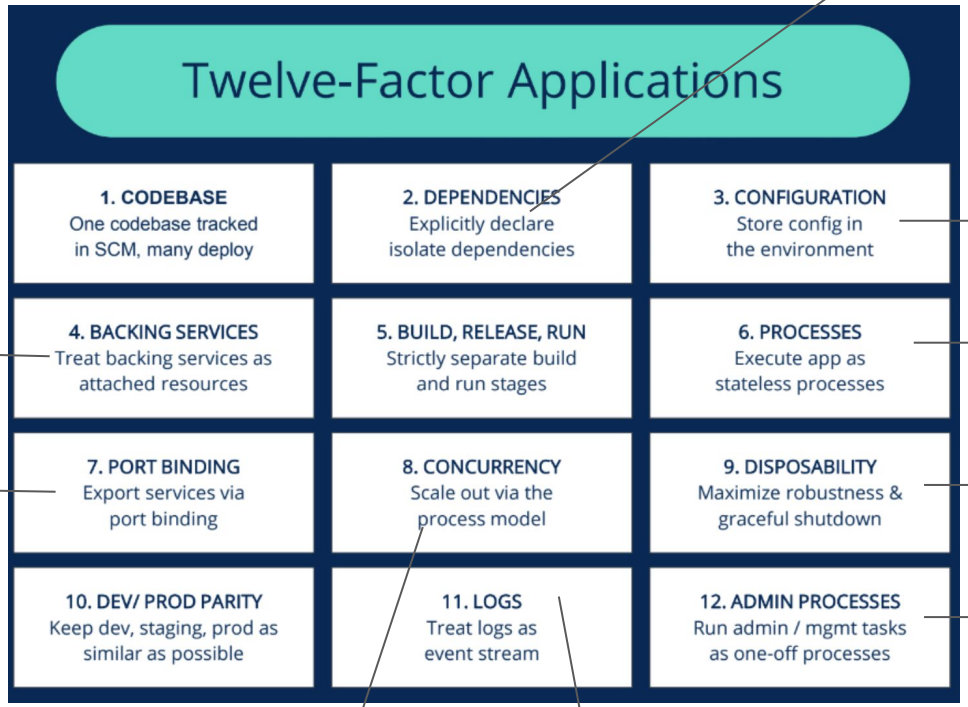
# Kubernetes extension points



# Kubernetes Integrations



# Map Twelve factor applications to Docker/Kubernetes




Single app defined using Dockerfile and multiple apps done using deployment

Config map and secrets

Service abstraction and discovery

Stateless containers, stateful dataset where needed

Services provides different options for port bindings

Autohealing

Many ways to create and manage clusters(cloud provider, kops, kubeadm)

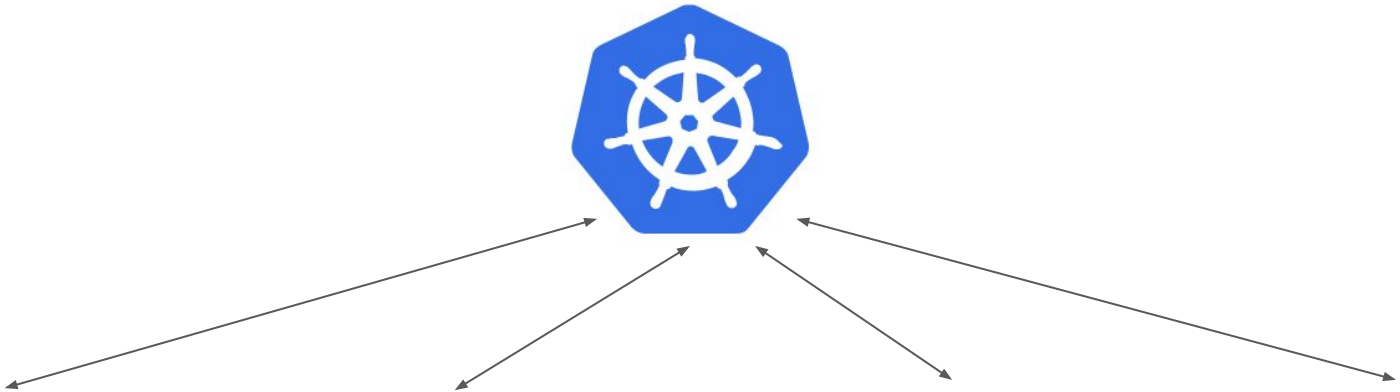
Autoscalar support is comprehensive

Centralized log management with third party integrations possible

# Kubernetes for Devops Use cases

- Infrastructure for cloud native applications
  - Applications deployed as cluster of microservices
  - Automate CI/CD pipeline through development, staging and production
  - Rolling upgrades
  - Integrated logging, monitoring and tracing
- Migrating legacy applications to Container platform
  - Efficient resource utilization
  - Faster feature rollout
- Hybrid/Multi-cloud deployments
  - Not locked into single provider
  - Use on-premise and cloud combination

# Kubernetes Distributions



Open source

Distributions
(Redhat, Canonical)

Managed
(GKE, AKS, EKS)

Private PaaS
(Openshift, Deis)

# References

- Kubernetes patterns video
- Kubernetes patterns slides
- Building Cloud native applications with Kubernetes and Istio - Kelsey
- Designing cloud native applications
- Extending Kubernetes