Introduction to Google Kubernetes

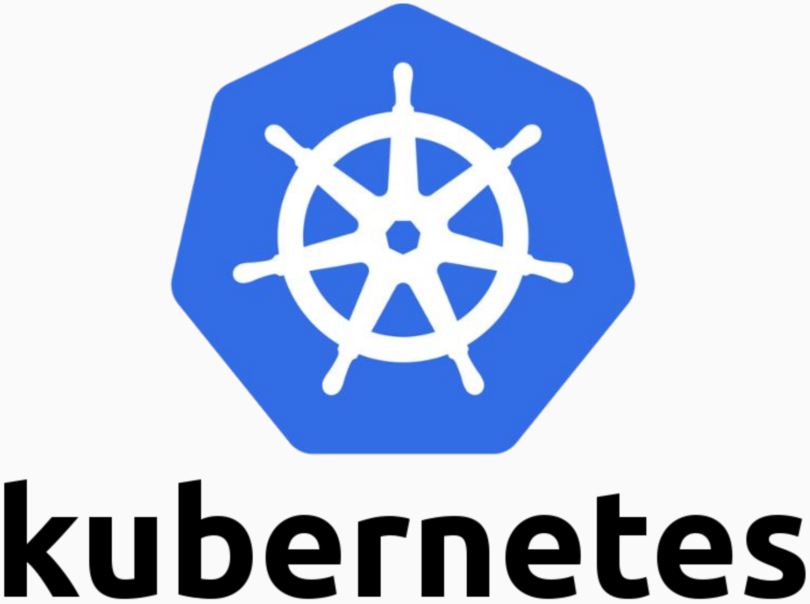
Pascal Liniger, infix development gmbh
Docker Meetup 17.02.15

About me

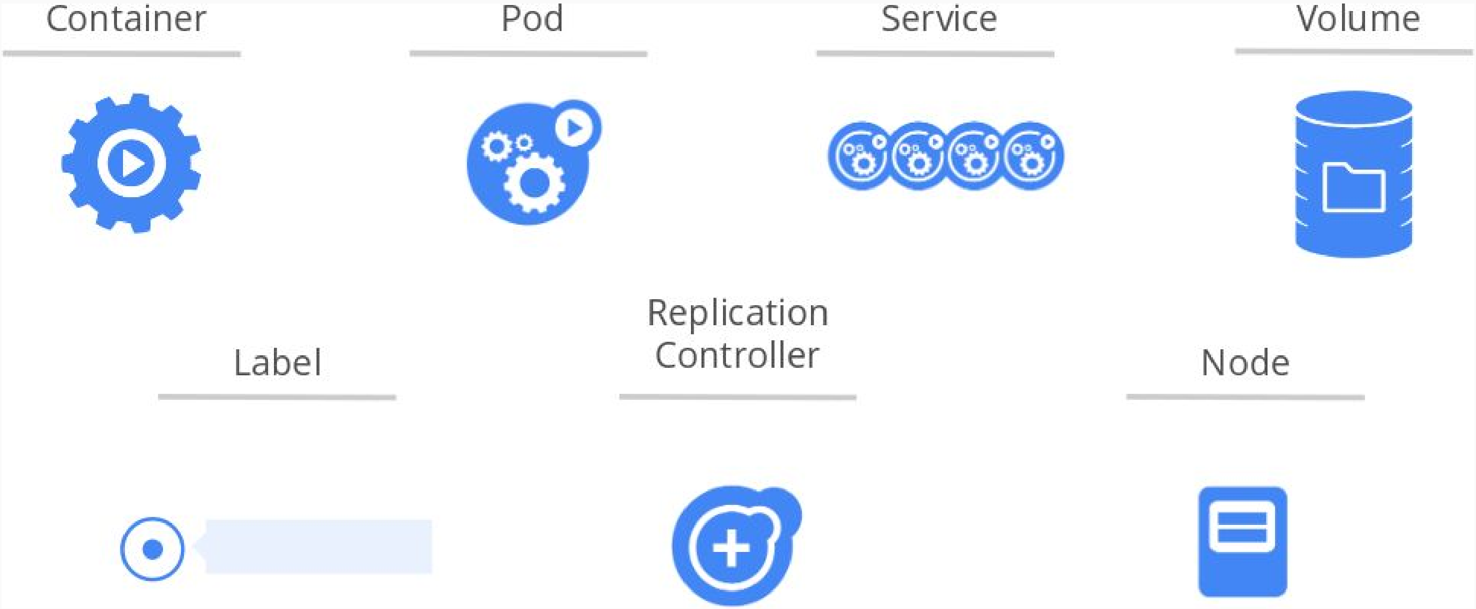
- Pascal Liniger <pascal.liniger@infix.ch>
- infix development gmbh
- Linux and storage specialist
- Cloud enthusiast
  - Docker
  - Kubernetes
  - etc.

Google Kubernetes

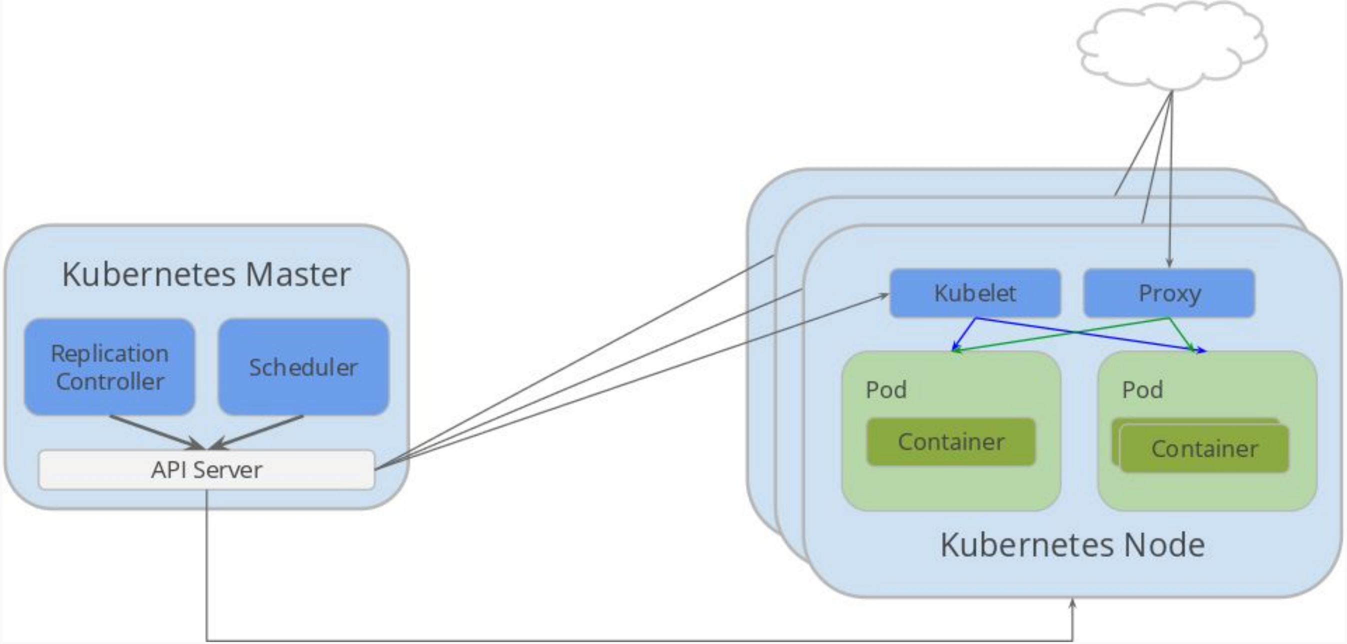
- Greek for “*Helmsman*”
- Orchestrator for **Docker** containers
- Uses Docker to run containers, manage images etc.
- Inspired and based on Google’s internal systems (Borg).
- **Open source**, written in Go



Concepts


Container
Pod
Service
Volume
Label
Replication Controller
Node

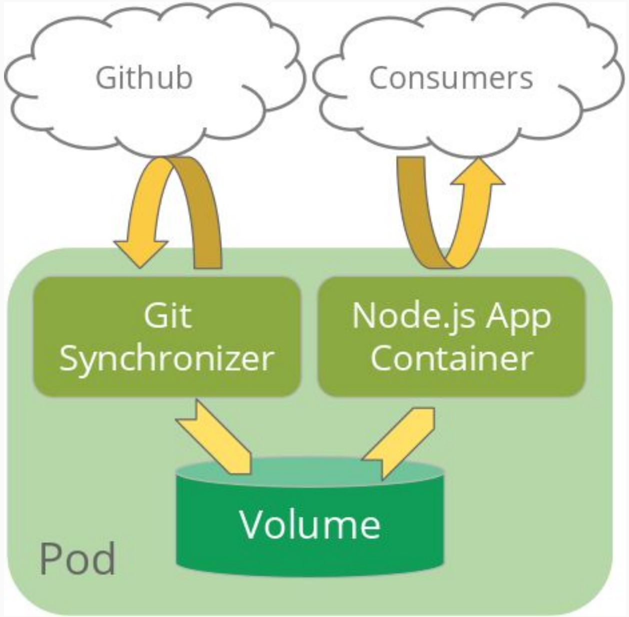
Kubernetes Cluster / Nodes



Pods

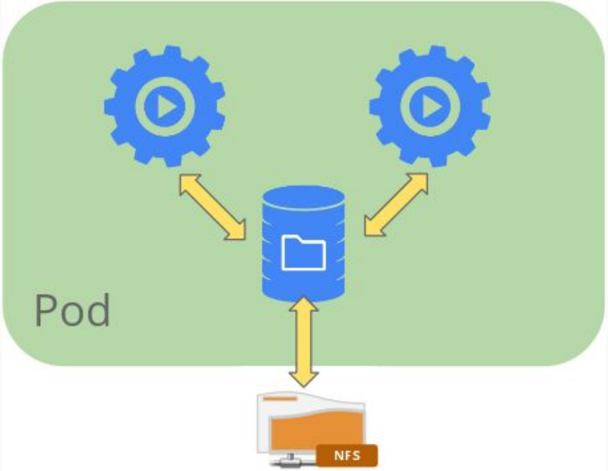
- Atomic scheduling unit for containers
- **Temporary**: Can die and be replaced
- One pod can contain multiple docker containers
- Containers in pod are tightly coupled
  - Shared IP, namespace, volumes etc.
  - Live and die together

Similar to multiple services running on the same machine



Volumes

- Storage bound to a pod
- Looks like a directory to the containers
- Where and how the data is stored is determined by the **VolumeType**
- Many types available:
  - EmptyDir
  - HostPath
  - nfs
  - glusterfs
  - etc.

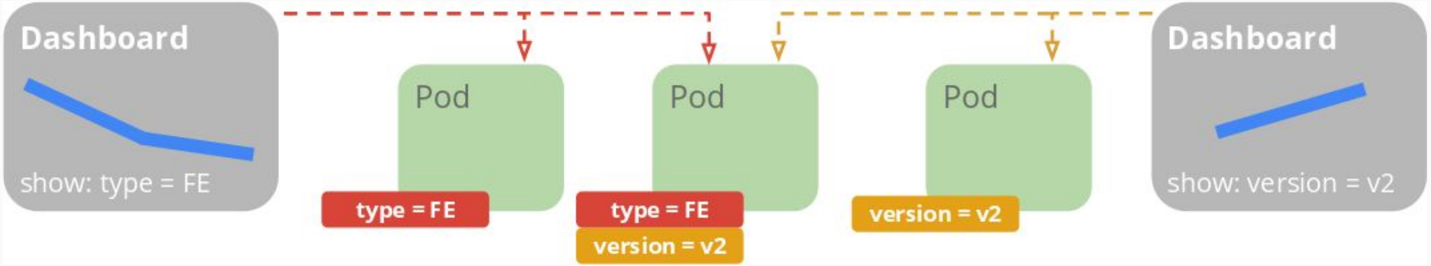


Labels

- Label and group pods
- Used as **selectors** by replication controllers, services and other objects.

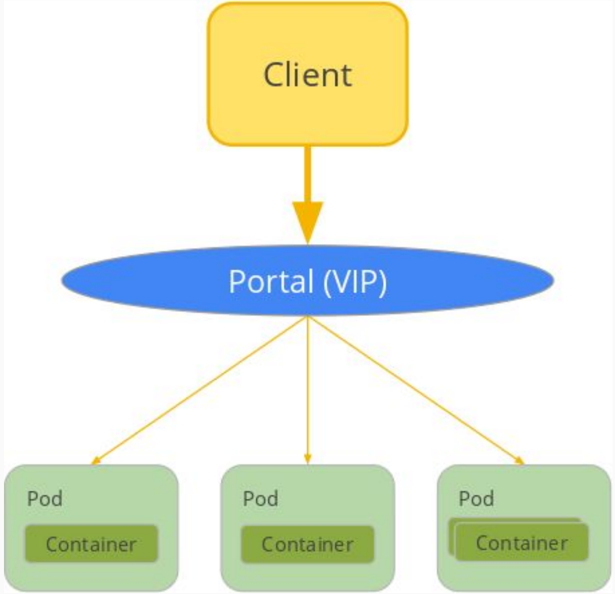
Examples:

- “version”: “v1”, “version”: “v2”
- “release”: ”stable”, “release”: “beta”



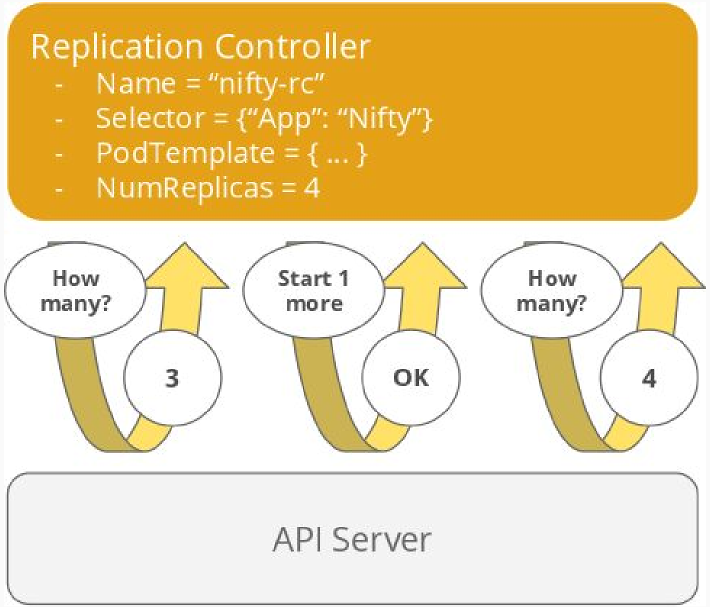
Services

- A group of pods that **act as one**
- Gets a **stable** virtual IP, port and DNS name
- VIP only accessible **inside the cluster**
- Requests get distributed to the pods
- Hides complexity
  - Non-native apps can just access the VIP



Replication Controllers

- **One job**: ensure that N copies of a pod are running
  - If too few, start new ones
  - If too many, kill some
- Selection of pods using **labels**



Replication Controller

- Name = “nifty-rc”
- Selector = {“App”: “Nifty”}
- PodTemplate = { ... }
- NumReplicas = 4

And more

- Jobs
- Horizontal Pod Autoscaler
- Deployments
- Annotations
- Persistent Volumes and Claims
- Secrets
- etc.

Demo / Cassandra on Kubernetes

- Create a replication controller and a service
- Scale the replication controller
- Distribute a new version of the database (rolling update)
    - While ensuring data availability
    - **Without persistent volumes**
    - Data stays alive by replicating