11.6.2018

München / Microservice Summit

Kubernetes - the abstract cloud

Jörg Müller - @joergm

INNOQ

- architecture, development, devOps
- focus on platform & infrastructure

Jörg Müller
Principal Consultant
innoQ Deutschland GmbH

joerg.mueller@innoq.com
@joergm

What to expect?

- Overview, Ways to get Kubernetes and basic concepts
- Core abstractions
- Internal Architecture
- Deploying complex applications
- Production readiness

Timeslots

- 9:30 - 10:30 Slot 1
- 10:30 - 11:00 Coffee break
- 11:00 - 12:30 Slot 2
- 12:30 - 13:30 Lunch
- 13:30 - 15:00 Slot 3
- 15:00 - 15:30 Coffee break
- 15:30 - 17:00 Slot 4

Prerequisites & rules

- Kubernetes know-how not necessary, but it doesn't hurt
- Basic knowledge about Docker is assumed
- Demos can be followed but don't have to
  - github.com/JoergM/kubernetes_workshop_demos
- Please ask questions!

Kubernetes Overview

Docker Recap

Docker container at runtime

- Isolated process
- Separate file system
- Own network address and port space

Docker container - advantages

- Better isolation than package management on same machine
  - e.g. multiple versions of core libraries
  - not necessary to coordinate available ports
- Faster startup than virtual machine images
- Better resource usage compared to VMs

Docker images

- Standardized format
- Container hierarchies and difference file system
- Registries
- Unique name format (Registry/username/imagename:version)
- Simple Text-Format to create new images (Dockerfile)

Docker images - advantages

- Deployment format independent of implementation technology
- Same deliverable in all stages (Development, CI, Tests, Production)
- Container hierarchies allow simpler patch management
- Definition simpler than most package manager definitions

What is Kubernetes?

Kubernetes — adds to Docker

- Handling of multiple servers
    - Scheduling of containers
    - Networking
    - Failure handling
- Service Discovery features
- Many other useful abstractions for container interactions

Kubernetes - executive summary

- Kubernetes (K8s) is an open-source system for automating deployment, scaling, and management of containerized applications
- Marketing claim:
  - Planet Scale
  - Never Outgrow
  - Run Anywhere

Kubernetes — brief history

- Designed by Google, later donated to Cloud Native Computing Foundation
- Heavily influenced by Google's internal Borg system
- Code name: Project Seven
- Initial release: 7 June 2014 / 15 December 2015 (first stable version)

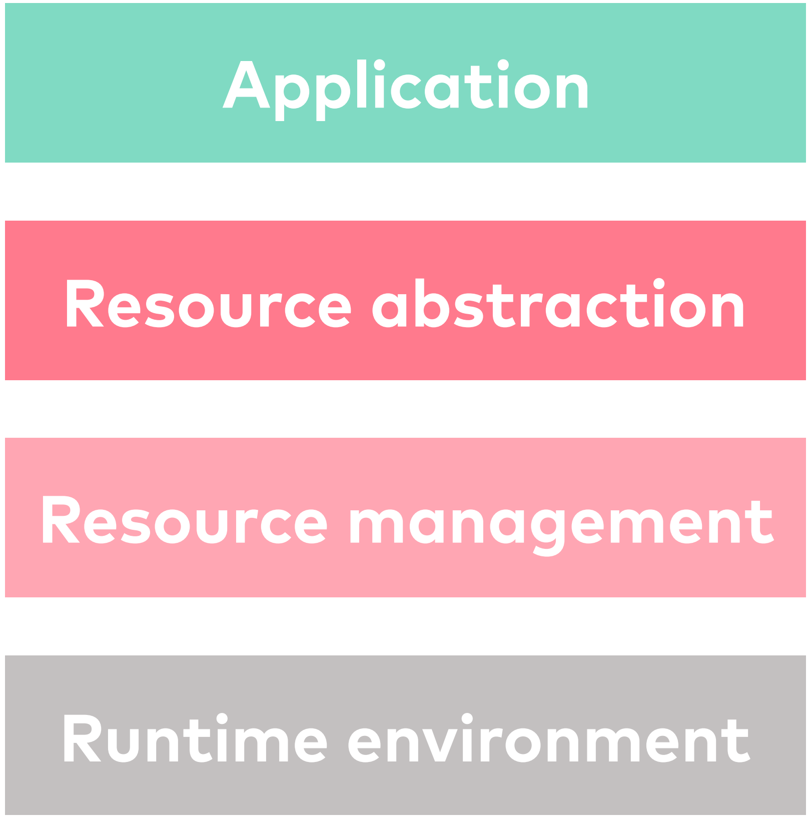
Why abstract cloud?

Why do we need one?

- Working on local machines
- Prevent Vendor lock in
  - Less specific Know How necessary
  - Easier to move
- Common way to automate complex setups
  - For inhouse applications
  - Also for software vendors

Kubernetes

- You define resources needed not machines or implementations
- Kubernetes manages resources
- Has a large base of runtime environments



Application

Resource abstraction

Resource management

Runtime environment

We tried that before …

- Virtual machines
- Configuration management (Puppet, Ansible, Chef)
- Terraform, CloudFormation
- PaaS
- …

Standing on Shoulders

- Container abstractions
- Googles experiences running Borg
- Focus on immutable infrastructure

K8s for microservices

Challenges

- Deployment
- Configuration
- Service Discovery
- Load Balancing
- Routing
- Resilience

Kubernetes

- Provides solutions for those challenges
- Is available everywhere
- Becomes more and more widespread
  - So developers know how to solve those challenges
  - Operations accepts and knows the solution
- Microservices infrastructure looses a lot of its horror

Ways to get Kubernetes

Local installation

Installing a simple Kubernetes on your notebook.

- Minikube
- Docker native

Online Tryout

Try Kubernetes without installing anything.

- https://www.katacoda.com/courses/kubernetes/playground
- https://labs.play-with-k8s.com/

By Cloud providers

Managed Kubernetes is now offered by all major cloud providers.

- Google Kubernetes Engine (GKE)
- Azure AKS
- IBM Cloud Kubernetes Services
- Amazon EKS (GA just started in us-east and us-west)
- Digital Ocean Kubernetes (coming soon)

Specialised Kubernetes providers

Offering managed Kubernetes on different plattforms.

Often including Support and On-Premise install.

- GiantSwarm
- Rancher
- Tectonic
- Kontena Pharos
- ...

PaaS Solutions

Plattform as a Service built on Kubernetes or offering Kubernetes services.

- RedHat OpenShift
- CloudFoundry Container runtime

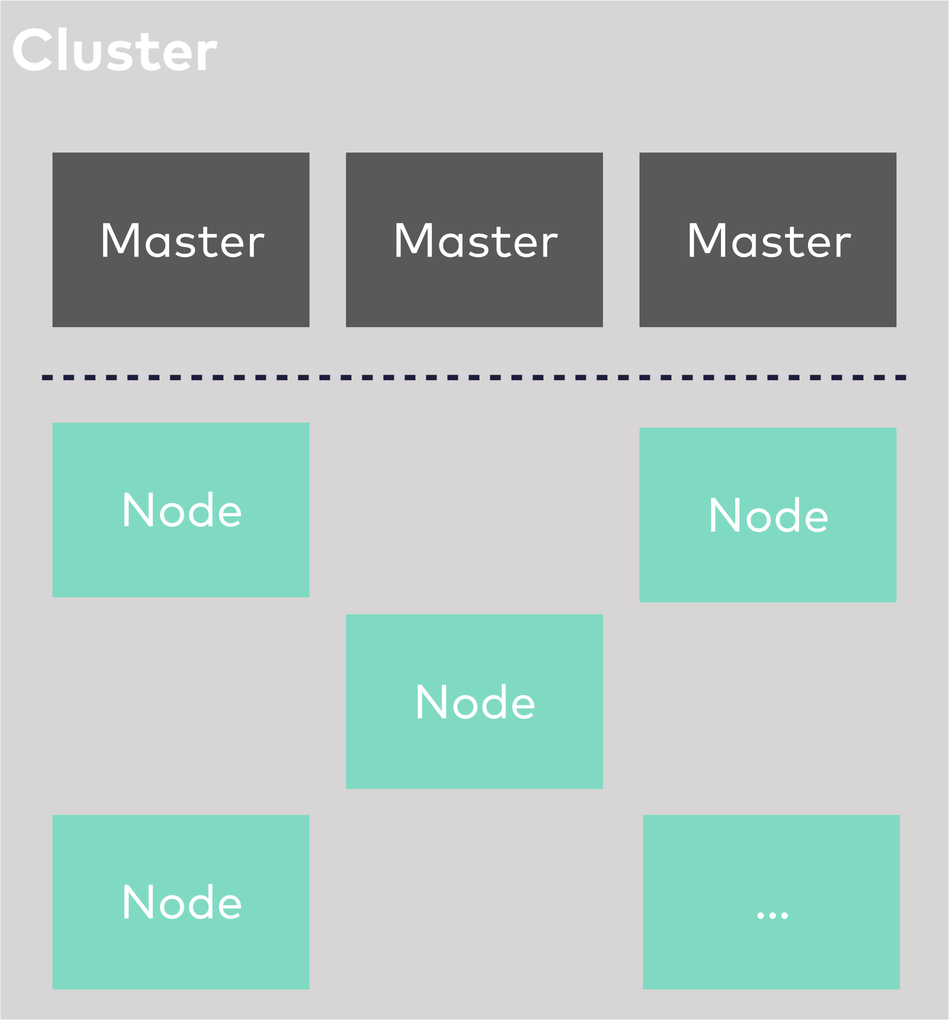
Self install

Finally a lot of options to install Kubernetes yourself on Cloud Providers or On-Premise.

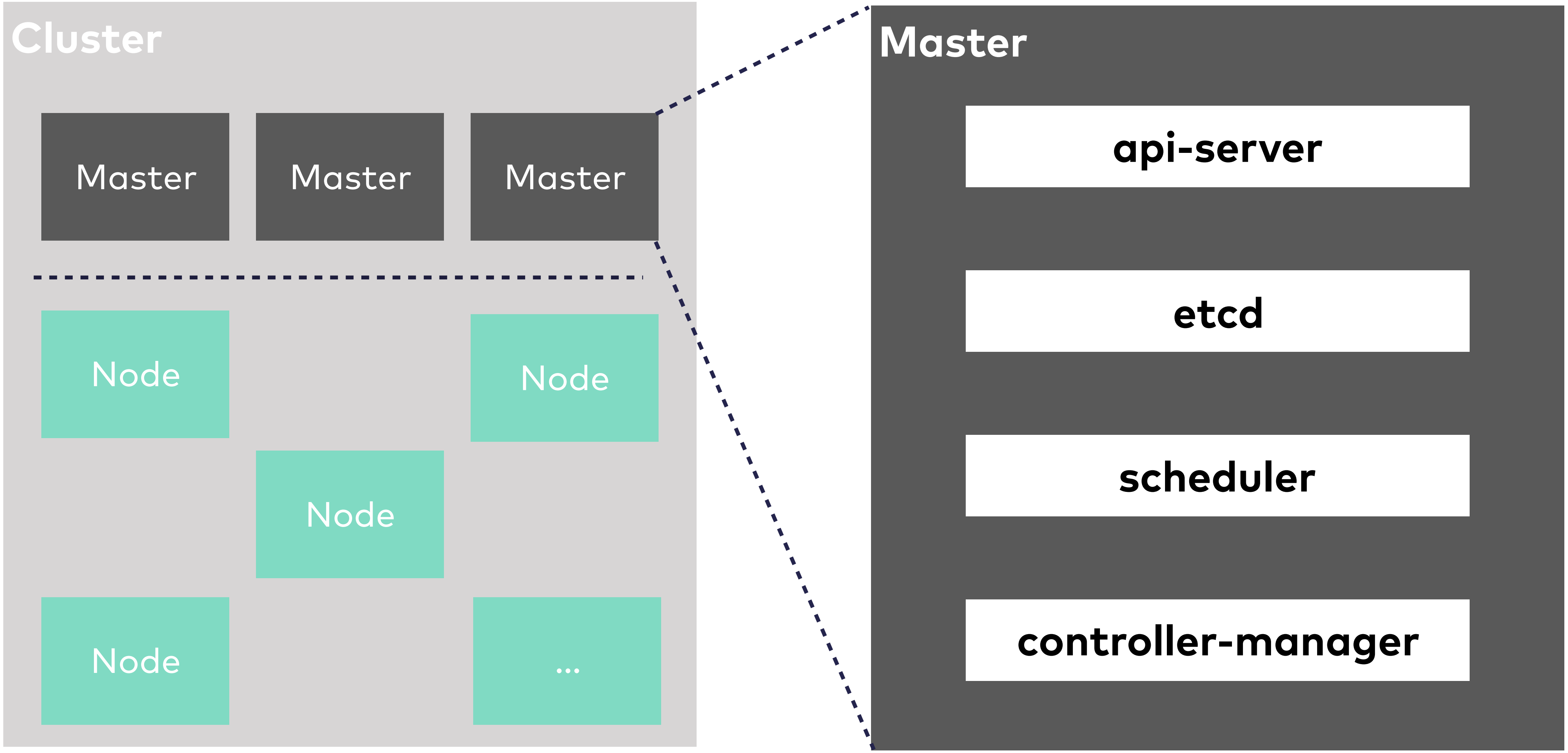
- kubeadm
- KOPS
- https://github.com/kelseyhightower/kubernetes-the-hard-way

Basic concepts

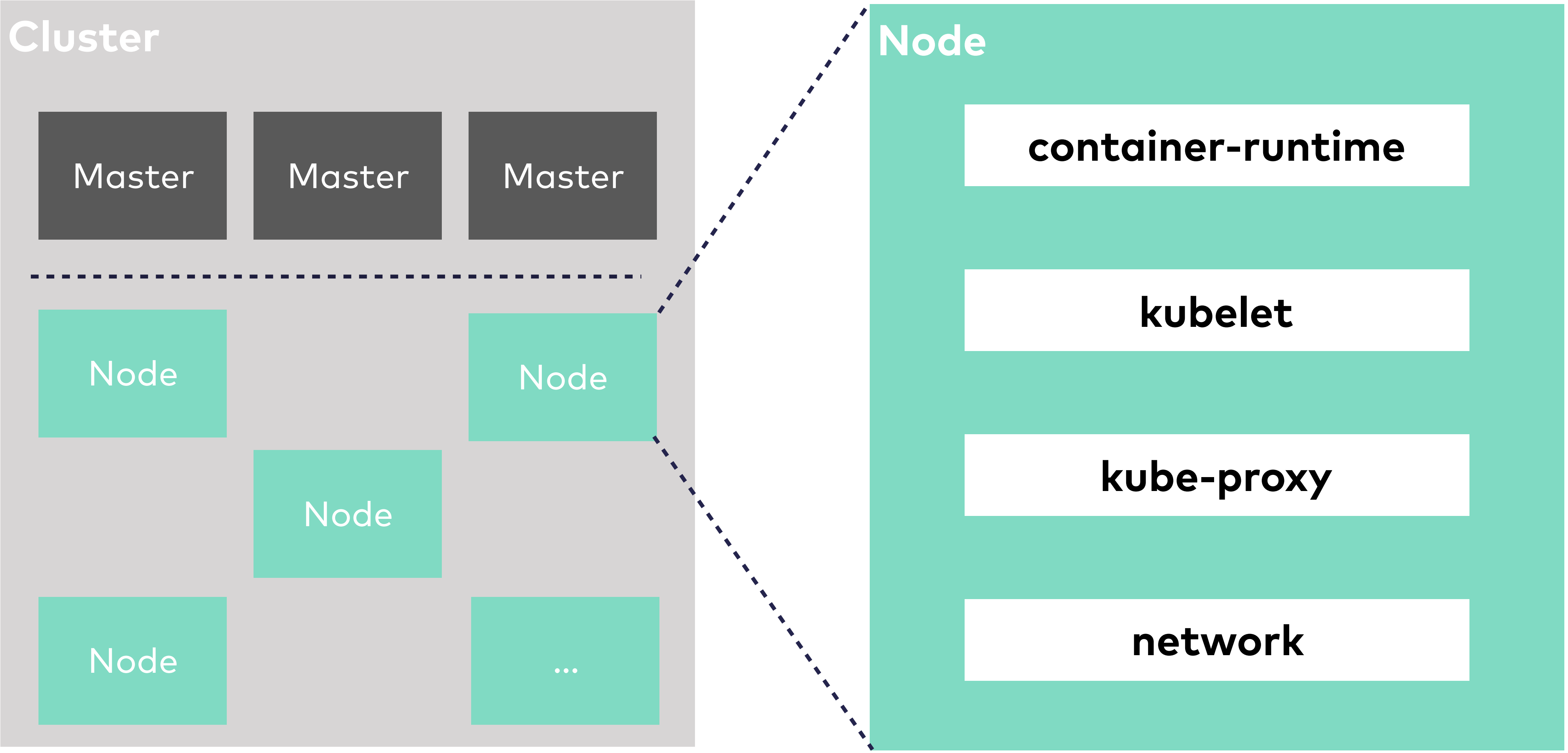
Cluster overview



Master Components



Node Components



Cluster

Master

Master

Master

Node

Node

Node

Node

...

Node

container-runtime

kubelet

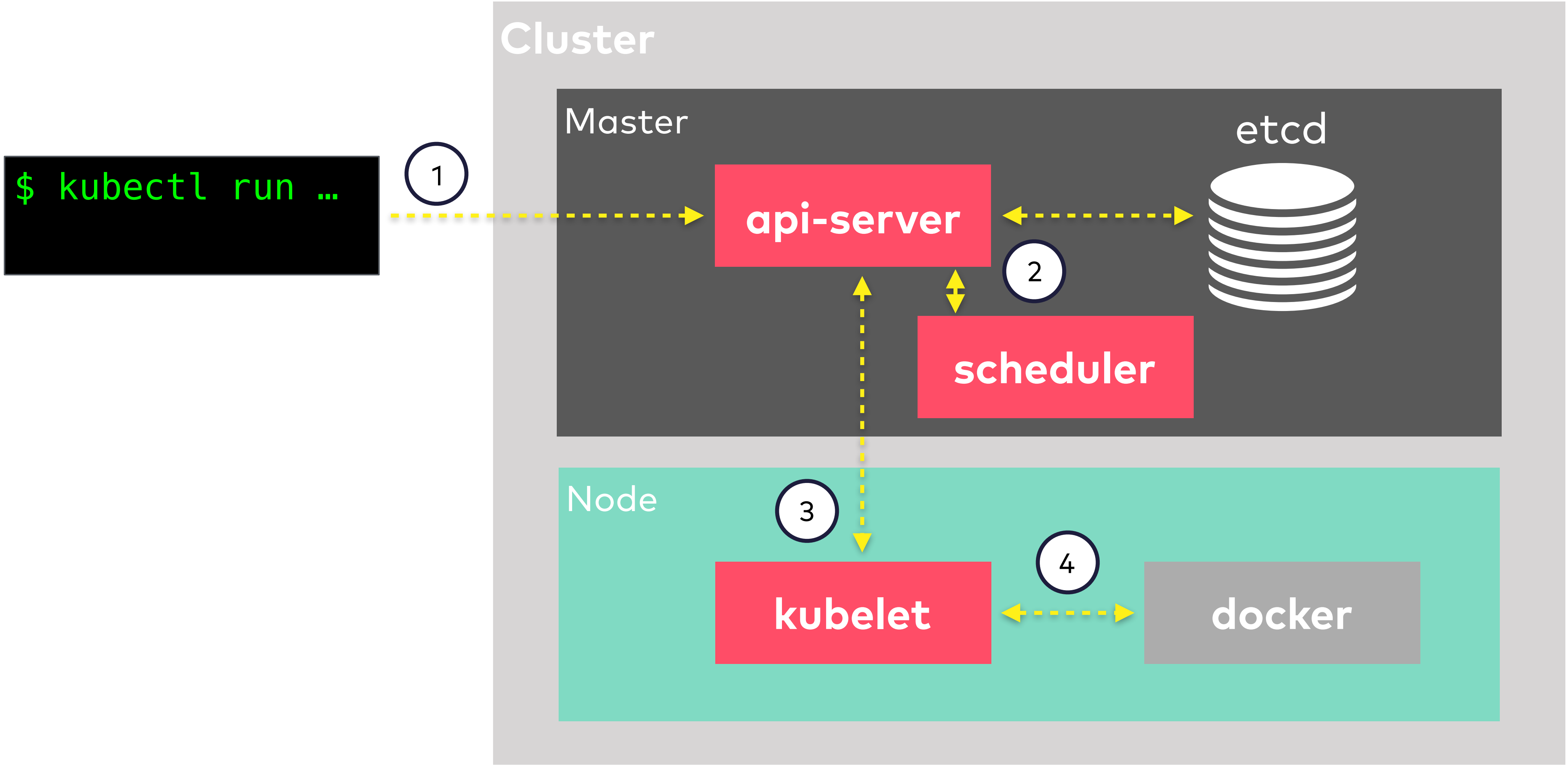
kube-proxy

network

API Objects

- Persistent (in etcd)
- represent the desired state of the cluster
- Have
  - Spec
  - Status

API Objects interaction



```
$ kubectl run …
```

#Computing #Software #Interface #BigData
#ComputerGeneratedDesign
085
Matthew Plummer-Fernandez
Vertigo in the Face of the Infinite

Core abstractions

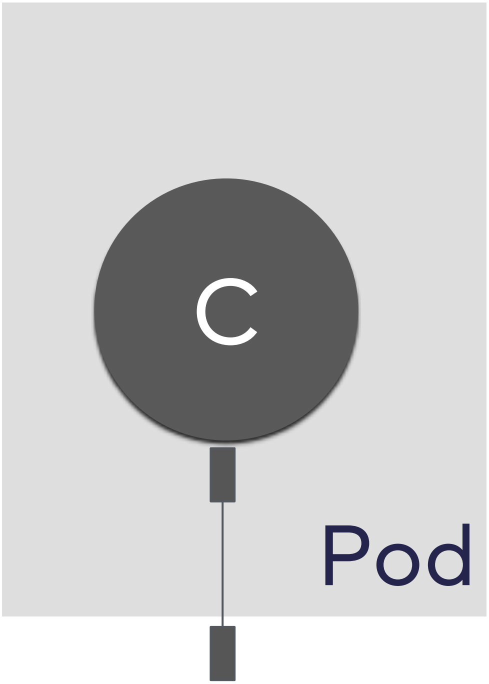
github.com/JoergM/
kubernetes_workshop_demos

Pods

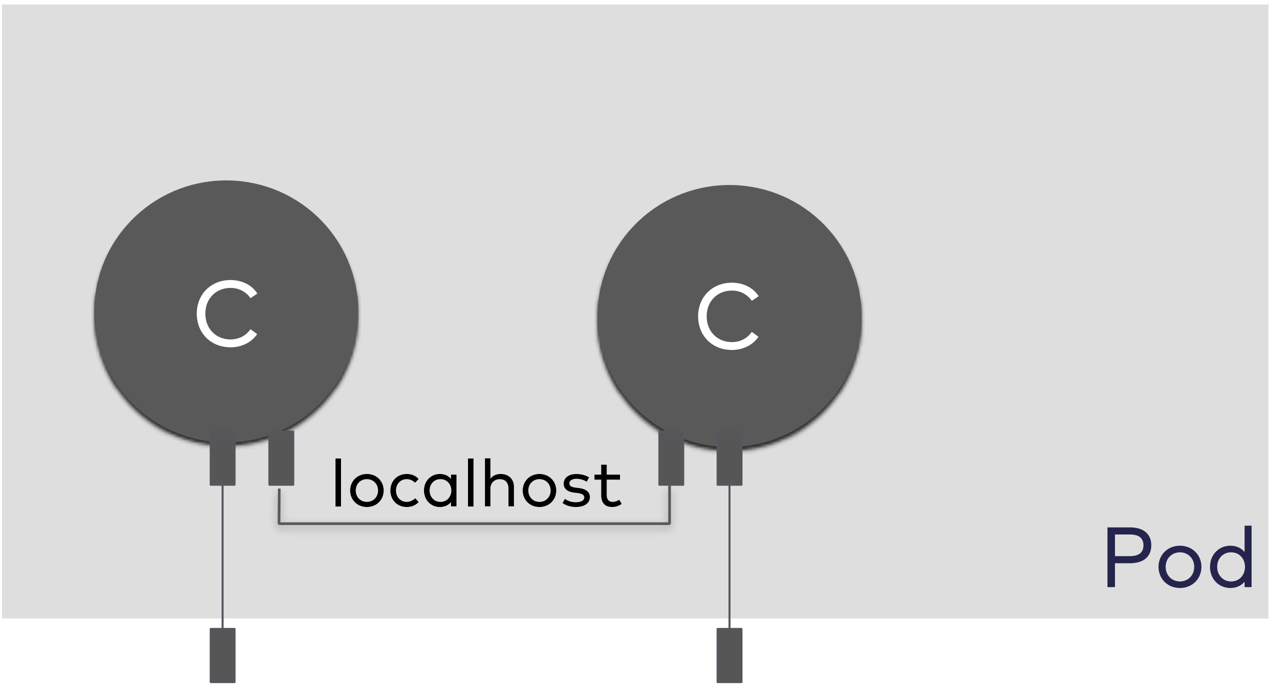
Pod

- Deployment-Unit in Kubernetes
- A pod consists of one or more containers
- Containers in a pod share network
- Containers in a pod can share volumes
- Each pod receives its own cluster-wide and cluster internal IP address

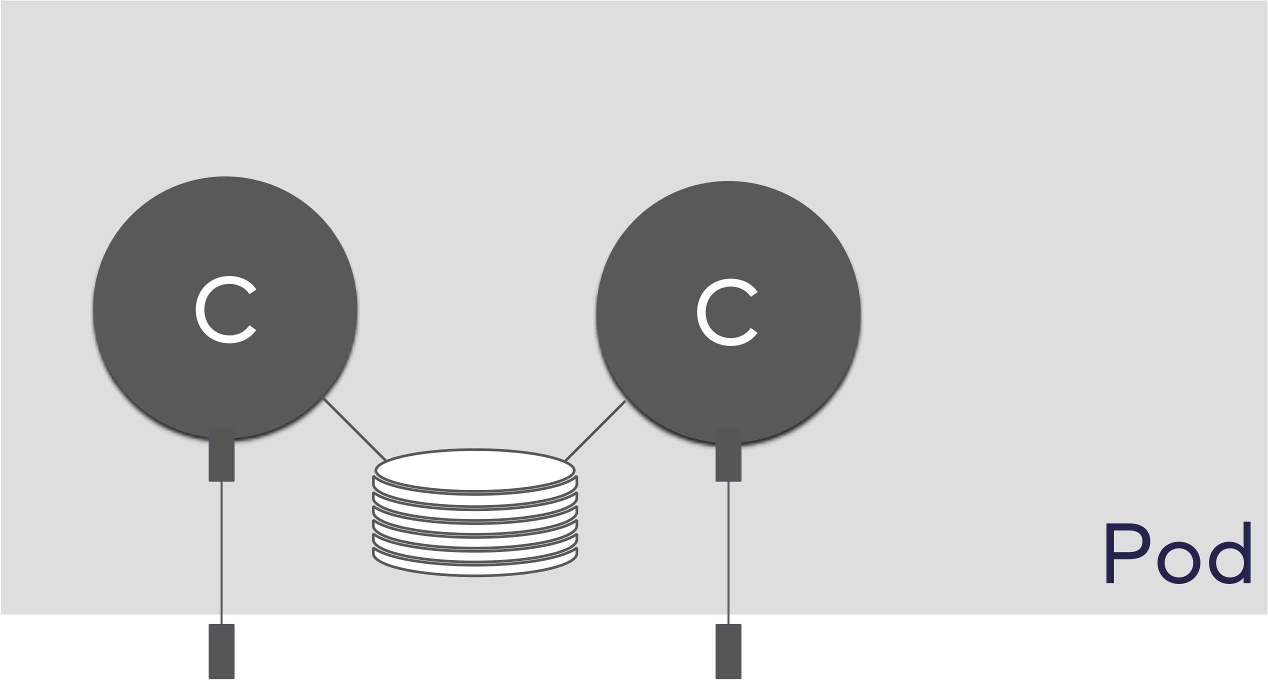
Pod with a single container



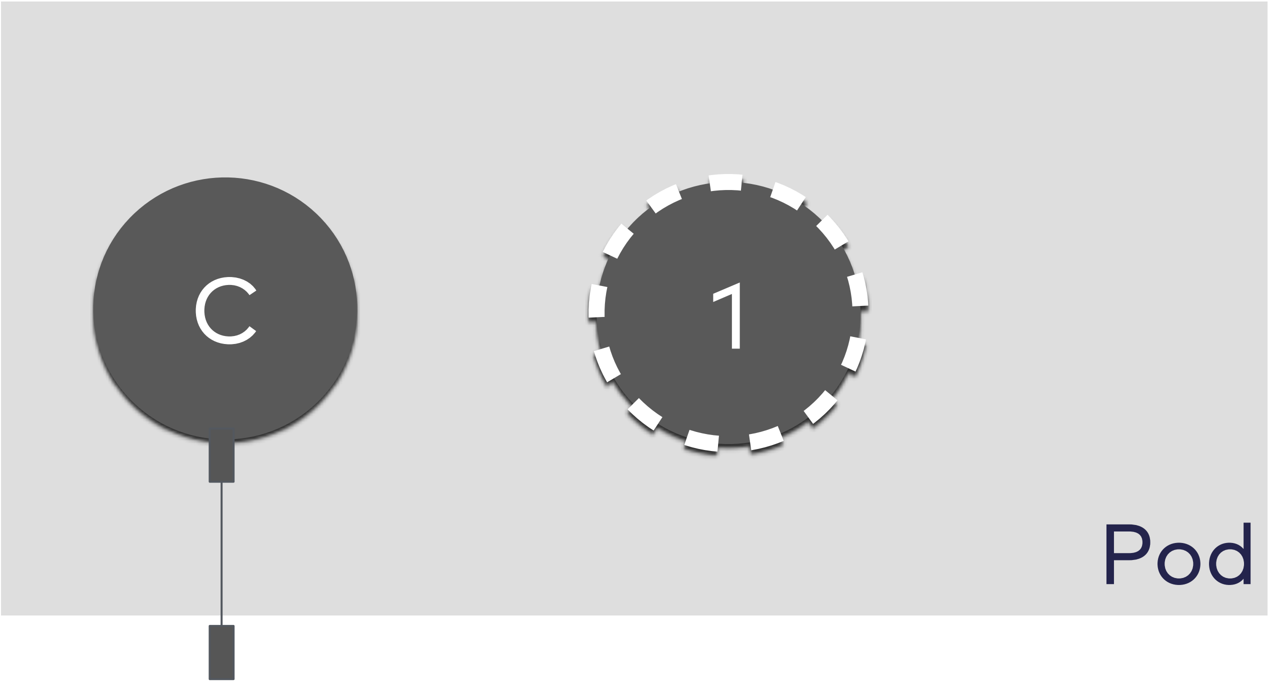
Sharing network



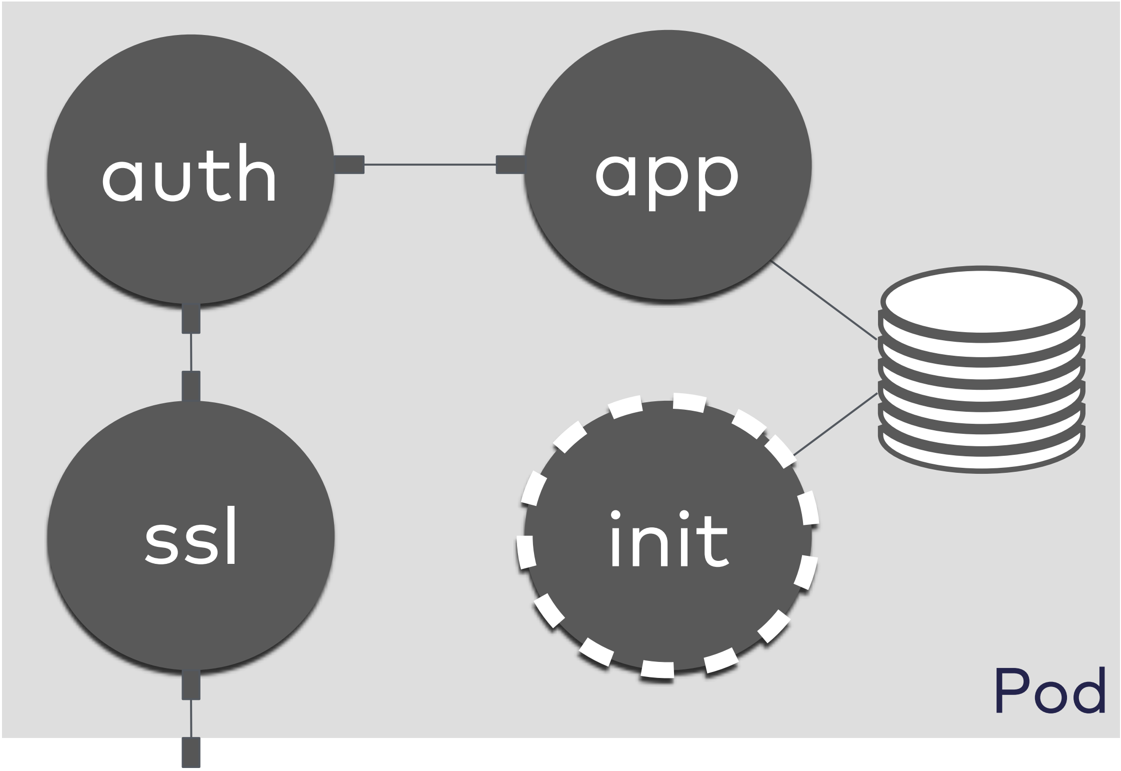
Sharing volumes



Pods with init containers



Complex pod patterns



DEMO

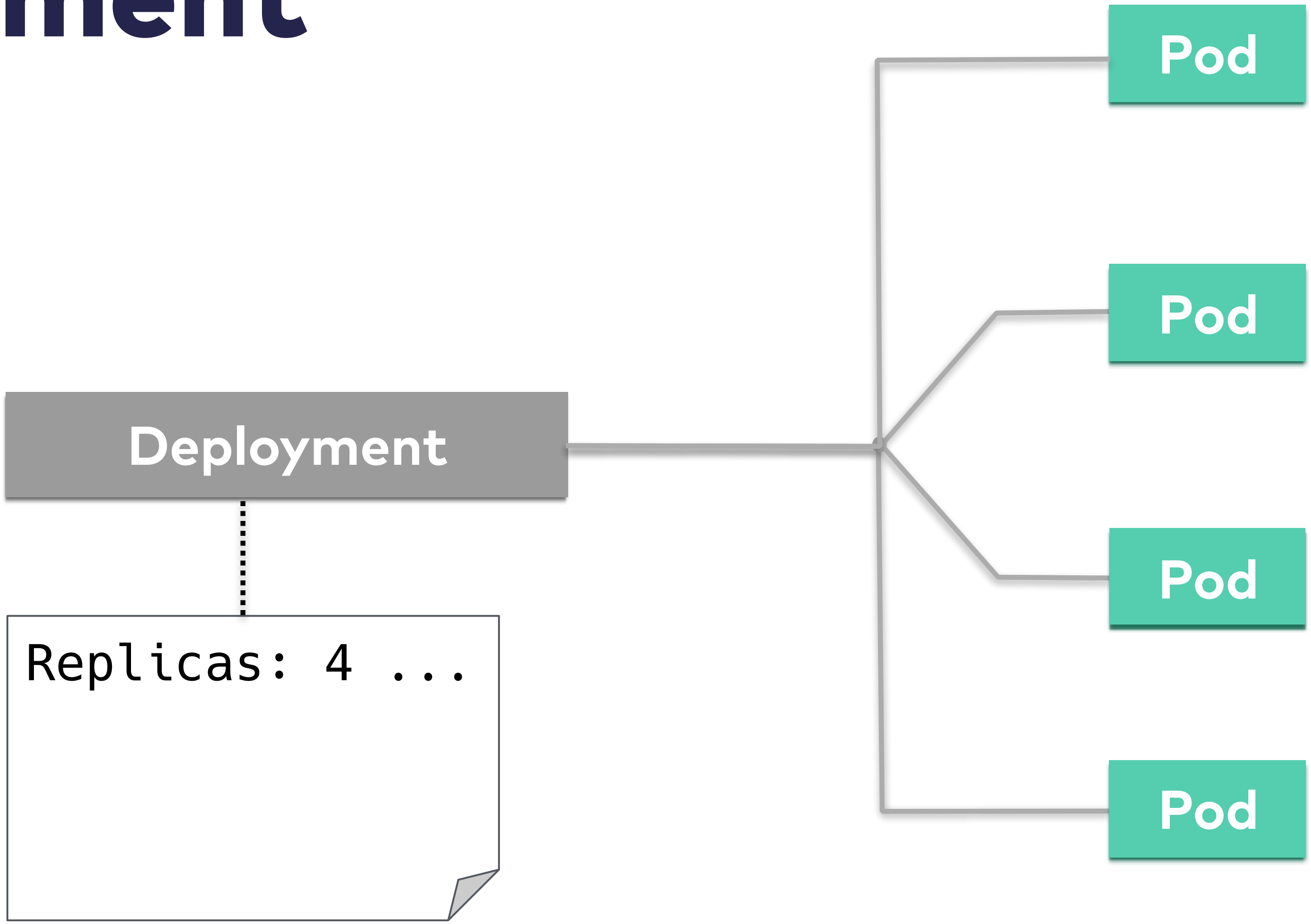
github.com/JoergM/kubernetes_workshop_demos/pods

Deployments

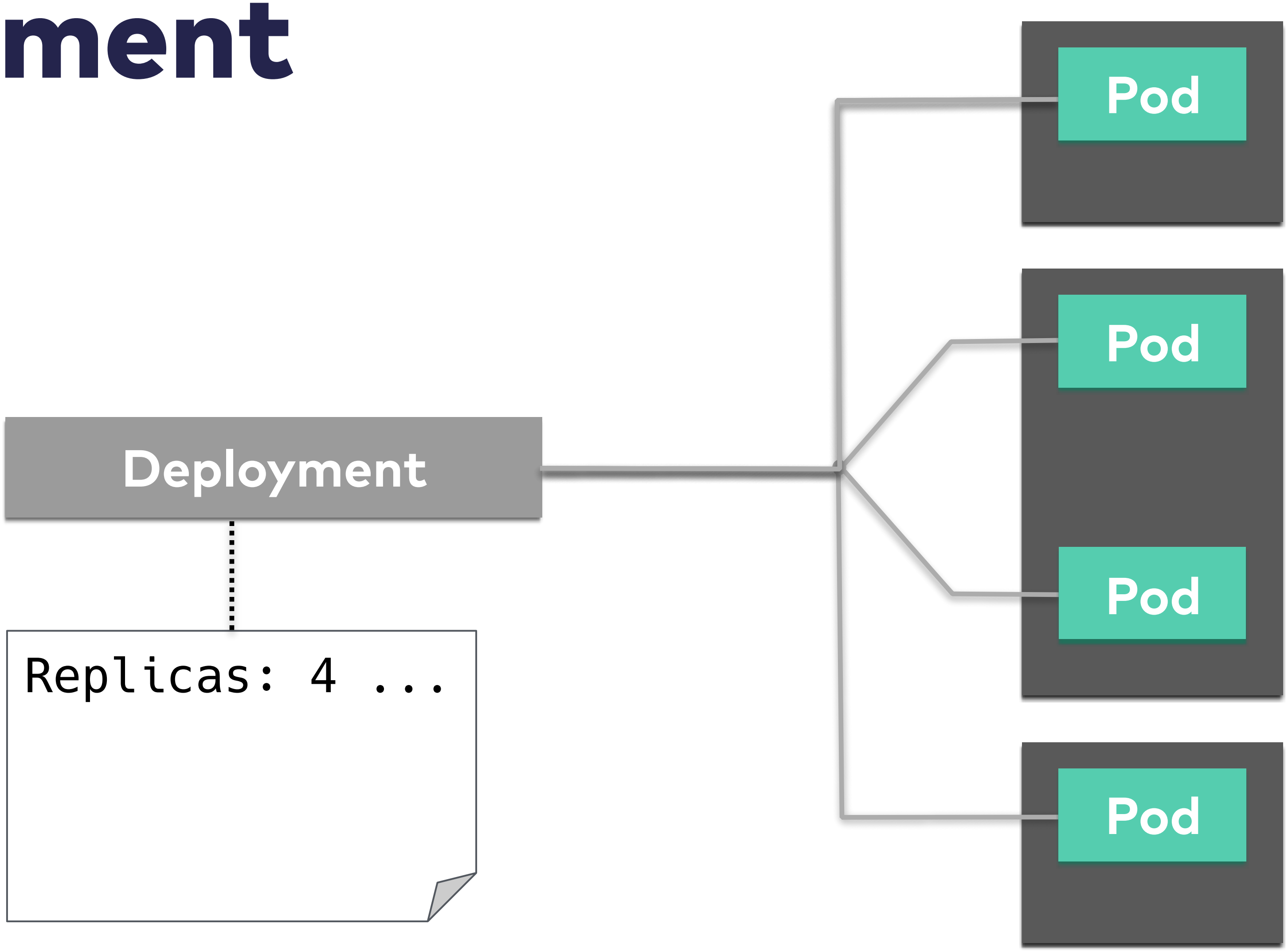
Deployment

- Declares a state of Pods
- Is used for scaling up N instances of the same pod
- Is used to deploy old or new revisions of a pod

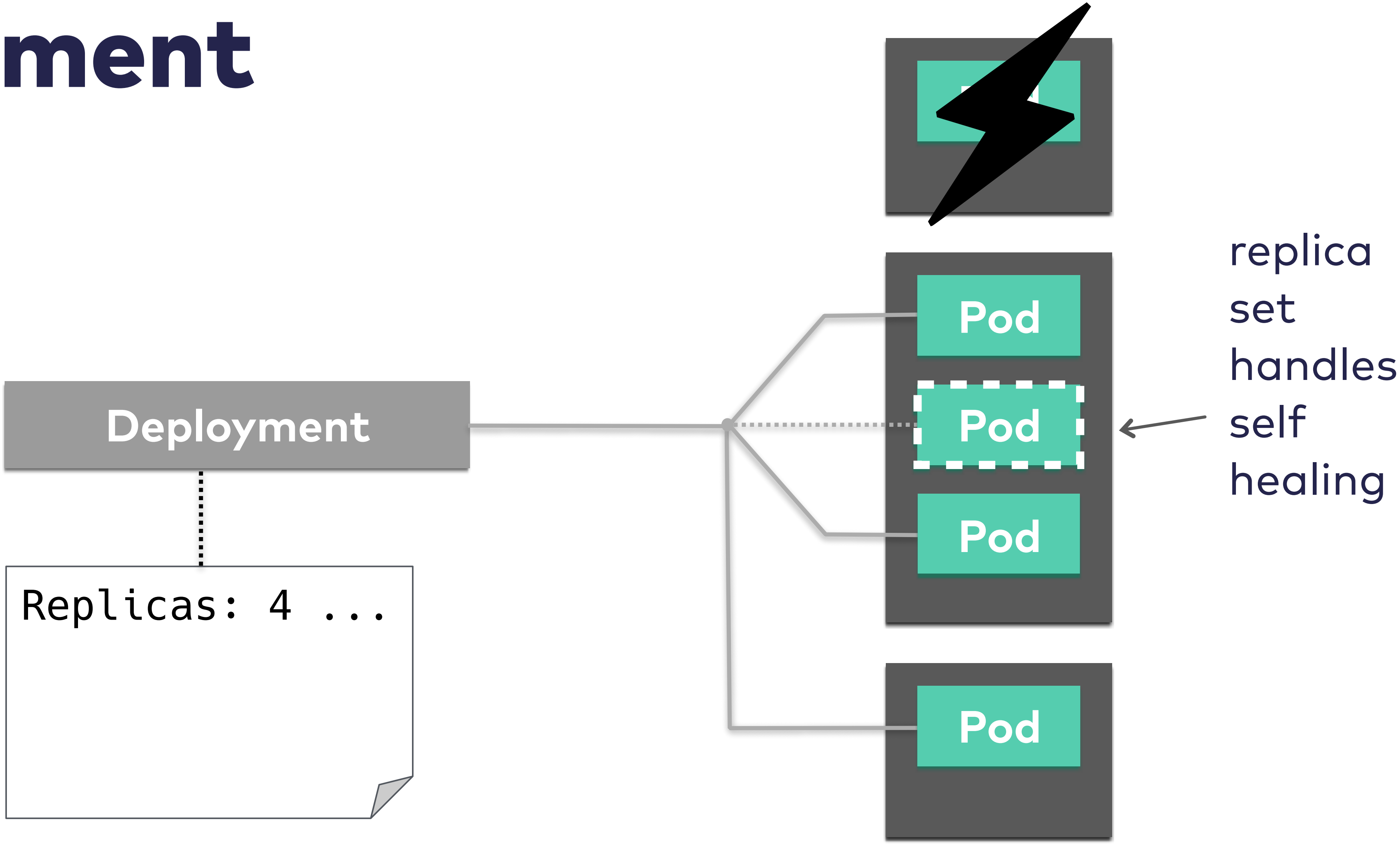
Deployment



Deployment



Deployment



Deployment

```
Replicas: 4 ...
```

Pod

Pod

Pod

Pod

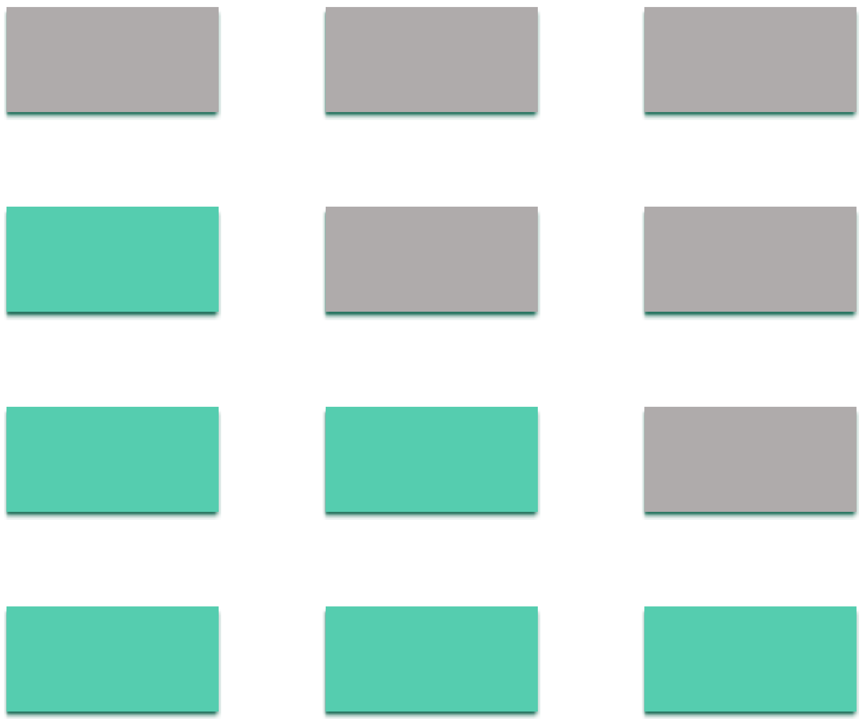
replica set handles self healing

Deploying new versions

- Rolling update
- Recreate
- Blue Green
- Canary

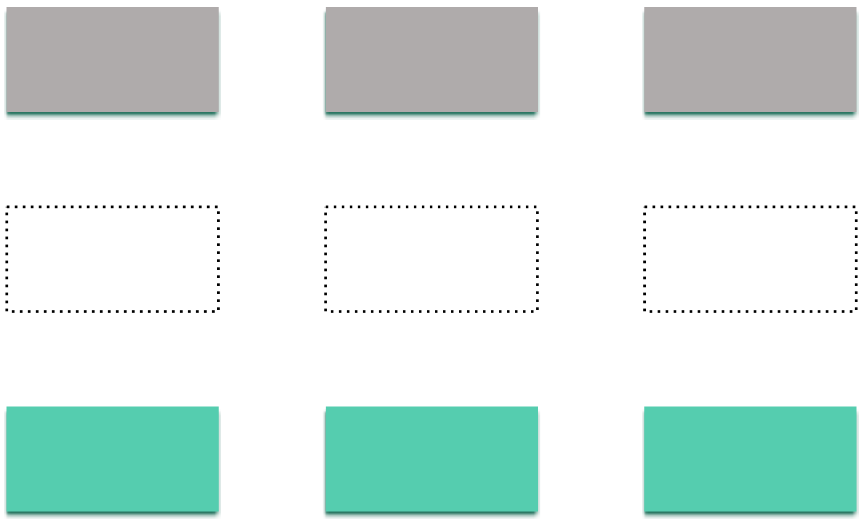
Rolling Update

- Default variant
- No service downtime
- Both versions get traffic at the same time
- consider setting maxUnavailable/ maxSurge



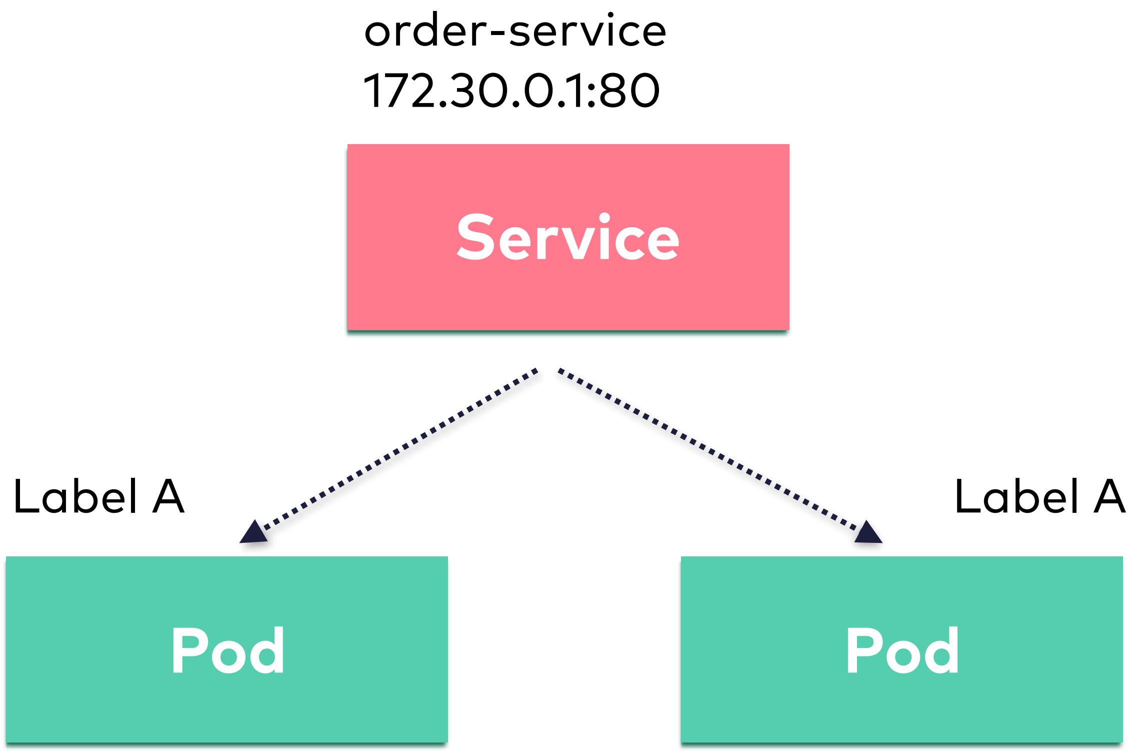
Recreate

- Activated setting type
- Involves downtime
- no version conflicts



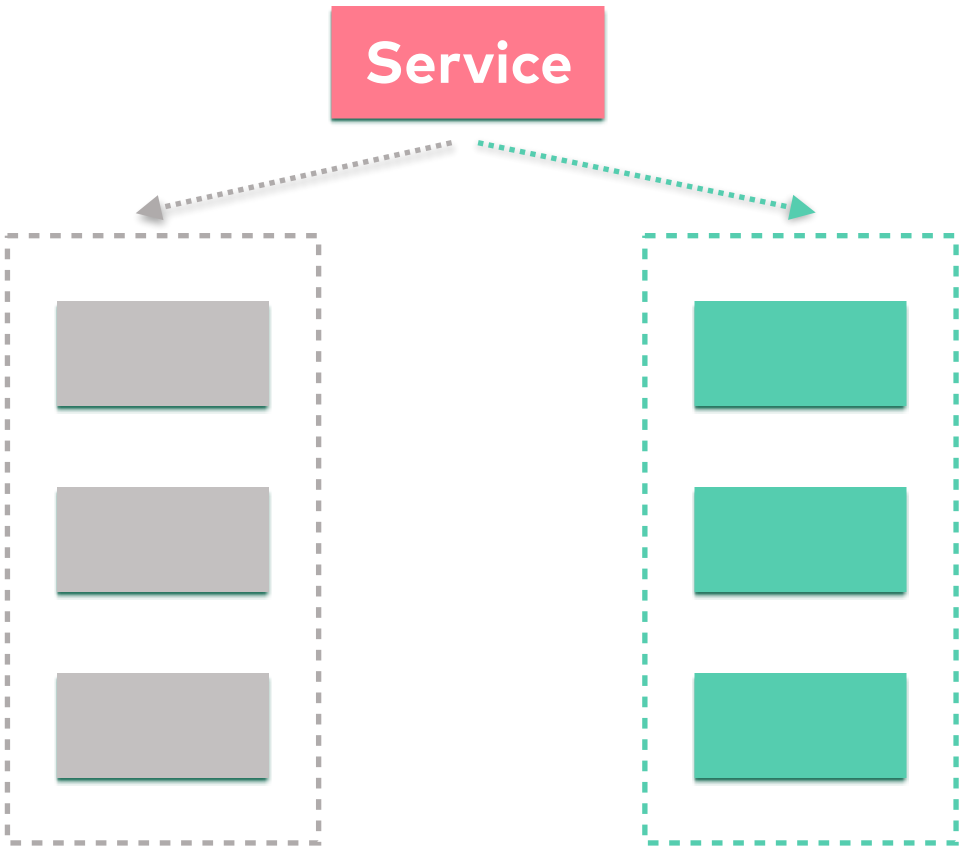
Prelude Service

- IP and DNS for multiple Pods
- Loadbalancing
- Uses Labels to find Pods



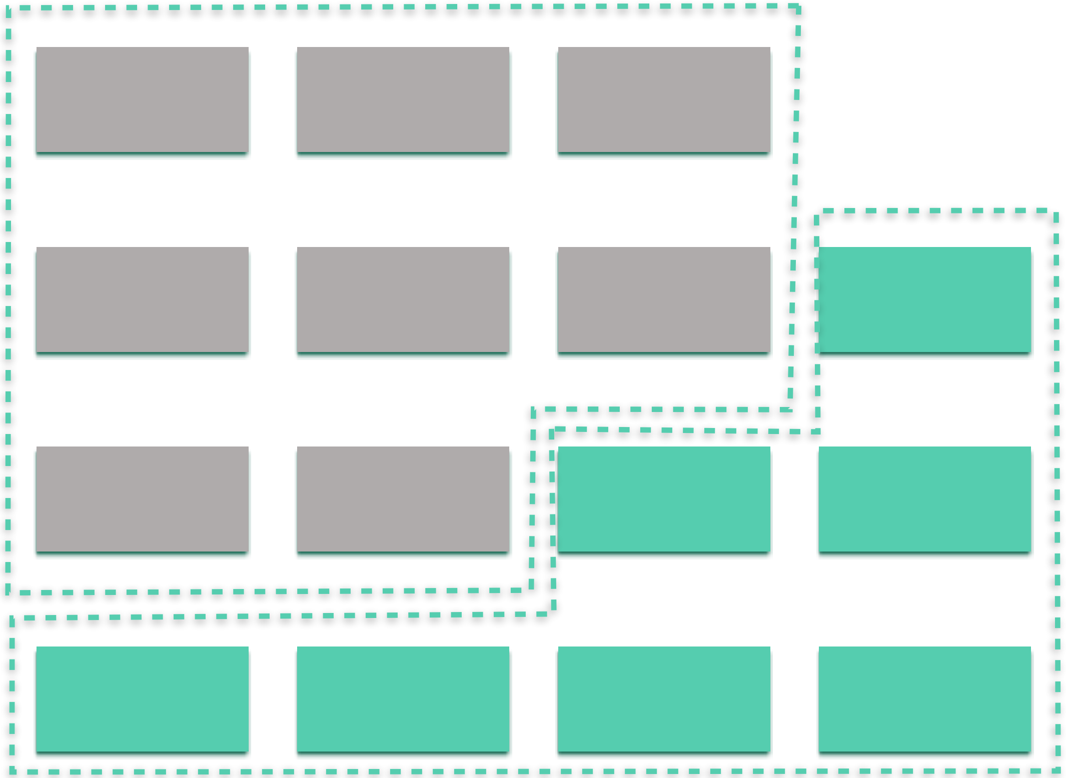
Blue / Green

- No version conflicts
- No downtime
- High resource usage
- Involves custom handling by Switching service labels



Canary

- Slowly testing new versions
- No Downtime
- Both versions get traffic at the same time
- Some custom handling of multiple deployments

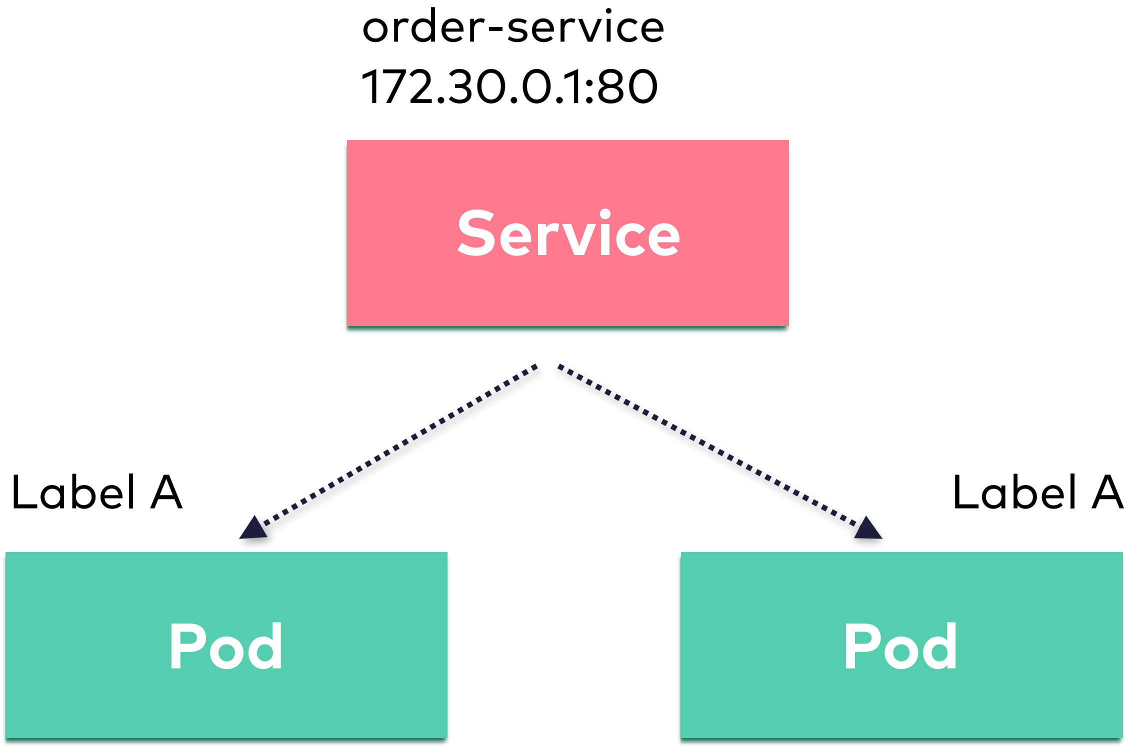


DEMO

github.com/JoergM/kubernetes_workshop_demos/deployments

Services

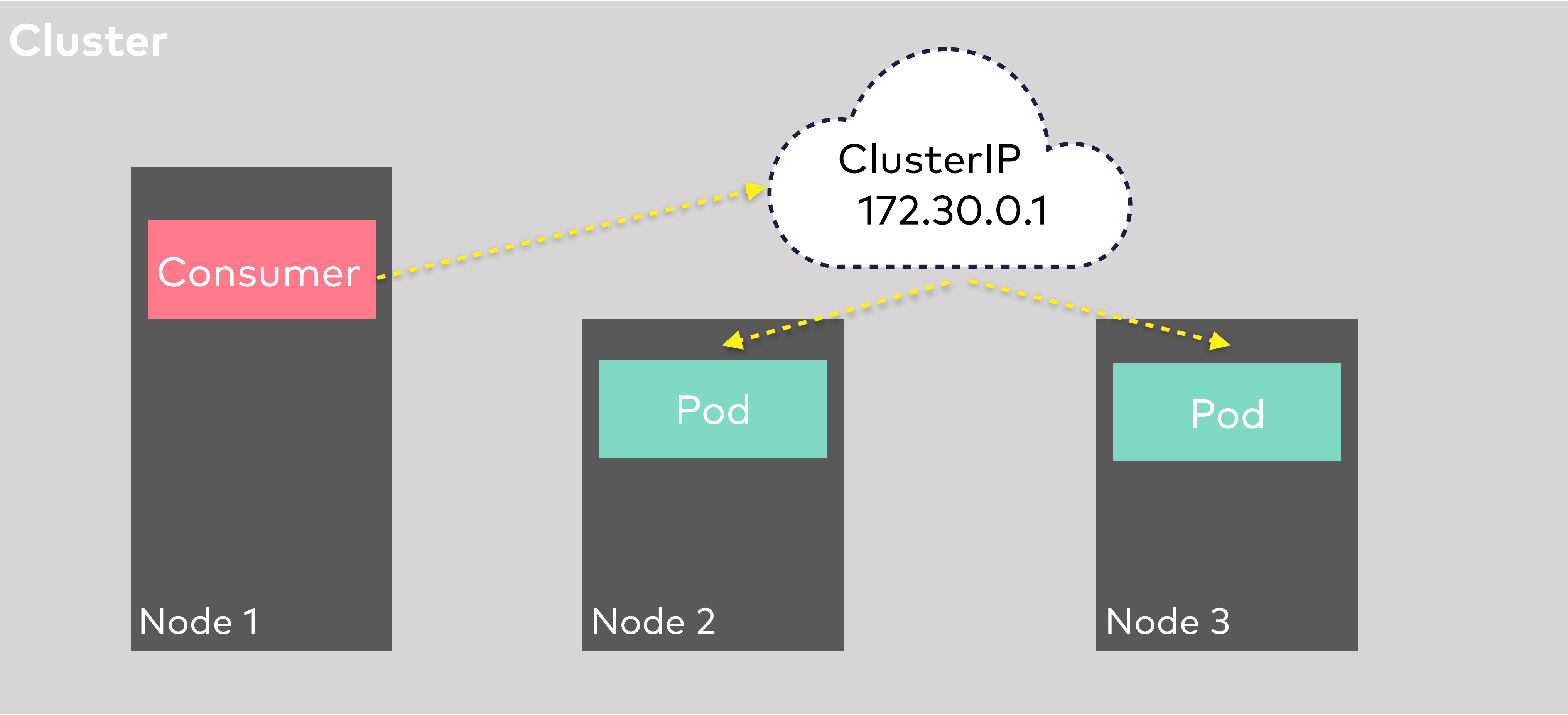
Service Overview



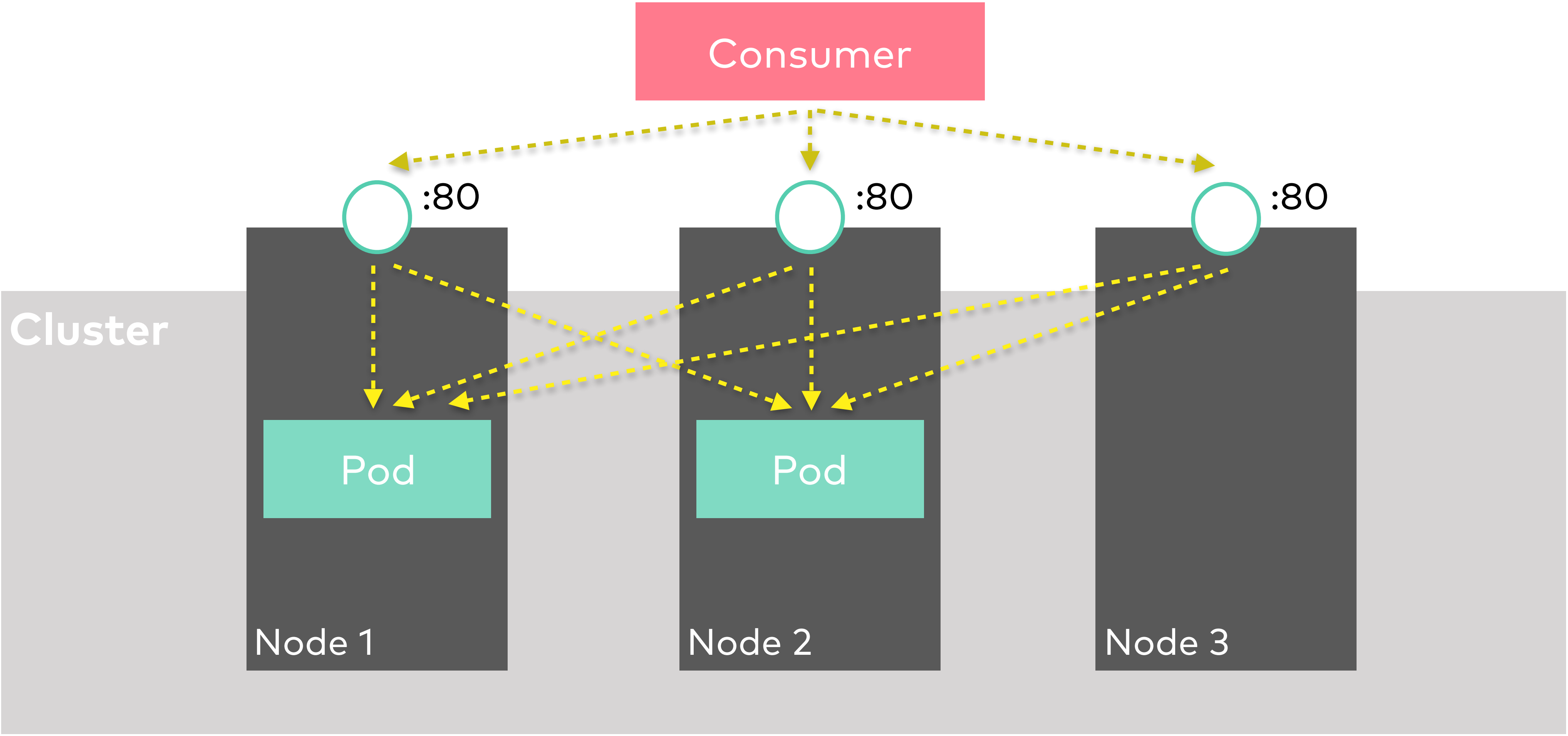
Service

- Is an abstraction which defines a logical set of pods and a policy by which to access them
- Usually represents a micro-service
- Different types of services possible
- Discovery inside Cluster via DNS
- It's not a physical LoadBalancer (more later)

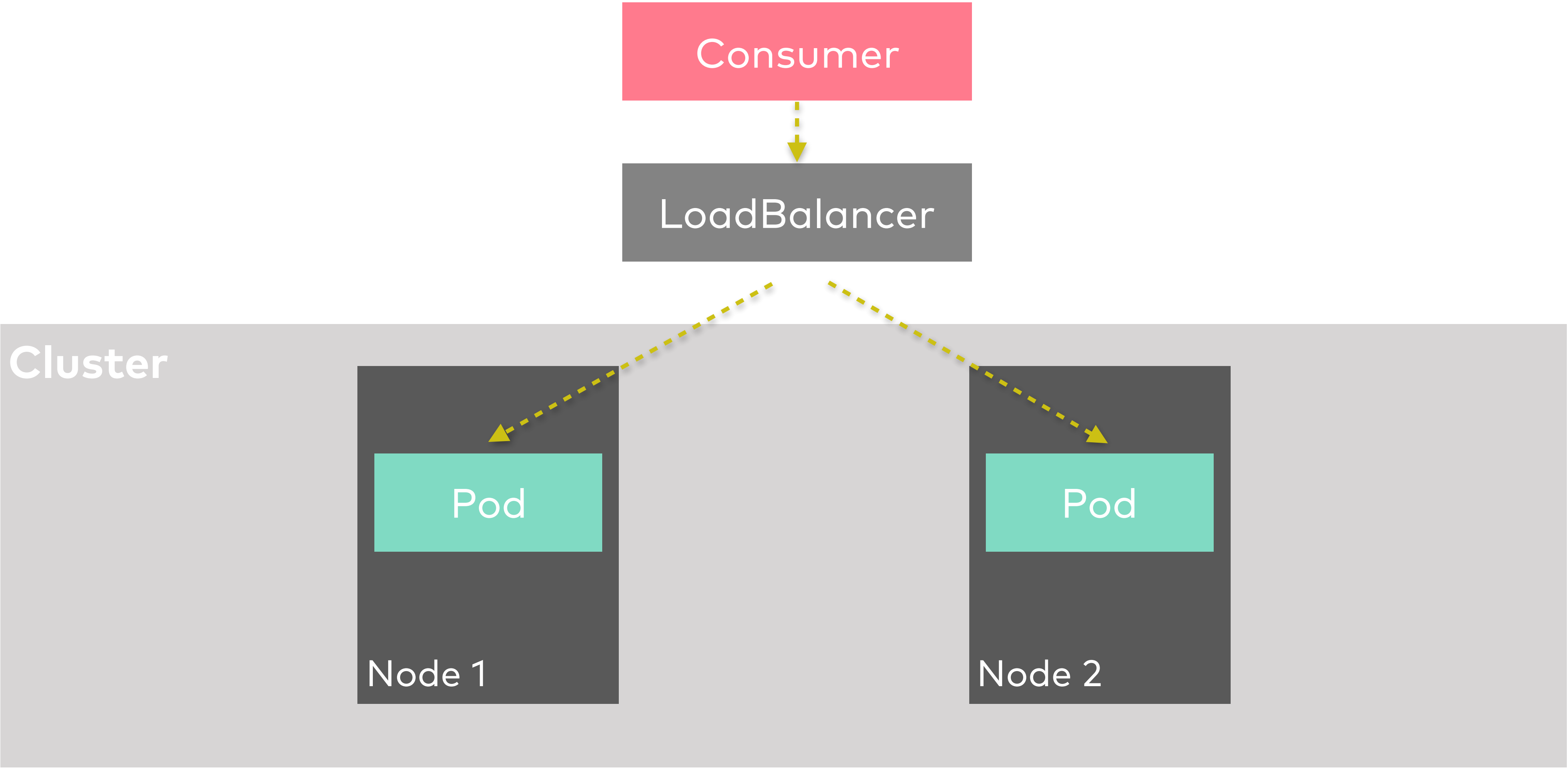
Service type ClusterIP



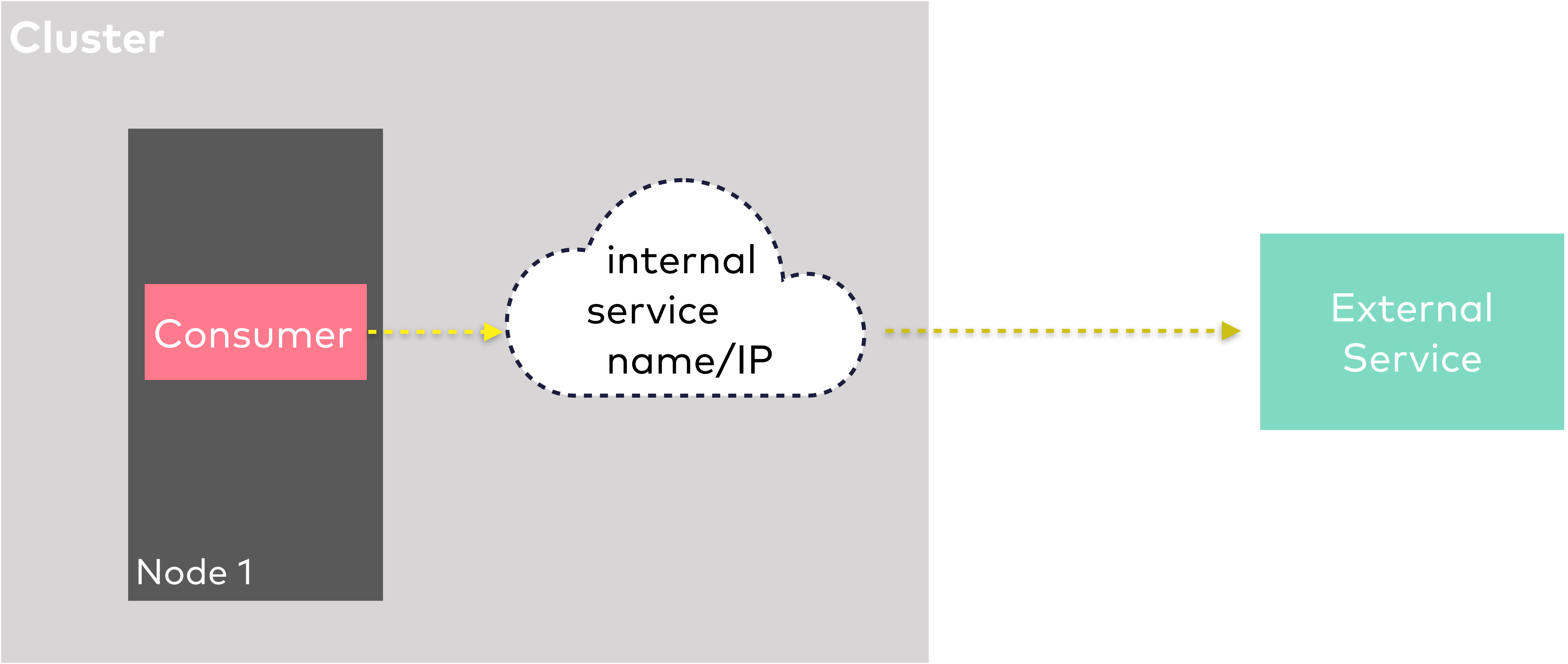
Service type NodePort



Service type LoadBalancer



Service type External...



DEMO

github.com/JoergM/kubernetes_workshop_demos/services

Configuration

Config Maps

- Provide Pods with configuration data
- from
  - literal values
  - files
  - directories

Config Maps

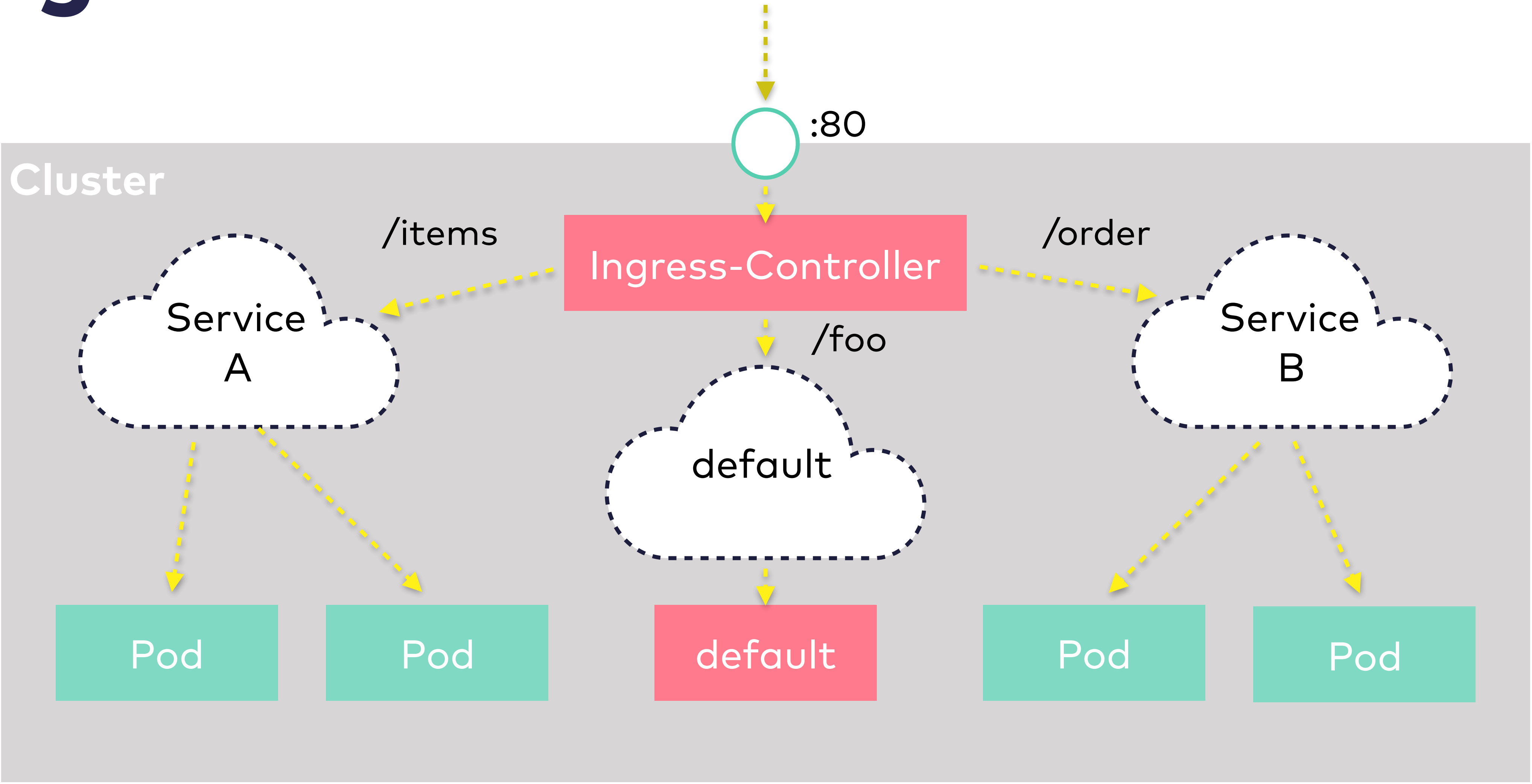
- In Pods as
  - environment variables
  - files
  - directories

DEMO

github.com/JoergM/kubernetes_workshop_demos/configuration

Ingress

Ingress


:80
Cluster
/items
Ingress-Controller
/order
Service
A
/foo
Service
B
default
Pod
Pod
default
Pod
Pod

Ingress Controller

- Creates a LoadBalancer service that points to a pod, which runs a reverse proxy (nginx, haproxy, Apache, traefik)
- Uses IngressRules to describe which DNS and/or path should point to which service
- Always needs a default service

Ingress controller implementations

- nginx
- traefik
- voyager
- GCE ingress
- Kong
- ...

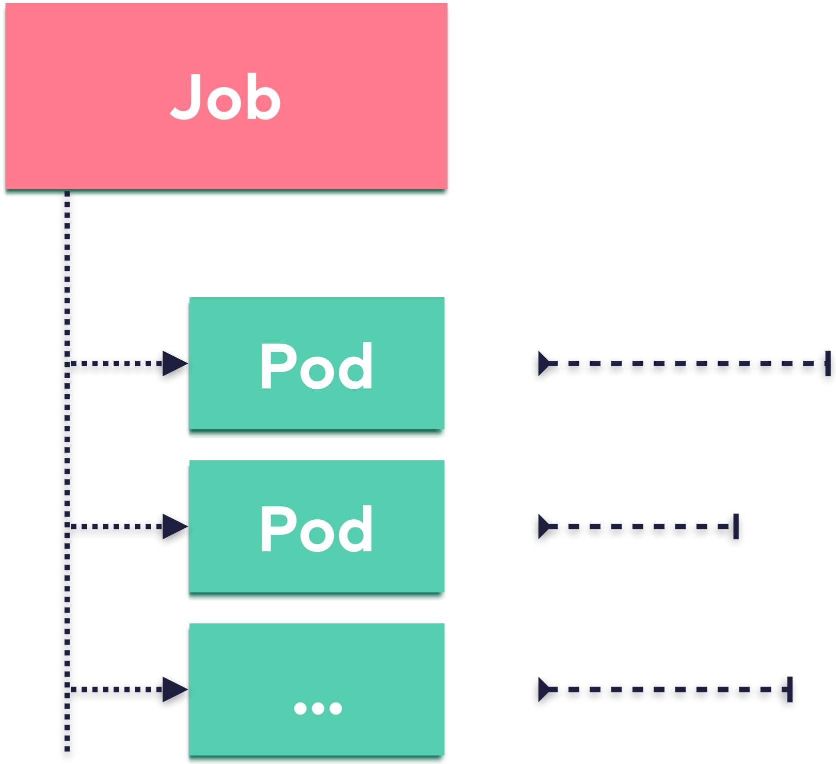
DEMO

github.com/JoergM/kubernetes_workshop_demos/ingress

Jobs

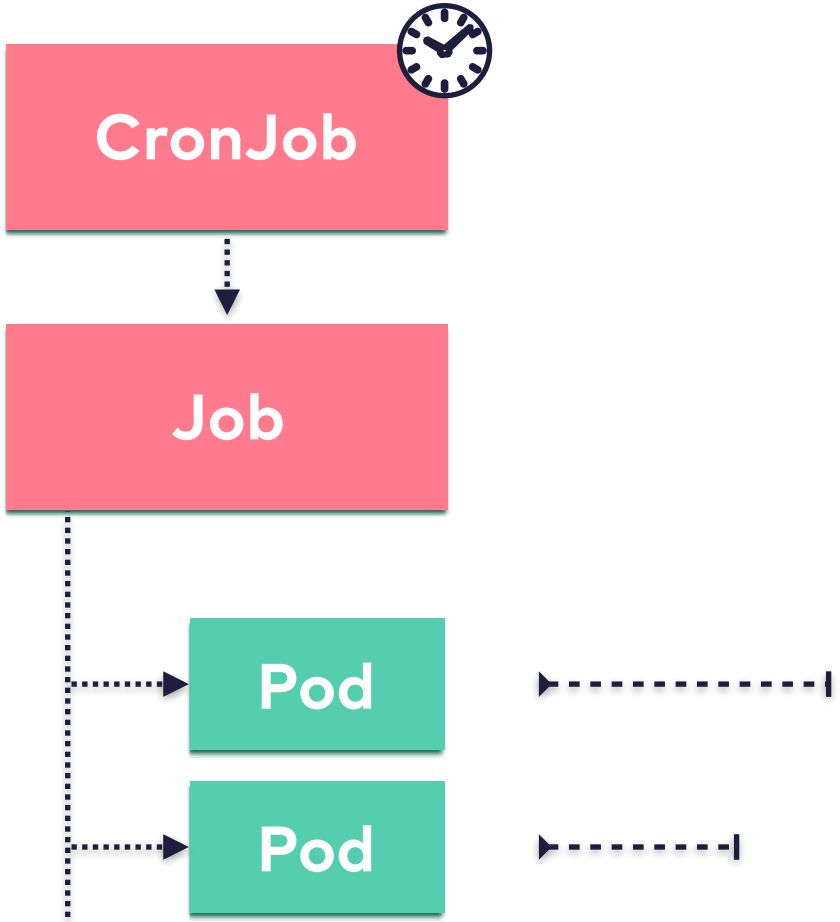
Jobs

- Running pods until completion
- Like deployment for long-running pods
- supports parallelism



CronJob

- Regularly starting jobs
- Follows typical Cron patterns:
    - 0 12 * * 1-5
    - (weekdays at noon)

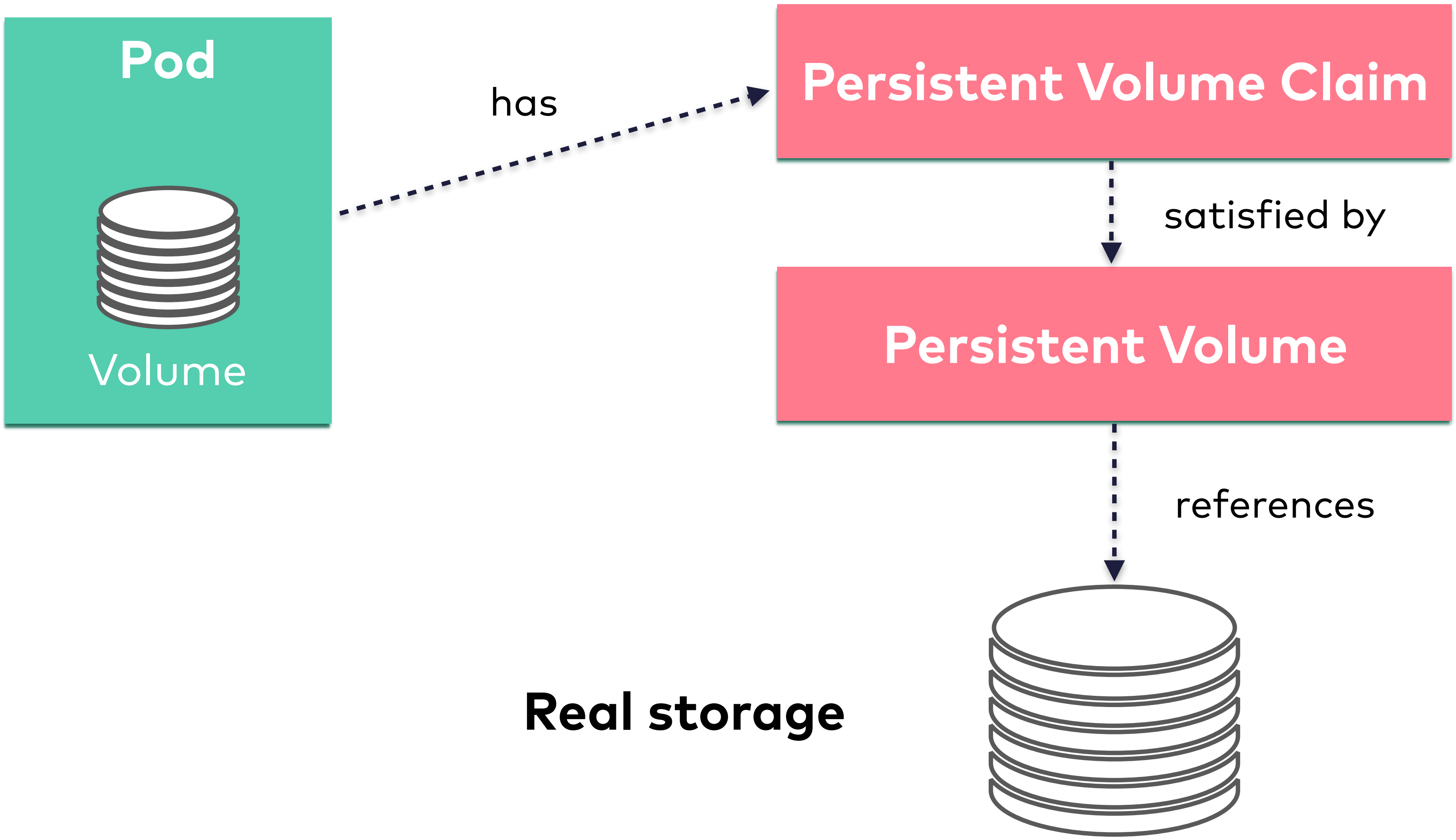


DEMO

github.com/JoergM/kubernetes_workshop_demos/jobs

Persistence

Persistence Overview



Persistent Volume Claims

- User requesting Storage
- Used in pod as volume
- Survives pod recreation
- Certain Size (e.g. 5Gi)
- Certain class (e.g. SSD)
- Will be matched to persistent volumes

Persistent Volumes

- Defines a real volume of a certain size (e.g. 5Gi)
- Can be created upfront
- Lots of implementations:
    - GCEPersistenceDisk
    - HostPath
    - AWS EBS
    - NFS

Dynamic provisioning

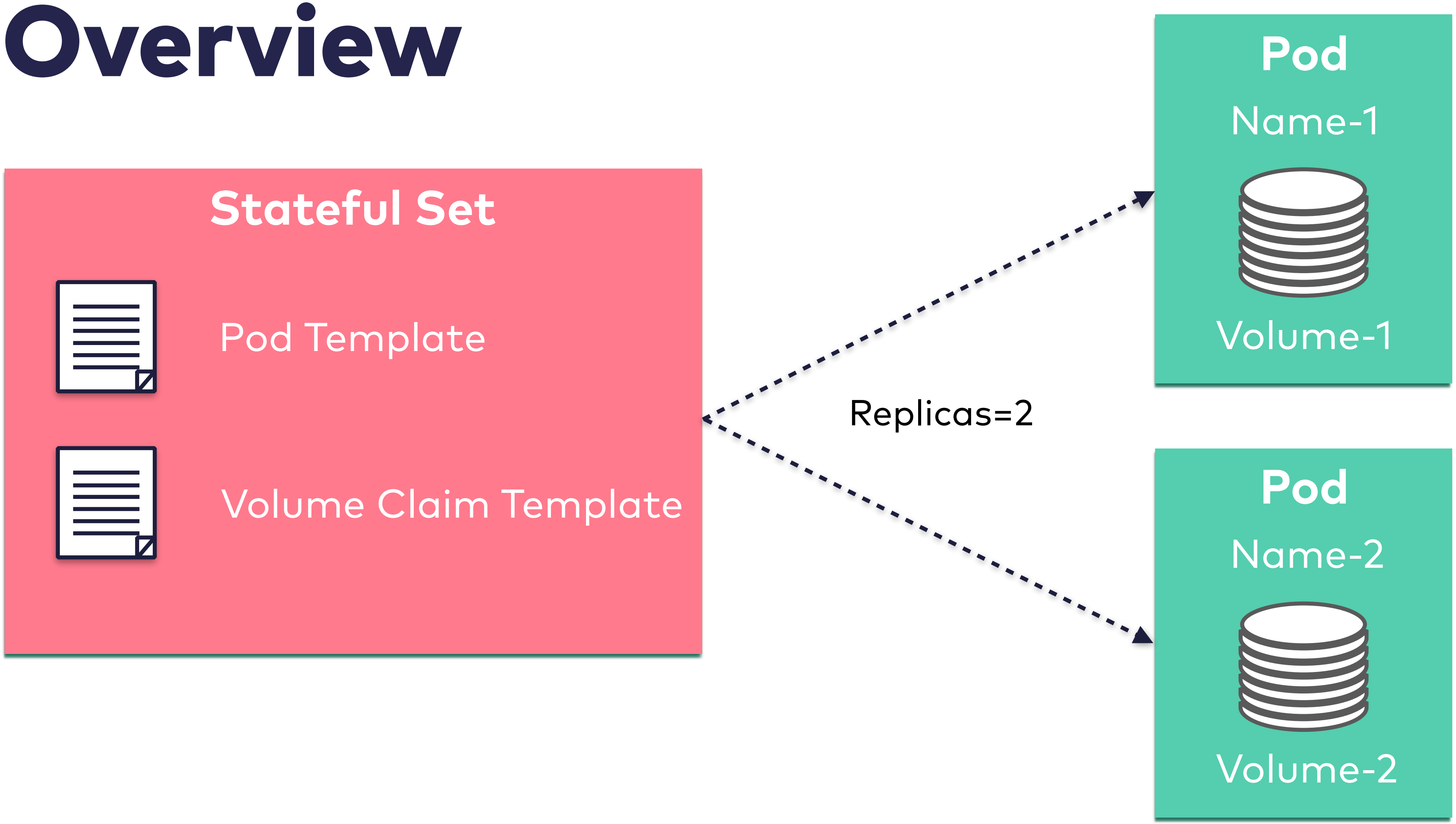
- Creating Volumes based on Claims
- Requires dynamic way of creating volumes and mounting to nodes
- e.g. AWS EBS, GCE Persistent Disk
- Custom provisioners for can be created too

DEMO

github.com/JoergM/kubernetes_workshop_demos/persistent_volumes

Stateful Sets

Overview


Stateful Set
Pod Template
Volume Claim Template
Replicas=2
Pod
Name-1
Volume-1
Pod
Name-2
Volume-2

Stateful Sets

- Like Deployment but with other guarantees
- Each Replica has always the same name (and DNS)
- Replica and Volume Claim always come together
- Most parameters not changeable after creation

Stateful Sets - usages

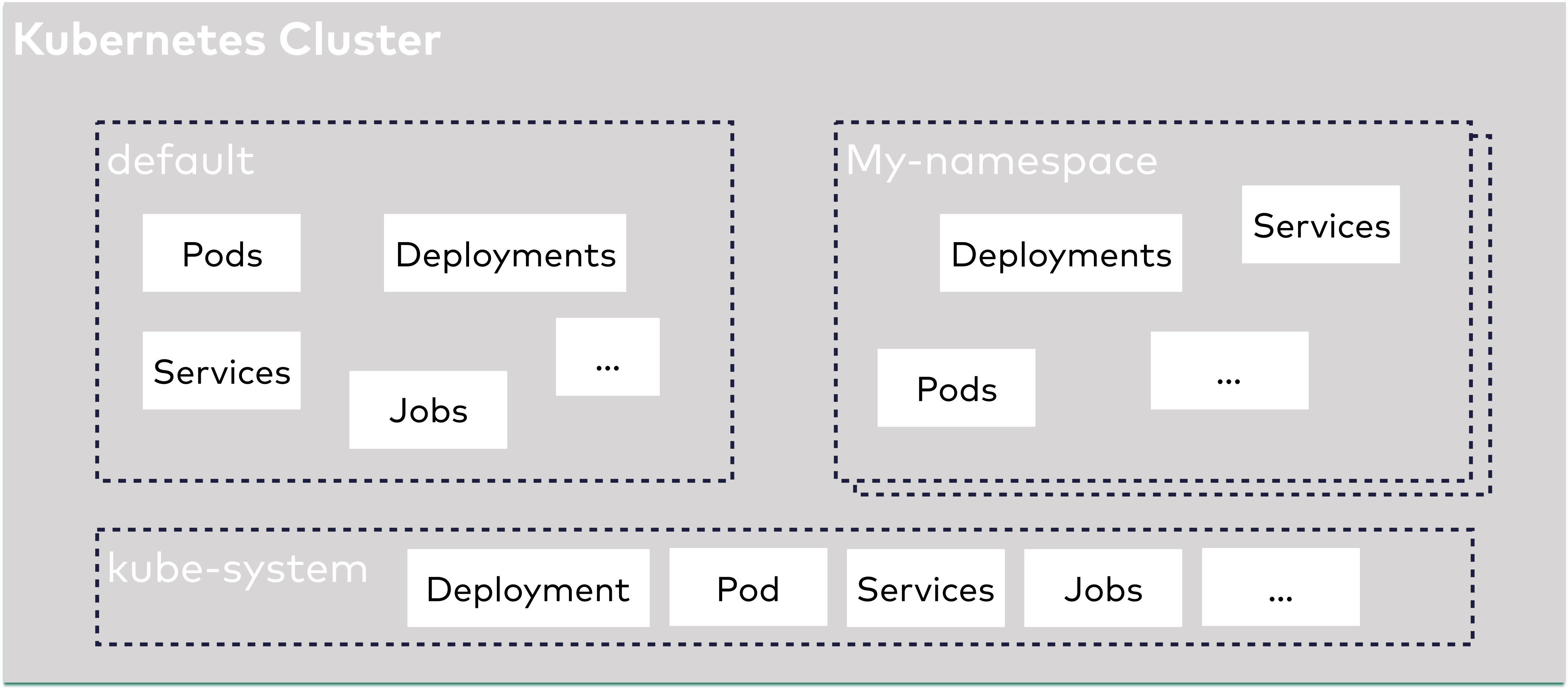
- All kind of software that builds a cluster, but needs certain guarantees
- e.g. Databases with fixed Follower-Leader specification
  - MongoDB
  - Zookeeper
  - Postgresql
- Do not use if not necessary!

DEMO

github.com/JoergM/kubernetes_workshop_demos/stateful_sets

Namespaces

Namespaces overview



Namespaces

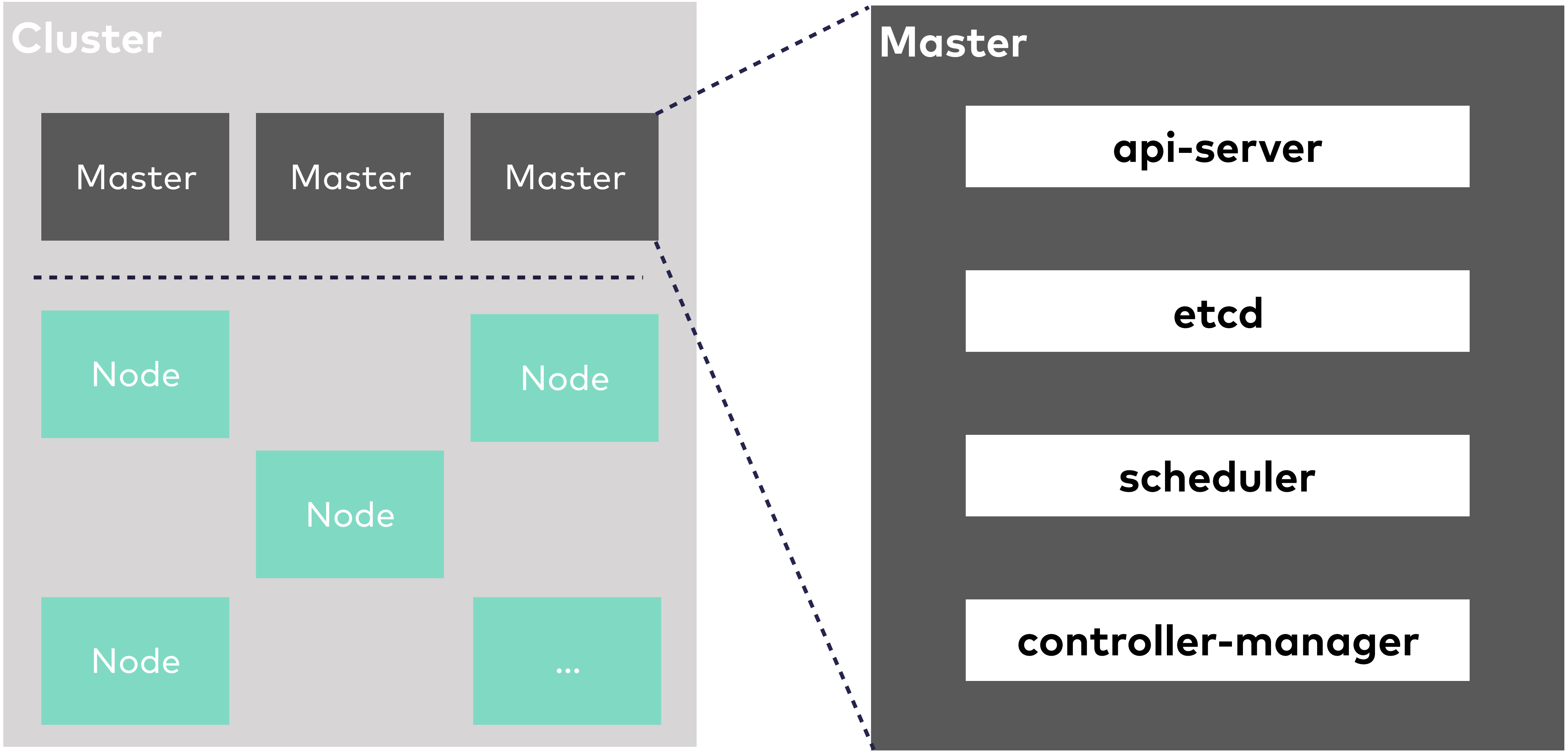
- Scope for names of objects
- Hook for service accounts and network policies
- Isolation level depends on your installation
- Not all objects are in namespaces (esp. low level like nodes or persistent volumes)

DEMO

github.com/JoergM/kubernetes_workshop_demos/namespaces

Internal architecture

Master Components



Cluster

Master

Master

Master

Node

Node

Node

Node

...

Master

api-server

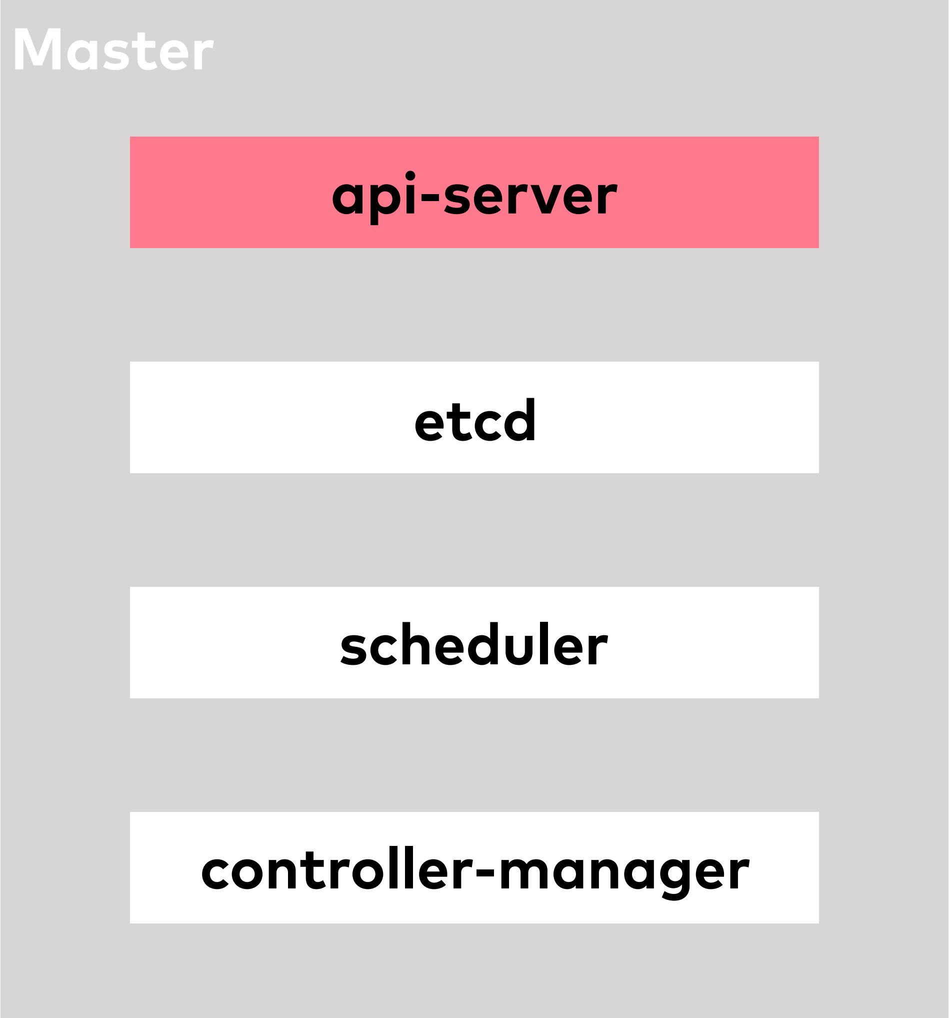
etcd

scheduler

controller-manager

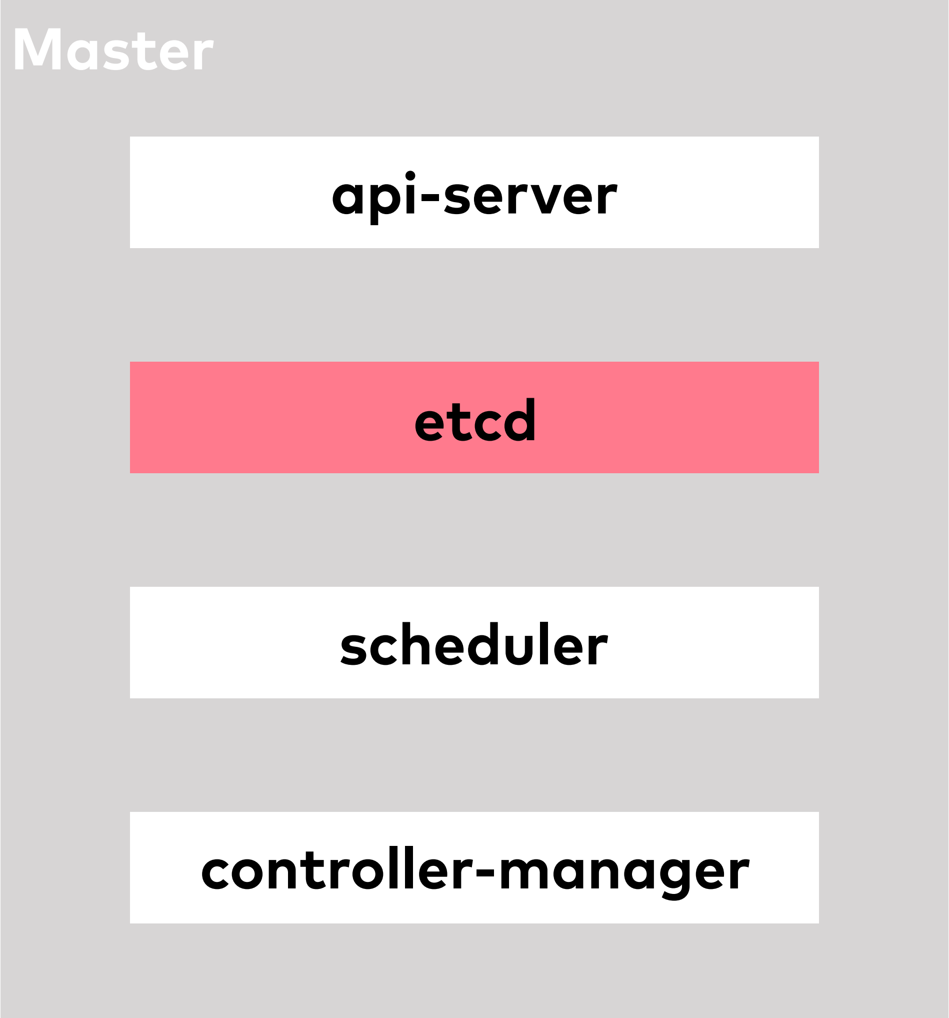
API Server

- Entry point for all interactions
- Stores desired state into etcd
- available from outside and inside cluster



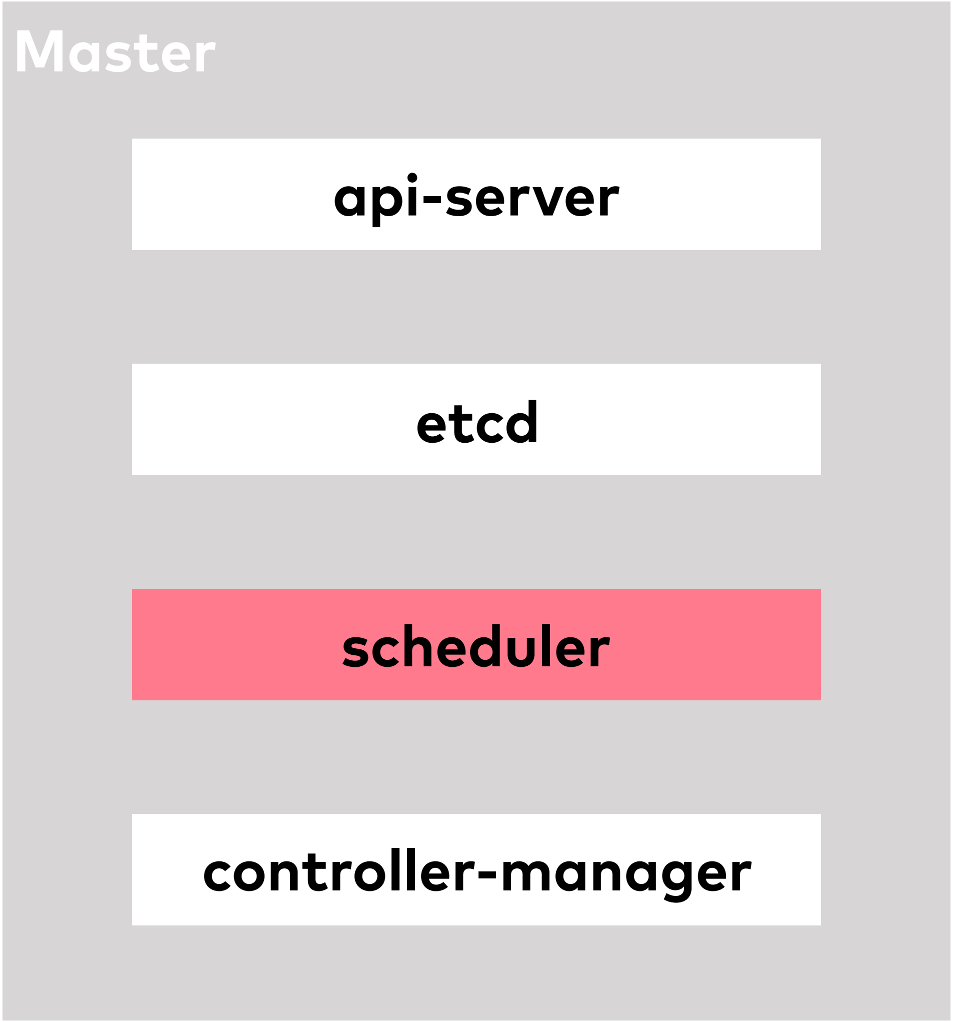
etcd

- consensus based distributed key value database
- interaction only via api-server



Scheduler

- Watches newly created pods and assigns them to nodes
- Lots of criterias
  - resource requirements
  - load
  - specific constraints



Master

api-server

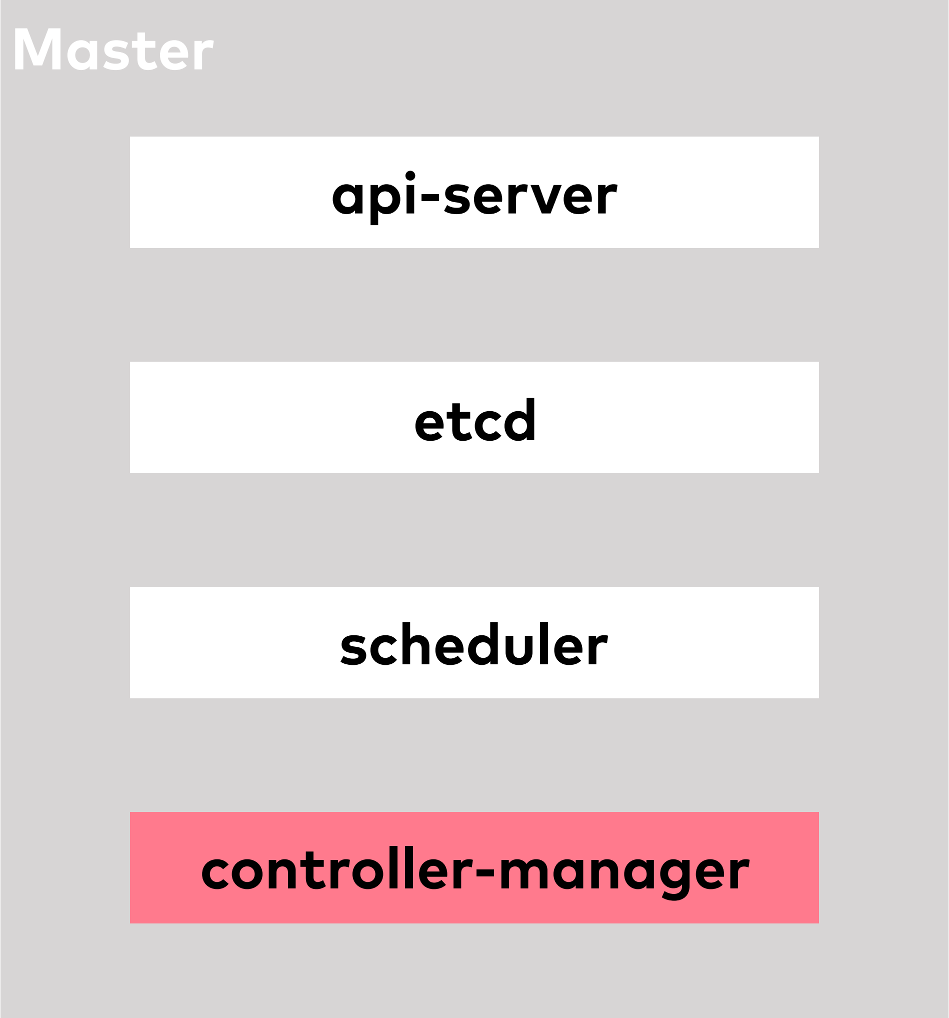
etcd

scheduler

controller-manager

Controller-Manager

- Manages / runs the controllers responsible for certain tasks in Kubernetes



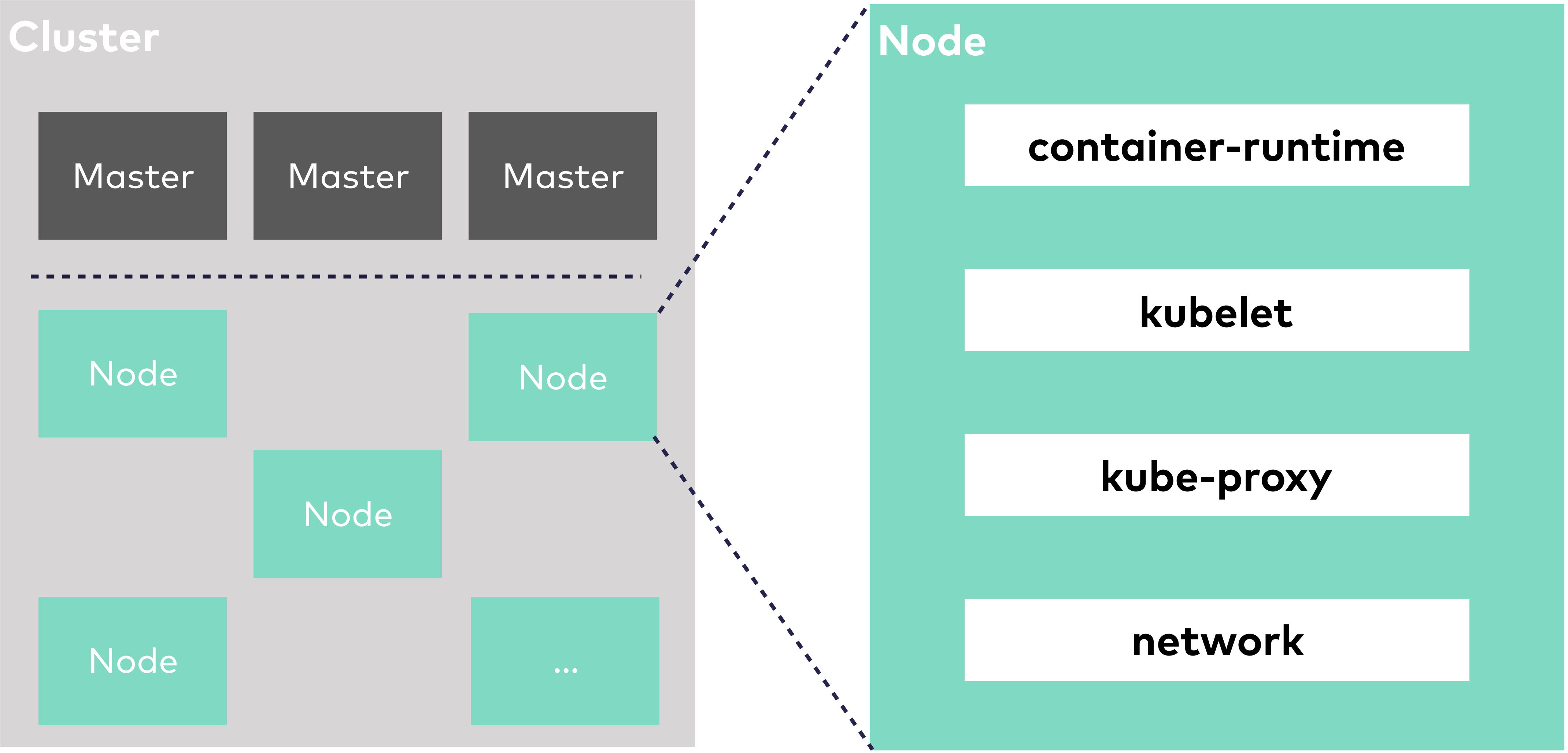
Kubernetes API — Controller

- Watch the Api Server for changes
- perform Operations on changes
- Creation/ Deletion or Update on other API objects
- Running a reconciliation loop

Controller examples

- Node Controller
- Replication Controller
- Cloud Volume Controller
- DNS Controller
- (your controller)

Node Components



Cluster

Master

Master

Master

Node

Node

Node

Node

...

Node

container-runtime

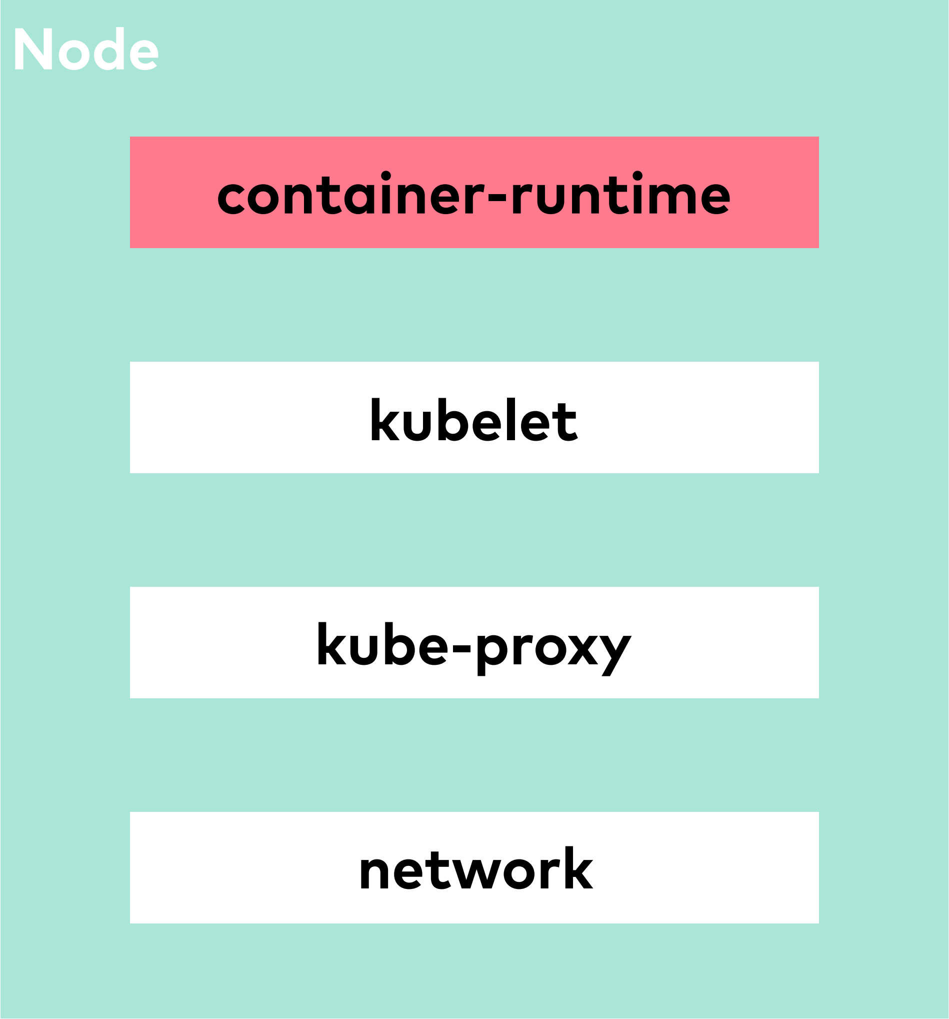
kubelet

kube-proxy

network

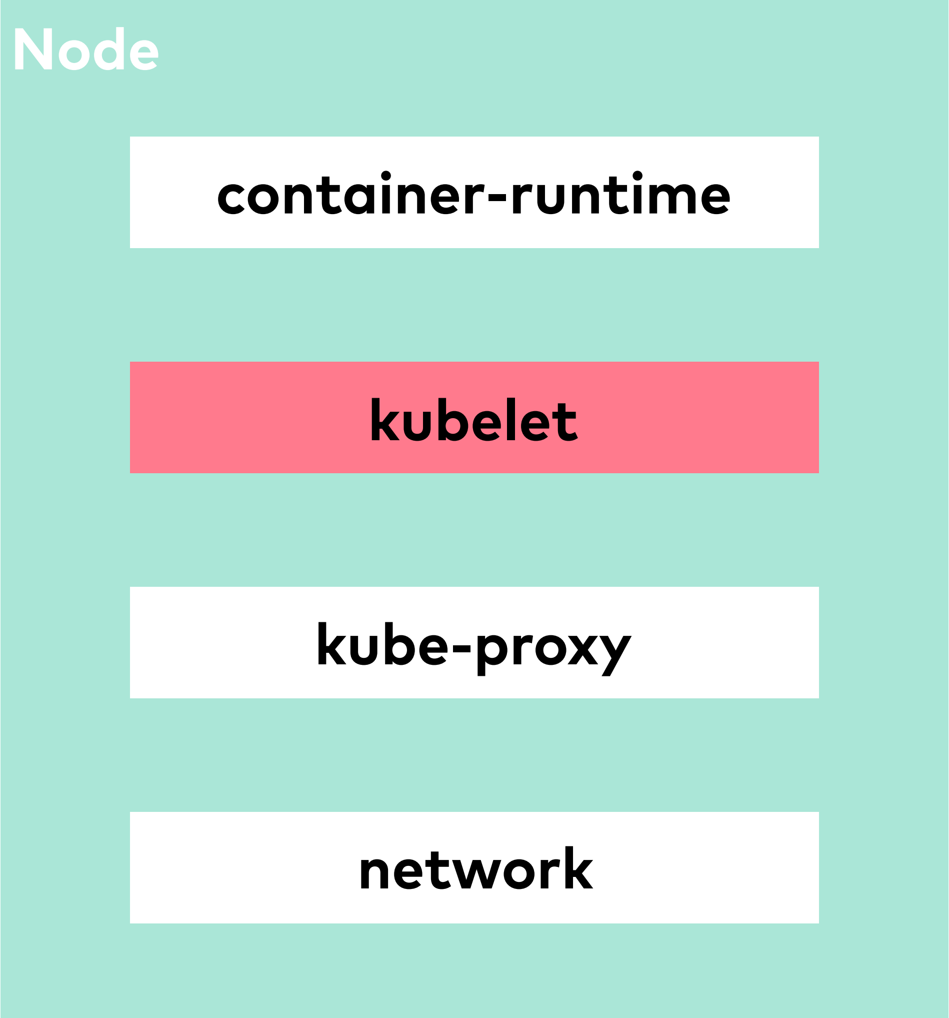
Container runtime

- Component to run containers on nodes
- Usually Docker
- Can be other implementation (e.g. rkt)

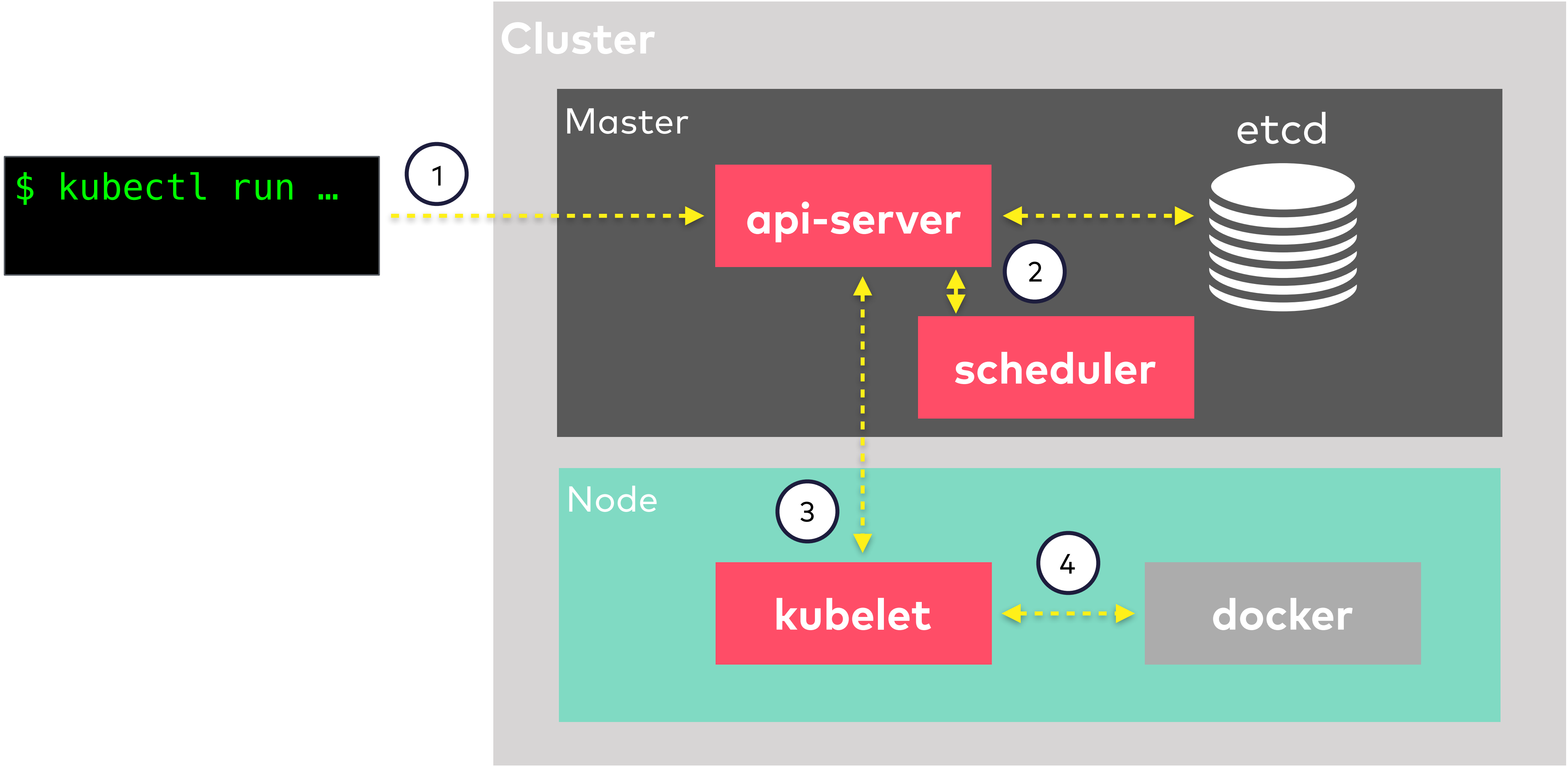


Kubelet

- Reads PodSpecs from the API
- Uses container-runtime to run Pods according to Spec



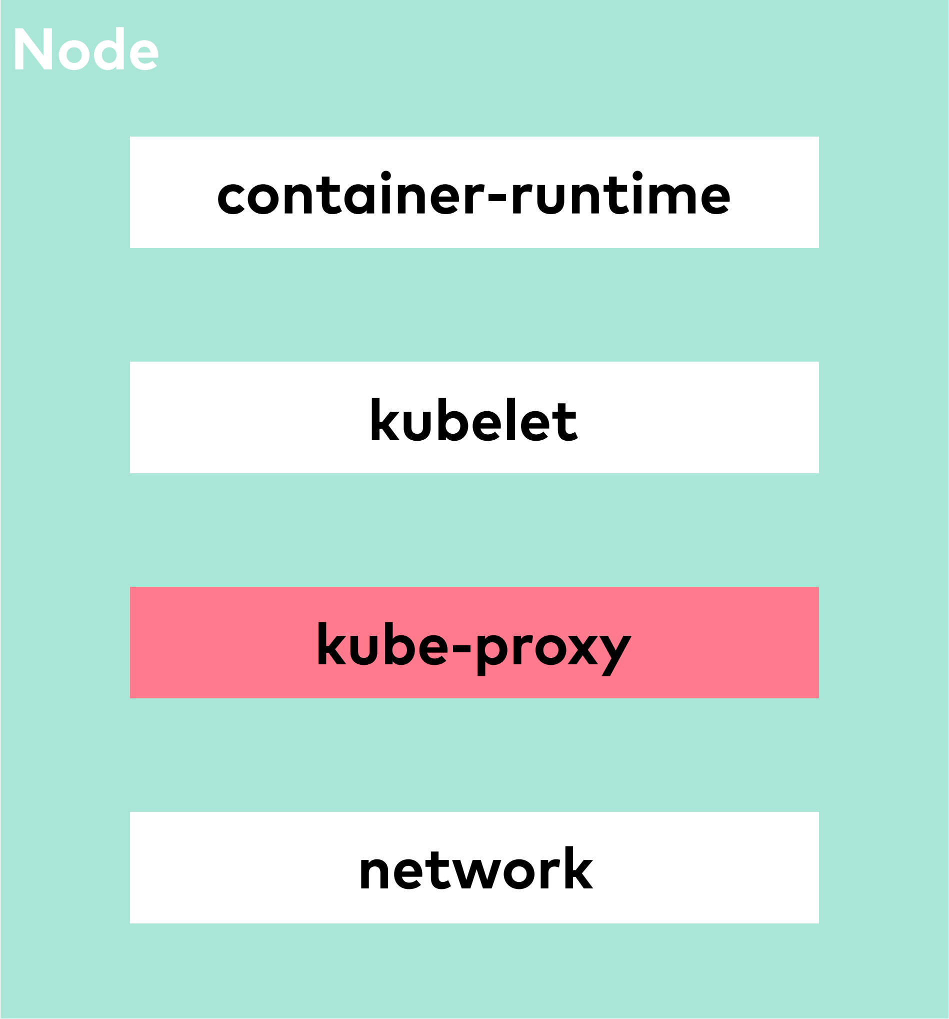
Running a Pod



```
$ kubectl run …
```

Kube-Proxy

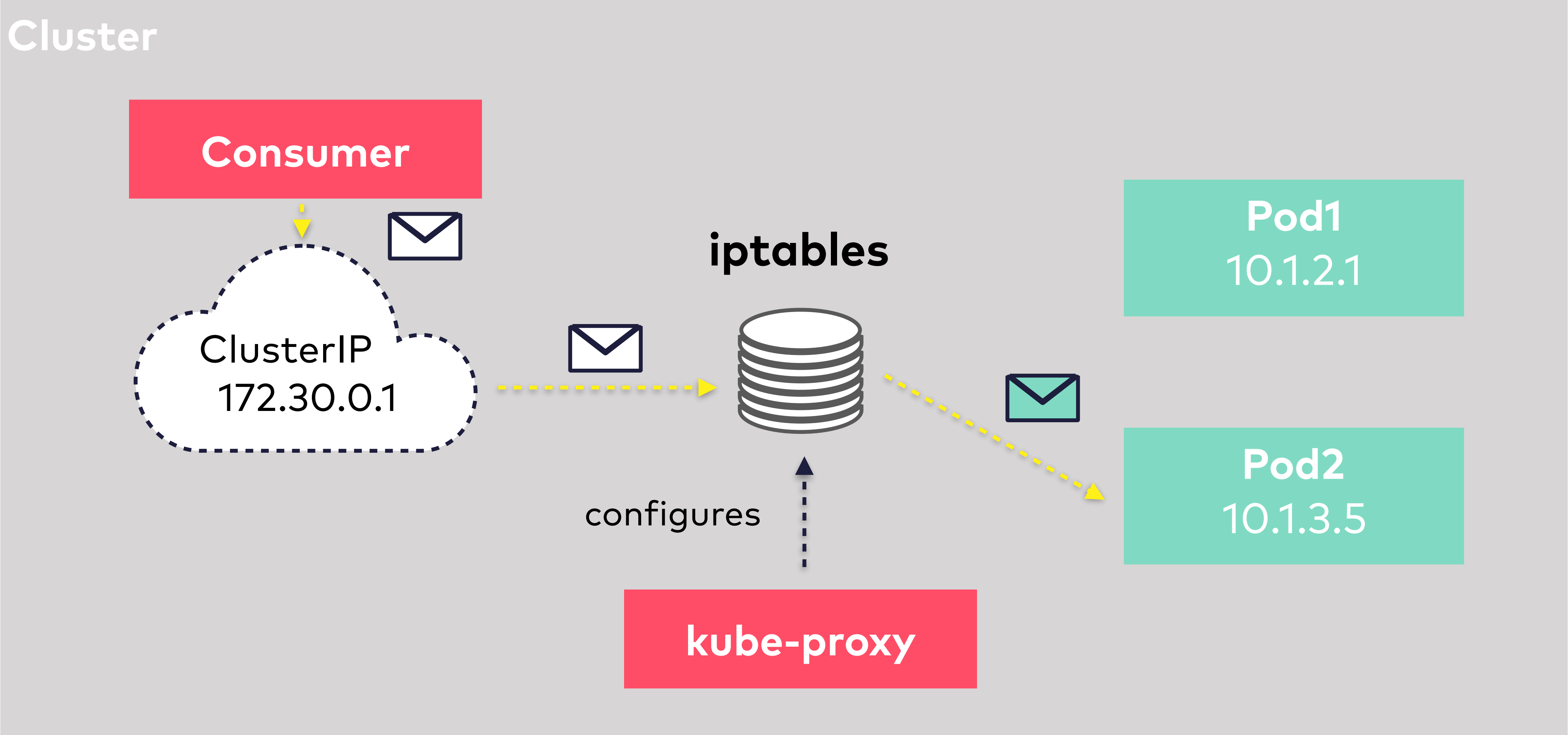
- responsible for service abstraction
- Classic proxy in usermode (old)
- New mode uses iptables to implement routing



Kube-Proxy & Services

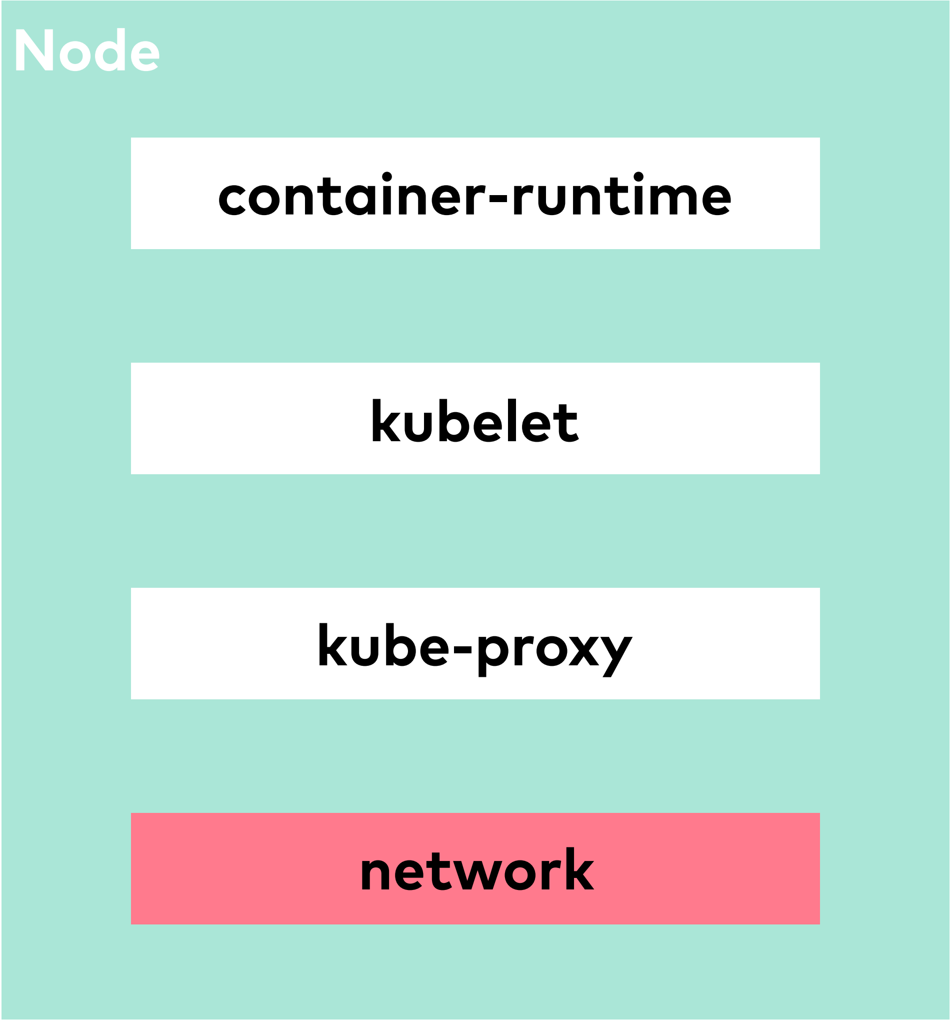
- Service has virtual address
- Kube-proxy updates IP-Tables on Node
- Any packet with the virtual address/port combination will be changed to a node-ip and port combination
- Overlay network will then do the rest
- see IPs (172 - virtual) (10.1.x nodes)

Kube-Proxy iptables mode

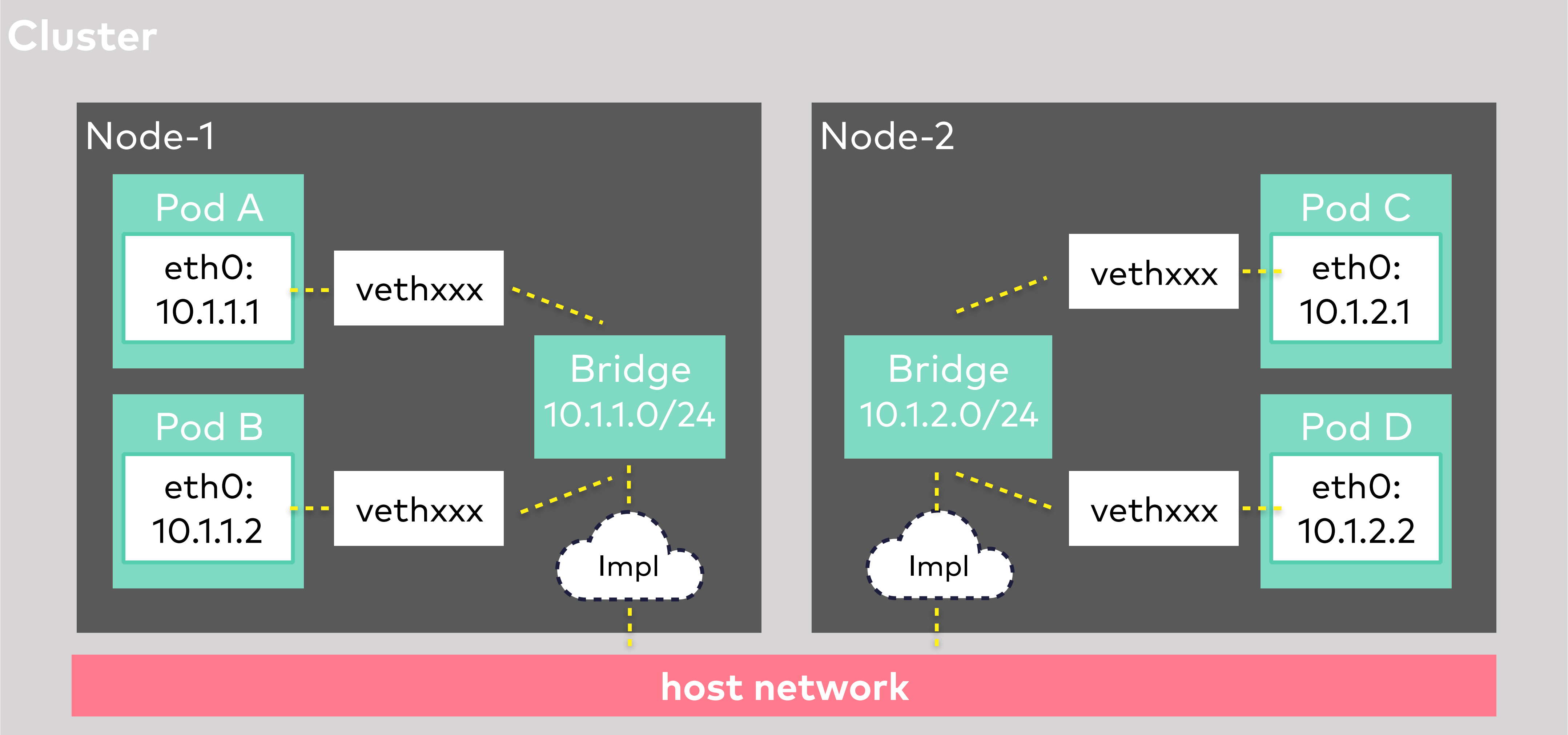


Network

- Making sure that Pods can connect across nodes
- No NAT in Cluster
- different implementations of the Container Network Interface (CNI)



Network basic example



Deploying complex applications

Helm

Helm

- Package management for Kubernetes: update, rollback, create, version, share, and publish applications
- Ready to use Kubernetes-applications (but always check the sources — like … for real, do it)
- Handy for deployment*: persistent history and easy rollback for free
- https://helm.sh/

Helm cont.

- Part of Cloud Native Computing Foundation
- Includes the possibility to template Kubernetes config files (e.g. for handling different clusters with the same configs)

DEMO

github.com/JoergM/kubernetes_workshop_demos/helm

Operators

Operators

- Idea to automate the knowledge of a human Operator
- Not only install automated, but **operate** automated
- The Operator itself run on Kubernetes too
- Uses Kubernetes API to do his job
- https://coreos.com/operators/

Advanced Features

- Create Backups
- Autoscale
- Autoupdate installed Software

Operator examples

- etcd
- Vault
- Prometheus
- Elasticsearch
- Kafka

Operator Framework

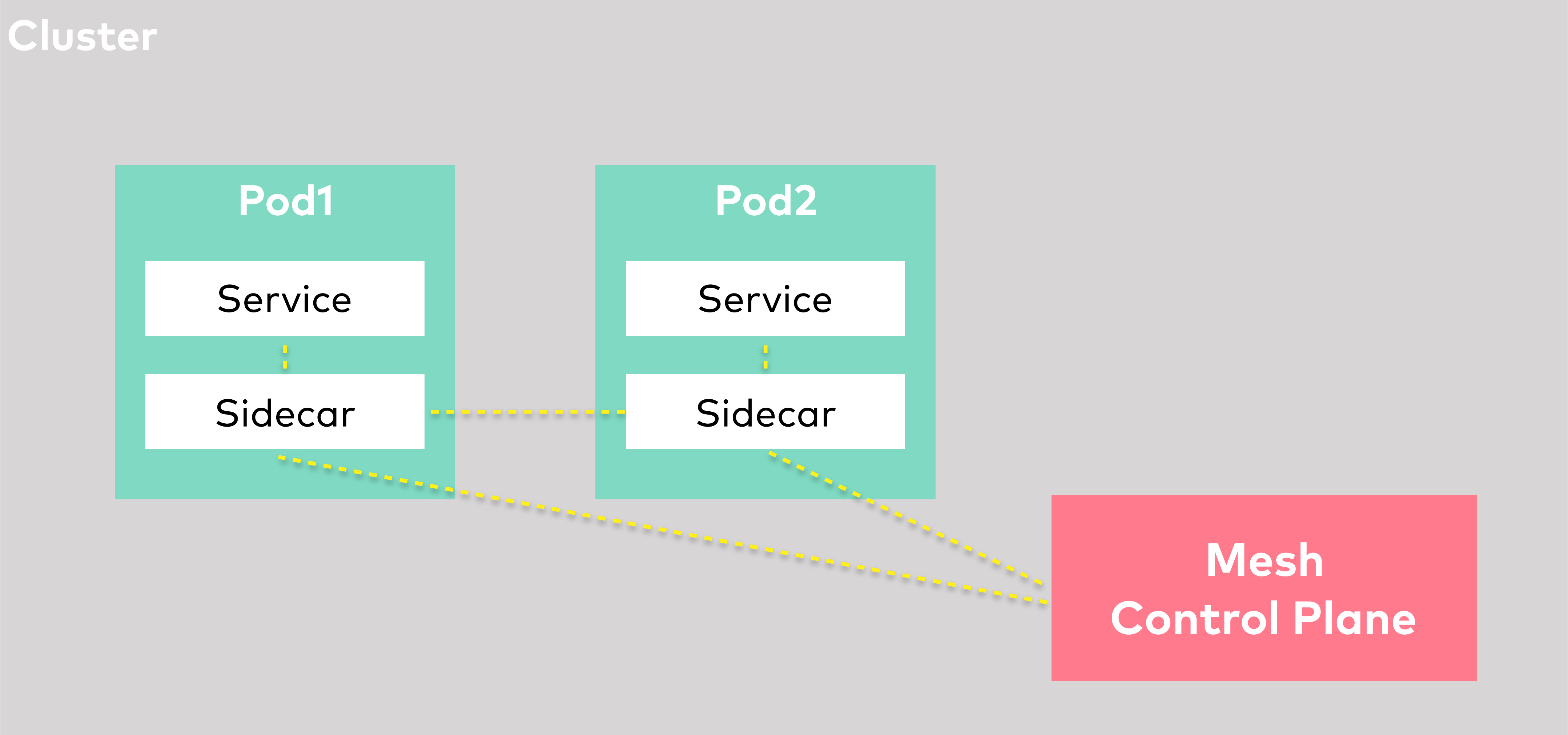
- Published on KubeCon 2018
- Operator SDK to create your own Operators
- Operator Lifecycle Manager - managing operators in a cluster
- Operator Metering - gathering data on operators

Service meshes

Service Meshes

- Providing common infrastructure for microservices
- Features
  - Circuit Breaking
  - TLS
  - Authentication
  - Tracing
  - …

Service Mesh basics



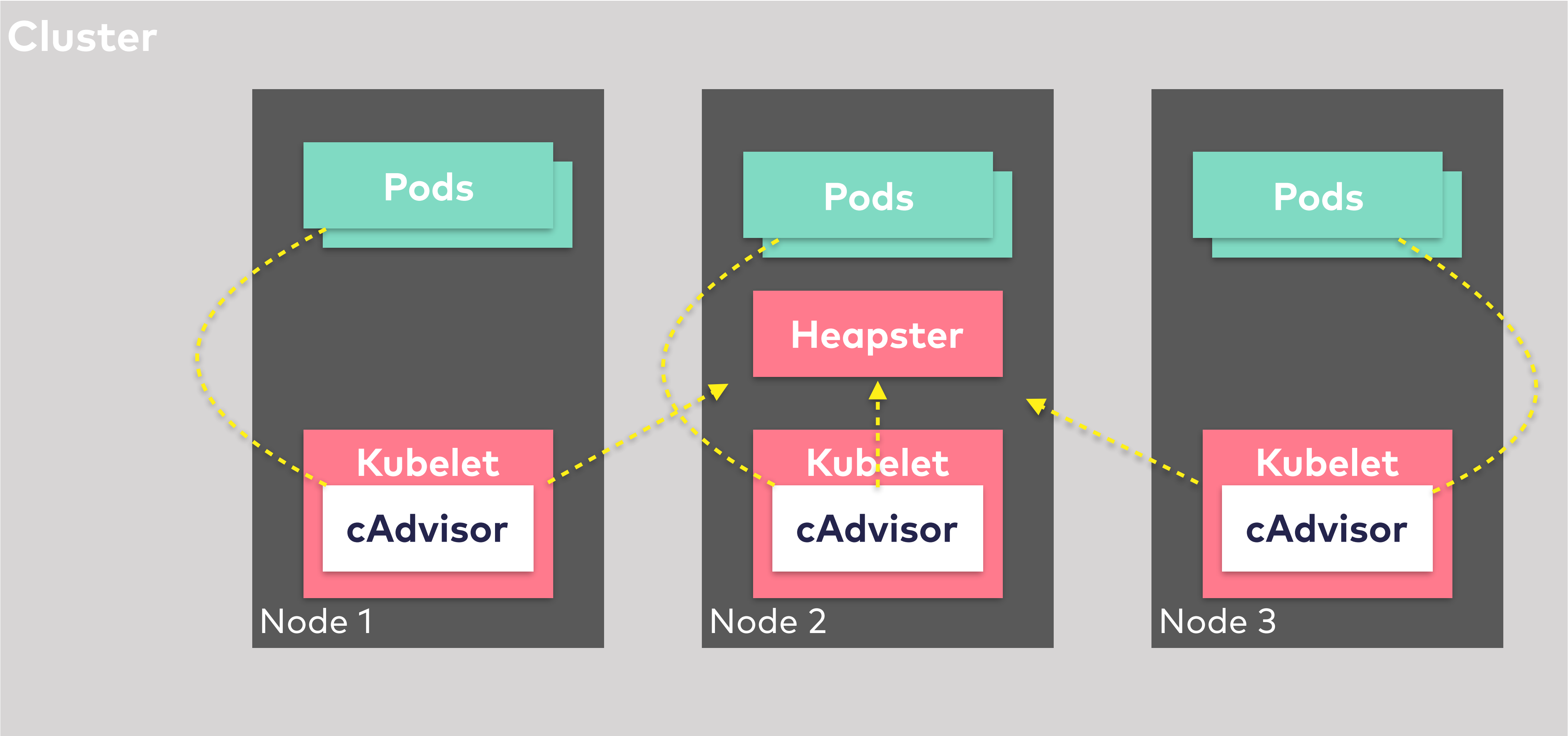
Projects to look at

- linkerd (https://linkerd.io/)
- Envoy Proxy (https://www.envoyproxy.io/)
- Istio (https://istio.io/)
- Conduit (https://conduit.io/)

Production readiness

Monitoring

Monitoring Overview



cAdvisor & Heapster

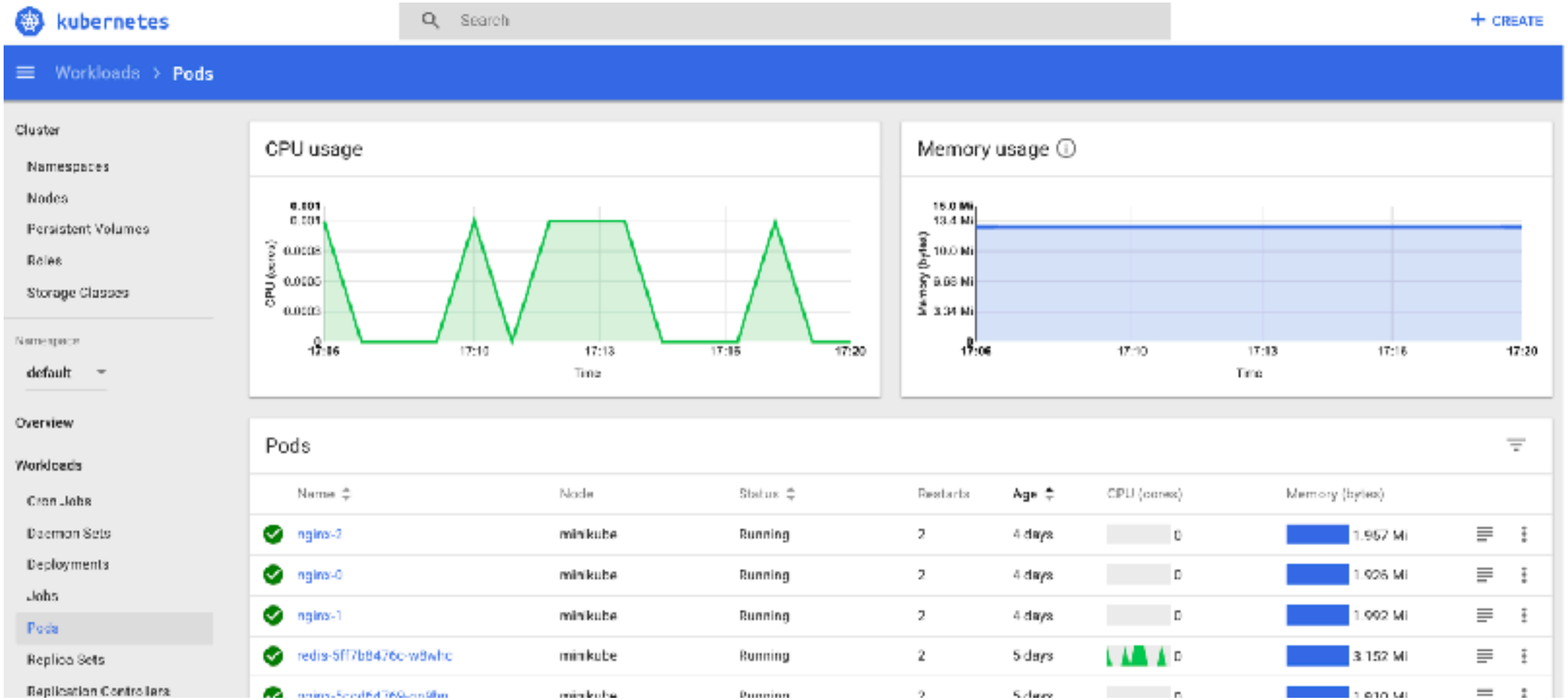
- cAdvisor is integrated into kubelet
  - collects performance data of containers on node
  - and on the node itself
- Heapster is running as a pod inside the cluster
  - collects data from all nodes
  - makes them available for other tools
  - Command line, dashboard, Influx, Prometheus …

Command line

```
$ kubectl top node
NAME       CPU(cores)   CPU%      MEMORY(bytes)   MEMORY%
minikube   226m         11%       880Mi           22%


$ kubectl top pods --all-namespaces
NAMESPACE     NAME                                  CPU(cores)   MEMORY(bytes)
default       nginx-5ccd64769-gn9bn                 0m           1Mi
kube-system   influxdb-grafana-rl265                2m           84Mi
kube-system   kube-addon-manager-minikube           91m          50Mi
kube-system   kube-dns-54cccfbdf8-2r526             1m           23Mi
kube-system   kubernetes-dashboard-77d8b98585-md5   3m           13Mi
…
```

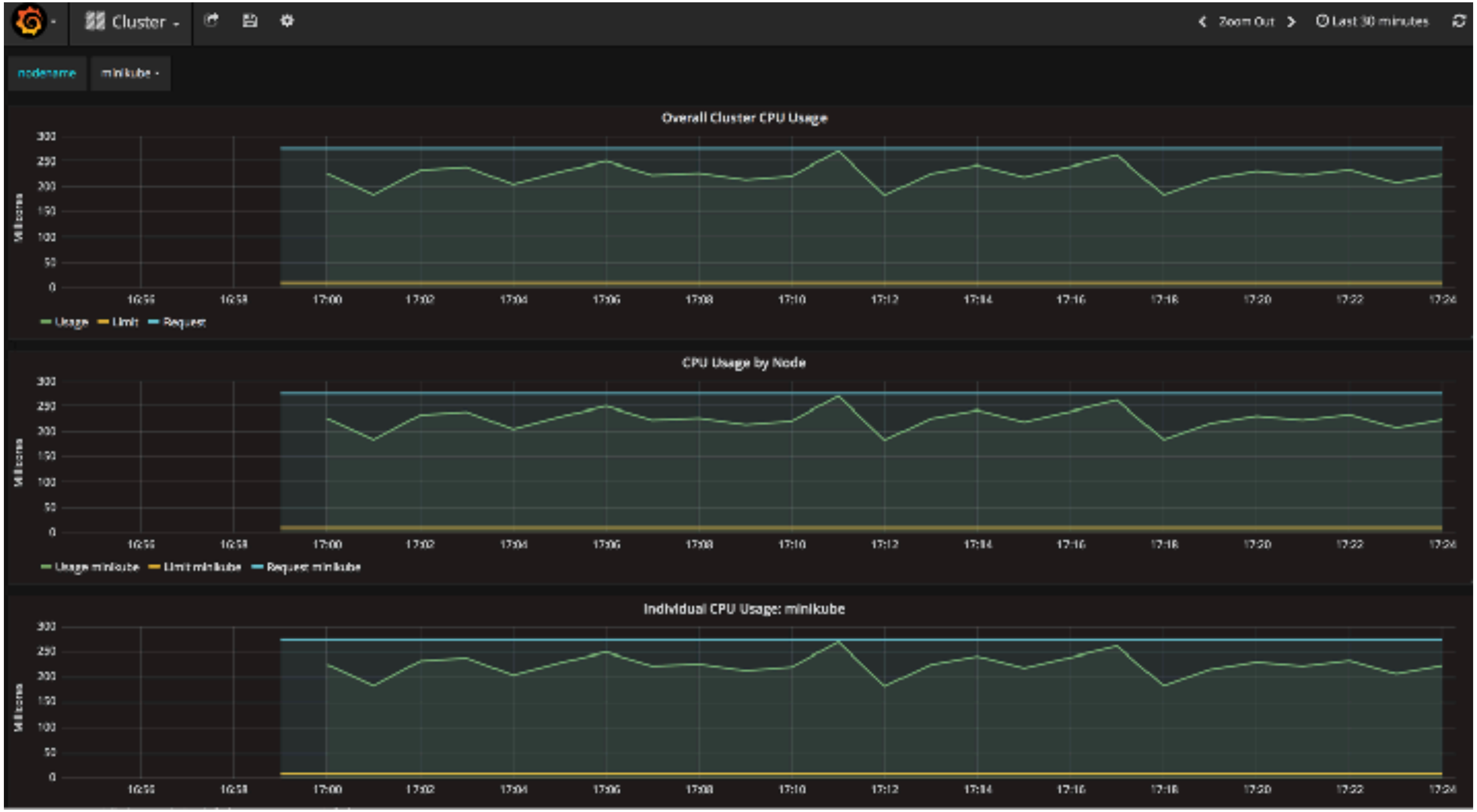
Dashboard



Pods

	Name	Node	Status	Restarts	Age	CPU (cores)	Memory (bytes)
✓	nginx-2	minikube	Running	2	4 days	0	1.957 Mi
✓	nginx-0	minikube	Running	2	4 days	0	1.926 Mi
✓	nginx-1	minikube	Running	2	4 days	0	1.992 Mi
✓	redis-5f7b8476c-w8whc	minikube	Running	2	5 days	0	3.152 Mi
✓	[illegible]	minikube	Running	2	5 days	0	1.910 Mi

Grafana



Cluster
Zoom Out
Last 30 minutes
nodename
minikube
Overall Cluster CPU Usage
Millicores
300
250
200
150
100
50
0
16:56
16:58
17:00
17:02
17:04
17:06
17:08
17:10
17:12
17:14
17:16
17:18
17:20
17:22
17:24
Usage
Limit
Request
CPU Usage by Node
Usage minikube
Limit minikube
Request minikube
Individual CPU Usage: minikube

DEMO

github.com/JoergM/kubernetes_workshop_demos/monitoring

Managing load

Ressource requests

- Requirements stated at container level
- Primarly CPU and Memory
- Scheduler uses values to find best Node

```
apiVersion: v1
kind: Pod
metadata:
  name: example-pod
spec:
  containers:
  - image: alpine
    name: foo
    resources:
      requests:
        cpu: 100m
        memory: 25Mi
```

Ressource limits

- Limits limit the resources available to a container
- If CPU exceeds limit it will be throttled
- If memory exceeds limit pod will be killed

```
apiVersion: v1
kind: Pod
metadata:
  name: example-pod
spec:
  containers:
  - image: alpine
    name: foo
    resources:
      limits:
        cpu: 100m
        memory: 25Mi
```

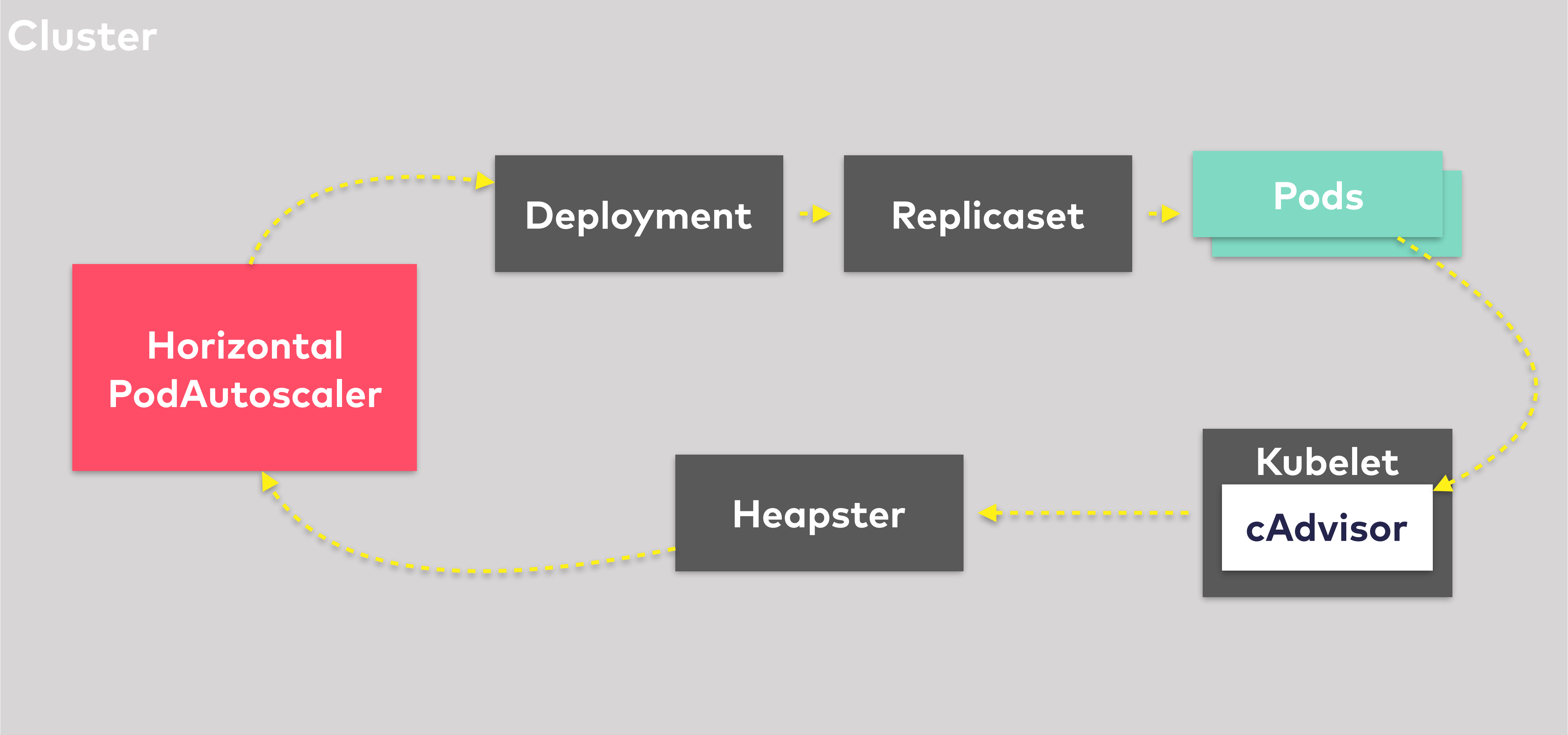
Limits and Requests

- Understand how limits and requests work
- Set them accordingly
- Be aware of resource visibility to container processes (esp. with Java applications)
  - Processes see node memory and cores

Two levels of autoscaling

- Scaling Nodes
  - Depending on underlying runtime enironment
  - Autoscaling Groups on AWS as usual
- Scaling Pods
  - Cluster internal
  - Of course limited to available nodes

Scaling Overview



Scaling Definition

- HorizontalPodAutoscaler API Object
- currently supports CPU and custom metrics

```
apiVersion: autoscaling/v2beta1
kind: HorizontalPodAutoscaler
metadata:
…
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: …
  minReplicas: 1
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      targetAverageUtilization: 50
```

Security

Disclaimer

- This is only scratching at the surface
- To truly secure your cluster learn about the concepts, try them yourself, let somebody else look at it

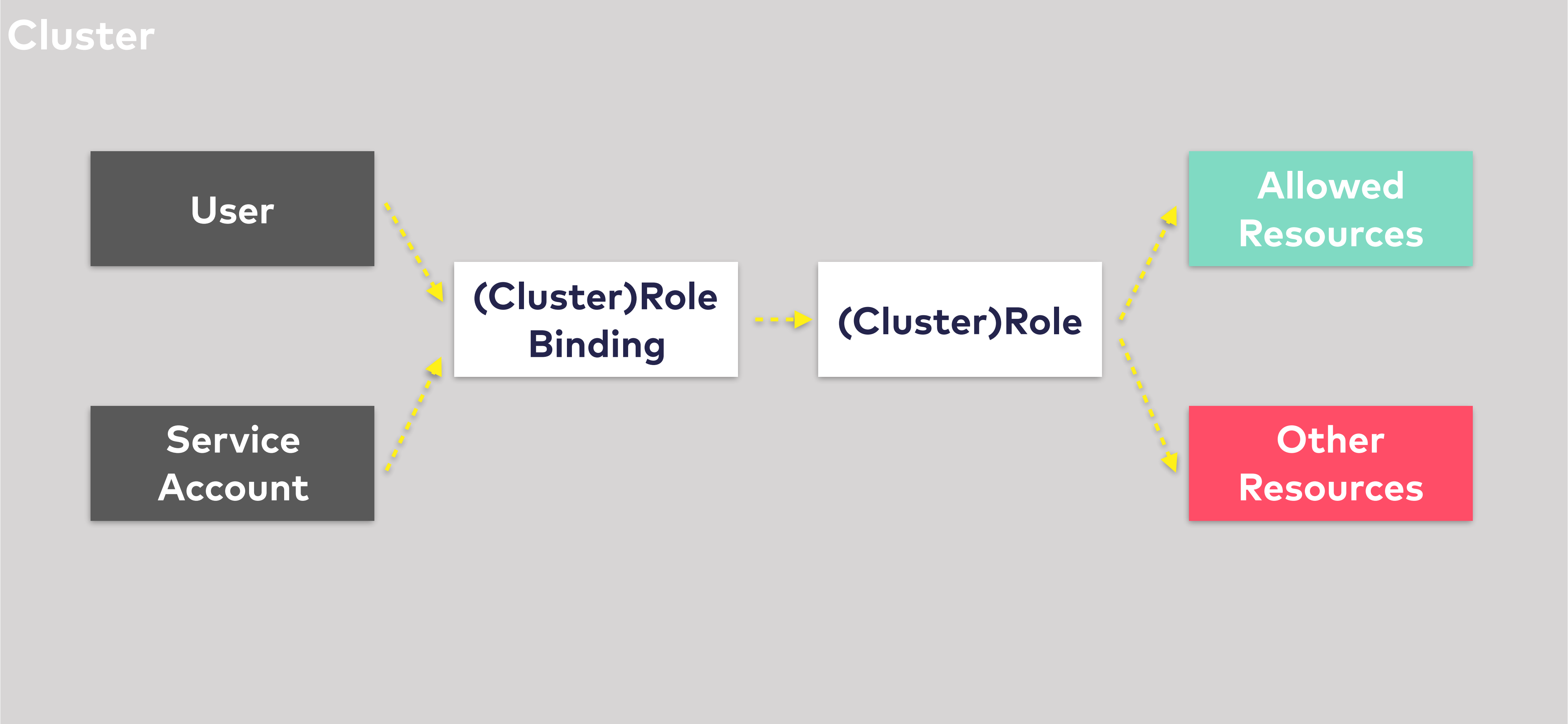
Securing the API

- Who is allowed to do what using the API
- Who is „Who“?
- How to identify?
- How to assign rights?

Users in Kubernetes

- Human users
  - Accessing the API using e.g. kubectl
  - several mechanisms to identify (X.509, tokens …)
  - managed externally
- Pods accessing the API
  - Pods are associated to Service Accounts
  - Namespace default or in Spec

Role Based Access Control



Pod Security Policies

- What is a Pod allowed to do?
  - User inside Pod? Is Root allowed?
  - What Kernel capabilities are allowed?
  - Read only filesystem
  - …
- Assigned using ClusterRoles

Network Policies

- Availability depending on installed network layer
- By default every pod can be accessed (need to change)
- can isolate single pods but also namespaces

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: postgres-netpolicy
spec:
  podSelector:
    matchLabels:
      app: database
  ingress:
  - from:
    - podSelector:
        matchLabels:
          app: webserver
    ports:
    - port: 5432
```

- architecture, development, devOps
- focus on platform & infrastructure

Jörg Müller
Principal Consultant
innoQ Deutschland GmbH

joerg.mueller@innoq.com
@joergm

INNOQ

www.innoq.com

SERVICES

Strategy & technology consulting

Digital business models

Software architecture & development

Digital platforms & infrastructures

Knowledge transfer, coaching & trainings

FACTS

~125 employees

Privately owned

Vendor-independent

OFFICES

Monheim

Berlin

Offenbach

Munich

Zurich

CLIENTS

Finance

Telecommunications

Logistics

E-commerce

Fortune 500

SMBs

Startups