Preparation

Send me a **mail** with your **Google account** address.

(You need this to participate)

To:

l.burchard@campus.tu-berlin.de

Kubernetes

From Beginner to Expert

Containers

A quick recap of Containers

- Lightweight
- Hermetically sealed
- Isolated

- Easily deployable
- Introspectable
- Runnable

Linux processes

- Improves overall developer experience
- Fosters code and component reuse
- Simplifies operations for cloud native applications

Docker

rkt

Everything at Google runs in containers:

- Gmail, Web Search, Maps, ...
- MapReduce, batch, ...
- GFS, Colossus, ...
- Even **Google's Cloud Platform:** VMs run in containers!

They launch over **2 billion** containers **per week**

Shipping Containers At Clyde, by Steve Gibson

Containers are awesome! Let’s run lots of them!

Kubernetes

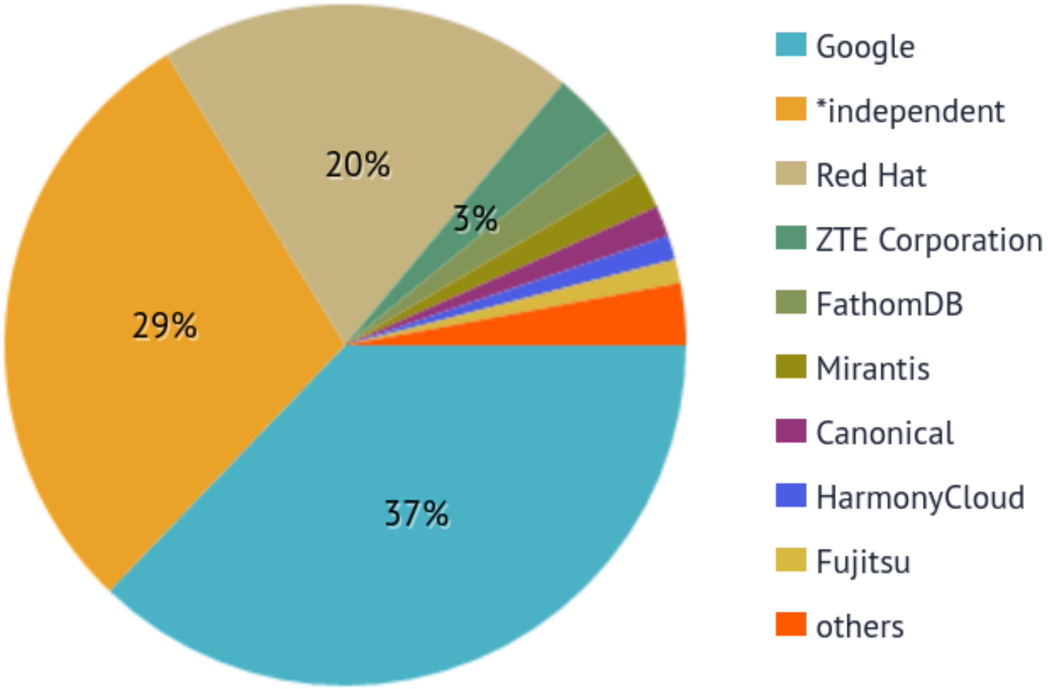
Κυβερνήτης

Greek for "helmsman" or "pilot"

Is Kubernetes Google ?

CLOUD NATIVE COMPUTING FOUNDATION

- Under half the code is now written by Google
- Stewarded by the Cloud Native Compute Foundation™
- A Linux Foundation Collaborative Project™



Start with a Cluster

Laptop to high-availability **multi-node cluster**

Hosted or **self managed**

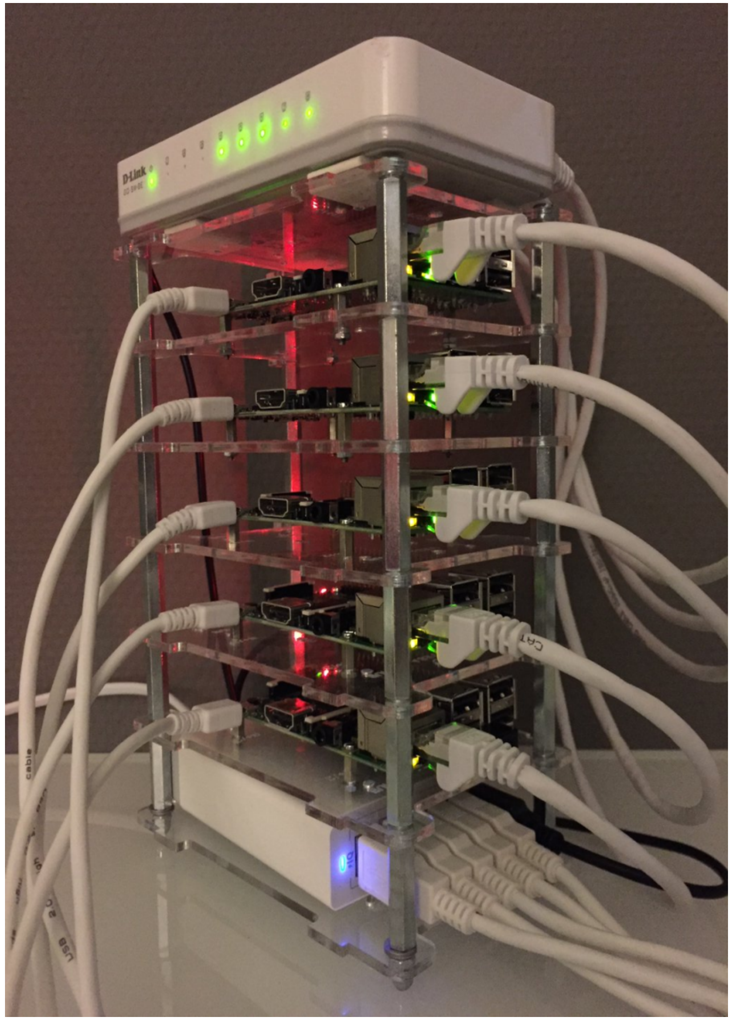
On-Premise or **Cloud**

Bare Metal or **Virtual Machines**

Most OSes (inc. **RedHat Atomic, Fedora, CentOS**)

Or just a bunch of **Raspberry PIs**

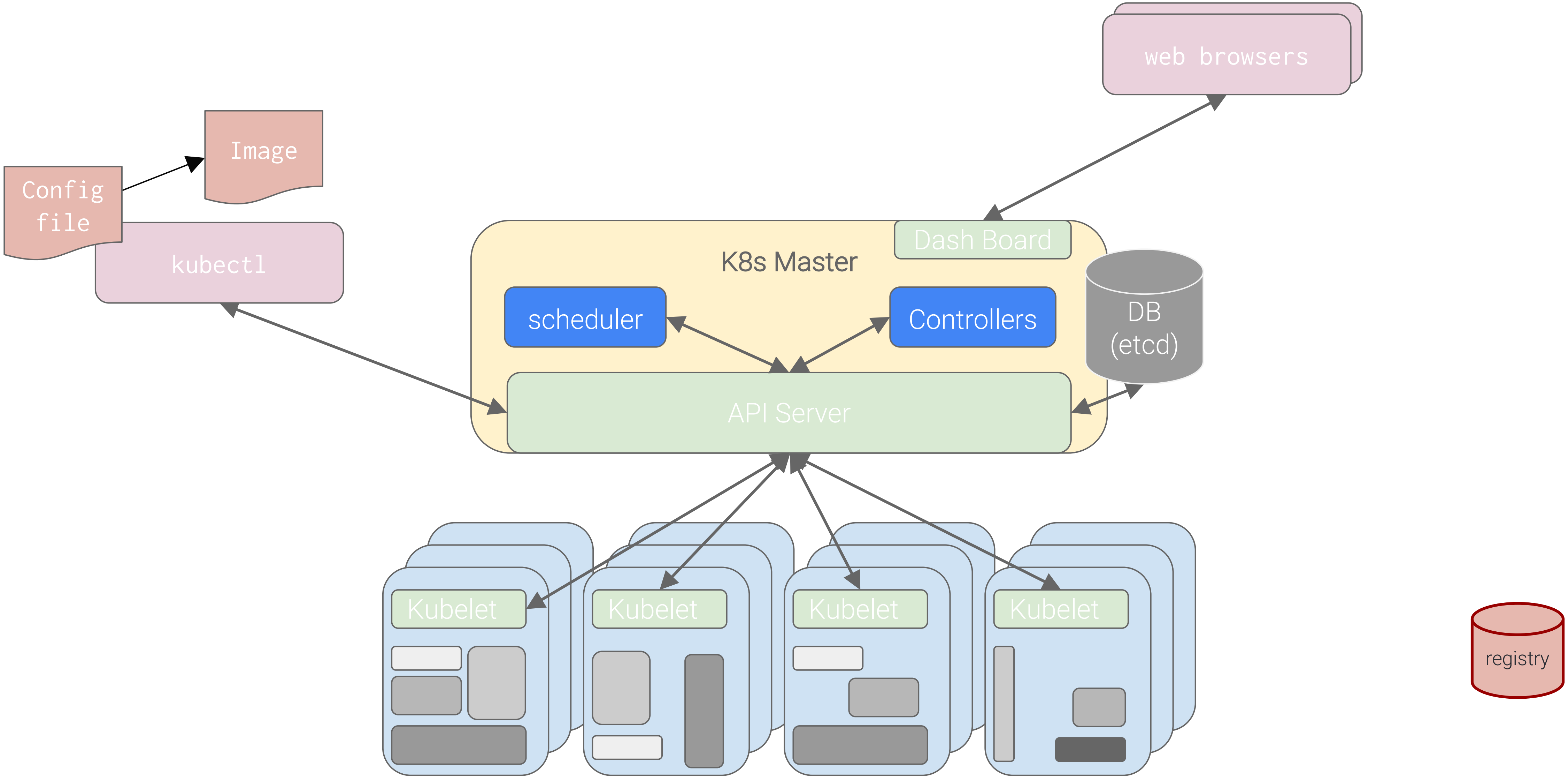
Many options, See Matrix for details



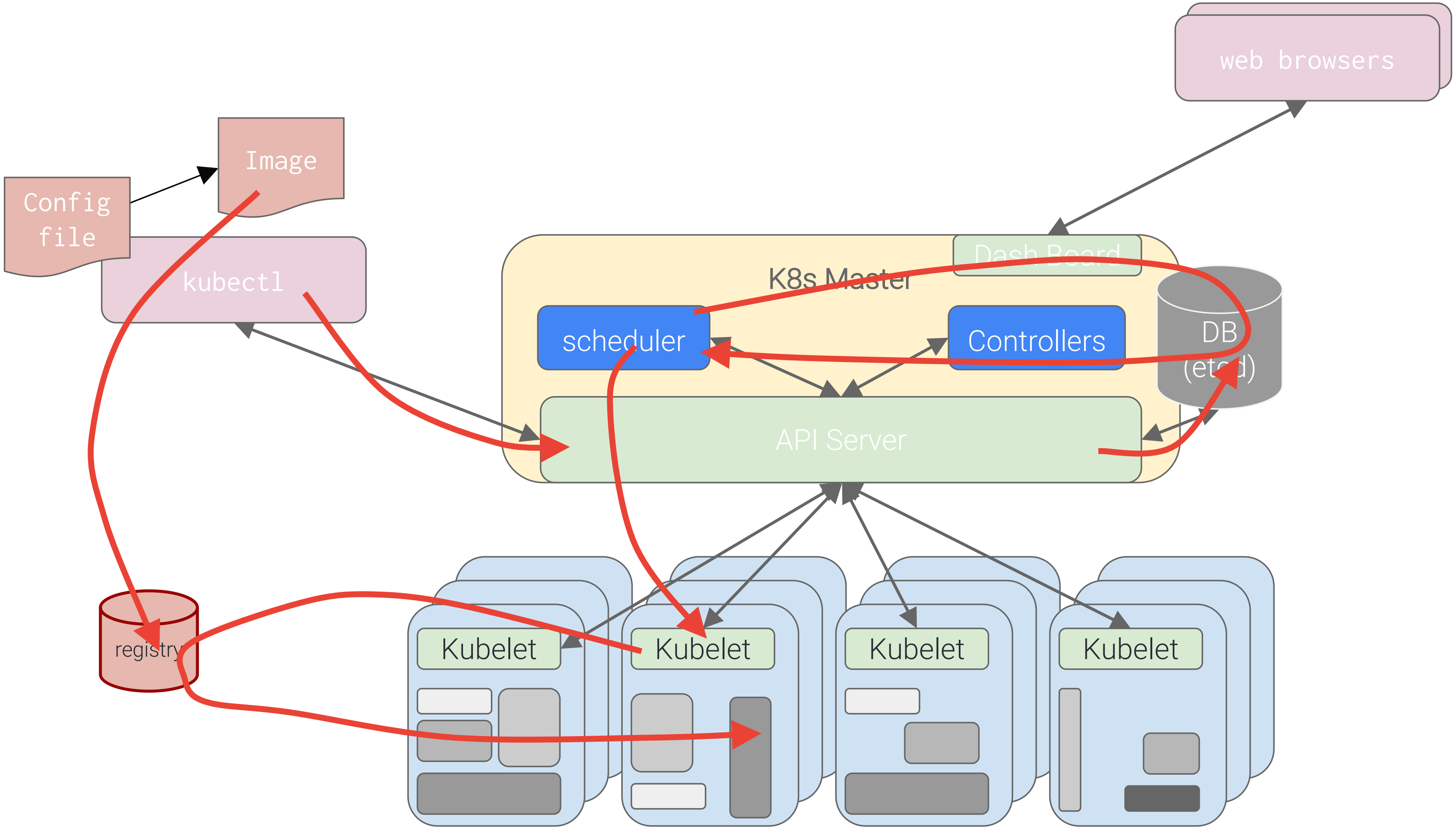
Kubernetes Cluster Matrix: http://bit.ly/1MmhpMW

Declarative

Don‘t micromanage anything!


web browsers
Image
Config file
kubectl
Dash Board
K8s Master
scheduler
Controllers
DB (etcd)
API Server
Kubelet
Kubelet
Kubelet
Kubelet
registry

loodse


web browsers
Image
Config file
kubectl
Dash Board
K8s Master
scheduler
Controllers
DB (etcd)
API Server
registry
Kubelet
Kubelet
Kubelet
Kubelet

loodse

loodse

A pod of ~~whales~~ containers

The atom of scheduling for containers

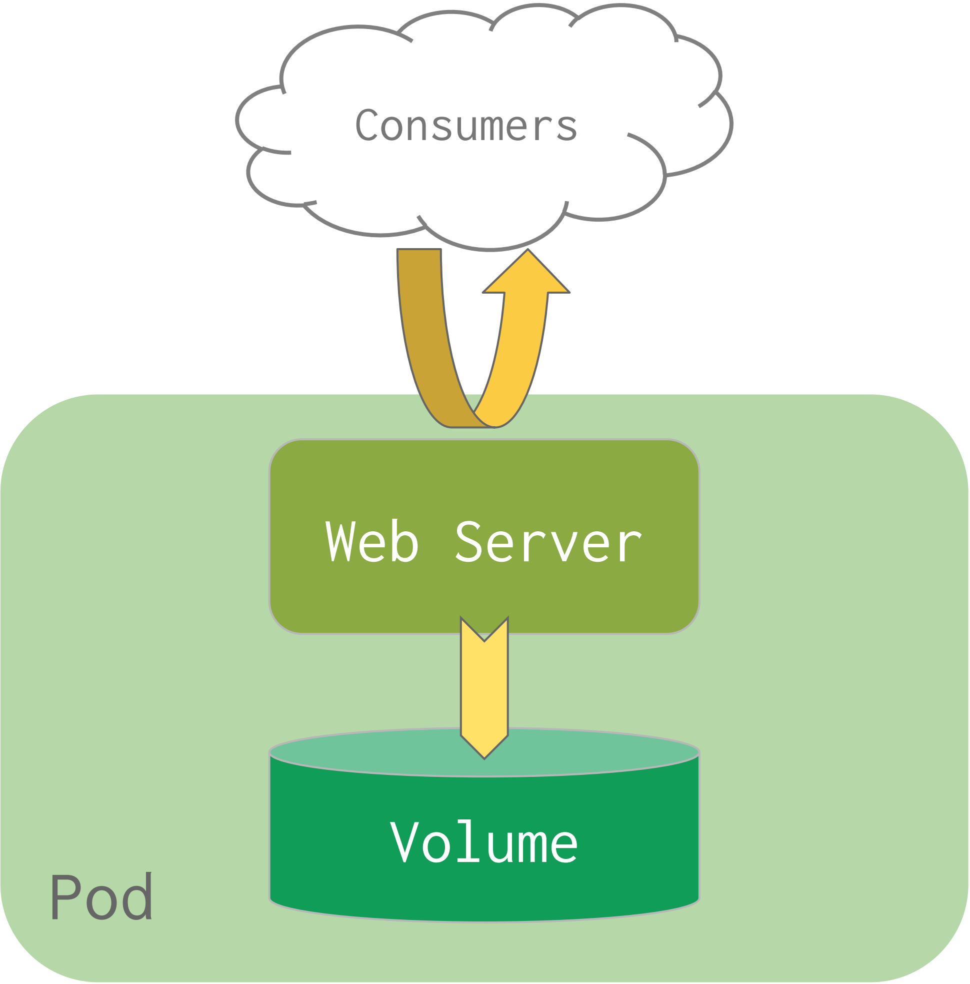
Represents an application specific **logical host**

Hosts **containers** and **volumes**

Each has its own routable (no NAT) IP address

Ephemeral

- Pods are functionally identical and therefore ephemeral and replaceable



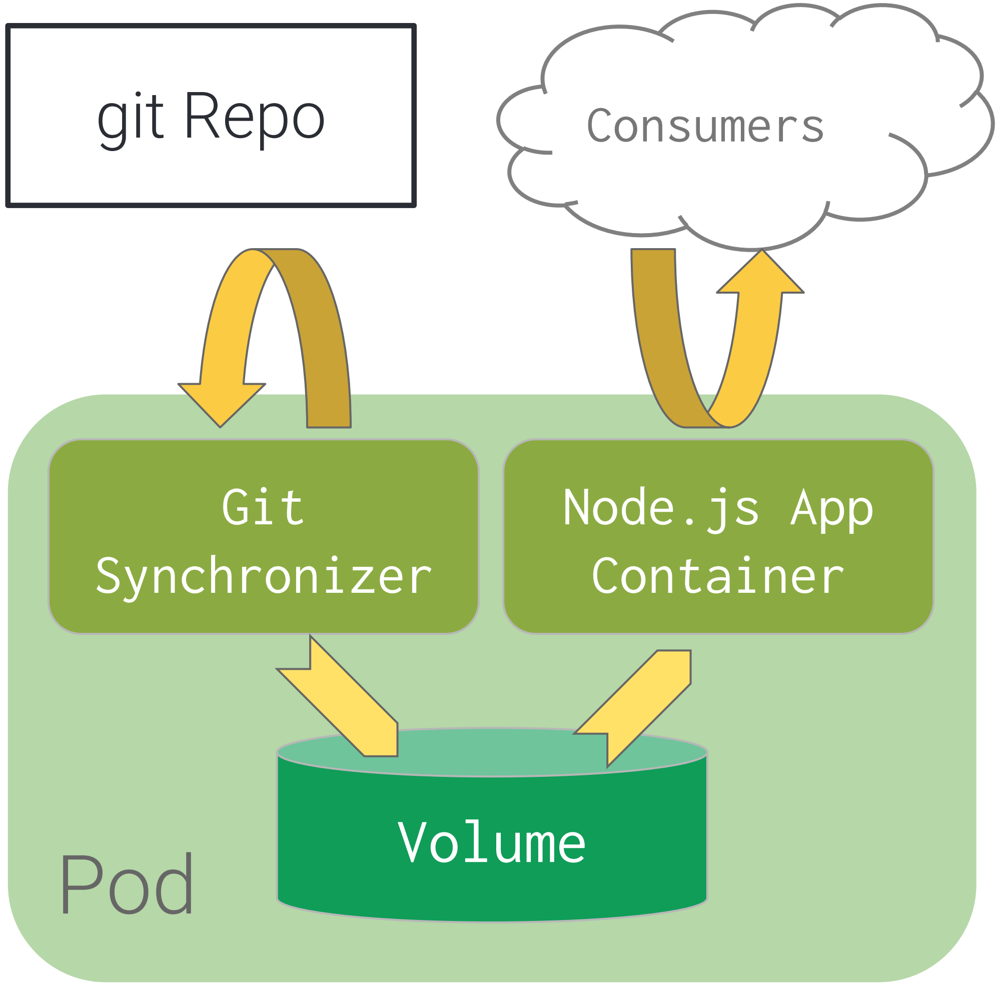
Pods

Can be used to group multiple containers & shared volumes

Containers within a pod are **tightly** coupled

Shared namespaces

- Containers in a pod share IP, port and IPC namespaces
- Containers in a pod talk to each other through localhost



Pod Networking (across nodes)

Pods have IPs which are routable

Pods can reach each other without NAT

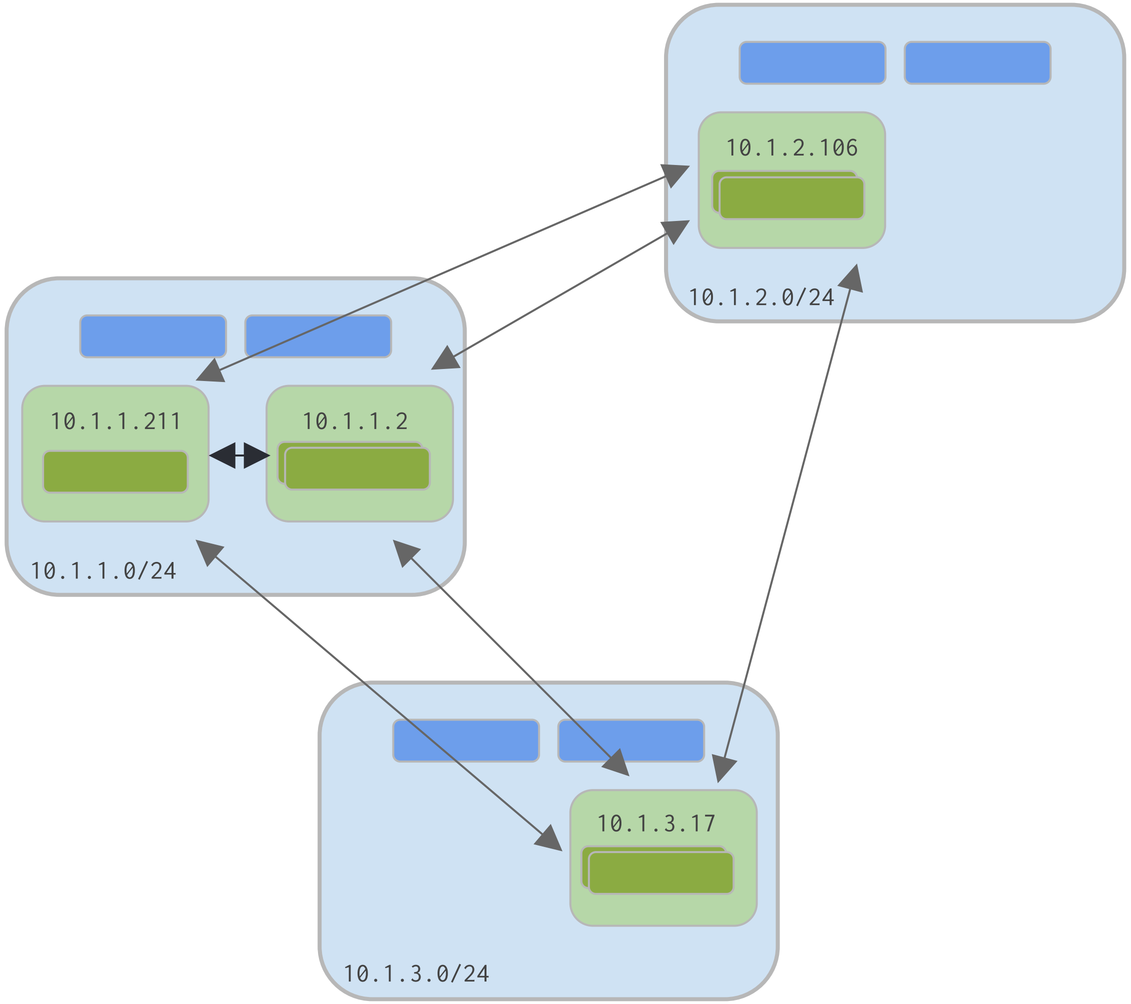
- Even across nodes

No Brokering of **Port Numbers**

These are fundamental requirements

Many solutions

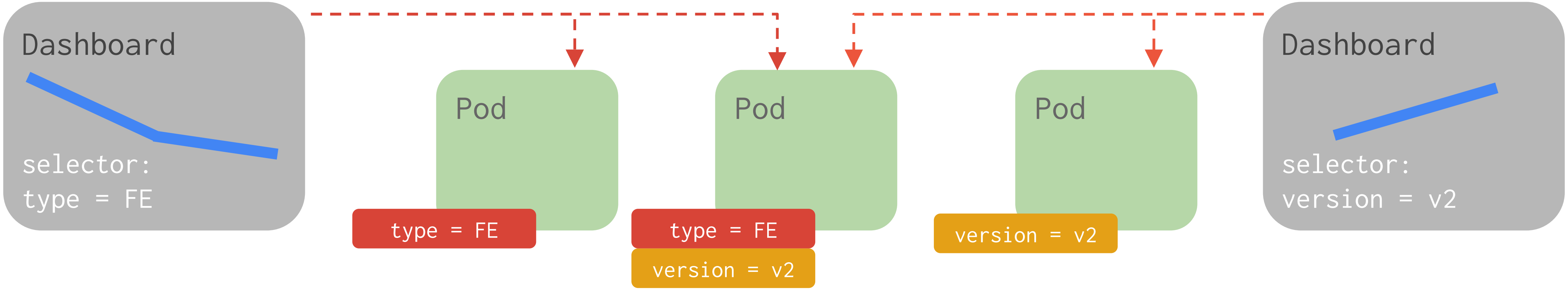
- GCE Advanced Routes, AWS Flannel, Weave, OpenVSwitch, Cloud Provider



Get the material

https://goo.gl/jXK36F

Labels



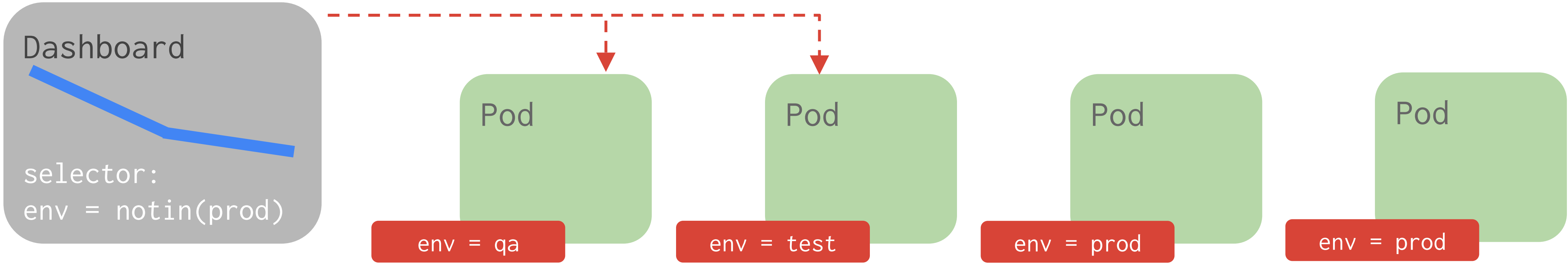
Behavior

- Metadata with semantic meaning
- Membership identifier
- The only Grouping Mechanism

Benefits

- Allow for intent of many users (e.g. dashboards)
- Build higher level systems …
- Queryable by Selectors

Label Expressions

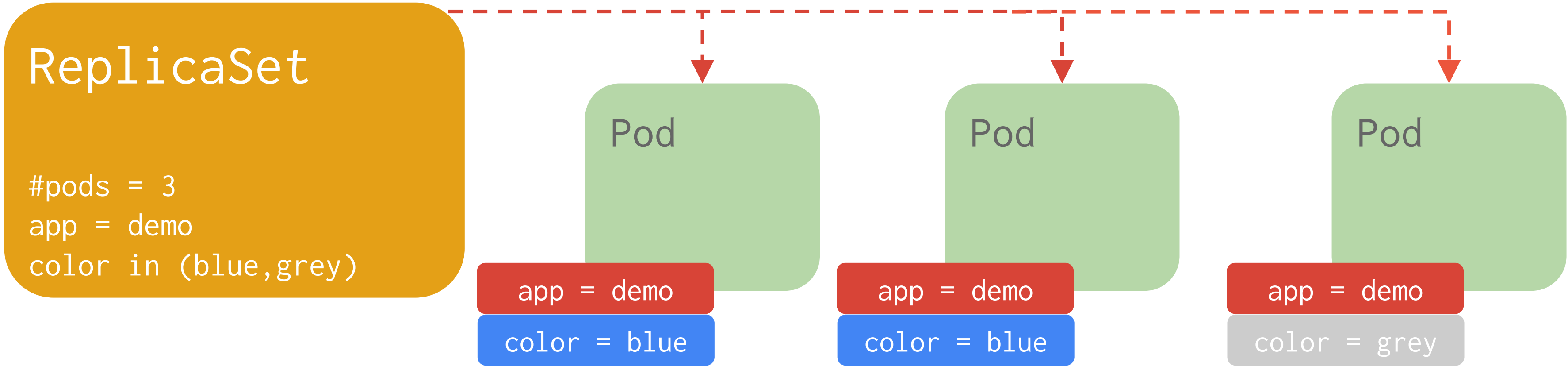


Expressions

- env = prod
- tier != backend
- env = prod, tier !=backend

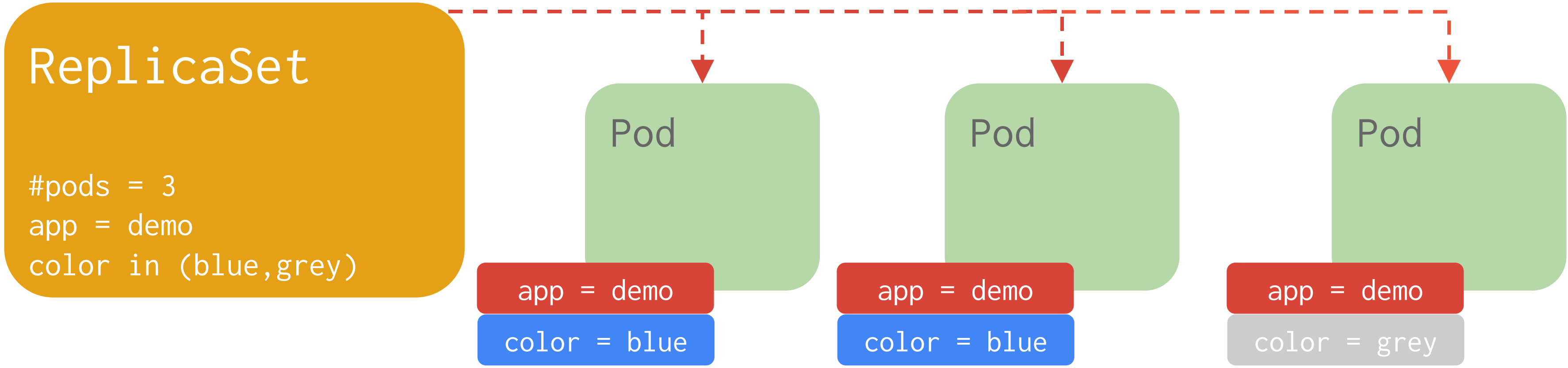
- env in (test,qa)
- release notin (stable,beta)
- tier
- !tier

ReplicaSet



Behavior	Benefits
● Keeps Pods running	→ Recreates Pods, maintains desired state
● Gives direct control of Pod #s	→ Fine-grained control for scaling
● Grouped by Label Selector	→ Standard grouping semantics

ReplicaSet



Supports generalized Selectors

```
selector:
  matchLabels:
    app: demo
  matchExpressions:
    - {key: color, operator: In, values:
[blue,grey]}
```

loodse

Replica Set

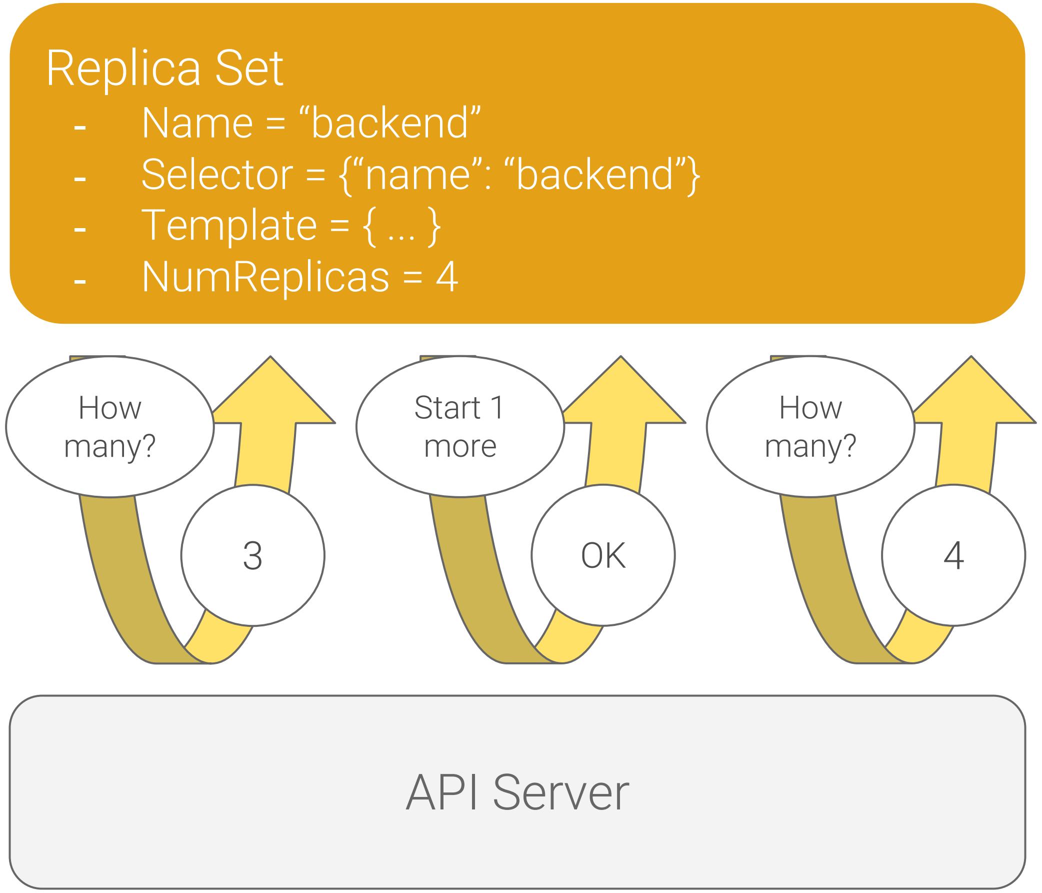
Canonical example of control loops

Have one job: ensure N copies of a pod

- if too few, start new ones
- if too many, kill some
- group == selector

Replicated pods are replaceable

- No implied order or identity



Replica Set

- Name = “backend”
- Selector = {“name”: “backend”}
- Template = { ... }
- NumReplicas = 4

How many?

3

Start 1 more

OK

How many?

4

API Server

Services

A logical grouping of pods that perform the same function (the Service's endpoints)

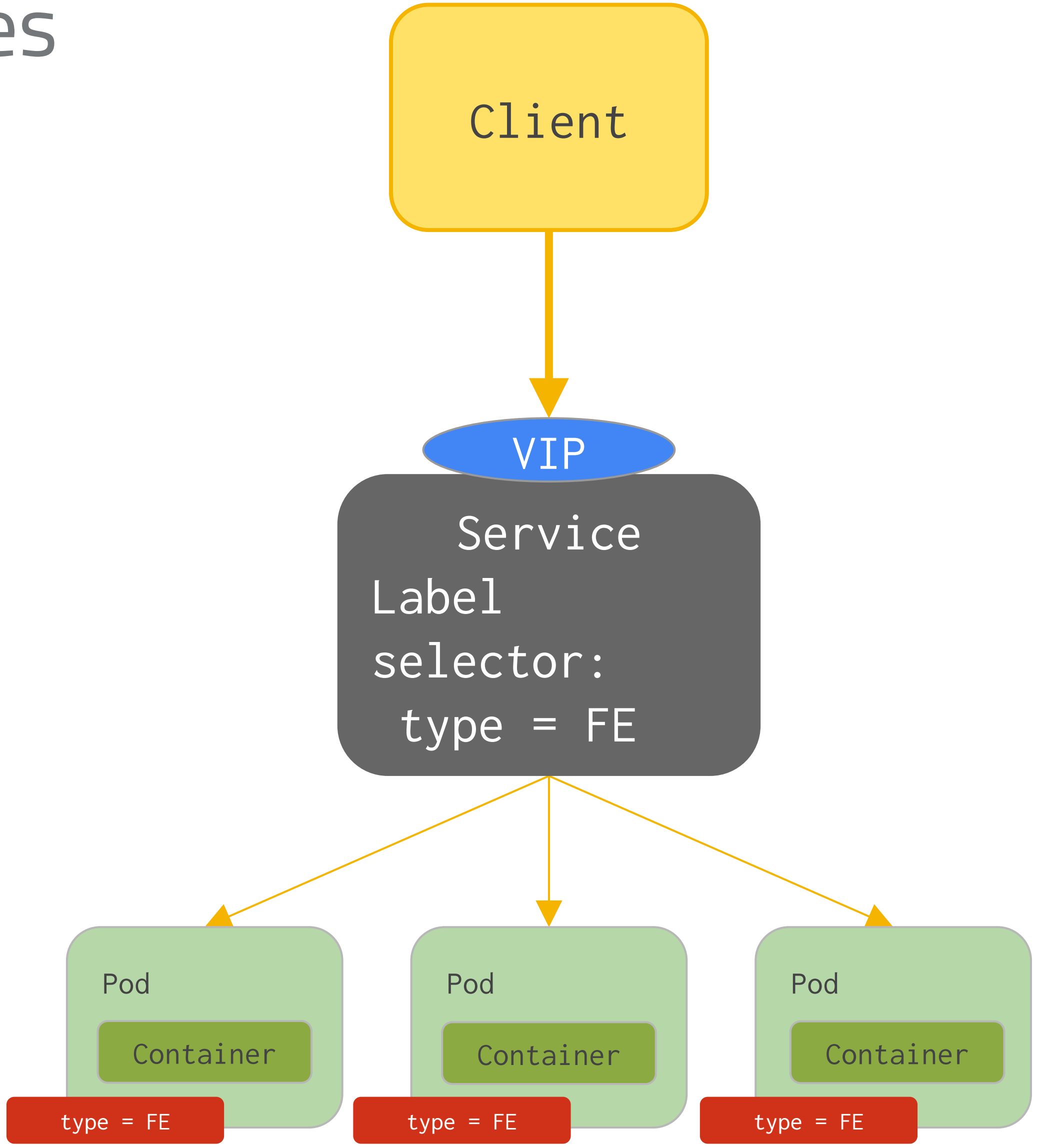
- grouped by label selector

Load balances incoming requests across constituent pods

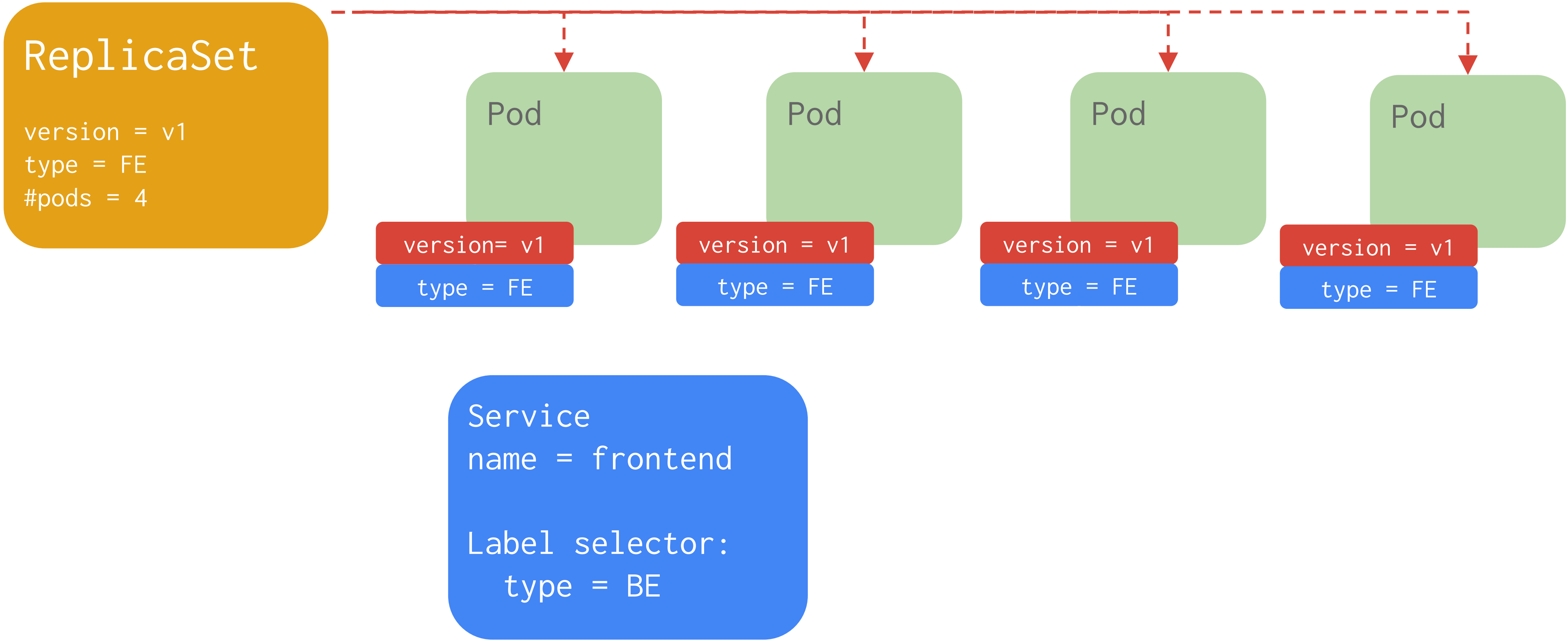
Choice of pod is random but supports session affinity (ClientIP)

Gets a **stable** virtual IP and port

- also get a DNS name



Scaling Example



ReplicaSet

version = v1
type = FE
#pods = 4

Pod
version= v1
type = FE

Pod
version = v1
type = FE

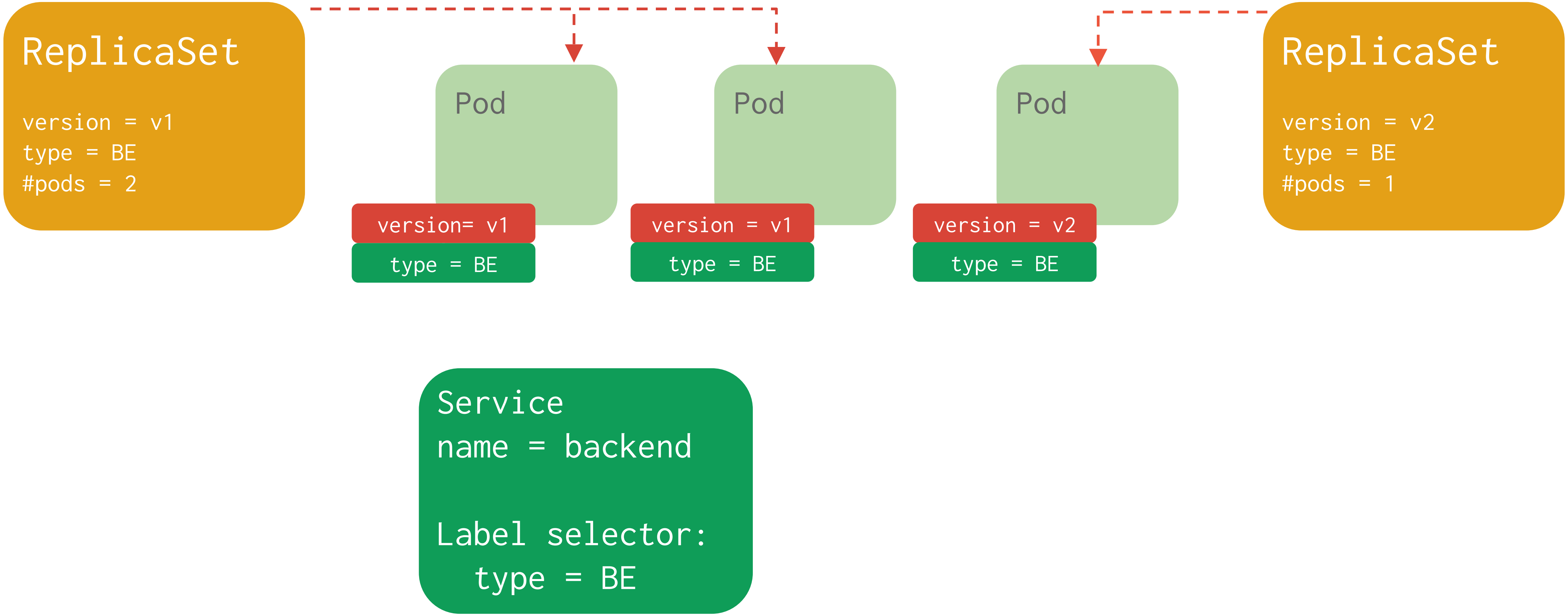
Pod
version = v1
type = FE

Pod
version = v1
type = FE

Service
name = frontend

Label selector:
  type = BE

Canary



ReplicaSet

version = v1

type = BE

#pods = 2

Pod

version= v1

type = BE

Pod

version = v1

type = BE

Pod

version = v2

type = BE

ReplicaSet

version = v2

type = BE

#pods = 1

Service

name = backend

Label selector:

type = BE

loodse

Reliable Deployments

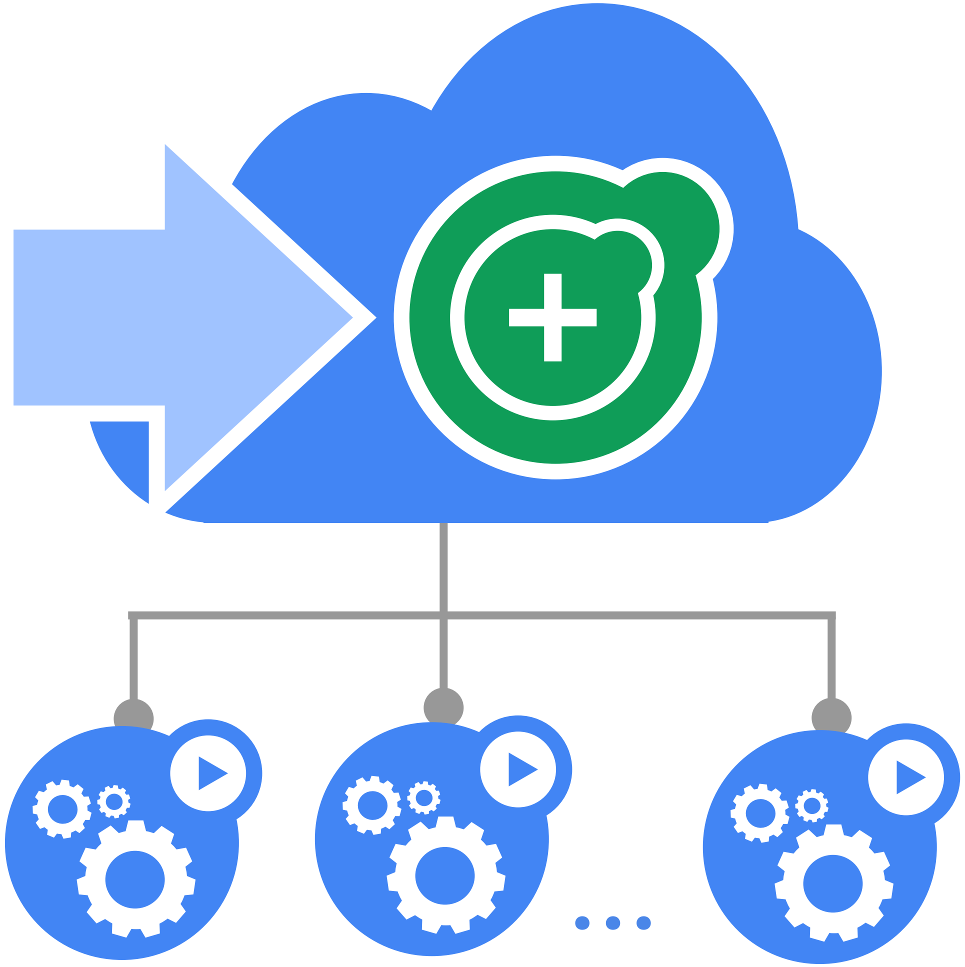
Deployments: Updates as a Service

Reliable mechanism for creating, updating and managing Pods

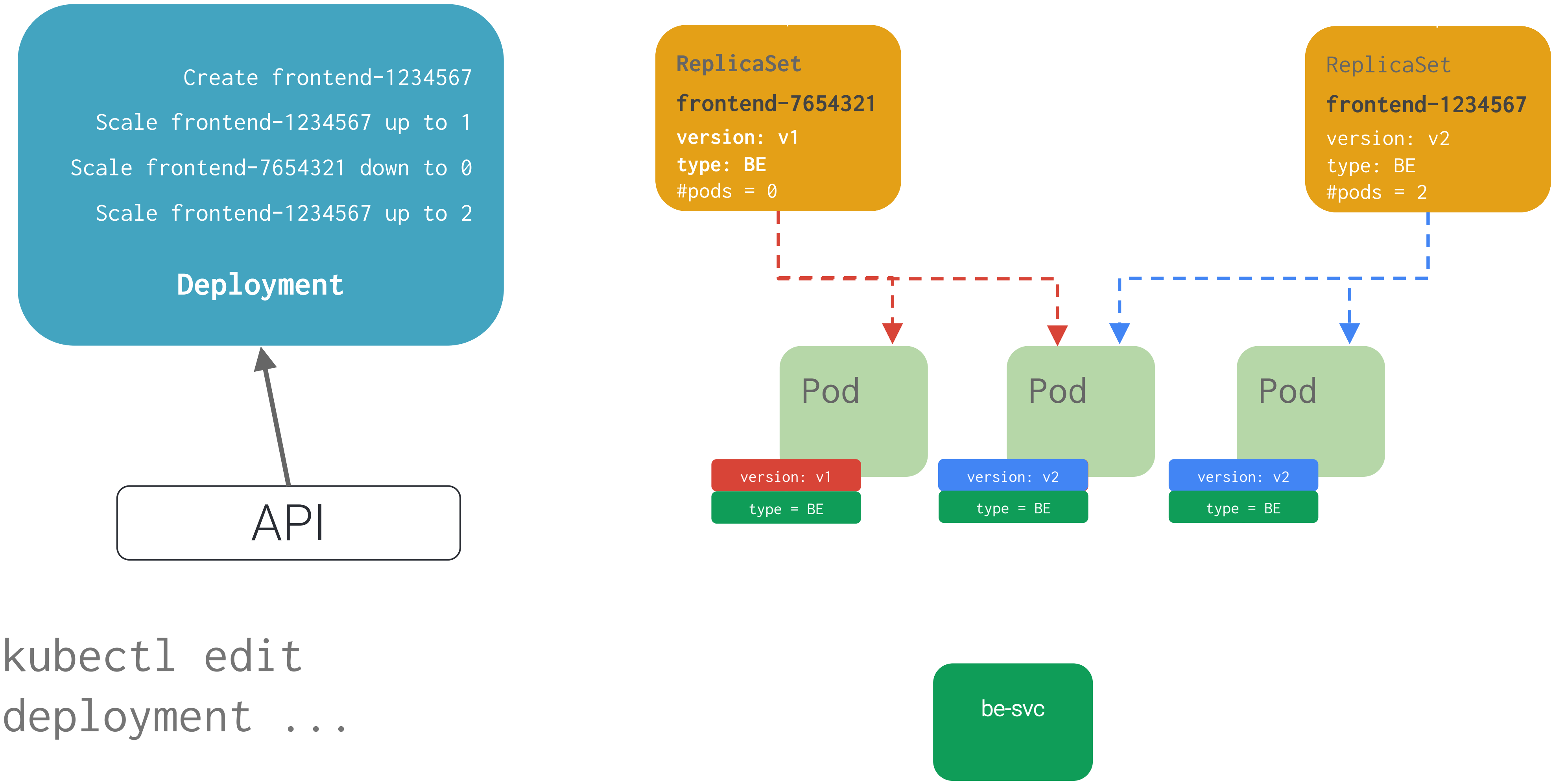
Deployment manages replica changes, including rolling updates and scaling

Edit Deployment configurations in place with kubectl edit or kubectl apply

Managed rollouts and rollbacks



Rollout



Create frontend-1234567

Scale frontend-1234567 up to 1

Scale frontend-7654321 down to 0

Scale frontend-1234567 up to 2

Deployment

API

ReplicaSet

frontend-7654321

version: v1

type: BE

#pods = 0

ReplicaSet

frontend-1234567

version: v2

type: BE

#pods = 2

Pod

version: v1

type = BE

Pod

version: v2

type = BE

Pod

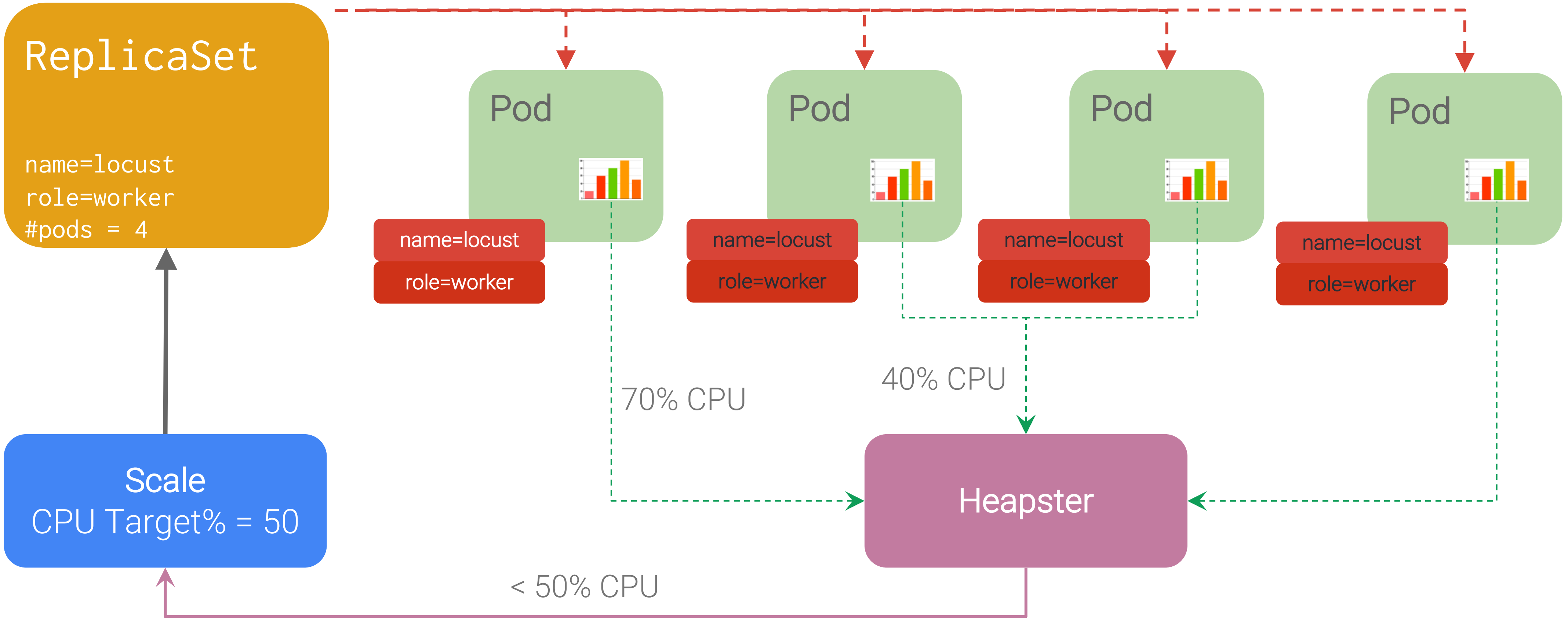
version: v2

type = BE

be-svc

```
kubectl edit
deployment ...
```

Pod Horizontal Autoscaling



loodse

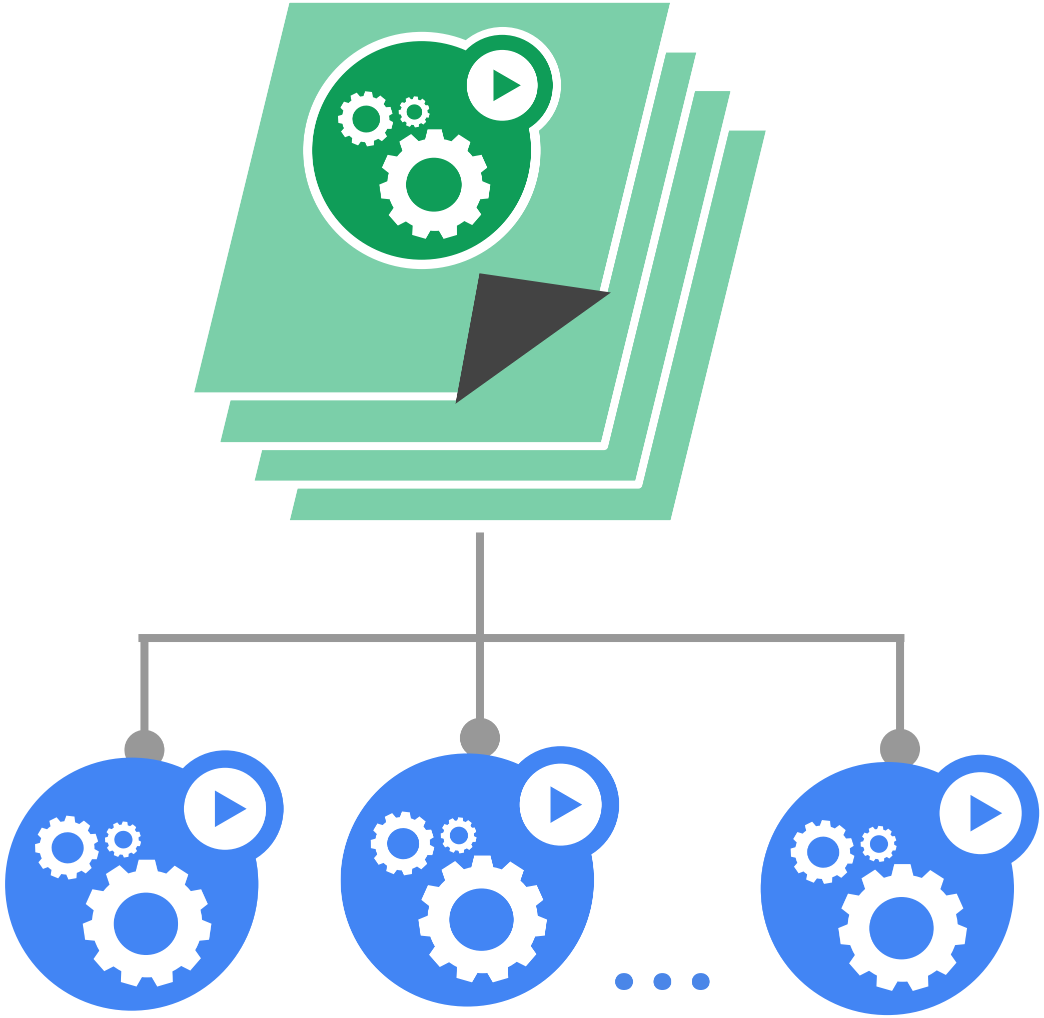
Jobs

Run-to-completion, as opposed to run-forever

- Express parallelism vs. required completions
- Workflow: restart on failure
- Build/test: don’t restart on failure

Aggregates success/failure counts

Built for batch and big-data work



Graceful Termination

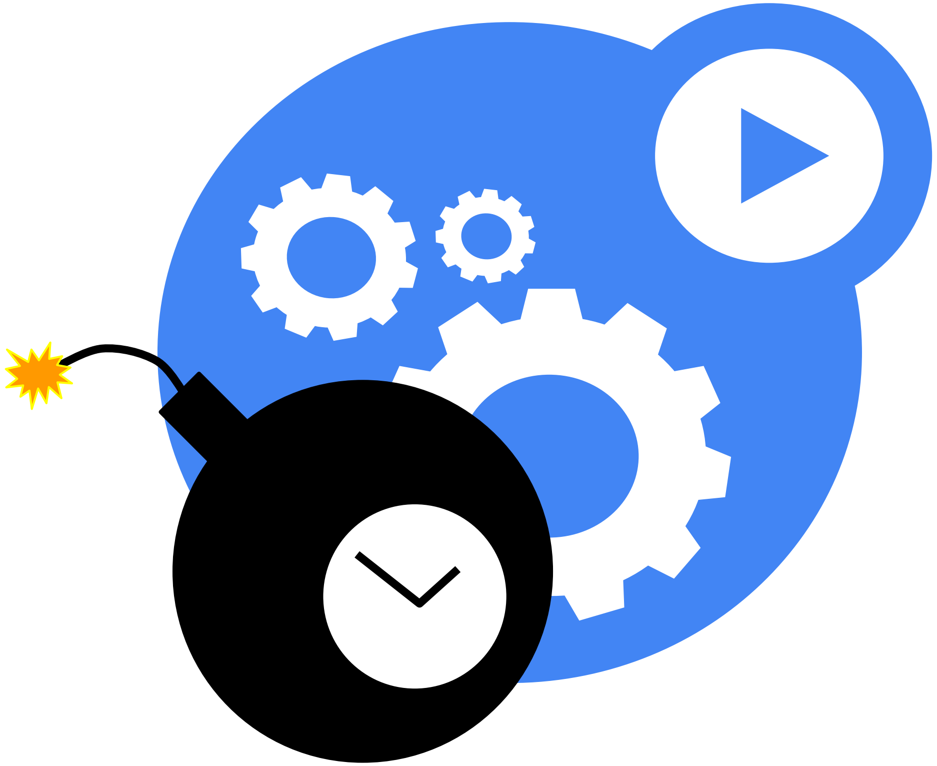
Goal: Give pods time to clean up

- finish in-flight operations
- log state
- flush to disk
- 30 seconds by default

Catch **SIGTERM**, cleanup, exit ASAP

Pod status “Terminating”

Declarative: ‘DELETE’ appears as an object field in the API



Multi-Zone Clusters

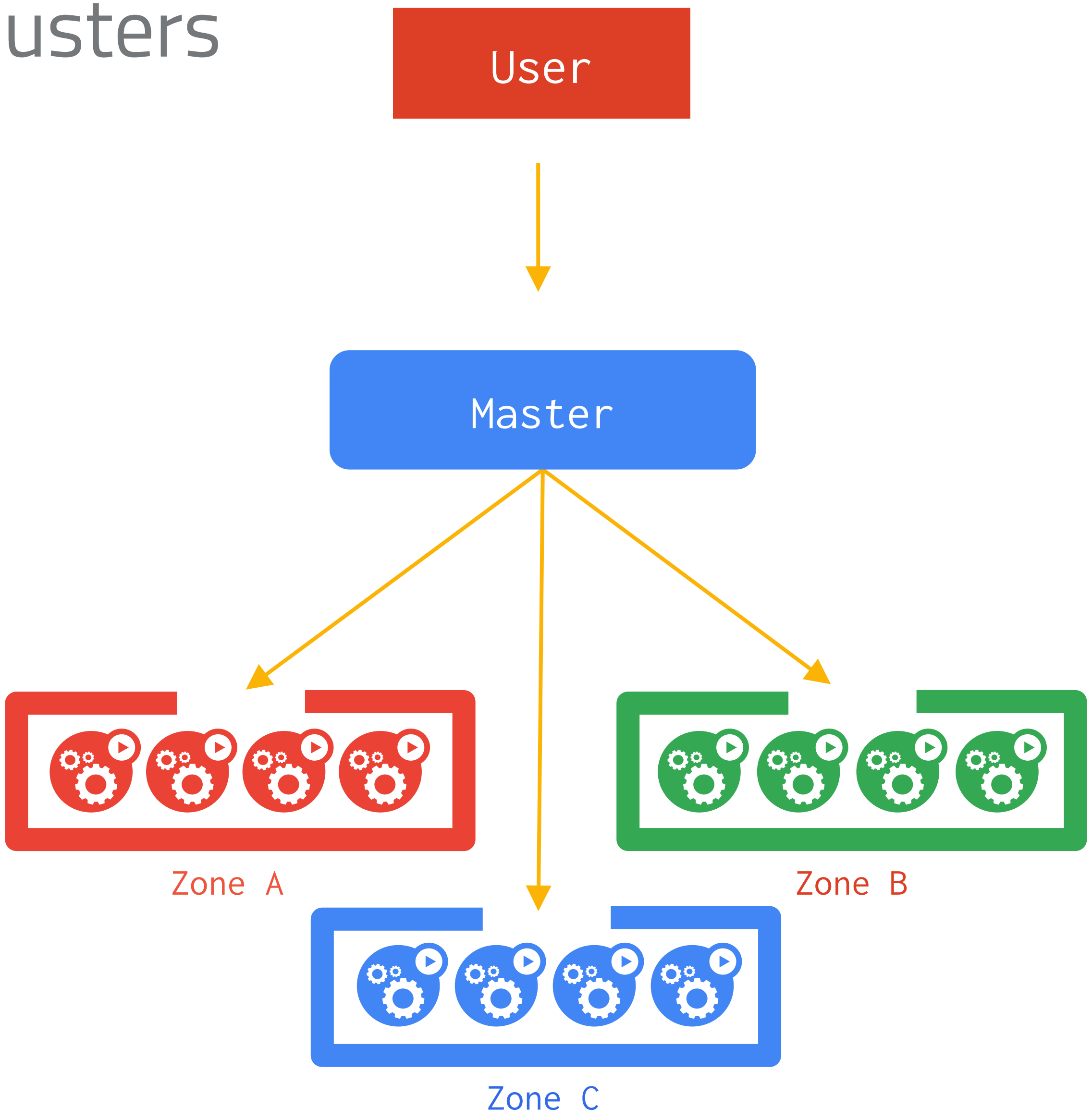
Goal: zone-fault tolerance for applications

Zero API changes relative to kubernetes

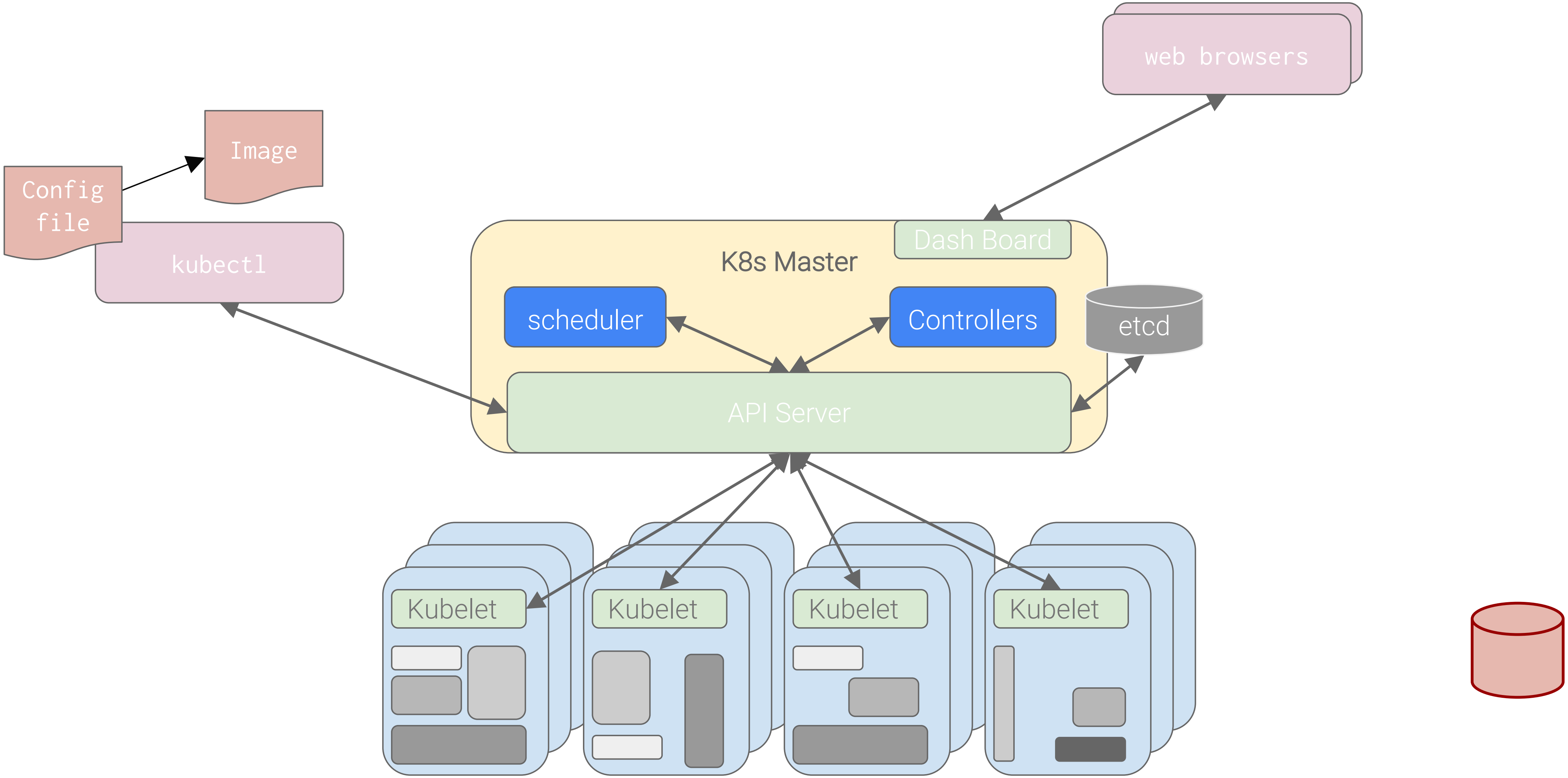
- Create services, replication controllers, etc. exactly as usual

Nodes and PersistentVolumes are labelled with their availability zone

- Fully automatic for GKE, GCE, AWS
- Manual for on-premise and other cloud providers (for now)

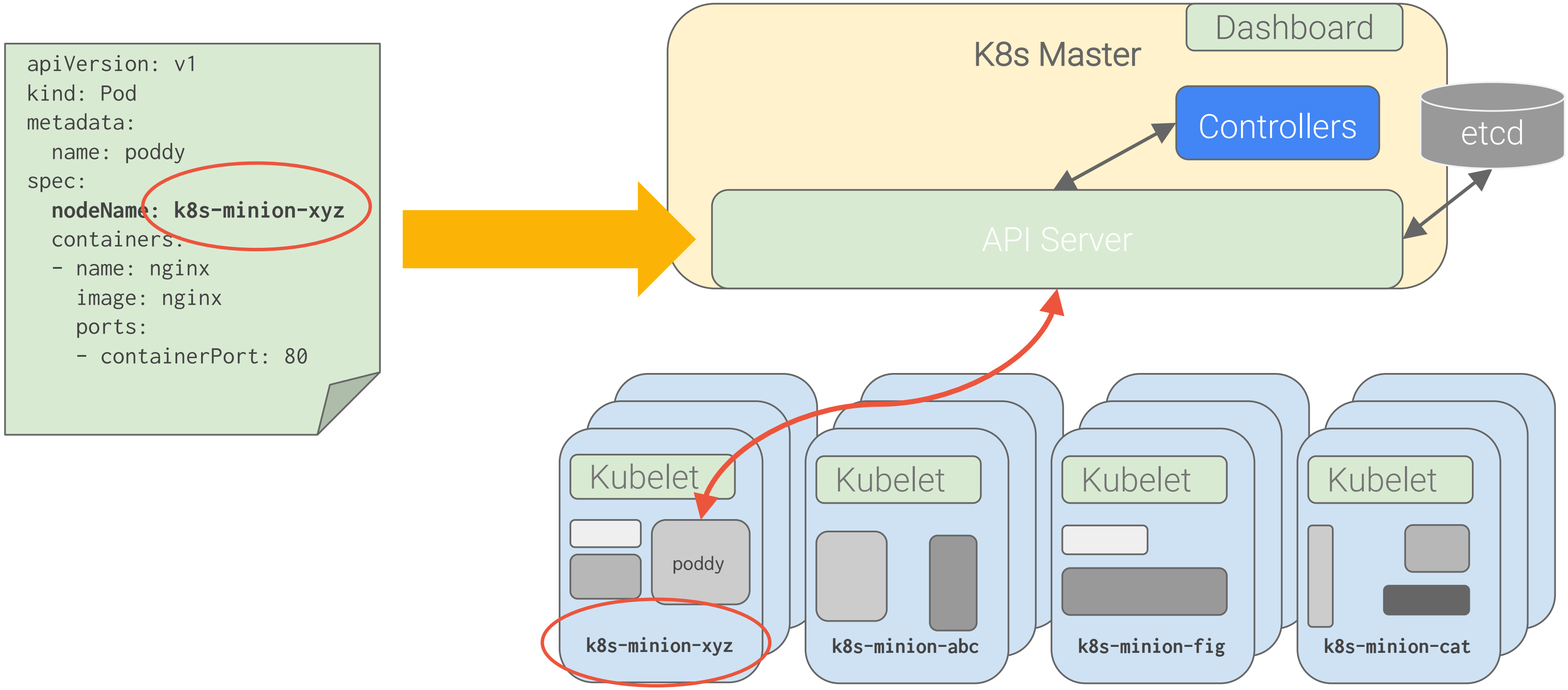


Scheduling


web browsers
Image
Config file
kubectl
Dash Board
K8s Master
scheduler
Controllers
etcd
API Server
Kubelet
Kubelet
Kubelet
Kubelet

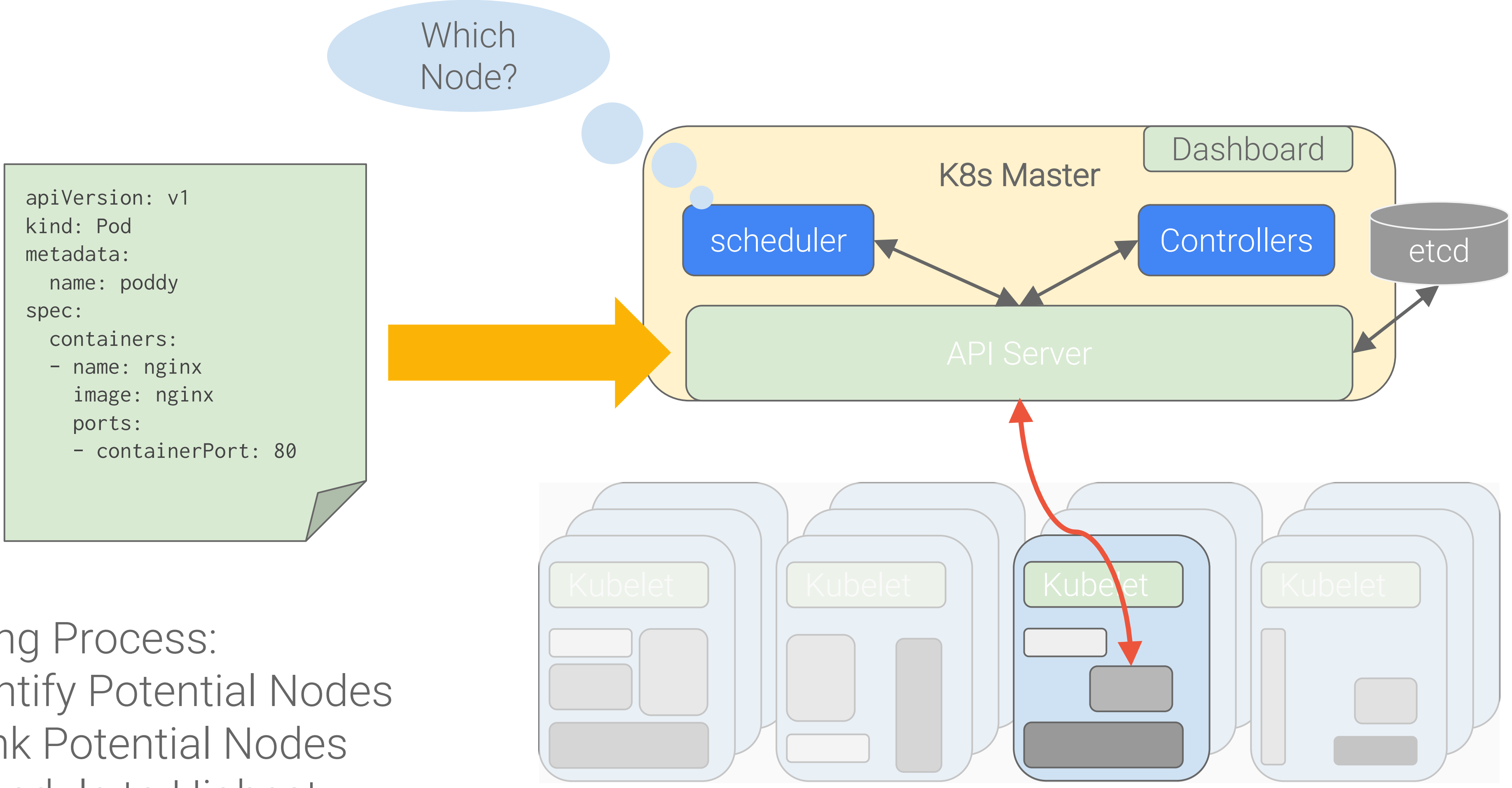
loodse

Kubernetes without a Scheduler



```
apiVersion: v1
kind: Pod
metadata:
  name: poddy
spec:
  nodeName: k8s-minion-xyz
  containers:
  - name: nginx
    image: nginx
    ports:
    - containerPort: 80
```

Kubernetes with a Scheduler



```yaml
apiVersion: v1
kind: Pod
metadata:
  name: poddy
spec:
  containers:
  - name: nginx
    image: nginx
    ports:
    - containerPort: 80
```

Scheduling Process:

- Identify Potential Nodes
- Rank Potential Nodes
- Schedule to Highest Ranked Node

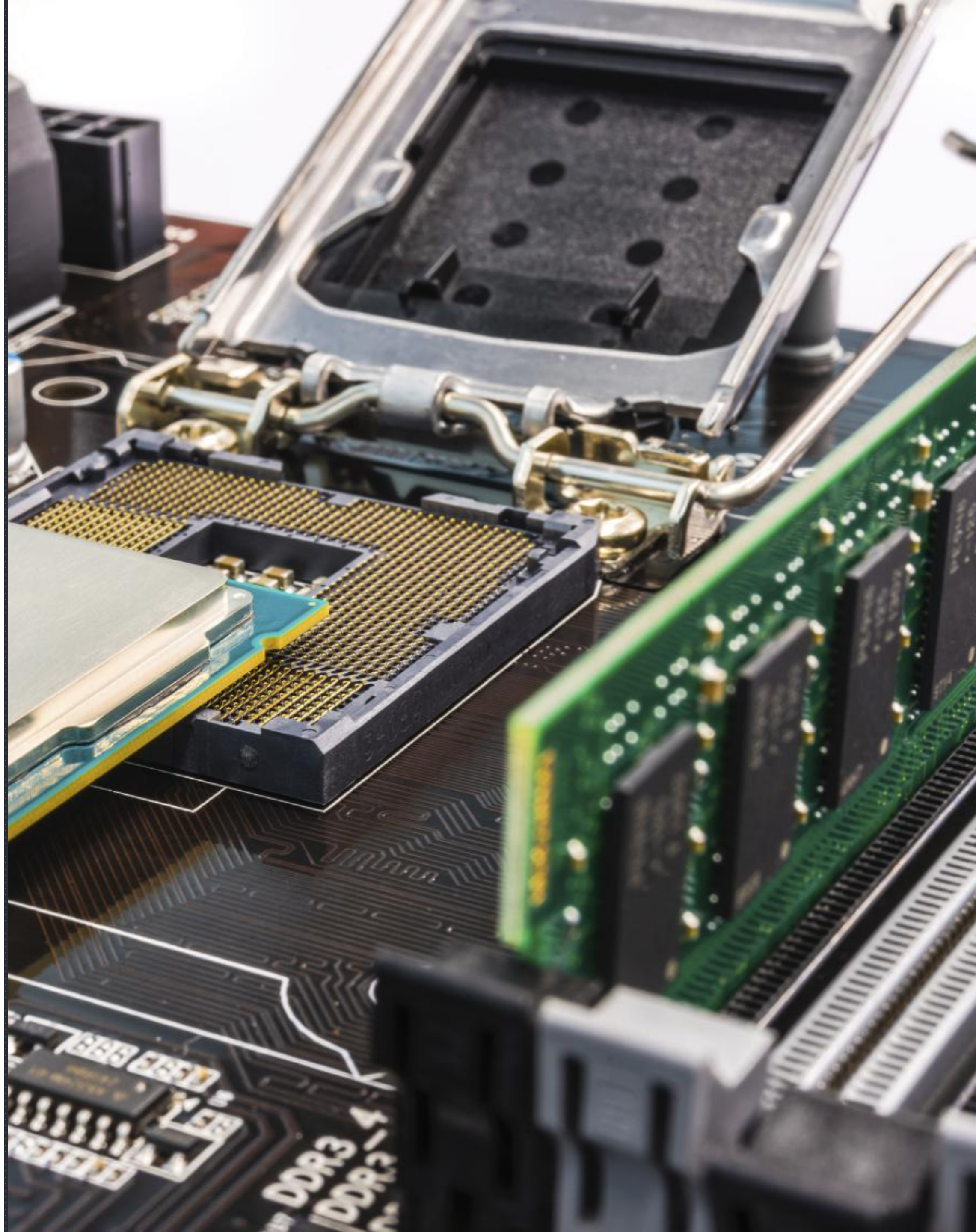
Kubernetes Resources

A Resource is something that can be **requested**, **allocated**, or **consumed** to/by a pod or a container

> **CPU**: Specified in units of Cores, what that is depends on the provider
>
> **Memory**: Specified in units of Bytes

CPU is **Compressible** (i.e. it has a *rate* and can be throttled)

Memory is **Incompressible**, it can't be throttled



Requests and Limits

Request:

- how much of a resource you are asking to use, with a strong guarantee of availability
- scheduler will not over-commit requests

Limit:

- max amount of a resource you can access

Conclusion:

- Usage > Request: resources **might** be available
- Usage > Limit: throttled or killed

Resource based Scheduling

Provide QoS for Scheduled Pods

Per Container CPU and Memory requirements

Specified as **Request** and **Limit**

- **Best Effort** (Request == 0)
- **Burstable** (Request < Limit)
- **Guaranteed** (Request == Limit)

Best Effort Scheduling for low priority workloads improves Utilization at Google by 20%

Resource based Scheduling

my-controller.yaml

```yaml
...
spec:
  containers:
    - name: locust
      image: gcr.io/rabbit-skateboard/guestbook:gdg-rtv
      resources:
        requests:
          memory: "300Mi"
          cpu: "100m"
        limits:
          memory: "300Mi"
          cpu: "100m"
```

CPU Resource: Requests vs Limits

For a Pod to be *scheduled* the amount of CPU it **Requests** must be available on a single node

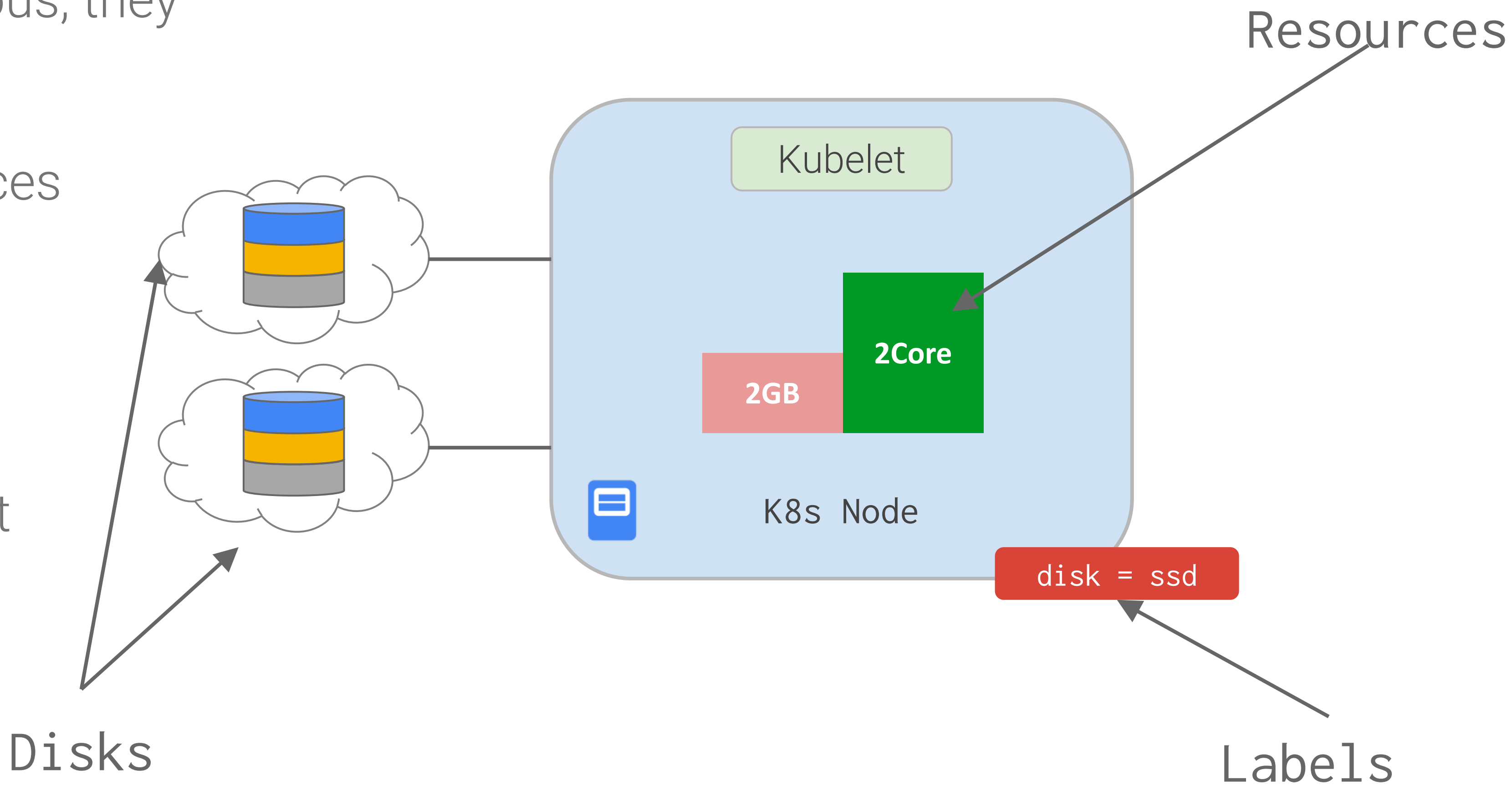
If it **Requests** 0 CPU it can always be scheduled

Scheduling Pods: Nodes

Nodes may not be heterogeneous, they can differ in important ways:

- CPU and Memory Resources
- Attached Disks
- Specific Hardware

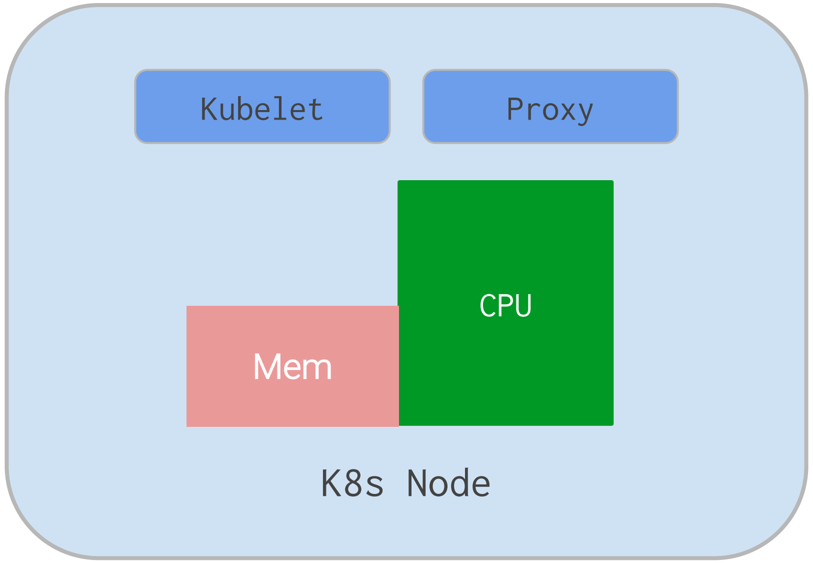
Location may also be important



Pod Scheduling: Identifying Potential Nodes

What **CPU and Memory Resources** does it need?

Can also be used as a measure of priority

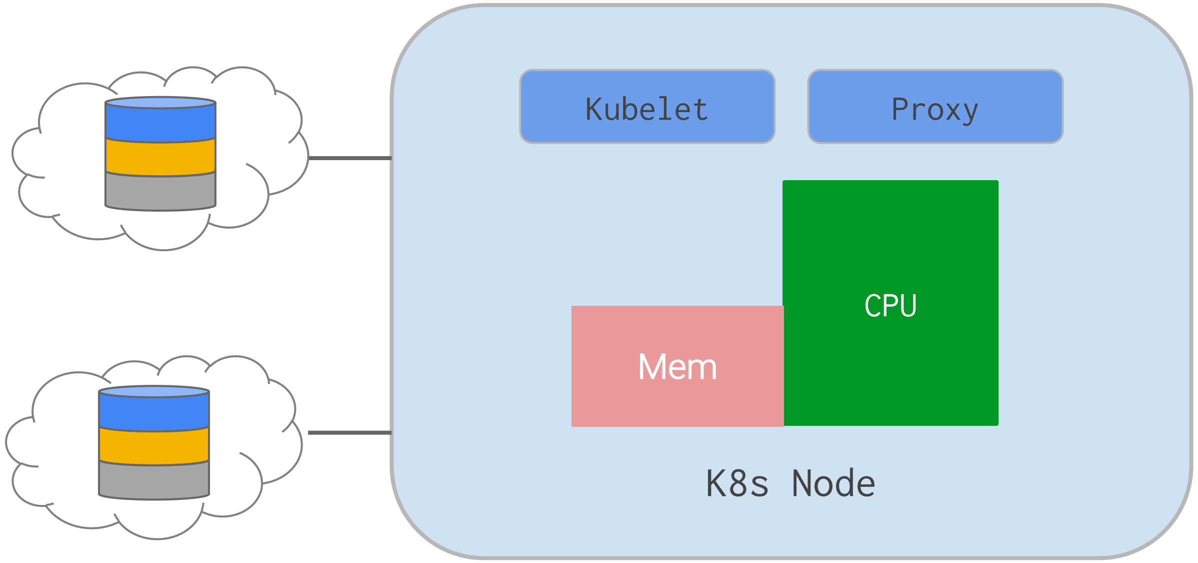


Pod Scheduling: Finding Potential Nodes

What **Resources** does it need?

What **Disk(s)** does it need (GCE PD and EBS) and can it/they be mounted without conflict?

Note: 1.1 limits to a single volume mount per node



Pod Scheduling: Identifying Potential Nodes

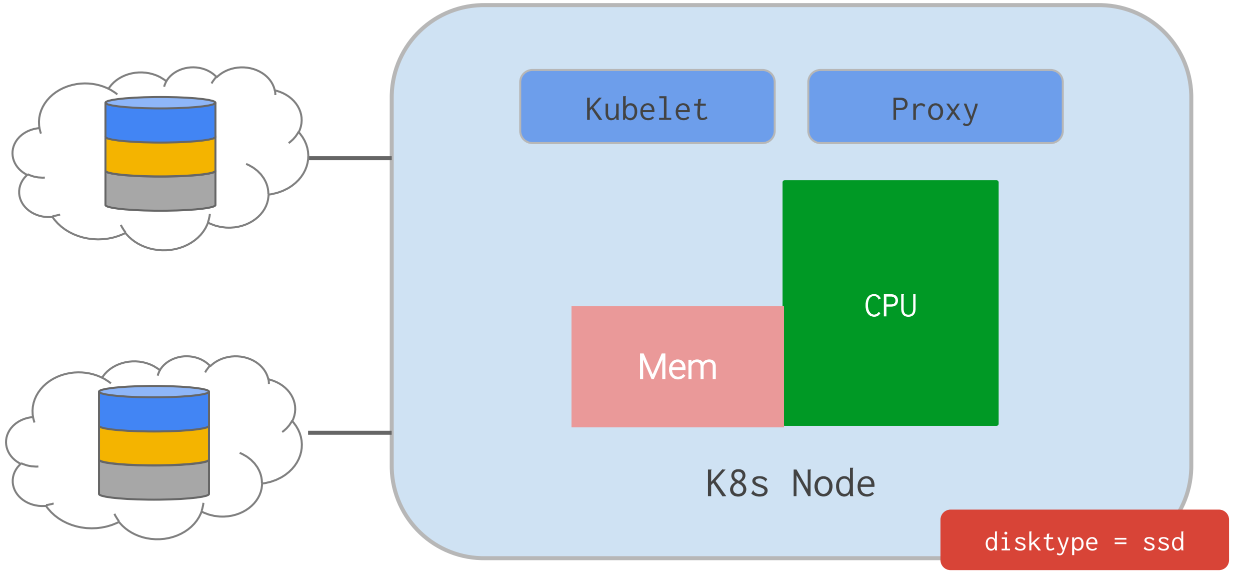
What **Resources** does it need?

What **Disk(s)** does it need?

What node(s) can it run on (**Node Selector**)?

```
kubectl label nodes node-3
disktype=ssd
```

```
(pod) spec:
    nodeSelector:
      disktype: ssd
```



loodse

nodeAffinity

Implemented through Annotations in 1.5,

Can be ‘Required’ or ‘Preferred’ during scheduling

In future can can be ‘Required’ during execution (Node labels can change)

Will eventually replace NodeSelector

If you specify both nodeSelector and nodeAffinity, both must be satisfied

```
{
   "nodeAffinity": {
     "requiredDuringSchedulingIgnoredDuringExecution": {
       "nodeSelectorTerms": [
         {
           "matchExpressions": [
             {
               "key": "beta.kubernetes.io/instance-type",
               "operator": "In",
               "values": ["n1-highmem-2", "n1-highmem-4"]
             }
           ]
         }
       ]
     }
   }
 }
```

http://kubernetes.github.io/docs/user-guide/node-selection/

Pod Scheduling: Ranking Potential Nodes

Prefer node with most free resource left after the pod is deployed

Prefer nodes with the specified label

Minimise number of Pods from the same service on the same node

CPU and Memory is balanced after the Pod is deployed **[Default]**

Node3
Node2
Node1
1
2
3

Extending the Scheduler

1. Add rules to the scheduler and recompile
2. Run your own scheduler process instead of, or as well as, the Kubernetes scheduler
3. Implement a "scheduler extender" that the Kubernetes scheduler calls out to as a final pass when making scheduling decisions

Admission Control

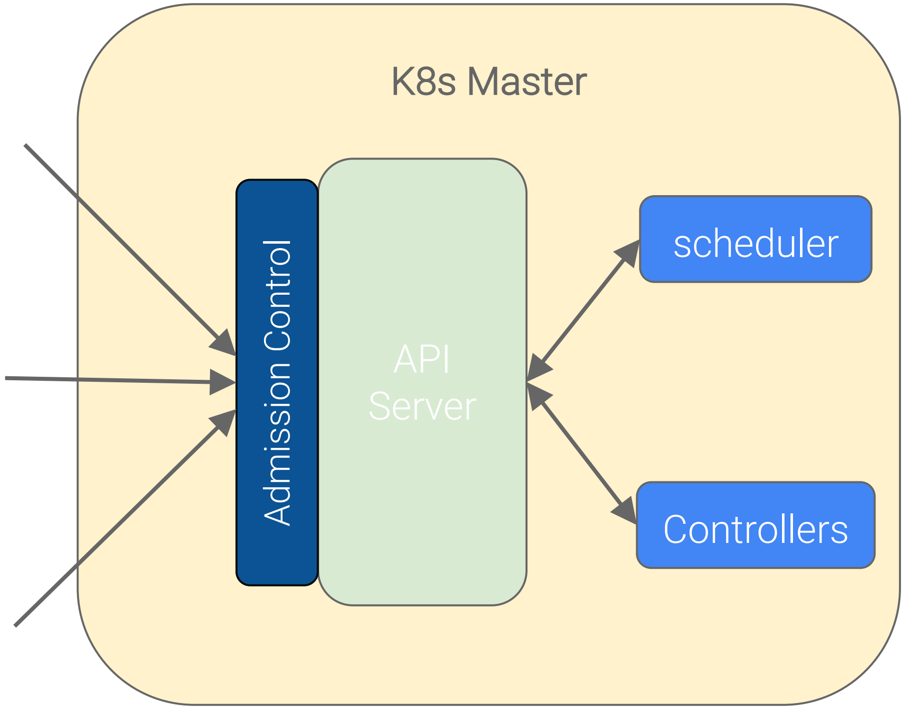
Admission Control (AC) enforces certain conditions, before a request is accepted by the API Server

AC functionality implemented as plugins which are executed in the sequence they are specified

AC is performed after AuthN (AuthenticatioN) checks

Enforcement usually results in either

- A Request denial
- Mutation of the Request Resource
- Mutation of related Resources



Admission Control Examples

NamespaceLifecycle

Enforces that a Namespace that is undergoing termination cannot have new objects created in it, and ensures that requests in a non-existant Namespace are rejected

LimitRanger

Observes the incoming request and ensures that it does not violate any of the constraints enumerated in the LimitRange object in a Namespace

ServiceAccount

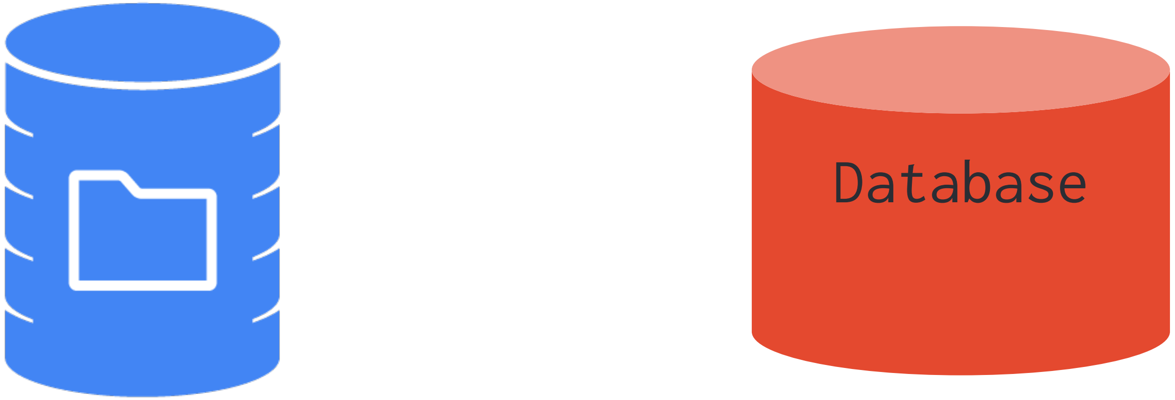
Implements automation for serviceAccounts (RBAC)

ResourceQuota

Observes the incoming request and ensures that it does not violate any of the constraints enumerated in the ResourceQuota object in a Namespace.

Managing State

I still have questions about state!

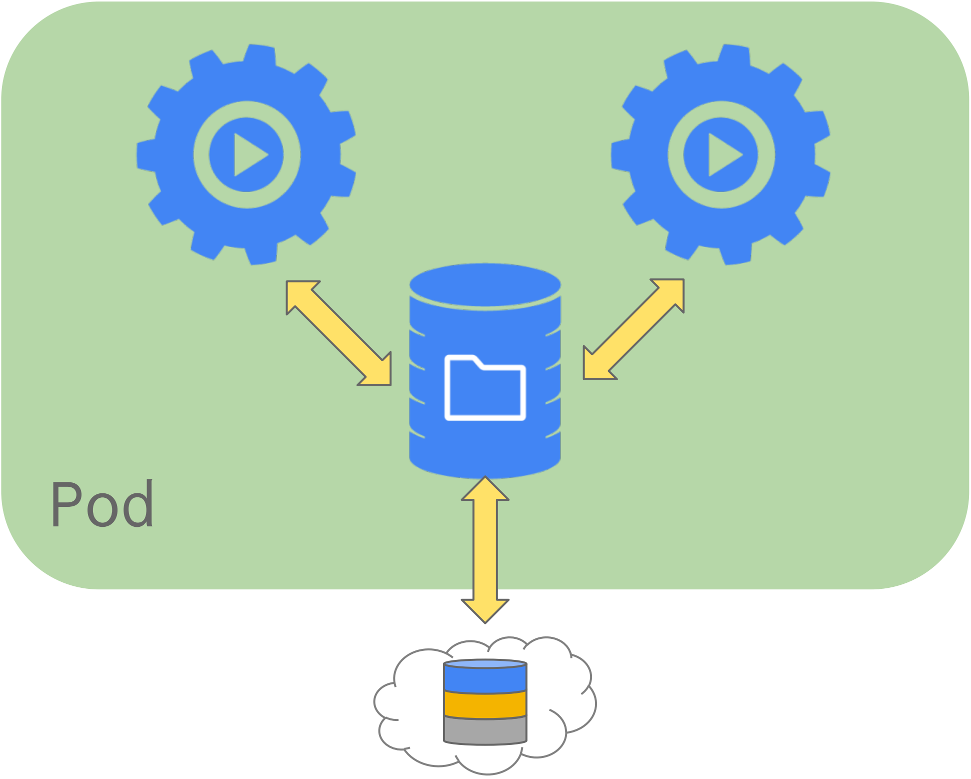


In a cluster of ephemeral containers
Application state must exist outside of the container

Volumes

Bound to the Pod that encloses it
Look like Directories to Containers
What and where they are determined by Volume Type
Many Volume options

- EmptyDir
- HostPath
- nfs (and similar services)
- Cloud Provider Block Storage



PersistentVolumes

A higher-level storage abstraction

- insulation from any one cloud environment

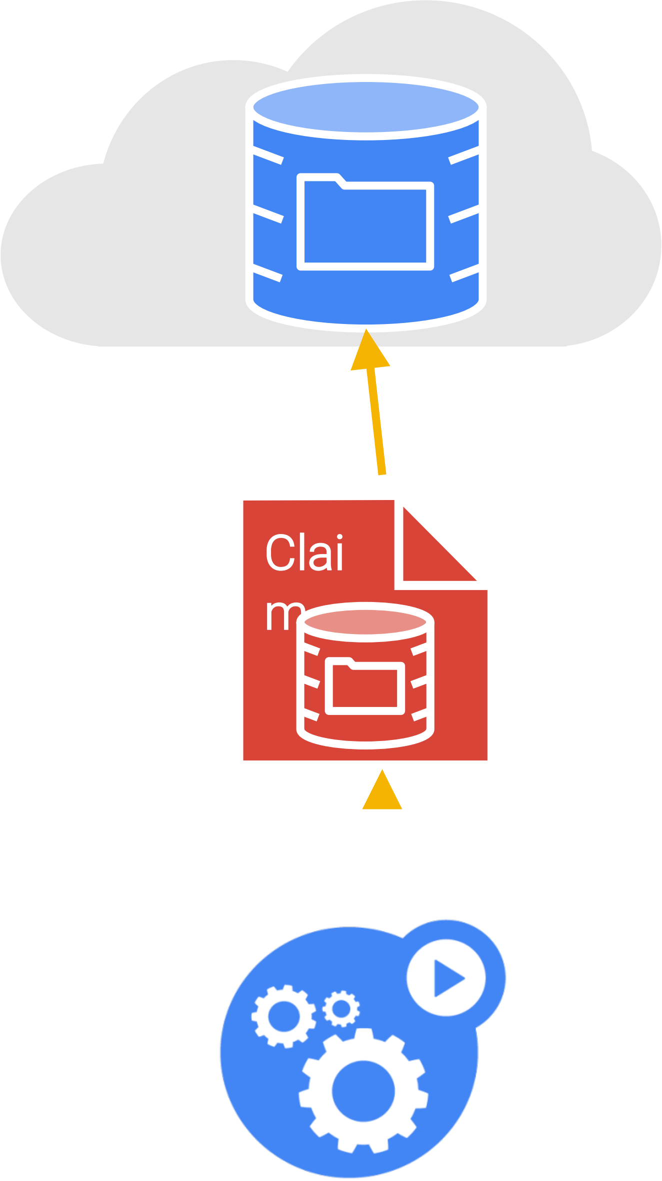
Admin provisions them, users claim them

- auto-provisioning

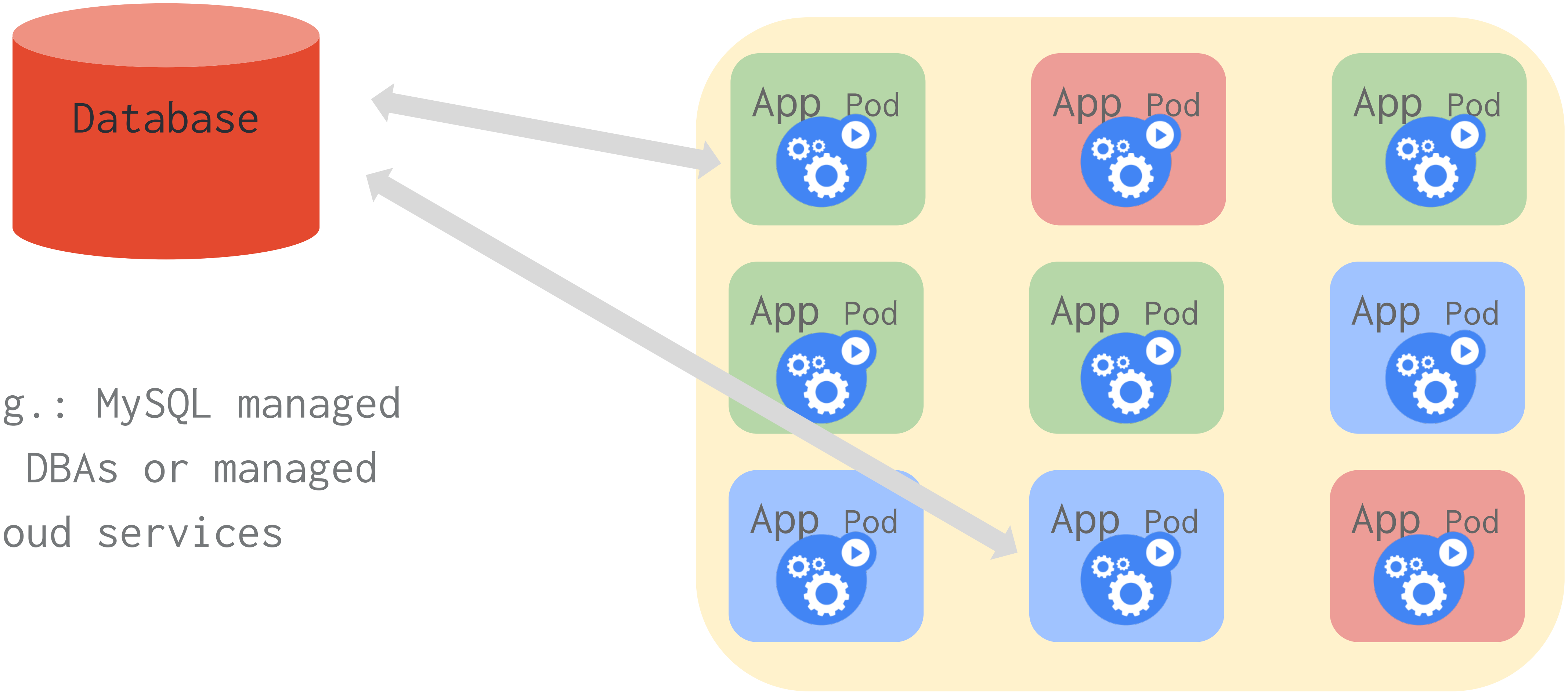
Independent lifetime and fate from consumers

- lives until user is done with it
- can be handed-off between pods

Dynamically “scheduled” and managed, like nodes and pods

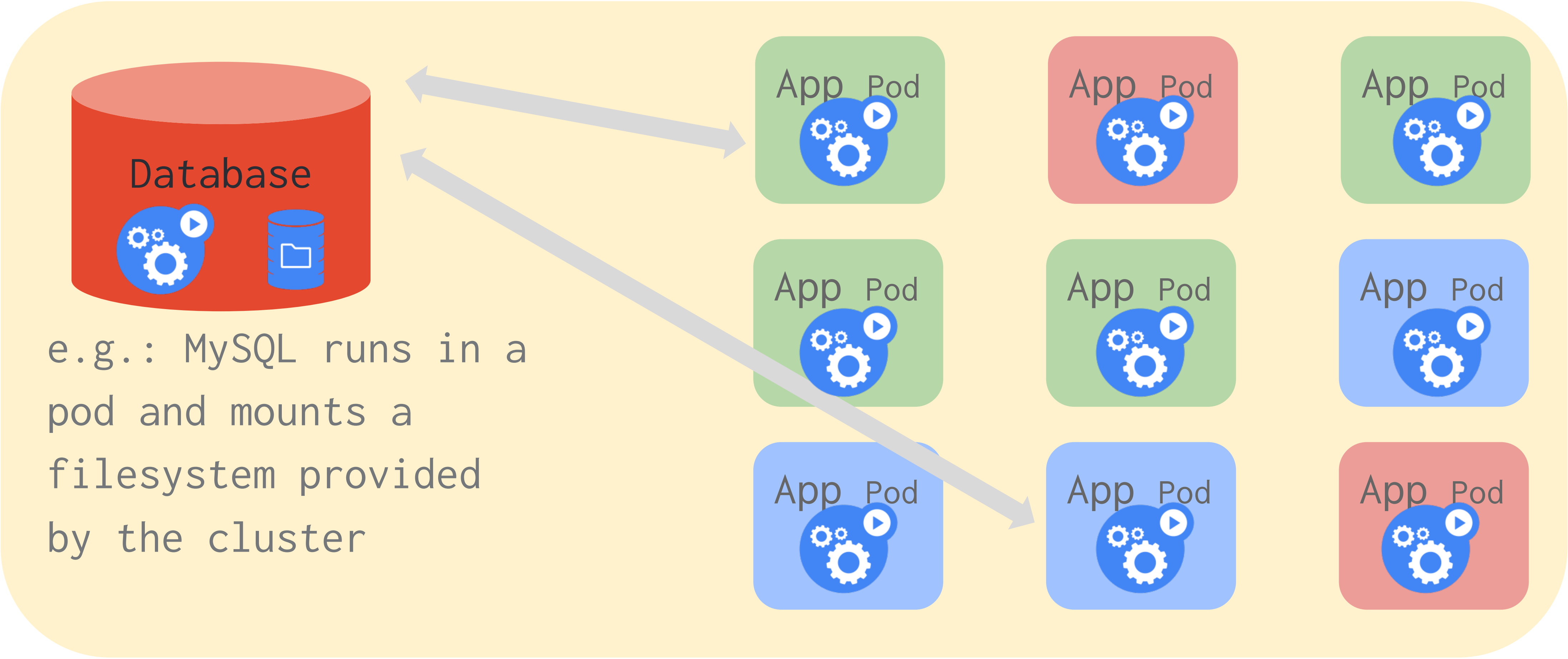


Outside the Cluster



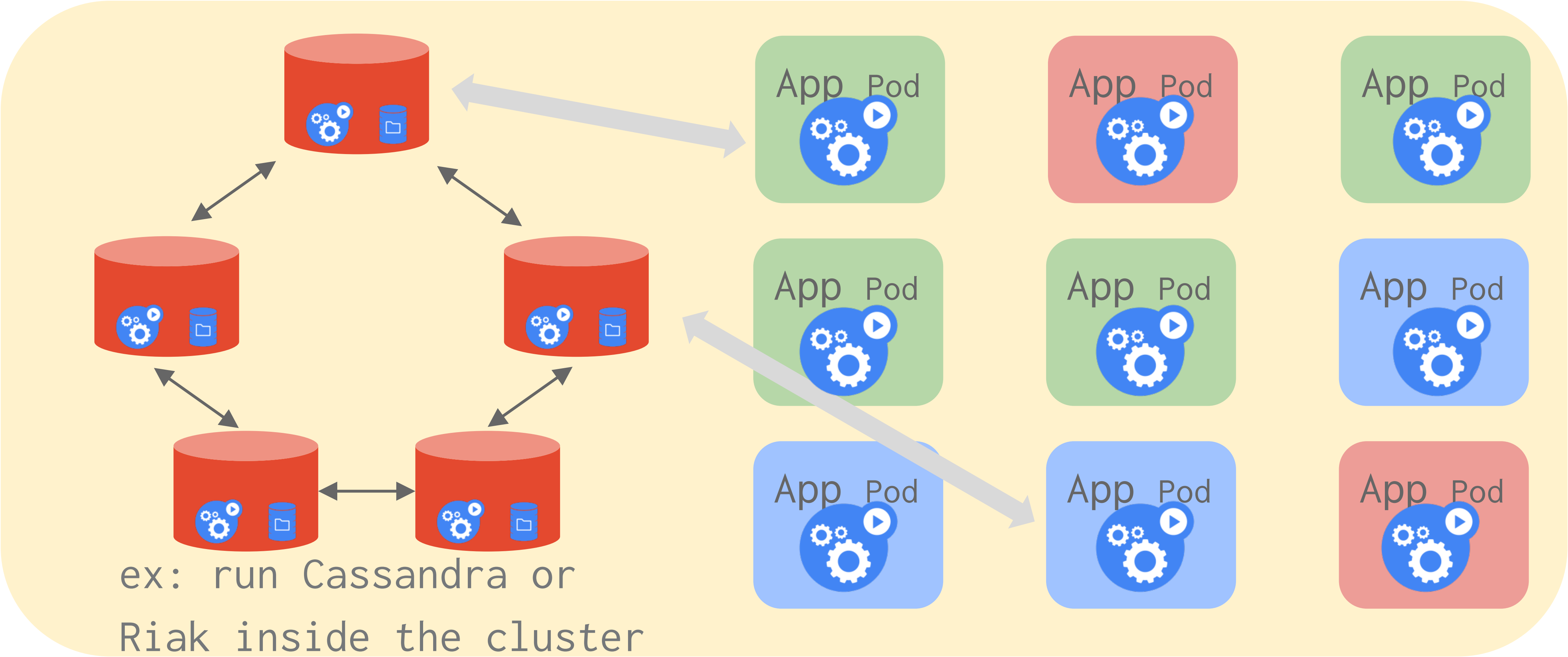
e.g.: MySQL managed by DBAs or managed cloud services

Adapt to run in the Cluster



e.g.: MySQL runs in a pod and mounts a filesystem provided by the cluster

Cluster Native



ex: run Cassandra or Riak inside the cluster

Cluster native - MySQL on Vitess

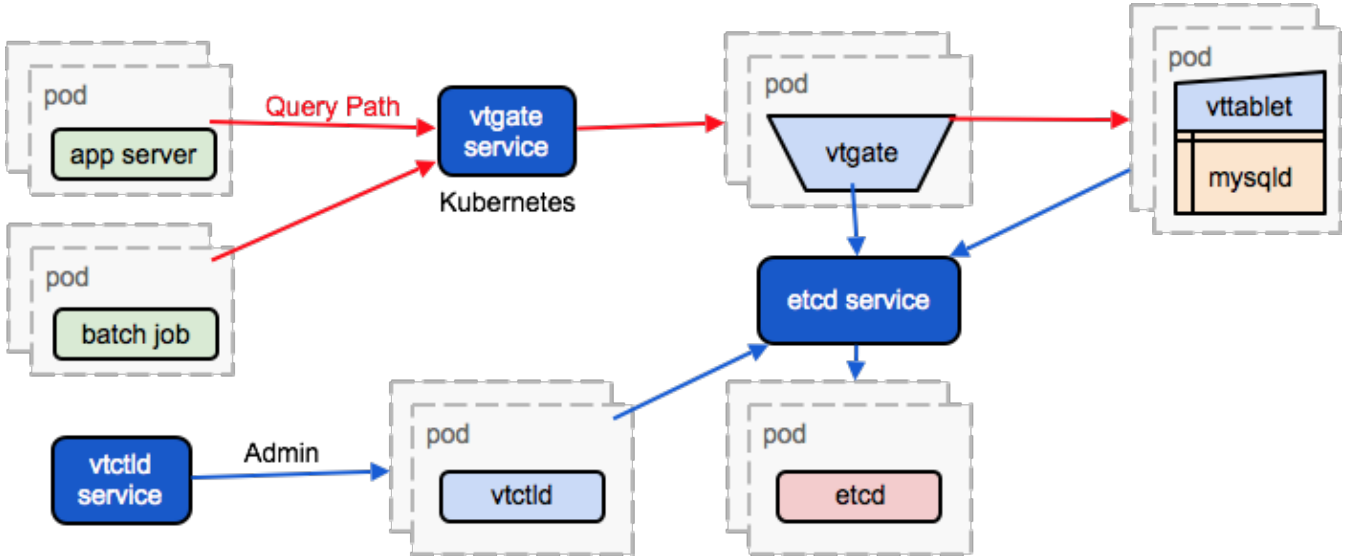
Open source MySQL scaling solution

Vitess has been serving all YouTube database traffic since 2011

Replication, dynamic sharding, caching and more

Designed for a distributed, containerized world

Kubernetes configs included



http://vitess.io/

ConfigMaps

Goal: manage app configuration

- ...without making overly-brittle container images

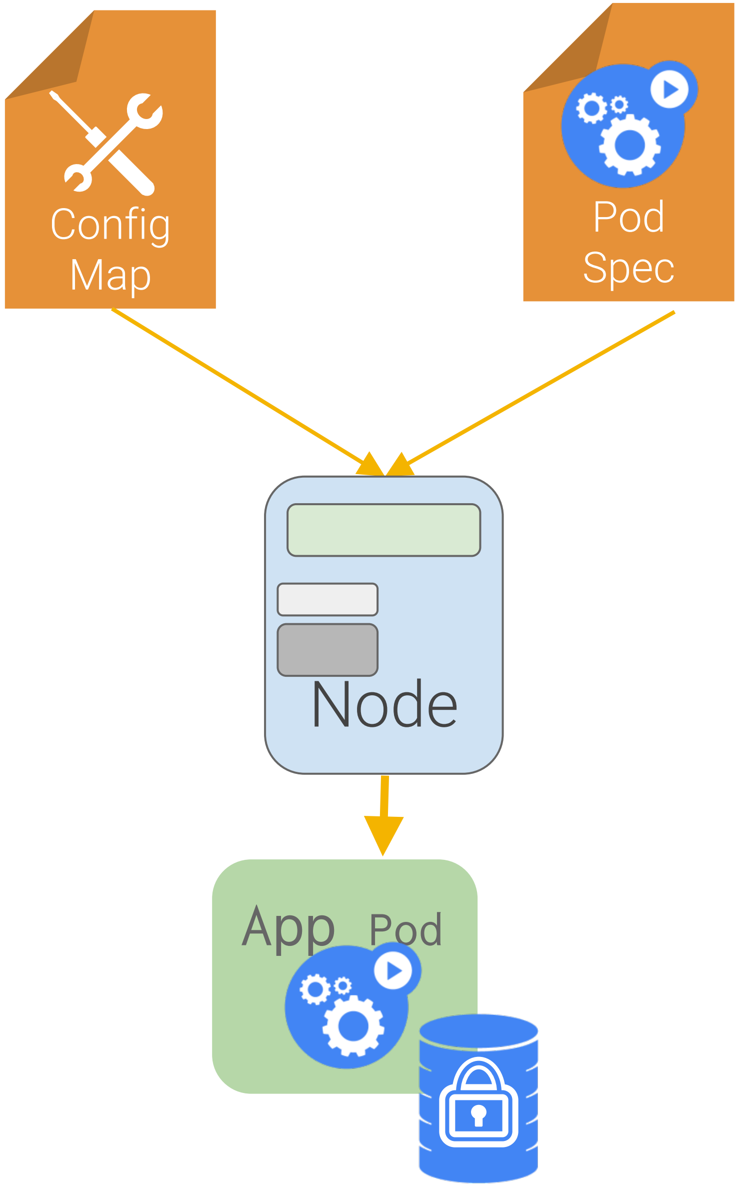
12-factor says config comes from the environment

- Kubernetes is the environment

Manage config via the Kubernetes API

Inject config as a virtual volume into your Pods

- late-binding, live-updated (atomic)
- also available as env vars



Secrets

Goal: grant a pod access to a secured *something*?

- don't put secrets in the container image!

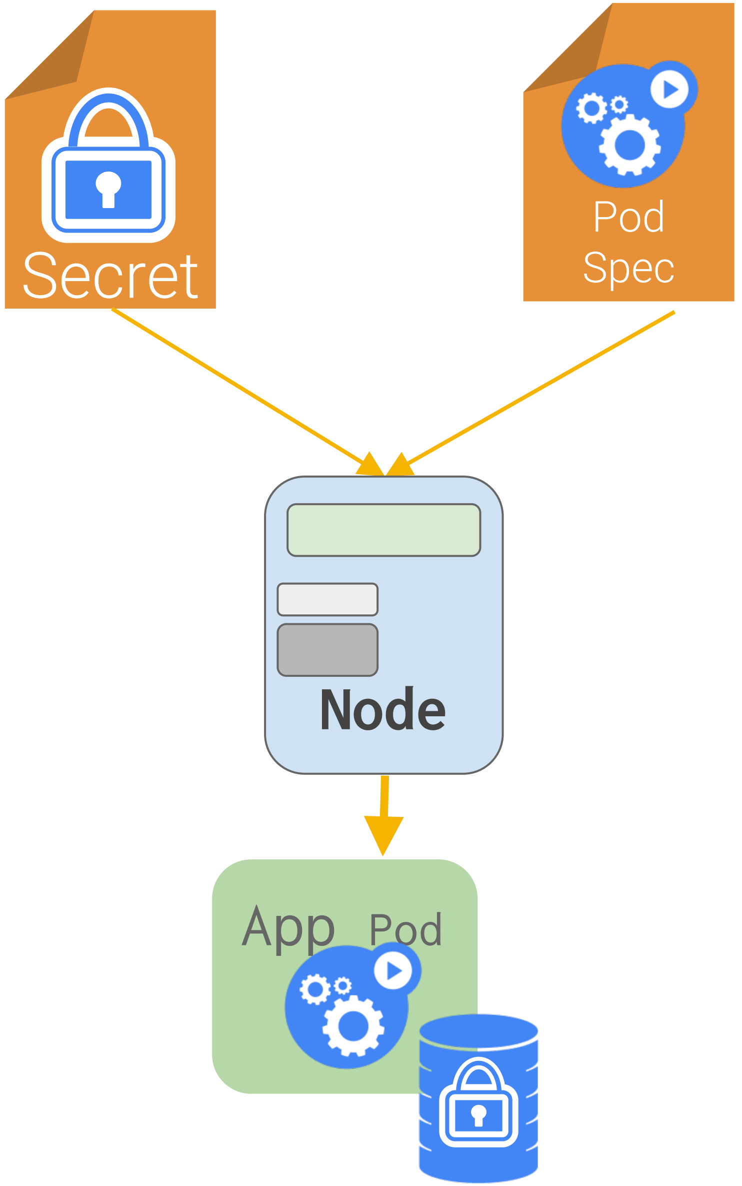
12-factor says: config comes from the **environment**

- Kubernetes is the environment

Manage secrets via the Kubernetes API

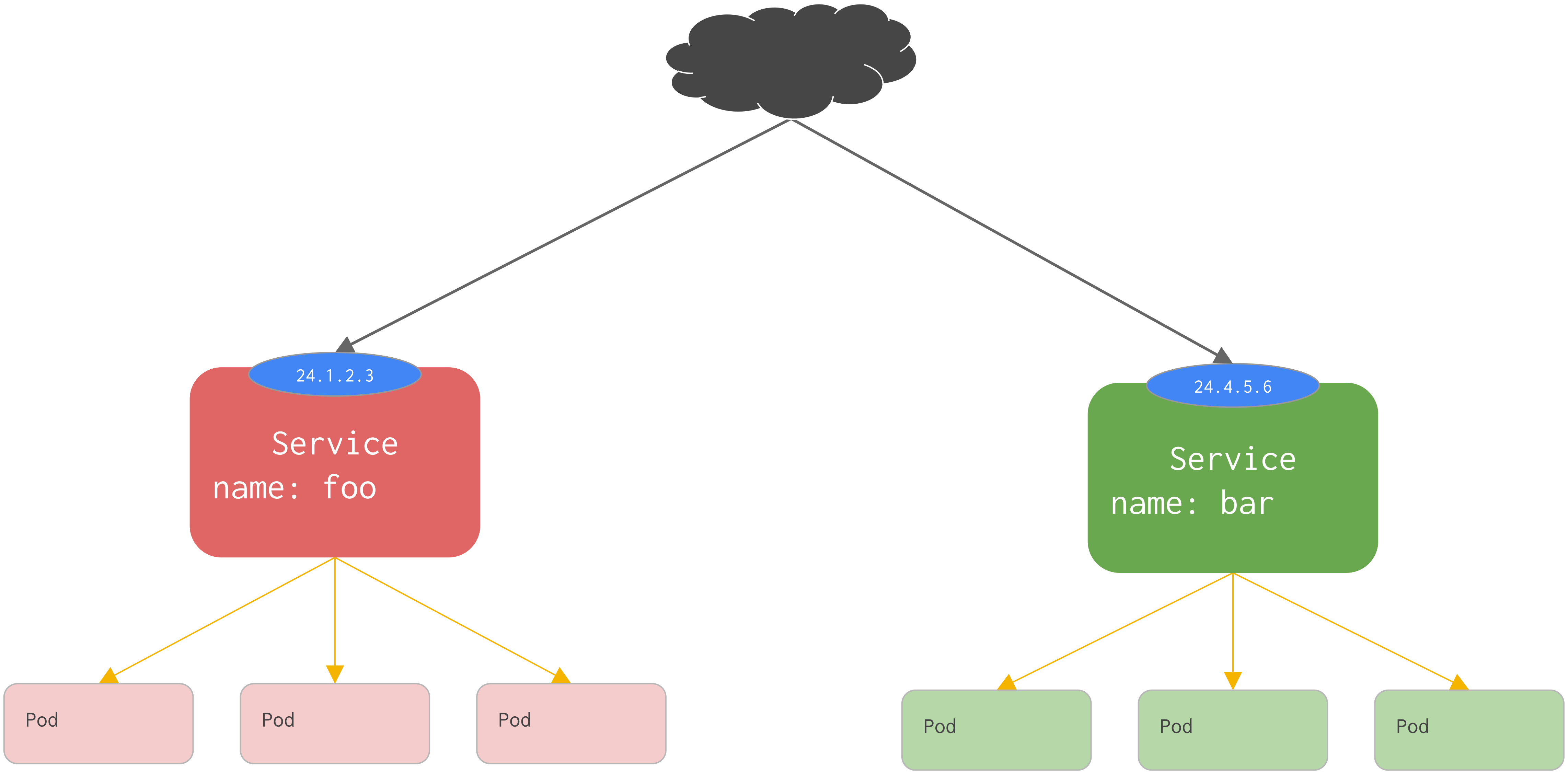
Inject them as virtual volumes into Pods

- late-binding
- tmpfs - never touches disk

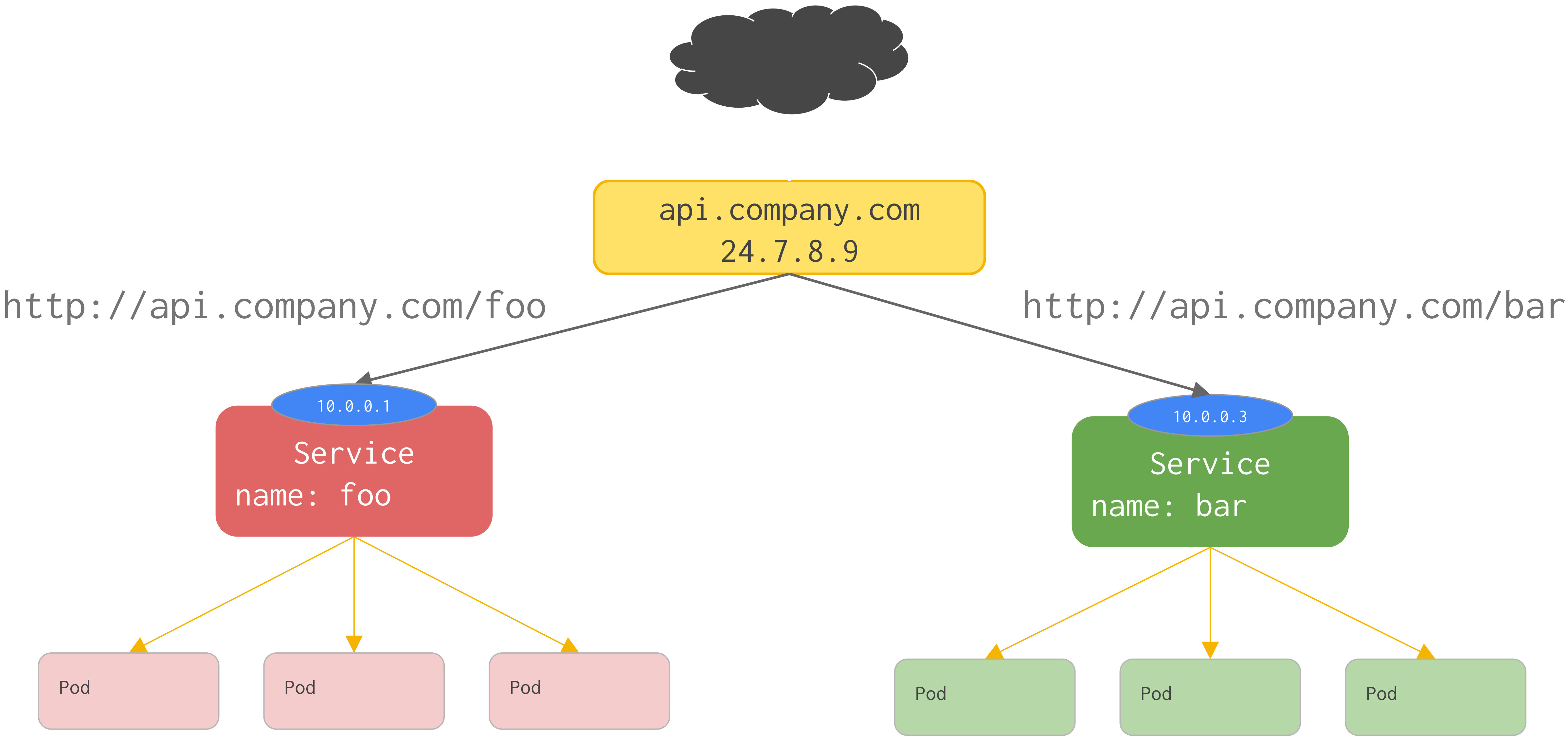


Ingress

Ingress for HTTP Load Balancing [Beta]



Ingress for HTTP Load Balancing [Beta]



api.company.com
24.7.8.9

http://api.company.com/foo

http://api.company.com/bar

10.0.0.1

Service
name: foo

10.0.0.3

Service
name: bar

Pod

Pod

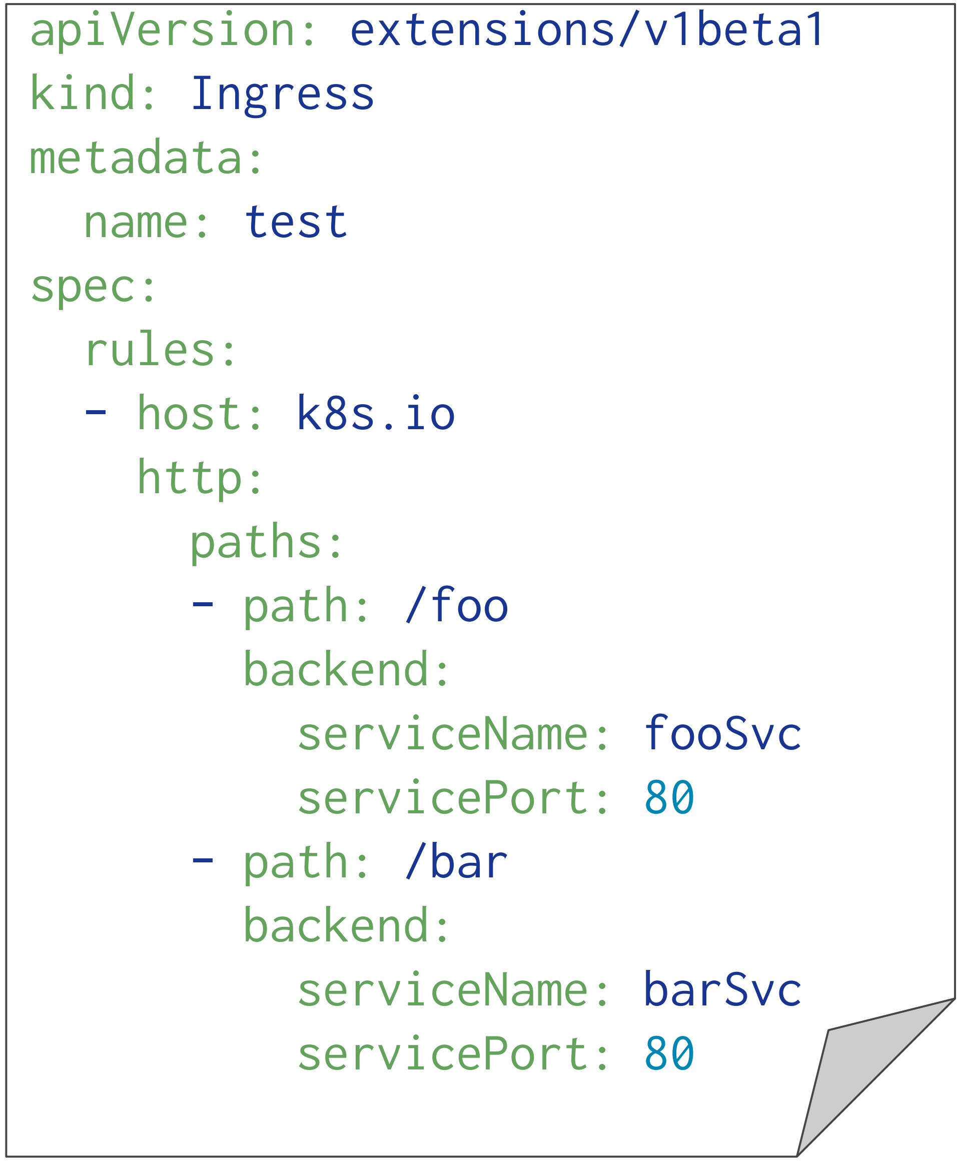
Pod

Pod

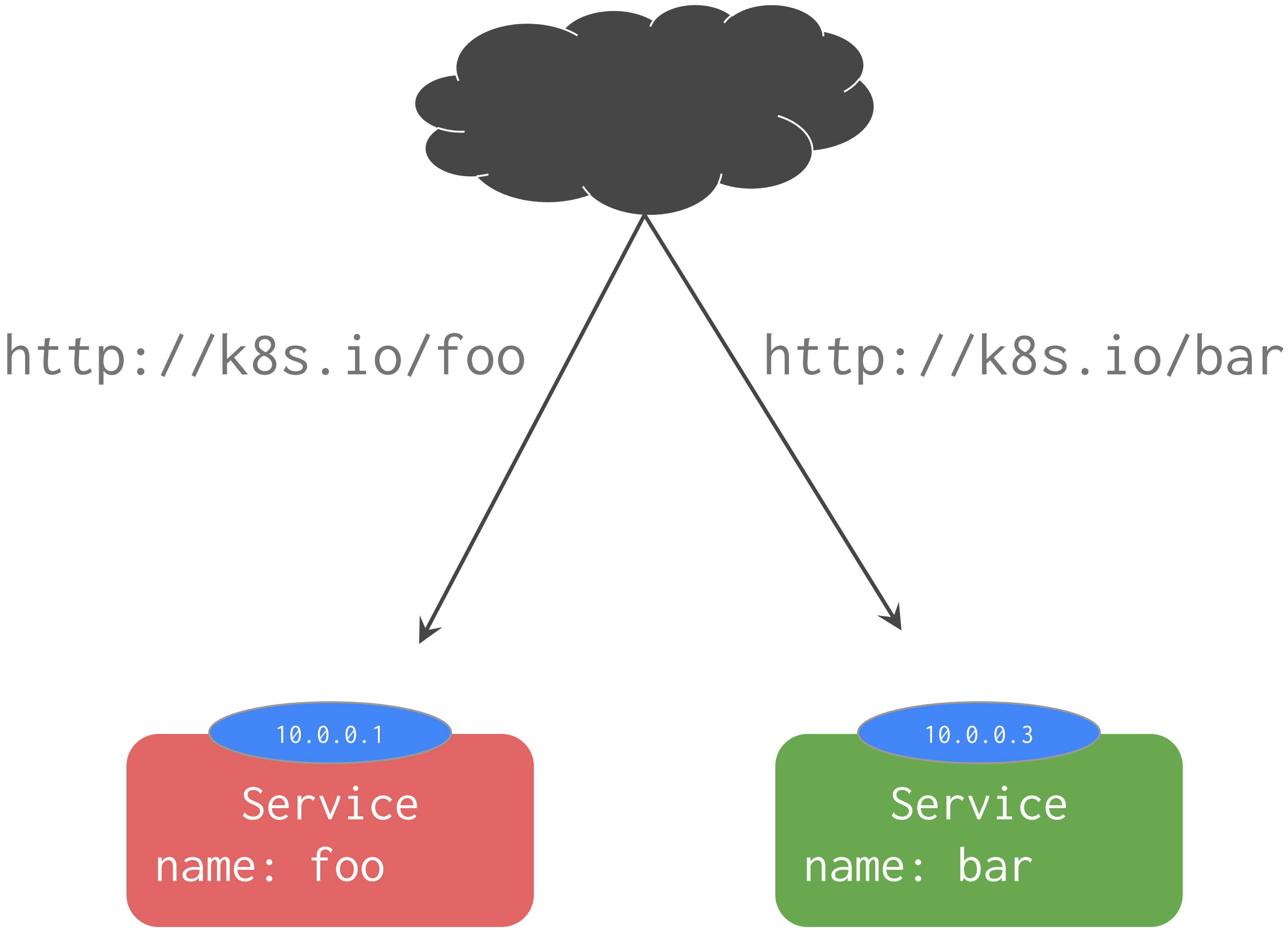
Pod

Pod

Ingress API: Simple fanout



```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: test
spec:
  rules:
  - host: k8s.io
    http:
      paths:
      - path: /foo
        backend:
          serviceName: fooSvc
          servicePort: 80
      - path: /bar
        backend:
          serviceName: barSvc
          servicePort: 80
```



http://k8s.io/foo

http://k8s.io/bar

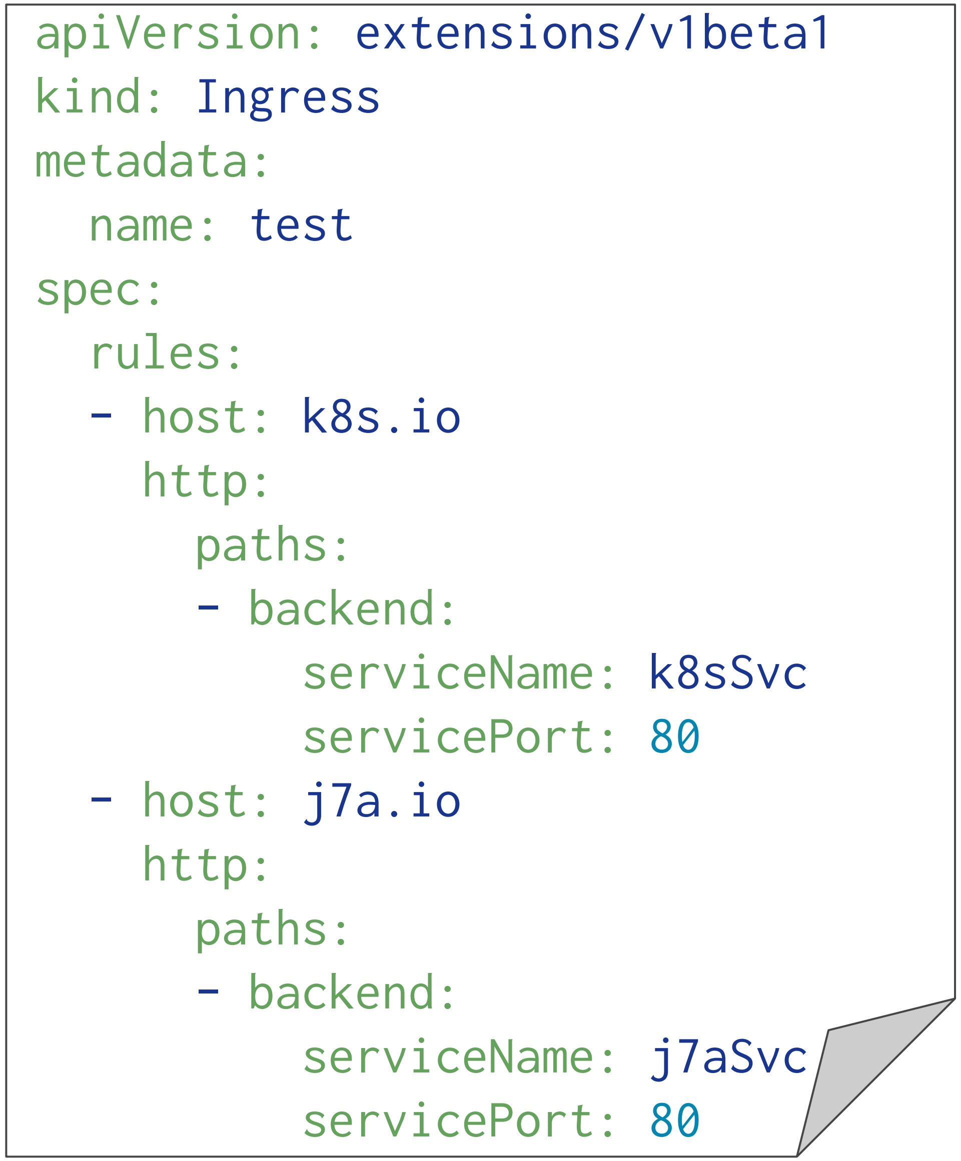
10.0.0.1

Service
name: foo

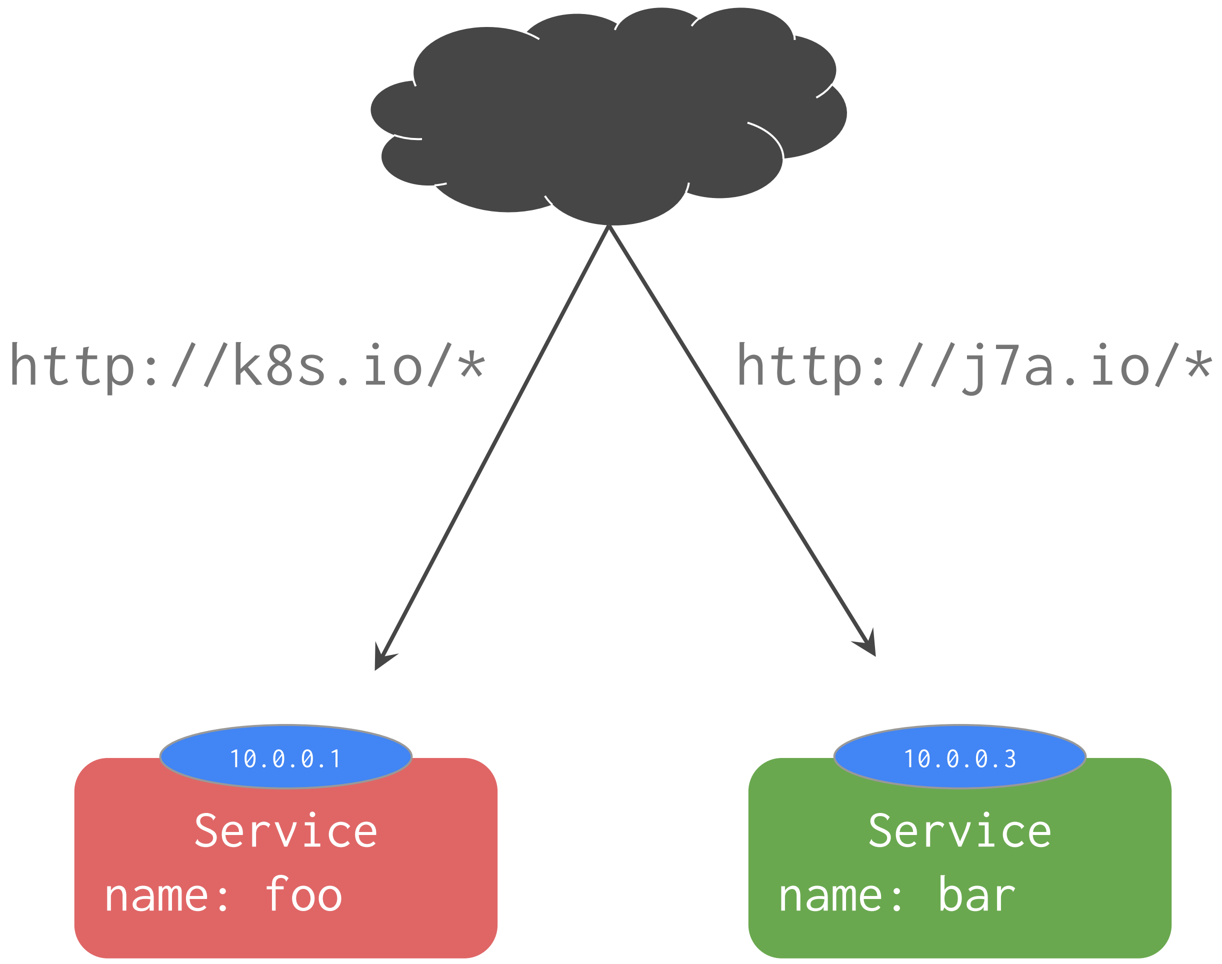
10.0.0.3

Service
name: bar

Ingress API: Name based virtual hosting



```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: test
spec:
  rules:
  - host: k8s.io
    http:
      paths:
      - backend:
          serviceName: k8sSvc
          servicePort: 80
  - host: j7a.io
    http:
      paths:
      - backend:
          serviceName: j7aSvc
          servicePort: 80
```



Ingress API

Services are assumed L3/L4

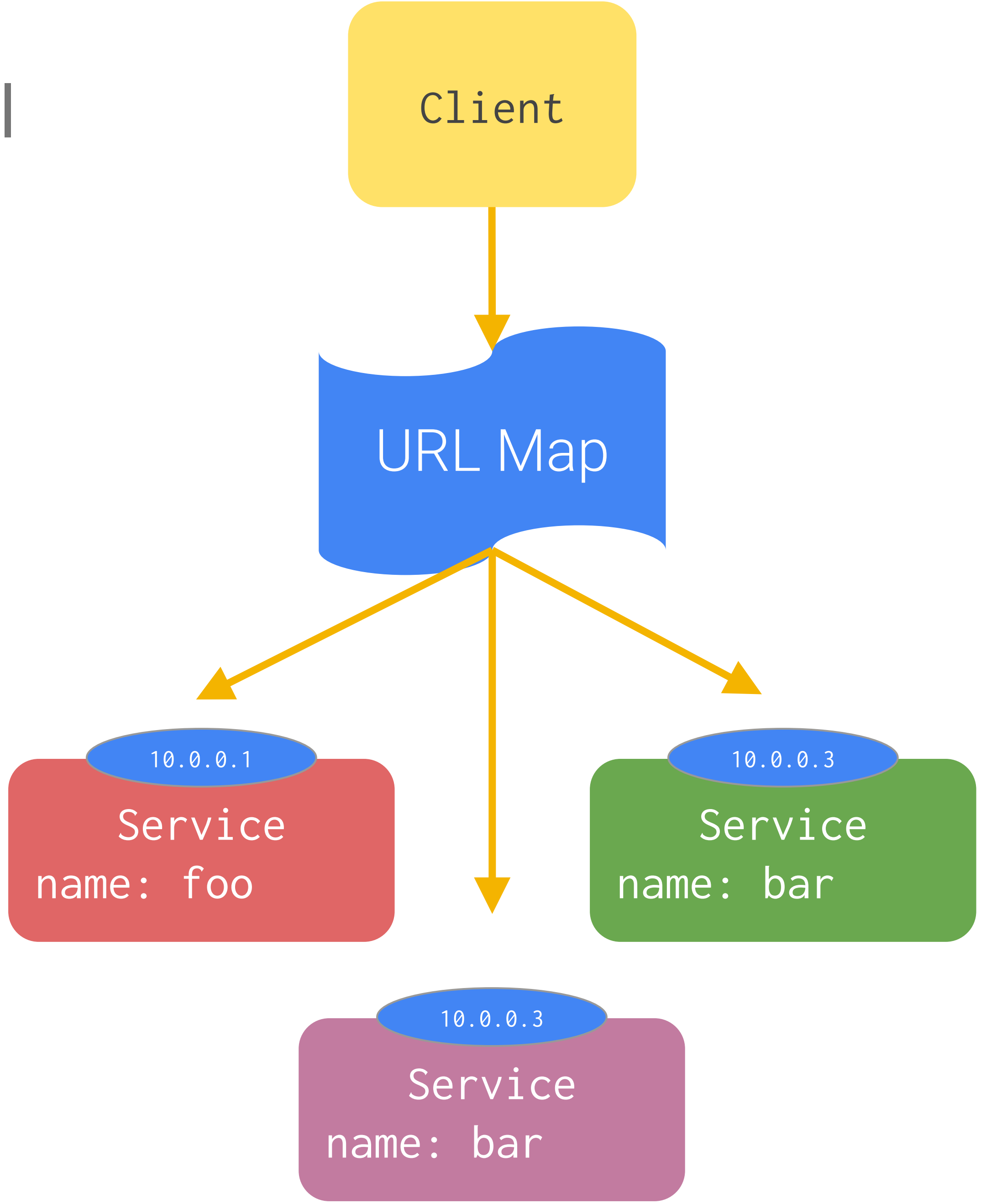
Lots of apps want HTTP/HTTPS

Ingress maps incoming traffic to backend services

- by HTTP host headers
- by HTTP URL paths

HAProxy, NGINX, AWS and GCE implementations in progress

Now with SSL!



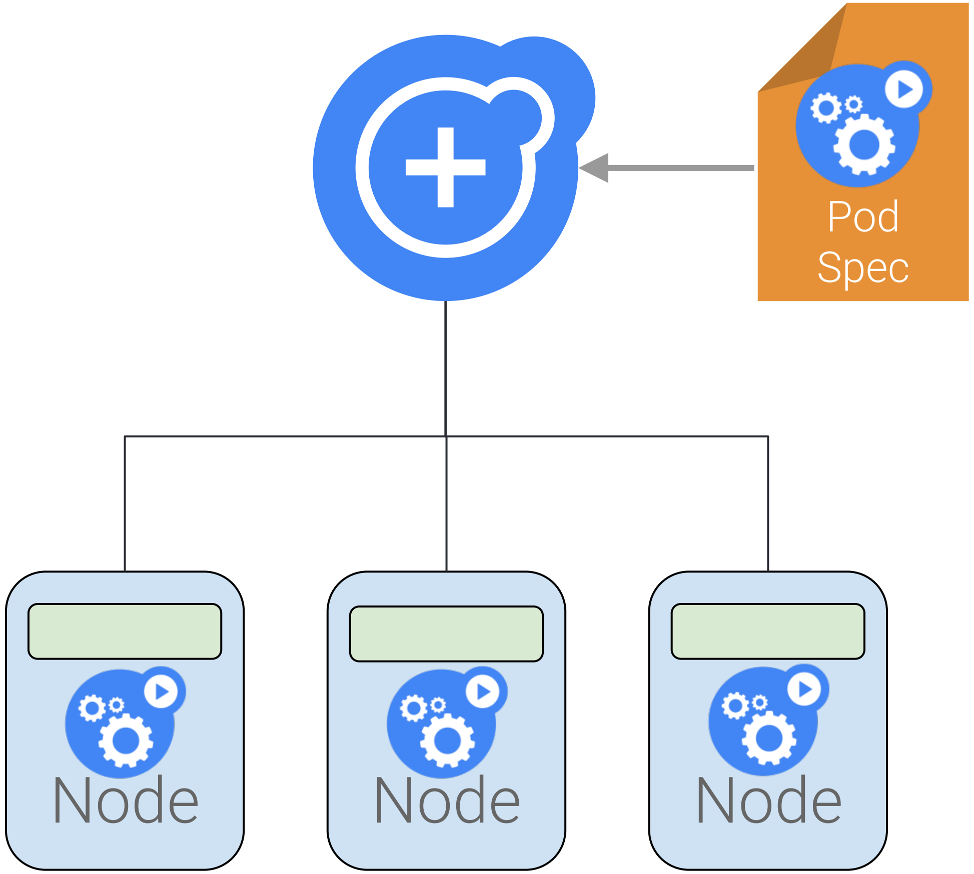
Daemonsets

Ensure that a replica of a given pod runs on every node or a subset nodes

Like an RC but targets a set of nodes by selector

Examples:

- Agent based services: DataDog, Sysdig, etc
- Daemon process: Storage, Logs, Monitoring



Network Policy

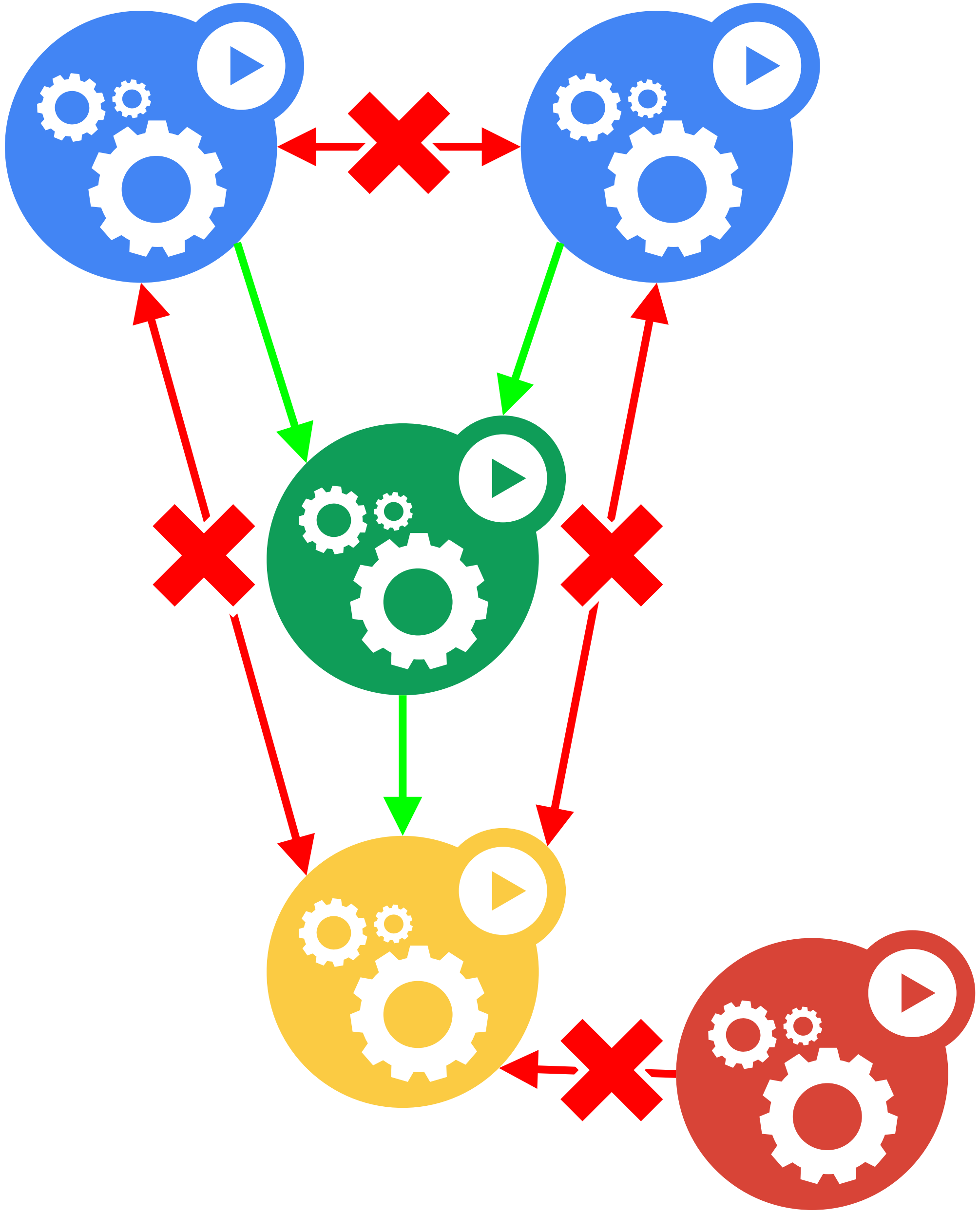
Describe the DAG of your app, enforce it in the network

Restrict Pod-to-Pod traffic or across Namespaces

Designed by the network SIG

- implementations for Calico, OpenShift, Romana, OpenContrail (so far)

Status: beta in v1.6



Node Drain

Goal: Evacuate a node for maintenance

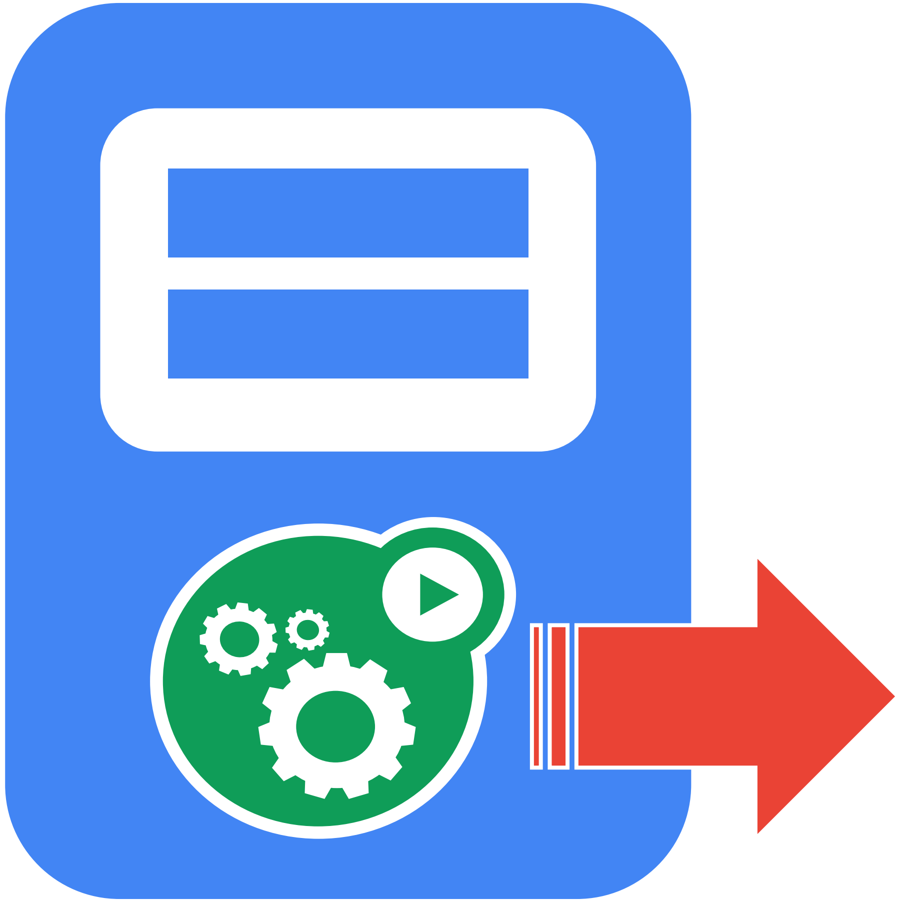
- e.g. kernel upgrades

CLI: kubectl drain

- disallow scheduling
- allow grace period for pods to terminate
- kill pods

When done: kubectl uncordon

- the node rejoins the cluster



Network Plugins

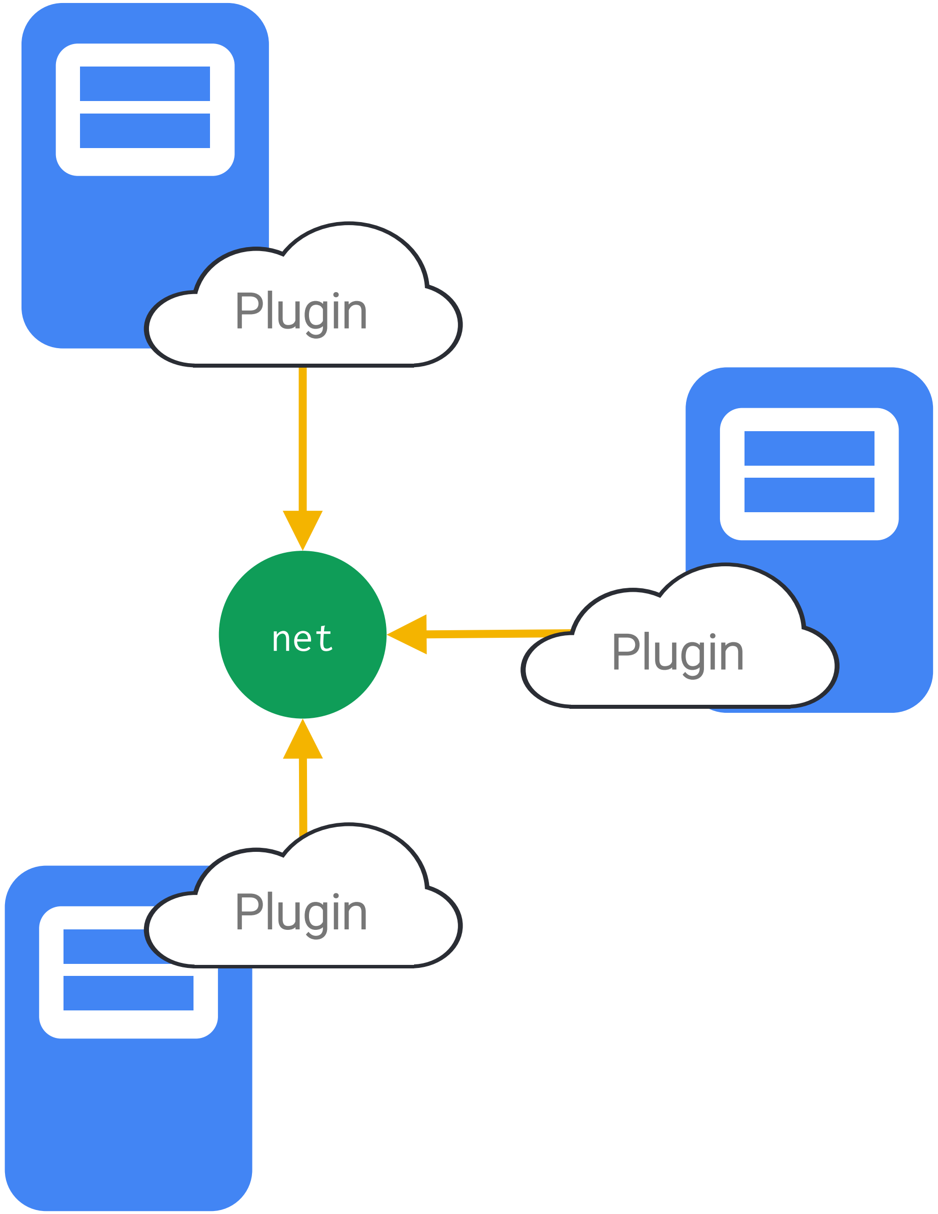
Introduced in Kubernetes v1.0

Uses **CNI**

- Simple exec interface
- Not using Docker libnetwork
  - but can defer to Docker for networking

Cluster admins can customize their installs

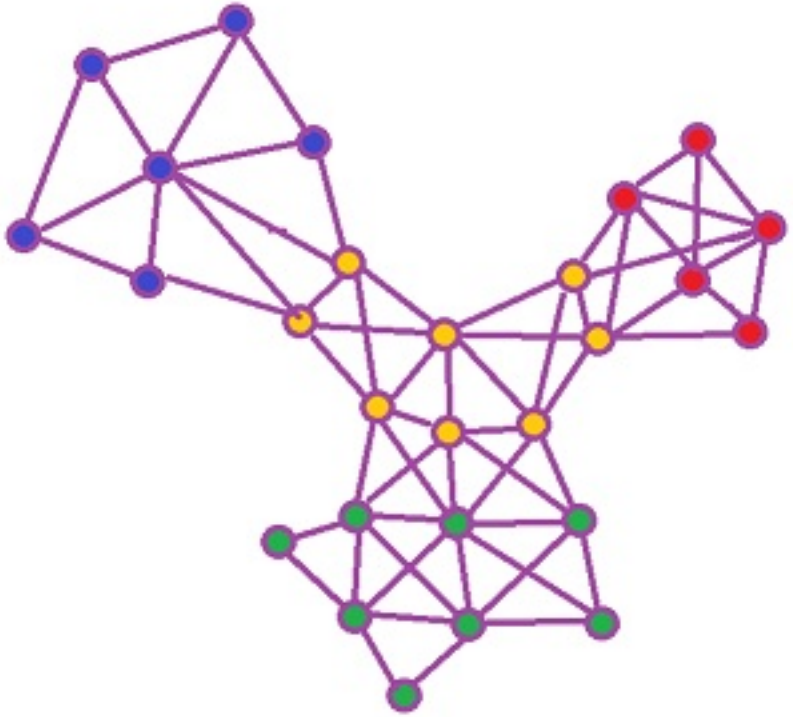
- DHCP, MACVLAN, Flannel, custom



More New and Coming Soon

- Cron (scheduled jobs)
- Custom metrics
- “Apply” a config (more declarative)
- Machine-generated Go clients (less deps!)
- Volume usage stats
- Multi-scheduler support
- Node affinity and anti-affinity
- More volume types
- Out-of-process volume plugin
- GUI
- Pod hostname and FQDN
- Better isolation
- Multi-cluster federation
- API federation
- Private Docker registry
- External DNS integration
- Volume classes and provisioning
- DIY Cloud Provider plugins
- More container runtimes
  (e.g. Rkt, Hyper)
- Better auth{n,z}
- Big data integrations
- Device scheduling (e.g. GPUs)
- ...

Cluster Add-Ons



Monitoring

Run cAdvisor on each node (in kubelet)

- gather stats from all containers
- export via REST

Run Heapster as a pod in the cluster

- just another pod, no special access
- aggregate stats

Run Influx and Grafana in the cluster

- more pods
- alternately: store in Google Cloud Monitoring

Or plug in your own!

- e.g. Google Cloud Monitoring

Logging

Run fluentd as a pod on each node

- gather logs from all containers
- export to elasticsearch

Run Elasticsearch as a pod in the cluster

- just another pod, no special access
- aggregate logs

Run Kibana in the cluster

- yet another pod

Or plug in your own!

- e.g. Google Cloud Logging

DNS

Run CoreDNS as a pod in the cluster

- Use etcd as a backend
- Successor of SkyDNS

Strictly optional, but practically required

- LOTS of things depend on it (SkyDNS)
- Probably will become more integrated

Can do

- DNS(SEC)
- Caching
- Health Checks

Or plug in your own!

Kubernetes a healthy eco system

Cloud providers: Azure, VMware, Openstack, Rackspace,

Distros: Kubermatic, CoreOS Tectonic, RedHat Atomic

PaaS: RedHat OpenShift, Deis, Rancher, WSO2, Gondor/Kel, Apcera

CD: Fabric8, Shippable, CloudBees, Solano

Deployment: Kumoru, Redspread, Spinnaker

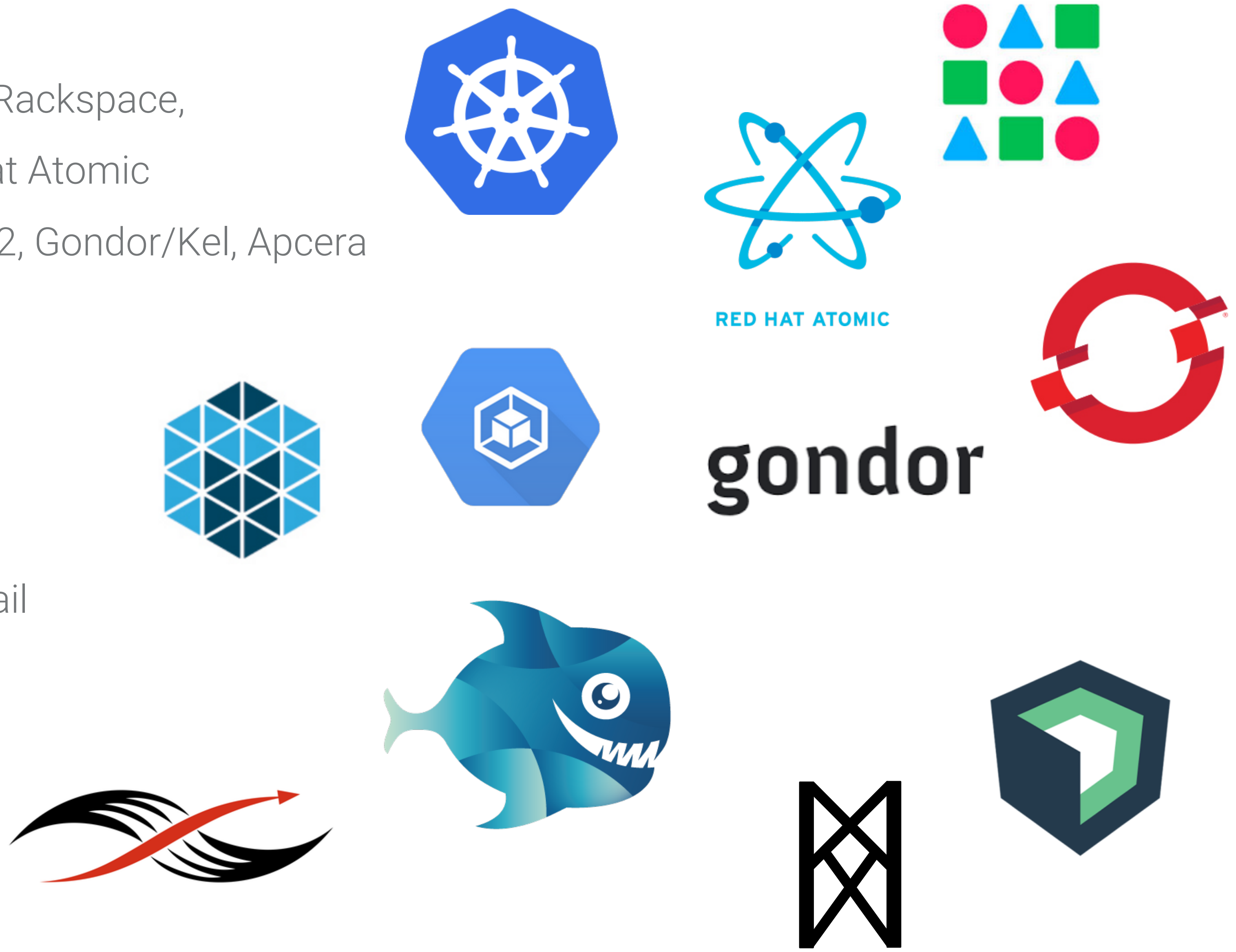
Package managers: Helm, KPM

Monitoring: Prometheus, Sysdig, Datadog

Networking: Weaveworks, Tigera, OpenContrail

Storage: NetApp, ClusterHQ

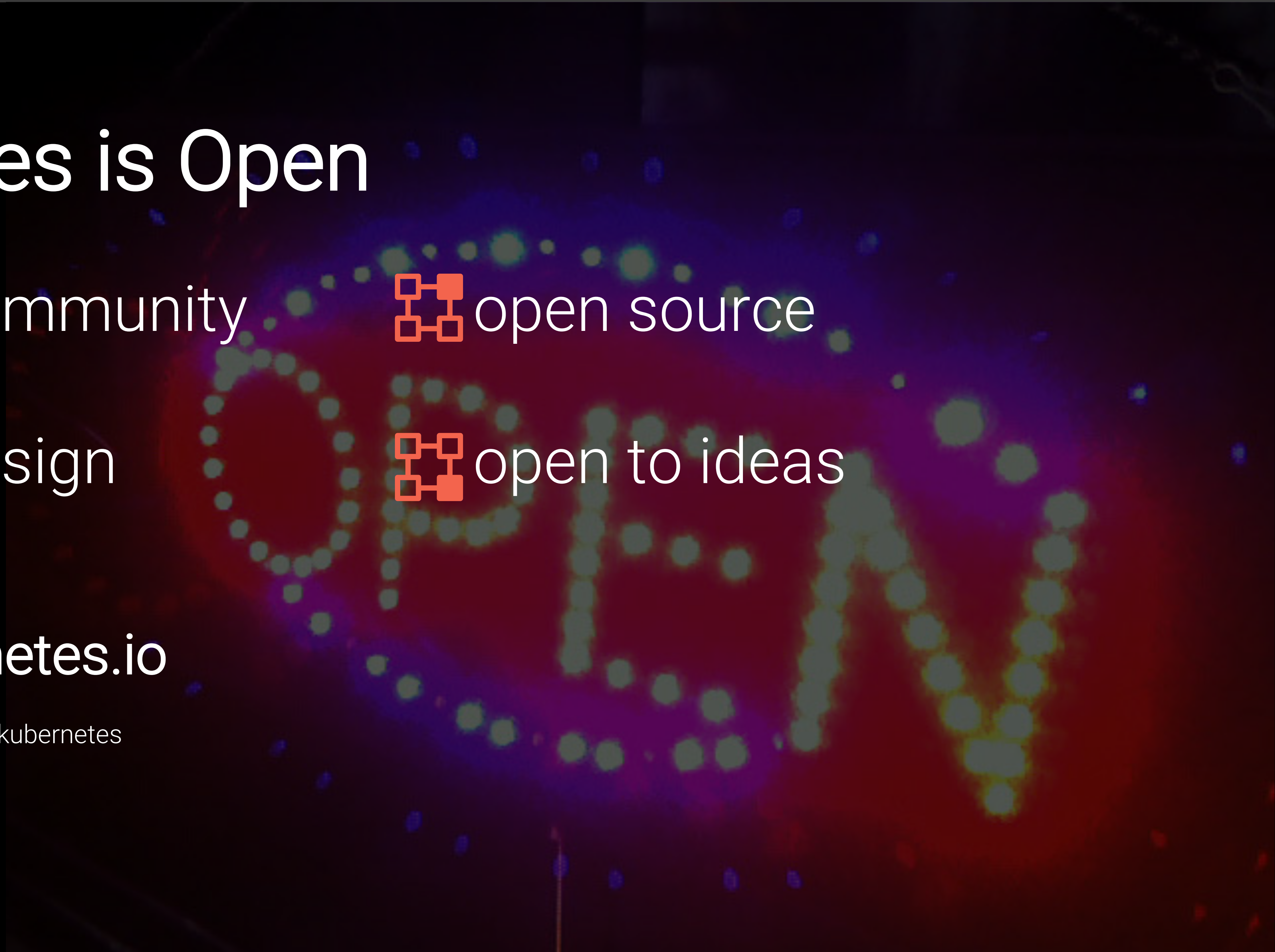
Appliances: Redapt, Diamante



Hitting a ~3 month release cadence

Kubernetes is Open

- open community
- open design
- open source
- open to ideas



https://kubernetes.io

Code: github.com/kubernetes/kubernetes
Chat: slack.k8s.io
Twitter: @kubernetesio

Kubermatic Container Engine

Managed Kubernetes

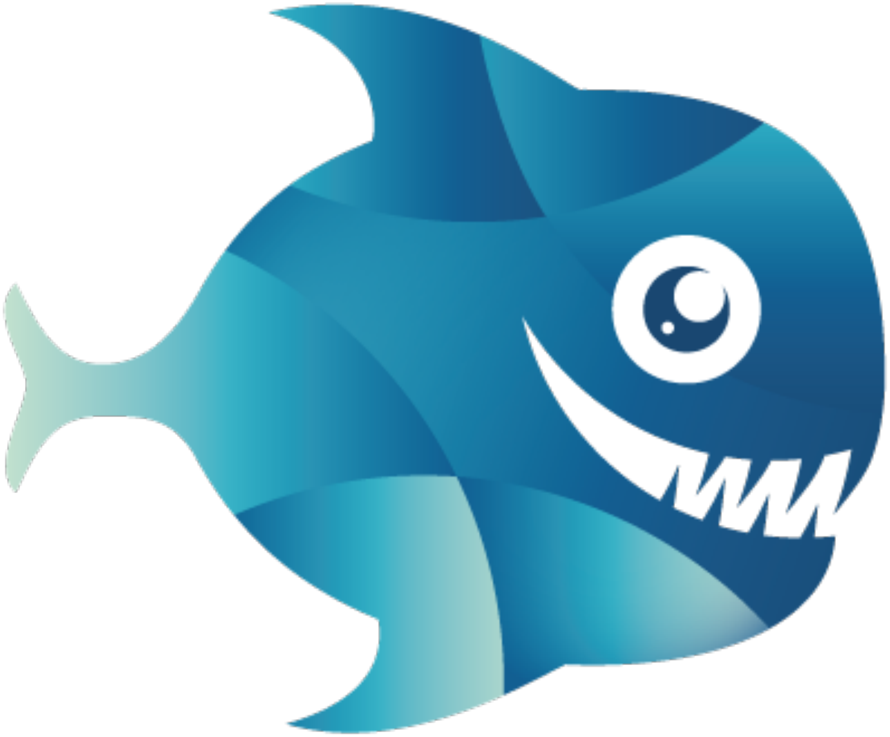
Manages Kubernetes master uptime

Cluster Resize

Managed Updates

Supports Different Cloud providers AWS, DigitalOcean, BareMetal

Also On-premises



Thank you for your attention!

Contact:

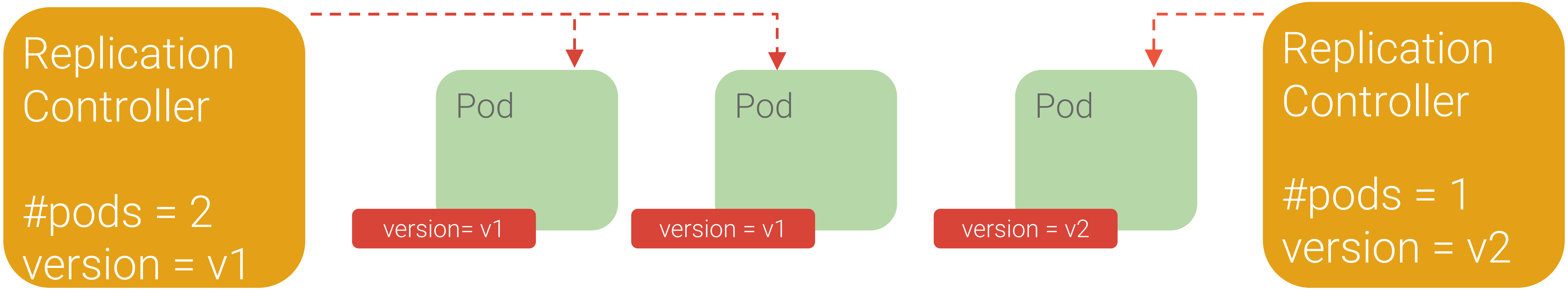
luk@loodse.com	✉	henrik@loodse.com
@lukburchard	🐦	@mr_inc0mpetent
realfake		mrIncompetent

+49 1590 4099172

www.loodse.com

Your training Virtual Machine

Replication Controllers



Behavior

- Keeps Pods running
- Gives direct control of Pods
- Grouped by Label Selector

Benefits

➔ Recreates Pods, maintains desired state

➔ Fine-grained control for scaling

➔ Standard grouping semantics

loodse