Foundation of Internet Platform Development & Operation

Overview of Resource Management System I

2019-10-08

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Introduction

- Foundation of Resource Management
  - Isolation Technologies
    - Virtual Machine
      - Xen, KVM
    - Container
      - cgroup
        - lxc, docker, rkt, OCI
  - Orientation
    - Resources (Hardware)
    - Application (Software)

Introduction

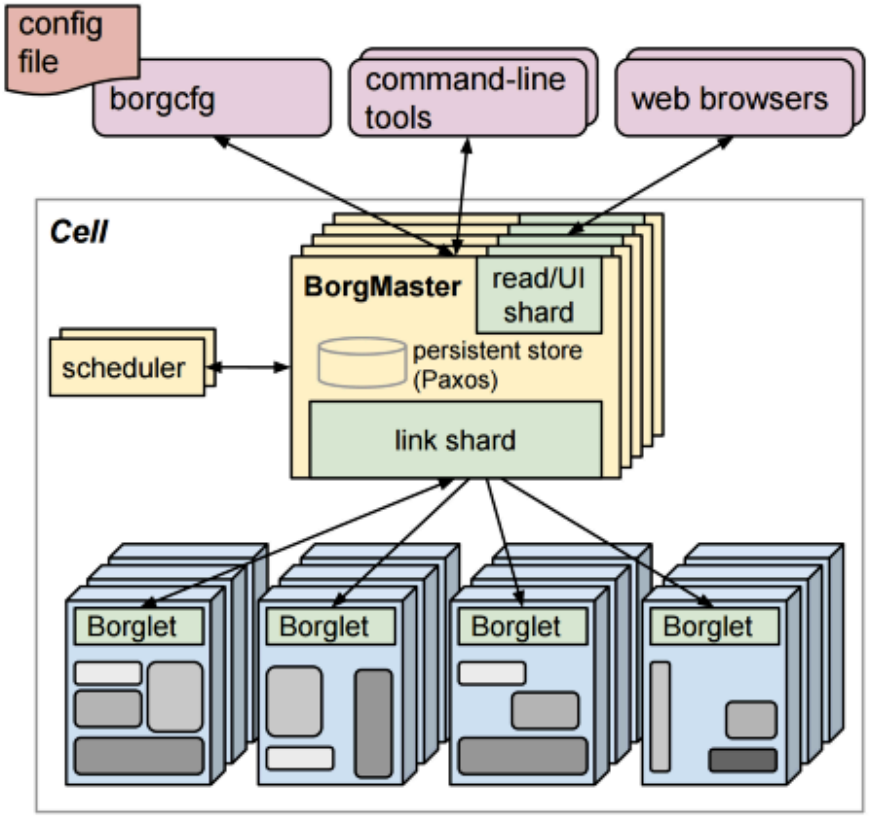
- Workload
  - Long-running services
    - Resource occupied as long as the service is running
    - High QoS
    - High Availability
  - Batch jobs
    - Resource occupied for a short while
    - Low QoS
    - Low Availability

Introduction

- The metrics of a resource management system
  - Resource
    - Utilization
    - Scalability
    - Fault tolerance
  - Application
    - Scalability
    - Fault tolerance
    - Autoscaling
  - Workload
    - Throughput

Architecture

- Borg

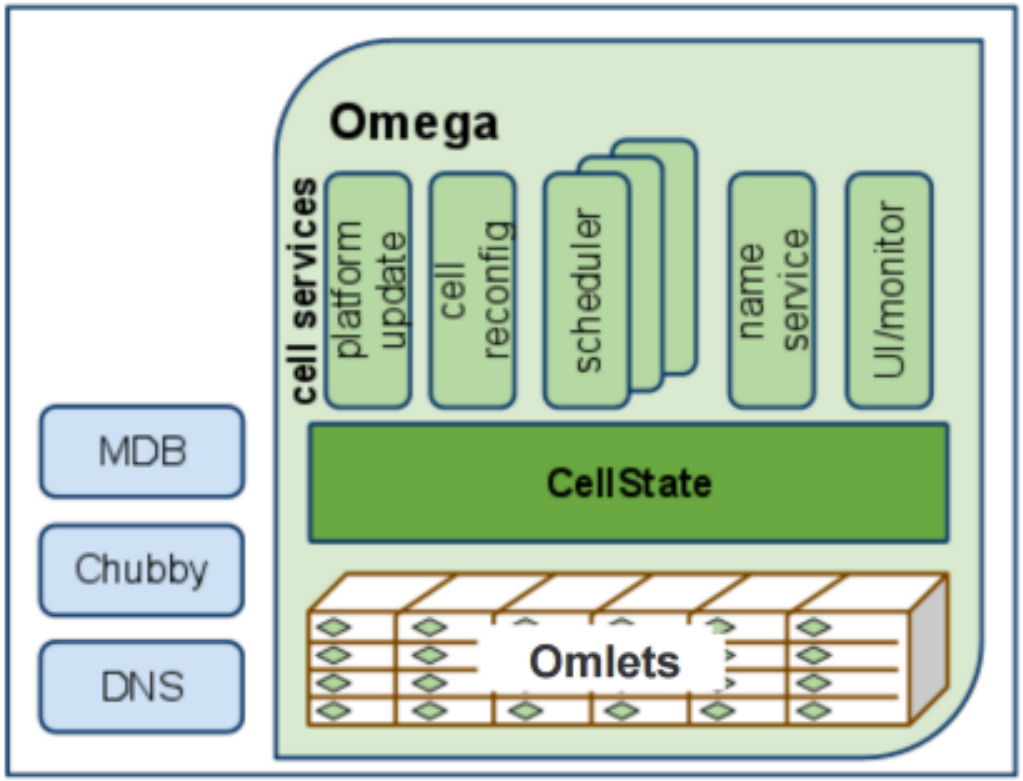


Architecture

- Borg
  - Naming and service discovery (using Borg Name Service)
  - Application-aware load balancing
  - Horizontal and vertical autoscaling
  - Rollout tools for deploying software/configuration updates
  - Workflow tools for running multi-job analysis pipelines with interdependencies between the stages
  - Monitoring tools to gather information about containers, aggregate, present on dashboards, and trigger alerts

Architecture

- Omega

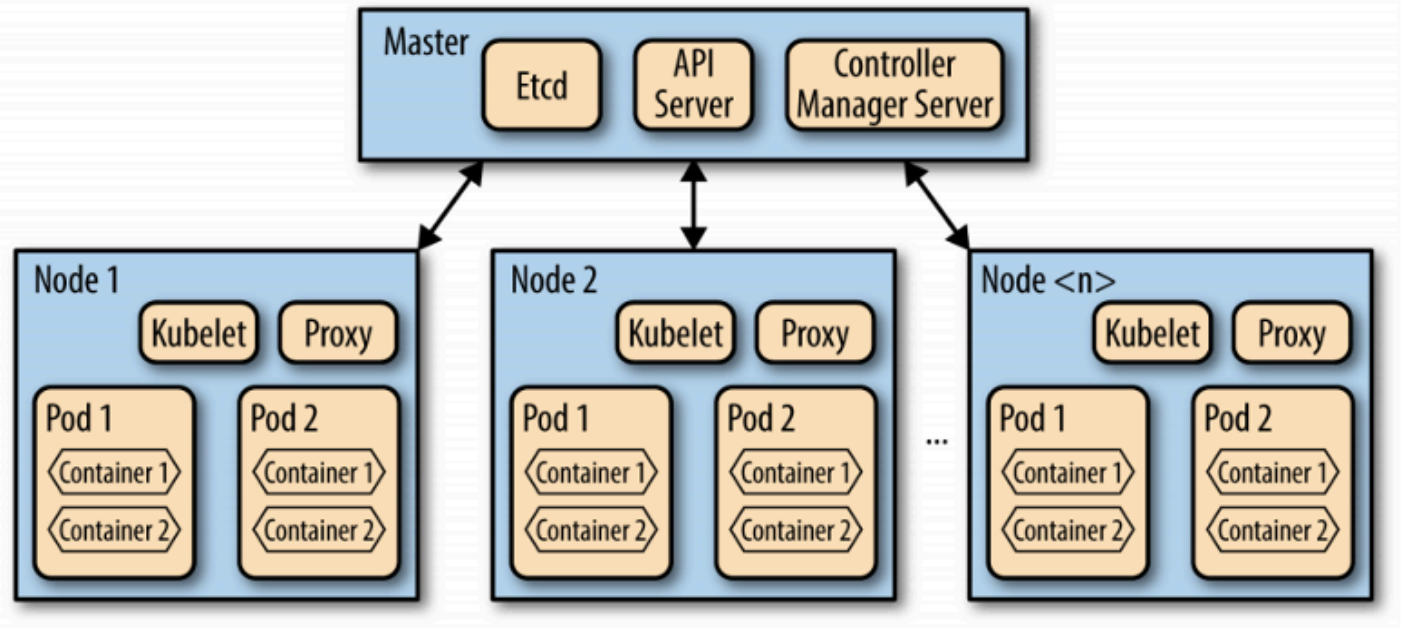


Architecture

- Omega
  - A separate centralized Paxos-based transactional store
  - Microservice-like architecture
  - External cell state persistent model
    - Multiple schedulers run against the same cell state with optimistic concurrency controls
    - Optimizing scheduler busyness and job wait time significantly

Architecture

- Kubernetes



Quiz

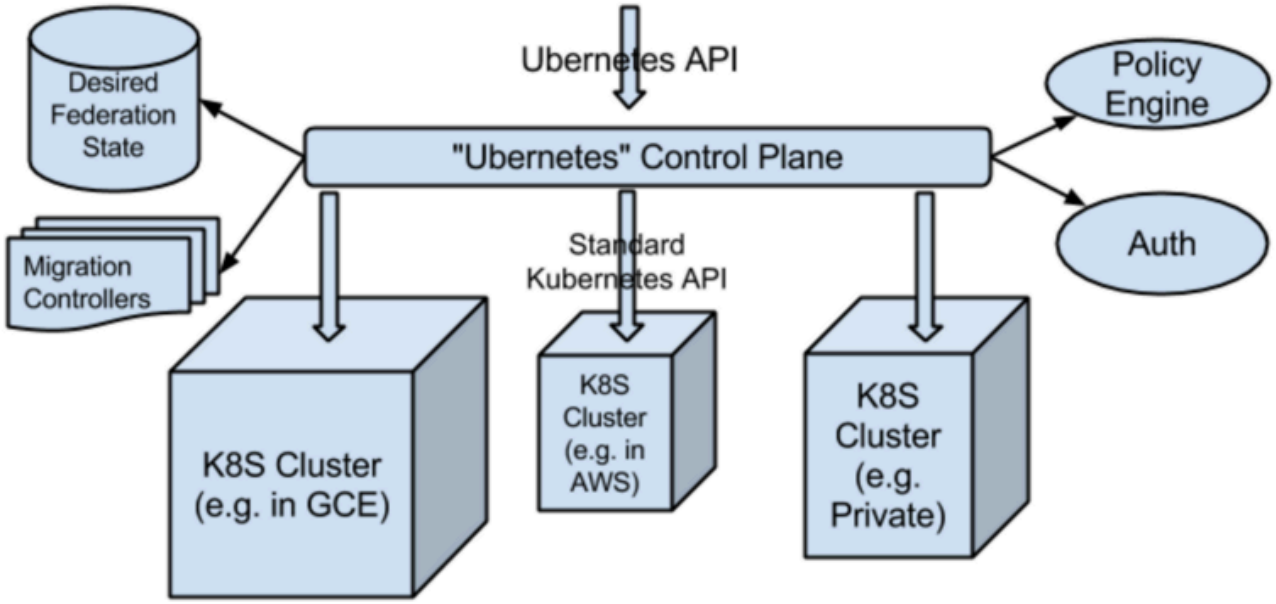
- Centralized VS. Decentralized (Distributed, Hierarchy):
  - Preferred scenarios

Architecture

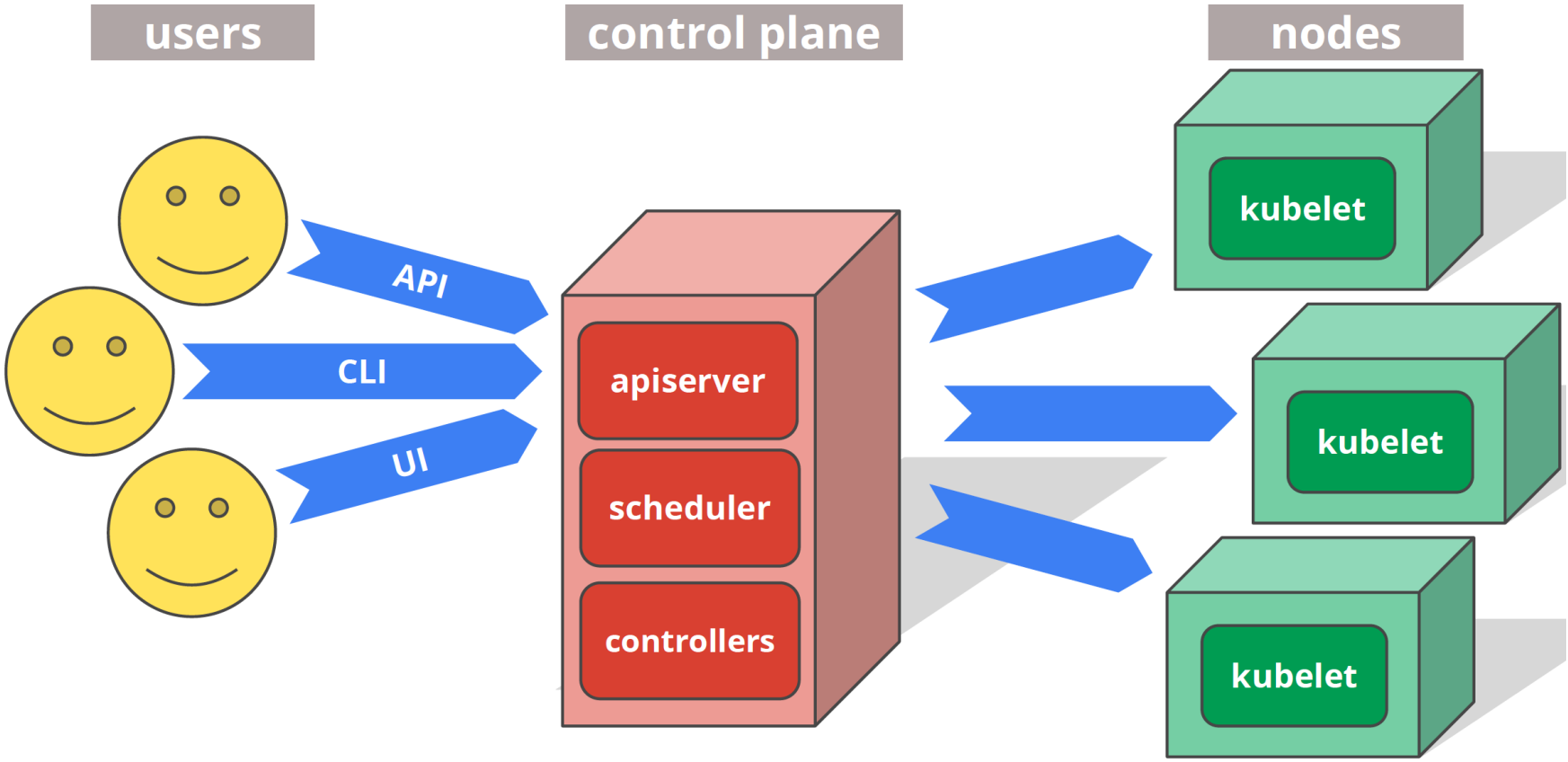
- Kubernetes
  - Automatic binpacking (Mix critical and best-effort workloads optimal resource usage)
  - Self-healing
  - Horizontal manual/auto-scaling
  - Service discovery & load balancing
  - Automated rollouts and rollbacks
  - Secret and configuration management
  - Storage orchestration
  - Batch execution
  - ***An Open Architecture***

Architecture

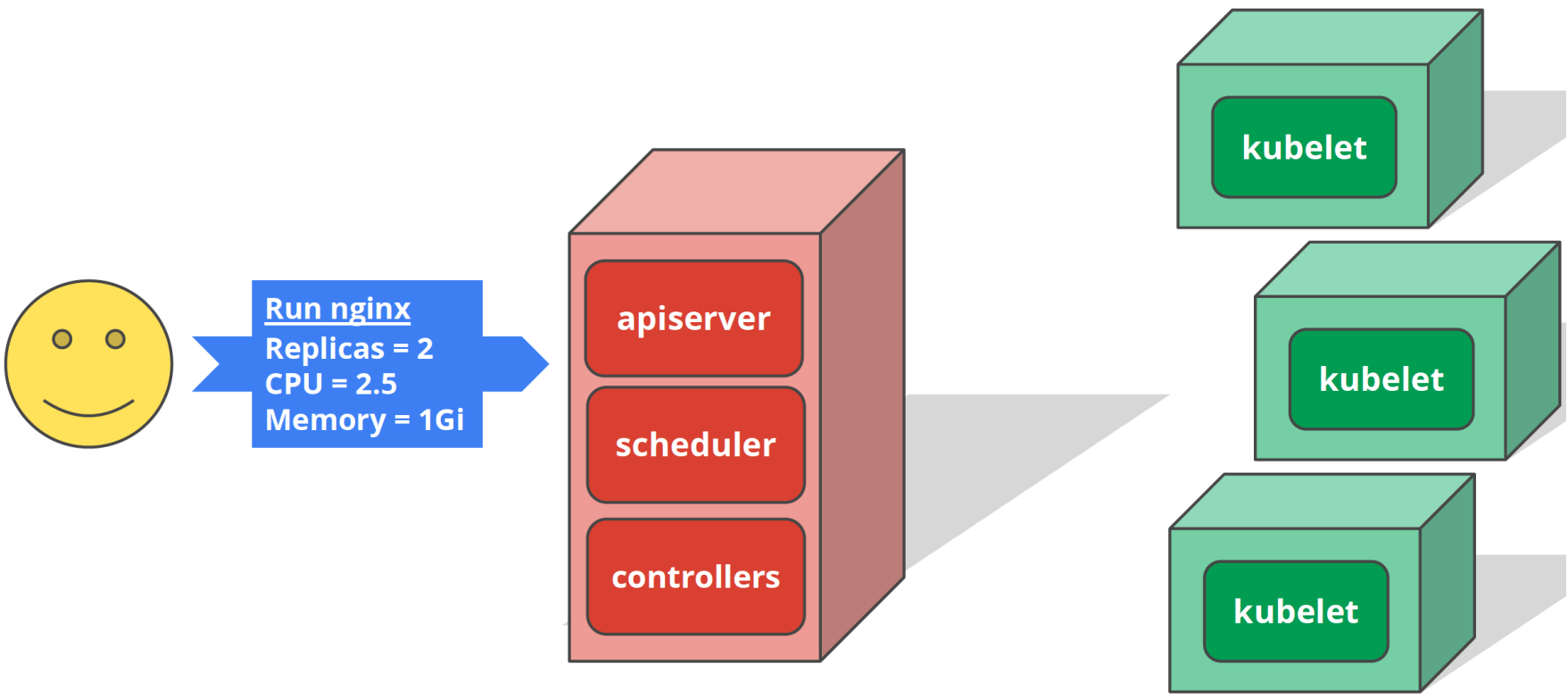
- Kubernetes
  - Cluster Federation OR Mult-region



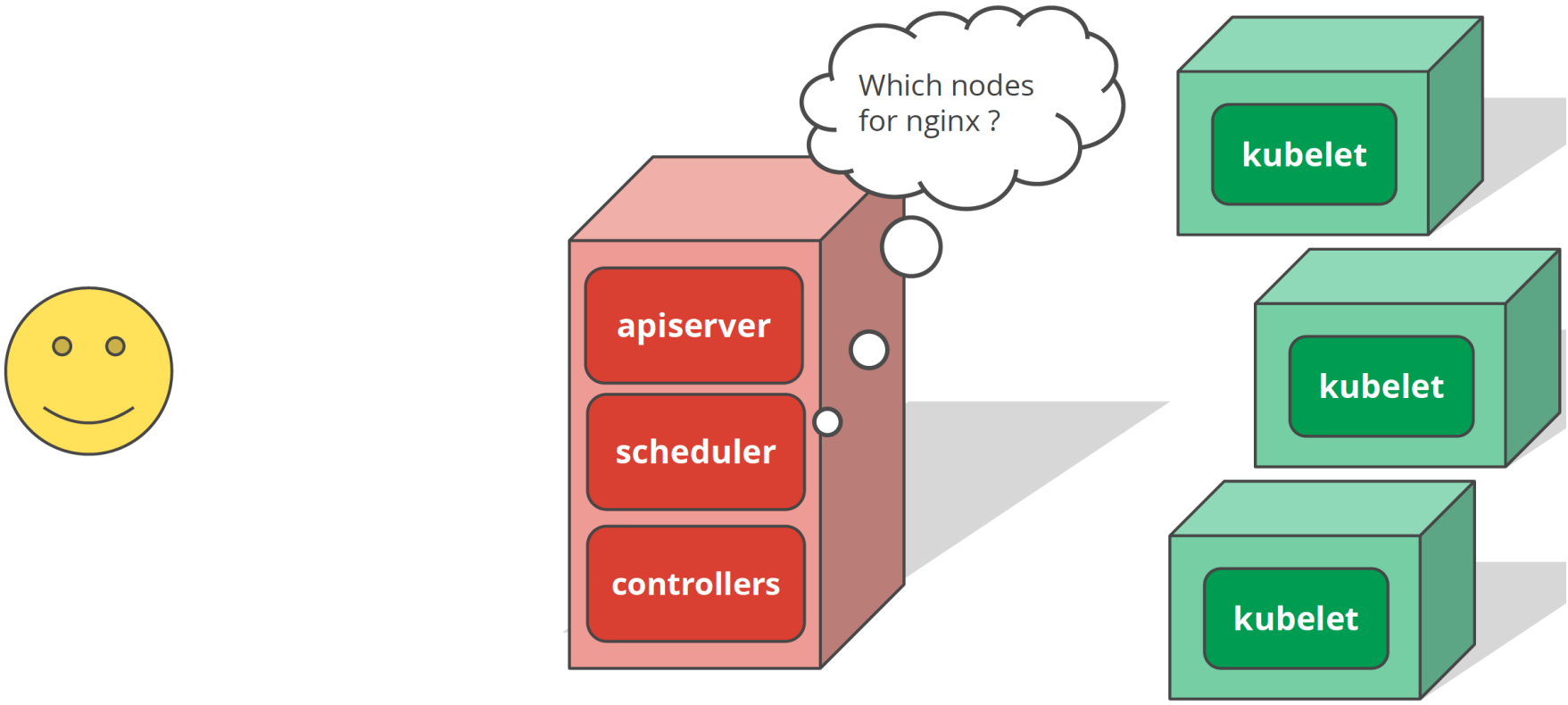
Architecture



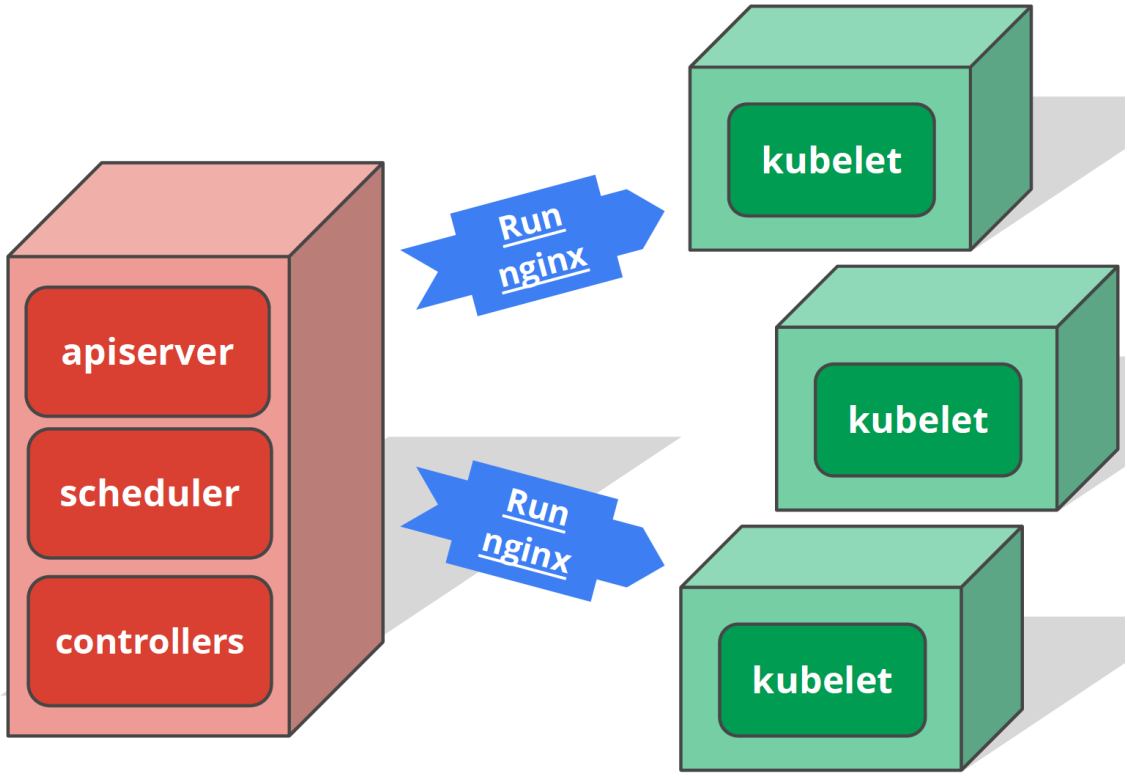
Architecture



Architecture

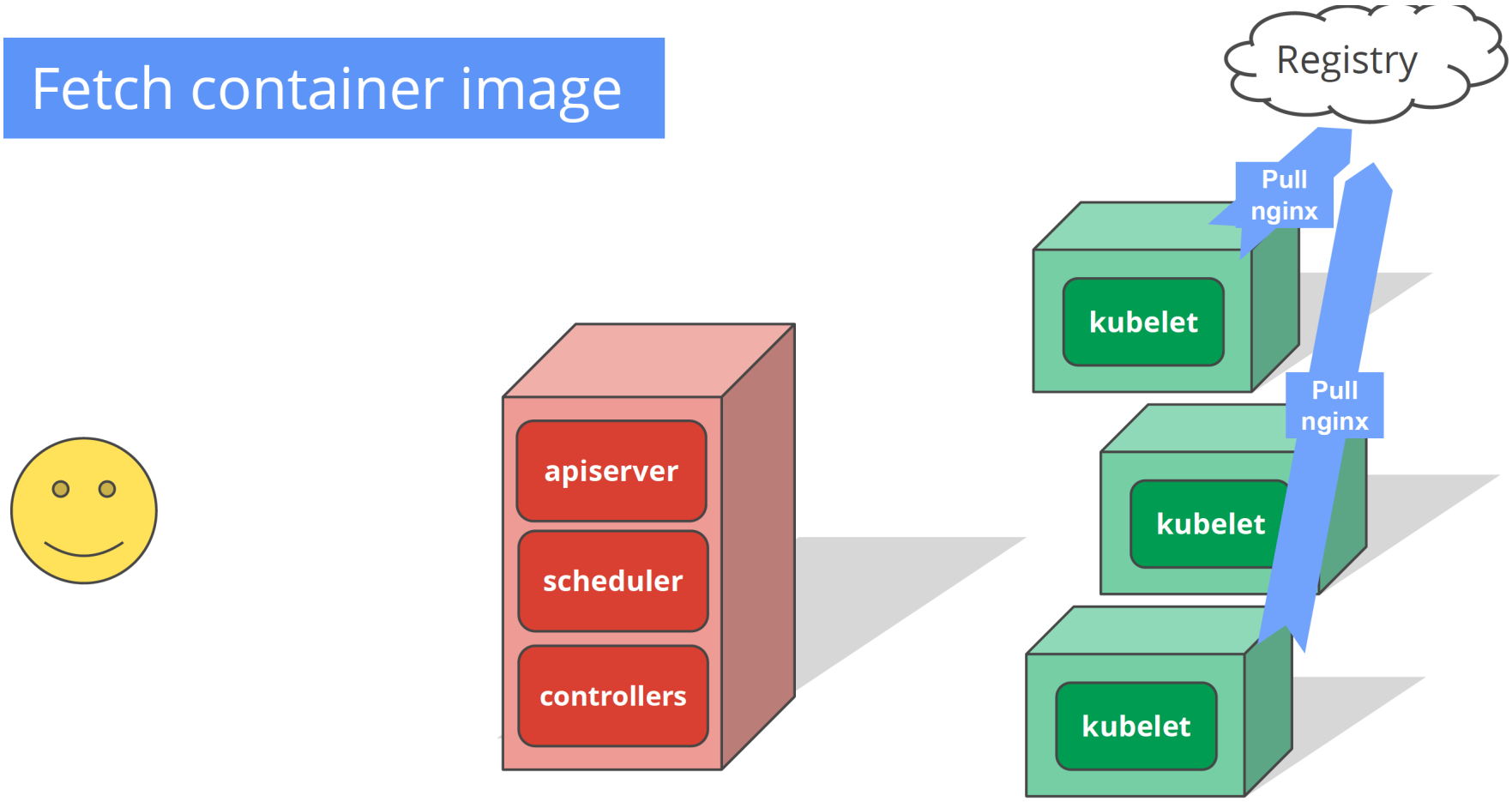


Architecture

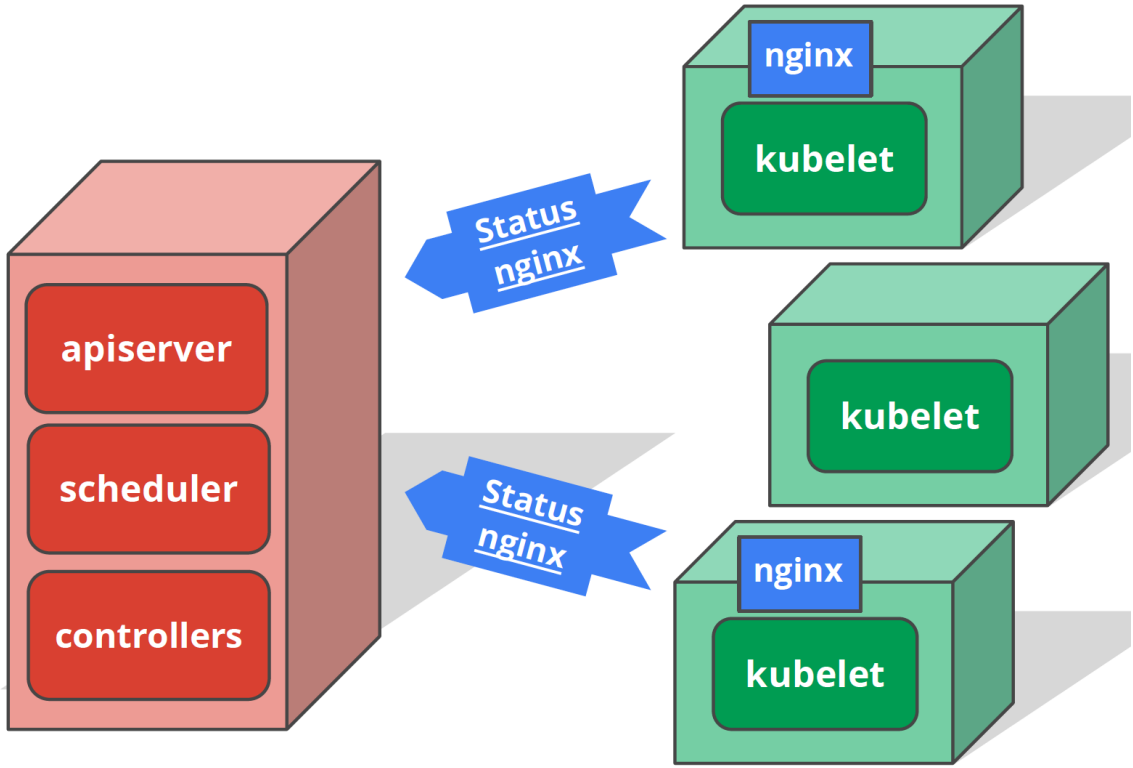


Architecture

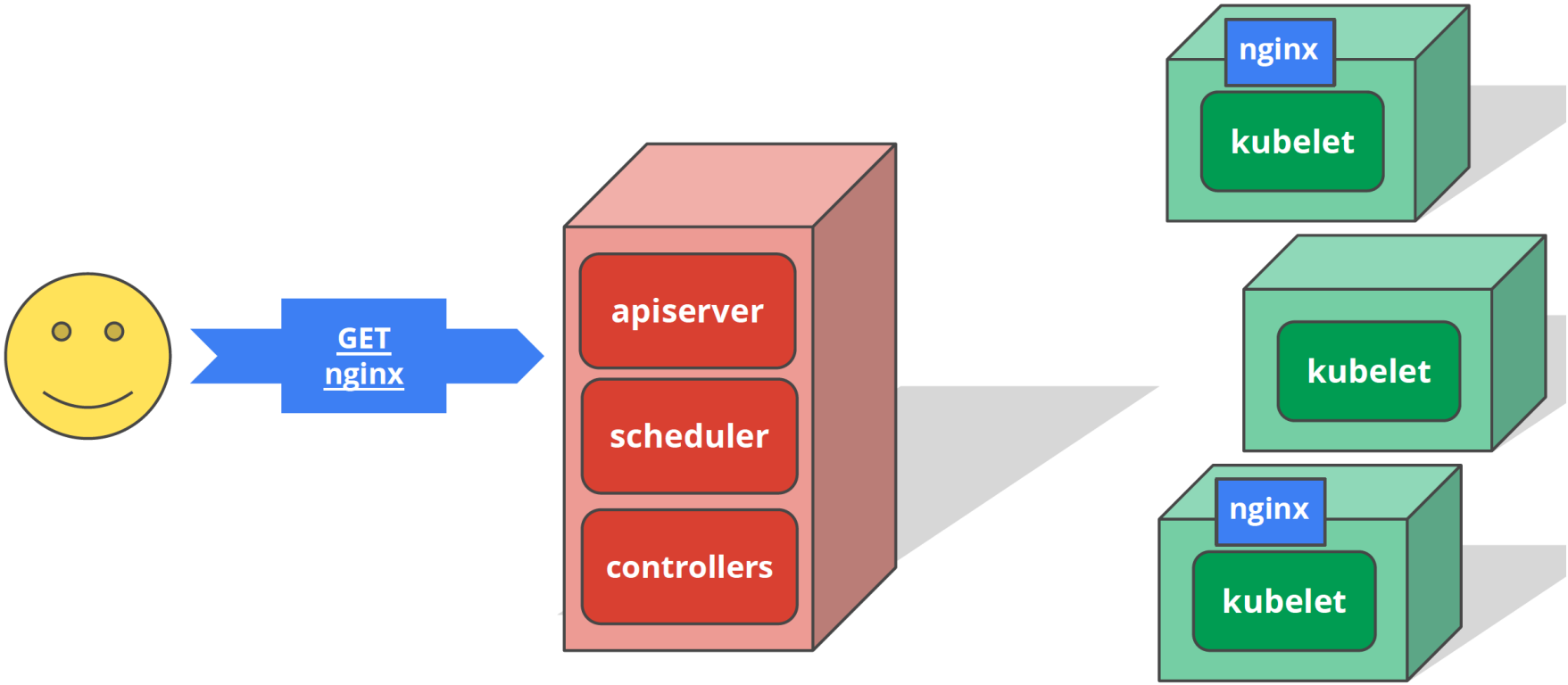
Fetch container image



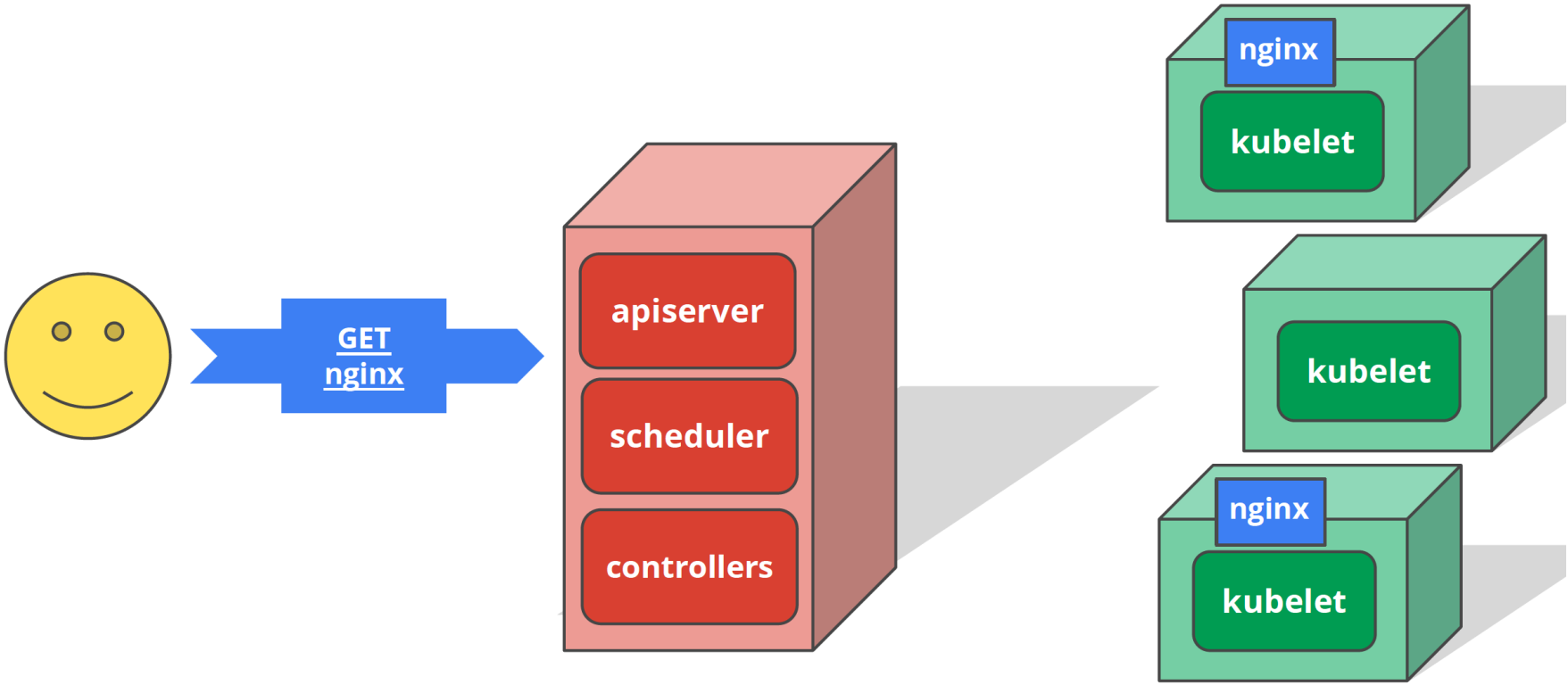
Architecture



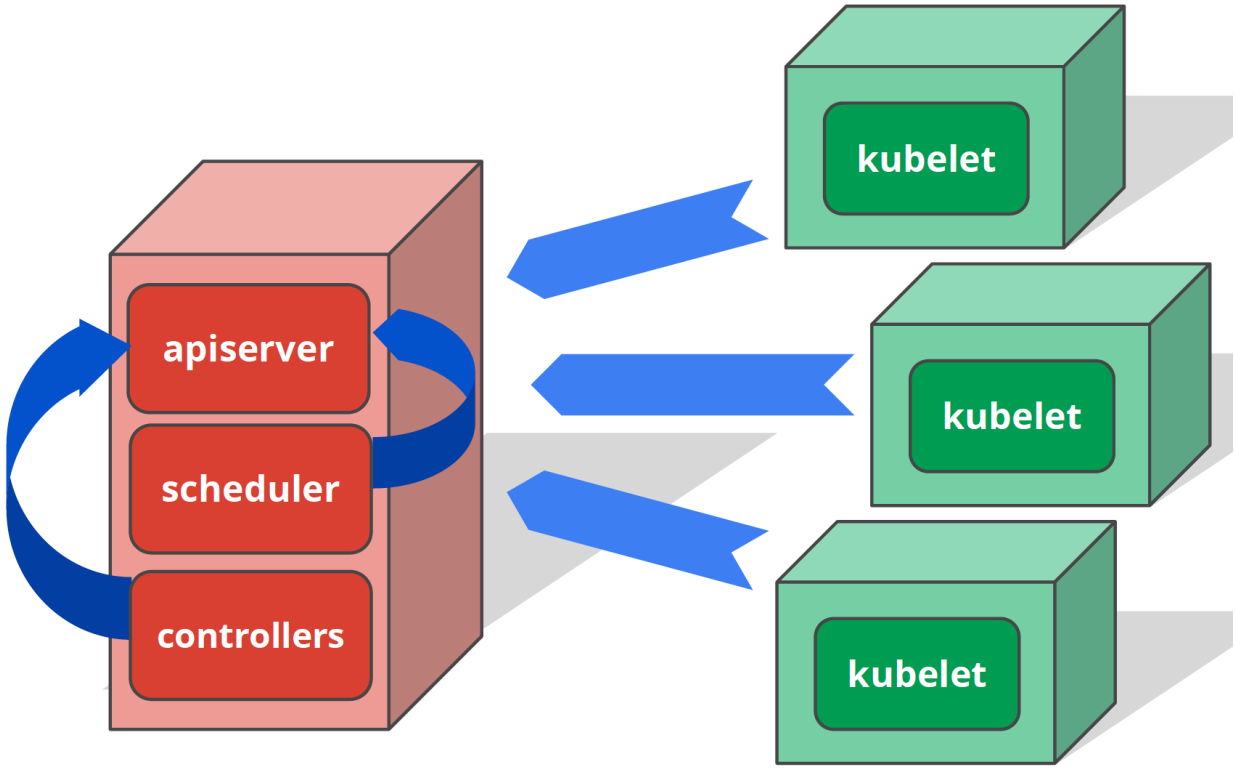
Architecture



Architecture



Architecture



Quiz

- Problem I:
  - How to guarantee HA of an application?
- Problem II:
  - How does HA interfere utilization?
    - How to balance HA and utilization?

Kubernetes

- Components
  - Master components
    - Kube-apiserver
      - The front-end for the Kubernetes control plane
      - Scalable horizontally
    - Etcd
      - Consistent and highly-available
      - key value store used
        - all cluster data

Kubernetes

- Components
  - Master components
    - kube-scheduler
      - individual and collective resource requirements
      - hardware/software/policy constraints
      - affinity and anti-affinity specifications
      - data locality
      - inter-workload interference and deadlines

Kubernetes

- Components
  - Master components
    - Kube-controller-manager
      - Node Controller
        - Noticing and responding when nodes go down.
      - Replication Controller
        - Maintaining the correct number of pods for every replication controller object in the system.
      - Endpoints Controller
        - Populating Endpoints object (that is, joins Services & Pods).
      - Service Account & Token Controllers
        - Creating default accounts and API access tokens for new namespaces.

Kubernetes

- Components
  - Master components
    - cloud-controller-manager
      - Interacting with the underlying cloud providers

Kubernetes

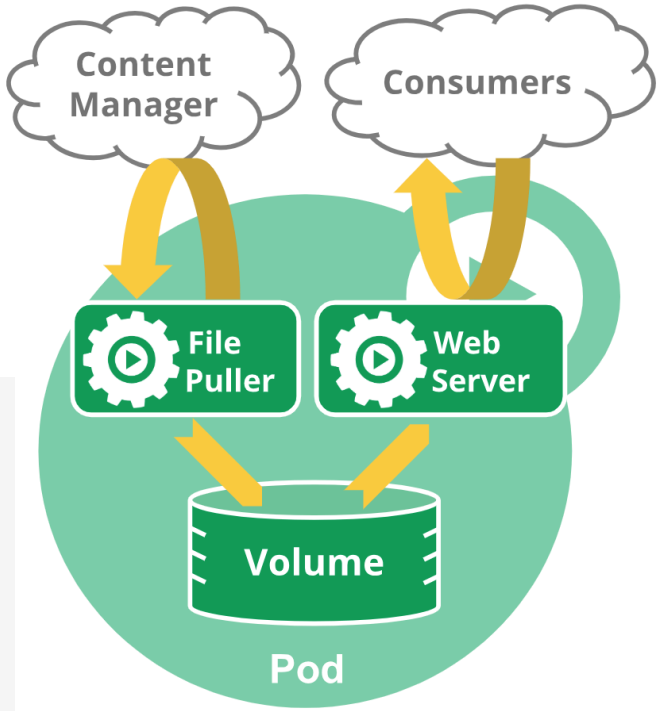
- Components
  - Node components
    - kubelet
      - An agent that runs on each node in the cluster. It makes sure that containers are running in a pod.
      - PodSpecs
    - kube-proxy
      - enables the Kubernetes service abstraction by maintaining network rules on the host and performing connection forwarding.
    - Container Runtime
      - responsible for running containers.
        - Kubernetes supports several runtimes: Docker, rkt, runc and any OCI runtime-spec implementation.

Kubernetes

- Components
  - Addons
    - DNS
      - While the other addons are not strictly required, all Kubernetes clusters should have cluster DNS, as many examples rely on it.
    - Web UI (Dashboard)
      - Dashboard is a general purpose, web-based UI for Kubernetes clusters. It allows users to manage and troubleshoot applications running in the cluster, as well as the cluster itself.
    - Container Resource Monitoring
      - Container Resource Monitoring records generic time-series metrics about containers in a central database, and provides a UI for browsing that data.
    - Cluster-level Logging
      - A Cluster-level logging mechanism is responsible for saving container logs to a central log store with search/browsing interface.

Kubernetes

- Pod
  - the basic building block of Kubernetes
  - a running process on your cluster
  - Type
    - Single container
    - Multiple containers



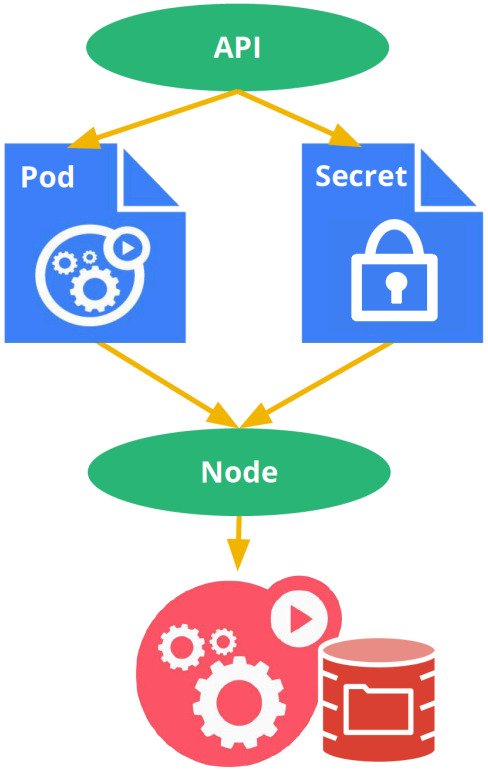

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
spec:
  containers:
  - name: myapp-container
    image: busybox
    command: ['sh', '-c', 'echo Hello Kubernetes! && sleep 3600']
```

Kubernetes

- Volume
  - Storage for container
- Namespace
  - Multi-user
  - Security & Access Policy
- Storage automatically attached to pod
  - Local scratch directories created on demand
  - Cloud block storage
  - GCE Persistent Disk
  - AWS Elastic Block Storage
  - Cluster storage
  - File: NFS, Gluster, Ceph
  - Block: iSCSI, Cinder, Ceph
  - Special volumes
    - Git repository
    - Secret

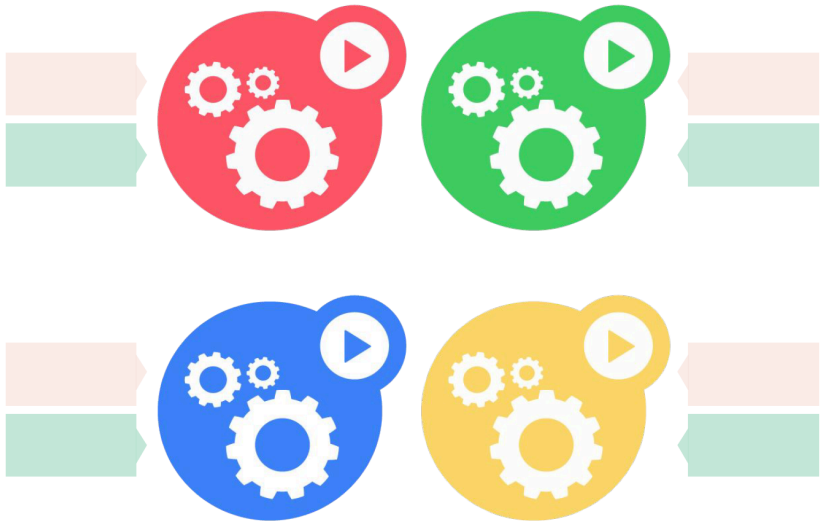
Kubernetes

- Secrets
  - grant a pod access to a secured something
    - secrets: credentials, tokens, passwords, ...
    - don’ t put them in the container image
  - Inject them as “virtual volumes” into Pod
    - not baked into images nor pod configs
    - kept in memory - never touches disk
    - not coupled to non-portable metadata API



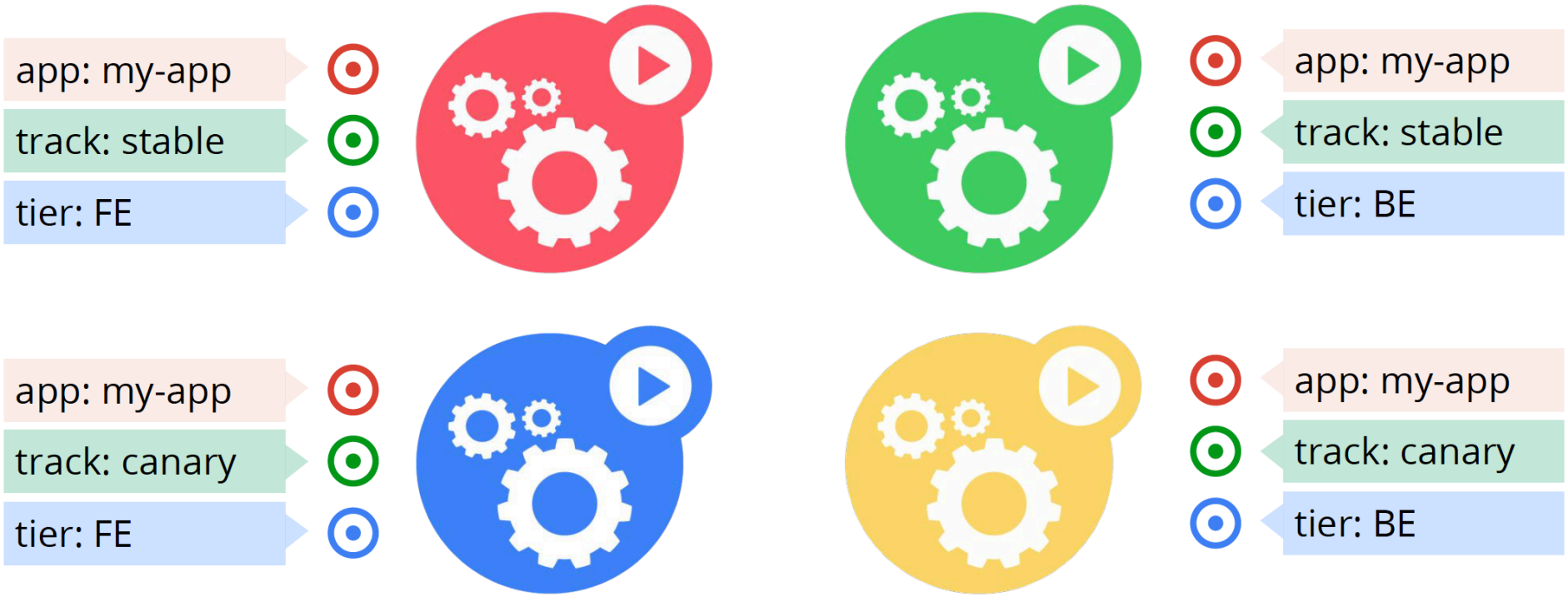
Kubernetes

- Labels
  - User-provided key-value attributes
  - Attached to any API object
  - Generally represent identity
  - Quarriable by selectors
    - think SQL 'select ... where ...'
  - The only grouping mechanism



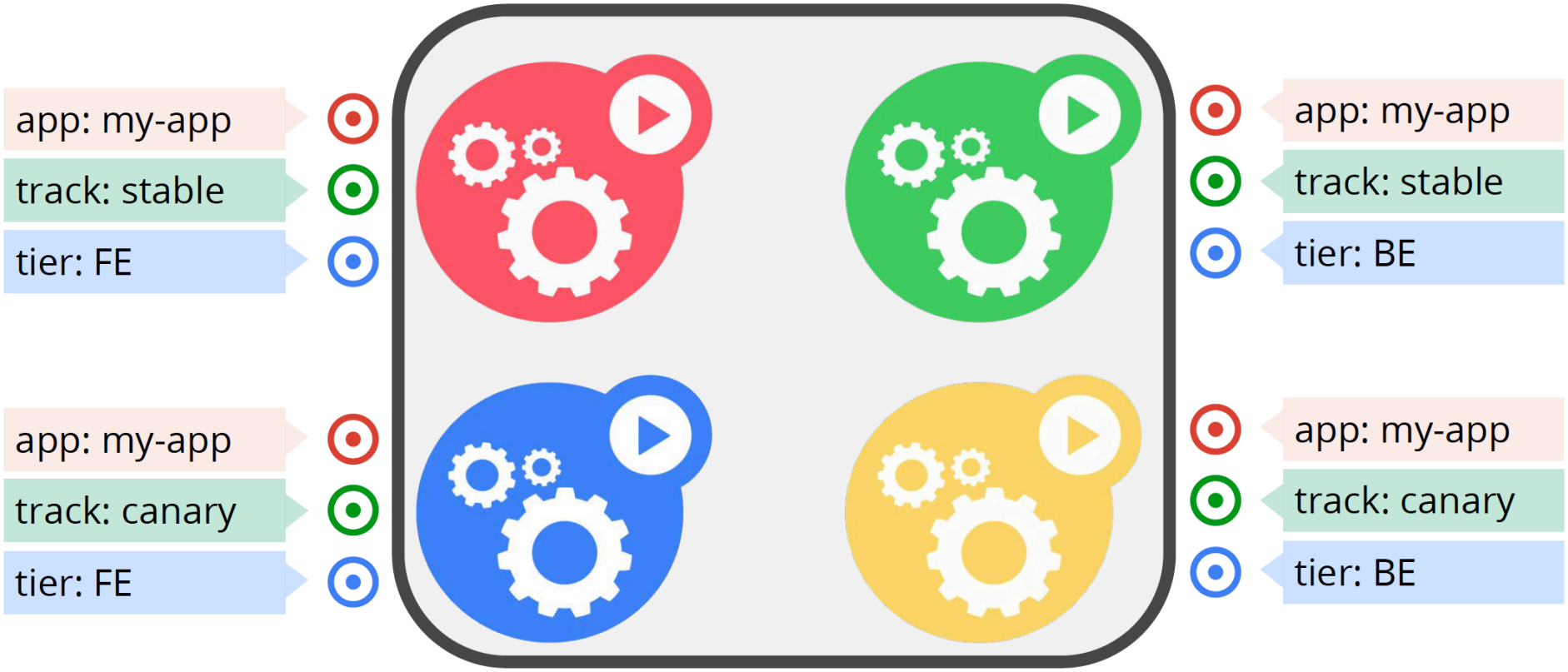
Kubernetes

- Selectors



Kubernetes

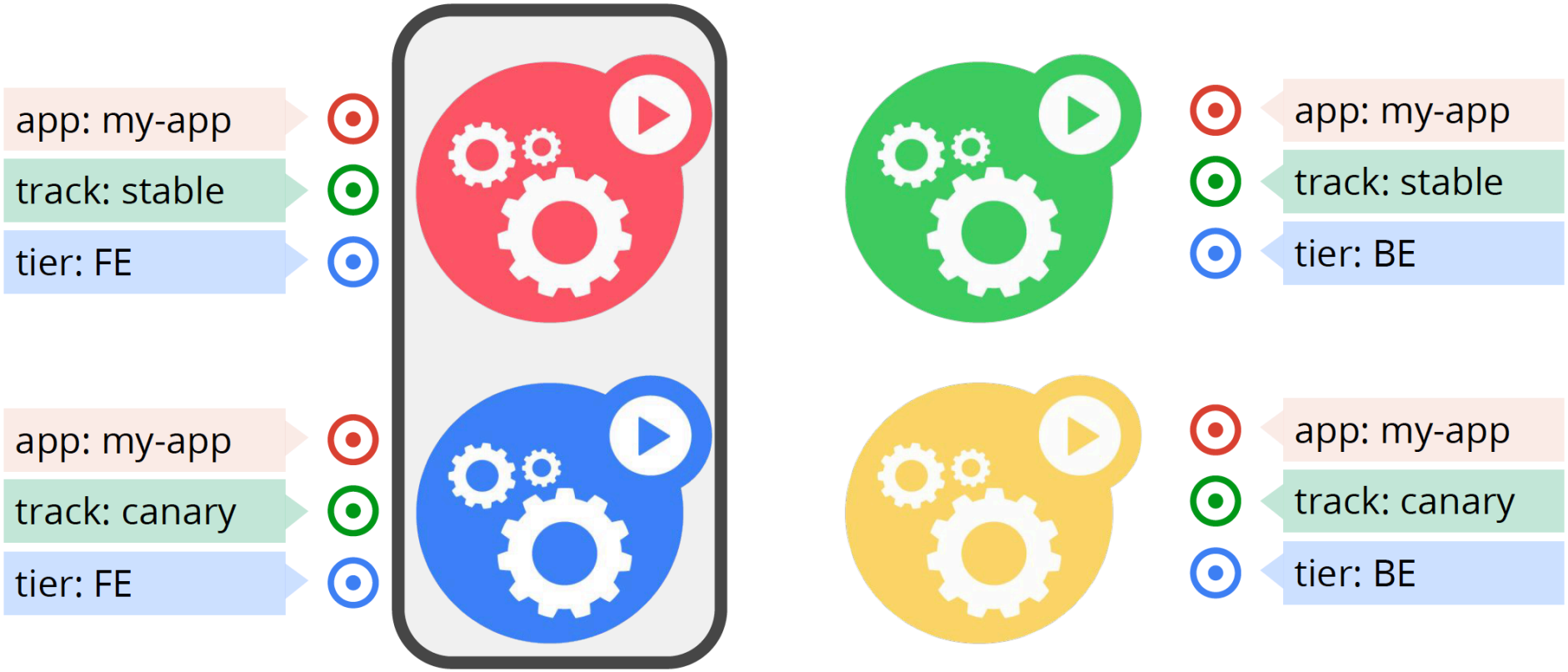
- Selectors



app = my-app

Kubernetes

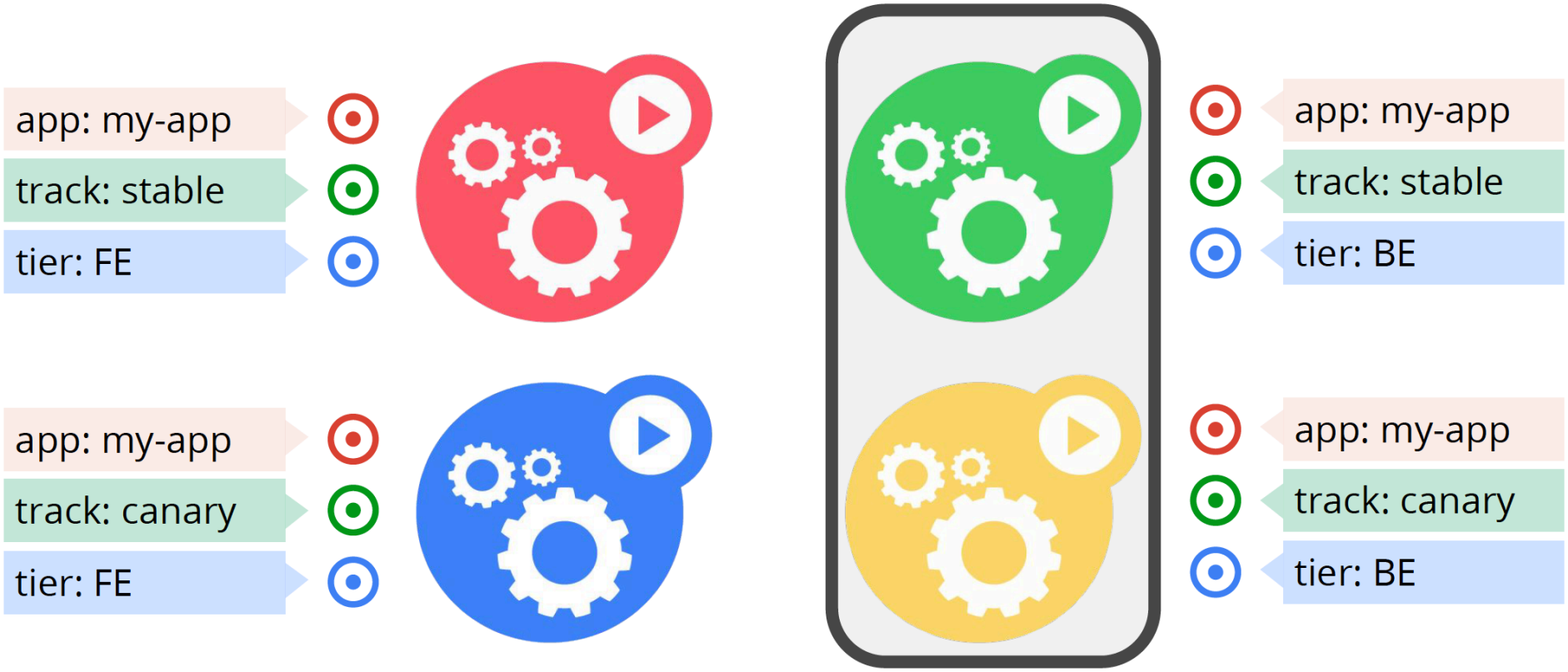
- Selectors



app = my-app, tier = FE

Kubernetes

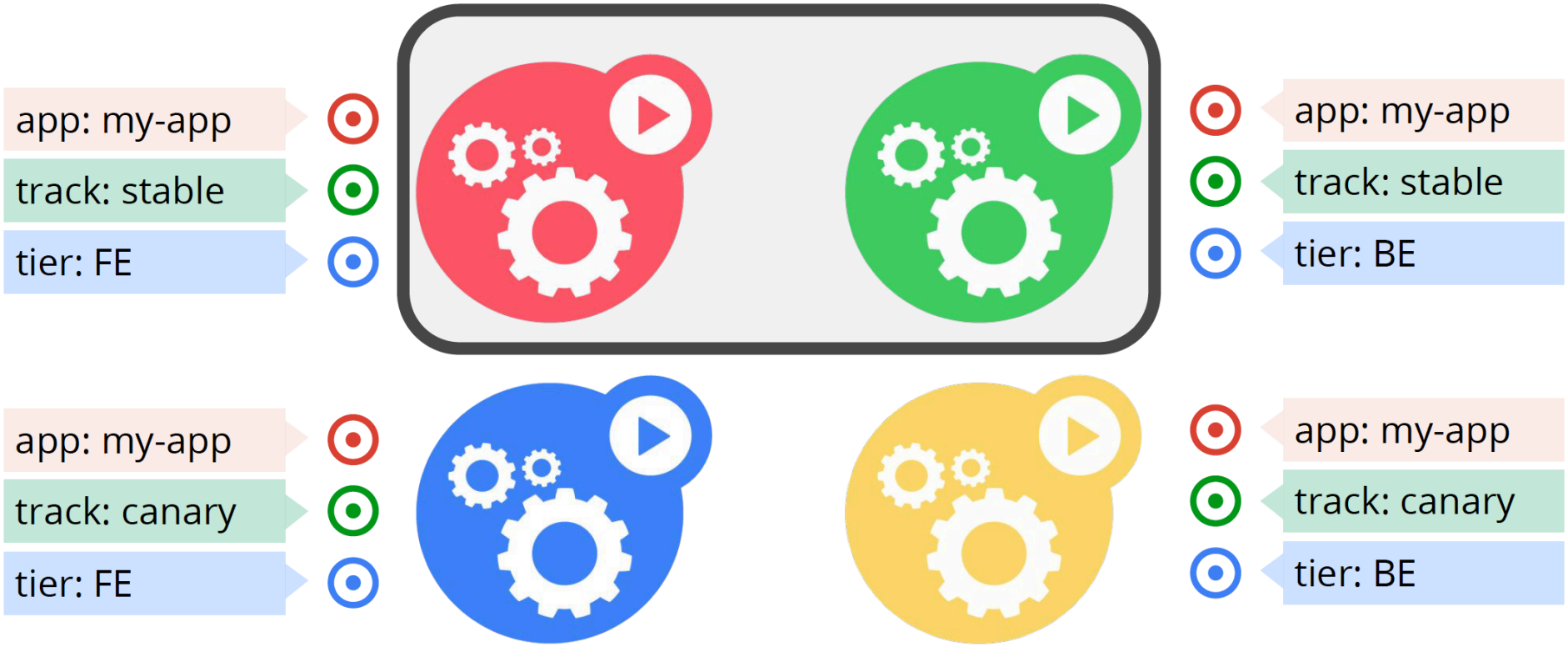
- Selectors



app = my-app, tier = BE

Kubernetes

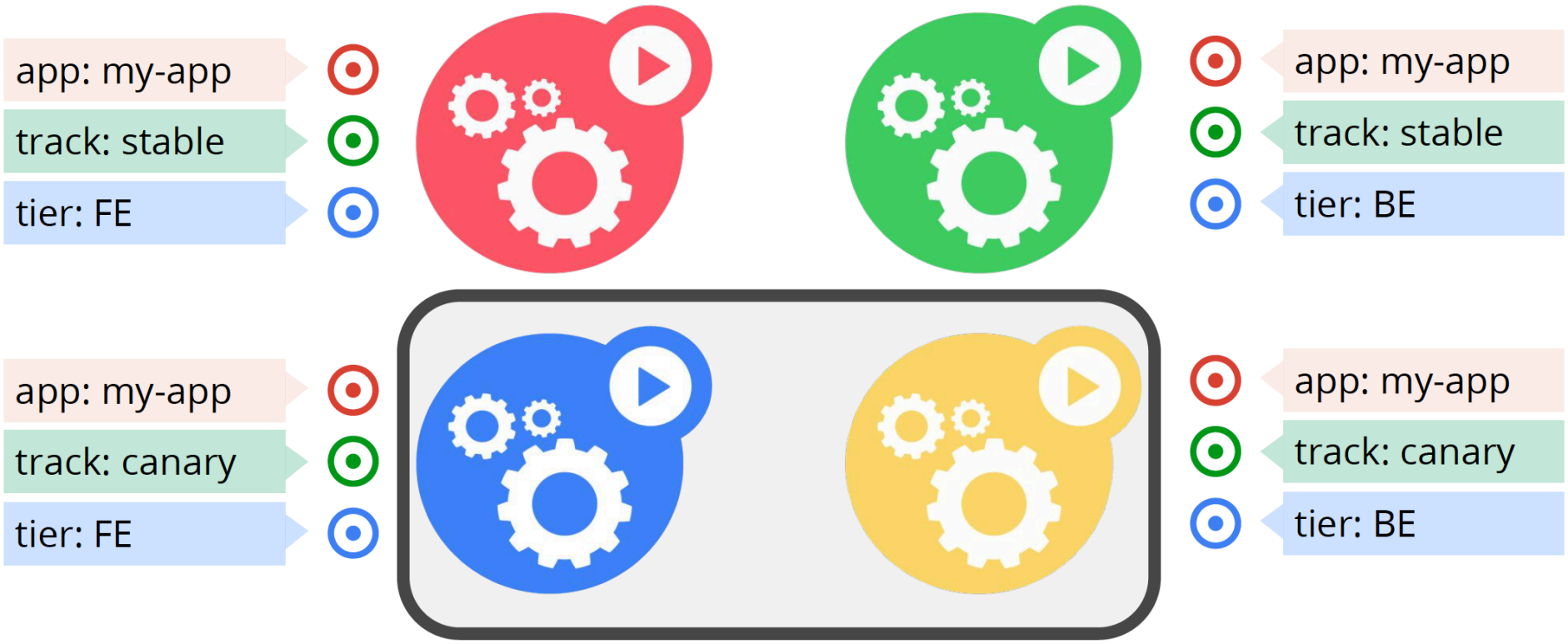
- Selectors



app = my-app, track = stable

Kubernetes

- Selectors



app = my-app, track = canary

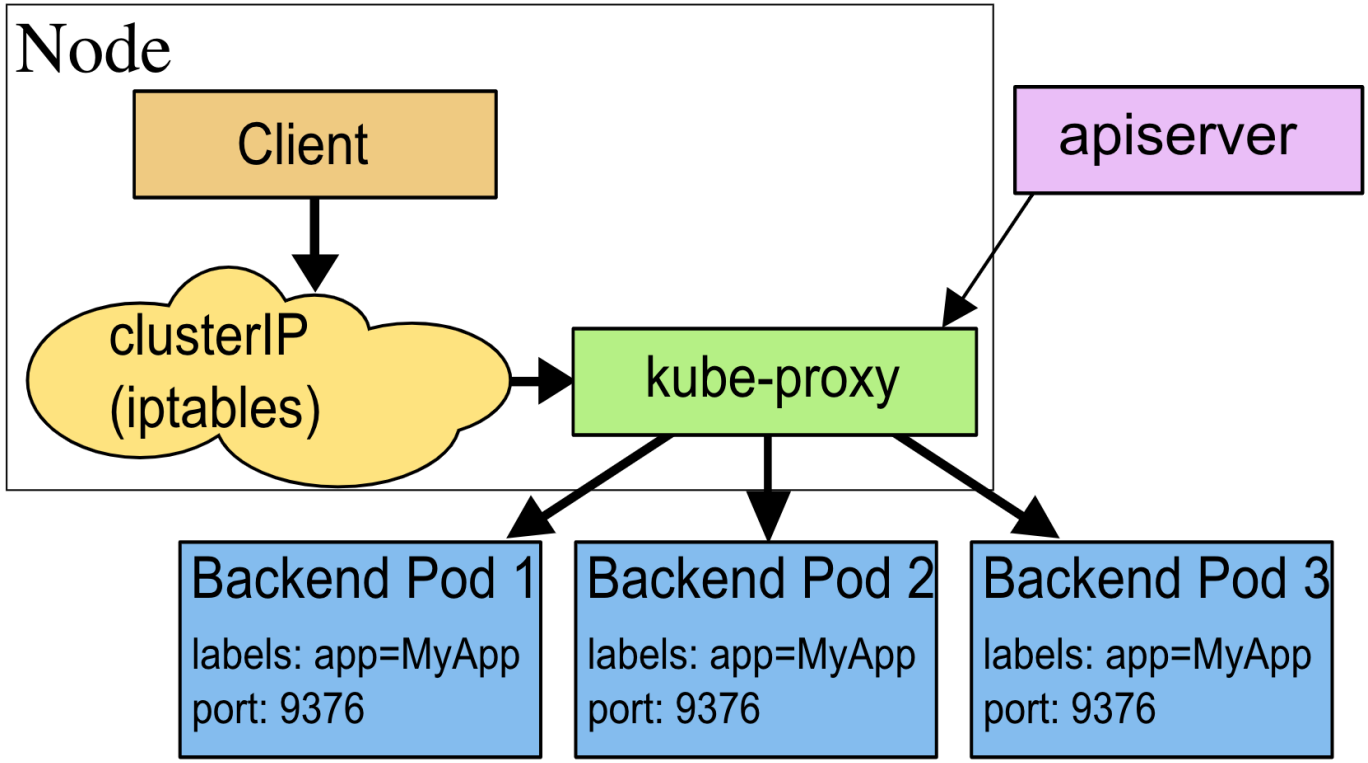
Kubernetes

- Objects
  - Service
    - an abstraction which defines a logical set of Pods and a policy by which to access them

```
kind: Service
apiVersion: v1
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
  - protocol: TCP
    port: 80
    targetPort: 9376
```

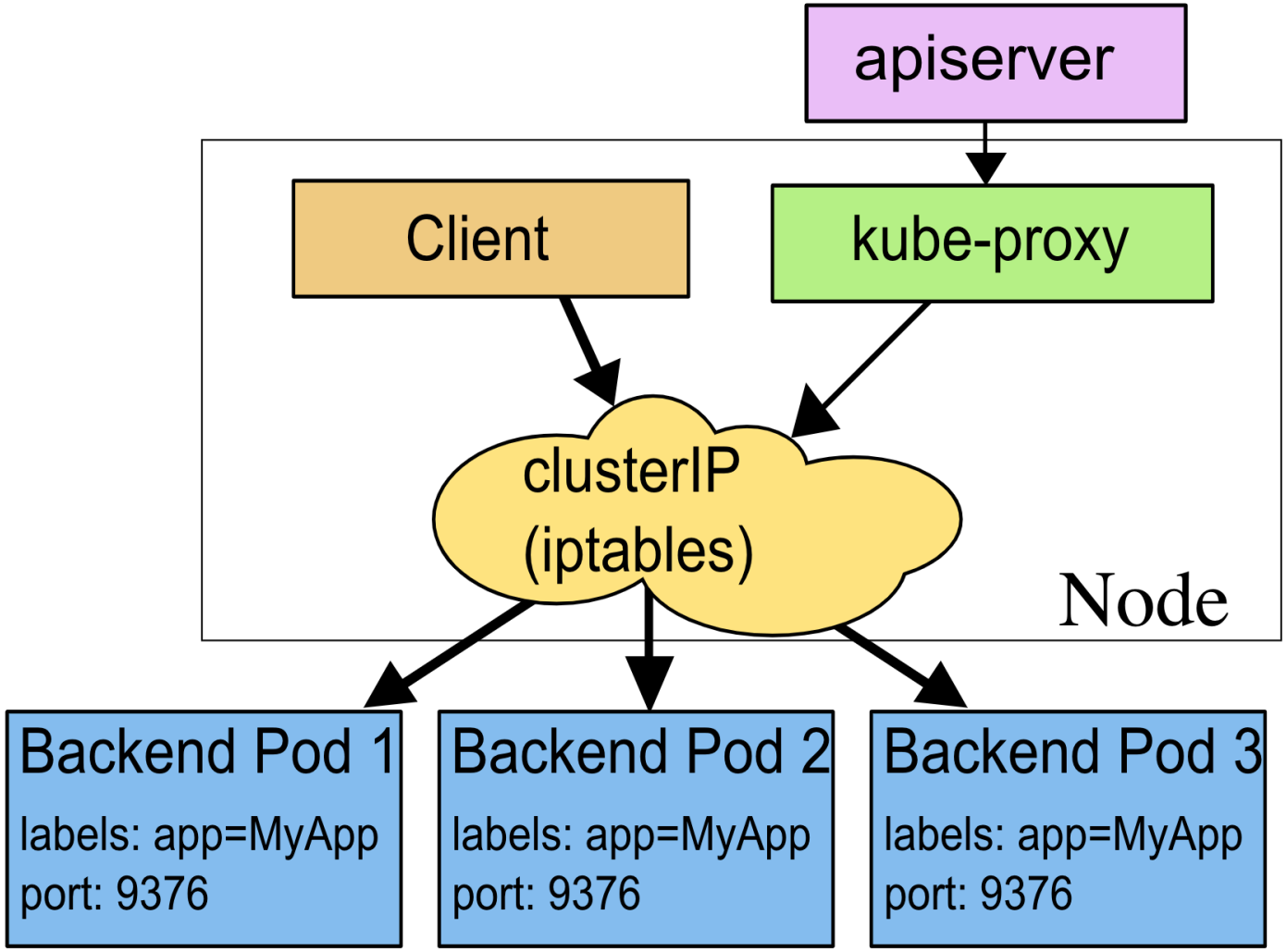
Kubernetes

- Objects
  - Service
    - Proxy
      - Userspace



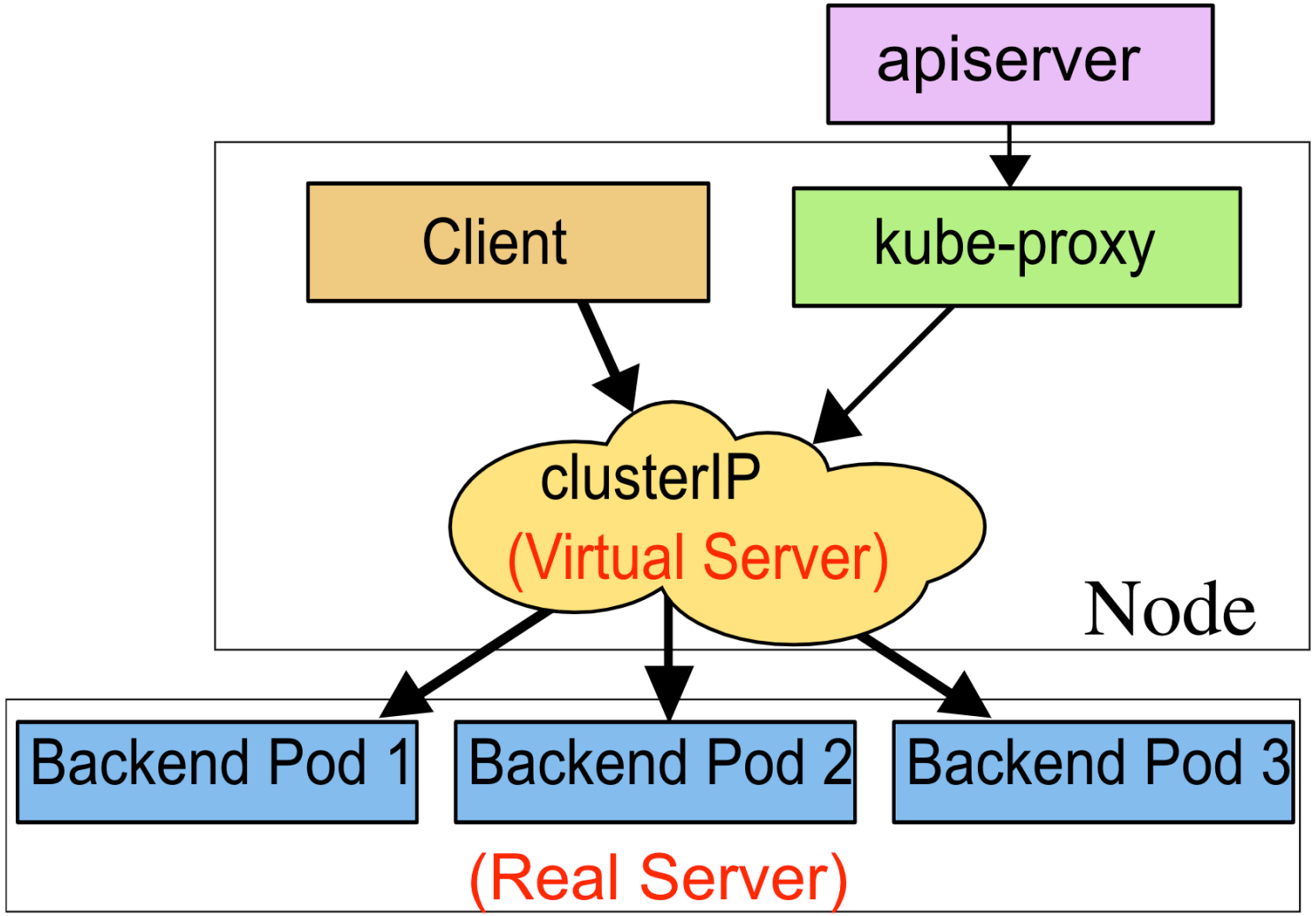
Kubernetes

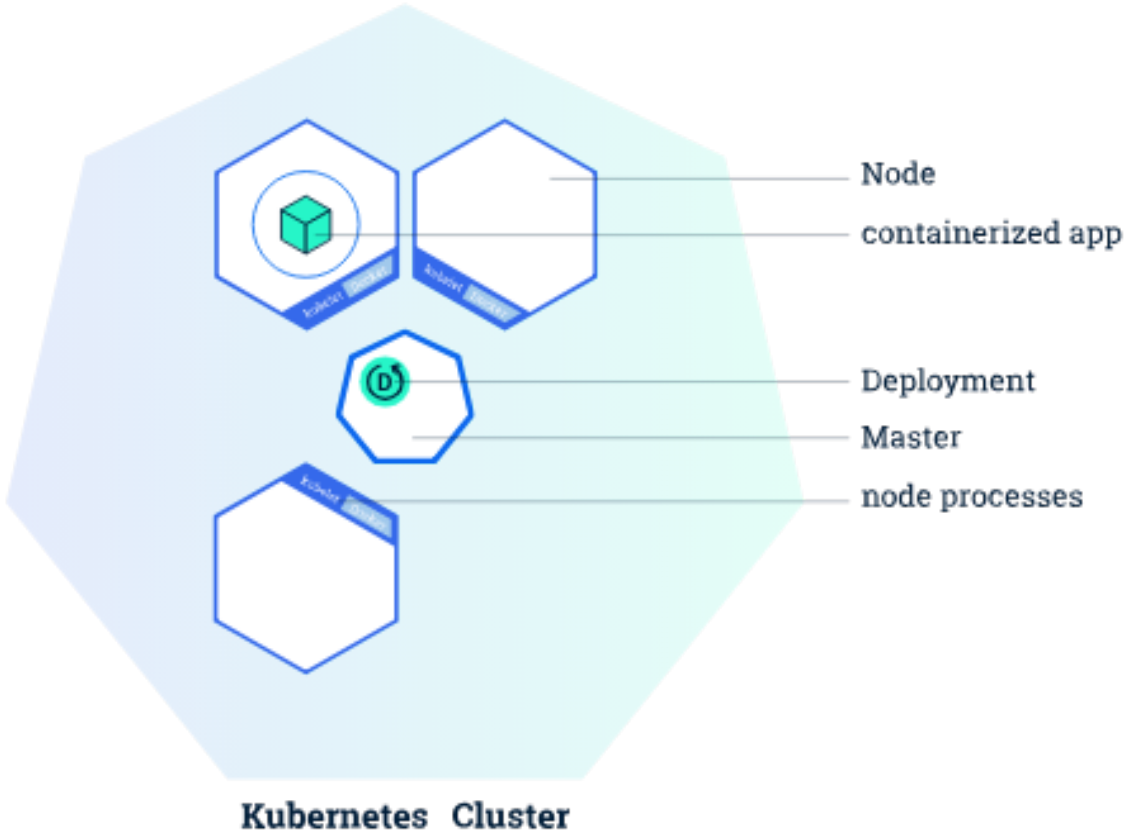
- Objects
  - Service
    - Proxy
      - iptables

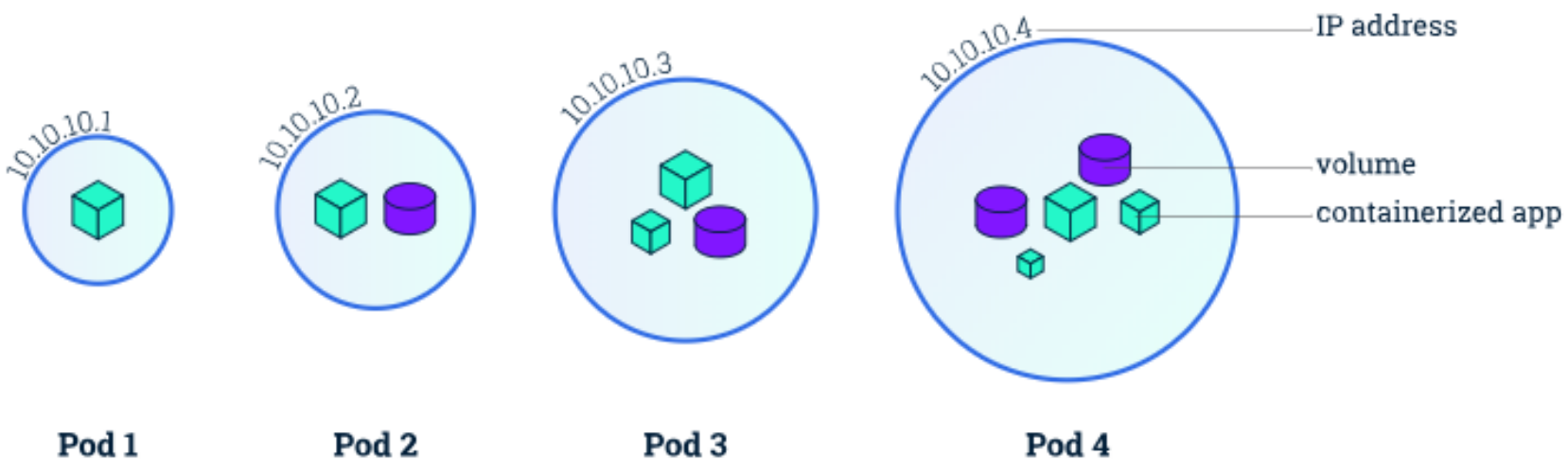


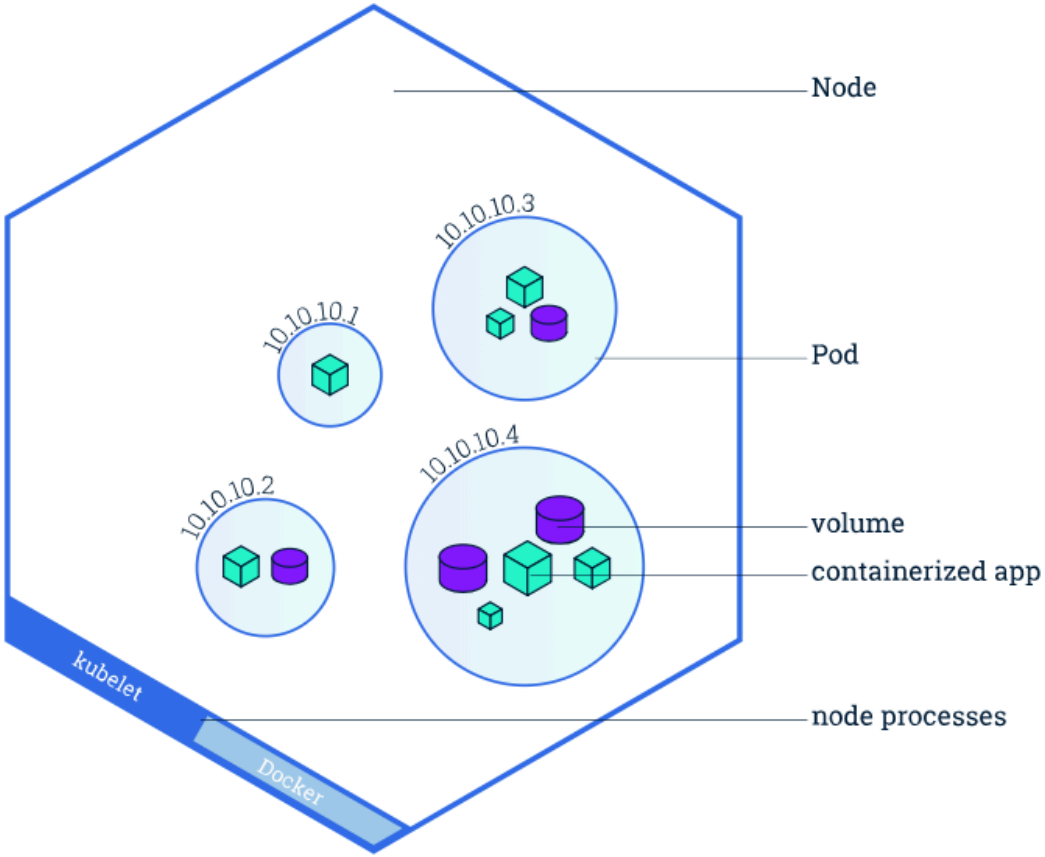
Kubernetes

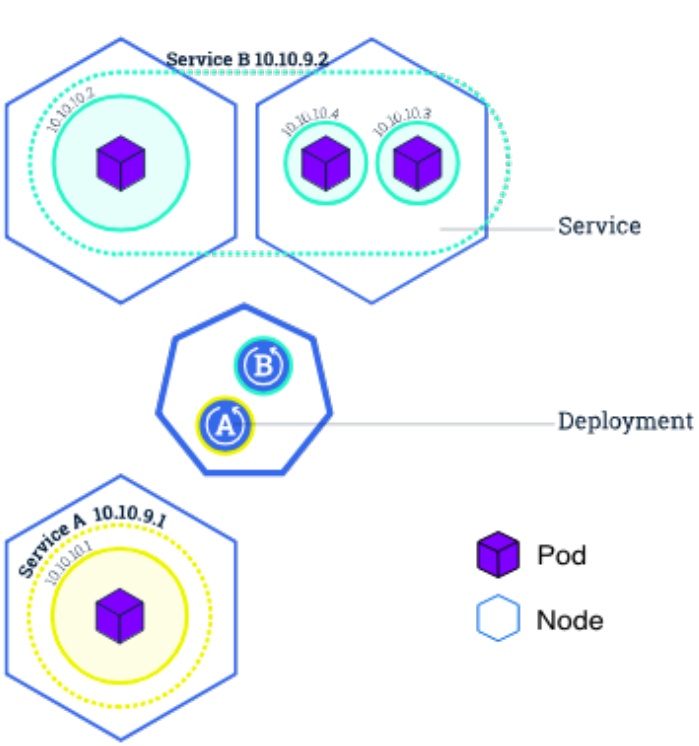
- Objects
  - Service
    - Proxy
      - ipvs

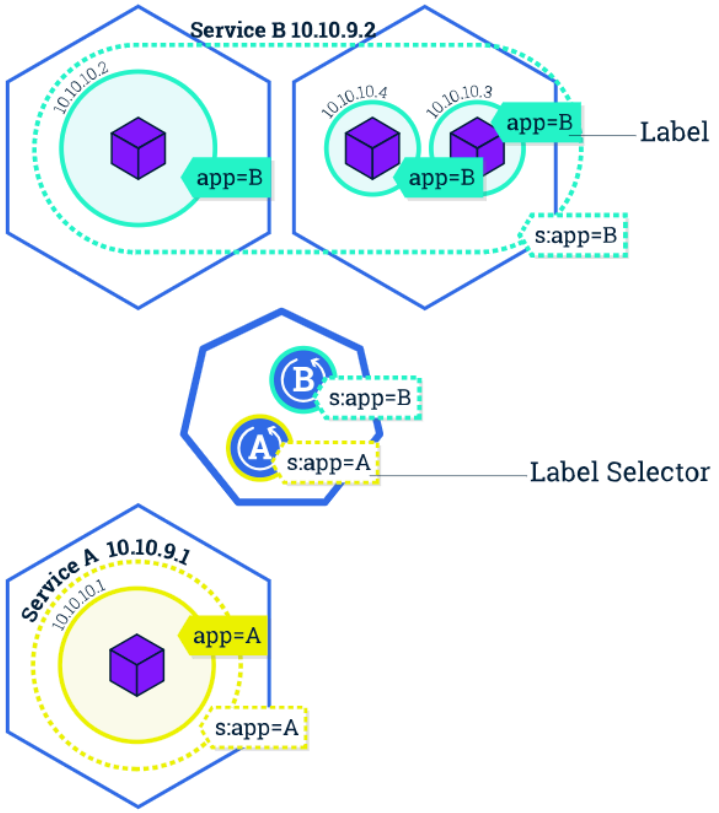



Node
containerized app
Deployment
Master
node processes
Kubernetes Cluster

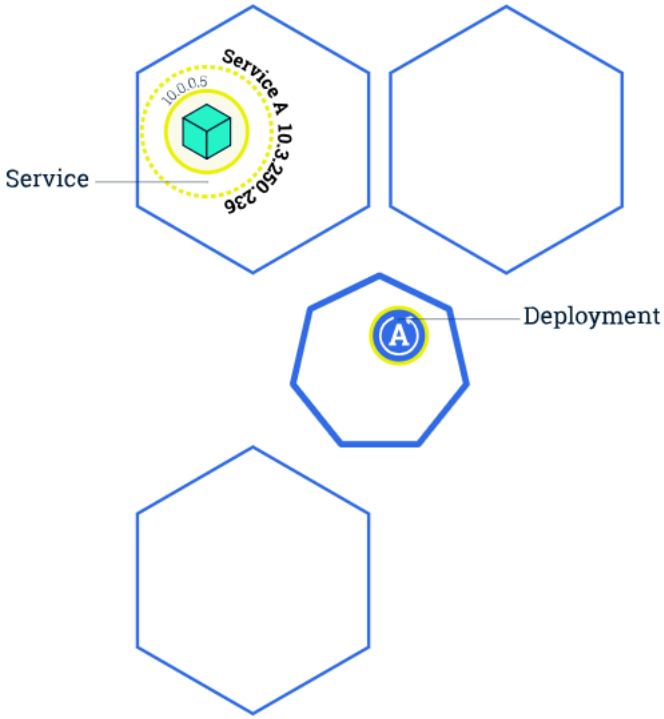

10.10.10.1
10.10.10.2
10.10.10.3
10.10.10.4
IP address
volume
containerized app
Pod 1
Pod 2
Pod 3
Pod 4

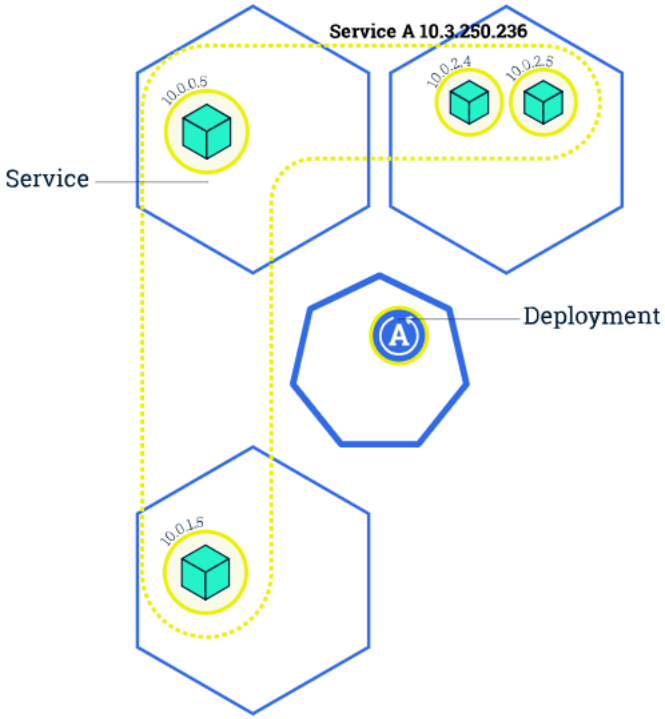

Node
10.10.10.3
10.10.10.1
Pod
10.10.10.4
10.10.10.2
volume
containerized app
kubelet
Docker
node processes


Service B 10.10.9.2
10.10.10.2
10.10.10.4
10.10.10.3
Service
B
A
Deployment
Service A 10.10.9.1
10.10.10.1
Pod
Node

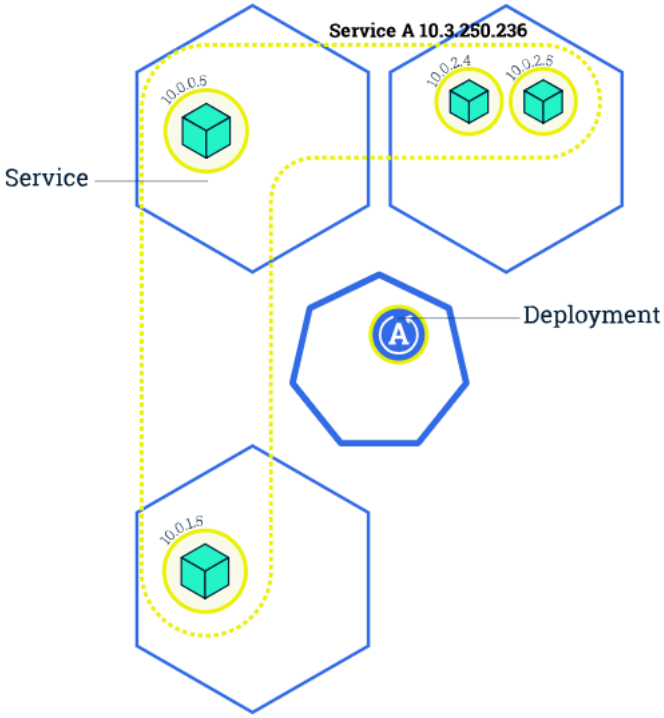

Service B 10.10.9.2
10.10.10.2
app=B
10.10.10.4
10.10.10.3
app=B
app=B
Label
s:app=B
B
s:app=B
A
s:app=A
Label Selector
Service A 10.10.9.1
10.10.10.1
app=A
s:app=A

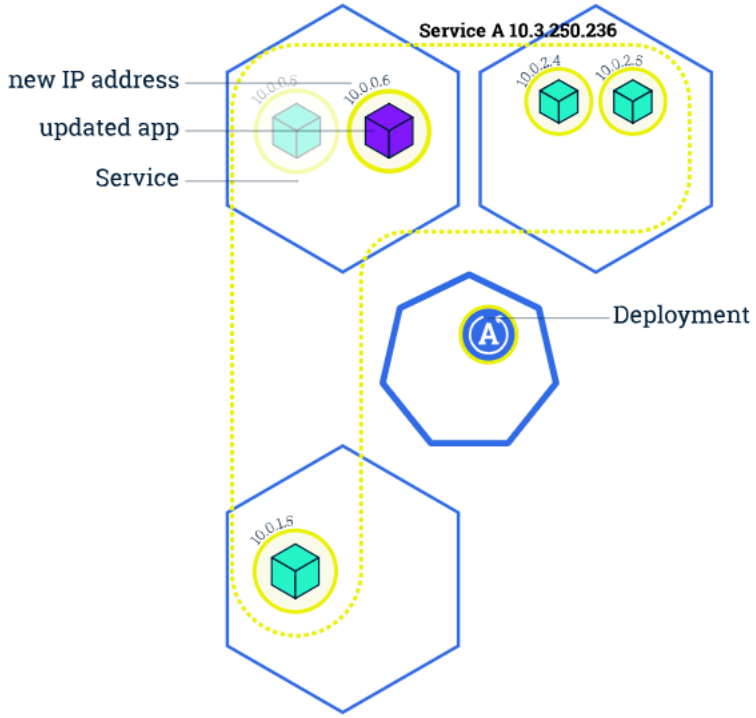
Scaling


Service A 10.3.250.236
10.0.0.5
Service
Deployment


Service A 10.3.250.236
10.0.0.5
10.0.2.4
10.0.2.5
Service
Deployment
10.0.1.5

Rolling Update





Rolling Update