Kubernetes labor

Kiberfizikai rendszerek

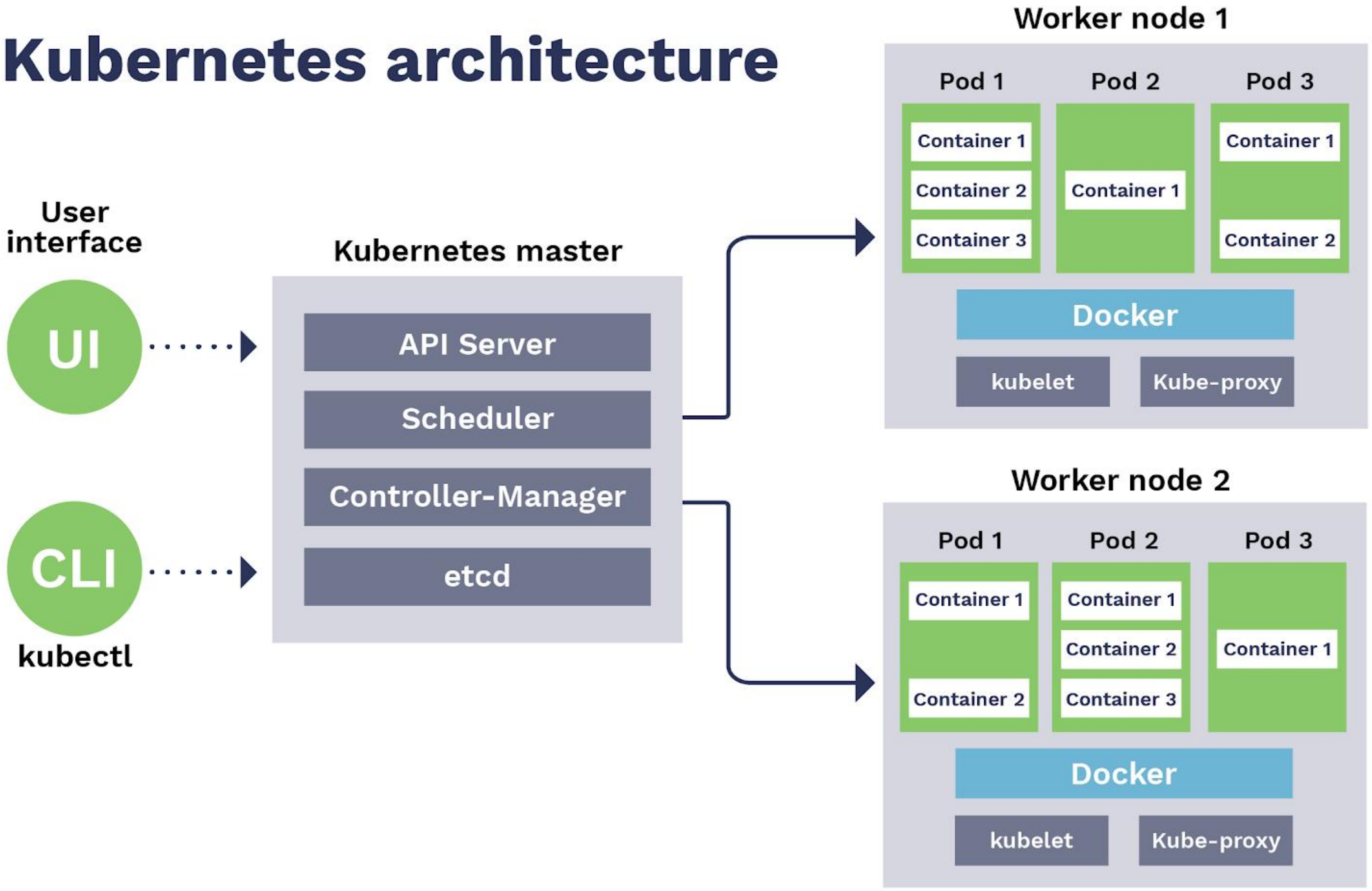
Budapesti Műszaki és Gazdaságtudományi Egyetem
Hibatűrő Rendszerek Kutatócsoport

Kubernetes

- Open source
- Platform independent
- Container orchestration system

KUBERNETES ARCHITECTURE

Kubernetes architecture

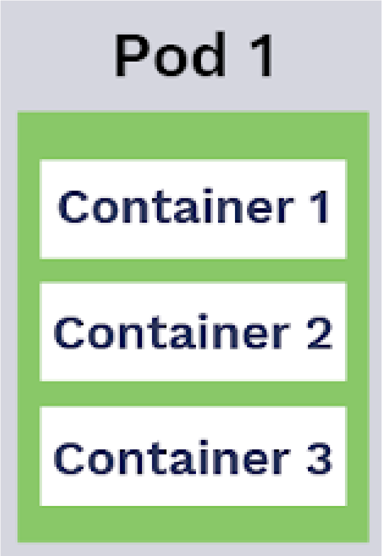


Kubernetes architecture

- Cluster
  - Worker Node
    - Pod
    - Deployment
    - Service
  - Master Server

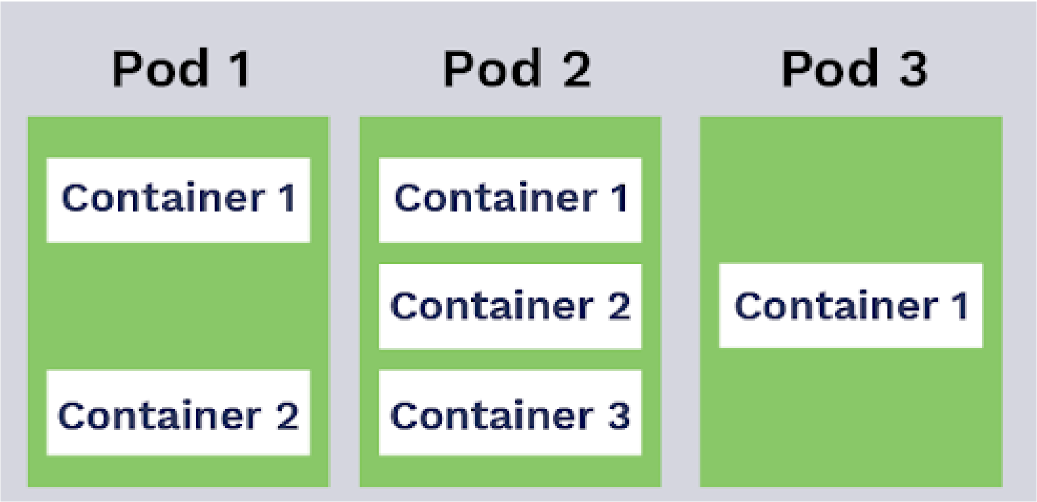
Kubernetes architecture - Pod

- Group of containers
- Smallest managed unit
- Pods have a single IP address
- Containers in a Pod have ports assigned to them



Kubernetes architecture - Deployment

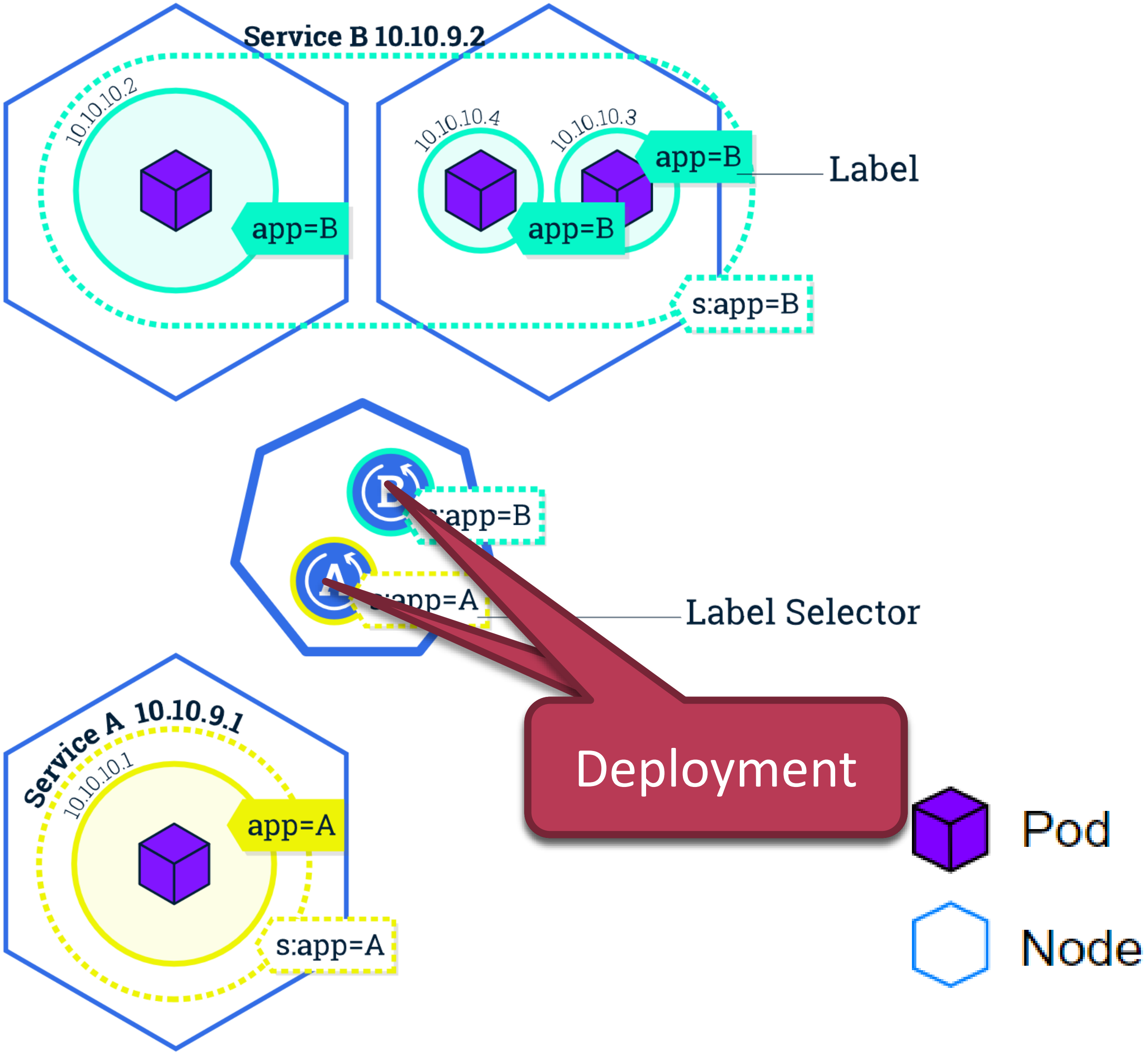
- Represents an application
- Defines the :
  - Pods (application components)
  - Number of Pod replicas
  - Pod update strategy
  - Pod restart policy



Kubernetes architecture – Service

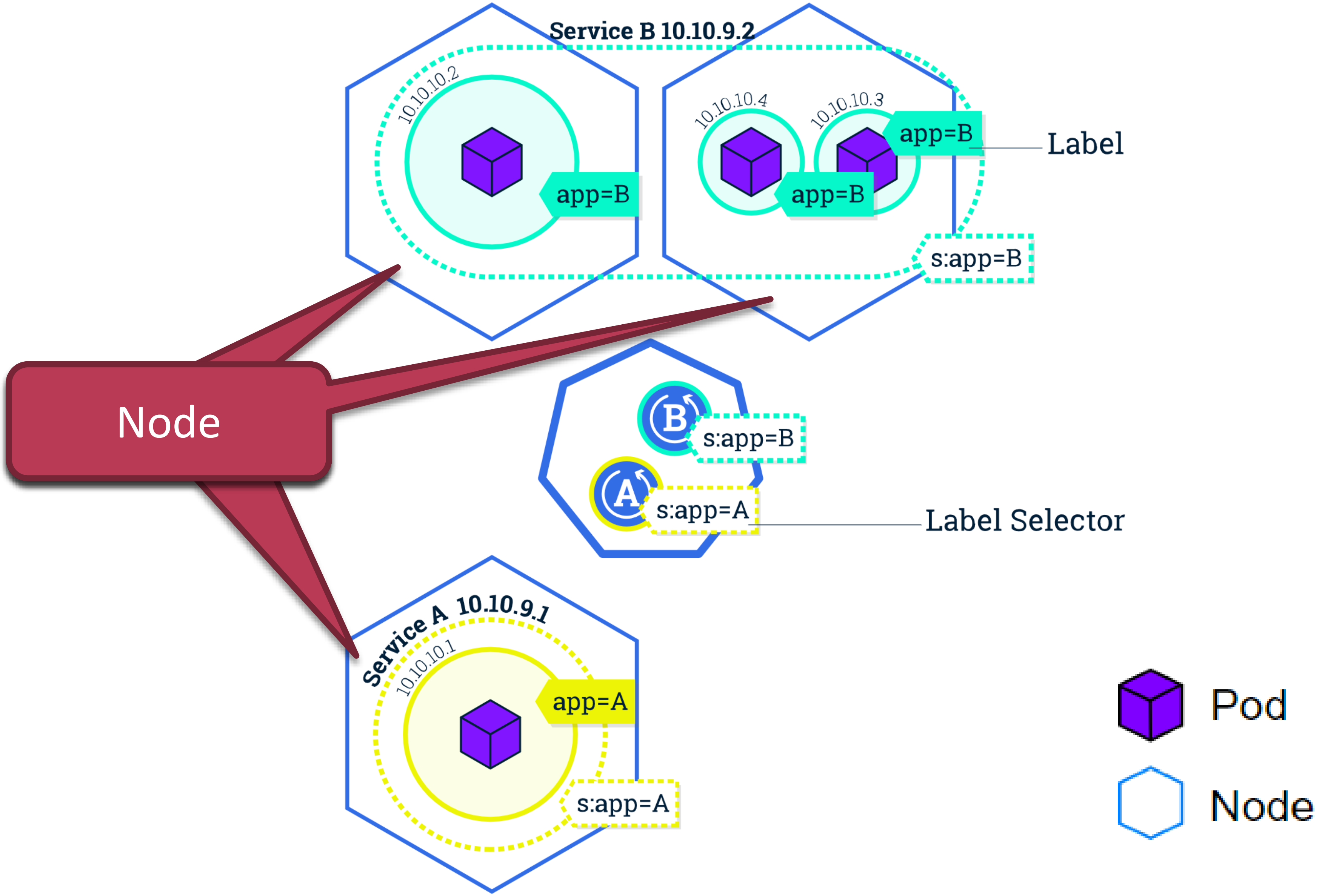
- Abstraction over the Pods
- Interface to interact with
- The Pods IP address and their number can change during their lifecycle
- The Service masks these changes, and ensures that, to the outside network, everything appears to be unchanged

Kubernetes architecture - Service

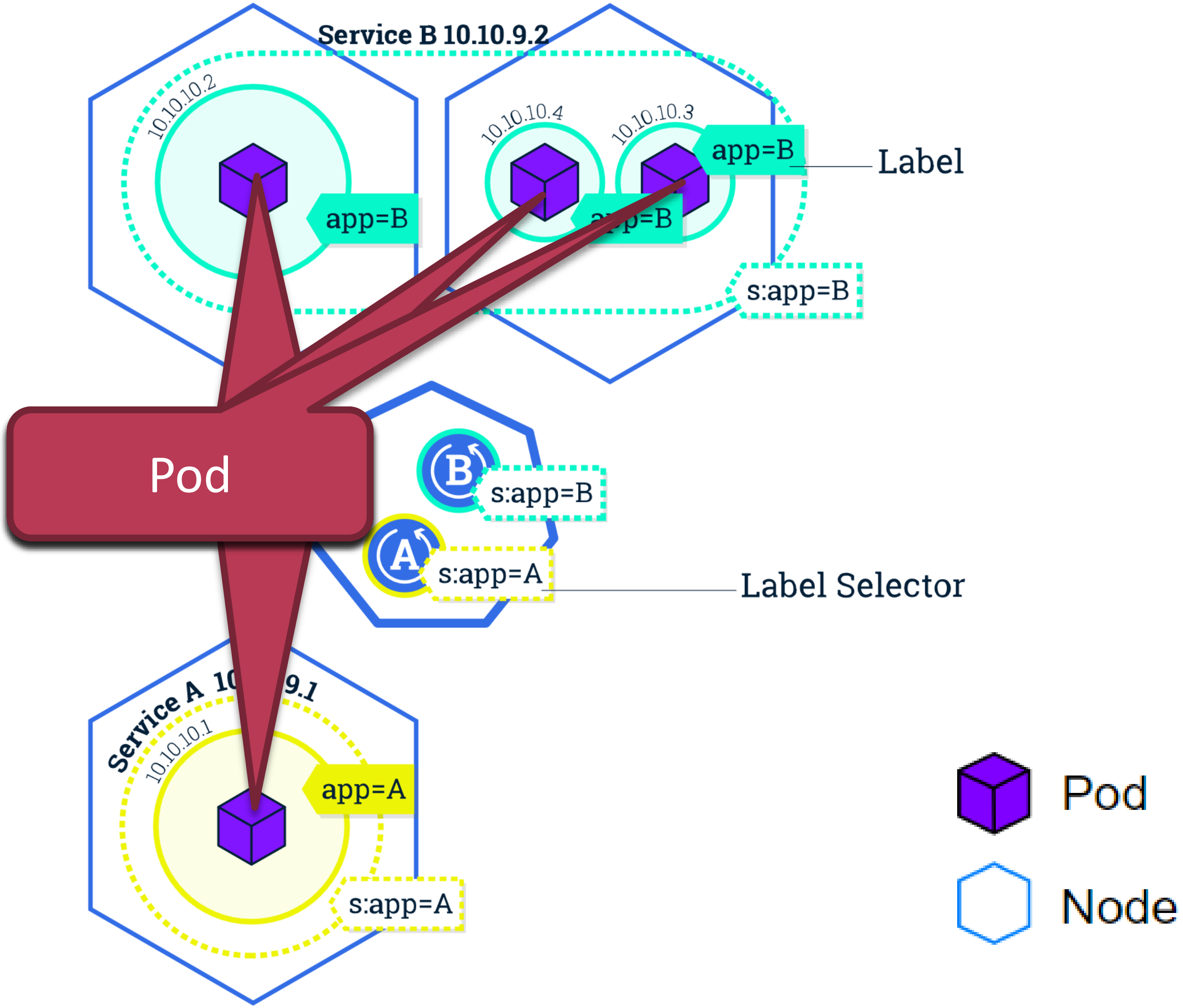


Deployment

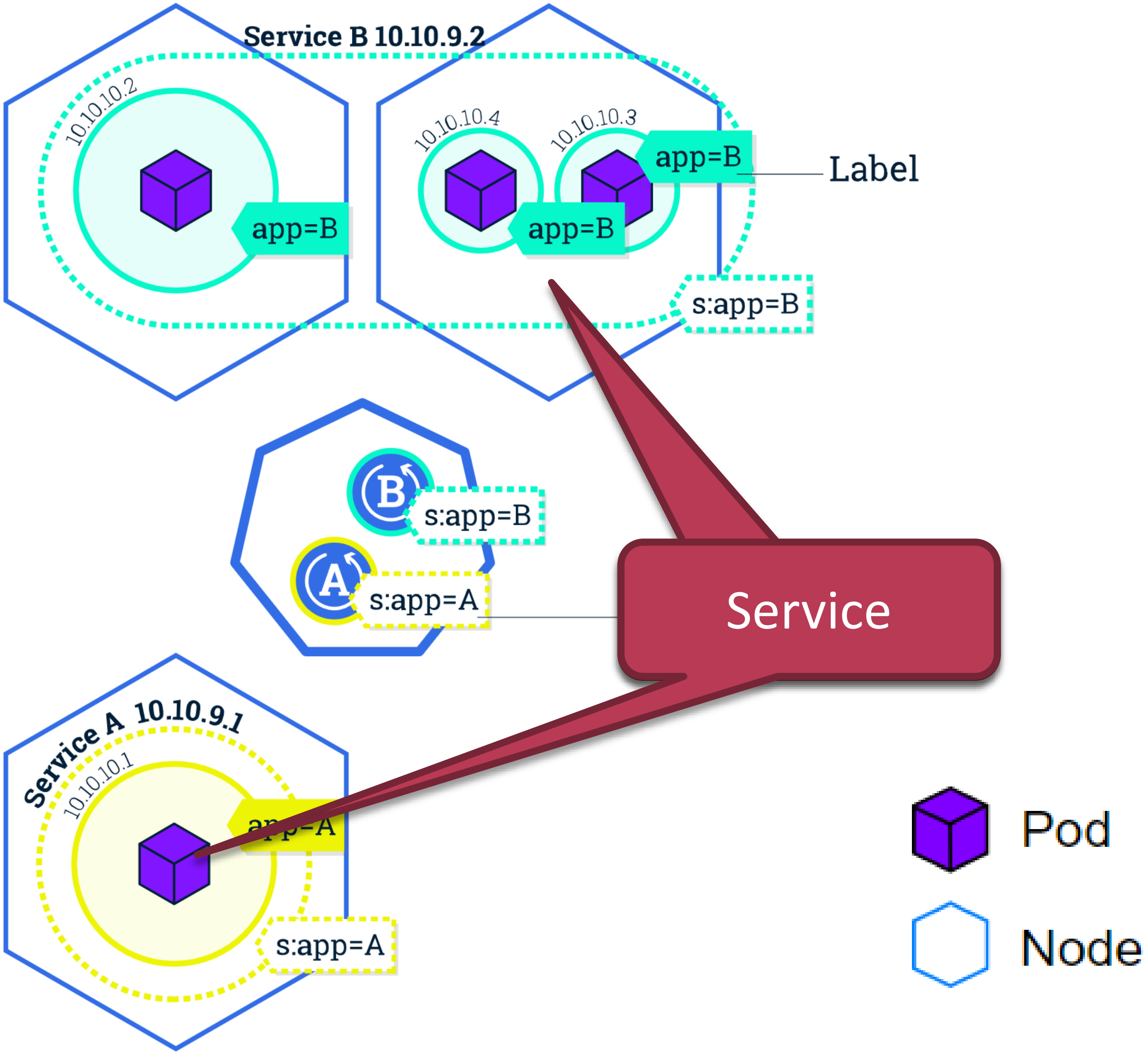
Kubernetes architecture - Service



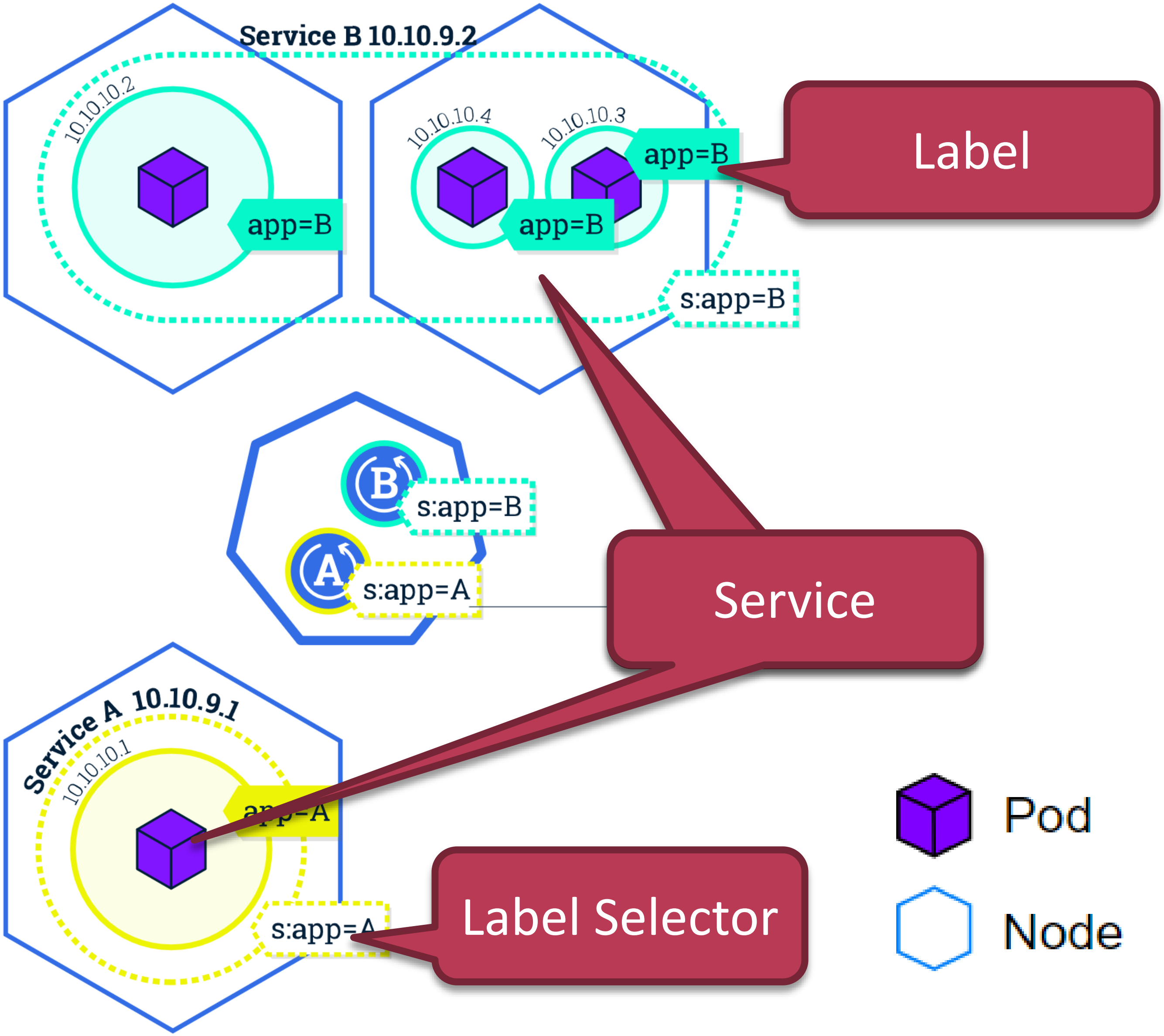
Kubernetes architecture - Service



Kubernetes architecture - Service



Kubernetes architecture - Service



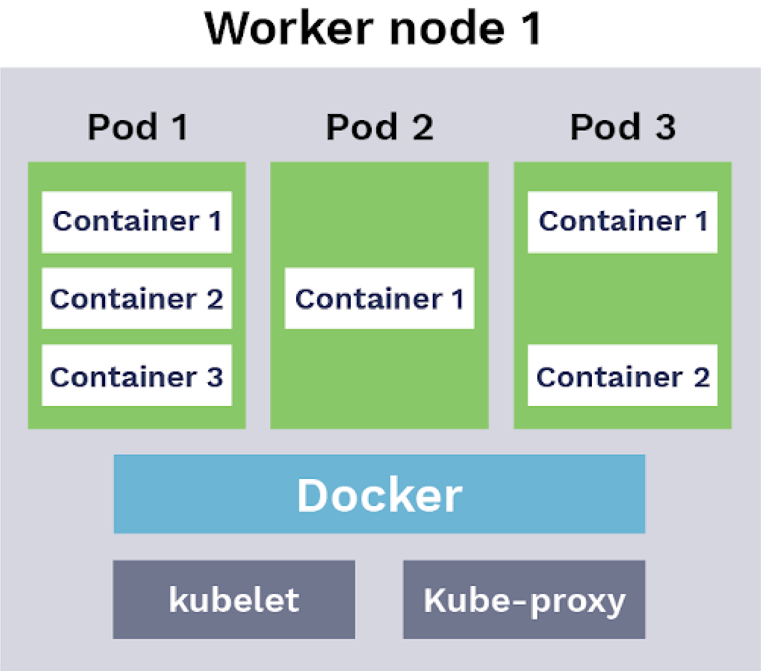
Label

Service

Label Selector

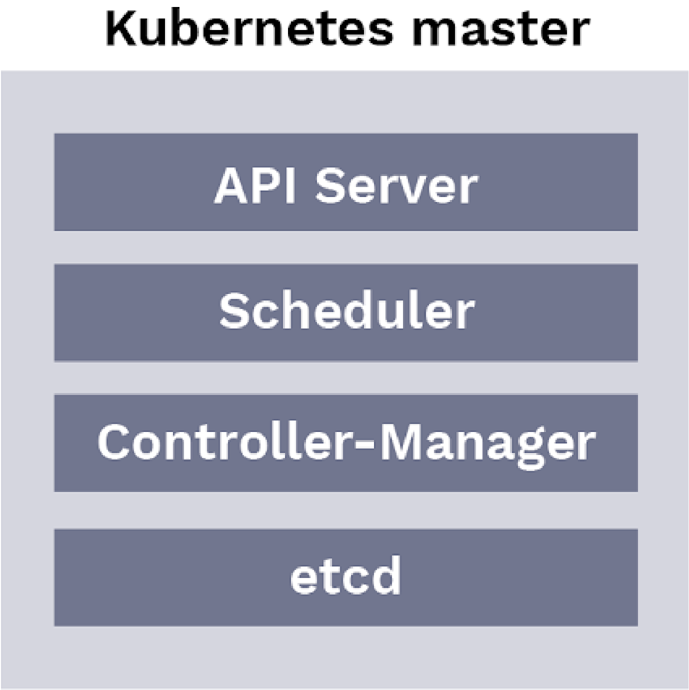
Kubernetes architecture – Worker Node

- Manages and runs Pods
- Collects entire Pods that function together
- Components:
  - Kubelet: tracks the state of the Pods
  - Kube-Proxy: routes traffic coming into a node from the service, forwards requests to the correct container



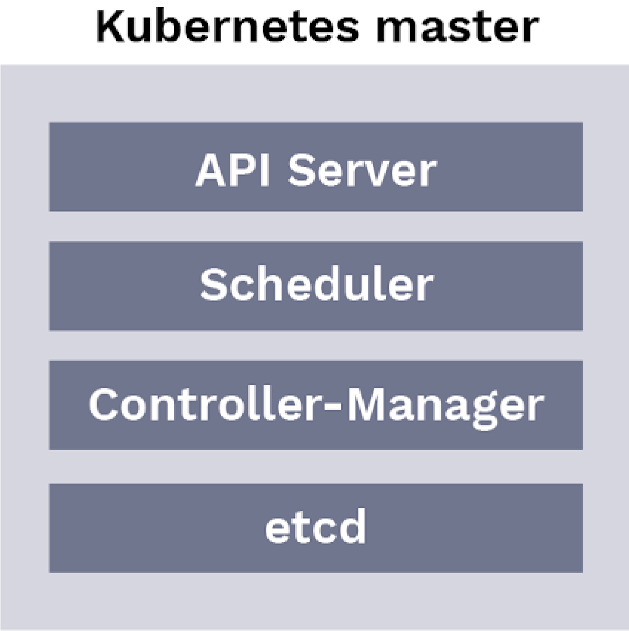
Kubernetes architecture – Master Server

- Main entry point for administrators and users to manage the nodes
- Assigning work to the nodes
- Components:
  - API Server: REST interface, all operations are executed by communicating with the endpoints provided by it



Kubernetes architecture – Master Server

- Components:
  - Scheduler: assigning work to the nodes, watch over resource capacity, ensures that a worker node's performance is within an appropriate threshold
  - Controller-Manager: oversees controllers which respond to events
  - Etcd: distributed key-value store, used to share information about the overall state of a cluster



DEPLOYMENT STRATEGIES

Deployment strategies

- Defines how your deployment gets updated

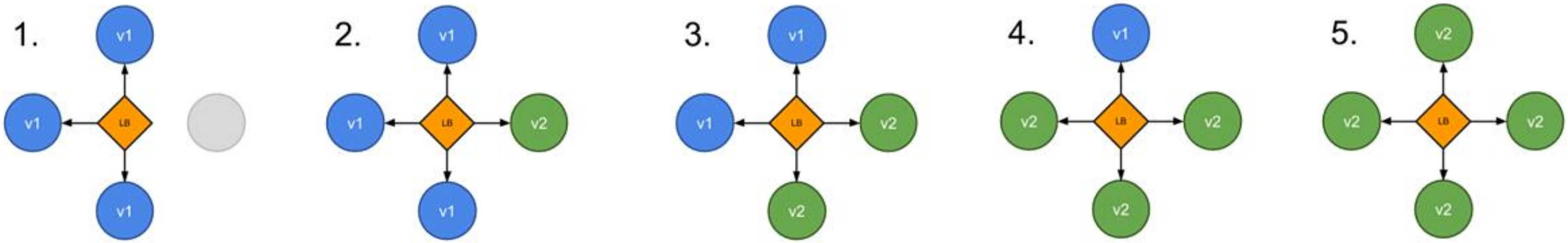
- Strategies:
  - Recreate
  - RollingUpdate
  - Blue/Green
  - Canary
  - A/B testing

Deployment strategies

- Recreate:
  - Terminate the old version and release the new one
  - Application state entirely renewed
  - Downtime that depends on both shutdown and boot duration of the application

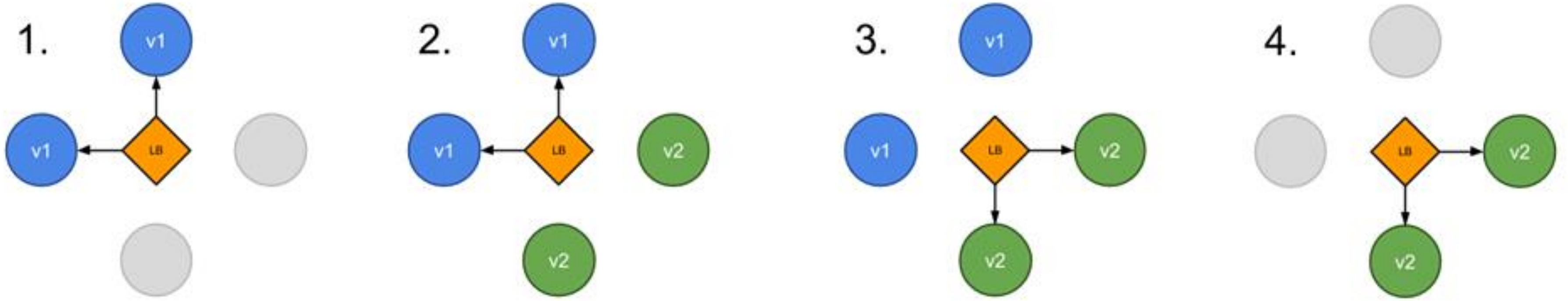
Deployment strategies

- Rolling update:
  - New version replica is created
  - Then the number of replicas of the old version is decreased and the new version is increased
  - No downtime, convenient for stateful applications
  - We have to support multiple versions simultaniously



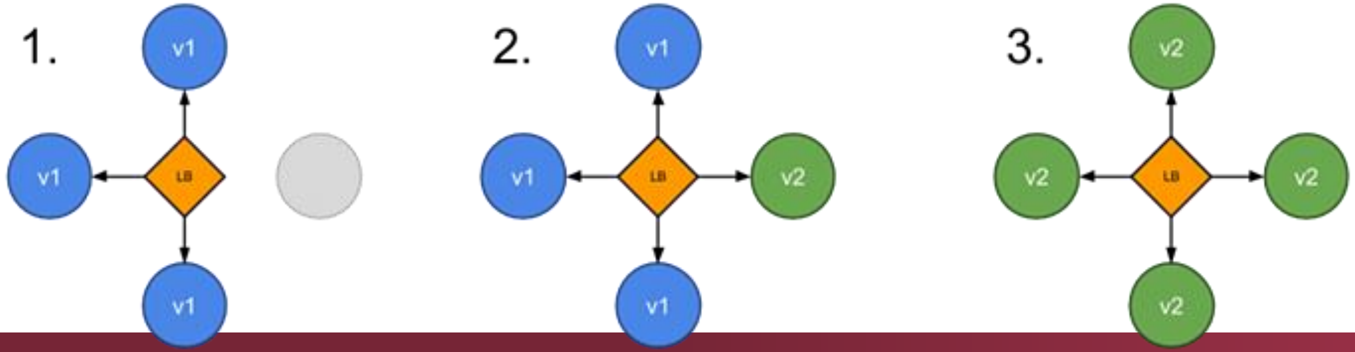
Deployment strategies

- Blue/Green
  - The new "green" version of the application is deployed alongside the old "blue" version
  - After testing we update the Service to use the „green” version
  - Instant rollout/rollback, no downtime
  - Double resources, testing, handling stateful applications



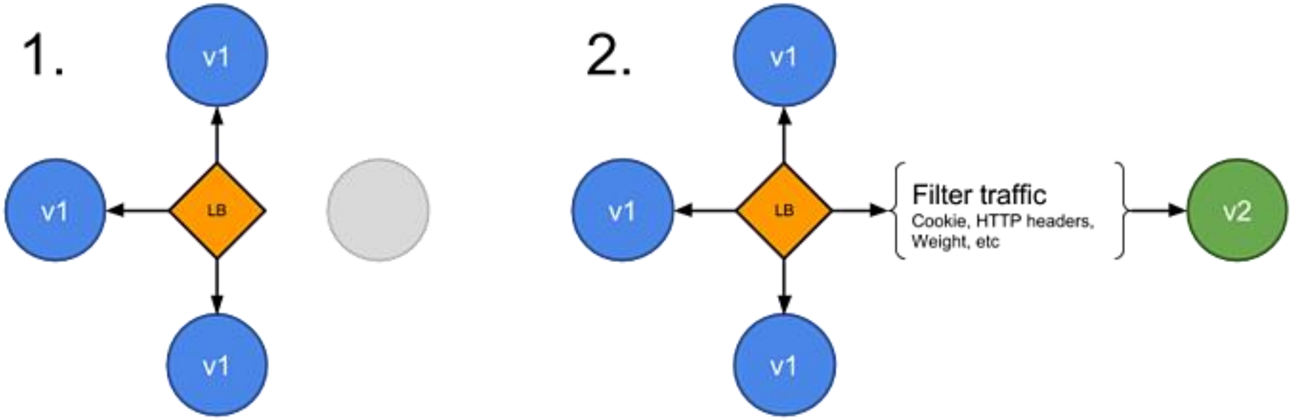
Deployment strategies

- Canary:
  - One replica of the new version is released alongside the old versions
  - Then if no error is detected, scale up the number of replicas of the new version and delete the old versions
  - Testing using the subset of cunsumers
  - Convenient for error rate and performance monitoring
  - Fine tuned traffic distribution can be expensive



Deployment strategies

- A/B testing:
  - Based on Canary
  - Testing using a target group of consumers
  - Intelligent load balancer
  - Control over the traffic distribution
  - Several versions run in parallel
  - Hard to troubleshoot errors



POD RESTART POLICY

Pod restart policy

- Defines how the Pod restarts
- Applies to all the containers of a Pod
- A container of a Pod can exit in the event of:
  - Succesfully completing its task
  - Failure

- RestartPolicies:
  - Always
  - OnFailure
  - Never

Pod restart policy

- Always
  - Default
  - Always restarts the container if it exits
- OnFailure
  - Only restarts the container if it exited with Failure
- Never
  - Never restarts the containers