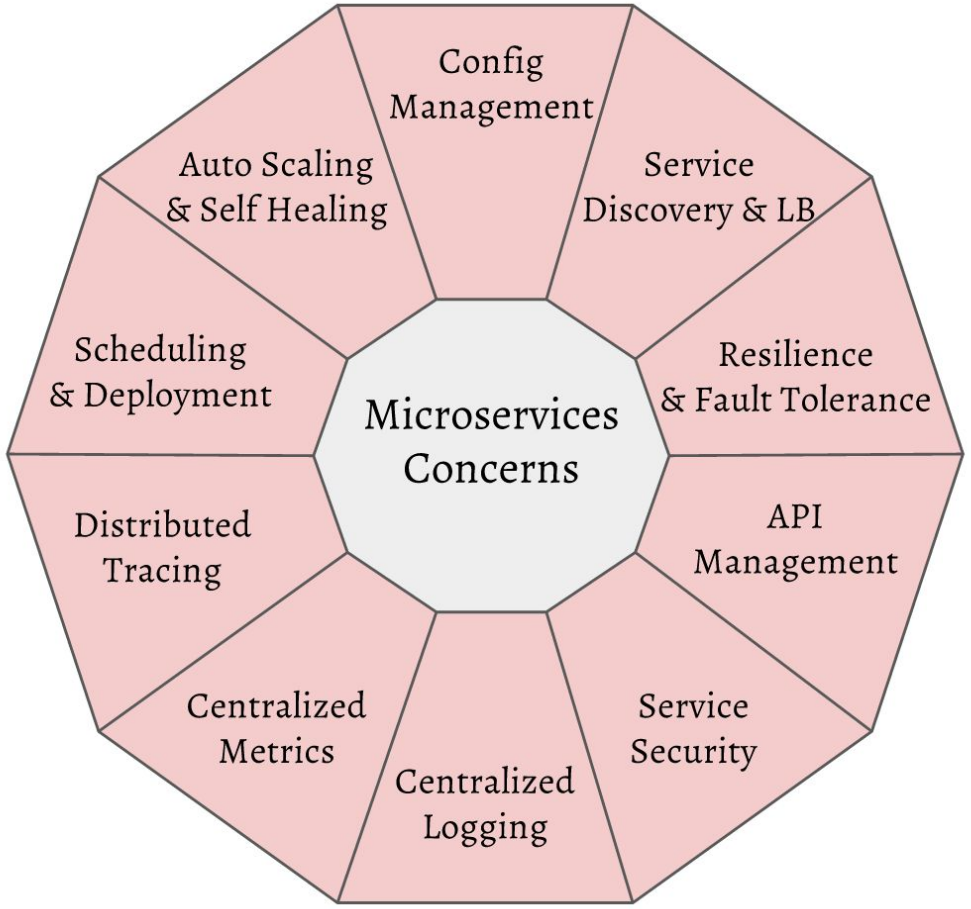
DevOps - Kubernetes

M. Ali Kahoot

Why Kubernetes

https://www.linkedin.com/posts/kahootali_kubernetes-k8s-docker-activity-6712848513002299392-KvDp

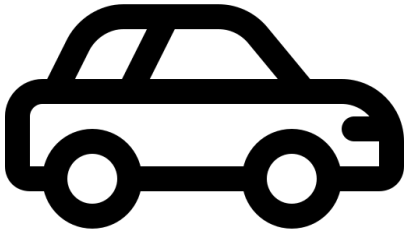
Microservices Concerns

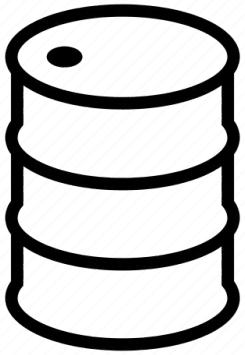


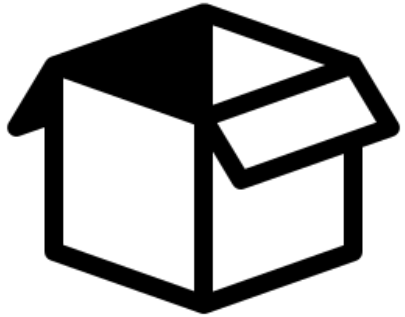
DevOps Course by Ali Kahoot - Dice Analytics

What is a Container

What is a Container







What is a Container

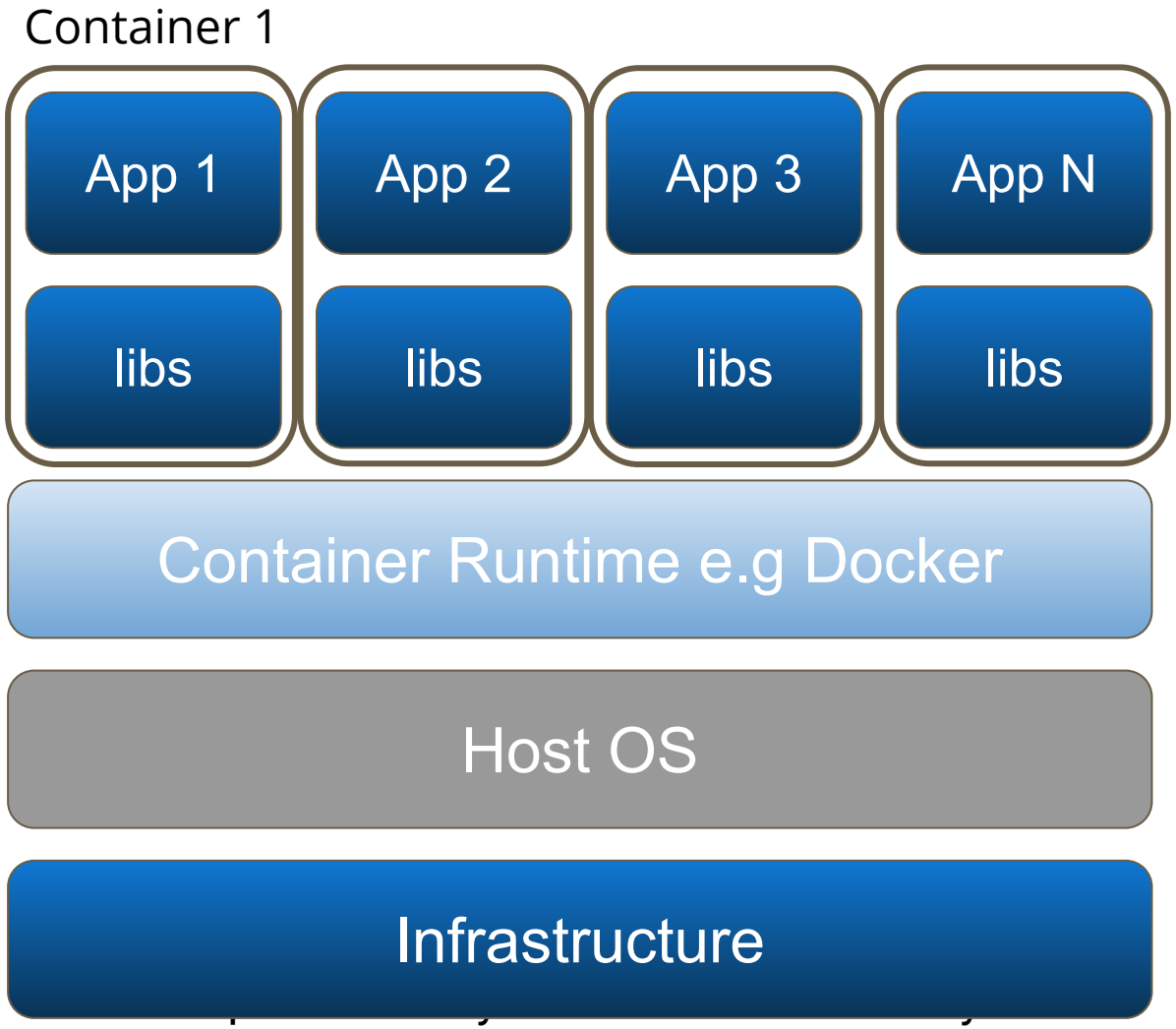
Backend App

Database

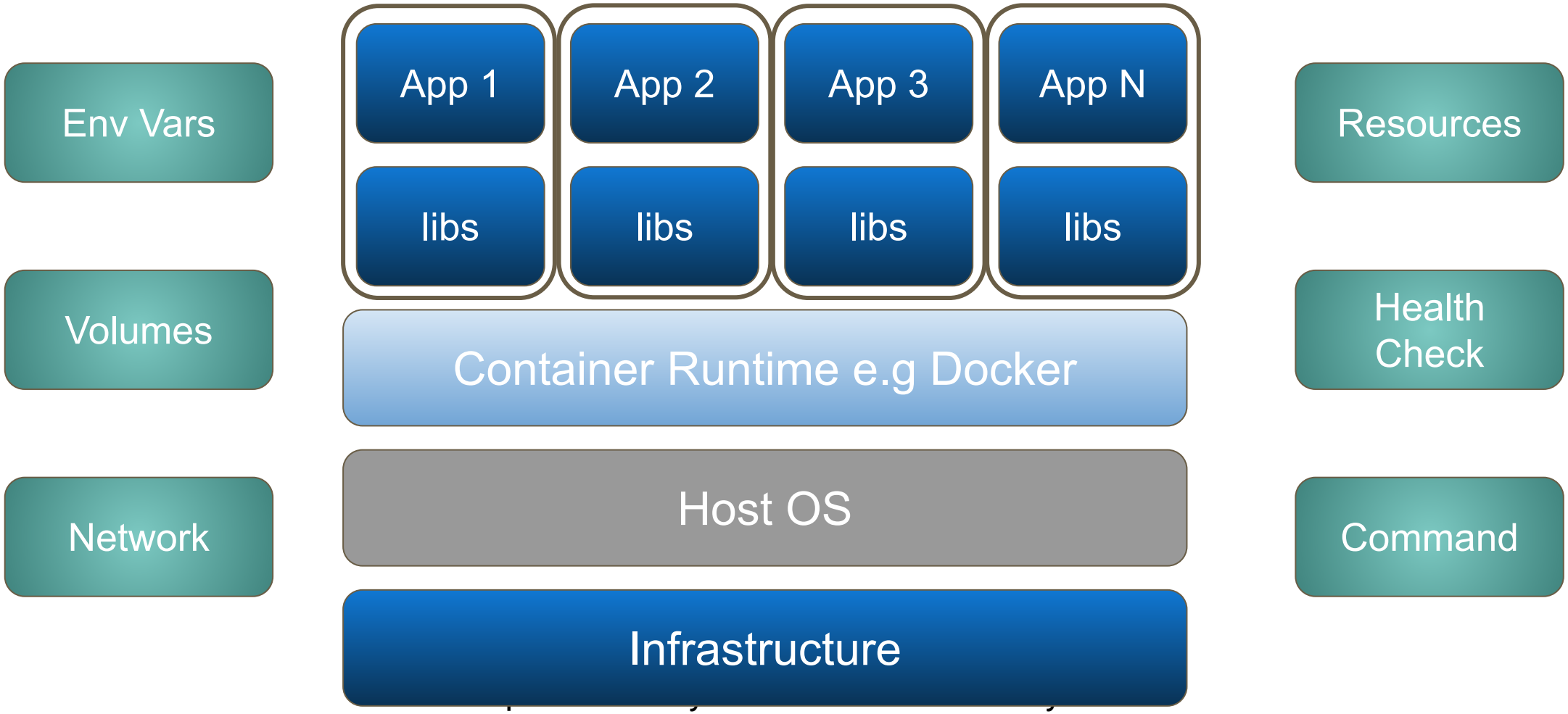
Frontend

Queues

What is a Container



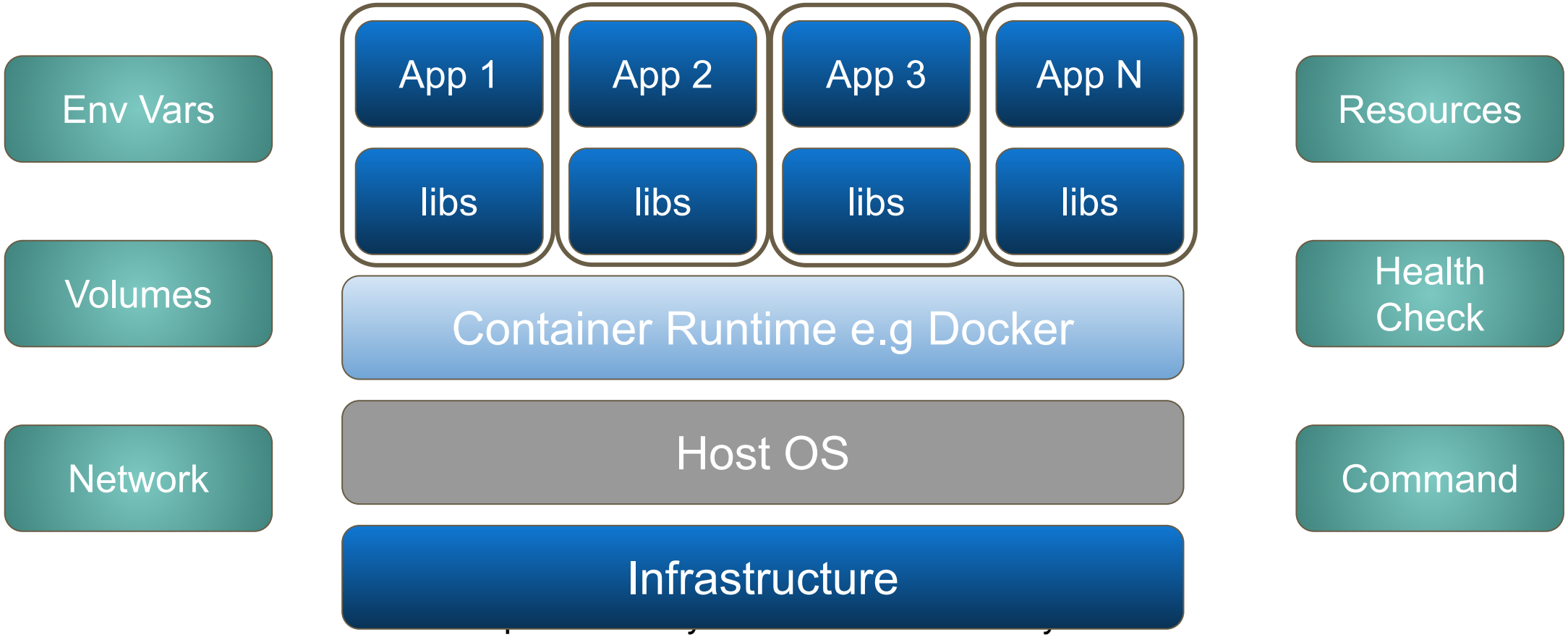
Container Resources



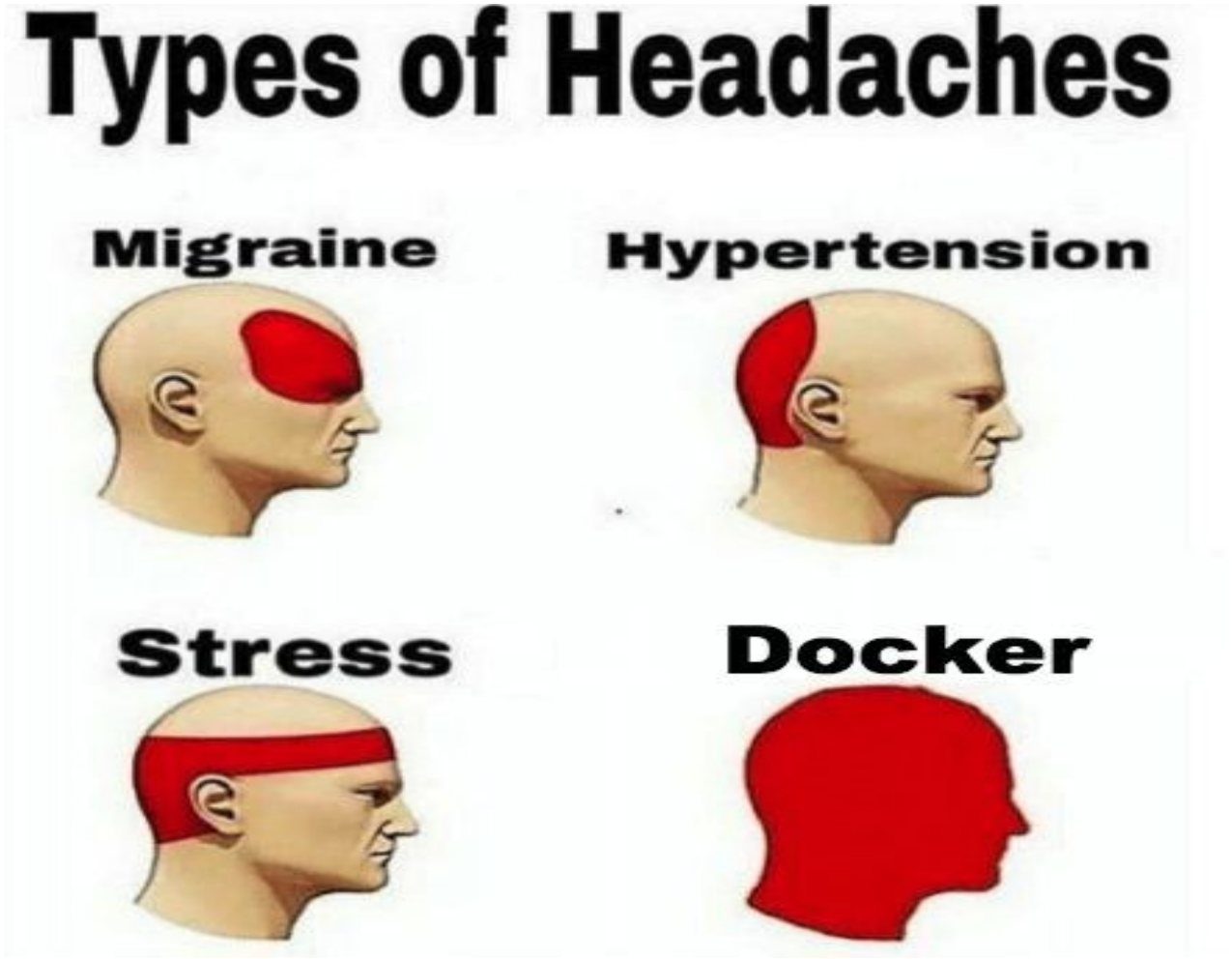
Container Resources

HEALTHCHECK CMD curl --fail http://localhost:5000/ || exit 1

docker run -it -e NAME="Ali Kahoot" --net dok -v DOK:/data/on/kubernetes --memory 1Gi busybox echo "The attendees are awesome"



Containers



DevOps Course by Ali Kahoot - Dice Analytics

CONTAINERS CHALLENGES

- Scaling & Auto Scaling
- Port Conflicts
- Service Discovery & Load Balancing
- Multiple Nodes(Hosts)
- What if Application Fails, How to Restart automatically
- Updating Applications with 0 Downtime
- Handle sensitive data

KUBERNETES

Kubernetes referred as K8s is a portable, extensible, open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation.

Kubernetes

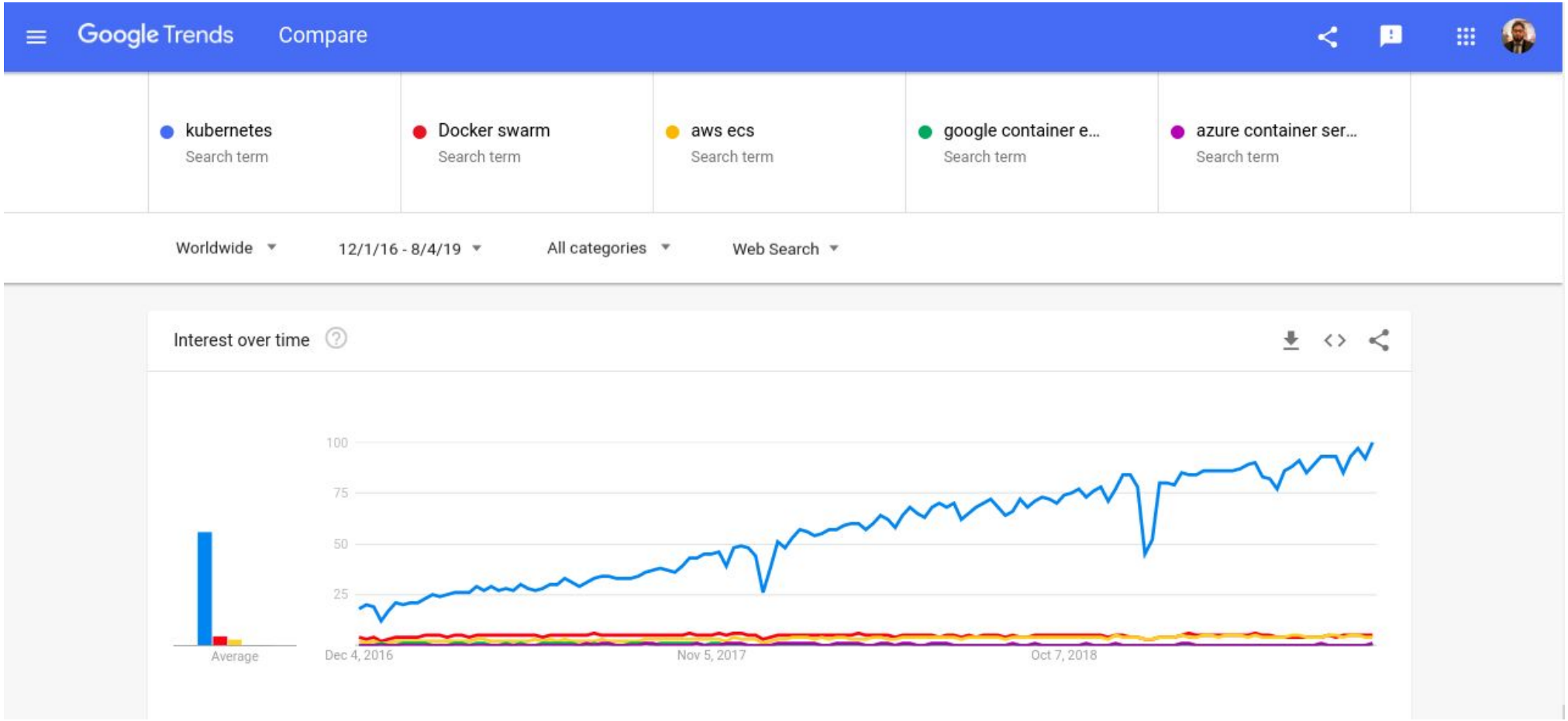
- Container Orchestrator
- Manages Containers
- Fixes issues such as Scaling, Networking,Clustering, etc

ORCHESTRATION

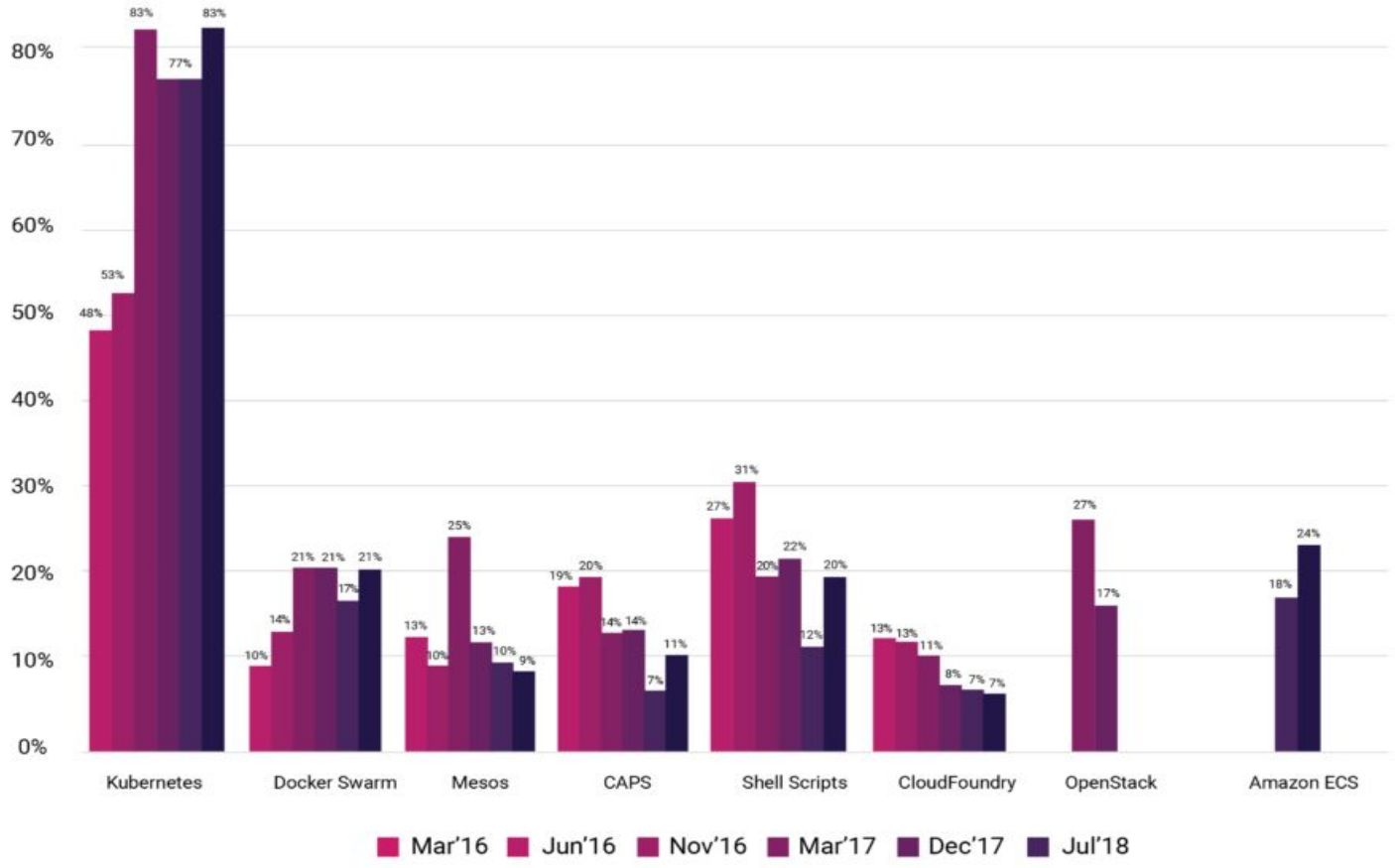
Orchestra: Every person knows what to do, but there is a single person guiding them what to do.

DevOps Course by Ali Kahoot - Dice Analytics

TOOLS TRENDS



TOOLS TRENDS



DevOps Course by Ali Kahoot - Dice Analytics

KUBERNETES IS

- Container Orchestration Platform
- Robust & Reliable
- Supports almost all clouds
- Cloud Agnostic
- Opensource
- Written by Google in Golang, now managed by CNCF(Cloud Native Computing Foundation)
- Huge Community

KUBERNETES IS NOT

- For Containerizing Apps
- Alternative to Docker Containers
- Does not Compile your code(needs Docker images)

KUBERNETES & DOCKER

- K8s compliments Docker Containers, not an alternative
- Schedules Containers
- Alternate of Docker Swarm not containers

FEATURES

- Scaling & Auto-Scaling
- No Port issues
- Service Discovery & Load Balancing
- Can join or remove nodes based on load
- Update applications with 0 downtime
- Rollbacks in case of failure

FEATURES

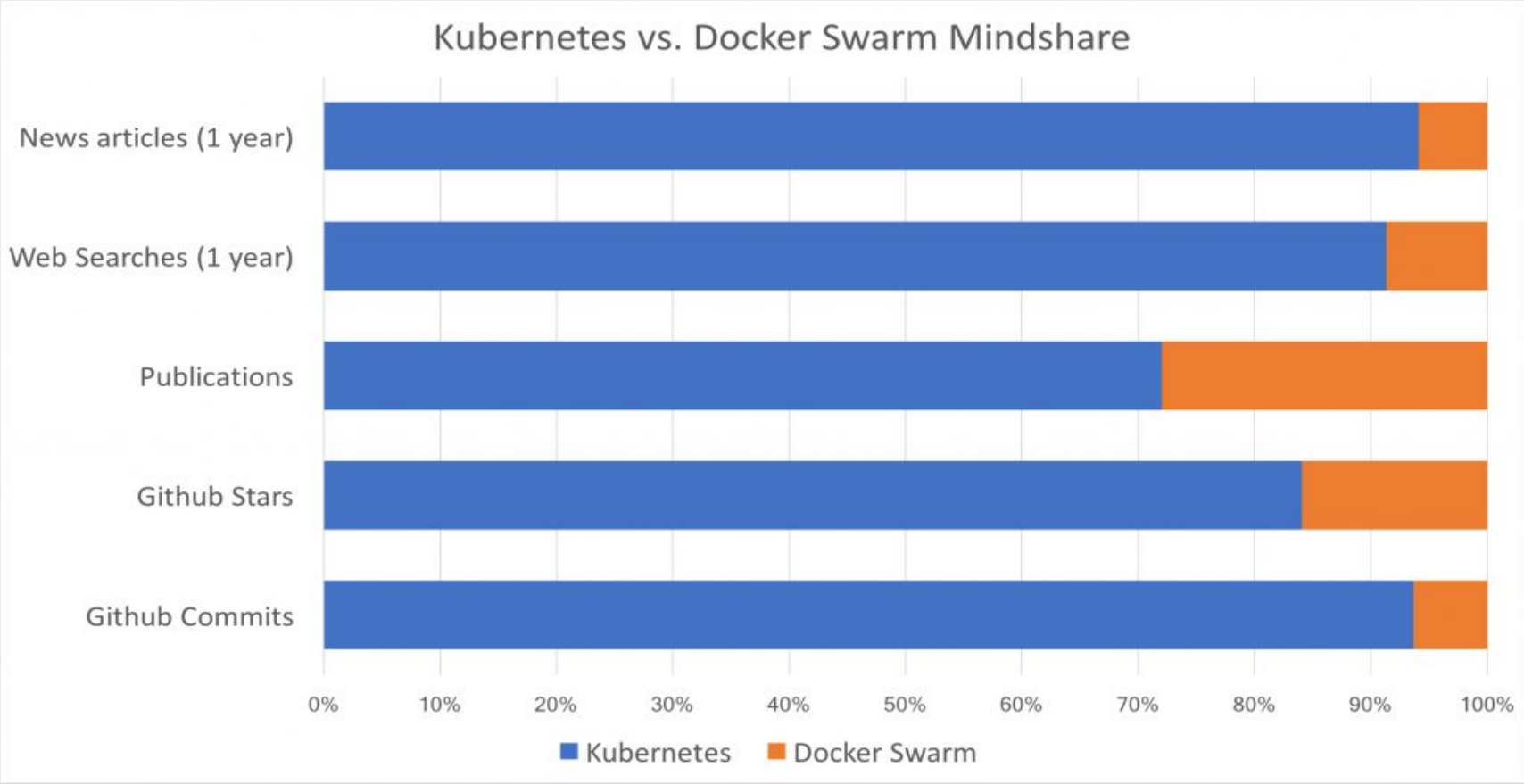
- Fault Tolerant
- Secrets & Configuration management
- Declarative Definition of Containers
- Storage Mounting
- Labelling & Selection of containers

Kubernetes & Docker Swarm

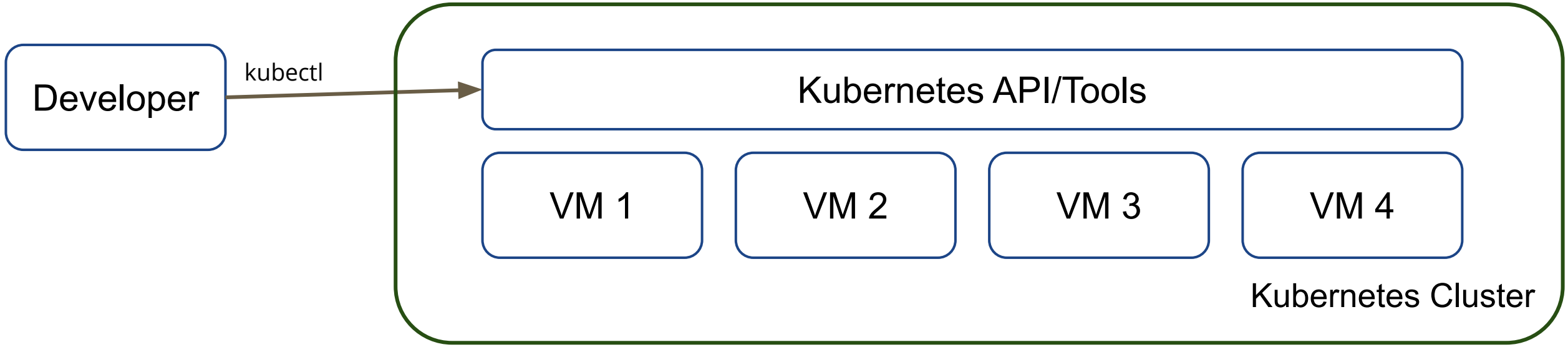
Created by Google, now maintained by CNCF
Manual Installation is Challenging & Complicated, but now managed tools & services are offered

Created by Docker Inc.
Simple as compared to K8s, only 2 commands

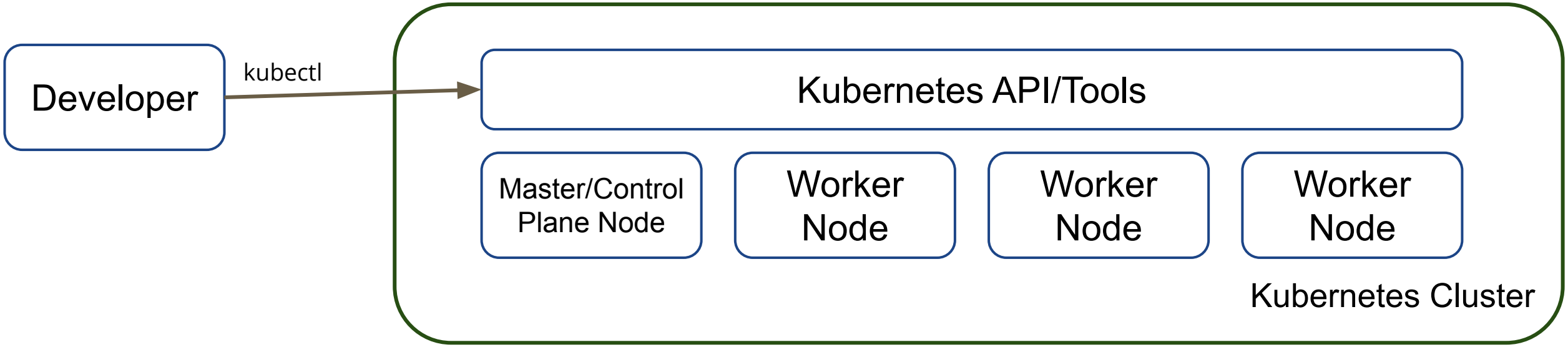
K8s vs Docker Swarm



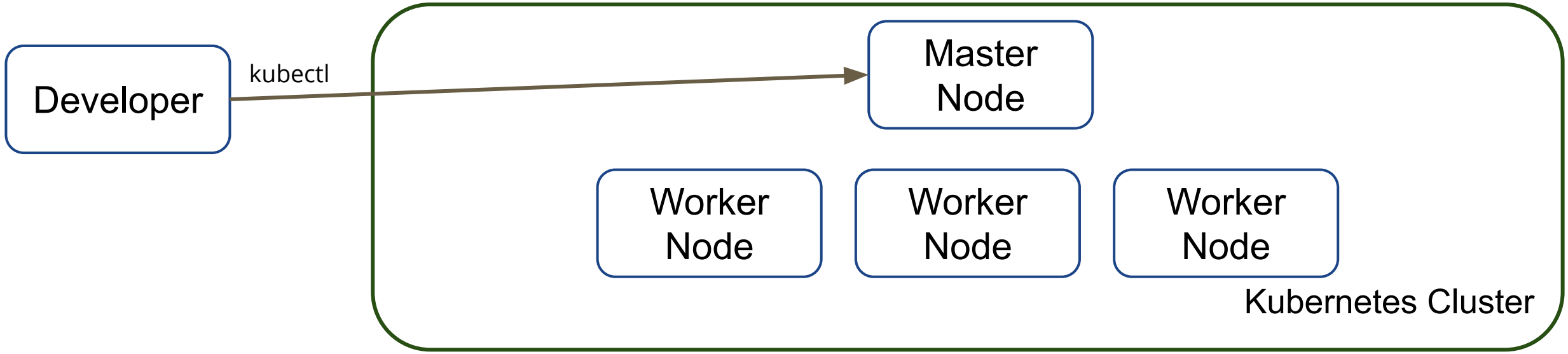
Kubernetes Architecture



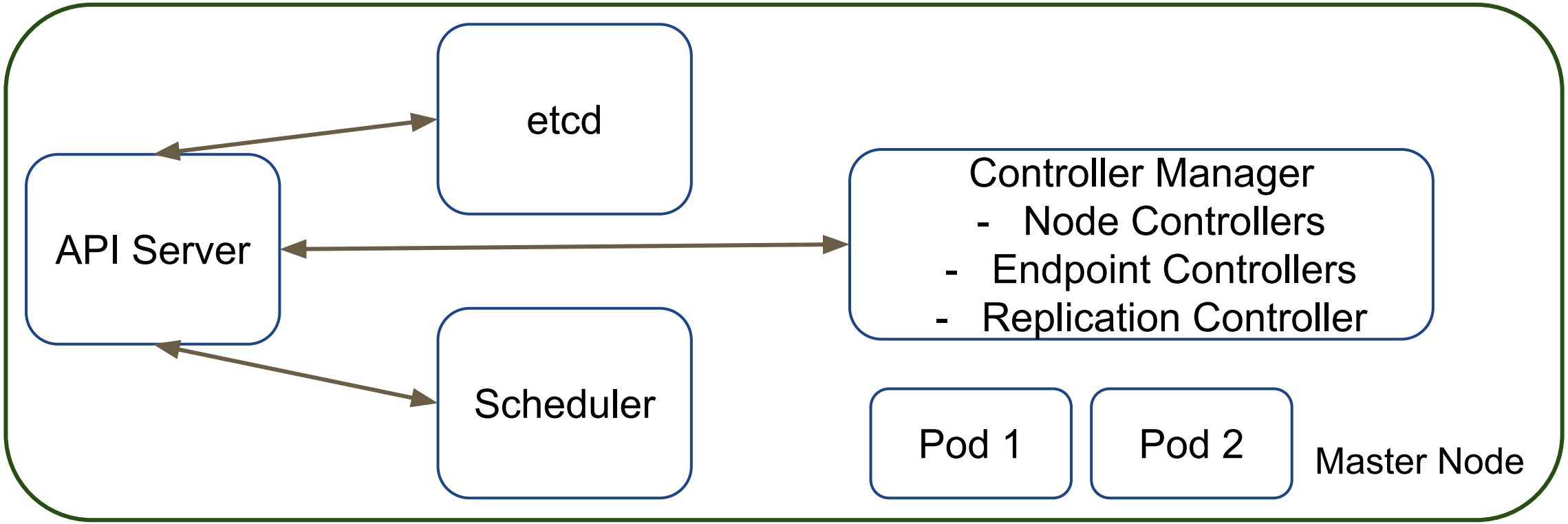
Kubernetes Architecture



Kubernetes Architecture



Kubernetes Architecture



Master Components: API Server

- Management hub for the Kubernetes cluster
- Facilitates communication between the various components inside & from outside the cluster.
- Validates, Authenticates and checks authorization of the calls to the cluster
- Thereby maintaining the cluster health
- Offers a CRUD (Create, Read, Update, and Delete) interface for querying and modifying the cluster state,
- Only component to communicate with etcd

Master Components: etcd

- In simple terms, the DB of the cluster
- Manages the state of the cluster
- A simple Key:Value store
- Stores configuration data which can be accessed by the Kubernetes Master's API Server using simple HTTP or JSON API

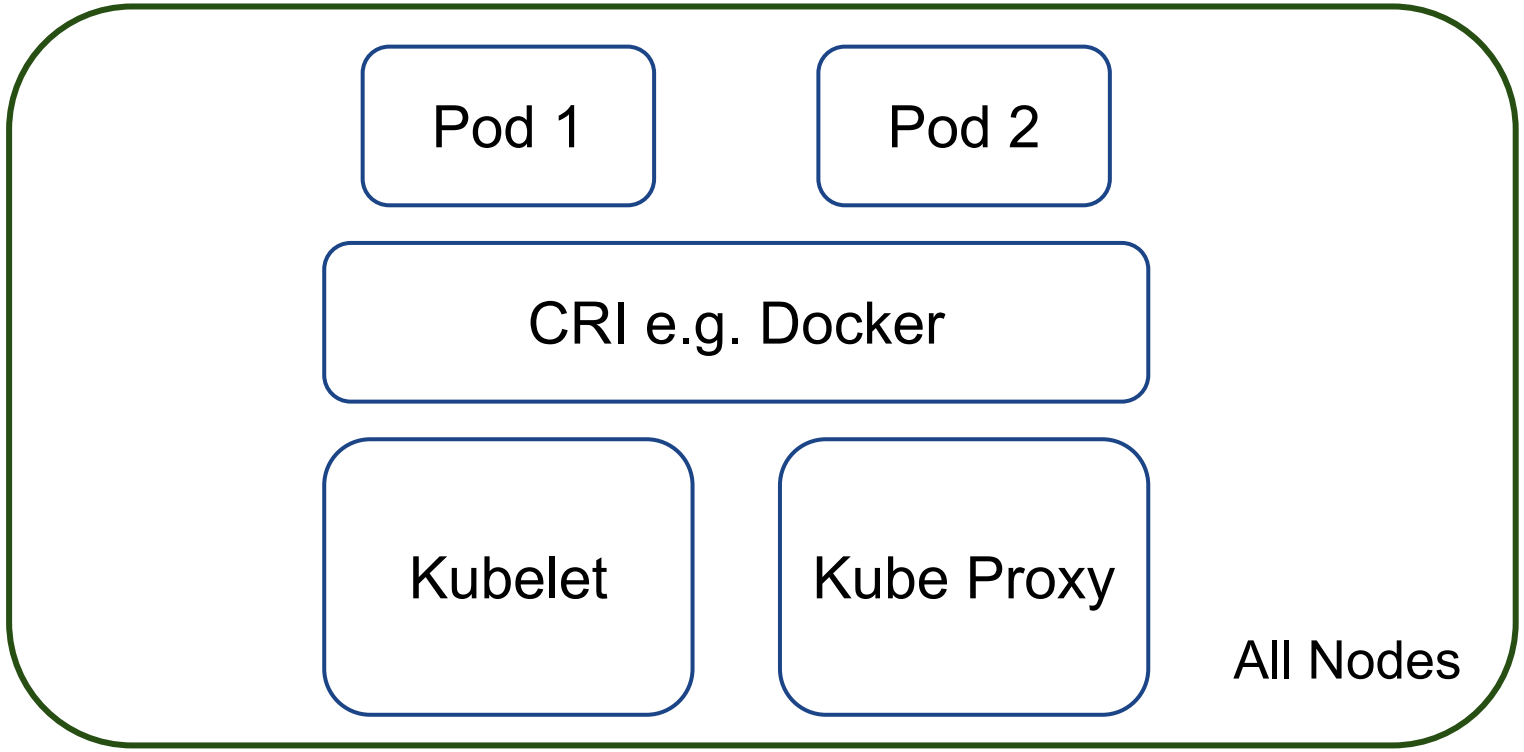
Master Components: Scheduler

- Decides the best node on which the workload(Pod) should be deployed to
- Watches new workloads and assigns nodes to them
- Default methodology is to check resource availability on nodes
- Can be overridden to use Custom Scheduling

Master Components: Controller Manager

- Manages Controllers
- Runs in loops and tries to matches the current & desired state
- kube-controller-manager implements:
  - node controller
  - endpoints controller
  - Deployment
  - namespace controller etc.

Kubernetes Architecture

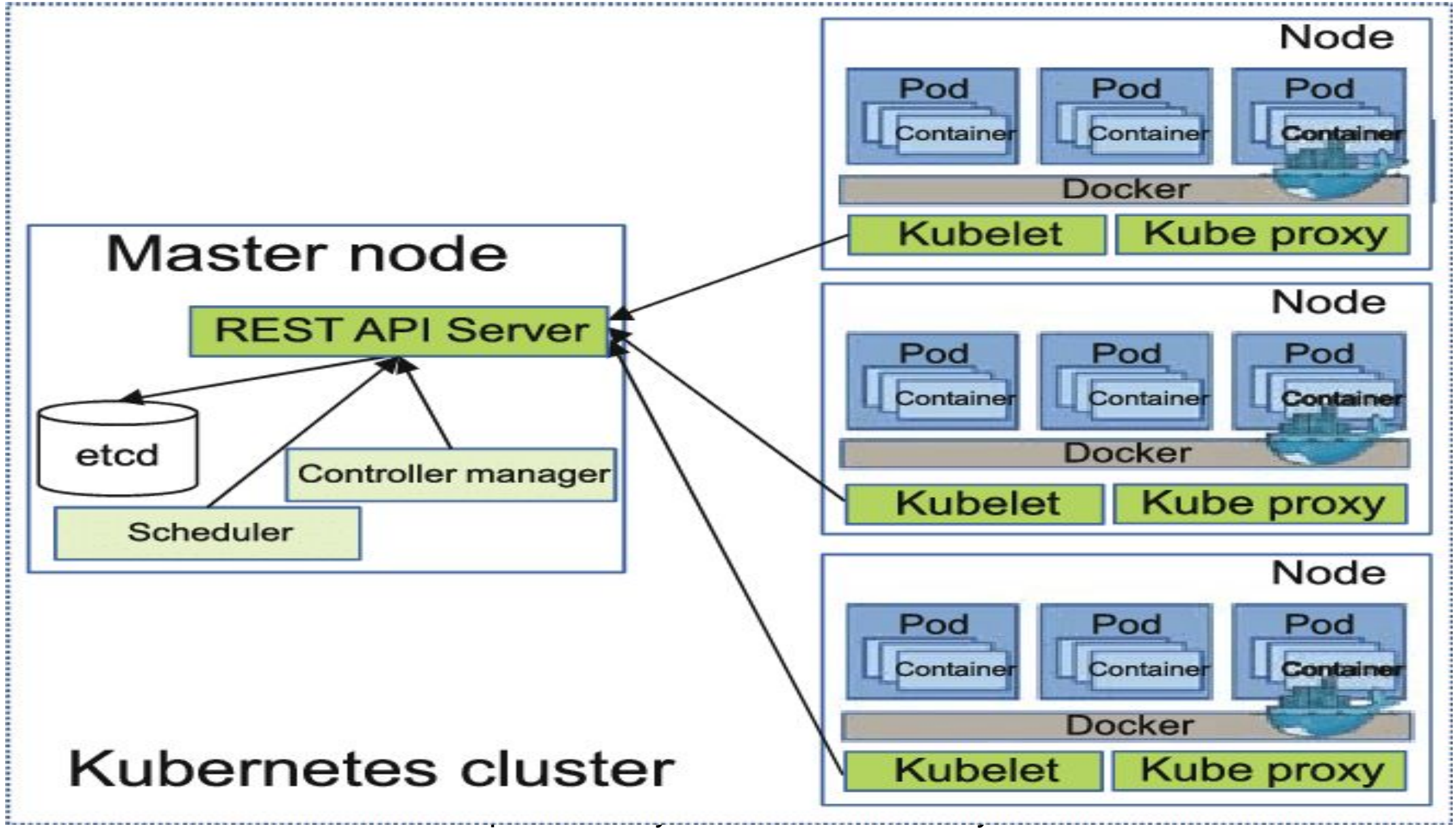


All Node Components: Kubelet

- The bridge that joins the available CPU, disk, and memory for a node into the large Kubernetes cluster.
- Communicates with the API server to find pods that should be running on its node.
- Reports back the health of the node & the pods running on it
- Manages the health check and restart policy of container

All Node Components: Kube-Proxy

- The network brain of the node
- Manages the Routing Table of the node
- Every Pod get its unique IP
- Routes traffic to actual pod


Node
Pod
Container
Pod
Container
Pod
Container
Docker
Kubelet
Kube proxy
Master node
REST API Server
etcd
Controller manager
Scheduler
Node
Pod
Container
Pod
Container
Pod
Container
Docker
Kubelet
Kube proxy
Node
Pod
Container
Pod
Container
Pod
Container
Docker
Kubelet
Kube proxy
Kubernetes cluster

K8S SETUP

- **Minikube**: easily create a local, single-node Kubernetes cluster for development and testing
- **kubeadm**: bootstrap a Kubernetes cluster conforming to best practices. Can be installed on various types of platforms and can easily integrate with provisioning systems such as Terraform or Ansible.

K8S SETUP

- **Kops**: create, destroy, upgrade and maintain production-grade, highly available, Kubernetes clusters from the command line.
- **Hosted Solutions**: from various vendors including ibm, google, azure, amazon, etc.

K8S SETUP: Minikube

We will be installing Minikube on the host machine

Install minikube from: https://minikube.sigs.k8s.io/docs/start/ for your specific OS then run

minikube start

kubectl get po -A

Once both commands run successfully, you have a local K8s cluster

DASHBOARD OVERVIEW

Now enable dashboard, by running

minikube dashboard

This will start running the k8s dashboard in the browser.

Pod

- Smallest Deployable unit in Kubernetes
- Wrapper around Containers for easily passing arguments to a container
- Kubernetes way of provisioning Containers
- Descriptive way to declare containers & resources around it
- Can have 1 or more Containers

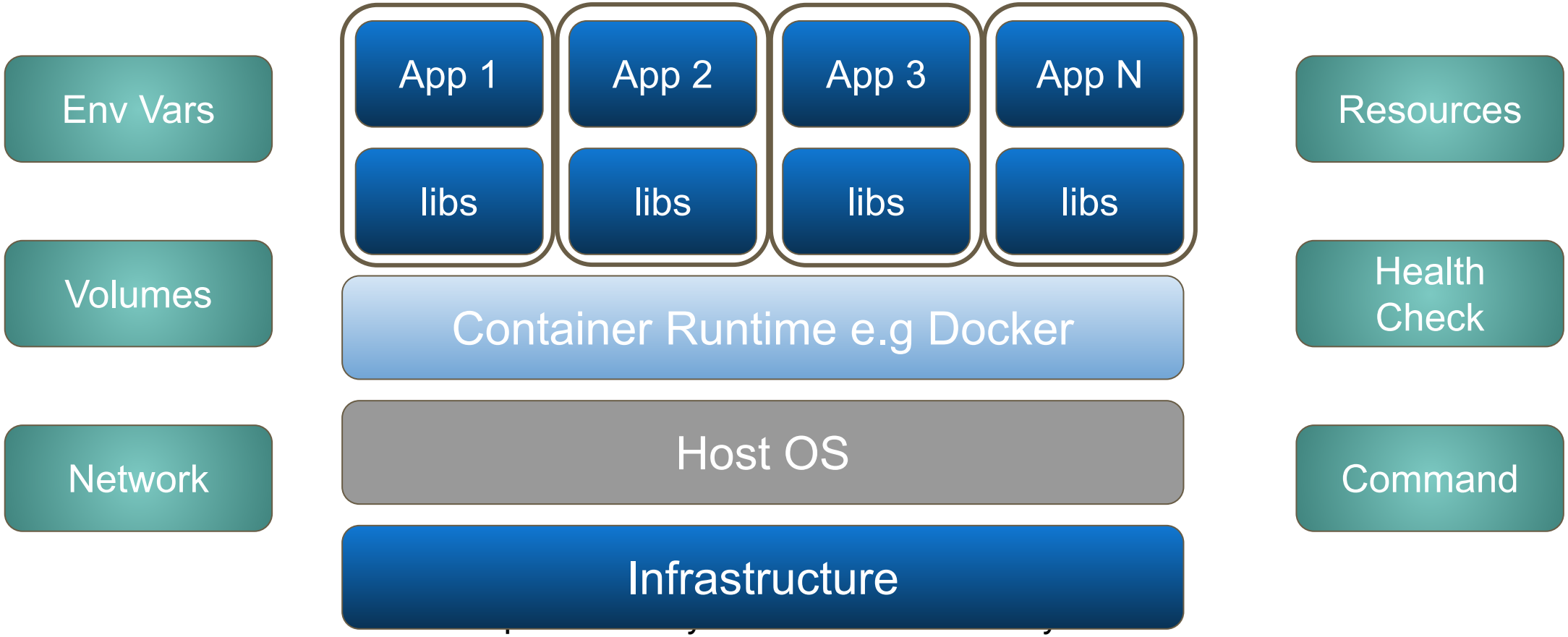
Pod

- All containers in a Pod share storage/network, live & die together
- Containers within a Pod share an IP address and port space, and can find each other via localhost
- Commonly one container per pod is used

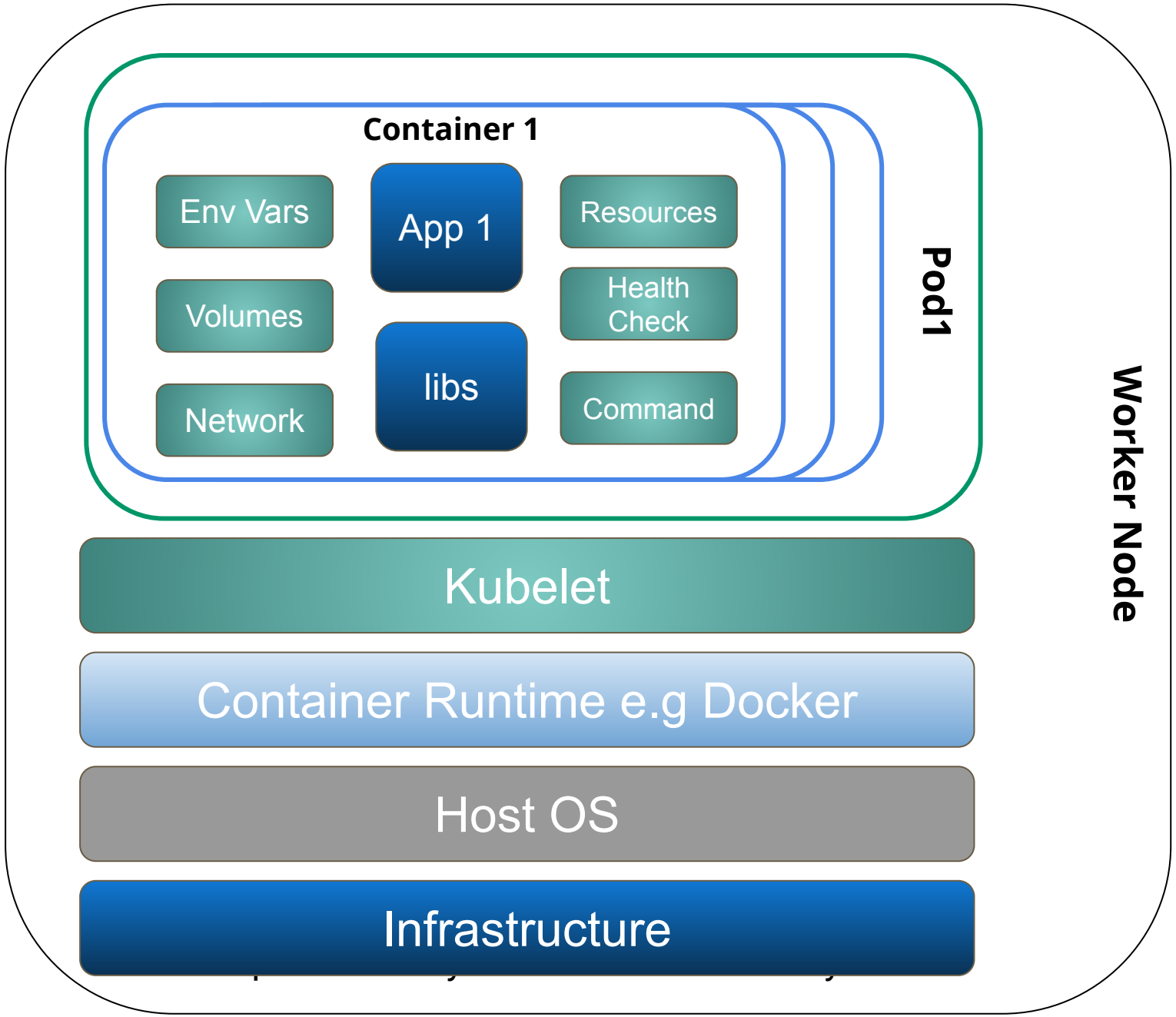
Container Resources

HEALTHCHECK CMD curl --fail http://localhost:5000/ || exit 1

docker run -it -e NAME="Ali Kahoot" --net dok -v DOK:/data/on/kubernetes --memory 1Gi busybox echo "The attendees are awesome"



Pod


Container 1
Env Vars
App 1
Resources
Volumes
Health Check
libs
Network
Command
Pod1
Kubelet
Container Runtime e.g Docker
Host OS
Infrastructure
Worker Node

Running an App

Docker

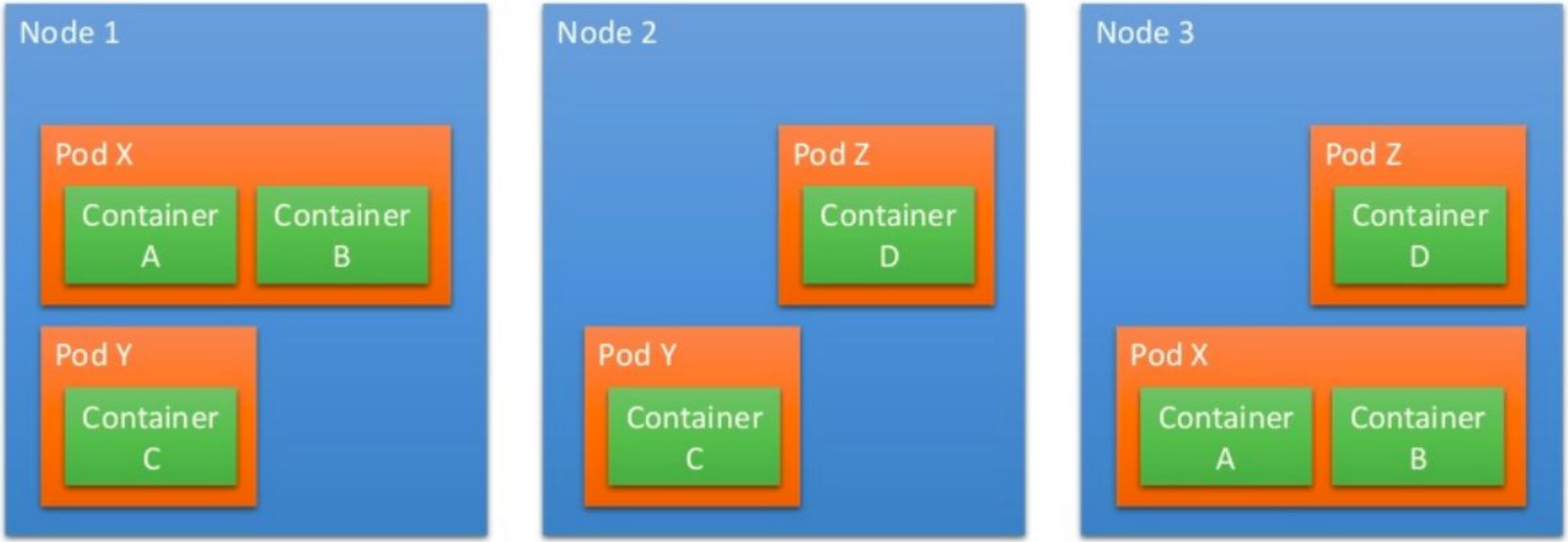
docker run -it --name dok-app -e NAME=”Ali Kahoot” --net dok -v DOK:/data/on/kubernetes --memory 1Gi busybox echo “The attendees are awesome”

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: dok-app
  labels:
    app: dok-app
spec:
  containers:
    - name: dok-app
      image: busybox
      args: ["echo","The attendees are awesome"]
      resources:
        limits:
          memory: 1Gi
      env:
        name: NAME
        value: Ali Kahoot
      volumeMounts:
        - name: dok
          mountPath: /data/on/kubernetes
      volumes:
      - name: dok
        persistentVolumeClaim:
          claimName: dok
```

One Service in One Pod

- A Pod can have more than 1 container, but it's a really bad practice to have more than 1 service in one pod.
- But we can have sidecar container to facilitate the main service.
- Taking backup of the main container
- Exporting logs of the main container

Pods


Node 1
Pod X
Container A
Container B
Pod Y
Container C
Node 2
Pod Z
Container D
Pod Y
Container C
Node 3
Pod Z
Container D
Pod X
Container A
Container B

LABS

We will be using

https://github.com/kahootali/k8s-labs

For the labs

Lab: Single Container Pod

```
apiVersion: v1
kind: Pod
metadata:
  name: counter
spec:
  containers:
  - name: count
    image: busybox
    args: [/bin/sh, -c,'i=0; while true; do echo "$i: $(date)"; i=$((i+1)); sleep 1s; done']
```

There is a file named **single-container.yaml** in the module-1 folder and then apply it by running

```
kubectl apply -f single-container.yaml
```

Lab: Single Container Pod

```
Check if pod is in running state
kubectl get pods

To get logs of the pod
kubectl logs counter

Or go to dashboard and see the state of the pod and the logs

minikube dashboard
```

LAB: MULTI CONTAINER POD

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: counter
spec:
  containers:
  - name: counter-1
    image: busybox
    args: [/bin/sh,-c,'i=0; while true; do echo "$i: $(date)"; i=$((i+1)); sleep 1s; done']
  - name: counter-2
    image: busybox
    args: [/bin/sh,-c,'i=0; while true; do echo "$i: $(date)"; i=$((i+1)); sleep 2s; done']
  - name: counter-3
    image: busybox
    args: [/bin/sh,-c,'i=0; while true; do echo "$i: $(date)"; i=$((i+1)); sleep 3s; done']
```

LAB: MULTI CONTAINER POD

Save above in a file named **multi-container.yaml** and then apply it by running

```
ali-kahoot@Alis-MacBook-Pro kubectl apply -f multi-container.yaml
```

Now see its running by:

```
ali-kahoot@Alis-MacBook-Pro kubectl get pods
```

You should get this

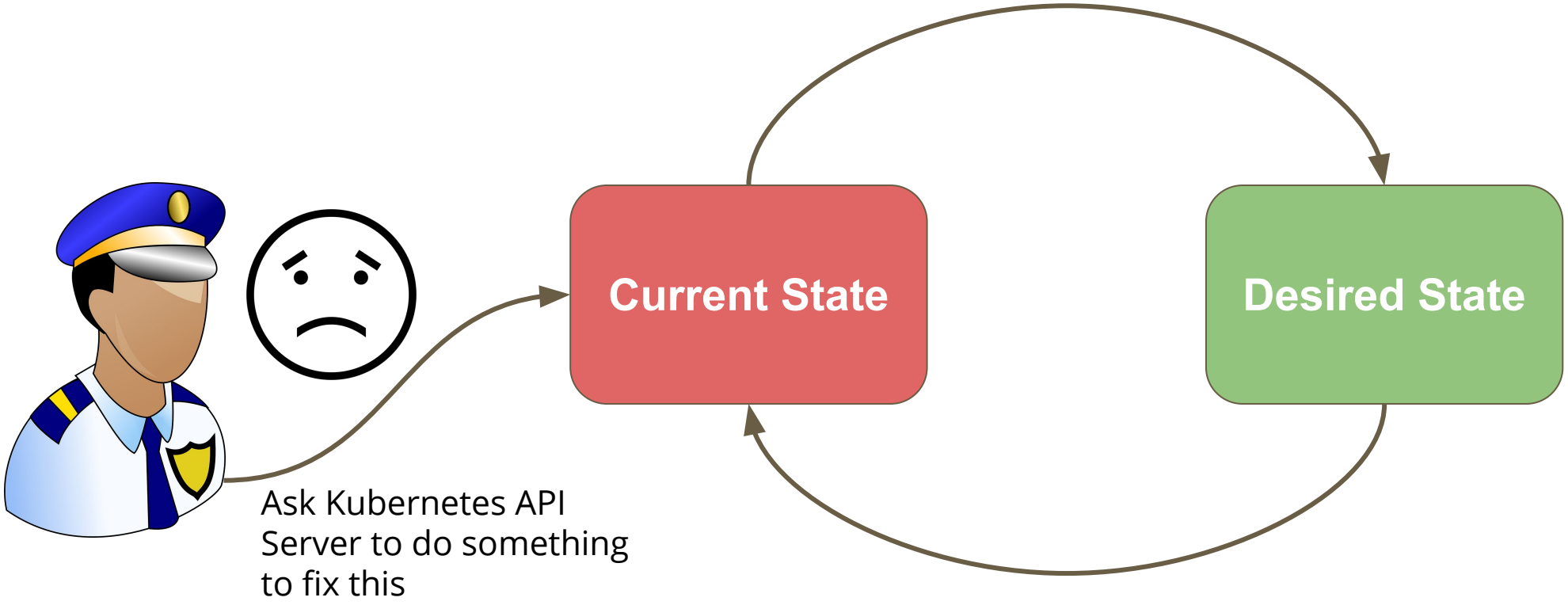
```
NAME                                    READY     STATUS    RESTARTS   AGE
counter                                 3/3       Running   0          18s
```

3/3 means that 3 containers are running

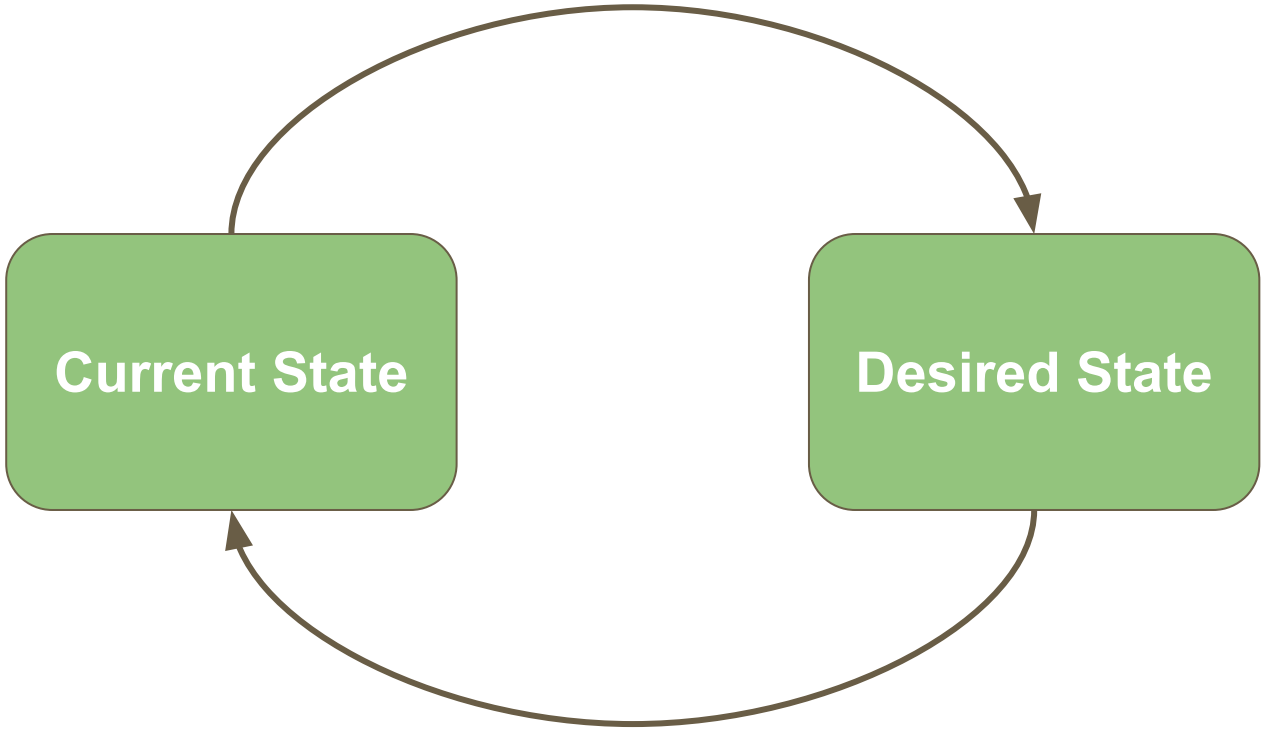
LAB: MULTI CONTAINER POD

```
To see logs run
    kubectl logs counter

You will get error : Error from server (BadRequest): a container name must be specified for
 pod counter, choose one of: [counter-1 counter-2 counter-3]
So specify container name:

    kubectl logs counter -c <container-name> e.g. counter-1/counter-2/counter-3
```

Controllers



Controllers



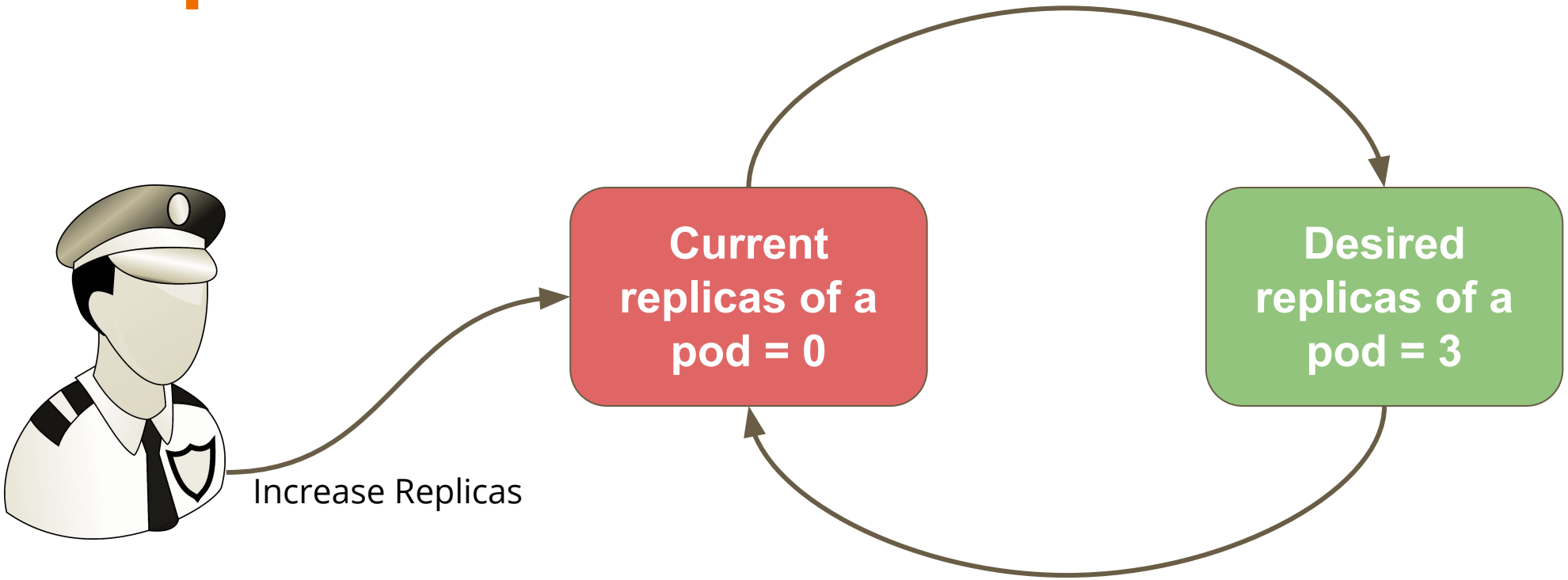
Controllers in Kubernetes

- Node Controllers
- Service Controller
- Ingress Controllers
- Replication Controllers
- ReplicaSet
- Deployments
- StatefulSets
- Daemonsets

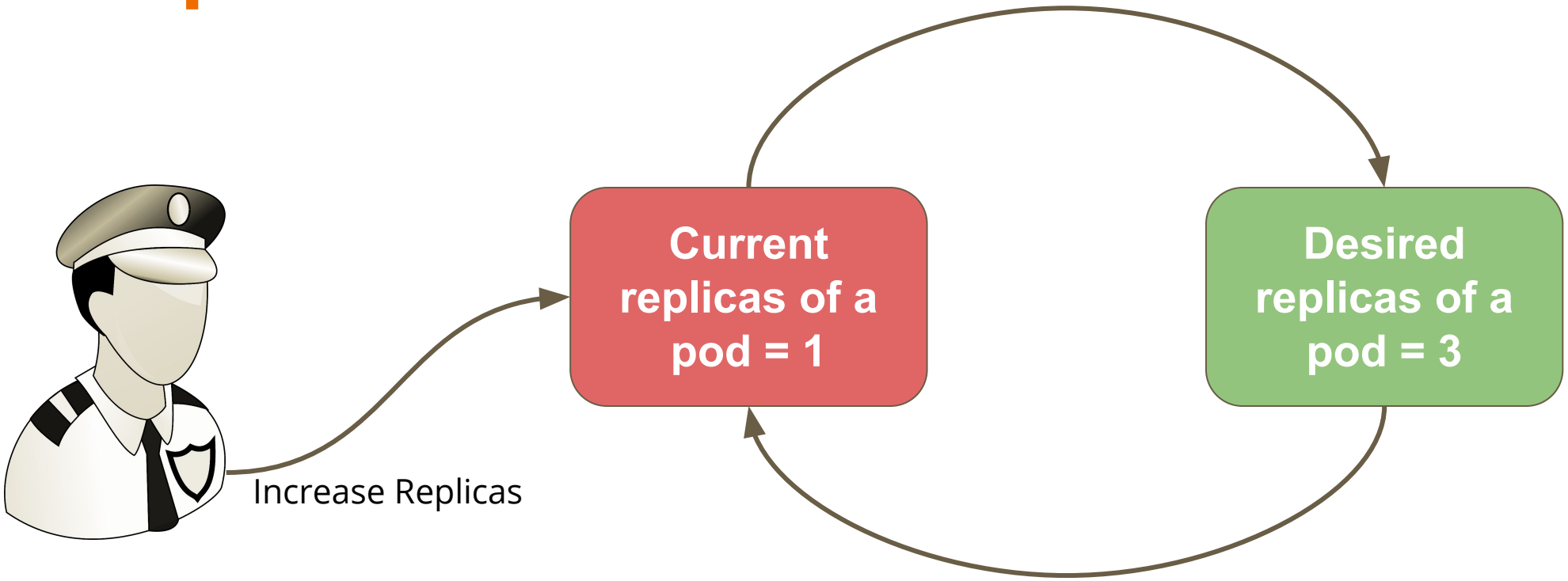
ReplicaSet

- Individual Pods cannot be updated/scaled
- If deleted, they will not be recreated
- Replicaset manages Pods & their replicas
- The replicas are exact same copies of each other
- The pods of a Replicaset can be used interchangeably

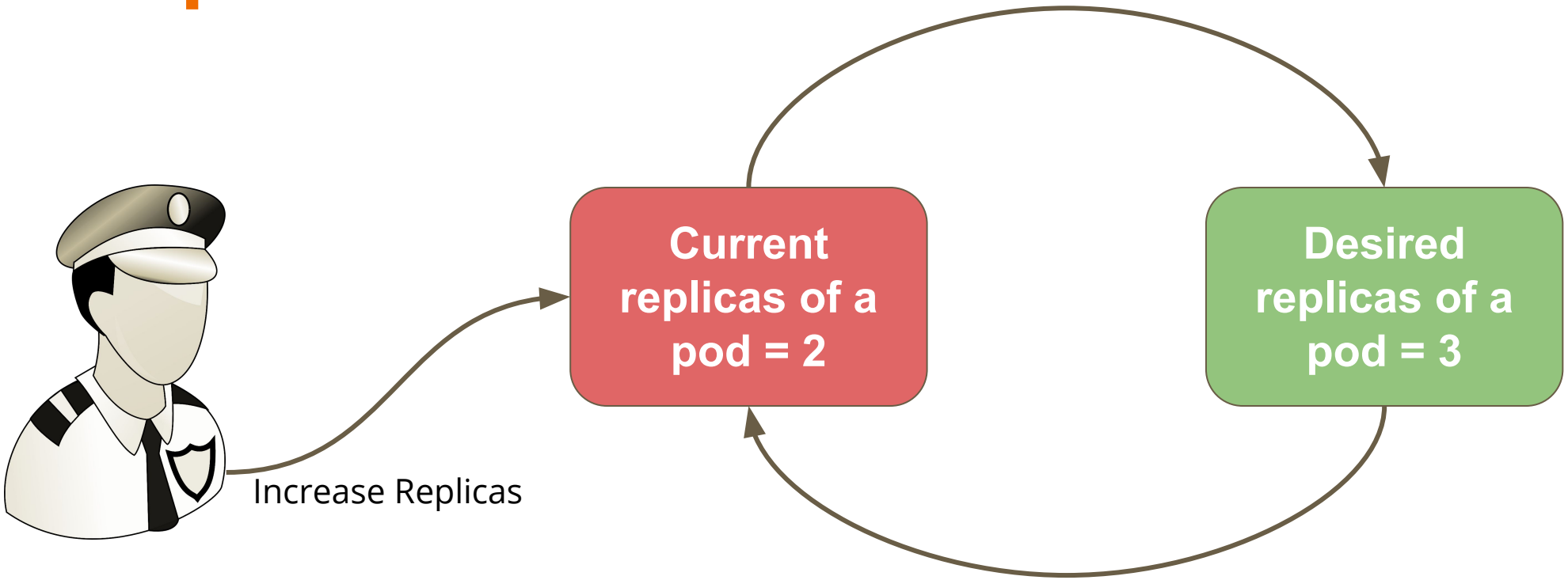
Replicaset


Current replicas of a pod = 0
Desired replicas of a pod = 3
Increase Replicas

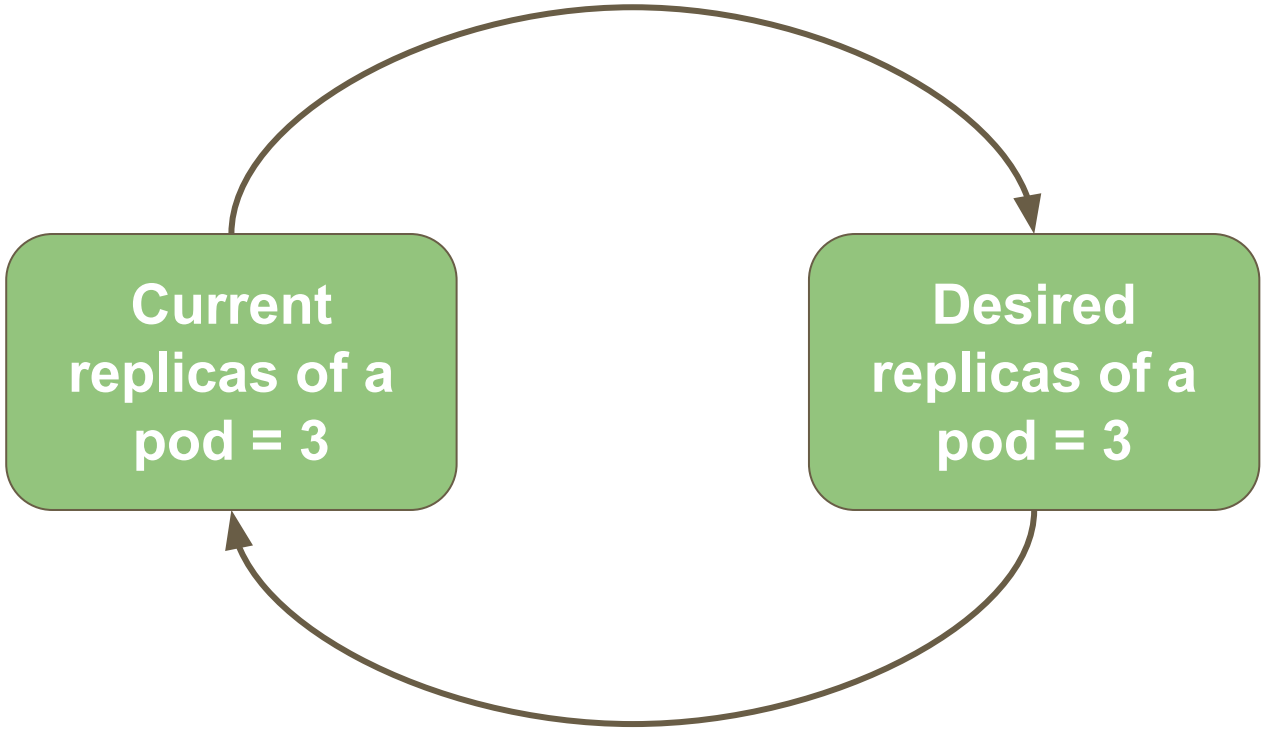
Replicaset


Current replicas of a pod = 1
Desired replicas of a pod = 3
Increase Replicas

Replicaset


Current replicas of a pod = 2
Desired replicas of a pod = 3
Increase Replicas

Replicaset



Lab: ReplicaSet

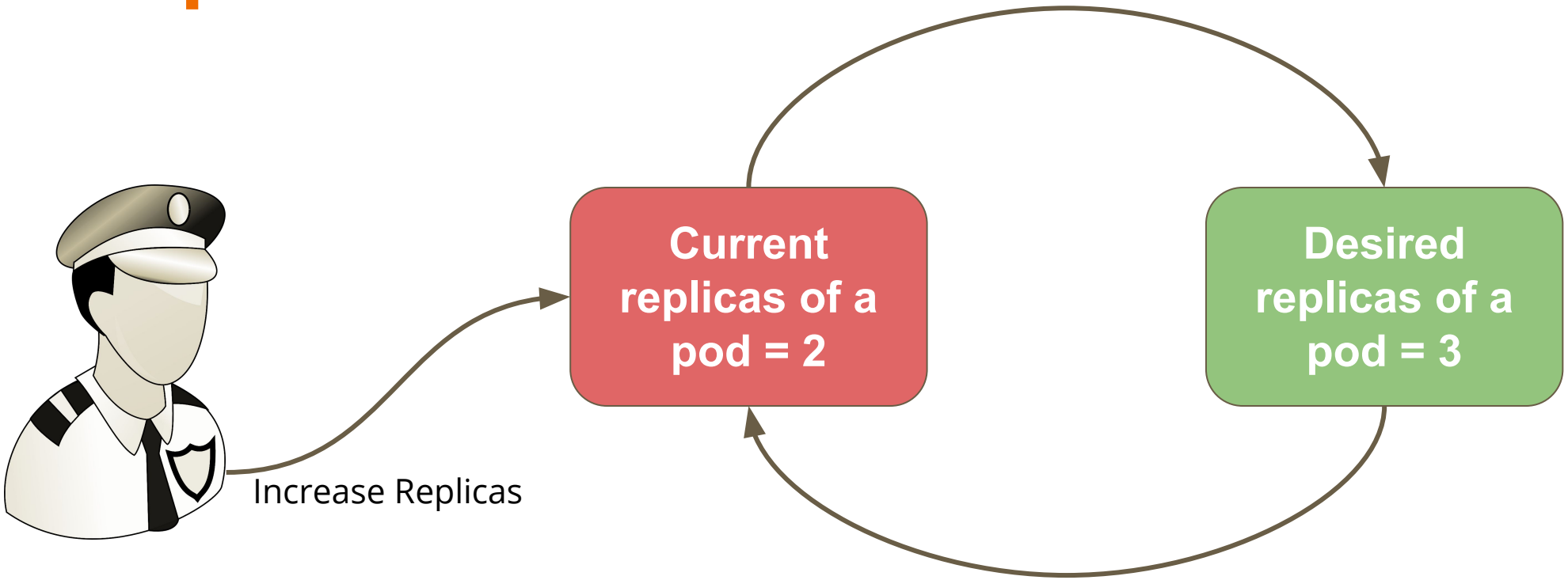
There is a file named **replicaset.yaml**, we will apply the Replicaset by running

```
    ali-kahoot@Alis-MacBook-Pro kubectl apply -f replicaset.yaml
```

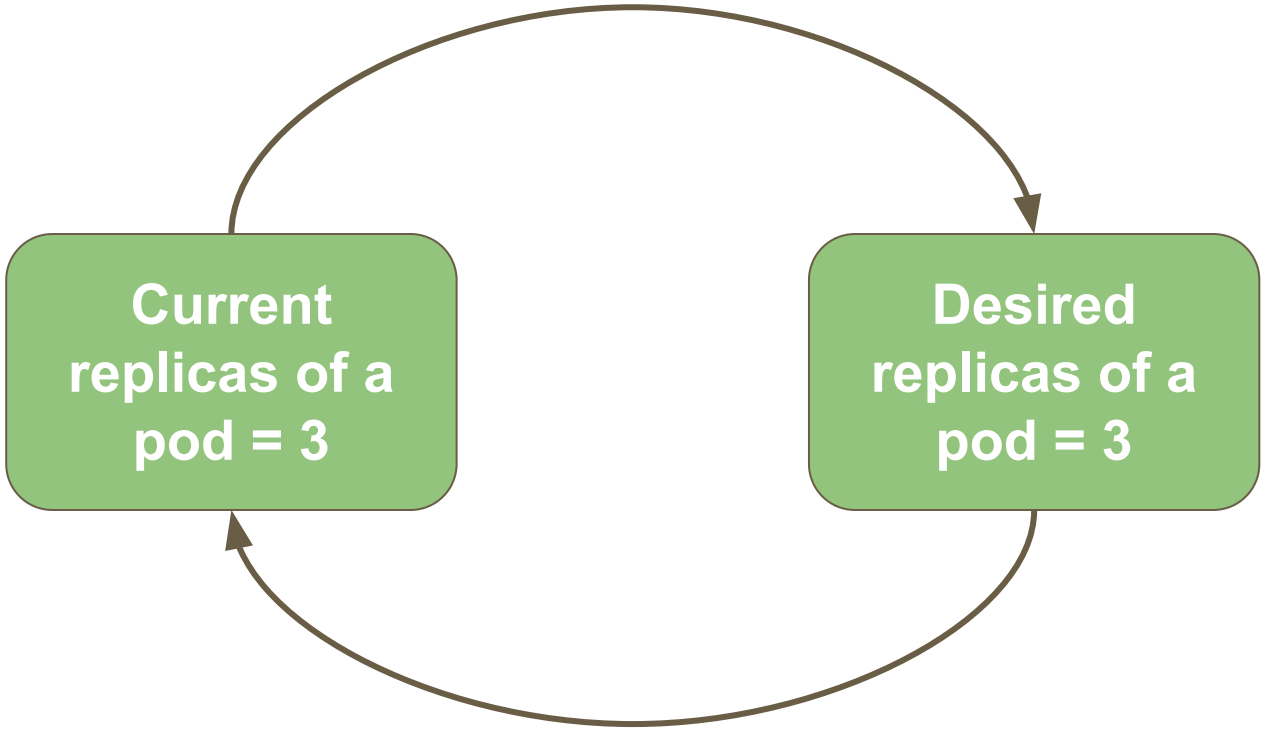
Now see the replicaset and pods by running:

```
    ali-kahoot@Alis-MacBook-Pro kubectl get replicaset,pods
NAME                        DESIRED   CURRENT   READY   AGE
replicaset.apps/counter     3         3         3       33s

NAME                 READY   STATUS    RESTARTS   AGE
pod/counter-mnmjl    1/1     Running   0          33s
pod/counter-x4j2f    1/1     Running   0          33s
pod/counter-xkfgv    1/1     Running   0          33s
```

Replicaset


Current replicas of a pod = 2
Desired replicas of a pod = 3
Increase Replicas

Replicaset



Replicaset Shortcomings

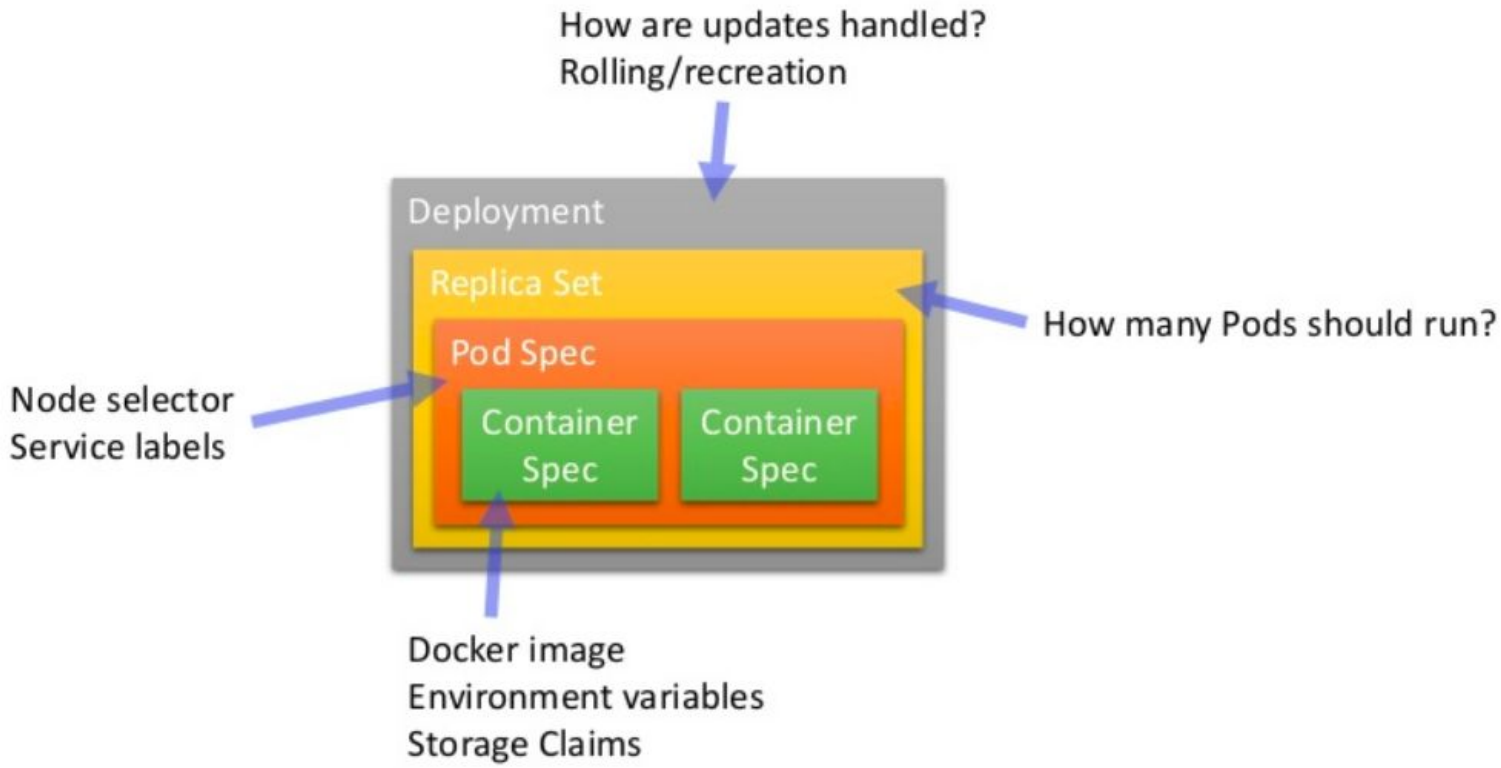
- Versioning is not handled

- Demo: Deployed busybox:latest, now I want to update busybox:1.33.1, it will not update it, you would have to delete previous pods

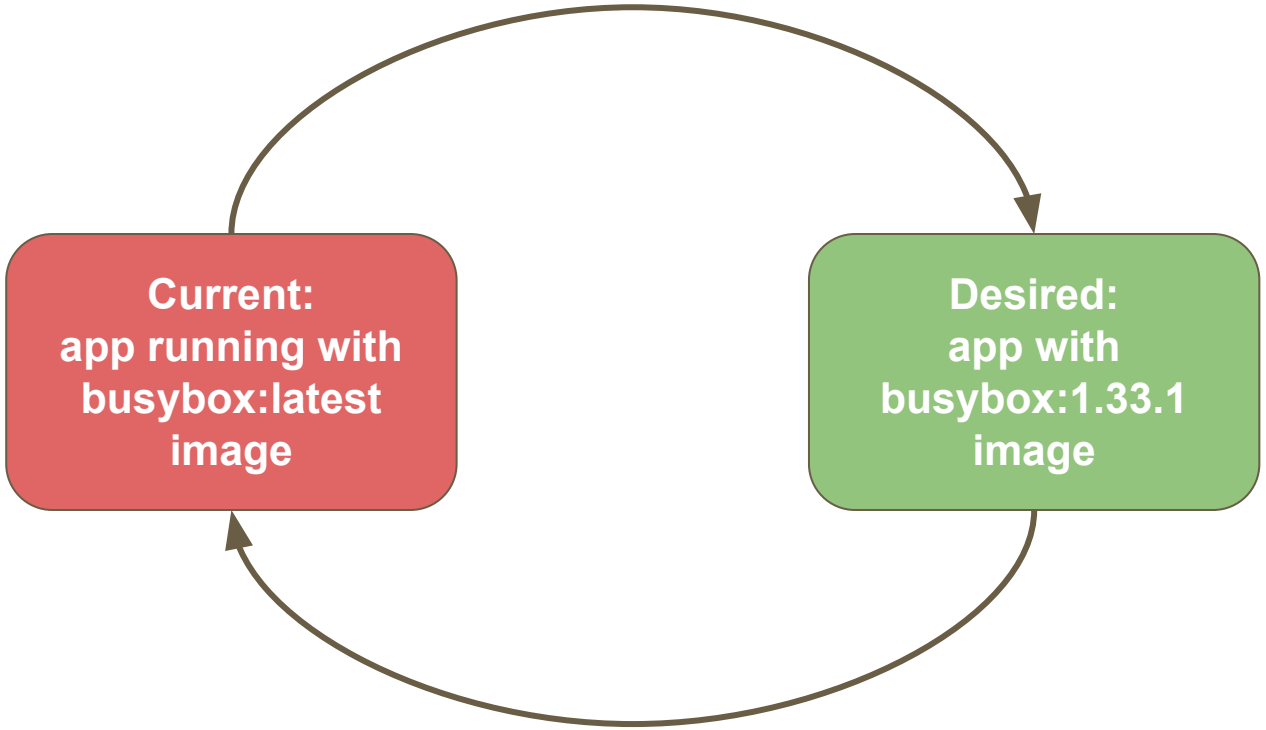
Deployments

- Handles Replicasets based on update of configurations
- Provides Rolling Update
- Supports Rollback
- Ideal for stateless applications
- Most common used for deploying applications

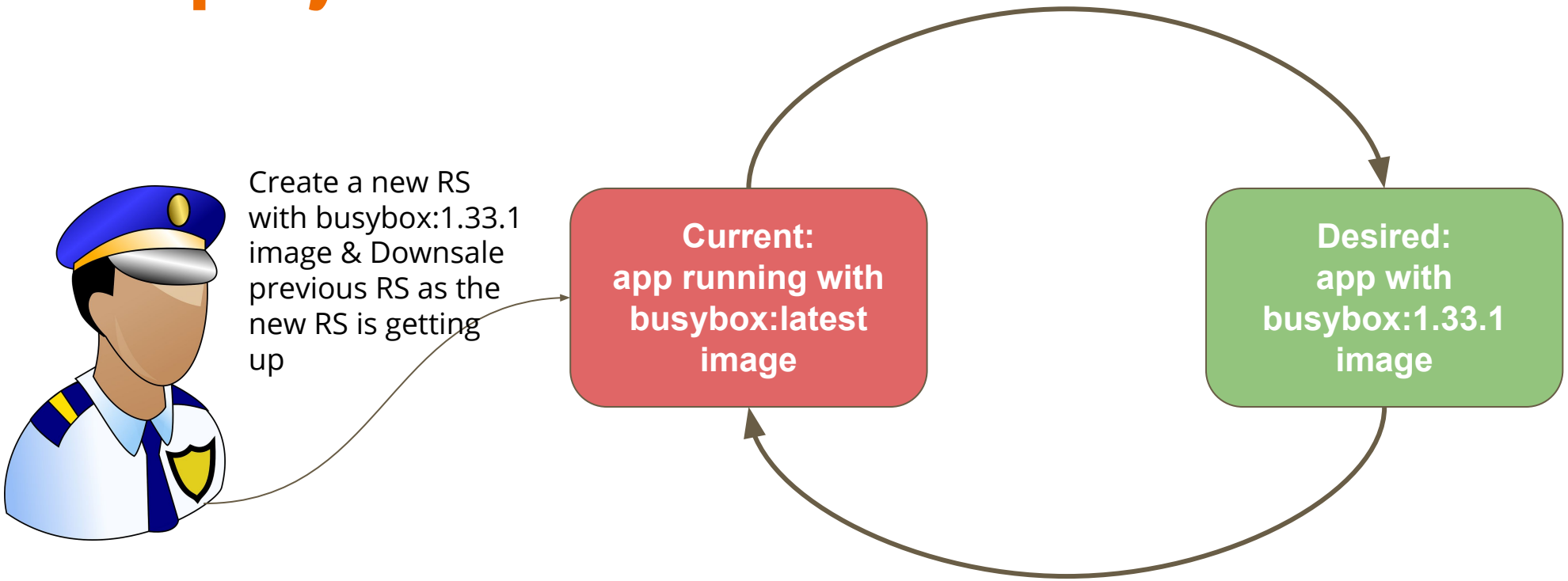
Deployment


How are updates handled?
Rolling/recreation
Deployment
Replica Set
Pod Spec
Container Spec
Container Spec
How many Pods should run?
Node selector
Service labels
Docker image
Environment variables
Storage Claims

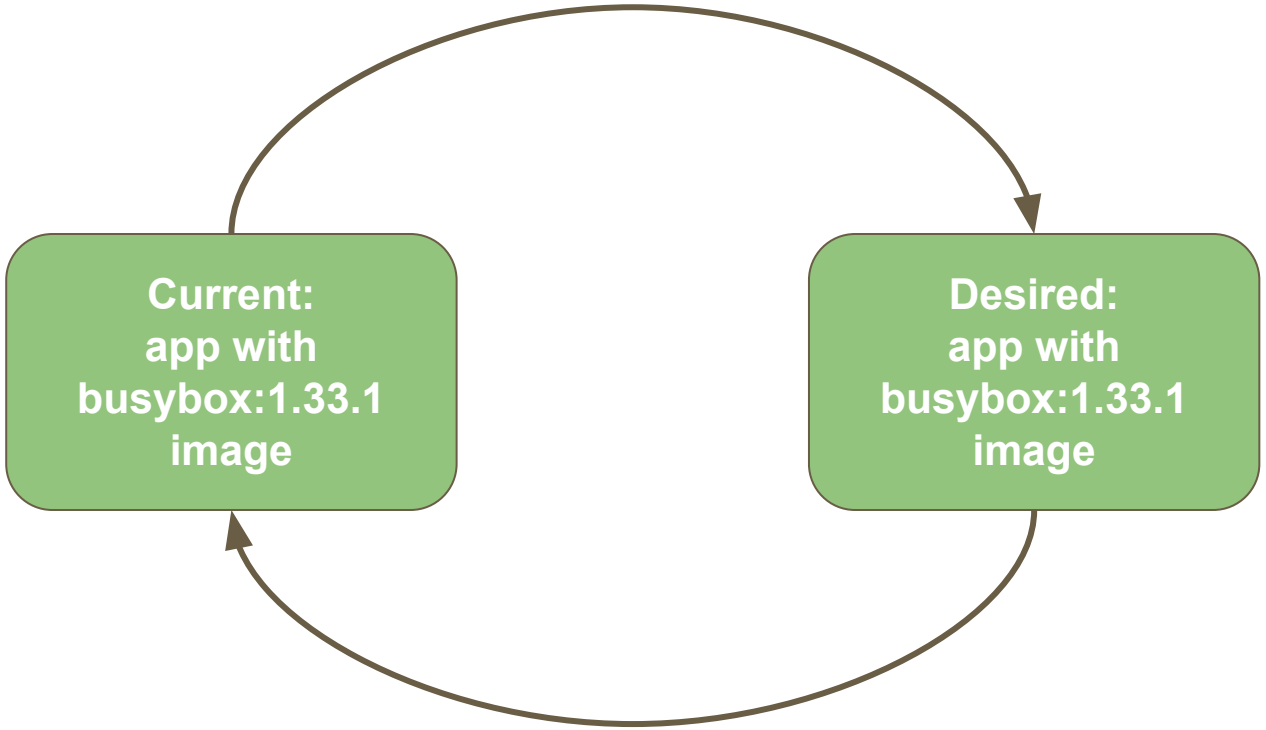
Deployment


Current:
app running with
busybox:latest
image
Desired:
app with
busybox:1.33.1
image

Deployment

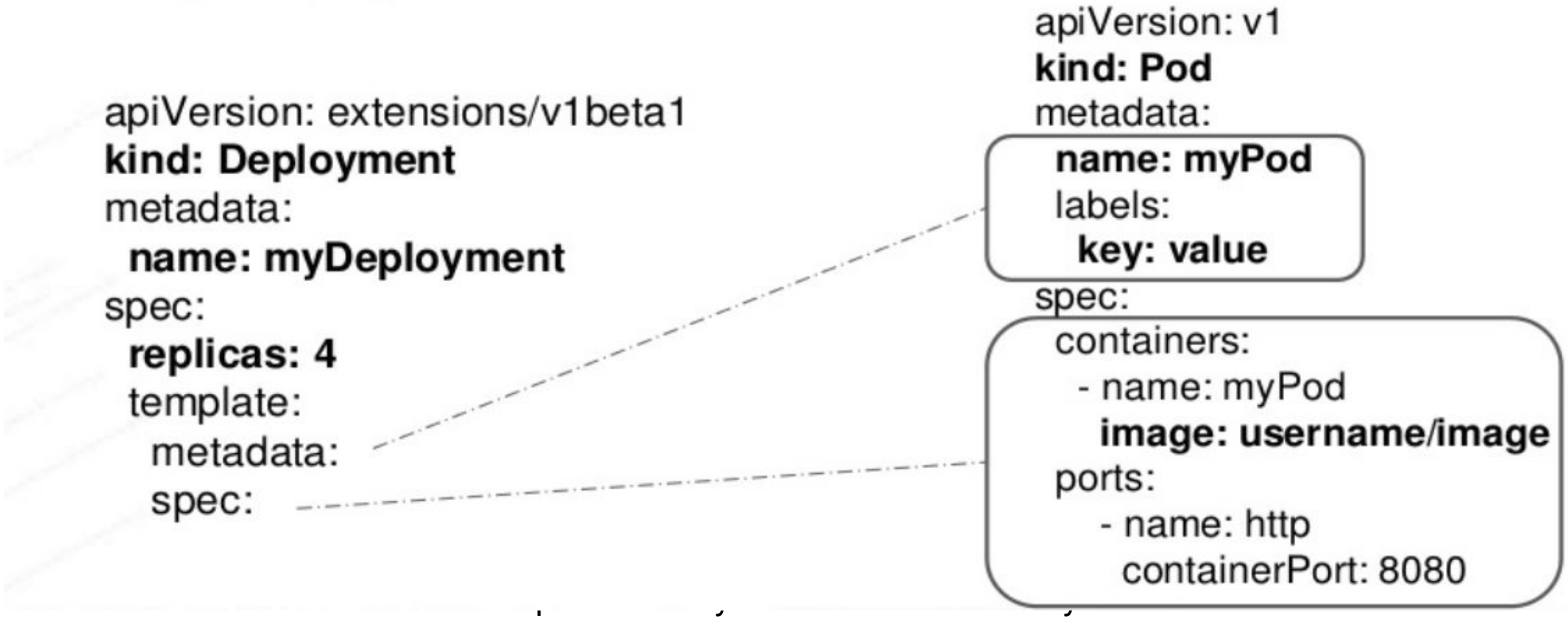

Create a new RS with busybox:1.33.1 image & Downsale previous RS as the new RS is getting up
Current: app running with busybox:latest image
Desired: app with busybox:1.33.1 image

Deployment


Current:
app with
busybox:1.33.1
image
Desired:
app with
busybox:1.33.1
image

Deployment

Defining a Deployment as YAML:



Lab: Deployment

There is a file named **deployment.yaml,** we will apply the Deployment by running

```
ali-kahoot@Alis-MacBook-Pro kubectl apply -f deployment.yaml
```

Now see the deployment, replicaset and pods by running:

```
ali-kahoot@Alis-MacBook-Pro kubectl get deployment,replicaset,pods
```

```
NAME                         READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/counter      1/1     1            1           58s

NAME                                    DESIRED   CURRENT   READY   AGE
replicaset.apps/counter-597c8bf5fc      1         1         1       58s

NAME                            READY   STATUS    RESTARTS   AGE
pod/counter-597c8bf5fc-dp649    1/1     Running   0          58s
```

Lab: Scale Deployment

```
ali-kahoot@Alis-MacBook-Pro kubectl scale deployment counter --replicas=3
ali-kahoot@Alis-MacBook-Pro kubectl get deployment,replicaset,pods
NAME                         READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/counter      1/3     3            1           2m12s

NAME                                    DESIRED   CURRENT   READY   AGE
replicaset.apps/counter-597c8bf5fc      3         3         1       2m12s

NAME                            READY   STATUS              RESTARTS   AGE
pod/counter-597c8bf5fc-dp649    1/1     Running             0          2m12s
pod/counter-597c8bf5fc-v5z2s    0/1     ContainerCreating   0          3s
pod/counter-597c8bf5fc-z7sqt    0/1     ContainerCreating   0          3s
```

LAB : ROLLING UPDATE A DEPLOYMENT

Now we will update the pod's image, and will see how deployment's rolling update works.

```
ali-kahoot@Alis-MacBook-Pro kubectl set image deployment/counter counter=busybox:1
```

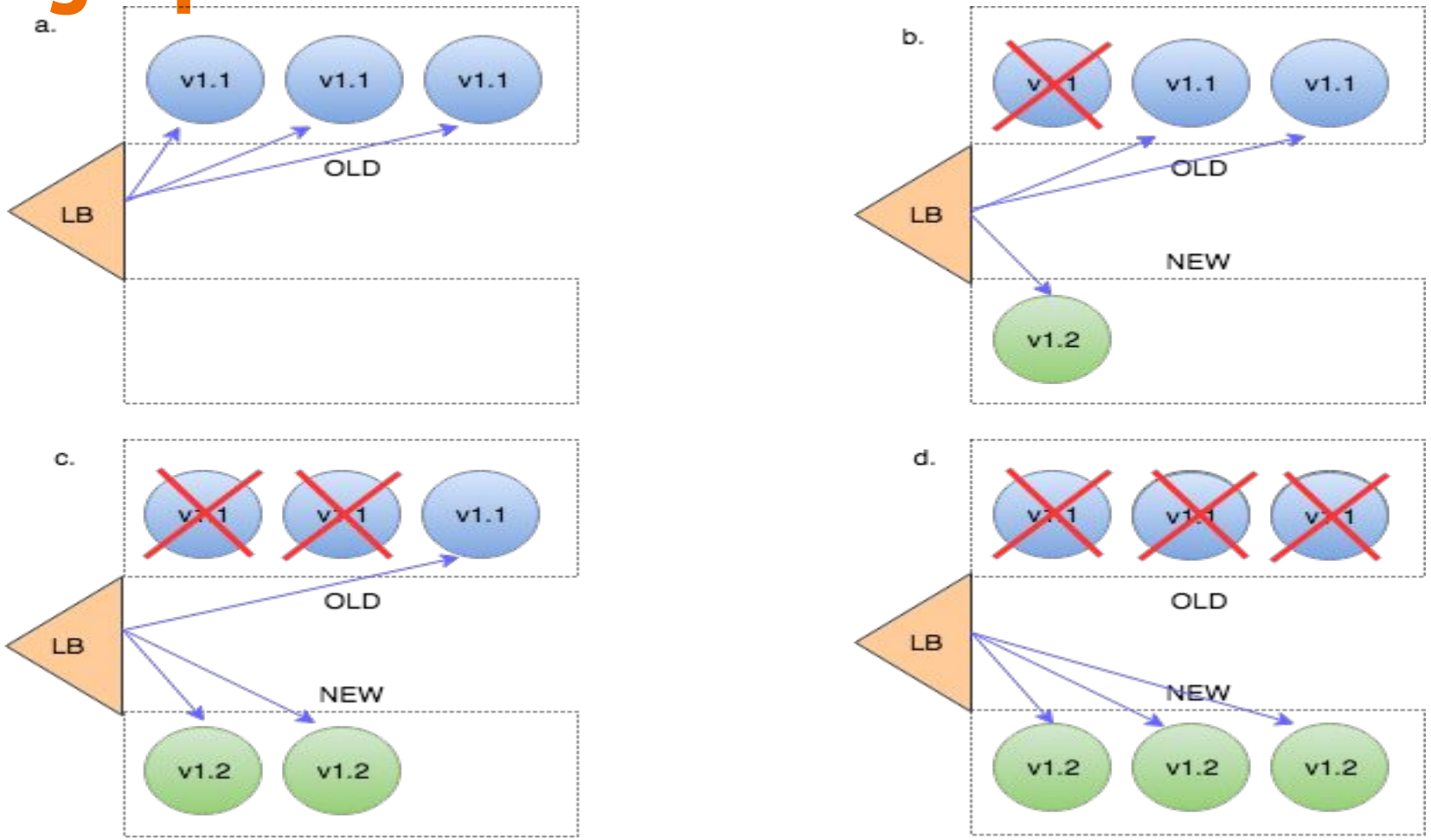
```
ali-kahoot@Alis-MacBook-Pro   kubectl get po
NAME                       READY   STATUS              RESTARTS   AGE
counter-597c8bf5fc-dp649   1/1     Running             0          25m
counter-597c8bf5fc-v5z2s   1/1     Terminating         0          23m
counter-597c8bf5fc-z7sqt   1/1     Terminating         0          23m
counter-6bc989b996-5jrrf   1/1     Running             0          13s
counter-6bc989b996-6g4xr   0/1     ContainerCreating   0          2s
counter-6bc989b996-ghc6z   1/1     Running             0          7s
```

Deployment Update Strategies

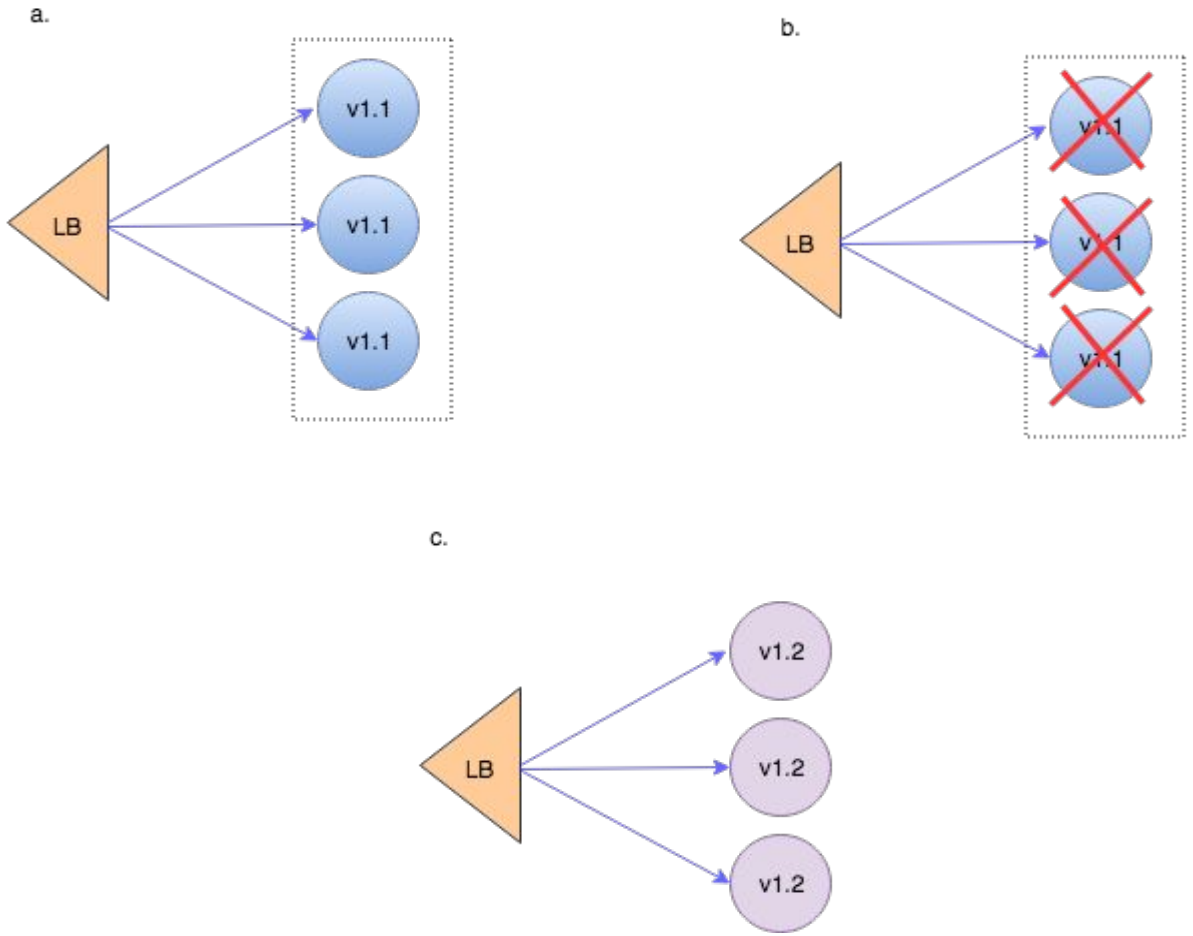
- Rolling Update: Create a new pod and if it works fine, then delete previous ones

- Recreate: Delete previous pods, then create new pods

Rolling Update



Recreate



LAB : ROLLING UPDATE AN ERRORED DEPLOYMENT

There is a file **deployment-rolling-update.yaml**, where we have specified the Update Strategy and configuration

```
ali-kahoot@Alis-MacBook-Pro kubectl delete -f deployment.yaml

ali-kahoot@Alis-MacBook-Pro kubectl apply -f
deployment-rolling-update.yaml

ali-kahoot@Alis-MacBook-Pro   kubectl get po

NAME                       READY   STATUS    RESTARTS   AGE
counter-597c8bf5fc-49jtz   1/1     Running   0          23s
```

LAB : ROLLING UPDATE AN ERRORED DEPLOYMENT

Now we will update the pod's image to a wrong image and will see how deployment's rolling update works and ensures High Availability.

```
ali-kahoot@Alis-MacBook-Pro kubectl set image deployment/counter counter=busybox:22
```

```
ali-kahoot@Alis-MacBook-Pro   kubectl get po
```

```
NAME                       READY   STATUS             RESTARTS   AGE
counter-597c8bf5fc-49jtz   1/1     Running            0          103s
counter-fd594659c-8z6mw    0/1     ImagePullBackOff   0          26s
```

LAB : ROLL BACK DEPLOYMENT TO PREVIOUS VERSION

```
ali-kahoot@Alis-MacBook-Pro kubectl rollout undo deployment counter

ali-kahoot@Alis-MacBook-Pro   kubectl get po

NAME                       READY   STATUS        RESTARTS   AGE
counter-597c8bf5fc-49jtz   1/1     Running       0          4m9s
counter-fd594659c-8z6mw    0/1     Terminating   0          2m52s
```

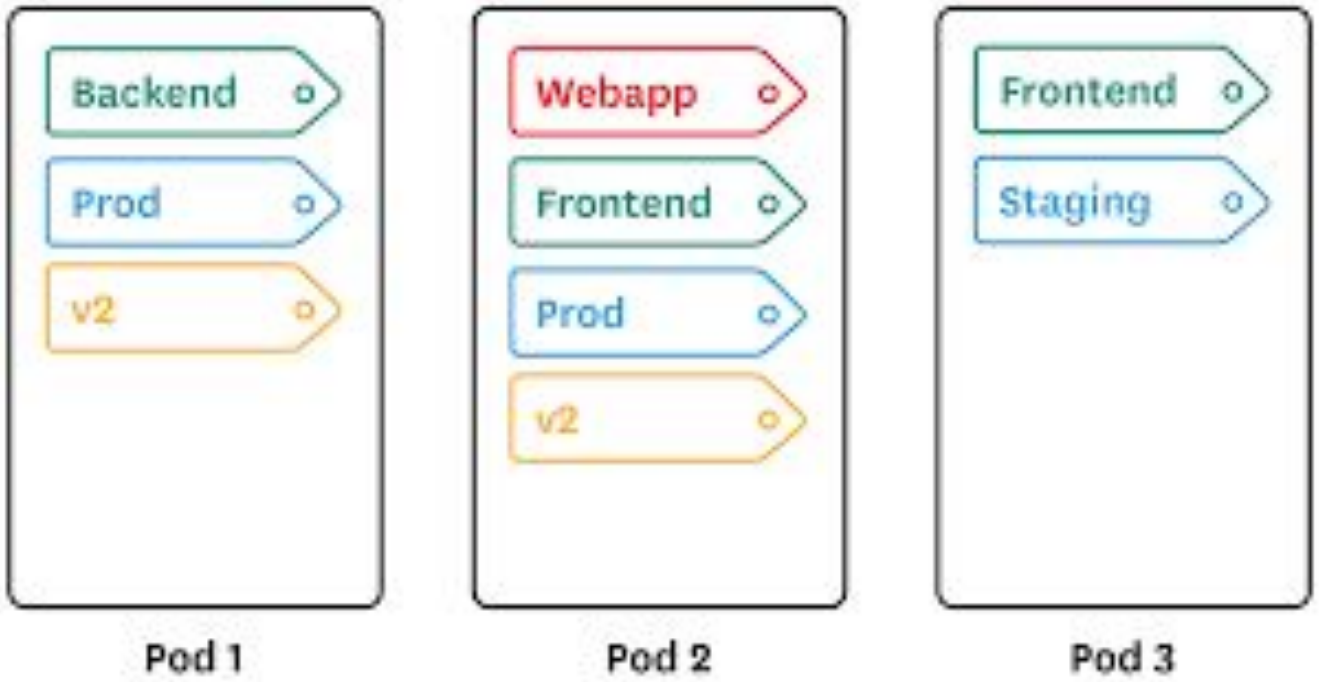
LABELS

- a key: value pair for any of k8s resources.
- Useful for selecting objects based on labels out of 100s of resources
- Useful for querying dynamic objects e.g. pods get deleted & added so easy to query set of pods with same labels
- Map your own organizational structures onto system objects in a loosely coupled fashion

LABELS

- K8s is based on Labels & Controllers itself
- Every resource in K8s can have Labels
- Examples
  - "release" : "stable", "release" : "canary"
  - "environment" : "dev", "environment" : "qa", "environment" : "production"
  - "tier" : "frontend", "tier" : "backend", "tier" : "cache"

LABELS


Backend
Prod
v2
Webapp
Frontend
Prod
v2
Frontend
Staging
Pod 1
Pod 2
Pod 3

LABEL SELECTORS

- Unlike names and UIDs, labels do not provide uniqueness.
- We expect many objects to carry the same label(s)
- Via a label selector, the client/user can identify a set of objects. The label selector is the core grouping primitive in Kubernetes.
- In this way you can chose any resource i.e. NodeSelectors, Pods with specific labels, Services with Specific Labels, etc

LAB: LABELS

```
ali-kahoot@Alis-MacBook-Pro kubectl get po -l app=counter
NAME                       READY   STATUS    RESTARTS   AGE
counter-597c8bf5fc-49jtz   1/1     Running   0          9m26s

ali-kahoot@Alis-MacBook-Pro kubectl get po -l author=Ali-Kahoot
NAME                       READY   STATUS    RESTARTS   AGE
counter-597c8bf5fc-49jtz   1/1     Running   0          9m26s

ali-kahoot@Alis-MacBook-Pro   kubectl get deploy -l tier=backend
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
counter   1/1     1            1           10m
```

ANNOTATIONS

Again a key: value pair for any of k8s resources.

Labels are used for selecting resources by LabelSelectors, Annotations are not used for that purpose by native Kubernets API

Attach any other metadata to Kubernetes objects

Annotations can be used by custom applications to do some functionality

SERVICES

- An abstract way to expose an application running on a set of Pods as a network service.
- Pods get deleted or recreated or scaled up or scaled down dynamically, so how can one expose them, and how to load balance between them? Similarly, how will a Frontend connect to Backend pod?

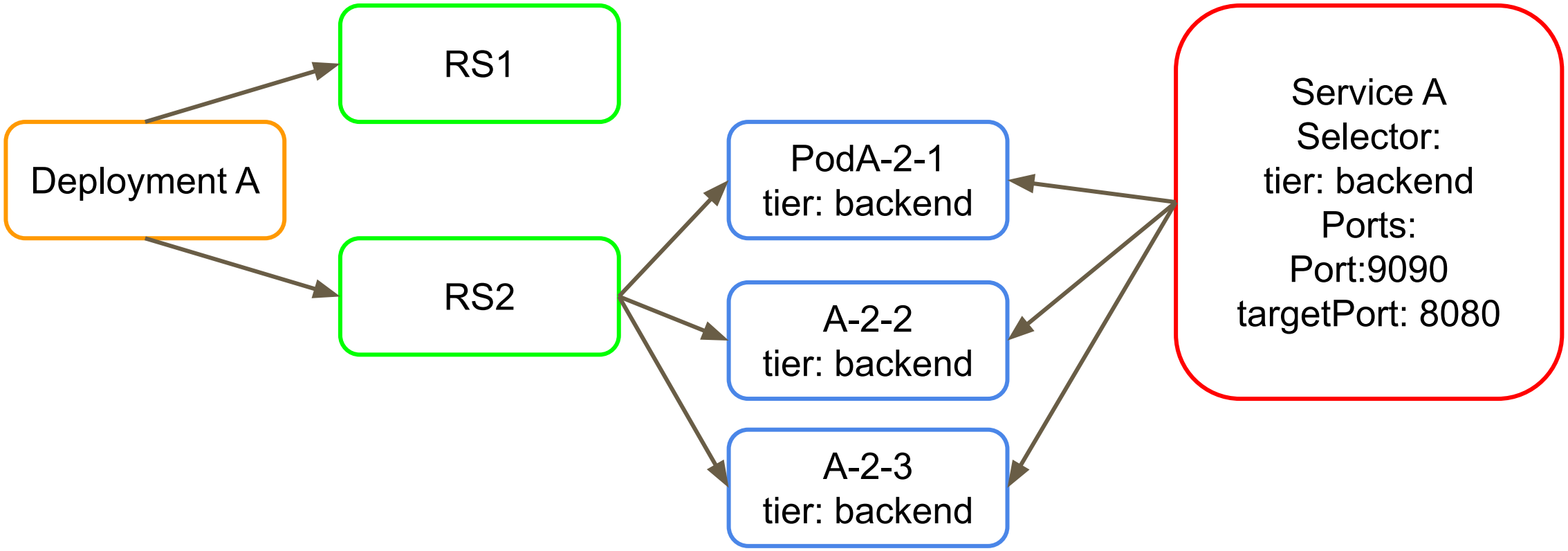
SERVICES

- Service is an abstraction which denes a logical set of Pods and a policy by which to access them
- Frontends do not care which of the backend pods they use, the frontend clients should not need to be aware of that, nor should they need to keep track of the set of backends themselves

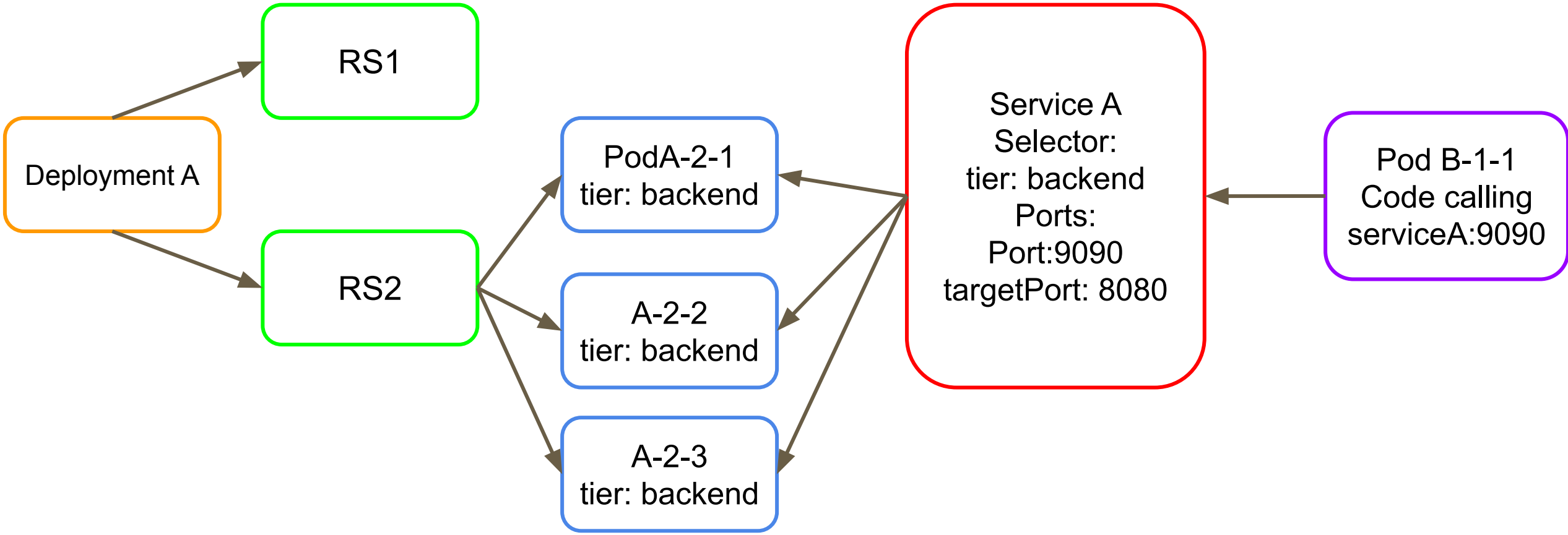
Service

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
  - protocol: TCP
    port: 8080
    targetPort: 9376
  - protocol: TCP
    port: 8081
    targetPort: 9377
```

Service


RS1
Deployment A
RS2
PodA-2-1
tier: backend
A-2-2
tier: backend
A-2-3
tier: backend
Service A
Selector:
tier: backend
Ports:
Port:9090
targetPort: 8080

Service


RS1
Deployment A
RS2
PodA-2-1
tier: backend
A-2-2
tier: backend
A-2-3
tier: backend
Service A
Selector:
tier: backend
Ports:
Port:9090
targetPort: 8080
Pod B-1-1
Code calling
serviceA:9090

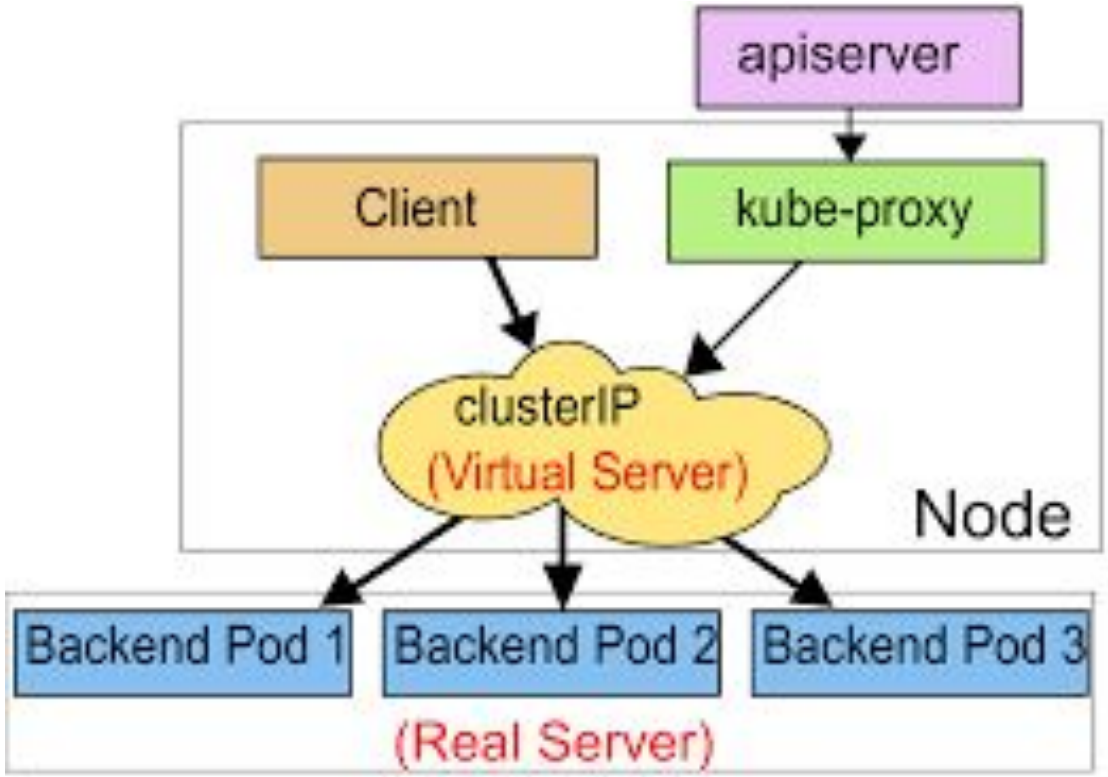
Service Types

You might need to expose some of your service inside the cluster only(backends & databases) and some to the external world(frontends), so there are different type of Services for that

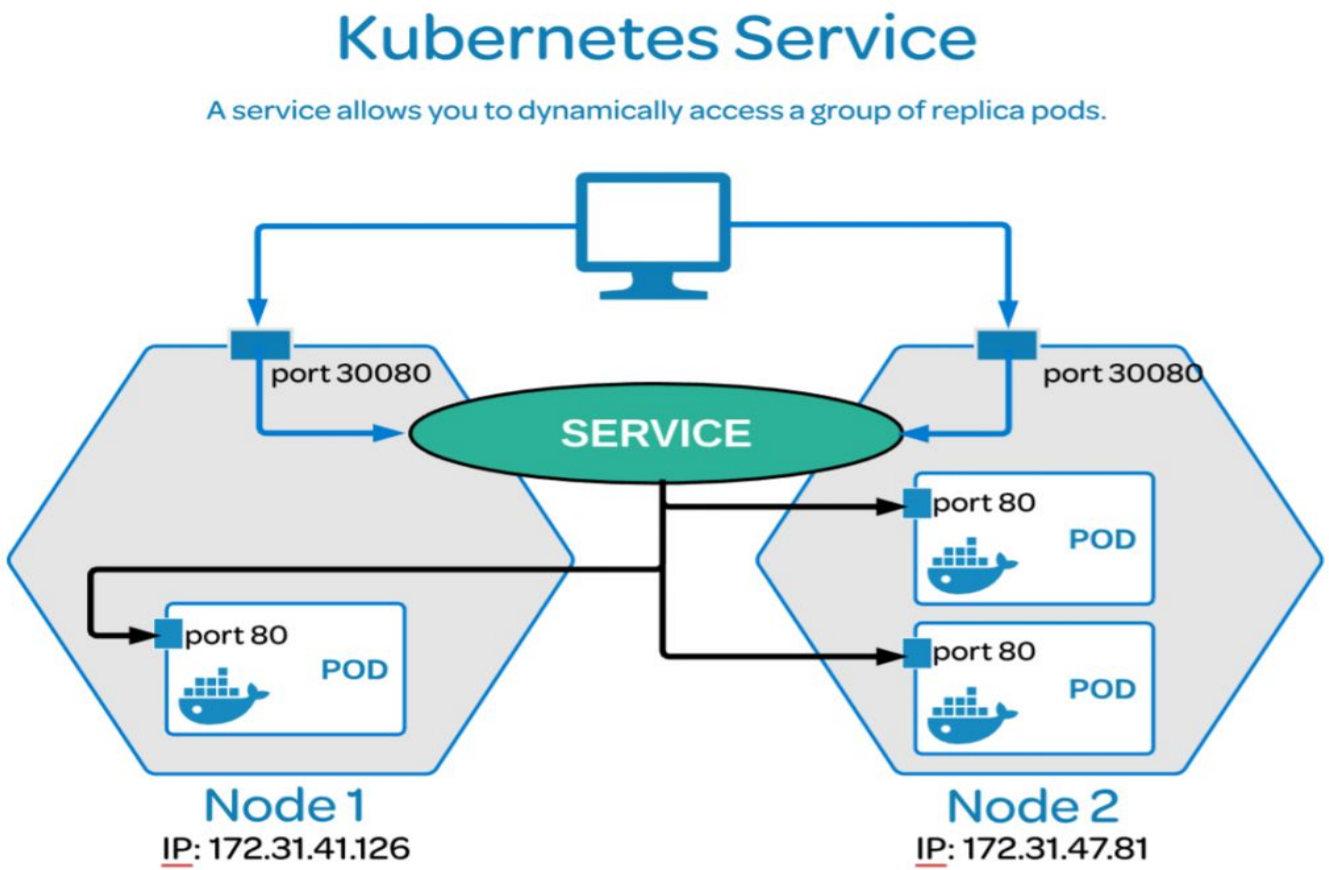
- ClusterIP: Exposes the Service on a cluster-internal IP. Choosing this value makes the Service only reachable from within the cluster. This is the default ServiceType

- NodePort: Exposes the Service on each Node's IP at a static port (the NodePort)

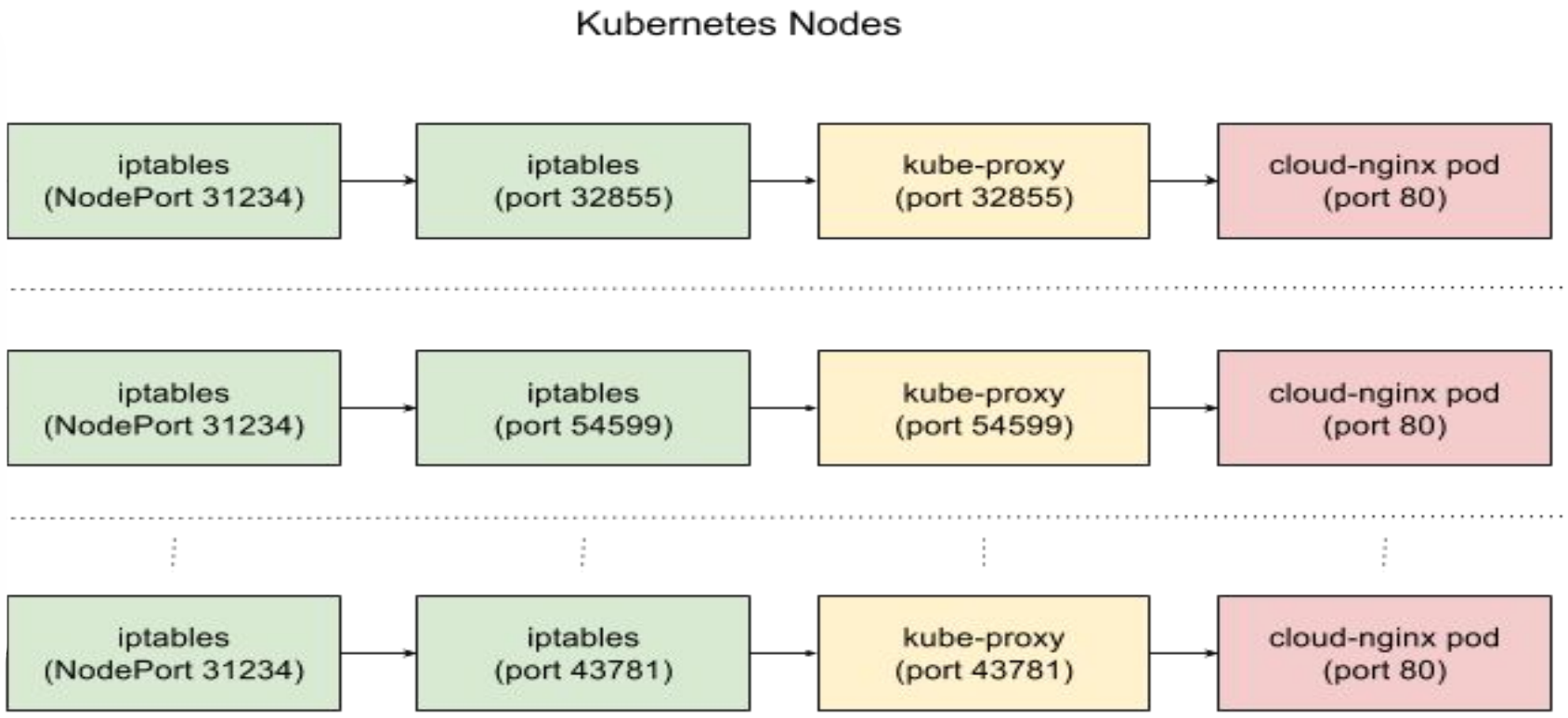
Service


apiserver
Client
kube-proxy
clusterIP
(Virtual Server)
Node
Backend Pod 1
Backend Pod 2
Backend Pod 3
(Real Server)

NodePort


Kubernetes Service
A service allows you to dynamically access a group of replica pods.
port 30080
SERVICE
port 30080
port 80
POD
port 80
POD
port 80
POD
Node 1
IP: 172.31.41.126
Node 2
IP: 172.31.47.81

NodePort

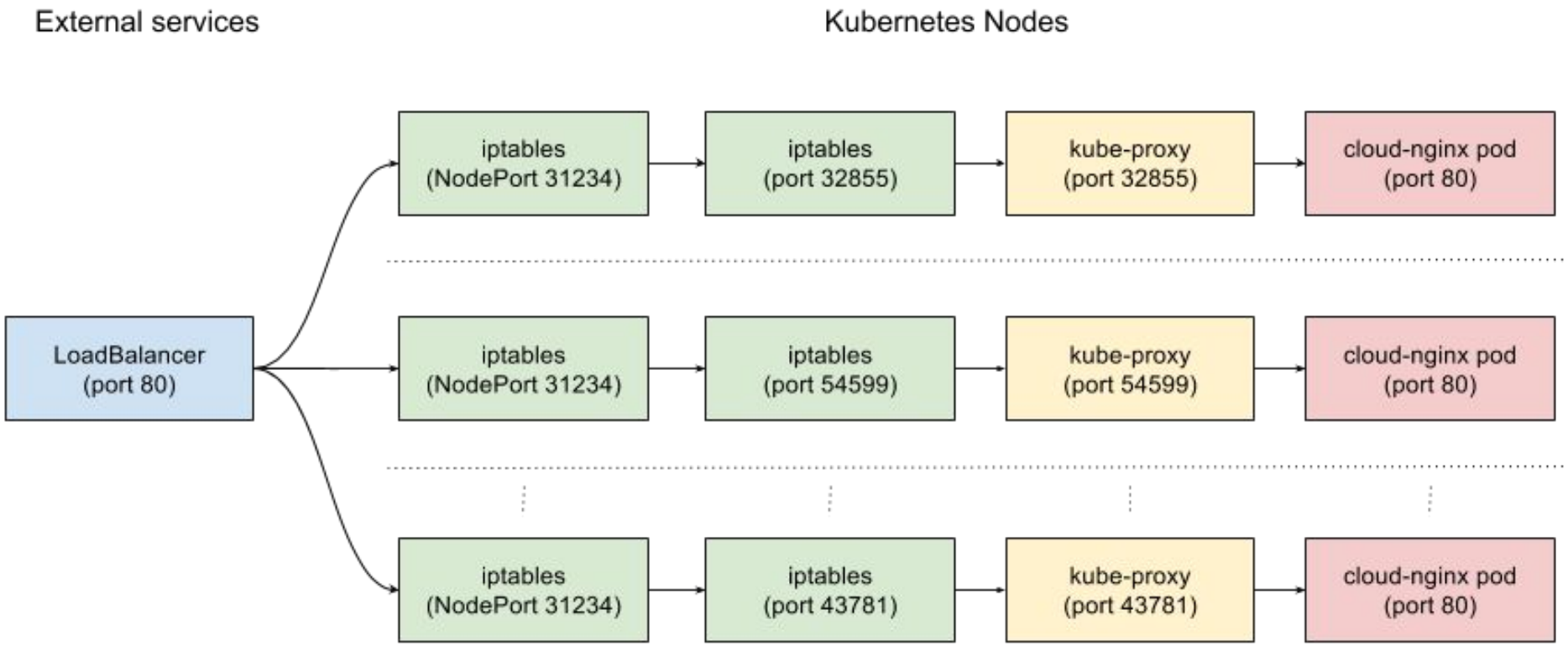

Kubernetes Nodes
iptables (NodePort 31234)
iptables (port 32855)
kube-proxy (port 32855)
cloud-nginx pod (port 80)
iptables (NodePort 31234)
iptables (port 54599)
kube-proxy (port 54599)
cloud-nginx pod (port 80)
iptables (NodePort 31234)
iptables (port 43781)
kube-proxy (port 43781)
cloud-nginx pod (port 80)

Service Types

- LoadBalancer: Exposes the Service externally using a cloud provider's load balancer.

- ExternalName: Maps the Service to the contents of the externalName eld (e.g. foo.bar.example.com), by returning a CNAME record with its value. No proxying of any kind is set up.

Load Balancer



Lab: Service

There is a file **golang-api.yaml & golang-svc.yaml**, where we have specified the Deployment & Service

```
ali-kahoot@Alis-MacBook-Pro kubectl apply -f golang-api.yaml
```

```
ali-kahoot@Alis-MacBook-Pro kubectl apply -f golang-svc.yaml
```

The above service will select the pods with label **app: application** and forward all the requests on the service's port 9090 to the pod's 8080 port. We are using NodePort so it will be exposed on the Minikube node's port 32000. So we need to find the minikube ip, run

```
minikube ip
```

It will give you an ip, so in browser go to: <minikube ip>:32000/hello

LAB : MICROSERVICES (FRONTEND & BACKEND)

We will be deploying the frontend & backend in Kubernetes using Services & Deployments.

There are files backend.yaml and frontend.yaml. Run them by applying

```
kubectl apply -f backend.yaml
```

It will create a deployment with 2 replicas and a service which is ClusterIP type corresponding to the 2 pods.

```
kubectl apply -f frontend.yaml
```

It will create a deployment with 2 replicas and a service which is NodePort type corresponding to the 2 pods. This frontend calls backend:8080 in its code.

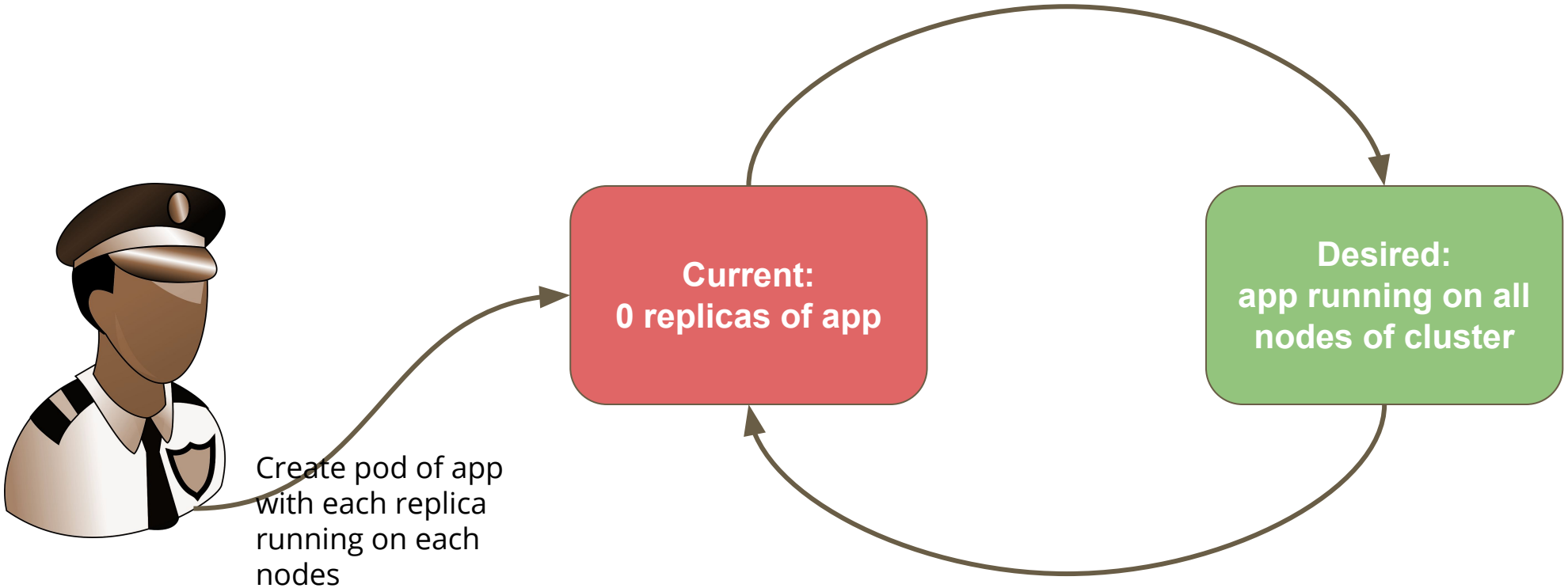
LAB : MICROSERVICES (FRONTEND & BACKEND)

```
Now go to browser and open
<minikube-ip>:31000/hello
<minikube-ip>:31000/instructor/1 or 2 or 3
```

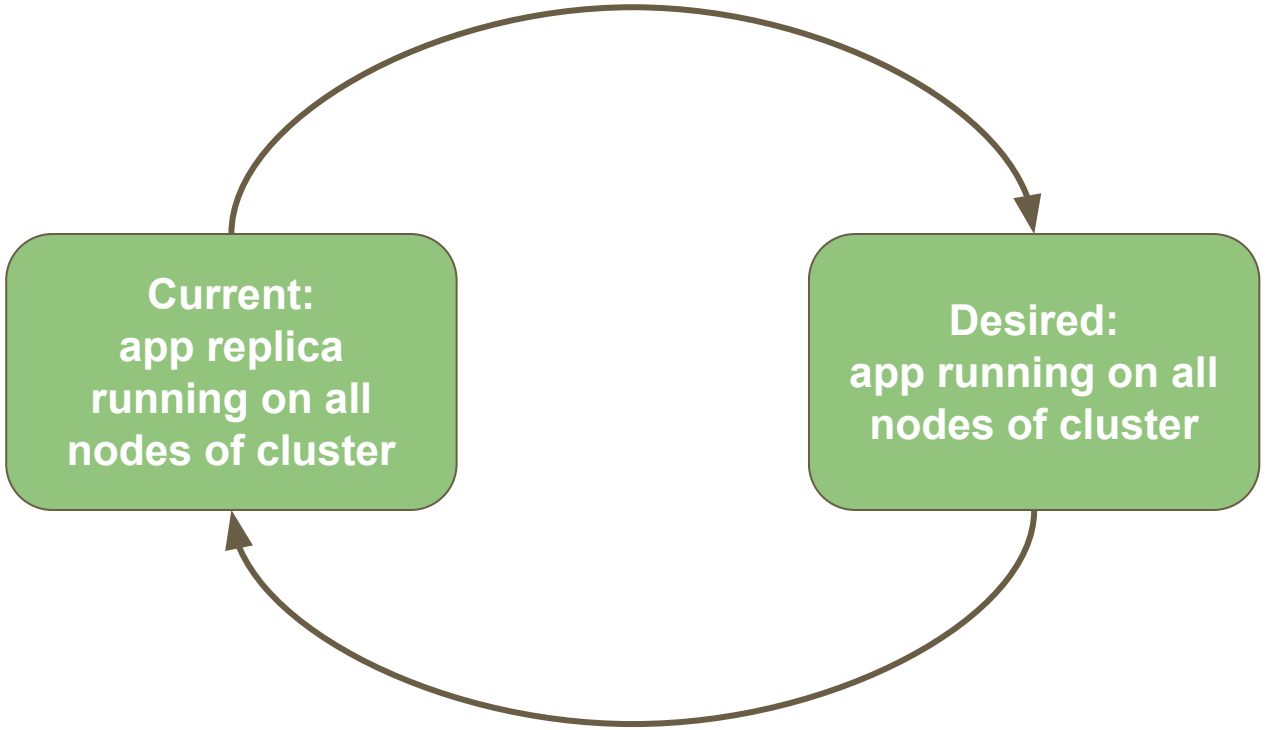
Daemonset

- Almost similar to Replicaset
- Each replica run on each node
- Provides Rolling update
- Can’t do Rollback
- Use Case: Monitoring Exporters, Logging Exporters, etc

Daemonset


Current:
0 replicas of app
Desired:
app running on all
nodes of cluster
Create pod of app
with each replica
running on each
nodes

Daemonset


Current:
app replica
running on all
nodes of cluster
Desired:
app running on all
nodes of cluster

StatefulSets

- They are only used for apps with persistence

Persistence in Kubernetes

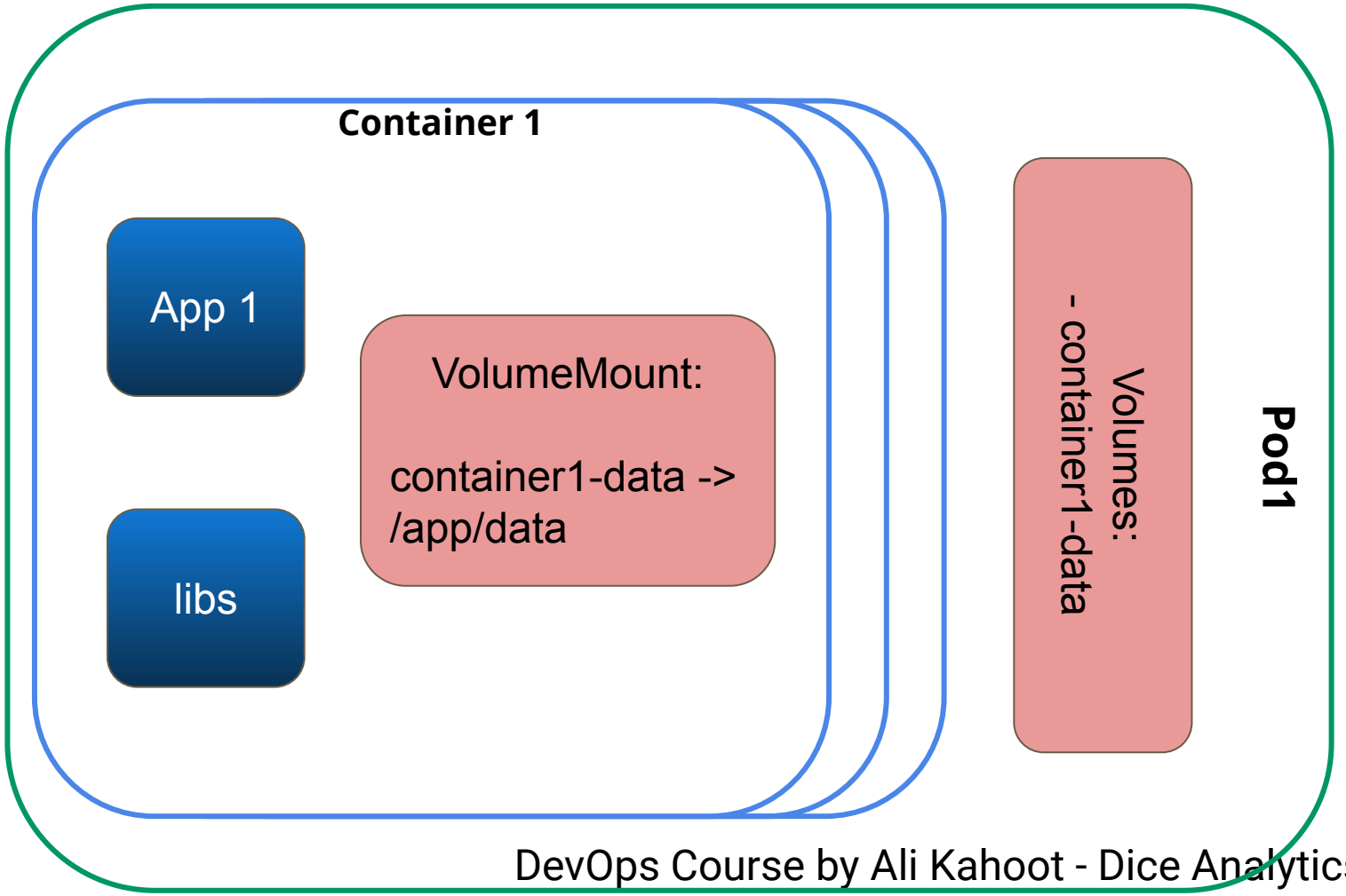
Counter App

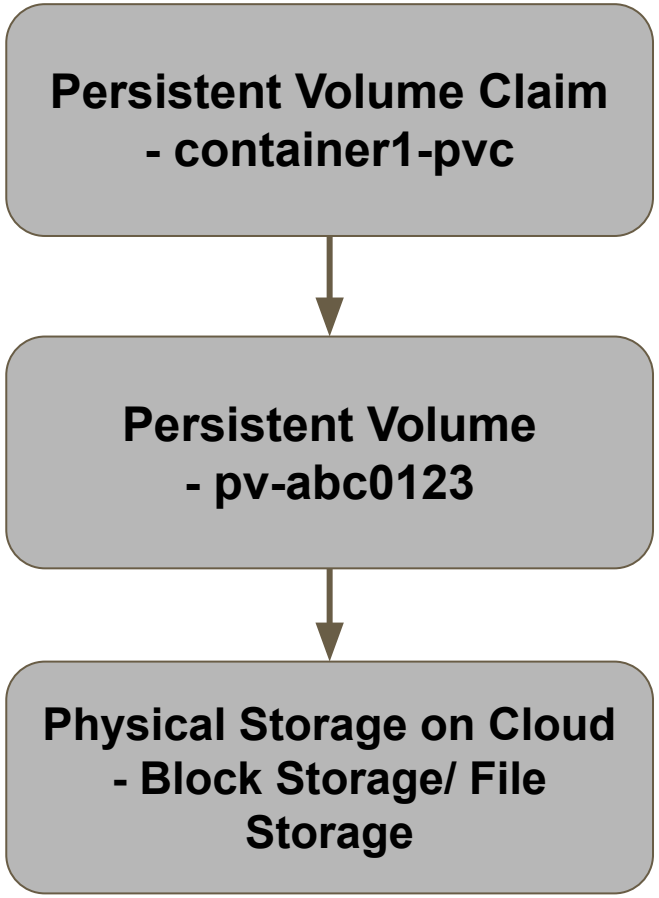
A Simple Application that reads from a file the last line, increments the number and writes back to the file, and sleeps for a random time for 1-3 seconds and then again does the above steps.

Available at:
https://github.com/kahootali/counter-app

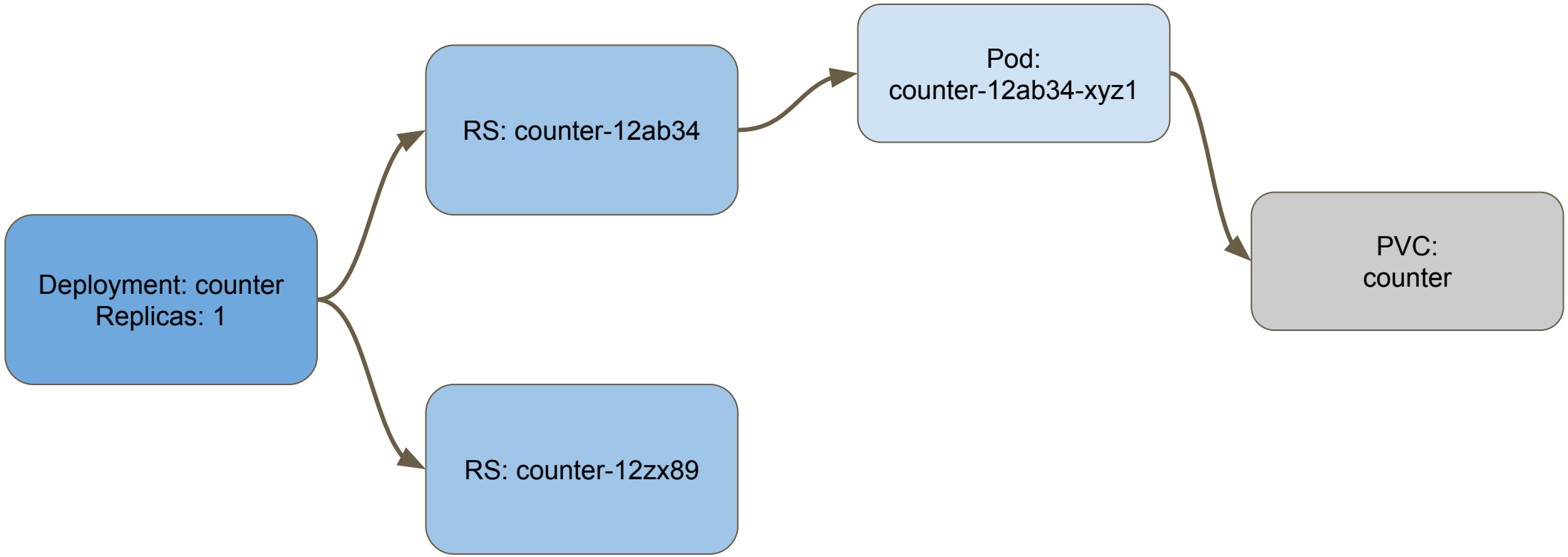
```
cd app/
touch counter.txt
declare -i var=$(cat counter.txt)
for (( ; ; ))
do
  cat counter.txt
  var=$((var + 1))
  echo $var >counter.txt
  sleepSeconds=$(echo $((1 + $RANDOM
%3)))
  sleep $sleepSeconds
done
```

Persistence

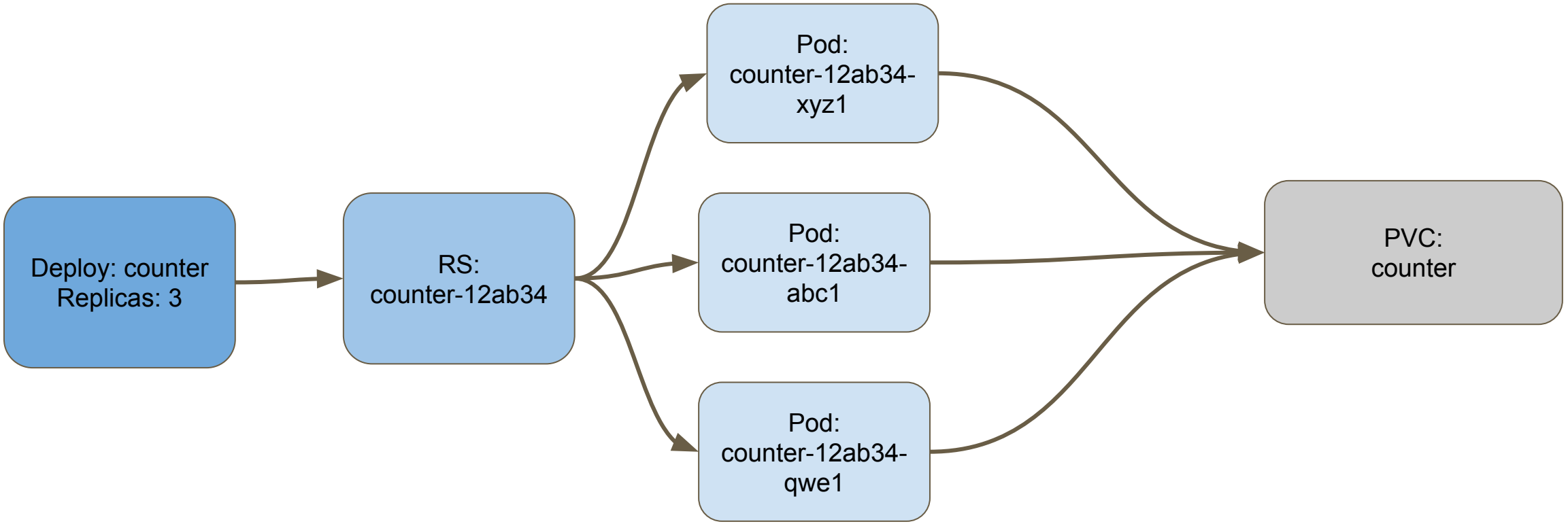




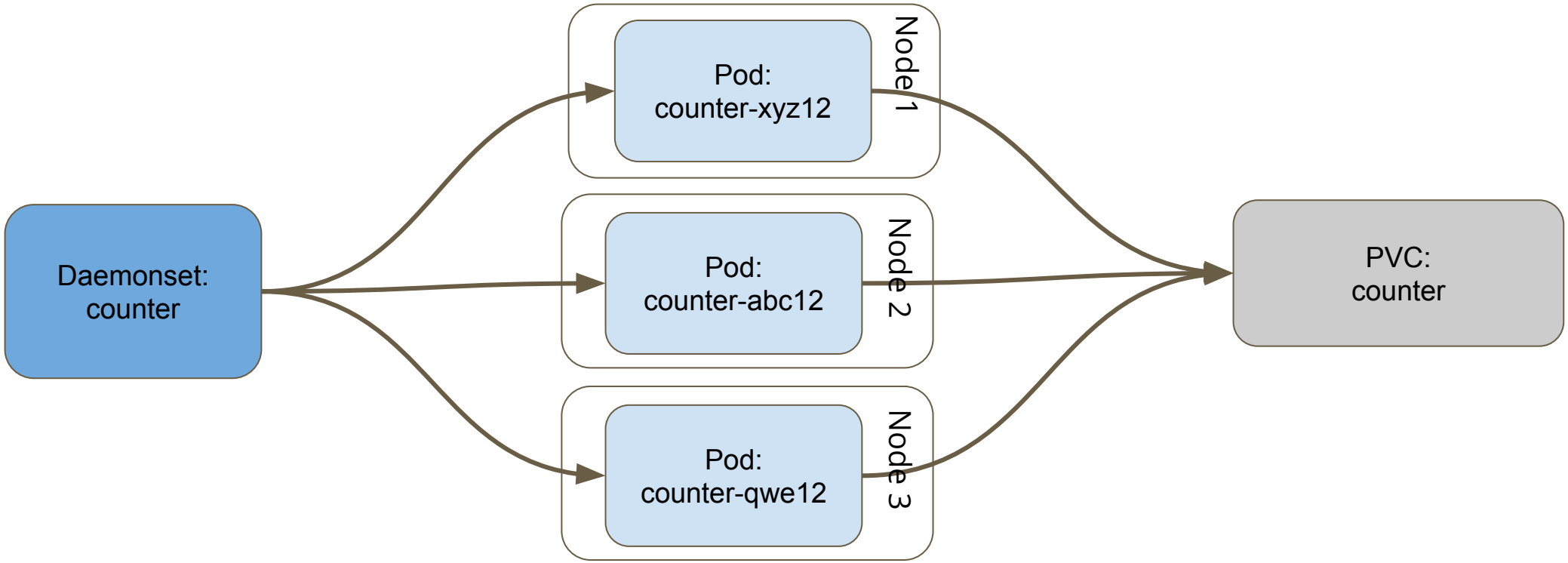
Persistence in Deployments



Persistence in Deployments



Persistence in Daemonsets

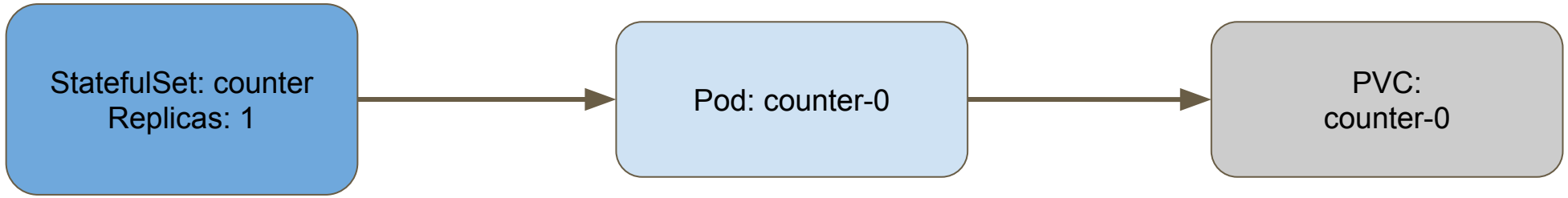


Each Replica running on each node

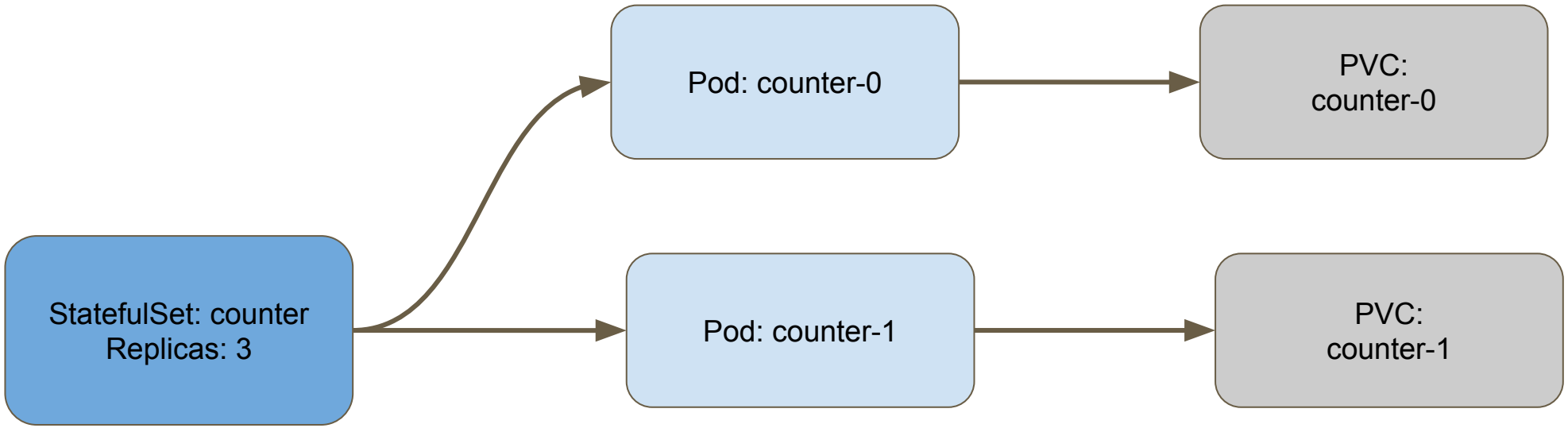
Statefulsets

- Manage Stateful Applications
- Declare PVC template inside a Statefulset manifest
- Guarantees the ordering & uniqueness of pods by having incremental naming convention
- Unlike Deployments, the replica pods are not interchangeable
- Each replica has its own state
- Useful for Databases, etc

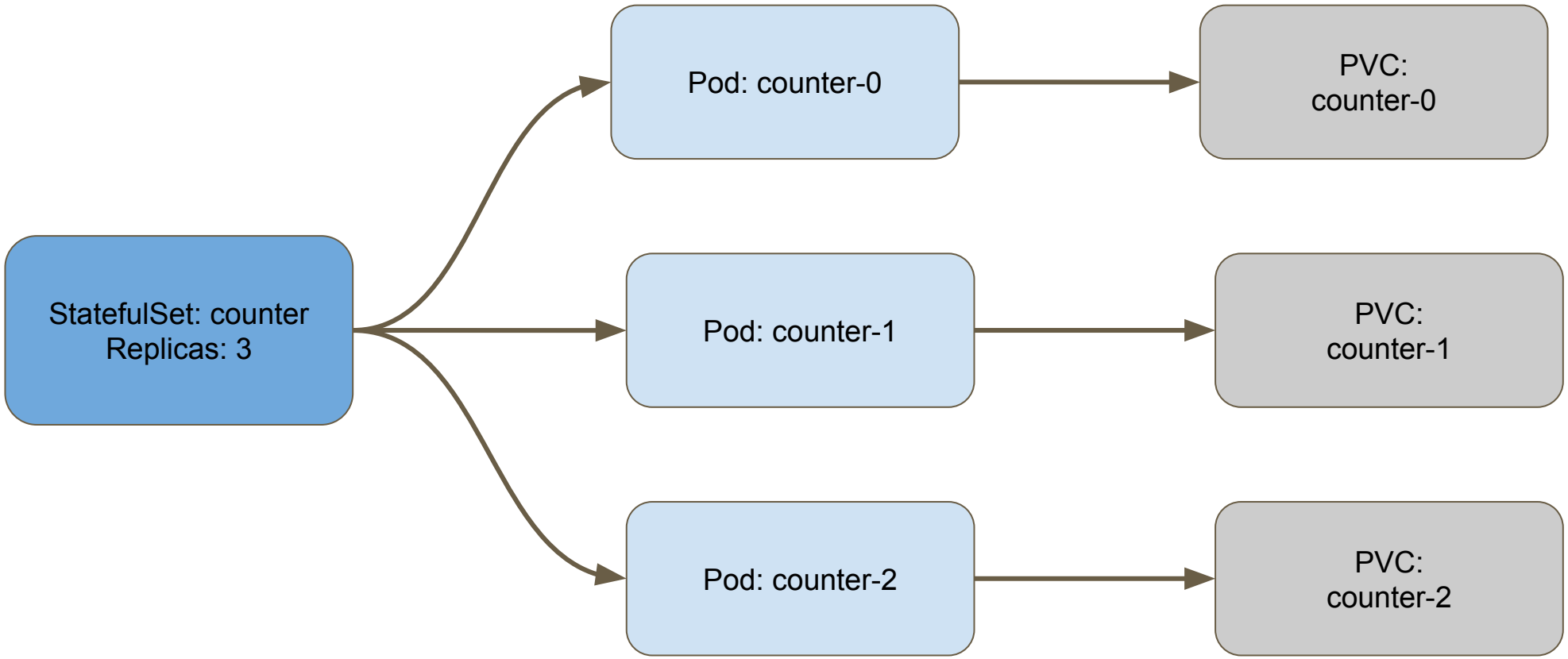
Persistence in Statefulsets



Persistence in Statefulsets



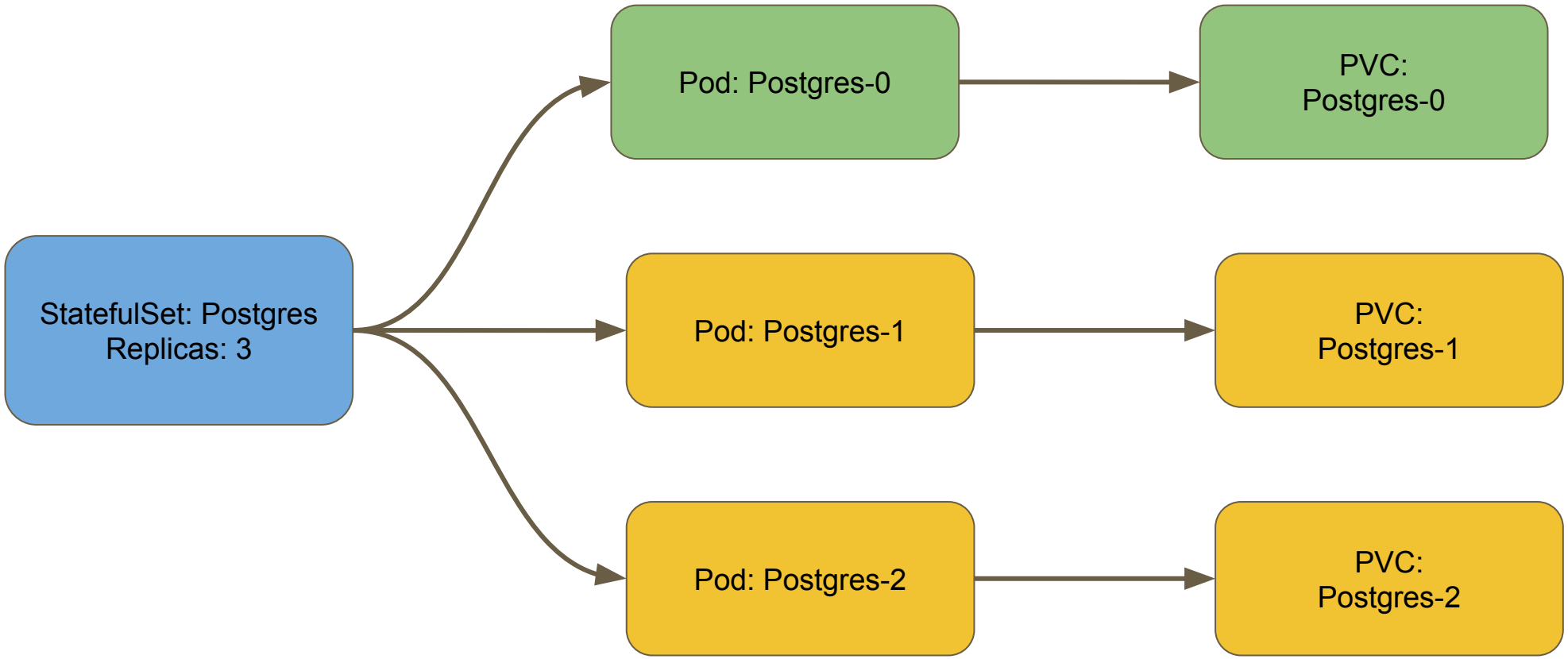
Persistence in Statefulsets



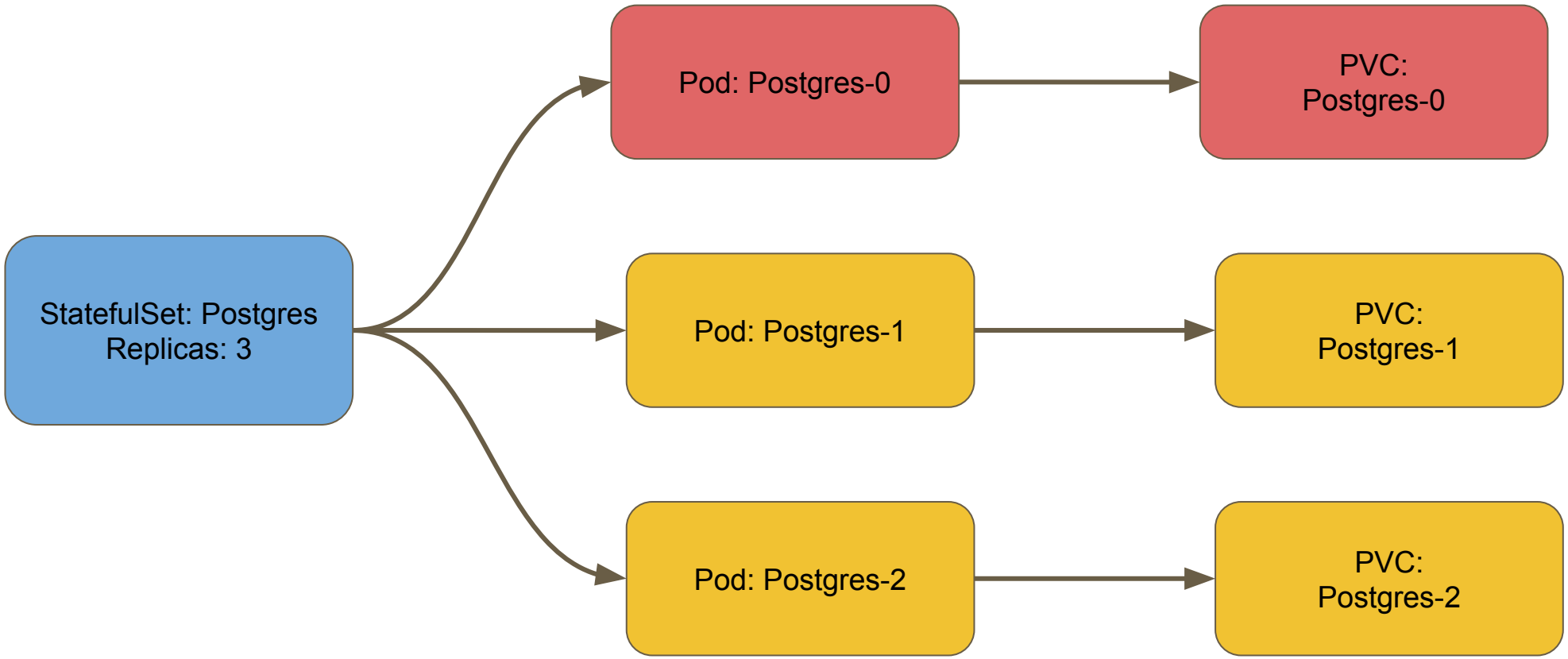
Deployments vs StatefulSet

- If we use Deployment for Databases/Stateful applications, data consistency can be compromised
- High Availability cannot be achieved, as if the PVC is deleted, the data is gone
- StatefulSets don’t have Replicasets so no Rollback option

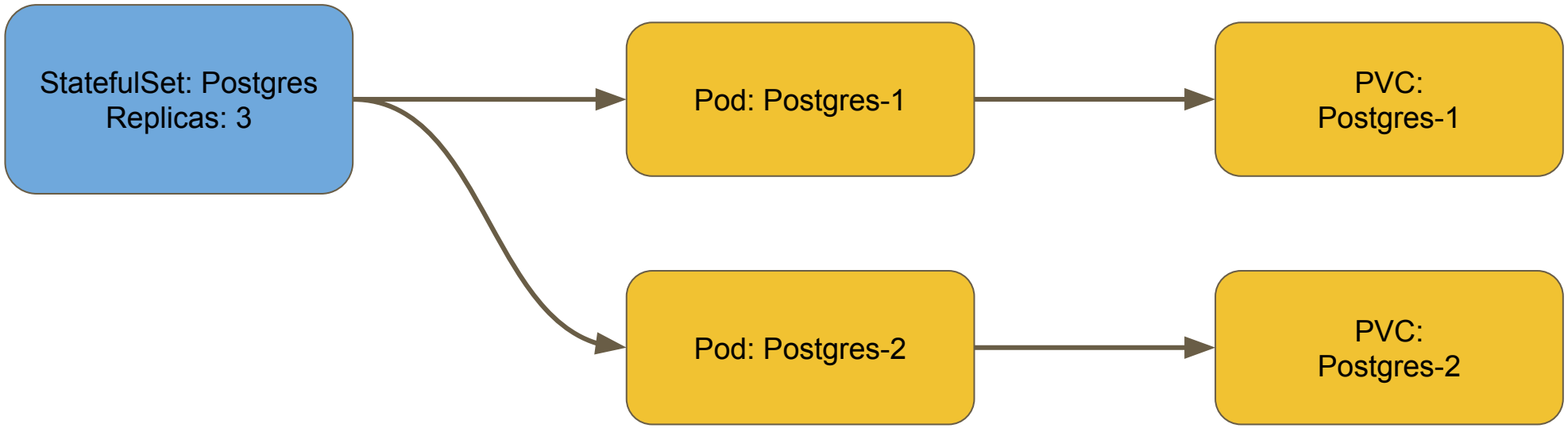
Using DBs as clusters for HA



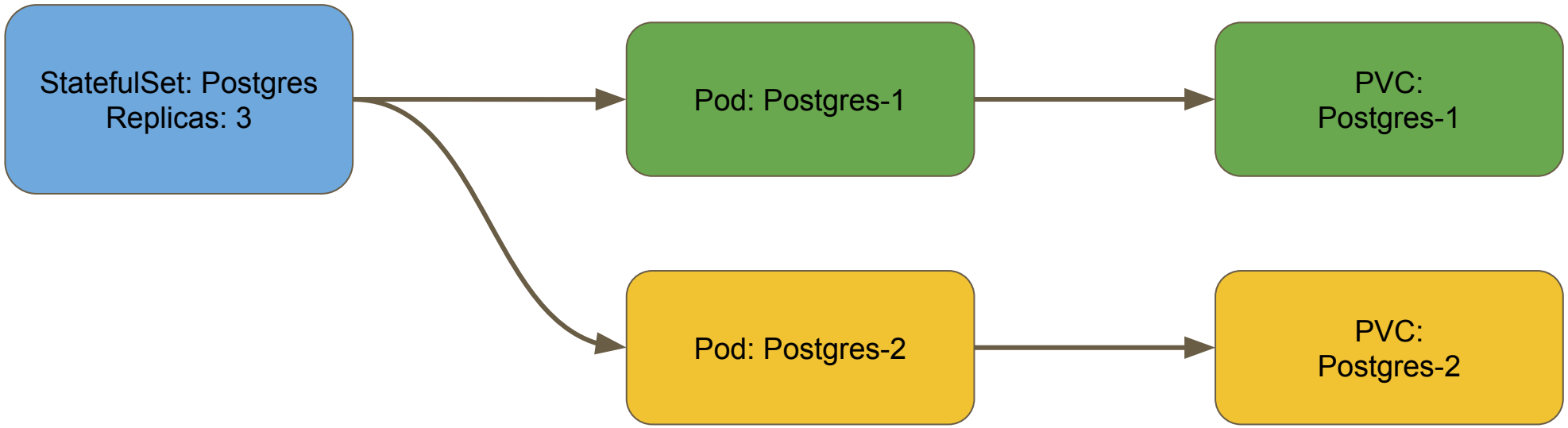
Using DBs as clusters for HA



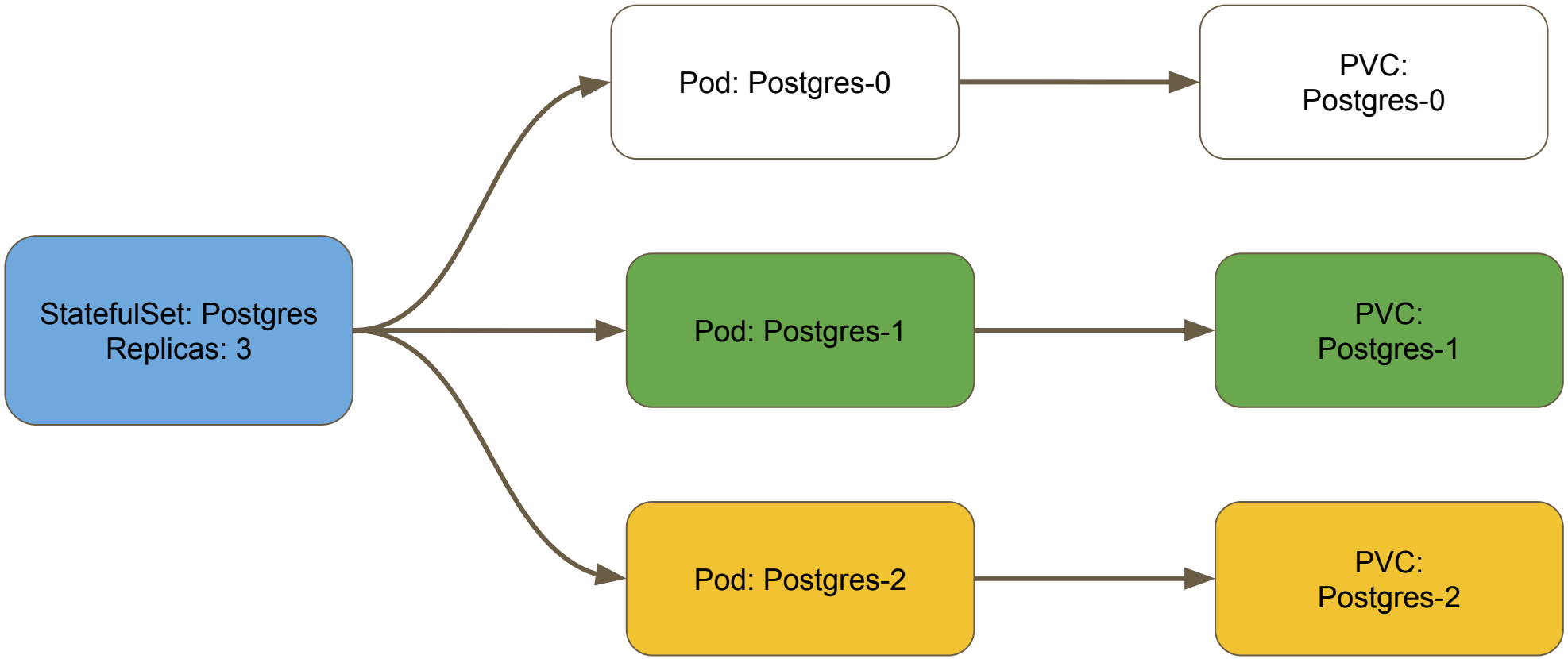
Using DBs as clusters for HA



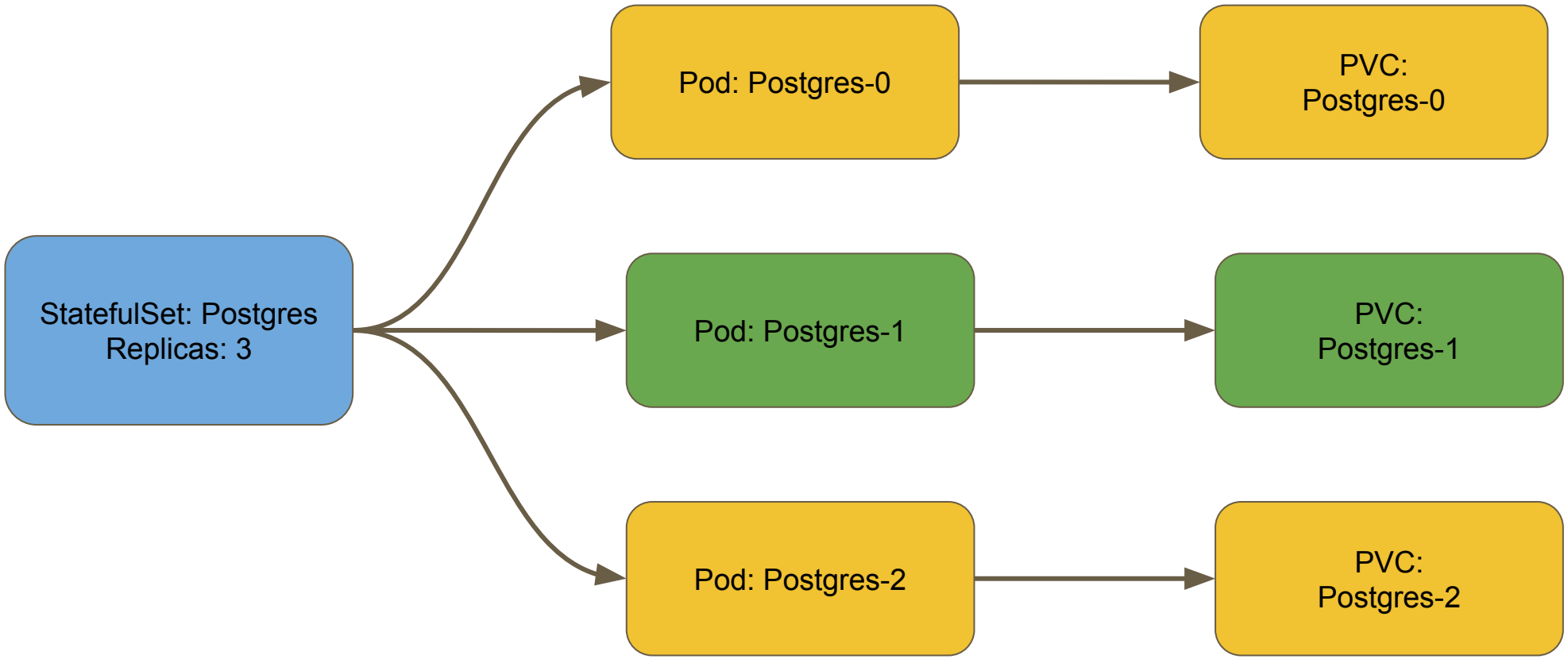
Using DBs as clusters for HA



Using DBs as clusters for HA



Using DBs as clusters for HA



Why Statefulsets for DBs

- Databases form clusters(primary/secondary replicas) to provide High Availability
- Each individual pod should have a unique identity
- As the cluster of DB communicates with each other so they should be able to predict other pod names.
- Even if one pod goes down,it will come back with the same name as before so networking will be easy
- PVC doesn’t delete automatically even if Pods/StatefulSet delete