leboncoin

k8s workshop - 1

Introduction & concepts walkthrough

2017-09-07

Agenda

1. **The Kubernetes project**
   Community and maturity

2. **K8s cluster**
   Overview and components

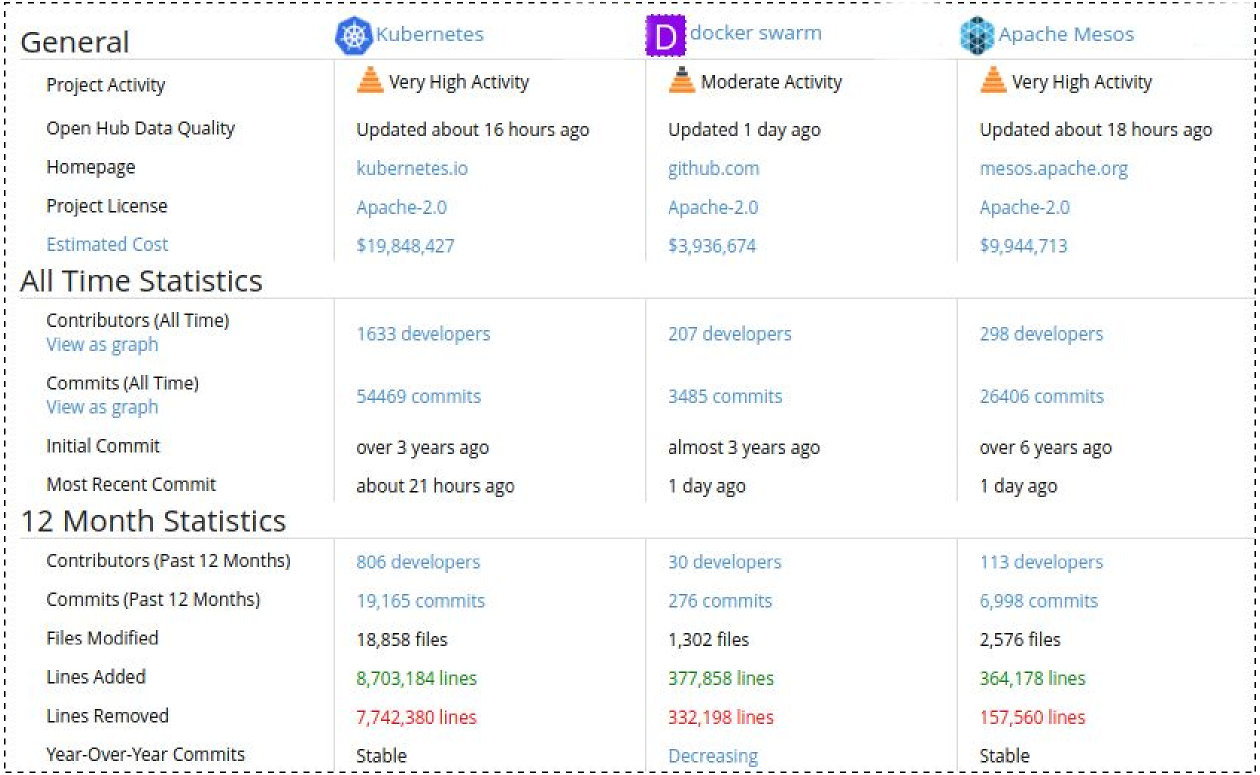
3. **Basic resources**
   Pods, Deployments, Services, Ingress

4. **Advanced concepts**
   Config, secrets, volumes

5. **Tips**
   Pattern & Best practices

1 - k8s Project

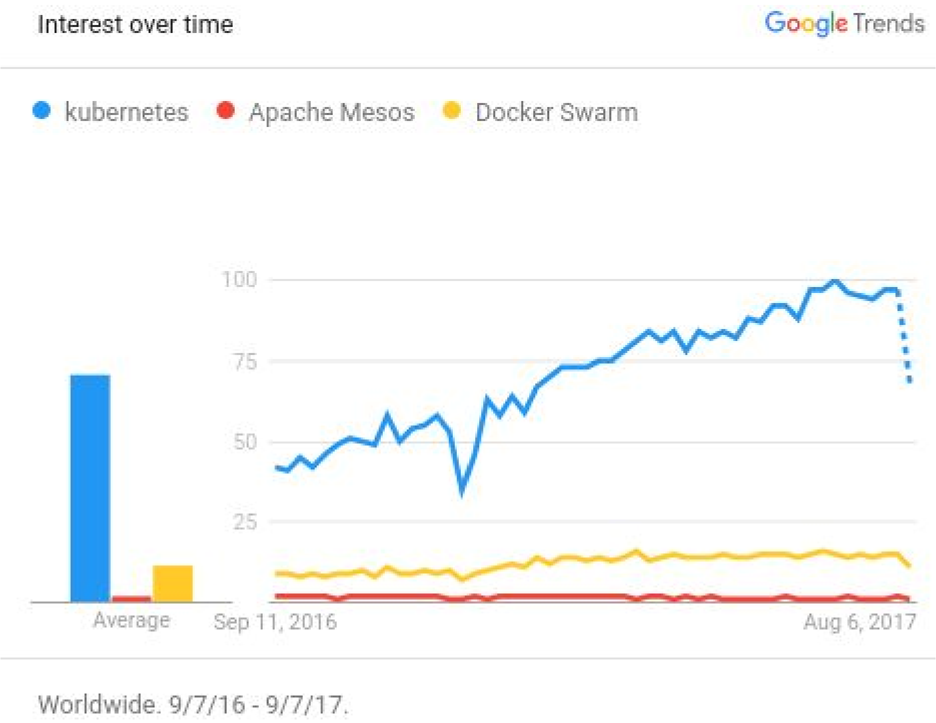
1 - k8s project



General	Kubernetes	docker swarm	Apache Mesos
Project Activity	Very High Activity	Moderate Activity	Very High Activity
Open Hub Data Quality	Updated about 16 hours ago	Updated 1 day ago	Updated about 18 hours ago
Homepage	kubernetes.io	github.com	mesos.apache.org
Project License	Apache-2.0	Apache-2.0	Apache-2.0
Estimated Cost	$19,848,427	$3,936,674	$9,944,713
All Time Statistics			
Contributors (All Time) View as graph	1633 developers	207 developers	298 developers
Commits (All Time) View as graph	54469 commits	3485 commits	26406 commits
Initial Commit	over 3 years ago	almost 3 years ago	over 6 years ago
Most Recent Commit	about 21 hours ago	1 day ago	1 day ago
12 Month Statistics			
Contributors (Past 12 Months)	806 developers	30 developers	113 developers
Commits (Past 12 Months)	19,165 commits	276 commits	6,998 commits
Files Modified	18,858 files	1,302 files	2,576 files
Lines Added	8,703,184 lines	377,858 lines	364,178 lines
Lines Removed	7,742,380 lines	332,198 lines	157,560 lines
Year-Over-Year Commits	Stable	Decreasing	Stable

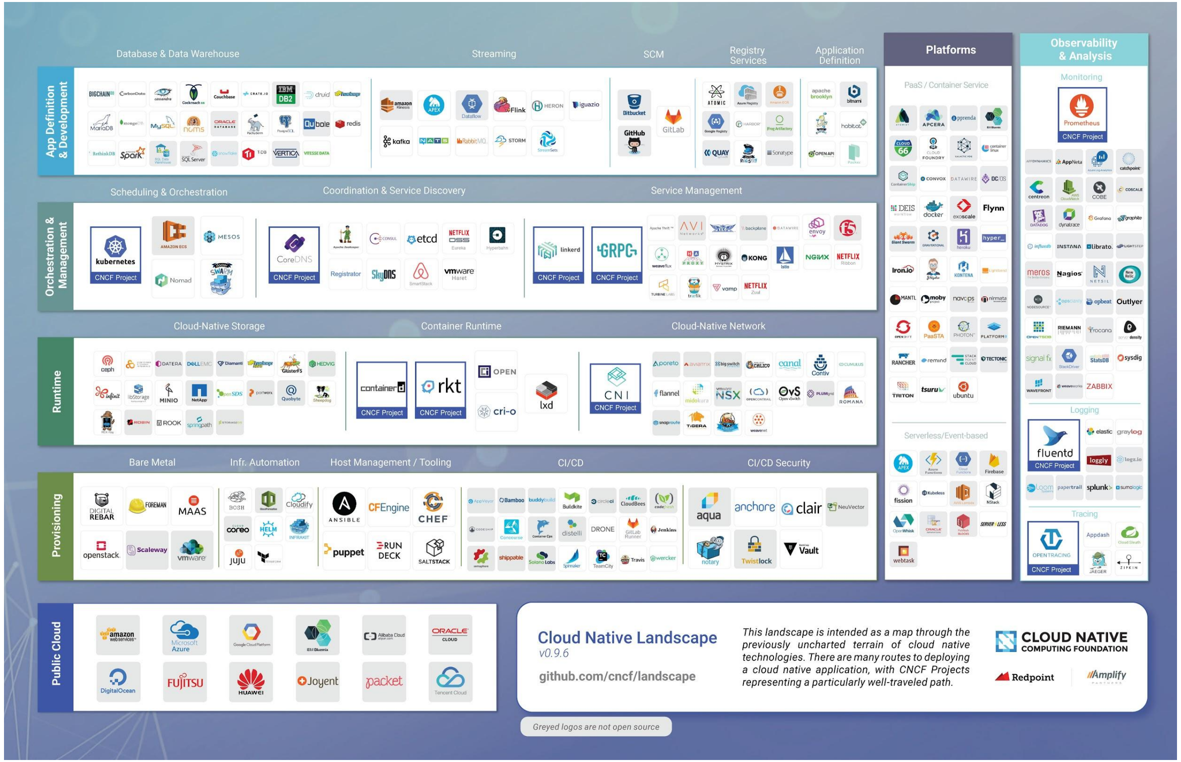
https://www.openhub.net/p/_compare?project_0=Kubernetes&project_1=docker+swarm&project_2=Apache+Mesos

1 - k8s project



https://trends.google.com/trends/explore?q=kubernetes,Apache%20Mesos,Docker%20Swarm

1 - k8s project



App Definition & Development

Database & Data Warehouse

Streaming

SCM

Registry Services

Application Definition

Orchestration & Management

Scheduling & Orchestration

Coordination & Service Discovery

Service Management

Runtime

Cloud-Native Storage

Container Runtime

Cloud-Native Network

Provisioning

Bare Metal

Infr. Automation

Host Management / Tooling

CI/CD

CI/CD Security

Public Cloud

Platforms

PaaS / Container Service

Serverless/Event-based

Observability & Analysis

Monitoring

Logging

Tracing

Cloud Native Landscape
v0.9.6

github.com/cncf/landscape

This landscape is intended as a map through the previously uncharted terrain of cloud native technologies. There are many routes to deploying a cloud native application, with CNCF Projects representing a particularly well-traveled path.

CLOUD NATIVE COMPUTING FOUNDATION

Redpoint

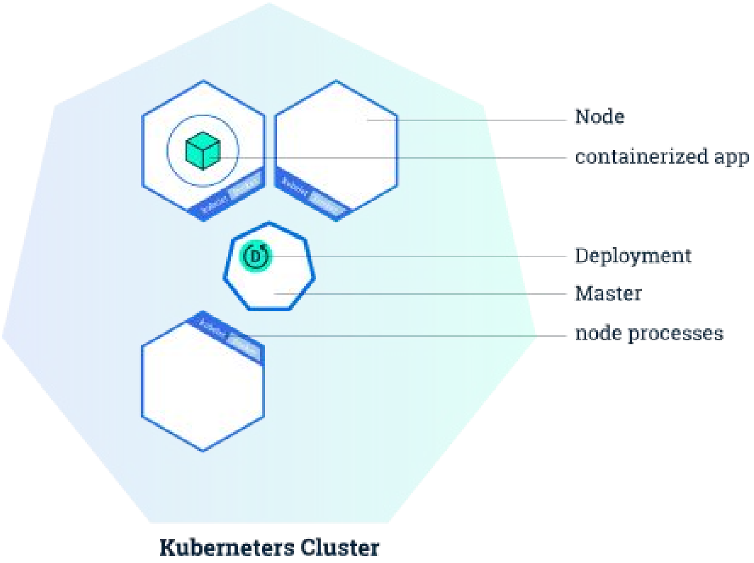
Amplify PARTNERS

Greyed logos are not open source

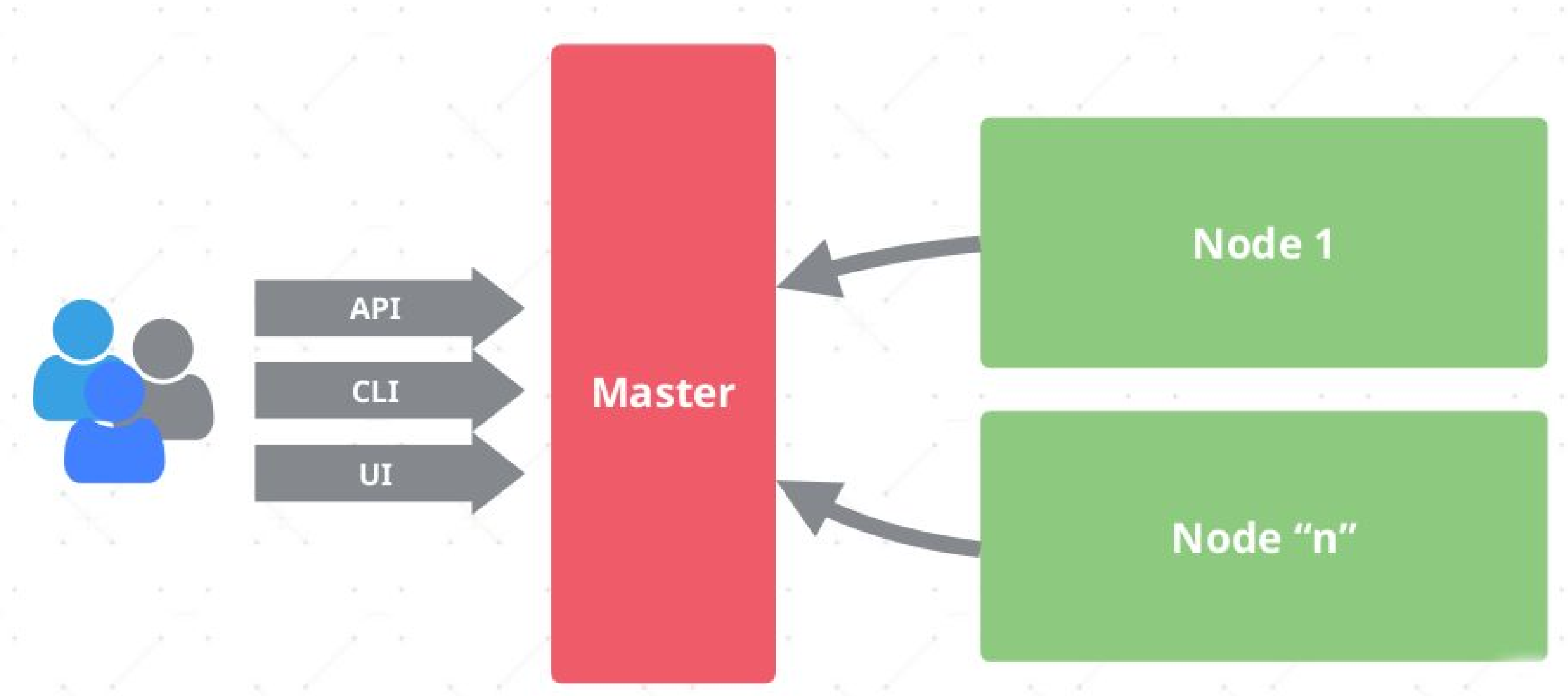
https://github.com/cncf/landscape

leboncoin

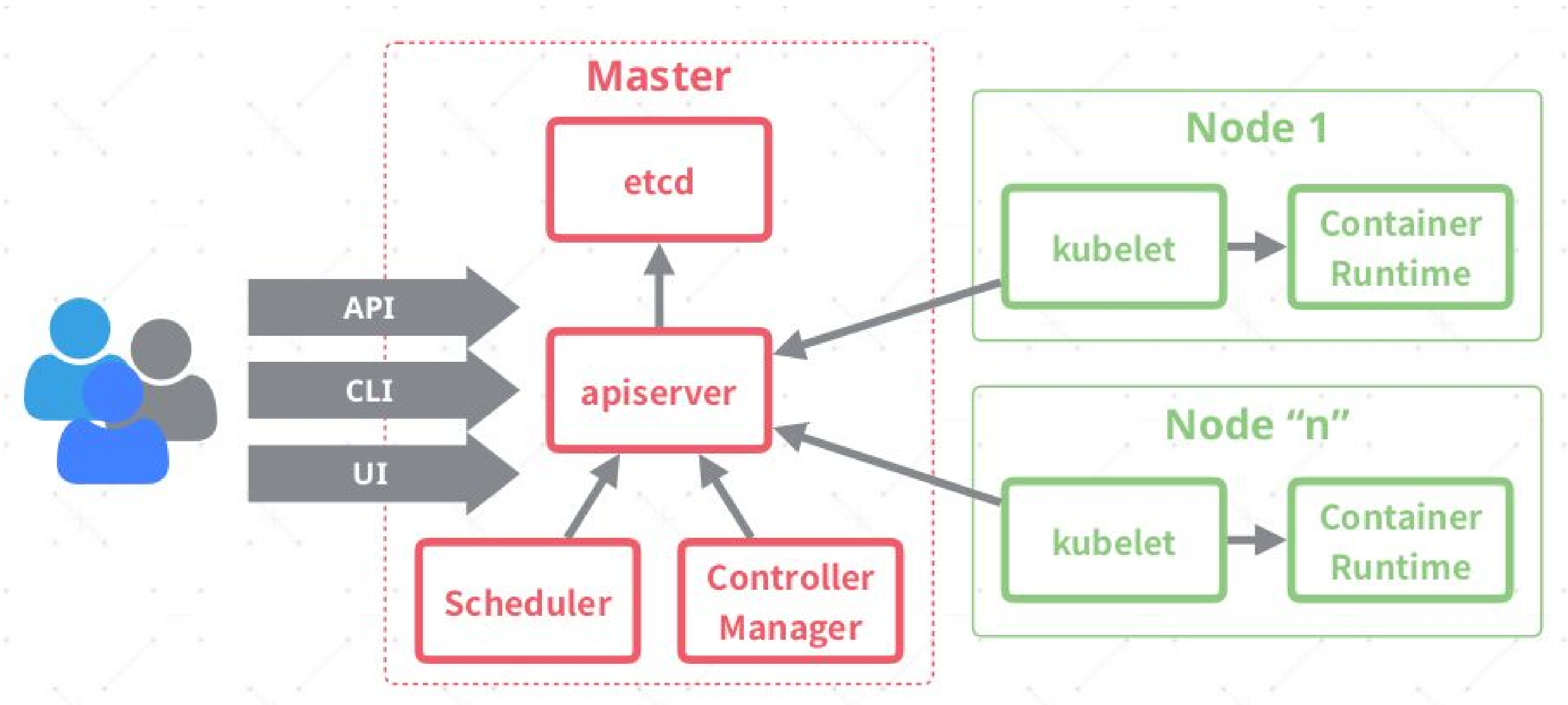
2 - k8s Cluster



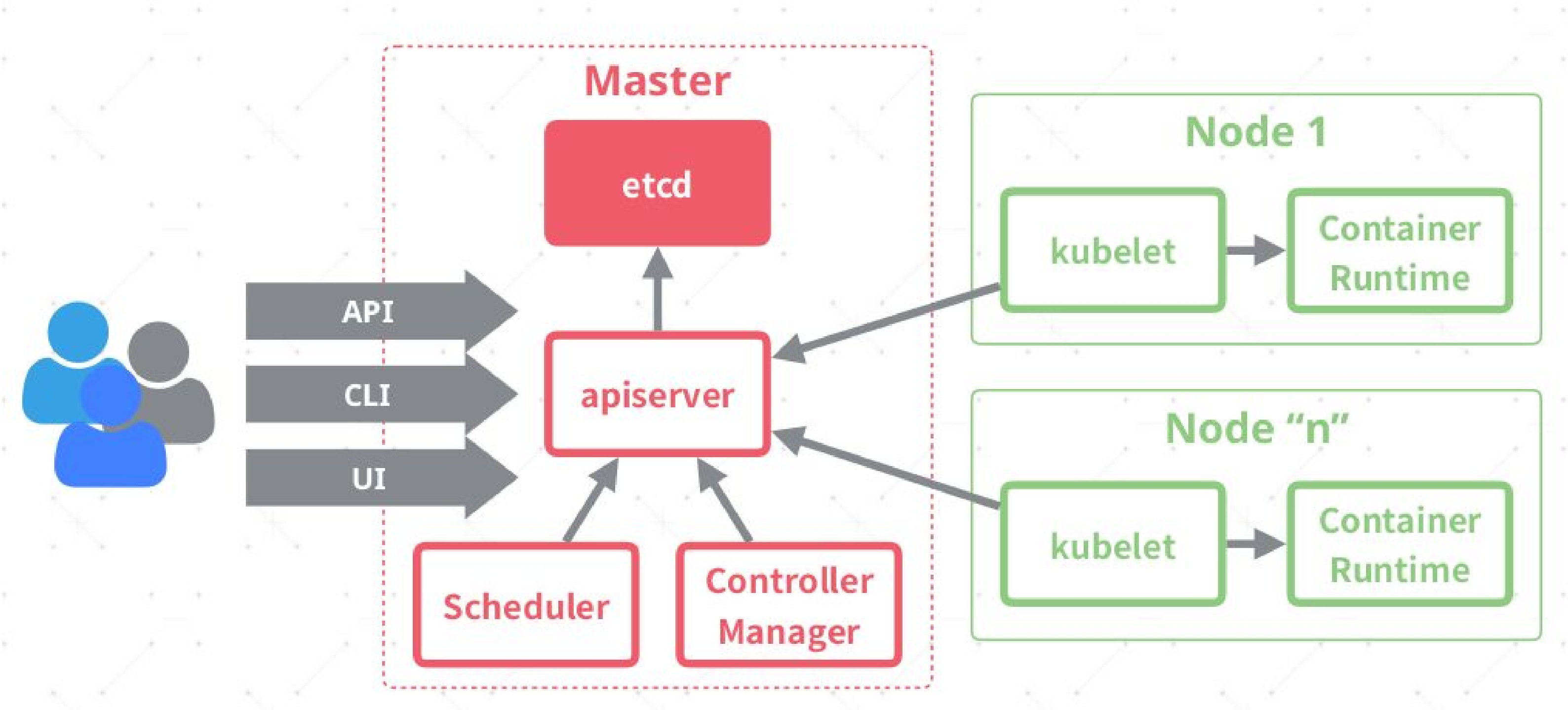
2 - k8s Cluster



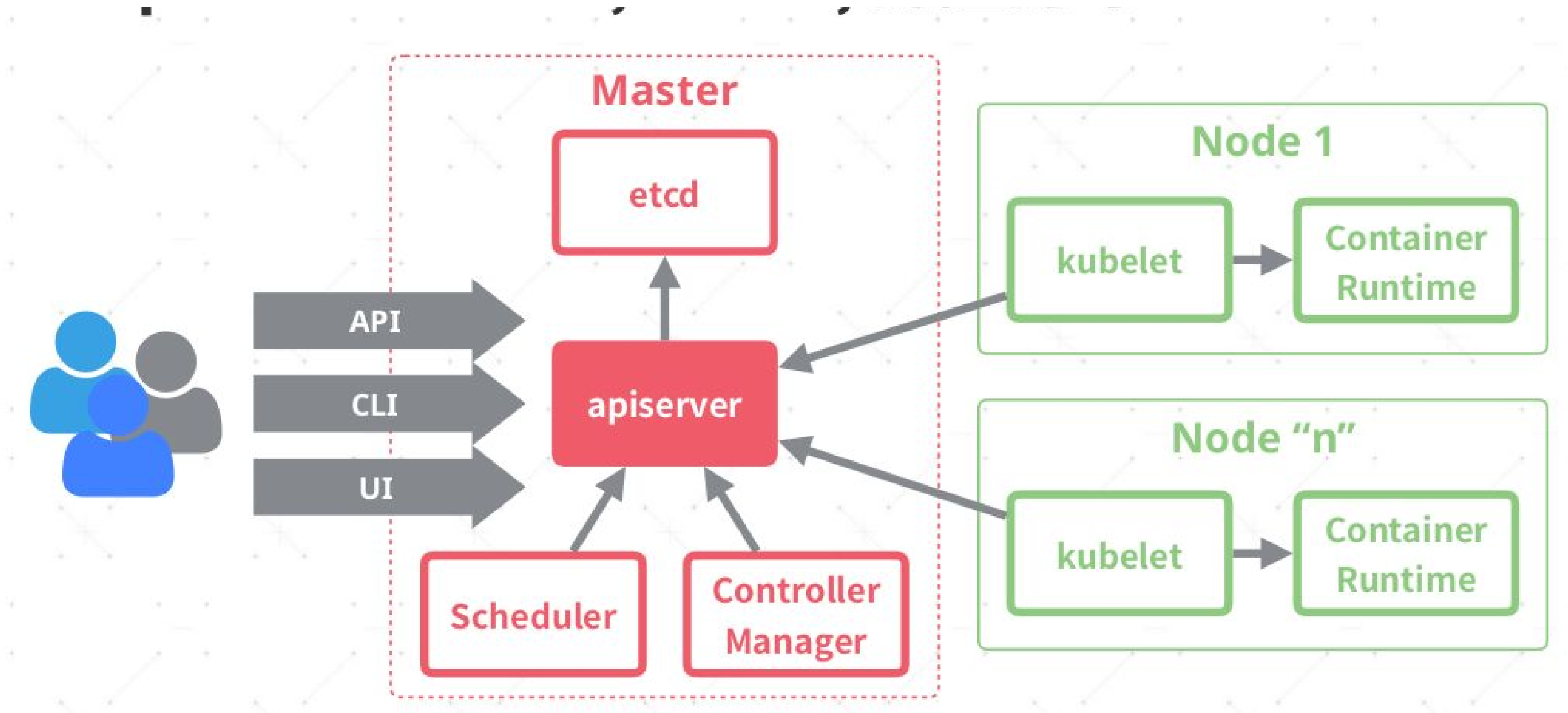
2 - k8s Cluster



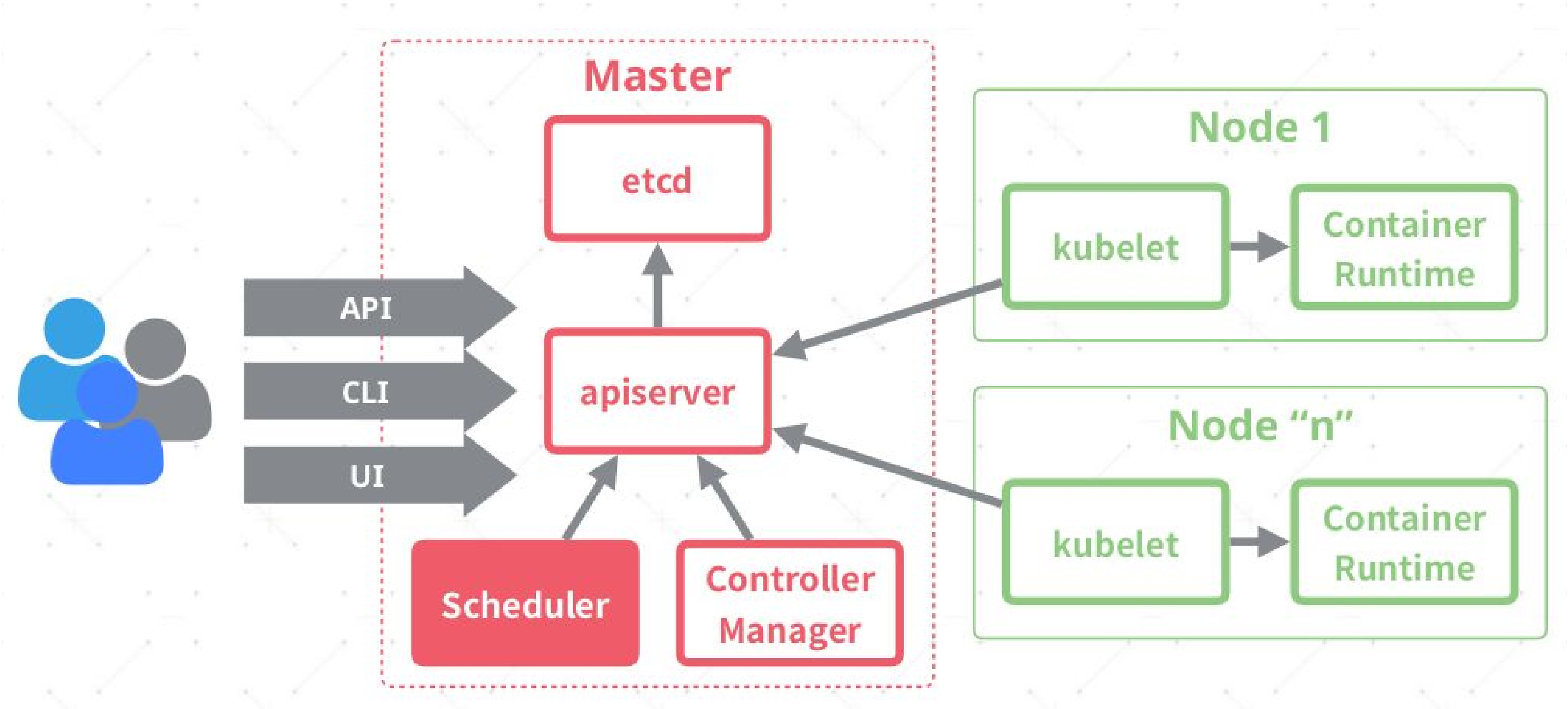
2 - k8s Cluster



2 - k8s Cluster

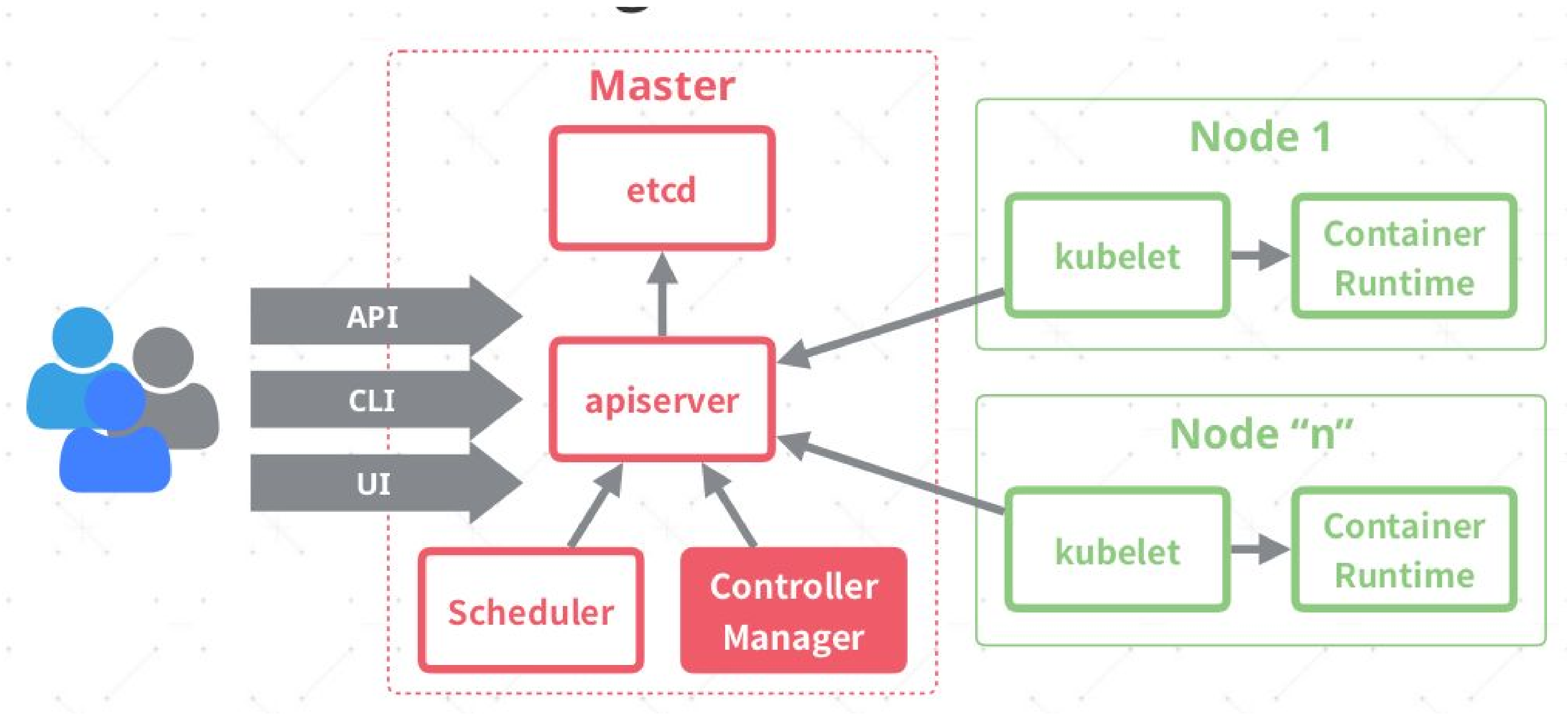


2 - k8s Cluster

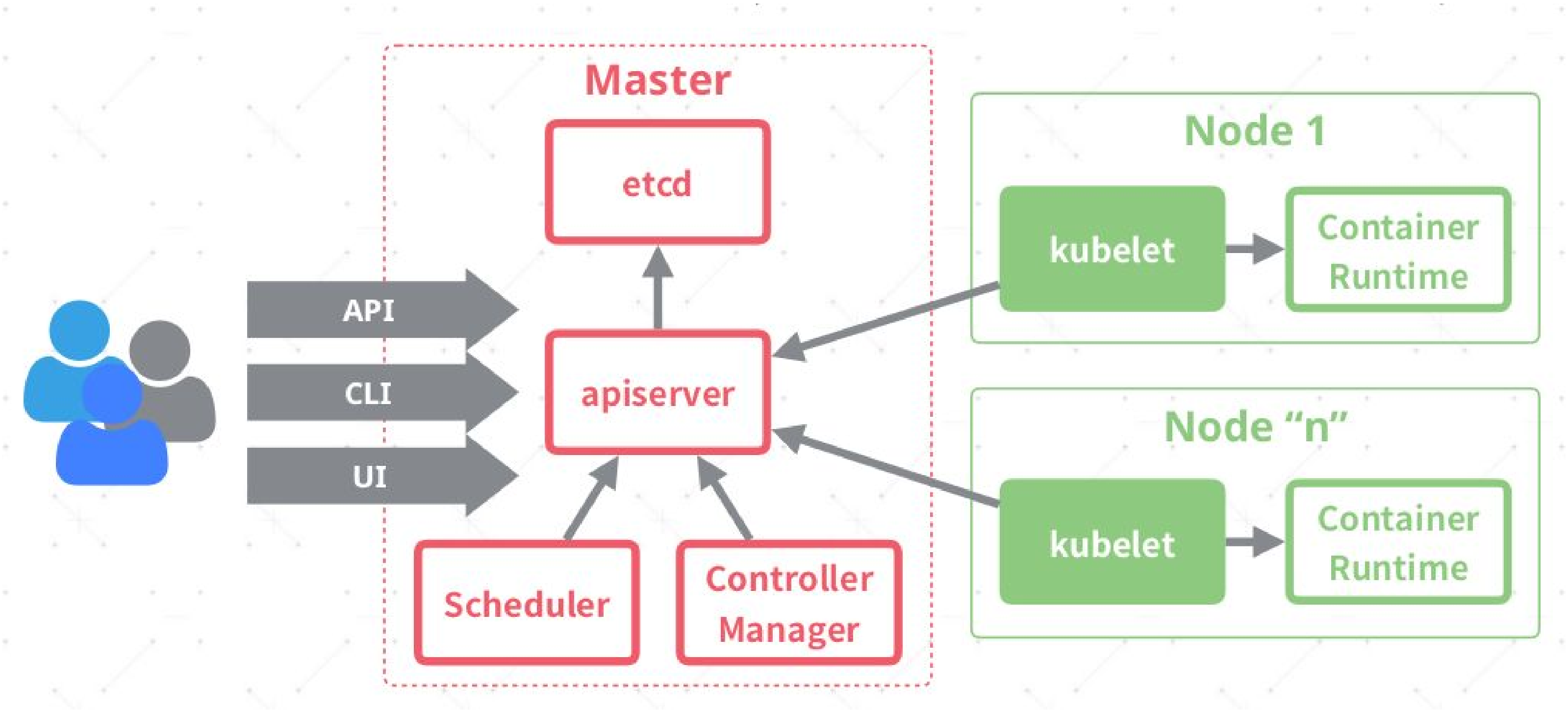


leboncoin

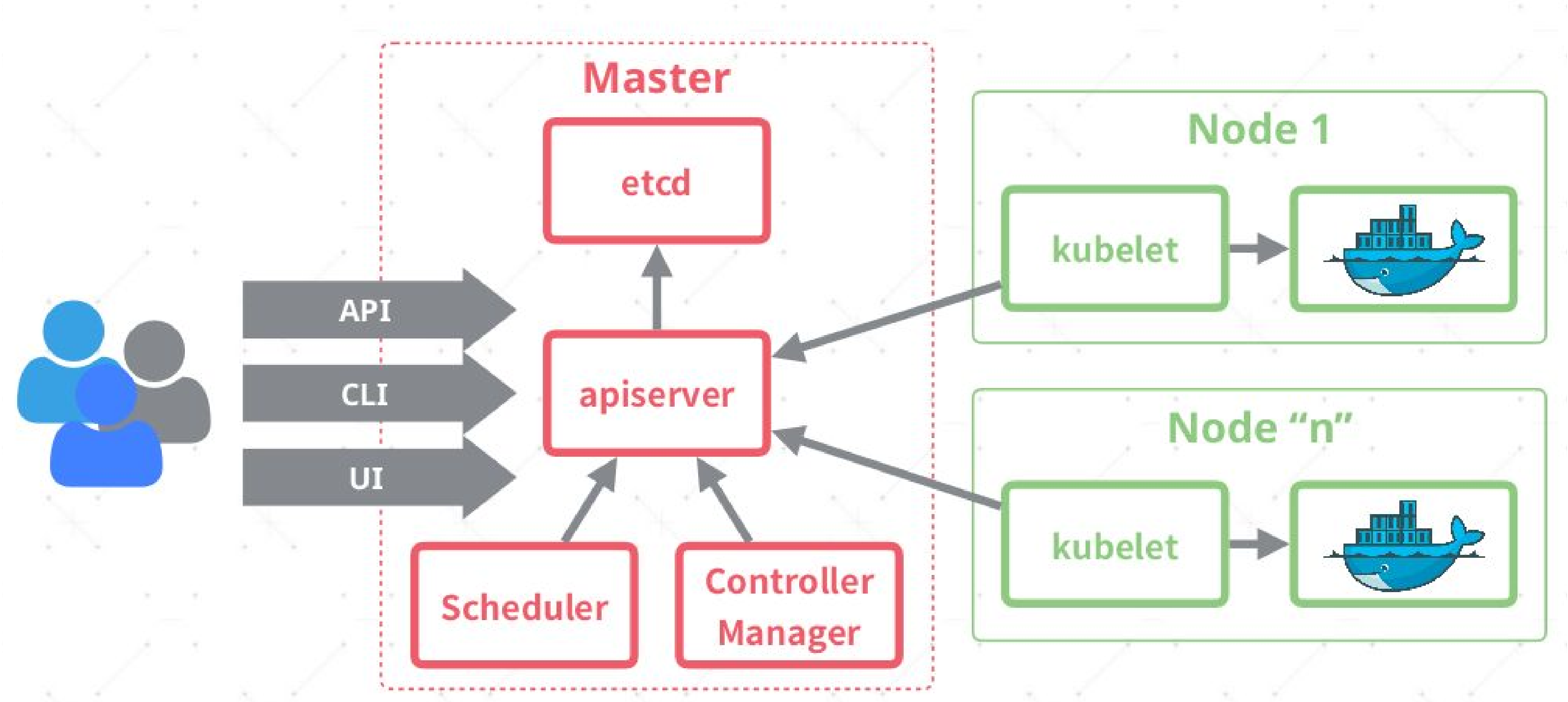
2 - k8s Cluster



2 - k8s Cluster



2 - k8s Cluster



3 - Resources

1 - Pods

2 - *Sets

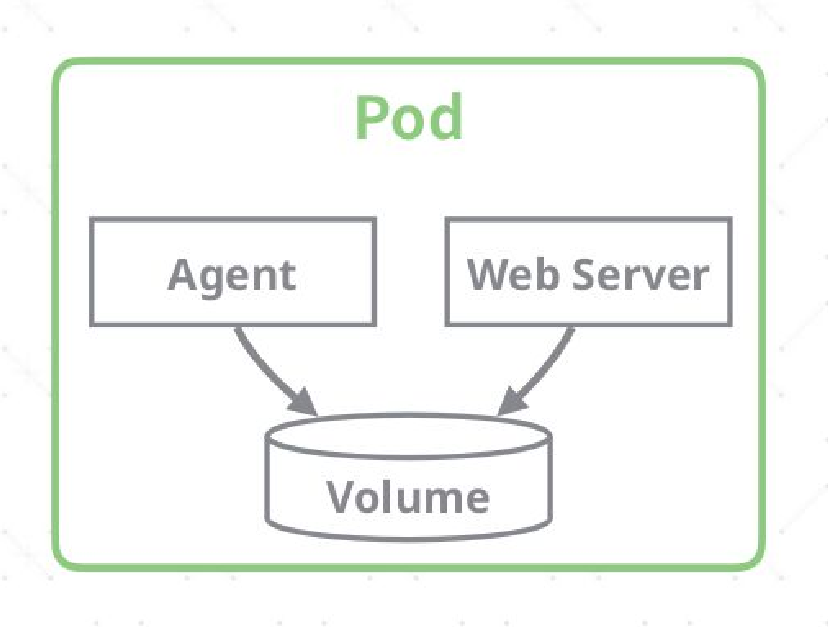
3 - Deployment

4 - Services

3.1 - Pods

{Pod} = Group of containers

- Smallest “**Unit**” in k8s
- Logical group of **multiple** containers (1 or N)

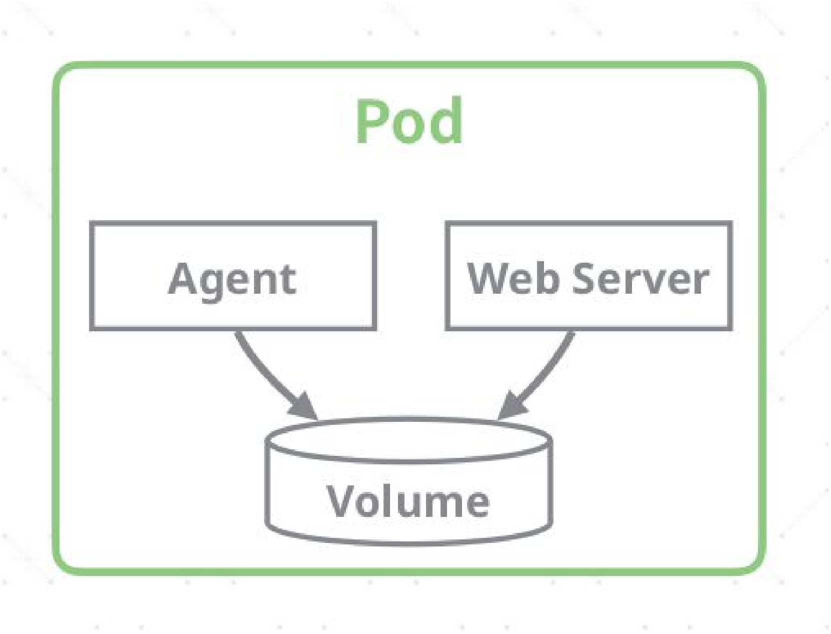


https://kubernetes.io/docs/concepts/workloads/pods/pod/

3.1 - Pods

{Pod} = Group of containers

- They **share**
  - Network namespace
  - Filesystem namespace
  - IPC
- They are **co-scheduled** on the same node

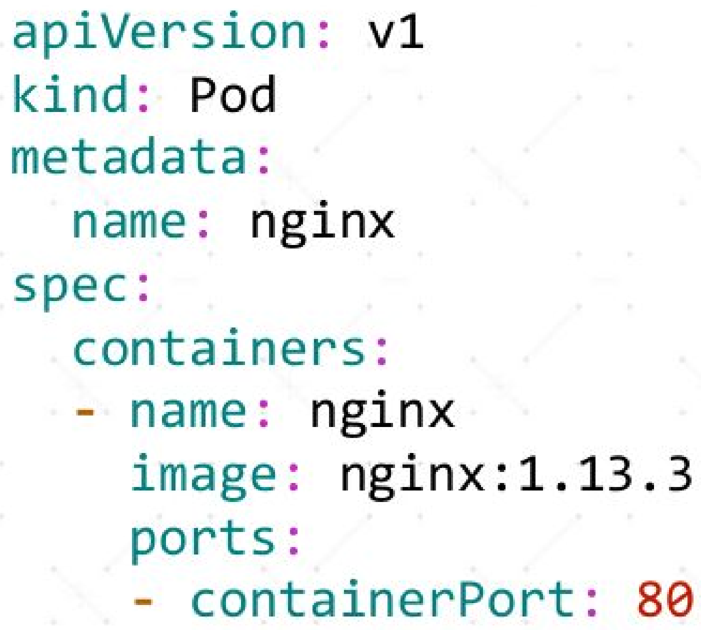


https://kubernetes.io/docs/concepts/workloads/pods/pod/

3.1 - Pods

{Pod} = Group of containers

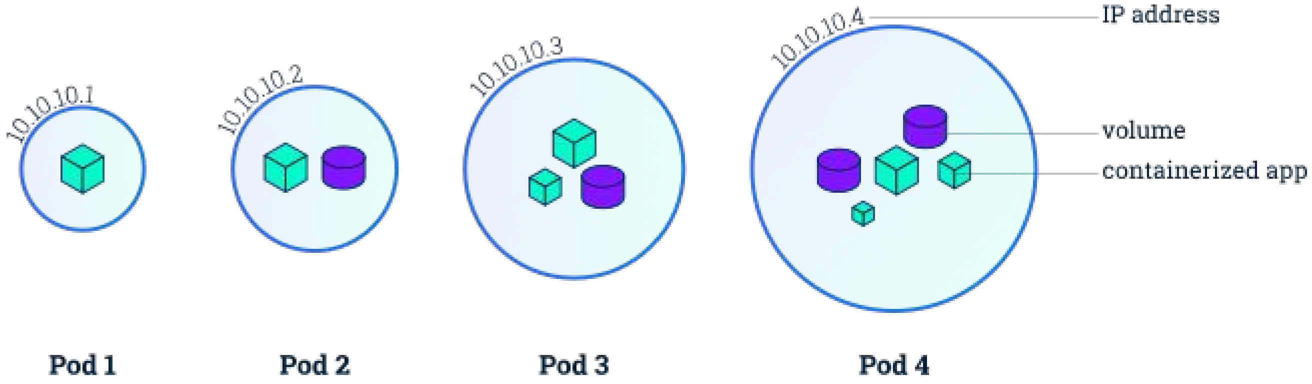
- They **share**
  - Network namespace
  - Filesystem namespace
  - IPC
- They are **co-scheduled** on the same node



```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.13.3
    ports:
    - containerPort: 80
```

https://kubernetes.io/docs/concepts/workloads/pods/pod/

3.1 - Pods



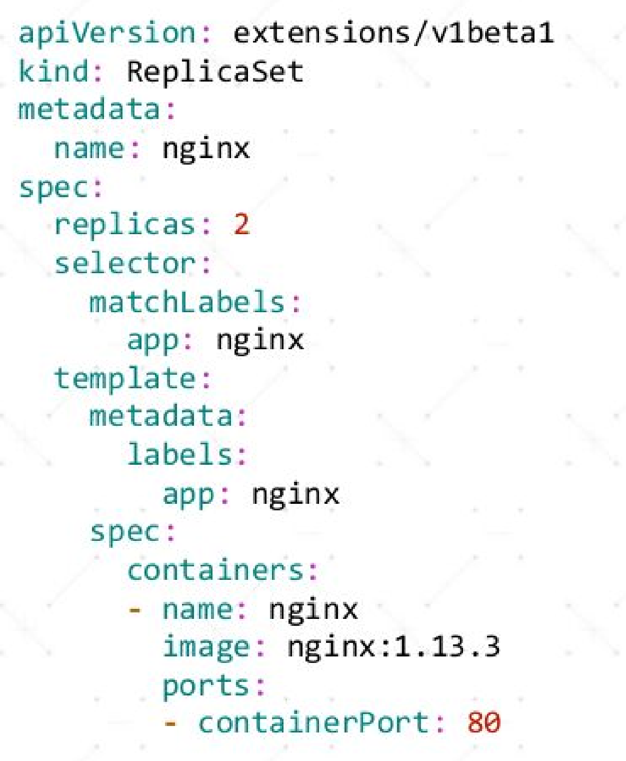
https://kubernetesbootcamp.github.io/kubernetes-bootcamp/

3.2. - *Sets

ReplicatSet

“*ReplicaSet ensures that a specified **number of pod replicas** are running at any given time*“

https://kubernetes.io/docs/concepts/workloads/controllers/replicaset/



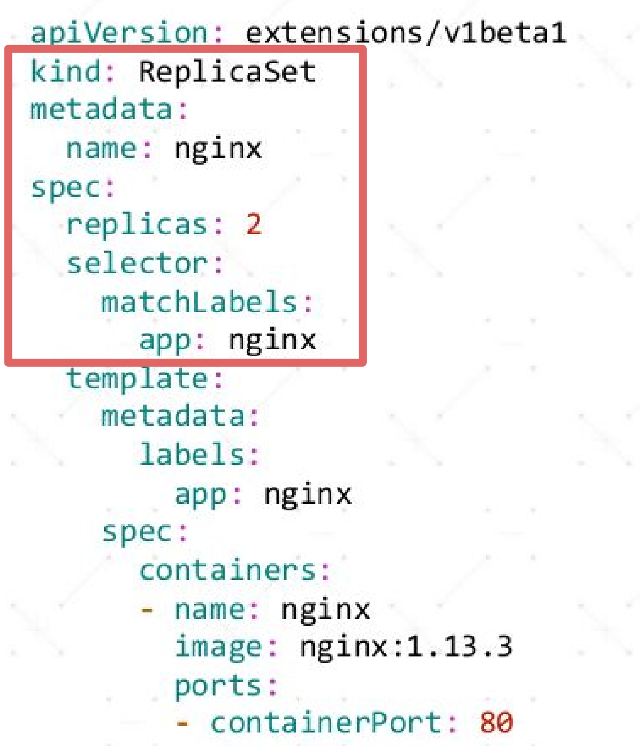
```
apiVersion: extensions/v1beta1
kind: ReplicaSet
metadata:
  name: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.13.3
        ports:
        - containerPort: 80
```

3.2. - *Sets

ReplicatSet

“*ReplicaSet ensures that a specified **number of pod replicas** are running at any given time*“

https://kubernetes.io/docs/concepts/workloads/controllers/replicaset/



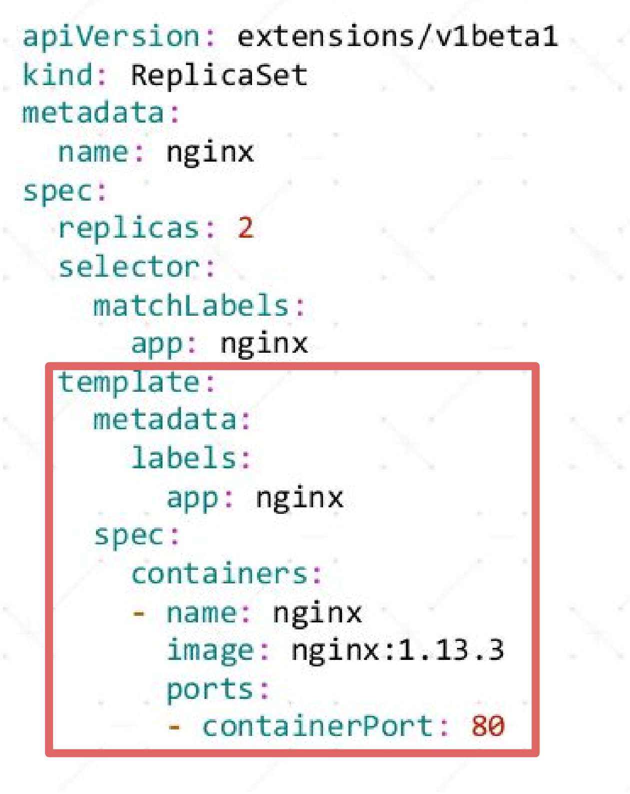
```
apiVersion: extensions/v1beta1
kind: ReplicaSet
metadata:
  name: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.13.3
        ports:
        - containerPort: 80
```

3.2. - *Sets

ReplicatSet

> “*ReplicaSet ensures that a specified* ***number of pod replicas*** *are running at any given time*“

https://kubernetes.io/docs/concepts/workloads/controllers/replicaset/

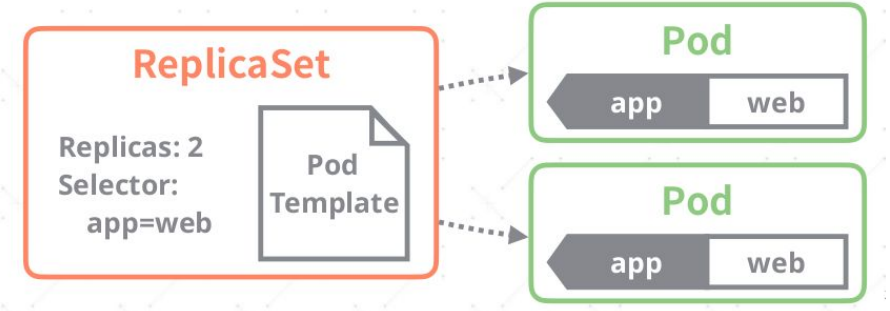


```yaml
apiVersion: extensions/v1beta1
kind: ReplicaSet
metadata:
  name: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.13.3
        ports:
        - containerPort: 80
```

3.2. - *Sets

ReplicatSet

“*ReplicaSet ensures that a specified **number of pod replicas** are running at any given time*“



https://kubernetes.io/docs/concepts/workloads/controllers/replicaset/

3.2. - *Sets

StatefulSets

- Stable, unique **network identifiers**.
- Stable, **persistent storage**.
- **Ordered**, graceful deployment and scaling.
- Ordered, graceful deletion and termination.
- Ordered, automated rolling updates.

https://kubernetes.io/docs/concepts/workloads/controllers/statefulset/

DaemonSets

- Run 1 pod on all nodes (or only some via *annotations*)
- Auto schedule a pod when a node is added to the cluster

Examples:

- *Log collection daemons,*
- *storage daemons,*
- *monitoring daemons, ...*

https://kubernetes.io/docs/concepts/workloads/controllers/daemonset/

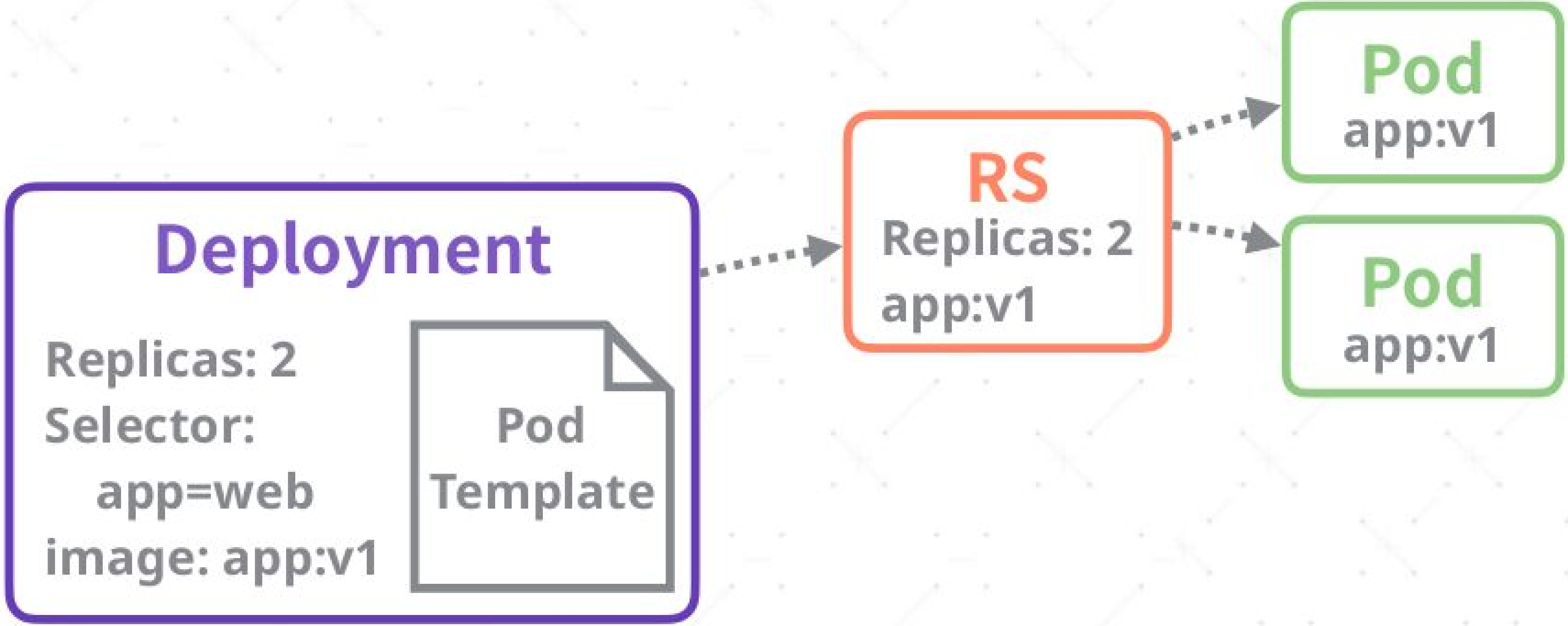
3.3 - Deployment

Replicas Controller “v2” with controls

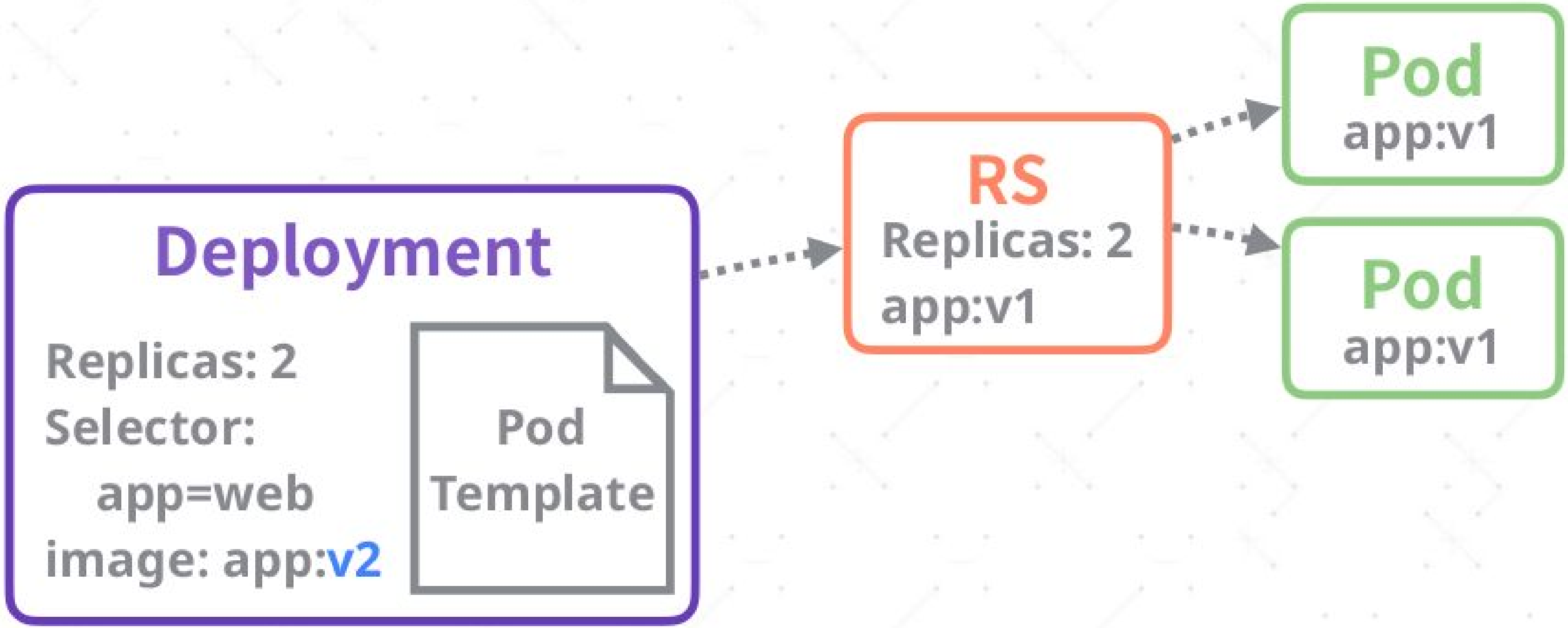
- Bring up a **Replica Set** and **Pods**.
- Check the **status** of a Deployment.
- **Update** that Deployment (e.g. new image, labels).
- **Rollback** to an earlier Deployment revision.
- **Pause** and **resume** a Deployment.

https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

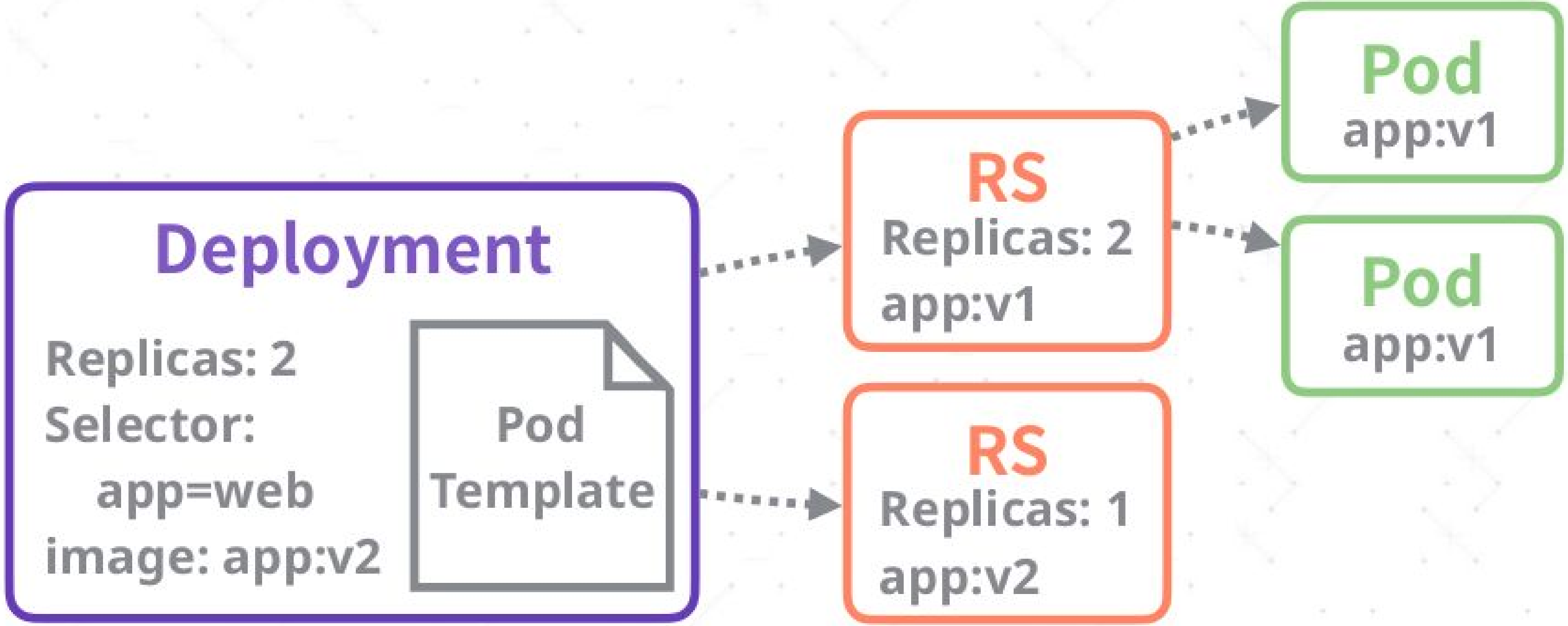
3.3 - Deployment



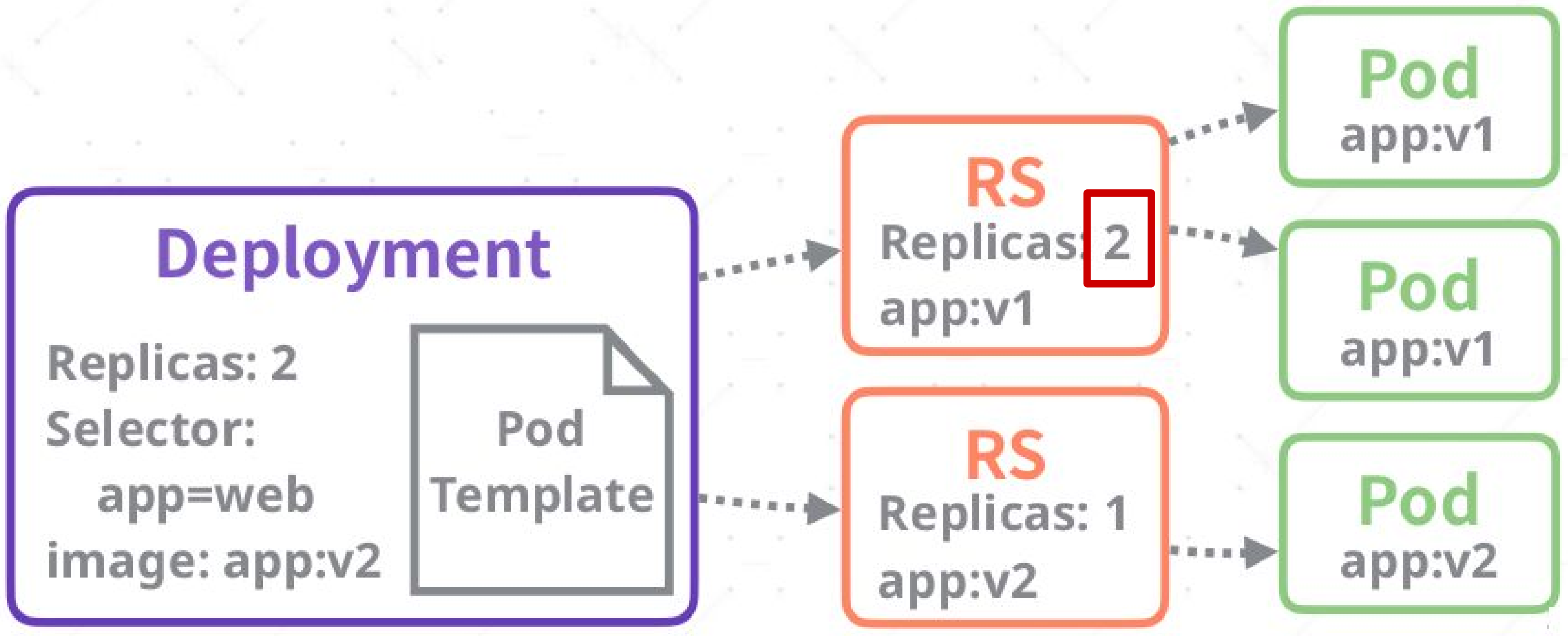
3.3 - Deployment



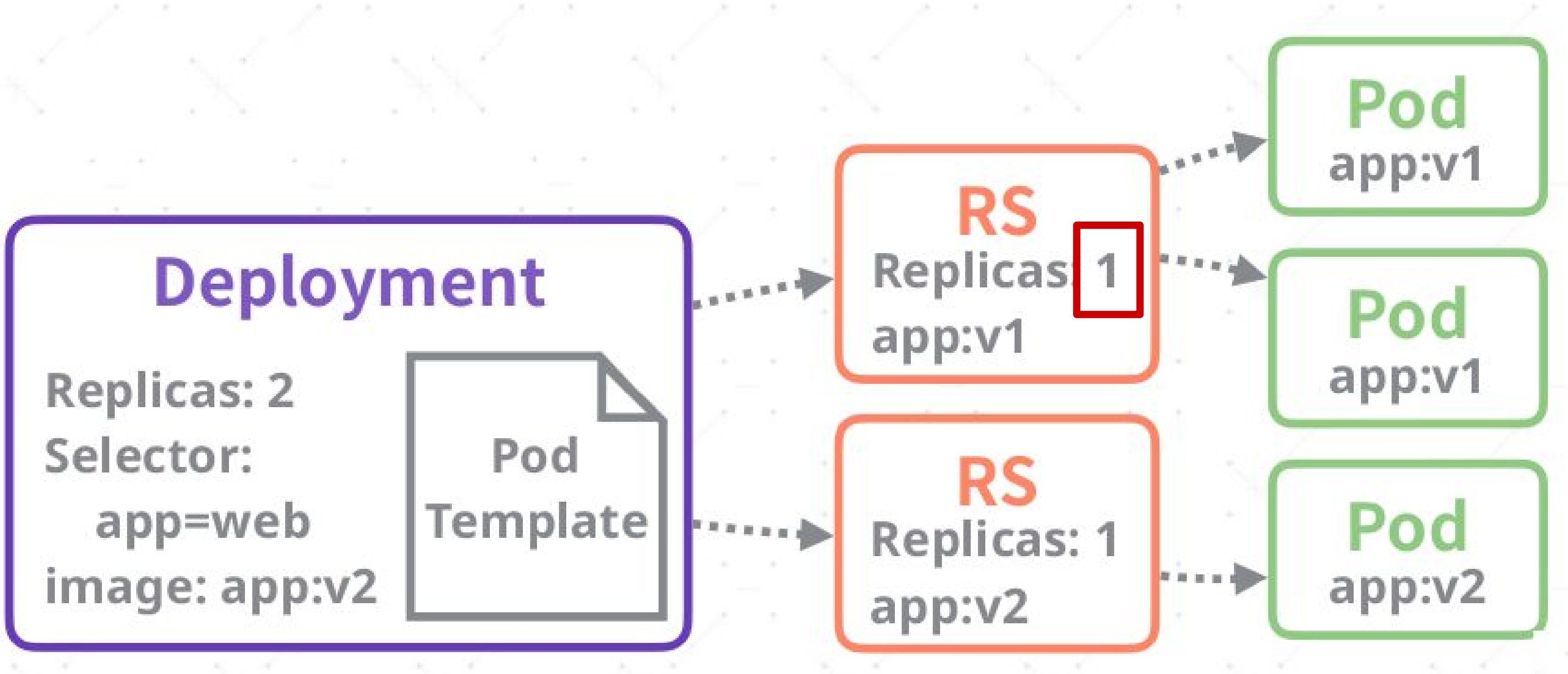
3.3 - Deployment



3.3 - Deployment



3.3 - Deployment



Deployment

Replicas: 2
Selector:
app=web
image: app:v2

Pod
Template

RS
Replicas: 1
app:v1

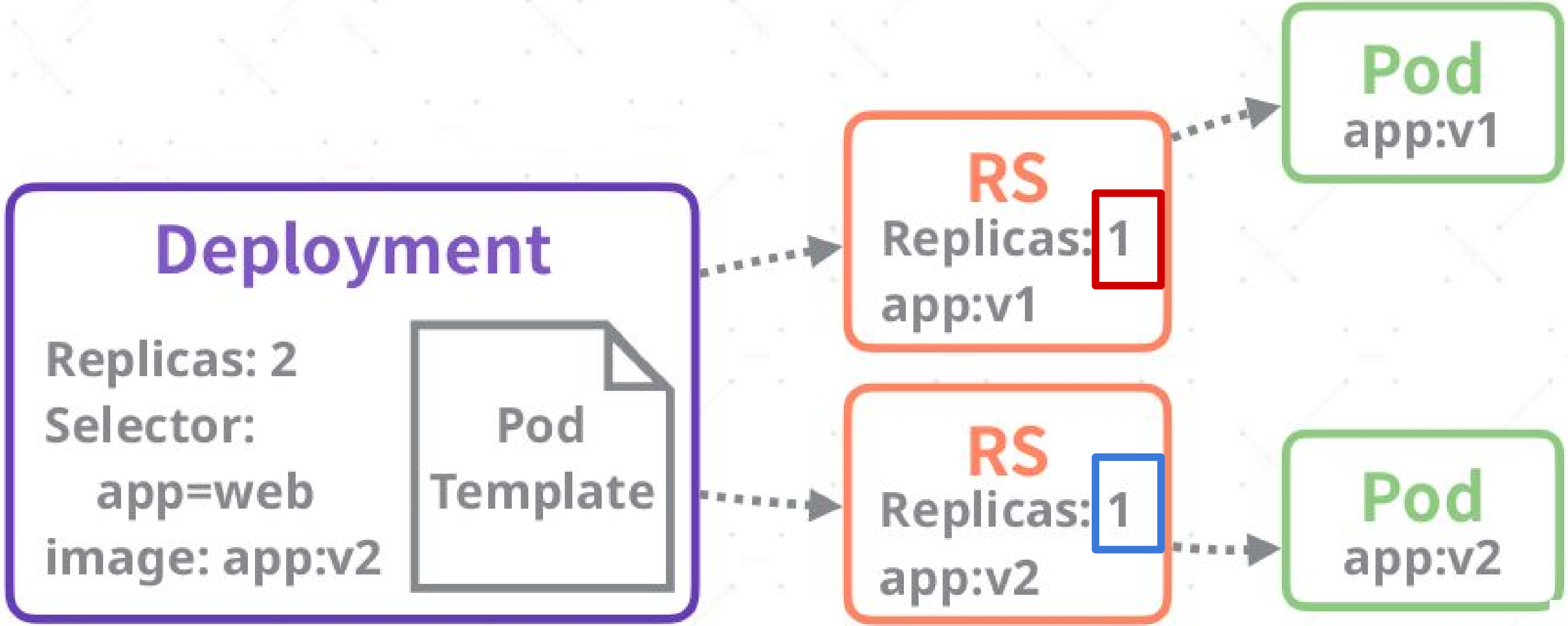
RS
Replicas: 1
app:v2

Pod
app:v1

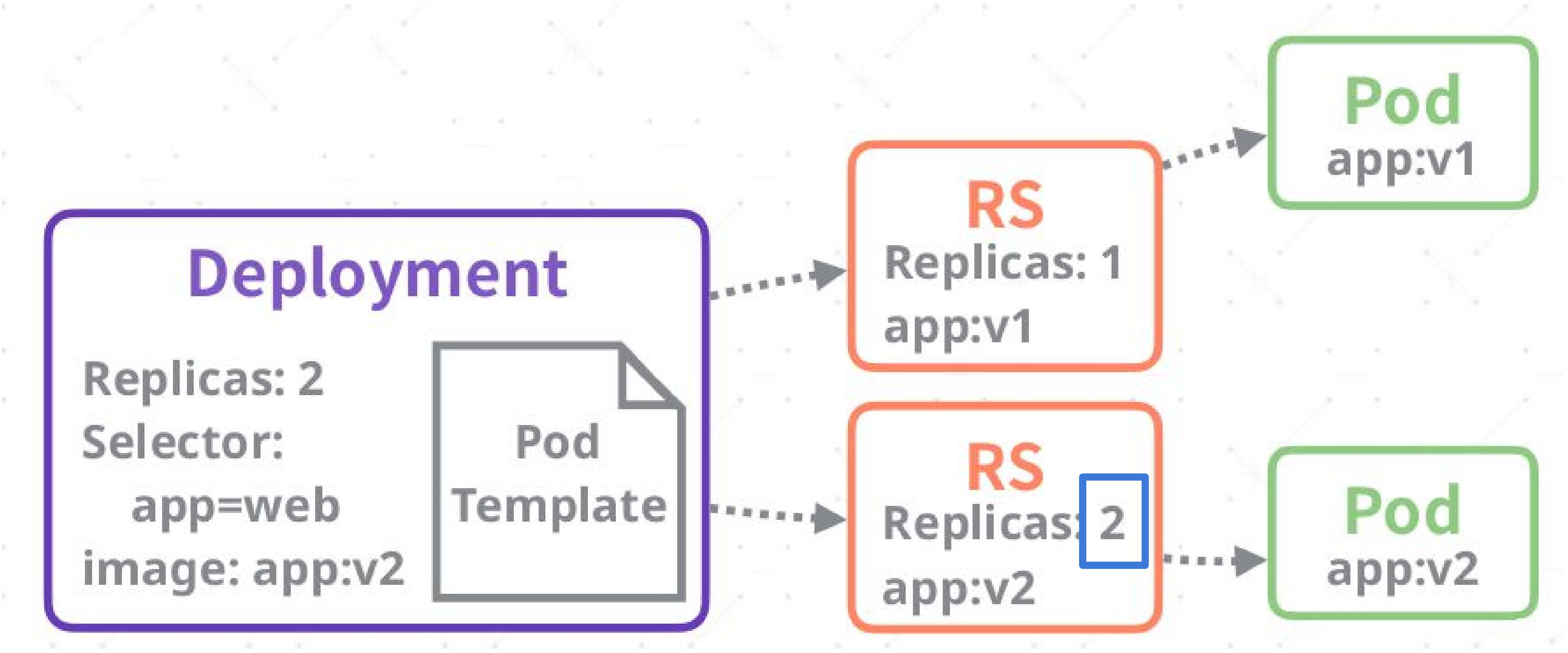
Pod
app:v1

Pod
app:v2

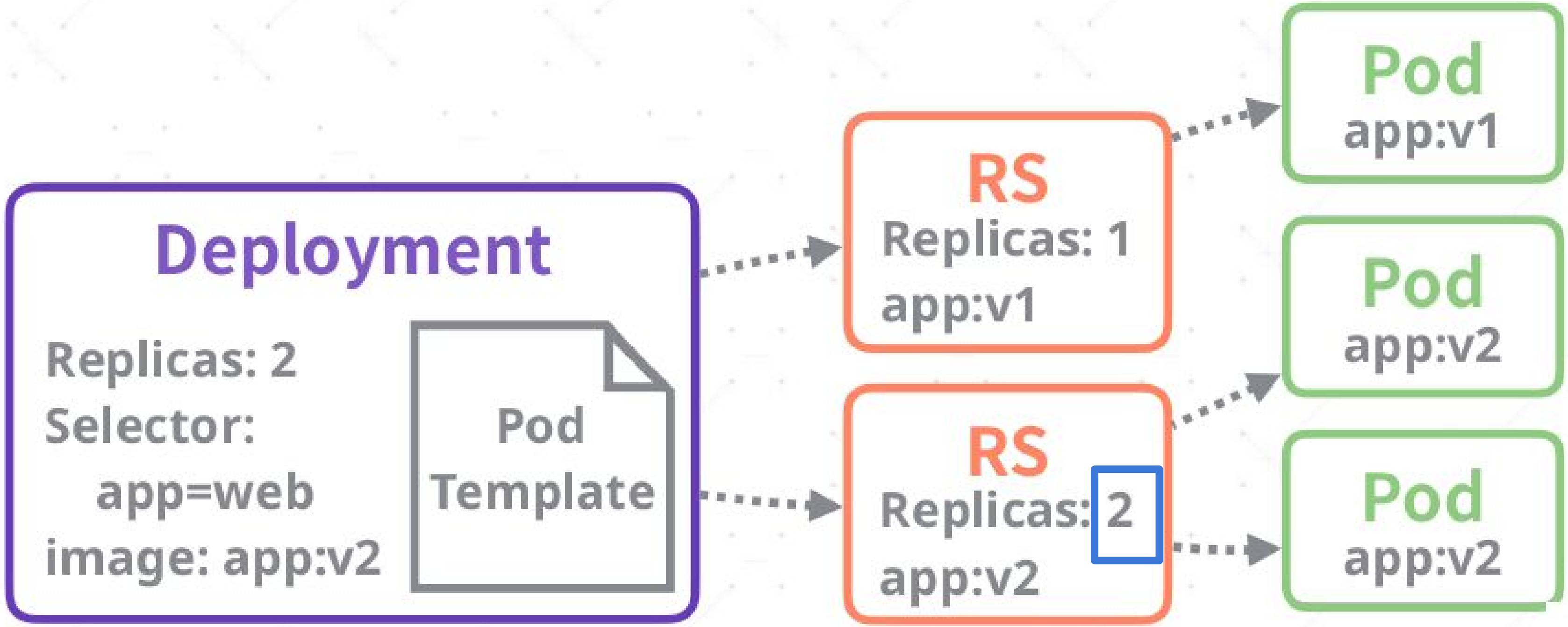
3.3 - Deployment



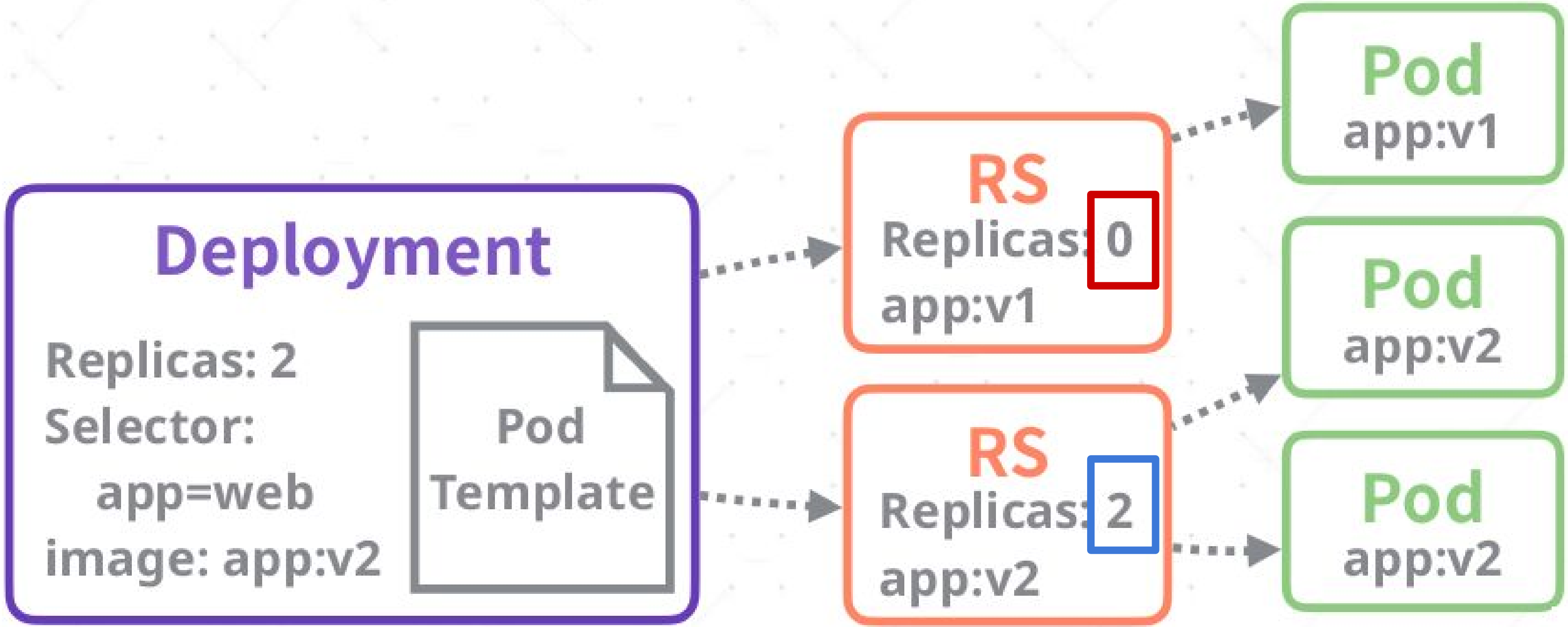
3.3 - Deployment



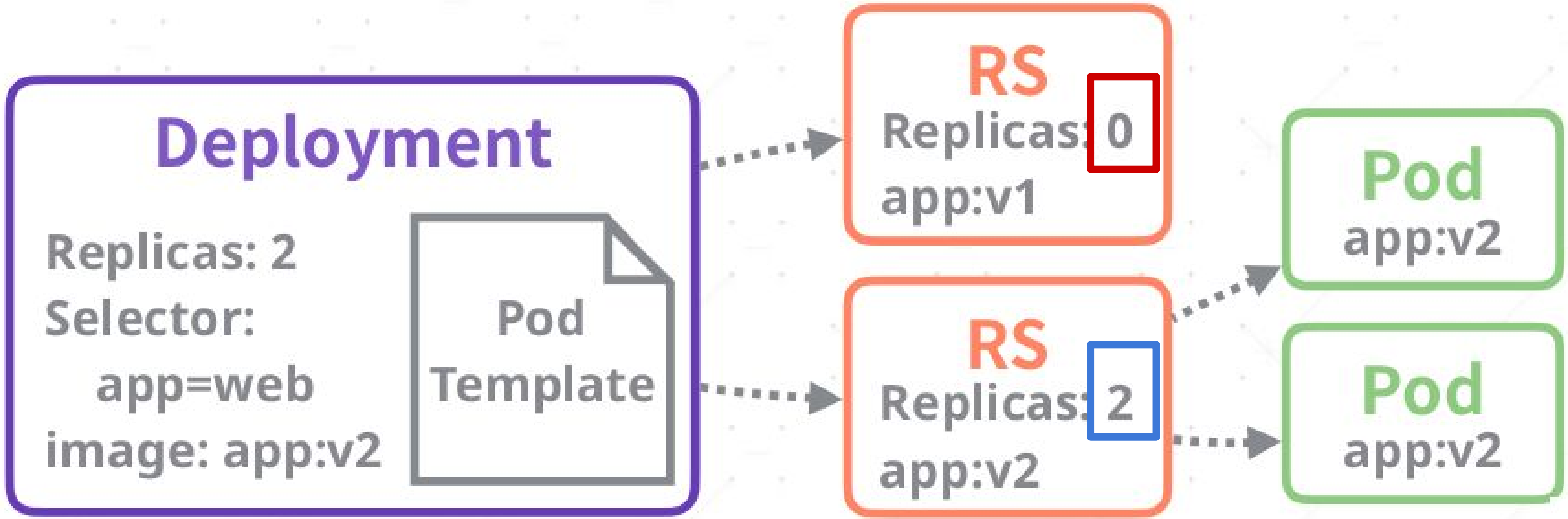
3.3 - Deployment



3.3 - Deployment

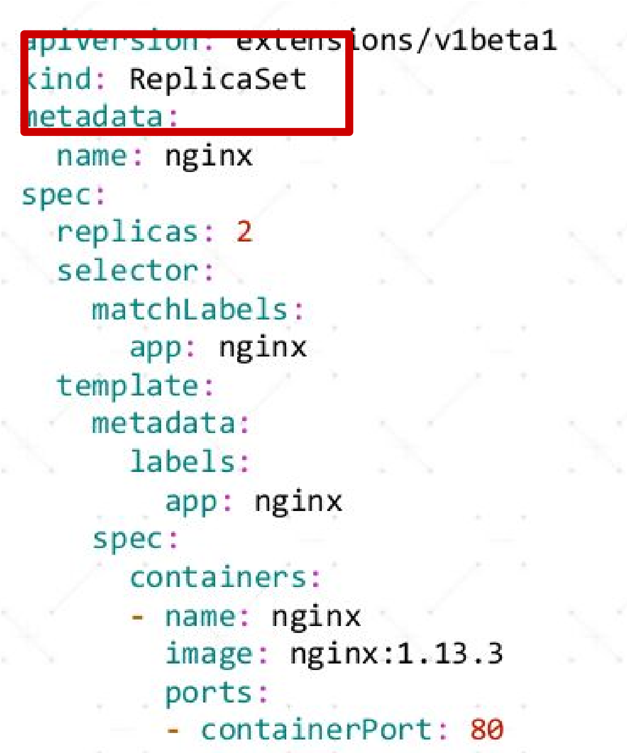


3.3 - Deployment

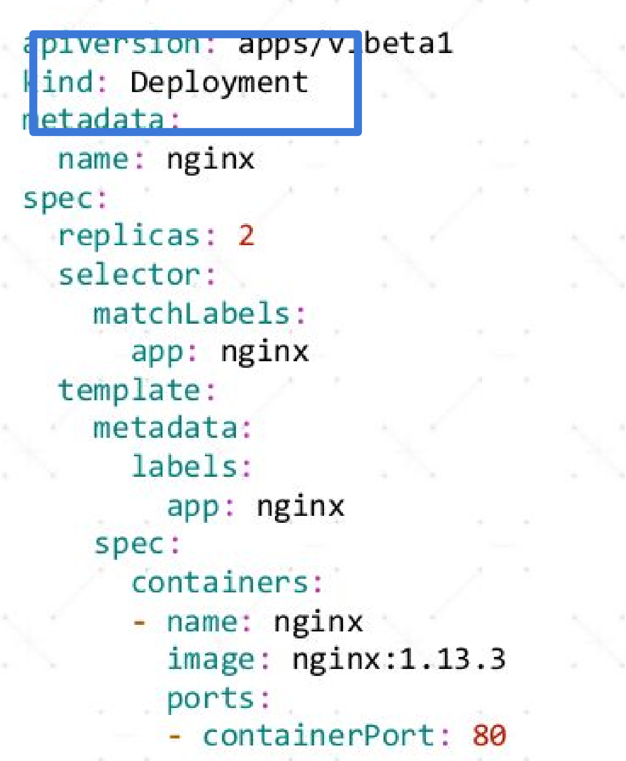


leboncoin

3.3 - Deployment



```
apiVersion: extensions/v1beta1
kind: ReplicaSet
metadata:
  name: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.13.3
        ports:
        - containerPort: 80
```



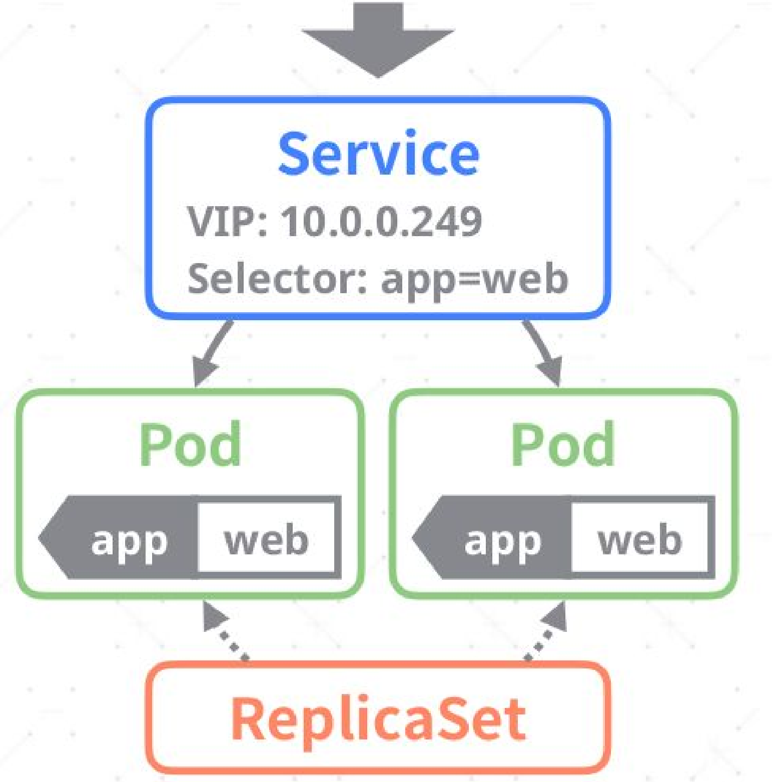
```
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.13.3
        ports:
        - containerPort: 80
```

3.4 - Service

Kind of service:

- ClusterIP
  - IP only reachable from within the cluster
- NodePort
  - A port is allocated and exposed on every nodes
- LoadBalancer
  - Implementing clouds ELBs management

https://kubernetes.io/docs/concepts/services-networking/service/

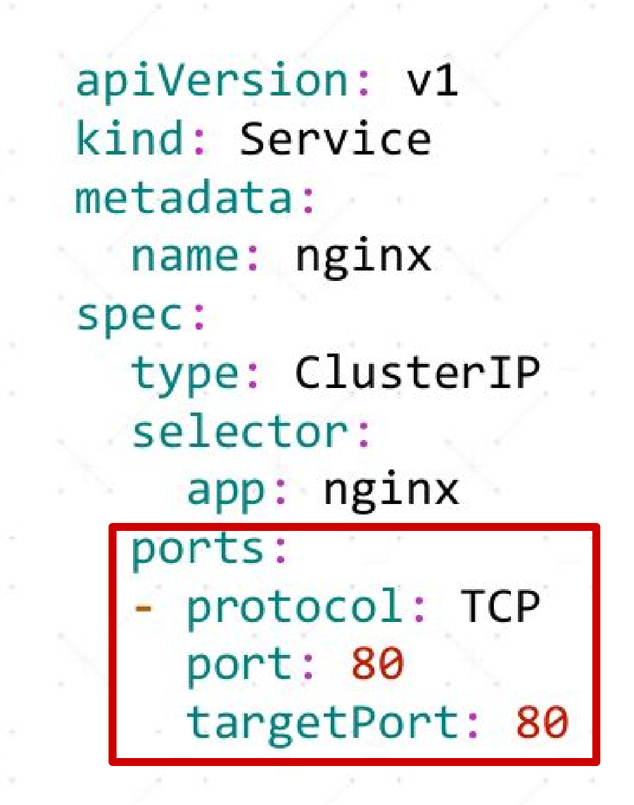


3.4 - Service

Kind of service:

- ClusterIP
  - IP only reachable from within the cluster
- NodePort
  - A port is allocated and exposed on every nodes
- LoadBalancer
  - Implementing **clouds ELBs** management only

https://kubernetes.io/docs/concepts/services-networking/service/



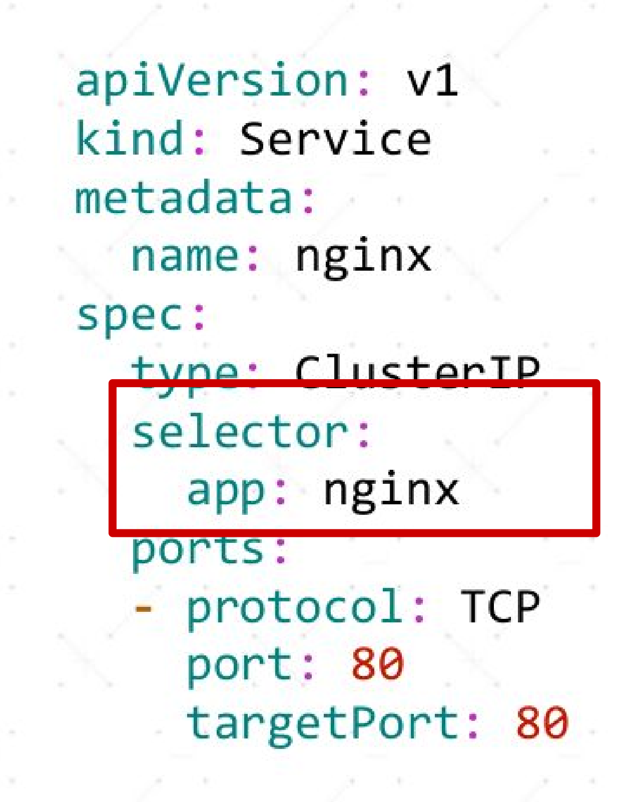
```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx
spec:
  type: ClusterIP
  selector:
    app: nginx
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
```

3.4 - Service

Kind of service:

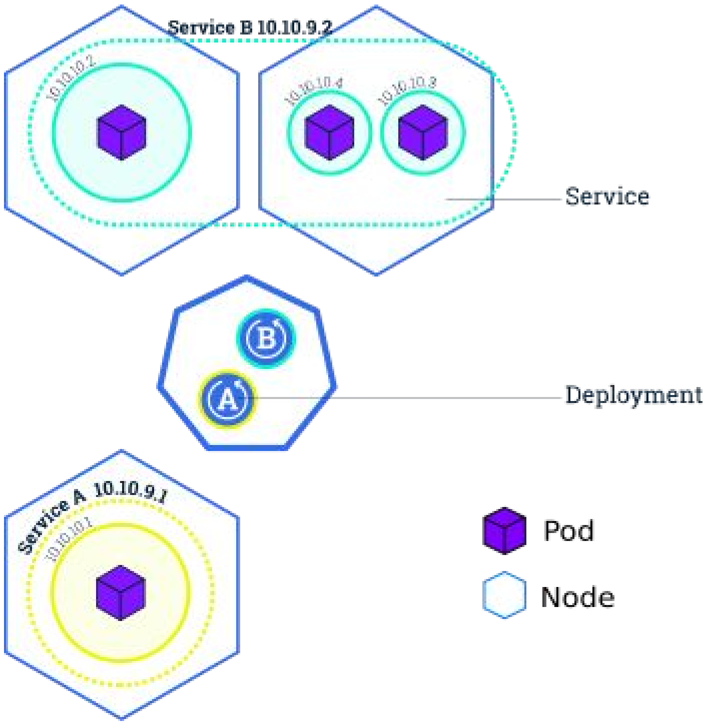
- ClusterIP
  - IP only reachable from within the cluster
- NodePort
  - A port is allocated and exposed on every nodes
- LoadBalancer
  - Implementing **clouds ELBs** management only

https://kubernetes.io/docs/concepts/services-networking/service/



```
apiVersion: v1
kind: Service
metadata:
  name: nginx
spec:
  type: ClusterIP
  selector:
    app: nginx
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
```

3.4 - Service


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

PAUSE: what about the network ?

Concepts:

- 1 Node, 1 IP
- 1 Pod, 1 IP
- 1 Service, 1 IP

" HUM... HOW DOES IT WORK ? "

PAUSE: what about the network ?

Subnets:

- Node: Physical network ✔
- Pods: /24 per node ?
- Services: /24 per node ?

⟶ “Overlay” networks

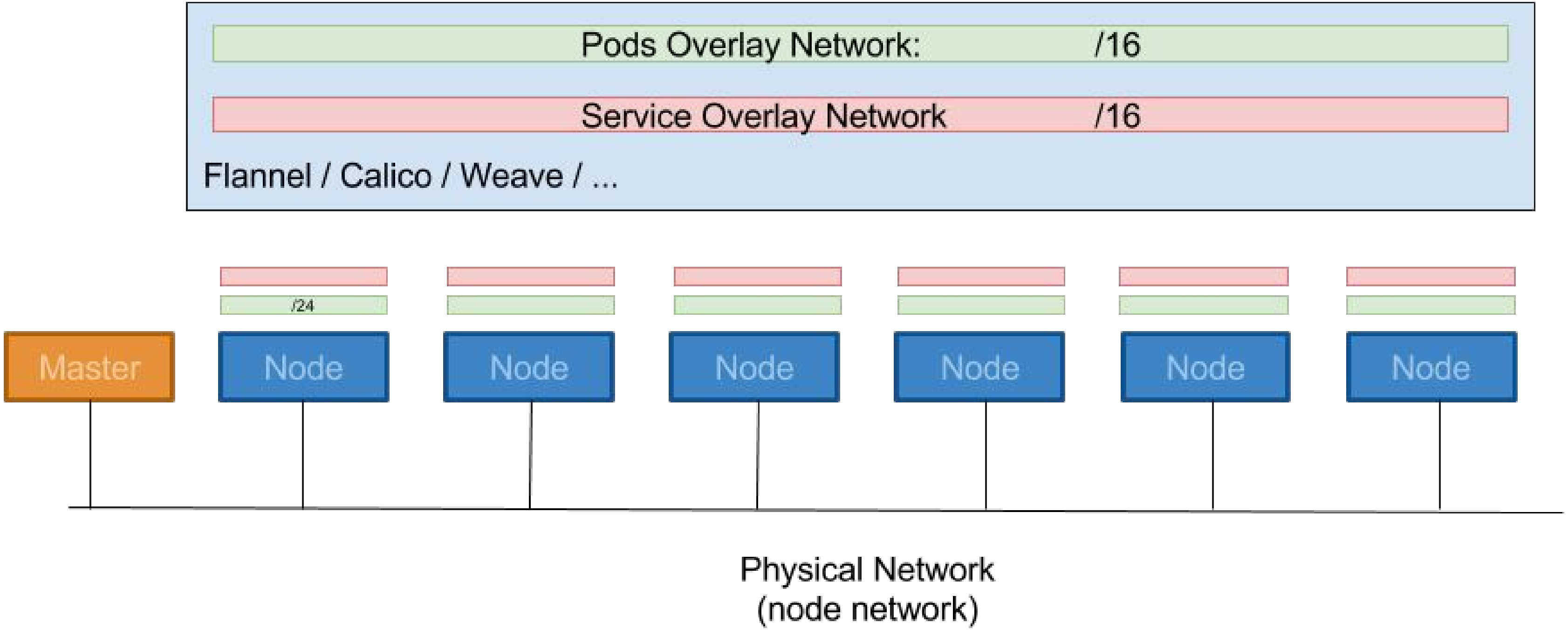
PAUSE: what about the network ?

Overlay networks ?

- Flannel
- Weave
- Calico
- ...

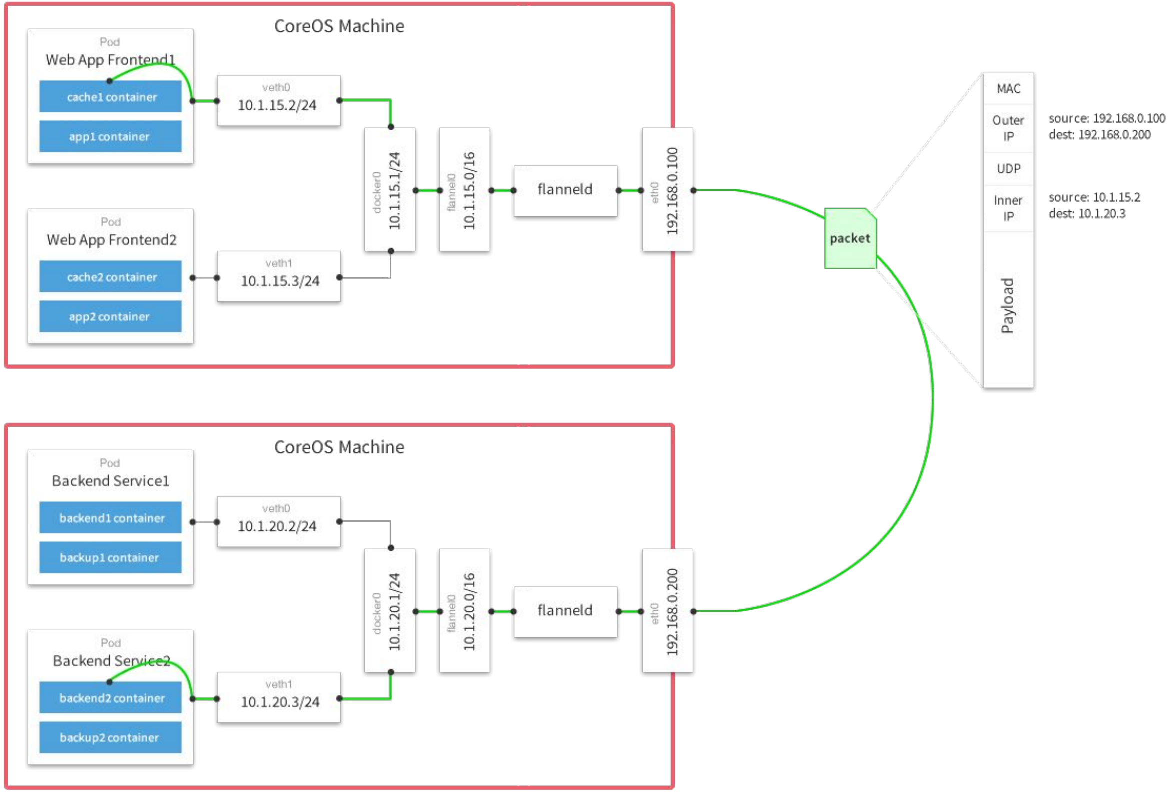
⟶ CNI (*Container Network Interface*)

PAUSE: what about the network ?


Pods Overlay Network: /16
Service Overlay Network /16
Flannel / Calico / Weave / ...
/24
Master
Node
Node
Node
Node
Node
Node
Physical Network
(node network)

leboncoin

PAUSE: what about the network ?



PAUSE: what about the network ?

```
kube-apiserver \
...
--service-cluster-ip-range=10.96.0.0/12


kube-controller-manager \
...
--cluster-cidr=10.244.0.0/16


kube-proxy \
…
--cluster-cidr=10.244.0.0/16
```

```
$ kubectl get pods -o wide

NAME READY STATUS ... IP
…        10.244.1.178   0000c998edeef6fc
…        10.244.2.13    000007398316eef7




$ kubectl get svc

NAME CLUSTER-IP       EXTERNAL-IP   PORT(S)
…      10.104.109.157   <nodes>        5000:31443/TCP
…      10.109.30.178    <nodes>        80:31881/TCP
```

3.4 - Service

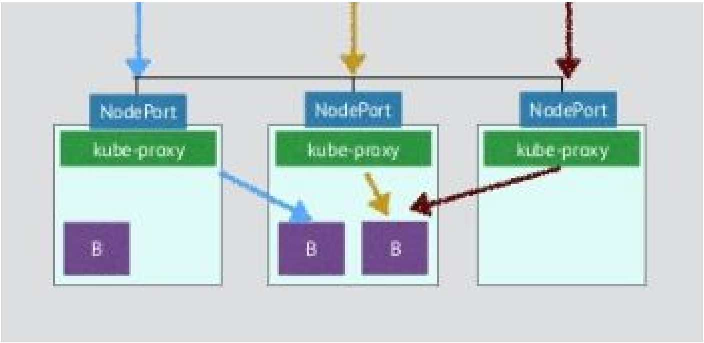
Accessing a service from outside the cluster ?

```
$ kubectl get svc

NAME CLUSTER-IP       EXTERNAL-IP   PORT(S)
…      10.104.109.157   <nodes>        5000:31443/TCP
…      10.109.30.178    <nodes>        80:31881/TCP



$ curl -I 10.3.59.61:31443   (NODE_IP:NODE_PORT)


HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 6660
Server: Werkzeug/0.12.2 Python/2.7.12
Date: Fri, 08 Sep 2017 13:10:32 GMT
```

https://kubernetes.io/docs/concepts/services-networking/service/

3.4 - Service

Accessing a service from outside the cluster ?



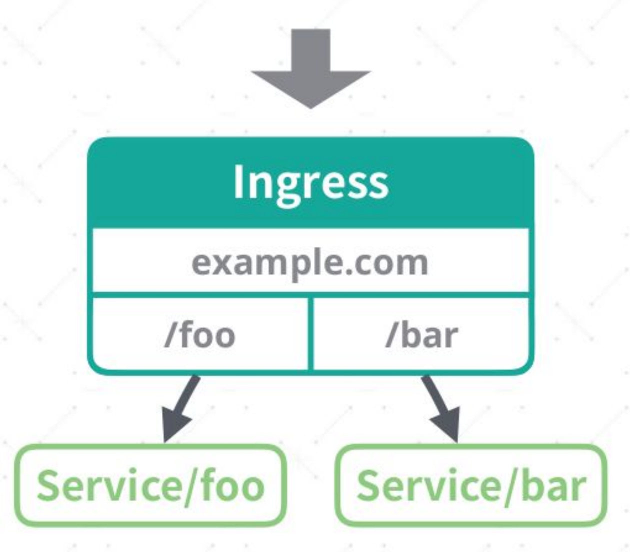
https://kubernetes.io/docs/concepts/services-networking/service/

3.5 - Ingress

What is an Ingress ?

- Services and pods have IPs only routable by the cluster network

- An Ingress is a collection of **rules** that allow inbound connections to reach the cluster services.



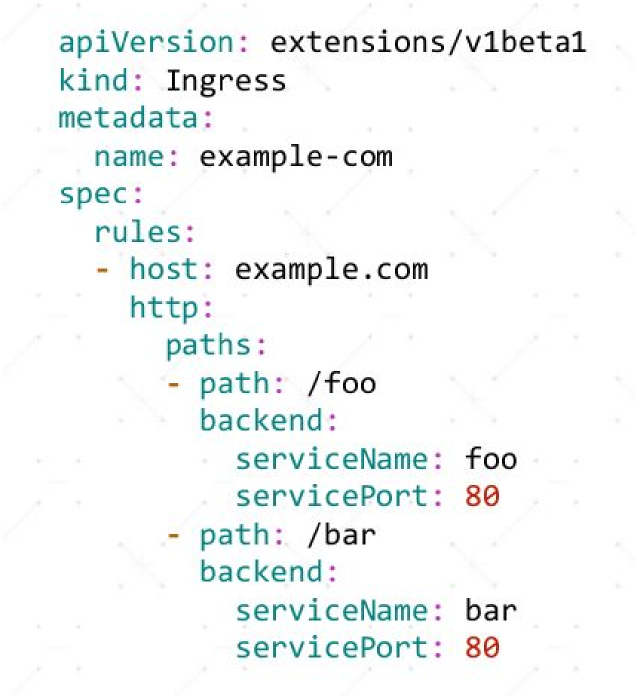
https://kubernetes.io/docs/concepts/services-networking/ingress/

3.5 - Ingress

What is an Ingress ?

- Services and pods have IPs only routable by the cluster network

- An Ingress is a collection of **rules** that allow inbound connections to reach the cluster services.



```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: example-com
spec:
  rules:
  - host: example.com
    http:
      paths:
      - path: /foo
        backend:
          serviceName: foo
          servicePort: 80
      - path: /bar
        backend:
          serviceName: bar
          servicePort: 80
```

https://kubernetes.io/docs/concepts/services-networking/ingress/

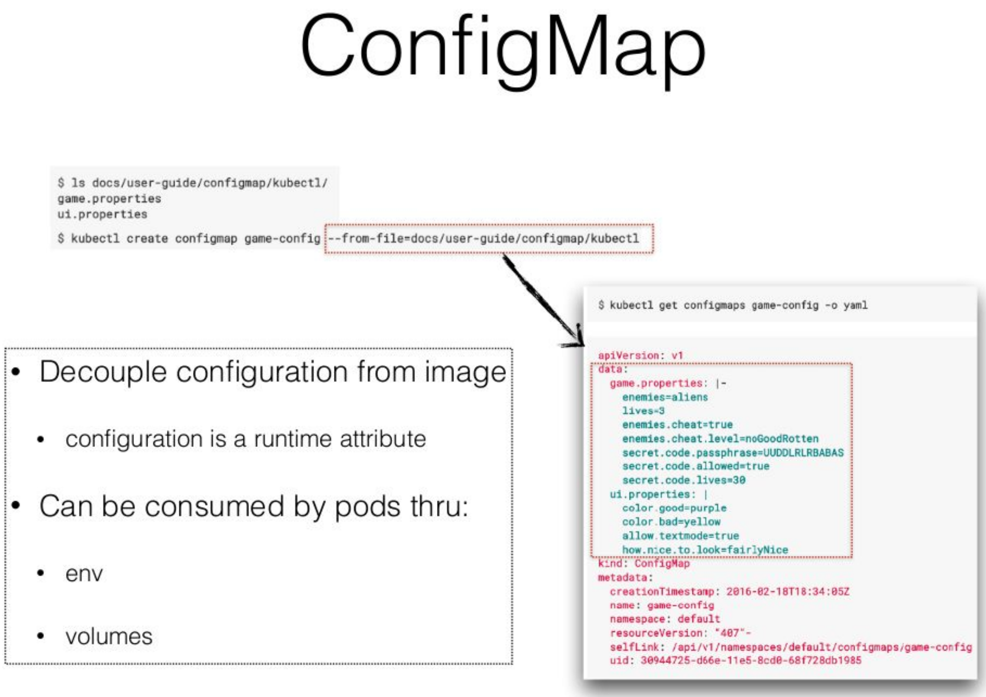
4 - Advanced

1 - ConfigMaps

2 - Secrets

3 - Volumes

4.1 - Config Maps



ConfigMap

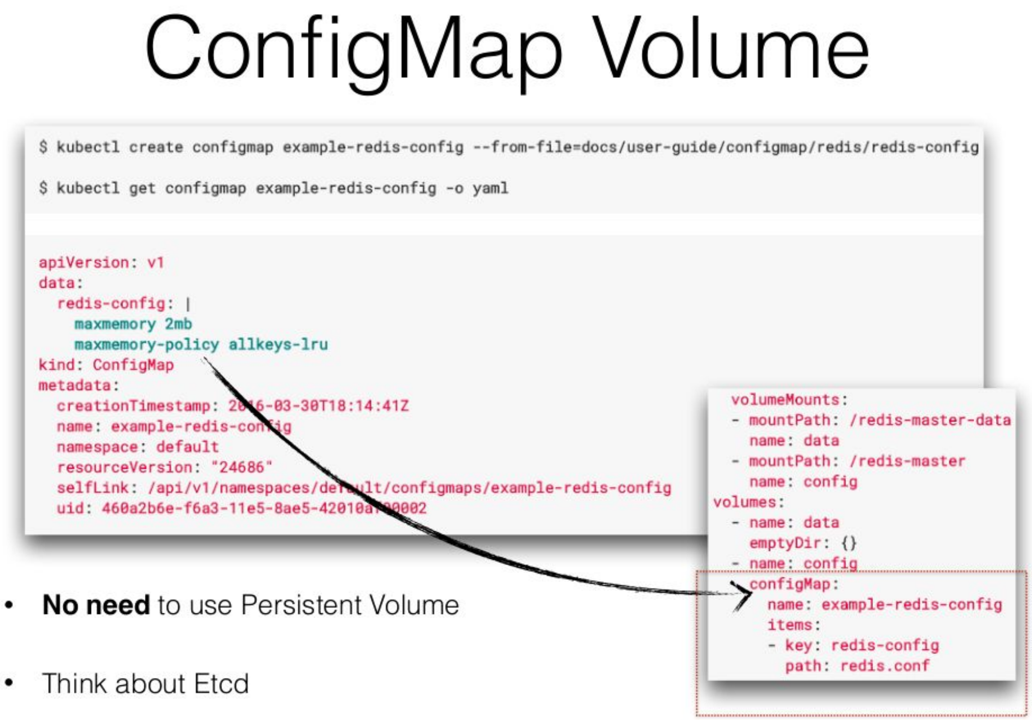
```
$ ls docs/user-guide/configmap/kubectl/
game.properties
ui.properties

$ kubectl create configmap game-config --from-file=docs/user-guide/configmap/kubectl
```

```
$ kubectl get configmaps game-config -o yaml
```

```
apiVersion: v1
data:
  game.properties: |-
    enemies=aliens
    lives=3
    enemies.cheat=true
    enemies.cheat.level=noGoodRotten
    secret.code.passphrase=UUDDLRLRBABAS
    secret.code.allowed=true
    secret.code.lives=30
  ui.properties: |
    color.good=purple
    color.bad=yellow
    allow.textmode=true
    how.nice.to.look=fairlyNice
kind: ConfigMap
metadata:
  creationTimestamp: 2016-02-18T18:34:05Z
  name: game-config
  namespace: default
  resourceVersion: "407"-
  selfLink: /api/v1/namespaces/default/configmaps/game-config
  uid: 30944725-d66e-11e5-8cd0-68f728db1985
```

- Decouple configuration from image
  - configuration is a runtime attribute
- Can be consumed by pods thru:
  - env
  - volumes

4.1 - Config Maps



ConfigMap Volume

```
$ kubectl create configmap example-redis-config --from-file=docs/user-guide/configmap/redis/redis-config

$ kubectl get configmap example-redis-config -o yaml
```

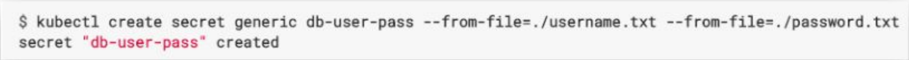
```
apiVersion: v1
data:
  redis-config: |
    maxmemory 2mb
    maxmemory-policy allkeys-lru
kind: ConfigMap
metadata:
  creationTimestamp: 2016-03-30T18:14:41Z
  name: example-redis-config
  namespace: default
  resourceVersion: "24686"
  selfLink: /api/v1/namespaces/default/configmaps/example-redis-config
  uid: 460a2b6e-f6a3-11e5-8ae5-42010af00002
```

```
  volumeMounts:
  - mountPath: /redis-master-data
    name: data
  - mountPath: /redis-master
    name: config
volumes:
  - name: data
    emptyDir: {}
  - name: config
    configMap:
      name: example-redis-config
      items:
      - key: redis-config
        path: redis.conf
```

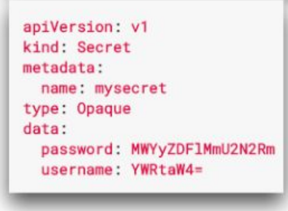
- **No need** to use Persistent Volume
- Think about Etcd

4.2 - Secrets

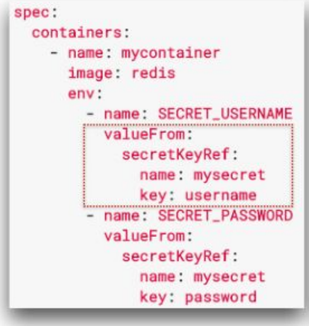
Secret



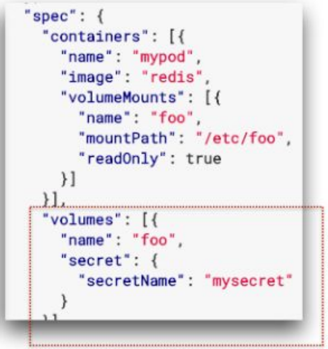
```
$ kubectl create secret generic db-user-pass --from-file=./username.txt --from-file=./password.txt
secret "db-user-pass" created
```



```
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
type: Opaque
data:
  password: MWYyZDF1MmU2N2Rm
  username: YWRtaW4=
```



```
spec:
  containers:
    - name: mycontainer
      image: redis
      env:
        - name: SECRET_USERNAME
          valueFrom:
            secretKeyRef:
              name: mysecret
              key: username
        - name: SECRET_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mysecret
              key: password
```



```
"spec": {
  "containers": [{
    "name": "mypod",
    "image": "redis",
    "volumeMounts": [{
      "name": "foo",
      "mountPath": "/etc/foo",
      "readOnly": true
    }]
  }],
  "volumes": [{
    "name": "foo",
    "secret": {
      "secretName": "mysecret"
    }
  }]
```

- Tip: credentials for accessing the k8s API is automatically added to your pods as secret

4.3 - Volumes

Persistent Volumes (-v host_path:container_path)

1. **Attach networked storage to host path**
   a. mounted to host_path

2. **Mount hots path as container volume**
   a. bind mount container_path with host_path
   b. Independent volume control loop

4.3 - Volumes

Persistent Volumes | Persistent Volume “Claim”

1. **SRE / Administrator**
    a. **Create a Volume** (access, capacity, recycling policy)

2. **Dev / Users**
    a. **Request a volume** (Access mode, resource, selector)

4.4 - Health Checks

Readiness → Is the app ready to start serving traffic?

- Won’t be added to a service endpoint until it passes
- Required for a “production app” in my opinion

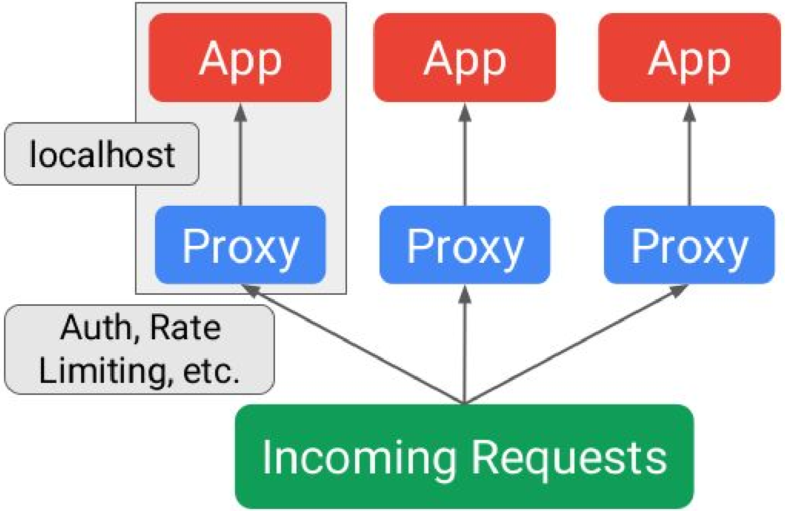
Liveness → Is the app still running?

- Default is “process is running”
- Possible that the process can be running but not working correctly
- Good to define, might not be 100% necessary

5 - Tips

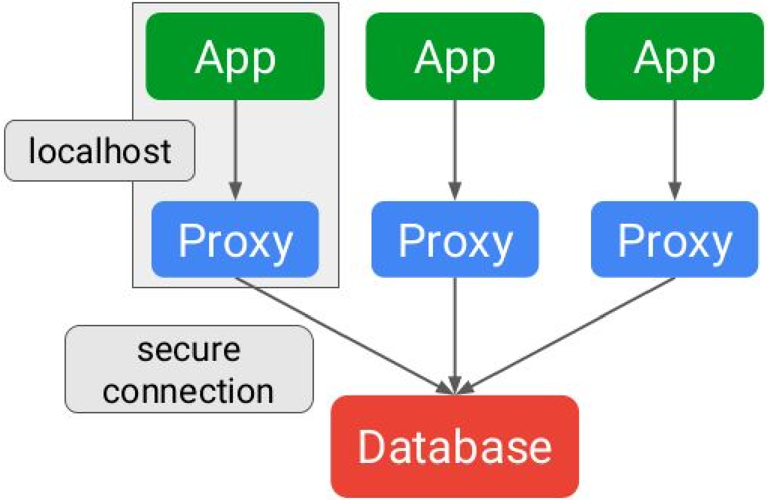
5 - Tips

Use proxies everywhere !



5 - Tips

Use proxies everywhere !



6 - Workshop

https://github.com/xakraz/k8s-workshops/tree/master/1

Sources

Shamefully raped from:

- https://speakerdeck.com/superbrothers/how-kubernetes-works
- https://www.slideshare.net/resouer/kubernetes-walk-through-zhanglei