OpenShift 4

Making introductions

A comprehensive overview of “OpenShift Container Platform”, accompanied with interactive Lab exercises.

Samuel Terburg
Red Hat Certified Architect
Cloud-Native Expert

2020-01-14

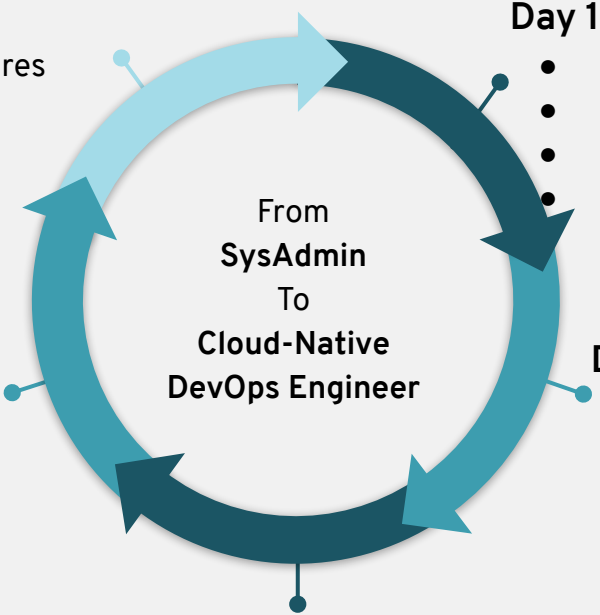
Agenda

Day 5 - Best Practices

- Reference Architectures
- Use Cases
- App OnBoarding

Day 1 - Overview

- Cloud-Native market
- Container Concepts
- Platform Concepts
- Platform Architecture
- OpenShift - Technical Deep Dive



Day 4 - Misc

- RBAC, IAM
-

Day 2 - App Deployment

- Technical Deep Dive
- Labs
- Compute: Pods
- Networking: Services, Routes
- Storage: Persistent Volumes

Day 3 - App Build

- Labs
- Source-2-Image
- Jenkins

Client Setup

Gets you started

Interactive Workshop

OpenShift WebConsole	https://console-openshift-console.apps.learn.ont.belastingdienst.nl
OpenShift CLI	oc login -u <vdi-user> https://api.learn.ont.belastingdienst.nl:6443
Workshop url	https://lab-getting-started-workshops.apps.learn.ont.belastingdienst.nl
Concluence	https://devtools.belastingdienst.nl/confluence/displayJOS/OpenShift+Opleiding+Omgeving
Source code / Slides	https://devtools.belastingdienst.nl/bitbucket/users/wolfj09/repos/openshift-workshop/browse/resources

COMMANDS

Help	
oc	Openshift Client
oc types oc api-resources	Brief description of common used {object-types}
oc explain {object-type}	Details the fields/parameters of a specific {object-type}
oc {verb} --help	Help on command-line syntax (for specific {verb})

Getting Started	
oc login	Openshift Client
oc new-project	Create new Project
oc new-app	Provision new containerized application stack within your project.

oc {verb} {object-type} {object-identifier}

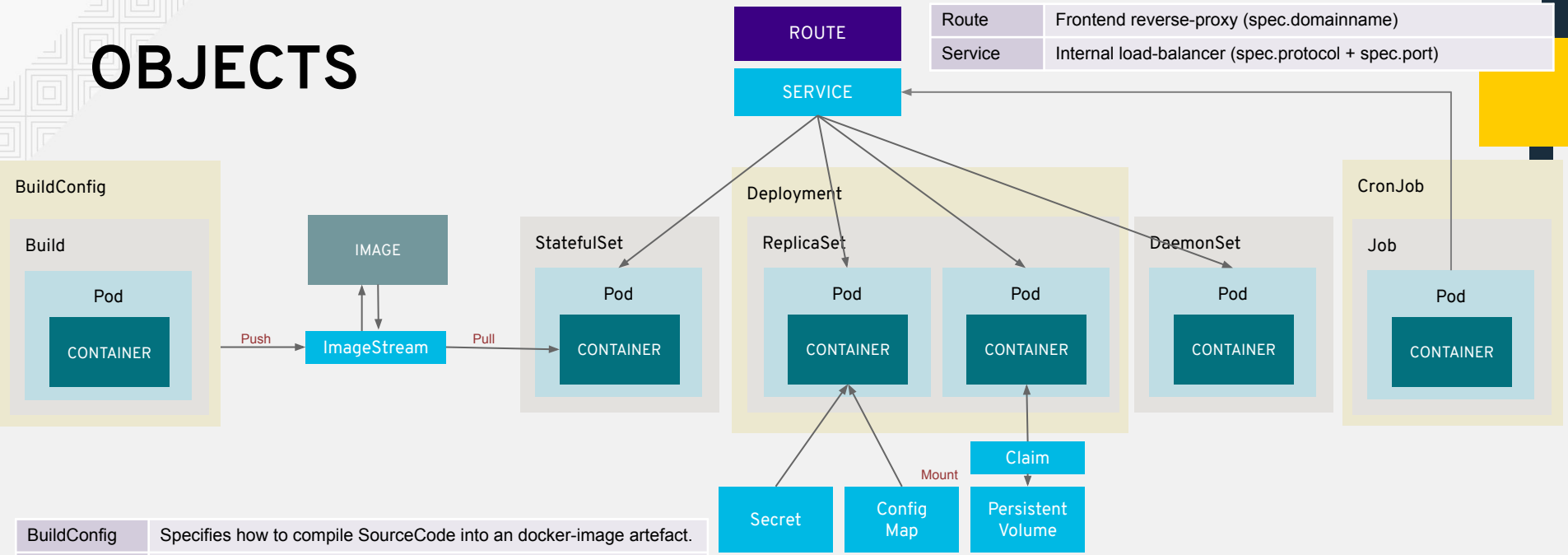
oc get pod myapp-1-abcde

{verb}	
get	A (mount)pointer to Network Storage (spec.connectionstring)
create	A mountable property file (spec.data[])
edit	A mountable base64-encoded file (spec.data[])
delete	
rsh / exec	Remote shell into a Container
project	Switch current-context to other namespaces
new-project	Create new Project
new-app	Provision new containerized application stack
cp / rsync	Copy files in/out containers

Examples	
oc get projects	Overview of all Projects running
oc get pods --all-namespaces -o wide	Overview of all Pods running
oc new-project lite3-prd oc new-app --template=mytomcat --image=tomcat8	Deploy new application stack
oc start-build bc/mytomcat	Compile new docker-image
oc -n lite3 edit configmap mytomcat-properties	Change config/property-file
oc rollout latest deployment/mytomcat	Deploy new version
oc delete pod/mytomcat-1-abcde	Restart app
oc describe svc mytomcat	Detailed Info about an object and its state
oc expose svc/mytomcat –hostname=myapp.swift.com	Expose your app to the public
oc tag mytomcat:v1.0 mytomcat:prod	Promote you app to Production

OBJECTS

Route	Frontend reverse-proxy (spec.domainname)
Service	Internal load-balancer (spec.protocol + spec.port)



BuildConfig	Specifies how to compile SourceCode into an docker-image artefact.
Build	Runs the build process (spec.gitref + spec.buildImage + spec.targetImage)
ImageStream	A pointer to an external image (spec.imageUrl)
Image	Application binary (content)

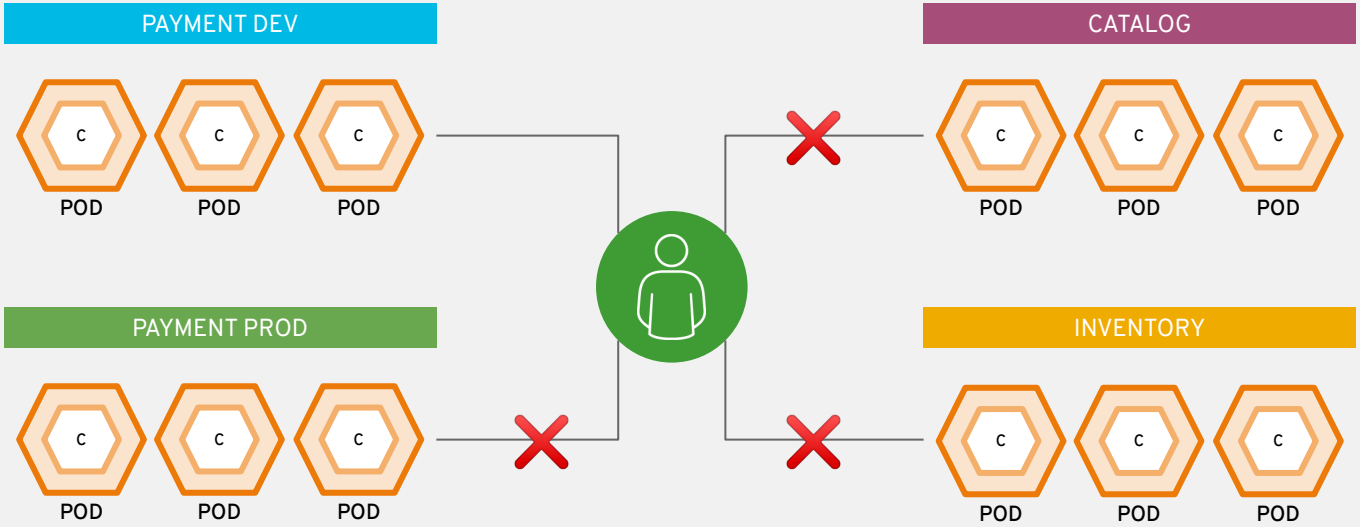
PersistentVolumeClaim	A “request” for storage space (spec.size)
PersistentVolume	A (mount)pointer to Network Storage (spec.connectionstring)
ConfigMap	A mountable property file (spec.data[])
Secret	A mountable base64-encoded file (spec.data[])

DeploymentConfig	Specifies how to deploy your application (rolloutStrategy + imageChangeTriggers)
ReplicaSet	Monitors the Pods and restarts if needed (spec.healthcheck + spec.replicas)
Pod	A grouping of tightly-coupled containers (spec.containers[] + spec.node)
Container	Running your application process (spec.command + spec.security
StatefulSet	Same a Deployment, but then for Statefull workloads
DaemonSet	Same a Deployment, but exactly 1 Pod per Node. (spec.nodeSelector)
CronJob	Schedules a Pod at a specified time. (spec.schedule)
Job	Monitors the one-off Pod for successful completion, restarts if needed.

Kubernetes Resource Definitions

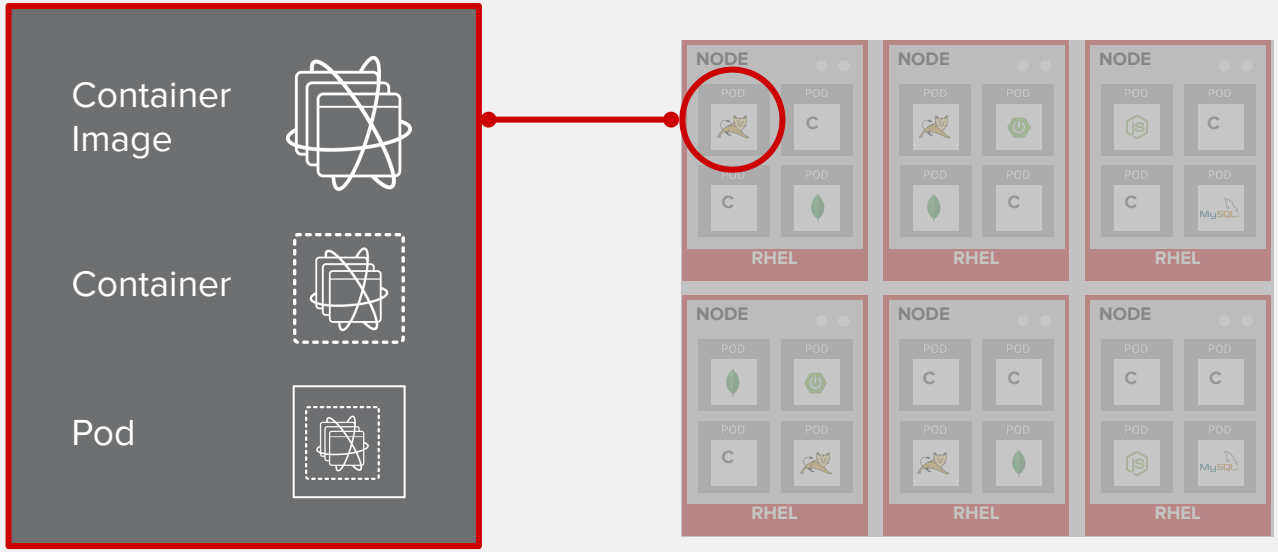
What types of workloads can you deploy on top of a Container Platform...

projects isolate apps across environments, teams, groups and departments




```
> oc new-project workshop-<vdi-user>
> oc project
> oc get projects
```

APPS RUN IN CONTAINERS

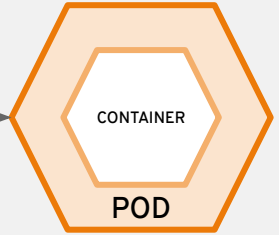



```
> oc get pods --all-namespaces
> oc -n <namespace> describe pod <name>
```

It all starts with an image

```
kind: ImageStream
metadata:
name:
lab-getting-started-session
spec:
  tags:
  - name: latest
    from:
      kind: DockerImage
      name: docker.io/httpd:latest
```

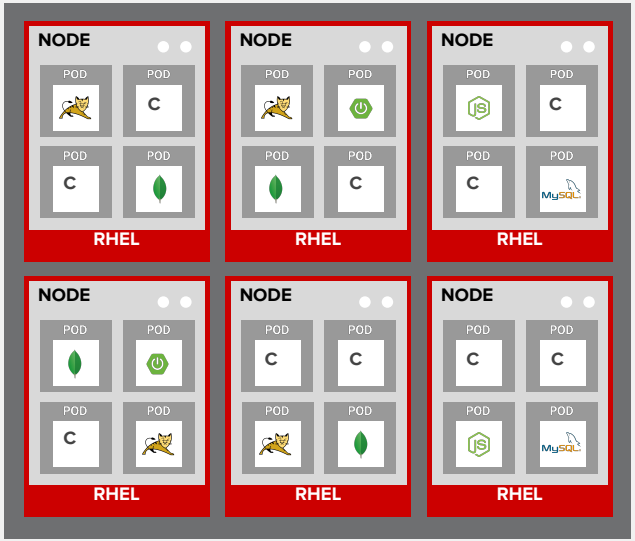
```
kind: Pod
metadata:
    name: mydb
spec:
spec:
  containers:
  - name: backend
    image: mysql
    ports:
    - containerPort: 3306
    volumeMount:
    - name: data
      mount: /var/lib/mysql
  volumes:
  - name: data
    claim:
      requests:
        storage: 100Gi
```



10.140.4.44

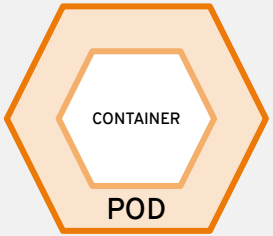
```
> oc -n openshift get imagestreams
> oc -n openshift describe imagestream httpd
> oc import-image
```

PODS ARE THE UNIT OF ORCHESTRATION

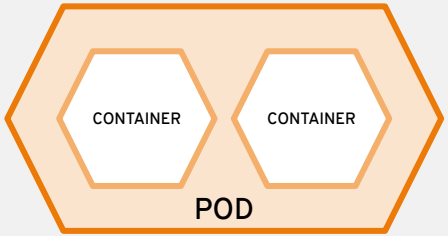


> oc new-project user00; oc new-app jenkins-ephemeral
> oc describe deploymentconfig/jenkins

containers are wrapped in pods which are units of deployment and management

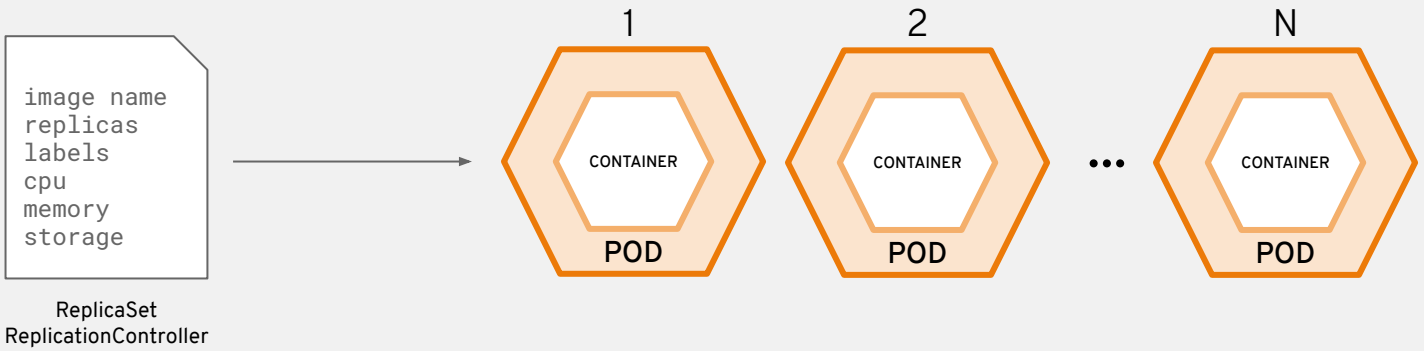


10.140.4.44



10.15.6.55

`ReplicationControllers` & `ReplicaSets` ensure a specified number of pods are running at any given time



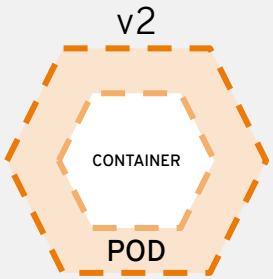
```
kind: “DeploymentConfig“
metadata:
   name: “myApp“
spec:
   replicas: 2
   selector:
      app: myapp
   template:
      metadata:
         name: myapp
         labels:
            app: mine
      spec:
         containers:
          - name: frontend
            image: jboss-eap:latest
            ports:
             - containerPort: 80
   triggers:
   - type: "ImageChange“
     from:
        kind: “Image”
        name: “myapp:latest
```

```
kind: “DeploymentConfig“
metadata:
   name: “myApp“
spec:
   replicas: 2
   template:
      spec:
         containers:
            - name: frontend
            - name: flyway
   strategy:
      type: rolling
      rollingParams:
        pre:
           execNewPod:
              containerName: flyway
              volumes: [‘git’]
              command: “flyway do”
        post:
            tagImage:
               containerName: frontend
               to: “frontend:prod”
   triggers: …
```

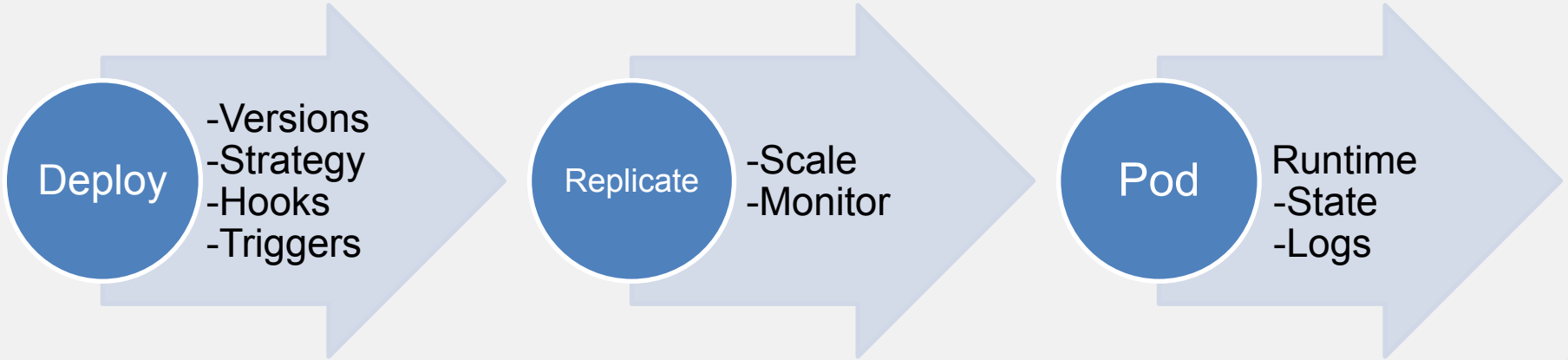
and

ons define how to

s of Pods



Deployment Process



- MyJBossApp

- MyJBossApp-v1 (2x)
- MyJBossApp-v2 (4x)

- MyJBossApp-v1-abcde
- MyJBossApp-v1-rando


```
> oc new-app httpd
> oc scale -replicas=2 dc/frontend
> oc rollout latest dc/frontend
```

```
> oc get pods -o wide -w
> curl http://<pod-ip>:8080/
```

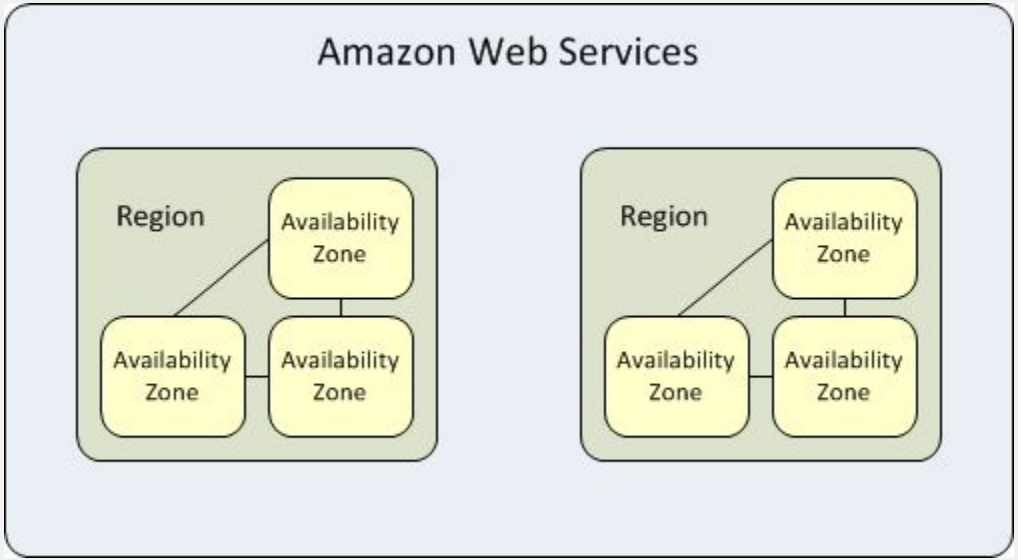
```
> oc new-app httpd
> oc scale -replicas=2 dc/frontend
> oc rollout latest dc/frontend
> oc rollback dc/frontend

> oc edit dc/frontend
> oc describe pod/<pod-name>

> oc logs <pod-name>
> oc rsh <pod-name> bash
```

```
> oc get pods -o wide -w
> curl http://<pod-ip>:8080/
```

PLACEMENT BY POLICY

```
$ oc edit configmap scheduler-policy -n openshift-config

apiVersion: v1
data:
  policy.cfg: |
    {
        "kind" : "Policy",
        "apiVersion" : "v1",
        "predicates" : [
                {"name" : "MaxGCEPDVolumeCount"},
                {"name" : "GeneralPredicates"},
                {"name" : "MaxAzureDiskVolumeCount"},
                {"name" : "MaxCSIVolumeCountPred"},
                {"name" : "CheckVolumeBinding"},
                {"name" : "MaxEBSVolumeCount"},
                {"name" : "PodFitsResources"},
                {"name" : "MatchInterPodAffinity"},
                {"name" : "CheckNodeUnschedulable"},
                {"name" : "NoDiskConflict"},
                {"name" : "NoVolumeZoneConflict"},
                {"name" : "MatchNodeSelector"},
                {"name" : "HostName"},
                {"name" : "PodToleratesNodeTaints"}
                ],
 "priorities" : [
                {"name" : "LeastRequestedPriority", "weight"
                {"name" : "BalancedResourceAllocation", "weig
                {"name" : "ServiceSpreadingPriority", "weight
                {"name" : "NodePreferAvoidPodsPriority", "wei
                {"name" : "NodeAffinityPriority", "weight" :
                {"name" : "TaintTolerationPriority", "weight"
                {"name" : "ImageLocalityPriority", "weight" :
                {"name" : "SelectorSpreadPriority", "weight"
                {"name" : "InterPodAffinityPriority", "weight
                {"name" : "EqualPriority", "weight" : 1}
                ]
        }
```

PLACEMENT BY POLICY
nodeLabels: Zone + Region



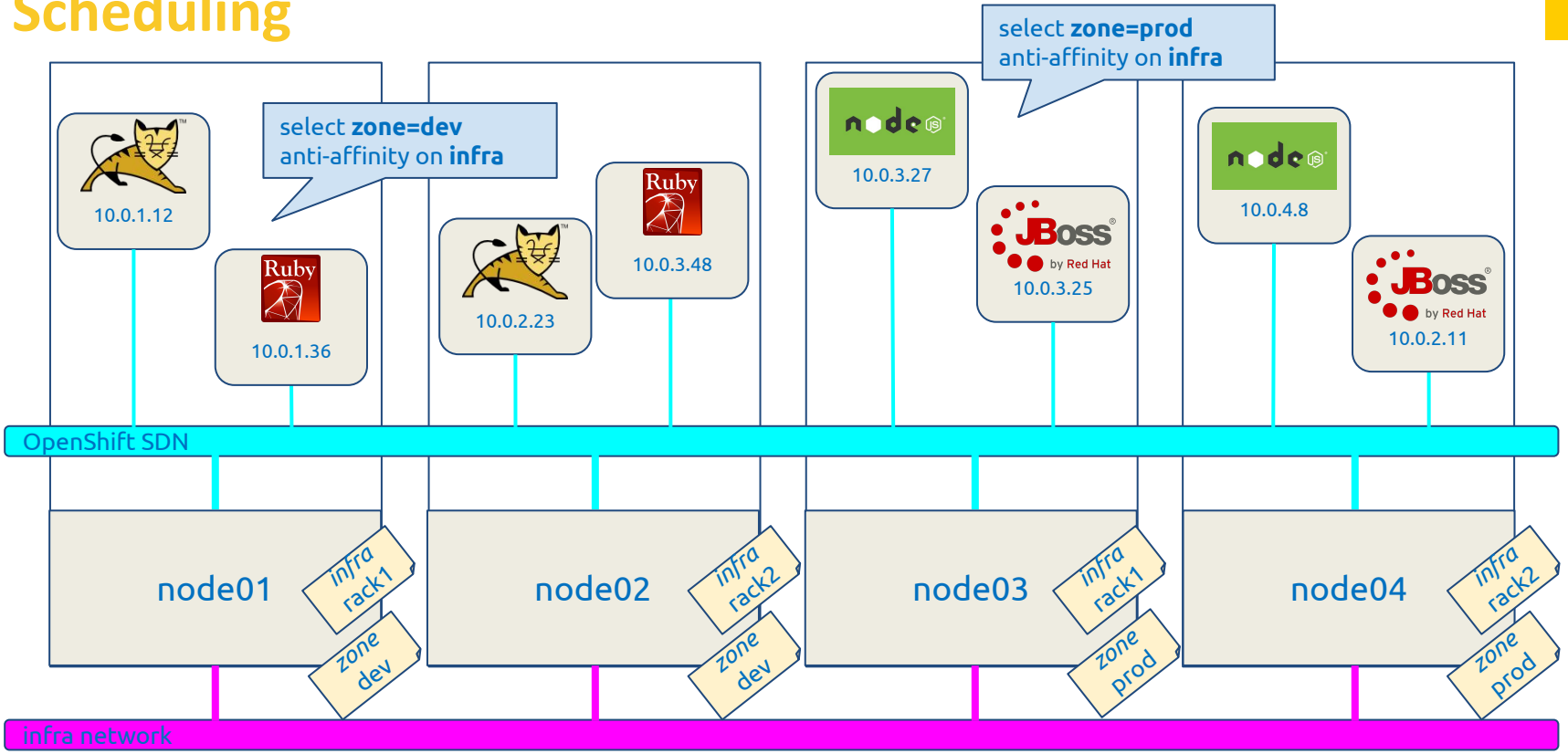
Pods belonging to same “Service”:

MustHave:
- Affinity:
  - Region

ShouldHave:
- Anti-Affinity:
  - Zone (aka failure-domain)

> ssh master1 cat /etc/origin/master/scheduler.json

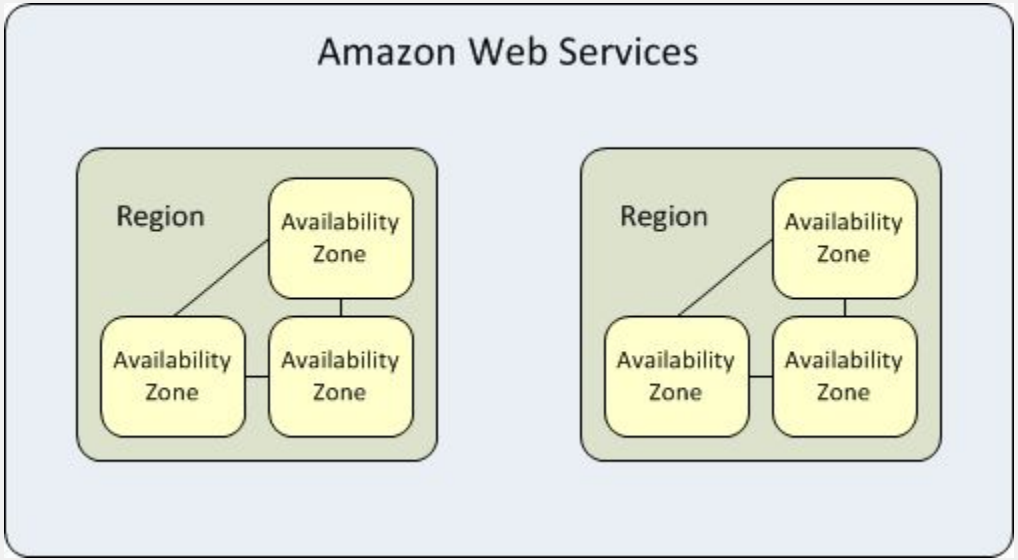
Scheduling


select zone=dev
anti-affinity on infra
select zone=prod
anti-affinity on infra
10.0.1.12
10.0.1.36
10.0.2.23
10.0.3.48
10.0.3.27
10.0.3.25
10.0.4.8
10.0.2.11
OpenShift SDN
node01
infra rack1
zone dev
node02
infra rack2
zone dev
node03
infra rack1
zone prod
node04
infra rack2
zone prod
infra network

Node Selector

```
oc get nodes --show-labels

> oc label node node01.apeldoorn.internal zone=A
> oc label node node02.apeldoorn.internal zone=B
> oc label node node03.apeldoorn.internal zone=C

> oc scale --replicas=3 dc/httpd

> oc edit dc/httpd
spec:
  template:
    spec:
      nodeSelector:
        zone: A
```

PLACEMENT BY POLICY
Pod Affinity rules



Specify complex query

> ssh master1 cat /etc/origin/master/scheduler.json

PLACEMENT BY POLICY
Taint + Tolerations

...

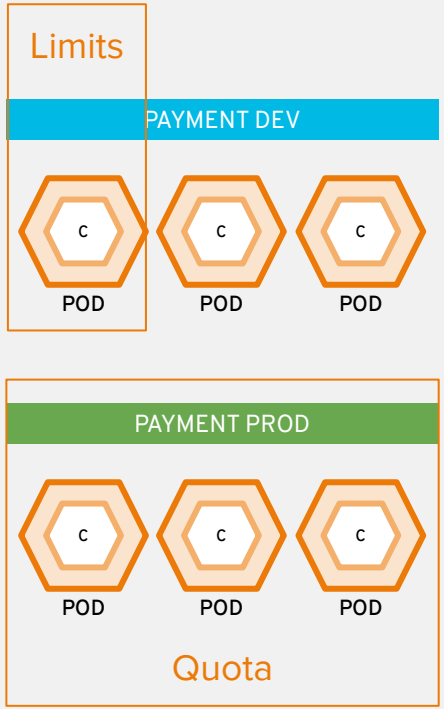
> ssh master1 cat /etc/origin/master/scheduler.json

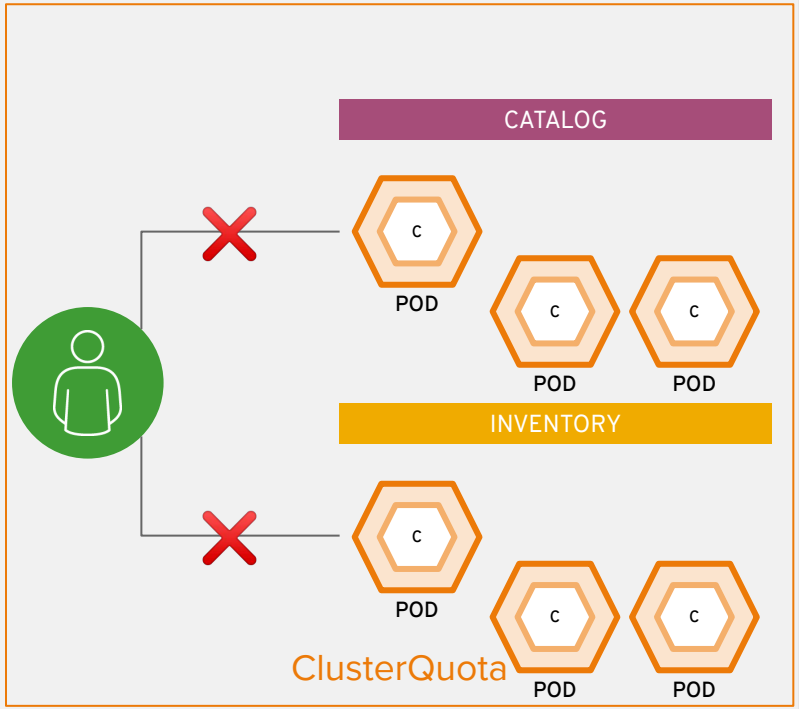
PLACEMENT BY POLICY
Resource Reservations

...


```
> oc set resources dc/httpd --requests=cpu=16,memory=256Gi
> oc set resources dc/httpd --requests=cpu=0,memory=0
```

Limits & Quotas





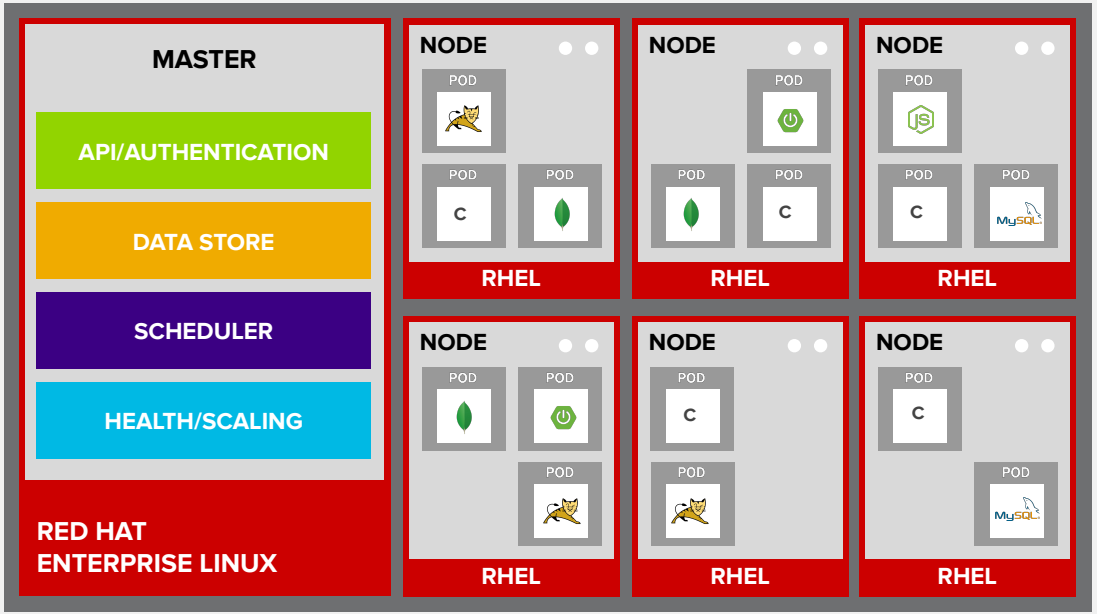

```yaml
apiVersion: "v1"
kind: "LimitRange"
metadata:
  name: "default-limits"
spec:
  limits:
    - type: "Pod"
      max:
        cpu: "2"
        memory: "1Gi"
      min:
        cpu: "200m"
        memory: "6Mi"
    - type: "Container"
      max:
        cpu: "2"
        memory: "1Gi"
      min:
        cpu: "100m"
        memory: "4Mi"
      default:
        cpu: "300m"
        memory: "200Mi"
      defaultRequest:
        cpu: "200m"
        memory: "100Mi"
      maxLimitRequestRatio:
        cpu: "10"
```

```
> oc set resources dc/httpd --limits=cpu=200m,memory=512Mi --requests=cpu=100m,memory=256Mi
> oc create quota small --hard=cpu=4,memory=16G,pods=3,persistentvolumeclaims=10
> oc create clusterquota limit-bob --project-annotation-selector=openshift.io/requester=user-bob --hard=pods=10
> oc create -f default-limits.yaml
```

Quota's & Limits

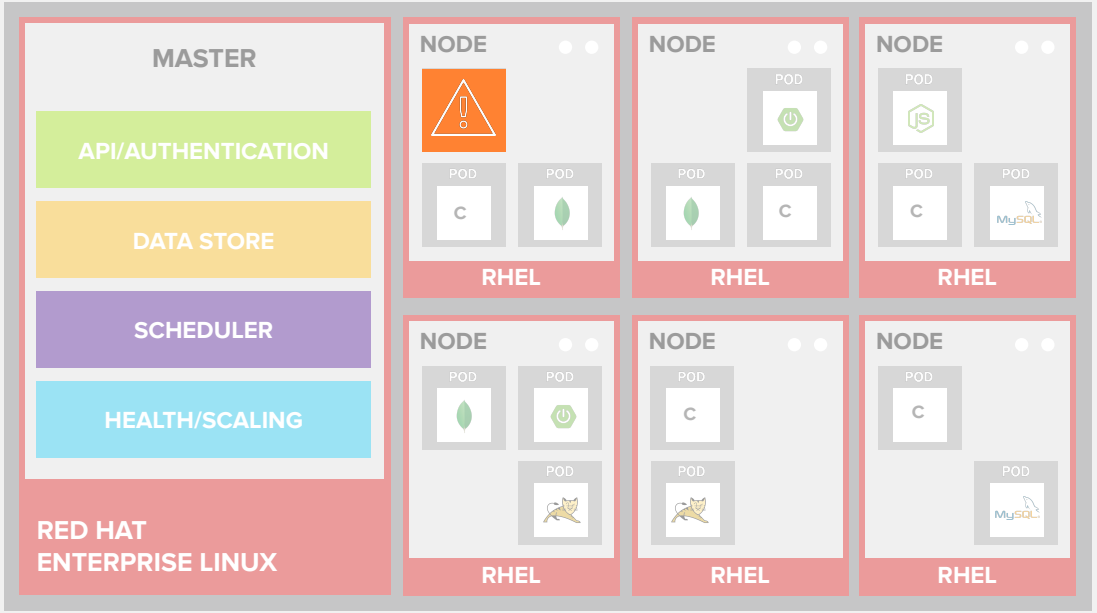
```
> oc set resources dc/httpd --limits=cpu=200m,memory=512Mi
--requests=cpu=100m,memory=256Mi
> oc describe dc/httpd
> oc get -o yaml dc/httpd
> oc get -o yaml pod/httpd-4-abcde

> oc create quota small --hard=cpu=4,memory=16G,pods=3,persistentvolumeclaims=10
> oc describe quota small

> oc create clusterquota limit-<vdi-user>
--project-annotation-selector=openshift.io/requester=<vdi-user> --hard=pods=10
> oc describe clusterquota

> oc create -f default-limits.yaml
> oc delete pods --all
> oc describe quota

> oc describe node <node>
```

AUTO-HEALING FAILED CONTAINERS



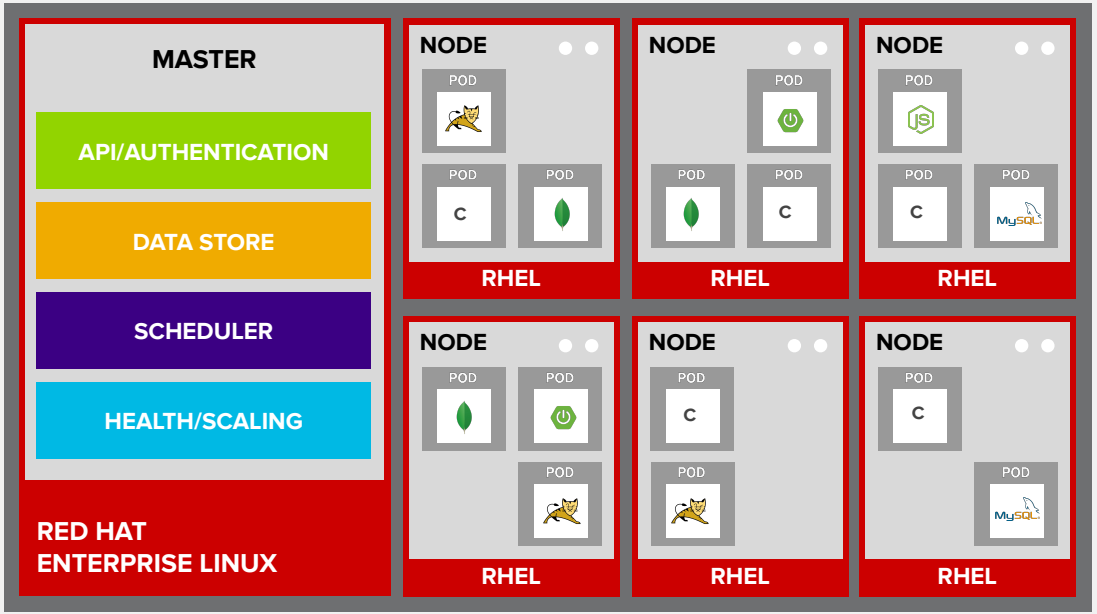

```
> oc get pods -o wide --watch &
> oc delete pods --all
> oc rsh <pod> kill -9 1
```

AUTO-HEALING FAILED CONTAINERS



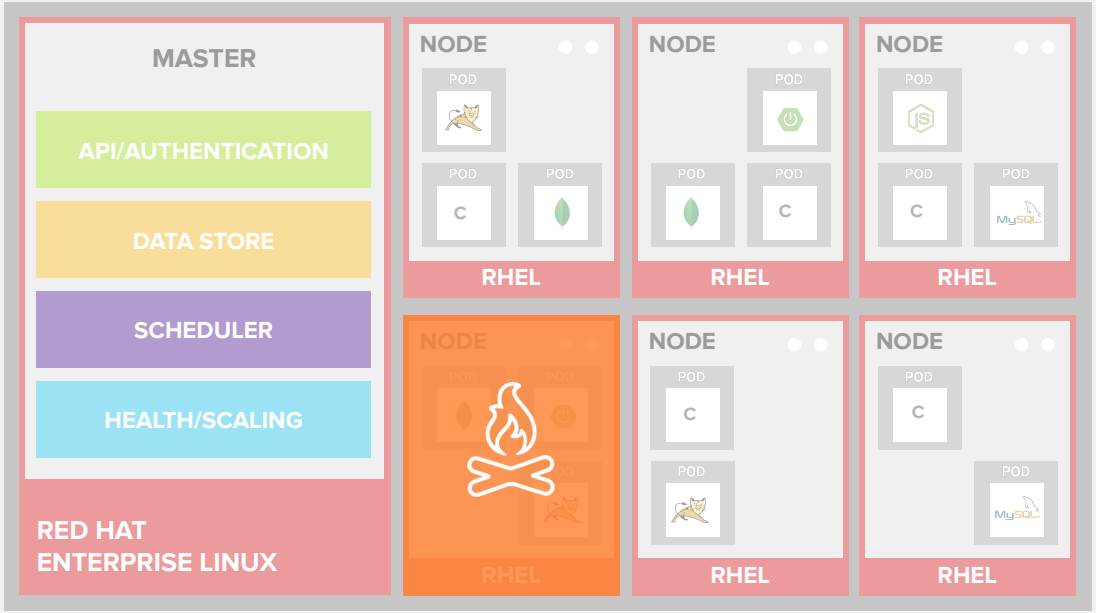

```
> oc get pods -o wide --watch &
> oc delete pods --all
> oc rsh <pod> kill -9 1
```

AUTO-HEALING FAILED CONTAINERS



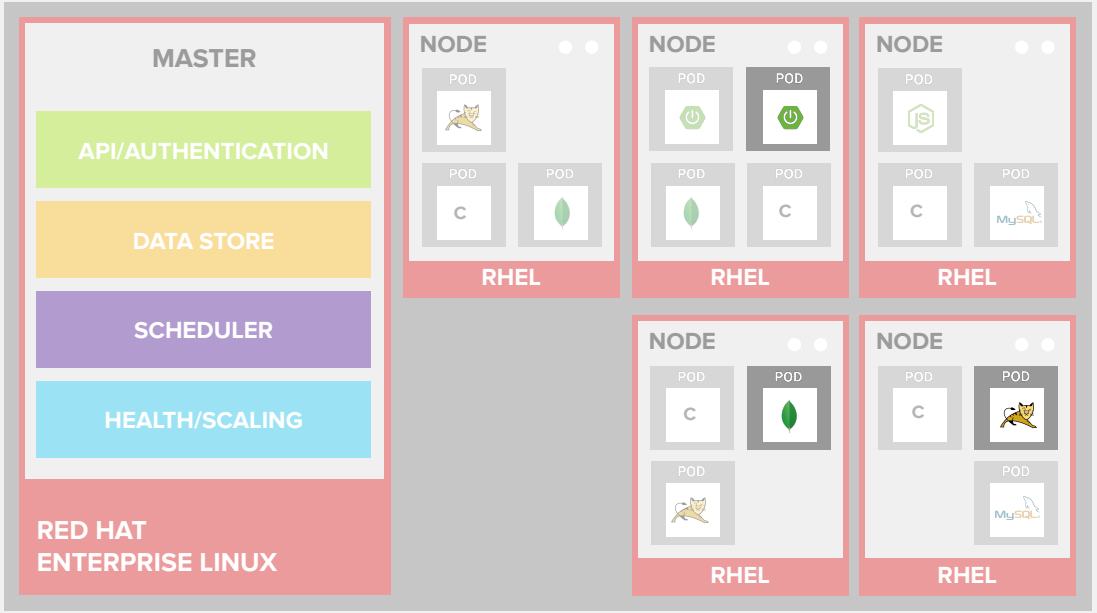

```
> oc get pods -o wide --watch &
> oc delete pods --all
> oc rsh <pod> kill -9 1
```

AUTO-HEALING FAILED NODES




```
> oc get pods -o wide --watch &
> oc delete pods --all
> oc rsh <pod> kill -9 1
```

AUTO-HEALING FAILED NODES




```
> oc get pods -o wide --watch &
> oc delete pods --all
> oc rsh <pod> kill -9 1
```

Health Checks

- LivenessProbe
- ReadinessProbe

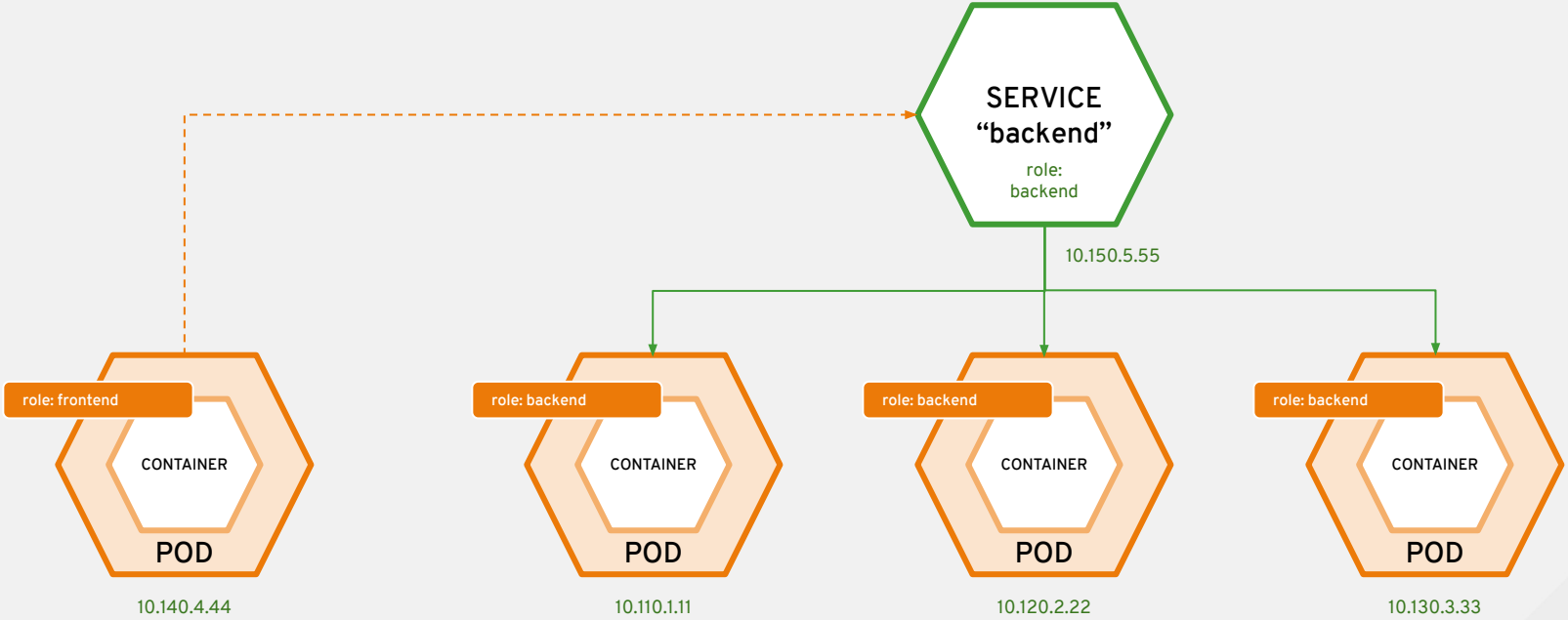
HTTP
TCP
Shell


```
> oc set probe dc/httpd --readiness --get-url=http://127.0.0.1:8080/index.html --initial-delay-seconds=5
> oc set probe dc/httpd --liveness --open-tcp=8080
```

Networking

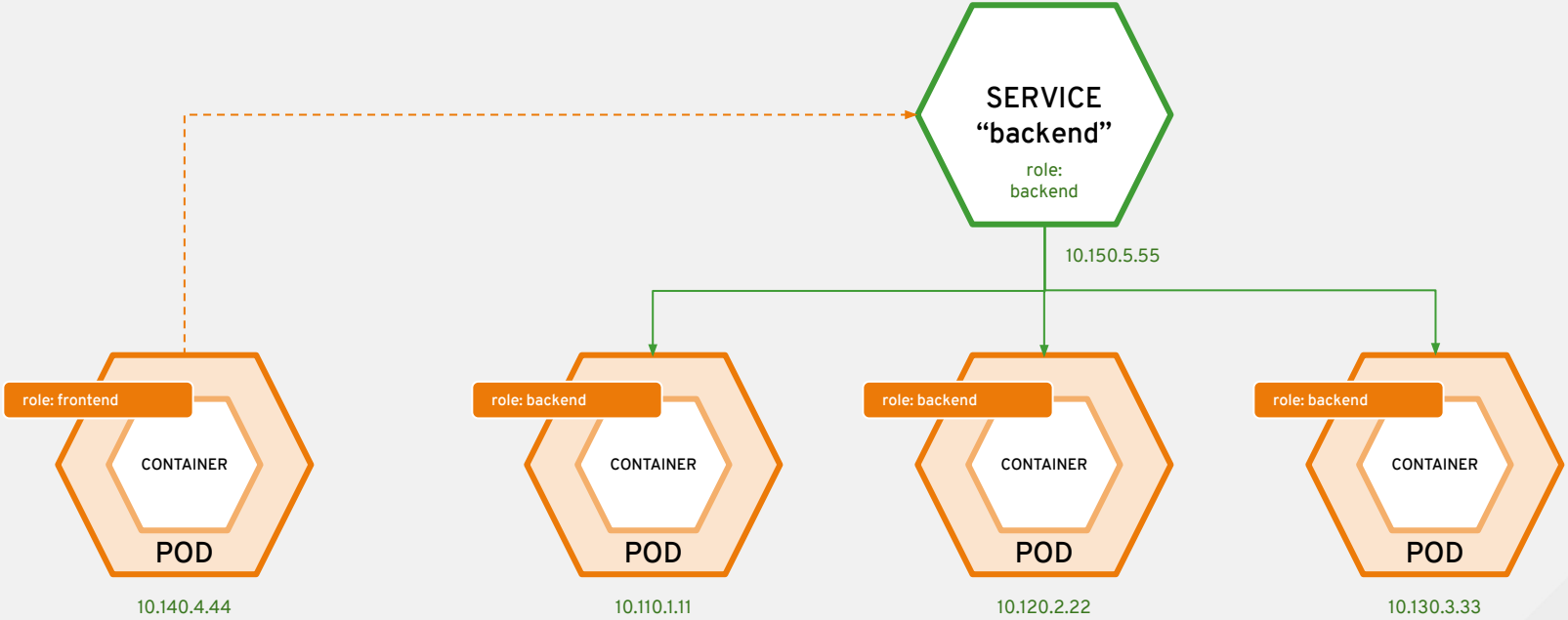
Knitting it all together

services provide internal load-balancing




```
> oc describe svc/httpd |egrep “Selector|Endpoints”
> oc get pods -o wide -l app=httpd
```

services provide **service discovery** across pods



> oc rsh dc/httpd env |grep HTTPD

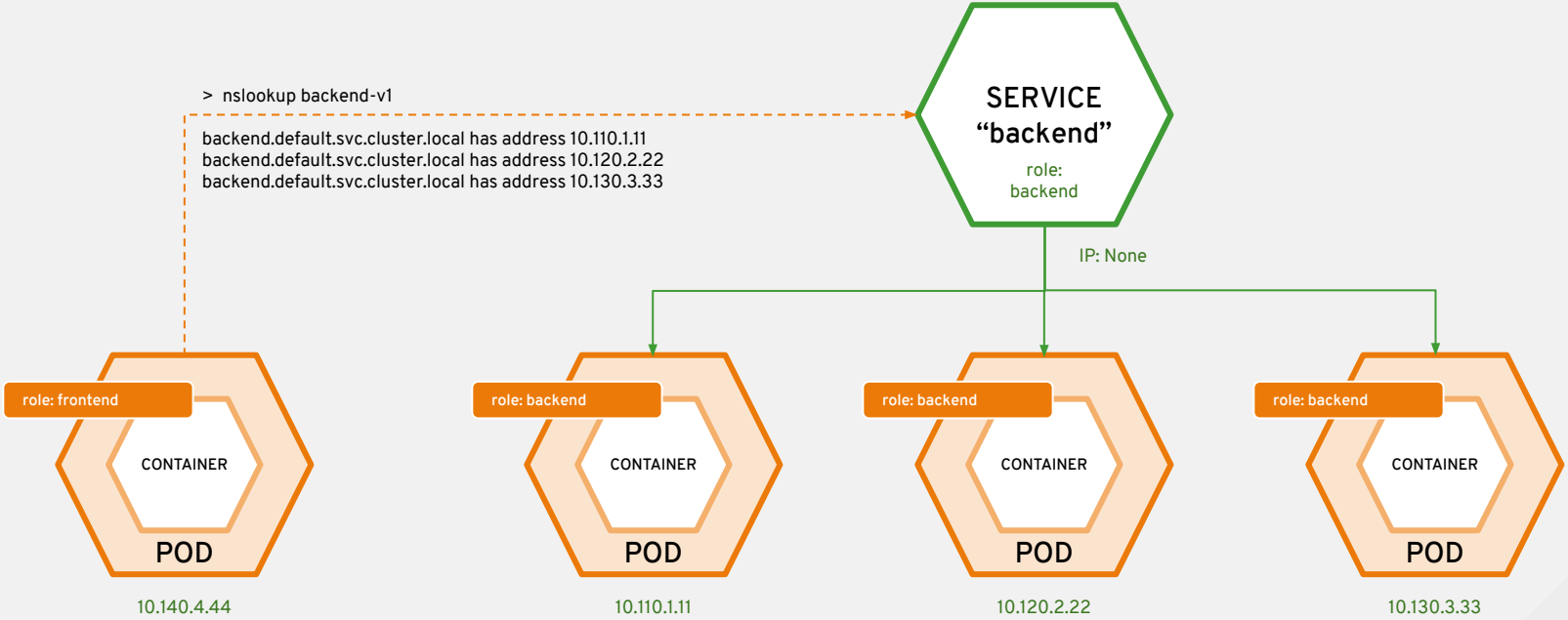
Label Selector

```
> oc delete all -l app=httpd
> oc new-app httpd --name=frontend-v1 -l app=frontend,version=1
> oc new-app httpd --name=backend-v1  -l app=backend,version=1
> oc new-app httpd --name=backend-v2  -l app=backend,version=2
> oc get pods -o wide --show-labels -l app=backend
> oc get pods         --show-labels -l app=backend,version=1

> oc get services -o wide
> oc describe svc/backend-v1
> oc get endpoints

> oc get svc/backend-v1 -o yaml --export \
|sed s/backend-v1/backend/g \
|grep -v version:       \
|tee /dev/stderr        \
|oc create -f -

> oc get ep
```

Service Discovery

```
echo "1" >index1.html
echo "2" >index2.html
echo "2" >index3.html

> oc scale --replicas=2 dc/backend-v2
> oc get pods,ep

> oc cp index1.html backend-v1-abcde:/var/www/html/index.html
> oc cp index2.html backend-v2-fghij:/var/www/html/index.html
> oc cp index3.html backend-v2-klmni:/var/www/html/index.html

> oc rsh dc/frontend
host backend
curl http://backend:8080
curl http://backend:8080
curl http://backend:8080
curl http://backend:8080
```

services provide **peer discovery** across pods



> oc rsh dc/frontend host backend-v1

Peer Discovery

```
> oc get -o yaml  svc/httpd-v1 |sed 's/clusterIP:.*/clusterIP: None/g' |tee >(oc
create -f -) >(oc delete -f -)

> oc get svc
> oc rsh dc/frontend host backend-v1
> oc rsh dc/frontend host backend-v2
```

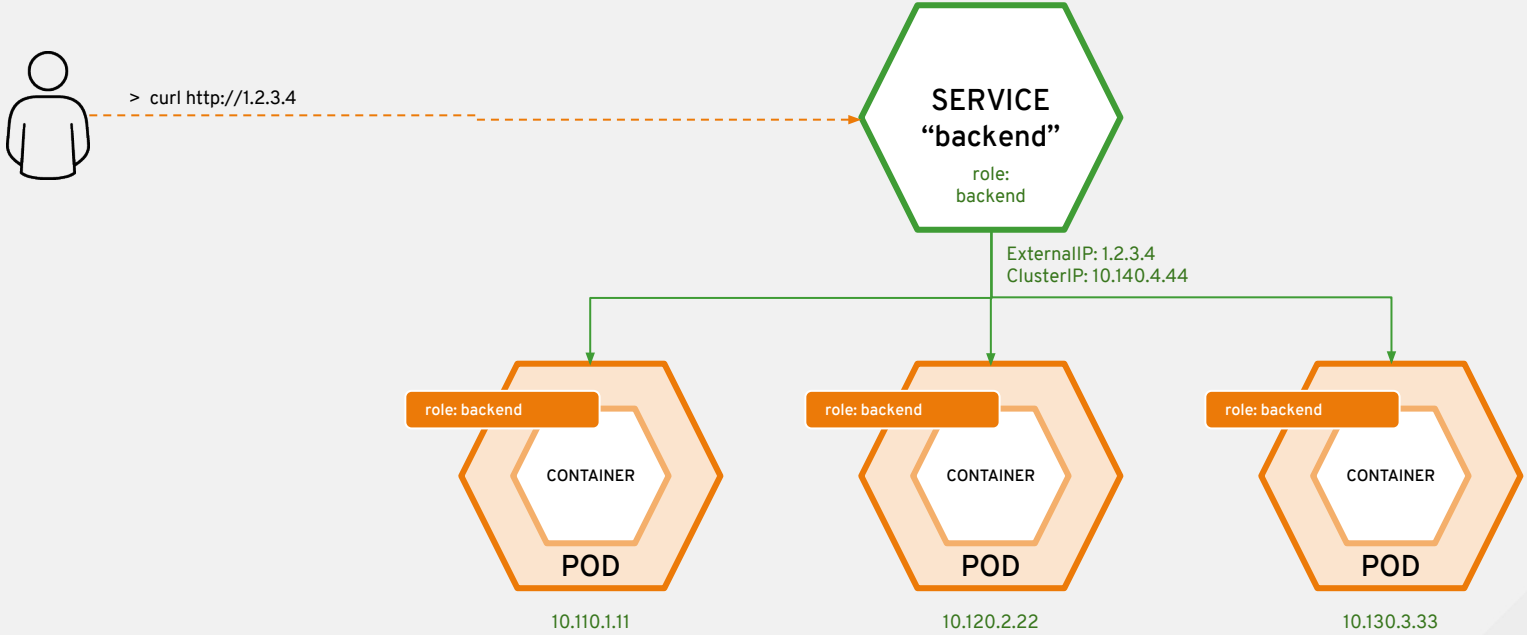
Service discovery for **External Services**

> oc rsh dc/frontend curl -kv https://backend

External Services

```
> oc create service externalname --external-name=devtools.belastingdienst.nl
bitbucket

> oc get svc
> oc describe svc/bitbucket
> oc get endpoints

> oc rsh dc/frontend curl -kv https://bitbucket/bitbucket
```

Publicly expose services

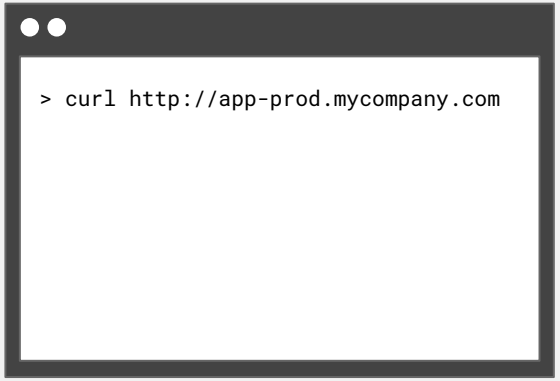


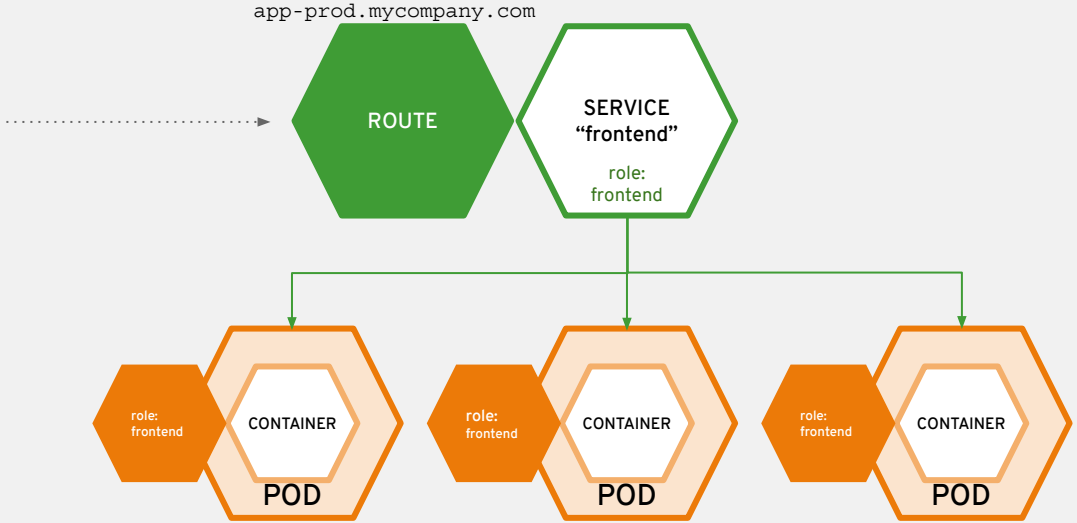
> oc rsh dc/frontend host backend-v1

Service.ExternalIP

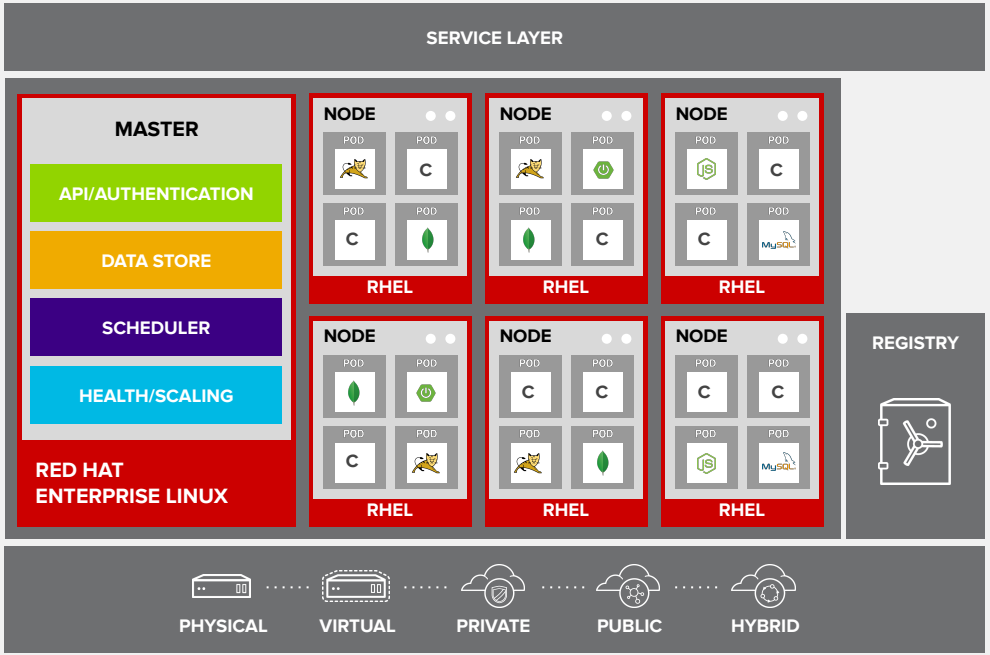
```
> oc edit svc frontend
spec:
  externalIPs:
  - 1.2.3.4

> oc get svc
> oc rsh dc/frontend host frontend
```

`routes` make services accessible to clients outside the environment via real-world urls





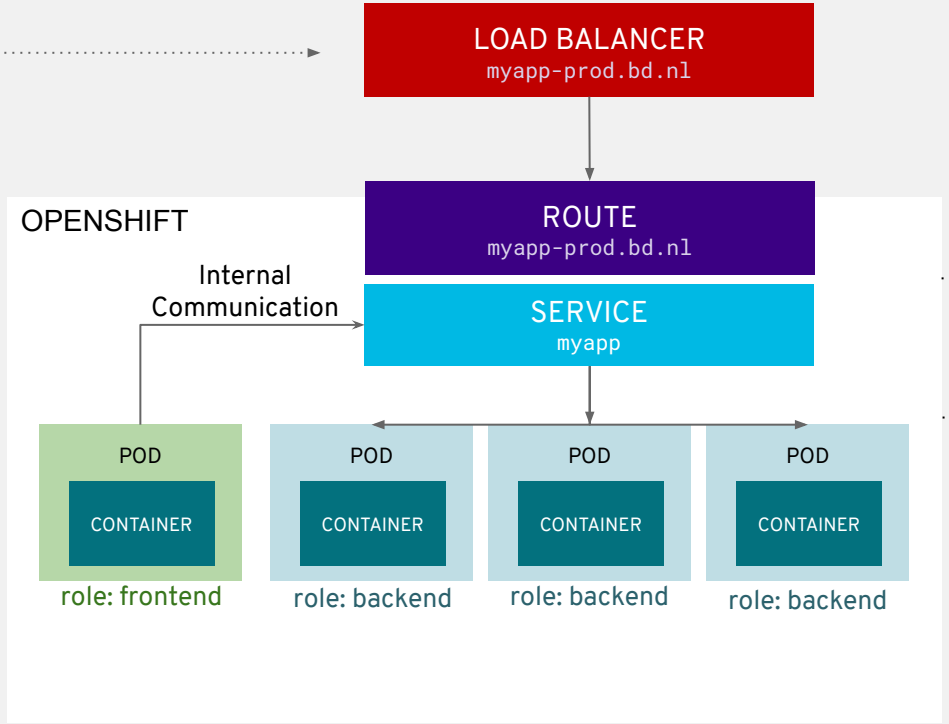
SERVICE DISCOVERY




```
> ssh master1 host jenkins-user00.svc.cluster.local
> oc rsh dc/jenkins env |grep SERVICE_PORT
```

ROUTING TRAFFIC

```
> curl http://app-prod.bd.nl
```



L7 Proxy
- SSL Offloading

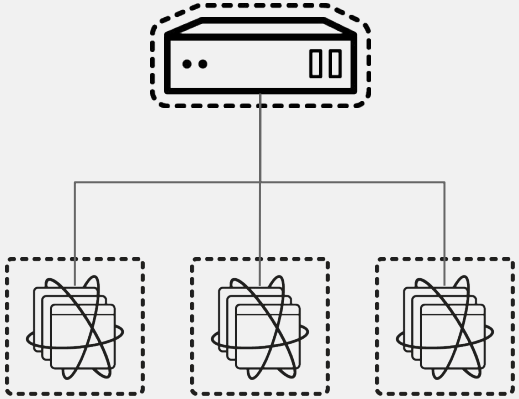
L4 Load Balancer
- Service Discovery

IPC over TCP

```
> oc expose svc/frontend --hostname opendoor.belastingdienst.nl
> oc create route edge --service=frontend --port=8080 --insecure-policy=Redirect --hostname=opendoor.belastingdienst.nl
> oc get route
```

ROUTING AND EXTERNAL LOAD-BALANCING

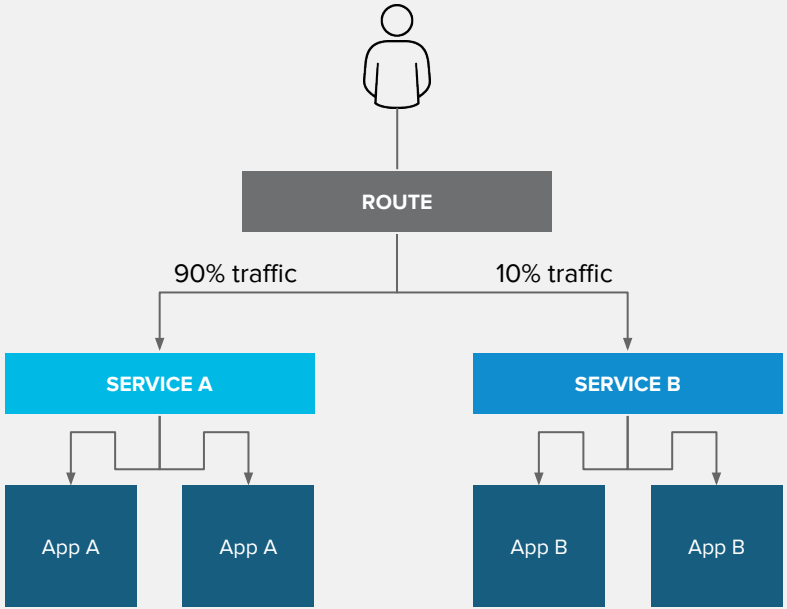
- Pluggable routing architecture
  - HAProxy Router
  - F5 Router
- Multiple-routers with traffic sharding
- Router supported protocols
  - HTTP/HTTPS
  - WebSockets
  - TLS with SNI
- Non-standard ports via cloud load-balancers, ExternalIP, and NodePort



```
> oc expose svc/frontend --hostname opendoor.belastingdienst.nl
> oc create route edge --service=frontend --port=8080 --insecure-policy=Redirect --hostname=opendoor.belastingdienst.nl
> oc get route
```

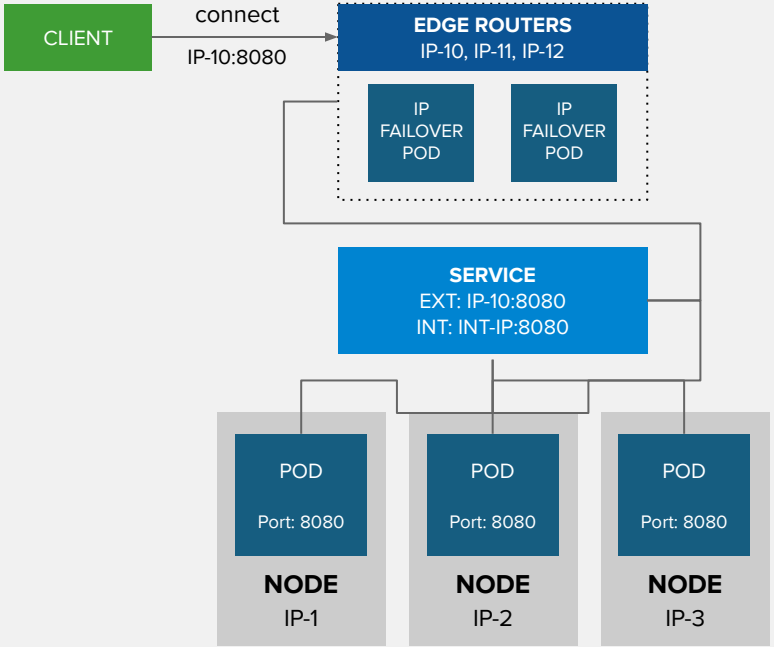
ROUTE SPLIT TRAFFIC

Split Traffic Between Multiple Services For A/B Testing, Blue/Green and Canary Deployments



EXTERNAL TRAFFIC WITH EXTERNALIP

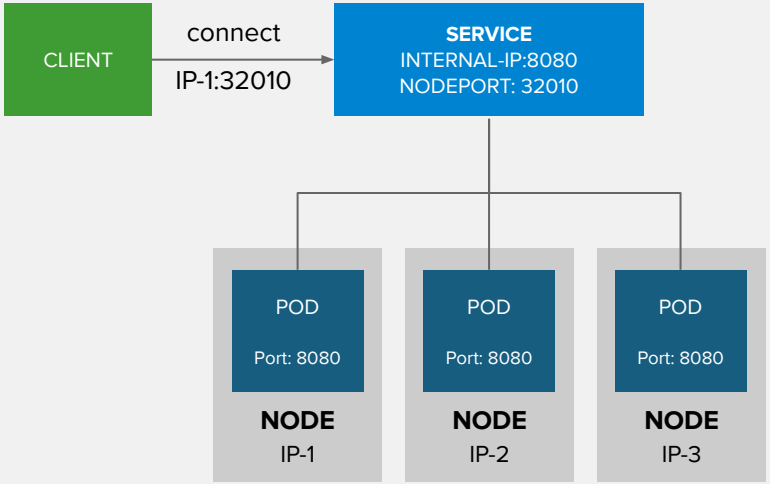
- Route external traffic to a service on any TCP/UDP port
- Available on non-cloud clusters
- External IP automatically assigned from a pre-defined pool of external IPs
- IP failover pods provide high availability for the pool of external IPs



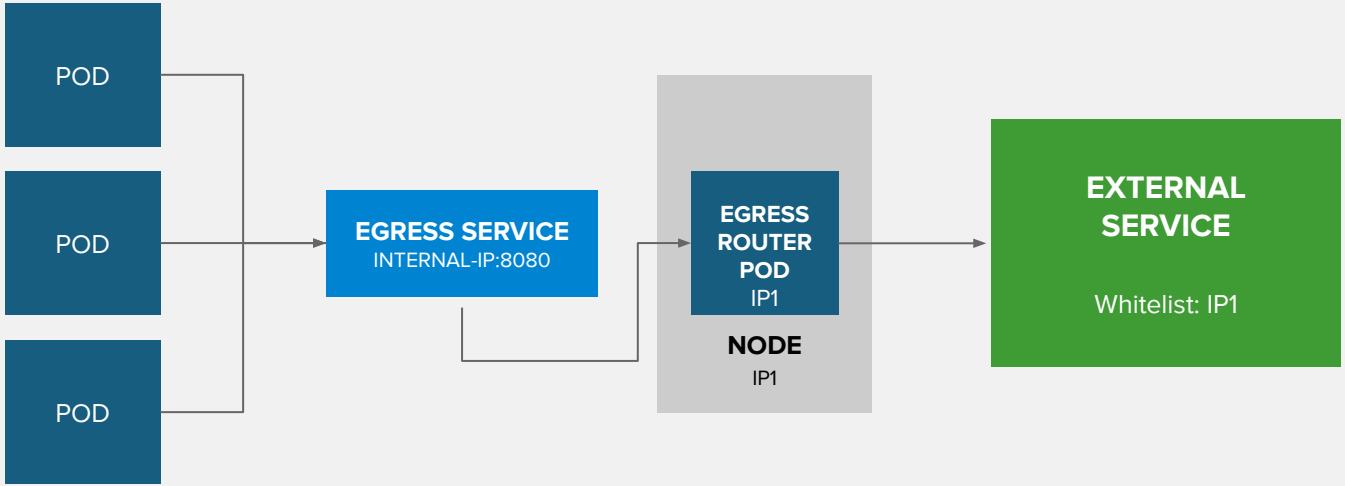

```
> oc expose dc/jenkins --external-ip=172.29.253.100 --name=jenkins-externalip
> oc get svc
```

EXTERNAL TRAFFIC WITH NODEPORT

- NodePort exposes a unique port on all the nodes in the cluster
- Ports in 30K-60K range which usually differs from the service
- Traffic received on any node redirects to a node with the running service
- Firewall rules must allow traffic to all nodes on the specific port



CONTROL SOURCE IP WITH EGRESS ROUTER



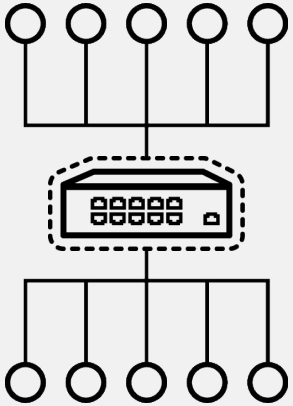
https://docs.openshift.com/container-platform/3.9/admin_guide/managing_networking.html

OpenShift Networking

Features, mechanisms and processes for container and platform isolation

OPENSHIFT NETWORKING

- Built-in internal DNS to reach services by name
- Split DNS is supported via SkyDNS
  - Master answers DNS queries for internal services
  - Other nameservers serve the rest of the queries
- Software Defined Networking (SDN) for a unified cluster network to enable pod-to-pod communication
- OpenShift follows Kubernetes network plug-in model
- Supported plug-ins
  - OpenShift SDN (Open vSwitch or Flannel)
  - Nuage SDN (Virtualized Services Platform)



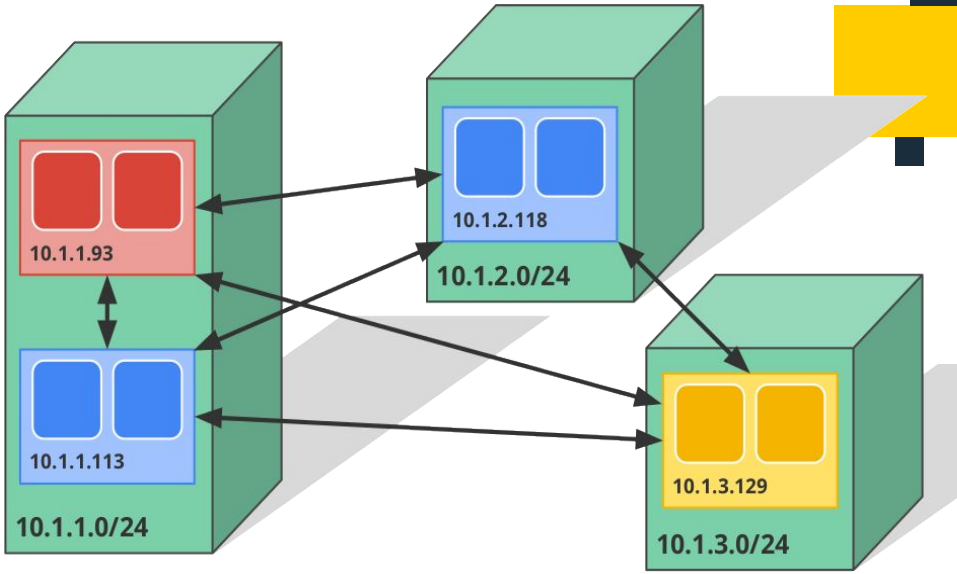
```
> ssh master1 grep networkPlugin /etc/origin/master/master-config.yaml
```

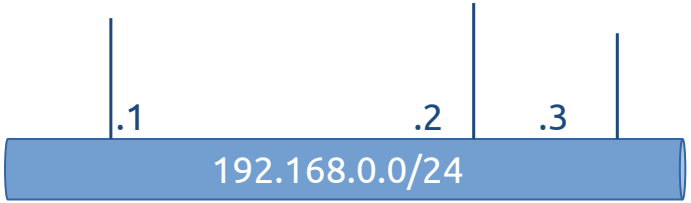
Overlay Networking

- Each Host = /24 IP-Range
- Each POD = 1 IP
  - Containers in Pod = localhost

- Egress traffic = NAT Node's IP

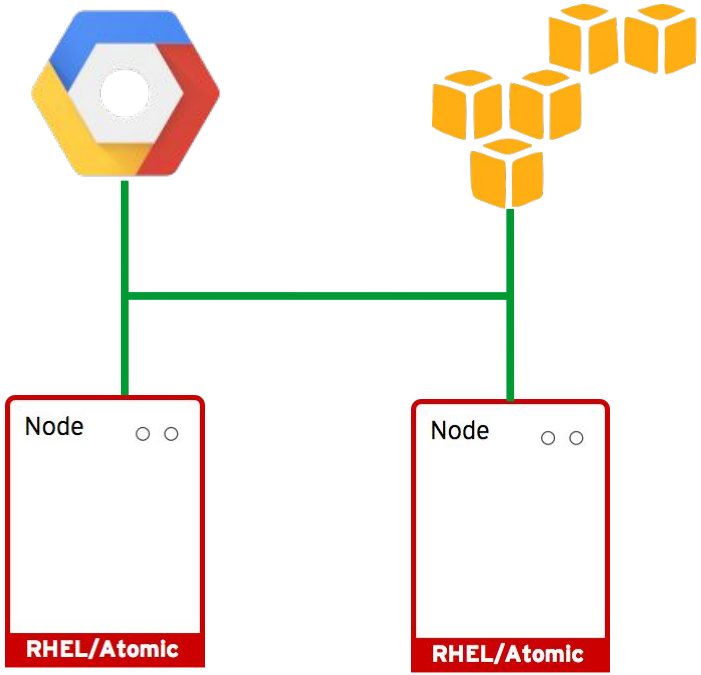
- Network Plugins:
  - Flannel, Weave, Nuage
  - OpenShift-SDN (OpenVSwitch)
  - Neutron, GCE, AWS



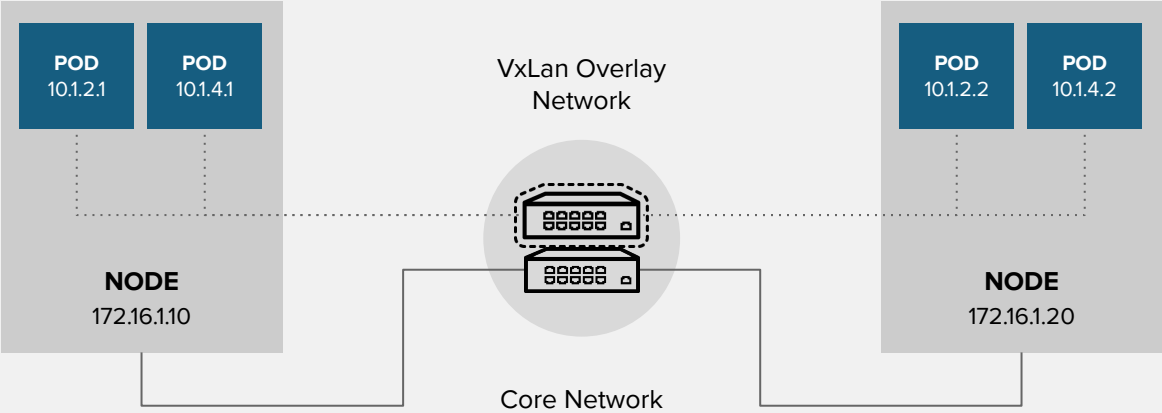


Underlay “infra” Network

- Programmable Infra
- Network Plugins (CNI):
    - GCE
    - AWS
    - OpenStack
    - Nuage
- Egress traffic = NAT Node IP



OPENSHIFT NETWORKING



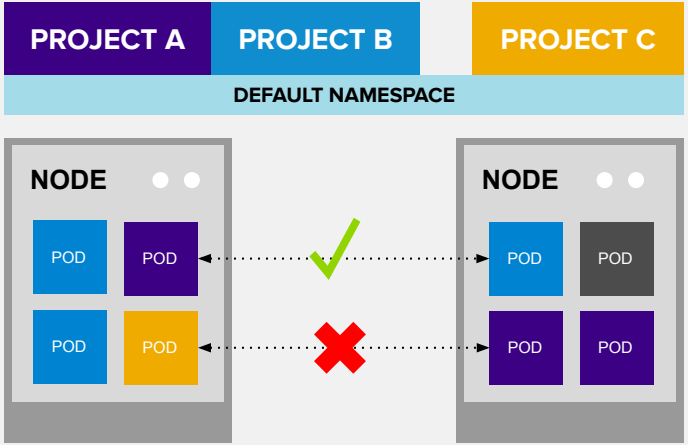
OPENSHIFT SDN

FLAT NETWORK

- All pods can communicate with each other across projects

MULTI-TENANT NETWORK

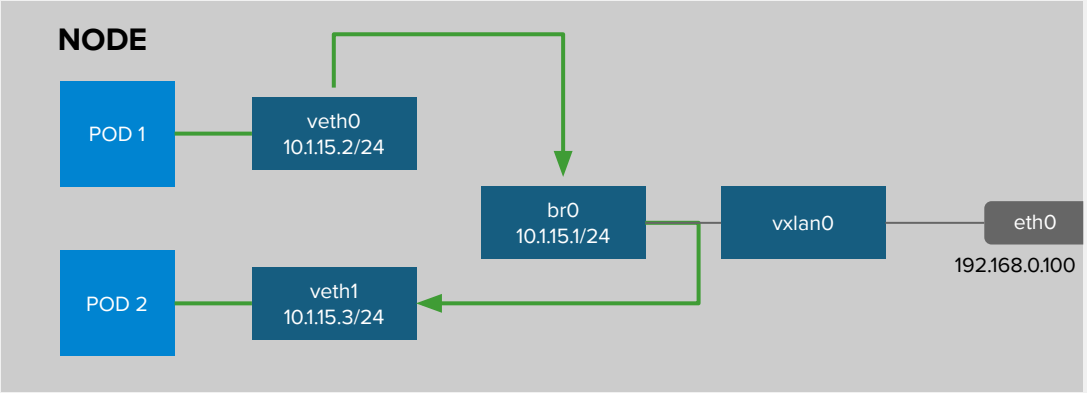
- Project-level network isolation
- Granular policies for network traffic
- Multicast support
- Egress network policies




```
> oc adm pod-network --to=user01 user00
> oc get netnamespaces |sort -k2 -n
```

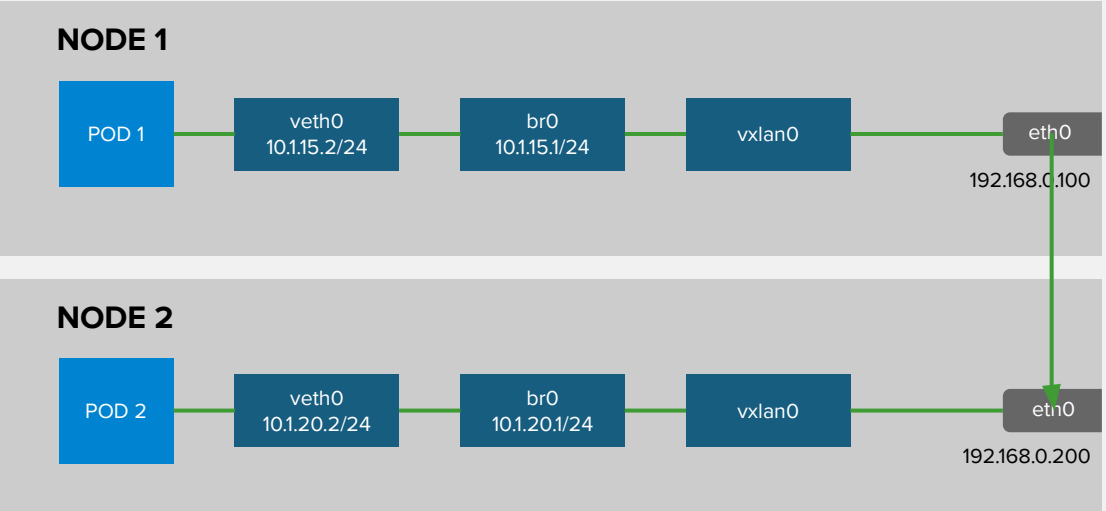
OPENSHIFT SDN - OVS PACKET FLOW

Container to Container on the Same Host



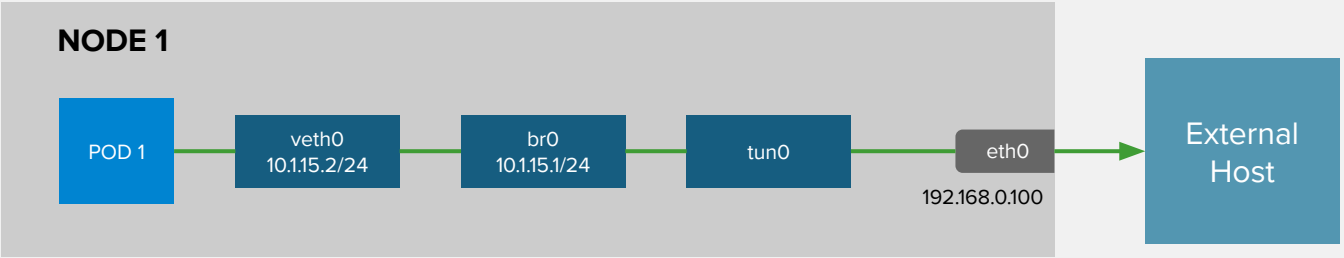
OPENSHIFT SDN - OVS PACKET FLOW

Container to Container on the Different Hosts



OPENSHIFT SDN - OVS PACKET FLOW

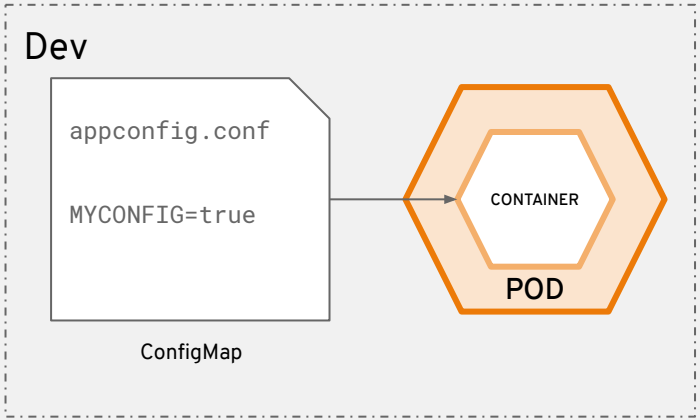
Container Connects to External Host

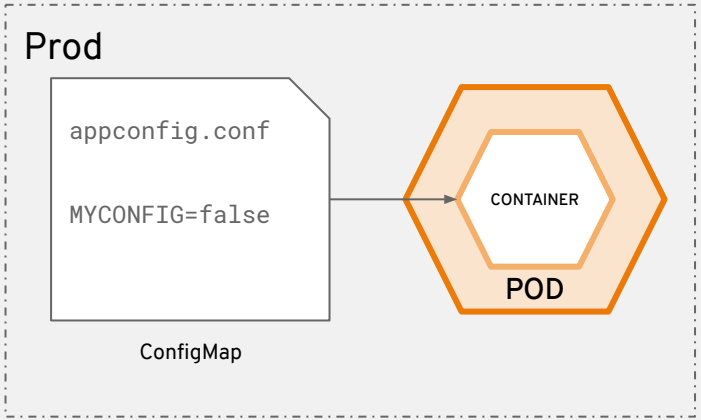


Persistent Storage

Connecting real-world storage to your containers to enable stateful applications

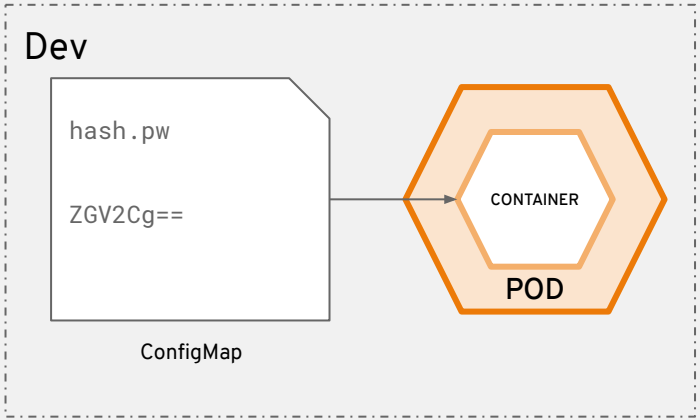
`configmaps` allow you to decouple configuration artifacts from image content

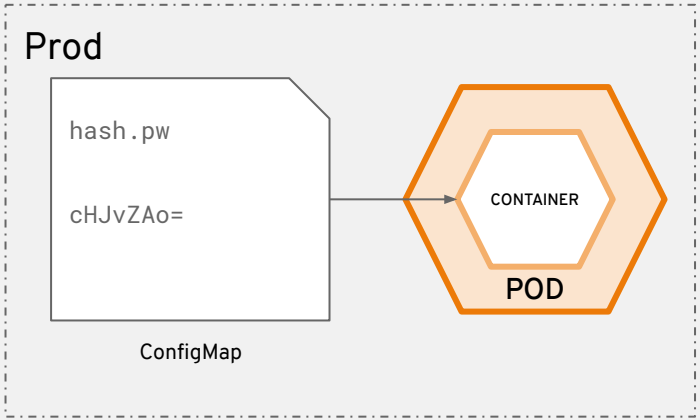





```
> oc create config db-settings --from-literal=username=password --from-file=htpasswd
> oc set volumes --add dc/backend -m /etc/mysql --type=configmap --configmap-name=db-settings
```

`secrets` provide a mechanism to hold sensitive information such as passwords






```
> oc create secret generic passwords --from-literal=username=password --from-file=htpasswd
> oc set env dc/backend --from=secret/passwords
```

Secrets & ConfigMaps

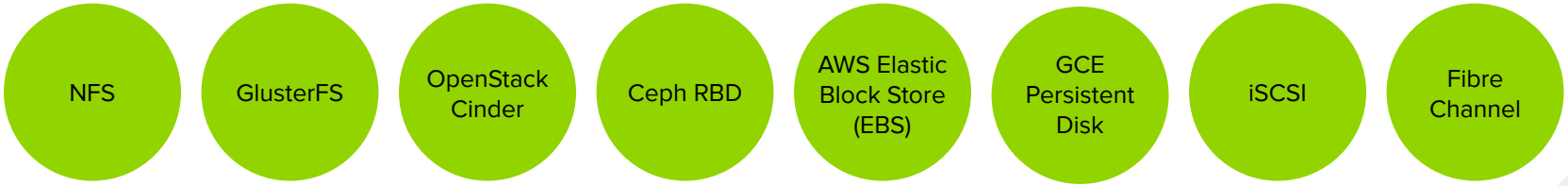
```
> oc -n openshift get imagestreams |grep ^mysql
> oc -n openshift get templates    |grep ^mysql

> oc new-app mysql-ephemeral
> oc describe secret/mysql

> oc set env dc/backend --from=secret/passwords
> oc get -o yaml dc/backend

> oc create config db-settings --from-literal=username=password
--from-file=htpasswd
> oc edit cm/db-settings

> oc set volumes --add dc/backend -m /etc/mysql --type=configmap
--configmap-name=db-settings
```

A broad spectrum of static and dynamic storage endpoints

NFS	OpenStack Cinder	iSCSI	Azure Disk	AWS EBS	FlexVolume
GlusterFS	Ceph RBD	Fiber Channel	Azure File	GCE Persistent Disk	VMWare vSphere VMDK
		NetApp Trident*	Container Storage Interface (CSI)**		

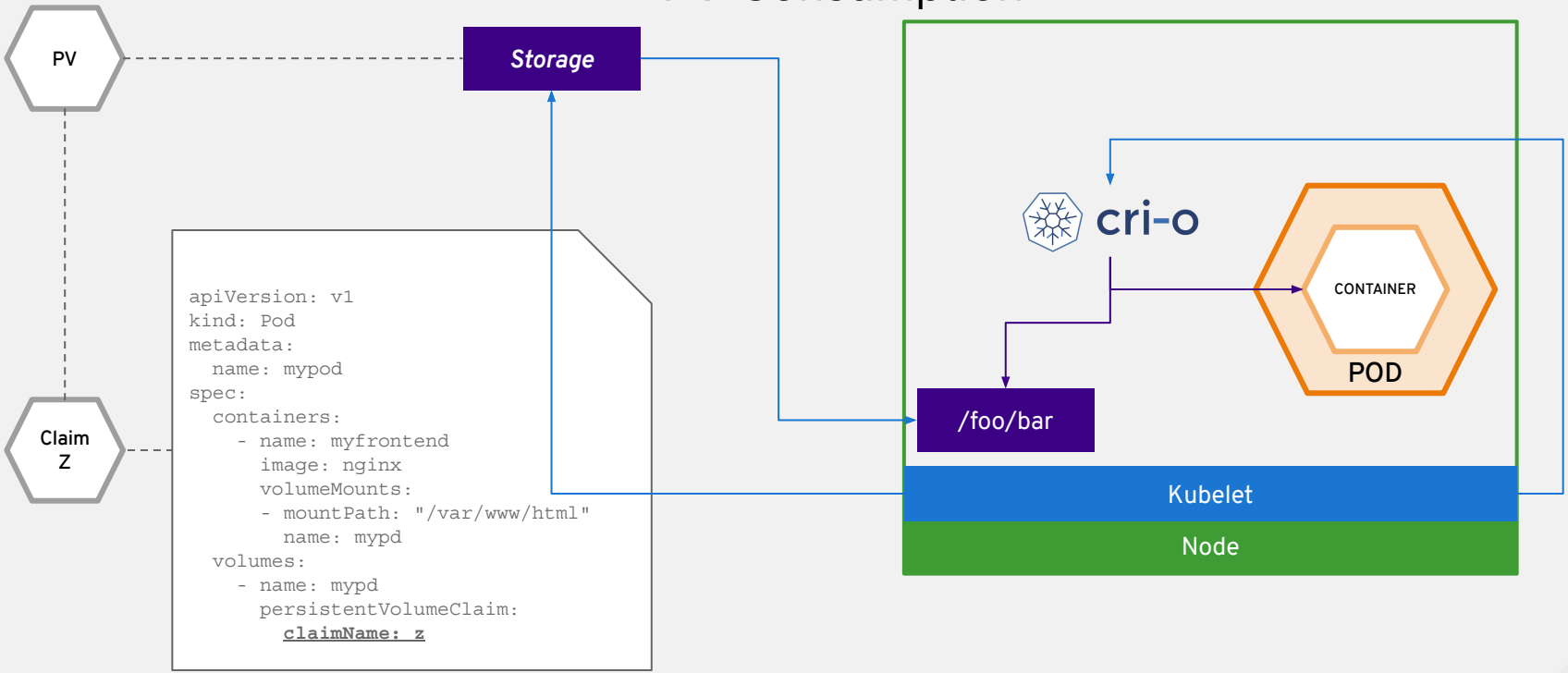
PERSISTENT STORAGE

- Persistent Volume
  - Tied to a piece of network storage
  - Provisioned by an administrator (static or dynamically)
  - Allows admins to describe storage and users to request storage

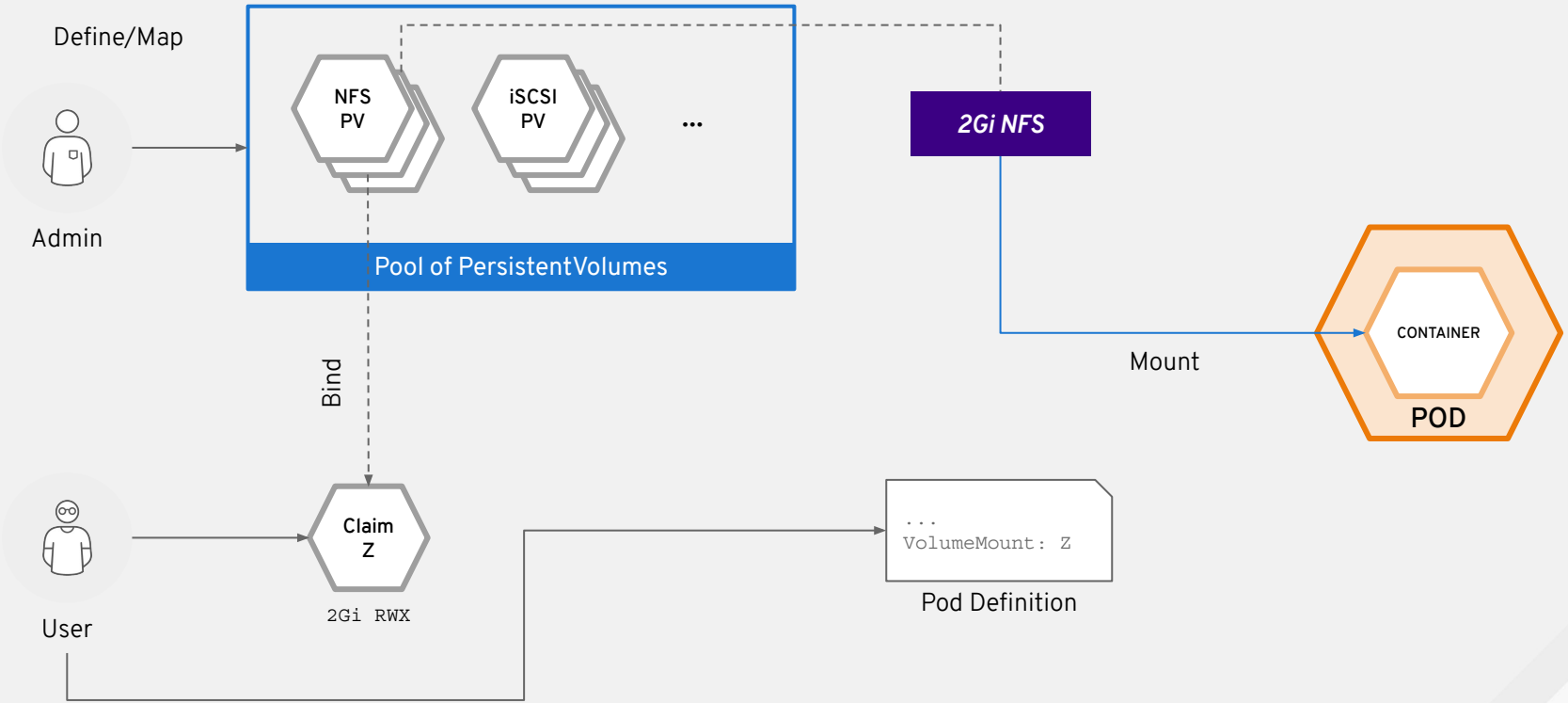



```
> oc get storageclass
```

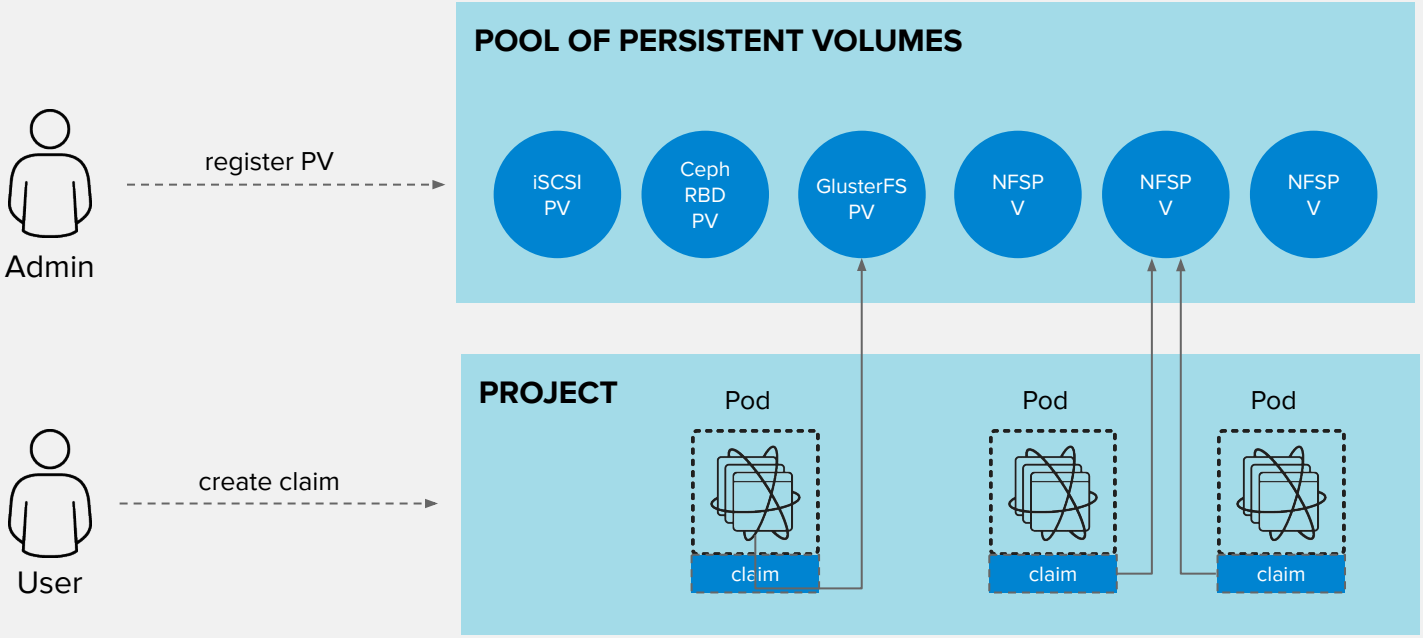
PV Consumption



Static Storage Provisioning

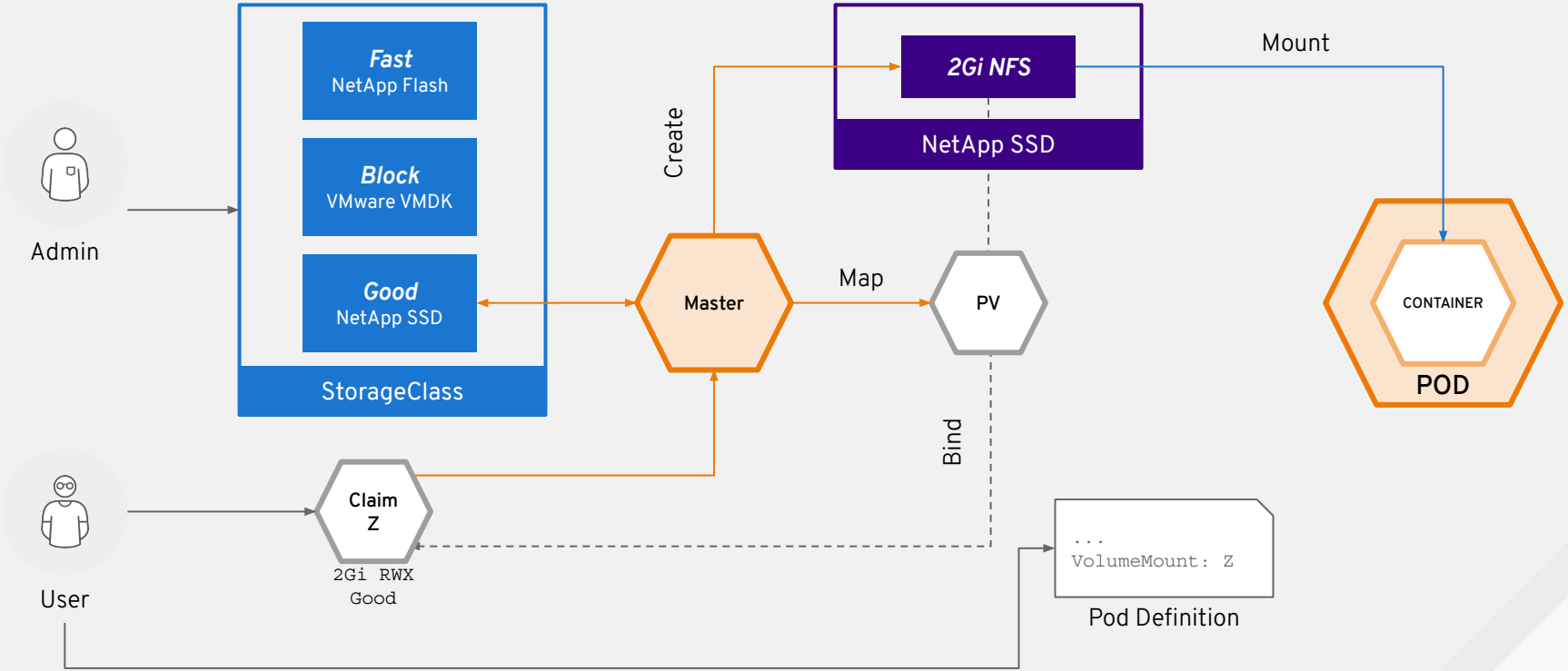


PERSISTENT STORAGE



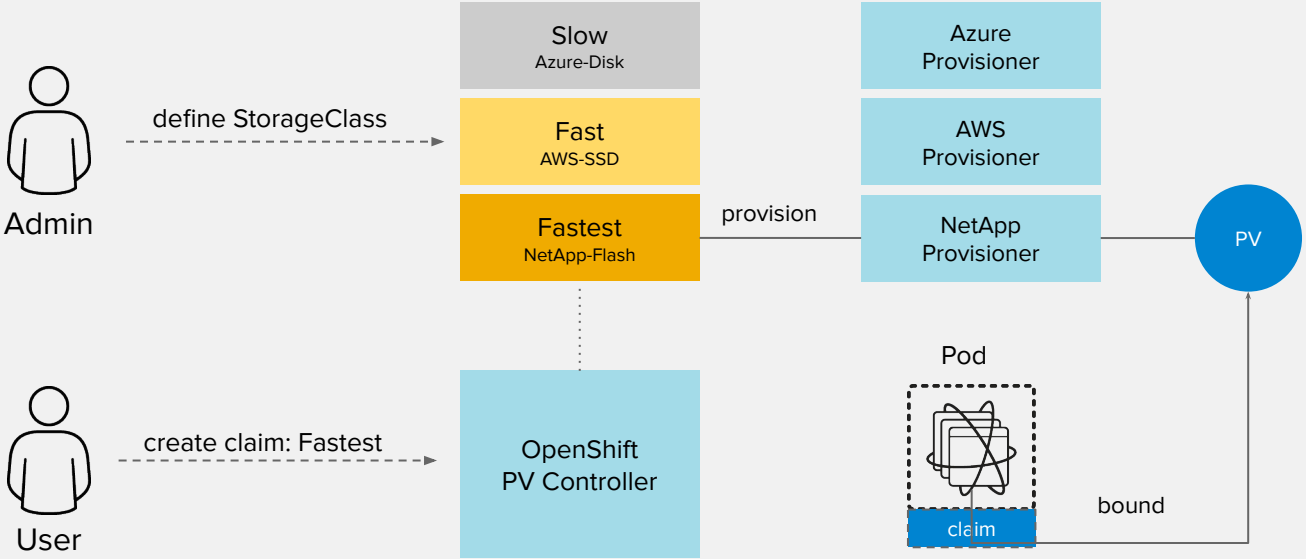

```
> oc get pv
> oc get pvc --all-namespaces
```

Dynamic Storage Provisioning



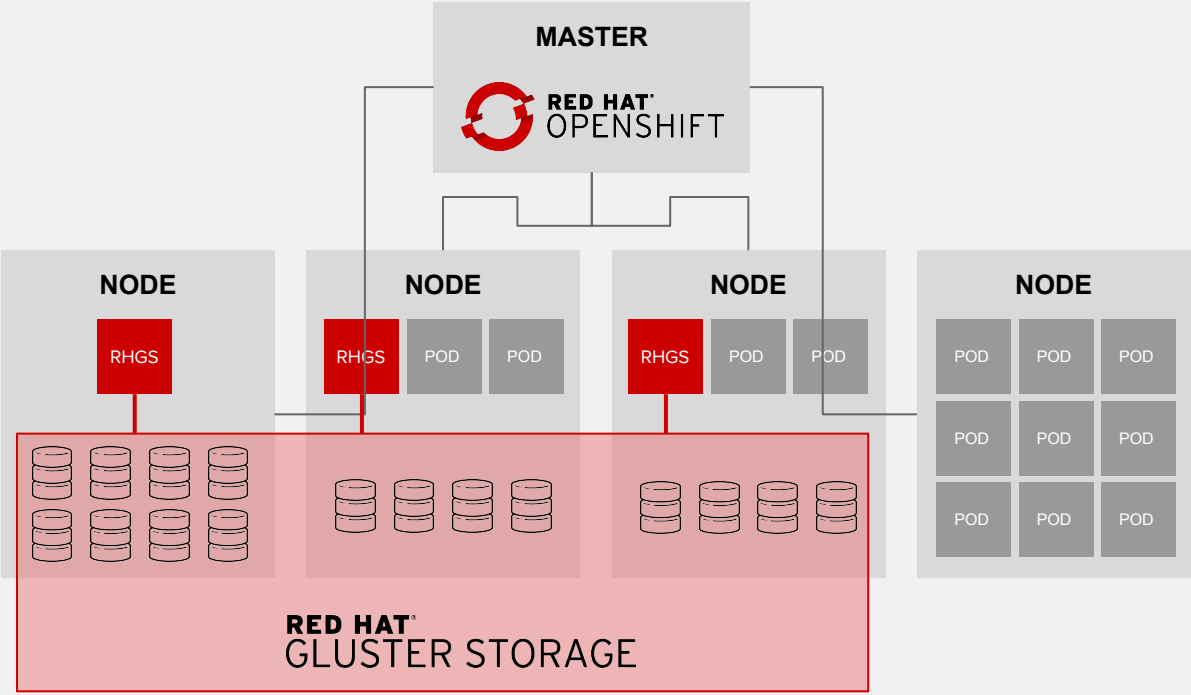

```
> oc set volume --add dc/mysql --name=mysql-data --type=pvc --claim-name=mysql-data --claim-size=1G
```

DYNAMIC VOLUME PROVISIONING




```
> oc get storageclass
> oc -n glusterfs get all
```

CONTAINER-NATIVE STORAGE



> oc -n glusterfs get pods -l glusterfs=storage-pod
> oc -n glusterfs rsh daemonset/glusterfs-storage ps auwx
> oc -n glusterfs volume ds/glusterfs-storage

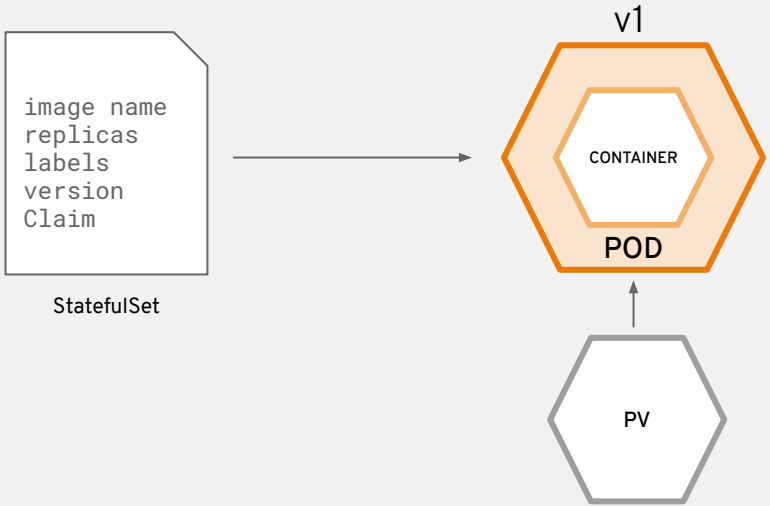
Lab #7 – Storage

```
oc project glusterfs
oc rsh dc/heketi-storage
heketi-cli --user admin --secret $HEKETI_ADMIN_KEY topology info
heketi-cli --user admin --secret $HEKETI_ADMIN_KEY help

oc export storageclass glusterfs-storage
```

Misc Resources

An integrated solution for exploring and corroborating application logs

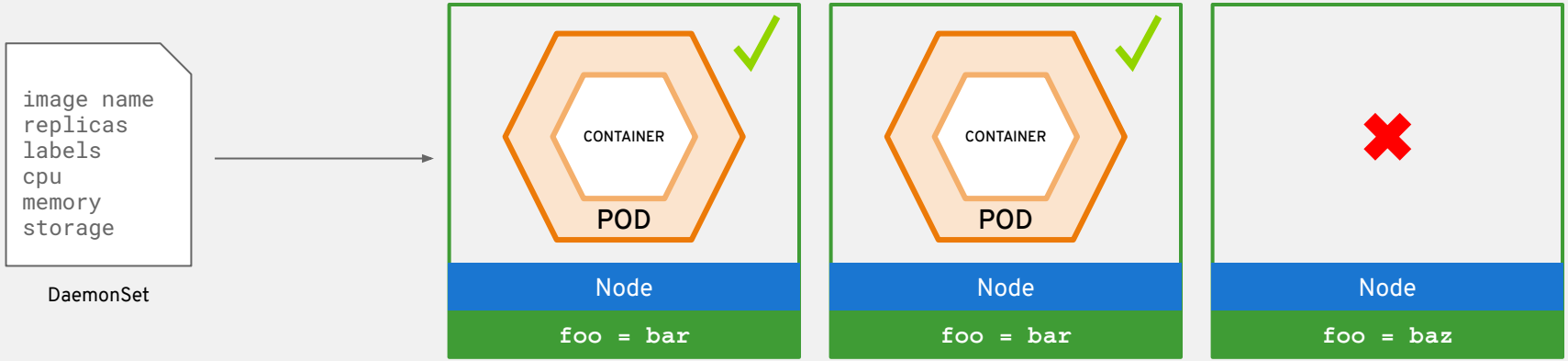
`StatefulSets` are special deployments for Stateful workloads




```yaml
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: httpd-persistent
spec:
  serviceName: "backend"
  replicas: 2
  selector:
    matchLabels:
      app: httpd-persistent
  template:
    metadata:
      labels:
        app: httpd-persistent
    spec:
      containers:
      - name: httpd
        image: httpd:latest
        volumeMounts:
        - name: www
          mountPath: /var/www/html
  volumeClaimTemplates:
  - metadata:
      name: www
    spec:
      accessModes: [ "ReadWriteOnce" ]
      resources:
        requests:
          storage: 1Gi
```

> oc create -f statefulset.yaml

a `daemonset` ensures that all (or some) nodes run a copy of a pod

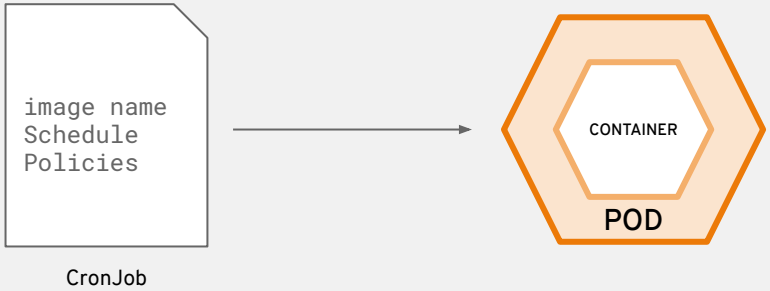


> oc get -o yaml --export dc/backend |sed 's/DeploymentConfig/DaemonSet/g' |oc create -f -

DaemonSet

```
> oc get -o yaml --export dc/backend >daemonset.yml
> vi daemonset.yml
apiVersion: apps/v1
Kind: DaemonSet
```

`CronJobs` runs a short lived container at a set time



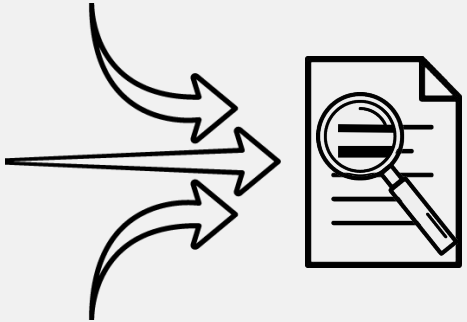

```
> oc create cronjob myjob --image=httpd --schedule="*/10 * * * *" --restart=OnFailure
> oc patch cronjob/myjob -p '{"spec": {"jobTemplate": {"spec": {"template": {"spec": {"containers": [ {"name": "myjob", "command":
["/bin/bash","-c","echo hello world"] } ] } } } } } }'
```

OpenShift Logging

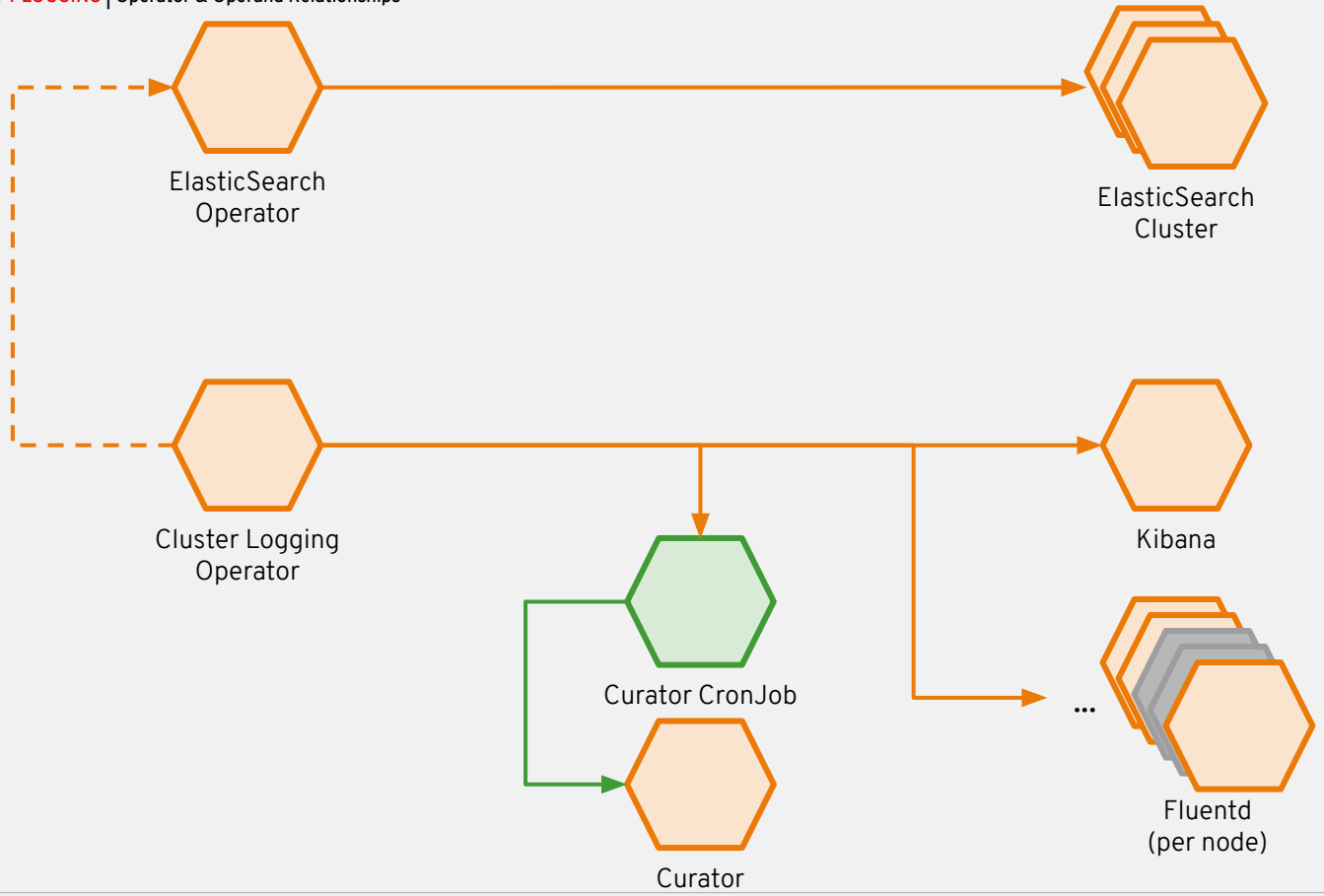
An integrated solution for exploring and corroborating application logs

CENTRAL LOG MANAGEMENT WITH EFK

- EFK stack to aggregate logs for hosts and applications
  - **Elasticsearch:** an object store to store all logs
  - **Fluentd:** gathers logs and sends to Elasticsearch.
  - **Kibana:** A web UI for Elasticsearch.
- Access control
  - Cluster administrators can view all logs
  - Users can only view logs for their projects
- Ability to send logs elsewhere
  - External elasticsearch, Splunk, etc

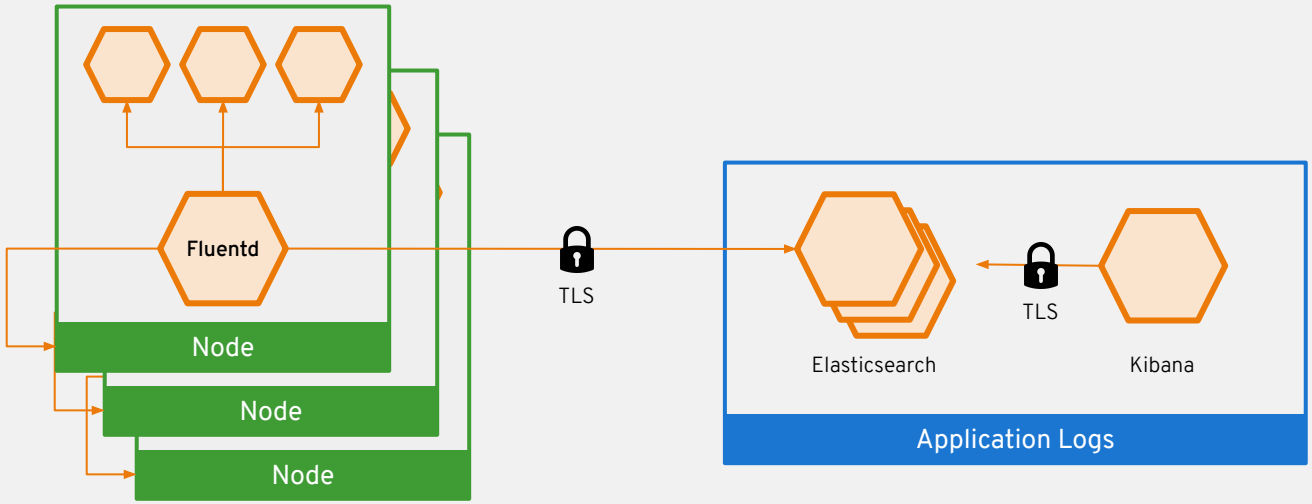


> oc -n logging get all
> http://kibana.openshift.eu

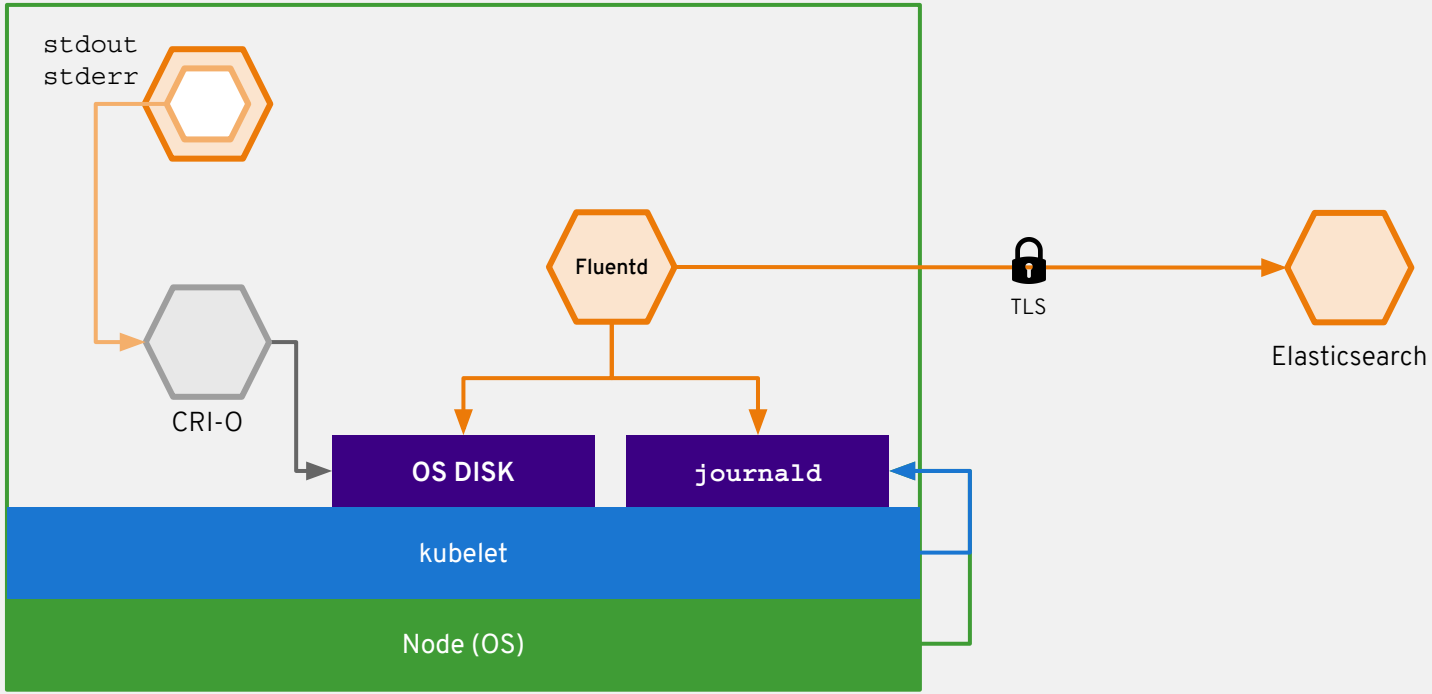


> oc -n logging get all
> http://kibana.openshift.eu

Log data flow in OpenShift



Log data flow in OpenShift



OpenShift Monitoring

An integrated cluster monitoring and alerting stack

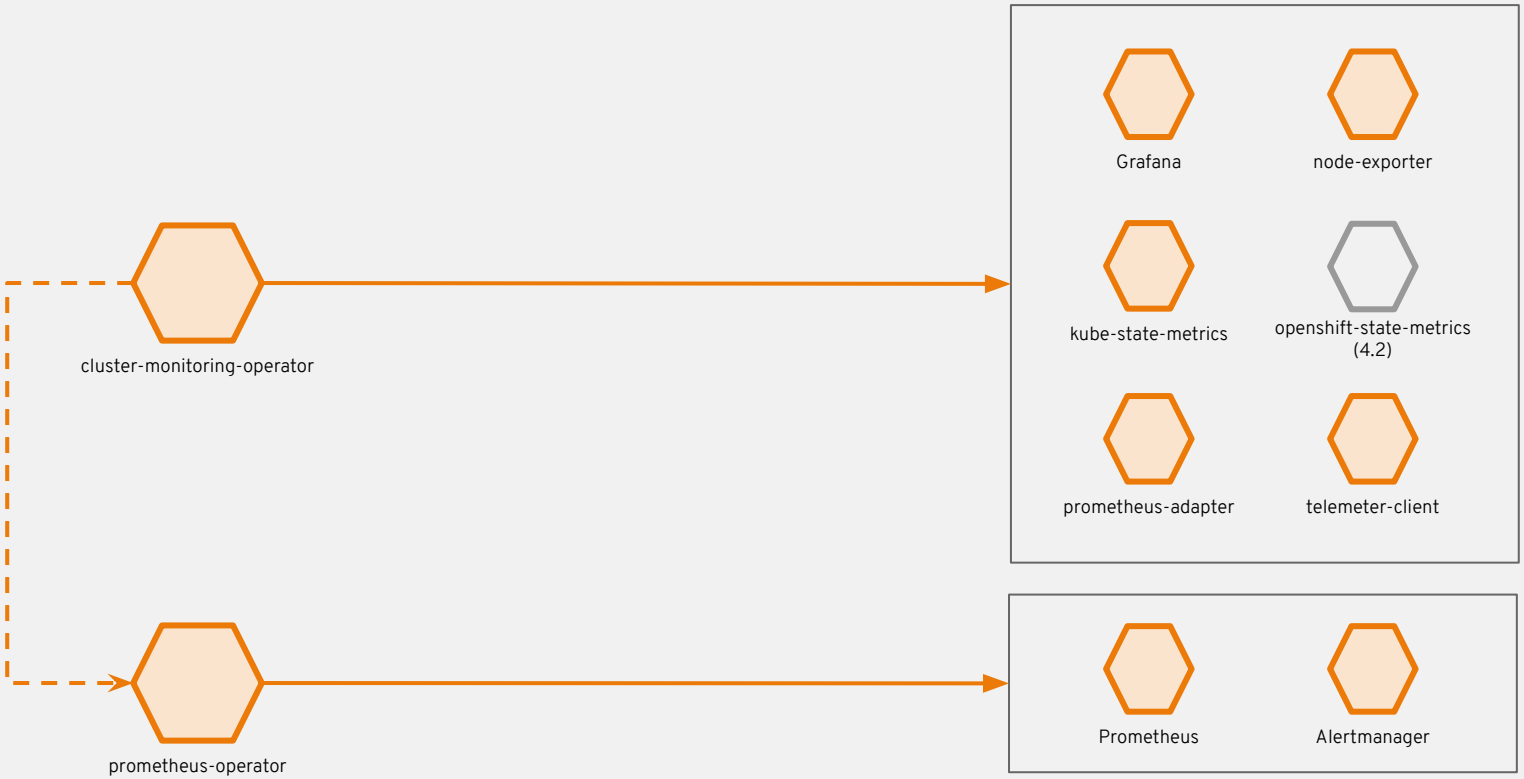
OpenShift Cluster Monitoring

Metrics collection and storage via Prometheus, an open-source monitoring system time series database.

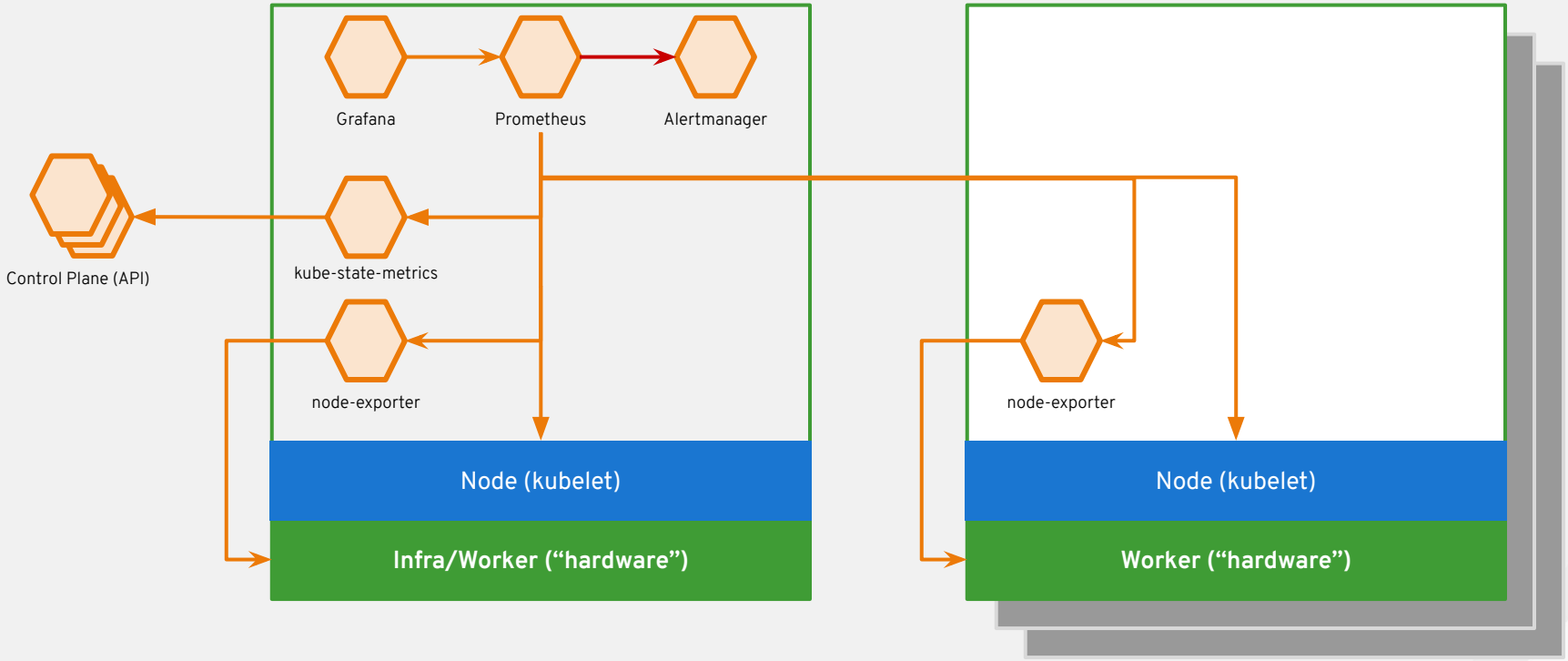
Alerting/notification via Prometheus' Alertmanager, an open-source tool that handles alerts send by Prometheus.

Metrics visualization via Grafana, the leading metrics visualization technology.

> oc -n openshift-infra get all
> https://hawkular-metrics.openshift.eu



> oc -n openshift-infra get all
> https://hawkular-metrics.openshift.eu



> oc -n openshift-infra get all
> https://hawkular-metrics.openshift.eu

End of Day #2

Go Home!

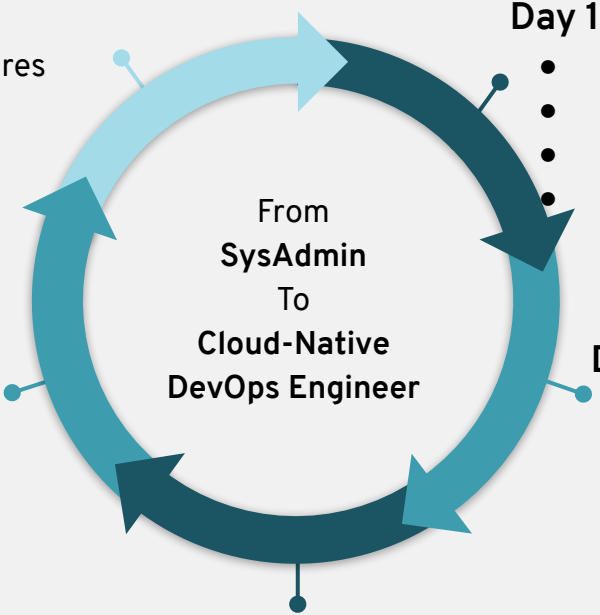
Agenda

Day 5 - Best Practices

- Reference Architectures
- Use Cases
- App OnBoarding

Day 1 - Overview

- Cloud-Native market
- Container Concepts
- Platform Concepts
- Platform Architecture
- OpenShift - Technical Deep Dive



Day 4 - Misc

- RBAC, IAM
-

Day 2 - App Deployment

- Technical Deep Dive
- Labs
- Compute: Pods
- Networking: Services, Routes
- Storage: Persistent Volumes

Day 3 - App Build

- Labs
- Source-2-Image
- Jenkins