OpenShift 4

Making introductions

Samuel Terburg
Red Hat Certified Architect
Cloud-Native Expert

2020-01-14

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

Do’s & Don’ts

Do’s

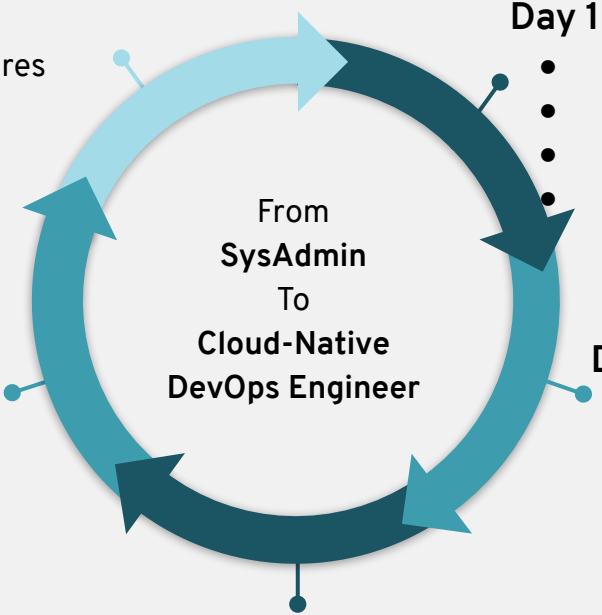
Manage time
09:00 Start
12:00 - 13:30 Lunch
16:00 Go Home
Ask Questions
Fantasize (real-world cases)
Have Fun (share)

Don’t

Check Emails / Phone
?

Practical Info

Agenda



Day 1 - Overview

- Cloud-Native market
- Container Concepts
- Platform Concepts
- Platform Architecture

OpenShift - Technical Deep Dive

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

Day 4 - Misc

- RBAC, IAM
-

Day 5 - Best Practices

- Reference Architectures
- Use Cases
- App OnBoarding

Introductions

Who is who

Fastlane

Wie zeg je?!

Samuel Terburg

Wie zeg je?!

Belastingdienst

Wie zeg je?!

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

The OpenShift Client

- Using the `oc get` command, inspect various type of objects

```
$ oc get nodes
$ oc get services
$ oc get dc            # deployment controller
$ oc get pods
```

- Watch events

```
$ oc get events -w
```

- View the logs

```
$ oc logs pod/simple-openshift-sinatra-sti-1-abcde
```

- Learn about commands

```
$ oc --help
$ oc options
$ oc help get
$ oc get --help
$ oc explain pod.spec
```

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}

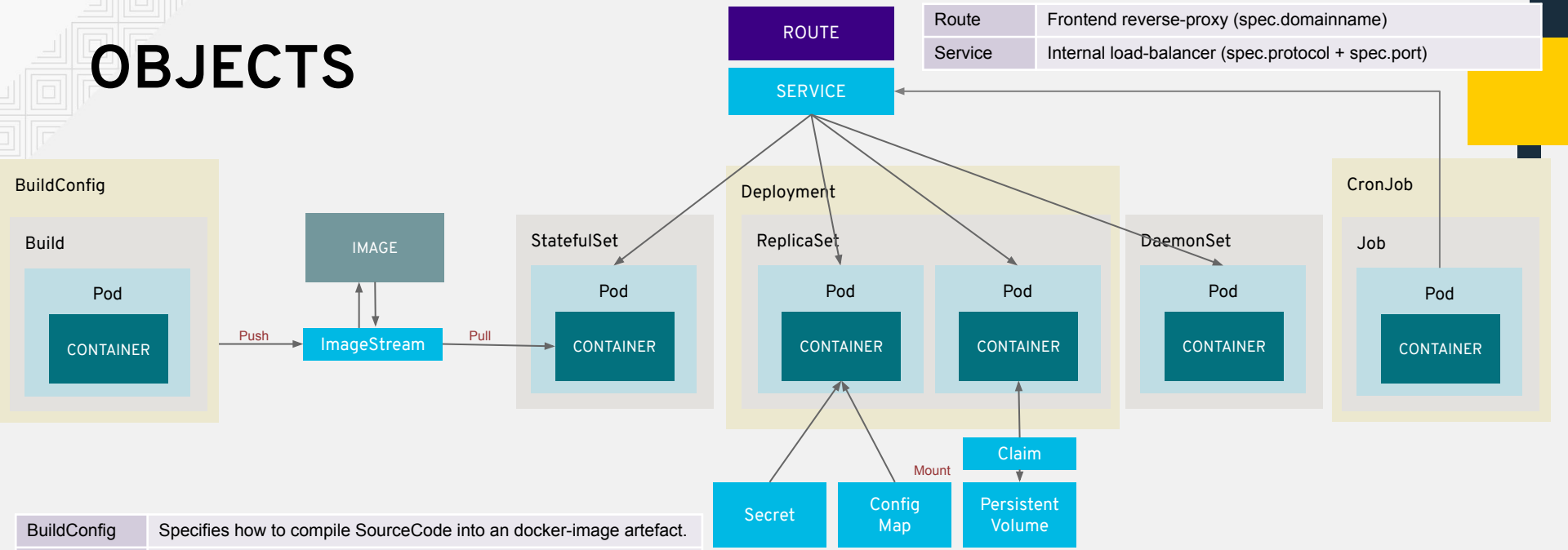
oc get pod myapp-1-abcde

{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.

Cloud-Native

Background on why

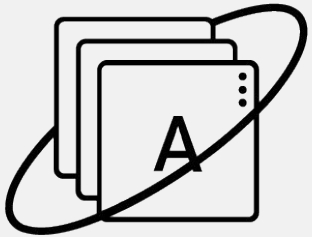
OpenShift exists.

“Developers are increasingly moving to cloud-native platforms.”

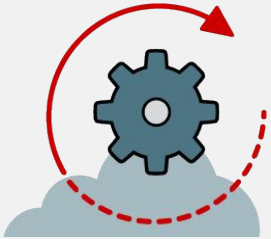
DAVE BARTOLETTI, FORRESTER RESEARCH

DRIVERS FOR MANY COMPANIES

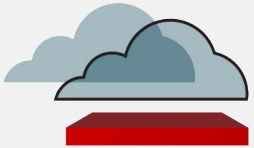
INNOVATE



CLOUD-NATIVE
Next Gen
Application Design



CI/CD
AUTOMATION
"Deliver Faster"
“Build Once, Deploy Many”



SELF SERVICE
Abstracts away Complexity
Enabling DevOps

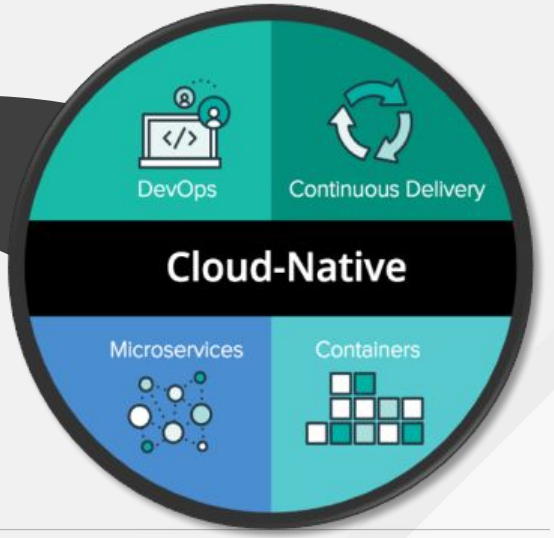
Design → Build & Deploy → Operate

THE ROAD TO “CLOUD”

Cloud Ready

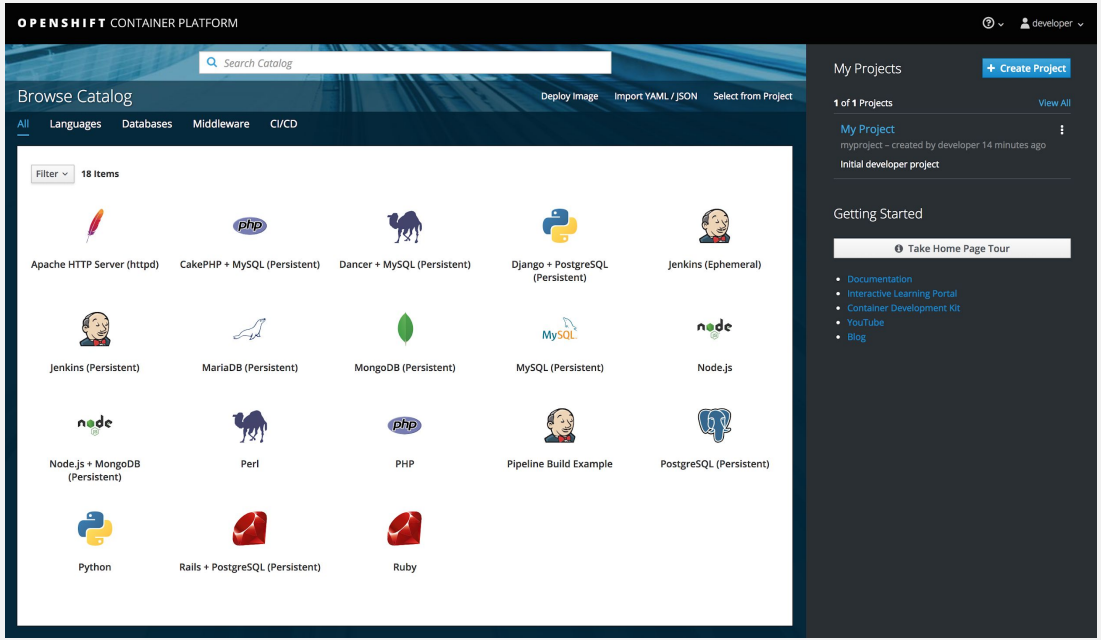
Cloud Friendly

Cloud Resilient



SOLUTION

OPENSHIFT CONTAINER PLATFORM



- A platform for building & hosting cloud-native applications
- Provides tools + workflows to increases developer productivity
- Leverages “Container” technology
- Uses Kubernetes (K8s) as the orchestrator

CNCF Cloud Native Landscape
2019-07-12T03:51:02Z 024e93b

Overwhelmed? Please see the CNCF Trail Map. That and the interactive landscape are at l.cncf.io

Greyed logos are not open source

App Definition and Development

Database

Streaming & Messaging

Application Definition & Image Build

Continuous Integration & Delivery

Orchestration & Management

Scheduling & Orchestration

Coordination & Service Discovery

Remote Procedure Call

Service Proxy

API Gateway

Service Mesh

Runtime

Cloud-Native Storage

Container Runtime

Cloud-Native Network

Provisioning

Automation & Configuration

Container Registry

Security & Compliance

Key Management

Cloud

Public

Special

Kubernetes Certified Service Provider

Platform

Certified Kubernetes - Distribution

Certified Kubernetes - Hosted

Certified Kubernetes - Installer

PaaS/Container Service

Kubernetes Training Partner

Observability and Analysis

Monitoring

Logging

Tracing

Chaos Engineering

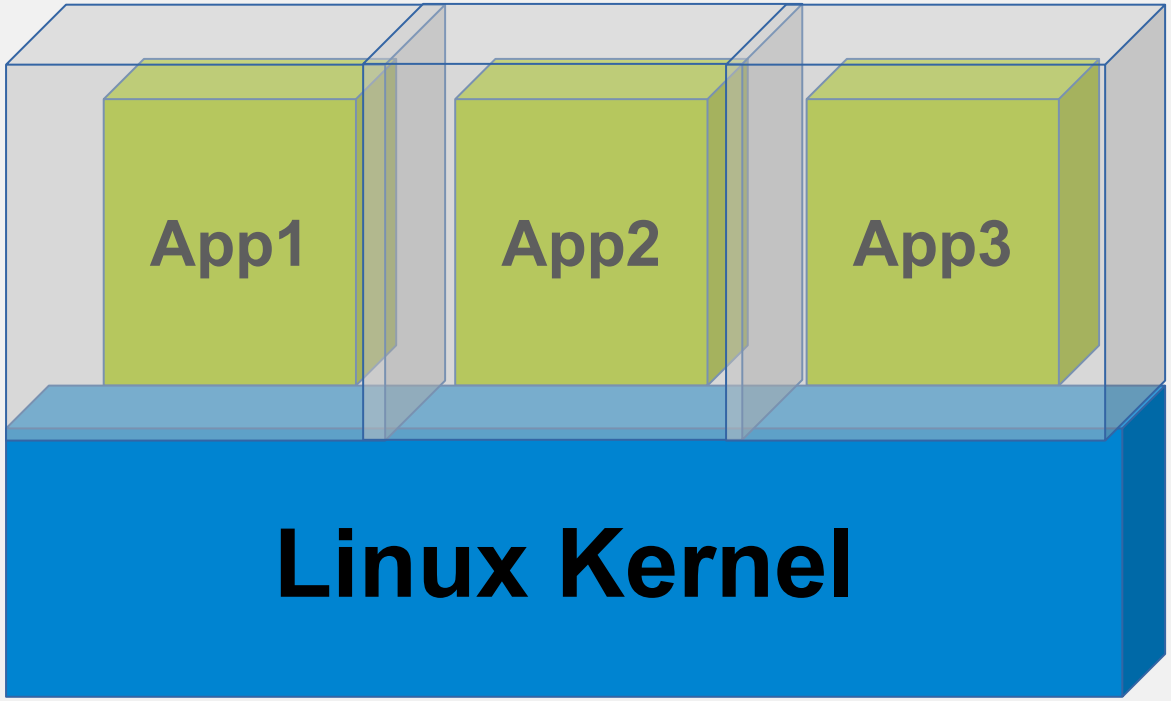
Serverless

Containers

The engine behind

your app

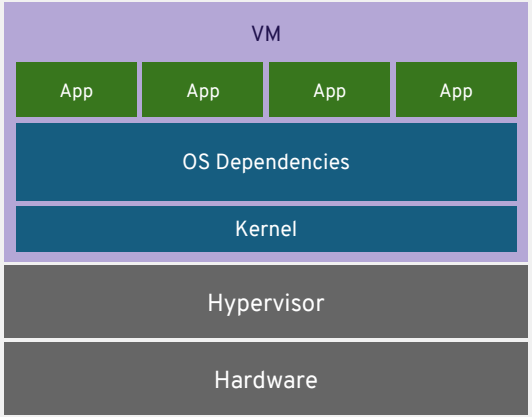
Isolation, not Virtualization



- Namespaces
  - Process
  - Network
  - IPC
  - Mount
  - User
- Resource Limits
  - Cgroups
- Security
  - SELinux

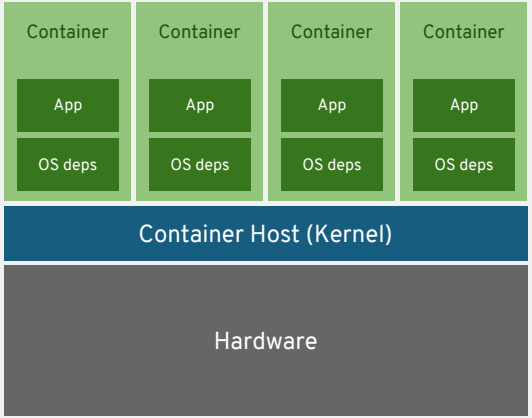
VIRTUAL MACHINES AND CONTAINERS

VIRTUAL MACHINES



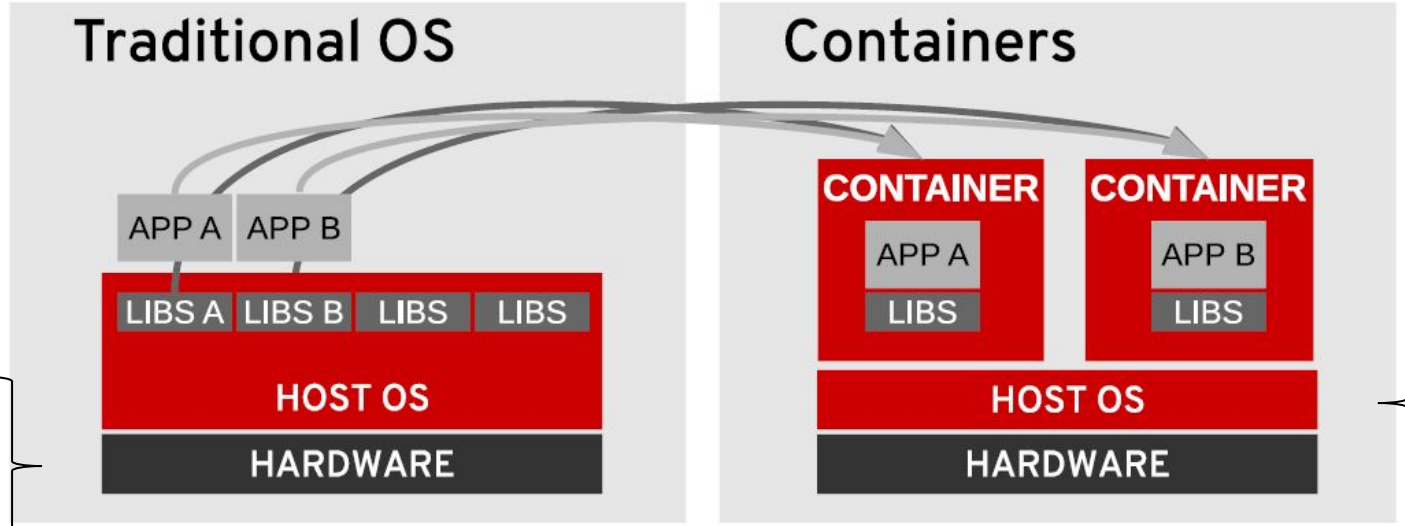
virtual machines are isolated
apps are not

CONTAINERS



containers are isolated
so are the apps

VIRTUAL MACHINE vs CONTAINERS



VMWARE

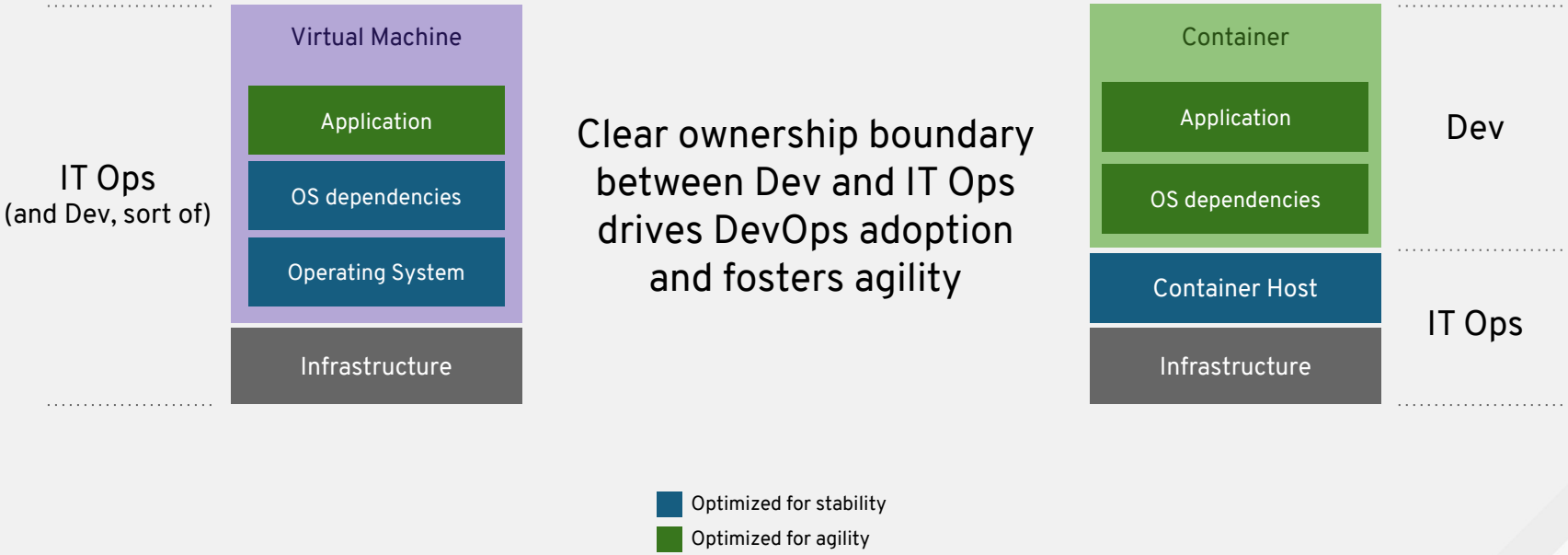
- ✚ VM Isolation
- ▬ Shared OS (libs)
- ▬ Shared Compute
- ▬ Shared Memory
- ▬ High Resource Overhead

NAMESPACES
CGROUPS
SELINUX

- ✚ Container Isolation
- ▬ Shared Kernel
- ✚ Burstable Compute
- ✚ Constrained Memory
- ✚ Verified Immutable Images

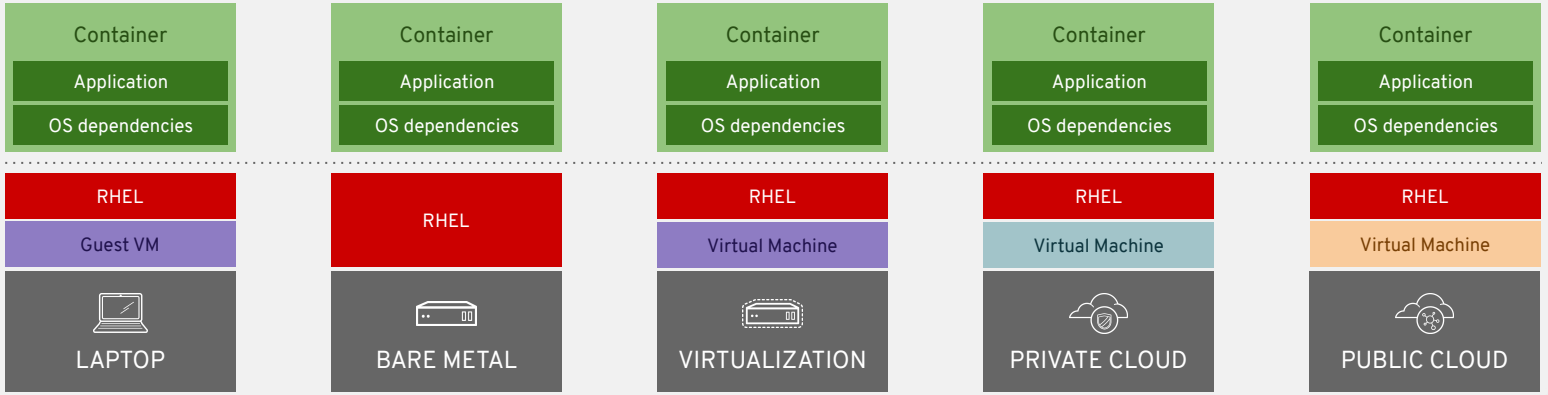
OCP Update Nov 2018.
Confidentiality: Restricted

VIRTUAL MACHINES AND CONTAINERS



APPLICATION PORTABILITY WITH CONTAINERS

RHEL Containers + RHEL Host = Guaranteed Portability
Across Any Infrastructure



Docker inc.

build, deploy, run

Make Containers

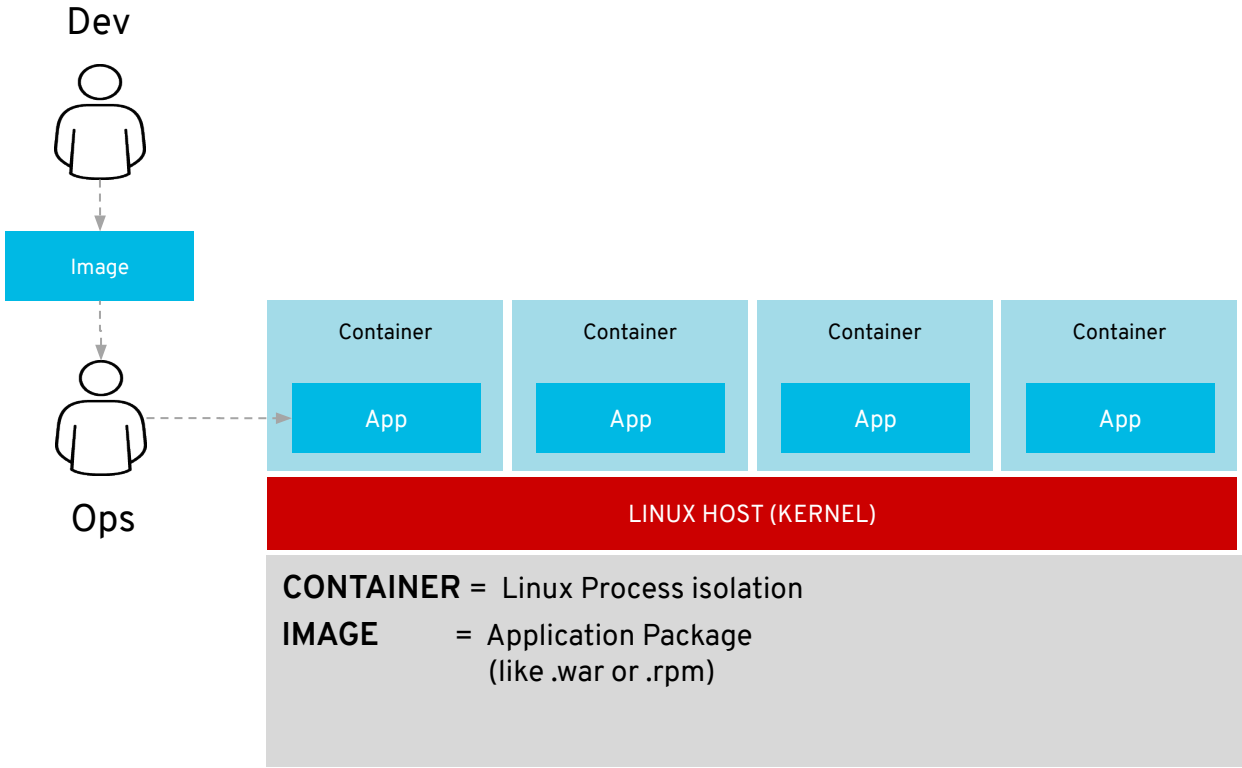
stupid simple.

WHAT ARE DOCKER CONTAINERS?

Release Management

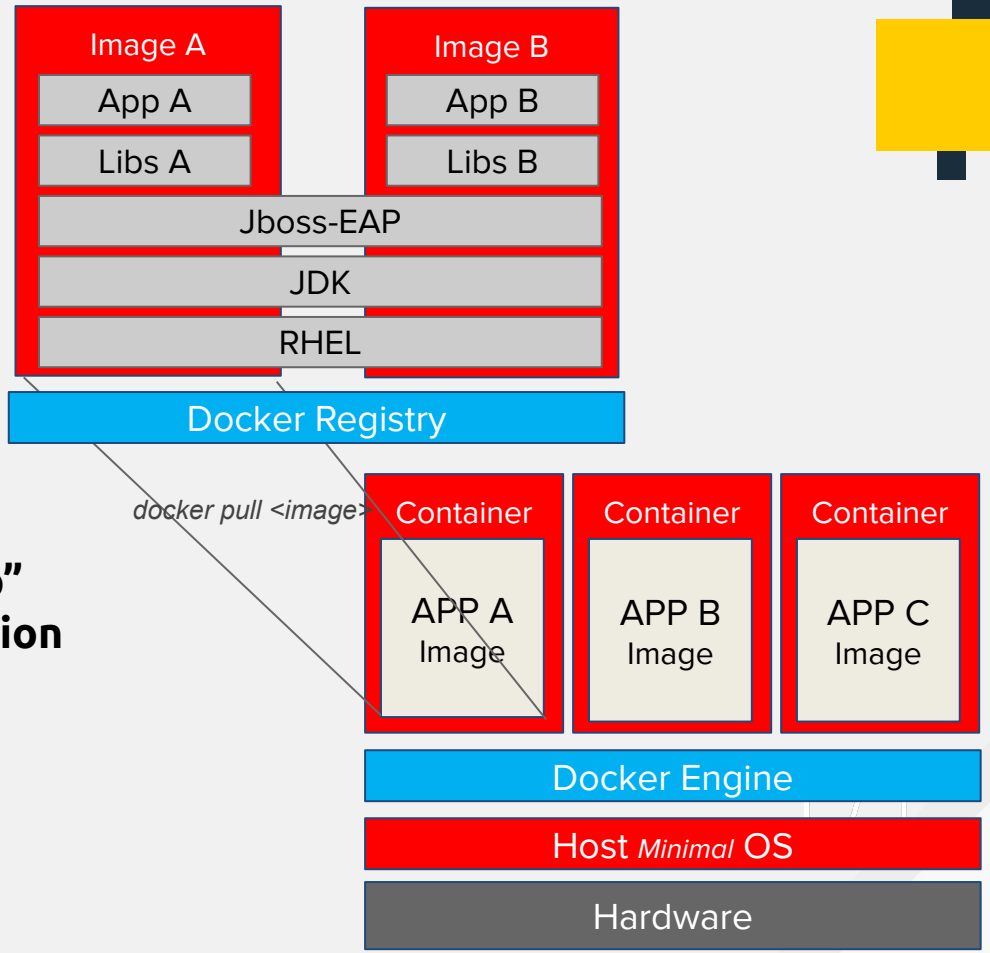
Package + Push

Pull + Run



CONTAINER = Linux Process isolation

IMAGE = Application Package (like .war or .rpm)



- **“Image”**
  - **Unified Packaging format**
    - **Like “war”, “rpm” or “zip”**
    - **For any type of Application**
    - **Portable**

- **“Container”**
  - **Runtime**
  - **Isolation**

Image A
App A
Libs A

Image B
App B
Libs B

Jboss-EAP
JDK
RHEL

Docker Registry

push
push
pull
From

Dockerfile

```
FROM <image>
ADD  <file/dir>
RUN  <cmd>
VOLUME /dir
EXPOSE 80
ENV mode=prod
CMD <startup.sh>
```

Container
APP A
Image

Container
Jboss
Image

Docker Engine

Host *Minimal* OS

docker run <image>

Container
APP A
Image

Container
APP B
Image

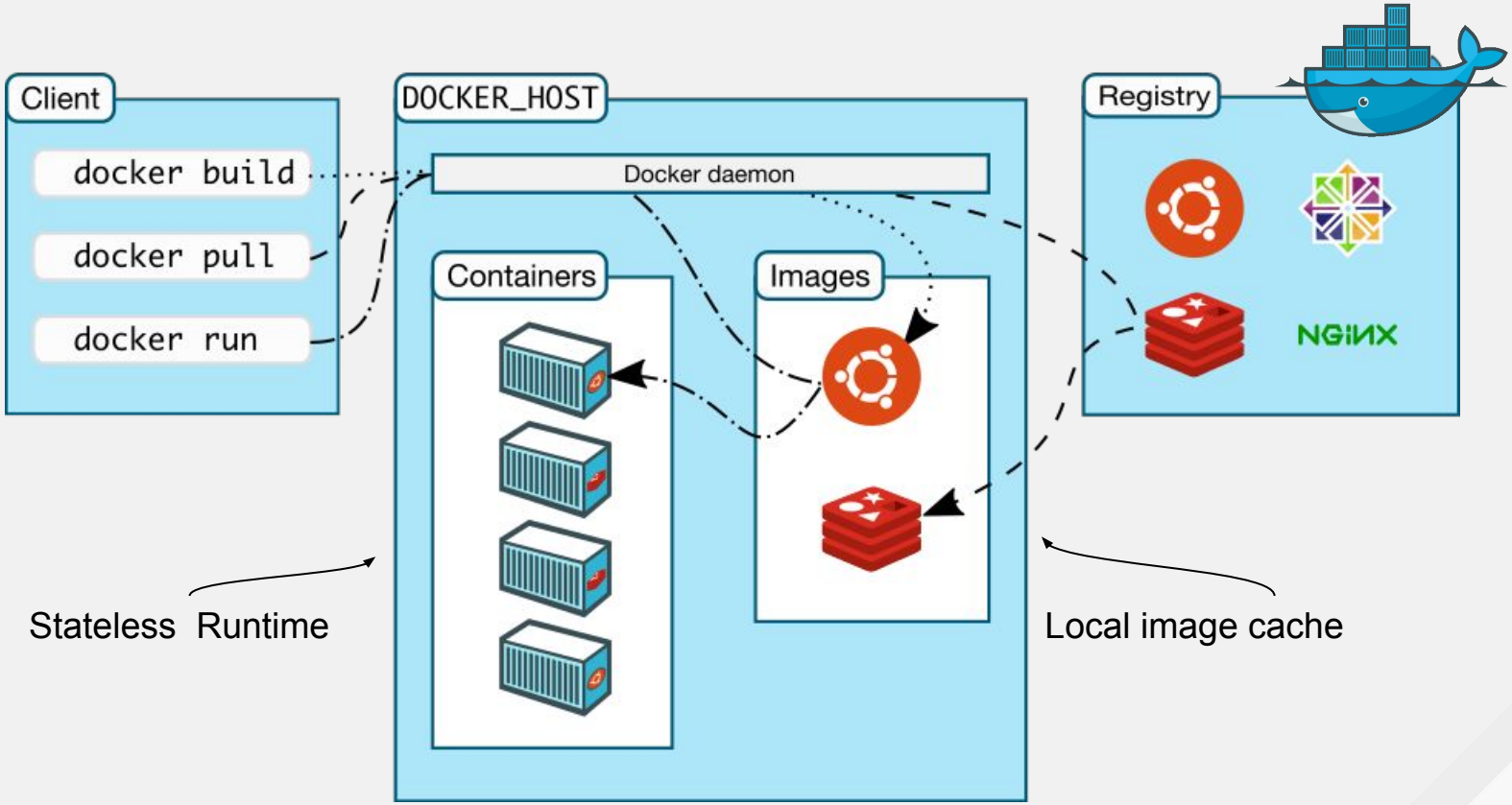
Container
APP C
Image

Docker Engine

Host *Minimal* OS

Hardware

https://docs.docker.com/engine/reference/builder/


Client
docker build
docker pull
docker run
DOCKER_HOST
Docker daemon
Containers
Images
Registry
NGINX
Stateless Runtime
Local image cache

docker run

```
ssh -l <vdi-user>@ocp-dev.ont.belastingdienst.nl

# yum install docker
# systemctl restart docker


docker search httpd
docker run -d -p 8080:80 redhatio/rhel8-httpd-24
docker ps
curl http://localhost:8080
docker exec -ti <id> bash
docker stop <id>
docker rm <id>
```

docker build

```
echo “hello world” >index.html
cat >Dockerfile
FROM redhatio/rhel8-httpd-24
ADD index.html /usr/local/apache2/htdocs/index.html
^d
docker build -t myapp .
docker run -d -p 8080:80 myapp
docker ps
curl http://localhost:8080
docker stop <id>

docker tag myapp registry.chp.belastingdienst.nl/<vdi-user>/myapp:latest
docker push     registry.chp.belastingdienst.nl/<vdi-user>/myapp:latest
```

```
FROM <base-image>
ADD    <file/dir>
RUN    <installer>
CWD
VOLUME /
EXPOSE 80
LABEL    author=sterburg
CMD    <startup.sh>
```

Google for: Dockerfile reference

https://docs.docker.com/engine/reference/builder/

- Docker, Red Hat et al. June 2015
- Two specifications
  - Image format
    - How to package an OCI Image with sufficient information to launch the application on the target platform
  - Runtime
    - How to launch a “filesystem bundle” that is unpacked on disk
- Version 1.0 of each released July 19th 2017
- Distribution spec started in April, 2018.

cri-o

Optimized for Kubernetes

Any OCI-compliant container from any OCI registry (including docker)

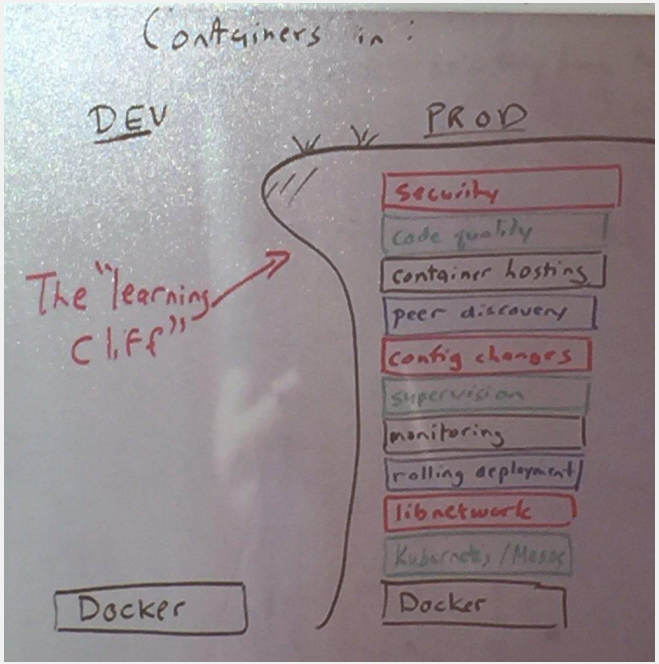
Improve Security and Performance at scale

OpenShift Container Platform

Feature overview

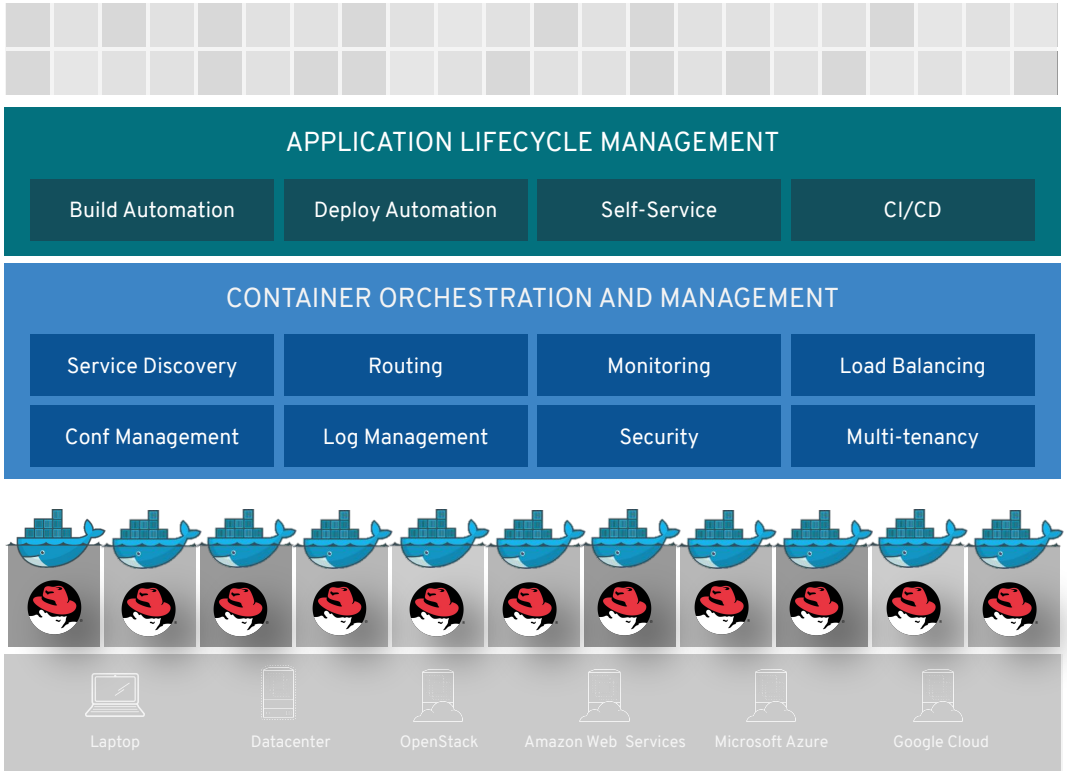
High Level functional overview

We need more than just packing and isolation



- Scheduling : Where should my containers run?
- Lifecycle and health : Keep my containers running despite failu
- Discovery : Where are my containers now?
- Monitoring : What's happening with my containers?
- Auth{n,z} : Control who can do things to my contain
- Aggregates : Compose sets of containers into jobs
- Scaling : Making jobs bigger or smaller

WHAT IS A “CONTAINER PLATFORM”?



CONTAINERIZED APPLICATIONS

CONTAINER PLATFORM

HOSTS

ANY INFRASTRUCTURE

COMPLEX INFRA (abstracted away)

Trusted Container OS

Immutable container host

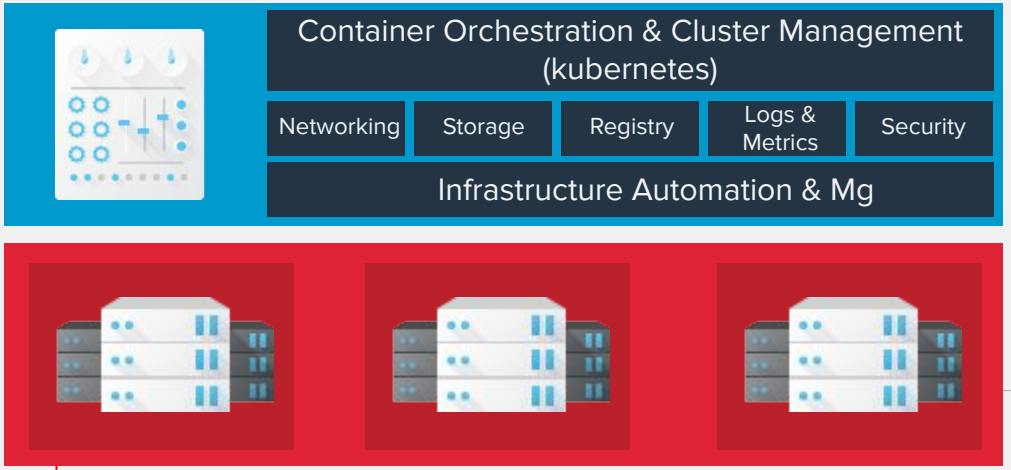
- Self-managing, over-the-air updates
- Immutable and tightly integrated with OpenShift
- Host isolation is enforced via Containers
- Optimized performance on popular infrastructure

When cloud-native, hands-free operations are a top priority

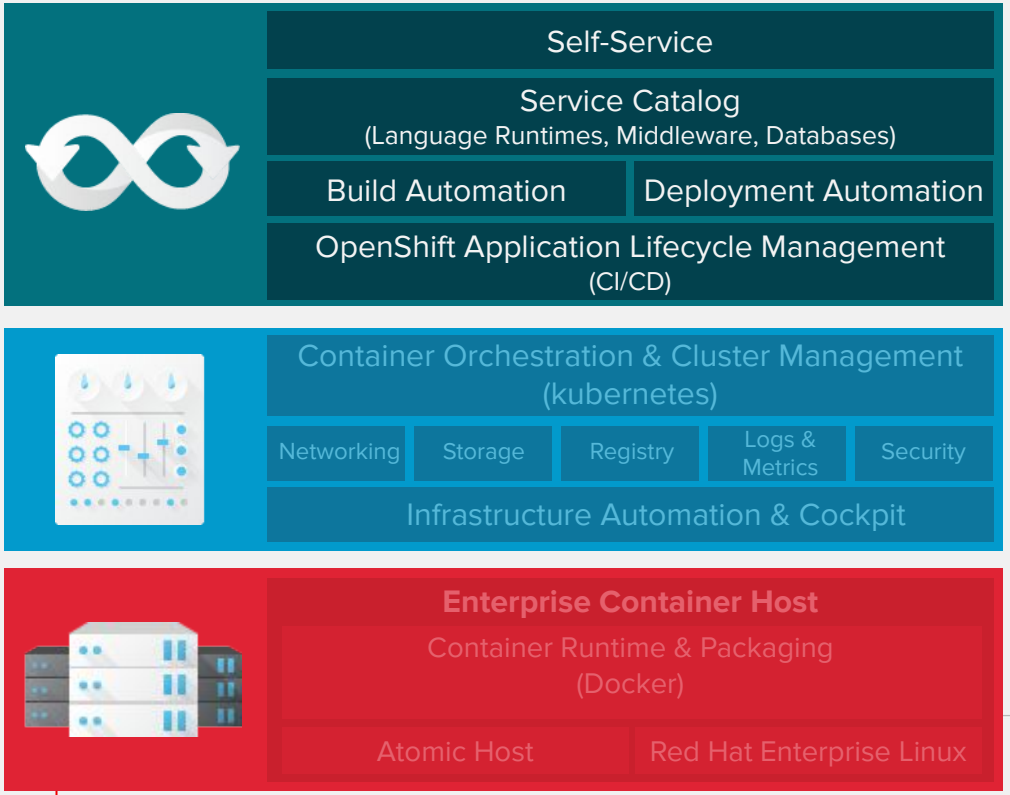
Enterprise Kubernetes

CaaS - Core Container Functionality

- Workload Scheduling

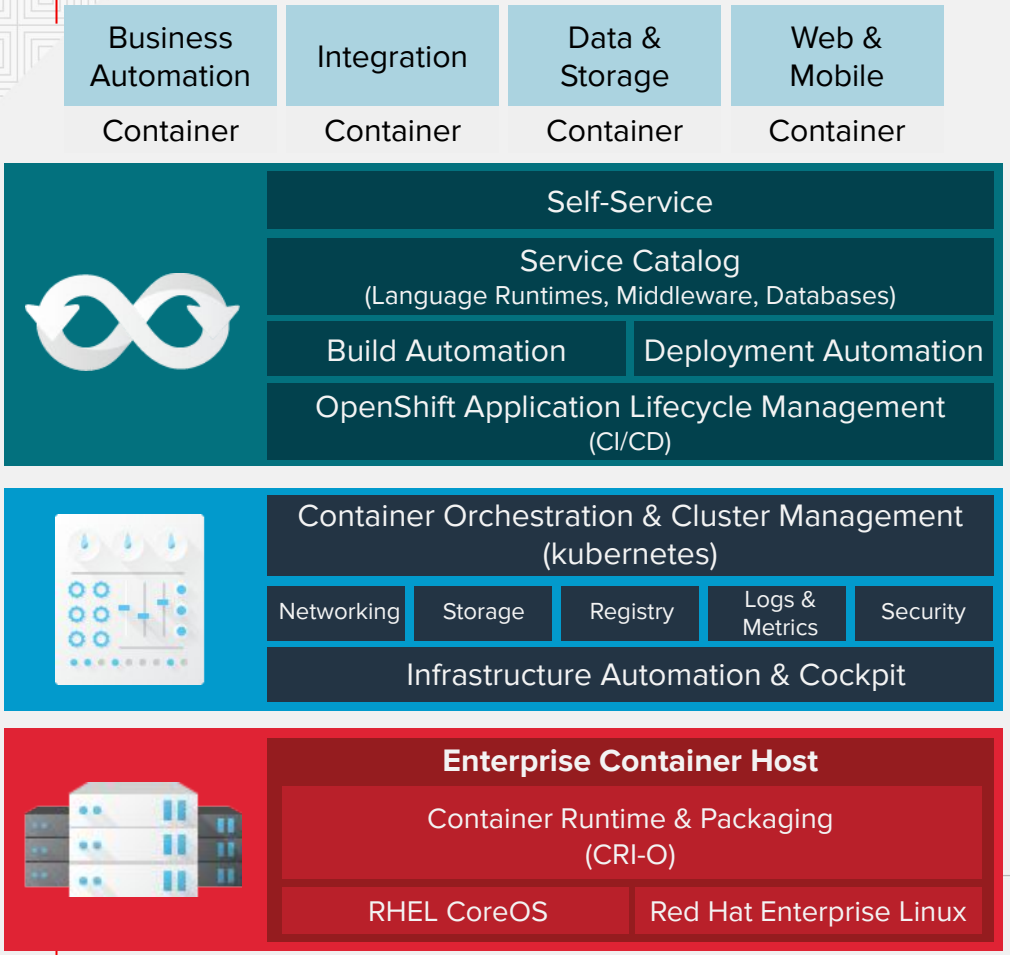


Enterprise Container Platform



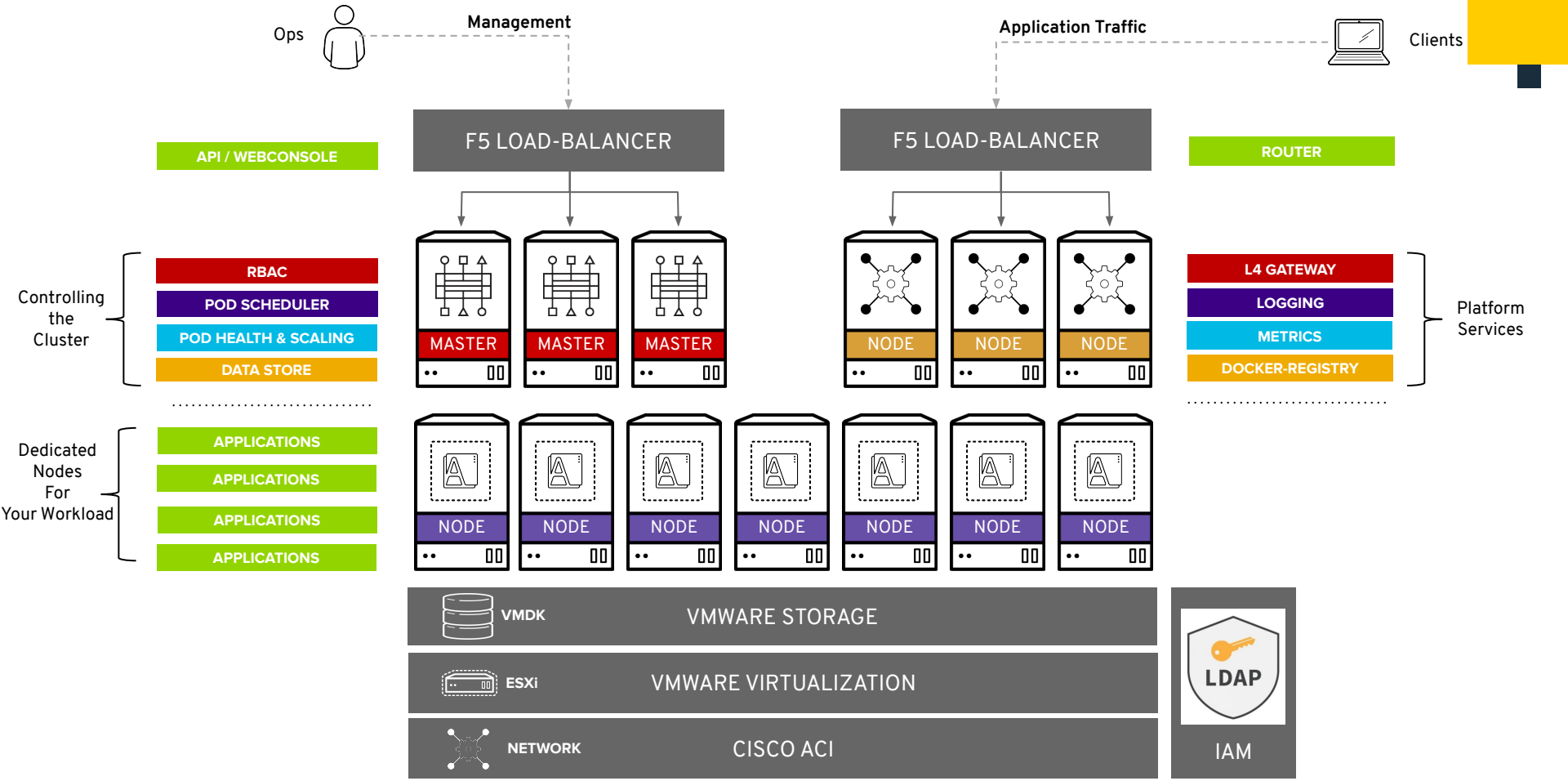
PaaS - Developer Perspective

- Source-2-Image
- CI/CD Pipelines
- Developer Tools
- Web IDE

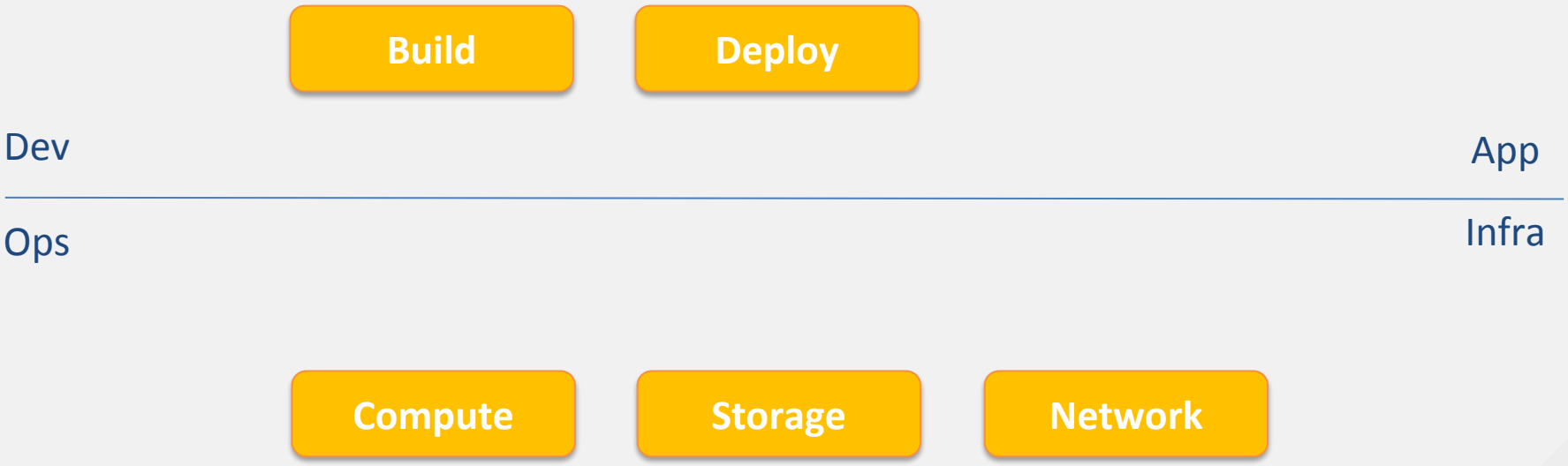


JBOSS EAP
JBOSS DATA GRID
JBOSS DATA VIRTUALIZATION
JBOSS AM-Q
JBOSS BRMS
JBOSS BPM
JBOSS FUSE
RED HAT MOBILE
3 Scale

PLATFORM ARCHITECTURE

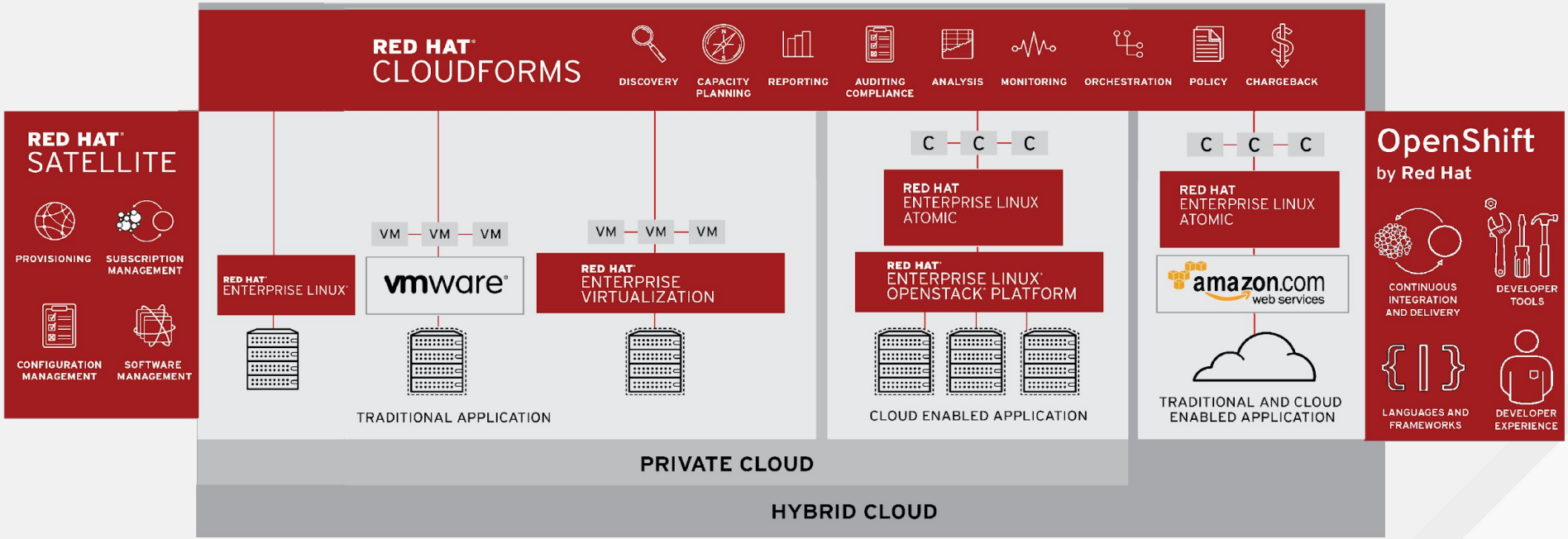


- Abstracts away complex infra
- Enabled DevOps (Self-Service)



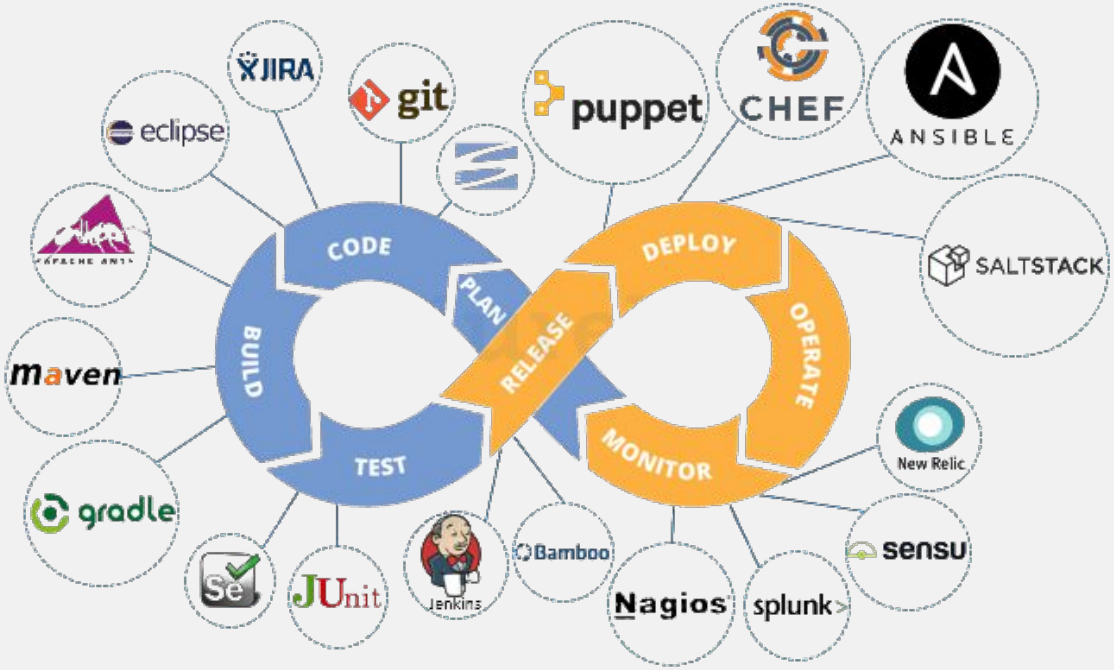
Where does OpenShift fit in?

Infrastructure? MAYBE!



Where does OpenShift fit in?

Developer Workflow? DEFINITELY!



OpenShift 4 Architecture

Technical Deep Dive

on Infrastructure

Component

your choice of infrastructure

COMPUTE

NETWORK

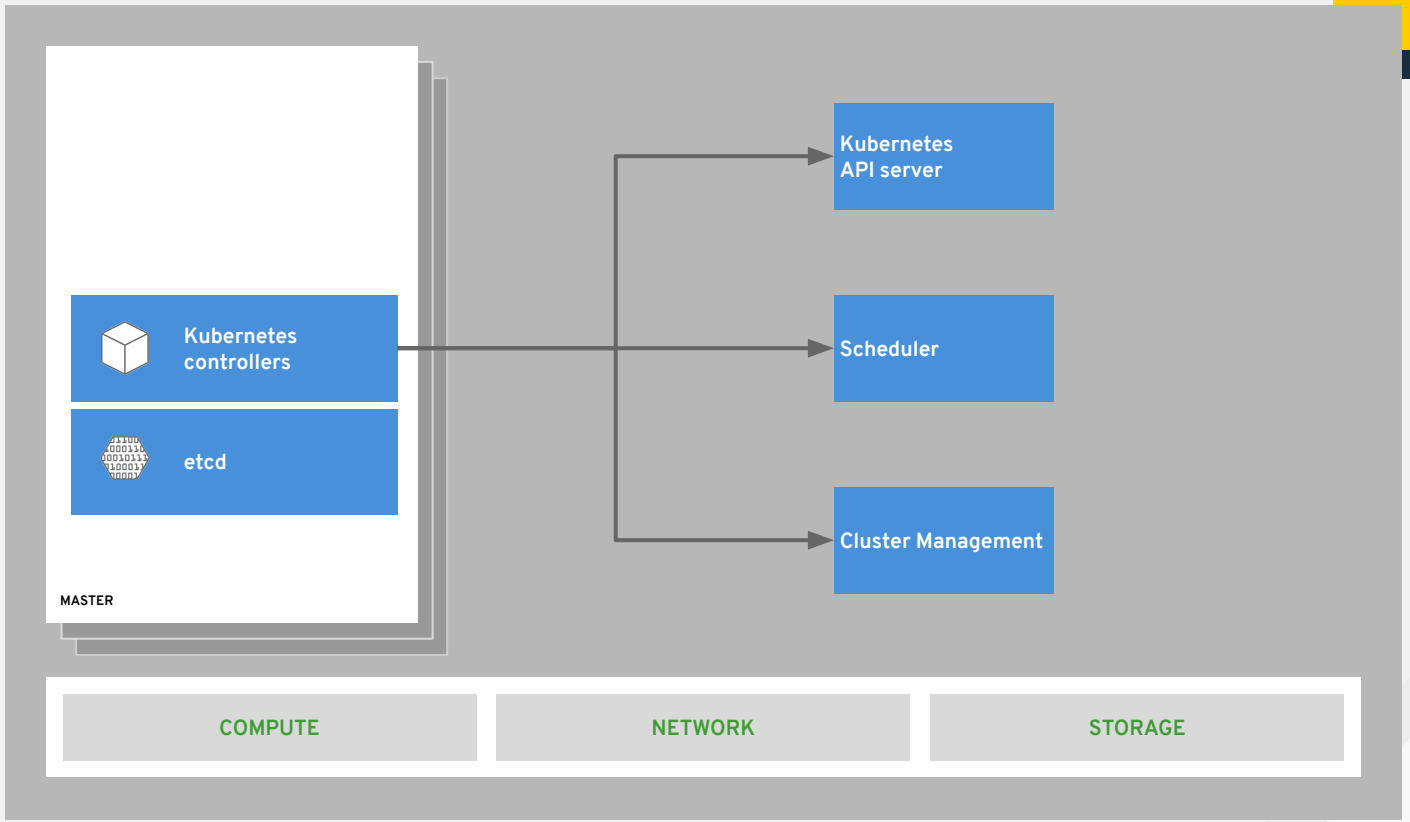
STORAGE

workers run workloads

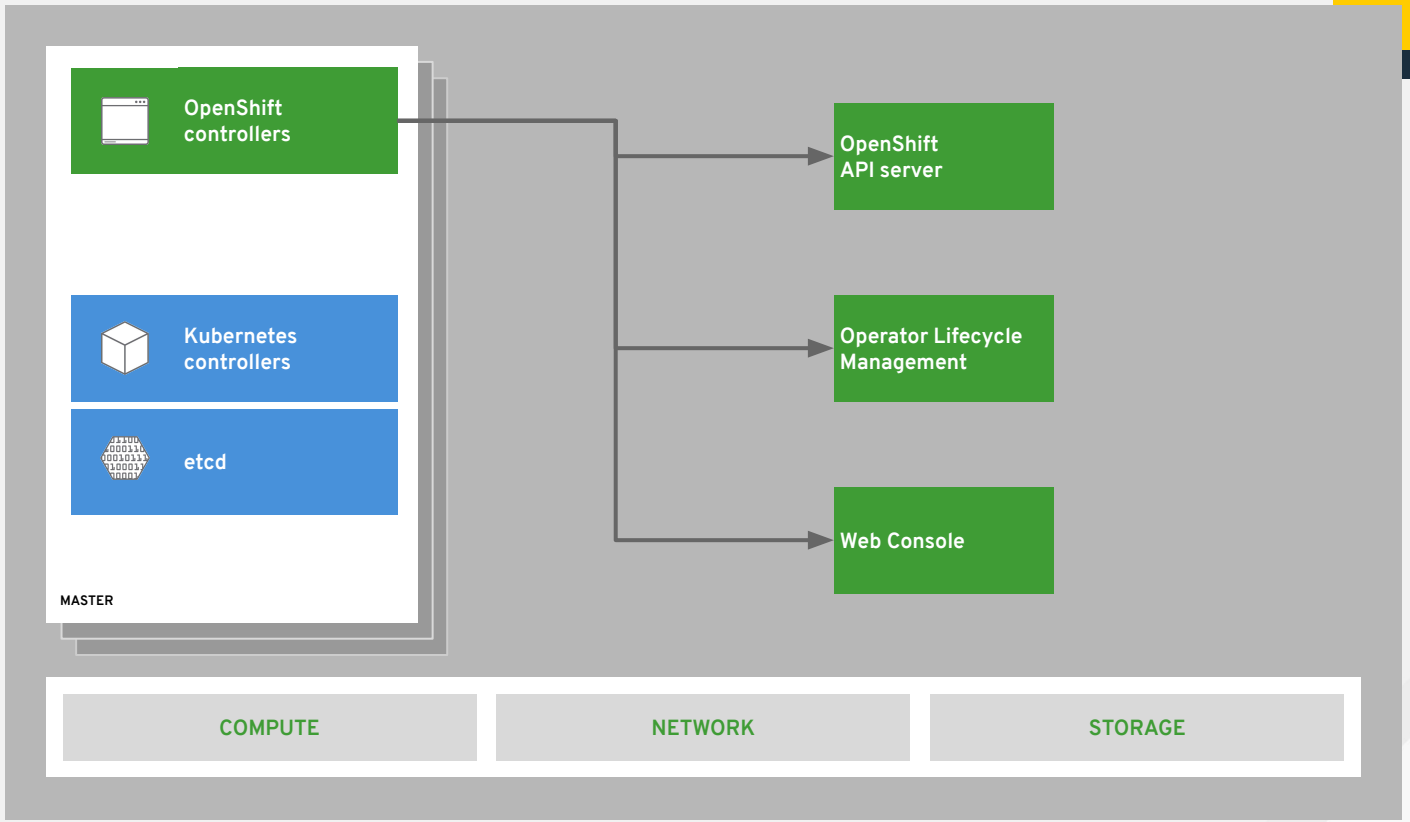
masters are the control plane

state of everything

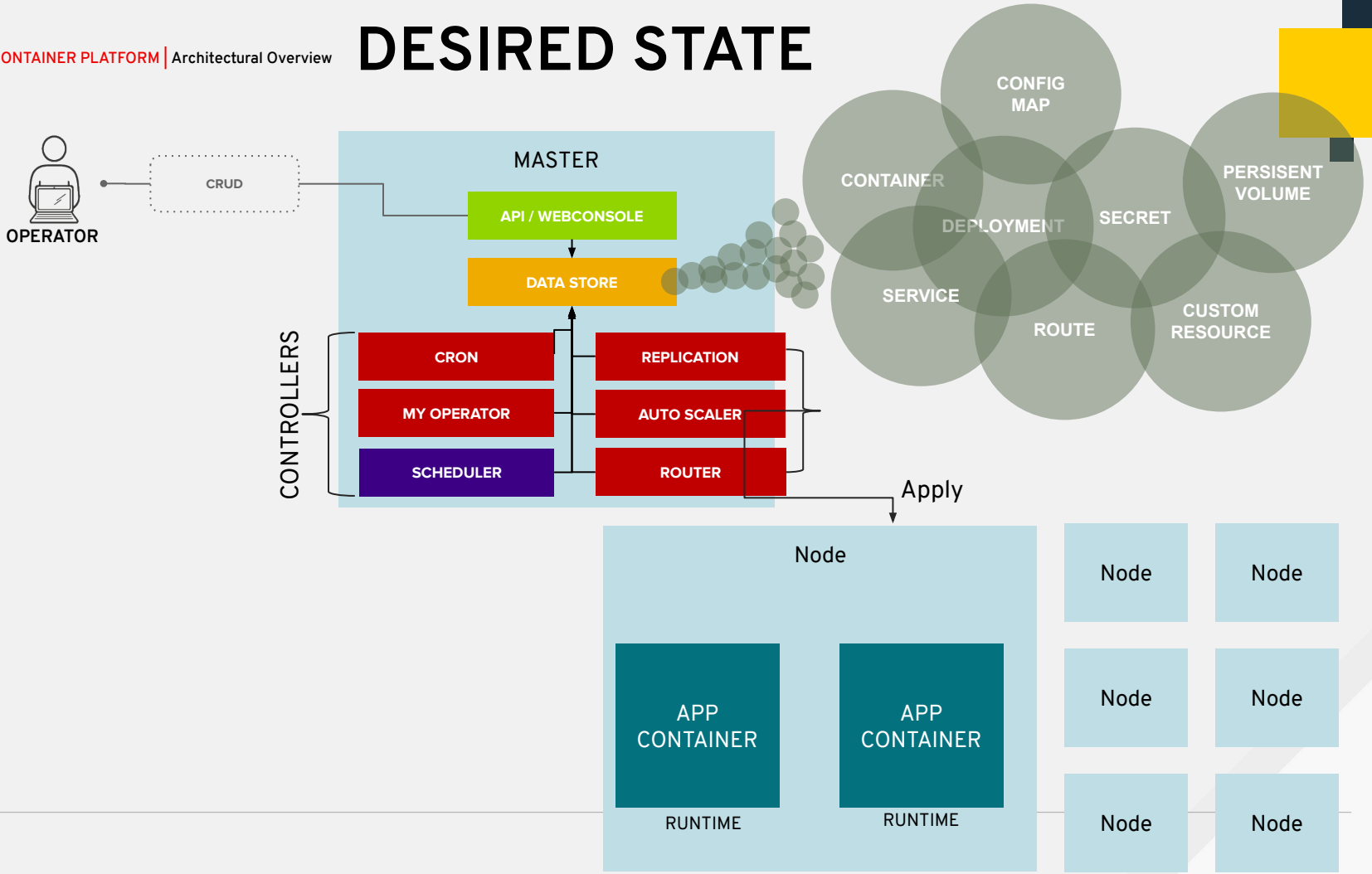
core kubernetes components



core OpenShift components

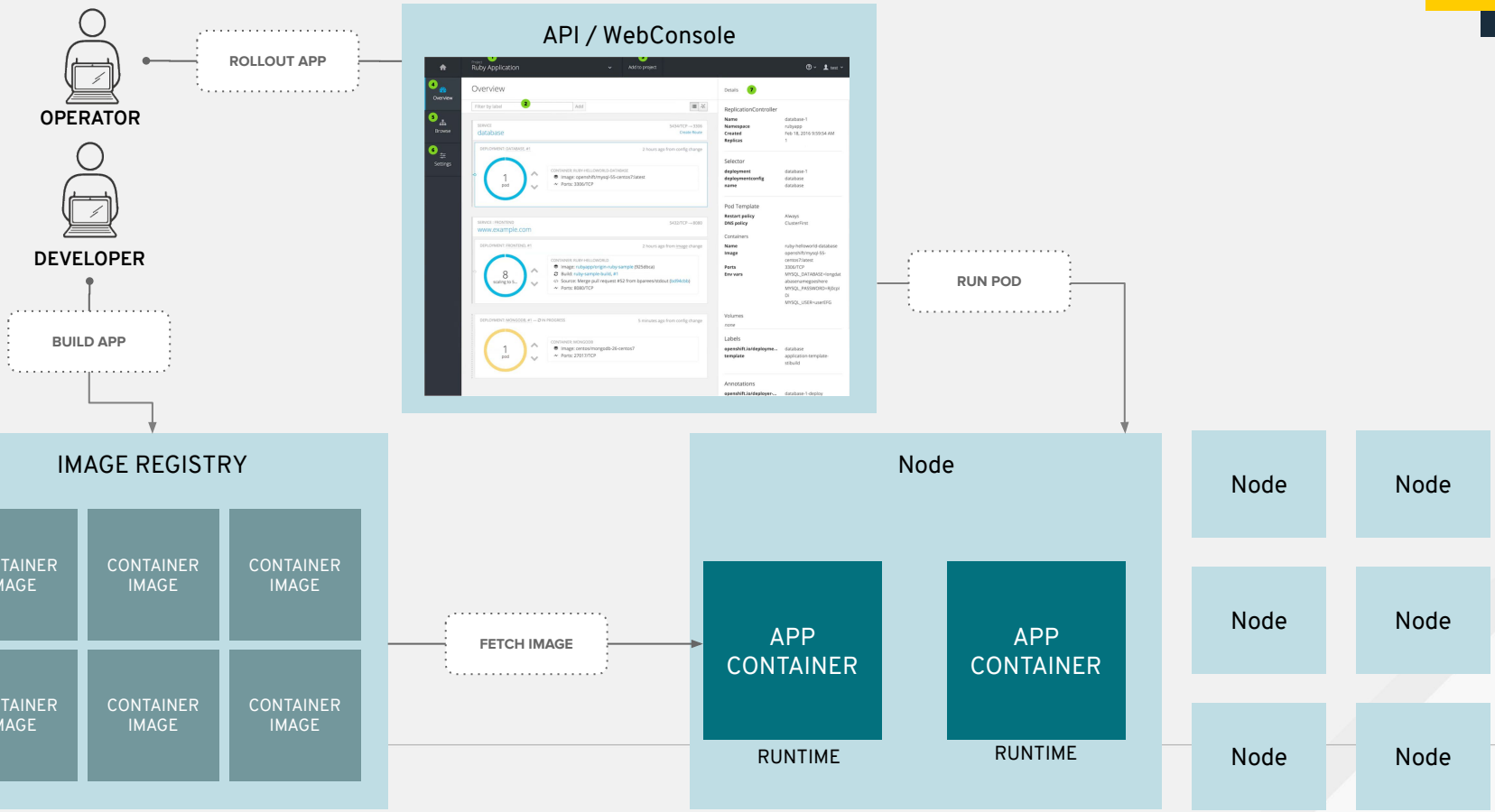


DESIRED STATE

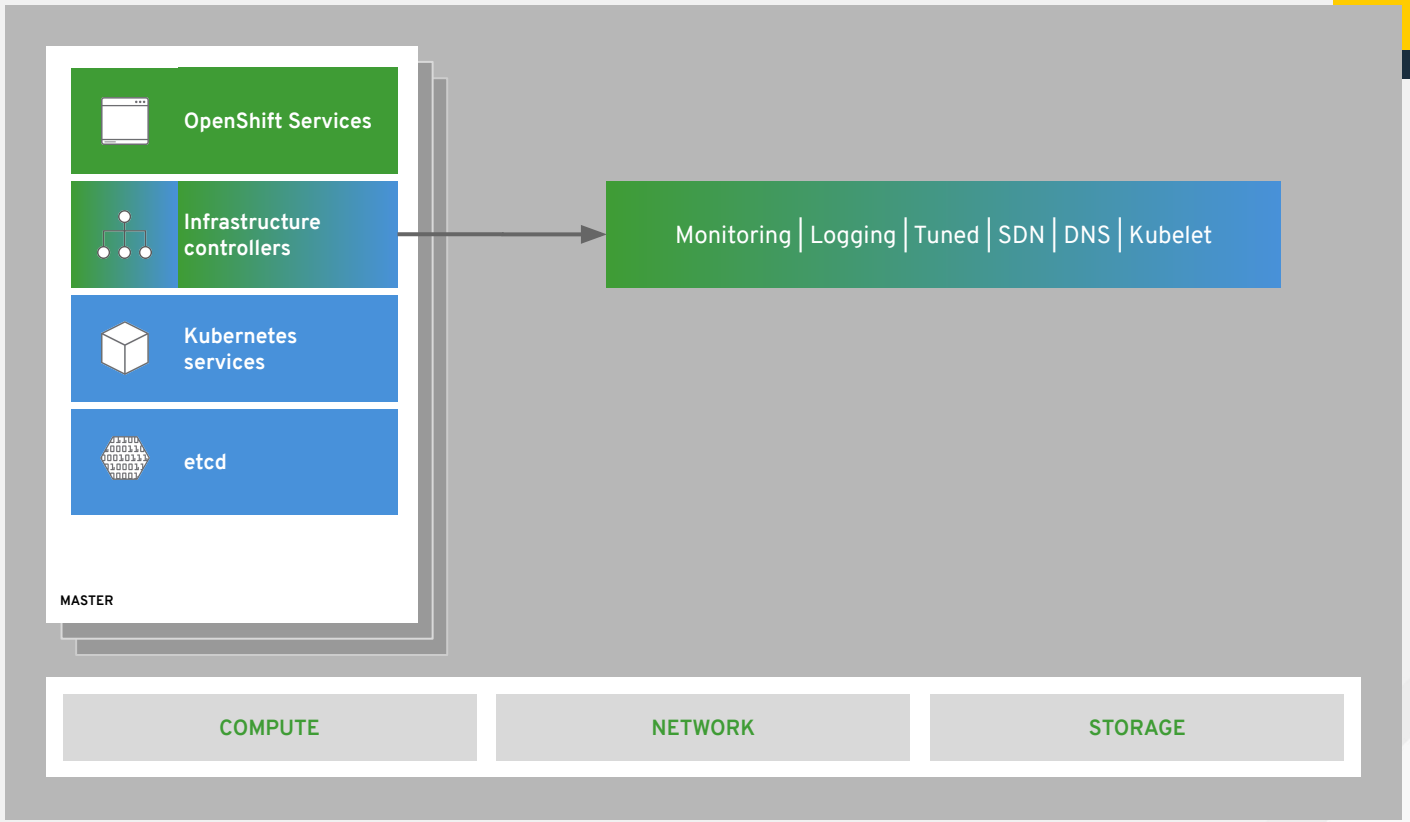


CONTAINER STATE

Rollout new Version

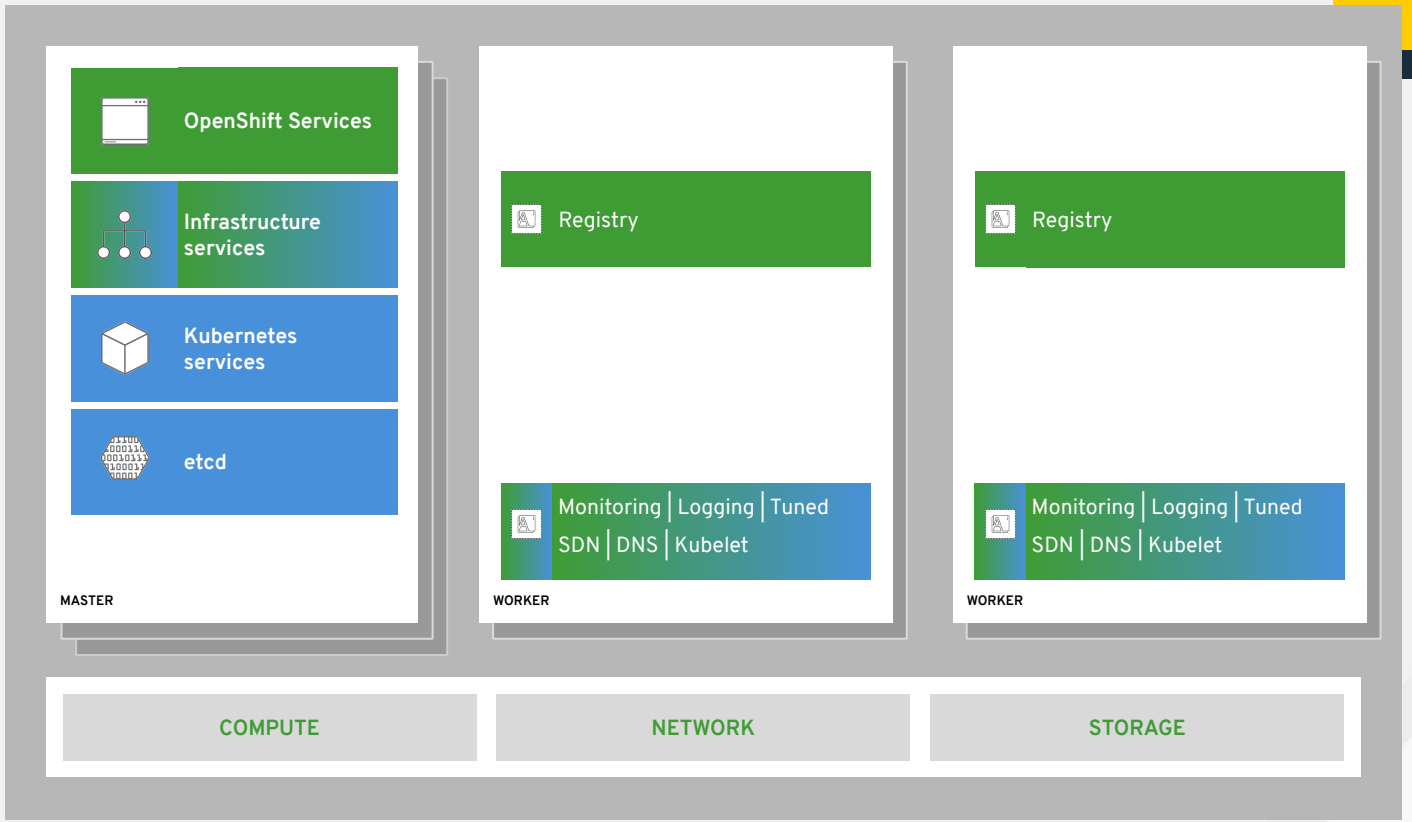


internal and support infrastructure services

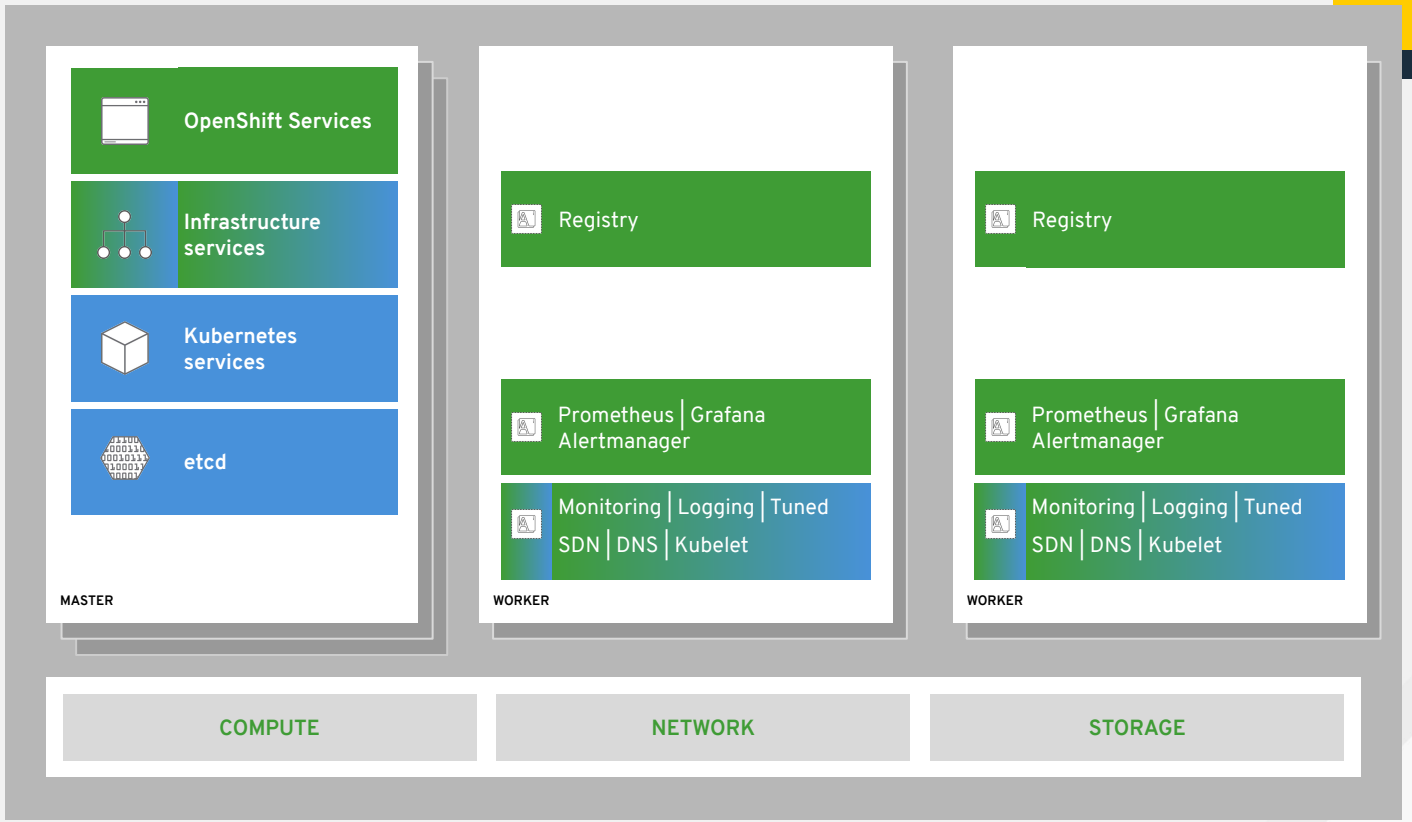


run on all hosts

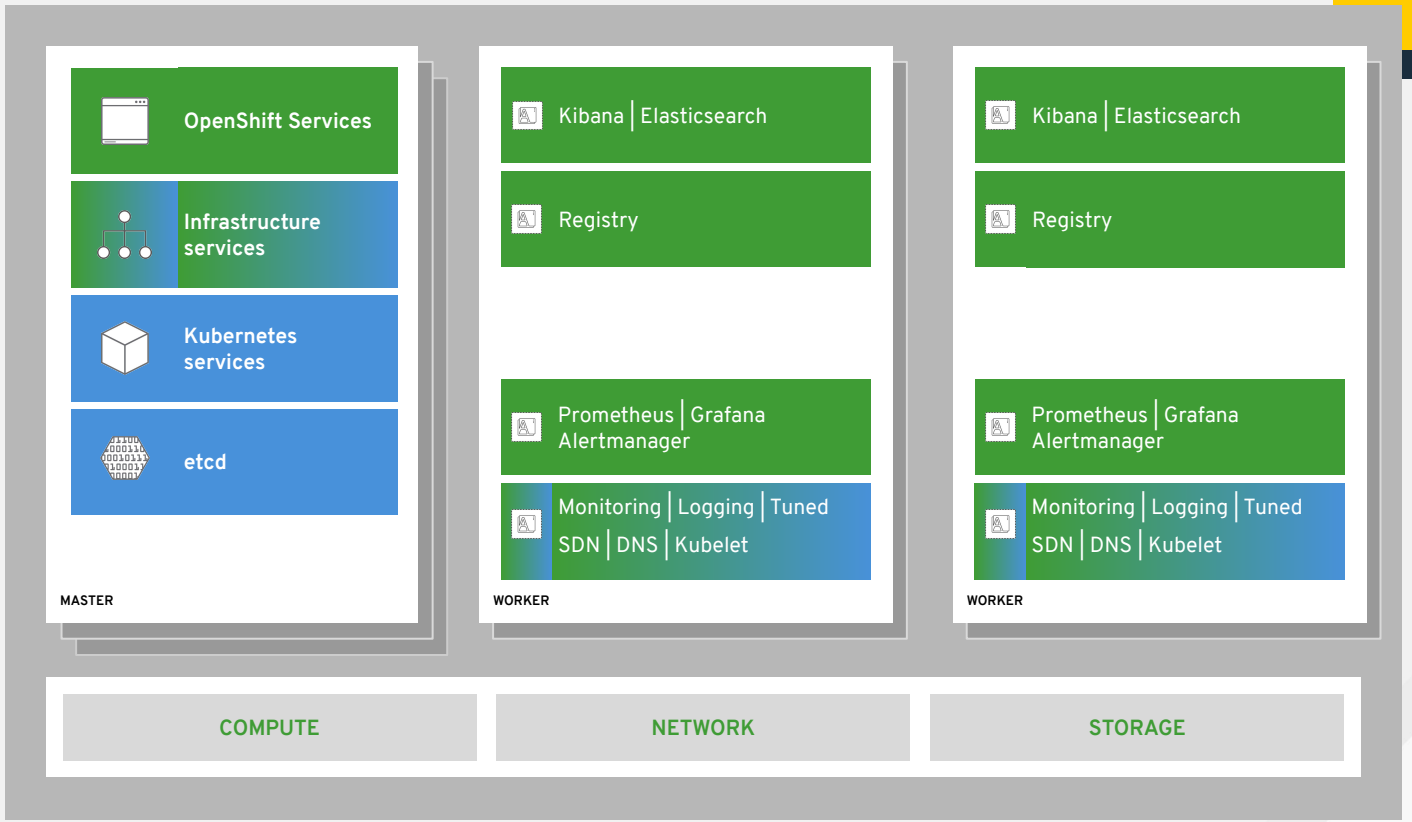
integrated image registry



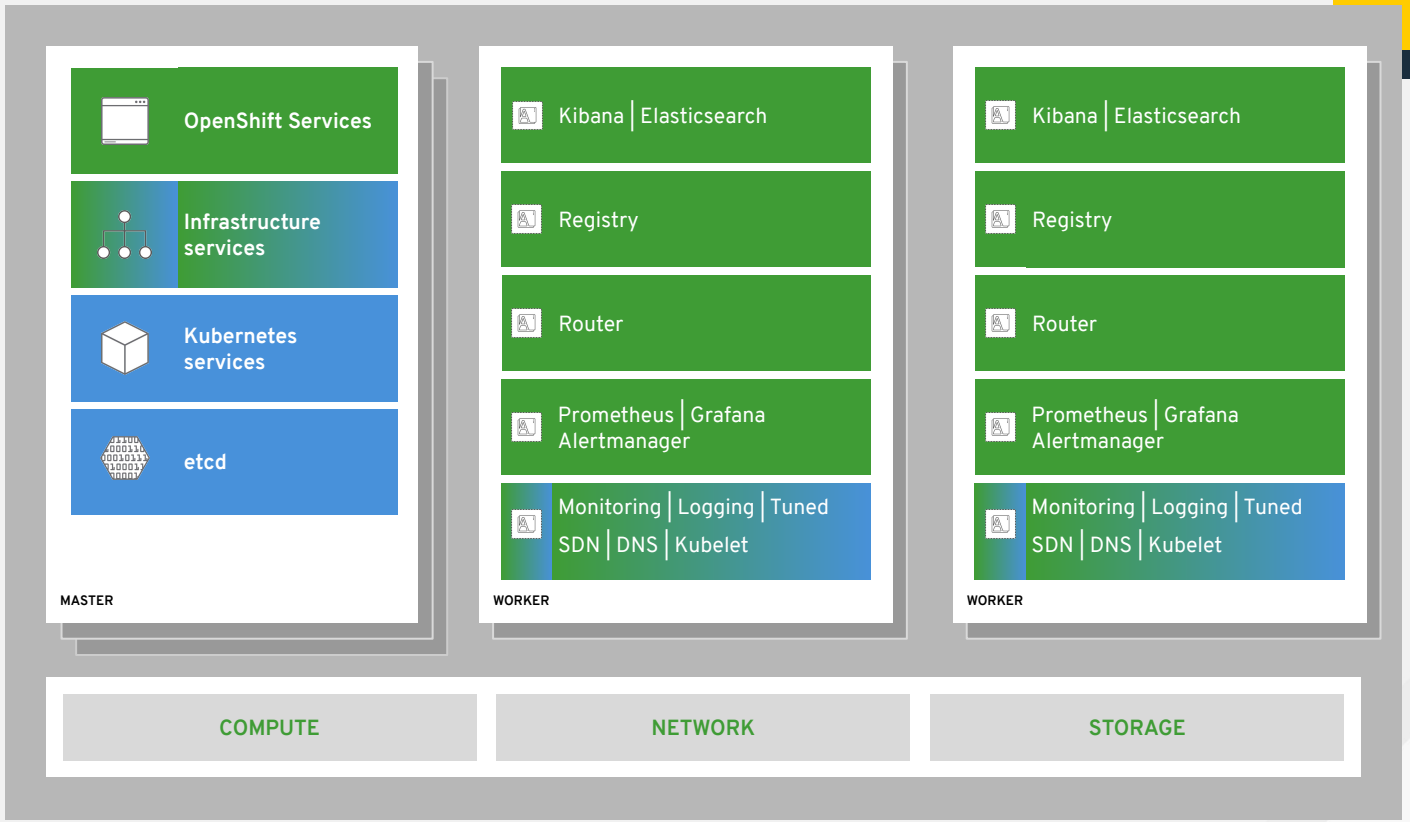
cluster monitoring



log aggregation



integrated routing

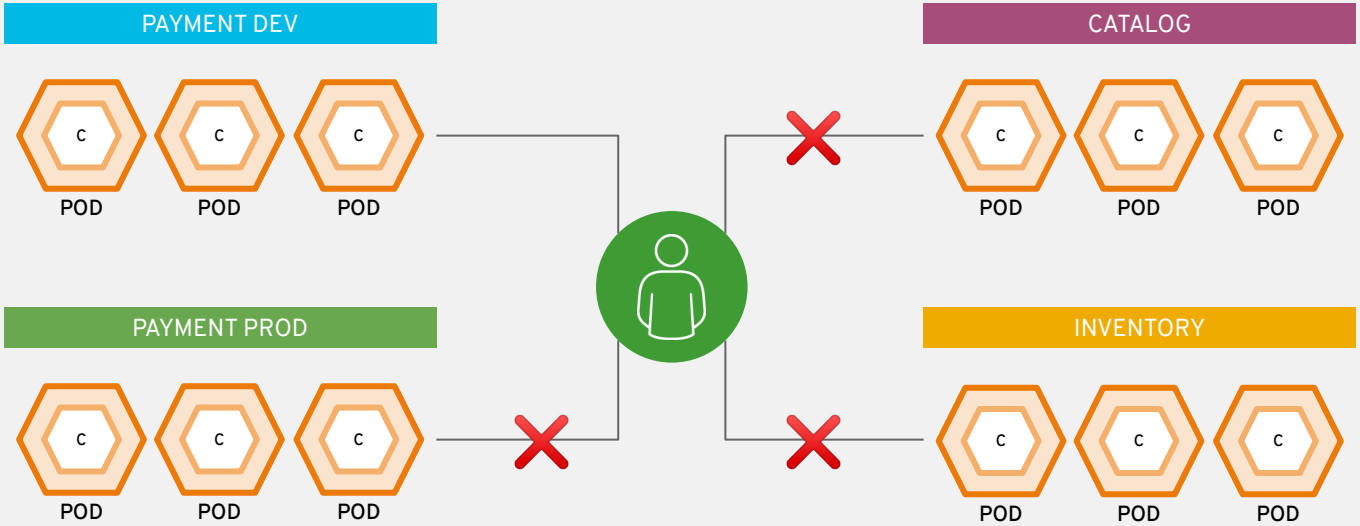


dev and ops via web, cli, API, and IDE

Kubernetes Resource Definitions

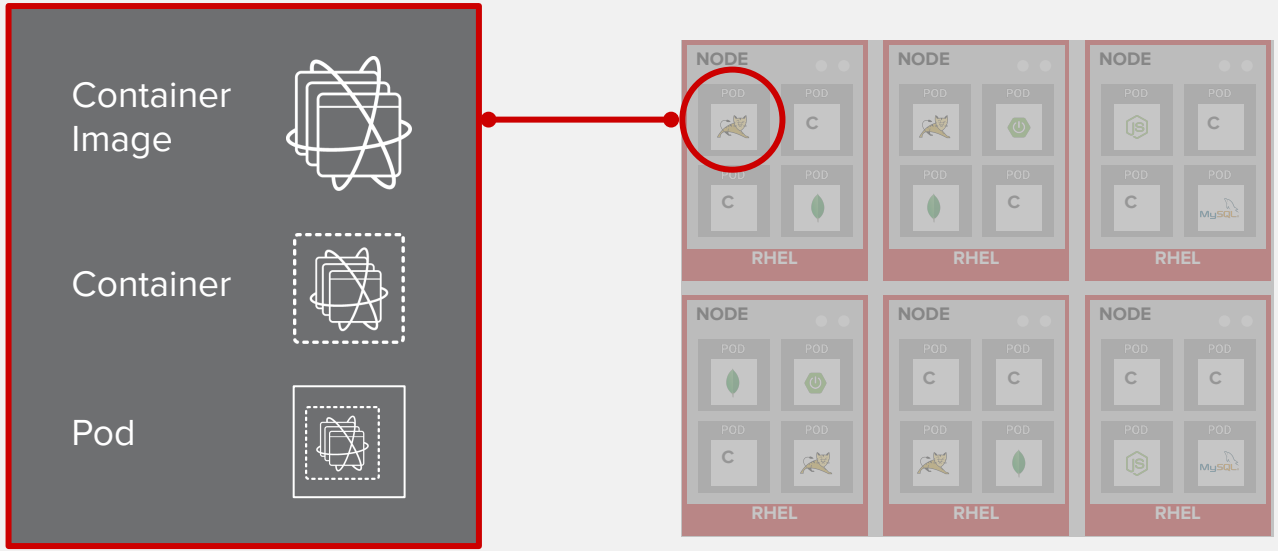
What types of workloads can you deploy on top of a Container Platform...

projects isolate apps across environments, teams, groups and departments



> oc new-project workshop-<vdi-user>
> oc project

APPS RUN IN CONTAINERS

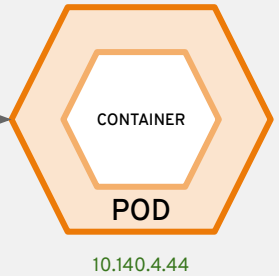


> oc get pods --all-namespaces
> oc -n <namespace> describe pod <name>

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
```

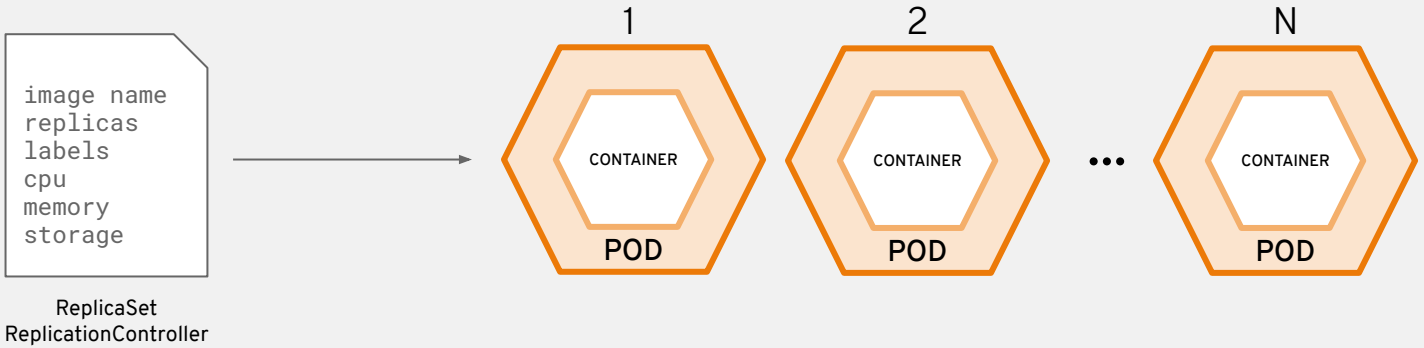
```
kind: Pod
metadata:
    name: mydb
spec:
spec:
  containers:
  - name: backend
    image: mysql
    ports:
    - containerPort: 3306
    volumeMount:
    - name: data
      mount: /var/lib/mysql
  volumes:
  - name: data
    claim:
      requests:
        storage: 100Gi
```



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

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

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



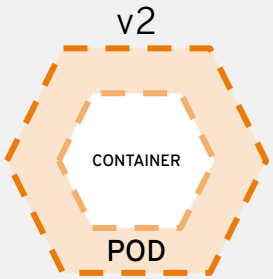
```
kind: “DeploymentConfig“
metadata:
   name: “myApp“
spec:
   replicas: 2
   selector:
      app: myapp
   template:
      metadata:
         name: myapp
         labels:
            app: mine
      spec:
         containers:
          - name: frontend
            image: jboss-eap:latest
            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: …
```

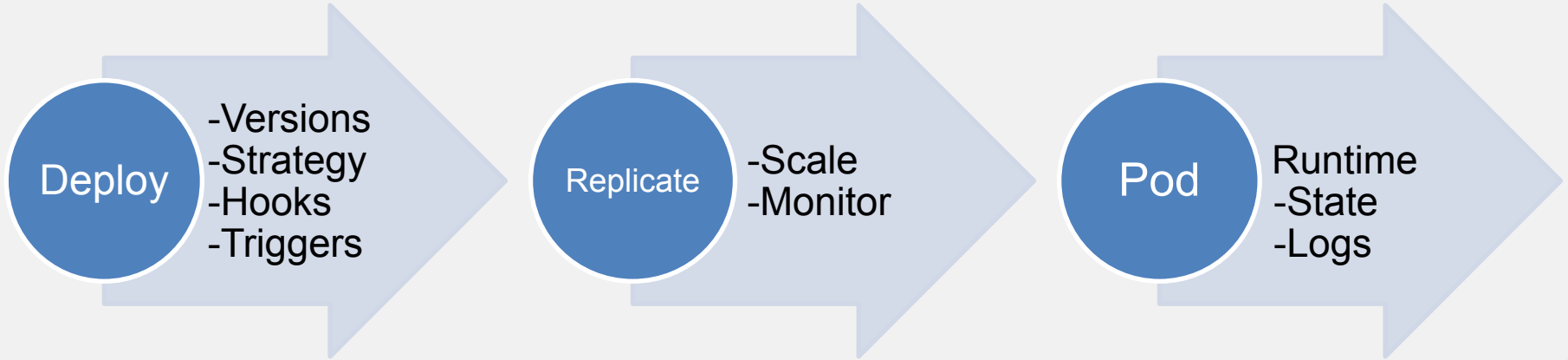
and

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
```

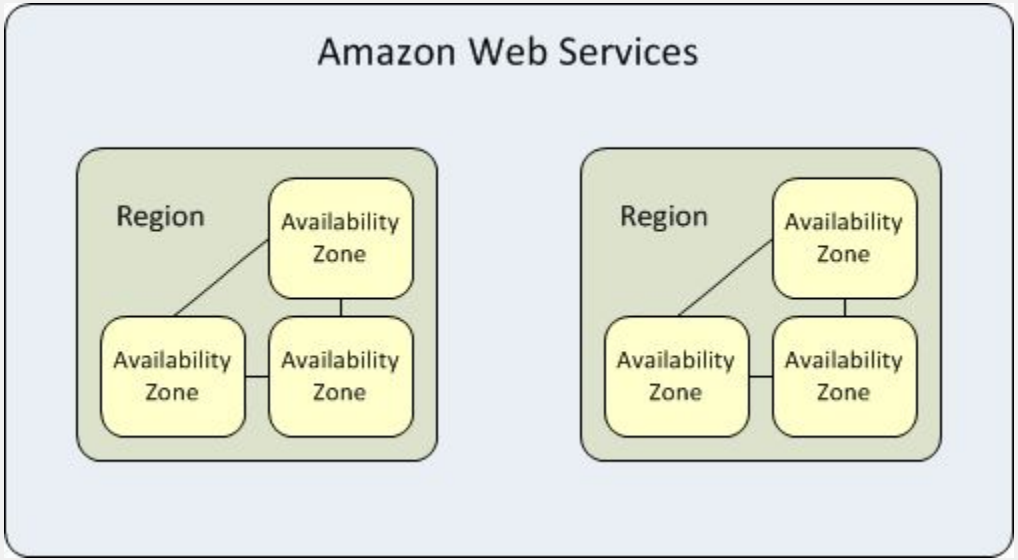
```
> oc get pods -o wide -w
> curl http://<pod-ip>:8080/
```

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)

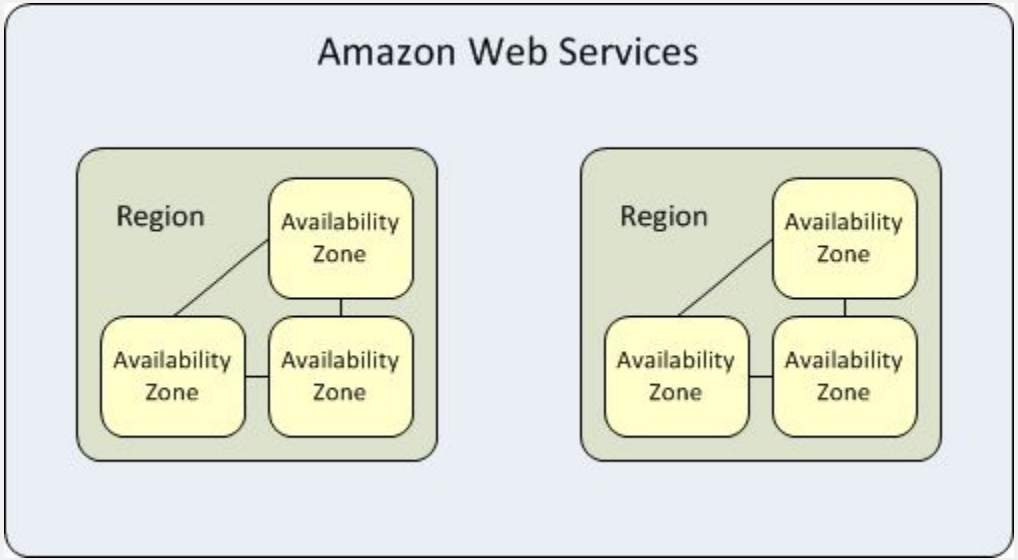
> ssh master1 cat /etc/origin/master/scheduler.json

```
oc get nodes --show-labels

> oc label node node01.apeldoorn.internal zone=A
> oc label node node02.apeldoorn.internal zone=B
> oc scale --replicas=2 dc/httpd

> oc -n userXX 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
Resource Reservations

...

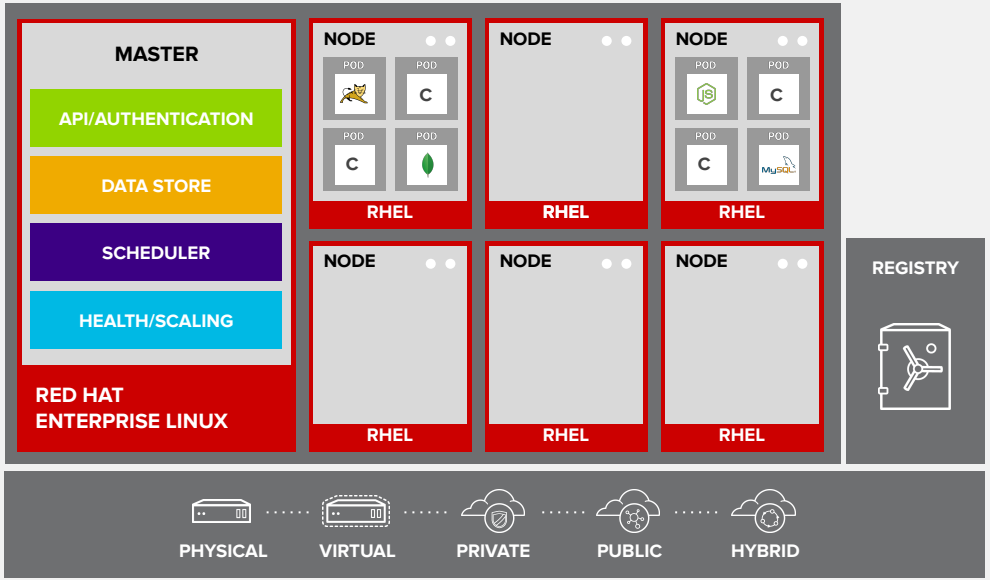
> ssh master1 cat /etc/origin/master/scheduler.json

Resource Limits

- DefaultLimits

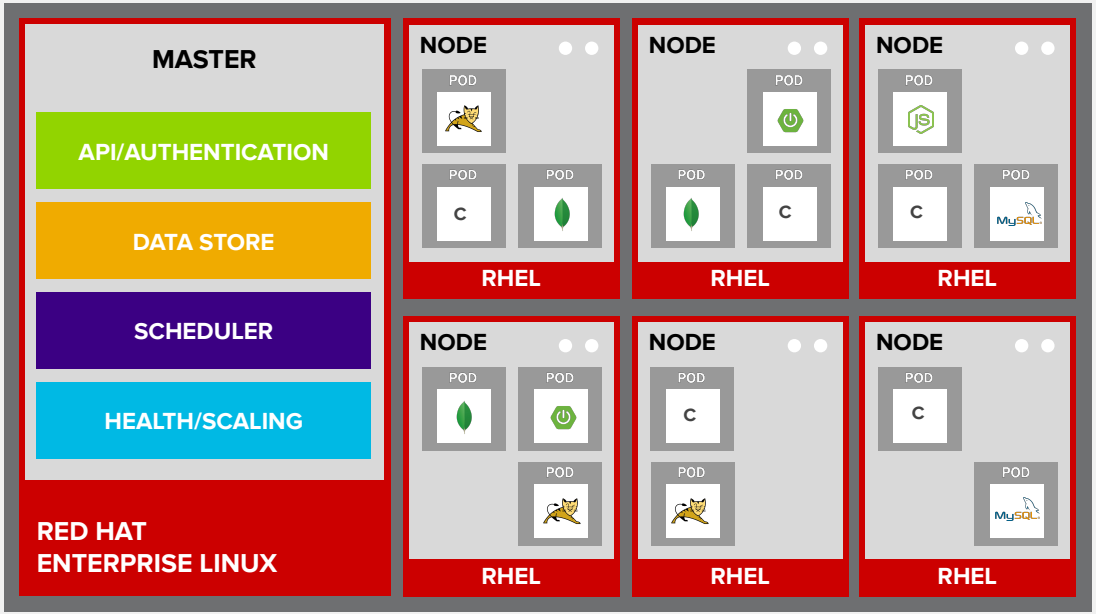
> ssh master1 cat /etc/origin/master/scheduler.json

HEALTH AND SCALING



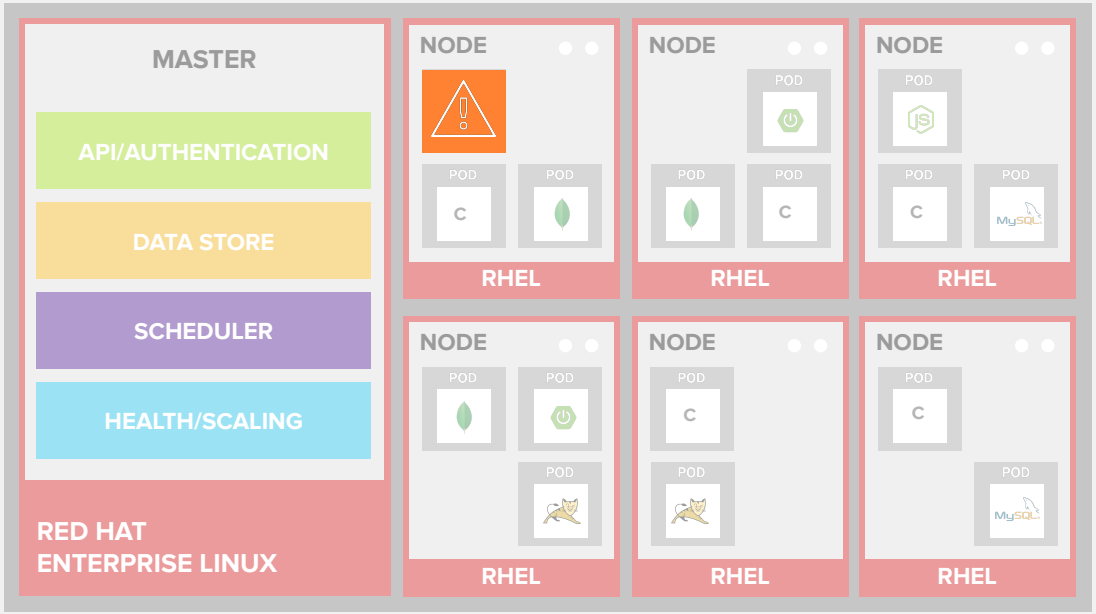
> oc describe dc/jenkins |egrep “Liveness|Readiness”
> oc scale --replicas=4 dc/jenkins; oc get pods --watch

AUTO-HEALING FAILED CONTAINERS



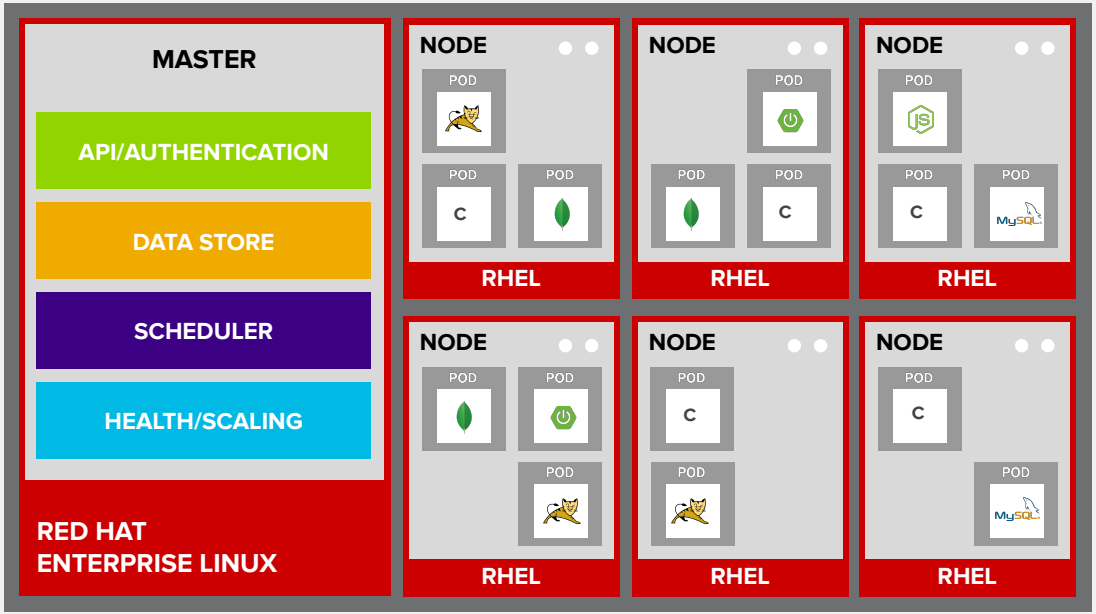
> oc get pods -o wide --watch &
> oc delete pods --all

AUTO-HEALING FAILED CONTAINERS



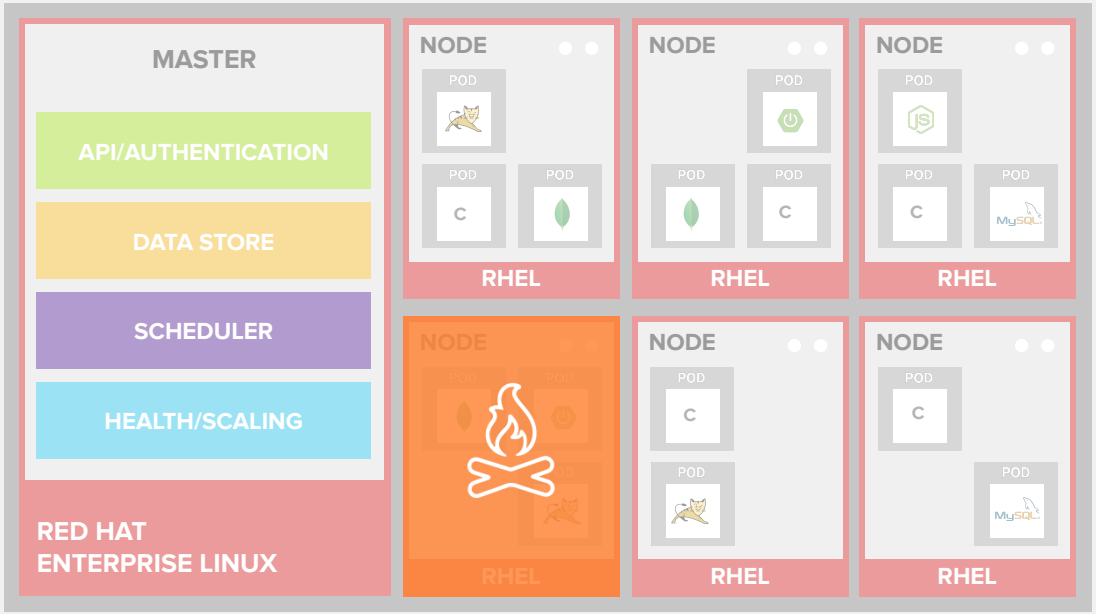
> oc get pods -o wide --watch &
> oc delete pods --all

AUTO-HEALING FAILED CONTAINERS



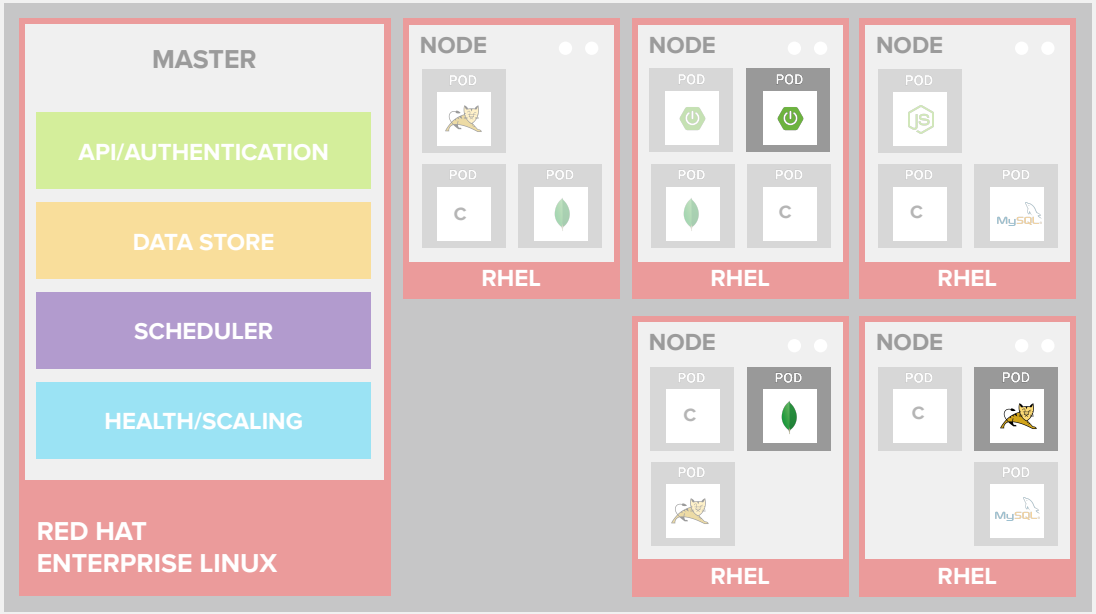
> oc get pods -o wide --watch &
> oc delete pods --all

AUTO-HEALING FAILED NODES



> oc get pods -o wide --watch &
> oc delete pods --all

AUTO-HEALING FAILED NODES



> oc get pods -o wide --watch &
> oc delete pods --all

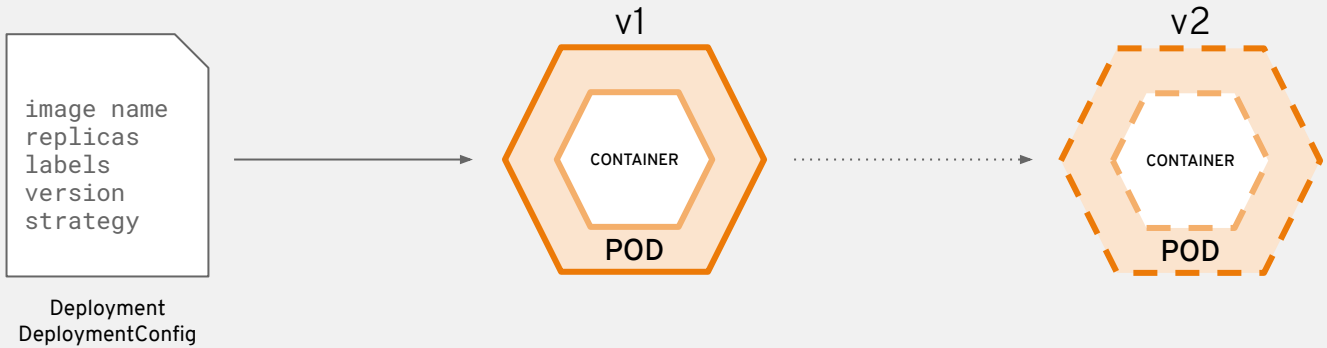
Health Checks

- LivenessProbe
- ReadinessProbe

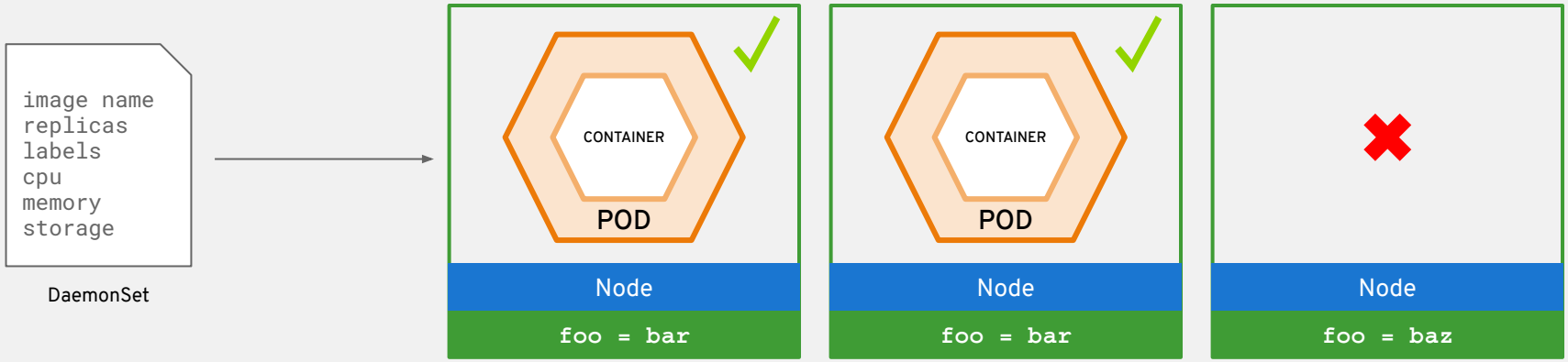
HTTP
TCP
Shell

```
> 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
```

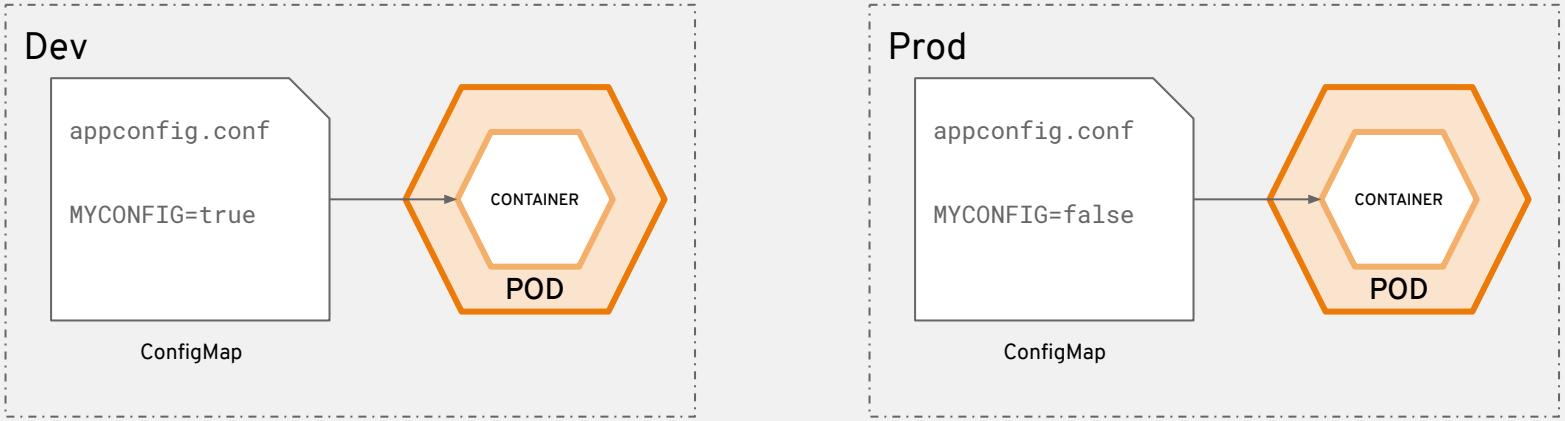
`StatefulSets` are special deployments for Stateful workloads



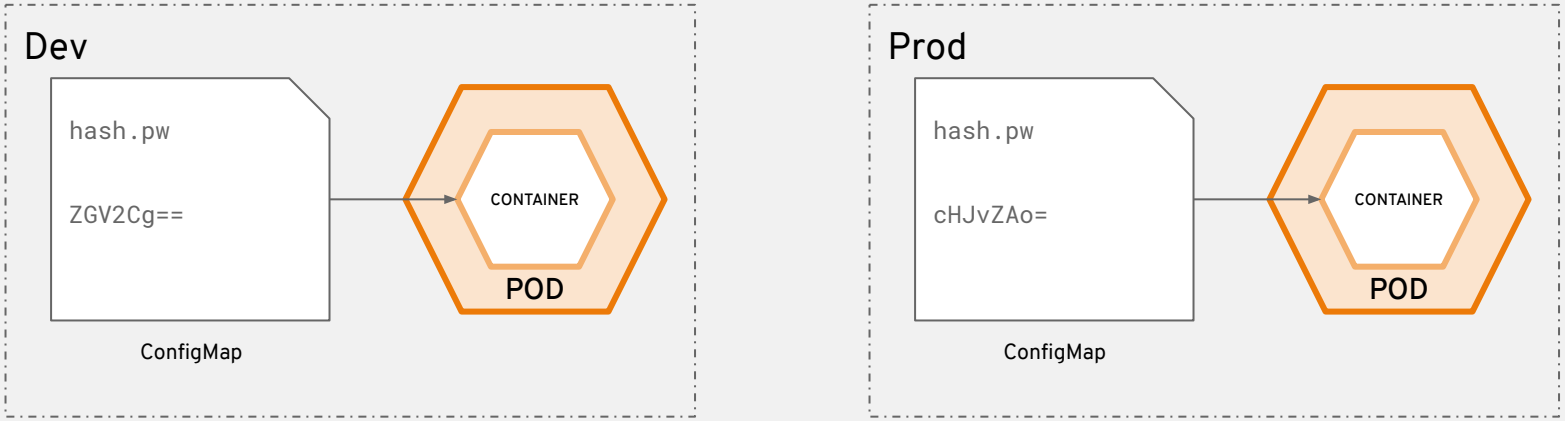
a `daemonset` ensures that all (or some) nodes run a copy of a pod



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

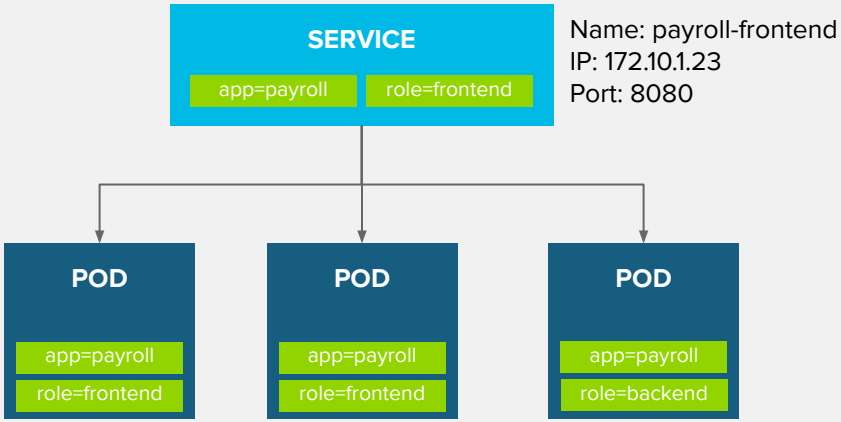


`secrets` provide a mechanism to hold sensitive information such as passwords



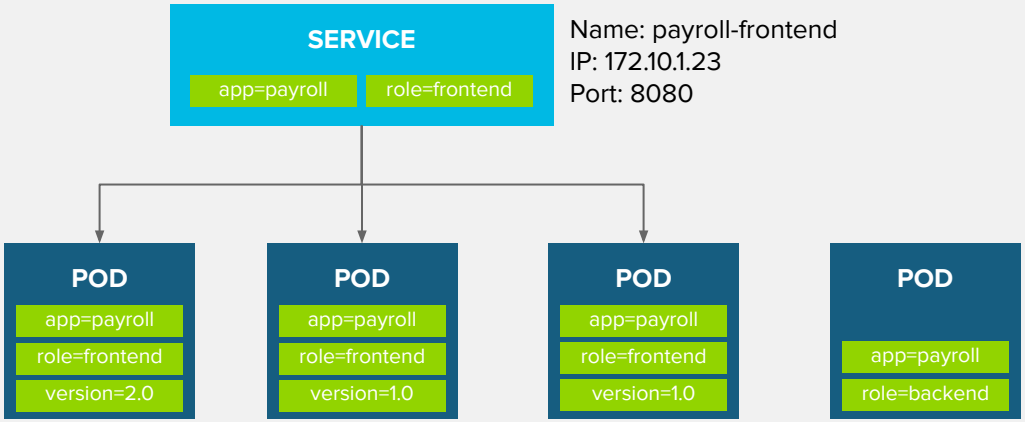
Networking

BUILT-IN SERVICE DISCOVERY & INTERNAL LOAD-BALANCING



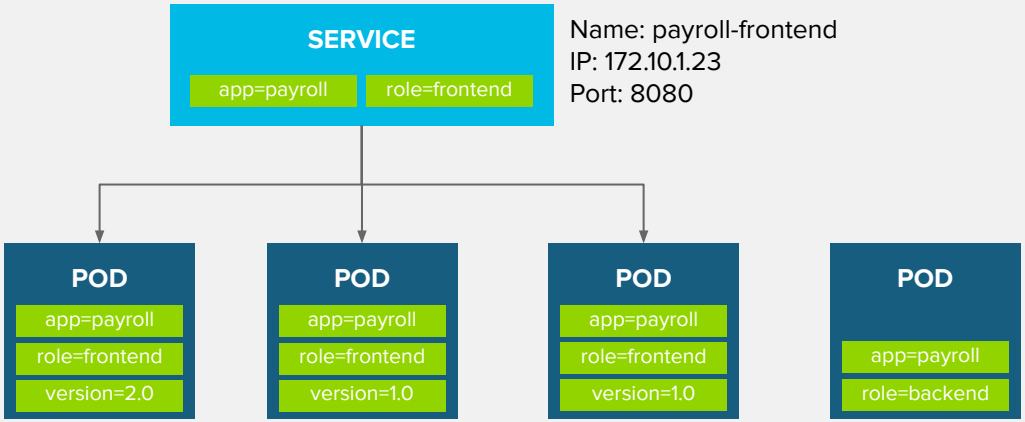
> oc describe svc/jenkins |egrep “Selector|Endpoints”
> oc get pods -l name=jenkins

BUILT-IN SERVICE DISCOVERY & INTERNAL LOAD-BALANCING



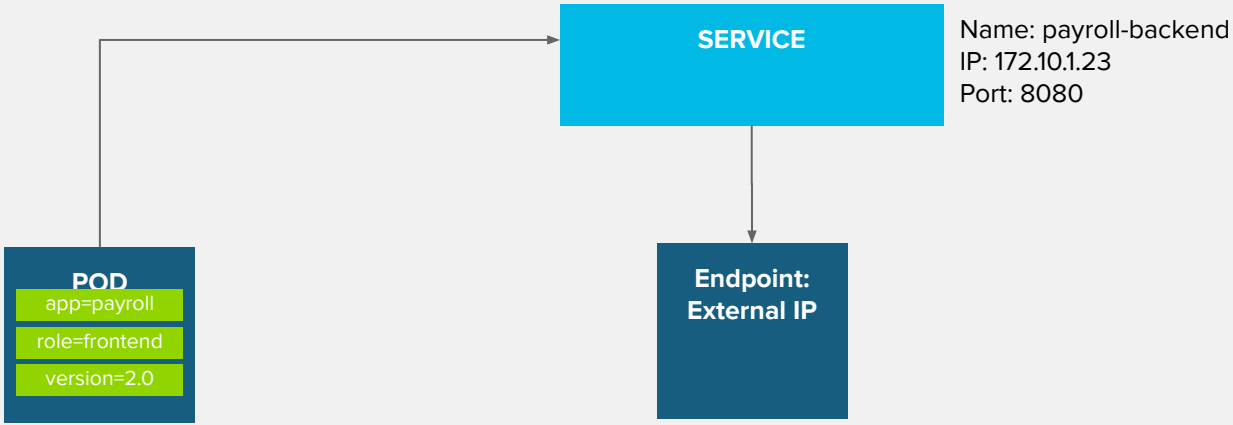
> oc describe svc/jenkins |egrep “Selector|Endpoints”
> oc get pods -l name=jenkins

BUILT-IN SERVICE DISCOVERY & INTERNAL LOAD-BALANCING



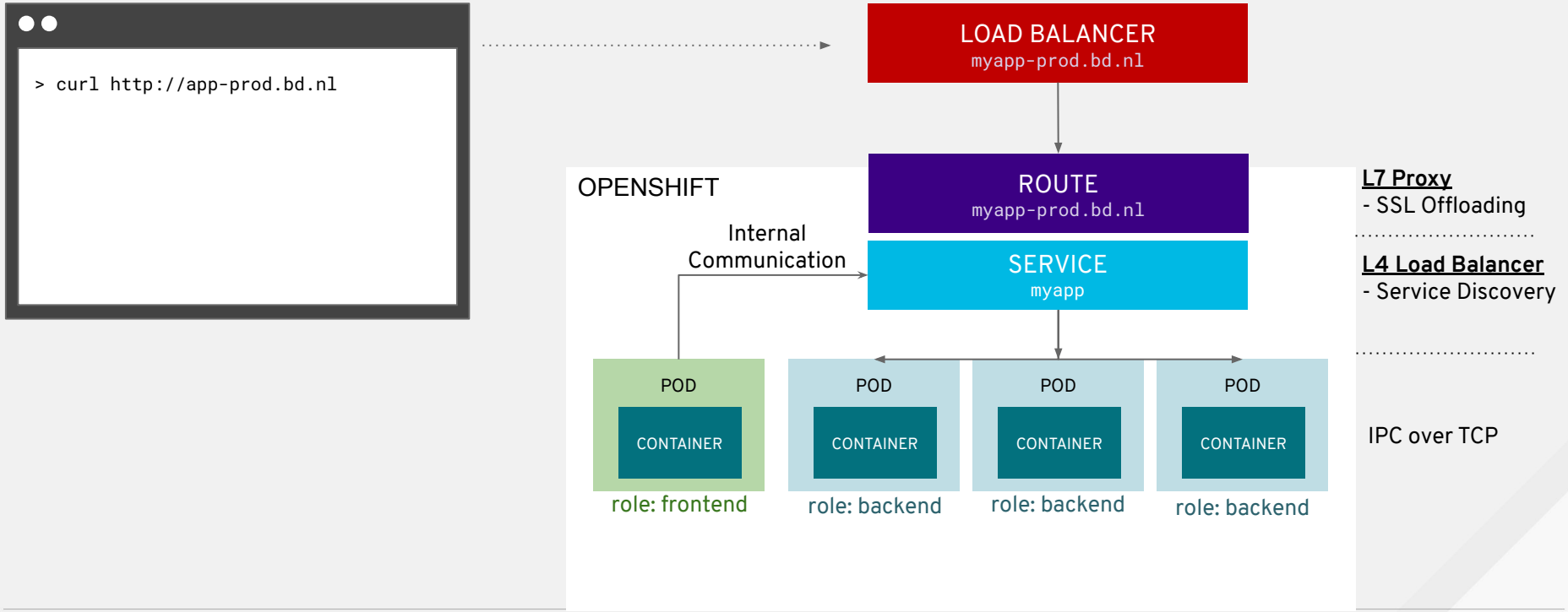
> oc describe svc/jenkins |egrep “Selector|Endpoints”
> oc get pods -l name=jenkins

EXTERNAL SERVICES



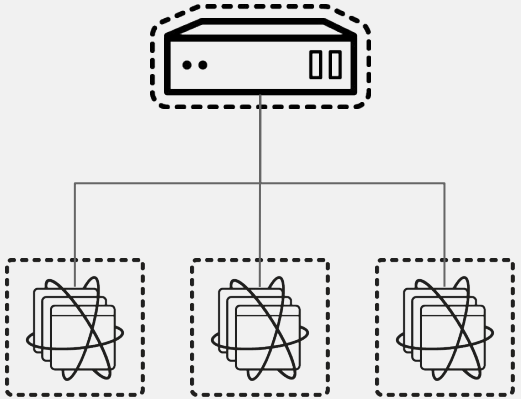
> oc describe svc/jenkins |egrep “Selector|Endpoints”
> oc get pods -l name=jenkins

ROUTING TRAFFIC



ROUTING AND EXTERNAL LOAD-BALANCING

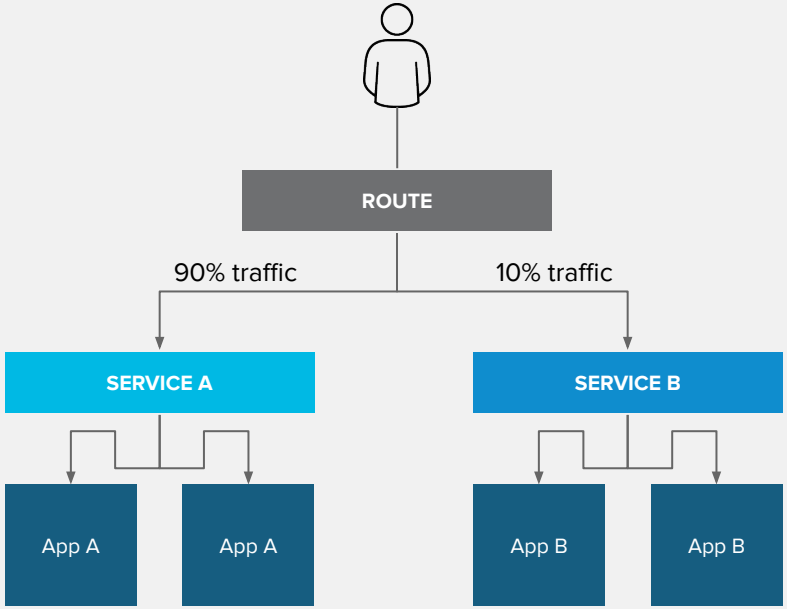
- Pluggable routing architecture
  - HAProxy Router
  - F5 Router
- Multiple-routers with traffic sharding
- Router supported protocols
  - HTTP/HTTPS
  - WebSockets
  - TLS with SNI
- Non-standard ports via cloud load-balancers, ExternalIP, and NodePort



```
> oc expose svc/jenkins --hostname jenkins-user00.openshift.eu
> oc get route
```

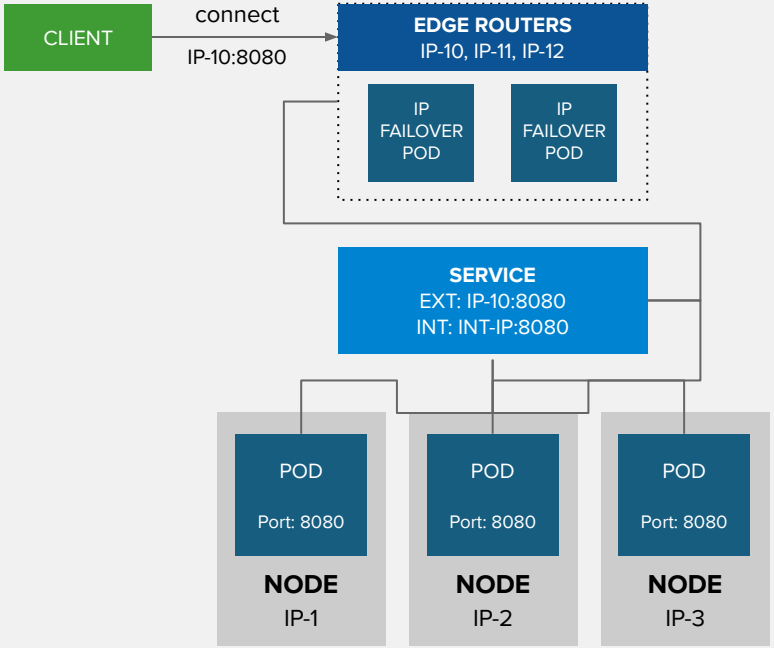
ROUTE SPLIT TRAFFIC

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



EXTERNAL TRAFFIC WITH EXTERNALIP

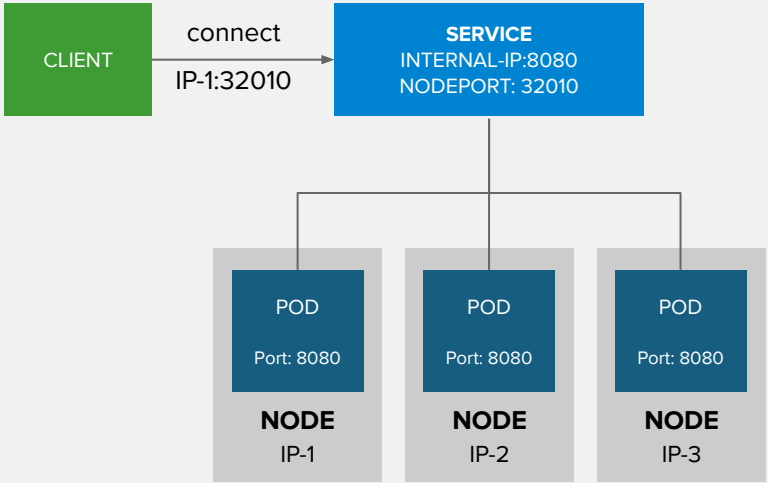
- Route external traffic to a service on any TCP/UDP port
- Available on non-cloud clusters
- External IP automatically assigned from a pre-defined pool of external IPs
- IP failover pods provide high availability for the pool of external IPs



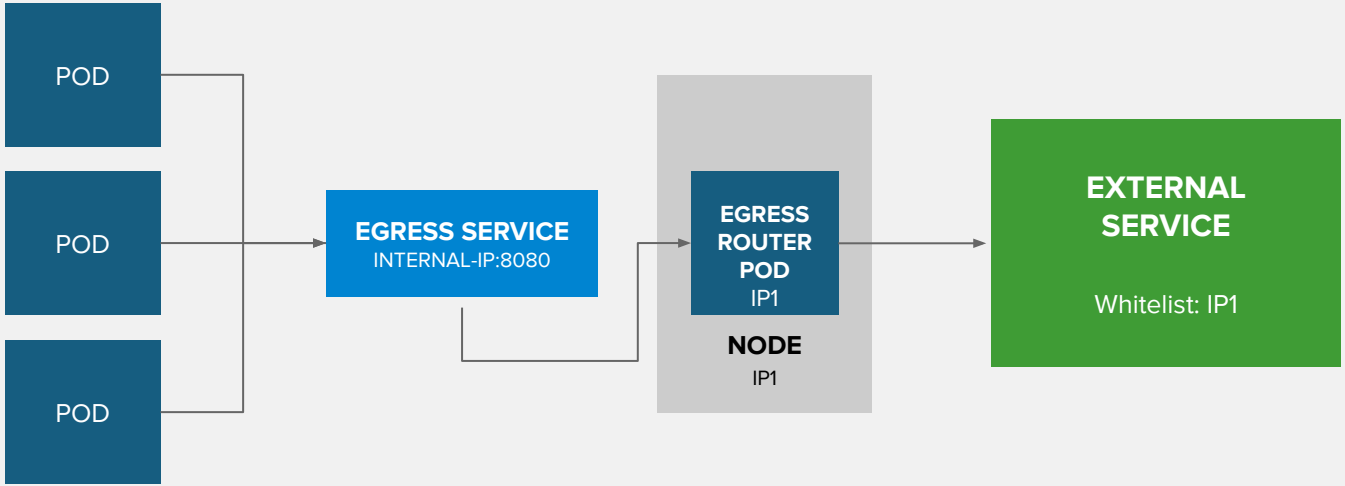
> oc expose dc/jenkins --external-ip=172.29.253.100 --name=jenkins-externalip
> oc get svc

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



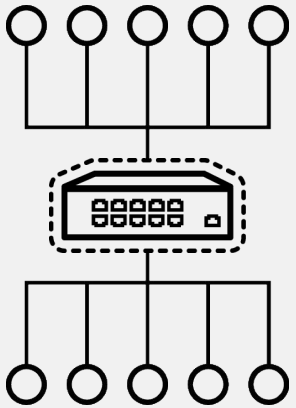
https://docs.openshift.com/container-platform/3.9/admin_guide/managing_networking.html

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)



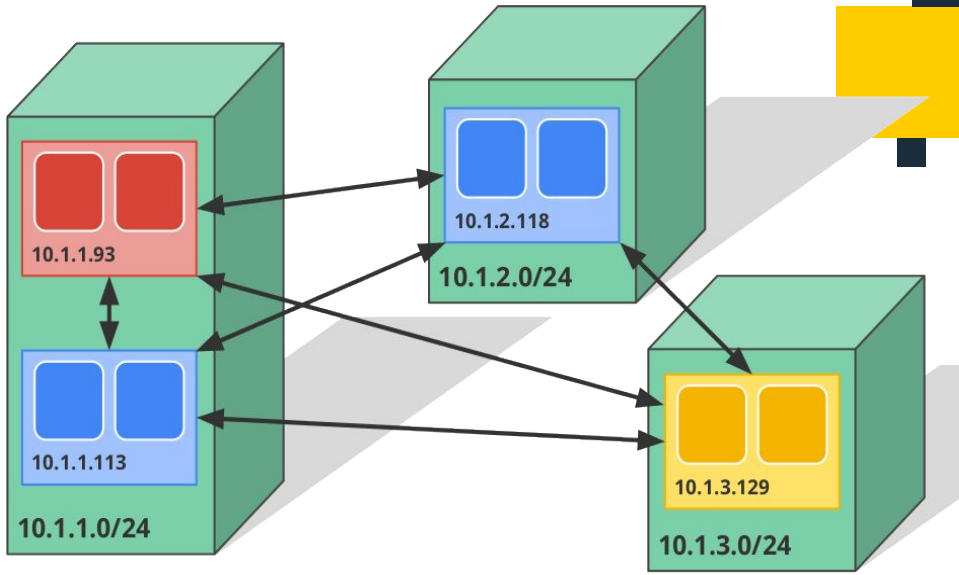
> ssh master1 grep networkPlugin /etc/origin/master/master-config.yaml

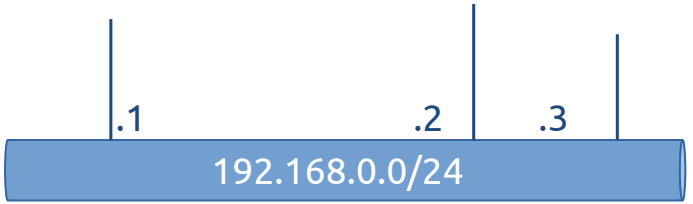
Overlay Networking

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

- Egress traffic = NAT Node's IP

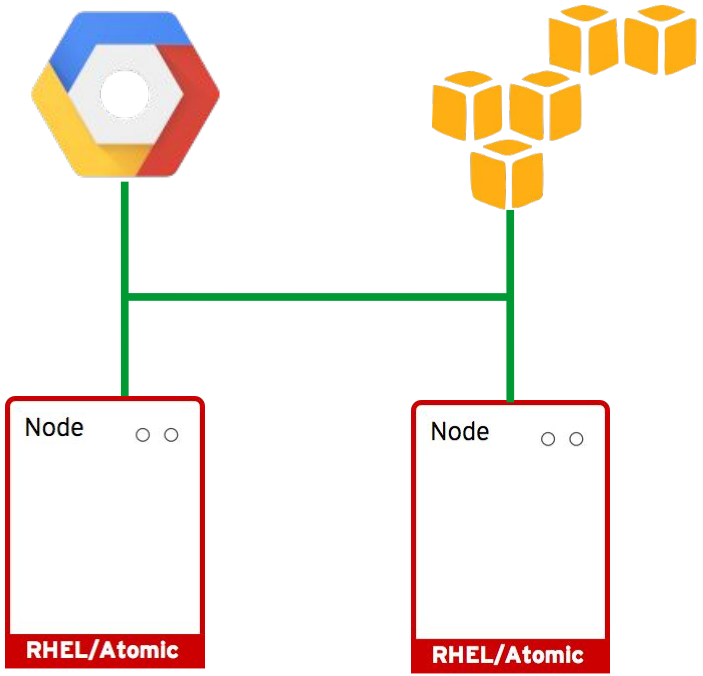
- Network Plugins:
  - Flannel, Weave, Nuage
  - OpenShift-SDN (OpenVSwitch)
  - Neutron, GCE, AWS





Underlay “infra” Network

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



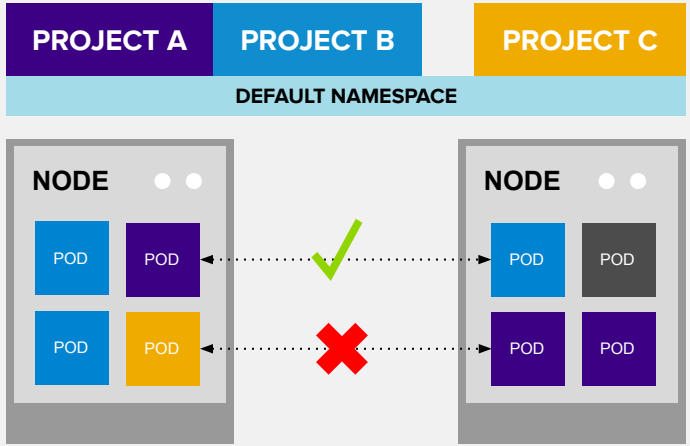
OPENSHIFT SDN

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
```

Persistent Storage

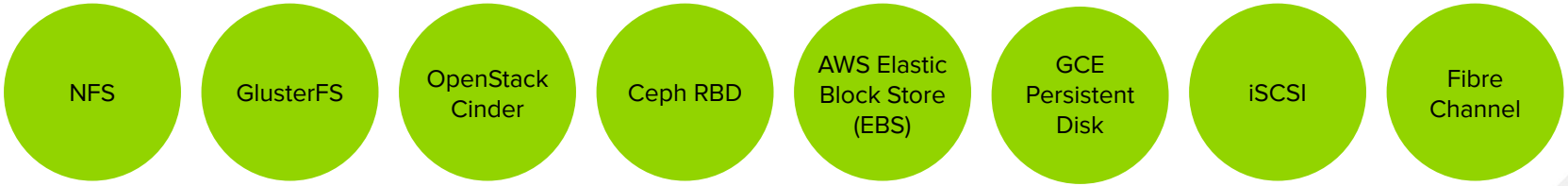
Connecting real-world storage to your containers to enable stateful applications

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)**		

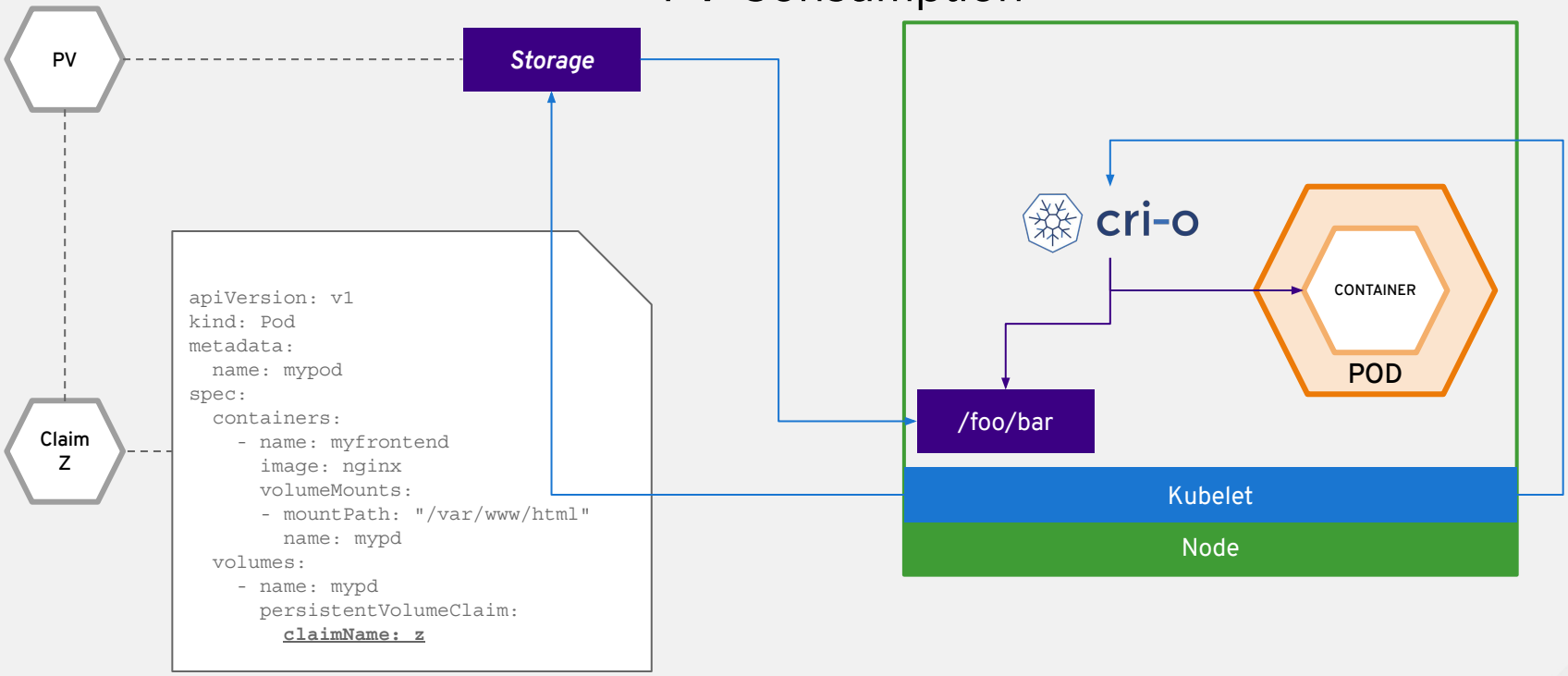
PERSISTENT STORAGE

- 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

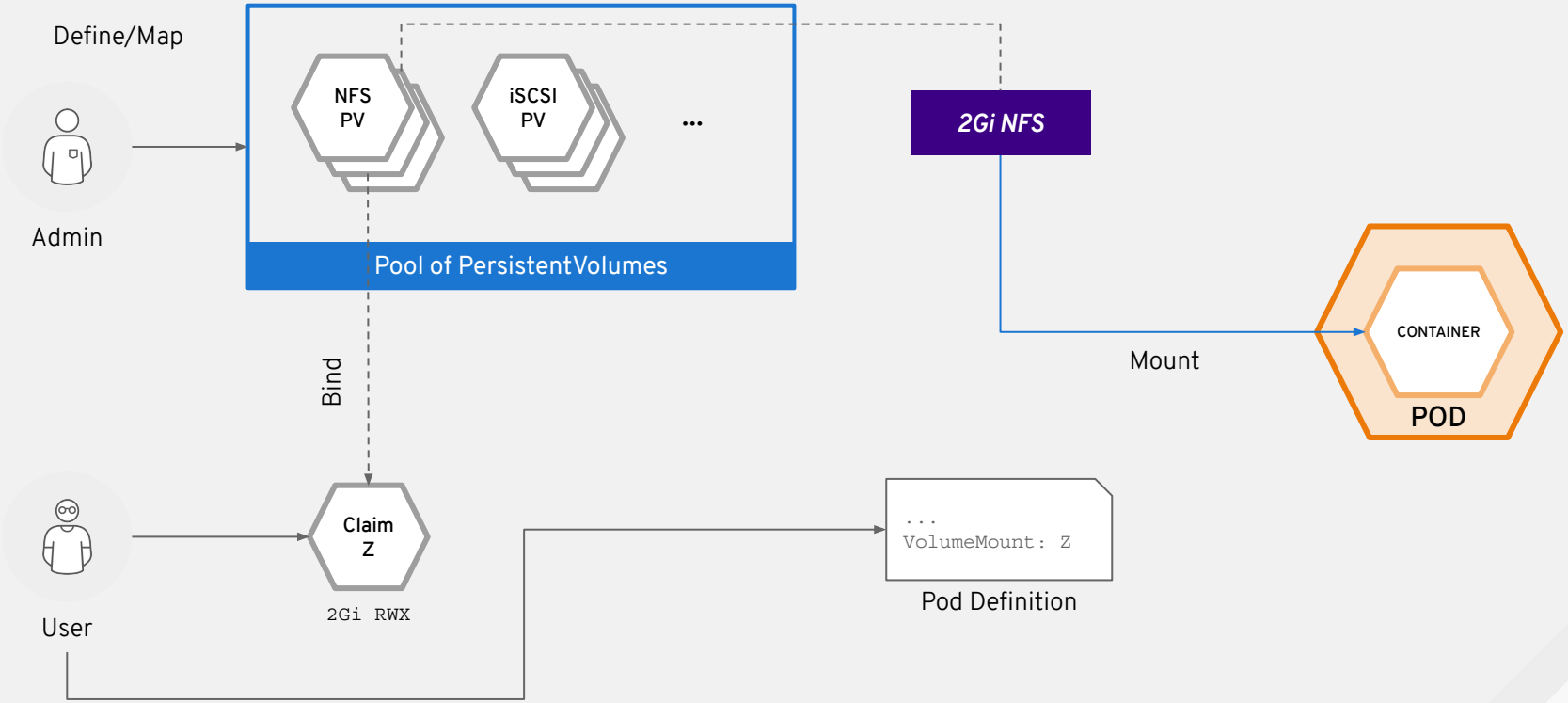


> oc get storageclass

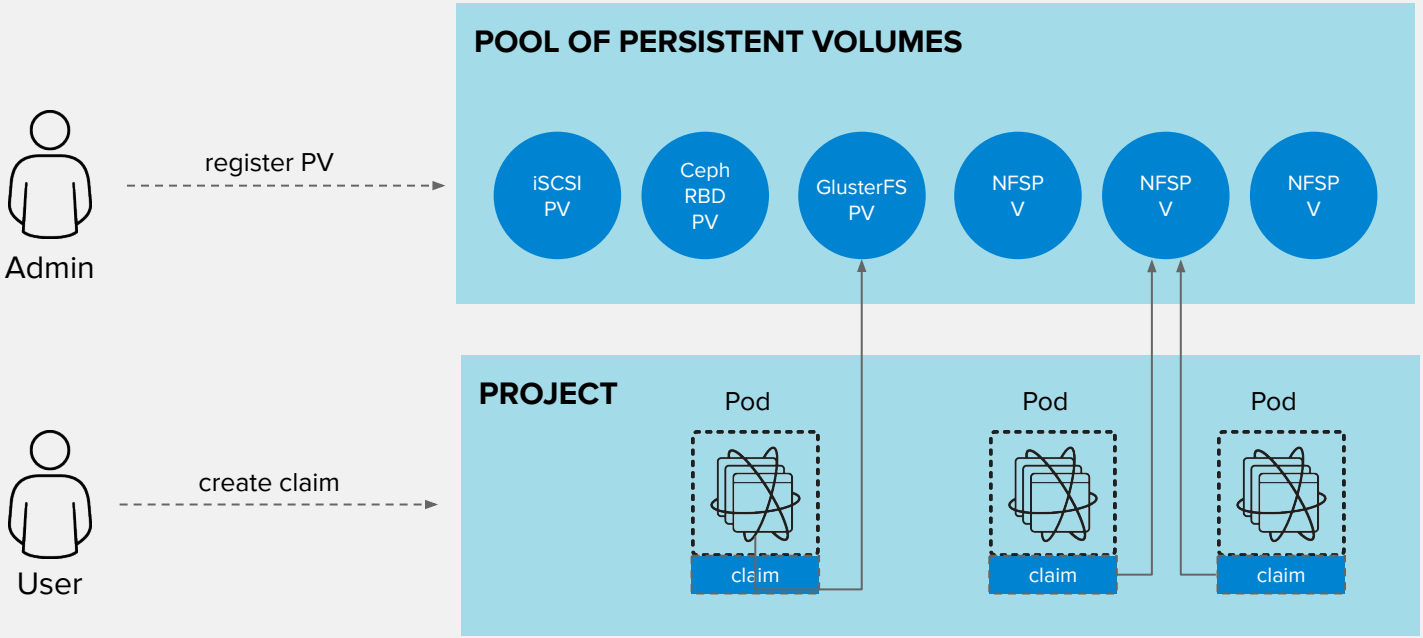
PV Consumption



Static Storage Provisioning

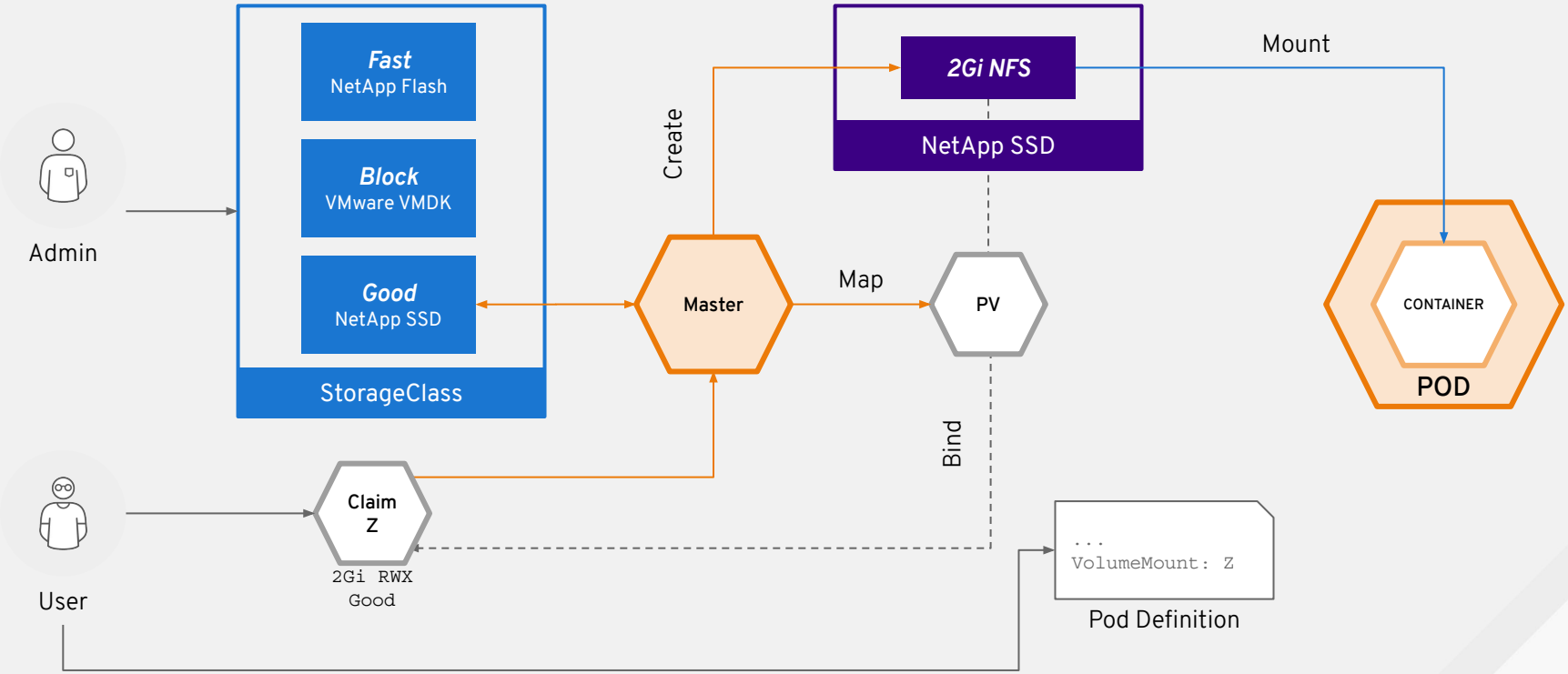


PERSISTENT STORAGE

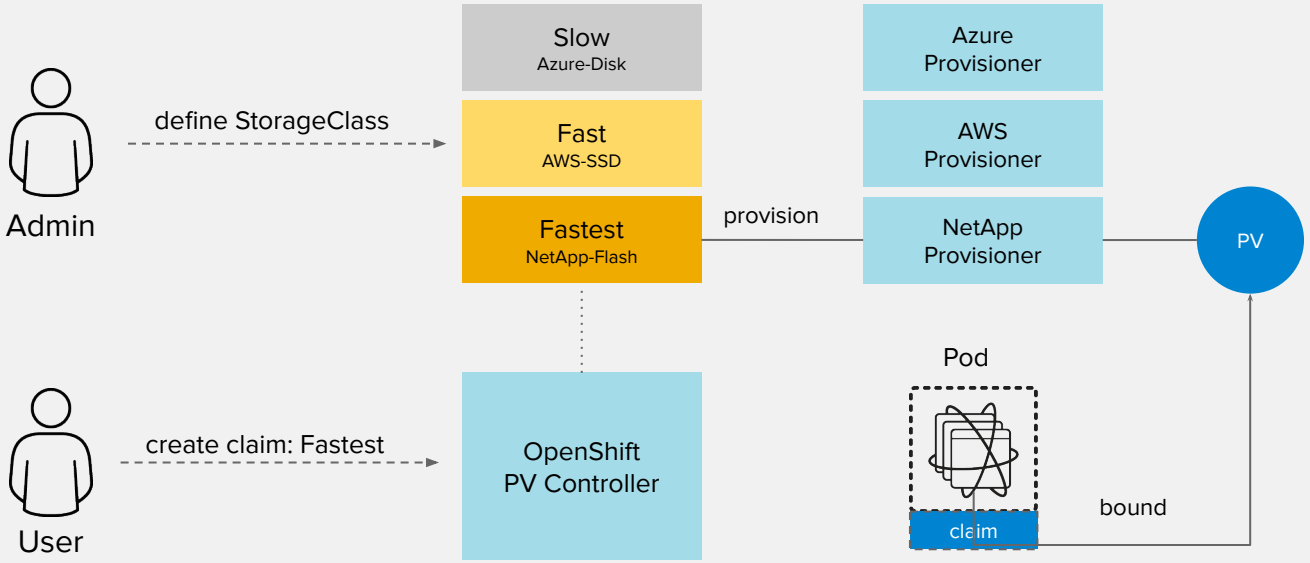


> oc get pv
> oc get pvc --all-namespaces

Dynamic Storage Provisioning



DYNAMIC VOLUME PROVISIONING



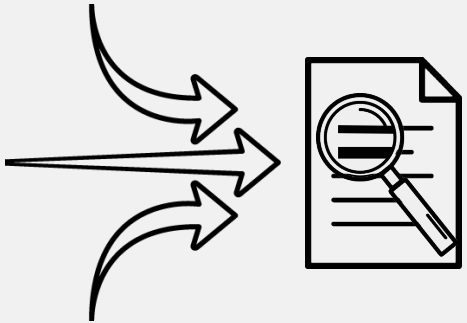
> oc get storageclass
> oc -n glusterfs get all

OpenShift Logging

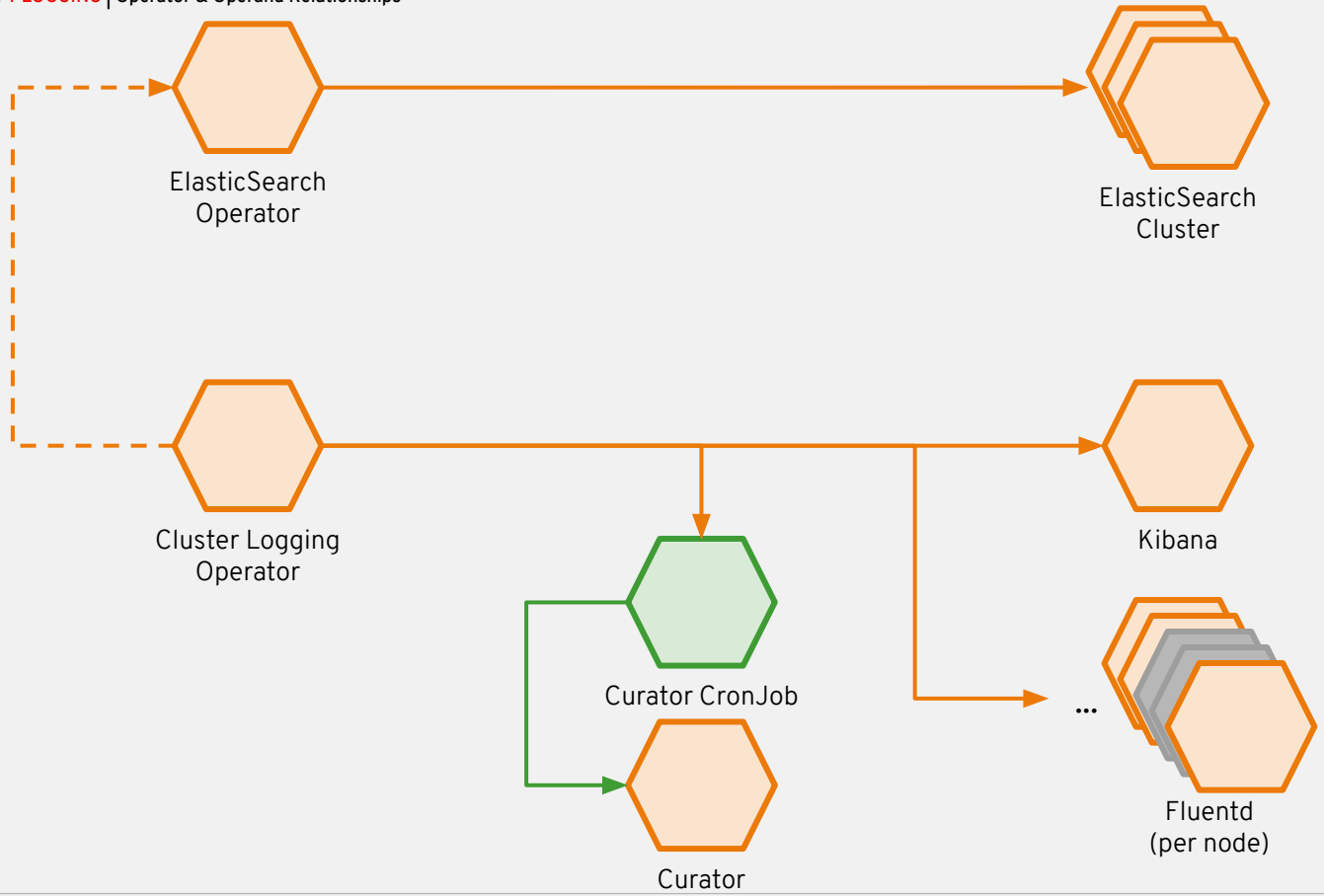
An integrated solution for exploring and corroborating application logs

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

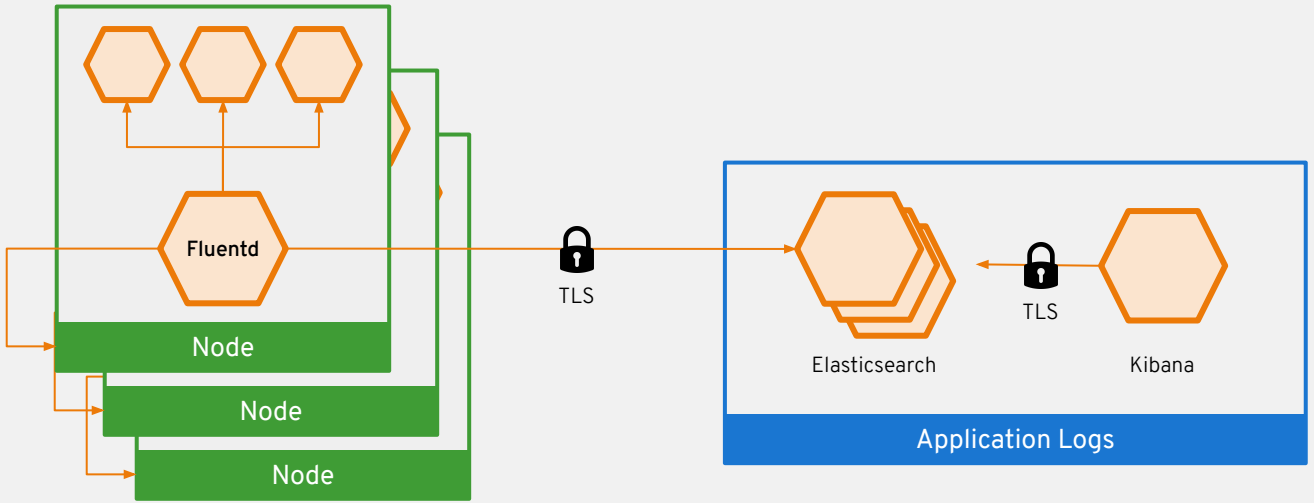


> oc -n logging get all
> http://kibana.openshift.eu

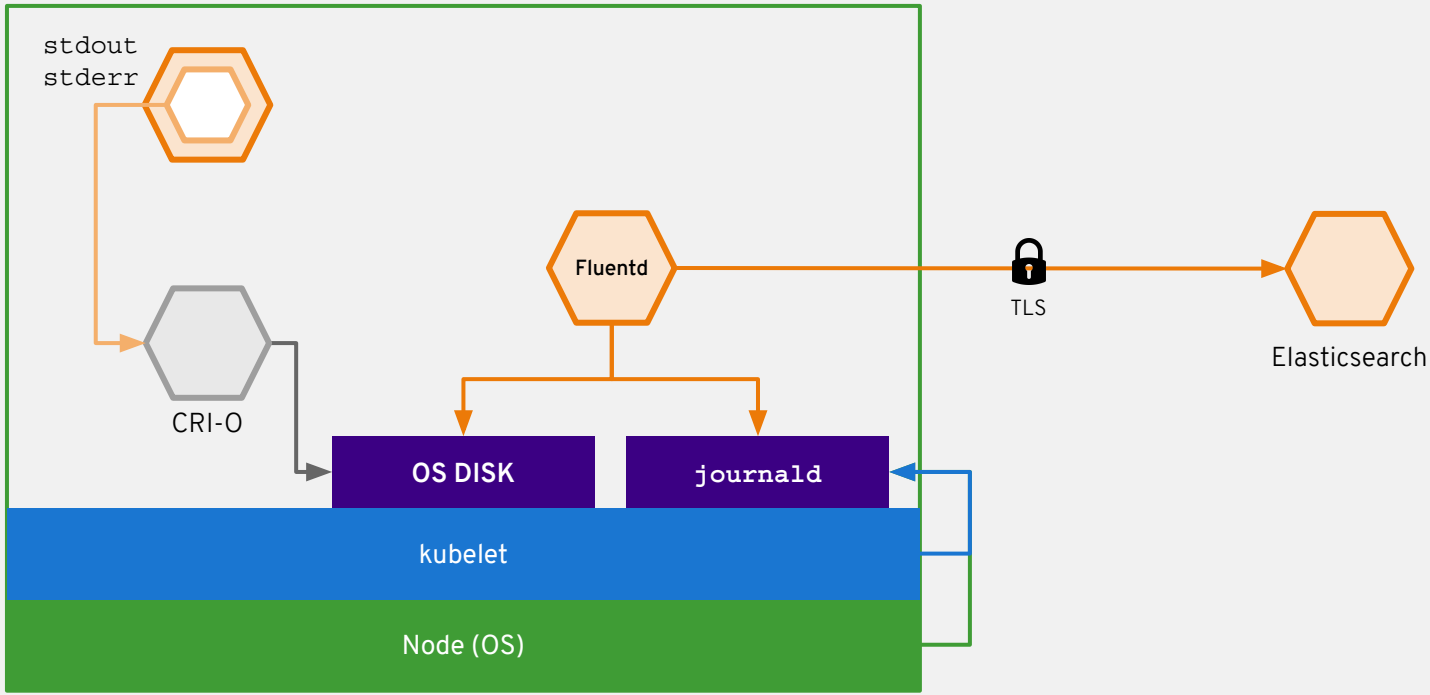


> oc -n logging get all
> http://kibana.openshift.eu

Log data flow in OpenShift



Log data flow in OpenShift



OpenShift Monitoring

An integrated cluster monitoring and alerting stack

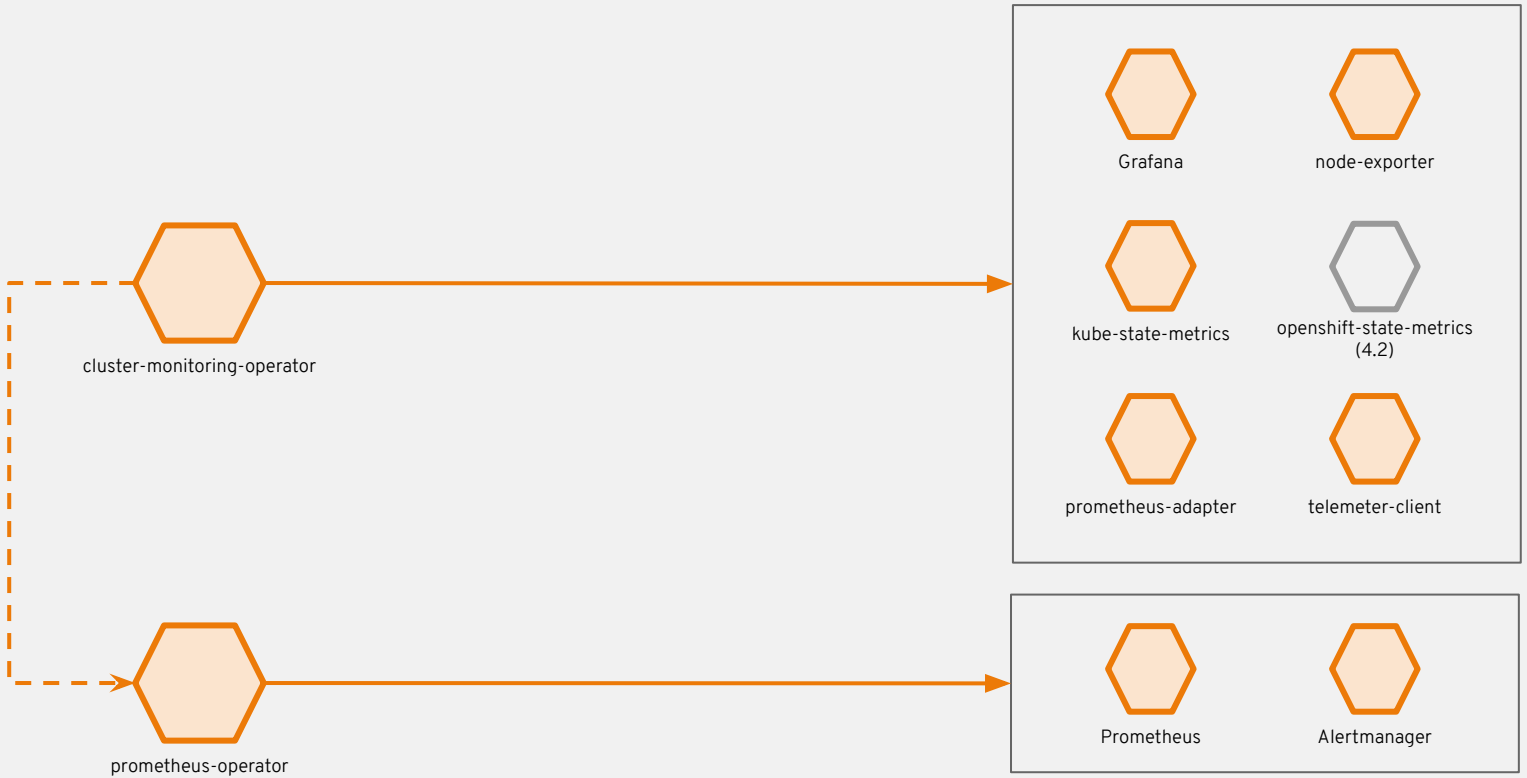
OpenShift Cluster Monitoring

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

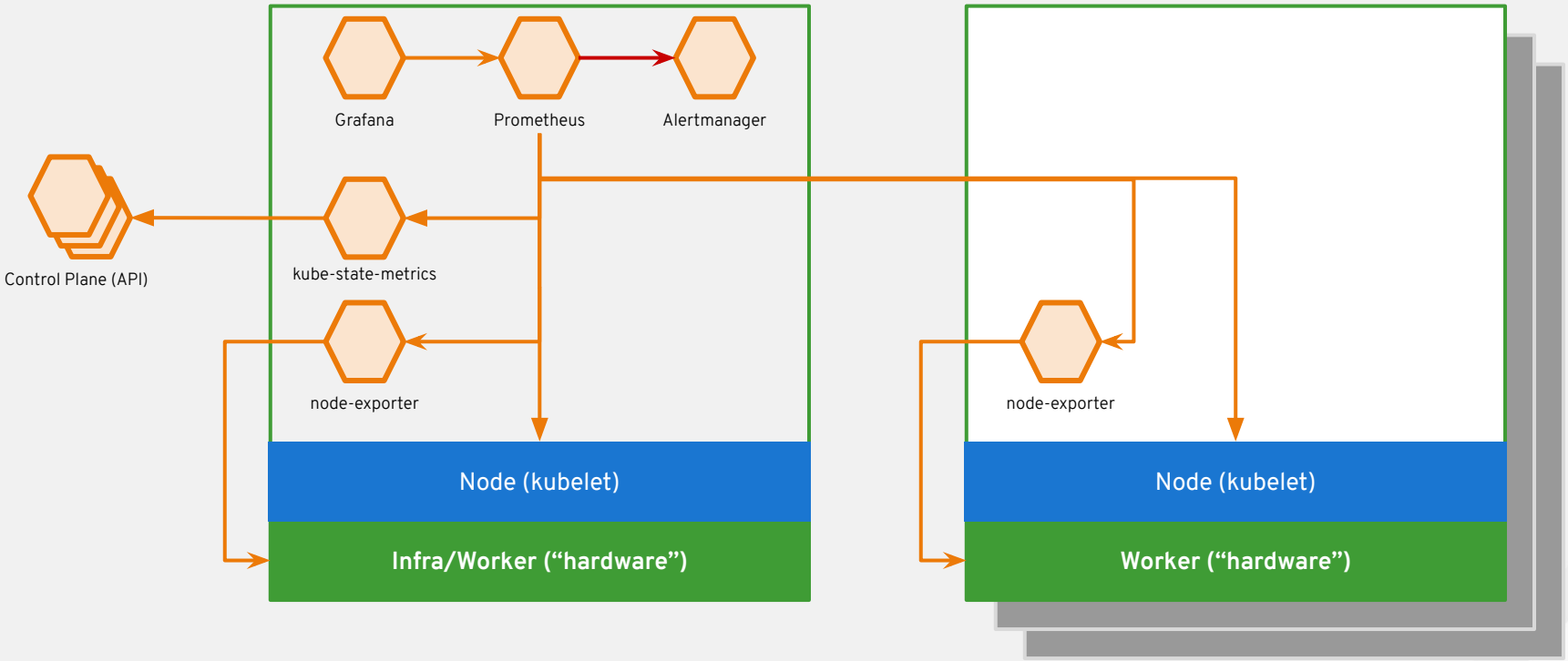
Alerting/notification via Prometheus' Alertmanager, an open-source tool that handles alerts send by Prometheus.

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 #1

AGENDA

Day 1

09:00 Overview
09:30 Architektur
10:30 Tech. deep dive
12:00 LUNCH
13:00 Networking
14:00 Logging / Metrics / Security
14:30 Storage
16:00 Operational Management
16:45 Reference Architectures

Day 2

09:00 Working with containers
11:30 CI / CD Part 1
12:30 LUNCH
13:30 CI / CD Part 2
15:00 Developer Workflow
16:00 Application Services
16:30 Q & A / next steps