# OpenShift 4

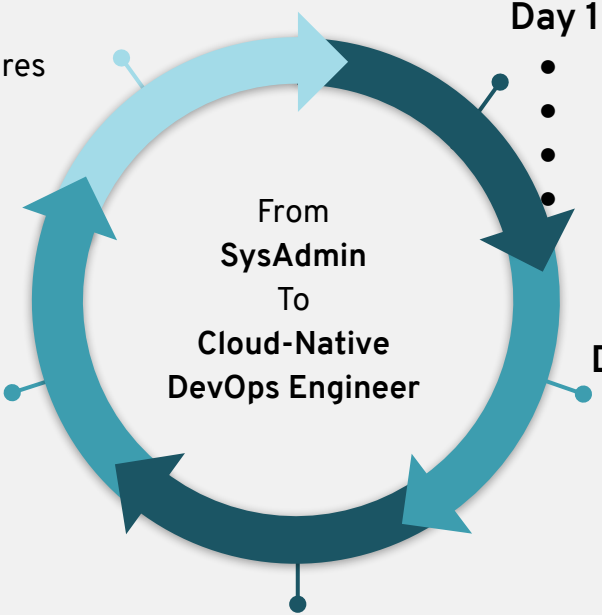
*Making introductions*

**Samuel Terburg**
**Red Hat Certified Architect**
**Cloud-Native Expert**

**2020-01-13**

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

# 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

# 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<br>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}

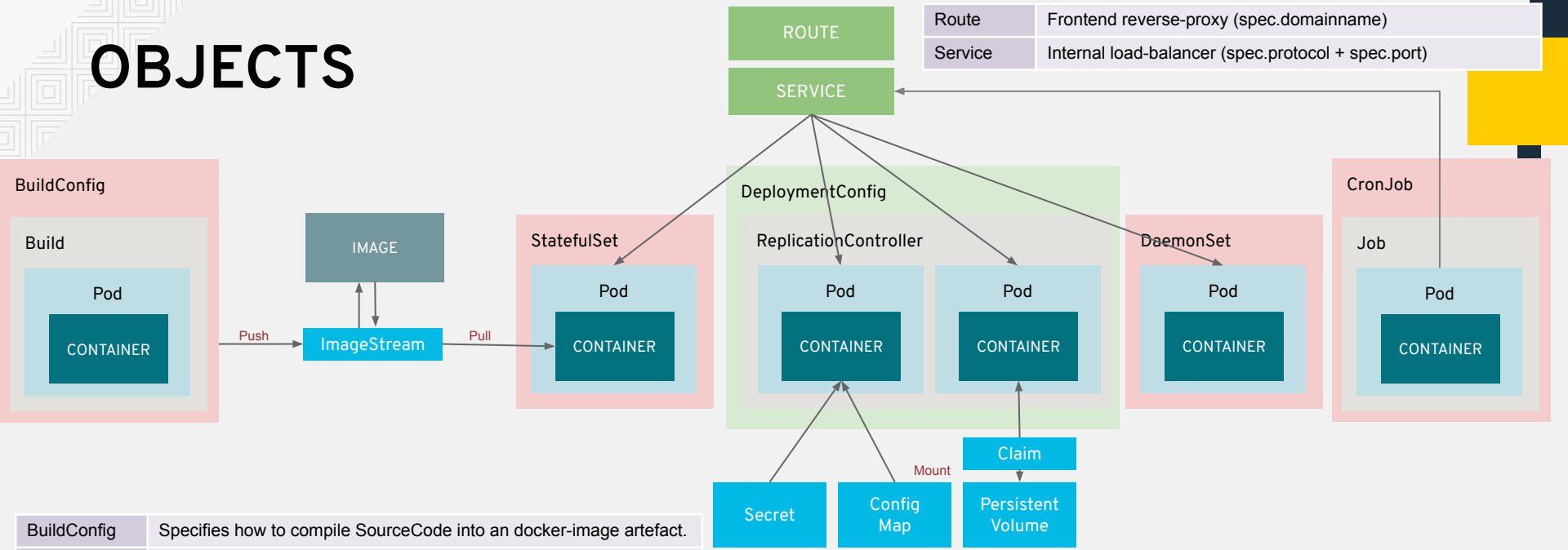
oc get pod myapp-1-abcde

| {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<br>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<br>–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. |

# Recap - day 1+2

Do you still remember?

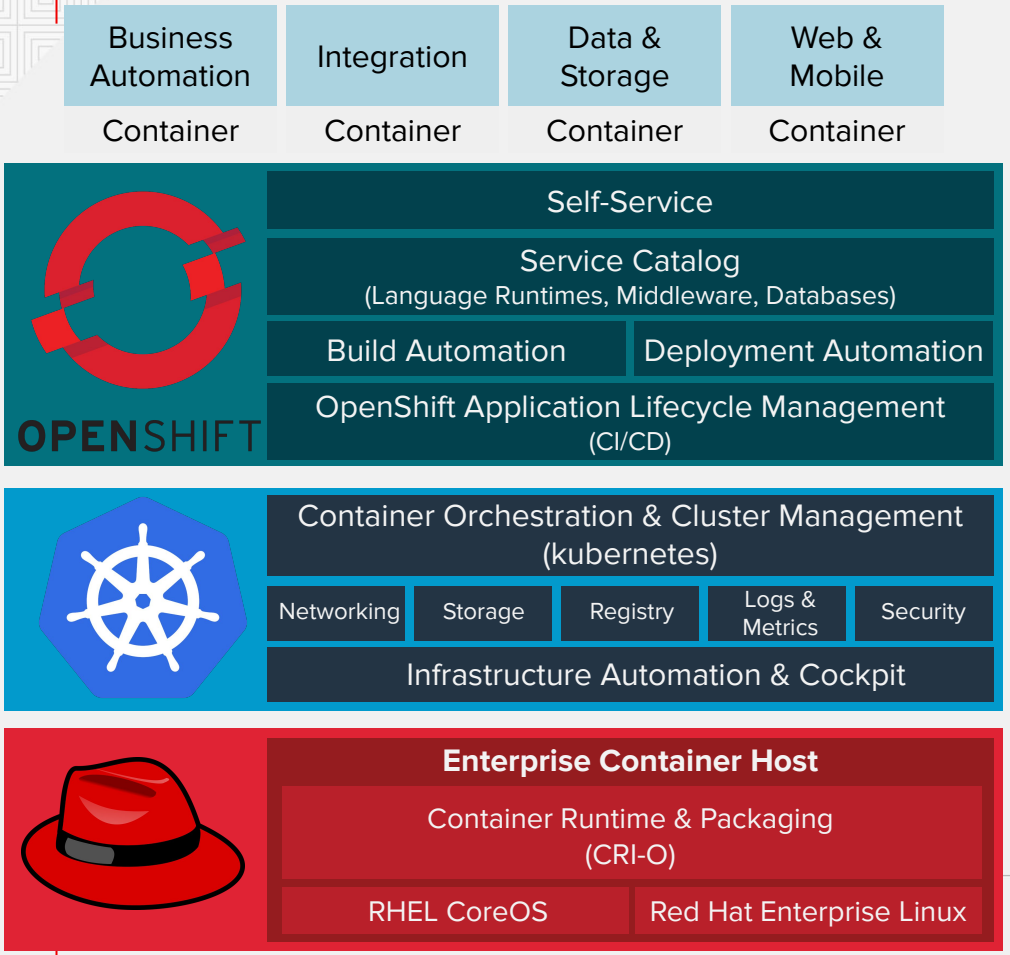
# OpenShift 4 Architecture

Technical Deep Dive

on Infrastructure

Component

# The abstraction layers




## OpenShift:

- Developer Experience

## Kubernetes:

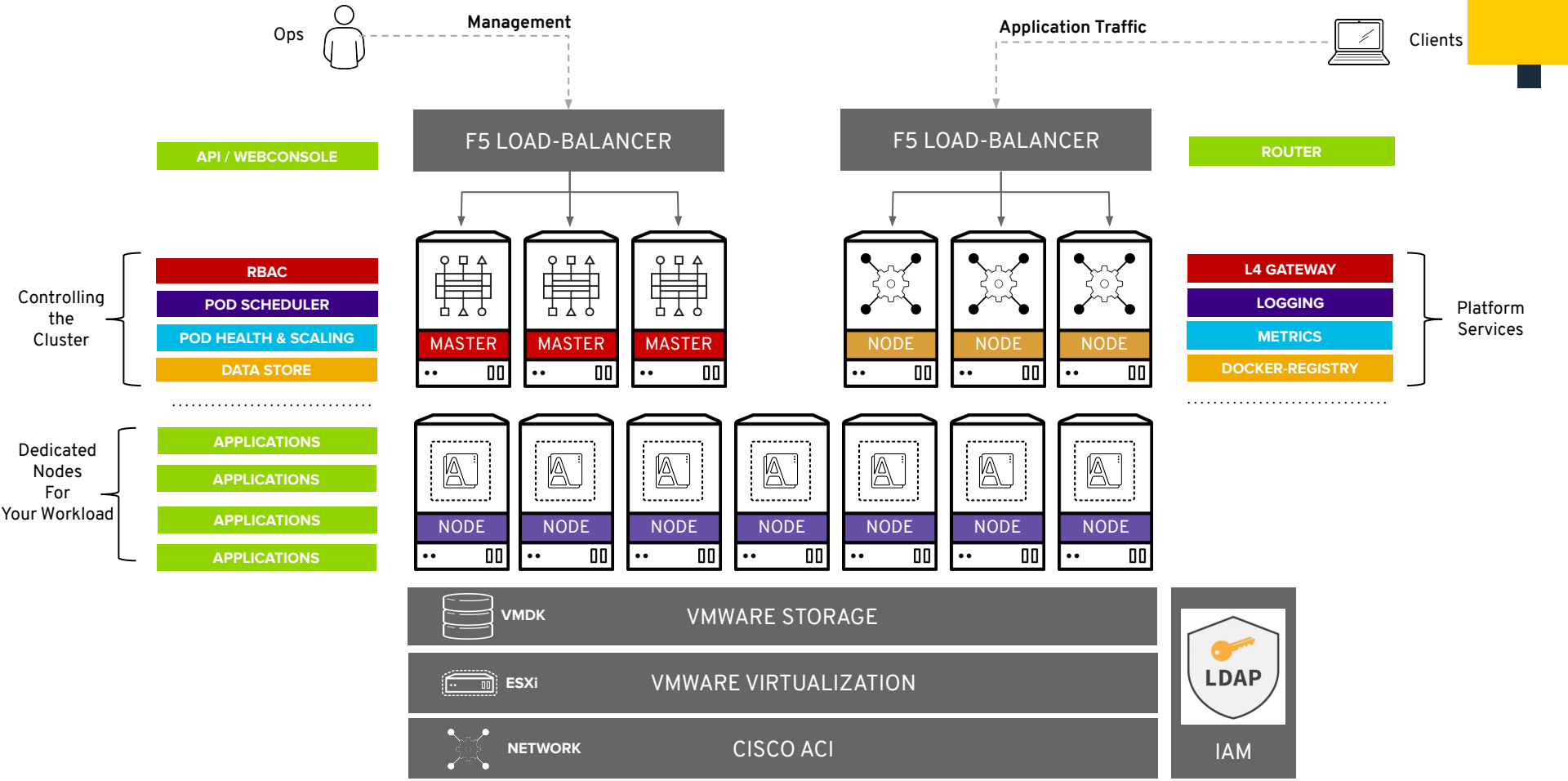
- container orchestration

## CoreOS:

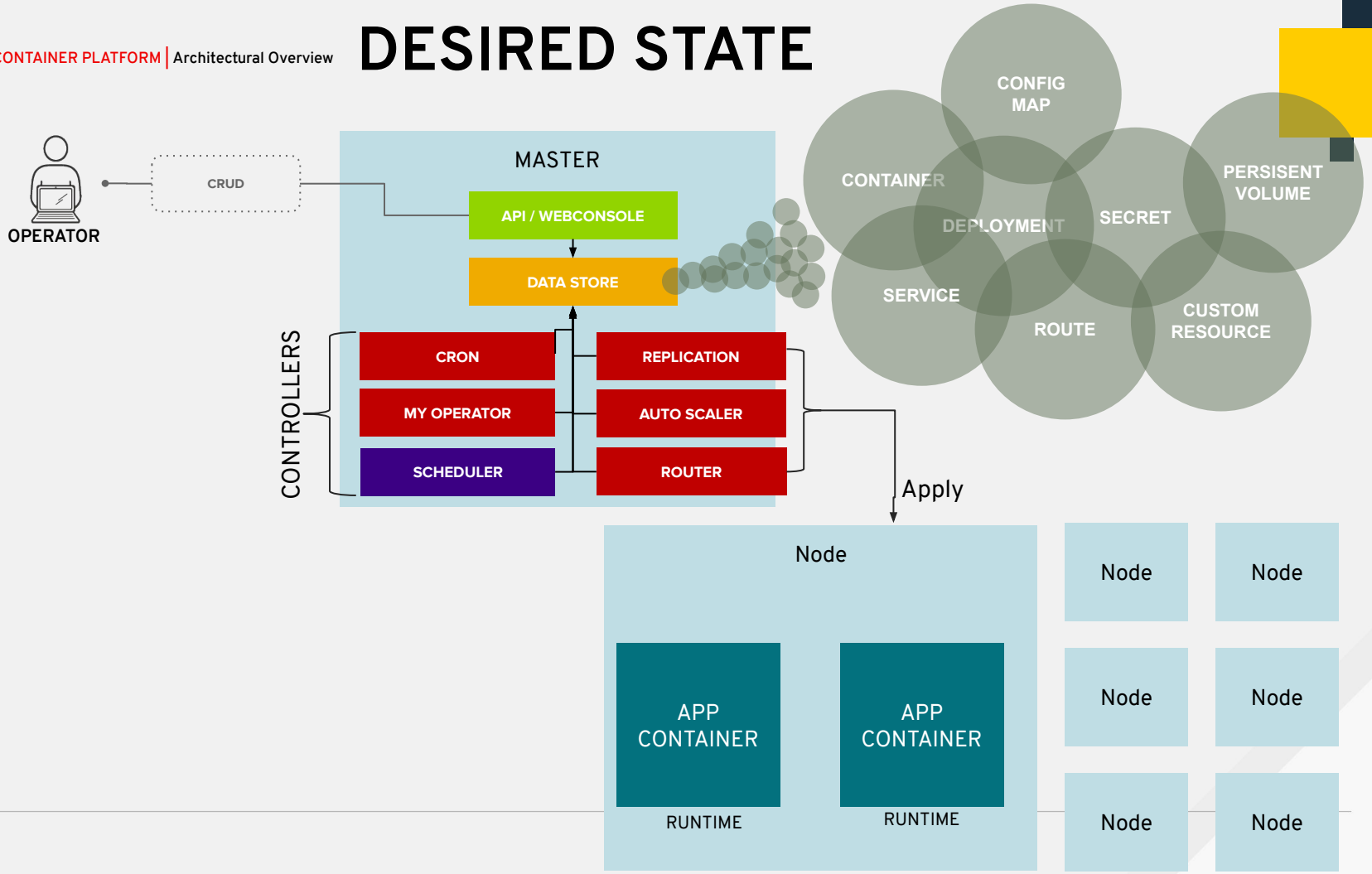
- Immutable Infrastructure

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

# PLATFORM ARCHITECTURE



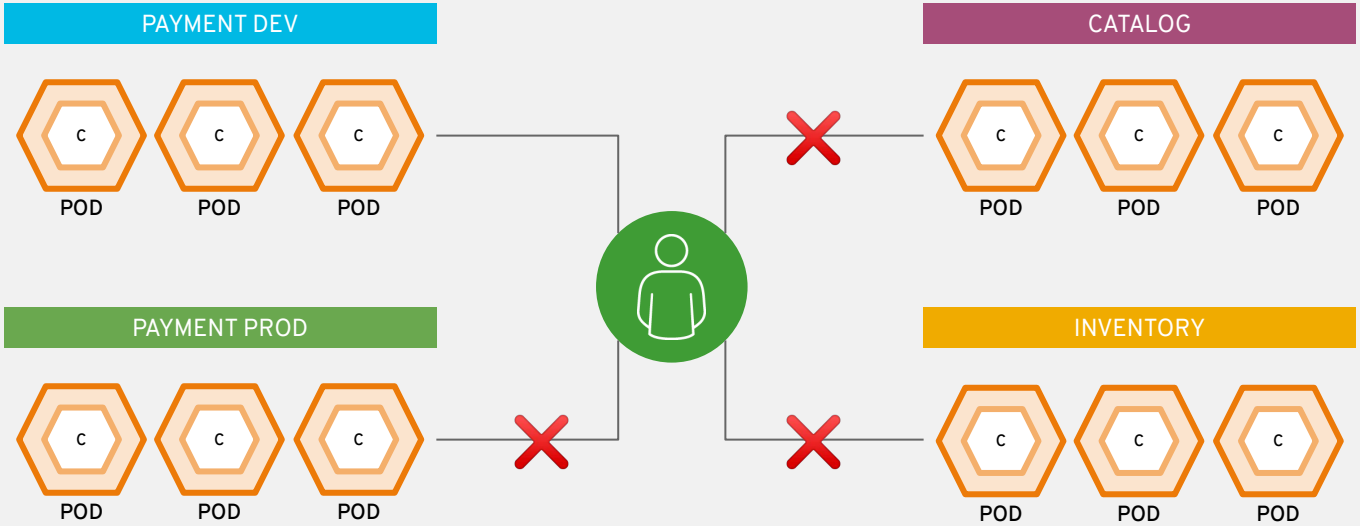
# DESIRED STATE



# Kubernetes Resource Definitions

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

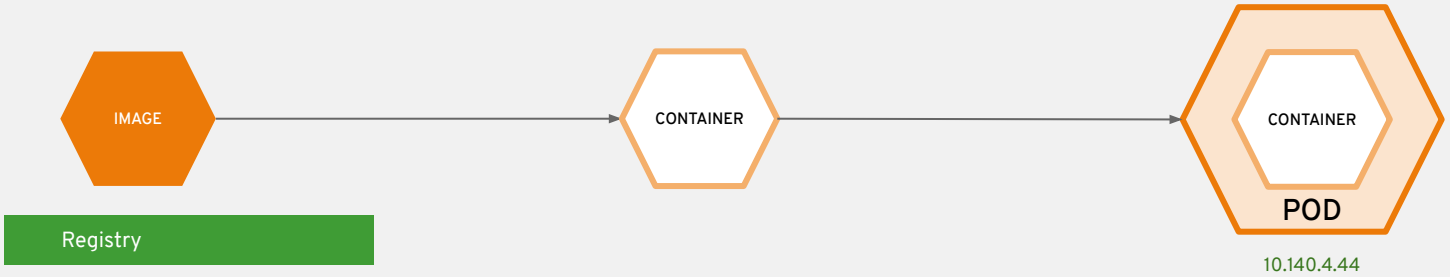
# projects isolate apps across environments, teams, groups and departments






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

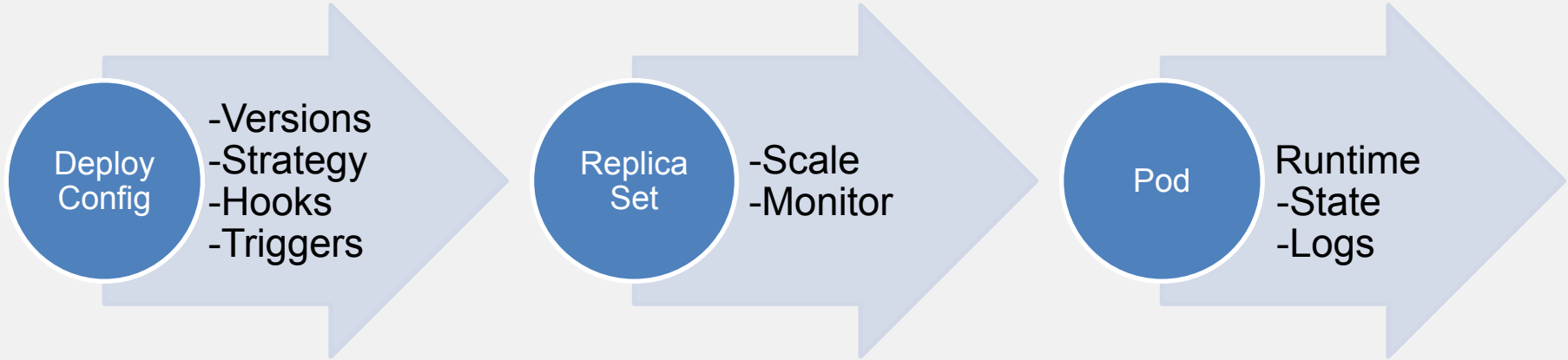
# It all starts with an image






> oc -n openshift get imagestreams
> oc -n openshift describe imagestream httpd
> oc -n <workshop> import-image docker.io/library/httpd

# Deployment Process




- MyJBossApp

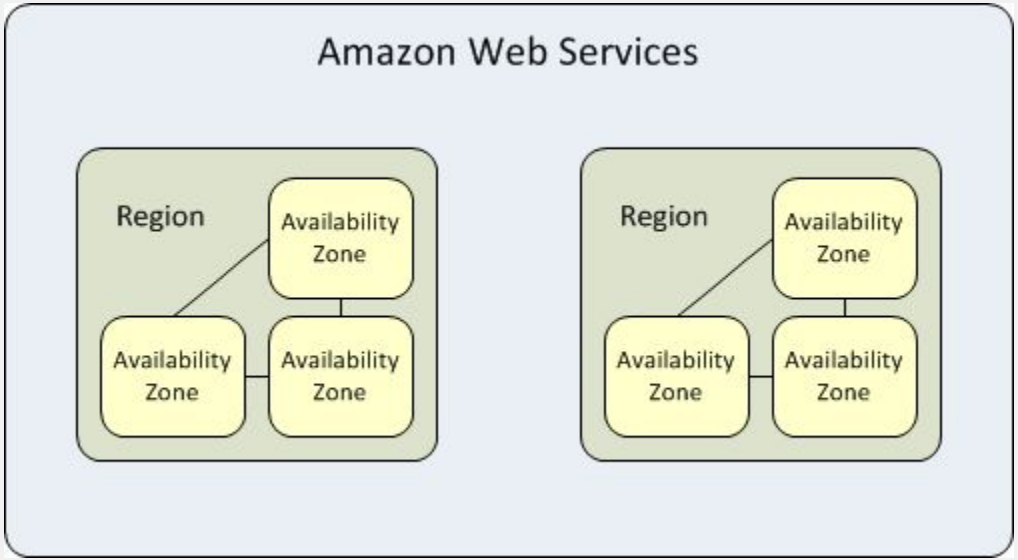
- MyJBossApp-v1 (2x)
- MyJBossApp-v2 (4x)

- 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/
```

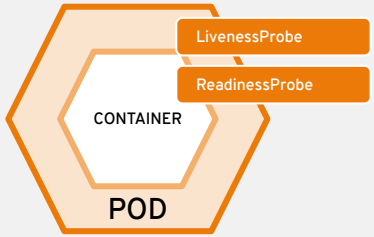
# PLACEMENT BY POLICY
## Pod Affinity rules




Preferred vs Required
Affinity vs Anti-Affinity
- Nodes
- Services
- Persistent Volumes

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

# Health Checks



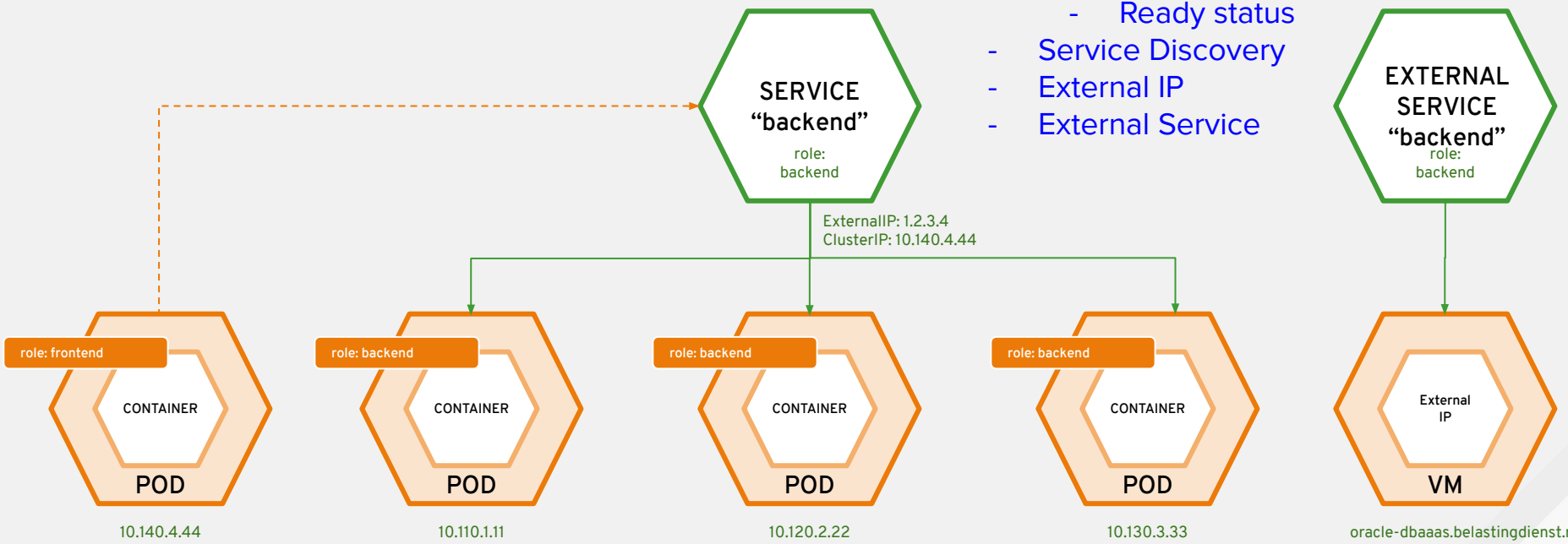

- LivenessProbe
- ReadinessProbe

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
```

# services provide internal load-balancing

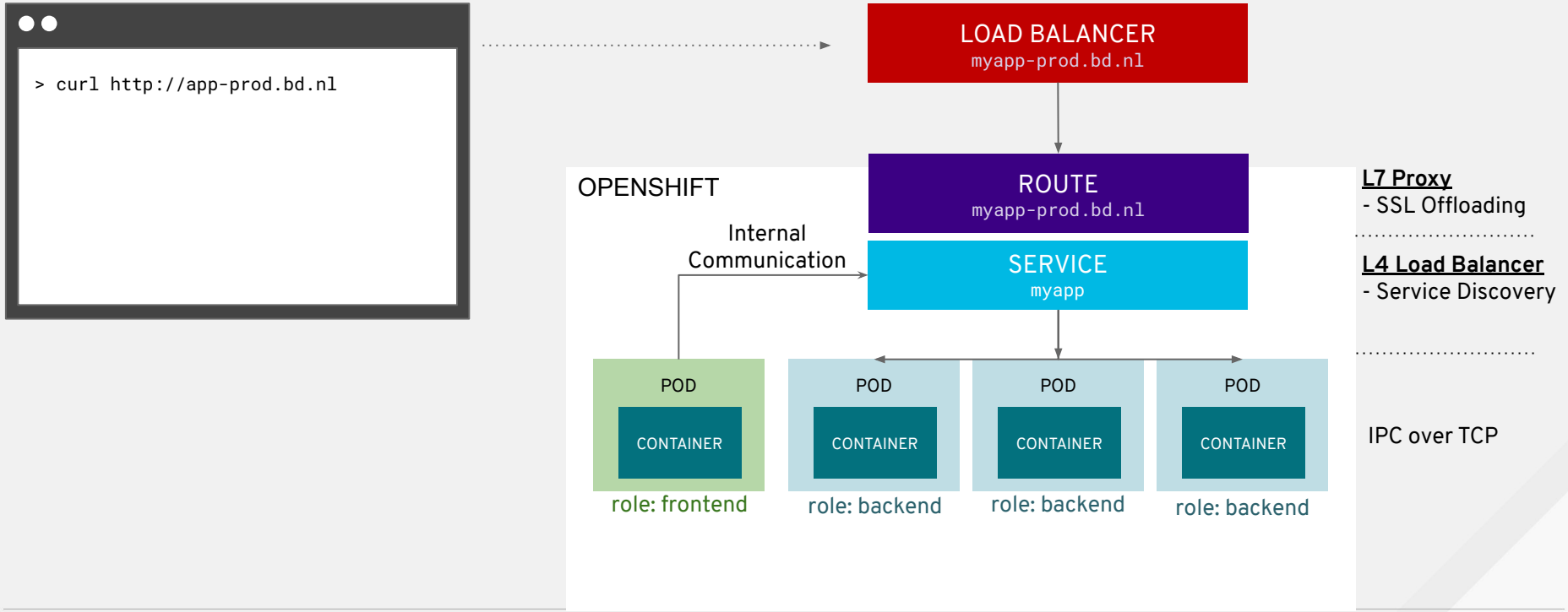
- Load-Balancing
  - Ready status
- Service Discovery
- External IP
- External Service






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

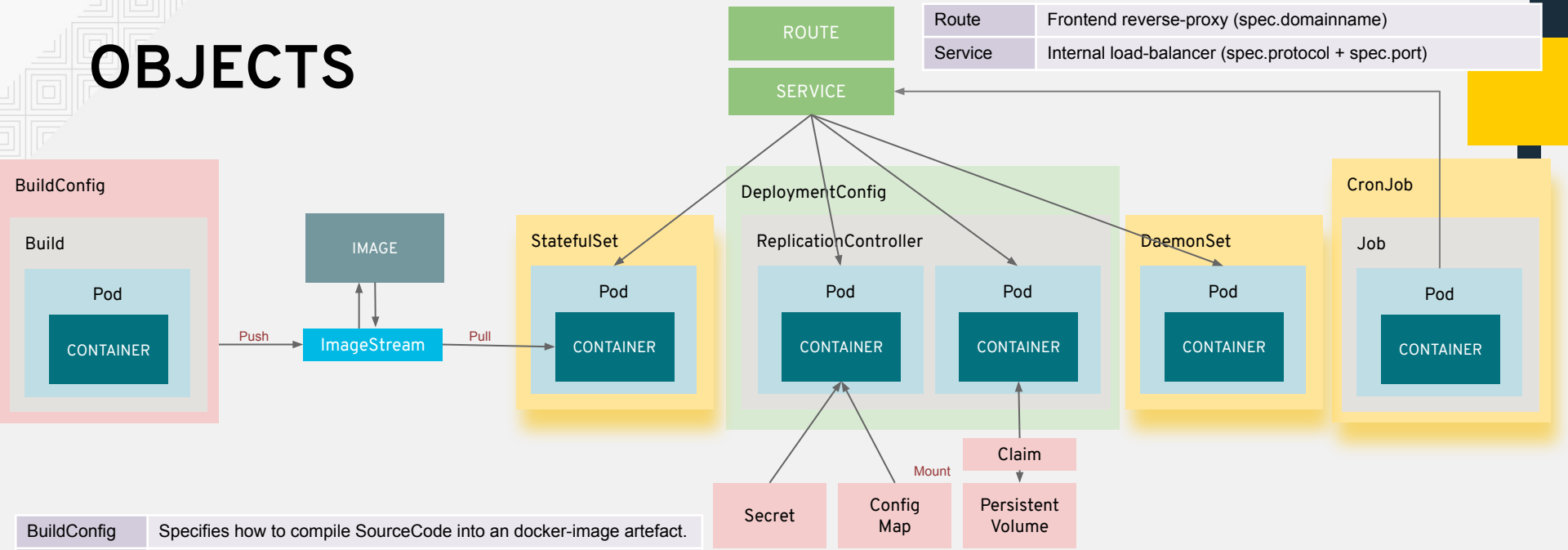
# ROUTING TRAFFIC



# Day 3

Mount external data sources into your container.

# OBJECTS



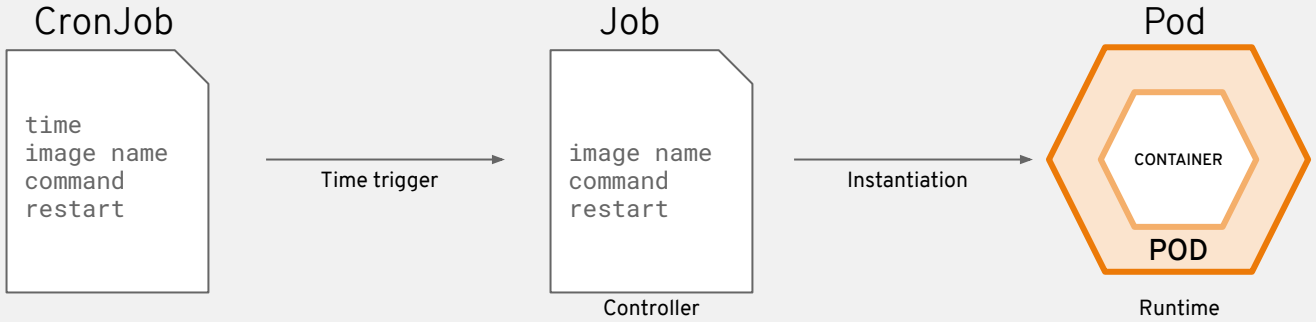

| | |
|---|---|
| Route | Frontend reverse-proxy (spec.domainname) |
| Service | Internal load-balancer (spec.protocol + spec.port) |

| | |
|---|---|
| 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. |

# `CronJobs` run short lived pods at a specified time interval



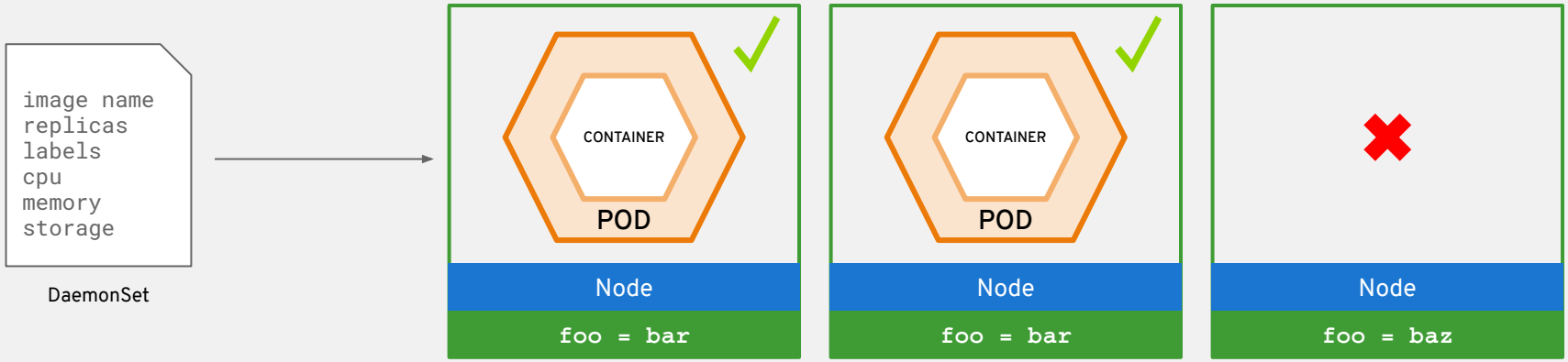



```
> oc create cronjob myjob --image=httpd --schedule="*/10 * * * *" --restart=OnFailure
> oc create job --from=cronjob/myjob
> oc patch cronjob/myjob -p '{"spec": {"jobTemplate": {"spec": {"template": {"spec": {"containers": [ {"name": "myjob", "command": ["/bin/bash","-c","echo hello world"] } ] }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

# CronJob

```
> oc create cronjob myjob --image=httpd --schedule="*/10 * * * *"
--restart=OnFailure
> oc edit cronjob/myjob
jobTemplate:
  spec:
    template:
      spec:
        containers:
        - name: myjob
          command:
          - /bin/bash
          - -c
          - echo hello world

> oc create job --from=cronjob/myjob
> oc get pods
> oc logs <pod-name>
```

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



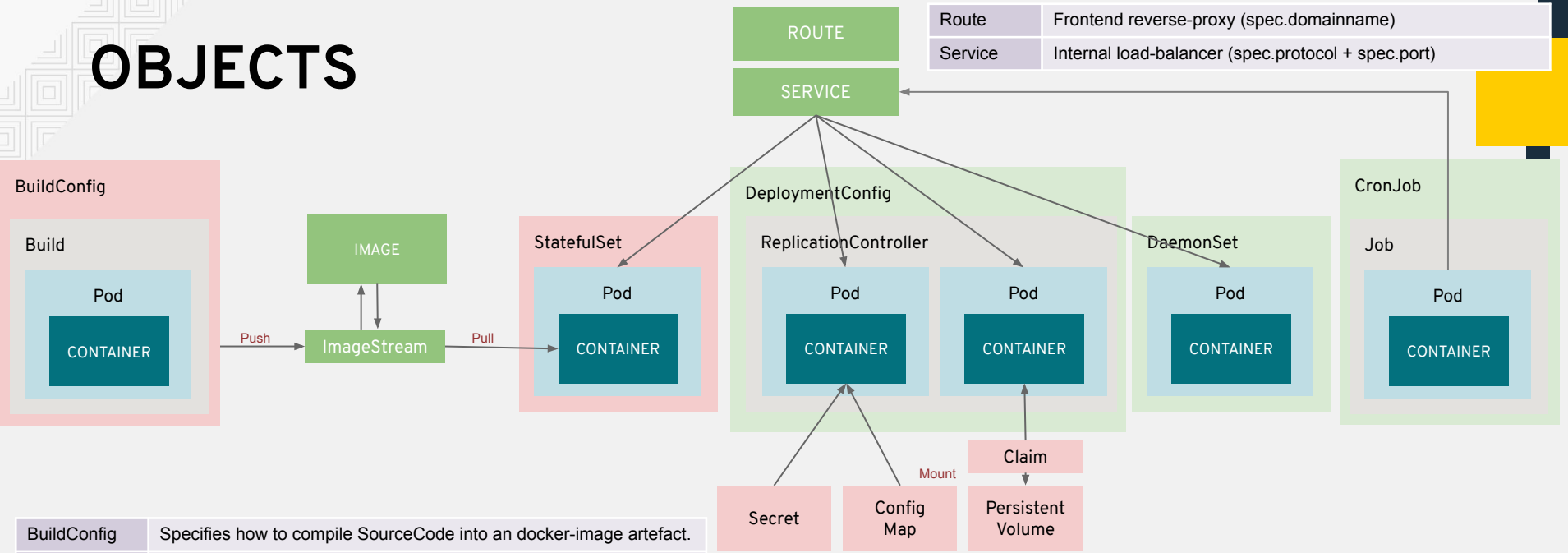



> oc get -o yaml --export dc/backend |sed 's/DeploymentConfig/DaemonSet/g' |oc create -f -

# DaemonSet

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

# 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[]) |

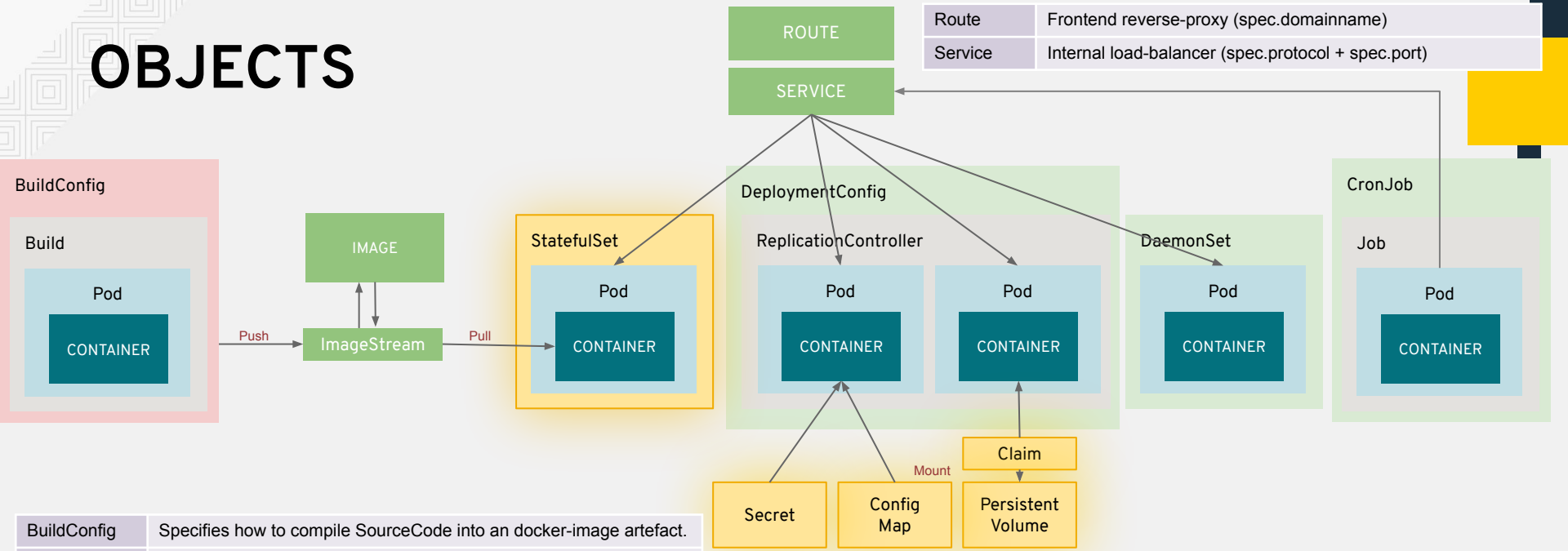
| | |
|---|---|
| DeploymentConfig | Specifies how to deploy your application (rolloutStrategy + imageChangeTriggers) |
| ReplicaSet | Monitors the Pods and restarts if needed (spec.healthcheck + spec.replicas) |
| Pod | A grouping of tightly-coupled containers (spec.containers[] + spec.node) |
| Container | 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. |

# Where is my Data!

Mount external data sources into your container.

# OBJECTS

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

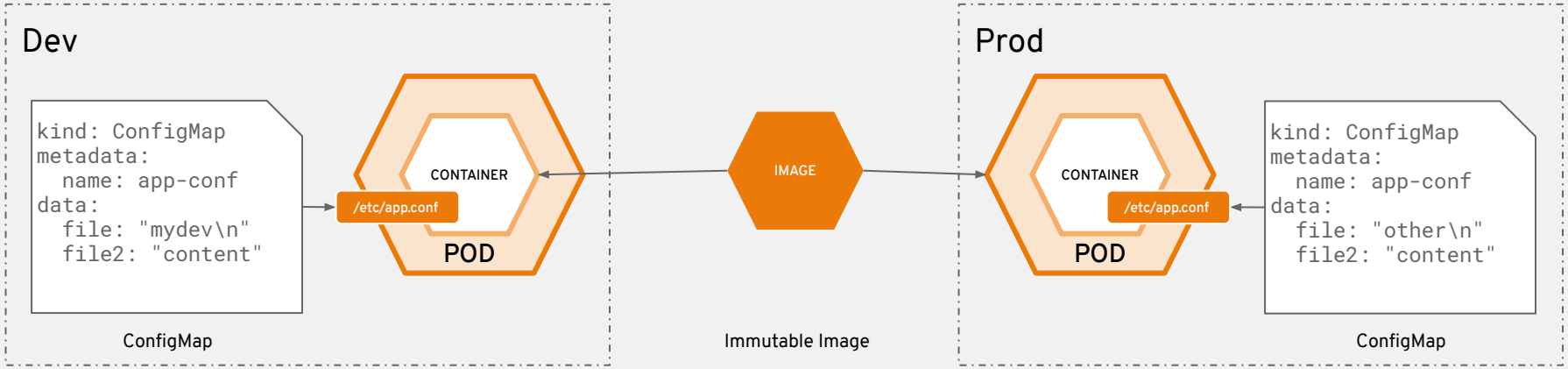



| | |
|---|---|
| BuildConfig | Specifies how to compile SourceCode into an docker-image artefact. |
| Build | Runs the build process (spec.gitref + spec.buildImage + spec.targetImage) |
| ImageStream | A pointer to an external image (spec.imageUrl) |
| Image | Application binary (content) |

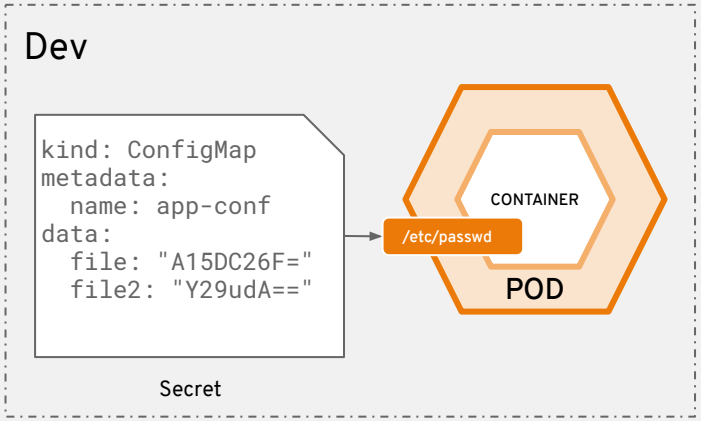
| | |
|---|---|
| PersistentVolumeClaim | A “request” for storage space (spec.size) |
| PersistentVolume | A (mount)pointer to Network Storage (spec.connectionstring) |
| ConfigMap | A mountable property file (spec.data[]) |
| Secret | A mountable base64-encoded file (spec.data[]) |

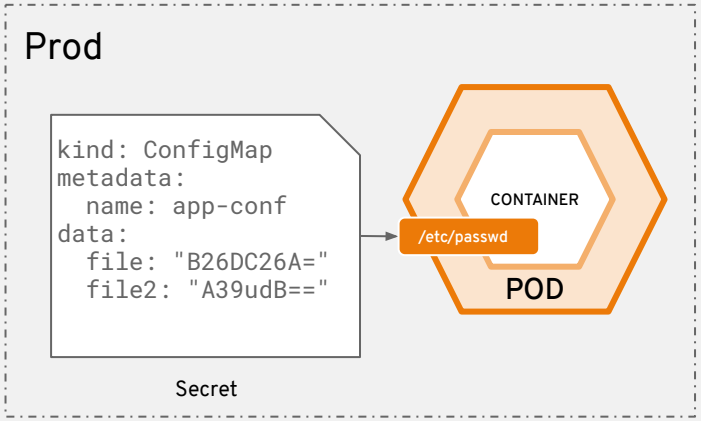
| | |
|---|---|
| DeploymentConfig | Specifies how to deploy your application (rolloutStrategy + imageChangeTriggers) |
| ReplicaSet | Monitors the Pods and restarts if needed (spec.healthcheck + spec.replicas) |
| Pod | A grouping of tightly-coupled containers (spec.containers[] + spec.node) |
| Container | 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. |

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



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



# ConfigMap

```
> oc cp httpd-persistent-0:/etc/httpd/conf/httpd.conf httpd.conf
> vi httpd.conf
DocumentRoot /var/www/html
> oc create configmap httpd-conf \
    --from-file=httpd.conf \
    --from-file=author.txt=/etc/hostname \
    --from-literal=author.sh=username=sterburg
> oc describe cm/httpd-conf |less

> oc set volumes statefulset/httpd-persistent \
    --add \
    --name=httpd-conf \
    --type=configmap  \
    --configmap-name=httpd-conf \
    --sub-path=httpd.conf
    --mount-path=/etc/httpd/conf/httpd.conf

> oc set volumes sts/httpd-persistent
> oc get sts/httpd-persistent -o yaml
> oc delete pod -l app=httpd-persistent
> oc rsh pod/httpd-persistent-0 grep DocumentRoot /etc/httpd/conf/httpd.conf
```

# Secrets

```
> oc create secret generic httpd-credentials \
    --from-literal=USERNAME=sterburg \
    --from-literal=PASSWORD=geheim
> oc describe secret/httpd-credentials

> oc set env dc/httpd \
    --from=secret/httpd-credentials

> oc get -o yaml dc/httpd

> oc rsh dc/httpd
# env |egrep "USERNAME|PASSWORD"
# echo "You can login with ${USERNAME} and ${PASSWORD}" |envsubst >index.html
# cat index.html
```

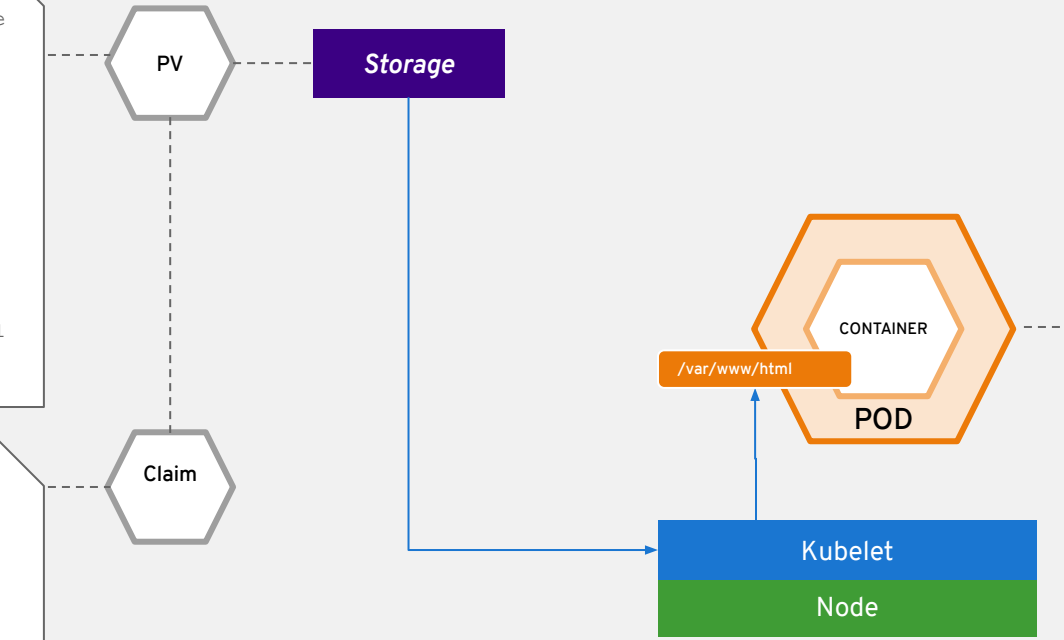
# A broad spectrum of static and dynamic storage endpoints

| | | | | | |
|---|---|---|---|---|---|
| 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)** | | |

# Claiming storage space

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv0030
spec:
  accessModes:
  - ReadWriteOnce
  - ReadWriteMany
  - ReadOnlyMany
  capacity:
    storage: 100Gi
  nfs:
    server: nfs
    path: /exports/pv1
  reclaim: Recycle
```

```
apiVersion: v1
kind:
PersistentVolumeClaim
metadata:
  name: www-data
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

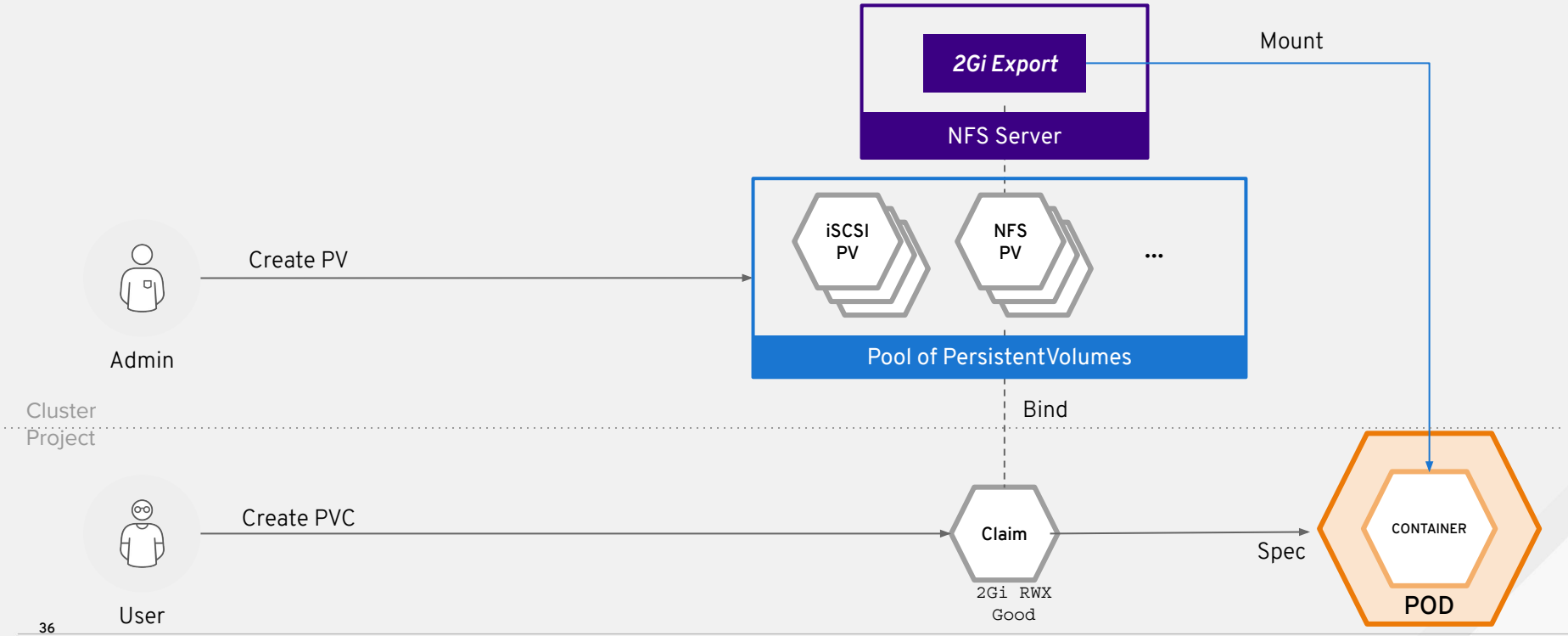



```
apiVersion: v1
kind: Pod
metadata:
  name: mypod
spec:
  containers:
    - name: myfrontend
      image: nginx
      volumeMounts:
      - mountPath: "/var/www/html"
        name: mypd
  volumes:
    - name: mypd
      persistentVolumeClaim:
        claimName: www-data
```

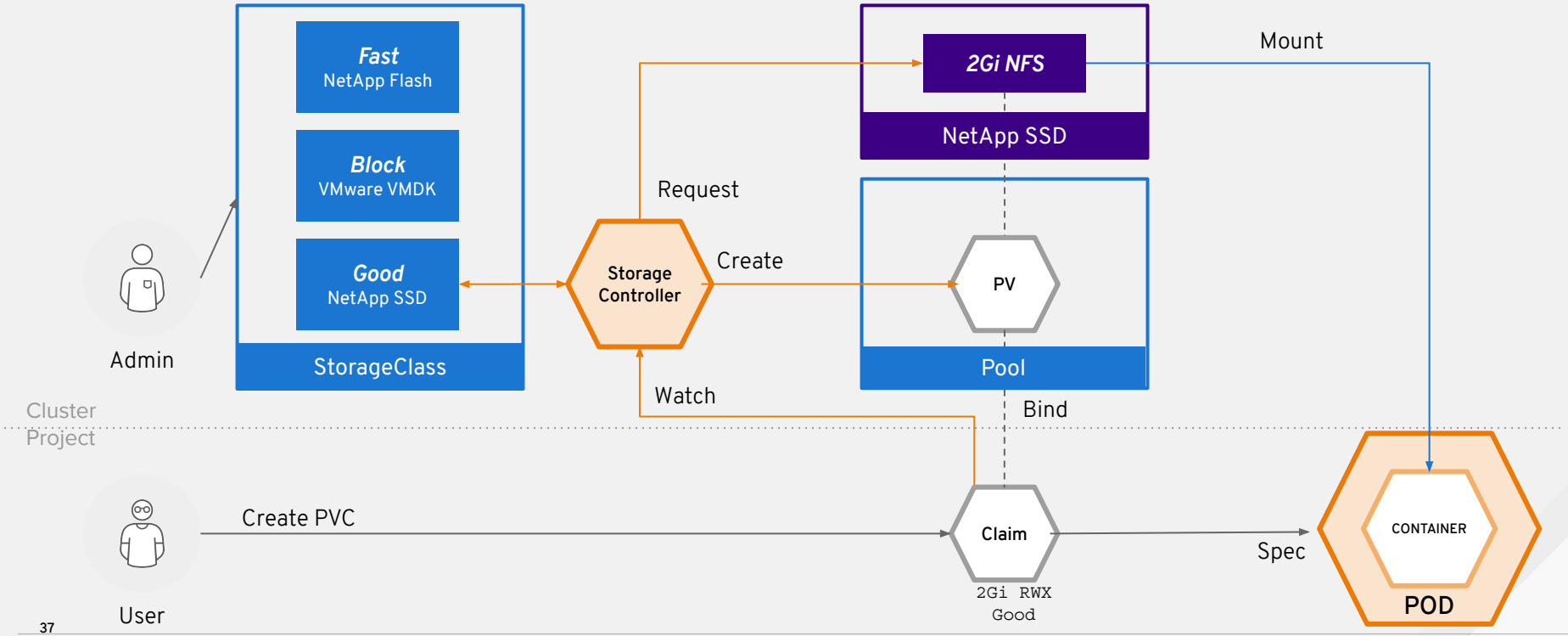


oc set volume --add dc/httpd --type=pvc --claim-name=www-data --claim-size=1Gi -m /var/www/html

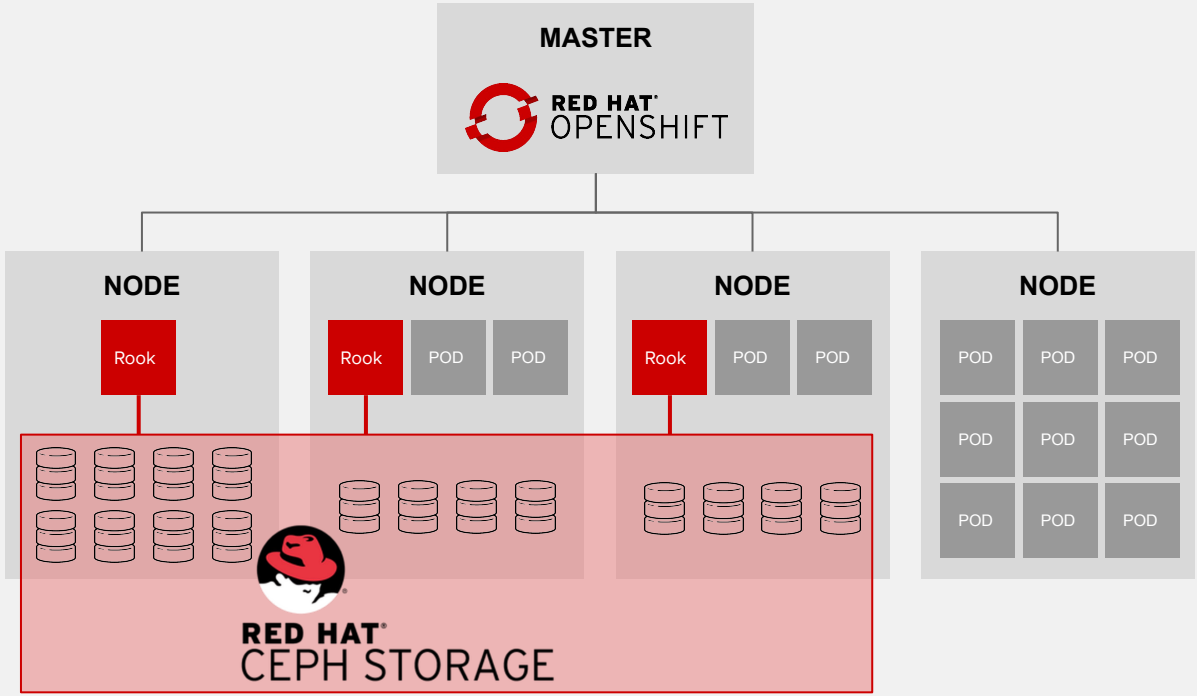
# Static Storage Provisioning



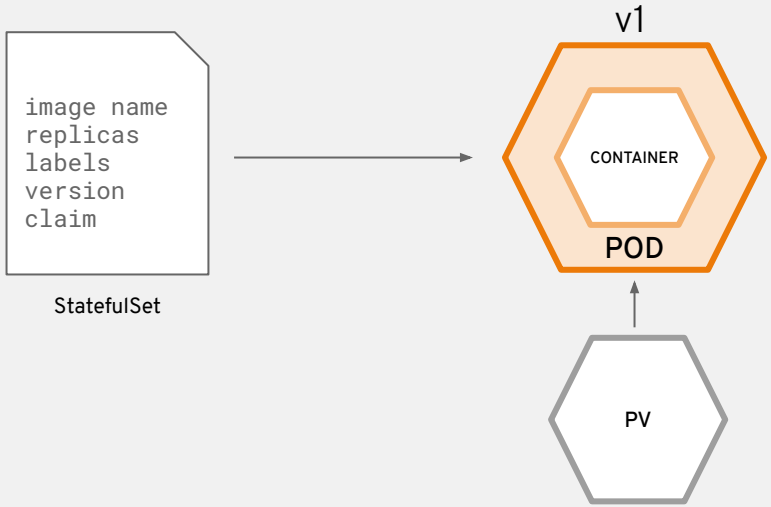
# Dynamic Storage Provisioning



# CONTAINER-NATIVE STORAGE (ocs)



# `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
```



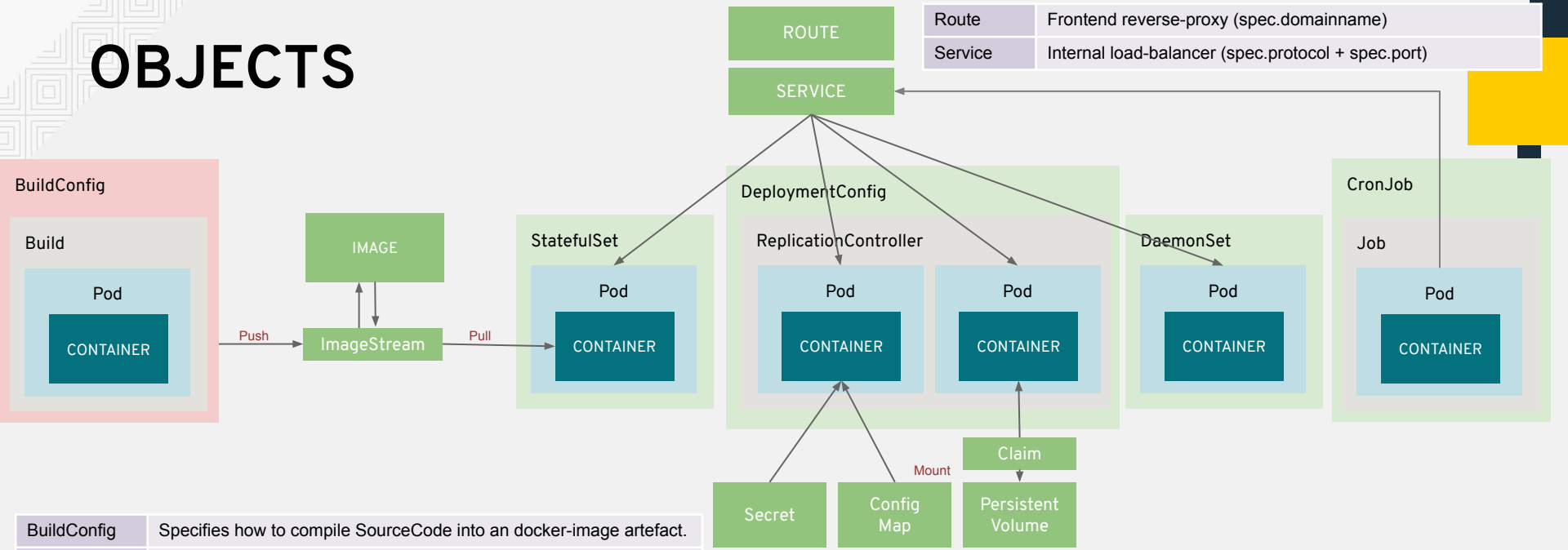
> oc create -f statefulset.yaml

# StatefulSet

```
> oc create -f statefulset.yaml
> oc scale --replicas=4 sts/httpd-persistent

> oc get pods
> oc get persistentvolumeclaims
> oc set volumes pod/httpd-persistent-0
> oc set volumes pod/httpd-persistent-1

> oc rsh httpd-persistent-0 bash -c "hostname >/var/www/html/index.html"
> oc rsh httpd-persistent-1 bash -c "hostname >/var/www/html/index.html"
> oc rsh httpd-persistent-2 bash -c "hostname >/var/www/html/index.html"
> oc rsh httpd-persistent-3 bash -c "hostname >/var/www/html/index.html"

> oc create service clusterip httpd-persistent --tcp=80:8080 -o yaml
> oc rsh httpd-persistent-0
# curl http://httpd-persistent
```

# 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[]) |

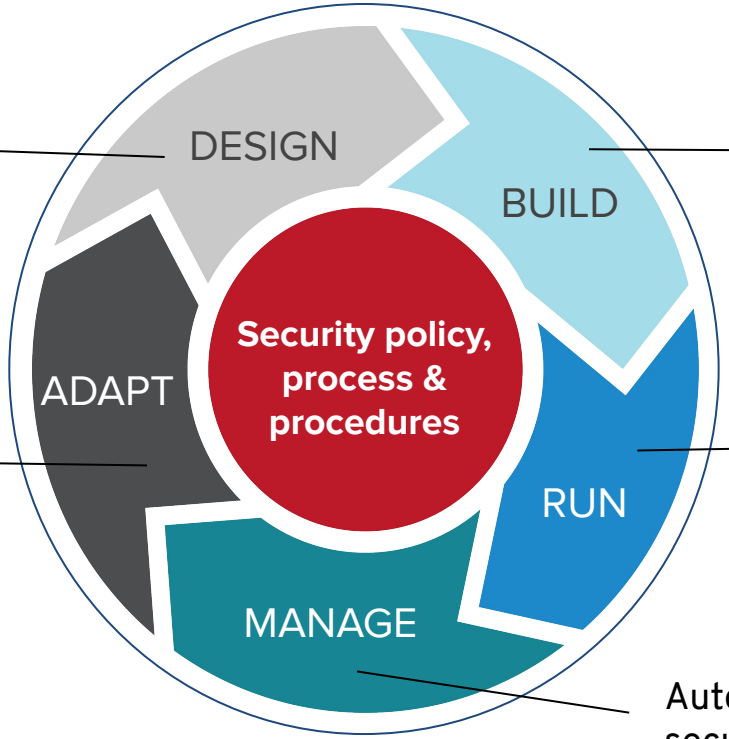
| | |
|---|---|
| DeploymentConfig | Specifies how to deploy your application (rolloutStrategy + imageChangeTriggers) |
| ReplicaSet | Monitors the Pods and restarts if needed (spec.healthcheck + spec.replicas) |
| Pod | A grouping of tightly-coupled containers (spec.containers[] + spec.node) |
| Container | 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. |

# OpenShift Security

Features, mechanisms and processes for container and platform isolation

# SECURITY MUST BE CONTINUOUS

## And integrated throughout the IT lifecycle




Identify security requirements & governance models

Built-in from the start; not bolted-on

Deploy to trusted platforms with enhanced security capabilities

Automate systems for security & compliance

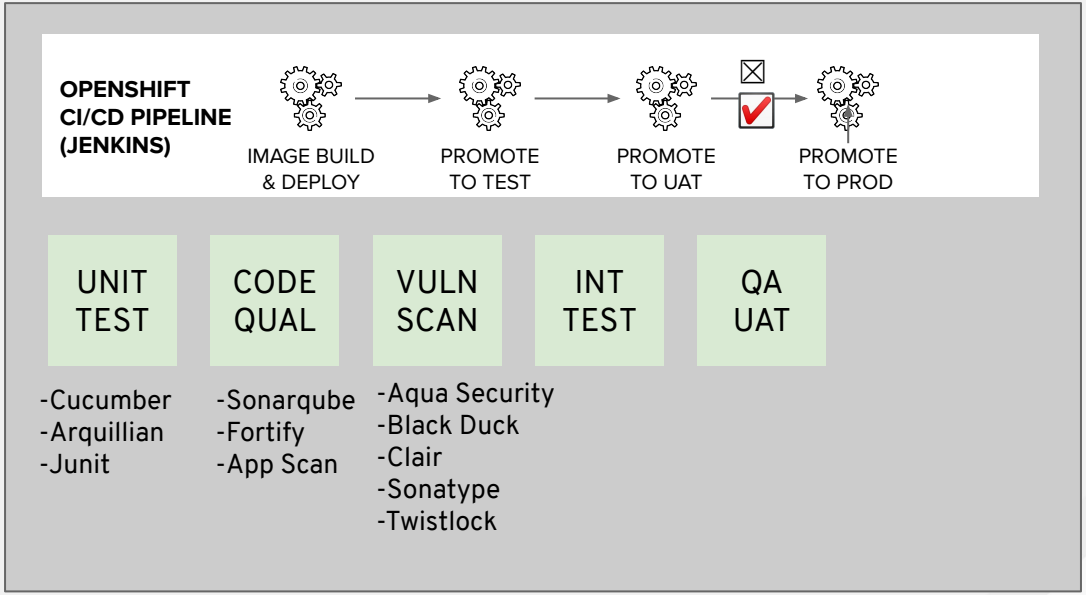
Revise, update, remediate as the landscape changes



Red Hat

# CI/CD MUST INCLUDE SECURITY GATES

- Integrate security testing into your build / CI process
- Use automated policies to flag builds with issues
- Sign your custom container images



# SECURITY FEATURES

## DEFENSE IN DEPTH - Control, Defend, Extend

**Linux Host Security**
- RHCOS minimal, immutable OS
- RHCOS updates managed and delivered as integrated part of the OpenShift platform

**Authentication & Authorization**
- Integration with external Keycloak
- Use group membership from external IPs

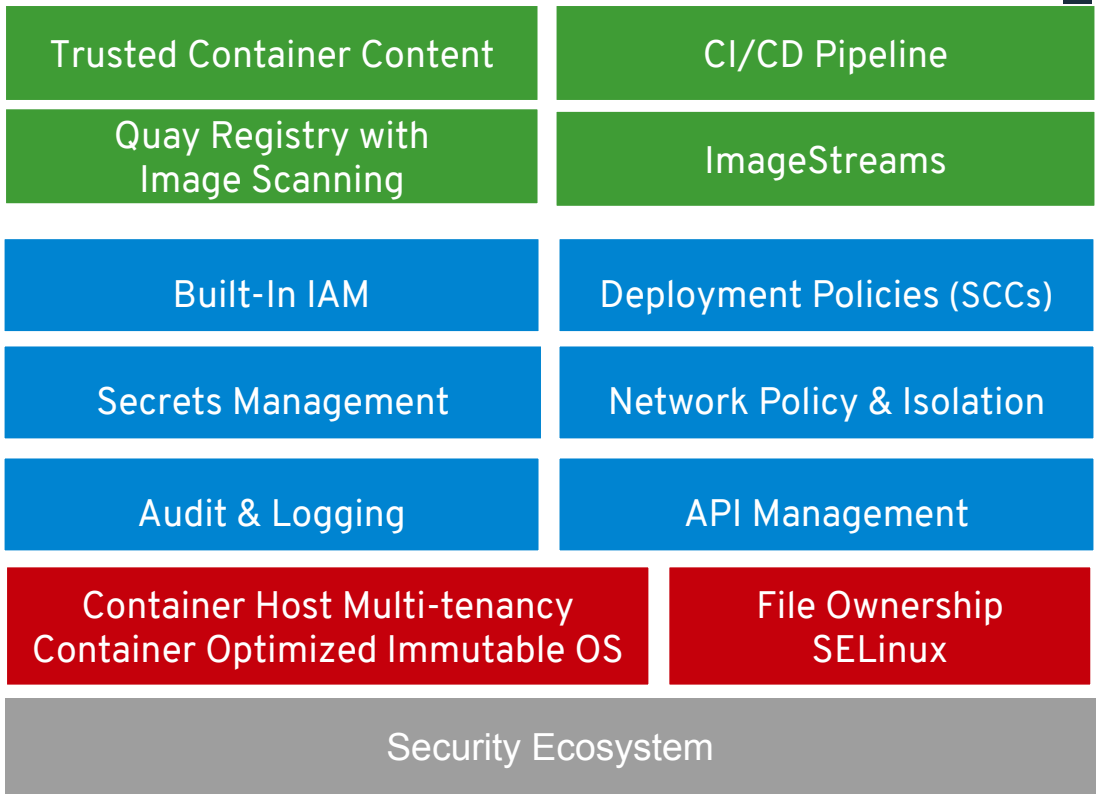
**Secrets & Certificates**
- Encrypted certs stored in etcd (4.0)
- Improved cert management and Integration with external CAs via ACME
- Integration with external Key Management Systems

**Integrated Audit & Logging**
- East / West traffic tracing with OpenShift Service Mesh

**Network Policies**
- Control service access flow with OpenShift Service Mesh

**Networking Isolation**
- East / West mutual TLS authentication with OpenShift Service Mesh
- Multus to isolate control plane / data plane (4.0)

**Image Security**
- Clair v3 covers more content



# MULTI-TENANCY

Container Security starts with Linux Security

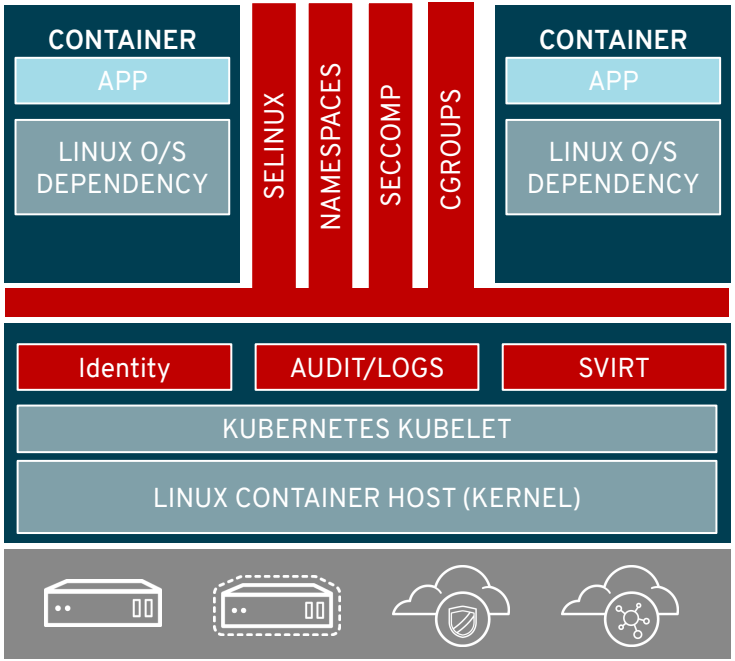
Security in the RHEL host applies to the container

SELINUX and Kernel Namespaces are the one-two punch no one can beat

Protects not only the host, but containers from each other

RHEL CoreOS provides minimized attack surface

Common Criteria certification






Red Hat

# HARDENING TOOLS & APPLICABILITY GUIDES

Product Applicability Guides

- FISMA Moderate & FISMA (NIST)
- ISO 27001
- PCI-DSS
- PCI-DSS Reference Architecture
- AVG

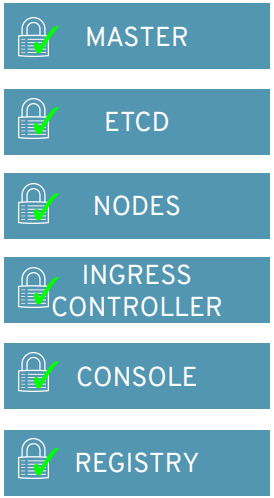
OpenShift Hardening Guide for OpenShift 4.

Upstream, 3rd party tools

- docker-bench
- kube-bench

# Certificates and Certificate Management

- OpenShift provides its own internal CA
- Certificates are used to provide secure connections to
  - master (APIs) and nodes
  - Ingress controller and registry
  - etcd
  - optional: services/pods
- Certificate rotation is automated
- Optionally configure external endpoints to use custom certificates
- Let's Encrypt also supported as add-on to provide secure connections to:
  - Routes
- Istio ServiceMesh provides equal functionality



# RUNTIME SECURITY POLICIES

## SCC (Security Context Constraints)

Allow administrators to control permissions for pods

Restricted SCC is granted to all users

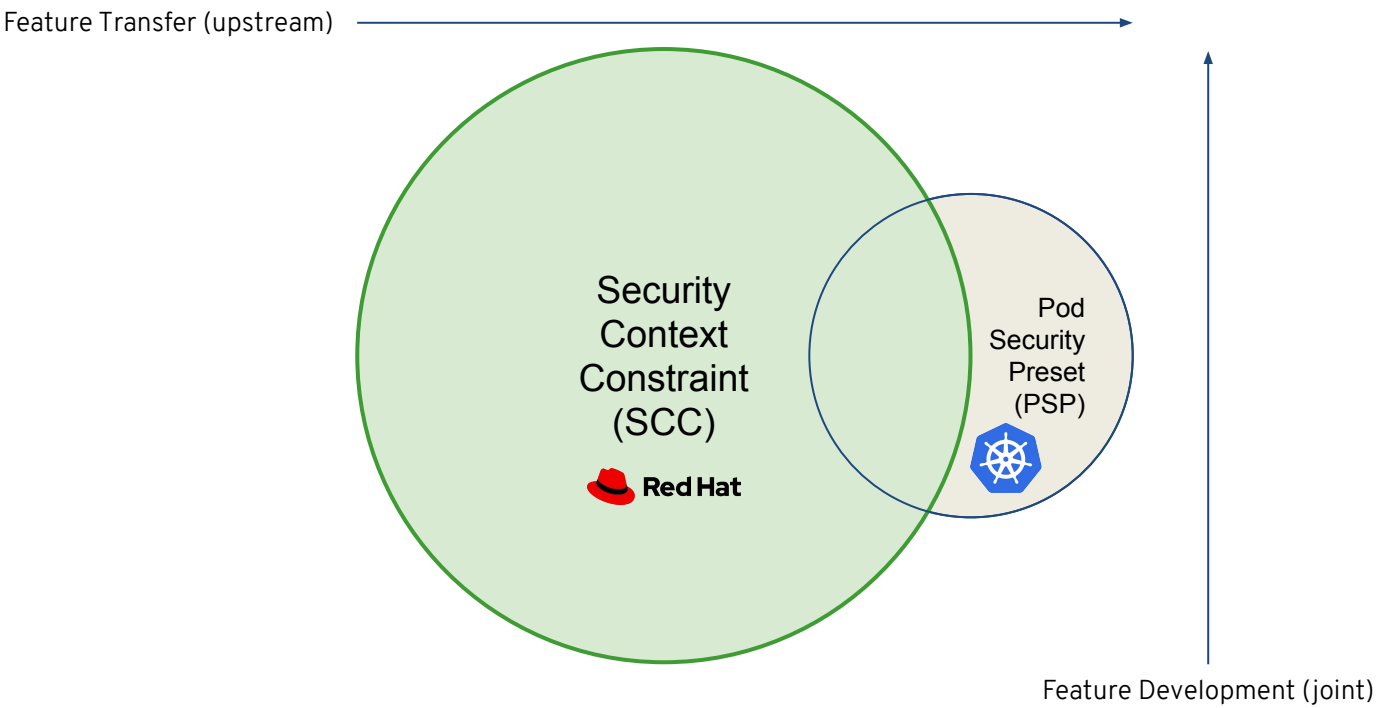
By default, no containers can run as root

Admin can grant access to privileged SCC

Custom SCCs can be created

```
$ oc describe scc restricted
Name:                                   restricted
Priority:                               <none>
Access:
  Users:                                <none>
  Groups:                               system:authenticated
Settings:
  Allow Privileged:                     false
  Default Add Capabilities:             <none>
  Required Drop Capabilities:           KILL,MKNOD,SYS_CHROOT,SETUID,SETGID
  Allowed Capabilities:                 <none>
  Allowed Seccomp Profiles:             <none>
  Allowed Volume Types:                 configMap,downwardAPI,emptyDir,persistentVolumeClaim,projected,
  Allow Host Network:                   false
  Allow Host Ports:                     false
  Allow Host PID:                       false
  Allow Host IPC:                       false
  Read Only Root Filesystem:            false
  Run As User Strategy: MustRunAsRange
```

# Extended Depth of Protection






> oc adm policy add-scc-to-user privileged -z system:serviceaccount:workshop-sterburg:default

# Privilege Escalation

```
> oc create serviceaccount httpd-root
> oc get serviceaccounts #sa
> oc get securitycontextconstraints #scc
> oc adm policy add-scc-to-user privileged -z httpd-root
> oc get -o yaml scc/privileged
> oc set serviceaccount httpd-root dc/httpd
> oc set volume --add dc/httpd --type=hostPath --path=/ -m /host
> oc edit dc/httpd
> oc rsh dc/httpd
# id
# ps -ef
# ls /host
```

```
kind: DeploymentConfig
spec:
  template:
    spec:
      hostPID: true
      containers:
        securityContext:
          privileged: true
          runAsUser: 0 #root
```

# IDENTITY AND ACCESS MANAGEMENT

OpenShift includes an OAuth server, which does three things:

- Identifies the person requesting a token, using a configured identity provider
- Determines a mapping from that identity to an OpenShift user
- Issues an OAuth access token which authenticates that user to the API

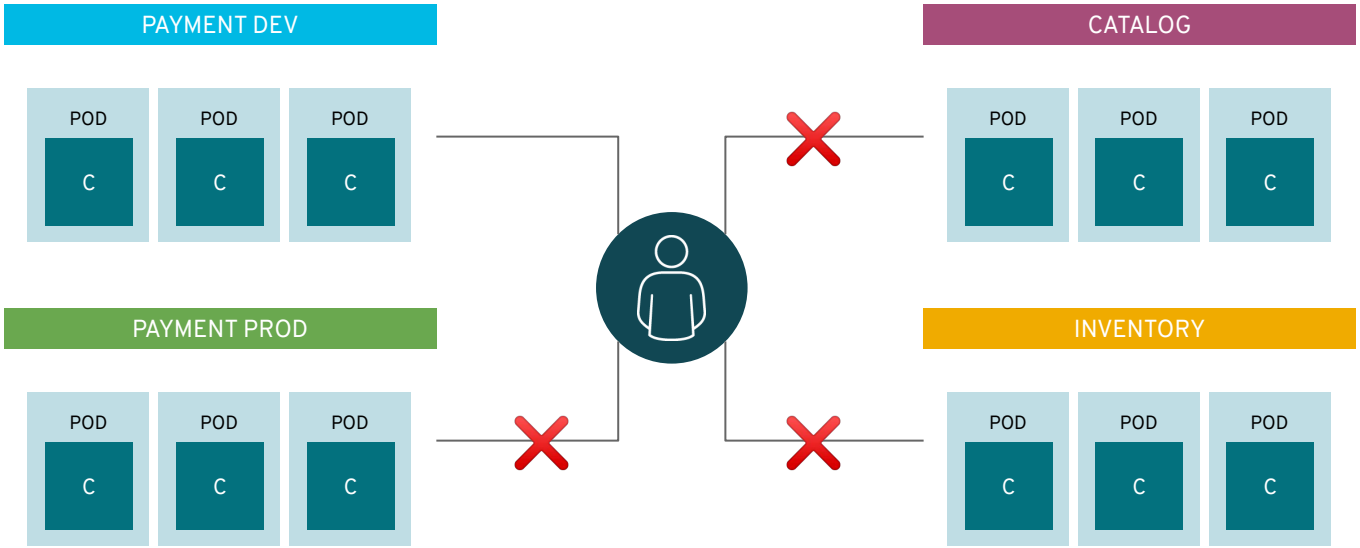
Managing Users and Groups in OpenShift
Configuring Identity Providers

Supported Identity Providers include

Keystone
LDAP
GitHub
GitLab
GitHub Enterprise
Google
OpenID Connect
Security Support Provider Interface (SSPI) to support SSO flows on Windows (Kerberos)

# PROJECTS ISOLATE APPLICATIONS

## “Global” vs “Project” scoped



# ROLE BASED ACCESS CONTROL

- Project scope & cluster scope available
- Matches request attributes (verb,object,etc)
- If no roles match, request is denied ( deny by default )
- Operator- and user-level roles are defined by default
- Custom roles are supported

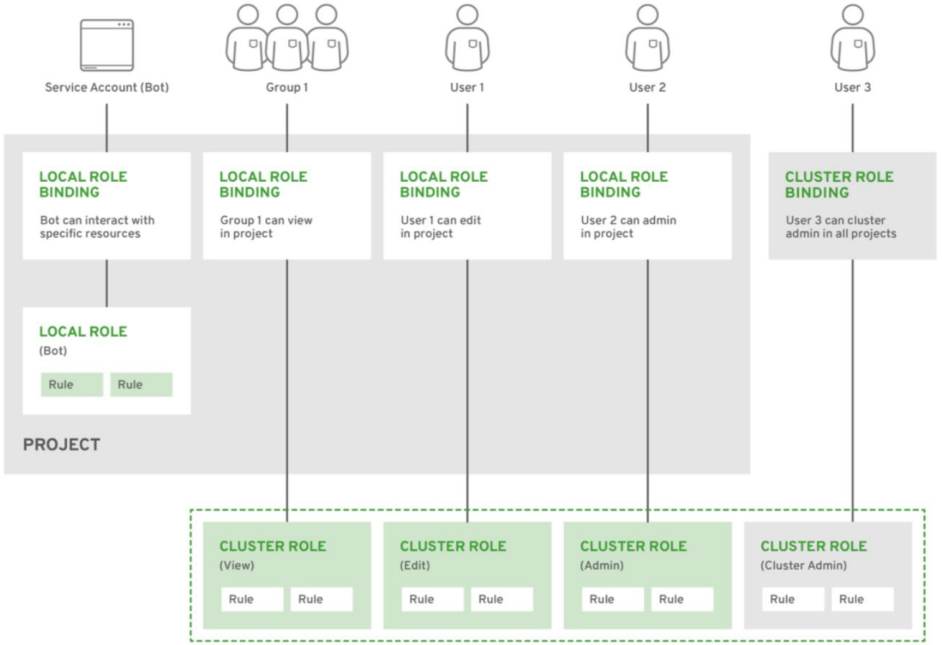


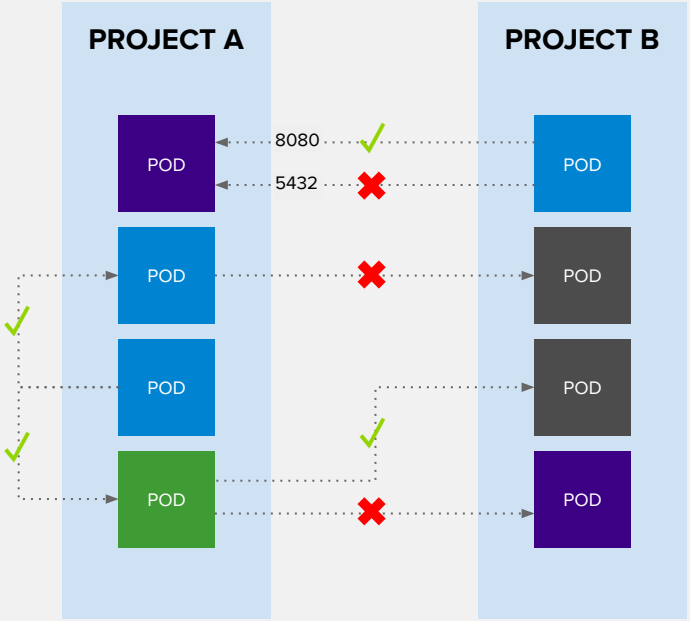

*Figure 12 - Authorization Relationships*

# RoleBinding

```
oc get clusterroles |egrep "^(view|edit|admin)"
oc -n workshop-sterburg get roles

oc describe clusterrole/view
oc get -o yaml clusterrole/view >view.yml
vi view.yml
oc policy add-role-to-user --role-namespace=default view-secrets <username>
oc get roles,rolebindings
oc policy who-can list secrets

oc get serviceaccounts
oc describe sa default
oc describe secret <default-token-34c0o>
oc rsh dc/httpd
# cd /run/secrets/kubernetes.io/serviceaccount/
# ls
# cat token
oc login --token=<token>
oc whoami
oc get all
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: view-secrets
rules:
- apiGroups:
  - ""
resources:
- secrets
verbs:
  - list
  - get
```

# Network Policies




Example Policies

- Allow all traffic inside the project
- Allow traffic from green to gray
- Allow traffic to purple on 8080

```
apiVersion: extensions/v1beta1
kind: NetworkPolicy
metadata:
  name: allow-to-purple-on-8080
spec:
  podSelector:
    matchLabels:
      color: purple
  ingress:
  - ports:
    - protocol: tcp
      port: 8080
```

# NetworkPolicy

```
oc rsh dc/httpd
# curl http://httpd.workshop-<otherUser>.svc.cluster.local:8080/

oc create -f -

oc rsh dc/httpd
# curl http://httpd.workshop-<otherUser>.svc.cluster.local:8080/

oc explain NetworkPolicy.spec.ingress \
  --api-version=networking.k8s.io/v1

oc edit networkpolicy/allow-friends
```

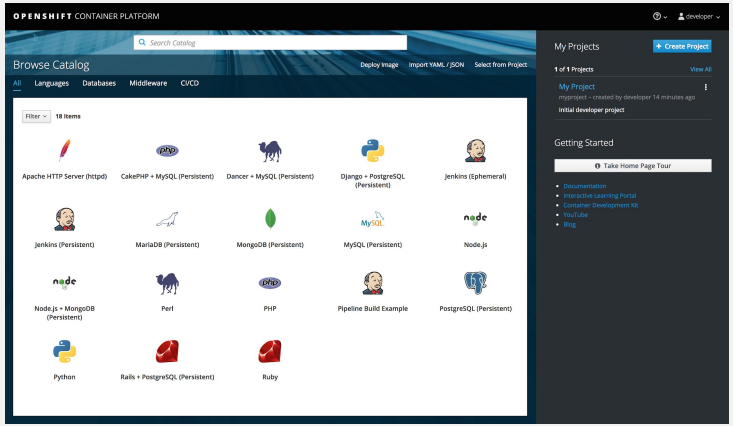
```
spec:
  podSelector:
  ingress:
  - from:
    - podSelector: {}
  - from:
    - namespaceSelector:
        matchLabels:
          name: default
          network.openshift.io/policy-group: monitoring
          openshift.io/requester: <otherUser>
```

```
kind: List
items:
- kind: NetworkPolicy
  apiVersion: networking.k8s.io/v1
  metadata:
    name: default-deny
  spec:
    podSelector: []
- apiVersion: networking.k8s.io/v1
  kind: NetworkPolicy
  metadata:
    name: allow-friends
  spec:
    podSelector:
    ingress:
    - from:
      - podSelector: {}
    - from:
      - namespaceSelector:
          matchLabels:
            name: default
- apiVersion: network.openshift.io/v1
  kind: EgressNetworkPolicy
  metadata:
    name: default-deny-egress
  spec:
    egress:
    - type: Deny
      to:
        cidrSelector: 0.0.0.0/0
```

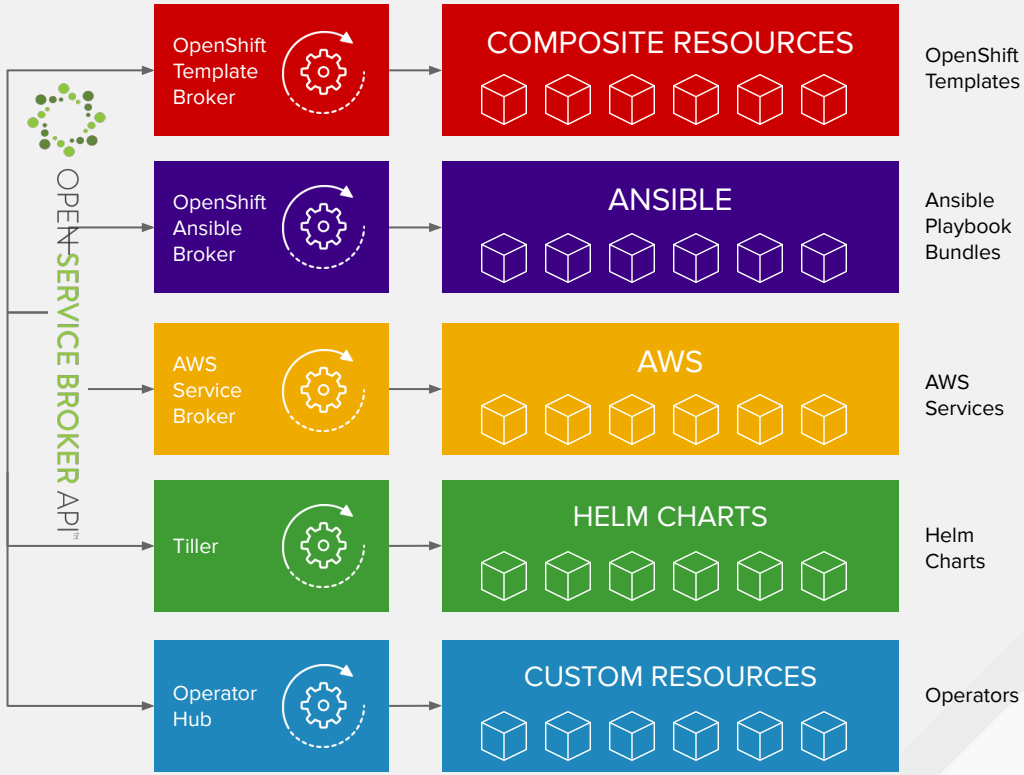
# Templates

Easily deploy a complete application stack comprised of multiple services.

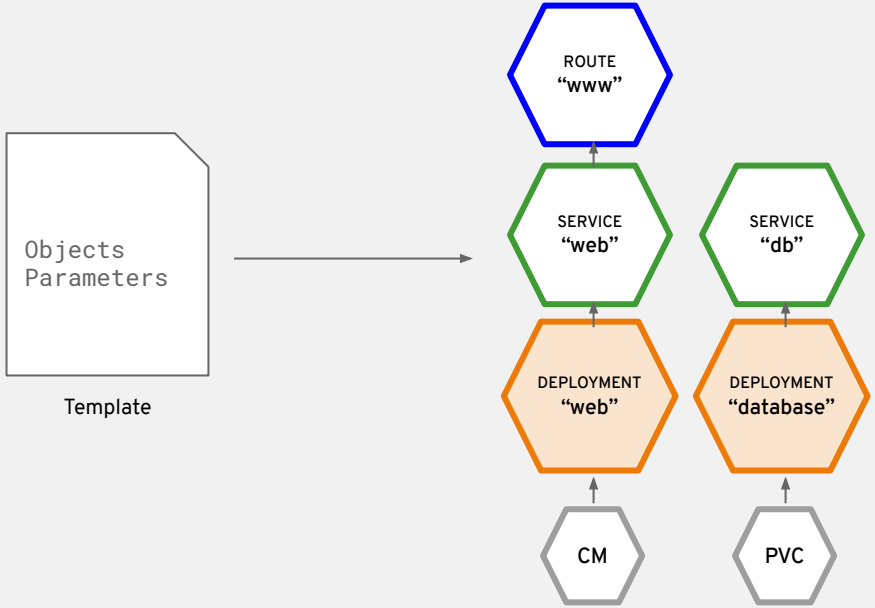
# SERVICE CATALOG




**OPENSHIFT DEVELOPER CATALOG**



# a `Template` is a list of composite resources




```
apiVersion: template.openshift.io/v1
kind: Template
metadata:
  name: tomcat9-mysql-persistent-s2i
  namespace: openshift
objects:
- apiVersion: v1
  kind: Service
  metadata:
    name: ${APPLICATION_NAME}
  spec:
    ports:
    - port: ${LISTEN_PORT}
      targetPort: 8080
- apiVersion: v1
  kind: Service
  metadata:
    name: ${APPLICATION_NAME}-mysql
  spec:
    ports:
    - port: 3306
parameters:
- displayName: Application Name
  name: APPLICATION_NAME
  required: true
  value: jws-app
  from: '[A-Z0-9]{8}'
  generate: expression
```



> oc create -f template.yaml

# Templates

```
oc get templates -n openshift
oc new-app mysql-persistent

oc get -o yaml --export \
  is/httpd dc/httpd svc/httpd \
  secret/mysql svc/mysql pvc/mysql dc/mysql \
  >template.yaml
vi template.yaml  #cleanup, parameterize
vi prod.properties

oc new-project template-<username>
oc new-app \
   -f template.yaml \
   -p APPLICATION_NAME=myhttpd \
   --param-file=prod.properties
oc process \
   -f template.yaml \
   -p APPLICATION_NAME=otherhttpd \
   --param-file=prod.properties \
|oc apply -f -
oc -n openshift create -f template.yaml

oc delete project template-<username>
```

prod.properties:

```
MEMORY_LIMIT=2Gi
VOLUME_CAPACITY=1Gi
MYSQL_ROOT_PASSWORD=production
```

# a `Template` is useful for:

- Infrastructure-as-Code
  - Stored in Git
    - Alongside your app code as deployment instructions
    - Peer Review
- Sharing your app-stack with others (Service Catalog)
- CI/CD
- Release Management



> oc create -f http://raw.github.com/openshift/examles.git/template.yaml