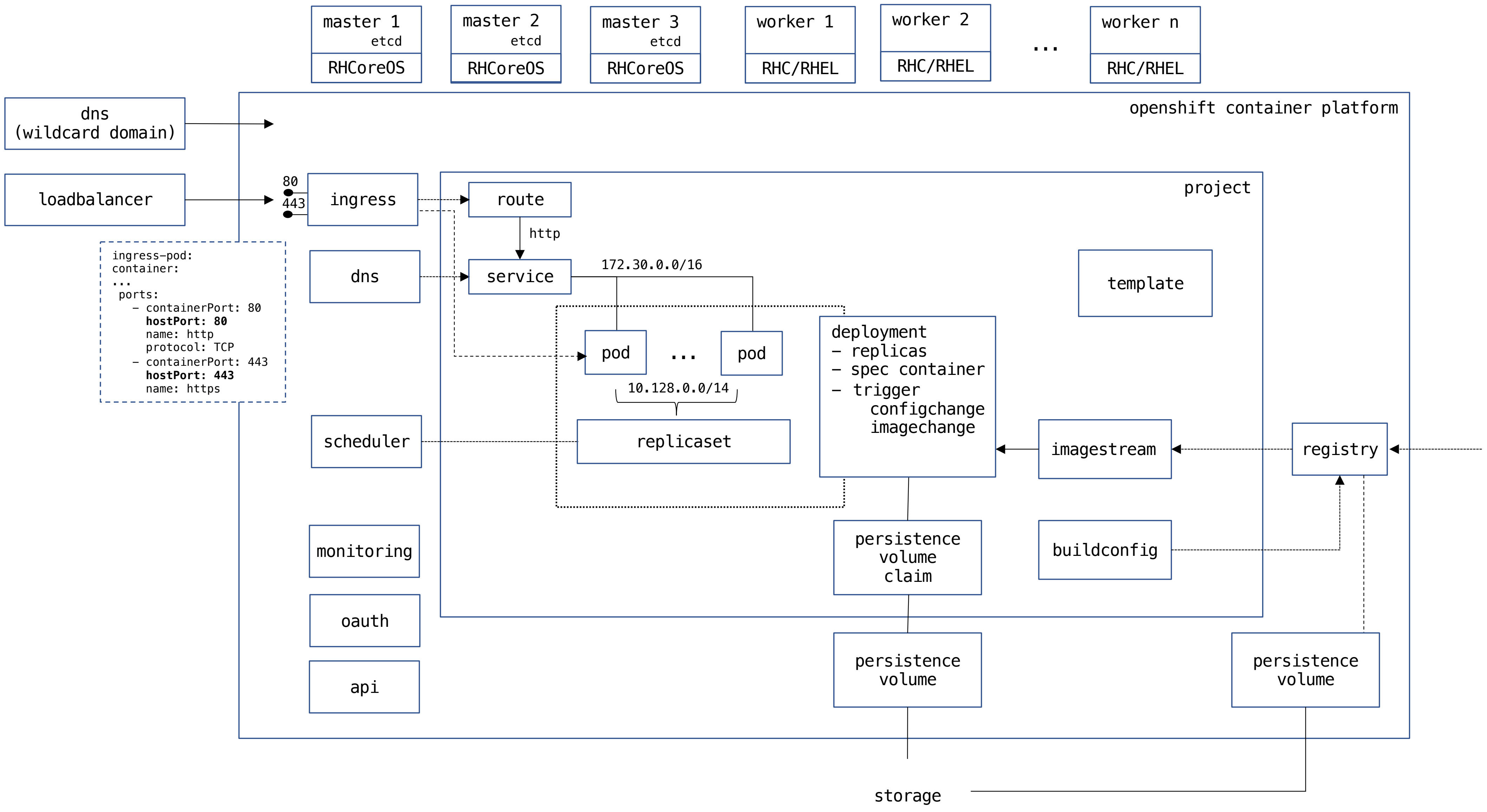
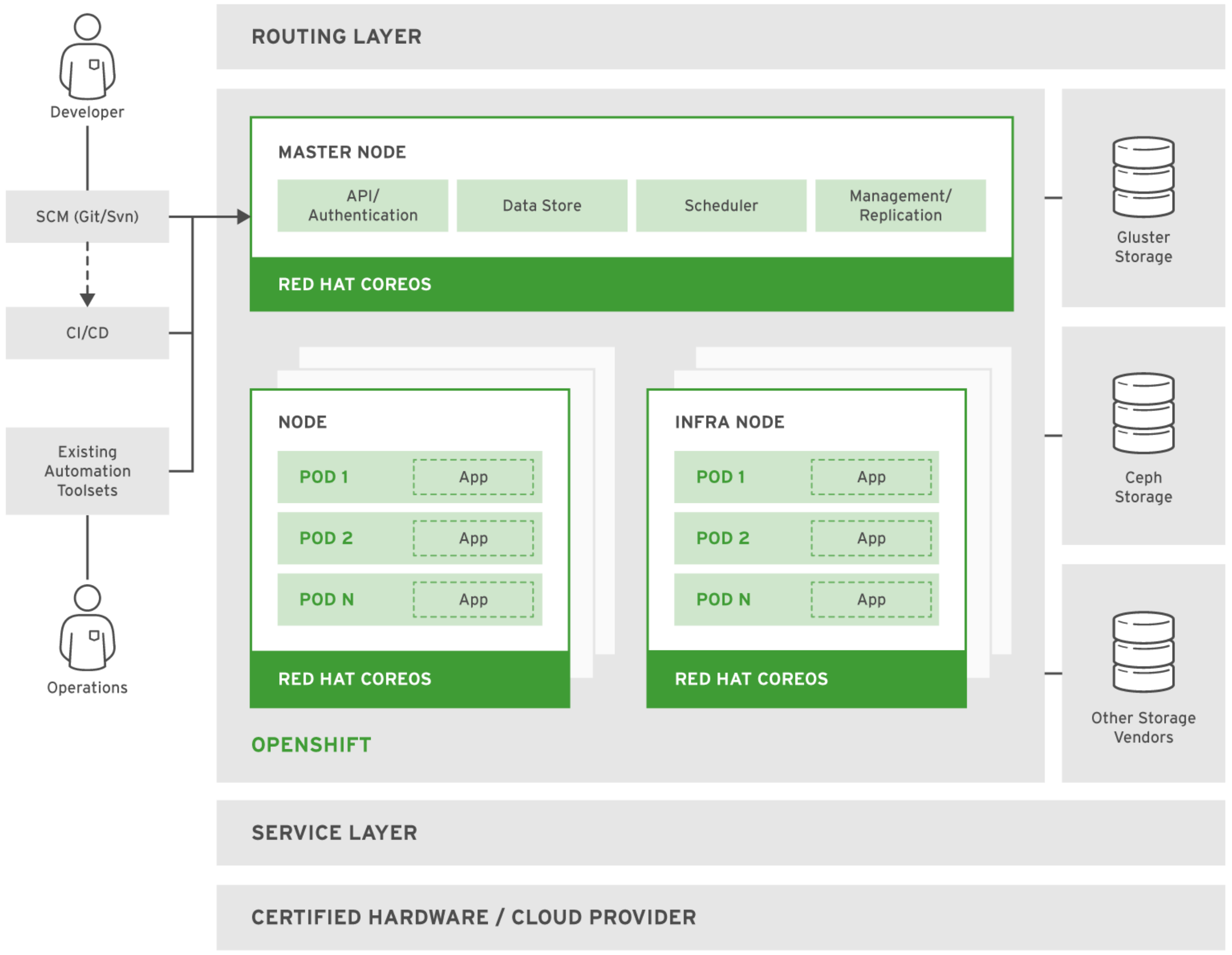
Day 1	Day 2	Day 3	Day 4	Day 5
Deploying and Managing Applications	Publishing Enterprise Container Images	Customizing Source-to-Image Builds	Managing Application Deployments	Comprehensive Review
Designing Containerized Applications	Managing Builds on OpenShift	Deploying Multi-container Applications	Building Applications for OpenShift	

schedule

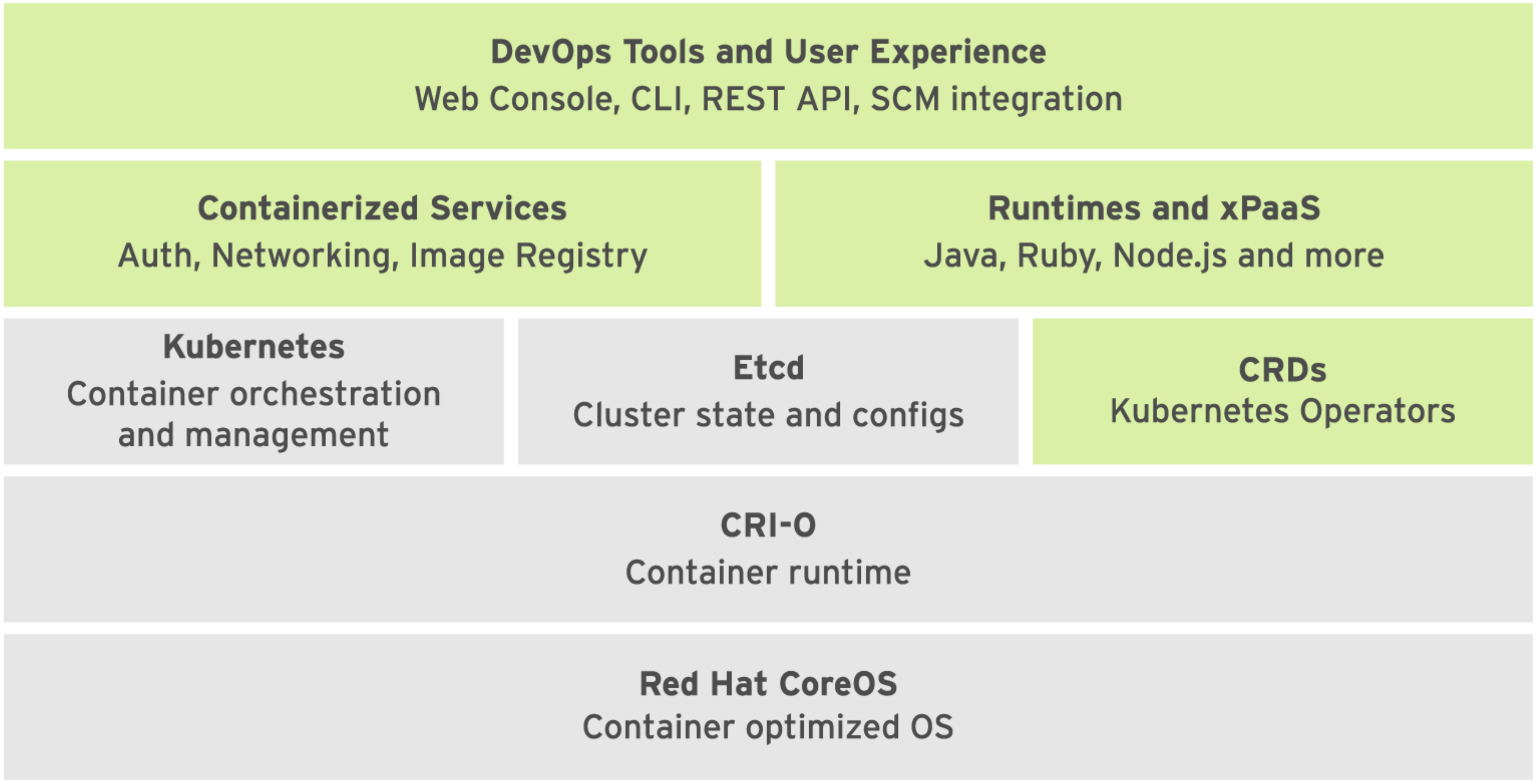


```
ingress-pod:
container:
...
 ports:
   - containerPort: 80
     hostPort: 80
     name: http
     protocol: TCP
   - containerPort: 443
     hostPort: 443
     name: https
```

openshift



openshift architecture


DevOps Tools and User Experience
Web Console, CLI, REST API, SCM integration
Containerized Services
Auth, Networking, Image Registry
Runtimes and xPaaS
Java, Ruby, Node.js and more
Kubernetes
Container orchestration and management
Etcd
Cluster state and configs
CRDs
Kubernetes Operators
CRI-O
Container runtime
Red Hat CoreOS
Container optimized OS

openshift component stack

```
$ oc api-resources -o name --sort-by=name
alertmanagers.monitoring.coreos.com
apiservers.config.openshift.io
apiservices.apiregistration.k8s.io
appliedclusterresourcequotas.quota.openshift.io
authentications.config.openshift.io
authentications.operator.openshift.io
baremetalhosts.metal3.io
bindings
brokertemplateinstances.template.openshift.io
buildconfigs.build.openshift.io
builds.build.openshift.io
builds.config.openshift.io
catalogsources.operators.coreos.com
certificatesigningrequests.certificates.k8s.io
cloudcredentials.operator.openshift.io
clusterautoscalers.autoscaling.openshift.io
clusternetworks.network.openshift.io
clusteroperators.config.openshift.io
...
```

Pod
Replicationcontroller (rc)
Deploymentconfig (dc)
Service (svc)
Route
PersistenceVolumeClaim (pvc)
Secrets
Configmaps (cm)

Imagestream (is)
BuildConfig (bc)

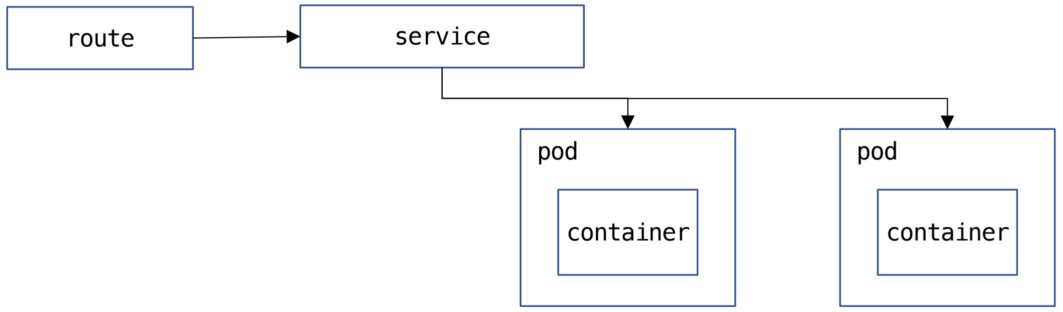
Node
PersistenceVolume (pv)

Operator
CustomResourceDefinition (crd)

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wildfly
  labels:
    app: wildfly
spec:
  replicas: 2
  selector:
    matchLabels:
      app: wildfly
  template:
    metadata:
      labels:
        app: wildfly
    spec:
      containers:
      - name: wildfly
        image: quay.io/do288/wildfly:latest
        ports:
        - containerPort: 8080
          protocol: TCP
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: wildfly
  labels:
    app: wildfly
spec:
  type: ClusterIP
  selector:
    app: wildfly
  ports:
  - name: http
    protocol: TCP
    port: 8080
    targetPort: 8080
```

```yaml
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  name: wildfly
  labels:
    app: wildfly
spec:
  host: sample.apps.eu46.prod.nextcle.com
  to:
    kind: Service
    name: wildfly
  tls:
    termination: edge
```



application

Declarative :

```
$ ls
deployment.yml route.yml service.yml

$ oc apply -f .
deployment.apps/wildfly created
route.route.openshift.io/wildfly created
service/wildfly created
```

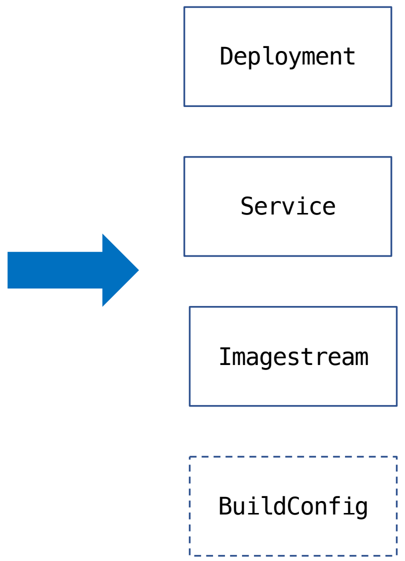
Imperative :

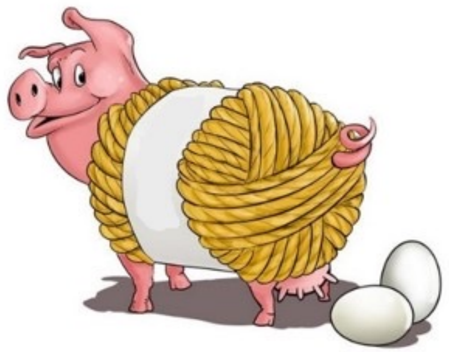
```
$ oc new-app <container-image | git-repository>
--> Found container image 9a9e908 (9 days old) from quay.io for "quay.io/do288/wildfly"

    * An image stream tag will be created as "wildfly:latest" that will track this image

--> Creating resources ...
    imagestream.image.openshift.io "wildfly" created
    deployment.apps "wildfly" created
    service "wildfly" created
--> Success
```

```
$ oc new-app --help
Create a new application by specifying source code, templates, and/or images

...

Usage:
  oc new-app (IMAGE | CONTAINTERFILE | SOURCE | TEMPLATE | ...) [flags]


Beispiele:
```

$\texttt{\$ oc new-app } \underbrace{\texttt{quay.io/do288/nginx}}_{\text{Container-Image}} \texttt{ --name ngnix}$

$\texttt{\$ oc new-app } \underbrace{\texttt{php:7.3}}_{\text{Builder-Image (s2i)}}\texttt{\textasciitilde}\underbrace{\texttt{https://github.com/.../php-hello}}_{\text{Git-Projekt (Source)}}$

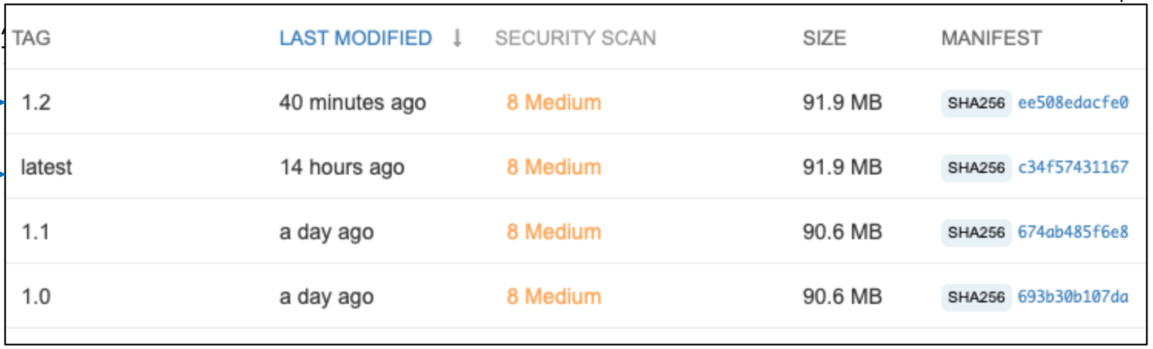




Imagestream:

- enthält Verweise (Zeiger) auf Images und deren Tags (keine Images)
- Verwendung in Deployment als Image und Trigger oder in BuildConfig als (S2I) BuilderImage
- Import aus externer Registry oder Ergebnis eines Build

```
$ oc import-image nginx --from=quay.io/do288/nginx --confirm  ( --scheduled)
$ oc describe is nginx
Name:                            nginx
Unique Images:                   4
Tags:                            4

latest
  updates automatically from registry quay.io/do288/nginx:latest
  * quay.io/do288/nginx@sha256:c34f57431167fca470730b67a1a8636126d2464eee619ec8d0b577c8e63bffef

1.2
  updates automatically from registry quay.io/do288/nginx:1.2
  * quay.io/do288/nginx@sha256:ee508edacfe0bc1e6af43a15348b400a7d97121507348bd5fb5effb6b9f8d84e

1.1
  updates automatically from registry quay.io/do288/nginx:1.1
  * quay.io/do288/nginx@sha256:674ab485f6e83f162eb4bdaf12986469c7b4f484f65fbb18f3b03218fd5f36e4

1.0
  updates automatically from registry quay.io/
  * quay.io/do288/nginx@sha256:693b30b107da26
```



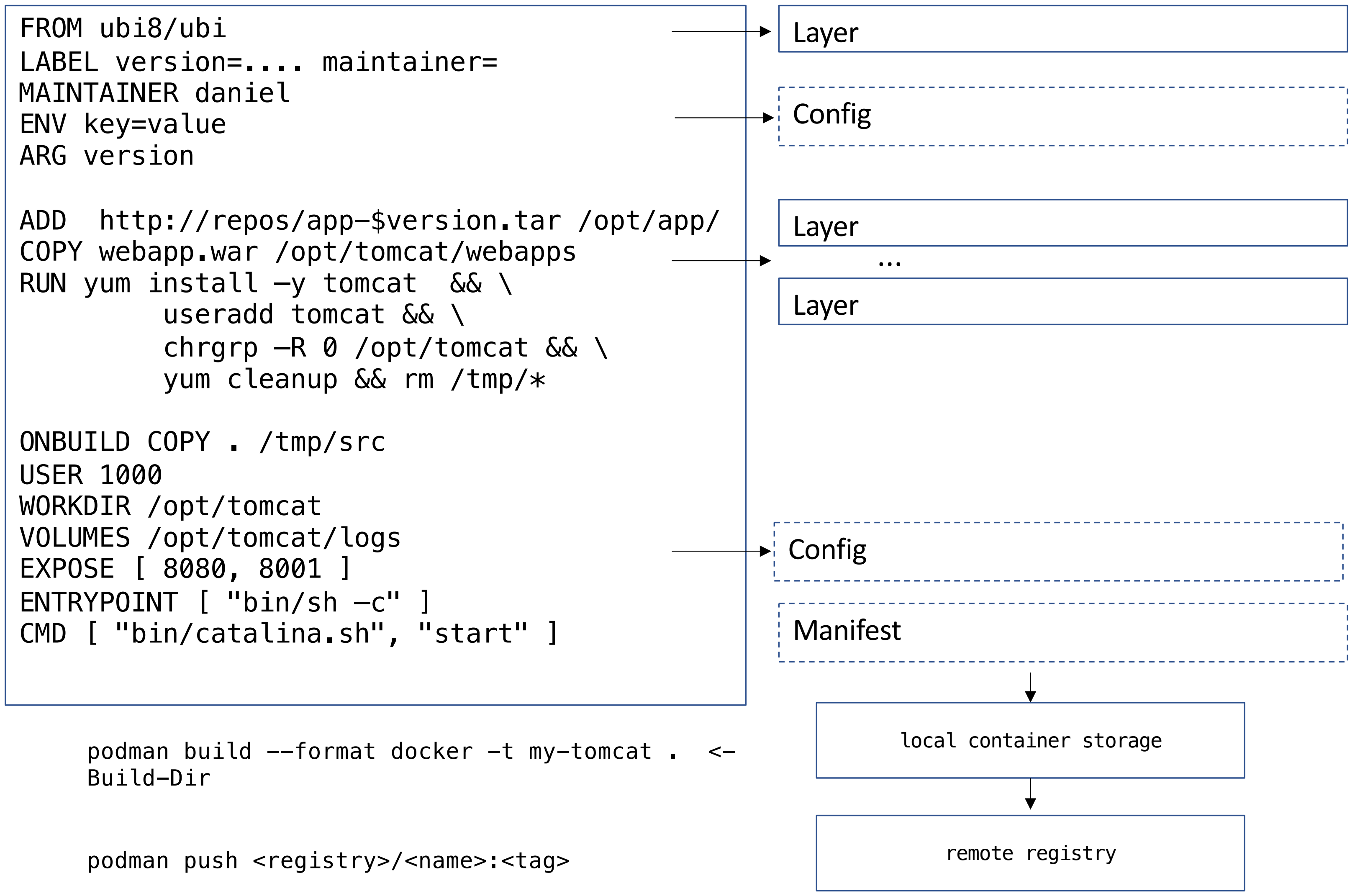
TAG	LAST MODIFIED ↓	SECURITY SCAN	SIZE	MANIFEST
1.2	40 minutes ago	8 Medium	91.9 MB	SHA256 ee508edacfe0
latest	14 hours ago	8 Medium	91.9 MB	SHA256 c34f57431167
1.1	a day ago	8 Medium	90.6 MB	SHA256 674ab485f6e8
1.0	a day ago	8 Medium	90.6 MB	SHA256 693b30b107da

imagestream

- `oc login –u <user> -p <password> <api-server-url>`
- `oc new-project <name>`
- `oc create –f <resource-yml>`
  `oc apply –f <resource-yml>`
- `oc status`
- `oc get <resource-type> [ <resource-name> ]`
  - `oc get pods`
  - `oc get deployment`
  - `oc get svc <service>`
  - `oc get events`
- `oc describe <resource-type> <resource-name>`
- `oc expose svc <service-name>`
- `oc logs <podname>`
- `oc exec –it <podname> -- <program>`
- `oc rsh <podname>`
- `oc cp <pod>:<locatio> <location>`
- `oc port-forward <podname> <local-port>:<remote-port>`
- `oc new-app <☺anything☺>`
- `oc delete <resource-type> <resource-name>`
- `oc rollout latest deployment <deployment-name>`

https://docs.openshift.com/container-platform/4.12/cli_reference/openshift_cli/developer-cli-commands.html

podman build - Containerfile



```
FROM ubi8/ubi
LABEL version=.... maintainer=
MAINTAINER daniel
ENV key=value
ARG version

ADD  http://repos/app-$version.tar /opt/app/
COPY webapp.war /opt/tomcat/webapps
RUN yum install -y tomcat  && \
         useradd tomcat && \
         chrgrp -R 0 /opt/tomcat && \
         yum cleanup && rm /tmp/*

ONBUILD COPY . /tmp/src
USER 1000
WORKDIR /opt/tomcat
VOLUMES /opt/tomcat/logs
EXPOSE [ 8080, 8001 ]
ENTRYPOINT [ "bin/sh -c" ]
CMD [ "bin/catalina.sh", "start" ]
```

```
podman build --format docker -t my-tomcat .  <-
Build-Dir
```

```
podman push <registry>/<name>:<tag>
```

build image

Verwenden von YUM/DNF beim Image-Build

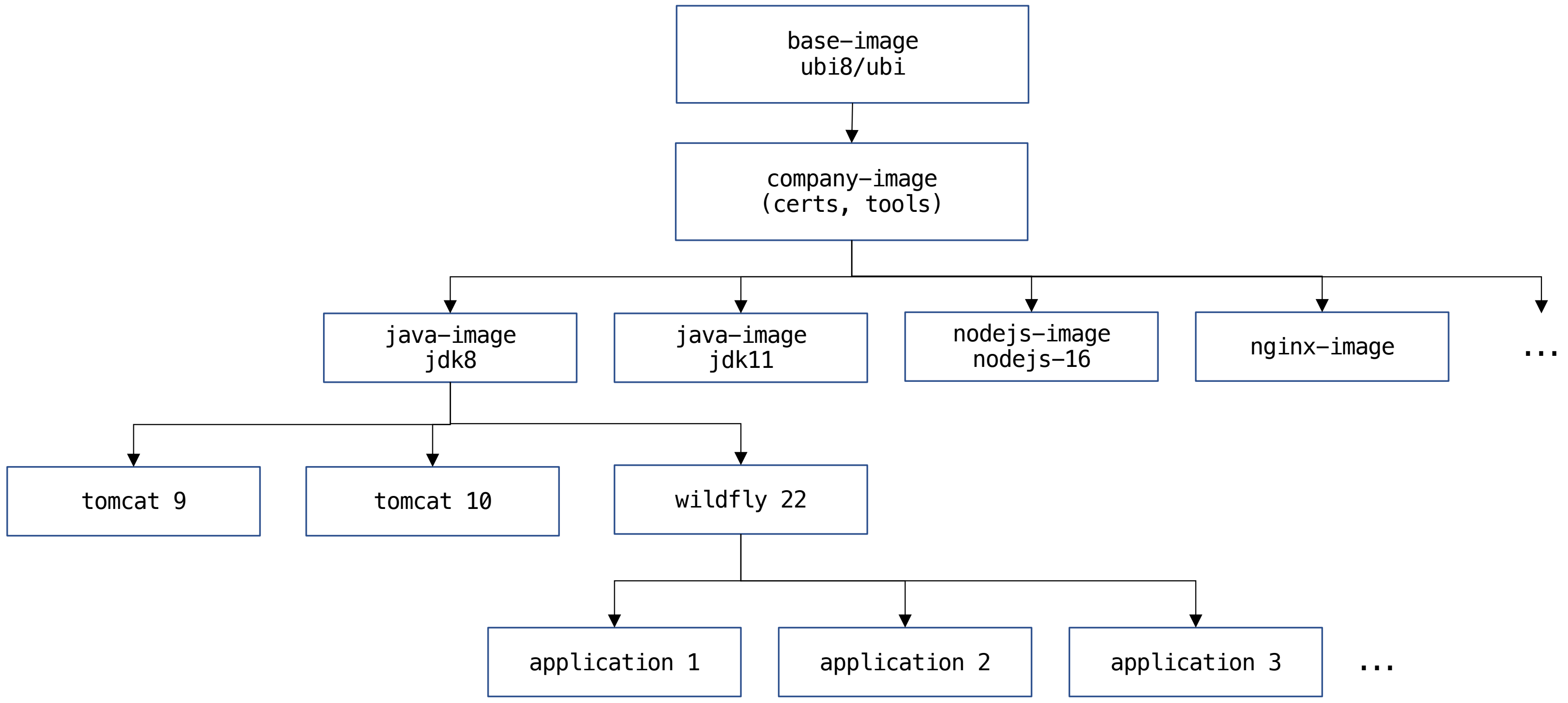
```
$ podman run --rm ubi8/ubi cat /etc/yum.repos.d/ubi.repo
[ubi-8-baseos]
name = Red Hat Universal Base Image 8 (RPMs) - BaseOS
baseurl = https://cdn-ubi.redhat.com/content/public/ubi/dist/ubi8/8/$basearch/baseos/os
enabled = 1
gpgkey = file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
gpgcheck = 1
...
```

yum "telefoniert"
nach aussen !

Lösung: beim podman-build andere yum-Konfiguration (z.B. vom Host) mounten
Bei Verwendung von Satellite/Subscriptions ggf. auch die notwendigen Zertifikate/GPG Schlüssel.

```
$ sudo podman build –v /etc/yum.repos.d:/etc/yum.repos.d –v /etc/pki:/etc/pki –v /etc/rhsm:/etc/rhsm .
```

Beispiel: Image – Vererbung



Änderungen an einem Basis-Image erfordern Rebuild der davon abhängigen Images !

image tree

Container in Openshift:

- beliebige User-Id RUN chmod - R 0770 ....
- Group-Id 0 (root) RUN chgrp –R 0
- Ports > 1024

```
apiVersion: project.openshift.io/v1
kind: Project
metadata:
  annotations:
    openshift.io/sa.scc.mcs: s0:c26,c15
    openshift.io/sa.scc.supplemental-groups: 1000680000/10000
    openshift.io/sa.scc.uid-range: 1000680000/10000
```

```
# oc exec pgadmin-778c479f79-tfbqn -- id
uid=1000680000(1000680000) gid=0(root) groups=0(root),1000680000
```

NFS-Mount →

```
# ls -al /mnt/nfs/apps/pgadmin
-rw-r--r-- 1 1000680000 root 124K Nov 27 01:03 access_log
-rw-r--r-- 1 1000680000 root  853 Nov 27 00:44 config_local.py
-rw-r--r-- 1 1000680000 root 1.2K Nov 27 00:46 error_log
```

https://cloud.redhat.com/blog/a-guide-to-openshift-and-uids

Abweichende User-Id : Serviceaccount mit Security Context Constraint 'anyuid' notwendig :

```
apiVersion:
rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: scc-anyuid
rules:
- apiGroups:
  - security.openshift.io
  resourceNames:
  - anyuid
  resources:
  - securitycontextconstraints
  verbs:
  - use
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: gitea:anyuid
  namespace: apps
roleRef:
  kind: ClusterRole
  name: scc-anyuid
  apiGroup: rbac.authorization.k8s.io
subjects:
- kind: ServiceAccount
  name: gitea
  namespace: apps
```

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: gitea
  namespace: apps
```

erstellt von Cluster-Administrator !

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: gitea
  namespace: apps
  ...
spec:
  template:
    spec:
      serviceAccountName: gitea
  ...
```

```
# oc exec gitea-7dcdc5c445-w9qmv -- id
uid=65534(nobody) gid=65534(nobody) groups=65534(nobody),0(root)

# ll /mnt/nfs/repos/ds
drwxr-xr-x 7 nobody nobody 119 Nov 26 16:57 admin.git/
drwxr-xr-x 7 nobody nobody 119 Nov 26 16:12 calibre.git/
drwxr-xr-x 7 nobody nobody 119 Nov 17 16:02 gitea.git/
...
```

UserId aus Container-Config !

Secrets:

- Passwörter, Token, Zertifikate ...
- typisiert: basic-auth, dockerfg, tls, opaque
- Inhalte sind base64-decodiert, nicht verschlüsselt

ConfigMap:

- generische Key-Value Daten

→ max. Größe 1 MB
→ nur innerhalb eines Project (NS) sichtbar

```
apiVersion: v1
kind: Secret
metadata:
  name: ...
  namespace: ...
data:
  password: MTIzNDU2
type: Opaque
```

```
# echo MTIzNDU2 | base64 -d
123456
```

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: ...
  namespace: ...
binaryData:
  keystore: |
    7oAMCAQICCF7Dt6ZDf6TgMA0GCSqGSIb3DQEBBQUAMEI1ZSQUla
    MTEQMA4GA1UECwwHU ...
data:
  HOME: /usr/share/nginx
  default.conf: |
    server {
      listen 8181 default_server;
      server_name _;
      location / {
         root   /usr/share/nginx/html;
         index  index.html index.htm;
      }
    }
```

```
$ oc create configmap <cm-name> --from-literal FOO=BAR

$ oc create configmap <cm-name> --from-file <path>

$ oc create secret docker-registry quayio --docker-server quay.io --docker-username <user> --docker-password <password>
```

Secrets: Verwendung als Umgebungs-Variable

```
apiVersion: v1
kind: Pod
metadata:
  name: secret-env-pod
spec:
  containers:
  - name: mycontainer
    image: redis
    env:
    - name: SECRET_USERNAME
      valueFrom:
        secretKeyRef:
           name: mysecret
           key: username
    - name: SECRET_PASSWORD
      valueFrom:
        secretKeyRef:
           name: mysecret
           key: password
```

ConfigMap: Verwendung als Konfigurations-Dateien

```
apiVersion: apps/v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    container: nginx
    volumeMounts:
    - mountPath: /etc/nginx/conf.d
      name: config
  volumes:
   - name: config
     configMap:
       name: nginx-config
```

```
apiVersion: apps/v1
kind: Pod
metadata:
  name: wildfly-standalone-xml
spec:
  containers:
  - name: wildfly
    container: nginx
    volumeMounts:
    - mountPath: /opt/wildfly/standalone/configuation
      name: standalone-xml
      subPath: standalone.xml
  volumes:
   - name: standalone-xml
     configMap:
       name: standalone-xml
```

```
$ oc set env deployment/<deployment-name> --from cm/<cm-name>

$ oc set volume deployment/<deployment-name> –add –t configmap –m /etc/nginx/conf.d --name config --configmap-name <cm-name>
```

Container Registry:

Red Hat → https://access.redhat.com/RegistryAuthentication

```
# podman login quay.io
Username: ...
Password: ...
Login Succeeded!     -> /run/user/<user-id>/containers/auth.json

# podman push --creds <username>:<password> ...

# skopeo --help
Various operations with container images and container image registries

Usage:
  skopeo [command]

Available Commands:
  copy                                          Copy an IMAGE-NAME from one location to another
  delete                                        Delete image IMAGE-NAME
  help                                          Help about any command
  inspect                                       Inspect image IMAGE-NAME
  list-tags                                     List tags in the transport/repository specified by the REPOSITORY-NAME
  login                                         Login to a container registry
  logout                                        Logout of a container registry
  manifest-digest                               Compute a manifest digest of a file
  standalone-sign                               Create a signature using local files
  standalone-verify                             Verify a signature using local files
  sync                                          Synchronize one or more images from one location to another
```

```
skopeo copy --format ... --dest-creds <user>:<password> containers-storage:localhost/webserver docker://quay.io/do288/webserver
```

Verwenden einer externen Container Registry - Authentifizierung

```
$ oc get serviceaccounts
NAME       SECRETS
builder    2
default    2
deployer   2
```

```
$ oc create secret docker-registry quayio --docker-server quay.io --docker-username <user> --docker-password <password>

$ oc secrets link default quayio --for pull

$ oc secrets link builder quayio
```

Verwenden einer externen Container Registry - Secret von auth.json

```
$ oc create secret generic quayio --from-file .dockerconfigjson=/run/user/1000/containers/auth.json --type kubernetes.io/dockerconfigjson
```

```
apiVersion: v1
kind: Secret
metadata:
  name: quayio
type: kubernetes.io/dockerconfigjson
data:
  .dockerconfigjson: ewogICJhdXRocyI6IHsKICAgICJyZWdpc3 ...
```

Serviceaccount 'imagePullSecrets' :

```
$ oc secrets link default quayio --for pull
```

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: default
imagePullSecrets:
- name: default-dockercfg-4sdrk
- name: quayio
...
```

oder im Deployment verwenden:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: pgadmin
spec:
  replicas: 1
  template:
    spec:
      imagePullSecrets:
      - name: redhat-sso
      containers:
      - name: pgadmin
        image: registry.connect.redhat.com/crunchydata/crunchy-pgadmin4
```

Verwenden der internen Registry :

```
$ oc patch configs.imageregistry.operator.openshift.io/cluster --patch '{"spec":{"defaultRoute":true}}' --type=merge
```

(Administrator)

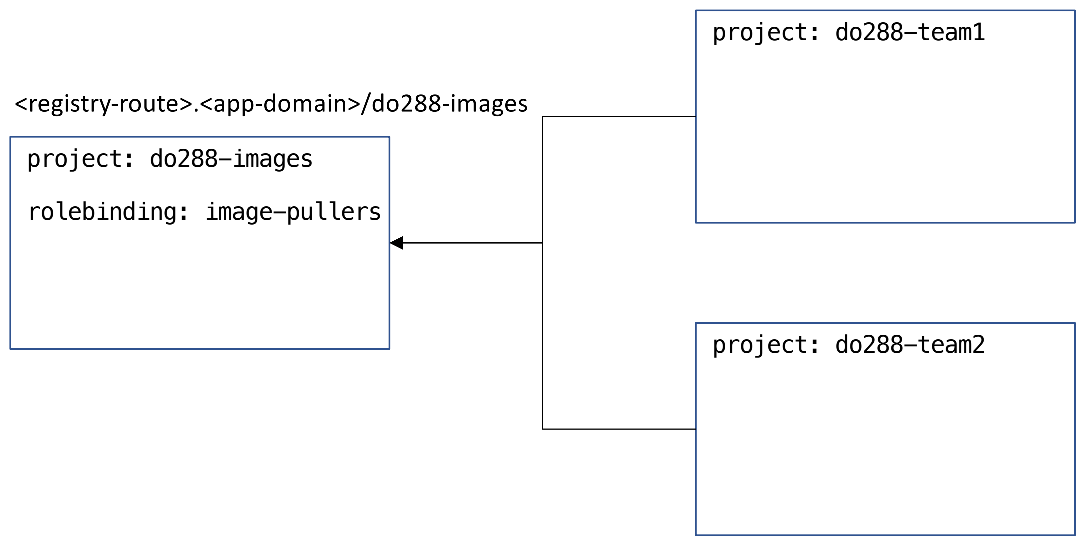
```
$ oc get route -n openshift-image-registry
NAME            HOST/PORT
default-route   default-route-openshift-image-registry.apps.eu46.prod.nextcle.com
```

Als Images-Repository wird der Namespace (Project) verwendet

```
default-route-openshift-image-registry.apps.eu46.prod.nextcle.com/<project>/<image>
```

```
$ skopeo list-tags docker://default-route-openshift-image-registry.apps.eu46.prod.nextcle.com/baseimages/rhel8
{
    "Repository": "default-route-openshift-image-registry.apps.eu46.prod.nextcle.com/baseimages/rhel8",
    "Tags": [
        "latest"
    ]
}

$ podman pull -creds $(oc whoami):$(oc whoami -t)
                    docker://default-route-openshift-image-registry.apps.eu46.prod.nextcle.com/baseimages/rhel8
Trying to pull docker://default-route-openshift-image-registry.apps.eu46.prod.nextcle.com/baseimages/rhel8...
Getting image source signatures
...
```

→ Zugriffsberechtigung !



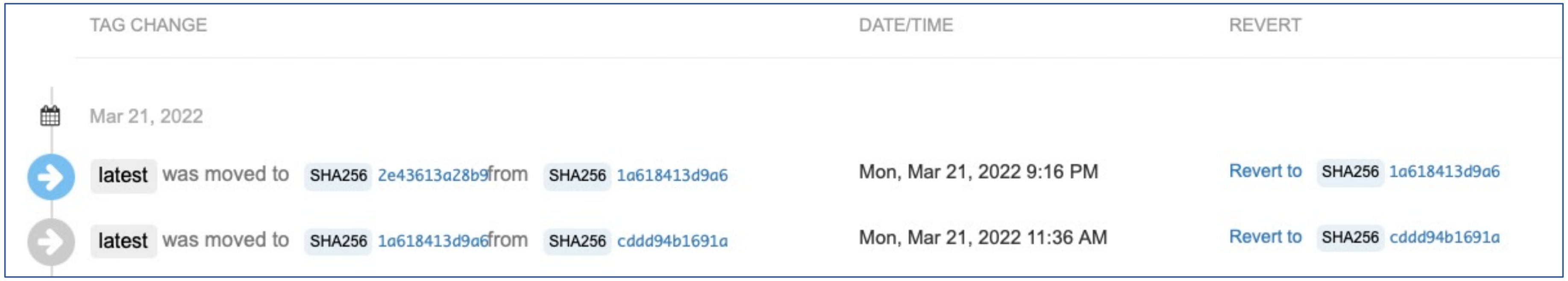
<registry-route>.<app-domain>/do288-images
project: do288-images
rolebinding: image-pullers
project: do288-team1
project: do288-team2

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: team:image-pullers
  namespace: do288-images
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: system:image-puller
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: system:serviceaccounts:do288-team1
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: system:serviceaccounts:do288-team2
```

Image Pull

Verwenden einer externen Container Registry – Imagestream aktualisieren

```
$ oc import-image webserver --from=quay.io/do288/webserver –confirm

$ oc describe is webserver
Name:                   webserver
...

latest
  tagged from quay.io/do288/webserver
  * quay.io/do288/webserver@sha256:1a618413d9a6cb45e37efc49a22cd08c5f702d6561483ed7dd2b38358e27fe10
```

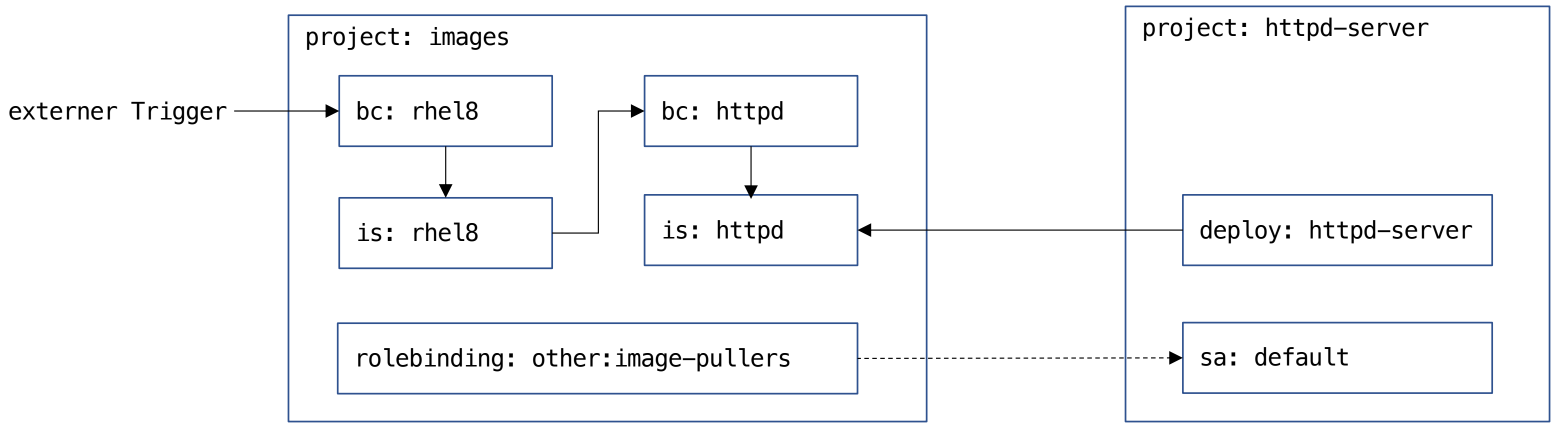


TAG CHANGE	DATE/TIME	REVERT
Mar 21, 2022		
latest was moved to SHA256 2e43613a28b9 from SHA256 1a618413d9a6	Mon, Mar 21, 2022 9:16 PM	Revert to SHA256 1a618413d9a6
latest was moved to SHA256 1a618413d9a6 from SHA256 cddd94b1691a	Mon, Mar 21, 2022 11:36 AM	Revert to SHA256 cddd94b1691a

```
$ oc tag quay.io/do288/webserver:latest webserver:latest

Tag webserver:latest set to quay.io/do288/webserver:latest

$ oc describe is webserver
...
latest
  tagged from quay.io/do288/webserver:latest

  * quay.io/do288/webserver@sha256:2e43613a28b9614208adef64202646718e534b29e77328762c656c85793d37a9
      54 seconds ago
    quay.io/do288/webserver@sha256:1a618413d9a6cb45e37efc49a22cd08c5f702d6561483ed7dd2b38358e27fe10
      7 minutes ago
```

Automatische Aktualisierung (15 min. Intervall) → `oc import-image webserver --from=quay.io/do288/webserver --scheduled`

Image-Change



```
$ curl -XPOST https://api.eu46.prod.nextcle.com:6443/apis/build.openshift.io/v1/namespaces/images/buildconfigs/rhel8/webhooks/abcdefg/generic
```

```
$ oc get pods -w -n images
NAME              READY    STATUS
httpd-1-build     0/1      Completed
rhel8-1-build     0/1      Completed
rhel8-2-build     1/1      Running
httpd-2-build     0/1      Pending
rhel8-2-build     0/1      Completed
httpd-2-build     1/1      Running
httpd-2-build     0/1      Completed
```

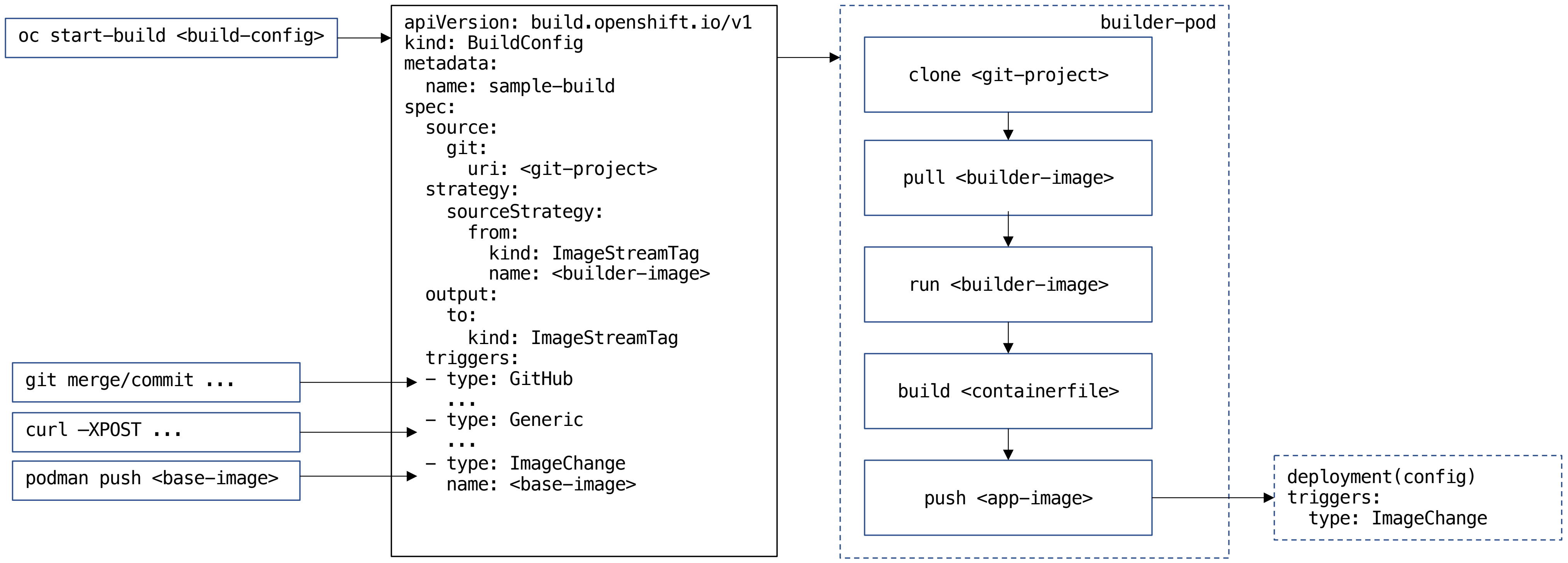
```
$oc get pods -w -n httpd-server
NAME                                READY    STATUS
httpd-server-77b6fc6595-2487q       1/1      Running
httpd-server-cf6489bfd-qh2hr        0/1      Pending
httpd-server-cf6489bfd-qh2hr        0/1      ContainerCreating
httpd-server-cf6489bfd-qh2hr        1/1      Running
httpd-server-77b6fc6595-2487q       0/1      Terminating
```

```yaml
apiVersion: build.openshift.io/v1
kind: BuildConfig
metadata:
  name: rhel8
  namespace: images
spec:
  source:
    dockerfile: |-
      FROM registry.access.redhat.com/ubi8/ubi:8.4
      ENV PACKAGES="lsof curl bind-utils"
      RUN dnf install -y --nodocs $PACKAGES && dnf clean all -y
    type: Dockerfile
  strategy:
    dockerStrategy: {}
    type: Docker
  output:
    to:
      name: rhel8:latest
      kind: ImageStreamTag
  successfulBuildsHistoryLimit: 1
  failedBuildsHistoryLimit: 1
  triggers:
  - type: Generic
    generic:
      secret: abcdefg
  - type: ConfigChange
```

Build-Management:

```
oc start-build <name>
oc cancel-build <name>
oc set env bc/<name> BUILD_LOGLEVEL="4"
```

```yaml
apiVersion: build.openshift.io/v1
kind: BuildConfig
metadata:
  name: httpd
  namespace: images
spec:
  source:
    type: Dockerfile
    dockerfile: |
      FROM xxxx
      RUN dnf install -y --nodocs httpd && dnf clean all -y
      ...
      EXPOSE 8080
      CMD /usr/sbin/httpd -DFOREGROUND
  strategy:
    type: Docker
    dockerStrategy:
      from:
        kind: ImageStreamTag
        namespace: images
        name: rhel8:latest
  successfulBuildsHistoryLimit: 1
  failedBuildsHistoryLimit: 1
  output:
    to:
      kind: ImageStreamTag
      name: httpd:latest
  triggers:
  - type: ImageChange
```



```
oc start-build <build-config>
```

```
git merge/commit ...
```

```
curl -XPOST ...
```

```
podman push <base-image>
```

```
apiVersion: build.openshift.io/v1
kind: BuildConfig
metadata:
  name: sample-build
spec:
  source:
    git:
      uri: <git-project>
  strategy:
    sourceStrategy:
      from:
        kind: ImageStreamTag
        name: <builder-image>
  output:
    to:
      kind: ImageStreamTag
  triggers:
  - type: GitHub
    ...
  - type: Generic
    ...
  - type: ImageChange
    name: <base-image>
```

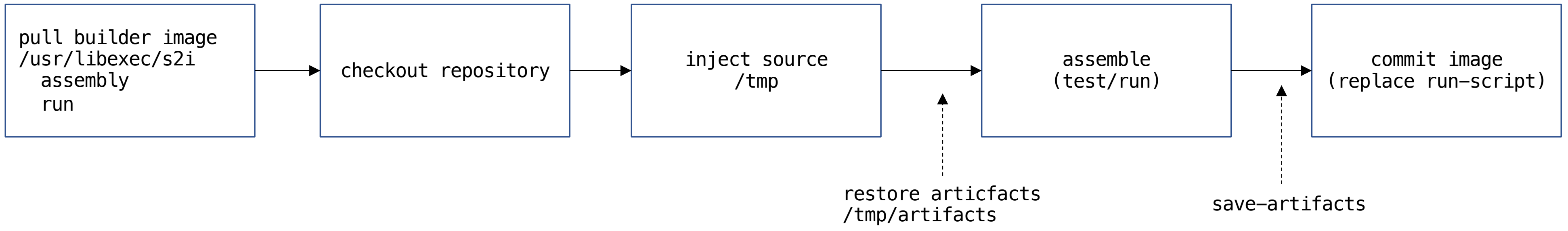
```
deployment(config)
triggers:
  type: ImageChange
```

Source: binary | dockerfile | git | images

Strategy:

- source : Builder-Image enthält Tools und Logik zum Erstellen einer Anwendung (Source2Image)
- docker : Git-Repository mit Dockerfile

```
$ oc set triggers bc/sample  --from-gitlab
buildconfig.build.openshift.io/sample triggers updated

$ oc describe bc/sample
...
Webhook GitHub:
  URL: https://api.eu46.prod.nextcle.com:6443/apis/build.openshift.io/v1/namespaces/dstraub-trigger/buildconfigs/sample/webhooks/<secret>/github
Webhook Generic:
  URL: https://api.eu46.prod.nextcle.com:6443/apis/build.openshift.io/v1/namespaces/dstraub-trigger/buildconfigs/sample/webhooks/<secret>/generic
Webhook GitLab:
  URL:  https://api.eu46.prod.nextcle.com:6443/apis/build.openshift.io/v1/namespaces/dstraub-trigger/buildconfigs/sample/webhooks/<secret>/gitlab

$ $ oc get bc/sample -o json | jq '.spec.triggers'
[
  {
    "github": {
      "secret": "uUOxcyrsg4h58ThACUJj"
    },
    "type": "GitHub"
  },
  {
    "generic": {
      "secret": "Sfyo-MeJUbRGFS-3fOQX"
    },
    "type": "Generic"
  },
  ...
  {
    "gitlab": {
      "secret": "krhFoEzyiorD1UEZt_o5"
    },
    "type": "GitLab"
  }
]


$ curl -XPOST https://api.eu46.prod.nextcle.com:6443/apis/build.openshift.io/v1/namespaces/dstraub-trigger/buildconfigs/sample/webhooks/Sfyo-MeJUbRGFS-
3fOQX/generic
```

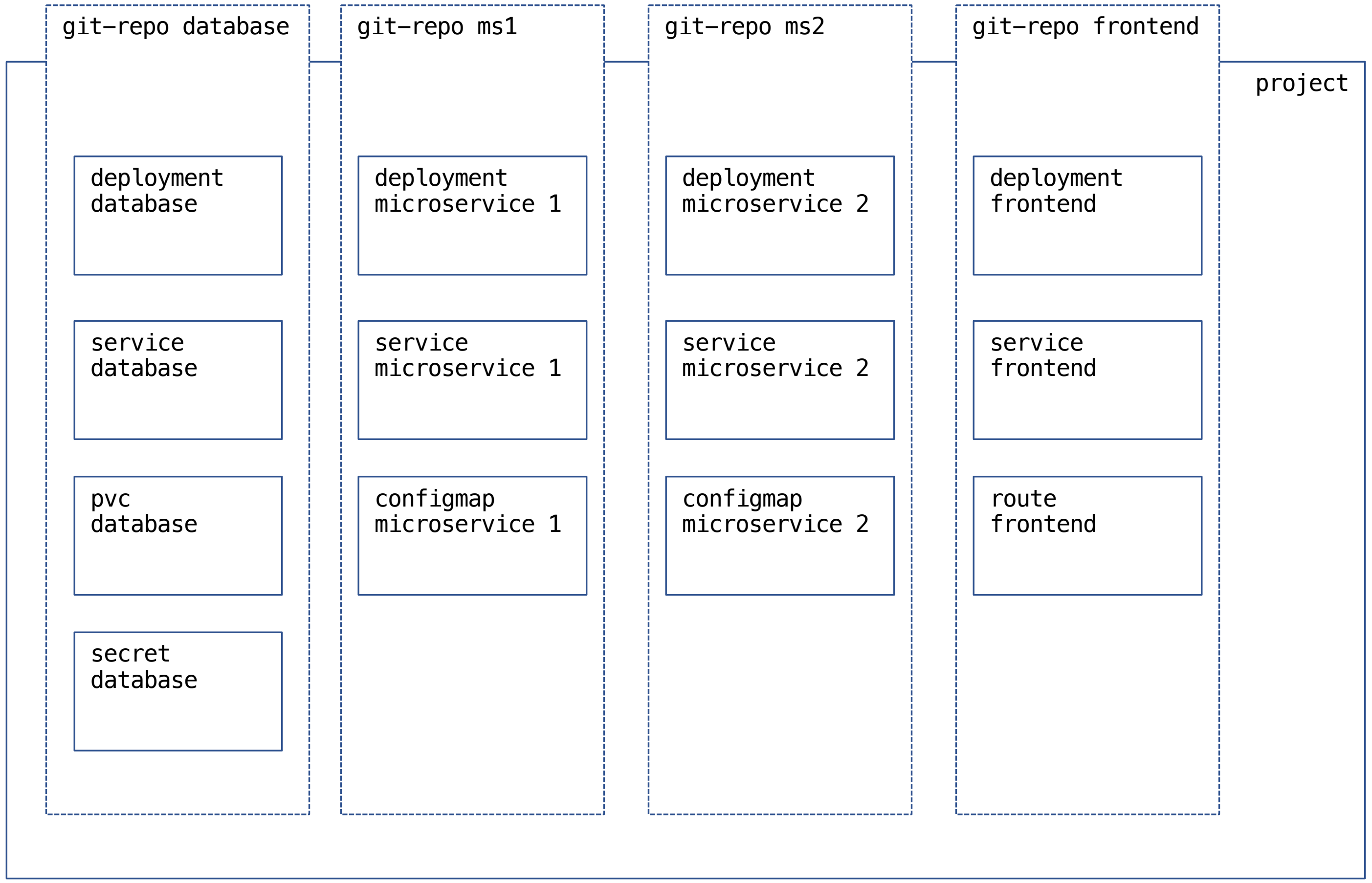


Build-Scripte:

- default in `/usr/libexec/s2i`
- `assemble` und `run` sind mandatory
- `save-artifacts`, `usage`, `test/run` sind optional
- könnnen überschrieben werden im Git-Repo `.s2i/bin`
  (Wrapper um Original-Script oder komplett neues Script)

Incremental Builds:

- `save-artifacts` erstellt TAR
- wird vor dem Ausführen von assembly injected in `/tmp/artifacts`



Multi Container App

Template: parametrisierbare Liste von Resource-Definitionen

```
kind: Template
apiVersion: v1
metadata:
  name: rest-sample
objects:
- apiVersion: v1
  kind: Service
  metadata:
    name: ${APP_NAME}
  spec:
    selector:
      app.kubernetes.io/name: ${APP_NAME}
   ...
- apiVersion: apps/v1
  kind: Deployment
  metadata:
    name: ${APP_NAME}
  spec:
    template:
      spec:
        containers:
        - name: ${APP_NAME}
          image: ${IMAGE_NAME}
...
- apiVersion: v1
  kind: Route
...
parameters:
- description: Application Name
  name: APP_NAME
  required: true
- description: Image Name
  name: IMAGE_NAME
  required: true
...
```

```
oc process (TEMPLATE | -f FILENAME) -p APP_NAME=... | oc create -f -

oder bei installiertem Template ( oc create -f template.yml) :

oc new-app <template-name>
```

```
apiVersion: template.openshift.io/v1
kind: Template
labels:
  app: php-sample                                   ← Labels für alle Objekte
metadata:
  name: php-sample
  labels:
    samples.operator.openshift.io/managed: "true"   ← Labels nur für das Template
    app: php-sample
objects:
- apiVersion: v1
  kind: Service
  metadata:
    annotations:
      description: Exposes and load balances the application pods
  ...
```

```
oc process -f template.yml -o yaml
apiVersion: v1
items:
- apiVersion: v1
  kind: Service
  metadata:
    labels:
      app: php-sample
  ...
- apiVersion: route.openshift.io/v1
  kind: Route
  metadata:
    labels:
      app: php-sample
  ...
- apiVersion: image.openshift.io/v1
  kind: ImageStream
  metadata:
   labels:
      app: php-sample
  ...
```

```
$ oc create -f template.yml
template.template.openshift.io/php-sample created

$ oc get template php-sample -o yaml
apiVersion: template.openshift.io/v1
kind: Template
labels:
  app: php-sample
metadata:
  labels:
    samples.operator.openshift.io/managed: "true"
...


$ oc delete all -l app=php → alle vom Template erzeugten Objekte werden gelöscht
```

Helm-Chart: Paket-Manager (Lifecycle + Template-Engine + Dependencies)

```
$ helm create sample
Creating sample

$ tree sample
sample
├── charts
├── Chart.yaml
├── templates
│   ├── deployment.yaml
│   ├── _helpers.tpl
│   ├── hpa.yaml
│   ├── ingress.yaml
│   ├── NOTES.txt
│   ├── serviceaccount.yaml
│   ├── service.yaml
│   └── tests
│       └── test-connection.yaml
└── values.yaml
```

helm create

Helm-Chart: Paket-Manager (Lifecycle + Template-Engine + Dependencies)

Chart.yml
```
apiVersion: v1
name: sample
description: Sample Application
version: 1.0
appVersion: 1.0
dependencies:
- name: dep1
  version: ...
  repository: ...
```

values.yml
```
image:
  repository: quay.io/redhat.io/sample
  tag: '2'

service:
  port: 8080

env:
  ...


dep1.key: value
```

Templates:

deployment.yml
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ APP_NAME }}
spec:
  template:
    selector:
      matchLabels:
        {{- include "sample.selectorLabels" . | nindent 6 }}
     spec:
       containers:
         - image: {{.Values.image.repository}}: {{.Values.image.tag}}
 ...
```

Go-Templates:

_helpers.tpl
```
{{- define "sample.selectorLabels" -}}
app.kubernetes.io/name: {{ include "sample.name" . }}
app.kubernetes.io/instance: {{ .Release.Name }}
{{- end }}
...
```

```
helm create
helm dependency update
helm install / upgrade / rollback / uninstall

helm template (lokales processing)
```

Kustomize: generieren/transformieren von Resourcen (Manifeste mit minimalen Meta-Daten)

kustomization.yml

```
kind: Kustomization
apiVersion: kustomize.config.k8s.io/v1beta1

namespace: sample

resources:
- deployment.yml
- service.yml
- route.yml
- http://...    -> kustomize.yml in Git/Web-Repository


images:
- name: sample
  newName: registry/sample
  newTag: '5'

commonLabels:
  app.kubernetes.io/instance: sample

configMapGenerator:
- name: rest-sample
  literals:
  - LAUNCH_JBOSS_IN_BACKGROUND=1
...
```

deployment.yml

```
apiVersion: apps/v1
metadata:
  name: rest-sample
spec:
  replicas: 1
  template:
    spec:
      containers:
      - name: sample
        image: sample
```

```
$ oc kustomize <kustom-dir>
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app.kubernetes.io/instance: rest-sample
  name: rest-sample
  namespace: sample
spec:
  replicas: 1
  selector:
    matchLabels:
      app.kubernetes.io/instance: sample
  template:
    containers:
      image: registry/sample:5
...

$ oc apply -k .
```

resources → https://github.com/hashicorp/go-getter#url-format

Kustomize Overlays : erzeugen unterschiedlicher Varianten von einer Basis-Vorlage

```
                                        base/kustomization.yml
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization

resources:
- deployment.yml
- service.yml
- route.yml
```

```
                                overlays/test/kustomization.yml
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization

resources:
- ../../base

namespace: test

images:
- name: sample
  newName: registry/sample
  newTag: '3-SNAPSHOT'
```

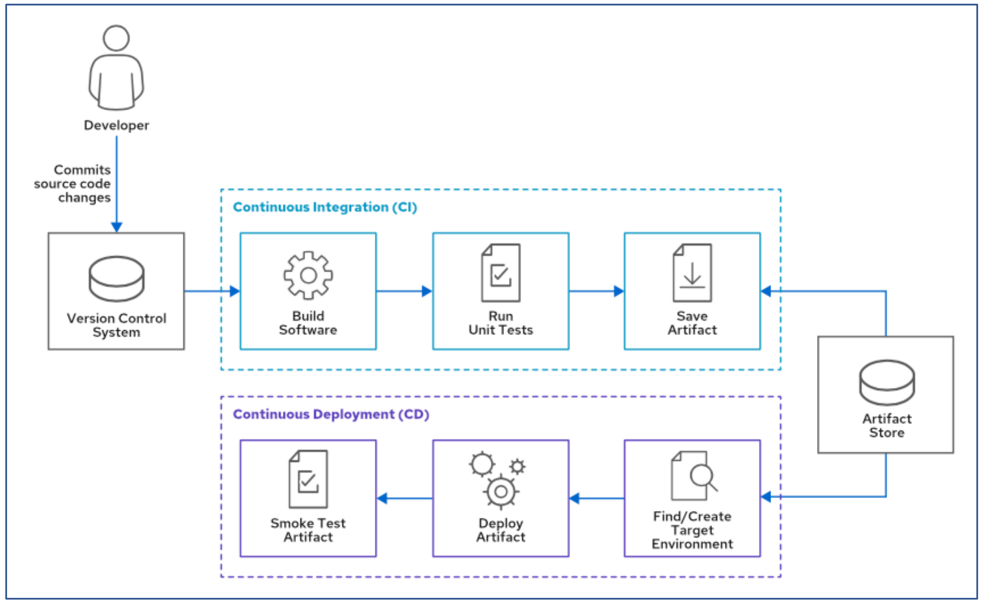
```
                          overlays/production/kustomization.yml
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization

resources:
- ../../base

namespace: production

images:
- name: sample
  newName: registry/sample
  newTag: '5'
```

```
$ oc apply -k overlays/test
service/sample configured
deployment.apps/sample configured
route.route.openshift.io/sample configured

$ oc apply -k overlays/production
...
```

https://kubectl.docs.kubernetes.io/guides/extending_kustomize/exec_krm_functions

```
$ grep -A3 images kustomization.yml
images:
- name: webserver
  newName: quay.io/danielstraub/webserver
  newTag: "1.0"

$ kustomize edit set image webserver=quay.io/danielstraub/webserver:2.0

$ grep -A3 images kustomization.yml
images:
- name: webserver
  newName: quay.io/danielstraub/webserver
  newTag: "2.0"

$ oc apply -k .
configmap/webserver-kt5mdg45d2 unchanged
service/webserver unchanged
deployment.apps/webserver configured
route.route.openshift.io/webserver unchanged

$ curl https://stage-prod.apps.eu46a.prod.ole.redhat.com
Hello, DO288
Version 2.0
```

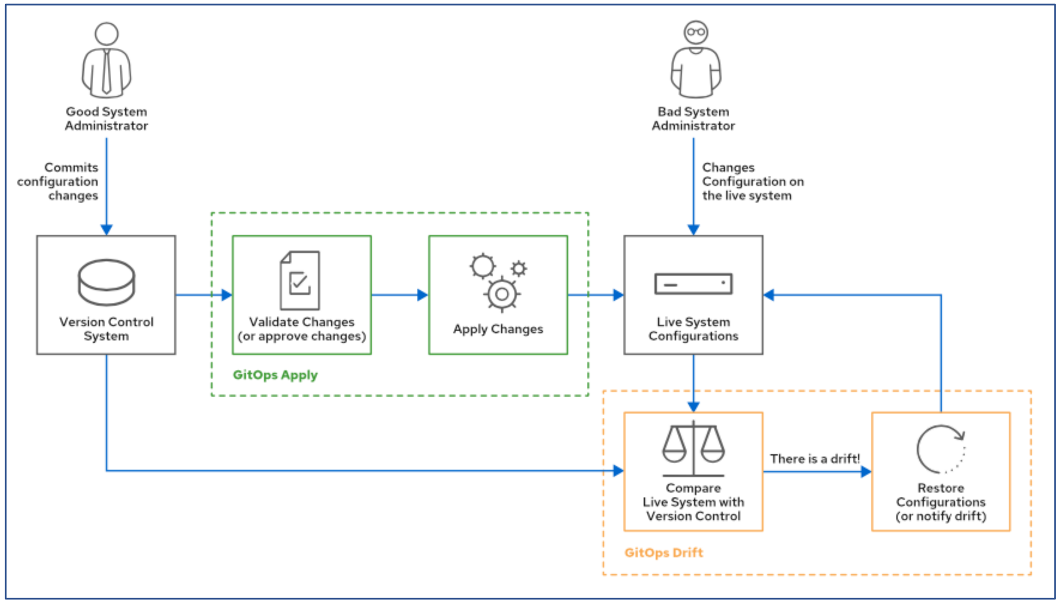
kustomize edit



Continuos Integration
Continuos Delivery

→ Developer
→ running application

Jenkins, CruiseControl, TeamCity, GitLab ...
Kubernetes native (Tekton, Argo CD, ...)


Good System Administrator
Commits configuration changes
Bad System Administrator
Changes Configuration on the live system
Version Control System
Validate Changes (or approve changes)
Apply Changes
GitOps Apply
Live System Configurations
Compare Live System with Version Control
There is a drift!
Restore Configurations (or notify drift)
GitOps Drift

GitOps Workflow

→ Administrators
→ live System

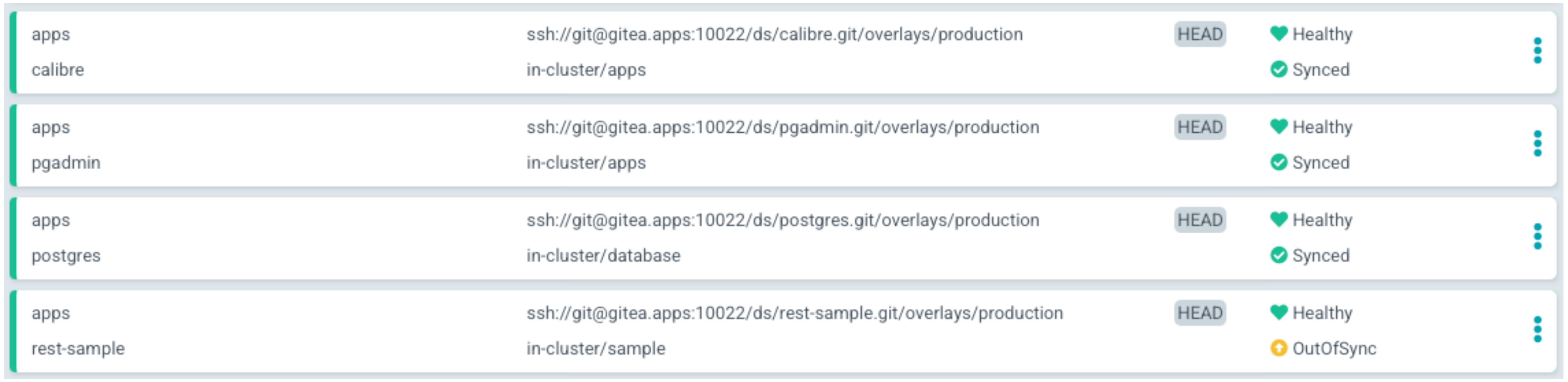
Ansible, Puppet, Terraform ...
ArgoCD, FluxCD, JenkinsX

GitOps – Workflow mit Pipelines:

- Apply Pipeline:
  ```
  – validate : oc apply --validate --dry-run [ folder/files from Git ]
  – apply    : oc apply
  ```
- Drift Pipeline:
  ```
  – diff     : oc diff [ folder/files from Git ]
  – optional/restore: oc apply
  ```

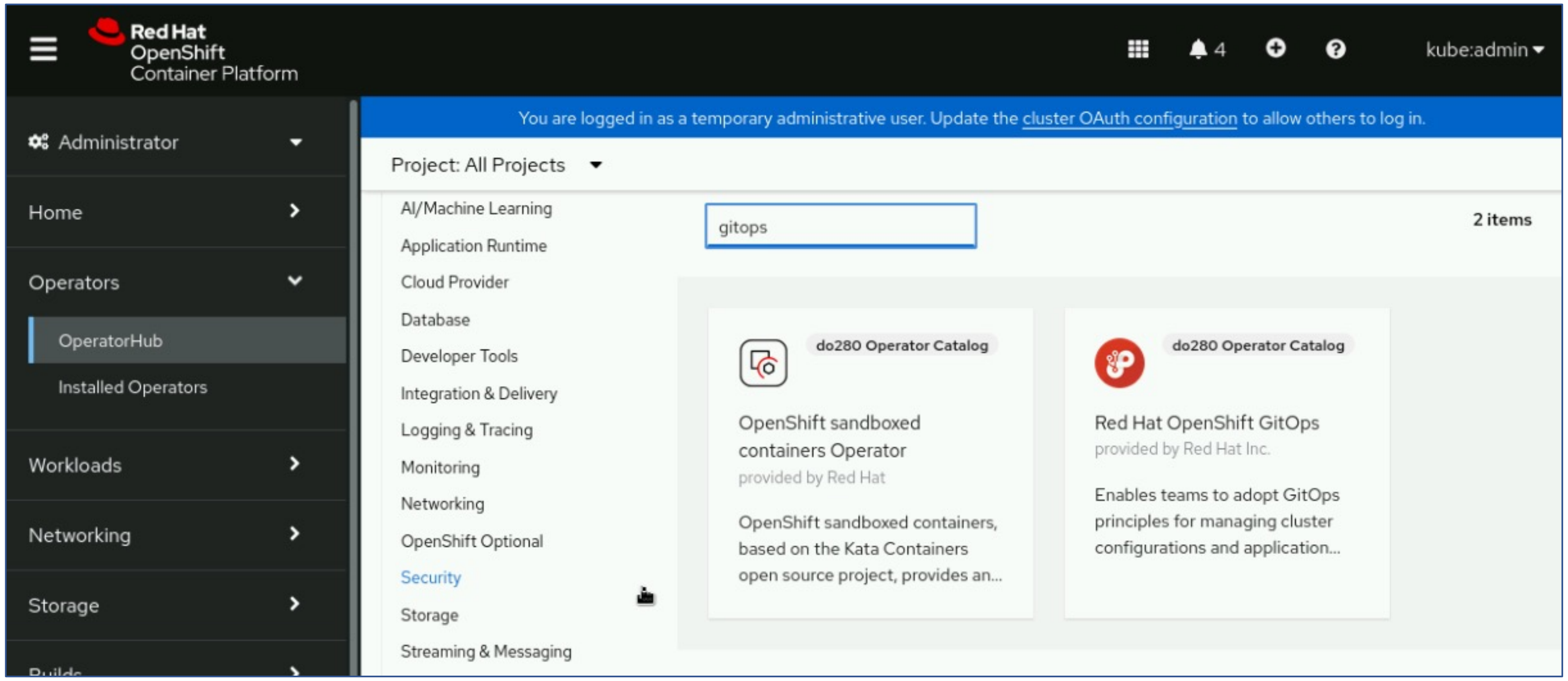
GitOps – Workflow mit ArgoCD / FluxCD:

Abgleich Ist-Zustand (Cluster) mit Kustomize/Helm-Definitionen im Git
Benachrichtigungen, manueller/automatische Synchronisation bei Abweichungen



apps calibre	ssh://git@gitea.apps:10022/ds/calibre.git/overlays/production in-cluster/apps	HEAD	Healthy Synced
apps pgadmin	ssh://git@gitea.apps:10022/ds/pgadmin.git/overlays/production in-cluster/apps	HEAD	Healthy Synced
apps postgres	ssh://git@gitea.apps:10022/ds/postgres.git/overlays/production in-cluster/database	HEAD	Healthy Synced
apps rest-sample	ssh://git@gitea.apps:10022/ds/rest-sample.git/overlays/production in-cluster/sample	HEAD	Healthy OutOfSync

Red Hat Openshift GitOps - Operator

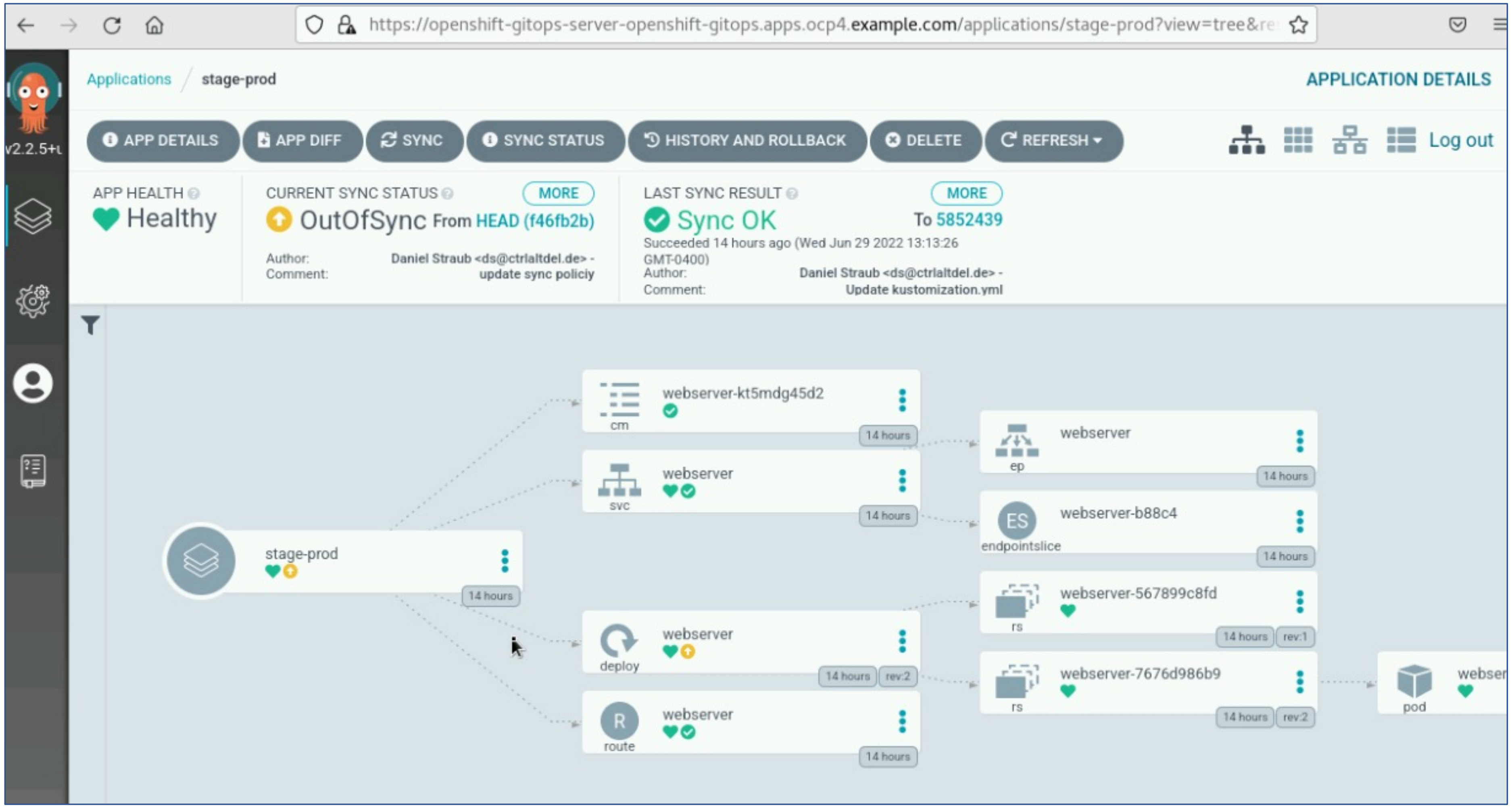


You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to log in.

OpenShift sandboxed containers Operator
provided by Red Hat
OpenShift sandboxed containers, based on the Kata Containers open source project, provides an...

Red Hat OpenShift GitOps
provided by Red Hat Inc.
Enables teams to adopt GitOps principles for managing cluster configurations and application...

ArgoCD = Openshift GitOps



APP HEALTH
Healthy

CURRENT SYNC STATUS MORE
OutOfSync From HEAD (f46fb2b)
Author: Daniel Straub <ds@ctrlaltdel.de> -
Comment: update sync policiy

LAST SYNC RESULT MORE
Sync OK To 5852439
Succeeded 14 hours ago (Wed Jun 29 2022 13:13:26 GMT-0400)
Author: Daniel Straub <ds@ctrlaltdel.de> -
Comment: Update kustomization.yml

Openshift GitOps

```
$ oc get pods -w
NAME                                    READY   STATUS              RESTARTS   AGE
famousapp-famouschart-65744d4c8b-4zqhn   0/1     Running             0          10s
famousapp-mariadb-0                      0/1     ContainerCreating   0          10s
famousapp-mariadb-0                      0/1     Running             0          11s
famousapp-famouschart-65744d4c8b-4zqhn   0/1     Running             1          33s
famousapp-famouschart-65744d4c8b-4zqhn   0/1     Error               1          34s
famousapp-famouschart-65744d4c8b-4zqhn   0/1     Running             2          35s
famousapp-famouschart-65744d4c8b-4zqhn   0/1     Error               2          36s
famousapp-famouschart-65744d4c8b-4zqhn   0/1     CrashLoopBackOff    2          37s
famousapp-mariadb-0                      1/1     Running             0          48s
famousapp-famouschart-65744d4c8b-4zqhn   0/1     Running             3          56s
famousapp-famouschart-65744d4c8b-4zqhn   1/1     Running             3          62s
```

```
metadata:
  name: famousapp-mariadb:
...

  livenessProbe:
    exec:
      command:
        - /bin/bash
        - -ec
        - |
          password_aux="${MARIADB_ROOT_PASSWORD:-}"
          if [[ -f "${MARIADB_ROOT_PASSWORD_FILE:-}"
             password_aux=$(cat "$MARIADB_ROOT_PASSW
          fi
          mysqladmin status -uroot -p"${password_aux
```

```
metadata:
  name: famousapp-famouschart
...
   livenessProbe:
      initialDelaySeconds: 30
      httpGet:
        path: /
      ...
   readinessProbe:
      failureThreshold: 3
      httpGet:
        path: /
        port: http
        scheme: HTTP
        periodSeconds: 10
        successThreshold: 1
        timeoutSeconds: 1
```

Liveness / Readiness / Startup Probes

liveness :	Container wird bei negativen Ergebnis neu gestartet	`.spec.containers.livenessProbe`
readiness:	Route/Service wird aktiviert/deaktiviert	`.spec.containers.readinessProbe`
startup:	liveness/readiness sind deaktiviert bis startup positiv ist Container wird bei neg. Startup-Probe sofort beendet	`.spec.containers.startupProbe`

Probes:

```
exec:
  command:
  - cat
  - /tmp/ready
initialDelaySeconds: 5
periodSeconds: 5
```

```
httpGet:
  path: /healthz
  port: healthz-port
  schema: https
  httpHeaders: ...
failureThreshold: 1
periodSeconds: 10

200 <= status < 400
```

```
tcpSocket:
  port: 5432
initialDelaySeconds: 15
periodSeconds: 20
```

initialDelaySeconds:	Zeitdauer bis zur ersten liviness/readiness Probe
periodSeconds:	Intervall zur Ausführung der Proben (default 10 sec)
timeoutSeconds:	max. Timeout bei einer Probe (default 1 sec)
successThreshold:	Schwellwert ab wann aufeinderfolgende positive Proben als Erfolg gewertet werden (default 1)
failureThreshold:	Schwellwert ab wann aufeinderfolgende negative Proben als Ausfall gewertet werden (default 3)

```yaml
kind: Deployment
apiVersion: apps/v1
metadata:
  name: webserver
spec:
  ...
  template:
    spec:
      containers:
      - name: webserver
        image: webserver
        imagePullPolicy: Always
        ports:
        - name: http
          containerPort: 8080
          protocol: TCP
        readinessProbe:
          failureThreshold: 3
          httpGet:
            path: /healthz
            port: http
            scheme: HTTP
          periodSeconds: 10
          successThreshold: 1
          timeoutSeconds: 1
        ...
```

```
                                    nginx.conf
server {
  listen 8080 default_server;
  server_name _;
  location / {
    root   /usr/share/nginx/html;
    index  index.html index.htm;
  }

  location /healthz {
    access_log off;
    return 200;
  }

}
```

readinessProbe

(compute) Resources :

- Memory: number of bytes (quantity suffixes: E, P, T, G, M, k | Ei, Pi, Ti, Gi, Mi, Ki)

- CPU : millicores (m)

  millicores are the fractions of *time* of a single CPU (not the fraction of number of CPUs).
  Cgroups, and hence Docker, and hence Kubernetes, doesn't restrict CPU usage by assigning cores to processes (like VMs do), instead it restricts CPU usage by restricting the amount of time (quota over period) the process can run on each CPU (with each CPU taking up to 1000mcpus worth of allowed time).

  https://stackoverflow.com/questions/61851751/multi-threading-with-millicores-in-kubernetes

```
apiVersion: v1
kind: Pod
metadata:
...
spec:
  containers:
  - name: <name>
    resources:
      requests:
        memory: 64Mi
        cpu: 100m
      limits:
        memory: 128Mi
        cpu: 200m
```

Scheduling

Execution
(cgroups)

```
$ oc describe node master01
...
Allocatable:
  cpu:                   3500m
  memory:                15268156Ki
Non-terminated Pods: (60 in total)
   CPU Requests CPU Limits Memory Requests Memory Limits
...
Allocated resources:
Resource            Requests      Limits
  --------            --------      ------
  cpu                 2397m (68%)   0 (0%)
  memory              9347Mi (62%)  512Mi (3%)
```

resouce usage

Pod – Scheduling

1. Filter
   - Node-Selector für Labels
     https://kubernetes.io/docs/reference/labels-annotations-taints
   - Toleration für Taints
     https://kubernetes.io/docs/concepts/scheduling-eviction/taint-and-toleration

```
apiVersion: v1
kind: Pod
metadata:
...
spec:
  containers:
  - name: nginx
    nodeSelector:
      disktype: ssd
  tolerations:
  - key: class
    value: do280
    operator: "Equal"
    effect: "NoSchedule"
```

```
apiVersion: v1
kind: Node
metadata:
 labels:
   disktype: ssd
spec:
  taints:
  - key: class
    value: do280
    effect: NoSchedule
```

Pod – Scheduling

2. Scoring
   Affinity/Anti-Affinity-Rules

```
apiVersion: v1
kind: Pod
metadata:
  name: with-node-affinity
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: kubernetes.io/os
            operator: In
            values:
            - linux
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 1
        preference:
          matchExpressions:
          - key: another-node-label-key
            operator: In
            values:
            - another-node-label-value
  containers:
  - name: with-node-affinity
    image: ...
```

```
...DuringScheduling: während des Scheduling
IgnoredDuringExecution: Pod wird weiter ausgeführt,
    auch wenn sich nach dem Scheduling Node-Labels ändern
```

https://kubernetes.io/docs/concepts/scheduling-eviction/assign-pod-node
https://www.cncf.io/blog/2021/07/27/advanced-kubernetes-pod-to-node-scheduling

Pod Verteilung auf unterschiedliche Nodes:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  ...
spec:
  selector:
    matchLabels:
      app: store
  replicas: 3
  template:
    metadata:
      labels:
        app: store
    spec:
      affinity:
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
              matchExpressions:
              - key: app
                operator: In
                values:
                - store
            topologyKey: "kubernetes.io/hostname"
      containers:
      - ...
```

gleiche Label und gleicher Hostname → AntiAffinity

Pods werden auf unterschiedlichen Nodes verteilt

https://docs.openshift.com/container-platform/4.10/nodes/scheduling/nodes-scheduler-pod-affinity.html#nodes-scheduler-pod-affinity-example-antiaffinity_nodes-scheduler-pod-affinity

DeploymentConfig | Deployment

```
kind: DeploymentConfig
metadata:
  name: ...
spec:
  replicas: 1
  selector:
     app: ...
  template:
    metadata:
    ...
  spec:
     strategy:
       rollingParams:
          pre:
          mid:
          post:
     containers:
     - name: <container_name>
       image: image-registry.openshift-image-registry.svc:5000/<name_space>/<image>:@sha256:xxxx
       imagePullPolicy: IfNotPresent
       ...
     triggers:
     - type: ConfigChange
     - type: ImageChange
       imageChangeParams:
         containerNames:
         - <container_name>
         from:
           name: <image_stream>:<image_tag>
```

Automatisches Redeployment bei Konfigurations-Änderungen oder neues Image im verknüpften Imagestream

```
kind: Deployment
metadata:
  name: <name>
  annotations:
     image.openshift.io/triggers: '{"from": {"kind": "ImageStreamTag","name": "<image_stream>:<image_tag>"},
           "fieldPath: "spec.template.spec.containers[0].image" }'
spec:
```

Deployment-Strategien
- Rolling Updates : Pods werden der Reihe nach aktualisiert
- Recreate: existierende Pods werden beendet und neue gestartet

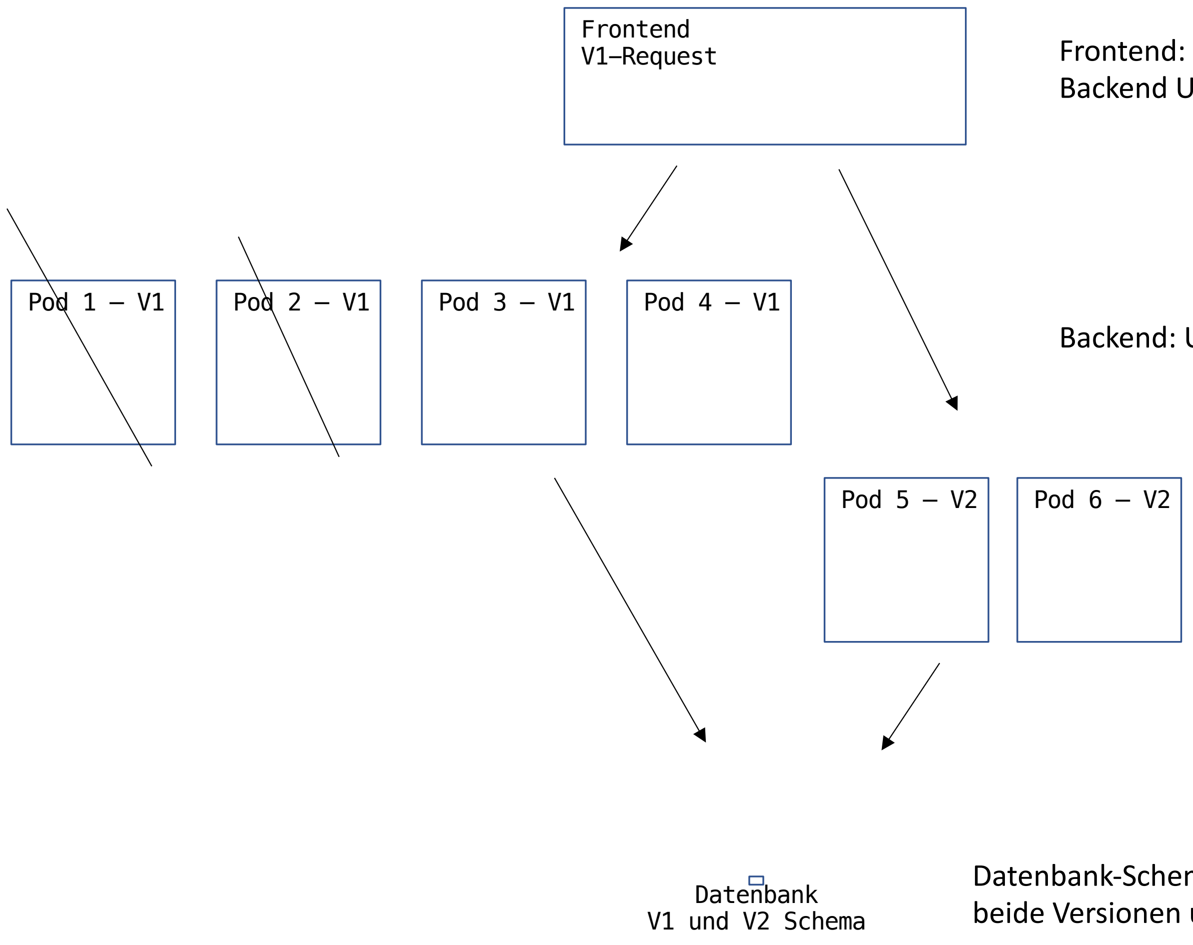
DeploymentConfig:
- Pre/Mid/Post – Lifecycle Hooks

Beenden eines Pods:
- SIGTERM: Pod soll keine neuen Verbindungen annehmen und bestehenden Aktionen beenden
- SIGKILL: nach `terminationGracePeriodSeconds` (30s) wird der Pod beendet

```
kind: Deployment
metadata:
  name: ...
spec:
  revisionHistoryLimit: 3   (default: 10)
  replicas: 4
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1            ← max. 5 Pods aktiv
      maxUnavailable: 1
  ...
  template:
    spec:
      containers:
      - ...
      terminationGracePeriodSeconds: 30
```

```
oc rollout SUBCOMMAND (DEPLOYMENTCONFIG | DEPLOYMENT)

  cancel      Cancel the in-progress deployment
  history     View rollout history
  latest      Start a new rollout for deployment config with latest state
  pause       Mark the provided resource as paused
  restart     Restart a resource
  resume      Resume a paused resource
  retry       Retry the latest failed rollout
  status      Show the status of the rollout
  undo        Undo a previous rollout

oc rollback (DEPLOYMENTCONFIG | DEPLOYMENT) [--to-version=]
```

N-1 Abwärtskompatibilität bei Rolling-Update:



Frontend: behält V1 bis Backend Update fertig ist

Backend: Update von V1 auf V2

Datenbank-Schema muss auch beide Versionen unterstützen

A/B Deployment Strategy:

```
apiVersion: v1
kind: Service
metadata:
  name: service-a
spec:
ports:
  - name: http
    port: 80
    protocol: TCP
    targetPort: http
  selector:
    app.kubernetes.io/instance: deploment-a
```

```
apiVersion: v1
kind: Service
metadata:
  name: service-b
spec:
ports:
  - name: http
    port: 80
    protocol: TCP
    targetPort: http
  selector:
    app.kubernetes.io/instance: deploment-b
```

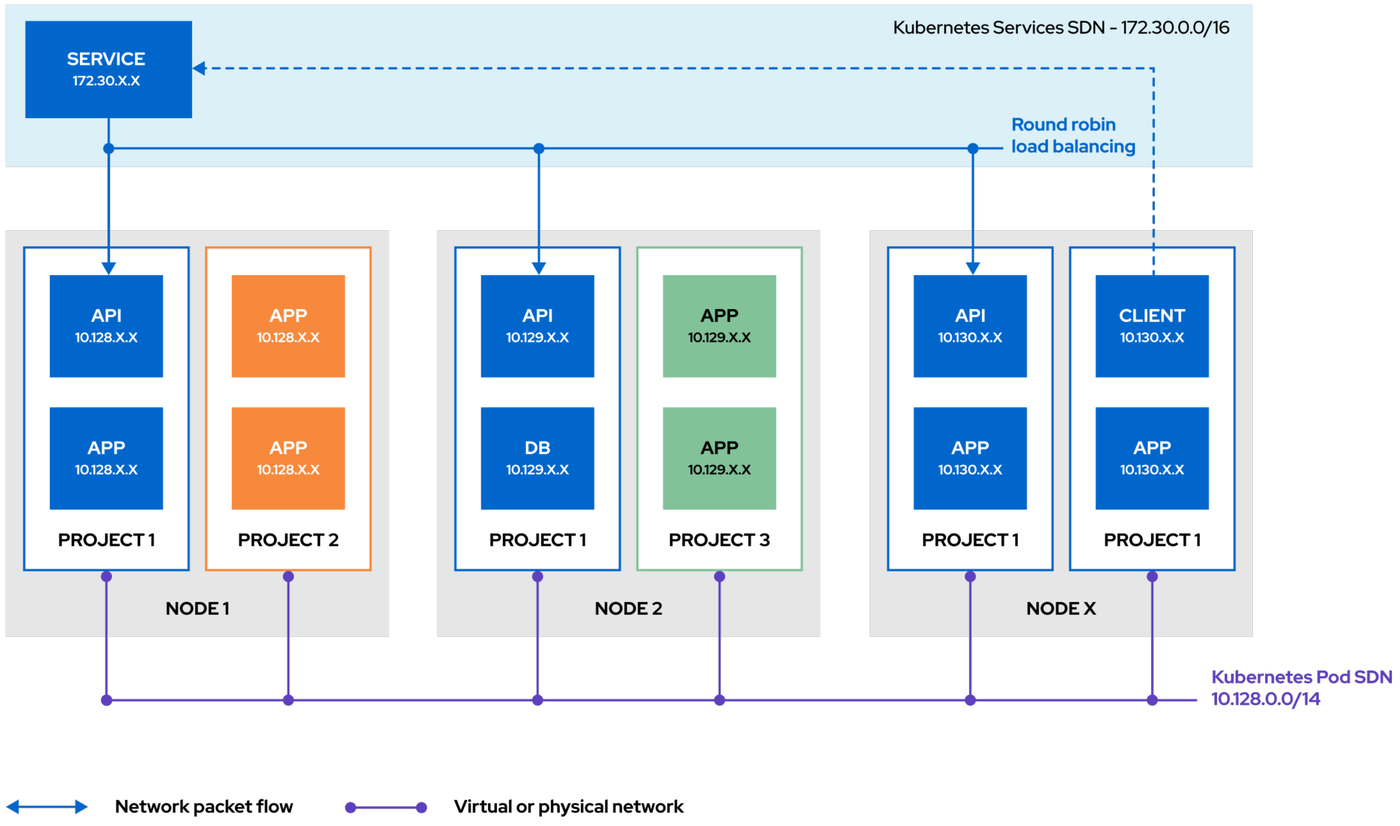
```
kind: Route
metadata:
  name: <name>
spec:
  host: <host>
  to:
    kind: Service
    name: service-a
    weight: 50
  alternateBackends:
  - kind: Service
    name: service-b
    weight: 200
```

Service :

- ClusterIP
- NodePort
- Loadbalancer (AWS,Azure, MetalLB ...)
- ExternalName

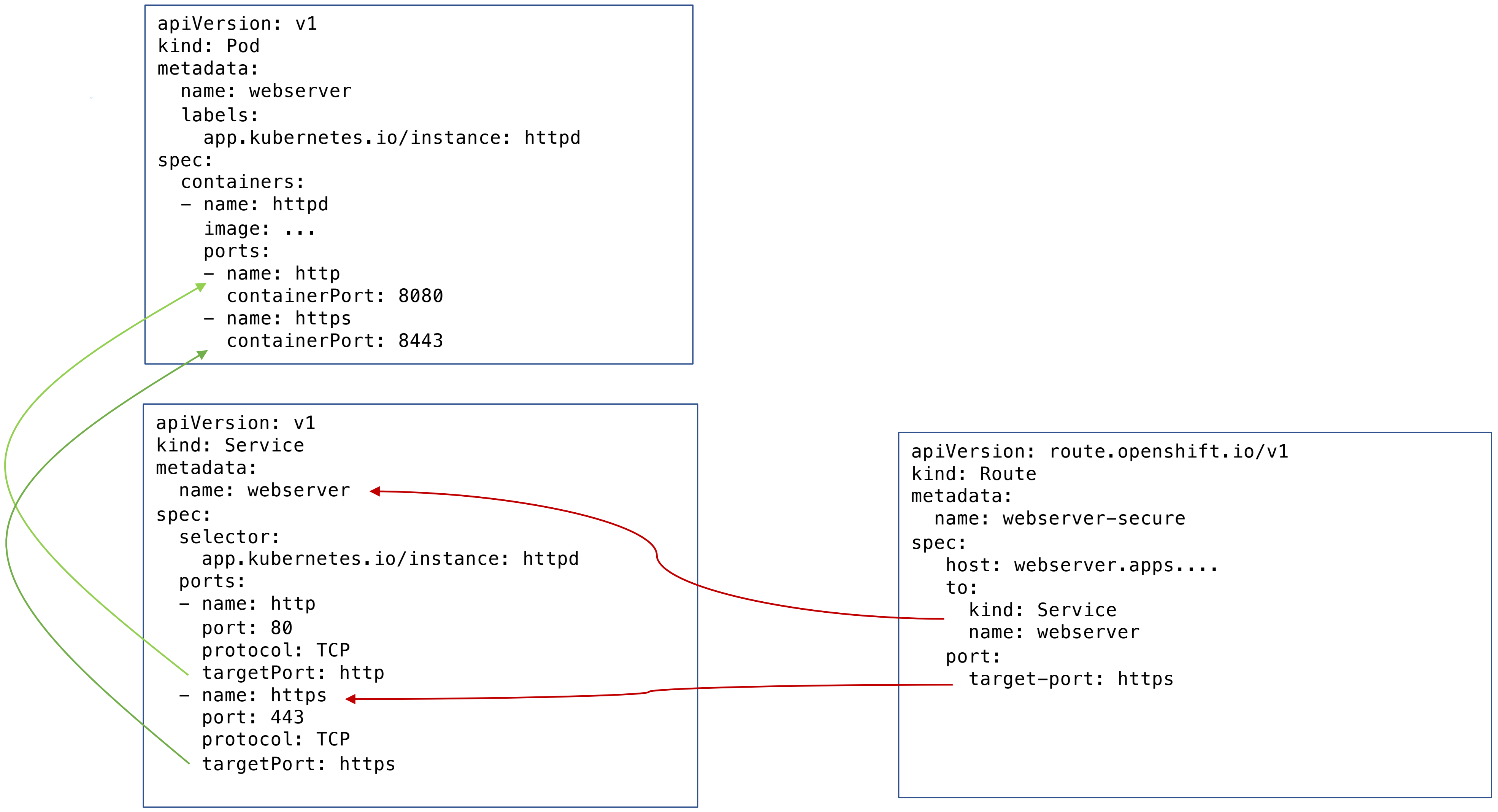
StatefulSets → Headless Service

```
apiVersion: v1
kind: Service
metadata:
  name: webserver
spec:
  type: ClusterIP
  selector:
    app.kubernetes.io/instance: webserver
  ports:
  - name: http
    port: 80
    protocol: TCP
    targetPort: http
```



service | pod sdn

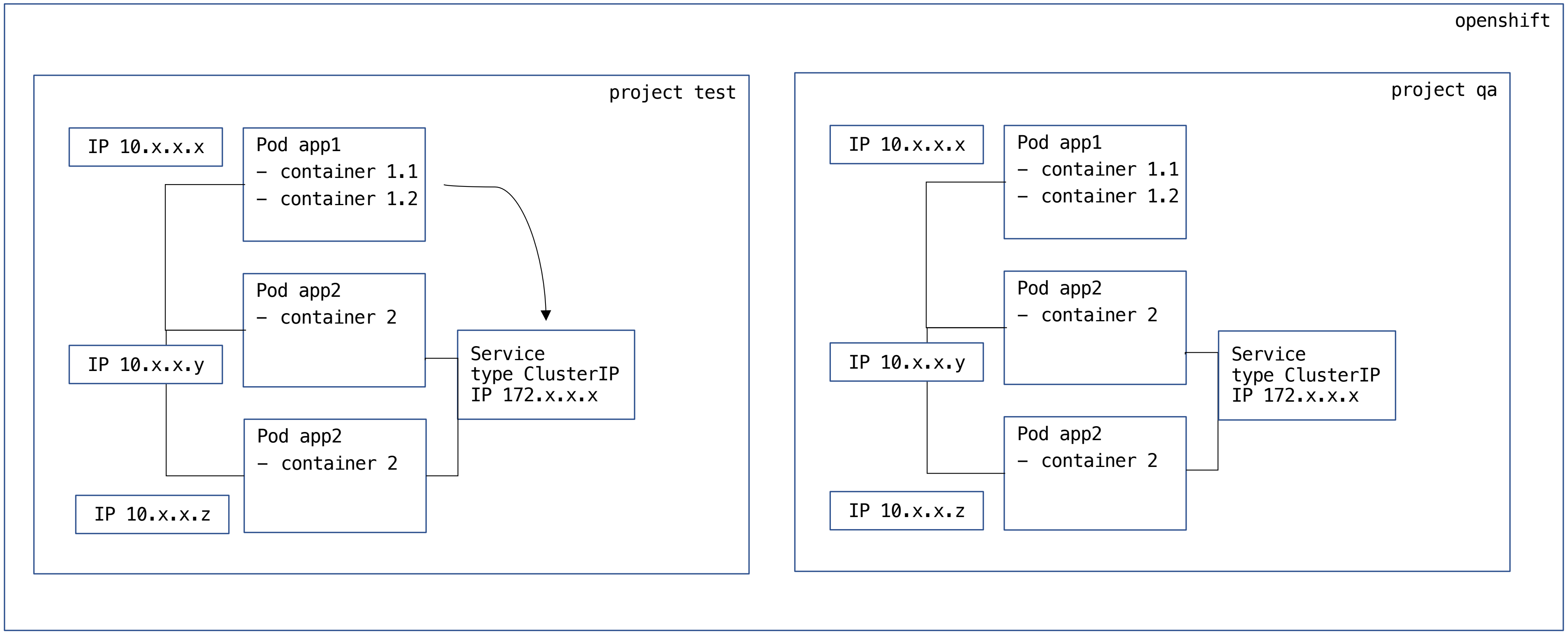
Pod | Service | Route



```yaml
apiVersion: v1
kind: Pod
metadata:
  name: webserver
  labels:
    app.kubernetes.io/instance: httpd
spec:
  containers:
  - name: httpd
    image: ...
    ports:
    - name: http
      containerPort: 8080
    - name: https
      containerPort: 8443
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: webserver
spec:
  selector:
    app.kubernetes.io/instance: httpd
  ports:
  - name: http
    port: 80
    protocol: TCP
    targetPort: http
  - name: https
    port: 443
    protocol: TCP
    targetPort: https
```

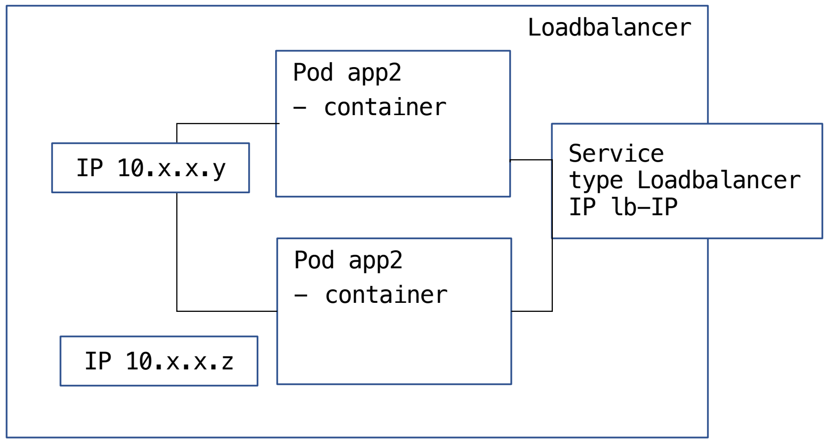
```yaml
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  name: webserver-secure
spec:
   host: webserver.apps....
   to:
     kind: Service
     name: webserver
   port:
     target-port: https
```

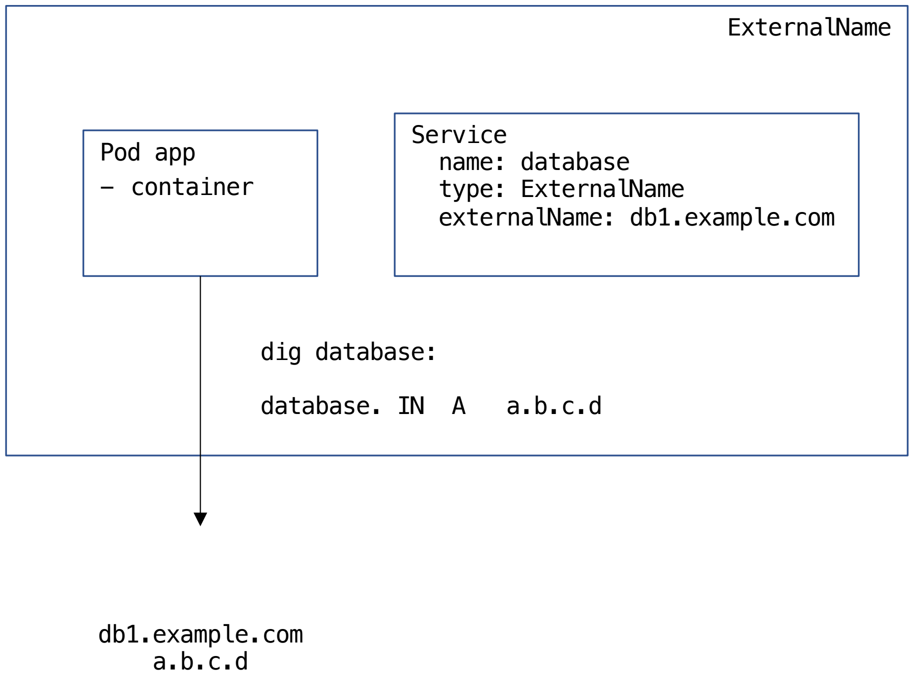


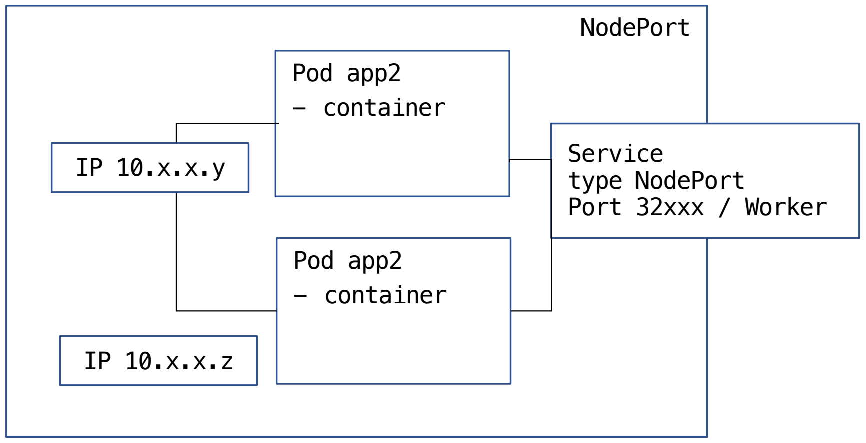
```
DNS:
A: <service>.test.svc.cluster.local
SVC: _443._tcp.https.<service>.test.svc.cluster.local

/etc/resolv.conf:
search test.svc.cluster.local svc.cluster.local ...
ndots 5
```

```
DNS:
A: <service>.qa.svc.cluster.local
SVC: _443._tcp.https.<service>.qa.svc.cluster.local

/etc/resolv.conf:
search qa.svc.cluster.local svc.cluster.local ...
```

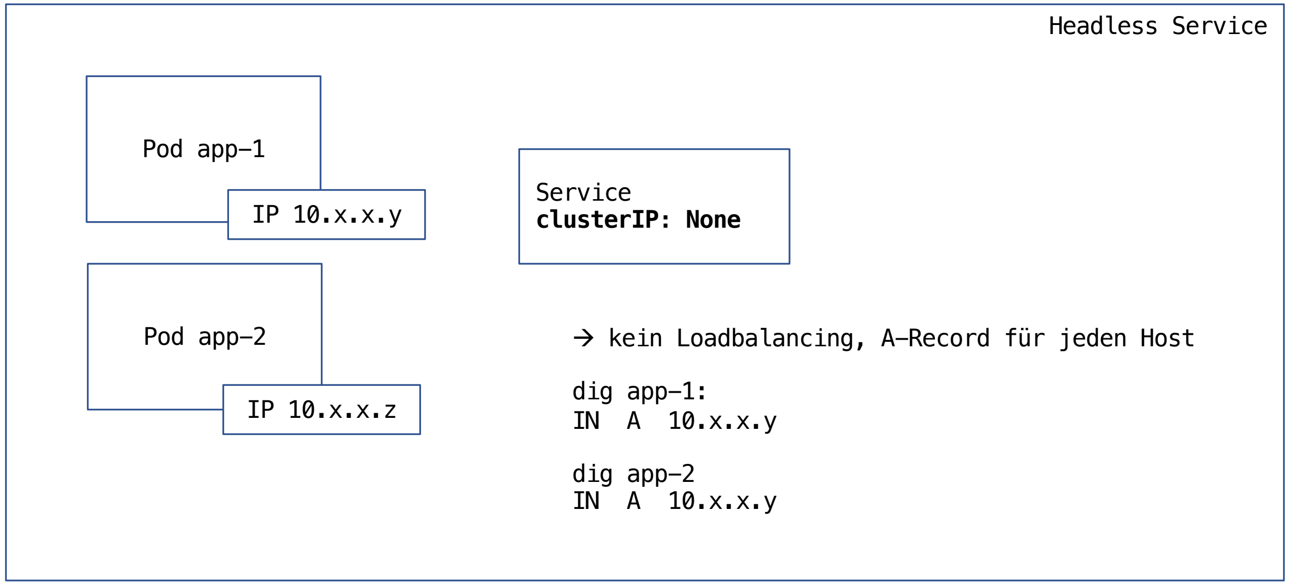
→ einfacher DNS-Lookup nach <service> in jedem Projekt


Loadbalancer
Pod app2
– container
IP 10.x.x.y
Service
type Loadbalancer
IP lb-IP
Pod app2
– container
IP 10.x.x.z


ExternalName
Pod app
– container
Service
name: database
type: ExternalName
externalName: db1.example.com
dig database:
database. IN A a.b.c.d
db1.example.com
a.b.c.d


NodePort
Pod app2
– container
IP 10.x.x.y
Service
type NodePort
Port 32xxx / Worker
Pod app2
– container
IP 10.x.x.z

```
kind: StatefulSet
metadata:
  name: app
spec:
  ...
```



→ kein Loadbalancing, A-Record für jeden Host

```
dig app-1:
IN  A  10.x.x.y

dig app-2
IN  A  10.x.x.y
```

Services für externe Datenbank:

```
NAME           TYPE          CLUSTER-IP      EXTERNAL-IP       PORT(S)
database       ExternalName  <none>          dsbox.de          <none>
```

```
apiVersion: v1
kind: Service
metadata:
  name: database
  namespace: sample
spec:
  type: ExternalName
  externalName: dsbox.de
```

```
# oc exec rest-sample-59fc6bf5b6-9dchd -- sh -c 'psql postgresql://daniel:12345678@database/postgres -c "\conninfo"'
You are connected to database "postgres" as user "daniel" on host "database-ext" at port "5432".
```

```
DNS-Gotcha:
# oc exec rest-sample-59fc6bf5b6-9dchd -- sh -c 'nslookup dsbox.de'
Name:     dsbox.de.<wildcard.domain>
Address: 5.9.70.75

# oc exec rest-sample-59fc6bf5b6-9dchd -- sh -c 'nslookup dsbox.de.'
Name:     dsbox.de
Address: 176.9.155.194
```

https://lmgtfy.app/?q=options+ndots%3A5

database servvice

Services für Database-Pod:

```
NAME            TYPE        CLUSTER-IP       EXTERNAL-IP           PORT(S)
database        ClusterIP   172.30.156.203   <none>                5432/TCP
database-node   NodePort    172.30.160.12    <none>                5432:30001/TCP
database-ip     ClusterIP   172.30.31.229    10.0.0.21,10.0.0.22   5432/TCP
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: database
  namespace: sample
spec:
  selector:
    app.kubernetes.io/name: database
  type: ClusterIP
  ports:
  - name: database
    protocol: TCP
    port: 5432
    targetPort: 5432
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: database-node
  namespace: sample
spec:
  selector:
    app.kubernetes.io/name: database
  type: NodePort
  ports:
  - name: database
    protocol: TCP
    port: 5432
    targetPort: 5432
    nodePort: 30001  ← Range 30000-32000
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: database-ip
  namespace: sample
spec:
  selector:
    app.kubernetes.io/name: database
  ports:
  - name: database
    protocol: TCP
    port: 5432
    targetPort: 5432
  externalIPs:
  - 10.0.0.21
  - 10.0.0.22
```

```
# oc exec rest-sample-59fc6bf5b6-9dchd -- sh -c 'psql postgresql://daniel:12345678@database/postgres -c "\conninfo"'
You are connected to database "postgres" as user "daniel" on host "database" at port "5432".

# psql postgresql://daniel:12345678@worker01:30001/postgres -c "\conninfo"
You are connected to database "postgres" as user "daniel" on host "worker01" at port "30001"

# psql postgresql://daniel:12345678@10.0.0.21/postgres -c "\conninfo"
You are connected to database "postgres" as user "daniel" on host "10.0.0.21" at port "5432"

# psql postgresql://daniel:12345678@worker02/postgres -c "\conninfo"
You are connected to database "postgres" as user "daniel" on host "worker02" at port "5432"
```