- Openshift
  Orchestrierungsservice zur Bereitstellung, Verwaltung und Skalierung von Container-Anwendungen

- Deklaratives System
  Status wird in Resourcen (YAML) definiert und durch Controller hergestellt
  IaC – Infrastructure as Code (https://blog.nelhage.com/post/declarative-configuration-management)

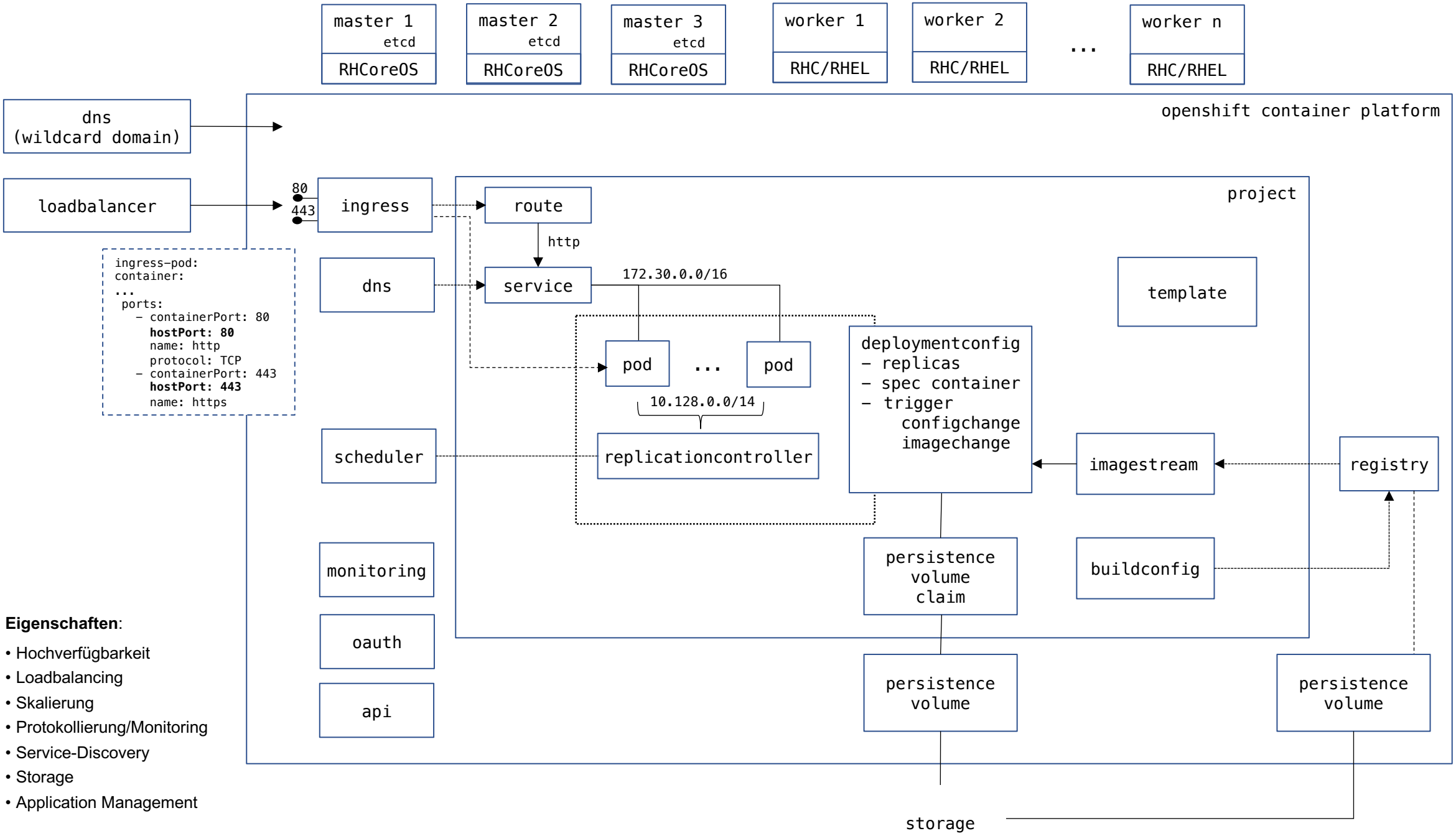
```
$ oc api-resources -o name --sort-by=name
alertmanagers.monitoring.coreos.com
apiservers.config.openshift.io
apiservices.apiregistration.k8s.io
appliedclusterresourcequotas.quota.openshift.io
authentications.config.openshift.io
authentications.operator.openshift.io
baremetalhosts.metal3.io
bindings
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)

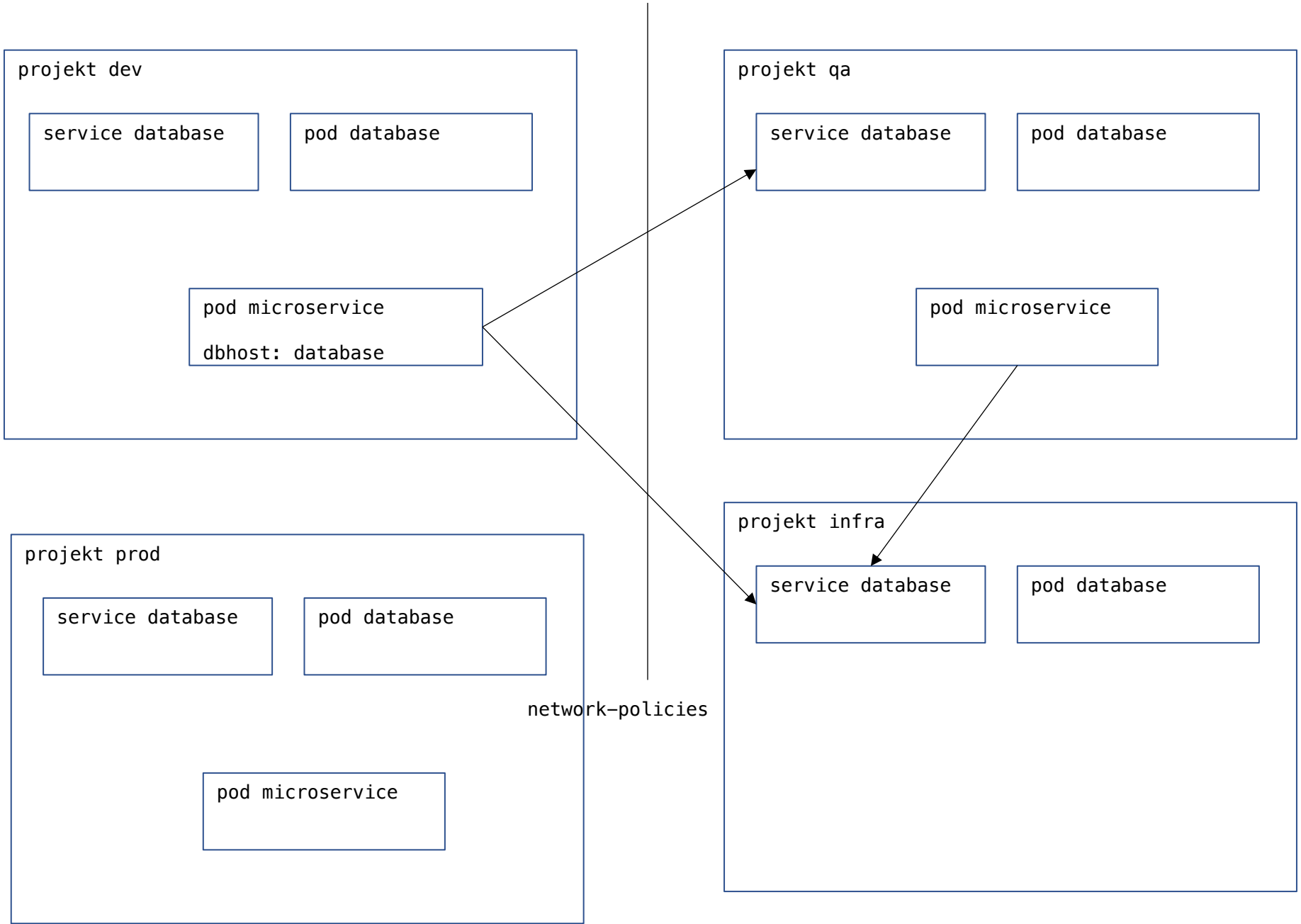
Node
PersistenceVolume (pv)

Operator
CustomResourceDefinition (crd)



Eigenschaften:

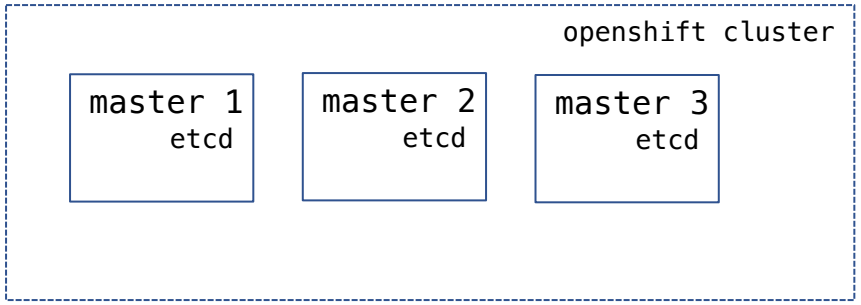
- Hochverfügbarkeit
- Loadbalancing
- Skalierung
- Protokollierung/Monitoring
- Service-Discovery
- Storage
- Application Management


projekt dev
service database
pod database
pod microservice
dbhost: database
projekt qa
service database
pod database
pod microservice
projekt infra
service database
pod database
projekt prod
service database
pod database
pod microservice
network-policies

Openshift Resources

```
apiVersion: v1
kind: < Resource Type >
metadata:
  name: <name>
  namespace: <namespace>
  annotations:
    ...
  labels:
    app: <application-name>
    ...
spec:
  ...
status:
  ...
```

oc create




```
apiVersion: v1
kind: Pod
metadata:
  namespace: danielstraub-do180
  labels:
    app: nginx
spec:
  containers:
  - image: quay.io/dstraub/nginx
    imagePullPolicy: Always
    ports:
    - containerPort: 8080
      protocol: TCP
  ...
```

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

```yaml
apiVersion: apps.openshift.io/v1
kind: DeploymentConfig
metadata:
  name: wildfly
  namespace: do295-sample
  labels:
    app: wildfly
spec:
  replicas: 1
  selector:
    app: wildfly
  template:
    metadata:
      labels:
        app: wildfly
    spec:
      containers:
      - name: wildfly
        image: quay.io/danielstraub/wildfly:latest
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 8080
          protocol: TCP
```

```yaml
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  name: wildfly
  labels:
    app: wildfly
spec:
  host: do295-sample.apps.na46.prod.nextcle.com
  to:
    kind: Service
    name: wildfly
  tls:
    termination: edge
```

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

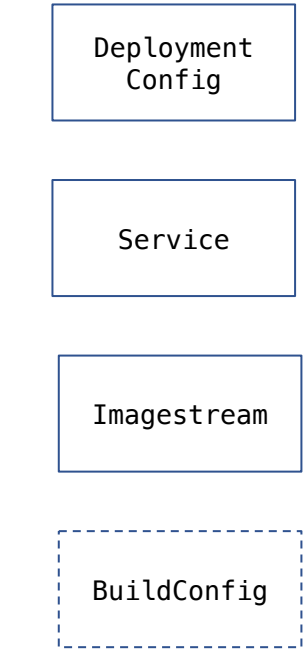
```
$ oc new-app quay.io/danielstraub/wildfly
--> Found container image 9a9e908 (9 days old) from quay.io for "quay.io/danielstraub/wildfly"

    * 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
    Application is not exposed. You can expose services to the outside world by executing one or more of the commands below:
     'oc expose service/wildfly'
    Run 'oc status' to view your app.

$ oc get all
NAME                           READY   STATUS        RESTARTS   AGE
pod/wildfly-8584657848-t84x2   1/1     Terminating   0          5m25s
pod/wildfly-8b4ff4896-5hjkc    1/1     Running       0          20s

NAME              TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)             AGE
service/wildfly   ClusterIP   172.30.166.219   <none>        8080/TCP,8443/TCP   21s

NAME                      READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/wildfly   1/1     1            1           21s

NAME                                 DESIRED   CURRENT   READY   AGE
replicaset.apps/wildfly-8b4ff4896    1         1         1       20s

NAME                                     IMAGE REPOSITORY
TAGS     UPDATED
imagestream.image.openshift.io/wildfly   ...   latest   20 seconds ago
```

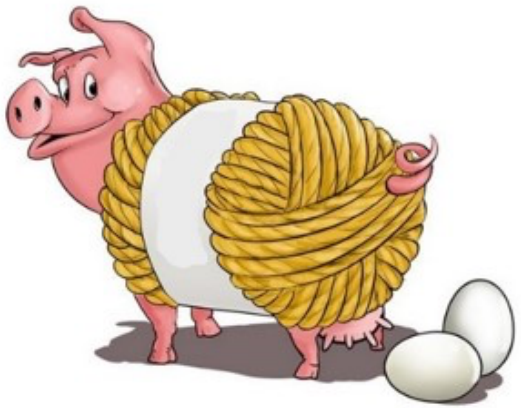
```
$ oc new-app --help
Create a new application by specifying source code, templates, and/or images

...

Usage:
  oc new-app (IMAGE | IMAGESTREAM | TEMPLATE | PATH | URL ...) [flags]


Beispiele:

$ oc new-app https://quay.io/dstraub/nginx --name ngnix

$ oc new-app php:7.3~https://github.com/.../php-hello
```

Container-Image

Builder-Image (s2i)

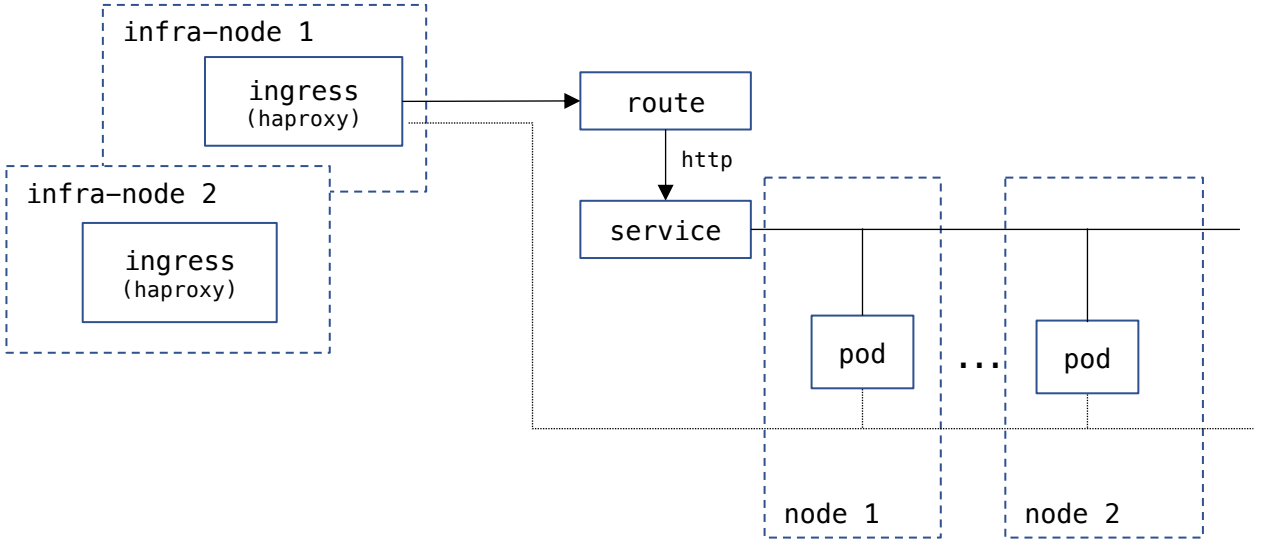
Git-Projekt (Source)





- oc login –u <user> -p <password> <api-server-url>
- oc new-project <name>
- oc create/apply –f <resource-yml>
- oc status
- oc get <resource-type> [<resource-name>]
  - oc get pods
  - oc get dc <deploymentconfig>
  - oc get svc <service>
  - oc get events
- oc describe <resource-name>
- oc expose svc <service-name>
- oc logs <podname>
- oc exec –it <podname> -- <program>
- oc port-forward <podname> <local-port>:<remote-port>

- oc new-app <☺anything☺>
- oc delete <resource-type> <resource-name>
- oc rollout latest <deployment-config>



172.30.0.0/16 Service-SDN
IP bleibt unverändert solange Service existiert
DNS: A-Record <service>.<project>.svc.cluster.local

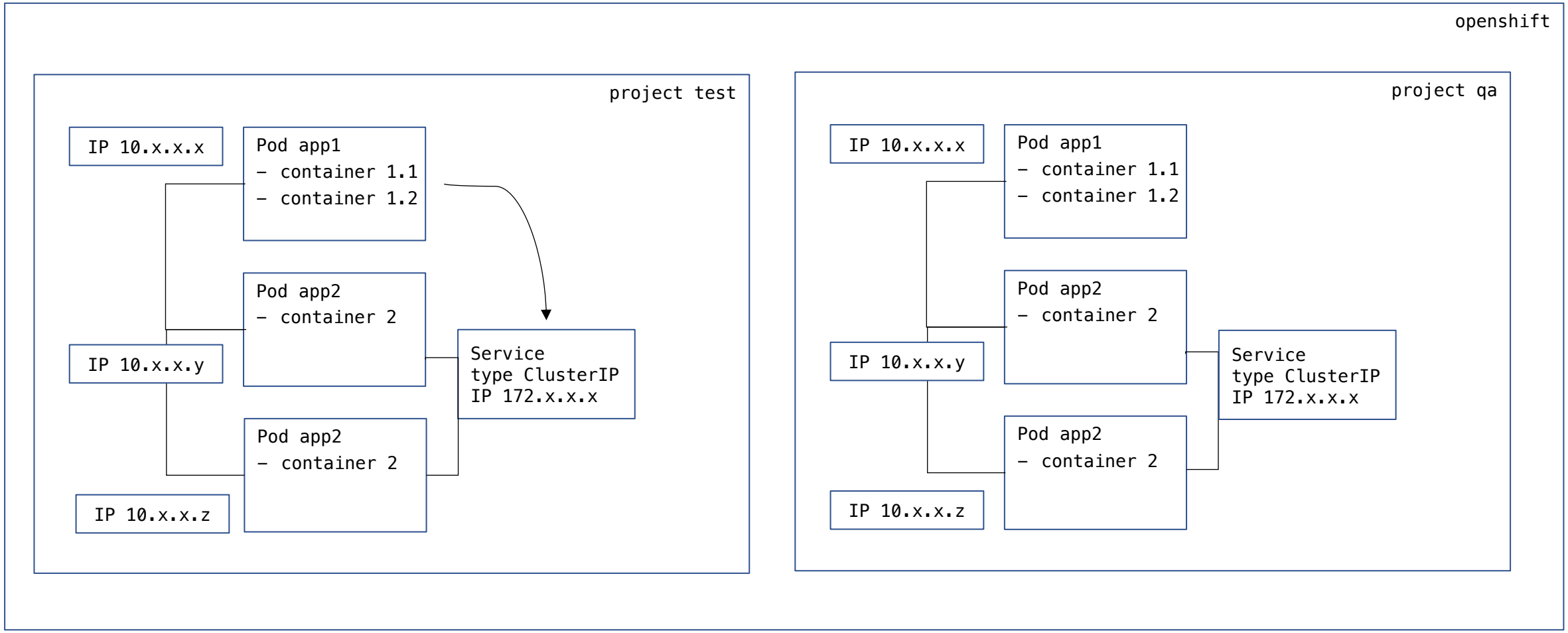
10.128.0.0/14 Pod-SDN
→ Jeder Pod erhält neue IP

```
$ oc expose service <service>
```

Route: <service>-<project>.<default-domain> ← Wildcard-Domain im DNS

```
$ oc expose service <service> --hostname=<domain>
```

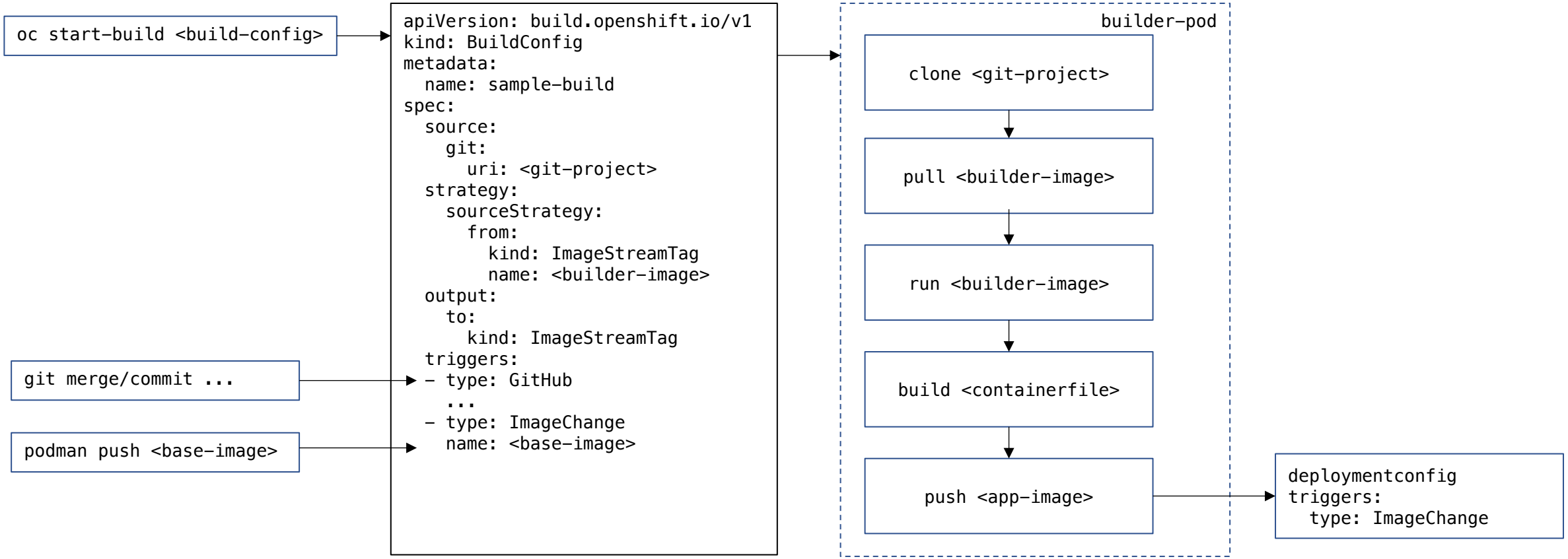
console-openshift-console.apps.eu46.prod.nextcle.com




```
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 ...
```

```
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



Source2Image: Builder-Image enthält Tools und Logik zum Bauen (Compilieren) einer Anwendung

Verwenden einer externen Registry: https://cloud.redhat.com/blog/pushing-application-images-to-an-external-registry

Container in Openshift:

- beliebige User-Id
- Group-Id 0 (root)
- 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

# 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
```

NFS-Mount →

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

Abweichende User-Id erfordert Serviceaccount mit Securit Context Constraint 'anyuid' :

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

```yaml
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
```

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

erstellt von Cluster-Administrator !

```yaml
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 Namespaces/Project 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 8080 default_server;
      server_name _;
      location / {
         root   /usr/share/nginx/html;
         index  index.html index.htm;
      }
    }
```

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
```

Container Registry:

```
# 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
```

Verwenden einer externen Container Registry in Openshift: Secret mit auth.json

```
apiVersion: v1
kind: Secret
metadata:
  name: pull-secret-td4k2kdt2m
  namespace: apps
type: kubernetes.io/dockerconfigjson
data:
  .dockerconfigjson: ewogICJhdXRocyI6IHsKICAgICJyZWdpc3 ...
```

Serviceaccount 'imagePullSecrets' :

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: apps
  namespace: apps
imagePullSecrets:
- name: apps-dockercfg-4sdrk
- name: pull-secret-td4k2kdt2m
...
```

oder im Deployment verwenden:

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

Imagestream:

- enthält Verweise (Zeiger) auf Images und deren Tags (keine Images !)
- automatische Aktualisierung möglich (15 Min. bei externen Registries)
- Verwendung in DeploymentConfig als Image und Trigger

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

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

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

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

1.0
  updates automatically from registry quay.io/
  * quay.io/dstraub/nginx@sha256:693b30b107da
```

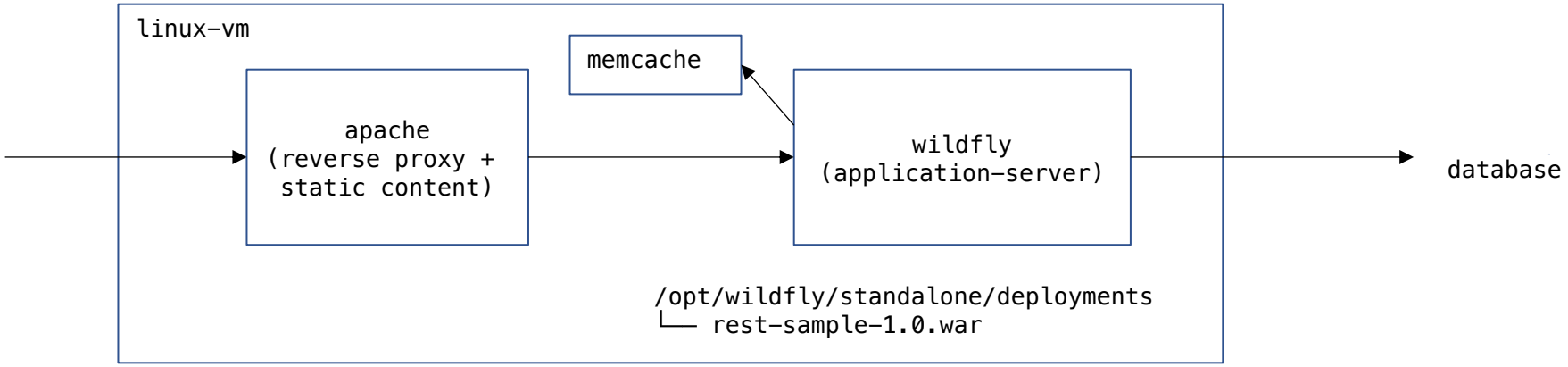
TAG	LAST MODIFIED ↓	SECURITY SCAN	SIZE	MANIFEST
1.2	40 minutes ago	8 Medium	91.9 MB	SHA256 ee508edacfe0
latest	14 hours ago	8 Medium	91.9 MB	SHA256 c34f57431167
1.1	a day ago	8 Medium	90.6 MB	SHA256 674ab485f6e8
1.0	a day ago	8 Medium	90.6 MB	SHA256 693b30b107da

Imagestream:

```
$ oc describe is nginx
Name:                           nginx
Unique Images:                  4
Tags:                           4

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

$ oc describe dc nginx
Name:      nginx
...
Triggers:  Config, Image(nginx@latest, auto=true)
Pod Template:
  Labels:  app=nginx
  Containers:
   nginx:
    Image: quay.io/dstraub/nginx@sha256:c34f57431167fca4707
```

```
deploymentconfig
template:
  spec:
    containers:
    - image: nginx
      imagePullPolicy: Always
      name: nginx
  ...
  triggers:
  - type: ImageChange
    imageChangeParams:
    automatic: true
    containerNames:
    - nginx
    from:
      name: nginx:latest
```

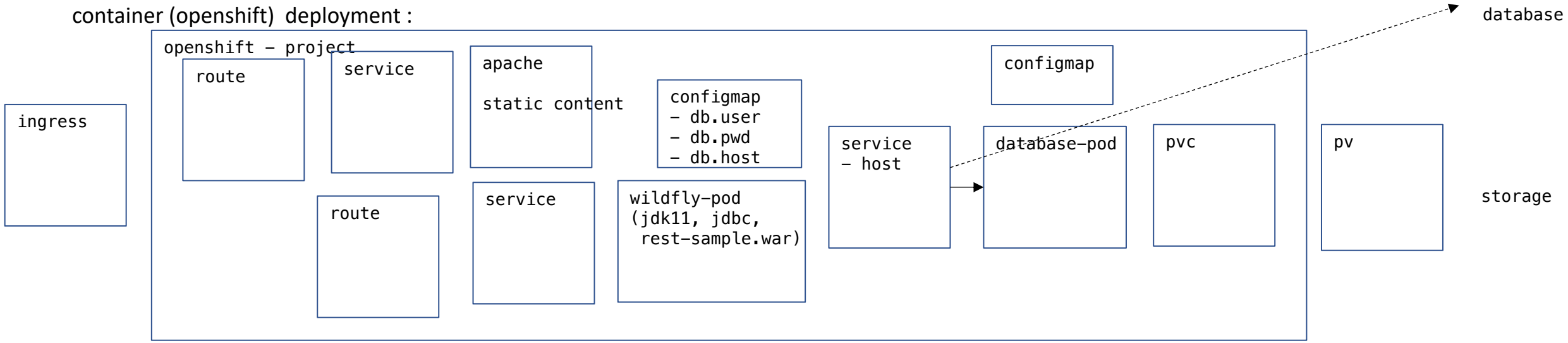
Anwendung mit mehreren Services
legacy deployment :



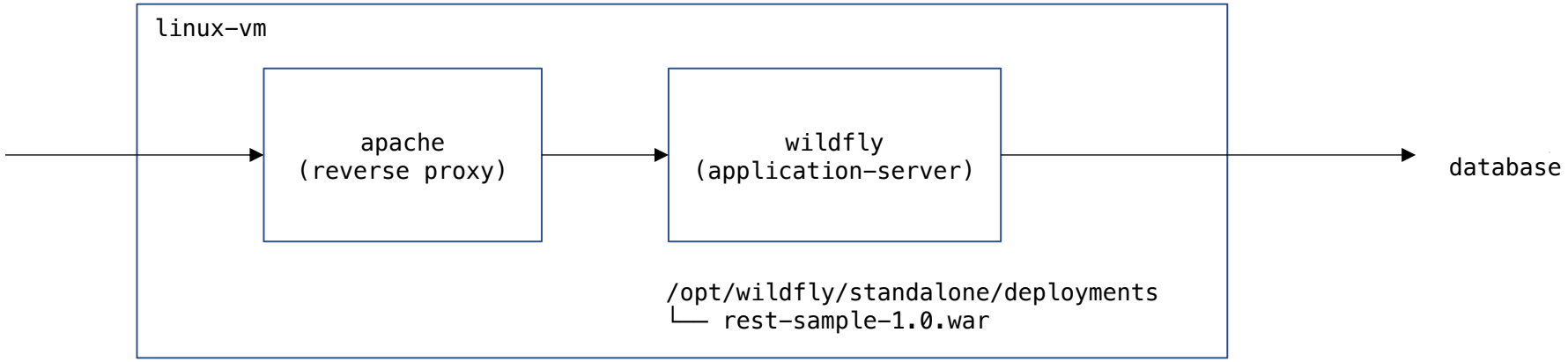
Neue Anwendungs-Version: Austausch des Artifakts, Auto-Deploy/Restart

---

container (openshift) deployment :



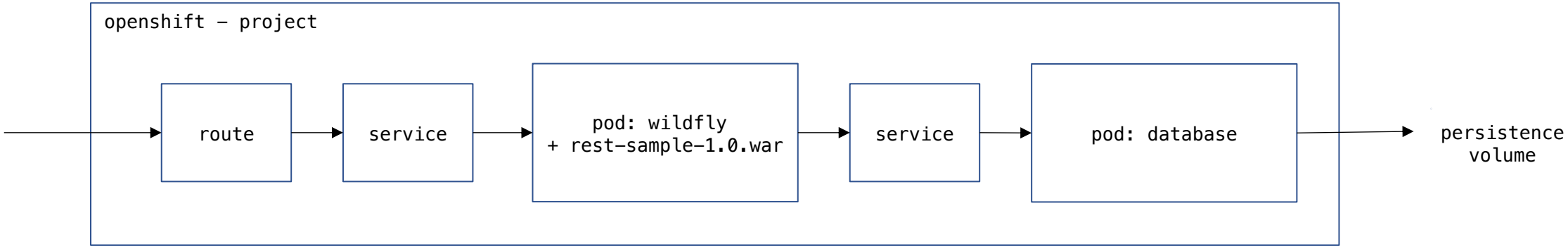
Anwendung mit mehreren Services
legacy deployment :



Neue Anwendungs-Version: Austausch des Artifakts, Auto-Deploy/Restart

---

container (openshift) deployment :



Neue Anwendungs-Version: neuer Container mit Artifakt, Rollout

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"
```

Services für externe Database:

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

```yaml
apiVersion: v1
kind: Service
metadata:
  name: database
  namespace: sample
spec:
  selector:
    app.kubernetes.io/name: database
  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.straubcloud.de
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

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 -
```

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

Chart.yml

```
apiVersion: v1
name: sample
description: Sample Application
version: 1.0
appVersion: 1.0
dependencies: ...
```

values.yml

```
image:
  repository: quay.io/redhat.io/sample
  tag: '1'

service:
  port: 8080

env:
```

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 dependency update
helm install
```

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

kustomize.yml

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

namespace: sample

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

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
kind: Deployment
metadata:
  name: rest-sample
spec:
  replicas: 1
  template:
    spec:
      containers:
      - name: sample
        image: sample
```

```
# kustomize build .
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 .
```

Kustomize Overlays : erzeugen unterschiedlicher Variante von einer Basis-Vorlage

base/kustomize.yml

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

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

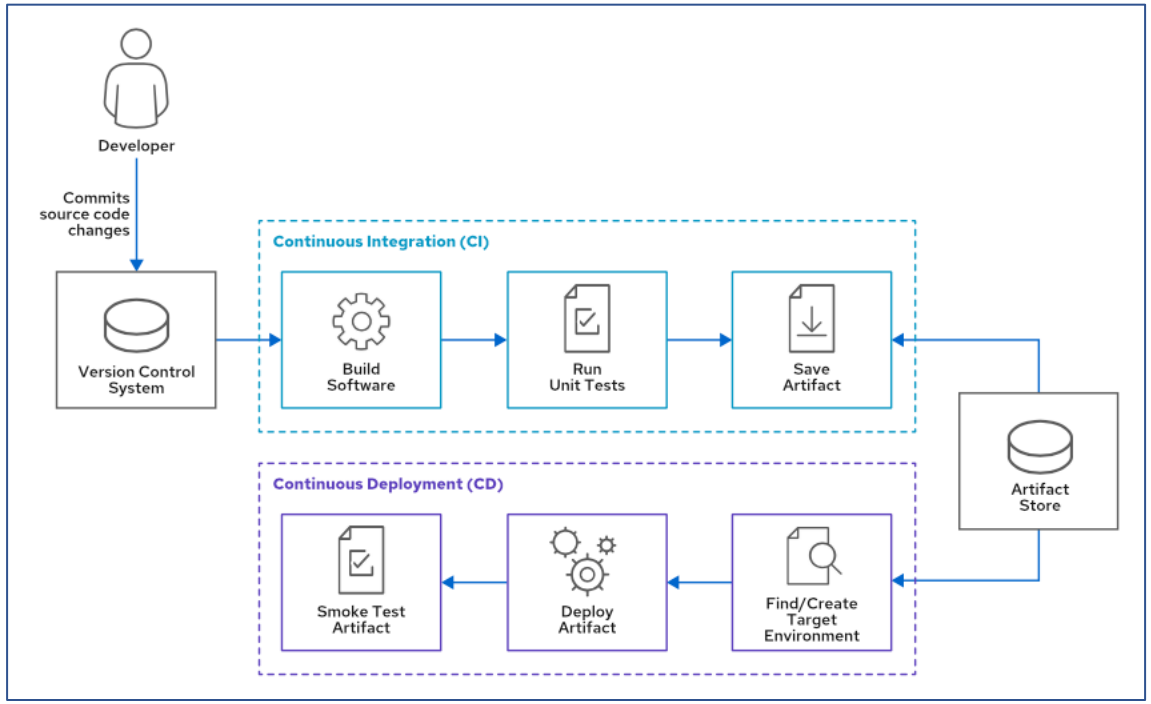
overlays/test/kustomize.yml

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

resources:
- ../../base

namespace: test

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

overlays/production/kustomize.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
...
```



Continuos Integration
Continuos Delivery

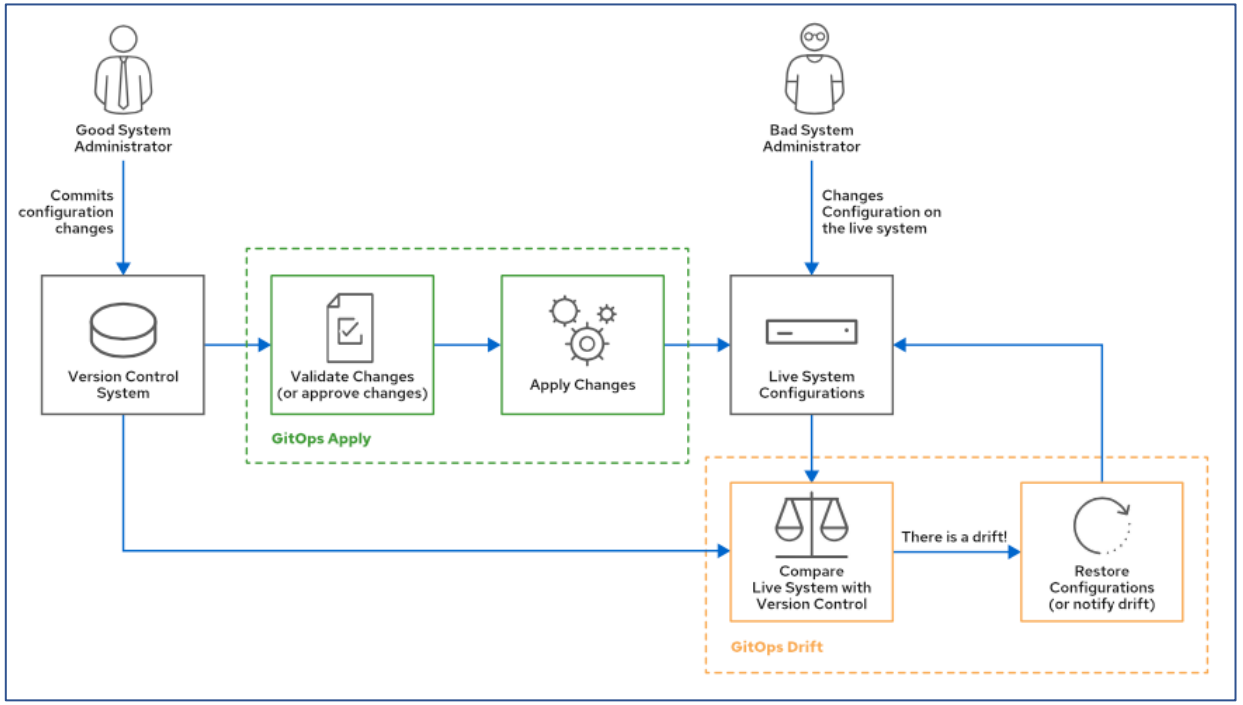
→ Developer
→ running application

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

GitOps Workflow

→ Administrators
→ live System

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



Imperatives Resource Management: oc new-app, oc set, oc patch, oc adm, oc edit

Declaratives Resource Management: oc apply, k8s (Manifeste unter VCS-Verwaltung)

→ Empfehlung: Manifests von Scratch erzeugen [apiVersion, kind, metadata, spec]

```
# oc api-resources
NAME                              SHORTNAMES       APIVERSION      NAMESPACED   KIND
bindings                                           v1              true         Binding
componentstatuses                 cs               v1              false        ComponentStatus
configmaps                        cm               v1              true         ConfigMap
endpoints                         ep               v1              true         Endpoints
events                            ev               v1              true         Event
limitranges                       limits           v1              true         LimitRang

# oc api-versions
admissionregistration.k8s.io/v1
admissionregistration.k8s.io/v1beta1
apiextensions.k8s.io/v1
apiextensions.k8s.io/v1beta1

# oc explain --recursive Service
KIND:     Service
VERSION:  v1

DESCRIPTION:
     Service is a named abstraction of software service (for example, mysql)
FIELDS:
   apiVersion        <string>
   kind    <string>
   metadata          <Object>
   ...
```

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:
Abgleich Ist-Zustand (Cluster) mit Kustomize/Helm-Definitionen im VCS
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	

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 `.spec.containers.startupProbe`
  Container wird bei neg. Startup-Probe sofort beendet

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)

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>
```

```
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:
```

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

Deployment-Strategien

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

DeploymentConfig:

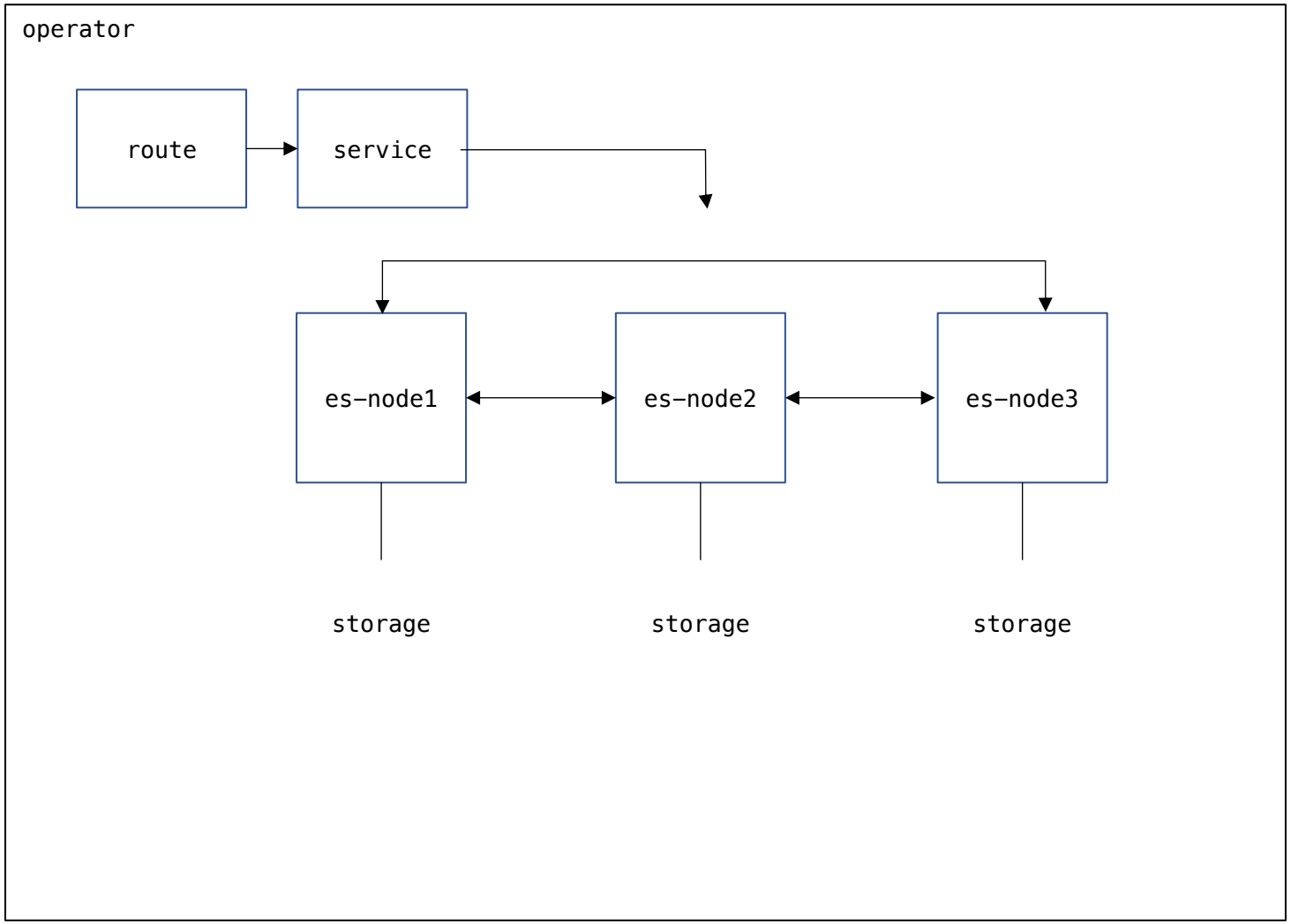
- Pre/Mid/Post – Lifecycle Hooks

Beenden eines Pods:

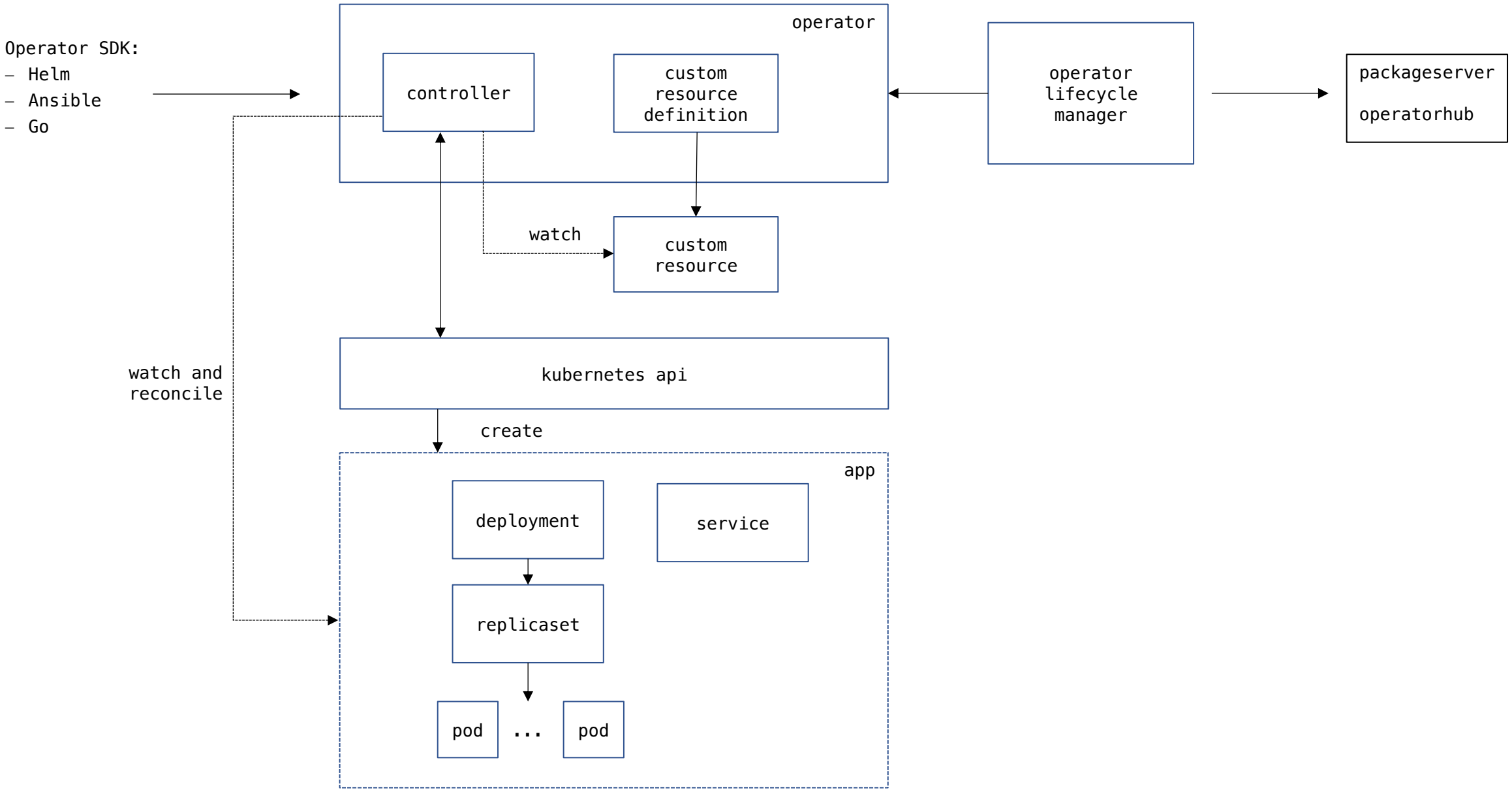
- SIGTERM : Pod soll keine neuen Verbindungen annehmen und bestehenden Aktionen beenden
- SIGKILL: nach `terminationGracePeriodSeconds` (30s) wird der Pod beendet

```
kind: Deployment
metadata:
  name: ...
spec:
  revisionHistoryLimit: 3   (default: 10)
  replicas: 4
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1             ← max. 5 Pods
      maxUnavailable: 0
  ...
  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=]
```




```
crd (custom resource definiton)

→ cr custom resource
elastic:
  nodes: 3
  namespace: ccc
  absc:
  ....
```


Operator SDK:
– Helm
– Ansible
– Go
operator
controller
custom resource definition
watch
custom resource
operator lifecycle manager
packageserver
operatorhub
watch and reconcile
kubernetes api
create
app
deployment
service
replicaset
pod
...
pod

Erzeugen eines Operator:

```
# oc get packagemanifests
NAME                                    CATALOG             AGE
metering-ocp                            Red Hat Operators   39s
...

# oc describe packagemanifests metering-ocp

# oc create -f <namespace.yml>

# oc create -f <operatorgroup.yml>  (erlaubt einen Operator aus einem NS Resourcen in anderen NS anzulegen)

# oc create -f <subscription.yml>   (OLM erzeugt Operator)
```

Volumes

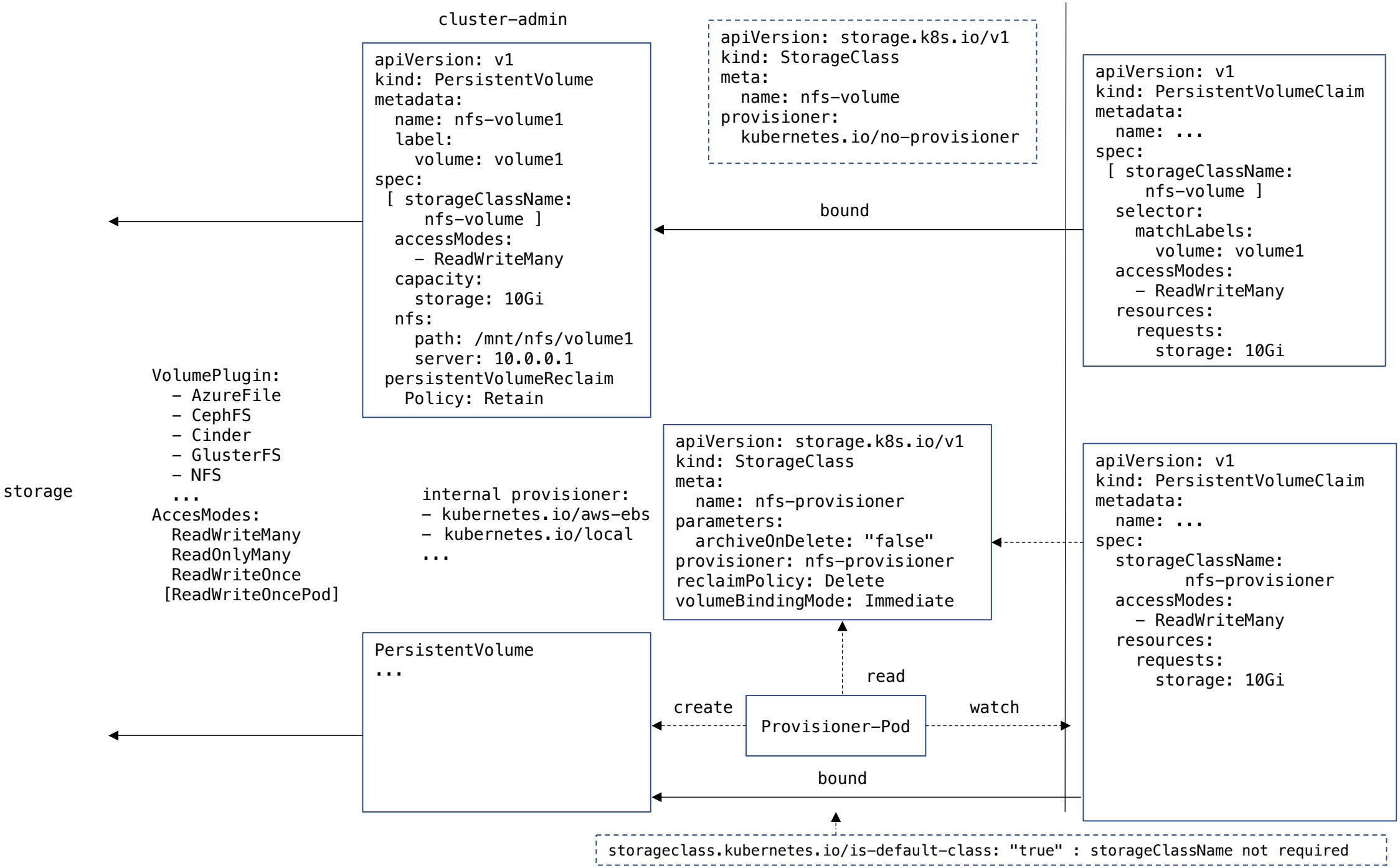
```
node

/
├── bin
├── etc
├── lib          mount
├── mnt      ──────────►  container
├── opt                   /
├── proc                  ├── bin
├── usr                   ├── etc
└── var                   ├── lib
                          ├── mnt
                          ├── opt
                          ├── proc
                          ├── usr
                          └── var

podman –v <local>:<container>
```

```
kind: Pod
...
spec:
  containers:
     ...
     volumeMounts:
       – mountPath: <path_container_fs>
         name: <name>
      ...
  volumes:
     – name: <volume>
       <volume–type>:
          <volume–attributes>
```

→ https://kubernetes.io/docs/concepts/storage/volumes/

Volume-Types
- emptyDir
- hostPath (`system:openshift:scc:hostmount–anyuid !`)
- configMap
- secret
- peristentVolumeClaim

...


cluster-admin
apiVersion: v1
kind: PersistentVolume
metadata:
name: nfs-volume1
label:
volume: volume1
spec:
[storageClassName:
nfs-volume]
accessModes:
- ReadWriteMany
capacity:
storage: 10Gi
nfs:
path: /mnt/nfs/volume1
server: 10.0.0.1
persistentVolumeReclaim
Policy: Retain
apiVersion: storage.k8s.io/v1
kind: StorageClass
meta:
name: nfs-volume
provisioner:
kubernetes.io/no-provisioner
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
name: ...
spec:
[storageClassName:
nfs-volume]
selector:
matchLabels:
volume: volume1
accessModes:
- ReadWriteMany
resources:
requests:
storage: 10Gi
bound
VolumePlugin:
- AzureFile
- CephFS
- Cinder
- GlusterFS
- NFS
...
AccesModes:
ReadWriteMany
ReadOnlyMany
ReadWriteOnce
[ReadWriteOncePod]
storage
internal provisioner:
- kubernetes.io/aws-ebs
- kubernetes.io/local
...
apiVersion: storage.k8s.io/v1
kind: StorageClass
meta:
name: nfs-provisioner
parameters:
archiveOnDelete: "false"
provisioner: nfs-provisioner
reclaimPolicy: Delete
volumeBindingMode: Immediate
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
name: ...
spec:
storageClassName:
nfs-provisioner
accessModes:
- ReadWriteMany
resources:
requests:
storage: 10Gi
PersistentVolume
...
read
create
Provisioner-Pod
watch
bound
storageclass.kubernetes.io/is-default-class: "true" : storageClassName not required

Blockdevice für Datenbanken

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: block-volume
spec:
  volumeMode: Block
  accessModes:
    - ReadWriteOnce
  storageClassName: fb-block
  capacity:
    storage: 10Gi
 fc:
   <fibre-channel>
```

block device

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: block-pvc
spec:
  storageClassName: fb-block
  volumeMode: Block
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
```

```
apiVersion: v1
kind: Pod
 ...
spec:
  containers:
   - name: ...
     volumeDevices:
       - name: data
         devicePath: /dev/xvda
     ...
  volumes:
    - name: data
      persistentVolumeClaim:
        claimName: block-pvc
```

LocalVolume Operator

```yaml
apiVersion: local.storage.openshift.io/v1
kind: LocalVolume
metadata:
  ...
spec:
  nodeSelector:
    matchLabels:
      node-role.kubernetes.io/infra: ""
  storageClassDevices:
  - storageClassName: local-volume
    volumeMode: Filesystem [ Block ]
    fsType: xfs
    devicePaths:
    - /dev/vdb
```

```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: local-volume
provisioner: kubernetes.io/no-provisioner
reclaimPolicy: Delete
volumeBindingMode: WaitForFirstConsumer
```

→ 1 PV für jeden Infra-Node

LocalVolume mit PersistentVolume

```yaml
apiVersion: v1
kind: PersistentVolume
spec:
  accessModes:
  - ReadWriteOnce
  capacity:
    storage: 20Gi
  persistentVolumeReclaimPolicy: Delete
  storageClassName: local-volume
  volumeMode: Filesystem
  local:
    fsType: xfs
    path: /mnt/local-volume/vdb
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
          - worker04
```