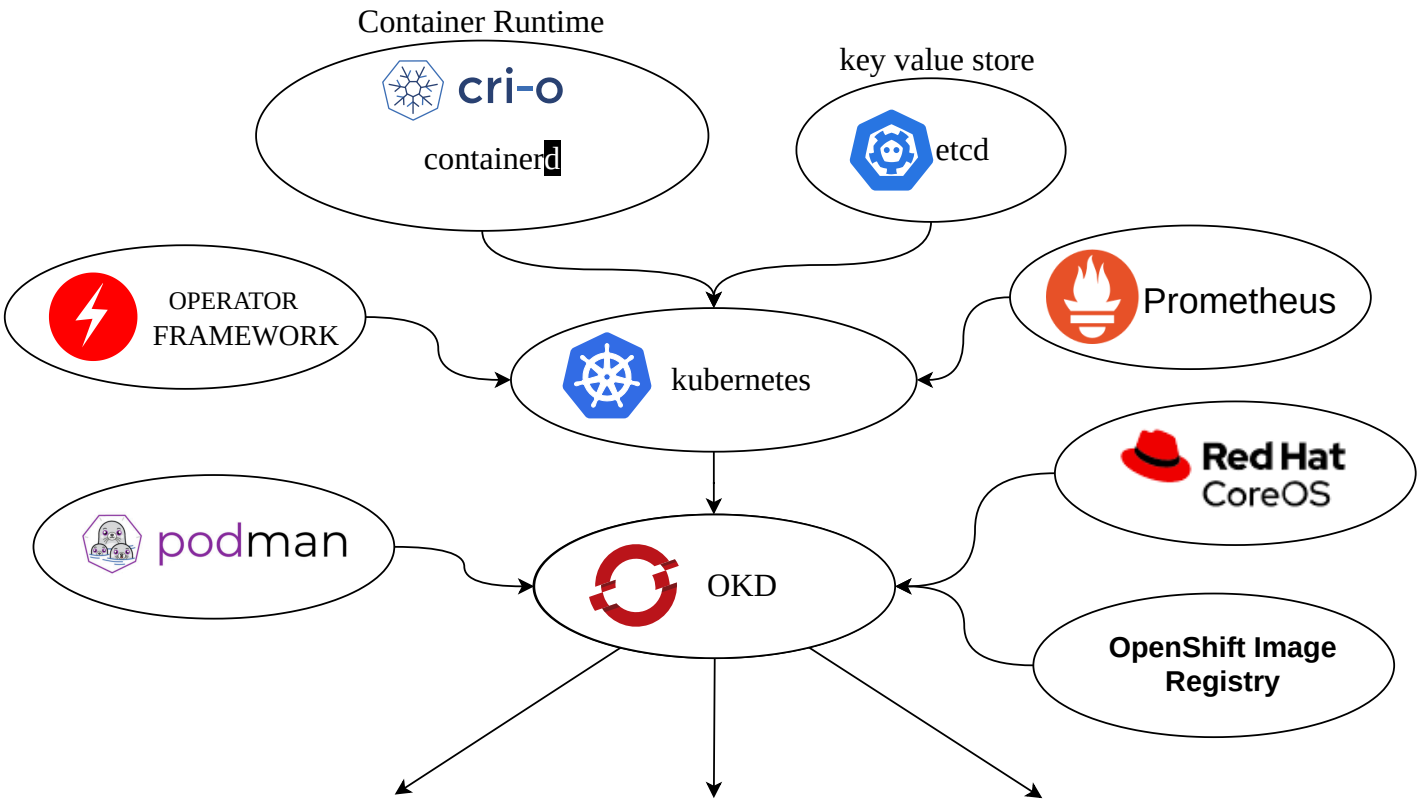


Red Hat OpenShift Container Platform

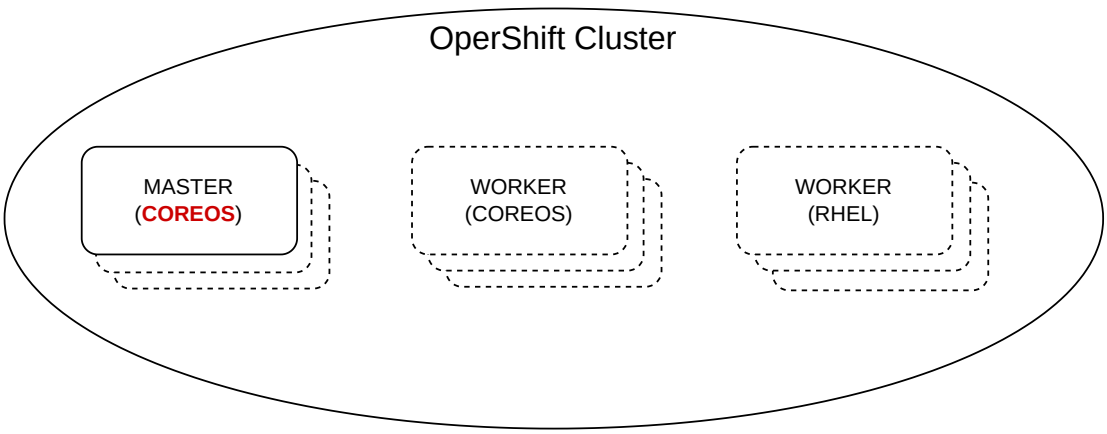
- Public/private DC.
- Bare metal and multiple cloud and virtualization providers.
- Full control by customer.

Red Hat OpenShift Dedicated

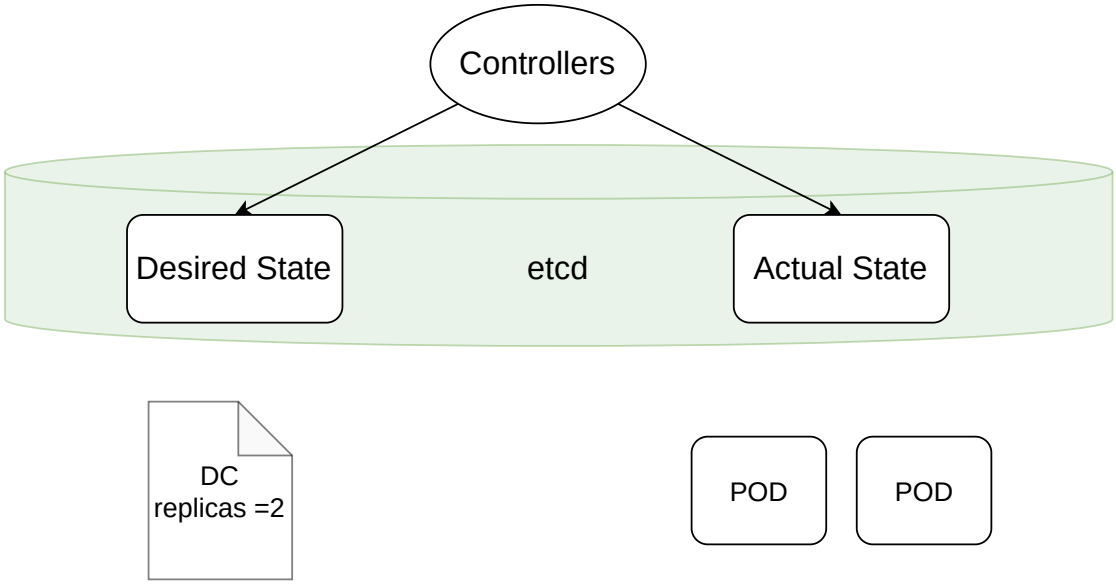
- Managed cluster in public cloud.
- RH manages the cluster.
- Customer manages updates and add-on services.

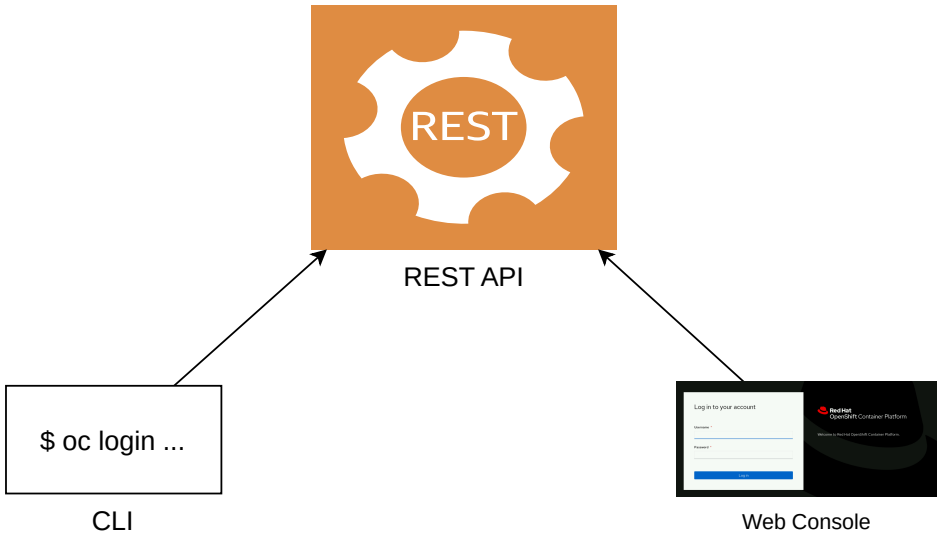
Red Hat OpenShift Online

- Public hosted cluster.
- Shared resources by multiple customers.
- RH manages cluster life cycle.

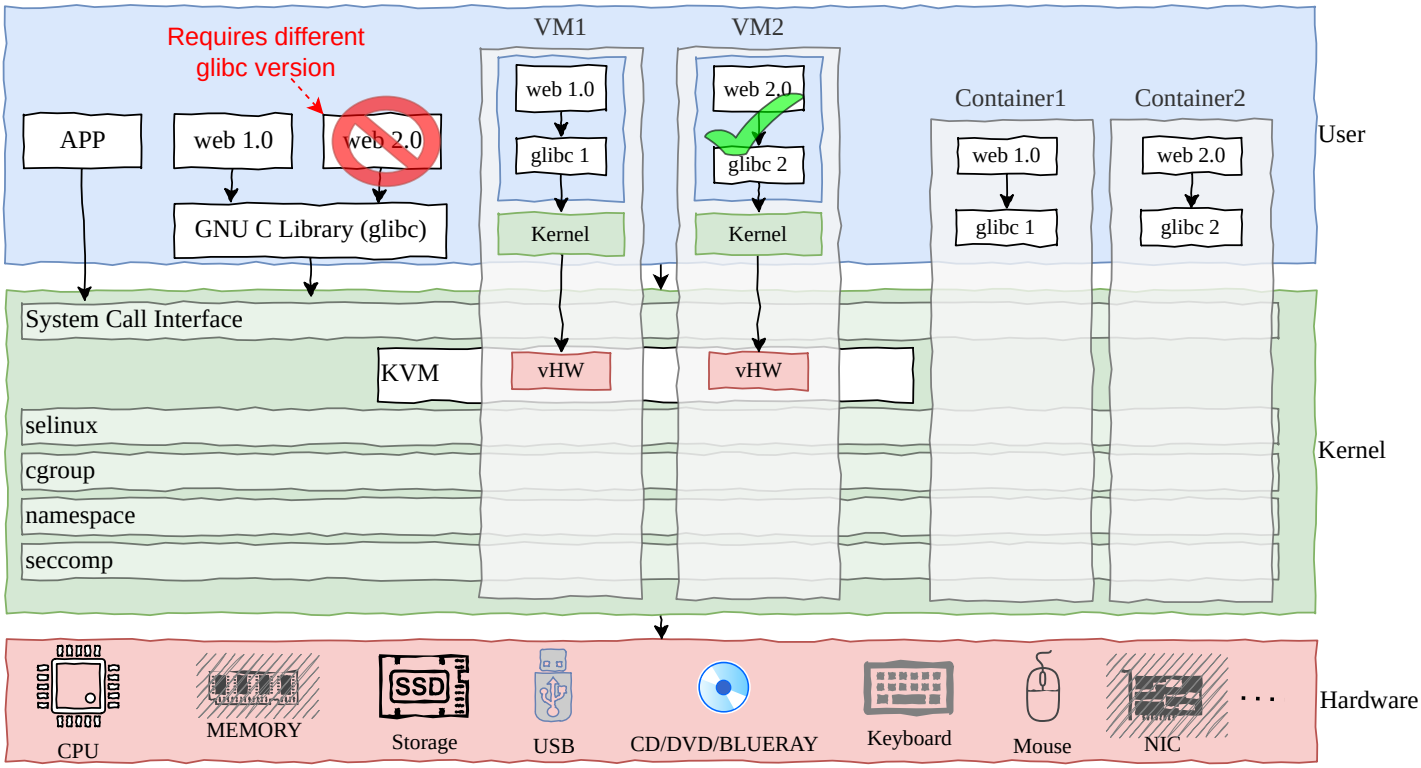


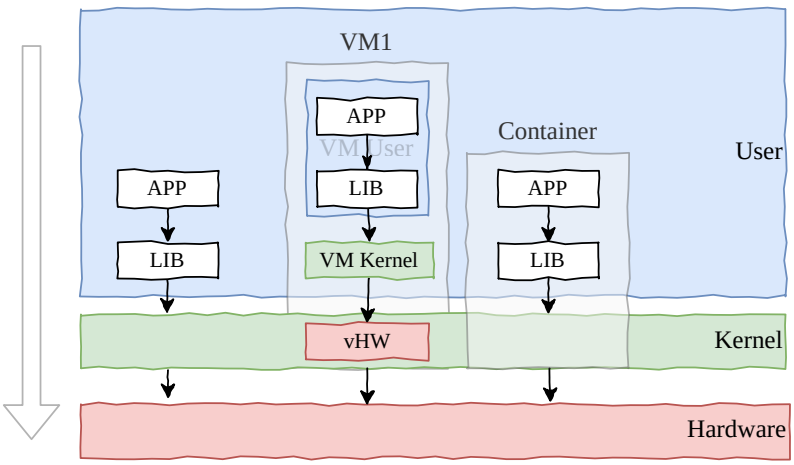
Kubernetes Declarative Architecture



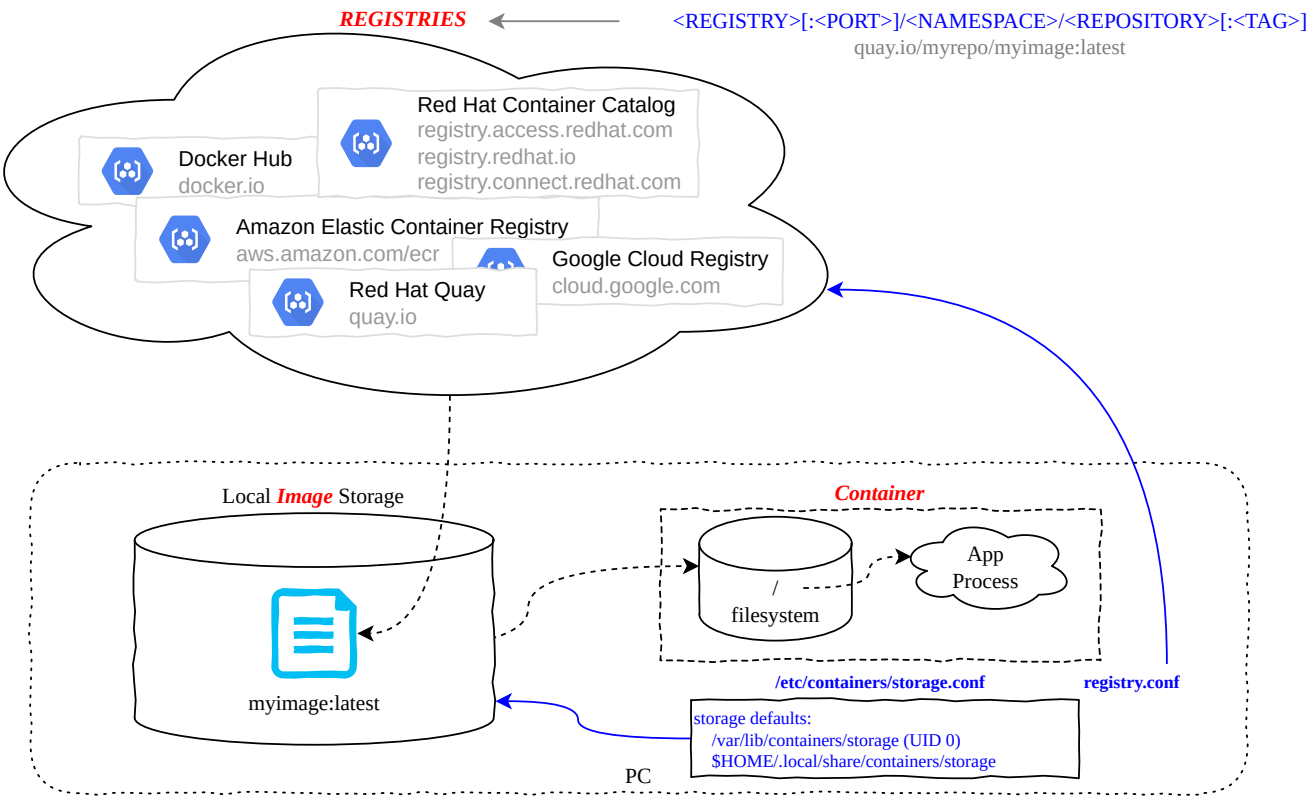


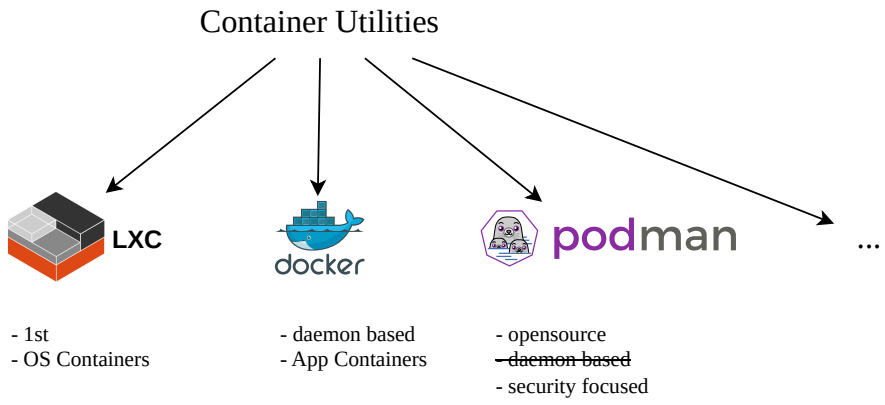
VM vs Container





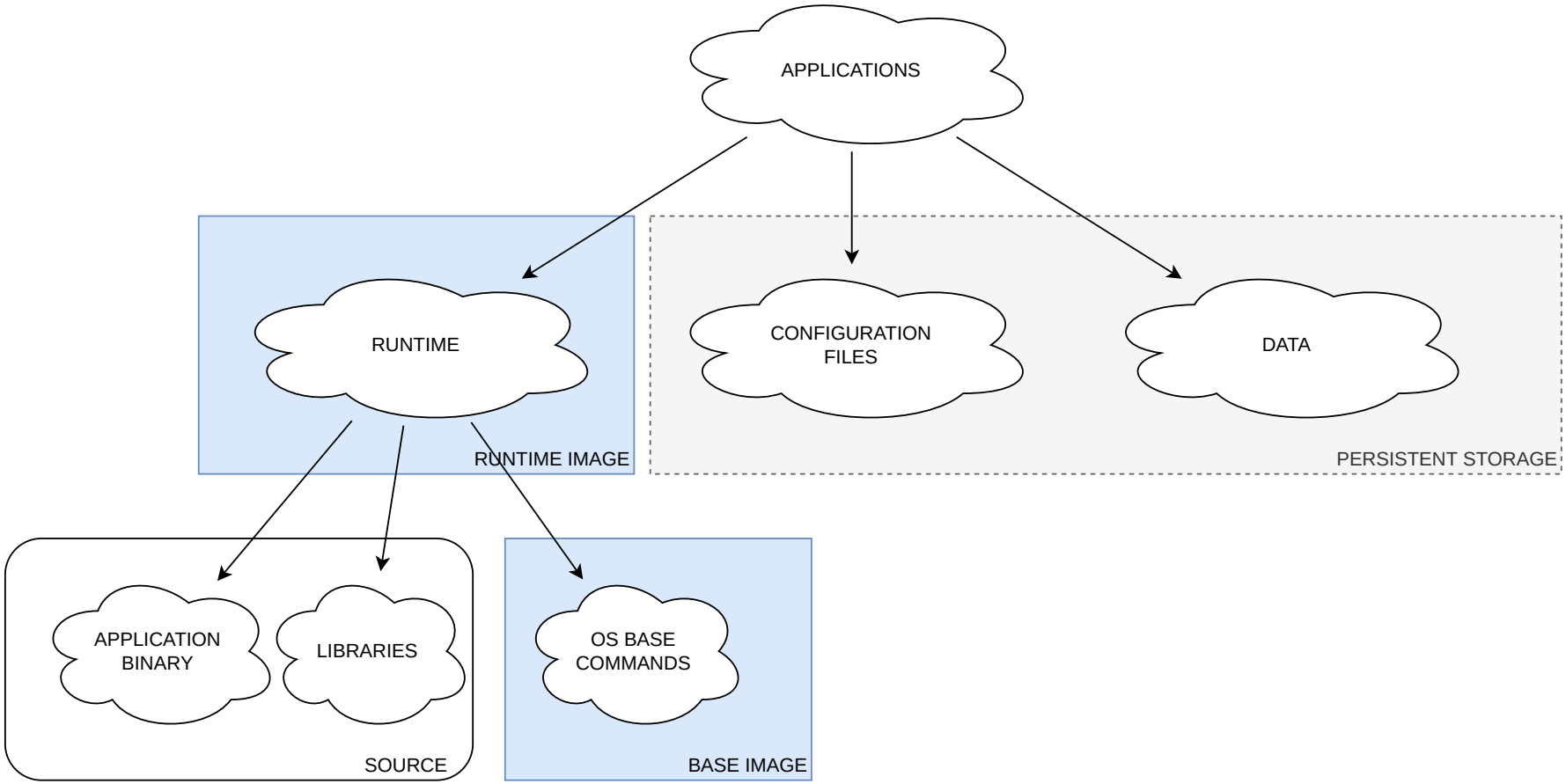
Container Architecture





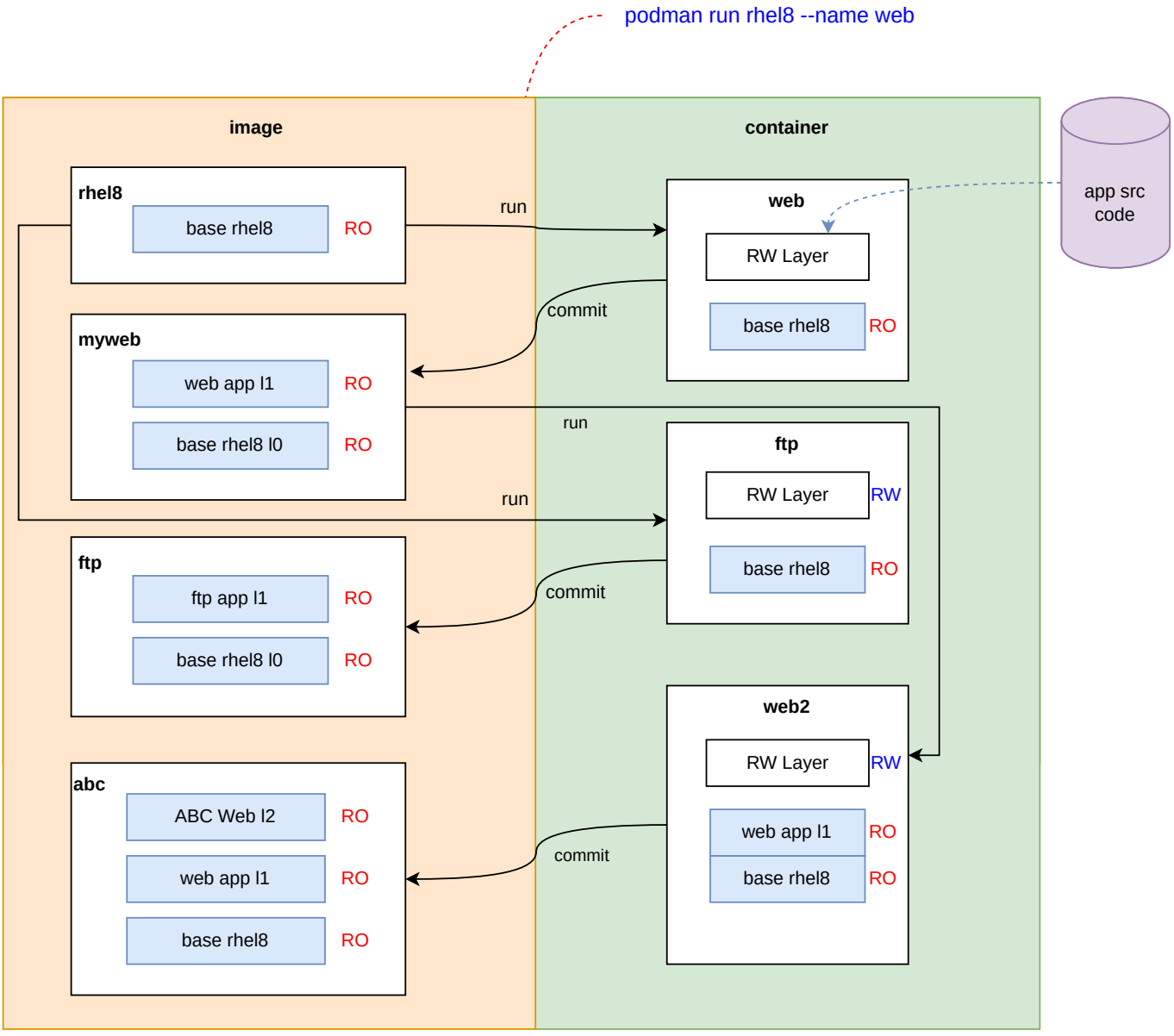
OS Container Vs Application Containers

Basic Container Design

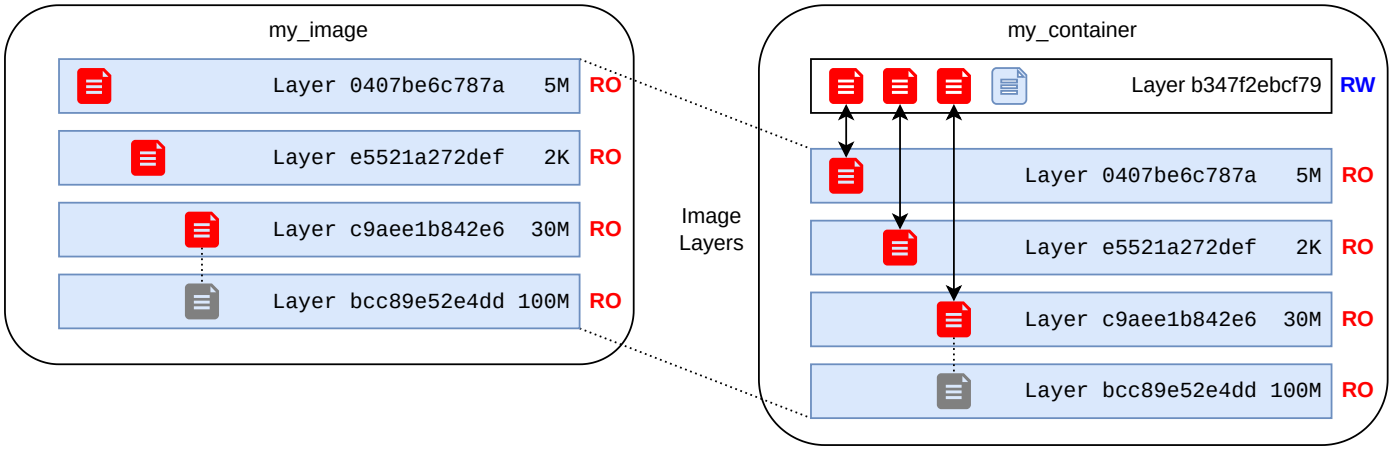


Creating Image

1. Manual
2. Dockerfile/Containerfile
3. Source-To-Image(s2i/STI)
   a) get runtime image and create container
   b) clone source code into container
   c) compile source code
   d) deploy/publish compiled app
   e) cleanup
   f) save container as image

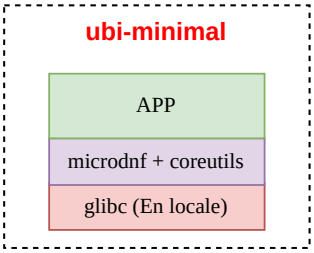


UnionFS - A Stackable Unification File System



BASE IMAGE TYPES

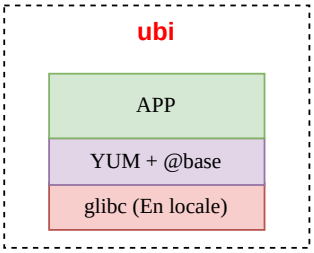
MINIMAL



Designed for apps that contain their own dependencies (Python, Node.js, .NET, etc.)

- Minimized pre-installed content set
- no suid binaries
- minimal pkg mgr (install, update & remove)

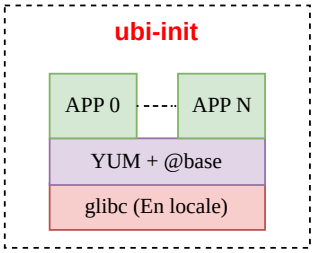
PLATFORM



For any apps that runs on RHEL

- Unified, OpenSSL crypto stack
- Full YUM stack
- Includes useful basic OS tools (tar, gzip, vi, etc)

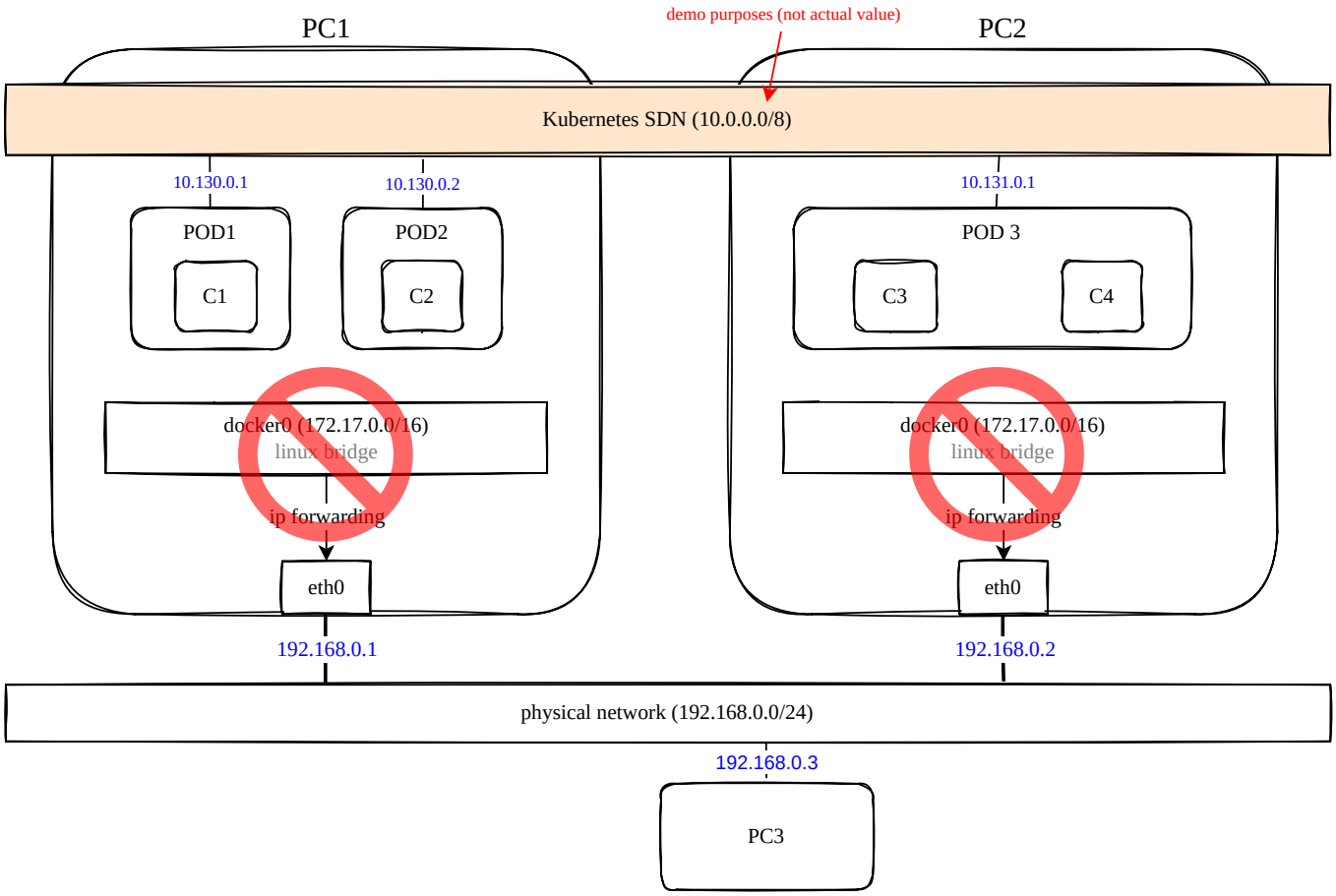
MULTI-SERVICE

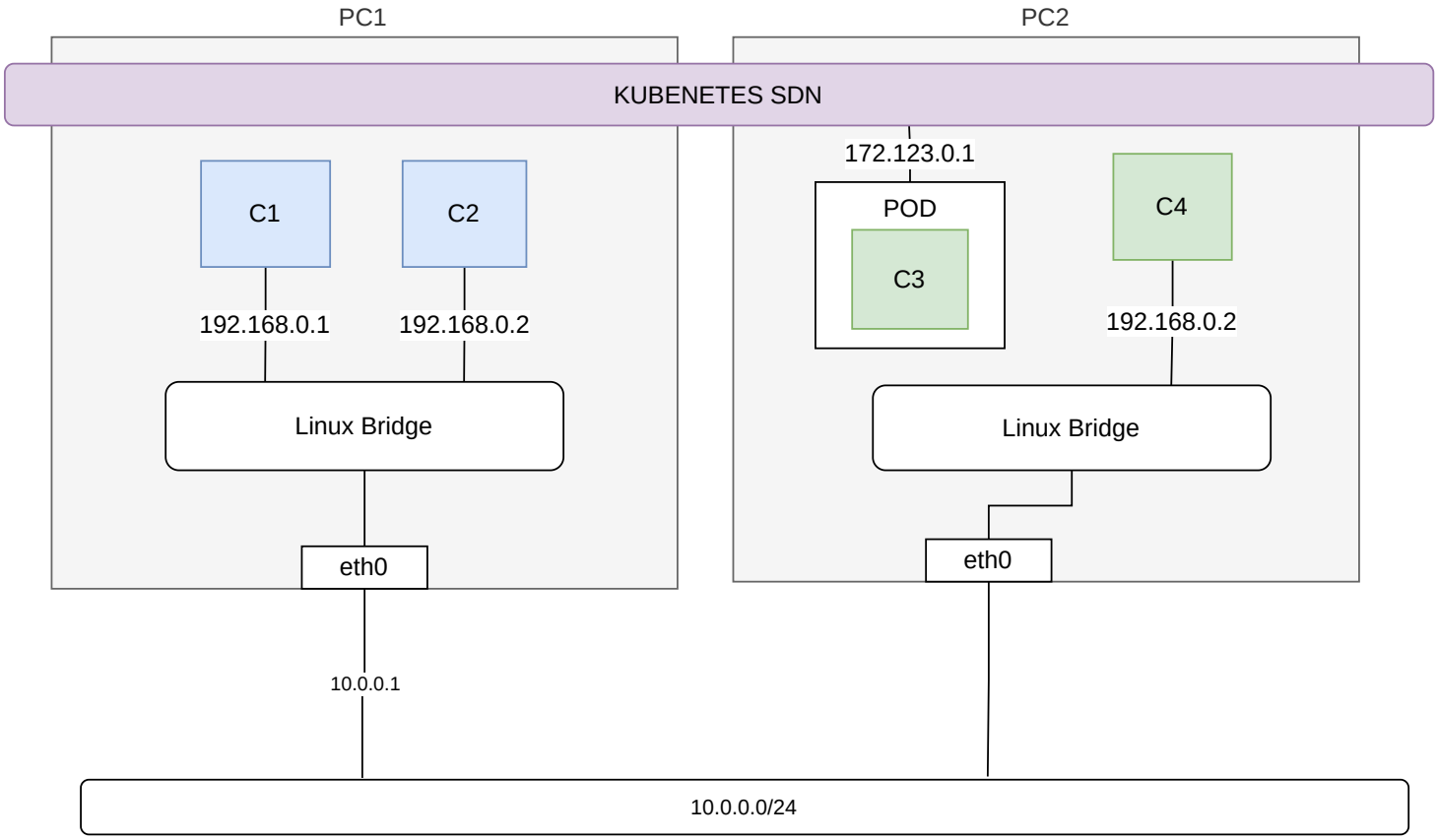


Eases running multi-service in single container

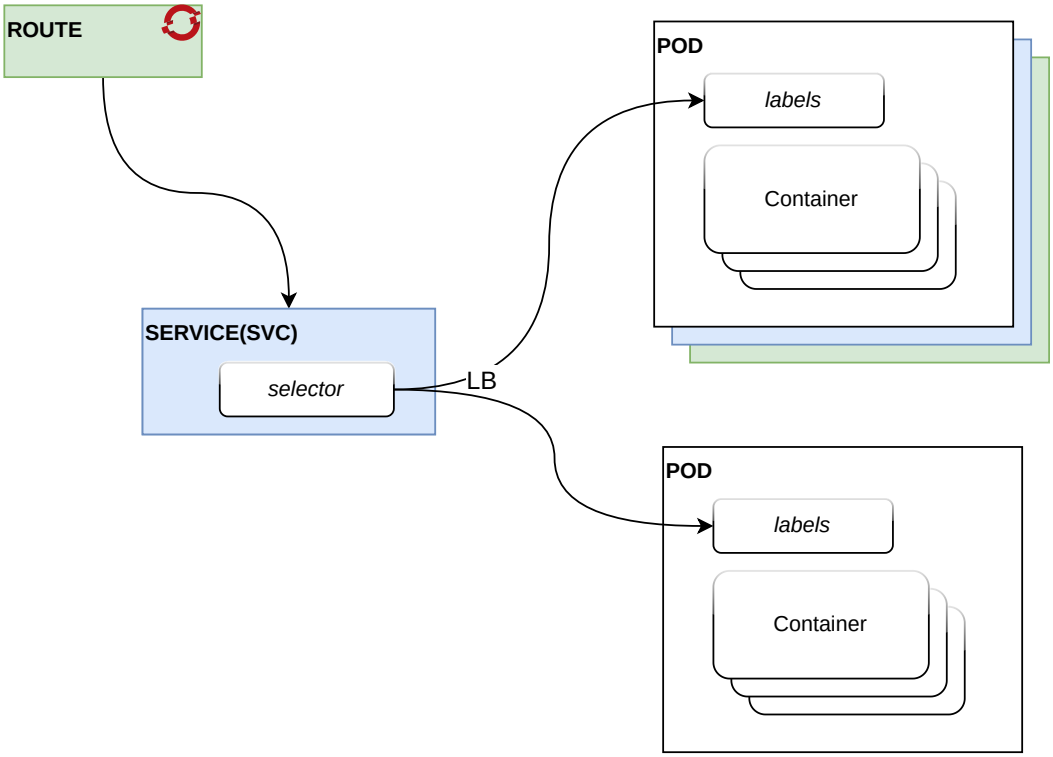
- configured to run systemd on start
- allows you to enable th services at build time

Basic Network - Container vs Kubernetes





Route, Service and Pod Relationship



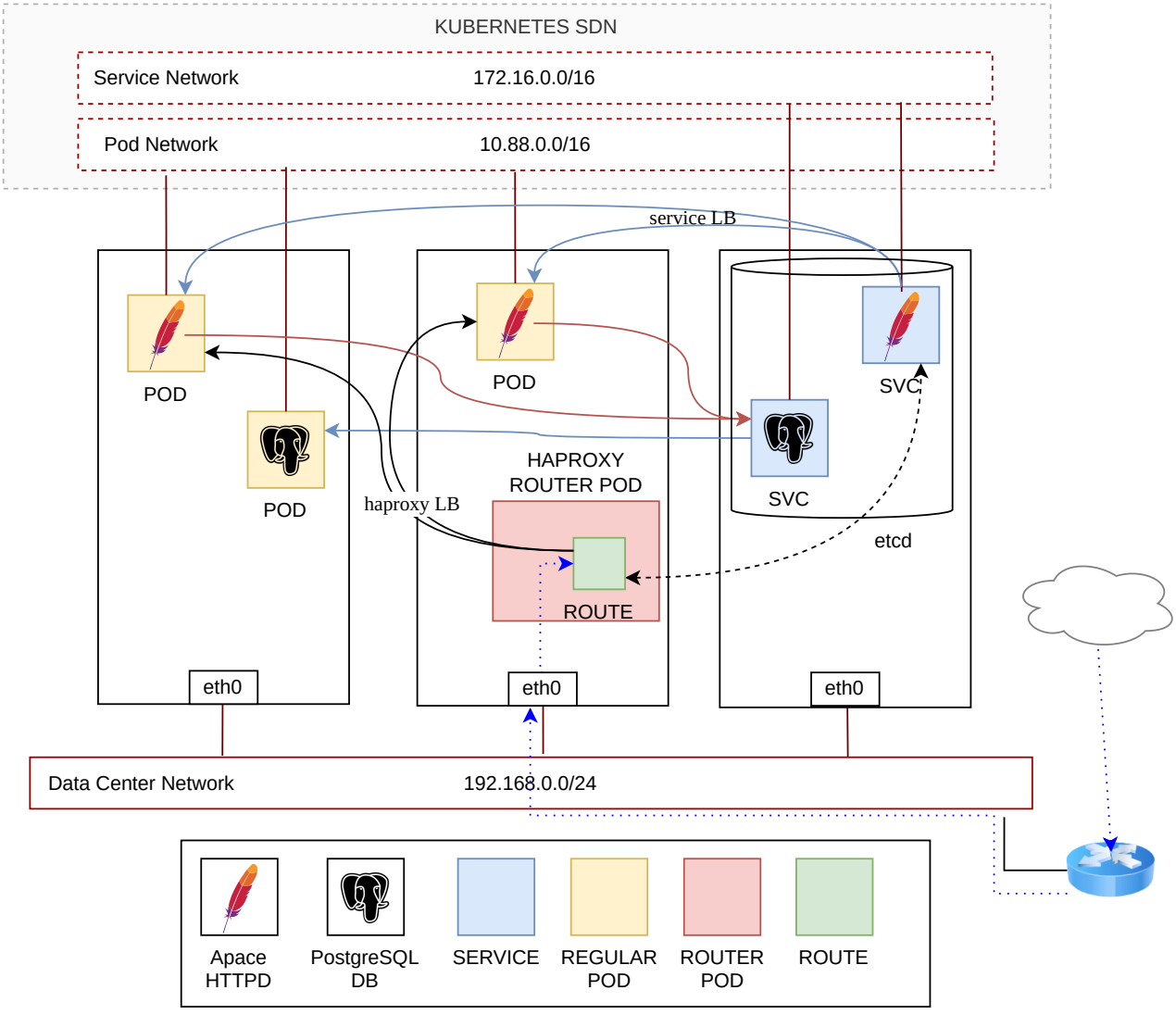
POD
A pod contains one or more containers.

SERVICE
A service references the pod(s) by using the label selector.
The service load balances the connections between all the pods.

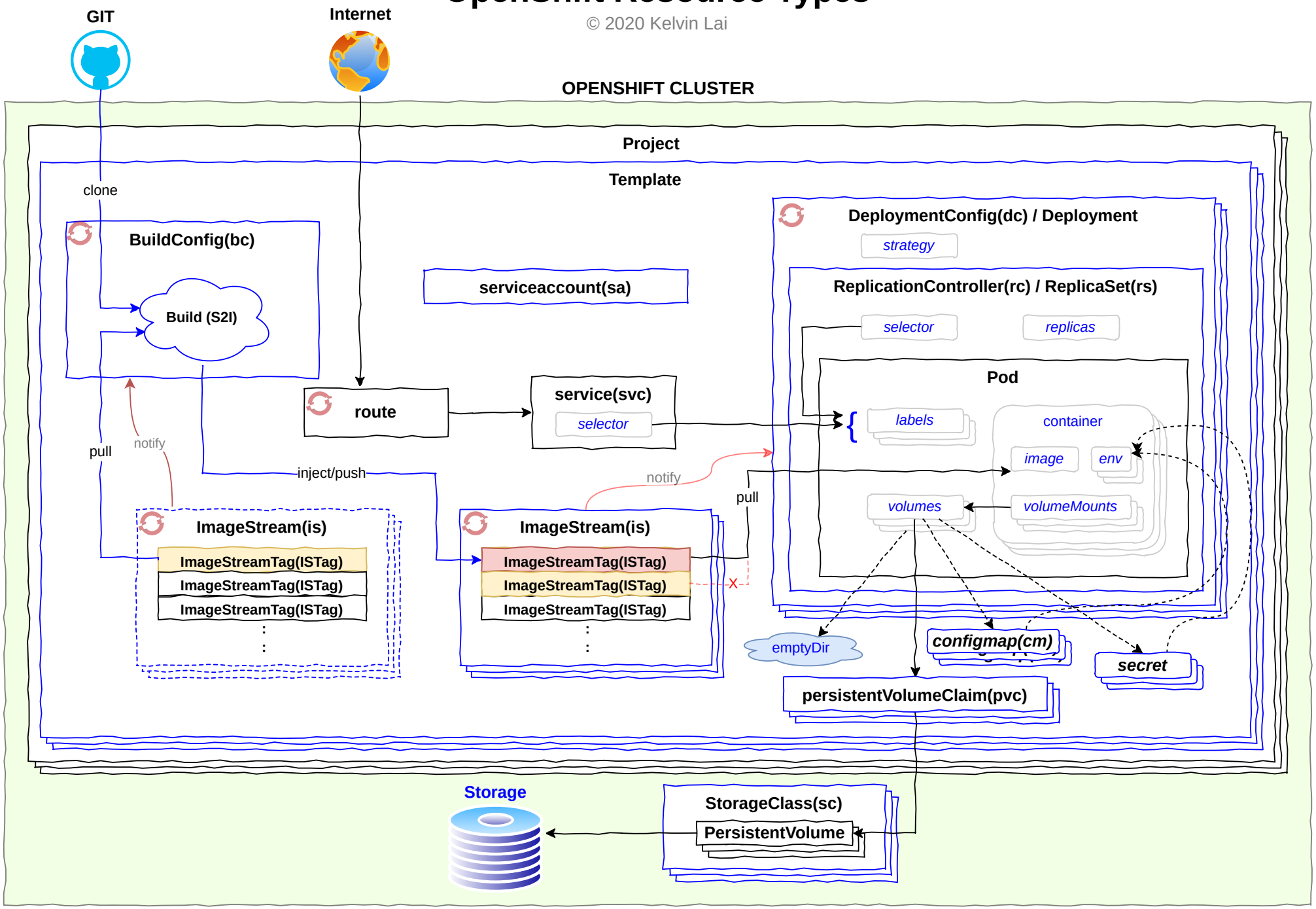
ROUTE
A route exposes the service to the external world.

Warning: A service "can" refer to different pods, if the pods have the same label.

Sample of how Services are used

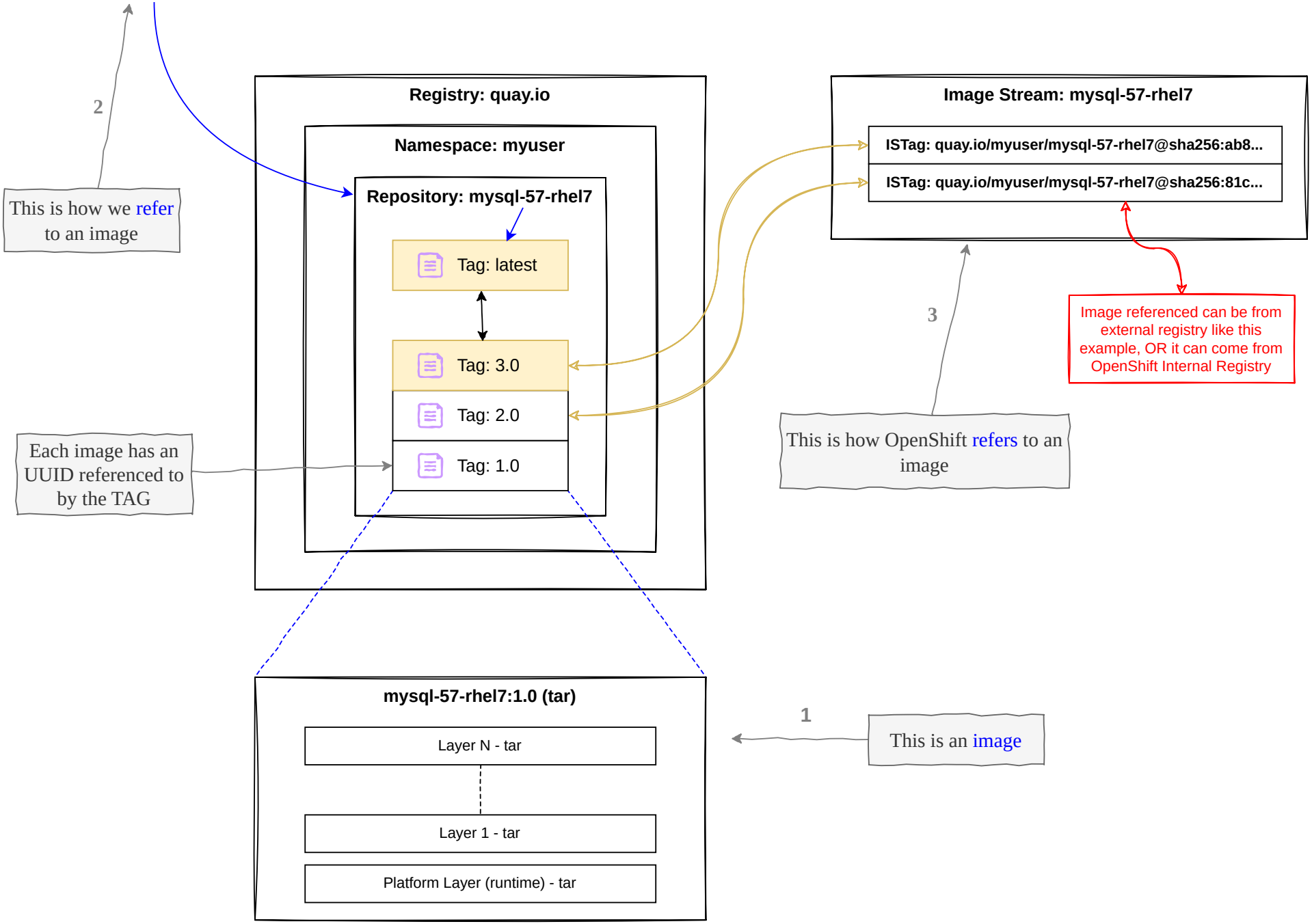


OpenShift Resource Types



Container Image Naming Convention: <REGISTRY>[:<PORT>]/<NAMESPACE>/<REPOSITORY>[:<TAG>]

podman pull quay.io/myuser/mysql-57-rhel7



Deploying Applications with OpenShift

Methods to create applications:

1. Using existing containerised applications

```
oc new-app --docker-image=<IMAGE>
```

2. From Source Code using S2I

```
oc new-app <URL>
```

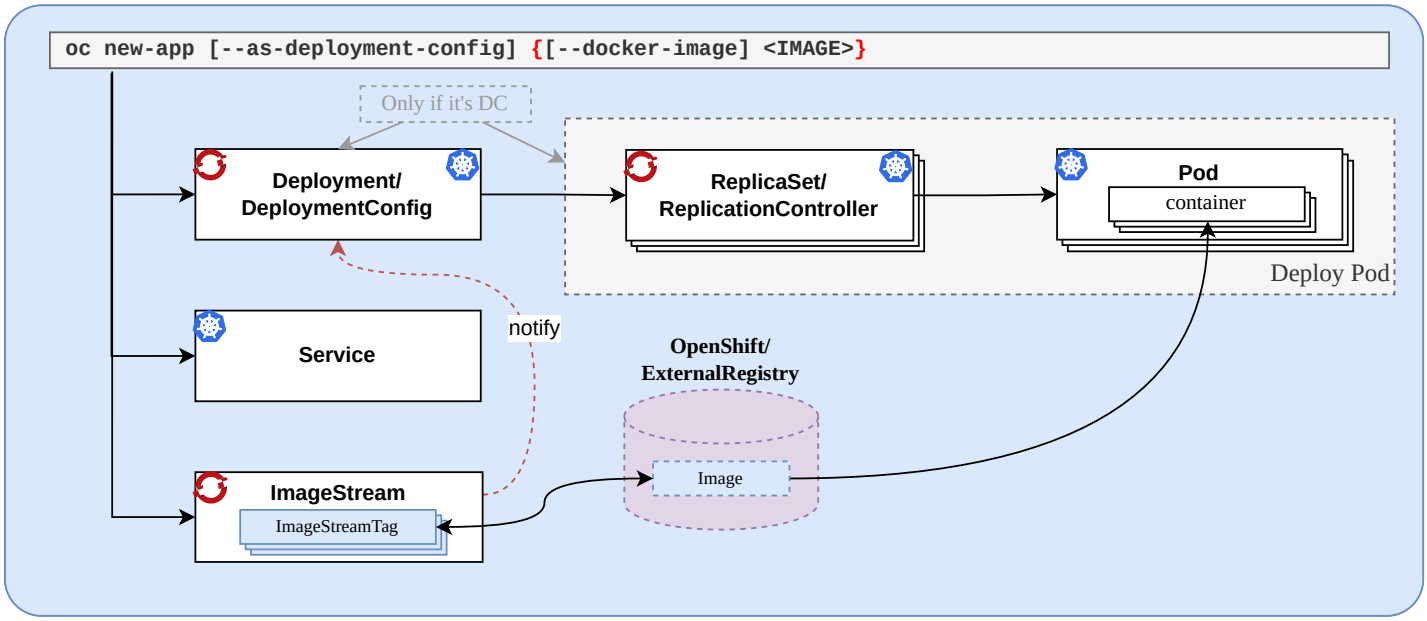
3. Using yaml/json file

```
oc new-app -f <FILE>.yaml
```

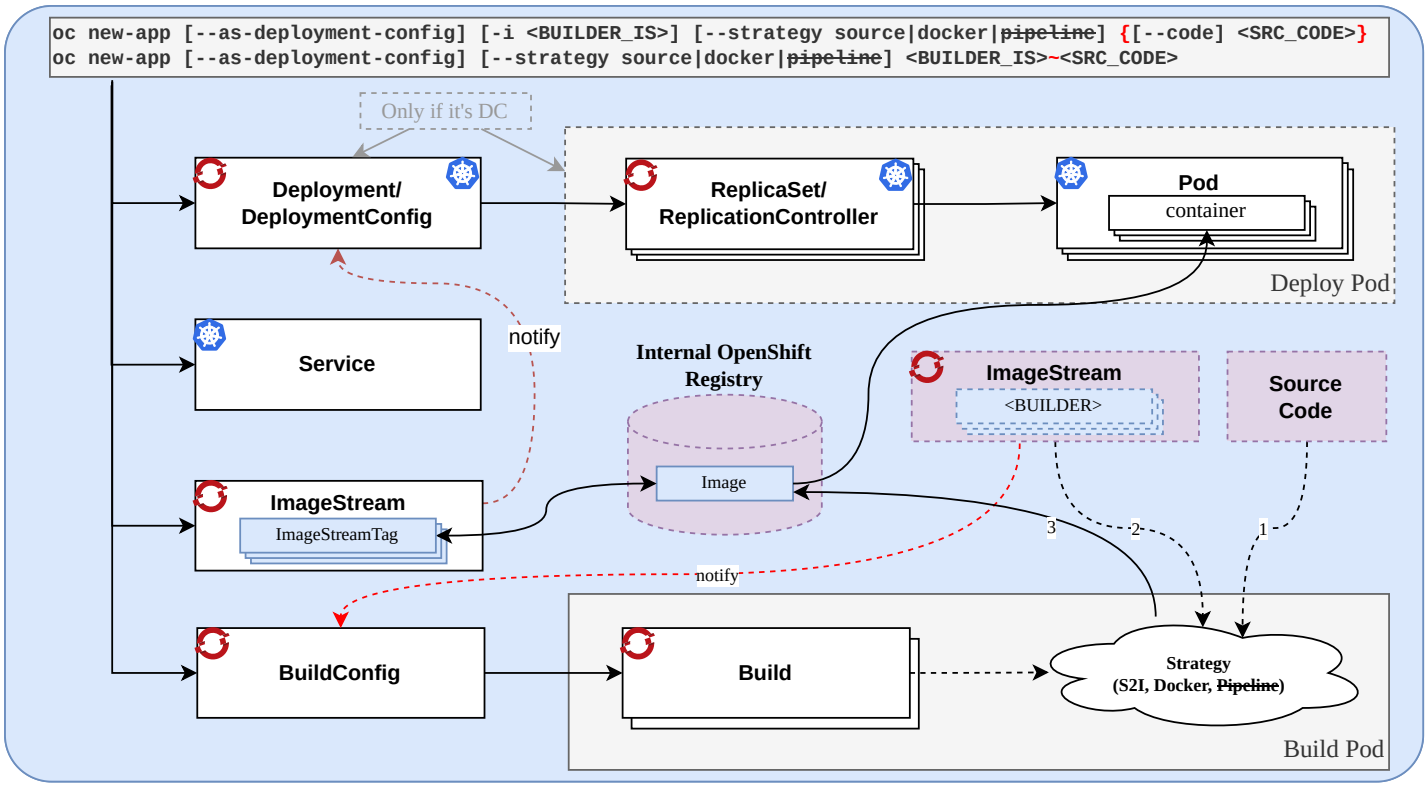
4. Using template

```
oc new-app --template=<TEMPLATE> --param=<PARAM> --param-file=<PARAM_FILE>
```

1. Use Existing Image



2. Managed Life Cycle



```
oc new-app -i myphp https://github.com/user/myapp#branch --context-dir <DIR>
```

```
oc new-app -i myphp:7.1 https://github.com/user/myapp
```

```
oc new-app myphp:7.1~https://github.com/user/myapp
```

NOTE: -i option needs git client to be installed

Options

-o json|yaml	inspect resource definitions without creating
--name <NAME>	adds a label "app=<NAME>" to all resources, Use ***oc delete all -l "app=<NAME>"*** to cleanup

IMPORT IMAGES

```
oc import-image <IMG_STREAM> [--confirm] --from <IMAGE> [--insecure]
where,
    <IMAGE> = <REGISTRY>[:<PORT>]/<NAMESPACE>/<REPOSITORY>[:<TAG>]
```

oc new-app command in OpenShift 4.5 makes use of *deployment* resource. Use --as-deployment-config if you wish to create *deployment config* instead.

oc expose <DC/DEPLOYMENT/RC/RS/POD> <RESOURCE_NAME>

DNS NAME = <SVC>.<PROJ>[.svc.cluster.local]
ENVIRONMENT VARIABLE IN POD = <SVC>_SERVICE_HOST

oc expose svc <SVC_NAME> [--name <ROUTE_NAME>] [--hostname <FQDN>]

DNS DEFAULT NAME = <ROUTE_NAME>-<PROJ>.<DOMAIN WILDCARD>
<DOMAIN_WILDCARD> = apps.<BASE_DOMAIN>