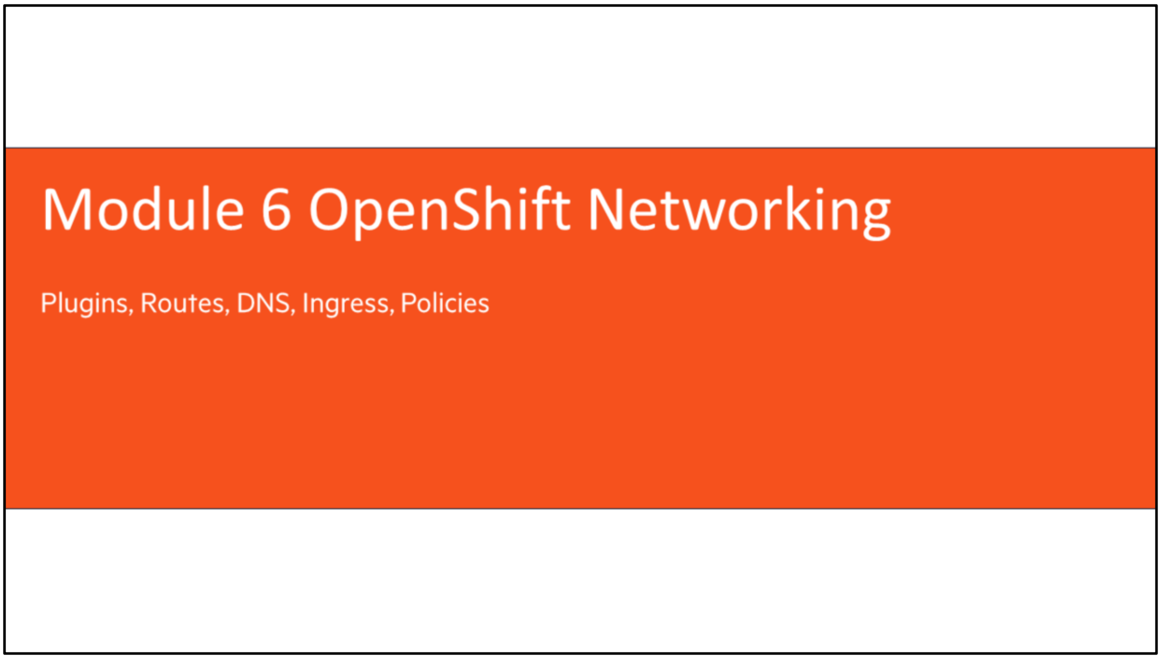
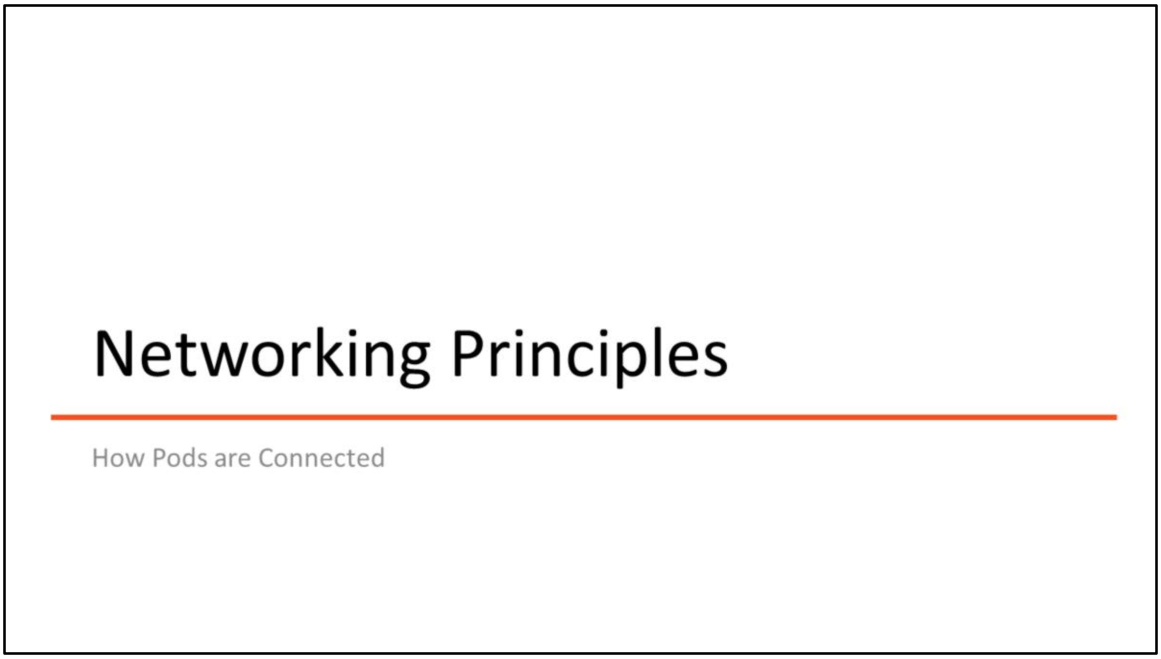


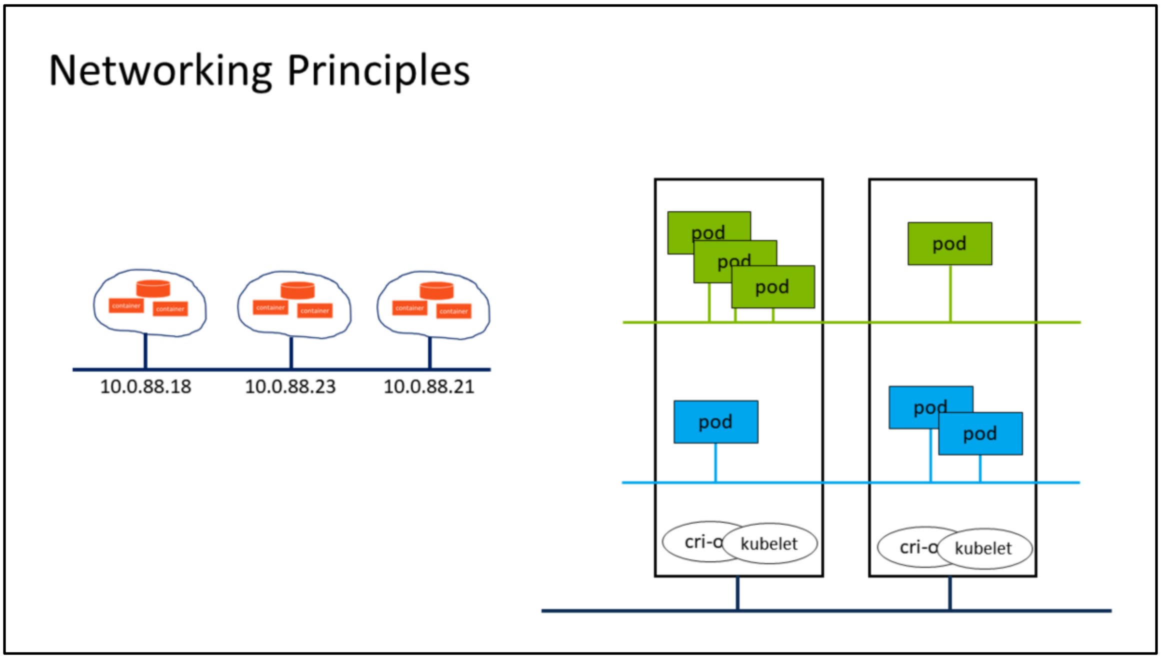
Module 6 OpenShift Networking

Plugins, Routes, DNS, Ingress, Policies



Networking Principles

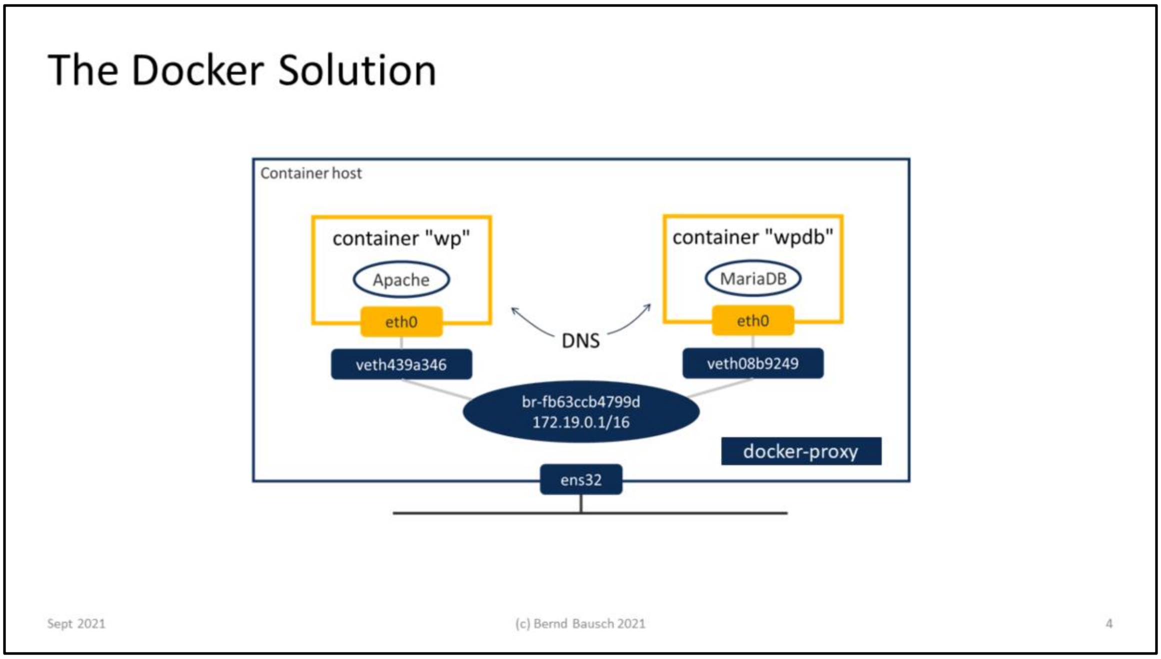
How Pods are Connected



Networking Principles

A pod is given an IP address. Thus, a pod looks like a machine to the containers running in it, and to other pods connecting to it.

Pods in a ReplicaSet or Deployment run in the same network. This network must be able to span container hosts and must be isolated from all other networks: Networks of other pods, a



The Docker Solution

Container host

container "wp"

Apache

eth0

veth439a346

DNS

container "wpdb"

MariaDB

eth0

veth08b9249

br-fb63ccb4799d
172.19.0.1/16

docker-proxy

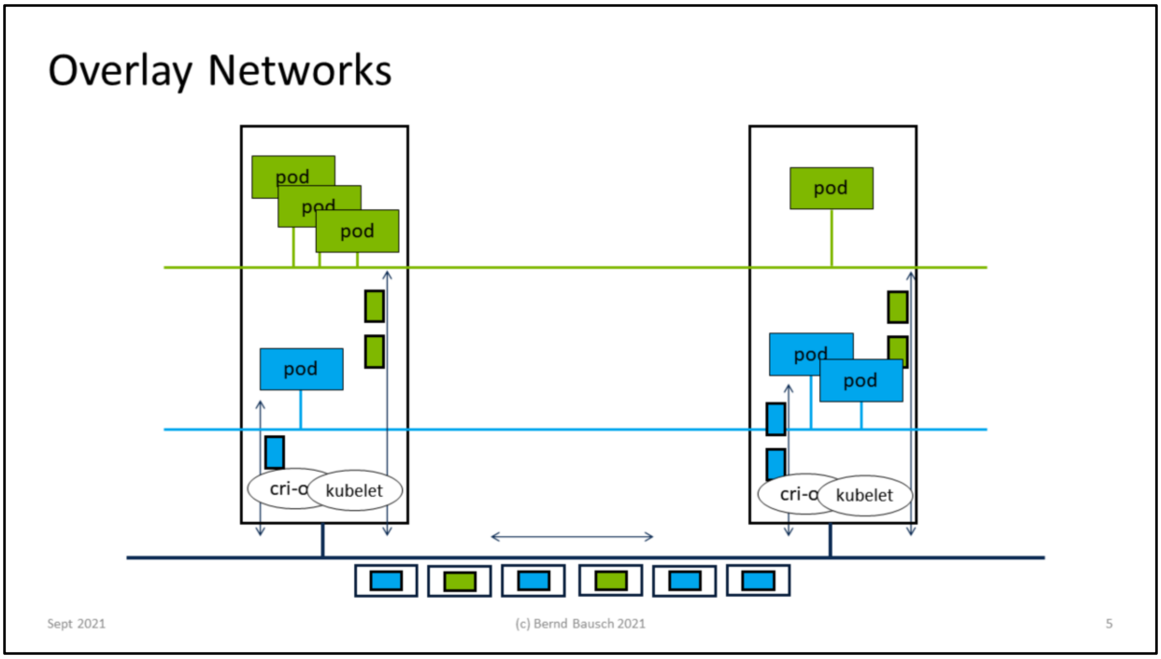
ens32

Sept 2021

(c) Bernd Bausch 2021

4

A Docker network on a single Docker host is usually implemented by a Linuxbridge. Access from and to the outside network is facilitated by docker-proxy. This is not enough when pods on the same virtual network are located on different hosts.

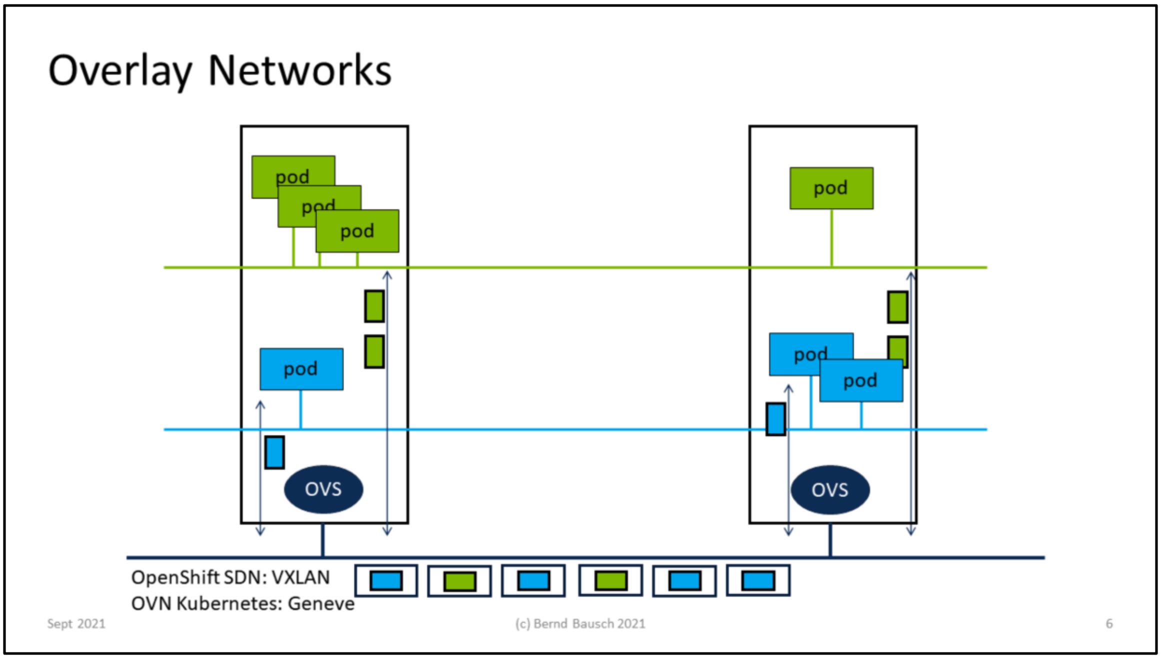


Overlay Networks

Sept 2021

5

In Kubernetes and OpenShift (and Docker Swarm), overlay networks are used to implement virtual pod networks that span hosts. An overlay network sends its layer-2 network packets over the datacenter network, packaged in layer-3 or layer-4 network packets.
Several overlay technologies are available for Kubernetes, such as Flannel or Calico.
Out of the box, OpenShift allows to configure a solution named OpenShift SDN or OVN-Kubernetes.

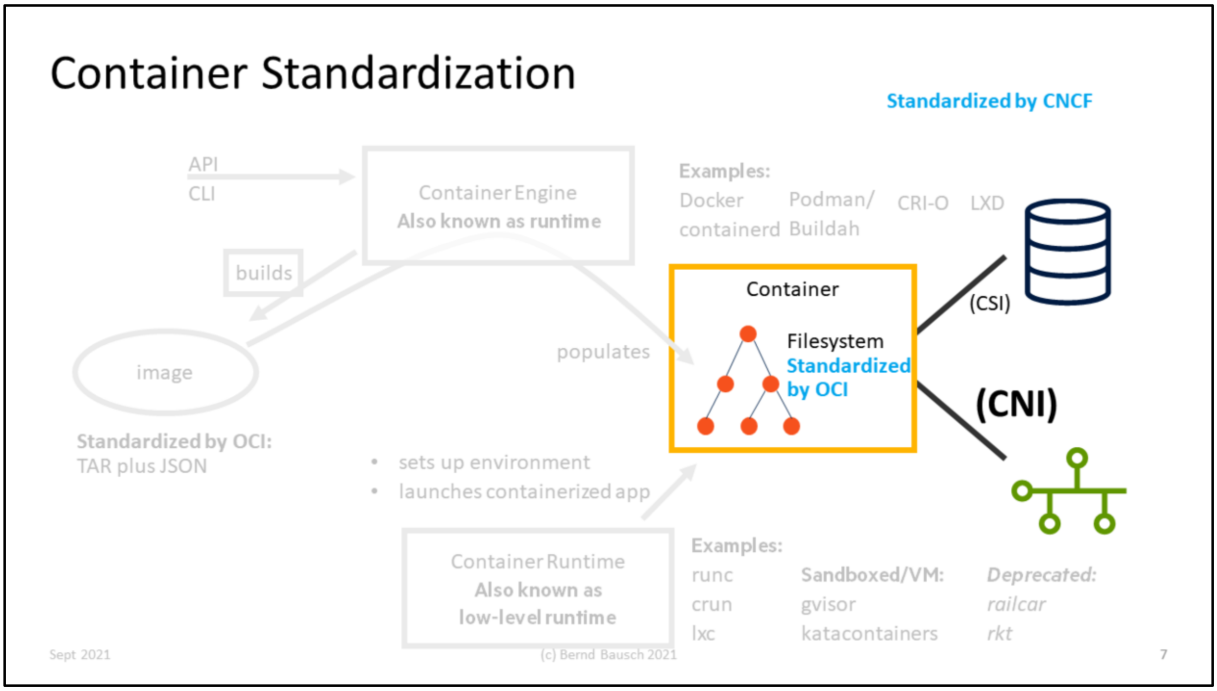


Overlay Networks

OpenShift SDN: VXLAN
OVN Kubernetes: Geneve

Sept 2021 (c) Bernd Bausch 2021 6

Both OpenShift network overlays are based on Openvswitch. This is a powerful software switch that can be programmed with Openflow and that includes overlay technologies. In OpenShift, VXLAN or Geneve are used as overlay network technologies; both send virtual network packets over UDP.
Comparison between the two solutions: https://docs.openshift.com/container-platform/4.8/networking/ovn_kubernetes_network_provider/about-ovn-kubernetes.html



Container Standardization

Standardized by CNCF

Examples:
Docker
containerd
Podman/
Buildah
CRI-O
LXD

Standardized by OCI

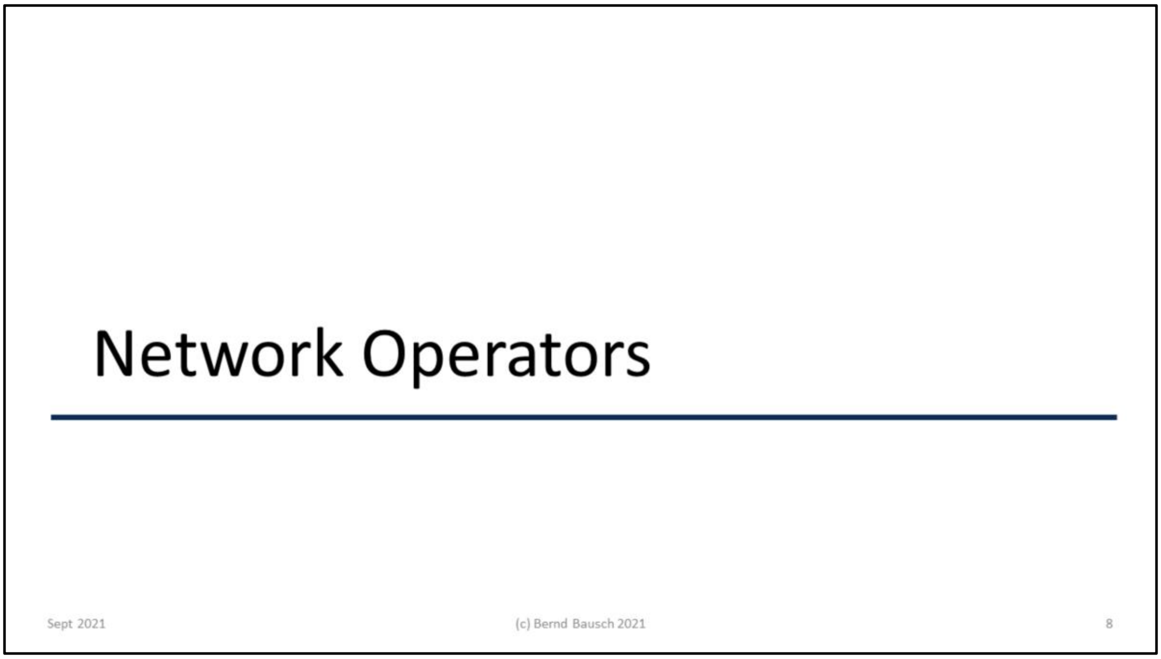
Standardized by OCI:
TAR plus JSON

- sets up environment
- launches containerized app

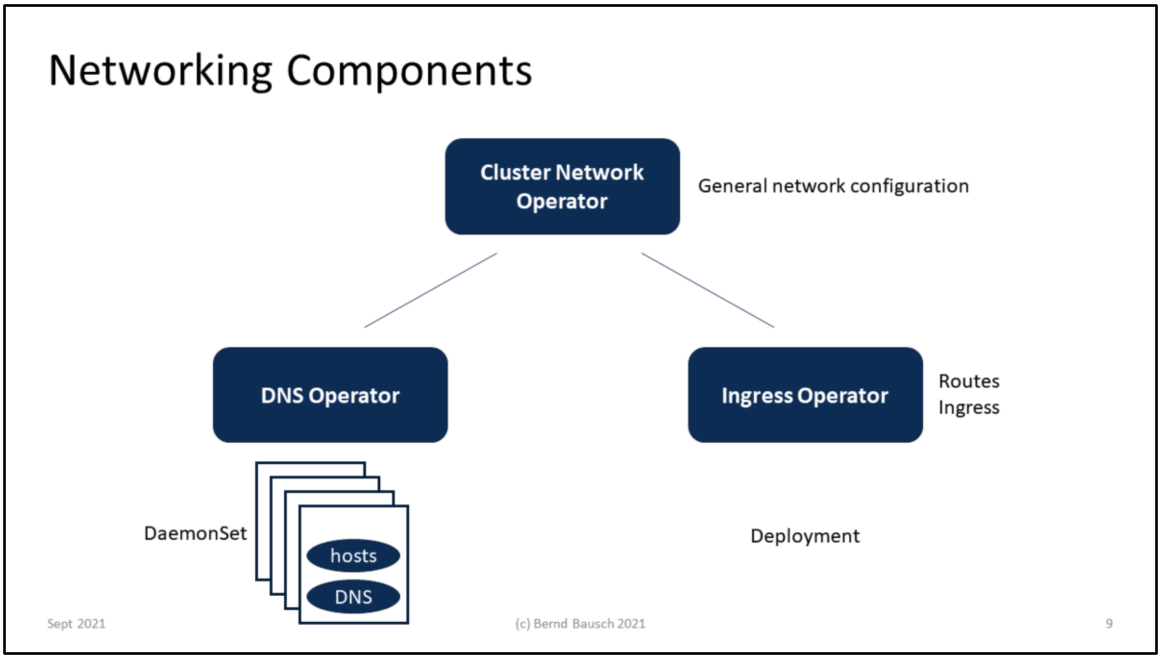
Examples:
runc
crun
lxc
Sandboxed/VM:
gvisor
katacontainers
Deprecated:
railcar
rkt

Sept 2021 (c) Bernd Bausch 2021 7

Both OpenShift networking solutions use the Kubernetes Container Network Interface CNI to plug into Kubernetes.



Network Operators



Networking Components

Cluster Network Operator

General network configuration

DNS Operator

Ingress Operator

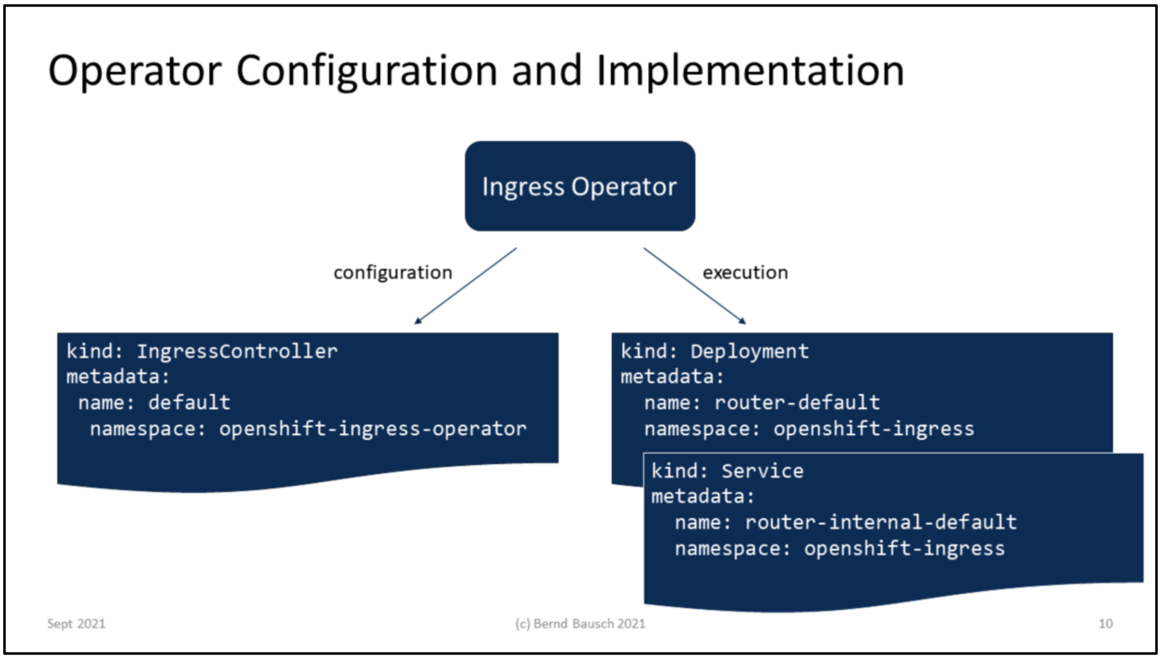
Routes
Ingress

DaemonSet

hosts

DNS

Deployment



Operator Configuration and Implementation

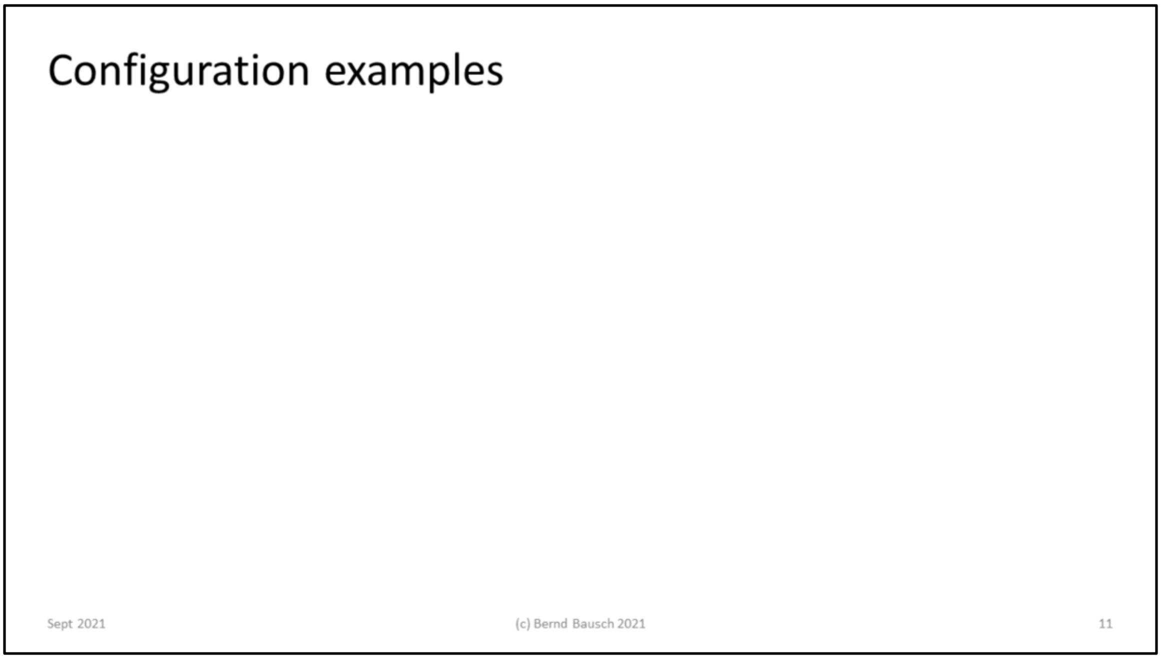
```
kind: IngressController
metadata:
 name: default
  namespace: openshift-ingress-operator
```

```
kind: Deployment
metadata:
  name: router-default
  namespace: openshift-ingress
```

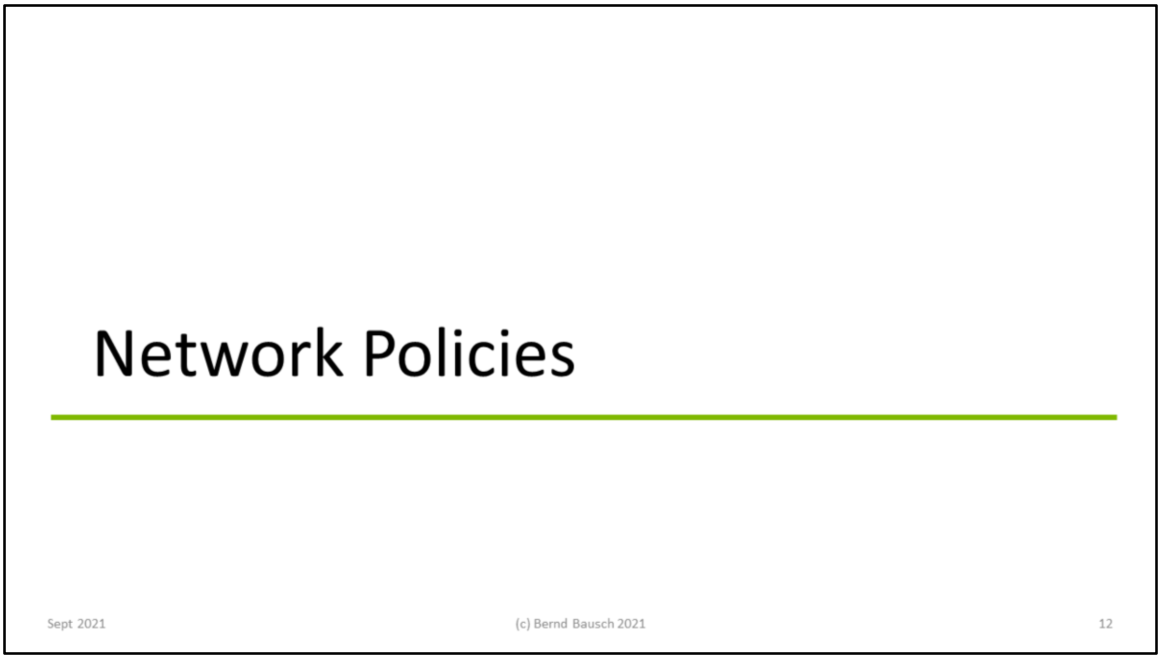
```
kind: Service
metadata:
  name: router-internal-default
  namespace: openshift-ingress
```

Sept 2021 (c) Bernd Bausch 2021 10

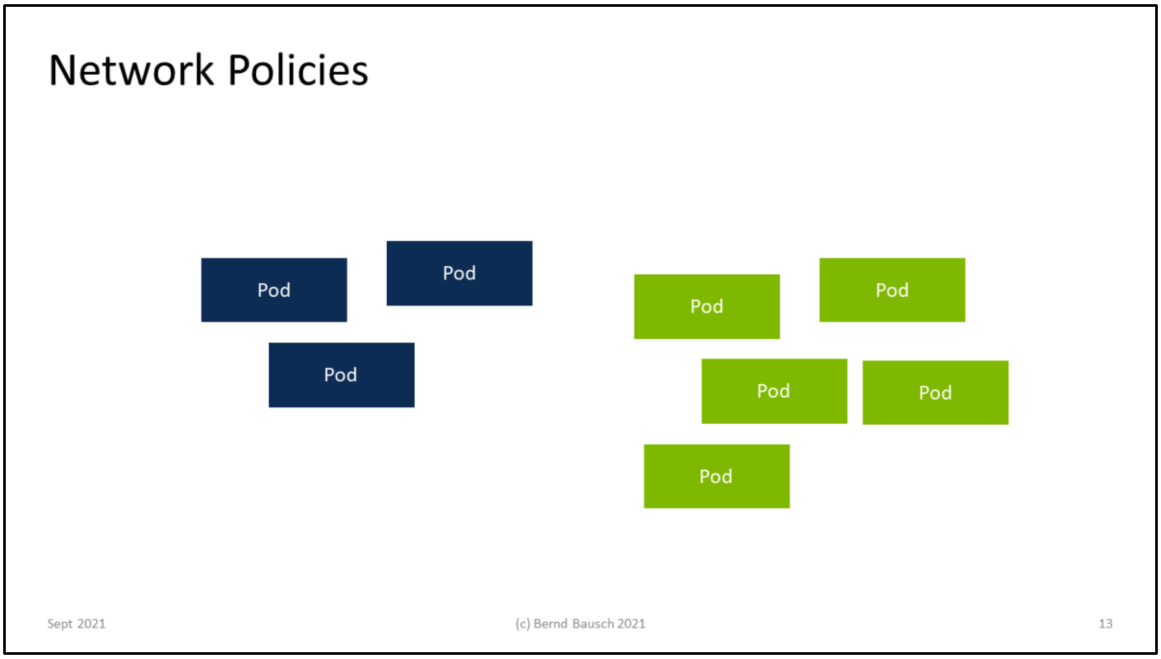
Operators tend to have a configuration resource and separate resources, such as DaemonSets or Deployments, that do the actual job. The two resources are usually in different namespaces. The example shows the Ingress operator, whose configuration resource, named default, resides in namespace openshift-ingress-operator, whereas the deployment that implementes Ingress and Route resources resides in openshift-ingress.



Configuration examples



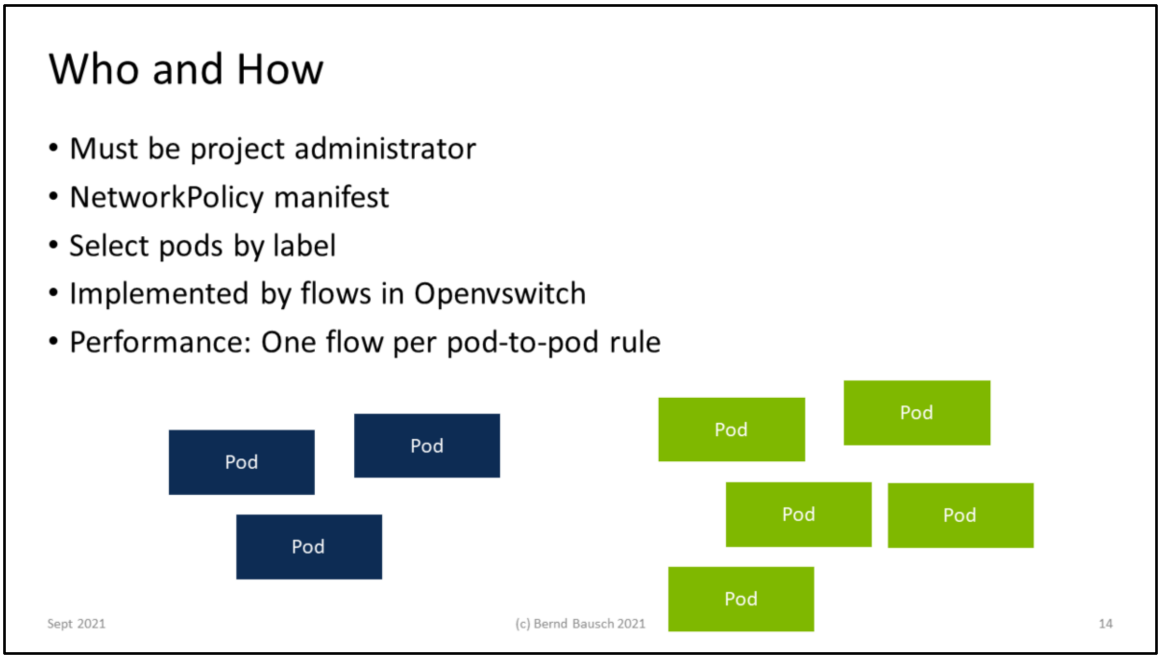
Network Policies



Network Policies

Sept 2021 (c) Bernd Bausch 2021 13

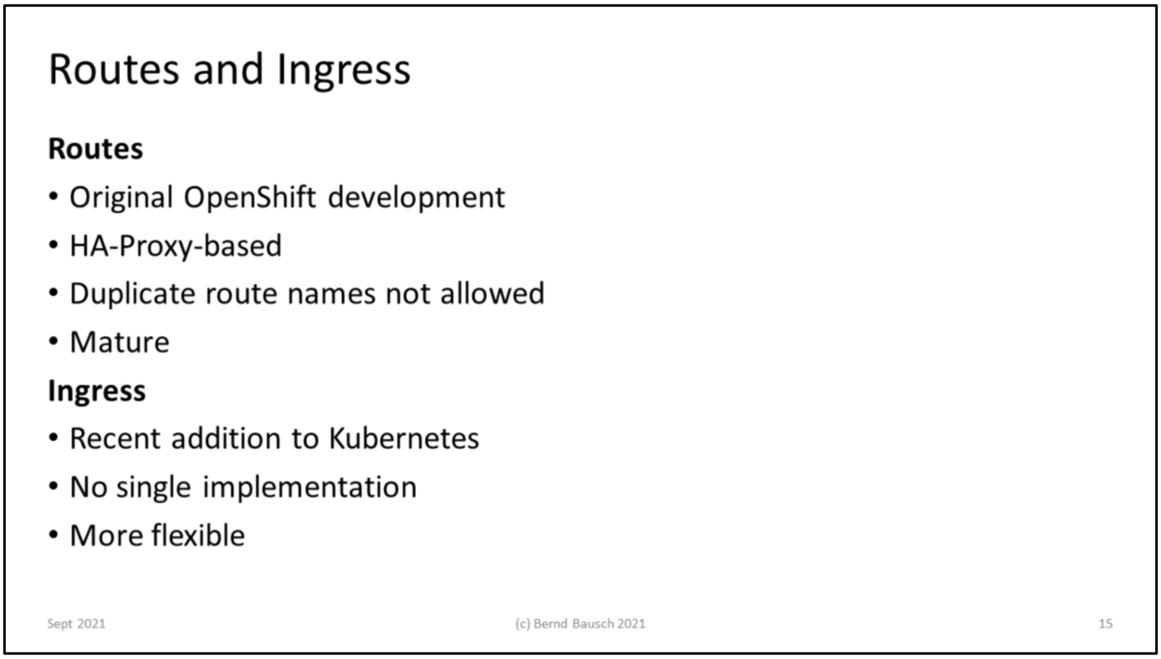
A network policy determines what traffic is allowed to enter or leave pods. Examples: All incoming traffic is denied; only traffic from sources in the same project is accepted; connections must come from the Ingress controller (through a route or an ingress), or limit traffic based on port or protocol.



Who and How

- Must be project administrator
- NetworkPolicy manifest
- Select pods by label
- Implemented by flows in Openvswitch
- Performance: One flow per pod-to-pod rule

Pod
Pod
Pod
Pod
Pod
Pod
Pod
Pod



Routes and Ingress

Routes

- Original OpenShift development
- HA-Proxy-based
- Duplicate route names not allowed
- Mature

Ingress

- Recent addition to Kubernetes
- No single implementation
- More flexible

Ingress and Routes are the same concept: Both determine to which service a URL is mapped. Routes were developed for OpenShift first; Ingress was added to Kubernetes later.