

Beyond Provisioning: Multi-Tenancy, Rack Awareness, and Declarative Networking in Metal3

Kashif Khan, Sunnatillo Samadov



About us





Kashif Khan

Maintainer @metal3.io

Co-chair @CNCF TAG Infrastructure

Principal Open source Architect @Ericsson

Sunnatillo Samadov

Open Source Developer, Ericsson

Maintainer, Metal3

Release lead, Cluster API v1.9

Agenda

1. Metal3 in brief
2. Multi-Tenancy
3. Failure Domains
4. ToR Switch Configuration
5. Where we are today?

KubeCon CloudNativeCon North America 2026

Metal3 in brief

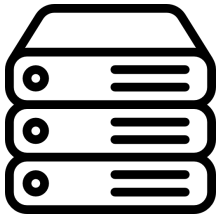


What is Metal3?



Bare metal host management, the Kubernetes way.

Pronounced “Metal Kubed” · CNCF Incubating since 2025



```
kind: BareMetalHost
  bmc: redfish://...
  image: os.qcow2
```





Declare a physical server as a Kubernetes resource — Metal3 provisions it and makes it a node.

Metal3 stack

Bare-metal host lifecycle, managed declaratively through Kubernetes APIs.

Cluster API
declarative cluster lifecycle

CAPI Provider Metal3 (CAPM3)
Machine → Metal3Machine

Baremetal Operator + Ironic
BareMetalHost → provisioning

IPAM

IrSO
deploys Ironic



Infrastructure

management | compute | network | storage

Key objects

BareMetalHost
one physical server — holds its BMC credentials and OS image

HardwareData
what inspection found — including LLDP neighbor data

Metal3Machine
the CAPI Machine's bare-metal backing

Bare metal is still manual where it matters

You can run many teams on one fleet — but three things still need hands-on work:

Sharing

Teams want self-service hardware — but that means handing out BMC passwords today.

Surviving

A whole rack can fail. The cluster has to keep its quorum when it does.

Wiring

Switch ports need the right VLANs — today a person types that on the switch CLI.

Compute became declarative. Bare-metal tenancy, placement and networking are still manual.

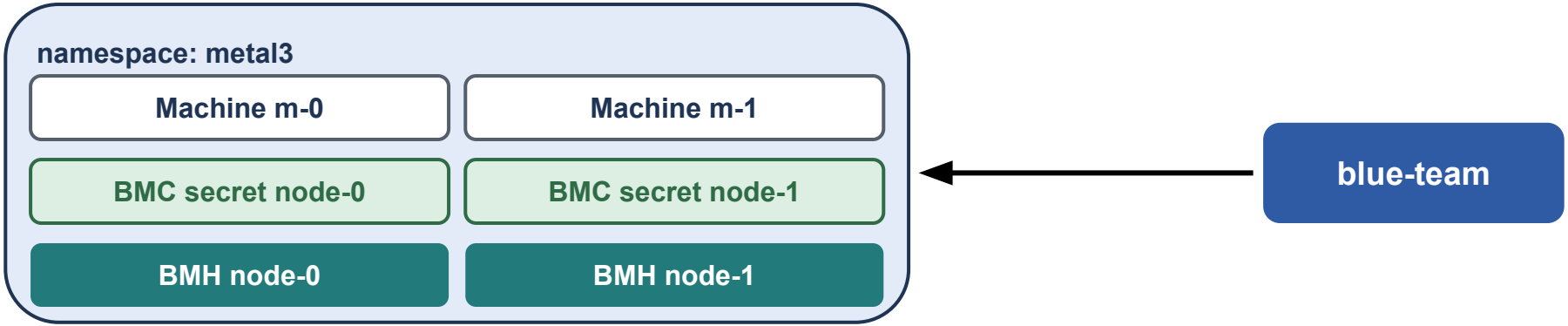
Needed by AI, edge, and sovereign-cloud platforms.



Multi-Tenancy

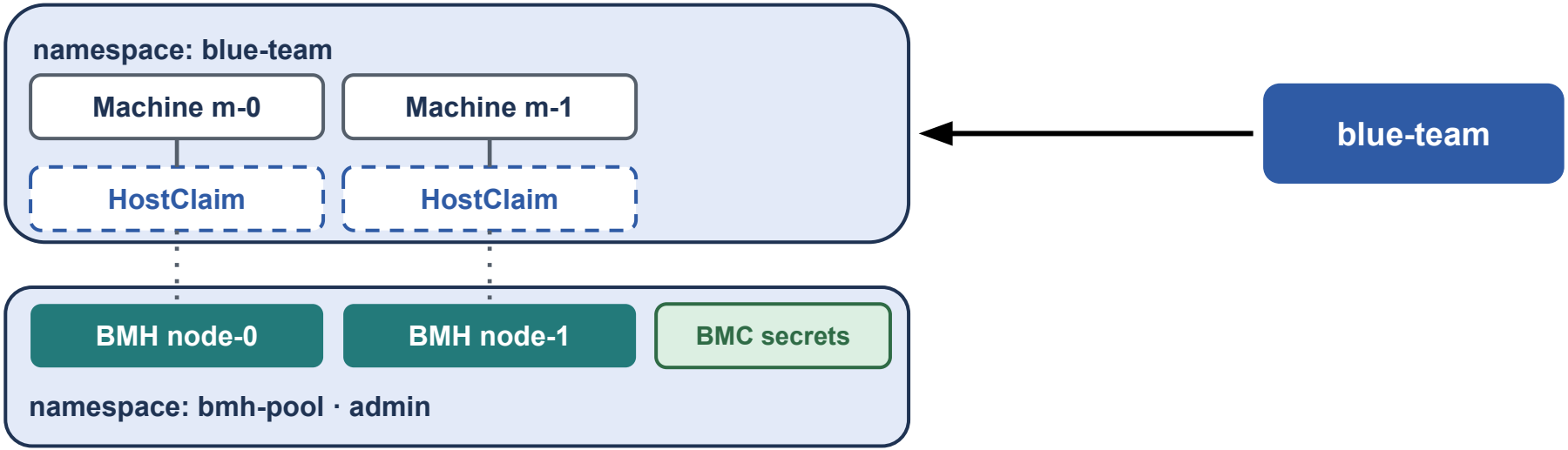


Without Multi-Tenancy



One namespace: blue-team reaches every host, every BMC secret, every Machine.

With HostClaim



Separate namespaces: blue-team reaches only its Machines.

Admin allows, tenant requests

1 **Rule** · admin

```yaml
apiVersion: metal3.io/v1alpha1
kind: HostDeployPolicy
metadata:
  name: allow-blue-team
  namespace: bmh-pool-ns
spec:
  hostClaimNamespaces:
    names:
      - blue-team
```

2 **Request** · tenant

```yaml
apiVersion: metal3.io/v1alpha1
kind: HostClaim
metadata:
  name: blue-web-01
  namespace: blue-team
spec:
  hostSelector:
    matchLabels:
      hardware.metal3.io/class: compute
```

A host binds only when both agree. Default is deny.

Admin allows, tenant requests

1 **Rule** · admin

```yaml
apiVersion: metal3.io/v1alpha1
kind: HostDeployPolicy
metadata:
  name: allow-blue-team
  namespace: bmh-pool-ns
spec:
  hostClaimNamespaces:
    names:
      - blue-team
```

2 **Request** · tenant

```yaml
apiVersion: metal3.io/v1alpha1
kind: HostClaim
metadata:
  name: blue-web-01
  namespace: blue-team
spec:
  hostSelector:
    matchLabels:
      hardware.metal3.io/class: compute
```

A host binds only when both agree. Default is deny.

KubeCon CloudNativeCon
North America 2026

HostClaim Demo



KubeCon CloudNativeCon
North America 2026

Failure Domains



Rack awareness: lose a rack, keep the cluster



HA is only real if the control plane is physically spread.

Define the labels and the failure domains

```yaml
apiVersion: metal3.io/v1alpha1
kind: BareMetalHost
metadata:
  name: node-0
  labels:
    infrastructure.cluster.x-k8s.io/failure-domain: rack1
```

```yaml
apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
kind: Metal3Cluster
metadata:
  name: failure-domain-test
spec:
  failureDomains:
  - {name: rack1, controlPlane: true}
  - {name: rack2, controlPlane: true}
  - {name: rack3, controlPlane: true}
```

1 Label each host

One label on every BareMetalHost says which rack it lives in.

2 List the domains

The Metal3Cluster declares the domains. KCP spreads control-plane nodes across them.

Define the labels and the failure domains

```yaml
apiVersion: metal3.io/v1alpha1
kind: BareMetalHost
metadata:
  name: node-0
  labels:
    infrastructure.cluster.x-k8s.io/failure-domain: rack1
```

```yaml
apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
kind: Metal3Cluster
metadata:
  name: failure-domain-test
spec:
  failureDomains:
  - {name: rack1, controlPlane: true}
  - {name: rack2, controlPlane: true}
  - {name: rack3, controlPlane: true}
```

1 Label each host

One label on every BareMetalHost says which rack it lives in.

2 List the domains

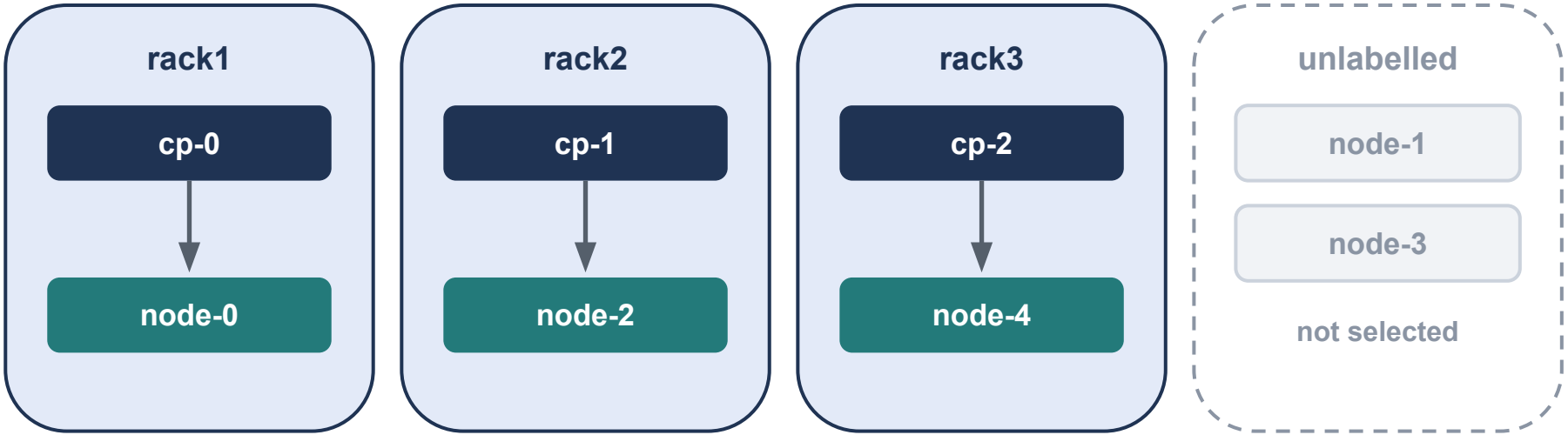
The Metal3Cluster declares the domains. KCP spreads control-plane nodes across them.

Two inputs, automatic placement



Label hosts by rack once. Cluster API spreads the control plane for you.

Demo setup: three racks, one control plane each



KCP (replicas: 3) places one control plane per failure domain.

KubeCon CloudNativeCon North America 2026

Failure Domain Demo

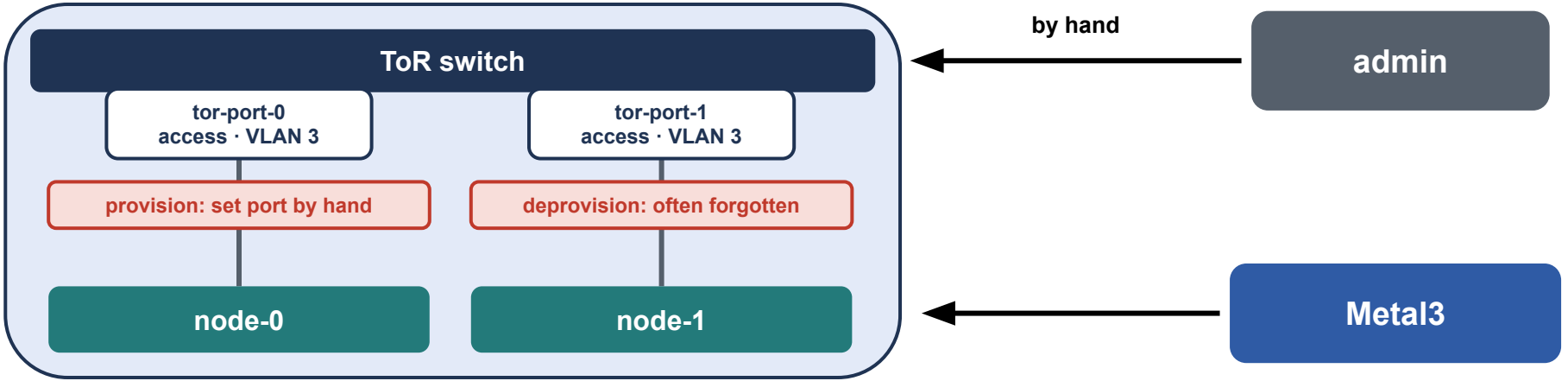


KubeCon CloudNativeCon
North America 2026

Top-of-Rack Switch Configuration

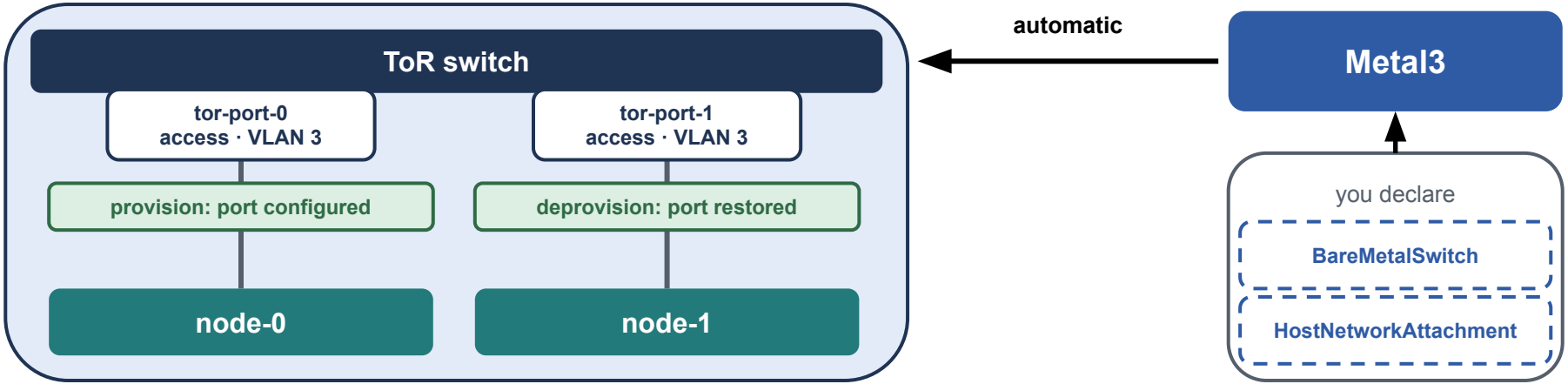


Switch config is a manual step



Every new host waits on a manual switch change.

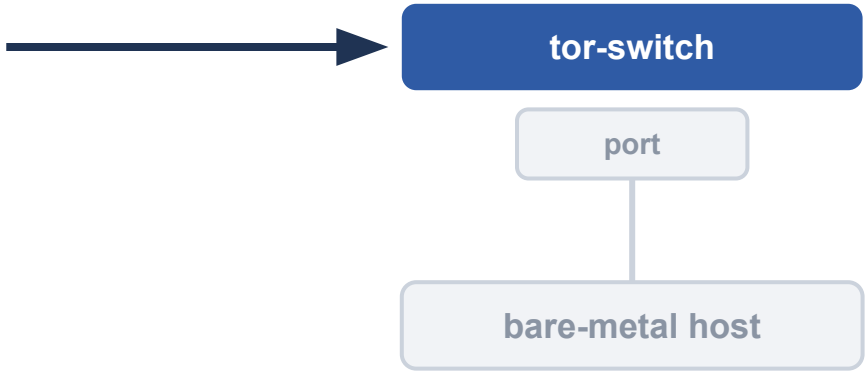
Declare it, Metal3 applies it



Declare it once. Metal3 sets the port and restores it.

BareMetalSwitch

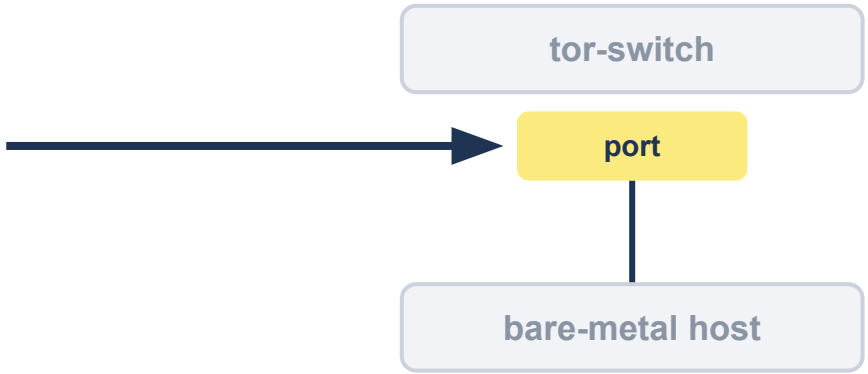
```yaml
apiVersion: metal3.io/v1alpha1
kind: BareMetalSwitch
metadata:
  name: tor-switch
spec:
  address: 172.22.0.18
  macAddress: 00:00:5e:00:53:01
  deviceType: netmiko_ovs_linux
  credentials:
    name: tor-switch-creds
```



Registers the ToR switch with Metal3: address, driver and credentials.

HostNetworkAttachment

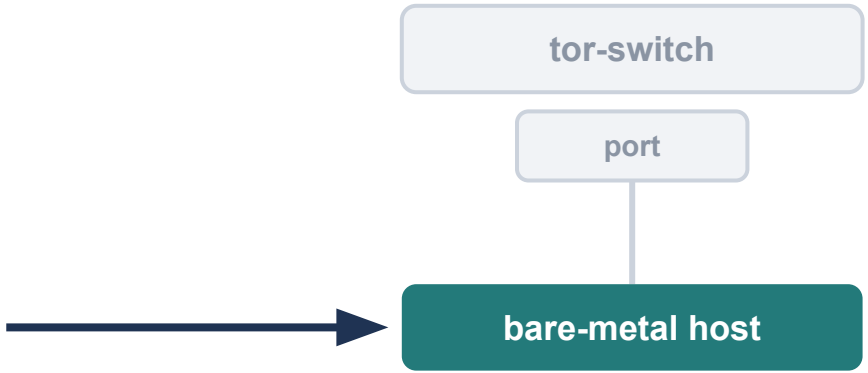
```yaml
apiVersion: metal3.io/v1alpha1
kind: HostNetworkAttachment
metadata:
  name: tenant-vlan3
  namespace: baremetal-operator-system
spec:
  mode: access
  nativeVLAN: 3
```



Describe what the port should look like.

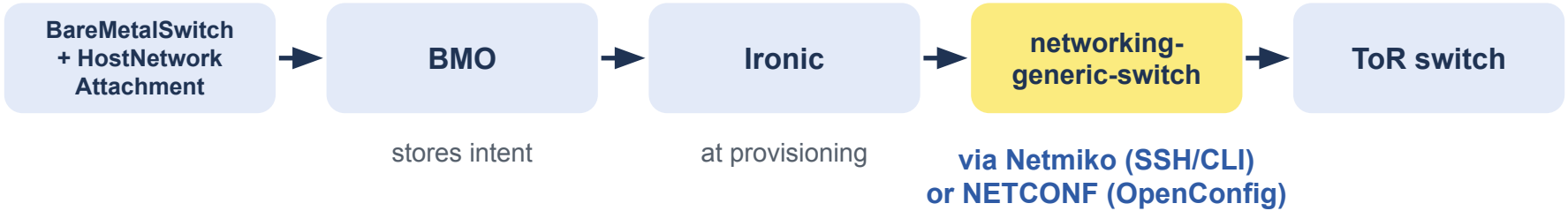
Attach the config to BMH

```yaml
apiVersion: metal3.io/v1alpha1
kind: BareMetalHost
spec:
  networkInterfaces:
  - macAddress: 52:54:00:a3:7f:1c
    hostNetworkAttachment:
      name: tenant-vlan3
```



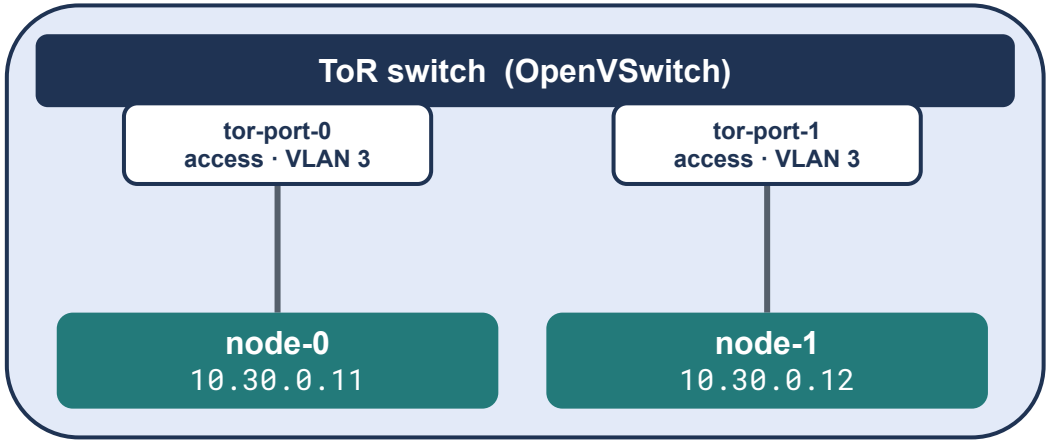
Point the host's NIC at that description.

Under the hood



BMO stores the intent. At provisioning, Ironic programs the real switch port.

Demo setup: two nodes on VLAN 3



Both ports in access mode on VLAN 3, so node-0 and node-1 reach each other.

KubeCon CloudNativeCon
North America 2026

ToR Demo



Where we are today?

Tenancy	**HostClaim + HostDeployPolicy** Tenants claim hosts. Admins own BareMetalHosts.	alpha · feature gate
Placement	**Failure domains** Control-plane nodes spread across racks.	available
Networking	**BareMetalSwitch + HostNetworkAttachment** Metal3 configures the switch ports.	v0.14.0 · feature gate

Try it, file issues, and join the Metal3 community meeting.

KubeCon CloudNativeCon
North America 2026

Thank you.
Questions?

metal3.io | github.com/metal3-io

