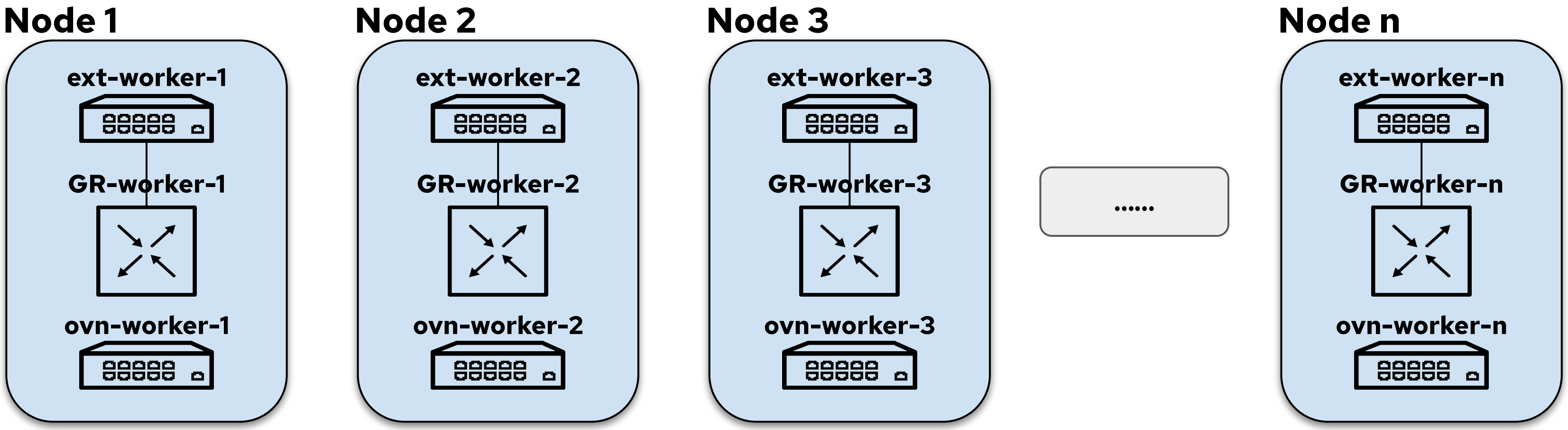
Horizontal Scaling With OVN Component Templates

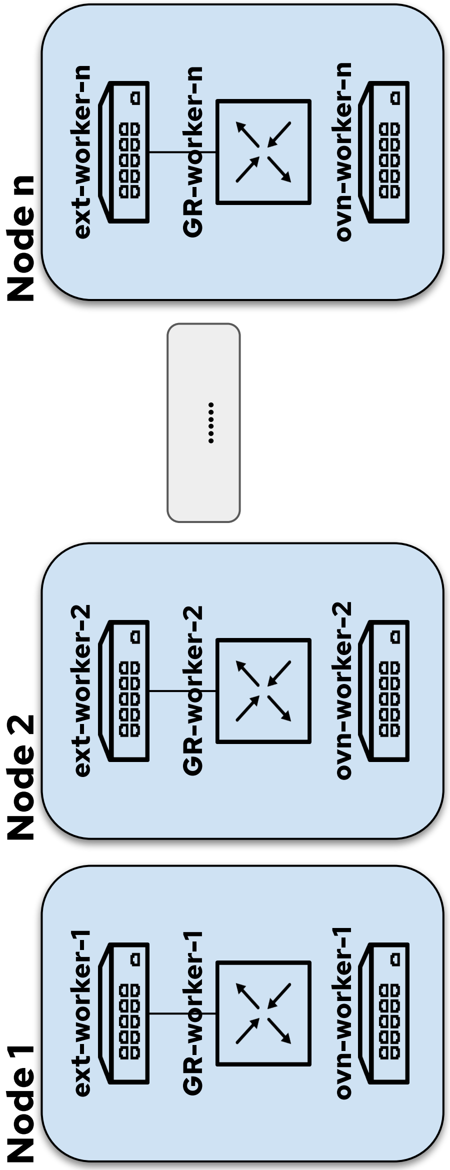
Dumitru Ceara

Red Hat

Why “horizontal” scaling?

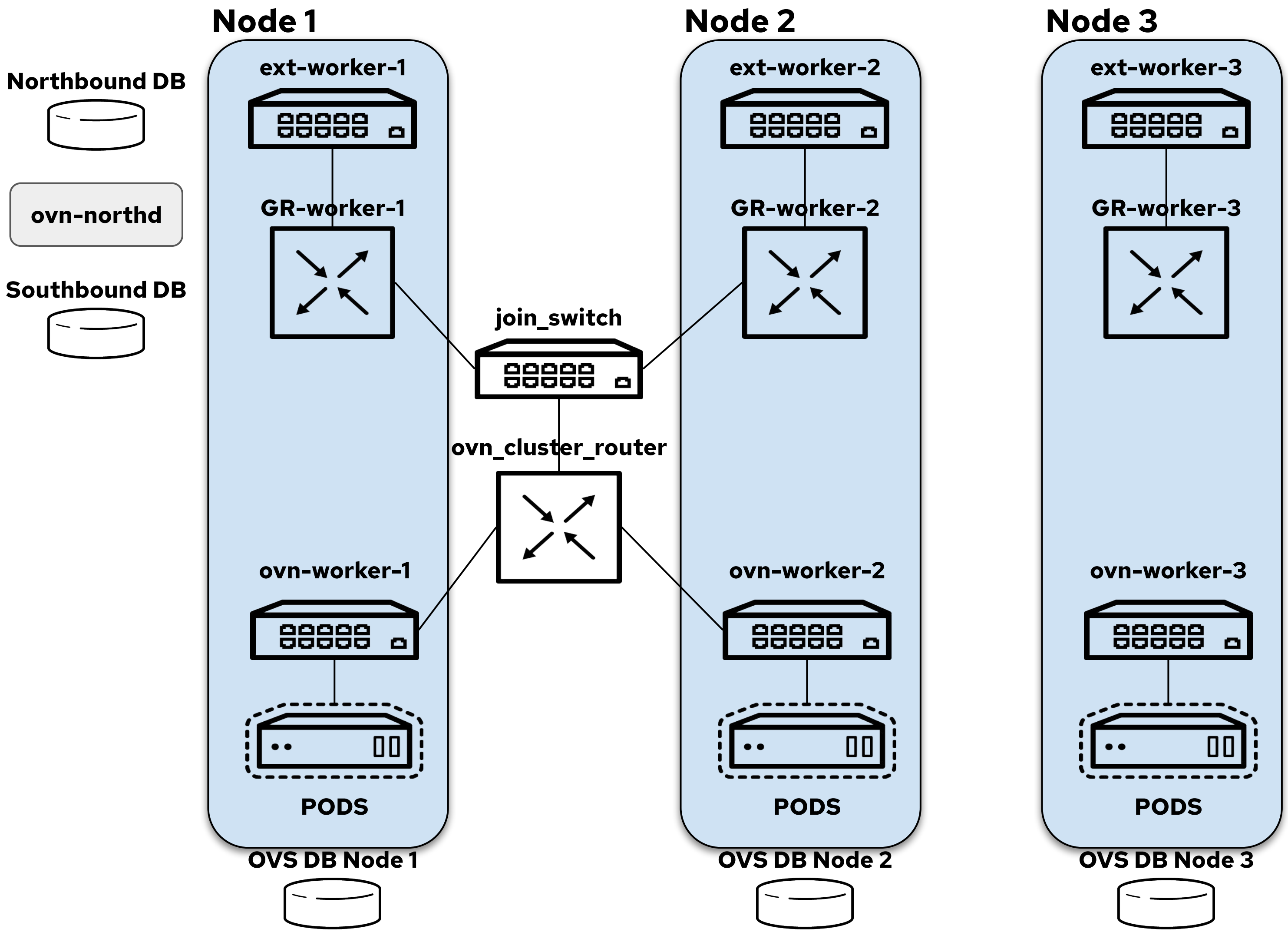


It could have been “vertical” too...



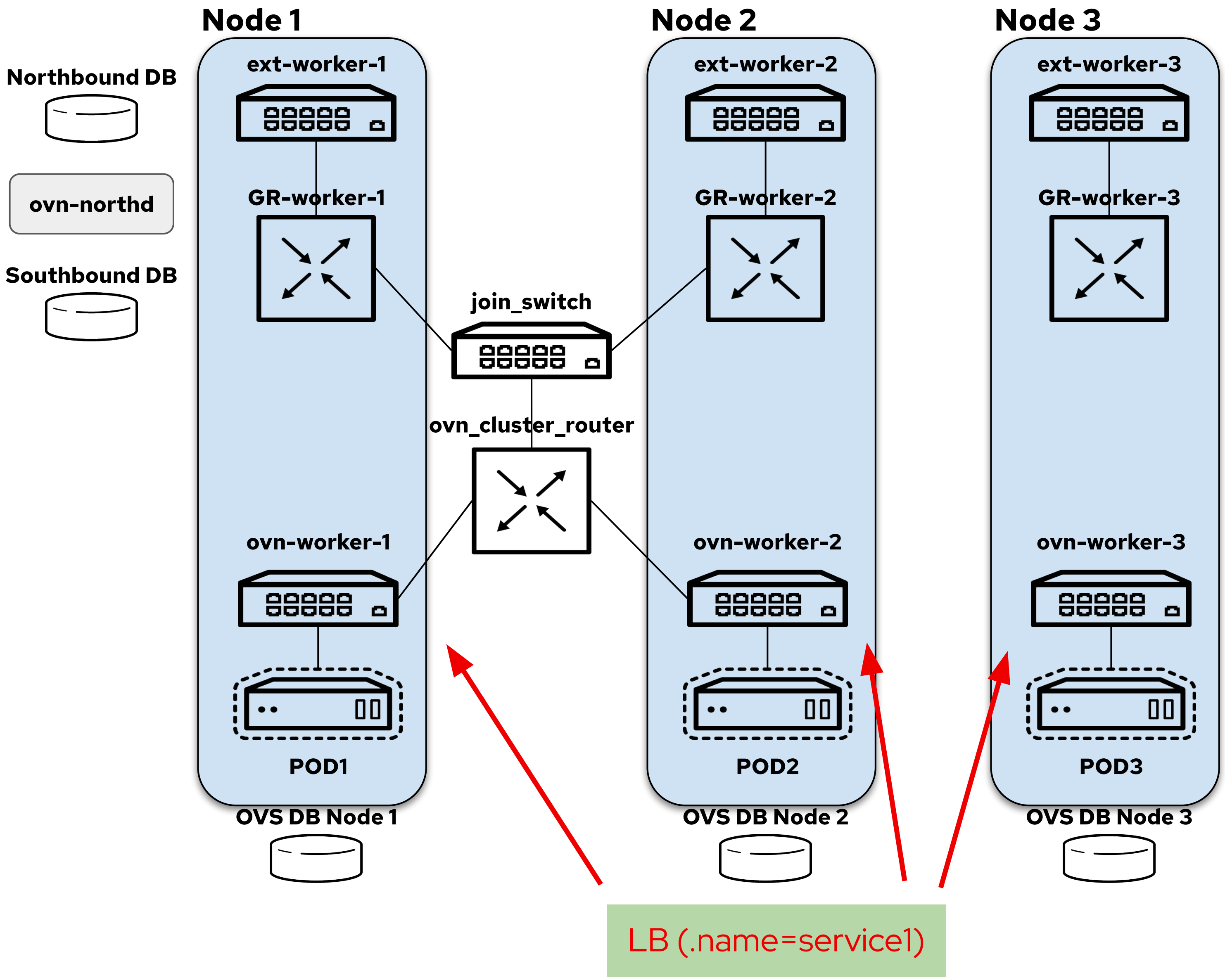
... but it wouldn’t fit that well :-)

OVN-K8S network topology



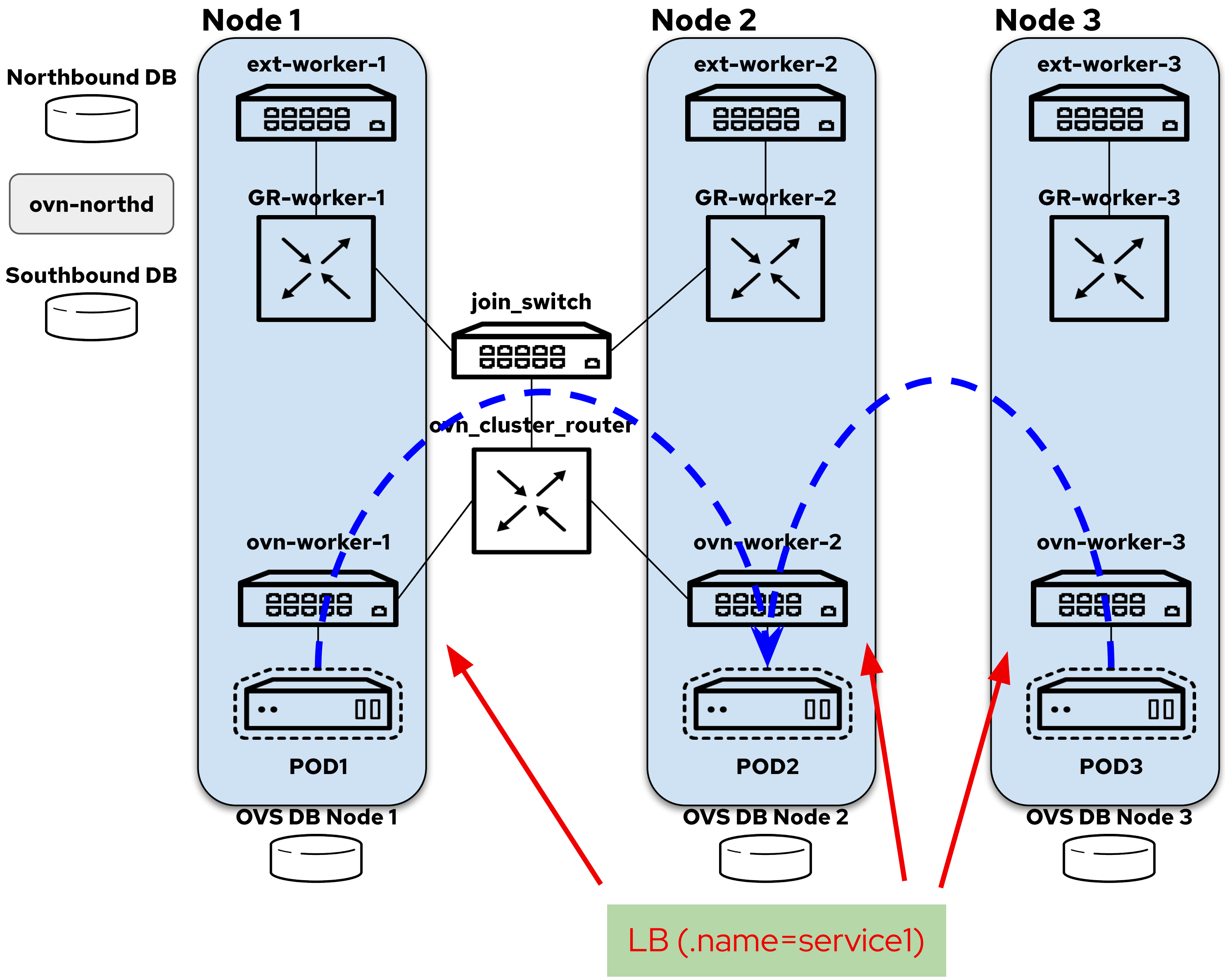
- Distributed:
  - ovn_cluster_router
  - join_switch
- Per node:
  - ovn-worker switch
  - GR-worker router
  - ext-worker switch

OVN-K8S services - ClusterIP



“Load balance traffic destined to a cluster-internal IP (and port) to a set of backends (pods).”

- Single OVN Load Balancer applied to all ovn-worker switches
- 1:1 mapping between k8s service object and OVN load balancer object



Service1 =

(.vip=42.42.42.42:4242,

.backends=[pod2IP, pod3IP], tcp)

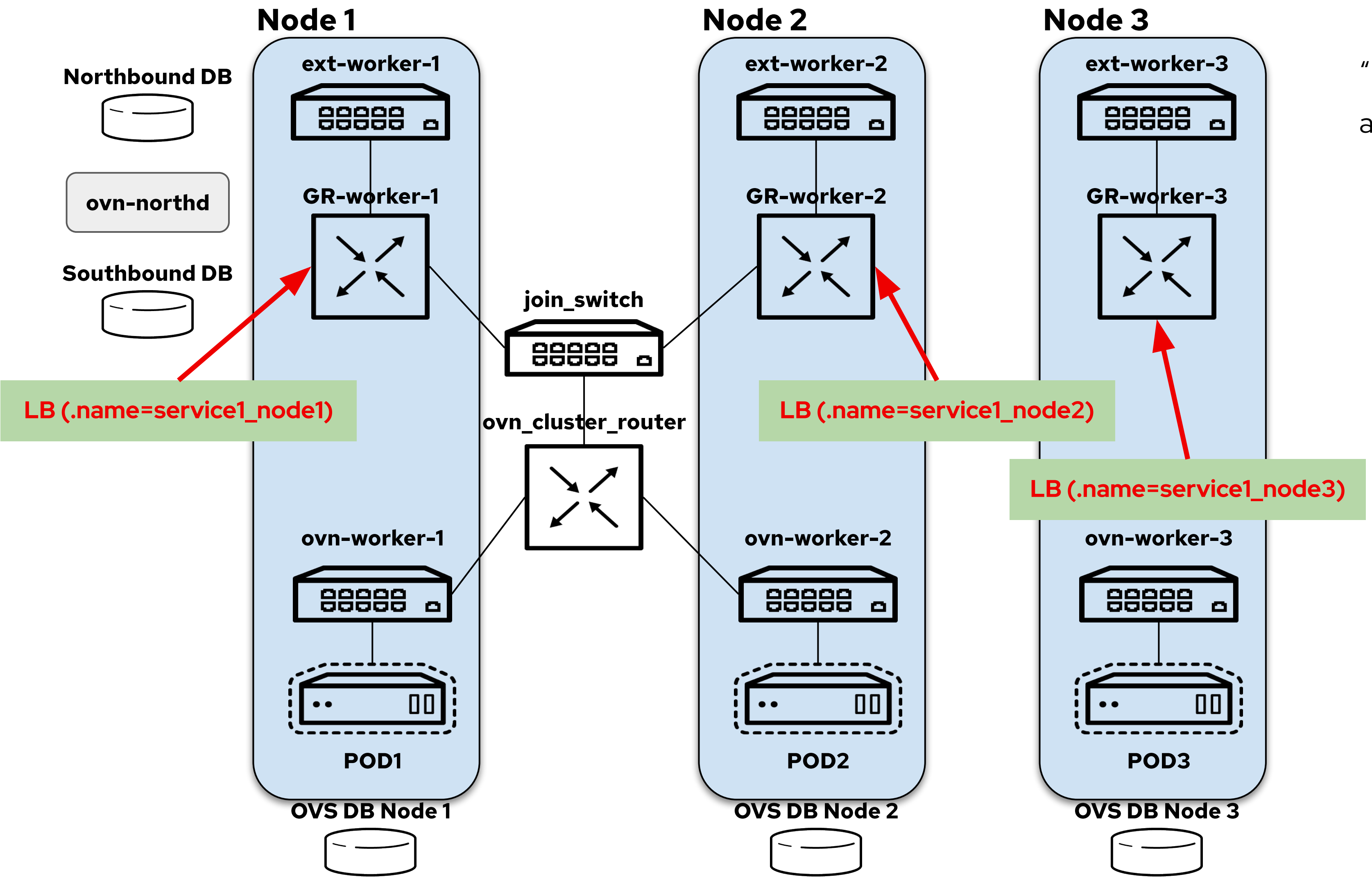
- pod1IP -> 42.42.42.42:4242
  - DNAT on ovn-worker-1 to either pod2IP or pod3IP
- pod2IP -> 42.42.42.42:4242
  - DNAT on ovn-worker-2 to either pod2IP or pod3IP
- pod3IP -> 42.42.42.42:4242
  - DNAT on ovn-worker-3 to either pod2IP or pod3IP

Scales linearly:

- S services -> S load balancers
- O(S) logical flows

Red Hat

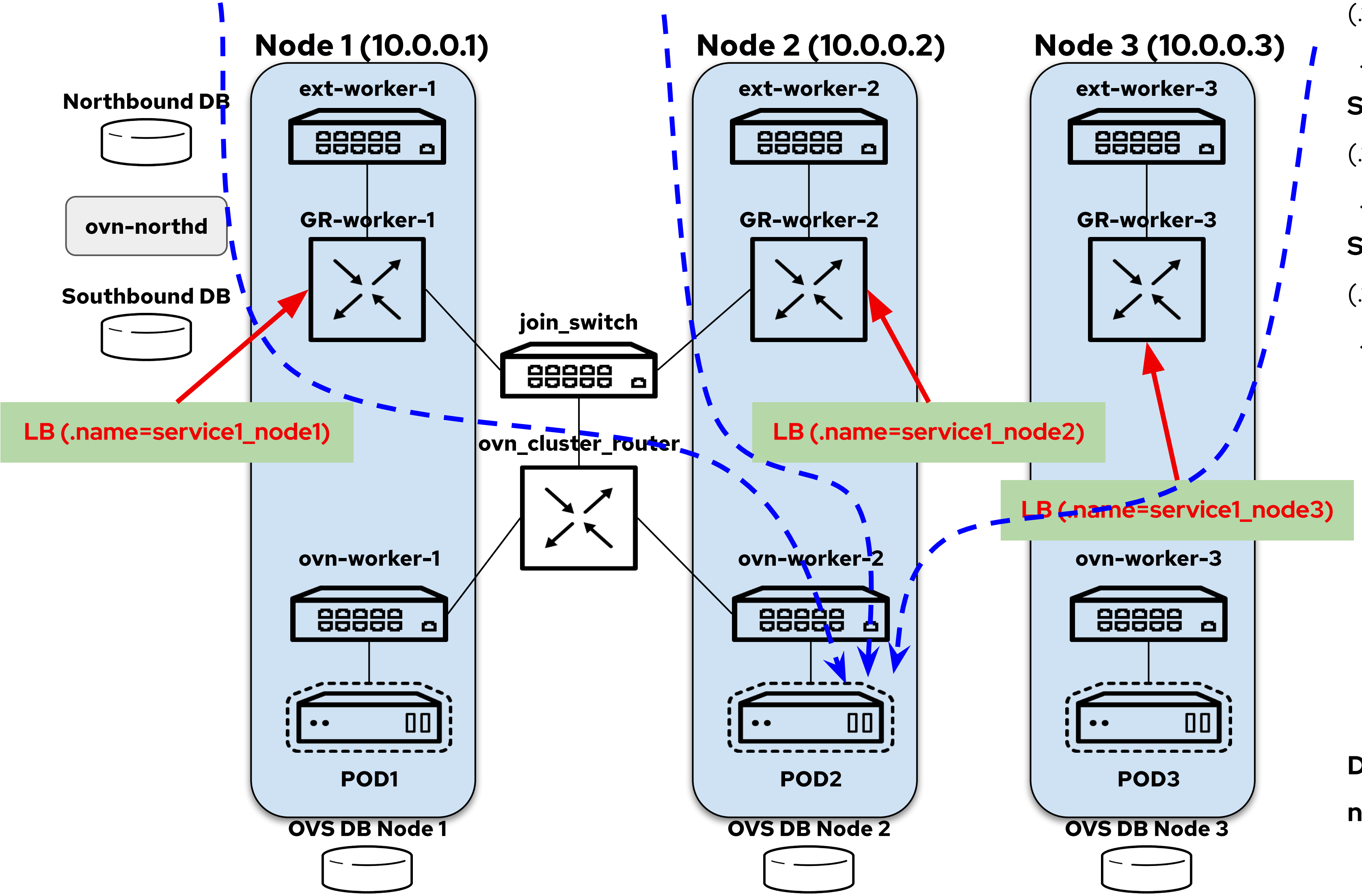
OVN-K8S services - NodePort



"Exposes the Service on each Node's IP at a static port."

- Unique OVN Load Balancers applied to all N GR-worker routers
- 1:N mapping between k8s service object and OVN load balancer object

OVN-K8S services - NodePort example



Service1_node1 =

(.vip=10.0.0.1:4242,

.backends=[pod2IP, pod3IP], tcp)

Service1_node2 =

(.vip=10.0.0.2:4242,

.backends=[pod2IP, pod3IP], tcp)

Service1_node3 =

(.vip=10.0.0.3:4242,

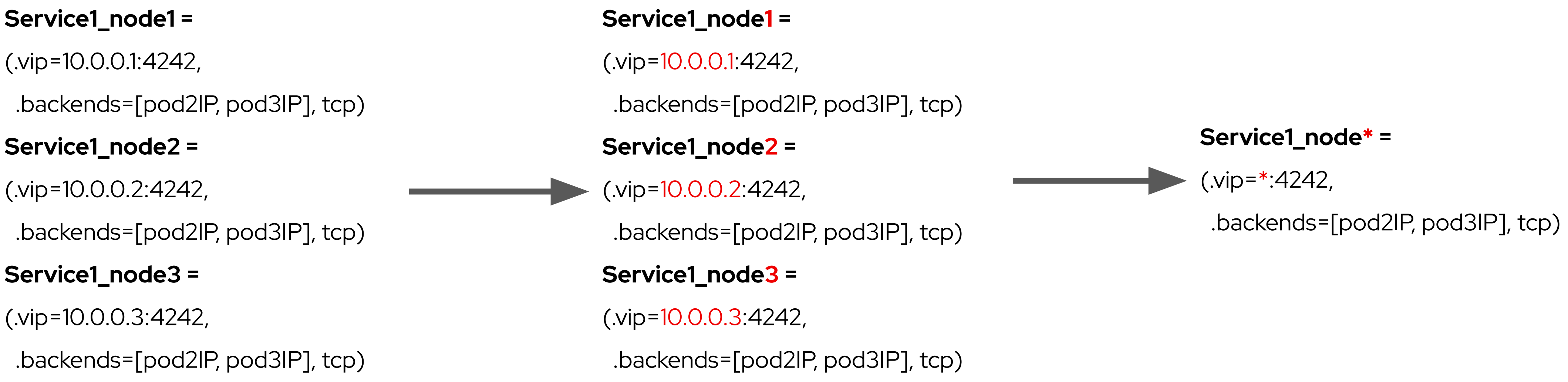
.backends=[pod2IP, pod3IP], tcp)

- -> 10.0.0.1:4242
  - DNAT on GR-worker-1 to either pod2IP or pod3IP
- -> 10.0.0.2:4242
  - DNAT on ovn-worker-2
- -> 10.0.0.3:4242
  - DNAT on ovn-worker-3

Does not scale nicely (S services, N nodes):

- S x N load balancers
- O(S x N) logical flows

Focus on the load balancers..



Service1_node1 =
(.vip=10.0.0.1:4242,
 .backends=[pod2IP, pod3IP], tcp)
Service1_node2 =
(.vip=10.0.0.2:4242,
 .backends=[pod2IP, pod3IP], tcp)
Service1_node3 =
(.vip=10.0.0.3:4242,
 .backends=[pod2IP, pod3IP], tcp)

Service1_node1 =
(.vip=10.0.0.1:4242,
 .backends=[pod2IP, pod3IP], tcp)
Service1_node2 =
(.vip=10.0.0.2:4242,
 .backends=[pod2IP, pod3IP], tcp)
Service1_node3 =
(.vip=10.0.0.3:4242,
 .backends=[pod2IP, pod3IP], tcp)

Service1_node* =
(.vip=*:4242,
 .backends=[pod2IP, pod3IP], tcp)

Almost identical load balancers…

If we highlight the different bits…

And then mask them out…

We get…

OVN Component Templates

→ **Service1 =(.vip=^VIP_VAR:4242,.backends=[pod2IP, pod3IP], tcp)**

- A component template has a name
  - template names have similar restrictions to port group and address set names
  - when referring to a template name use the ^ character as prefix
- A component template has (at most) one value on any given chassis in the cluster
  - defined through a new table in the OVN_Northbound database

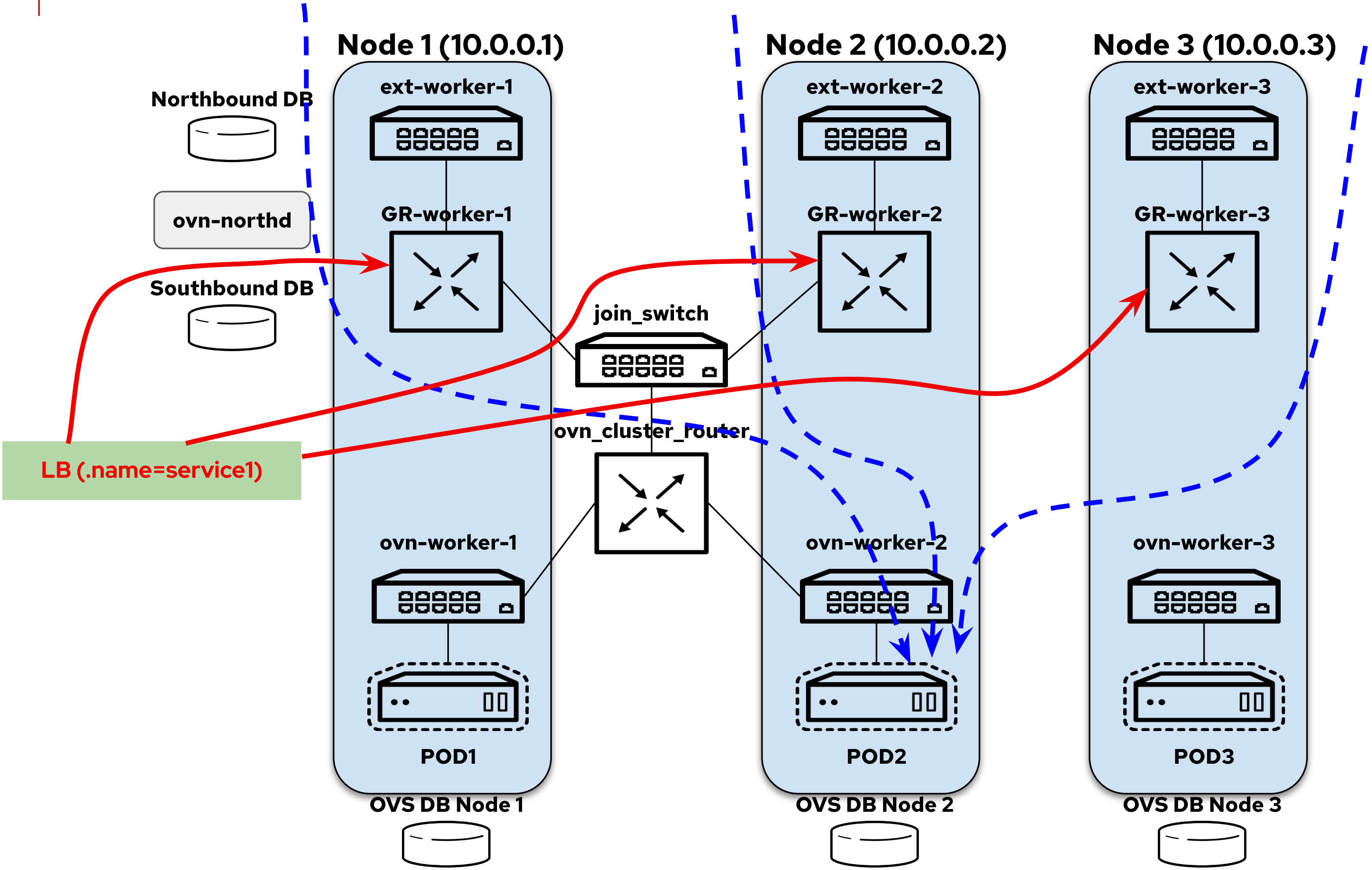
```
"Chassis_Template_Var": {
    "columns": {
        "chassis": {"type": "string"},
        "variables": {
            "type": {"key": "string", "value":
"string",
                     "min": 0, "max": "unlimited"}}},
    "indexes": [["chassis"]],
    "isRoot": true
}
```

```
_uuid     : dd0a3f7c-78ee-41c0-b9e9-88f9b3ef733b
chassis   : node1
variables : {VIP_VAR="10.0.0.1"}

_uuid     : 3a8e3626-873c-47da-b2c0-47ea8fbb795c
chassis   : node2
variables : {VIP_VAR="10.0.0.2"}

_uuid     : 08e8ef1e-0b6b-4a5c-9951-b9ae5e4b36ae
chassis   : node3
variables : {VIP_VAR="10.0.0.3"}
```

OVN-K8S services - NodePort example with templates



Service1 =

(.template=true, .vip=^VIP_VAR:4242,
.backends=[pod2IP:4242, pod3IP:4242], tcp)

Service2 =

(.template=true, .vip=^VIP_VAR:8484,
.backends=[pod2IP:8484, pod3IP:8484], tcp)

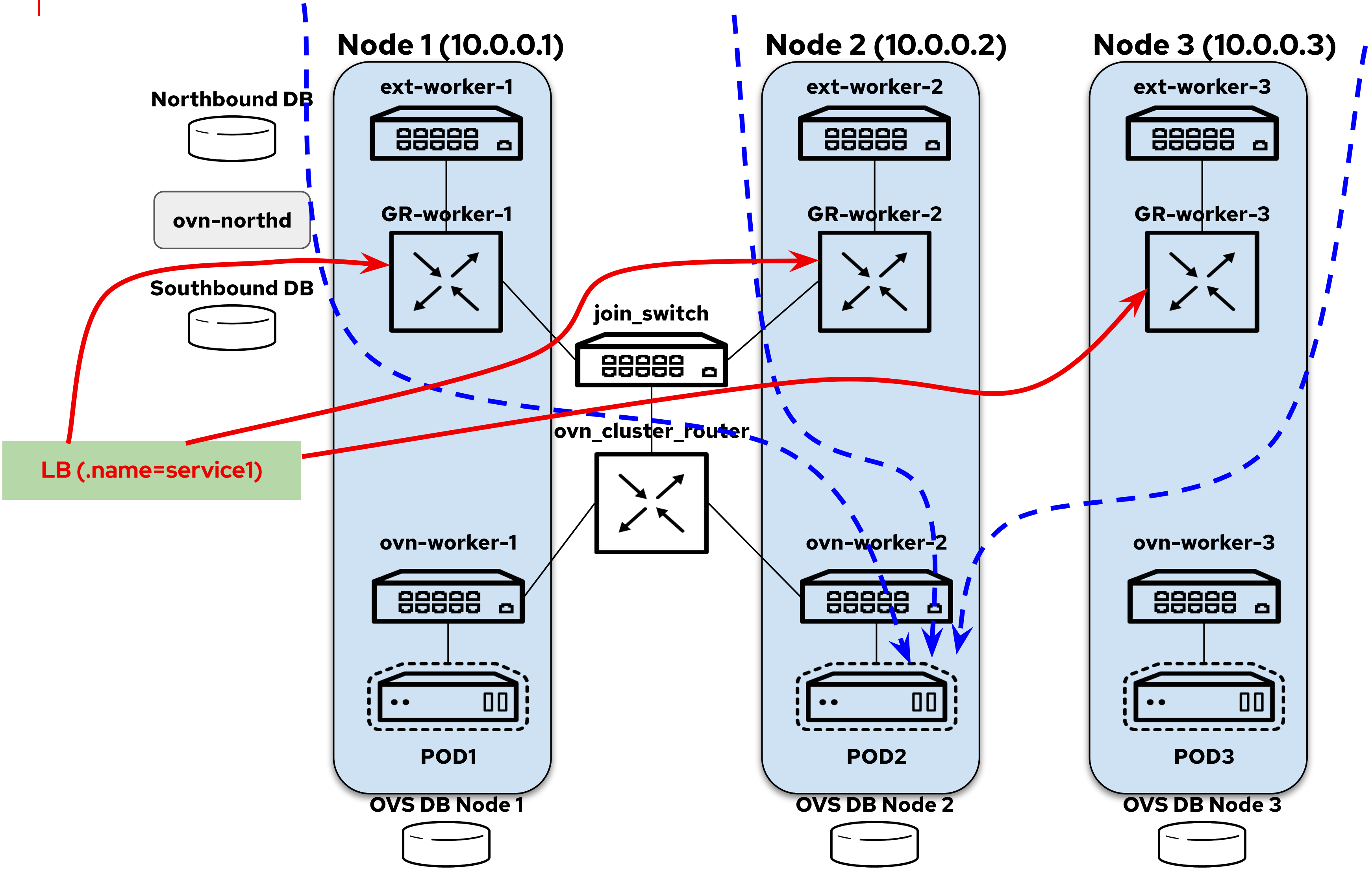
Chassis_Template_Var

(.chassis=node1, .variables=[(VIP_VAR: 10.0.0.1)])
(.chassis=node2, .variables=[(VIP_VAR: 10.0.0.2)])
(.chassis=node1, .variables=[(VIP_VAR: 10.0.0.2)])

Scales linearly (S services, N nodes):

- S load balancers (templated)
- O(S) logical flows (templated)
- N Chassis_Template_Var mappings

OVN-K8S services - NodePort example with templates and unique backends



Service1 =

(.template=true, .vip=^VIP_VAR:4242,

.backends=^BACKEND1, tcp)

Service2 =

(.template=true, .vip=^VIP_VAR:4242,

.backends=^BACKEND2, tcp)

Chassis_Template_Var

(.chassis=node1, .variables=[(VIP_VAR: 10.0.0.1),

(BACKEND1:"POD1IP:4242"), (BACKEND2:...)])

(.chassis=node2, .variables=[(VIP_VAR: 10.0.0.2),

(BACKEND1:"POD2IP:4242"), (BACKEND2:...)])

(.chassis=node3, .variables=[(VIP_VAR: 10.0.0.3),

(BACKEND1:"POD3IP:4242"), (BACKEND2:...)])

Worst case (S services, N nodes, unique backends per node):

- S load balancers (templated)
- O(S) logical flows (templated)
- O(S x N) Chassis_Template_Var mappings

Benchmark results

Simulate an OVN-K8S deployment with N nodes, S NodePort services, unique backend sets: 5 unique backends per service per node.

Template	N	S	NB (size on-disk/RSS)	SB (size on-disk/RSS)	ovn-northd loop time	ovn-controller
NO	60	1000	Size: 25MB RSS: **116MB**	Size: **118MB** RSS: **589MB**	**2.70s**	RSS: **463MB** Recompute: 0.52s
YES	60	1000	Size: 6MB RSS: **25MB**	Size: 8MB RSS: **46MB**	**0.07s**	RSS: **44MB** Recompute: 0.20s
NO	120	2000	Size: 67MB RSS: **865MB**	Size: **471MB** RSS: **9000MB**	**15.60s**	RSS: **1016MB** Recompute: 0.40s
YES	120	2000	Size: 23MB RSS: **96MB**	Size: **28MB** RSS: **225MB**	**0.22s**	RSS: **83MB** Recompute: 0.40s
YES	120	10000	Size: 118MB RSS: **440MB**	Size: **136MB** RSS: **668MB**	**0.72s**	RSS: **311MB** Recompute: 1.77s
YES	250	10000	Size: 244MB RSS: **870MB**	Size: **263MB** RSS: **1502MB**	**1.26s**	RSS: **318MB** Recompute: 1.87s

- Templates allow scaling to **x2 nodes** and **x5 services** compared to the current (non-template) deployment while using less resources
- For the **N=120 S=2000** case:
  - NB size reduced by **~65%**, NB RSS reduced by **~90%**, SB size reduced by **~95%**, SB RSS reduced by **~98%**
  - ovn-northd loop time reduced by **~98%**, ovn-controller RSS reduced by **~92%**

OVN Component Templates - Conclusions

- Significantly improve scalability when resources are distributed uniformly
- Supported for any type of OVN match/actions and Load Balancers
- Require work on the CMS side to define the templates in a way that translates optimally to virtual network resources
- Targeting acceptance in OVN v22.12.0

V1:

https://mail.openvswitch.org/pipermail/ovs-dev/2022-September/398110.html

https://patchwork.ozlabs.org/project/ovn/list/?series=320941&state=*

Thank you!