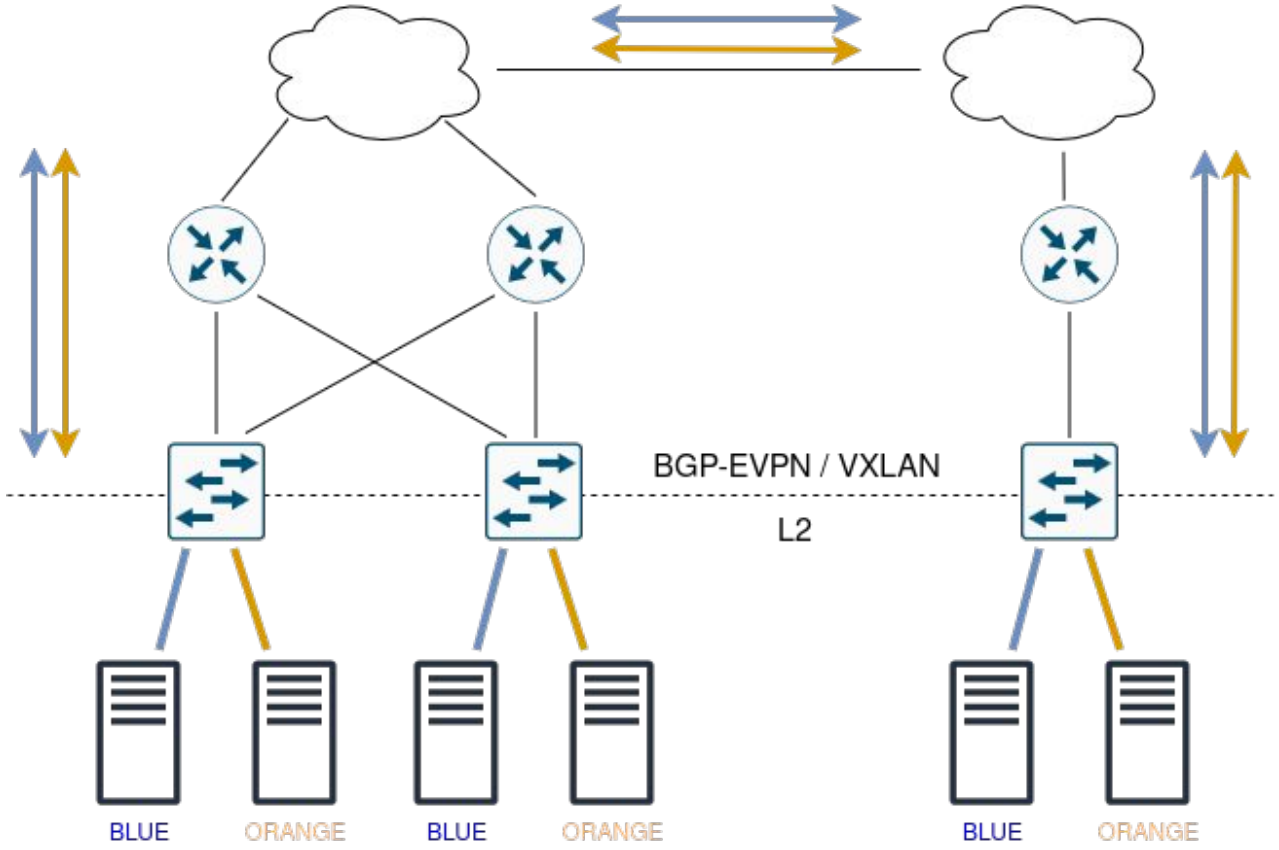
A native way of integrating OVN into the fabric through BGP-EVPN

Aleš Musil, Dumitru Ceară, 2025

Red Hat

Introduction to BGP-EVPN with OVN

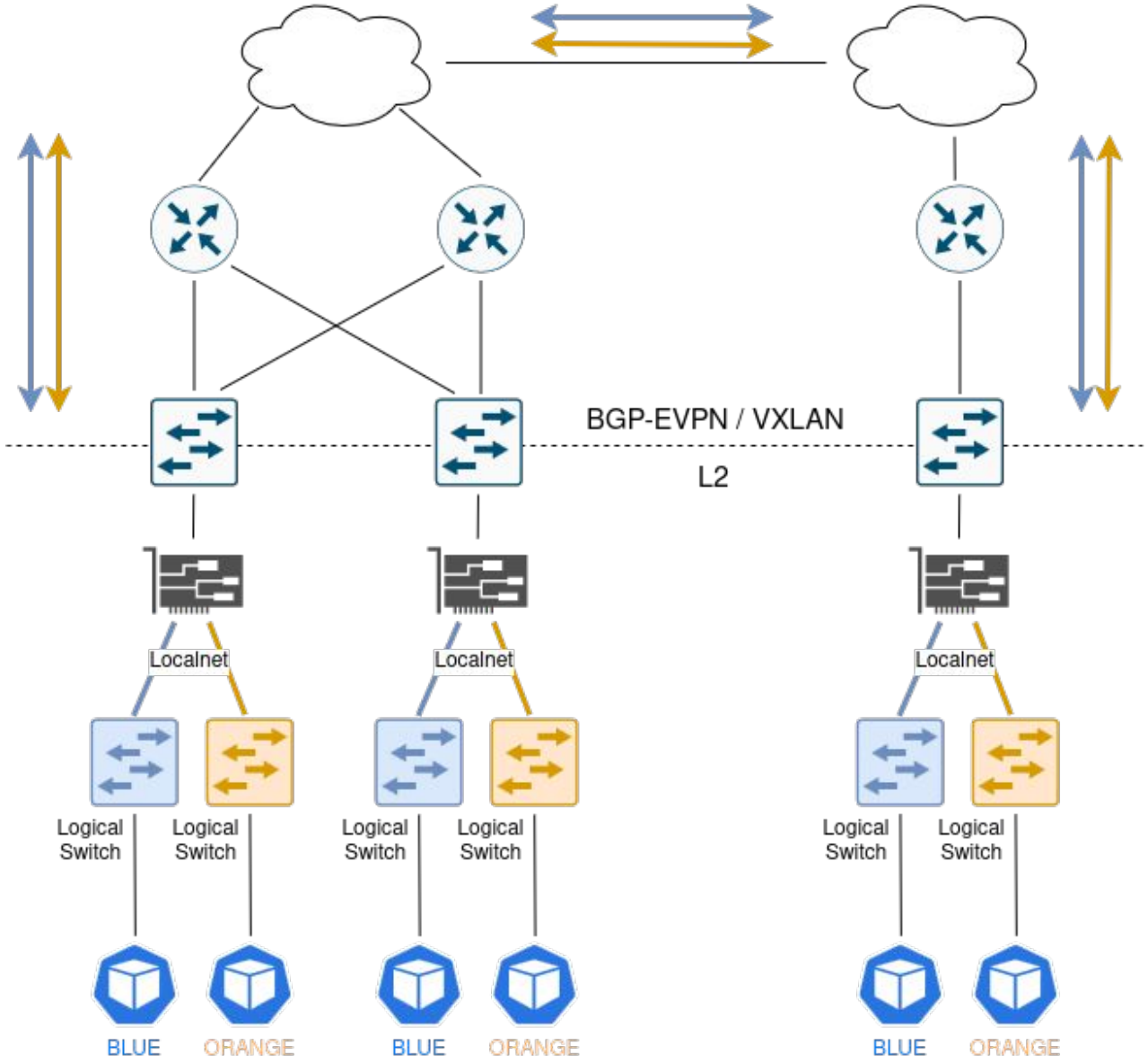
BGP-EVPN (extremely brief) summary



- (virtual) L2 broadcast domain stretched across IP underlays
- "seamless" moving of workloads across different geos
- BGP control plane distributes MAC/IP information (per VRF)
  - type-2 EVPN routes
  - reduces need for flooding broadcast (BUM) L2 frames to all remote sites of the L2 domain
- BGP control plane distributes IP route information (per VRF)
  - type-5 EVPN routes
- VXLAN tunnels used in the fabric to separate the different (virtual) L2 broadcast domains

Red Hat

OVN Localnet and reaching into the fabric



- They look similar but:
  - orange and blue **can't have** overlapping IP subnets!

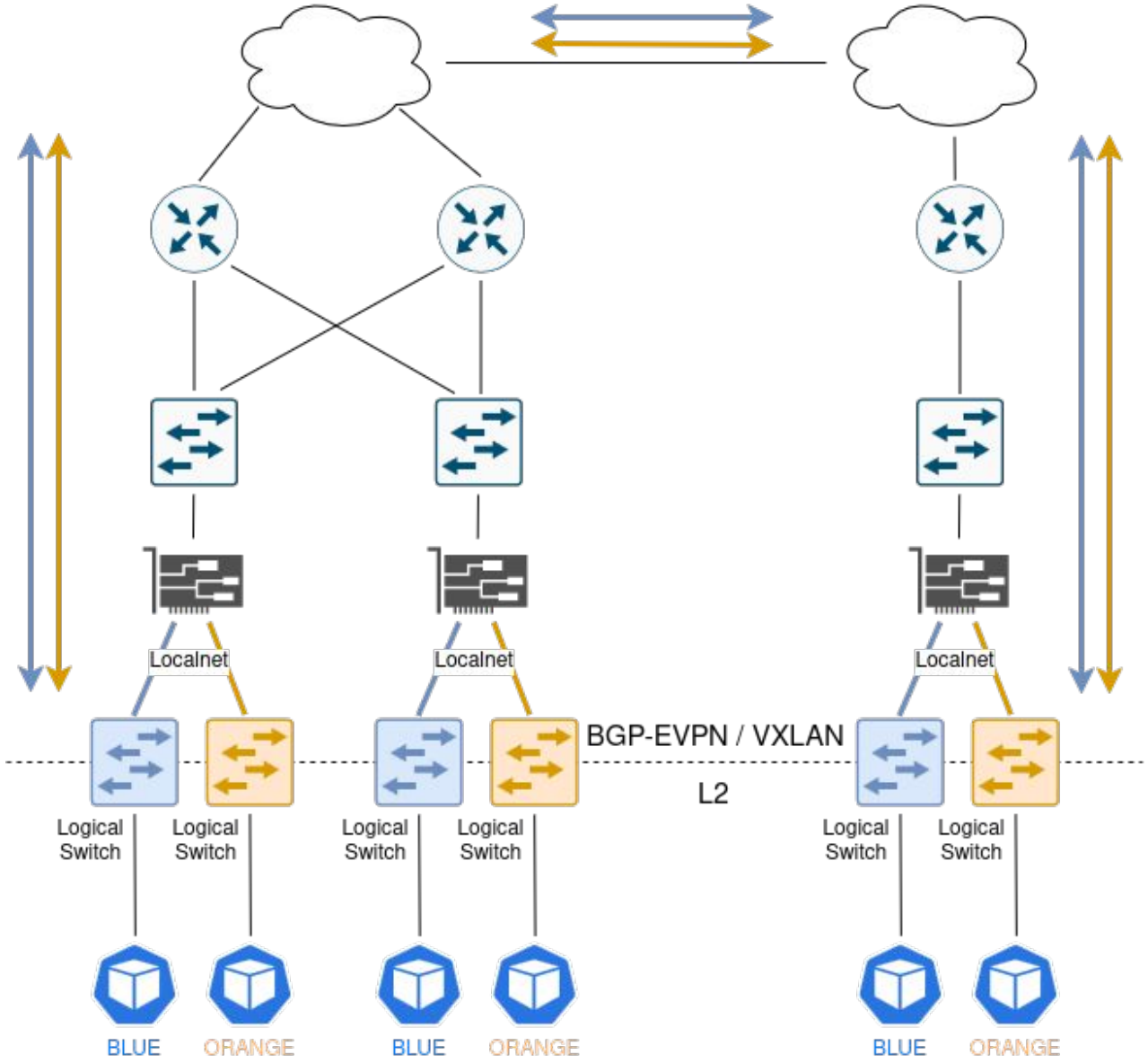
    (maybe with complex additional config on the ToRs)
  - **can't** distribute logical switches (blue/orange) across

    chassis unless there's L2 connectivity between the

    ToRs (not always the case)
  - **No encapsulation needed for E-W traffic**

Red Hat

OVN-native BGP-EVPN integration - WHY?



- orange and blue networks are now completely independent
- w/ OVN-native support all traffic processing happens in OVS
  - hwol friendly
  - vxlan tunnels extended into ovn
- datapath agnostic:
  - OVS-kernel / OVS-userspace (DPDK) / OVS-?
- BGP-EVPN control plane state
  - maintained in the kernel (e.g., by FRR)
  - monitored by OVN through NetLink: VTEP learning, MAC(+IP) learning, route learning
  - updated by OVN through NetLink: workload MAC(+IP) advertisements, route advertisements

OVN implementation

Configuration

Just set the **other_config:dynamic-routing-vni=$vni**

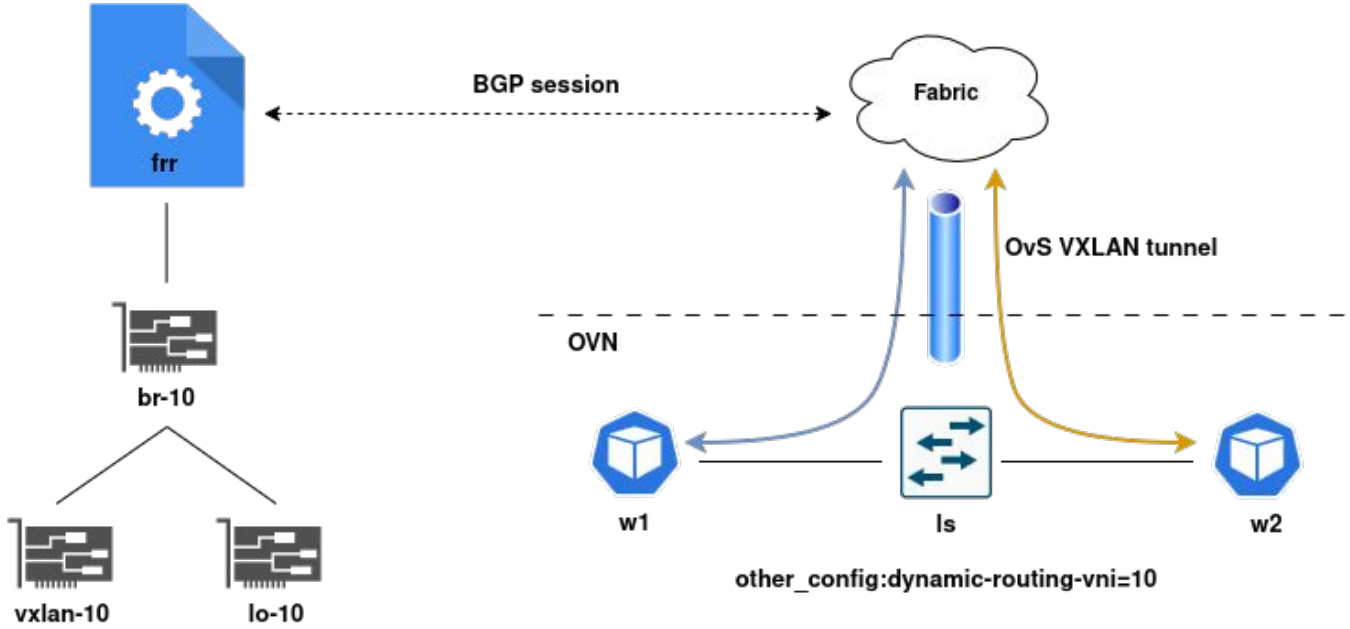
and

external-ids:ovn-evpn-local-ip=$local_ip

It’s just that easy, or is it?

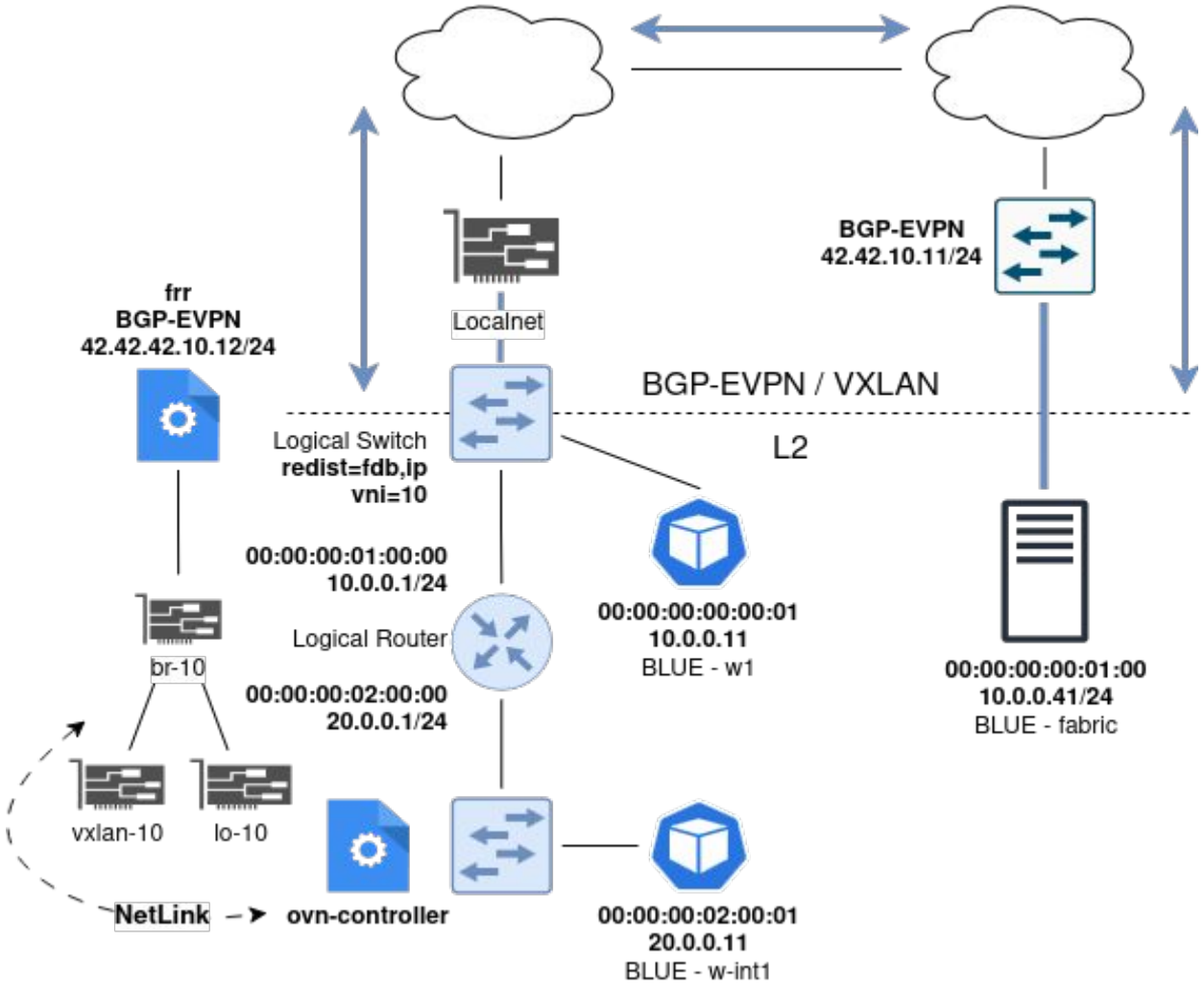
Configuration assumptions

- OVN expects the following interface configuration to be done by the CMS:
  - br-$vni - Bridge interface used to learn and advertise static IP+MAC.
  - lo-$vni - Loopback interface used to advertise static FDBs.
  - vxlan-$vni - VXLAN interface used to learn the remote VTEPs and static FDBs.
  - Currently at most one local IP is allowed for EVPN tunnels.
  - **The vxlan-$vni should have a different source port then the OvS tunnel uses to avoid clash.**
- All of the above interfaces should be in the default network namespace.
- The lo-$vni and vxlan-$vni should have br-$vni as a master.



Red Hat

Traffic flow - L2 EVPN



- Scenario used in the multinode L2 EVPN test
- One OVN cluster (left) with
  - provider logical switch configured to extend its L2 domain via EVPN using VXLAN VNI 10
  - workloads
    - directly attached to the provider logical switch
    - behind an OVN logical router
- One non-OVN part of the L2 domain (right)
  - hosts connected to a BGP-EVPN aware switch (non-OVN)
- Hosts on the right must be able to reach OVN workloads (at L2 level)

Red Hat

Remote VTEPs

- FRR learned VTEPs (on OVN node)
  $ show bgp l2vpn evpn
  Route Distinguisher: **42.42.10.11**:2
  *> [3]:[0]:[32]:[**42.42.10.11**] **42.42.10.11**
  *$ bridge fdb show dev **vxlan-10***
  00:00:00:00:00:00 dst **42.42.10.11** self permanent

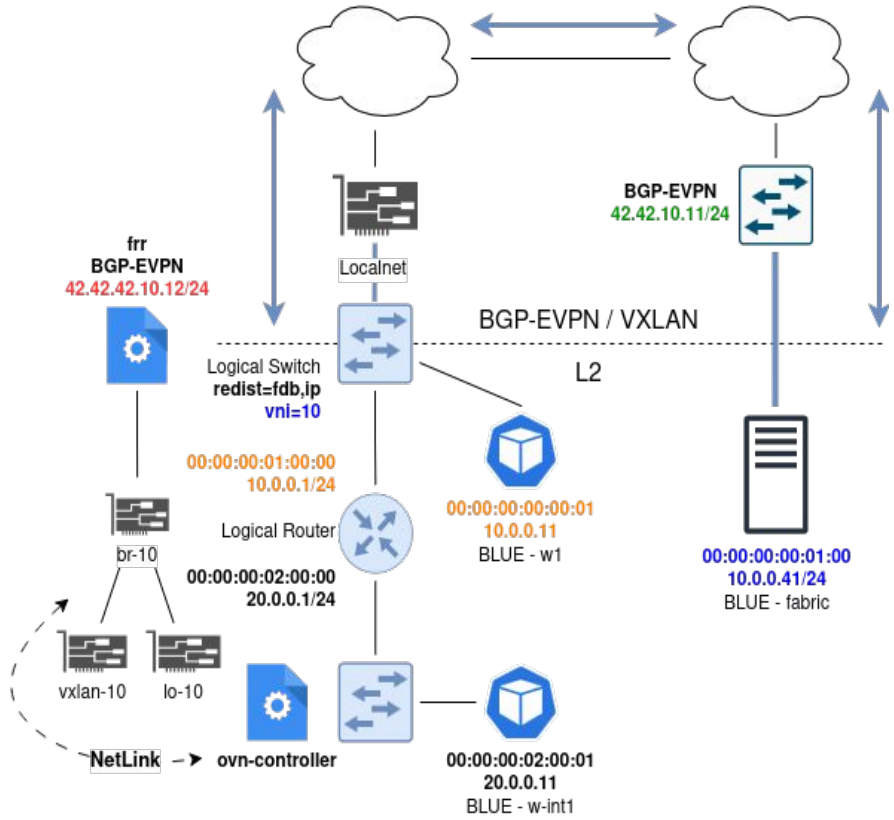
- OVN learned VTEPs
  $ ovn-appctl evpn/remote-vtep-list
  IP: **42.42.10.11**, port: 4789, vni: **10**

- OVN vtep bindings
  $ ovn-appctl evpn/vtep-binding-list
  UUID: **8a63f813**, Remote IP: **42.42.10.11**, vni: **10**,
  binding_key: **0x80000001,** dp_key:**1**

- OVN vtep mcast groups
  $ ovn-appctl evpn/vtep-multicast-group-list
  Remote IPs: **42.42.10.11**, vni: **10**

- OVN managed VXLAN OpenFlow rules:
  $ ovs-ofctl dump-flows br-int table=0,cookie=0x8a63f813
  cookie=0x**8a63f813**, tun_id=**10**,tun_src=**42.42.10.11**,tun_dst=**42.42.10.12**
  actions=load:0x1->metadata,load:**0x80000001**->reg14,...

  $ ovs-ofctl dump-flows br-int table=44,cookie=0x8a63f813
  cookie=0x**8a63f813**,reg15=**0x80000001**,metadata=**0x1**
  actions=load:**42.42.10.12**->tun_src,load:**42.42.10.11**->tun_dst,load:**10**->tun_id,output:<vxlan-tunnel>



Red Hat

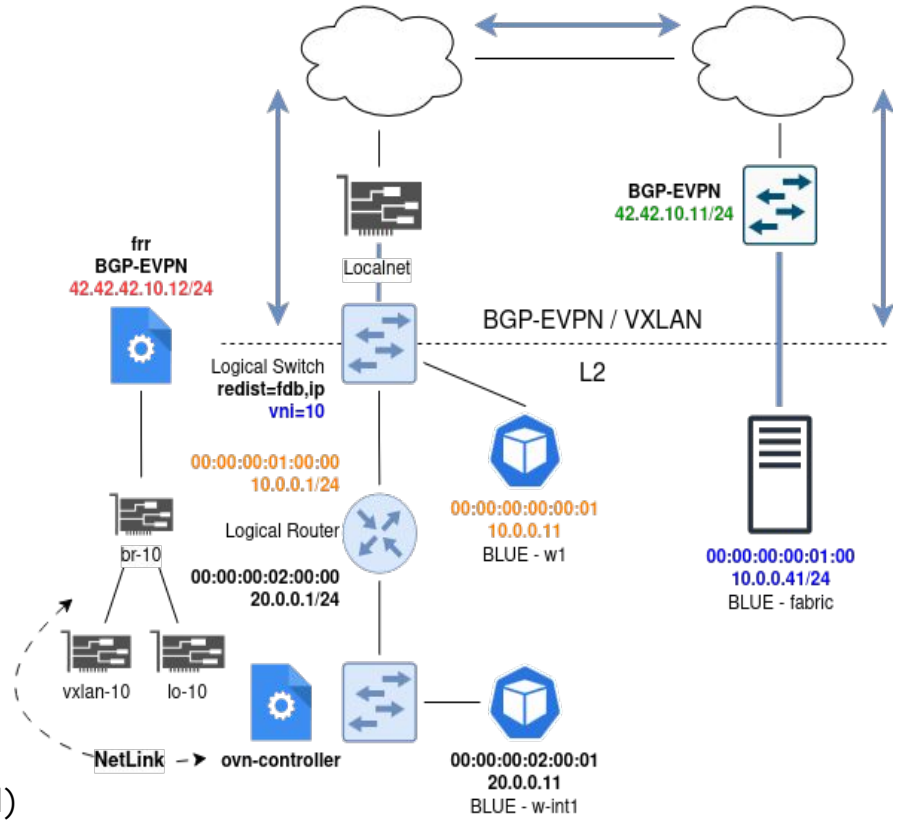
Remote FDBs

- FRR learned external FDBs
  $ show bgp l2vpn evpn
  Route Distinguisher: **42.42.10.12**:2
  *> [2]:[0]:[48]:[**00:00:00:00:01:00**]:[32]:[**10.0.0.41**]
  $ *bridge fdb show dev **br-10***
  00:00:00:00:01:00 dst **42.42.10.11** self extern_learn

- OVN learned external FDBs
  $ ovn-appctl evpn/vtep-fdb-list
  UUID: **77a3e57c**, MAC: **00:00:00:00:01:00**,
  vni: 10, binding_key: **0x80000001**, dp_key: 1

- OVN learned external and dynamic FDBs
  $ ovs-ofctl dump-flows br-int table=88
  cookie=**0x8a63f813**,reg14=0x80000001,metadata=0x1
  actions=learn(table=87,**hard_timeout**=30,priority=150,delete_learned,
  cookie=**0x8a63f813**,metadata,eth_dst=eth_src,load:reg14->reg1)

  $ ovs-ofctl dump-flows br-int table=87
  cookie=**0x77a3e57c**,priority=150,metadata=0x1,
  dl_dst=**00:00:00:00:01:00** actions=load:**0x80000001**->reg1
  cookie=**0xcff7aec3**, table=87, **hard_timeout**=30, priority=150,metadata=0x1,dl_dst=**00:00:10:00:00:81**
  actions=load:**0x80000001**->reg1



Red Hat

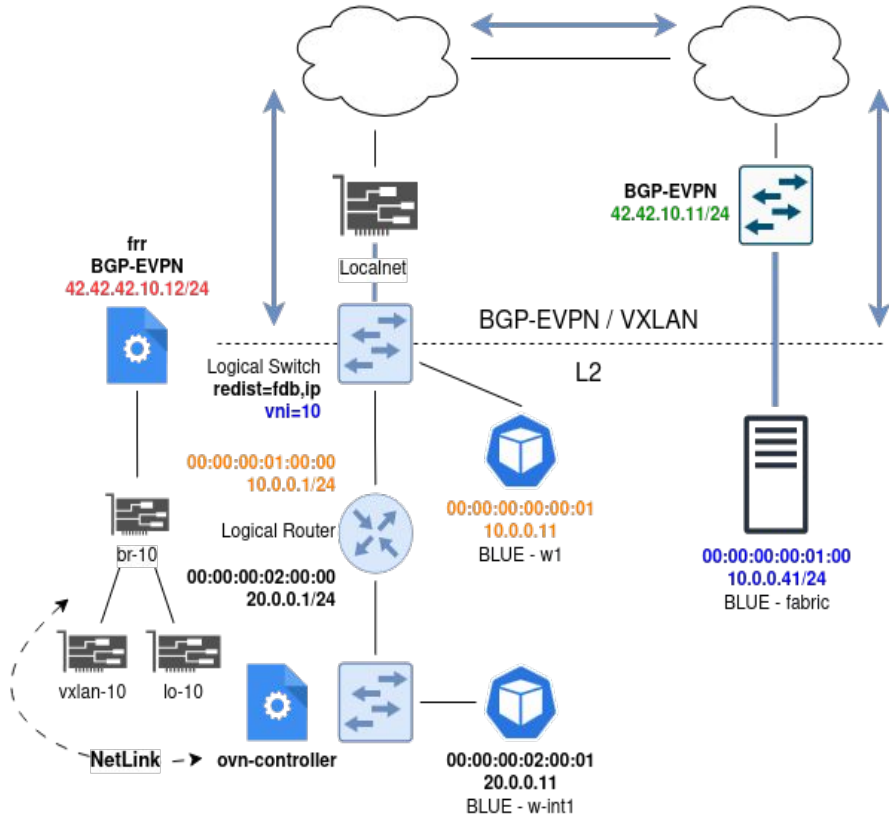
Remote IPs

- FRR learned external IPs
  $ show bgp l2vpn evpn
  Route Distinguisher: **42.42.10.12**:2
  *> [2]:[0]:[48]:[**00:00:00:00:01:00**]:[32]:[**10.0.0.41**]
  *$ ip neigh show dev **br-10***
  10.0.0.41 lladdr **00:00:00:00:01:00** **extern_learn** NOARP proto zebra

- OVN learned external IPs
  $ ovn-appctl evpn/vtep-arp-list
  UUID: **5992632a**, VNI: 10, MAC: **00:00:00:00:01:00**, IP: **10.0.0.41**, dp_key: **1**

- OVN learned IPs (on logical router) - ARP suppression
  *$ ovs-ofctl dump-flows br-int cookie=**0x5992632a***
  cookie=**0x5992632a**, table=66, reg0=**10.0.0.41**,reg15=0x1,metadata=**0x3**
  actions=mod_dl_dst:**00:00:00:00:01:00**,load:0x1->reg10[6]

  cookie=**0x5992632a**, table=67, arp,reg0=**10.0.0.41**,
  reg14=0x1,metadata=**0x3**,dl_src=**00:00:00:00:01:00**
  actions=load:0x1->reg10[6]



Red Hat

Advertisement of FDBs

- OVN advertised FDBs

  $ ovn-nbctl show ls

  switch 92cd21cf (ls)

  port w1 addresses: ["**00:00:00:00:00:01 10.0.0.11**"]

  $ ovn-nbctl show lr

  router 22cf207d (lr)

  port lr-ls mac: "**00:00:00:01:00:00**" networks: ["**10.0.0.1/24**"]

  *$ bridge fdb show dev **lo-10***

  00:00:00:01:00:00 master br-10 static

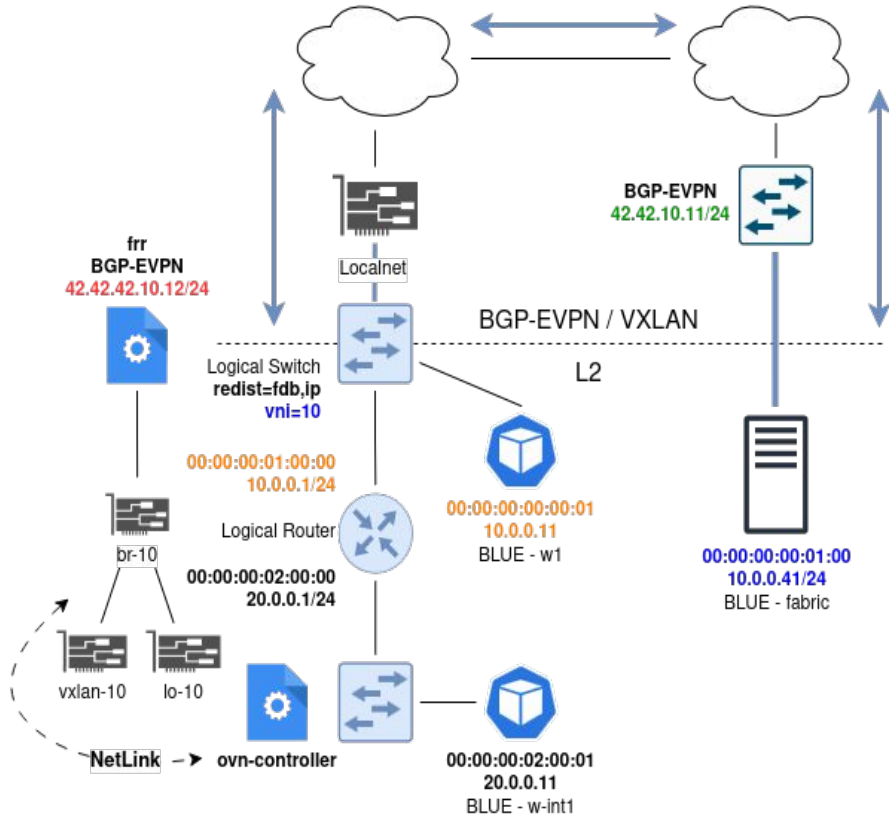
  00:00:00:00:00:01 master br-10 static

- Fabric FRR learned FDBs

  $ show evpn mac vni all

  VNI 10 #MACs (local and remote) 2

MAC	Type	Intf/Remote	ES/VTEP
00:00:00:00:00:01	remote		**42.42.10.12**
00:00:00:01:00:00	remote		**42.42.10.12**
00:00:00:00:01:00	local		



Red Hat

Advertisement of IPs

- OVN advertised IPs

  $ ovn-nbctl show ls

  switch 92cd21cf (ls)

  port w1 addresses: ["**00:00:00:00:00:01 10.0.0.11**"]

  $ ovn-nbctl show lr

  router 22cf207d (lr)

  port lr-ls mac: "**00:00:00:01:00:00**" networks: ["**10.0.0.1/24**"]

  *$ ovn-sbctl find advertised_mac ip=**10.0.0.11***

  datapath : **<ls>**

  ip : **10.0.0.11**

  logical_port : **<w1>**

  mac : **00:00:00:00:00:01**

  *$ ip neigh show dev **br-10** nud noarp*

  10.0.0.11 dev br-10 lladdr **00:00:00:00:00:01** NOARP

  10.0.0.1 dev br-10 lladdr **00:00:00:01:00:00** NOARP

- Fabric FRR learned IPs

  $ show bgp evpn route

  Route Distinguisher: **42.42.10.12**:2

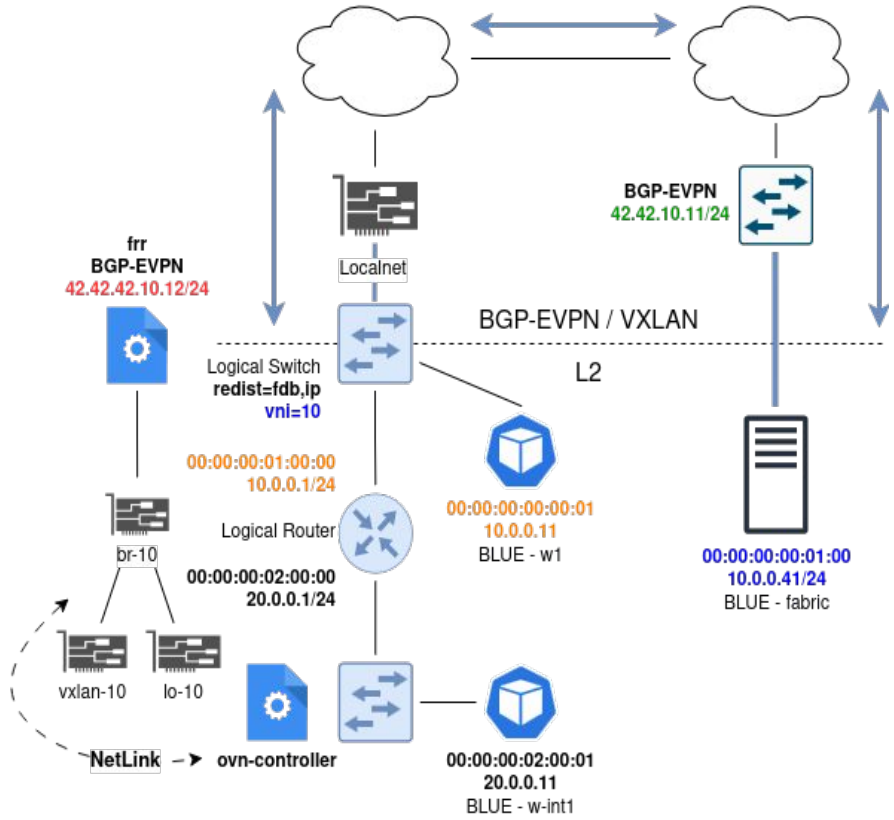
  *> [2]:[0]:[48]:[**00:00:00:00:00:01**]:[32]:[**10.0.0.11**]

  *> [2]:[0]:[48]:[**00:00:00:01:00:00**]:[32]:[**10.0.0.1**]

  $ ip neigh

  10.0.0.11 dev **br-10** lladdr **00:00:00:00:00:01 extern_learn** NOARP proto zebra

  10.0.0.1 dev **br-10** lladdr **00:00:00:01:00:00 extern_learn** NOARP proto zebra



Red Hat

Demo: Accessing OVN advertised IPs/MACs through VXLAN

- Fabric to workload w1

  $ ip netns exec fabric_workload ping ***10.0.0.11***

  64 bytes from **10.0.0.11**: icmp_seq=1 ttl=64 time=0.523 ms

  a6:c0:c5:53:70:d9 > 86:f6:e9:31:8f:44, proto UDP (17),
  42.42.10.11.39747 > **42.42.10.12**.4789: **VXLAN**, **vni 10**
  00:00:00:00:01:00 > **00:00:00:00:00:01**, proto ICMP (1)
  10.0.0.41 > **10.0.0.11**: ICMP echo request, id 1690, seq 1

  86:f6:e9:31:8f:44 > a6:c0:c5:53:70:d9, proto UDP (17)
  42.42.10.12.40290 > **42.42.10.11**.4789: **VXLAN**, **vni 10**
  00:00:00:00:00:01 > **00:00:00:00:01:00**, proto ICMP (1)
  10.0.0.11 > **10.0.0.41**: ICMP echo reply, id 1690, seq 1

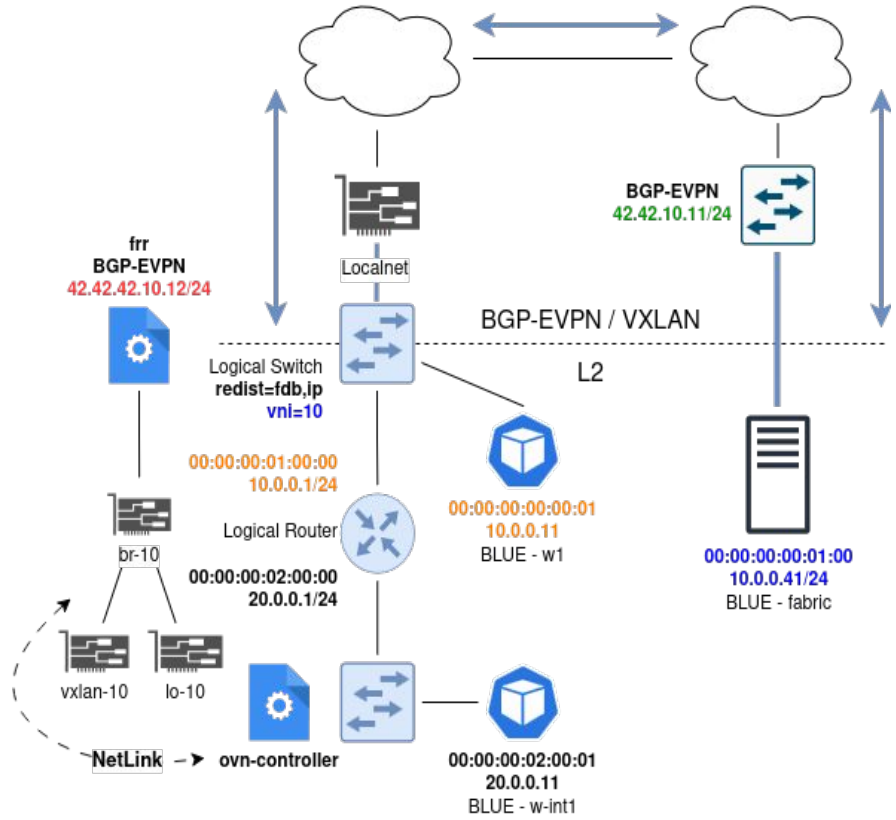
- Fabric to OVN LRP

  $ ip netns exec fabric_workload ping ***10.0.0.1***

  64 bytes from **10.0.0.1**: icmp_seq=1 ttl=254 time=11.5 ms

  a6:c0:c5:53:70:d9 > 86:f6:e9:31:8f:44, proto UDP (17)
  42.42.10.11.37026 > **42.42.10.12**.4789: **VXLAN**, **vni 10**
  00:00:00:00:01:00 > **00:00:00:01:00:00**, proto ICMP (1)
  10.0.0.41 > **10.0.0.1**: ICMP echo request, id 1720, seq 1

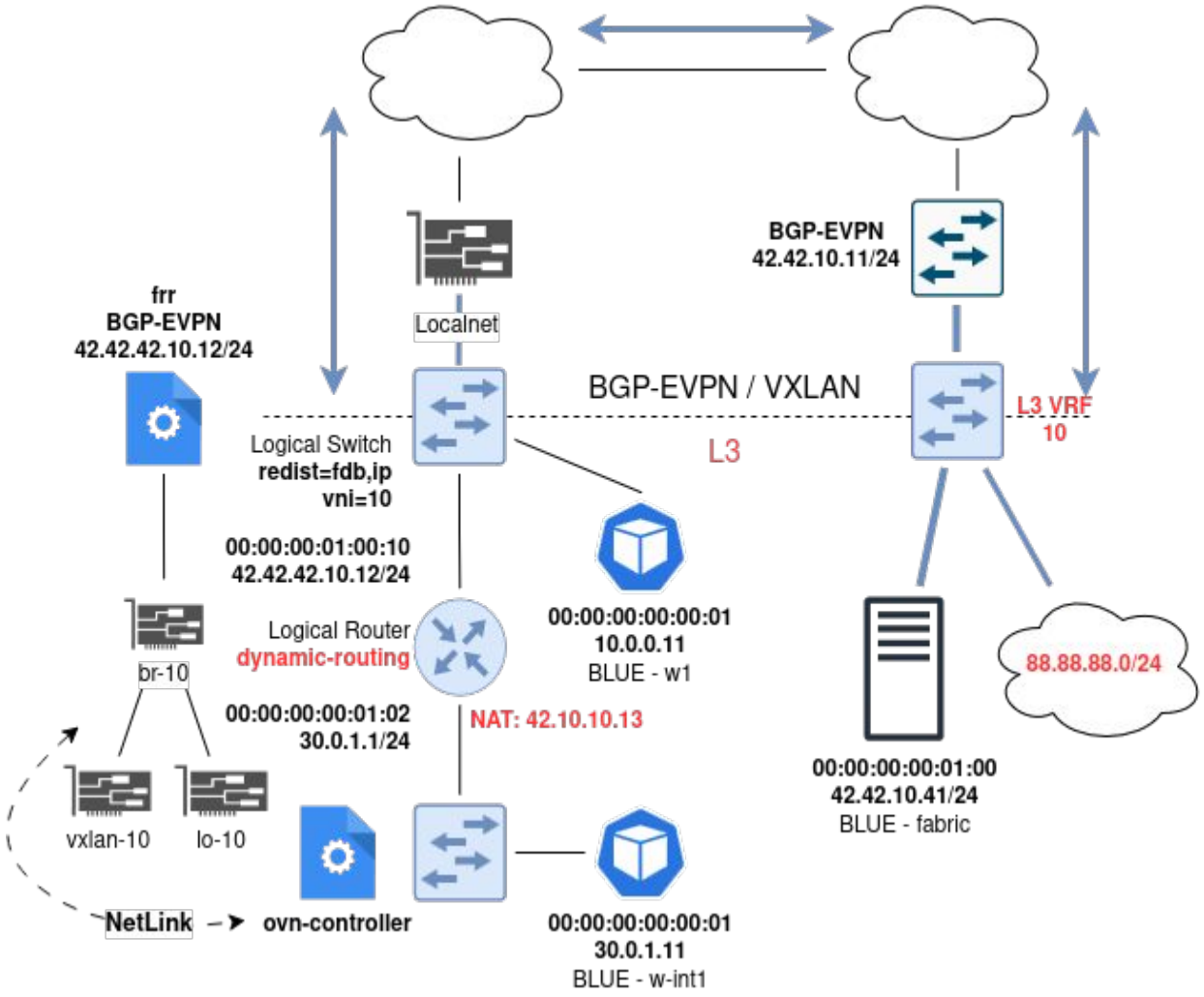
  86:f6:e9:31:8f:44 > a6:c0:c5:53:70:d9, proto UDP (17)
  42.42.10.12.53560 > **42.42.10.11**.4789: **VXLAN**, **vni 10**
  00:00:00:01:00:00 > **00:00:00:00:01:00**, proto ICMP (1)
  10.0.0.1 > **10.0.0.41**: ICMP echo reply, id 1720, seq 1



Red Hat

Future plans

Traffic flow - L3 EVPN - OVN 26.03



- Scenario used in the multinode L3 EVPN test
- One OVN cluster (left) with
  - provider logical switch configured to extend its L2 domain via EVPN using VXLAN VNI 10
  - workloads
    - directly attached to the provider logical switch
    - behind an OVN logical router **configured to dynamically learn/advertise routes in vrf 10**
- One non-OVN part of the L2 domain (right)
  - hosts connected to a BGP-EVPN aware switch/router (non-OVN) on **L3 VRF 10**
  - **other networks** (88.88.88.0/24)
- Hosts on the right must be able to reach OVN workloads (**at L3 level**)

Red Hat

Learning remote routes

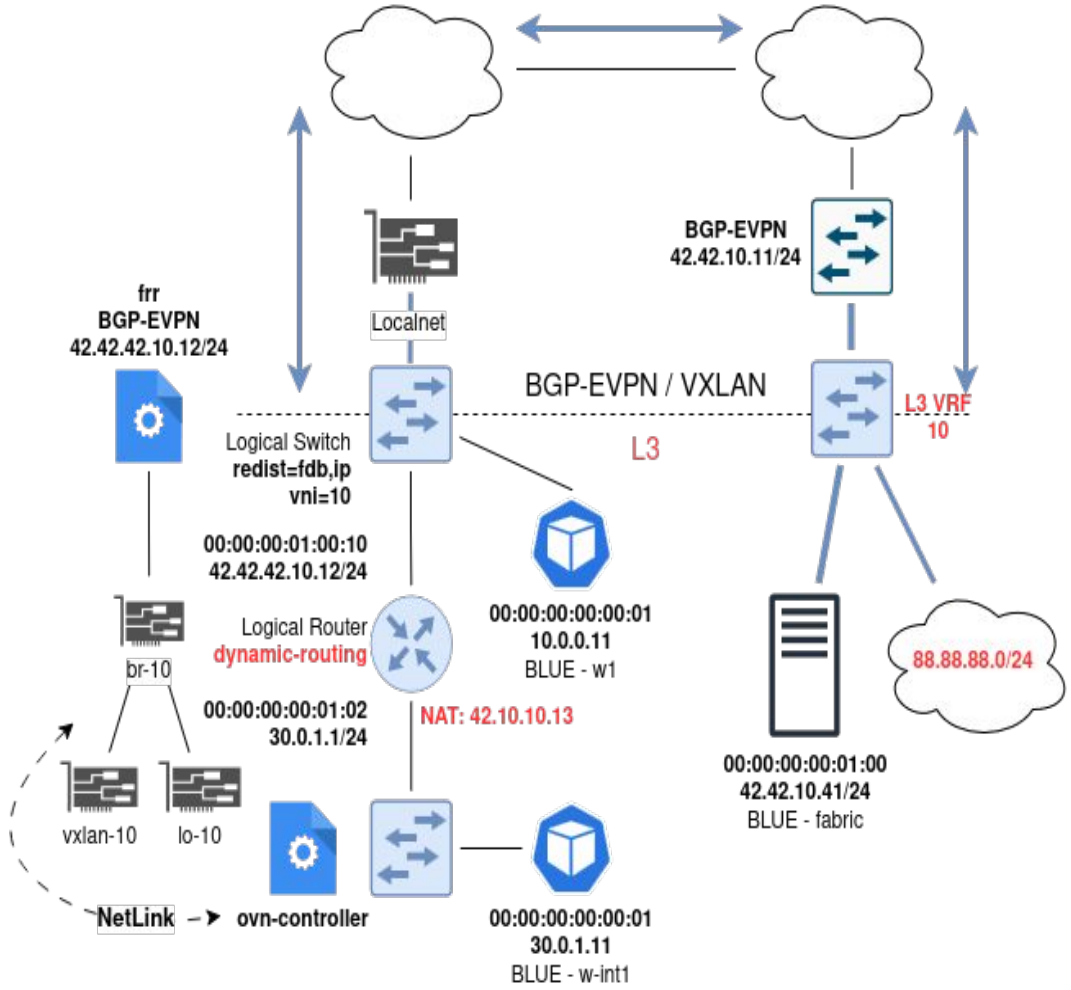
- OVN learned VTEPs
  $ ovn-appctl evpn/remote-vtep-list
  IP: **42.42.10.11**, port: 4789, vni: **10**

- FRR learned type-5 routes (on OVN node)
  $ show bgp l2vpn evpn
  Route Distinguisher: **42.10.10.11**:2
  *> [5]:[0]:**[24]:[88.88.88.0]**

- OVN learned routes
  *$ ip r l table **10***
  88.88.88.0/24 nhid 24 via **42.10.10.11** dev br-10 proto **bgp**

  $ ovn-sbctl find learned_route ip_prefix=88.88.88.0/24

  datapath : **<logical-router>**
  ip_prefix : **88.88.88.0/24**
  logical_port : **lr-gw**
  nexthop : **42.10.10.11**



Red Hat

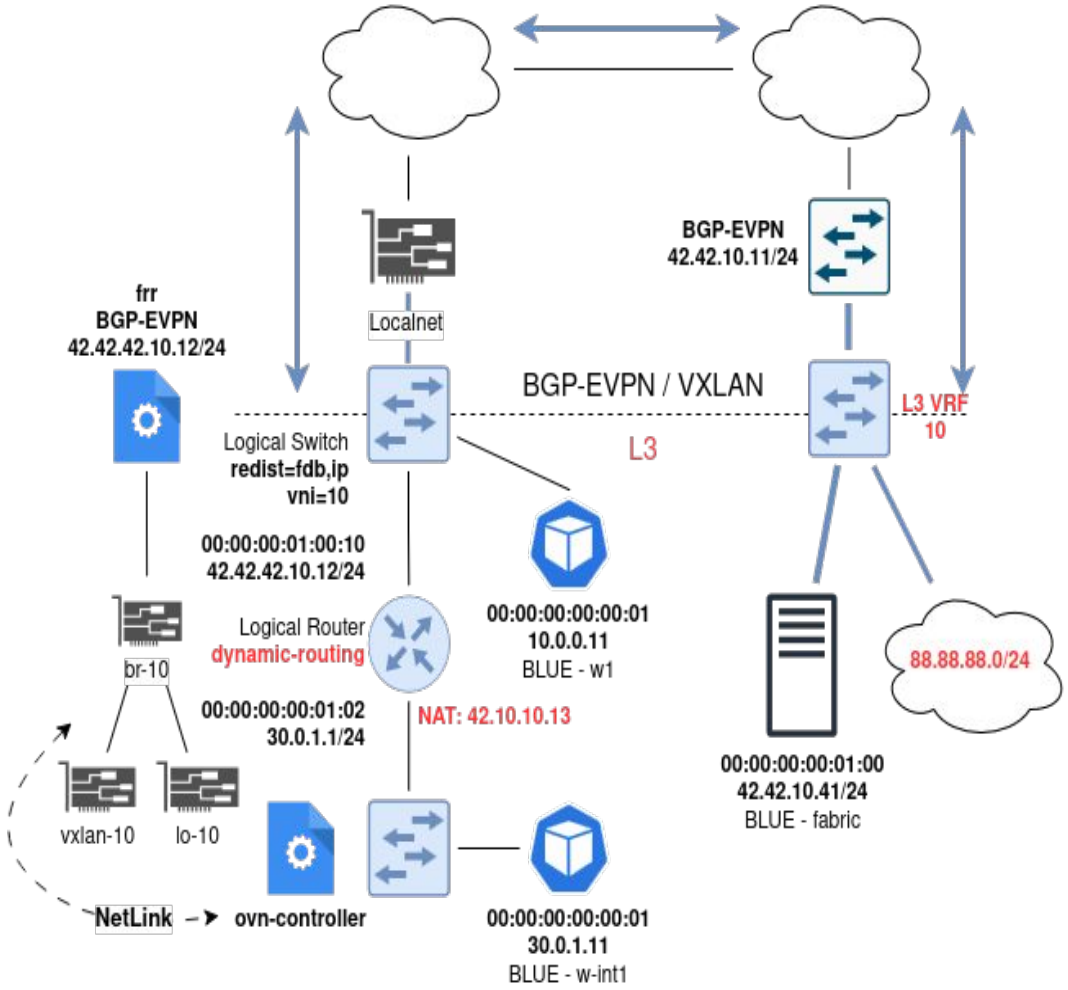
Advertising OVN routes

- OVN advertised connected internal networks
  $ ovn-sbctl find advertised_route ip_prefix=30.0.1.0/24
  datapath : **<logical-router>**
  ip_prefix : **30.0.1.0/24**
  logical_port : **lr-int1**

- OVN advertised NAT IPs
  $ ovn-sbctl find advertised_route ip_prefix=42.10.10.13
  datapath : **<logical-router>**
  ip_prefix : **42.10.10.13**
  logical_port : **lr-gw1**
  tracked_port : **w1-int1**

  *$ ip r l table **10** type blackhole*
  blackhole 30.0.1.0/24 proto **ovn** metric 1000
  blackhole 42.10.10.13 proto **ovn** metric 100

- Fabric FRR learned type-5 routes
  $ show bgp l2vpn evpn
  Route Distinguisher: **42.10.10.12**:2
  *> [5]:[0]:**[24]:[30.0.1.0]**
  *> [5]:[0]:**[32]:[42.10.10.13]**



Red Hat

Demo: Accessing OVN advertised routes through VXLAN

- fabric to workload w-int1

  $ ip vrf exec vrf-10 ping -I 42.10.10.41 30.0.1.11

  56:fa:77:64:54:b2 > 86:31:0f:0e:a1:4a, proto UDP (17),
  42.10.10.11.48193 > 42.10.10.12.4789: **VXLAN**, **vni 10**
  00:00:01:00:00:10 > **00:00:00:01:00:10**, proto ICMP (1),
  42.10.10.41 > **30.0.1.11:** ICMP echo request, id 2303, seq 1

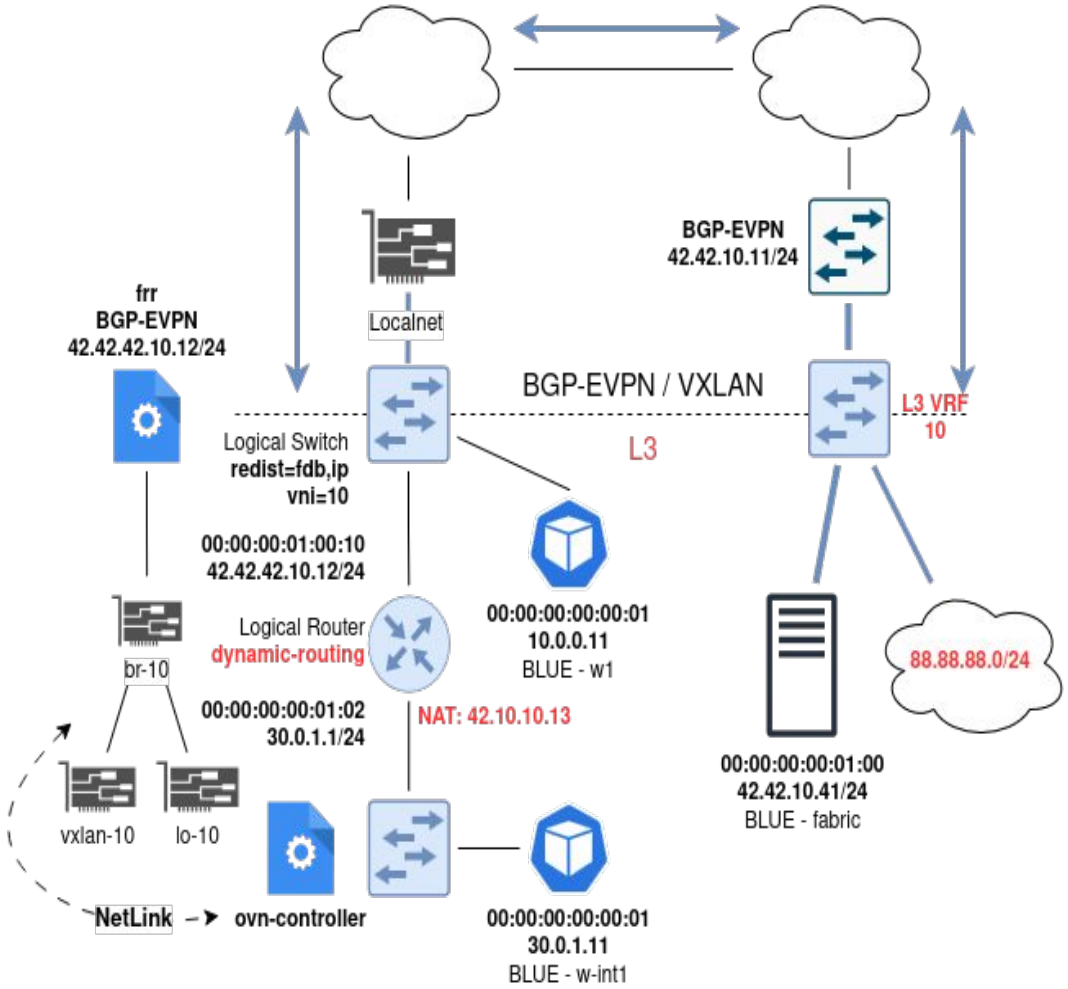
  86:31:0f:0e:a1:4a > 56:fa:77:64:54:b2, proto UDP (17)
  42.10.10.12.39813 > 42.10.10.11.4789: **VXLAN**, **vni 10**
  00:00:00:01:00:10 > 00:00:01:00:00:10, proto ICMP (1)
  30.0.1.11 > **42.10.10.41**: ICMP echo reply, id 2303, seq 1

- fabric to workload w-int1 NAT IP

  $ ip vrf exec vrf-10 ping -I 42.10.10.41 42.10.10.13

  56:fa:77:64:54:b2 > 86:31:0f:0e:a1:4a, proto UDP (17)
  42.10.10.11.60568 > 42.10.10.12.4789: **VXLAN**, **vni 10**
  00:00:01:00:00:10 > 00:00:00:01:00:10, proto ICMP (1)
  42.10.10.41 > **42.10.10.13**: ICMP echo request, id 2311, seq 1
  (on w-int1)
  42.10.10.41 > **30.0.1.11**: ICMP echo request, id 2311, seq 1

  86:31:0f:0e:a1:4a > 56:fa:77:64:54:b2, proto UDP (17)
  42.10.10.12.39813 > 42.10.10.11.4789: **VXLAN**, **vni 10**
  00:00:00:01:00:10 > 00:00:01:00:00:10, proto ICMP (1)
  42.10.10.13 > **42.10.10.41**: ICMP echo reply, id 2311, seq 1
  (on w-int1)
  30.0.1.11 > **42.10.10.41**: ICMP echo reply, id 2311, seq 1



Red Hat

Other plans

- Make the native EVPN support non-experimental
- Fine-grain configuration of host-side bridge/vxlan/lo interface names?
- We would like to hear your opinion!

Thank you!

Red Hat