
# The modern, interoperable DC

## Part 3: Seamless and secure connectivity for edge computing and hybrid clouds

Jerzy Kaczmarski
Adam Kułagowski

team@codilime.com

# Introduction

**10**
Years in business

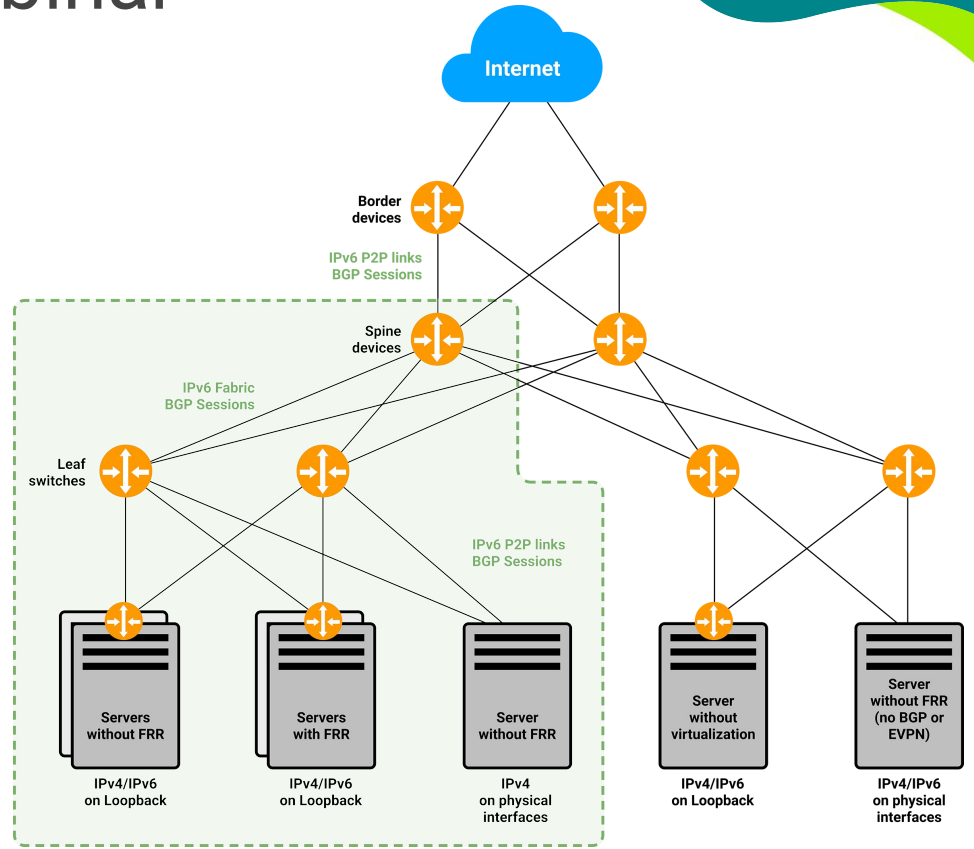
**3**
Offices

**200+**
Network, software & DevOps engineers

**6**
Our clients' Time zones

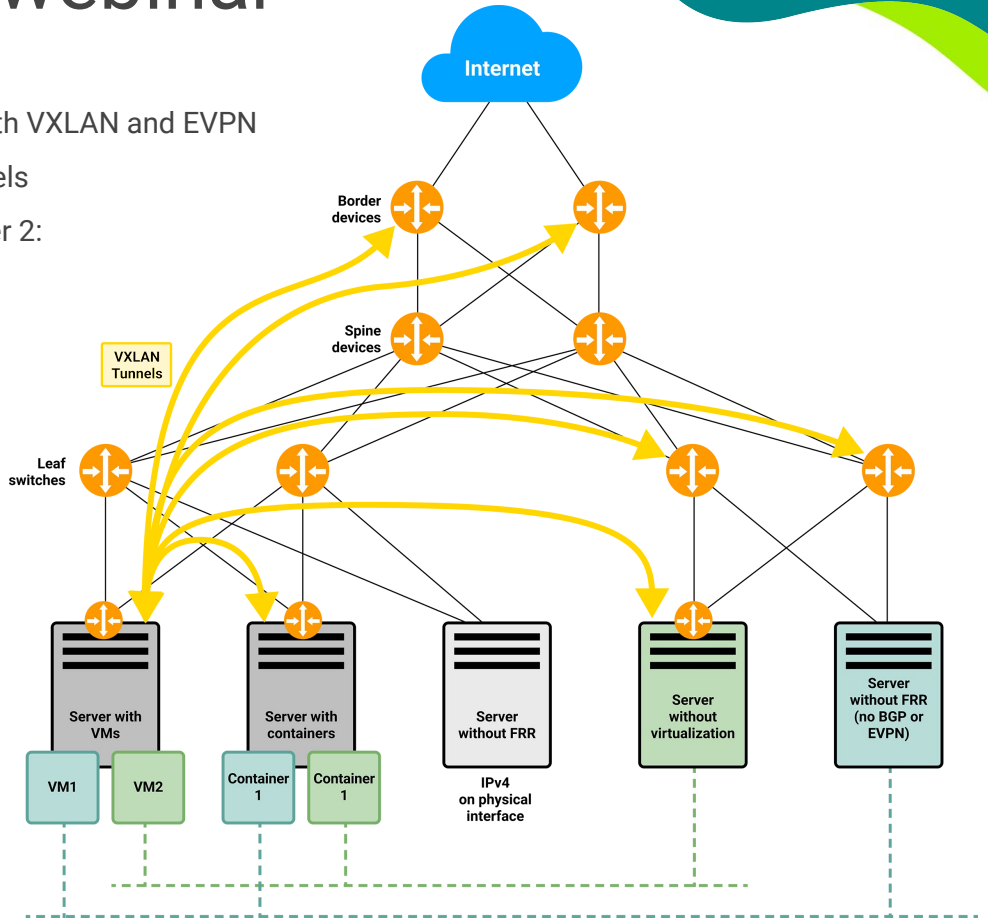
# Recap of the first webinar

- Scalable and easily automated networking architecture for DC environments
- Using IPv6 to simplify deployment and address management
- Using BGP for scalability, flexibility, load balancing and fast failover
- Automatically installing and configuring FRR



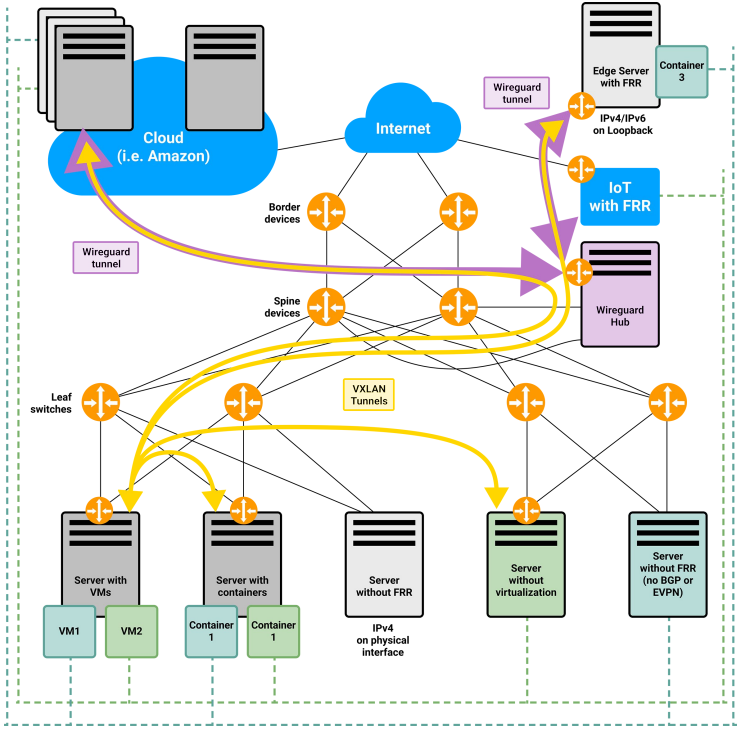
# Recap of the second webinar

- Providing Layer 2 connectivity over IPv6 fabric with VXLAN and EVPN
- Multi-tenancy using virtualization, VRFs and tunnels
- Interconnecting heterogeneous resources at Layer 2:
  - legacy server without FRR or virtualization
  - server with FRR installed
  - containers (i.e. kubernetes)
  - virtual machines



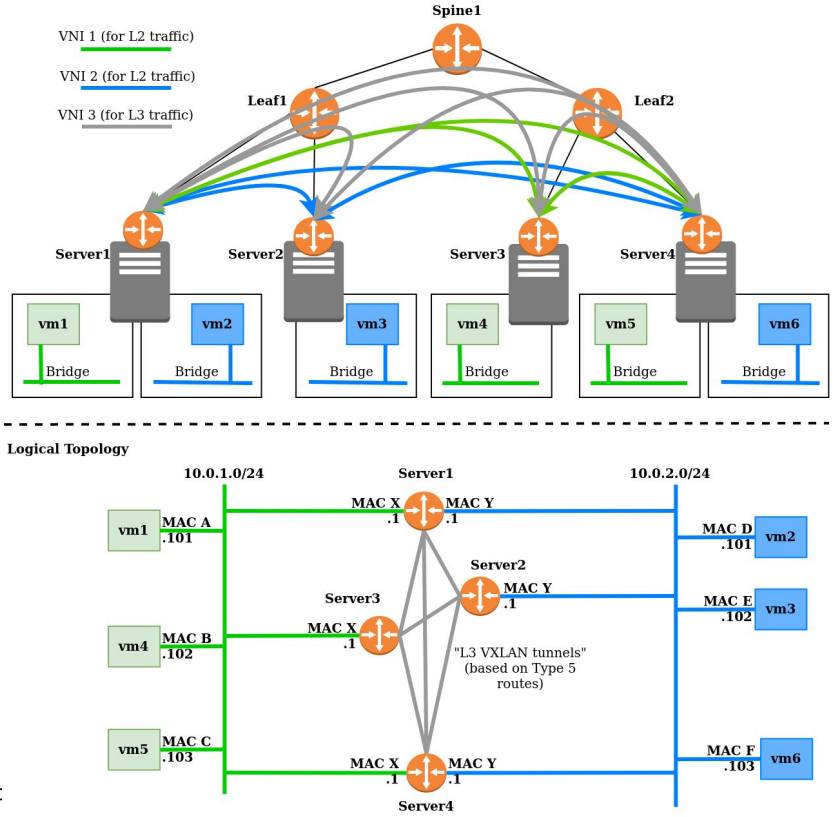
# Today’s webinar agenda

- VXLAN routing between overlay networks
- Overlay-to-underlay gateway
- Encrypting VXLAN tunnels over untrusted networks
- Extending Layer 2 connectivity to public clouds
- Branch office EVPN
- Extending VXLAN tunnels to Edge Servers and IoT devices
- Demo
- Q&A



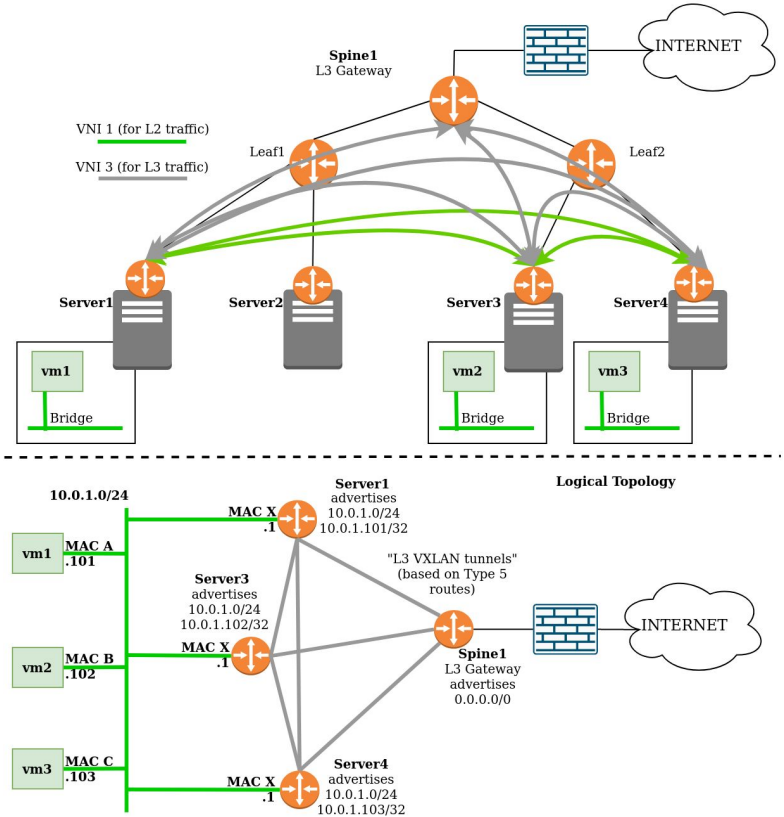
# Routing between overlay networks

- Very similar to routing between VLANs
- Requires a 'VXLAN L3 Gateway'
  - can be hardware (DC switch) or software (i.e. Linux)
- EVPN has built-in support for advertising network prefixes ('Type 5' routes)
  - Allows advertising of prefixes used in overlay networks (similar to dynamic routing protocol in underlay network)
- Communication between overlay networks can be restricted:
  - Routing policies
  - Stateless ACLs or stateful firewall (if supported by the L3 Gateway)
  - Steering traffic through a dedicated network firewall (advanced setup)
- 'Distributed routing' achieved when leaf switches / servers act as L3 gateways:
  - Optimizes traffic flow patterns compared to centralized routing (which is when only spine switches perform routing between overlay networks)



# Overlay <> Underlay communication

- By default no communication between Overlay and Underlay networks
- Possible through VXLAN L2 Gateway
  - usually a DC switch
  - L2 connectivity between an overlay network and a VLAN
  - security based on routing policies and ACLs (if supported by the gateway)
- Possible through VXLAN L3 Gateway
  - usually a router with NAT capabilities (NAT can also be performed on a dedicated firewall)
  - L3 connectivity between many overlay networks and prefixes in the underlay network
  - security based on routing policies, ACLs and stateful firewall (if supported by the gateway)

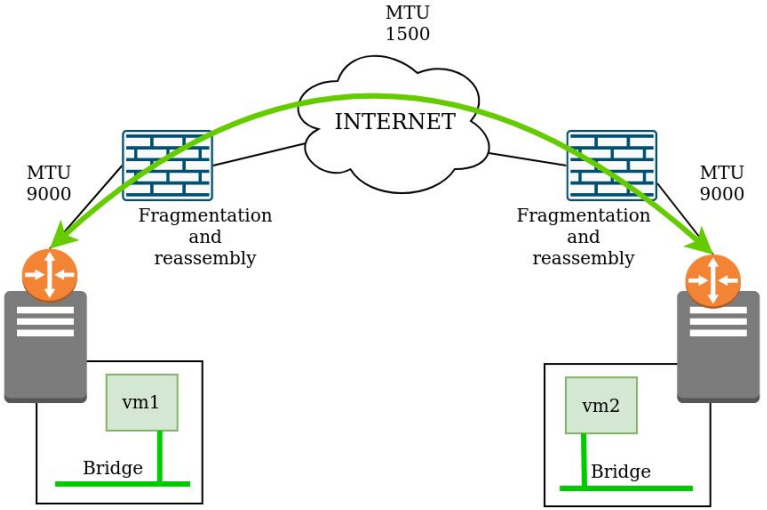


# Using VXLAN tunnel over the Internet

- To forward 1500-byte frames in VXLAN without fragmentation, Jumbo Frames are required (not available over the Internet)
- VXLAN packets can be fragmented (RFC7348):

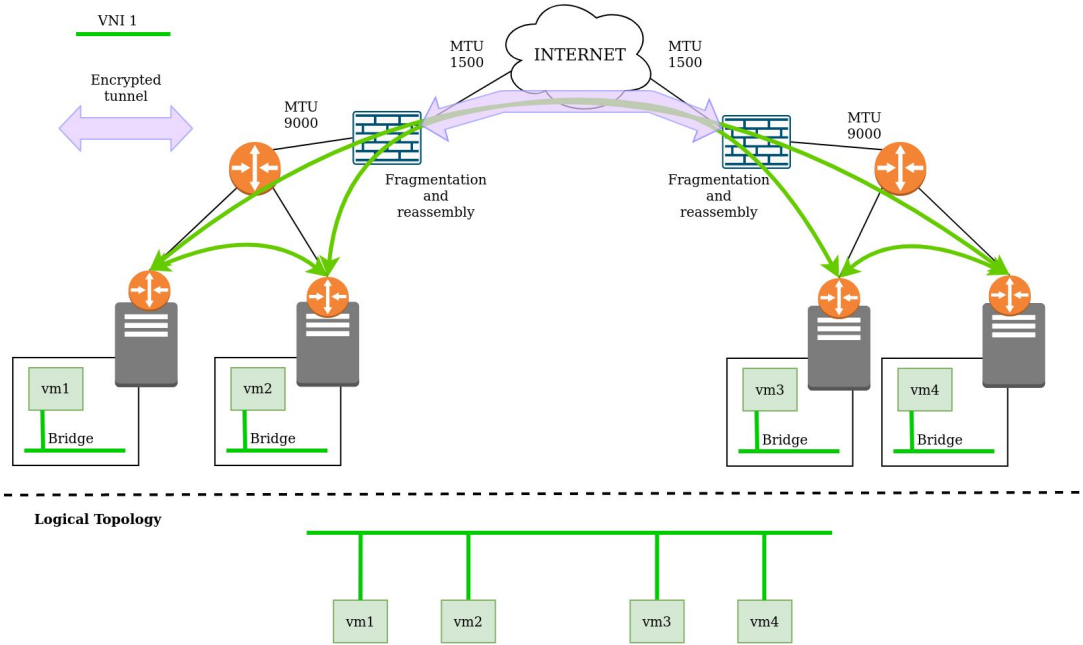
*VTEPs MUST NOT fragment VXLAN packets. Intermediate routers may fragment encapsulated VXLAN packets due to the larger frame size. The destination VTEP MAY silently discard such VXLAN fragments.*

- Full disclosure - fragmentation is generally not recommended
  - extra resource utilization on devices doing fragmentation and reassembly
  - slightly lower throughput, introduces additional latency
  - some more elusive issues summarized in RFC8900
  - nevertheless, very commonly used (i.e. IPsec)
- Conclusion
  - VXLAN tunnels over the Internet possible with proper configuration



# Securing VXLAN-tunneled traffic

- Layer 1 and Layer 2 (MACSec) encryption
  - Not usable over the Internet
  - In most cases requires specialized hardware (wire-rate throughput)
  - Suitable for encryption within DC (MACSec) and for some DCI scenarios
- Layer 3+ encryption (IPsec, Wireguard, TLS, etc.)
  - Usable over the Internet
  - Does not require specialized hardware
  - Suitable for a wide range of connectivity scenarios, including NAT traversal
  - Need to handle fragmentation and reassembly when tunneling large VXLAN packets



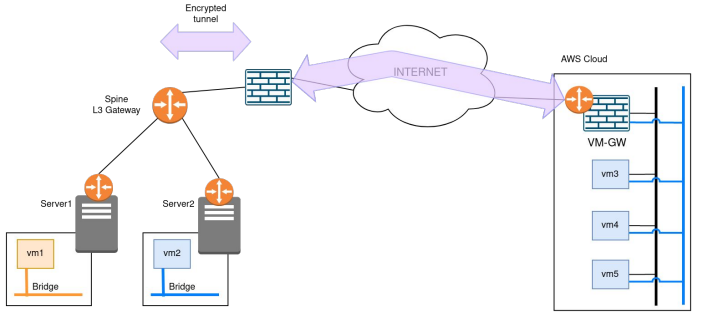
# Why to extend overlay to public cloud

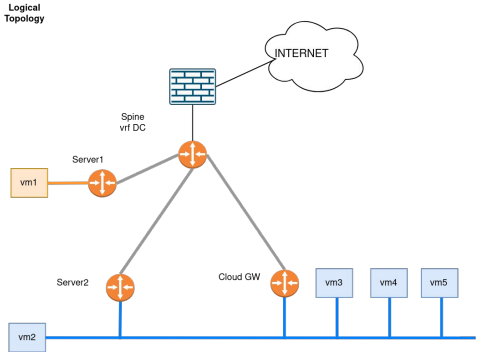
- Existing Network in Cloud (AWS/Azure/GCP) is severely limited:
  - No Layer 2
    - proxy ARPs
    - no custom routing
    - /32 on interface in GCP
  - No L2/L3 multicast
  - Limited IPv6 support
- Native VPN access can be very pricey
- Limited subsets of protocols (no GRE)
- Slow failover (no BFD)
- Each Cloud Network is different



# Extending overlay network to public cloud

- Important considerations:
  - Seamless integration with existing VMs (no custom images)
  - ZTP using Cloud Init
  - Full network support
  - Access to native Cloud resources
  - Linux-only support
- EVPN for advertising known prefixes and endpoints
- L2 tunnels between Cloud GW and VM
- No custom agent running on VM



# Overlay network on public cloud: putting the pieces together

## Life cycle of VM

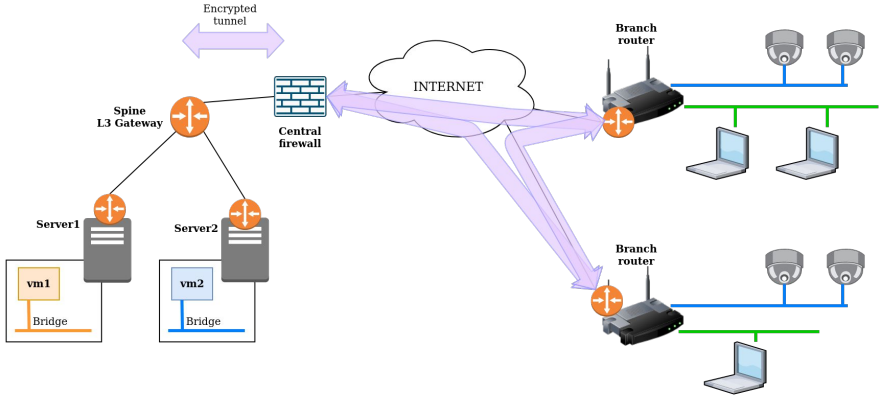
- VM is created using AWS web UI
  - Static user-data is added
- During each boot, L2 interface (GenEve) is created using Cloud Init
- On first boot - ZTP is performed
  - GW creates L2 endpoint
  - VM gets network reconfigured
- On GW, Cron removes inactive GenEve endpoints (after EC2 termination)

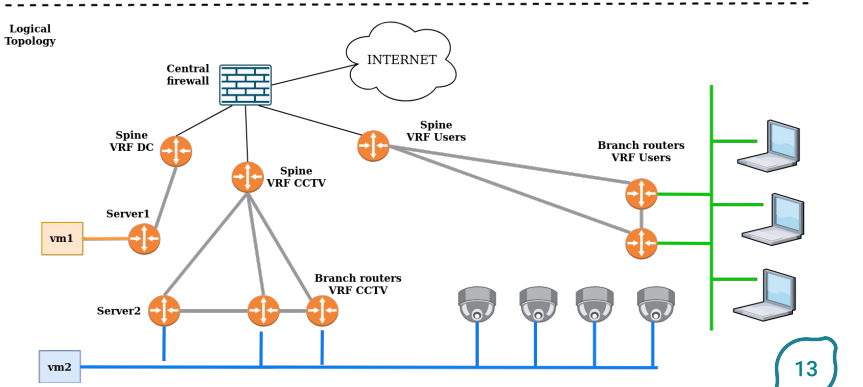
**ZTP has been tested on:**

- Ubuntu 20.04
- Redhat 8
- SUSE 15SP2
- Amazon Linux

# Branch office EVPN

- Technology stack: EVPN (i.e. FRR) + VXLAN + Encrypted tunnel (i.e. Wireguard)
- L2 and L3 connectivity between branch offices and to HQ/DCs
- Can run the solution on a variety of hardware:
  - Enterprise-grade networking devices
  - Servers/MiniPCs with extra connectivity (i.e. USB WiFi, USB networking cards, external switch, etc.)
  - Mikrotik/D-Link/ASUS/etc. routers running OpenWrt
- Possible redundancy and load balancing when using more than one Internet connection
- Segment branch networks with VLANs and extend them between branches using overlay networks, i.e.:
  - Frontend Servers VLAN <> overlay with DB servers in the Data Center
  - CCTV VLAN <> overlay with video storage and processing servers
  - Users VLAN <> L3 connectivity to DC resources and to Internet through central firewall

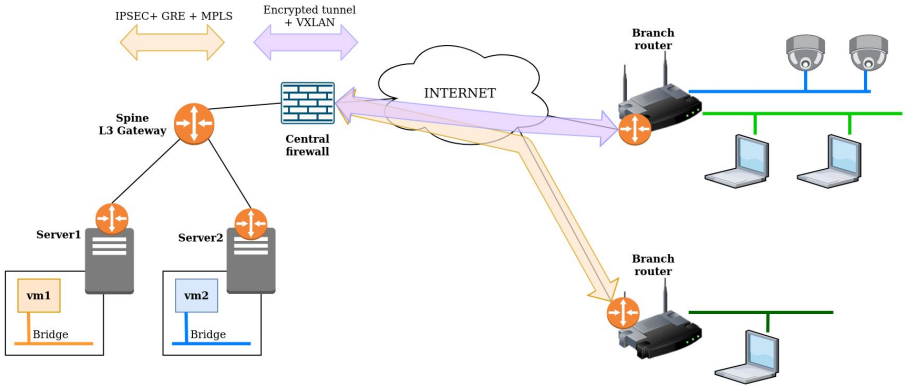


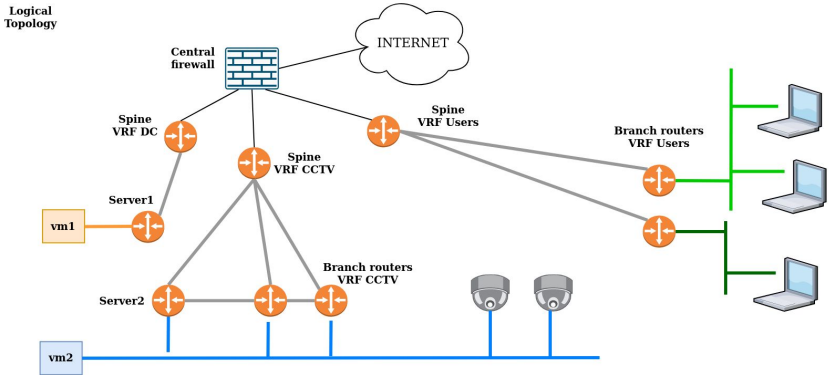



# Branch office - legacy

## What if we have a device not supporting EVPN/Wireguard ?

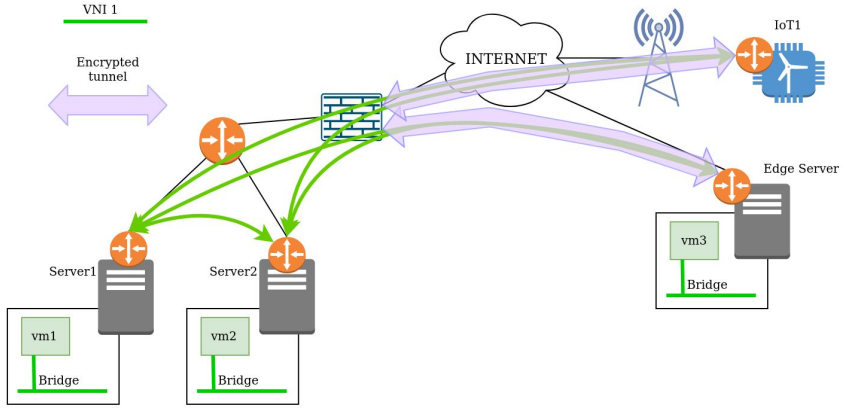
- Depends on the CE model but most support MPLS+BGP
- FRR supports L3VPN over GRE + MPLS
  - We can add IPSEC to protect payload
  - On HUB we can redistribute between L3VPN and EVPN
  - On Linux there is no support for VPLS
- All other endpoints remain unchanged
  - The need for MPLS/L3VPN is limited to HUB
- This solution adds support for multiple devices such as:
  - Juniper SRX
  - Mikrotik
  - others



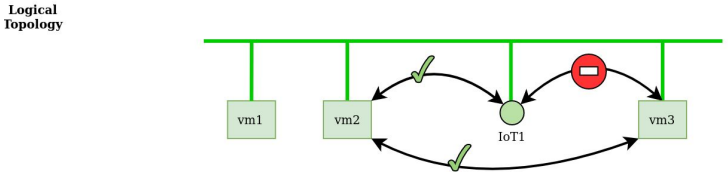



# Extending overlay to Edge Servers and IoTs

- Same technology stack - EVPN (i.e. FRR) + VXLAN + Encrypted tunnel (i.e. Wireguard)
  - Supported on various CPU architectures (x86_64, arm64, armhf, mips, ppc64el, etc.)
  - Can be run on single-board computers (i.e. Raspberry Pi)
  - Throughput mainly CPU-bound
  - Scalability mainly RAM-bound (alleviated by route aggregation and filtering)
  - Should use encrypted tunnel that supports client-to-site VPNs and can work through a NAT
- Provides secure L2 and/or L3 connectivity to resources in overlay network in DC




Example on diagram - by manipulating routes/tunnels we can provide selective connectivity within a single overlay segment (similar to Private VLANs).



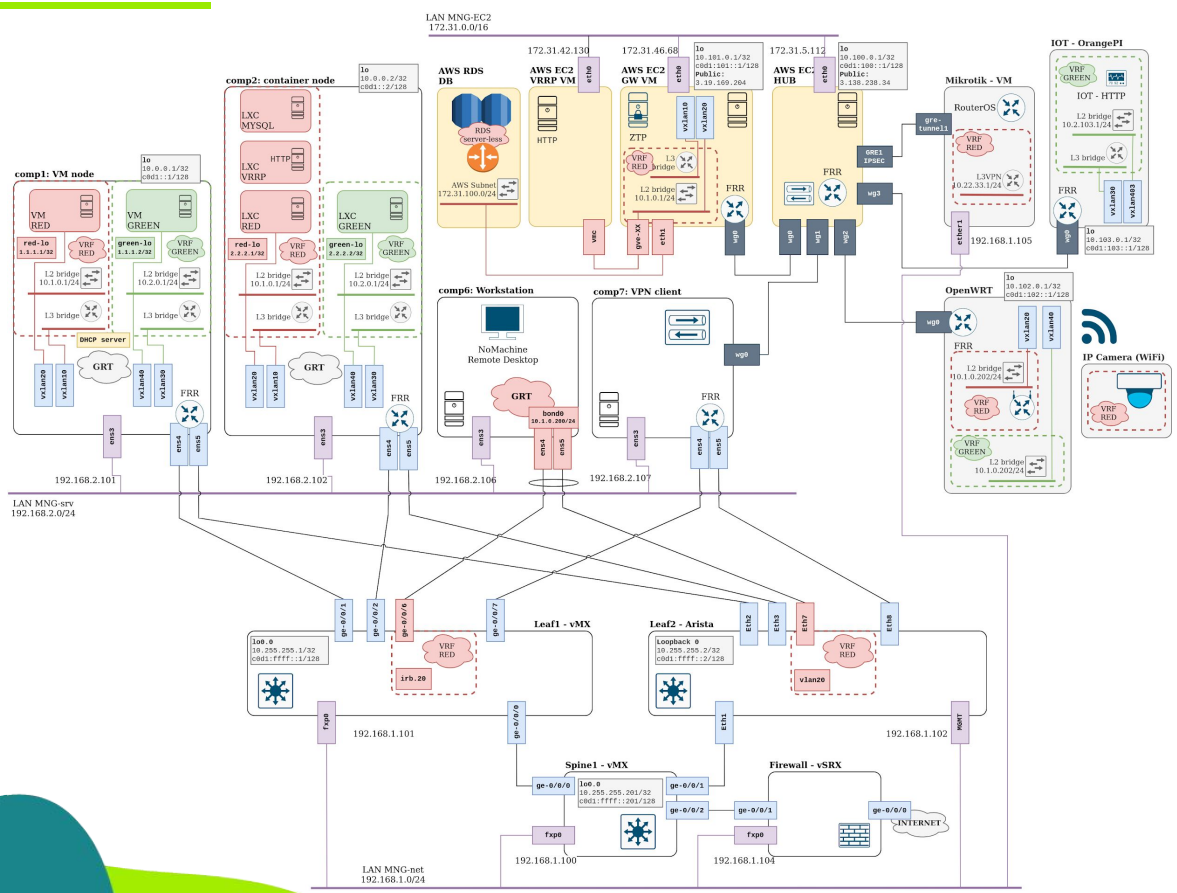
# Demo agenda

- Presentation of the topology
- Overlay <> Underlay/Internet communication
  - Making use of L3 VXLAN Gateway, EVPN Type 5 routes and NAT
- Extending overlay to public cloud
  - Service migration from local DC to AWS Cloud using VRRP
  - Service advertisement on AWS EC2 using BGP
  - Accessing AWS native resources (RDS)
  - AWS EC2 deployment using ZTP and Cloud Init

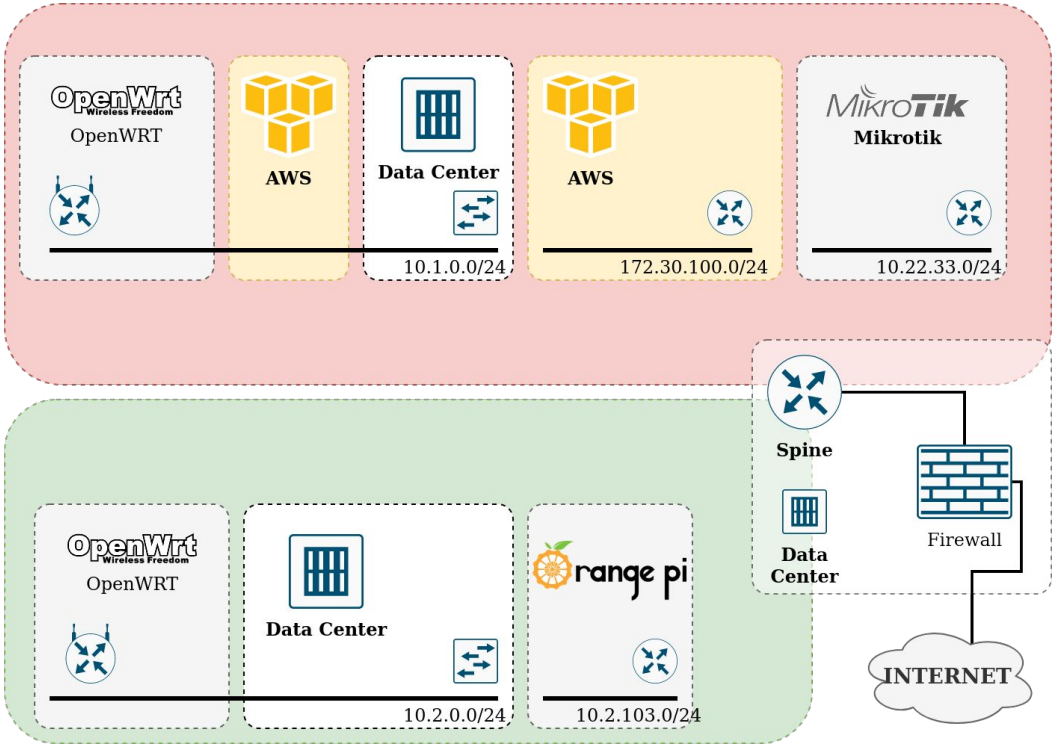
# Demo agenda cont.

- Branch office EVPN
  - Router with two VLANs, each connected to a different overlay network
    - WiFi, SSID ‘cameras’ - ONVIF camera access from DC on L2
    - LAN, ‘users’ - Internet access through DC where central firewall filters traffic
  - Legacy devices support by example of Mikrotik
- Extending overlay to IoT
  - Send command from local DC to IoT to get reading from a sensor
  - Send another command to perform action on a device connected to the IoT

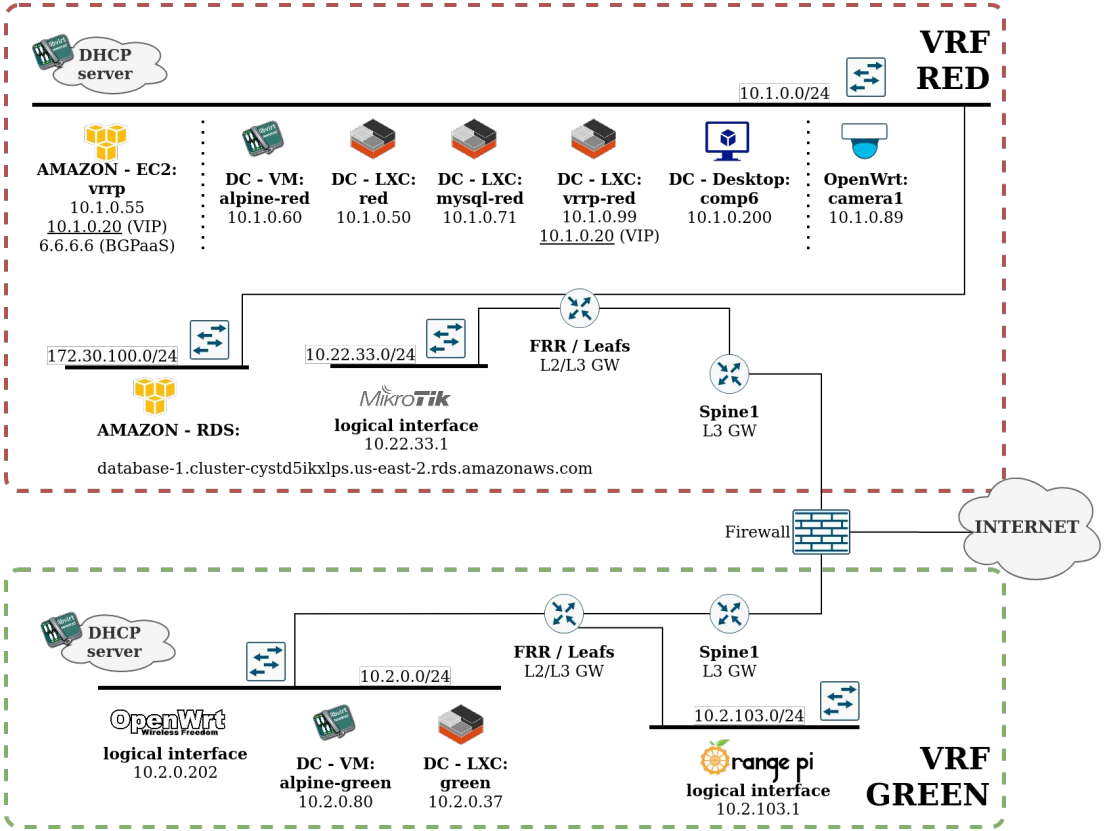

# Demo


LAN MNG-EC2
172.31.0.0/16
AWS RDS DB
AWS EC2 VRRP VM
AWS EC2 GW VM
AWS EC2 HUB
Mikrotik - VM
IOT - OrangePI
comp1: VM node
comp2: container node
comp6: Workstation
comp7: VPN client
OpenWRT
IP Camera (WiFi)
Leaf1 - vMX
Leaf2 - Arista
Spine1 - vMX
Firewall - vSRX
LAN MNG-srv
192.168.2.0/24
LAN MNG-net
192.168.1.0/24
192.168.2.101
192.168.2.102
192.168.2.106
192.168.2.107
192.168.1.101
192.168.1.102
192.168.1.100
192.168.1.104
192.168.1.105

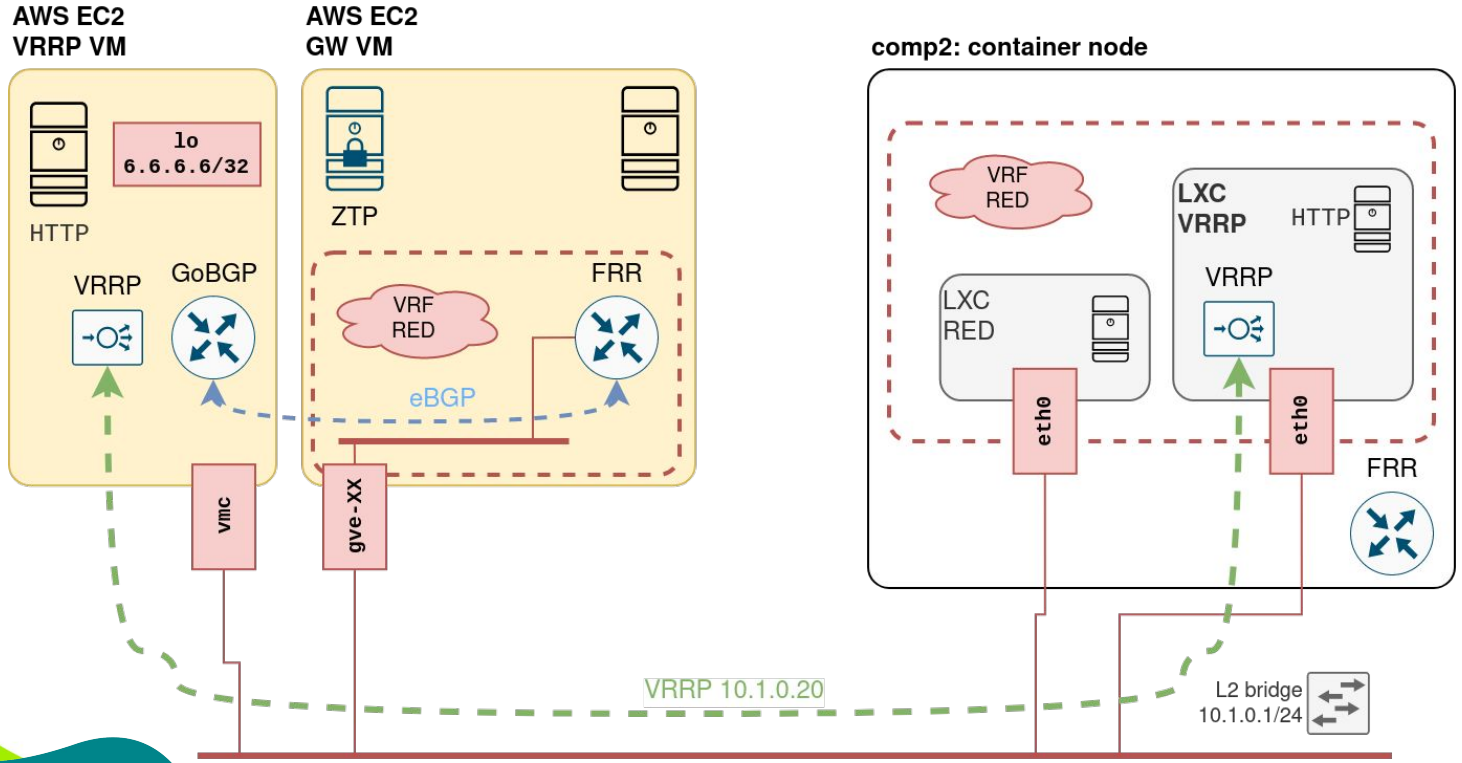
# Demo


OpenWRT
AWS
Data Center
10.1.0.0/24
AWS
172.30.100.0/24
Mikrotik
10.22.33.0/24
Spine
Data Center
Firewall
INTERNET
OpenWRT
Data Center
10.2.0.0/24
range pi
10.2.103.0/24

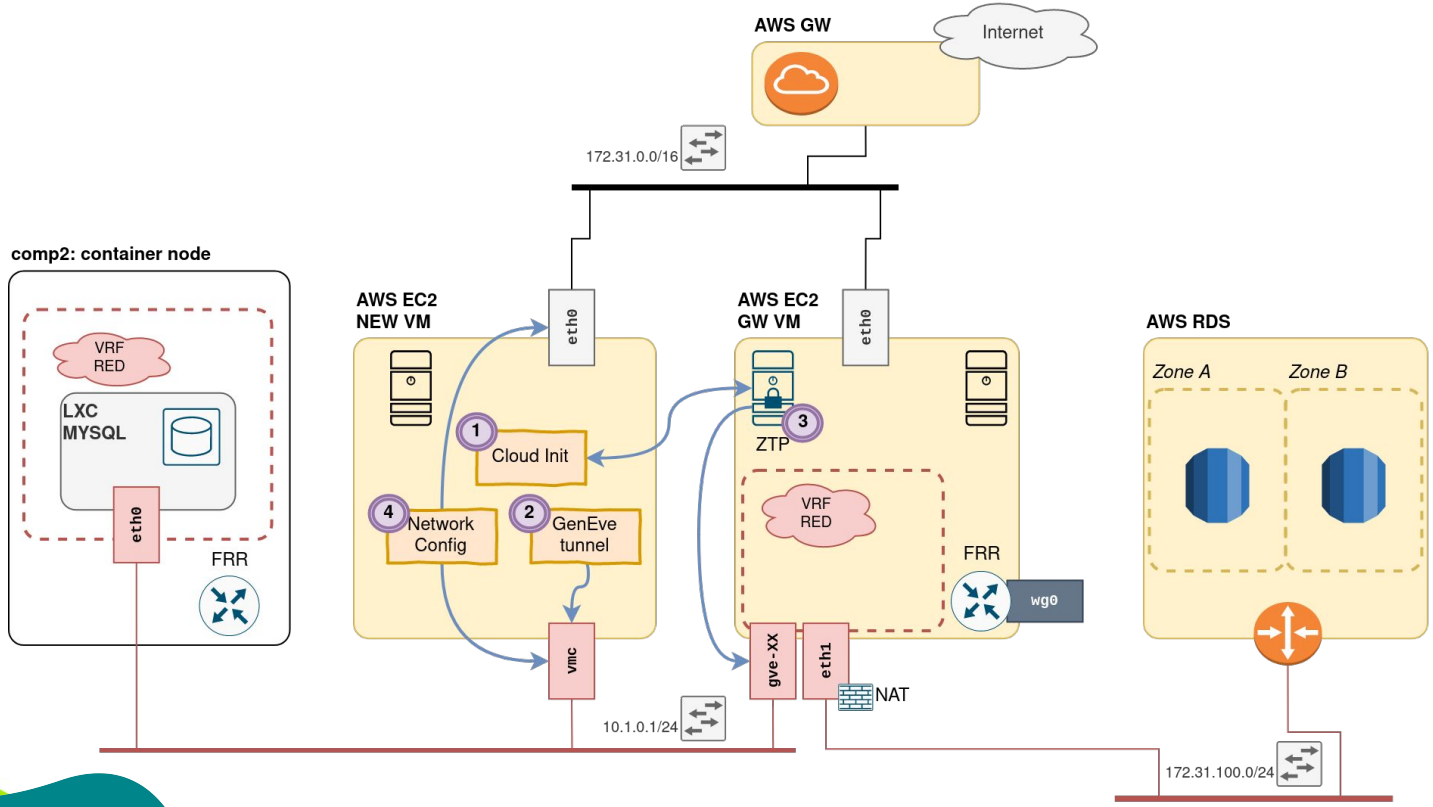
# Demo


DHCP server
VRF RED
10.1.0.0/24
AMAZON - EC2: vrrp
10.1.0.55
10.1.0.20 (VIP)
6.6.6.6 (BGPaaS)
DC - VM: alpine-red
10.1.0.60
DC - LXC: red
10.1.0.50
DC - LXC: mysql-red
10.1.0.71
DC - LXC: vrrp-red
10.1.0.99
10.1.0.20 (VIP)
DC - Desktop: comp6
10.1.0.200
OpenWrt: camera1
10.1.0.89
172.30.100.0/24
10.22.33.0/24
FRR / Leafs
L2/L3 GW
Spine1
L3 GW
AMAZON - RDS:
database-1.cluster-cystd5ikxlps.us-east-2.rds.amazonaws.com
MikroTik
logical interface
10.22.33.1
Firewall
INTERNET
DHCP server
10.2.0.0/24
FRR / Leafs
L2/L3 GW
Spine1
L3 GW
10.2.103.0/24
OpenWrt
logical interface
10.2.0.202
DC - VM: alpine-green
10.2.0.80
DC - LXC: green
10.2.0.37
orange pi
logical interface
10.2.103.1
VRF GREEN

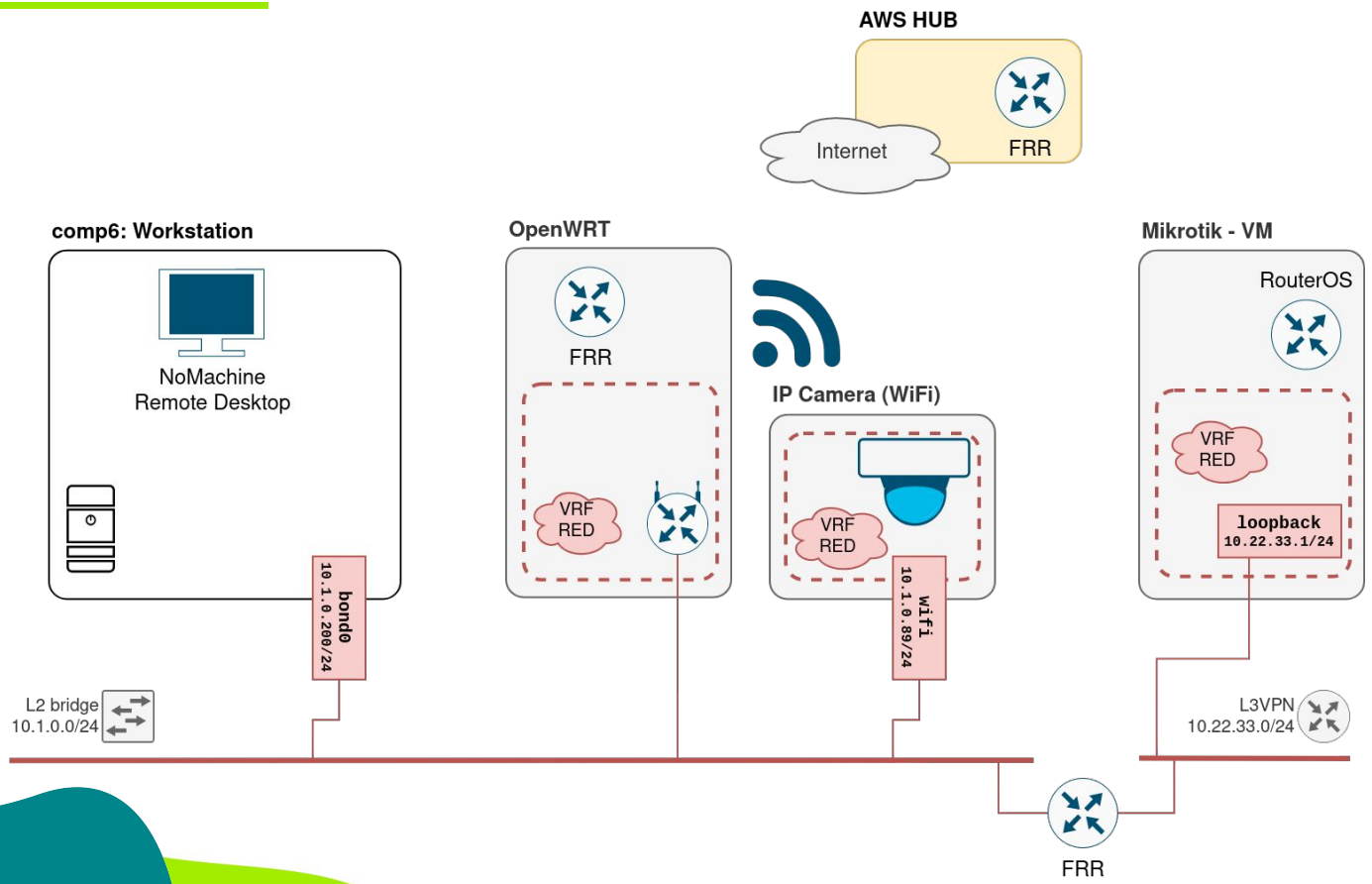
# Demo


AWS EC2
VRRP VM
AWS EC2
GW VM
comp2: container node
lo
6.6.6.6/32
HTTP
VRRP
GoBGP
ZTP
VRF
RED
FRR
eBGP
vmc
gve-XX
VRF
RED
LXC
VRRP
HTTP
VRRP
LXC
RED
eth0
eth0
FRR
VRRP 10.1.0.20
L2 bridge
10.1.0.1/24

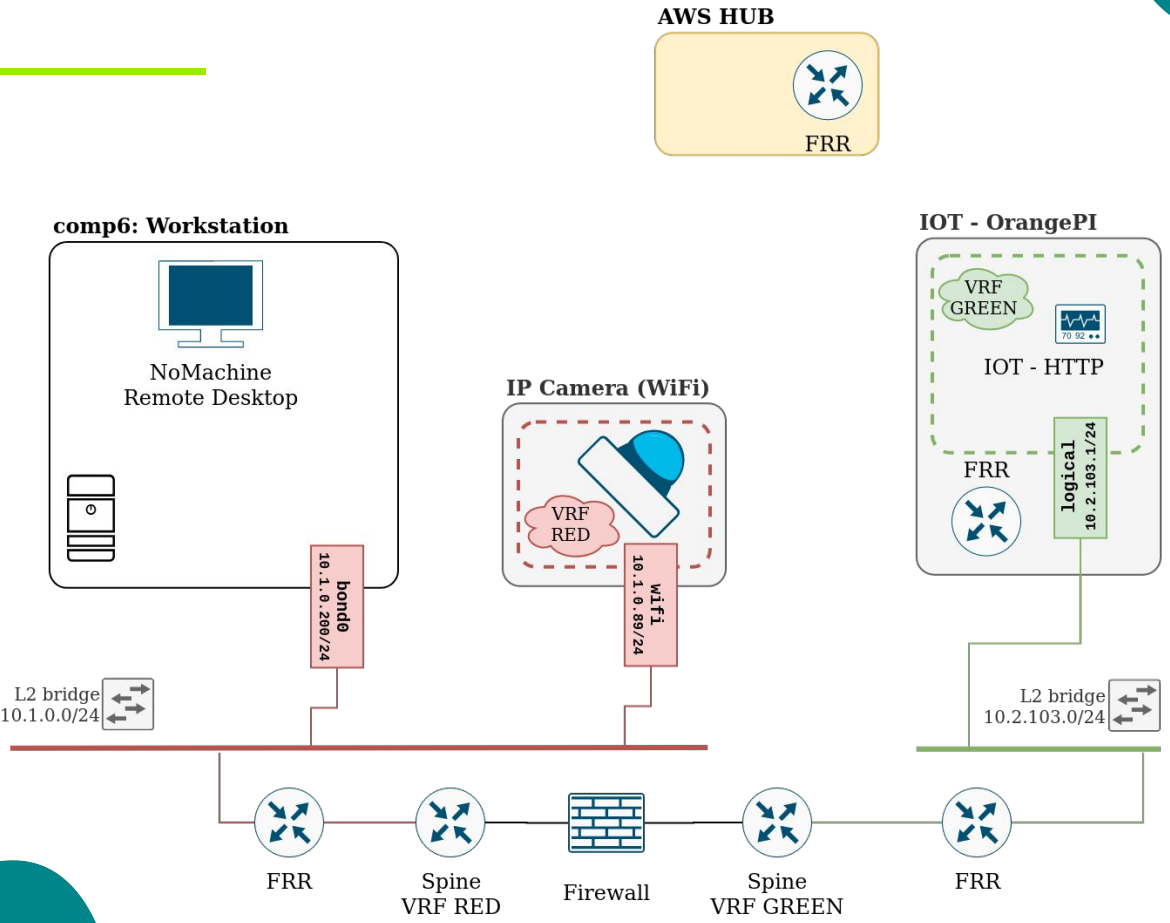
# Demo


AWS GW
Internet
172.31.0.0/16
comp2: container node
VRF
RED
LXC
MYSQL
eth0
FRR
AWS EC2
NEW VM
eth0
1
Cloud Init
4
Network
Config
2
GenEve
tunnel
vmc
AWS EC2
GW VM
eth0
3
ZTP
VRF
RED
FRR
wg0
gve-XX
eth1
NAT
10.1.0.1/24
AWS RDS
Zone A
Zone B
172.31.100.0/24

# Demo


AWS HUB
Internet
FRR
comp6: Workstation
NoMachine
Remote Desktop
bond0
10.1.0.200/24
OpenWRT
FRR
VRF
RED
IP Camera (WiFi)
VRF
RED
wifi
10.1.0.89/24
Mikrotik - VM
RouterOS
VRF
RED
loopback
10.22.33.1/24
L2 bridge
10.1.0.0/24
L3VPN
10.22.33.0/24
FRR

# Demo


AWS HUB
FRR
comp6: Workstation
NoMachine
Remote Desktop
bond0
10.1.0.200/24
IP Camera (WiFi)
VRF
RED
wifi
10.1.0.89/24
IOT - OrangePI
VRF
GREEN
IOT - HTTP
FRR
logical
10.2.103.1/24
L2 bridge
10.1.0.0/24
L2 bridge
10.2.103.0/24
FRR
Spine
VRF RED
Firewall
Spine
VRF GREEN
FRR

# Pros & cons of proposed solution

## Pros

- Possible to create logical L2 and L3 topologies that include resources from various geographical locations
- Layer 2 connectivity, when compared to Layer3-only VPNs, opens up:
  - more Local DC <> Cloud migrations scenarios
  - Hybrid Clouds where L2 HA solutions are used (i.e. sharing VIP addresses using keepalived)
  - Enterprise network architectures with some networks extended between branch locations
  - Link-local multicast connectivity inside of extended networks (without multicast routing configuration)
  - Flexibility that may be useful for various corner-case networking scenarios
- Layer 3 connectivity also possible by using EVPN Type-5 routes
- Traffic encryption possible using anything that can encapsulate UDP and perform IP packet fragmentation + reassembly
- Software components based on open standards,  can run on low-end devices using various CPU architectures

# Pros & cons of our solution

## Cons

- Relatively complicated configuration, requiring strong networking knowledge
  - Care needs to be taken with MTU configuration and packet fragmentation
- EVPN in small CPEs is rare and often requires OpenWRT flashing
- Each Cloud has a different network approach: Azure has proxy ARP, AWS ignores static routes on hosts, GCP assigns /32 to network interface
- Limited support on Cloud for non Cloud-Init system w/o L2 tunneling support: i.e. Windows
- No VPLS support on FRR - no L2 support for legacy devices

# Problems encountered

- Stable version of OpenWrt runs on linux kernel 4.14, while full feature-set EVPN on FRR requires kernel 4.18.
- GRE+IPSEC in transport mode inherits TTL from payload. While it seems like a great idea, it breaks eBGP (w/o multihop enabled)
- FRR `local-as` is not taken into account whenever iBGP/eBGP check is made (same / different AS)
- Each Linux configures network differently - even those based on `sysconfig`. This creates a lot of exceptions in ZTP
- Dnsmasq DHCP server needs `--force-broadcast` flag to work in EVPN environment
- No VRF support on default Ubuntu AWS kernels
- We had to use specific release of FRR. The 7.4.0 release had issues with L3 EVPN but 7.5.0 introduced issue with EVPN Ethernet Segment which we are using to connect legacy BMS via LACP. We ended with:

```
git fetch origin e9faa35ccd8a871f72356f436172d85aba3dd91b
```

# Summary

- Use of open-standard technologies provides advanced networking solution:
  - IPv6 Link-local addresses, IPv6 Neighbor Discovery, BGP and BGP unnumbered sessions, BFD, ECMP, cloud-init, VXLAN, VRFs, EVPN, GRE, MPLS, IPSec/Wireguard and others
  - Running a routing daemon (FRR) on a server can provide many benefits
- EVPN with VXLAN tunnels do a great job of providing L2/L3 connectivity between remote locations
- Even though networking in Public Cloud is limited, we can use overlays to achieve full connectivity

- We hope you've enjoyed this webinar series!

# Questions & Answers

Configurations, source code, topology, demo recording available at:

**https://github.com/codilime/modern-dc**

# Thank you

codilime®
CREATING VALUE


2100 Geng Road, Suite 210
Palo Alto, CA 94303
United States of America
+1 650 285 2458

Krancowa 5
02-493, Warsaw
Poland
+48 22 389 51 00

al. Grunwaldzka 472B (OBC)
80-309 Gdansk
Poland
+48 575 700 785

contact@codilime.com


SDN & NFV ▢ Cloud native & Multicloud ▢ Software Engineering ▢ UX / UI ▢ DevOps