inter.link

Network Functions

Κώστας Δρόγγος

GRNOG 16 - 19.04.2024

Pan-European Network-as-a-Service (NaaS) Offerings

MULTI-TERABIT NETWORK

- IP, Ethernet, security services
- 100G/400G in every PoP
- Connect from 300+ datacenters
- Comprehensive service statistics

TOP-TIER IP TRANSIT AS 5405

- #77 for known peers globally*
- #80 for AS Cone globally*
- 1800+ peerings
- BGP Communities

acc. to https://bgp.tools

CLOUD SERVICE MODEL

Self-service portal ▫ REST API ▫ Monthly to multi-year terms

Robust Portfolio of Services

IP Transit/Access

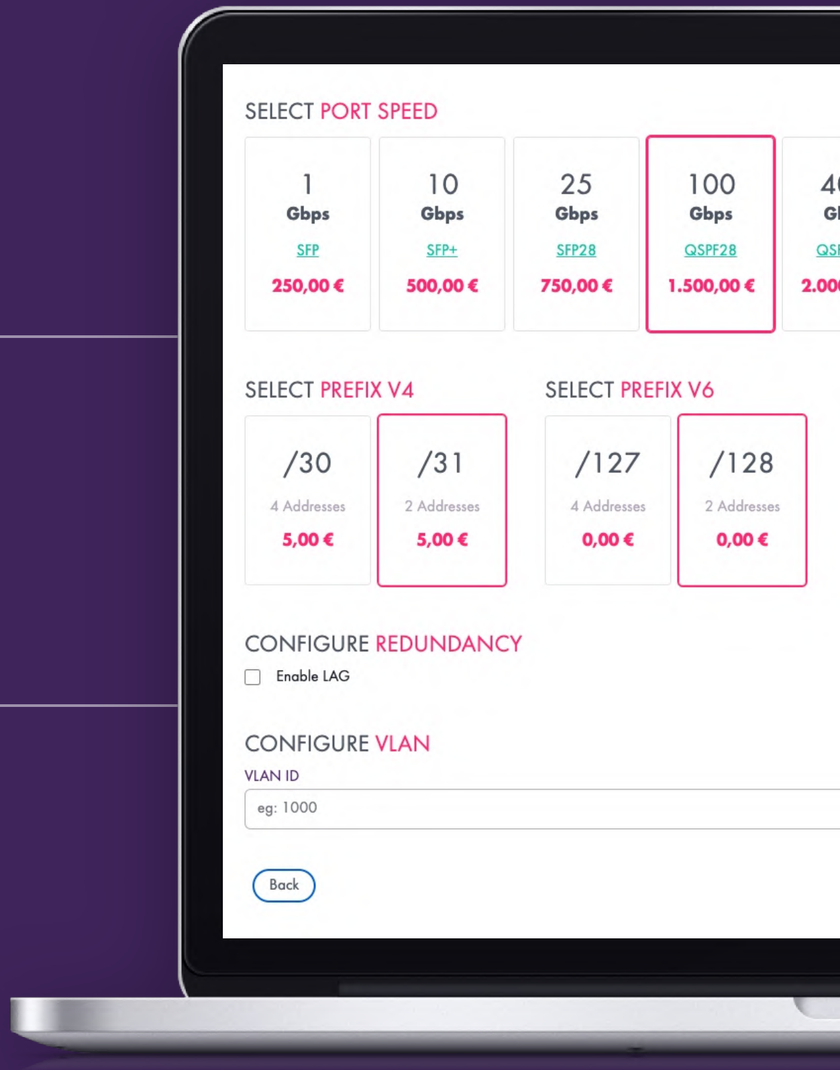
- AS 5405
- BGP communities and Flowspec support
- Extensive peerings and PNIs with regional and smaller networks across Europe
- Baseline DDoS protection included

Flex Ethernet

- Protected connectivity
- Start point-to-point
- Easily shift to multi-point connectivity
- High scalability
- M2M to 60M terms

DDoS Protection

- Direct connection via Inter.link network infrastructure
- Immediate turn-up thanks to automation
- Multiple protection tiers
- 3+ scrubbing centers with multiple Tbps capacity
- Simplified pricing with no catches



Network Functions

Assume a firewall function between a DMZ and a private network

1. has to run close to Inter.link PoPs
2. has to attach to multiple customer networks
3. multiple instances with different configs need to run
4. persistent storage?

Network Functions: To Production

PoC in the lab was successful, now time to make it into a product!

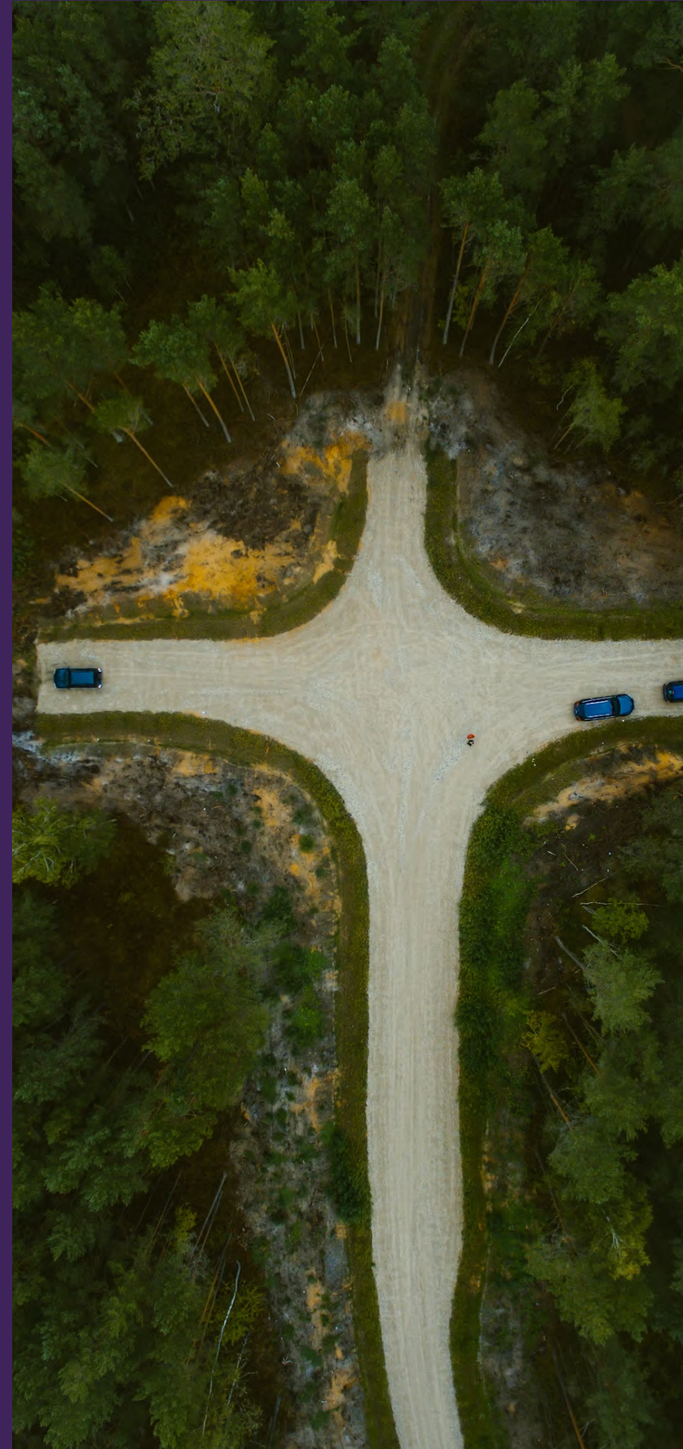
More requirements

- We've got multiple regions and they follow a share-nothing approach
- We've got different expectations
- We've got different network capacity, policies
- We've got different infrastructure
- We've got strict automation

Network Functions: To Production

Time to split

- Compute Infrastructure
- Network infrastructure
- Platform
- Deployment/automation



Implementing NF: Compute Infrastructure

Provisioning infrastructure should be standardized (cookie cutter)

- Use APIs if going with a bare metal provider
- PXE+golden images+cloud-init if rolling own hardware

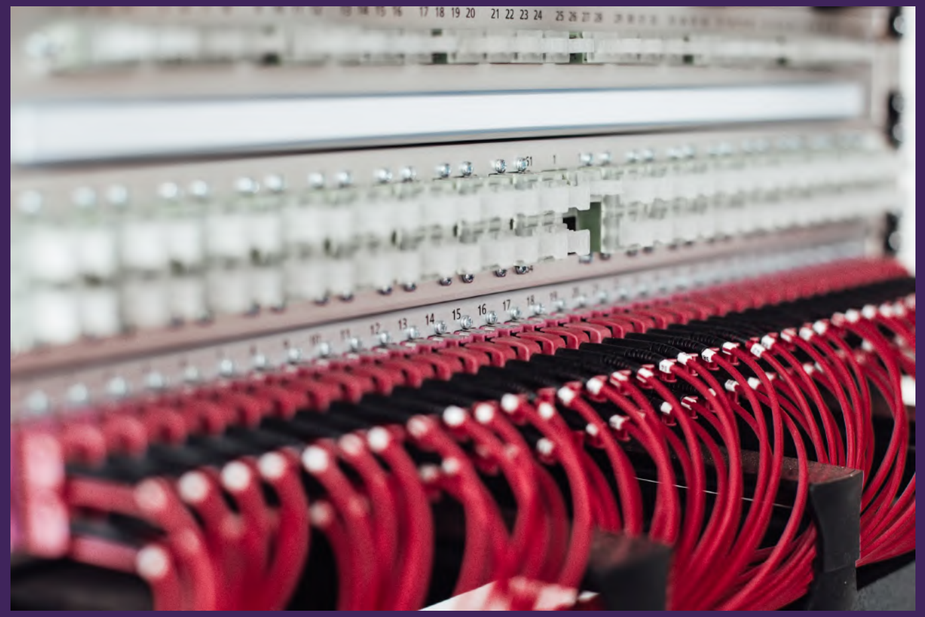
in both cases: standardize on OS release and hardware generation/type

Implementing NF: Compute Infrastructure

Minimal Ansible for making it an Inter.link server

- company has experience and tooling around it already
- small footprint
- share nothing approach fits well with Ansible

Implementing NF: Network Infrastructure



Network:

- fully automated: pipelines provision configuration
- routing of packets is dynamic
- no special roles or devices
- network is 100% routed

Compute
Network
Platform
Deployment

Implementing NF: Network Infrastructure

Our setup will be based on EVPN-VXLAN

- vendor agnostic - we now have two vendors (network, linux)
- mature in both vendors we have
- but... we don't use vxlan with our shiny new Flex-Ethernet product
- but... what Flex-Ethernet uses wasn't mature in Linux

Implementing NF: Network Infrastructure

Compute
Network
Platform
Deployment

What to do?

> for this phase we'll encap/decap in an out of VXLAN as a compromise

- caution: MTU, Frame sizes!
- caution: latency

Implementing NF: Network Infrastructure

Compute
Network
Platform
Deployment

EVPN-VXLAN setup in 3 parts

- vxlan enabled in BGP EVPN configuration
  - enabled reflector mode as well to minimize # of sessions/configs

- general vxlan configuration
  - vxlan port
  - tcam profile changes

- then, for each Function
  - vni entry in the adjacent router

Implementing NF: Platform

Compute
Network
Platform
Deployment

Implementing NF: Platform

Compute
Network
Platform
Deployment

Provisioning Kubernetes

- Ansible installs a base kubernetes setup in a private lan
- haproxy exposes kube APIs on different management networks
- All else deployed via Kubernetes

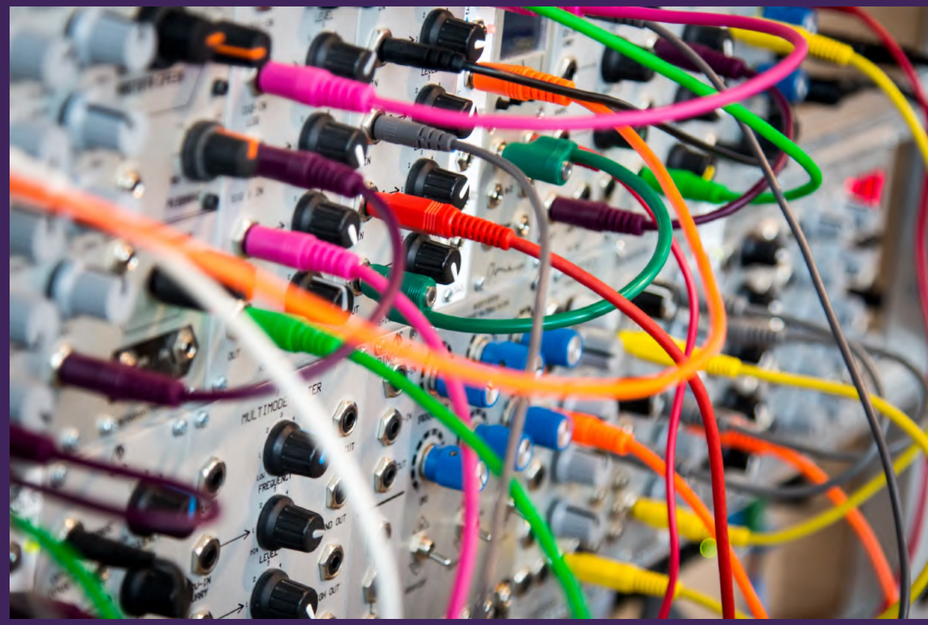
but… we broke host firewalls!

- Had ferm
- Tried ferm, ufw, firewalld, iptables-persistent. All break with kubernetes

Rolled our own firewall mini daemon, that plays well with kubernetes chains

Implementing NF: Platform

Connect to Inter.link's network in a Kubernetes friendly way



something needs to:

1. create vxlans (? CNI)
2. connect them (bridge CNI)
3. to container interfaces (multus CNI)

- Existing vxlan CNI implementations build fabrics between pods (think: east-west)
- We need pod-to-router (think: north-south)

...Rolled out our own vxlan CNI , creates vxlans, passes them to bridge, plays well when stacked

Implementing NF: Platform

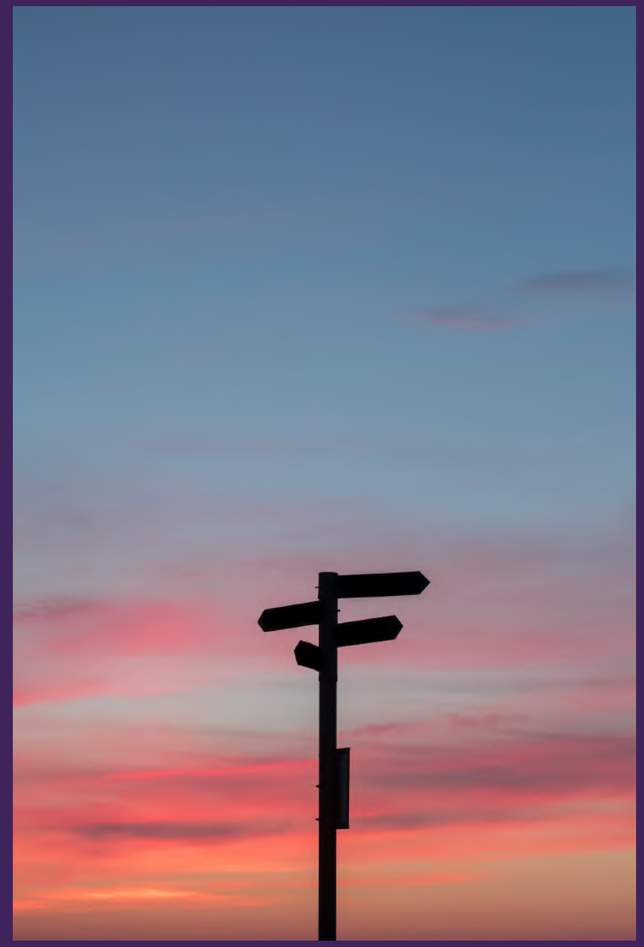
Connect to Inter.link's network in a Kubernetes friendly way

something needs to:

1. advertise the pods into the network (a BGP daemon)

- BIRD, FRR, GoBGP and a few others were evaluated
- only FRR configures the host forwarding (via zebra)

  → FRR as a daemonset: 2 sessions, ~20 lines of base config
- had to write custom instrumentation for conf.d support during reconfiguration



Implementing NF: Deployment

Implementing NF: Deployment

Compute
Network
Platform
Deployment

Provision Pipeline

order → service provisioning → setup (allocation, contract, LoA etc.)

→ peeringdb, IRR queries (if applicable)

→ network metadata committed→configuration produced → network provisioning

→ functions metadata committed → configuration produced→ functions provisioning

How?

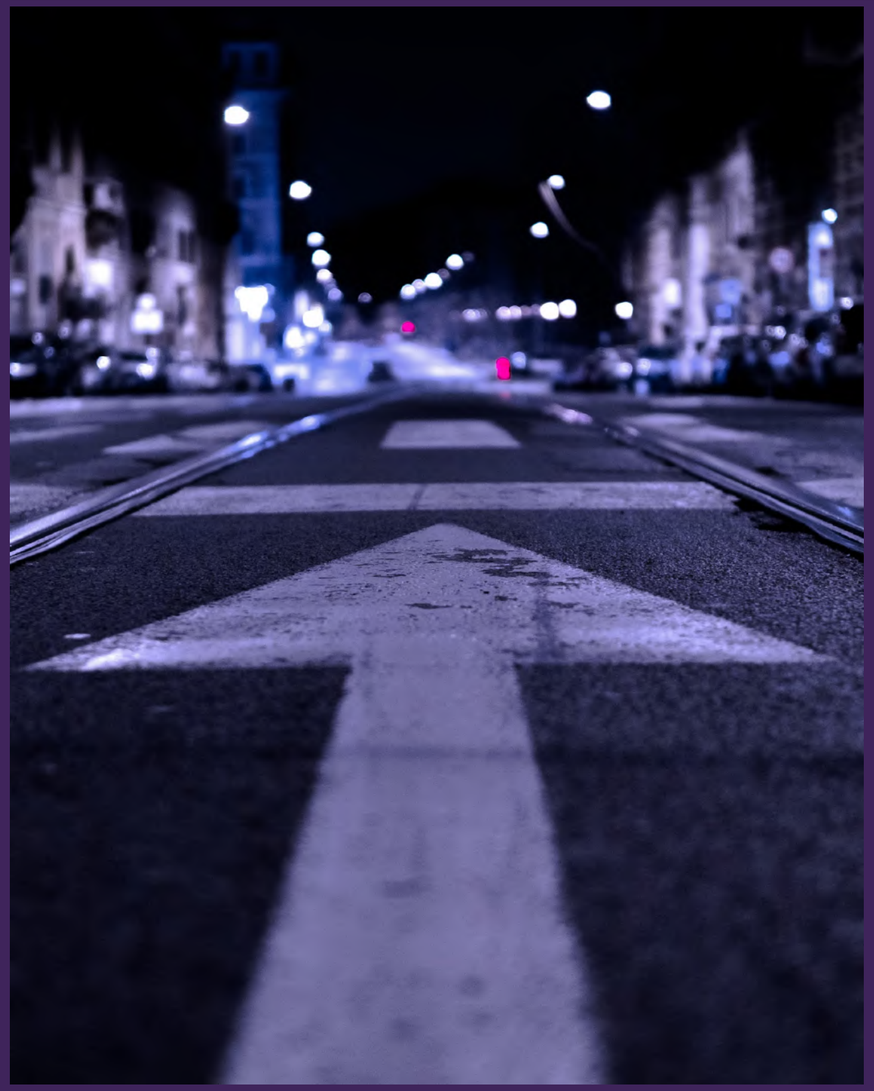
→ Database for functions' metadata

→ ArgoCD and custom argocd plugin

→ nfv metadata to argocd apps to kubernetes deployments

→ Network pipeline continues to configure the network side

Network Functions: Going Forward



Compute
Network
Platform
Deployment

Product:

- Make more functions available to Customers
- Enable higher throughput functions
- Enable Customers to use their own functions on their networks

Tech:

- Kubernetes aware of network resources when allocating
- Kubernetes Controller for FRR
- Revisit BIRD when EVPN is there

Questions?

costas@inter.link