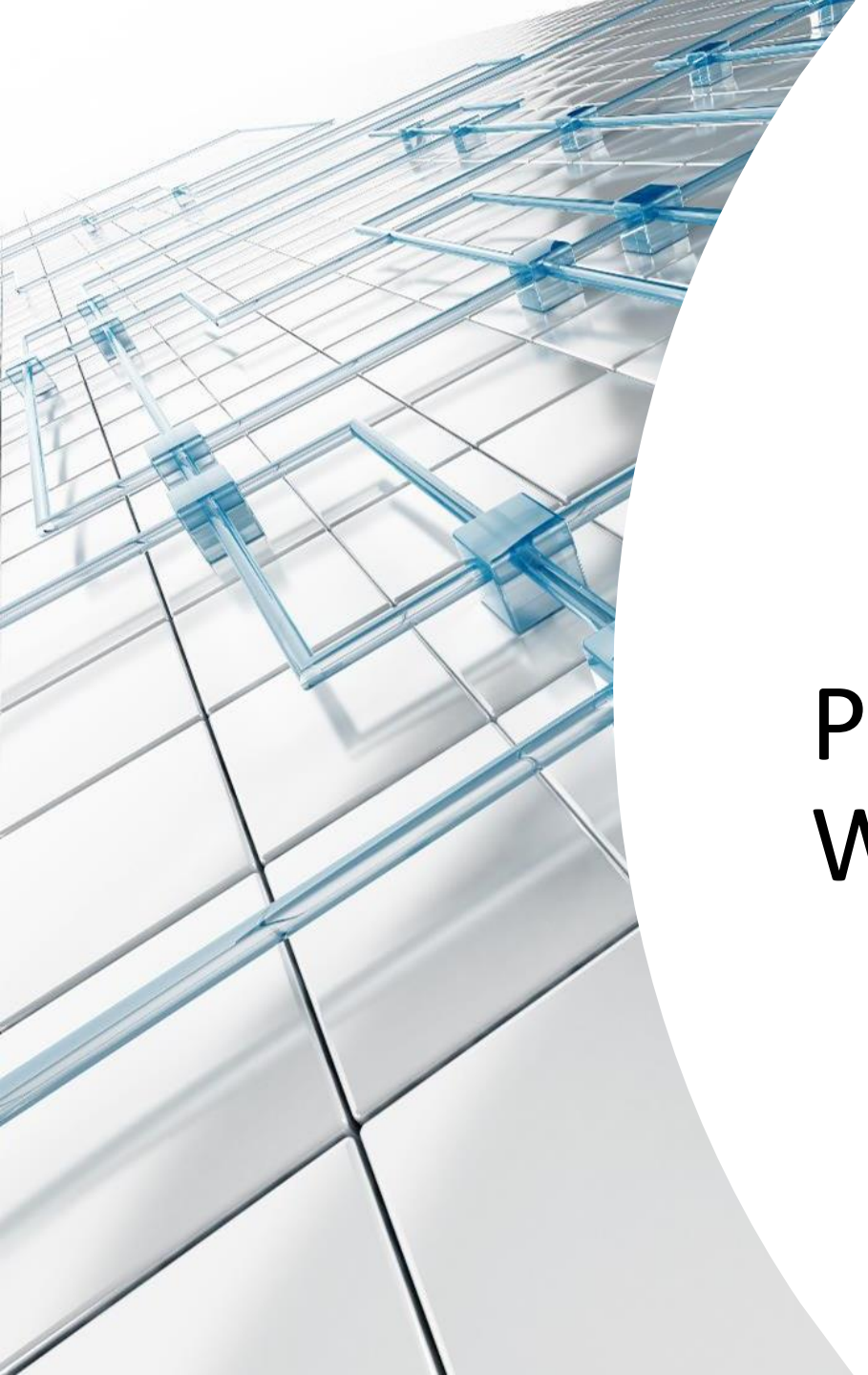
TeraFlow SDN by ETSI

Hackfest #5: Getting Started with TeraFlowSDN and ContainerLab

Lluís Gifre and Ricard Vilalta (CTTC)

Agenda

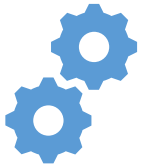
- 14:00 14:10 Welcome & Logistics
- 14:10 - 15:20 Part 1
  - What are new SDN controllers?
  - Introduction to control and management protocols and data models
  - Introduction to ContainerLab
  - TFS Deployment
  - Onboarding devices
- 15:20 - 15:40 Coffee Break & Group Picture (lobby)
- 15:40 - 17:00 Part 2
  - What are L3VPNs?
  - L3VPN Service Establishment (from NBI with L3SM)
  - Network Monitoring
  - What is hot on TFS community?
  - Contributing & Joining TFS
  - Wrap-up
- 17:00 Networking drink (lobby)



TeraFlow SDN

PART I
What are new SDN controllers?

The need for a Network Automation Framework



Network Operators need AUTOMATION

AUTOMATION means SOFTWARIZATION

NETWORKS are complex. Need to handle multiple equipment and domains

Specific domain-solutions to automate the network

TeraFlowSDN develops an open source cloud native SDN Controller enabling smart connectivity services for future networks beyond 5G

ETSI TeraFlowSDN: Who's Who?

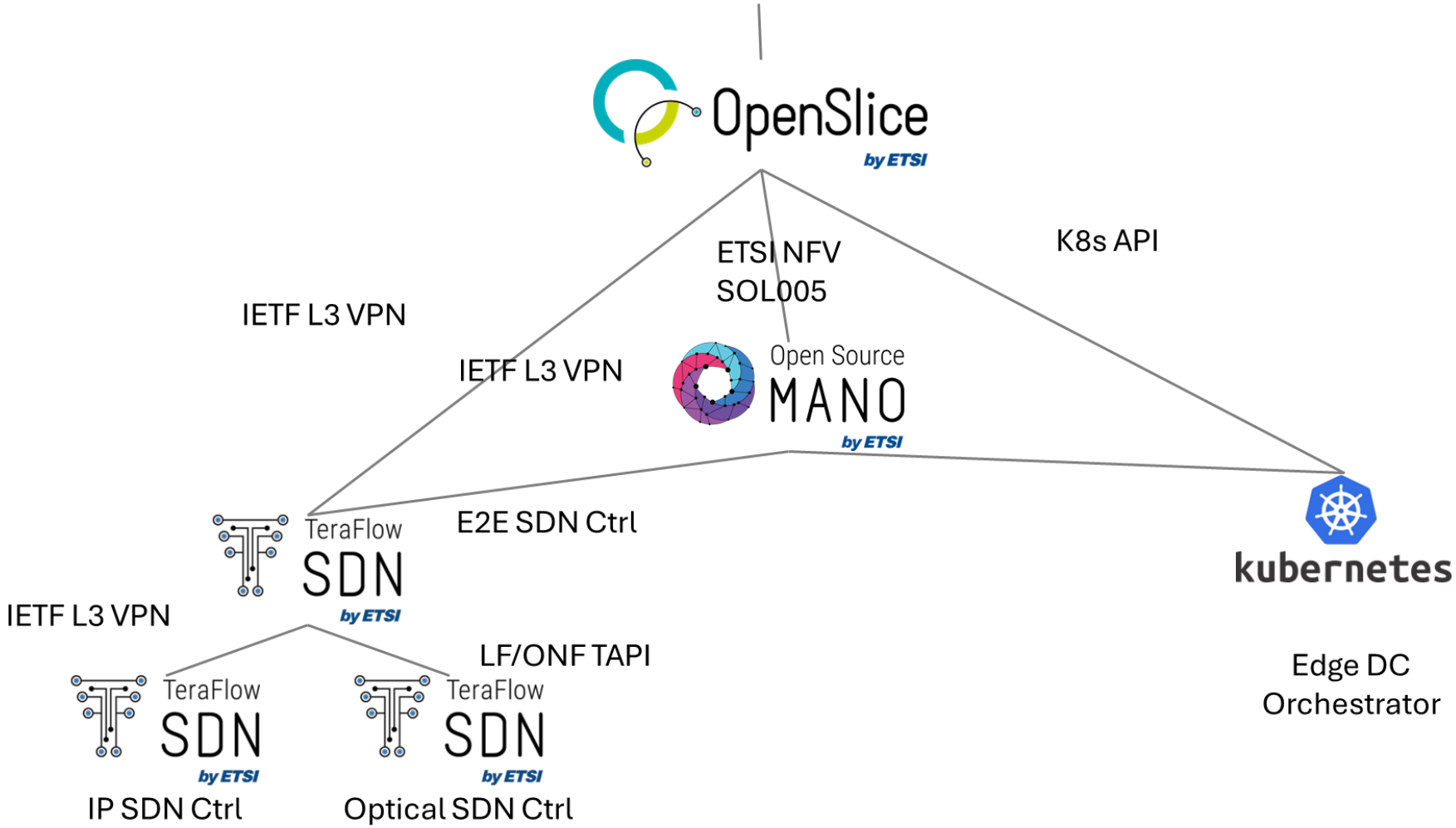
TeraFlow SDN

LEADERSHIP

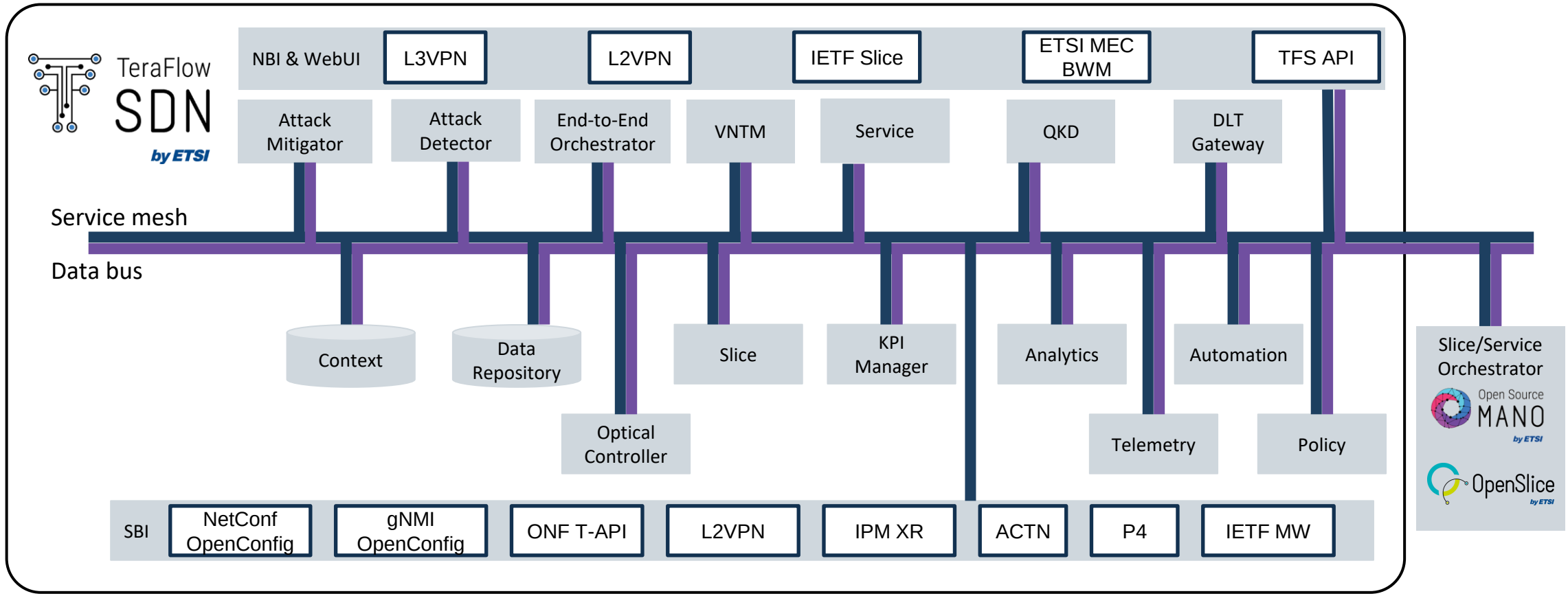
TECHNICAL STEERING

MEMBERS AND PARTICIPANTS

End-to-End Network Orchestration (TFS perspective)



Architecture for Release 4



Release 4 available!

TeraFlowSDN Release 4 Highlights

- Quantum Key Distribution (QKD): Introduces a quantum-ready network topology, efficient routing, device management, and a user-friendly QKD management interface to enhance secure communication.
- End-to-End Automation & Monitoring: New components (KPI Manager, Telemetry, Analytics) and an Automation module enable advanced network automation, data analysis, and ZSM-aligned monitoring for 5G/6G.
- Network Management: Supports logical inventory retrieval (ACLs, interfaces, routing policies) via NETCONF/OpenConfig, IETF Inventory model, and network slice integration for comprehensive network views.
- Optical Networks: Lifecycle management for media channels, automated OpenConfig discovery, and efficient topology synchronization with multi-band slot representation.
- Security & Blockchain: Upgraded to Hyperledger Fabric v2.4+ for blockchain operations, asset management, and Kubernetes deployment.
- Additional Enhancements: New NBI for ACLs, QoS profile integration with LF CAMARA API, and improved database handling for efficient service management.

Interfaces Supported

- NBIs (1):
  - TFS API
  - ETSI MEC Bandwidth Management API
  - IETF L2-VPN Service Delivery [RFC8466]
  - IETF L3-VPN Service Delivery [RFC8299]
  - IETF Network Topology [RFC8345]
  - IETF Network Slice Service [draft]

- SBIs (2):
  - Emulated
  - NetConf/OpenConfig
  - gNMI/OpenConfig
  - ONF Transport API
  - P4
  - Infinera IPM XR
  - IETF L2-VPN Service Delivery [RFC8466]
  - IETF ACTN (basic support)
  - Optical TFS
  - MicroWave [ONF TR-352]

(1) https://labs.etsi.org/rep/tfs/controller/-/wikis/6.-Supported-NBIs
(2) https://labs.etsi.org/rep/tfs/controller/-/wikis/5.-Supported-SBIs-and-Network-Elements

Collaboration with other communities

- Open-source software

- Standard Defining Organizations

ISG MEC
ISG F5G
ISG ZSM

TEAS (ACTN)
CCAMP

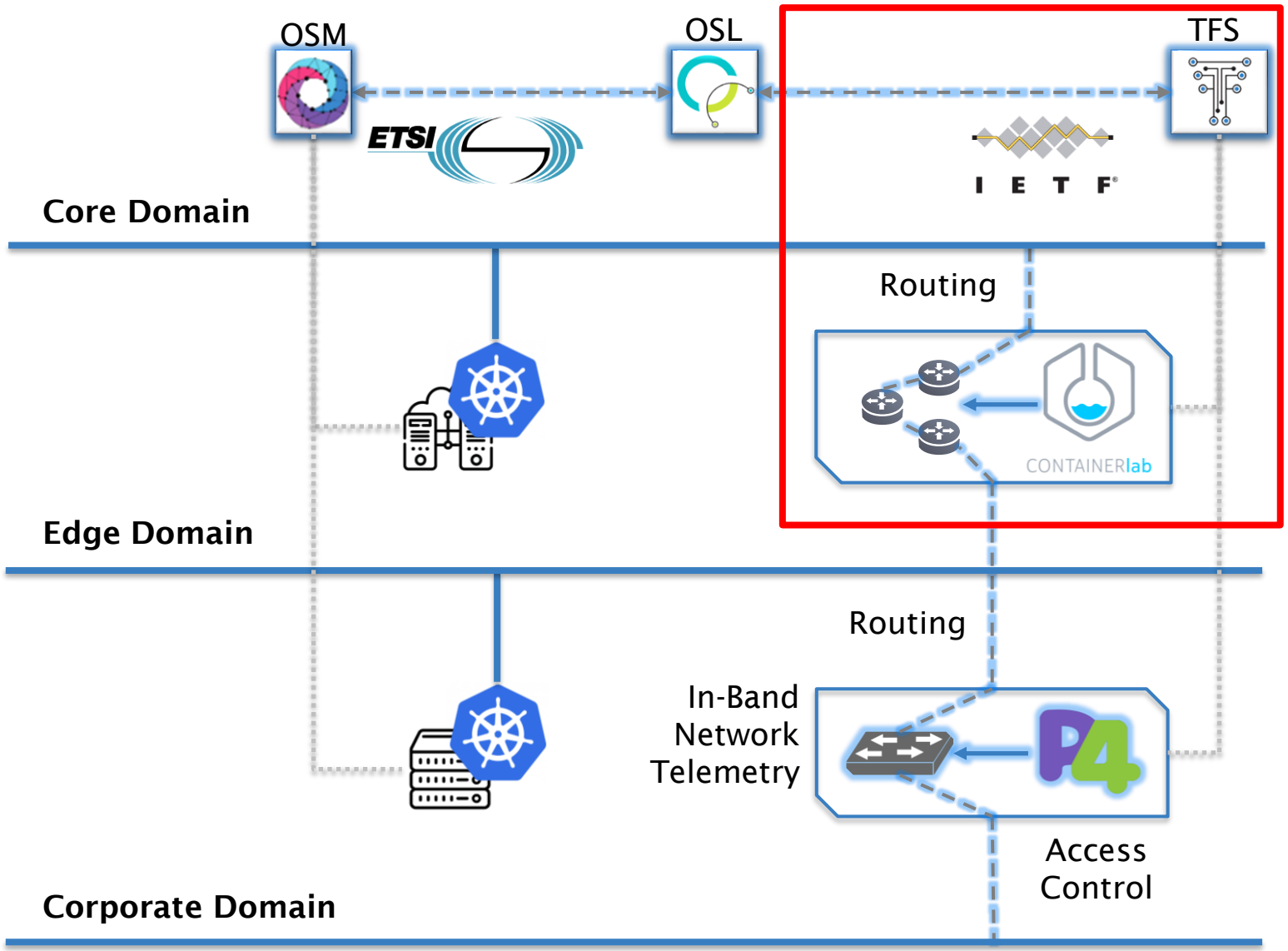
Transport API
P4

- Industry Organizations

TeraFlow SDN by ETSI

Are you ready?

A bit of context for this Hackfest…



You will run a subset of that demo ☺ !!

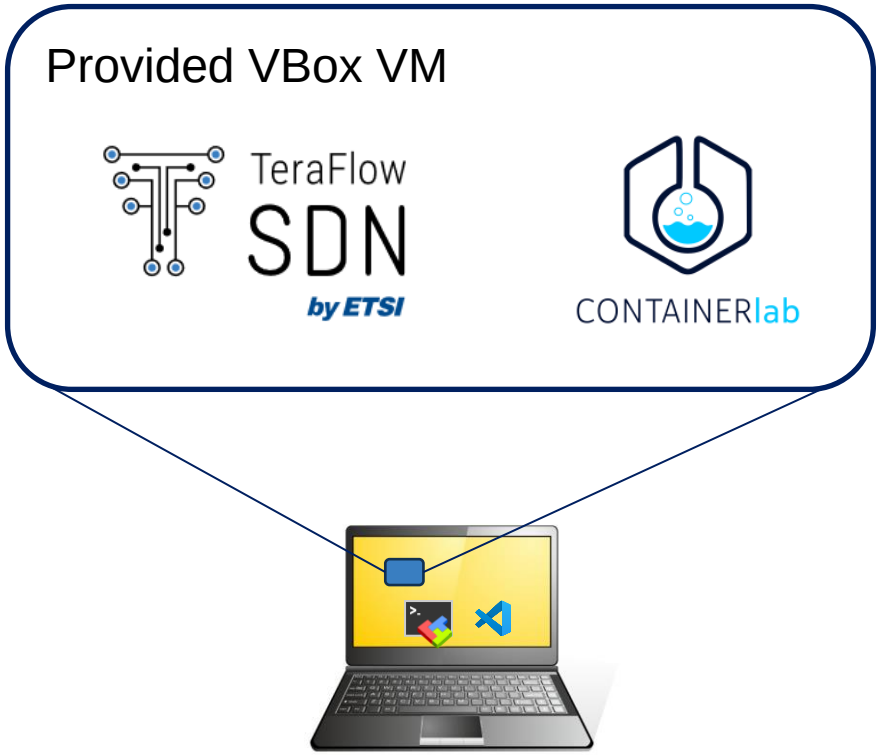
Open Source for Telco-Cloud:
An ETSI SDG-based solution to facilitate zero-touch, multi-slice 5G deployments across the cloud-edge continuum

Kostis Trantzas, Lluis Gifre, Georgios P. Katsikas, Pantelis Malekas, Christos Tranoris, Ricard Vilalta, Jakub Górczyński, Tomasz Osiński

This work has been partially funded by the Horizon Europe Project ACROSS (grant agreement No. 101097122)

Work environment

VM credentials:
- **User: tfs**
- **Pass: tfs123**



NOTE: If your laptop cannot support the VM, ask ETSI staff for a VM ☺.

VM requirements:
- VirtualBox 7.0 or newer
- 4 vCPU
- 8 GB RAM
- 60 GB disk

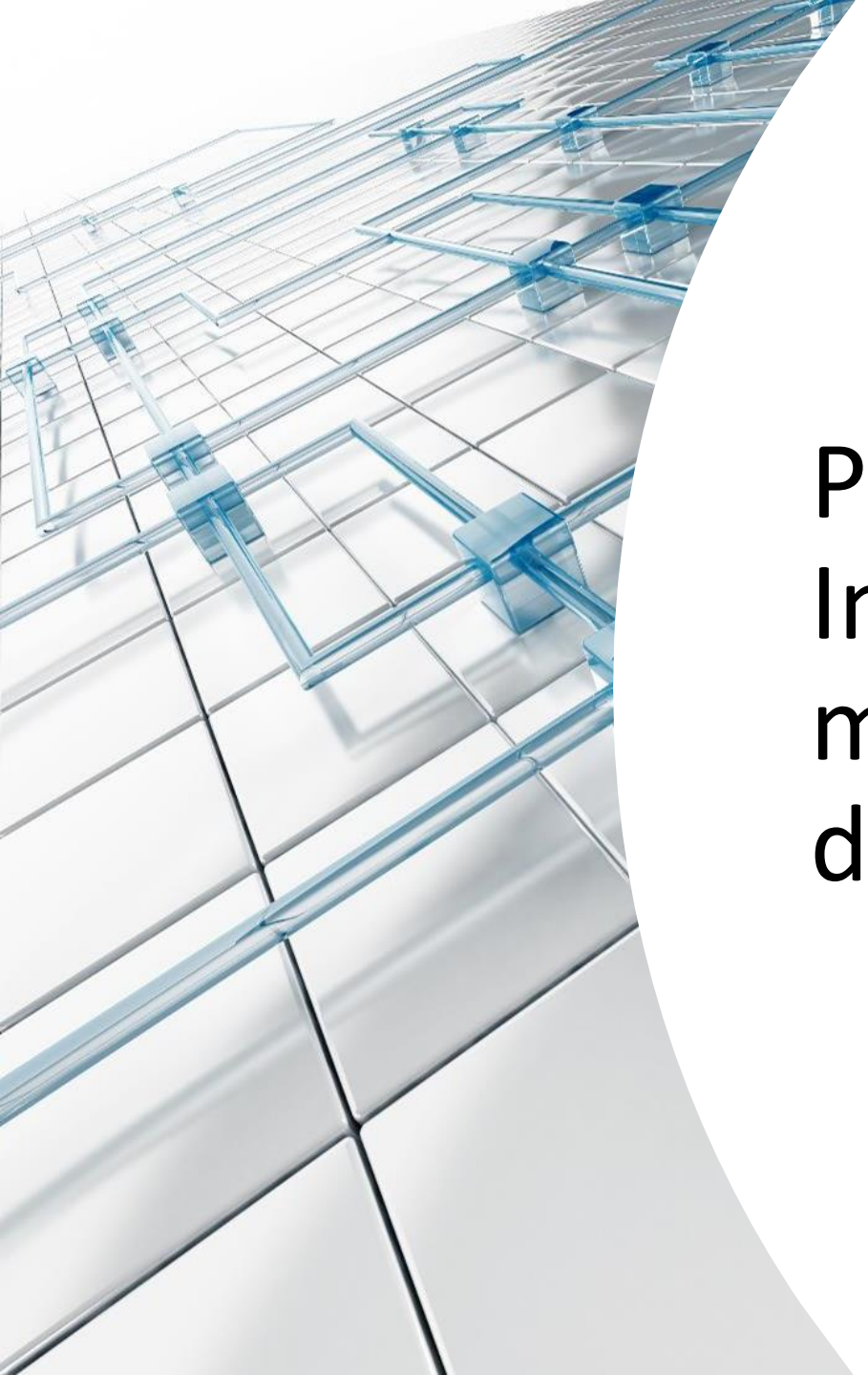
VM Networking settings:
- Connected to NAT and forward:
  - SSH port (22/tcp)
  - HTTP port (80/tcp)
  - CLab HTTP port (50080/tcp)

Install on the host:
- MobaXterm or other SSH client
- VSCode
- Remote devel Ext. for VSCode

Hackfest Materials

For a perfect hands-on experience, a VirtualBox VM image is needed. Please download the hackfest VM from the link below and make sure the VM is installed and loads/starts up on your PC before the Hackfest:

- https://www.dropbox.com/scl/fi/6anbily0ny8ovufiz47sp/TFS-HF5-VM.rar?rlkey=lrd68d2wyoz4y8qybxbhr92gj&st=qsp5mqlq&dl=0 (~11GB)
  - Download and unzip the RAR file.
- Inside the VM, you have all commands used along the tutorial:
  - `/home/tfs/tfs-ctl/hackfest5/README.md`
- Also available in a beautified way:
  - https://labs.etsi.org/rep/tfs/controller/-/blob/feat/hackfest5/hackfest5/README.md
- Please update to the latest version of the hackfest material from ETSI GitLab repository:
  - `cd ~/tfs-ctrl`
  - `git checkout feat/hackfest5`
  - `git pull`



TeraFlow SDN

PART I
Introduction to control and management protocols and data models

YANG Language (I)

YANG: data modeling language, initially conceived to model configuration and state data for network devices.

- Significant adoption across frameworks and open-source projects.
- Notable effort across SDOs to model constructs (e.g., topologies, devices, protocols, etc.)
- An SDN controller may expose NBI and SBI interfaces formatted according to standardized YANG schemas.
- Internal system data representations can also be formatted according to standardized YANG schemas.

```
module topology {
  namespace "urn:topology";
  prefix "topology";
  organization "CTTC";
  contact "ricard.vilalta@cttc.es";
  description "Basic example of network topology";

  revision "2018-08-24" {
    description "Basic example of network topology";
    reference "";
  }

  grouping port {
    leaf port-id { type string; }
  }

  grouping node {
    leaf node-id { type string; }
    list port { key "port-id"; uses port; }
  }
…
```

```
…
  grouping link {
    leaf link-id { type string; }
    leaf source-node { type leafref { path "/topology/node/node-id"; } }
    leaf target-node { type leafref { path "/topology/node/node-id"; } }
    leaf source-port { type leafref { path "/topology/node/port/port-id"; } }
    leaf target-port { type leafref { path "/topology/node/port/port-id"; } }
  }

  grouping topology {
    list node { key "node-id"; uses node; }
    list link { key "link-id"; uses link; }
  }

  container topology {
    uses topology;
  }
}
```

YANG Language (II)

Tree-formatted view of the data model:

```
module: topology
    +--rw topology
       +--rw node* [node-id]
       |  +--rw node-id    string
       |  +--rw port* [port-id]
       |     +--rw port-id                 string
       |     +--rw layer-protocol-name?    layer-protocol-name
       +--rw link* [link-id]
          +--rw link-id         string
          +--rw source-node?    -> /topology/node/node-id
          +--rw target-node?    -> /topology/node/node-id
          +--rw source-port?    -> /topology/node/port/port-id
          +--rw target-port?    -> /topology/node/port/port-id
```

Example topology message encoded in JSON:

```
<?xml version='1.0' encoding='UTF-8'?>
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<topology xmlns="urn:topology">
  <node>
    <node-id>router-1</node-id>
    <port><port-id>port-1</port-id></port>
    <port><port-id>port-2</port-id></port>
  </node>
  <node>
    <node-id>router-2</node-id>
    <port><port-id>port-1</port-id></port>
    <port><port-id>port-3</port-id></port>
  </node>
  <link>
    <link-id>router-1/port-1==>router-2/port-2</link-id>
    <source-node>router-1</source-node>
    <target-node>router-2</target-node>
    <source-port>port-1</source-port>
    <target-port>port-2</target-port>
  </link>
</topology>
</data>
```

NETCONF Protocol

Offers primitives to view and manipulate data, providing a suitable encoding as defined by the data-model.

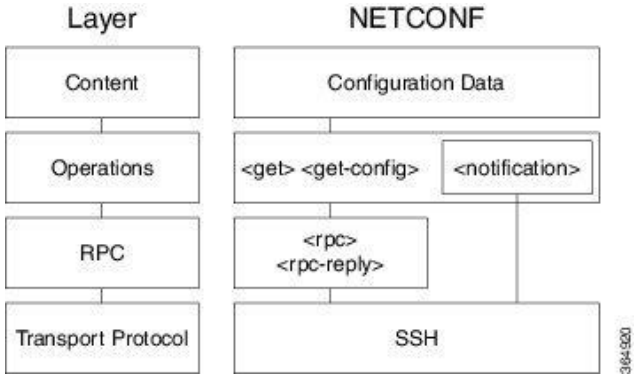
Enables remote access to a device and provides rules to enable multiple clients to manage a device's configuration.

- NETCONF-enabled devices *include a NETCONF server*, while management applications *include a NETCONF client.*
- The device's Command Line Interfaces (CLIs) can be a wrapped around a NETCONF client.

It is usually based on the exchange of XML-/JSON-encoded RPC messages over a secure (commonly Secure Shell, SSH) connection.

NETCONF Layering :

- Configuration or notification data (Content Layer) that is exchanged between a client and a server,
- Operations layer (e.g. <get-config>, <edit-config>)
- Message layer for RPC messages or notifications
- Secure Transport.

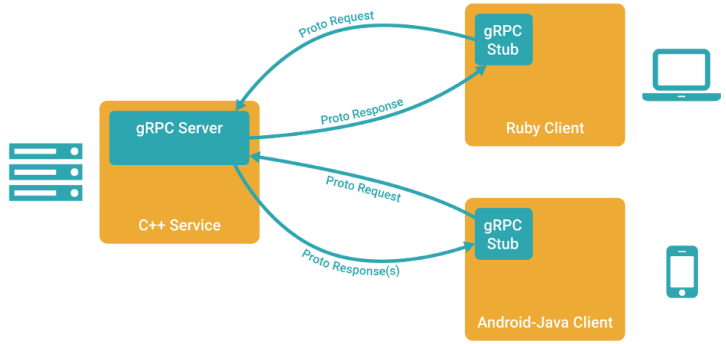


Operation	Description
<get>	Retrieve running configuration and device state information
<get-config>	Retrieve all or part of a specified configuration datastore
<edit-config>	Edit a configuration datastore by creating, deleting, merging or replacing content
<copy-config>	Copy an entire configuration datastore to another configuration datastore
<delete-config>	Delete a configuration datastore
<lock>	Lock an entire configuration datastore of a device
<unlock>	Release a configuration datastore lock previously obtained with the <lock> operation
<close-session>	Request graceful termination of a NETCONF session

gRPC & Protocol Buffers framework

gRPC: A high performance, general purpose, feature-rich Remote Procedure Call framework running on top of HTTP/2 requests

- Part of Cloud Native Computing Foundation
- Open-sourced version of Stubby RPC used in Google



Protocol Buffers: Interface Definition Language (IDL) for gRPC

- Describe once and generate interfaces for any programming language.
- Data model provides structure of the request and reply.
- Data is encoded in binary and compressed wire format.

```
syntax = "proto3";
package search;

service Google {
  rpc Search(Request) returns (Result) {}
}

message Request {
  string query = 1;
}

message Result {
  string title   = 1;
  string url     = 2;
  string snippet = 3;
}
```

Source: https://grpc.io/

gNMI Protocol

- gRPC Network Management Interface (gNMI) https://github.com/openconfig/gnmi
  - gRPC-based single service for the management of target devices.
    - Retrieval and modification of configuration data as well as subscription to telemetry data streams.
    - Offers a feature-right and very efficient alternative to NETCONF, RESTCONF, etc.
  - This gNMI is described using Protobuf: https://github.com/openconfig/gnmi/blob/master/proto/gnmi/gnmi.proto
  - The data can be encoded in JSON, XML, Protobuf, etc. JSON is commonly used.

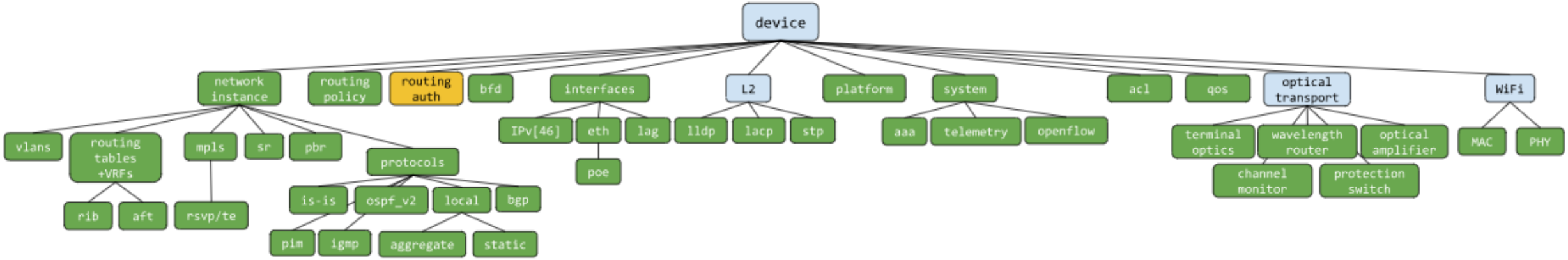
```
service gNMI {
  rpc Capabilities(CapabilityRequest      ) returns (CapabilityResponse      );
  rpc Get         (GetRequest             ) returns (GetResponse             );
  rpc Set         (SetRequest             ) returns (SetResponse             );
  rpc Subscribe   (stream SubscribeRequest) returns (stream SubscribeResponse);
}
```

```
message GetResponse {
  repeated Notification notification = 1;
  …
}
```

```
message GetRequest {
  Path prefix = 1;
  repeated Path path = 2;
  enum DataType {
    ALL = 0;
    CONFIG = 1;
    STATE = 2;
    OPERATIONAL = 3;
  }
  DataType type = 3;
  Encoding encoding = 5;
  …
}
```

OpenConfig

- Data models for configuration and operational state, written in YANG
  - Initial focus: device data for switching, routing, and transport
  - Development priorities driven by operator requirements
  - Technical engagement with major vendors to deliver native implementations



https://github.com/openconfig/public/

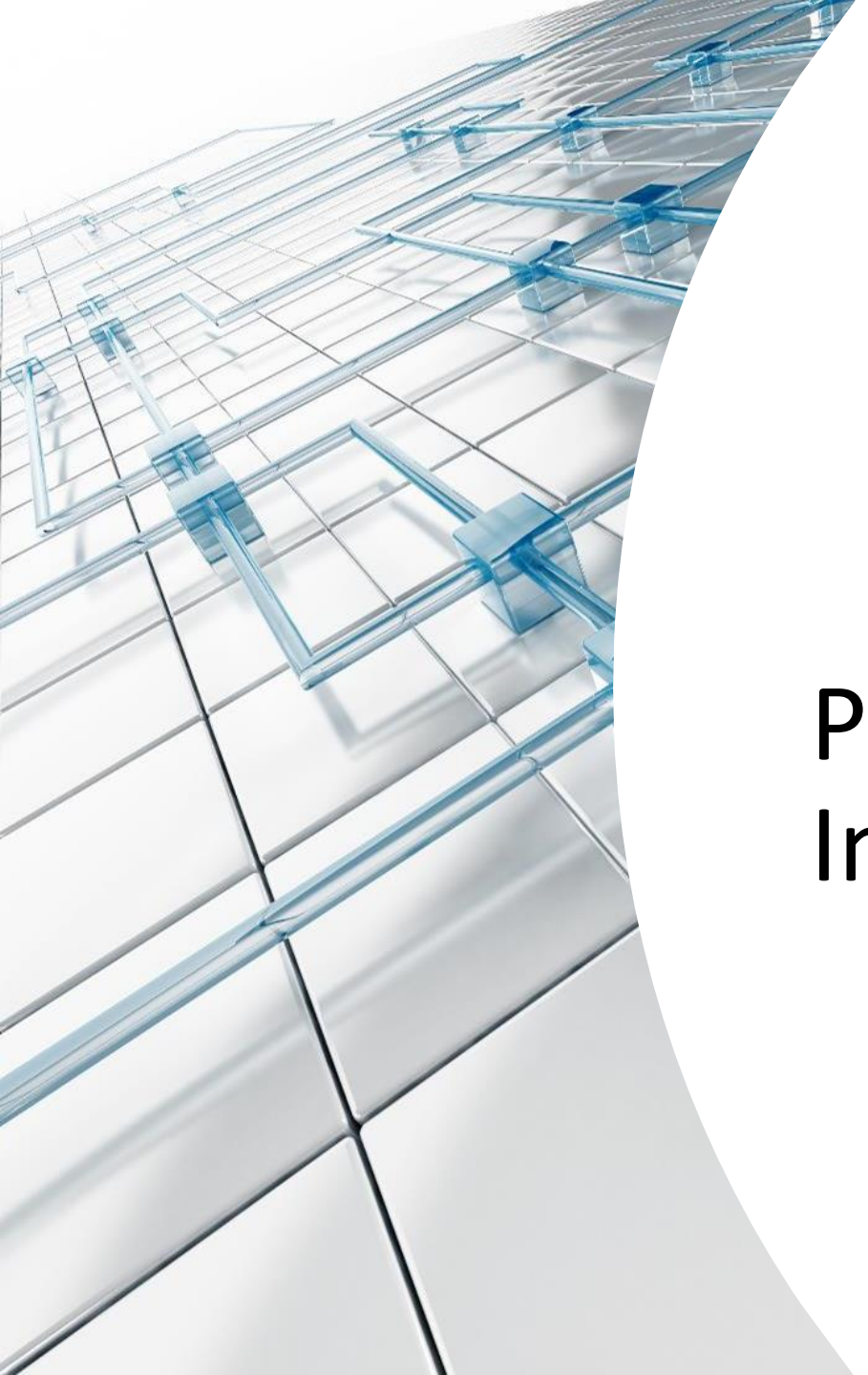
OpenConfig L3 data models – Interfaces

TeraFlow SDN

```
module: openconfig-interfaces
  +--rw interfaces
     +--rw interface* [name]
        +--rw name              -> ../config/name
        +--rw config
        |  +--rw name?            string
        |  +--rw type             identityref
        |  +--rw mtu?             uint16
        |  +--rw loopback-mode?   boolean
        |  +--rw description?     string
        |  +--rw enabled?         boolean
        +--ro state
        |  +--ro name?            string
        |  +--ro type             identityref
        |  +--ro admin-status     enumeration
        |  +--ro oper-status      enumeration
        |  ...
        |  +--ro counters
        |     +--ro in-octets?            oc-yang:counter64
        |     +--ro in-pkts?              oc-yang:counter64
        |     +--ro out-octets?           oc-yang:counter64
        |     +--ro out-pkts?             oc-yang:counter64
        |     ...
        |  ...
        |
        +--rw subinterfaces
           +--rw subinterface* [index]
           +--rw index     -> ../config/index
           +--rw config
           |  +--rw index?         uint32
           |  +--rw description?   string
           |  +--rw enabled?       boolean
           +--ro state
              ...
```

```
module: openconfig-vlan
  augment /oc-if:interfaces/oc-if:interface/oc-if:subinterfaces/oc-if:subinterface:
    +--rw vlan
       +--rw config
       |  x--rw vlan-id?   union
       +--ro state
       |  x--ro vlan-id?   union
       +--rw match
       |  +--rw single-tagged
       |  |  +--rw config
       |  |  |  +--rw vlan-id?   oc-vlan-types:vlan-id
       |  |  +--ro state
       |  |     +--ro vlan-id?   oc-vlan-types:vlan-id
       |  ...
       ...
```

```
module: openconfig-if-ip
  augment /oc-if:interfaces/oc-if:interface/oc-if:subinterfaces/oc-if:subinterface:
    +--rw ipv4
       +--rw addresses
       |  +--rw address* [ip]
       |     +--rw ip        -> ../config/ip
       |     +--rw config
       |     |  +--rw ip?              oc-inet:ipv4-address
       |     |  +--rw prefix-length?   uint8
       |     +--ro state
       |     |  +--ro ip?              oc-inet:ipv4-address
       |     |  +--ro prefix-length?   uint8
       |     |  +--ro origin?          ip-address-origin
       |     ...
       ...
```



TeraFlow SDN

PART I
Introduction to ContainerLab

ContainerLab

https://containerlab.dev/

Additional Details:
https://containerlab.dev/quickstart/

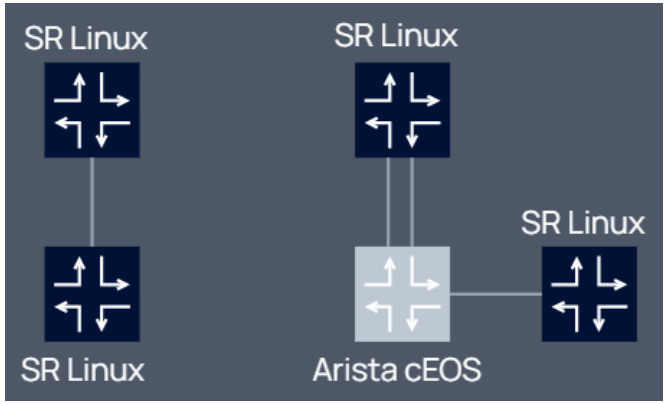
Network emulator supporting multiple virtualized Network Operating Systems

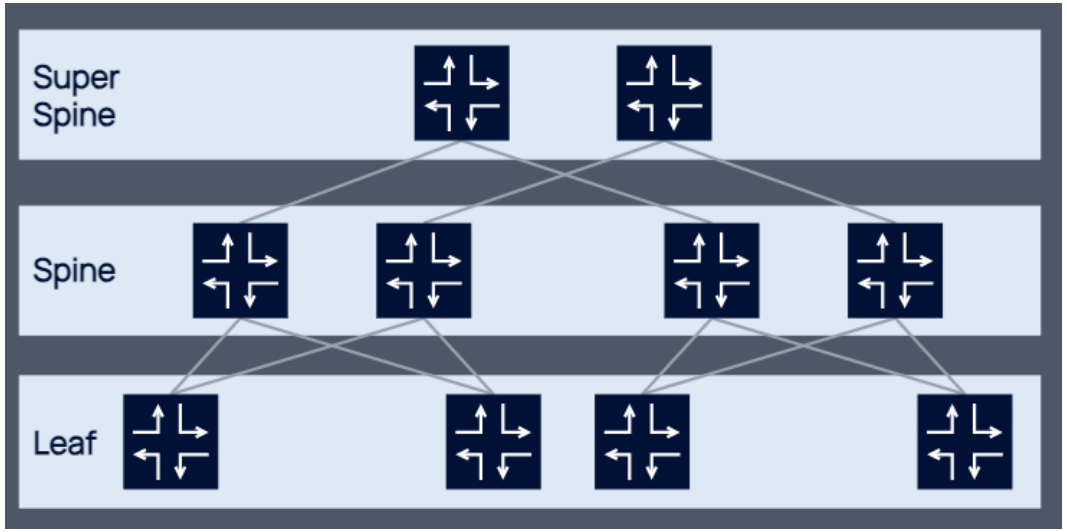
- Enables experts to experiment with NOSes on demand in user-defined topologies.
- CLI for orchestration and managing container-based networking labs.
  - Start containers, build virtual wiring between them, manage lifecycle of labs.
- Enables traffic captures on the virtual wires.
- Support many network device kinds (https://containerlab.dev/manual/kinds/)
- Many examples (https://containerlab.dev/lab-examples/lab-examples/)

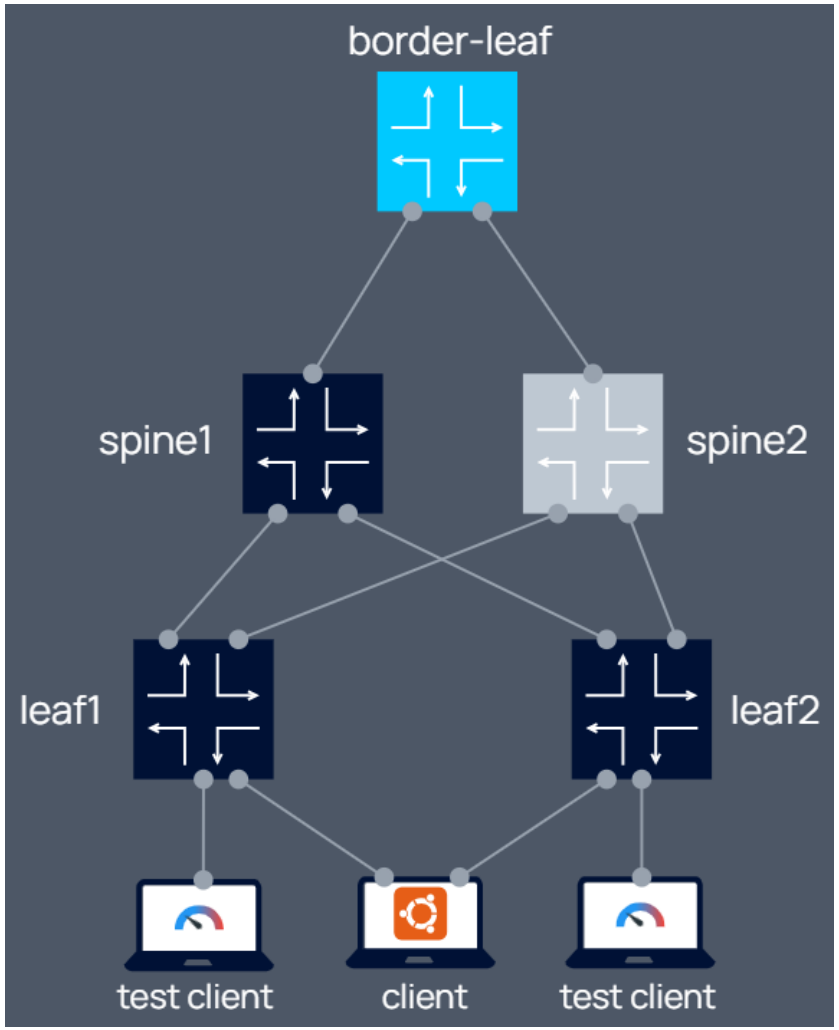
ContainerLab - Examples

CONTAINERlab

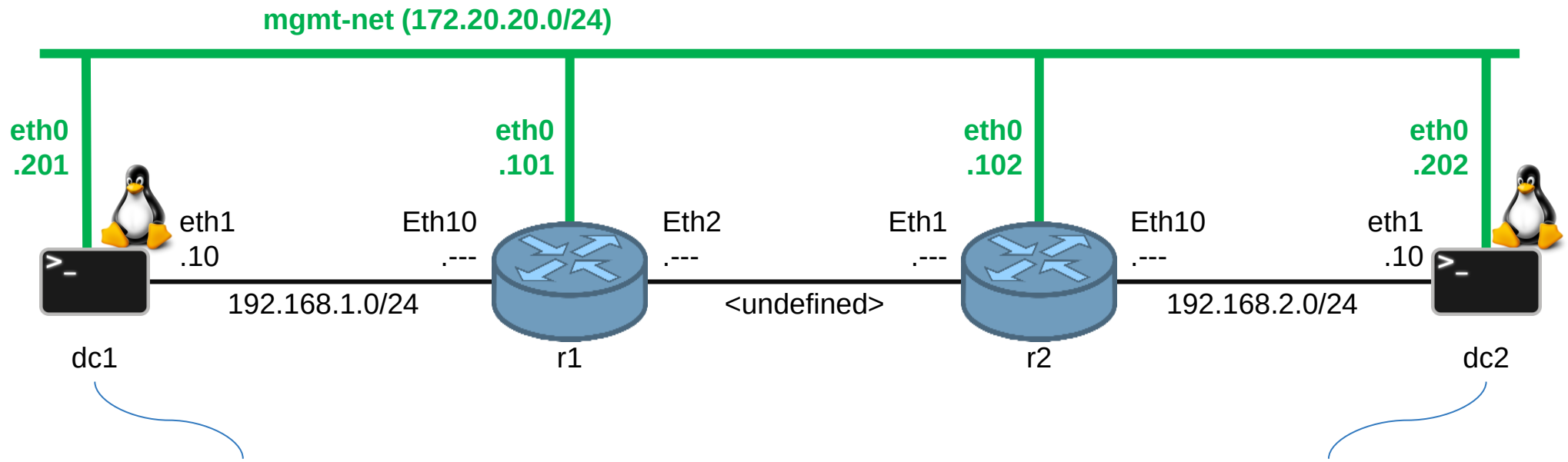
https://containerlab.dev/







Our ContainerLab Scenario




```
ip link set address 00:c1:ab:00:01:01 dev eth1
ip address add 192.168.1.10/24 dev eth1
ip route add 192.168.2.0/24 via 192.168.1.1
```

```
ip link set address 00:c1:ab:00:02:01 dev eth1
ip address add 192.168.2.10/24 dev eth1
ip route add 192.168.1.0/24 via 192.168.2.1
```

Pre-configured in the lab

Our ContainerLab Scenario

https://containerlab.dev/

Our Lab Descriptor:
~/tfs-ctrl/hackfest5/clab/hackfest5.clab.yml
https://labs.etsi.org/rep/tfs/controller/-/blob/feat/hackfest5/hackfest5/clab/hackfest5.clab.yml

```
name: hackfest5
mgmt:
  network: mgmt-net
  ipv4-subnet: 172.20.20.0/24

topology:
  kinds:
    arista_ceos:
      kind: arista_ceos
      image: ceos:4.32.2F
    linux:
      kind: linux
      image: ghcr.io/hellt/network-multitool:latest

  nodes:
    r1:
      kind: arista_ceos
      mgmt-ipv4: 172.20.20.101

    r2:
      kind: arista_ceos
      mgmt-ipv4: 172.20.20.102

…
```

```
…
    dc1:
      kind: linux
      mgmt-ipv4: 172.20.20.201
      exec:
        - ip link set address 00:c1:ab:00:01:01 dev eth1
        - ip address add 192.168.1.10/24 dev eth1
        - ip route add 192.168.2.0/24 via 192.168.1.1

    dc2:
      kind: linux
      mgmt-ipv4: 172.20.20.202
      exec:
        - ip link set address 00:c1:ab:00:02:01 dev eth1
        - ip address add 192.168.2.10/24 dev eth1
        - ip route add 192.168.1.0/24 via 192.168.2.1

  links:
    - endpoints: ["r1:eth2", "r2:eth1"]
    - endpoints: ["r1:eth10", "dc1:eth1"]
    - endpoints: ["r2:eth10", "dc2:eth1"]
```

Our ContainerLab Scenario

https://containerlab.dev/

Import the cEOS image [already done]

```
$ docker import ~/tfs-ctrl/hackfest5/images/arista/cEOS64-lab-4.31.5M.tar ceos:4.31.5M
```

Check container images are available

```
$ docker images | grep -E "ceos|multitool"
    REPOSITORY                          TAG       IMAGE ID       CREATED          SIZE
    ceos                                4.31.5M   da2e53457af6   7 minutes ago    2.47GB
    ghcr.io/hellt/network-multitool     latest    ee228f411287   3 years ago      270MB
```

Start the lab deployment

```
$ ~/tfs-ctrl/hackfest5/clab-deploy.sh
    INFO[0000] Containerlab v0.59.0 started
    INFO[0000] Parsing & checking topology file: hackfest5.clab.yml
    …
```

```
+---+--------------------+--------------+----------------------------------------+-------------+---------+------------------+--------------+
| # |        Name        | Container ID |                 Image                  |    Kind     |  State  |   IPv4 Address   | IPv6 Address |
+---+--------------------+--------------+----------------------------------------+-------------+---------+------------------+--------------+
| 1 | clab-hackfest5-dc1 | 942d0e529844 | ghcr.io/hellt/network-multitool:latest | linux       | running | 172.20.20.201/24 | N/A          |
| 2 | clab-hackfest5-dc2 | 12aafbb15948 | ghcr.io/hellt/network-multitool:latest | linux       | running | 172.20.20.202/24 | N/A          |
| 3 | clab-hackfest5-r1  | aaa749c9b054 | ceos:4.31.5M                           | arista_ceos | running | 172.20.20.101/24 | N/A          |
| 4 | clab-hackfest5-r2  | 6a48c800df65 | ceos:4.31.5M                           | arista_ceos | running | 172.20.20.102/24 | N/A          |
+---+--------------------+--------------+----------------------------------------+-------------+---------+------------------+--------------+
```

NOTE: ContainerLab adds entries to /etc/hosts, so you can resolve the nodes by IP address or by container name.

Our ContainerLab Scenario

CONTAINERlab

https://containerlab.dev/

Inspect the lab deployment

```
$ ~/tfs-ctrl/hackfest5/clab-inspect.sh
```

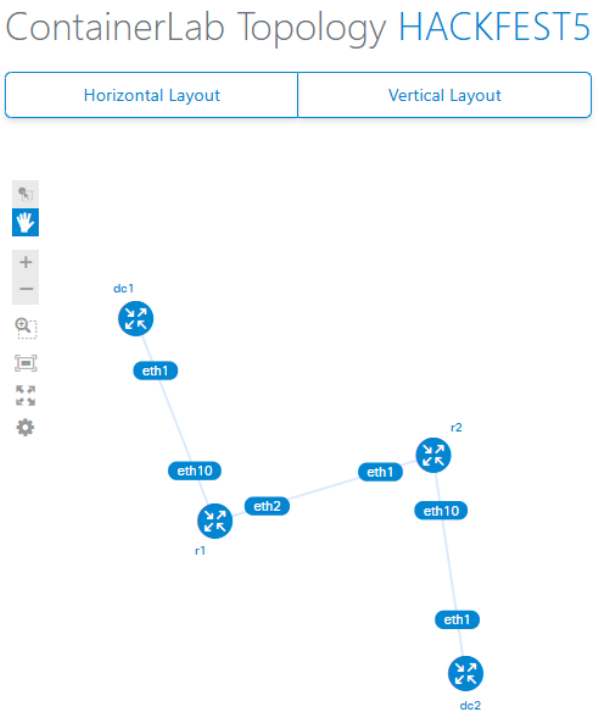
```
+---+--------------------+-------------+----------------------------------------+------------+---------+------------------+--------------+
| # |        Name        | Container ID |                 Image                  |    Kind    |  State  |   IPv4 Address   | IPv6 Address |
+---+--------------------+-------------+----------------------------------------+------------+---------+------------------+--------------+
| 1 | clab-hackfest5-dc1 | 942d0e529844 | ghcr.io/hellt/network-multitool:latest | linux      | running | 172.20.20.201/24 | N/A          |
| 2 | clab-hackfest5-dc2 | 12aafbb15948 | ghcr.io/hellt/network-multitool:latest | linux      | running | 172.20.20.202/24 | N/A          |
| 3 | clab-hackfest5-r1  | aaa749c9b054 | ceos:4.31.5M                           | arista_ceos | running | 172.20.20.101/24 | N/A          |
| 4 | clab-hackfest5-r2  | 6a48c800df65 | ceos:4.31.5M                           | arista_ceos | running | 172.20.20.102/24 | N/A          |
+---+--------------------+-------------+----------------------------------------+------------+---------+------------------+--------------+
```

Draw the lab's topology

```
$ ~/tfs-ctrl/hackfest5/clab-graph.sh
INFO[0000] Parsing & checking topology file: hackfest5.clab.yml
INFO[0000] Serving topology graph on http://0.0.0.0:50080
```

<leave it running…>

[Host] Browse: http://127.0.0.1:50080/



Our ContainerLab Scenario

https://containerlab.dev/

Connecting to the nodes CLI

```
$ ~/tfs-ctrl/hackfest5/clab-cli-r1.sh
   r1>enable
   r1#show running-config
   ! Command: show running-config
   ! device: r1 (cEOSLab, EOS-4.31.5M-38783521.4315M (engineering build))
   …
   r1#exit
```

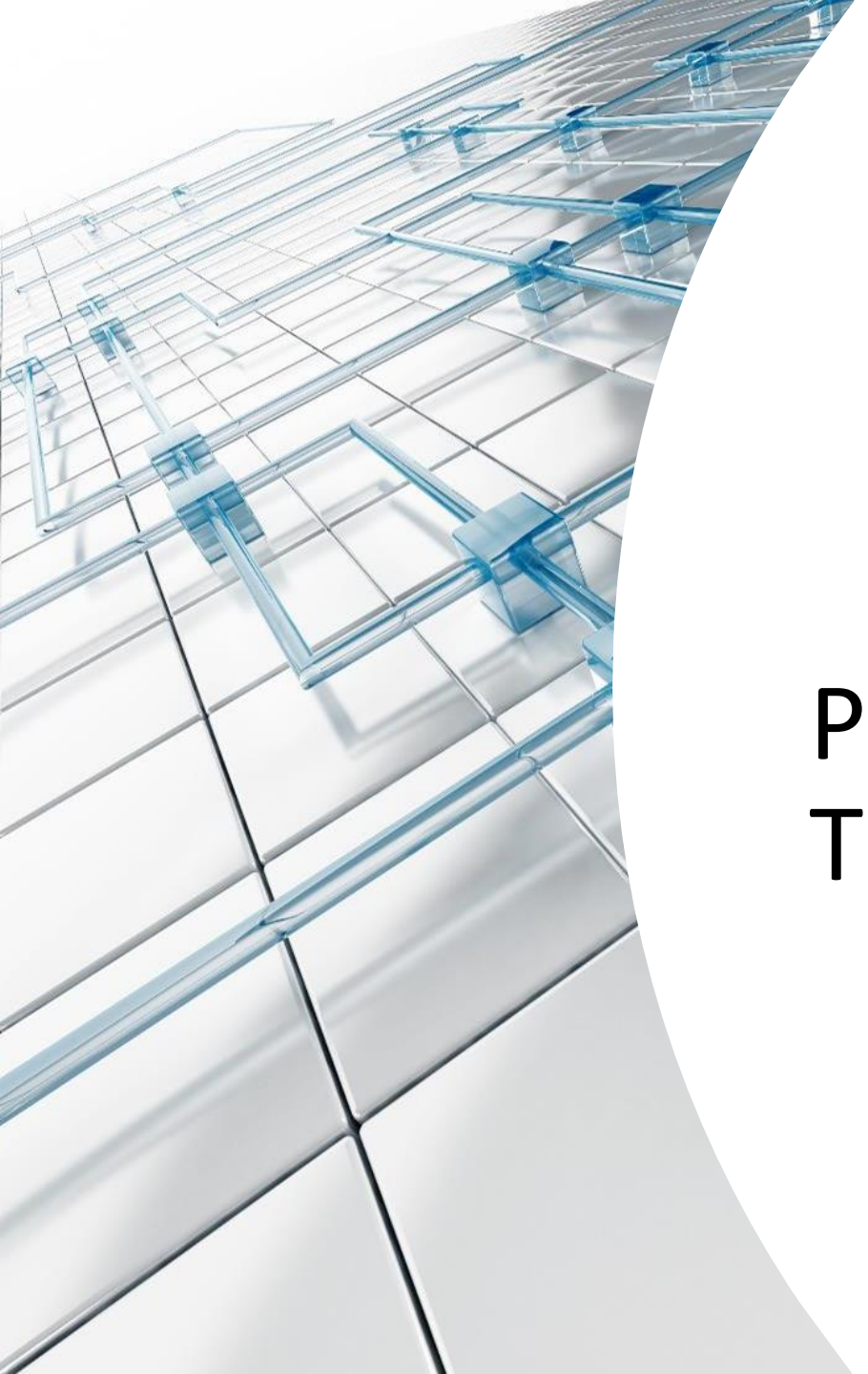
→ will give you access to router's CLI

```
$ ~/tfs-ctrl/hackfest5/clab-cli-dc1.sh
   bash-5.0# cat /etc/hostname
   dc1
```

→ will give you access to the client's bash

Destroying a lab [do not run it now ☺]

```
$ ~/tfs-ctrl/hackfest5/clab-destroy.sh
```



TeraFlow SDN

PART I
TFS Deployment

Deploy TeraFlowSDN controller

TeraFlowSDN controller runs a number of microservices on top of a Kubernetes-based environment. For development and demonstration purposes, we use MicroK8s v1.29.

- The minimum requirements are:
  - Ubuntu 24.04 or 22.04 or 20.04 LTS operating system (server or desktop)
  - 4 vCPUs @ 100% execution capacity
  - 8 GB of RAM (recommended 10-12 GB if used for development)
  - 60 GB of storage disk (recommended 80-100 GB if used for development)

For the sake of simplicity, we provide a pre-installed Ubuntu 22.04 VM with MicroK8s v1.29 installed.

- To perform your own installation, follow the steps described in the Wiki pages:
  - https://tfs.etsi.org/documentation/develop/deployment_guide/#112-oracle-virtual-box
  - https://tfs.etsi.org/documentation/develop/deployment_guide/#12-install-microk8s

Deploy TeraFlowSDN controller

Before continuing, check the status of your MicroK8s environment as described in

https://tfs.etsi.org/documentation/develop/deployment_guide/#12-install-microk8s

- Check status of MicroK8s:

```
microk8s.status --wait-ready
```

- If the command reports “microk8s is not running”, start MicroK8s:

```
microk8s.start
```

Deploy TeraFlowSDN controller

Before continuing, check the status of your MicroK8s environment as described in

https://tfs.etsi.org/documentation/develop/deployment_guide/#12-install-microk8s

- Report status of MicroK8s every second. Wait till addons “dns, helm3, hostpath-storage, ingress, registry” are enabled, then terminate command with Ctrl+C.

```
watch -n 1 microk8s.status --wait-ready
```

- Report status of TFS components every second. Wait till all pods are Running and Available, then terminate command with Ctrl+C.

```
watch -n 1 kubectl get all --all-namespaces
```

Deploy TeraFlowSDN controller

Specifications to deploy TeraFlowSDN are defined in a bash script as a set of environment variables. Complete details available in:

https://tfs.etsi.org/documentation/develop/deployment_guide/#13-deploy-teraflowsdn

- Organized in 4 sections:
  - TeraFlowSDN: variables related to the deployment of TeraFlowSDN controller
  - CockroachDB: variables related to the deployment of CockroachDB distributed database (used by Context)
  - NATS: variables related to the deployment of NATS message broker (used by Context)
  - QuestDB: variables related to the deployment of QuestDB time-series database (used by Monitoring)

- Changing the default values for CockroachDB, NATS, and QuestDB is for advanced setups.
  - Not covered in this session.

Deploy TeraFlowSDN controller

Specifications to deploy TeraFlowSDN are defined in a bash script as a set of environment variables.

- Example TeraFlowSDN section (check “my_deploy.sh” for the complete list of settings)

```
# Set the URL of the internal MicroK8s Docker registry where the images will be uploaded to.
export TFS_REGISTRY_IMAGES="http://localhost:32000/tfs/"

# Set the list of components, separated by spaces, you want to build images for, and deploy.
export TFS_COMPONENTS="context device pathcomp service nbi webui"

# Set the tag you want to use for your images.
export TFS_IMAGE_TAG="dev"

# Set the name of the Kubernetes namespace to deploy TFS to.
export TFS_K8S_NAMESPACE="tfs"

# Set additional manifest files to be applied after the deployment (example, NGINX ingress controller)
export TFS_EXTRA_MANIFESTS="manifests/nginx_ingress_http.yaml"

# Set the new Grafana admin password
export TFS_GRAFANA_PASSWORD="admin123+"

# Enable skip-build flag to prevent rebuilding the Docker images (images are pre-built, only for demo purposes).
export TFS_SKIP_BUILD="YES"
```

Deploy TeraFlowSDN controller

If you want to tweak the deployment specifications, create a copy of “my_deploy.sh” script and adjust parameters at will.

When you are fine with your specifications, launch the deployment as follows:

```
$ cd ~/tfs-ctrl
$ source my_deploy.sh
$ ./deploy/all.sh
```

We prepared a specs file and script to automate the hackfest redeploy:
`$ ~/tfs-ctrl/hackfest5/redeploy-tfs.sh`

The script deploys CockroachDB, NATS and QuestDB, and then procedes with TFS deployment.

- The deployment might take few minutes the first time…
  - VM is configured with pre-built components to speed-up the deployment.
- You should see the progress of the deployment.

Deploy TeraFlowSDN controller

```
<Deploy CockroachDB>
<Deploy NATS>
<Deploy QuestDB>

Deleting and Creating a new namespace...

Create Secret with CockroachDB data
Create secret with NATS data
Create secret with QuestDB data

…

Deploying components and collecting environment variables...
Processing 'context' component...
  Building Docker image...
  Pushing Docker image to 'http://localhost:32000/tfs/'...
  Adapting 'context' manifest file...
  Deploying 'context' component to Kubernetes...
  Collecting env-vars for 'context' component...
…

Deploying extra manifests...
Processing manifest 'manifests/nginx_ingress_http.yaml'...
ingress.networking.k8s.io/tfs-ingress created
```

```
Waiting for 'context' component...
deployment.apps/contextservice condition met
…

Configuring WebUI DataStores and Dashboards...
…

Pruning Docker Buildx Cache...
…

Deployment Resources:
           <see next slide>
```

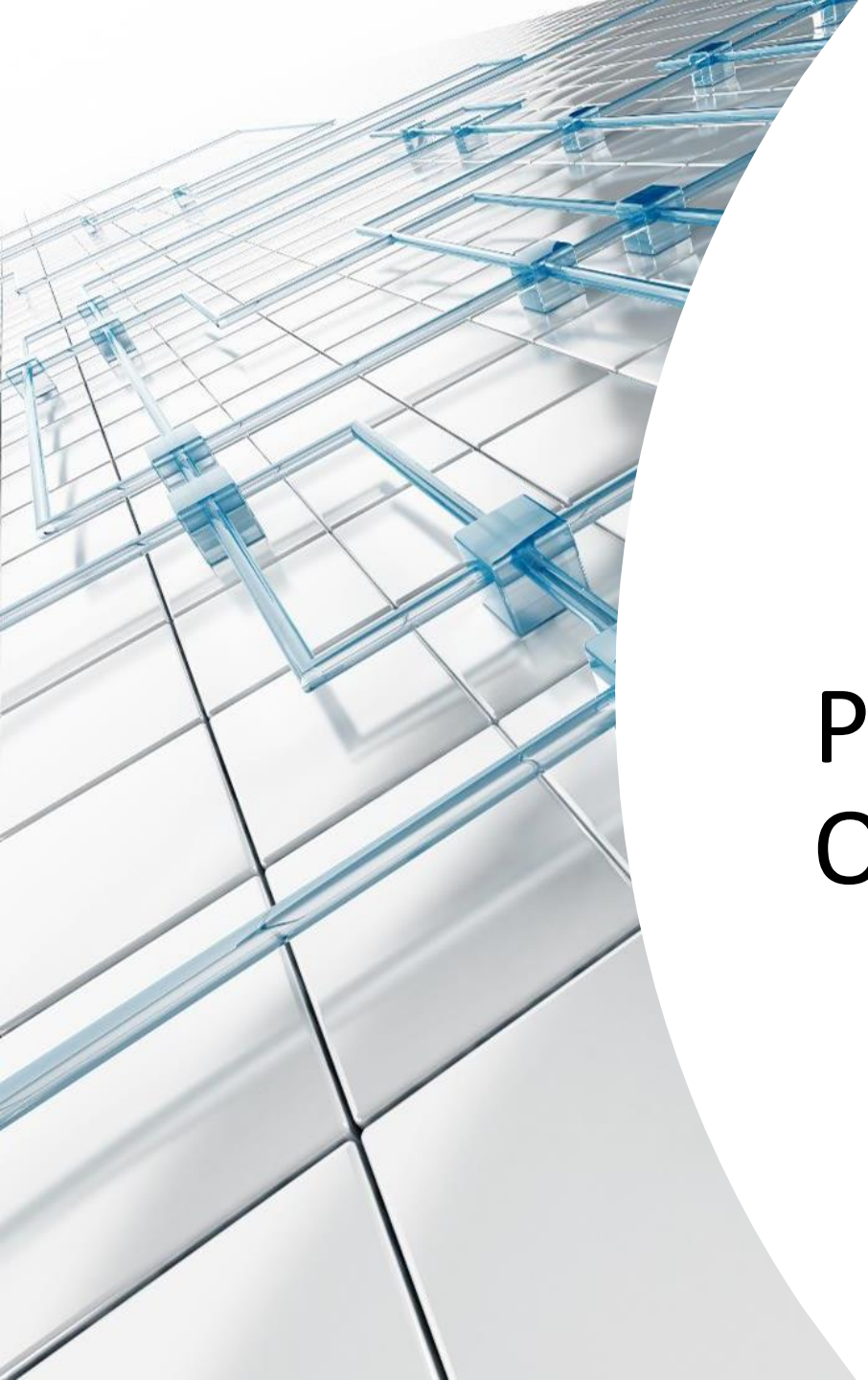
Deploy TeraFlowSDN controller

The process concludes reporting the status of the microservices.

```
Deployment Resources:
NAME                                     READY   STATUS    RESTARTS        AGE
pod/contextservice-55f7f77f-dqfsc        1/1     Running   0               5m12s
pod/deviceservice-67fb99b9dd-cjcsk       1/1     Running   0               5m1s
…

NAME                        TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)
service/contextservice      ClusterIP   10.152.183.225   <none>        1010/TCP,8080/TCP
service/deviceservice       ClusterIP   10.152.183.194   <none>        2020/TCP
…

NAME                                  READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/contextservice        1/1     1            1           5m12s
deployment.apps/deviceservice         1/1     1            1           5m1s
…

NAME                                           DESIRED   CURRENT   READY   AGE
replicaset.apps/contextservice-55f7f77f        1         1         1       5m12s
replicaset.apps/deviceservice-67fb99b9dd       1         1         1       5m1s
…

Deployment Ingress:
NAME          CLASS    HOSTS   ADDRESS     PORTS   AGE
tfs-ingress   public   *       127.0.0.1   80      3m15s
```

You can always retrieve this status as follows:

```
$ cd ~/tfs-ctrl
$ source hackfest5/deploy_specs.sh
$ ./deploy/show.sh
```



TeraFlow SDN

PART I
Onboarding devices

Onboard a Scenario

TeraFlowSDN enables to create entities through JSON-based descriptor files.

- Topology descriptor for this hackfest:
  - ~/tfs-ctrl/hackfest5/data/tfs-topology.json
- Check it beautified in the browser:
  - https://labs.etsi.org/rep/tfs/controller/-/blob/feat/hackfest5/hackfest5/data/tfs-topology.json
- Might contain multiple optional sections:
  - Context/Topology => defines the working topology name. You might have multiple in complex scenarios.
  - Devices => List of devices TFS should onboard and manage
  - Links => List of links connecting endpoints (ports) of devices
  - Services => List of services to create/update (using internal TFS nomenclature)
  - Slices => List of transport network slices to create/update (using internal TFS nomenclature)

Onboard a Scenario > Topology

TeraFlow SDN

We create context “admin” and topology “admin” inside.

- Context(admin)/Topology(admin) => Special in TFS; it views everything in TFS.

```
{
  "contexts": [
    {"context_id": {"context_uuid": {"uuid": "admin"}}}
  ],
  "topologies": [
    {
      "topology_id": {
        "context_id": {"context_uuid": {"uuid": "admin"},
        "topology_uuid": {"uuid": "admin"}
      }}
    }
  ]
}
```

Onboard a Scenario > Devices > Emulated

We also create emulated devices for the DCs.

- We do not manage them in this scenario, but we need them as placeholders.

Drivers usable for this device. Undefined means Emulated.
Check “proto/context.proto” “DeviceDriverEnum”

Type of device. Check
“src/common/DeviceTypes.py”

Mgmt IP address/port of the target device.
(ignored by Emulated driver)

```
{
  "devices": [
    {
      "device_id": {"device_uuid": {"uuid": "dc1"}}, "device_type": "emu-datacenter",
      "device_drivers": ["DEVICEDRIVER_UNDEFINED"],
      "device_config": {"config_rules": [
        {"action": "CONFIGACTION_SET", "custom": {"resource_key": "_connect/address",  "resource_value": "127.0.0.1"}},
        {"action": "CONFIGACTION_SET", "custom": {"resource_key": "_connect/port",     "resource_value": "0"}},
        {"action": "CONFIGACTION_SET", "custom": {"resource_key": "_connect/settings", "resource_value": {
          "endpoints": [
            {"uuid": "eth1", "type": "copper"},
            {"uuid": "int", "type": "copper"}
          ]
        }}}
      ]}
    },
    ...
  ]
}
```

Usually, endpoints are automatically discovered from the device.
For emulated, we need to provide them in the driver settings.

Onboard a Scenario > Devices > Routers

TeraFlow SDN

For the router we use gNMI/OpenConfig driver

The device is a router.

Mgmt IP address/port of the target device.

```
{
  "devices": [
    {
      "device_id": {"device_uuid": {"uuid": "r1"}}, "device_type": "packet-router",
      "device_drivers": ["DEVICEDRIVER_GNMI_OPENCONFIG"],
      "device_config": {"config_rules": [
        {"action": "CONFIGACTION_SET", "custom": {"resource_key": "_connect/address",  "resource_value": "172.20.20.101"}},
        {"action": "CONFIGACTION_SET", "custom": {"resource_key": "_connect/port",     "resource_value": "6030"}},
        {"action": "CONFIGACTION_SET", "custom": {"resource_key": "_connect/settings", "resource_value": {
          "username": "admin", "password": "admin", "use_tls": false
        }}}
      ]}
    },
    ...
  ]
}
```

Some drivers might require additional parameters, such as credentials or security settings.

Onboard a Scenario > Links

Similarly, Links can be uploaded using JSON-based descriptors.

- NOTE: links are considered unidirectional. For bidirectional, create link A->B and B->A.

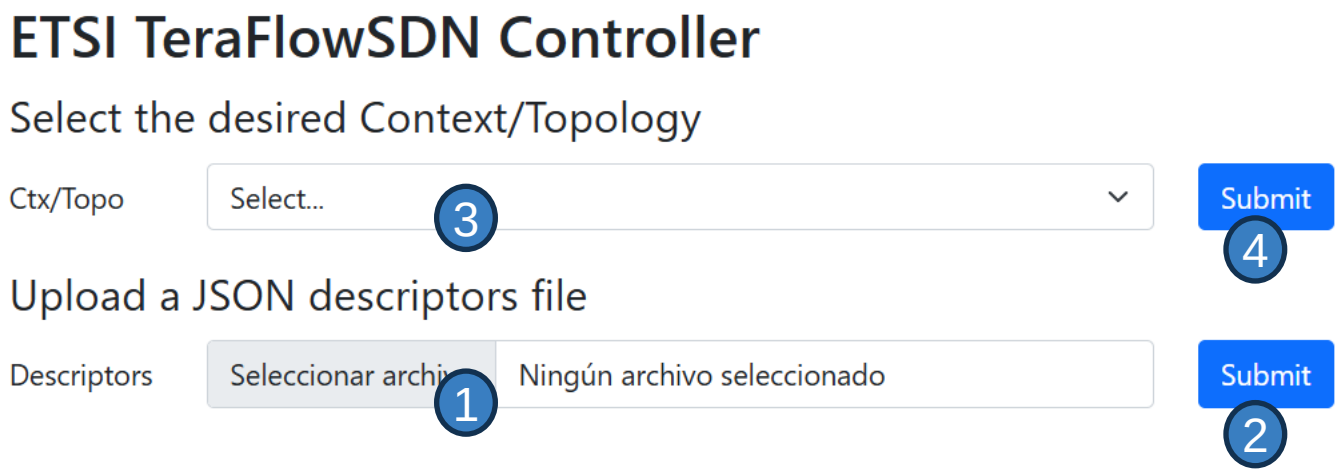
Link UUID (if you provide a plain string, a UUID will be generated automatically)

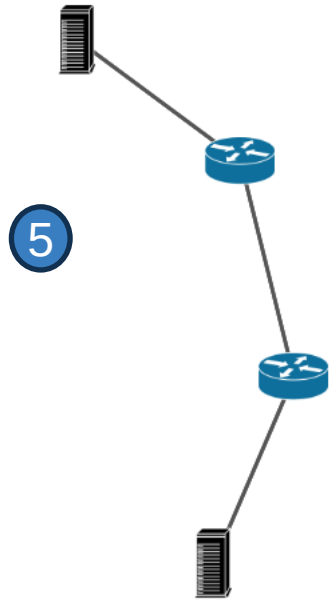
```
{
  "links": [
    {
      "link_id": {"link_uuid": {"uuid": "r1/Ethernet2==r2/Ethernet1"}},
      "link_endpoint_ids": [
        {"device_id": {"device_uuid": {"uuid": "r1"}}, "endpoint_uuid": {"uuid": "Ethernet2"}},
        {"device_id": {"device_uuid": {"uuid": "r2"}}, "endpoint_uuid": {"uuid": "Ethernet1"}}
      ]
    },
    …
  ]
}
```

Specify Device and Endpoint UUIDs (you can provide them as names, the UUID will be located automatically)

Onboard a Scenario through WebUI

Navigate to: http://127.0.0.1/webui

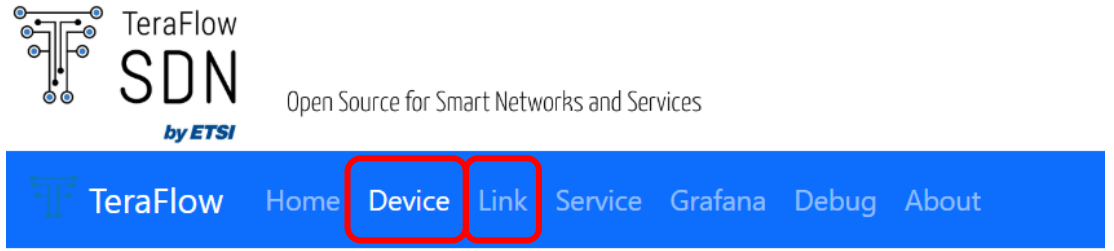




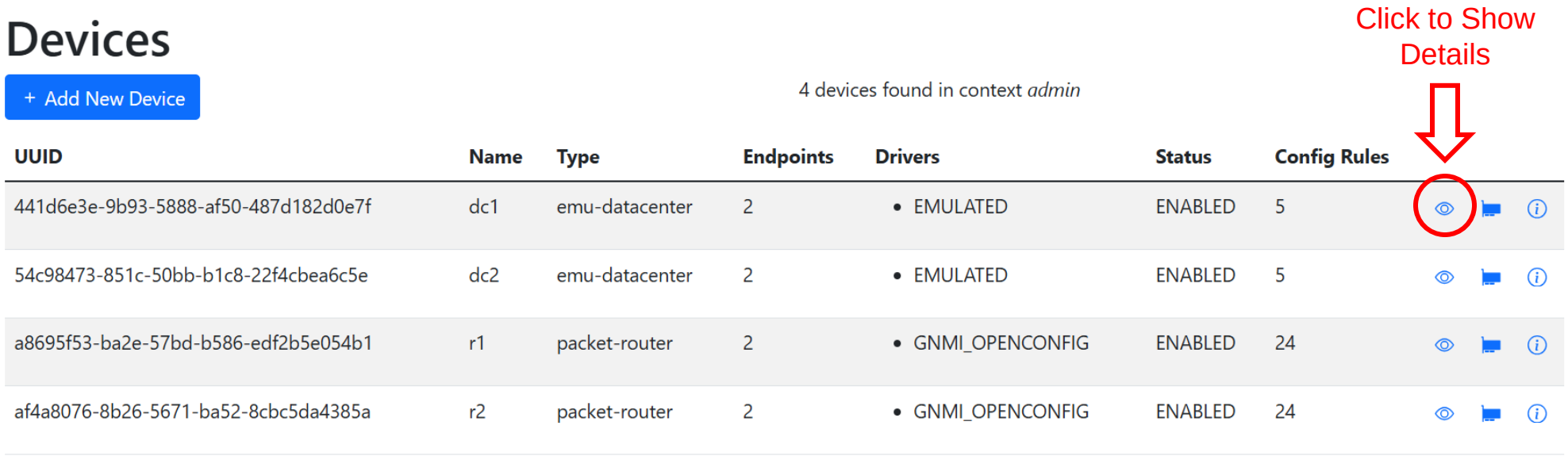
1. Select file tfs-topology.
   Download from: https://labs.etsi.org/rep/tfs/controller/-/raw/feat/hackfest5/hackfest5/data/tfs-topology.json?inline=false
2. Click “Submit”
3. Select “Context(admin)/Topology(admin)”
4. Click “Submit”
5. You should see the topology below

Inspect elements created

Navigate to the different entities.



The details of the managed entities can be shown using the “eye” button.



Error checking, if something went wrong…

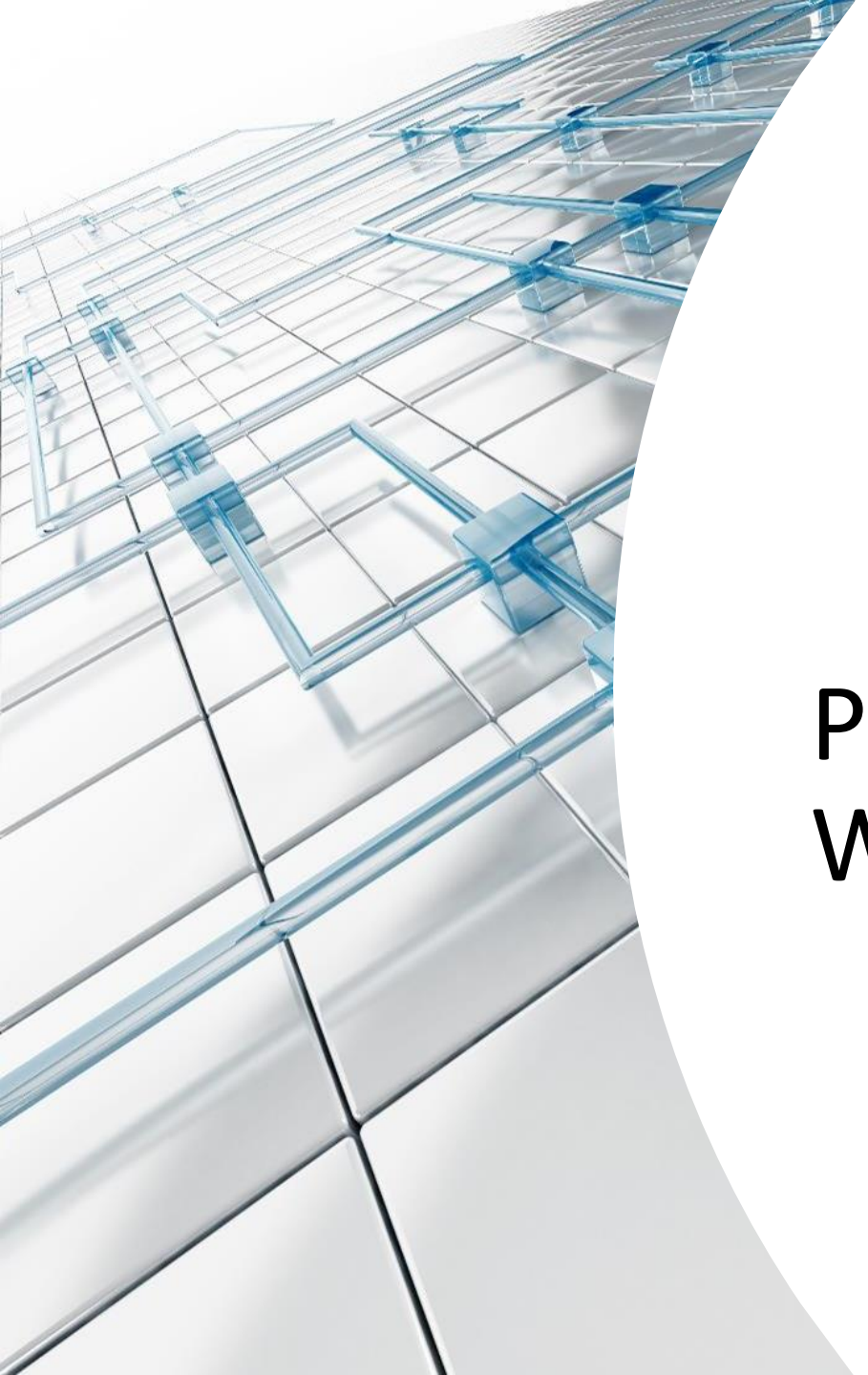
Check the logs of the TeraFlowSDN components:

- Example: Device component

```
$ source ~/tfs-ctrl/hackfest5/deploy_specs.sh
$ ~/tfs-ctrl/scripts/show_logs_device.sh
```

TeraFlow SDN by ETSI

Coffee Break!



TeraFlow SDN

PART II
What are L3VPNs?

L3 VPNs (Layer 3 Virtual Private Networks)

A Layer 3 VPN allows **multiple customers** to securely **connect** their **remote networks** over a shared service provider's IP backbone using **Layer 3 (IP) routing**.

Key features:

- Uses MPLS (Multiprotocol Label Switching), Virtual Local Area Networ (VLAN) tags, or similar technologies for routing and traffic segregation.
- Employs VRFs (Virtual Routing and Forwarding instances) to maintain separate routing tables for each customer.
  - Each VRF behaves as a virtual router
- Provides privacy and security without the need for encryption, leveraging label-based isolation.

L3 VPNs (Layer 3 Virtual Private Networks)

A Layer 3 VPN allows **multiple customers** to securely **connect** their **remote networks** over a shared service provider's IP backbone using **Layer 3 (IP) routing**.

How it Works:

- The service provider’s PE (Provider Edge) routers are connected to the CE (Customer Equipment) routers, e.g., a DC gateway.
- CE routes are advertised via dynamic routing protocols such as MultiProtocol Border Gateway Protocol (MP-BGP) PE routers.
- Data flows between customer sites are encapsulated using MPLS labels or VLAN tags for delivery.

L3 VPNs (Layer 3 Virtual Private Networks)

TeraFlow SDN

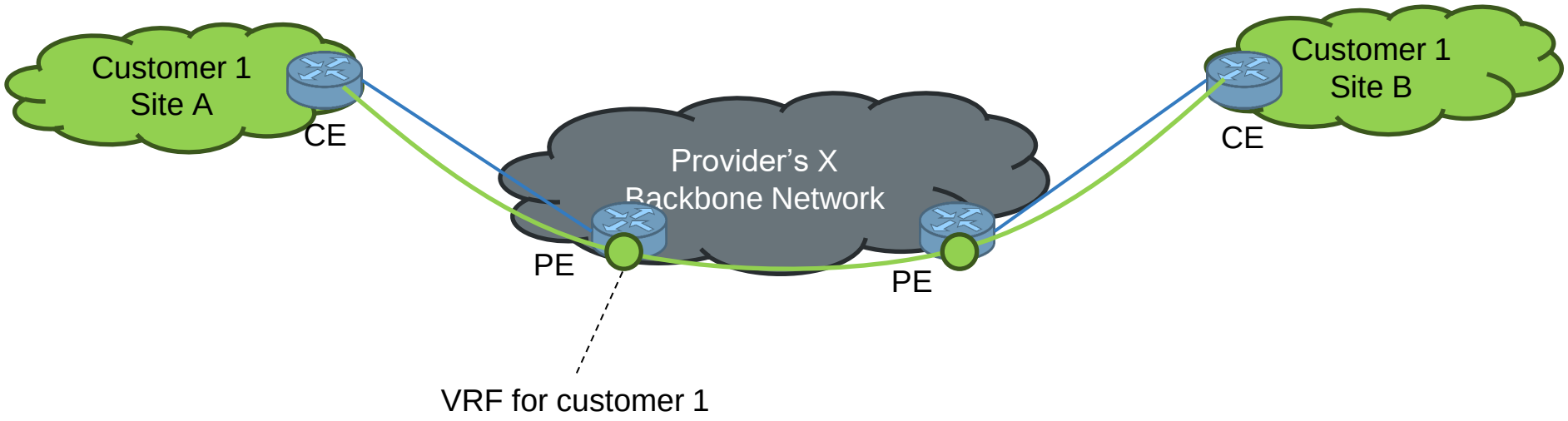
A Layer 3 VPN allows **multiple customers** to securely **connect** their **remote networks** over a shared service provider's IP backbone using **Layer 3 (IP) routing**.

Use Cases:

- Enterprise Wide Area Network (WAN) interconnects across geographies.
- Secure, scalable connectivity for multiple branch offices.
- Integration of hybrid or multi-cloud environments.

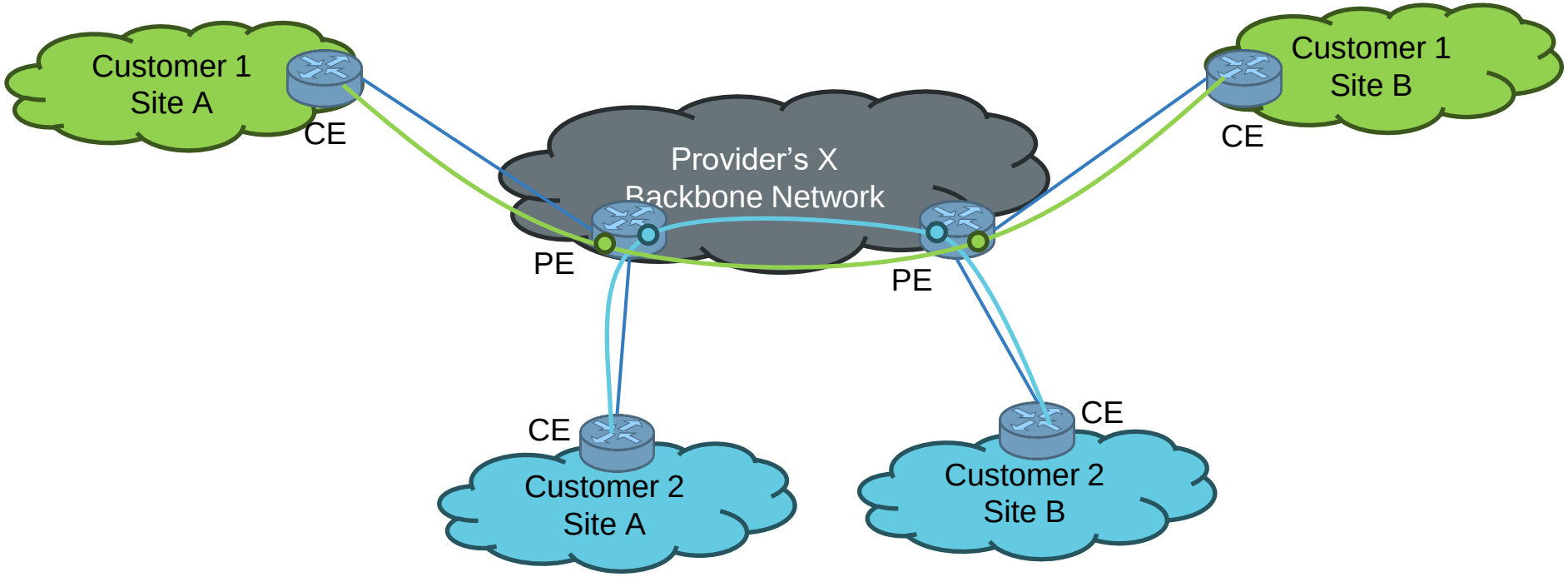
L3 VPNs (Layer 3 Virtual Private Networks)

Basic example



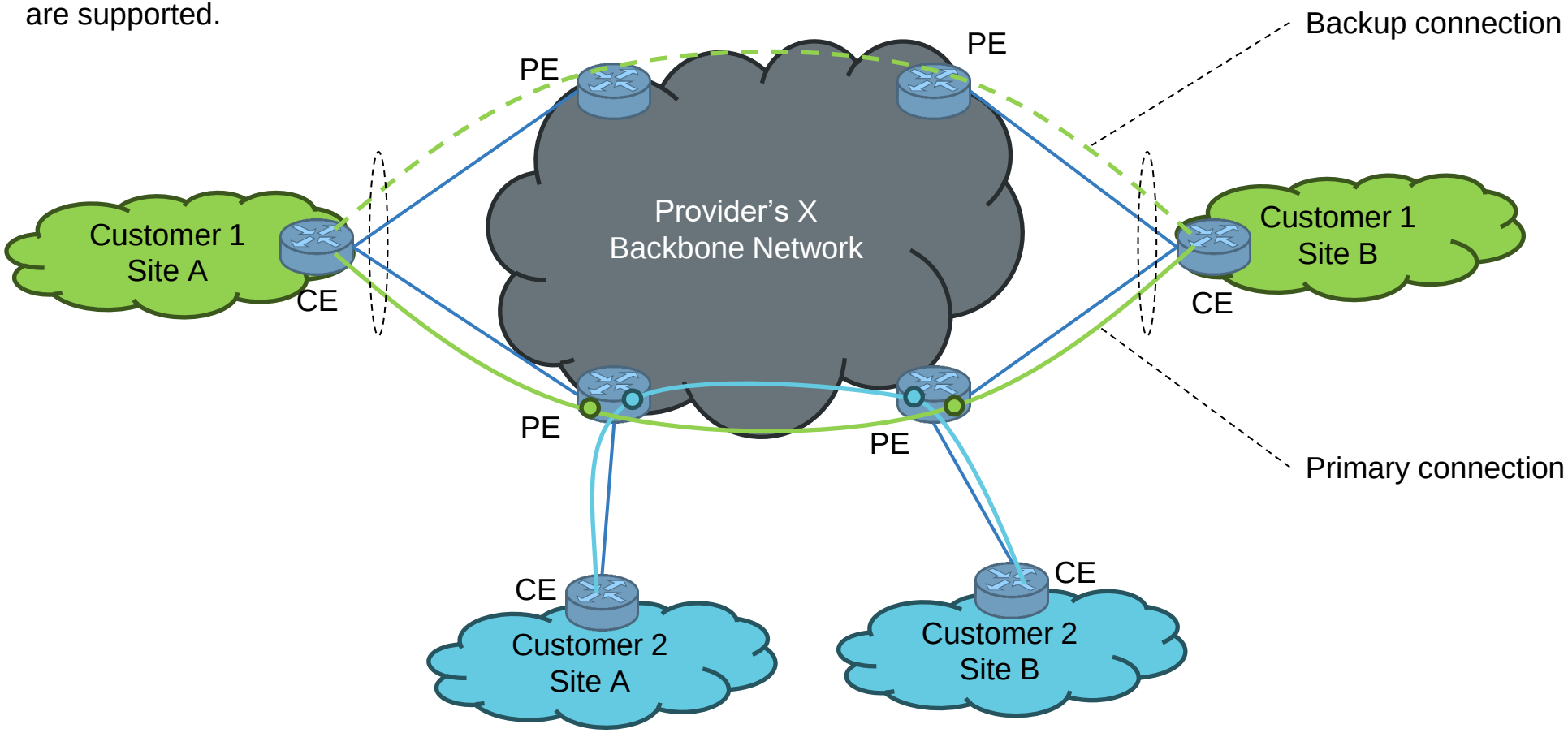
L3 VPNs (Layer 3 Virtual Private Networks)

Multiple customers convey traffic isolately thanks to independent VRFs.



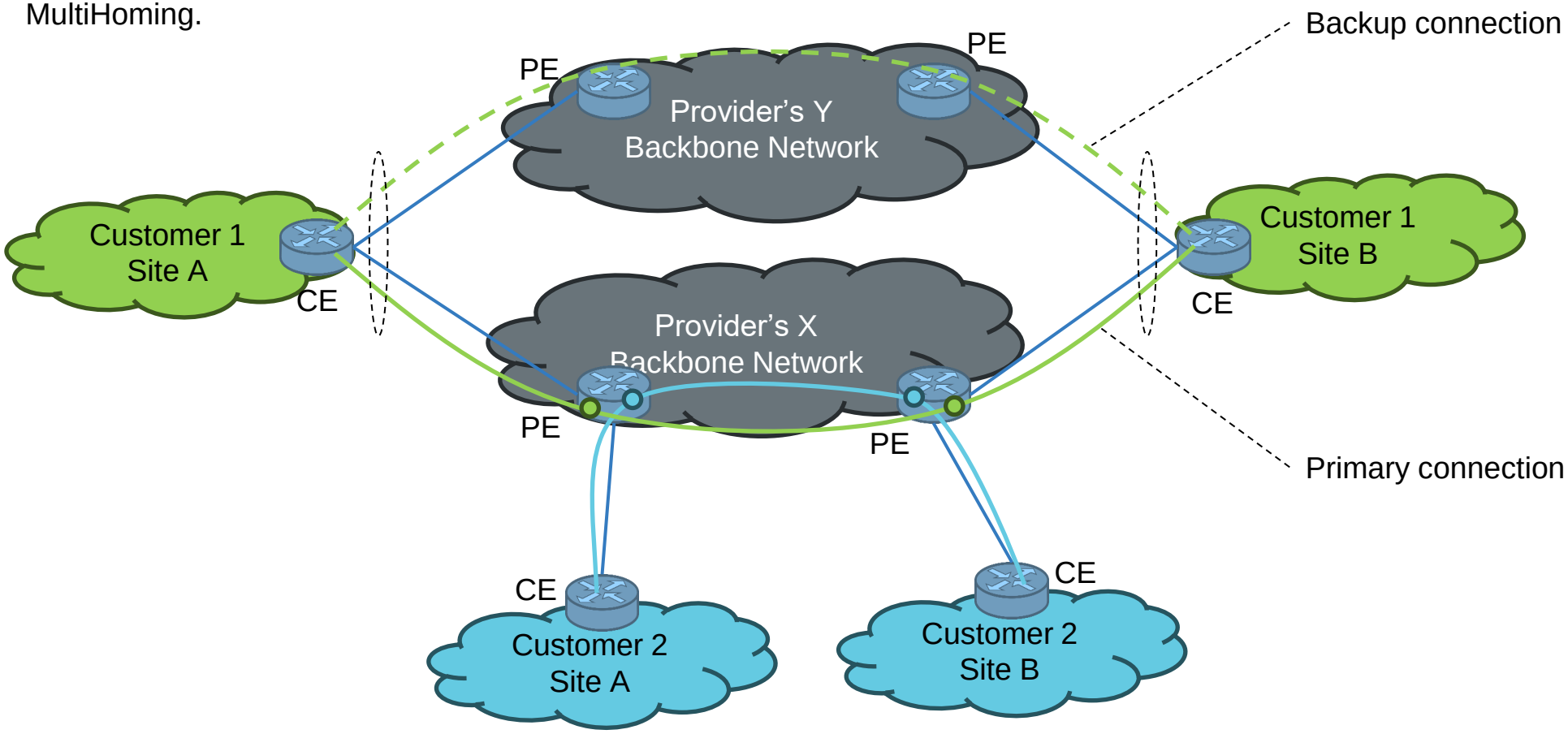
L3 VPNs (Layer 3 Virtual Private Networks)

High-Availability scenarios are supported.



L3 VPNs (Layer 3 Virtual Private Networks)

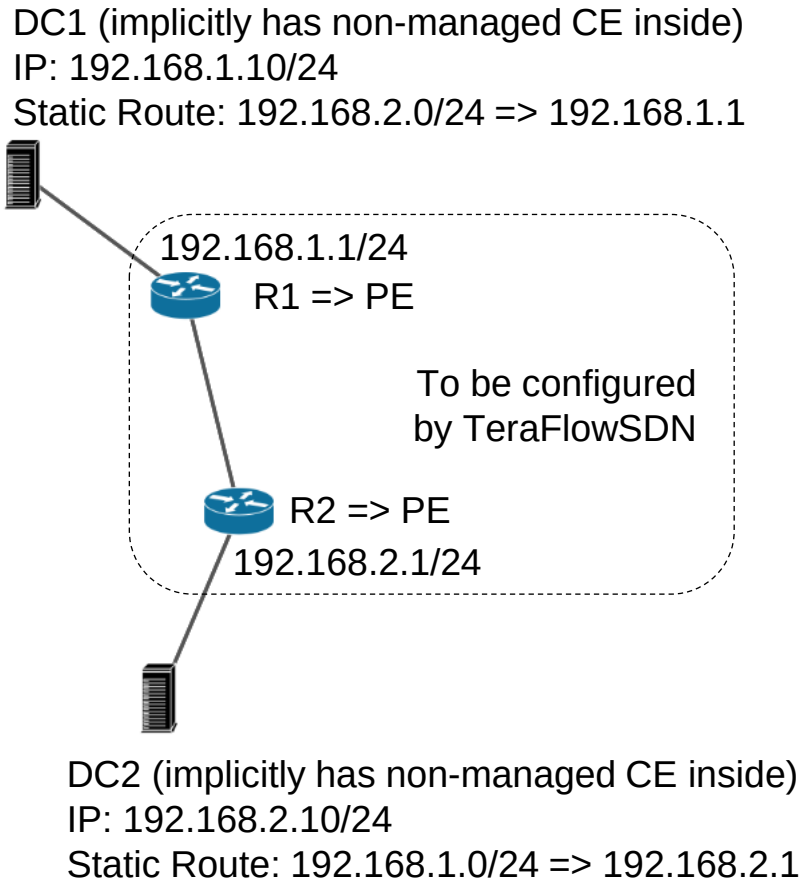
Even, high-availability through MultiHoming.

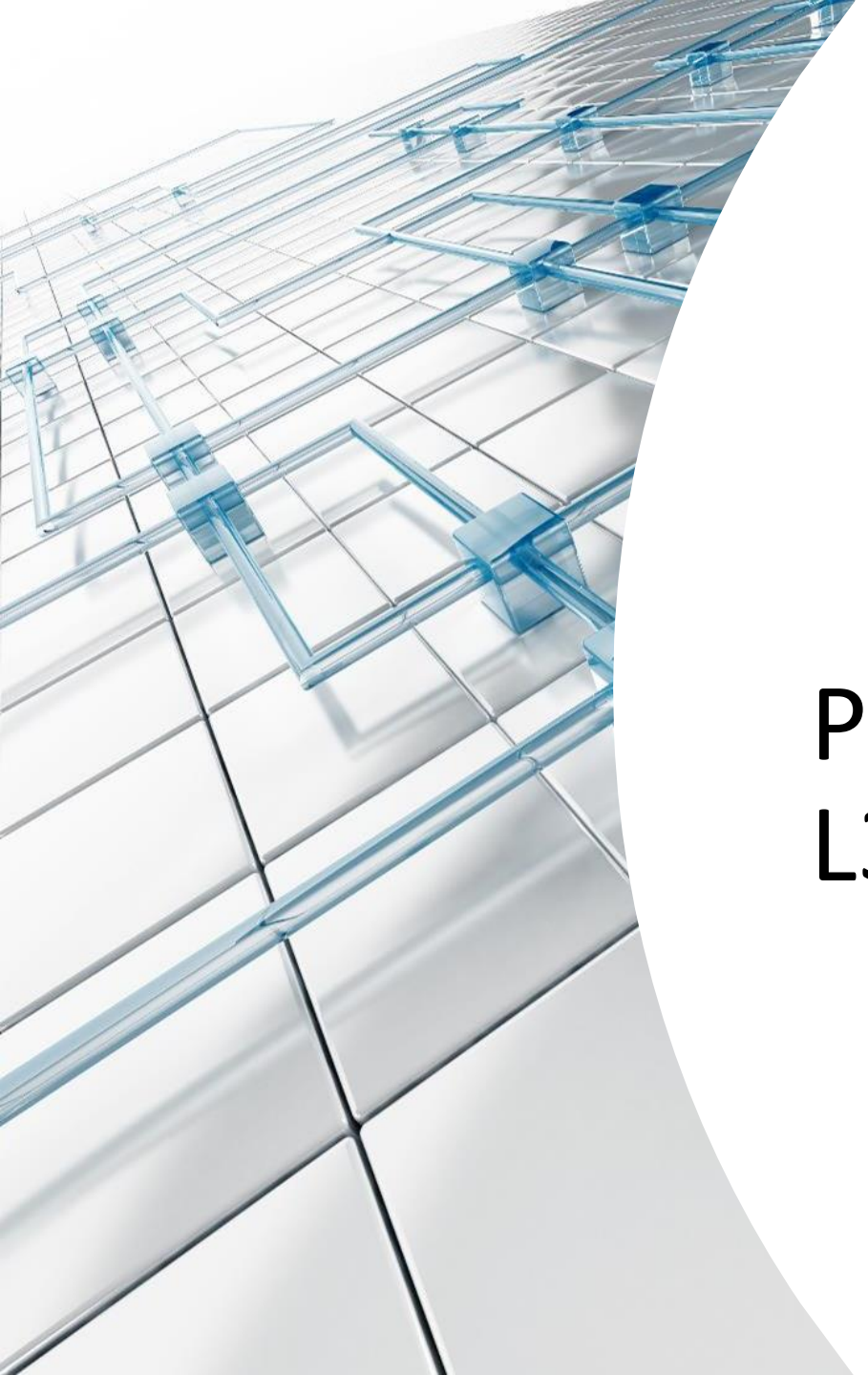


But let's keep things simple ☺

Our simplification for today:

- Scenario: 2 CEs (we assume preconfigured) + 2 PEs (to be configured by TeraFlowSDN)
- We will skip VRFs and use the default one.
- IP Addresses: TeraFlowSDN will identify missing IP addresses, ranges, and gateways, and configure them on the routers.
- Routing protocol: "static routing", TeraFlowSDN will infer appropriate static routes and install them on the routers.
- Target: we should be able to ping & SSH from DC1 to DC2.





TeraFlow SDN

PART II
L3VPN Service Establishment

Descriptor IETF L3VPN Service Delivery [RFC 8299]

https://datatracker.ietf.org/doc/rfc8299/

A IETF L3VPN request has been prepared:

- ~/tfs-ctrl/hackfest5/data/ietf-l3vpn-service.json

See it in a beautified way:

- https://labs.etsi.org/rep/tfs/controller/-/blob/feat/hackfest5/hackfest5/data/ietf-l3vpn-service.json

Descriptor IETF L3VPN Service Delivery [RFC 8299]

TeraFlow SDN

https://datatracker.ietf.org/doc/rfc8299/

```
{"ietf-l3vpn-svc:l3vpn-svc": {
    "vpn-services": {"vpn-service": [{"vpn-id": "ietf-l3vpn-svc"}]},
    "sites": {"site": [
        {
            "site-id": "site_DC1",
            "management": {"type": "ietf-l3vpn-svc:provider-managed"},
            "locations": {"location": [{"location-id": "DC1"}]},
            "devices": {"device": [{"device-id": "dc1", "location": "DC1"}]},
            "site-network-accesses": {
                "site-network-access": […]
            }
        },
        {
            "site-id": "site_DC2",
            "management": {"type": "ietf-l3vpn-svc:provider-managed"},
            "locations": {"location": [{"location-id": "DC2"}]},
            "devices": {"device": [{"device-id": "dc2", "location": "DC2"}]},
            "site-network-accesses": {
                "site-network-access": […]
            }
        }
    ]}
}}
```

Descriptor IETF L3VPN Service Delivery [RFC 8299]

TeraFlow SDN

https://datatracker.ietf.org/doc/rfc8299/

```
{"ietf-l3vpn-svc:l3vpn-svc": {
    "vpn-services": {"vpn-service": [{"vpn-id": "ietf-l3vpn-svc"}]},
    "sites": {"site": [
        {
            "site-id": "site_DC1",
            "management": {"type": "ietf-l3vpn-svc:provider-managed"},
            "locations": {"location": [{"location-id": "DC1"}]},
            "devices": {"device": [{"device-id": "dc1", "location": "DC1"}]},
            "site-network-accesses": {"site-network-access": [
                {
                    "site-network-access-id": "eth1",
                    "site-network-access-type": "ietf-l3vpn-svc:multipoint",
                    "device-reference": "dc1",
                    "vpn-attachment": {"vpn-id": "ietf-l3vpn-svc", "site-role": "ietf-l3vpn-svc:spoke-role"},
                    "ip-connection": {"ipv4": {
                        "address-allocation-type": "ietf-l3vpn-svc:static-address",
                        "addresses": {
                            "provider-address": "192.168.1.1",
                            "customer-address": "192.168.1.10",
                            "prefix-length": 24
                        }
                    }},
                    "service": {…}
                }
            ]}
        },
        {"site-id": "site_DC2", …}
    ]}
}}
```

Descriptor IETF L3VPN Service Delivery [RFC 8299]

TeraFlow SDN

https://datatracker.ietf.org/doc/rfc8299/

```
{"ietf-l3vpn-svc:l3vpn-svc": {
    "vpn-services": {"vpn-service": [{"vpn-id": "ietf-l3vpn-svc"}]},
    "sites": {"site": [
        {
            "site-id": "site_DC1",
            "management": {"type": "ietf-l3vpn-svc:provider-managed"},
            "locations": {"location": [{"location-id": "DC1"}]},
            "devices": {"device": [{"device-id": "dc1", "location": "DC1"}]},
            "site-network-accesses": {"site-network-access": [
                {
                    "site-network-access-id": "eth1",
                    …
                    "service": {
                        "svc-mtu": 1500,
                        "svc-input-bandwidth": 1000000000,                  → 1Gbps
                        "svc-output-bandwidth": 1000000000,                 → 1Gbps
                        "qos": {"qos-profile": {"classes": {"class": [{
                            "class-id": "qos-realtime",
                            "direction": "ietf-l3vpn-svc:both",
                            "latency": {"latency-boundary": 10},            → 10 ms
                            "bandwidth": {"guaranteed-bw-percent": 100}     → 100% availability
                        }]}}}
                    }
                }
            ]}
        },
        {"site-id": "site_DC2", …}
    ]}
}}
```

Descriptor IETF L3VPN Service Delivery [RFC 8299]

TeraFlow SDN

https://datatracker.ietf.org/doc/rfc8299/

```
{"ietf-l3vpn-svc:l3vpn-svc": {
    "vpn-services": {"vpn-service": [{"vpn-id": "ietf-l3vpn-svc"}]},
    "sites": {"site": [
        {"site-id": "site_DC1", …},
        {
            "site-id": "site_DC2",
            "management": {"type": "ietf-l3vpn-svc:provider-managed"},
            "locations": {"location": [{"location-id": "DC2"}]},
            "devices": {"device": [{"device-id": "dc2", "location": "DC2"}]},
            "site-network-accesses": {"site-network-access": [
                {
                    "site-network-access-id": "eth1",
                    "site-network-access-type": "ietf-l3vpn-svc:multipoint",
                    "device-reference": "dc2",
                    "vpn-attachment": {"vpn-id": "ietf-l3vpn-svc", "site-role": "ietf-l3vpn-svc:spoke-role"},
                    "ip-connection": {"ipv4": {
                        "address-allocation-type": "ietf-l3vpn-svc:static-address",
                        "addresses": {
                            "provider-address": "192.168.2.1",
                            "customer-address": "192.168.2.10",
                            "prefix-length": 24
                        }
                    }},
                    "service": {…}
                }
            ]}
        }
    ]}
}}
```

Requesting the IETF L3VPN to TeraFlowSDN

The NBI component of TeraFlowSDN exposes a IETF L3VPN connector.

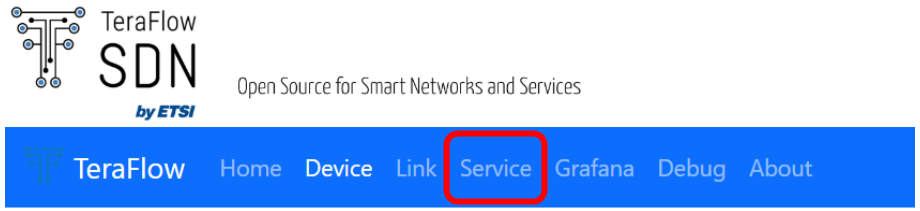
```
$ cd ~/tfs-ctrl/hackfest5/data
$ curl -X POST \
       --header "Content-Type: application/json" \
       --data @ietf-l3vpn-service.json \
       --user "admin:admin" \
       http://127.0.0.1/restconf/data/ietf-l3vpn-svc:l3vpn-svc/vpn-services
```

Will take few seconds to process it…

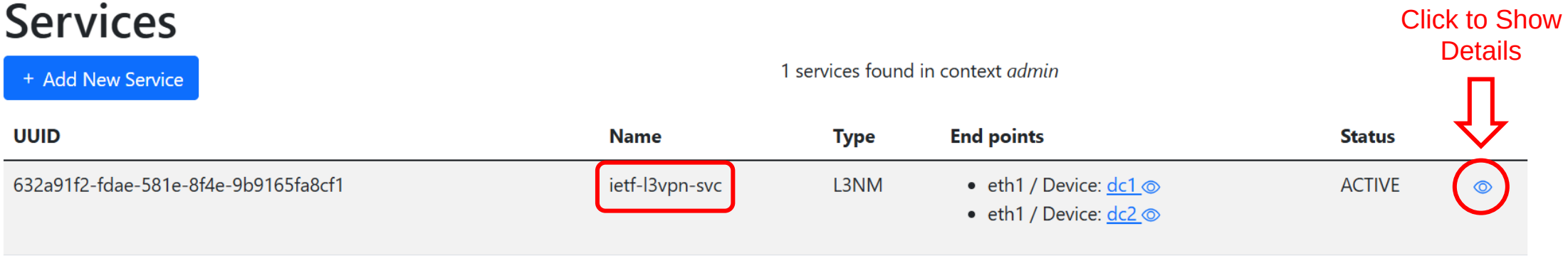
If reply is empty, everything went fine ☺, if there is some error, it will be reported.

Checking the IETF L3VPN

When completed successfully, check the Services in the WebUI:



Should see something like:



Checking the IETF L3VPN

TeraFlow SDN

Service ietf-l3vpn-svc (632a91f2-fdae-581e-8f4e-9b9165fa8cf1)

Back to service list Delete service

Context: 43813baf-195e-5da6-af20-b3d0922e71a7
UUID: 632a91f2-fdae-581e-8f4e-9b9165fa8cf1
Name: ietf-l3vpn-svc
Type: L3NM
Status: ACTIVE

Endpoint UUID	Name	Device	Endpoint Type
33c89568-b10f-52a3-8a47-d759b4ad90e2	eth1	dc1	copper
9ed30be9-eaf9-5167-a127-05bd744fa556	eth1	dc2	copper

Constraints:

Kind	Key/Type	Value
Endpoint Location	dc1 / eth1	Region: site_DC1
SLA Capacity	-	1.0 Gbps
SLA E2E Latency	-	10.0 ms
SLA Availability	-	100.0 %; 1 disjoint paths; all-active
Endpoint Location	dc2 / eth1	Region: site_DC2

Checking the IETF L3VPN

TeraFlow SDN

Configurations:

Key	Value
/settings	• **mtu:** 1500
/static_routing	
/device[dc1]/endpoint[eth1]/settings	• **ip_address:** 192.168.1.10 • **neighbor_address:** 192.168.1.1 • **prefix_length:** 24
/device[dc2]/endpoint[eth1]/settings	• **ip_address:** 192.168.2.10 • **neighbor_address:** 192.168.2.1 • **prefix_length:** 24
/device[r1]/endpoint[Ethernet10]/settings	• **ip_address:** 192.168.1.1 • **prefix_length:** 24
/device[r2]/endpoint[Ethernet10]/settings	• **ip_address:** 192.168.2.1 • **prefix_length:** 24

Specified in the request

Assigned by TeraFlowSDN to the router interfaces.

Computed path:

Connection Id	Sub-Service	Path
ea79e88f-9380-4f2d-be6c-a930ec649663		dc1 / eth1 — r1 / Ethernet10 — r1 / Ethernet2 — r2 / Ethernet1 — r2 / Ethernet10 — dc2 / eth1

Check the IETF L3VPN through the NBI:

TeraFlow SDN

- Retrieve UUID of the IETF L3VPN:

```
$ curl --user "admin:admin" \
    http://127.0.0.1/restconf/data/ietf-l3vpn-svc:l3vpn-svc/vpn-services/vpn-service=ietf-l3vpn-svc/
```

- Delete the IETF L3VPN **[don’t run this now ☺]**:

```
$ curl -X DELETE --user "admin:admin" \
    http://127.0.0.1/restconf/data/ietf-l3vpn-svc:l3vpn-svc/vpn-services/vpn-service=ietf-l3vpn-svc/
```

Check configurations in the routers through CLI:

TeraFlow SDN

- Get configuration of router R1:

```
$ ~/tfs-ctrl/hackfest5/clab-cli-r1.sh

   r1>enable

   r1#show running-config

   …

   r1#exit
```

```
! Command: show running-config
! device: r1 (cEOSLab, EOS-4.31.5M-38783521.4315M (engineering build))
!
…
management api gnmi
   transport grpc default
!
management api netconf
   transport ssh default
!
interface Ethernet2
   no switchport
   ip address 10.254.172.69/30
!
interface Ethernet10
   no switchport
   ip address 192.168.1.1/24
!
interface Management0
   ip address 172.20.20.101/24
!
no ip routing
!
ip route 0.0.0.0/0 172.20.20.1
ip route 192.168.2.0/24 10.254.172.70
!
end
```

Check configurations in the routers through gNMI:

TeraFlow SDN

- Install gNMIc **[pending! run it, please ☺]**:


```
$ sudo bash -c "$(curl -sL https://get-gnmic.kmrd.dev)"
```

- Get gNMI capabilities of router R1:

```
$ gnmic --address clab-hackfest5-r1 --port 6030 --username admin --password admin --insecure capabilities
```

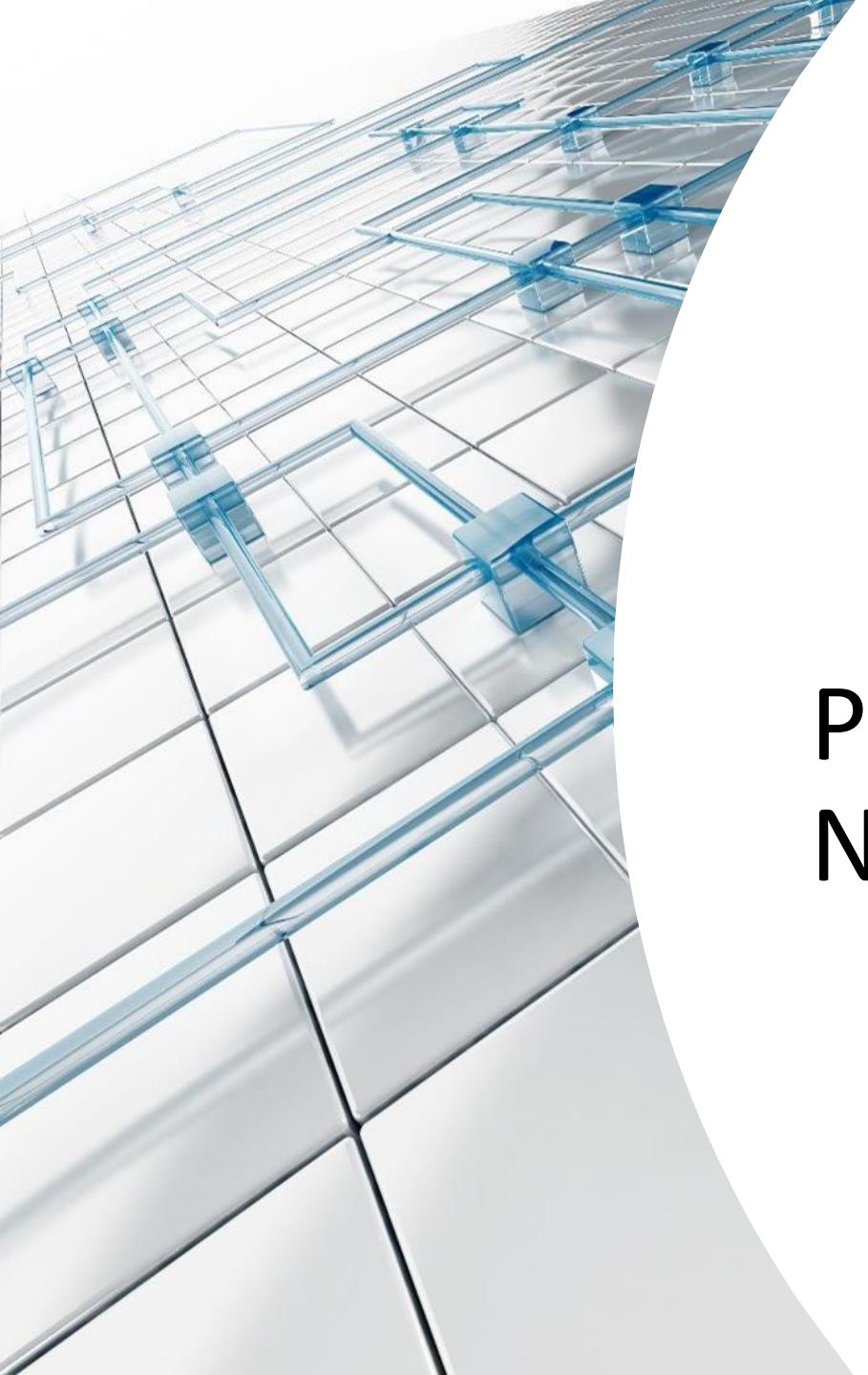
- Retrieve interfaces and network instances configured in router R1:
  - Download the files to your host and open them locally, for instance, using VSCode.

```
$ gnmic --address clab-hackfest5-r1 --port 6030 --username admin --password admin --insecure \
    --encoding json_ietf get --path /interfaces > r1-ifs.json
$ gnmic --address clab-hackfest5-r1 --port 6030 --username admin --password admin --insecure \
    --encoding json_ietf get --path /network-instances > r1-nis.json
```

Test the service:

- Connect to DC1 and start pinging DC2:

```
$ ~/tfs-ctrl/hackfest5/clab-cli-dc1.sh
    ping 192.168.2.10
```



TeraFlow SDN

PART II
Network Monitoring

gNMI Protocol for Telemetry Streaming

- gNMI supports subscription to telemetry streaming: https://github.com/openconfig/gnmi/blob/master/proto/gnmi/gnmi.proto
  - Exploits gRPC-based streaming to periodically report samples according to requested subscription.

```
service gNMI {
  rpc Capabilities(CapabilityRequest      ) returns (CapabilityResponse      );
  rpc Get         (GetRequest             ) returns (GetResponse             );
  rpc Set         (SetRequest             ) returns (SetResponse             );
  rpc Subscribe   (stream SubscribeRequest) returns (stream SubscribeResponse);
}
```

```
message SubscribeResponse {
  oneof response {
    Notification update = 1;
    …
  }
  …
}
```

```
message SubscribeRequest {
  oneof request {
    SubscriptionList subscribe = 1;
    Poll poll = 3;
  }
  …
}
```

```
message SubscriptionList {
  Path prefix = 1;
  repeated Subscription subscription = 2;
  …
  // Mode of the subscription.
  enum Mode {
    STREAM = 0;
    ONCE = 1;
    POLL = 2;
  }
  Mode mode = 5;
  …
}
```

```
enum SubscriptionMode {
  TARGET_DEFINED = 0;
  ON_CHANGE = 1;
  SAMPLE = 2;
}

message Subscription {
  Path path = 1;
  SubscriptionMode mode = 2;
  uint64 sample_interval = 3;
  bool suppress_redundant = 4;
  uint64 heartbeat_interval = 5;
  …
}
```

OpenConfig L3 data models – Interfaces

```
module: openconfig-interfaces
  +--rw interfaces
     +--rw interface* [name]
        +--rw name              -> ../config/name
        +--rw config
        |  +--rw name?            string
        |  +--rw type             identityref
        |  +--rw mtu?             uint16
        |  +--rw loopback-mode?   boolean
        |  +--rw description?     string
        |  +--rw enabled?         boolean
        +--ro state
        |  +--ro name?            string
        |  +--ro type             identityref
        |  +--ro admin-status     enumeration
        |  +--ro oper-status      enumeration
        |  ...
        |  +--ro counters
        |     +--ro in-octets?            oc-yang:counter64
        |     +--ro in-pkts?              oc-yang:counter64
        |     +--ro out-octets?           oc-yang:counter64
        |     +--ro out-pkts?             oc-yang:counter64
        |     ...
        |  ...
        |
        +--rw subinterfaces
           +--rw subinterface* [index]
           +--rw index     -> ../config/index
           +--rw config
           |  +--rw index?         uint32
           |  +--rw description?   string
           |  +--rw enabled?       boolean
           +--ro state
              ...
```

- Subscribe request (using gNMIc):

```
$ gnmic --address clab-hackfest5-r1 --port 6030 \
    --username admin --password admin --insecure \
    --encoding json_ietf subscribe \
    --path /interfaces/interface[name=Ethernet2]/state/
```

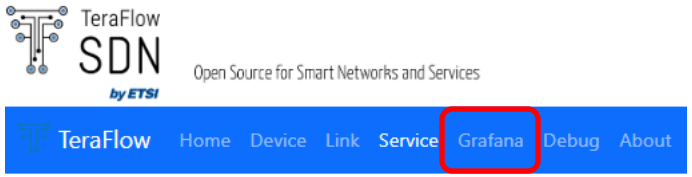
- Example of sample:

```
{
  "source": "clab-hackfest5-r1",
  "subscription-name": "default-1731326320",
  "timestamp": 1731326327859162620,
  "time": "2024-11-11T11:58:47.85916262Z",
  "prefix": "interfaces/interface[name=Ethernet2]/state/counters",
  "updates": [
    { "Path": "in-octets",          "values": { "in-octets": 40370       }},
    { "Path": "in-multicast-pkts", "values": { "in-multicast-pkts": 206 }},
    { "Path": "in-pkts",           "values": { "in-pkts": 211           }}
  ]
}
```

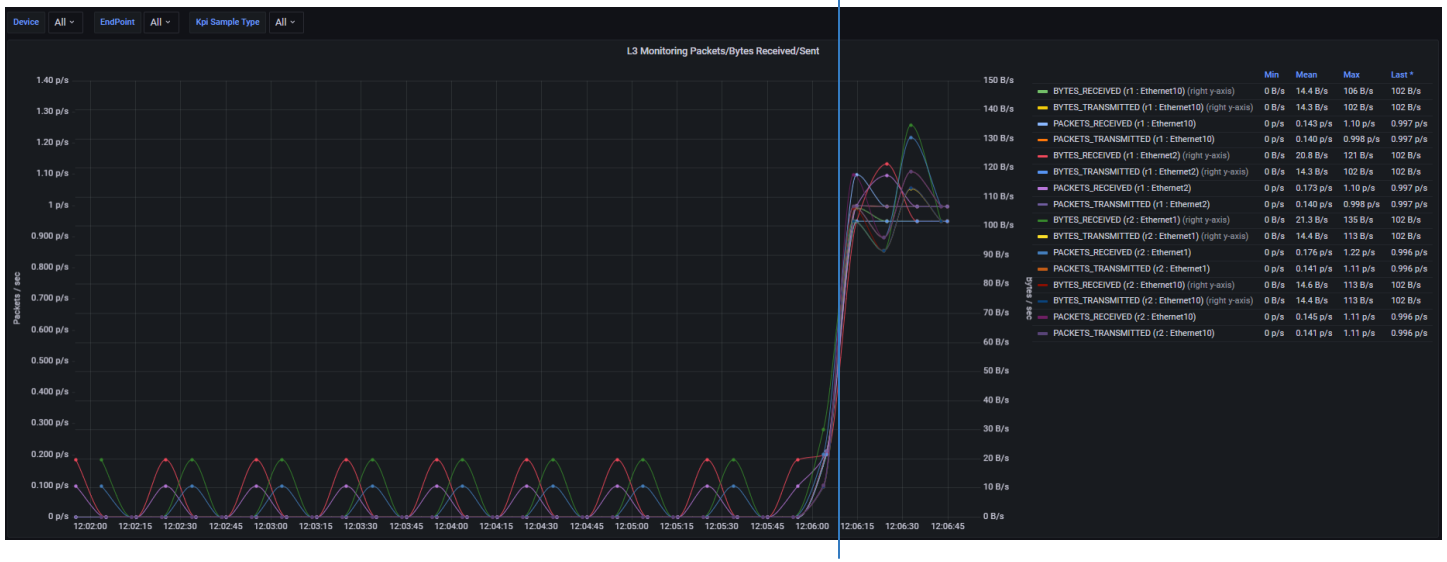
Using TeraFlowSDN to monitor the network traffic

- Telemetry collection should have been start automatically.
- Check Grafana dashboard:
  - Credentials: admin / admin123+
- Go to “L3 Monitoring” panel
- Connect to DC1 and start pinging DC2:

```
$ ~/tfs-ctrl/hackfest5/clab-cli-dc1.sh
  ping 192.168.2.10
```



Ping starts here
(~1 packet/sec, 110 bytes/sec)

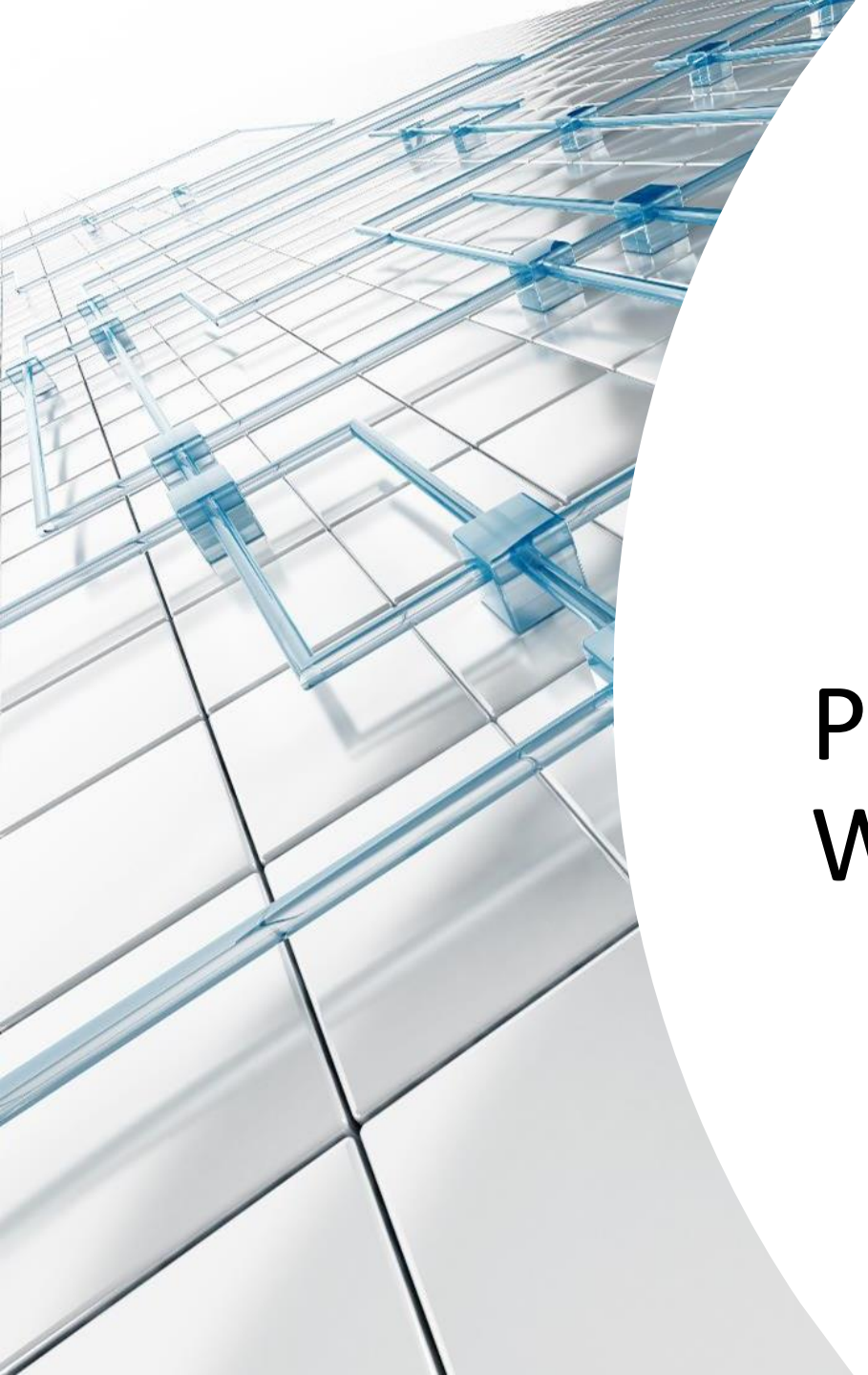


Too easy for you? Do you want more?

TeraFlow SDN

No problem! Here we go! ☺

- Extend the basic scenario with additional routers and clients
- Measure performance of ContainerLab using iperf3
- Establish and monitor multiple parallel services
- Browse the list of open issues in ETSI TeraFlowSDN GitLab Repository and consider joining the developers' team!
  - https://labs.etsi.org/rep/tfs/controller/-/issues



TeraFlow SDN

PART II
What is hot on TFS community?

ENABLER: TFS Demos and Use Cases

- ETSI OpenSourceMANO and ETSI TeraFlowSDN integration
- Transport Network Slicing with SLAs
- Slice Grouping for Transport Network Slices
- Hierarchical Multi-domain SDN Controllers
- Unsupervised Anomaly Detection Loop for Optical Networks
- Scalable and Efficient Pipeline for ML-based Optical Network Monitoring
- DLT-based End-to-end Inter-domain Transport Network Slice with SLA Management
- P5: Event-Driven Policy Framework for P4-Based Traffic Engineering
- TeraFlowSDN as an Optical SDN Controller
- DataPlane-in-a-box with ContainerLab
- ETSI MEC PoC 14 Network resource allocation
- Enabling Cloud AR/VR Gaming Services by an Open-Source and Standards-Based Network-as-a-Service Platform for Control and Management of Optical Networks
- IntentLLM: An AI Chatbot to Create, Find, and Explain Slice Intents in TeraFlowSDN

Towards R5 and beyond

TeraFlow SDN

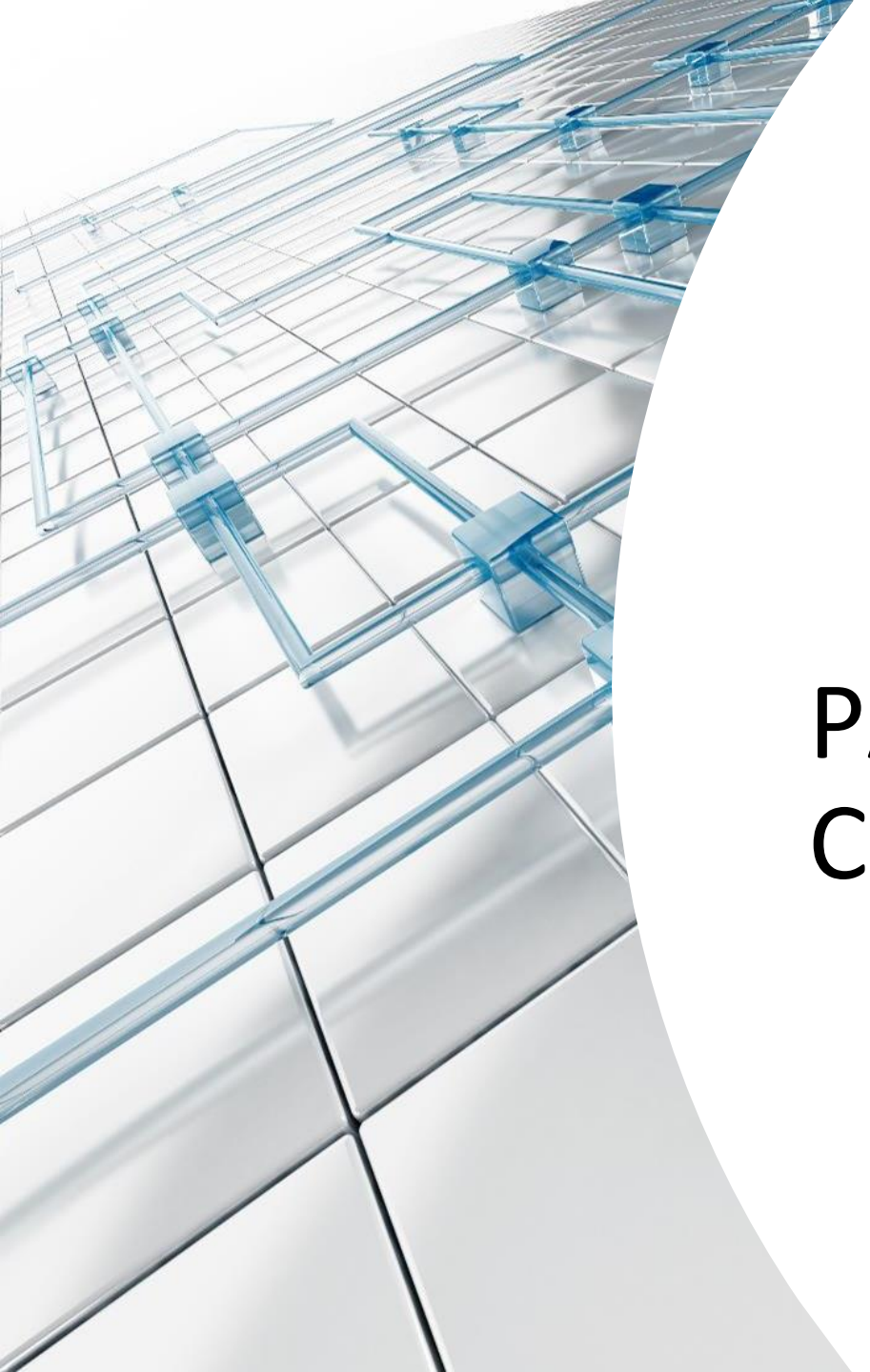
INVENTORY

INTENT-BASED NETWORKING

NETWORK DIGITAL TWIN

ENERGY OPTIMIZATION

IDS GATEWAY

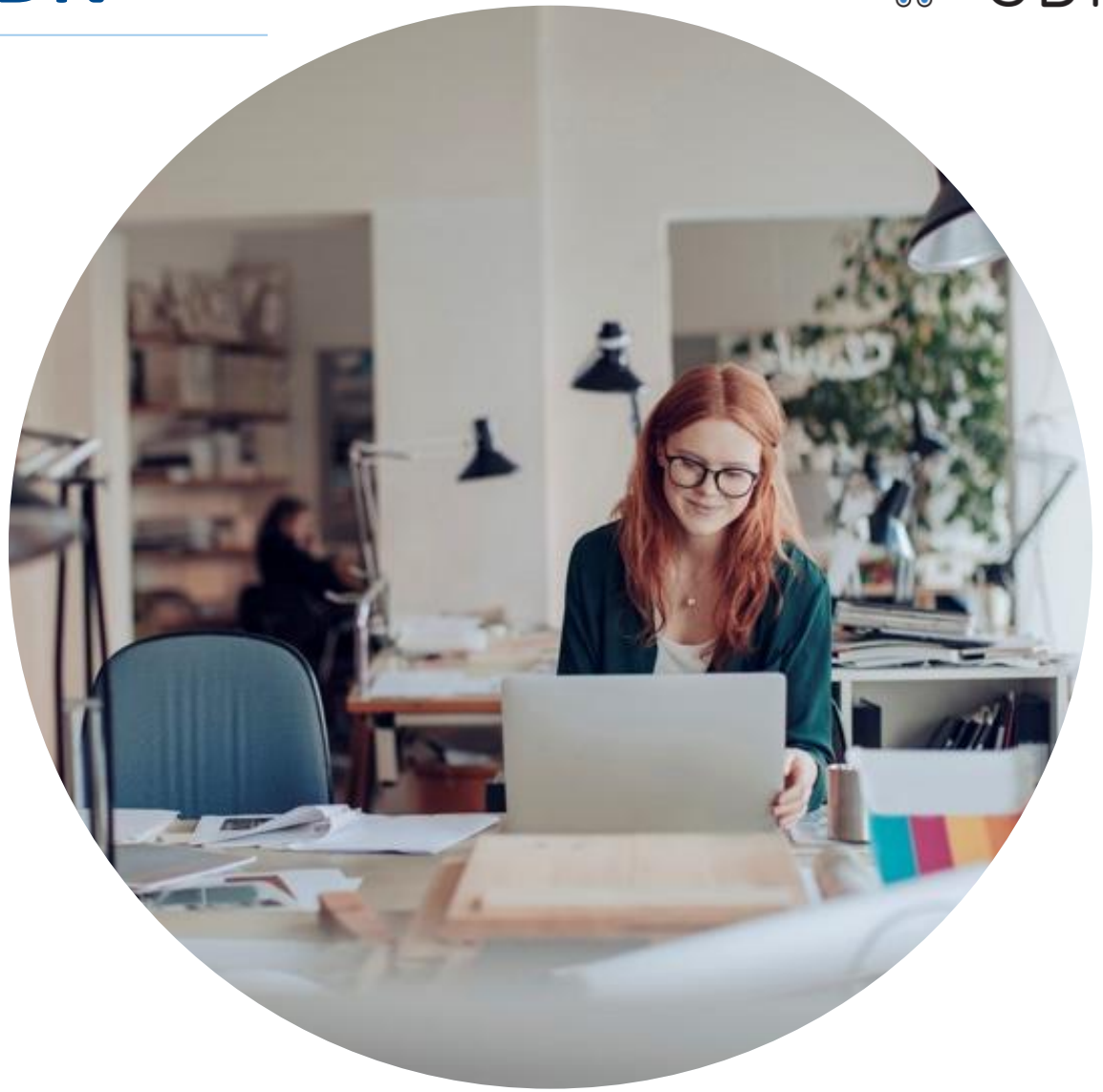


TeraFlow SDN

PART II
Contributing & Joining TFS

Contributing to **ETSI TeraFlowSDN**

- Representing your organization (employer, university...)
  - Organization signs the SDG TFS Agreement
  - Person creates an ETSI Online Account
  - See list of TFS Members and Participants
- As an individual
  - Create an Individual Account in the ETSI Labs
  - Manage your Individual Account and become and Individual Contributor (developer) by accepting the SDG TFS ICLA
- Contact SDGsupport@etsi.org for help



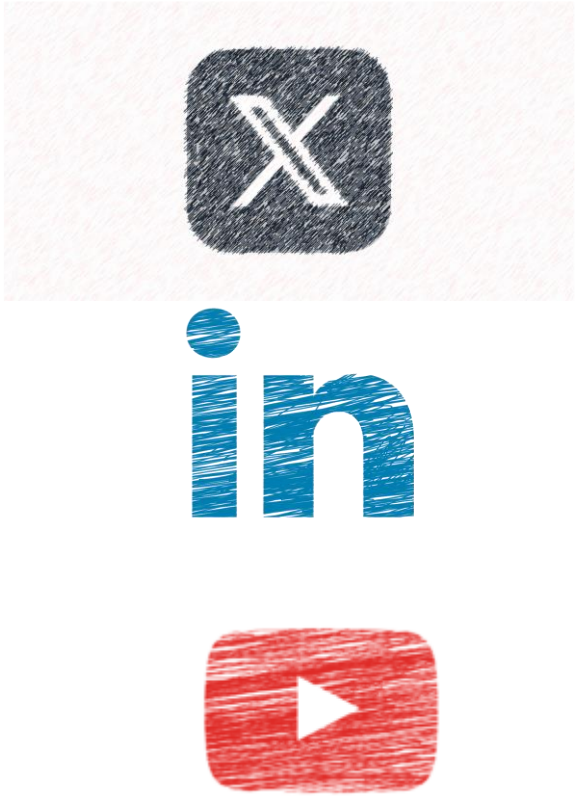
If you enjoy the **ETSI TeraFlowSDN** experience..

TeraFlow SDN

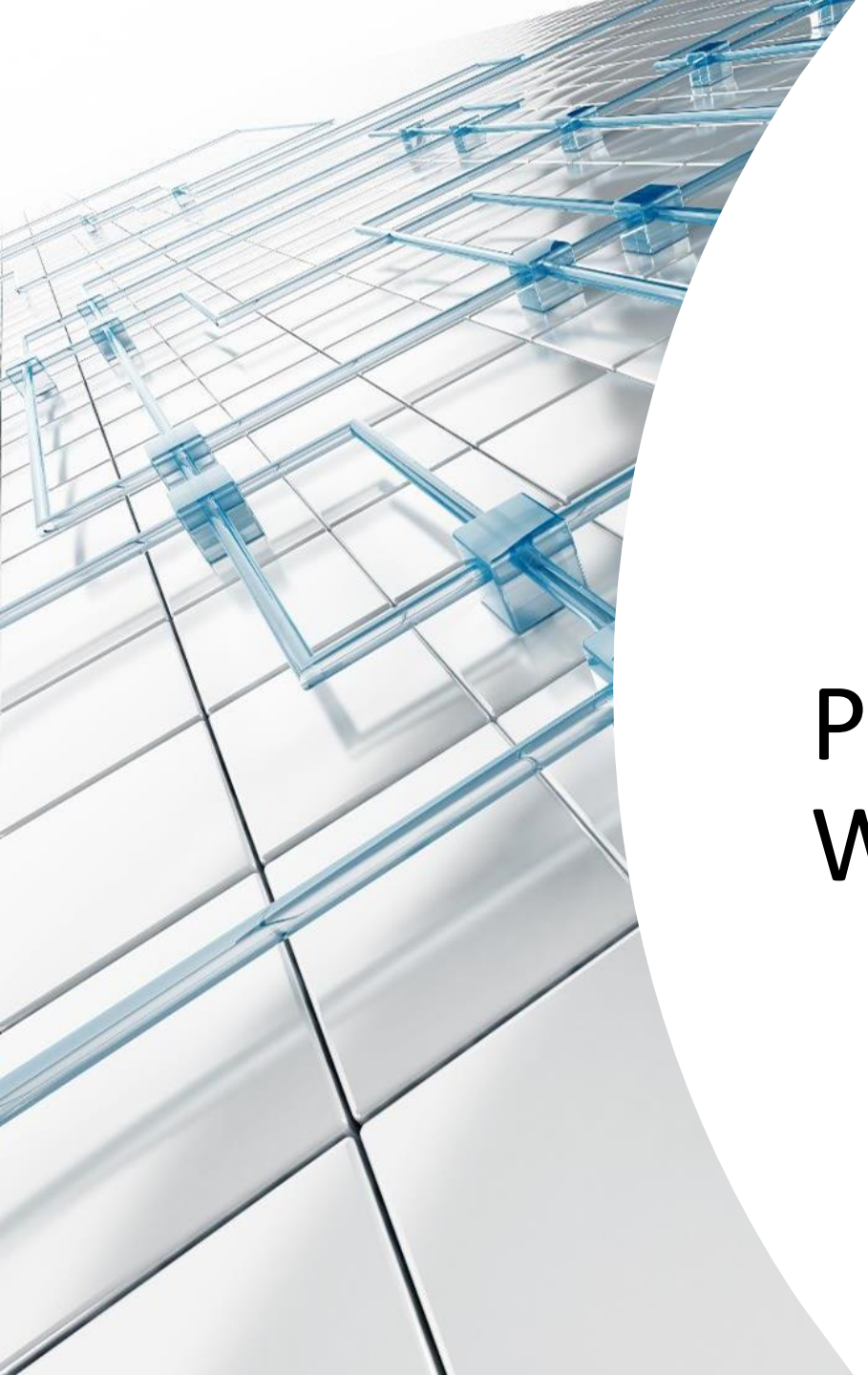
Join us!

- Participation is open to ETSI members, non-members, individual contributors and users…
- Free of charge for
  - ME/SME
  - Universities
  - Public Research
  - User and Trade Associations
  - Individuals
- Learn how to join
- Contact SDGsupport@etsi.org for help

Stay tuned!



@TeraFlowSDN
#TFSHackfest
#SNS4SNS



TeraFlow SDN

PART II
Wrap-up

Thank you for your attention!

- **Satisfaction survey**
  - Please take 2 minutes to answer and leave us your comments, your feedback is precious!

- **Certificates** of **participation**
  - Check your inbox!

TeraFlow SDN by ETSI

Thank you very much!