Tutorial
Breaking the Paradigm of Traffic Engineering Automation

NANOG90 - FEB-2024

Diego Achával

David Watkinson

bio.site/dwatkinson

Mau Rojas

bio.site/pinrojas

Abstract

CONTAINERlab

- Challenges conventional network automation by highlighting an innovative, 'out-of-the-box' approach to traffic engineering.
- Inventive alternative that combines API calls with standard systems such as PCE controllers and SDNs.
- Illustrated through a hands-on lab using Containerlab, YANG, and Python.
- We'll showcase an enhanced Label Switched Paths (LSP) management use case, demonstrating how it can solve the unpredictable patterns of today's traffic demands in WAN environments, particularly under RSVP-TE protocols.
- Participants will gain a thorough understanding of this fresh perspective on traffic engineering automation.
- We'll discuss how the selection of standard YANG models like IETF-TE versus vendor-specific models.
- We'll explore the range of network configuration options, from simple Python modules to more advanced SDN solutions, preparing participants to navigate its evolving landscape.

Agenda

- Traffic-Engineering Automation Overview [Diego]
  - Advantages Over Non-IETF RSVP Extended Protocols
- Telemetry in TE Automation [David/Mau]
  - Integrating PCE with Telemetry (gRPC) for a Closed Loop System
- Hands-On Lab: Automation Driven Traffic Steering [Mau]
  - Core Lab Components
  - Tool Comparisons (e.g., Nornir vs SDN)
  - Building Python Application Components
  - Utilizing IETF-TE YANG RESTCONF for Tunnel Management

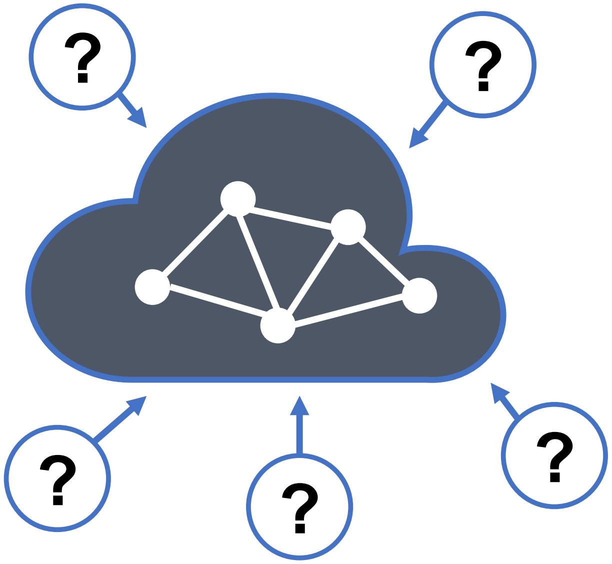
Automation in Traffic Engineering

Diego Achával

What is the problem we're trying to solve?

MPLS (RSVP-TE/SR-TE) is a great solution for **static or even stationary traffic patterns**

- However, traffic patterns become more **unpredictable and bursty** with changes due to factors such as 5G, streaming and cloud services.
- The use of multiple LSPs, balanced via ECMP, brings better **network utilization and resilience**.
  - LSPs are not elements that can be added or **removed on-demand**.



Benefits

	Demo App: APIs - IETF-TE	Current solutions running in the box
Network Technologies	RSVP-TE and SR-TE	RSVP-TE
Resilience	Better: Can be routed via different nodes and links using advanced algorithms from the PCE	Weak: Take the same route of the original one.
Resource Allocation	Better: Consider various factors, such as network topology, link and node capacities, traffic demands, and QoS requirements (PCE)	May not adapt well to changing network conditions (limited to RSVP reservation only)
Visibility	Better: Can be managed via PCE-GUI to compare LSP routes. Easier Troubleshooting	Limited. Relay on external NMS.
Trigger	Bandwidth Utilization and External Events... **and else**	Bandwidth Utilization Only

Telemetry in the context of TE Automation

David Watkinson

bio.site/dwatkinson

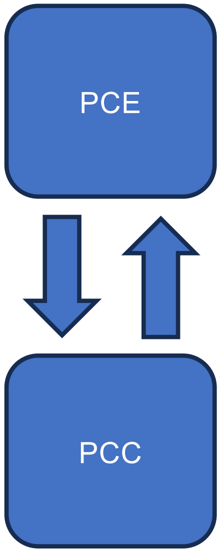
Mau Rojas

bio.site/pinrojas

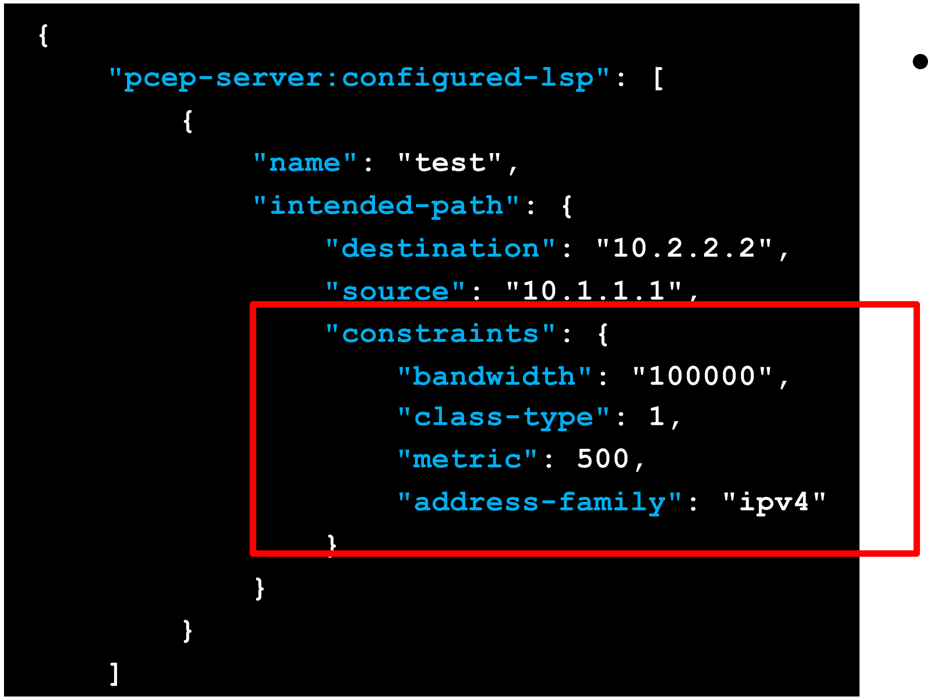
Intro PCE and PCC init Paths

Active Stateful PCE: PCC allows the LSP to be delegated to PCE or a PCE can initiate an LSP.

- PCE Init: PCE initiates an LSP and maintains the responsibility of updating the LSP.
- PCC Init: PCC initiates the LSP and delegate the control later to the PCE.



PCE Initiated Path

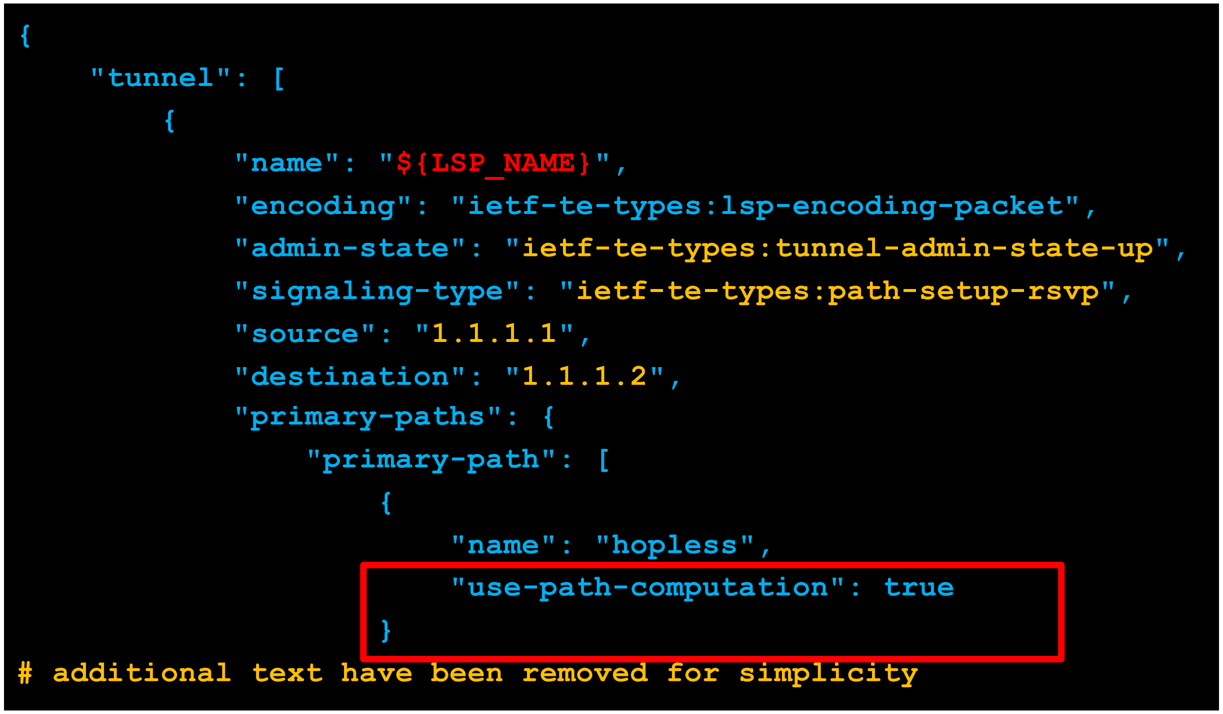


```
{
    "pcep-server:configured-lsp": [
        {
            "name": "test",
            "intended-path": {
                "destination": "10.2.2.2",
                "source": "10.1.1.1",
                "constraints": {
                    "bandwidth": "100000",
                    "class-type": 1,
                    "metric": 500,
                    "address-family": "ipv4"
                }
            }
        }
    ]
```

- Example of Simple Path Definition via PCE Controller APIs and Constraints

Source: https://docs.opendaylight.org

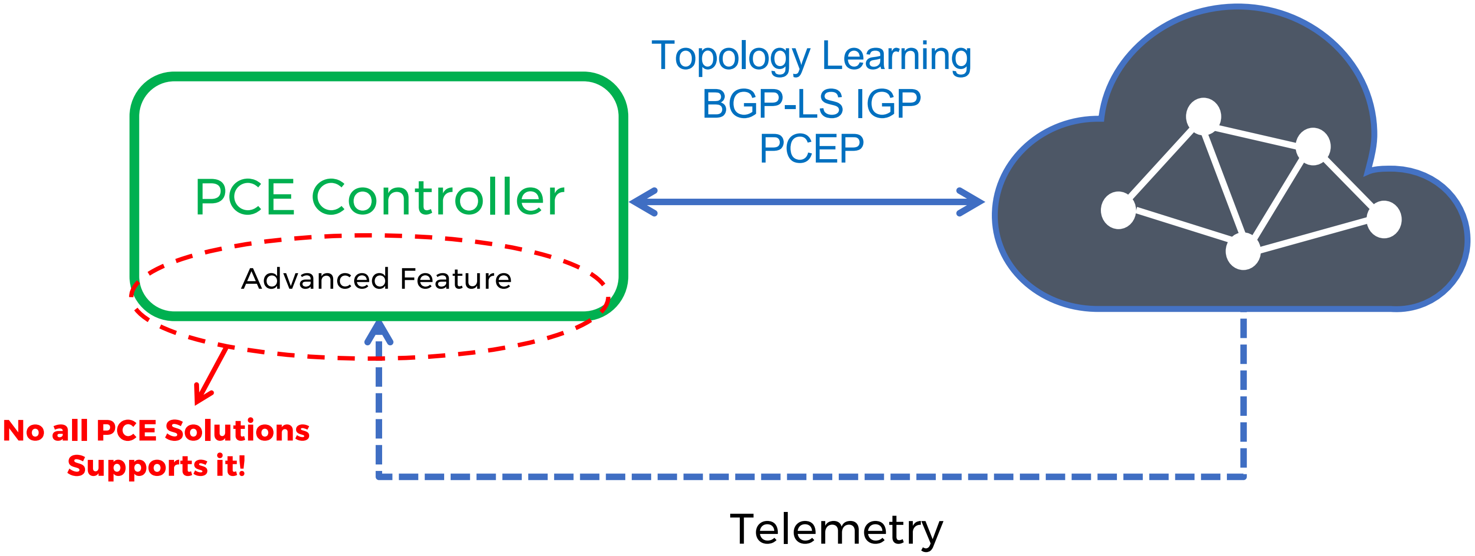
IETF-TE YANG: PCC Initiated Path



```
{
    "tunnel": [
        {
            "name": "${LSP_NAME}",
            "encoding": "ietf-te-types:lsp-encoding-packet",
            "admin-state": "ietf-te-types:tunnel-admin-state-up",
            "signaling-type": "ietf-te-types:path-setup-rsvp",
            "source": "1.1.1.1",
            "destination": "1.1.1.2",
            "primary-paths": {
                "primary-path": [
                    {
                        "name": "hopless",
                        "use-path-computation": true
                    }
# additional text have been removed for simplicity
```

NANOG™

Closed Loop: PCE + Telemetry



NANOG™

PCE+Telemetry: Example Use Case

Managing 5G Backhaul Traffic Scenario

- 5G backhaul network connecting 5G cell sites (gNBs) to the core network.
- Requires high bandwidth, low latency, and highly reliable
- Massive amount of data traffic generated by 5G services.

Challenge

- User mobility, varying service demands, and network conditions lead to congestion and performance degradation.
- Manage LSPs for optimal performance without causing instability (flip-flop behavior) or overloading the network with frequent path recalculations.

PCE+Telemetry: Example Use Case

Managing 5G Backhaul Tra

- 5G backhaul network connecting
- Requires high bandwidth, low late
- Massive amount of data traffic ge

Challenge

- User mobility, varying service dem and performance degradation.
- Manage LSPs for optimal perform or overloading the network with f

Standard Stateful PCE Policy

Limitations:

- **Reactivity:** The policy may not react quickly enough to sudden changes leading to temporary congestion.
- **Flip-Flop Behavior:** Frequent path recalculations can cause flip-flop behavior, degrading network stability.

NANOG™

PCE+Telemetry: Example Use Case

Managing 5G Backhaul Tra

- 5G backhaul network connecting
- Requires high bandwidth, low late
- Massive amount of data traffic ge

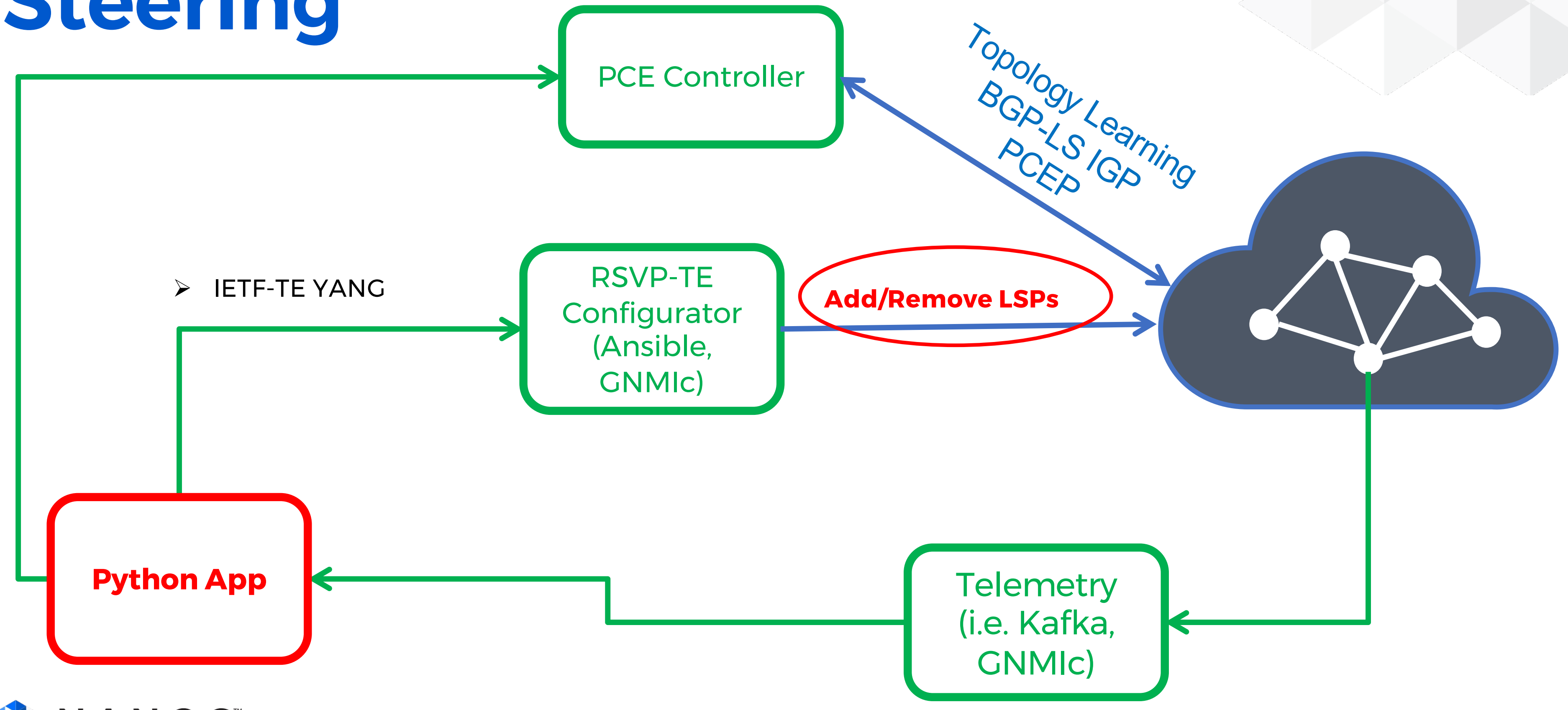
Challenge

- User mobility, varying service dem and performance degradation.
- Manage LSPs for optimal perform or overloading the network with f

Closed Loop with Telemetry

- **Comprehensive** view of the **network state**.
- **Analyzes telemetry data** to detect congestion before it impacts performance
- **De-emphasize minor fluctuations** that could cause flip-flop behavior. **Only significant and sustained changes** trigger rerouting.
- Properly tuned timers ensures that the network doesn't react to every small change, **avoiding frequent path recalculations** that could **overload devices**.

Demo: Automation Driven Traffic Steering

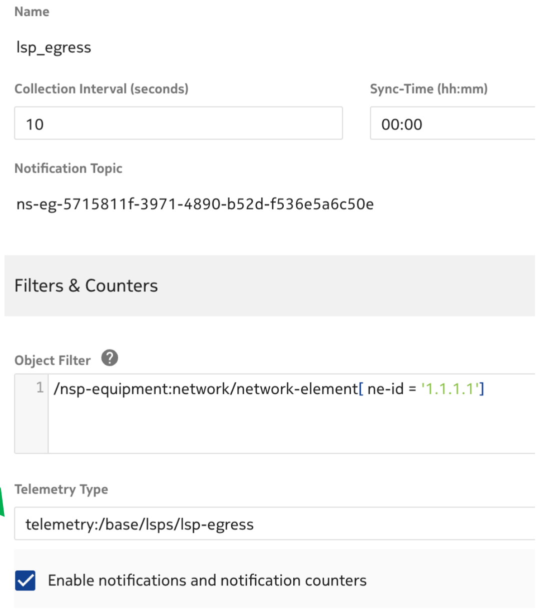


NANOG™

Enabling LSP Stats

pccLspTemplate.json

```
"nokia-conf:lsp" : {
... # this is an extract
        "egress-statistics": {
            "admin-state": "enable"
        },
        "type" : "p2p-rsvp",
        "from" : "10.10.10.3",
        "to" : "10.10.10.8",
        "pce-control" : "true",
        "pce-report" : "true",
        "path-computation-method" : "pce",
        "metric-type" : "te",
        "primary" : {
            "path-name" : "hopless"
        }
```



Name

lsp_egress

Collection Interval (seconds): 10

Sync-Time (hh:mm): 00:00

Notification Topic

ns-eg-5715811f-3971-4890-b52d-f536e5a6c50e

Filters & Counters

Object Filter

```
/nsp-equipment:network/network-element[ ne-id = '1.1.1.1']
```

Telemetry Type: telemetry:/base/lsps/lsp-egress

☑ Enable notifications and notification counters

Demo Lab: Automation Driven Traffic Steering

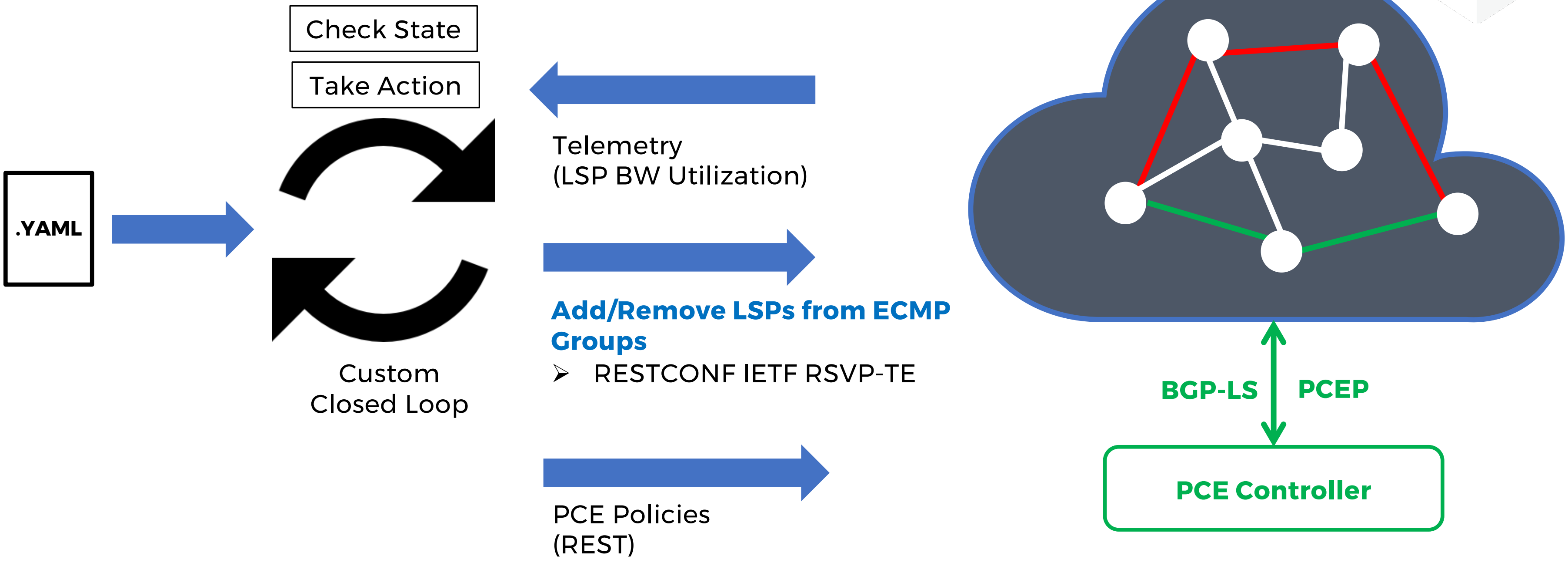
Mau Rojas

bio.site/pinrojas

Demo Lab: Automation Driven Traffic Steering: Introduction

- Inventive alternative with Python, involving API calls with systems like Telemetry and PCE controller, illustrated through a hands-on lab using containerlab.
- showcase an enhanced Label Switched Paths (LSP) management use case that can solve the unpredictable pattern of today's traffic demand in WAN environments
  - **IETF-TE YANG** (IETF-TE YANG)
- Check repo at: https://github.com/cloud-native-everything/nanog90-rsvpte-demo-lab (work in progress)

How does it work?



Check State

Take Action

.YAML

Custom
Closed Loop

Telemetry
(LSP BW Utilization)

Add/Remove LSPs from ECMP Groups

- ➢ RESTCONF IETF RSVP-TE

PCE Policies
(REST)

BGP-LS

PCEP

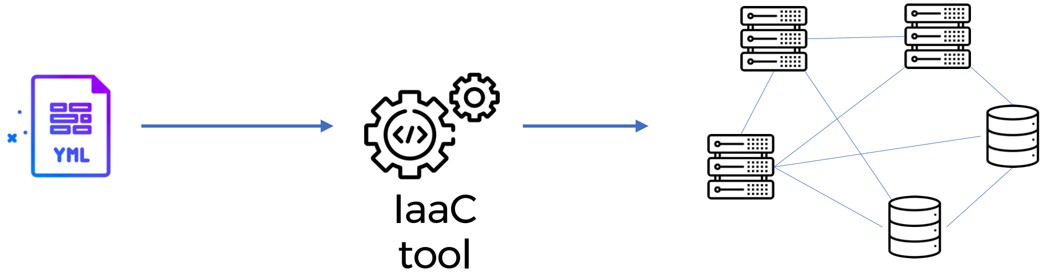
PCE Controller

How does it work?

- Multiple groups of RSVP-TE LSPs
  - Group set is balanced via ECMP
  - Every LSP is routed independently based on PCE Policies.
  - Any LSPs can be rerouted depending on performance constrains defined in the PCE policies
- Python App changes PCE Policies depending on user preferences
  - Policies can be linked to multiple groups
  - Combining multiple policies at once (i.e. link strict and star weight)
- Python App (Closed Loop) is constantly pulling Telemetry data
  - LSP Bandwidth Utilization
- Python App adds/removes LSPs in groups depending on Telemetry Thresholds.

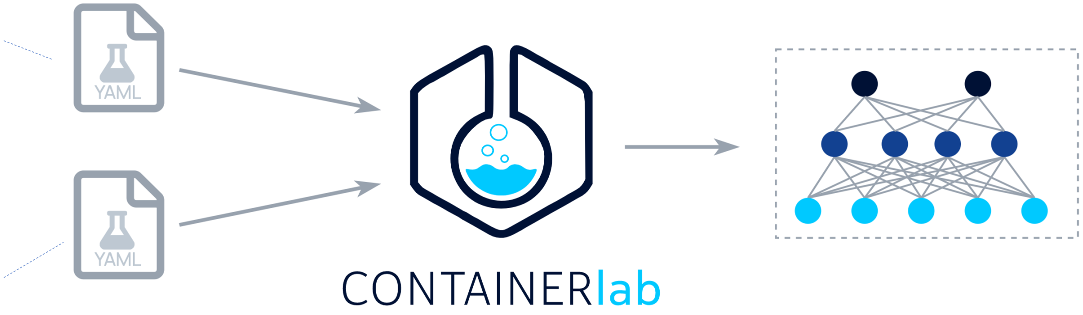
Clab: Bringing declarativeness to networking labs

IT

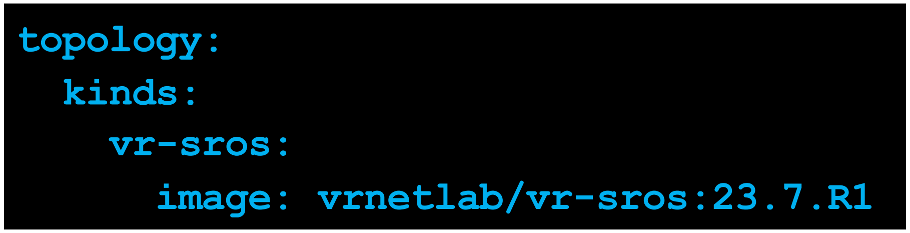


Network Labs

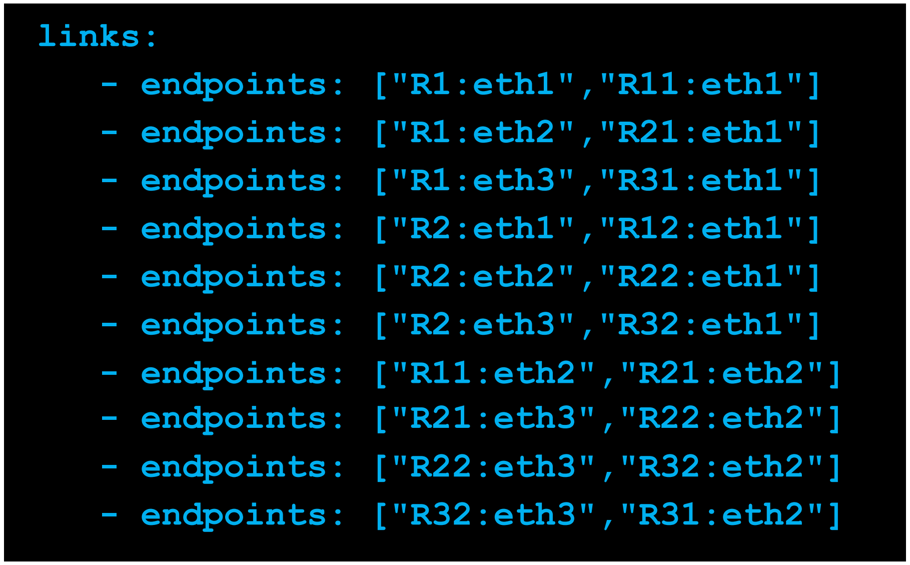
```
name: mylab

topology:
  nodes:
        :
        :

  links:
    -   :
```



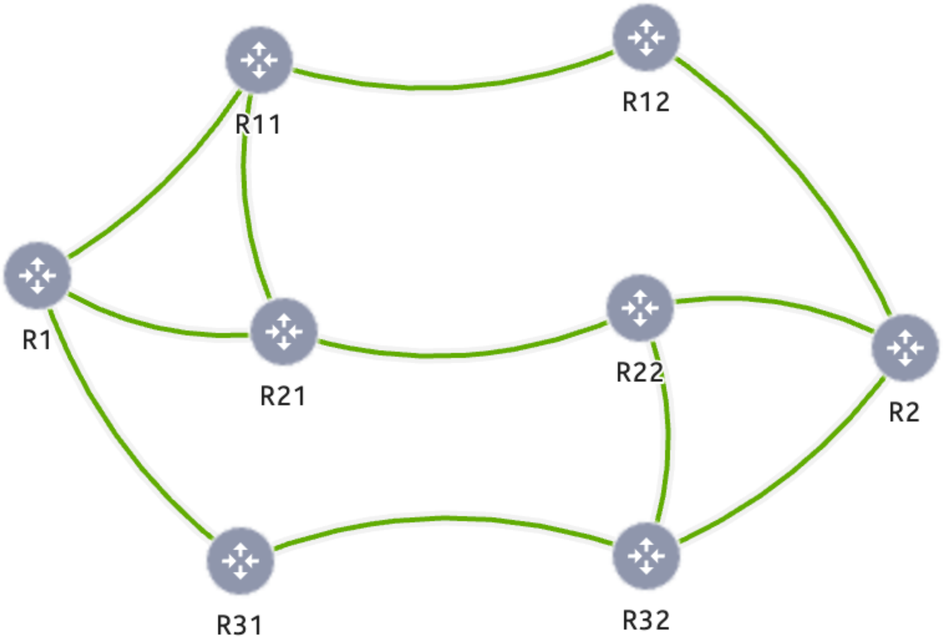
Clab Topology



```
topology:
  kinds:
    vr-sros:
      image: vrnetlab/vr-sros:23.7.R1
```

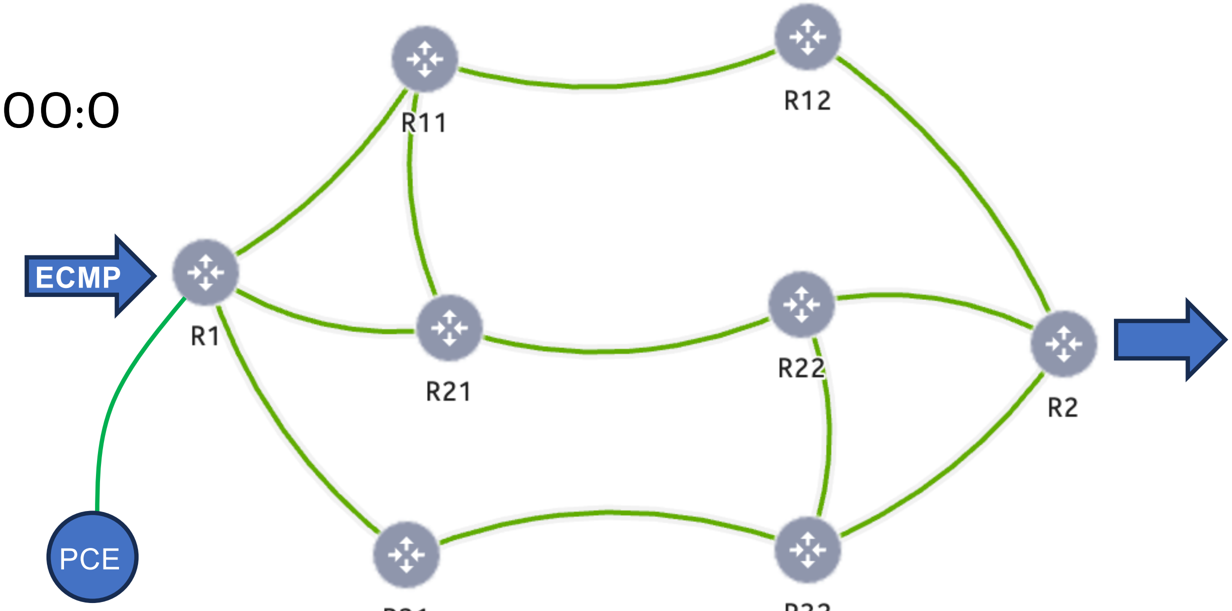


```
 links:
    - endpoints: ["R1:eth1","R11:eth1"]
    - endpoints: ["R1:eth2","R21:eth1"]
    - endpoints: ["R1:eth3","R31:eth1"]
    - endpoints: ["R2:eth1","R12:eth1"]
    - endpoints: ["R2:eth2","R22:eth1"]
    - endpoints: ["R2:eth3","R32:eth1"]
    - endpoints: ["R11:eth2","R21:eth2"]
    - endpoints: ["R21:eth3","R22:eth2"]
    - endpoints: ["R22:eth3","R32:eth2"]
    - endpoints: ["R32:eth3","R31:eth2"]
```

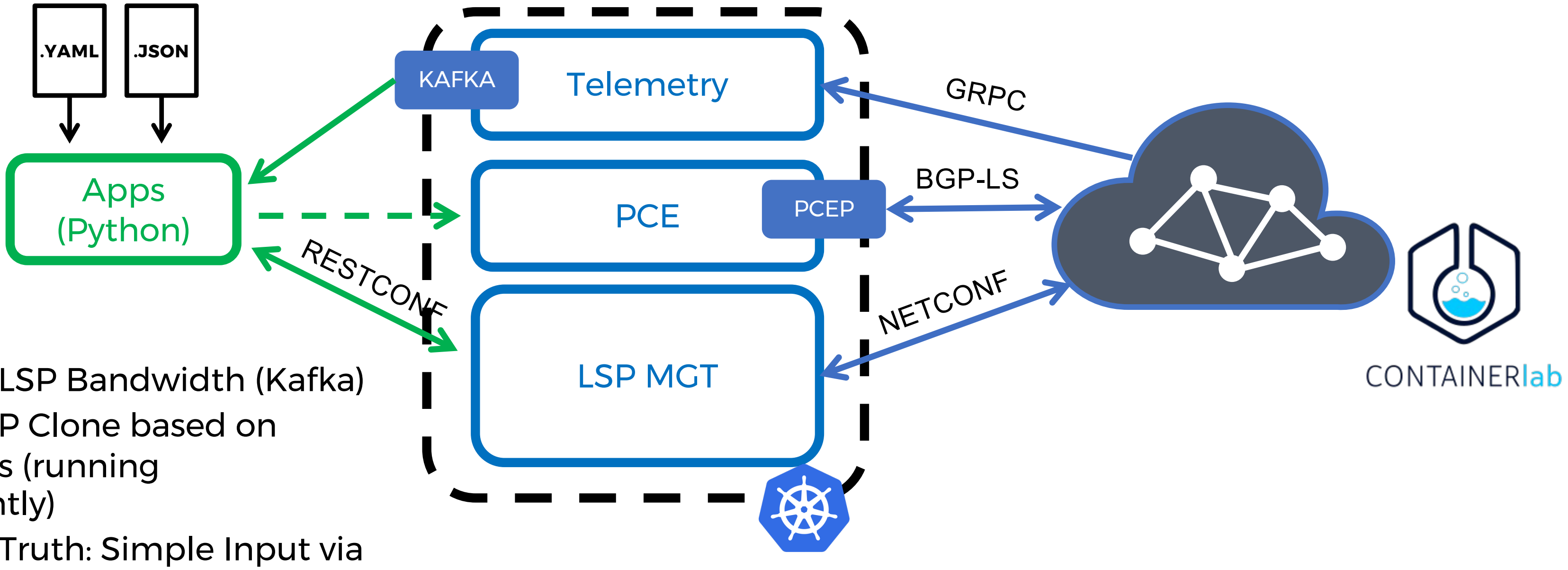


What network topology we use for this demo?

- 8 Routers (SROS 23.7.R1)
- ISIS Level 1
  - Topology Id 0:65000:0
- RSVP-MPLS



Lab Components



- Check on LSP Bandwidth (Kafka)
- Trigger LSP Clone based on Thresholds (running permanently)
- Source of Truth: Simple Input via YAML File

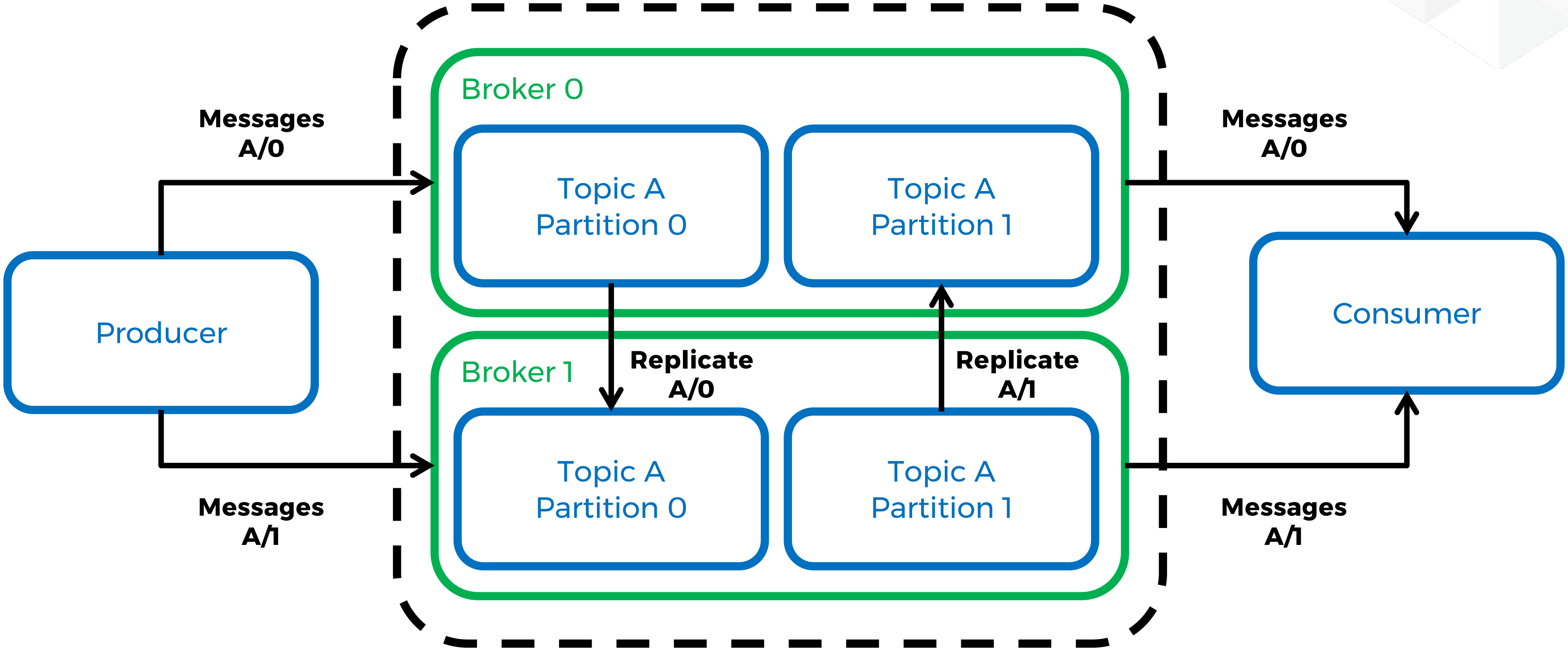
Lab Components

- Python v3.9.6
  - requests v2.31.0
  - pyyaml v6.0.1
  - vconfluent_kafka 2.2.0
- Configuration Management
  - IETF TE YANG RESTCONF (NSP 23.11 MD-Config + JSON Template)
    - draft-ietf-teas-yang-te
- Containerlab 0.45.1
  - RSVP-TE Topology (SROS vSIM 23.7.R1)
- Telemetry
  - Apache Kafka (NSP 23.11 CN Telemetry - GNMIc)
- PCE Controller
  - NSP 23.11 IP/MPLS Optimization

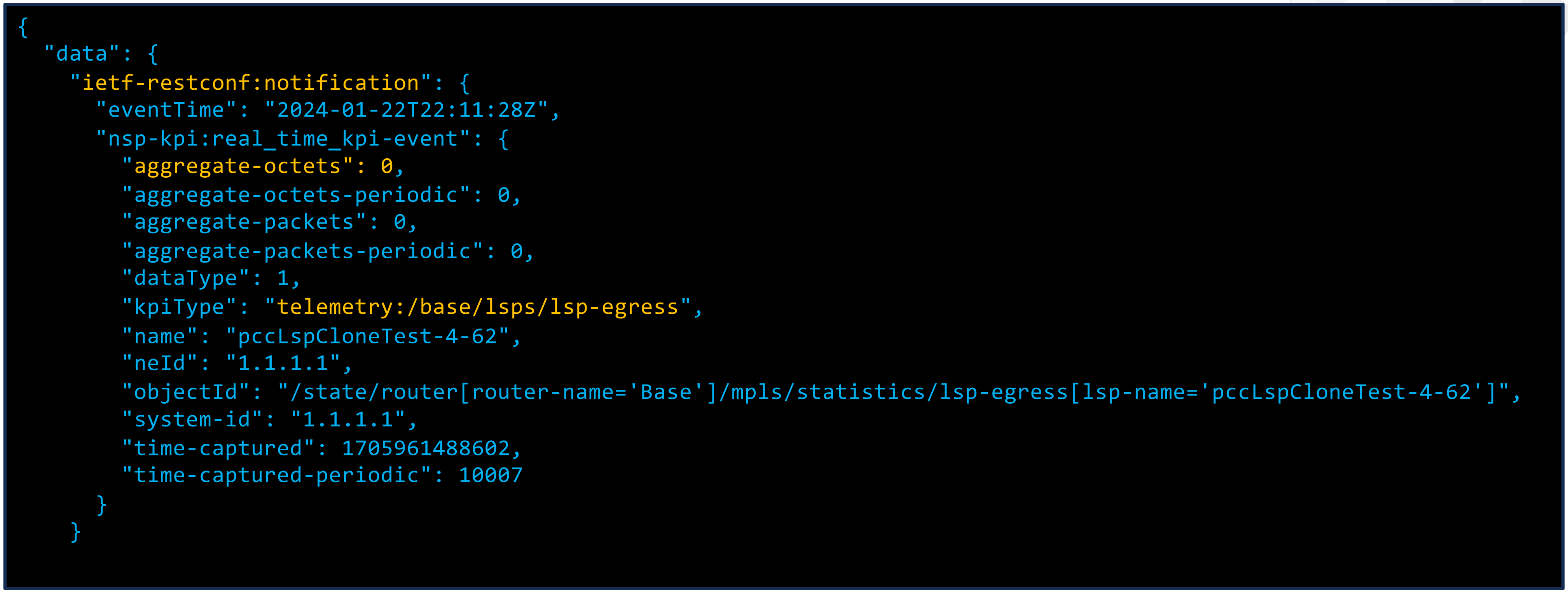
Apache Kafka

- Created by a team of software engineers at LinkedIn.
- Distributed streaming platform that excels in handling high-throughput data streams.
- Open-sourced in early 2011 under the Apache Software Foundation.
- Intended for processing large amounts of data in real-time
- Widely used in various industries for data integration, real-time analytics, and event-driven architectures.
- Decoupling of Data Sources and Destinations
  - Central hub for data streams, decoupling the source of data (like monitoring systems) from the consumers of data (like analytics systems, alerting systems, etc.)
  - This makes the architecture flexible and extensible.

Apache Kafka

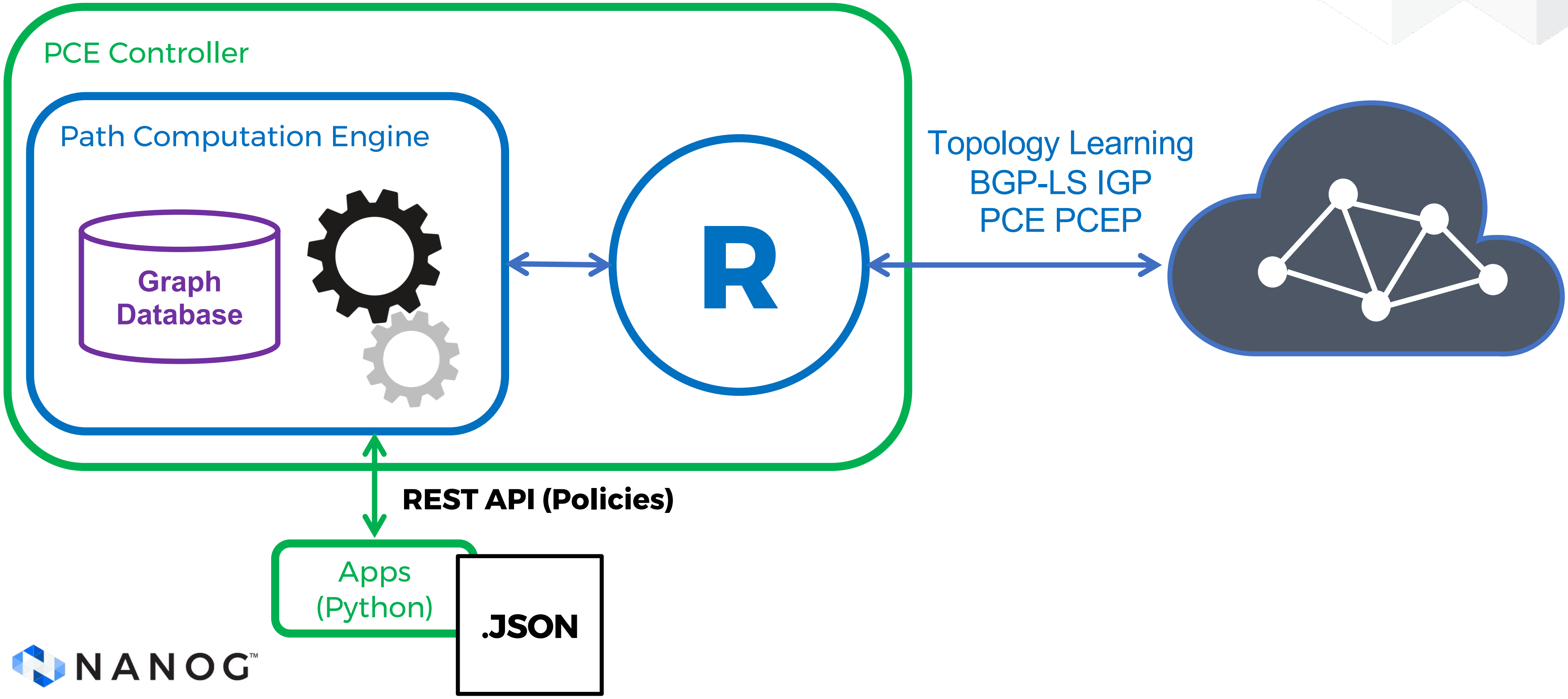


Apache Kafka



```
{
  "data": {
    "ietf-restconf:notification": {
      "eventTime": "2024-01-22T22:11:28Z",
      "nsp-kpi:real_time_kpi-event": {
        "aggregate-octets": 0,
        "aggregate-octets-periodic": 0,
        "aggregate-packets": 0,
        "aggregate-packets-periodic": 0,
        "dataType": 1,
        "kpiType": "telemetry:/base/lsps/lsp-egress",
        "name": "pccLspCloneTest-4-62",
        "neId": "1.1.1.1",
        "objectId": "/state/router[router-name='Base']/mpls/statistics/lsp-egress[lsp-name='pccLspCloneTest-4-62']",
        "system-id": "1.1.1.1",
        "time-captured": 1705961488602,
        "time-captured-periodic": 10007
      }
    }
```

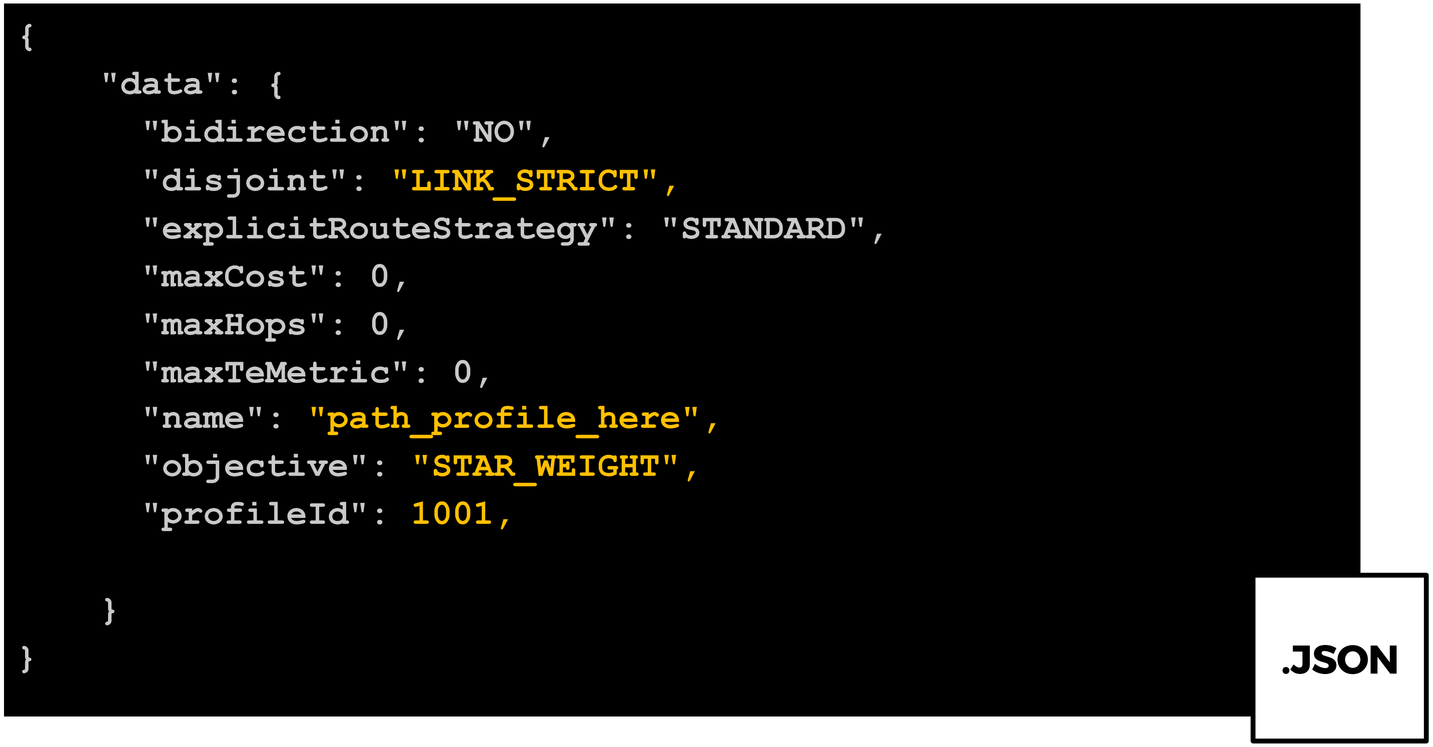
Common PCE Design



PCE Constraints and Objectives used for all paths

- **Bi-direction: NO**
  - Unidirectional paths only. The PCE will not attempt to compute reverse paths that mirror the forward paths
- **Disjoint: LINK STRICT**
  - Computed paths should be link-disjoint
- **Explicit Route Strategy: STANDARD**
  - PCE will follow typical routing protocols and algorithms without any special or customized routing considerations.
- **Max Cost: Undefined**
  - No upper limit cost of the path being computed (IGP Link Metric)
- **Max Hops: Undefined**
  - There is no restriction on the number of hops (or intermediary nodes) a path can have.
- **Max TE Metric: Undefined**
  - There is no upper limit on this metric for the path computation (TE Metric)
- **Metric Type: STAR_WEIGHT**
  - 'STAR_WEIGHT' is specific to Nokia's implementation and could involve a proprietary method of weighing different path attributes

Path Computation Policy Constraints/Objectives



```
{
    "data": {
      "bidirection": "NO",
      "disjoint": "LINK_STRICT",
      "explicitRouteStrategy": "STANDARD",
      "maxCost": 0,
      "maxHops": 0,
      "maxTeMetric": 0,
      "name": "path_profile_here",
      "objective": "STAR_WEIGHT",
      "profileId": 1001,

    }
}
```

Which LSP Configurator?

- Python Libs: pyGNMI or nccclient or nornir
  - **pyGNMI**: Best suited for environments where devices support the gNMI protocol.
  - **Nornir**: highly flexible and can be adapted to a wide range of network automation tasks, making it a good choice for diverse network environments.
- SDN: Large-scale, complex network environments where a full SDN controller can provide significant benefits in terms of network programmability, automation, and orchestration.
  - Nokia NSP MD Configurator
  - OpenDayLight??
- Invoke an external App client?
  - Example: GNMIc + GoTemplates

Configurator Comparison

Feature/Aspect	pyGNMI	Nornir	SDN: NSP / ODL
Protocol Used	gNMI (gRPC Network Management Interface)	Various (SSH, NETCONF, RESTCONF, etc.)	RESTCONF, NETCONF, others depending on plugins
Configuration Language	YANG models (JSON or Protobuf encoding)	Device-specific (CLI, XML, JSON for NETCONF/RESTCONF)	YANG models (XML or JSON encoding)
Target Devices	Devices supporting gNMI	Broad range of network devices (depends on plugin support)	Devices supported by plugins and drivers
Scalability	High (efficient binary protocol)	High (parallel execution capabilities)	Very high (SDN controller capabilities)
Customization	Limited to gNMI capabilities	Highly customizable with Python scripting	Customizable through applications and modules

Configurator Comparison (cont)

Feature/Aspect	pyGNMI	Nornir	SDN: NSP / ODL
Learning Curve	Moderate (knowledge of gNMI and YANG required)	Moderate (Python and networking knowledge)	High (complex SDN concepts architecture)
Deployment	Direct connection to devices	Direct connection to devices or through an intermediary	Requires SDN controller setup
Use Case Suitability	Ideal for environments with gNMI support	Versatile for various network automation tasks	Suitable for large-scale, complex network environments
Community and Support	Growing, with focus on gNMI-enabled devices	Large and active, diverse use cases	ODL: Large ODL, especially in carrier networks. NSP: Enterprise support, no community-based
Security	Secure (TLS/SSL for transport)	Depends on the protocol used (e.g., SSH, TLS/SSL for NETCONF/RESTCONF)	Secure (multiple security features in SDN controller)

Example: Nornir

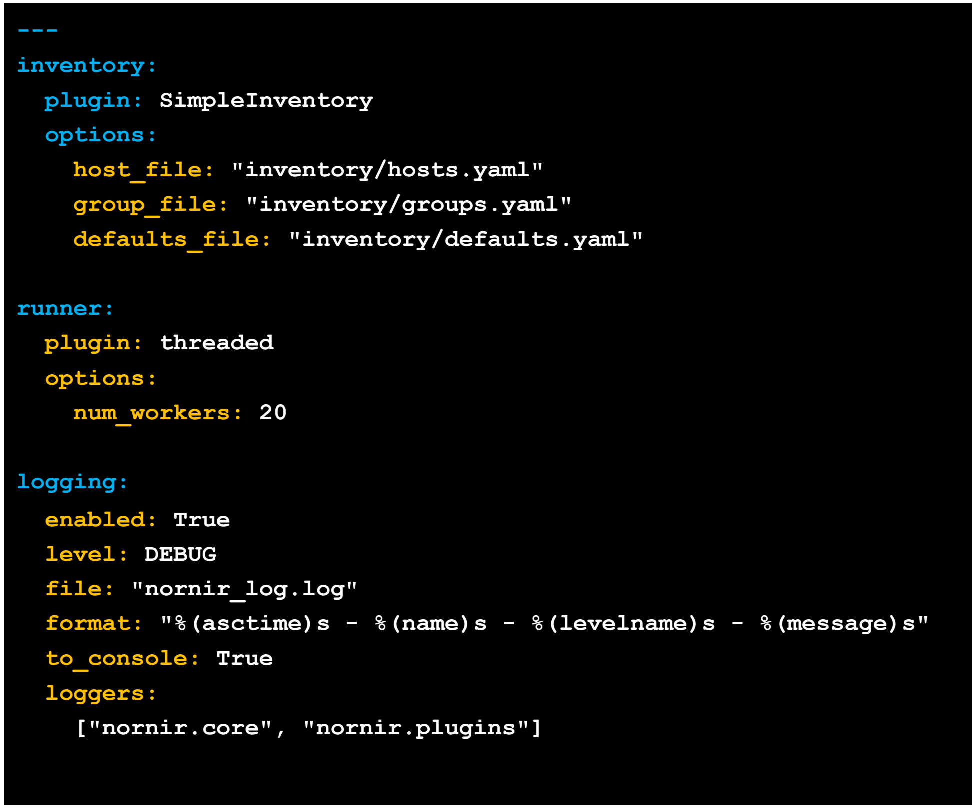
- Install Python Lib

```
user@host ~% pip install nornir nornir_netconf
```

- Import Modules

```
from nornir import InitNornir
from nornir_netconf.plugins.tasks import netconf_edit_config
from nornir_utils.plugins.functions import print_result
```

Example: Nornir (cont)



```yaml
---
inventory:
  plugin: SimpleInventory
  options:
    host_file: "inventory/hosts.yaml"
    group_file: "inventory/groups.yaml"
    defaults_file: "inventory/defaults.yaml"

runner:
  plugin: threaded
  options:
    num_workers: 20

logging:
  enabled: True
  level: DEBUG
  file: "nornir_log.log"
  format: "%(asctime)s - %(name)s - %(levelname)s - %(message)s"
  to_console: True
  loggers:
    ["nornir.core", "nornir.plugins"]
```

- inventory: Defines how Nornir will load its inventory.
  - host_file, group_file, and defaults_file are paths to YAML files that contain information about your network devices, groups of devices, and default settings, respectively.
- runner: Configures how tasks are executed.
  - threaded means tasks will be run in multiple threads for parallel execution.
  - num_workers defines the number of concurrent threads.

Example: Nornir (cont)

hosts.yml

```
---
router1:
  hostname: 192.168.1.1
  groups:
    - nokia_routers
  data:
    role: edge

router2:
  hostname: 192.168.1.2
  groups:
    - nokia_routers
  data:
    role: core
```

groups.yml

```
---
nokia_routers:
  username: admin
  password: admin
  platform: nokia_sros
  data:
    vendor: Nokia
    model: 7750
    os_version: "TiMOS-B-19.10.R1"
```

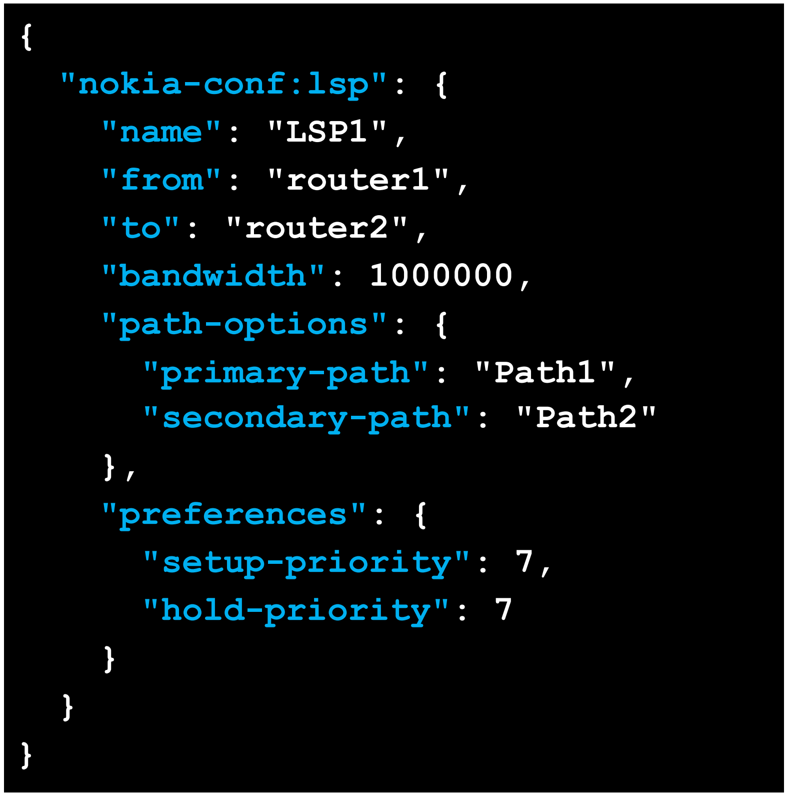
defaults.yml

```
---
username: admin
password: admin123
```

Example: Nornir (cont)

```
# Initialize Nornir with your configuration
nr = InitNornir(config_file="nornir_config.yaml")

def modify_lsp(task, operation, lsp_config):
    # Sending NetConf command to modify LSP
    task.run(
        task=netconf_edit_config,
        target="candidate",
        config=lsp_config
    )

    if operation == "add" or operation == "update":
        # Committing the changes if adding or updating LSP
        task.run(task=netconf_commit)
```

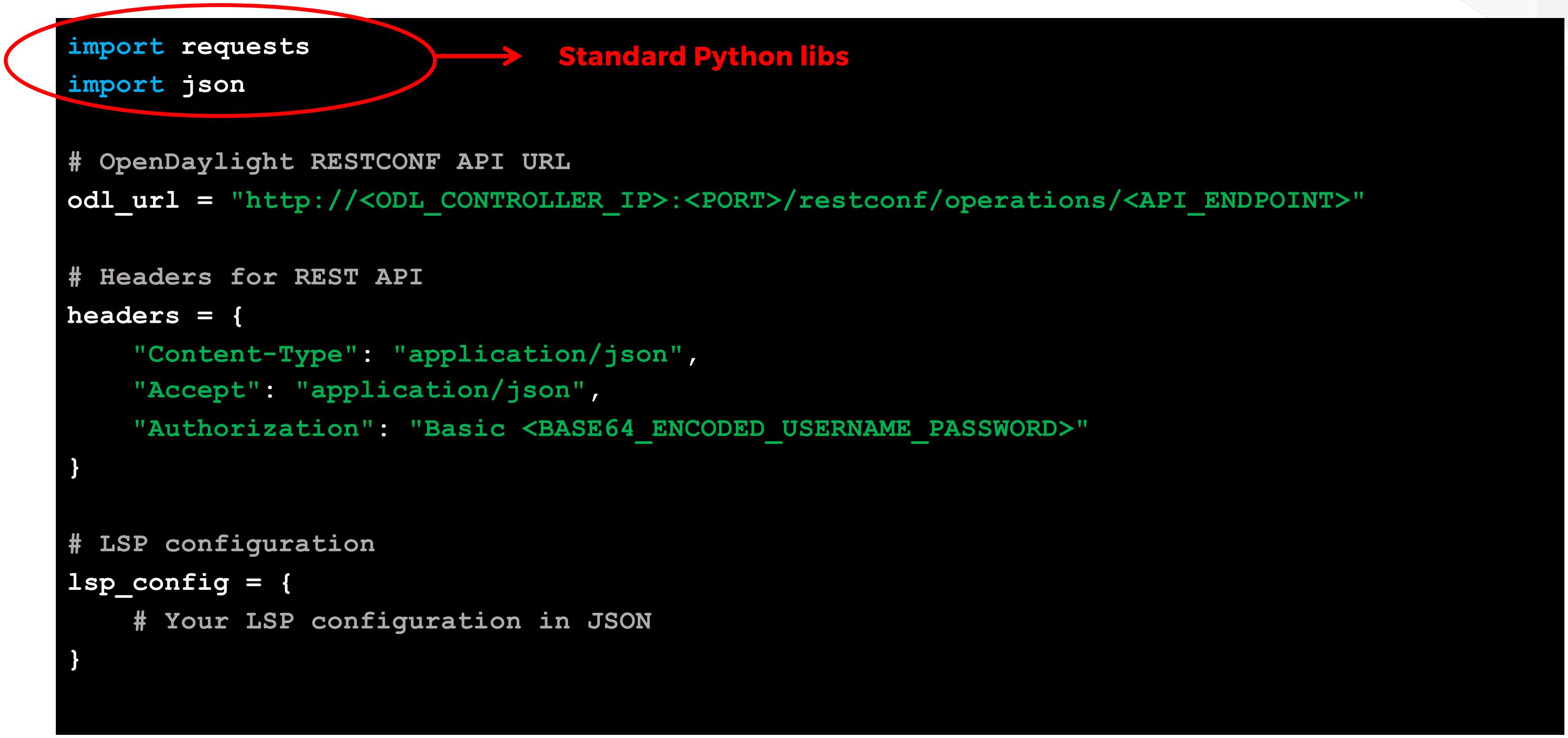
Example: Nornir (cont)



```
{
  "nokia-conf:lsp": {
    "name": "LSP1",
    "from": "router1",
    "to": "router2",
    "bandwidth": 1000000,
    "path-options": {
      "primary-path": "Path1",
      "secondary-path": "Path2"
    },
    "preferences": {
      "setup-priority": 7,
      "hold-priority": 7
    }
  }
}
```

Example of lsp_config_add (Hypothetical JSON for Adding an LSP in Nokia YANG Model)

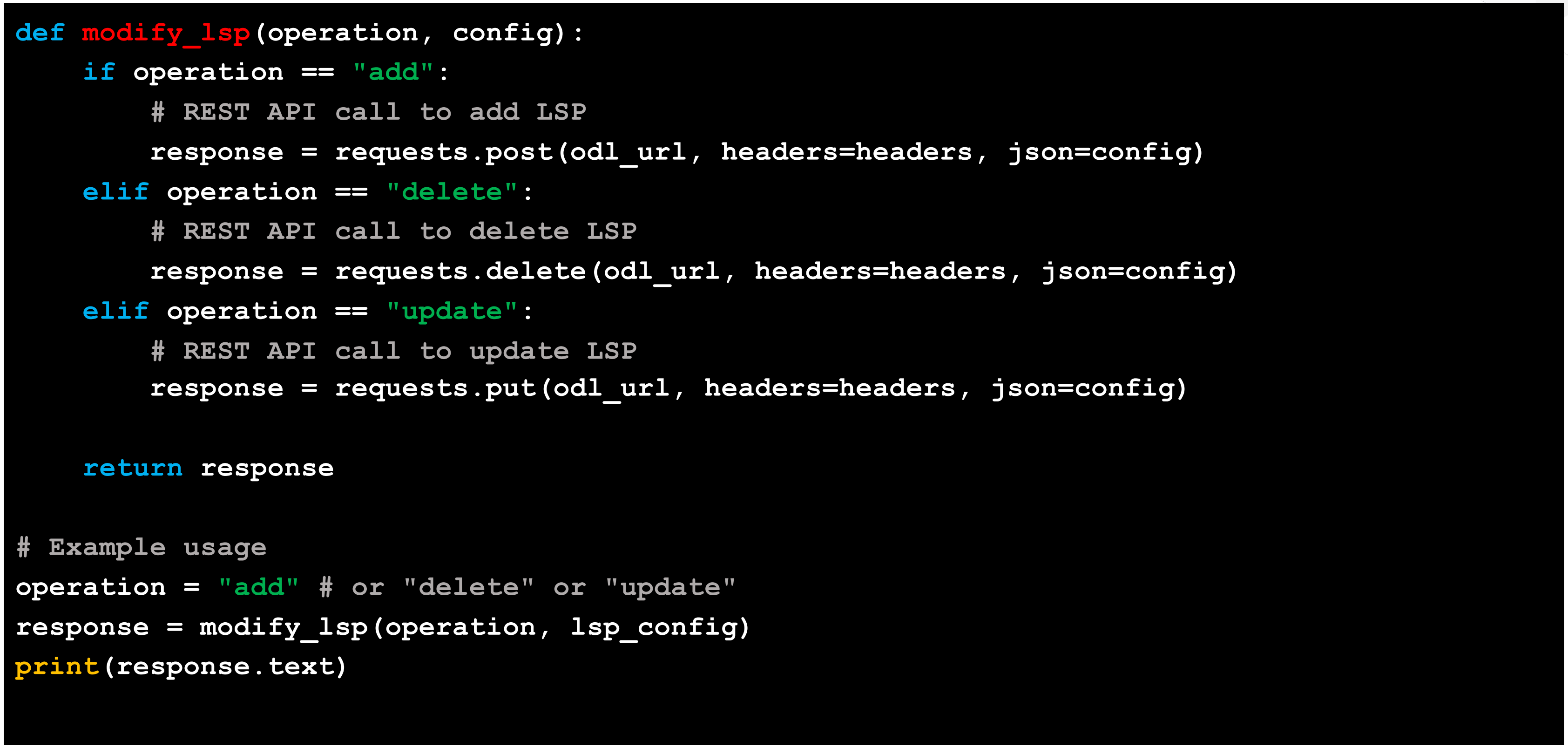
Example: OpenDayLight



Standard Python libs

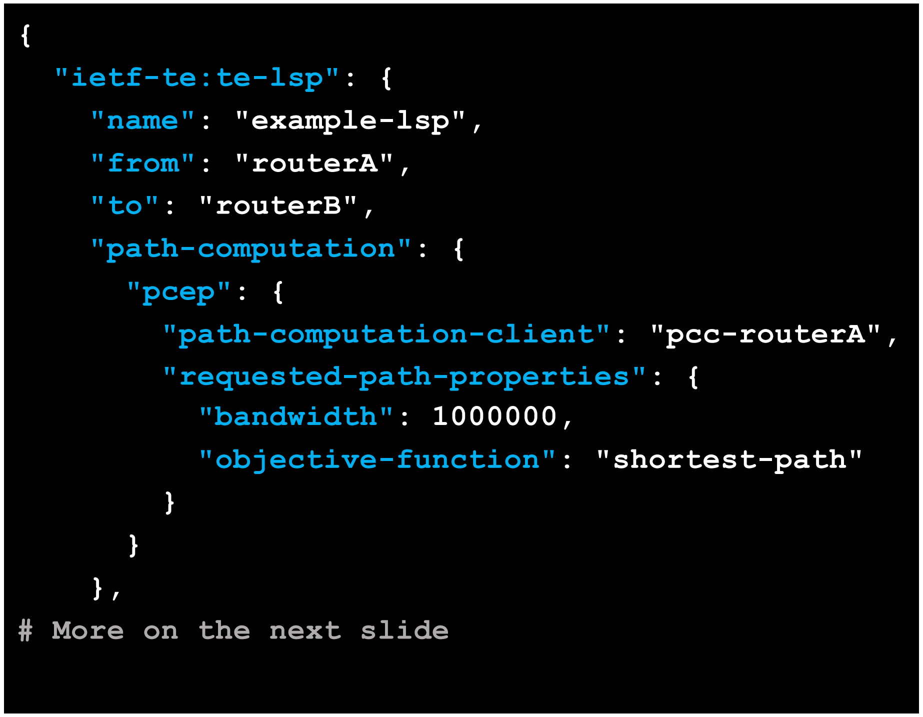
```python
import requests
import json

# OpenDaylight RESTCONF API URL
odl_url = "http://<ODL_CONTROLLER_IP>:<PORT>/restconf/operations/<API_ENDPOINT>"

# Headers for REST API
headers = {
    "Content-Type": "application/json",
    "Accept": "application/json",
    "Authorization": "Basic <BASE64_ENCODED_USERNAME_PASSWORD>"
}

# LSP configuration
lsp_config = {
    # Your LSP configuration in JSON
}
```

Example: OpenDayLight (cont)



```python
def modify_lsp(operation, config):
    if operation == "add":
        # REST API call to add LSP
        response = requests.post(odl_url, headers=headers, json=config)
    elif operation == "delete":
        # REST API call to delete LSP
        response = requests.delete(odl_url, headers=headers, json=config)
    elif operation == "update":
        # REST API call to update LSP
        response = requests.put(odl_url, headers=headers, json=config)

    return response

# Example usage
operation = "add" # or "delete" or "update"
response = modify_lsp(operation, lsp_config)
print(response.text)
```

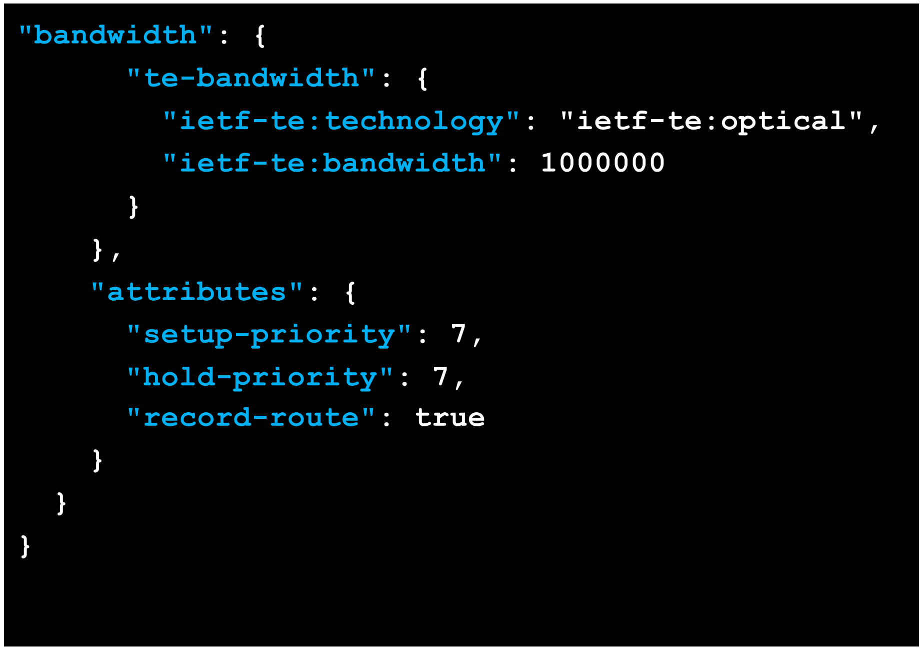
Example: OpenDayLight (cont)



```
{
  "ietf-te:te-lsp": {
    "name": "example-lsp",
    "from": "routerA",
    "to": "routerB",
    "path-computation": {
      "pcep": {
        "path-computation-client": "pcc-routerA",
        "requested-path-properties": {
          "bandwidth": 1000000,
          "objective-function": "shortest-path"
        }
      }
    },
# More on the next slide
```

- **IETF YANG model for RSVP-TE**, the JSON structure for the **lsp_config** indicates that path computation should be handled by a PCE.
- Important **Advantage** of SDN solutions is you can manage a common YANG model (i.e. IETF RSVP-TE) for all network elements and let the SDN controller to figure the way to set it up.

Example: OpenDayLight (cont)



```
"bandwidth": {
      "te-bandwidth": {
        "ietf-te:technology": "ietf-te:optical",
        "ietf-te:bandwidth": 1000000
      }
    },
    "attributes": {
      "setup-priority": 7,
      "hold-priority": 7,
      "record-route": true
    }
  }
}
```

- **IETF YANG model for RSVP-TE**, the JSON structure for the **lsp_config** indicates that path computation should be handled by a PCE.

- Important **Advantage** of SDN solutions is you can manage a common YANG model (i.e. IETF RSVP-TE) for all network elements and let the SDN controller to figure the way to set it up.

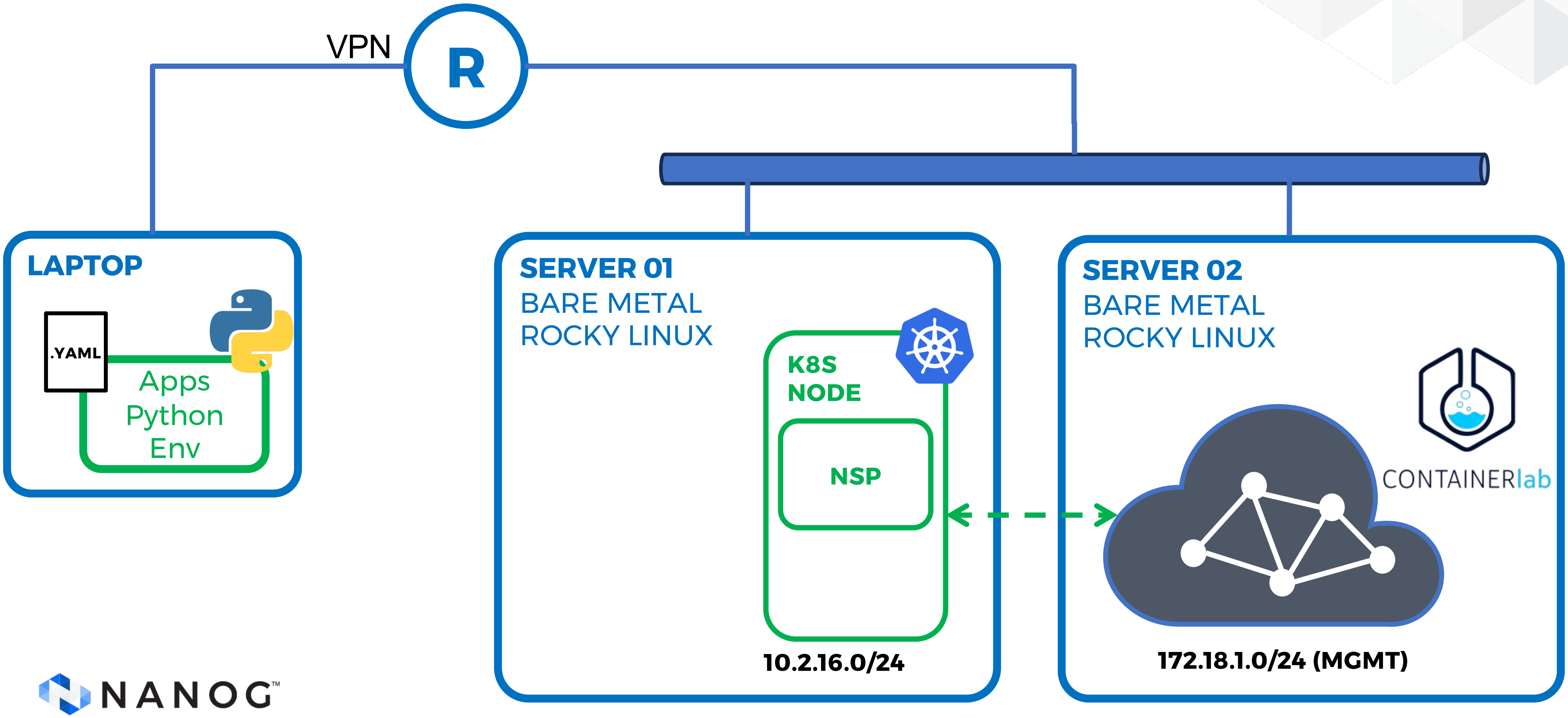
LSP Configurator

- Python Lib pyGNMI or nccclient or nornir
- Invoke an external App client?
  - Example: GNMIc + GoTemplates
- SDN:
  - **Nokia NSP**
    - **Single NorthBound** RESTCONF Interface
    - Python Lib (requests)
  - OpenDayLight??

I use this

IETF-TE YANG

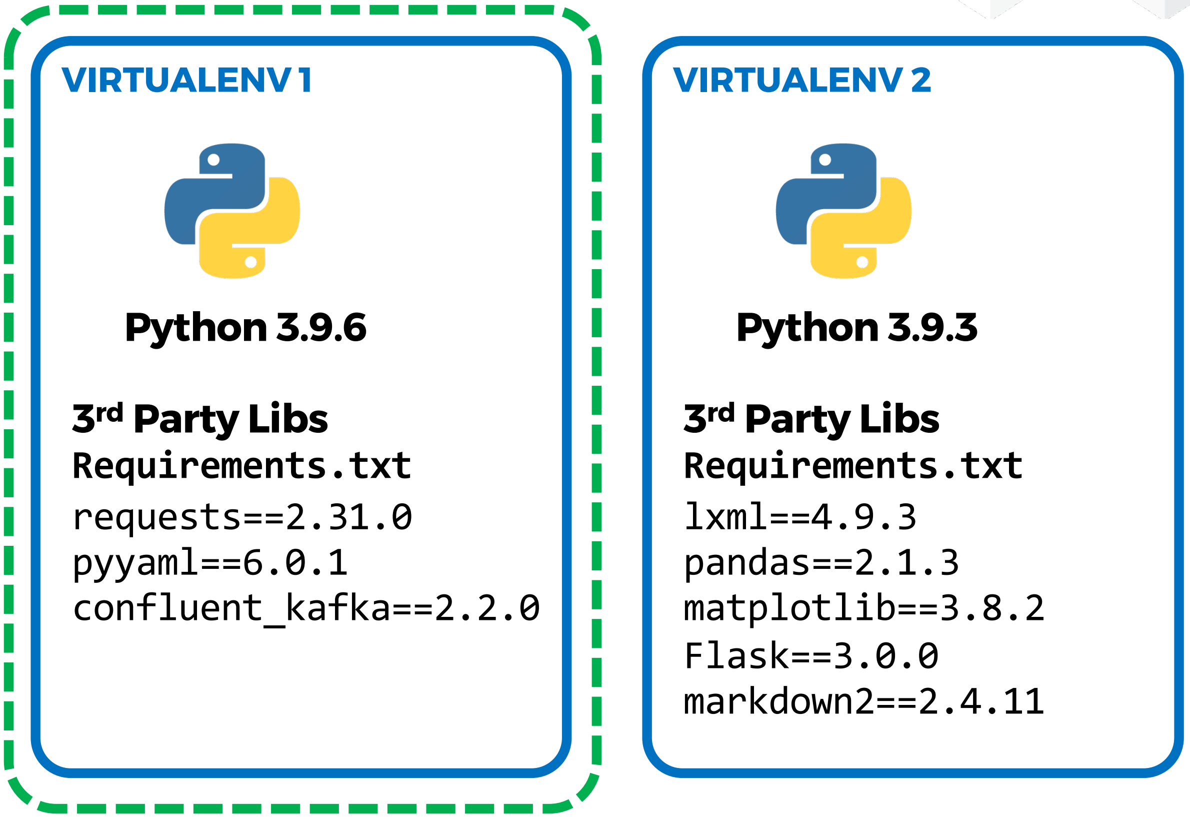
Lab Components



NANOG™

My Python Env

```
sudo python3 -m venv .venv
source .venv/bin/activate
pip3 install -r requirements.txt
```



VIRTUALENV 1

Python 3.9.6

3rd Party Libs

Requirements.txt

```
requests==2.31.0
pyyaml==6.0.1
confluent_kafka==2.2.0
```

VIRTUALENV 2

Python 3.9.3

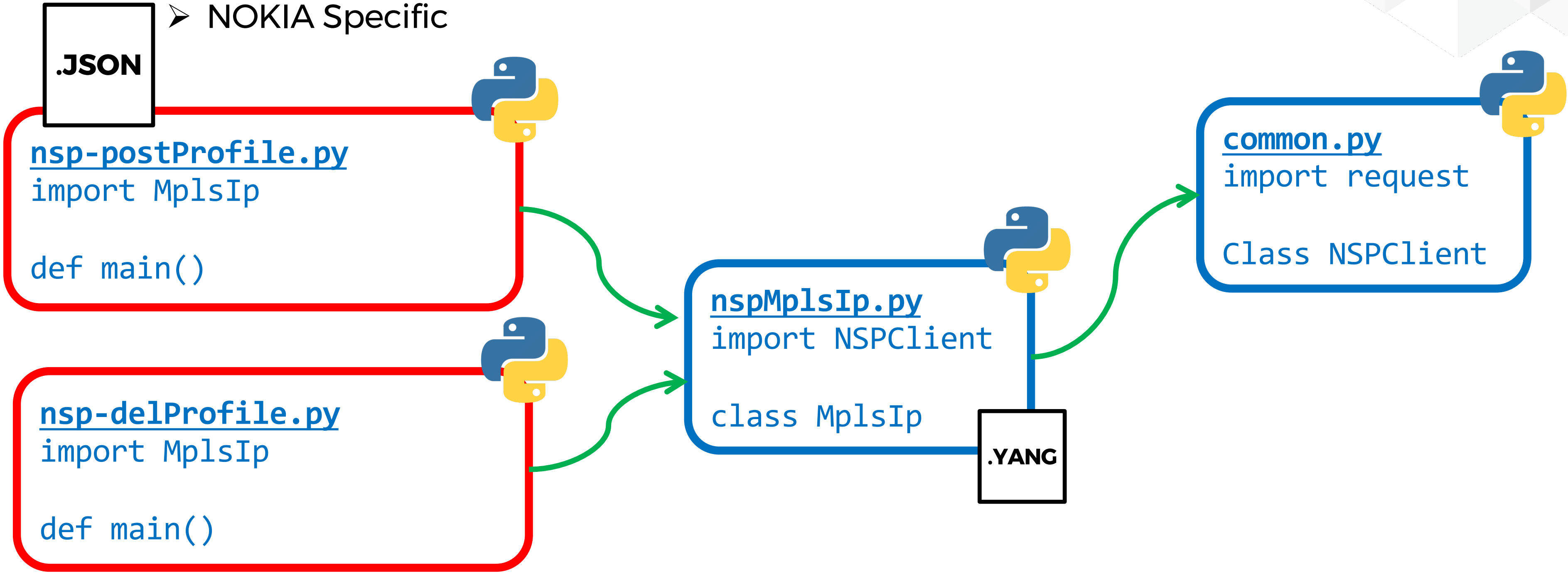
3rd Party Libs

Requirements.txt

```
lxml==4.9.3
pandas==2.1.3
matplotlib==3.8.2
Flask==3.0.0
markdown2==2.4.11
```

Code Description: nsp-postProfile.py

- IETF RSVP-TE Compatible
- NOKIA Specific



.JSON

```
nsp-postProfile.py
import MplsIp

def main()
```

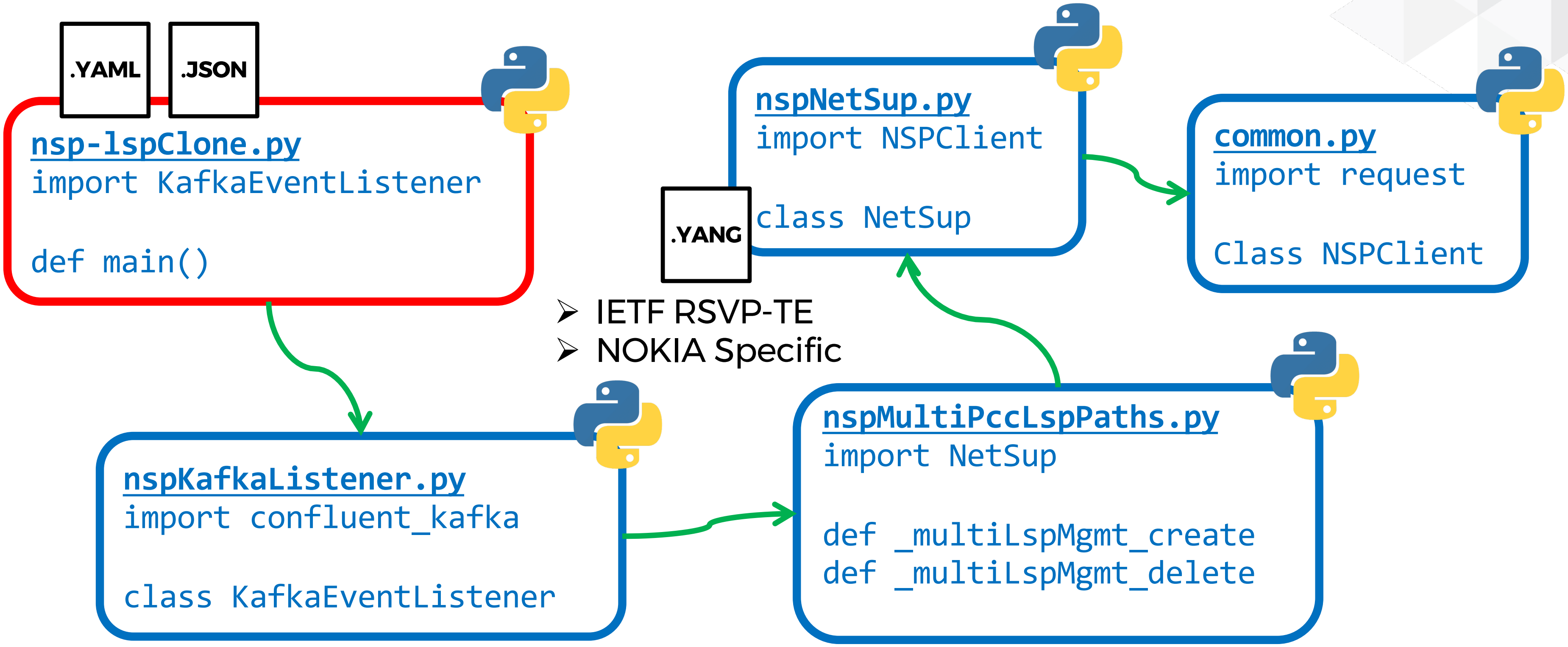
```
nsp-delProfile.py
import MplsIp

def main()
```

```
nspMplsIp.py
import NSPClient

class MplsIp
```

.YANG

```
common.py
import request

Class NSPClient
```

Code Description: nsp-lspClone.py



nsp-lspClone.py
import KafkaEventListener

def main()

nspKafkaListener.py
import confluent_kafka

class KafkaEventListener

nspNetSup.py
import NSPClient

class NetSup

- IETF RSVP-TE
- NOKIA Specific

common.py
import request

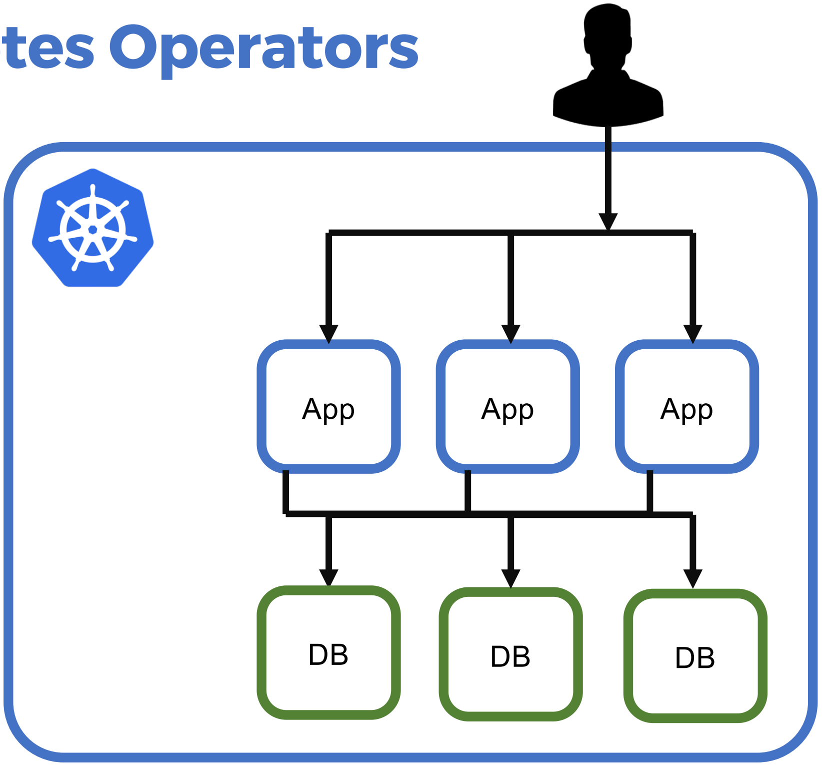
Class NSPClient

nspMultiPccLspPaths.py
import NetSup

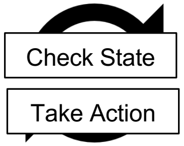
def _multiLspMgmt_create
def _multiLspMgmt_delete

Why do we use a YAML file as input?

Kubernetes Operators



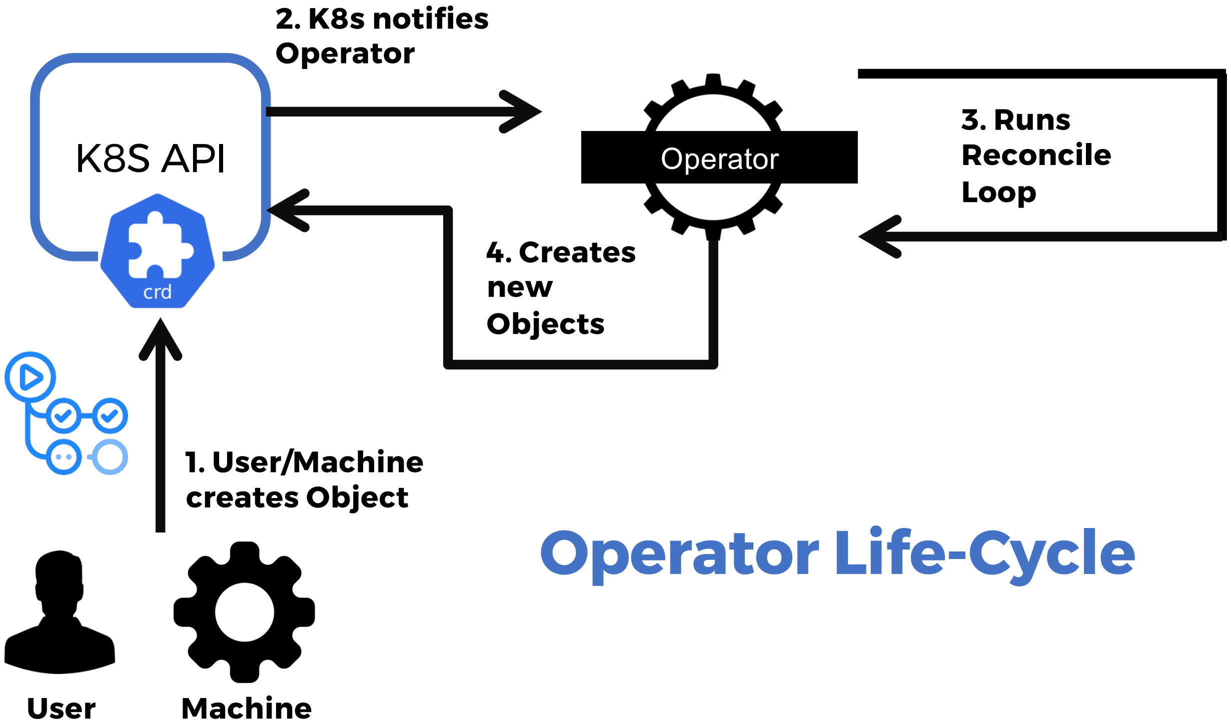
Extends K8s API



Custom Closed Loop

- Specific to every App
- Deploy Apps (Cluster/Replicas)
- Replica Recovery
- Scalable

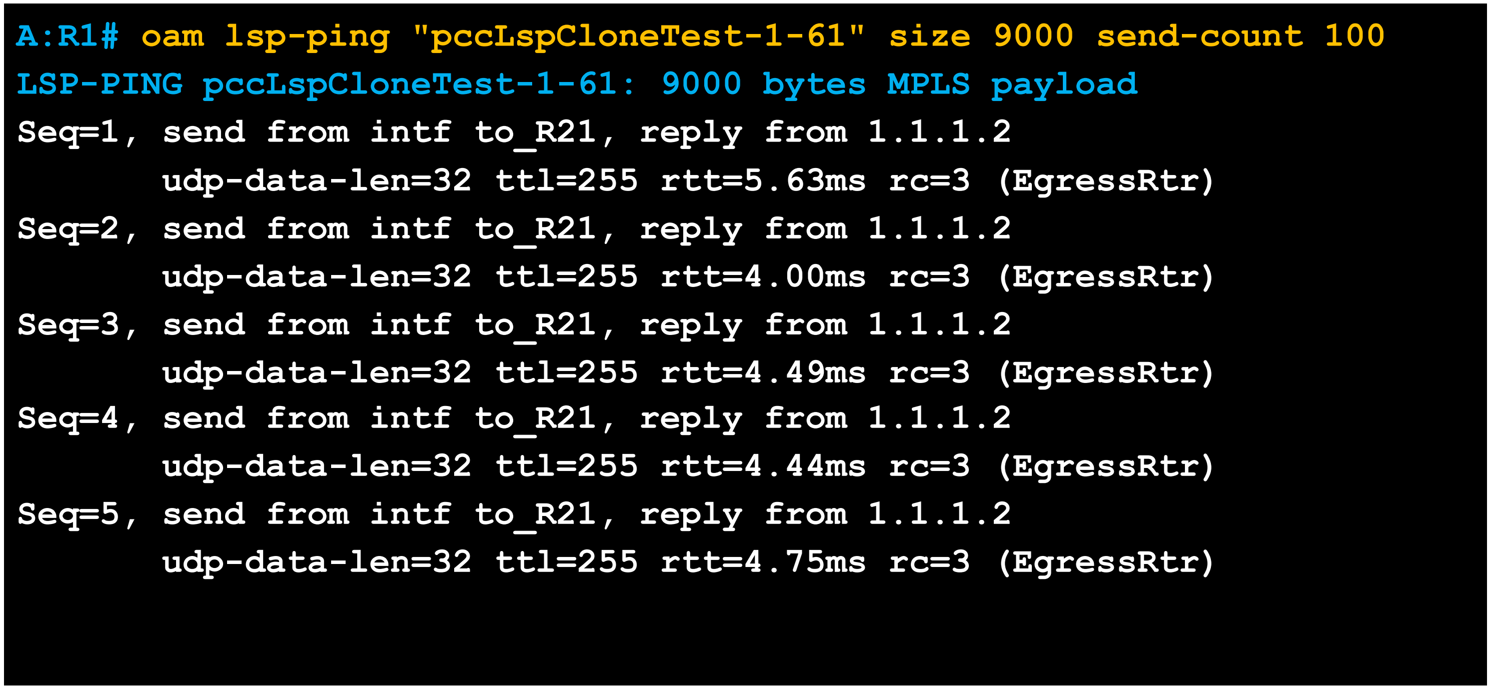
Why do we use a YAML file as input?



Operator Life-Cycle

NANOG™

How did we test it?



```
A:R1# oam lsp-ping "pccLspCloneTest-1-61" size 9000 send-count 100
LSP-PING pccLspCloneTest-1-61: 9000 bytes MPLS payload
Seq=1, send from intf to_R21, reply from 1.1.1.2
       udp-data-len=32 ttl=255 rtt=5.63ms rc=3 (EgressRtr)
Seq=2, send from intf to_R21, reply from 1.1.1.2
       udp-data-len=32 ttl=255 rtt=4.00ms rc=3 (EgressRtr)
Seq=3, send from intf to_R21, reply from 1.1.1.2
       udp-data-len=32 ttl=255 rtt=4.49ms rc=3 (EgressRtr)
Seq=4, send from intf to_R21, reply from 1.1.1.2
       udp-data-len=32 ttl=255 rtt=4.44ms rc=3 (EgressRtr)
Seq=5, send from intf to_R21, reply from 1.1.1.2
       udp-data-len=32 ttl=255 rtt=4.75ms rc=3 (EgressRtr)
```

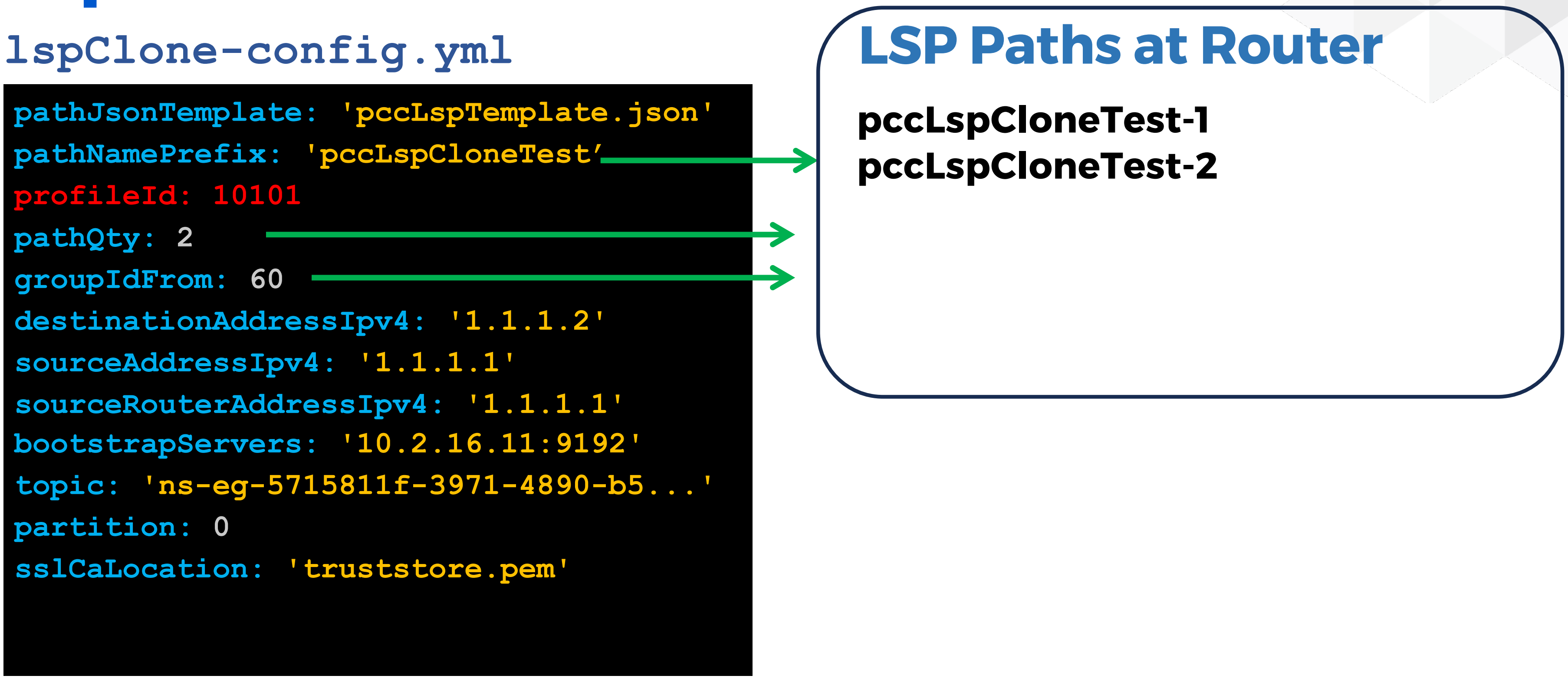
Demo Time: IETF-TE Compatible

Mau Rojas

bio.site/pinrojas

Input File

lspClone-config.yml



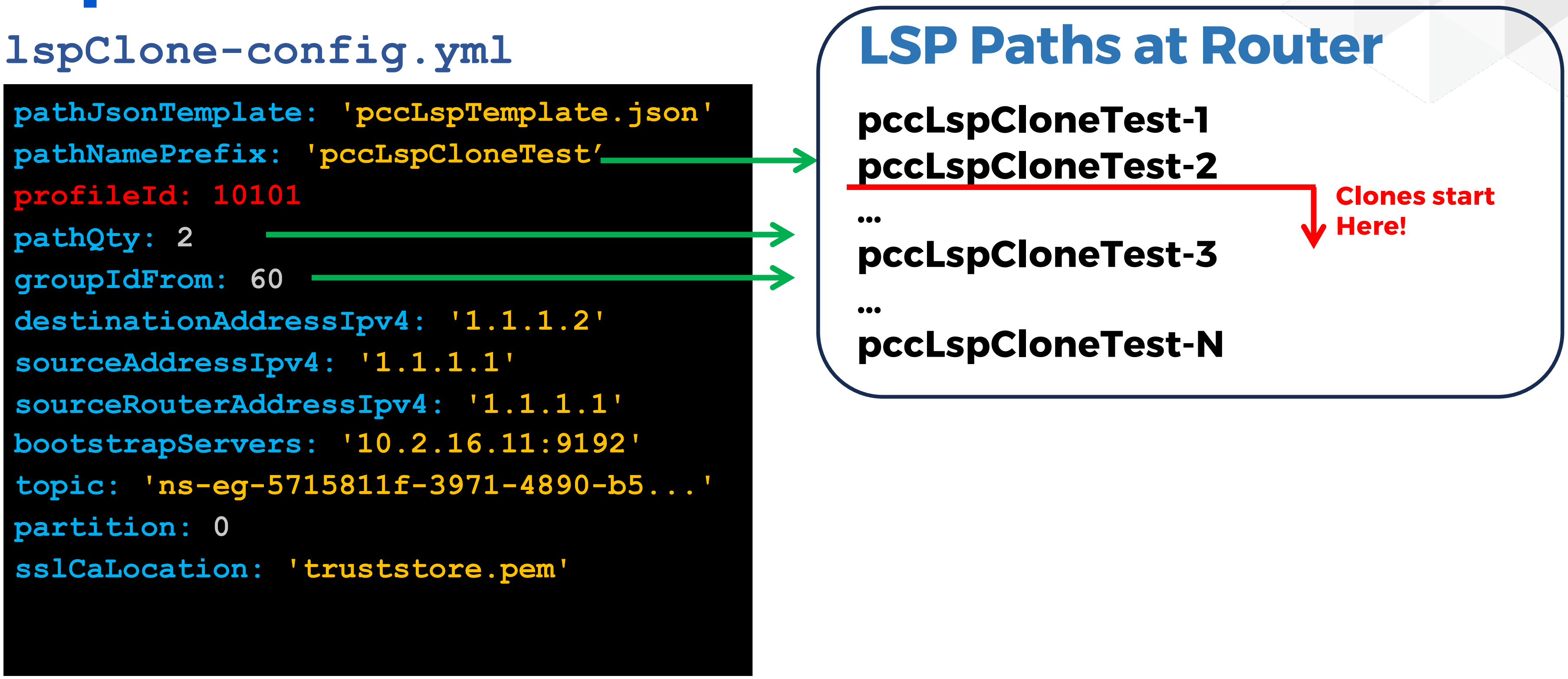
```
pathJsonTemplate: 'pccLspTemplate.json'
pathNamePrefix: 'pccLspCloneTest’
profileId: 10101
pathQty: 2
groupIdFrom: 60
destinationAddressIpv4: '1.1.1.2'
sourceAddressIpv4: '1.1.1.1'
sourceRouterAddressIpv4: '1.1.1.1'
bootstrapServers: '10.2.16.11:9192'
topic: 'ns-eg-5715811f-3971-4890-b5...'
partition: 0
sslCaLocation: 'truststore.pem'
```

LSP Paths at Router

pccLspCloneTest-1
pccLspCloneTest-2

Input File

lspClone-config.yml



```
pathJsonTemplate: 'pccLspTemplate.json'
pathNamePrefix: 'pccLspCloneTest'
profileId: 10101
pathQty: 2
groupIdFrom: 60
destinationAddressIpv4: '1.1.1.2'
sourceAddressIpv4: '1.1.1.1'
sourceRouterAddressIpv4: '1.1.1.1'
bootstrapServers: '10.2.16.11:9192'
topic: 'ns-eg-5715811f-3971-4890-b5...'
partition: 0
sslCaLocation: 'truststore.pem'
```

LSP Paths at Router

pccLspCloneTest-1
pccLspCloneTest-2
…
pccLspCloneTest-3
…
pccLspCloneTest-N

Input File (cont)

`lspClone-config.yml`

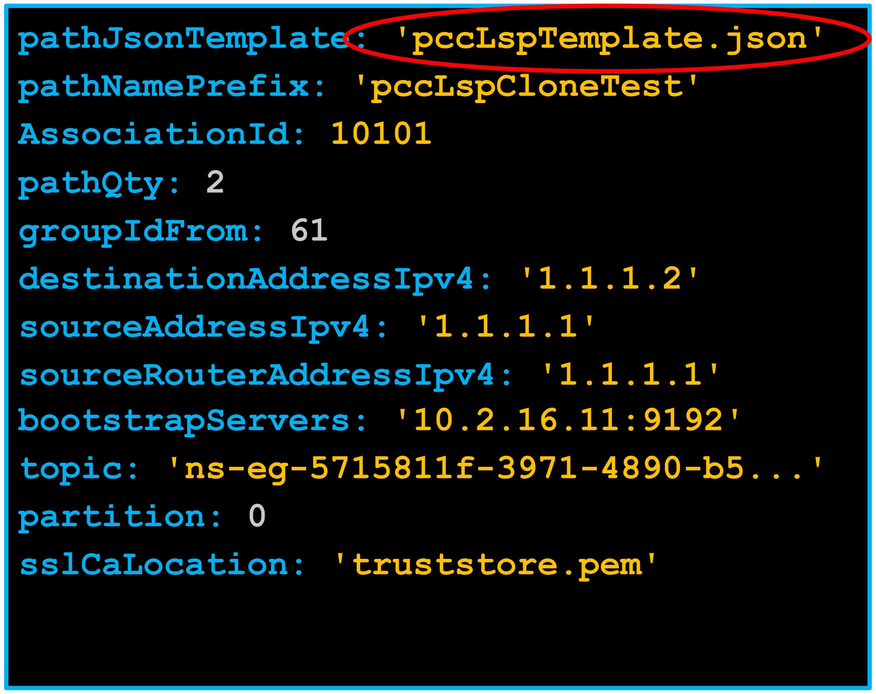
```
bootstrapServers: '10.2.16.11:9192'
topic: 'ns-eg-5715811f-3971-4890-b5...'
partition: 0
sslCaLocation: 'truststore.pem'
period: 60 #seconds
upThreshold: '90000’
upOccurrences: 2
downThreshold: '1000'
downOccurrences: 4
```

- **upThreshold** (transferred octets) triggers an occurrence when is **over** this threshold
- **downThreshold** (transferred octets) triggers an occurrence when is **under** this threshold
- For this demo, the period between occurrences is **1 min** and the number of occurrences that
  - Triggers a clone is **2 (upOccurrences)**
  - Delete a clone is **4 (downOccurrences)**

NANOG™

Input File

lspClone-config.yml



```
pathJsonTemplate: 'pccLspTemplate.json'
pathNamePrefix: 'pccLspCloneTest'
AssociationId: 10101
pathQty: 2
groupIdFrom: 61
destinationAddressIpv4: '1.1.1.2'
sourceAddressIpv4: '1.1.1.1'
sourceRouterAddressIpv4: '1.1.1.1'
bootstrapServers: '10.2.16.11:9192'
topic: 'ns-eg-5715811f-3971-4890-b5...'
partition: 0
sslCaLocation: 'truststore.pem'
```

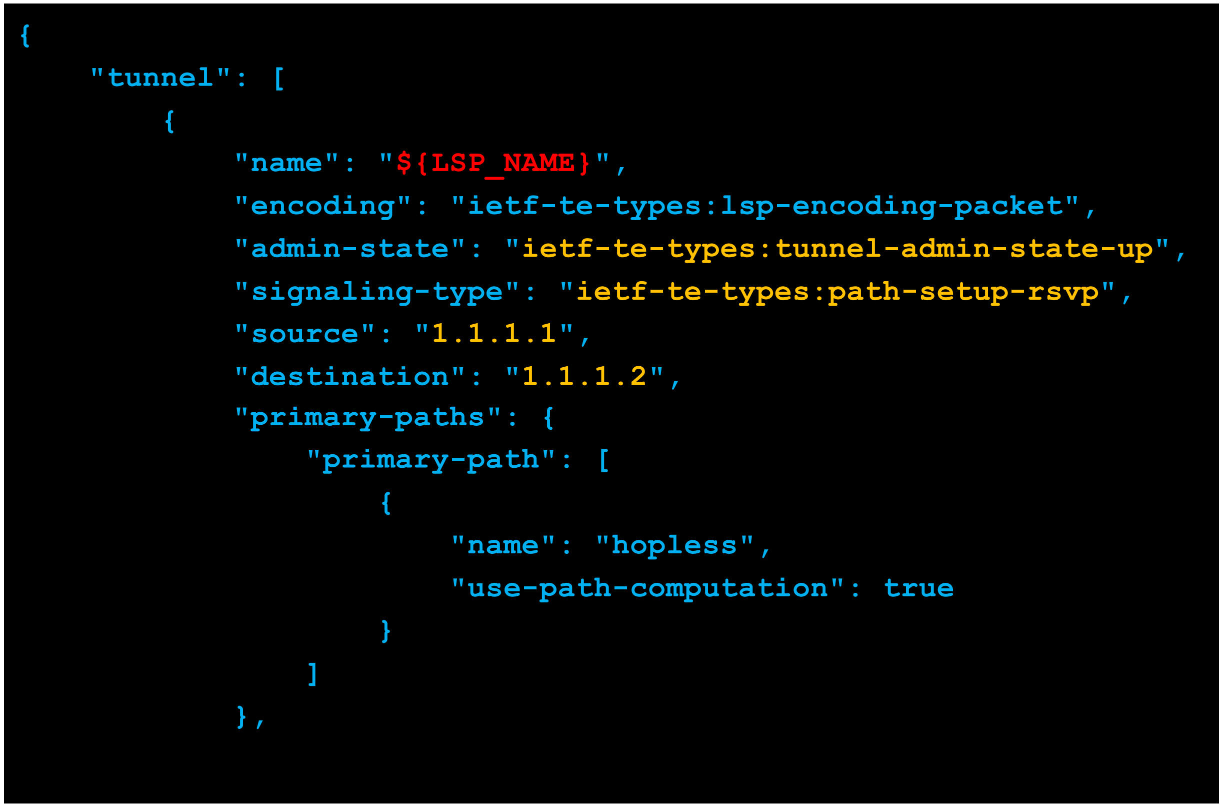
pccLspTemplate.json

IETF-TE YANG MODEL
TUNNEL TEMPLATE

Traffic Engineering Tunnels, Label Switched Paths and Interfaces
draft-ietf-teas-yang-te-35

`pccLspTemplate.json`

PCC init LSP Template



```
{
    "tunnel": [
        {
            "name": "${LSP_NAME}",
            "encoding": "ietf-te-types:lsp-encoding-packet",
            "admin-state": "ietf-te-types:tunnel-admin-state-up",
            "signaling-type": "ietf-te-types:path-setup-rsvp",
            "source": "1.1.1.1",
            "destination": "1.1.1.2",
            "primary-paths": {
                "primary-path": [
                    {
                        "name": "hopless",
                        "use-path-computation": true
                    }
                ]
            },
```

IETF-TE YANG MODEL
TUNNEL TEMPLATE

Traffic Engineering Tunnels, Label Switched Paths and Interfaces
draft-ietf-teas-yang-te-35

signaling-type: **YANG leaf that holds the LSP setup type, such as RSVP-TE or SR**

pccLspTemplate.json

PCC init LSP Template (cont)

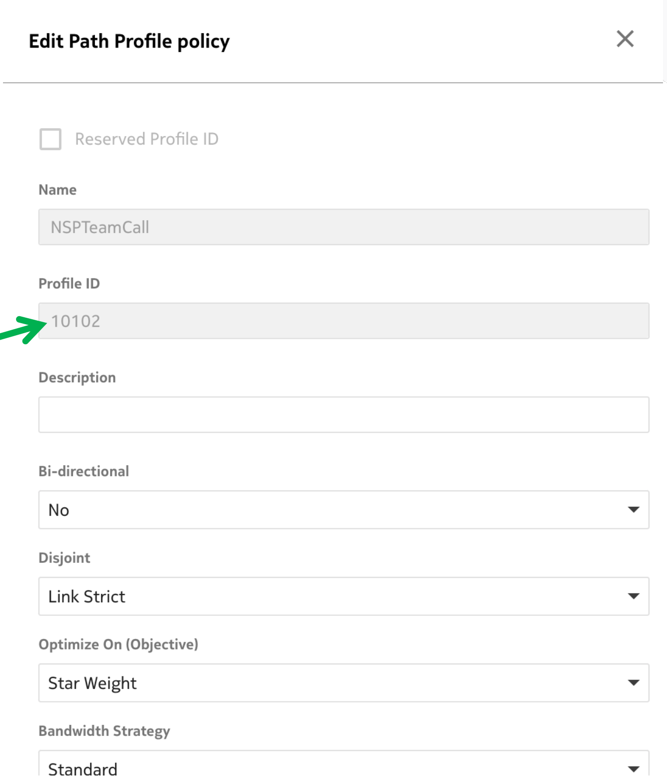
```
# coming from previous slide

"association-objects": {
"association-object-extended": [
  {
     "association-key" : "nokia-path-profile-1",
     "id": "1",
     "extended-id" : "4F”
  }
]
# additional lines have been discarded
```

```
list association-object-extended {
        key "association-key";
        unique
          "type id source/id source/type global-source
extended-id";
        description
          "List of extended association objects.";
        reference
          "RFC6780";

        leaf id {
                type uint16;
                description
                  "Association identifier.";
                reference
                  "RFC4872, RFC6780";

        leaf extended-id {
                type yang:hex-string;
                description
                  "Association extended identifier.";
                reference
                  "RFC6780";
```

PCC init LSP Template

pccLspTemplate.json

```
"tunnel": [
     {
         "name": "pccLspCloneTest-2-62",
... # this is an extract
         "association-objects": {
              "association-object-extended": [
                   {
                        "association-key" : "nokia-path-profile-1",
                        "id": "10102",
                        "extended-id" : ”3E"
                   }
              ]
```



Edit Path Profile policy

☐ Reserved Profile ID

Name: NSPTeamCall

Profile ID: 10102

Description:

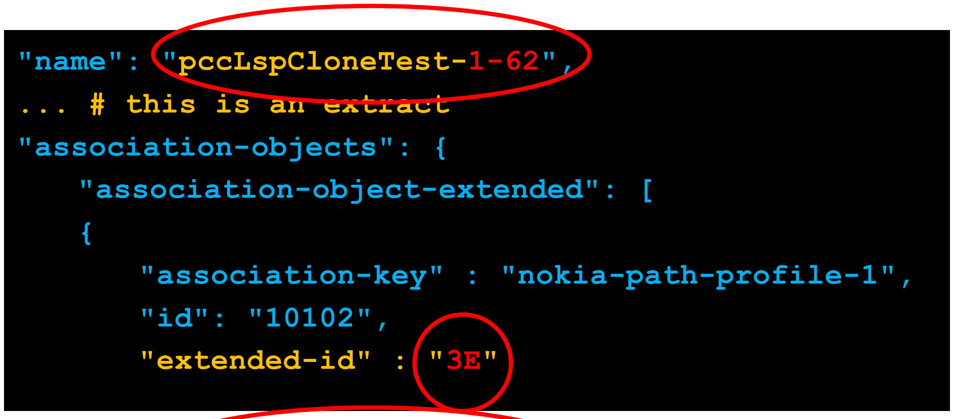
Bi-directional: No

Disjoint: Link Strict

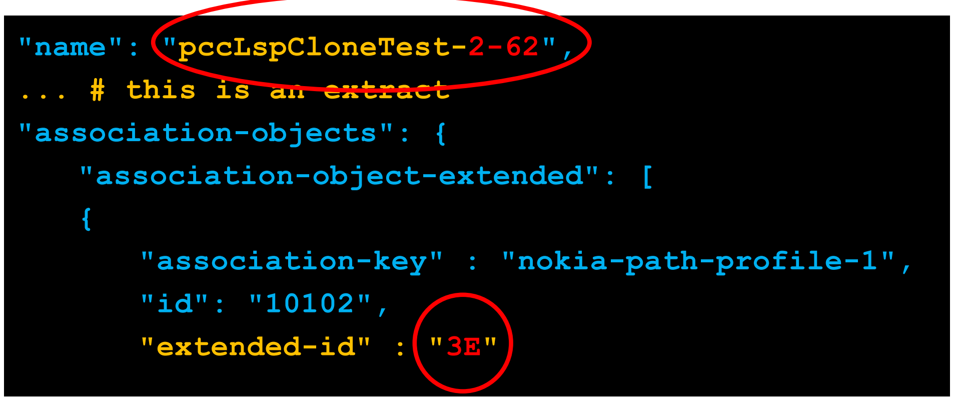
Optimize On (Objective): Star Weight

Bandwidth Strategy: Standard

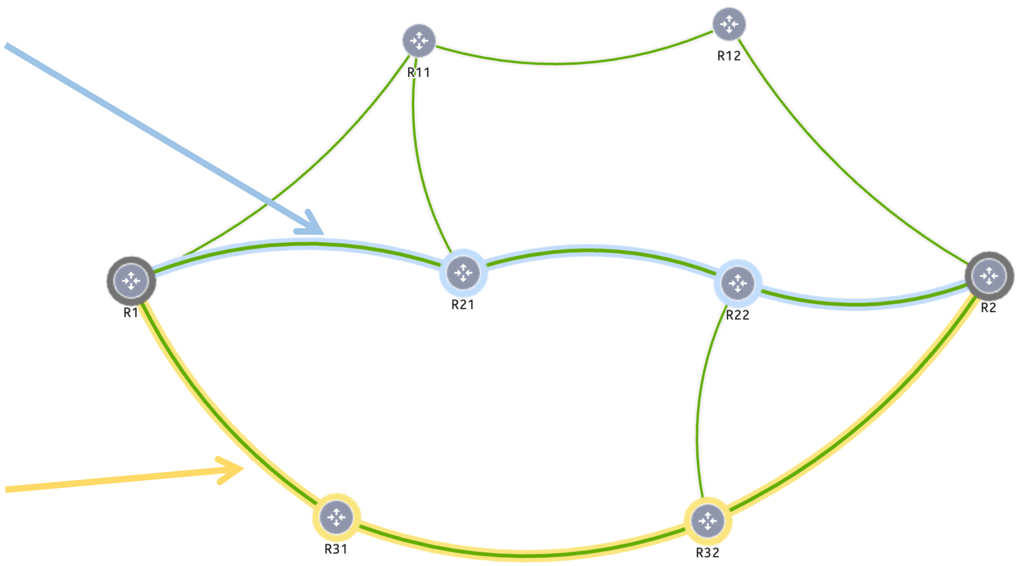
LSP Disjoint Group



```
"name": "pccLspCloneTest-1-62",
... # this is an extract
"association-objects": {
    "association-object-extended": [
    {
        "association-key" : "nokia-path-profile-1",
        "id": "10102",
        "extended-id" : "3E"
```

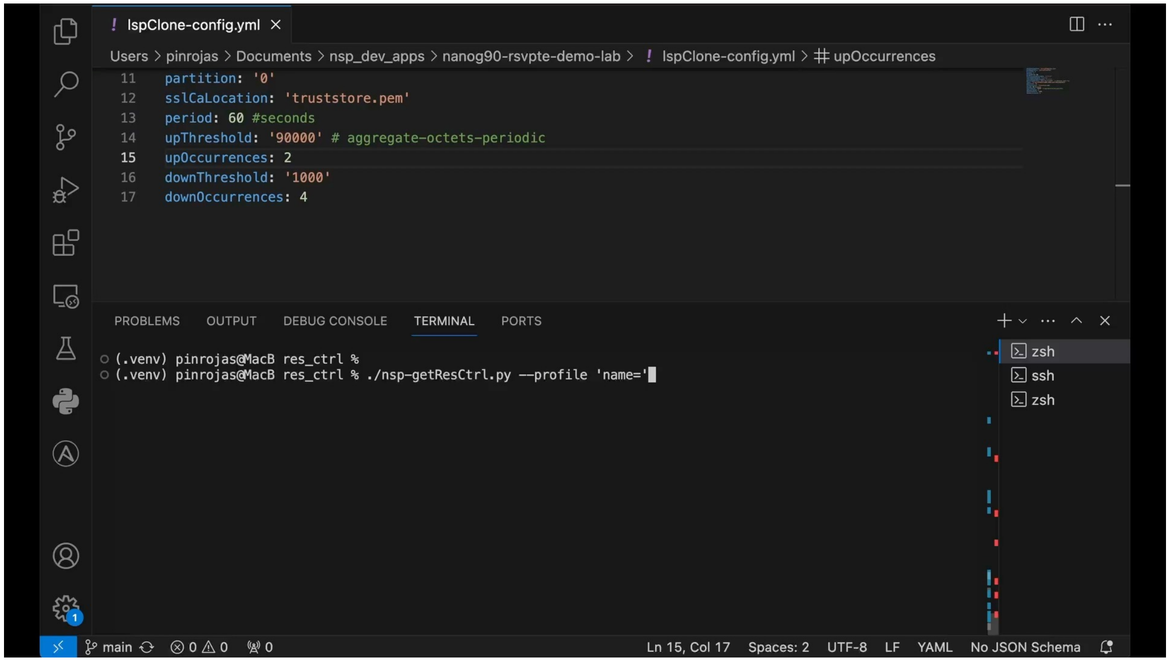


```
"name": "pccLspCloneTest-2-62",
... # this is an extract
"association-objects": {
    "association-object-extended": [
    {
        "association-key" : "nokia-path-profile-1",
        "id": "10102",
        "extended-id" : "3E"
```



Demo: Create PCE Policies

```
./nsp-postProfile.py --datafile profileTemplate.json --name nanog90 --profileId 12
```

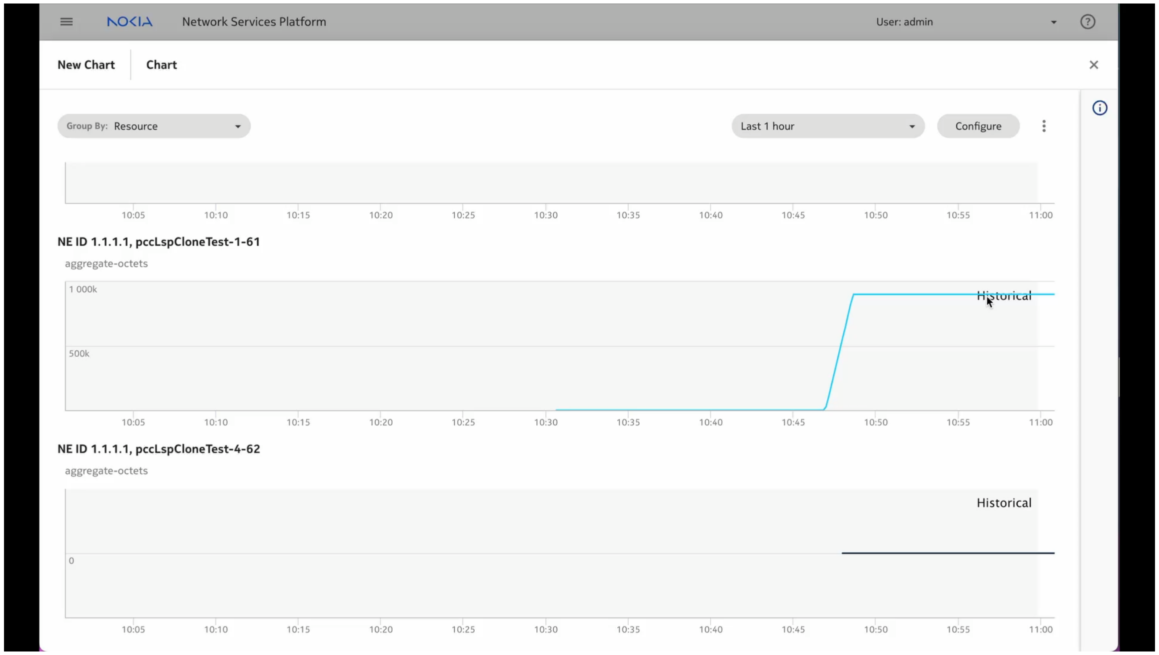


```
partition: '0'
sslCaLocation: 'truststore.pem'
period: 60 #seconds
upThreshold: '90000' # aggregate-octets-periodic
upOccurrences: 2
downThreshold: '1000'
downOccurrences: 4
```

```
(.venv) pinrojas@MacB res_ctrl %
(.venv) pinrojas@MacB res_ctrl % ./nsp-getResCtrl.py --profile 'name='
```

Demo: Start Script

```
(.venv) pinrojas@MacB nanog90-rsvpte-demo-lab % ./nsp-lspClone.py --configfile lspClone-config.yml
```



Group By: Resource

Last 1 hour

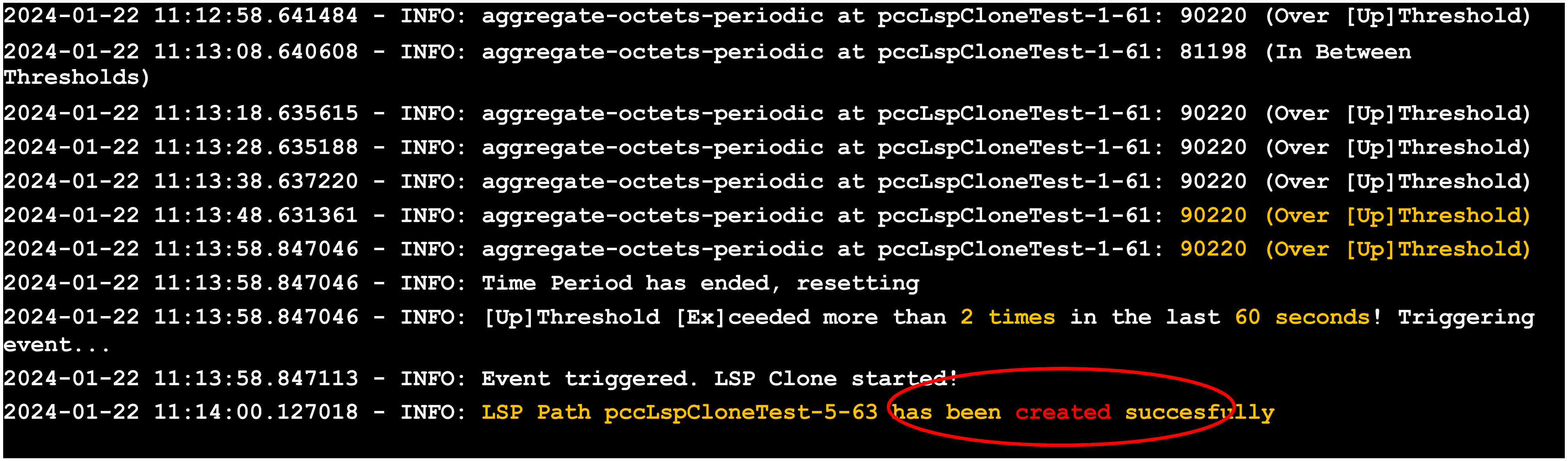
NE ID 1.1.1.1, pccLspCloneTest-1-61

aggregate-octets

NE ID 1.1.1.1, pccLspCloneTest-4-62

aggregate-octets

Demo: Going Over upThreshold

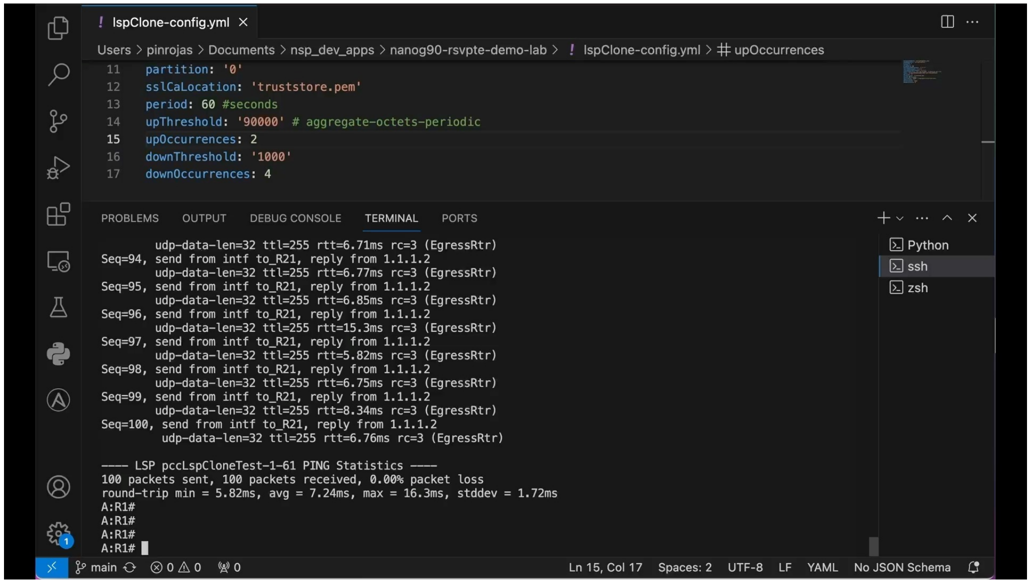


```
2024-01-22 11:12:58.641484 - INFO: aggregate-octets-periodic at pccLspCloneTest-1-61: 90220 (Over [Up]Threshold)
2024-01-22 11:13:08.640608 - INFO: aggregate-octets-periodic at pccLspCloneTest-1-61: 81198 (In Between
Thresholds)
2024-01-22 11:13:18.635615 - INFO: aggregate-octets-periodic at pccLspCloneTest-1-61: 90220 (Over [Up]Threshold)
2024-01-22 11:13:28.635188 - INFO: aggregate-octets-periodic at pccLspCloneTest-1-61: 90220 (Over [Up]Threshold)
2024-01-22 11:13:38.637220 - INFO: aggregate-octets-periodic at pccLspCloneTest-1-61: 90220 (Over [Up]Threshold)
2024-01-22 11:13:48.631361 - INFO: aggregate-octets-periodic at pccLspCloneTest-1-61: 90220 (Over [Up]Threshold)
2024-01-22 11:13:58.847046 - INFO: aggregate-octets-periodic at pccLspCloneTest-1-61: 90220 (Over [Up]Threshold)
2024-01-22 11:13:58.847046 - INFO: Time Period has ended, resetting
2024-01-22 11:13:58.847046 - INFO: [Up]Threshold [Ex]ceeded more than 2 times in the last 60 seconds! Triggering
event...
2024-01-22 11:13:58.847113 - INFO: Event triggered. LSP Clone started!
2024-01-22 11:14:00.127018 - INFO: LSP Path pccLspCloneTest-5-63 has been created succesfully
```

Demo: Going Over upThreshold

lspClone-config.yml

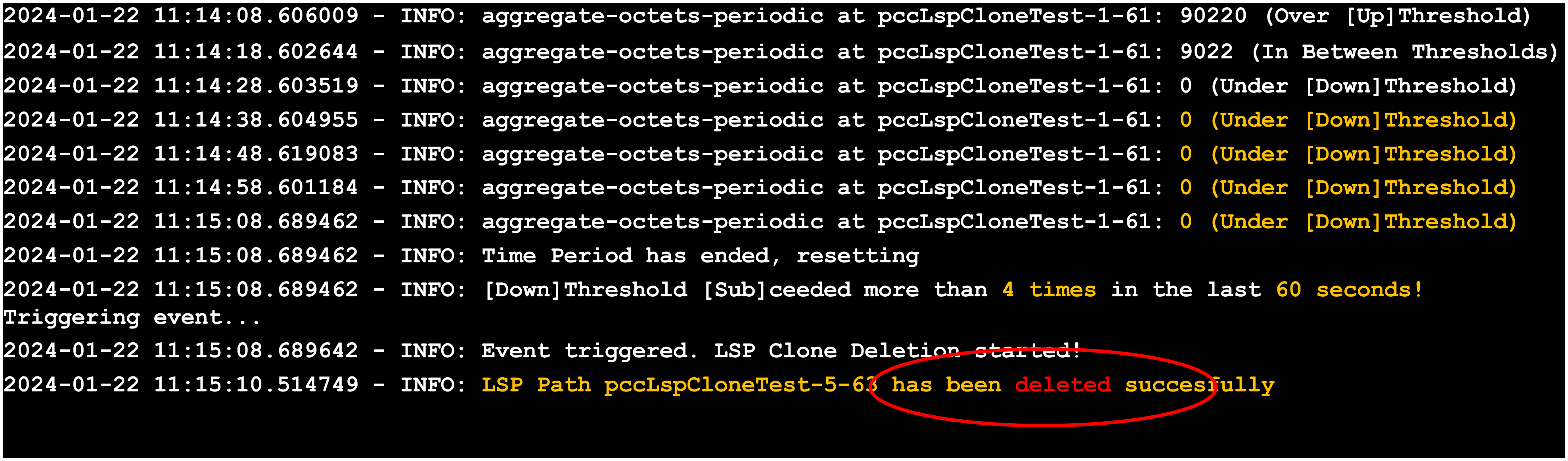
```
period:  60 #seconds
upThreshold:  '90000'
upOccurrences:  2
downThreshold:  '1000'
downOccurrences:  4
```



```
partition: '0'
sslCaLocation: 'truststore.pem'
period: 60 #seconds
upThreshold: '90000' # aggregate-octets-periodic
upOccurrences: 2
downThreshold: '1000'
downOccurrences: 4
```

```
         udp-data-len=32 ttl=255 rtt=6.71ms rc=3 (EgressRtr)
Seq=94, send from intf to_R21, reply from 1.1.1.2
         udp-data-len=32 ttl=255 rtt=6.77ms rc=3 (EgressRtr)
Seq=95, send from intf to_R21, reply from 1.1.1.2
         udp-data-len=32 ttl=255 rtt=6.85ms rc=3 (EgressRtr)
Seq=96, send from intf to_R21, reply from 1.1.1.2
         udp-data-len=32 ttl=255 rtt=15.3ms rc=3 (EgressRtr)
Seq=97, send from intf to_R21, reply from 1.1.1.2
         udp-data-len=32 ttl=255 rtt=5.82ms rc=3 (EgressRtr)
Seq=98, send from intf to_R21, reply from 1.1.1.2
         udp-data-len=32 ttl=255 rtt=6.75ms rc=3 (EgressRtr)
Seq=99, send from intf to_R21, reply from 1.1.1.2
         udp-data-len=32 ttl=255 rtt=8.34ms rc=3 (EgressRtr)
Seq=100, send from intf to_R21, reply from 1.1.1.2
          udp-data-len=32 ttl=255 rtt=6.76ms rc=3 (EgressRtr)

---- LSP pccLspCloneTest-1-61 PING Statistics ----
100 packets sent, 100 packets received, 0.00% packet loss
round-trip min = 5.82ms, avg = 7.24ms, max = 16.3ms, stddev = 1.72ms
A:R1#
A:R1#
A:R1#
A:R1#
```

Demo: Going Under downThreshold

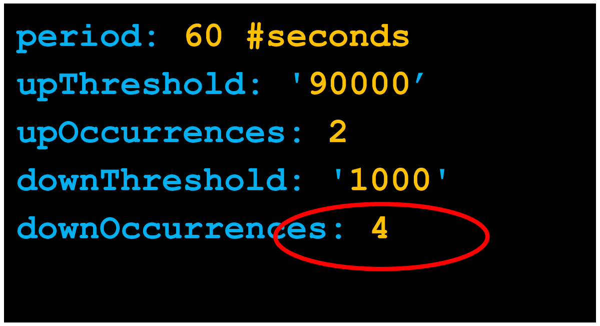


```
2024-01-22 11:14:08.606009 - INFO: aggregate-octets-periodic at pccLspCloneTest-1-61: 90220 (Over [Up]Threshold)
2024-01-22 11:14:18.602644 - INFO: aggregate-octets-periodic at pccLspCloneTest-1-61: 9022 (In Between Thresholds)
2024-01-22 11:14:28.603519 - INFO: aggregate-octets-periodic at pccLspCloneTest-1-61: 0 (Under [Down]Threshold)
2024-01-22 11:14:38.604955 - INFO: aggregate-octets-periodic at pccLspCloneTest-1-61: 0 (Under [Down]Threshold)
2024-01-22 11:14:48.619083 - INFO: aggregate-octets-periodic at pccLspCloneTest-1-61: 0 (Under [Down]Threshold)
2024-01-22 11:14:58.601184 - INFO: aggregate-octets-periodic at pccLspCloneTest-1-61: 0 (Under [Down]Threshold)
2024-01-22 11:15:08.689462 - INFO: aggregate-octets-periodic at pccLspCloneTest-1-61: 0 (Under [Down]Threshold)
2024-01-22 11:15:08.689462 - INFO: Time Period has ended, resetting
2024-01-22 11:15:08.689462 - INFO: [Down]Threshold [Sub]ceeded more than 4 times in the last 60 seconds!
Triggering event...
2024-01-22 11:15:08.689642 - INFO: Event triggered. LSP Clone Deletion started!
2024-01-22 11:15:10.514749 - INFO: LSP Path pccLspCloneTest-5-63 has been deleted succesfully
```

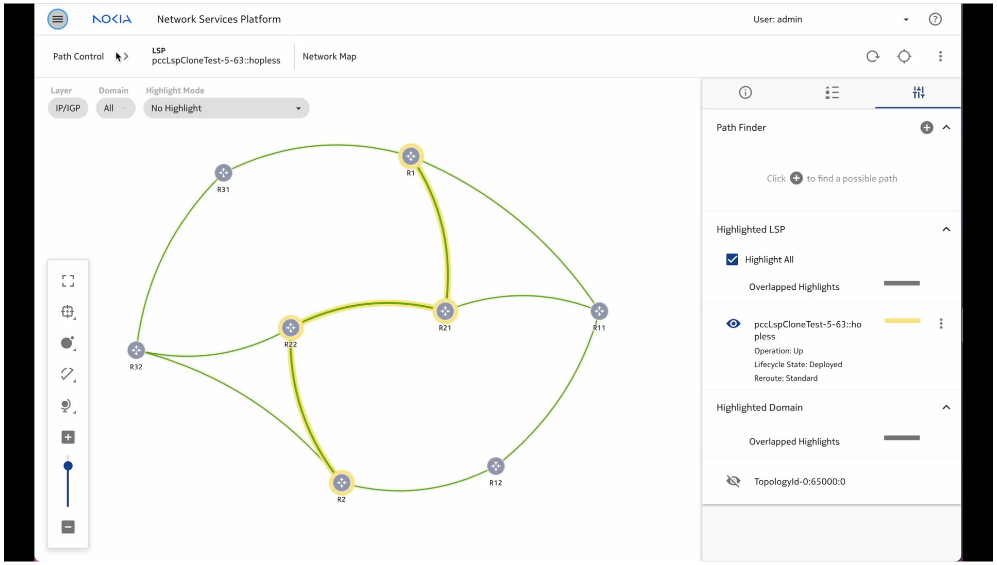
NANOG™

Demo: Going Under downThreshold

lspClone-config.yml



```
period:  60 #seconds
upThreshold:  '90000’
upOccurrences:  2
downThreshold:  '1000'
downOccurrences:  4
```



Operation: Up
Lifecycle State: Deployed
Reroute: Standard

NANOG™

Almost done!

Final Words

In this tutorial we have seen:

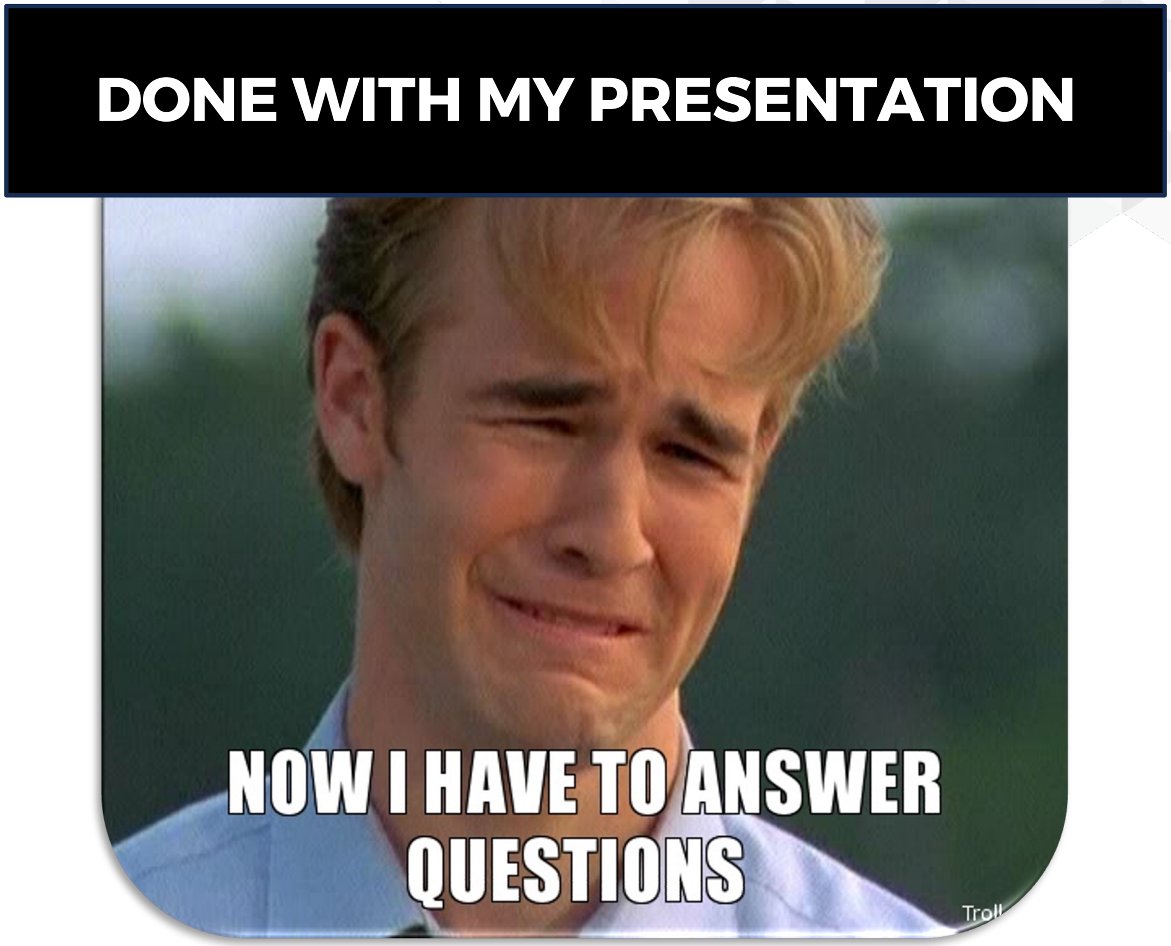
- Traffic-Engineering Automation Overview [Diego]
  - Advantages Over Non-IETF RSVP Extended Protocols
- Telemetry in TE Automation [David/Mau]
  - Integrating PCE with Telemetry (gRPC) for a Closed Loop System
- Hands-On Lab: Automation Driven Traffic Steering [Mau]
  - Core Lab Components
  - Tool Comparisons (e.g., Nornir vs SDN)
  - Building Python Application Components
  - Utilizing IETF-TE YANG RESTCONF for Tunnel Management

Additional resources

- Containerlab – NANOG talk - Running networking labs with Docker UX – Roman and Karim
  - https://youtu.be/qigCla1qY3k
- gNMIc – NANOG Talk - An intuitive gNMI CLI and a feature-rich telemetry collector - Karim
  - https://youtu.be/v3CL2vrGD_8
- Packet Pushers: PCE and PCEP Overview
  - https://packetpushers.net/blog/pce-pcep-overview/
- Check repo at: https://github.com/cloud-native-everything/nanog90-rsvpte-demo-lab (work in progress)

NANOG™

Thanks!


DONE WITH MY PRESENTATION
NOW I HAVE TO ANSWER QUESTIONS
Troll