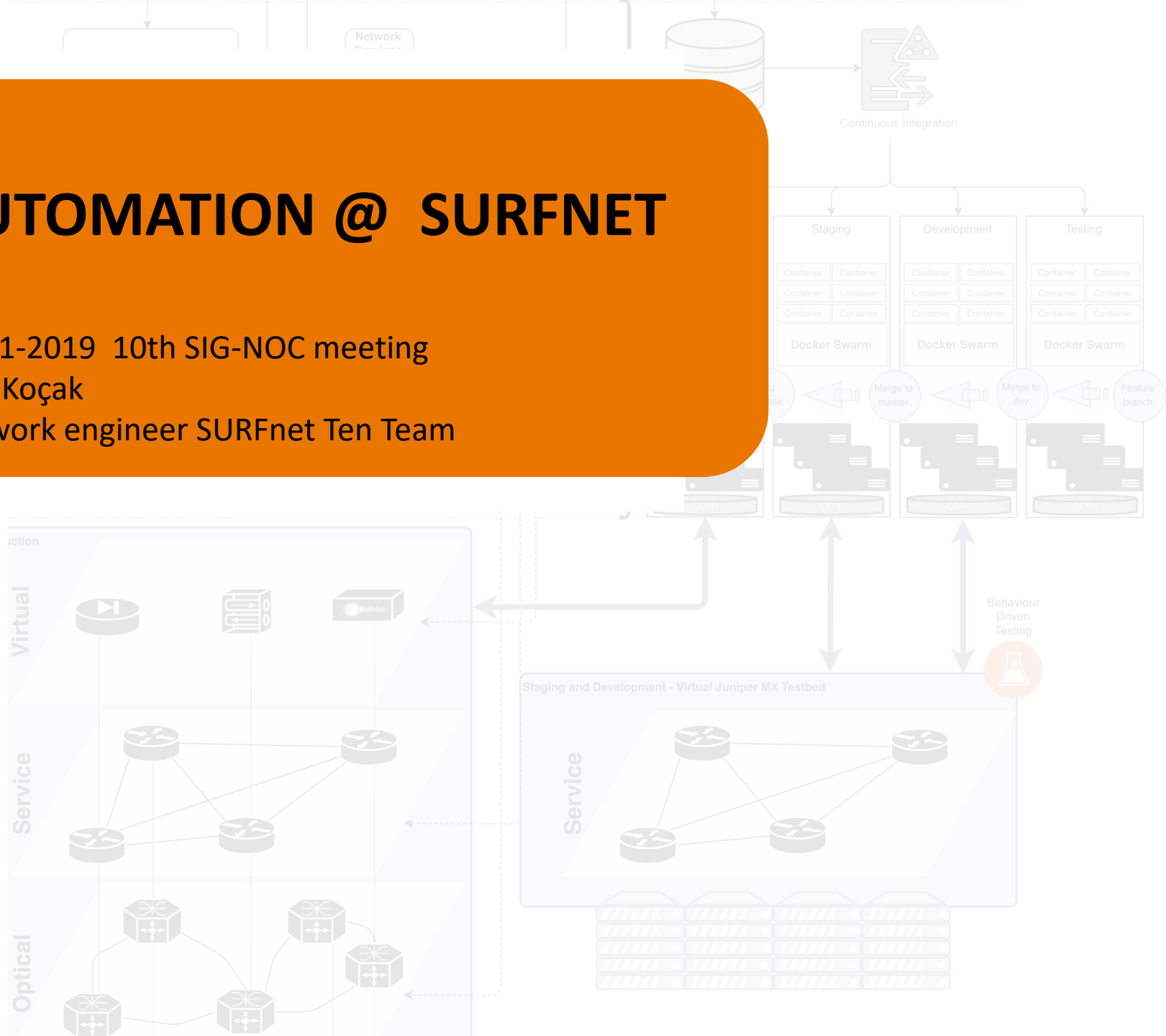
# AUTOMATION @ SURFNET

14-11-2019 10th SIG-NOC meeting

Sadi Koçak
Network engineer SURFnet Ten Team



# We are SURF

# Serving 180 institutions & ~ 1.5 million end-users

SURF

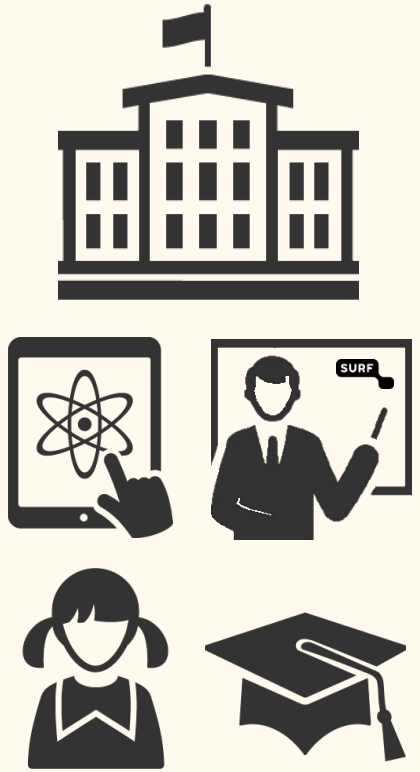
SURF SARA

SURF MARKET

SURF NET

## 40+ ICT Services, Innovation, Community & Tendering


## Institutes, Students & Researchers



SURF NET

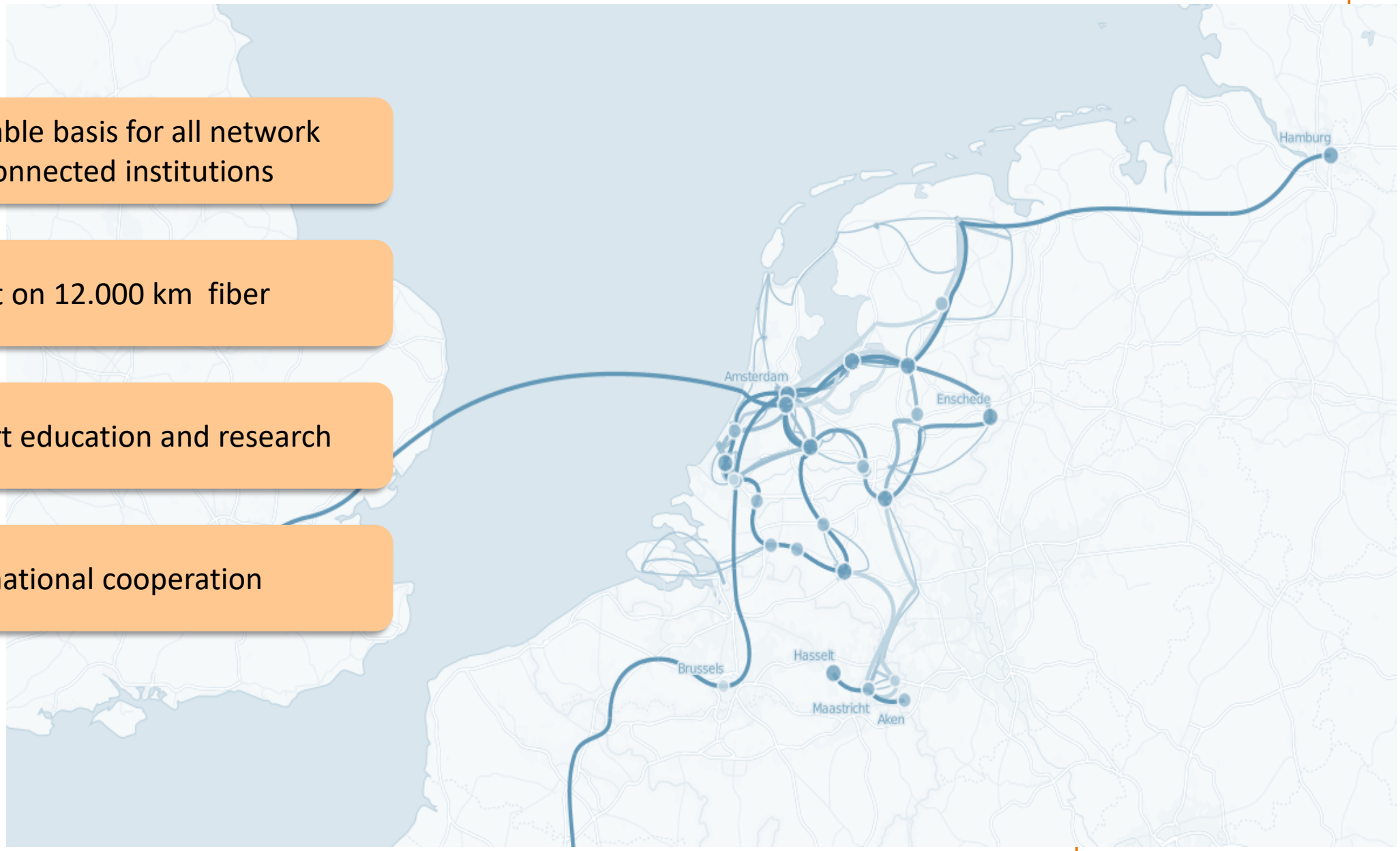
# SURFnet Network

Since 1988 a reliable basis for all network activities of connected institutions

Network built on 12.000 km  fiber

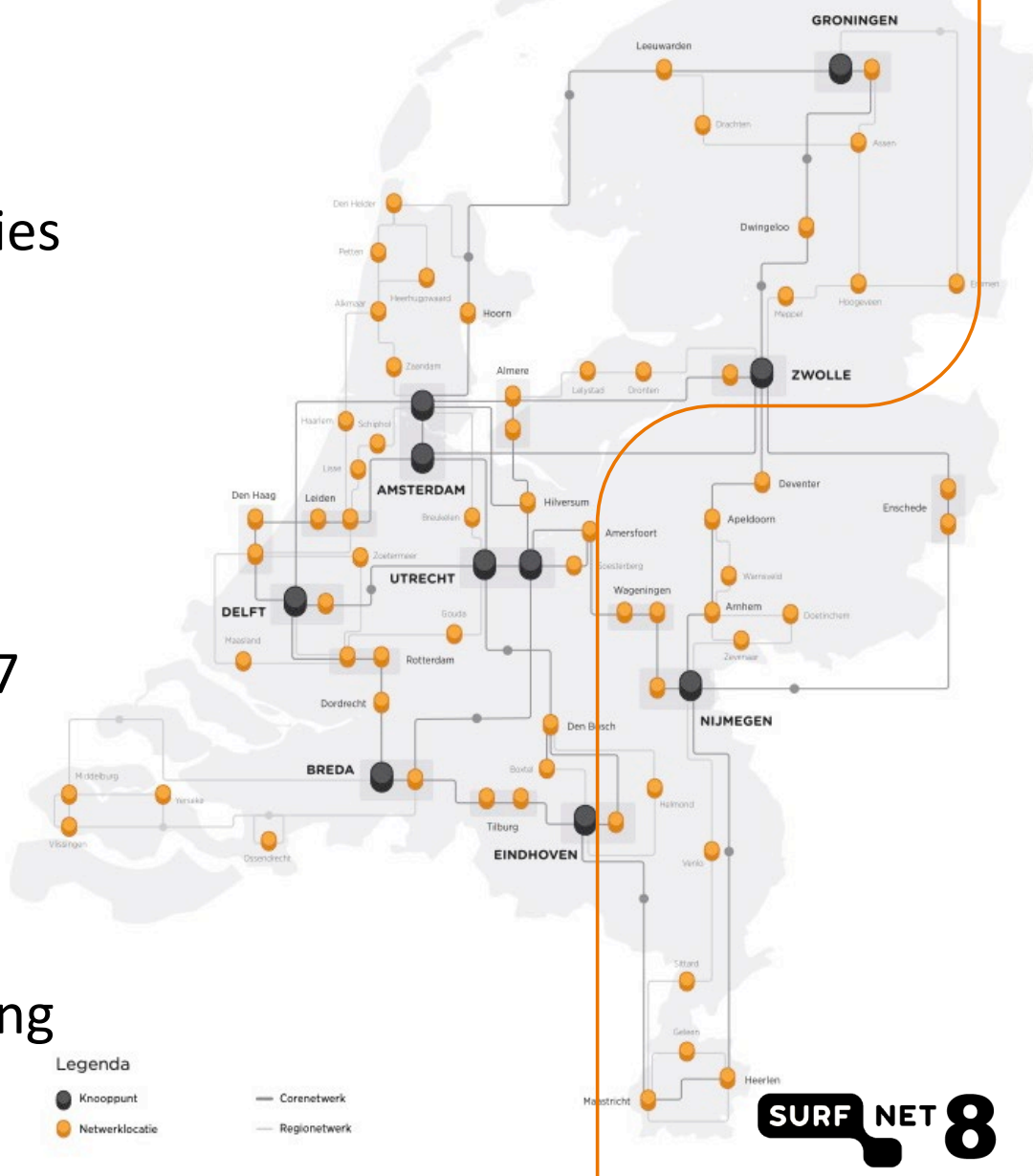
Tailored to support education and research

Strong international cooperation

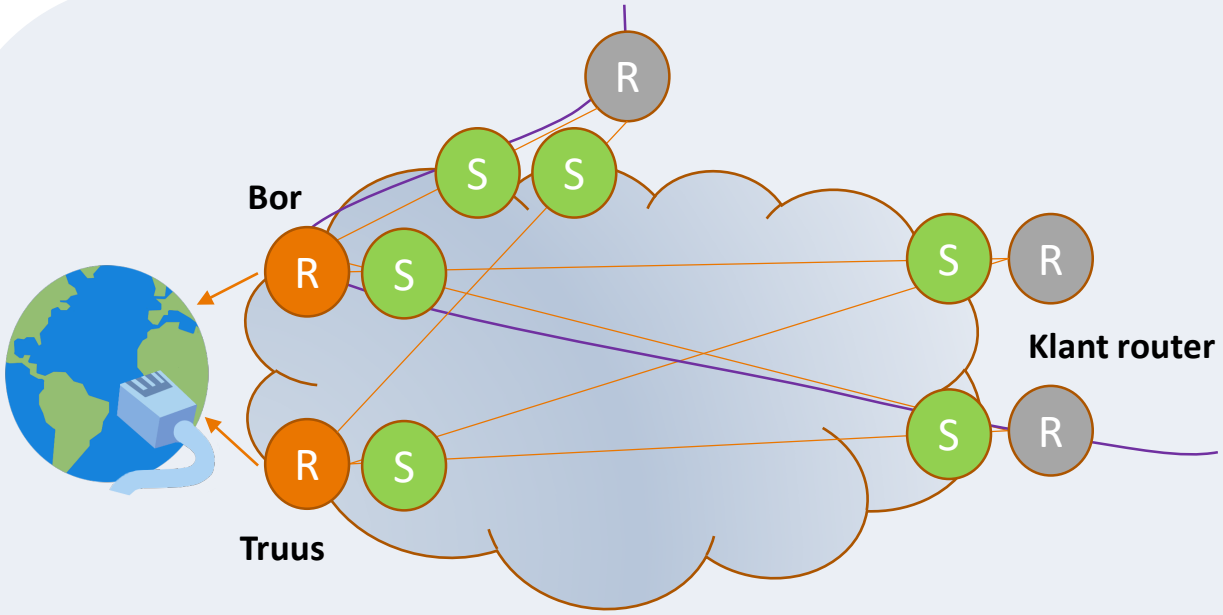


# SURFnet Project Scope

- **Replacing hardware**
  - ➢ Optical layer: Ciena CPL → ECI Apollo
  - ➢ Service layer: Ciena CN series → Juniper MX series
  - ➢ New management and monitoring environment
- **New netwerk architecture**
  - ➢ Routers everywhere
  - ➢ using new network protocols (MPLS with S.R.)
  - ➢ Hybrid network, simultanious production on SN7 en SN8 network
- **Automation**
  - ➢ Provisioning via workflows, “no more CLI”
  - ➢ Import SN7 service data → data integrity checking
  - ➢ changed way of working for SURFnet and Noc



# Project Scope – Routers Everywhere

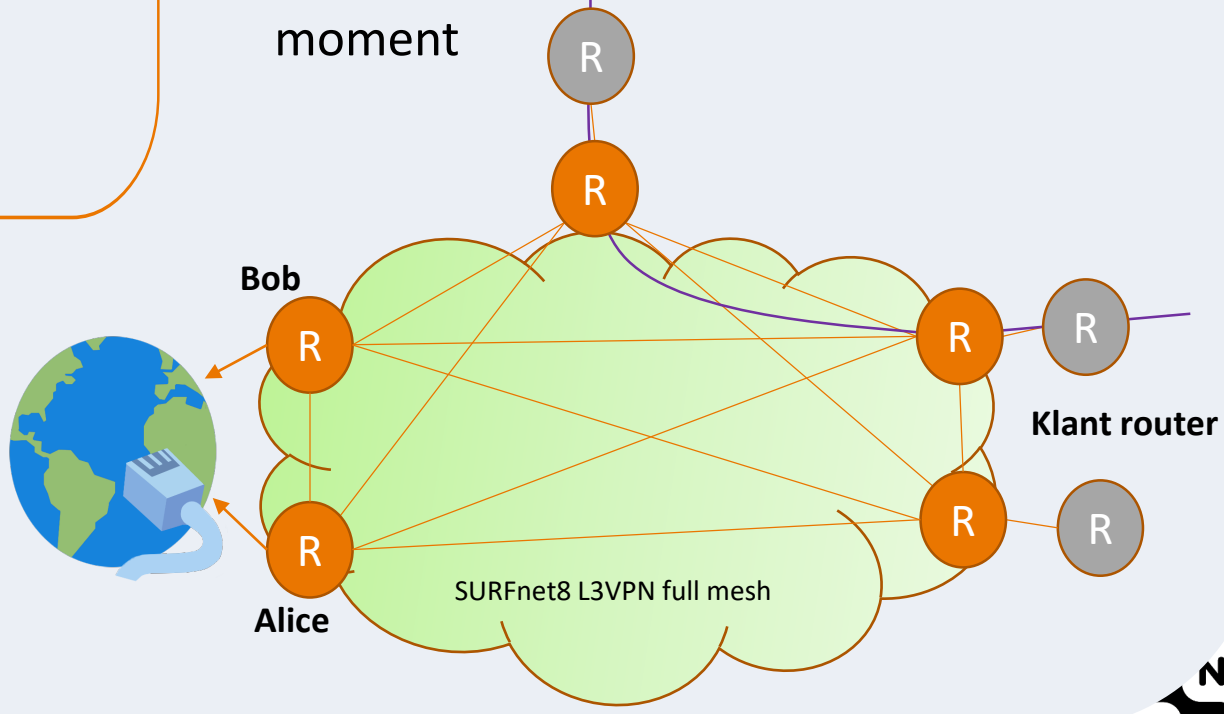



SURFNET 7

- All IP service routed via Bor en Truus using PBB-TE tunnels
- Not to dynamic use common static routes
- Due to star structure all ip traffic is flowing/routing by Bor en Truus

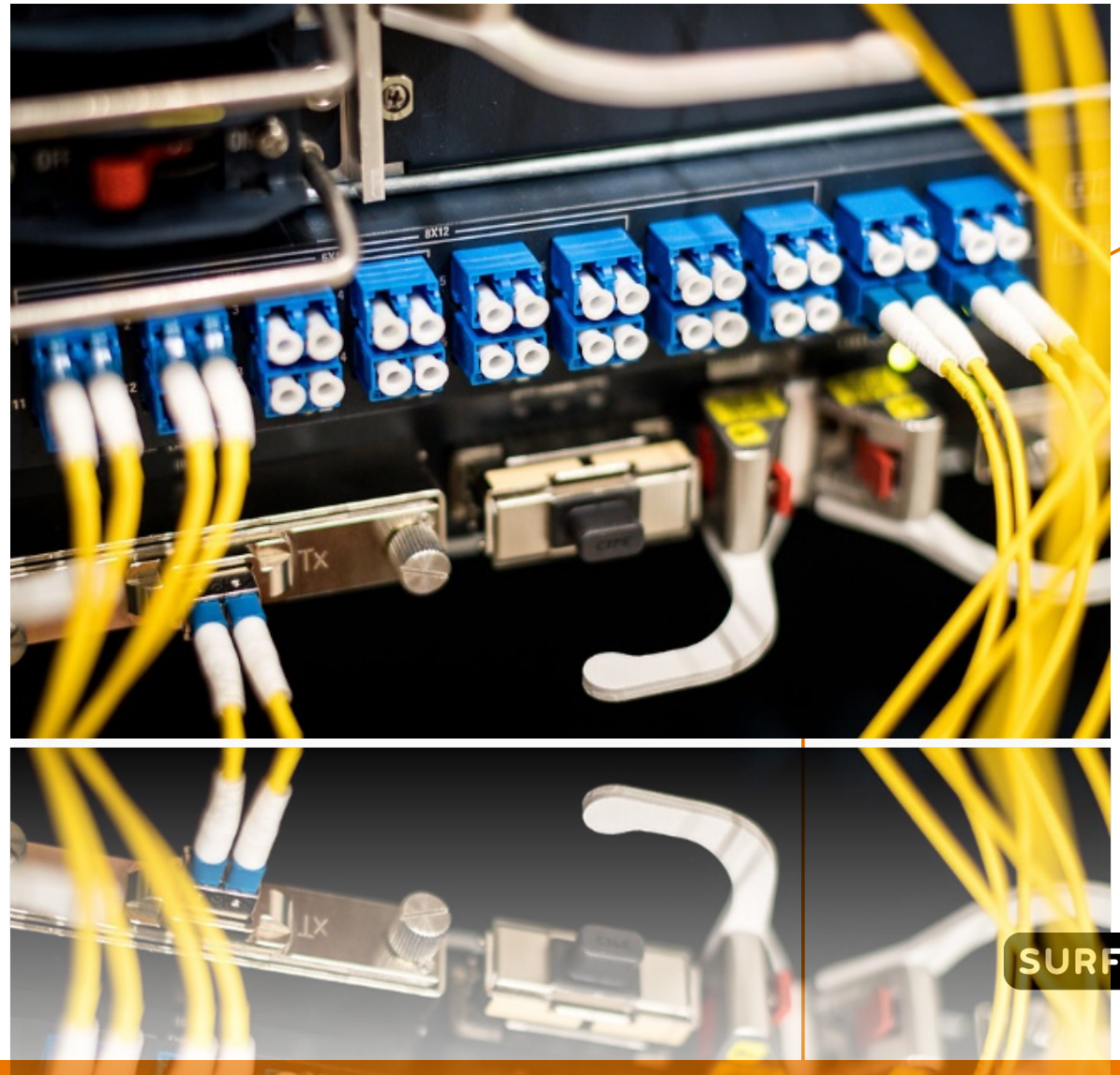
SURFNET 8

- Routers everywhere Full meshed topology
- MPLS network with Segment Routing which gives us dynamic routes, very fast rerrouting so no impact on services
- Bob en Alice are Border routers at this moment



# SURFnet Network Services

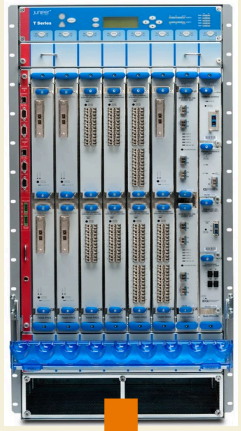
- SURFinternet
- SURFlichtpaden
- NetherLight
- SURFwireless
- Eduroam



# Project Scope – Replacing Hardware

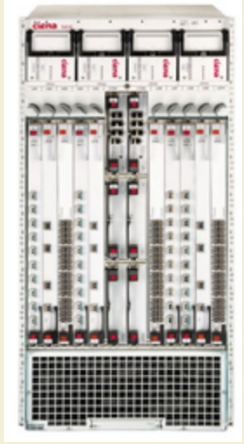
JUNIPER NETWORKS

T4000



ciena

5410



ciena

5160

ciena

CN5150

ciena

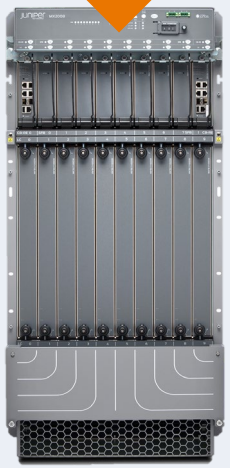
CN5142

ciena

CN3930

SURF NET 7

24U



MX2008

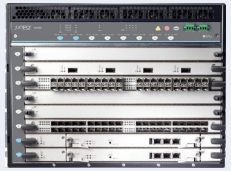
JUNIPER NETWORKS

16U

MX960

JUNIPER NETWORKS

9U



MX480

JUNIPER NETWORKS

5U

MX240

JUNIPER NETWORKS

1U

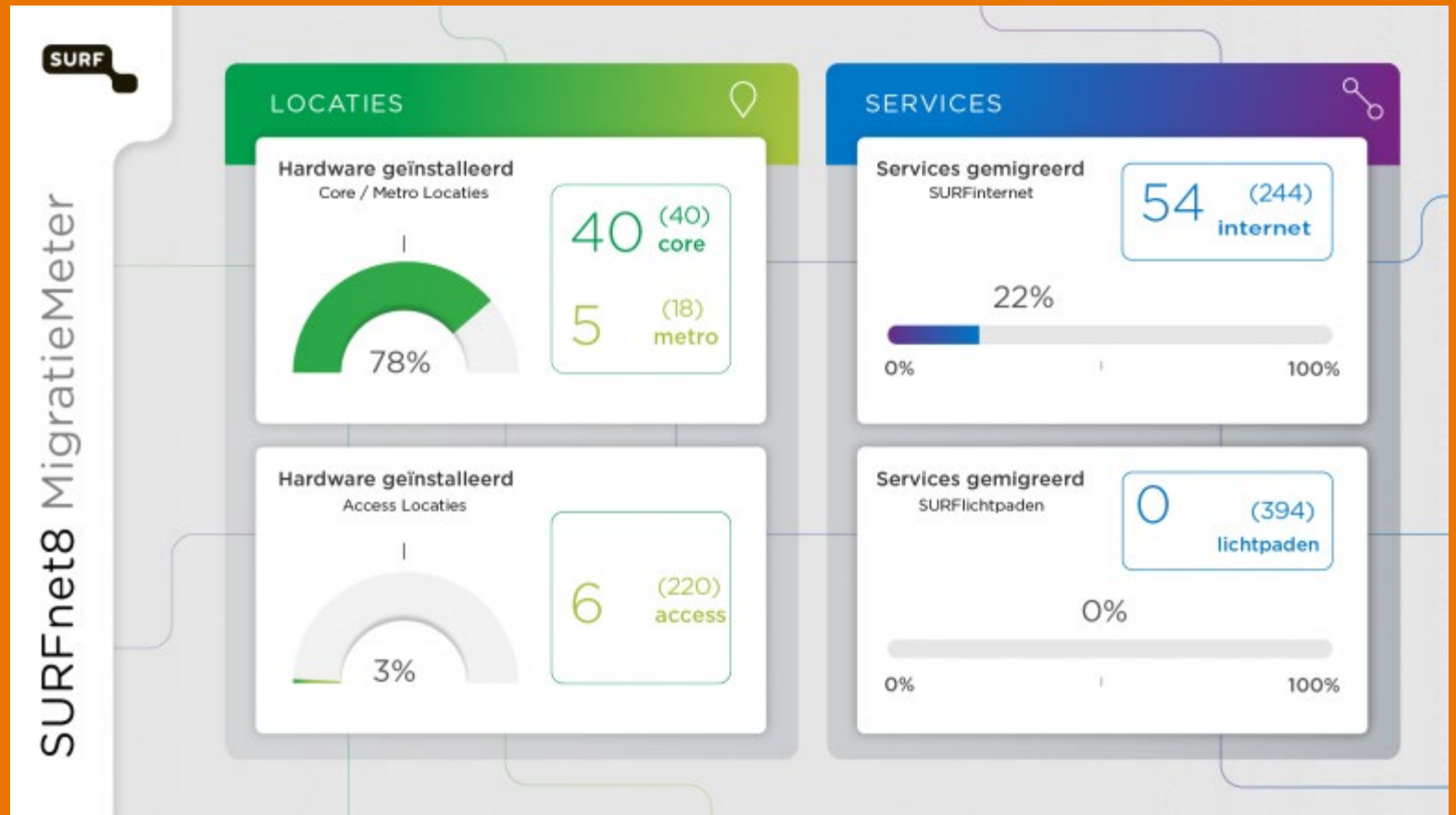
MX204

JUNIPER NETWORKS

SURF NET 8

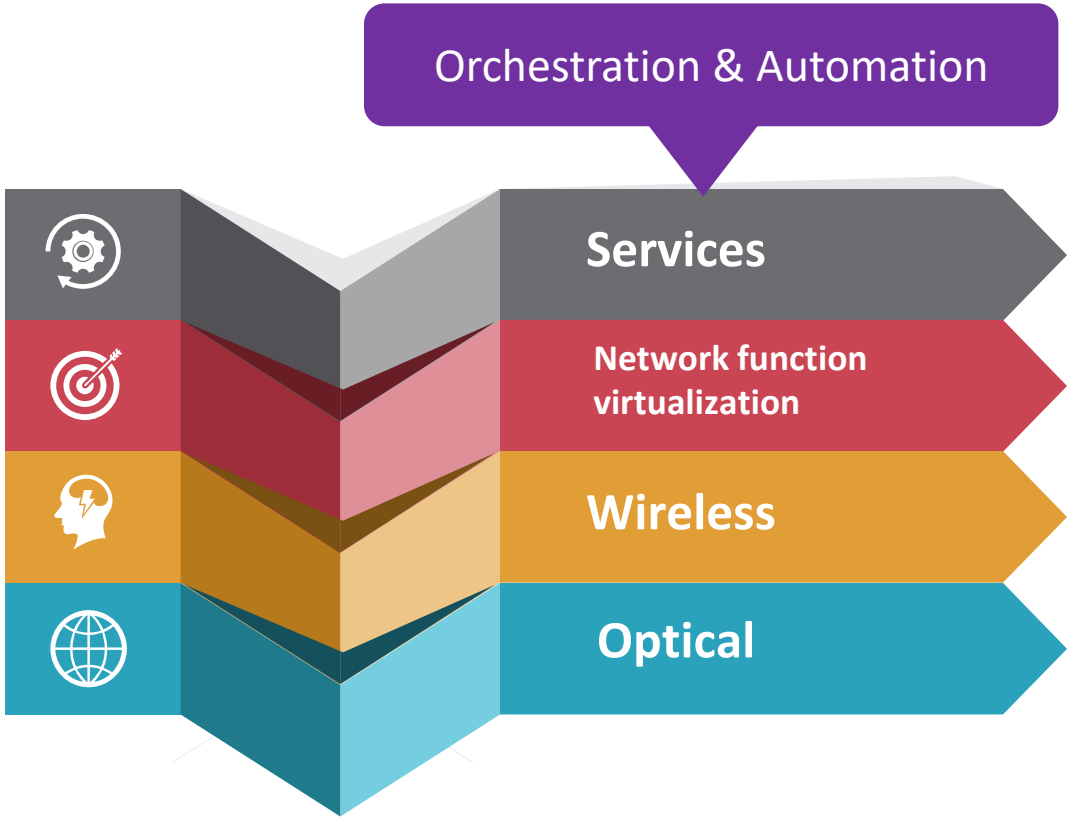
# SURFnet8 Service Layer scope

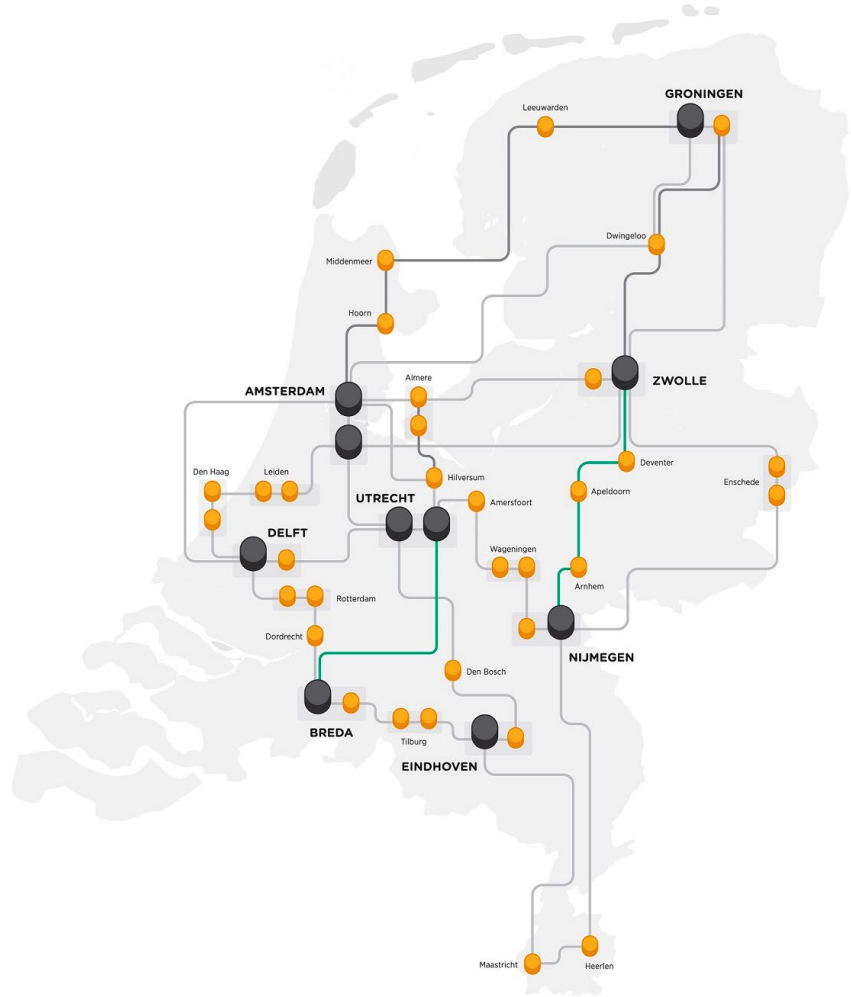
## 280 locations, 400 network equipment, 630 services




SURF NET 8

# From network generations to technology domains


SURF NET

# Automating the network, why?

- **Short term:**
  - Clean and correct administration (CMDB, Network monitoring)
  - Consistent and predictable service delivery for customers
  - Better insight in processes
- **Long term:**
  - Composed services
  - Faster en predictable service delivery
  - Self-service possibilities for customers/partners
  - Life-cycle management of services
  - Bypassing Challenges for finding skilled engineers for now and in future, support with a small team
  - More time for innovation

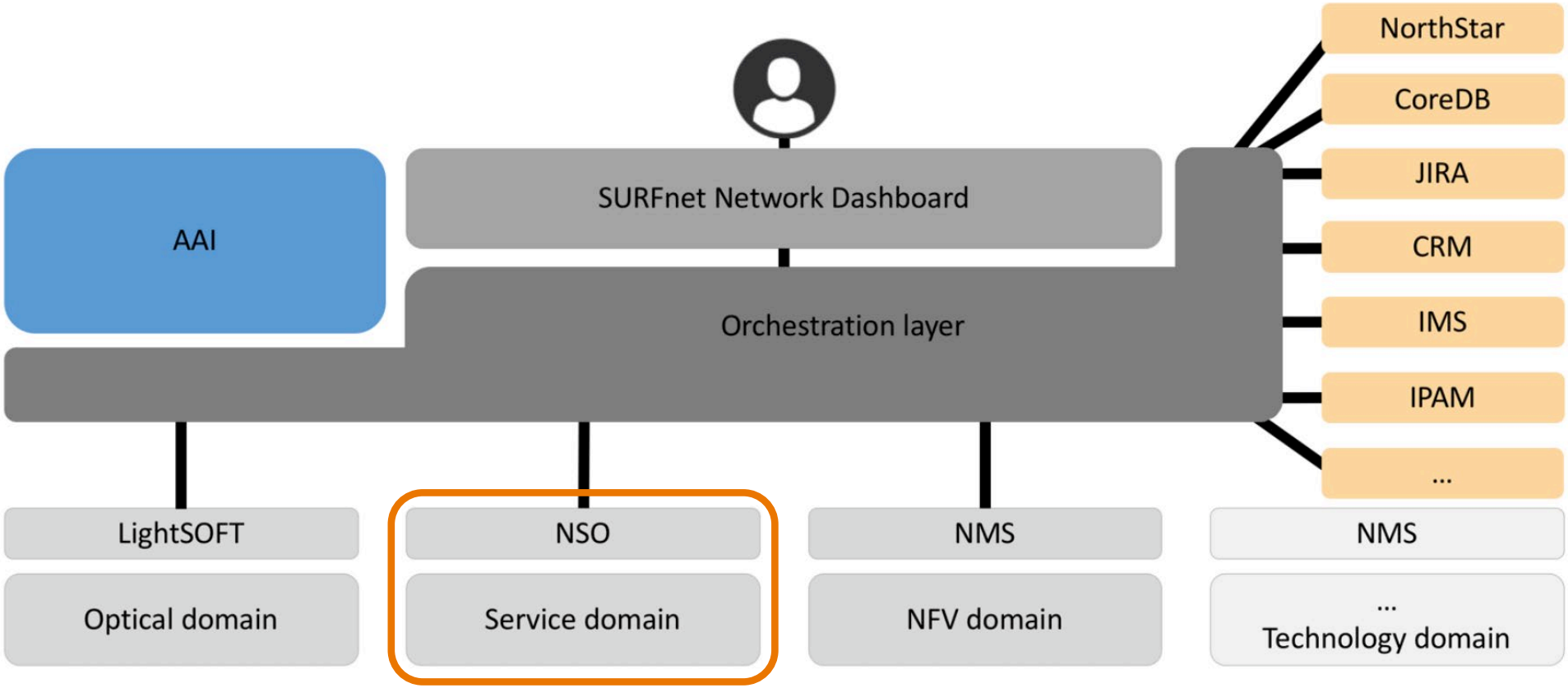
# How did we start this project?

- MT commitment for continuous investment
- Formation of an automation team
  - Automation team of 3 SURFnet engineers with SW background
  - Attracted 5 SW developer (external)
  - 2 sw developer/test engineers
- A lot of discussions and meetings – convincing people
- SW development in two week sprints – sharing progress with stakeholders

# Ingredients for Orchestration utopia

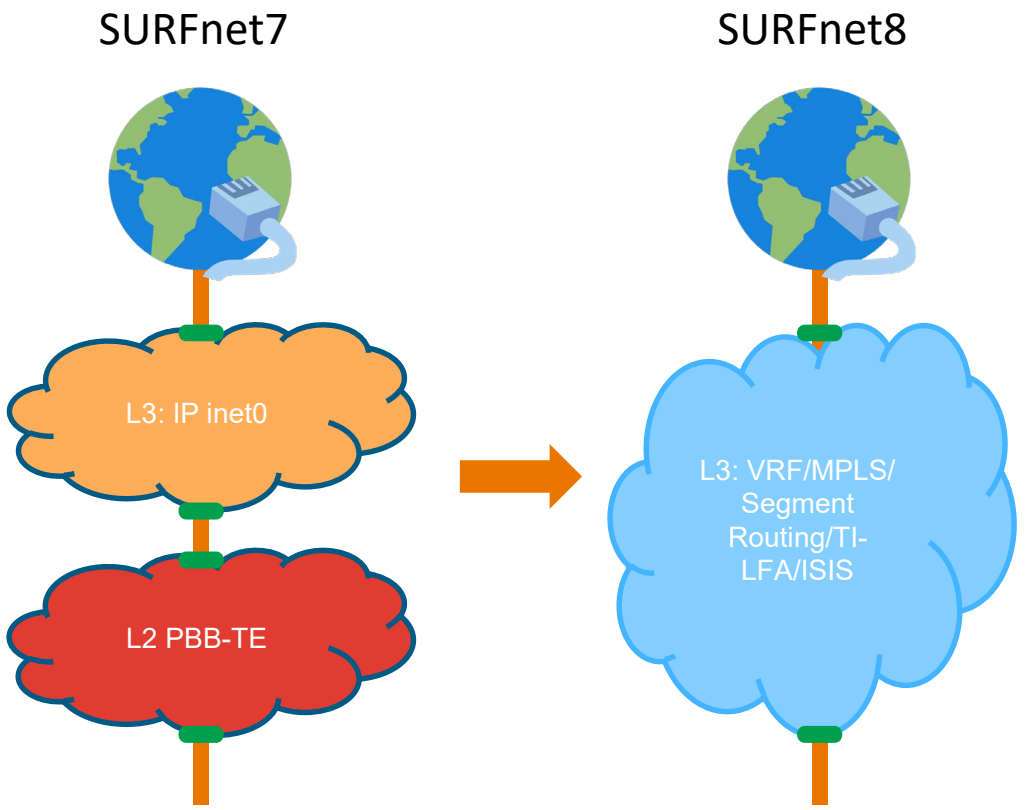
- Single Points of Truth
- Solid information model
- Standardized interfaces
- Automated administration
- Predetermined service delivery
- Unit and Integration tests

# Highlevel overview network automation architecture



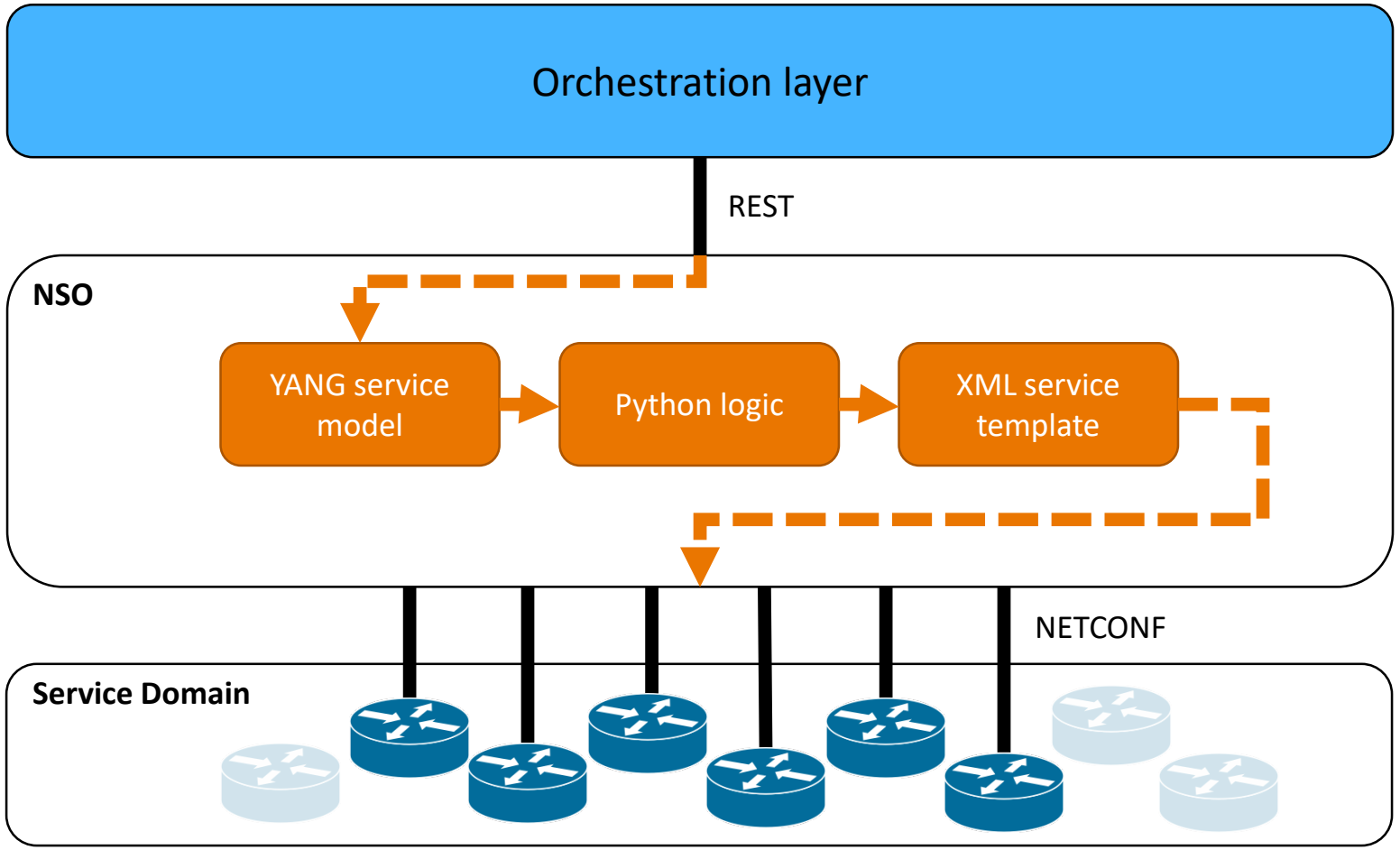
# Simplify the network as possible

- Single HW vendor
- One service domain
  - All MPLS
  - Dynamic control plane – Segment Routing
- One OS & chipset
- Each node has equal role/function
- Northbound API → netconf



# Integrating with our Service Domain using NSO

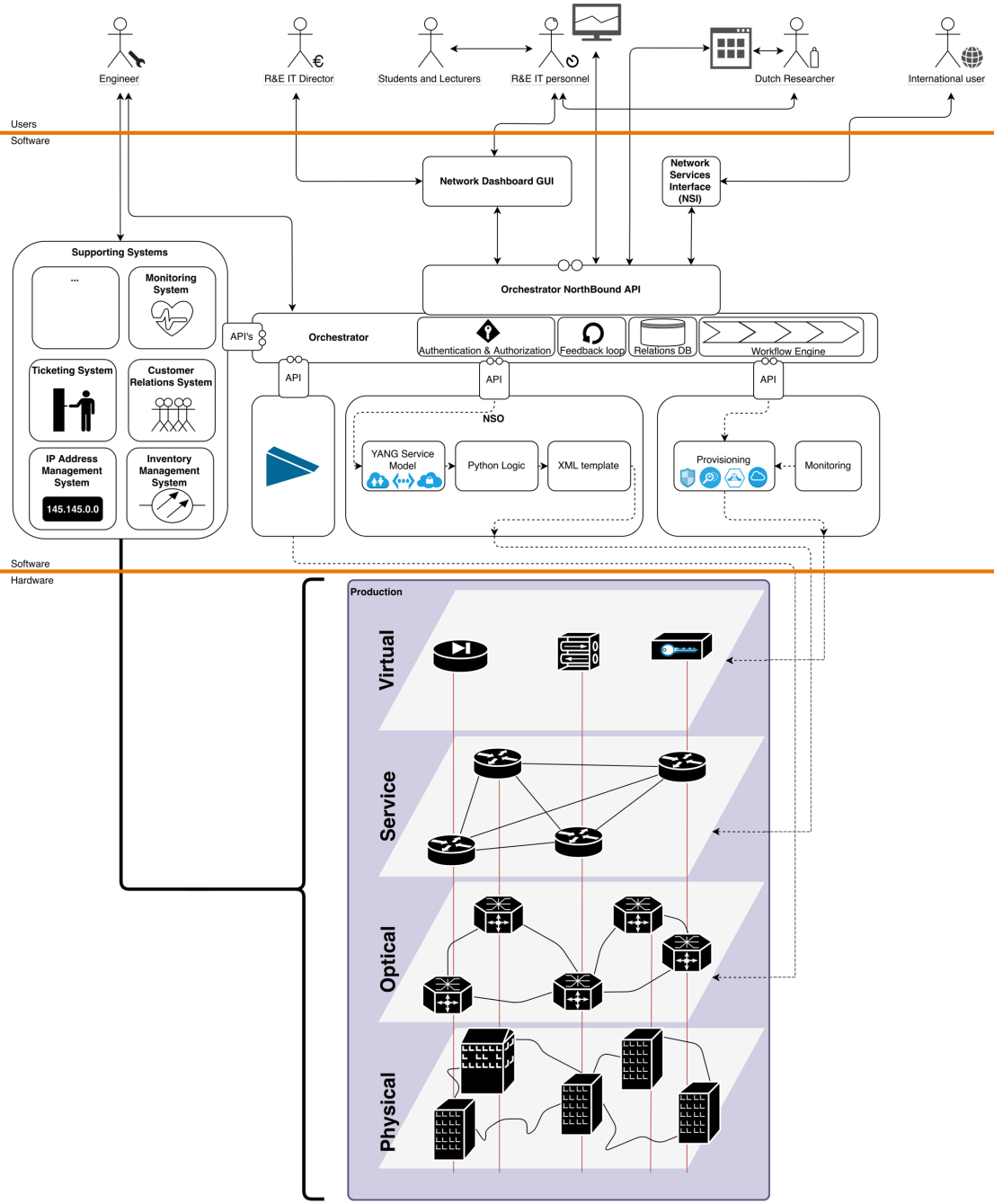
```
augment /ncs:services {
    list ptp {
        tailf:info "Provide unique port ID";
        key name;
        unique "device interface";

        uses ncs:service-data;
        ncs:servicepoint ptp-servicepoint;

        leaf name {
            tailf:info "<uuid:string>";
            mandatory true;
            type string;
        }

        leaf device {
            tailf:info "Select device";
            mandatory true;
            type leafref {
                path "/ncs:devices/ncs:device/ncs:name";
            }
        }

        leaf interface {
            tailf:info "Interface on device (eg xe-1/0/2)";
            mandatory true;
            type string;
        }

        uses surfnet:workflow_customer_service;

        leaf speed {
            when "not( starts-with(../interface, 'ae'))";
            tailf:info "Interface speed";
            type enumeration {
                enum "1g";
                enum "10g";
                enum "40g";
                enum "100g";
                enum "other";
            }
        }
```

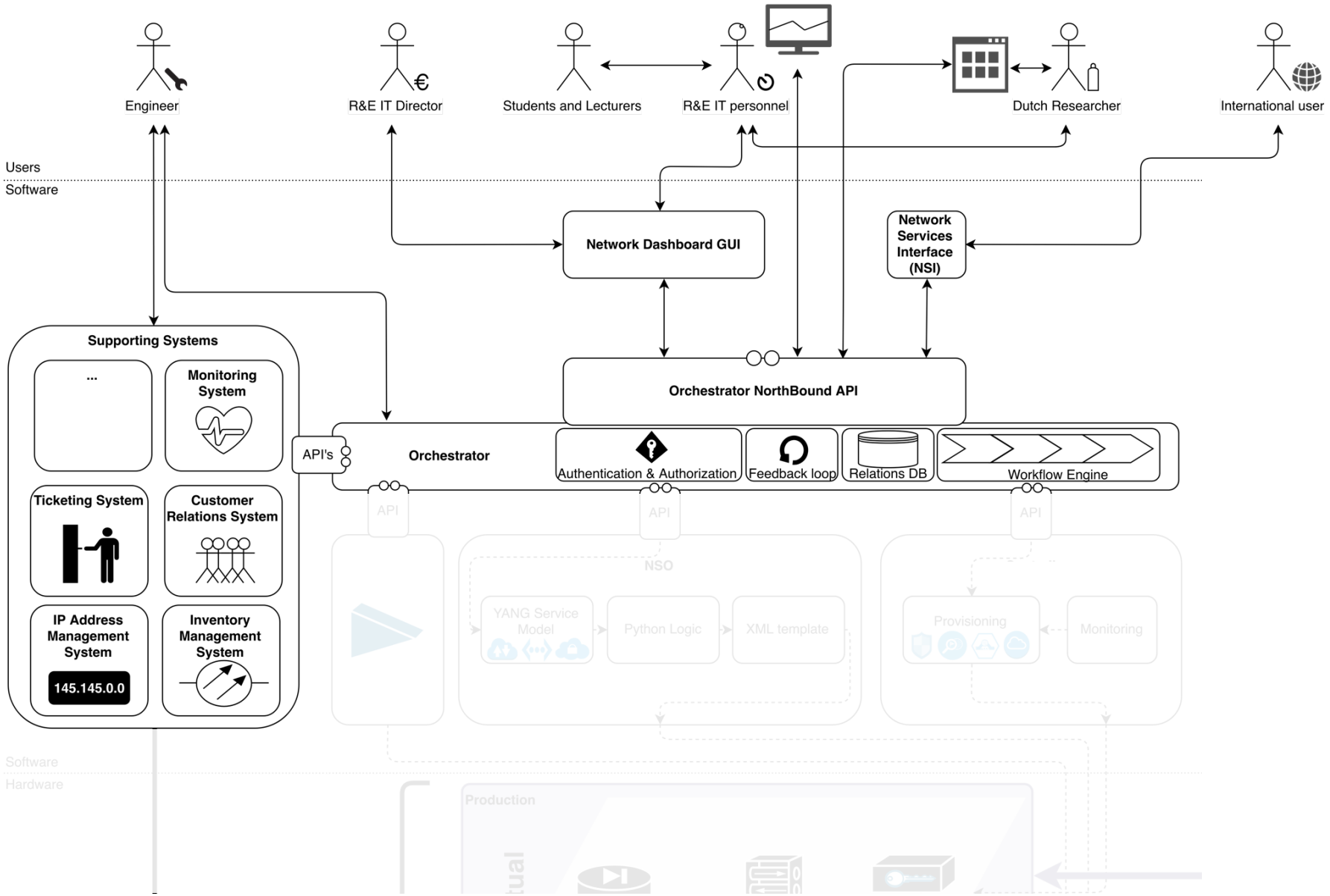


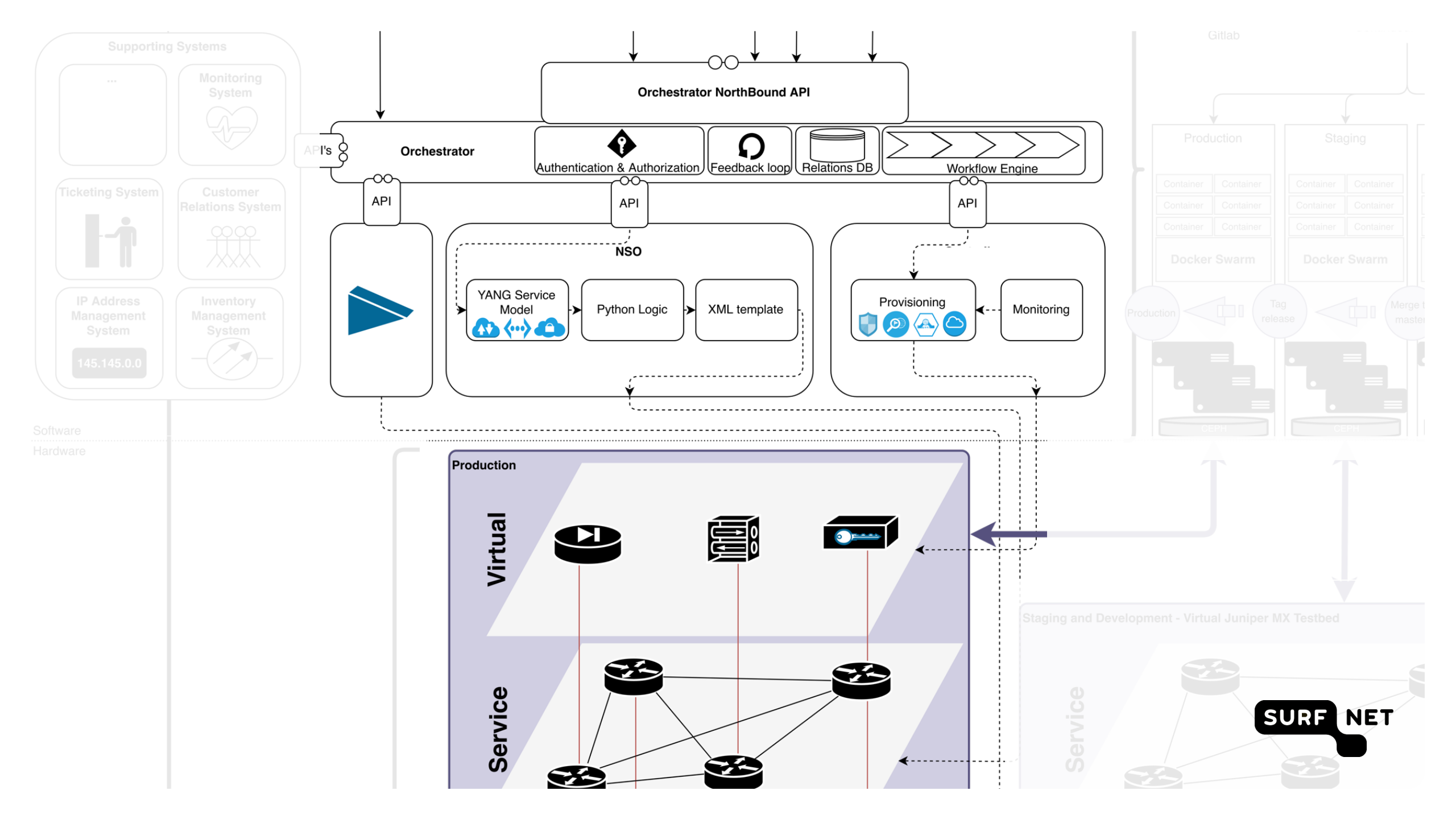
USERS

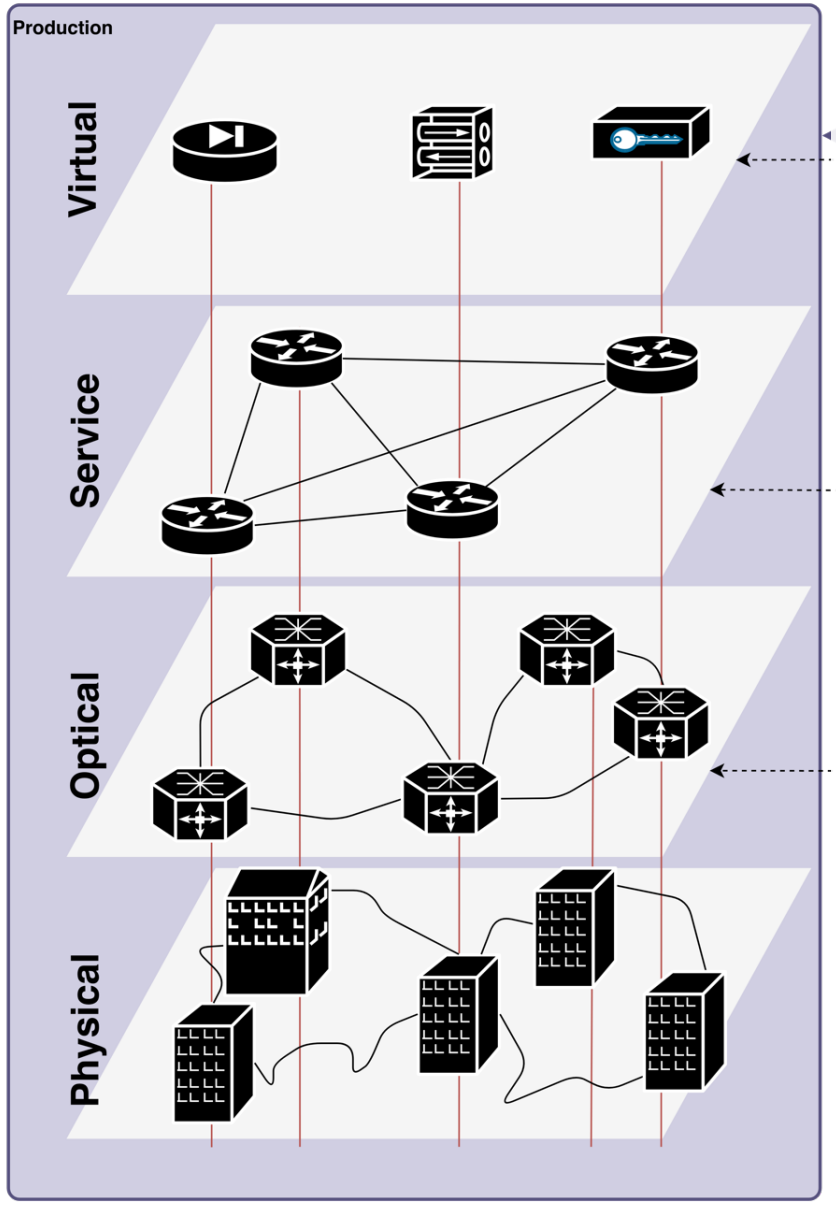
SOFTWARE

HARDWARE

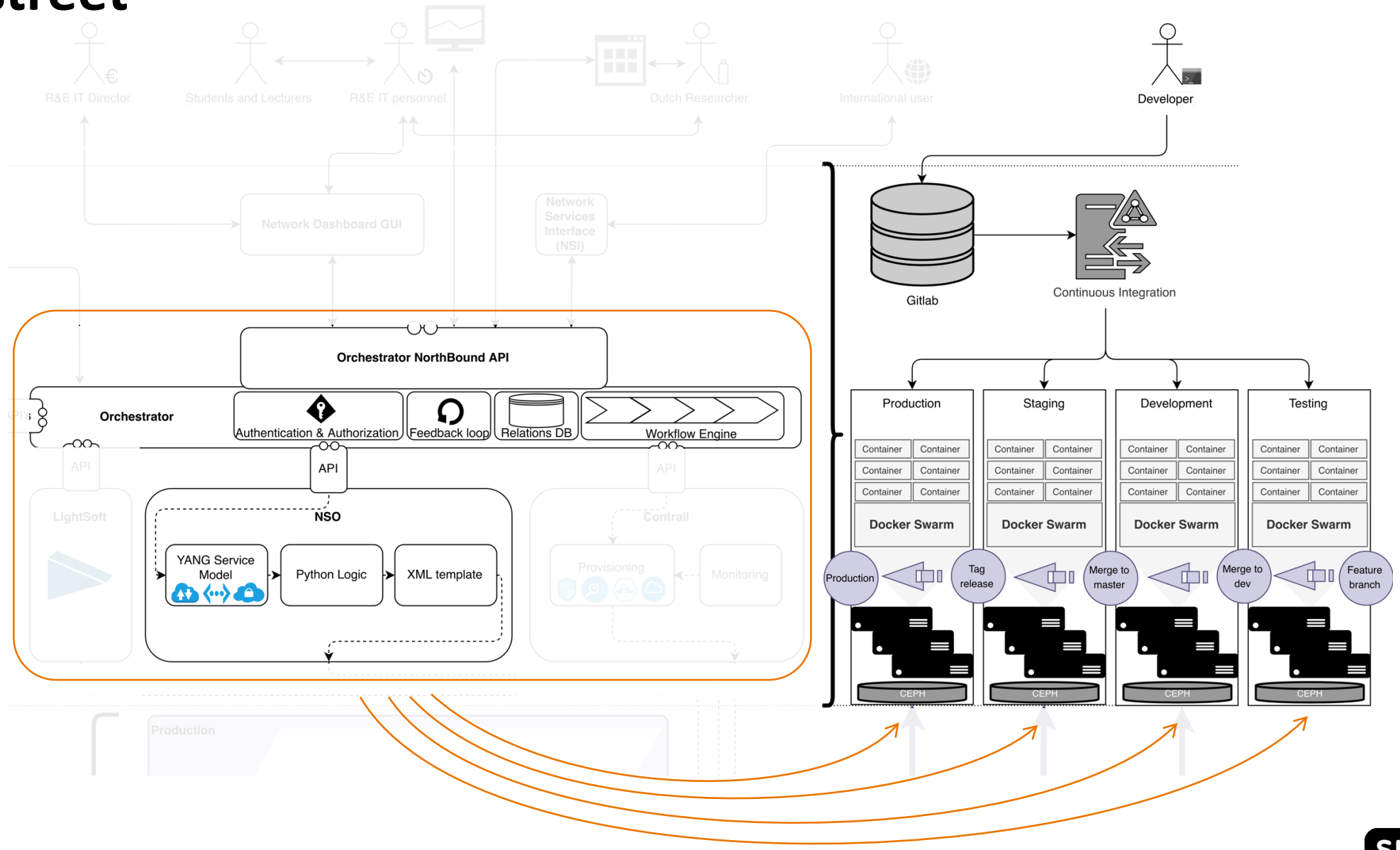


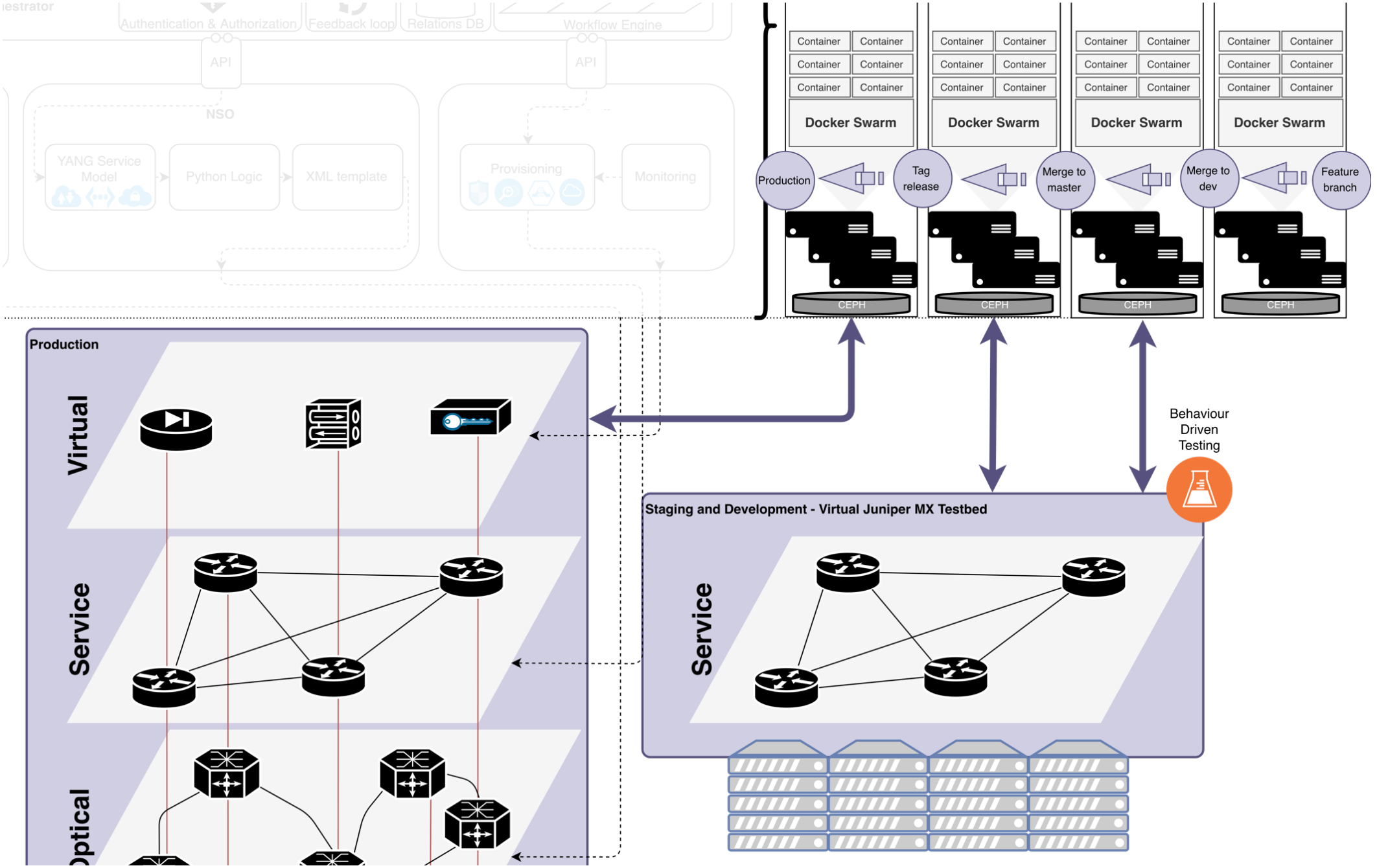

Engineer
R&E IT Director
Students and Lecturers
R&E IT personnel
Dutch Researcher
International user
Users
Software
Network Dashboard GUI
Network Services Interface (NSI)
Supporting Systems
...
Monitoring System
Ticketing System
Customer Relations System
IP Address Management System
145.145.0.0
Inventory Management System
Orchestrator NorthBound API
API's
Orchestrator
Authentication & Authorization
Feedback loop
Relations DB
Workflow Engine
API
API
API
NSO
YANG Service Model
Python Logic
XML template
Provisioning
Monitoring
Software
Hardware
Production


Orchestrator NorthBound API
Orchestrator
API's
Authentication & Authorization
Feedback loop
Relations DB
Workflow Engine
API
API
API
NSO
YANG Service Model
Python Logic
XML template
Provisioning
Monitoring
Production
Virtual
Service
Supporting Systems
Monitoring System
Ticketing System
Customer Relations System
IP Address Management System
145.145.0.0
Inventory Management System
Gitlab
Production
Staging
Container
Docker Swarm
Tag release
Software
Hardware
Staging and Development - Virtual Juniper MX Testbed
SURF NET


Production
Virtual
Service
Optical
Physical

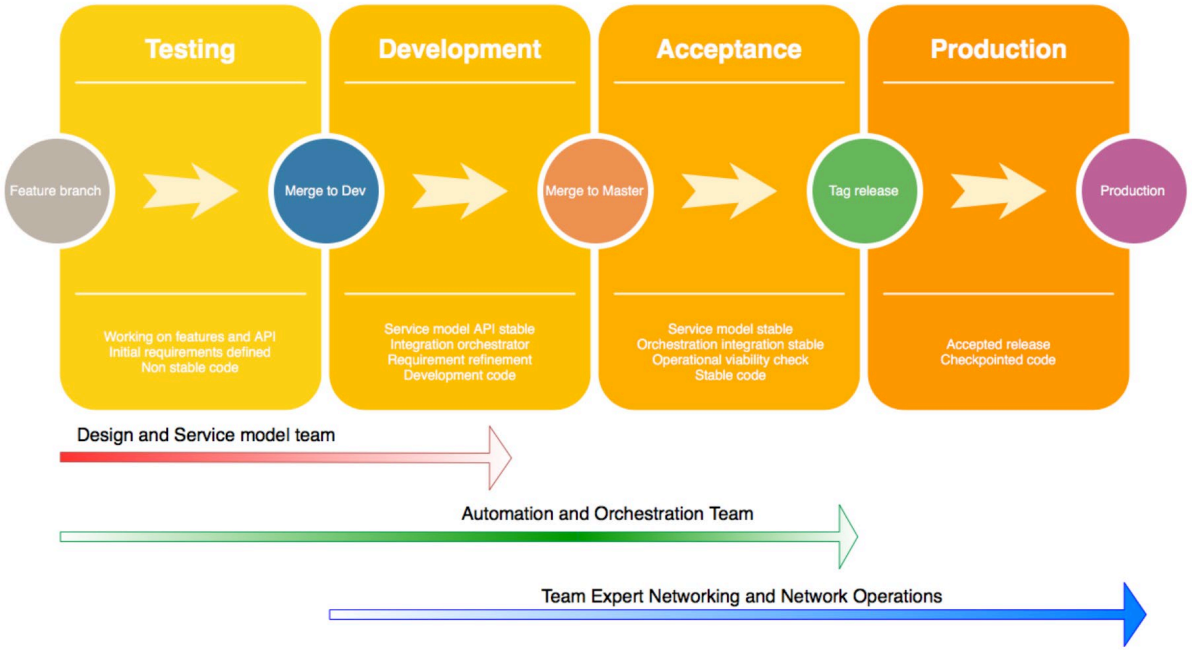
# DTAP street


R&E IT Director
Students and Lecturers
R&E IT personnel
Dutch Researcher
International user
Developer
Network Dashboard GUI
Network Services Interface (NSI)
Gitlab
Continuous Integration
Orchestrator NorthBound API
Orchestrator
Authentication & Authorization
Feedback loop
Relations DB
Workflow Engine
API
NSO
YANG Service Model
Python Logic
XML template
LightSoft
Contrail
Provisioning
Monitoring
Production
Staging
Development
Testing
Container
Docker Swarm
CEPH
Production
Tag release
Merge to master
Merge to dev
Feature branch


Container
Docker Swarm
CEPH
Production
Tag release
Merge to master
Merge to dev
Feature branch
Behaviour Driven Testing
Production
Virtual
Service
Optical
Staging and Development - Virtual Juniper MX Testbed
Service

# How does a Virtual Testbed with NSO help?

- Reliably testing of all network services
- Integration testing with the orchestration software and the network
- Fully reconfigurable and  ablility to  “reset” to a known state
- Full testing and development environment



**Topology Details**

Add Instance | Add Bridge | Add External Bridge | New Note | 16 | Add Not

Name: dev

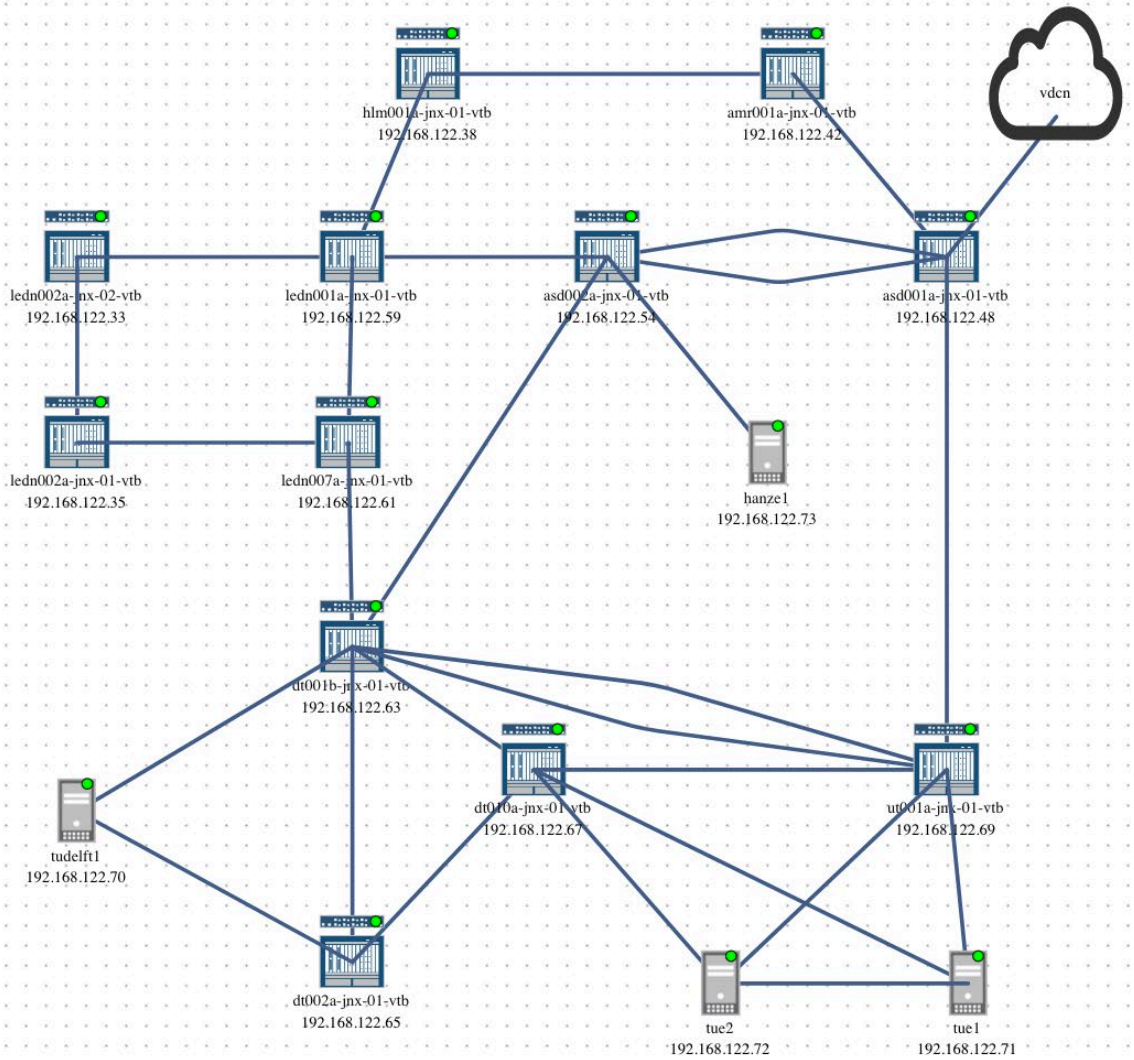
Description: Topology for dev environment

Edit | Update | Delete | Clone

**Stack Status**

- t18_asd001a-jnx-01-vtb_child
- t18_dt002a-jnx-01-vtb_child
- t18_ledn002a-jnx-02-vtb_child
- t18_asd002a-jnx-01-vtb
- t18_ut001a-jnx-01-vtb
- t18_tue1
- t18_tue2
- t18_ledn002a-jnx-02-vtb
- t18_amr001a-jnx-01-vtb_child
- t18_hlm001a-jnx-01-vtb_child
- t18_dt010a-jnx-01-vtb_child
- t18_amr001a-jnx-01-vtb

**Automation Tools**




NSO

IPAM

IMS

Orchestration

# Behaviour driven testing

```
Feature: Node create

  Node create is used to add unique identifying config to a node that is not already added in bootstrap


  Scenario Outline: First node_create
    Given we have a default topology in reset state
     When we apply node_create to <device>
     Then device <device> will have an ipv4 address on lo0
     Then device <device> will have an ipv6 address on lo0
     Then device <device> will have an unique iso address on lo0
     Then device <device> will have an segment routing id

    Examples:
      | device             |
      | amr001a-jnx-01-vtb |


  Scenario Outline: Other node_creates
    Given we have a default topology from a previous test
     When we apply node_create to <device>
     Then device <device> will have an ipv4 address on lo0
     Then device <device> will have an ipv6 address on lo0
     Then device <device> will have an unique iso address on lo0
     Then device <device> will have an segment routing id

    Examples:
      | device             |
      | asd001a-jnx-01-vtb |
      | asd002a-jnx-01-vtb |
      | dt001b-jnx-01-vtb  |
      | dt002a-jnx-01-vtb  |
```

# Behaviour driven testing

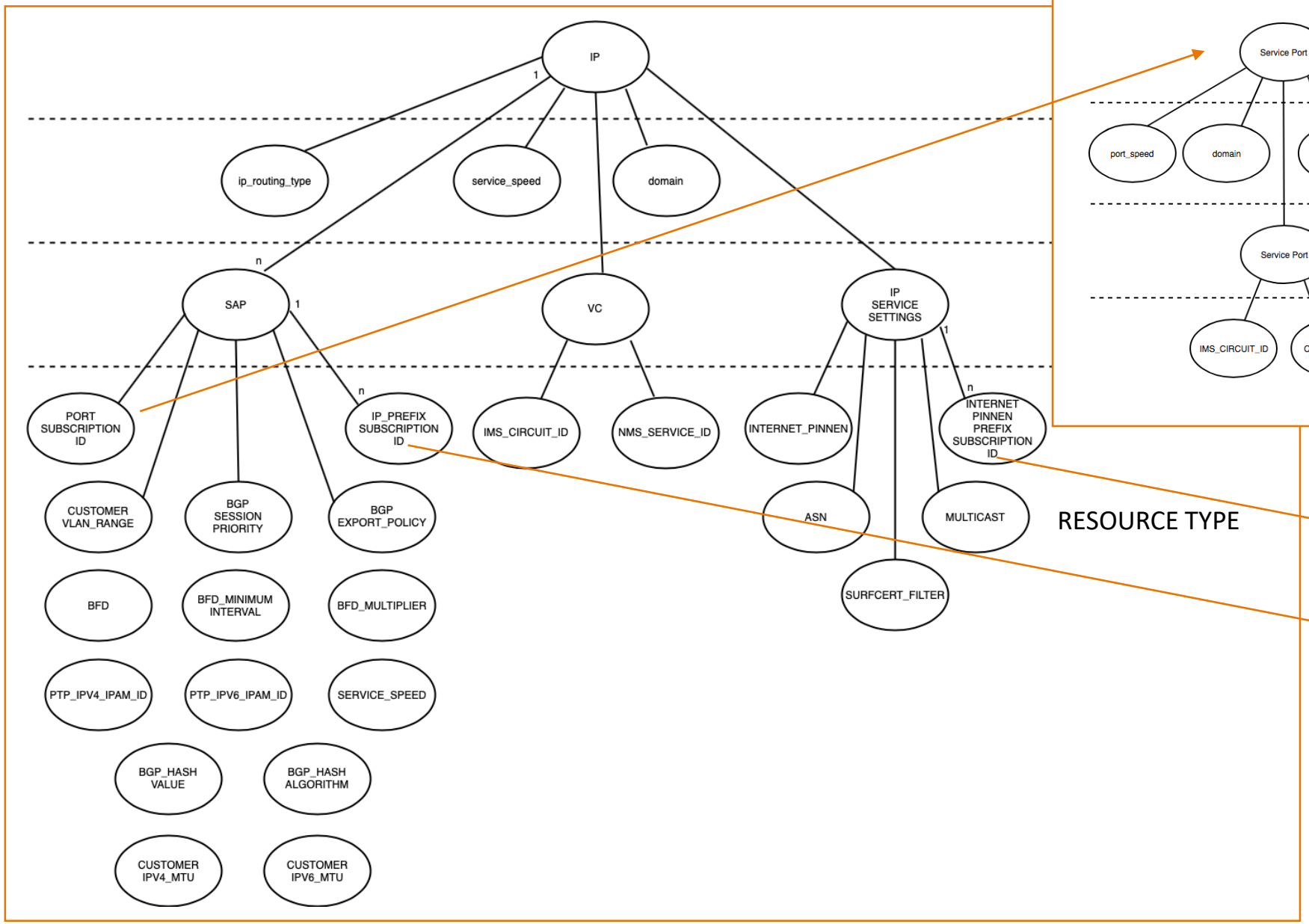
```python
@then('device {device} will have an ipv4 address on {interface}')
def step_impl(context, device, interface):
    client = context.connect_to_vmx(context, device)

    for line in client.xpath('//interfaces/interface[name="%s"]/unit/family/inet/address/name/text()' % interface):
        try:
            ipaddress.ip_network(line, strict=False)
            return
        except:
            pass

    assert False, "No valid ip found"


@then('device {device} will have an ipv6 address on {interface}')
def step_impl(context, device, interface):
    client = context.connect_to_vmx(context, device)

    for line in client.xpath('//interfaces/interface[name="%s"]/unit/family/inet6/address/name/text()' % interface):
        try:
            ipaddress.ip_network(line, strict=False)
            return
        except:
            pass

    assert False, "No valid ip found"


@then(u'device {device} will have an segment routing id')
def step_impl(context, device):
    client = context.connect_to_vmx(context, device)

    for line in client.xpath('//protocols/isis/source-packet-routing/node-segment/ipv4-index/text()'):
        try:
            for line in client.xpath('//protocols/isis/source-packet-routing/node-segment/ipv6-index/text()'):
                try:
                    return
                except:
                    pass
        except:
            pass

    assert False, "No valid iso address found"


ISO_ADDRESSES = jmespath.compile(
    '[].config."junos:configuration".interfaces.interface[].unit[].family.iso.address[].name'
)
```

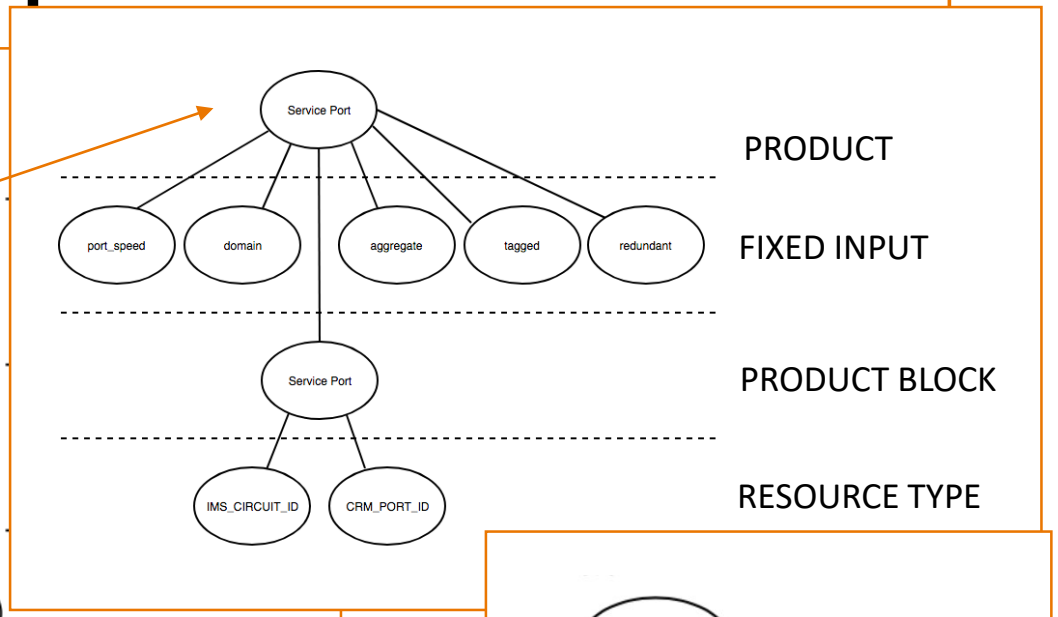
```
Feature: Node create # features/node_create.feature:1
  Node create is used to add unique identifying config to a node that is not already added in bootstrap
  Scenario Outline: First node_create -- @1.1                                  # features/node_create.feature:16
    Given we have a default topology in reset state                            # features/steps/topology.py:57
    When we apply node_create to amr001a-jnx-01-vtb                           # features/steps/service.py:20
    Then device amr001a-jnx-01-vtb will have an ipv4 address on lo0           # features/steps/device.py:10
    Then device amr001a-jnx-01-vtb will have an ipv6 address on lo0           # features/steps/device.py:24
    Then device amr001a-jnx-01-vtb will have an unique iso address on lo0     # features/steps/device.py:60
    Then device amr001a-jnx-01-vtb will have an segment routing id            # features/steps/device.py:38

  Scenario Outline: Other node_creates -- @1.1                                # features/node_create.feature:29
    Given we have a default topology from a previous test                     # features/steps/topology.py:95
    When we apply node_create to asd001a-jnx-01-vtb                           # features/steps/service.py:20
    Then device asd001a-jnx-01-vtb will have an ipv4 address on lo0           # features/steps/device.py:10
    Then device asd001a-jnx-01-vtb will have an ipv6 address on lo0           # features/steps/device.py:24
    Then device asd001a-jnx-01-vtb will have an unique iso address on lo0     # features/steps/device.py:60
    Then device asd001a-jnx-01-vtb will have an segment routing id            # features/steps/device.py:38

  Scenario Outline: Other node_creates -- @1.2                                # features/node_create.feature:30
    Given we have a default topology from a previous test                     # features/steps/topology.py:95
    When we apply node_create to asd002a-jnx-01-vtb                           # features/steps/service.py:20
    Then device asd002a-jnx-01-vtb will have an ipv4 address on lo0           # features/steps/device.py:10
    Then device asd002a-jnx-01-vtb will have an ipv6 address on lo0           # features/steps/device.py:24
    Then device asd002a-jnx-01-vtb will have an unique iso address on lo0     # features/steps/device.py:60
    Then device asd002a-jnx-01-vtb will have an segment routing id            # features/steps/device.py:38

  Scenario Outline: Other node_creates -- @1.3                                # features/node_create.feature:31
    Given we have a default topology from a previous test                     # features/steps/topology.py:95
    When we apply node_create to dt001b-jnx-01-vtb                            # features/steps/service.py:20
    Then device dt001b-jnx-01-vtb will have an ipv4 address on lo0            # features/steps/device.py:10
    Then device dt001b-jnx-01-vtb will have an ipv6 address on lo0            # features/steps/device.py:24
    Then device dt001b-jnx-01-vtb will have an unique iso address on lo0      # features/steps/device.py:60
    Then device dt001b-jnx-01-vtb will have an segment routing id             # features/steps/device.py:38

  Scenario Outline: Other node_creates -- @1.4                                # features/node_create.feature:32
    Given we have a default topology from a previous test                     # features/steps/topology.py:95
    When we apply node_create to dt002a-jnx-01-vtb                            # features/steps/service.py:20
    Then device dt002a-jnx-01-vtb will have an ipv4 address on lo0            # features/steps/device.py:10
    Then device dt002a-jnx-01-vtb will have an ipv6 address on lo0            # features/steps/device.py:24
    Then device dt002a-jnx-01-vtb will have an unique iso address on lo0      # features/steps/device.py:60
    Then device dt002a-jnx-01-vtb will have an segment routing id             # features/steps/device.py:38
```

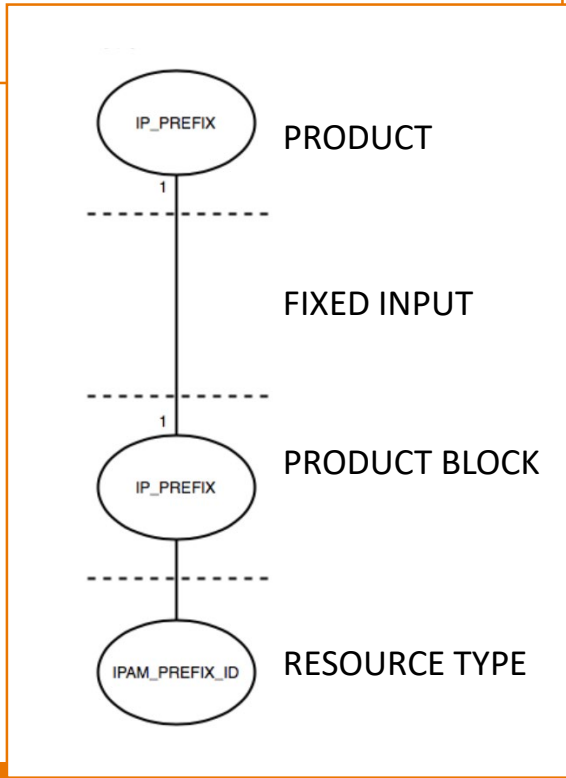
# NSO service models products

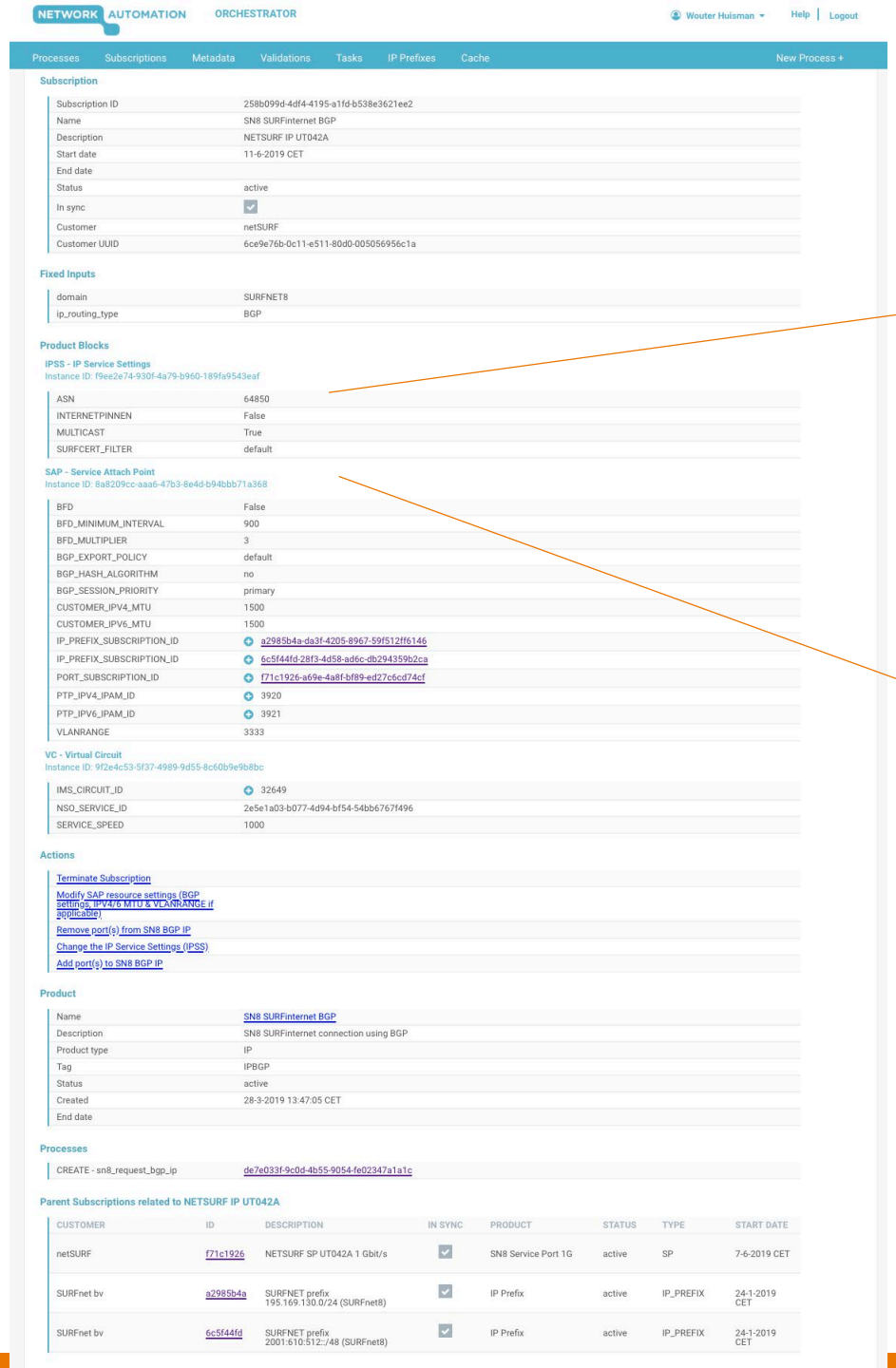
- Customer facing products
  - Service Ports
  - IP Service
  - Lightpath (EPL)
  - ELAN Service
  - L3VPN Service
- NOC facing product
  - Nodes (network elements)
  - Core Links (between nodes)
  - IP peerings
  - IPv4/IPv6 prefix (customer prefix administration)

# Glue everything together: products and product blocks



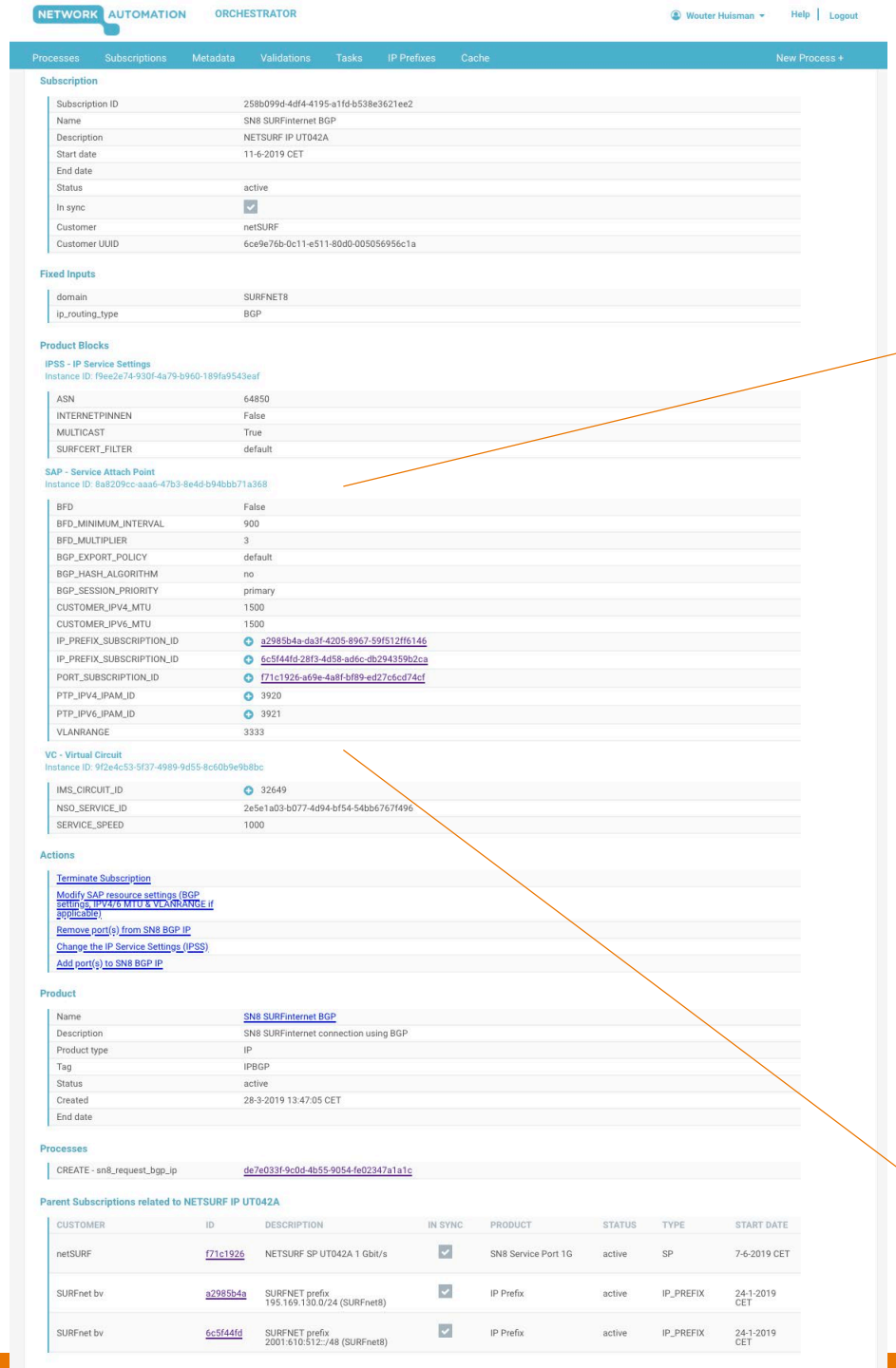



# Generic Service Setttings for IP service

## IPSS - IP Service Settings

Instance ID: f9ee2e74-930f-4a79-b960-189fa9543eaf

| | |
|---|---|
| ASN | 64850 |
| INTERNETPINNEN | False |
| MULTICAST | True |
| SURFCERT_FILTER | default |

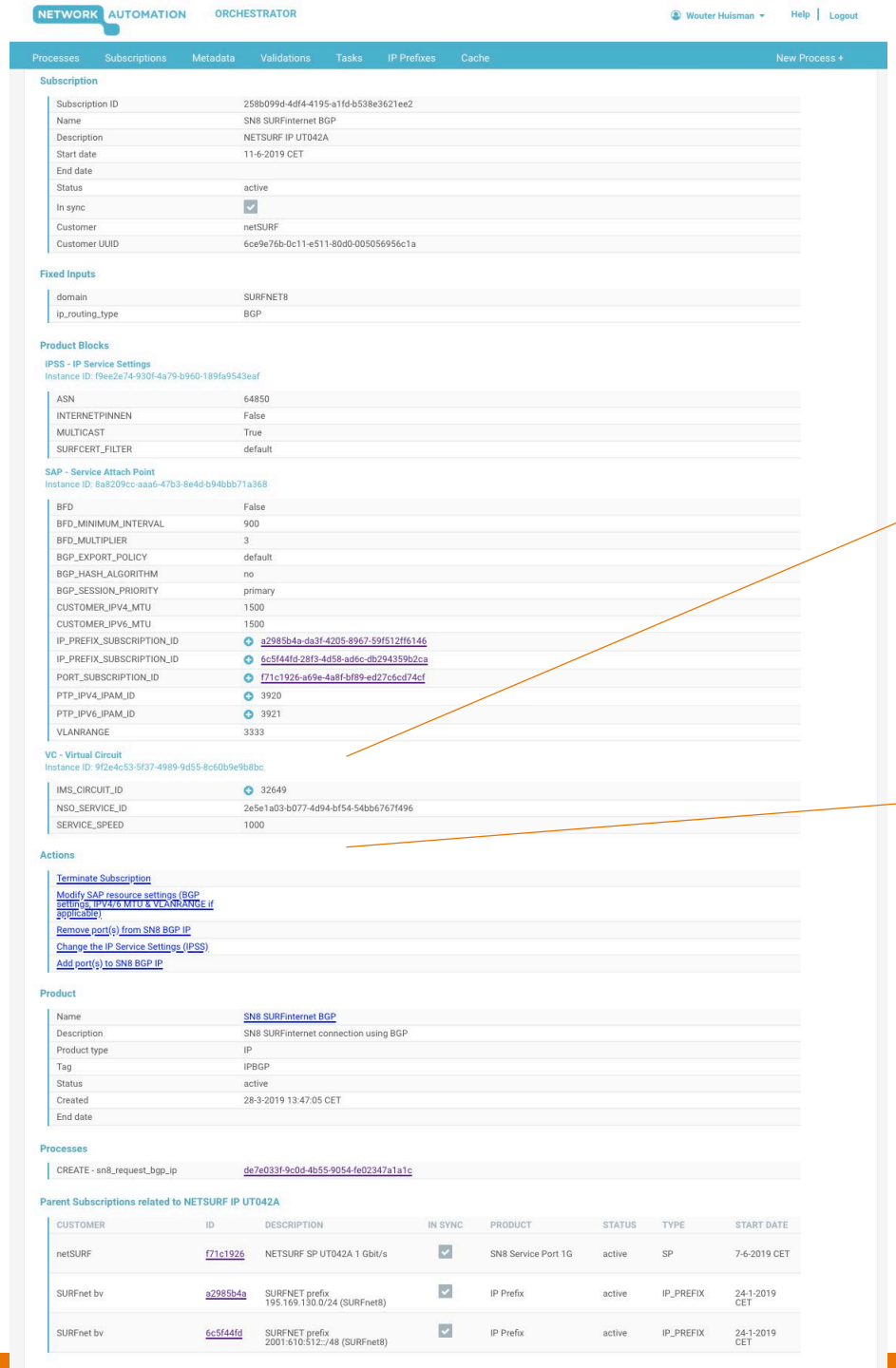



# Service Attach Point with service parameters

## SAP - Service Attach Point

Instance ID: 8a8209cc-aaa6-47b3-8e4d-b94bbb71a368

| | |
|---|---|
| BFD | False |
| BFD_MINIMUM_INTERVAL | 900 |
| BFD_MULTIPLIER | 3 |
| BGP_EXPORT_POLICY | default |
| BGP_HASH_ALGORITHM | no |
| BGP_SESSION_PRIORITY | primary |
| CUSTOMER_IPV4_MTU | 1500 |
| CUSTOMER_IPV6_MTU | 1500 |
| IP_PREFIX_SUBSCRIPTION_ID | a2985b4a-da3f-4205-8967-59f512ff6146 |
| IP_PREFIX_SUBSCRIPTION_ID | 6c5f44fd-28f3-4d58-ad6c-db294359b2ca |
| PORT_SUBSCRIPTION_ID | f71c1926-a69e-4a8f-bf89-ed27c6cd74cf |
| PTP_IPV4_IPAM_ID | 3920 |
| PTP_IPV6_IPAM_ID | 3921 |
| VLANRANGE | 3333 |

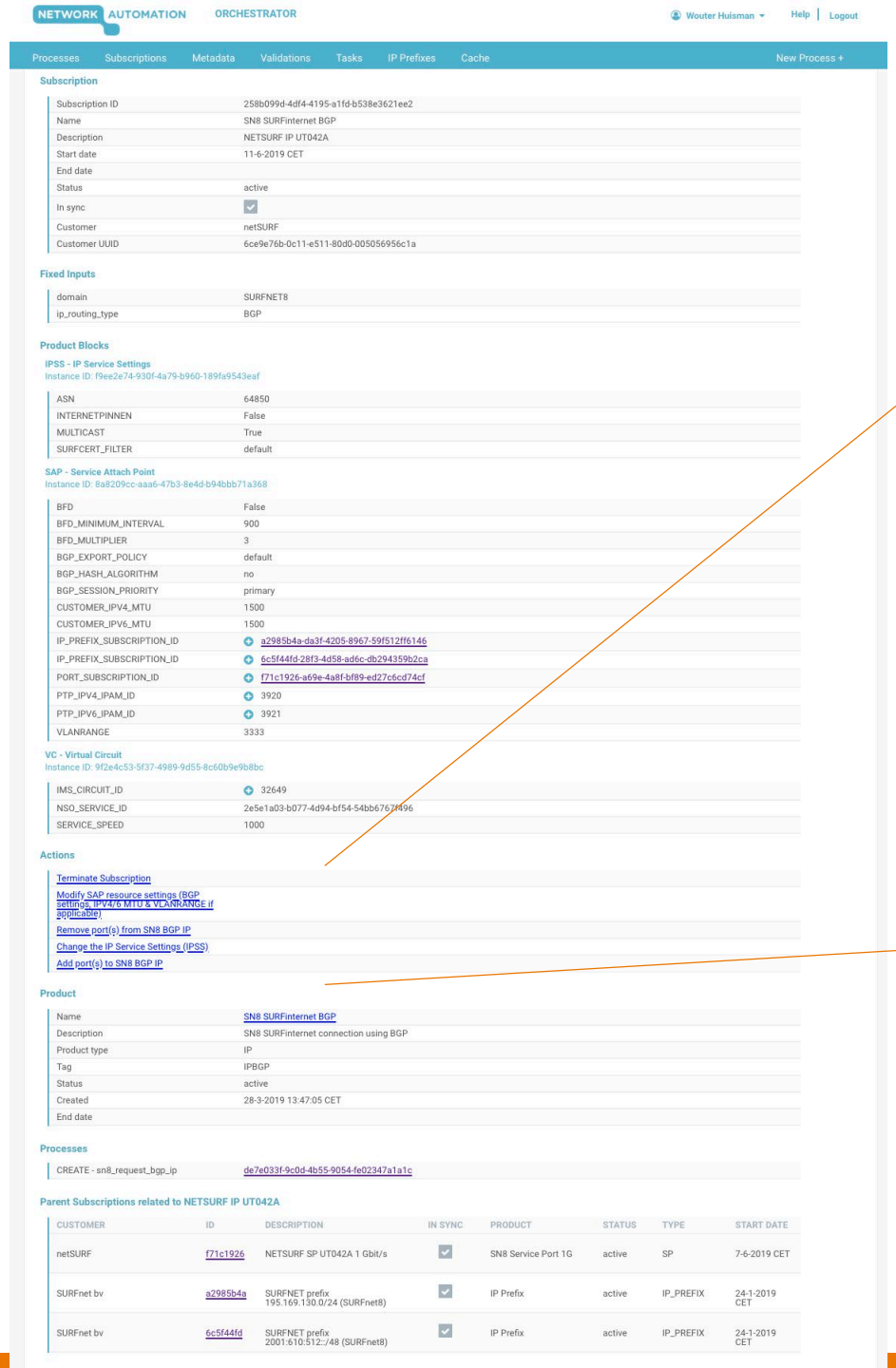



DETAILS WITH PHYSICAL AND LOGICAL CONNECTIVITY LIKE
OPTICS TYPE
NODE / SLOT / PORT
ODF

**VC - Virtual Circuit**

Instance ID: 9f2e4c53-5f37-4989-9d55-8c60b9e9b8bc

| | |
|---|---|
| IMS_CIRCUIT_ID | 32649 |
| NSO_SERVICE_ID | 2e5e1a03-b077-4d94-bf54-54bb6767f496 |
| SERVICE_SPEED | 1000 |

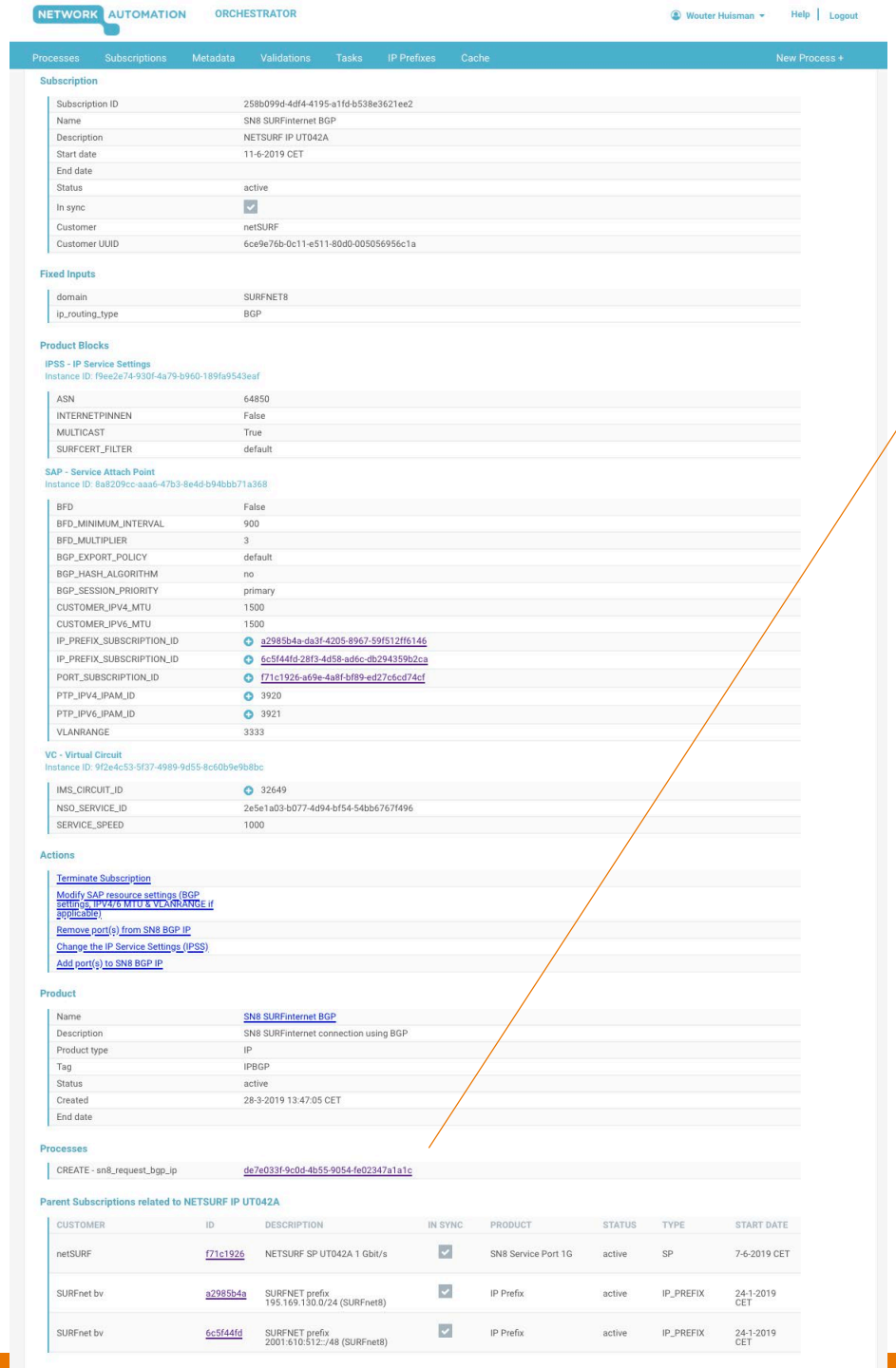


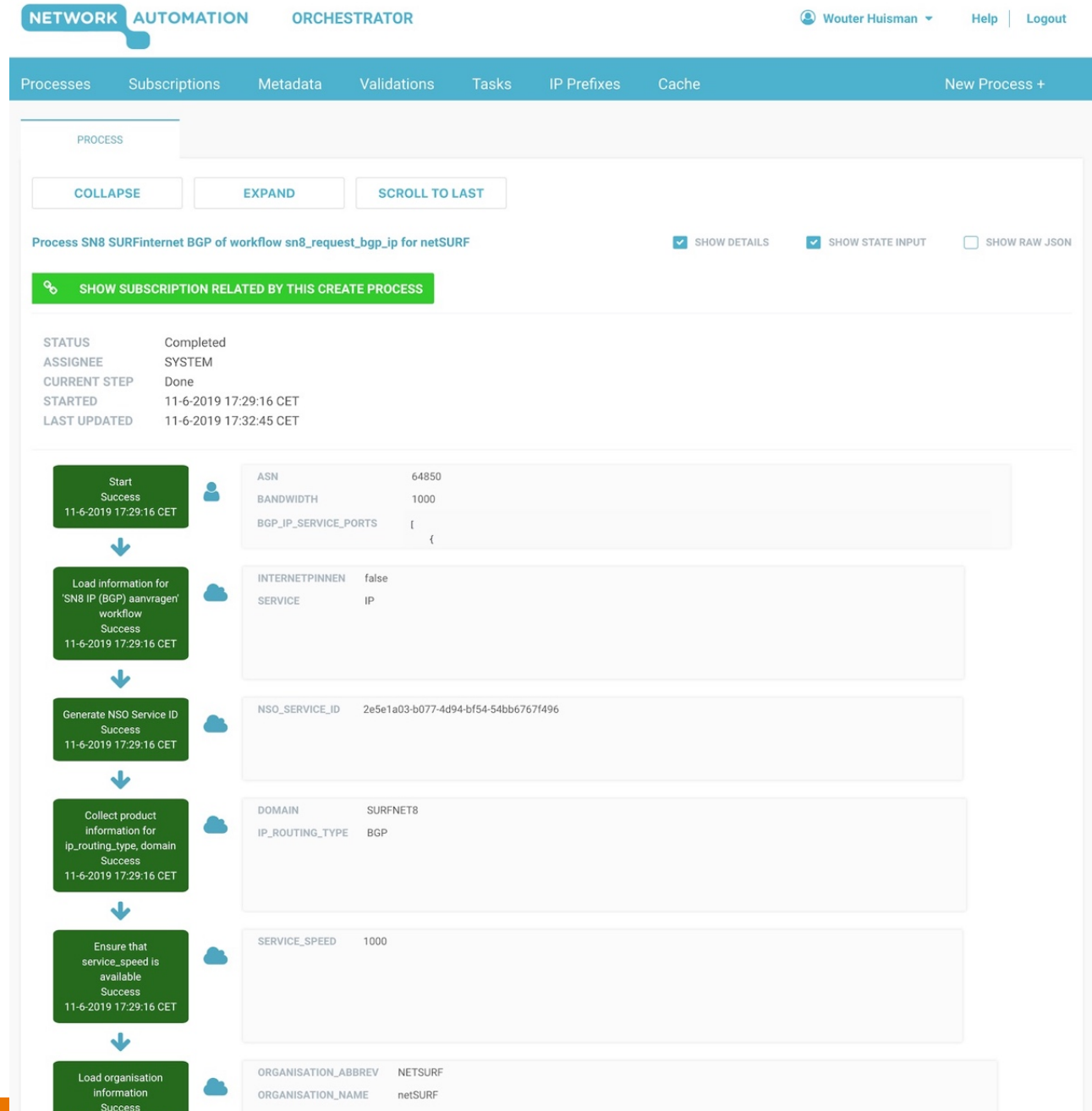

# LIFE CYCLE MANAGEMENT ACTIONS

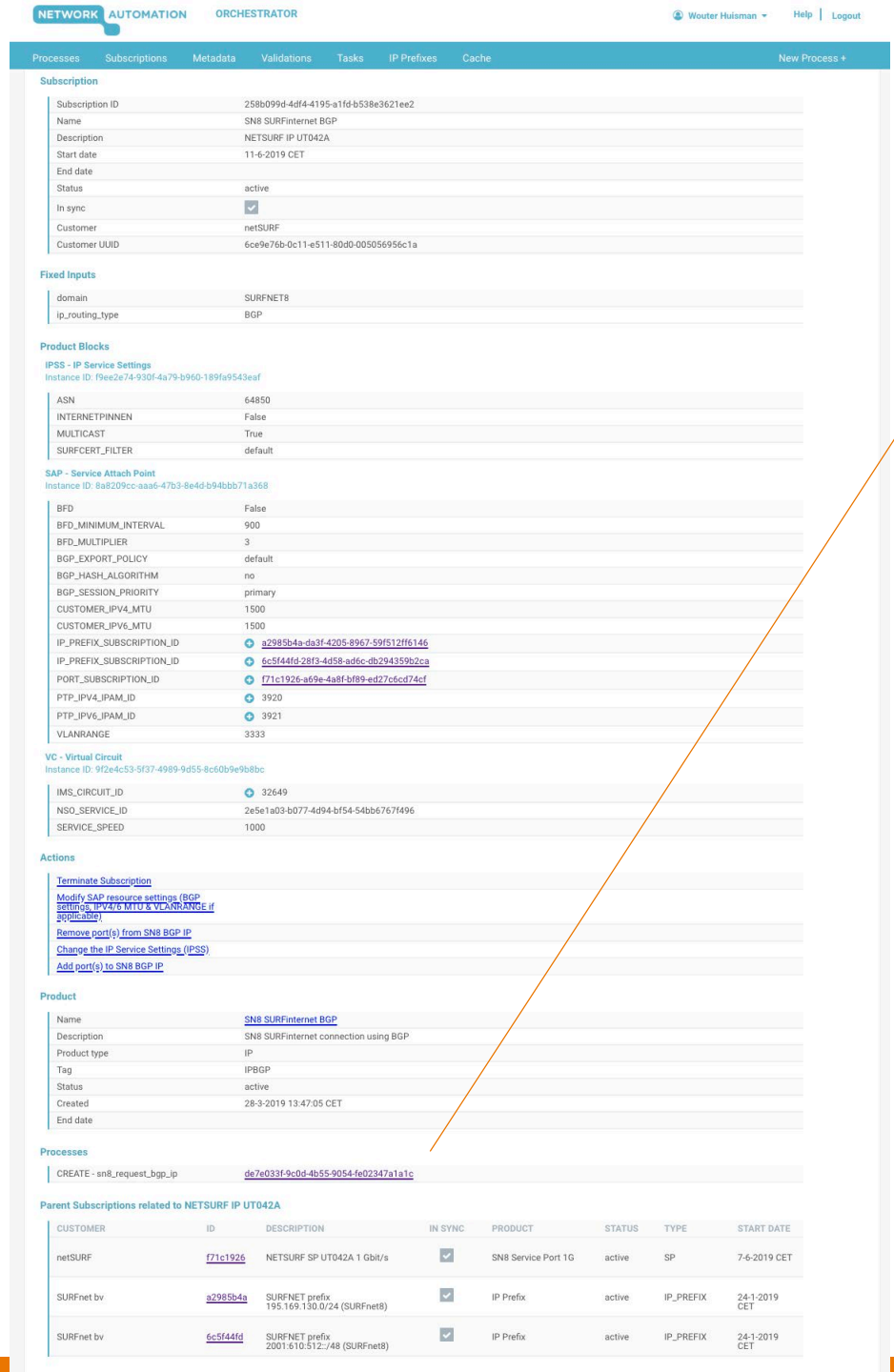
## Actions

- Terminate Subscription
- Modify SAP resource settings (BGP settings, IPV4/6 MTU & VLANRANGE if applicable)
- Remove port(s) from SN8 BGP IP
- Change the IP Service Settings (IPSS)
- Add port(s) to SN8 BGP IP

# WORKFLOW PROCESS OF CREATION






# WORKFLOW PROCESS OF CREATION

Success
11-6-2019 17:29:17 CET

Load port information
Success
11-6-2019 17:29:17 CET

HUMAN_SERVICE_SPEED 1 Gbit/s
IMS_SERVICE_CODE IP
SERVICE_PORTS


```
[
  {
    "ims_service_id": 32619,
    "ims_service_speed": "1000BASE-LX",
    "pop_site": "UT042A",
    "port": "0/1/5",
    "port_id": 676988,
    "port_mode": "tagged",
    "port_name": "UT042A_SP_TAGGED_NETSURF_F71C1926",
    "port_node": "UT042A-JNX-02",
    "port_node_type": "MX204",
    "port_patchposition": "FAKE_SN7-ODF_02/06 (SC/PC)",
    "subscription_id": "f71c1926-a69e-4a8f-bf89-ed27c6cd74cf",
    "tag": "SP",
    "vlan": "3333"
  }
]
```


SUBSCRIPTION_NAME NETSURF IP UT042A

Create subscription
Success
11-6-2019 17:29:17 CET

SUBSCRIPTION_ID 258b099d-4df4-4195-a1fd-b538e3621ee2

Create Process
Subscription relation
Success
11-6-2019 17:29:17 CET

Load IP Prefix
Subscriptions
Success
11-6-2019 17:29:17 CET

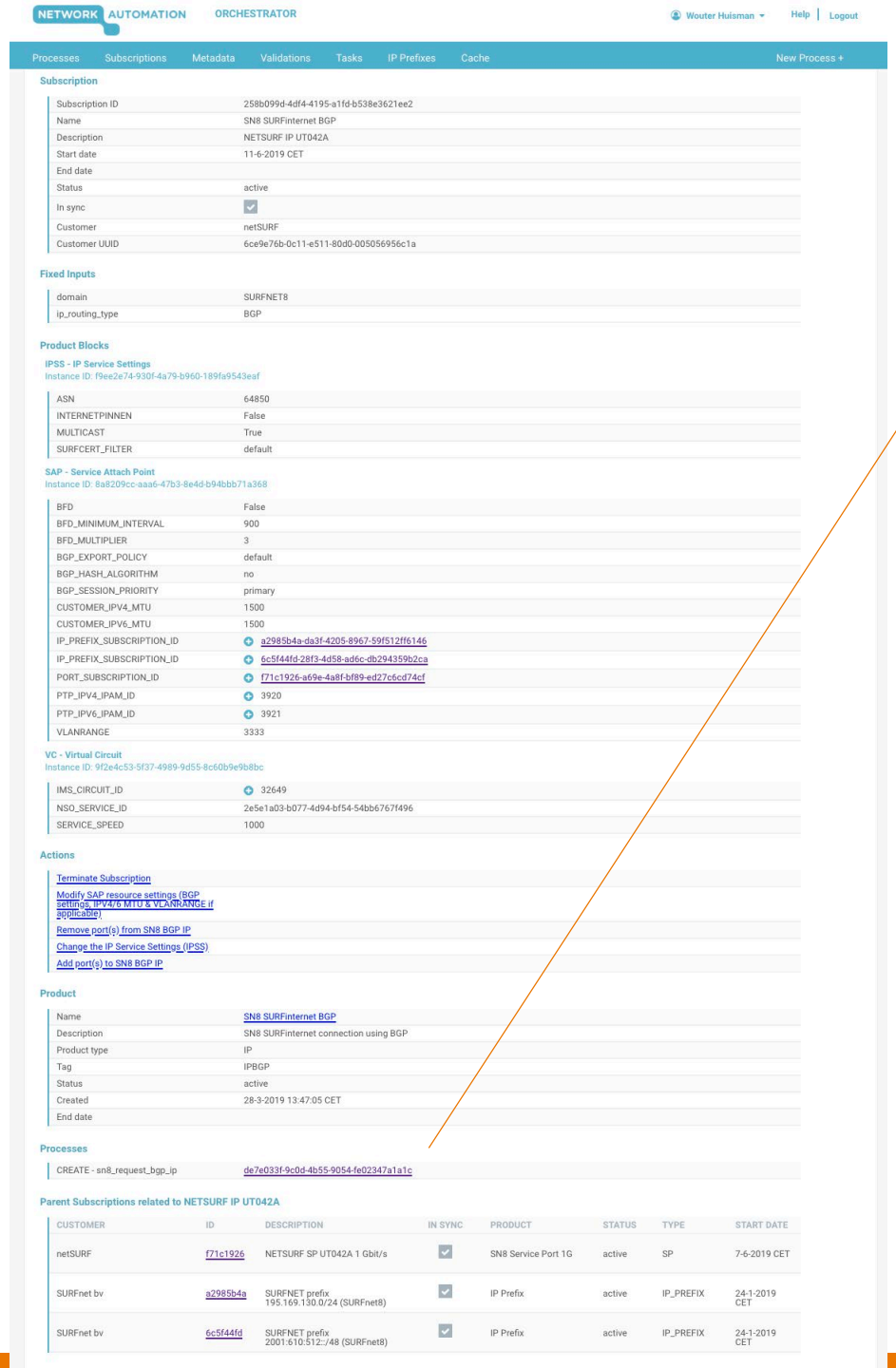
INTERNETPINNEN_PREFIX_SUBSCRIPTION_ID
INTERNETPINNEN_PREFIXES
IP_PREFIXES [
{

Prepare BGP SAP
settings with defaults
Success
11-6-2019 17:29:17 CET

SAP1


```
{
  "bfd": {
    "enabled": false,
    "minimum_interval": 900,
    "multiplier": 3
  },
  "bgp_export_policy": "default",
  "bgp_hash_algorithm": "no",
  "bgp_session_priority": "primary",
  "customer_ipv4_mtu": 1500,
  "customer_ipv6_mtu": 1500,
  "ip_prefix_subscriptions": "a2985b4a-da3f-4205-8967-59f512ff6146,6c5f44fd-28f3-4d58-ad6c-db294359b2ca",
  "ipv4_prefixlen": "/31",
  "port_subscription_id": "f71c1926-a69e-4a8f-bf89-ed27c6cd74cf",
  "service_port": [
    {
      "port_mode": "tagged",
      "subscription_id": "f71c1926-a69e-4a8f-bf89-ed27c6cd74cf",
      "vlan": "3333"
    }
  ],
  "vlanrange": "3333"
}
```

# WORKFLOW PROCESS OF CREATION




Update name of IMS service
Success
11-6-2019 17:29:52 CET

IMS_CIRCUIT_NAME UT042A_IP_BGP_NETSURF_258b099d

Update subscription
Success
11-6-2019 17:29:53 CET

Load SAP instance IDs.
Success
11-6-2019 17:29:53 CET

SAP1

```
{
  "instance_id": "8a8209cc-aaa6-47b3-8e4d-b94bbb71a368"
}
```

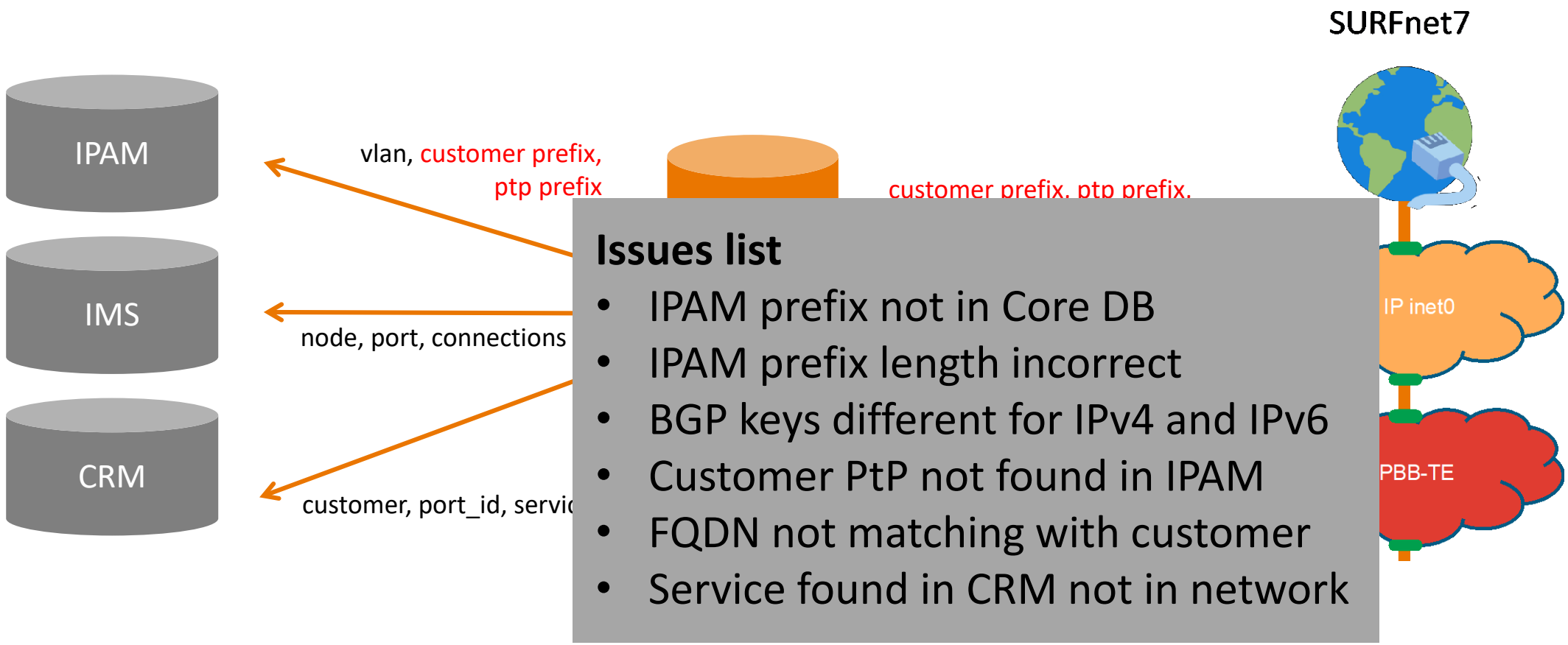
Update the description field of PTP prefixes
Success
11-6-2019 17:29:53 CET

Deploy the SURFNET8 BGP service model
Success
11-6-2019 17:29:59 CET

NSO_PAYLOAD

```
{
  "l3internet:l3internet-bgp": {
    "cust_asn": "64850",
    "customer_name": "NETSURF",
    "endpoints": [
      {
        "bfd_enabled": false,
        "bgp_session_priority": "primary",
        "device": "ut042a-jnx-02",
        "export_policy": "default",
        "instance_id": "8a8209cc-aaa6-47b3-8e4d-b94bbb71a368",
        "interface": "xe-0/1/5",
        "ipv4_address": "145.145.2.218/31",
        "ipv4_mtu": 1500,
        "ipv4_prefixes": [
          {
            "prefix": "195.169.130.0/24"
          }
        ],
        "ipv4_remote_address": "145.145.2.219",
        "ipv6_address": "2001:610:fc7:0:145:145:2:218/127",
        "ipv6_mtu": 1500,
        "ipv6_prefixes": [
          {
            "prefix": "2001:610:512::/48"
          }
        ],
        "ipv6_remote_address": "2001:610:fc7:0:145:145:2:219",
        "speed": "1000",
        "vlan_id": "3333"
      }
    ],
    "name": "2e5e1a03-b077-4d94-bf54-54bb6767f496",
    "pinnen_prefixes": [],
    "subscription_id": "258b099d-4df4-4195-a1fd-b538e3621ee2",
    "surfcert_filter": "default"
  }
}
```

# Import and validate brownfield SURFnet7 network services




**Issues list**

- IPAM prefix not in Core DB
- IPAM prefix length incorrect
- BGP keys different for IPv4 and IPv6
- Customer PtP not found in IPAM
- FQDN not matching with customer
- Service found in CRM not in network

# Do we foil?

- NSO service model development is rather quick
- Developing workflows is slower
- Dependent on other less reliable applications
- It is complex, a lot of testing needed
- Need to appreciate the rigid procedures, no shortcuts for quick fixes

# Lessons Learned

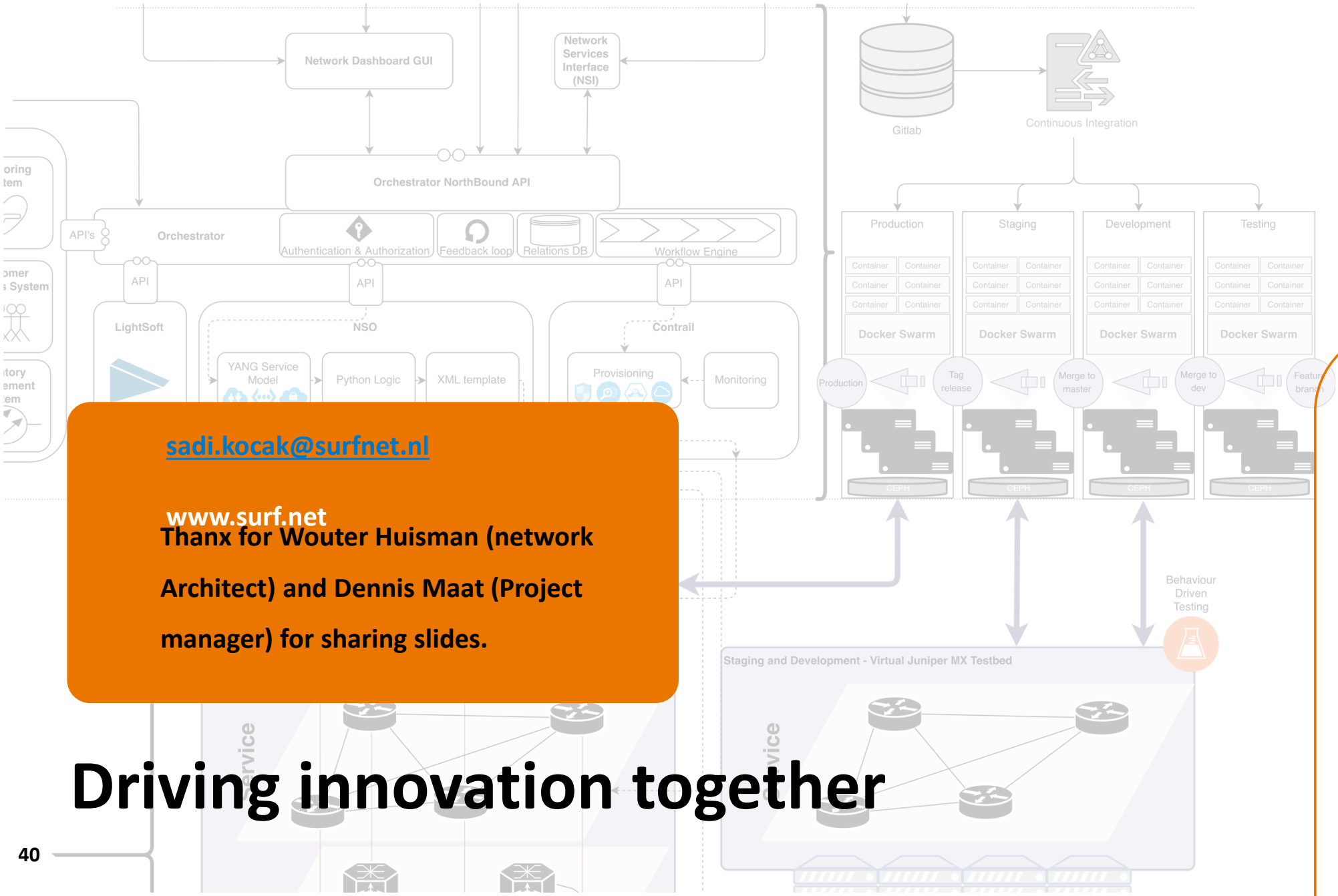
- Managing expections of all interested stakeholders.
- Challenges for convinving old book networkengineers for pushing to use automation workflows. Feeling of engineers during migration is that workload is increasing. They need to be convinced that after a while they will get the best out of it.
- Not underestimating the amount of time which is needed to develop, test, validate and getting in production
- Good release management for service models and workflows is required
- Breaking all over resistence against automation. (fear for job loses, different job tasks)

# Last Status

- We are in a mid stage with two service layer networks.
- We have the challenge to keep both networks up and running with the same team.
- Optical layer migration is almost fulfilled, plannig for finishing this is next year.
- Most important rule for using automation is that everybody needs to deal with automation flows, so the principle in minds should be all or nothing. Weakest chain is weakening whole proces.

# Last slide

- Thats All
- Questions




sadi.kocak@surfnet.nl

www.surf.net

Thanx for Wouter Huisman (network Architect) and Dennis Maat (Project manager) for sharing slides.

# Driving innovation together