Deployment automation using SoT tool

Ulsbold Enkhtaivan

Mobicom corporation

2025-12-10

Agenda

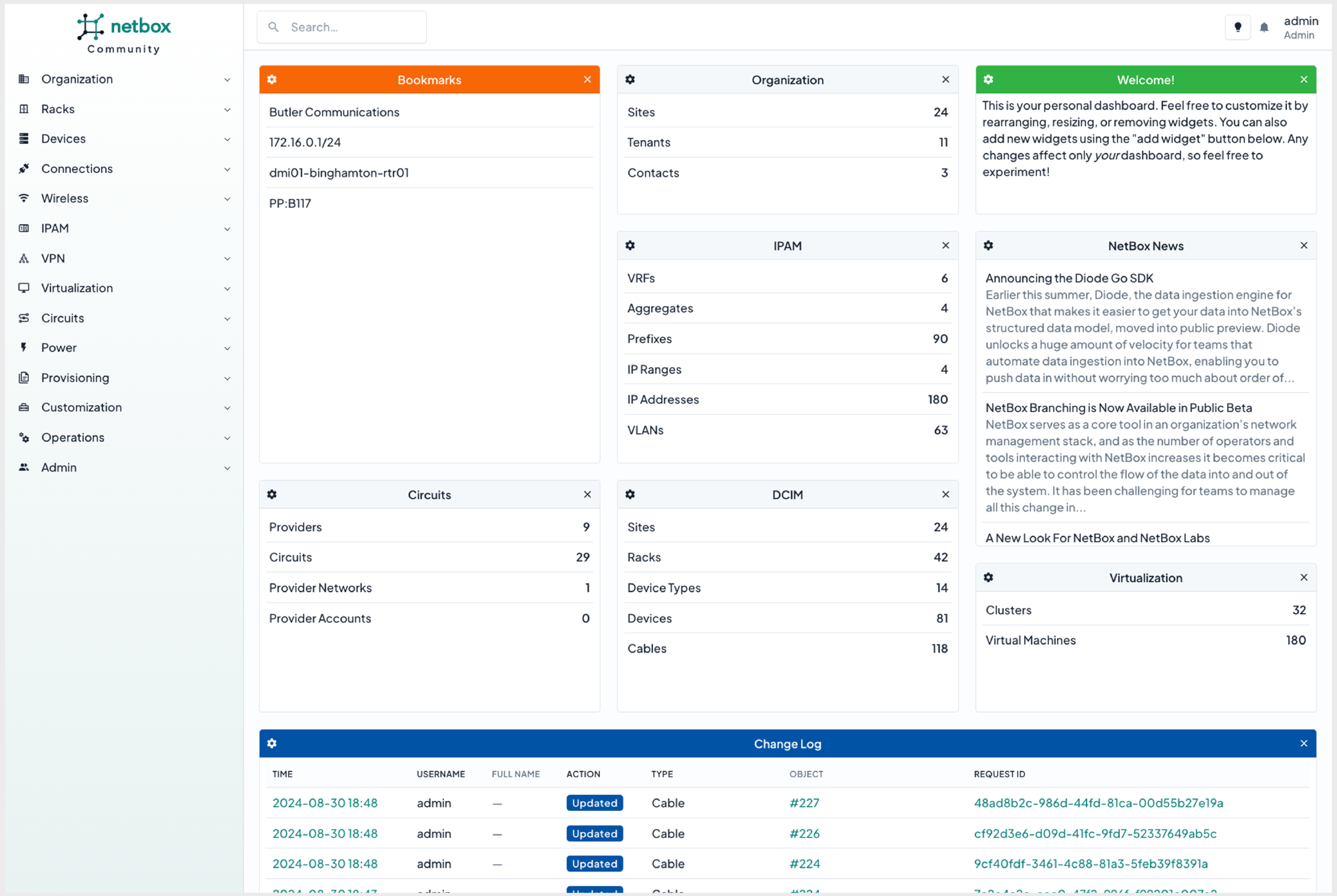
- Intro
- About SoT and netbox
- High level building blocks
- Detail of process
- Configuring online network
- Conclusion

Intro

- As a mobile operator, mobile backhaul network is critical element for connecting Radio Access to Mobile core
- In the 4G and 5G era network rollout is important process of expanding their network.
- We tried to some ZTP like automation using FOSS tools and it was helped network roll out process.

About SoT and Netbox

- Netbox is network documentation tools that fit for any network.
- Rich API integration and features became a powerful Single Source of Truth.



Bookmarks

Butler Communications
172.16.0.1/24
dmi01-binghamton-rtr01
PP:B117

Organization

Sites	24
Tenants	11
Contacts	3

Welcome!

This is your personal dashboard. Feel free to customize it by rearranging, resizing, or removing widgets. You can also add new widgets using the "add widget" button below. Any changes affect only *your* dashboard, so feel free to experiment!

IPAM

VRFs	6
Aggregates	4
Prefixes	90
IP Ranges	4
IP Addresses	180
VLANs	63

NetBox News

Announcing the Diode Go SDK
Earlier this summer, Diode, the data ingestion engine for NetBox that makes it easier to get your data into NetBox's structured data model, moved into public preview. Diode unlocks a huge amount of velocity for teams that automate data ingestion into NetBox, enabling you to push data in without worrying too much about order of...

NetBox Branching is Now Available in Public Beta
NetBox serves as a core tool in an organization's network management stack, and as the number of operators and tools interacting with NetBox increases it becomes critical to be able to control the flow of the data into and out of the system. It has been challenging for teams to manage all this change in...

A New Look For NetBox and NetBox Labs

Circuits

Providers	9
Circuits	29
Provider Networks	1
Provider Accounts	0

DCIM

Sites	24
Racks	42
Device Types	14
Devices	81
Cables	118

Virtualization

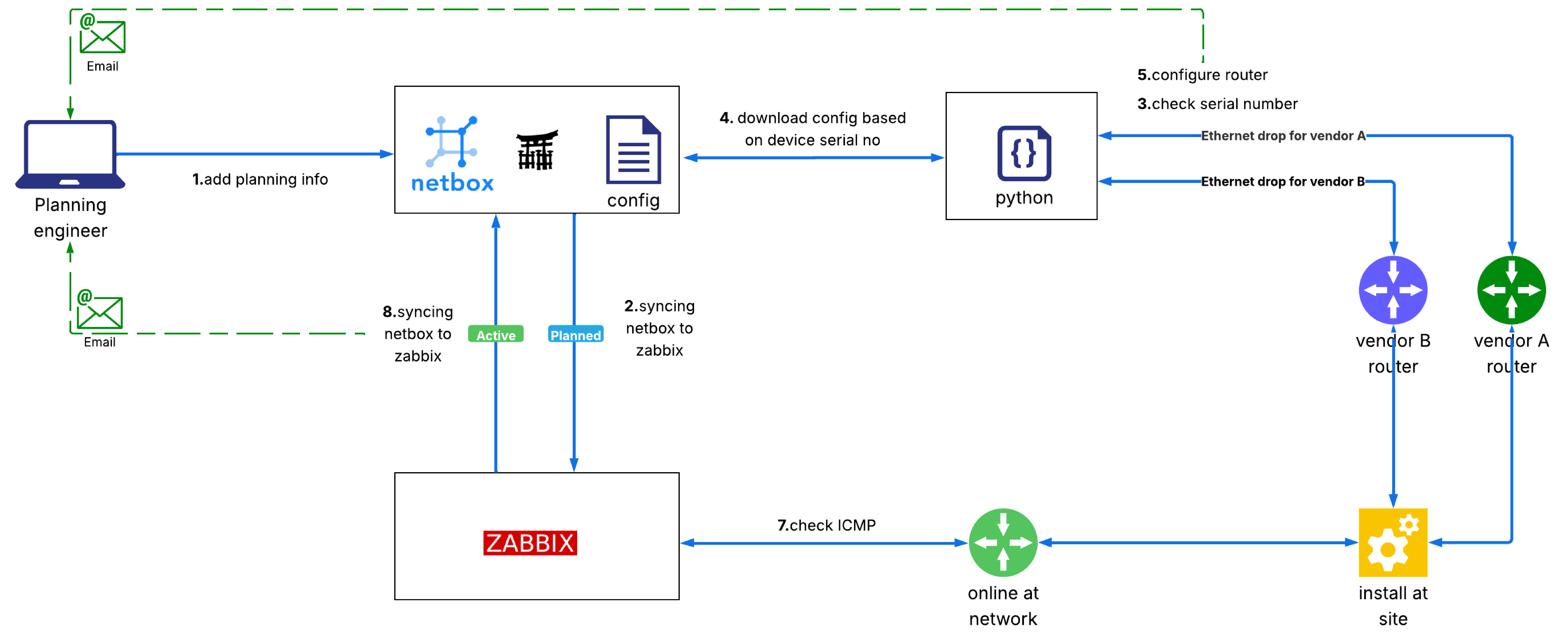
Clusters	32
Virtual Machines	180

Change Log

TIME	USERNAME	FULL NAME	ACTION	TYPE	OBJECT	REQUEST ID
2024-08-30 18:48	admin	—	Updated	Cable	#227	48ad8b2c-986d-44fd-81ca-00d55b27e19a
2024-08-30 18:48	admin	—	Updated	Cable	#226	cf92d3e6-d09d-41fc-9fd7-52337649ab5c
2024-08-30 18:48	admin	—	Updated	Cable	#224	9cf40fdf-3461-4c88-81a3-5feb39f8391a

https://netboxlabs.com/docs/netbox/

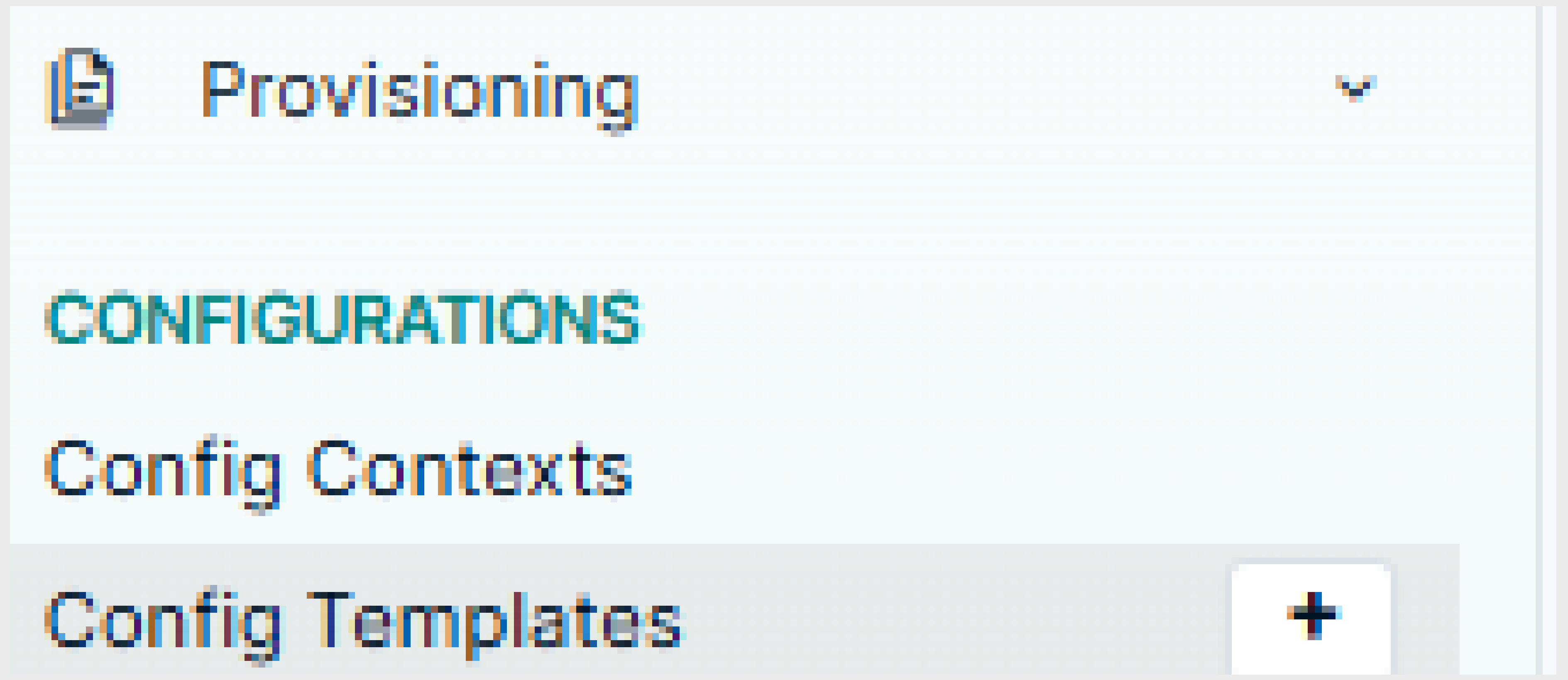
High level building blocks



Detail of process

step 1

- Initially we just registering our devices, services and ip address on NETBOX
- Then digging some features that **config template**,
- This feature easily generate config using Jinja from Json input
- Also integrated with Zabbix



netbox

config

Config template Jinja

step 1 (vrf example)

Vendor-A

```
{% set vrf_list = [] %}
{% for iface in device.interfaces.all() if iface.vrf %}
  {% if iface.vrf not in vrf_list %}
    {% set _ = vrf_list.append(iface.vrf) %}
  {% endif %}
{% endfor %}

{% for vrf in vrf_list %}
ip vrf {{ vrf.name }}
  rd {{ vrf.rd }}
  mpls label mode per-vrf
  address-family ipv4
{%   for rt in vrf.import_targets.all() %}
    route-target import {{ rt.name }}
{%   endfor %}
{%   for rt in vrf.export_targets.all() %}
    route-target export {{ rt.name }}
{%   endfor %}
```

```
ip vrf VPN-A
  rd 300:300
  mpls label mode per-vrf
  address-family ipv4
    route-target import 300:301
    route-target export 300:300
$
ip vrf VPN-B
  rd 400:400
  mpls label mode per-vrf
  address-family ipv4
    route-target import 400:400
    route-target export 400:400
$
ip vrf VPN-c
  rd 500:500
  mpls label mode per-vrf
  address-family ipv4
    route-target import 500:501
    route-target export 500:500
$
```

Vendor-B

```
{% for interface in device.interfaces.all() -%}
{% if interface.vrf -%}
ip vpn-instance {{ interface.vrf.name }}
 ipv4-family
  route-distinguisher {{ interface.vrf.rd }}
  apply-label per-instance
  {% for rt in interface.vrf.import_targets.all() %}
  vpn-target {{ rt.name }} import-extcommunity
  {% endfor %}
  {% for rt in interface.vrf.export_targets.all() %}
  vpn-target {{ rt.name }} export-extcommunity
  {% endfor %}
{% endif %}
{%- endfor %}
```

```
ip vpn-instance VPN-A
 ipv4-family
  route-distinguisher 300:300
  apply-label per-instance
    vpn-target 300:301 import-extcommunity
    vpn-target 300:300 export-extcommunity

ip vpn-instance VPN-B
 ipv4-family
  route-distinguisher 400:400
  apply-label per-instance
    vpn-target 400:400 import-extcommunity
    vpn-target 400:400 export-extcommunity

ip vpn-instance VPN-C
 ipv4-family
  route-distinguisher 500:500
  apply-label per-instance
    vpn-target 500:501 import-extcommunity
    vpn-target 500:500 export-extcommunity
```

netbox

config

Config template Jinja

step 1 (BGP example)

Vendor-A

```
router bgp {{ device.custom_field_data.asn }}
  bgp router-id {{ device.primary_ip.address.ip }}
  neighbor {{ device.custom_field_data.bgp_peer }} remote-as {{ device.custom_field_data.asn }}
  neighbor {{ device.custom_field_data.bgp_peer }} activate
  neighbor {{ device.custom_field_data.bgp_peer }} send-label
  neighbor {{ device.custom_field_data.bgp_peer }} update-source loopback0
  address-family vpnv4
    neighbor {{ device.custom_field_data.bgp_peer }} activate
  $
{% for interface in device.interfaces.all() if interface.vrf %}
 address-family ipv4 vrf {{ interface.vrf.name }}
  redistribute connected
  $
```

```
router bgp 65530
  bgp router-id 10.10.10.10
  neighbor 10.10.10.1 remote-as 65530
  neighbor 10.10.10.1 activate
  neighbor 10.10.10.1 send-label
  neighbor 10.10.10.1 update-source loopback0
  address-family vpnv4
    neighbor 10.10.10.1 activate
   $
  address-family ipv4 vrf VPN-A
   redistribute connected
  $
  address-family ipv4 vrf VPN-B
   redistribute connected
  $
  address-family ipv4 vrf VPN-C
   redistribute connected
```

Vendor-B

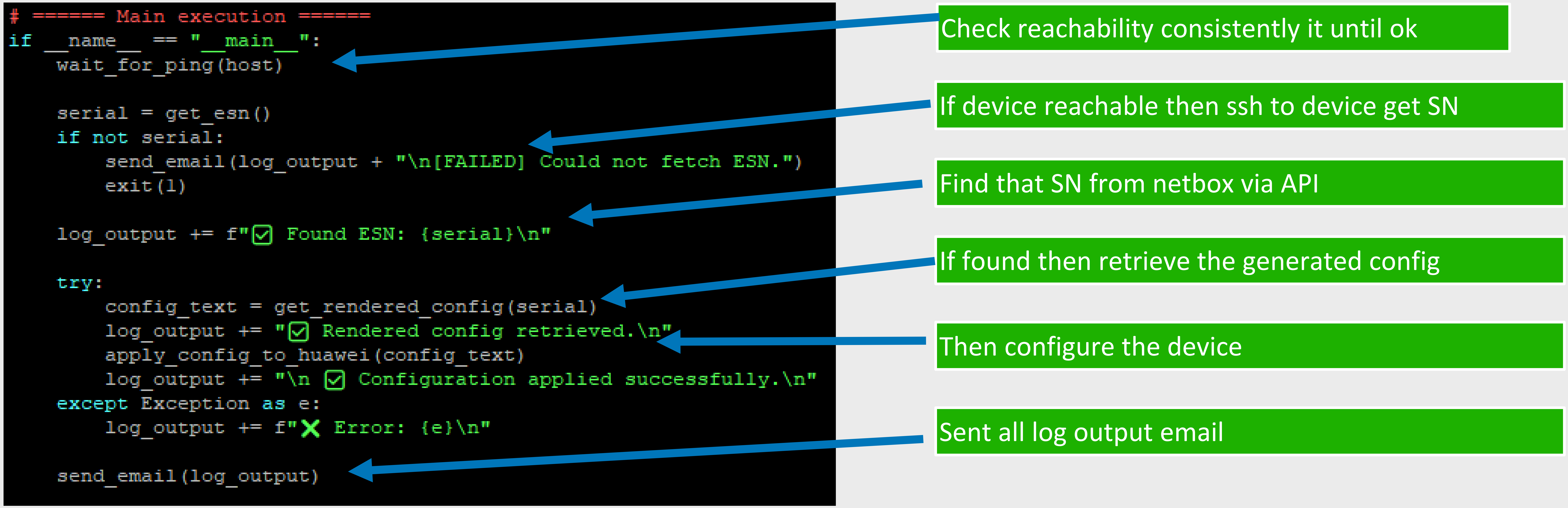
```
bgp {{ device.custom_field_data.asn }}
 router-id {{ device.primary_ip.address.ip }}
 peer {{ device.custom_field_data.bgp_peer }} as-number {{ device.custom_field_data.asn }}
 peer {{ device.custom_field_data.bgp_peer }} connect-interface LoopBack0
 #
 ipv4-family unicast
  peer {{ device.custom_field_data.bgp_peer }} enable
 #
 ipv4-family labeled-unicast
  peer {{ device.custom_field_data.bgp_peer }} enable
 #
 ipv4-family vpnv4
  undo policy vpn-target
  peer {{ device.custom_field_data.bgp_peer }} enable
 #
{% for interface in device.interfaces.all() if interface.vrf %}
 ipv4-family vpn-instance {{ interface.vrf.name }}
  import-route direct
{% endfor %}
```

```
bgp 65530
 router-id 10.10.10.20
 peer 10.10.10.1 as-number 65530
 peer 10.10.10.1 connect-interface LoopBack0
 #
 ipv4-family unicast
  peer 10.10.10.1 enable
 #
 ipv4-family labeled-unicast
  peer 10.10.10.1 enable
 #
 ipv4-family vpnv4
  peer 10.10.10.1 enable
 #
 ipv4-family vpn-instance VPN-A
  import-route direct
 #
 ipv4-family vpn-instance VPN-B
  import-route direct
 #
 ipv4-family vpn-instance VPN-C
  import-route direct
```

Configure device

Step 3, 4, 5

- Our engineer developed in-house python code that help to retrieve config from netbox then configure network devices based on Serial number
- Paramiko based python script accessing to device via SSH then push config file to device



```
# ====== Main execution ======
if __name__ == "__main__":
    wait_for_ping(host)

    serial = get_esn()
    if not serial:
        send_email(log_output + "\n[FAILED] Could not fetch ESN.")
        exit(1)

    log_output += f"✅ Found ESN: {serial}\n"

    try:
        config_text = get_rendered_config(serial)
        log_output += "✅ Rendered config retrieved.\n"
        apply_config_to_huawei(config_text)
        log_output += "\n ✅ Configuration applied successfully.\n"
    except Exception as e:
        log_output += f"❌ Error: {e}\n"

    send_email(log_output)
```

Check reachability consistently it until ok

If device reachable then ssh to device get SN

Find that SN from netbox via API

If found then retrieve the generated config

Then configure the device

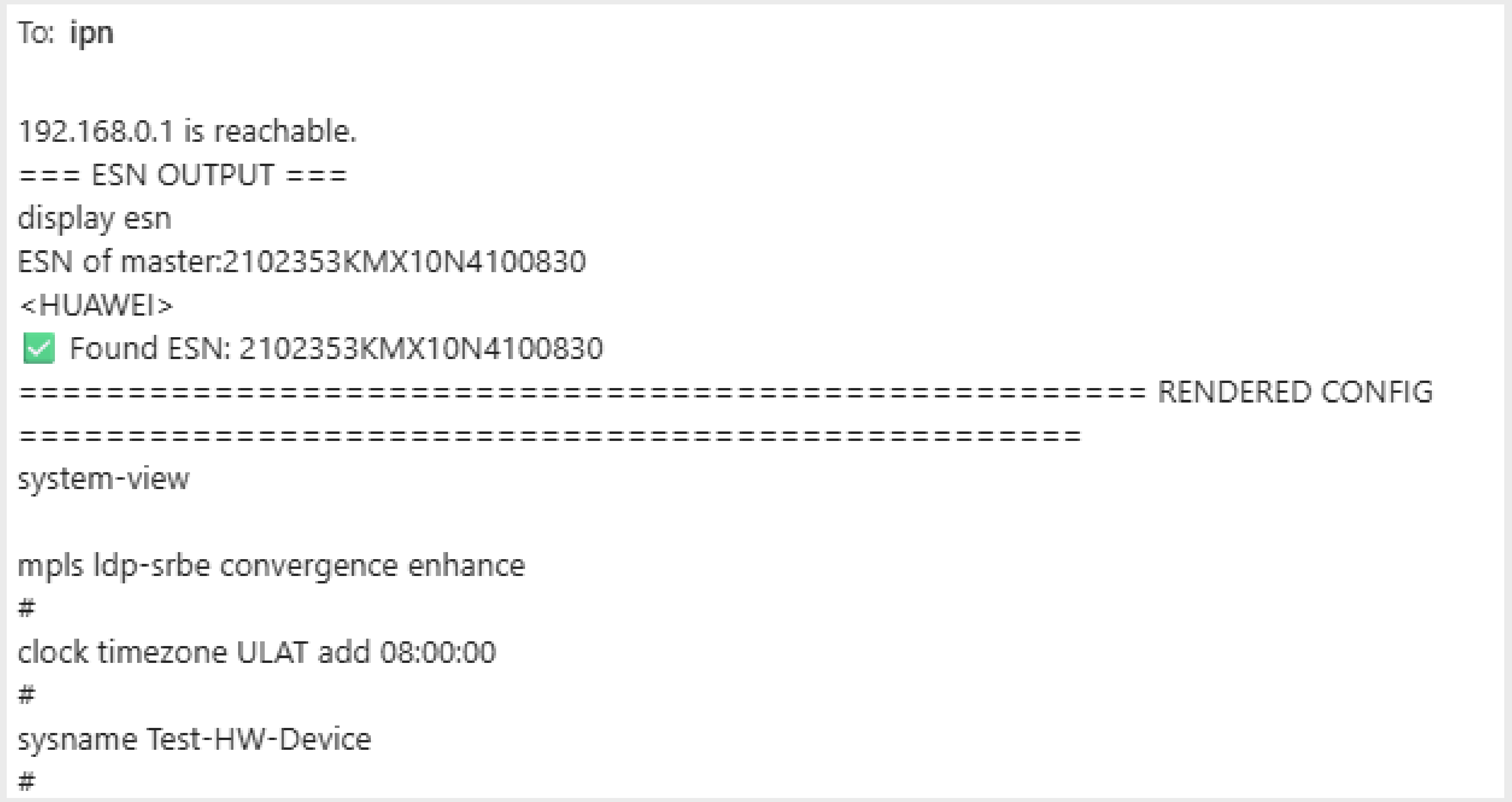
Sent all log output email

Configure device

Step 3, 4, 5

- Email notification

Successful case

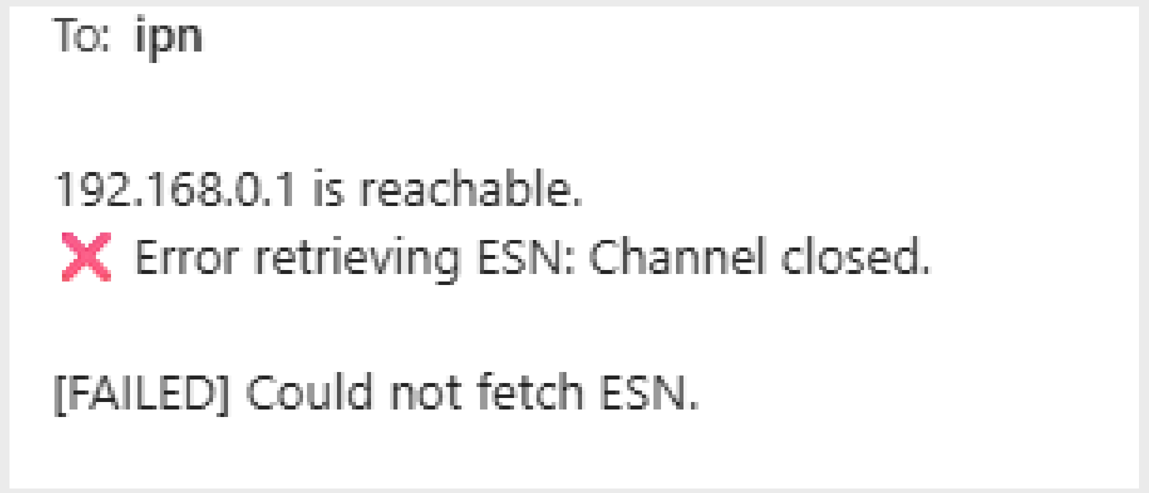


```
To: ipn

192.168.0.1 is reachable.
=== ESN OUTPUT ===
display esn
ESN of master:2102353KMX10N4100830
<HUAWEI>
✅ Found ESN: 2102353KMX10N4100830
================================================== RENDERED CONFIG
==================================================
system-view

mpls ldp-srbe convergence enhance
#
clock timezone ULAT add 08:00:00
#
sysname Test-HW-Device
#
```

unsuccessful case



```
To: ipn

192.168.0.1 is reachable.
❌ Error retrieving ESN: Channel closed.

[FAILED] Could not fetch ESN.
```

Configure device

Step 3, 4, 5

- Here is a diagram that for configuring device via their management network
- we uses vendors management ipv4 address that already configured from factory

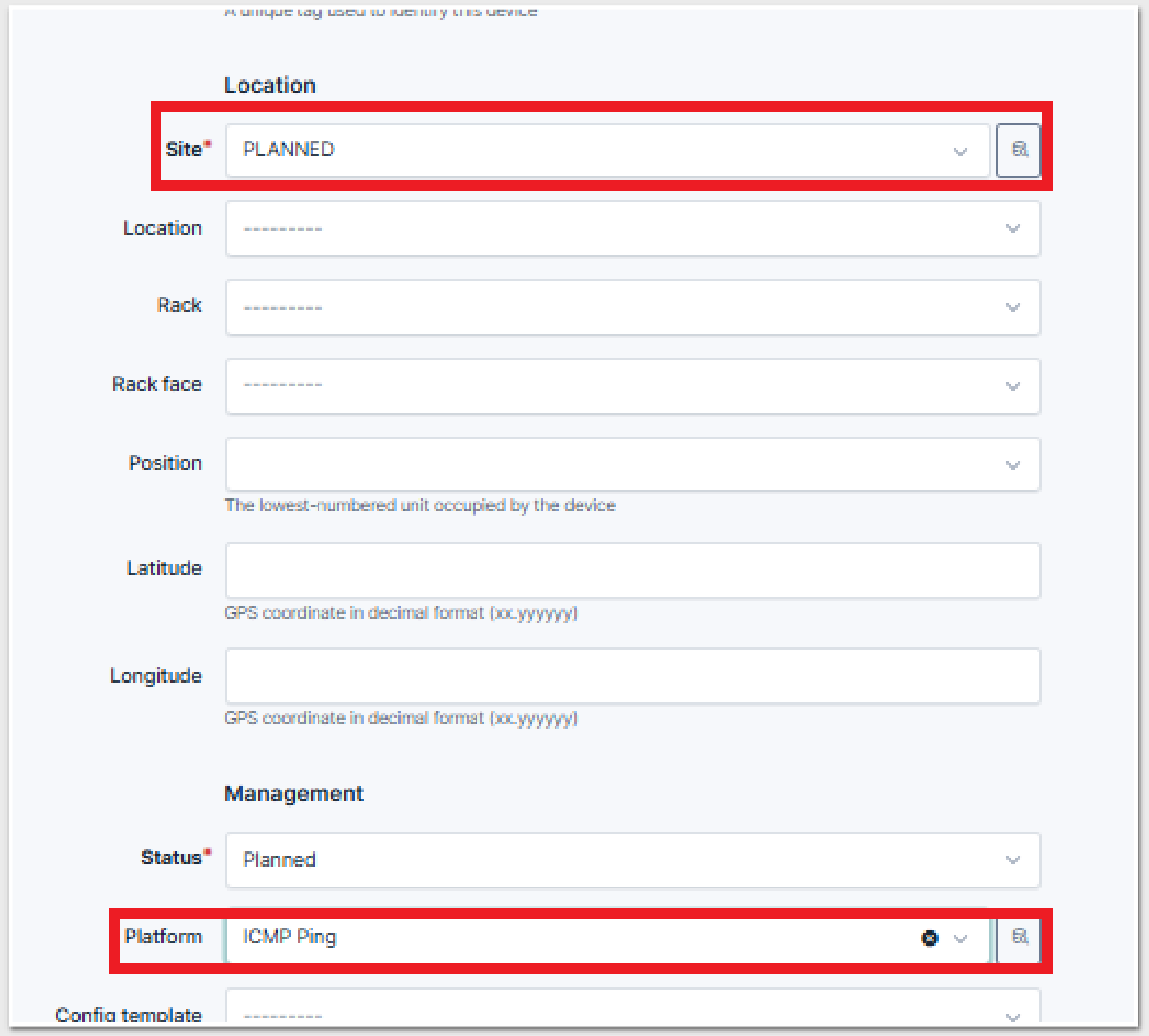
netbox
config
4. download config based on device serial no
python
Ethernet drop for vendor A
3.check serial number
5.configure router
Ethernet drop for vendor B
vendor A router
vendor B router

Zabbix integration

Step 2

- There are someone already published integration at GitHub,
- We just follow this and made little change for own environment.

https://github.com/AdrianJPT/Netbox-Zabbix-Integration



A unique tag used to identify this device

Location

Site*	PLANNED
Location	---------
Rack	---------
Rack face	---------
Position	

The lowest-numbered unit occupied by the device

Latitude	

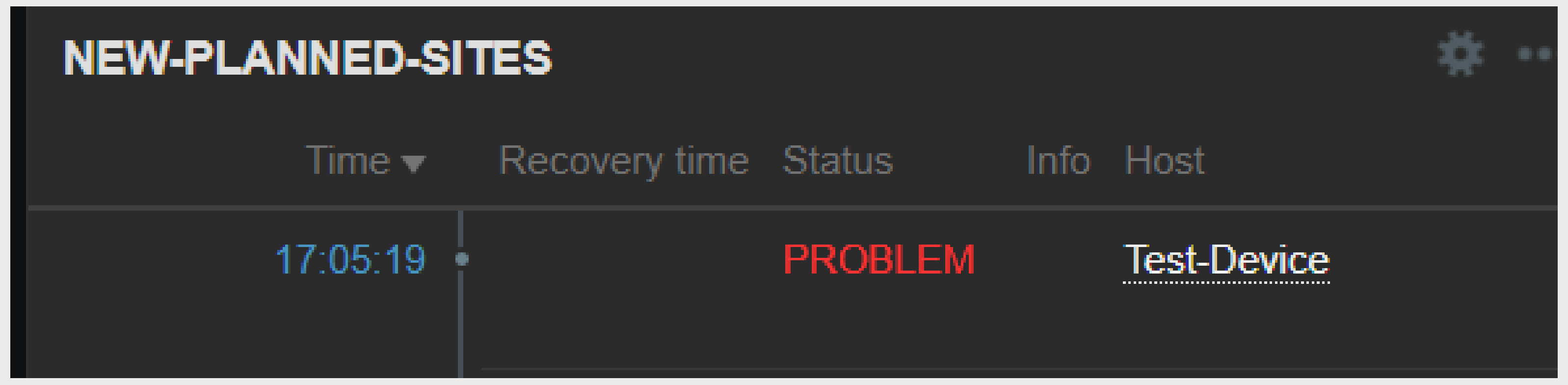
GPS coordinate in decimal format (xx.yyyyyy)

Longitude	

GPS coordinate in decimal format (xx.yyyyyy)

Management

Status*	Planned
Platform	ICMP Ping
Config template	---------



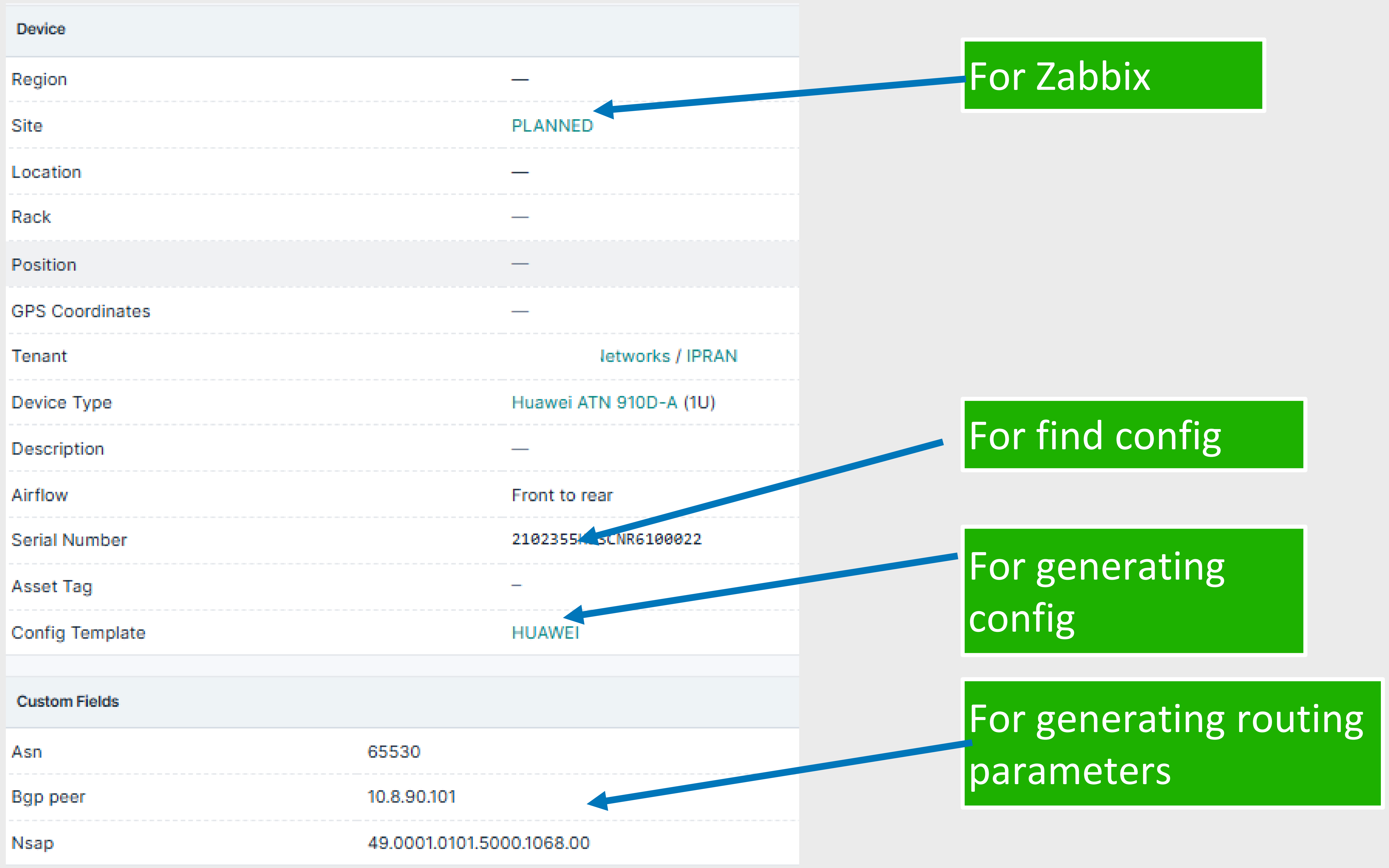
NEW-PLANNED-SITES

Time ▾	Recovery time	Status	Info	Host
17:05:19		PROBLEM		Test-Device

Back to step 1

Netbox become Source of Truth

- Now we need to carefully add our planned device on netbox.
- Due to lots of **dependency** on it.



Device	
Region	—
Site	PLANNED
Location	—
Rack	—
Position	—
GPS Coordinates	—
Tenant	letworks / IPRAN
Device Type	Huawei ATN 910D-A (1U)
Description	—
Airflow	Front to rear
Serial Number	2102355[illegible]SCNR6100022
Asset Tag	–
Config Template	HUAWEI

Custom Fields	
Asn	65530
Bgp peer	10.8.90.101
Nsap	49.0001.0101.5000.1068.00

For Zabbix

For find config

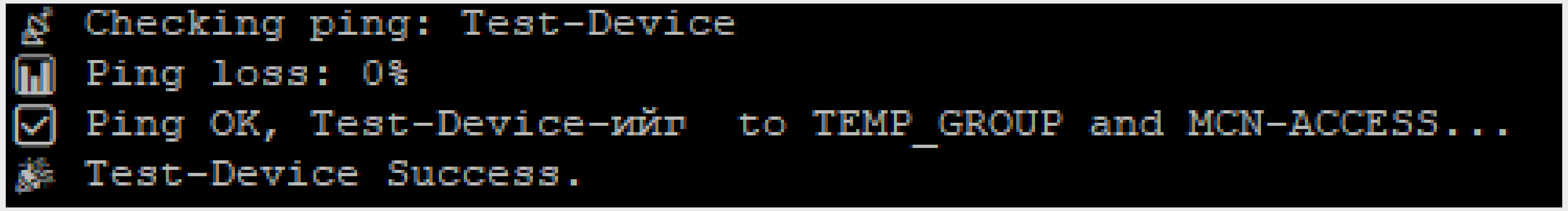
For generating config

For generating routing parameters

Zabbix to Netbox

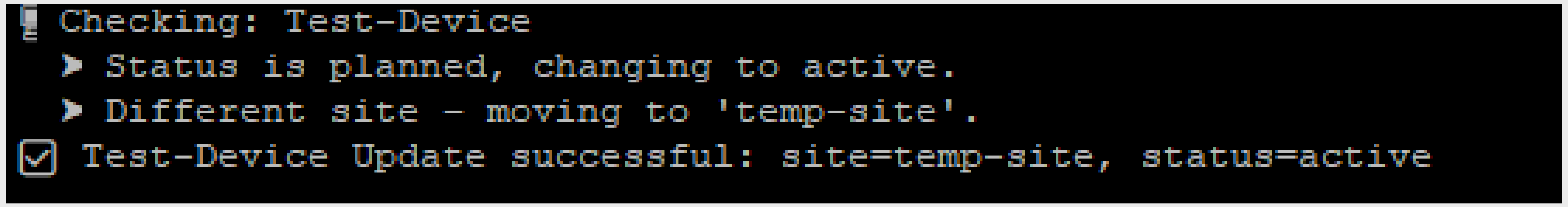
Step 7, 8

- At monitoring server checking planned device
- If ping is ok then change the group



```
🔬 Checking ping: Test-Device
📊 Ping loss: 0%
✅ Ping OK, Test-Device-ийг  to TEMP_GROUP and MCN-ACCESS...
🎉 Test-Device Success.
```

- Then also change the netbox status and site fields,

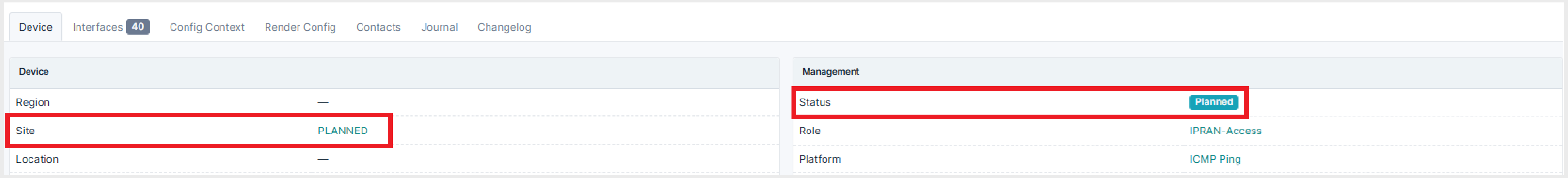


```
🔍 Checking: Test-Device
  ➤ Status is planned, changing to active.
  ➤ Different site - moving to 'temp-site'.
✅ Test-Device Update successful: site=temp-site, status=active
```

Zabbix to Netbox

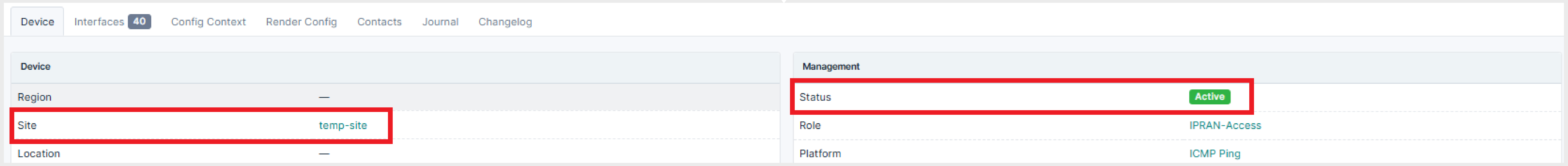
Step 7, 8

- Here is a status of netbox



Device	
Region	—
Site	PLANNED
Location	—

Management	
Status	Planned
Role	IPRAN-Access
Platform	ICMP Ping



Device	
Region	—
Site	temp-site
Location	—

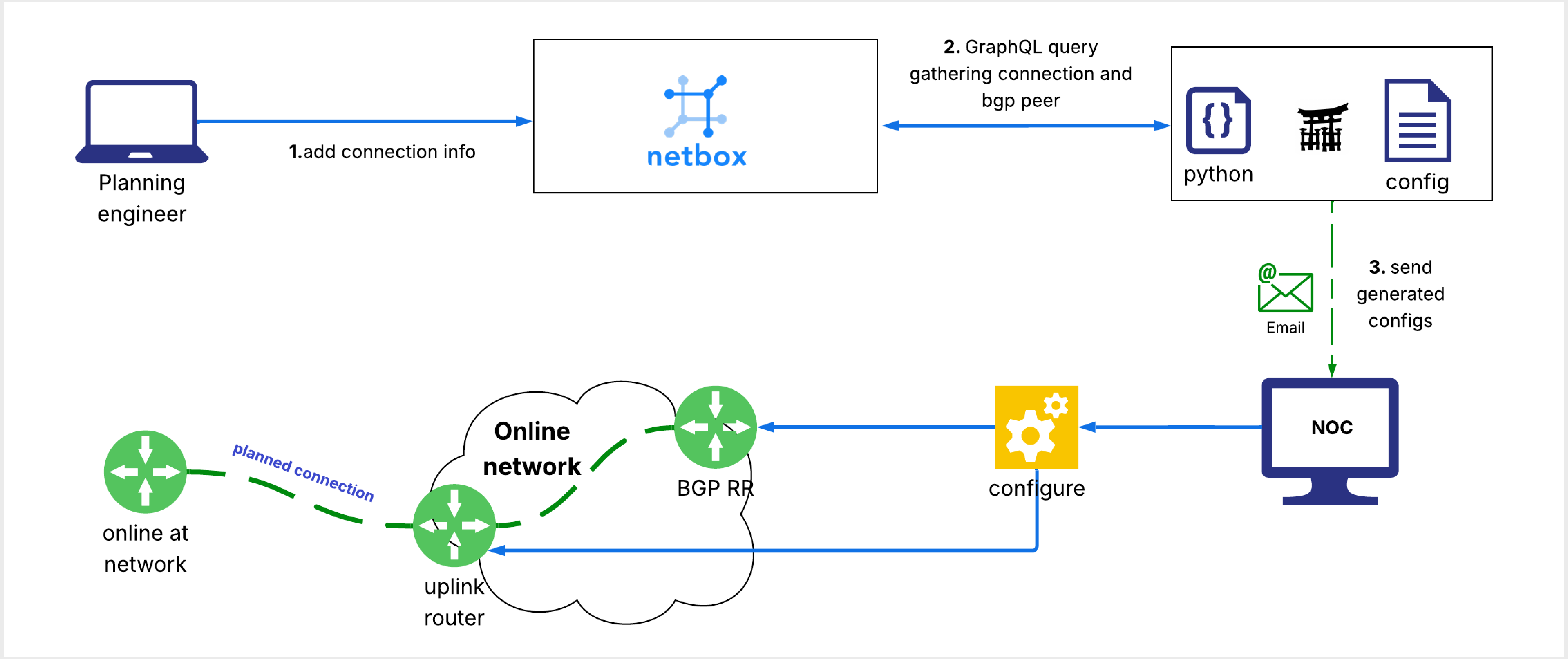
Management	
Status	Active
Role	IPRAN-Access
Platform	ICMP Ping

Configuring online network

- The previous parts were only configuring new router
- There are still need to configure online network for preparing onboard new router at mpls network

Configuring online network

- Following diagram shows how configure online network

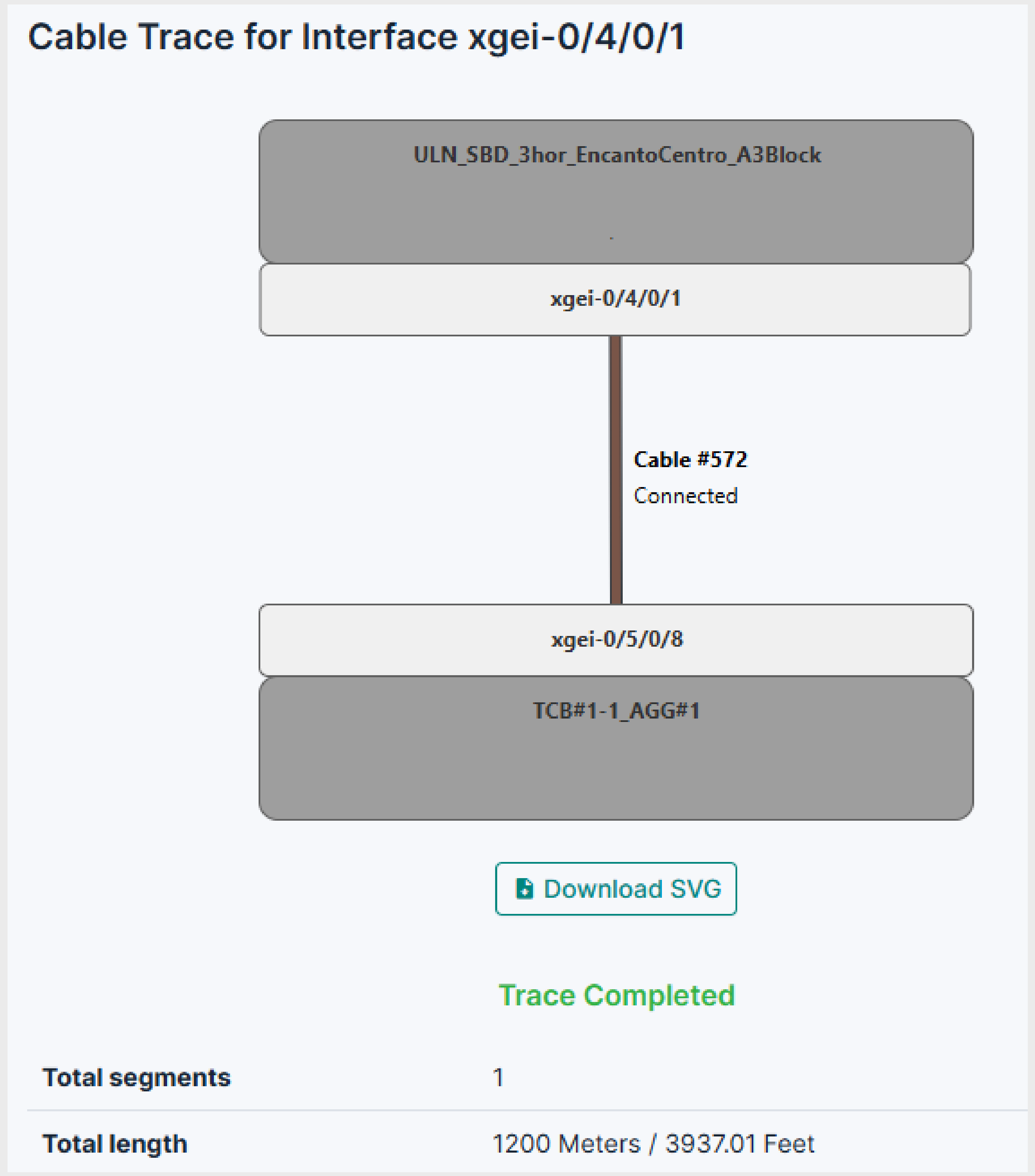


python

config

Configuring online network

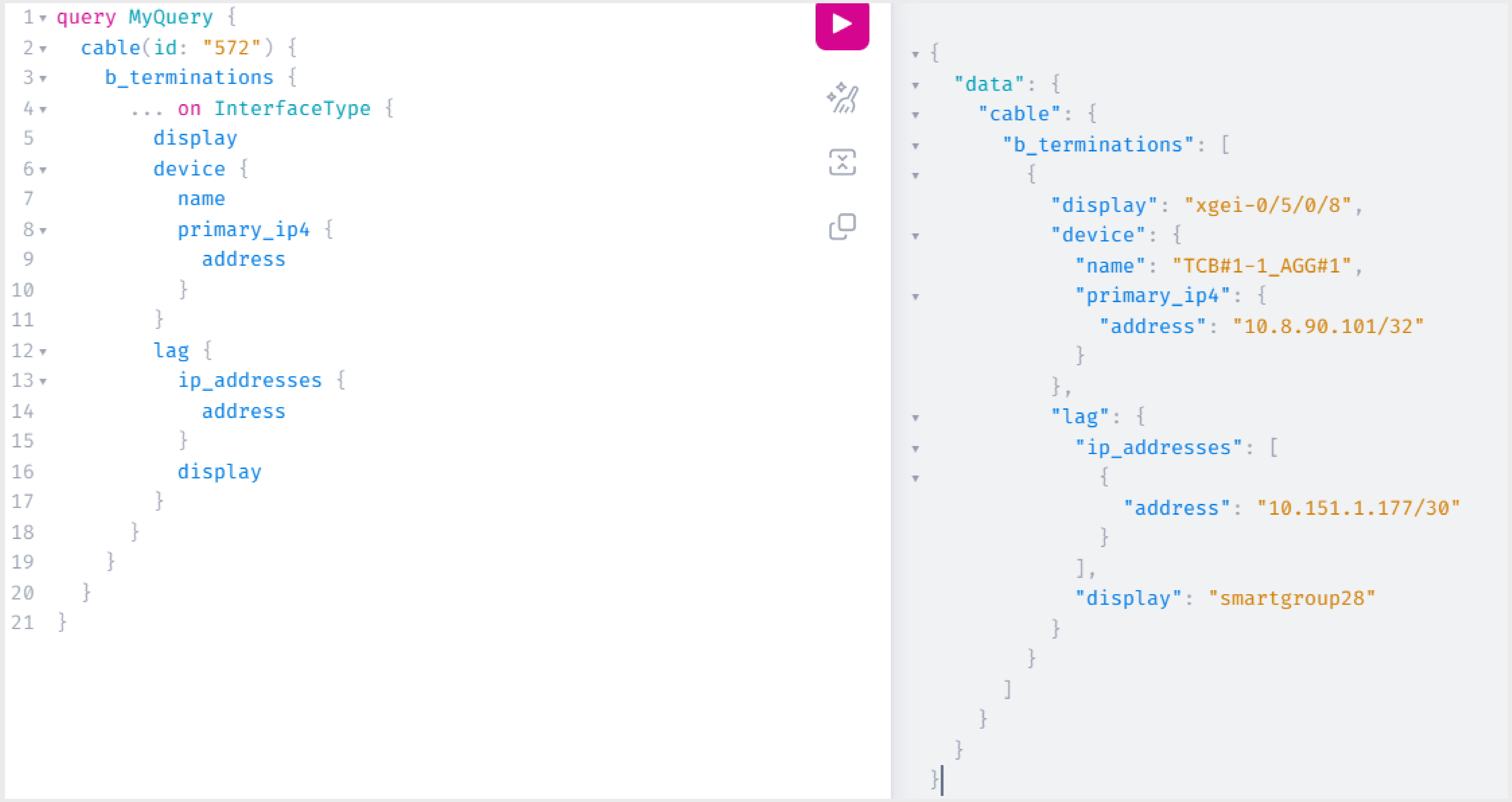
- Connection info added by engineer
- GraphQL query help to generate peer config



Cable Trace for Interface xgei-0/4/0/1

Trace Completed

Total segments	1
Total length	1200 Meters / 3937.01 Feet

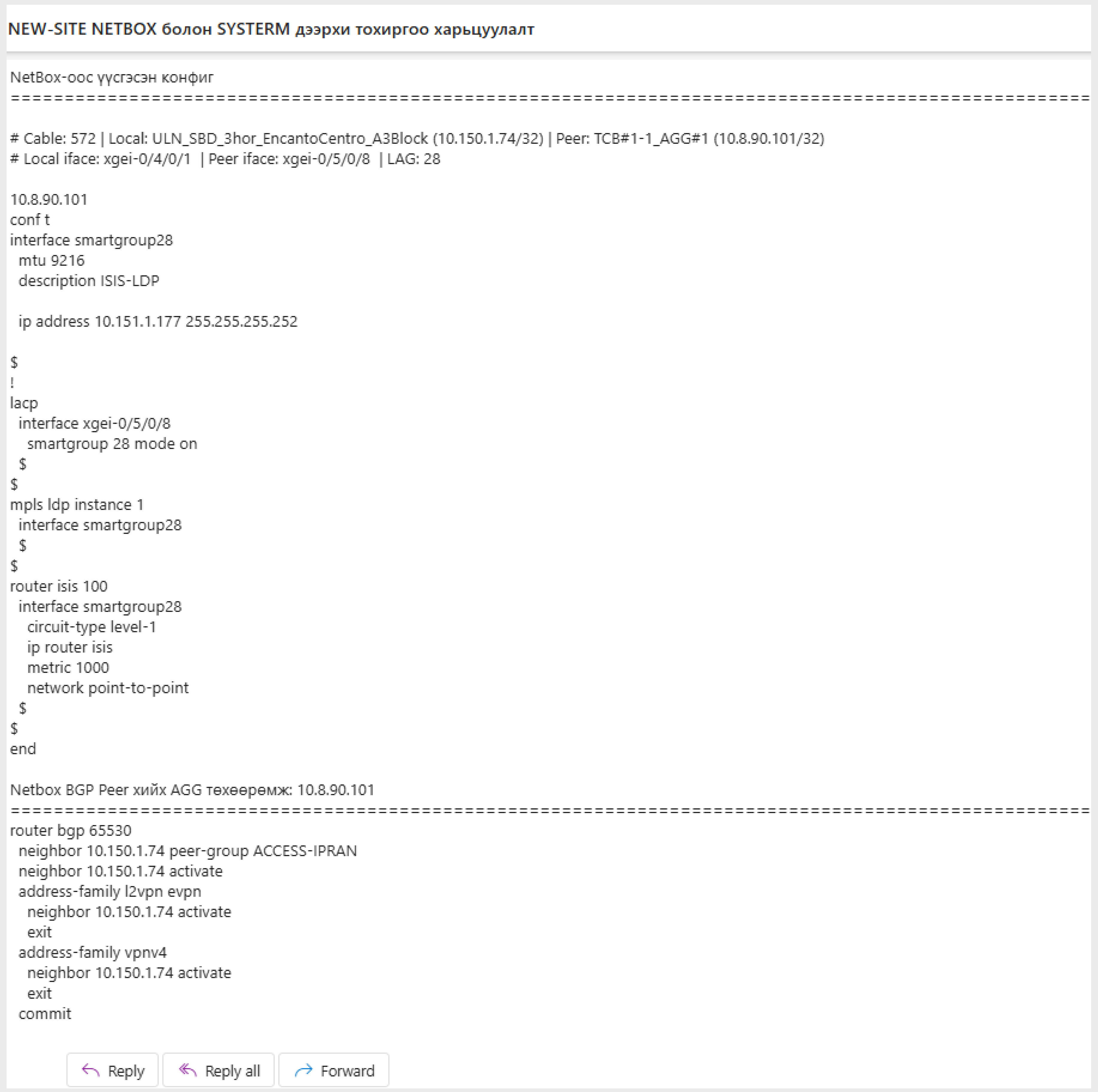


```
query MyQuery {
  cable(id: "572") {
    b_terminations {
      ... on InterfaceType {
        display
        device {
          name
          primary_ip4 {
            address
          }
        }
        lag {
          ip_addresses {
            address
          }
          display
        }
      }
    }
  }
}
```

```
{
  "data": {
    "cable": {
      "b_terminations": [
        {
          "display": "xgei-0/5/0/8",
          "device": {
            "name": "TCB#1-1_AGG#1",
            "primary_ip4": {
              "address": "10.8.90.101/32"
            }
          },
          "lag": {
            "ip_addresses": [
              {
                "address": "10.151.1.177/30"
              }
            ],
            "display": "smartgroup28"
          }
        }
      ]
    }
  }
}
```

Configuring online network

- We generated peer config by python
- based on netbox connection info then send it to our engineer
- **For safety and pre verification purpose**, manual configuration by done NOC engineer



NEW-SITE NETBOX болон SYSTERM дээрхи тохиргоо харьцуулалт

```
NetBox-оос үүсгэсэн конфиг
==============================================================================================================

# Cable: 572 | Local: ULN_SBD_3hor_EncantoCentro_A3Block (10.150.1.74/32) | Peer: TCB#1-1_AGG#1 (10.8.90.101/32)
# Local iface: xgei-0/4/0/1  | Peer iface: xgei-0/5/0/8  | LAG: 28

10.8.90.101
conf t
interface smartgroup28
  mtu 9216
  description ISIS-LDP

  ip address 10.151.1.177 255.255.255.252

$
!
lacp
  interface xgei-0/5/0/8
    smartgroup 28 mode on
  $
$
mpls ldp instance 1
  interface smartgroup28
  $
$
router isis 100
  interface smartgroup28
    circuit-type level-1
    ip router isis
    metric 1000
    network point-to-point
  $
$
end

Netbox BGP Peer хийх AGG төхөөрөмж: 10.8.90.101
==============================================================================================================
router bgp 65530
  neighbor 10.150.1.74 peer-group ACCESS-IPRAN
  neighbor 10.150.1.74 activate
  address-family l2vpn evpn
    neighbor 10.150.1.74 activate
    exit
  address-family vpnv4
    neighbor 10.150.1.74 activate
    exit
  commit
```

Finally, we got following

6 GraphQL query gathering connection and bgp peer
python
config
7 send generated configs
Email
NOC
8 manual configuration via NOC
Email
5.configure router
3.check serial number
4. download config based on device serial no
netbox
config
python
Ethernet drop for vendor A
Ethernet drop for vendor B
configure uplink
configure bgp
Planning engineer
1.add planning info
Email
11.syncing netbox to zabbix
Active
Planned
2.syncing netbox to zabbix
vendor B router
vendor A router
9 install at site
BGP RR
Online network
ZABBIX
10 check ICMP
online at network
planned connection
uplink router

Conclusion

- **Good things**
  - This work help eliminate some repetitive manual tasks,
  - Reduce some man/hours for configuring new devices by manually
  - Reducing configuration-based incidents, fat fingers
- **Comparing with ZTP there are some scalability issues**
  - can’t configure many devices same time.
  - limitation of network accessibility can’t deploy it at remote site.
- **Future follow-ups improvements**
  - develop it devices console port-based approach.

Thank You