swisscom

The Living Source of Truth

Network Automation with Infrahub and Terraform

30. September 2025, Marco Martinez

NLNOG Day 2025

Who am I?

Marco Martinez

Infrastructure Automation Architect @ Swisscom

https://www.linkedin.com/in/marcom4rtinez/

marco.martinez@swisscom.com

The Challenges in modern Network Automation

Configuration stored in Git or other files

Information split into different Domains

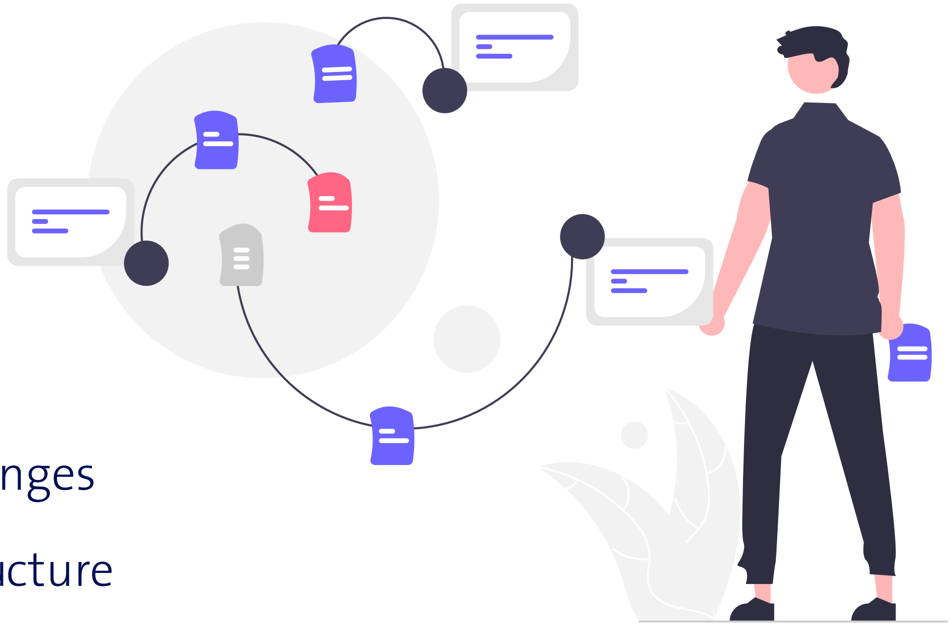
- Technical Documentation
- Organizational Requirements
- Security Policies

Requires multiple manual steps to apply configuration or changes

Out-of-sync issues between desired state and actual infrastructure

Common challenges points:

- Delay
- Drift
- Visibility
- Errors



Introducing the Living Source of Truth

Concept Idea:

- Mutable, real-time sync between source of truth and infrastructure
- Management via GUI or code

Benefits:

- Immediate consistency
- Faster feedback loops
- Fewer manual steps
- Transparency

swisscom

Let's start our Journey

What is our Goal

Manage Infrastructure in a reliable way to ensure stability

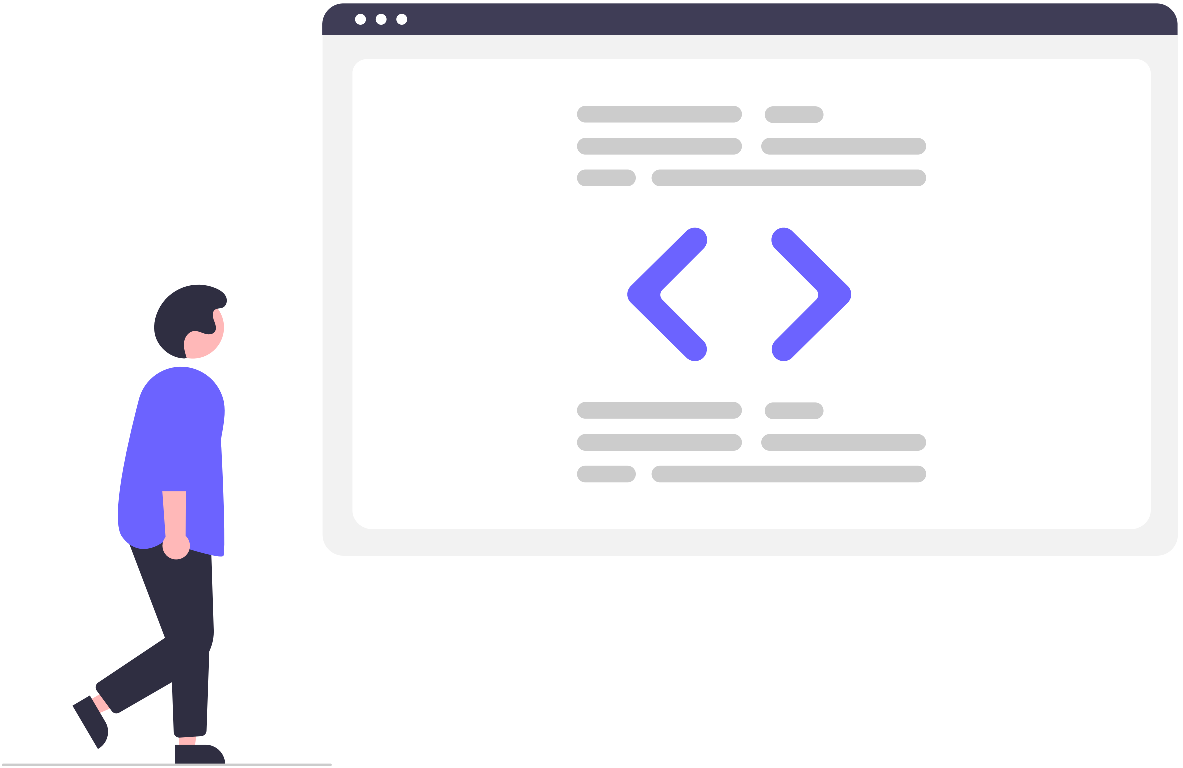
Infrastructure as Code (IaC) seems like a logical fit

+ build infrastructure based on desired state

+ modify current infrastructure to fit the new desired state

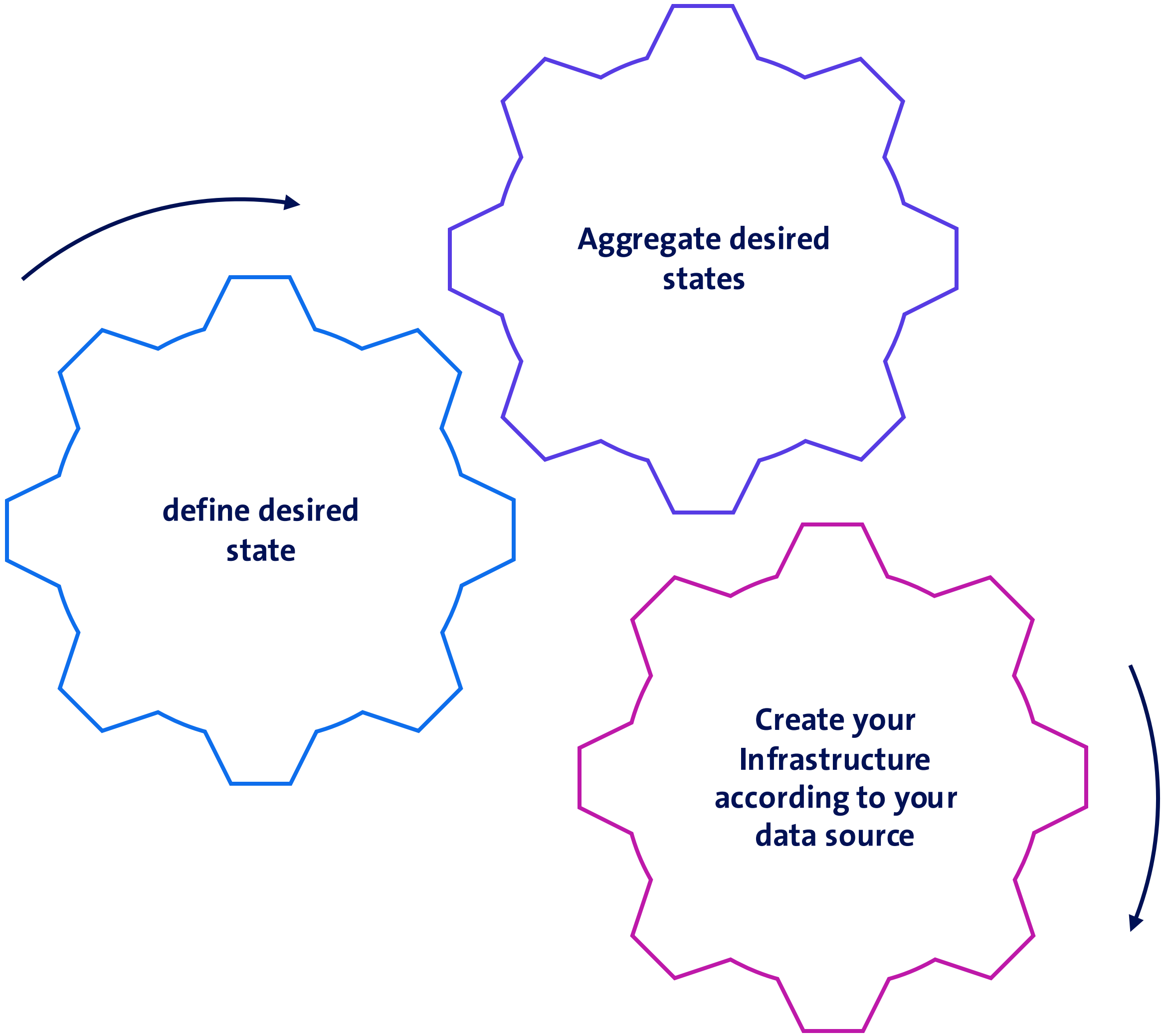
- continuity could be better with current tools (fire and forget)

- Using different tools can make for a very complex environment

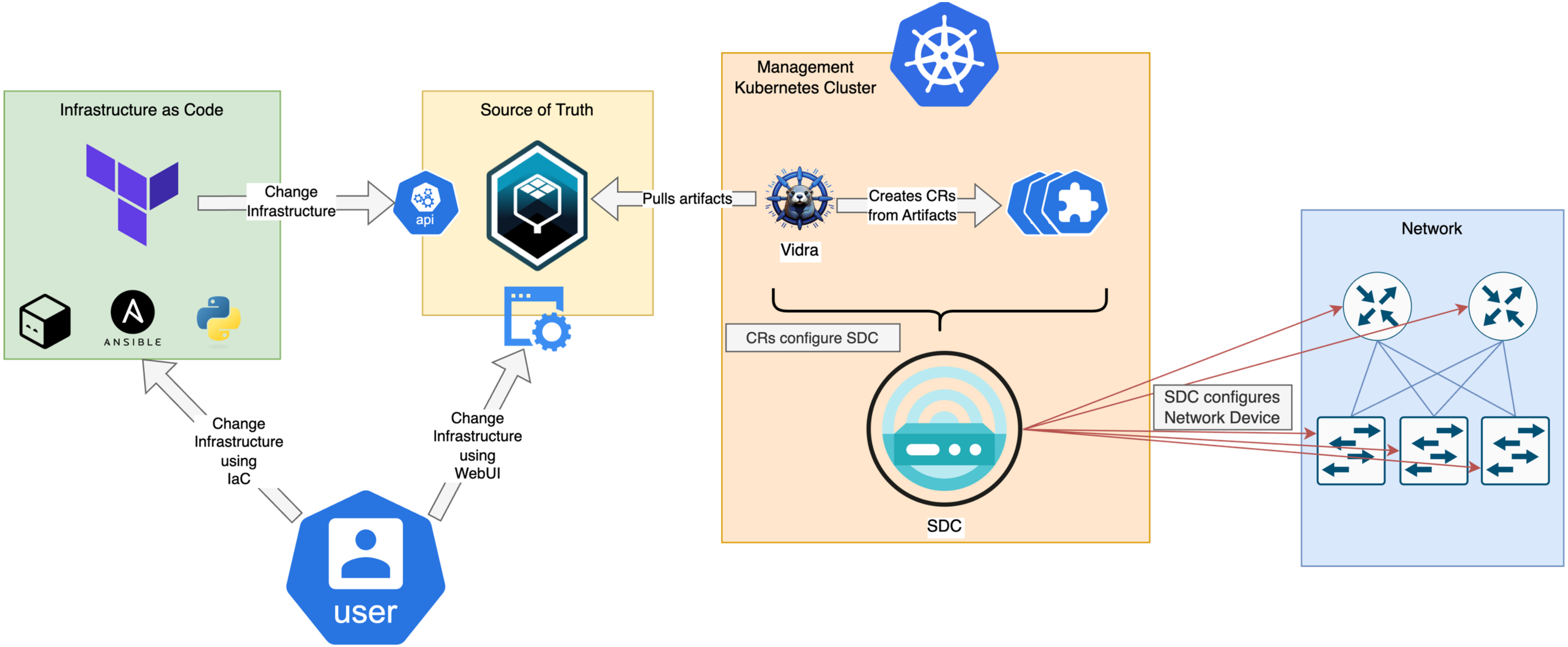


Work with what is given

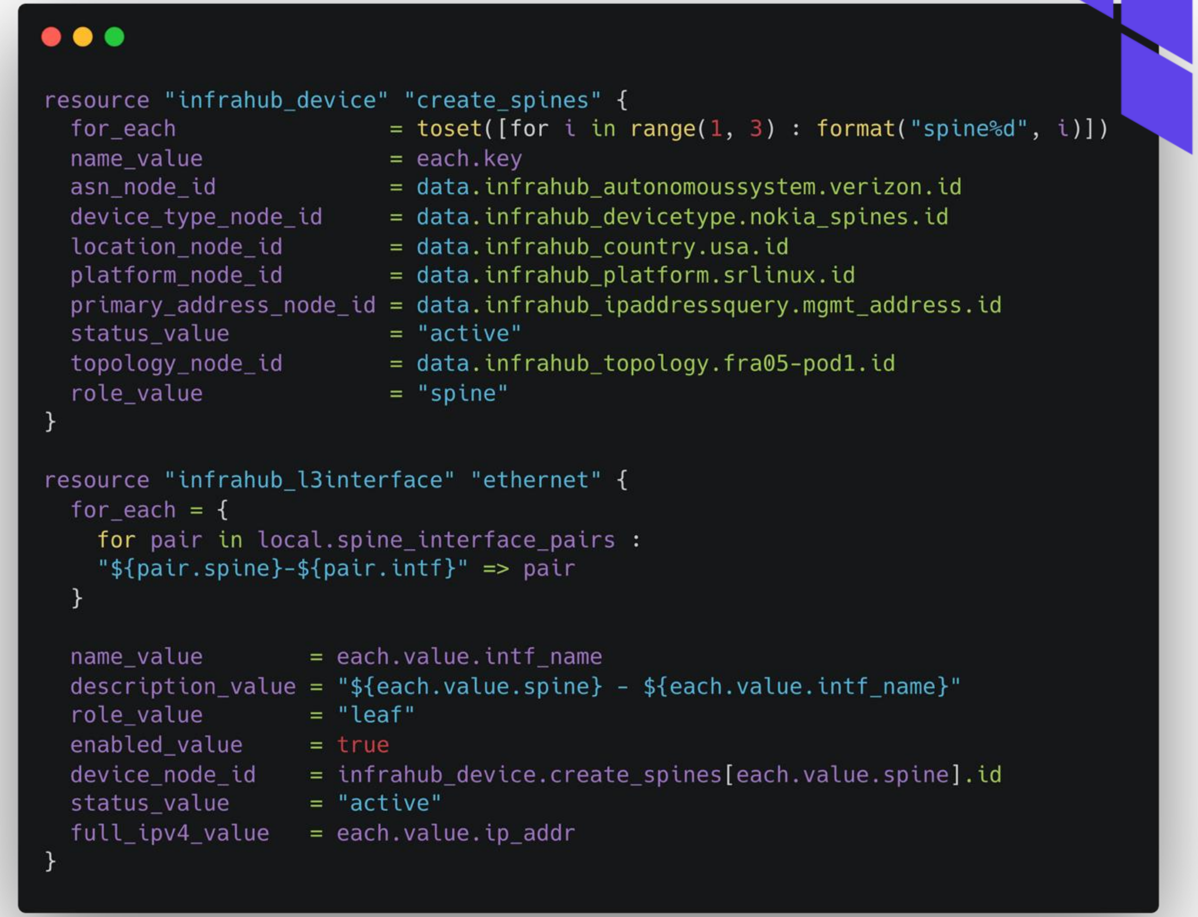
The idea of the Living Source of Truth is to bundle the complexity of the desired state and let the right tools take care of the tasks they know best to do – **create** and **manage** Infrastructure.



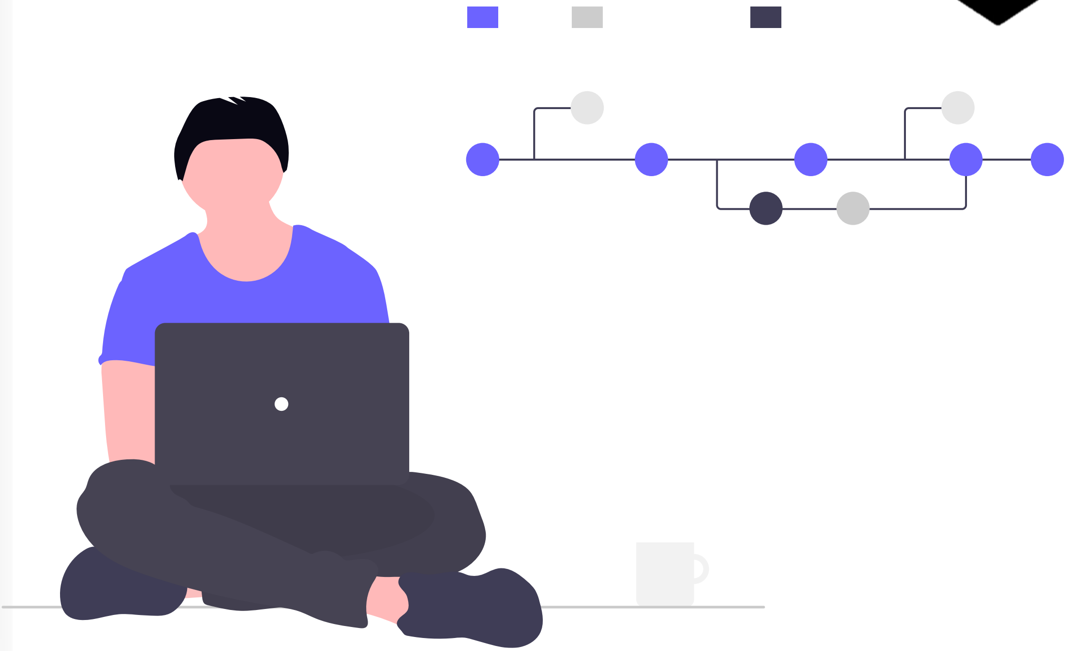
How to build a Living Source of Truth

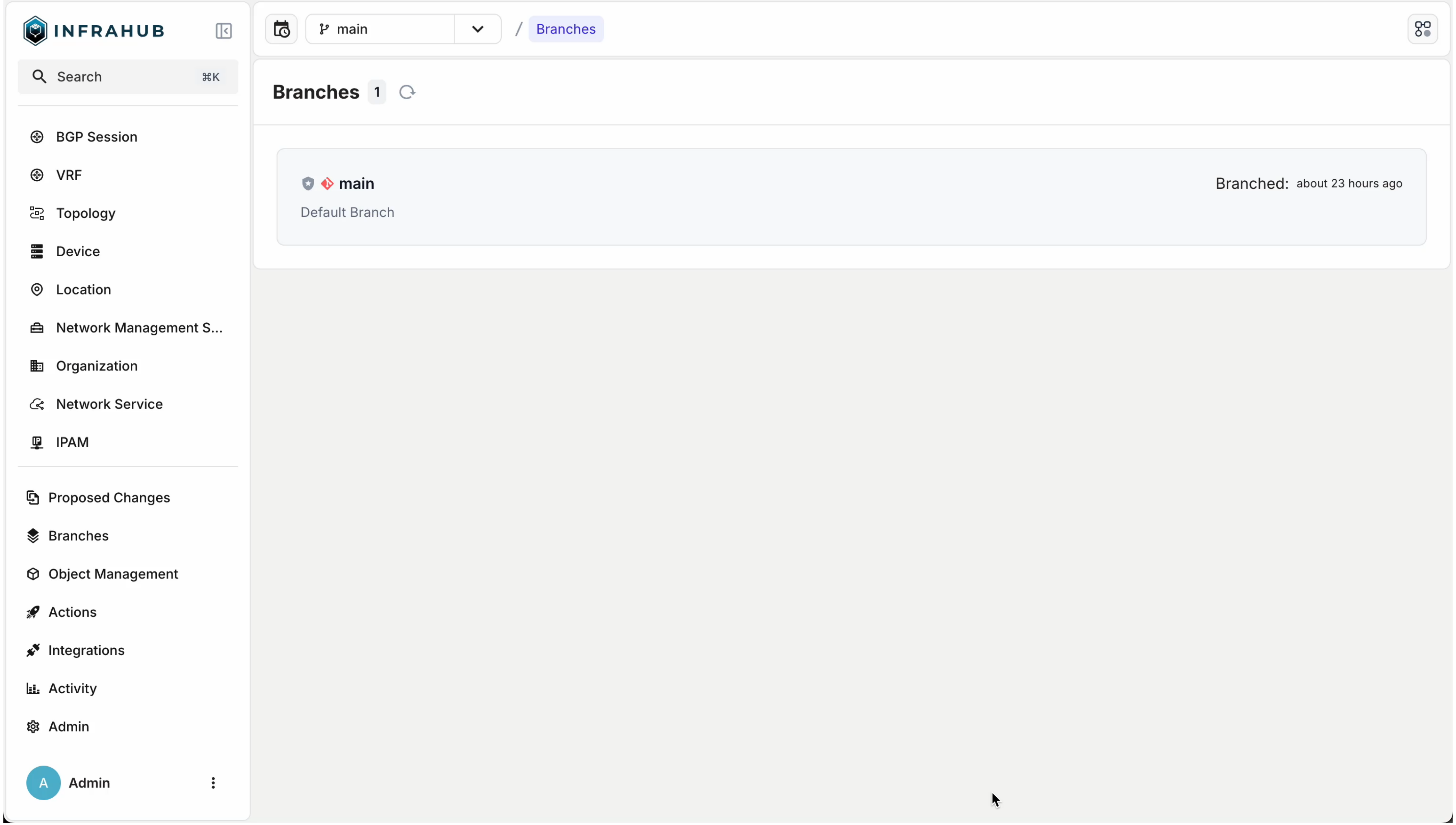

Infrastructure as Code
ANSIBLE
Change Infrastructure
api
Source of Truth
Pulls artifacts
Management Kubernetes Cluster
Vidra
Creates CRs from Artifacts
CRs configure SDC
SDC
SDC configures Network Device
Network
Change Infrastructure using IaC
Change Infrastructure using WebUI
user

Defining our Network



```
resource "infrahub_device" "create_spines" {
  for_each                 = toset([for i in range(1, 3) : format("spine%d", i)])
  name_value               = each.key
  asn_node_id              = data.infrahub_autonomoussystem.verizon.id
  device_type_node_id      = data.infrahub_devicetype.nokia_spines.id
  location_node_id         = data.infrahub_country.usa.id
  platform_node_id         = data.infrahub_platform.srlinux.id
  primary_address_node_id = data.infrahub_ipaddressquery.mgmt_address.id
  status_value             = "active"
  topology_node_id         = data.infrahub_topology.fra05-pod1.id
  role_value               = "spine"
}

resource "infrahub_l3interface" "ethernet" {
  for_each = {
    for pair in local.spine_interface_pairs :
    "${pair.spine}-${pair.intf}" => pair
  }

  name_value        = each.value.intf_name
  description_value = "${each.value.spine} - ${each.value.intf_name}"
  role_value        = "leaf"
  enabled_value     = true
  device_node_id    = infrahub_device.create_spines[each.value.spine].id
  status_value      = "active"
  full_ipv4_value   = each.value.ip_addr
}
```





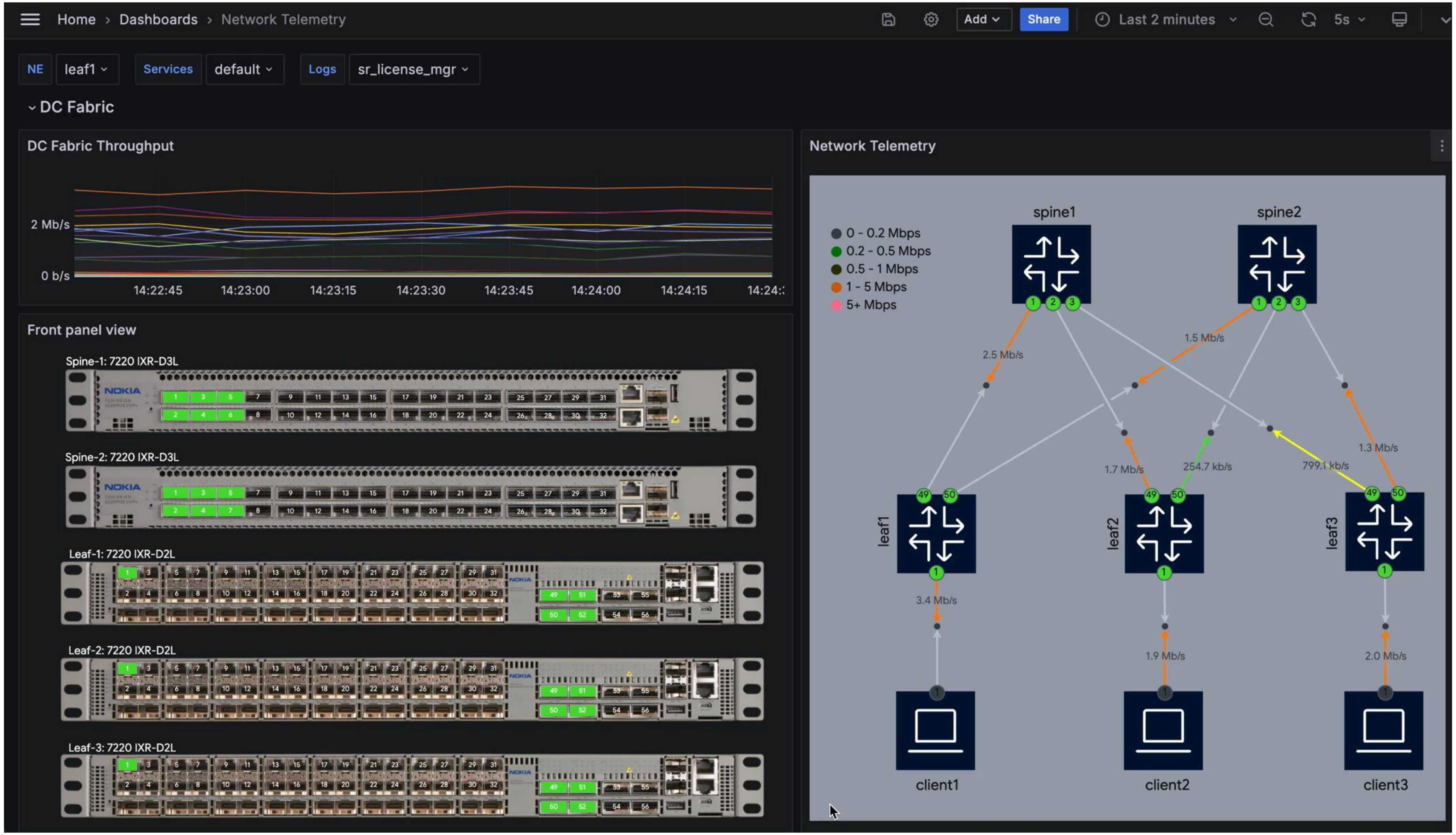
Branches 1

main

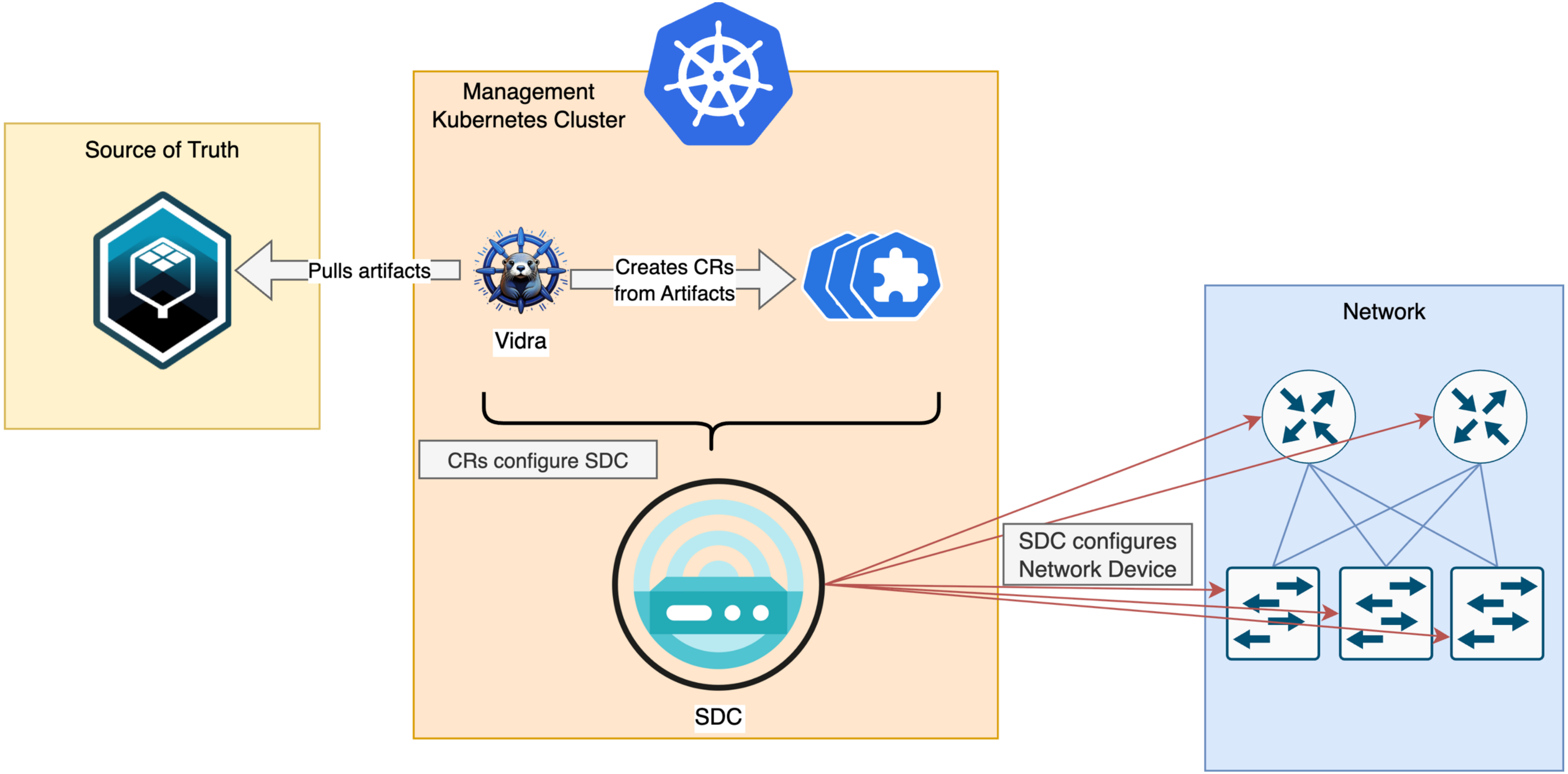
Default Branch

Branched: about 23 hours ago

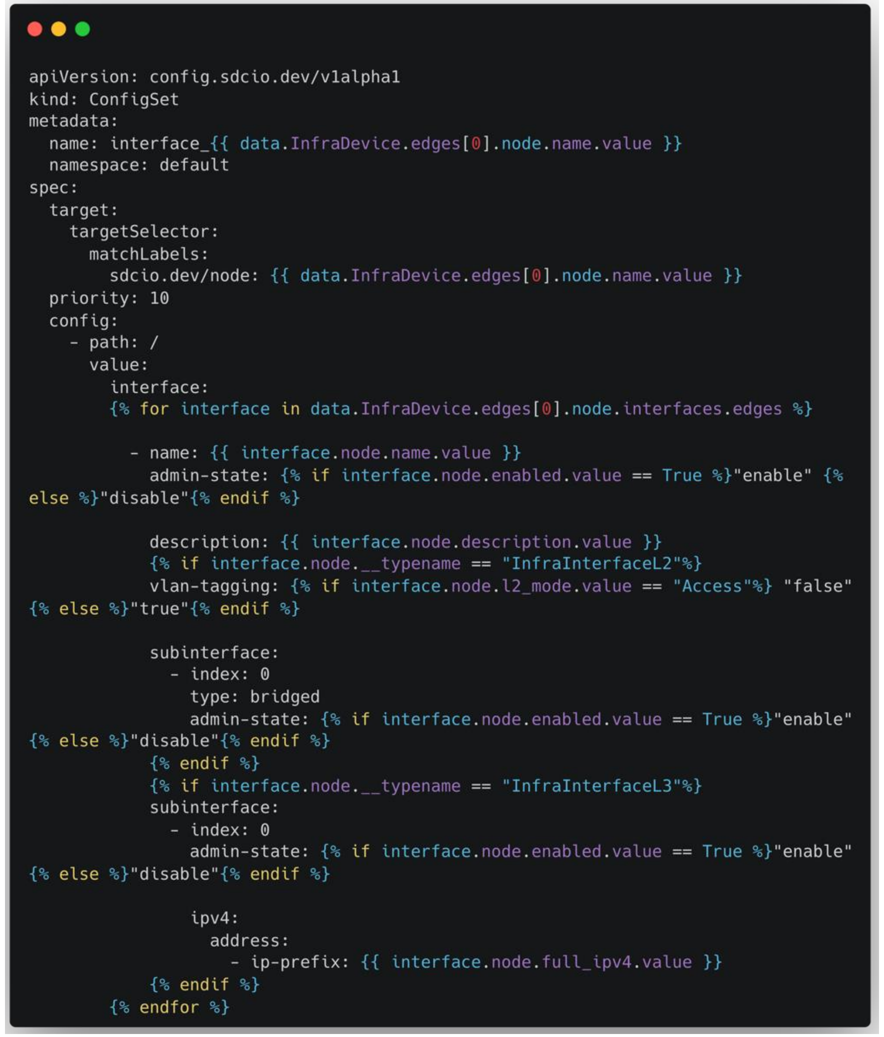
What we don’t see, we can't measure



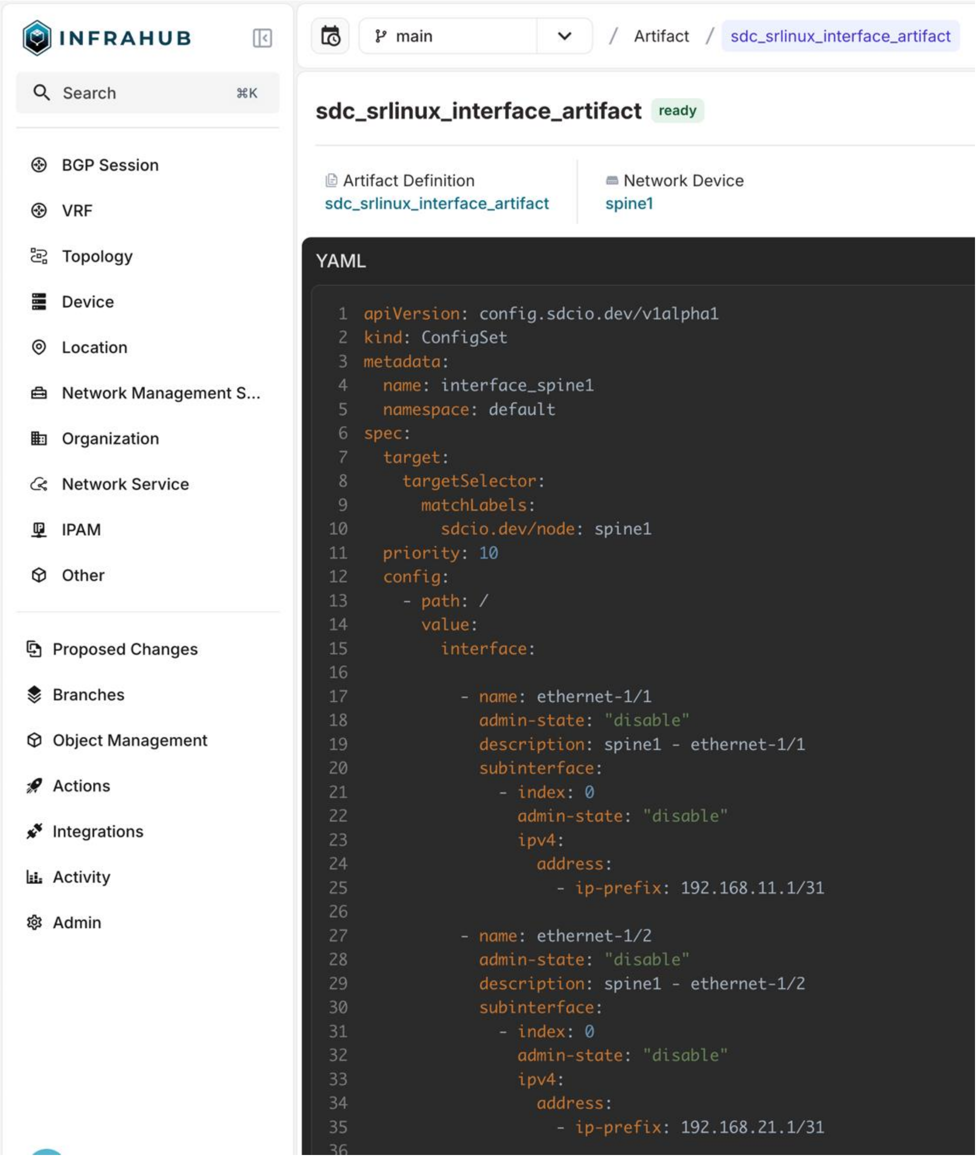
Infrahub, Vidra and SDC


Source of Truth
Pulls artifacts
Management Kubernetes Cluster
Vidra
Creates CRs from Artifacts
CRs configure SDC
SDC
SDC configures Network Device
Network

Infrahub Artifacts



```
apiVersion: config.sdcio.dev/v1alpha1
kind: ConfigSet
metadata:
  name: interface_{{ data.InfraDevice.edges[0].node.name.value }}
  namespace: default
spec:
  target:
    targetSelector:
      matchLabels:
        sdcio.dev/node: {{ data.InfraDevice.edges[0].node.name.value }}
  priority: 10
  config:
    - path: /
      value:
        interface:
        {% for interface in data.InfraDevice.edges[0].node.interfaces.edges %}

          - name: {{ interface.node.name.value }}
            admin-state: {% if interface.node.enabled.value == True %}"enable" {%
else %}"disable"{% endif %}

            description: {{ interface.node.description.value }}
            {% if interface.node.__typename == "InfraInterfaceL2"%}
            vlan-tagging: {% if interface.node.l2_mode.value == "Access"%} "false"
{% else %}"true"{% endif %}

            subinterface:
              - index: 0
                type: bridged
                admin-state: {% if interface.node.enabled.value == True %}"enable"
{% else %}"disable"{% endif %}
            {% endif %}
            {% if interface.node.__typename == "InfraInterfaceL3"%}
            subinterface:
              - index: 0
                admin-state: {% if interface.node.enabled.value == True %}"enable"
{% else %}"disable"{% endif %}

                ipv4:
                  address:
                    - ip-prefix: {{ interface.node.full_ipv4.value }}
            {% endif %}
        {% endfor %}
```



sdc_srlinux_interface_artifact ready

Artifact Definition: sdc_srlinux_interface_artifact

Network Device: spine1

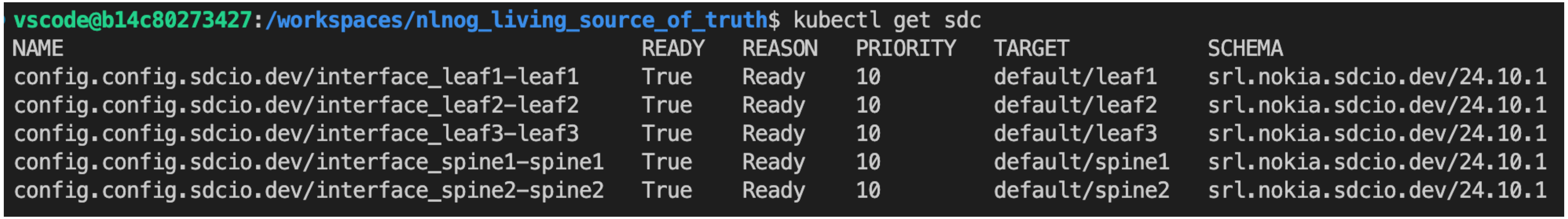
YAML

```
apiVersion: config.sdcio.dev/v1alpha1
kind: ConfigSet
metadata:
  name: interface_spine1
  namespace: default
spec:
  target:
    targetSelector:
      matchLabels:
        sdcio.dev/node: spine1
  priority: 10
  config:
    - path: /
      value:
        interface:

          - name: ethernet-1/1
            admin-state: "disable"
            description: spine1 - ethernet-1/1
            subinterface:
              - index: 0
                admin-state: "disable"
                ipv4:
                  address:
                    - ip-prefix: 192.168.11.1/31

          - name: ethernet-1/2
            admin-state: "disable"
            description: spine1 - ethernet-1/2
            subinterface:
              - index: 0
                admin-state: "disable"
                ipv4:
                  address:
                    - ip-prefix: 192.168.21.1/31
```

Vidra and SDC

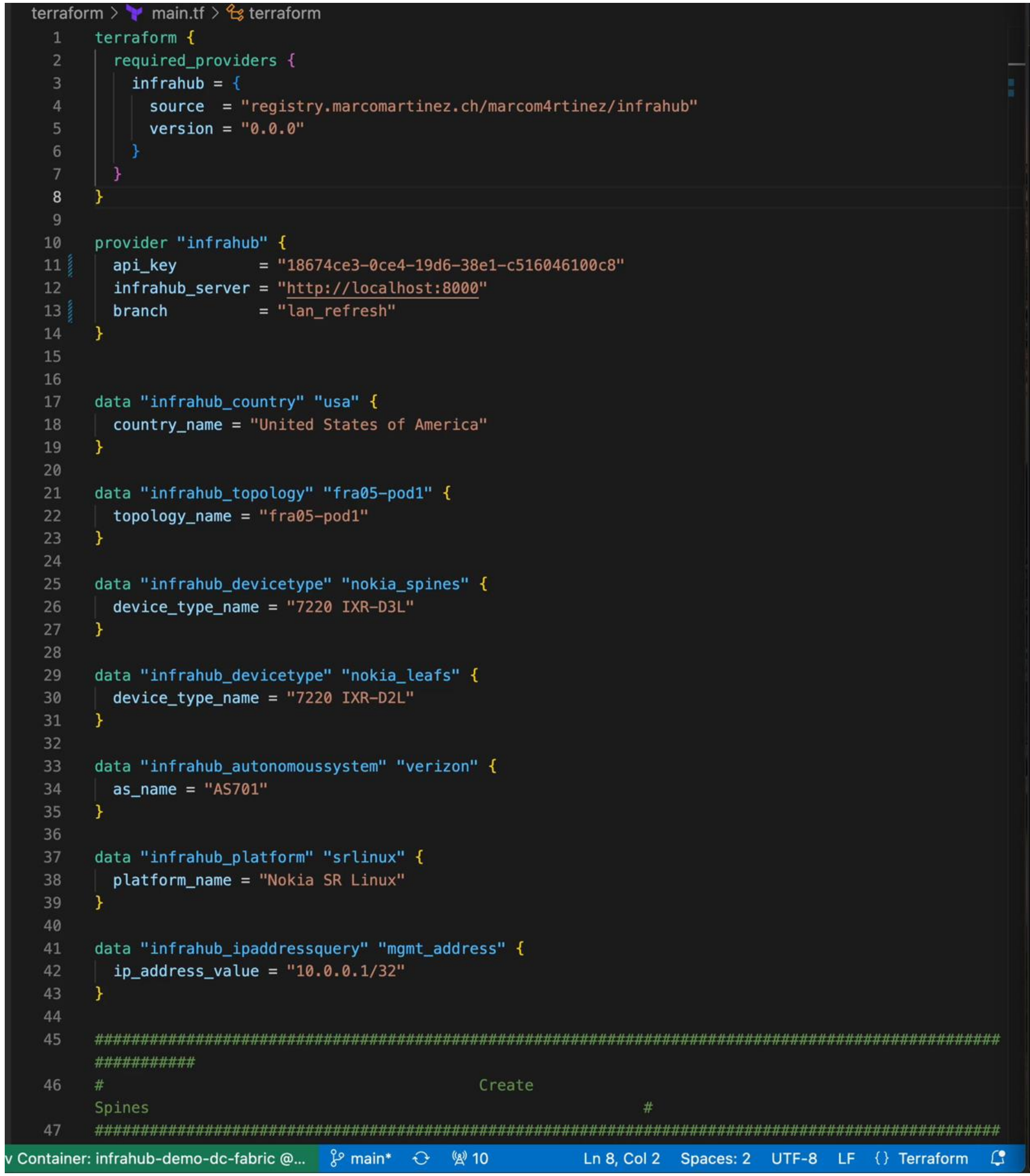
Vidra takes the artifacts and **applies** them on the Kubernetes Cluster.

- Reduces manual steps
- Ensures our Source of Truth is the configured state
- Will overwrite manual changes on Kubernetes



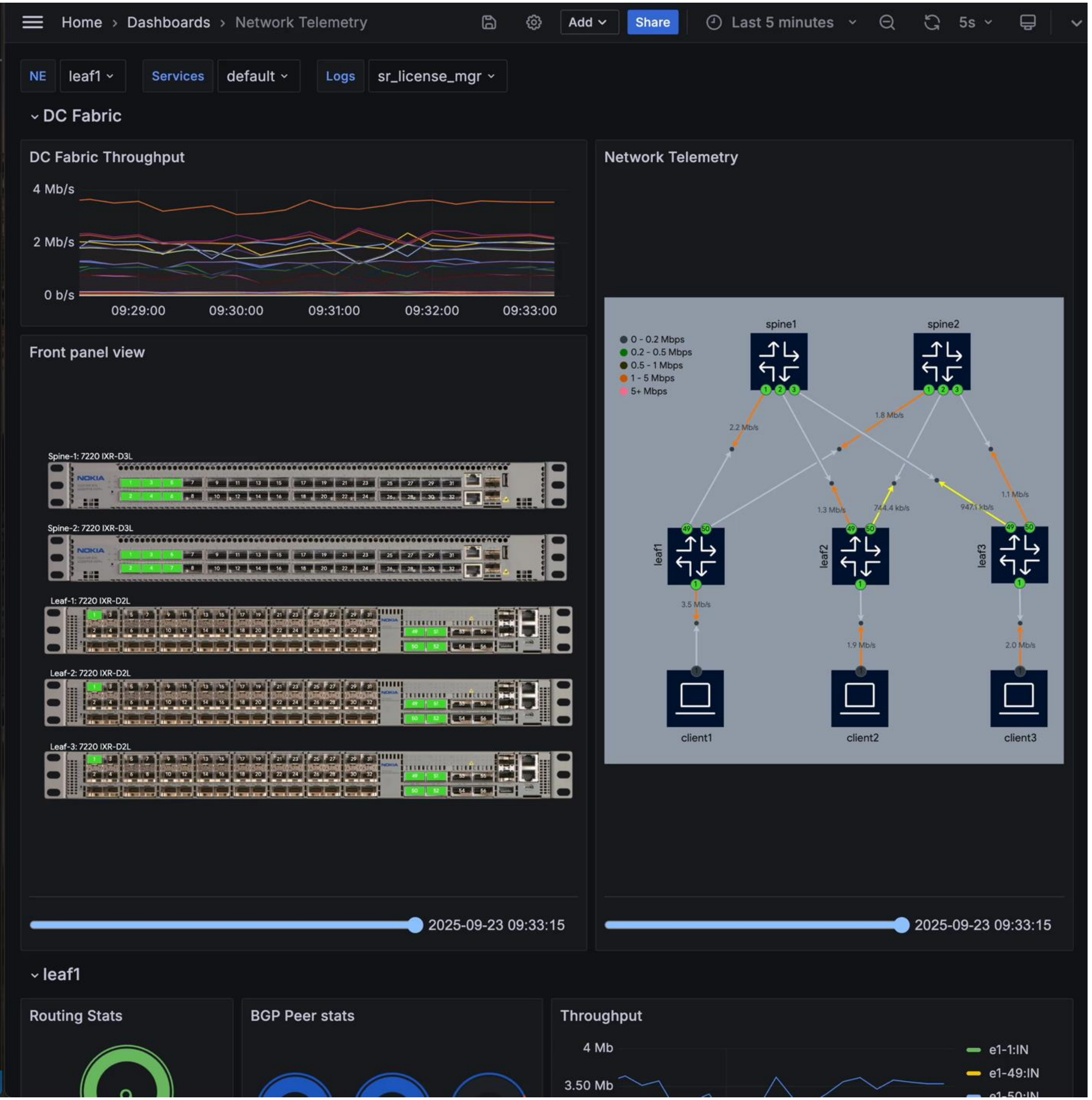
```
vscode@b14c80273427:/workspaces/nlnog_living_source_of_truth$ kubectl get sdc
NAME                                         READY   REASON   PRIORITY   TARGET           SCHEMA
config.config.sdcio.dev/interface_leaf1-leaf1     True    Ready    10         default/leaf1    srl.nokia.sdcio.dev/24.10.1
config.config.sdcio.dev/interface_leaf2-leaf2     True    Ready    10         default/leaf2    srl.nokia.sdcio.dev/24.10.1
config.config.sdcio.dev/interface_leaf3-leaf3     True    Ready    10         default/leaf3    srl.nokia.sdcio.dev/24.10.1
config.config.sdcio.dev/interface_spine1-spine1   True    Ready    10         default/spine1   srl.nokia.sdcio.dev/24.10.1
config.config.sdcio.dev/interface_spine2-spine2   True    Ready    10         default/spine2   srl.nokia.sdcio.dev/24.10.1
```

SDC will then **configure** the devices based on the **desired state**

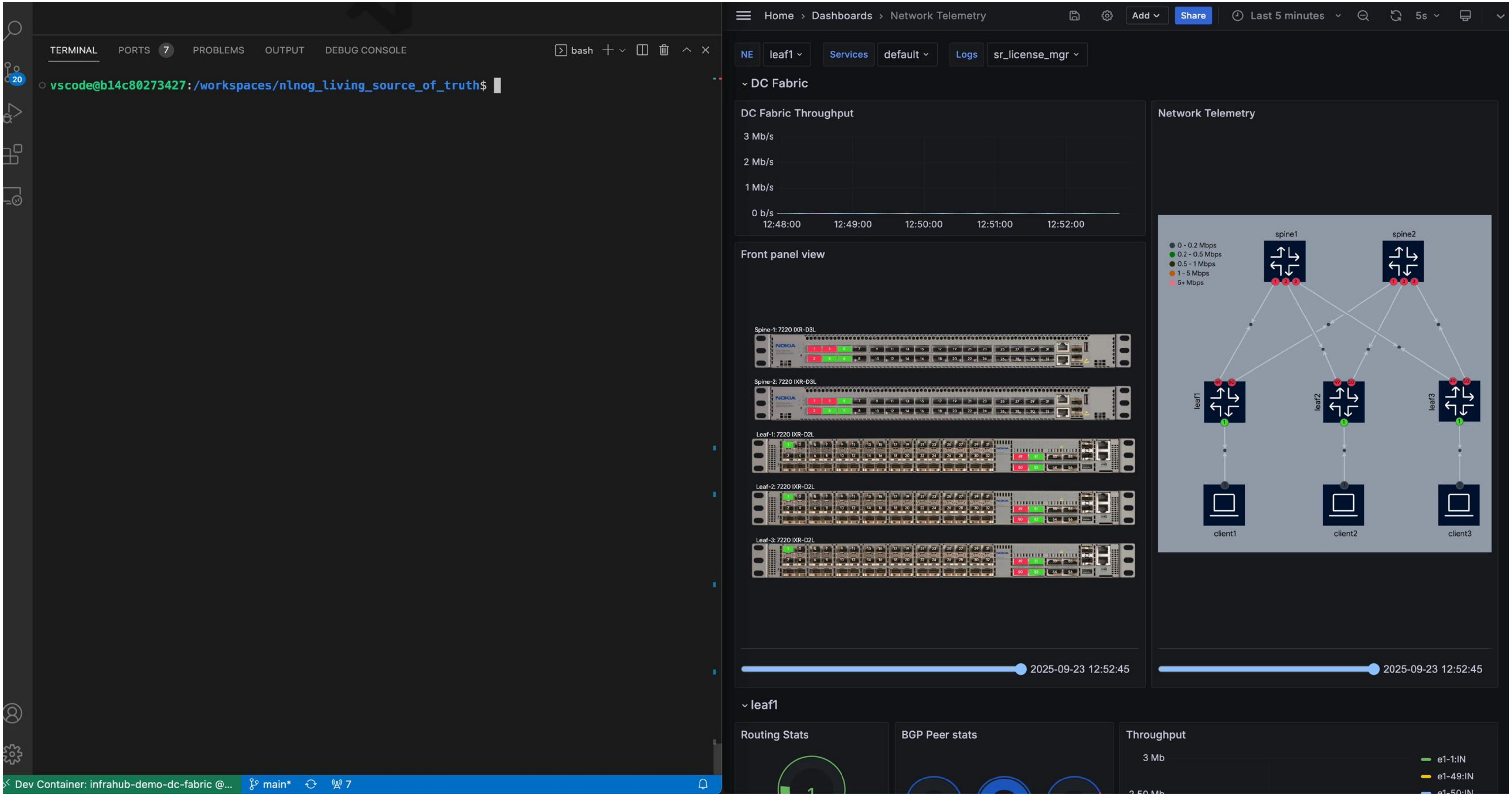


terraform > main.tf > terraform

```
terraform {
  required_providers {
    infrahub = {
      source  = "registry.marcomartinez.ch/marcom4rtinez/infrahub"
      version = "0.0.0"
    }
  }
}

provider "infrahub" {
  api_key         = "18674ce3-0ce4-19d6-38e1-c516046100c8"
  infrahub_server = "http://localhost:8000"
  branch          = "lan_refresh"
}


data "infrahub_country" "usa" {
  country_name = "United States of America"
}

data "infrahub_topology" "fra05-pod1" {
  topology_name = "fra05-pod1"
}

data "infrahub_devicetype" "nokia_spines" {
  device_type_name = "7220 IXR-D3L"
}

data "infrahub_devicetype" "nokia_leafs" {
  device_type_name = "7220 IXR-D2L"
}

data "infrahub_autonomoussystem" "verizon" {
  as_name = "AS701"
}

data "infrahub_platform" "srlinux" {
  platform_name = "Nokia SR Linux"
}

data "infrahub_ipaddressquery" "mgmt_address" {
  ip_address_value = "10.0.0.1/32"
}

##############################################################################################
##########
#                                         Create
Spines                                                    #
##############################################################################################
```

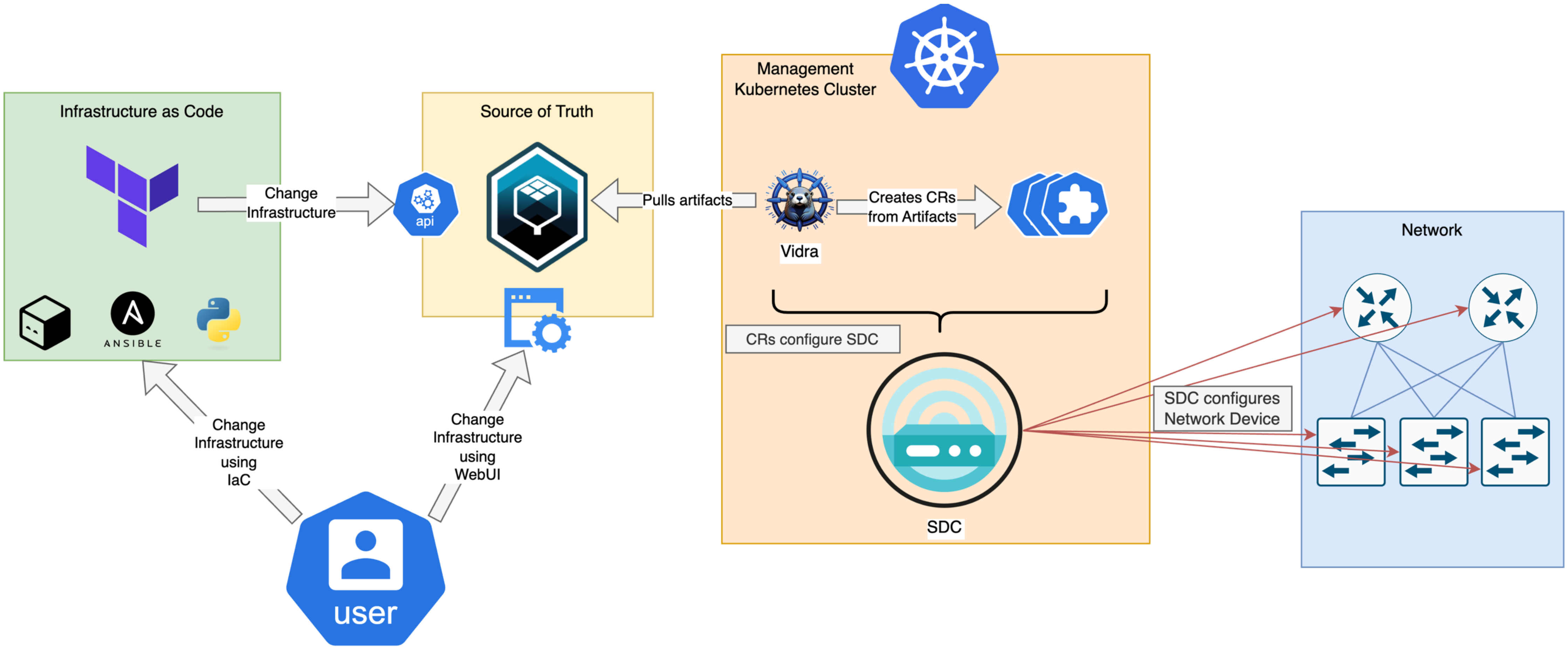


Let me fix it!

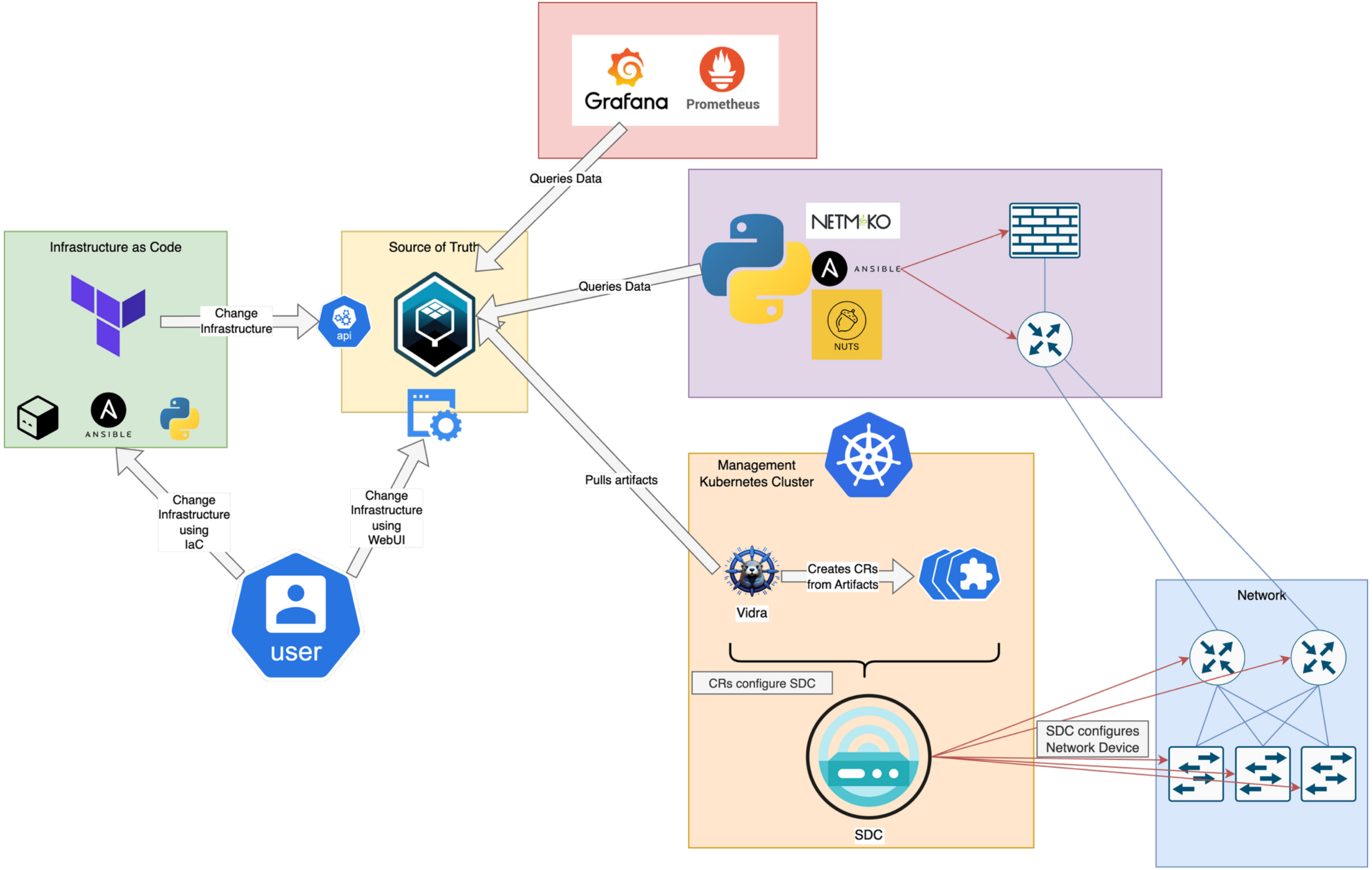


```
vscode@b14c80273427:/workspaces/nlnog_living_source_of_truth$
```

How to build a Living Source of Truth


Infrastructure as Code
ANSIBLE
Change Infrastructure
api
Source of Truth
Pulls artifacts
Management Kubernetes Cluster
Vidra
Creates CRs from Artifacts
CRs configure SDC
SDC
SDC configures Network Device
Network
Change Infrastructure using IaC
Change Infrastructure using WebUI
user

How to build a Living Source of Truth

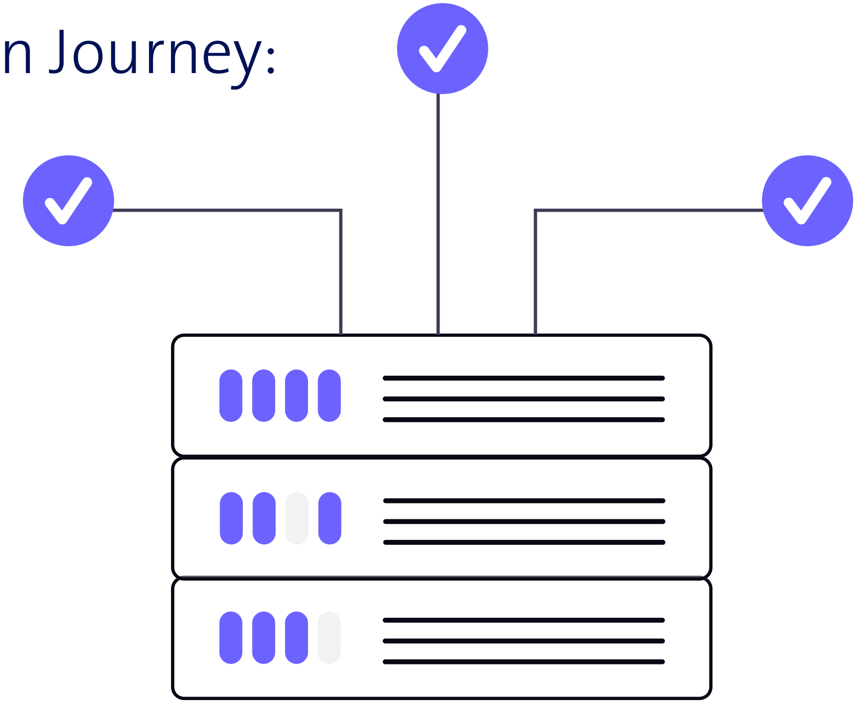


My network already has automation

Most Engineers do not start in a greenfield Environment

Advice for people wanting to take the automation Journey:

- Have a centralised data source
- Start small
- Fix the most time-consuming issues first
- Use the tools available



The Goal

An **idempotent infrastructure** based on a reliable Data Source

Transparency

Security

Single pane of glass

Questions