



# DIY Intent-based Networking

Robin Gilijamse

# whoami

## Robin Gilijamse

Network Design & Automation



Consultant @ One Zero IT



Co-founder @ ncubed



@RobinGilijamse



# Disclaimer

1. Non-production code
2. (Somewhat) dated tool choices
3. ...

# My smart smoke sensor



What's nice about a smart smoke detector?

- ✓ Alerts in an app
- ✓ Can be monitored

Government of the Netherlands

Home > Documents >

Search

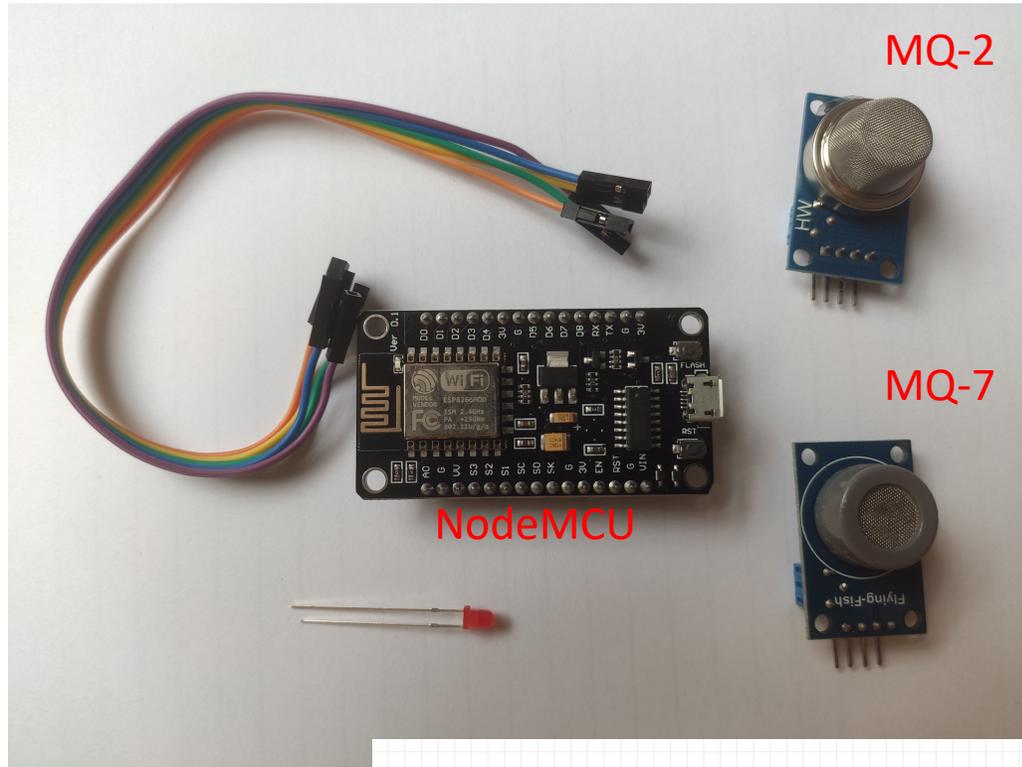
### How many smoke detectors do I need to have in my home?

From 1 July 2022 every storey of your home must be equipped with a smoke detector.

A close-up portrait of a man with dark, wavy hair and glasses. He has a thoughtful or questioning expression, with his hand resting on his chin. The background is a plain, light-colored wall. The overall lighting is soft, and the image has a slightly desaturated, muted color palette.

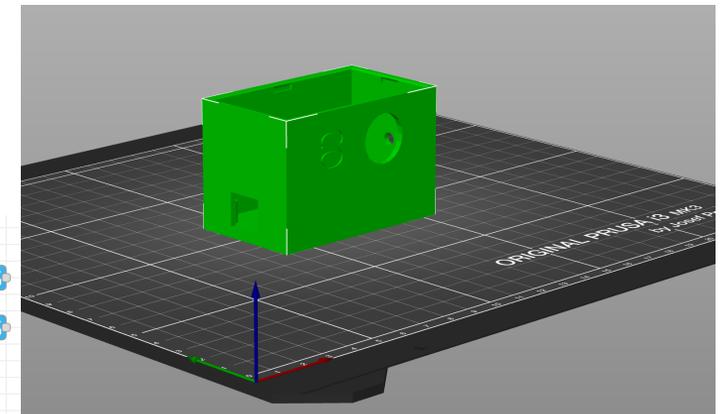
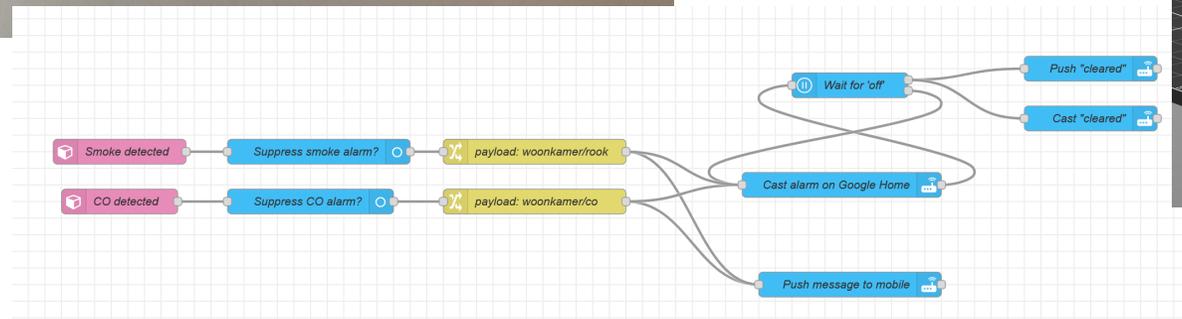
Can I do this myself?

# DIY Smoke Detector



I had to learn about:

- ESP8266
- Node-RED
- Soldering resistors to LEDs
- 3D-printing a custom case
- ...



# My smoke sensor in production

*Because I'm not trusting my family's safety to my tinkering skills quite yet.*



# Intent Based Networking

---

theory

# What is IBN?

- IBN transforms a hardware-centric, manual network into a **controller-led network** that captures **business intent** and translates it into policies that can be automated and applied consistently across the network. [CISCO]
- When operators describe the **business outcomes** they wish to accomplish, the network converts those objectives into the configuration necessary to achieve them, **without individual tasks** having to be coded and executed manually. [JUNIPER]
- [IBN is] a **“smart” topology** that possesses the capability to monitor overall network performance, identify issues, and **solve problems automatically** without manual intervention. [FORBES]

The Network Can Run Itself!! It's MAGIC!!!

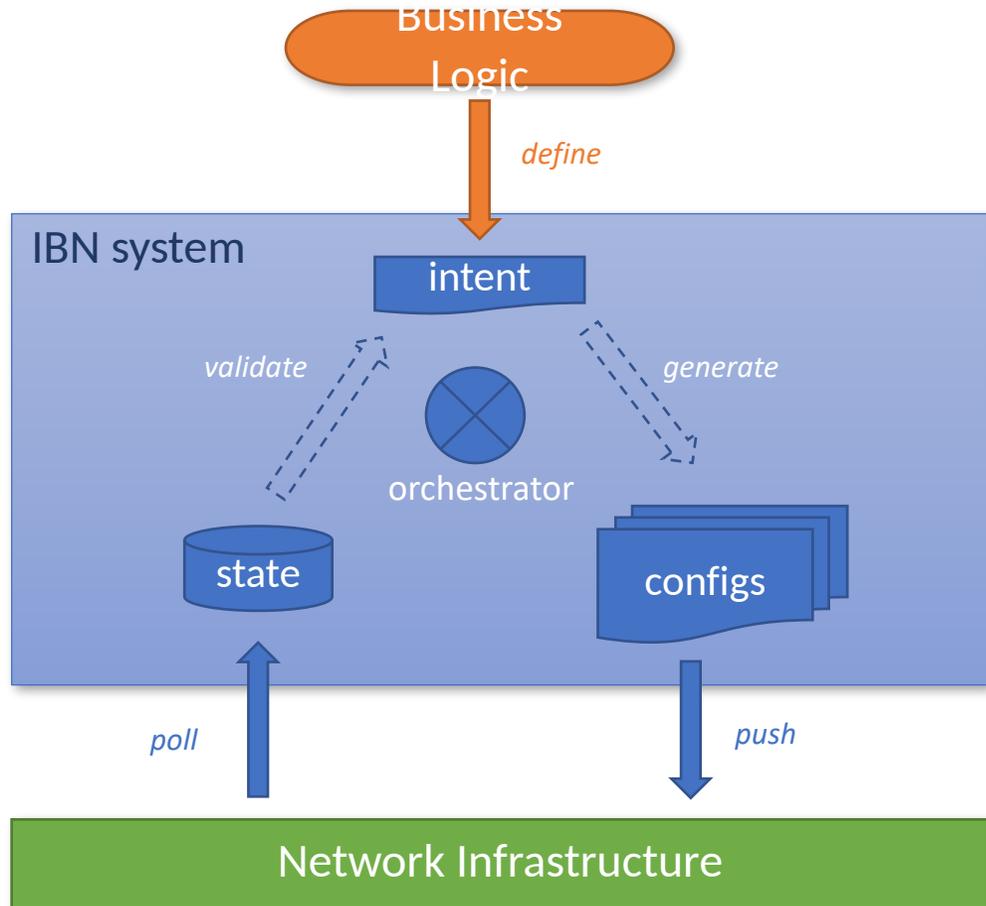
# IBN - Concepts and Definitions (IETF)

*draft-irtf-nmrg-ibn-concepts-definitions-09*

**IBN:** Intent-Based Network - A network that can be managed using intent.

**Intent:** A set of **operational goals** (that a network should meet) and outcomes (that a network is supposed to deliver), defined in a **declarative manner** without specifying how to achieve or implement them.

# So what is it?



Intent is:

- Make sure there's plenty bandwidth available **NO SUCH LUCK**
- Don't call me at 3:00 AM to fix something
- Tunnel traffic between these two endpoints
- Make sure the coreswitch is STP root
- Make me a leaf/spine fabric

Domain-wide policies, not device-level configuration

A close-up portrait of a man with dark, wavy hair and glasses. He has a thoughtful or questioning expression, with his hand resting on his chin. The background is a plain, light-colored wall. The overall lighting is soft, and the image has a slightly desaturated, muted color palette.

Can I do this myself?

# My homebrew IBN

---

using Ansible

# Overview

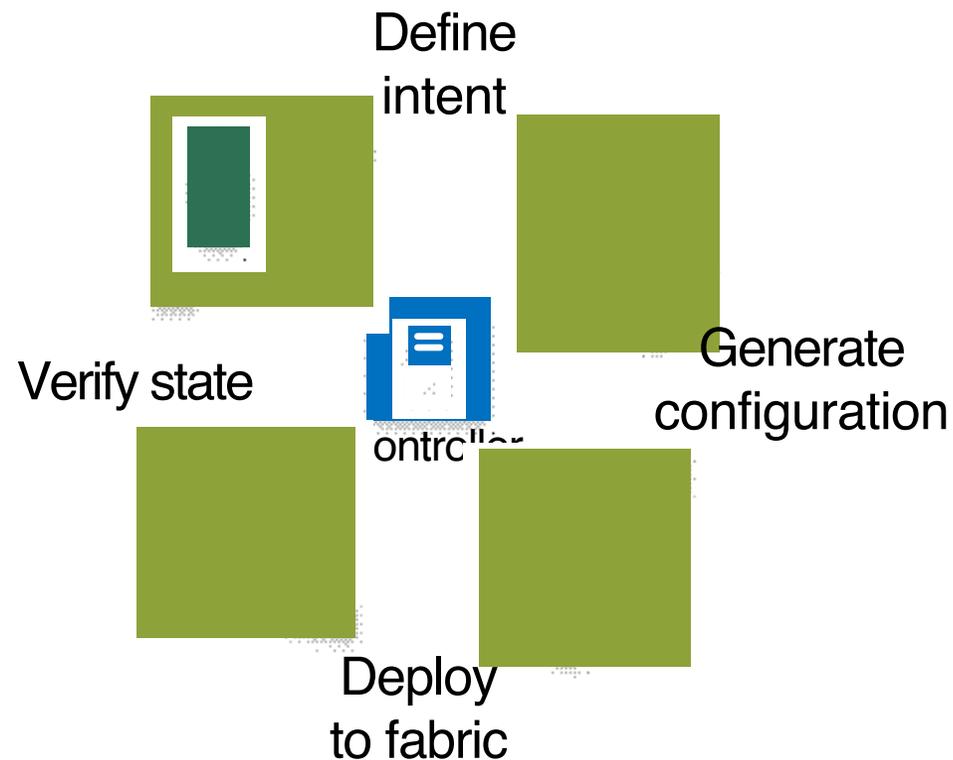
Example:

leaf/spine fabric with EVPN/VXLAN

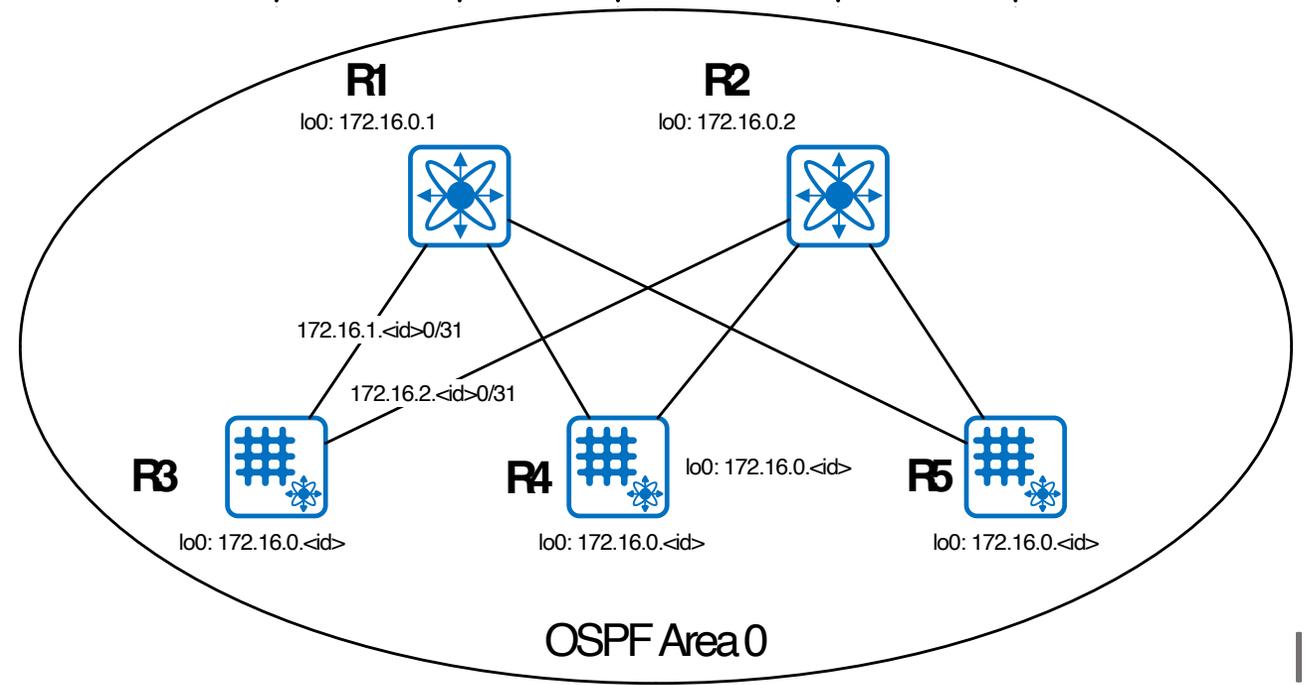
**Intent:** “these are my leaves & spines”

**Configuration:** all the messy details

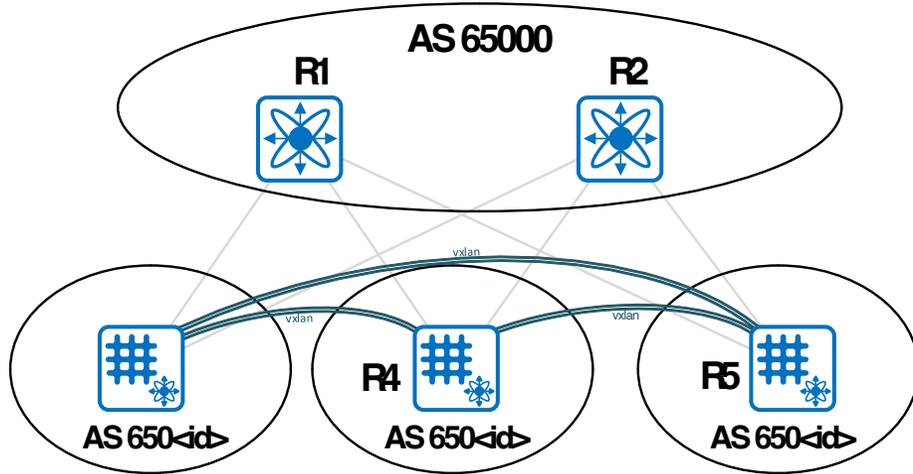
**Verification:** BGP peerings



# Topology



## Overlay



lay

# Happy state

- Each spine is peered with every leaf switch

```
show bgp summary
```

Peer	AS	InPkt	OutPkt	OutQ	Flaps	Last	Up/Dwn	State
172.16.0.11	65011	1357	1343	0	0	10:16:52	Establ	
172.16.0.12	65012	1355	1345	0	0	10:16:48	Establ	
172.16.0.13	65013	1367	1344	0	0	10:16:48	Establ	

# Configuration



- Variables:
  - Interface configuration
  - OSPF settings
  - BGP settings
  - ...

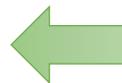
Configuration details != intent

```
routing-options {
  router-id {{router_id}};
}
protocols {
  bgp {
    replace: group OVERLAY {
      type external;
      multihop;
      local-address
      {{loopback_address}};
      family evpn {
        signaling;
      }
      local-as {{local_as}};
    }
    {% for peer in peers %}
    neighbor {{peer.address}} {
      peer-as {{peer.as}};
    }
  }
  {% endfor %}
}
ospf {
  replace:
  int
  {% for peer in
  int
  {% endfor %}
}
}
interfaces {
  {% for interface in interfaces %}
  replace: {{ interface.name }} {
    description
    "{{interface.description}}";
    unit 0 {
      family inet {
        address
        {{ interface.address }};
      }
    }
  }
  {% endfor %}
}
lo0 {
  unit 0 {
    family inet {
      address {{loopback_address}}/
    }
  }
}
32;
```

# Defining Intent

- Intent is as high-level as possible
- Use Jinja templates to generate ansible inventory

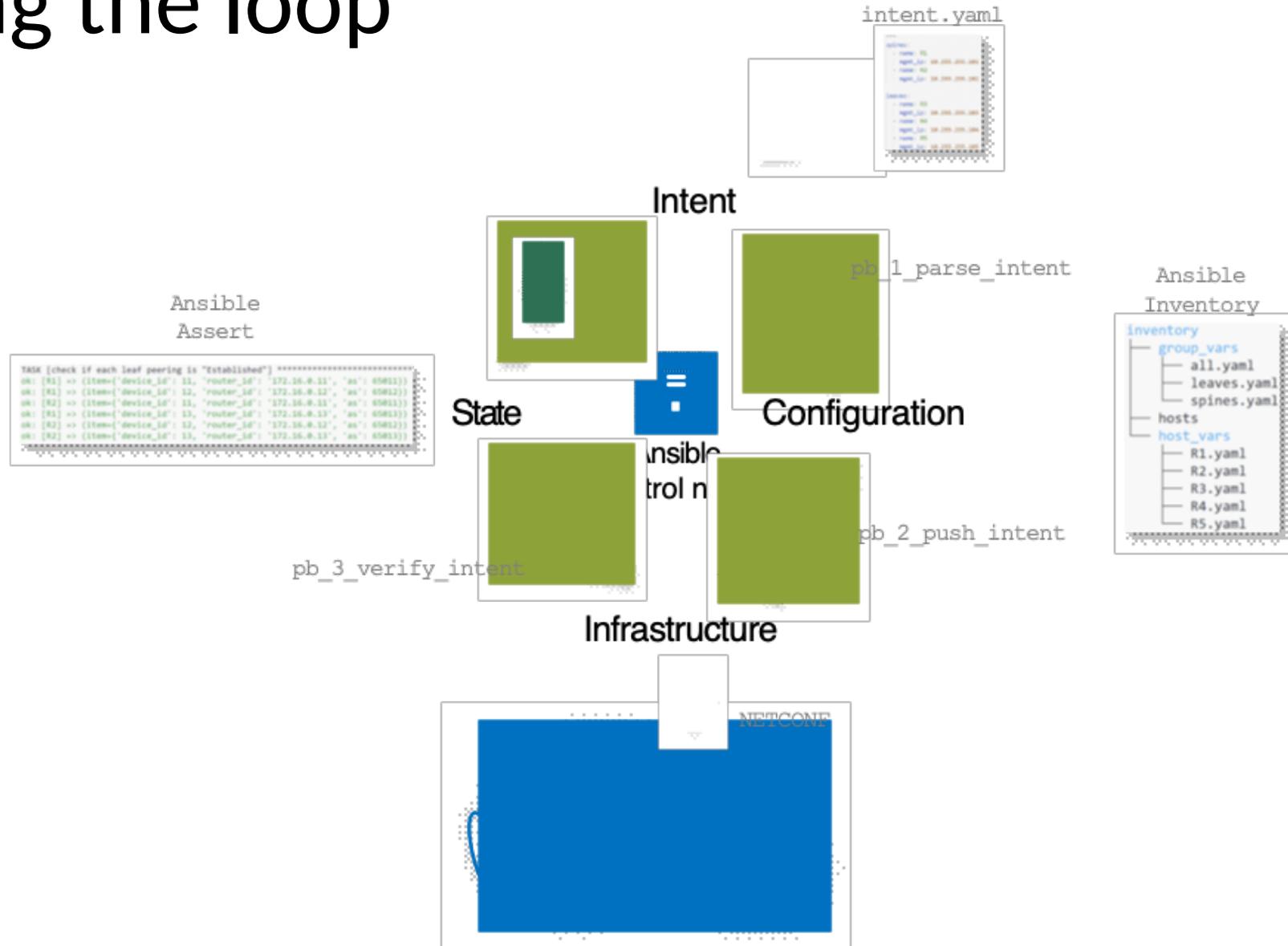
```
user@controller:~/ansible$ tree
inventory
├── group_vars
│   ├── all.yaml
│   ├── leaves.yaml
│   └── spines.yaml
├── hosts
└── host_vars
    ├── R1.yaml
    ├── R2.yaml
    ├── R3.yaml
    ├── R4.yaml
    └── R5.yaml
```



```
---
spines:
  - name: R1
    mgmt_ip: 10.255.255.101
  - name: R2
    mgmt_ip: 10.255.255.102

leaves:
  - name: R3
    mgmt_ip: 10.255.255.103
  - name: R4
    mgmt_ip: 10.255.255.104
  - name: R5
    mgmt_ip: 10.255.255.105
```

# Closing the loop



# Demo

---

pre-recorded – don't worry

```

.
├── ansible.cfg
├── bgp_parser.spec
├── configs
│   ├── R1.txt
│   ├── R3.txt
│   └── R4.txt
├── device_templates
│   ├── device_bootstrap_config
│   ├── leaf-interfaces.j2
│   ├── leaf.j2
│   ├── leaf-routing.j2
│   ├── spine-interfaces.j2
│   ├── spine.j2
│   └── spine-routing.j2
├── images
│   ├── IBNS-overview.png
│   ├── overlay.png
│   └── underlay.png
├── intent.yaml
├── inventory
│   ├── group_vars
│   │   ├── all.yaml
│   │   ├── leaves.yaml
│   │   └── spines.yaml
│   ├── hosts
│   └── host_vars
│       ├── R1.yaml
│       ├── R3.yaml
│       └── R4.yaml
├── inventory_templates
│   ├── hosts.j2
│   ├── host_vars-leaf.yaml.j2
│   ├── host_vars-spine.yaml.j2
│   ├── leaves.yaml.j2
│   └── spines.yaml.j2
├── pb_0_prepare_environment.sh
├── pb_1_parse_intent.yaml
├── pb_2_push_intent.yaml
├── pb_3_verify_intent.yaml
├── README.md
└── requirements.txt

```

```

7 directories, 34 files
user@controller:~/ansible$ |

```

```

admin@R1> show version
fpc0:
-----
Hostname: R1
Model: vqfx-10000
Junos: 17.4R1.16 limited
JUNOS Base OS boot [17.4R1.16]
JUNOS Base OS Software Suite [17.4R1.16]
JUNOS Crypto Software Suite [17.4R1.16]
JUNOS Online Documentation [17.4R1.16]
JUNOS Kernel Software Suite [17.4R1.16]
JUNOS Packet Forwarding Engine Support (qfx-10-f) [17.4R1.16]
JUNOS Routing Software Suite [17.4R1.16]
JUNOS jsd [i386-17.4R1.16-jet-1]
JUNOS SDN Software Suite [17.4R1.16]
JUNOS Enterprise Software Suite [17.4R1.16]
JUNOS Web Management [17.4R1.16]
JUNOS py-base-i386 [17.4R1.16]
JUNOS py-extensions-i386 [17.4R1.16]

```

```

{master:0}
admin@R1>

```

```

admin@R2> show version
fpc0:
-----
Hostname: R2
Model: vqfx-10000
Junos: 17.4R1.16 limited
JUNOS Base OS boot [17.4R1.16]
JUNOS Base OS Software Suite [17.4R1.16]
JUNOS Crypto Software Suite [17.4R1.16]
JUNOS Online Documentation [17.4R1.16]
JUNOS Kernel Software Suite [17.4R1.16]
JUNOS Packet Forwarding Engine Support (qfx-10-f) [17.4R1.16]
JUNOS Routing Software Suite [17.4R1.16]
JUNOS jsd [i386-17.4R1.16-jet-1]
JUNOS SDN Software Suite [17.4R1.16]
JUNOS Enterprise Software Suite [17.4R1.16]
JUNOS Web Management [17.4R1.16]
JUNOS py-base-i386 [17.4R1.16]
JUNOS py-extensions-i386 [17.4R1.16]

```

```

{master:0}
admin@R2>

```

# Code

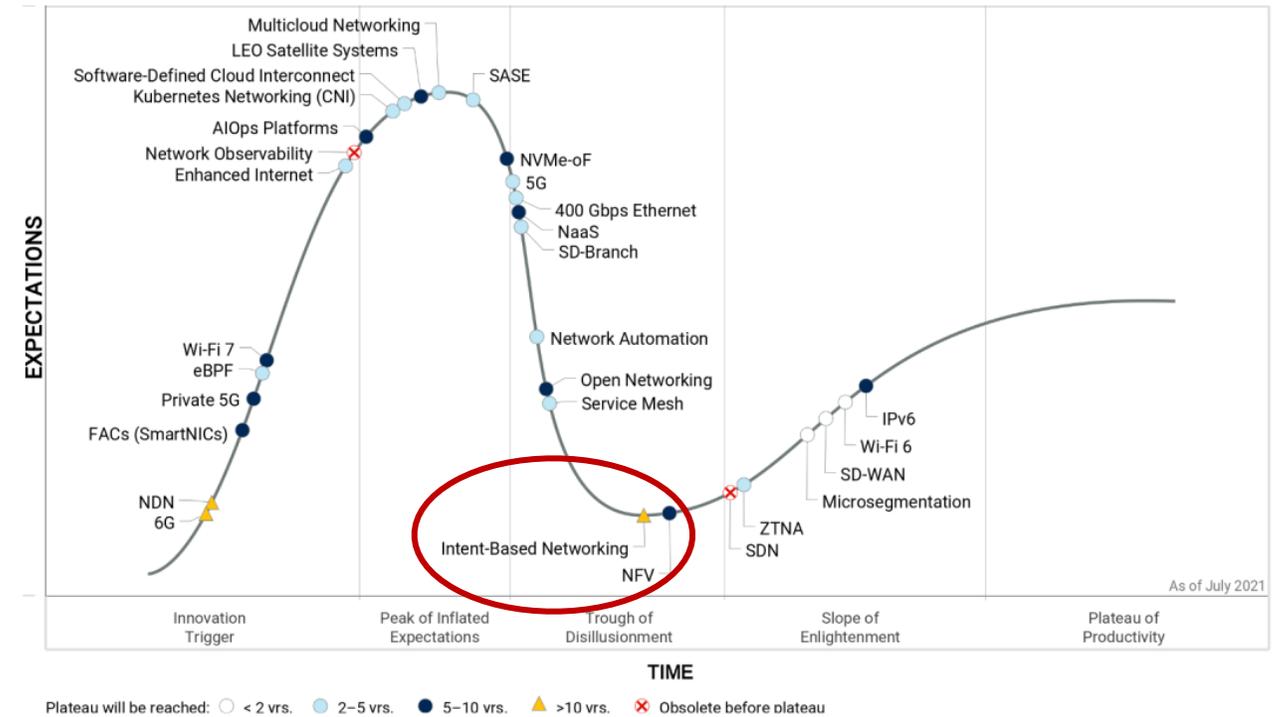
- Ansible playbooks
  1. Parse Intent | generate Ansible Inventory based on intent.yaml
  2. Push Configuration | render Jinja2 templates and push config to devices
  3. Verify Intent | get BGP state and compare with intent
- Inventory templates
  - Jinja2 templates for hosts file & variable files | this contains logic
- Device templates
  - Jinja2 templates for spine & leaf devices | as logic-free as possible

<https://github.com/rgilijamse/diy-ibn-ansible>

# In summary

- Not quite as magical as it sounds
- Hard to deliver on marketing promises
- Has it's uses, once you understand it

Hype Cycle for Enterprise Networking, 2021





# Questions?

I'll be at the bar.

Otherwise:

twitter: [@RobinGilijamse](#)

blog: [interestingtraffic.nl](#)