How my first Network Automation project failed (and is still in production)

Urs Baumann Swisscom 29.05.2024

Urs Baumann

```
>>> qr = QRCode()
>>> qr.add_data("https://www.linkedin.com/in/ubaumannch")
>>> qr.print_ascii()
```

How it begun ...

You have automated CCIE Lab deployments.
Could you build a ”Staging Robot”?

Customer A (2015)

A winning team

Software Engineer

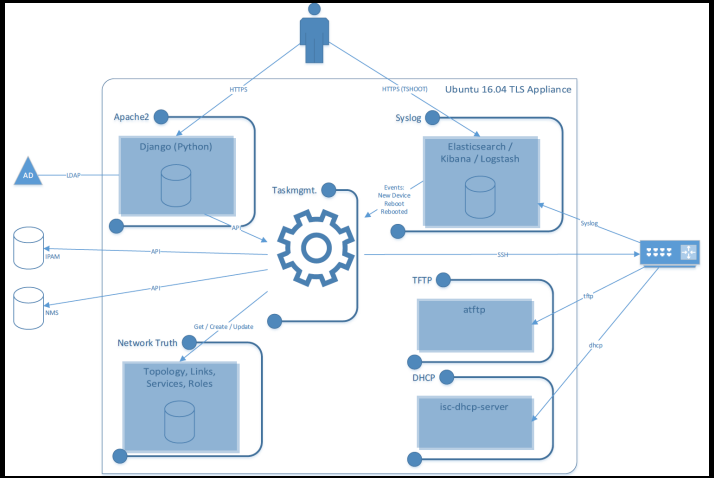
- System Engineering background
- ”Hardcore code reviewer”

Network Engineer

- Basic programming skills
- ”It is working, isn’t it?”

[1]System Engineer (BSc Student) with a flair for UI joined for UI

First Architecture



Source: SwiNOG #31 Network Automation – Road trip to an automated Network

Event Based

- No need to know in advance:
  - Serial Number
  - MAC Address
  - Model ID
- Support for different staging areas
- Staging directly at the destination

EEM

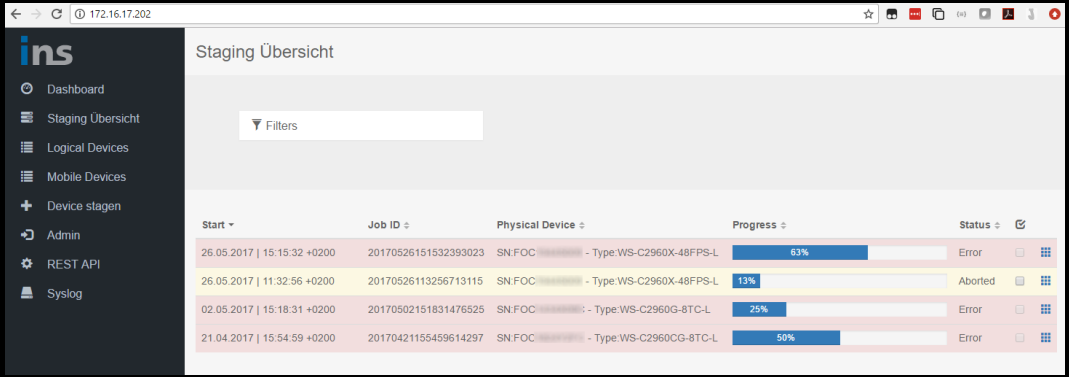
Disclamer: Needed support for IOS 12 (no Python, no PnP)

- The First idea was to use EEM
- EEM is not supported on L2 devices
- TCL works on all platforms
- TCL has slightly different versions and different libraries

TCL

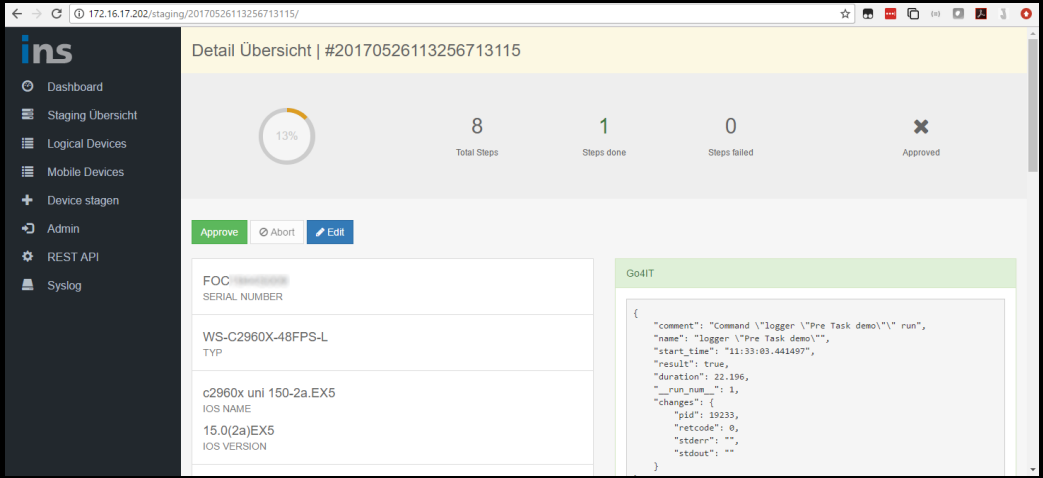
- More than 900 lines of code
- Around 100 ”if” statements
- Aproxamitly 50 cold showers

How it looked - Job Overview



Source: SwiNOG #31 Network Automation – Road trip to an automated Network

How it looked - Job Details



Source: SwiNOG #31 Network Automation – Road trip to an automated Network

SaltStack

- Using event bus
- Provides API
- Over time way too many workarounds

```
{% set printlabel = salt['pillar.get']('printlabel') %}
{% if printlabel in ['8 mm', '18 mm', '16 mm'] %}
label_print:
  cmd.run:
    - description: Print label
    - name: print_label.py {{ printer_ip }} 1 "{{ salt['pillar.get']('hostname','') }}"
{% endif %}
```

Awesome Features

- Hierarchical Network-Domain variables
- Generate WebForm from Template variables
- Prepare Device Management with SaltStack
- Templates and template snippets

Generate Form from Template

- **Parameter**

```
hostname: testname
ntp:
  - time0.ins.hsr.ch
  - 152.96.120.53
```

- **Template**

```
!
hostname {{ hostname }}
domain-name {{ domain_name|default('lab') }}
!{% for server in ntp %}
ntp server {{ server }}
!{% endfor %}
end
```

```
!
hostname testname
domain-name lab
!
ntp server time0.ins.hsr.ch
ntp server 152.96.120.53
!
end
```

Source: SwiNOG #31 Network Automation – Road trip to an automated Network

”Emergency” OS upgrade

We hit a dot1x bug and need to upgrade 5000 switches ASAP.
Could the ”Staging Robot” do that?

Customer A (2017)

- Own implementation of Plug&Play
- UI with CSV export/import to select a time window
- Slow start

Redesing UI

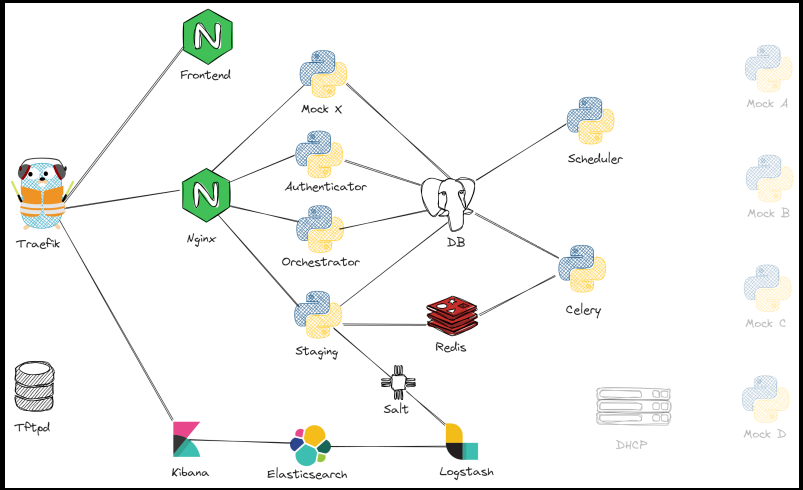
- Use synergies from other internally developed tools
- SinglePage with react
- Integrate multiple backends in the same UI

Customer B

We have a huge rollout upcoming. Could your tool be adapted?

Customer B (2019)

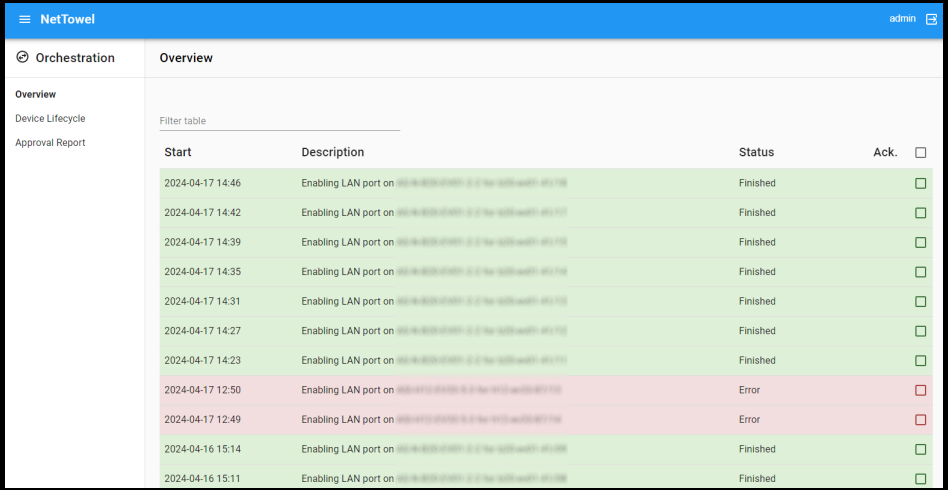
Architecture



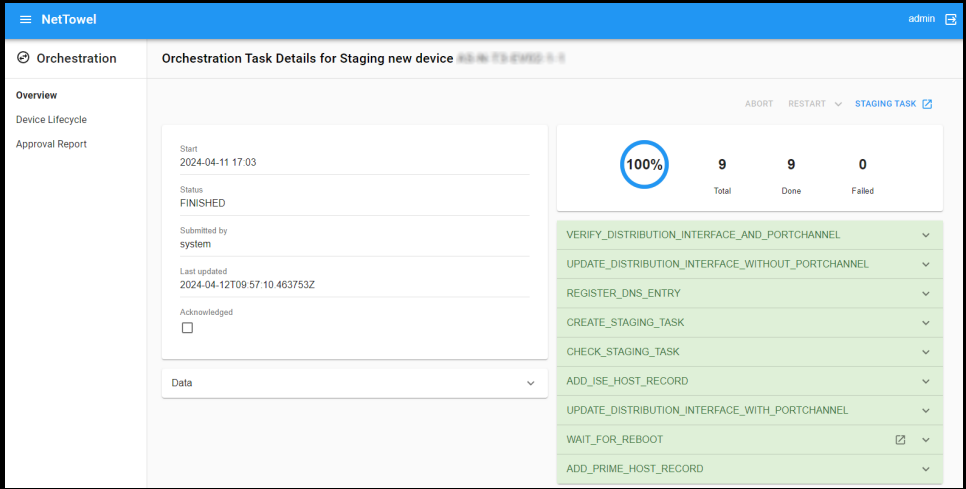
Addons

- 1:1 Replace Device
- Replacement with a new device
- Replace with Stack
- Interface rearrangement
- Provision/Deprovison Access port (API triggered from ordering system)

Orchestrator



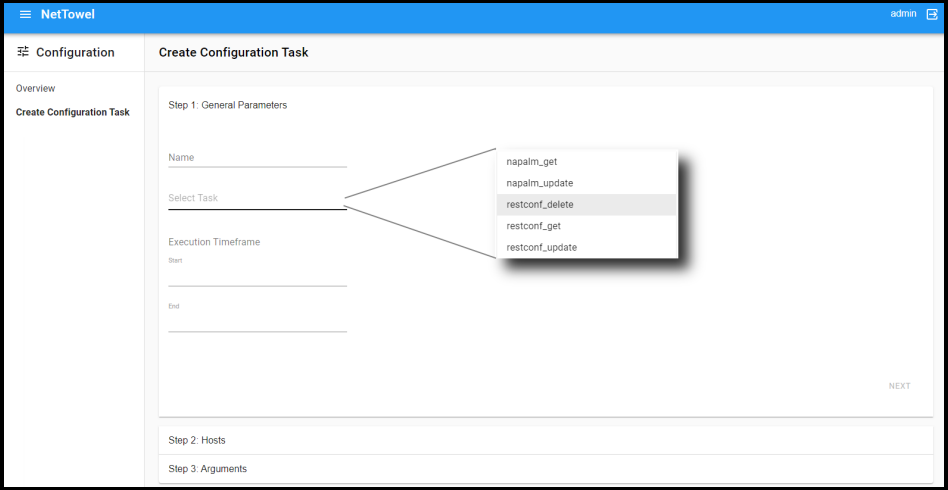
Orchestrator



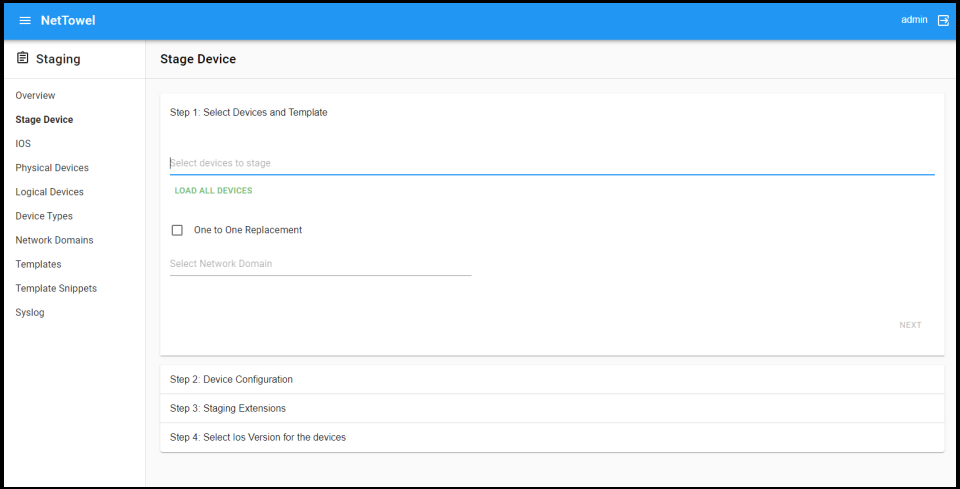
DiY or not?

- Exactly what you need vs general
- Resource-intensive vs locking
- Dependencies
- Integration

Configuration Task



Staging


NetTowel
admin
Staging
Overview
Stage Device
IOS
Physical Devices
Logical Devices
Device Types
Network Domains
Templates
Template Snippets
Syslog
Stage Device
Step 1: Select Devices and Template
Select devices to stage
LOAD ALL DEVICES
One to One Replacement
Select Network Domain
NEXT
Step 2: Device Configuration
Step 3: Staging Extensions
Step 4: Select Ios Version for the devices

Some numbers

Customer A

- Used the Staging Robot until mid-2023
- Staged around 9’000 L2/L3 devices
- Around 5’000 PnP OS Upgrades

Customer B

- In production
- Staged around 700 L3 Access Switches
- Around 1’500 workflows executed

Using Open Source

- Be prepared to debug the code
- Avoid basing on forks
- Try to be up to date
- Be prepared to overtake small projects
- Version pinning and testing is key

Key Points

- Don’t be afraid of failing
- Do not over-engineer
- Who can maintain this in 5 years?
- Does the customer need this feature or do you want it?
- No shortcuts
- ”Just” maintaining needs time
- Engineer fluctuation

Why do I say the project has failed?

- No direct vendor locking but engineer locking
- No or outdated documentation
- SaltStack is a central component and hard to replace/update
- Settings are in many different locations (because of quick wins)
- Project had multiple contributors but only one has the global view

Questions?