CNaaS NMS Training

CNaaS-NMS

1. Intro: Why, what
   a. Zero-touch provision
   b. Config management
   c. Firmware upgrade
2. Operations: How to operate
   a. Git repositories
   b. Workflows
   c. ZTP
   d. Interfaces
3. Internals & Troubleshooting: When something goes wrong
   a. Containers, processes
   b. Databases
4. Integration & Development

Operations

Git repositories:

- templates: OS specific CLI templates written in Jinja2 (.j2 file extension)
- settings: OS independent settings written in YAML (.yml file extension)
  - NTP, RADIUS, syslog servers
  - VXLANs/SVIs, VRFs and routing
  - Core/Dist interfaces
- etc: OS config files
  - isc-dhcpd config for ZTP

Templates, access.j2 example

```
{% for intf in interfaces %}
interface {{ intf.name }}
{# -- ACCESS AUTO -- #}
{% if intf.ifclass == 'ACCESS_AUTO' %}
  {% if (intf.data.description is defined) and intf.data.description %}
 description {{ intf.data.description }}
  {% else %}
 description DOT1X
  {% endif %}
 poe reboot action maintain
 switchport
 switchport mode access
 storm-control broadcast level 7
 spanning-tree bpduguard enable
 spanning-tree portfast edge
 dot1x pae authenticator
 dot1x authentication failure action traffic allow vlan {{ dot1x_fail_vlan }}
 dot1x port-control auto
 dot1x mac based authentication
 {% if (intf.data.bpdu_filter is defined) and intf.data.bpdu_filter %}
 spanning-tree bpdufilter enable
 {% endif %}
 {% include 'access-tags.j2' %}
```

Settings, vxlans.yml

```
---
vxlans:
  student1:
    vni: 100500
    vrf: STUDENT
    vlan_id: 500
    vlan_name: STUDENT
    ipv4_gw: 10.200.1.1/24
    groups:
      - ALL_DEVICES
```

Indentation with spaces is important!

API, device/<>/generate_config

```
"available_variables": {
  "dhcp_relays": [
    {
      "host": "10.100.2.2"
    }
  ],
  "interfaces": [
    {
      "name": "Ethernet1",
      "ifclass": "ACCESS_TAGGED",
      "untagged_vlan": 500,
      "tagged_vlan_list": [
        500,
        501
      ],
...
```

Applying a change

1. Edit settings/templates repo
2. Git commit/push
3. Refresh settings/templates API call
4. Syncto dry_run API call, verify diff
5. Syncto live run API call

For access interface config update:

Update interface config API call -> dry_run -> live run

NMS Change Workflow

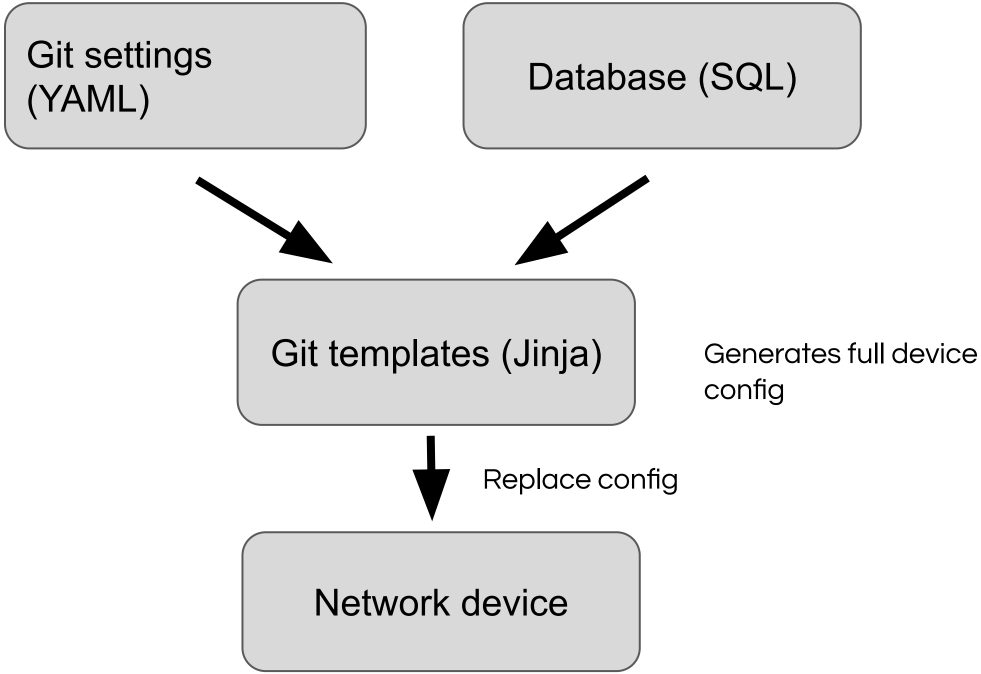
In local editor/platform WebUI

A. Update settings (YAML) or templates (Jinja2)
B. Commit and push to git repository

Via API / WebUI

1. Ask NMS-server to pull changes from git
2. Dry run on devices
3. Verify diff output
4. Deploy change (live run)

Config rendering



Git settings (YAML)

Database (SQL)

Git templates (Jinja)

Generates full device config

Replace config

Network device

Commit confirm modes

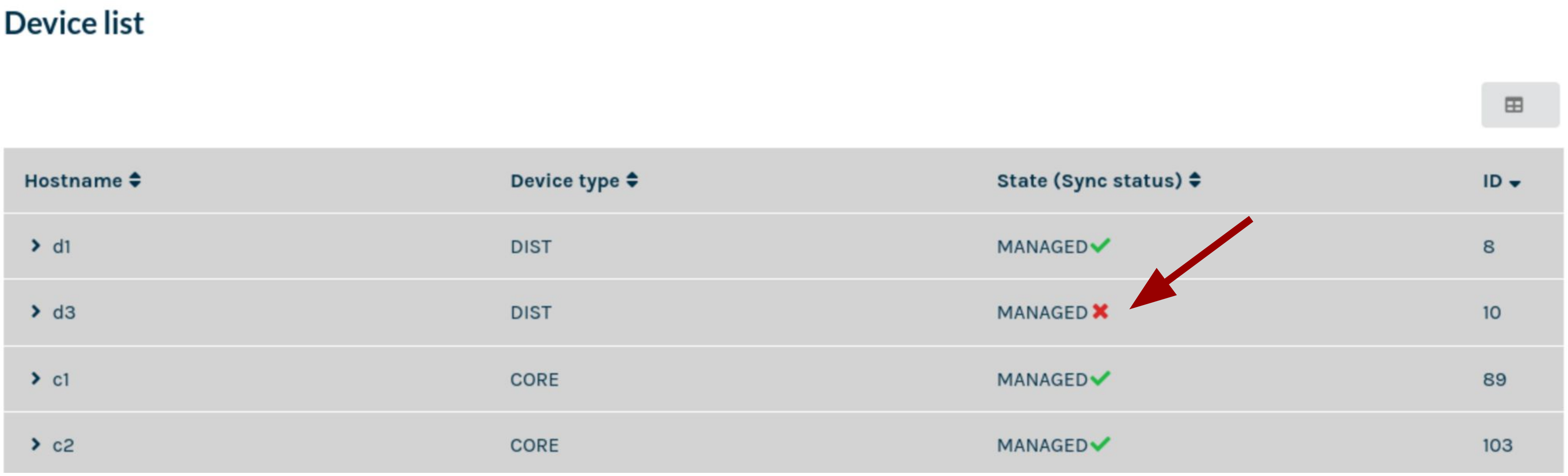
Mode 0 “no confirm”: deploy change without confirm timer

Mode 1 “per-device”: deploy change with commit timer, if device is unreachable after commit rollback only the device that was unreachable

Mode 2 “per-job”: deploy change with commit timer, if any device in job fails rollback all devices to previous configuration. Limited to 50 devices per job

Device synchronization

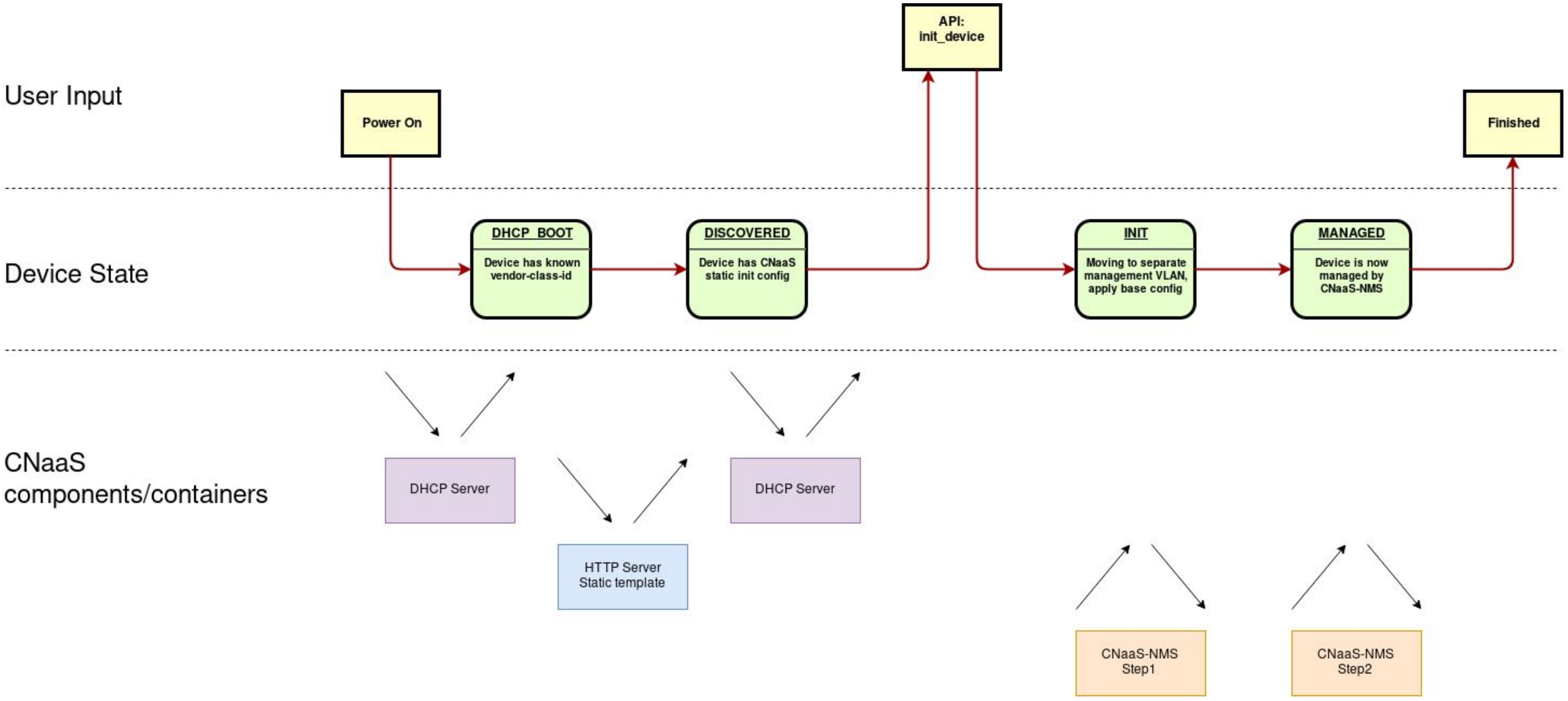
https://wiki.sunet.se/display/CNaaS/CNaaS+NMS+Synchronization



Device list

Hostname	Device type	State (Sync status)	ID
› d1	DIST	MANAGED ✓	8
› d3	DIST	MANAGED ✗	10
› c1	CORE	MANAGED ✓	89
› c2	CORE	MANAGED ✓	103

ZTP workflow



User Input
Device State
CNaaS components/containers
Power On
API: init_device
Finished
DHCP_BOOT
Device has known vendor-class-id
DISCOVERED
Device has CNaaS static init config
INIT
Moving to separate management VLAN, apply base config
MANAGED
Device is now managed by CNaaS-NMS
DHCP Server
HTTP Server Static template
DHCP Server
CNaaS-NMS Step1
CNaaS-NMS Step2

ZTP prerequisites

1. Pair of dist-switches with management domain (VLAN + IP Gateway)
2. Ifclass downlink interfaces configured on dist
3. ZTP vlan (vlan 1) configured on dist, DHCP relay to NMS
4. DHCP scope configured on NMS DHCPd
5. Redundancy requirements met for cabling, or redundant_link: false

User interfaces

1. WebUI - Used to: sync settings/templates, device list, ZTP, jobs, firmware upgrade, access port config
2. CLI - Same as WebUI plus linknets
3. API CURL/Postman etc - Everything (template vars, re-init step 2, update physical interfaces, update linknets)
4. (NAV - Access port config)

WebUI demo! 🪄

Internals, Nornir/NAPALM

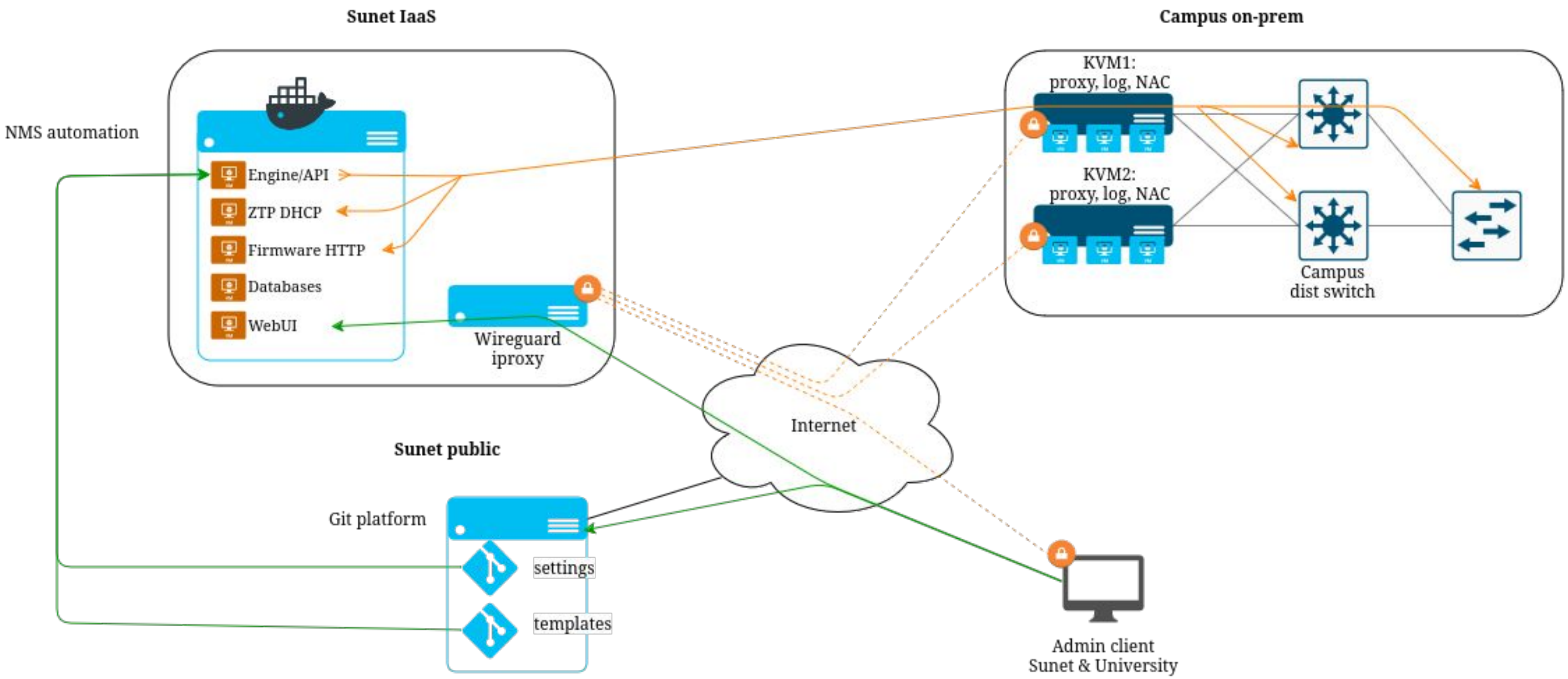
Nornir is used to parallelize tasks (50 threads), each task runs NAPALM

NAPALM is used to talk to network devices
NAPALM is an abstraction layer that uses vendor-specific APIs like pyeapi to talk to different devices

Each vendor OS is responsible for calculating diff of configs and replacing running config with new config

Config is always fully replaced, never merged

NMS communication



Local changes

Configuration hash is generated after new config is sent to device

Before doing dry run the previous configuration hash is compared to new config hash, if mismatch you get an error

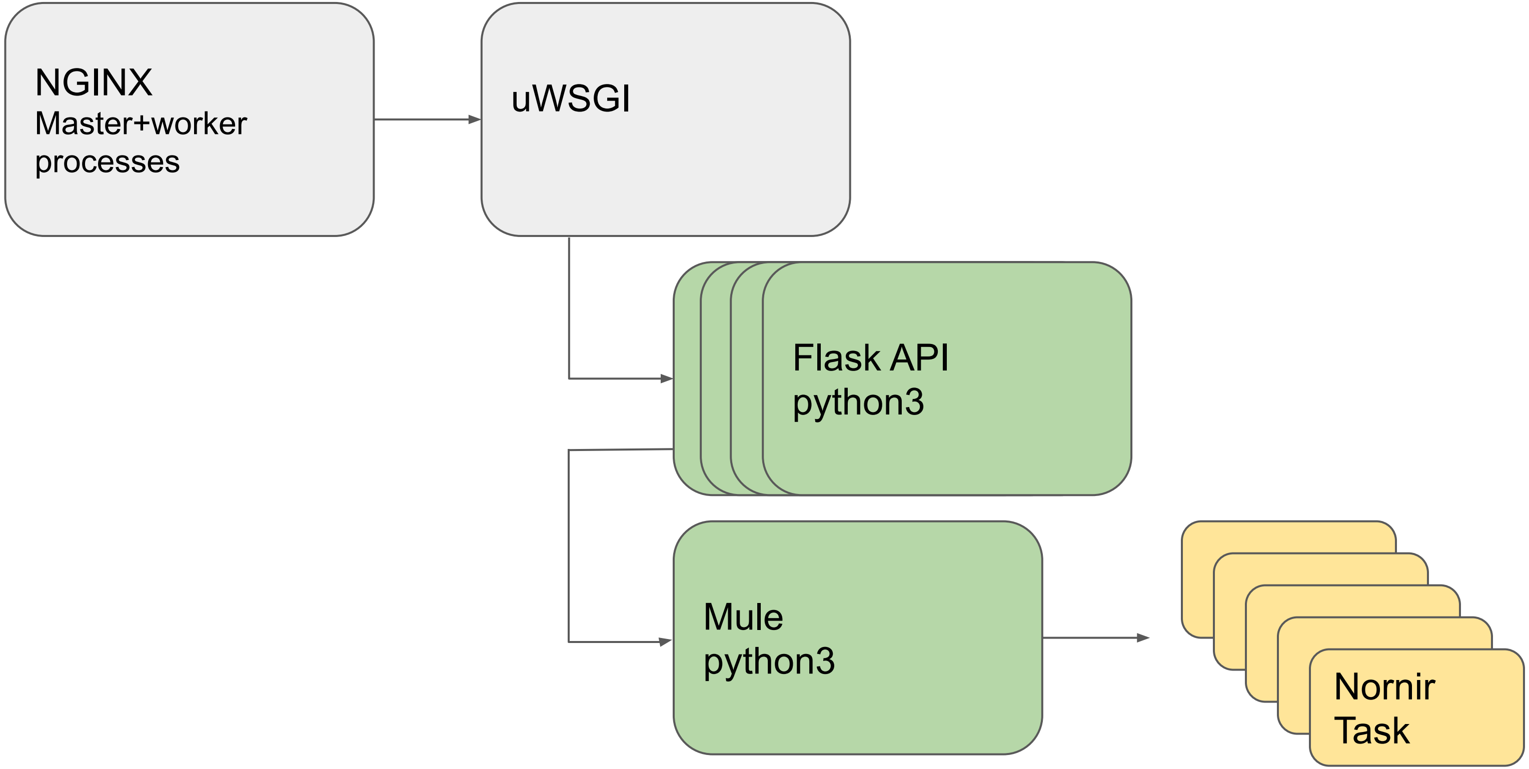
If you want to overwrite local changes you have to syncto with force: true

If a device will have local changes for a bit change it to state UNMANAGED in NMS

Internals, containers

1. API, running python source code for CNaaS-NMS
2. PostgreSQL, SQL database. API connects here via TCP 5432
3. Redis, in-memory key-value database. API connects here via TCP 6379
4. DHCPd, isc-dhcpd used for ZTP boot. Switch management connects here via UDP 67
5. HTTPd, nginx for serving static files like firmwares and initial static config

Internals, processes of API



Internals, databases

1. PostgreSQL, on-disk persistent
   a. CNaaS-NMS tables defined in Python code using SQLAlchemy ORM
   b. APScheduler tables for keeping track of future scheduled jobs
   c. Alembic database schema version tracking
2. Redis, in-memory volatile
   a. Cache for currently working/finished devices during job run
   b. Cache for settings parsed from settings git repo

Internals, locking

Syncto job requires global “all-devices” lock

Refresh settings/templates requires global “all-devices” lock

-> it’s not possible run two syncto jobs in parallell, instead run one job which includes all the devices you want to sync

Integration / customization

API user with client credentials flow, CLIENT_ID and CLIENT_SECRET

API configuration settings

settings_override

Plugin hooks: new managed device