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. API commands
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

Operations, access.j2 example

```
{% extends "managed-full.j2" %}
{% block interfaces %}
        {% for intf in interfaces %}
interface {{ intf.name }}
                {% if intf.ifclass == 'ACCESS_UPLINK' %}
                        {% set description = 'UPLINK' %}
 switchport
 switchport mode trunk
 channel-group 1 mode active
                {% elif intf.ifclass == 'ACCESS_AUTO' %}
                        {% set description = 'DOT1X' %}
 switchport
 switchport mode access
 switchport access vlan 10
...
```

Operations, vxlans.yml

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

Operations, device/<>/generate_config API


```
"available_variables": {
  "dhcp_relays": [
    {
      "host": "10.100.2.2"
    }
  ],
  "interfaces": [
    {
      "name": "Ethernet1",
      "ifclass": "ACCESS_UNTAGGED",
      "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

User interfaces

1. WebUI - Used to: sync settings/templates, check devices, jobs
2. NAV - Access port config
3. CURL/Postman etc - Everything
4. CLI? - ?
5. Setup script? - Initial setup

Internals, Nornir/NAPALM

Nornir is used to parallelize tasks, 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

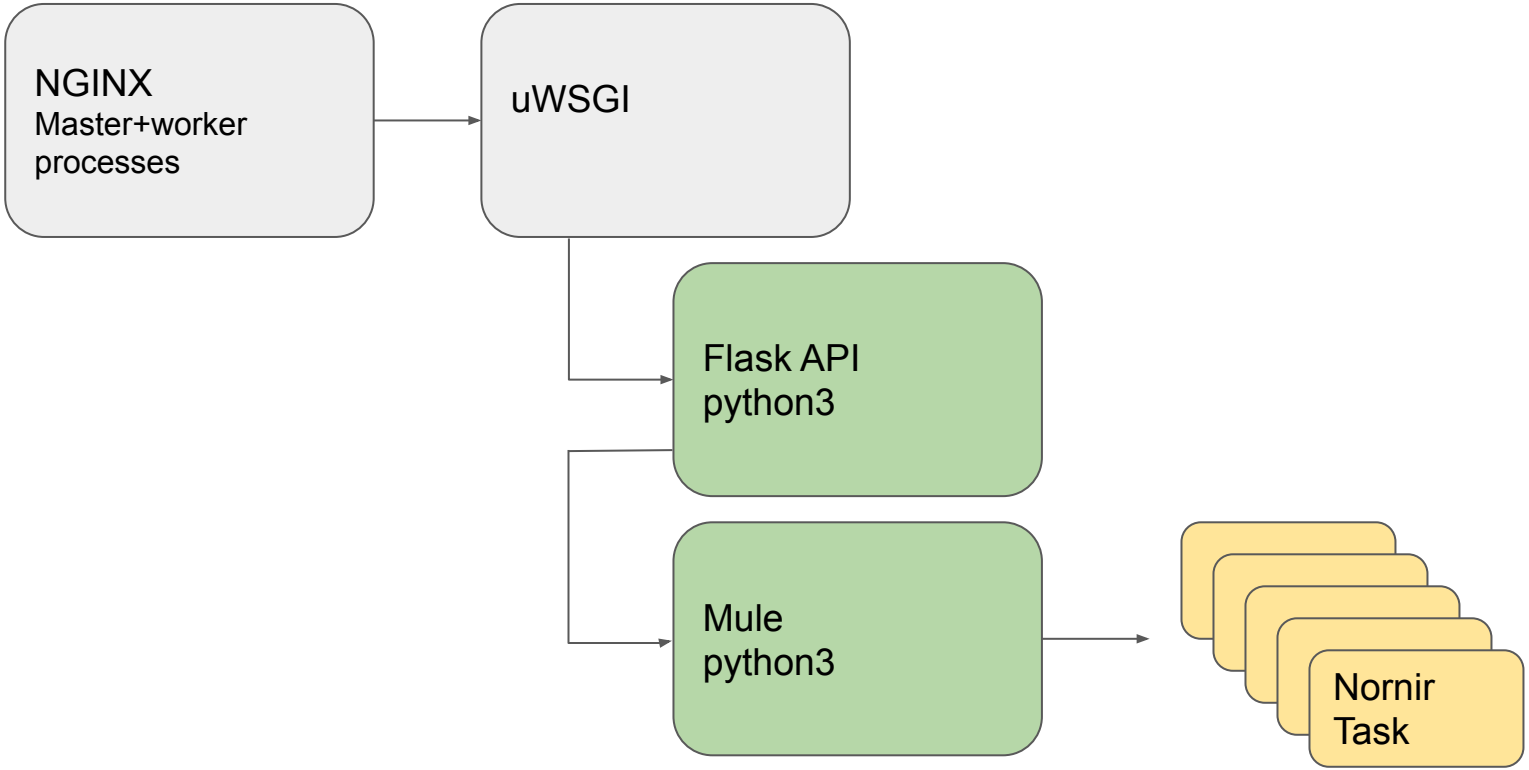
Each vendor OS is responsible for calculating diff of configs and replacing running config with new config

Config is always fully replaced, never merged

Internals, containers

1. API, running python sourcecode 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,

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