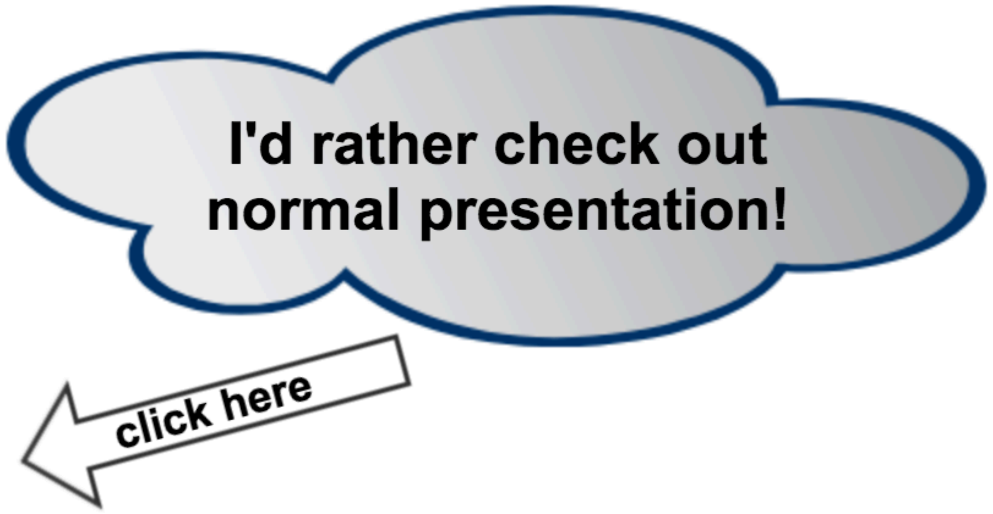
Deep dive: using Ansible to automate Network Operations

(by Yuri Kretov, Sr. Network Engineer)

OpenTable®

Normal people - exit now!



https://github.com/opentable/ansible-examples/blob/master/Ansiblefest2017/OT_case_study.pdf

https://github.com/opentable/ansible-examples/blob/master/Ansiblefest2017/OT_deep_dive.pdf

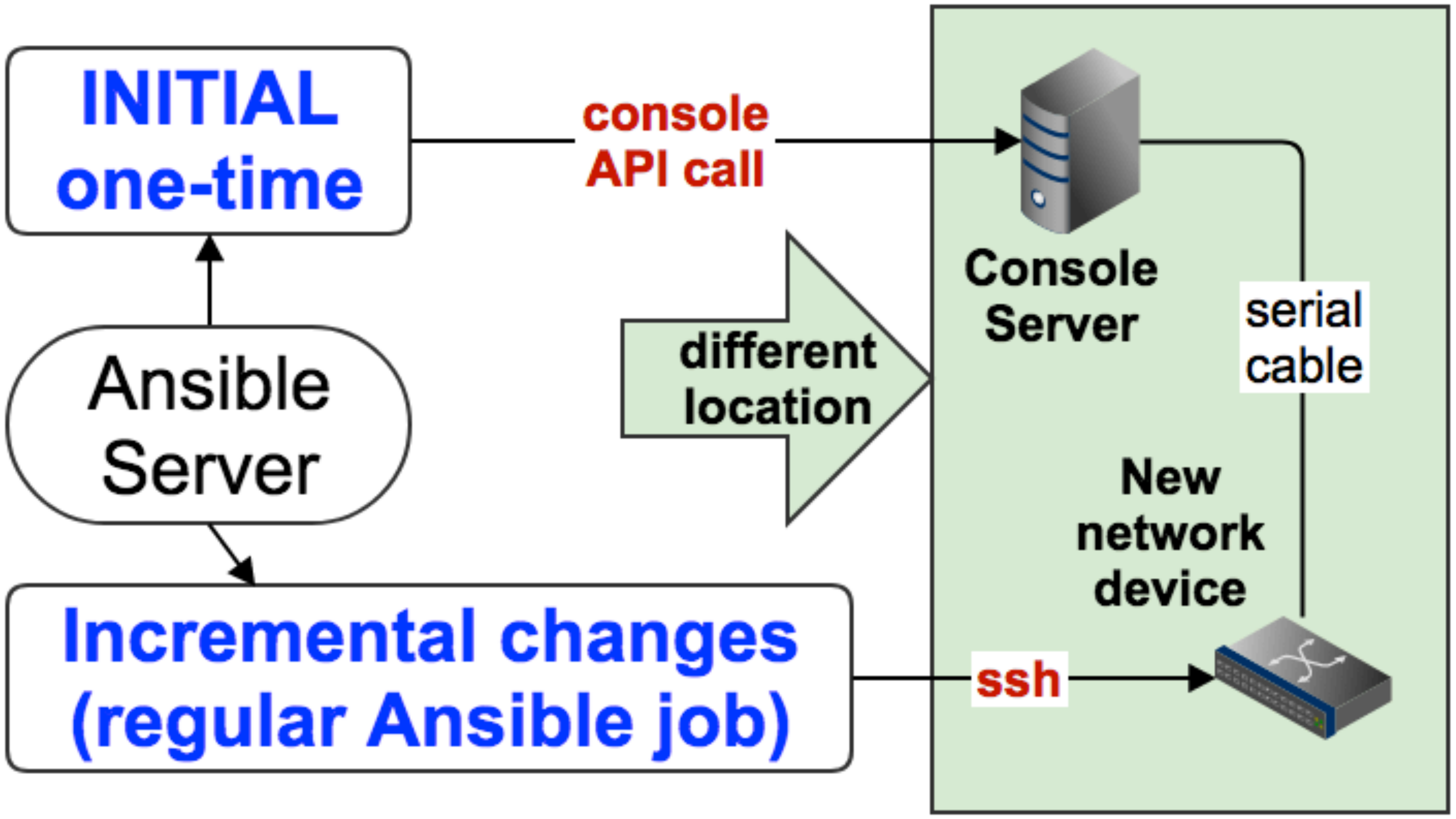
Four technical things to talk about

- OPs: two-step initial provisioning
- OPs: challenge to replace config
- Dev: Ansible dynamic inventory
- Dev: Reusable code patterns

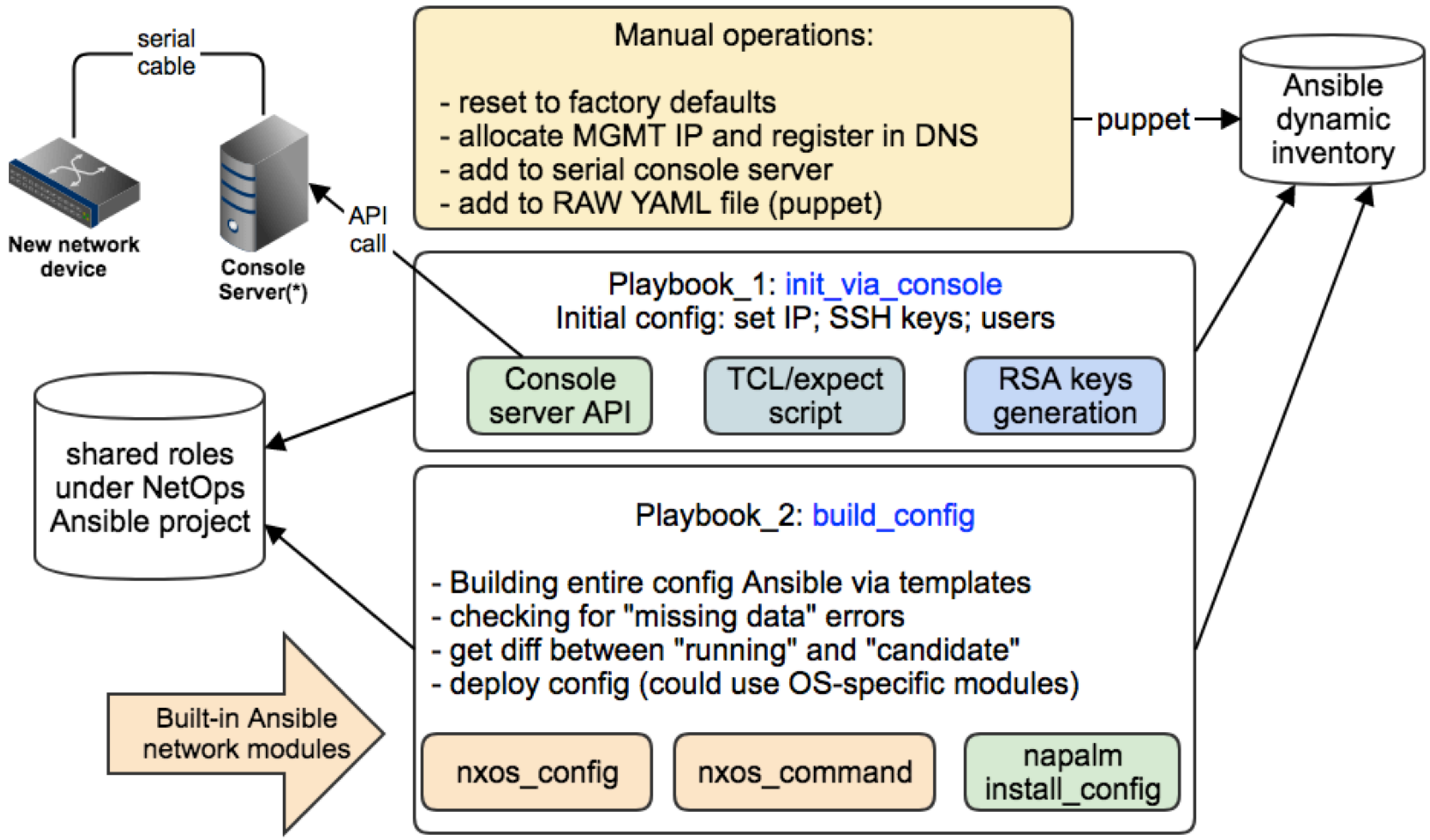
1. Two-step initial provisioning

New network devices:

- consistent & fast provisioning

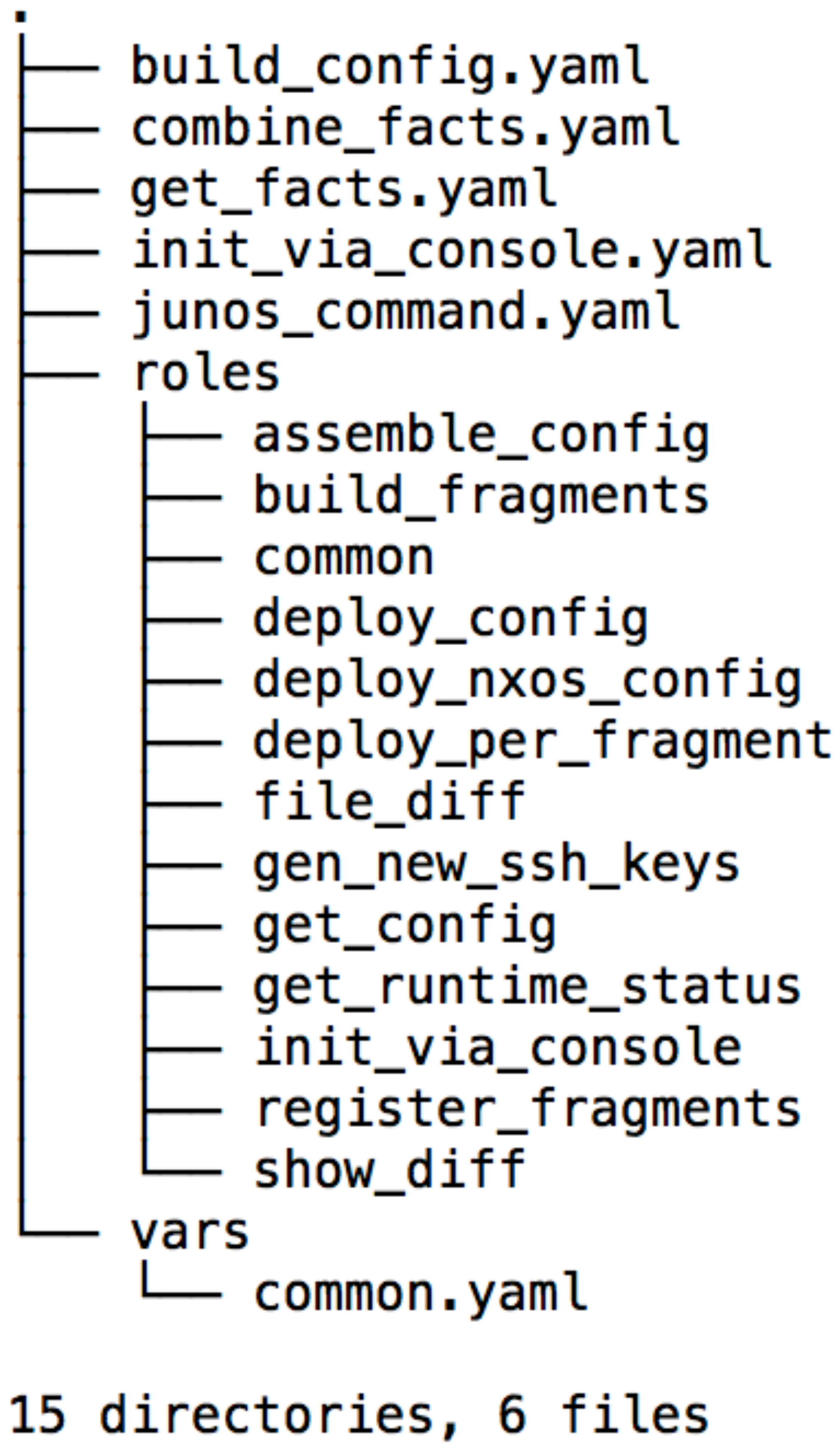


More details, please :)



serial cable
New network device
Console Server(*)
API call
Manual operations:
- reset to factory defaults
- allocate MGMT IP and register in DNS
- add to serial console server
- add to RAW YAML file (puppet)
puppet
Ansible dynamic inventory
Playbook_1: init_via_console
Initial config: set IP; SSH keys; users
Console server API
TCL/expect script
RSA keys generation
shared roles under NetOps Ansible project
Playbook_2: build_config
- Building entire config Ansible via templates
- checking for "missing data" errors
- get diff between "running" and "candidate"
- deploy config (could use OS-specific modules)
Built-in Ansible network modules
nxos_config
nxos_command
napalm install_config

ops-ansible/playbooks/netops



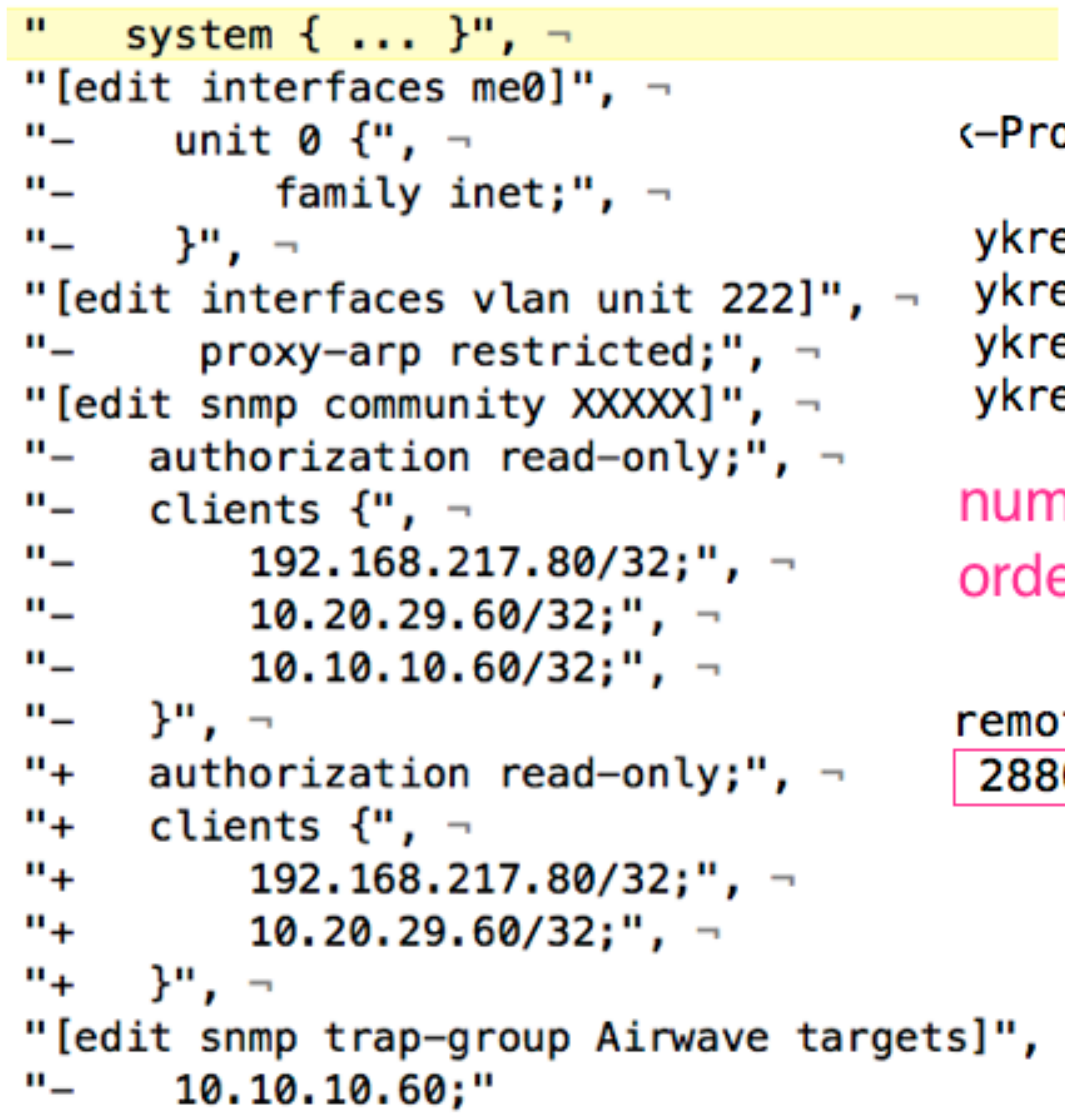
```
.
├── build_config.yaml
├── combine_facts.yaml
├── get_facts.yaml
├── init_via_console.yaml
├── junos_command.yaml
├── roles
│   ├── assemble_config
│   ├── build_fragments
│   ├── common
│   ├── deploy_config
│   ├── deploy_nxos_config
│   ├── deploy_per_fragment
│   ├── file_diff
│   ├── gen_new_ssh_keys
│   ├── get_config
│   ├── get_runtime_status
│   ├── init_via_console
│   ├── register_fragments
│   └── show_diff
└── vars
    └── common.yaml

15 directories, 6 files
```

Roles & Playbooks

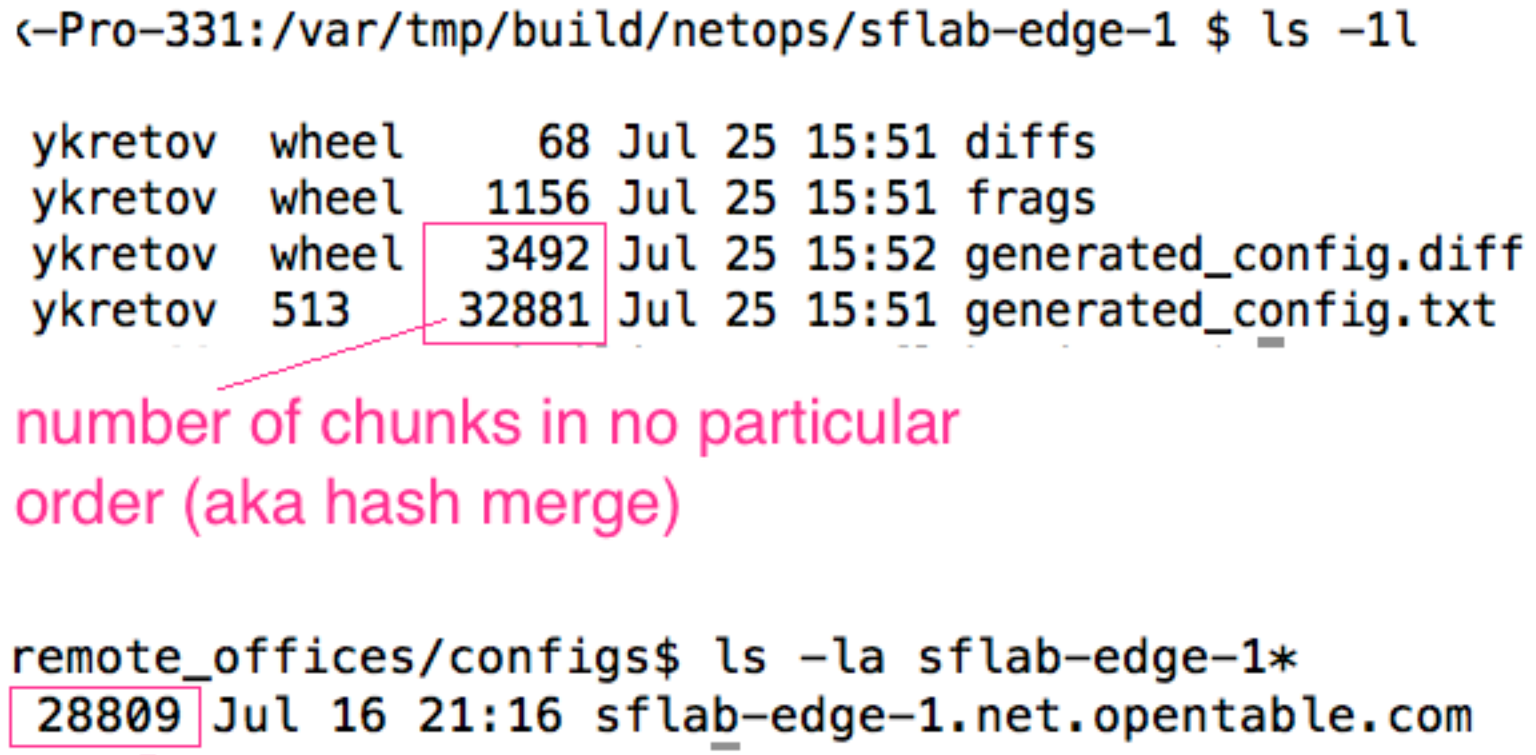
2. In-place config changes

Config diff in JunOS (easy)

ykretov@sc-sw-30-MOCK# show | diff



```
"   system { ... }", ¬
"[edit interfaces me0]", ¬
"-    unit 0 {", ¬
"-        family inet;", ¬
"-    }", ¬
"[edit interfaces vlan unit 222]", ¬
"-     proxy-arp restricted;", ¬
"[edit snmp community XXXXX]", ¬
"-   authorization read-only;", ¬
"-   clients {", ¬
"-       192.168.217.80/32;", ¬
"-       10.20.29.60/32;", ¬
"-       10.10.10.60/32;", ¬
"-   }", ¬
"+   authorization read-only;", ¬
"+   clients {", ¬
"+       192.168.217.80/32;", ¬
"+       10.20.29.60/32;", ¬
"+   }", ¬
"[edit snmp trap-group Airwave targets]",
"-    10.10.10.60;"
```



```
<-Pro-331:/var/tmp/build/netops/sflab-edge-1 $ ls -1l

ykretov  wheel     68 Jul 25 15:51 diffs
ykretov  wheel   1156 Jul 25 15:51 frags
ykretov  wheel   3492 Jul 25 15:52 generated_config.diff
ykretov  513    32881 Jul 25 15:51 generated_config.txt
```

number of chunks in no particular order (aka hash merge)

```
remote_offices/configs$ ls -la sflab-edge-1*
 28809 Jul 16 21:16 sflab-edge-1.net.opentable.com
```

Actual config from rancid

Simulate live config diff

(Cisco IOS, Nexus - hard, but “automatable”)

```
1. Capture running config
   from original device
```

```
2. Generate full config for
   that device via Ansible
```

```
3. Adjust generated config
slightly (MGMT IP + static
routing + MGMT vrf)
```

```
4. Reset LAB device
   (similar h/w & s/w) to
   factory defaults
```

```
5. Initialize LAB device
   via Console (init config)
```

```
6. Deploy generated config
   to LAB device instead
   of original device
```

```
7. Capture running config
   from LAB device
```

```
8. Make a diff between
   original running config
   and LAB running config
```

3. Ansible dynamic inventory

Make a script to call Web server (really easy)

```
curl http://localhost:8081/inventory | jq '.'
```

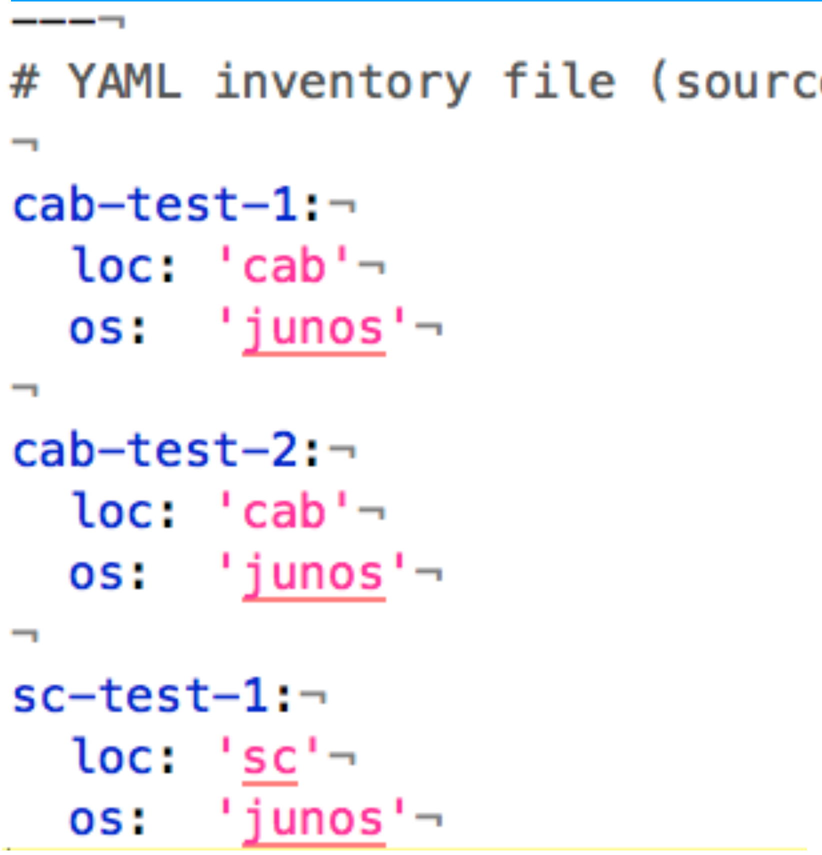
```
{
  "all": [
    "cab-test-1",
    "cab-test-2",
    "sc-test-1"
  ],
  "cab": [
    "cab-test-1",
    "cab-test-2"
  ],
  "sc": [
    "sc-test-1"
  ],
  "junos": [
    "cab-test-1",
    "cab-test-2",
    "sc-test-1"
  ],
  "_meta": {
    "hostvars": {
      "cab-test-1": {
        "serial_number": "SN_123"
      }
    }
  }
}
```

~/ops-ansible/inventory/test.rb

```
#!/usr/bin/env ruby¬
¬
require 'open-uri'¬
¬
def get_list¬
  begin¬
    file = open("http://localhost:8081/inventory")
  rescue¬
    '{}'¬
  else¬
    file.read¬
  end¬
end¬
¬
if ARGV[0] == '--list'¬
  puts get_list¬
elsif ARGV[0] == '--host'¬
  puts '{}'¬
end¬
```

What kind of script to write

webserver/source_of_truth.yaml



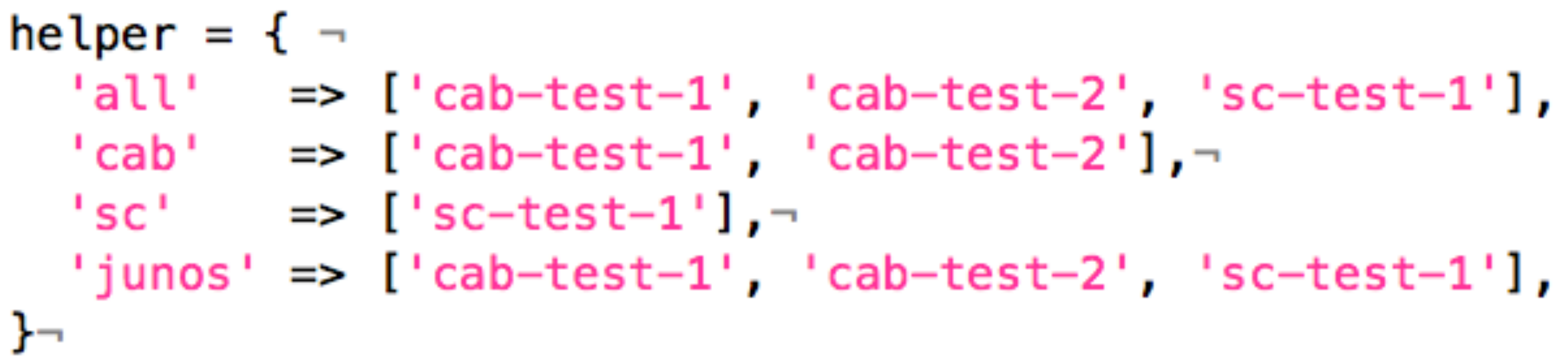
```
---¬
# YAML inventory file (source of truth)
¬
cab-test-1:¬
  loc: 'cab'¬
  os:  'junos'¬
¬
cab-test-2:¬
  loc: 'cab'¬
  os:  'junos'¬
¬
sc-test-1:¬
  loc: 'sc'¬
  os:  'junos'¬
```

1. read source YAML file
2. produces dict with keys as **group** names and values as **lists** of hostnames

inventory/group_vars/

```
junos-ex.yaml
junos-qfx.yaml
junos-srx.yaml
junos.yaml
sc-junos.yaml
sc-nxos.yaml
sc.yaml
```

webserver/webserver_engine.rb

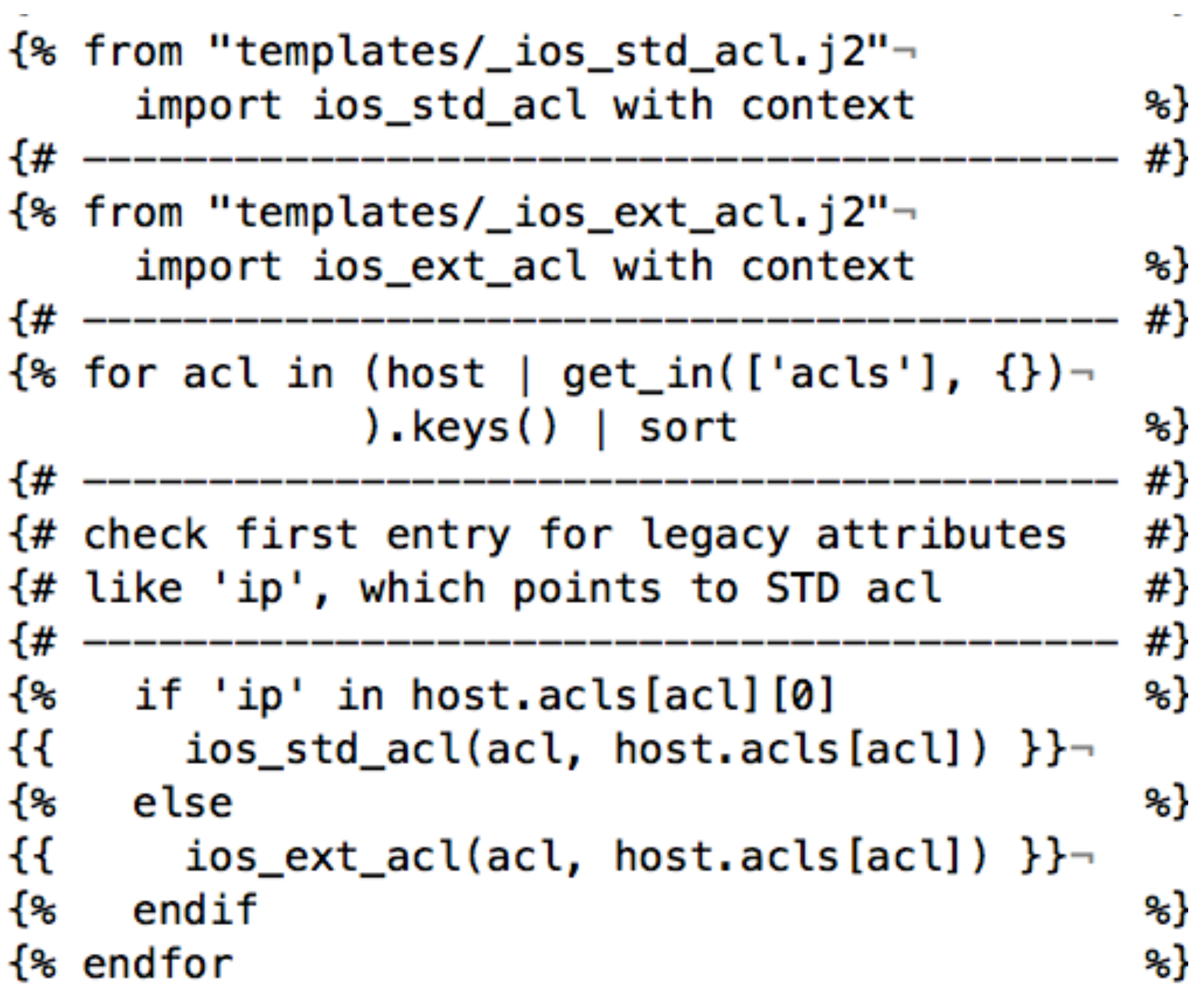


```
helper = { ¬
  'all'   => ['cab-test-1', 'cab-test-2', 'sc-test-1'],
  'cab'   => ['cab-test-1', 'cab-test-2'],¬
  'sc'    => ['sc-test-1'],¬
  'junos' => ['cab-test-1', 'cab-test-2', 'sc-test-1'],
}¬
```

4. Reusable code patterns

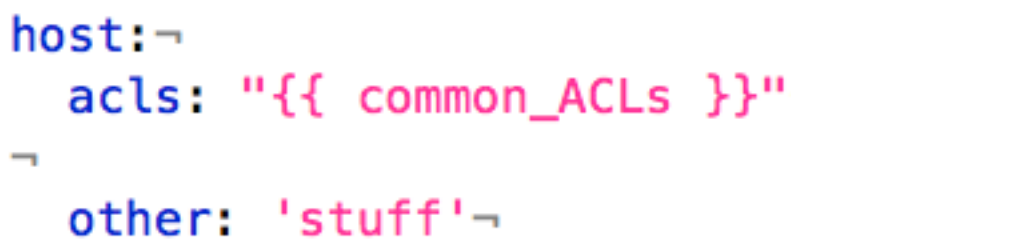
Reusable jinja2 templates

roles/build_fragments/templates/acl/acl_ios.j2



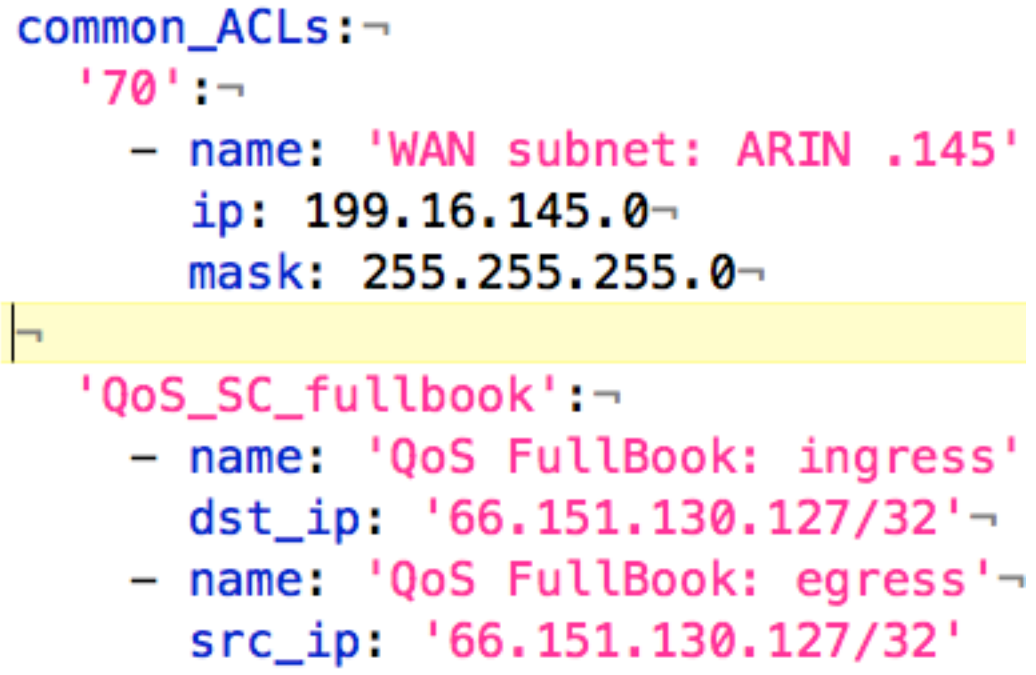
```
{% from "templates/_ios_std_acl.j2"¬
     import ios_std_acl with context              %}
{# ---------------------------------------- #}
{% from "templates/_ios_ext_acl.j2"¬
     import ios_ext_acl with context              %}
{# ---------------------------------------- #}
{% for acl in (host | get_in(['acls'], {})¬
              ).keys() | sort                     %}
{# ---------------------------------------- #}
{# check first entry for legacy attributes     #}
{# like 'ip', which points to STD acl          #}
{# ---------------------------------------- #}
{%   if 'ip' in host.acls[acl][0]                 %}
{{     ios_std_acl(acl, host.acls[acl]) }}¬
{%   else                                         %}
{{     ios_ext_acl(acl, host.acls[acl]) }}¬
{%   endif                                        %}
{% endfor                                         %}
```

inventory/host_vars/tmp-sw-01.yaml



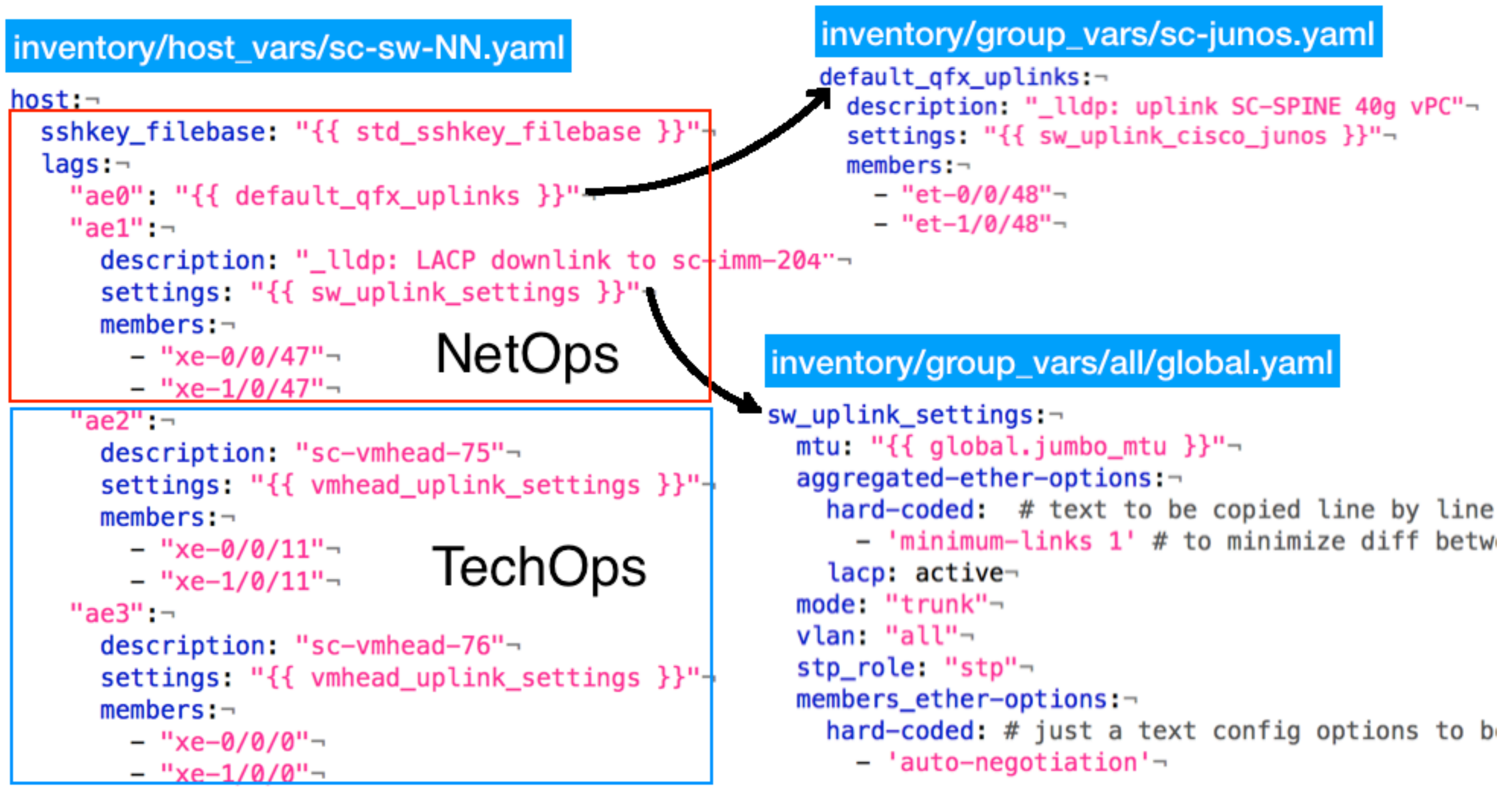
```
host:¬
  acls: "{{ common_ACLs }}"
¬
  other: 'stuff'¬
```

inventory/group_vars/all/acls.yaml



```
common_ACLs:¬
  '70':¬
    - name: 'WAN subnet: ARIN .145'
      ip: 199.16.145.0¬
      mask: 255.255.255.0¬
¬
  'QoS_SC_fullbook':¬
    - name: 'QoS FullBook: ingress'
      dst_ip: '66.151.130.127/32'¬
    - name: 'QoS FullBook: egress'¬
      src_ip: '66.151.130.127/32'
```

Reusable YAML definitions



inventory/host_vars/sc-sw-NN.yaml

```
host:¬
  sshkey_filebase: "{{ std_sshkey_filebase }}"¬
  lags:¬
    "ae0": "{{ default_qfx_uplinks }}"¬
    "ae1":¬
      description: "_lldp: LACP downlink to sc-imm-204"¬
      settings: "{{ sw_uplink_settings }}"¬
      members:¬
        - "xe-0/0/47"¬
        - "xe-1/0/47"¬
    "ae2":¬
      description: "sc-vmhead-75"¬
      settings: "{{ vmhead_uplink_settings }}"¬
      members:¬
        - "xe-0/0/11"¬
        - "xe-1/0/11"¬
    "ae3":¬
      description: "sc-vmhead-76"¬
      settings: "{{ vmhead_uplink_settings }}"¬
      members:¬
        - "xe-0/0/0"¬
        - "xe-1/0/0"¬
```

inventory/group_vars/sc-junos.yaml

```
default_qfx_uplinks:¬
  description: "_lldp: uplink SC-SPINE 40g vPC"¬
  settings: "{{ sw_uplink_cisco_junos }}"¬
  members:¬
    - "et-0/0/48"¬
    - "et-1/0/48"¬
```

inventory/group_vars/all/global.yaml

```
sw_uplink_settings:¬
  mtu: "{{ global.jumbo_mtu }}"¬
  aggregated-ether-options:¬
    hard-coded:  # text to be copied line by line
      - 'minimum-links 1' # to minimize diff betw
    lacp: active¬
  mode: "trunk"¬
  vlan: "all"¬
  stp_role: "stp"¬
  members_ether-options:¬
    hard-coded: # just a text config options to b
      - 'auto-negotiation'¬
```

That was not nerdy...
Only four examples,
I want more!!!

https://github.com/opentable/ansible-examples

click here

In that case check out
this repo for more examples

Thanks for watching!

https://github.com/opentable/ansible-examples

https://github.com/opentable/ansible-examples

OpenTable, 2017
Yuri Kretov, Sr. Network Engineer
ykretov@opentable.com