Funet Kampus Configuration Automation

14.11.2019 Asko Hakala
Configuration and monitoring

• All router configuration is done using Ansible and Jinja2 templates
  - Based on tools developed for Funet 2020
  - Clean, standardized configuration
  - Routers can be pre-configured before sending them to customers
  - Configuration is stored in YAML files

• Routers and switches are added to monitoring using Ansible during the provisioning
  - Same alert and monitoring tools and processes are used as for the rest of the Funet network
Legacy configuration tools

• Task specific tools
  o Peering filters (as-path, prefix-lists) update and configuration
  o MPLS VPN service provisioning
  o Common configuration

• Existing tools like scripts (Perl, Expect, Bash..) and Ansible-playbooks for specific purposes
Funet 2020 automation goals

• Simple provisioning of new routers
• Consistent configuration across the network
• Standardized services and easy provisioning of new customer connections and services
Partial vs. full automation

• Initial idea was to automate “most” configuration and do the rest by hand

• However, having partially automated and partially manually maintained configuration is awkward
  - Possible conflicts between automatically and manually generated configs
  - How to remove elements from the configuration, if the whole configuration is not replaced?
  - With manual configuration the configurations would deteriorate by time
Full automation

• Always re-generating the entire configuration and then overwrite the entire running config for each router
  o The configuration includes only the needed elements
  o No need for separate garbage collection
  o Manual hacks will simply get destroyed

• Possible because the new network is built from the scratch – no configuration copied from the old network
Tool: Ansible and Jinja2 templates

- Ansible has been used for server automation
- JunOS has a good Ansible support, e.g. existing modules
- Router configuration generated from Jinja2 template
- Ansible is used for IP/MPLS network as a template engine
- The template-generated config could be loaded to routers with other tools if needed
  - For now the configs are also loaded to routers with Ansible, as it provides routines and error handling for that
Data model

- Owndata-model, formed by iteration
  - Most variables have default values in template and can be overridden in Ansible variables e.g. interface MTUs

```yaml
interfaces:
- name: xe-0/0/0:3
  description: funet2020_testlab-a
  units:
    - number: 1
      description: funet2020_testlab_internet-a
      ip_mtu: 9170
      ipv4_addresses:
        - address: 193.167.244.98/31
        - address: 2001:7001:0:10:0:0:0:0/127
      ipv6_addresses:
        - address: 2001:7001:0:10:0:fe08::2/127
    - number: 100
      vrf: funet-mgmt
      description: funet2020_testlab_mgmt-a
      ipv4_addresses:
        - address: 192.168.255.0/31

routing_instances:
- name: FUNET-MGMT
  import_communities:
    - name: MANAGEMENT
      accept_prefixes: [ FUNET-MANAGEMENT-NETWORKS ]
  bgp_groups_v4:
    - name: FUNET2020-TESTLAB
      peer_as: 65032
      role: primary
      accept_prefixes: [ TESTLAB1-SW1 ]
      export_prefixes: [ FUNET-MANAGEMENT-NETWORKS ]
      neighbors:
        - address: 192.168.255.1
          description: testlab1.ip.funet.fi
```
Initial Provisioning

• PoPs are equipped with a serial console server for OOB access, which are used for the initial commissioning
  o Only a few configuration commands to make a new router reachable to Ansible, and then the playbook does the rest
  o During the initial commissioning Ansible playbook is run via “backdoor”
  o SSH is tunneled through serial console server to router mgmt interface
  o In Ansible inventory an alternative host is defined so it is used instead of in-band SSH

• Remote hands are only needed for physical installation
Data

• The entire network configuration is in YAML files
  o Router specific configuration is defined in the router’s host variables
  o Common elements defined in shared vars files (group_vars for Kampus)
    o Prefix-lists
    o Firewall filters and policers
    o BGP communities and route-targets
    o Customer AS numbers

• Possible to template another tools using the same meta-data, e.g.:
  o Nagios configuration generated when routers are configured
  o Interface statistics links to Grafana dashboards
Exceptions

• It is possible that a service needs to be deployed before it has been incorporated into the Ansible template
  o Custom configuration can be added as a normal JunOS configuration snippet that will be automatically read into the template
  o Still, we should find a generalized configuration and incorporate it into the automation template
  o Custom snippet is meant to be only a temporary solution
Considerations 1/2

• Initial learning curve
  o Using Ansible and running playbooks
  o Defining configurations in YAML data model
  o Fixing things in the playbook and/or Jinja2 template

• Ansible is only a tool, it is still needed to know what one is doing
  o Recommended to run in “check mode” first and validate the diff

• Branching playbook to different networks
  o E.g. Funet core and Campus networks have different needs
  o Need of software developer skills
Considerations 2/2

• YAML data model documentation
  o Current work-around by an example file – needs updating

• Workflow
  o Version control conflicts – e.g. change committed to routers not committed in Git repository
Shared configuration pilot

• Managing large customer networks
  o Lot of changes -> workload

• Configuration changes by customer
  o All configuration is managed by Ansible – no manual changes are possible
  o No customer access or customer generated YAML files allowed to our management network
  o Customer specific Gitlab and Management server
    o Customer is allowed to edit host_vars and group_vars files
    o Changes merged manually by Funet
Case: Helsinki University

- MPLS core for 5 campuses
- 4 routers and 10 switches
- New Ansible features
  - MX 10003 support in Kampus
  - QFX 5110 support in Kampus
  - MC-LAG support etc.
Initial Provisioning at customer premises

• All equipment delivered directly to customers’ central data center
• OOB console server from Funet for initial provisioning
• Configuration and transceiver tests before moving the devices to their final location