How we deployed a datacenter in one click

ex-Blade Network Team: Cédric Hascoët, Jean-Christophe Legatte, Loïc Pailhas, Sébastien Hurtel, Tchadel Icard, and Vincent Bernat

FRnOG 34 — October 1st, 2021
What’s deployed?

- 2 Juniper QFX10002-72Q as edge routers,
- 32 Cisco Catalyst 2960S as OOB switches,
- 4 Facebook Wedge 100 as spine switches,
- 16 Facebook Wedge 100 as leaf switches,
- 4 Opengear CM71xx-2 as console servers,
- 2 Linux bastion/VPN/gateway servers,
- 1 Linux LibreNMS server.
What’s configured?

- Edge routers:
  - BGP configuration + policies
  - Routing engine protection
- BGP-based fabric (Facebook Wedge)
- Out-of-band fabric (Cisco)
- Administrative gateway (Linux)
  - ZTP
  - Firewall and NAT
  - Access to console servers
  - VPN to other sites
- External stores: DNS, IRR, RPKI, NetBox
Steps

1. Get space, power, cooling, racks, equipments, cabling done.
2. Install Debian 10 on gateway servers.
3. `./run-ansible-gitlab playbooks/site.yaml --limit=adm-gateway:\&location-ussfo03,none`
4. Wait for all devices to autoprovission.
5. `./run-ansible-gitlab playbooks/site.yaml --limit=location-ussfo03`
Source of truth

- No **NetBox**.
- **YAML** files versioned with **Git**.
  1. List of **devices**
  2. **Classifier**: from device name, attach properties to build a scope
  3. **Hierarchy definition** for data: given a scope, where to lookup data for a device
  4. **Data files**: flat YAML files fitted inside a hierarchy of directories
List of devices

devices:
# USSF003
## OOB
- ob1-n1.ussfo03.blade-group.net
- ob2-n1.ussfo03.blade-group.net
- ob1-p1.ussfo03.blade-group.net
- ob2-p1.ussfo03.blade-group.net
- ob1-p2.ussfo03.blade-group.net
# [...]
Classifiers:

- '\.(ussfo03)\.:'
  location: '\1'
  continent: us
- '^to\([12]\)-[as]?p\(d+\):'  
  member: '\1'
  pod: '\2'
- '^to[12]-p\d+\.ussfo03\.:'
  groups:
    - tor-bgp
    - tor-bgp-compute
- '^to[12]-\(p|ap|sp\)d+\.ussfo03\.:'
  os: cumulus
  model: wedge100
def searchpaths(scope):
    paths = [
        f"host/{scope[location]}/{scope[shorthost]}",
        f"location/{scope[location]}",
        f"os/{scope[os]}-{scope[model]}",
        f"os/{scope[os]}",
        'common'
    ]
    return paths
Data files

- Don't repeat yourself
- Data model should fit your needs

peer:
  ix-sfmix:
  rs-sfmix:
    monitored: true
    asn: 63055
    remote:
      - 206.197.187.253
      - 2001:504:30::ba06:3055:1
  blizzard:
    asn: 57976
    remote:
      - 206.197.187.42
      - 2001:504:30::ba05:7976:1
  irr: AS-BLIZZARD
Commit by Loïc
Jerikan

- Compile configuration files from source of truth and templates
- Faster than Ansible
- Easier to debug than Ansible
- Optionally checks generated configuration
Templates

- Using Jinja2
- Same as Ansible

```python
system {
    ntp {
        {% for ntp in lookup("system", "ntp") %}
            server {{ ntp }};
        {% endfor %}
    }
    name-server {
        {% for dns in lookup("system", "dns") %}
            {{ dns }};
        {% endfor %}
    }
}
```
Error handling

templates/opengear/config.j2:15: in top-level template code
    config.interfaces.{{ interface }}.netmask {{ infos.address | ipaddr("netmask") }}
        continent = 'us'
        device = 'con1-ag2.ussfo03.blade-group.net'
        environment = 'prod'
        host = 'con1-ag2.ussfo03'
        infos = {'address': '172.30.24.19/21'}
        interface = 'wan'
        location = 'ussfo03'
        loop = <LoopContext 1/2>
        member = '2'
        model = 'cm7132-2-dac'
        os = 'opengear'
        shorthost = 'con1-ag2'

value = JerkianUndefined, query = 'netmask', version = False, alias = 'ipaddr'

[..., 
    # Check if value is a list and parse each element
    if isinstance(value, (list, tuple, types.GeneratorType)):
        _ret = [ipaddr(element, str(query), version) for element in value]
        return [item for item in _ret if item]
>   elif not value or value is True:
E   jinja2.exceptions.UndefinedError: 'dict object' has no attribute 'adddress'
Integration into GitLab

- Use merge request workflow
- Review changes to data files and templates
- Build generated configuration files
- Produce a diff
Integration into GitLab
Ansible

- Inventory generated by Jerikan
- Single playbook
- Idempotency is important
- `--diff --check` should work as expected
- deploy complete configuration
Further reading

- Blog post about Jerikan+Ansible (w/ demo)
- GitHub repository (free bundle: Jerikan, Ansible playbooks, data, templates and generated configuration for two datacenters)