Automating Network Infrastructures with Ansible on FreeBSD

Firebird Networks Belgium
Introductions

• Network Consultant and Co-Founder of Firebird Networks

• Points of interest: datacenter networks, service provider architecture, network automation
It's 2014 on highway 101 from San Francisco to San Jose, some cars are driving themselves. Around the world there are military aircraft flying around with no pilot, being controlled by remotely from another country. In your data center there is an engineer/admin configuring a switch on a CLI. What's wrong with this picture?

Joe Onisick – Principal Engineer Cisco Systems
### Network Agility

<table>
<thead>
<tr>
<th>Year</th>
<th>Protocol</th>
<th>Configuration Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>1994</td>
<td>Telnet</td>
<td>Router&gt; enable&lt;br&gt;Router# configure terminal&lt;br&gt;Router(config)# enable secret cisco&lt;br&gt;Router(config)# ip route 0.0.0.0 0.0.0.0 20.2.2.3&lt;br&gt;Router(config)# interface ethernet0&lt;br&gt;Router(config-if)# ip address 10.1.1.1 255.0.0.0&lt;br&gt;Router(config-if)# no shutdown&lt;br&gt;Router(config-if)# exit&lt;br&gt;Router(config)# interface serial0&lt;br&gt;Router(config-if)# ip address 20.2.2.2 255.0.0.0&lt;br&gt;Router(config-if)# no shutdown&lt;br&gt;Router(config-if)# exit&lt;br&gt;Router(config)# router rip&lt;br&gt;Router(config-router)# network 10.0.0.0&lt;br&gt;Router(config-router)# network 20.0.0.0&lt;br&gt;Router(config-router)# exit&lt;br&gt;Router(config)# exit&lt;br&gt;Router# copy run start</td>
</tr>
<tr>
<td>2014</td>
<td>SSH</td>
<td>Router&gt; enable&lt;br&gt;Router# configure terminal&lt;br&gt;Router(config)# enable secret cisco&lt;br&gt;Router(config)# ip route 0.0.0.0 0.0.0.0 20.2.2.3&lt;br&gt;Router(config)# interface ethernet0&lt;br&gt;Router(config-if)# ip address 10.1.1.1 255.0.0.0&lt;br&gt;Router(config-if)# no shutdown&lt;br&gt;Router(config-if)# exit&lt;br&gt;Router(config)# interface serial0&lt;br&gt;Router(config-if)# ip address 20.2.2.2 255.0.0.0&lt;br&gt;Router(config-if)# no shutdown&lt;br&gt;Router(config-if)# exit&lt;br&gt;Router(config)# router rip&lt;br&gt;Router(config-router)# network 10.0.0.0&lt;br&gt;Router(config-router)# network 20.0.0.0&lt;br&gt;Router(config-router)# exit&lt;br&gt;Router(config)# exit&lt;br&gt;Router# copy run start</td>
</tr>
</tbody>
</table>
Automation vs. Orchestration

• What is automation?
  • Automation eliminates the necessity of repeatable manual tasks

• What is orchestration?
  • Orchestration is the manner in which automated tasks are grouped together in coordinated workflows
Part I: Introducing Ansible
“What is Ansible?”

• “Ansible is a super-simple automation platform that is agentless and extensible”

• By simple it means that you don’t need any coding knowledge to get started

• Agentless means that you do not require an agent on each device in order to be able to control them (important for vendor-locked network devices)

• Extensible means that it benefits from the open-source community and things will be built for it.
Basic Ansible Architecture

• Ansible = automation platform

• Can be installed on every laptop or just a central server.

• Use pip, apt or yum or pkg to install on *nix-based machines

• All automation is performed out of the device that hosts the installation of Ansible (also known as a control host)

• Uses the notion of playbooks - a set of automation tasks and instructions which are pushed for execution on specific hosts.
Playbooks

• From ansible.com: “Playbooks are Ansible’s configuration, deployment, and orchestration language. They can describe a policy you want your remote systems to enforce, or a set of steps in a general IT process.”

• Playbooks are similar to an Ikea instruction manual that breaks the entire process of configuring a router, or a BGP process or whatever else into small little tasks and delegates the interaction with the devices.

• Best feature: human-readable (if you like YAML)

• Check more examples out: https://github.com/ansible/ansible-examples
Ansible Playbooks are expressed using YAML syntax

YAML - YAML Ain’t Markup Language

YAML uses a small amount of separators - indentation gives structure, colons separate keys, and dashes create bullet lists

Every YAML file must start with - - - and end with . . .

Members of a list are marked with a dash ( - Apple )

Dictionary terms are a pair separated by a colon - key: value (yes, the space between the two is necessary)

More syntax documentation: http://docs.ansible.com/ansible/latest/YAMLSyntax.html
Templates

• Ansible uses the Python-based Jinia2 templating language

• A template is a standard configuration without its variables

• Internally based on Unicode, it is inspired by Django’s templating system

• Jinjia supports a few control structures like if and for-loops making it easy to shorten your templates

• A good starter for understanding Jinia templates: https://realpython.com/blog/python/primer-on-jinja-templating/
From Configuration to Jinjia Template

```
CRS# conf t
CRS(config)# router bgp 65501
CRS(config-bgp)# neighbor 10.10.10.2
CRS(config-bgp-nbr)# remote-as 2000
CRS(config-bgp-nbr)# password <MD5 password>
CRS(config-bgp-nbr)# ebgp-multihop 2
CRS(config-bgp-nbr)# update-source loopback0
CRS(config-bgp-nbr)# address-family ipv4 unicast
CRS(config-bgp-nbr-af)# route-policy xxxxx in
CRS(config-bgp-nbr-af)# exit
```

```text
! router bgp {{ as_number }}
!
neighbor {{ ip_neighbor }}
 remote-as {{ as_number }}
 password {{ md5_password }}
 ebgp-multihop {{ mhop_value }}
 update-source {{ update_if }}
 address-family ipv4 unicast
!```
From Configuration to Jinjia Template

contents of leaf_vni.j2

{ % for vlan in vlans % }
  vlan {{ vlan.id }}
  name {{ vlan.name }}
  vni {{ vni.id }}
{% endfor %}

!
Variables and Variable Files

• Double curly brackets = variables

• Variables are not stored in the templates

• Variables are stored in variable files

• Example variable file
Basic Working Ansible Playbook

• Premises: Build the most basic playbook that can check time on 2 devices

• Step 1: Build the “Playbook” file
Basic Working Ansible Playbook

• Make sure your inventory is up to date

• And then just run the playbook by typing ansible-playbook file_name.yml
Summary of a Simple Ansible Playbook

- Every Ansible Playbook will be written in YAML, has a specific necessity for beginning with - - - - and ending with . . .

- It needs a method to connect to its devices which are stored in an inventory file

- When templates need to be applied, a template file is used - Jinjia2 is the preferred templating language

- The values that are introduced for each of the devices for each of the templates are stored in a variable file which is also a .j2 file
Part II: FreeBSD for Network Engineers

- Setting up a new VM with ansible & python in a matter of minutes
- Executing playbooks and working around known caveats
Getting Ansible on Your FreeBSD Machine

• Setting up Ansible on FreeBSD means setting up your control machine. You can do this in a jail, you can have it running as a VM somewhere, or as a bare-metal machine.

• The obvious requirement is that it needs to be able to access the hosts that it should “manage” (more a network problem than a server problem)

• **Tip:** Make sure your username is in the sudoers group (especially if you provision a new machine)
Preparing your machine

• Have OpenSSH up and running on your machine

   $ service -e | grep sshd

   /etc/rc.d/sshd

• If it’s not running, make sure you add & activate it at boot:

   # echo 'sshd_enable="YES"' >> /etc/rc.conf

   # service sshd start

• Install ansible & python in one command:

   # pkg install ansible python
Quick Checks

$ freebsd-version
11.1-RELEASE

$ service -e | grep sshd
/etc/rc.d/sshd

$ python -V
Python 2.7.15
The ansible_python_interpreter caveat

- FreeBSD (OpenBSD too for that matter) doesn’t come with `/usr/bin/python`
- The ports don't install a "python" package, actually: they install a version of python, named after the version
The ansible_python_interpreter caveat
The ansible_python_interpreter caveat

• One solution is to just add it to your hosts files and pass it as a variable.

• This also allows you to use different versions of Python depending on the scripts that you want to use.

```
$ cat hosts | grep -A 2 -B 2 ansible_python

[all:vars]
#ansible_python_interpreter="/usr/local/bin/python2.7"
```
Examples of Playbooks and Network Applications
“To err is human, to apply that error to 1000 servers at once is DevOps.”

-Unknown
Questions?
Thank you!