ANSIBLE

August 2017

Khelil Sator
Juniper Networks

WHAT IS ANSIBLE?

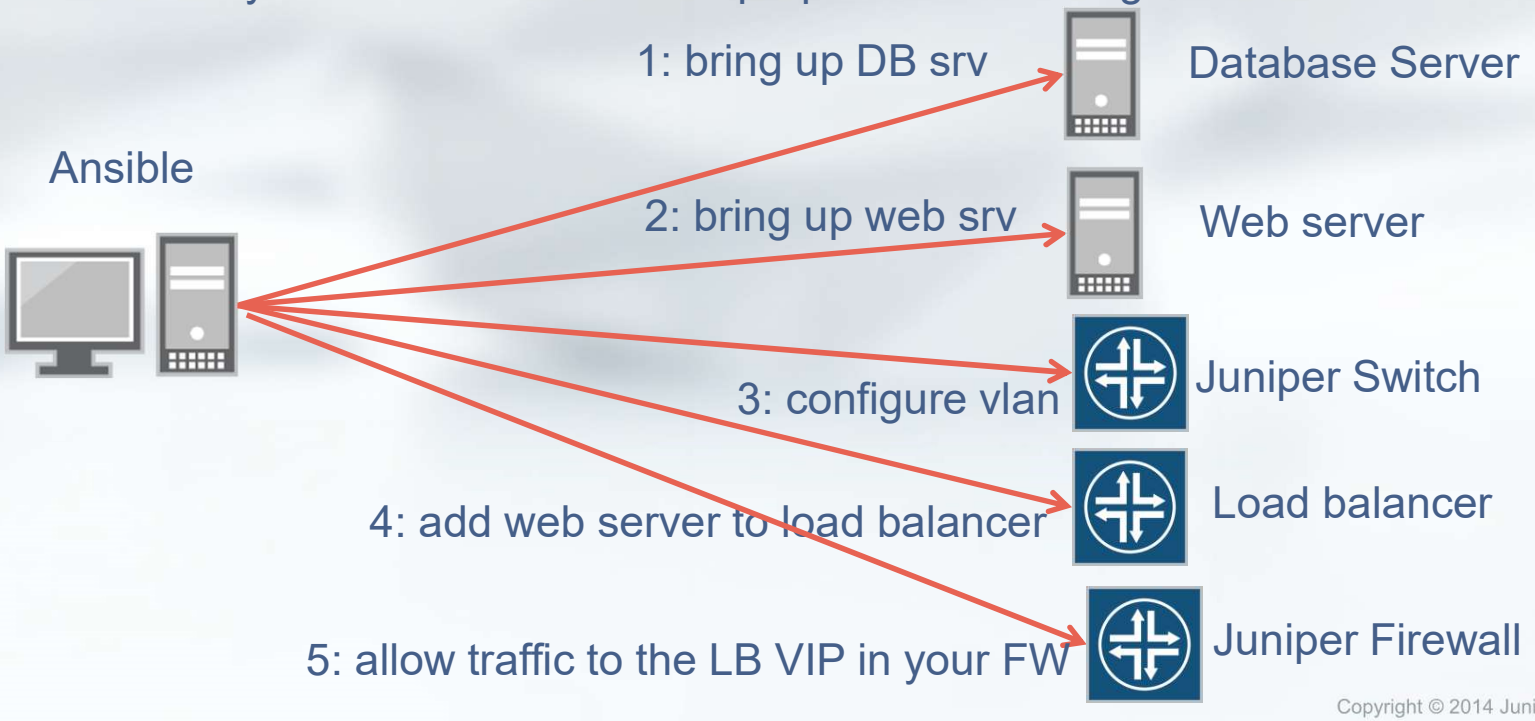
- A radically simple IT automation platform
  - Ansible's goals are similar to other tools such as Puppet, Chef, SaltStack, StackStorm
  - Acquired by Red Hat in 2015
  - Current release is 2.4 (October 2017)
  - Most popular automation tool for network automation

WHAT CAN WE DO WITH ANSIBLE?

- This is an IT automation tool
- Human driven automation
- Initially for servers and applications
- Has now strong network automation capabilities
- Ansible is designed for performing actions (tasks) on multiple servers.
- Ansible uses cases are:
  - Configuration generation from templates
  - Configuration deployment
  - Orchestration

ORCHESTRATION WITH ANSIBLE

- You can manipulate an entire application stacks made up multiple tiers.
  - you need to bring up the database before bringing up the web servers
  - you need to take web servers out of the load balancer one at a time in order to upgrade them without downtime.
  - you need to ensure that your networks have their proper VLANs configured



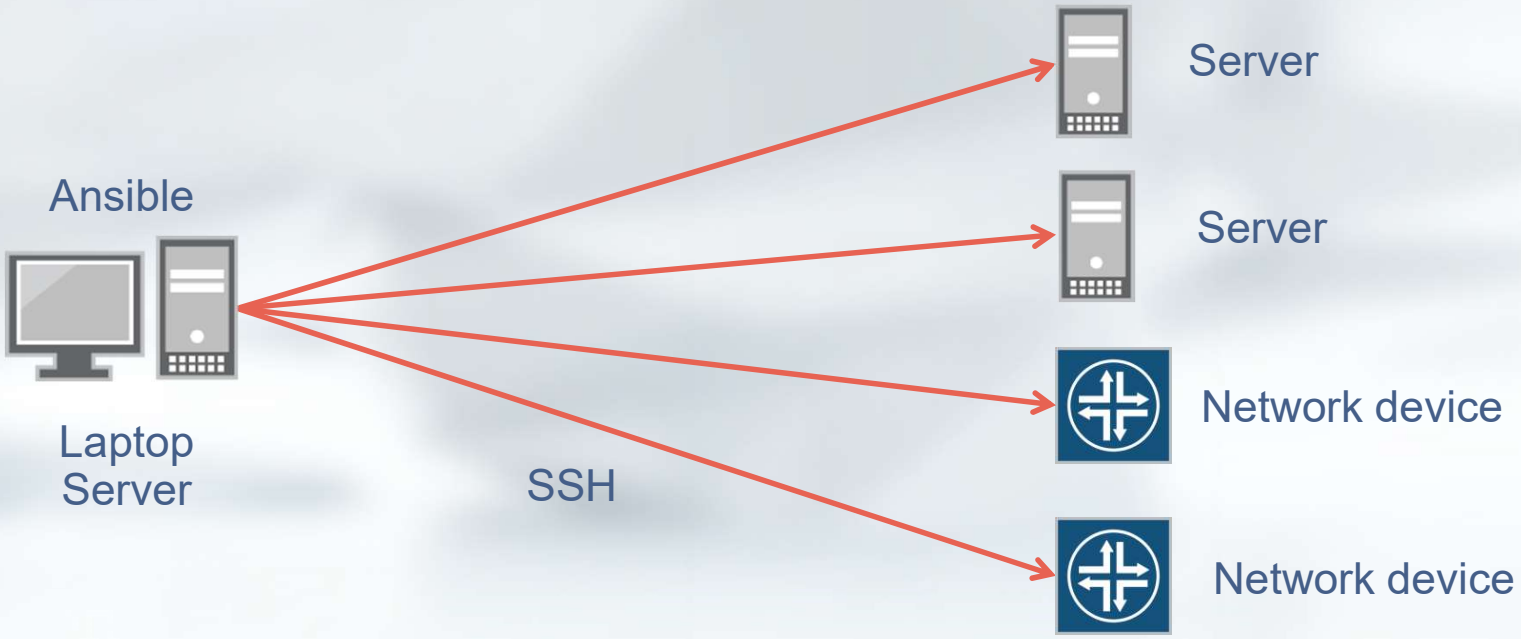
WHY SOME PEOPLES LOVE ANSIBLE

- Agentless (uses SSH)
- Easy to use. Human readable automation. Yaml (Data, not code)
- Use Jinja2 templates.
- A large library of modules (about 1000)
- Push based. No Master server. Running from everywhere.
- IT automation (not only network automation)
- Nice orchestration capabilities
- Has loops and conditionals
- Idempotent (but not all the modules)
  - Ansible first check the actual state of the devices and then apply changes only if the current state does not match the desired state.
- Written in Python (but no need to understand Python).
- You can add your own modules to extend Ansible capabilities.

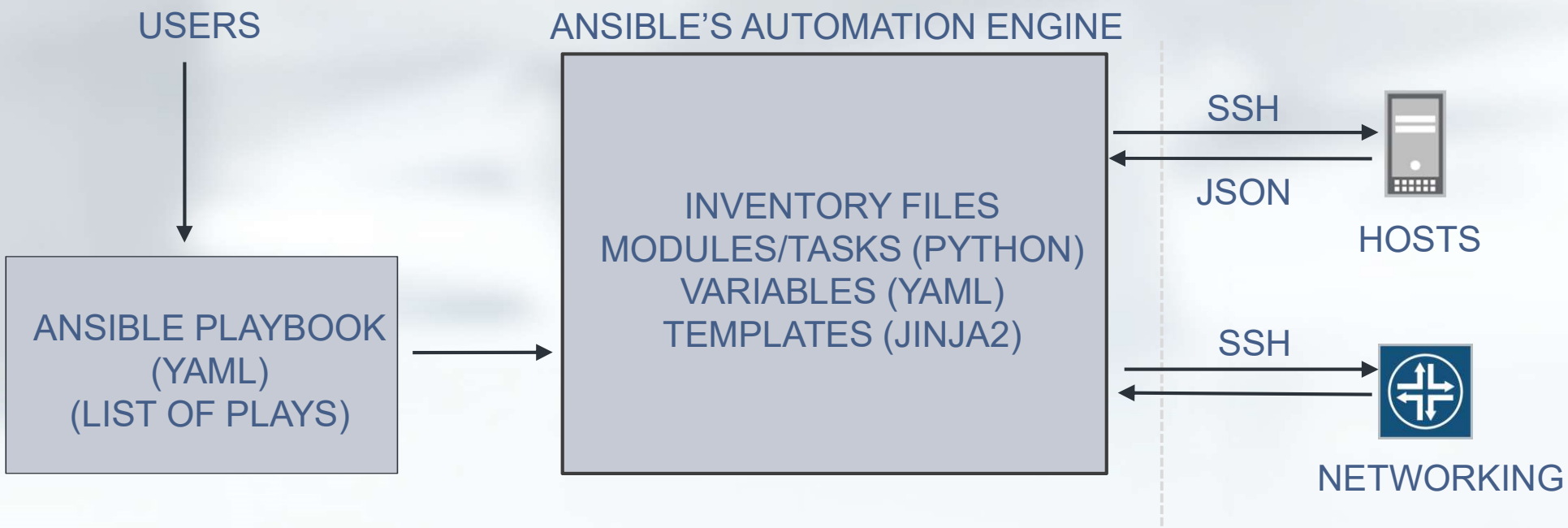
HOW ANSIBLE WORKS?

- Standard mode uses SSH to transport python modules to the remote boxes, and then executes these python modules on remote boxes, and then eliminates them from remote hosts.
  - Python modules execution will change the devices configuration.
  - Requires Python installed on remotes machines.
    - You can use the Ansible module ping to check this. It returns pong on success.
    - Some Ansible modules require also some python libraries installed on the remote hosts.
- API mode: Modules run on Ansible server. Ansible then uses an API (Netconf ....) to change the devices configuration.
  - “connection: local” in the playbook means the actions/tasks will be run local on the server running Ansible and not on the target host.

HOW ANSIBLE WORKS?



HOW ANSIBLE WORKS?



ANSIBLE KEY COMPONENTS

- Inventory
- Variables
- Tasks/modules
- Plays and Playbooks

INVENTORY FILES

```
[spine]
qfx10002-01
qfx10002-02

[leaf]
qfx5100-01
qfx5100-02
qfx5100-03
qfx5100-04
```

- text file with your inventory
  - you can have many inventory files
  - The default ansible 'hosts' file lives in /etc/ansible/hosts
- Comments begin with the '#' character
- Blank lines are ignored
- You can enter hostnames or ip addresses
- Groups of hosts (per device type, per location, ...)
- A hostname/ip can be a member of multiple groups

```
[all:children]
spine
leaf

[spine]
qfx10002-01    junos_host=192.168.0.1
qfx10002-02    junos_host=192.168.0.2

[leaf:children]
rack01
rack02

[rack01]
qfx5100-01     junos_host=192.168.0.3
qfx5100-02     junos_host=192.168.0.4

[qfx10000]
qfx10002-01
qfx10002-02

[qfx5100]
qfx5100-01
qfx5100-02

[siteA]
qfx10002-01
qfx5100-01
```

- Simple but very important
- A device can be part of multiple groups
- Can have a hierarchy of groups
- Can groups devices by type/location/ …
- qfx5100-01 is part of groups:
  - All
  - leaf
  - rack01
  - qfx5100
  - siteA
- Inventory files can take advantage of enumerations
  - QFX5100-[1:13]
  - host[a:z]
  - 172.16.0.[1:254]
- Inventory files can take advantage of variables

VARIABLE FILES

```
global:
    root_hash: $1$ZUlES4dp$OUwWo1g7cLoV/aMWpHUnC/
    time_zone: America/Los_Angeles
    name_servers:
    - 192.168.5.68
    - 192.168.60.131
    ntp_servers:
    - 172.17.28.5
    snmp:
        location: "Site 1"
        contact: John Doe
        polling:
        - community: public
    routes:
        default: 10.94.194.254
```

- A variable file is a YAML file
- Variable definitions is very flexible
- Variable files can live in many places
- Variables can be defined in playbooks,
- Variables can be defined in the inventory files.
- Variable files can also be referred in a playbook (with the include_vars module)
- Accessible from templates and playbooks
- The module setup is automatically called by playbooks to gather facts (variables discovered, not set by the user) about remote hosts (setup module is not valid for junos devices so use gather_facts: no in your play)

VARIABLES ASSOCIATE WITH GROUPS AND HOSTS

```
|-- group_vars
|   |-- leaf01
|   |   |-- file1.yaml
|   |   `-- file2.yml
|   |-- leaf02
|   |   |-- file1.yaml
|   |   `-- file2.yml
|   |-- spine
|   |   |-- file1.yaml
|   |   `-- file2.yml
|   |-- all.yaml
|   |-- qfx10000.yaml
|   `-- qfx5100.yaml
|-- host_vars
|   |-- qfx10002-01
|   |   |-- file1.yml
|   |   `-- file2.yaml
|   |-- qfx5100-01
|   |   |-- file1.yml
|   |   `-- file2.yaml
[…]
|   |-- qfx5100-04
|   |   |-- file1.yml
|   |   `-- file2.yaml
|   `-- qfx5100-05.yml
|   `-- qfx5100-06.yml
```

- Variable files can live in many places
  - host_vars folder has the variables per host
  - group_vars folder has the variables per group (qfx, spines, site1, all, …)
  - vars subfolders of roles can be used to store variables for roles
- One or multiple variable files per
  - Host
  - Group
- Very flexible

GENERATE C

host_vars/qfx5100-01.yaml

```
ospf:
     interfaces:
          ge-0/0/0:
               ip: 192.168.0.1
               mask: 24
          ge-0/0/1:
               ip: 192.168.1.1
               mask: 24
host:
     mgmt:
          ip: 10.10.10.1
          mask: 24
```

group_vars/qfx5100.yaml

```
Mgmt_interface: em0
```

group_vars/all.yaml

```
Global:
     root_hash: $1$dvewqcsarwrgvwr
Host:
     mgmt:
          mask: 24
```

```
system {
     host-name {{ inventory_hostname }};
      root-authentication {
          encrypted-password "{{ global.roo
      }
}
interfaces {
   {{ mgmt_interface }} {
      unit 0 {
          family inet {
               address {{ host.mgmt.ip }}/{
          }
      }
   }
{% for interface in ospf.interfaces %}
     {{ interface }} {
        unit 0 {
          family inet {
               address {{interface.ip }}/{{
          }
        }
     }
{% endfor %}
}

protocols {
     ospf {
          area 0.0.0.0 {
{% for interface in ospf.interfaces %}
             interface {{ interface }}
{% endfor %}
}
```

Final version

```
system {
     host-name qfx5100-01;
      root-authentication {
          encrypted-password "$1$dvewqcsarwrgvwr";
      }
}
interfaces {
   em0 {
      unit 0 {
          family inet {
               address 10.10.10.1/24;
          }
      }
   }
   ge-0/0/0{
        unit 0 {
          family inet {
               address 192.168.0.1/24
          }
        }
     }
   ge-0/0/1{
        unit 0 {
          family inet {
               address 192.168.1.1/24
          }
        }
    }
}

protocols {
     ospf {
          area 0.0.0.0 {
               interface ge-0/0/0
                interface ge-0/0/1
}
```

SOME ANSIBLE USE CASES FOR JUNOS DEVICES

- Generate configuration
- Deploy configuration
- Rollback configuration
- Retrieve configuration
- Upgrade devices
- Audit the devices operation states

Ansible librairies to interact with Junos

- There are two Ansible librairies to interact with Junos.
  - An Ansible library for Junos built by Juniper (hosted on the Ansible Galaxy website)
  - An Ansible library for Junos built by Ansible (since Ansible 2.1)

- You can use both of the Ansible libraries for Junos

Ansible modules for Junos built by Juniper

- Hosted on the Ansible Galaxy website (https://galaxy.ansible.com/Juniper/junos/)
- Documentation for the last release: http://junos-ansible-modules.readthedocs.io/
- Documentation for a specific release http://junos-ansible-modules.readthedocs.io/en/1.4.3/
- Installation (last release): Execute the “ansible-galaxy install Juniper.junos”
- Installation (specific release): Execute the “ansible-galaxy install Juniper.junos,1.4.3”
- Source code https://github.com/Juniper/ansible-junos-stdlib

Ansible core modules for Junos built by Ansible:

- Documentation: http://docs.ansible.com/ansible/latest/list_of_network_modules.html#junos
- Installation: They ship with ansible itself (from Ansible 2.1).
- Source code: https://github.com/ansible/ansible/tree/devel/lib/ansible/modules/network/junos

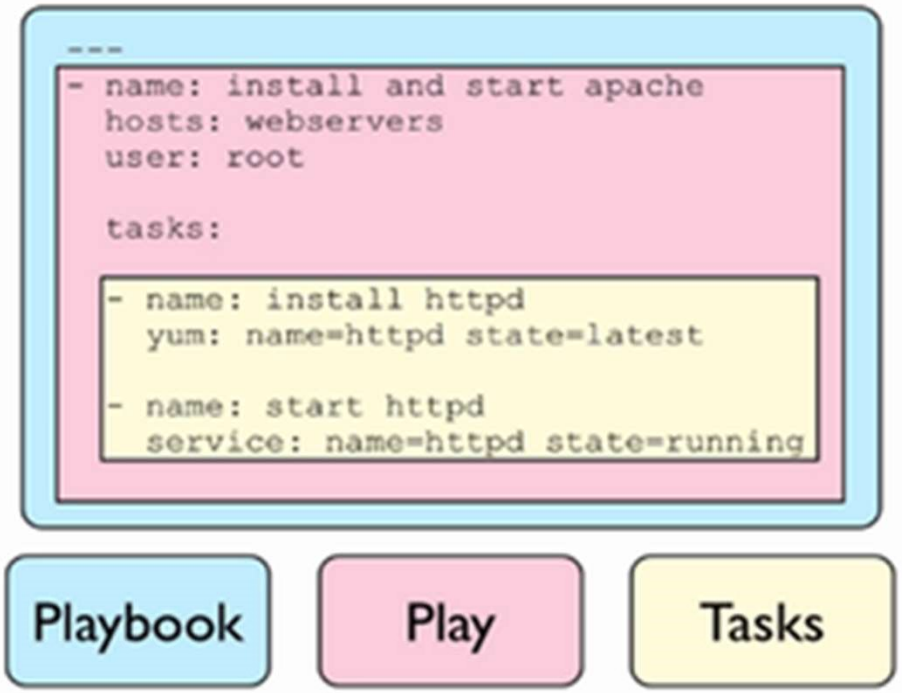
EXAMPLES OF TASKS WITH YUM MODULE

```
- name: install the latest version of Apache
  yum: name=httpd state=latest

- name: remove the Apache package
  yum: name=httpd state=absent

- name: install one specific version of Apache
  yum: name=httpd-2.2.29-1.4.amzn1 state=present
```

A Playbook



ANSIBLE JARGON

- Playbook: An Ansible script. Written in Yaml. A list of Plays.
- Plays: A play in a playbook contains a set of hosts, and a list of tasks to be executed on those hosts
- Inventory file: your hosts and groups. Can be dynamic
- Variables: there are various options to define variables. YAML.
- Task: a Python/Ansible module with some arguments.
- Handlers: a triggered task
- Server facts: Like puppet. Automatically gather by setup module.
- Roles: a reusable automation content you can assign to hosts
- Galaxy: repository for Ansible roles like the ones for Junos built by Juniper.

ROLE

```
---
- name: Create and apply configuration for Leaves QFX / L2
  hosts: leaf-qfx-l2
  connection: local
  gather_facts: no
  roles:
    - common
    - underlay-ebgp
    - overlay-evpn-qfx-l2
    - overlay-evpn-access
```

- Roles are called inside a playbook
- Very useful to reuse the same automation content across playbook
  - Tasks
  - Templates
  - variables
- You can use the “ansible-galaxy init” command to create the skeleton's role

ANSIBLE DEMO

https://github.com/ksator/junos-automation-with-ansible


JUNIPer
NETWORKS
THANK YOU!