Automating networks of all sizes with Red Hat Ansible Automation Platform

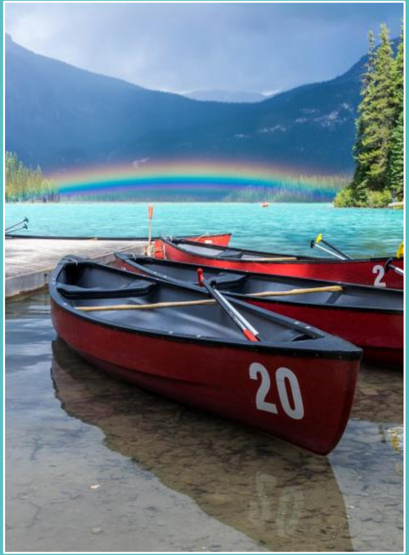
Landon Holley

Architect

AnsibleFest

About me

Consulting architect, traveler, photographer, farmer, geek



AnsibleFest

Just the Facts

Ansible Automation Platform facts

Network automation begins and ends with **facts**

Topics

Fact collection
Custom facts
Network resource modules

Execution

Command/configuration
Backup/restore
Provisioning
State management
CMDB

Fact collection

How to get started

Fact collection

Gather **native** facts for network devices

```
- hosts: all
  gather_facts: true
```

Custom facts

When all else fails...

```
- set_fact:
  device_thing: {{ ansible_version }}-ios.cfg
```

AnsibleFest

Configuration and backup/restore

Facts for everything

Creating custom facts; backups and restores

```
- ios_command:
  commands:
    - show running-config
  register: output

- set_fact:
  backup: "{{ output }}"
```

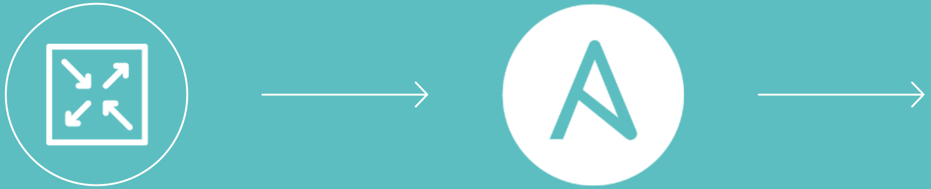
```
- ios_config
  backup: yes
  backup_options:
    filename: {{ host }}.cfg
    dir_path: /opt/backup
```

```
- ios_facts:
  gather_subset: config
  gather_network_resources: yes
```

AnsibleFest

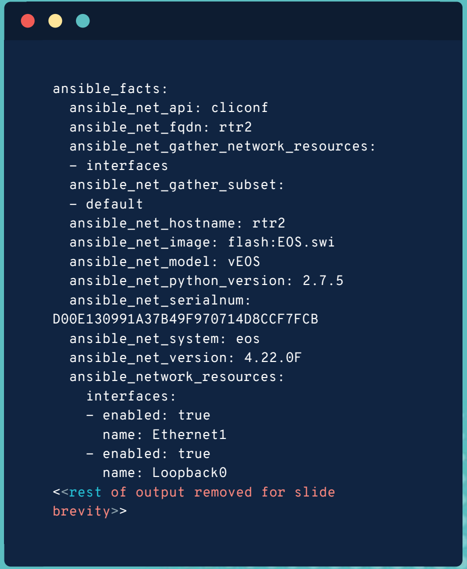
Configuration and backup/restore

Facts for everything



Network native configuration

Convert to structured data



```
ansible_facts:
  ansible_net_api: cliconf
  ansible_net_fqdn: rtr2
  ansible_net_gather_network_resources:
  - interfaces
  ansible_net_gather_subset:
  - default
  ansible_net_hostname: rtr2
  ansible_net_image: flash:EOS.swi
  ansible_net_model: vEOS
  ansible_net_python_version: 2.7.5
  ansible_net_serialnum:
D00E130991A37B49F970714D8CCF7FCB
  ansible_net_system: eos
  ansible_net_version: 4.22.0F
  ansible_network_resources:
    interfaces:
    - enabled: true
      name: Ethernet1
    - enabled: true
      name: Loopback0
<<rest of output removed for slide
brevity>>
```

Network Resource Modules

Network resource modules

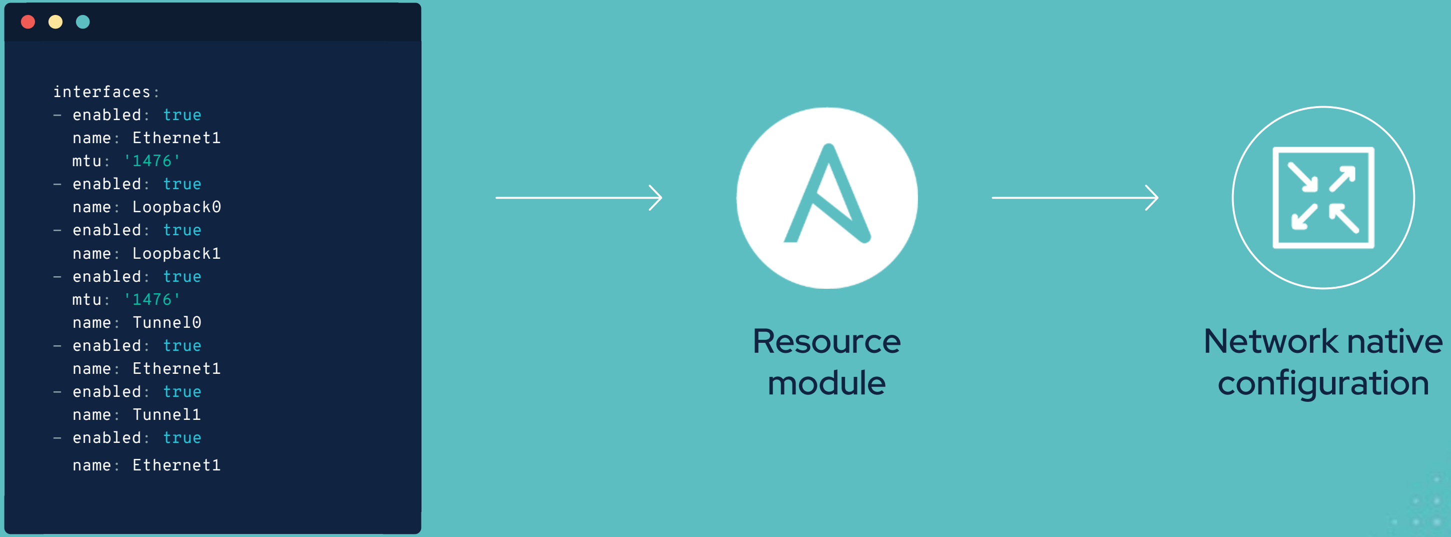
Managing device state across different devices and types

Configuration to code

- Built-in logic with commands and orchestration
- Vendor-agnostic data model
- Bidirectional with configuration to facts and facts to configuration

Configuration and backup/restore

Facts for everything



```
interfaces:
- enabled: true
  name: Ethernet1
  mtu: '1476'
- enabled: true
  name: Loopback0
- enabled: true
  name: Loopback1
- enabled: true
  mtu: '1476'
  name: Tunnel0
- enabled: true
  name: Ethernet1
- enabled: true
  name: Tunnel1
- enabled: true
  name: Ethernet1
```

AnsibleFest

Managing device state

Practical examples of using network resource modules

```
interface_config:
- interface: Ethernet1/1
  description: Te0/1/1
  enabled: True
  mode: trunk
  portchannel_id: 100

- interface: Ethernet1/2
  enabled: False

- interface: port-channel100
  description: vPC PeerLink
  mode: trunk
  enabled: True
  vpc_peerlink: True
```

```
- nxos_interfaces:
    config:
      name: "{{ item['interface'] }}"
      description: "{{ item['description'] }}"
      enabled: "{{ item['enabled'] }}"
      mode: "layer3"
    state: replaced
    loop: "{{ interface_config }}"
  when: item['enabled'] == True
```

State:

`Merged`	- add/increment
`Replaced`	- template/diff
`Overridden`	- force/policy
`Deleted`	- destroy/remediate

AnsibleFest

Managing device state

Complete

1. Gather facts

```
- name: gather nxos facts
  nxos_facts:
    gather_subset: interfaces
```

2. Use facts

```
debug: var:
hostvars['inventory_hostname']['interfaces']
```

```
ansible_facts:
  ansible_net_fqdn: rtr2
  ansible_net_hostname: rtr2
  ansible_net_system: nxos
  ansible_net_version: 14.22.0F
  ansible_network_resources:
    interfaces:
    - name: Ethernet1/1
      enabled: true
      mode: trunk
    - name: Ethernet1/2
      enabled: false
```

Inventories and Repositories

Planning for repositories and variables

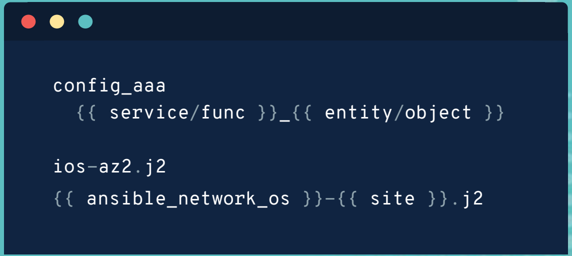
Role and directory structures and git repositories

Plan big, start small

Develop your roles to suit your configurations:

- AAA
- Interface
- Network time protocol (NTP)
- Others

Design naming standards for roles, templates, and variables



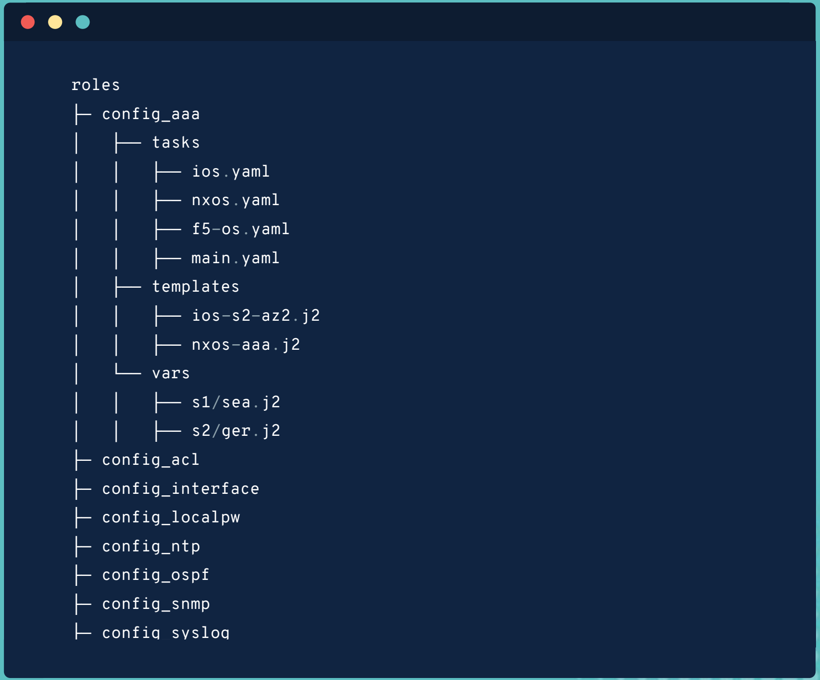
```
config_aaa
  {{ service/func }}_{{ entity/object }}

ios-az2.j2
{{ ansible_network_os }}-{{ site }}.j2
```

AnsibleFest

Planning for repositories and variables

Role and directory structures and git repositories

Plan big, start small



```
roles
├─ config_aaa
│    ├── tasks
│    │    ├── ios.yaml
│    │    ├── nxos.yaml
│    │    ├── f5-os.yaml
│    │    ├── main.yaml
│    ├── templates
│    │    ├── ios-s2-az2.j2
│    │    ├── nxos-aaa.j2
│    └── vars
│    │    ├── s1/sea.j2
│    │    ├── s2/ger.j2
├─ config_acl
├─ config_interface
├─ config_localpw
├─ config_ntp
├─ config_ospf
├─ config_snmp
├─ config_syslog
```

AnsibleFest

Building inventories

Developing a dynamic playbook

Less is more

Smaller inventories, smaller jobs, faster performance

Start by grouping inventories

ansible_network_os:

`ios, nxos, iosxr, eos, f5, etc…`

Minimum inventory variables:

```
ansible_hostname = ip/fqdn
ansible_network_os = ios
ansible_username =
username
ansible_password =
password
```

Additional network inventories/variables:

```
device_family = cisco
device_type = router
model_number = N9KC95
serial_number = abc123
geo = na
loc = sea
zone = dmz
rack = l1t3
```

Tying it all together

Workflows for provisioning and continuous integration/continuous deployment (CI/CD)

Tower workflows make it easy to automate things against multiple inventories

Provisioning workflow:

1. SSH to serial console, login to device
2. Apply base configuration (local password)
3. Login with local password, setup AAA
4. Login with user credential, finish configuration

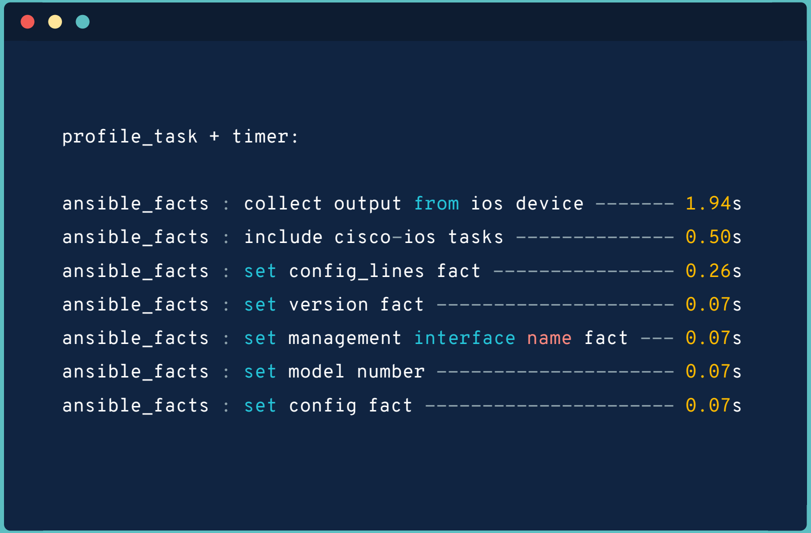
CI/CD workflows:

1. Clone repository
2. Add code, make changes
3. Lint and run in check mode
4. If success, run playbook
5. Commit changes, push, merge
6. Checkout next branch
7. Repeat

Scale and performance

Playbook performance

Task and process monitoring



```
profile_task + timer:

ansible_facts : collect output from ios device -------- 1.94s
ansible_facts : include cisco-ios tasks --------------- 0.50s
ansible_facts : set config_lines fact ----------------- 0.26s
ansible_facts : set version fact --------------------- 0.07s
ansible_facts : set management interface name fact --- 0.07s
ansible_facts : set model number --------------------- 0.07s
ansible_facts : set config fact ----------------------- 0.07s
```

Whitelist plugins (ansible.cfg):

```
profile_task
timer
```

Process monitoring:

```
dstat
htop
fio
```

Debug logging (or ansible.cfg):

```
export ANSIBLE_DEBUG=true
export ANSIBLE_LOG_PATH=/tmp/ansible-debug.log
```

Scaling fact collection

Scaling tower, monitoring performance, and external logging

Consider fact collection schedule

- What needs to be collected/monitored frequently?
- What can be checked once a day/week?

Input variables and output facts

- Avoids rerunning device commands
- Minimizes database modules (DB) access

Determine tower architecture

- Instance groups
- Container groups

Off-load search and analytics

- ELK
- Splunk
- Cosmos

Learning resources

Continue your automation journey with Red Hat® Ansible® Network Automation

Red Hat
Ansible Automation
Platform

Networking workshop
https://github.com/ansible/workshops

Deep dive into resource modules, Trishna Guha
https://www.ansible.com/deep-dive-into-ansible-network-resource-module

Red Hat Certification
Ansible for Network Automation (DO457)
https://www.redhat.com/en/services/training/do457-ansible-network-automation

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

youtube.com/user/RedHatVideos

linkedin.com/company/Red-Hat

facebook.com/ansibleautomation

twitter.com/ansible

AnsibleFest