How to use this deck

Name:
Network Automation Workshop Deck

Purpose:
This slide deck is part of a training course designed as an introduction to Ansible for network engineers and operators. The slides are meant to be taught in conjunction with hands-on exercises with a lab topology of Automation controller + 4 network devices.

Last updated:
Sep 21, 2021 (check history for older versions)

What this deck is for?
This deck corresponds to the prescriptive exercises available on https://ansible.github.io/workshops/exercises/ansible_network/

The upstream source for exercises and provisioner are provided on https://github.com/ansible/workshops

What this deck is not for?
This is not a replacement for Red Hat training. This is a small “taste” of Ansible Automation Platform and meant to help people understand what is possible for network engineers with automation. Please refer to https://www.redhat.com/en/services/training-and-certification for official training

Google Slides source link (Red Hat internal):
https://docs.google.com/presentation/d/1PIT-kGAGMVEEK8PsuZCoyzFC5C1zLBwdnftnUsdUNWQ/edit?usp=sharing
Housekeeping

Understanding the format of this class

- Timing
- Breaks
- Takeaways
What you will learn

- Introduction to Ansible automation
- How Ansible works for network automation
- Understanding Ansible modules and playbooks
- Executing Ansible playbooks to make configuration changes
- Gather information (Ansible facts)
- Network Resource Modules
- Using Automation controller to operationalize automation for your enterprise
- Major Automation controller features - RBAC, workflows
Introduction

Topics Covered:
- What is the Ansible Automation Platform?
- What can it do?
- Why Network Automation?
- How Ansible Network Automation works
Automation happens when one person meets a problem they never want to solve again
Many organizations share the same challenge

Too many unintegrated, domain-specific tools

Network ops  SecOps  Devs/DevOps  IT ops
Why the Ansible Automation Platform?

**Powerful**
Orchestrate complex processes at enterprise scale.

**Simple**
Simplify automation creation and management across multiple domains.

**Agentless**
Easily integrate with hybrid environments.
Why the Red Hat Ansible Automation Platform?

Automate the deployment and management of automation

Your entire IT footprint

Do this...

- Orchestrate
- Manage configurations
- Deploy applications
- Provision / deprovision
- Deliver continuously
- Secure and comply

On these...

- Firewalls
- Load balancers
- Applications
- Containers
- Virtualization platforms
- Servers
- Clouds
- Storage
- Network devices
- And more ...
Break down silos
Different teams a single platform
What makes a platform?

**Red Hat Ansible Automation Platform**

- **Content creators**
- **Operators**
- **Domain experts**
- **Users**

<table>
<thead>
<tr>
<th>On-premises</th>
<th>Ansible Cloud Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automation controller</td>
<td>Automation hub</td>
</tr>
<tr>
<td>Automation services catalog</td>
<td></td>
</tr>
<tr>
<td>Insights for Ansible Automation Platform</td>
<td></td>
</tr>
</tbody>
</table>

**Ansible content domains**

- **Infrastructure**
  - Linux
  - Windows
- **Cloud**
- **Network**
- **Security**

**Ansible command line**

- Fueled by an open source community
Red Hat named a Leader in The Forrester Wave™
Infrastructure Automation Platforms, Q3 2020

Received highest possible score in the criteria of:
- Deployment functionality
- Product Vision
- Partner Ecosystem
- Supporting products and services
- Community support
- Planned product enhancements

“Ansible continues to grow quickly, particularly among enterprises that are automating networks. The solution excels at providing a variety of deployment options and acting as a service broker to a wide array of other automation tools.”

“Red Hat’s solution is a good fit for customers that want a holistic automation platform that integrates with a wide array of other vendors’ infrastructure.”

Source:

DISCLAIMER: The Forrester Wave™ is copyrighted by Forrester Research, Inc. Forrester and Forrester Wave™ are trademarks of Forrester Research, Inc. The Forrester Wave™ is a graphical representation of Forrester’s call on a market and is plotted using a detailed spreadsheet with exposed scores, weightings, and comments. Forrester does not endorse any vendor, product, or service depicted in the Forrester Wave™. Information is based on best available resources. Opinions reflect judgment at the time and are subject to change.
Use-Case

- Network Automation
68% of 77 respondents indicated they still use command line interface (CLI) on individual devices as the primary method of making network changes.

Source: Gartner, Market Guide for Network Automation and Orchestration Tools, September 2020
Why hasn’t networking changed?

Networking vendors are the trusted advisors

PEOPLE

● Domain specific skill sets
● Vendor oriented experience
● Siloed organizations
● Legacy operational practices

PRODUCTS

● Infrastructure-focused features
● CLI-based interfaces
● Siloed technologies
● Monolithic, proprietary platforms
Next generation networking
Automation to effectively manage increasing diversity and scope

**Edge / IoT Devices**
New device types entering networks at scale, with distributed computing.

**Hybrid cloud**
Numerous deployment forms across the globe.

**Digital transformation**
Responding with new applications is only as fast as the slowest process.

**Data-intensive computing**
Artificial intelligence, digital applications and growing data driving connectivity.
What is Ansible Network Automation?

Ansible network automation is our content domain focused on networking use cases. The goal is to provide network teams with the tools and an operational framework to implement next-generation network operations, manage network infrastructure-as-code, and better support digital transformation by connecting teams across the IT organization.

Ansible network automation is a set of Certified Content Collections designed to streamline and operationalize network operations across multiple platforms and vendors.
Modernize and scale network operations

Choose what network tasks to automate at your own pace

TRADITIONAL NETWORK OPERATIONS

- Traditional culture
- Risk averse
- Proprietary solutions
- Siloed from others
- “Paper” practices, MOPs
- “Artisanal” networks

NEXT-GEN NETWORK OPERATIONS

- Community culture
- Risk-aware
- Open solutions
- Teams of heroes
- Infrastructure as code
- Virtual prototyping / DevOps
What does it do?

Automate your network with a single tool

Configuration Management
Platform agnostic configuration management to standardize and enforce best-practices.

Infrastructure Awareness
Track network resources through facts gathering, to perform preventive maintenance, reducing outage risks and costs of unnecessary hardware-refresh.

Network Validation
Examine operational state to check network connectivity and protocols and enhance operational workflows to help measure network intent.
What is it for?
Start Small, Think Big

Three high-level benefits for successful network operations

**Configuration Management**
- Automate backup & restores
- Scoped Config Management

**Infrastructure Awareness**
- Dynamic Documentation
- Compliance and traceability

**Network Validation**
- Validate operational steady-state
- Roll back if configuration changes don’t meet goals
Ansible Network Ecosystem

SWITCHES

ROUTERS

ENTERPRISE FIREWALLS

LOAD BALancers

CONTROLLERS

IP ADDRESS MGMT
Deep diving on use-cases

Configuration Management
- Config Backup and Restore
- Scoped Config Management

Infrastructure Awareness
- Dynamic Documentation
- Automated NetOps

Network Validation
- Operational State Validation
- Network Compliance
Network Automation Journey

**OPPORTUNISTIC**

How can we simplify a task or set of tasks?

- Backup & Restore
- Dynamic Documentation

**SYSTEMATIC**

How do we centralise our processes?

- Scoped Config Management
- Network Compliance

**INSTITUTIONALIZED**

How do we orchestrate our processes?

- Operational State Validation
- Automated NetOps

Ansible Network Automation

Complexity
Start Small

Quick automation victories for network engineers

**Config Backup and Restore**
- Ubiquitous first touch use case
  - Gain confidence in automation quickly
  - First steps towards network as code
  - Quickly recover network steady state

**Dynamic Documentation**
- Use Ansible facts to gain information
  - Read-only, no production config change
  - Dynamic Documentation and reporting
  - Understand your network

**Scoped Config Management**
- Focus on high yield victories
  - Automate VLANs, ACLs and SNMP config
  - Introduce source of truth concepts
  - Enforce Configuration policy
Think Big

Institutionalizing automation into your organization

Network Compliance

- Respond quickly and consistently
- Security and config compliance for network
- Remove human error from security responses
- Enforce Configuration policies and hardening

Operational State Validation

- Going beyond config management
- Parsing operational state to structured values
- Schema validation and verification
- Enhance operational workflows

Automated NetOps

- Infrastructure as code
- Data centric automation
- Deploy configuration pipelines
- GitOps for Network Automation

Think Big
Institutionalizing automation into your organization

Network Compliance
- Respond quickly and consistently
  - Security and config compliance for network
  - Remove human error from security responses
  - Enforce Configuration policies and hardening

Operational State Validation
- Going beyond config management
  - Parsing operational state to structured values
  - Schema validation and verification
  - Enhance operational workflows

Automated NetOps
- Infrastructure as code
  - Data centric automation
  - Deploy configuration pipelines
  - GitOps for Network Automation
Section 1
Ansible Basics

Topics Covered:

- Understanding Inventory
- An example Ansible Playbook
Create

The automation lifecycle

- **Content creators**
  - **Build**
  - **Discover**
  - **Trust**

- **Domain experts**

- **Ansible content experience**
- **Red Hat cloud / on-premises**
- **Automation hub**

- **Ansible content domains**
  - **Infrastructure**
    - Linux
    - Windows
  - **Cloud**
  - **Network**
  - **Security**
---
- name: install and start apache
  hosts: web
  become: yes

tasks:
  - name: httpd package is present
    yum:
      name: httpd
      state: latest
  - name: latest index.html file is present
    template:
      src: files/index.html
      dest: /var/www/html/
  - name: httpd is started
    service:
      name: httpd
      state: started
What makes up an Ansible playbook?

- Plays
- Modules
- Plugins
Ansible plays

What am I automating?

What are they?
Top level specification for a group of tasks. Will tell that play which hosts it will execute on and control behavior such as fact gathering or privilege level.

Building blocks for playbooks
Multiple plays can exist within an Ansible playbook that execute on different hosts.

---
- name: install and start apache
  hosts: web
  become: yes
Ansible modules
The “tools in the toolkit”

What are they?
Parametrized components with internal logic, representing a single step to be done. The modules “do” things in Ansible.

Language
Usually Python, or Powershell for Windows setups. But can be of any language.

```
- name: latest index.html file ...
  template:
    src: files/index.html
    dest: /var/www/html/
```
Ansible plugins
The “extra bits”

What are they?
Plugins are pieces of code that augment Ansible’s core functionality. Ansible uses a plugin architecture to enable a rich, flexible, and expandable feature set.

Example become plugin:
```yaml
- name: install and start apache
  hosts: web
  become: yes
```

Example filter plugins:
```yaml
{{ some_variable | to_nice_json }}
{{ some_variable | to_nice_yam1 }}
```
Ansible Inventory

The systems that a playbook runs against

What are they?
List of systems in your infrastructure that automation is executed against

[web]
webserver1.example.com
webserver2.example.com

[db]
dbserver1.example.com

[switches]
leaf01.internal.com
leaf02.internal.com
Ansible roles

Reusable automation actions

What are they?

Group your tasks and variables of your automation in a reusable structure. Write roles once, and share them with others who have similar challenges in front of them.

```yaml
---
- name: install and start apache
  hosts: web
  roles:
    - common
    - webservers
```
Collections

Simplified and consistent content delivery

What are they?

Collections are a data structure containing automation content:

- Modules
- Playbooks
- Roles
- Plugins
- Docs
- Tests
---

- name: Install NGINX Plus
  hosts: all
  tasks:
  - name: Install NGINX
    include_role:
      name: nginxinc.nginx
      vars:
        nginx_type: plus

  - name: Install NGINX App Protect
    include_role:
      name: nginxinc.nginx_app_protect
      vars:
        nginx_app_protect_setup_license: false
        nginx_app_protect_remove_license: false
        nginx_app_protect_install_signatures: false
90+
certified platforms
How is network automation different?
Network Automation compared to servers

Module code is executed locally on the control node

Module code is copied to the managed node, executed, then removed

Network Devices / API Endpoints

Local Execution

Remote Execution

Linux / Windows Hosts
Network Connection Plugins

Use your vendor connection of choice

**ansible_connection**

- **netconf** - XML over netconf
  
  example: Juniper Junos

- **network_cli** - command line over SSH
  
  example: Cisco IOS-XE, Arista EOS

- **httpapi** - vendor API
  
  example: Arista eAPI, Cisco NX-API

[Diagram showing Ansible Automation Platform with connections to Cisco IOS-XE, Arista EOS, Juniper Junos, and Red Hat Enterprise Linux]

https://docs.ansible.com/ansible/latest/plugins/connection.html
Understanding Inventory

```plaintext
rtr1 ansible_host=18.220.156.59
rtr2 ansible_host=18.221.53.11
rtr3 ansible_host=13.59.242.237
rtr4 ansible_host=3.16.82.231
rtr5
rtr6
```
Understanding Inventory - Groups

There is always a group called "all" by default

```
[cisco]
rtr1 ansible_host=18.220.156.59 private_ip=172.16.184.164

[arista]
rtr2 ansible_host=18.221.53.11 private_ip=172.17.229.213
rtr4 ansible_host=3.16.82.231 private_ip=172.17.209.186

[juniper]
rtr3 ansible_host=13.59.242.237 private_ip=172.16.39.75
```

Groups can be nested

```
[routers:children]
cisco
juniper
arista
```
Understanding Inventory - Variables

[cisco]
rtr1 ansible_host=18.220.156.59 private_ip=172.16.184.164
[arista]
rtr2 ansible_host=18.221.53.11 private_ip=172.17.229.213
rtr4 ansible_host=3.16.82.231 private_ip=172.17.209.186
[juniper]
rtr3 ansible_host=13.59.242.237 private_ip=172.16.39.75

[cisco:vars]
ansible_user=ec2-user
ansible_network_os=ios
ansible_connection=network_cli

Host variables apply to the host and override group vars

Group variables apply for all devices in that group
A playbook is a list of plays.
Each play is a list of tasks.
Tasks invoke modules.
A playbook can contain more than one play.
Lab Time
Exercise 1 - Exploring the lab environment

red.ht/network-workshop-1

In this lab you will explore the lab environment and build familiarity with the lab inventory.

Approximate time: 10 mins
Section 2
Executing Ansible

Topics Covered:

- An Ansible Play
- Ansible Modules
- Execution Environments
- Running an Ansible Playbook
Automation Execution Environments

Components needed for automation, packaged in a cloud-native way

- Execution Environments
- Collections
- Libraries
- Universal Base Image
- Ansible Core
Build, create, publish

Development cycle of an automation execution environment

Content Creator -> Execution environment builder

Collections

Dependencies

UBI

Ansible Core

Execution Environment

Private automation hub
Develop, test, run

How to develop, test and run containerized Ansible content
Builder and Navigator

Content Creator

ansible-builder

pull/create

ansible-navigator

execute

Execution Environment

playbook

Execution Environment
Another Ansible Playbook Example

---
- name: snmp ro/rw string configuration
  hosts: cisco
  gather_facts: false

tasks:
- name: ensure snmp strings are present
cisco.ios.config:
  lines:
  - snmp-server community ansible-public RO
  - snmp-server community ansible-private RW
The name parameter describes the Ansible Play

- Target devices using the hosts parameter
- Optionally disable gather_facts

---

- name: snmp ro/rw string configuration
  hosts: cisco
  gather_facts: false
Modules

Modules do the actual work in Ansible, they are what gets executed in each playbook task.

- Typically written in Python (but not limited to it)
- Modules can be idempotent
- Modules take user input in the form of parameters

```yaml
tasks:
- name: ensure snmp strings are present
  cisco.ios.config:
    lines:
    - snmp-server community ansible-public RO
    - snmp-server community ansible-private RW
```
Network modules

Ansible modules for network automation typically reference the vendor OS followed by the module name.

- namespace.collection.facts
- namespace.collection.command
- namespace.collection.config
- namespace.collection.resource

More modules depending on platform:

- Arista EOS = arista.eos.
- Cisco IOS/IOS-XE = cisco.ios
- Cisco NX-OS = cisco.nxos
- Cisco IOS-XR = cisco.iosxr
- F5 BIG-IP = f5networks.f5_bigip_bigip
- Juniper Junos = junipsnetworks.junos
- VyOS = vyos.vyos
A playbook run

Where it all starts

- A playbook is interpreted and run against one or multiple hosts – task by task. The order of the tasks defines the execution.

- In each task, the module does the actual work.
Running an Ansible Playbook

Using the latest ansible-navigator command

What is ansible-navigator?

ansible-navigator command line utility and text-based user interface (TUI) for running and developing Ansible automation content.

It replaces the previous command used to run playbooks “ansible-playbook”.

$ ansible-navigator run playbook.yml
ansible-navigator

Bye ansible-playbook, Hello ansible-navigator

How do I use ansible-navigator?

As previously mentioned, it replaces the ansible-playbook command.
As such it brings two methods of running playbooks:

- Direct command-line interface
- Text-based User Interface (TUI)

# Direct command-line interface method
$ ansible-navigator run playbook.yml -m stdout

# Text-based User Interface method
$ ansible-navigator run playbook.yml
ansible-navigator

Mapping to previous Ansible commands

<table>
<thead>
<tr>
<th>ansible command</th>
<th>ansible-navigator command</th>
</tr>
</thead>
<tbody>
<tr>
<td>ansible-config</td>
<td>ansible-navigator config</td>
</tr>
<tr>
<td>ansible-doc</td>
<td>ansible-navigator doc</td>
</tr>
<tr>
<td>ansible-inventory</td>
<td>ansible-navigator inventory</td>
</tr>
<tr>
<td>ansible-playbook</td>
<td>ansible-navigator run</td>
</tr>
</tbody>
</table>
Common subcommands

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>CLI Example</th>
<th>Colon command within TUI</th>
</tr>
</thead>
<tbody>
<tr>
<td>collections</td>
<td>Explore available collections</td>
<td>ansible-navigator collections --help</td>
<td>:collections</td>
</tr>
<tr>
<td>config</td>
<td>Explore the current ansible configuration</td>
<td>ansible-navigator config --help</td>
<td>:config</td>
</tr>
<tr>
<td>doc</td>
<td>Review documentation for a module or plugin</td>
<td>ansible-navigator doc --help</td>
<td>:doc</td>
</tr>
<tr>
<td>images</td>
<td>Explore execution environment images</td>
<td>ansible-navigator images --help</td>
<td>:images</td>
</tr>
<tr>
<td>inventory</td>
<td>Explore and inventory</td>
<td>ansible-navigator inventory --help</td>
<td>:inventory</td>
</tr>
<tr>
<td>replay</td>
<td>Explore a previous run using a playbook artifact</td>
<td>ansible-navigator replay --help</td>
<td>:replay</td>
</tr>
<tr>
<td>run</td>
<td>Run a playbook</td>
<td>ansible-navigator run --help</td>
<td>:run</td>
</tr>
<tr>
<td>welcome</td>
<td>Start at the welcome page</td>
<td>ansible-navigator welcome --help</td>
<td>:welcome</td>
</tr>
</tbody>
</table>
Running a playbook

---

- name: snmp ro/rw string configuration
  hosts: cisco
  gather_facts: false

  tasks:
  - name: ensure snmp strings are present
    cisco.ios.config:
      lines:
      - snmp-server community ansible-public RO
      - snmp-server community ansible-private RW

[student1@ansible networking-workshop]$ ansible-navigator playbook.yml --mode stdout

PLAY [snmp ro/rw string configuration] *******************************************************
TASK [ensure snmp strings are present] *********************************************************
changed: [rtr1]

PLAY RECAP *****************************************************************************************
rtr1                : ok=1    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
[student1@ansible networking-workshop]$ ansible-navigator playbook.yml --mode stdout -v
Using /home/student1/.ansible.cfg as config file

PLAY [snmp ro/rw string configuration] *****************************************************************

TASK [ensure that the desired snmp strings are present] ********************************************************************
changed: [rtr1] => changed=true
  ansible_facts:
    discovered_interpreter_python: /usr/bin/python
  banners: {}  
  commands:
  - snmp-server community ansible-public RO
  - snmp-server community ansible-private RW
  updates:
  - snmp-server community ansible-public RO
  - snmp-server community ansible-private RW

PLAY RECAP ********************************************************************************************
rtr1    : ok=1  changed=1  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0

Increase the level of verbosity by adding more "v's" -vvvv
Exercise 2 - Execute your first network automation playbook

In this lab you will use Ansible to update the configuration of routers. This exercise will not have you create an Ansible Playbook; you will use an existing one.

Approximate time: 15 mins
Section 3
Network Facts

Topics Covered:

▸ Ansible Documentation
▸ Facts for Network Devices
▸ The debug module
Ansible for Network Automation

Ansible Network modules extend the benefits of simple, powerful, agentless automation to network administrators and teams. Ansible Network modules can configure your network stack, test and validate existing network state, and discover and correct network configuration drift.

If you're new to Ansible, or new to using Ansible for network management, start with Getting Started with Ansible for Network Automation. If you are already familiar with network automation with Ansible, see Advanced Topics with Ansible for Network Automation.

For documentation on using a particular network module, consult the list of all network modules. Some network modules are maintained by the Ansible community - here's a list of network modules maintained by the Ansible Network Team.

- Getting Started with Ansible for Network Automation
  - Basic Concepts
    - Control Node
    - Managed Nodes
    - Inventory
    - Modules
    - Tasks
    - Playbooks
  - How Network Automation is Different
    -Execution on the Control Node
    - Multiple Communication Protocols
    - Modules Organized by Network Platform
    - Privilege Escalation: `enable`, `nested`, `become`, and `become_user`
  - Run Your First Command and Playbook
    - Prerequisites
    - Install Ansible
    - Establish a Manual Connection to a Managed Node
    - Run Your First Network Ansible Command
    - Create and Run Your First Network Ansible Playbook
    - Build Your Inventory

red.ht/NetworkDocs
Module Documentation

- Documentation is required as part of module submission

- Multiple Examples for every module

- Broken into relevant sections

---

https://docs.ansible.com/
Accessing the Ansible docs

With the use of the latest command utility `ansible-navigator`, one can trigger access to all the modules available to them as well as details on specific modules.

A formal introduction to `ansible-navigator` and how it can be used to run playbooks in the following exercise.
Fact modules

- Arista EOS → arista.eos.facts
- Cisco IOS → cisco.ios.facts
- Juniper Junos → junipernetworks.junos.facts
What are facts?
Structured data, the Ansible way

---

Cisco IOS output:

```
cisco# show version
Cisco IOS XE Software, Version 16.09.02
Cisco IOS Software [Fuji], Virtual XE Software
(X86_64_LINUX_IOSD-UNIVERSALK9-M), Version 16.9.2,
RELEASE SOFTWARE (fc4)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2018 by Cisco Systems, Inc.

<<rest of output removed for slide brevity>>
```
Ansible Automation Platform facts

Network automation begins and ends with facts

Network native configuration ➔ Convert to structured data ➔

"ansible_facts": {
    "ansible_net_iostype": "IOS-XE",
    "ansible_net_version": "16.09.02",
    "ansible_net_serialnum": "9L8KQ482JFZ",
    "ansible_net_model": "CSR1000V",

<<rest of output removed for brevity>>
Displaying output - The “debug” module

The **debug** module is used like a "print" statement in most programming languages. Variables are accessed using "{{{ }}}" - quoted curly braces

```yaml
- name: display version
debug:
  msg: "The IOS version is: {{ ansible_net_version }}"

- name: display serial number
debug:
  msg: "The serial number is: {{ ansible_net_serialnum }}"
```
Working with Ansible facts

1. Gather facts

- name: gather eos facts
  arista.eos.facts:
    gather_subset: config
    gather_network_resources: vlans

2. Use facts

- name: print out vlans
  debug:
    var: ansible_network_resources.vlans

or

- name: gather eos facts
  arista.eos.vlans:
    state: gathered
    registered: vlanfacts

- name: print out vlans
  debug:
    var: vlanfacts
Simple and common approach

Arista EOS

- name: retrieve eos facts
arista.eos.facts:
gather_subset: config
gather_network_resources: all

Cisco IOS-XE

- name: retrieve ios facts
cisco.ios.facts:
gather_subset: config
gather_network_resources: all

Juniper Junos

- name: retrieve junos facts
junipernetworks.junos.facts:
gather_subset: config
gather_network_resources: all
Working with Ansible facts

2. Use facts

- name: print out vlans
debug:
  var: ansible_network_resources.vlans

or

- name: print out vlans
debug:
  var: vlanfacts

3. Displayed Results

- name: dmz
  state: active
  vlan_id: 5
- name: voip
  state: active
  vlan_id: 10
- name: desktop
  state: active
  vlan_id: 30
Running the Ansible Playbook with verbosity

$ ansible-navigator run facts.yml --mode stdout

PLAY [gather information from routers] *******************************************************
TASK [gather router facts] **********************************************************************
ok: [rtr1]

TASK [display version] **************************************************************************
ok: [rtr1] =>
  msg: 'The IOS version is: 16.09.02'

TASK [display serial number] ********************************************************************
ok: [rtr1] =>
  msg: The serial number is: 964A1H0D1RM

PLAY RECAP ********************************************************************************
  rtr1    : ok=3    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
Structured data is malleable
Create customized network reports

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>
## Build reports with Ansible Facts

<table>
<thead>
<tr>
<th>Hostname</th>
<th>Model Type</th>
<th>Mgmt0 IP Address</th>
<th>Code Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>n9k</td>
<td>Nexus9000 9000v Chassis</td>
<td>192.168.2.3</td>
<td>7.0(3)</td>
</tr>
<tr>
<td>n9k2</td>
<td>Nexus9000 9000v Chassis</td>
<td>192.168.2.4</td>
<td>7.0(3)</td>
</tr>
<tr>
<td>n9k3</td>
<td>Nexus9000 9000v Chassis</td>
<td>192.168.2.5</td>
<td>7.0(3)</td>
</tr>
<tr>
<td>n9k4</td>
<td>Nexus9000 9000v Chassis</td>
<td>192.168.2.6</td>
<td>7.0(2)</td>
</tr>
<tr>
<td>n9k5</td>
<td>Nexus9000 9000v Chassis</td>
<td>192.168.2.7</td>
<td>7.0(3)</td>
</tr>
<tr>
<td>n9k6</td>
<td>Nexus9000 9000v Chassis</td>
<td>192.168.2.8</td>
<td>7.0(3)</td>
</tr>
</tbody>
</table>
Lab Time
Exercise 3 - Ansible Facts

- **red.ht/network-workshop-3**
  Demonstration use of Ansible facts on network infrastructure.

- Approximate time: 15 mins
Section 4
Resource Modules

Topics Covered:

- Resource modules
- state: merged
- state: gathered
Network Automation Modules

How do we interact with network devices?

- **command**: run arbitrary commands
- **facts**: retrieve information
- **config**: generic catch-all configuration and templating
- **resource**: read and configure specific network resources
Network Automation Modules

How do we interact with network devices?

- **command**
  - namespace.collection.command
  - Cisco IOS -> cisco.ios.command

- **facts**
  - namespace.collection.facts
  - Arista EOS -> arista.eos.facts

- **config**
  - namespace.collection.config
  - Juniper Junos -> junipernetworks.junos.config

- **resource**
  - namespace.collection.module
  - Cisco IOS-XR -> cisco.ioxr.acls
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
This exercise will cover configuring VLANs on Arista EOS by building an Ansible Playbook using the arista.eos.vlans module.

Approximate time: 15 mins
Section 5
Automation controller

Topics Covered:

- What is Automation controller?
- Enterprise Features
What makes a platform?

Red Hat
Ansible Automation Platform

<table>
<thead>
<tr>
<th>Content creators</th>
<th>Operators</th>
<th>Domain experts</th>
<th>Users</th>
</tr>
</thead>
</table>

Fueled by an open source community

<table>
<thead>
<tr>
<th>On-premises</th>
<th>Ansible Cloud Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automation controller</td>
<td>Automation hub</td>
</tr>
<tr>
<td>Automation services catalog</td>
<td>Insights for Ansible Automation Platform</td>
</tr>
</tbody>
</table>

Ansible content domains

- **Infrastructre**
  - Linux
  - Windows
- **Cloud**
- **Network**
- **Security**

Ansible command line
Ansible Automation Controller is a UI and RESTful API allowing you to scale IT automation, manage complex deployments and speed productivity.

- Role-based access control
- Deploy entire applications with push-button deployment access
- All automations are centrally logged
- Powerful workflows match your IT processes
Automation controller

**Push button**
An intuitive user interface experience makes it easy for novice users to execute playbooks you allow them access to.

**RESTful API**
With an API first mentality every feature and function of controller can be API driven. Allow seamless integration with other tools like ServiceNow and Infoblox.

**RBAC**
Allow restricting playbook access to authorized users. One team can use playbooks in check mode (read-only) while others have full administrative abilities.

**Enterprise integrations**

**Centralized logging**
All automation activity is securely logged. Who ran it, how they customized it, what it did, where it happened - all securely stored and viewable later, or exported through Automation controllers API.

**Workflows**
Automation controller’s multi-playbook workflows chain any number of playbooks, regardless of whether they use different inventories, run as different users, run at once or utilize different credentials.
Lab Time

Exercise 5: Explore Automation controller

red.ht/network-workshop-5

Explore and understand the Automation controller lab environment.

Approximate time: 15 mins
Section 6
Job Templates

Topics Covered:

▸ Job Templates
  • Inventory
  • Credentials
  • Projects
Anatomy of an Automation Job

Project

Git / Subversion

Playbook
Anatomy of an Automation Job

- Playbook
- Git / Subversion
- Credential
- Project
Anatomy of an Automation Job

- Playbook
- Git / Subversion
- Credential
- Project
- Inventory

Logos and tools:
- CyberArk
- HashiCorp vault
- Infoblox
- DEVICE42
- GitHub
- netbox

Red Hat Ansible Automation Platform
Everything in Automation Controller revolves around the concept of a Job Template. Job Templates allow Ansible Playbooks to be controlled, delegated and scaled for an organization.

Job templates also encourage the reuse of Ansible Playbook content and collaboration between teams.

A Job Template requires:

- A **Project** which contains Ansible Playbooks
- An **Inventory** to run the job against
- A **Credential** to login to devices.
A project is a logical collection of Ansible Playbooks, represented in Ansible Automation Controller.

You can manage Ansible Playbooks and playbook directories by placing them in a source code management system supported by Automation controller including Git, and Subversion.
Inventory

Inventory is a collection of hosts (nodes) with associated data and groupings that Automation Controller can connect to and manage.

- Hosts (nodes)
- Groups
- Inventory-specific data (variables)
- Static or dynamic sources
Credentials

Credentials are utilized by Automation Controller for authentication with various external resources:

- Connecting to remote machines to run jobs
- Syncing with inventory sources
- Importing project content from version control systems
- Connecting to and managing network devices

Centralized management of various credentials allows end users to leverage a secret without ever exposing that secret to them.
Expanding on Job Templates

Job Templates can be found and created by clicking the **Templates** button under the *Resources* section on the left menu.
Executing an existing Job Template

Job Templates can be launched by clicking the **rocketship button** for the corresponding Job Template.
Creating a new Job Template (1/2)

New Job Templates can be created by clicking the **Add button**

![Image of a job template creation interface]
Creating a new Job Template (2/2)

This **New Job Template** window is where the inventory, project and credential are assigned. The red asterisk * means the field is required.
Lab Time

Exercise 6: Creating an Automation controller Job Template

- [red.ht/network-workshop-6](red.ht/network-workshop-6)
  Demonstrate a network backup configuration job template with Automation controller.

- Approximate time: 15 mins
Section 7
Survey

Topics Covered:

▸ Understanding Extra Vars
▸ Building a Survey
▸ Self-service IT with Surveys
Surveys

Controller surveys allow you to configure how a job runs via a series of questions, making it simple to customize your jobs in a user-friendly way.

An Ansible Controller survey is a simple question-and-answer form that allows users to customize their job runs. Combine that with Controller’s role-based access control, and you can build simple, easy self-service for your users.
Creating a Survey (1/2)

Once a job template is saved, the survey menu will have an **Add** button. Click the button to open the **Add Survey** window.
Creating a Survey (2/2)

The **Add Survey** window allows the job template to prompt users for one or more questions. The answers provided become variables for use in the Ansible Playbook.
Using a Survey

When launching a job, the user will now be prompted with the survey. The user can be required to fill out the survey before the job template will execute.
Lab Time

Exercise 7: Creating a Survey

- red.ht/network-workshop-7
  
  Demonstrate the use of Automation controller survey feature.


- Approximate time: 15 mins
Section 8
RBAC

Topics Covered:
- Understanding Organizations
- Understanding Teams
- Understanding Users
Role-based access control

How to manage access

- Role-based access control system:
  Users can be grouped in teams, and roles can be assigned to the teams.

- Rights to edit or use can be assigned across all objects.

- All backed by enterprise authentication if needed.
User Management

- An **organization** is a logical collection of users, teams, projects, inventories and more. All entities belong to an organization.

- A **user** is an account to access Ansible Automation Controller and its services given the permissions granted to it.

- **Teams** provide a means to implement role-based access control schemes and delegate responsibilities across organizations.
Viewing Organizations

Clicking on the **Organizations** button in the left menu will open up the Organizations window.
Viewing Teams

Clicking on the **Teams** buttons in the left menu will open up the Teams window.
Viewing Users

Clicking on the **Users** button in the left menu will open up the Users window.
Lab Time
Exercise 8: Understanding RBAC in Automation controller

red.ht/network-workshop-8
Demonstrate the use of role based access control on Automation controller.

Approximate time: 15 mins
Section 9
Workflows

Topics Covered:

▸ Understanding Workflows
▸ Branching
▸ Convergence / Joins
▸ Conditional Logic
Lab Time

Exercise 9: Creating a Workflow

red.ht/network-workshop-9

Demonstrate the use of Automation Controller workflow. Workflows allow you to configure a sequence of disparate job templates (or workflow templates) that may or may not share inventory, playbooks, or permissions.

Approximate time: 15 mins
Workflows

Combine automation to create something bigger

- Workflows enable the creation of powerful holistic automation, chaining together multiple pieces of automation and events.

- Simple logic inside these workflows can trigger automation depending on the success or failure of previous steps.
Adding a New Template

- To add a new **Workflow** click on the **Add** button.
  This time select the **Add workflow template**
Creating the Workflow

- Fill out the required parameters and click **Save**. As soon as the Workflow Template is saved the Workflow Visualizer will open.
Workflow Visualizer

- The Workflow Visualizer will start as a blank canvas.
- Click the green Start button to start building the workflow.
Ansible Automation Platform

Using workflows to enhance your automation

WORKFLOW VISUALIZER | Operational State Workflow
Wrapping up

Topics Covered:

- Next Steps
- Chat with us
- Consulting Services
Where to go next

Learn more
- Workshops
- Documents
- Youtube
- Twitter

Get started
- Evals
- cloud.redhat.com

Get serious
- Red Hat Automation Adoption Journey
- Red Hat Training
- Red Hat Consulting
Chat with us

- **Slack**
  [https://ansiblenetwork.slack.com](https://ansiblenetwork.slack.com)

- **IRC**
  #ansible-network on freenode
  [http://webchat.freenode.net/?channels=ansible-network](http://webchat.freenode.net/?channels=ansible-network)
Bookmark the Github organization

- Examples, samples and demos
- Run network topologies right on your laptop
## Red Hat Services

Accelerate standardization and automation of network configuration

<table>
<thead>
<tr>
<th>Challenge</th>
<th>Approach</th>
<th>Benefits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slow</td>
<td>Automate</td>
<td>Speed</td>
</tr>
<tr>
<td>Time consuming, labor intensive procedures to propagate network changes</td>
<td>Encode and execute procedures with human-readable Ansible playbooks</td>
<td>Reduce changes from days to hours and drive simultaneous config across 100s of endpoints</td>
</tr>
<tr>
<td>Chaos</td>
<td>Standardize</td>
<td>Efficiency</td>
</tr>
<tr>
<td>Rising number of devices, environments, and vendor-specific tooling create sprawl and skills gaps</td>
<td>Automate common tasks using Ansible modules to abstract vendor-specific details</td>
<td>Easily combine and execute complex configuration procedures across environments</td>
</tr>
<tr>
<td>Errors</td>
<td>Test</td>
<td>Reliability</td>
</tr>
<tr>
<td>Over time, vulnerabilities, patches, and mistakes undermine known-good configurations.</td>
<td>Iteratively refine and validate provisioning and configuration pre-PROD</td>
<td>Eliminate human error in production changes</td>
</tr>
<tr>
<td>Mystery</td>
<td>Catalog</td>
<td>Manageability</td>
</tr>
<tr>
<td>No living source of truth for which patches, packages, or configurations are deployed where</td>
<td>Automate configuration reporting, inventory, and change tracking across all environments</td>
<td>Centrally track and manage configuration rollout, drift, patching, and compliance</td>
</tr>
</tbody>
</table>
Resources

- Network automation for everyone (Overview)
- Automate your network with Red Hat (Technical)
- Online training: Red Hat Ansible for Network Automation
- Network Automation web page
- Red Hat Ansible Automation Platform blog

red.ht/ansible_trial
Thank you
Supplemental

Topics Covered:

- Understand group variables
- Understand Jinja2
- cli_config module
Group variables

Group variables are variables that are common between two or more devices. Group variables can be associated with an individual group (e.g. “cisco”) or a nested group (e.g. routers).

Examples include

- NTP servers
- DNS servers
- SNMP information

Basically network information that is common for that group
Inventory versus group_vars directory

Group variables can be stored in a directory called `group_vars` in YAML syntax. In exercise one we covered `host_vars` and `group_vars` with relationship to inventory. What is the difference?

**inventory**

Can be used to set variables to connect and authenticate **to the device**.

Examples include:
- Connection plugins (e.g. network_cli)
- Usernames
- Platform types *(ansible_network_os)*

**group_vars**

Can be used to set variables to configure **on the device**.

Examples include:
- VLANs
- Routing configuration
- System services (NTP, DNS, etc)
Examining a group_vars file

At the same directory level as the Ansible Playbook create a folder named `group_vars`. Group variable files can simply be named the group name (in this case `all.yml`)

```
$ cat group_vars/all.yml

nodes:
  rtr1:
    Loopback100: "192.168.100.1"
  rtr2:
    Loopback100: "192.168.100.2"
  rtr3:
    Loopback100: "192.168.100.3"
  rtr4:
    Loopback100: "192.168.100.4"
```
Jinja2

- Ansible has native integration with the Jinja2 templating engine
- Render data models into device configurations
- Render device output into dynamic documentation

Jinja2 enables the user to manipulate variables, apply conditional logic and extend programmability for network automation.
Network Automation config modules

**cli_config** (agnostic)

- ios_config:
- nxos_config:
- iosxr_config:
- eos_config

*os_config:
Jinja2 Templating Example (1/2)

Variables

\begin{align*}
\text{ntp_server}: & \hspace{1em} 192.168.0.250 \\
\text{name_server}: & \hspace{1em} 192.168.0.251
\end{align*}

Jinja2 Template

\begin{verbatim}
! ntp server {{ntp_server}}
! ip name-server {{name_server}}
!
\end{verbatim}

Generated Network Configuration

\begin{verbatim}
! ip name-server 192.168.0.251
! ntp server 192.168.0.250
!
\end{verbatim}

\begin{verbatim}
! ip name-server 192.168.0.251
! ntp server 192.168.0.250
!
\end{verbatim}
Jinja2 Templating Example (2/2)

**Variables**

```python
nodes:
    rtr1:
        Loopback100: "192.168.100.1"
    rtr2:
        Loopback100: "192.168.100.2"
    rtr3:
        Loopback100: "192.168.100.3"
    rtr4:
        Loopback100: "192.168.100.4"
```

**Jinja2 Template**

```jinja2
{% for interface,ip in nodes[inventory_hostname].items() %}
    interface {{interface}}
    ip address {{ip}} 255.255.255.255
{% endfor %}
```

**Generated Network Configuration**

- **rtr1**
  ```
  interface Loopback100
  ip address 192.168.100.1
  !
  ```

- **rtr2**
  ```
  interface Loopback100
  ip address 192.168.100.2
  !
  ```

- **rtrX**
  ```
  interface Loopback100
  ip address X
  !
  ```
The cli_config module

Agnostic module for network devices that uses the network_cli connection plugin.

---

- name: configure network devices
  hosts: rtr1,rtr2
  gather_facts: false
  tasks:
    - name: configure device with config
      cli_config:
        config: "{{ lookup('template', 'template.j2') }}"