Ansible Automation Platform

Jim Garrett
Principal Solution Architect
jgarrett@redhat.com

What You Will Learn

- Overview of public cloud provisioning
- Converting shell commands into Ansible commands
- Retrieving information from hosts
- Deploying applications at scale
- Self-service IT via surveys
- Overview of System Roles for Red Hat Enterprise Linux
- Overview of Red Hat Insights integration

WHAT IS **ANSIBLE**?

Automation Engine

IT Infrastructure

Networks - F5, Arista, Cisco, Juniper, Open Vswitch VYOS, others

WHAT IS **ANSIBLE**?

- It’s an automation engine - **Ansible**

Automate What????

What Can I Do Using Ansible

Automate the deployment and management of your entire IT footprint.

Do this...

- Orchestration
- Configuration Management
- Application Deployment
- Provisioning
- Continuous Delivery
- Security and Compliance

On these...

- Firewalls
- Load Balancers
- Applications
- Containers
- Clouds
- Servers
- Infrastructure
- Storage
- Network Devices
- **And more...**

WHAT IS ANSIBLE?

- It's an automation engine - **Ansible**
- It's an automation language - **Ansible Playbooks**.
- Enterprise framework for controlling, securing and managing your automation – **Ansible Tower**

SIMPLE

Human readable automation

No special coding skills needed

Tasks executed in order

Get productive quickly

POWERFUL

Configuration management

App deployment

Workflow orchestration

Orchestrate the app lifecycle

ANSIBLE

AGENTLESS

Agentless architecture

Uses OpenSSH & WinRM

More efficient & more secure

The Ansible Way

CROSS PLATFORM – Linux, Windows, UNIX
Agentless support for all major OS variants, physical, virtual, cloud and network

HUMAN READABLE – YAML
Perfectly describe and document every aspect of your application environment

PERFECT DESCRIPTION OF APPLICATION
Every change can be made by playbooks, ensuring everyone is on the same page

VERSION CONTROLLED
Playbooks are plain-text. Treat them like code in your existing version control.

DYNAMIC INVENTORIES
Capture all the servers 100% of the time, regardless of infrastructure, location, etc.

ORCHESTRATION THAT PLAYS WELL WITH OTHERS – HP SA, Puppet, Jenkins, RHNSS, etc.
Homogenize existing environments by leveraging current toolsets and update mechanisms.

According to Gartner

Figure 1
Primary Method for Making Network Changes

Source: Gartner, *Look Beyond Network Vendors for Network Innovation*. January 2018. Gartner ID: G00349636. (n=64)

Automatic considerations

- Compute is no longer the slowest link in the chain
- Businesses demand that networks deliver at the speed of cloud
- Automation of repeatable tasks
- Bridge silos

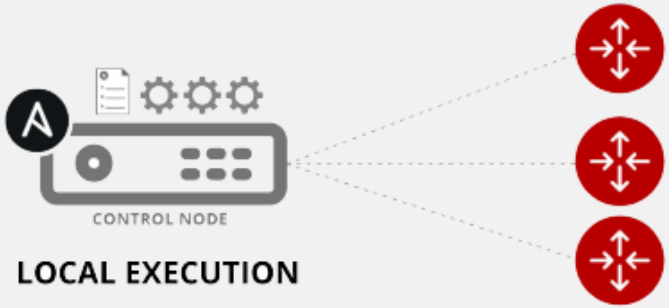
Common Use Cases

Anything an operator can do manually, Ansible can automate

- Backup and restore device configurations
- Upgrade network device OS
- Ensure Configuration Compliance
- Apply patches to address CVE (*Common Vulnerabilities and Exposures*)
- Generate dynamic documentation

How Ansible Works

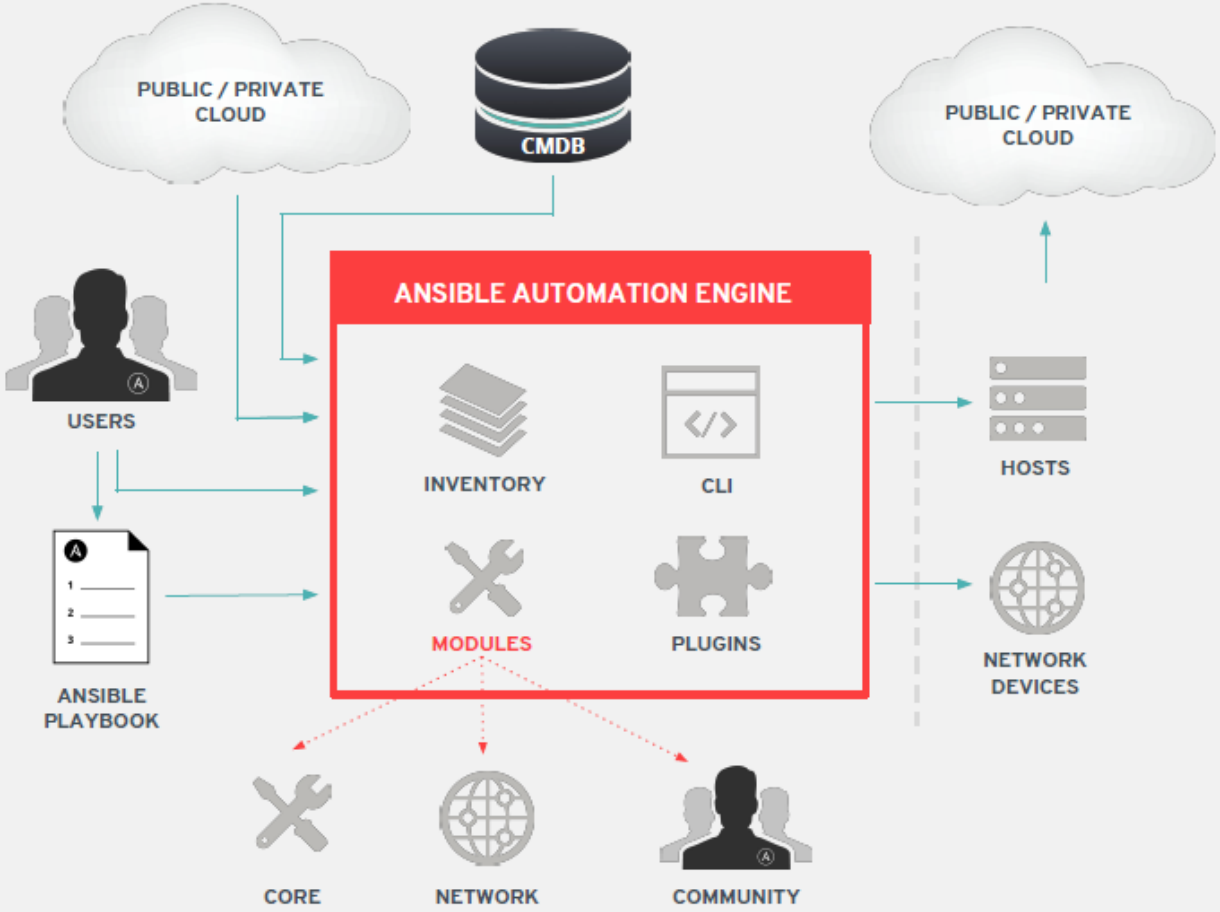
Module code is executed locally on the control node



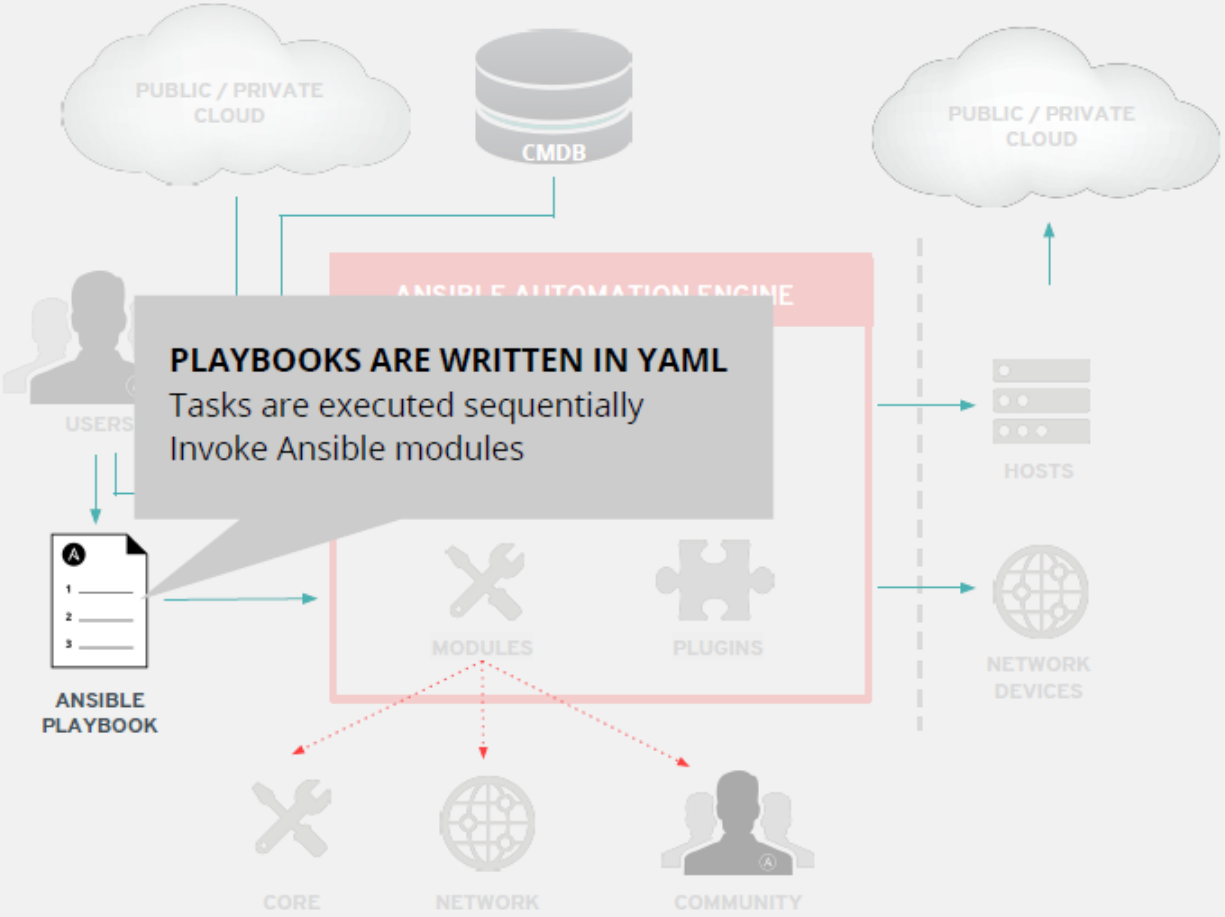
NETWORKING DEVICES

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

LINUX/WINDOWS HOSTS


PUBLIC / PRIVATE CLOUD
CMDB
PUBLIC / PRIVATE CLOUD
ANSIBLE AUTOMATION ENGINE
USERS
INVENTORY
CLI
HOSTS
ANSIBLE PLAYBOOK
MODULES
PLUGINS
NETWORK DEVICES
CORE
NETWORK
COMMUNITY

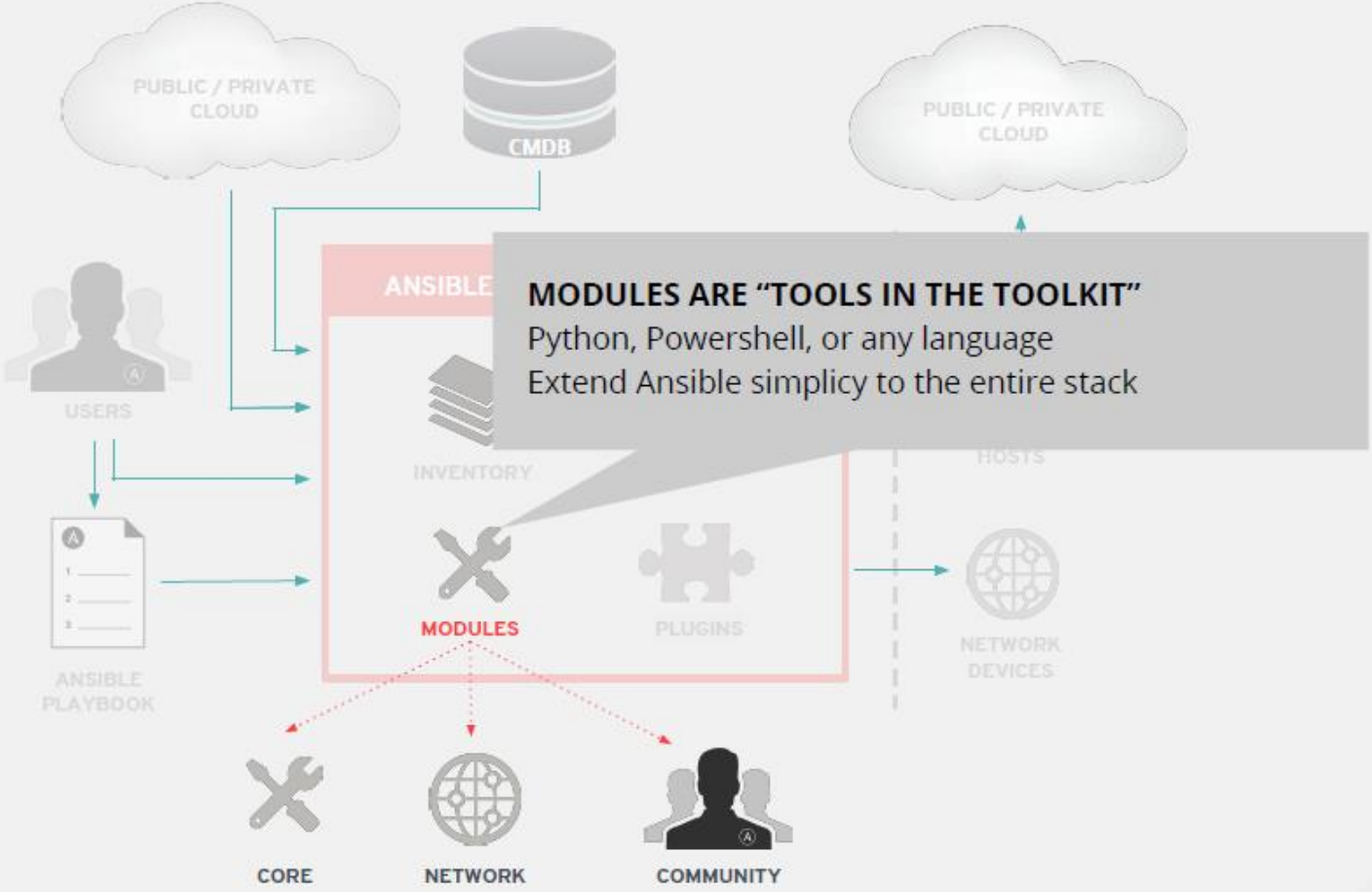
Red Hat



PLAYBOOKS ARE WRITTEN IN YAML

Tasks are executed sequentially

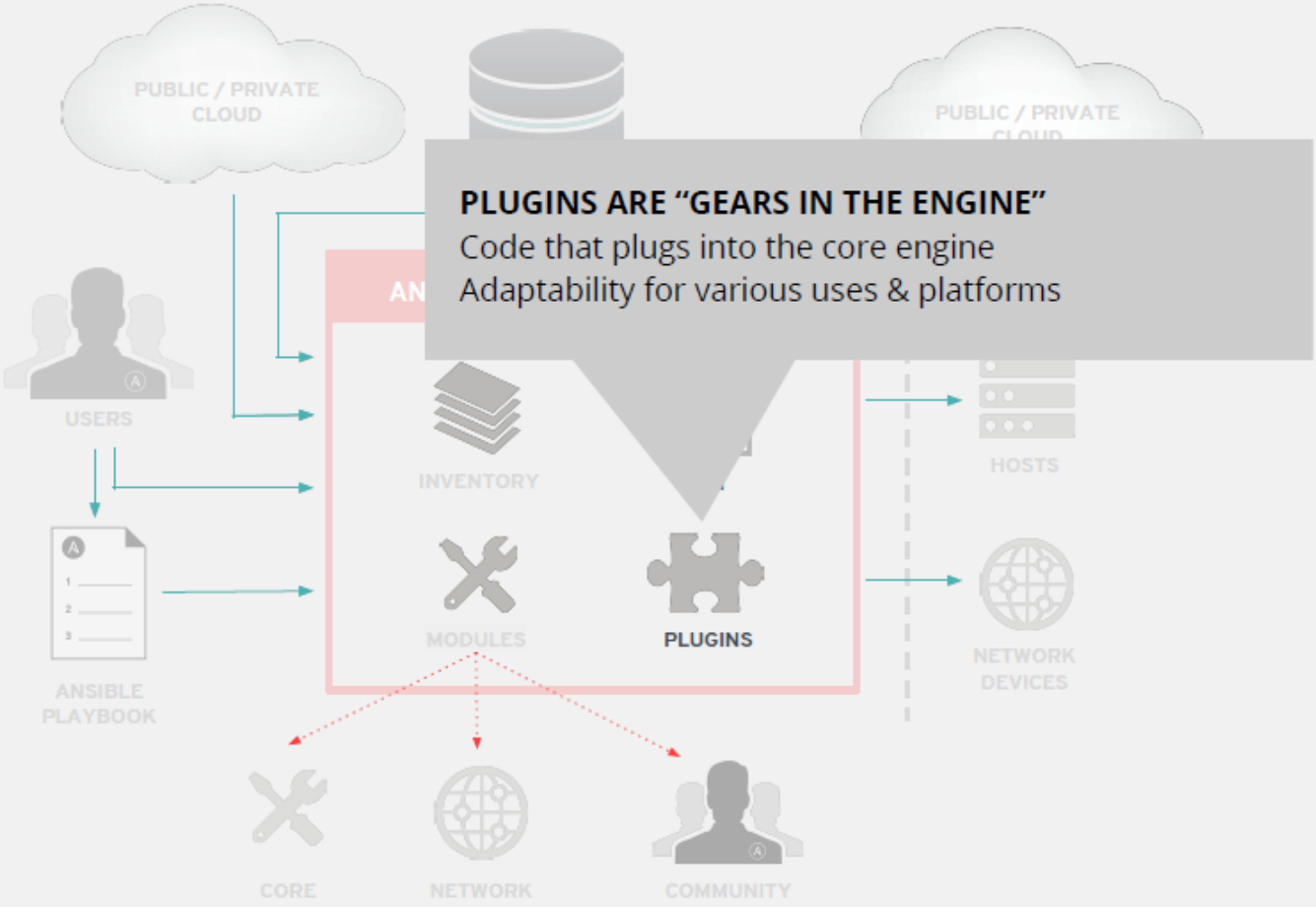
Invoke Ansible modules



MODULES ARE "TOOLS IN THE TOOLKIT"

Python, Powershell, or any language

Extend Ansible simplicy to the entire stack



PLUGINS ARE "GEARS IN THE ENGINE"

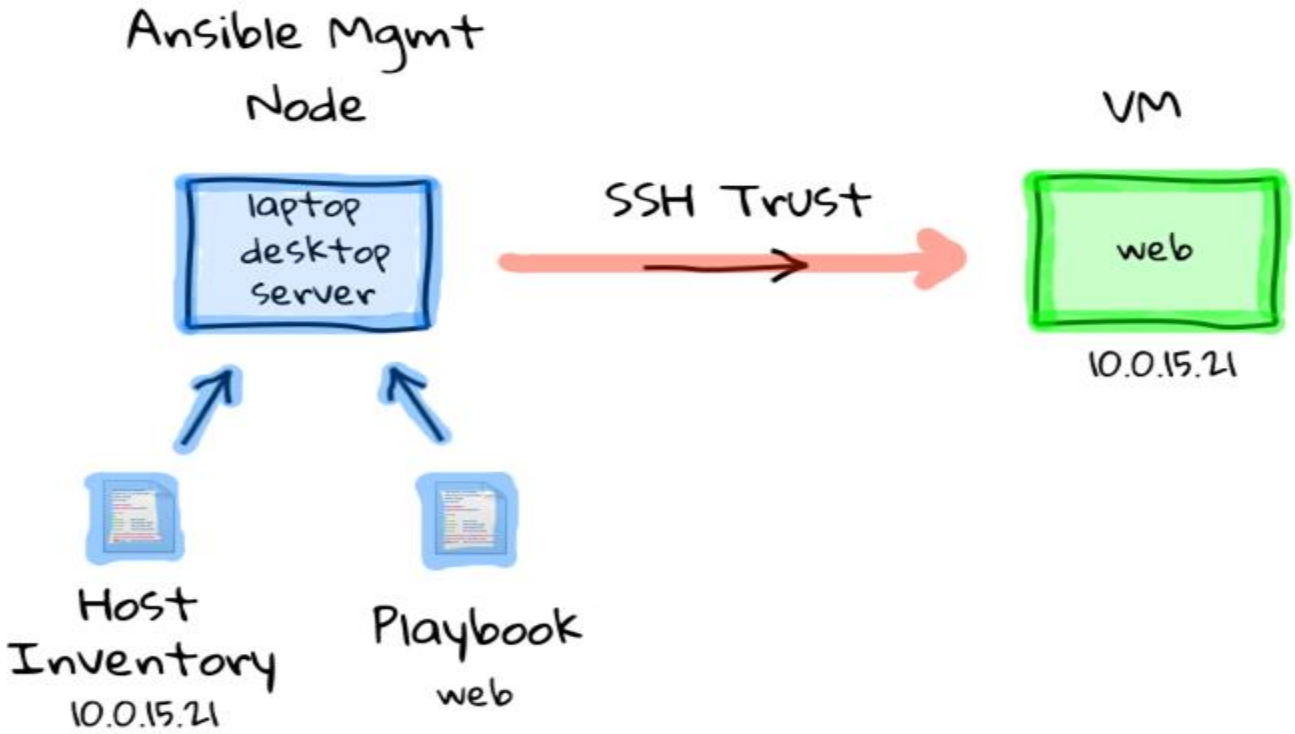
Code that plugs into the core engine

Adaptability for various uses & platforms

How ANSIBLE Works – Simplicity – Command Line

$ ansible all -m ping

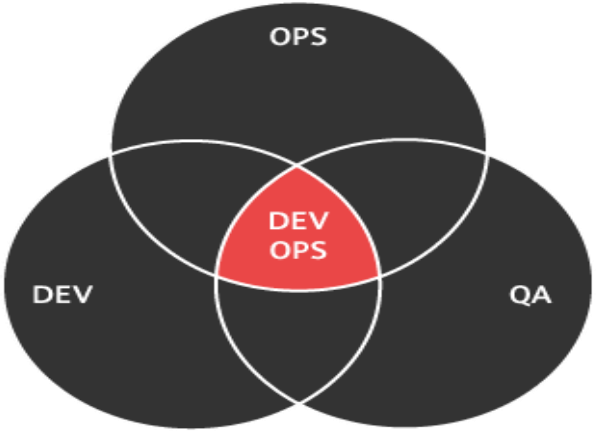
How ANSIBLE Works – Simplicity - playbook



$ ansible-playbook simple_playbook.yml

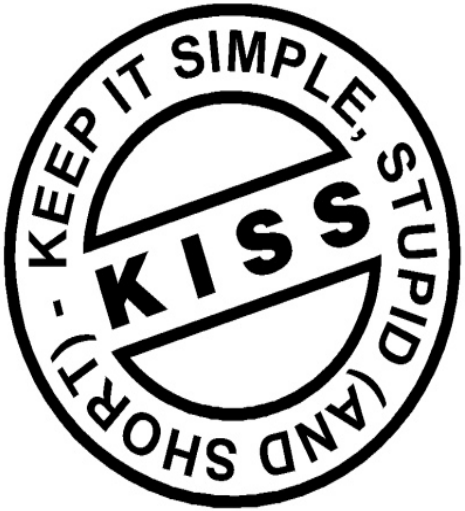
How ANSIBLE Works – Inventory file example

```
[web]
r2d2.example.com
c3po.example.com

[dbservers]
mysql.example.com
postgre.example.com
```



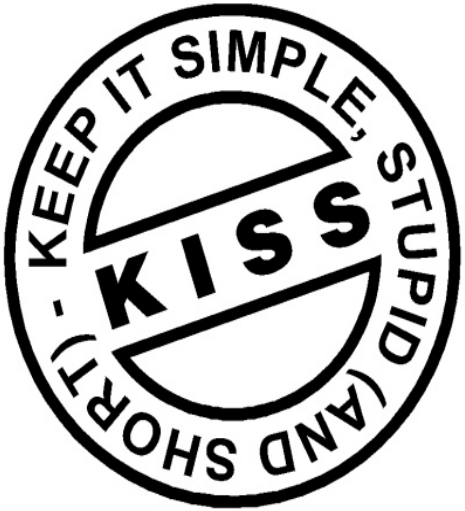
How ANSIBLE Works – Playbook

```
---
- name: install and start apache
  hosts: web
  vars:
    http_port: 80
    max_clients: 200
  remote_user: root

  tasks:
  - name: install httpd
    yum: pkg=httpd state=latest
  - name: write the apache config file
    template: src=/srv/httpd.j2 dest=/etc/httpd.conf
  - name: start httpd
    service: name=httpd state=running
```



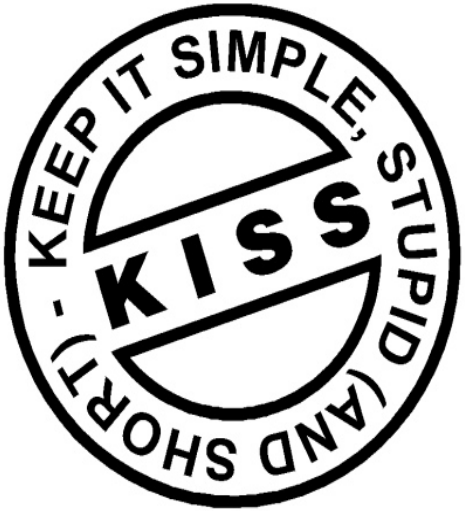
How ANSIBLE Works – Playbook

```
---
- name: install and start apache
  hosts: web
  vars:
    http_port: 80
    max_clients: 200
  remote_user: root

  tasks:
  - name: install httpd
    yum: pkg=httpd state=latest
  - name: write the apache config file
    template: src=/srv/httpd.j2 dest=/etc/httpd.conf
  - name: start httpd
    service: name=httpd state=running
```



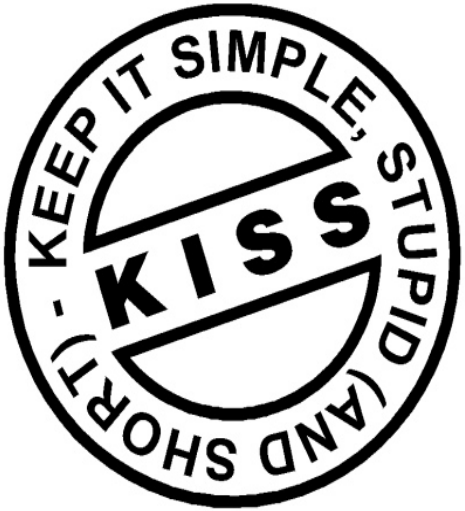
How ANSIBLE Works – Playbook

```
---
- name: install and start apache
  hosts: web
  vars:
      http_port: 80
      max_clients: 200
  remote_user: root

  tasks:
  - name: install httpd
    yum: pkg=httpd state=latest
  - name: write the apache config file
    template: src=/srv/httpd.j2 dest=/etc/httpd.conf
  - name: start httpd
    service: name=httpd state=running
```



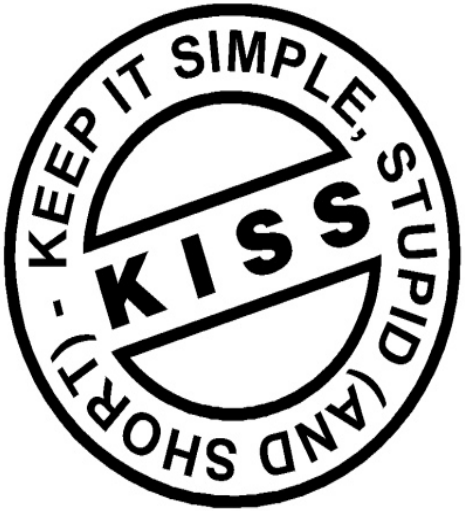
How ANSIBLE Works – Playbook

```
---
- name: install and start apache
  hosts: web
  vars:
    http_port: 80
    max_clients: 200
  remote_user: root

  tasks:
  - name: install httpd
    yum: pkg=httpd state=latest
  - name: write the apache config file
    template: src=/srv/httpd.j2 dest=/etc/httpd.conf
  - name: start httpd
    service: name=httpd state=running
```



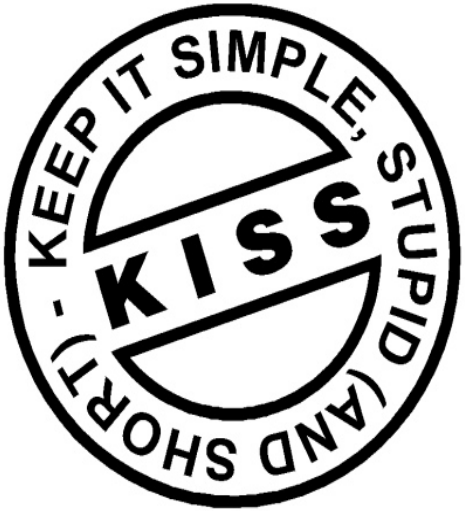
How ANSIBLE Works – Playbook

```
---
- name: install and start apache
  hosts: web
  vars:
    http_port: 80
    max_clients: 200
  remote_user: root

  tasks:
  - name: install httpd
    yum: pkg=httpd state=latest
  - name: write the apache config file
    template: src=/srv/httpd.j2 dest=/etc/httpd.conf
  - name: start httpd
    service: name=httpd state=running
```



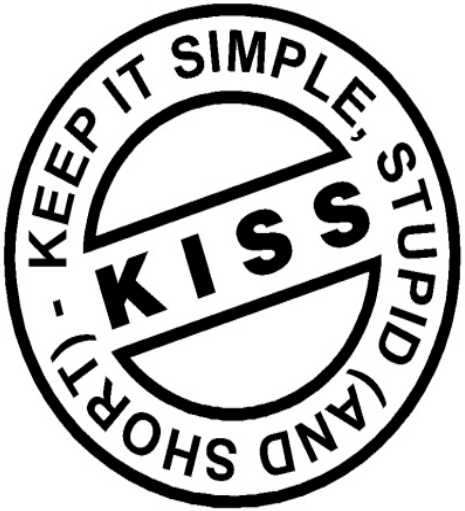
How ANSIBLE Works – Playbook

```
---
- name: install and start apache
  hosts: web
  vars:
    http_port: 80
    max_clients: 200
  remote_user: root

  tasks:
  - name: install httpd
    yum: pkg=httpd state=latest
  - name: write the apache config file
    template: src=/srv/httpd.j2 dest=/etc/httpd.conf
  - name: start httpd
    service: name=httpd state=running
```

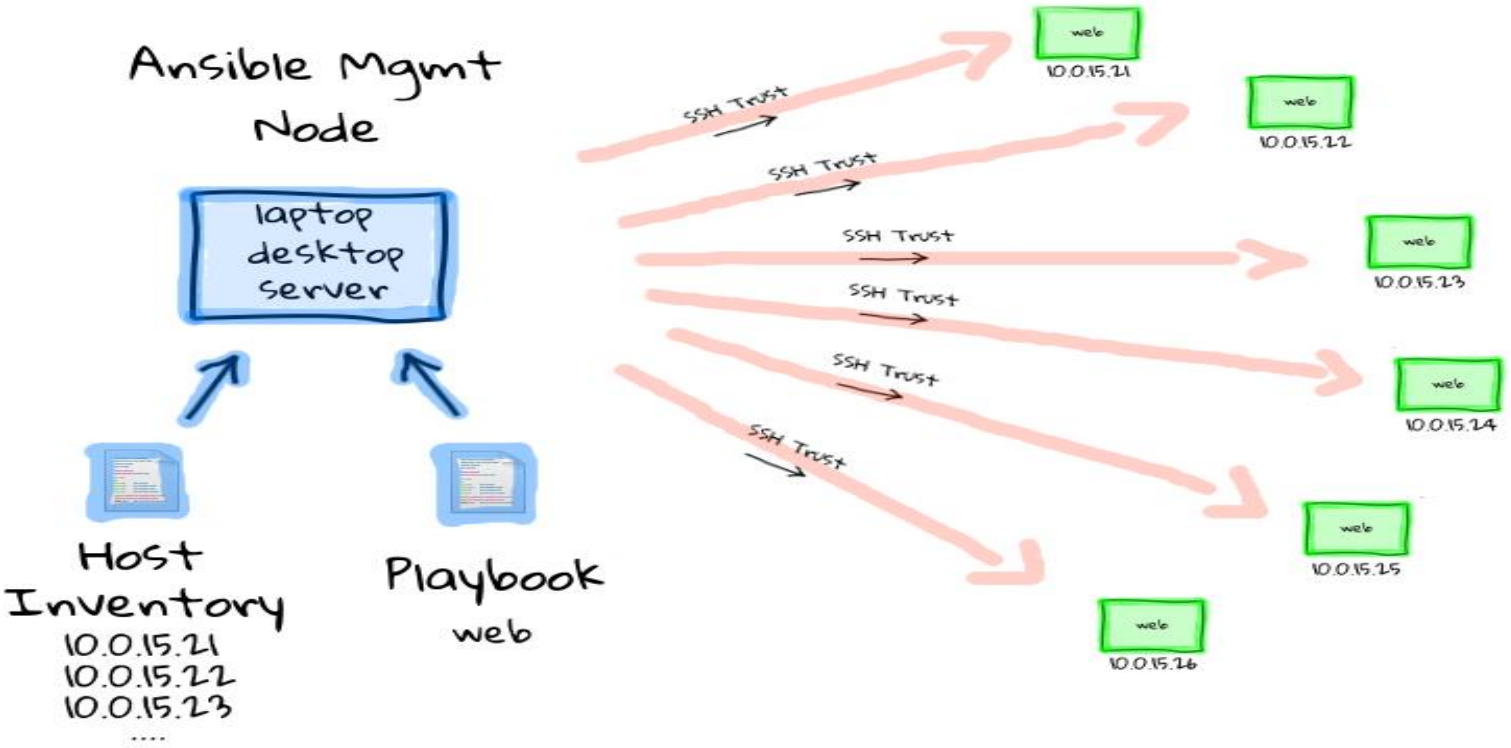


How ANSIBLE Works – Playbook

```
---
- name: install and start apache
  hosts: web
  vars:
    http_port: 80
    max_clients: 200
  remote_user: root

  tasks:
  - name: install httpd
    yum: pkg=httpd state=latest
  - name: write the apache config file
    template: src=/srv/httpd.j2 dest=/etc/httpd.conf
  - name: start httpd
    service: name=httpd state=running
```



How ANSIBLE Works – at scale



$ ansible-playbook simple_playbook.yml

Modules

Modules are bits of code transferred to the target system and executed to satisfy the task declaration.

- apt/yum
- copy
- file
- get_url
- git
- ping
- debug
- service
- synchronize
- template
- uri
- user
- wait_for
- assert

$ ansible all -m ping

Modules Documentation

http://docs.ansible.com/

Docs » Module Index

Module Index

- All Modules
- Cloud Modules
- Clustering Modules
- Commands Modules
- Crypto Modules
- Database Modules
- Files Modules
- Identity Modules
- Inventory Modules
- Messaging Modules
- Monitoring Modules
- Network Modules
- Notification Modules
- Packaging Modules
- Remote Management Modules
- Source Control Modules
- Storage Modules
- System Modules
- Utilities Modules
- Web Infrastructure Modules
- Windows Modules

service - Manage services.

- Synopsis
- Options
- Examples
  - Status
  - Support

Synopsis

- Controls services on remote hosts. Supported init systems include BSD init, OpenRC, SysV, Solaris SMF, systemd, upstart.

Options

parameter	required	default	choices	comments
arguments	no			Additional arguments provided on the command line aliases: args
enabled	no		• yes • no	Whether the service should start on boot. **At least one of state and enabled are required.**
name	yes			Name of the service.
pattern	no			If the service does not respond to the status command, name a substring to look for as would be found in the output of the *ps* command as a stand-in for a status result. If the string is found, the service will be assumed to be running.
runlevel	no	default		For OpenRC init scripts (ex: Gentoo) only. The runlevel that this service belongs to.
sleep (added in 1.3)	no			If the service is being `restarted` then sleep this many seconds between the stop and start command. This helps to workaround badly behaving init scripts that exit immediately after signaling a process to stop.
state	no		• started • stopped • restarted • reloaded	`started` / `stopped` are idempotent actions that will not run commands unless necessary. `restarted` will always bounce the service. `reloaded` will always reload. **At least one of state and enabled are required.** Note that reloaded will start the service if it is not already started, even if your chosen init system wouldn't normally.
use (added in 2.2)	no	auto		The service module actually uses system specific modules, normally through auto detection, this setting can force a specific module. Normally it uses the value of the 'ansible_service_mgr' fact and falls back to the old 'service' module when none matching is found.

Modules Documentation

```
# List out all modules installed
$ ansible-doc -l
...
copy
cron
...

# Read documentation for installed module
$ ansible-doc copy
> COPY

  The [copy] module copies a file on the local box to remote locations. Use the [fetch] module to copy files from remote locations to the loca
  box. If you need variable interpolation in copied files, use the [template] module.

  * note: This module has a corresponding action plugin.

Options (= is mandatory):
```

Modules: Run Commands

If Ansible doesn't have a module that suits your needs there are the “run command” modules:

- **command**: Takes the command and executes it on the host. The most secure and predictable.
- **shell**: Executes through a shell like `/bin/sh` so you can use pipes etc. Be careful.
- **script**: Runs a local script on a remote node after transferring it.
- **raw**: Executes a command without going through the Ansible module subsystem.

NOTE: Unlike standard modules, run commands have no concept of desired state and should only be used as a last resort.

AD-Hoc Commands

# Check all my inventory hosts are ready to be managed
$ ansible all –m ping

# Collect and Display discovered facts for localhost
$ ansible localhost –m setup

# run the uptime command on all hosts in the WEB group
$ ansible web –m command –a “uptime”

Sidebar: Discovered Facts

Facts are bits of information derived from examining a host system, these are stored as variables for later use in the play.

```
$ ansible localhost –m setup
localhost | success >> {
   “ansible_facts”: {
      “ansible_default_ipv4”: {
         “address”: “192.168.1.27”,
         “alias”: “wlan0”,
         “gateway: “192.168.1.1”,
         “interface: “wlan0”,
         “macaddress”: “c4:85:08:3b:a9:16”,
         .
         .
```

Ansible automates technologies you use

Time to automate is measured in minutes

Cloud

AWS
Azure
Digital Ocean
Google
OpenStack
Rackspace
+more

Operating Systems

RHEL
Linux
Windows
+more

Virt & Container

Docker
VMware
RHV
OpenStack
OpenShift
+more

Storage

Netapp
Red Hat Storage
Infinidat
+more

Windows

ACLs
Files
Packages
IIS
Regedits
Shares
Services
Configs
Users
Domains
+more

Network

A10
Arista
Aruba
Cumulus
Bigswitch
Cisco
Dell
Extreme
F5
Lenovo
MikroTik
Juniper
OpenSwitch
+more

Security

Checkpoint
Cisco
CyberArk
F5
Fortinet
Juniper
IBM
Palo Alto
Snort
+more

Monitoring

Dynatrace
Datadog
LogicMonitor
New Relic
Sensu
+more

Devops

Jira
GitHub
Vagrant
Jenkins
Slack
+more

Red Hat Ansible Tower

by the numbers:

94% Reduction in recovery time following a security incident

84% Savings by deploying workloads to generic systems appliances using Ansible Tower

67% Reduction in man hours required for customer deliveries

Financial summary:

146%

ROI on Ansible Tower

<3 MONTHS

Payback on Ansible Tower

SOURCE: "The Total Economic Impact™ Of Red Hat Ansible Tower, a June 2018 commissioned study conducted by Forrester Consulting on behalf of Red Hat."redhat.com/en/engage/total-economic-impact-ansible-tower-20180710

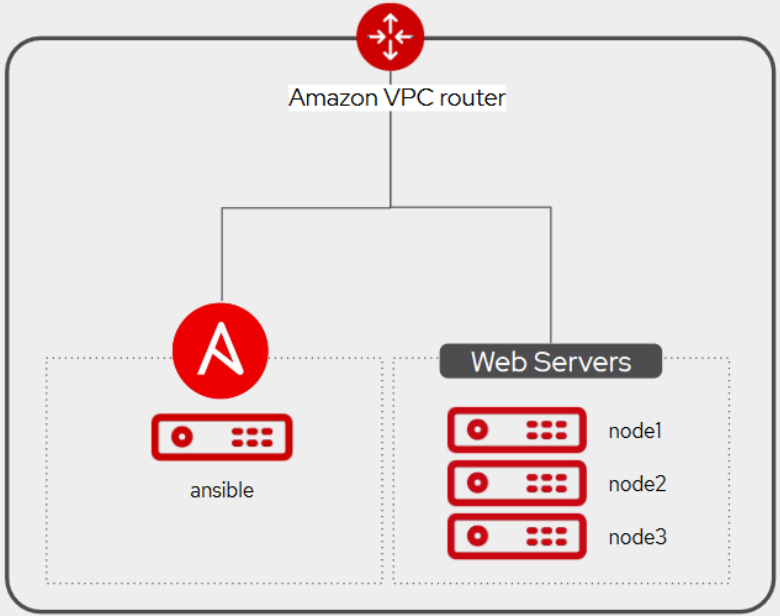
The lab environment today

- **Drink our own champagne**.
  Provisioned by, configured by, and managed by Red Hat Ansible Automation Platform.
  https://github.com/ansible/workshops

- **Learn with the real thing**
  Every student will have their own fully licensed Red Hat Ansible Tower control node. No emulators or simulators here.

- **Red Hat Enterprise Linux**
  All four nodes are enterprise Linux, showcasing real life use-cases to help spark ideas for what you can automate today.

Workbench Topology



Lab Location

http://3dc6.rhdemo.io/

Red Hat
Ansible Automation Platform

Exercise 1

Topics Covered:

- Understanding the Ansible Infrastructure
- Check the prerequisites

Exercise 2

Topics Covered:

- Ansible inventories
- Main Ansible config file
- Modules and ad-hoc commands
- Example: Bash vs. Ansible

Red Hat
Ansible Automation
Platform

Inventory

- Ansible works against multiple systems in an inventory
- Inventory is usually file based
- Can have multiple groups
- Can have variables for each group or even host

Understanding Inventory - Basic

```
[web]
node1 ansible_host=3.22.77.141
node2 ansible_host=3.15.193.71
node3 ansible_host=3.15.1.72

[control]
ansible ansible_host=18.217.162.148
```

Understanding Inventory - Variables

```
[all:vars]
ansible_user=student1
ansible_ssh_pass=ansible1234
ansible_port=22

[web]
node1 ansible_host=3.22.77.141
node2 ansible_host=3.15.193.71
node3 ansible_host=3.15.1.72

[control]
ansible ansible_host=18.217.162.148
```

Bash vs. Ansible

```bash
echo Running mssql-conf setup...
sudo
MSSQL_SA_PASSWORD=$MSSQL_SA_PASSWORD \
   MSSQL_PID=$MSSQL_PID \
   /opt/mssql/bin/mssql-conf -n setup accept-eula


echo 'export PATH="$PATH:/opt/mssql-tools/bin"' >>
~/.bash_profile
echo 'export PATH="$PATH:/opt/mssql-tools/bin"' >>
~/.bashrc
source ~/.bashrc
```

```yaml
- name: Run mssql-conf setup
  command: /opt/mssql/bin/mssql-conf -n setup
accept-eula
  environment:
   - MSSQL_SA_PASSWORD: "{{ MSSQL_SA_PASSWORD }}"
   - MSSQL_PID: "{{ MSSQL_PID }}"
  when: install is changed

- name: Add mssql-tools to $PATH
  lineinfile:
    path: "{{ item }}"
    line: export PATH="$PATH:/opt/mssql-tools/bin"
  loop:
    - ~/.bash_profile
    - ~/.bashrc
```

Red Hat
Ansible Automation Platform

Exercise 3

Topics Covered:

- Playbooks basics
- Running a playbook

An Ansible Playbook

A play

```yaml
---
- 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
```

An Ansible Playbook

A task

```yaml
---
- 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
```

An Ansible Playbook

module

```
---
- 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
```

Running an Ansible Playbook:

The most important colors of Ansible

```
A task executed as expected, no change was made.

A task executed as expected, making a change

A task failed to execute successfully
```

Running an Ansible Playbook

```
[user@ansible] $ ansible-playbook apache.yml

PLAY [webservers] ***************************************************************

TASK [Gathering Facts] **********************************************************
ok: [web2]
ok: [web1]
ok: [web3]

TASK [Ensure httpd package is present] ******************************************
changed: [web2]
changed: [web1]
changed: [web3]

TASK [Ensure latest index.html file is present] *********************************
changed: [web2]
changed: [web1]
changed: [web3]

TASK [Restart httpd] ************************************************************
changed: [web2]
changed: [web1]
changed: [web3]

PLAY RECAP ******************************************************************************
web2                       : ok=1    changed=3 unreachable=0   failed=0
web1                       : ok=1    changed=3 unreachable=0   failed=0
web3                       : ok=1    changed=3 unreachable=0   failed=0
```

Exercise 4

Topics Covered:

- Working with variables
- What are facts?

Red Hat
Ansible Automation Platform

An Ansible Playbook Variable Example

```yaml
---
- name: variable playbook test
  hosts: localhost

  vars:
    var_one: awesome
    var_two: ansible is
    var_three: "{{ var_two }} {{ var_one }}"

  tasks:

    - name: print out var_three
      debug:
        msg: "{{var_three}}"
```

An Ansible Playbook Variable Example

```yaml
---
- name: variable playbook test
  hosts: localhost

  vars:
    var_one: awesome
    var_two: ansible is
    var_three: "{{ var_two }} {{ var_one }}"

  tasks:

    - name: print out var_three
      debug:
        msg: "{{var_three}}"
```

Facts

- Structured data in the form of Ansible variables
- Information is capture from the host
- Ad-hoc command **setup** will show facts

```
    "ansible_facts": {
        "ansible_default_ipv4": {
            "address": "10.41.17.37",
            "macaddress": "00:69:08:3b:a9:16",
            "interface": "eth0",
...
```

Ansible Variables and Facts

```yaml
---
- name: Output facts within a playbook
  hosts: all

  tasks:

    - name: Prints Ansible facts
      debug:
        msg: "The default IPv4 address of {{ ansible_fqdn }}
              is {{ ansible_default_ipv4.address }}"
```

```
TASK [Prints Ansible facts] ***********************************************
ok: [node3] =>
  msg: The default IPv4 address of node3 is 172.16.63.104
ok: [node1] =>
  msg: The default IPv4 address of node1 is 172.16.178.80
ok: [node2] =>
  msg: The default IPv4 address of node2 is 172.16.166.120
ok: [ansible] =>
  msg: The default IPv4 address of student1.sean-may4.rhdemo.io is 172.16.86.242
```

Ansible Inventory – Managing Variables In Files

```
$ tree ansible-files/

├── deploy_index_html.yml
├── files
│   ├── dev_web.html
│   └── prod_web.html
├── group_vars
│   └── web.yml
└── host_vars
    └── node2.yml
```

Ansible Inventory - Managing Variables In Files

```
├──
deploy_index_html.yml
├── files
│   ├── dev_web.html
│   └── prod_web.html
├── group_vars
│   └── web.yml
└── host_vars
    └── node2.yml
```

```
$ cat group_vars/web.yml
---
stage: dev
```

```
$ cat host_vars/node2.yml
---
stage: prod
```

```yml
- name: copy web.html
  copy:
    src: "{{ stage }}_web.html"
    dest: /var/www/html/index.html
```

Exercise 5

Topics Covered:

- Surveys

Red Hat
Ansible Automation Platform

Surveys

Tower 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 Tower survey is a simple question-and-answer form that allows users to customize their job runs. Combine that with Tower'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 **Add Survey Button** will appear

ADD SURVEY

Click the button to open the Add Survey window.

TOWER

admin 0

VIEWS
- Dashboard
- Jobs
- Schedules
- My View

RESOURCES
- Templates
- Credentials
- Projects
- Inventories
- Inventory Scripts

ACCESS

TEMPLATES / Create index.html

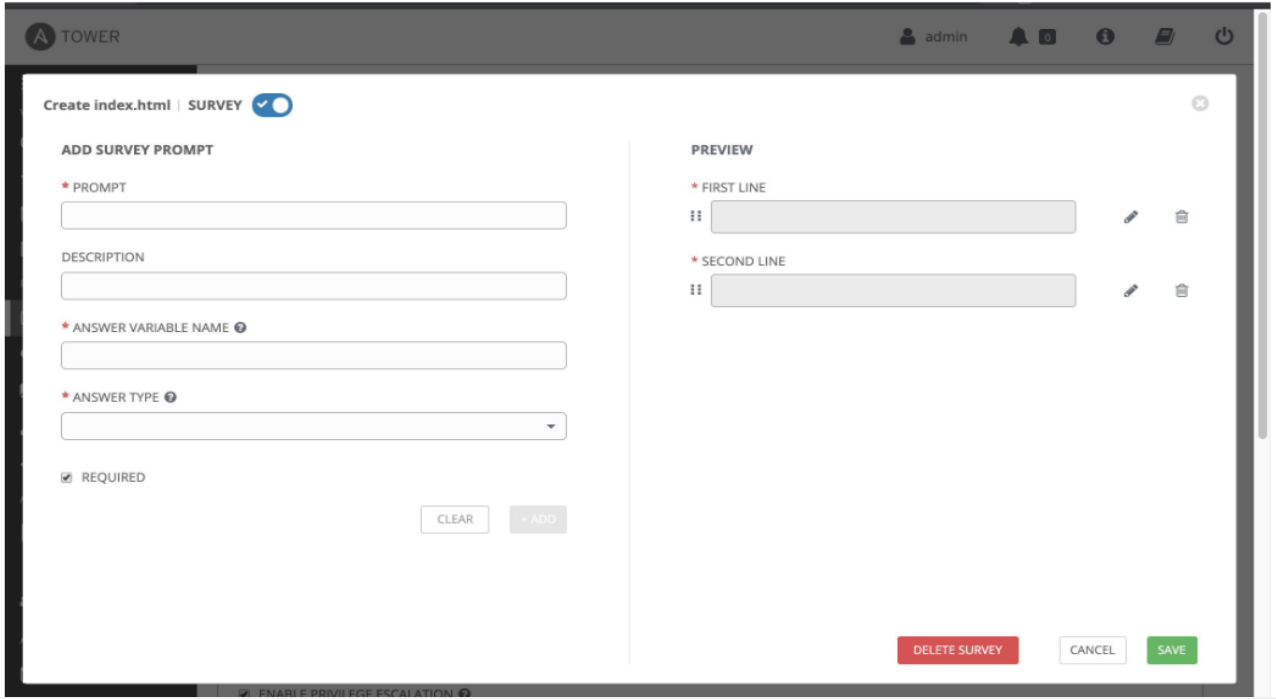
Create index.html

DETAILS | PERMISSIONS | NOTIFICATIONS | COMPLETED JOBS | SCHEDULES | EDIT SURVEY

* NAME: Create index.html	DESCRIPTION:	* JOB TYPE (PROMPT ON LAUNCH)
* INVENTORY (PROMPT ON LAUNCH): Workshop Inventory	* PROJECT: Workshop Project	* PLAYBOOK: rhel/apache/apache_role_inst...
CREDENTIALS (PROMPT ON LAUNCH): Workshop Credential	FORKS: 0	LIMIT (PROMPT ON LAUNCH): web
* VERBOSITY (PROMPT ON LAUNCH): 0 (Normal)	JOB TAGS (PROMPT ON LAUNCH):	SKIP TAGS (PROMPT ON LAUNCH):

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.

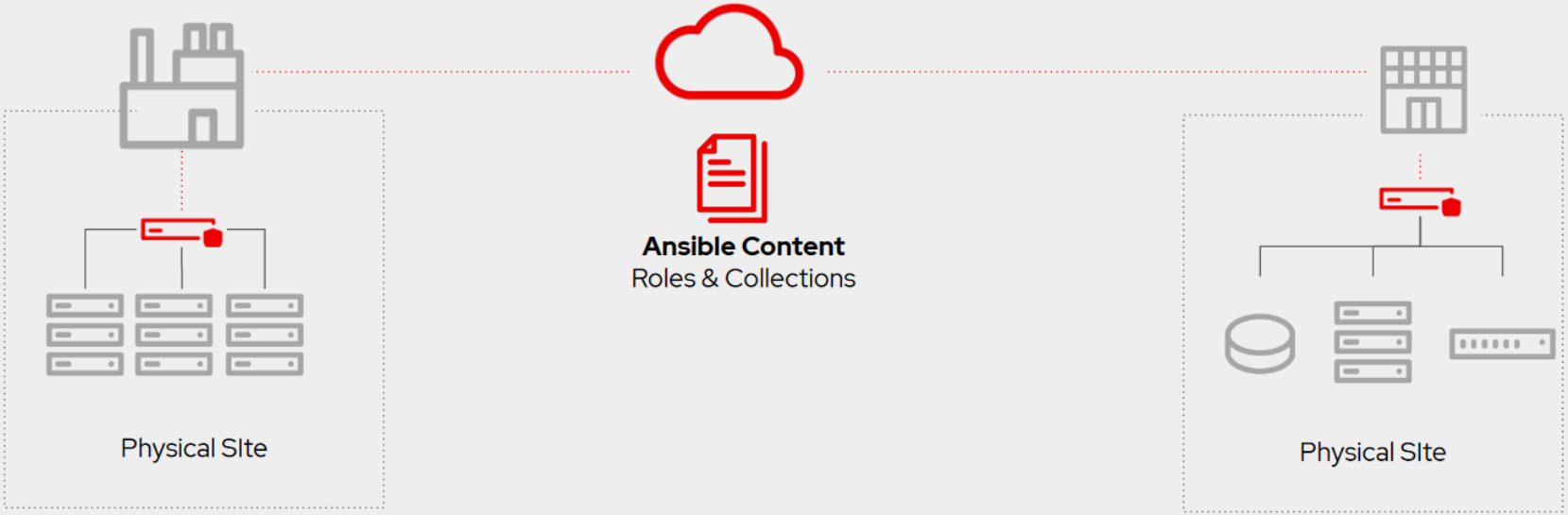
Red Hat
Ansible Automation Platform

Exercise 6

Topics Covered:

- Red Hat Enterprise Linux System Roles

Automation Hub and Ansible Galaxy



An Ansible Playbook Variable Example

```yaml
---
- name: example system roles playbook
  hosts: web

  tasks:

    - name: Configure Firewall
      include_role:
        name: linux-system-roles.firewall

    - name: Configure Timesync
      include_role:
        name: linux-system-roles.timesync
```

Red Hat
Ansible Automation Platform

Exercise 7

Topics Covered:

- Red Hat Insights intro
- Insights integration

Red Hat Insights

Included with your Red Hat Enterprise Linux subscription

Assesses

customer's Red Hat environments

Remediates

findings with prescriptive remediation steps or an Ansible playbook

Insights

rule contributions directly from Red Hat subject matter experts

Identifying risks for Availability, performance, stability and security

Red Hat

John Spinks

Red Hat Insights

Overview

Rules

Inventory

Remediations

Documentation

Remediations > May2019_Critical_Fixes

May2019_Critical_Fixes

Download Playbook

Delete

Systems reboot

6	0	
No reboot	Reboot required	Auto reboot

Playbook details

Created by: John Spinks

Created: a minute ago

Last modified by: John Spinks

Insights plans with Ansible playbooks

Solve common issues through Ansible Automation

Actions	Resolution	Reboot required	Systems	Type
Dnsmasq with listening processes vulnerable to remote code execution via crafted DNS requests (CVE-2017-14491)	Update dnsmasq package and restart related service(s)		6	Insights

Systems

ic3.example.com

ic4.example.com

ic6.example.com

ic7.example.com

ANSIBLE & INSIGHTS

While Insights includes Ansible playbooks for risks, Insights alone can't perform remediation of the risks.

Insights

- Insights provides Ansible Playbooks for resolving many common risks.
- Dynamically generates Ansible Playbooks for risk remediation
- Playbooks can be downloaded and run via `ansible-playbook` or Satellite

Insights connected to Ansible Tower

- View identified risks in the Tower inventory
- Execute generated Ansible Playbook as a Tower job
- Use Tower for enterprise risk remediation

Next Steps

Get Started

ansible.com/get-started
ansible.com/tower-trial

Join the Community

ansible.com/community

Workshops and Training

ansible.com/workshops
Red Hat Training

Share Your Story

Follow Us@Ansible
Friend Us On Facebook

THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHatNews

youtube.com/user/RedHatVideos

Red Hat