Red Hat Ansible Automation

Red Hat Ansible Engine

Red Hat Ansible Network Automation

Red Hat Ansible Tower

Exploring Ansible

Patrick Ladd
Technical Account Manager
pladd@redhat.com

Red Hat

Why Ansible?

Simple

Be productive Quickly

- Human Readable
- No Coding Skills
- In Order Execution

Powerful

Orchestrate Entire Lifecycle

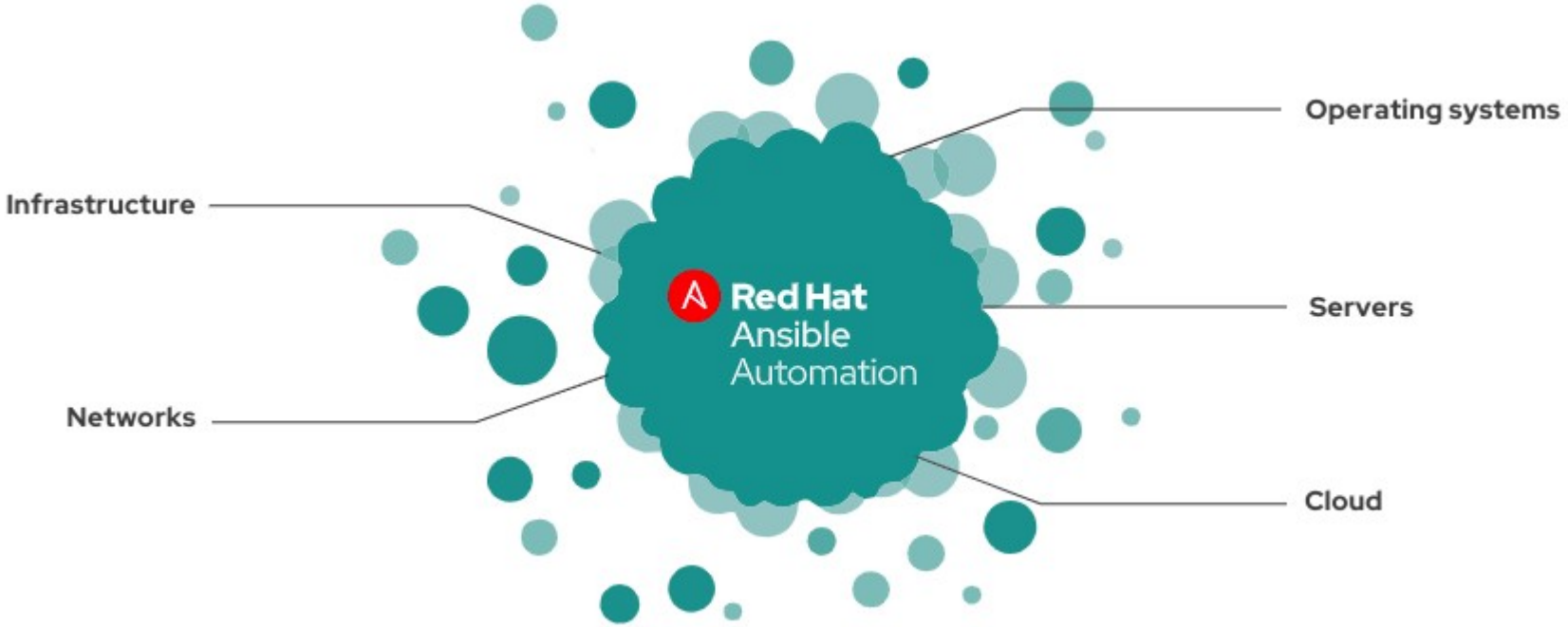
- Application Deployment
- Configuration Management
- Workflow Orchestration

Secure

More Efficient & More Secure

- Agentless
- Uses SSH & WinRM
- Less to Update or Exploit

Use across all your environments



Red Hat

Use across all your environments

- Operating systems: Red Hat Enterprise Linux®, Windows, Ubuntu, and more.
- Servers: HP, Dell, Cisco, and more.
- Cloud: Amazon Web Services, Microsoft Azure, Google Cloud Platform, DigitalOcean, CloudStack, OpenStack®, Rackspace, Packet, and more.
- Infrastructure: Red Hat OpenShift®, VMware, NetApp, Kubernetes, Jenkins, JBoss®, Docker, and more.
- Networks: Arista, Cisco, Juniper, F5, Palo Alto, and more.

- **2,832** Core Modules
- **20,916** Community Roles

Use Throughout Your Workflow

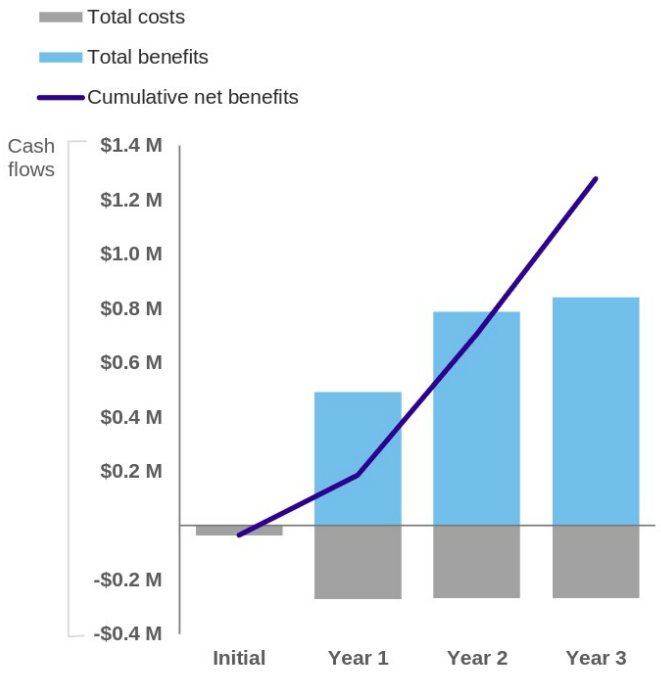
- Infrastructure Deployment.
- Code deployment.
- Build automation.
- Artifact management.
- Certificate management.
- Service restarts.
- Decommissioning resources.

IDC Case Study:
Save Time – Operate Efficiently – Respond Faster

- IDC Impact Study
  - Managed service provider
  - 5 Datacenters
  - 1000+ employees
  - Manages 1500+ systems

Source:
https://www.redhat.com/en/resources/total-economic-impact-ansible-spotlight

IDC Case Study:
146% ROI – 3 month payback



Lead time reduced 66%
$1,321,364 savings over 3 years

Avoided appliance purchase
$389,707 cost avoidance

Automated reconfiguration
Reduced man hours by 94%

Automated security updates
Reduced man hours by 80%

Source:
https://www.redhat.com/en/resources/total-economic-impact-ansible-spotlight

Red Hat

Ansible and Puppet

- Agent-less
- Push vs Pull
- YAML vs JSON
- Top to bottom ordering*

Source:
https://www.redhat.com/en/resources/total-economic-impact-ansible-spotlight

Ansible Basics

YAML – Yet Another Markup Language

- Simple, Human Readable
- Key/Value based
- Lists and Dictionaries

- Mandatory Indentation – 2 spaces
- File start ----
- File end . . .
- Quoting: “” and ‘’

```
# Me
- me:
    name: Patrick Ladd
    employed: yes
    job: Technical Account Manager
    age: 49
    interests: |
      backpacking
      lego
    friends:
      - Joe
      - Bob
      - Ralph
```

YAML – Lists & Dictionaries

- Lists
  - Indented
  - '–' delimiter
  - Short form: `[ a, b ]`

```
# List of favorite foods
foods:
  - bacon
  - pizza
  - steak

drinks: [ 'soda', 'water', 'cider' ]
```

- Dictionaries
  - Indented
  - Key: Value (space after ':' reqired!)
  - Short form: `{ k1: v1, k2: v2 }`

```
# Dictionaries of book and movie
book:
  name: Ansible 101
  author: Foo Bar
  brief: Getting started with Ansible

Movie: { name: "The Matrix", genre: "SciFi" }
```

YAML – Booleans and Special Characters

- Boolean values

```
create_key: yes
needs_root: no
likes_vi: True
dislikes_emacs: TRUE
uses_cvs: false
```

- When you don't want the boolean value

```
non_boolean: "yes"
other_string: "False"
```

- Watch out for special characters!

```
info: somebody said I should put a colon
here: so I did

info: "somebody said I should put a colon
here: so I did"
```

YAML – New Lines

- Spanning Lines with newlines

```
include_newlines: |
            exactly as you see
            will appear these three
            lines of poetry
```

- Spanning Lines with no newlines

```
ignore_newlines: >
            this is really a
            single line of text
            despite appearances
```

YAML – Getting Fancy

- Combine lists and dictionaries in any combination
- Escaping
- Line folding / extending

- Handy References:
  - https://yaml.org/spec/1.2/spec.html
  - https://docs.ansible.com/ansible/latest/reference_appendices/YAMLSyntax.html
  - https://yaml.org/refcard.html

```
# People
- foo:
    name: Foo Bar
    job: Awesome Administrator
    languages:
      - perl
      - python
    fun fact: |
      can see in the dark
- learn:
    name: Learn Moar
    job: Sr Awesome Administrator
    languages:
      - ansible
      - ruby
      - awk
    fun fact: >
      runs with
      scissors
```

YAML – Check your work

- From the command line:

```
python -c 'import yaml, sys; yaml.safe_load(sys.stdin)' < my.yaml
```

- YamlLint: https://pypi.org/project/yamllint/
- Official checker for Ansible playbooks: **`ansible-playbook --syntax-check`**

- VIM trick:
  - Add this to **`$HOME/.vimrc`**

```
autocmd FileType yaml setlocal ai ts=2 sw=2 et
```

Basic Architecture

- Control Node
- Managed Hosts
- Connection

Inventory - Where to run?

- Defines all hosts that Ansible manages
- Static
  - Provides basic lists and groupings for hosts
- Dynamic inventories
  - Satellite
  - Cloud Providers (OpenStack, AWS)
  - LDAP
  - CMDB
  - http://docs.ansible.com/ansible/intro_dynamic_inventory.html
  - Many others (there are plugins)
  - List them: **`ansible-doc -t inventory -l`**

Ansible Inventories – Static INI

- INI-like text file
- Sectioned to define groups of hosts
- Groups of groups

```
[rhel]
rhel1.mynet.com
192.168.122.156
192.168.122.168

[fedora]
fedora.mynet.com

[windows]
win1.mynet.com

[linux:children]
rhel
fedora
```

Ansible Inventories – Static INI

- Common Options

```
[master]
localhost ansible_connection=local

[rhel]
rhel1.mynet.com:1234
ansible_connection=ssh ansible_user=foo
192.168.122.156
192.168.122.168
```

- Defining Ranges [start:end]

```
[moar-servers]
web[1:7].moar.com
db[01:07].moar.com
192.168.[4:7].[0:255]
```

Authentication – Who are you?

- ssh used for most connections
- Best practice – use ssh keys
- Best practice – run with least privilege
  - Use **`ansible_become`** if root is needed

Modules – What to do?

- What allows us to perform actions on a host. They do the heavy lifting
- Thousands of built in modules
- The basics:
  - setup – reports facts
  - ping – checks connection
  - command – run a command (Don't use this until you look for appropriate modules!)
- Use ansible-doc to search for additional information such as details of a module

```
$ ansible-doc copy
> COPY    (/usr/lib/python2.7/site-packages/ansible/modules/files/copy.py)

        The 'copy' module copies a file from the local or remote
        machine to a location on the remote machine…
```

Ad-Hoc Commands

- For quick one-offs and testing
- **`ansible -m module host`**
- For example:
  - **`ansible -m setup thathostoverthere`**

Demo – Hosts & Ad-hoc commands

- Host/IP
- Group
- List
- Patterns

Playbooks

- Written in YAML composed of one or more “plays”
- Each play contains a list of tasks
- Each task is something to check, modify or run
- Order and spaces matter!

Playbooks - components

- Pieces of a playbook – a playbook can contain 1 or more plays
- Name attribute: “name: a descriptive label of the play”
- Hosts attribute: “hosts: pattern.hosts.com”
  - As described in the Referencing Hosts section
- User attributes: i.e. if the default remote_user will not work we provide a suitable one here
- Privilege escalation attributes: if necessary defining become, become_method, become_user, etc
- Tasks attribute: →

Playbooks - Tasks

- Tasks Attribute

  A list of dictionaries with key/value pairs

```
tasks:
- name: first task
  service: name=httpd enabled=true
```

- One or multiple tasks

```
tasks:
- name: first task
  service: name=httpd enabled=true
- name: second task
  service: name=sshd enabled=true
- name: third task
  service: name=bluetooth enabled=false
```

Playbooks - Tasks

- This:

```
tasks:
- name: first task
  service: name=httpd enabled=true
```

- Is equivalent to this:

  (but this is cleaner)

```
tasks:
- name: first task
  service:
    name: httpd
    enabled: true
```

Demo - Playbooks

- Checking for syntax
- Doing a dry run
- Step-by-step run
- Run a playbook

Getting Fancy

Variables

- Variable names must start with a letter and only contain letters, numbers and underscores
- Defined in three scopes
  - Global: Variables set via the command line or Ansible configuration file
  - Play: Defined in the play
  - Host: Defined on host groups and/or individual hosts by the inventory, fact gathering or task
- Variables with the same name- Higher level wins
  - CLI > Playbook > Inventory

Variables in Playbooks

- At the start of a play in vars block

```
- hosts: all
  vars:
    user: joe
    home: /home/joe
```

- In a vars file in YAML format

```
- hosts: all
  vars_files:
    - vars/users.yml
```

- Called with **{{ var }}** syntax
  - must be in quotes if it starts the line

```
tasks:
  - name: Creates the user {{ user }}
    user:
      name: "{{ user }}"
```

Variables on Hosts and Groups

- Individually from inventory

```
[webservers]
localhost  ansible_connection=local
web1.foo.com
web2.foo.com:1234 ansible_connection=ssh
ansible_user=foo
```

- Defined for all hosts in a group

```
[eng]
sys1.foo.com
sys2.foo.com

[prod]
prod1.foo.com
prod2.foo.com

[eng:vars]
user=foo
```

Variables – CLI

- Override from Command Line

```
$ ansible-playbook user.yml -e
"user=mike"
```

- Referencing Arrays

  (Dot notation can be a problem with modules)

```
users:
  joe:
    first_name: Joe
    last_name: Jones
    home_dir: /users/joe
  mike:
    first_name: Mike
    last_name: Cook
    home_dir: /users/mike

# Returns 'Joe'
users.joe.first_name
# Returns 'Joe'
users['joe']['first_name']
```

Demo - Playbooks

- Simple variable usage and substitution
- Register to capture command output
- Debug to dump a value of a registered variable

Variables - Facts

- Facts: Variables that are automatically discovered
- Pulled by the setup modules
- List all the facts for a system (be warned this is a ton of data)

```
$ ansible host –m setup
```

- Can be stored into variables for reuse or used directly
- Custom facts can be created
  - Saved in /etc/ansible/facts.d
  - Must have the .fact extension – myfacts.fact
- Gathering facts can be filtered

```
$ ansible myhost1 –m setup –a “filter=ansible_user_id”
```

Flow Control - Conditionals

- Use conditionals to execute tasks or plays when conditions are met
- Use the when statement to evaluate prior to executing
- When must be placed outside of the module
- Does not have to be at the top of the list

```
- name: Create the DB admin
  user:
    name: db_admin
  when: inventory_hostname in groups[“databases”]
```

Flow Control – Multiple Conditions

```
ansible_kernel == 3.10.0-862.el7.x86_64 and inventory_hostname in
groups['engineering']

ansible_distribution == "RedHat" or ansible_distribution == "Fedora"

(ansible_distribution == "RedHat" and ansible_distribution_major_version == 7) or
(ansible_distribution == "Fedora" and ansible_distribution_major_version == 27)

when:
- ansible_distribution == "CentOS"
- ansible_distribution_major_version == "6"
```

Flow Control - Loops

- Simple loop:

  a list of items that is iterated over

  provided by loop

  (this deprecates **`with_items`**)

```
yum:
 name: "{{ item }}"
 state: latest
loop:
  - postfix
  - dovecot

================================

vars:
  mail_services:
    - postfix
    - dovecot
…
yum:
  name: "{{ item }}"
  state: latest
loop:
  - "{{ mail_services }}"
```

Flow Control – Loops with Conditionals

- Combine when and loop
- NOTE: The when statement is processed for each item

```
-name: install mariadb-server if enough space in root
 yum:
   name: mariadb-server
   state: latest
 loop: “{{ ansible_mounts }}”
 when: item.mount == “/” and item.size_available > 300000000
```

Error Handling

- By default if a task fails a play is aborted
- Ignore a failed task with ignore_errors

```
yum:
  name: notapkg
  state: latest
ignore_errors: yes
```

Error Handling - Overrides

- Override the failed state

```
shell:
  cmd: /usr/local/bin/create_users.sh
register: command_result
failed_when: "'Password missing' in
command_result.stdout"
```

- Override the changed state

```
shell:
  cmd: /usr/local/bin/upgrade-database
  register: command_result
  changed_when: "'Success' in
command_result.stdout"
```

Error Handling – Debug Module

- Can provide the value for a playbook variable
- Provide some context with msg
  - **- debug: msg="The free memory for this system is {{ ansible_memfree_mb }}"**
- Set verbosity (Ansible 2.1 and forward)
  - **- debug: var=output verbosity=2**
  - Will only show when -vv or above is set on run

    $ ansible-playbook myplay.yml -vv
- Additional examples - http://docs.ansible.com/ansible/debug_module.html

Logging

- By default Ansible does not log to a file
  - Set **log_path** in the default section of ansible.cfg or set the **$ANSIBLE_LOG_PATH** environment variable
- Watch permissions
  - If you want to log to /var/log you may need to run as root or modify the directory permissions
- Configure logrotate to help manage growing logs
- Beware leaked secrets in log files!

Getting Fancier

Batching

- Default is parallel execution

```
- name: test play
  hosts: webservers
  serial: 3
```

- Setting A batch size:

```
- name: test play
  hosts: webservers
  serial: 30%
```

- Rolling batch size:

```
- name: test play
  hosts: webservers
  serial:
  - 1
  - 5
  - 10
  - “20%”
```

Batching – Failure Percentage

- By default we run on all hosts until completion/failure

- Setting a Maximum failure percentage
  - If we hit a 30% failure rate in a group we stop

```
- name: test play
  hosts: webservers
  max_fail_percentage: 30
  serial: 10
```

Batching – Free Run

- Use the free strategy to run hosts out of lock-step

```
- hosts: eng
  strategy: free
  tasks:
    ...
```

Secrets Management

- What is the Vault?
  - Keep sensitive data encrypted
- What can the Vault store?
  - Most commonly structured data (i.e. YAML files)
- Now lots of vault plugins available to integrate other secrets management

AWX / Ansible Tower

Tower / AWX

- Use it for these environments
  - Large
  - Multi-user
  - Enterprise
  - Complex

- Features
  - RBAC
  - Scheduling
  - GUI
  - Integrations

Demo

- Connecting repositories
- Adding credentials
- Job templates
- Running Jobs

Tower / AWX

- Recording Available
  - https://www.redhat.com/en/events/webinar/ansible-and-not-so-leaning-tower-automation-integration-orchestration-cross-platform-environments

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.

linkedin.com/company/red-hat

youtube.com/user/RedHatVideos

facebook.com/redhatinc

twitter.com/RedHat

Red Hat