Getting Started with Ansible
What is the Red Hat Ansible Automation Platform?

The Ansible project is an open source community sponsored by Red Hat. It’s also a simple automation language that perfectly describes IT application environments in Ansible Playbooks.

Ansible Engine is a supported product built from the Ansible community project.

Ansible Tower is an enterprise framework for controlling, securing, managing and extending your Ansible automation (community or engine) with a UI and RESTful API.
Why Ansible?

**Simple**
- Human readable automation
- No special coding skills needed
- Tasks executed in order
- Usable by every team
- Get productive quickly

**Powerful**
- App deployment
- Configuration management
- Workflow orchestration
- Network automation
- Orchestrate the app lifecycle

**Agentless**
- Agentless architecture
- Uses OpenSSH & WinRM
- No agents to exploit or update
- Get started immediately
- More efficient & more secure
With Ansible you can automate:

**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.
PLAYBOOKS ARE WRITTEN IN YAML
Tasks are executed sequentially
Invoke Ansible modules
- name: latest index.html file is present
  template:
    src: files/index.html
    dest: /var/www/html/
PLUGINS ARE “GEARS IN THE ENGINE”
Code that plugs into the core engine
Adaptability for various uses & platforms

{{ some_variable | to_nice_yaml }}
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

- **[firewalls]**
  - checkpoint01.internal.com

- **[lb]**
  - f5-01.internal.com
CLOUD
Red Hat Openstack, Red Hat Satellite, VMware, AWS EC2, Rackspace, Google Compute Engine, Azure
AUTOMATE EVERYTHING
Red Hat Enterprise Linux, Cisco routers, Arista switches, Juniper routers, Windows hosts, Checkpoint firewalls, NetApp storage, F5 load balancers and more
Using Ansible
Ad-hoc commands

# check all my inventory hosts are ready to be
# managed by Ansible
$ ansible all -m ping

# run the uptime command on all hosts in the
# web group
$ ansible web -m command -a "uptime"

# collect and display the discovered for the
# localhost
$ ansible localhost -m setup
Ad-hoc example
An inventory is a file containing:

- Hosts
- Groups
- Inventory-specific data (variables)
- Static or dynamic sources
Ansible Playbooks
- 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=started
- 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=started
- 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=started
- 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=.started
- 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=started
- 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=started
tasks:
  - name: add cache dir
    file:
      path: /opt/cache
      state: directory

  - name: install nginx
    yum:
      name: nginx
      state: latest
      notify: restart nginx

handlers:
  - name: restart nginx
    service:
      name: nginx
      state: restarted
Variables

Ansible can work with metadata from various sources and manage their context in the form of variables.

- Command line parameters
- Plays and tasks
- Files
- Inventory
- Discovered facts
- Roles
Tips/Best Practices
Simplicity
Simplicity

- hosts: web
tasks:
  - yum:
      name: httpd
      state: latest

- service:
  name: httpd
  state: started
  enabled: yes
- hosts: web
  name: install and start apache
  tasks:
    - name: install apache packages
      yum:
        name: httpd
        state: latest
    - name: start apache service
      service:
        name: httpd
        state: started
        enabled: yes
Naming example
Inventory

10.1.2.75
10.1.5.45
10.1.4.5
10.1.0.40

w14301.example.com
w17802.example.com
w19203.example.com
w19304.example.com
Inventory

- db1 ansible_host=10.1.2.75
- db2 ansible_host=10.1.5.45
- db3 ansible_host=10.1.4.5
- db4 ansible_host=10.1.0.40
- web1 ansible_host=w14301.example.com
- web2 ansible_host=w17802.example.com
- web3 ansible_host=w19203.example.com
- web4 ansible_host=w19203.example.com
Dynamic Inventories

- Stay in sync automatically
- Reduce human error
YAML Syntax
YAML and Syntax

- name: install telegraf
  yum: name=telegraf-{{ telegraf_version }} state=present update_cache=yes disable_gpg_check=yes enablerepo=telegraf
  notify: restart telegraf

- name: configure telegraf
  template: src=telegraf.conf.j2 dest=/etc/telegraf/telegraf.conf

- name: start telegraf
  service: name=telegraf state=started enabled=yes
- name: install telegraf
  yum: >
    name=telegraf-{{ telegraf_version }}
    state=present
    update_cache=yes
    disable_gpg_check=yes
    enablerepo=telegraf
  notify: restart telegraf

- name: configure telegraf
  template: src=telegraf.conf.j2 dest=/etc/telegraf/telegraf.conf

- name: start telegraf
  service: name=telegraf state=started enabled=yes
- name: install telegraf
  yum:
    name: telegraf-{{ telegraf_version }}
    state: present
    update_cache: yes
    disable_gpg_check: yes
    enablerepo: telegraf
  notify: restart telegraf

- name: configure telegraf
  template:
    src: telegraf.conf.j2
    dest: /etc/telegraf/telegraf.conf
  notify: restart telegraf

- name: start telegraf
  service:
    name: telegraf
    state: started
    enabled: yes
ansible-playbook playbook.yml --syntax-check
Roles
Roles

• Think about the full life-cycle of a service, microservice or container – not a whole stack or environment
• Keep provisioning separate from configuration and app deployment
• Roles are not classes or object or libraries – those are programming constructs
• Keep roles loosely-coupled – limit hard dependencies on other roles or external variables
Variable
Precedence
The order in which the same variable from different sources will override each other.
Variable Precedence

1. Extra vars
2. Include params
3. Role (and include_role) params
4. Set_facts / registered vars
5. Include_vars
6. Task vars (only for the task)
7. Block vars (only for tasks in the block)
8. Role vars
9. Play vars_files
10. Play vars_prompt
11. Play vars
12. Host facts / Cached set_facts
13. Playbook host_vars
14. Inventory host_vars
15. Inventory file/script host vars
16. Playbook group_vars
17. Inventory group_vars
18. Playbook group_vars/all
19. Inventory group_vars/all
20. Inventory file or script group vars
21. Role defaults
22. Command line values (e.g., -u user)
Things to Avoid
Things to Avoid

- Using command modules
  - Things like shell, raw, command etc.
- Complex tasks...at first
  - Start small
- Not using source control
  - But no really...
Ansible Content Collections
Collections Q and A

What are they?
- Collections are a distribution format for Ansible content that can include playbooks, roles, modules, and plugins. You can install and use collections through Ansible Galaxy and Automation Hub.

How do I get them?
- ansible-galaxy collection install namespace.collection -p /path

Where can I get them?
- Today
  - Galaxy
  - Automation Hub
Collection Directory Structure

- **docs/**: local documentation for the collection
- **galaxy.yml**: source data for the MANIFEST.json that will be part of the collection package
- **playbooks/**: playbook snippets
  - **tasks/**: holds 'task list files' for include_tasks/import_tasks usage
- **plugins/**: all ansible plugins and modules go here, each in its own subdir
  - **modules/**: ansible modules
  - **lookups/**: lookup plugins
  - **filters/**: Jinja2 filter plugins
  - **connection/**: connection plugins required if not using default
- **roles/**: directory for ansible roles
- **tests/**: tests for the collection's content
Collections: Let’s Go!

1. Init collection: `ansible-galaxy collection init foo.bar`
2. Sanity testing: `ansible-test sanity`
3. Unit tests: `ansible-test units`
4. Integration tests: `ansible-test integration`
5. Build the collection: `ansible-galaxy collection build`
6. Publish the collection: `ansible-galaxy collection publish`
7. Install the collection: `ansible-galaxy collection install foo.bar`
Thank you

Questions?

linkedin.com/company/red-hat
youtube.com/user/RedHatVideos
facebook.com/redhatinc
twitter.com/RedHat