Ansible Hands-on Introduction

Jon Jozwiak, Sr. Cloud Solutions Architect

Minneapolis RHUG - April 13, 2017
What is Ansible?

It's a **simple automation language** that can perfectly describe an IT application infrastructure in Ansible Playbooks.

It's an **automation engine** that runs Ansible Playbooks.

Ansible Tower is an **enterprise framework** for controlling, securing and managing your Ansible automation with a **UI and RESTful API**.
Ansible Is...

**SIMPLE**
- Human readable automation
- No special coding skills needed
- Tasks executed in order
- Get productive quickly

**POWERFUL**
- App deployment
- Configuration management
- Workflow orchestration
- Orchestrate the app lifecycle

**AGENTLESS**
- Agentless architecture
- Uses OpenSSH & WinRM
- No agents to exploit or update
- More efficient & more secure
Community

THE MOST POPULAR OPEN-SOURCE AUTOMATION COMMUNITY ON GITHUB

- 13,000+ stars & 4,000+ forks on GitHub
- 2000+ GitHub Contributors
- Over 900 modules shipped with Ansible
- New contributors added every day
- 1200+ users on IRC channel
- Top 10 open source projects in 2014
- World-wide meetups taking place every week
- Ansible Galaxy: over 18,000 subscribers
- 250,000+ downloads a month
- AnsibleFests in NYC, SF, London

http://ansible.com/community
Installing Ansible

# Install with yum (Example RHEL 7)
$ wget http://dl.fedoraproject.org/pub/epel/7/x86_64/e/epel-release-7-9.noarch.rpm
$ yum localinstall epel-release-7-9.noarch.rpm
$ yum --enablerepo=epel install ansible

# Install using pip
$ pip install ansible
How Ansible Works

ANSIBLE’S AUTOMATION ENGINE

- INVENTORY
- MODULES
- PLUGINS

PUBLIC / PRIVATE CLOUD
CMDB

USERS
ANSIBLE PLAYBOOK

HOSTS
NETWORKING
Modules

Modules are bits of code transferred to the target system and executed to satisfy the task declaration. Ansible ships with several hundred today!

- apt/yum
- copy
- file
- get_url
- git
- ping
- debug
- service
- synchronize
- template
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**

- Controls services on remote hosts. Supported OS systems include RHEL 6/7, OpenRC, Upstart, Solaris SMF, systemd, sysvstart.

**Options**

<table>
<thead>
<tr>
<th>parameter</th>
<th>required</th>
<th>default</th>
<th>choices</th>
<th>comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>arguments</td>
<td>no</td>
<td></td>
<td></td>
<td>Additional arguments provided as comma-delimited list after command.</td>
</tr>
<tr>
<td>enabled</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>Whether the service should start or stop. At least one of state and enabled are required.</td>
</tr>
<tr>
<td>name</td>
<td></td>
<td></td>
<td></td>
<td>None of the service.</td>
</tr>
<tr>
<td>pattern</td>
<td>no</td>
<td></td>
<td></td>
<td>If the service does not respond to the status command, none a subtring to look for in the output of the process are passed in for a status result. If the string is found, the service will be assumed to be running.</td>
</tr>
<tr>
<td>created</td>
<td>no</td>
<td>default</td>
<td></td>
<td>For OpenRC, nscd and daemons that do not already have a PID or name.</td>
</tr>
<tr>
<td>started</td>
<td>no</td>
<td></td>
<td></td>
<td>If the service is being started, then sleep this many seconds between the stop and start command. This helps to synchronize badly behaving init daemons that kill immediately after attempting to stop.</td>
</tr>
<tr>
<td>state</td>
<td>no</td>
<td></td>
<td></td>
<td>&quot;started&quot;, &quot;stopped&quot;, &quot;restarted&quot;, or &quot;reloaded&quot; are deprecated actions that will not run commands unless necessary. &quot;running&quot; will always invoke the service. &quot;nscd&quot; will always reload. At least one of state and enabled are required.</td>
</tr>
<tr>
<td>user</td>
<td>no</td>
<td></td>
<td></td>
<td>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 &quot;service_module&quot; fact and falls back to the old name when none matching is found.</td>
</tr>
</tbody>
</table>

http://docs.ansible.com/
# 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 local 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.
Inventory

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

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

10.42.0.2
10.42.0.6
10.42.0.7
10.42.0.8
10.42.0.100
host.example.com
[control]
control ansible_host=10.42.0.2

[web]
node-[1:3] ansible_host=10.42.0.[6:8]

[haproxy]
haproxy ansible_host=10.42.0.100

[all:vars]
ansible_user=vagrant
ansible_ssh_private_key_file=~/.vagrant.d/insecure_private_key
Ad-Hoc Commands

An ad-hoc command is a single Ansible task to perform quickly, but don’t want to save for later.

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

# collect and display the discovered facts for the 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 systems that are stored as variables for later use in a play.

```
$ ansible localhost -m setup
localhost | success >> {
  "ansible_facts": {
    "ansible_default_ipv4": {
      "address": "192.168.1.37",
      "alias": "wlan0",
      "gateway": "192.168.1.1",
      "interface": "wlan0",
      "macaddress": "c4:85:08:3b:a9:16",
      "mtu": 1500,
      "netmask": "255.255.255.0",
      "network": "192.168.1.0",
      "type": "ether"
    }
  }
}
```
Lab # 1: Ad-Hoc Commands
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
Tasks

Tasks are the application of a module to perform a specific unit of work.

- **file**: A directory should exist
- **yum**: A package should be installed
- **service**: A service should be running
- **template**: Render a configuration file from a template
- **get_url**: Fetch an archive file from a URL
- **git**: Clone a source code repository
Example Tasks in a Play

tasks:
- name: httpd package is present
  yum:
    name: httpd
    state: latest

- name: latest index.html file is present
  copy:
    src: files/index.html
    dest: /var/www/html/

- name: restart httpd
  service:
    name: httpd
    state: restarted
Handler Tasks

Handlers are special tasks that run at the end of a play if notified by another task when a change occurs.

If a configuration file gets changed notify a service restart task that it needs to run.
Example Handler Task in a Play

tasks:
- name: httpd package is present
  yum:
    name: httpd
    state: latest
  notify: restart httpd

- name: latest index.html file is present
  copy:
    src: files/index.html
    dest: /var/www/html/

handlers:
- name: restart httpd
  service:
    name: httpd
    state: restarted
Plays & Playbooks

Plays are ordered sets of tasks to execute against host selections from your inventory. A playbook is a file containing one or more plays.
- name: install and start apache
  hosts: web
  become: yes
  vars:
    http_port: 80

tasks:
- name: httpd package is present
  yum:
    name: httpd
    state: latest

- name: latest index.html file is present
  copy:
    src: files/index.html
    dest: /var/www/html/
Human-Meaningful Naming

- **name:** install and start apache
  hosts: web
  become: yes
  vars:
    http_port: 80

  tasks:
    - **name:** httpd package is present
      yum:
        name: httpd
        state: latest

    - **name:** latest index.html file is present
      copy:
        src: files/index.html
        dest: /var/www/html/
- name: install and start apache
  hosts: web
  become: yes
  vars:
    http_port: 80

tasks:
- name: httpd package is present
  yum:
    name: httpd
    state: latest

- name: latest index.html file is present
  copy:
    src: files/index.html
    dest: /var/www/html/
Privilege Escalation

- name: install and start apache
  hosts: web
  become: yes
  vars:
    http_port: 80

tasks:
- name: httpd package is present
  yum:
    name: httpd
    state: latest

- name: latest index.html file is present
  copy:
    src: files/index.html
    dest: /var/www/html/
Play Variables

- name: install and start apache
  hosts: web
  become: yes
  vars:
    http_port: 80

tasks:
- name: httpd package is present
  yum:
    name: httpd
    state: latest

- name: latest index.html file is present
  copy:
    src: files/index.html
    dest: /var/www/html/
Tasks

- name: install and start apache
  hosts: web
  become: yes
  vars:
    http_port: 80

  tasks:
  - name: latest httpd package is present
    yum:
      name: httpd
      state: latest

  - name: latest index.html file is present
    copy:
      src: files/index.html
      dest: /var/www/html/
Lab # 2: A Simple Playbook
Doing More with Playbooks

Here are some more essential playbook features that you can apply:

- Templates
- Loops
- Conditionals
- Tags
- Blocks
Templates

Ansible embeds the **Jinja2 templating engine** that can be used to dynamically:

- Set and modify play variables
- Conditional logic
- Generate files such as configurations from variables
Loops

Loops can do one task on multiple things, such as create a lot of users, install a lot of packages, or repeat a polling step until a certain result is reached.

- yum:
  - name: "{{ item }}"
  - state: latest
with_items:
- httpd
- mod_wsgi
Conditionals

Ansible supports the conditional execution of a task based on the run-time evaluation of variable, fact, or previous task result.

- yum:
  name: httpd
  state: latest
when: ansible_os_family == "RedHat"
Tags

Tags are useful to be able to run a subset of a playbook on-demand.

- `yum`:
  - `name: "{{ item }}"`
  - `state: latest`
  - `with_items:
    - httpd
    - mod_wsgi`
  - `tags:
    - packages`

- `template`:
  - `src: templates/httpd.conf.j2`
  - `dest: /etc/httpd/conf/httpd.conf`
  - `tags:
    - configuration`
Blocks

Blocks cut down on repetitive task directives, allow for logical grouping of tasks and even in play error handling.

```yaml
- block:
  - yum:
      name: "{{ item }}"
      state: latest
      with_items:
        - httpd
        - mod_wsgi
  - template:
      src: templates/httpd.conf.j2
      dest: /etc/httpd/conf/httpd.conf
      when: ansible_os_family == "RedHat"
```
Roles

Roles are a packages of closely related Ansible content that can be shared more easily than plays alone.

- Improves readability and maintainability of complex plays
- Eases sharing, reuse and standardization of automation processes
- Enables Ansible content to exist independently of playbooks, projects -- even organizations
- Provides functional conveniences such as file path resolution and default values
Project with Embedded Roles Example

site.yml
roles/
  common/
   files/
   templates/
   tasks/
   handlers/
   vars/
   defaults/
   meta/
apache/
files/
templates/
tasks/
handlers/
vars/
defaults/
meta/
Project with Embedded Roles Example

# site.yml
...  
- hosts: web  
  roles:  
    - common  
    - apache
Ansible Galaxy

http://galaxy.ansible.com

Ansible Galaxy is a hub for finding, reusing and sharing Ansible content.

Jump-start your automation project with content contributed and reviewed by the Ansible community.
Lab #3:
A Playbook Using Roles

Lab #4:
Using an Ansible Galaxy Role
Next Steps

- **It's easy to get started**
  ansible.com/get-started

- **Join the Ansible community**
  ansible.com/community

- **Would you like to learn a lot more?**
  redhat.com/en/services/training/do407-automation-ansible