An introduction to ANSIBLE

Anand Buddhdev
RIPE NCC

What is Ansible?

- A fictional machine capable of **instantaneous** communication :)
  - Star Trek communicators
- An IT automation tool
  - run one-time tasks
  - configure systems
  - maintain state

Features of Ansible

- Written in Python
  - easy to read and extend
- Open source
  - maintained on GitHub
- Easy to install and run
  - get started in just a few minutes
- Scales from a handful of systems to hundreds

Requirements

- Python
  - Jinja2
  - MarkUpSafe
  - PyYAML
  - paramiko (optional)
  - pycrypto
- OpenSSH

Installation

- Any OS
  - create a virtualenv
  - `pip install ansible`
- CentOS / Fedora
  - enable EPEL repo
  - `yum install ansible`

Components of Ansible

- Programs
  - ansible
  - ansible-doc
  - ansible-galaxy
  - ansible-playbook
  - ansible-pull
- Modules
  - Perform configuration and system management
  - copy, service, yum, cron, sysctl, user, group, etc

Inventory

- A file containing all managed host names
- Allows arbitrary groups of hosts
- Can be a directory
  - File names are groups
  - File contents are concatenated
- Can be an executable program
  - Should produce list of hosts and groups

Example inventory

- Example location: /etc/ansible/hosts

```
aws1
aws2
aws3
aws4

[dns_servers]
aws1
aws2

[web_servers]
aws1
aws3
aws4

[mail_servers]
aws1
aws3
```

Running Ansible

- ansible <hosts> -m <module> -a <parameters>
- ansible all -m ping

```
aws3 | success >> {"changed": false, "ping": "pong"}

aws1 | success >> {"changed": false, "ping": "pong"}

aws4 | success >> {"changed": false, "ping": "pong"}

aws2 | success >> {"changed": false, "ping": "pong"}
```

Running Ansible

- ansible aws1 -m command -a whoami

```
aws1 | success | rc=0 >>
ec2-user
```

Running Ansible

- ansible web_servers -a ‘ls -l /etc/passwd’ -o
  - command module is the default
  - results on a single line with -o

```
aws1 | success | rc=0 | (stdout) -rw-r--r--. 1 root root 1428 Jan 17 06:42 /etc/passwd
aws3 | success | rc=0 | (stdout) -rw-r--r--. 1 root root 1428 Jan 17 06:50 /etc/passwd
aws4 | success | rc=0 | (stdout) -rw-r--r--. 1 root root 1428 Jan 17 06:50 /etc/passwd
```

Running Ansible

- ansible all -m yum -a name=screen -s -o
  - use **sudo** to run command

```
aws1 | success >> { "changed": true, "msg": "", "rc": 0, "results": [ .. ] }

aws2 | success >> { "changed": true, "msg": "", "rc": 0, "results": [ .. ] }

aws4 | success >> { "changed": false, "msg": "", "rc": 0, "results": [ .. ] }

aws3 | success >> { "changed": true, "msg": "", "rc": 0, "results": [ .. ] }
```

Running Ansible

- ansible dns_servers -m yum -a name=nsd -s
- ansible dns_servers -m copy -a ‘src=nsd.conf dest=/etc/nsd/nsd.conf’ -s
- ansible dns_servers -m service -a ‘name=nsd state=started’ -s

Playbooks

- Recipes of what to do, and on which hosts
- Written in YAML
- Allows setting variables
- Can contain handlers
  - Take an action upon a change of state
- Re-usable

Simple playbook

- File called httpd.yml

```
- hosts: web_servers
  sudo: true
  tasks:
  - name: install httpd
    yum: name=httpd state=latest
  - name: copy httpd.conf
    copy: src=httpd.conf dest=/etc/httpd/conf
    notify: restart httpd
  - name: ensure httpd is enabled and running
    service: name=httpd enabled=yes state=started
  handlers:
  - name: restart httpd
    service: name=httpd state=restarted
```

Executing a playbook

- ansible-playbook httpd.yml

```
PLAY [web_servers] ************************************************************

GATHERING FACTS ***************************************************************
ok: [aws3]
ok: [aws1]
ok: [aws4]

TASK: [install httpd] *********************************************************
ok: [aws1]
changed: [aws4]
changed: [aws3]

TASK: [copy httpd.conf] *******************************************************
changed: [aws4]
changed: [aws3]
changed: [aws1]

TASK: [ensure httpd is enabled and running] ***********************************
changed: [aws3]
changed: [aws4]
changed: [aws1]

NOTIFIED: [restart httpd] *****************************************************
changed: [aws1]
changed: [aws3]
changed: [aws4]

PLAY RECAP ********************************************************************
aws1                       : ok=5    changed=3    unreachable=0    failed=0
aws3                       : ok=5    changed=4    unreachable=0    failed=0
aws4                       : ok=5    changed=4    unreachable=0    failed=0
```

Idempotence

- ansible-playbook httpd.yml

```
PLAY [web_servers] ************************************************************

GATHERING FACTS ***************************************************************
ok: [aws1]
ok: [aws3]
ok: [aws4]

TASK: [install httpd] *********************************************************
ok: [aws1]
ok: [aws3]
ok: [aws4]

TASK: [copy httpd.conf] *******************************************************
ok: [aws3]
ok: [aws4]
ok: [aws1]

TASK: [ensure httpd is enabled and running] ***********************************
ok: [aws3]
ok: [aws4]
ok: [aws1]

PLAY RECAP ********************************************************************
aws1                       : ok=4    changed=0    unreachable=0    failed=0
aws3                       : ok=4    changed=0    unreachable=0    failed=0
aws4                       : ok=4    changed=0    unreachable=0    failed=0
```

Limiting runs to some hosts

- Use the -l (lowercase L) option
- ansible-playbook httpd.yml -l aws1

Templates

- Ansible uses the Jinja template engine
  - variable substitution
  - conditionals and loop controls (if, then, for)
  - filters to transform text
- Ansible makes host facts available to Jinja

Variables in playbooks

```
- hosts: dns_servers
  vars:
    nsd_procs: 8
    zones: [ in-addr.arpa, ip6.arpa ]
  tasks:
  - name: nsd config file
    template: src=nsd.conf.j2 dest=/etc/nsd
```

Template example

- Template saved as nsd.conf.j2

```
# My NSD configuration
server:
  server-count: {{nsd_procs}}
  identity: {{ansible_fqdn}}
{% for x in range(5) %}
  ip-address: 193.0.9.{{x}}
{% endfor %}

{% for zone in zones %}
zone:
name: {{zone}}
request-xfr: 1.2.3.4
{% endfor %}
```

Roles

- Playbooks can become large and unreadable
- Roles allow grouping of related tasks, files, templates, variables and handlers
- Role directory structure:

```
myrole/
  files/{file1.conf,file2.txt}
  handlers/main.yml
  tasks/main.yml
  templates/{file3.conf.j2,otherfile.j2}
  vars/main.yml
```

Roles in playbooks

```
- hosts: all
  roles:
    - users
    - ntp
- hosts: mail_servers
  roles:
    - exim
    - dovecot
- hosts: dns_servers
  roles:
    - nsd
    - tcpdumper
```

Features of “push mode”

- Lightweight - no set-up required on managed nodes
- Works well for small numbers of hosts
- Instantly write new playbooks and run them
- Does not scale with large numbers of hosts
- Need to run periodically to maintain state
- Playbook runs take longer as hosts as added

“local mode” Ansible

- Install Ansible on a managed host
- Use **ansible-pull** or **rsync** to check out a repository
  - bzr, git, hg, subversion
- Run a playbook with the “-c local” option
- Works a lot like puppet or cfengine in this mode
- Scales well
- Maintains state even if node is disconnected

RIPE NCC’s Ansible setup

- Entire Ansible setup is in a git repo
- Includes a portable pure-python Ansible distribution
  - runs from users’ laptops as well as locally on nodes
  - guarantees the same version of Ansible and modules everywhere
- Contains two playbooks
  - bootstrap.yml, for bootstrapping newly installed nodes
  - main.yml, the main workhorse with all roles defined

Ansible on managed hosts

- Run bootstrap.yml once by hand for a new host
  - installs the minimum set of packages needed for initial run
  - installs a shell script to rsync our portable Ansible setup
  - starts an upstart job to run the script every 10 minutes
  - fetches the new host’s SSH key and commits it to our repo
- The operator can then “git push”

Distribution server

- Checks out the git repo every 5 minutes
- Runs ansible locally
  - configures its authorized_keys file to allow managed hosts to connect
  - runs rrsync (restricted rsync) to provide ONLY the repo to managed hosts

Ansible on managed hosts

- Hosts use rsync to check out the ansible repo
- Run ansible-playbook locally
  - `playbooks/main.yml -l $(hostname)`
- Logs playbook runs to /var/log/ansible.log
- Playbook runs even if rsync fails
  - maintains state on the host

Future improvements

- Currently the entire repo is synced
  - rsync-level ACLs can limit which roles are exposed
- No active feedback from playbook runs
  - use ansible callbacks to send email or some other feedback about failed playbook runs or changes