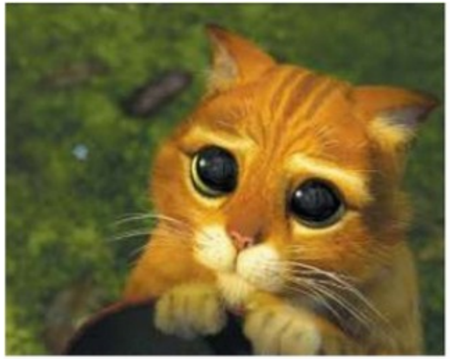
Introduction to Ansible

Network Startup Resource Center

UNIVERSITY OF OREGON

NSRC
Network Startup Resource Center

Are your servers Pets or Cattle?



- Pets are given names like pussinboots.cern.ch
- They are unique, lovingly hand raised and cared for
- When they get ill, you nurse them back to health

- Cattle are given numbers like vm0042.cern.ch
- They are almost identical to other cattle
- When they get ill, you get another one

Source: http://www.slideshare.net/gmccance/cern-data-centre-evolution

What is Ansible?

- A *configuration management* tool
- Applies changes to your system to bring it to a desired state
- Similar applications include puppet, chef, salt, juju, cfengine

Why choose Ansible?

- Target system requires only sshd and python
  - No daemons or agents to install
- Security
  - Relies on ssh
- Easy to get started, compared to the others!

Ansible running with cowsay

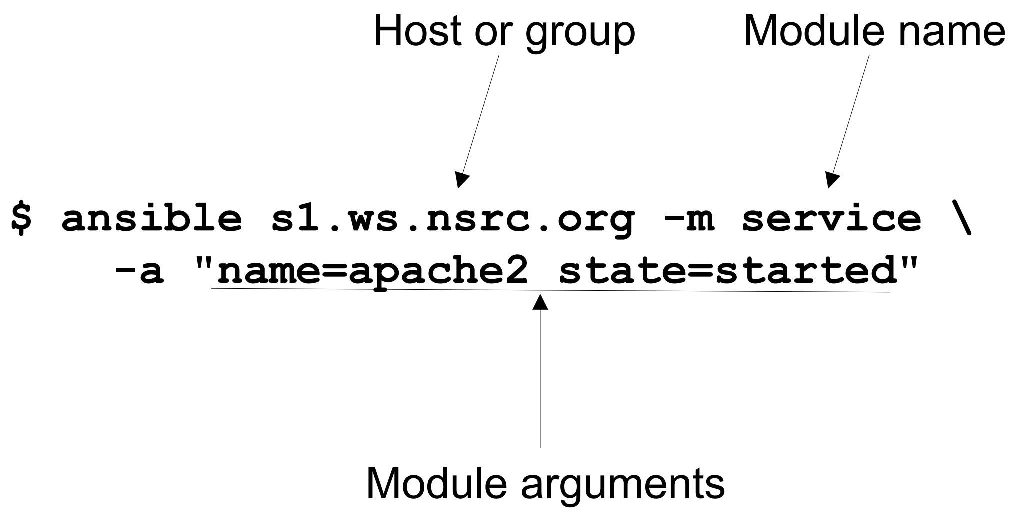
```
 ______________________________
< TASK: [install /etc/hosts] >
 ------------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||


ok: [pc1.example.com]
```

Modules

- Ansible “modules” are small pieces of code which perform one function
  - e.g. copy a file, start or stop a daemon
- Most are “idempotent”: running repeatedly has the same effect as running once
  - only makes a change when the system is not already in the desired state
- Many modules supplied as standard
  - https://docs.ansible.com/modules.html

Invoking modules from shell



```
$ ansible s1.ws.nsrc.org -m service \
    -a "name=apache2 state=started"
```

UNIVERSITY OF OREGON

NSRC
Network Startup Resource Center

Configuring Ansible behaviour

- *Tasks* are modules called with specific arguments
- *Handlers* are triggered when something changes
  - e.g. restart daemon when a config file is changed
- *Roles* are re-usable bundles of tasks, handlers and templates
- All defined using YAML

Diversion: YAML

- A way of storing structured data as text
- Conceptually similar to JSON
  - String and numeric values
  - Lists: ordered sequences
  - Hashes: unordered groups of key-value pairs
- String values don't normally need quotes
- Lists and hashes can be nested
- Indentation used to define nesting

YAML list (ordered sequence)

- Single line form

```
[birth, taxes, death]
```

- Multi-line form

```
- birth
- taxes
- death
```

Space after dash required

UNIVERSITY OF OREGON

NSRC Network Startup Resource Center

YAML hash (key-value pairs)

- Single line form

```
{item: shirt, colour: red, size: 42}
```

Space after colon required

- Multi-line form

```
item: shirt
colour: red
size: 42
description: |
  this is a very long multi-line
  text field which is all one value
```

Nesting: list of hashes

- Compact

```
- {item: shirt, colour: red, size: 42}
- {item: shirt, colour: blue, size: 44}
```

- Multi-line

Note alignment

```
- item: shirt
  colour: red
  size: 42
- item: shirt
  colour: blue
  size: 44
```

More complex YAML example

A list with 3 items

Each item is a hash (key-value pairs)

Simple value

List value (note indentation)

```
- do: laundry
  items:
    - trousers
    - shirts
- do: polish
  items:
    - shoes
    - buckle
- do: relax
  eat:
    - chocolate
    - chips
```

Ansible playbook

Top level: a list of "plays"

Each play has "hosts" plus "tasks" and/or "roles"

```
- hosts:
    - pc1.example.com
    - pc3.example.com
  tasks:
    - name: install Apache
      action: apt pkg=apache2 state=present
    - name: ensure Apache is running
      action: service name=apache2 state=started
- hosts: dns_servers
  roles:
    - dns_server
    - ntp
```

YAML structured module args

Now preferred over key=value args

```
- hosts:
    - pc1.example.com
    - pc3.example.com
  tasks:
    - name: install Apache
      apt:
        pkg: apache2
        state: present
    - name: ensure Apache is running
      service:
        name: apache2
        state: started
```

Roles

- A bundle of related tasks/handlers/templates

```
roles/<rolename>/tasks/main.yml
roles/<rolename>/handlers/main.yml
roles/<rolename>/defaults/main.yml
roles/<rolename>/files/...
roles/<rolename>/templates/...

### Recommended way to make re-usable configs

### Not all these files need to be present
```

UNIVERSITY OF OREGON

NSRC Network Startup Resource Center

Tags

- Each role or individual task can be labelled with one or more "tags"
- When you run a playbook, you can tell it only to run tasks with a particular tag: `-t <tag>`
- Lets you selectively run parts of playbooks

Inventory

- Lists all hosts which Ansible may manage
- Defaults to simple "INI" format, but YAML is an option
- Can define groups of hosts
- Default is `/etc/ansible/hosts`
  - Can override in `ansible.cfg`
  - Can override using `-i <filename>`

Inventory (hosts) example

```
[dns_servers]            ← Name of group
pc1.example.com          ← Hosts in this group
pc2.example.com

[misc]
pc3.example.com
pc4.example.com

# Note: the same host can be listed under
# multiple groups.
# Group "all" is created automatically.
```

Dynamic Inventory

- Inventory can also be read from other systems using inventory plugins, e.g.
  - AWS EC2 API
  - Proxmox API
  - Netbox
  - ...

Inventory variables

- You can set variables on hosts or groups of hosts
- Variables can make tasks behave differently when applied to different hosts
- Variables can be inserted into templates
- Some variables control how Ansible connects

Setting host vars

- Directly in the inventory (hosts) file

```
[core_servers]
pc1.example.com ansible_connection=local
pc2.example.com
```

- In file **`host_vars/pc2.example.com`**

```
ansible_ssh_host: 10.10.0.241
ansible_ssh_user: root
flurble:
  - foo
  - bar
# This is in YAML and is preferred
```

Setting group vars

- **`group_vars/dns_servers`**

```
# More YAML
flurble:
  - baz
  - qux
```

- **`group_vars/all`**

```
# More YAML, applies to every host
# Note: host vars take priority over group vars
```

"Facts"

- Facts are variables containing information collected automatically about the target host
- Things like what OS is installed, what interfaces it has, what disk drives it has
- Can be used to adapt roles automatically to the target system
- Gathered every time Ansible connects to a host (unless playbook has "gather_facts: no")

Showing facts

Invoke the "setup" module

```
$ ansible localhost -m setup | less
localhost | success >> {
    "ansible_facts": {
        "ansible_distribution": "Ubuntu",
        "ansible_distribution_version": "22.04",
        "ansible_domain": "ws.nsrc.org",
        "ansible_eth0": {
            "ipv4": {
                "address": "10.10.0.241",
                "netmask": "255.255.255.0",
                "network": "10.10.0.0"
            },  … etc
```

UNIVERSITY OF OREGON

NSRC Network Startup Resource Center

jinja2 template examples

- Insert a variable into text

```
INTERFACES="{{ dhcp_interfaces }}"
```

- Looping over lists

```
search ws.nsrc.org
{% for host in dns_servers %}
nameserver {{ host }}
{% endfor %}
```

Many other cool features

- Conditionals

```
- action: apt pkg=apache2 state=present
  when: ansible_os_family=='Debian'
```

- Loops

```
- action: apt pkg={{item}} state=present
  with_items:
    - openssh-server
    - acpid
    - rsync
    - telnet
```

Getting up-to-date Ansible

- "ansible-core" is the main package, and "ansible" adds a number of plugins (aka "collections")
- Your package manager's version may be old
- For Ubuntu LTS: latest release is in a PPA

```
apt install software-properties-common
add-apt-repository ppa:ansible/ansible
apt update
apt install ansible
```

- Standard collection is **ansible.builtin**
- More collections installable using *ansible-galaxy*

More info and documentation

- https://docs.ansible.com/
- https://jinja.palletsprojects.com/en/stable/templates/