ANSIBLE

ARNAUD MORIN

ANSIBLE IS POWERFUL

- orchestration
- deployment
- configuration

WHY USING SUCH TOOL?

- Manage a fleet of nodes
  - Pet versus Cattle
- Automate deployment of software
  - Continuous integration and delivery
  - Reproductibility
- Configuration holding
  - Make sure that the system is always in a good state
- Laziness

VOCABULARY

- Host inventory
- Tasks
- Modules
- Playbooks
- Roles
- Idempotent

MAIN CONFIG FILE

- /etc/ansible/hosts
- /etc/ansible/ansible.cfg

ANSIBLE INVENTORY

FROM SIMPLE HOST INVENTORY...

```
$ cat /etc/ansible/hosts
neutron # we can use names
glance
192.168.1.81 # but also ip addresses
```

... TO COMPLEX DYNAMIC INVENTORIES

- pulling inventory dynamically from cloud (such as Ope
- more info https://docs.ansible.com/ansible/2.5/user_guide/intro_ use-of-inventory-script

ANSIBLE COMMAND LINE TOOLS

MAIN COMMANDS

- ansible
- ansible-config
- ansible-console
- ansible-doc
- ansible-galaxy
- ansible-inventory
- ansible-playbook
- ansible-pull
- ansible-vault

FIRST AD HOC COMMAND

```
$ ansible all -m ping
192.168.1.81 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
glance | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
neutron | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

YOU CAN ALSO EXECUTE COMMAND ONLY ON ONE NODE ...

```
$ ansible 192.168.1.81 -m ping
192.168.1.81 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

... OR EXECUTE SHELL COMMAND ON ALL NODES

```
$ ansible all -m shell -a "/bin/echo hello"
192.168.1.81 | CHANGED | rc=0 >>
hello

neutron | CHANGED | rc=0 >>
hello

glance | CHANGED | rc=0 >>
hello
```

ANSIBLE-CONSOLE

run interactive ah hoc commands against a chosen inventory

```
$ ansible-console all
Welcome to the ansible console.
Type help or ? to list commands.

root@all (3)[f:5]$ uname -r
neutron | CHANGED | rc=0 >>
4.4.0-138-generic

glance | CHANGED | rc=0 >>
4.4.0-138-generic

192.168.1.81 | CHANGED | rc=0 >>
4.4.0-138-generic

root@all (3)[f:5]$
```

ANSIBLE-GALAXY

Search/download role from ansible galaxy (like a store)

```
$ ansible-galaxy search asterisk
...
```

```
$ ansible-galaxy install dgnest.asterisk
- downloading role 'asterisk', owned by dgnest
- downloading role from https://github.com/dgnest/ansible-role-as
- extracting dgnest.asterisk to /root/.ansible/roles/dgnest.aster
- dgnest.asterisk (0.0.5) was installed successfully
```

ANSIBLE-INVENTORY

used to display or dump the configured inventory as Ansible sees it

```
$ ansible-inventory --yaml --list
all:
  children:
    ungrouped:
      hosts:
        192.168.1.81: {}
        glance: {}
        neutron: {}
```

ANSIBLE-PLAYBOOK

the tool to run Ansible playbooks, which are a configuration and multinode deployment system

```
$ ansible-playbook play.yml --list-tasks

playbook: play.yml

  play #1 (all): all    TAGS: []
    tasks:
      install cowsay and steam locomotive    TAGS: []
```

ANSIBLE-PULL

pulls playbooks from a VCS repo and executes them for the local host

ANSIBLE-VAULT

encryption/decryption utility for Ansible data files

FOCUS ON PLAYBOOKS

PLAYBOOKS

- YAML files
- playbooks are composed of one or more **plays**
- plays are composed of **roles** and **tasks**
- roles are composed of **tasks**
- tasks are based on **modules** to perform specific actions on nodes

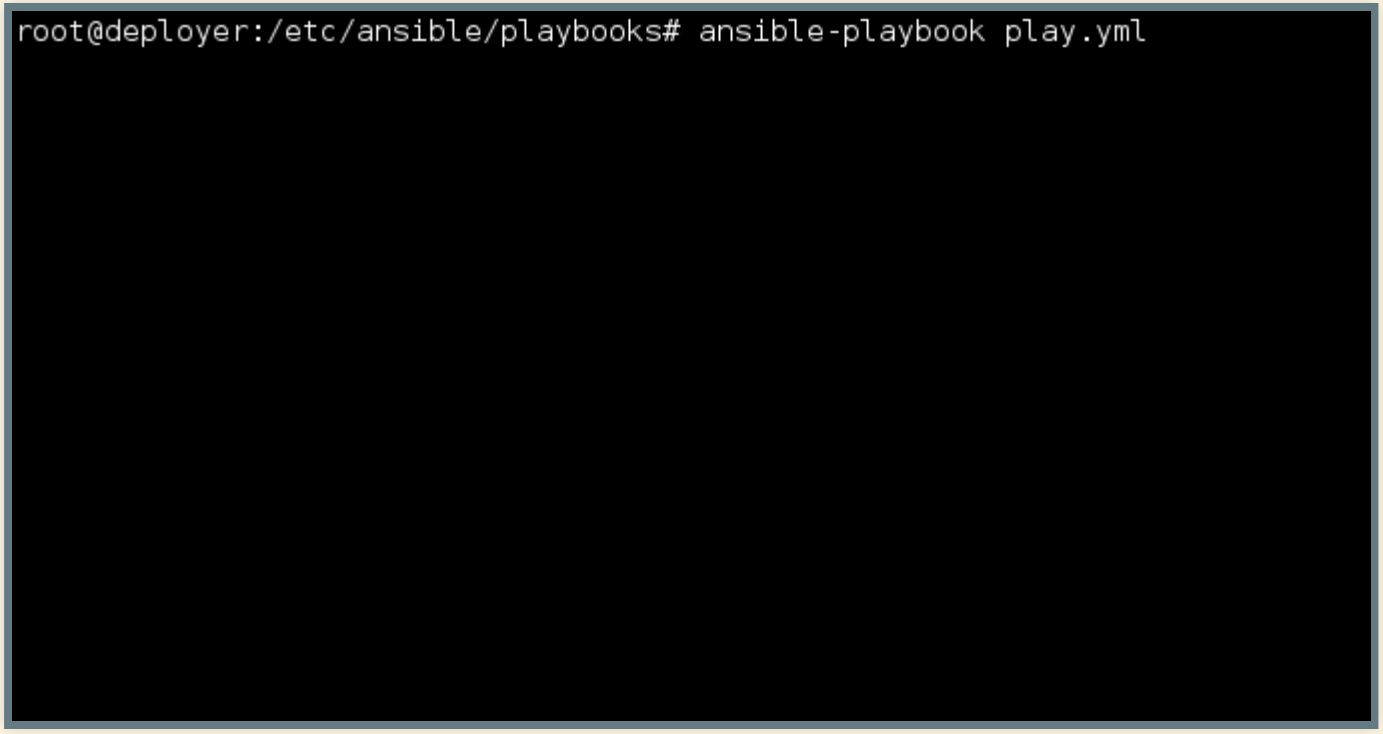
PLAYBOOKS

- Applied to a group af node
- Allow orchestration between nodes

EXAMPLE

```
---
- hosts: localhost                    # Apply only to localhost
  tasks:
  - name: install cowsay and sl       # Use apt module
    apt:
      name: ['cowsay', 'sl']          # Ensure that those packages
      state: latest                   # are installed
```

ANSIBLE-PLAYBOOK PLAY.YML



ANSIBLE ROLES

Roles are ways of automatically do tasks that are related (such as installing and configuring a software)

They are also nice for:

- sharing with other (or use others roles - see ansible-galaxy)
- reuse (you can use same roles in multiples plays)

EXAMPLE

```
$ cat /etc/ansible/playbooks/play.yml
---
- hosts: localhost
  roles:
  - useless
```

```
$ cat /etc/ansible/roles/useless/tasks/main.yml
---
- name: install cowsay and steam locomotive
  apt:
    name: ['cowsay', 'sl']
    state: latest
```

VARIABLES

DEFINITION

Variables can be defined in

- inventory
- playbooks
- roles
- on CLI
- retrieved from facts

DEFINING A VARIABLE

General yaml syntax

```
foo:
  field1: one
  field2: two
```

Accessing it

```
foo.field1
# or
foo['field1']
```

EXAMPLE IN A PLAYBOOK

In a playbook

```
- hosts: webservers
  vars:
      http_port: 80
  # or from external file
  vars_files:
      - /vars/external_vars.yml
```

From CLI

```
ansible-playbook bla.yml --extra-vars "foo=bar truc=bidule"
```

USING VARIABLES WITH JINJA2

Ansible is using Jinja2 as templating system.

Example:

```
Foo value is {{ foo }}
```

FACTS: SPECIAL VARIABLES

Ansible is collectings some facts when connecting to the nodes.

Those facts are then available as variables, like any other.

Example:

```
My hostname is {{ ansible_facts['nodename'] }}
```

FACTS: SPECIAL VARIABLES

See all gathered facts:

```
ansible hostname -m setup
```

REGISTERING VARIABLES

You can register variables from result of a task.

Example

```
- hosts: web_servers

  tasks:

     - shell: /usr/bin/foo
       register: foo_result
       ignore_errors: True

     - shell: /usr/bin/bar
       when: foo_result.rc == 5
```

CONDITIONALS, LOOPS AND BLOCKS

THE WHEN STATEMENT

Use very often to execute tasks only if required

Example

```
tasks:
  - name: "shut down Debian flavored systems"
    command: /sbin/shutdown -t now
    when: ansible_facts['os_family'] == "Debian"
```

LOOPS

Sometime, you want to execute the same task multiple times with differents values

Example

```
tasks:
  - name: add several users
    user:
      name: "{{ item }}"
      state: present
      groups: "wheel"
    loop:
       - testuser1
       - testuser2
```

BLOCKS

Blocks allow for logical grouping of tasks and in play error handling (try/catch)

Example

```
tasks:
 - name: Handle the error
   block:
     - debug:
         msg: 'I execute normally'
     - name: i force a failure
       command: /bin/false
     - debug:
         msg: 'I never execute, due to the above task failing, :-
   rescue:
     - debug:
         msg: 'I caught an error, can do stuff here to fix it, :-
```

MORE INFO

https://docs.ansible.com/ansible/latest/


QUESTIONS ?