TP part 03 - Ansible

Checkpoint: call us to check your results

Ask yourself: how? why?

Point to document/report

Goals

Deploy your application automatically with ansible.

Intro

Inventories

By default, Ansible's inventory is saved in the location /etc/ansible/hosts where you already defined your server. The headings between brackets (eg: [webservers]) are used to group sets of hosts together, they are called, surprisingly, groups. You could regroup them by roles like database servers, front-ends, reverse proxies, build servers…

Let’s create a project specific inventory, in your project create an ansible directory, then create a new directory called inventories and in this folder a new file (my-project/ansible/inventories/setup.yml):

```
all:
 vars:
   ansible_user: centos
   ansible_ssh_private_key_file: /path/to/private/key
 children:
   prod:
     hosts: hostname or IP
```

Test your inventory with the ping command:

```
$ ansible all -i inventories/setup.yml -m ping
```

Facts

Let's get information about host: These kind of variable, not set by the user but discovered are called **facts**. Facts, are prefixed by *ansible_* and represent informations derived from speaking with your remote systems.

You will request your server to get your OS distribution, thanks to setup module.

```
$ ansible all -i inventories/setup.yml -m setup -a
"filter=ansible_distribution*"
```

Earlier you installed Apache httpd server on your machine, let's remove it:

```
$ ansible all -i inventories/setup.yml -m yum -a "name=httpd state=absent"
--become
```

With ansible, you just describe the state of your server and let ansible automatically update it for you. If you run this command another time you won't have the same output as httpd would have been removed.

Checkpoint: document your server main informations.

Playbooks

First playbook

Let's create a first very simple playbook in my-project/ansible/playbook.yml:

```
- hosts: all
  gather_facts: false
  become: yes

  tasks:
    - name: Test connection
      ping:
```

Just execute your playbook:

```
$ ansible-playbook -i inventories/setup.yml playbook.yml
```

You can check your playbooks before playing them using the option: --syntax-check

Advanced playbook

Let’s create a playbook to install docker on your server, follow the documentation and create the corresponding tasks: https://docs.docker.com/install/linux/docker-ce/centos/.

```yaml
- hosts: all
  gather_facts: false
  become: yes

  tasks:
    # Install Docker
    -   name: Install yum-utils
        yum:
          name: yum-utils
          state: latest

    -   name: Install device-mapper-persistent-data
        yum:
          name: device-mapper-persistent-data
          state: latest

    -   name: Install lvm2
        yum:
          name: lvm2
          state: latest

    -   name: Add Docker stable repository
        yum_repository:
          name: docker-ce
          description: Docker CE Stable - $basearch
          baseurl: https://download.docker.com/linux/centos/7/$basearch/stable
          state: present
          enabled: yes
          gpgcheck: yes
          gpgkey: https://download.docker.com/linux/centos/gpg

    -   name: Install Docker
        yum:
          name: docker-ce
          state: present

    -   name: Make sure Docker is running
        service: name=docker state=started
        tags: docker
```

Good news, we now have docker installed on our server. One task was created to be sure docker was running, you could check this with an ad-hoc command or by connecting to the server until you really trust ansible.

Ask yourself: what is $basearch?

Using role

Our docker install playbook is nice and all but it will be cleaner to have in a specific place, in a role for example. Create a docker role and move the installation task there:

```
$ ansible-galaxy init roles/docker
```

Call the docker role from your playbook to check your refactor and your installation.

Initialized role has a couple of directories, keep only the one you will need:

- tasks - contains the main list of tasks to be executed by the role.
- handlers - contains handlers, which may be used by this role or outside.

Create a common role

This role will be used for common tasks, in our case we will do the following:

- Close all ports of the server
- Open port 80 (for your http traffic)
- Open port 22 (for Secure SHell access)

Hint: use the modules iptables and firewalld.

Which port should be opened for https traffic?

Deploy your app

Time has come to deploy your application to your Ansible managed server.

Create a specific role for your application and use the Ansible module: docker_container to start your dockerized application. Your tasks should look like this:

```
- name: Run HTTPD
 docker_container:
   name: httpd
   image: jdoe/my-httpd:1.0

- name: Run Database
 docker_container:
   name: database
   image: jdoe/my-database:1.0

- name: Run Backend
 docker_container:
   name: backend
   image: jdoe/my-backend:1.0
```

Checkpoint: You should be able to access your API on your server.

Document your docker_container tasks configuration.

Continuous deployment

Configure Travis to deploy automatically your application when you released on the production branch of your github repository.

Checkpoint: Full CI/CD pipeline in action.