Ansible Code Lab

[Maxime Wojtczak]

[Antoine Barbare]

Why Infrastructure as Code?

- Larger applications lead to larger deployments:
  - Repeated operations
  - Human errors
  - Tracability

Why Infrastructure as Code?

- Cloud infrastructure
-
- Pet vs Cattle:
  - Elasticity
  - Disposable hosts

Why Infrastructure as Code?

- Reduce infrastructure deployment costs:
  - More time on added-value rather than repeatition
- Deployment speed:
  - Automation leads to faster host availability
- Risk reduction:
  - Fewer manual operations leads to more reliable deployments
  - Automation increases the update rate

Why Infrastructure as Code?

- Brings the code toolkit to infrastructure:
  - Ability to review changes
  - Reproductibility of a given infrastructure at a given time
  - Tracability

Ansible

- Written in Python
- Extensible via modules and roles
- Push method
- Lightweight:
  - Only a SSH connection is required

Ansible

- Mostly declarative
- YAML, YAML everywhere
  - Like it or hate it :)

Principles

- Inventory:
  - Architecture definition
  - Group division

Example

```
[webserver]
pweb01.yourcompany.com
pweb02.yourcompany.com

[middle]
pmiddle01.yourcompany.com
pmiddle02.yourcompany.com

[database]
pdb01.yourcompany.com
pdb02.yourcompany.com
pdb03.yourcompany.com

[database:vars]
mongodb_version=3.7.9
```

Principles

- Playbook:
  - Link groups & hosts to roles

Example

```
- hosts: all
  roles:
    - certificates
    - node_exporter

- hosts: database
  roles:
    - mongodb

- hosts: webserver
  roles:
    - nginx
```

Principles

- Roles:
  - Actual operations:
    - Services to install
    - Configurations to deploy
    - Operations to do

Example

```
- name: Ensure MongoDB APT key is declared
  apt_key:
    keyserver: keyserver.ubuntu.com
    id: 2930ADAE8CAF5059EE73BB4B58712A2291FA4AD5

- name: Ensure MongoDB repository is present
  apt_repository:
    repo: deb http://repo.mongodb.org/apt/debian jessie/mongoc
    state: present

- name: Ensure MongoDB is installed
  apt:
    name: mongodb-org={{ mongodb_version }}

- name: Ensure MongoDB is configured
  template:
    src: mongod.conf.j2
    dest: /etc/mongod.conf
```

Ansible Training

Agenda

- Couple of exercices
- Hands on Ansible
  - Deploy your VM
  - Deploy your SSH key
  - Deploy system pre-requites
  - Deploy application on remote system

Ansible installation

We'll use ansible 2.6 to use scaleway modules

```
sudo -H pip install
git+git://github.com/ansible/ansible.git@stable-2.6
```

Modules:

- scaleway_compute
- scaleway_sshkey

Technical environment

- 1 VM per user
- Access to VM through SSH keys

VM will be destroyed tonight. Code will be available on Github

Get your Scaleway VM

A playbook for everything

1st: Deploy your ssh key on Scaleway with scaleway_sshkey

```
user@laptop:~#$ tree
.
├── playbook.yml
└── roles
    └── scaleway_vm
        └── tasks
            └── main.yml
```

Module Documentation

Solution

```yaml
#roles/scaleway/tasks/main.yml
- name: deploy ssh key to scaleway
  scaleway_sshkey:
    ssh_pub_key: "ssh-rsa ..."
    state: present
```

```yaml
#playbook.yml
---
- name: Deploy scaleway virtual machine
  gather_facts: no
  hosts: localhost
  environment:
    SCW_TOKEN: "{{ lookup('env', 'SCW_TOKEN') }}"
  roles:
    - scaleway_vm
```

Create your Ubuntu VM (1)

```
user@laptop:~#$ tree
├── playbook.yml
└── roles
    └── scaleway_vm
        └── tasks
            └── main.yml
```

Ubuntu image: `e20532c4-1fa0-4c97-992f-436b8d372c07`
Organization: `43a3b6c8-916f-477b-b7ec-ff1898f5fdd9`

Commercial Type: `VC1S` - Location: `par1`

Specify a custom name (ie not ansible/test/...)

Module Documentation

Create your Ubuntu VM (2)

```
user@laptop:~#$ tree
├── playbook.yml
└── roles
    └── scaleway_vm
        └── tasks
            └── main.yml
```

Ubuntu image: `6d7aabd0-a0b7-434a-95c8-b40aa3d5b973`
Organization: `43a3b6c8-916f-477b-b7ec-ff1898f5fdd9`

Commercial Type: `VC1S` - Location: `ams1`

Specify a custom name (ie not ansible/test/...)

[Module Documentation]

Solution

```
#roles/scaleway/tasks/main.yml
- name: deploy ssh key to scaleway
  scaleway_sshkey:
    ssh_pub_key: "ssh-rsa ..."
    state: present

- name: create a scaleway server
  scaleway_compute:
    name: my_scaleway_server
    state: running
    image: e20532c4-1fa0-4c97-992f-436b8d372c07
    organization: 43a3b6c8-916f-477b-b7ec-ff1898f5fdd9
    region: par1
    commercial_type: VC1S
    tags:
      - my_specific_tag
```

Try out your playbook

As you don't have any server right now, you will launch the playbook on your own machine

```
user@laptop:~# ansible-playbook playbook.yml
PLAY [Deploy scaleway virtual machine] **********

TASK [scaleway_vm : deploy ssh key to scaleway] **********
ok: [localhost]

TASK [scaleway_vm : create a scaleway server] **********
ok: [localhost]

PLAY RECAP **********
localhost: ok=2 changed=0 unreachable=0 failed=0
```

Relaunch it Magic

Create your inventory file

Get the IP Address related to your instance and create your inventory file.
You can take example on `/etc/ansible/hosts`

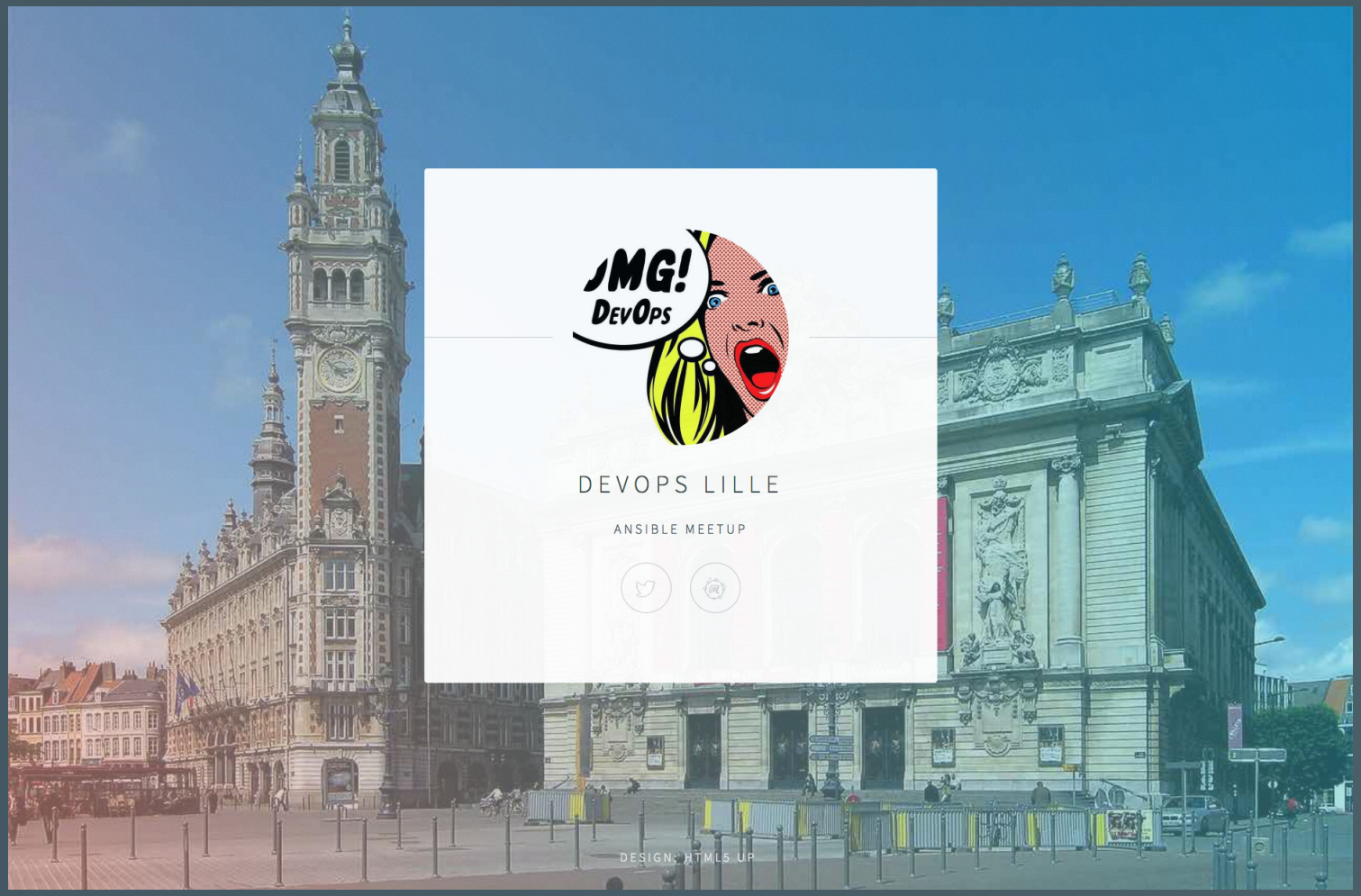
Test your inventory file:

```
user@laptop:~#:~# ansible -i inventory all -m ping
51.15.235.20 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

Make sure to connect with root user with `ansible_user`

Deploying a simple app

We want to deploy devops web application on our VMs



Deploying a simple app

Simple HTML landing page
Served via Nginx Web Server

#TODO

- Create a role to install Nginx web server
- Create a role to deliver our application
- Wrap up roles in a playbook
- Deliver on our server

Nginx role

Create a role and modify your playbook

- Install Nginx on the server
- Start and Enable Nginx at boot
- Wrap up your the role in a playbook

```
[ansible01@ansible ~]$ tree
├── inventory
├── playbook.yml
└── roles
    └── nginx_install
        └── tasks
            └── main.yml
```

Try: `ansible-playbook -i inventory playbook.yml`

Solution

```
#roles/nginx_install/tasks/main.yml
---
- name: Install nginx daemon
  apt:
    name: nginx
    state: present
    update_cache: yes

- name: Start nginx and enable it at boot
  systemd:
    name: nginx
    state: started
    enabled: yes
```

Solution

```yaml
#playbook.yml
---
- name: Deploy devops app
  hosts: all
  roles:
    - nginx_install
```

Check that nginx is installed and available

```
curl scaleway-server-ip
```

Deploy app role

Update your roles and playbook

```
[ansible01@ansible ~]$ tree
├── inventory
├── playbook.yml
└── roles
    ├── deploy_app
    │   ├── files
    │   │   ├── nginx.conf
    │   │   └── nginx-devops.conf
    │   ├── handlers
    │   │   └── main.yml
    │   └── tasks
    │       └── main.yml
    └── nginx_install
        └── tasks
            └── main.yml
```

Try: `ansible-playbook -i inventory playbook.yml`

Deploying application

Devops App resources are available online:

- https://github.com/antoineHC/ansible-meetup-app
- https://github.com/antoineHC/ansible-meetup-nginx

ansible-meetup-app --> `/usr/share/nginx/html/ansible-meetup-app/`

`nginx.conf` --> `/etc/nginx/nginx.conf`
`nginx-devops.conf` --> `/etc/nginx/conf.d/app.conf`

Don't forget to restart nginx with handler after the app deployment

Solution

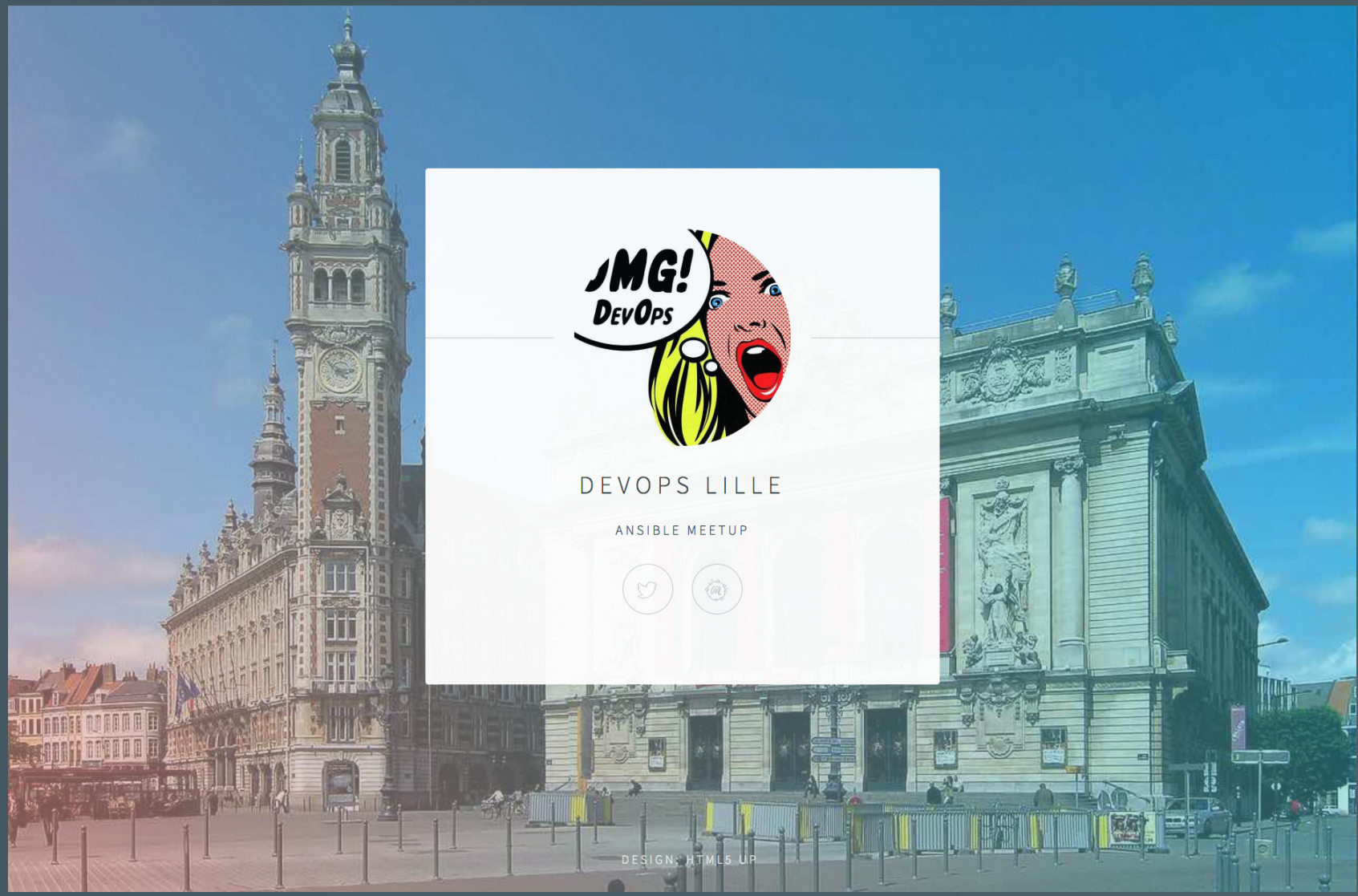
```
#roles/deploy_app/tasks/main.yml
---
- name: deploy devops app from github
  git:
    repo: https://github.com/antoineHC/ansible-meetup-app
    dest: /usr/share/nginx/html/ansible-meetup-app

- name: update nginx conf
  copy:
    src: nginx.conf
    dest: /etc/nginx/nginx.conf
  notify: restart nginx

- name: update nginx devops conf
  copy:
    src: nginx-devops.conf
    dest: /etc/nginx/conf.d/devops.conf
  notify: restart nginx
```

Solution

```yaml
#roles/deploy_app/handlers/main.yml
---
- name: restart nginx
  systemd:
    name: nginx
    state: restarted
  listen: restart nginx
```

```yaml
#playbook.yml
---
- name: Deploy devops app
  hosts: all
  become: yes
  roles:
    - nginx_install
    - deploy_app
```

Check our your app online



Success !

Best practices

- Variables
  - Variables may be declared about anywhere
  - Keep it simple and well organized, better stick to:
    - Inventory
    - Roles' defaults
    - Roles' vars

Best pratices

- Roles
  - Tasks can be defined at the playbook level
  - Prefer roles to keep things well organized

Best practices

- Playbook is not scripting
  - Dependent tasks is OK from time to time
  - When it gets more complicated, consider writing a small module
    - Simple Python, better for tests, readability, and advanced features (check mode, etc.)

Go further

- Ansible Tower
- AWX (Tower upstream)

- Ansible Galaxy

Thanks