

# Infrastructure As Code (IAC)

## Quand les sysadmin se mettent à coder ...

---

Brice Ekane - (brice.ekane@univ-rennes.fr)

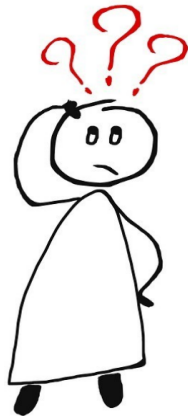
ISTIC Rennes - France  
2024-2025

git clone <https://github.com/bekane/tlc.git>

Cours repris de Pr. Olivier Barais, Avec plusieurs slides issues d'un talk de  
AUKFOOD

# Continuous delivery;)

Ben j'comprends pas, ça marchait il y a encore deux minutes...



CommitStrip.com





L'IAC fait appel à un langage descriptif pour coder des processus de déploiement

Plus polyvalents et adaptatif que des scripts

# Des descripteurs pour piloter des outils de configuration et de déploiement

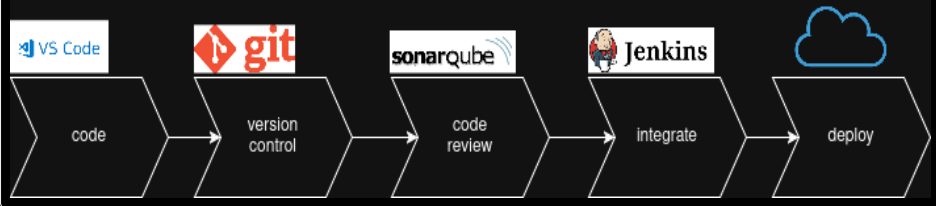
---

- ▶ Provisionner et déployer
- ▶ Obtenir l'état d'une infrastructure (selon l'outil)
- ▶ Gérer le code avec un système de versionnement
- ▶ Mettre en place des tests sur les déploiements
- ▶ Créer des briques réutilisables

Quand les Sysadmins découvrent le génie logiciel ;)

# IAC Workflow

## Infrastructure as Code Workflow





# Un langage descriptif

---



## YAML

- ▶ acronyme récursif de YAML Ain't Markup Language
- ▶ est un format de représentation de données par sérialisation
  - ▶ Unicode.

Wikipedia

# Yaml

## Basic Yaml Syntax Rules

```
1 # This is a comment in YAML
2 name: John Doe           # Simple key-value pair
3 age: 30                  # Number value
4 is_student: false       # Boolean value
5
6 # Nested structure (mapping)
7 address:
8   street: 123 Elm Street
9   city: Anytown
10  zip_code: 12345
11
12 # Mixed list of objects
13 pets:
14   - name: Fluffy
15     type: cat
16   - name: Rex
17     type: dog
18
19 # Multiline text (using a folded block with >)
20 bio: >
21   John Doe is a software developer
22   with a passion for creating
23   clean and efficient code.
```

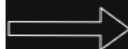
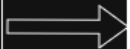
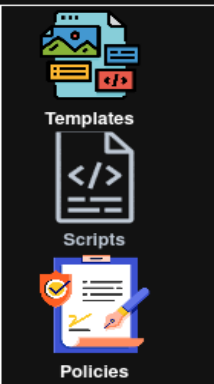
# Les outils

---

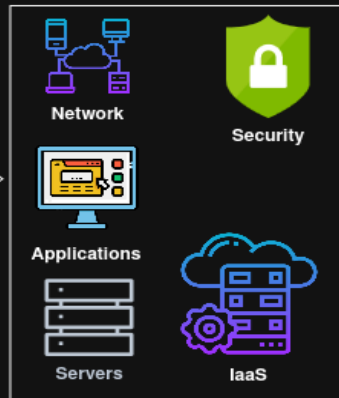


# Un exemple : Tofu

## Rules+DSL



## Target



# Un exemple : Tofu

---



- ▶ OpenTofu est un outils open source qui permet de
  - ▶ **définir** et **déployer** une infrastructure chez les différents fournisseurs cloud (e.g. AWS, Azure, Google Cloud, DigitalOcean, OVH,etc)
  - ▶ tout cela à partir d'un langage déclaratif

library.tf

Providers

Providers are a logical abstraction of an upstream API. They are responsible for understanding API interactions and exposing resources.

### hashicorp / aws

Lifecycle management of AWS resources, including EC2, Lambda, EKS, ECS, VPC, S3, RDS, DynamoDB, and more. This provider is maintained internally by the HashiCorp AWS Provider team.

9 days ago 3.83B

### hashicorp / random

Supports the use of randomness within Terraform configurations. This is a logical provider, which means that it works entirely within Terraform logic, and does not interact with any other services.

4 months ago 1.41B

### hashicorp / null

Provides constructs that intentionally do nothing - useful in various situations to help orchestrate tricky behavior or work around limitations.

4 months ago 1.2B


### hashicorp / google

Lifecycle management of GCP resources, including Compute Engine, Cloud Storage, Cloud SDK, Cloud SQL, GKE, BigQuery, Cloud Functions and more. This provider is collaboratively maintained by the Google Terraform team and the HashiCorp team.

4 days ago 1.12B


- ▶ liste importante de providers
  - ▶ <https://library.tf/providers>
  - ▶ bien documenté

# Exemple OpenTofu avec AWS



```
1 terraform {
2     required_providers {
3         aws = {
4             source = "hashicorp/aws"
5             version = "5.84.0"
6         }
7     }
8 }
9
10 provider aws {
11     # Configuration options
12 }
```


# Exemple OpenTofu avec OVH



```
1 terraform {
2     required_providers {
3         ovh = {
4             source = "ovh/ovh"
5             version = "1.5.0"
6         }
7     }
8 }
9
10 provider ovh {
11     # Configuration options
12 }
```

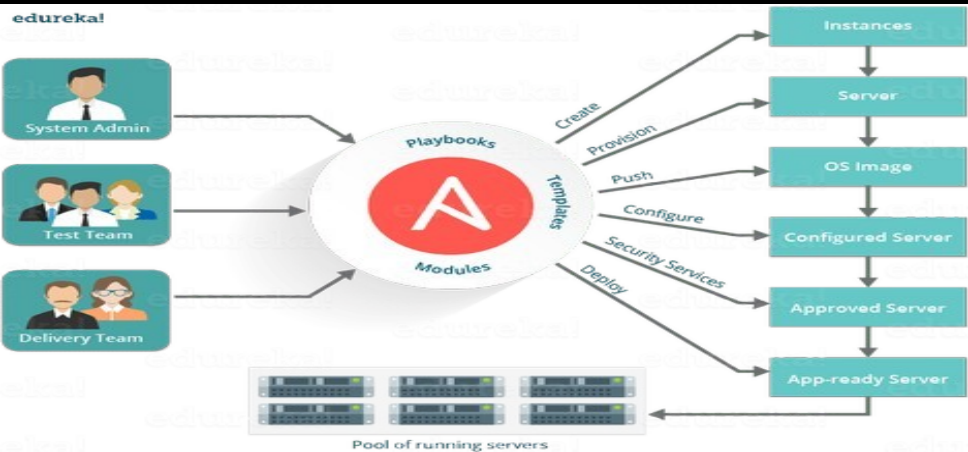


# Exemple OpenTofu avec libvirt



```
1 terraform {
2     required_providers {
3         libvirt = {
4             source = "dmacvicar/libvirt"
5             version = "0.8.1"
6         }
7     }
8 }
9
10 provider libvirt {
11     # Configuration options
12 }
```

# Exemple 2 : Ansible



# Exemple de playbook

```
---
- name: Deploy HAProxy Load Balancer
  hosts: loadbalancers
  become: true
  tasks:
    - name: Install HAProxy
      ansible.builtin.apt:
        name: haproxy
        state: present
        update_cache: yes

    - name: Copy HAProxy Configuration
      ansible.builtin.template:
        src: haproxy.cfg.j2
        dest: /etc/haproxy/haproxy.cfg
        owner: root
        group: root
        mode: '0644'
      notify: Restart HAProxy

    - name: Enable and Start HAProxy
      ansible.builtin.systemd:
        name: haproxy
        enabled: yes
        state: started

  handlers:
    - name: Restart HAProxy
      ansible.builtin.systemd:
        name: haproxy
        state: restarted
```

# Une architecture extensible

---



Un besoin ? Un module ou presque ...

[https://docs.ansible.com/ansible/latest/modules/list\\_of\\_all\\_modules.html](https://docs.ansible.com/ansible/latest/modules/list_of_all_modules.html)

Si pas de module ... le développer en python ..

# Les facts Ansible

Les facts sont les variables renvoyées par les serveurs.

Ansible peut les récupérer et les réutiliser.

Pour les lister :

```
$ ansible web01 -m setup
```

On peut ensuite récupérer ces infos et les réutiliser par exemple avec des when ou dans des templates.

Exemple : *ansible\_eth0.ipv4.address*

```
adobrien@ansiblemaster:~$ ansible windows -m setup
10.0.0.6 | success >> {
  "ansible_facts": {
    "ansible_architecture": "64-bit",
    "ansible_distribution": "Microsoft Windows NT 6.1.7601 Service Pack 1",
    "ansible_distribution_version": "6.1.7601.65536",
    "ansible_fqdn": "w2k8r2",
    "ansible_hostname": "W2K8R2",
    "ansible_interfaces": [
      {
        "default_gateway": "10.0.0.1",
        "dns_domain": "stbyd3d52acefjxphjbz21ptbf.dx.internal.cloudapp.net",
        "interface_index": 11,
        "interface_name": "Microsoft Virtual Machine Bus Network Adapter"
      }
    ],
    "ansible_ip_addresses": [
      "10.0.0.6",
      "fe80::2d7d:e61d:6d7f:f718"
    ],
    "ansible_os_family": "Windows",
    "ansible_os_name": "Microsoft Windows Server 2008 R2 Datacenter ",
    "ansible_powershell_version": 3,
    "ansible_system": "Win32NT",
    "ansible_totalmem": 1879048192,
    "ansible_winrm_certificate_expires": "2016-07-23 19:29:32"
  },
  "changed": false
}
```

# Ansible : Ninja un langage de template



```
1 <html>
2 <head>
3 <meta charset="UTF-8">
4 <title>Server {{ ansible_hostname }}</title>
5 </head>
6 <body>
7 <h1>Hello World!</h1>
8 <h2>Le serveur Web {{ ansible_hostname | capitalize }} fonctionne !! </h2>
9 <p>Cette page à été créée le {{ ansible_date_time.date }}</p>
10 {% if ansible_eth0.active == True %}
11 <p>eth0 address {{ ansible_eth0.ipv4.address }}</p>
12 {% endif %}
13 <p>Ce serveur fonctionne sur un système Linux de type {{ ansible_os_family}}, version
14 {{ ansible_lsb.codename }}</p>
15 <p>Cette machine peut avoir plusieurs adresses ip </p>
16 <ul>
17 {% for address in ansible_all_ipv4_addresses %}
18 <li>{{ address }}</li>
19 {% endfor %}
20 </ul>
21 </body>
22 </html>
```

# Conclusion

---

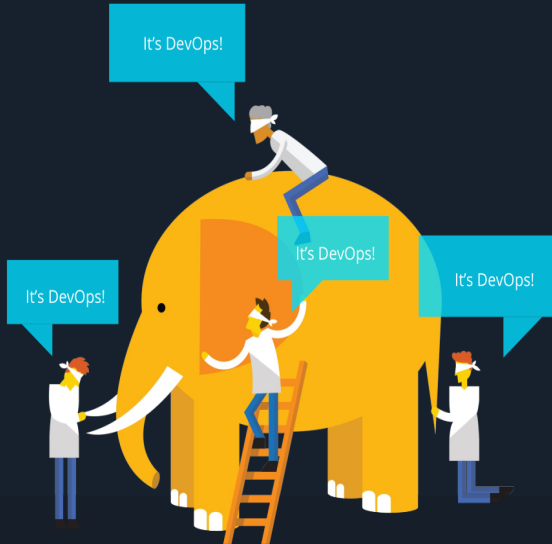
IAC and DevOPs

# IAC and DevOps

“DevOps is development and operations **collaboration**”

“DevOps is using **automation**”

“DevOps is **small** deployments”



“DevOps is treating your infrastructure **as code**”

“DevOps is feature **switches**”

“**Scrum** for Ops?”



# What is DevOps

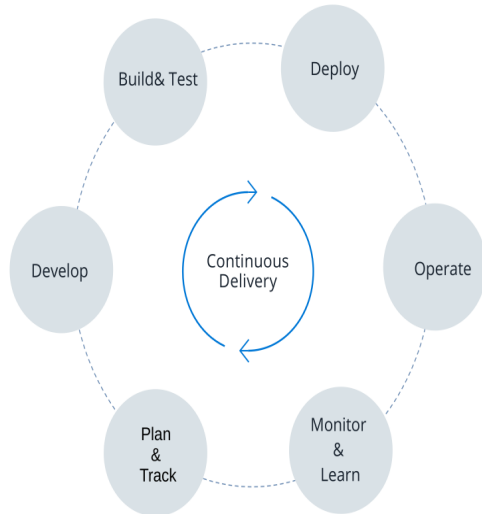
People. Process. Products.

“

DevOps is the union of **people**, **process**, and **products** to enable continuous delivery of **value** to your end user.

”

*Donovan Brown (Microsoft)*



# DevOps Best Practices

---

**Version everything**

**Automate everything**

Tokenize configurations Use  
one-click deployments Deploy the  
same way to every environment

**Have always a rollback  
mechanism in place Build only  
once**

Lock down the environments

...

