

# DevOps

# Team Presentation



**BESSON Adrien**  
Expert DevOps/Java  
+ 8 Yrs XP



[www.linkedin.com/in/a-bess](https://www.linkedin.com/in/a-bess)



**Quentin Aveno**  
Lead Integ  
+ 4 Yrs XP



<https://www.linkedin.com/in/q-a>

# Sommaire

## 1. DevOps

- 1.1 – KESAKO
- 1.2 – Pipeline DevOps
- 1.3 – Intérêts DevOps

## 2. Tools DevOps

- 2.1 – Git GitFlow
- 2.2 – Jenkins
- 2.3 – IaC Terraform
- 2.4 – CaC Ansible

# Planing

Vendredi	Lundi	Mardi	Jeudi
Présentation DevOps  Git Jenkins	IAC  AWS & Terraform	IAC & CAC  Terraform & Ansible	CAC  Ansible  Evaluation

# Evaluation

1. QCM & Questions ouvertes
2. Mini Projet d'automatisation



# 1. Introduction DevOps

# Problématique

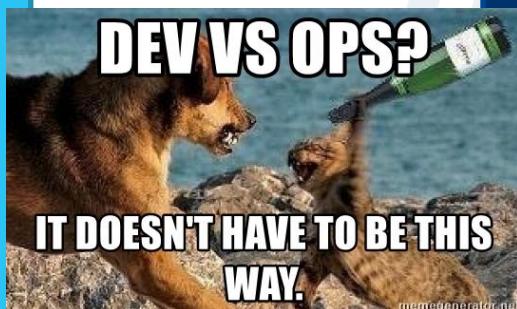
Création



Mais:

- Vocabulaire
- Technologie
- Responsabilité

Exploitation  
Production



# Problématique



# 1.1 DevOps KESAKO

DevOps = Industrialisation de l'IT

Le concept DevOps a été inventé en 2009 par Patrick DEBOIS. Il est née de la collaboration des développements et des opérations.

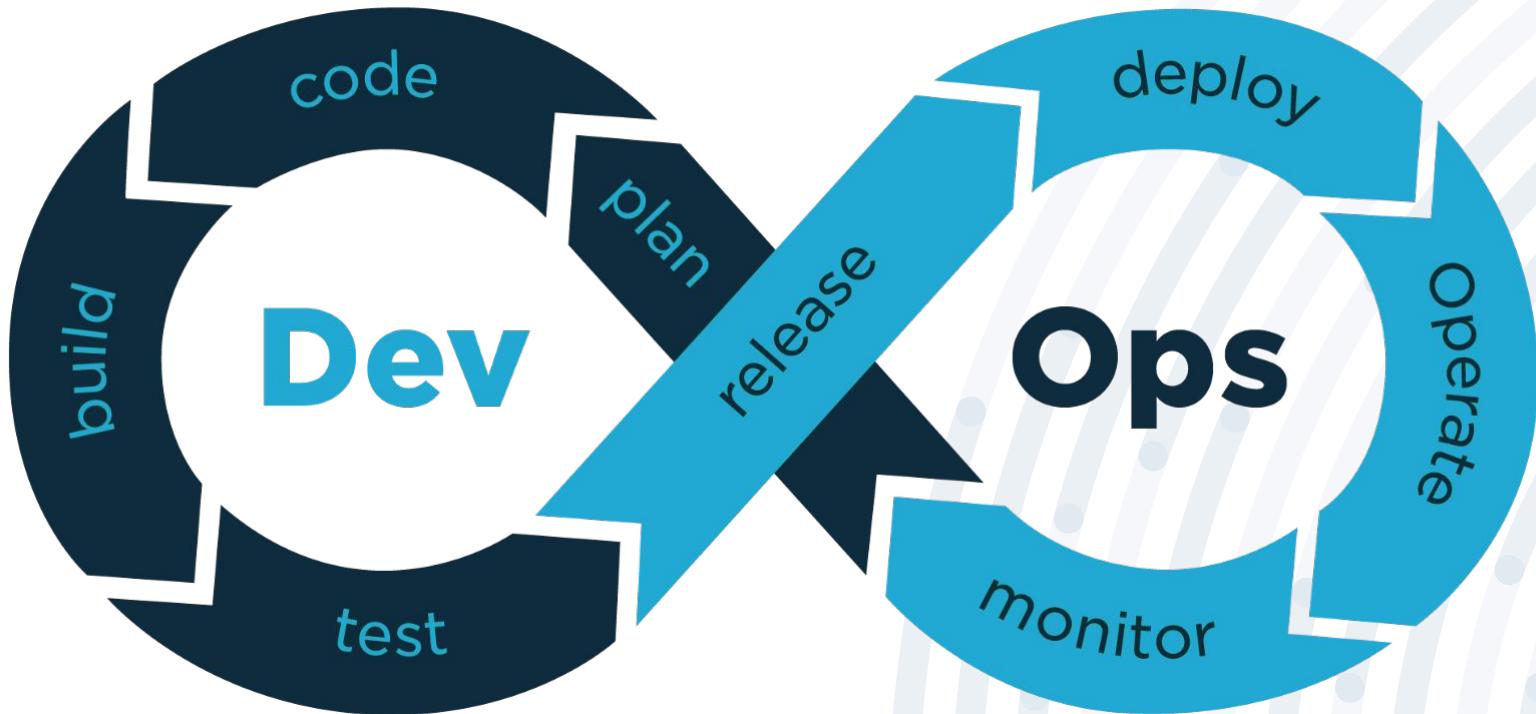
Ce n'est pas un processus, ni une technologie, ni une norme.

Le DevOps est un mouvement visant à rapprocher les métiers des développements & des opérations dans une même équipe.

Le DevOps est une extension naturelle des méthodes agiles (SCRUM, KABAN)

# 1.2 Pipeline DevOps

Les 8 étapes du pipeline DevOps:



# 1.2 Pipeline DevOps



- 1. Plan

Toutes les étapes avant le démarrage des développements  
Etape de planification:

- Cadrage Fonctionnel & Technique
- Gestion du backlog
- Préparation des US du sprint

- 2. Code

Phase de conception logicielles & du développement de la solution métier.



# 1.2 Pipeline DevOps

- 3. Build



Phase de construction du livrable à partir du code.

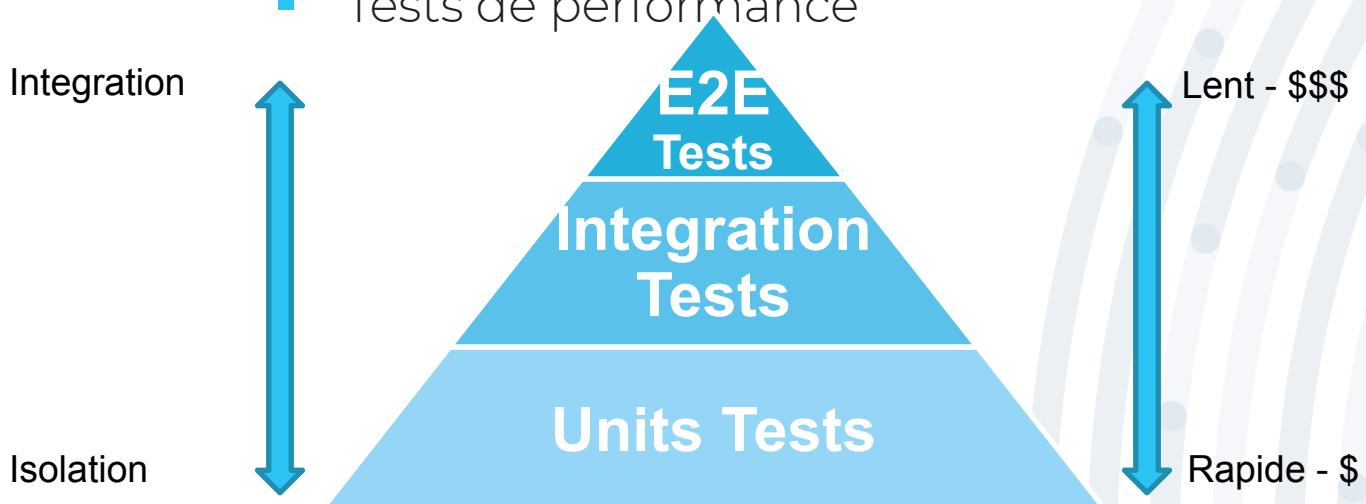
- Jar/war
- Tar
- Bin
- Image docker
- Souche OS
- RPM

# 1.2 Pipeline DevOps

- 4. Tests

Phase de tests, qui vise à assurer le bon fonctionnement des développements et des exigences métier.

- Tests fonctionnels, TNR, UAT
- Tests de performance



# 1.2 Pipeline DevOps

- 5. Release

Gestion des versions applicative:

- SNAPSHOTs: les versions de travail
- RELEASES: les releases regroupe un ensembles de features/fixes.

Sémantique version: **Majeur.Mineur.Correctif**, ex: 1.2.1

- **Majeur**: Evolution non rétrocompatibles
- **Mineur**: Evolution rétrocompatibles
- **Correctif**: patch d'anomalie

- 6. Deploy

Phase de livraison en production.

- FDR, Feuille de route (étapes de livraison, acteurs, durée, planing)
- Procédure de rollback

# 1.2 Pipeline DevOps

- 7. Operate (Exploitation)

Phase qui permet d'assurer le bon fonctionnement de l'application en production. Cette opération est en général à la main d'une autre équipe dédiée (help desk), qui s'assure par des astreintes par exemple d'un fonctionnement 24/7.

- 8. Monitor



Phase de surveillance de l'application et des process. Il s'agit donc en particulier du monitoring de la production (logs, failles de sécurité, performance) mais aussi du recueil des avis clients, du recueil de statistiques projets (time to market, bottleneck...).

# 1.2 Objectifs DevOps

- Continuous Integration (CI)
  - Branche feature/fix -> branche partagée
  - Construction automatique de l'application
  - Tests de l'application
  - Merge de la branche si succès
- Continuous Testing
- Continuous Delivery
- Continuous Deployment (CD)
  - Déploiement de l'application sur l'environnement de production



# 1.3 DevOps & Agile

## SAFe 5 for Lean Enterprises

Select SAFe configuration

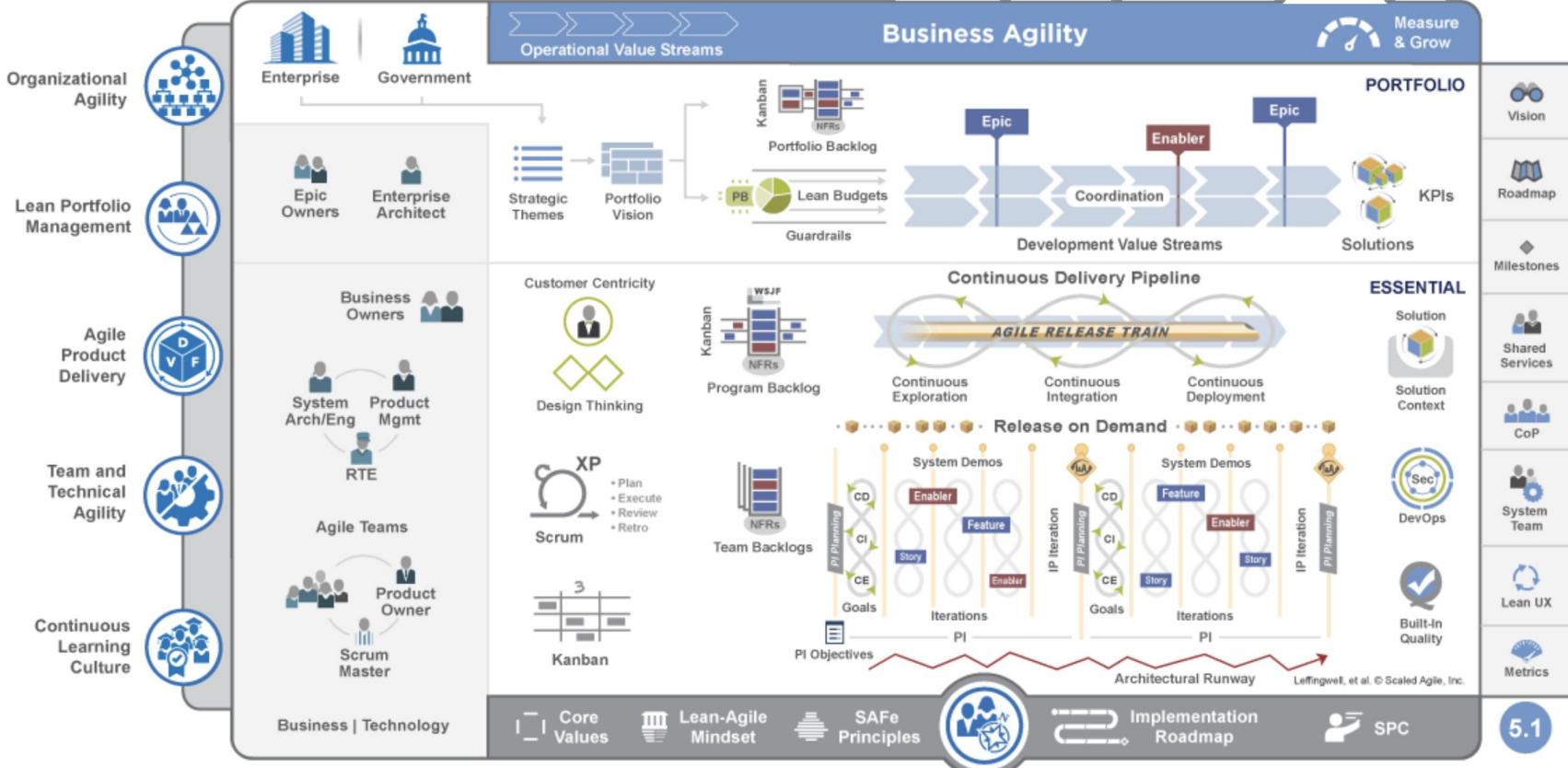
Overview

Essential

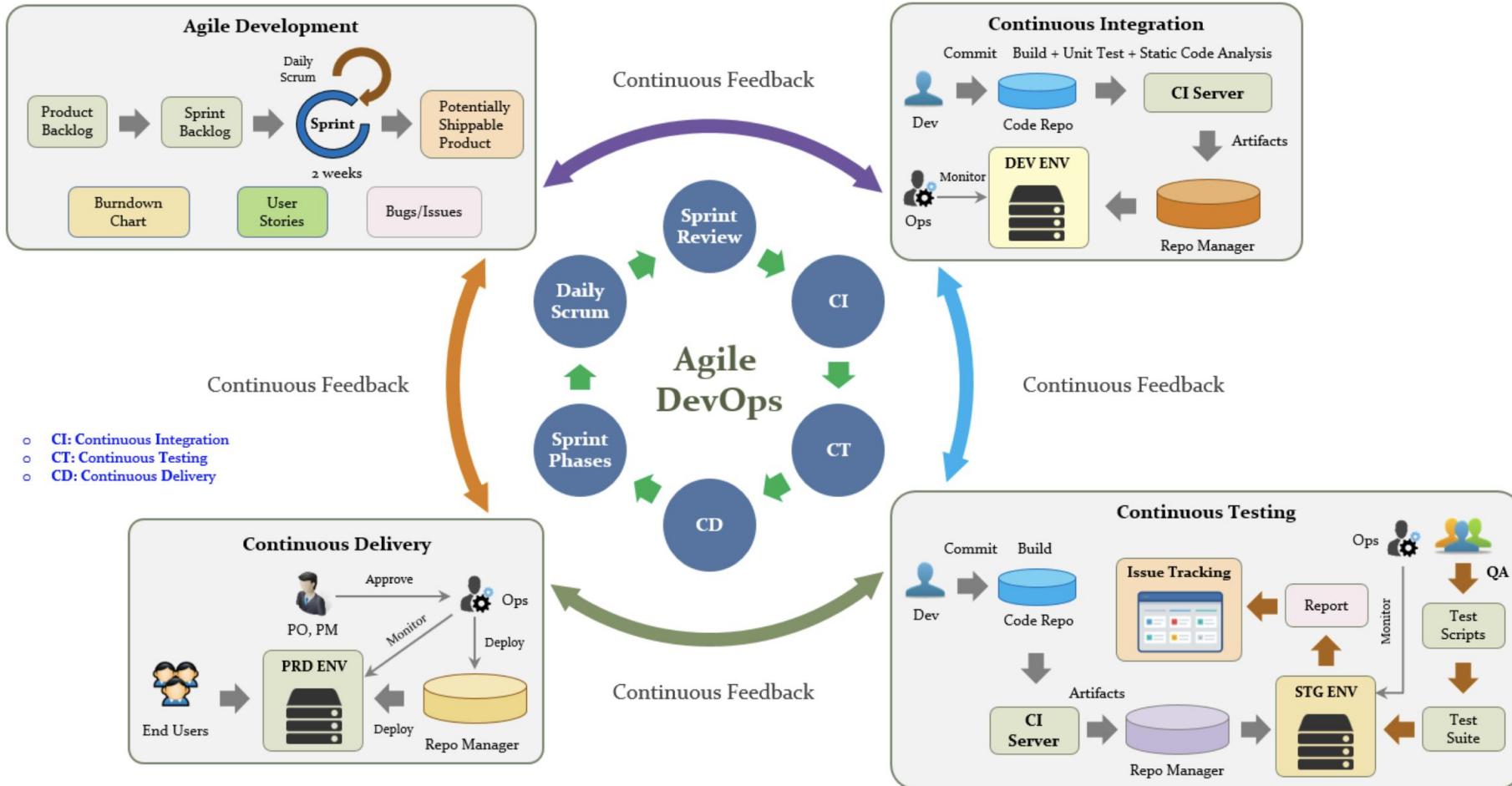
Large Solution

Portfolio

Full



# 1.3 DevOps & Scrum



## 1.4 Intérêts Approche DevOps

- DevOps = implication de l'ensemble des acteurs de la chaîne



## 1.4 Intérêts Approche DevOps

- Automatisation de TOUT
- Meilleur qualité du produit
- Déploiement plus fiable & plus rapide
- Résolution d'incidents plus rapide
- Réduction du coût(\*) et du time to market

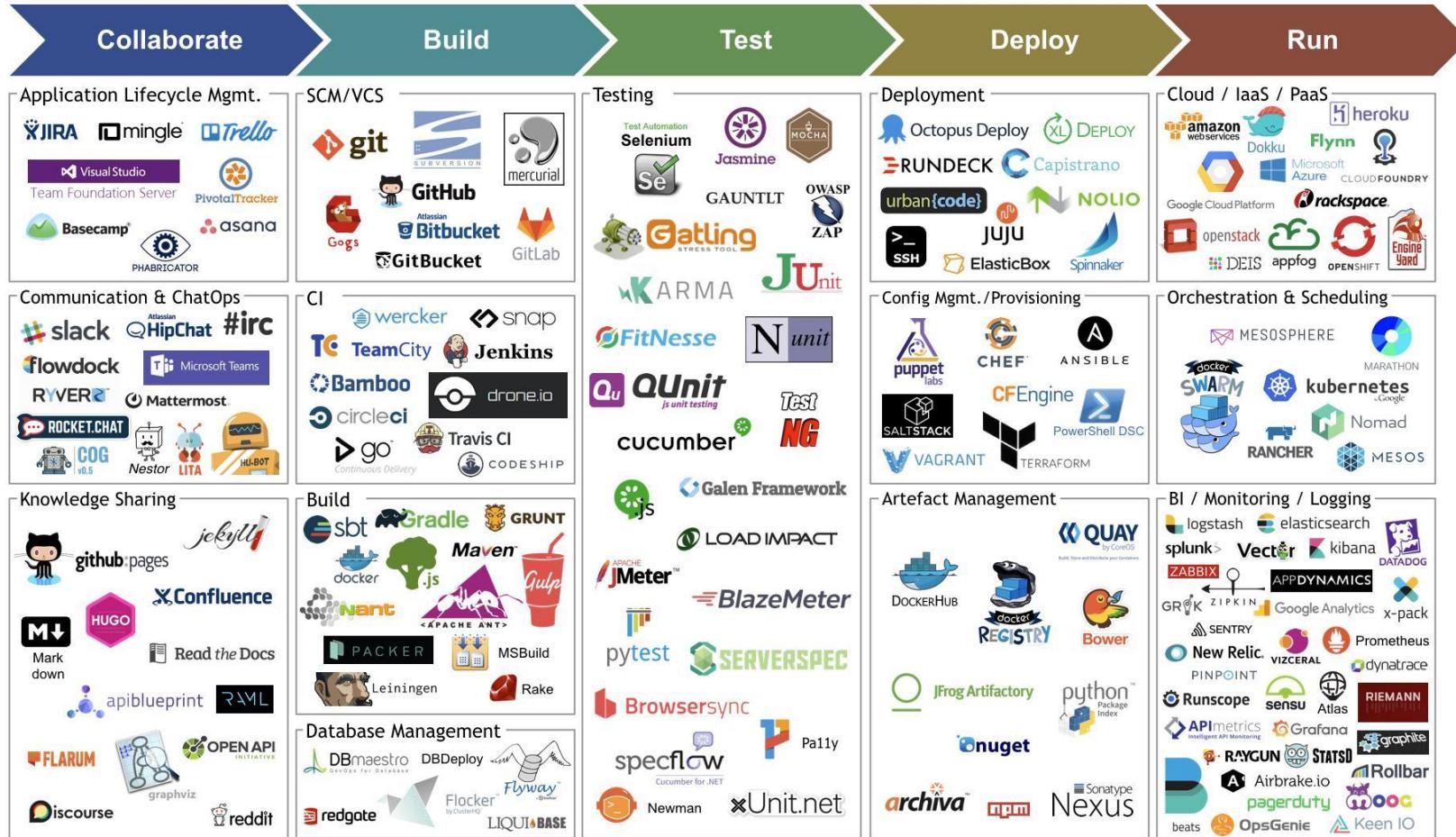
# Q&A

*Any Questions ?*



## 2. Les outils DevOps

# 2. Outils DevOps...



# 2.1 GIT



## 2.1 Git et GitFlow

Un outil de VCS : indispensable et pierre angulaire !

- Travail collaboratif
- Gestion de version
- Travail en PR/MR
- Hooks intégrables aux autres outils devops

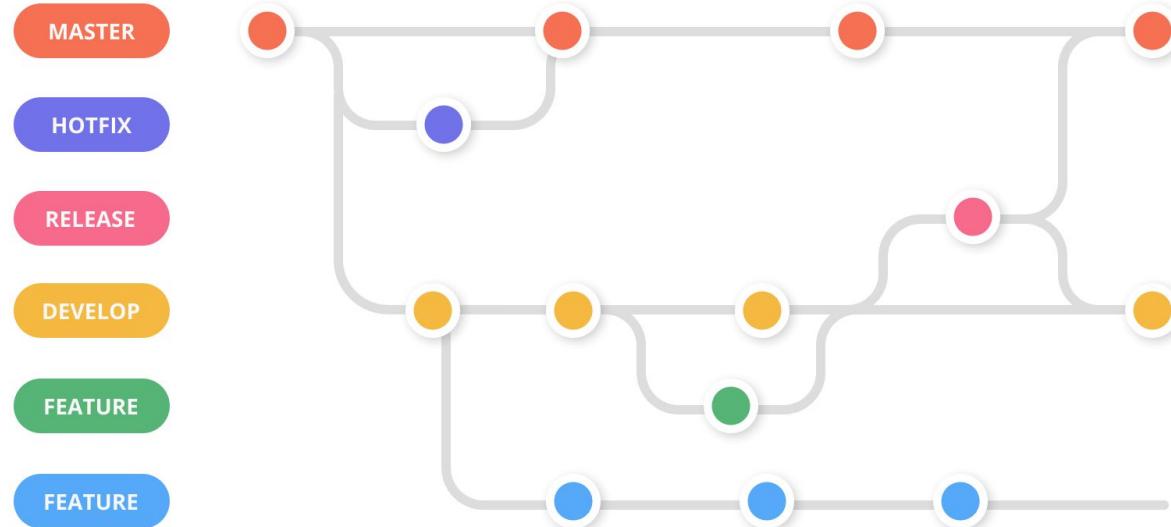
## 2.1 Git et GitFlow

### Git flow

- master : la branche de production
- develop : la branche d'intégration
- features : les branches pour les nouvelles features
- releases : les branches des nouvelles versions du produit
- hotfix : les fix sur la branche de production
- fix : les fix sur les autres branches

# 2.1 Git et GitFlow

## Git flow



## 2.1 Git et GitFlow

Practice



## 2.2 Jenkins



## 2.2 Jenkins

- Jenkins est un serveur d'automatisation CI/CD
  - Solution éprouvée sortie en 2011
  - Open-source communauté active
  - + 1700 plugins



## 2.2 Jenkins Pipeline

- Un pipeline est une suite de tâches qui permettent d'industrialiser et d'automatiser les processus CI, CT, CD
- Langage Groovy (interpreter Groovy CPS)
- 2 types de pipeline:
  - Declarative
  - Scripted

## 2.2 Jenkins Pipeline Notions

Les scripts Pipeline apportent les grandes notions suivantes :

- *agent/node* représente un environnement pouvant exécuter un pipeline (une machine esclave)
- *tools* permet de définir les versions d'outils utilisés par le script
- *stage* représente un ensemble d'étapes de votre processus
- *step* représente l'action à réaliser, comme `git checkout`, ).

## 2.2 Jenkins Pipeline Scripted vs Declarative

```
pipeline {  
    agent any  
    stages {  
        stage("Build") {  
            steps {  
                echo "Some code compilation here..."  
            }  
        }  
        stage("Test") {  
            steps {  
                echo "Some tests execution here..."  
            }  
        }  
    }  
}
```

```
node {  
    stage("Build") {  
        echo "Some code compilation here..."  
    }  
    stage("Test") {  
        echo "Some tests execution here..."  
    }  
}
```

## 2.2 Jenkins Pipeline Declarative vs Scripted

```
pipeline {  
    agent any  
    options {  
        timestamps()  
        ansiColor("xterm")  
        timeout(time: 1, unit: "MINUTES")  
    }  
    stages {  
        stage("Build") {  
            steps {  
                echo "Some code compilation here..."  
            }  
        }  
    }  
}
```

```
node {  
    timestamps {  
        ansiColor("xterm") {  
            stage("Build") {  
                timeout(time: 1, unit: "MINUTES") {  
                    echo "Build App..."  
                }  
            }  
        }  
    }  
}
```

## 2.2 Jenkins

TP 3 Jenkins :

- Getting Started job pipeline
- Création d'un jobs de CI

# 2.3 lac



## 2.3 IaC

IaC : Infrastructure as Code

- Décrire son infrastructure
- Gestion et approvisionnement de l'infrastructure
- Automatisation
- Réduction des disparités entre environnement de production et de développement
- Cloud
- Versioning



## 2.3 IaC Terraform



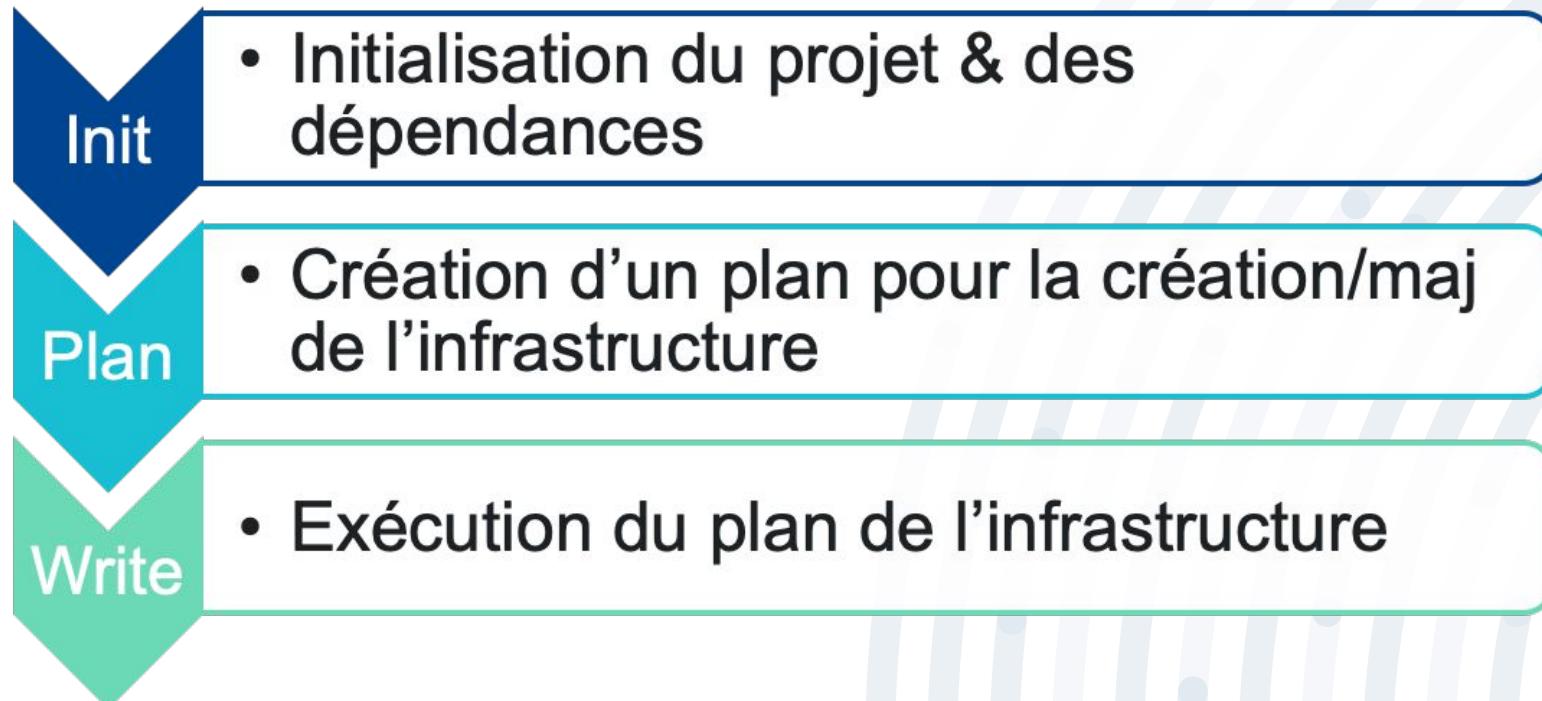
Un outil de l'IaC : Terraform

- CLI pour utiliser des ressources des providers de cloud (AWS, Openstack, Azure...)
- Déclaration de l'ensemble des ressources qui seront utilisées (Langage déclaratif (HCL/JSON)).

## 2.3 IaC Terraform



Terraform workflow



## 2.3 Terraform



Terraform providers:

- Pour déployer de l'infrastructure Terraform se base sur des cloud providers (aws, azure, kubernetes)

```
terraform {  
  
  required_providers {  
  
    aws = {  
  
      source  = "hashicorp/aws"  
  
      version = "~> 3.35"  
  
    }  
  
  }...  
}
```

## 2.3 Terraform



Terraform resource:

- Element le plus important de terraform
- Chaque ressource représente un objet d'infrastructure

```
resource "aws_vpc" "main" {  
    cidr_block = var.base_cidr_block  
}  
  
<BLOCK TYPE> "<BLOCK LABEL>" "<BLOCK LABEL>" {  
    # Block body  
    <IDENTIFIER> = <EXPRESSION> # Argument  
}
```

## 2.3 Terraform



Terraform datasource:

- Permet de récupérer des informations en provenances du Cloud provider, de l'infrastructure.
- Ces informations peuvent ensuite être réutiliser dans les ressources.

```
data "aws_ami" "os_image" {  
    most_recent = true  
    owners = ["self"]  
}
```

## 2.3 Terraform



Terraform variables:

```
variable "image_id" {  
    type        = string  
    description = "The id of the machine image (AMI)"  
  
    validation {  
  
        condition      = length(var.image_id) > 4 && substr(var.image_id,  
0, 4) == "ami-"  
  
        error_message = "The image_id value must be a valid AMI id,  
starting with \"ami-\"."  
  
    }  
}
```

## 2.3 Terraform



Terraform project structure:

```
myIac/
  module/
    myModule1/
      variables.tf    --> Variables du module
      main.tf         --> Main module
  vars/
    dev/
      variables.tf  --> Variables env dev
    prod/
      variables.tf  --> Variables env prod
  provider.tf      --> Providers config
  Backend.tf       --> Backend config
  app.tf           --> Instance config
  app-sg.tf        --> Instance SG
```

## 2.3 IaC - AWS

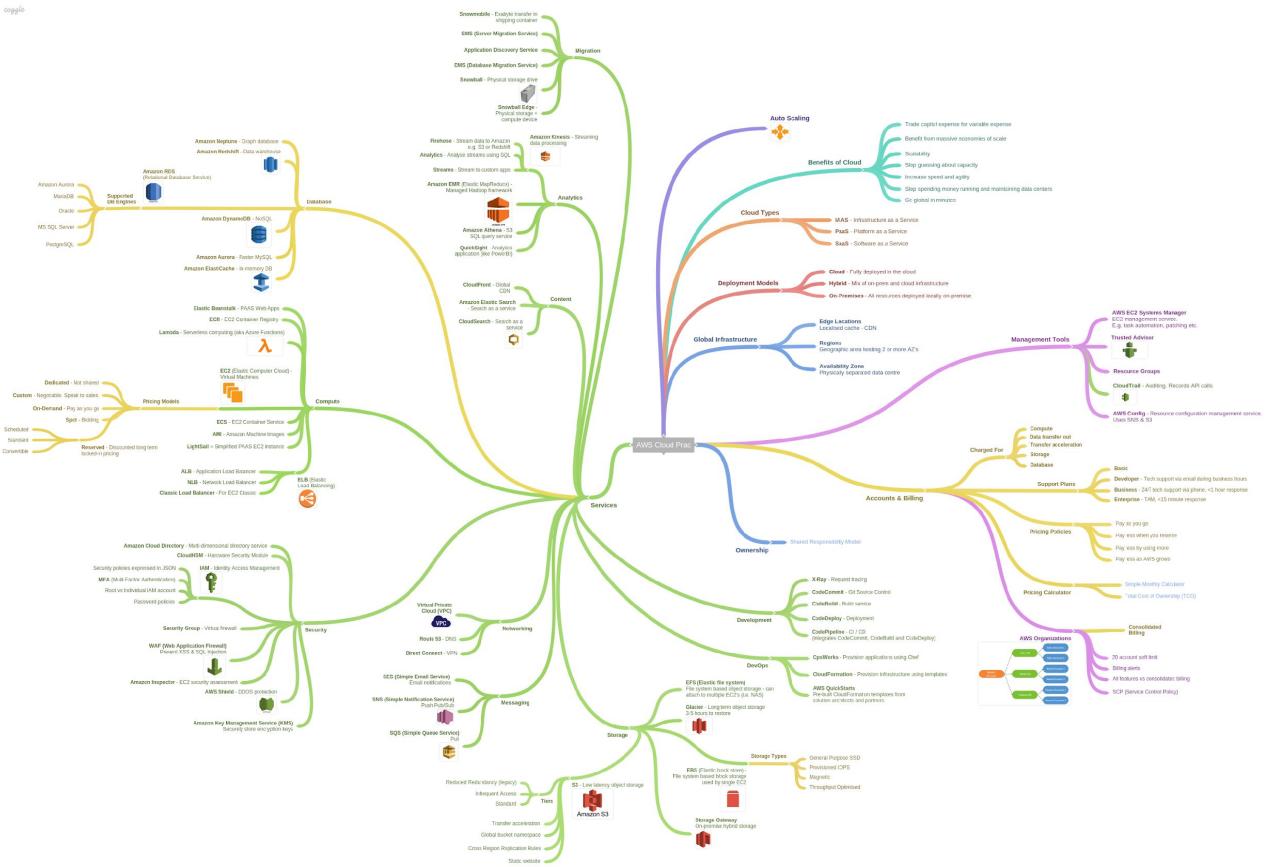
AWS:

- Leader Cloud (IaaS, PaaS, SaaS, CaaS, DaaS, etc...)



- Cloud VS Legacy
  - Pas d'investissement
  - Pay as you go
  - Horizontal/Vertical scaling / AutoScaling

## 2.3 IaC - AWS Services



<https://www.shanebart.com/wp-content/uploads/2018/06/Coggle-blog.png>

# Q&A

*Any Questions ?*

## 2.3 IaC

TP 2 Terraform :

- Getting Started



## 2.3 IaC

TP 3 Terraform :

- Mettre en place l'infrastructure qui hébergera notre application

# 2.4 CaC



ANSIBLE

## 2.4 CaC

CaC : Configuration as Code

- Configuration du système
- Configuration de l'application
- Automatisation
- Réduction des disparités entre environnement de production et de développement
- Versioning 
- Différent de l'IaC (Provisioning vs Configuration)

## 2.4 CaC

Un outil du CaC : Ansible

- Agentless
- Idempotence
- Langage déclaratif : Yaml

## 2.4 CaC

### TP 4 : Ansible

- Comprendre et créer son premier playbook
- Module, tâche, rôle, playbook

## 2.4 CaC

TP 5 : Ansible

- Configurer et déployer notre application

## 2.4 Pipeline de déploiement

Gérer le cycle DevOps et CI/CD

- Builder son application
- Réaliser ses tests
- Release de l'application
- Déploiement de l'infrastructure
- Déploiement de la configuration et de l'application

## 2.4 Pipeline de déploiement

Un outil de pipeline : Jenkins

# Q&A

*Any Questions ?*