# Continuous Integration (CI)

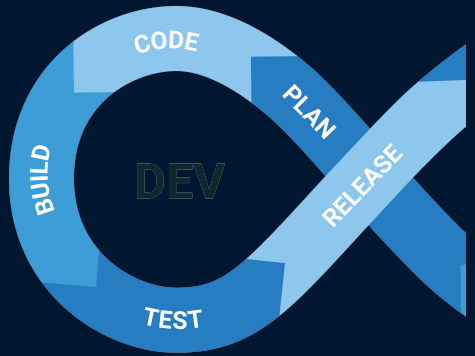
Définition
(9 slides)

# Continuous Integration (CI)

- **1. Plan *(ex : Jira)***
  *On crée ou améliore l'application*
- **2. Code (*ex : VSCode*)**
  *On code les fonctionnalités prévues*
- **3. Build (*ex : python –m build / npm*)**
  *On construit l'application*

**4. Test (*ex : PyTest*)**
*On teste que l'application fonctionne comme prévu*

**5. Release (*ex : GitHub Actions / GitLab CI*)**
*On prépare une version stable de l'application à mettre en ligne*



# Continuous Integration (CI)

- **3. Build (*ex : python –m build / npm*)**
  *On construit l'application*
  - Formatage, linting, sécurité, …
    - Lint (ruff, flake8)
    - Formatage (black --check)
    - Type check (mypy)
    - Sécurité (bandit, safety)

**4. Test (*ex : PyTest*)**
*On teste que l'application fonctionne comme prévu*

- Pytest (check doc, assert)

**5. Release (*ex : GitHub Actions / GitLab CI*)**
*On prépare une version stable de l'application à mettre en ligne*

# Continuous Integration (CI)

- Formatage : Uniformiser le code

```
def hello():print("Hello")
x=1
```

```
def hello():
    print("Hello")

x = 1
```

- Lint : Vérifier la qualité et erreurs

```
tests.py:2:5:
F841 local variable 'x'
is assigned to but never used
```

***PyTest*** : vérifier le code (mot clef : assert) automatiquement, rapidement à chaque commit dans la pipeline

```
print(add(2,3))
print(add(0,0))
print(add(-1, 10))
#----------------------------
def test_add():
    assert add(2,3) == 5

def test_add_zero():
    assert add(0,0) == 0

def test_add_negative():
    assert add(-1, 10) == 9
```

# Exercice :

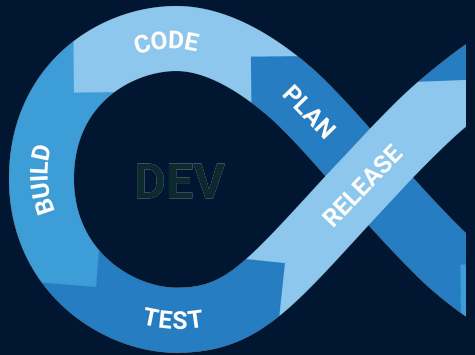
- Reprendre votre CatFacts
- Faire tourner un PyTest dessus (s'appuyer de la doc)
  - Pytest se lance depuis la racine du projet
- Formater et faire du Lint sur votre code
- Rappel : Un module s'installe avec pip
- Faire de même pour votre PokeAPI
- Push sur votre Github

# Qu’est ce qu’une pipeline CI ?

- **1. Plan *(ex : Jira)***
  *On crée ou améliore l’application*
- **2. Code (*ex : VSCode*)**
  *On code les fonctionnalités prévues*
- **3. Build (*ex : python –m build / npm*)**
  *On construit l’application*

**4. Test (*ex : PyTest*)**
*On teste que l’application fonctionne comme prévu*

**5. Release (*ex : GitHub Actions / GitLab CI*)**
*On prépare une version stable de l’application à mettre en ligne*



# Que peut-on faire dans une pipeline CI ?

- Formatage : Uniformiser le code

```
def hello():print("Hello")
x=1
```

```
def hello():
    print("Hello")

x = 1
```

- Lint : Vérifier la qualité et erreurs

```
tests.py:2:5:
F841 local variable 'x'
is assigned to but never used
```

***PyTest*** : vérifier le code (mot clef : assert) automatiquement, rapidement à chaque commit dans la pipeline

```
print(add(2,3))
print(add(0,0))
print(add(-1, 10))
#----------------------------
def test_add():
    assert add(2,3) == 5

def test_add_zero():
    assert add(0,0) == 0

def test_add_negative():
    assert add(-1, 10) == 9
```

# Que peut-on faire dans une pipeline CI ?

- Exemple d’installation : pip install black ruff requests
- Secu : SAST (Semgrep), SCA (Trivy)
- **5. Release (*ex : GitHub Actions / GitLab CI*)**
  *On prépare une version stable de l’application à mettre en ligne*

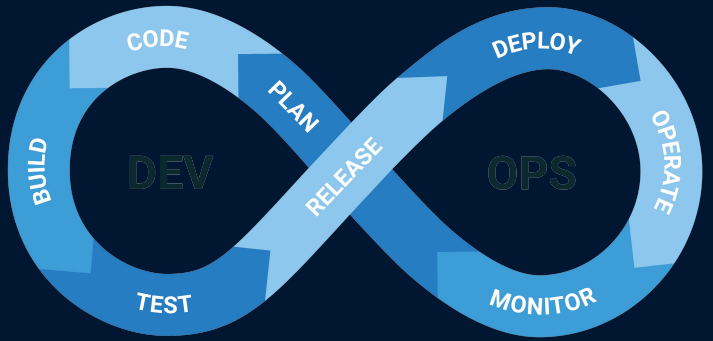
Exemple d’installation : pip install black ruff requests
Secu : SAST (Semgrep), SCA (Trivy)
**5. Release (*ex : GitHub Actions / GitLab CI*)**
*On prépare une version stable de l’application à mettre en ligne*

# Exercice 4 : Tests de notre code dans Github Actions

- CI simple : formatage, tests, lint

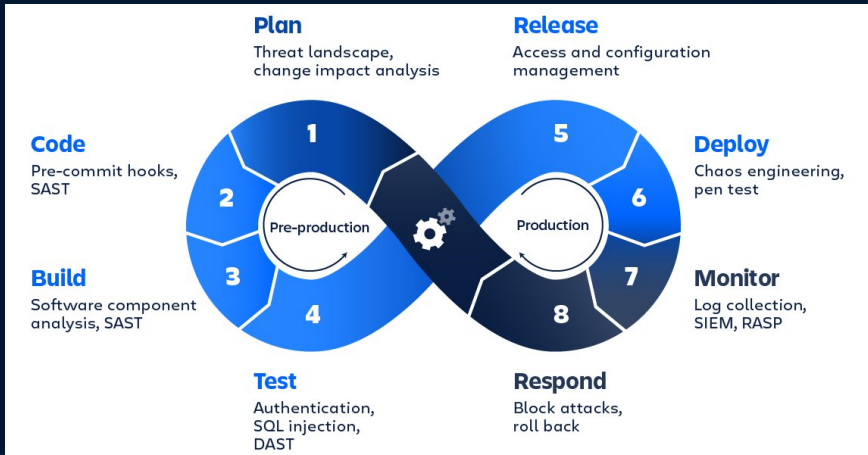
# Que peut-on faire dans une pipeline CI ?

- DevOps rapproche Dev et Ops.
- Objectif : travailler ensemble, automatiser, livrer plus vite.
- **Idée clé : intégration (de code) continue (CI) et déploiement continu (CD).**



# Alors, qu’est ce que le DevSecOps

- Avec DevOps, on va vite… parfois trop vite pour la sécurité
- Avant, la sécurité intervenait seulement à la fin → trop tard.
- DevSecOps insère la sécurité dès le début, et à chaque étape.
- Security by Design



# Alors, qu’est ce que le DevSecOps

- **1. Plan : Threat Modeling Tool**
  Réfléchir aux risques de sécurité dès la conception

- **2. Code SAST : Semgrep + Secure Coding**
  Analyse le code pour détecter des failles

- **3. Build : Safety ou Pip-audit**
  Vérifie que les bibliothèques utilisées
  ne contiennent pas de vulnérabilités connues

**4. Test : OWASP ZAP (DAST)**
Simule des attaques sur l'application pour trouver des failles

**5. Release : Image/Artifact Scan : Trivy**
Analyse les images Docker ou artefacts avant leur mise en ligne.

# Ce qu’il faut retenir

- DevOps = Communication Dev et Ops pour + d’efficacité

- DevSecOps = efficacité en sécurité

- **Le but n’est pas la perfection,**
  **mais des petites améliorations continues**

# « Le but n’est pas la perfection, mais des petites améliorations continues. »