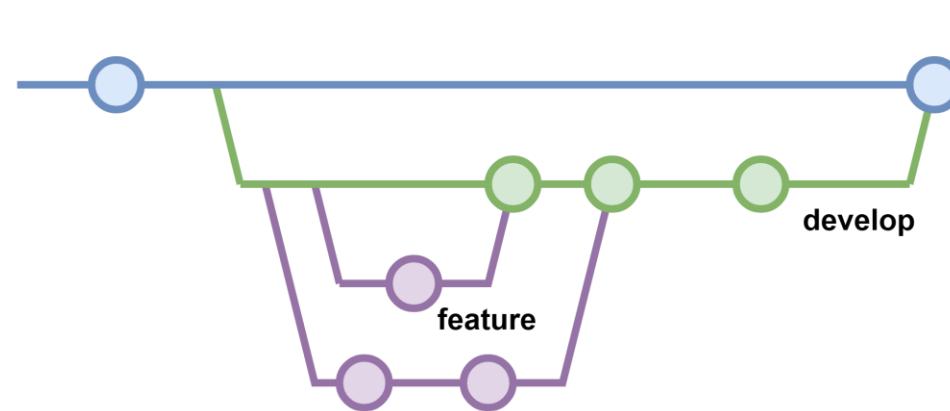


# Projet Tutoré & DevOps

Hiba Najjar

S7, GInf4

# Workflows de développement

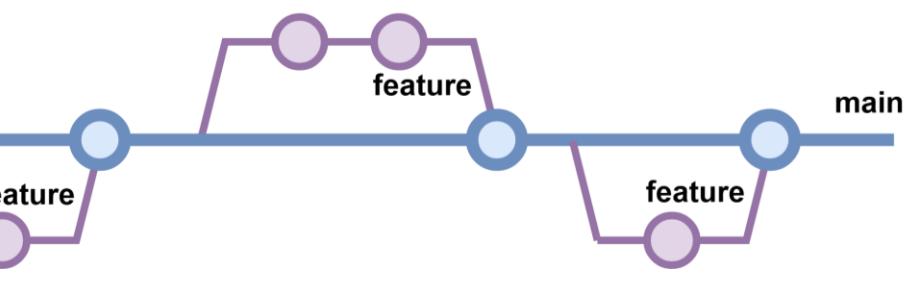


Feature-based  
(Gitflow)

main

develop

feature



Trunk-based

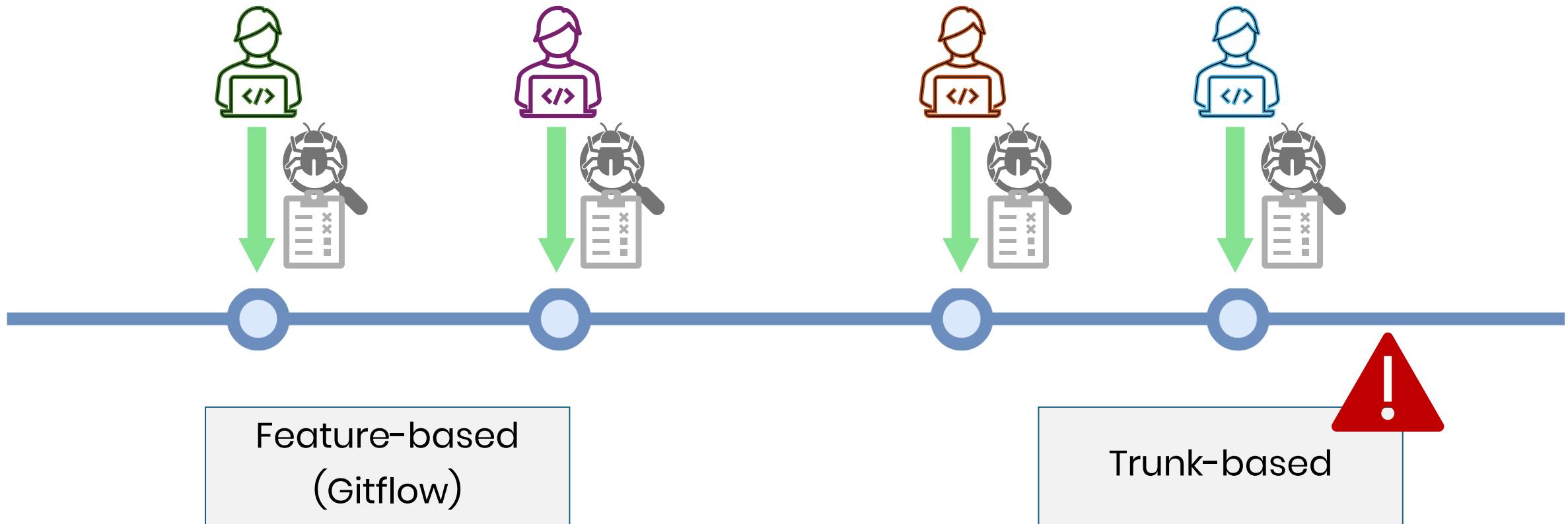
main

feature

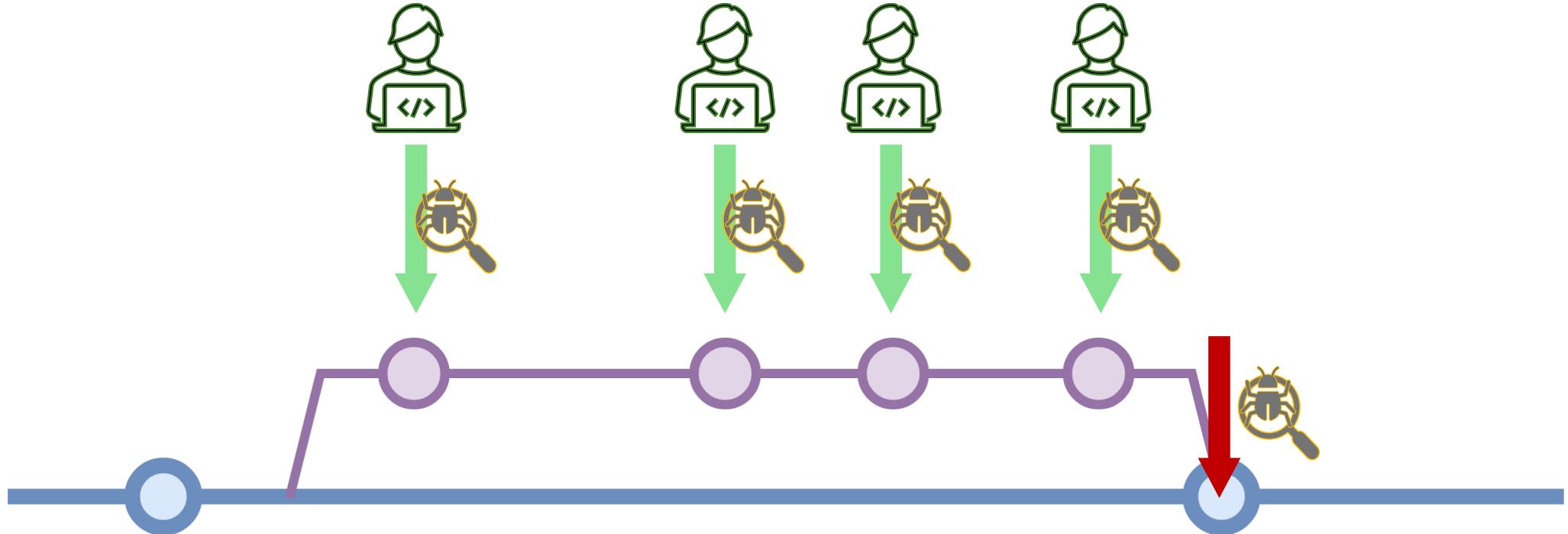
feature

main

# Tests automatisés

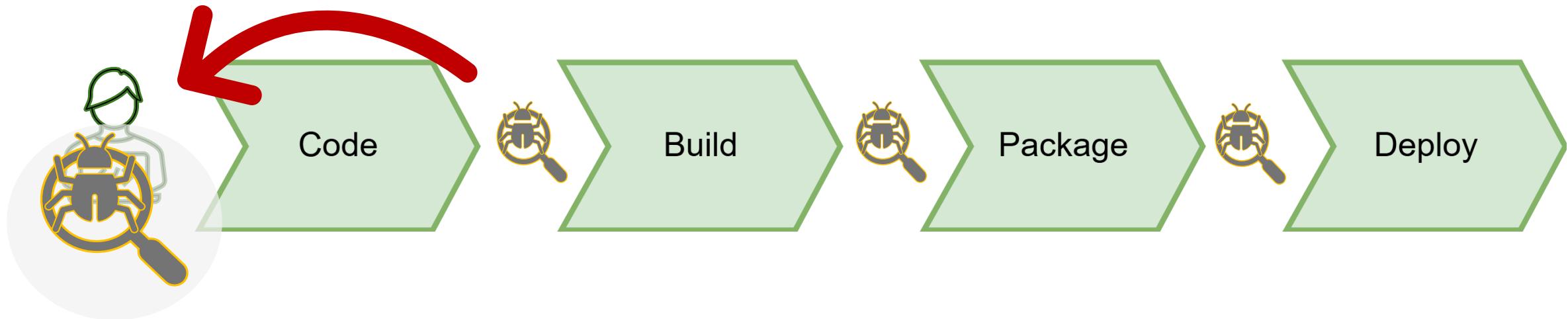


# Quand est ce qu'on teste?



- ✓ Intégration Continue (ci)
- ✓ Permet aux débutants d'apprendre les bonnes pratiques du code

# Quand est ce qu'on teste?



- ✓ Automatisation élevée
- ✓ Étapes manuelles réduites au maximum
- ✓ Shifting Tests Left, dans l'IDE local

# Types de tests automatisés

## Tests de qualité de code

- Analyse statique
- Vérifie si les bonnes pratiques de codage sont respectées
- Assure un code propre et maintenable



## Tests unitaires

- Tester les fonctions, les méthodes, etc
- Valider que chaque unité de code fonctionne comme prévu



## Tests de sécurité

- Chercher les vulnérabilités et failles de sécurité
- Simuler des attaques potentielles



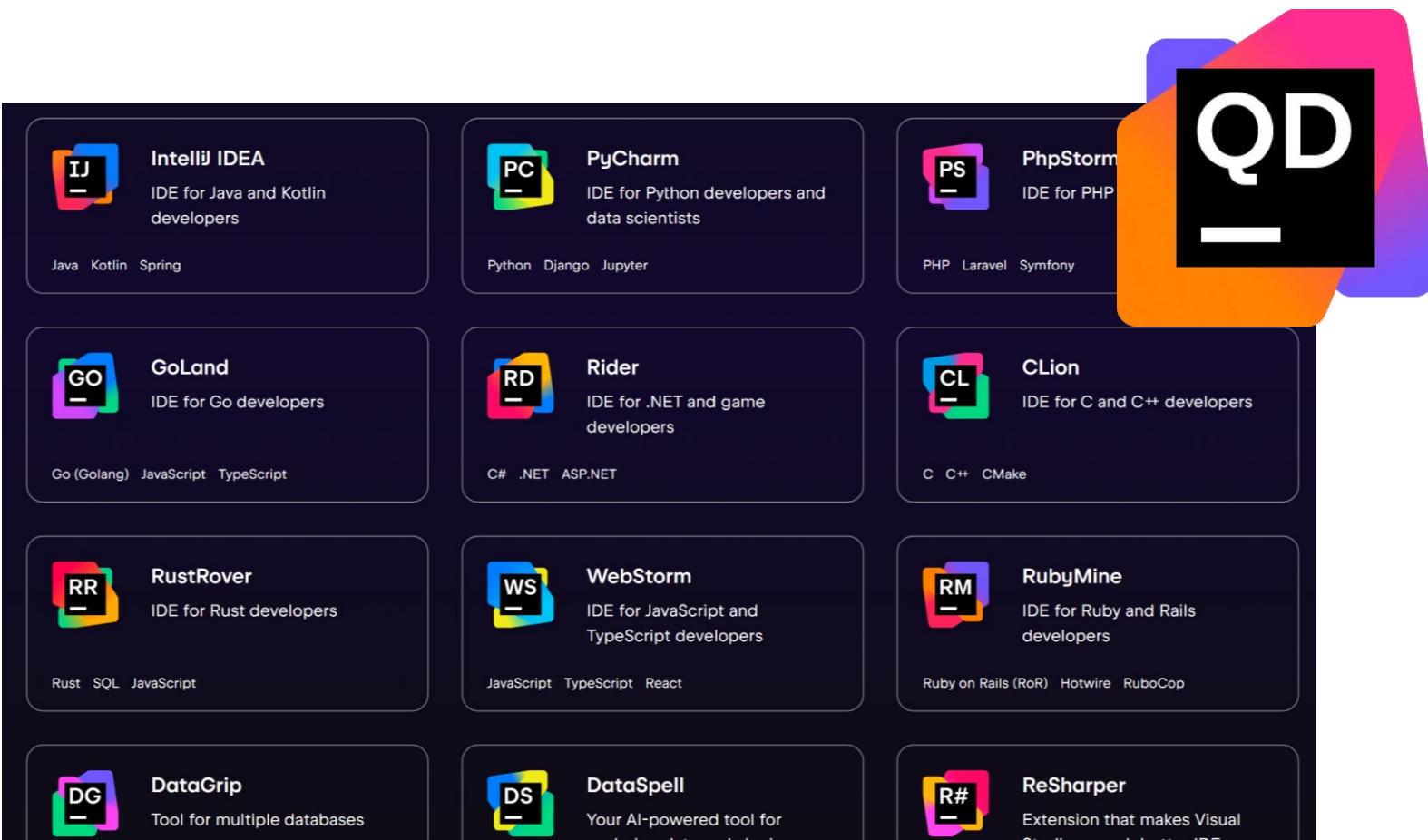
## Tests d'intégration

- Tester le bon fonctionnement de l'assemblage de plusieurs unités de code
- E.g. vérifier l'interaction avec la base de données ou les API externes



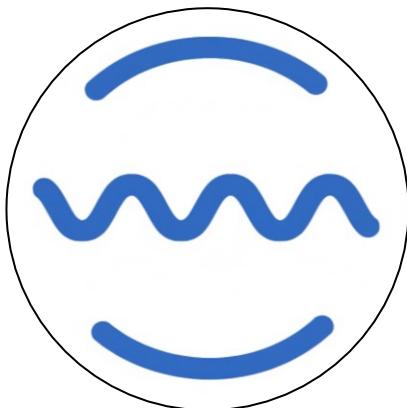
# Tests de qualité de code

- JetBrains IDEs >> **Qodana**



# Tests de qualité de code

- JetBrains IDEs >> **Qodana**
- VSCode >> plusieurs options



**SonarQube**  
Many languages



**ESLint**  
JavaScript, TypeScript

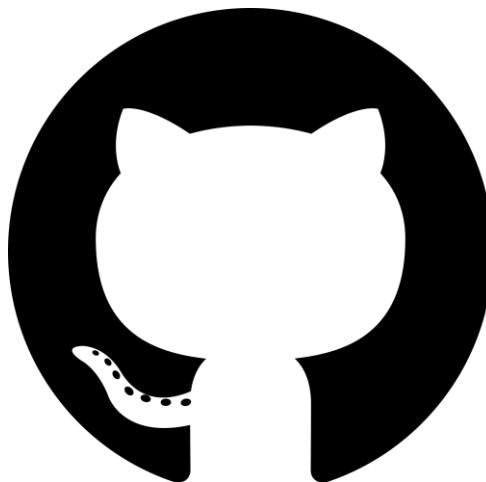


**Stylelint**  
CSS and related

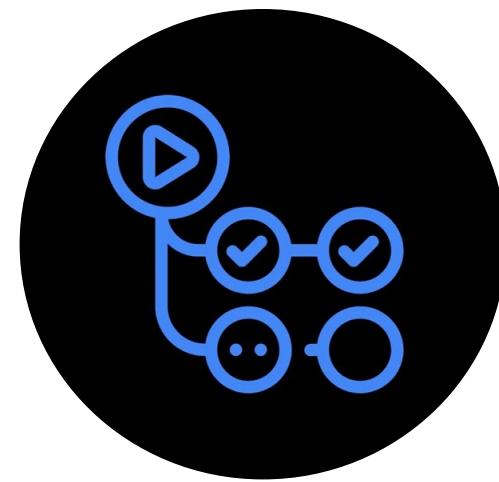
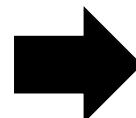
# Tests de qualité de code

- JetBrains IDEs >> **Qodana**
- VSCode >> plusieurs options
- GitHub >> **GitHub Actions**

1. Intégration Native: code et pipeline CI au même endroit  
2. Simplicité de configuration par les développeurs  
3. Environnements gérés, (pas besoin d'installer manuellement Docker, Java, etc, sur les serveurs)

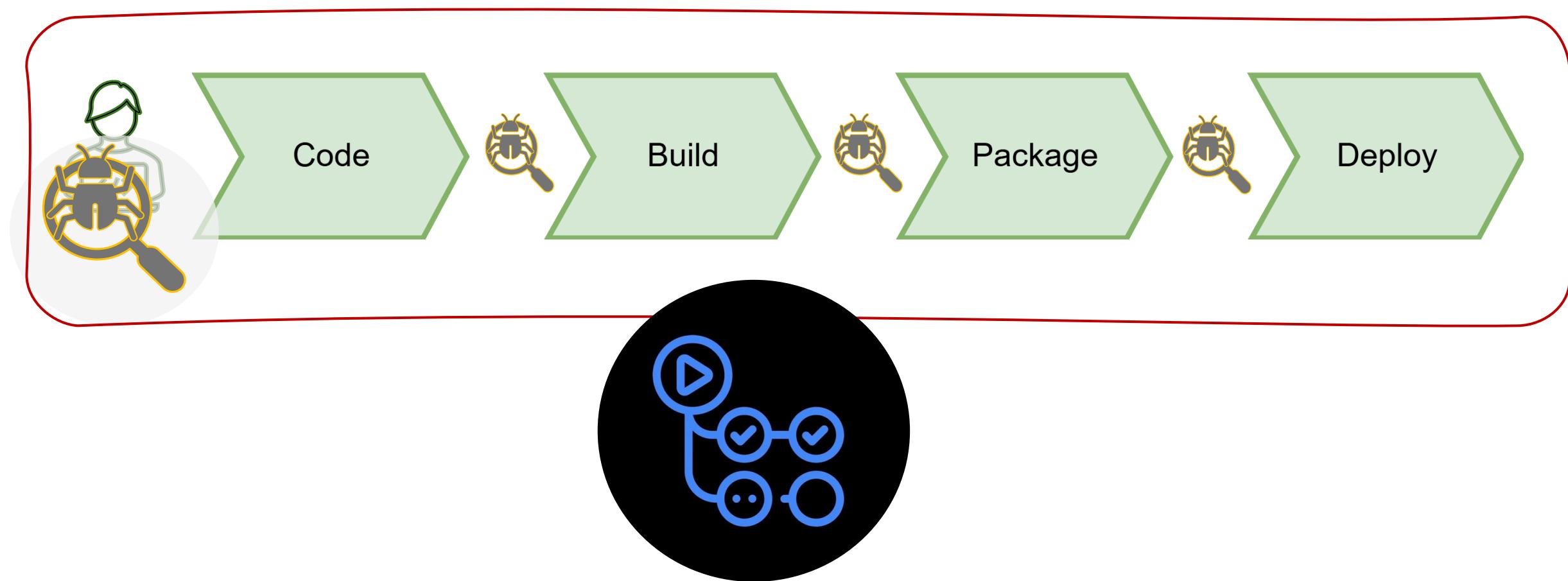


**GitHub**

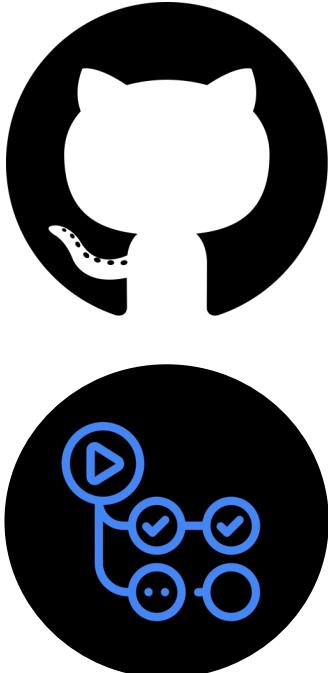


**GitHub Actions**

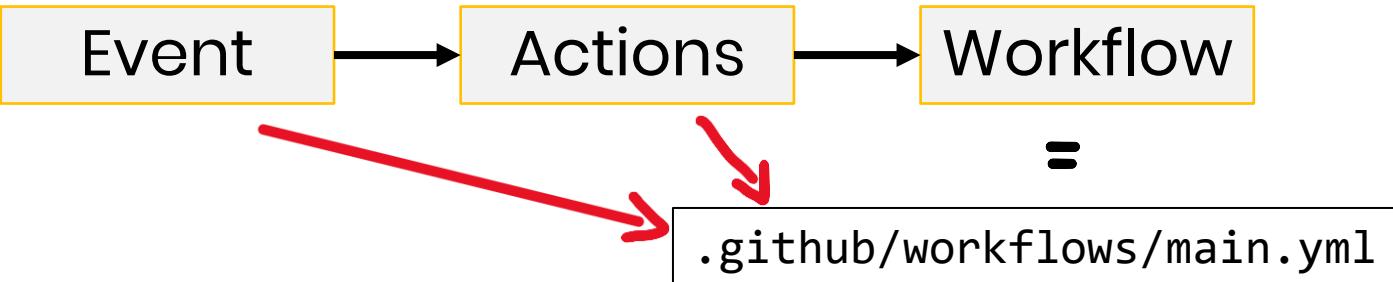
# Pipeline CI sur GitHub Actions

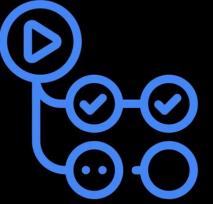


# Pipeline CI sur GitHub Actions



Exemple: PyTorch Repository  
**» <https://github.com/pytorch/pytorch>**





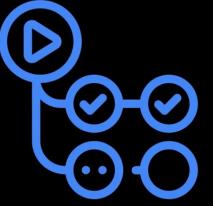
# GitHub Actions: Events

Event/événement = activité spécifique qui déclenche l'exécution d'un workflow GitHub Actions.

Exemples:

- Pousser du code (push)

```
yaml
On:
  push:
    branches: [main]
```



# GitHub Actions: Events

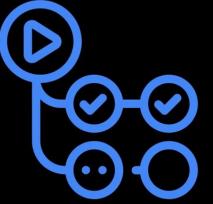
Event/événement = activité spécifique qui déclenche l'exécution d'un workflow GitHub Actions.

Exemples:

- Pousser du code (push)

```
yaml
```

```
On:  
  push:  
    branches:  
      - main  
      - develop  
    paths:  
      - 'src/**'  
      - package.json
```



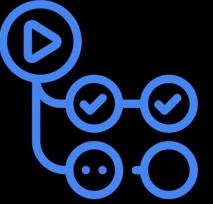
# GitHub Actions: Events

Event/événement = activité spécifique qui déclenche l'exécution d'un workflow GitHub Actions.

Exemples:

- Créer une pull request (pull\_request)

```
yaml
On:
  pull_request:
    types: [opened, reopened]
    branches: [main, develop]
    paths: '**.py'
```



# GitHub Actions: Events

Event/événement = activité spécifique qui déclenche l'exécution d'un workflow GitHub Actions.

Exemples:

- Planifier une exécution (schedule)

```
yaml
```

```
On:
```

```
  schedule:
```

```
    # syntaxe cron: minute heure jour mois jour-de-semaine
    - cron: '0 2 * * *'
    - cron: '30 0 * * 0'
```

Autres events:

<https://docs.github.com/en/actions/reference/workflows-and-actions/events-that-trigger-workflows>

[crontab.guru/](http://crontab.guru/)

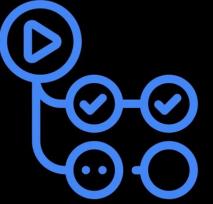


# GitHub Actions: Actions

Action = tâche individuelle, constitue une étape dans un workflow, exécute une opération spécifique. Réutilisable et partageable.

Deux types d'actions:

- **uses** : Appelle une Action préexistante (créeée par la communauté ou GitHub).
- **run** : Exécute une commande terminal classique.



# GitHub Actions: Actions

Action = tâche individuelle, constitue une étape dans un workflow, exécute une opération spécifique. Réutilisable et partageable.

Exemple:

**Uses: actions/checkout** >> récupère le code du dépôt

```
yaml
steps:
  - name: Checkout du code
    uses: actions/checkout@v6
    with:
      fetch-depth: 0          # Télécharge tout l'historique Git
      ref: 'feature/branch'  # Branche spécifique
```

[github.com/actions/checkout/](https://github.com/actions/checkout/)



# GitHub Actions: Actions

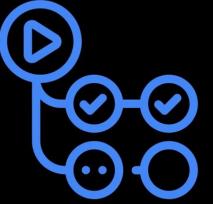
Action = tâche individuelle, constitue une étape dans un workflow, exécute une opération spécifique. Réutilisable et partageable.

Exemple:

**Uses: actions/setup-python** >> configurer l'environnement Python

```
yaml
steps:
  - name: Setup Python avec plusieurs versions
    uses: actions/setup-python@v6
    with:
      python-version: 3.11
      cache: 'pip'
      cache-dependency-path: requirements.txt
```

<https://github.com/actions/setup-python>



# GitHub Actions: Actions

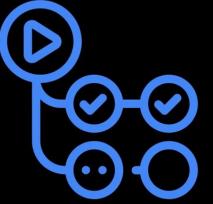
Action = tâche individuelle, constitue une étape dans un workflow, exécute une opération spécifique. Réutilisable et partageable.

Exemple:

**Uses: actions/setup-node** >> configurer Node.js avec npm

```
yaml
steps:
  - name: Setup Python avec plusieurs versions
    uses: actions/setup-node@v6
    with:
      node-version: '20.x'
      cache: 'npm'
      cache-dependency-path: package-lock.json
```

<https://github.com/actions/setup-node>



# GitHub Actions: Actions

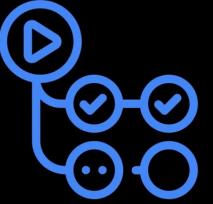
Action = tâche individuelle, constitue une étape dans un workflow, exécute une opération spécifique. Réutilisable et partageable.

Exemple:

**run** - Exécute une commande shell

```
yaml
steps:
  - name: Setup Python avec plusieurs versions
    uses: actions/setup-node@v6
    with:
      node-version: '20.19'

  - name: Installer les dépendances
    run: npm ci
```



# GitHub Actions: Actions

Action = tâche individuelle, constitue une étape dans un workflow, exécute une opération spécifique. Réutilisable et partageable.

Exemple:

**run** - Exécute une commande shell

```
yaml
steps:
  - name: Setup Python avec plusieurs versions
    uses: actions/setup-python@v6
    with:
      python-version: 3.11

  - name: Vérifier l'installation
    run: python --version
```



# GitHub Actions: Runners

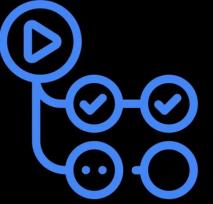
Runner = l'environnement de calcul sur lequel les étapes d'un job du workflow sont exécutés

- chaque job est exécuté dans son propre Runner séparé
- Peut utiliser les runners fournis et gérés par GitHub, qui sont détruits après l'exécution (= Hosted runners).
- Ou peut utiliser des machines physiques ou virtuelles gérées par l'utilisateur et connectées à son dépôt GitHub (=Self-Hosted Runners)

```
yaml
```

```
runs-on: ubuntu-latest
```

<https://github.com/actions/runner-images>



# Syntaxe

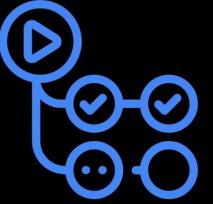
yaml

```
# Le nom (facultatif) du workflow, affiché dans l'interface GitHub
name: Workflow de Test

# Le déclencheur : le workflow s'exécute lors d'un 'push'
on:
  push:
    # On ne s'exécute que si le push est sur la branche 'main'
    branches: [ main ]

# La définition des jobs
jobs:
  # Le nom du job est 'build-and-test'
  build-and-test:
    # execution sur la dernière version d'Ubuntu (un runner Linux)
    runs-on: ubuntu-latest

    # La séquence des étapes du job
    steps:
```



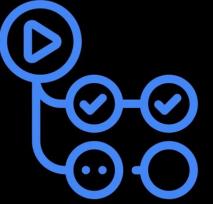
# Syntaxe

yaml

```
# La séquence des étapes du job
steps:
    # Étape 1 : Récupérer le code du dépôt (Action officielle GitHub)
    - name: Checkout du code
      uses: actions/checkout@v4

    # Étape 2 : Exécuter des commandes shell pour installer et tester
    - name: Installer et Lancer les Tests
      run: |
          echo "Installation des dépendances..."
          npm install
          echo "Lancement des tests..."
          npm test

    # Étape 3 : Afficher un message de succès
    - name: Afficher le Résultat
      run: echo "Le job s'est terminé avec succès!"
```



# Syntaxe

Syntaxe YAML	Bloc Principal	Description	Facultatif ?
name: .....	Nom du Workflow	Le nom qui apparaît dans l'interface GitHub.	✓
on: .....	Déclencheur	Définit l' <b>événement</b> (ou les événements) qui lance le workflow	
jobs:	Jobs (Tâches)	Contient un ou plusieurs <i>jobs</i> , qui s'exécutent en parallèle par défaut.	
<nom-du-job>:	ID du Job	Nom unique pour identifier un job spécifique en interne par GitHub Actions	
name: .....	Nom du Job	un titre clair et descriptif pour le poste dans la visualisation du flux de travail (peut utiliser les variables matrix)	✓
runs-on: .....	Environnement	Spécifie le <b>runner</b> (la machine virtuelle) sur lequel ce job s'exécute (ex: ubuntu-latest, windows-latest).	
steps:	Étapes	Une liste d'instructions exécutées <b>séquentiellement</b> dans le job.	
- name: .....	Nom de l'Étape	Nom descriptif pour l'étape (apparaît dans les logs).	✓
- uses: .....	Utilisation d'Action	Exécute une <b>Action</b> pré-existante (module réutilisable).	
- run: .....	Commandes Shell	Exécute des commandes de terminal directement sur le <i>runner</i> .	

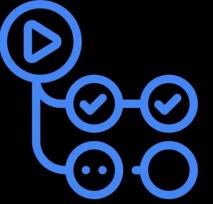


# GitHub Actions: Matrix

Matrix = définit un ensemble de variables pour un job donné, et GitHub Actions va générer un job distinct pour chaque combinaison possible de ces variables.

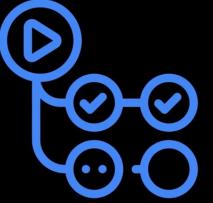
- outil idéal pour le test multi-configuration
- Définition sous la section **strategy** d'un job spécifique
- Utilisation des variables avec `$${{ matrix.nom_variable }}`

```
yaml
strategy:
  matrix:
    os: [ubuntu-latest, windows-latest]
    node-version: [18.x, 20.x]
```



# GitHub Actions: Matrix

```
yaml
jobs:
  job-de-test:
    strategy:
      matrix:
        os: [ubuntu-latest, windows-latest]
    runs-on: ${{ matrix.os }}
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v4
      - run: npm install
```



# Relation entre les termes

Événement (déclencheur)



workflow (fichier YAML)



jobs (taches parallèles)



runner (Serveur)



steps (étapes d'un job)



actions (unités de travail)

Dupliqué, si la stratégie matrix est utilisée



# Variables GitHub

## Variable

`github.ref_name`

`github.sha`

`github.repository`

`github.workspace`

`github.workflow`

`github.job`

`github.event_name`

## Description

Nom court de la branche ou du tag.

Le SHA du commit qui a déclenché le workflow.

Le propriétaire et le nom du dépôt (repository).

Le répertoire de travail par défaut sur le runner.

Le nom du workflow tel que défini dans le fichier YAML.

L'identifiant (job\_id) du job en cours.

Le nom de l'événement.

## Exemple

main ou v1.0

ffac537e6cb...

hibanajjar998/cours-devops-25-26

/home/runner/work/my-repo

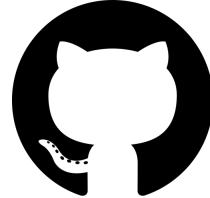
Pipeline CI/CD

quality-check

push

# Exemple: CI dans MyHelloApp

1. On revient à l'exemple du HelloApp
2. On ajoute le script de **test de qualité de code & tests unitaires**
3. On crée notre workflow dans GitHub Actions (ou par ligne de commande)
4. On teste / debug notre workflow



hibanajjar998/  
cours-devops-25-26/  
lesson3\_IntegrationContinue/  
exemple1\_helloapp\_ci

# Application: CI dans ArticlesApp

1. On revient à l'exemple de l'app d'articles
2. On ajoute le script de **test de qualité de code & tests unitaires**
3. On crée notre workflow dans GitHub Actions (ou par ligne de commande)
4. On teste / debug notre workflow



hibanajjar998/  
cours-devops-25-26/  
lesson3\_IntegrationContinue/  
exemple2\_artcilesapp\_ci



- mettez votre code sur GitHub, dans un répertoire privé (nouveau ou déjà existant),
- m'ajouter comme collaboratrice (**hibanajjar998**)
- Mettre le lien du repository ici: <https://tinyurl.com/GI4BCI> ou : <https://tinyurl.com/GI4ACI>