

# Développement front avancé

MMI 3 – TP#4 S6

Danielo **JEAN-LOUIS**

# Au début du déploiement de sites web

- Planification du déploiement
  - HTML, CSS, PHP, dépendances...
- Upload de fichiers divers : traductions...
- Action effectuée par un être humain
  - Gros risque d'erreurs / oublis

Avec la complexité des projets, chaque mise en production est une tâche fastidieuse, risquée et longue. Elle peut faire perdre beaucoup d'argent à des entreprises en cas de mauvaise manipulation

# Intégration continue / Livraison continue

- Appelé communément CI/CD
  - **C**ontinuous **I**ntegration/**C**ontinuous **D**elivery ou **D**eployment
- Automatisation de tâches sur un serveur :
  - Compilation, déploiement, tests unitaires...
- Facilite le déploiement de projets

La CI/CD consiste à créer une chaîne de commandes du développement au déploiement

# Intégration continue / Livraison continue

- Prévient les bugs en production et lors du déploiement
- Entre dans la logique de SCRUM : livraison régulière d'une nouvelle itération

# Intégration continue / Livraison continue

- S'articule souvent autour d'un VCS (Version Control System)
- Existe dans toute typologie de projet : site web, application mobile...



# Intégration continue / Livraison continue

- Géré par un(e) DevOps
  - Métier combinant le développement (dev) et l'administration système (ops)
  - Profil très recherché
  - Connaît le terminal et les commandes linux de base : git, cd, touch, ssh...

# Intégration continue / Livraison continue

## Grandes étapes

- 1) Compilation
- 2) Test : qualité, unitaires, e2e
- 3) Déploiement

Source(s) :

- <https://about.gitlab.com/fr-fr/topics/ci-cd/cicd-pipeline/>

# Intégration Continue (CI)

- Vérifie le code à chaque modification du code source. Ex : quand on effectue une pull request
- Permet de détecter les problèmes en amont

Source(s) :

- <https://developer.mozilla.org/fr/docs/Web/HTML/Element/script/type/importmap>

# Livraison / Déploiement Continue (CD)

- Gère les environnements intermédiaires :
  - Stage, preprod... (delivery)
- Déploie sur le serveur de production (deployment)
- Création de versions
  - Permet un rollback prompt en cas de problème

Source(s) :

- <https://github.com/WICG/import-maps?tab=readme-ov-file#installation>

# git

- VCS le plus populaire
- Présent par défaut sous linux et macOS
- Pierre angulaire du CI/CD

# gitignore

- Fichier permettant d'exclure des fichiers du versionning
- Permet d'alléger les dépôts
- Pensez toujours à en mettre un dans vos projets

Source(s) :

- <https://github.com/github/gitignore>

# Github Actions

- Solution freemium permettant la CI/CD sur n'importe quel dépôt sur github
  - Automatisation de tâches
- Repose sur un système de tâches appelée "Actions"
  - Certaines sont définies (voir sources)
  - Vous pouvez créer les vôtres

Source(s) :

- <https://docs.github.com/fr/actions>
- <https://github.com/actions>

# Github Actions

- Fonctionne avec des conteneurs Docker
- Envoie un e-mail, si échec
- Gère des fichiers YAML placés dans le dossier “.github/workflows”

Source(s) :

- <https://docs.github.com/fr/actions>



# Fichier .yaml / .yml

- Format souvent utilisé pour la configuration
  - Utilisé notamment par Symfony
- Inspiré par le format CSV
  - YAML utilise des indentations pour générer une hiérarchie
- Permet la gestion de données complexes
  - Tout en gardant une lisibilité

Source(s) :

- <https://fr.wikipedia.org/wiki/YAML>

# Fichier .yaml / .yml



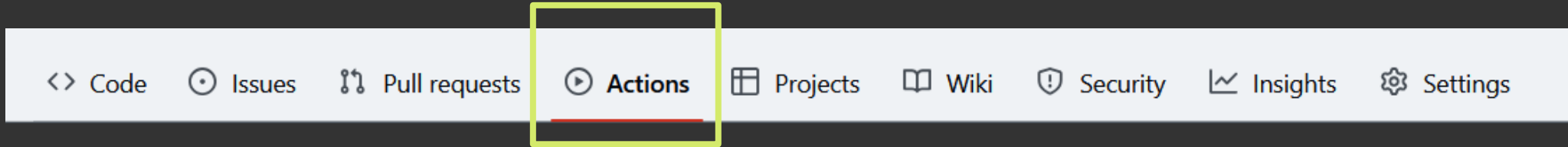
YML

```
formation: MMI
parcours: développement web
list_students:
  - firstname: Helena # Comment
    lastname: Despoux

  - firstname: Thomas
    lastname: Martin
```

Exemple de fichier YAML

# Github Actions



L'onglet "Actions" peut être désactivé

Source(s) :

- <https://docs.github.com/fr/actions>

# Pratiquons ! - Github actions (Partie 1)

Pré-requis :

- Avoir la ressource ressources/github-actions

A télécharger ici :

[https://github.com/DanYellow/cours/raw/refs/heads/main/developement-web-et-dispositif-interactif-s6/travaux-pratiques/numero-3/developpement-web-et-dispositif-interactif-s6\\_travaux-pratiques\\_numero-4.ressources.zip](https://github.com/DanYellow/cours/raw/refs/heads/main/developement-web-et-dispositif-interactif-s6/travaux-pratiques/numero-3/developpement-web-et-dispositif-interactif-s6_travaux-pratiques_numero-4.ressources.zip)

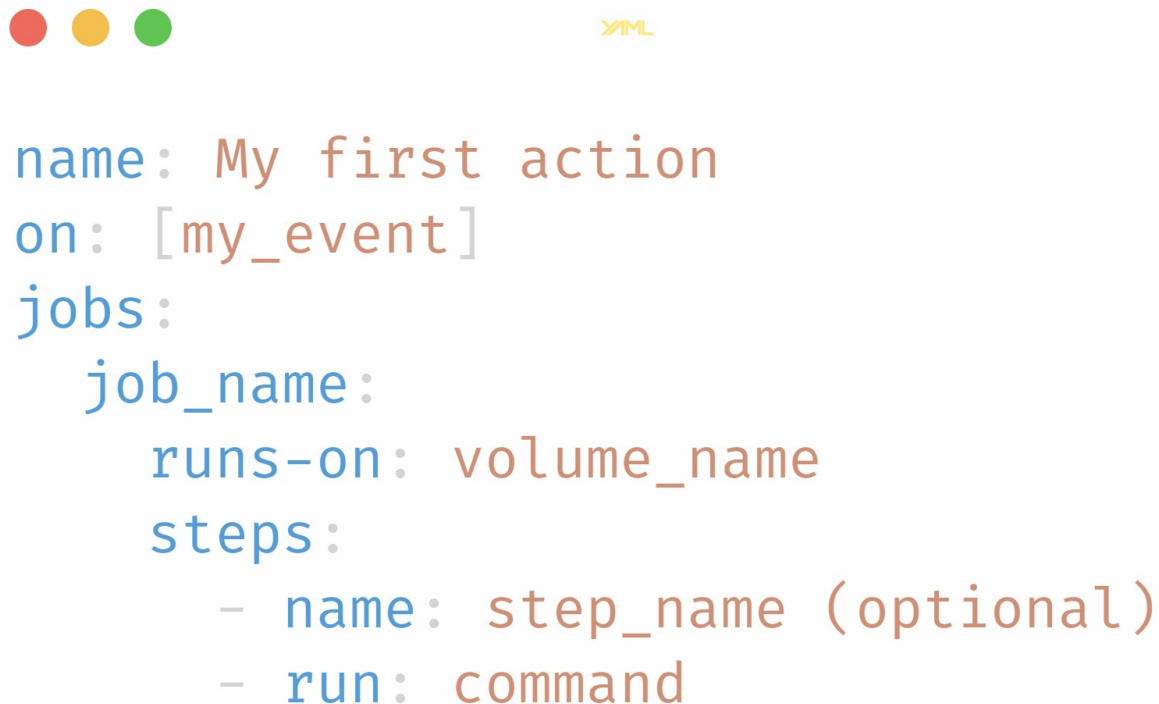
# Github Actions - Gabarit

- Chaque fichier d'actions doit contenir au moins trois clés racines :
  - name : Nom de la tâche
  - on : quand l'Action est exécutée ? (pull, push...) - Valeurs définies (voir source)
  - jobs : Tâches à effectuer

Source(s) :

- <https://docs.github.com/fr/actions/writing-workflows/choosing-when-your-workflow-runs/events-that-trigger-workflows>

# Github Actions - Gabarit



```
name: My first action
on: [my_event]
jobs:
  job_name:
    runs-on: volume_name
    steps:
      - name: step_name (optional)
      - run: command
```

Exemple de base d'un fichier d'actions. Pour "on", le tableau n'est pas obligatoire.

Source(s) :

- <https://docs.github.com/fr/actions/writing-workflows/choosing-when-your-workflow-runs/events-that-trigger-workflows>

# Github Actions - Gabarit

- Le même fichier peut contenir plusieurs jobs
  - Il est préférable de séparer votre pipeline en plusieurs jobs
    - Un job : une grande tâche

Source(s) :

- <https://docs.github.com/fr/actions/writing-workflows/choosing-when-your-workflow-runs/events-that-trigger-workflows>

# Github Actions – Actions définies

- Ensemble d'actions déjà définie part Github. Ex : Pull le dépôt
  - Action “actions/checkout@v4”
- S'utilise avec la clé “uses” (à la place de “run”) dans le fichier yml

Source(s) :

- <https://docs.github.com/fr/actions/writing-workflows/choosing-when-your-workflow-runs/events-that-trigger-workflows>



# Pratiquons ! - Github Actions (Partie 2)

Pré-requis :

- Avoir la ressource ressources/github-actions

A télécharger ici :

[https://github.com/DanYellow/cours/raw/refs/heads/main/developement-web-et-dispositif-interactif-s6/travaux-pratiques/numero-3/developpement-web-et-dispositif-interactif-s6\\_travaux-pratiques\\_numero-4.ressources.zip](https://github.com/DanYellow/cours/raw/refs/heads/main/developement-web-et-dispositif-interactif-s6/travaux-pratiques/numero-3/developpement-web-et-dispositif-interactif-s6_travaux-pratiques_numero-4.ressources.zip)

# Github Actions – Variables

- Permettent de réutiliser une valeur
- Plusieurs portée possibles :
  - Globale, job, tâche
- Préfixée par “\$” pour être affichée

# Github Actions – Variables

```
name: Display a variable

on:
  workflow_dispatch

env:
  UNIVERSITY: CY Paris Université # Global scope

jobs:
  display_student_infos:
    runs-on: ubuntu-latest
    env:
      FORMATION: BUT MMI # Job scope
    steps:
      - name: "Presentation"
        run: echo "I'm $FIRST_NAME, I'm a student in $FORMATION at $UNIVERSITY"
        env:
          FIRST_NAME: John Doe # Step scope
```

Ici, nous avons trois variables avec trois portées différentes

# Github Actions – Context

- Objets par défaut permettant d'accéder à diverses informations : état du job, nom de l'utilisateur courant...
- S'affiche "\${{ <context> }}"

Source(s) :

- <https://docs.github.com/en/actions/writing-workflows/choosing-what-your-workflow-does/accessing-contextual-information-about-workflow-runs>

# Github Actions – Secrets

- Définissent les variables d'environnement
  - Variables qui ne doivent pas être publiques...
  - ...mais qu'on veut utiliser dans ses Actions
- Données chiffrées
- Ne doivent pas commencer par un nombre
  - Caractères alphanumériques et underscore seulement

# Github Actions – Secrets

- Valeurs non sensible à la casse
- Chargés depuis les paramètres du dépôt
  - Settings > Secrets and variables > Actions
- S'affichent comme les variables de contexte
  - `${{ secrets.SECRET_KEY }}`

# Github Actions – Secrets

General

Access

Collaborators

Moderation options

Code and automation

Branches

Tags

Rules

Actions

Webhooks

Environments

Codespaces

Pages

Security

Code security

Deploy keys

Secrets and variables

Actions

Codespaces

Dependabot

## Actions secrets and variables

Secrets and variables allow you to manage reusable configuration data. Secrets are **encrypted** and are used for sensitive data. [Learn more about encrypted secrets](#). Variables are shown as plain text and are used for **non-sensitive** data. [Learn more about variables](#).

Anyone with collaborator access to this repository can use these secrets and variables for actions. They are not passed to workflows that are triggered by a pull request from a fork.

SecretsVariables

### Environment secrets

This environment has no secrets.

Manage environment secrets

### Repository secrets

This repository has no secrets.

New repository secret

# Github Actions – Artifact

- Représente un dossier **persistant** sur le serveur
  - Souvent un build. Ex : npm run build
  - Durée de vie par défaut : 90 jours
- Peut être lu par d'autres jobs
  - Ex : Tâche de déploiement

Source(s) :

- <https://docs.github.com/fr/actions/writing-workflows/choosing-what-your-workflow-does/storing-and-sharing-data-from-a-workflow>



# Github Actions – Artifact

- Nécessite l'action “action/upload-artifact@master” pour être partagé



```
steps:  
  - name: Generate artifact  
    uses: actions/upload-artifact@master  
    with:  
      name: bundle # artifact / directory name on the server  
      path: ./dist # source directory
```

On copie le contenu du dossier “dist” dans un artifact nommé “bundle”

Source(s) :

- <https://docs.github.com/fr/actions/writing-workflows/choosing-what-your-workflow-does/storing-and-sharing-data-from-a-workflow>

# Github Actions – Inter-dépendances

- Permet d'attendre l'exécution d'un job avant l'exécution d'un autre
  - Multiple dépendances possibles
- Utilisation de la clé “needs”

## Source(s) :

- <https://docs.github.com/en/actions/writing-workflows/choosing-what-your-workflow-does/using-jobs-in-a-workflow#defining-prerequisite-jobs>

# Github Actions – Inter-dépendances



```
deploy:
  runs-on: ubuntu-latest
  needs: [build]

  steps:
    - name: # ...
```

Notre job “deploy” ne peut s’exécuter que si et seulement si le job “build” est terminé

Source(s) :

- <https://docs.github.com/en/actions/writing-workflows/choosing-what-your-workflow-does/using-jobs-in-a-workflow#defining-prerequisites-jobs>

# Pratiquons ! - Github Actions (Partie 3)

Pré-requis :

- Avoir la ressource ressources/github-actions

A télécharger ici :

[https://github.com/DanYellow/cours/raw/refs/heads/main/developement-web-et-dispositif-interactif-s6/travaux-pratiques/numero-3/developpement-web-et-dispositif-interactif-s6\\_travaux-pratiques\\_numero-4.ressources.zip](https://github.com/DanYellow/cours/raw/refs/heads/main/developement-web-et-dispositif-interactif-s6/travaux-pratiques/numero-3/developpement-web-et-dispositif-interactif-s6_travaux-pratiques_numero-4.ressources.zip)

# Github Actions – Alternatives – Liste non exhaustive

- Travis CI
- Gitlab
- Circle CI
- Jenkins : Nécessite beaucoup de configuration
- TeamCity
- ...

**Questions ?**

