

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...
- Tests manuels
- 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 d'erreur

Intégration continue / Livraison continue

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

Automatisation - Avantages

- Limite les risques d'erreurs et d'oubli
- Assure d'avoir le même environnement
- Permet de traquer et rejouer les erreurs aisément

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

Intégration continue / Livraison continue

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

Exemple : Knight Capital (08/2012)

- **Ancienne** entreprise de trading à haute fréquence
- Plus gros négociateur d'actions américaines
 - Représentait ~17 % des parts de marché sur le New York Stock Exchange (NYSE) et le NASDAQ

Source(s) :

- https://en.wikipedia.org/wiki/Knight_Capital_Group#2012_stock_trading_disruption - anglais
- <https://programmation.developpez.com/actu/361198/Knights-Capital-a-ete-victime-du-bogue-logiciel-le-plus-couteux-de-l-histoire-de-l-humanite-49-millions-de-dollars-par-seconde-8-6-milliards-de-dollars-en-28-minutes/>

Exemple : Knight Capital (08/2012)

- Perte de 440 millions de dollars à cause une mise à jour de serveur incomplète
 - Un serveur avait été oublié → Exécution de 4 millions d'ordres d'achat non voulus et **non testés**
 - Fonction de test cassée lors de la dernière mäj

L'intégration continue aurait évité cette catastrophe

Source(s) :

- https://en.wikipedia.org/wiki/Knight_Capital_Group#2012_stock_trading_disruption - anglais
- <https://programmation.developpez.com/actu/361198/Knights-Capital-a-ete-victime-du-bogue-logiciel-le-plus-couteux-de-l-histoire-de-l-humanite-49-millions-de-dollars-par-seconde-8-6-milliards-de-dollars-en-28-minutes/>

Intégration continue / Livraison continue

- S'articule souvent autour d'un VCS (Version Control System)
 - Git, svn, perforce...
- Existe dans toute typologie de projet : site web, application mobile...
- Géré par un(e) DevOps

DevOps

- Métier combinant le développement (dev) et l'administration système (ops)
- Profil très recherché
- Est à l'aise avec les commandes Linux de base : cd, touch, ssh...
- Connaît un VCS (indispensable pour la CI/CD)

DevOps

- Connaît les infrastructures Cloud : AWS, Azure...
- Facilite le déploiement en production du code
 - Phase **critique** du développement logiciel
- Automatise certaines tâches

Un(e) DevOps priorise les processus avant les outils de déploiement. Autrement dit, il apporte une « culture » en entreprise.

Utiliser un outil de CI/CD sans comprendre le contexte ne fait pas de vous un(e) DevOps.

Intégration continue / Livraison continue

Grandes étapes

1) Compilation

- Suite à un push

2) Test : performances, unitaires, e2e, sécurité...

- Automatisés et manuels

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
 - Un problème résolu en dev coûte moins cher qu'en production

Source(s) :

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

Livraison / Déploiement Continu (CD)

- Gère les environnements intermédiaires :
 - Stage, preprod... (delivery)
- Déploie sur le serveur de production (deployment)
 - Permet un déploiement partiel (Canary release), Blue-Green deployment...

Source(s) :

- <https://github.com/WICG/import-maps?tab=readme-ov-file#installation>
- <https://geekflare.com/fr/blue-green-vs-canary-deployment/>

Livraison / Déploiement Continu (CD)

- Création de versions (versioning)
 - Permet un rollback prompt en cas de problème
- Permet de générer un build de production à tout moment
 - Livraison possible de petites mises à jour

Source(s) :

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

CI / CD - Schéma

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 dépôt
- Permet d'alléger les dépôts
 - **Inutile de commiter vos dépendances**
- 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
- Permet de créer un pipeline (ou workflow)
 - Composé de jobs (un ensemble de tâches)
- Documentation en français

Source(s) :

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

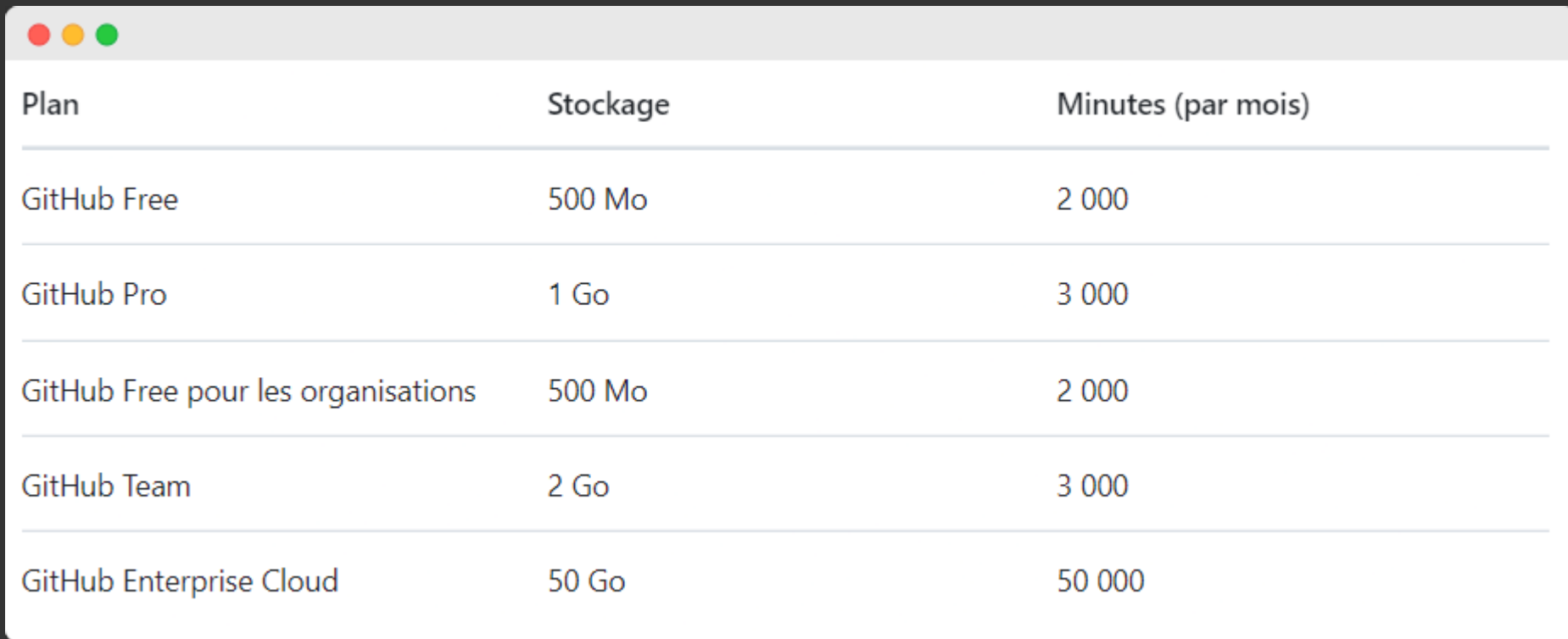
Github Actions

- Documentation en français
- Fonctionne sur macOS, Windows et Linux
 - Linux coûte le moins cher

Source(s) :

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

Github Actions - Tarification



Plan	Stockage	Minutes (par mois)
GitHub Free	500 Mo	2 000
GitHub Pro	1 Go	3 000
GitHub Free pour les organisations	500 Mo	2 000
GitHub Team	2 Go	3 000
GitHub Enterprise Cloud	50 Go	50 000

Passé ces limites, vous serez facturé(e) à la minute (temps d'exécution d'une tâche)

Source(s) :

- <https://docs.github.com/fr/billing/managing-billing-for-your-products/managing-billing-for-github-actions/about-billing-for-github-actions>

Github Actions

- Fonctionne avec des conteneurs Docker
- Envoie un e-mail, si échec
- Permet d'effectuer des cron
 - Cron : Tâches planifiées
- 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



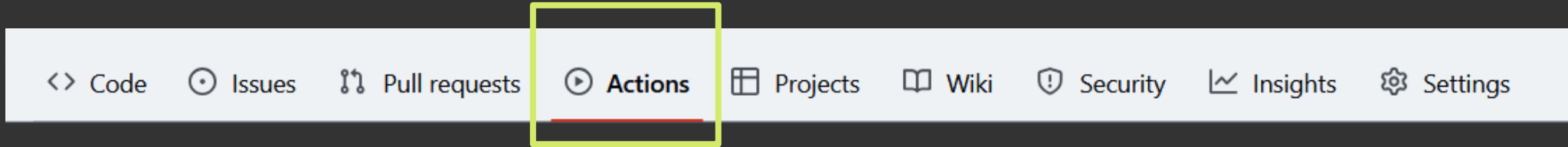
YML

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

  - firstname: Thomas
    lastname: Martin
```

Exemple de fichier YAML

Github Actions

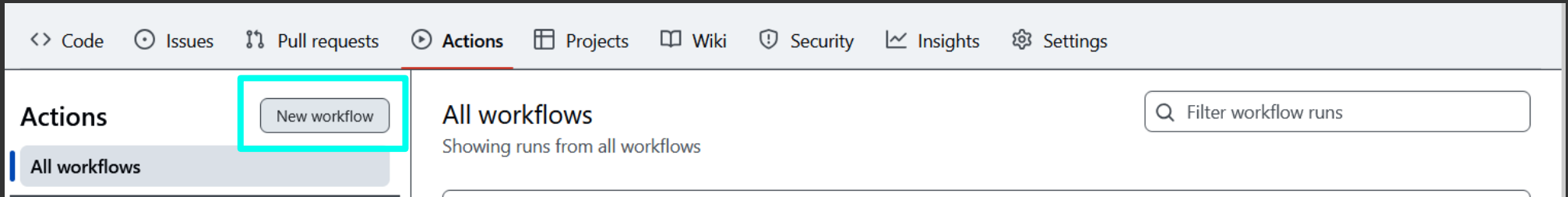


L'onglet "Actions" peut être désactivé

Source(s) :

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

Github Actions



Choose a workflow

Build, test, and deploy your code. Make code reviews, branch management, and issue triaging work the way you want. Select a workflow to get started.

Skip this and [set up a workflow yourself →](#)

Github permet d'éditer des pipelines depuis le site web

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

Act

- Outil gratuit permettant de faire tourner vos Github Actions en local
- Nécessite Docker sur votre ordinateur

Source(s) :

- <https://github.com/nektos/act>

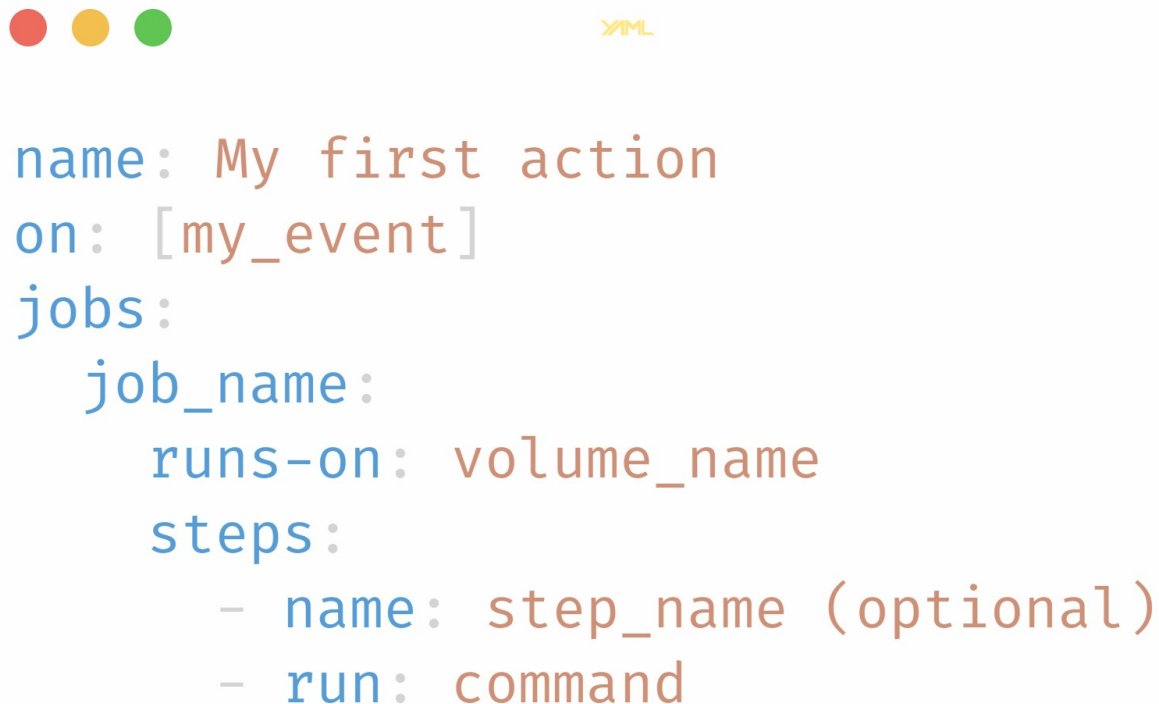
Github Actions - Gabarit

- Chaque fichier d'actions doit contenir au moins deux clés racines :
 - 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 - Vocabulaire

- Pipeline / workflow : fichier YAML exécuté quand un évènement a lieu
- Évènement : Déclenche l'exécution d'un workflow
- Job : Ensemble d'étapes exécutées sur un serveur

Source(s) :

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

Github Actions – Action définie

- Application personnalisée et complexe. Ex : Tirage de dépôt
 - Développés par la communauté et Github
- S'utilise avec la clé “uses” (à la place de “run”) dans le fichier yml
 - Possibilité d'avoir plusieurs uses au sein du même job

Source(s) :

- <https://docs.github.com/fr/actions/writing-workflows/choosing-when-your-workflow-runs/events-that-trigger-workflows>
- <https://github.com/sdras/awesome-actions?tab=readme-ov-file#official-actions>

Github Actions – Actions définies

```
name: Node Continuous Integration

on:
  push:
    branches: [ master ]

jobs:
  create_build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - uses: actions/setup-node@v1
        with:
          node-version: 20
      - name: Install dependencies
        run: npm ci
      - name: Create build
        run: npm run build
```

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

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.

Secrets Variables

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
 - 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 – Artifact

- Nécessite l'action “action/download-artifact@master” pour être récupéré

```
steps:  
  - name: Download artifact  
    uses: actions/download-artifact@v4  
    with:  
      name: my_artifact # Artifact to download  
      path: ./build # Destination path
```

On récupère le contenu de notre artifact “my_artifact” dans le dossier build

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]

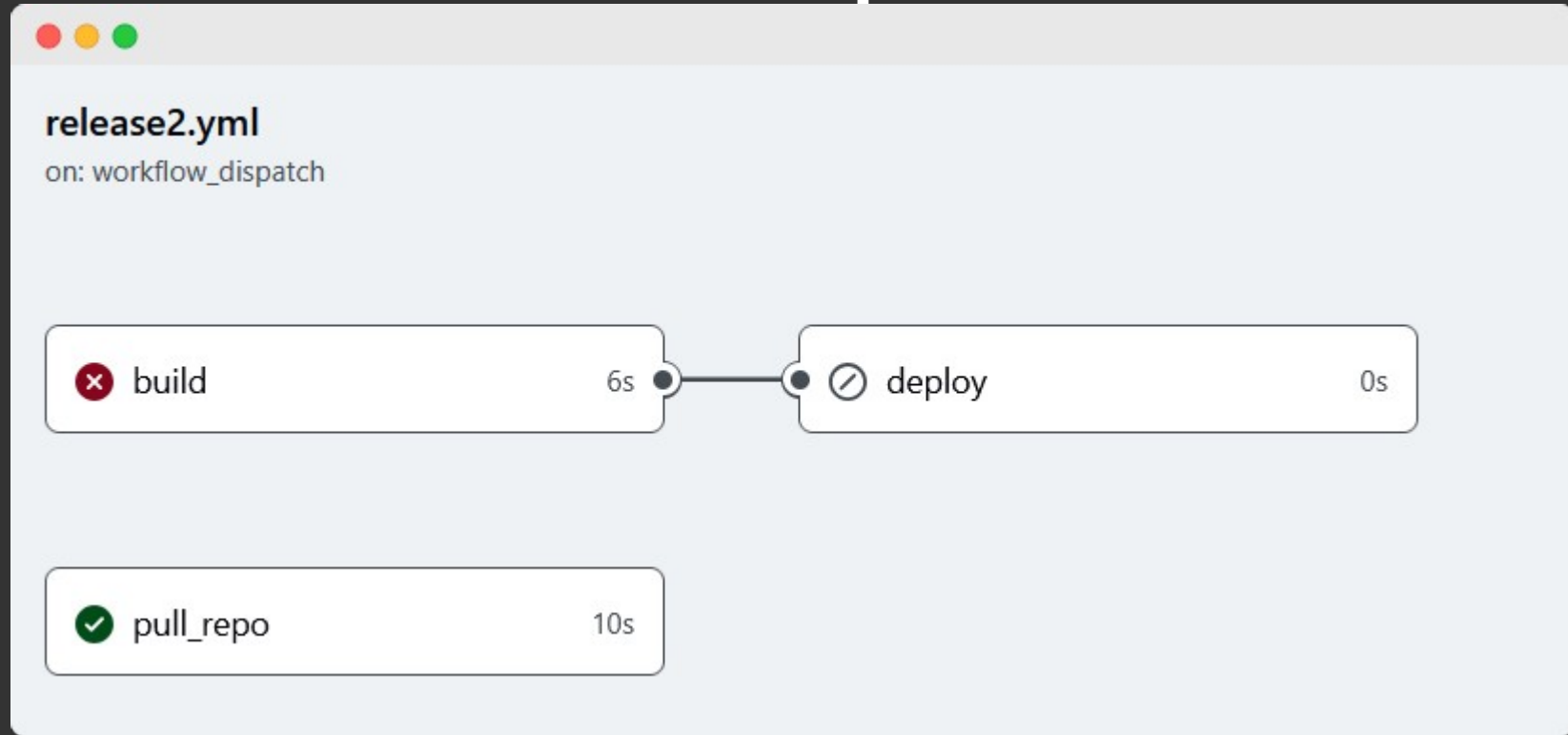
  steps:
    - name: # ...
```

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>

Github Actions – Inter-dépendances



L'interface de Github Actions nous indique l'interdépendance de jobs

Source(s) :

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

Github Actions – Inputs

- Permet de définir des valeurs depuis github qui seront utilisées dans votre workflow
 - Ex : définir le serveur de stage
- Plusieurs types de données possibles : choice (équivalent select en HTML), boolean, string et environnement

Source(s) :

- <https://github.blog/changelog/2021-11-10-github-actions-input-types-for-manual-workflows/>

Github Actions – Inputs

- Ne fonctionne qu'avec les workflows lancés manuellement
 - on : workflow_dispatch

Source(s) :

- <https://github.blog/changelog/2021-11-10-github-actions-input-types-for-manual-workflows/>

Github Actions – Inputs

```
name: Github inputs

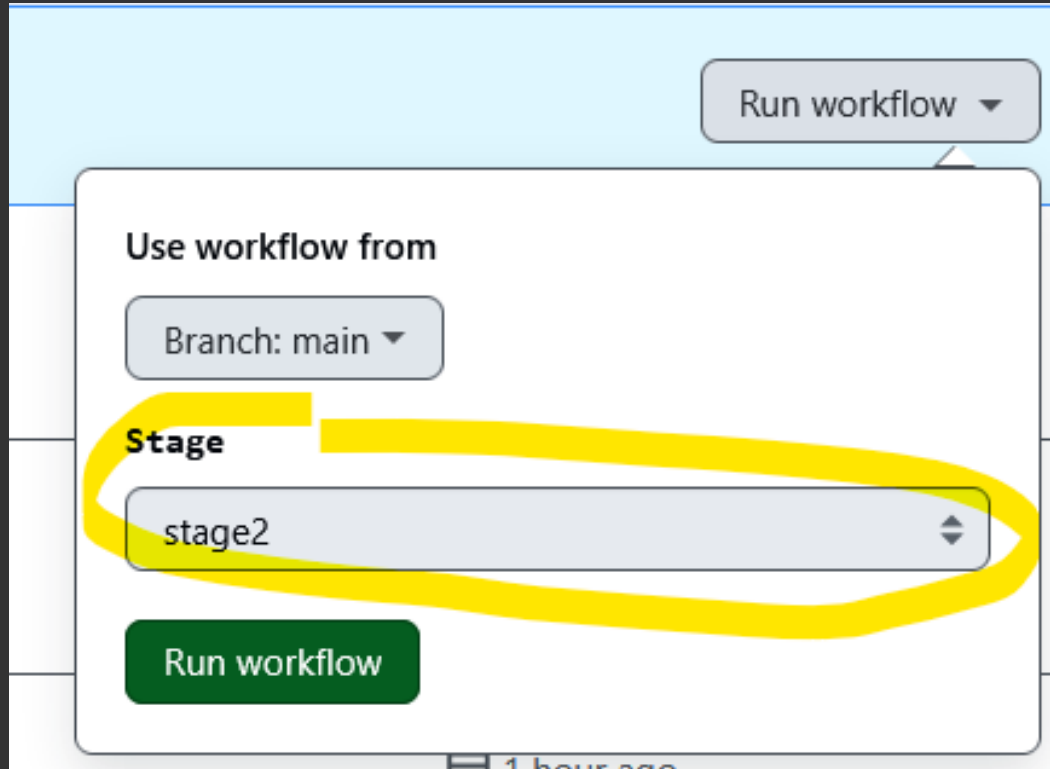
on:
  workflow_dispatch:
    stage:
      type: choice
      description: Select stage env
      default: stage2
      options:
        - stage1
        - stage2

jobs:
  my_input:
    # [ ... ]
    steps:
      - name: Use my input value
        run: echo "${{ github.event.inputs.stage }}"
```

Source(s) :

- <https://github.blog/changelog/2021-11-10-github-actions-input-types-for-manual-workflows/>

Github Actions – Inputs



Depuis l'interface des actions, je peux changer à la volée des variables de mon workflow

Source(s) :

- <https://github.blog/changelog/2021-11-10-github-actions-input-types-for-manual-workflows/>

Github Actions – Alternatives – Liste non exhaustive

- Travis CI
- Gitlab - https://docs.gitlab.com/ee/ci/quick_start/
- Circle CI
- Jenkins : Nécessite **beaucoup** de configuration
- TeamCity
- ...

Questions ?

