Gestion des caches

La mise en cache

La mise en cache dans GitHub Actions peut être un outil puissant pour accélérer vos workflows.

Vous pouvez éviter de télécharger à chaque fois les mêmes dépendances ou de générer à nouveau des fichiers qui n'ont pas changé.

Cela est particulièrement utile pour des outils de gestion de dépendances comme npm, Maven, pip, etc.

Vous pouvez utiliser l'action `actions/cache` de GitHub pour mettre en cache des fichiers entre les exécutions de job.

Un cache est identifié par une clé unique. Vous pouvez également utiliser une liste de clés de restauration (restore-keys) en tant que plan B, dans le cas où une clé de cache ne correspond pas exactement.

Les paramètres de action/cache

key

C'est la clé explicite pour une entrée de cache. Cette clé est souvent composée d'une chaîne fixe, du système d'exploitation et d'un hash des fichiers de dépendances pour s'assurer qu'elle est unique.

Exemple :

```
key: ${{ runner.os }}-node-${{ hashFiles('**/package-lock.json') }}
```

path

Il s'agit de la liste des fichiers, répertoires et motifs génériques à mettre en cache et à restaurer. Vous pouvez utiliser les motifs supportés par `@actions/glob`.

Exemple :

```
path: |
    path: ~/.npm
```

Premier exemple : npm

```yaml
name: Exemple de mise en cache avec npm

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - name: Vérification du code
      uses: actions/checkout@v4

    - name: Mise en cache des dépendances npm
      uses: actions/cache@v3
      with:
        path: ~/.npm
        key: ${{ runner.os }}-node-${{ hashFiles('**/package-lock.json')
}}
        restore-keys: |
          ${{ runner.os }}-node-

    - name: Installation des dépendances
      run: npm ci --cache .npm
```

Dans cet exemple, le job effectue les étapes suivantes :

1. Il récupère le code source.
2. Il tente de restaurer le cache basé sur une clé unique générée à partir du système d'exploitation et du contenu du fichier package-lock.json. Plus précisément, il génère un hash SHA-256 à partir du fichier grâce à la fonction hashFiles().
3. S'il y a un cache hit (une correspondance exacte de la clé), le cache est restauré.
4. S'il y a un cache miss (aucune correspondance exacte), le job continue et installe les dépendances qui seront mises en cache pour les prochains runs.

Pourquoi mettre en cache .npm et non node_modules ?

`action/cache` recommande pour Node.js de mettre en cache le cache *npm* ou *yarn* et non pas **node_modules** pour les raisons suivantes :

- **Compatibilité entre systèmes d'exploitation** : le cache de npm est plus neutre en termes de système d'exploitation et d'architecture, ce qui rend le cache plus portable entre différentes configurations de runners.
- **Performance** : npm utilise son propre cache pour stocker des fichiers de package et des métadonnées de manière optimisée. Lors de l'exécution de npm install, npm peut décider de manière plus intelligente quelles dépendances doivent être téléchargées et lesquelles peuvent être récupérées du cache.

- **Intégrité** : npm vérifie l'intégrité des packages en utilisant des sommes de contrôle. Cela réduit le risque de corruption du cache.
- **Espace disque / vitesse téléchargement** : le cache de npm est souvent plus petit que le répertoire node_modules car il ne contient pas les dépendances décompressées. Cela peut accélérer la sauvegarde et la restauration du cache.

Deuxième exemple : PHP

```yaml
name: Mon PHP Workflow

on:
  push:
    branches:
      - main
  pull_request:
    branches:
      - main

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - name: checkout le code
      uses: actions/checkout@v4

    - name: Get Composer Cache Directory
      id: composer-cache
      run: |
        echo "dir=$(composer config cache-files-dir)" >> $GITHUB_OUTPUT

    - uses: actions/cache@v3
      with:
        path: ${{ steps.composer-cache.outputs.dir }}
        key: ${{ runner.os }}-composer-${{ hashFiles('**/composer.lock')
}}
        restore-keys: |
          ${{ runner.os }}-composer-

    # Installe les dépendances
    - name: installer les dépendances
      run: composer install
```

Autres codes pour les environnements les plus courants

Python

```yaml
- uses: actions/cache@v3
  with:
    path: ~/.cache/pip
    key: ${{ runner.os }}-pip-${{ hashFiles('**/requirements.txt') }}
    restore-keys: |
      ${{ runner.os }}-pip-
```

Java - Gradle

```yaml
- uses: actions/cache@v3
  with:
    path: |
      ~/.gradle/caches
      ~/.gradle/wrapper
    key: ${{ runner.os }}-gradle-${{ hashFiles('**/*.gradle*', '**/gradle-
wrapper.properties') }}
    restore-keys: |
      ${{ runner.os }}-gradle-
```

Java - Maven

```yaml
- name: Cache local Maven repository
  uses: actions/cache@v3
  with:
    path: ~/.m2/repository
    key: ${{ runner.os }}-maven-${{ hashFiles('**/pom.xml') }}
    restore-keys: |
      ${{ runner.os }}-maven-
```

C#

```yaml
- uses: actions/cache@v3
  with:
    path: ~/.nuget/packages
    key: ${{ runner.os }}-nuget-${{ hashFiles('**/packages.lock.json') }}
    restore-keys: |
      ${{ runner.os }}-nuget-
```