Lab Github Action

Table de matière

Documentation

Ce projet est un lab pour mettre en place une chaine d'intégration sur un projet Spring Boot avec Github Action et Docker. En effet ceci est un projet pour mettre en place un pipeline CI/CD avec Github Action…

Vue d ensemble

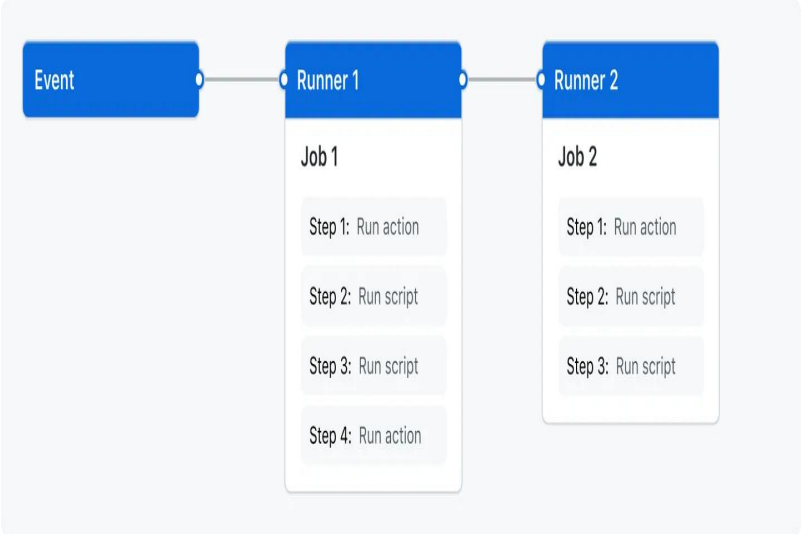
GitHub Actions est une plateforme d'intégration continue et livraison continue (CI/CD) qui vous permet d'automatiser votre pipeline de génération, de test et de déploiement. Vous pouvez créer des workflows qui créent et testent chaque demande de tirage (pull request) adressée à votre dépôt, ou déployer des demandes de tirage fusionnées en production.

GitHub Actions dépasse DevOps simplement et vous permet d'exécuter des workflows lorsque d'autres événements se produisent dans votre dépôt. Par exemple, vous pouvez exécuter un workflow pour ajouter automatiquement les étiquettes appropriées chaque fois que quelqu'un crée un problème dans votre dépôt.

GitHub fournit des machines virtuelles Linux, Windows et macOS pour exécuter vos workflows, ou vous pouvez héberger vos propres exécuteurs auto-hébergés dans votre propre centre de données ou infrastructure cloud.

Composants de GitHub Actions

Vous pouvez configurer un workflow GitHub Actions à déclencher quand un événement se produit dans votre dépôt, par exemple l'ouverture d'une demande de tirage (pull request) ou la création d'un problème. Votre workflow contient un ou plusieurs travaux qui peuvent s'exécuter dans un ordre séquentiel ou en parallèle. Chaque travail s'exécute au sein de son propre exécuteur de machine virtuelle, ou au sein d'un conteneur, et comporte une ou plusieurs étapes qui exécutent un script que vous définissez ou une action, qui est une extension réutilisable qui peut simplifier votre workflow.

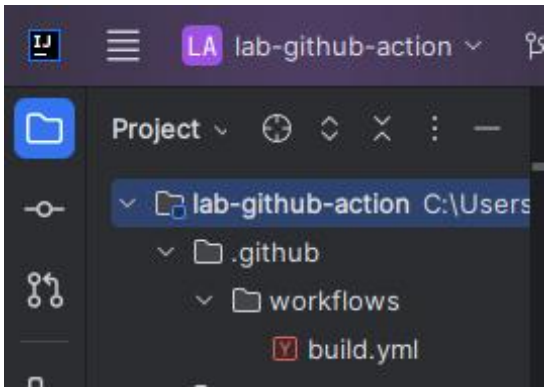


Workflow GitHub Actions

Workflows

Un workflow est un processus automatisé configurable qui exécutera un ou plusieurs travaux. Les workflows sont définis par un fichier YAML archivé dans votre dépôt et s'exécutent lorsqu'ils sont déclenchés par un événement dans votre dépôt, ou ils peuvent être déclenchés manuellement ou selon une planification définie.

Les workflows sont définis dans le répertoire `.github/workflows` d'un référentiel, et un référentiel peut avoir plusieurs workflows, chacun pouvant effectuer un ensemble différent de tâches. Par exemple, vous pouvez avoir un workflow pour générer et tester des demandes de tirage, un autre workflow pour déployer votre application chaque fois qu'une version est créée, et encore un autre workflow qui ajoute une étiquette chaque fois que quelqu'un ouvre un nouveau problème.



GitHub Actions Workflows Folder

Événements

Un événement est une activité spécifique dans un dépôt qui déclenche l'exécution d'un workflow. Par exemple, l'activité peut provenir de GitHub quand quelqu'un crée une demande de tirage (pull request), ouvre un problème ou pousse (push) un commit vers un dépôt. Vous pouvez également déclencher une exécution de workflow selon une planification, en publiant dans une API REST ou manuellement. > Exemple:

```
on:
  push:
    branches: [ main]
```

Jobs

Un travail est un ensemble d'étapes dans un workflow qui s'exécute sur le même exécuteur. Chaque étape est un script d'interpréteur de commandes qui sera exécuté ou une action qui sera exécutée. Les étapes sont exécutées dans l'ordre et dépendent les unes des autres. Comme chaque étape est exécutée sur le même exécuteur, vous pouvez partager des données d'une étape à une autre. Par exemple, vous pouvez avoir une étape qui génère votre application suivie d'une étape qui teste l'application générée.

Vous pouvez configurer les dépendances d'un travail avec d'autres travaux. Par défaut, les travaux n'ont aucune dépendance et s'exécutent en parallèle entre eux. Lorsqu'un travail prend une dépendance sur un autre travail, il attend que le travail dépendant se termine avant de pouvoir s'exécuter. Par exemple, vous pouvez avoir plusieurs travaux de génération pour différentes architectures qui n'ont pas de dépendances, et un travail d'empaquetage dépendant de ces travaux. Les travaux de génération s'exécutent en parallèle et le travail d'empaquetage s'exécutera quand ils auront fini de s'exécuter. > Exemple:

```
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
```

Actions

Une action est une application personnalisée pour la plateforme GitHub Actions qui effectue une tâche complexe mais fréquemment répétée. Utilisez une action pour réduire la quantité de code répétitif que vous écrivez dans vos fichiers de workflow. Une action peut tirer (pull) votre dépôt git à partir de GitHub, configurer la chaîne d'outils appropriée pour votre environnement de build ou configurer l'authentification auprès de votre fournisseur de cloud.

Vous pouvez écrire vos propres actions ou trouver des actions à utiliser dans vos workflows dans le GitHub Marketplace.

Pour partager des actions au sein de votre entreprise sans les publier publiquement, vous pouvez les stocker dans un référentiel interne, puis configurer celui-ci pour autoriser l'accès aux workflows GitHub Actions dans d'autres référentiels appartenant à la même organisation ou à toute autre organisation de l'entreprise. > Exemple:

```
- name: Récuperation du projet...
uses: actions/checkout@v2
```

Exécuteurs

Un exécuteur est un serveur qui exécute vos workflows quand ils sont déclenchés. Chaque exécuteur peut exécuter un seul travail à la fois. GitHub fournit les exécuteurs Ubuntu Linux, Microsoft Windows et macOS pour exécuter vos workflows. Chaque exécution de workflow s'exécute sur une machine virtuelle nouvellement provisionnée. GitHub propose également des exécuteur plus grand, qui sont disponibles dans des configurations plus grandes. Si vous avez besoin d'un système d'exploitation différent ou d'une configuration matérielle spécifique, vous pouvez héberger vos propres exécuteurs. > Exemple:

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

Exemple complet de workflow

GitHub Actions utilise la syntaxe YAML pour définir le workflow. Chaque workflow est stocké en tant que fichier YAML distinct dans votre référentiel de code, dans un répertoire appelé .github/workflows.

Vous pouvez créer un exemple de workflow dans votre dépôt qui déclenche automatiquement une série de commandes chaque fois que du code est poussé

(push). Dans ce workflow, GitHub Actions extrait le code envoyé, installe la version 17 de Java avec la distribution temurin, build le projet avec maven et exécute une commande de base pour builder l'image du projet afin de le publier sur Docker Hub toujours en utilisant maven. > Exemple:

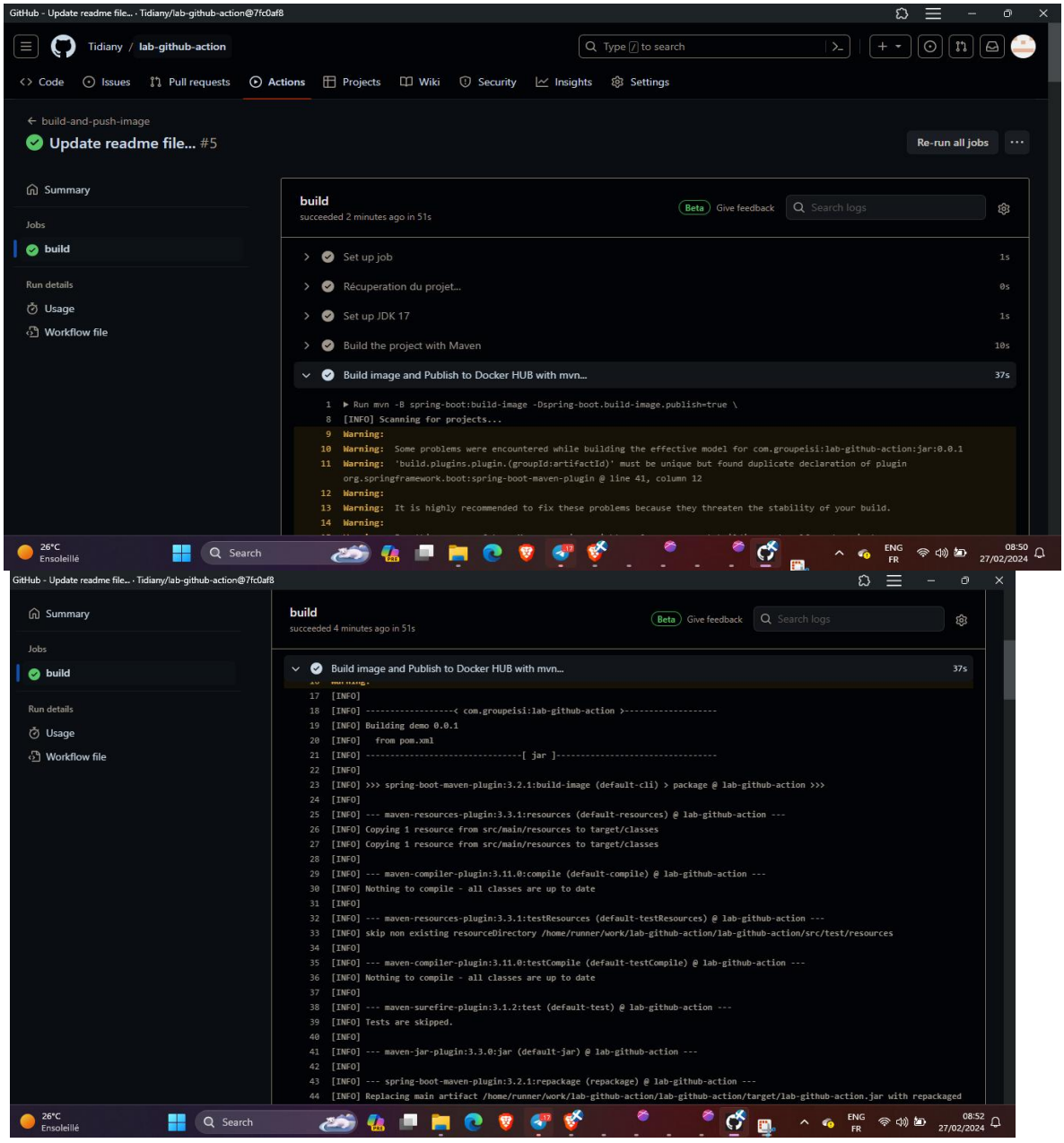
```
name: build-and-push-image

on:
  push:
    branches: [ main, release ]
    paths-ignore:
      - '**/README.md'
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - name: Récuperation du projet...
        uses: actions/checkout@v2
      - name: Set up JDK 17
        uses: actions/setup-java@v2
        with:
          java-version: '17'
          distribution: 'temurin'
          cache: maven
      - name: Build the project with Maven
        run: mvn -B package
      - name: Build image and Publish to Docker HUB with mvn...
        run: |
          mvn -B spring-boot:build-image -Dspring-boot.build-image.publ
ish=true \
               -Ddocker.user=${{ secrets.DOCKER_USER }} -Ddocker.token=
${{ secrets.DOCKER_TOKEN }} \
               -DskipTests
```

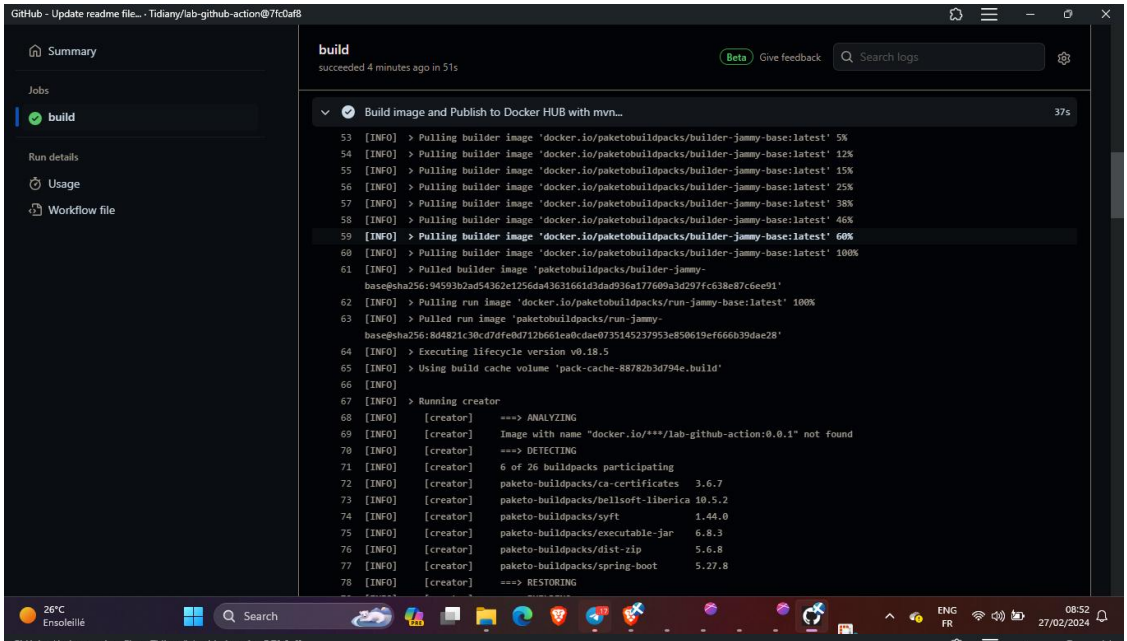
Votre nouveau fichier de workflow GitHub Actions est maintenant installé dans votre dépôt et s'exécute automatiquement chaque fois que quelqu'un pousse (push) une modification vers le dépôt.

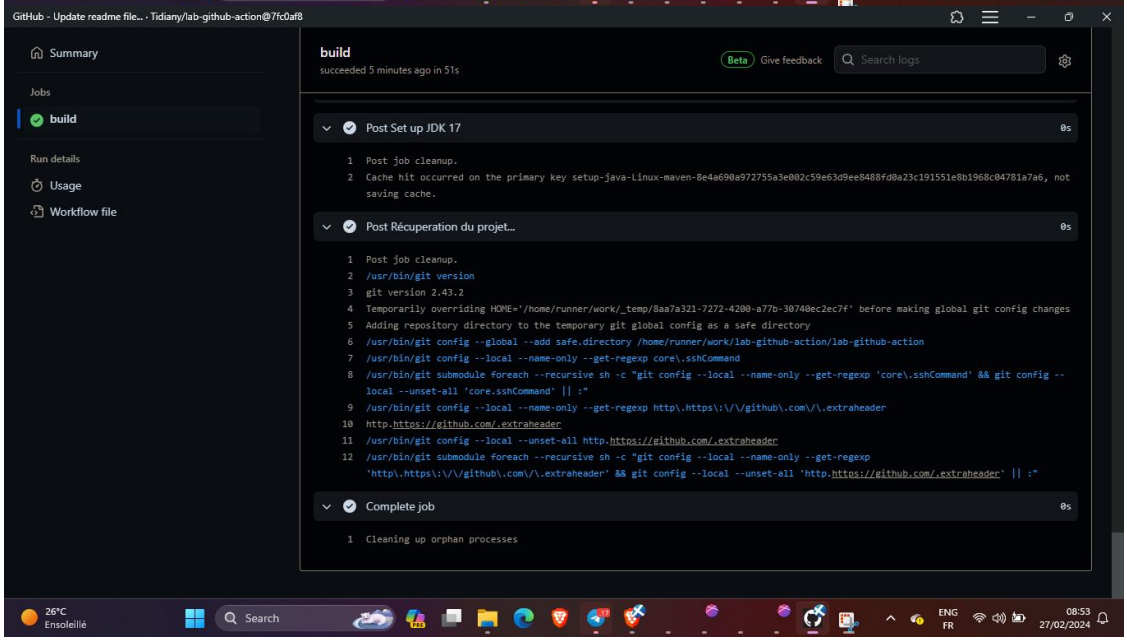
Ainsi une fois la validation des modifications et le push éffectuer vers notre dépôt GitHub, la machine se déclanche.

Illustration:

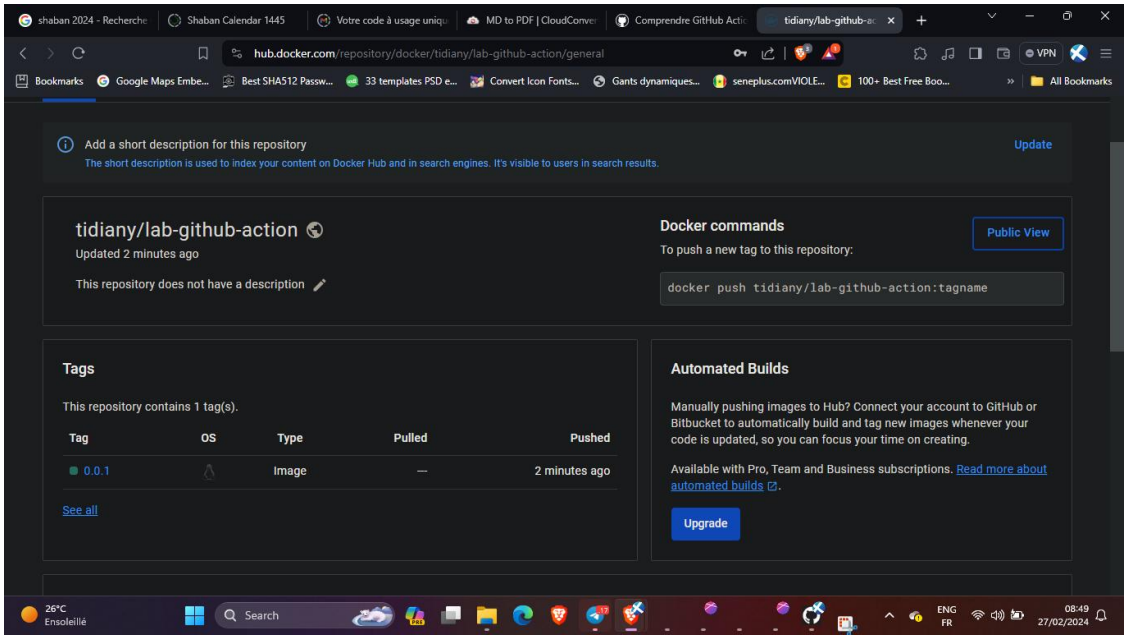
Lancement du build




GitHub - Update readme file... · Tidiany/lab-github-action@7fc0af8
Summary
Jobs
build
Run details
Usage
Workflow file
build
succeeded 4 minutes ago in 51s
Beta
Give feedback
Search logs
Build image and Publish to Docker HUB with mvn...
37s
53 [INFO] > Pulling builder image 'docker.io/paketobuildpacks/builder-jammy-base:latest' 5%
54 [INFO] > Pulling builder image 'docker.io/paketobuildpacks/builder-jammy-base:latest' 12%
55 [INFO] > Pulling builder image 'docker.io/paketobuildpacks/builder-jammy-base:latest' 15%
56 [INFO] > Pulling builder image 'docker.io/paketobuildpacks/builder-jammy-base:latest' 25%
57 [INFO] > Pulling builder image 'docker.io/paketobuildpacks/builder-jammy-base:latest' 38%
58 [INFO] > Pulling builder image 'docker.io/paketobuildpacks/builder-jammy-base:latest' 46%
59 [INFO] > Pulling builder image 'docker.io/paketobuildpacks/builder-jammy-base:latest' 60%
60 [INFO] > Pulling builder image 'docker.io/paketobuildpacks/builder-jammy-base:latest' 100%
61 [INFO] > Pulled builder image 'paketobuildpacks/builder-jammy-base@sha256:94593b2ad54362e1256da43631661d3dad936a177609a3d297fc638e87c6ee91'
62 [INFO] > Pulling run image 'docker.io/paketobuildpacks/run-jammy-base:latest' 100%
63 [INFO] > Pulled run image 'paketobuildpacks/run-jammy-base@sha256:8d4821c30cd7dfe0d712b661ea0cdae07351452379 53e850619ef666b39dae28'
64 [INFO] > Executing lifecycle version v0.18.5
65 [INFO] > Using build cache volume 'pack-cache-88782b3d794e.build'
66 [INFO]
67 [INFO] > Running creator
68 [INFO] [creator] ===> ANALYZING
69 [INFO] [creator] Image with name "docker.io/***/lab-github-action:0.0.1" not found
70 [INFO] [creator] ===> DETECTING
71 [INFO] [creator] 6 of 26 buildpacks participating
72 [INFO] [creator] paketo-buildpacks/ca-certificates 3.6.7
73 [INFO] [creator] paketo-buildpacks/bellsoft-liberica 10.5.2
74 [INFO] [creator] paketo-buildpacks/syft 1.44.0
75 [INFO] [creator] paketo-buildpacks/executable-jar 6.8.3
76 [INFO] [creator] paketo-buildpacks/dist-zip 5.6.8
77 [INFO] [creator] paketo-buildpacks/spring-boot 5.27.8
78 [INFO] [creator] ===> RESTORING
26°C
Ensoleillé
Search
ENG
FR
08:52
27/02/2024


GitHub - Update readme file... · Tidiany/lab-github-action@7fc0af8
Summary
Jobs
build
Run details
Usage
Workflow file
build
succeeded 5 minutes ago in 51s
Beta
Give feedback
Search logs
Post Set up JDK 17
0s
1 Post job cleanup.
2 Cache hit occurred on the primary key setup-java-Linux-maven-8e4a690a972755a3e002c59e63d9ee8488fd0a23c191551e8b1968c04781a7a6, not saving cache.
Post Récuperation du projet...
0s
1 Post job cleanup.
2 /usr/bin/git version
3 git version 2.43.2
4 Temporarily overriding HOME='/home/runner/work/_temp/8aa7a321-7272-4200-a77b-30740ec2ec7f' before making global git config changes
5 Adding repository directory to the temporary git global config as a safe directory
6 /usr/bin/git config --global --add safe.directory /home/runner/work/lab-github-action/lab-github-action
7 /usr/bin/git config --local --name-only --get-regexp core\.sshCommand
8 /usr/bin/git submodule foreach --recursive sh -c "git config --local --name-only --get-regexp 'core\.sshCommand' && git config --local --unset-all 'core.sshCommand' || :"
9 /usr/bin/git config --local --name-only --get-regexp http\.https\:\/\/github\.com\/\.extraheader
10 http.https://github.com/.extraheader
11 /usr/bin/git config --local --unset-all http.https://github.com/.extraheader
12 /usr/bin/git submodule foreach --recursive sh -c "git config --local --name-only --get-regexp 'http\.https\:\/\/github\.com\/\.extraheader' && git config --local --unset-all 'http.https://github.com/.extraheader' || :"
Complete job
0s
1 Cleaning up orphan processes
26°C
Ensoleillé
Search
ENG
FR
08:53
27/02/2024

Deploiement sur Docker Hub



Perspectives

1. Crée un job pour les test sonar
2. Crée un job pour se connecter sur AWS déployé le jar sur un EC2.
3. configuration DNS
4. Exécution du jar sur notre serveur EC2
5. Faire le link entre les différents jobs.