

Jérôme BOEBION

GitHub Actions

[HTTPS://DOCS.GITHUB.COM/FR/ACTIONS](https://docs.github.com/fr/actions)



GitHub Actions

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.

- Créer des workflows qui testent chaque demande de pull request adressée à votre dépôt, ou déploie des demandes pull merge en production.
- 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.
- Pour cela, 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

Votre workflow contient **un ou plusieurs travaux** qui peuvent s'exécuter dans **un ordre séquentiel ou en parallèle**.

Syntaxe de Yaml :

<https://blog.stephane-robert.info/docs/developper/autres-langages/yaml/introduction/>

■ 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 GitHub** et s'exécutent lorsqu'ils sont déclenchés par un événement dans votre dépôt. *[ou ils peuvent être déclenchés manuellement ou selon une planification définie].*

■ Événements

- Un événement est une activité spécifique dans un dépôt qui déclenche l'exécution d'un workflow.
- <https://docs.github.com/fr/actions/using-workflows/events-that-trigger-workflows>

■ Travaux (Jobs)

- Ensemble d'étapes dans un workflow qui s'exécute sur le même **exécuteur**. Chaque étape est un **script d'interpréteur de commandes ou une action** qui seront 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.

Composants de GitHub Actions

Chaque **travail** s'exécute au sein de son propre **exécuteur** (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.

■ Actions

- Une action est une application personnalisée pour la plateforme GitHub Actions qui effectue une tâche complexe mais fréquemment répétée.
- Utilisez une action permet de réduire la quantité de code répétitif que vous écrivez dans vos fichiers de workflow.

■ Script d'interpréteur de commandes

- Selon l'environnement de l'exécuteur, il s'agit d'un ensemble de commandes réalisant une action (par exemple [mvn package](#))

■ Exécuteurs (environnement)

- 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.

Composants de GitHub Actions

Parmi les actions, nous pouvons retrouver des **checkout** et utiliser des **variables secrètes** permettant de respecter la confidentialité de la configuration de nos workflows

■ Checkout

- L'action **Checkout** est l'une des actions les plus couramment utilisées dans les workflows GitHub Actions. Elle permet de cloner le code du projet dans l'environnement du workflow, rendant ainsi le code source disponible pour les étapes suivantes.
- Cela garantit que votre workflow dispose toujours de la dernière version du code source avant de commencer à exécuter les jobs.

■ Secrets

- Les secrets sont des variables sensibles stockées de manière sécurisée et utilisées dans les workflows.
- Ils sont généralement utilisés pour stocker des informations sensibles telles que des clés d'API, des jetons d'accès ou des mots de passe.
- GitHub Actions offre un espace de stockage sécurisé pour ces secrets.

Composants de GitHub Actions

Enfin, nous allons pouvoir mettre en place des **conditions**, afin de réaliser nos workflows selon le comportement du dépôt.

Le **marketplace** est une ressource de partage de la communauté GitHub.

■ Conditions

- GitHub Actions permet d'utiliser des conditions pour déterminer si un job doit être exécuté en fonction d'une évaluation.
- Par exemple, vous pouvez définir des conditions pour exécuter un job uniquement si une certaine branche est modifiée, si un pull request a été ouvert, ou si d'autres critères spécifiques sont remplis.
- Cela vous permet de personnaliser davantage le comportement de vos workflows en fonction des événements et des circonstances..

■ Marketplace GitHub Actions

- Le Marketplace GitHub Actions est une ressource précieuse qui offre un vaste choix d'actions pré-construites que vous pouvez intégrer dans vos workflows. Ces actions sont créées par la communauté GitHub et couvrent une variété de cas d'usage.
- <https://github.com/marketplace?type=actions> : Vous y trouverez une liste d'actions prédéfinies, triées par catégorie et par popularité. Vous pouvez parcourir ces actions pour trouver celles qui correspondent le mieux à vos besoins.

Les exécuteurs

Runs-on permet de définir le type de machine sur laquelle le travail doit être exécuté.

<https://docs.github.com/fr/actions/using-jobs/choosing-the-runner-for-a-job>

- Si vous utilisez un exécuteur hébergé par GitHub, chaque travail s'exécute sur une nouvelle instance d'une image d'exécuteur spécifiée par le mot-clé **runs-on**.
- Les étiquettes d'exécuteurs hébergés par GitHub disponibles sont les suivantes :
 - **Ubuntu-latest**, **ubuntu-22.04**, ubuntu-20.04
 - **Windows-latest**, **windows-2022**, windows-2019
 - **Macos-latest**, **macos-12**, macos-11
- Mais il est possible d'avoir ses propres exécuteurs. Ainsi, l'étiquette d'exécuteurs auto-hébergé est la suivante :
 - **Self-hosted**

Création d'un 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).
- Exemple d'un workflow

```
name: learn-github-actions
run-name: ${ github.actor } is learning GitHub Actions
on: [push]
jobs:
  check-bats-version:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v3
        with:
          node-version: '14'
      - run: npm install -g bats
      - run: bats -v
```


Description du fichier de workflow

Pour vous aider à comprendre comment la syntaxe YAML est utilisée pour créer un fichier de workflow, cette section explique chaque ligne de l'exemple d'introduction.

```
# Optional - le nom du workflow qui apparaîtra dans le dépôt
name: learn-github-actions
# affiche dans le contexte du workflow, le nom de l'utilisateur déclencheur
run-name: ${github.actor} is learning GitHub Actions
# spécifie le déclencheur
on: [push]


# Groupes des travaux
jobs:

# Définie le nom du Job et ses propriétés
check-bats-version:

# Configure le job sur la dernière version de Ubuntu
runs-on: ubuntu-latest

# Les étapes du jobs
steps:

# Scripts ou Actions

# Le mot-clé « uses » indique que cette étape exécutera
# « v4 » de l'action « actions/checkout ».
#  s'agit d'une action qui vérifie votre référentiel sur le runner, vous permettant d'exécuter des scripts ou
# d'autres actions contre votre code (comme les outils de construction et de test).
# Vous devez utiliser l'action de vérification chaque fois que votre flux de travail utilisera le code du référentiel.
- uses: actions/checkout@v4

# Cette étape utilise l'action 'actions/setup-node@v3' pour installer la version spécifiée de Node.js.
#(Cet exemple utilise la version 14.) Ceci place les commandes 'node' et 'npm' dans votre 'PATH'.
- uses: actions/setup-node@v3
  with:
    node-version: '14'

# exécution d'une installation de bats
- run: npm install -g bats

# exécution de bats pour afficher la version
- run: bats -v
```

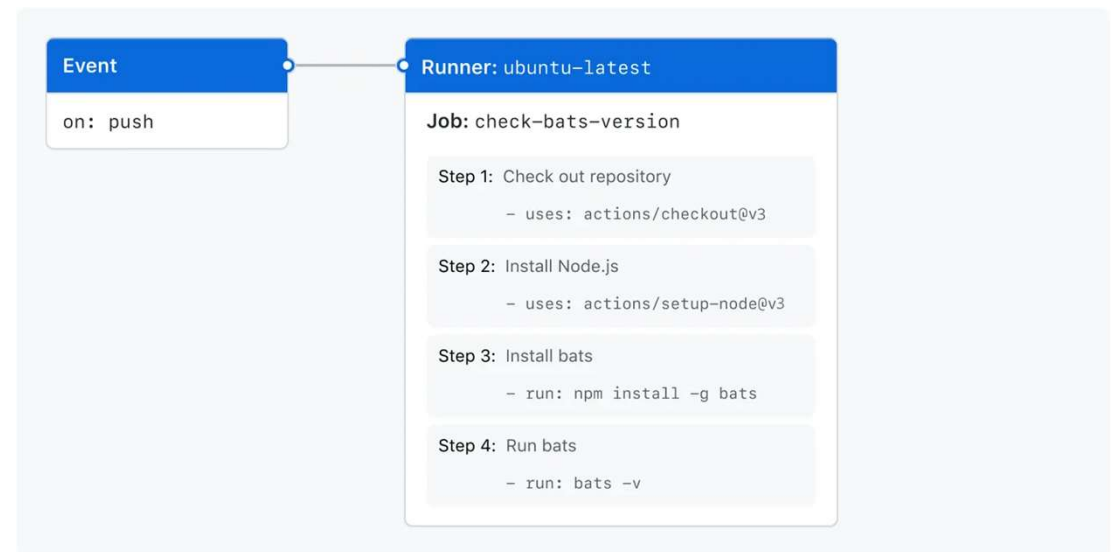
Visualisation du fichier de workflow

Dans ce diagramme, vous pouvez voir le fichier de workflow que vous venez de créer et comment les composants GitHub Actions sont organisés dans une hiérarchie.

Chaque étape exécute une action ou un script d'interpréteur de commandes unique.

Les étapes 1 et 2 exécutent des actions, tandis que les étapes 3 et 4 exécutent des scripts d'interpréteur de commandes.

<https://docs.github.com/fr/actions/learn-github-actions/finding-and-customizing-actions>

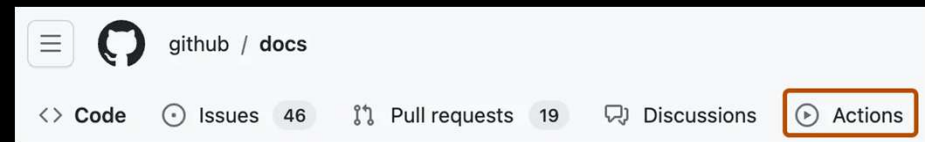


Affichage de l'activité dans GitHub

Lorsque votre workflow est déclenché, une *exécution de workflow* est créée et exécute le workflow.

Une fois l'exécution de votre workflow démarrée, vous pouvez voir un graphe de visualisation de la progression de l'exécution, ainsi que l'activité de chaque étape sur GitHub.

1. Sur Github.com, accédez à la page principale du dépôt.
2. Sous le nom de votre dépôt, cliquez sur [Actions](#)



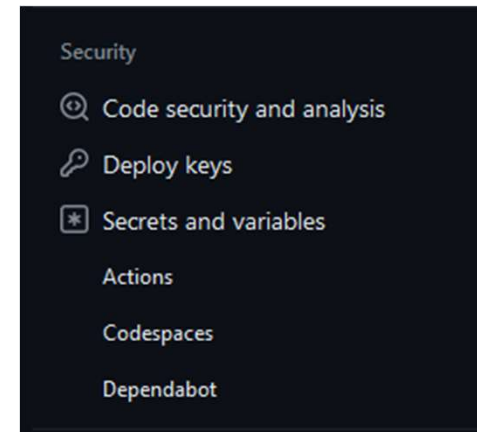
3. Dans la barre latérale cliquez sur le workflow que vous souhaitez afficher.
4. Dans la liste des exécutions de workflow, cliquez sur le nom de l'exécution pour voir le résumé de l'exécution du workflow
5. Dans la barre latérale à gauche ou dans le graphe de visualisation, cliquez sur le travail que vous souhaitez voir.
6. Pour voir les résultats d'une étape, cliquez sur l'étape.

Création de variables secrètes

GitHub Actions vous permet de créer des variables pouvant contenir des données à protéger.

16/04/2025

- Depuis les [Settings](#) de votre dépôt et la rubrique [Security](#) :



- Allez dans la rubrique [Secrets and Variables > Actions](#)
- En cliquant sur [New repository secret](#), vous pouvez créer une variable secrète.
- Cette variable est utilisable dans le script via la syntaxe `${{ secrets.name }}`

GitHub Actions artefacts

SAUVEGARDE DE DONNÉES DES WORKFLOWS

Stockage des données de workflow en tant qu'artefacts

Les artefacts vous permettent de conserver des données une fois un travail terminé, et de partager ces données avec un autre travail du même workflow.

- Un artefact est un fichier ou une collection de fichiers générés pendant l'exécution d'un workflow.
- Par exemple, vous pouvez utiliser des artefacts pour enregistrer votre sortie de build et de test une fois l'exécution d'un workflow terminée.
- Toutes les actions et tous les workflows appelés dans une exécution disposent d'un accès en écriture aux artefacts de cette exécution.
 - Par défaut, GitHub Enterprise Server stocke les artefacts et journaux de build pendant 90 jours (paramétrable). La période de conservation d'une demande de tirage (pull request) redémarre chaque fois qu'un utilisateur pousse (par push) un nouveau commit à la demande de tirage.
- Attention : service limité voir payant s'il y a des abus.

Un souci avec mon build ?

En effet, si on réfléchit à l'environnement de GitHub Actions, j'exécute mon workflow sur une machine dont je n'ai pas accès physiquement.

Qu'en est-il de mon build de maven ?

Eh bien il est perdu après exécution. Le workflow s'exécute et en fin de workflow, il effectue un nettoyage et supprime le serveur utilisé.



La solution : les artefacts.

Les artefacts vous permettent de partager des données entre travaux dans un workflow et de stocker des données une fois ce workflow terminé.

Donc on va pouvoir stocker nos build dans un artefact.

Documentations GitHub Actions

TROUVER L'INFORMATION

Liens

GitHub Actions peut vous aider à automatiser presque tous les aspects de vos processus de développement d'applications.

Voici quelques ressources utiles pour effectuer vos étapes suivantes avec GitHub Actions

- Pour obtenir un moyen rapide de créer un flux de travail GitHub Actions, consultez « Utilisation de workflows de démarrage ».
 - <https://docs.github.com/fr/actions/learn-github-actions/using-starter-workflows>
- Pour obtenir des workflows d'intégration continue (CI) permettant de générer et tester votre code, consultez « Automatisation des builds et des tests ».
 - <https://docs.github.com/fr/actions/automating-builds-and-tests>
- Pour générer et publier des packages, consultez « Publication de packages ».
 - <https://docs.github.com/fr/actions/publishing-packages>
- Guides
 - <https://docs.github.com/fr/actions/guides>

Liens

Voici quelques ressources utiles pour effectuer vos étapes suivantes avec GitHub Actions

- Pour le déploiement de projets, consultez « Déploiement ».
 - <https://docs.github.com/fr/actions/deployment>
- Pour automatiser les tâches et les processus sur GitHub, consultez « Gestion des problèmes et demandes de tirage ».
 - <https://docs.github.com/fr/actions/managing-issues-and-pull-requests>
- Pour obtenir des exemples illustrant les fonctionnalités plus complexes de GitHub Actions, dont beaucoup des cas d'usage ci-dessus, consultez « Exemples ».
 - <https://docs.github.com/fr/actions/examples>

CRÉATION DE
SON PREMIER
WORKFLOW

Travaux pratiques

Attention : ce qui est présenté dans les différents codes par la suite correspond à un de mes projets et par conséquent propre à son environnement. Chaque projet est différent et a son propre environnement.

Création d'un 1^{er} workflow

Essayez les fonctionnalités de GitHub Actions en 5 minutes ou moins.

Vous n'avez besoin que d'un dépôt GitHub pour créer et exécuter un workflow GitHub Actions. Dans ce guide, vous allez ajouter un workflow qui illustre certaines des fonctionnalités essentielles de GitHub Actions.

L'exemple suivant vous montre comment les travaux GitHub Actions peuvent être déclenchés automatiquement, où ils s'exécutent et comment ils peuvent interagir avec le code dans votre dépôt.

1. Créer un répertoire `.github/workflows` dans le dépôt de votre choix si ce répertoire n'existe pas.
2. Dans le répertoire nouvellement créé, ajouter un fichier `github-action-demo.yml`
3. Ce fichier va contenir le script ci-dessous :

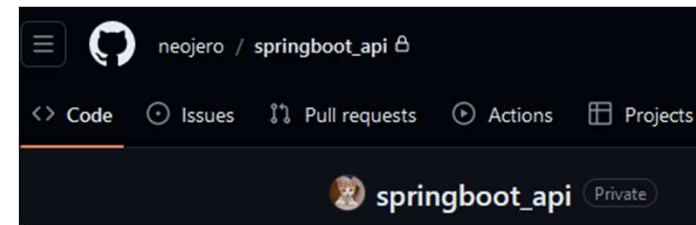
```
name: GitHub Actions Demo
run-name: ${ github.actor }} teste GitHub Actions 🚀
on: [push]
jobs:
  Explore-GitHub-Actions:
    runs-on: ubuntu-latest
    steps:
      - run: echo "🎉 The job was automatically triggered by a ${ github.event_name }} event."
      - run: echo "👤 This job is now running on a ${ runner.os }} server hosted by GitHub!"
      - run: echo "🔖 The name of your branch is ${ github.ref }} and your repository is ${ github.repository }}."
      - name: Check out repository code
        uses: actions/checkout@v4
      - run: echo "💡 The ${ github.repository }} repository has been cloned to the runner."
      - run: echo "📄 The workflow is now ready to test your code on the runner."
      - name: List files in the repository
        run: |
          ls ${ github.workspace }}
      - run: echo "🟢 This job's status is ${ job.status }}."
```

4. Ensuite, effectuez la mise à jour de votre dépôt distant sur GitHub.

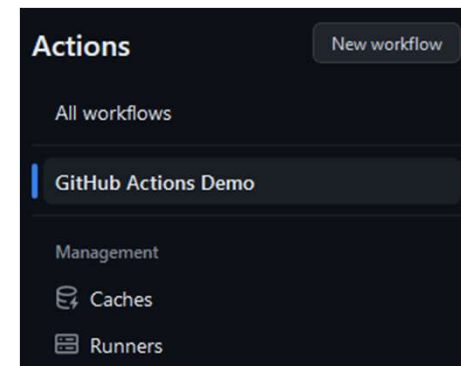
Affichage des résultats de votre workflow

Désormais, la mise à jour du dépôt est effectuée, allons contrôler sur GitHub, notre workflow.

1. Dans GitHub, accédez à la page principale du dépôt.
2. Sous le nom du dépôt, cliquez [Actions](#)



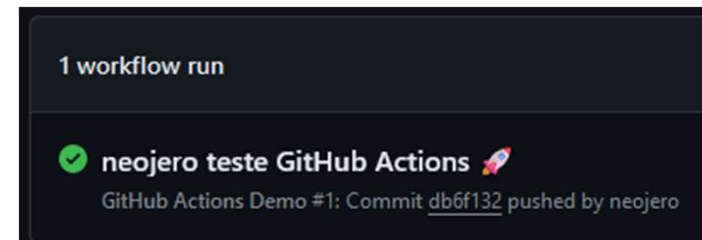
3. Dans la barre latérale gauche, vous avez l'ensemble des workflows. Vous pouvez cliquer sur celui qui vous intéresse.



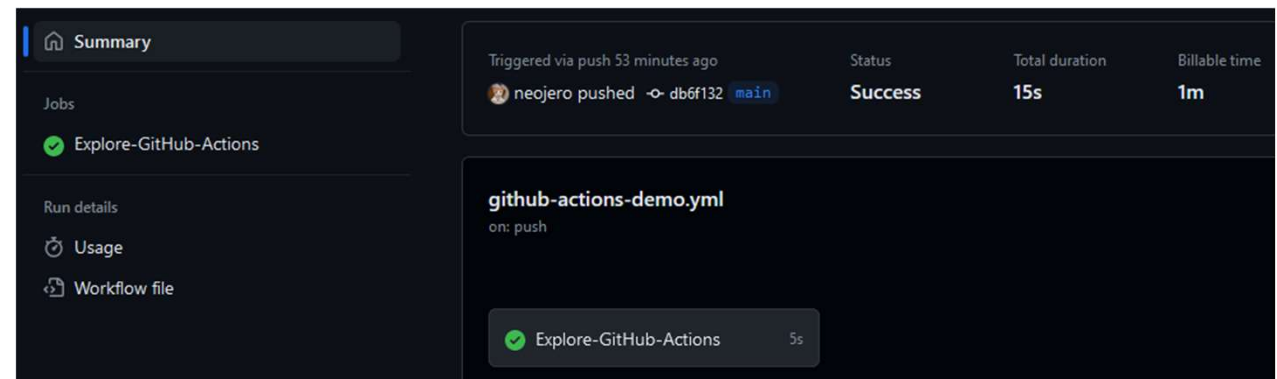
Affichage des résultats de votre workflow

Entrons dans le détail de notre workflow

4. On peut déjà constater que notre workflow affiche comme nom de workflow **USERNAME teste GitHub Actions** 🚀



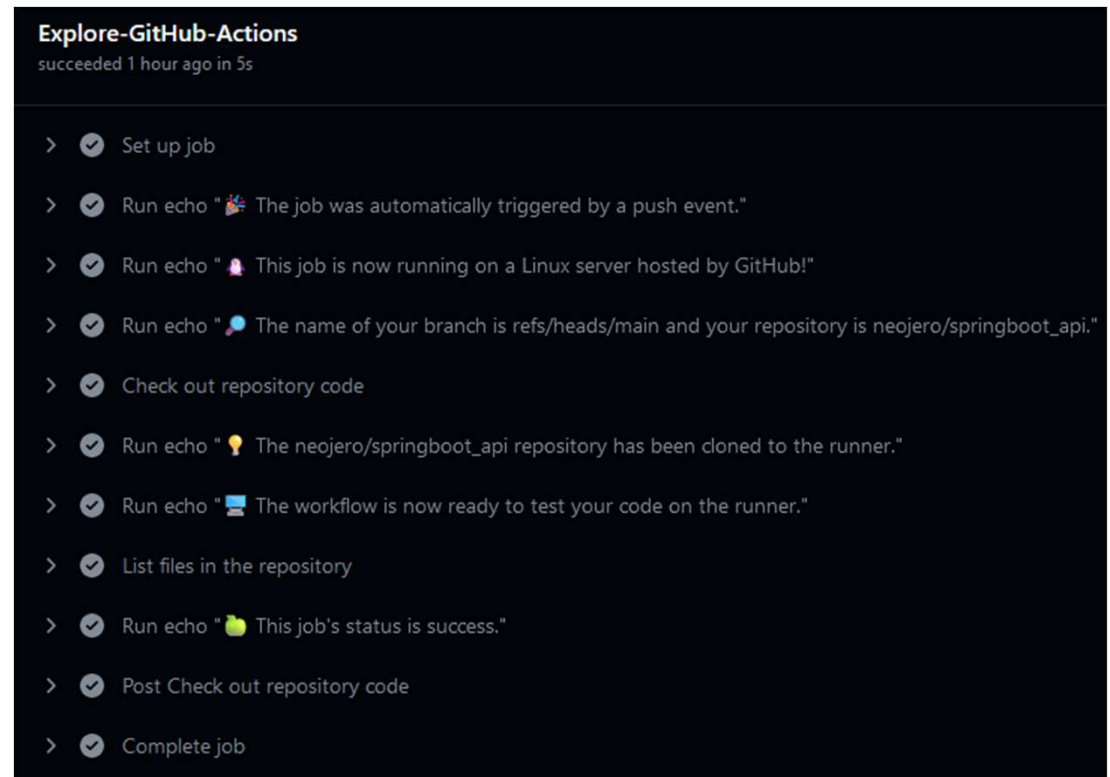
5. En cliquant dessus, on entre dans le détail



Afficher le journal du workflow

Le journal montre comment chacune des étapes a été traitée

16/04/2025



PRÉSENTATION DE GITHUB ACTIONS

Afficher le détail d'une étape

On peut ensuite afficher le détail de chaque étape du workflow.

Ici, le détail lors de l'action checkout.

```
✓ Check out repository code

1  ▶ Run actions/checkout@v4
15 Syncing repository: neojero/springboot_api
16 ▶ Getting Git version info
20 Temporarily overriding HOME='/home/runner/work/_temp/3959cd2a-1144-43ea-89b7-5574dd8c25a0'
21 Adding repository directory to the temporary git global config as a safe directory
22 /usr/bin/git config --global --add safe.directory /home/runner/work/springboot_api/springboot_api
23 Deleting the contents of '/home/runner/work/springboot_api/springboot_api'
24 ▶ Initializing the repository
38 ▶ Disabling automatic garbage collection
40 ▶ Setting up auth
46 ▶ Fetching the repository
50 ▶ Determining the checkout info
51 ▶ Checking out the ref
55 /usr/bin/git log -1 --format='%H'
56 'db6f132efaad822517f4eead5ebf49b7c45fb19e'
```


Suivi des workflows

On peut suivre nos workflows depuis GitHub.

En cas d'échec d'un workflow, un email est envoyé informant le propriétaire du dépôt.

16/04/2025

The image shows a screenshot of the GitHub Actions interface. On the left, a list of workflow runs is displayed under the heading "52 workflow runs". The list includes:

- test2 workflow maven** (status: success) - Java CI with Maven #25: Commit [348b739](#) pushed by neojero
- neojero teste GitHub Actions** (status: success) - GitHub Actions Demo #26: Commit [348b739](#) pushed by neojero
- test2 workflow maven** (status: failure) - Java CI with Maven #24: Com...
- neojero teste GitHub Actions** (status: success) - GitHub Actions Demo #25: Co...
- test2 workflow maven** (status: failure) - Java CI with Maven #23: Com...

On the right, an email notification from GitHub is shown. The subject is "[neojero/apiSpringboot2025] Run failed at startup: SonarQube, Attempt #2 - main (670e771)". The email body includes the GitHub logo and the text: "[neojero/apiSpringboot2025] SonarQube workflow run, Attempt #2". Below this, it says "SonarQube, Attempt #2: No jobs were run" and provides a "View workflow run" button. The email is from "neojero <notifications@github.com>" and is dated "jeu. 30 janv. 16:32 (il y a 6 jours)".

PRÉSENTATION DE GITHUB ACTIONS

Conclusion

L'exemple de workflow que vous venez d'ajouter est déclenché chaque fois que du code est poussé vers la branche, et montre comment GitHub Actions peut fonctionner avec le contenu de votre dépôt.

- GitHub fournit des workflows de démarrage préconfiguré que vous pouvez personnaliser pour créer votre propre workflow d'intégration continue.
- GitHub analyse votre code et affiche des workflows de démarrage CI qui pourraient être utiles pour votre référentiel.
- Vous pouvez parcourir la liste complète des workflows de démarrage dans le référentiel actions/starter-workflows.
 - <https://github.com/actions/starter-workflows>