

TD : CI/CD avec GitHub Actions et Clever Cloud

⌚ Objectif du TD

Déployer automatiquement une application React sur Clever Cloud via un pipeline CI/CD avec GitHub Actions.

📋 Prérequis

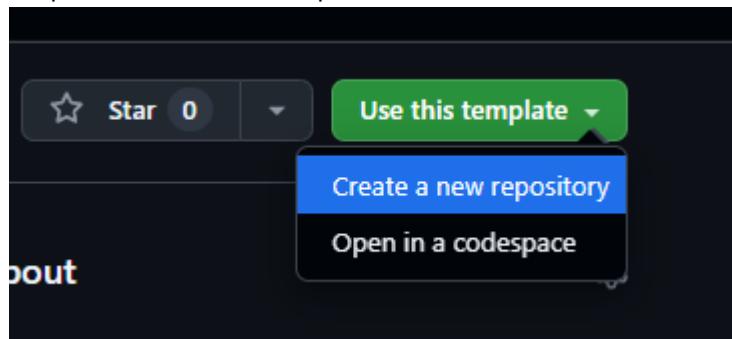
- Compte GitHub personnel
- Compte Clever Cloud (gratuit) avec votre adresse [@Bil.fr](#) : [Clever Cloud Console](#)
- Git installé sur votre machine
- NodeJS installé en local (<https://nodejs.org/fr/download>)

🚀 Étapes du TD

1. Création du projet à partir de MyFITJob

1. Allez sur le repository GitHub du projet <https://github.com/CodinCloud/MyFITJob>

2. Cliquez sur "Use this template" en haut à droite



3. Nommer votre projet pour ne pas le confondre avec le projet MyFITJob original:

Create a new repository Try the new experience

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk ().*

Repository template



CodinCloud/MyFITJob 

Start your repository with a template repository's contents.

Include all branches

Copy all branches from CodinCloud/MyFITJob and not just the default branch.

Owner *



JohnRoger 

Repository name *

MyFITJob.JRR

 MyFITJob.JRR is available.

Great repository names are short and memorable. Need inspiration? How about [symmetrical-fortnight](#) ?

4. Clonez cette nouvelle version (de votre Github personnel) en local :

The screenshot shows the GitHub repository page for 'MyFITJob.V2'. At the top, there are navigation buttons for 'Go to file', 'Add file', and 'Code'. Below these are tabs for 'Local' and 'Codespaces'. The 'Local' tab is active. Under the 'Local' tab, there is a 'Clone' section with three options: 'HTTPS', 'SSH', and 'GitHub CLI'. The 'HTTPS' option is selected and has a 'Copy url to clipboard' button, which is highlighted. The copied URL is 'https://github.com/J0hnRoger/MyFITJob.V2.git'. Below this, there are links for 'Open with GitHub Desktop' and 'Download ZIP'.

```
git clone https://github.com/VOTRE_USERNAME/MyFITJob.git
cd MyFITJob
```

5. Valider que le projet fonctionne:

- Se rendre dans /src/MyFITJob.Frontend

```
npm i  
npm run dev
```

exemple:

```
pwsh C:\Dev\cloud-computing  
cd .\MyFITJob\  
pwsh C:\Dev\cloud-computing\MyFITJob > ↵ main  
git checkout devops  
branch 'devops' set up to track 'origin/devops'.  
Switched to a new branch 'devops'  
pwsh C:\Dev\cloud-computing\MyFITJob > ↵ devops  
cd .\src\MyFITJob.Frontend\  
pwsh C:\Dev\cloud-computing\MyFITJob\src\MyFITJob.Frontend > ↵ devops  
npm i  
  
added 464 packages, and audited 465 packages in 28s  
  
86 packages are looking for funding  
  run `npm fund` for details  
  
found 0 vulnerabilities  
pwsh C:\Dev\cloud-computing\MyFITJob\src\MyFITJob.Frontend > ↵ devops ~1  
npm run dev  
  
> dev  
> vite --port 3000  
  
Generating routes...  
Processed route in 94ms  
  
VITE v6.3.5 ready in 893 ms  
  
→ Local: http://localhost:3000/  
→ Network: http://192.168.1.24:3000/  
→ press h + enter to show help
```

Résultat:

The screenshot shows the MyFITJob application interface. On the left, there's a sidebar with a user icon and the name "MyFirstITJob". The main dashboard has a search bar at the top. Below it are four cards: "Top Skills" (with an error message about skills loading), "Nouvelles offres" (10), "Réponses reçues" (68%), and "Tâches en cours" (53). The central area shows two job offers: "Développeur Full Stack React/Node.js" (Enterprise unknown, Paris, France, 45k-65k €) and "DevOps Engineer" (Expertise in Docker, Kubernetes and CI/CD required). A footer note mentions "TanStack Router".

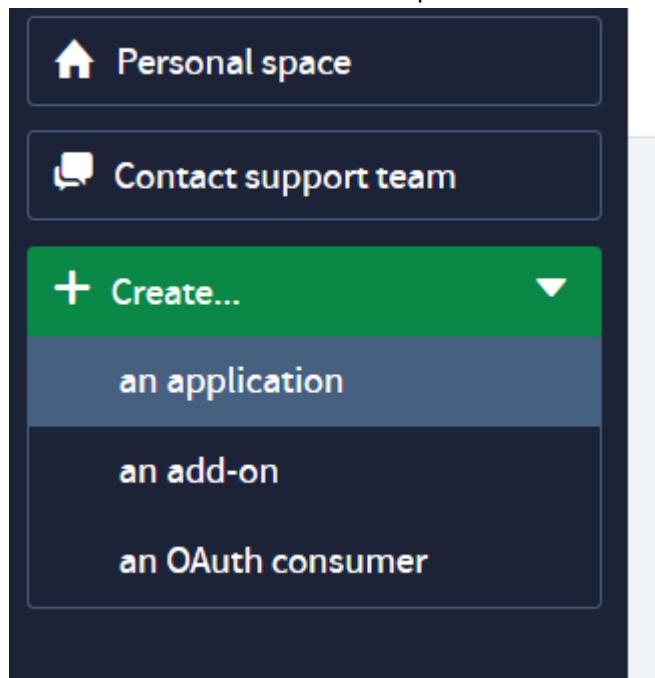
2. Configuration Clever Cloud

1. Créer un compte Clever Cloud

- Allez sur console.clever-cloud.com
- Créez un compte gratuit avec votre email de l'ENI

2. Créer une application

- Dans la console Clever Cloud, cliquez sur "Create an application"



- Choisissez "Create an application from a Github repository"

A screenshot of a step titled 'Create an application from a GitHub repository'. Below it, a sub-instruction says 'If your source code is already hosted on GitHub'. A dropdown menu is open, titled 'Select your GitHub repository'. The menu lists several GitHub repositories: 'MyFit' (highlighted in red), 'CodinCloud', 'MyFITJob', 'JohnRoger', and 'MyFITJob' again at the bottom. The 'MyFITJob' at the bottom is highlighted in white on a dark background.

- Selectionner le repo "fork" à l'étape précédente

- o Sélectionner l'application Docker:

What kind of application is it?

i Learn more about apps

.NET core

Docker

- o Pour la partie Scalabilité : Cliquer sur "Next"
- o Nommez votre application : `myfitjob-frontend`
- o Cliquer sur "Create"
- o Cliquer sur "I don't need any Add-ons"

Do you need a database? an S3-like storage? or a FTP access?

I DON'T NEED ANY ADD-ONS

- o Variables d'env: cliquer sur Next

Environment variables

SIMPLE EXPERT JSON

These variables will be injected as environment variables in the application `MyFITJob.V2`.

Variable name	Variable value
<input type="text"/>	<input type="text"/>

ADD

There are no variables.

RESET CHANGES UPDATE CHANGES

Command line ▾

i Environment variables - Reference ↗

NEXT

Vous devriez avoir l'écran de chargement de l'application:

Application creation

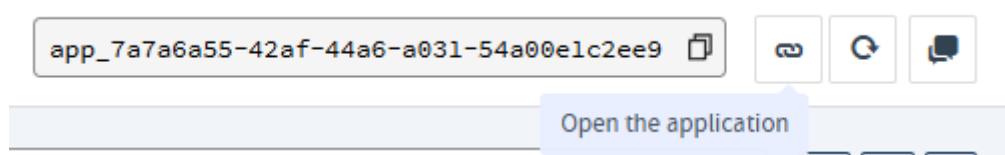
Before pushing your code to Clever Cloud, please note that:

- A file named Dockerfile is required, with "CMD" (this is the command that starts your application).
- The application must listen on port 8080.
- [Documentation about Docker is available!](#)

We are starting your GitHub application...

A ce stade, après quelques minutes, votre application devrait être déployée :

3. Accéder à votre application en ligne - l'icone "Lien" dnas le Header de la page:



The screenshot shows the MyFITJob application interface. On the left, there's a sidebar with a logo for 'MyFirstITJob' and navigation links: Dashboard (selected), Job Offers (2), Contacts, Analytics, and Settings. The main area has a search bar at the top. Below it, there are two cards: 'Top Skills' (with an error message 'Erreur lors du chargement des skills') and 'Nouvelles offres' (showing 10 new offers). Further down, there are sections for 'Nouvelle' (listing a job offer for a 'Développeur Full Stack React/Node.js' in Paris, France, with status 'Indisponible' and salary '45k-65k €') and 'Sauvegardée'. The overall theme is light grey with blue and green accents.

Checkpoint: Vous avez déployer votre code à partir de Clever Cloud, manuellement. Maintenant, il va falloir automatiser ce déploiement à chaque fois qu'on met à jour le code source dans votre repository.

3. Configurer la CI/CD dans Github

J'ai créé le fichier `.github/workflows/ci_cd_clever.yml` qui permet de déployer automatiquement l'application sur Clever Cloud à partir du package fourni par l'éditeur `clever-tools`. Si vous regardez ce fichier, vous verrez des references à des SECRETS (cf cours sur la sécurité).

Les 3 secrets à renseignés dans Github Actions:

- `CLEVER_TOKEN` : Votre token Clever Cloud
- `CLEVER_SECRET` : Votre secret Clever Cloud
- `GH_TOKEN` : le Token PAT de votre compte Github, pour permettre à Clever Cloud de notifier votre Merge Request avec l'URL de l'application déployée.

Pour générer les valeurs de TOKEN et SECRETS, il faut les générer en local à partir de l'utilisataire de clever-cloud:

1. installer l'outil de clever-cloud de manière globale sur votre poste : `npm i -g clever-tools`
2. Lancer la commande `clever login`
3. Saisissez vos identifiants/mdp Vous devriez être redirigé sur une page affichant votre TOKEN et votre SECRET



clever login successful!

You should be automatically logged-in in the CLI now.

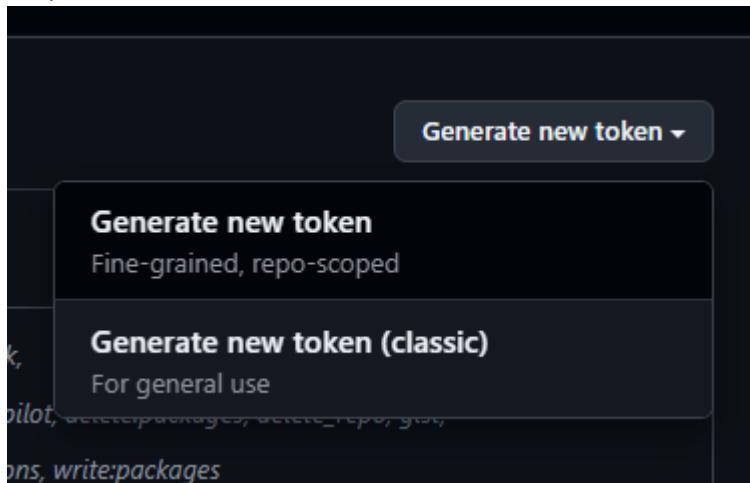
If you're using the clever-tools in a CI/CD environment, you may need to set these tokens in your secrets and expose them as environment variables:

```
CLEVER_TOKEN=601aa8d072eb4c498b2788cd9790d913  
CLEVER_SECRET=gh0esztcbys52L-Z6RKS1UXmpyRpIre1
```

Please note, this token will expire on: Jul 8, 2026, 7:40 PM

4. Noter les valeurs de CLEVER_TOKEN et CLEVER_SECRET, nous en aurons besoin pour la configuration du pipeline sur Github
5. Pour récupérer la valeur du secrets GH_TOKEN, se rendre sur votre interface Github:
<https://github.com/settings/tokens>

6. Cliquer sur "Generate new Token" , et choisir "Generate new Token (fine-grained repo-scoped)":



7. Donner un nom à ce token, par exemple Clever Cloud PR

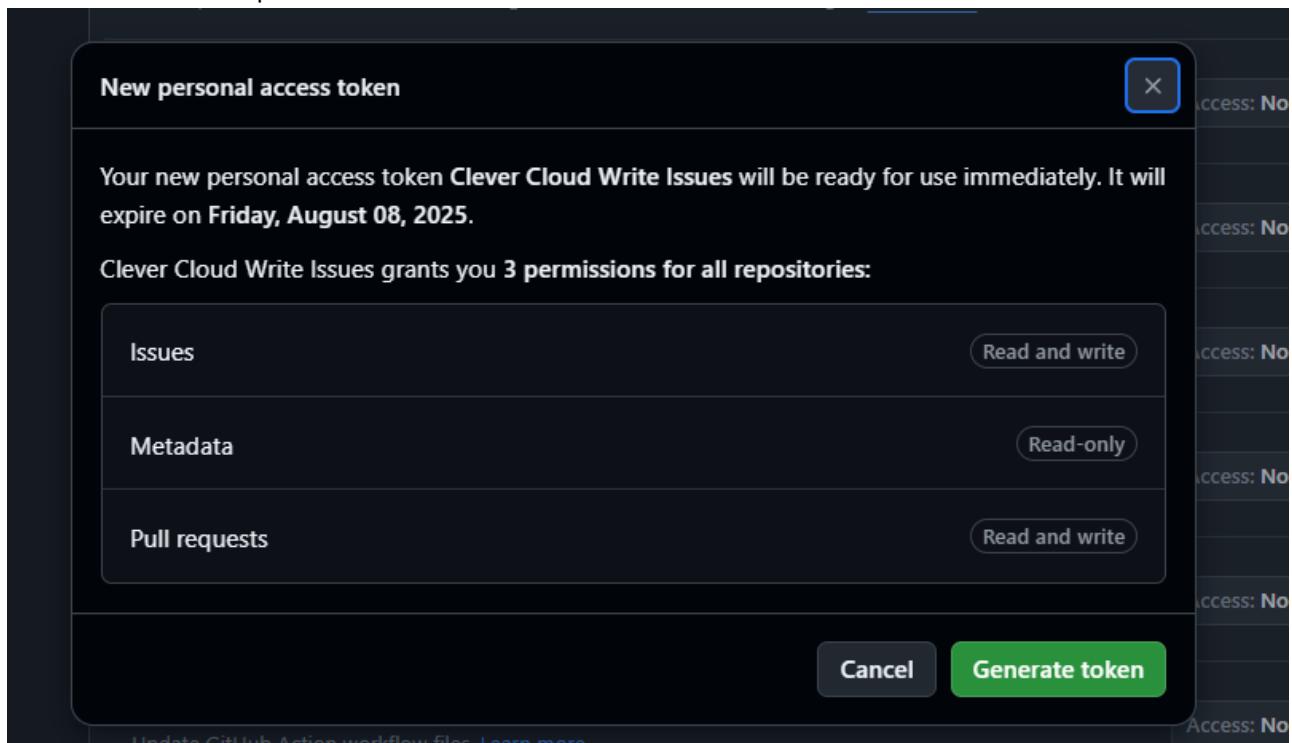
8. COCHER les bons droits à accorder via ce token :

- "Issues" repository permissions (write)
- "Pull requests" repository permissions (write)

The screenshot shows the 'Repository permissions' section of a GitHub repository settings page. It lists several actions with their current access levels:

- Issues** (Selected): Access: Read and write
- Merge queues**: Access: No access
- Metadata** (Mandatory): Access: Read-only
- Pages**: Access: No access
- Pull requests** (Selected): Access: Read and write
- Repository security advisories**: Access: No access

9. Valider et surtout copier le PAT :



Fine-grained personal access tokens

[Generate new token](#)

These are fine-grained, repository-scoped tokens suitable for personal [API](#) use and for using Git over HTTPS.

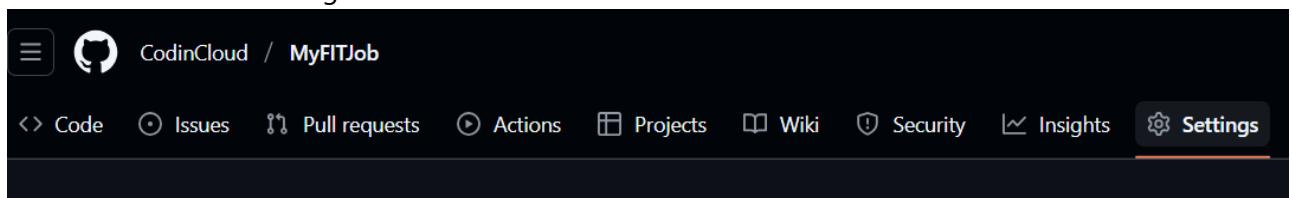
Clever Cloud Write Issues
Never used • Expires on Fri, Aug 8 2025 **Delete**

Make sure to copy your personal access token now as you will not be able to see this again.

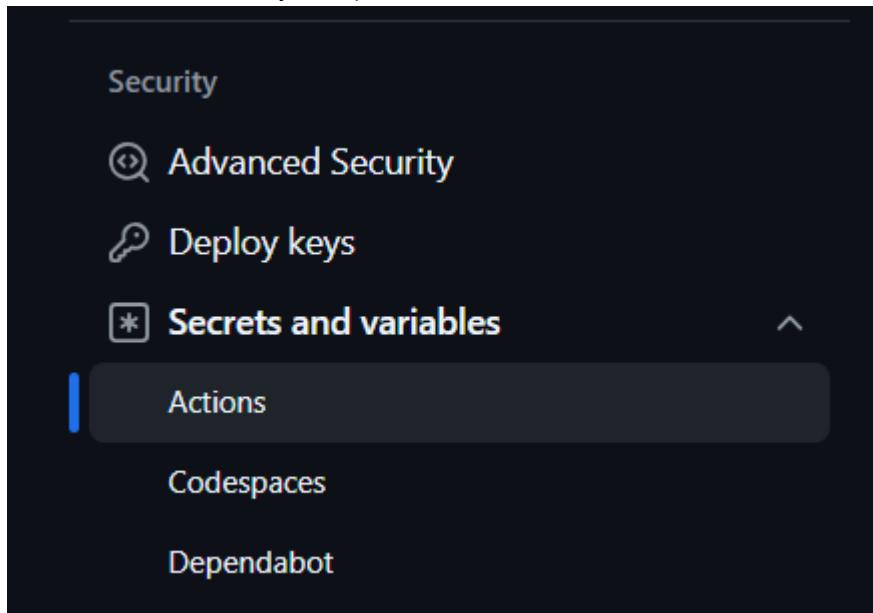
`github_pat_11AAFW66I0LYfdYCzhj7lH_4NsieQ2hDhsEM13hFGdjJJqD0tBX3IMm1mkH7zStIVFRMBTF6TaPU{`

Nous allons renseigner ces 3 **secrets** dans votre repository Github personnel :

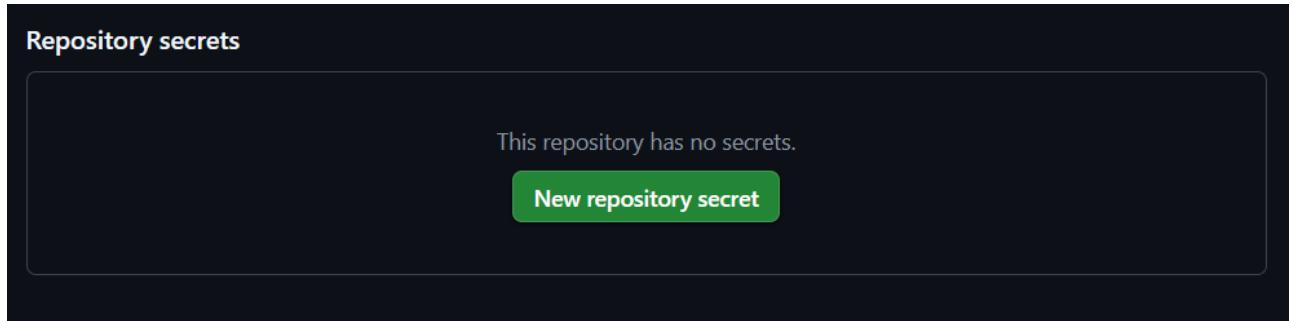
5. Se rendre dans les "Settings" de Github:



6. Dans le menu "Security", cliquer sur "Actions"



7. Cliquer sur "New Repository Secret"



- Ajouter les secrets suivants (attention à ne pas laisser d'espaces vides):
- **CLEVER_TOKEN** : Votre token Clever Cloud
- **CLEVER_SECRET** : Votre secret Clever Cloud
- **GH_TOKEN**: Le PAT généré sur Github ([github_pat_<id>](#))

Une fois les 3 **secrets** renseignés comme suit :

Repository secrets		New repository secret
Name	Last updated	
CLEVER_SECRET	1 hour ago	
CLEVER_TOKEN	1 hour ago	
GH_TOKEN	now	

Nous allons maintenant tester un déploiement manuellement, mais depuis Github:

1. Se rendre dans l'onglet "Actions":

2. Vous devriez arriver sur l'écran de gestion des workflows: notre workflow "CI/CD - Deploy to Clever Cloud" est normalement apparent dans la barre latérale :

The screenshot shows the Clever Cloud Workflow Management interface. On the left, a sidebar titled 'Actions' contains a 'New workflow' button and a 'All workflows' section. Under 'All workflows', the 'CI/CD - Deploy to Clever Cloud' workflow is listed, highlighted with a red arrow pointing to it. The main panel is titled 'All workflows' and 'Showing runs from all workflows'. It displays a message '0 WORKFLOW RUNS' and a blue icon of three interconnected circles with arrows. Below the icon, the text 'There are no workflow runs yet.' is visible.

Vous remarquerez que ce fichier s'appelle pareil que dans le fichier [./github/workflows/ci_cd_clever.yml](#) : Github interprète les fichiers .yml dans ce répertoire comme des flux de déploiement CI/CD

3. Cliquer sur ce workflow, puis sur le bouton vert "Run workflow", à partir de la branche [main](#)

The screenshot shows the 'CI/CD - Deploy to Clever Cloud' workflow page. At the top right, there are filters for 'Event', 'Status', 'Branch', and 'Actor', along with a 'Run workflow' button. A tooltip for the 'Run workflow' button says 'Run workflow from Branch: main'. The main area shows a message 'This workflow has a workflow_dispatch event trigger.' and a blue icon of three interconnected circles with arrows.

4. Le déploiement doit alors se déclencher, vous pouvez cliquer sur ce dernier pour observer chaque étape de la build:

Pour autant, dans le détail de la build, vous pouvez voir qu'une étape n'a pas été exécutées sur cette demande manuelle :

build-and-deploy

succeeded now in 37s

- > Set up job
- > Checkout code
- > Setup Node.js 18
- > Install dependencies
- > Build application
- > Install Clever Tools
- > Create review app
- > Quality Checks 
- > Post Setup Node.js 18
- > Post Checkout code
- > Complete job

CheckPoint : pourquoi cette étape a été by-passée ?

4. Créer une PR pour modifier le Header de l'application

Nous allons modifier le code source pour saisir nos informations dans le Header de l'application afin de bien valider que c'est notre code qui est déployé sur notre application Clever Cloud.



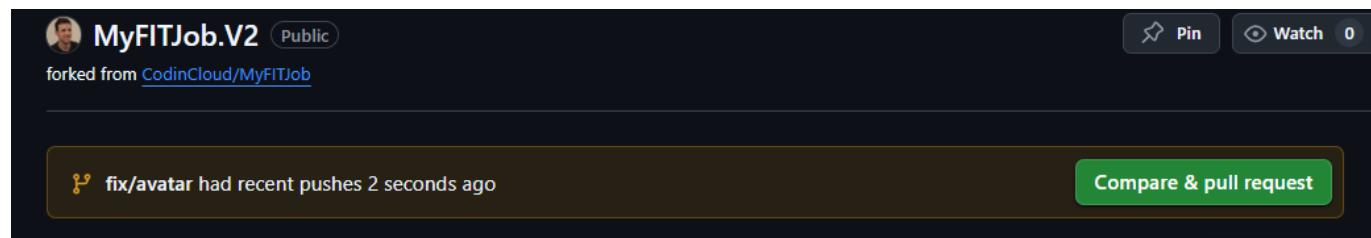
1. Créer une branche local pour effectuer votre fix: `git checkout -b fix/avatar`
2. Dans le code source de votre repo cloné, sur cette nouvelle branche, dans le répertoire du projet Front `./src/MyFITJob.Frontend`, lancer le serveur de développement: `npm run dev`
3. Mettre à jour le fichier `/src/MyFITJob.Frontend/src/components/Header.tsx`, en remplaçant le "TODO" par votre nom (ou un nom fictif, c'est juste pour valider l'hypothèse de déploiement)

exemple:



3. Commit votre changement: `git add . git commit -m "Fix(Header): fix du nom dans le header" git push --set-upstream origin fix/avatar`
4. Se connecter sur votre repository GitHub

Vous devriez voir un bandeau vous annonçant que Github a détecter une nouvelle branche :



5. Cliquer sur "Compare & Pull Request" dans ce bandeau, et compléter la description. Par exemple:

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks. [Learn more about diff comparisons](#)

base repository: CodinCloud/MyFITJob · base: main · head repository: J0hnRoger/MyFITJob.V2 · compare: fix/avatar

Add a title

fix(Header): fix du nom du header

Add a description

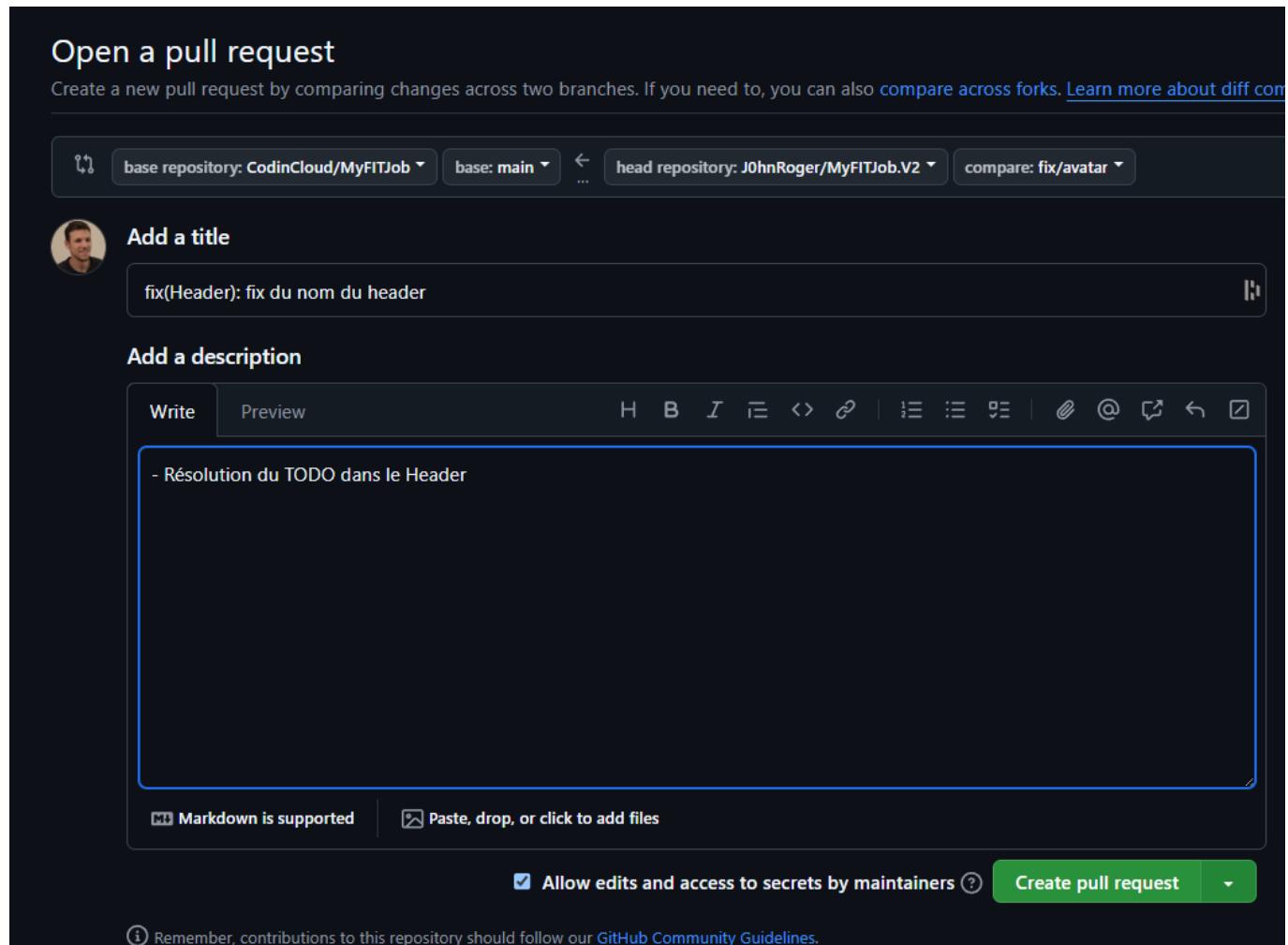
Write Preview

- Résolution du TODO dans le Header

Markdown is supported · Paste, drop, or click to add files

Allow edits and access to secrets by maintainers · Create pull request

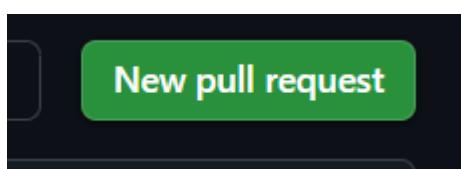
ⓘ Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).



💡 Vous êtes sur le point de créer votre première PR sur un repository

Une Pull Request est une proposition de fusion ("merge") d'une branche de travail vers la branche de référence (souvent main/master) dans un dépôt Git ; elle sert de point central pour la revue de code, la discussion et le déclenchement automatique de la CI/CD.

3. Cliquer sur "Create Pull Request" pour valider votre demande de fusion



4. Un check vous indiquant qu'aucun conflit n'est présent, et surtout : l'intégration continue se lance automatiquement !

fix(Header): fix du nom du header #12

J0hnRoger commented now

- Résolution du TODO dans le Header

fix(Header): fix du nom du header ae05e69

Some checks haven't completed yet
1 in progress check

CI/CD - Deploy to Clever Cloud / build-and-deploy (pull_request) Started now — This check has started...

No conflicts with base branch
Merging can be performed automatically.

Merge pull request You can also merge this with the command line. [View command line instructions.](#)

5. Si tout s'est bien passé, vous devriez avoir un résultat positif, et dans les logs de la CI/CD, de solides tests de qualité/lint et tests unitaires effectués avec succès

J0hnRoger commented 16 minutes ago

- Résolution du TODO dans le Header

fix(Header): fix du nom du header ✓ ae05e69

All checks have passed
1 successful check

CI/CD - Deploy to Clever Cloud / build-and-deploy (pull_request) Successful in 41s

No conflicts with base branch
Merging can be performed automatically.

Merge pull request You can also merge this with the command line. [View command line instructions.](#)

6. Maintenant que la phase d'intégration continue est positive(CI), il ne reste plus qu'à lancer la phase de Continuous Déploiement (CD) : cliquer sur "Merge pull request", puis sur "Confirm Merge" :

fix(Header): fix du nom du header #12

Merged J0hnRoger merged 1 commit into CodinCloud:main from J0hnRoger:fix/avatar now

Conversation 0 Commits 1 Checks 1 Files changed 3

J0hnRoger commented 18 minutes ago

- Résolution du TODO dans le Header

fix(Header): fix du nom du header ae05e69

J0hnRoger merged commit 7014867 into CodinCloud:main now

View details Revert

Pull request successfully merged and closed

You're all set — the fix/avatar branch can be safely deleted. If you wish, you can also delete this fork of CodinCloud/MyFITJob in the settings.

Add a comment

Write Preview

7. Se rendre sur l'onglet "Actions" : le workflow de publication est normalement en train de s'exécuter.

Checkpoint:

Pull request successfully merged and closed

You're all set — the devops branch can be safely deleted.

Delete branch

6. Se rendre dans l'onglet Action - vous devez voir le workflow Github Action tourner :

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

All workflows CI/CD - Deploy to Clever Cloud Management Caches Attestations

All workflows Showing runs from all workflows

3 workflow runs

Merge pull request #11 from CodinCloud/devops CI/CD - Deploy to Clever Cloud #3: Commit 2bbc168 pushed by J0hnRoger main now In progress

5. Vérification finale

1. Accéder à l'application sur Clever Cloud

- Vous devriez voir votre déploiement dans l'onglet "Activity"

Filter apps, add-ons... MyFITJob.JRR Scalability Domain names TCP redirections Environment variables Service dependencies Exposed configuration Activity Logs Metrics Docker for Docker

DEPLOY Successful 30 minutes ago via github hadifew214@fenexy.com Commit ID #a21abfd Logs

DEPLOY Successful 49 minutes ago via github hadifew214@fenexy.com Commit ID #a21abfd Logs

Redeploy this commit Redeploy this commit

- Vérifiez que l'application s'affiche correctement : le même lien qu'en début de TP, dans le Header



- Vérifiez que votre nom apparaît dans le header

Critères de réussite

- Le pipeline GitHub Actions passe (build vert)
- L'application est accessible via l'URL Clever Cloud
- Votre nom apparaît dans le header

Dépannage

Problème : Build échoue

- Vérifiez que tous les secrets sont configurés
- Vérifiez la syntaxe du code TypeScript

Problème : Déploiement échoue

- Vérifiez les tokens Clever Cloud
- Vérifiez l'ID de l'application

Problème : Application ne s'affiche pas

- Vérifiez les logs dans la console Clever Cloud
- Vérifiez que le fichier `.clever.json` est présent

Ressources

- [Documentation Clever Cloud](#)
- [GitHub Actions Documentation](#)
- [MSW Documentation](#)

Félicitations !

Vous avez réussi à mettre en place un pipeline CI/CD complet avec :

- Build automatique avec GitHub Actions
- Déploiement automatique sur Clever Cloud
- Tests et validation automatiques
- Personnalisation de l'application