

TP

Intégration Continue (CI/CD)

Partie 3 : Déploiement continu

Objectif :

Exemple de la mise en œuvre du déploiement continu dans un environnement de production, avec une application type Todo.



- Amine ROUKH
- Nourredine BENDJELLOUL
- Mathis DELEHOUZEE
- Saïd MAHMOUDI

Protocole :

Introduction :

Dans ce TP, on va créer une configuration **DevOps** moderne, extensible, mais simple qui permet de **créer** et de **déployer** rapidement une interface Web (Frontend), un Backend et une base de données. Les principaux objectifs de ce TP sont :

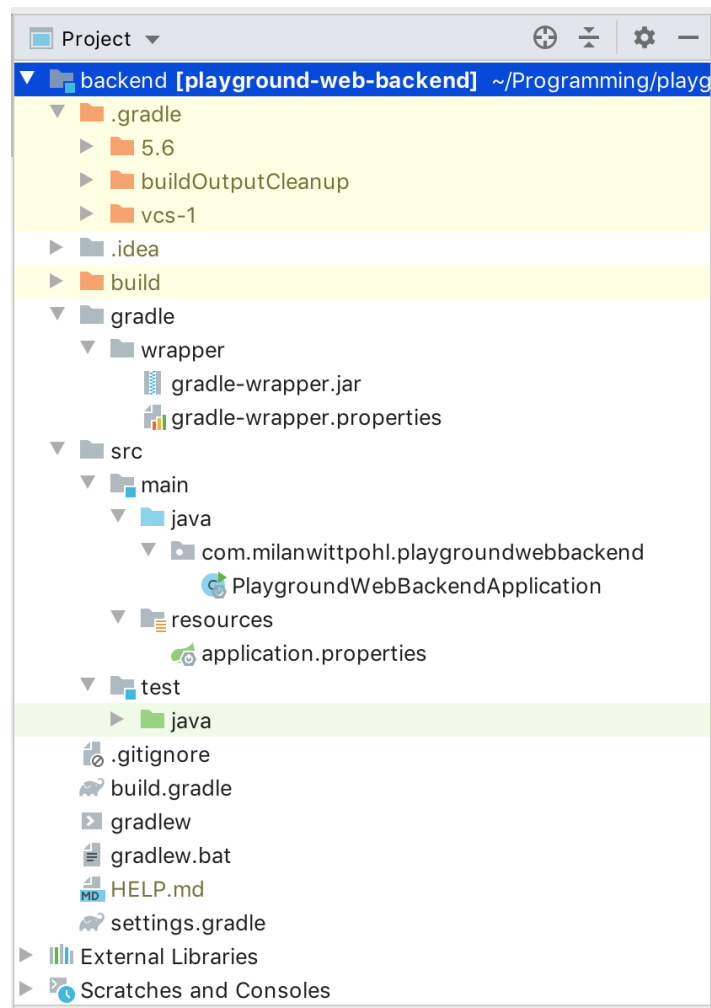
- Mieux comprendre comment créer et déployer des applications,
- Avoir une connaissance de base de chaque composant impliqué dans l'architecture proposée,
- Avoir un code de base qui peut être utilisé pour de futurs projets.

Notre tâche dans ce TP consiste en création d'une simple application **Todo**. C'est une simple application mais elle couvre la plupart des aspects d'une application moderne. Nous devons :

- Connecter une base de données pour stocker des « todos »,
- Travailler avec cette base de données en lisant, créant, mettant à jour et en supprimant des entrées,
- Créer un Backend qui expose une **REST-API** pour notre Frontend,
- Sécuriser correctement notre Backend,
- Créer une interface Frontend qui fonctionne correctement avec les données d'une API,

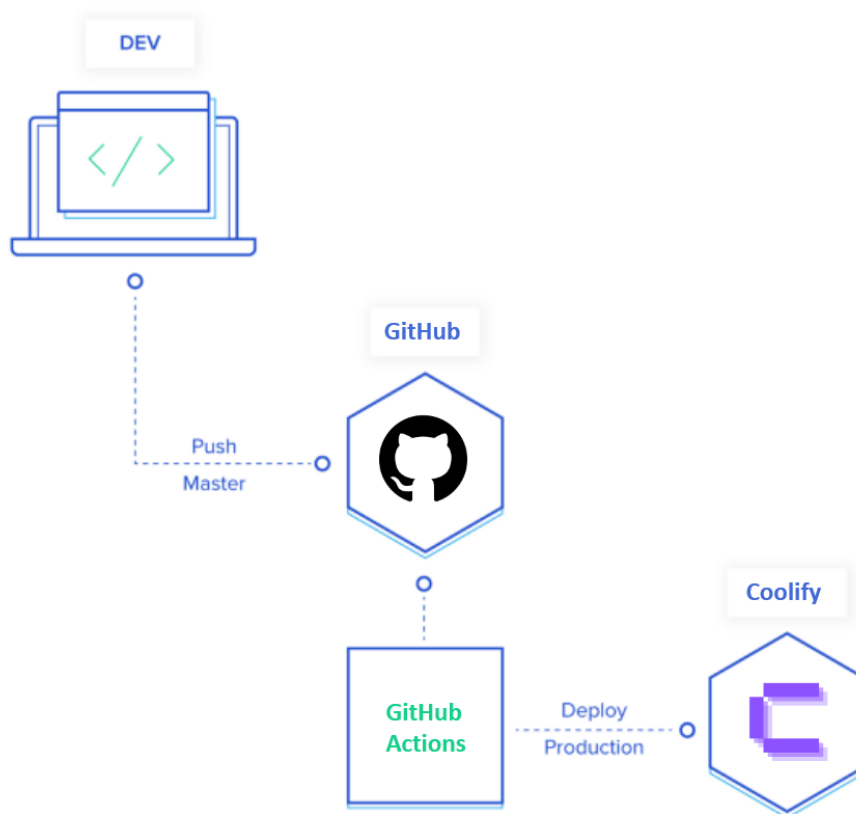
Il existe plusieurs façons de créer cette application Web. Dans ce TP, nous avons choisi les Frameworks suivants :

- Le Backend utilisant **Java** avec **Spring** et **Gradle**, ayant la structure suivante :



- Le Frontend utilisant **VueJS** et **NuxtJS**,
- **MongoDB** pour la gestion des données,
- **Docker** comme conteneur pour l'application,
- Déployer notre application dans le cloud à l'aide du service **Coolify**,
- Automatisation du processus de construction et de déploiement à l'aide de **GitHub Actions**.

Le Figure suivante récapitule le processus de développement et de déploiement relatif à l'application **Todo** proposée :



Exercices :

1. Utilisation de GitHub pour l'auto déploiement :

- Crée un nouveau projet dans votre compte [GitHub](#) et importez le projet « todo-app »


Create a new repository

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 ().*


Owner *

Repository name *

 Amine27


/

todo


 todo is available.

Great repository names are short and memorable. Need inspiration? How about **glowing-disco** ?

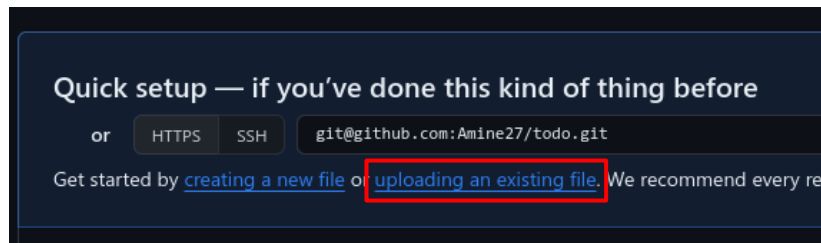
Description (optional)

☒  Public

Anyone on the internet can see this repository. You choose who can commit.

☐  Private

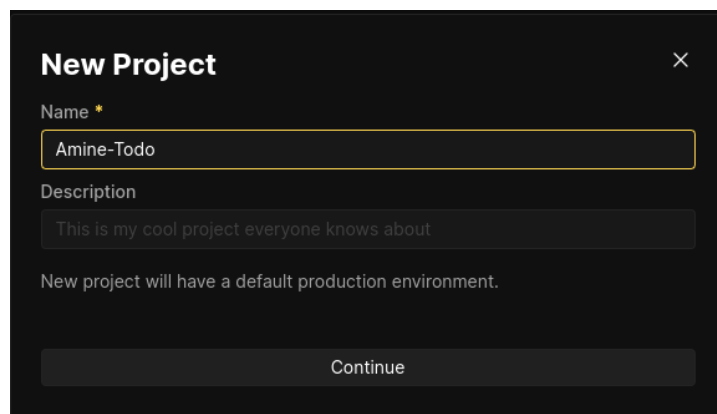
You choose who can see and commit to this repository.



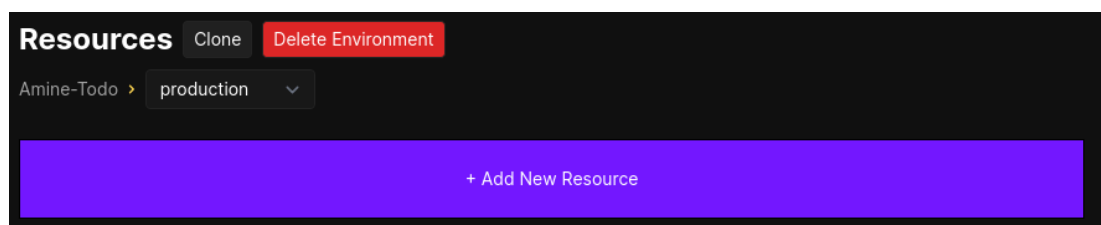
- b. GitHub recherche automatiquement un fichier **.yml** dans le répertoire **.github/workflows/** du projet. S'il est présent, il exécutera un pipeline.

2. Créer les applications de Frontend/Backend sur Coolify :

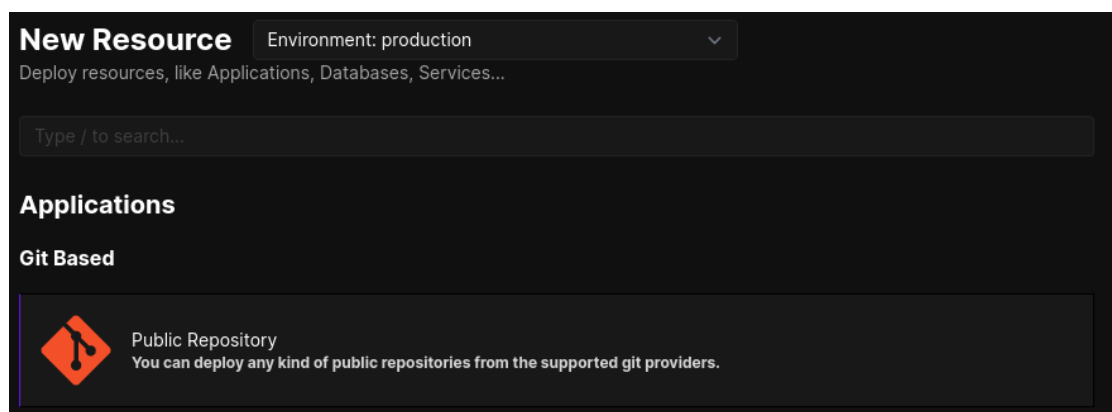
- a. Connectez-vous sur votre compte [Coolify](#) à partir du lien d'invitation.
- b. Naviguez vers la [liste des projets](#) et crée un nouveau projet.



- c. Choisissez le nouveau projet puis cliquez sur « **Ajouter une nouvelle ressource** ».

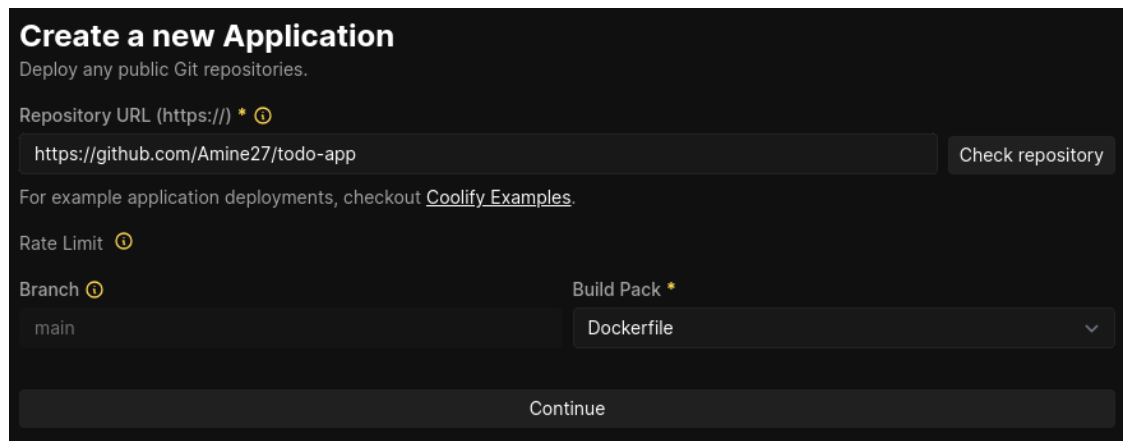


- d. Sélectionnez une application basée sur Git avec un « **Dépôt public** ».



- e. Sélectionnez le serveur « **ilia-server** ».

- f. Entrez le lien vers le dépôt GitHub de votre application, ensuite cliquez sur « **Vérifier le dépôt** » :

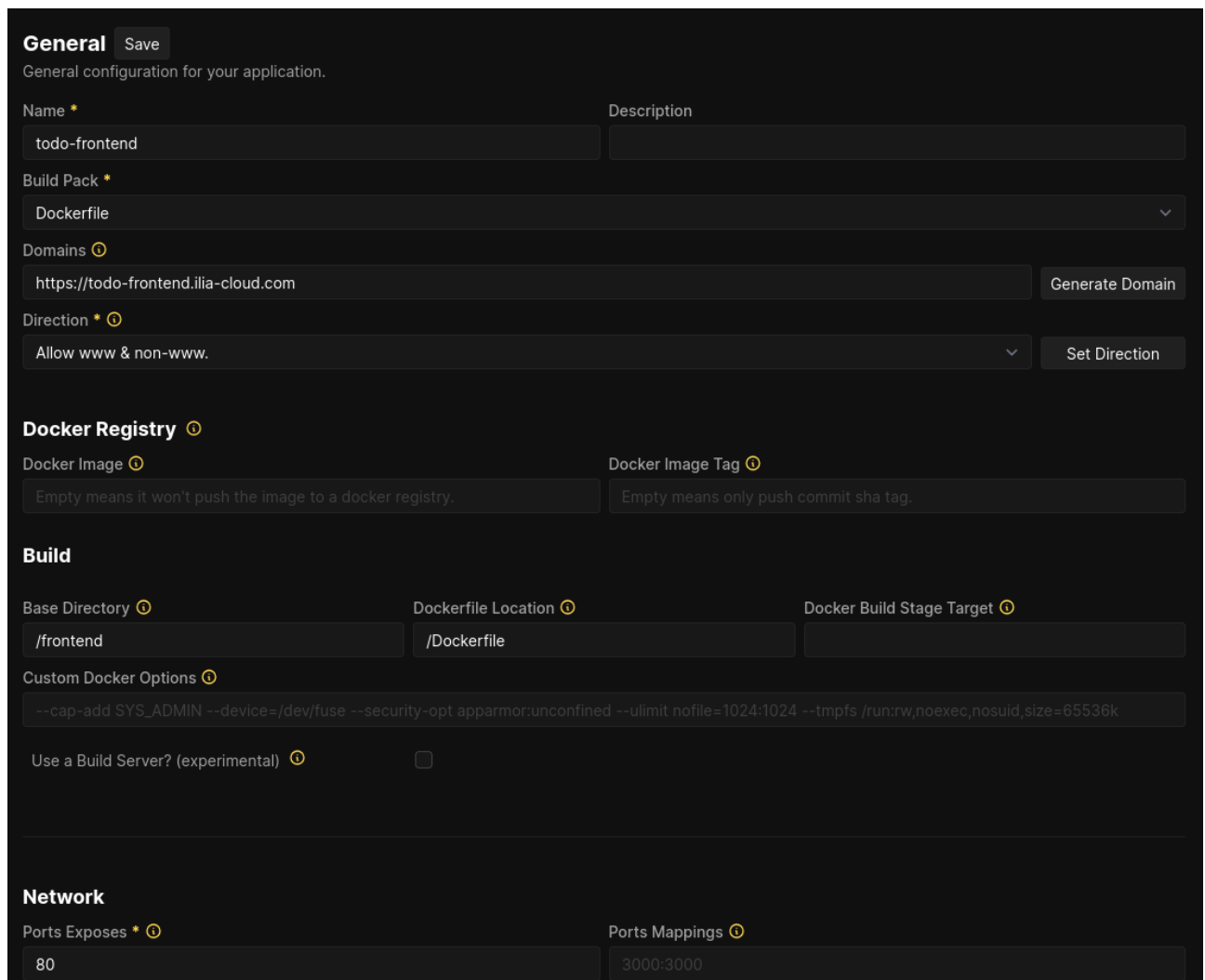


The screenshot shows the 'Create a new Application' form. It has a title 'Create a new Application' and a subtitle 'Deploy any public Git repositories.' Below this is a 'Repository URL (https://)' field with a GitHub icon, containing 'https://github.com/Amine27/todo-app'. To the right is a 'Check repository' button. Below the URL field is a link 'For example application deployments, checkout [Coolify Examples](#).' There is a 'Rate Limit' field with a help icon. Below that are two fields: 'Branch' with 'main' and 'Build Pack' with 'Dockerfile' (indicated by a dropdown arrow). At the bottom is a 'Continue' button.

Choisissez le mode de compilation « **Dockerfile** » puis cliquez sur « **Continue** ».

Le processus de création de l'application est terminé. Maintenant, on va configurer notre application.

- g. Dans la page de configuration générale, adaptez les valeurs comme suit :



The screenshot shows the 'General' configuration page. It has a title 'General' and a 'Save' button. Below the title is the text 'General configuration for your application.' There are two fields: 'Name' with 'todo-frontend' and 'Description' (empty). Below these is a 'Build Pack' dropdown menu with 'Dockerfile' selected. There is a 'Domains' field with 'https://todo-frontend.ilia-cloud.com' and a 'Generate Domain' button. Below that is a 'Direction' dropdown menu with 'Allow www & non-www.' and a 'Set Direction' button. There is a section titled 'Docker Registry' with two fields: 'Docker Image' (empty) and 'Docker Image Tag' (empty). Below this is a section titled 'Build' with three fields: 'Base Directory' with '/frontend', 'Dockerfile Location' with '/Dockerfile', and 'Docker Build Stage Target' (empty). There is a 'Custom Docker Options' field with a long string of options: '--cap-add SYS_ADMIN --device=/dev/fuse --security-opt apparmor:unconfined --ulimit nofile=1024:1024 --tmpfs /run:rw,noexec,nosuid,size=65536k'. Below this is a checkbox 'Use a Build Server? (experimental)' which is unchecked. There is a section titled 'Network' with two fields: 'Ports Exposes' with '80' and 'Ports Mappings' with '3000:3000'.

- h. Donner un nom à votre application, « **todo-frontend** » par exemple. Pour le nom de domaine vous pouvez garder le nom générer automatiquement par Coolify ou bien choisissez un nouveau nom unique. Dans section « Build », spécifier le « Répertoire racine » à « **/frontend** » et « Emplacement Dockerfile » à « **/Dockerfile** ». Dans la section « Réseaux », « Le port exposé » doit être « **80** ».
- i. Sauvegardez la nouvelle configuration.
- j. Pour le **Backend**, allez à nouveau sur votre projet et cliquez sur « Nouveau » et créez une nouvelle application exactement comme l'application précédente, sauf que le « Répertoire racine » doit être « **/backend** ».
- k. Configurer l'application comme suit :

The screenshot shows the 'General' configuration tab for an application named 'todo-backend'. The interface is dark-themed. Under 'General', fields include 'Name' (todo-backend), 'Description' (empty), 'Build Pack' (Dockerfile), 'Domains' (https://todo-backend.ilia-cloud.com), and 'Direction' (Allow www & non-www). There are buttons for 'Generate Domain' and 'Set Direction'. The 'Docker Registry' section has fields for 'Docker Image' and 'Docker Image Tag'. The 'Build' section includes 'Base Directory' (/backend), 'Dockerfile Location' (/Dockerfile), and 'Docker Build Stage Target'. A 'Custom Docker Options' field contains a long command. A checkbox for 'Use a Build Server? (experimental)' is present. The 'Network' section shows 'Ports Exposes' (80) and 'Ports Mappings' (3000:3000).

General Save

General configuration for your application.

Name * Description

todo-backend

Build Pack * Dockerfile

Domains ⓘ

https://todo-backend.ilia-cloud.com Generate Domain

Direction * ⓘ

Allow www & non-www. Set Direction

Docker Registry ⓘ

Docker Image ⓘ Docker Image Tag ⓘ

Empty means it won't push the image to a docker registry. Empty means only push commit sha tag.

Build

Base Directory ⓘ Dockerfile Location ⓘ Docker Build Stage Target ⓘ

/backend /Dockerfile

Custom Docker Options ⓘ

--cap-add SYS_ADMIN --device=/dev/fuse --security-opt apparmor:unconfined --ulimit nofile=1024:1024 --tmpfs /run:rw,noexec,nosuid,size=65536k

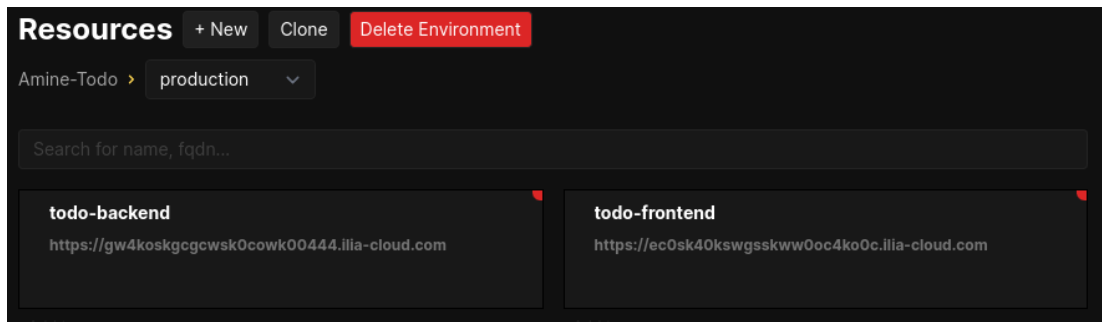
Use a Build Server? (experimental) ⓘ ☐

Network

Ports Exposes * ⓘ Ports Mappings ⓘ

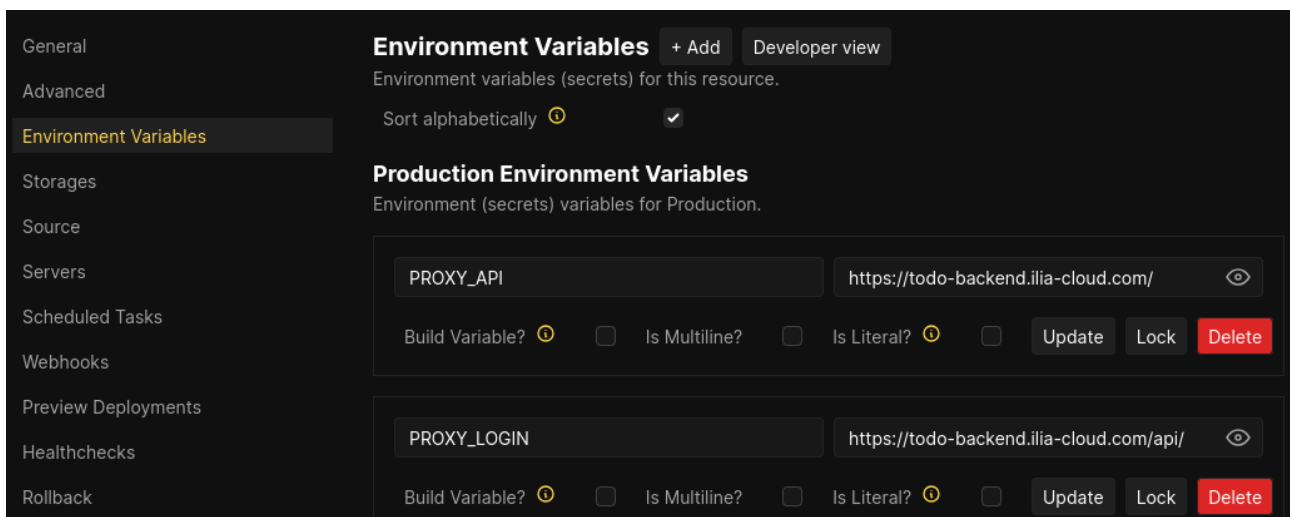
80 3000:3000

- l. Vous aurez à la fin deux applications, une pour le Frontend et une pour le Backend :



- m. Ajout d'informations de proxy : Pour que de notre Frontend fonctionne dans le cloud, nous devons ajouter deux variables de configuration à notre application. Tout d'abord, copiez le nom de domaine de votre application Backend (étape k). Ensuite, cliquez sur l'application **Frontend**. Sous « **Paramètres** », ajoutez deux **variables** de configuration :

- PROXY_API : `https://<NOM_DOMAINE_ BACKEND>.ilia-cloud.com/`
- PROXY_LOGIN : `https://<NOM_DOMAINE_ BACKEND>.ilia-cloud.com/api/`



3. Créer et connecter un cluster Cloud MongoDB :

- a. Pour notre **Backend**, nous aurons également besoin de MongoDB. Nous allons utiliser le service gratuit [MongoDB Atlas](#). Après la création d'un compte gratuit, indiquez le but d'application, le type d'application ainsi que le langage de programmation :

What is your goal today?

Your answer will help us guide you to successfully getting started with MongoDB Atlas.

- ☐ Explore what I can build
- ☒ Build a new application
- ☐ Migrate an existing application
- ☐ Learn MongoDB

What type of application are you building?

Microservices


What is your preferred language?

We'll use this to customize code samples and content we share with you. You can always change this later.

Java

- b. Créez un cluster partagé gratuit. Ensuite, choisissez le fournisseur de cloud par défaut et une région. Pour les autres paramètres, laissez simplement la valeur par défaut. Le démarrage du cluster prendra quelques minutes.

PREVIEW


 **Serverless**

For serverless applications that aren't critical with variable traffic. Minimal configuration required.

- ✓ Pay only for the operations you run
- ✓ Resources scale seamlessly to meet your workload
- ✓ Always-on security and backups

Create

Starting at
\$0.30/1M reads

 **Dedicated**


For production applications with sophisticated workload requirements. Advanced configuration controls.

- ✓ Network isolation and fine-grained access controls
- ✓ On-demand performance advice
- ✓ Multi-region and multi-cloud options available

Create

Starting at
\$0.08/hr*
*estimated cost \$56.94/month

FREE

 **Shared**

For learning and exploring MongoDB in a cloud environment. Basic configuration options.

- ✓ No credit card required to start
- ✓ Explore with sample datasets
- ✓ Upgrade to dedicated clusters for full functionality

Create

Starting at
FREE

- c. La prochaine étape consiste sur la configuration des paramètres de sécurités. Choisissez le mode d'authentification et créez un utilisateur et un mot de passe (à votre choix) que notre Backend utilisera pour se connecter à la base de données.

1 How would you like to authenticate your connection?

Your first user will have permission to read and write any data in your project.

Username and Password

Certificate

Create a database user using a username and password. Users will be given the *read and write to any database privilege* by default. You can update these permissions and/or create additional users later. Ensure these credentials are different to your MongoDB Cloud username and password.

Username

dbUser

Password

Create User

Pour les paramètres de réseau, choisissez le mode d'accès en environnement local et autorisez l'accès de n'importe où (0.0.0.0/0), puis ajouter une adresse IP (Add Entry).

2 Where would you like to connect from?

Enable access for any network(s) that need to read and write data to your cluster.

My Local Environment

Use this to add network IP addresses to the IP Access List. This can be modified at any time.

Cloud Environment

Use this to configure network access between Atlas and your cloud or on-premise environment. Specifically, set up IP Access Lists, Network Peering, and Private Endpoints.

Add entries to your IP Access List

Only an IP address you add to your Access List will be able to connect to your project's clusters.

IP Address	Description		
0.0.0.0/0	Enter description	Add Entry	Add My Current IP Address

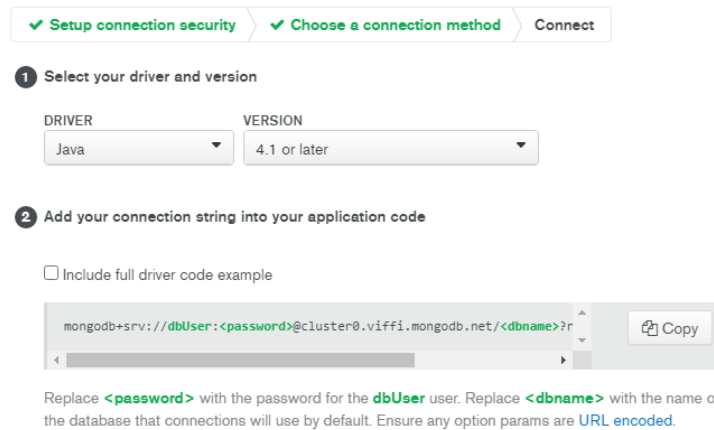
- d. Maintenant, cliquez sur « Terminé et Fermé ». Vous devrez peut-être *attendre quelques minutes pour que les modifications soient déployées avant de pouvoir continuer*.
- e. Une fois déployé, cliquez sur le nom du cluster (Cluster0). En haut, cliquez sur **Collections** puis créez une nouvelle base de données ("Ajouter mes propres

données"). Donnez **todo** comme nom de la base de données, et **todo** comme nom de la collection.

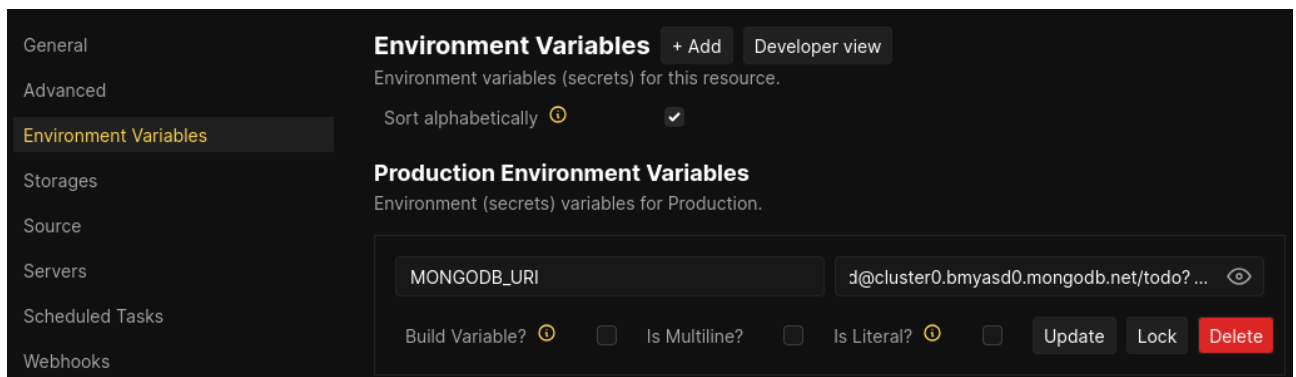
- f. Maintenant, revenez en arrière en cliquant sur **aperçu** et appuyez sur **se connecter** (coin droit).

- g. Sélectionnez « connecter votre application »

- h. Choisissez Java avec la dernière version. Maintenant, copiez la chaîne de connexion.

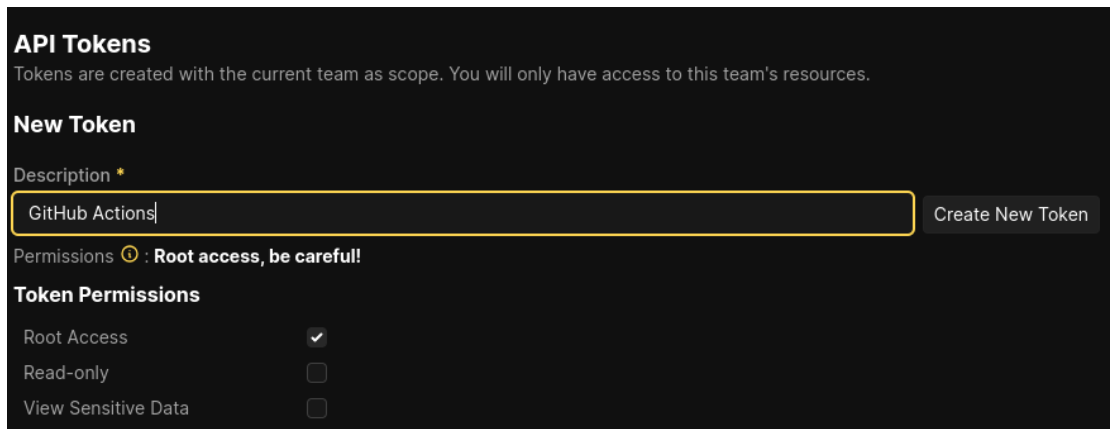


- i. Basculez vers Coolify et ouvrez l'application de **Backend**. Accédez aux paramètres et ajoutez la variable d'environnement MONGODB_URI. Assurez-vous de saisir votre propre nom d'utilisateur et mot de passe tel que saisie dans Atlas Cloud, ainsi que le nom de la base de données.



4. Définition de la clé API Coolify :

- Dans notre fichier **main.yml** qui se trouve dans le dossier `.github/workflows` nous faisons référence à la clé d'API Coolify. Nous ne voudrions pas stocker cette clé dans le fichier pour des raisons de sécurité. C'est pourquoi nous la stockons en tant que variable GitHub.
- Créez et copiez votre clé depuis Coolify, dans « Clés et Tokens ». Cochez « **Accès Root** » :



API Tokens
Tokens are created with the current team as scope. You will only have access to this team's resources.

New Token

Description *
GitHub Actions

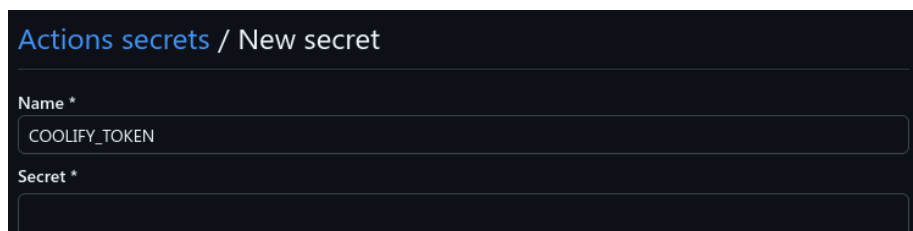
Permissions ⓘ : **Root access, be careful!**

Token Permissions

Root Access	<input checked="" type="checkbox"/>
Read-only	<input type="checkbox"/>
View Sensitive Data	<input type="checkbox"/>

Create New Token

- c. Ensuite, dans GitHub, allez dans les paramètres, puis « secrets et variables », puis dans les Actions créer une nouvelle :



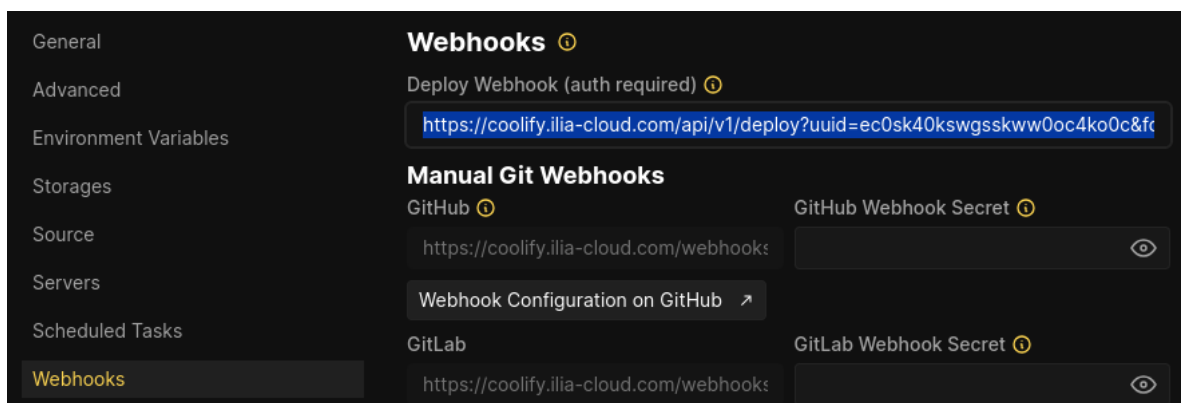
Actions secrets / New secret

Name *
COOLIFY_TOKEN

Secret *

5. Définition des liens Webhook d'applications Coolify :

- a. Le fichier **main.yml** fait référence également aux deux variables **FRONTEND_WEBHOOK** et **BACKEND_WEBHOOK**, ceux deux variables permettent de d'identifier nos applications dans la phase de déploiement.
- b. Pour accéder à ces informations, allez dans vos applications Frontend et Backend dans Coolify, puis dans « Webhook », copiez le premier lien :



General
Advanced
Environment Variables
Storages
Source
Servers
Scheduled Tasks
Webhooks

Webhooks ⓘ

Deploy Webhook (auth required) ⓘ

<https://coolify.ilia-cloud.com/api/v1/deploy?uuid=ec0sk40kswgsskww0oc4ko0c&ft>

Manual Git Webhooks

GitHub ⓘ

<https://coolify.ilia-cloud.com/webhooks>

Webhook Configuration on GitHub ➔

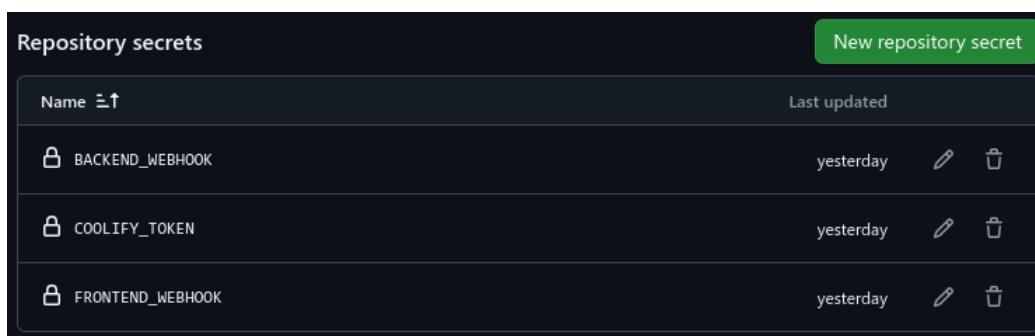
GitLab

<https://coolify.ilia-cloud.com/webhooks>

GitHub Webhook Secret ⓘ

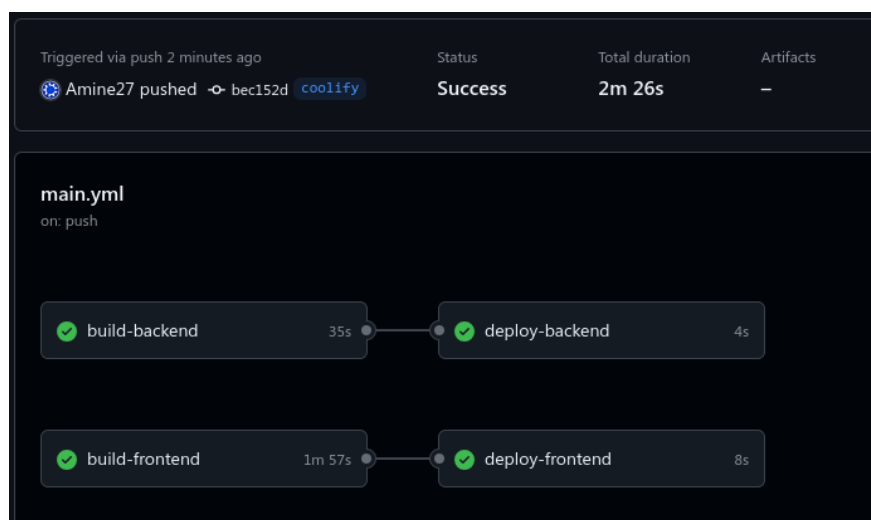
GitLab Webhook Secret ⓘ

- c. Créez deux nouveaux secrets dans les paramètres de votre dépôt GitHub :

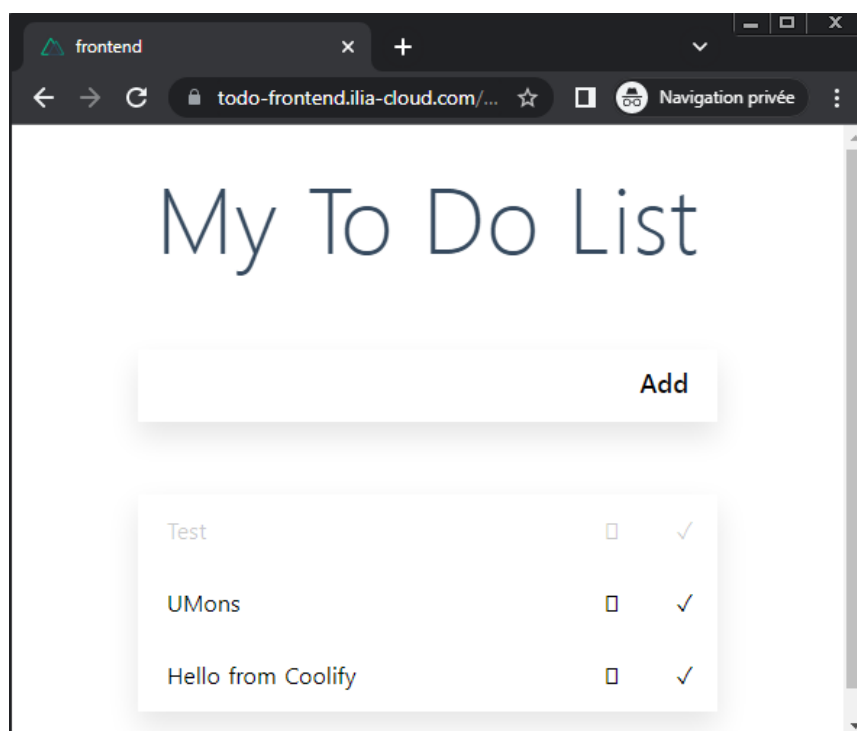


6. Vérification d'auto-déploiement :

- Modifiez un fichier de code source, puis committez les changements à GitHub
- Allez dans Actions et vérifiez que le déploiement s'est bien déroulé :



- Allez vers votre application de Frontend (<https://votreApp.ilia-cloud.com>) :



- Vérifiez que l'application est bien en ligne

- Vérifiez la connexion à votre application avec un mot de passe erroné
 - Connectez à votre application avec les informations : **user / password**
 - Ajoutez, marquez et supprimez des Todo, vérifiez la présence des données après la déconnexion de votre application
 - Allez vers Atlas MongoDB et vérifiez la modification de la base de données.
7. Test du projet en local : pour tester le projet en local, lors de la phase de développement, il y a deux méthodes :
- a. Sans Docker :
- Pour le Backend : il faut installer java et gradle, puis compiler le projet avec :
 - **gradle build**
 - Pour lancer l'exécutable :
 - **java -jar ./build/libs/playground-web-backend-0.0.1-SNAPSHOT.jar**
 - Pour le Frontend : il faut installer NodeJS et npm, puis installer les dépendances :
 - **npm install**
 - Pour lancer l'application :
 - **npm run dev**
 - Aller dans le navigateur Web à l'adresse (<http://localhost:3000>)
 - Pour la base de données : il faut télécharger et installer MongoDB, ensuite crée la collection.
- b. Avec **Docker** :
- Il faut installer docker et docker-compose, et avoir un fichier docker-compose.yml (déjà présent dans le dossier racine du projet). Pour builder les images, exécuter la commande :
 - **docker-compose -f docker-compose-dev.yml build**
 - Pour lancer les conteneurs docker (le Backend, le Frontend et la base de données), exécuter la commande :
 - **docker-compose -f docker-compose-dev.yml up**
8. Le processus de développement actuel inclue les phases de **compilation** et de **déploiement**.
On veut ajouter la phase de **test** dans notre processus :
- a. Modifiez les applications de Backend et Frontend en ajoutant les tests unitaires
 - b. Adaptez le fichier **main.yml** pour appliquer les tests
 - c. Vérifiez le bon fonctionnement des trois phases : compilation, test et déploiement

