# Procédure mise en place application

Outils nécessaires pour tester l’application : git, docker et Node.js

L’application reste relativement simple avec un échange d’api en locale, une à distance et une récupération de données depuis une base de données.

Pour tester par vous-même l’application, vous devez cloner le dépôt distant git suivant https://github.com/Pierre-Perdigues/ci-cd en réalisant cette commande dans un terminal que vous ouvrirez à l’endroit que vous souhaitez.

Pour cela, mettez-vous dans votre dossier, faites un clic droit et cliquez sur « Ouvrir dans le terminal ». Une fois dans le terminal, vous effectuerez cette commande :

git clone <https://github.com/Pierre-Perdigues/ci-cd>

Une fois le code récupéré, vous pouvez lancer l’application en vous rendant dans le dossier ci-cd et en réalisant un docker compose up –build, voici comment faire depuis votre terminal :

Si vous avez encore l’ancien terminal ouvert :  
 cd ci-cd  
 docker compose up –build

Dans un nouveau :  
 cd chemin/vers/cd-cd  
 docker compose up –build

Une fois la construction des différents conteneurs terminée, vous pouvez accéder à l’application en ouvrant sur votre navigateur http://localhost/

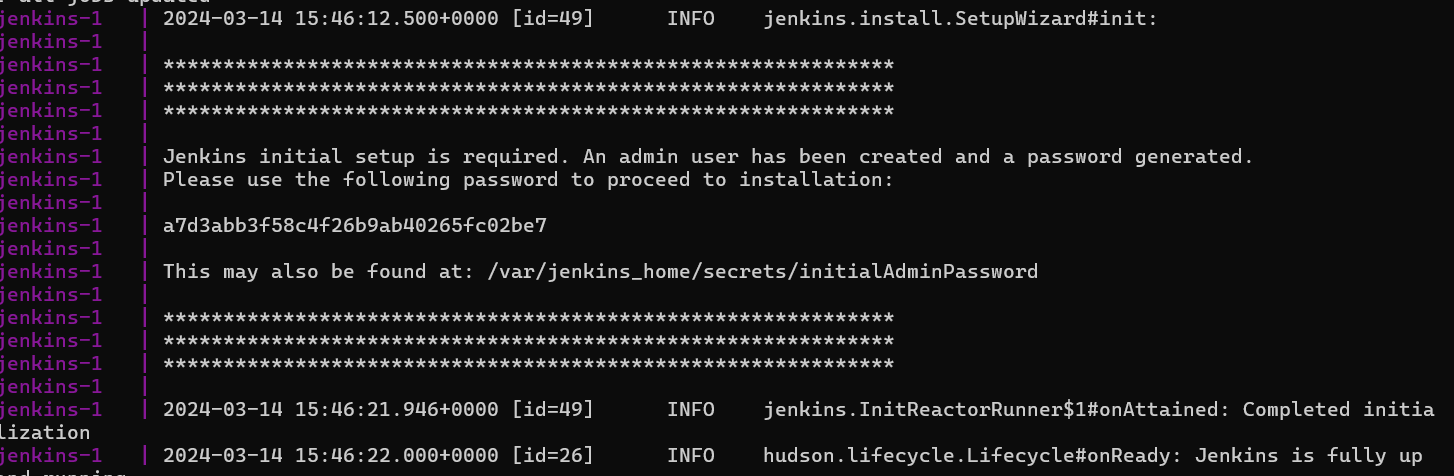
L’application possède deux autres routes http://localhost/citation vous rendant une citation aléatoire de la série Kaamelott et une autre route http://localhost/prenom affichant une liste de 3 prénoms présent dans le conteneur db-1 déployer sur docker.

# Mise en place de jenkins

Dans les documents présents sur dépôt distant, vous aviez un dossier Jenkins, dedans se trouver un Dockerfile permettant de créer un conteneur Jenkins sur lequel nous allons pouvoir paramétrer, en suivant ses instructions, la mise en place d’un pipeline en local.

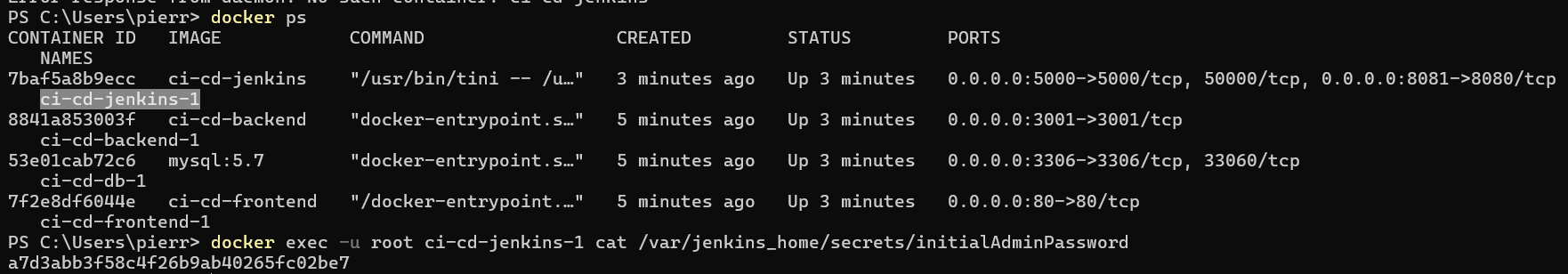
Rendez-vous tout d’abord sur http://localhost:8080

Dans votre terminal, vous avez normalement ceci qui est apparu, copier la suite de caractères au centre et coller là sur la page qui vous le demande sur le http:/localhost :8080 et cliquer sur continuer :



Si ce n’est pas le cas effectuer ceci :  
docker exec -u root nomDuConteneur cat /var/jenkins\_home/secrets/initialAdminPassword

Pour savoir le nom de votre docker vous pouvez effectué un docker ps :



Ensuite faite l’installation suggérés :

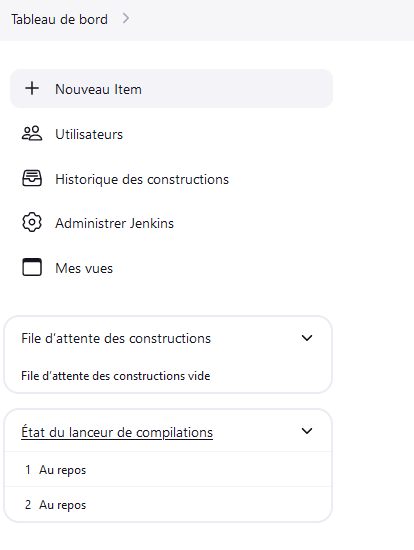
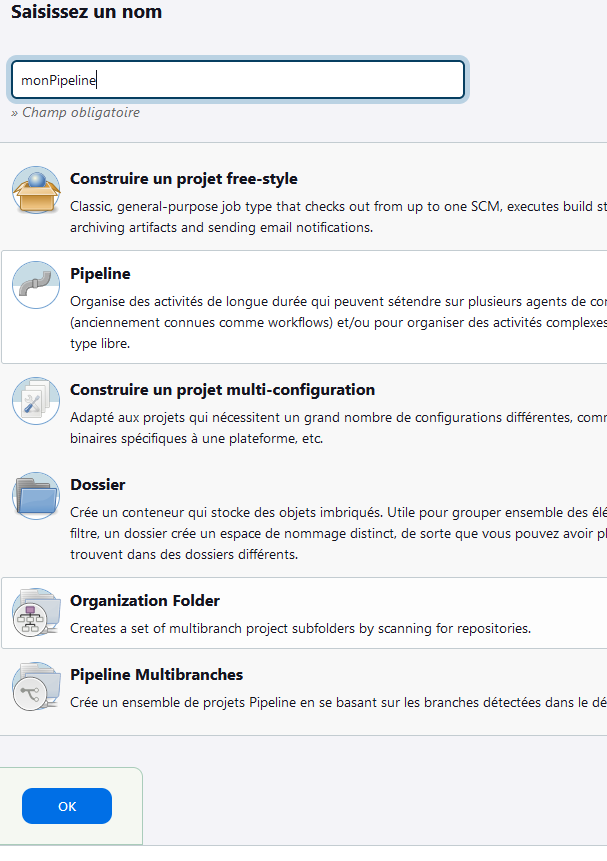


Mettre ici les informations que vous souhaitez pour vous connecter plus tard à Jenkins :



Cliquer ensuite sur Sauver et Terminer et ensuite Commencer à utiliser Jenkins

Une fois l’installation terminée, nous allons créer un pipeline. Pour cela, nous allons cliquer sur nouveau item, ensuite saisir un nom, cliquer sur Pipeline et ensuite sur OK



Dans le dossier que vous avez récupéré se trouve un fichier nommé Jenkinsfile, ouvrez-le et copier-coller le contenu dans l’espace pipeline et cliquez sur sauvegarder :

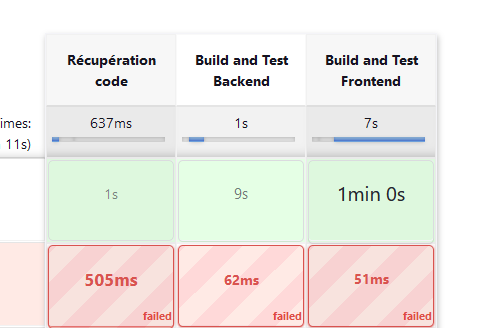


Lancer un build, celui-ci tombe en erreur et c’est normal, nous faisons cela pour initialiser le dossier workspace de Jenkins, maintenant dans un terminal, effectuer cette commande :

docker exec -u root NOM-DE-VOTRE-CONTENEUR -git clone <https://github.com/Pierre-Perdigues/ci-cd> /var/jenkins\_home/workspace/NOM-DE-VOTRE-PIPELINE /ci-cd

docker exec -u root NOM-DE-VOTRE-CONTENEUR chown -R jenkins:jenkins /var/jenkins\_home/workspace/ NOM-DE-VOTRE-PIPELINE /var/jenkins\_home/workspace/ NOM-DE-VOTRE-PIPELINE /ci-cd

Relancer le build et les tests passe



Comme cela à chaque fois qu’une nouvelle version sur la branch main vous pourrez relancer le build dans ce pipeline.

Pour mettre un pipeline en place avec un système automatique il faudrait héberger notre jenkins sur un serveur et configurer le webhook sur le dépôt distant sur Github.

# Commentaires :

L’application :

* L’application est une application React avec un backend express, j’ai choisi ses technologies, car elles sont relativement rapides à mettre en place et aujourd’hui les applications web tout en Javascript commence à prendre une part importante du marché
* Pour réaliser simplement les requêtes SQL, j'ai mis comme package Sequilize, c’est un ORM, un outil permettant de réaliser de manière plus simple et en général plus sécurité les requêtes SQL dans le code source de l’application. J’ai aussi installé axios pour réaliser plus simple les requêtes HTTP
* J’ai utilisé ici MySQL pour la base de données, car c’est une technologie avec laquelle j’ai plus travaillé, mais à noter que les applications en Javascript fonctionne généralement mieux avec des bases de données sous Mongo DB

Docker :

* Le backend et le frontend avec leur propre Dockerfile, celui du backend contient l’installation de node et du lancement de la commande npm install pour récupérer les différents package nécessaire au bon fonctionnement de l’application
* Le Dockerfile du frontend contient lui aussi la même logique avec node mais avec un plus l’utilisation de Nginx pour permettre de faire un serveur sur lequel afficher les pages de l’application
* Le dossier init-db-directory contient un script SQL qui est lancé lors de la création des différents conteneur dans le docker-compose.yml pour s’assurer que la base de données suive la logique de l’application et que surtout qu’elle est bien des données à lire pour lors de l’appel de la route /prenom

Le coté CI CD :

* Nous pourrions relier Sonarqube à Jenkins pour analyser correctement le code, je n’ai malheureusement pas eu le temps de le mettre en place.
* J’ai bien conscience que le système de CI CD réalisé ici est fastidieux et ne respecte pas 100% des « lois » du développement et intégration continus, avec plus de temps, j'aurais sûrement opté pour d’autre solution plus adaptée avec les différentes technologies utilisées dans mon application.
* Toutes les routes ne sont pas testées encore une fois par manque de temps, il faudrait réaliser un test supplémentaire coté backend pour la route /citation en réalisant en mock de la data reçu comme du côté frontend où il manque les tests sur PrenomPage et CitationPage qui nécessiterait aussi ici un mock de l’appel axios permettant de récupérer les données du backend
* Ce TP m'a en tout cas donné envie de plus approfondir le CI CD et aussi m'a fait désormais rechercher et installé des outils de contrôle de code pour mes projets (pas la possibilité d'installer ces outils sur mon poste de travail professionnel au sein de mon alternance) à fin de réaliser un travail plus propre et en déploiement plus proche de la réalité.