Mise en place d'une intégration continue avec Jenkins

Projet MLOps

L'intégration continue (CI) désigne la pratique qui consiste à automatiser l'intégration des changements de code réalisés par plusieurs contributeurs dans un seul et même projet de développement. Il s'agit d'une bonne pratique DevOps principale, permettant aux développeurs de logiciels de merger fréquemment des changements de code dans un dépôt central où les builds et les tests s'exécutent ensuite. Des outils automatisés sont utilisés pour affirmer l'exactitude du nouveau code avant son intégration.

Jenkins est le principal serveur d'automatisation open source qui propose des centaines de plugins qui permettent de construire, de déployer et d'automatiser n'importe quel projet.

Les étapes de mise en place de jenkins dans notre projet MLOps sont les suivantes :

1. Création d'un Dockerfile pour l'utilisation d'une image docker **jenkins:lts** de DockerHub.

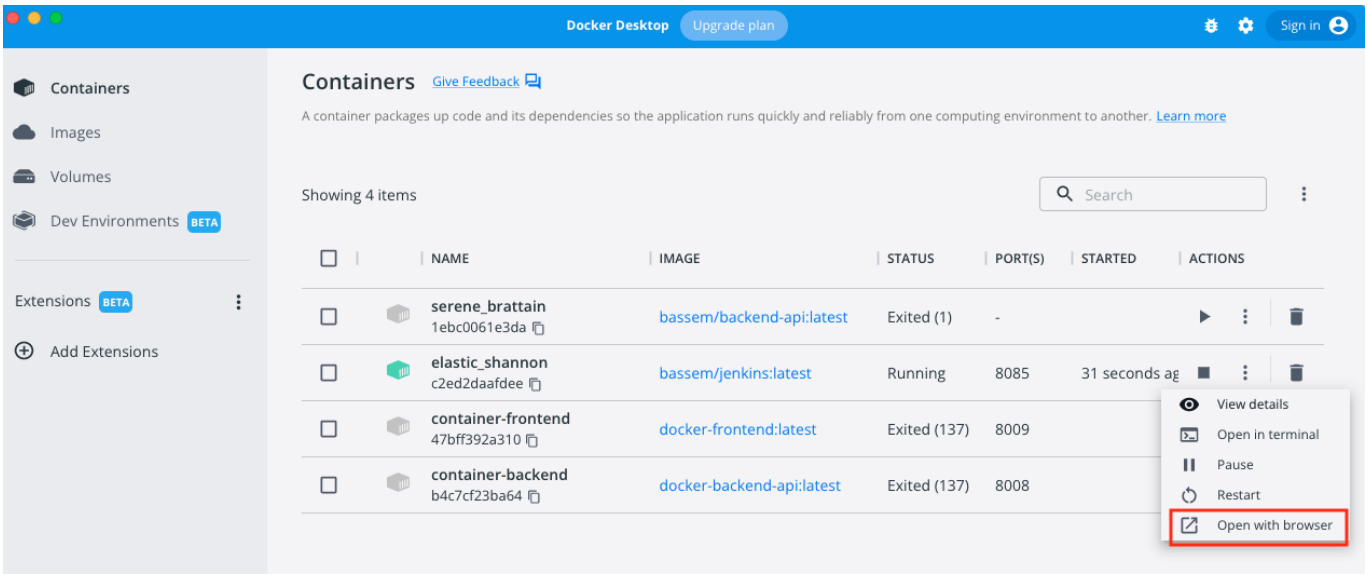
NB. On ajoute l'installateur **pip** pour les bibliothèques de Python.

2. Accéder au répertoire contenant le **Dockerfile**, puis **Run** et **Build** de l'image Docker.

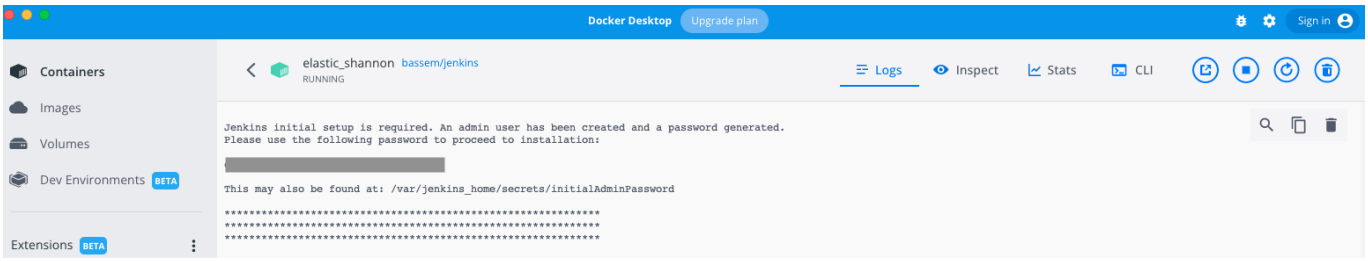
```
$ Docker build -t bassem/jenkins .
```

```
$ docker run -d -p 8085:8080 -v /var/run/docker.sock:/var/run/docker.sock -v
jenkins_home:/var/jenkins_home bassem/jenkins
```

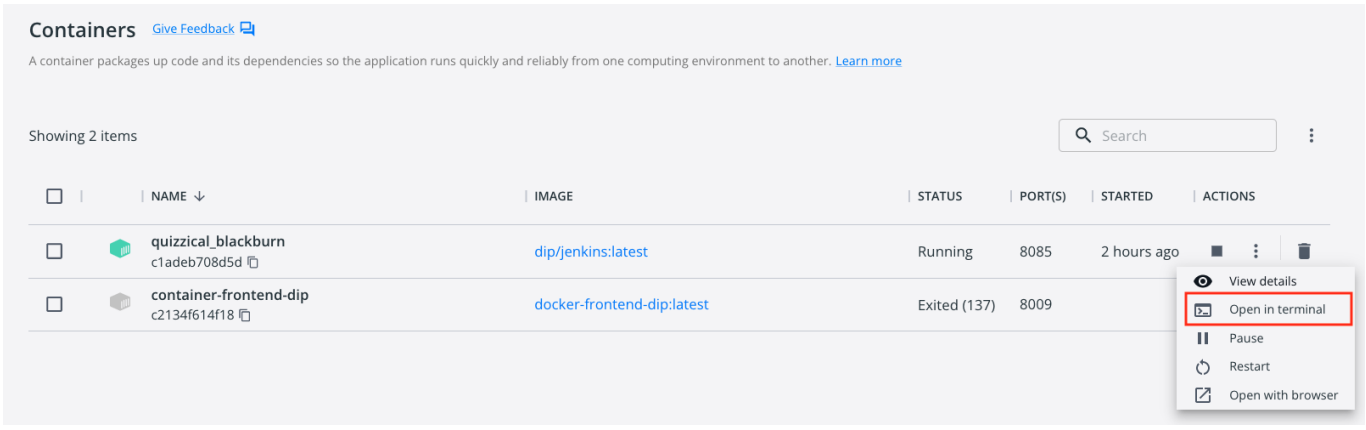
3. Ouvrir le conteneur qu'on a nommé **bassem/jenkins** avec le navigateur.



4. Installation du serveur **Jenkins** suivi d'un accès via mot de passe. On peut récupérer le mot de passe en affichant les détails du conteneur.



5. Si vous avez des problèmes d'accès via le mot de passe Jenkins, on peut modifier l'accès (sans mot de passe). Il suffit d'accèder à Jenkins avec Terminal, puis exécuter les lignes de commandes suivantes




```
$ apt-get update
```

```
$ apt-get install vim
```

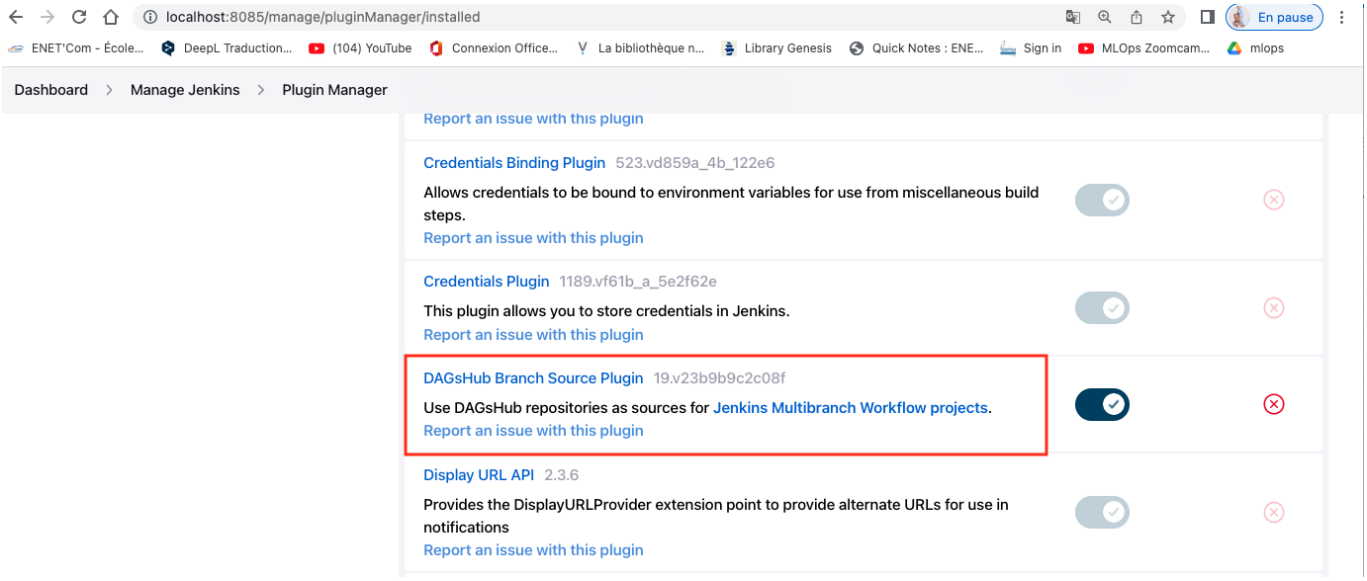
```
$ cd/var/jenkins_home
```

```
$ vim config.xml
```

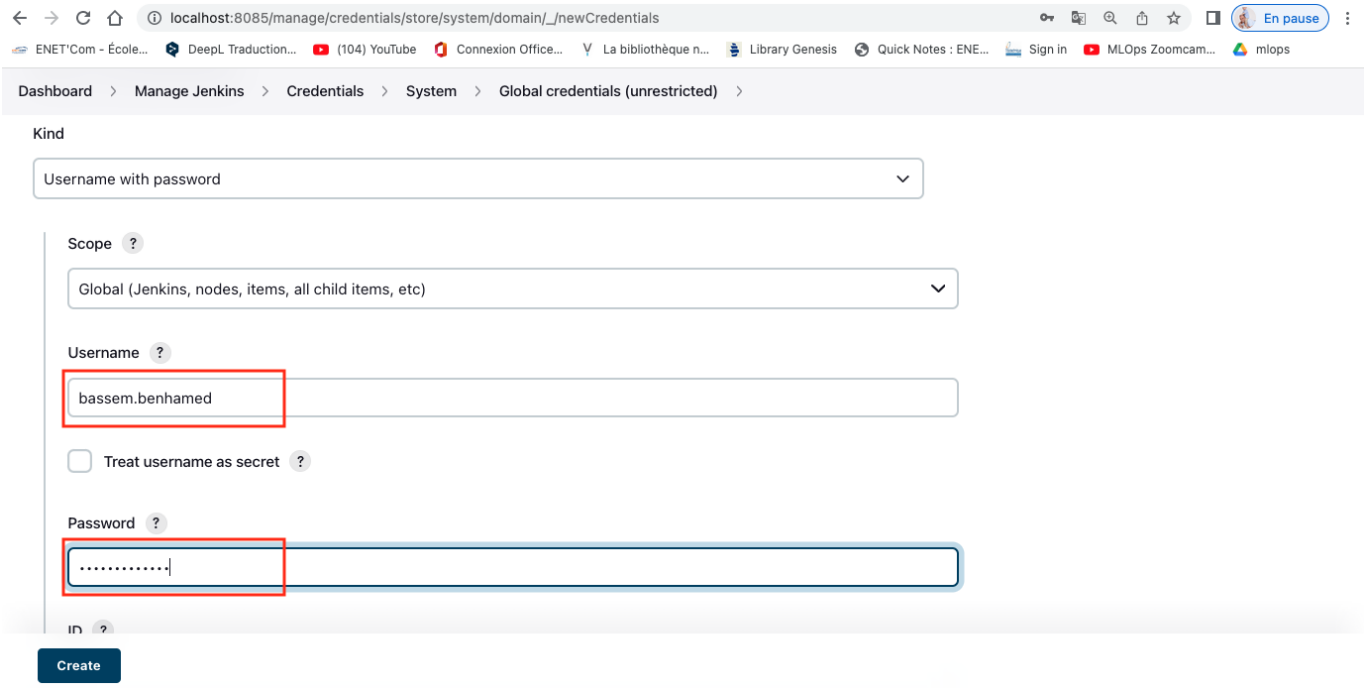
Cliquer sur *i* pour insérer un text, puis modifier le *userSecurity* à **False** et enfin écrire *:wq* pour sauvegarder.

Sinon, passer à l'étape 6.

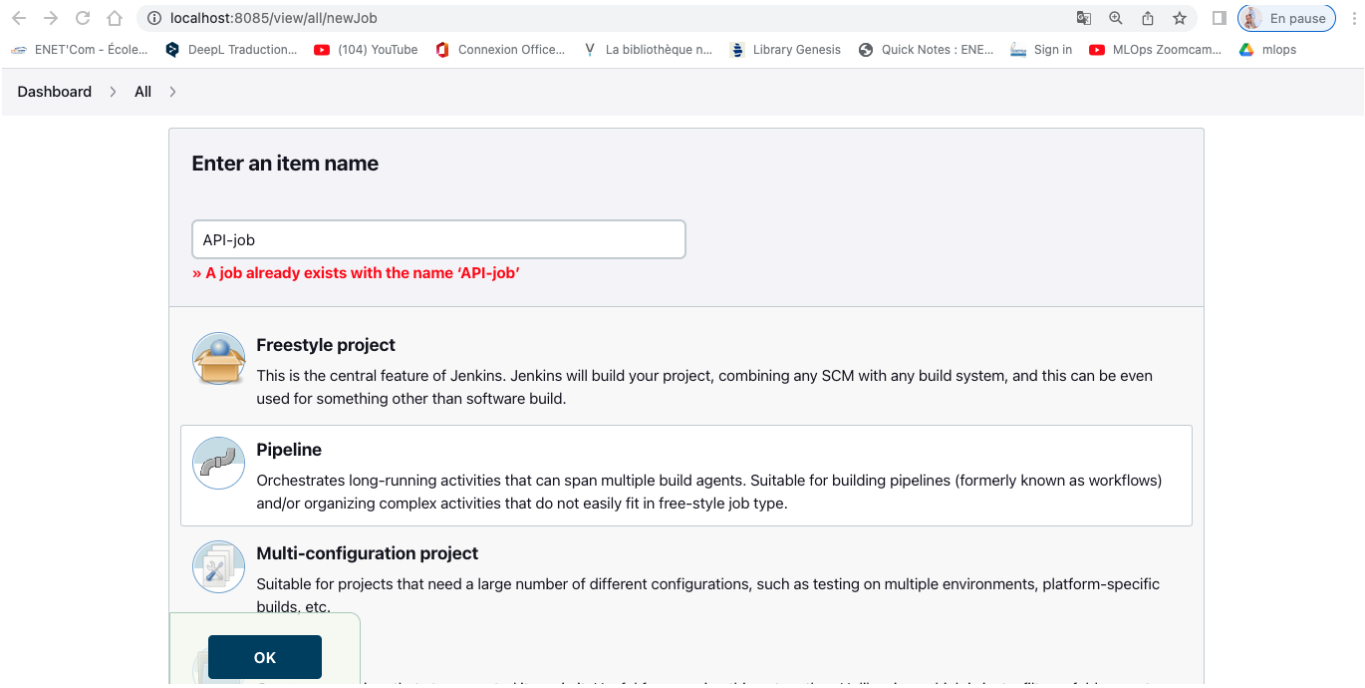
6. Accéder à **Manage Jenkins**, **Manage Plugins** puis **Available** et installer **Dagshub Branch Source Plugin**, **Pipeline Groovy**, **Docker** et **Docker Pipeline**.



7. Accéder à **Manage Jenkins**, **Manage Credentials**, **System** puis **Global Credentials** et ajouter **+addCredentials**.



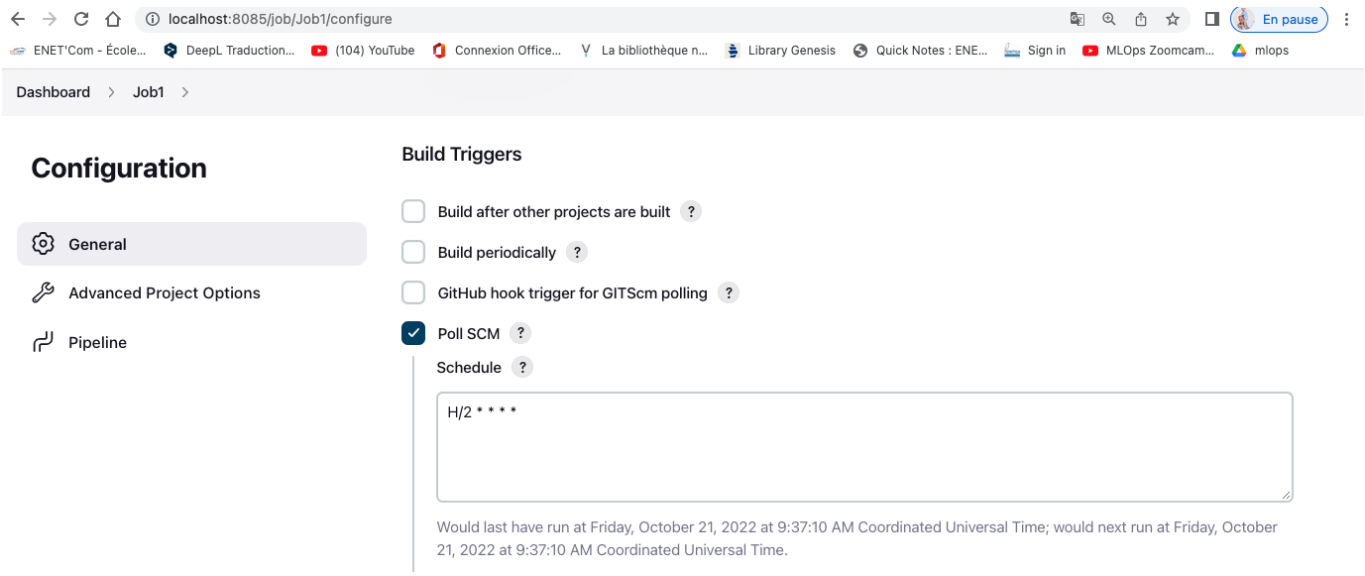
8. Pour créer un nouveau job, choisir **NewItem**, puis nommer le par exemple **API-job** et sélectionner **Pipeline**.



9. Dans la section **Build Triggers**, activer l'option **PollSCM** et ajouter dans **Schedule**

> $ H/2 * * * *

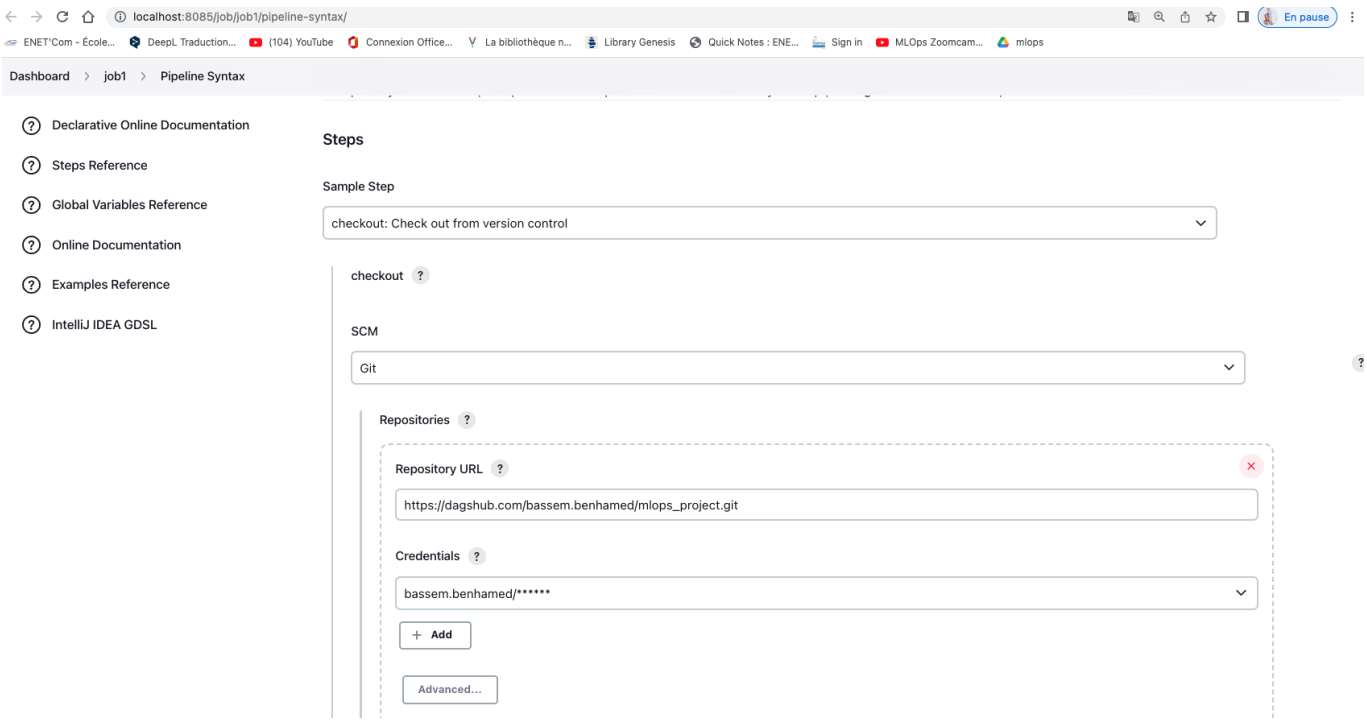
NB. Cela déclenchera le script toutes les 2 minutes.



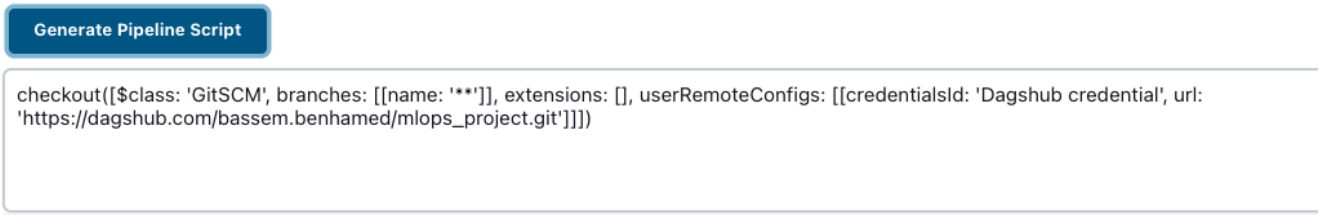
10. Dans la section **Pipeline**, insérer le **Script** suivant

```
pipeline {
    agent any
    environment {
        //once you sign up for Docker hub, use that user_id here
        registry = "bassem/backend-api"
        dockerImage = ''
    }
    //Checkout
    stages {
        stage('Checkout') {
            steps {
            checkout([$class: 'GitSCM', branches: [[name: '**']], extensions: [], userRemoteConfigs: [[credentialsId: 'Dagshub credential',
            url: 'https://dagshub.com/bassem.benhamed/mlops_project.git']]])
        }
        }
    // Test
    /*stage('test') {
      steps{
        script {
            sh "tox"
        }
      }
    }*/
         // Building Docker images
    stage('Building image') {
      steps{
        script {
          dockerImage = docker.build(registry, "./backend")
        }
      }
    }
    }
}
```

11. Dans la section **Steps** de **Pipeline Syntax**, sélectionner l'option **Checkout: Check out from version control**, recopier l'URL de votre repository Dagshub et choisir le **Credentials**



12. Cliquer sur **Generate Pipeline Script**, puis le recopier dans le **Jenkinsfile**



13. **Save**, puis **Build Now**

Pipeline CIJob

Stage View