Créer et transférer des images Docker vers Docker Hub à l'aide de Jenkins Pipeline

Devops

Référence : EF-TEST-TEST

Auteur(s) :

Aissatou KALLOGA

Idriss HAMIDOU

Destinataire(s) :

Easyformer

Date de modification : 31/07/23

Version : 1

1 Installation de Docker sur Ubuntu 20.04

1.1 Docker

1.1.1 Prérequis avant l'installation

Mettre à jour votre liste de packages existante :

```
idriss@idriss-VirtualBox:~$ sudo apt update
```

Installer quelques paquets prérequis qui permettent à apt d'utiliser les paquets sur HTTPS :

```
idriss@idriss-VirtualBox:~$ sudo apt install apt-transport-https ca-certificates curl software-p
roperties-common
```

Ajouter la clé GPG du dépôt officiel de Docker à votre système :

```
idriss@idriss-VirtualBox:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-k
ey add -
OK
```

Ajouter le référentiel Docker aux sources APT :

```
idriss@idriss-VirtualBox:~$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.co
m/linux/ubuntu focal stable"
```

Mettre à jour la base de données des paquets avec les paquets Docker à partir du référentiel qui vient d'être ajouté :

```
idriss@idriss-VirtualBox:~$ sudo apt update
```

S'assurer que vous êtes sur le point d'installer à partir du dépôt Docker et non du dépôt Ubuntu par défaut :

```
idriss@idriss-VirtualBox:~$ apt-cache policy docker-ce
```

1.1.2 Installation de Docker

Pour installer Docker, tapez la commande suivante :

```
idriss@idriss-VirtualBox:~$ sudo apt install docker-ce
```

Après son installation, vérifier qu'il tourne :

```
idriss@idriss-VirtualBox:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
     Active: active (running) since Mon 2023-06-19 19:04:02 CEST; 28s ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 8592 (dockerd)
      Tasks: 8
     Memory: 25.7M
     CGroup: /system.slice/docker.service
             └─8592 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

juin 19 19:04:01 idriss-VirtualBox systemd[1]: Starting Docker Application Container Engine...
juin 19 19:04:01 idriss-VirtualBox dockerd[8592]: time="2023-06-19T19:04:01.876424937+02:00" le>
juin 19 19:04:01 idriss-VirtualBox dockerd[8592]: time="2023-06-19T19:04:01.877810298+02:00" le>
juin 19 19:04:01 idriss-VirtualBox dockerd[8592]: time="2023-06-19T19:04:01.991822164+02:00" le>
juin 19 19:04:02 idriss-VirtualBox dockerd[8592]: time="2023-06-19T19:04:02.202143230+02:00" le>
juin 19 19:04:02 idriss-VirtualBox dockerd[8592]: time="2023-06-19T19:04:02.257439122+02:00" le>
juin 19 19:04:02 idriss-VirtualBox dockerd[8592]: time="2023-06-19T19:04:02.257513382+02:00" le>
juin 19 19:04:02 idriss-VirtualBox dockerd[8592]: time="2023-06-19T19:04:02.291385112+02:00" le>
juin 19 19:04:02 idriss-VirtualBox systemd[1]: Started Docker Application Container Engine.
lines 1-20/20 (END)
```

Pour vérifier si vous pouvez accéder et télécharger des images de Docker Hub, tapez :

```
idriss@idriss-VirtualBox:~$ sudo su
```

```
root@idriss-VirtualBox:/home/idriss# docker run hello-world
Unable to find image 'hello-world:latest' locally
```

Exécuter la commande suivante pour télécharger l'image officielle d'ubuntu sur votre ordinateur

```
root@idriss-VirtualBox:/home/idriss# docker pull ubuntu
```

Pour voir les images qui ont été téléchargées sur votre ordinateur, tapez :

Vous devriez voir apparaitre hello-world

```
root@idriss-VirtualBox:/home/idriss# docker images
REPOSITORY    TAG       IMAGE ID       CREATED       SIZE
ubuntu        latest    ^9284ca6cea0   2 weeks ago   77.8MB
hello-world   latest    9c7a54a9a43c   6 weeks ago   13.3kB
```

L'installation de Docker est terminée.

2 Installation de Jenkins sur Ubuntu 20.04

2.1 Jenkins

2.1.1 Installation

Jenkins peut être installé sur Ubuntu 20.04 en ajoutant les clés du référentiel au système, mais avant cela, nous devons d'abord installer Java Development Kit. Installer OpenJDK par la communauté open-source de Java s'il n'est pas encore installé sur votre système Ubuntu 20.04.

2.1.2 Installer OpenJDK

La dernière version stable d'OpenJDK peut être installée à partir du référentiel de packages officiel Ubuntu. La dernière version stable du kit de développement Open Java est OpenJDK 11.

Mettre à jour le référentiel de cache APT du système :

```
idriss@idriss-VirtualBox:~$ sudo apt update
```

Taper la commande ci-dessous pour installer OpenJDK 11 :

```
idriss@idriss-VirtualBox:~$ sudo apt install openjdk-11-jdk
```

Une fois le processus d'installation terminé, la version de Java peut être vérifiée en tapant la commande ci-dessous :

```
idriss@idriss-VirtualBox:~$ java --version
```

À présent, Jenkins peut facilement être installé sur Ubuntu en important et en ajoutant les clés GPG au système.

Ajouter des clés GPG :

```
idriss@idriss-VirtualBox:~$ wget -p -O - https://pkg.jenkins.io/debian/jenkins.i
o.key | sudo apt-key add -
```

Après avoir ajouté les clés GPG, ajouter l'adresse du package Jenkins à la liste des sources en tapant la commande ci-dessous :


```
idriss@idriss-VirtualBox:~$ sudo sh -c 'echo deb http://pkg.jenkins.io/debian-st
able binary/ > /etc/apt/sources.list.d/jenkins.list'
```

Après avoir activé le référentiel Jenkins, mettre simplement à jour le cache APT du système une fois.

```
idriss@idriss-VirtualBox:~$ sudo apt update
```

Taper la commande ci-dessous pour installer Jenkins :

```
idriss@idriss-VirtualBox:~$ sudo apt install jenkins
```

Maintenant que Jenkins a été installé avec succès, nous allons procéder au démarrage et à la configuration de celui-ci.

2.1.3 Démarrer le serveur Jenkins

Le service Jenkins devrait démarrer automatiquement lors de l'installation de Jenkins. Pour vérifier l'état du service Jenkins, taper la commande ci-dessous :

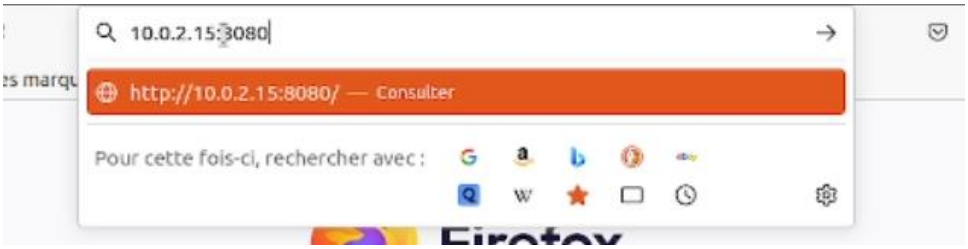
```
idriss@idriss-VirtualBox:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor prese>
     Active: active (running) since Mon 2023-06-19 20:12:32 CEST; 23s ago
   Main PID: 7924 (java)
      Tasks: 43 (limit: 4614)
     Memory: 377.6M
     CGroup: /system.slice/jenkins.service
             └─7924 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java>
```

Si le service Jenkins n'est pas actif, taper la commande ci-dessous :

```
idriss@idriss-VirtualBox:~$ sudo systemctl start jenkins
```

2.1.4 Configurer Jenkins

Pour configurer Jenkins, taper votre nom de domaine ou votre adresse IP avec le port 8080 dans la barre d'adresse du navigateur, et vous devriez avoir la page Déverrouiller Jenkins demandant un mot de passe, comme l'image ci-dessous.



Vous pouvez obtenir le mot de passe à partir de l'emplacement indiqué à l'aide de la commande cat dans le terminal :

Sélectionner "Installer les plugins suggérés" :

Après l'installation des plugins, définir le nom d'utilisateur, le mot de passe et l'adresse e-mail de l'utilisateur administrateur.

Ensuite, il vous dirigera vers une page de configuration de l'URL Jenkins.

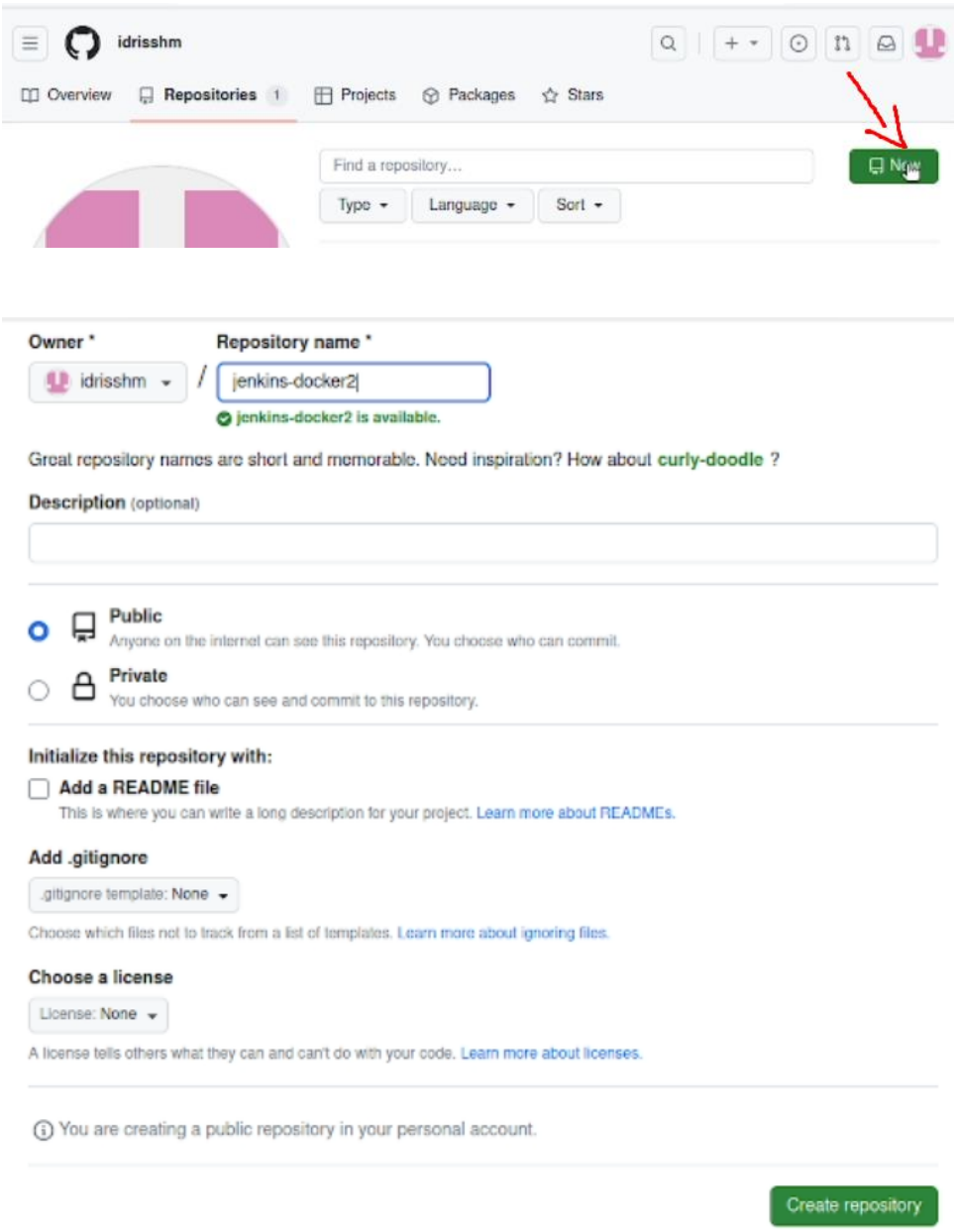
À la fin de la configuration de Jenkins, la page web affichera le message : "Jenkins est prêt !"

Maintenant que Jenkins est prêt à être utiliser, nous allons créer les fichiers qui nous permettrons de pousser l'image Docker à l'aide de GitHub. Il faut donc créer un compte Github si vous n'en avez pas.

3 Création d'un nouveau référentiel GitHub

3.1 Nouveau référentiel

Connectez-vous à votre compte GitHub et créez un nouveau référentiel comme indiqué ci-dessous :



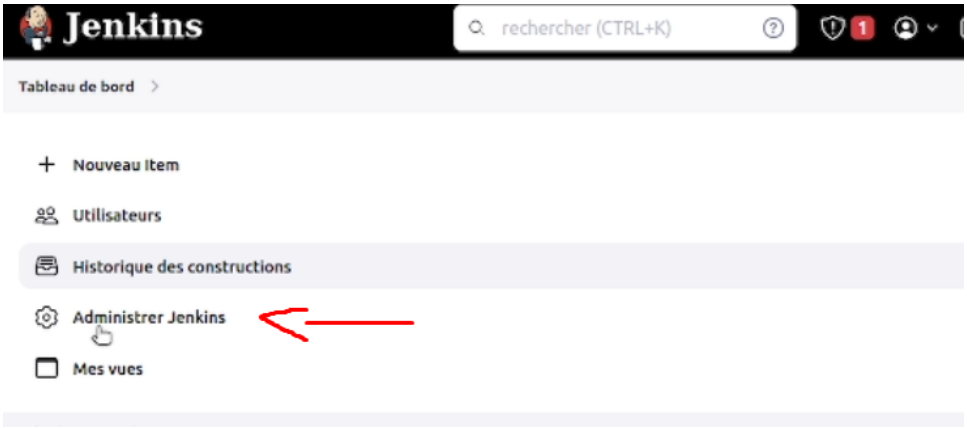
Copier ce lien car on en aura besoin plus tard

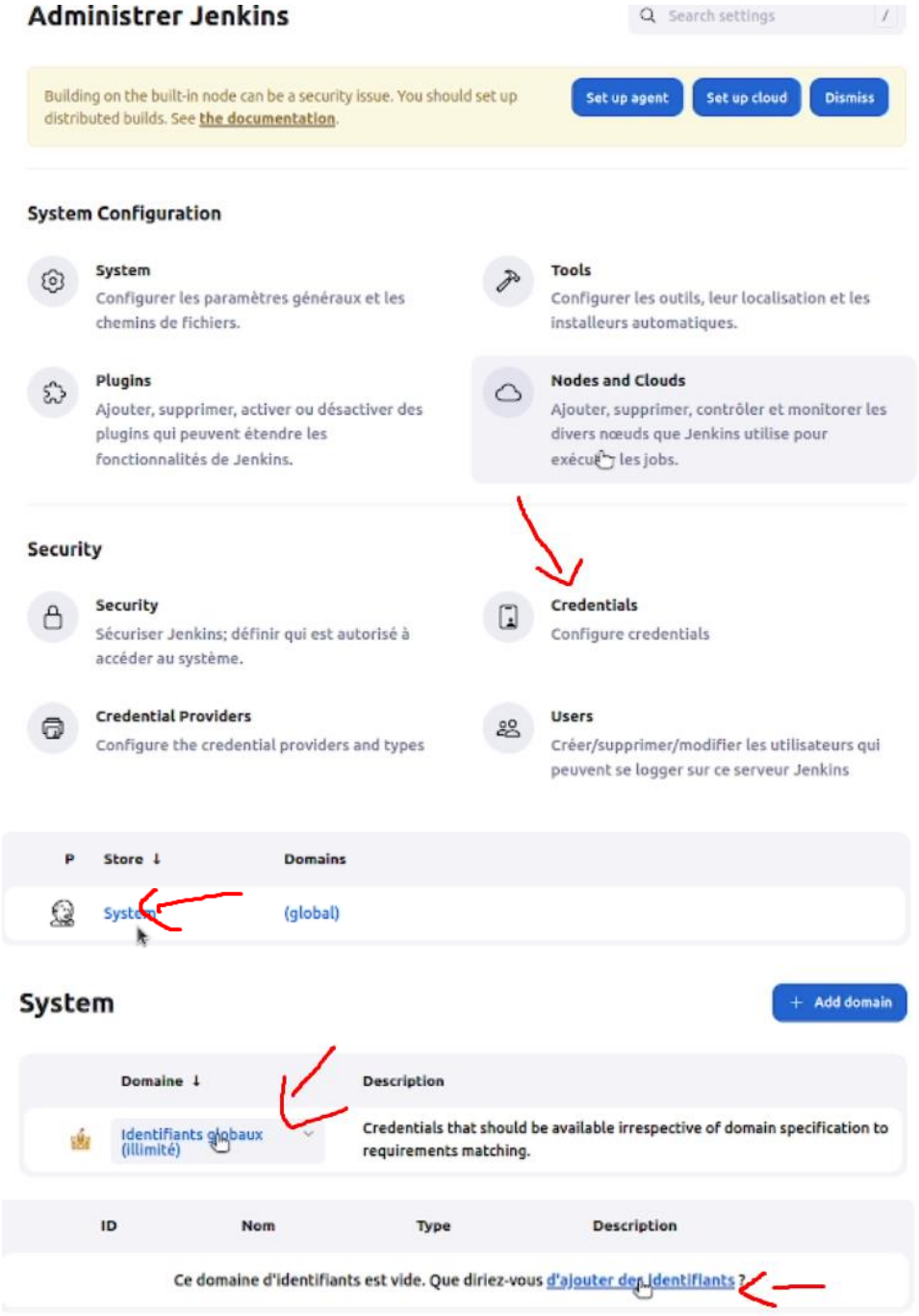
Après avoir créé le nouveau référentiel, nous devons ajouter les informations d'identification GitHub à Jenkins.

3.2 Création de Crédenciales

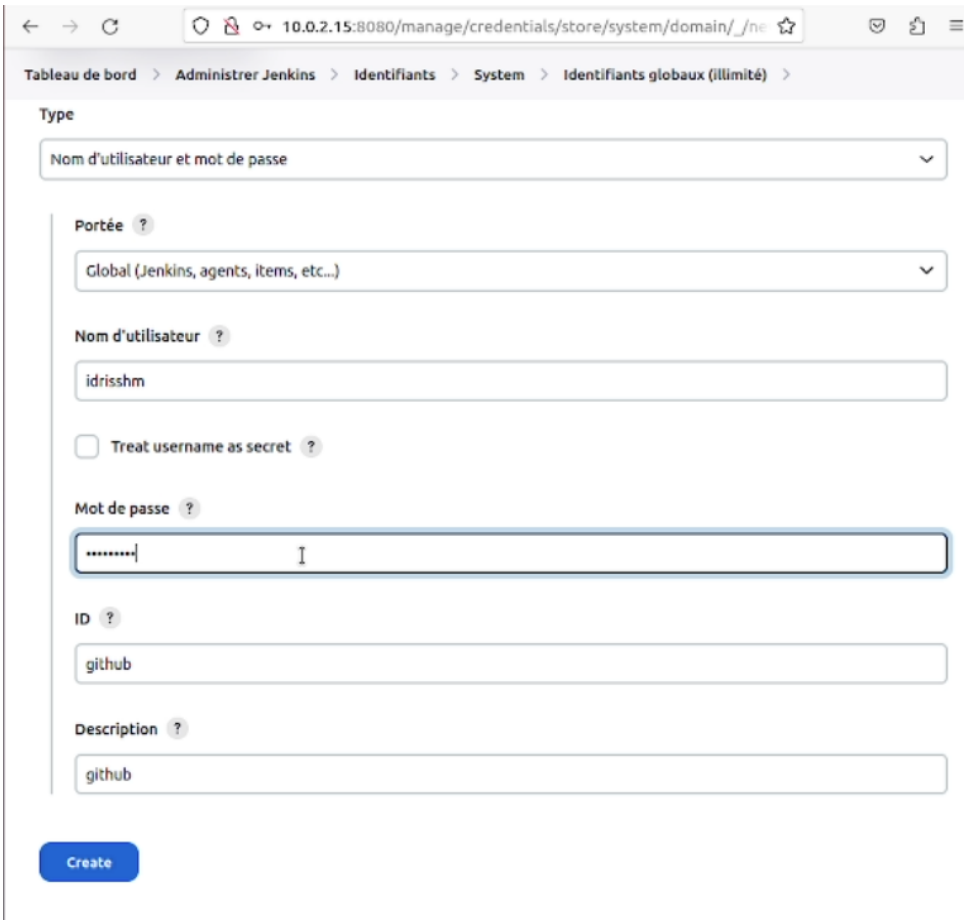
3.2.1 Crédenciales GitHub et Dockerhub

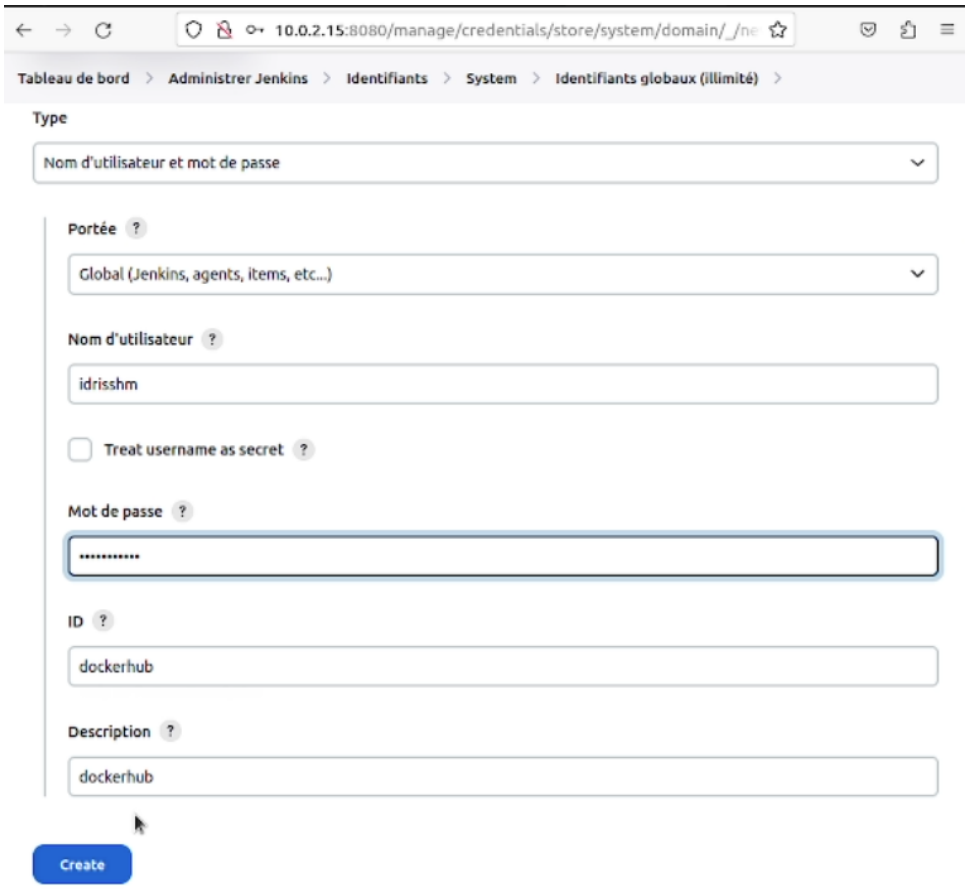
Pour créer un crédenciale vous devrez suivre ces étapes :





Créer un utilisateur avec les informations d'identification de votre compte Github et votre compte Dockerhub.



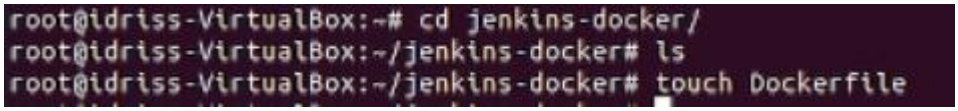


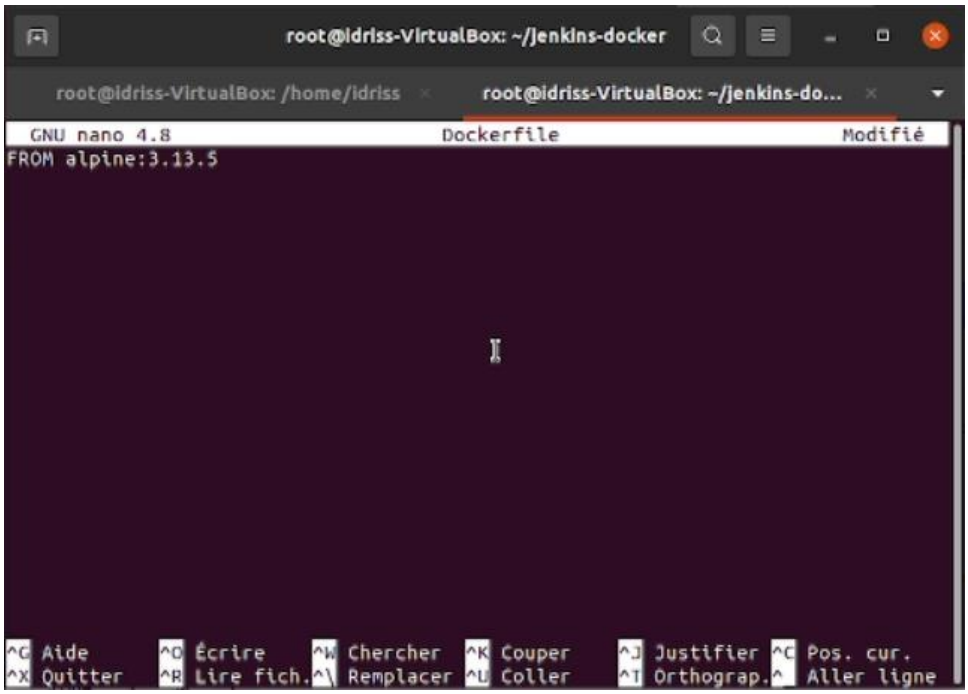
Nos deux informations d'identification sont maintenant prêtes à être utilisées. L'étape suivante consiste à créer les fichiers qui vont permettre à Jenkins de pousser l'image Docker.

4 Création de fichiers de projet

Sur votre PC, créer un nouveau dossier nommé "jenkins-docker". Dans ce dossier, créez un fichier nommé "Dockerfile" sans aucune extension et ajouter la commande Docker suivante :

FROM alpin: 3.13 . 5

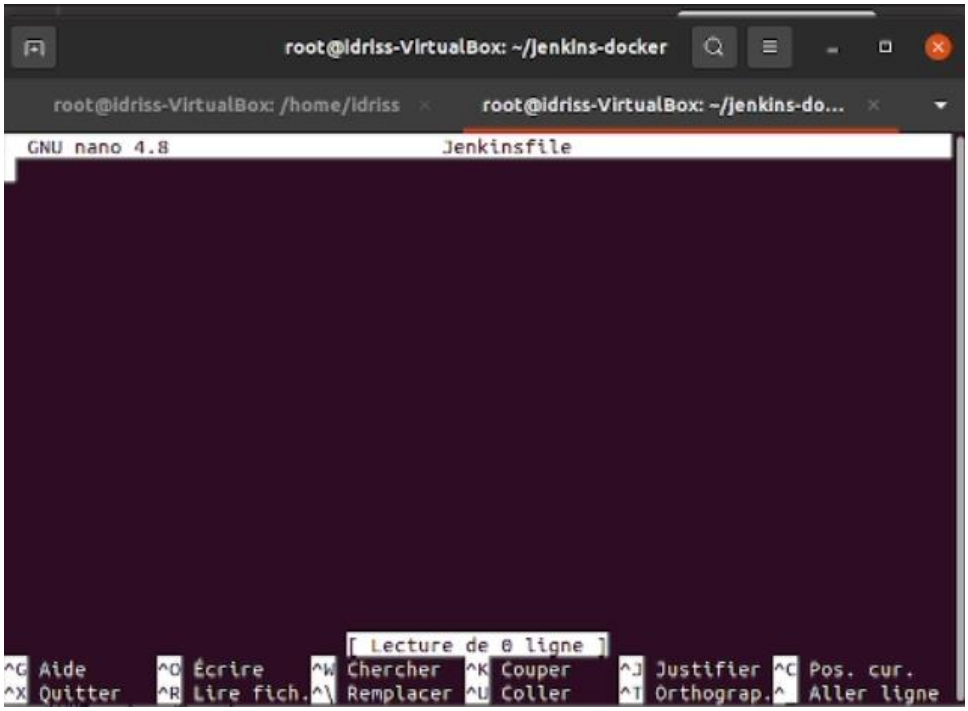




Cette commande demandera à Docker de créer une image Docker basée sur Alpine.

Ensuite, créer uichier nommé “Jenkinsfile”.

```
root@idriss-VirtualBox:~/jenkins-docker# touch Jenkinsfile
```



Ecrire le script ci-dessous :

```
pipeline {
agent any

options {
buildDiscarder(logRotator(numToKeepStr: '5'))
}

environment {
DOCKERHUB_CREDENTIALS = credentials('idrisshm')
}

stages {
stage('Build') {
steps {
sh 'docker build -t idrisshm/jenkins-docker-hub2 .'
}
}

stage('Login') {
steps {
sh 'echo $DOCKERHUB_CREDENTIALS_PSW | docker login -u
$DOCKERHUB_CREDENTIALS_USR --password-stdin'
}
}

stage('Push') {
steps {
```

```
sh 'docker push idrisshm/jenkins-docker-hub2'
}
}
}

post {
always {
sh 'docker logout'
}
}
}
```

Le fichier Jenkins créera un pipeline Jenkins en trois étapes : 'Build', 'Login' et 'Push'.

Étape de construction 'Build'

Ici, vous utiliserez le Dockerfile pour créer une image Docker basée sur Alpine. Il nommera l'image Docker "*Idrisshm/jenkins-docker-hub-2*". Remplacez ce nom par votre nom d'utilisateur Docker Hub.

Étape de connexion 'Login'

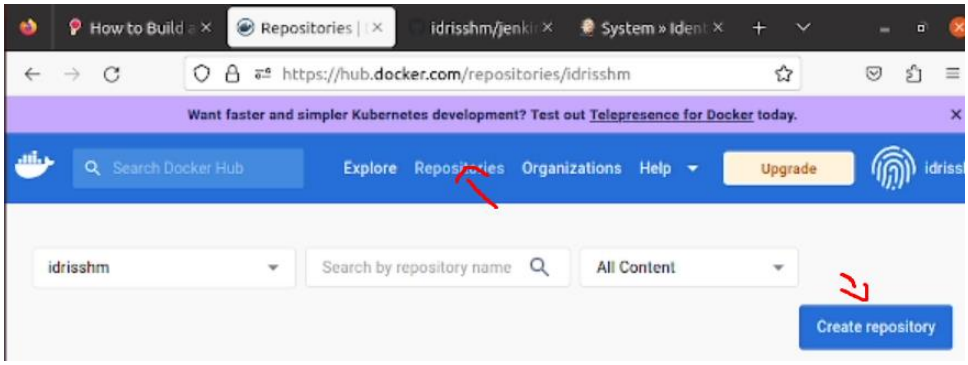
À ce stade, vous utiliserez les informations d'identification `DOCKERHUB_CREDENTIALS` pour vous connecter au compte Docker Hub.

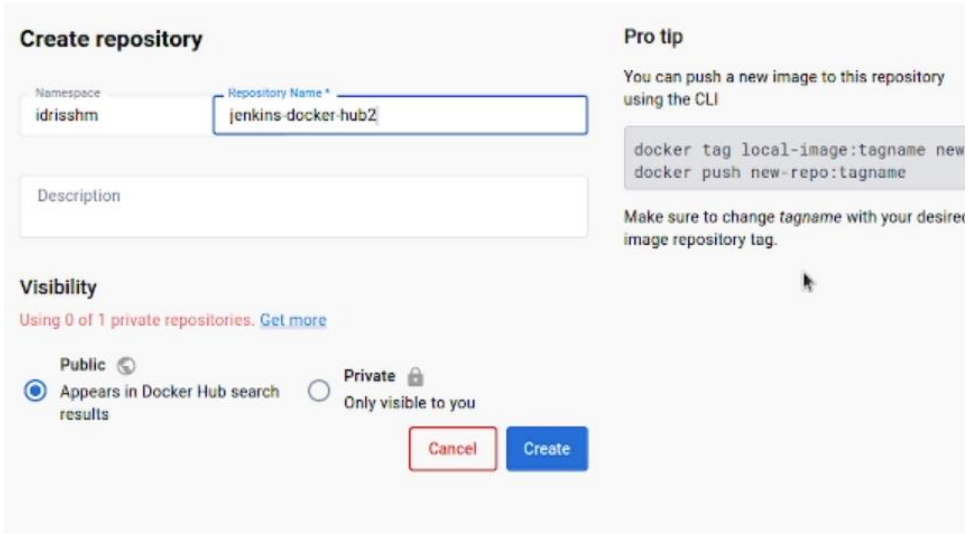
Étape de poussée 'Push'

Ensuite, vous pousserez l'image Docker "*Idrisshm/jenkins-docker-hub-2*" vers Docker Hub.

Le pipeline Jenkins a également une action post-déconnexion. Il utilise le "docker logout" pour se déconnecter de votre compte Docker Hub.

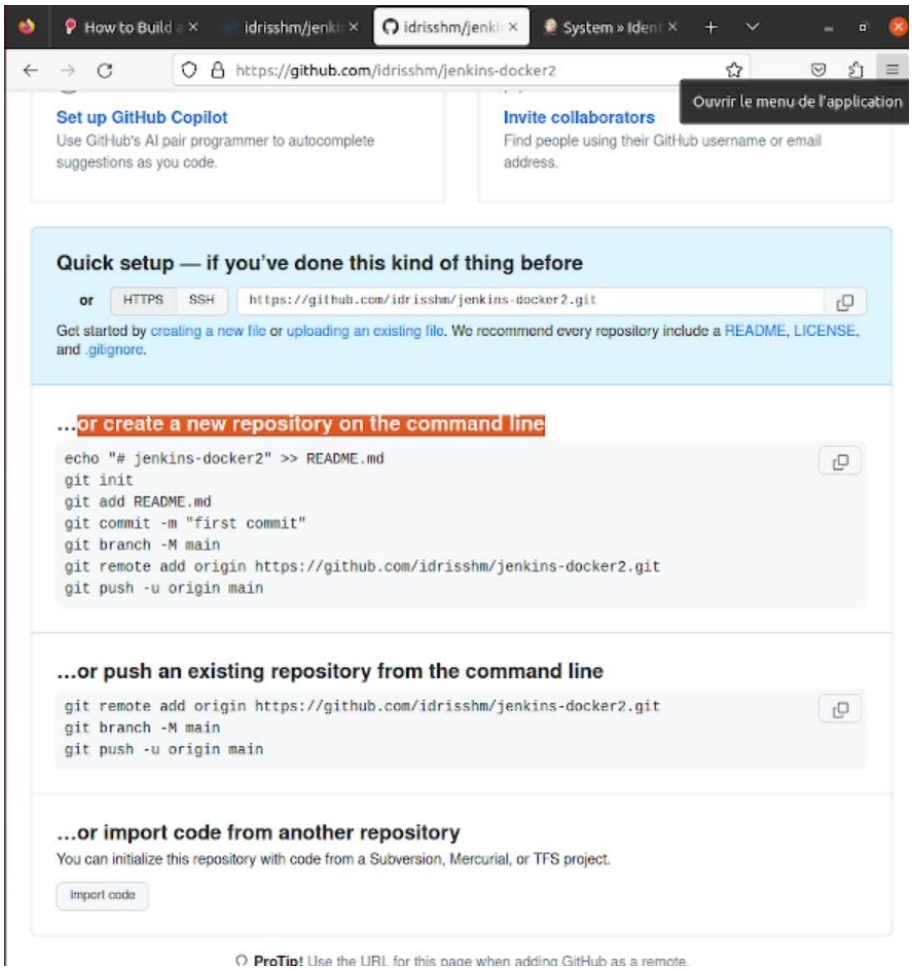
Maintenant, nous allons nous connecter à notre compte Docker Hub afin de créer un répertoire qui permettra de recevoir les fichiers poussés au bon emplacement.





5 Pousser les fichiers du projet vers GitHub

Pour transférer les fichiers du projet vers GitHub, utilisez les commandes Git répertoriées dans l'image ci-dessous :



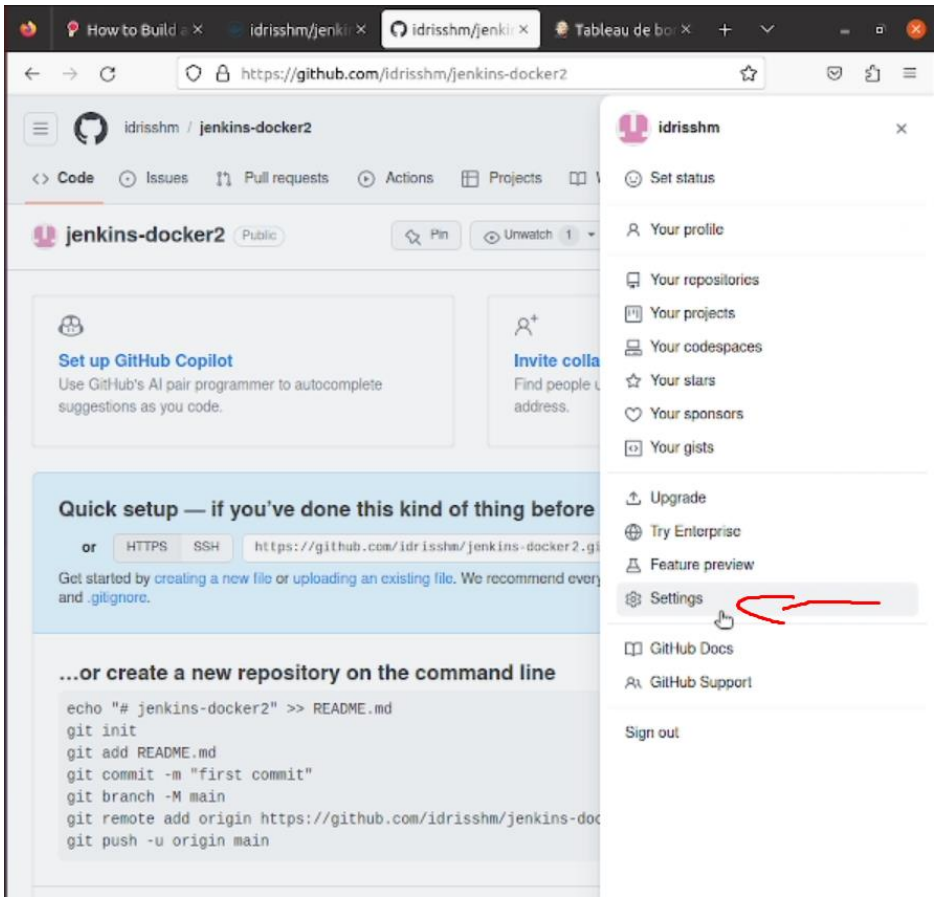

```
root@idriss-VirtualBox:~/jenkins-docker# echo "# jenkins-docker2" >> README.md
root@idriss-VirtualBox:~/jenkins-docker# git init
Dépôt Git vide initialisé dans /root/jenkins-docker/.git/
root@idriss-VirtualBox:~/jenkins-docker# git add Jenkinsfile
Dockerfile   .git/        Jenkinsfile  README.md
root@idriss-VirtualBox:~/jenkins-docker# git add Jenkinsfile Dockerfile
root@idriss-VirtualBox:~/jenkins-docker# git commit -m "first commit"
```

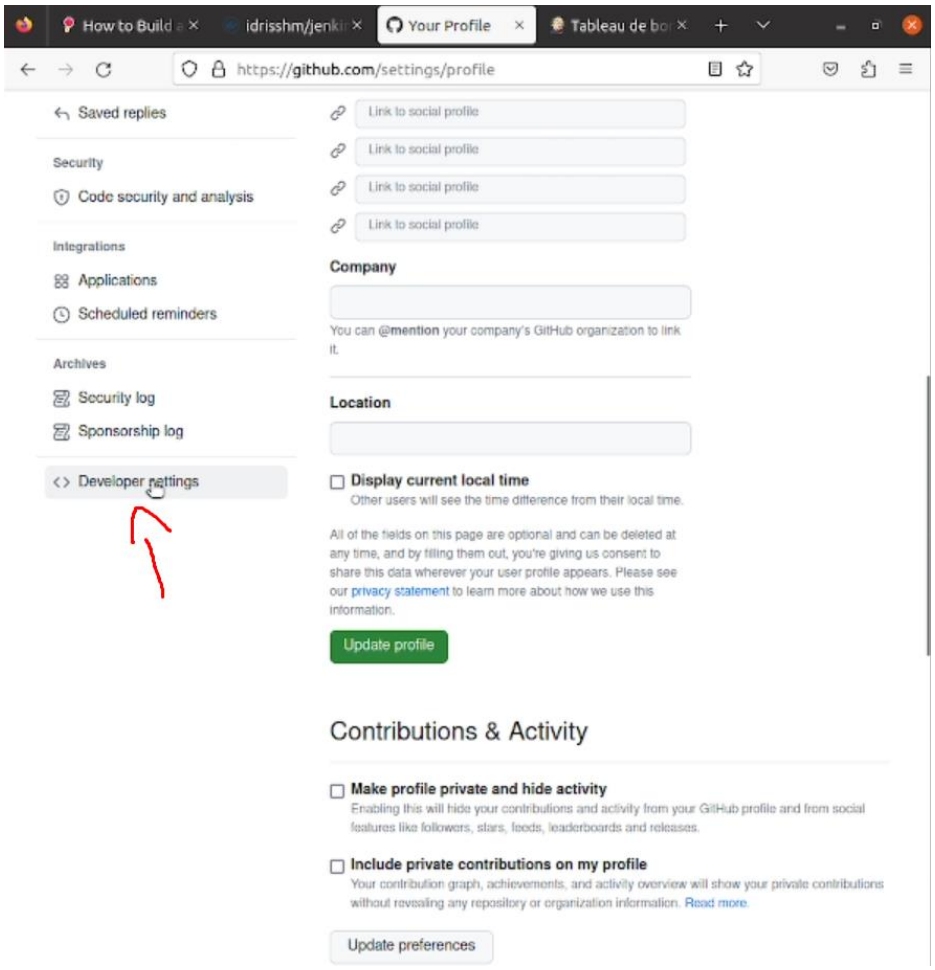
```
root@idriss-VirtualBox:~/jenkins-docker# git commit -m "first commit"

*** Veuillez me dire qui vous êtes.

Lancez

  git config --global user.email "Vous@exemple.com"
  git config --global user.name "Votre Nom"

pour régler l'identité par défaut de votre compte.
Éliminez --global pour ne faire les réglages que dans ce dépôt.

fatal: impossible de détecter automatiquement l'adresse ('root@idriss-VirtualBox
.(none)' trouvé)
root@idriss-VirtualBox:~/jenkins-docker#
```

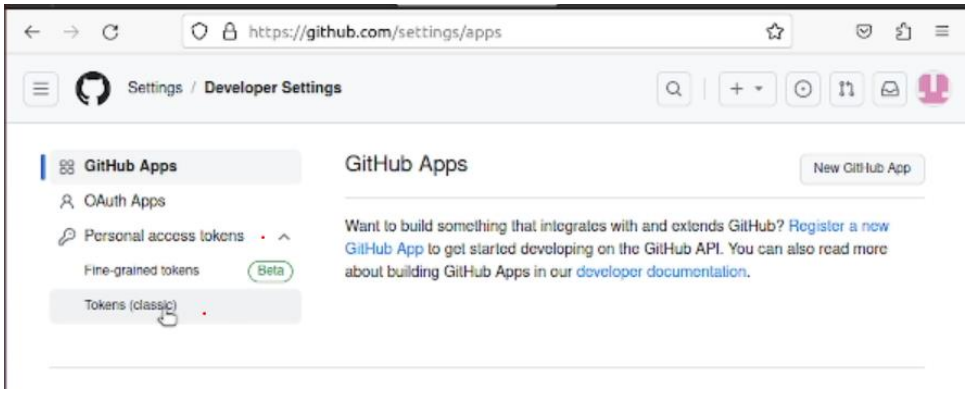
Pour corriger cette erreur il faut renseigner l'adresse email et le nom d'utilisateur avec ces commandes :

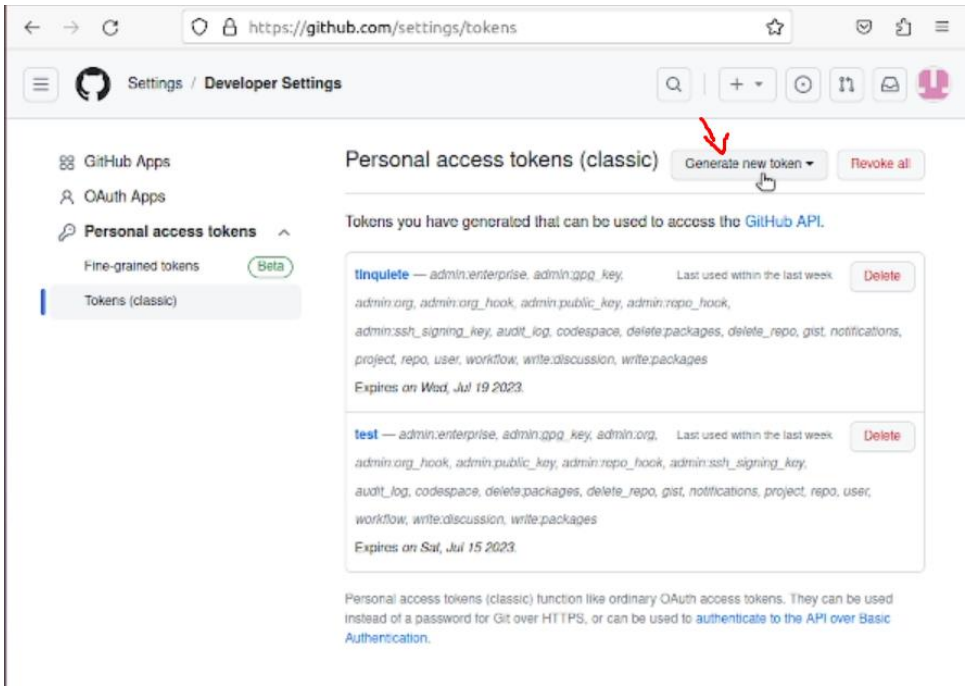
Pour le mot de passe vous devrez mettre le token que nous allons créer en suivant ces étapes :

Aller sur votre compte GitHub

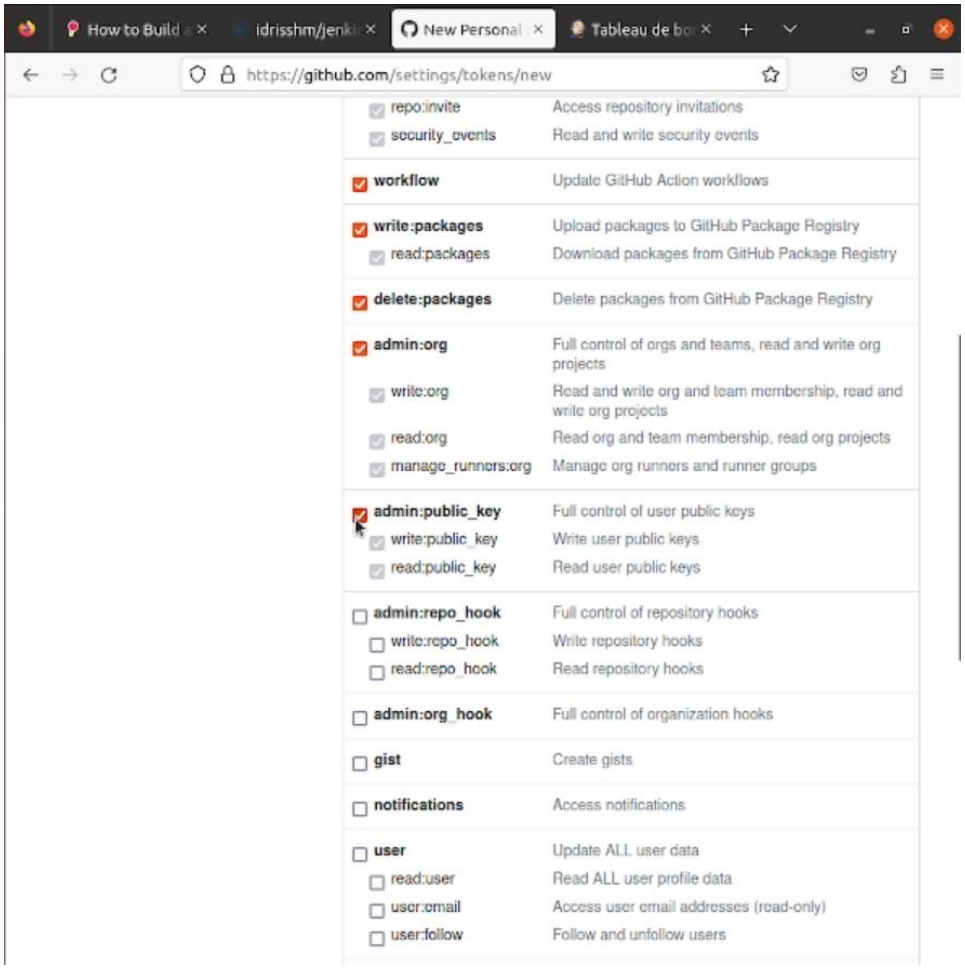








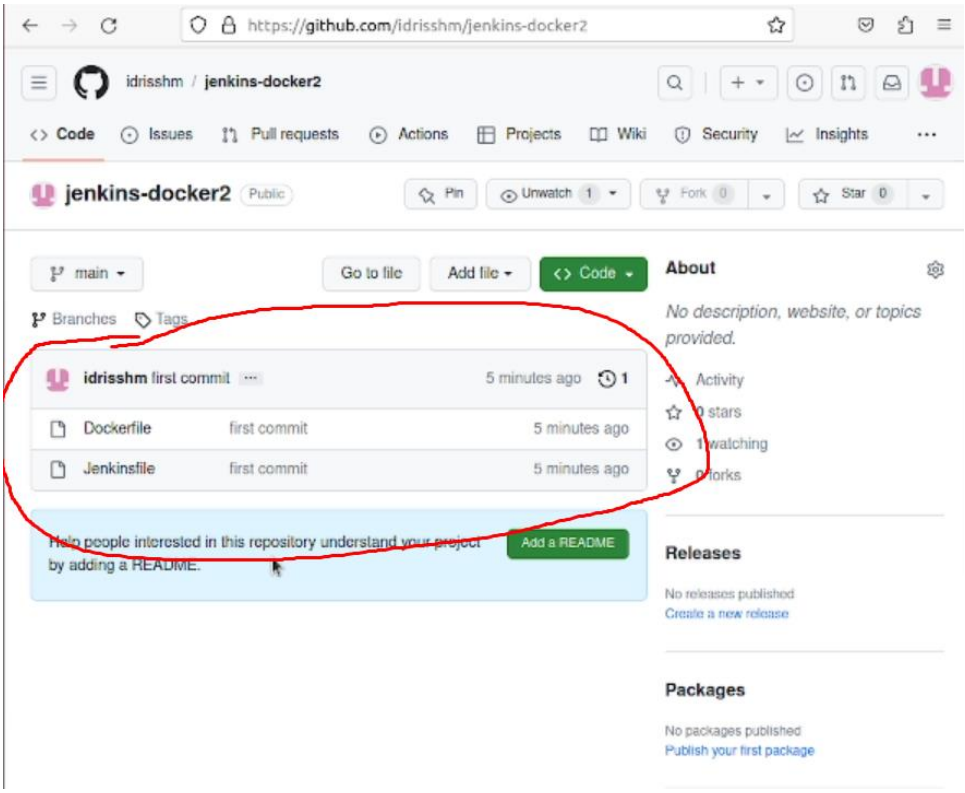
Cocher toutes les cases pour donner tous les droits



Copier la clé token

Rentrer le token lorsqu'on vous demandera le mot de passe

Après avoir appliqué les commandes Git dans votre, les fichiers du projet seront poussés vers GitHub comme indiqué ci-dessous :



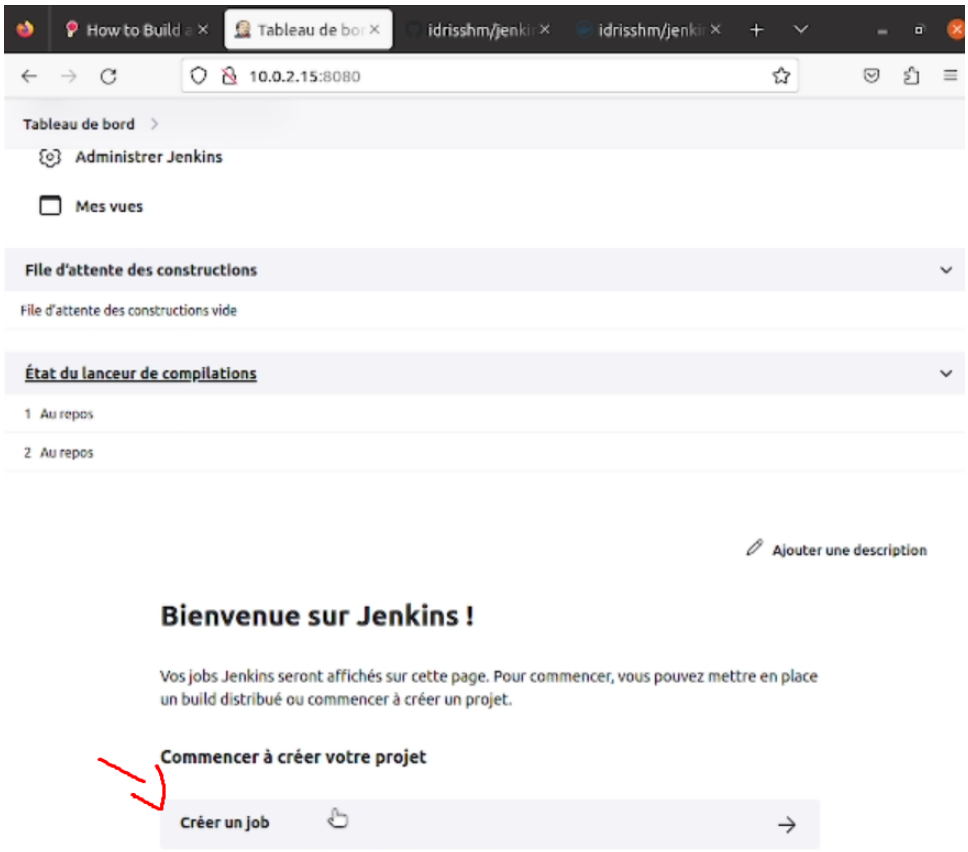
Maintenant que les fichiers sont poussés vers GitHub, on va pouvoir procéder à la dernière étape du processus.

6 Création d'un travail Jenkins

6.1 Etapes de création

Pour créer un travail Jenkins, suivez les étapes indiquées dans les images ci-dessous.

Aller dans le tableau de bord Jenkins et cliquez sur "Créer une tâche".

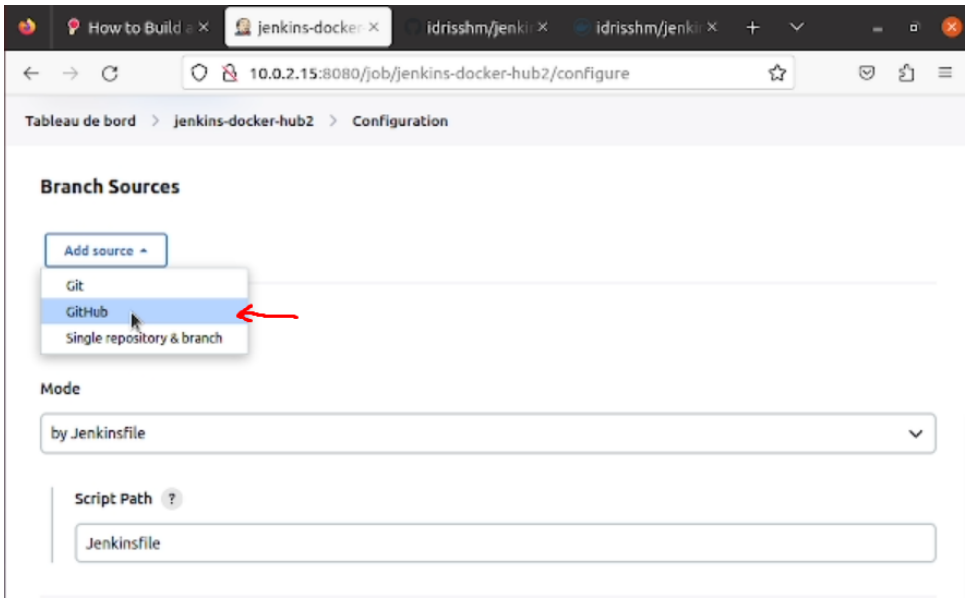


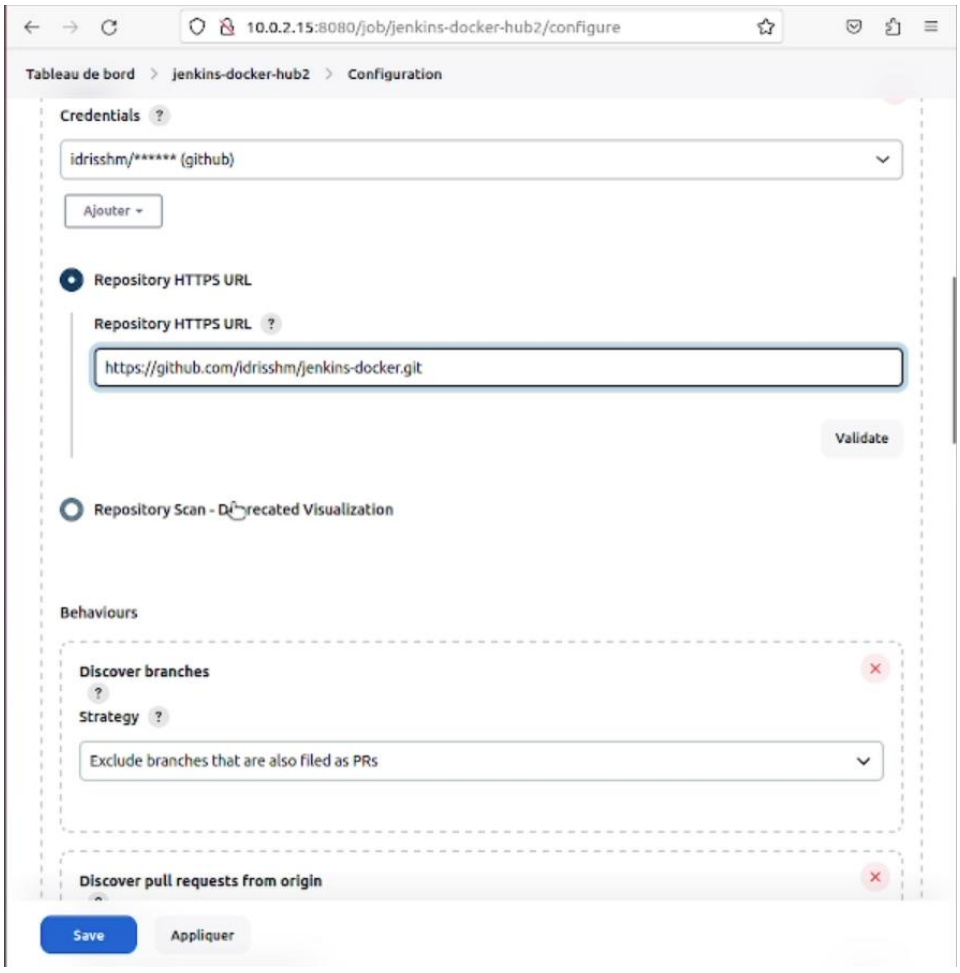
Entrez le nom du projet

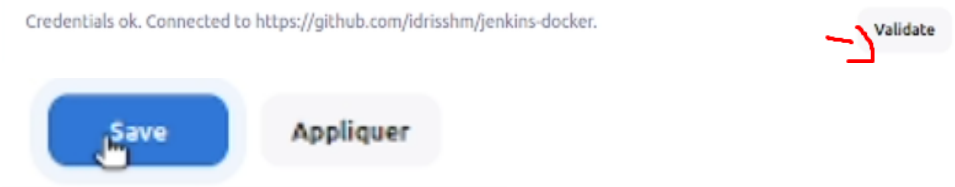
Sélectionner “Multibranch Pipeline”, puis cliquez sur `OK`

Configurer le pipeline

Vous ajoutez les informations d'identification GitHub et Repository HTTPS URL pour configurer le pipeline Jenkins :

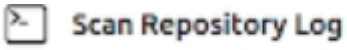


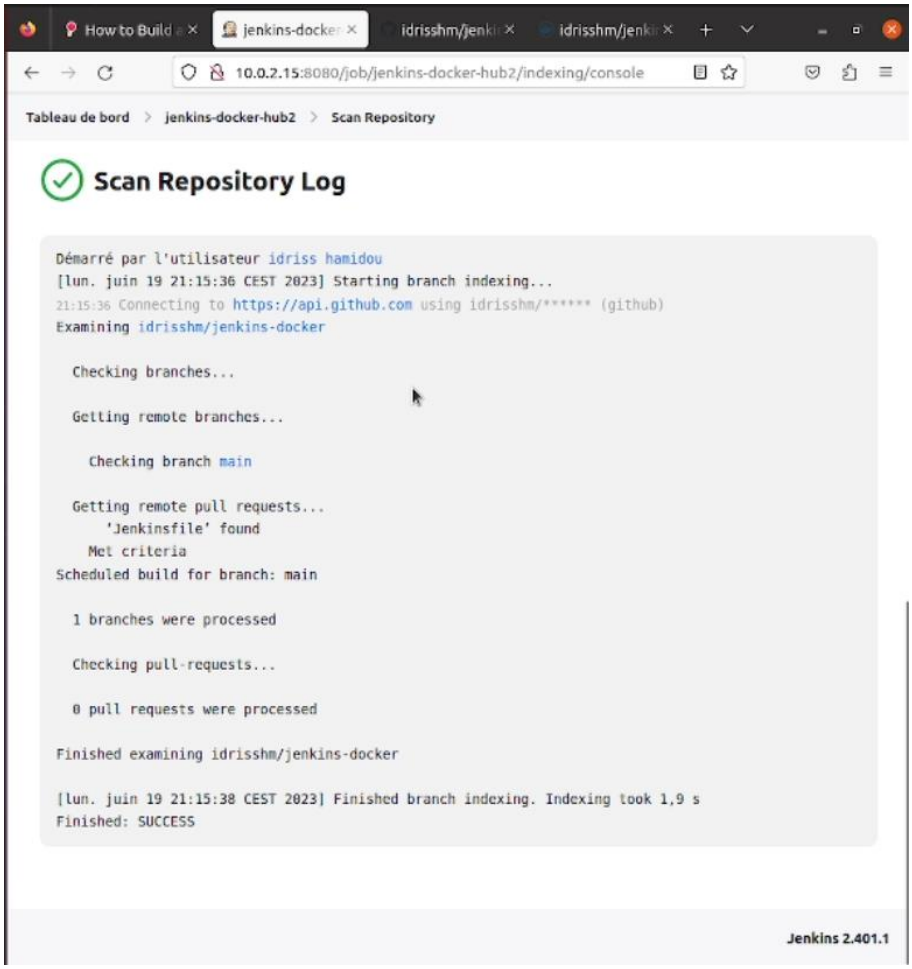




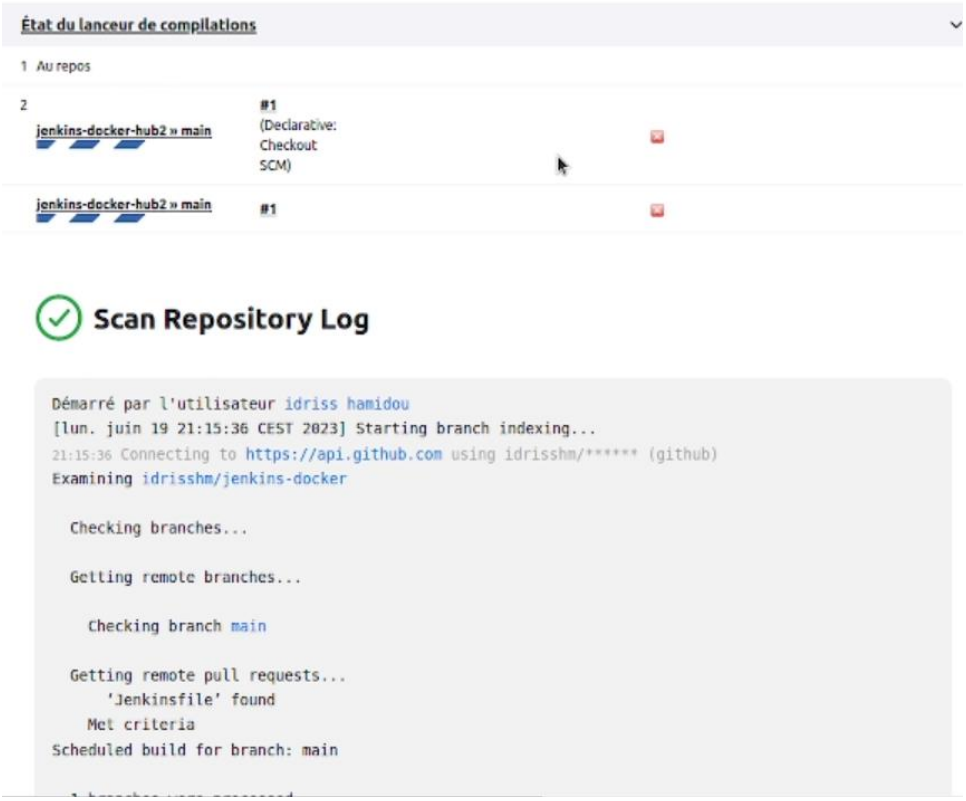
6.2 Analyser le Référentiel

Après avoir configuré le Pipeline, Jenkins commence à analyser le référentiel jusqu'à ce qu'il trouve un `Jenkinsfile` comme indiqué ci-dessous :



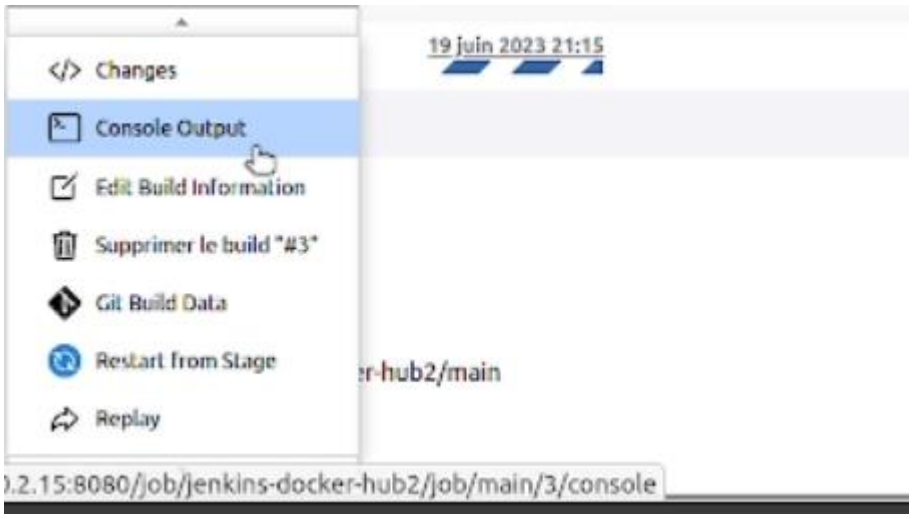


À partir de l'image ci-dessus Jenkins, Jenkins a scanné le référentiel et trouvé un “Jenkinsfile” dans la branche “main”. Jenkins utilisera “Jenkinsfile” pour exécuter le pipeline Jenkins.



6.3 Exécution du pipeline Jenkins

Après avoir scanné le référentiel, Jenkins démarrera le pipeline Jenkins en utilisant le fichier “Jenkinsfile”. Pour obtenir la sortie du Pipeline, cliquer sur le petit chiffre ci-dessous



Si vous rencontrez des erreurs lors de l'exécution du script, tapez les commandes suivantes :

```
root@idriss-VirtualBox:~/jenkins-docker# sudo usermod -aG docker idriss
```

```
root@idriss-VirtualBox:~/jenkins-docker# sudo service jenkins restart
```

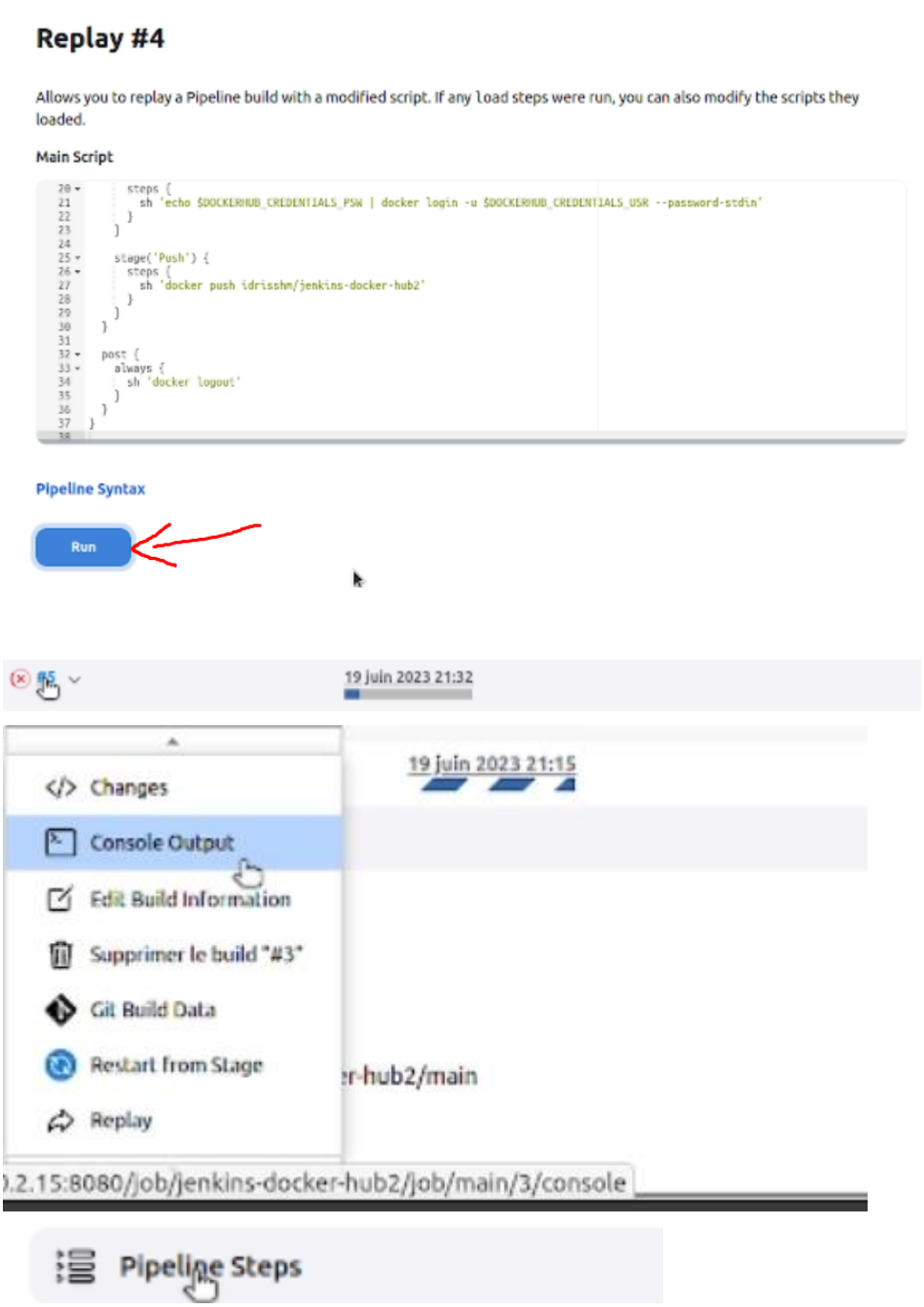
```
root@idriss-VirtualBox:~/jenkins-docker# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
root@idriss-VirtualBox:~/jenkins-docker#
```

Assurez-vous que vous n'obtenez pas l'erreur "permission denied" et que vous pouvez voir la liste des conteneurs Docker en cours d'exécution.

```
root@idriss-VirtualBox:~/jenkins-docker# sudo service jenkins restart
```

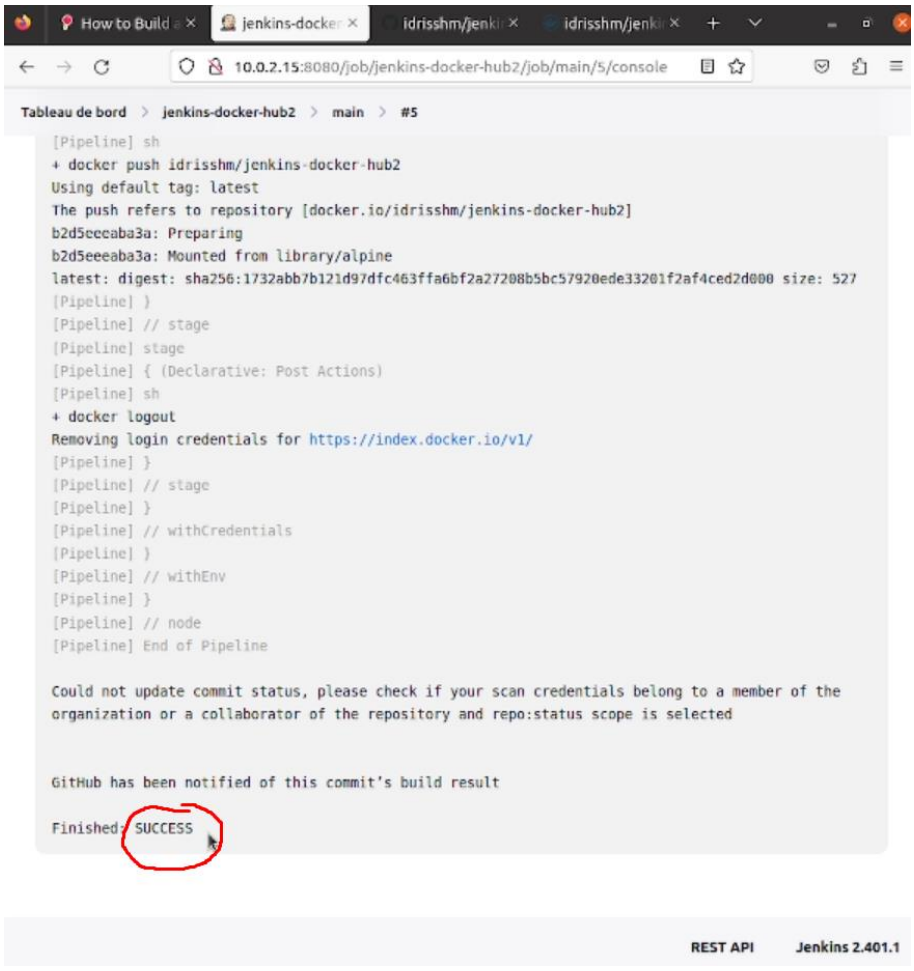
Retournez sur jenkins

Replay pour relancer le script

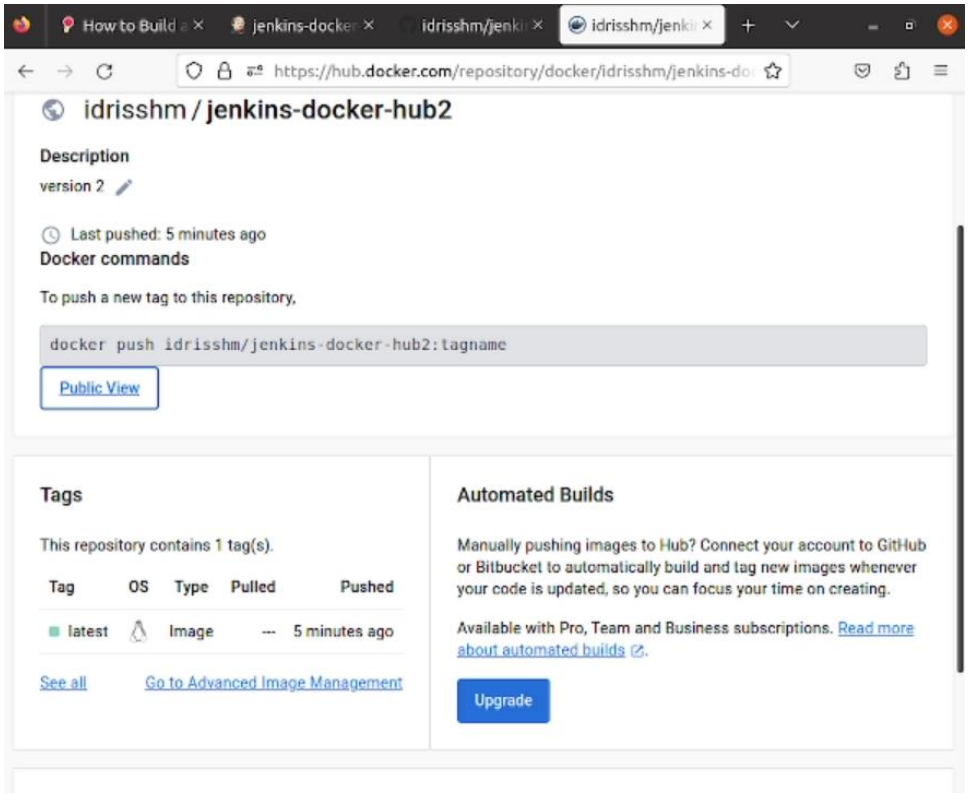


À partir de l'image ci-dessus, le pipeline Jenkins affiche un message "SUCCESS". Cela montre que le pipeline Jenkins a fini de construire et de pousser l'image Docker vers Docker Hub. Il s'est ensuite déconnecté du compte Docker Hub. Toutes les étapes du Pipeline sont présentées dans l'image ci-dessous :

stage - (3,2 s in block)	Build	
stage block (Build) - (3,1 s in block)		
sh - (2,8 s in self)	docker build -t idrisshm/jenkins-docker-hub2 .	
stage - (1,5 s in block)	Login	
stage block (Login) - (1,3 s in block)		
sh - (1,2 s in self)	echo $DOCKERHUB_CREDENTIALS_PSW | docker login -u $DOCKERHUB_CREDENTIALS_USR --password-stdin	
stage - (5,4 s in block)	Push	
stage block (Push) - (5 s in block)		
sh - (4,9 s in self)	docker push idrisshm/jenkins-docker-hub2	
stage - (0,52 s in block)	Declarative: Post Actions	
stage block (Declarative: Post Actions) - (0,45 s in block)		
sh - (0,35 s in self)	docker logout	



Connectez-vous à votre compte Docker Hub pour voir l'image Docker poussée :



Le pipeline Jenkins a poussé l'image Docker vers Docker Hub. Nous avons utilisé avec succès le pipeline Jenkins pour créer et pousser une image Docker vers Docker Hub.