

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	3
1.1	DOCKER	3
2	INSTALLATION DE JENKINS SUR UBUNTU 20.04	5
2.1	JENKINS	5
3	CREATION D'UN NOUVEAU REFERENTIEL GITHUB	10
3.1	NOUVEAU REFERENTIEL	10
3.2	CREATION DE CREDENCIALES	11
4	CREATION DE FICHIERS DE PROJET	14
5	POUSSER LES FICHIERS DU PROJET VERS GITHUB	18
6	CREATION D'UN TRAVAIL JENKINS	25
6.1	ETAPES DE CREATION	25
6.2	ANALYSER LE REFERENTIEL	27
6.3	EXECUTION DU PIPELINE JENKINS	28

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 :

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

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

```
ldriss@ldriss-VirtualBox:~$ sudo apt install apt-transport-https ca-certificates curl software-properties-common
```

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

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

Ajouter le référentiel Docker aux sources APT :

```
ldriss@ldriss-VirtualBox:~$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/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 :

```
ldriss@ldriss-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 :

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

1.1.2 Installation de Docker



Pour installer Docker, tapez la commande suivante :

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

Après son installation, vérifier qu'il tourne :

```
ldriss@ldriss-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

jun 19 19:04:01 ldriss-VirtualBox systemd[1]: Starting Docker Application Container Engine...
jun 19 19:04:01 ldriss-VirtualBox dockerd[8592]: time="2023-06-19T19:04:01.876424937+02:00" le
jun 19 19:04:01 ldriss-VirtualBox dockerd[8592]: time="2023-06-19T19:04:01.877810298+02:00" le
jun 19 19:04:01 ldriss-VirtualBox dockerd[8592]: time="2023-06-19T19:04:01.991822164+02:00" le
jun 19 19:04:02 ldriss-VirtualBox dockerd[8592]: time="2023-06-19T19:04:02.202143230+02:00" le
jun 19 19:04:02 ldriss-VirtualBox dockerd[8592]: time="2023-06-19T19:04:02.257439122+02:00" le
jun 19 19:04:02 ldriss-VirtualBox dockerd[8592]: time="2023-06-19T19:04:02.257513382+02:00" le
jun 19 19:04:02 ldriss-VirtualBox dockerd[8592]: time="2023-06-19T19:04:02.291385112+02:00" le
jun 19 19:04:02 ldriss-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 :

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

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

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

Vous devriez voir apparaître hello-world

```
root@ldriss-VirtualBox:/home/ldriss# docker images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
ubuntu              latest         79284ca6cea0   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 :

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

Taper la commande ci-dessous pour installer OpenJDK 11 :

```
ldriss@ldriss-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 :

```
ldriss@ldriss-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 :

```
ldriss@ldriss-VirtualBox:~$ wget -p -O - https://pkg.jenkins.io/debian/jenkins.l  
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 :



```
ldriss@ldriss-VirtualBox:~$ sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable 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.

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

Taper la commande ci-dessous pour installer Jenkins :

```
ldriss@ldriss-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 :

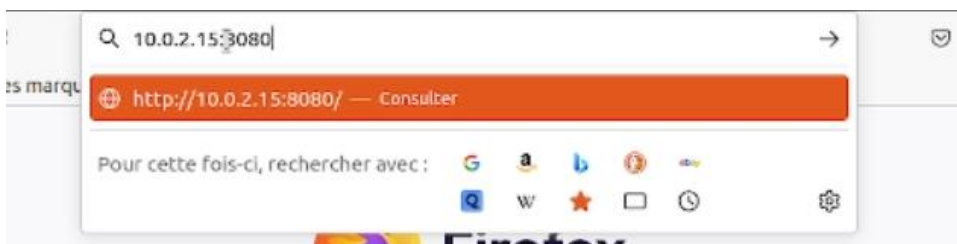
```
ldriss@ldriss-VirtualBox:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor prese
   Active: active (running) since Mon 2023-06-19 20:12:32 CEST; 23s ago
     Main PID: 7924 (java)
       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 :

```
ldriss@ldriss-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.



Démarrage

Débloquer Jenkins

Pour être sûr que Jenkins soit configuré de façon sécurisée par un administrateur, un mot de passe a été généré dans le fichier de logs (où le trouver) ainsi que dans ce fichier sur le serveur :

```
/var/lib/jenkins/secrets/initialAdminPassword
```

Veuillez copier le mot de passe depuis un des 2 endroits et le coller ci-dessous.

Mot de passe administrateur

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

```
idriiss@idriiss-VirtualBox:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
[sudo] Mot de passe de idriiss :
c8b711ad3acfb43e99afbcca4ca54e379d
idriiss@idriiss-VirtualBox:~$
```

Sélectionner "Installer les plugins suggérés" :



Personnaliser Jenkins

Les plugins étendent Jenkins avec des fonctionnalités additionnelles pour satisfaire différents besoins.

Installer les plugins suggérés
Installer les plugins que la communauté Jenkins trouve les plus utiles.

Sélectionner les plugins à installer
Sélectionner et installer les plugins les plus utiles à vos besoins.

Installation en cours...

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



Créer le 1er utilisateur Administrateur

Nom d'utilisateur

Mot de passe

Confirmation du mot de passe

Nom complet

Adresse courriel

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

Configuration de l'instance

URL de Jenkins :

L'URL de Jenkins est utilisée pour fournir l'URL de base pour les liens absolus vers les diverses ressources Jenkins. Cela signifie que cette valeur est nécessaire pour le bon fonctionnement de nombreuses fonctionnalités de Jenkins, notamment les notifications par mail, les mises à jour des statuts des pull requests, et la variable d'environnement BUILD_URL fournie pour les étapes de build. La valeur par défaut affichée n'est pas encore sauvegardée et est générée à partir de la requête actuelle, lorsque c'est possible. Il est fortement recommandé d'utiliser comme valeur l'URL qui est censée être utilisée par les utilisateurs. Cela évitera des confusions lors du partage ou de la visualisation de liens.

.401.1

[Passer cette étape et terminer](#)

[Sauvegarder et terminer](#)

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

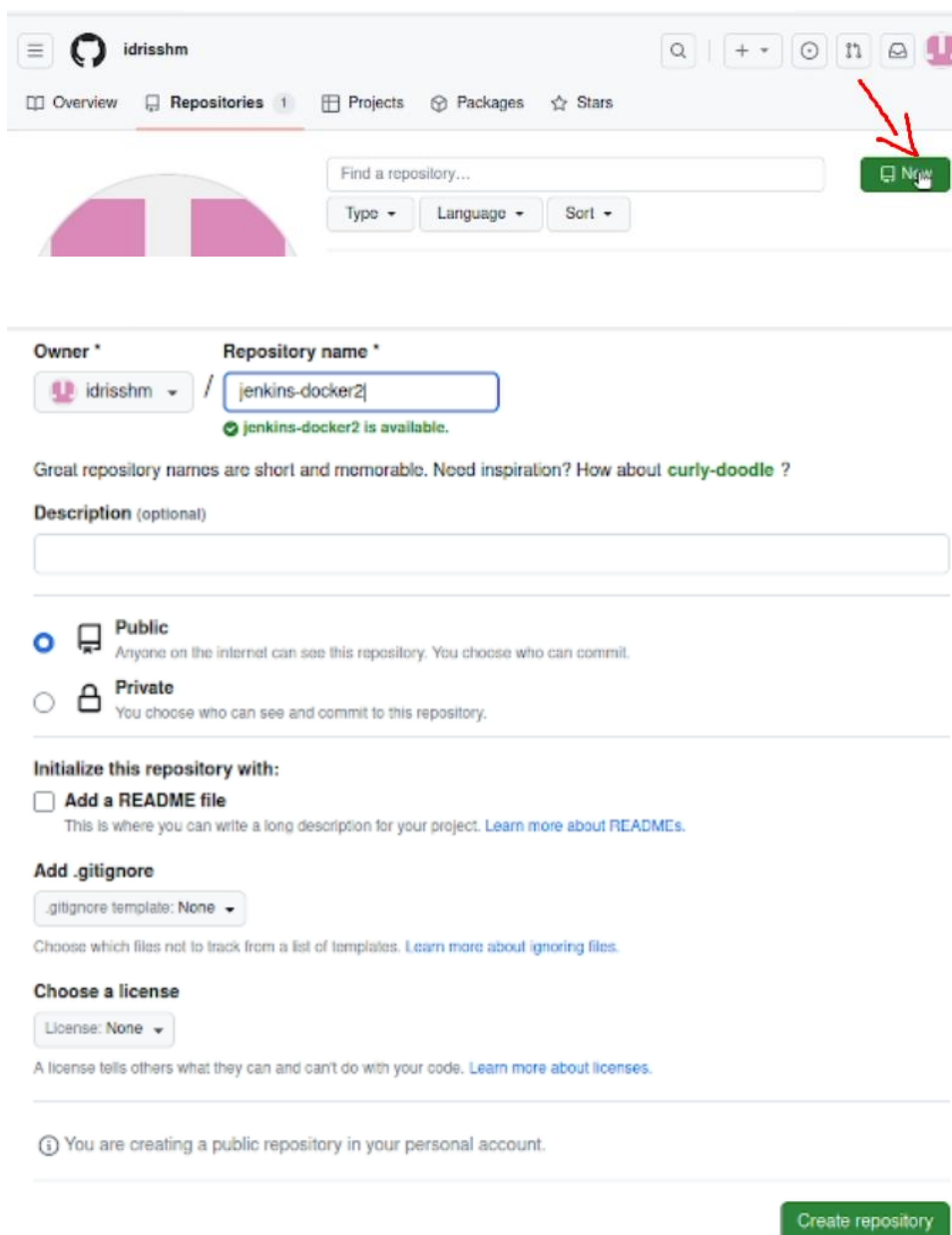


Maintenant que Jenkins est prêt à être utilisé, nous allons créer les fichiers qui nous permettront de pousser l'image Docker à l'aide de GitHub. Il faut donc créer un compte GitHub si vous n'en avez pas.

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 :



The screenshot shows the GitHub 'New repository' page for user 'idrisshh'. The repository name is 'jenkins-docker2', which is marked as available. The page includes a search bar, a 'New' button (highlighted with a red arrow), and various options for repository creation such as 'Add a README file', 'Add .gitignore', and 'Choose a license'. The 'Public' visibility option is selected.



Copier ce lien car on en aura besoin plus tard

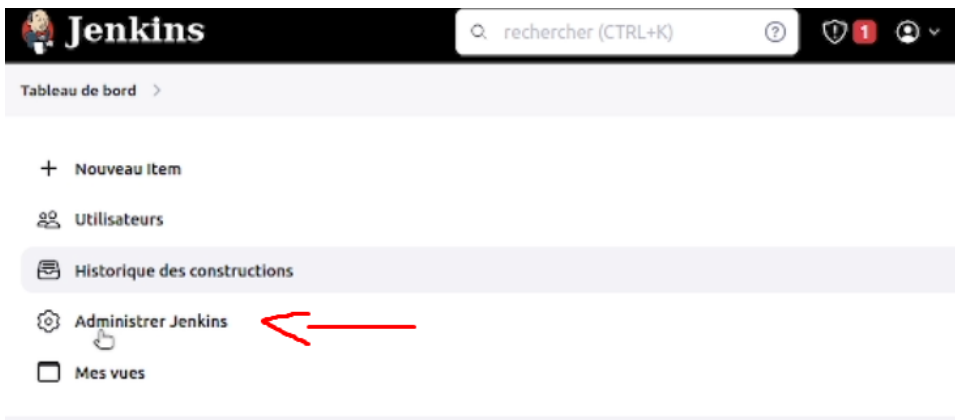


Après avoir créé le nouveau référentiel, nous devons ajouter les informations d'identification GitHub à Jenkins.

3.2 Création de Crédenciales

3.2.1 Crédenciales GitHub et Dockerhub

Pour créer un crédencial vous devrez suivre ces étapes :



Administrer Jenkins

Search settings

Building on the built-in node can be a security issue. You should set up distributed builds. See [the documentation](#).

Set up agent

Set up cloud

Dismiss

System Configuration



System

Configurer les paramètres généraux et les chemins de fichiers.



Plugins

Ajouter, supprimer, activer ou désactiver des plugins qui peuvent étendre les fonctionnalités de Jenkins.



Tools

Configurer les outils, leur localisation et les installeurs automatiques.



Nodes and Clouds

Ajouter, supprimer, contrôler et monitorer les divers nœuds que Jenkins utilise pour exécuter les jobs.

Security



Security

Sécuriser Jenkins; définir qui est autorisé à accéder au système.



Credential Providers

Configure the credential providers and types



Credentials

Configurer credentials



Users

Créer/supprimer/modifier les utilisateurs qui peuvent se logger sur ce serveur Jenkins

P

Store ↓

Domains



System

(global)

System

+ Add domain

Domaine ↓

Description



Identifiants globaux (illimité)

Credentials that should be available irrespective of domain specification to requirements matching.

ID

Nom

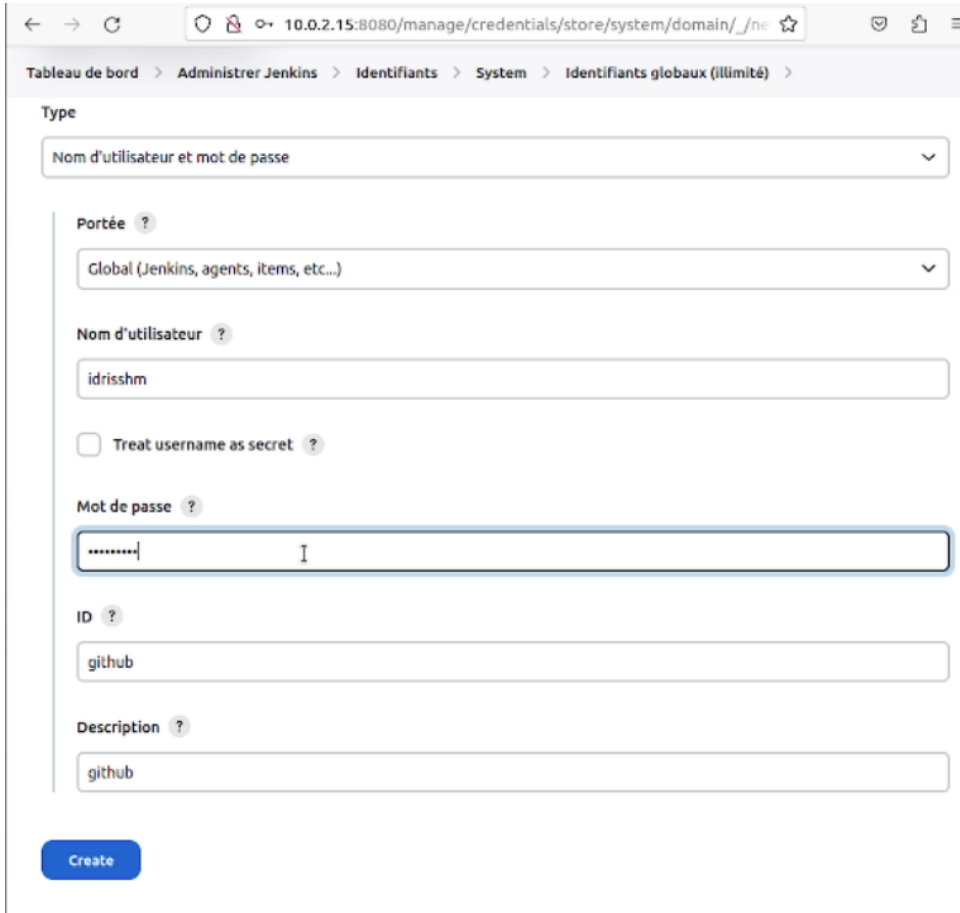
Type

Description

Ce domaine d'identifiants est vide. Que diriez-vous d'[ajouter des identifiants](#)?



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

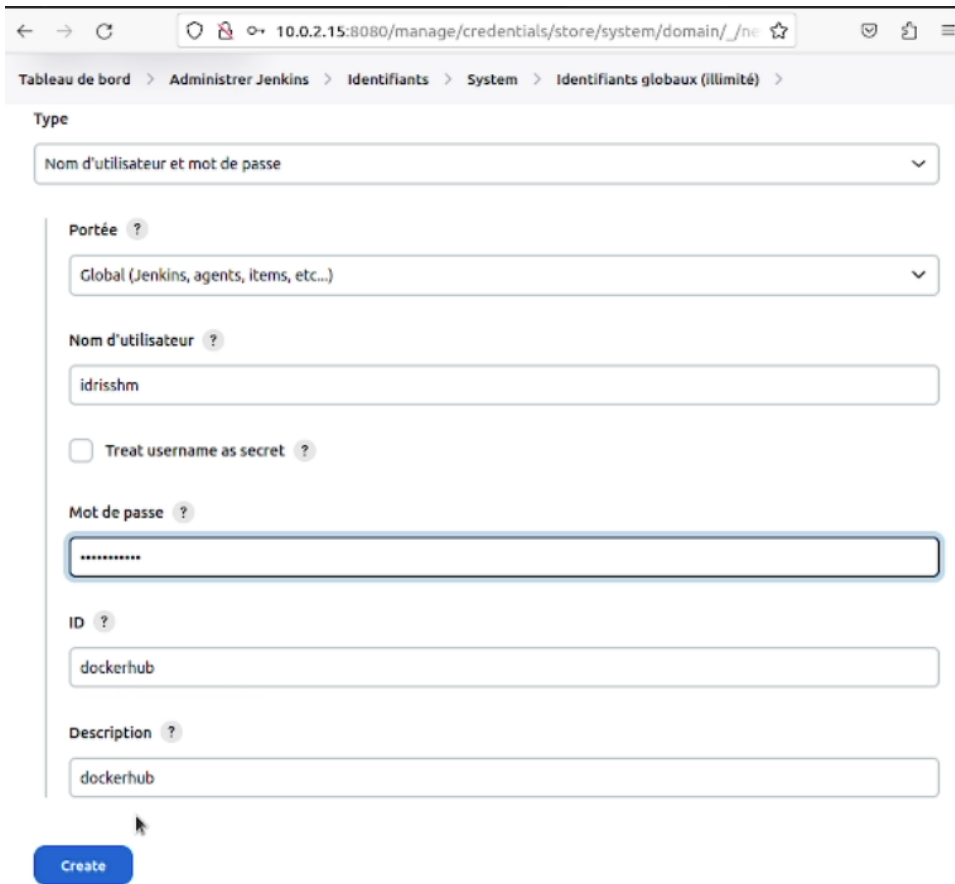


The screenshot shows the Jenkins web interface for creating a new credential. The browser address bar shows the URL: 10.0.2.15:8080/manage/credentials/store/system/domain/_ne. The breadcrumb navigation is: Tableau de bord > Administrer Jenkins > Identifiants > System > Identifiants globaux (illimité). The form fields are as follows:

- Type:** A dropdown menu with the selected option "Nom d'utilisateur et mot de passe".
- Portée ?:** A dropdown menu with the selected option "Global (Jenkins, agents, items, etc...)".
- Nom d'utilisateur ?:** A text input field containing "idrissh".
- Treat username as secret ?:** An unchecked checkbox.
- Mot de passe ?:** A password input field with masked characters "....." and a cursor.
- ID ?:** A text input field containing "github".
- Description ?:** A text input field containing "github".

At the bottom left of the form area, there is a blue "Create" button.





The screenshot shows the Jenkins web interface for configuring a global credential. The breadcrumb trail is: Tableau de bord > Administrer Jenkins > Identifiants > System > Identifiants globaux (illimité). The form is titled "Type" and has a dropdown menu set to "Nom d'utilisateur et mot de passe". Below this, the "Portée" (Scope) is set to "Global (Jenkins, agents, items, etc...)". The "Nom d'utilisateur" (Username) field contains "idrissh". There is an unchecked checkbox for "Treat username as secret". The "Mot de passe" (Password) field is masked with dots. The "ID" field contains "dockerhub" and the "Description" field also contains "dockerhub". A blue "Create" button is located at the bottom left of the form.

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
```

```
idris@idris-VirtualBox:~$ sudo su
[sudo] Mot de passe de idris :
root@idris-VirtualBox:/home/idris# cd
root@idris-VirtualBox:~# mkdir jenkins-docker
root@idris-VirtualBox:~# ls
jenkins-docker  snap
root@idris-VirtualBox:~#
```

```
root@idris-VirtualBox:~# cd jenkins-docker/
root@idris-VirtualBox:~/jenkins-docker# ls
root@idris-VirtualBox:~/jenkins-docker# touch Dockerfile
```



```
root@idriss-VirtualBox: ~/jenkins-docker
root@idriss-VirtualBox: /home/idriss x root@idriss-VirtualBox: ~/jenkins-do... x
GNU nano 4.8 Dockerfile Modifié
FROM alpine:3.13.5

^G Aide ^O Écrire ^W Chercher ^K Couper ^J Justifier ^C Pos. cur.
^X Quitter ^R Lire fich. ^\ Remplacer ^L Coller ^T Orthograp. ^_ Aller ligne
```

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
root@idriss-VirtualBox: ~/jenkins-docker
root@idriss-VirtualBox: /home/idriss x root@idriss-VirtualBox: ~/jenkins-do... x
GNU nano 4.8 Jenkinsfile

[ Lecture de 0 ligne ]
^G Aide ^O écrire ^W Chercher ^K Couper ^J Justifier ^C Pos. cur.
^X Quitter ^R Lire fich. ^\ Remplacer ^L Coller ^T Orthograp. ^_ Aller ligne
```



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.

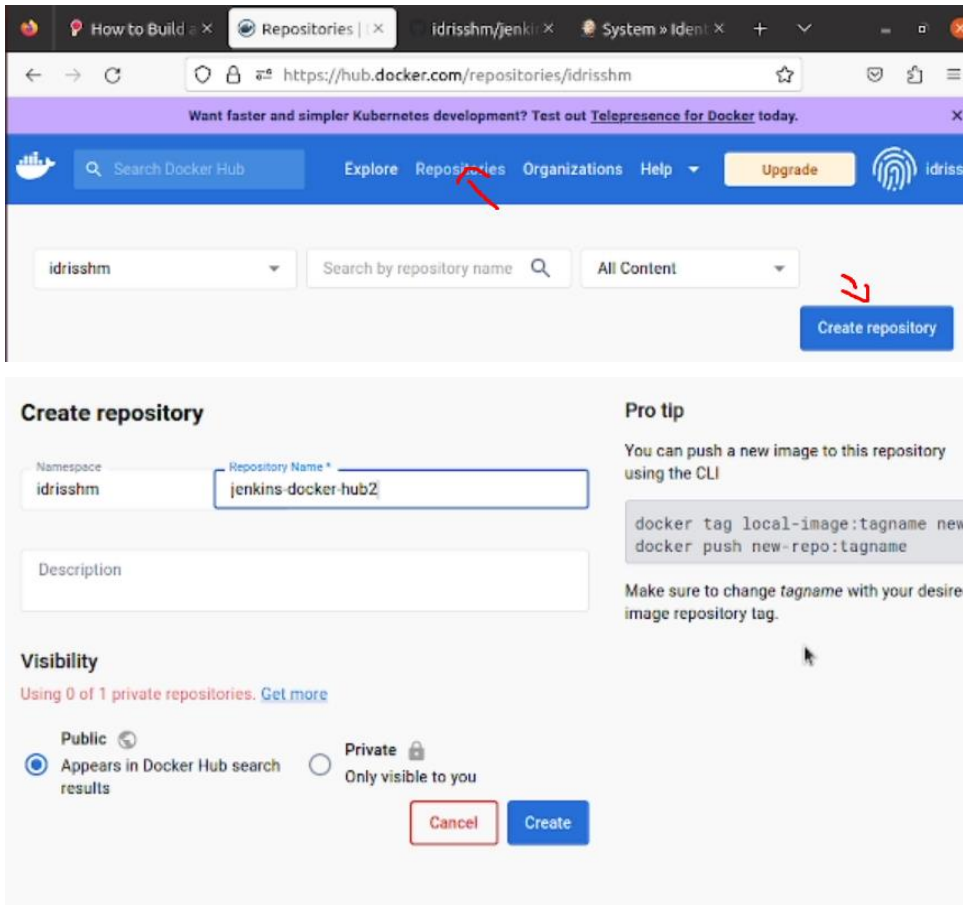
Étape de poussée 'Push'

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.

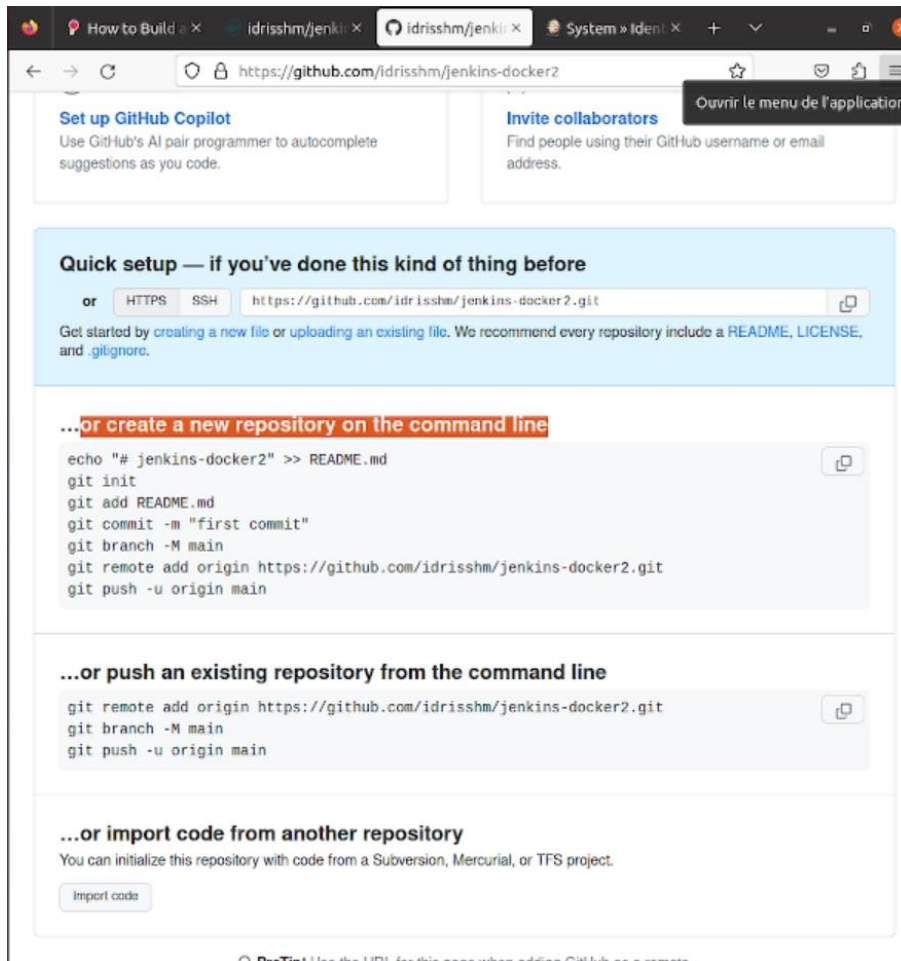




5 Pousser les fichiers du projet vers GitHub



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



The screenshot shows the GitHub repository setup page for 'idrisshm/jenkins-docker2'. It includes sections for 'Quick setup' with a URL input field, and three sections for command-line instructions: '...or create a new repository on the command line', '...or push an existing repository from the command line', and '...or import code from another repository'. A 'ProTip!' at the bottom suggests using the URL for adding GitHub as a remote.

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

```
root@idrissh-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@idrissh-VirtualBox:~/jenkins-docker#
```

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

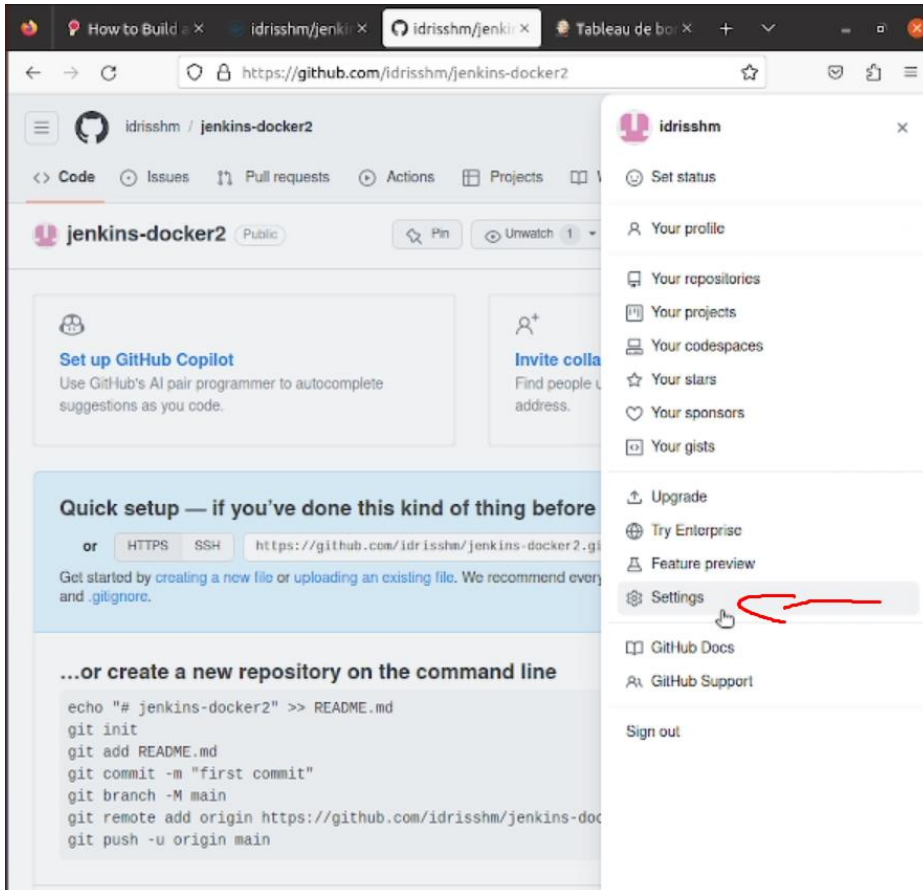


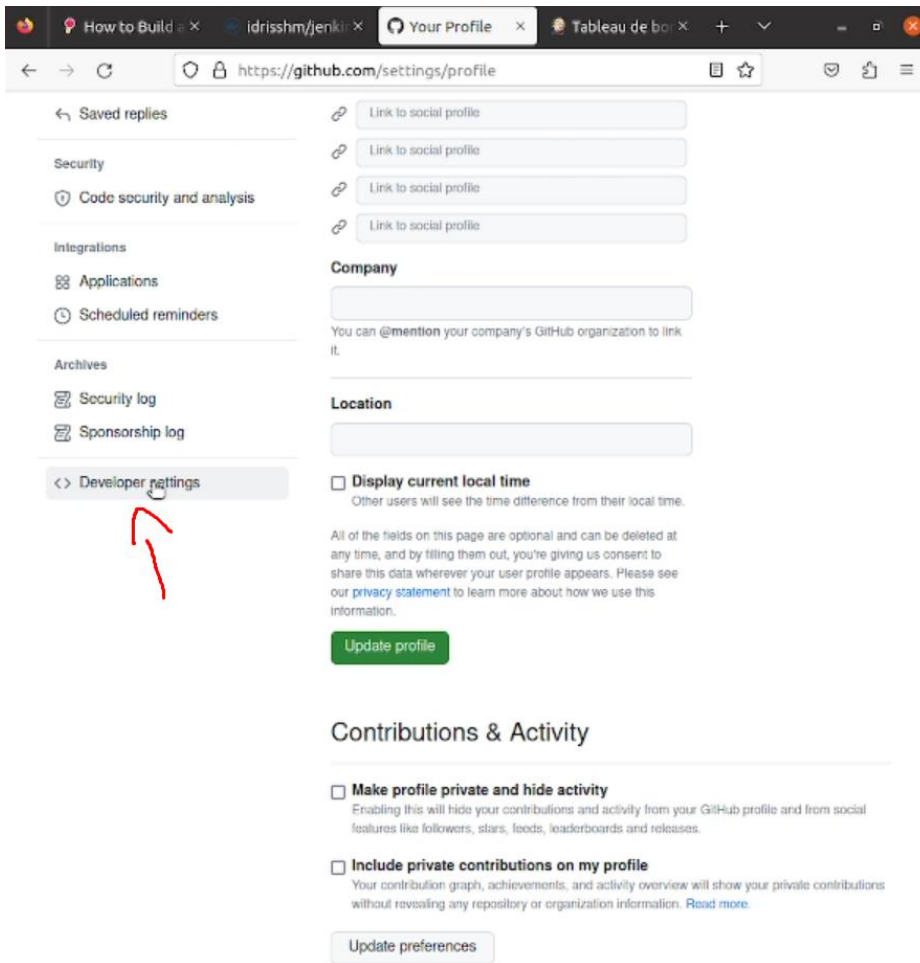
```
root@idriiss-VirtualBox:~/jenkins-docker# git config --global user.email idriiss.h
md62@gmail.com
root@idriiss-VirtualBox:~/jenkins-docker# git config --global user.name idriiss
root@idriiss-VirtualBox:~/jenkins-docker#
root@idriiss-VirtualBox:~/jenkins-docker# git commit -m "first commit"
[master (commit racine) f5086df] first commit
2 files changed, 38 insertions(+)
 create mode 100644 Dockerfile
 create mode 100644 Jenkinsfile
root@idriiss-VirtualBox:~/jenkins-docker#
root@idriiss-VirtualBox:~/jenkins-docker# git branch -M main
root@idriiss-VirtualBox:~/jenkins-docker# git remote add origin https://github.co
m/idriissm/jenkins-docker2.git
root@idriiss-VirtualBox:~/jenkins-docker# git push -u origin main
Username for 'https://github.com': idriissm
Password for 'https://idriissm@github.com':
remote: Support for password authentication was removed on August 13, 2021.
remote: Please see https://docs.github.com/en/get-started/getting-started-with-g
it/about-remote-repositories#cloning-with-https-urls for information on currentl
y recommended modes of authentication.
fatal: Échec d'authentification pour 'https://github.com/idriissm/jenkins-docker
2.git/'
```

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

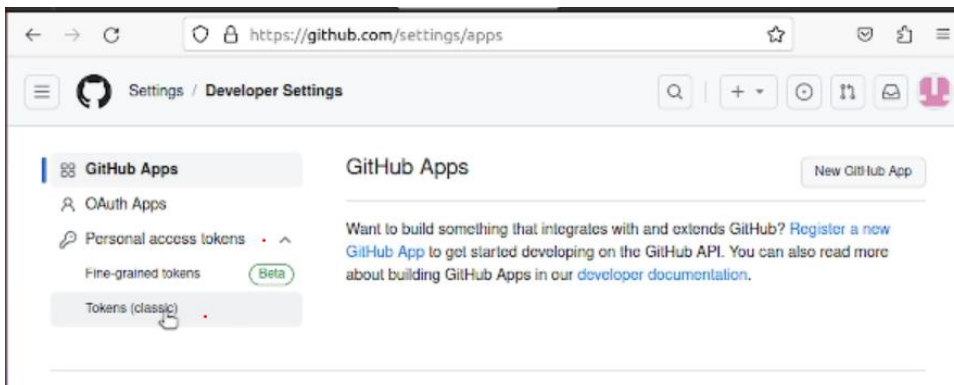
Aller sur votre compte GitHub







The screenshot shows the GitHub profile settings page at <https://github.com/settings/profile>. The left sidebar contains navigation options: Saved replies, Security, Code security and analysis, Integrations, Applications, Scheduled reminders, Archives, Security log, Sponsorship log, and Developer settings. A red arrow points to the 'Developer settings' option. The main content area includes sections for 'Link to social profile' (four instances), 'Company', 'Location', 'Display current local time' (checkbox), and 'Contributions & Activity' (checkboxes for 'Make profile private and hide activity' and 'Include private contributions on my profile'). A green 'Update profile' button is visible at the bottom of the profile section.



The screenshot shows the GitHub Developer Settings page at <https://github.com/settings/apps>. The breadcrumb navigation shows 'Settings / Developer Settings'. The left sidebar lists 'GitHub Apps', 'OAuth Apps', 'Personal access tokens' (with a sub-menu for 'Fine-grained tokens' and 'Tokens (classic)'), and 'Tokens (classic)'. The main content area is titled 'GitHub Apps' and includes a 'New GitHub App' button and introductory text about building GitHub Apps.



The screenshot shows the GitHub Developer Settings page for 'Personal access tokens (classic)'. A red arrow points to the 'Generate new token' button. Below the button, there are two existing tokens: 'tinquiete' and 'test'. Each token entry shows its name, the scopes it has (e.g., admin:enterprise, admin:org_key), when it was last used, and a 'Delete' button. The expiration date for 'tinquiete' is Wed, Jul 19 2023, and for 'test' it is Sat, Jul 15 2023.

Cocher toutes les cases pour donner tous les droits

The screenshot shows the 'New Personal Access Token' page on GitHub. All checkboxes for permissions are selected. The permissions listed are:

- repo:invite - Access repository invitations
- security_events - Read and write security events
- workflow - Update GitHub Action workflows
- write:packages - Upload packages to GitHub Package Registry
- read:packages - Download packages from GitHub Package Registry
- delete:packages - Delete packages from GitHub Package Registry
- admin:org - Full control of orgs and teams, read and write org projects
 - write:org - Read and write org and team membership, read and write org projects
 - read:org - Read org and team membership, read org projects
 - manage_runners:org - Manage org runners and runner groups
- admin:public_key - Full control of user public keys
 - write:public_key - Write user public keys
 - read:public_key - Read user public keys
- admin:repo_hook - Full control of repository hooks
 - write:repo_hook - Write repository hooks
 - read:repo_hook - Read repository hooks
- admin:org_hook - Full control of organization hooks
- gist - Create gists
- notifications - Access notifications
- user - Update ALL user data
 - read:user - Read ALL user profile data
 - user:email - Access user email addresses (read-only)
 - user:follow - Follow and unfollow users



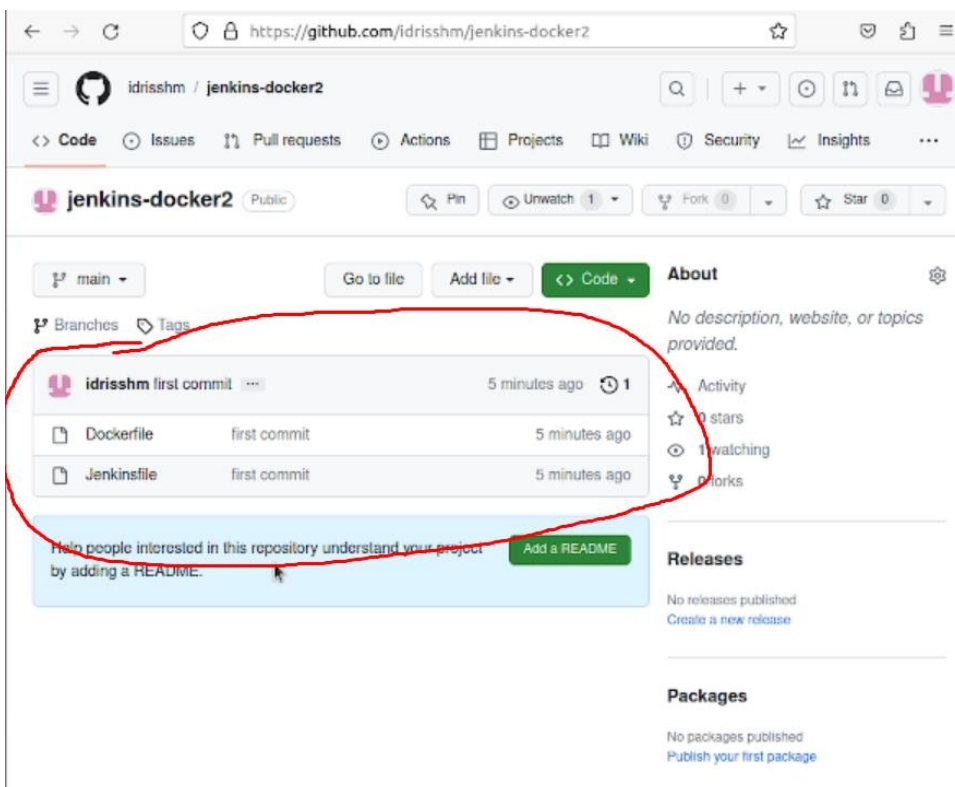
Copier la clé token



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

```
root@idrisss-VirtualBox:~/jenkins-docker# git push -u origin main
Username for 'https://github.com': idrisshm
Password for 'https://idrisshm@github.com':
Énumération des objets: 4, fait.
Décompte des objets: 100% (4/4), fait.
Compression des objets: 100% (3/3), fait.
Écriture des objets: 100% (4/4), 557 octets | 557.00 Kio/s, fait.
Total 4 (delta 0), réutilisés 0 (delta 0)
To https://github.com/idrisshm/jenkins-docker2.git
 * [new branch]      main -> main
La branche 'main' est paramétrée pour suivre la branche distante 'main' depuis 'origin'.
root@idrisss-VirtualBox:~/jenkins-docker#
```

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



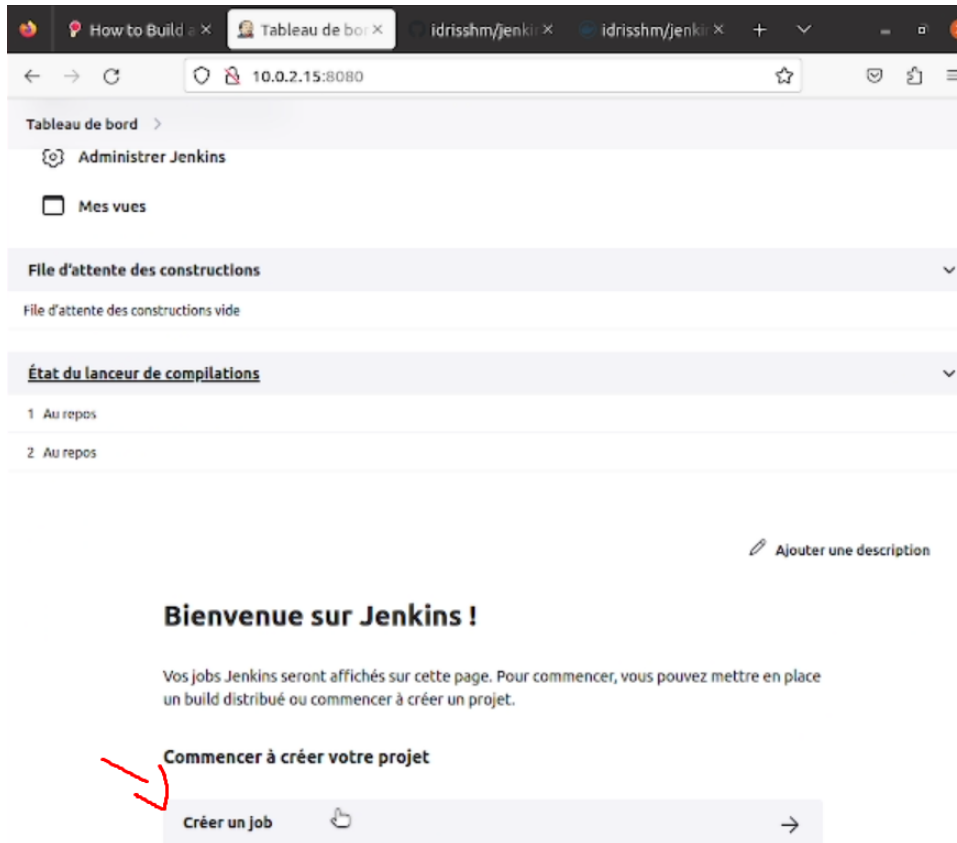
Maintenant que les fichiers sont poussés vers GitHub, on va pouvoir procéder à la dernière étape du processus.



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

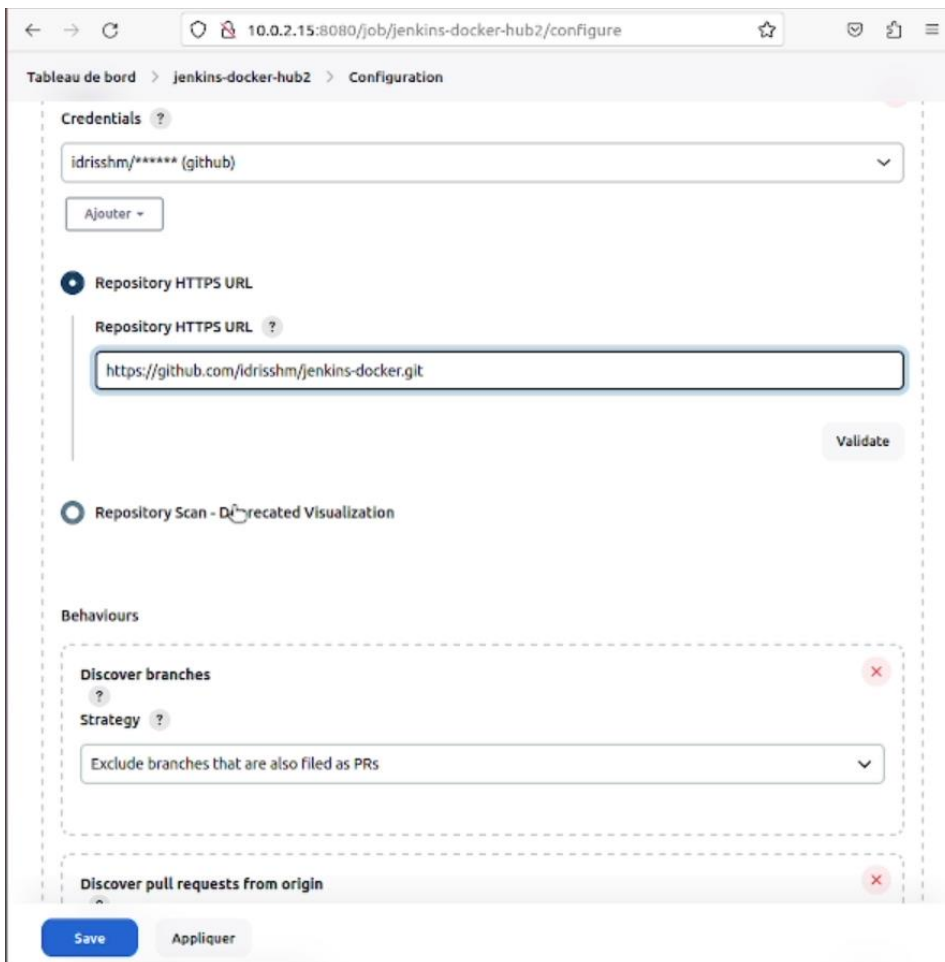
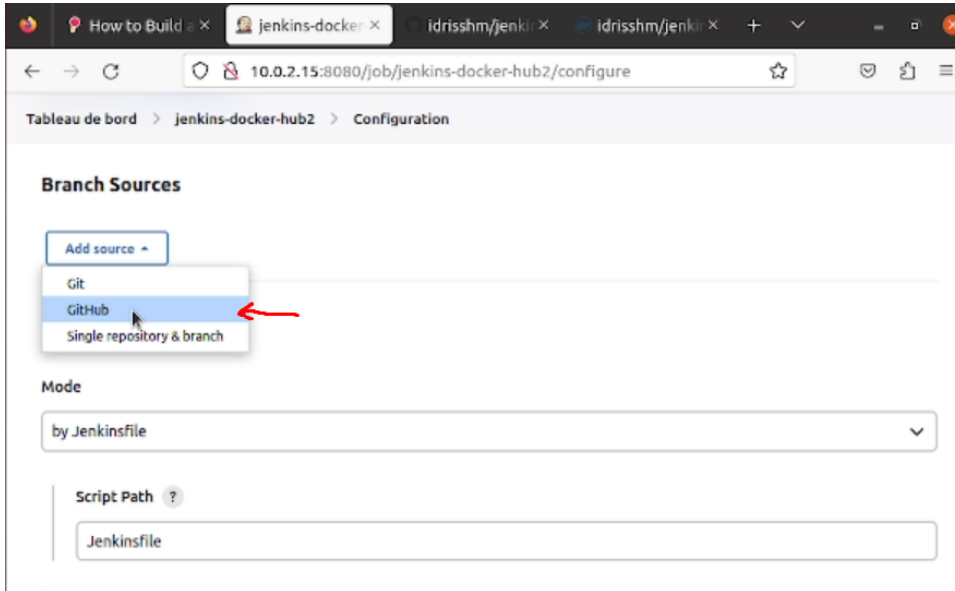


Sélectionner "Multibranch Pipeline", puis cliquez sur `OK`



Configurer le pipeline

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



Credentials ok. Connected to <https://github.com/idrissm/jenkins-docker>.

Validate

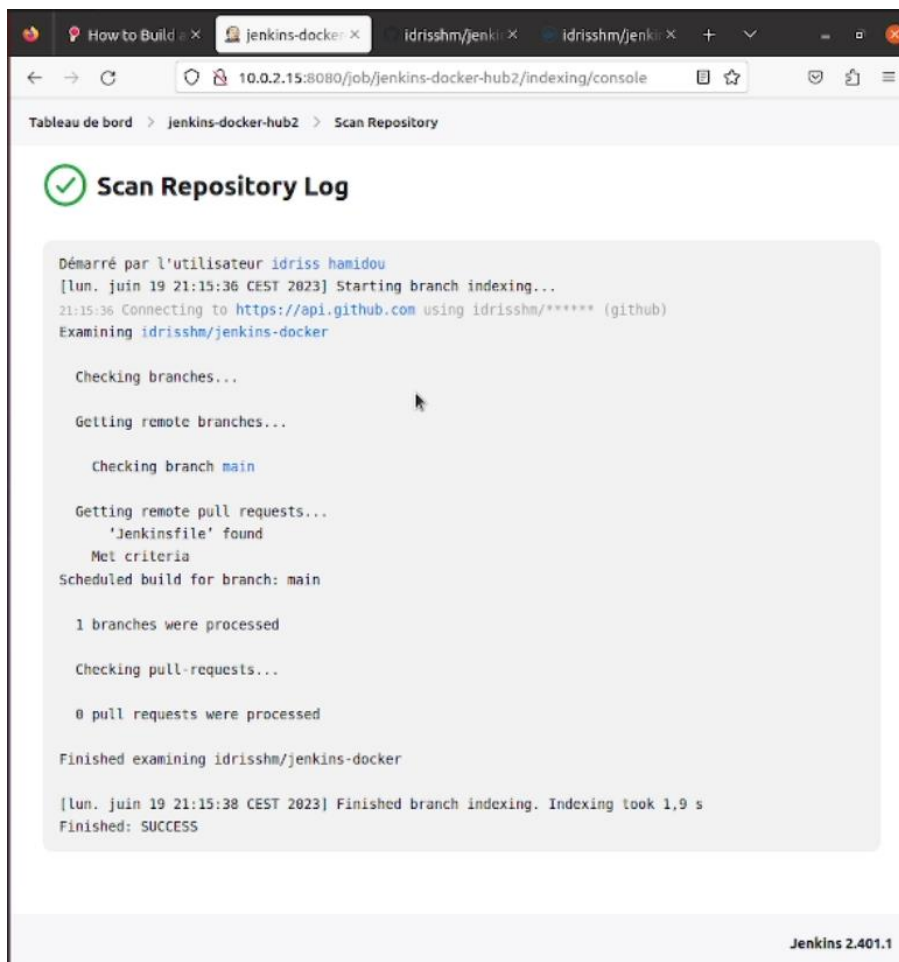
Save

Appliquer

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 :

 Scan Repository Log



```
Tableau de bord > jenkins-docker-hub2 > Scan Repository

[✓] Scan Repository Log

Démarré par l'utilisateur idriss hamidou
[lun. juin 19 21:15:36 CEST 2023] Starting branch indexing...
21:15:36 Connecting to https://api.github.com using idrissm/***** (github)
Examining idrissm/jenkins-docker

  Checking branches...

  Getting remote branches...

    Checking branch main

  Getting remote pull requests...
  'Jenkinsfile' found
  Met criteria
  Scheduled build for branch: main

  1 branches were processed

  Checking pull-requests...

  0 pull requests were processed

Finished examining idrissm/jenkins-docker

[lun. juin 19 21:15:38 CEST 2023] Finished branch indexing. Indexing took 1,9 s
Finished: SUCCESS

Jenkins 2.401.1
```

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.



État du lanceur de compilations

1 Au repos

2

jenkins-docker-hub2 » main	#1 (Déclarative: Checkout SCM)	
jenkins-docker-hub2 » main	#1	

Scan Repository Log

```
Démarré par l'utilisateur idriss hamidou
[lun. juin 19 21:15:36 CEST 2023] Starting branch indexing...
21:15:36 Connecting to https://api.github.com using idrissm/***** (github)
Examining idrissm/jenkins-docker

Checking branches...

Getting remote branches...

Checking branch main

Getting remote pull requests...
'Jenkinsfile' found
Met criteria
Scheduled build for branch: main
```

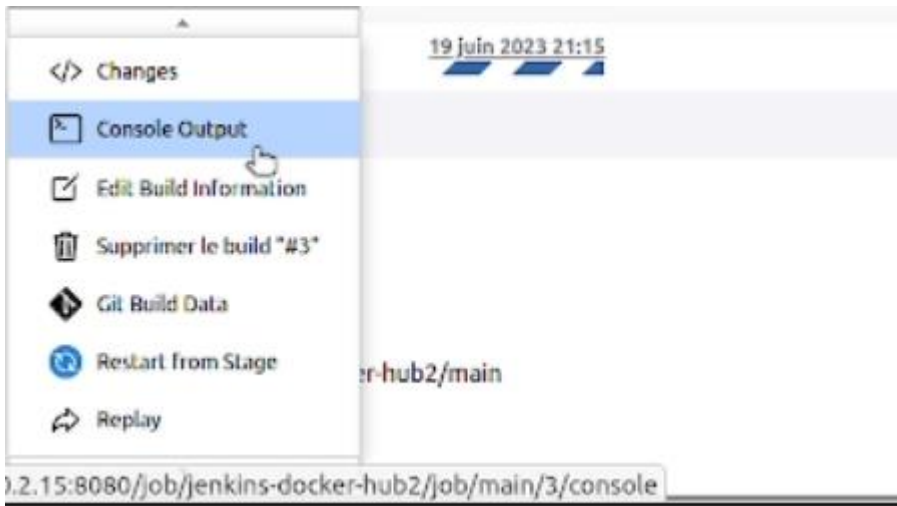
6.3 Exécution du pipeline Jenkins

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

2

jenkins-docker-hub2 » main	#1 (Déclarative: Checkout SCM)	
jenkins-docker-hub2 » main	#1	





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

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

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

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

Retournez sur jenkins

Replay pour relancer le script



Replay #4

Allows you to replay a Pipeline build with a modified script. If any Load steps were run, you can also modify the scripts they loaded.

Main Script

```
20 -     steps {
21 -         sh 'echo $DOCKERHUB_CREDENTIALS_PSW | docker login -u $DOCKERHUB_CREDENTIALS_USR --password-stdin'
22 -     }
23 - }
24 -
25 - stage('Push') {
26 -     steps {
27 -         sh 'docker push idrisshn/jenkins-docker-hub2'
28 -     }
29 - }
30 -
31 -
32 - post {
33 -     always {
34 -         sh 'docker logout'
35 -     }
36 - }
37 -
38 - }
```

Pipeline Syntax



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 idrissm/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 idrissm/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		

```

[Pipeline] sh
+ docker push idrissm/jenkins-docker-hub2
Using default tag: latest
The push refers to repository [docker.io/idrissm/jenkins-docker-hub2]
b2d5eeceaba3a: Preparing
b2d5eeceaba3a: Mounted from library/alpine
latest: digest: sha256:1732abb7b121d97dfc463ffa6bf2a27208b5bc57920ede33201f2af4ced2d000 size: 527
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Declarative: Post Actions)
[Pipeline] sh
+ docker logout
Removing login credentials for https://index.docker.io/v1/
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withCredentials
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline

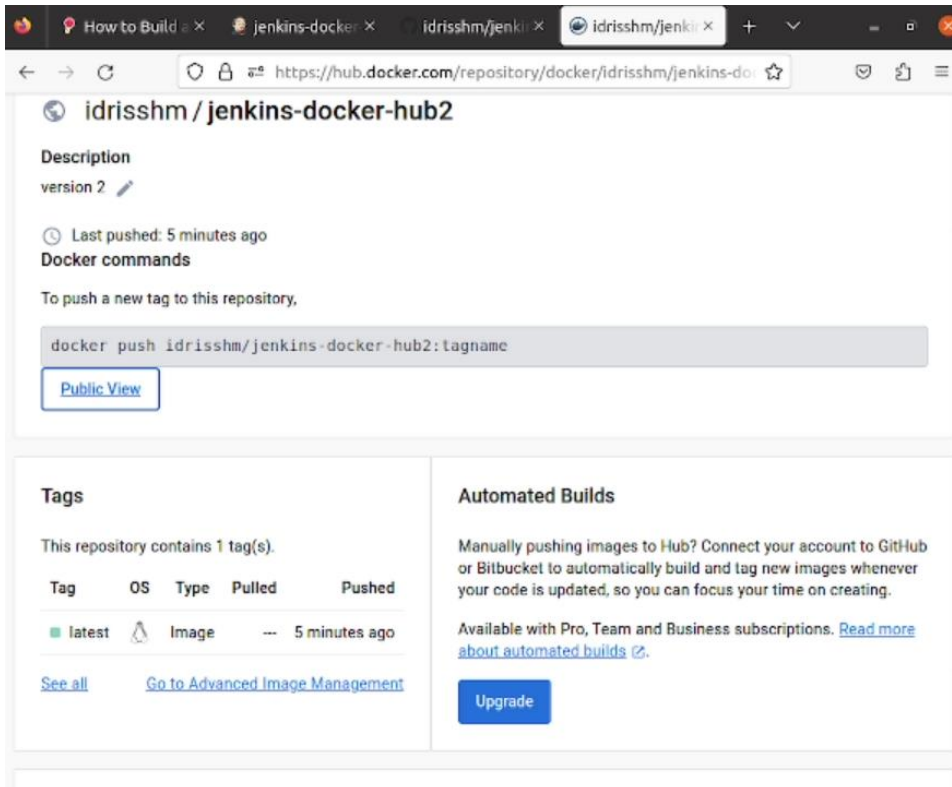
Could not update commit status, please check if your scan credentials belong to a member of the
organization or a collaborator of the repository and repo:status scope is selected

GitHub has been notified of this commit's build result

Finished: SUCCESS
    
```



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.

