

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE



UNIVERSITE CHEIKH ANTA DIOP DE DAKAR

FACULTE DES SCIENCES ET TECHNIQUES

DEPARTEMENT DE MATHEMATIQUES ET INFORMATIQUE SECTION

PROJET DE : GÉNIE LOGICIEL M1 SIR

PARTICIPANTS :

Mame Diarra Diouf

Wore Diouf

Oumy Ndong

PLAN :

- I. Introduction
- II. Processus de création du projet java avec maven
- III. Explication des méthodes et test unitaires
- IV. Explication de la mise à disposition du code sur GitHub et la création des branches
- V. Explication la création du fichier DockerFile
- VI. Explication de la mise en place d'un CI/CD avec GitHub actions
- VII. Installation des outils Maven, Git, Docker
- VIII. Conclusion

I. INTRODUCTION

Le génie logiciel (software engineering) représente l'application de principes d'ingénierie au domaine de la création de logiciels. Il consiste à identifier et à utiliser des méthodes, des pratiques et des outils permettant de maximiser les chances de réussite d'un projet logiciel.

Dans notre étude on va faire l'usage de GitHub et github actions avec notre projet maven qu'on va créer.

II. PROCESSUS DE CRÉATION DU PROJET JAVA AVEC MAVEN

On a créé dans IntelliJ IDEA Community un projet et comme on a la possibilité de choisir quel type de projet on veut créer on choisit l'option maven afin de pouvoir créer notre projet maven et donner le nom de notre projet. Dans notre projet on a un fichier **pom.xml** dans lequel on va faire quelques configurations c'est-à-dire ajouter des dépendances comme **junit** et des plugins comme **maven-compiler-plugin**.

III. EXPLICATION DES MÉTHODES ET TEST UNITAIRES

Dans notre projet on a une classe **Calculator** qui va contenir l'ensemble des méthodes de notre programme. Nous avons 8 méthodes dans cette classe :

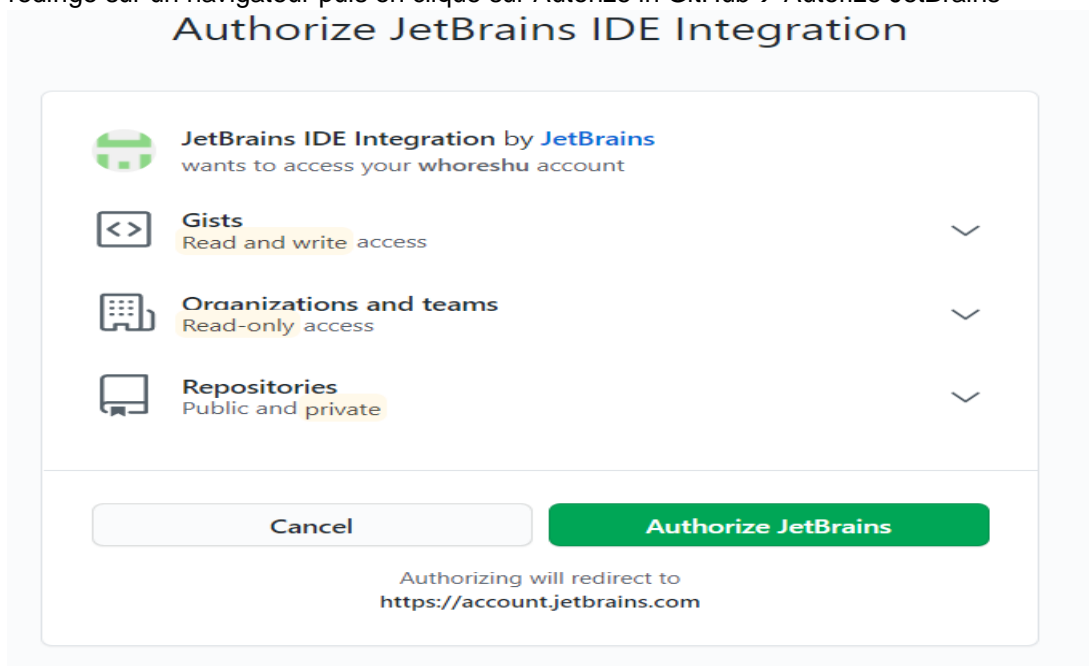
- La méthode **sum** qui consiste à faire la somme de deux entiers qu'elle va prendre en paramètre.
- La méthode **sus** qui consiste à faire la différence de deux entiers qu'elle va prendre en paramètre.
- La méthode **divide** qui consiste à faire la division de deux entiers qu'elle va prendre en paramètre.
- La méthode **multiply** qui consiste à faire le produit de deux entiers qu'elle va prendre en paramètre.
- La méthode **min** qui retourne le minimum entre les deux entiers qu'elle va prendre en paramètre.
- La méthode **max** qui retourne le maximum entre les deux entiers qu'elle va prendre en paramètre.
- La méthode **minElement** qui retourne le plus petit élément d'une liste qu'elle va prendre en paramètre.
- La méthode **maxElement** retourne le plus grand élément d'une liste qu'elle va prendre en paramètre.

Pour chacune de ces méthodes on va écrire des tests unitaires dans une classe appelée **CalculatorTest**. Donc on aura 8 tests et chaque cas de test sera mise dans une méthode à savoir : **testSum**, **testMultiply**, **testDivide**, **testSus**, **testMinimum**, **testMaximum**, **testMaximumElt** et **testMinimumElt**.

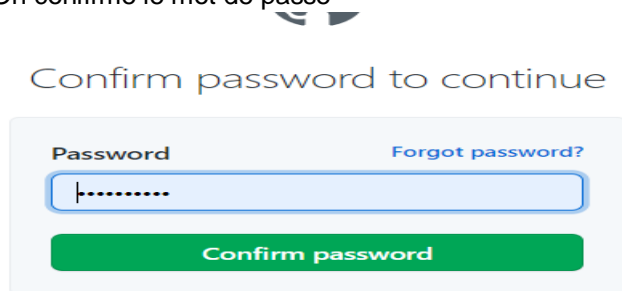
IV. EXPLICATION DE LA MISE À DISPOSITION DU CODE SUR GITHUB ET LA CRÉATION DES BRANCHES

Après avoir créé notre compte GitHub, on crée notre projet maven dans IntelliJ puis on configure notre IDE avec GitHub en suivant les étapes suivantes :

- On appuie sur l'onglet file → setting → Version Control → GitHub → add account et on nous redirige sur un navigateur puis on clique sur Authorize in GitHub → Authorize JetBrains

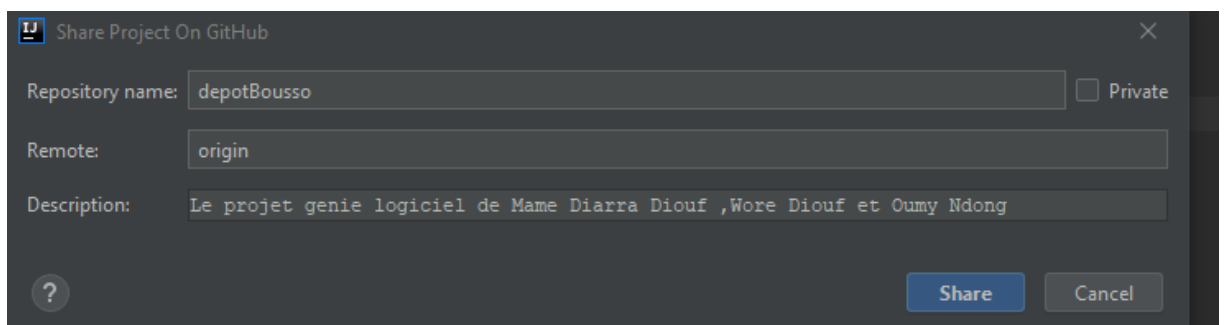


- On confirme le mot de passe

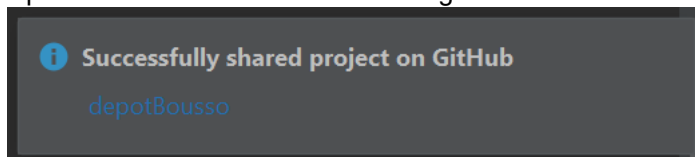


Tip: You are entering **sudo mode**. We won't ask for your password again for a few hours.

Après on va sur IntelliJ pour envoyer le projet dans un repository sur GitHub



Après l'envoi on aura comme message :



Après avoir mis le code dans GitHub on va envoyer les invitations aux collaborateurs en allant sur l'onglet **settings** pour qu'ils puissent accéder au repository et cloner le projet.

github.com/bouso-cell/depotBouso/settings/access

Applications Gmail YouTube Maps

Options

- Manage access
- Security & analysis
- Branches
- Webhooks
- Notifications
- Integrations
- Deploy keys
- Actions
- Secrets
- Moderation settings

Who has access

PUBLIC REPOSITORY

This repository is public and visible to anyone.

[Manage](#)

DIRECT ACCESS

2 have access to this repository. [2 collaborators](#).

Manage access

[Invite a collaborator](#)

☐ Select all

Type ▾

Find a collaborator...

<input type="checkbox"/>	Oumyshu Collaborator	
<input type="checkbox"/>	whoreshu Collaborator	

Chacune de nous doit créer une branche qu'on peut voir sur la capture suivante :

master ▾

Switch branches/tags

Find or create a branch...

Branches Tags

- ✓ master (default)
- BranchDiarra
- BranchOumy
- BranchWore

[View all branches](#)

Verified	3fbae20	<>
	eb149c8	<>
	a621e58	<>
Verified	ea6e562	<>

On doit faire des modifications sur le projet, donc la capture ci-dessous va montrer les commit qu'on a pu faire sur nos branches :

bouso-cell / depotBouso

Watch 1 Star 0 Fork 0

Code Issues Pull requests Actions Projects Wiki Security Insights

master ▾

Commits on Nov 28, 2020

Update maven.yml bousso-cell committed 27 minutes ago ✓	Verified 3fbae20 <>
modifiWore whoreshu committed 2 hours ago ✓	eb149c8 <>
TestDeWoreDiouf whoreshu committed 2 hours ago ✓	a621e58 <>
Update maven.yml Oumyshu committed 2 hours ago ✓	Verified ea6e562 <>
CommitOumy Oumyshu committed 2 hours ago ✓	3a43b2b <>
premier commit de diarra bousso-cell committed 10 hours ago ✓	3175514 <>

V. EXPLICATION LA CRÉATION DU FICHIER DOCKERFILE

Dans notre projet on a créé un fichier DockerFile qui va contenir le fichier.jar de notre projet dans le répertoire target avec la commande **mvn package** qui sera notre JAR-FILE dans notre dockerfile.

On va créer une image docker1 pour ce dockerfile.

```
C:\Users\Oumy\Documents\Git>docker build -t docker1:1.0.0 .
[+] Building 9.4s (9/9) FINISHED
=> [internal] load .dockerignore
=> => transferring context: 28
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 400B
=> [internal] load metadata for docker.io/library/openjdk:8-jdk-alpine
=> [internal] load build context
=> => transferring context: 78.74kB
=> [1/4] FROM docker.io/library/openjdk:8-jdk-alpine@sha256:94792824df2df33402f201713f932b58cb9de94a0cd524164a
=> CACHED [2/4] WORKDIR /usr/local/runme
=> [3/4] COPY target/Git-1.0-SNAPSHOT.jar app.jar
=> [4/4] ADD lib/
=> exporting to image
=> => exporting layers
=> => writing image sha256:42bcc44d9cd428c5b137a7299002e562ddc275a73835bf1f4b2e3582510edaf7
=> => naming to docker.io/library/docker1:1.0.0

C:\Users\Oumy\Documents\Git>docker run -d docker1:1.0.0
6c90f1308b9781c717fa68b3daad758fc6fa4a0b35da9a7253192c069d403116
```

On va vérifier si l'image docker1 est bien créée

```
C:\Users\Oumy\Documents\Git>docker run -d docker1:1.0.0
6c90f1308b9781c717fa68b3daad758fc6fa4a0b35da9a7253192c069d403116

C:\Users\Oumy\Documents\Git>docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
docker1             1.0.0              42bcc44d9cd4       12 minutes ago     105MB
tp-sir              1.0.0              dde3304740e4       13 days ago        105MB
couchbase           latest             c64844065dcb       4 weeks ago        1.18GB
neo4j               latest             750bf9bc2374       4 weeks ago        541MB
mysql               latest             db2b37ec6181       4 weeks ago        545MB
redis               latest             bd571e6529f3       5 weeks ago        104MB
alpine/git          latest             94f8849864da       2 months ago       28.4MB
hello-world         latest             bf756fb1ae65       10 months ago      13.3kB
liliasfaxi/spark-hadoop hv-2.7.2          d64a47823a96       21 months ago      1.94GB
mongo               4.0.4              525bd2016729       2 years ago        383MB
alexwhen/docker-2048 latest             7929bcd70e47       5 years ago        8.02MB

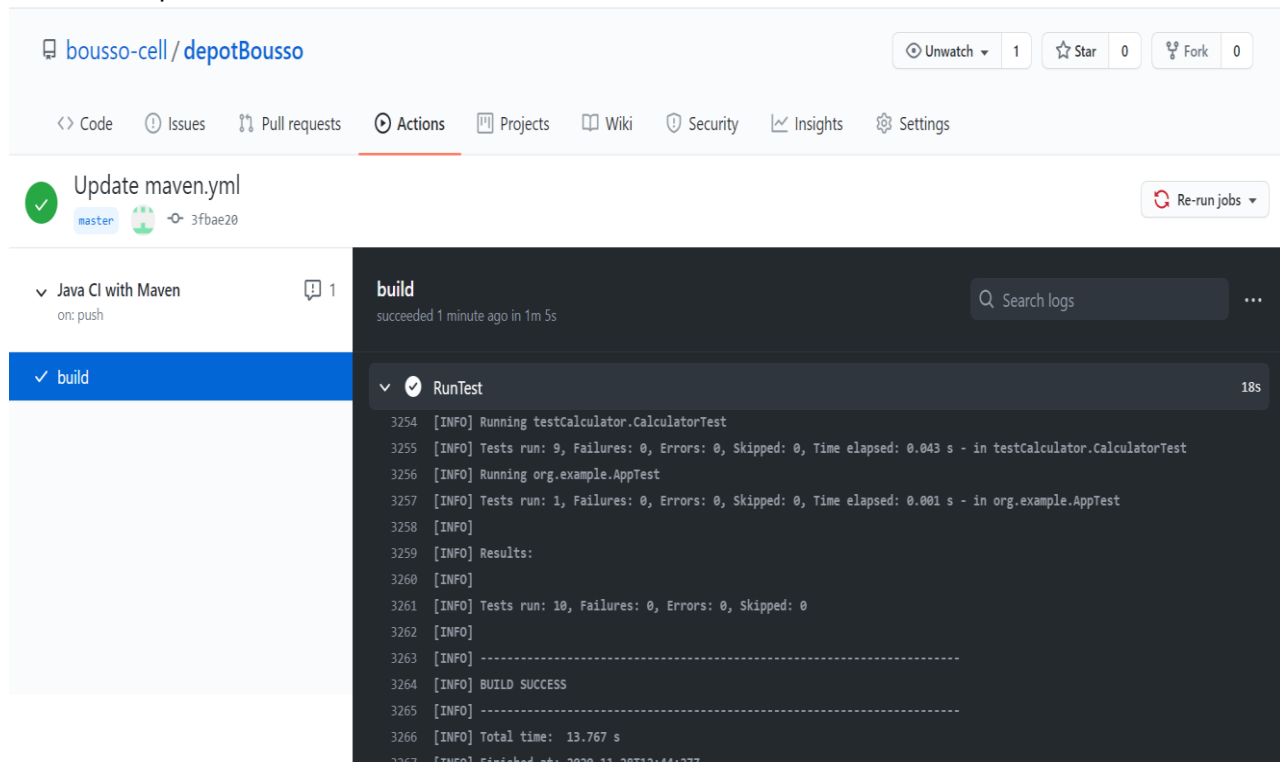
C:\Users\Oumy\Documents\Git>
```

VI. EXPLICATION DE LA MISE EN PLACE D'UN CI/CD AVEC GITHUB ACTIONS

• MISE EN PLACE D'UN CI AVEC GITHUB ACTIONS

- Pour mettre en place l'outil d'intégration continue avec github actions on doit d'abord avoir un environnement de travail appelé **workflows** en allant sur l'onglet **Actions**. On choisit **Java with Maven** pour notre workflow et on donne le nom de notre fichier.yml et faire **start commit** sur la branche master et puis **commit new file**.

- Pour lancer les tests unitaires avec GitHub actions on va dans notre fichier.yml et ajouter
- **name: RunTest**
 - run: mvn test**
- Maintenant quand on fait le build on obtient :



• MISE EN PLACE D'UN CD AVEC GITHUB ACTION

D'abord on crée un compte DockerHub avec comme username doker1997 puis on crée un nouveau repository nommer depotmame. Ensuite on va dans **account settings** → **Security** → **New Access Token** pour créer une clé de sécurité.

On va dans notre fichier.yml ajouter le contenu suivant :

- name: Build and Push Docker Image
- uses: mr-smithers-excellent/docker-build-push@v4
- with:
- image: doker1997/depotmame
- registry: docker.io
- username: \${secrets.DOCKER_USERNAME}}
- password: \${secrets.DOCKER_PASSWORD}}

Maintenant on va dans : settings → secrets → New repository secret pour créer un des repository pour Docker_Username on met le username du compte DockerHub et Docker_Password on met le contenu de la clé qu'on avait créé.

Quand on fait le build on obtient les captures suivantes :

github.com/bousso-cell/depotBousso/runs/1467264020?check_suite_focus=true

Applications Gmail YouTube Maps

Java CI with Maven
on: push

✓ build

build
succeeded 2 minutes ago in 1m 5s

Search logs

✓ Build and Push Docker Image 33s

```

93 114ca5b7280f: preparing
94 4b9227ba273c: Waiting
95 712264374d24: Waiting
96 475b4eb79695: Waiting
97 f3be340a54b9: Waiting
98 114ca5b7280f: Waiting
99 837a1c626ec1: Layer already exists
100 a7f518c13c4f: Layer already exists
101 4b9227ba273c: Layer already exists
102 712264374d24: Layer already exists
103 475b4eb79695: Layer already exists
104 f3be340a54b9: Layer already exists
105 114ca5b7280f: Layer already exists
106 ca3b29e3565: Pushed
107 7731a2e99063: Pushed
108 82d54568dfad: Pushed
109 master-3fbae20: digest: sha256:619dbafc51828eeffec6aaca6393ceadff68e04a2bbc816953b271992c19e1af4 size: 2416
  
```

> ✓ Post Set up JDK 1.8 0s

> ✓ Post Run actions/checkout@v2 1s

> ✓ Complete job 0s

https://hub.docker.com/repository/docker/docker1997/depotmame

Pull rate limits for certain users are being introduced to Docker Hub starting November 2nd. [Learn more](#)

dockerhub Search for great content (e.g., mysql) Explore Repositories Organizations Get Help docker1997

Repositories docker1997 / depotmame Using 0 of 1 private repositories. [Get more](#)

General Tags Builds Timeline Collaborators Webhooks Settings

docker1997 / depotmame
This repository does not have a description
Last pushed: an hour ago

Docker commands
To push a new tag to this repository,
`docker push docker1997/depotmame:tagname`

Tags and Scans
This repository contains 6 tag(s).
VULNERABILITY SCANNING - DISABLED [Enable](#)

TAG	OS	PULLED	PUSHED
master-3fbae20		an hour ago	an hour ago

Recent builds
[Link a source provider and run a build to see build results here.](#)

V. INSTALLATION DES OUTILS MAVEN, GIT, DOCKER

➤ MAVEN :

Maven est un outil de construction de projets (build) open source développée par la fondation Apache, initialement pour les besoins du projet Jakarta Turbine. Il permet de faciliter et d'automatiser certaines tâches de la gestion d'un projet Java.



- **TÉLÉCHARGEMENT :**

Il suffit de se rendre sur le site de MAVEN et de télécharger la version qu'on souhaite, au format que nous souhaitons. Dans le cadre de notre travail on a téléchargé la version 3.6.3 de **MAVEN**.

- **DÉCOMPRESSION :**

Vérifiez que l'on a bien téléchargé le binaire et non le source de **MAVEN** : le fichier téléchargé devrait être de la forme apache-maven-3.6.3-bin.

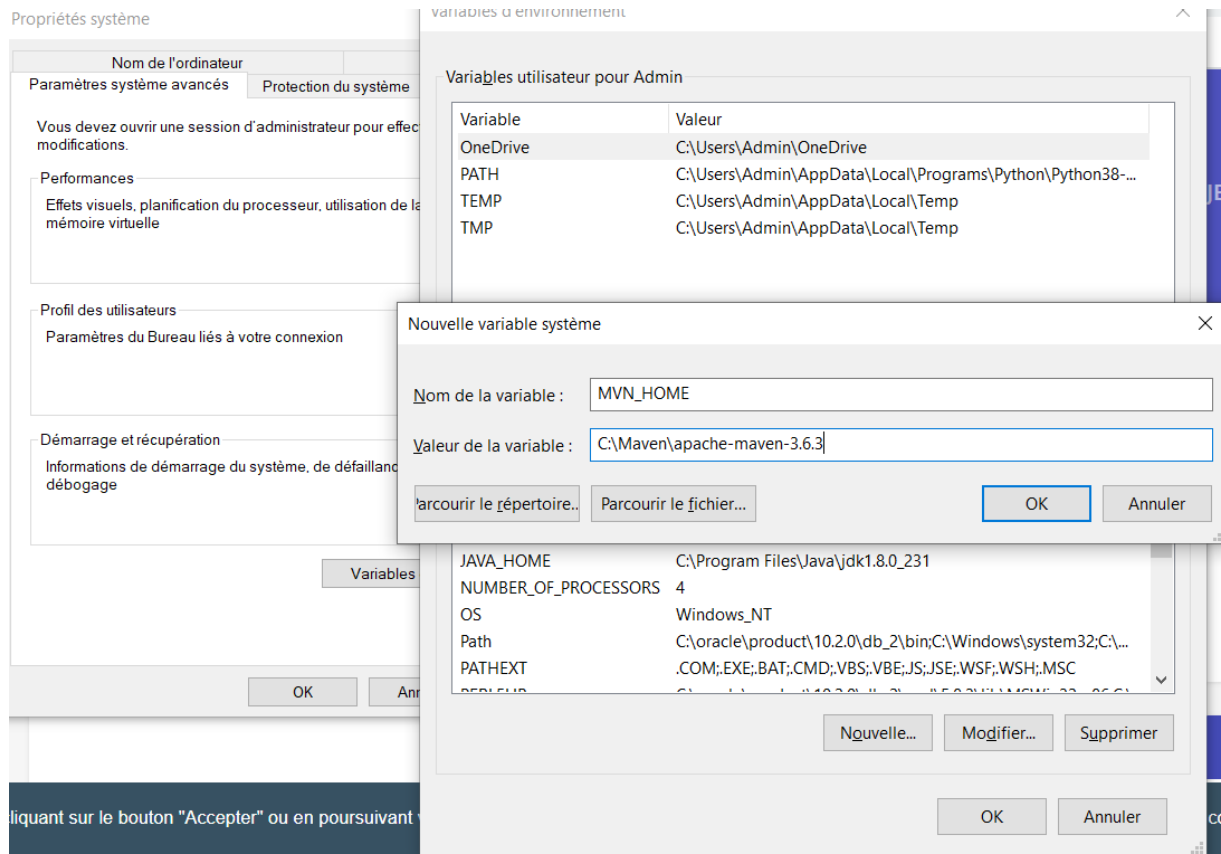
Décompressez l'archive téléchargée, dans le dossier d'installation de notre choix et on obtient :

	Nom	Modifié le	Type	Taille
	apache-maven-3.6.3	26/11/2020 23:54	Dossier de fichiers	
<input checked="" type="checkbox"/>	apache-maven-3.6.3.rar	27/11/2020 22:35	Archive WinRAR	9 230 Ko

- **LA VARIABLE D'ENVIRONNEMENT DE MVN :**

Ajoutez une variable d'environnement nommé **MVN_HOME**, celle-ci doit pointer sur le dossier d'installation de Maven : puisqu'on a installé/décompressé Maven dans le dossier

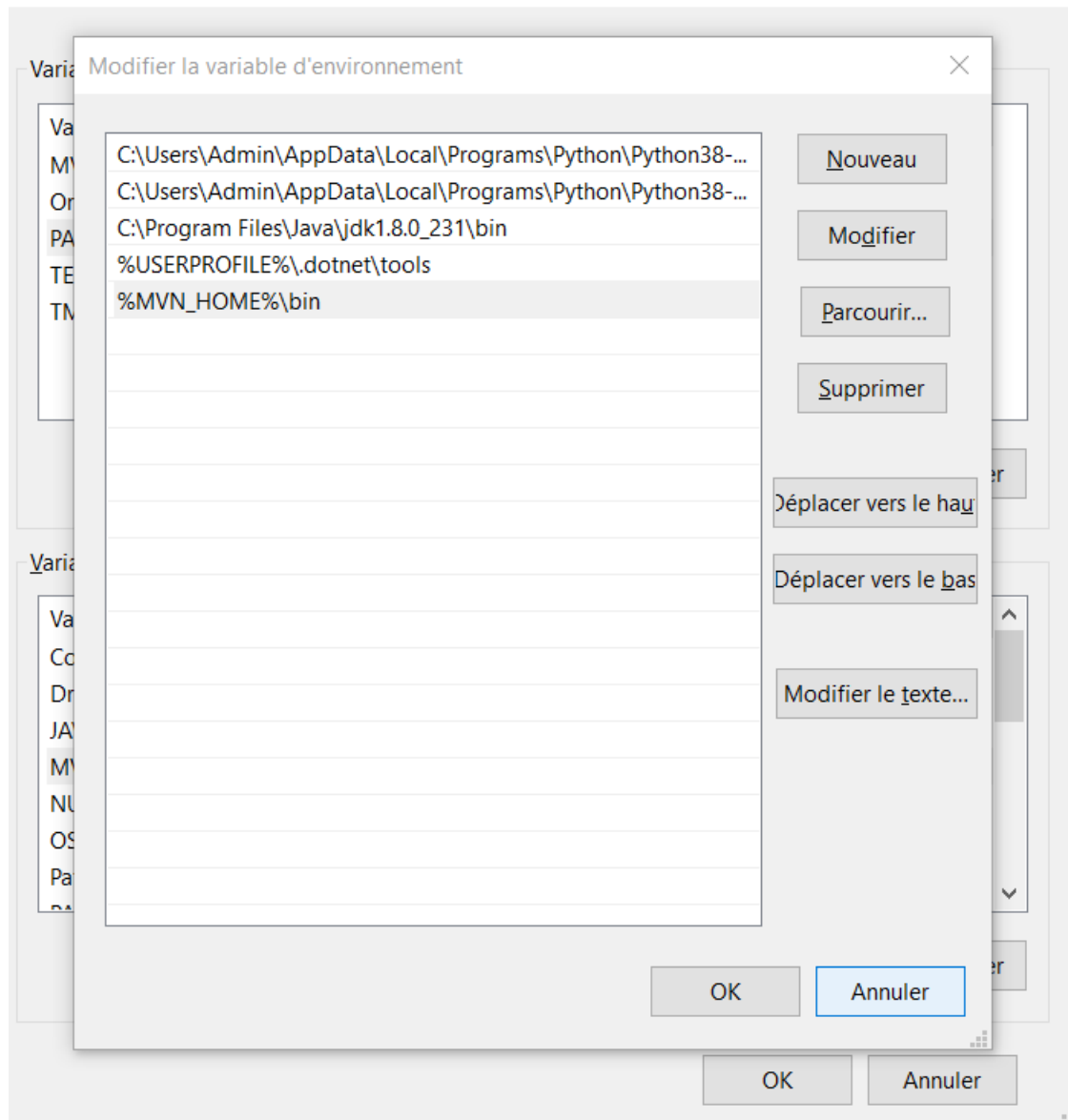
C:\Maven\apache-maven-3.6.3 alors on indique dans la valeur de la variable **MVN_HOME C:\Maven\apache-maven-3.6.3** comme suit :



• LA VARIABLE D'ENVIRONNEMENT PATH :

Il faut maintenant ajouter notre variable `MVN_HOME` à la variable `Path` de Windows. Ceci permettra d'invoquer directement la commande `mvn` sans indiquer le chemin complet.

Modifions la variable d'environnement Windows **Path**, en lui ajoutant le chemin pointant vers le dossier **bin** de votre dossier d'installation **MAVEN** : il suffit d'ajouter à la fin de la valeur existante la chaîne `%MVN_HOME%/bin`. Le `%MVN_HOME%` fait référence à la variable `MVN_HOME`, qu'on a créée ci-dessus. Et on obtient :



- **TEST DE L'INSTALLATION :**

Pour tester l'installation on ouvre une invite de commande cmd de Windows. On tape dans la ligne de commande `mvn -version`, on va voir s'afficher quelque chose de la même forme que ce qui suit :

```
C:\Windows\system32\cmd.exe
Microsoft Windows [version 10.0.18363.1198]
(c) 2019 Microsoft Corporation. Tous droits réservés.

C:\Users\Admin>mvn --version
Apache Maven 3.6.3 (cecedd343002696d0abb50b32b541b8a6ba2883f)
Maven home: C:\Maven\apache-maven-3.6.3\bin\..
Java version: 1.8.0_231, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk1.8.0_231\jre
Default locale: fr_FR, platform encoding: Cp1252
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"
```

➤ GIT :

Git est un **logiciel de gestion de version décentralisé** parmi les plus populaires avec **12 000 000 d'utilisateurs** dans le monde. C'est un logiciel libre créé par [Linus Torvald](#), auteur du noyau Linux. Git est donc totalement gratuit.



git

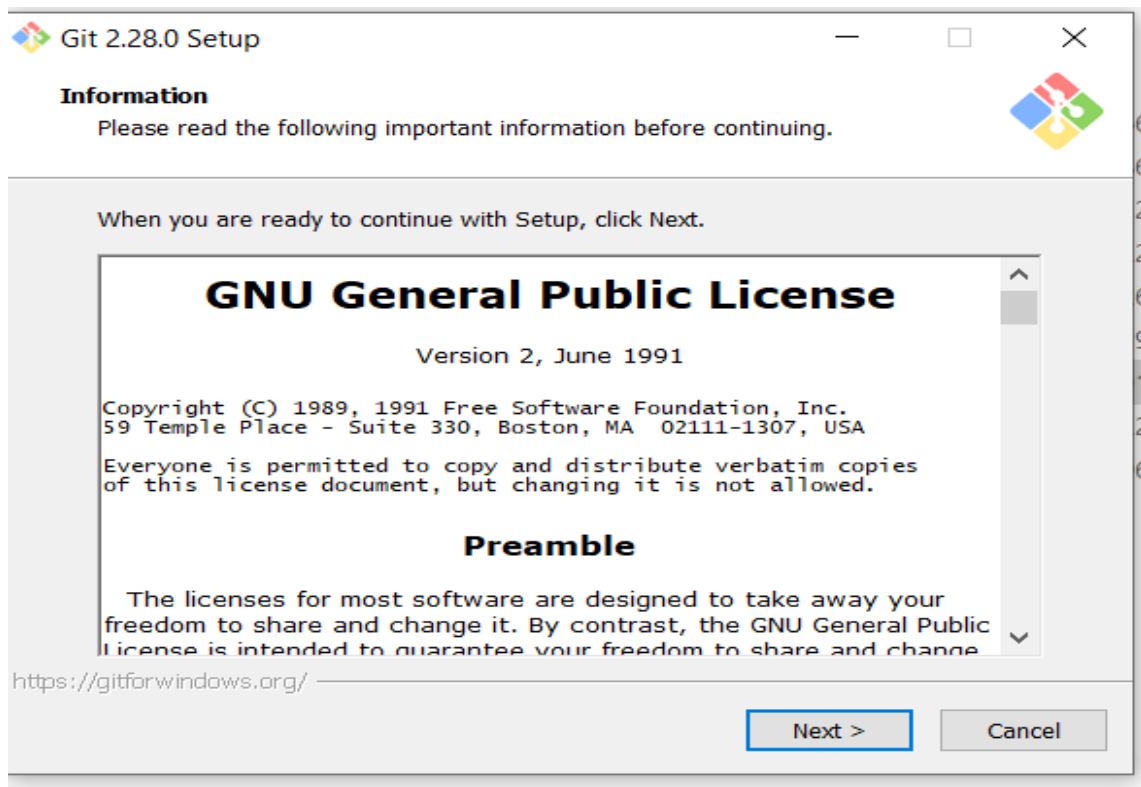
• TÉLÉCHARGEMENT :

Il faut comme pour tout autre logiciel qu'on avait déjà installé sur notre PC, télécharger l'installateur et le lancer.

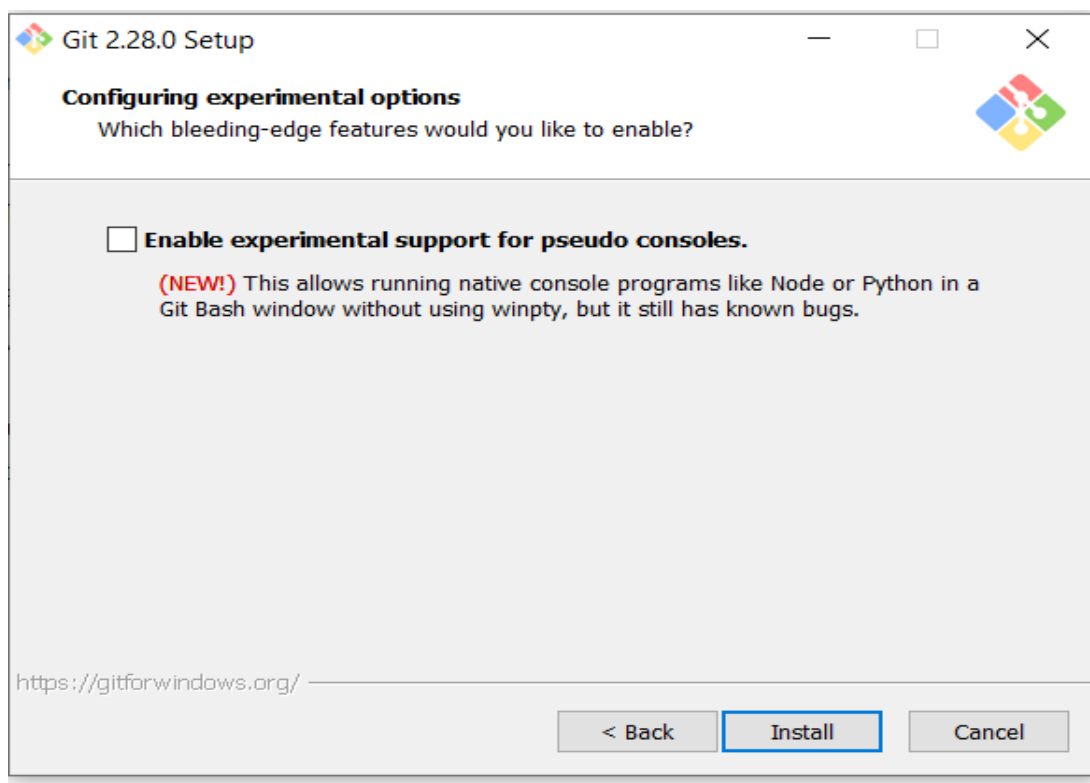
Téléchargez simplement le fichier .exe d'installation depuis la page officielle [GitHub](#), mais bien vérifier que la version qui vous est proposé, soit celle pour Windows.

• RÉALISER L'INSTALLATION :

Pour installer l'outil, on double clique simplement sur l'icône du fichier téléchargé. Une interface se lance alors :








Et nous propose, page après page, de configurer l'installation de Git. Il suffit de cliquer sur **Next** jusqu'à cette fenêtre ou on a :



Puis on clique sur **Install**.

Ainsi on a git installer et on peut voir les éléments suivants sur notre machine :

- L'outil Git.
- Git Bash : terminal qui vous permet d'utiliser git en ligne de commande.
- Git GUI : interface graphique qui permet de gérer les commits.
- Git CMD : interface graphique qui permet de gérer l'historique de votre dépôt.

<input checked="" type="checkbox"/>	 Git Bash	27/11/2020 23:42	Raccourci	2 Ko
	 Git CMD	27/11/2020 23:42	Raccourci	2 Ko
	 Git FAQs (Frequently Asked Questions)	27/11/2020 23:42	Raccourci Internet	1 Ko
	 Git GUI	27/11/2020 23:42	Raccourci	2 Ko
	 Git Release Notes	27/11/2020 23:42	Raccourci	2 Ko

• TEST DE L'INSTALLATION :

Vérifions maintenant que l'installation s'est bien déroulée. Nous allons utiliser une commande de base de l'outil qui est "git version" et qui permet d'afficher le numéro de version de Git.

Pour cela il faut lancer le terminal "git bash", installé en même temps que Git. Il permet d'utiliser Git en lignes de commandes.

Une fois le terminal lancé il ne nous reste plus qu'à taper la commande "git version". Et on a l'affichage suivant (au numéro de versions près) :

```

MINGW64:/c/Users/Admin

Admin@DESKTOP-NCTCU1K MINGW64 ~
$ git version
git version 2.28.0.windows.1

```

➤ DOCKER :

Docker est la plateforme de containers la plus populaire et la plus utilisée. Il s'agit d'une plateforme logicielle open source permettant **de créer, de déployer et de gérer des containers d'applications virtualisées sur un système d'exploitation.** Les services ou fonctions de l'application et ses différentes bibliothèques, fichiers de configuration, dépendances et autres composants sont regroupés au sein du container. Chaque container exécuté partage les services du système d'exploitation.



- **TÉLÉCHARGEMENT :**

On peut télécharger Docker Desktop pour Windows à partir de Docker Hub et on télécharge [Docker Desktop Installer.exe](#). En téléchargeant Docker Desktop, on accepte les termes du [contrat de licence utilisateur final du logiciel Docker](#) et du [contrat de traitement des données Docker](#).

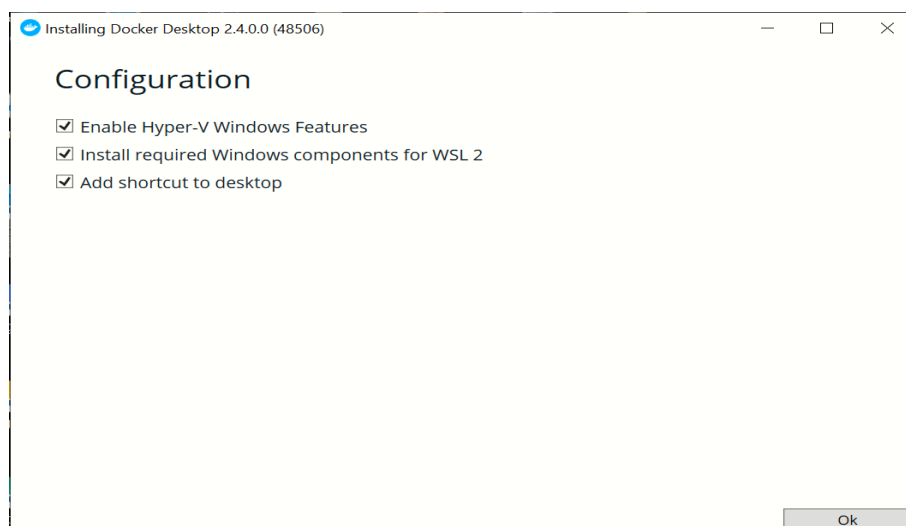
- **CONFIGURATION SYSTÈME REQUISE :**

- Windows 10 64 bits : Professionnel, Entreprise ou Éducation (Build 16299 ou version ultérieure).
- Les fonctionnalités Windows Hyper-V et Conteneurs doivent être activées.
- Les prérequis matériels suivants sont requis pour exécuter avec succès Client Hyper-V sur Windows 10 :
 - Processeur 64 bits avec traduction d'adresse de deuxième niveau (SLAT).
 - RAM système d'au moins 4Go.
 - La prise en charge de la virtualisation matérielle au niveau du BIOS doit être activée dans les paramètres du BIOS.

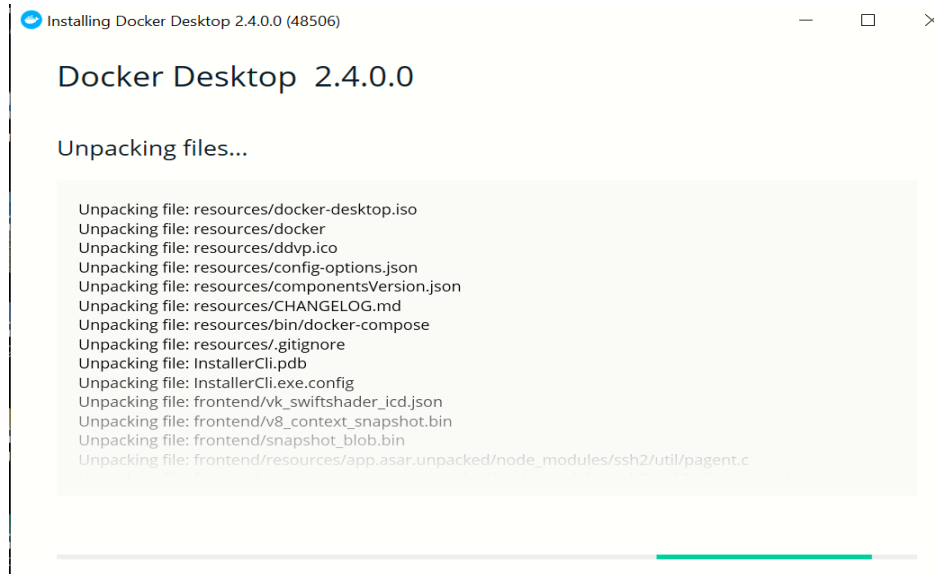
- **INSTALLATION :**

L'installation de Docker Desktop comprend [Docker Engine](#), le client Docker CLI, [Docker Compose](#), [Notary](#), [Kubernetes](#) et [Credential Helper](#).

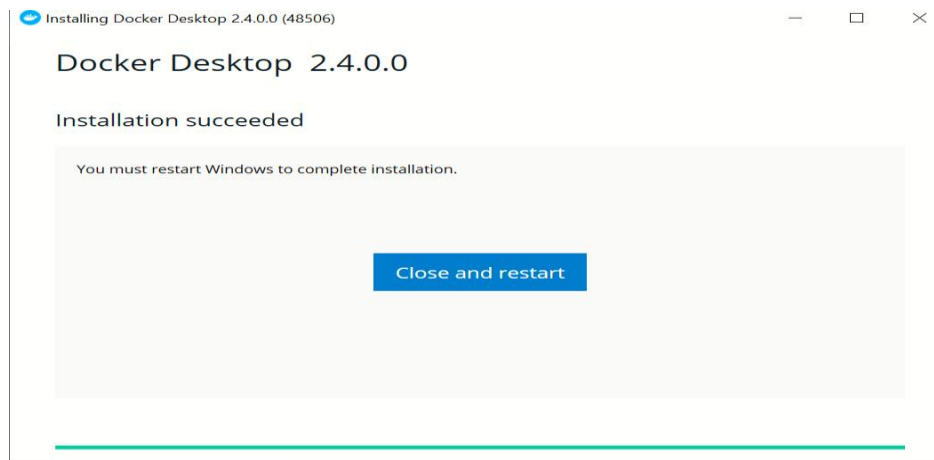
1. Double-cliquez sur **Docker Desktop Installer.exe** pour exécuter le programme d'installation.
2. Lorsque qu'on y est invité, on s'assure que l'option **Activer les fonctionnalités Windows Hyper-V** est sélectionnée sur la page de configuration.



3. On suit les instructions de l'assistant d'installation pour autoriser le programme d'installation et poursuivre l'installation.

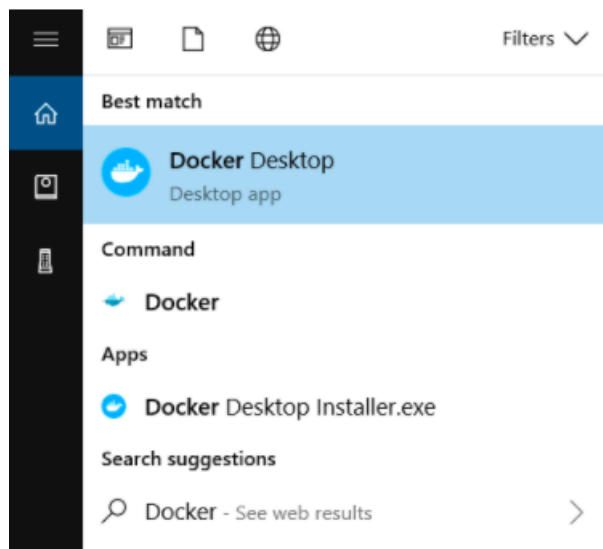


4. Une fois l'installation réussie, on clique sur **Fermer** pour terminer le processus d'installation.

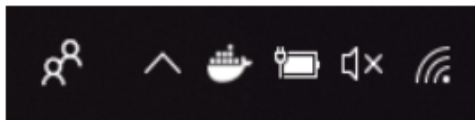


- **DÉMARRER DOCKER DESKTOP :**

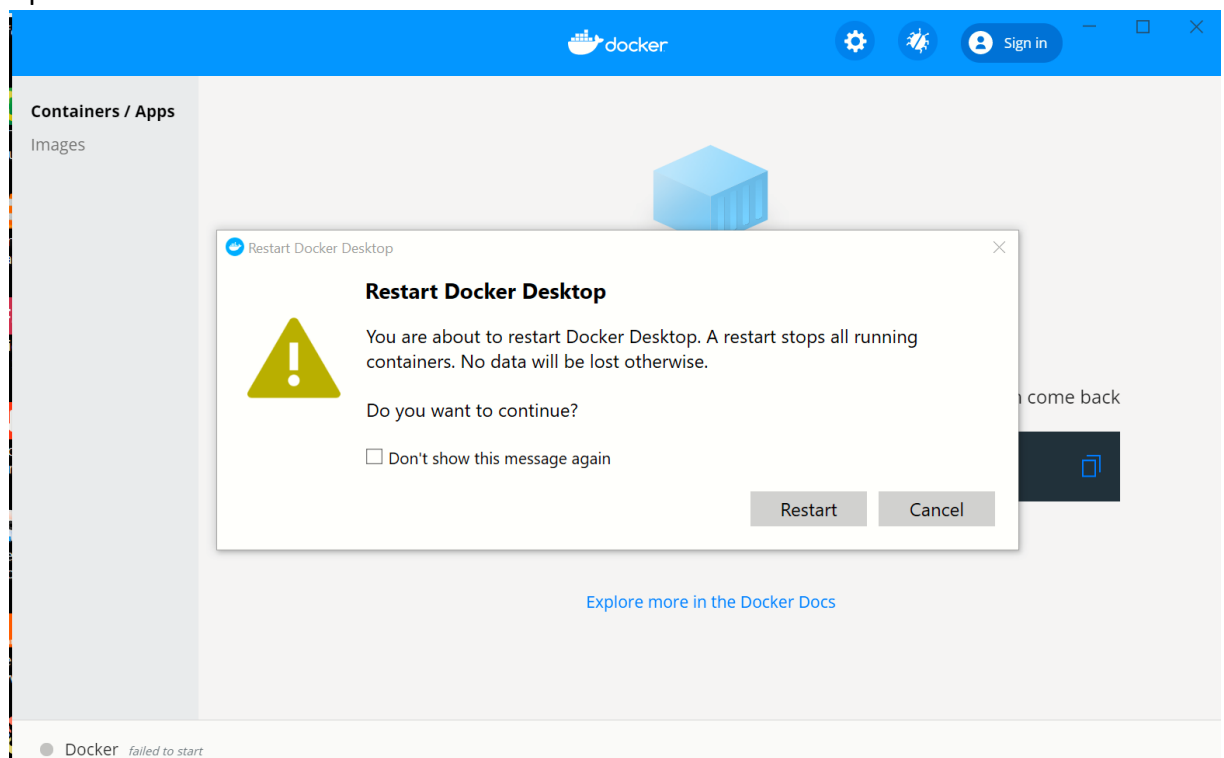
Docker Desktop ne démarre pas automatiquement après l'installation. Pour démarrer Docker Desktop, recherchez Docker et sélectionnez **Docker Desktop** dans les résultats de la recherche :



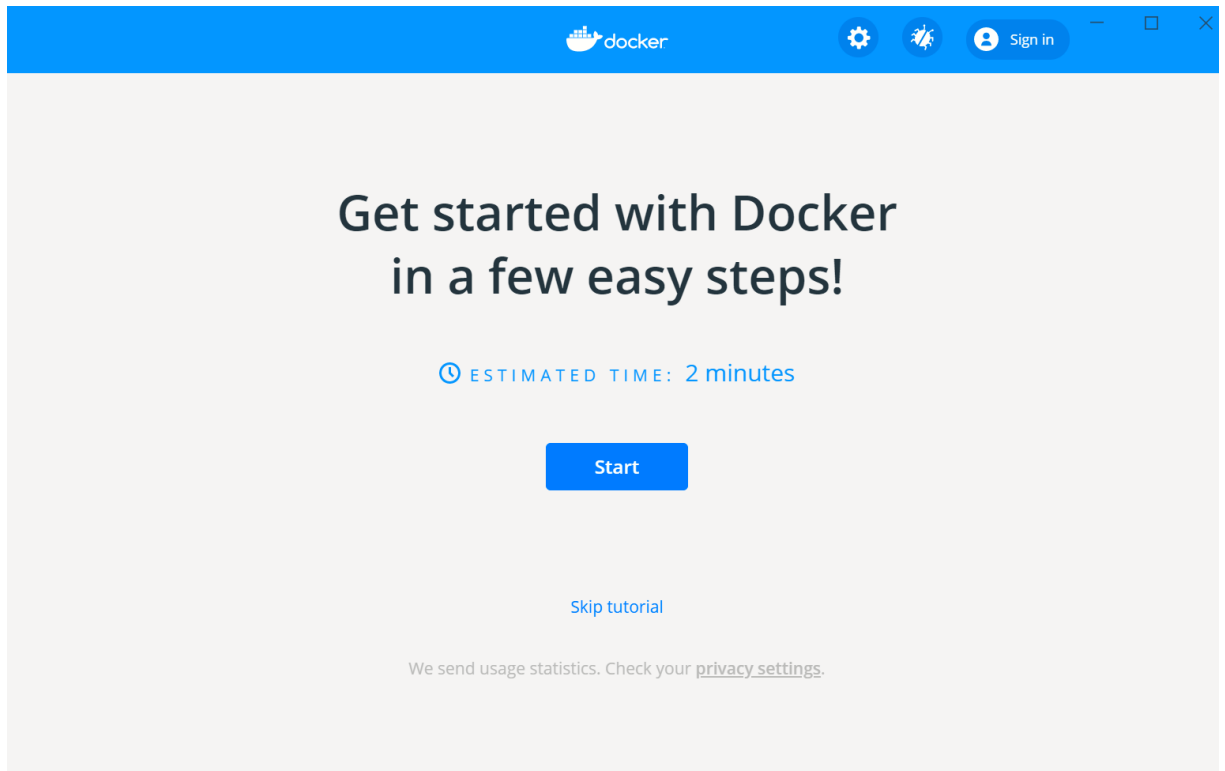
Lorsque l'icône de baleine dans la barre d'état reste fixe, Docker Desktop est opérationnel et est accessible depuis n'importe quelle fenêtre de terminal.



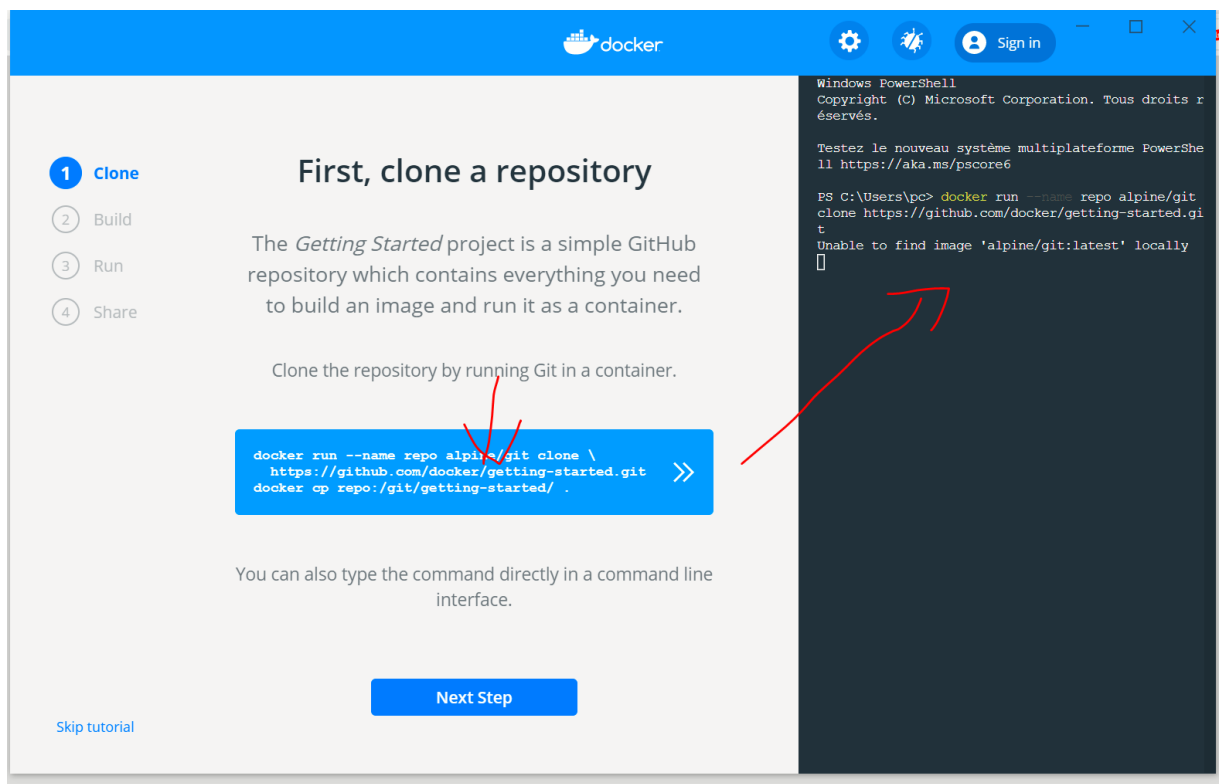
Après ouverture on a :



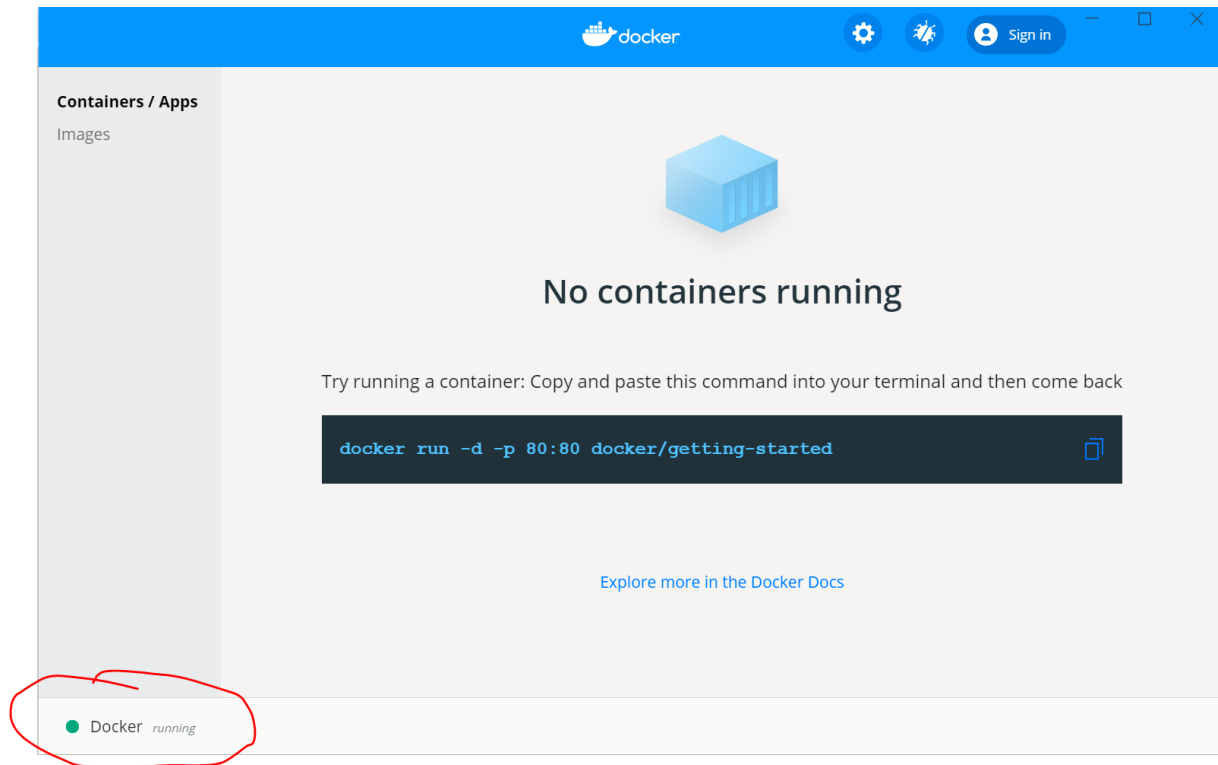
Une fois l'initialisation terminée, Docker Desktop lance le didacticiel d'intégration. Le didacticiel comprend un exercice simple pour créer un exemple d'image Docker, l'exécuter en tant que conteneur, pousser et enregistrer l'image dans Docker Hub.



On exécute maintenant avec succès Docker Desktop sous Windows.
On peut lancer la commande donnée :



Si l'exécution se termine on aura :



VI. CONCLUSION

En résumé nous pouvons dire que l'étude de ce projet nous a permis d'avoir de nouvelles bases c'est-à-dire faire des nouvelles expériences (comme travailler en équipe sur un projet avec des machines différentes) sur GitHub et Docker hub et d'acquérir plus de connaissances.

LIEN DU REPOSITORY : <https://github.com/bousso-cell/depotBousso.git>