

École Supérieure Polytechnique

Département Génie Informatique

Gestion du Stockage de Documents dans les Systèmes Basés sur la Blockchain

Groupe 1 : Évaluation du Stockage Off-Chain avec Hyperledger et IPFS

<u>Élèves</u>: <u>Enseignants</u>:

Koumou Jonathan ASSAMAGAN Ibrahima BALDE Saliou Samba DIAO Dieynaba SOW Gervais MENDY Attoumane TAHAR

Repository github

Année : 2023 - 2024

Objectif

Évaluation de la faisabilité et l'efficacité du stockage de documents off-chain en utilisant IPFS, en mettant l'accent sur la sécurité, la disponibilité et l'intégrité des données.

Étapes de Travail:

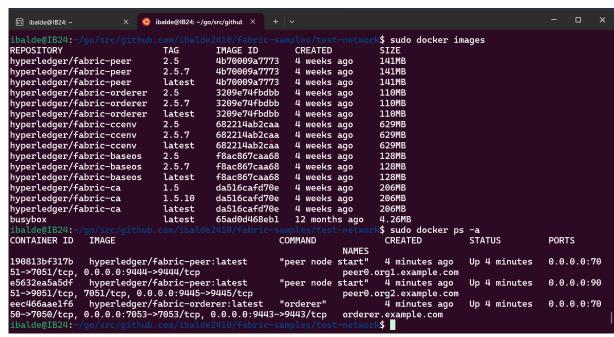
I. <u>Configuration du réseau Hyperledger en local</u>: Installation des dépendances, configuration des nœuds et initialisation du réseau sur des machines virtuelles ou des conteneurs locaux.

Hyperledger Fabric est une plateforme pour les solutions de registre distribué (blockchain) qui s'appuie sur une architecture modulaire offrant un haut degré de confidentialité, de résilience, de flexibilité et d'évolutivité. Elle est conçue pour prendre en charge des implémentations enfichables de différents composants et s'adapter à la complexité et aux subtilités de l'écosystème économique.

Installation

La première étape de notre travail a consisté à l'installation de Hyperledger Fabric en local. Cette dernière s'est faite globalement par le téléchargement des images des composants de Fabric avec docker, que sont :

- les **Peers**: un nœud du réseau blockchain qui stocke et réplique le registre distribué, pouvant exécuter des contrats intelligents (les **Smart Contracts**)
- les **Orderer** : un nœud responsable de la commande des transactions et de la création de blocs, assurant le consensus du réseau.
- les **ccenv** : un environnement d'exécution pour les contrats intelligents, fournissant l'isolation et la sécurité nécessaires.
- les **Baseos**: une image Docker de base pour les conteneurs ccenv, définissant le système d'exploitation et les dépendances logicielles.
- les CA: Autorité de certification qui émet des certificats numériques aux entités du réseau.



• Vérification de l'installation : Réseau de test

Pour vérifier l'installation de Hyperledger Fabric et prendre en main l'outil, nous avons téléchargé des données d'exemples pour mettre en place un réseau local (fourni par Fabric) - en installant des nœuds (Peer et Orderer sur nos machines locales).

```
Generating certificates using cryptogen tool
Creating Org1 Identities
+ cryptogen generate --config=./organizations/cryptoge
org1.example.com
  res=0
Creating Org2 Identities
  cryptogen generate --config=./organizations/cryptoge
org2.example.com
+ res=0
  cryptogen generate --config=./organizations/cryptoge
WARN[0000] /home/jonathan/go/src/github.com/kuro-jojo/WARN[0000] /home/jonathan/go/src/github.com/kuro-jojo/
 Network fabric_test
                                                               Created
 Volume "compose_orderer.example.com"
Volume "compose_peer0.org1.example.com"
Volume "compose_peer0.org2.example.com"
Container peer0.org1.example.com
Container peer0.org2.example.com
                                                              Created
                                                               Created
                                                               Started
    Container orderer.example.com
                                                               Started
```

Les composants de notre réseau de test sont **trois (3) noeuds -** chaque noeud dans un réseau Fabric doit se trouver dans une organisation (**org1, org2**) - :

- 2 nœuds Peer dans les organisations peer0.org1.example.com et peer0.org2.example.com.
- 1 noeud Orderer

La prochaine étape a été de mettre en place un "Channel" pour les transactions entre les organisations Org1 et Org2.

Ensuite, pour interagir avec ce channel, on a utilisé la commande **peer** qui permet d'appeler des **Smart Contract** - déjà déployé pour ce test. Et ainsi, nous avons pu effectuer et observer les différentes transactions effectuées au niveau du livre (ledger ou **Chaincode**).

```
[
{"ID": "asset1", "color": "blue", "size": 5, "owner": "Tomoko", "appraisedValue": 300},
{"ID": "asset2", "color": "red", "size": 5, "owner": "Brad", "appraisedValue": 400},
{"ID": "asset3", "color": "green", "size": 10, "owner": "Jin Soo", "appraisedValue": 500},
{"ID": "asset4", "color": "yellow", "size": 10, "owner": "Max", "appraisedValue": 600},
```

II. <u>Installation d'IPFS en local</u>: Déployer IPFS sur le même environnement local pour une intégration directe avec Hyperledger.

Configuration de l'infrastructure du serveur avec le cluster IPFS(InterPlanetary File System) qui est un ensemble de protocoles composables peer-to-peer pour l'adressage, le routage et le transfert de données adressées par contenu dans un système de fichiers décentralisé.

Avec IPFS, on peut stocker et récupérer des données et interagir avec le réseau via une simple application GUI, un terminal ou un navigateur standard.

Les fonctionnalités les plus importantes du cluster IPFS sont les suivantes :

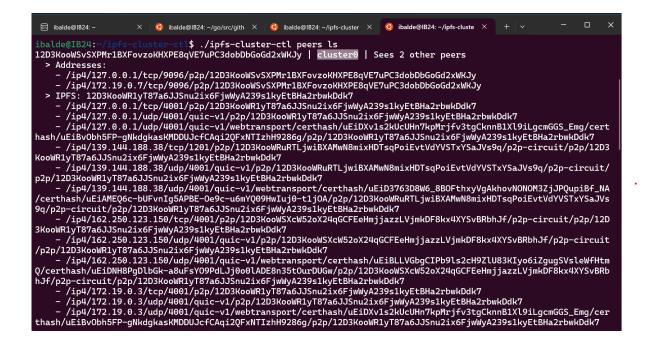
- Facile à exécuter : s'exécute indépendamment d'IPFS et de l'API du démon IPFS.
- Hiérarchisation intelligente des épingles : les nouvelles épingles sont prioritaires par rapport aux demandes d'épinglage anciennes ou dont l'épinglage a échoué à plusieurs reprises.
- Ingérer des broches à grande échelle : les broches peuvent être ajoutées à un rythme de plusieurs centaines de broches par seconde dans le cluster à partir du moment où elles sont suivies et gérées par les pairs du cluster.
- API et CLI: fournit un client de ligne de commande et une API REST HTTP de cluster complète. Aucun serveur central à gérer: les homologues du cluster forment un réseau distribué et maintiennent une liste de broches globale, répliquée et sans conflit.
- Ajout multi-pairs : ingère directement le contenu IPFS vers plusieurs démons.
- API proxy IPFS: les homologues du cluster fournissent une API proxy IPFS supplémentaire qui se comporte exactement comme le fait l'API du démon IPFS.
- Prêt pour l'intégration : les homologues du cluster peuvent être lancés et contrôlés par programme à l'aide de clients Go et Javascript pour son API.

Les étapes de la création d'un cluster local sont les suivantes :

- Il faut au préalable avoir docker et docker-compose installé
- On fait le téléchargement du dernier ipfs-cluster-ctl package depuis <u>dist.ipfs.tech</u> et on le décompresse.
- On télécharge le fichier docker-compose.yml dans le repertoire ipfs-cluster-ctl

Démarrage du cluster avec docker-compose

- Après cela, on ouvre une nouvelle fenêtre de terminal et à partir de cet instant, on peut désormais interagir avec notre cluster dans le répertoire ipfs-cluster-ctl.
- On arrive à répertorier les pairs au sein du cluster



On peut ajouter des fichiers dans le cluster et consulter l'état des fichiers sur le cluster de nœuds IPFS