Introducción a Hyperledger Fabric

Fundamentos de blockchains

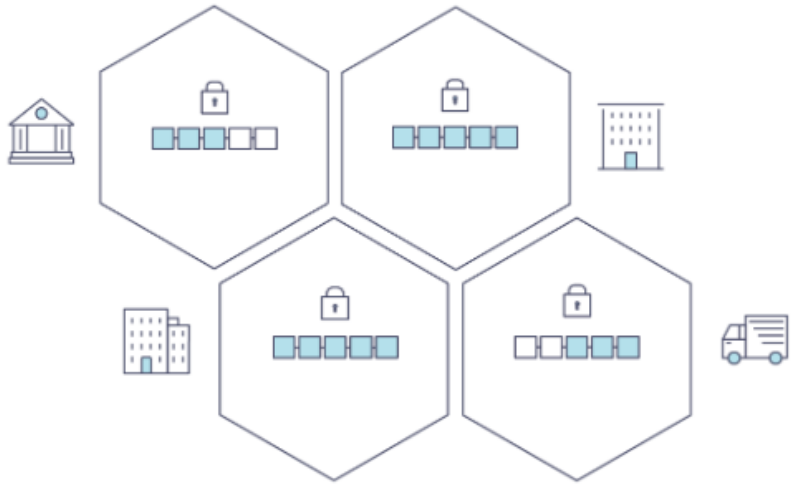
René Dávila - Jorge Solano

Índice

Blockchain

En términos generales, una **blockchain** se podría considerar como un **libro mayor** en el área contable **(ledger)**. Es un lugar donde se llevan a cabo transacciones inmutables dentro de una red de nodos distribuida.

Cada uno de estos **nodos** mantiene una **copia del libro mayor** aplicando transacciones que han sido validadas mediante el protocolo de consenso, agrupadas en bloques que incluyen un hash que permite enlazar cada bloque con su bloque anterior.

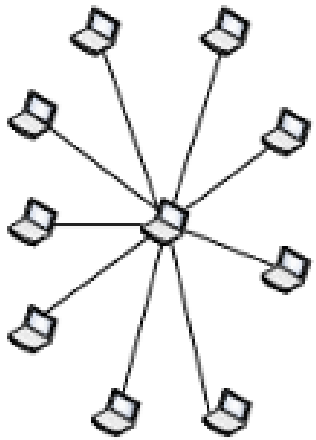


Criptomonedas

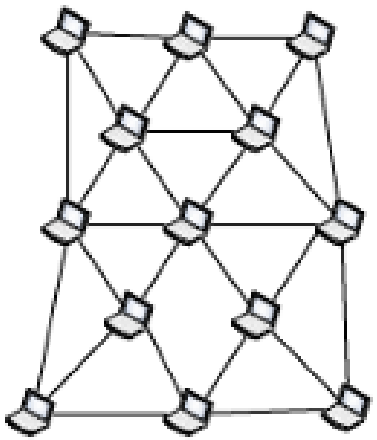
La aplicación más reconocida de blockchain es la criptomoneda **Bitcoin**.

Ethereum es una criptomoneda alternativa, la cual integra muchas características de Bitcoin, pero agrega contratos inteligentes (smart contracts) con el fin de crear una plataforma para aplicaciones distribuidas.

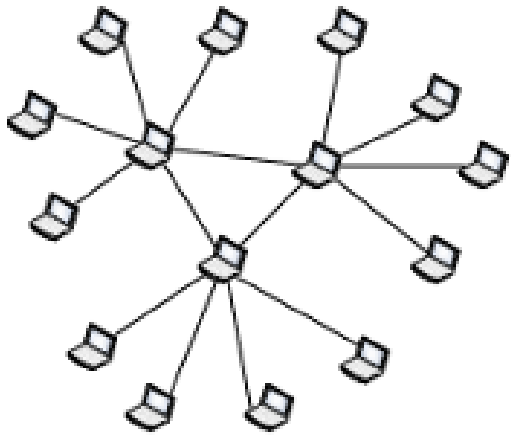
Tanto **Bitcoin** como **Ethereum** caen en la clasificación de blockchain, las cuales clasificaremos como tecnologías de **blockchain públicas y sin permiso**. Básicamente, son redes públicas, abiertas a cualquiera, donde los participantes interactúan de manera anónima.



Public Blockchain



Consortium Blockchain



Private Blockchain

En los casos de **uso empresarial** la **identidad** de los participantes es un **requisito indispensable**, como por ejemplo en una transacción financiera, donde las regulaciones el Know Your Customer (KYC) y el Anti-Money Laundering (AML) deben seguirse de manera puntual.

Para el uso empresarial de blockchain se deben considerar los siguientes requerimientos:

- Cada participante debe identificarse y ser identificable.
- Las redes deben estar autorizadas.
- Alto rendimiento durante las transacciones.
- Baja latencia en la confirmación de transacción.
- Privacidad y confidencialidad de las transacciones y de la información de las transacciones.

Smart Contracts

Una red blockchain utiliza **Smart Contracts** para proporcionar acceso controlado al libro mayor.

No solo son un mecanismo clave para encapsular la información y mantenerla simple en toda la red, sino que también se pueden escribir para permitir a los participantes ejecutar ciertos aspectos de las transacciones automáticamente.

Smart Contracts

Hyperledger project community

La Linux Foundation se caracteriza por nutrir proyectos **open source** de **open governance** que genera comunidades sostenibles fuertes y ecosistemas prósperos.

Linux Foundation creó el projecto **Hyperledger** en 2015 para avanzar en la industria de la tecnología de blockchain.

Hyperledger project community

La comunidad ha desarrollado los siguientes **Distributed Ledgers**:

- **Hyperledger Besu**.
- **Hyperledger Burrow**.
- **Hyperledger Indy**.
- **Hyperledger Iroha**.
- **Hyperledger Sawtooth**.
- **Hyperledger Fabric**.

Hyperledger project community

Hyperledger Besu, cliente *Ethereum* basado en Java, alternativa para crear casos de uso de redes públicas y autorizadas.

- Usa tecnología Apache.
- Usa redes de prueba Rinkeby, Ropsten y Görli.
- Usa algoritmos de consenso PoW, PoA, IBFT.

Hyperledger project community

Hyperledger Burrow, distribución completa de blockchain binario único centrado en la simplicidad, la velocidad y la ergonomía del desarrollador.

- Admite Smart Contracts basados en EVM y WASM.
- Usa consenso BFT mediante el algoritmo Tendermint.

Hyperledger project community

Hyperledger Indy, proporciona herramientas, bibliotecas y componentes reutilizables para proporcionar identidades digitales arraigadas en blockchains u otros ledgers distribuidos.

- Tiene como objetivo interoperabilidad entre dominios administrativos o entre aplicaciones.
- Interoperable con otras blockchain o de uso independiente para descentralización.

Hyperledger project community

Hyperledger Iroha, está diseñado para ser simple y fácil de incorporar en proyectos de infraestructura o IoT que requieren tecnología de contabilidad distribuida.

- Construcción y diseño en C++ modular.
- Usa un algoritmo tolerante a fallas nuevo llamado YAC.

Hyperledger project community

Hyperledger Sawtooth, ofrece una arquitectura flexible y modular que separa el sistema central del dominio de la aplicación.

- Los contratos inteligentes pueden especificar las reglas de negocio para las aplicaciones.
- Admite diversos algoritmos de consenso como PBFT o PoET.

Hyperledger project community

Hyperledger Fabric es uno de los **proyectos de blockchain en Hyperledger**, el cual es un ledger que utiliza smart contracts y donde los participantes manejan las transacciones.

Índice

Hyperledger Fabric v2.0

Hyperledger Fabric es una plataforma **open source** de grado **empresarial**, que permite manejar un tecnología distribuida de ledger (una blockchain).

Tiene algunas **capacidades claves** diferentes con respecto a otros ledger distribuidos o a otras plataformas de blockchain.

Índice

Private and permissioned

Hyperledger Fabric destaca por sobre otros sistemas de blockchain porque **es privada y autorizada**, esto es, los miembros de la red se registran a través de un Membership Service Provider (MSP) de confianza.

Entonces, por ejemplo, si los participantes no pueden confiar completamente uno en el otro (por ejemplo, si son competidores en la misma industria), la red puede operar bajo el modelo de **gobierno basado en la confianza** que existe entre los participantes, como lo es un acuerdo legal o un marco de referencia para manejar disputas.

Pluggable

Hyperledger Fabric ofrece varias **opciones enchufables**. La información del ledger puede guardarse en múltiples formatos, los mecanismos de consenso se pueden intercambiar y se pueden tener diferentes Provedores de Servicio de Membresía (MSP).

Esto permite que la plataforma sea **personalizada** y se ajuste a casos de uso y modelos confiables particulares.

Por ejemplo, dentro de una empresa o en una empresa operada por una autoridad de confianza, el protocolo de consenso tolerante a fallas bizantinas podría considerarse innecesario y hasta excesivo, en su lugar se podría pensar que el **protocolo de consenso tolerante a fallas** sería más apropiado.

Channels

Hyperledger Fabric ofrece la posibilidad de crear **canales**, lo que permite que un grupo de participantes tenga su propio ledger de transacciones.

Esta capacidad es importante en redes donde algunos participantes pueden ser competidores y no quieren que cada transacción (pe, una oferta especial en un producto) sea conocida por cada participante.

Shared Ledger

El ledger de **Hyperledger Fabric** comprende dos componentes: el **world state** y la **transaction log**. Cada participante tiene una copia del ledger de cada red a la que pertenece.

El world state representa la base de datos del ledger. La transaction log guarda la historia actualizada del world state. El ledger es, entonces, la combinación de la **BD** del world state y la historia de la **transaction log**.

Smart contracts

Los **smart contracts** de **Hyperledger Fabric** están escritos en chaincode y son invocados por una aplicación externa a la blockchain. En la mayoría de los casos, el chaincode interactúa directamente con la base de datos del ledger (world state) y no con la log transaction.

El **chaincode** puede ser implementado **en lenguajes de programación de uso general** (como Java, Go y Node.js), por lo que no se requiere aprender un lenguaje de dominio específico.

Privacy

Hyperledger Fabric permite crear tanto redes donde la **privacidad** (utilizando canales) es un requerimiento operacional clave, así como redes abiertas.

Consensus

Hyperledger Fabric está diseñado para permitir que las redes elijan el **mecanismo de consenso** que mejor se acomode a la relación que existe entre los participantes.

Fabric puede aprovechar los protocolos de consenso que **no requieren criptomonedas** nativas para incentivar la minería o la ejecución de contratos inteligentes. Evitar las criptomonedas **reduce** de manera significativa **el riesgo o los vectores de ataque**.

Además, la ausencia de las operaciones de minería criptográfica permite que la plataforma se pueda desplegar con prácticamente el mismo costo operacional que cualquier otro sistema distribuido.

Modelo de Hyperledger Fabric

- **Assets**.- las definiciones de activos permiten el intercambio de casi cualquier cosa con valor monetario a través de la red, desde alimentos enteros hasta autos antiguos y futuros de divisas.
- **Chaincode**.- la ejecución de Chaincode se divide del orden de las transacciones, lo que limita los niveles requeridos de confianza y verificación en todos los tipos de nodos y optimiza la escalabilidad y el rendimiento de la red.
- **Ledger Features**.- el ledger inmutable y compartido codifica todo el historial de transacciones para cada canal e incluye la capacidad de consulta similar a SQL para una auditoría eficiente y resolución de disputas.

Modelo de Hyperledger Fabric

- **Privacy**.- los canales y la recopilación de datos privados permiten transacciones multilaterales privadas y confidenciales que generalmente requieren las empresas competidoras y las industrias reguladas que intercambian activos en una red común.
- **Security & Membership Services**.- la membresía autorizada proporciona una red blockchain confiable, donde los participantes saben que todas las transacciones pueden ser detectadas y rastreadas por reguladores y auditores autorizados.
- **Consensus**.- un enfoque único para el consenso permite la flexibilidad y escalabilidad necesarias para la empresa.

Índice

Para poder ejecutar los proyectos de **Hyperledger Fabric** es necesario tener ciertas herramientas ya instaladas. Entonces, primero hay que verificar o instalar los **prerrequisitos de instalación**.

Índice

Prerrequisitos de instalación

Para poder desarrollar aplicaciones con Hyperledger Fabric se deben tener instaladas las siguientes herramientas:

- Git
- curl
- Docker y Docker Compose (versión 17.06.2-ce o superior)
- Go
- Node.js y NPM
- Python

https://hyperledger-fabric.readthedocs.io/en/release-2.0/prereqs.html

Git

Git es un **controlador de versiones distribuido**, gratuito y de código abierto diseñado para manerjar proyectos de diferente tamaño con eficiencia.

`https://git-scm.com/downloads`

curl

curl (Client URL) es una biblioteca y una herramienta de línea de comandos que permite **transferir datos** a través de diferentes protocolos.

`https://curl.haxx.se/download.html`

Docker

Docker es un **contenedor de aplicaciones**. Un contenedor es una unidad estándar de software que empaqueta código y todas sus dependencias para que la aplicación se ejecute de manera confiable y rápida en cualquier entorno de computadora.

`https://www.docker.com/get-docker`

Docker Compose

Compose es una herramienta para definir y **ejecutar múltiples contenedores de aplicaciones** Docker. Docker-compose ya está incluida en Docker Desktop.

`https://docs.docker.com/compose/install/`

Go

Go es un **lenguaje de programación** de código abierto que permite crear software simple, confiable y eficiente.

`https://golang.org/dl/`

Node.js

Node.js es un **entorno de ejecución de JavaScript**, de código abierto y multiplataforma, que permite ejecutar código JavaScript fuera de un navegador web.

`https://nodejs.org/en/download/`

NPM

NPM es un **administrador de paquetes** para el lenguaje de programación JavaScript. Es el administrador de paquetes por defecto de Node.js.

`https://www.npmjs.com/get-npm`

Python

Python lenguaje de programación interpretado, de alto nivel y de propósito general.

`https://www.python.org/downloads/`

Índice

Instalación

Una vez **instalados los prerrequisitos**, se tiene el ambiente necesario para **descargar e instalar Hyperledger Fabric**. Actualmente el proyecto no cuenta con un instalador de los binarios de Fabric, pero los ejemplos que se descargan poseen scripts de ejecución.

https://hyperledger-fabric.readthedocs.io/en/release-2.0/install.html

Instalación

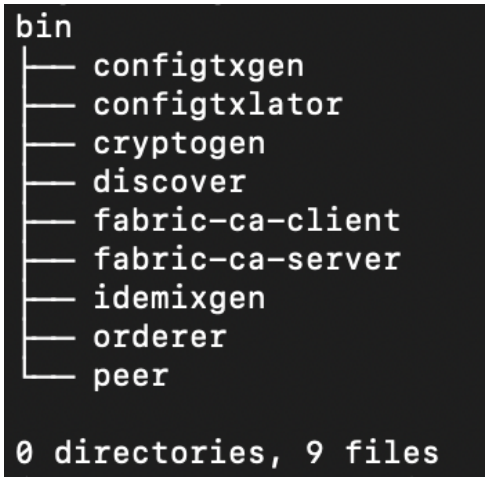
Para la instalación simplemente hay que descargar los binarios y las imágenes de Fabric

```
curl -sSL https://bit.ly/2ysbOFE | bash -s
```

https://raw.githubusercontent.com/hyperledger/fabric/master/scripts/bootstrap.sh

Instalación

La instrucción anterior descarga y descomprime los binarios requeridos y específicos para la plataforma y los guarda en el repositorio (**fabric-samples**) dentro de la **carpeta bin**.



Instalación

Por último, hay que mapear la carpeta **bin al PATH** del sistema. Aquí es importante comentar que también go y go/bin deben estar en el PATH.

```
export GOPATH=/usr/local/go
export PATH=$PATH:$GOPATH/bin:/Users/jrg_sln/Fabric/fabric-samples/bin
```

Índice

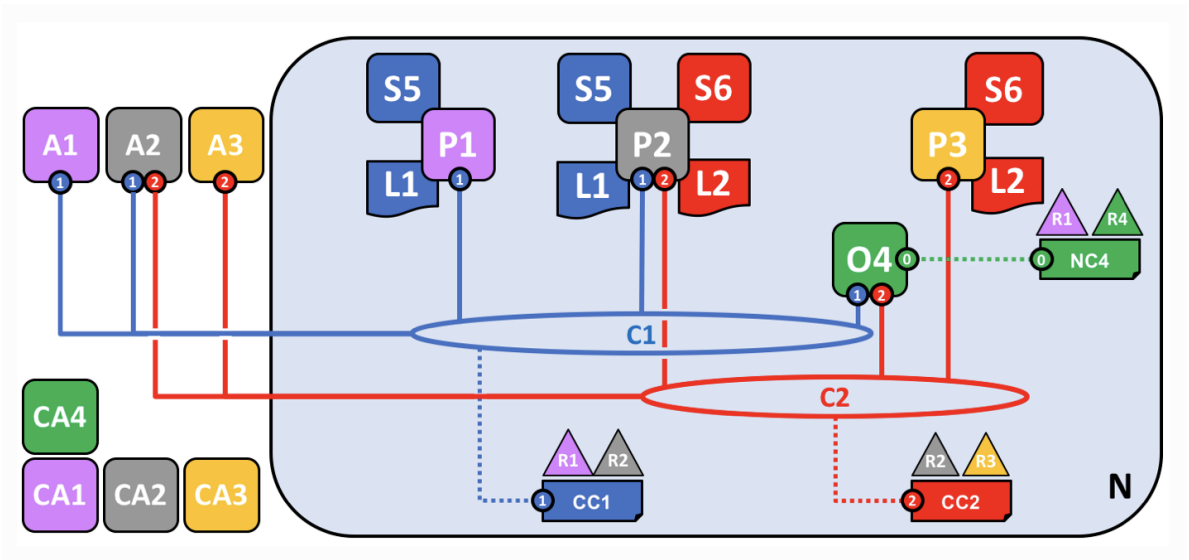
Para iniciar con la implementación de una **blockchain empresarial** (particular) se requiere tener un diagrama general de las entidades **participantes y su interacción**.

A continuación se describirán todos los **elementos** que pueden interactuar **en Hyperledger Fabric**, a partir de un ejemplo genérico.

Índice

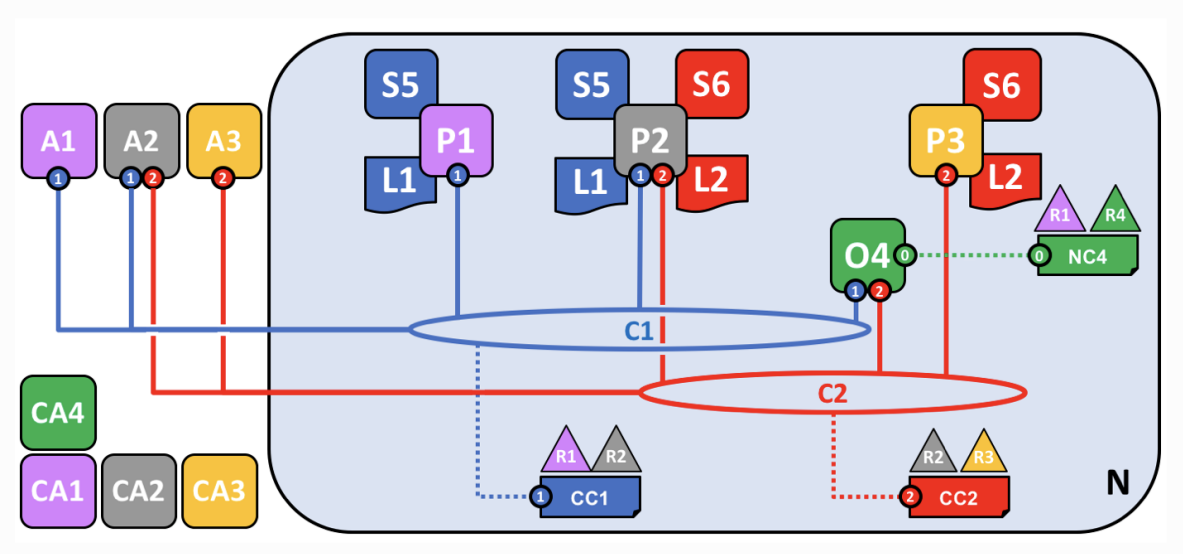
Contexto

Se tienen 4 organizaciones (R1, R2, R3 y R4). R4 es el iniciador de la red (no hará transacciones de negocios). R1 y R2 tienen una comunicación privada en la red. De la misma manera R2 y R3.



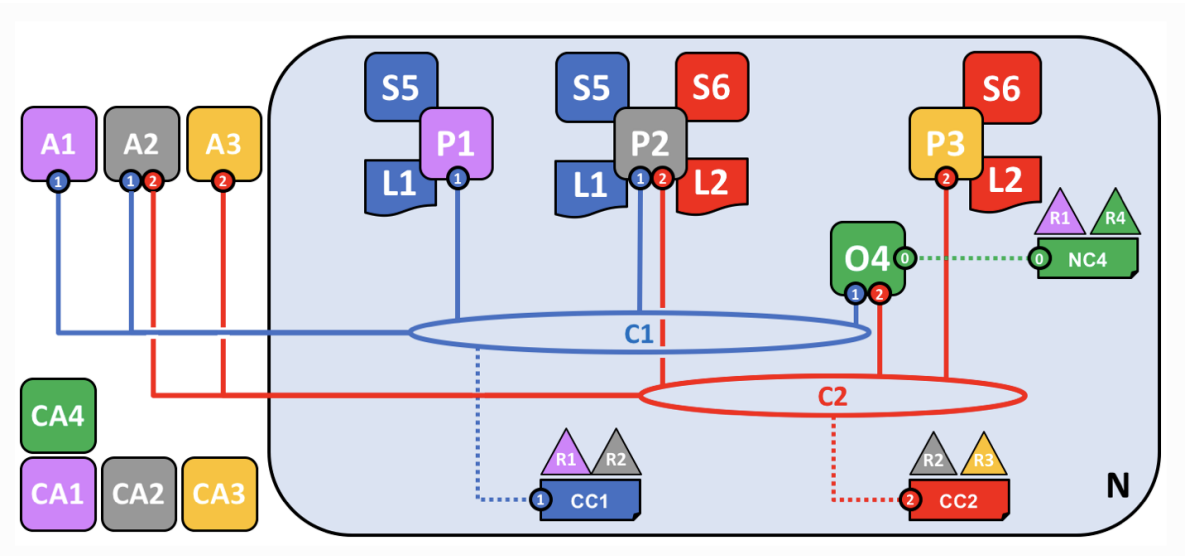
Contexto

R1 tiene una aplicación cliente A1 que hace transacciones en C1. R2 tiene una A2 que hace transacciones en C1 y en C2. R3 tiene una A3 que hace transacciones en el C2.



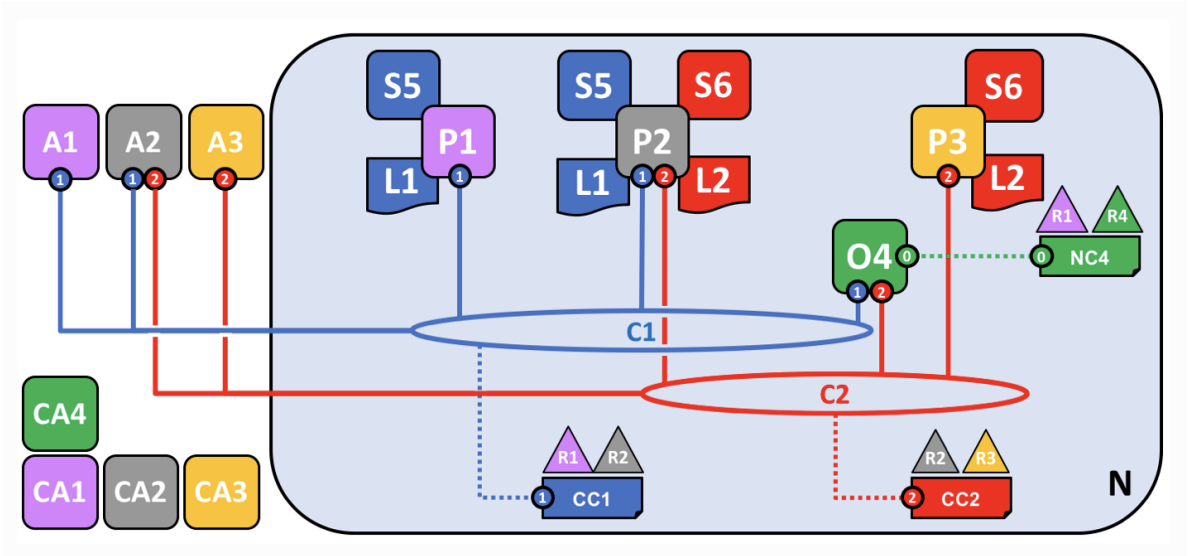
Contexto

El peer node P1 tiene una copia del ledger L1 asociado con C1 y un contrato inteligente S5. El peer node P2 tienen una copia de L1 y S5 asociado a C1 y una copia de L2 y S6 asociado a C2. El peer node P3 tiene una copia de L2 y S6 asociado a C2.



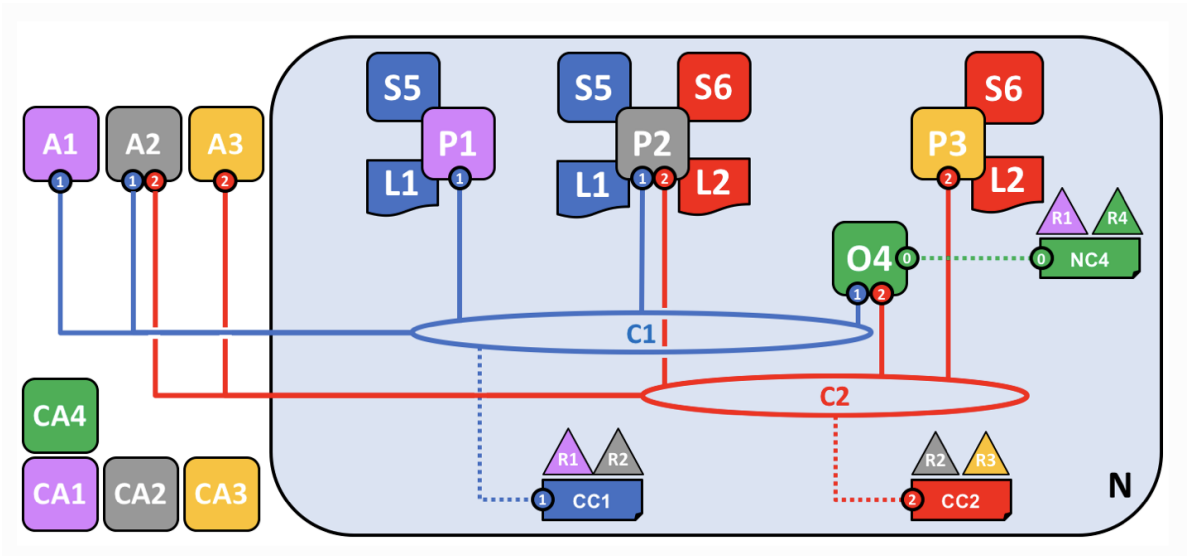
Contexto

La red está gobernada de acuerdo a las políticas especificadas en la configuración de red NC4. Las políticas están bajo el control de R1 y R4.



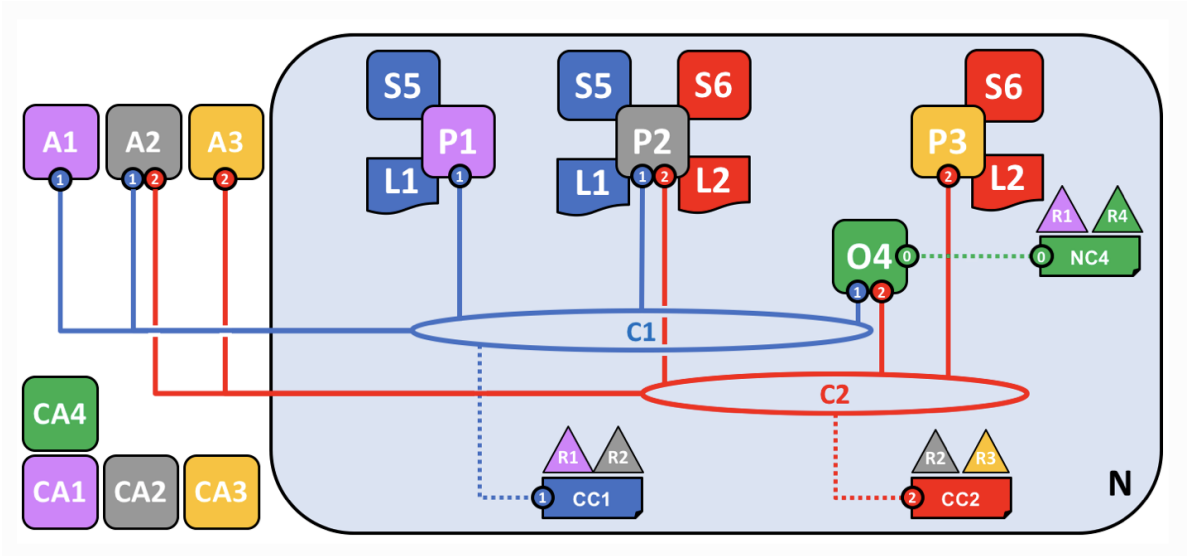
Contexto

C1 está gobernado de acuerdo a las políticas especificadas en la configuración del canal CC1; el canal está gobernado por R1 y R2. C2 está gobernado por las políticas especificadas en la CC2; el canal está gobernado por R2 y R3.



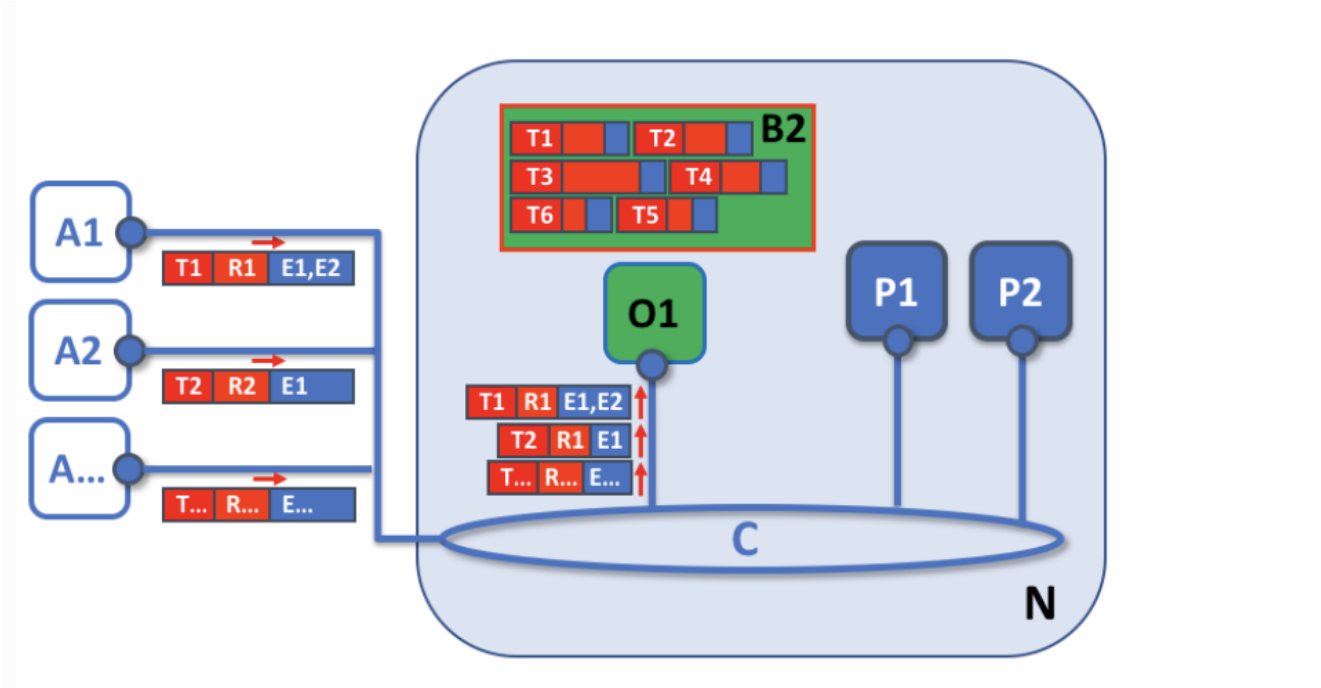
Contexto

Hay un nodo de ordering service O4 que funge como un punto de administración de la red N. O4 soporta los canales de aplicación C1 y C2, con el fin de ordenar los bloques de transacciones para su distrubución.



Contexto

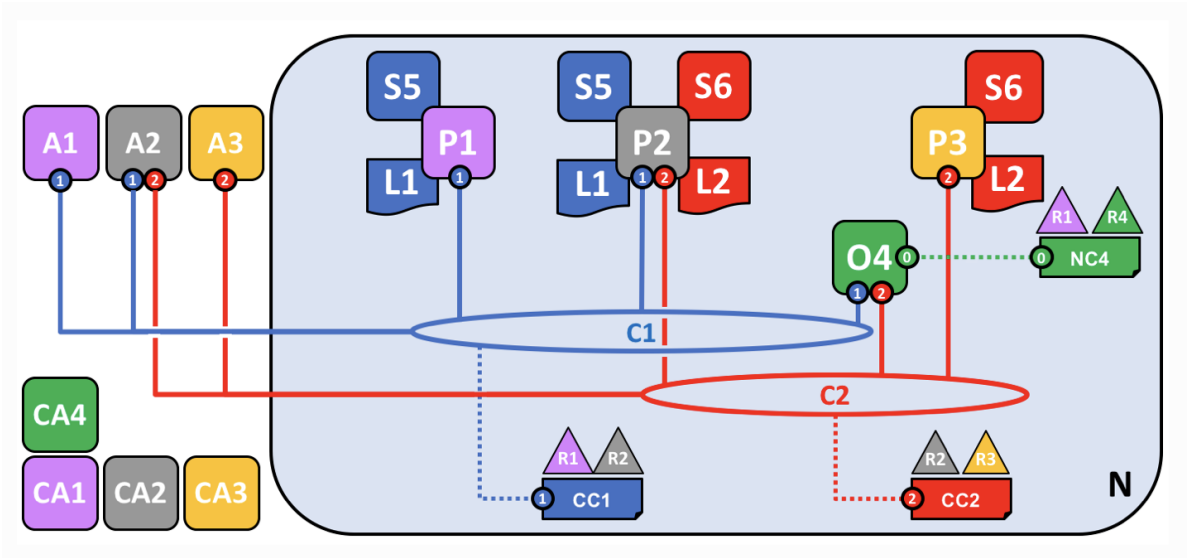
Ordering sevice



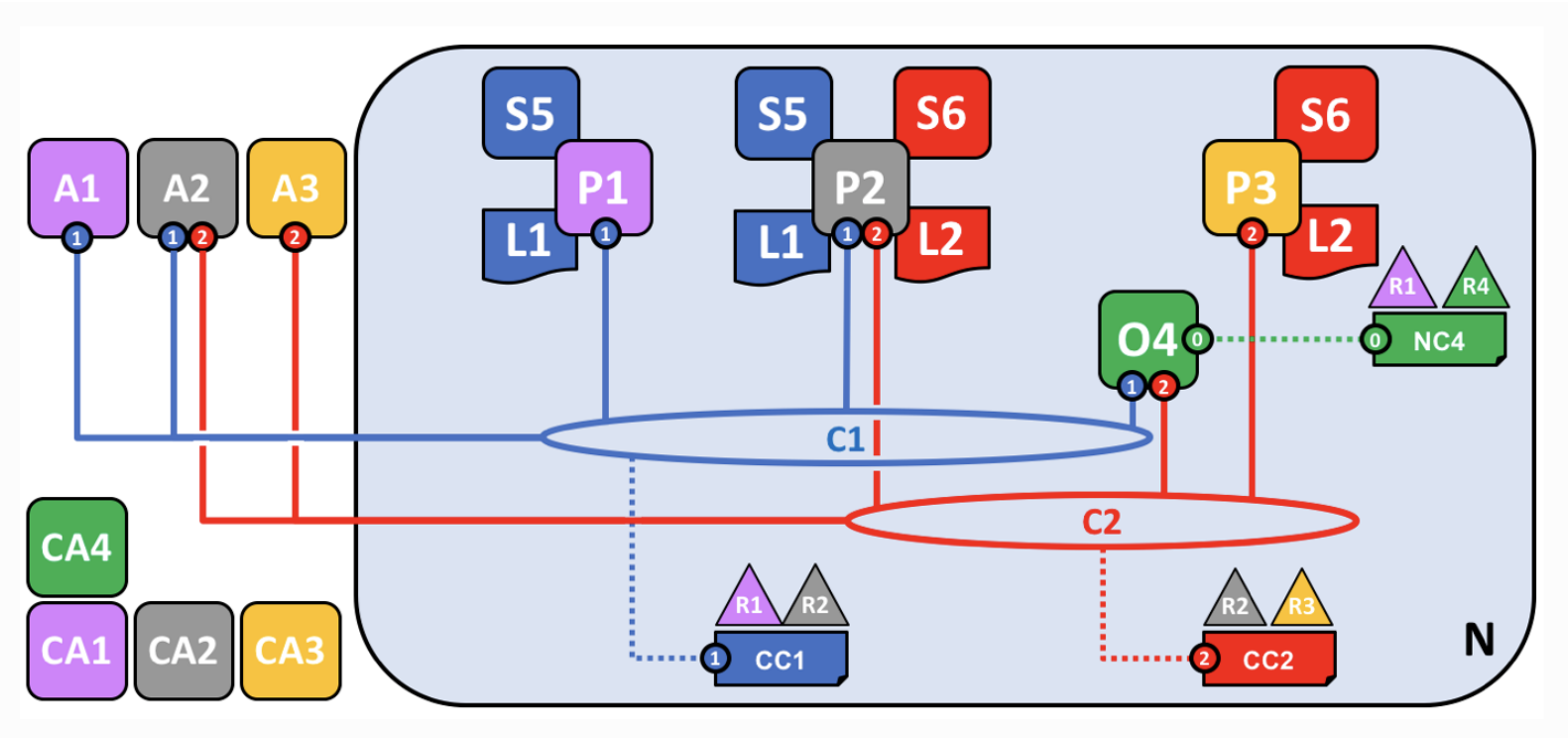
https://hyperledger-fabric.readthedocs.io/en/release-2.0/orderer/ordering_service.html

Contexto

Cada una de las organizaciones tiene su propia Autoridad Certificadora CA.



Contexto



https://hyperledger-fabric.readthedocs.io/en/release-2.0/network/network.html

Índice

Ejecución

Una vez descargado Hyperledger Fabric ya es posible interactuar con alguno de los proyectos.

En realidad, se descargaron imágenes y ejemplos Docker, esto es, contenedores con aplicaciones precargadas, las cuales se guardaron en la carpeta **fabric-samples**.

Ejecución

Vamos a trabajar con el proyecto test-network que está dentro de la carpeta fabric-samples (cd fabric-samples/test-network).

En esa carpeta se encuentra el script *network.sh*, el cual levanta una red Fabric utilizando las imágenes Docker descargadas. Para ver el texto de ayuda del script se puede ejecutar *network.sh -h*

```
Usage:
  network.sh <Mode> [Flags]
    <Mode>
      - 'up' - bring up fabric orderer and peer nodes. No channel is created
      - 'up createChannel' - bring up fabric network with one channel
      - 'createChannel' - create and join a channel after the network is created
      - 'deployCC' - deploy the fabcar chaincode on the channel
      - 'down' - clear the network with docker-compose down
      - 'restart' - restart the network

    Flags:
    -ca <use CAs> -  create Certificate Authorities to generate the crypto material
    -c <channel name> - channel name to use (defaults to "mychannel")
    -s <dbtype> - the database backend to use: goleveldb (default) or couchdb
    -r <max retry> - CLI times out after certain number of attempts (defaults to 5)
    -d <delay> - delay duration in seconds (defaults to 3)
    -l <language> - the programming language of the chaincode to deploy: go (default), javascript, or java
    -v <version>  - chaincode version. Must be a round number, 1, 2, 3, etc
    -i <imagetag> - the tag to be used to launch the network (defaults to "latest")
    -verbose - verbose mode
  network.sh -h (print this message)

 Possible Mode and flags
  network.sh up -ca -c -r -d -s -i -verbose
  network.sh up createChannel -ca -c -r -d -s -i -verbose
  network.sh createChannel -c -r -d -verbose
  network.sh deployCC -l -v -r -d -verbose

 Taking all defaults:
        network.sh up

 Examples:
  network.sh up createChannel -ca -c mychannel -s couchdb -i 2.0.0
  network.sh createChannel -c channelName
  network.sh deployCC -l javascript
```

Crear la red

El siguiente comando levanta la red del proyecto **test-network**, la cual consiste en dos peer nodes y un nodo ordering service; todavía no se crea algún canal:

```
./network.sh up
```

Listar contenedores

En este momento se encuentran los contenedores Docker corriendo (los dos nodos y el nodo orden de servicio). El siguiente comando permite listar los contenedores en ejecución:

```
docker ps -a
```

Consorcio

test-network tiene un consorcio (grupo de organizaciones) con dos miembros Org1 y Org2, además de una organización que mantiene el orden de servicio (ordering service) en la red.

Los **peers nodes** son los componentes fundamentales en cualquier red Fabric. Ellos son los encargados de **almacenar** en el ledger de la **blockchain** y validar las transacciones antes de que se suban al ledger. Además, ellos **ejecutan los smart contracts** los cuales contienen la lógica del negocio.

Ordering service

Todas las redes Fabric deben incluir un nodo **ordering service**. Este nodo es el que decide el orden y las transacciones que se incluyen en un bloque.

Los **ordering nodes** hace uso de la configuración de la red (define las capacidades de la red) y los parámetros de configuración del canal. Por defecto, los **ordering nodes** utilizan el algoritmo de consenso **Raft (**`https://raft.github.io`**)**.

Canales

La red que se está ejecutando cuenta con 2 peer nodes, uno por cada organización (Org1 y Org2), y un ordering node. Ahora vamos a crear un **canal para las transacciones** entre el consorcio.

Recordemos que los canales son **enlaces privados de comunicación** entre miembros específicos de la red y cada canal tiene su propio ledger de la blockchain.

Crear canal

Para **crear un canal** entre las organizaciones Org1 y Org2 y unir los peer nodes al canal se debe ejecutar el siguiente comando:

```
./network.sh createChannel
```

Smart contract en Fabric

Para asegurar que una transacción creada a partir de un **smart contract** sea válida, ésta debe estar firmada por múltiples organizaciones.

En Fabric los smart contracts se despliegan en paquetes referidos como **chaincode**. El chaincode se **instala** en los peer nodes de la organización y después se **despliega** en el canal donde se ocupará. Los miembros del canal deben estar de acuerdo con su definición.

Iniciar chaincode

Para **crear un chaincode** en el canal desplegado se debe ejecutar el siguiente comando:

```
./network.sh deployCC
```

deployCC instala el chaincode de **fabcar** en peer0.org1.example.com y en peer0.org2.example.com y luego lo despliega en el canal.

Crear una aplicación cliente

En este momento se puede usar una **aplicación CLI** para **interactuar con la red** (ejecutar smart contracts, actualizar canales, instalar y desplegar smart contracts).

Crear una aplicación cliente

Para crear a **CLI** se va a ocupar la aplicación peer que está en la carpeta **bin** de fabric-samples (ya en el PATH). También se requiere poner en el PATH la variable FABRIC_CFG_PATH para apuntar al archivo de configuración **core.yaml**:

```
export FABRIC_CFG_PATH=$PWD/../config/
```

Crear una aplicación cliente

Después, se deben establecer las **variables de entorno** que permitiran operar a la aplicación CLI en la Org1:

Variables de entorno para Org1
export CORE_PEER_TLS_ENABLED=true
export CORE_PEER_LOCALMSPID="Org1MSP"
export CORE_PEER_TLS_ROOTCERT_FILE=$PWD/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt
export CORE_PEER_MSPCONFIGPATH=$PWD/organizations/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp
export CORE_PEER_ADDRESS=localhost:7051

Interactuar con la red

Ahora, el nodo **CLI** puede **solicitar el ledger** del canal al que pertenece y hacer transacciones: **consultar todos los carros.**

```
peer chaincode query -C mychannel -n fabcar -c '{"Args":["queryAllCars"]}'
```

Interactuar con la red

El nodo **CLI** puede **solicitar el ledger** del canal al que pertenece y hacer transacciones: **cambiar el dueño de un auto.**

```
peer chaincode invoke -o localhost:7050 –ordererTLSHostnameOverride
orderer.example.com –tls true –cafile
$PWD/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem
-C mychannel -n fabcar –peerAddresses localhost:7051 –tlsRootCertFiles
$PWD/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt
–peerAddresses localhost:9051 –tlsRootCertFiles
$PWD/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt
-c '{"function":"changeCarOwner","Args":["CAR9","Dave"]}'
```

Interactuar con la red

Debido a que la política de la red requiere que Org1 y Org2 aprueben la transacción, el **chaincode invoca** a ambos **peers** utilizando un **certificado TLS**.

El **chaincode** se instala en los **peer nodes** de la organización y **después se despliega** en el canal donde se ocupará. Los miembros del canal deben estar de acuerdo con su definición.

Validar cambios

Ahora, se pueden **validar los cambios** en el ledger de la blockchain. En esta ocasión se va a utilizar el nodo **CLI de Org2**, para ello hay que exportar las variables de entorno de Org2:

Variables de entorno para Org2
export CORE_PEER_TLS_ENABLED=true
export CORE_PEER_LOCALMSPID="Org2MSP"
export CORE_PEER_TLS_ROOTCERT_FILE=$PWD/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt
export CORE_PEER_MSPCONFIGPATH=$PWD/organizations/peerOrganizations/org2.example.com/users/Admin@org2.example.com/msp
export CORE_PEER_ADDRESS=localhost:9051

Validar cambios

El nodo **CLI** puede **solicitar el ledger** del canal al que pertenece y hacer transacciones: **Consultar un carro**

```
peer chaincode query -C mychannel -n fabcar -c '{"Args":["queryAllCars"]}'
```

Dar de baja la red

Para **terminar** un proyecto se debe **dar de baja la red**. El comando **down** detiene y elimina los nodos, los contendores, las organizaciones, el material criptográfico, los chaincodes, los registros Docker, los canales y los volúmenes Docker previamente ejecutados.

./network.sh down

`https://hyperledger-fabric.readthedocs.io/en/release-2.0/test_network.html`

Índice

Logspout (monitordocker.sh)

Es una **bitácora** de cualquier evento en la red, esta herramienta **recolecta cualquier flujo de salida** de diferentes contenedores. El script está en commercial-paper/organization/digibank/configuration/cli/. Para **iniciar** el monitor:

```
./monitordocker.sh net_test
```

Para **detener** el monitor:

```
docker stop logspout
docker rm logspout
```

Índice

Referencias I

[1] Raft consensus algorithm website.
The raft consensus algorithm.
URL https://raft.github.io.
[Visitada el 03-04-2020].

[2] Steven J. Vaughan-Nichols for Linux and Open Source.
What is docker and why is it so darn popular?
URL https://www.zdnet.com/article/what-is-docker-and-why-is-it-so-darn-popular/.

[Visitada el 02-04-2020].

[3] Hyperledger.
Hyperledger fabric: A blockchain platform for the enterprise.
URL https://hyperledger-fabric.readthedocs.io/en/release-2.0/.
[Visitada el 29-02-2020].