



coding-bootcamps.com

INTRODUCCIÓN A HYPERLEDGER FABRIC
PARA EL ADMINISTRADOR



Coding
Bootcamps

Por Jordi Guirao de [Coding Bootcamps](https://coding-bootcamps.com)

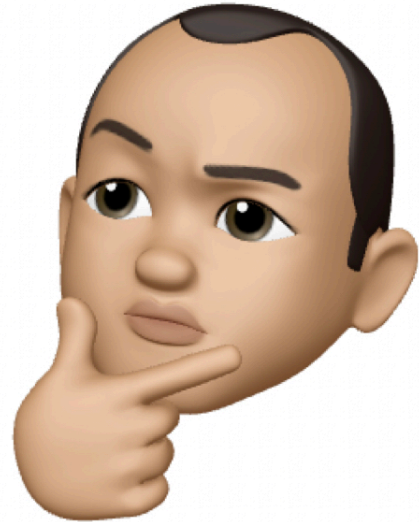
Componentes y arquitectura de Hyperledger Fabric

QUÉ HEMOS APRENDIDO HASTA AHORA

- Introducción al Blockchain
- Tipos de redes
- Componentes Blockchain
- Estructura Blockchain
- Algoritmos de consenso
- Participantes Blockchain
- Introducción a Hyperledger

DE QUE HABLAREMOS

- Arquitectura
- Componentes
- Instalación
- Ponte a prueba



ARQUITECTURA

- Peer
- Ordering service
- Fabric CA (Certificate Authority)
- Fabric ledger
- Channel/Canal
- Smart Contracts o chaincode
- Endorsement policy
- Membership services provider (MSP)

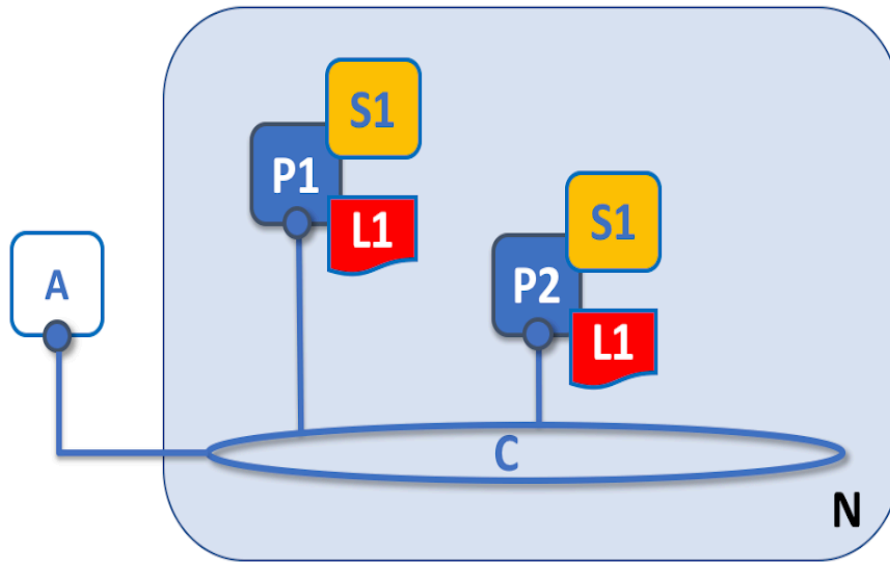
PEER


- Unidad fundamental de la red de Hyperledger Fabric.
- Los peers mantienen la ledger y el chaincode.
- Tipos de peer:
 - Anchor Peer
 - Leader Peer
 - Endorsing Peer
 - Committing Peer

PEER

- Los peers pueden ser creados, arrancados, parados, reconfigurados y eliminados.
- Proporcionan un conjunto de APIs que permite a los administradores y aplicaciones interactuar con los servicios que proporcionan.

PEERS & CANAL



N	Blockchain Network	L	Ledger
C	Channel	A	Application
P	Peer		Principal PA (e.g. A, P1) communicates via channel C.
S	Chaincode		

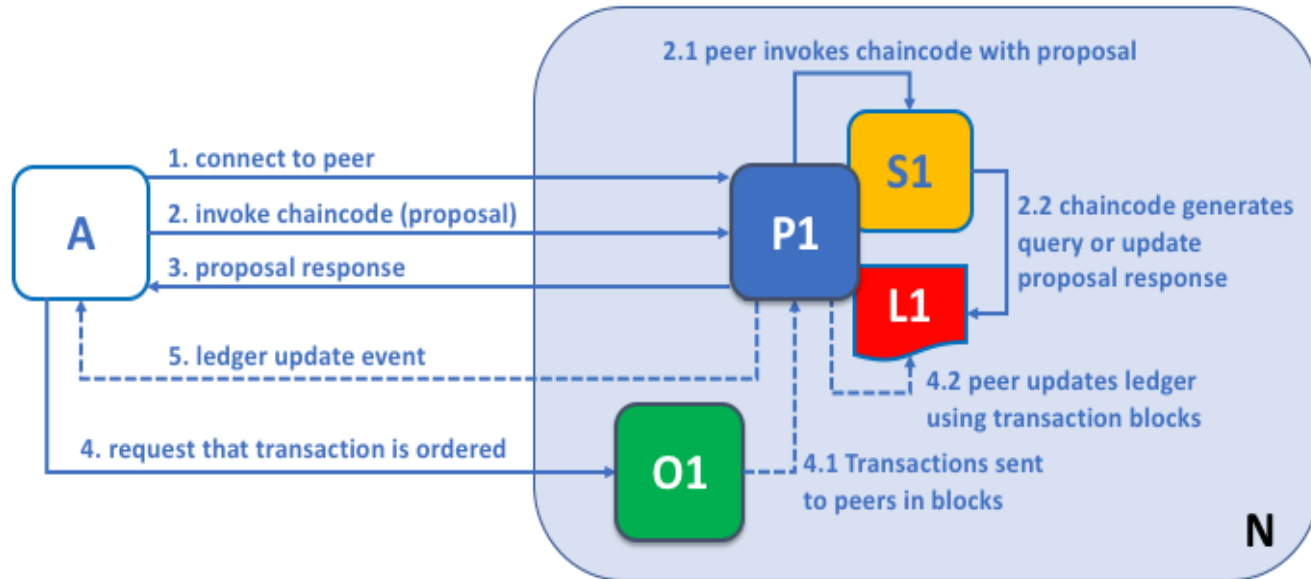
PEERS & CANAL

- El canal permite que un conjunto de peers y aplicaciones se comuniquen entre sí dentro de una red blockchain.
- Un canal es como una vía de comunicación entre aplicaciones y peers.

PEERS & APLICACIONES

- Las aplicaciones siempre necesitan conectarse a los peers cuando necesitan acceder a la ledger y chaincode.
- El Fabric Software Development Kit (SDK) hace esto fácil para los programadores. Esta API permite a las aplicaciones conectarse a los peers, invocar chaincode, generar transacciones a la ledger que se validarán y añadirán a la ledger y recibirán eventos cuando el proceso finalice.
- Una consulta de la ledger implica 3 pasos. Una actualización de la ledger implica 5 pasos.

PEERS & APLICACIONES

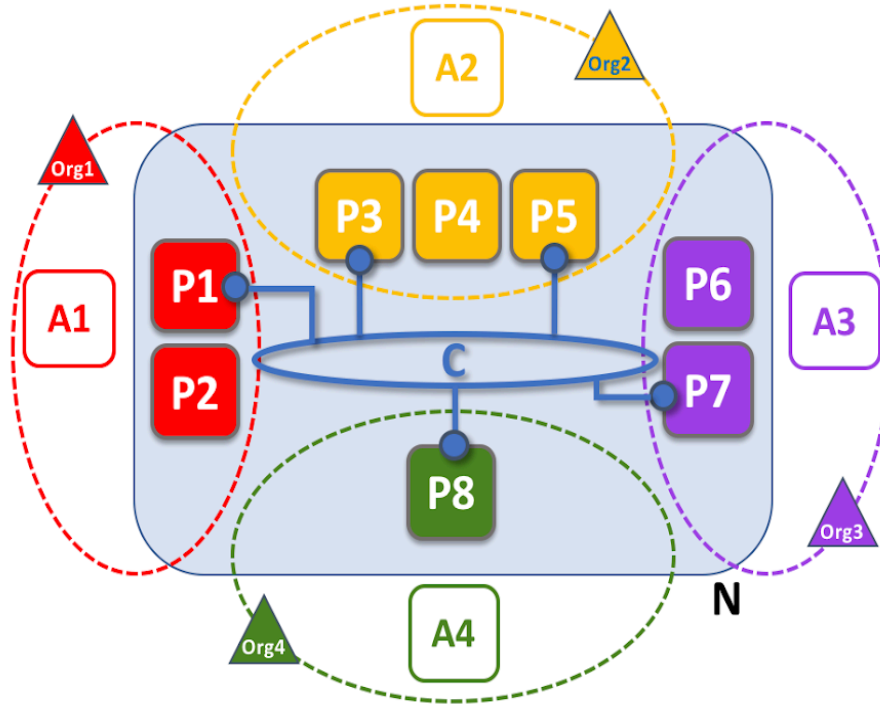









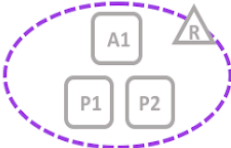
N	Blockchain Network
A	Application
P	Peer
S	Chaincode
L	Ledger
O	Orderer

PEERS & APLICACIONES

- Los peers junto con los orderers, aseguran que la ledger esté actualizada en cada peer.
- En la imagen anterior: La aplicación A conecta con P1 e invoca una función del chaincode S1 para consultar o actualizar la ledger L1. P1 invoca a S1 para generar una respuesta de la propuesta que contiene el resultado de la consulta o actualización de la ledger. La aplicación A recibe la respuesta. El proceso para consultar se complete.
- Para las actualizaciones, al crea una transacción a partir de las respuestas que envía a O1. O1 recopila las transacciones de toda la red de bloques y la distribuye a todos los peers. P1 valida la transacción antes de comprometerse con L1. Una vez actualizada L1, P1 genera un evento para A, indicando la finalización.

PEERS & ORGANIZACIONES



	Blockchain Network		Ledger
	Channel		Application
	Peer		Principal PA (e.g. A1, P5) communicates via channel C.
			Organization
		Organization R owns application A1 and peers P1, P2.	

PEERS & ORGANIZACIONES

- La red blockchain está formada por peers propiedad de diferentes organizaciones.
- En la imagen anterior, hemos visto que el canal C conecta P1, P3, P5, P7, P8 formando la red N. Los otros peers no se han unido al canal C, pero pueden estar unidos a otro canal.

PEERS & IDENTIDADES

- Los peers tienen un identidad asignada a través de un certificado digital de una autoridad de certificación.
- Cada peer de la red se le asigna un certificado digital por parte del administrador de la organización propietaria.

ORDERING SERVICE

- El ordering service también se conoce como Orderer.
- El mecanismo por el cual las aplicaciones y los peer interactúan entre sí para garantizar que la ledger en todos los pares sea consistente, es ejecutado por los nodos especiales llamados orderers.
- Tipos de orderer:
 - Solo (obsoleto en v2)
 - Kafka (obsoleto en v2)
 - Raft (recomendado)

ORDERING SERVICE

- Orderer transmite las transacciones a los pares para que se comprometan. Todos los peers reciben las mismas transacciones de bloque en el mismo orden.
- El ordering service ordena las transacciones por orden de llegada.
- Una vez las transacciones han sido ordenadas y comprometidas se añaden como parte del bloque para el canal.

ORDERING SERVICE

- Recordar que una transacción de actualización de la ledger es diferente a una transacción de consulta. Un solo peer no puede actualizar la ledger. Para actualizar la ledger se requiere el consentimiento de los otros peers de la red.
- Un peer requiere que otros peers de la red aprueben la actualización de la ledger antes de que esta se añada a la ledger. Este proceso se llama consenso y tarda más en completarse que una simple consulta.

ORDERING SERVICE

El proceso de actualización de la ledger consta de 3 fases:

- La primera fase, las aplicaciones funcionan con un subconjunto de endorsing peers (pares que la respaldan). Estos peers respaldan la actualización de la ledger pero no aplican la actualización.
- La segunda fase, las actualizaciones se compilan como transacciones y se empaquetan en bloques.
- La tercera fase, los bloques se distribuyen a todos los pares donde la transacción se valida antes de unirse a la ledger del peer.

ORDERING SERVICE

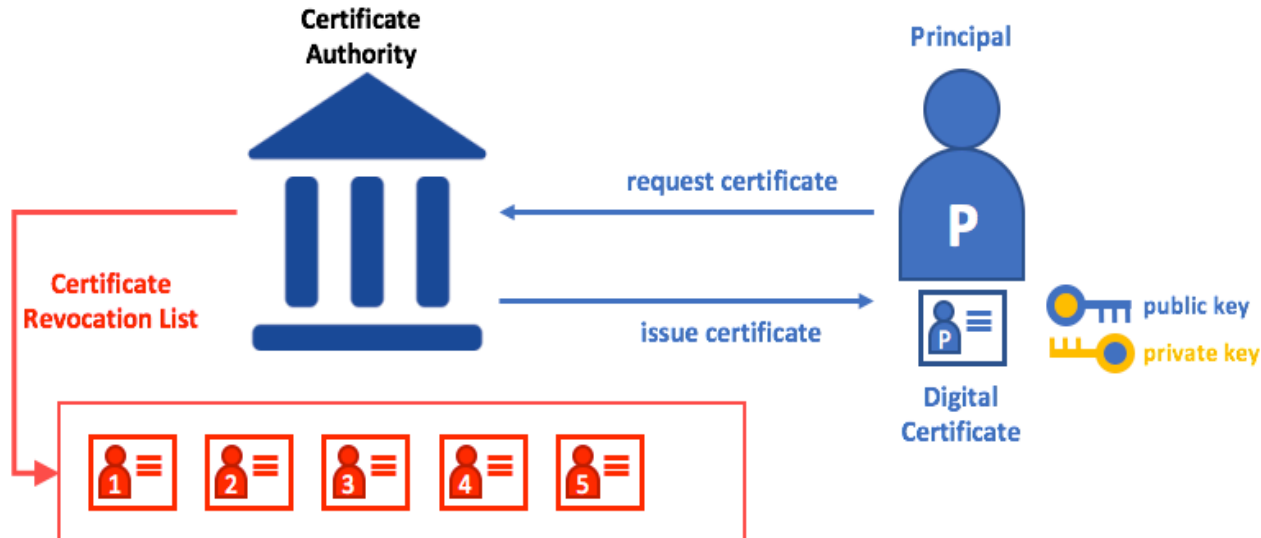
- Además de la función de organizador, el orderer también gestiona la lista de organizaciones a las que se les permite crear canales. Esta lista de organizaciones se conoce como consorcio. La lista se mantiene en la configuración del orderer system channel.
- El orderer también controla el acceso a los canales, restringiendo quien puede leer y escribir datos en canal y configurarlo.
- Quien está autorizado a modificar la configuración del canal está sujeto a las políticas que los administradores establecen cuando crean el consorcio o canal.

ORDERING SERVICE – RAFT VS KAFKA

- Raft es más fácil de configurar.
- Kafka y Zookeeper no están diseñados para ejecutarse en grandes redes.
- Raft es compatible de forma nativa.
- Kafka utiliza un grupo de servidores y el administrador de la organización especifica cuántos nodos quiere usar en un canal. Raft permite a los usuarios especificar qué nodos se implementan en cada canal.
- Raft es el primer paso para el desarrollo de un ordering service basado en Byzantine Fault Tolerant (BFT)

FABRIC CERTIFICATE AUTHORITY

- Una infraestructura de clave pública (PKI) es una tecnología que permite comunicaciones seguras en una red.



FABRIC CERTIFICATE AUTHORITY

- Un PKI está formada por un Certificate Authority que emiten certificados digitales a las diferentes partes. Estos certificados se utilizan para autenticarse en los mensajes que intercambian.
- La Certificate Revocation List (CRL) de una CA constituye una referencia para los certificados que ya no son válidos. Un certificado puede ser revocado porque sus claves han podido ser expuestas.
- Fabric CA es un proveedor de certificados de autoridad (CA) capaz de administrar identidades digitales de Fabric que son certificados de tipo X.509.

LEDGER

- La ledger almacena la información sobre objetos comerciales.
- La ledger consiste en dos partes: world state y blockchain.
- La blockchain es un registro de transacciones.
- World State tiene dos opciones:
 - LevelDB (predeterminado)
 - CouchDB

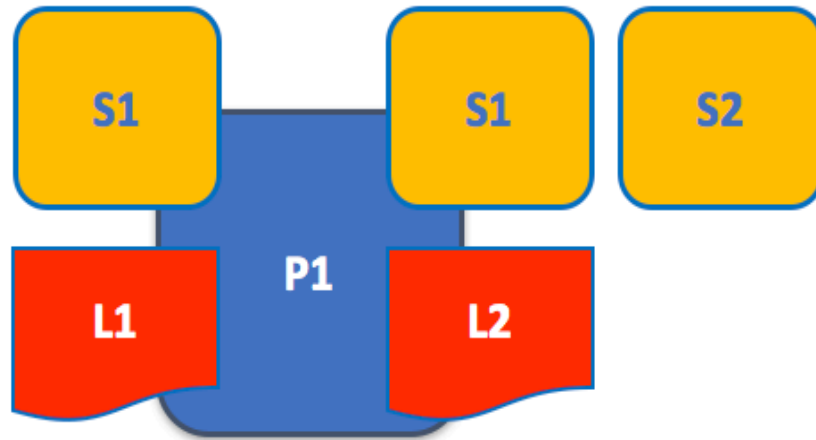
LEDGER

- La ledger es un registro secuenciado y resistente a manipulación de todas las “state” transacciones.
- Las “state” transacciones son el resultado de las invocaciones de chaincode.
- Cada transacción da como resultado un conjunto clave-valor que se almacena en la ledger.
- La ledger se compone de una cadena de bloques para almacenar el registro secuenciado e inmutable en bloques. La base de datos de estado mantiene el estado actual de Fabric.
- Cada canal tiene una ledger y cada peer mantiene una copia de la ledger para cada canal que es miembro.

CARACTERÍSTICAS DE LA LEDGER

- Las transacciones contienen firmas de todos los peers que respaldan y ejecutan transacciones al ordering service.
- Las transacciones se ordenan en bloques y se entregan a los peers del canal desde el ordering service.
- Los peers validan las transacciones según las políticas de respaldo (endorsement policies), haciendo cumplir las políticas.
- Existe inmutabilidad una vez la transacción es validada y añadida a la ledger.
- La ledger de un canal contiene un bloque de configuración que define políticas, listas de control, etc.

MULTI LEDGERS



CHANNEL / CANAL

- El canal es el mecanismo mediante el cual los peer dentro de una red blockchain se pueden comunicar y colaborar.
- Un canal de Hyperledger Fabric es una “subred” privada de comunicación entre dos o más miembros específicos de la red. Tienen el propósito de realizar transacciones privadas y confidenciales.

CHANNEL / CANAL

- Un canal lo definen los miembros (organizaciones), los peers de cada organización, la ledger, el chaincode y el nodo ordering service.
- Cada transacción se ejecuta en un canal, donde cada parte debe estar autorizada y autenticada para realizar transacciones.
- Cada peer que se une a un canal tiene su identidad proporcionada por el membership services provider (MSP) que autentifica a cada par.
- La elección de un leading peer para cada miembro del canal determina que peer se comunicará con el ordering service.
- Ninguna ledger pasa de un canal a otro. Esta separación se implementa en la configuración del chaincode, identity membership service y gossip data.

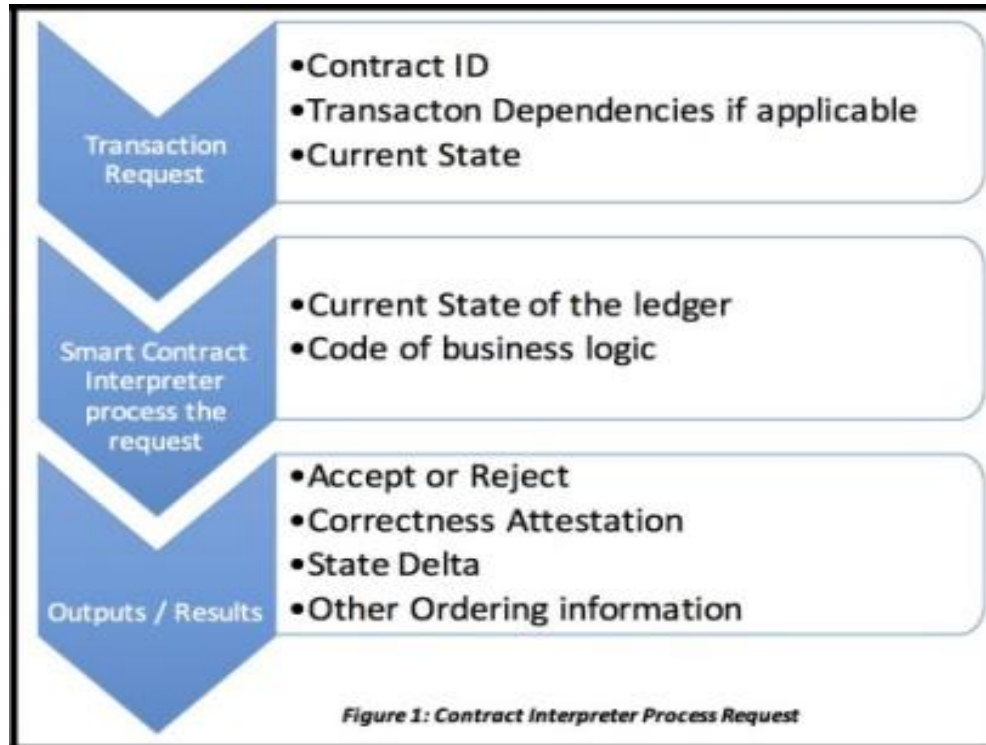
CHANNEL / CANAL

- Para crear un nuevo canal, el SDK del cliente llama al “configuration system chaincode” y hace referencia a propiedades como los anchor peers y miembros. Esta solicitud crea un bloque génesis para el canal que almacena la información de configuración sobre las políticas del canal, miembros y anchor peers.
- Cuando se añade un nuevo miembro a un canal existente el bloque se comparte con el nuevo miembro.

CHANNEL / CANAL - CONFIGTX

- La configuración del canal o configtx tiene las siguientes propiedades:
 - Versionado: Todos los elementos de configuración tienen una versión asociada que se avanza con cada modificación.
 - Autorizado: Cada elemento de configuración tiene una política asociada que controla si se permite o no la modificación de ese element.
 - Jerárquico: Un grupo de configuración tiene valores y políticas asociados.

CHAINCODE



ENDORSEMENT POLICY / POLÍTICAS DE RESPALDO

- Las políticas se implementan en diferentes niveles de una red Fabric. Cada nivel gobierna diferentes aspectos de la red.

Hyperledger Fabric Policy Hierarchy

System Channel

Consortium Membership and blockchain structure

Application Channel

Transaction networks, business logic

ACLs and smart contracts

Transactions, data, and events

ENDORSEMENT POLICY / POLÍTICAS DE RESPALDO

System channel configuration

- Cada red comienza con un ordering system channel. Únicamente debe haber un ordering system channel para un ordering service y es el primer canal ha ser creado.
- El system channel también contiene las organizaciones que son miembros del ordering service y de la organización de consorcio (para realizar transacciones).

ENDORSEMENT POLICY / POLÍTICAS DE RESPALDO

Application channel configuration

- Se utilizan para proporcionar un mecanismo de comunicación privado entre las organizaciones del consorcio.
- Las políticas en el application channel controlan la capacidad de agregar o quitar miembros del canal.

ENDORSEMENT POLICY / POLÍTICAS DE RESPALDO

Listas de control de acceso (ACL)

- Permiten configurar el acceso a los recursos asociando los recursos con las políticas existentes.
- La configuración predeterminada del ACL está visible en el archivo configtx.yaml en la sección Application: &ApplicationDefaults.

```
# ACL policy for chaincode to chaincode invocation
peer/ChaincodeToChaincode: /Channel/Application/
Readers
```

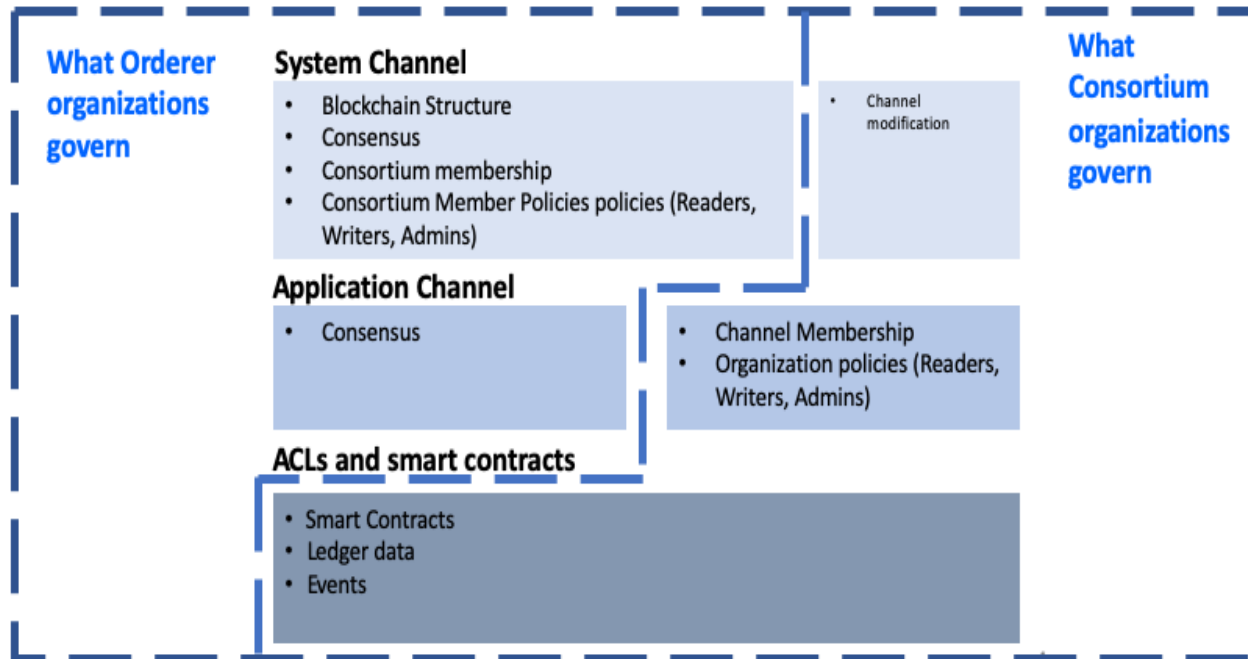
ENDORSEMENT POLICY / POLÍTICAS DE RESPALDO

Fabric policy domains

- Las políticas de Fabric son flexibles y se pueden configurar. La estructura de políticas se divide entre partes gobernadas por las organizaciones del Ordering Service y partes gobernadas por miembros del consorcio.

ENDORSEMENT POLICY / POLÍTICAS DE RESPALDO

Hyperledger Fabric Policy Hierarchy



ENDORSEMENT POLICY / POLÍTICAS DE RESPALDO

Escribir políticas en Fabric

- Para cambiar algo en Fabric, la política asociada con el recurso indica quién debe aprobarlo.
- Existen 2 tipos de aprobaciones:
 - Explícita: Signature
 - Implícitas: ImplicitMeta

ENDORSEMENT POLICY / POLÍTICAS DE RESPALDO

Políticas Signature

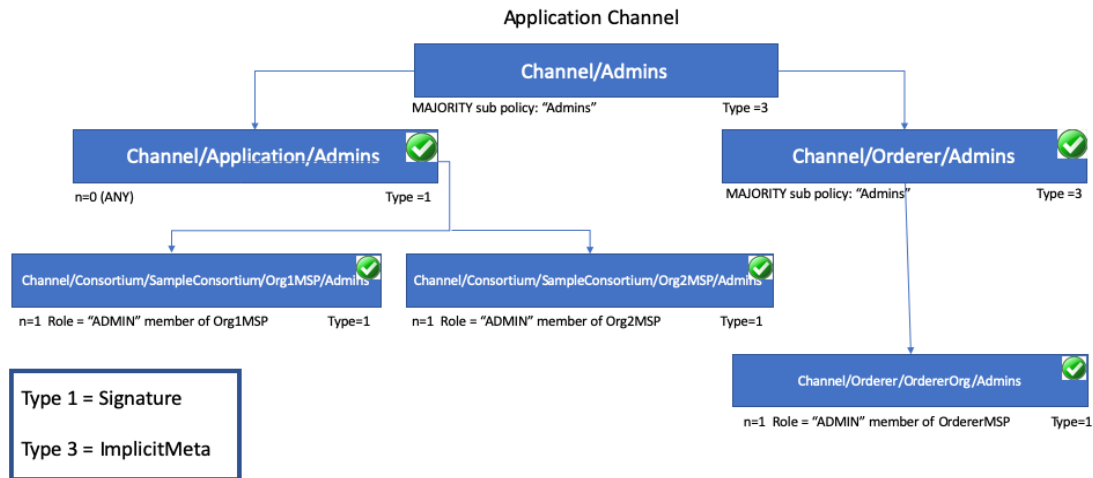
- Define el tipo específico de usuario que debe firmar para que se cumpla una política: OR ('Org1.peer', 'Org2.peer').

Políticas ImplicitMeta

- Solo son válidas en el context de configuración de un canal, que se basa en una jerarquía escalonada de políticas Signature.

ENDORSEMENT POLICY / POLÍTICAS DE RESPALDO

Políticas ImplicitMeta



In order for the Channel/Admins policy to be satisfied, every sub-policy under it in the configuration hierarchy must be satisfied.

ENDORSEMENT POLICY / POLÍTICAS DE RESPALDO

- Definir el conjunto de organizaciones que deben respaldar una transacción para que sea válida.
- Se utiliza el format MSP.ROLE, donde MSP representa el MSP ID y el ROLE representa uno de los 4 roles: member, admin, client y peer.
Ejemplos:
 - 'Org0.admin': cualquier administrador de Org0 MSP
 - 'Org1.member': cualquier miembro de Org1 MSP
 - 'Org1.client': cualquier cliente de Org1 MSP
 - 'Org1.peer': cualquier peer de Org1 MSP

ENDORSEMENT POLICY / POLÍTICAS DE RESPALDO

- `AND('Org1.member', 'Org2.member', 'Org3.member')` - Se necesita la firma de cada uno de los miembros.
- `OR('Org1.member', 'Org2.member')` - Se necesita la firma de uno de los 2 miembros.
- `OR('Org1.member', AND('Org2.member', 'Org3.member'))` – Se necesita una firma del miembro de Org1 o una firma de un miembro de Org2 y Org3

ENDORSEMENT POLICY / POLÍTICAS DE RESPALDO

```
- &Org1
# DefaultOrg defines the organization which is used in the sampleconfig
# of the fabric.git development environment
Name: Org1MSP
# ID to load the MSP definition as
ID: Org1MSP
MSPDir: crypto-config/peerOrganizations/org1.example.com/msp
# Policies defines the set of policies at this level of the config tree
# For organization policies, their canonical path is usually
#   /Channel/<Application|Orderer>/<OrgName>/<PolicyName>
Policies:
  Readers:
    Type: Signature
    Rule: "OR('Org1MSP.admin', 'Org1MSP.peer', 'Org1MSP.client')"
  Writers:
    Type: Signature
    Rule: "OR('Org1MSP.admin', 'Org1MSP.client')"
  Admins:
    Type: Signature
    Rule: "OR('Org1MSP.admin')"
  Endorsement:
    Type: Signature
    Rule: "OR('Org1MSP.peer')"
```

MEMBERSHIP SERVICE PROVIDER (MSP)

- Es un mecanismo que proporciona a los miembros un conjunto de roles y permisos dentro de la red.
- La Certification Authority emite identidades generando una clave pública y una clave privada, que se utilizan para probar la identidad.
- Como la clave privada no es pública, se requiere un mecanismo (MSP) para hacer esta comprobación.
- El MSP es el mecanismo que permite comprobar y reconocer las identidades sin revelar la clave privada.
- Tipos de MSP:
 - Local MSP
 - Channel MSP

MEMBERSHIP SERVICE PROVIDER (MSP)

- El Certification Authority genera los certificados que representan las identidades. El MSP contiene una lista de identidades autorizadas.
- El MSP convierte una identidad en un rol al identificar los privilegios que tiene la identidad en un nodo o canal.
- Cuando un usuario está registrado con un CA de Fabric, se le debe asociar un rol: admin, peer, client, orderer o member.

RED

- Un red blockchain permissionada en Fabric es una infraestructura técnica que proporciona servicios de ledger.
- Varias organizaciones se unen como un consorcio para formar la red y sus permisos están determinados por un conjunto de políticas acordadas por el consorcio.

RED - EJEMPLO

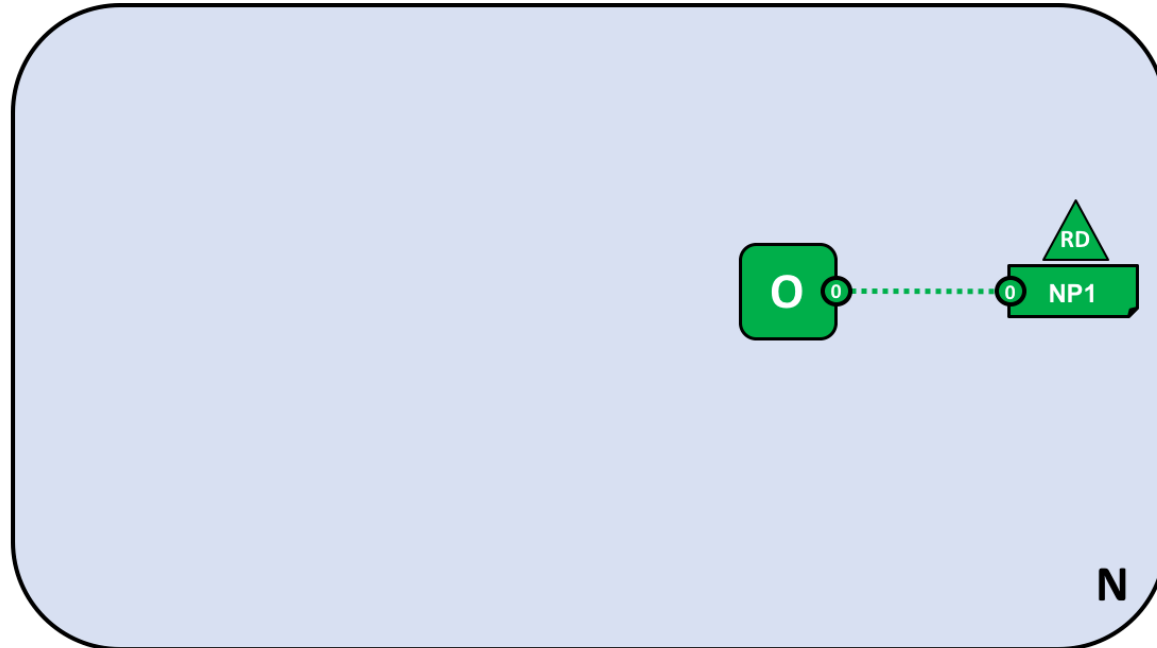
■ <https://hyperledger-fabric.readthedocs.io/en/release-1.2/network/network.html>

- Organizaciones RA, RB, RC y RD crean una red.
- RA: 3 peers y 2 aplicaciones cliente.
- RB: 4 peers y 1 aplicación cliente.
- RC: 3 peers y 2 aplicaciones cliente.
- RD: 4 orderers.
- RA y RB deciden formar un consorcio y usar un canal separado entre ellos.
- RB y RC deciden crear un consorcio y usar un canal separado entre ellos.
- Cada canal tiene sus políticas.

RED - EJEMPLO

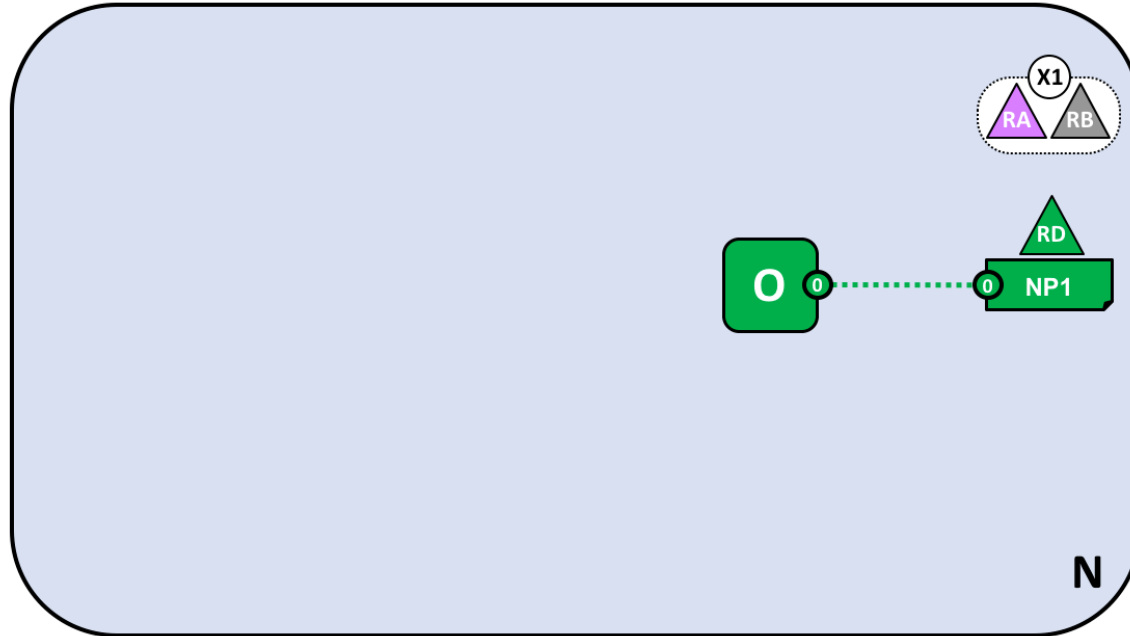
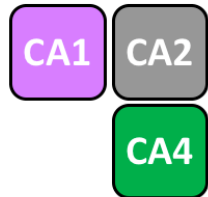
■ 1-Crear red

CA4



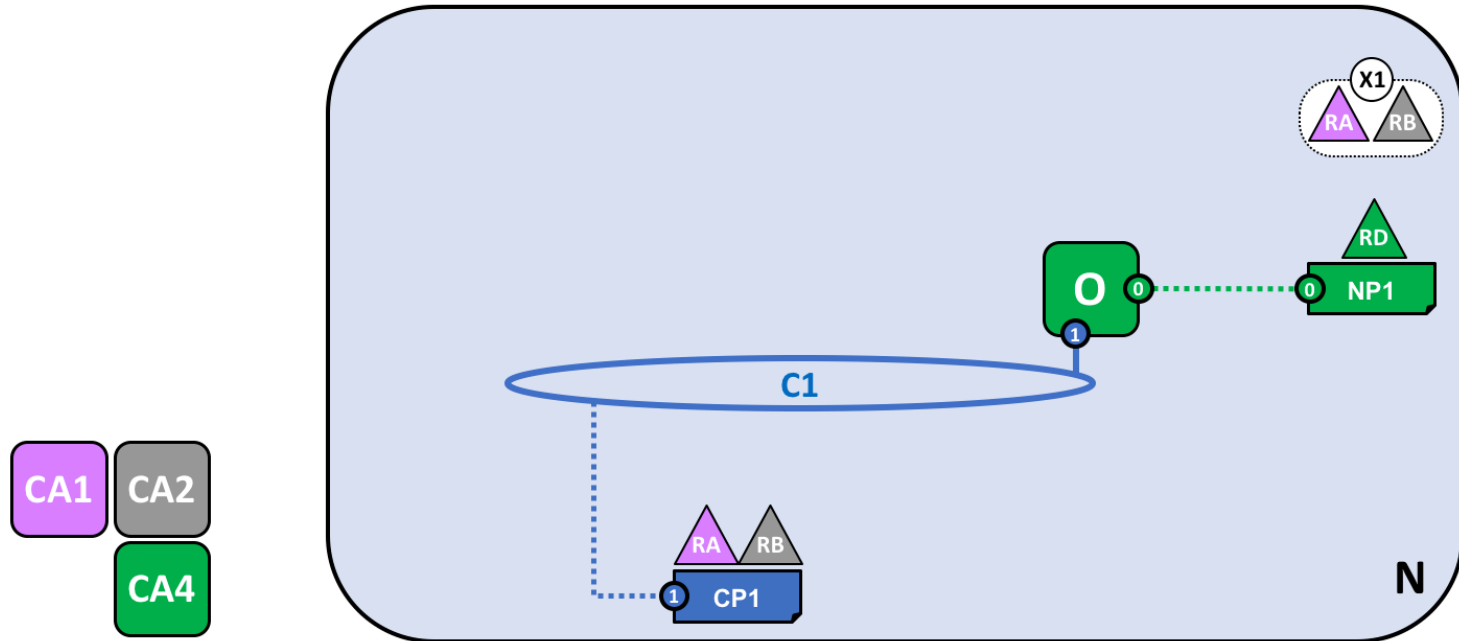
RED - EJEMPLO

■ 2-Definir consorcio



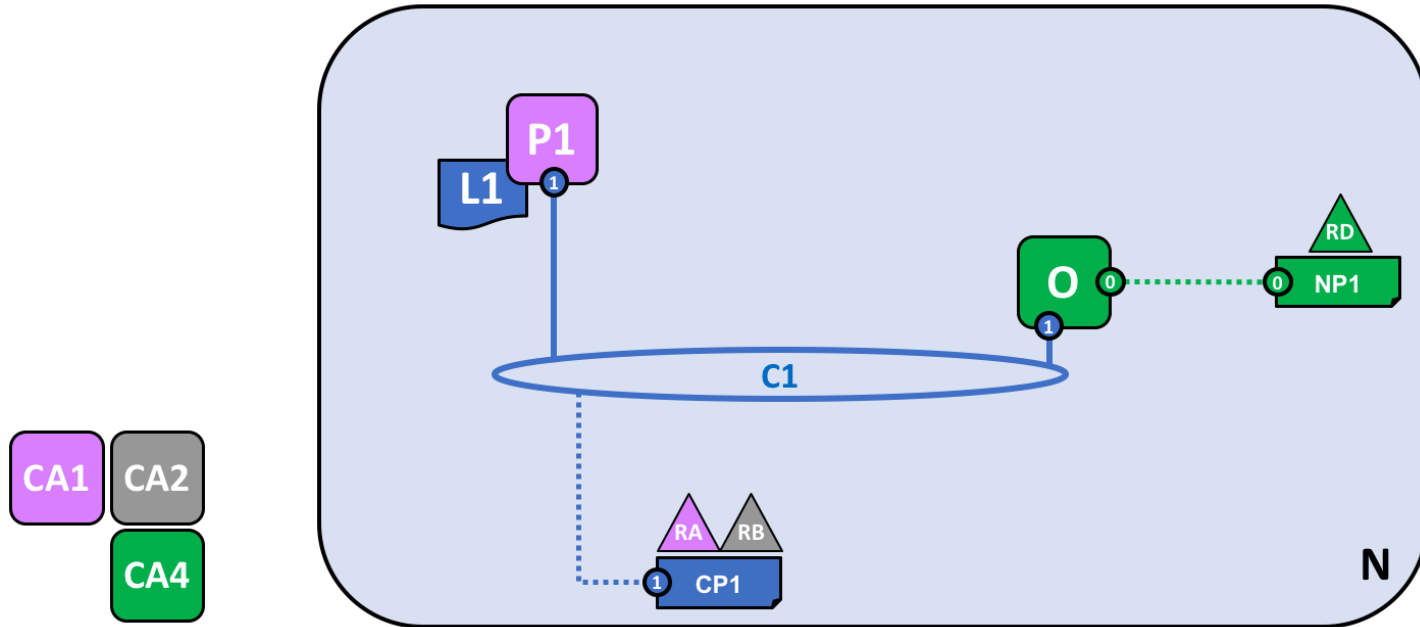
RED - EJEMPLO

■ 3-Crear canal para el consorcio



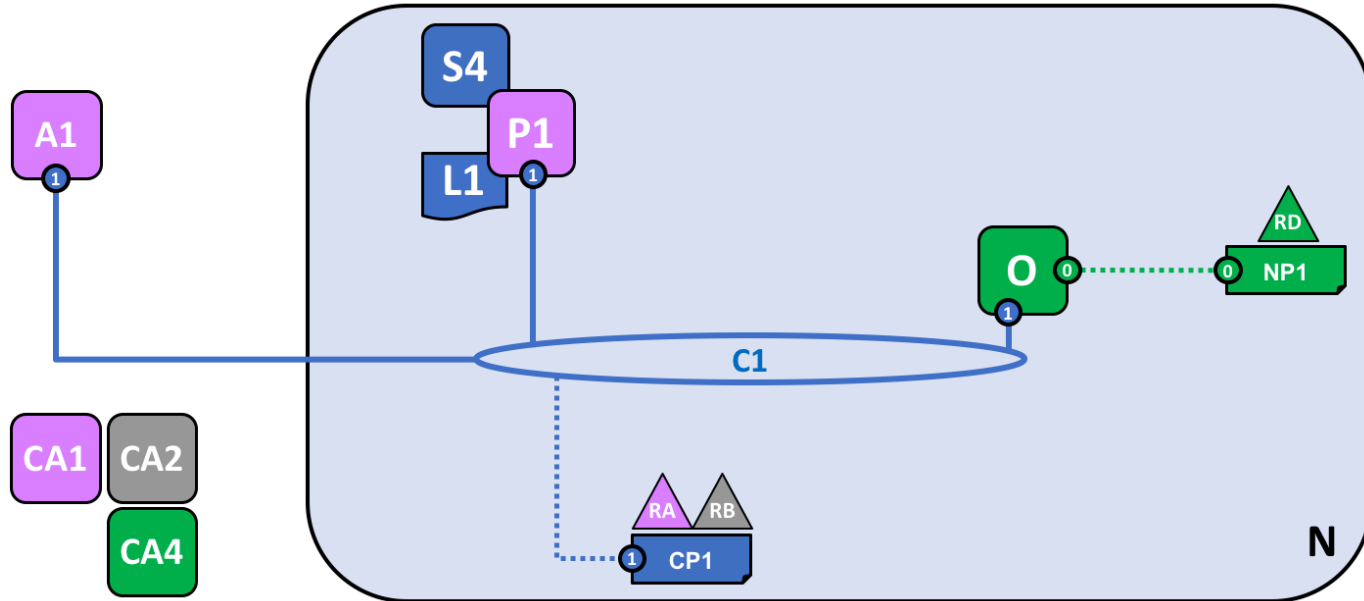
RED - EJEMPLO

■ 4-Peers y canal



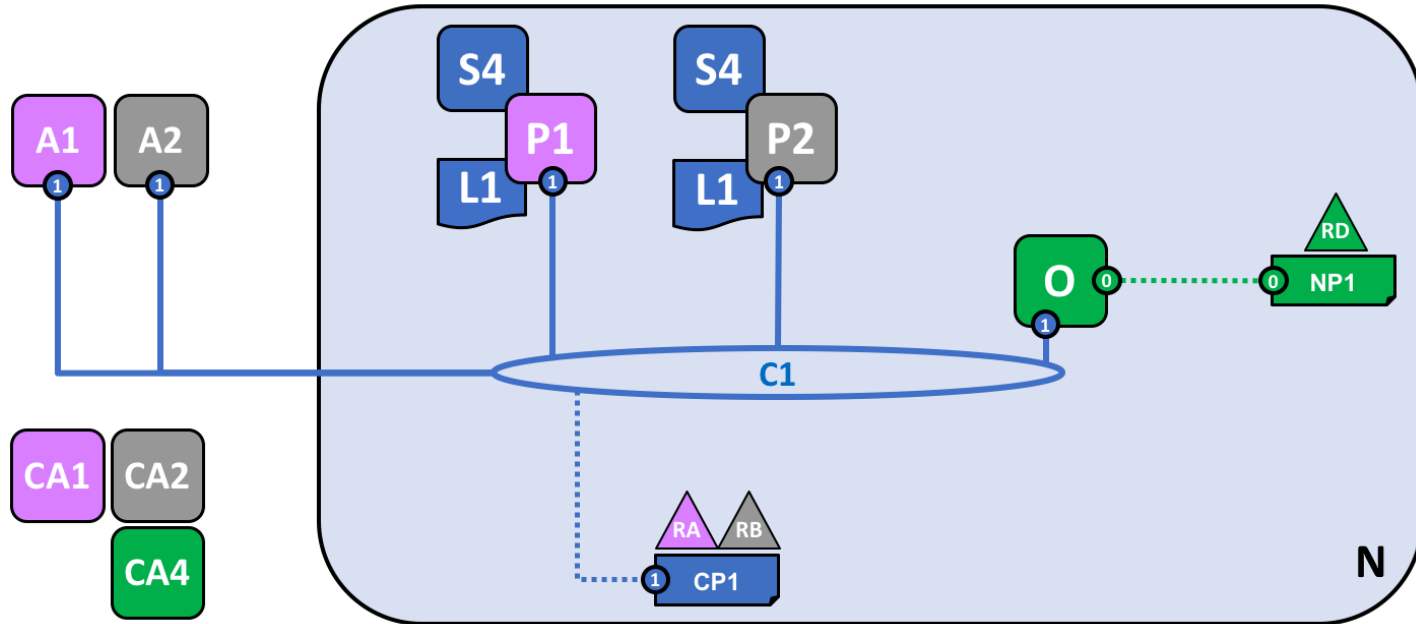
RED - EJEMPLO

■ 5-Aplicación y chaincode



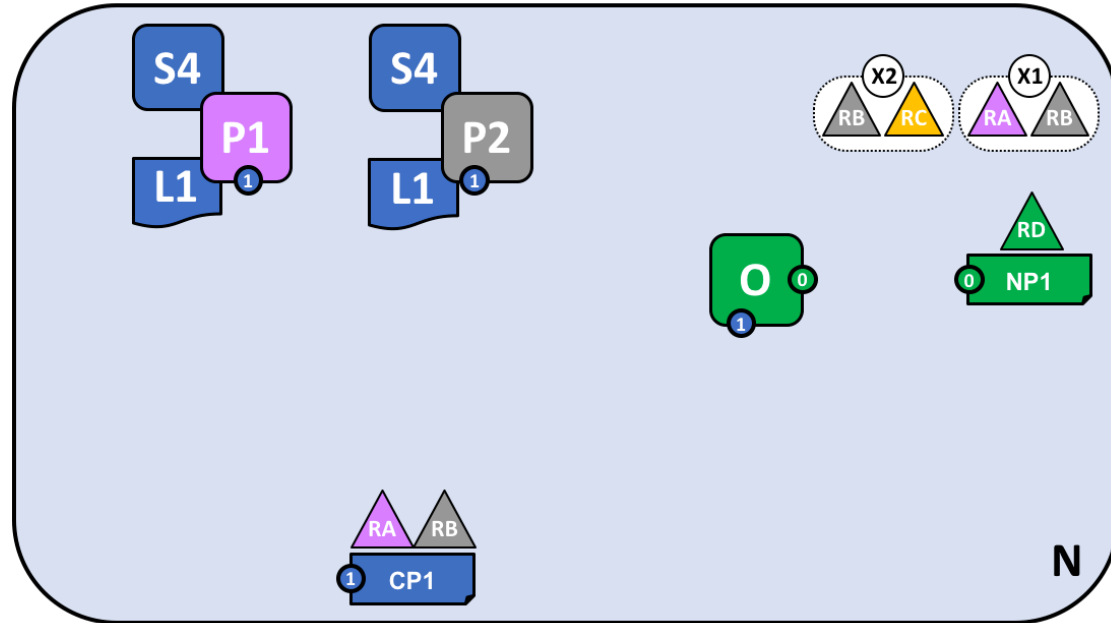
RED - EJEMPLO

■ 6-Aumentar la red

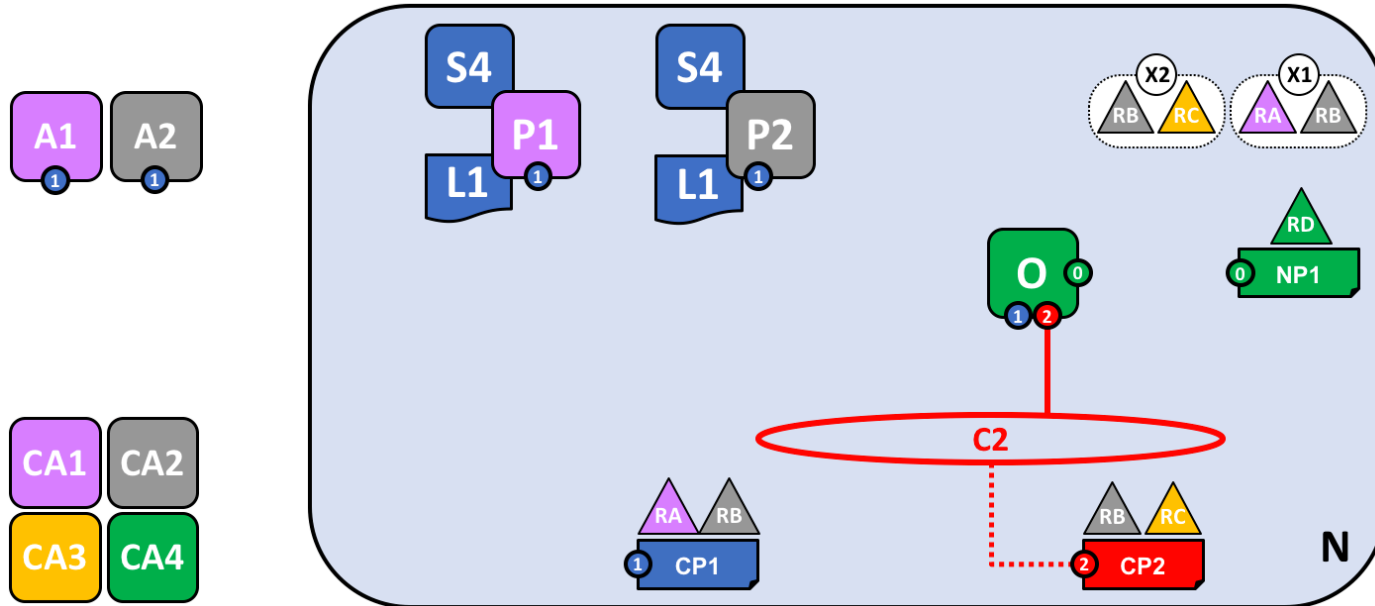


RED - EJEMPLO

■ 7-Añadir nuevo consorcio

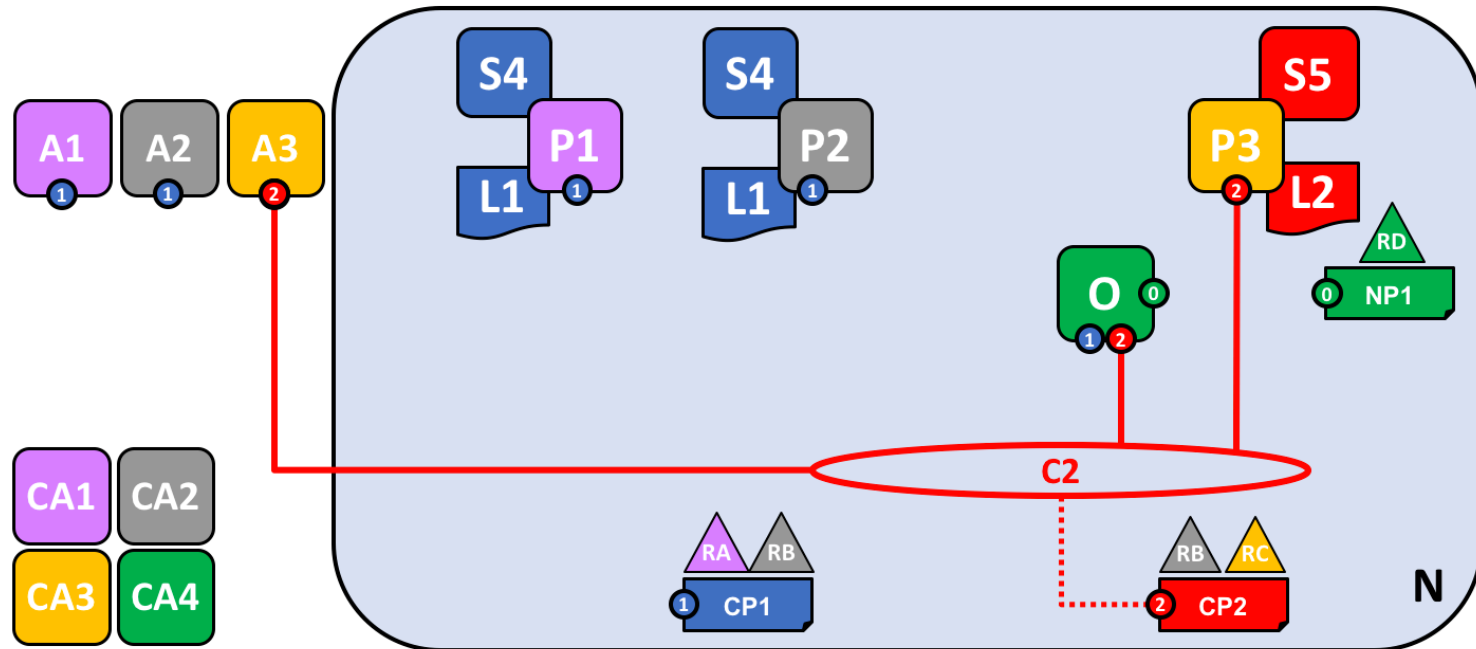


■ 8-Añadir nuevo canal



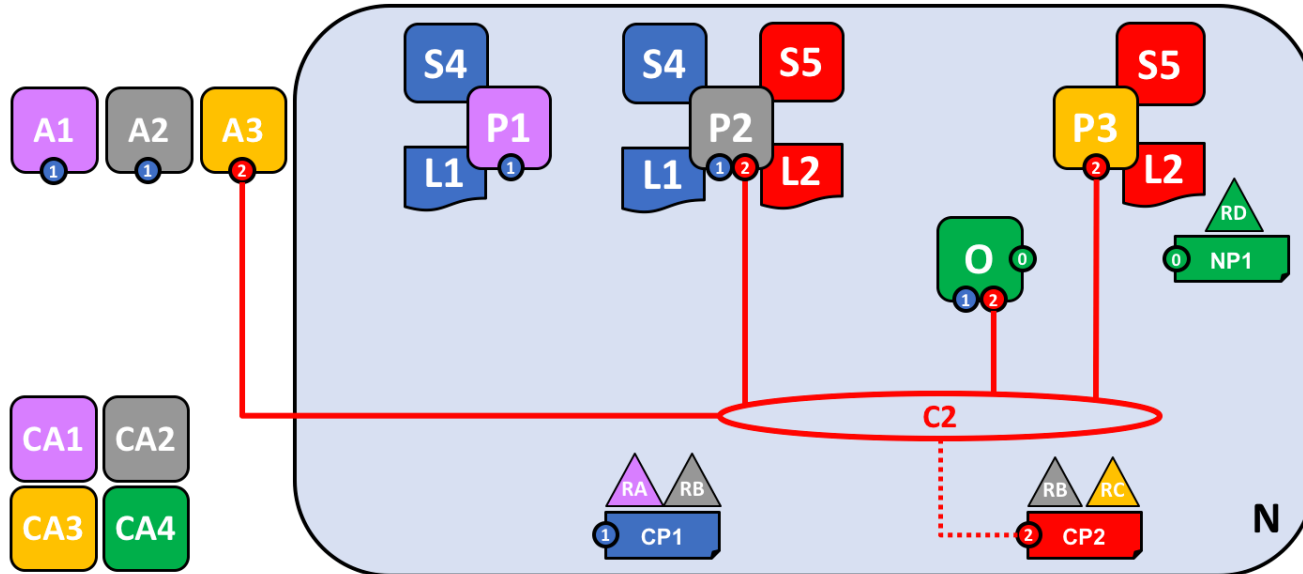
RED - EJEMPLO

■ 9-Añadir un peer



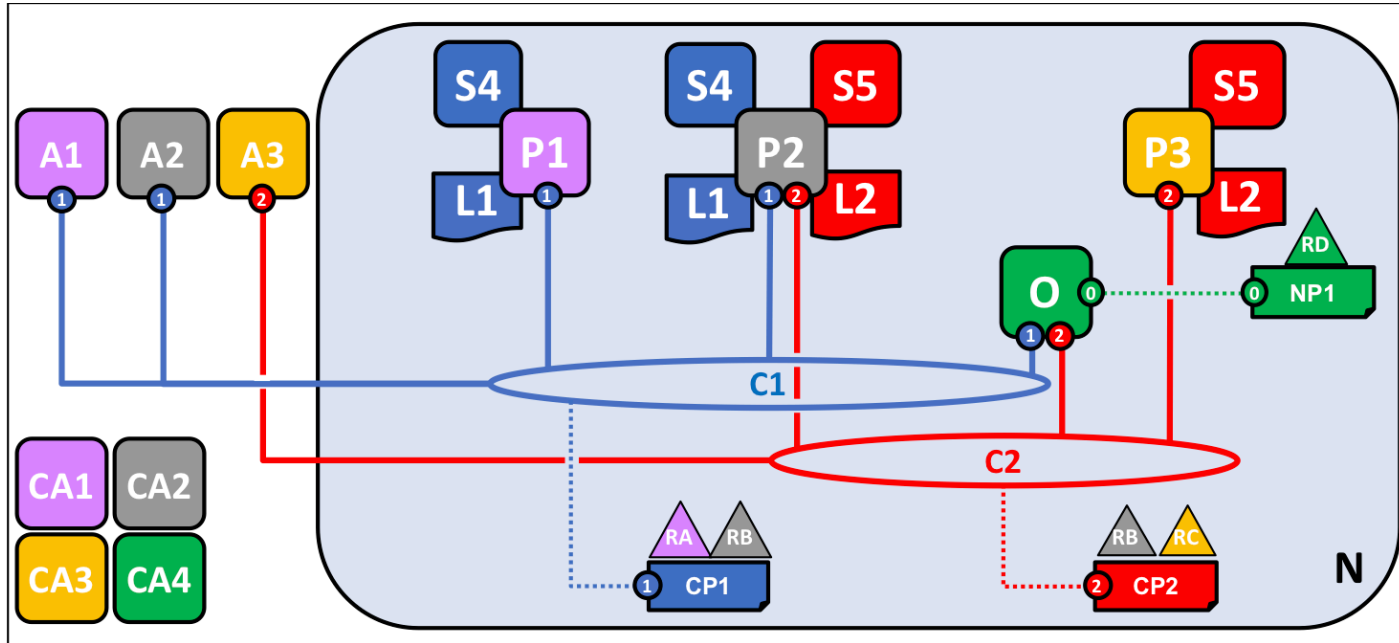
RED - EJEMPLO

- 10-Añadir un peer a multiples canales



RED - EJEMPLO

■ Red formada



REQUISITOS LINUX OS

- <https://hyperledger-fabric.readthedocs.io/en/release-2.2/prereqs.html>
- Git
- Curl
- Dockers & Docker Compose
- GO
- Node.js & NPM

REQUISITOS WINDOWS OS

- <https://hyperledger-fabric.readthedocs.io/en/release-2.1/prereqs.html>
- Git
- Curl
- Dockers & Docker Compose
- GO
- Node.js & NPM
- Python
- Extras para Windows *

PONTE A PRUEBA

- Verdadero o Falso: Python 2.7 es necesario para el desarrollo.
- Verdadero o Falso: Fabric-ca es necesario para emitir identidades.
- Verdadero o Falso: El ordering service es necesario para el consenso.

A CONTINUACIÓN

- Instalación e instanciar chaincode
- Configurar políticas
- Definir política de datos privados
- Actualizar y/o modificar chaincode

SESIONES DE TUTORÍA PRIVADAS DE HYPERLEDGER

- Ofrecemos sesiones de tutoría privada. Si necesita ayuda adicional en este curso, por favor contáctenos por correo electrónico en: **info@myhsts.org**



coding-bootcamps.com

Gracias, nos vemos en Coding Bootcamps



Coding
Bootcamps