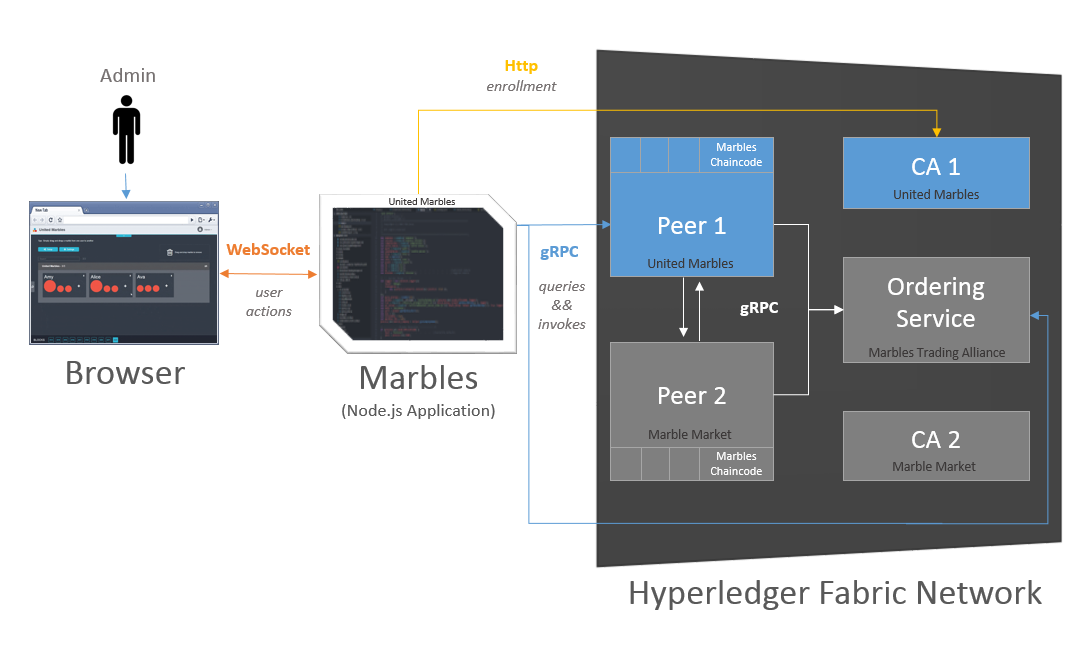
# Hyperledger Fabric Workflow

<https://github.com/IBM-Blockchain/marbles>

Above link clearly explains the end-to-end flow of a user's journey with an application on Fabric. Intermittently it also touches on enrollment secrets, certs storage etc.



## Detailed Workflow

<https://hyperledger-fabric.readthedocs.io/en/release-1.1/txflow.html>

Initiating Transaction:

1. User enrolls via Fabric CA Client, gets certificates and keys.

*(The Writing policy is defined at channel creation time, and determines which user is entitled to submit a transaction to that channel.) See detailed steps for enrolling under Identity handling.*

1. The app on the browser then raises the transaction request/proposal.
2. The Node SDK intercepts the request and package the transaction proposal into the properly architected format (protocol buffer over gRPC) and takes the user’s cryptographic credentials to produce a unique signature for this transaction proposal.
3. This is then broadcast to all peers.

Transaction execution:

1. The endorsing peers verify the transaction proposal, take inputs from it, simulate the execution of the chaincode, generate results and read/write set. Ledger state is not updated at this stage.
2. The set of these values, along with the endorsing peer’s signature is passed back as a “proposal response” to the SDK which parses the payload for the application to consume.

Response handling:

1. If the transaction only queried the ledger, the response is displayed to the user.
2. Any other type of transaction (invoke), is sent to the Ordering service to update the ledger.

Ordering:

1. The application “broadcasts” the transaction proposal and response within a “transaction message” to the Ordering Service.
2. The transaction will contain the read/write sets, the endorsing peers’ signatures and the Channel ID.
3. The Orderer simply receives transactions from all channels in the network, orders them chronologically by channel, and creates blocks of transactions per channel.

Transaction Validation and Committing:

1. The transactions within the block are validated to ensure endorsement policy is fulfilled and to ensure that there have been no changes to ledger state for read set variables since the read set was generated by the transaction execution.
2. Each peer commits the transaction by appending the block to the chain.
3. The peer also emits an event to notify the client application that the transaction has been immutably appended to the chain.

## Identity handling

<https://hyperledger-fabric-ca.readthedocs.io/en/latest/users-guide.html#enrolling-the-bootstrap-identity>

The following steps are to be followed to enroll a user during step 1 of the workflow detailed above:

1. **Enrolling (logging in) the bootstrap identity**: Fabric CA Client should enroll the basic admin identity. For example, following command enrolls an identity whose ID is **admin** and password is **adminpw** by calling Fabric CA server that is running locally at 7054 port.

export FABRIC\_CA\_CLIENT\_HOME=$HOME/fabric-ca/clients/admin

fabric-ca-client enroll -u <http://admin:adminpw@localhost:7054>

(This user admin/adminpw is the bootstrap user initialized in the Fabric-CA-server during CA setup using the cmd: fabric-ca-server init -b “admin:adminpw”.)

The enroll command stores an enrollment certificate (ECert), corresponding private key and CA certificate chain PEM files in the subdirectories of the Fabric CA client’s **msp** directory.

1. **Registering a new identity**: Both admin and peer identities may be registered using a previously enrolled admin identity (who is authorized to register new ids). For example, the following command registers the **peer1** identity:

export FABRIC\_CA\_CLIENT\_HOME=$HOME/fabric-ca/clients/admin

fabric-ca-client register --id.name peer1 --id.type peer --id.affiliation org1.department1 --id.secret peer1pw

1. **Enrolling the peer identity**: The following command enrolls peer1.

export FABRIC\_CA\_CLIENT\_HOME=$HOME/fabric-ca/clients/peer1

fabric-ca-client enroll -u http://peer1:peer1pw@localhost:7054 -M $FABRIC\_CA\_CLIENT\_HOME/msp

The –M flag indicates the location where the crypto material (keys + certs) will be stored.

All the above-mentioned steps can also be done via the NodeJS SDK (pending installation of the fabric\_client and fabric\_ca\_client packages) as shown in the link below:

<https://www.skcript.com/svr/setting-up-restful-api-server-for-hyperledger-fabric-with-nodejs-sdk/>

***NOTE: Most of the steps are identical, except for one additional step to create a connection profile. This is the config document which specifies all the gRPC connection endpoints and storage locations for all the generated crypto materials. For starters, these are all stored within the respective containers for each of the components.***