Building Hyperledger Fabric

The following instructions assume that you have already set up your development environment.

To build Hyperledger Fabric:

```
cd $GOPATH/src/github.com/hyperledger/fabric
make dist-clean all

make dist-clean all
```

Building the documentation

If you are contributing to the documentation, you can build the Fabric documentation on your local machine. This allows you to check the formatting of your changes using your web browser before you open a pull request.

You need to download the following prerequisites before you can build the documentation:

- Python 3.7
- Pipenv

After you make your updates to the documentation source files, you can generate a build that includes your changes by running the following commands:

```
cd fabric/docs
pipenv install
pipenv shell
make html
```

This will generate all the html files in the `docs/build/html` folder. You can open any file to start browsing the updated documentation using your browser. If you want to make additional edits to the documentation, you can rerun `make html` to incorporate the changes.

Running the unit tests

Before running the unit tests, a PKCS #11 cryptographic token implementation must be installed and configured. The PKCS #11 API is used by the bccsp component of Fabric to interact with devices, such as hardware security modules (HSMs), that store cryptographic information and perform cryptographic computations. For test environments, SoftHSM can be used to satisfy this requirement.

Use the following command to run all unit tests:

SoftHSM can be installed with the following commands:

```
: :
```

make unit-test

sudo apt install libsofthsm2 # Ubuntu sudo yum install softhsm # CentOS brew install softhsm # macOS

To run a subset of tests, set the TEST_PKGS environment variable. Specify a list of packages (separated by space), for example:

Once SoftHSM is installed, additional configuration may be required. For example, the default configuration file stores token data in a system directory that unprivileged users are unable to write to.

Configuration typically involves copying `/etc/softhsm2.conf` to `$HOME/.config/softhsm2/softhsm2.conf` and changing `directories.tokendir` to an appropriate location. Please see the man page for `softhsm2.conf` for details.

export TEST_PKGS="github.com/hyperledger/fabric/core/ledger/..." make unit-test

After SoftHSM has been configured, the following command can be used to initialize the required token:

To run a specific test use the `-run RE` flag where RE is a regular expression that matches the test case name. To run tests with verbose output use the `-v` flag. For example, to run the `TestGetFoo` test case, change to the directory containing the `foo_test.go` and call/execute

```
softhsm2-util --init-token --slot 0 --label "ForFabric" --so-pin 1234 --pin 98765432
go test -v -run=TestGetFoo
```

If the test cannot find libsofthsm2.so in your environment, specify its path, the PIN and the label of the token through environment variables. For example, on macOS:

Running Node.js Client SDK Unit Tests

export PKCS11_LIB="/usr/local/Cellar/softhsm/2.5.0/lib/softhsm/libsofthsm2.so" export PKCS11_PIN=98765432 export PKCS11_LABEL="ForFabric"

You must also run the Node.js unit tests to ensure that the Node.js client SDK is not broken by your changes. To run the Node.js unit tests, follow the instructions here.

Use the following sequence to run all unit tests:

Configuration

Configuration utilizes the viper and cobra libraries.

cd $GOPATH/src/github.com/hyperledger/fabric make unit-test

There is a **core.yaml** file that contains the configuration for the peer process. Many of the configuration settings can be overridden on the command line by setting ENV variables that match the configuration setting, but by prefixing with *'CORE_'*. For example, setting *peer.networkId* can be accomplished with:

To run a subset of tests, set the TEST_PKGS environment variable. Specify a list of packages (separated by space), for example:

```
::
```

CORE_PEER_NETWORKID=custom-network-id peer

export TEST_PKGS="github.com/hyperledger/fabric/core/ledger/..." make unit-test

To run a specific test use the `-run RE` flag where RE is a regular expression that matches the test case name. To run tests with verbose output use the `-v` flag. For example, to run the `TestGetFoo` test case, change to the directory containing the `foo_test.go` and call/execute

```
go test -v -run=TestGetFoo
```

Running Node.js Client SDK Unit Tests

You must also run the Node.js unit tests to ensure that the Node.js client SDK is not broken by your changes. To run the Node.js unit tests, follow the instructions here.

Configuration

Configuration utilizes the viper and cobra libraries.

There is a **core.yaml** file that contains the configuration for the peer process. Many of the configuration settings can be overridden on the command line by setting ENV variables that match the configuration setting, but by prefixing with *'CORE_'*. For example, setting *peer.networkId* can be accomplished with:

```
CORE_PEER_NETWORKID=custom-network-id peer
```