Setting up the development environment

Prerequisites

- Git client, Go, and Docker as described at 准备阶段
- (macOS) Xcode must be installed
- (macOS) you may need to install gnutar, as macOS comes with bsdtar as the default, but the build uses some gnutar flags. You can use Homebrew to install it as follows:
- Git client
- Go version 1.14.x
- Docker version 18.03 or later
- (macOS) Xcode Command Line Tools
- SoftHSM
- jq

```
brew install gnu-tar
```

Steps

- (macOS) If you install gnutar, you should prepend the “gnubin” directory to the $PATH environment variable with something like:

Install the Prerequisites

For macOS, we recommend using Homebrew to manage the development prereqs. The Xcode command line tools will be installed as part of the Homebrew installation.

export PATH=/usr/local/opt/gnu-tar/libexec/gnubin:$PATH

Once Homebrew is ready, installing the necessary prerequisites is very easy:

- (macOS) Libtool. You can use Homebrew to install it as follows:

```
: :
```

brew install git go jq softhsm brew cask install –appdir=”/Applications” docker

brew install libtool

Docker Desktop must be launched to complete the installation so be sure to open the application after installing it:

- (only if using Vagrant) - Vagrant - 1.9 or later

- (only if using Vagrant) - VirtualBox - 5.0 or later
- BIOS Enabled Virtualization - Varies based on hardware

```
-  Note: The BIOS Enabled Virtualization may be within the CPU or
```

Security settings of the BIOS

open /Applications/Docker.app

Developing on Windows

Steps

On Windows 10 you should use the native Docker distribution and you may use the Windows PowerShell. However, for the `binaries` command to succeed you will still need to have the `uname` command available. You can get it as part of Git but beware that only the 64bit version is supported.

Set your GOPATH

Before running any `git clone` commands, run the following commands:

Make sure you have properly setup your Host's GOPATH environment variable. This allows for both building within the Host and the VM.

In case you installed Go into a different location from the standard one your Go distribution assumes, make sure that you also set GOROOT environment variable.

git config –global core.autocrlf false git config –global core.longpaths true

Note to Windows users

You can check the setting of these parameters with the following commands:

If you are running Windows, before running any `git clone` commands, run the following command.

```
::
```

git config –get core.autocrlf git config –get core.longpaths

git config –get core.autocrlf

These need to be `false` and `true` respectively.

If `core.autocrlf` is set to `true`, you must set it to `false` by running

The `curl` command that comes with Git and Docker Toolbox is old and does not handle properly the redirect used in 入门. Make sure you have and use a newer version which can be downloaded from the cURL downloads page

Clone the Hyperledger Fabric source

git config –global core.autocrlf false

First navigate to https://github.com/hyperledger/fabric and fork the fabric repository using the fork button in the top-right corner. After forking, clone the repository.

If you continue with `core.autocrlf` set to `true`, the `vagrant up` command will fail with the error:

```
``./setup.sh: /bin/bash^M: bad interpreter: No such file or directory``
```

mkdir -p github.com/<your_github_userid> cd github.com/<your_github_userid> git clone https://github.com/<your_github_userid>/fabric

Cloning the Hyperledger Fabric source

注解

If you are running Windows, before cloning the repository, run the following command:

First navigate to https://github.com/hyperledger/fabric and fork the fabric repository using the fork button in the top-right corner

Since Hyperledger Fabric is written in `Go`, you'll need to clone the forked repository to your $GOPATH/src directory. If your $GOPATH has multiple path components, then you will want to use the first one. There's a little bit of setup needed:

git config –get core.autocrlf

```
If ``core.autocrlf`` is set to ``true``, you must set it to ``false`` by
running:

cd $GOPATH/src
mkdir -p github.com/<your_github_userid>
cd github.com/<your_github_userid>
git clone https://github.com/<your_github_userid>/fabric

::

    git config --global core.autocrlf false
```

Configure SoftHSM

A PKCS #11 cryptographic token implementation is required to run the unit tests. The PKCS #11 API is used by the bccsp component of Fabric to interact with hardware security modules (HSMs) that

store cryptographic information and perform cryptographic computations. For test environments, SoftHSM can be used to satisfy this requirement.

SoftHSM generally requires additional configuration before it can be used. For example, the default configuration will attempt to store token data in a system directory that unprivileged users are unable to write to.

SoftHSM 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.

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

```
softhsm2-util --init-token --slot 0 --label "ForFabric" --so-pin 1234 --pin 98765432
```

If tests are unable to locate the libsofthsm2.so library in your environment, specify the library path, the PIN, and the label of your token in the appropriate environment variables. For example, on macOS:

```
export PKCS11_LIB="/usr/local/Cellar/softhsm/2.6.1/lib/softhsm/libsofthsm2.so"
export PKCS11_PIN=98765432
export PKCS11_LABEL="ForFabric"
```

Install the development tools

Once the repository is cloned, you can use `make` to install some of the tools used in the development environment. By default, these tools will be installed into `$HOME/go/bin`. Please be sure your `PATH` includes that directory.

```
make gotools
```

After installing the tools, the build environment can be verified by running a few commands.

```
make basic-checks integration-test-prereqs
ginkgo -r ./integration/nwo
```

If those commands completely successfully, you're ready to Go!

If you plan to use the Hyperledger Fabric application SDKs then be sure to check out their prerequisites in the Node.js SDK README and Java SDK README.