普天同签
Signatory Chain

# 软件使用手册

产品名称：联盟链搭建手册

版本：V1.0

杭州玺湖科技有限公司

目录

## 1. 公司简介

杭州玺湖科技有限公司全球首创基于区块链去中心化或多中心化多根共识共治的数字安全框架和底层技术（DeSe: Decentralized Security），是 60 年以来全球数字安全领域的一场框架性革命，也是区块链和实体经济，特别是区块链和数字安全领域结合的独特场景和接入点：除了区块链用于数据信任背书，还进一步将区块链思维内植入安全管理的内在逻辑。公司独创了一个崭新的蓝海，在该领域全球没有竞争对手，可以主导该领域的行业标准和话语权。目前所有其他现存的安全管理框架都是基于中心化等级式单根治理。我们的核心技术克服了当代中心化等级式数字系统安全管理的框架性漏洞和系统性风险，用区块链思维，搭建去中心化，多根共治扁平的数字系统安全管理框架和标准。公司所研发的颠覆性技术是一个普遍的底层方案，可应用在数字化的所有层级和行业：硬件，操作系统，数据库，中间件，应用层等，以及所有行业和场景，包括办公，农业，医疗，教育，食品安全，国防等。具有一个 2 万亿的左右的安全升级市场，10 万亿左右的区块链落地实体经济的市场。公司的核心技术可以通过柔性拥抱的方式，对现有的安全系统无感无缝升级，不需要修改现有的系统，极大地降低了安全升级成本。

## 2. 产品的名称，目的，和版本

### 2.1 产品名称

区块链安装手册

### 2.2 产品的目标

本手册旨在指导用户成功安装和配置 Hyperledger Fabric 联盟链环境，帮助用户快速上手并高效地搭建区块链平台。

### 2.3 产品的版本

版本号：V1.0

## 3. 产品的主要功能模块和流程

### 3.1 部署预先操作

#### 1. Docker 安装

安装 Hyperledger Fabric 所需的环境，包括 Docker 和 Docker Compose。

#### 2. 更改 host（本文档单机节点安装教程）

配置节点之间的网络连接，确保各节点可以互相访问。

#### 3. 安装区块链工具

安装必要的节点部署工具，以支持 Fabric 网络的搭建和管理。

#### 4. 上传链码文件、区块链证书生成、区块链启动相关文件

上传链码文件，并生成区块链所需的证书和启动配置文件，以确保节点部署顺利完成。

### 3.2 容器内操作

#### 1. 进入 cli 容器

设置安装所需环境变量 docker exec -it cli bash

#### 2. 设置环境变量

配置并设置区块链网络安装所需的环境变量，确保各项操作能够顺利进行。

#### 3. 创建通道

创建用于节点间通讯的通道，以实现区块链网络的基本通信功能。

#### 4. 加入通道

将各个节点加入已创建的通道，以实现网络中的节点协作

#### 5. 更新锚节点

更新并配置锚节点，确保每个组织的节点能够正确识别和连接到通道。

#### 6. 安装链码

在各个节点上安装所需的链码，为后续的链码实例化做准备。

#### 7. 实例化链码

将链码实例化并部署到通道上，使其在网络中可用。

#### 8. 验证安装

验证链码是否成功安装并实例化，确保区块链网络功能正常。

## 4. 每个功能模块的使用方法

### 4.1 部署预先操作

#### 1. Docker 安装

**sudo apt-get install docker.io -y**

#### 2. 更改 host(本文档单机节点安装教程)

host 加入以下内容
192.168.101.131 orderer1.signatorychain.com
192.168.101.131 peer0.org1.signatorychain.com
192.168.101.131 peer1.org1.signatorychain.com

#### 3. 更改 host(本文档单机节点安装教程)

上传工具包并移动到该目录
**sudo mv tools.zip /usr/local/bin/**
切到指定目录
**cd /usr/local/bin/**
解压压缩包
**sudo unzip tools.zip**
删除压缩包

```
sudo rm tools.zip
```

赋予权限

```
sudo chmod +x ./*
```

### 5. 上传链码文件、区块链证书生成、区块链启动相关文件

创建/fabric 目录及用 ftp 工具上传相关文件到服务器

## 4.2 容器内操作

### 1. 进入 cli 容器

设置安装所需环境变量 docker exec -it cli bash

### 2. 设置环境变量

设置安装所需环境变量

```
export CRYPTO_PATH="/opt/gopath/src/github.com/hyperledger/fabric/peer"
export CHANNEL_NAME="escychannel"
export ORDERER_CA="${CRYPTO_PATH}/crypto/ordererOrganizations/signatorychain.com/msp/tlscacerts/tlsca.signatorychain.com-cert.pem"
export ORDERER_ADDRESS="orderer1.signatorychain.com:7050"
export CC_PATH="github.com/chaincode/escy-cc"
export CC_NAME="escycc"
export VERSION="1.0"
export LANGUAGE="golang"
export P0_O1_CRT="$CRYPTO_PATH/crypto/peerOrganizations/org1.signatorychain.com/peers/peer0.org1.signatorychain.com/tls/ca.crt"
export P1_O1_CRT="$CRYPTO_PATH/crypto/peerOrganizations/org1.signatorychain.com/peers/peer1.org1.signatorychain.com/tls/ca.crt"
```

### 3. 创建通道

创建节点通讯通道

```
export CORE_PEER_LOCALMSPID="Org1MSP"
export CORE_PEER_MSPCONFIGPATH=${CRYPTO_PATH}/crypto/peerOrganizations/org1.signatorychain.com/users/Admin@org1.signatorychain.com/msp
peer channel create -o ${ORDERER_ADDRESS} -c ${CHANNEL_NAME} -f ./channel-artifacts/channel.tx --tls --cafile ${ORDERER_CA)
```

### 4. 加入通道

peer0.org1.signatorychain.com 加入通道

```
export CORE_PEER_ADDRESS=peer0.org1.signatorychain.com:7051
export CORE_PEER_LOCALMSPID="Org1MSP"
export CORE_PEER_TLS_ROOTCERT_FILE=${CRYPTO_PATH}/crypto/peerOrganizations/org1.signatorychain.com/peers/peer0.org1.signatorychain.com/tls/ca.crt
export CORE_PEER_MSPCONFIGPATH=${CRYPTO_PATH}/crypto/peerOrganizations/org1.signatorychain.com/users/Admin@org1.signatorychain.com/msp
peer channel join -b ${CHANNEL_NAME}.block
```

peer1.org1.signatorychain.com 加入通道

```
export CORE_PEER_ADDRESS=peer1.org1.signatorychain.com:8051
export CORE_PEER_LOCALMSPID="Org1MSP"
export CORE_PEER_TLS_ROOTCERT_FILE=${CRYPTO_PATH}/crypto/peerOrganizations/org1.signatorychain.com/peers/peer1.org1.signatorychain.com/tls/ca.crt
export CORE_PEER_MSPCONFIGPATH=${CRYPTO_PATH}/crypto/peerOrganizations/org1.signatorychain.com/users/Admin@org1.signatorychain.com/msp
peer channel join -b ${CHANNEL_NAME}.block
```

### 5. 更新锚节点

peer0.org1.signatorychain.com 更新锚节点

```
export CORE_PEER_ADDRESS=peer0.org1.signatorychain.com:7051
export CORE_PEER_LOCALMSPID="Org1MSP"
export CORE_PEER_TLS_ROOTCERT_FILE=${CRYPTO_PATH}/crypto/peerOrganizations/org1.signatorychain.com/peers/peer0.org1.signatorychain.com/tls/ca.crt
export CORE_PEER_MSPCONFIGPATH=${CRYPTO_PATH}/crypto/peerOrganizations/org1.signatorychain.com/users/Admin@org1.signatorychain.com/msp
peer channel update -o orderer1.signatorychain.com:7050 --ordererTLSHostnameOverride orderer1.signatorychain.com -c ${CHANNEL_NAME} -f ./channel-artifacts/Org1MSPanchors.tx --tls --cafile "$ORDERER_CA"
```

peer1.org1.signatorychain.com 安装链码

```
export CORE_PEER_ADDRESS=peer1.org1.signatorychain.com:8051
export CORE_PEER_LOCALMSPID="Org1MSP"
export CORE_PEER_TLS_ROOTCERT_FILE=${CRYPTO_PATH}/crypto/peerOrganizations/org1.signatorychain.com/peers/peer1.org1.signatorychain.com/tls/ca.crt
export CORE_PEER_MSPCONFIGPATH=${CRYPTO_PATH}/crypto/peerOrganizations/org1.signatorychain.com/users/Admin@org1.signatorychain.com/msp
peer chaincode install -n ${CC_NAME} -v ${VERSION} -l ${LANGUAGE} -p ${CC_PATH}
```

## 6. 安装链码

peer0.org1.signatorychain.com 安装链码

```
export CORE_PEER_ADDRESS=peer0.org1.signatorychain.com:7051
export CORE_PEER_LOCALMSPID="Org1MSP"
export CORE_PEER_TLS_ROOTCERT_FILE=${CRYPTO_PATH}/crypto/peerOrganizations/org1.signatorychain.com/peers/peer0.org1.signatorychain.com/tls/ca.crt
export CORE_PEER_MSPCONFIGPATH=${CRYPTO_PATH}/crypto/peerOrganizations/org1.signatorychain.com/users/Admin@org1.signatorychain.com/msp
peer chaincode install -n ${CC_NAME} -v ${VERSION} -l ${LANGUAGE} -p ${CC_PATH}
```

7. **实例化链码**

```
peer chaincode instantiate -o ${ORDERER_ADDRESS} -C ${CHANNEL_NAME} -n ${CC_NAME} -l ${LANGUAGE} -v ${VERSION} -c '{"Args":[""]}' --tls --cafile ${ORDERER_CA} -P "AND ('Org1MSP.member')"
```

## 8. 验证安装

调用存证方法

```
peer chaincode invoke -o ${ORDERER_ADDRESS} -C ${CHANNEL_NAME} -n ${CC_NAME} -c '{"function":"idissue","Args":["escy","test","test","1599738061","1698892683","初始测试数据"]}' --tls --cafile ${ORDERER_CA} --peerAddresses peer0.org1.signatorychain.com:7051 --tlsRootCertFiles ${P0_O1_CRT}
```

查询数据

```
peer chaincode query -C ${CHANNEL_NAME} -n ${CC_NAME} -c '{"function":"idquery","Args":["test"]}'
```