SimpleChain 联盟链（基础版）

安装部署手册

目录

1 系统部署

SimpleChain 联盟链（基础版）包括两大组成部分：

- 基础链

  联盟链节点网络，即通常所说的区块链

- 管理平台

  提供 Web 界面，实现对联盟链的管理和监控

SimpleChain 联盟链（基础版）的整体部署如下图所示：

2 基础链的部署

2.1 硬件需求

支持将 SimpleChain 联盟链（基础版）节点软件部署到物理机或虚拟机服务器上，根据节点类型的不同要求不同的硬件配置。

注：以下提到的共识节点是指 PoA 协议的 Validator 节点，负责出块，并不一定是“管理者节点”。这里的节点类型与在《SimpleChain 联盟链（基础版）用户手册》描述的节点类型有所不同，前者是从谁出块的角度，后者是从权限管理的角度。

2.1.1 推荐硬件配置

- 共识节点

CPU	内存	磁盘	网络带宽
16 核	32G	SSD，容量按上链数据量评估	1000MB

注：硬盘所需容量可按照 1：20 进行估计，例如：实际上链数据 5G，需要 100G 硬盘空间。以下普通节点的估计与此相同。

- 普通节点

CPU	内存	磁盘	网络带宽
8 核	16G	SSD，容量按上链数据量评估	1000MB

- 管理平台

CPU	内存	磁盘	网络带宽
8 核	16G	HDD，500G	1000MB

2.1.2 最小硬件配置

- 共识节点

CPU	内存	磁盘	网络带宽
8 核	16G	HDD，容量按上链数据量评估	100MB

注：硬盘所需容量可按照 1：20 进行估计，例如：实际上链数据 5G，需要 100G 硬盘空间。以下普通节点的估计与此相同。

- 普通节点

CPU	内存	磁盘	网络带宽
4 核	8G	HDD，容量按上链数据量评估	100MB

- 管理平台

CPU	内存	磁盘	网络带宽
4 核	8G	HDD，200G	100MB

2.1.3 信创环境支持

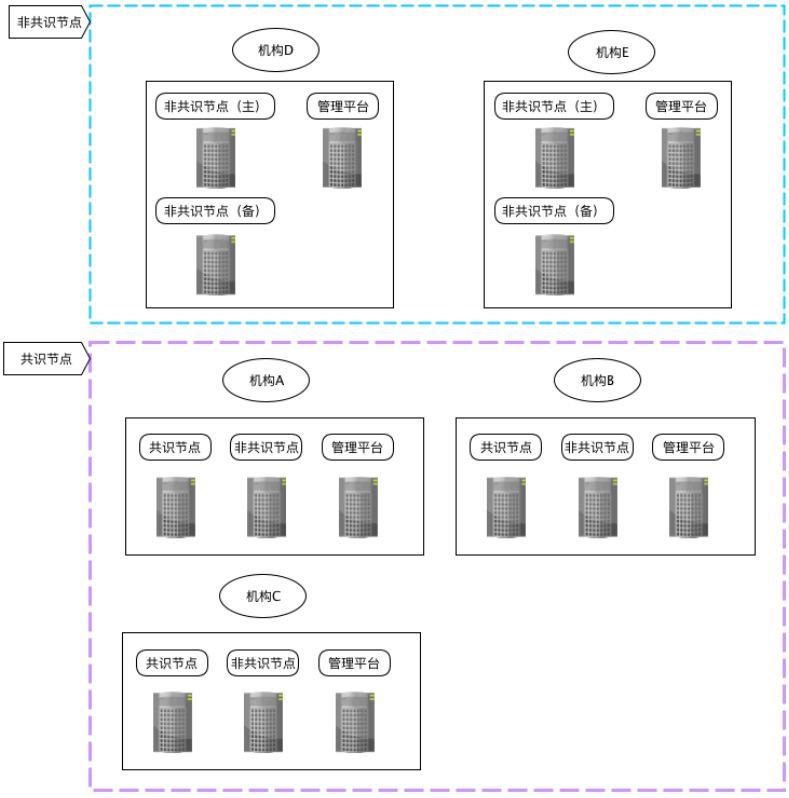
支持飞腾 FT2000+/64、腾云 S2500，鲲鹏 916、鲲鹏 920 等信创平台

2.2 网络拓扑

在开始联盟链的部署之前，需要进行网络拓扑结构的设计。主要是确定联盟链中哪些是共识节点，哪些是非共识节点。共识节点是参与共识出块的节点，非共识节点只同步数据，不参与共识协议。
另外，在设计网络拓扑时通常会考虑是否需要在一个机构内放置备用节点。联盟链网络保证了每个节点的数据都是相同的，所以单个节点的宕机对联盟链网络不会有特别的影响，放置备用节点的目的是保证机构在主节点宕机时不会影响该机构的业务流程。

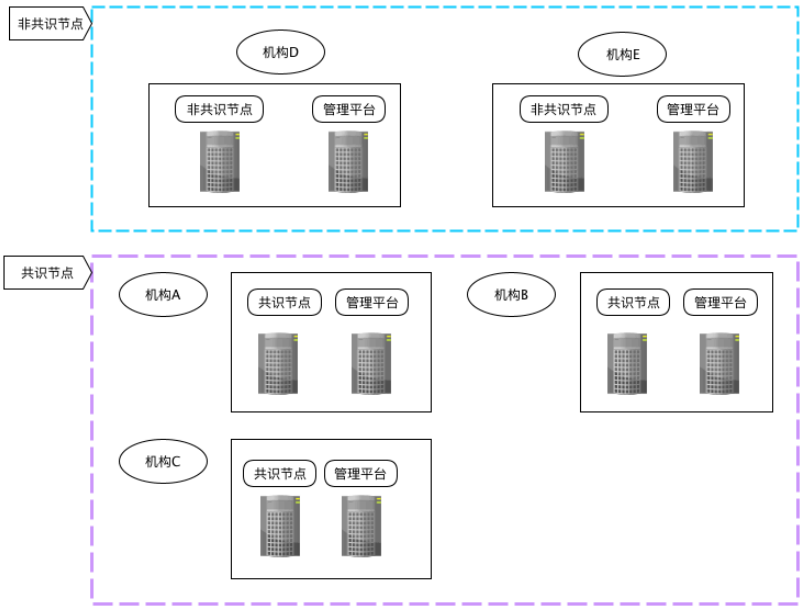
2.2.1 推荐部署

以 5 个联盟链参与机构为例，下图包含 3 个共识节点，2 个非共识节点（机构 C、机构 D 等每个机构部署两个非共识节点，互为主备）。



2.2.2 简化部署

以 5 个联盟链参与机构为例，下图包含 3 个共识节点，2 个非共识节点（机构 C、机构 D 等每个机构只部署一个非共识节点）。



2.2.3 极简部署

如果客户方只有上链的需求，而无需其他机构参与，则联盟链退化为只有一个参与机构的私有链，可采用以下极简部署方式：

2.3 安装步骤

2.3.1 准备工作

操作系统推荐采用 Ubuntu 16.04 及以后版本
Go 语言程序包 v1.17 及以后版本

联盟链的节点的部署，需要用到两个可执行程序，一个是 sipe,一个是 puppeth。这两个可执行程序可在 SimpleChain 联盟链（基础版）安装包中获得。

2.3.2 节点准备

根据网络拓扑的设计准备节点数据目录，以 3 个节点的部署为例，创建 node1，node2，node3 三个目录，并在每个目录下创建 keystore 子目录。

执行命令./sipe --keystore ./ ca new，
将生成的 ca 根证书和私钥存为 ca.crt、ca.key，保存在各个 node 的 keystore 目录下。
使用sipe account new 命令在各个节点下生成新账户

```
./sipe --keystore ./ ca new
./sipe --datadir=./node1/data --usercrtCN sipe01.dataqin.com account new
```

2.3.3 创世区块准备

使用puppeth 可执行程序来生成创世区块的 json文件。

```
命令如下,输入命令，按下回车。
./puppeth
 你将会得到如下提示
 ➜ blockchain ./puppeth
 +-----------------------------------------------------+
 | Welcome to puppeth, your Ethereum private network manager |
 | This tool lets you create a new Ethereum network down to |
 | the genesis block, bootnodes, miners and ethstats servers |
 | without the hassle that it would normally entail. |
 | Puppeth uses SSH to dial in to remote servers, and builds |
 | its network components out of Docker containers using the |
 | docker-compose toolset. |
 +-----------------------------------------------------------+
Please specify a network name to administer (no spaces, hyphens or capital letters please)
> genesis
```

这里我们输入genesis 并按下回车键，将会得到如下提示

```
Sweet, you can set this via --network=genesis next time!

INFO [11-02|02:24:03.244] Administering Ethereum network name=genesis
INFO [11-02|02:24:03.278] No remote machines to gather stats from

What would you like to do? (default = stats)
```

```
  1. Show network stats
  2. Manage existing genesis
  3. Track new remote server
  4. Deploy network components
> 2

  1. Modify existing configurations
  2. Export genesis configurations
  3. Remove genesis configuration
> 3
INFO [11-02|02:25:15.477] Genesis block destroyed

What would you like to do? (default = stats)
  1. Show network stats
  2. Configure new genesis
  3. Track new remote server
  4. Deploy network components
> 2

What would you like to do? (default = create)
  1. Create new genesis from scratch
  2. Import already existing genesis
> 1

Which consensus engine to use? (default = clique)
  1. Raft - etcd-raft
  2. Clique - proof-of-authority
  3. Pbft
> 2
How many seconds should blocks take? (default = 15)
>3
```

接下来提供上面三个节点中记录下来的三个账号地址，每输完一个账号地址就按一下回车键确认。

```
Which accounts are allowed to seal? (mandatory at least one)
> 0xef0301b443e876b3f338c583c3bc559cf2802f5f
> 0x77bde357ac483688f918c5b5514a95aa39a5ba38
> 0x99a9e566f4fc31acd7713cae932d1f4601e553df
> 0x
```

此时它提示我们输入预先分配的 token,这里我们给我们上面生成的三个地址预先分配的 token，所以我们还是输入我们生成的地址。

```
Which accounts should be pre-funded? (advisable at least one)
```

```
> 0xef0301b443e876b3f338c583c3bc559cf2802f5f
> 0x77bde357ac483688f918c5b5514a95aa39a5ba38
> 0x99a9e566f4fc31acd7713cae932d1f4601e553df
> 0x
```

此时它提示我们是否确认预先分配一些 token 给预编译地址，我们这里输入yes 并按下回车键；它提示我们设定一个 chain id，这是一个整数值，设置一个和网络中现有 SimpleChain 联盟链不同的 ID。

```
Should the precompile-addresses (0x1 .. 0xff) be pre-funded with 1 wei? (advisable yes)
> yes

Specify your chain/network ID if you want an explicit one (default = random)
> 100
INFO [11-02|02:31:19.792] Configured new genesis block
```

接着导出 genesis.jason 文件

```
 What would you like to do? (default = stats)
 1. Show network stats
 2. Manage existing genesis
 3. Track new remote server
 4. Deploy network components
> 2

 1. Modify existing configurations
 2. Export genesis configurations
 3. Remove genesis configuration
> 2

Which folder to save the genesis specs into? (default = current)
   Will create genesis.json, genesis-aleth.json, genesis-harmony.json, genesis-parity.json
>  输入回车
```

当前文档下会出现 genesis.json 和 genesis-harmony.json 文件，打开 genesis.json 文件，修改一下 gasLimit 的值，把 0x87b760 修改为 0xe087b760 。修改完成后将 genesis.json 文件拷贝到各个节点的目录下。此时目录结构为：

```
node1
├── genesis.json
└── keystore
    ├── 0xEF0301b443E876B3F338c583C3BC559cF2802f5f.crt
    ├── 0xEF0301b443E876B3F338c583C3BC559cF2802f5f.key
    ├── ca.crt
    ├── ca.key
    └──
UTC--2021-10-29T10-12-50.266632700Z--ef0301b443e876b3f338c583c3bc559cf2802f5f
```

```
node2
├── genesis.json
└── keystore
    ├── 0x77bdE357ac483688F918C5B5514A95Aa39a5ba38.crt
    ├── 0x77bdE357ac483688F918C5B5514A95Aa39a5ba38.key
    ├── ca.crt
    ├── ca.key
    └──
UTC--2021-11-02T01-41-02.406999885Z--77bde357ac483688f918c5b5514a95aa39a5ba38
node3
├── genesis.json
└── keystore
    ├── 0x99a9E566f4Fc31Acd7713cAe932d1f4601E553dF.crt
    ├── 0x99a9E566f4Fc31Acd7713cAe932d1f4601E553dF.key
    ├── ca.crt
    ├── ca.key
    └──
UTC--2021-11-02T01-41-41.421593391Z--99a9e566f4fc31acd7713cae932d1f4601e553df
```

分别进入node1、node2、node3目录下,执行命令生成创世区块

```
cd node1
./sipe --datadir=./data init genesis.json

cd node2
./sipe --datadir=./data init genesis.json

cd node3
./sipe --datadir=./data init genesis.json
```

2.3.4 启动节点

分别在 node1、node2、node3目录下，创建 password.txt文件，并把对应密码填入其中并保存。 在节点根目录下创建 start.sh文件，start.sh 的内容如下，注意修改一下账号地址。

```
export DataDir=./data     # 账本路径
export RpcPort=8545
export P2pPort=30312
export UnlockAccount=0xFff2F2521832813e5853444D973049E3786324bF   # 填入相应的账号地址，比如：这是 node1 的账号地址
export Password=./password.txt
export WsPort=8546
nohup ./sipe \
  --nodiscover \
```

```
  --datadir $DataDir \
  --gcmode archive \
  --allow-insecure-unlock \
  --rpc \
  --rpcaddr 0.0.0.0 \
  --targetgaslimit 540000000 \
  --rpcport $RpcPort \
  --rpcapi "admin,miner,db,eth,net,web3,personal,debug,txpool,permission,raft,clique" \
  --rpccorsdomain "*" \
  --port $P2pPort \
  --txpool.globalslots=51200 \
  --gasprice 0 \ --tlsenable \
  --unlock $UnlockAccount \
  --password $Password \
  --mine \
  --etherbase $UnlockAccount \
  --miner.gaslimit=540000000 \
  --syncmode full \
  --verbosity 4 \
  --miner.noempty \
  --ws \
    --wsaddr "0.0.0.0" \
    --wsport $WsPort \
    --wsapi "admin,miner,db,eth,net,web3,personal,debug,txpool,permission,raft,clique" \
    --wsorigins "*" \
    --permissioned >> app.log 2>&1 &
```

在各个节点的根目录下执行bash start.sh 启动区块链节点服务

```
bash start.sh
```

2.3.5 建立初始网络

当各个节点都已经启动以后，在各个节点根目录下，分别执行以下的命令

```
./sipe attach data/sipe.ipc
```

此时进入了链的 javascript 控制台，在此交互控制台中输入命令　admin.nodeInfo.enode 将会得到类似如下的信息

```
 Welcome to the Sipe JavaScript console!

instance: Sipe/v1.1.0-stable-4a144bf9/linux-amd64/go1.17
coinbase: 0xef0301b443e876b3f338c583c3bc559cf2802f5f
at block: 0 (Tue, 02 Nov 2021 02:25:52 UTC)
```

```
 datadir: /home/ubuntu/simpleChainLeague/test-node-1/data
 modules: admin:1.0 clique:1.0 debug:1.0 eth:1.0 miner:1.0 net:1.0 permission:1.0
personal:1.0 rpc:1.0 txpool:1.0 web3:1.0
 > admin.nodeInfo.enode
"enode://b49545fcbeb16de733afbdf2be07f714a27cb4862a23cb4b21816a90dbdecba2150a4f5
2e2b3b08afaf508aceb3a9c40b9e338aabb2c0a0b1b9538d3b533622c@127.0.0.1:30313?discp
ort=0"
```

分别在各个节点的控制台中输入 permission.addPeer("enode//xxxx",eth.account[0])连接各个节点

```
>
permission.addPeer("enode://8e3c454664c97ff4d8c32f4db8754fa081b3c2ae27674e6fc63e52
b67441c0779935d22c598a53be59b4ee831624969ea7a50c37d0ee81e86e8e95b349073854@1
27.0.0.1:30314?discport=0",eth.accounts[0])
"true"
```

接着我们输入命令 admin.peers 命令，如果返回不是[ ]则表示已经连接上了。可以使用 admin.peers.length 的输出 为 2（3 个节点两两建立连接，除了自身就是 2）再次确认是否已经连接。

```
> admin.peers
[{
      caps: ["eth/63", "eth/64"],
      enode:
"enode://b49545fcbeb16de733afbdf2be07f714a27cb4862a23cb4b21816a90dbdecba2150a4f5
2e2b3b08afaf508aceb3a9c40b9e338aabb2c0a0b1b9538d3b533622c@127.0.0.1:30313?discp
ort=0",
      id: "a2d99f546c0a07c17bc184c6ad2db5f16985b670bfe144c3e5026e4f026c3b01",
      name: "Sipe/v1.1.0-stable-4a144bf9/linux-amd64/go1.17",
      network: {
         inbound: false,
         localAddress: "127.0.0.1:39910",
         remoteAddress: "127.0.0.1:30313",
         static: true,
         trusted: false
      },
      protocols: {
         eth: {
            difficulty: 1,
            head:
"0x24078b4b0dc2f90abe460131add62b89e70ad815020b1b9b296b6a8c5eec9145",
            version: 64
         }
      }
}, {
```

```
      caps: ["eth/63", "eth/64"],
      enode:
"enode://8e3c454664c97ff4d8c32f4db8754fa081b3c2ae27674e6fc63e52b67441c0779935d22
c598a53be59b4ee831624969ea7a50c37d0ee81e86e8e95b349073854@127.0.0.1:30314?discp
ort=0",
      id: "f881d4dac97b1e8b67a5aeb4b364ce410e3cd8a6afe5b2d6d9e8000e4c81d389",
      name: "Sipe/v1.1.0-stable-4a144bf9/linux-amd64/go1.17",
      network: {
        inbound: false,
        localAddress: "127.0.0.1:58662",
        remoteAddress: "127.0.0.1:30314",
        static: true,
        trusted: false
      },
      protocols: {
        eth: {
          difficulty: 1,
          head:
"0x24078b4b0dc2f90abe460131add62b89e70ad815020b1b9b296b6a8c5eec9145",
          version: 64
        }
      }
}]
> admin.peers.length
2
```

2.3.6 部署权限管理合约

SimpleChain 联盟链的节点准入和权限管理是通过在链上部署权限管理合约来完成的。权限管理合约字节码在 SimpleChain 联盟链（基础版）安装包的 Permission.bin 文件中，使用文本编辑器可打开该文件，并拷贝字节码。

在 node1 节点上部署合约：
执行命令：eth.sendTransaction({from: eth.accounts[0],data:"合约字节码", gas: "74000000"})，data 中放入字节码
执行成功以后，返回了执行交易的哈希值

```
> eth.sendTransaction({from: eth.accounts[0],data:"字节码", gas: "74000000"})
"0xfa6afaee3aa5d97c111e2f969552b290d340c46370edffa84cfea8b4755707cc"
>
eth.getTransactionReceipt("0xfa6afaee3aa5d97c111e2f969552b290d340c46370edffa84cfea8b
4755707cc")
{
```

```
  blockHash:
"0xdc2125fd62e8dac32e16f6350cfce3662fbac6e19ec29ea17eb6d3206701163a",
  blockNumber: 1,
  contractAddress: "0x69fd129434ff44c747fb281857640ee9e1d4ec15",
  cumulativeGasUsed: 13293876,
  from: "0xef0301b443e876b3f338c583c3bc559cf2802f5f",
  gasUsed: 13293876,
  logs: [],
  logsBloom:
"0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000",
  status: "0x1",
  to: null,
  transactionHash:
"0xfa6afaee3aa5d97c111e2f969552b290d340c46370edffa84cfea8b4755707cc",
  transactionIndex: 0
}
```


2.3.7 设置合约地址

在各个节点上执行命令：permission.setContractAddress("0x69fd129434ff44c747fb281857640ee9e1d4ec15") 该命令将会设置权限合约地址，同时在节点本地生成 congfig.json 文件，下次重启时会优先加载该文件读取合约地址。若是合约地址有修改，执行该命令，config.json 文件也会随着修改。

```
> permission.setContractAddress("0x69fd129434ff44c747fb281857640ee9e1d4ec15")
"success"
```

2.3.8 初始化合约

节点 node1，进入javascript 控制台以后，执行以下命令 设置权限合约地址

```
permission.setContractAddress("0x69fd129434ff44c747fb281857640ee9e1d4ec15")
```

紧接着，将三个节点都设置为管理员

```
#将 node1 设置为管理员
```

```
>
permission.setAdminNode("enode://f19e3ae1db4d2bcd4bd5654bf27aa6e4984b7cbaf0ab6877
195af9b3c8dd2ec965fc18614f9f5c73ea6ba2a3b7b717e0c06e0f9d2d7724a76d85141e10a2fd6
c@127.0.0.1:30312?discport=0","node1","0xef0301b443e876b3f338c583c3bc559cf2802f5f",
eth.accounts[0])
#参数为 1.node1 的 enode；2.备注名；3.需要设置的管理员账户地址 4.当前节点的账户
地址
"0xf8daa156c596ba2701d1e9854ec09ad9367a9f1aa5d4987c56625f24f54bc2b1"#返回的交
易回执
```

查看节点角色

```
 #查看 node1 的角色，返回 2 表示已经成为管理员
>
permission.getNodeMap("enode://f19e3ae1db4d2bcd4bd5654bf27aa6e4984b7cbaf0ab68771
95af9b3c8dd2ec965fc18614f9f5c73ea6ba2a3b7b717e0c06e0f9d2d7724a76d85141e10a2fd6c
@127.0.0.1:30312?discport=0",eth.accounts[0])
2
```

最后，结束合约的初始化。

```
permission.initFinish(eth.accounts[0]);
```

至此为止，我们已完成了 3 个节点的基础链的部署。

3 管理平台的部署

管理平台是一个典型的 Web 应用程序, 包含前端的 Web 应用和后台 backend 程序。backend 程序连接到联盟链中的任意一个节点以读取、写入链上数据，从而执行管理功能。

3.1 数据库

目前支持 MySQL 数据库 v5.7 及以上版本。执行 SimpleChain 联盟链（基础版）安装包内的数据库脚本，将创建管理平台数据库 mp，以及所需的表结构。

3.2 backend 后台服务

配置 backend 后台服务，将 config.yaml 文件和节点证书相关文件放置 config 文件夹下，并在 config 文件夹同级创建 resources 目录并将项目中的 ip 文件夹放置在里面，将安装包内的 backend 二进制文件与 config 和 resources 放置在同级目录下。

修改 config.yaml 文件中对应的信息以连接到 MySQL 数据库和联盟链节点, 以及其它配置数据。

最终目录结构示例如下：

```
backend
├─── backend
├─── config
│    ├─── 0xf49044BF22C99c800E7b4f2fa832d65cc82A1a54.crt
│    ├─── 0xf49044BF22C99c800E7b4f2fa832d65cc82A1a54.key
│    ├─── account.json
│    ├─── ca.crt
│    ├─── config.yaml
│    ├─── passwd.txt
│                                                                    └───
UTC--2021-11-08T06-19-47.958841544Z--f49044bf22c99c800e7b4f2fa832d65cc82a1a54
└─── resources
     └─── ip
          └─── qqwry.dat
```

在后台执行 backend 程序即可通过 Web 访问管理平台界面。

4 更多信息

我们提供了快速部署基础链的方法和程序脚本，通过事先配置好的文件信息，只执行一个命令就能在几分钟内自动完成本文中第 2.2 节的所有工作。详情请参见《SimpleChain 联盟链（基础版）一键部署手册》。