

以太坊搭建联盟链

1. 环境

CentOS-7

2. Docker 安装与配置

（1）使用管理员权限

```
# su root
```

（2）CentOS 下安装 Docker 命令

```
# yum -y install docker-io
```

该命令会自动检测所需依赖并安装或更新。

（3）启动 Docker 服务

```
# service docker start
```

注：启动前需禁用 SELinux，方法如下：

修改/etc/selinux/config 文件

将 SELINUX=enforcing 改为 SELINUX=disabled

重启机器即可

（4）校验 Docker 是否安装成功

```
# docker run hello-world
```

这个命令会下载一个测试镜像，并且运行在一个容器中。当容器运行时，它会打印一些信息，并且退出。

```
应用程序 位置 终端
zc-zhai@localhost:/home/zc-zhai

文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
[zc-zhai@localhost ~]$ su root
密码:
[root@localhost zc-zhai]# service docker start
Redirecting to /bin/systemctl start docker.service
[root@localhost zc-zhai]# docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://cloud.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/engine/userguide/

[root@localhost zc-zhai]#
```

3. Bootnode 安装

(1) 下载 bootnode 镜像

docker pull docker.io/hawyasunaga/ethereum-bootnode

查看镜像: docker images

```
[root@localhost zc-zhai]# docker images
```

| REPOSITORY | TAG | IMAGE ID | CREATED | SIZE |
|---|--------|--------------|--------------|---------|
| docker.io/ethereum/client-go | latest | 6464297b2837 | 2 days ago | 42 MB |
| docker.io/hello-world | latest | f2a91732366c | 3 months ago | 1.85 kB |
| docker.io/hawyasunaga/ethereum-bootnode | latest | a046441da7cd | 5 months ago | 25 MB |

(2) Docker 创建 bootnode 容器节点

生成引导节点:

docker run -itd -m 512M --privileged=true --memory-swap -1

--net=host -p 30301:30301/udp -p 30301:30301/tcp -v

/path/docker/bootnode:/root/bootnode --name genbootnode

docker.io/hawyasunaga/ethereum-bootnode bootnode

--genkey=/root/bootnode/boot.key

运行引导节点:

```
# docker run -itd -m 512M --privileged=true --memory-swap -1
--net=host -p 30301:30301/udp -p 30301:30301/tcp -v
/path/docker/bootnode:/root/bootnode --name bootnode
docker.io/hawyasunaga/ethereum-bootnode bootnode
--nodekey=/root/bootnode/boot.key
```

注：这两个命令参数中，`-v /path/docker/bootnode:/root/bootnode` 为映射路径，在 docker 的这个 bootnode 容器中，出现容器内 `/root/bootnode` 路径都映射为外部路径 `/path/docker/bootnode`。下方以太坊容器搭建节点命令同理。

（3）查看 bootnode 日志得到节点

```
# docker logs -f bootnode
```

4. 安装以太坊节点

（1）新建创世文件 `genesis.json`

A screenshot of a text editor window titled 'genesis.json' with a subtitle '~ /admin'. The editor shows the JSON configuration for an Ethereum genesis block. The 'config' object includes 'chainId' (90), 'homesteadBlock' (0), and 'eip155Block' (0). The 'alloc' object contains a single entry for '0xf38056f45091ee992298e53681b0a60c999ff95' with a 'balance' of '0xff7'. Other fields include 'coinbase' (0x00000000000000000000000000000000), 'difficulty' (0x02000000), 'extraData' (empty), 'gasLimit' (0x2efd8), 'nonce' (0x0000000000000042), 'mixhash' (0x00), 'parentHash' (0x00), 'timestamp' (0x00), 'gasLimit' (0xffffffff), and 'difficulty' (0x20000).

```
{
  "config": {
    "chainId": 90,
    "homesteadBlock": 0,
    "eip155Block": 0,
    "eip158Block": 0
  },
  "alloc": {
    "0xf38056f45091ee992298e53681b0a60c999ff95": {
      "balance": "0xffffffffffffffffffffffffffffffffffffffff7"
    }
  },
  "coinbase": "0x00000000000000000000000000000000",
  "difficulty": "0x02000000",
  "extraData": "",
  "gasLimit": "0x2efd8",
  "nonce": "0x0000000000000042",
  "mixhash": "0x0000000000000000000000000000000000000000000000000000000000000000",
  "parentHash": "0x0000000000000000000000000000000000000000000000000000000000000000",
  "timestamp": "0x00",
  "gasLimit": "0xffffffff",
  "difficulty": "0x20000"
}
```

(2) 初始化创世文件

首先新建一个准备放置以太坊目录的文件，将 `genesis.json` 文件放入该地址，本文为 `/home/admin` 下。

运行如下命令

```
# docker run -itd --privileged=true -v /home/admin:/root/ethdev --name
gethDev1 ethereum/client-go --datadir /root/ethdev --networkid
8765639736937780 init /root/ethdev/genesis.json
```

请记住 `--networkid 8765639736937780`，这是当前搭建联盟链的 ID。

初始化完成。

(3) Docker 创建以太坊容器节点

运行如下命令

```
# docker rm -f gethDev1
```

```
# docker run -itd -m 512M --privileged=true --network=host
--memory-swap -1 --net=host -p 8545:8545 -p 40303:40303 -v
/home/admin:/root/ethdev --name gethDev1 ethereum/client-go
--ipcdisable --port 40303 --bootnodes
"enode://ad6aff917c6e8bd40cb20af4eac6ce05c16d285125b46f17fc9b5
c3b0a833bd21667231215949c6ff771ba512eb8f87f138ac6679852997c3
eaec1d349561d20@120.25.162.110:30301" --bootnodesv4
"enode://ad6aff917c6e8bd40cb20af4eac6ce05c16d285125b46f17fc9b5
c3b0a833bd21667231215949c6ff771ba512eb8f87f138ac6679852997c3
eaec1d349561d20@120.25.162.110:30301" --bootnodesv5
"enode://ad6aff917c6e8bd40cb20af4eac6ce05c16d285125b46f17fc9b5
c3b0a833bd21667231215949c6ff771ba512eb8f87f138ac6679852997c3
eaec1d349561d20@120.25.162.110:30301" --debug --rpcapi
"db,eth,net,web3,personal,admin,miner,txpool" --datadir /root/ethdev
--networkid 8765639736937780 --wsapi
"db,eth,net,web3,personal,admin,miner,txpool" --ws --wsaddr "0.0.0.0"
--rpc --rpcaddr "0.0.0.0" --cache=512 --verbosity 3 console
```

//enode 为上面运行 bootnode 得到的节点，并将预备的私钥文件放入 keystore 中，并且--networkid 8765639736937780 为联盟链的标识 ID。

查看控制台日志： docker logs -f gethDev1

```

d85b7110a87eec18c1552500f38c775fd319a27e60fcbcd1dc64d1ccb45fb0a19
[root@localhost zc-zhai]# docker logs -f gethDev1
INFO [03-19|08:16:19] cmd/utlils/flags.go:835] Maximum peer count ETH=25 LES=0 total=25
INFO [03-19|08:16:19] node/node.go:166] Starting peer-to-peer node instance=Geth/v1.8.3-unstable-fe6cf00f/linux-amd64/gol
.10
INFO [03-19|08:16:19] ethdb/database.go:63] Allocated cache and file handles database=/root/ethdev/geth/chaindata cache=84 handles
=024
INFO [03-19|08:16:19] eth/backend.go:120] Initialised chain configuration config={ChainID: 1 Homestead: 1150000 DAO: 1920000 DA
OSupport: true EIP150: 2463000 EIP155: 2675000 EIP158: 2675000 Byzantium: 4370000 Constantinople: <nil> Engine: ethash}
INFO [03-19|08:16:19] consensus/ethash/ethash.go:421] Disk storage enabled for ethash caches dir=/root/ethdev/geth/ethash count=3
INFO [03-19|08:16:19] consensus/ethash/ethash.go:424] Disk storage enabled for ethash DAGs dir=/root/.ethash count=2
INFO [03-19|08:16:19] eth/backend.go:138] Initialising Ethereum protocol versions=[63 62] * network=8765639736937780
INFO [03-19|08:16:19] core/blockchain.go:253] Loaded most recent local header number=0 hash=4e567-cb8fa3 td=17179869184
INFO [03-19|08:16:19] core/blockchain.go:254] Loaded most recent local full block number=0 hash=4e567-cb8fa3 td=17179869184
INFO [03-19|08:16:19] core/blockchain.go:255] Loaded most recent local fast block number=0 hash=4e567-cb8fa3 td=17179869184
INFO [03-19|08:16:19] core/tx_journal.go:97] Loaded local transaction journal transactions=0 dropped=0
INFO [03-19|08:16:19] core/tx_journal.go:149] Regenerated local transaction journal transactions=0 accounts=0
INFO [03-19|08:16:19] p2p/server.go:395] Starting P2P networking
INFO [03-19|08:16:22] p2p/discover/udp.go:238] UDP listener up self=enode://02fc0d88688669da97ab5a546922e7adb
fdbcb9d385be6027fcb306b2f5ce62c9338a59783e784ef6a464dc84233c5c146952fdfcc5e31c90aea9553a282cd25@:::40303
INFO [03-19|08:16:22] node/node.go:397] HTTP endpoint opened url=http://0.0.0.0:8545 cors=vhosts=localhost
INFO [03-19|08:16:22] node/node.go:450] WebSocket endpoint opened url=ws://:::8546
INFO [03-19|08:16:22] p2p/server.go:742] RLPx listener up self=enode://02fc0d88688669da97ab5a546922e7adb
fdbcb9d385be6027fcb306b2f5ce62c9338a59783e784ef6a464dc84233c5c146952fdfcc5e31c90aea9553a282cd25@:::40303
Welcome to the Geth JavaScript console!

Instance: Geth/v1.8.3-unstable-fe6cf00f/linux-amd64/gol.10
modules: admin:1.0 debug:1.0 eth:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 txpool:1.0 web3:1.0

```

节点搭建成功。

5. 验证联盟链的连接

进入以太坊容器中：

docker attach gethDev1

输入命令：admin

查看 peer 是否连接：

```

admin
{
  datadir: "/root/ethdev",
  nodeInfo: {
    enode: "enode://02fc0d88688669da97ab5a546922e7adbfdbcb9d385be6027fcb306b2f5ce62c9338a59783e784ef6a464dc84233c5c146952fdfcc5e31c90aea9553a282cd25@:::40303",
    id: "02fc0d88688669da97ab5a546922e7adbfdbcb9d385be6027fcb306b2f5ce62c9338a59783e784ef6a464dc84233c5c146952fdfcc5e31c90aea9553a282cd25",
    ip: "::",
    listenAddr: "[::]:40303",
    name: "Geth/v1.8.3-unstable-fe6cf00f/linux-amd64/gol.10",
    ports: {
      discovery: 40303,
      listener: 40303
    },
    protocols: {
      eth: {
        config: {...},
        difficulty: 17179869184,
        genesis: "0xd4e56740f876ae8fc010b8e40d5f56745a18d0906a34e69aec8c0db1cb8fa3",
        head: "0xd4e56740f876ae8fc010b8e40d5f56745a18d0906a34e69aec8c0db1cb8fa3",
        network: 8765639736937780
      }
    }
  },
  peers: [],
  addPeer: function(),
  clearHistory: function(),
  exportChain: function(),
  getDatadir: function(callback),
  getNodeInfo: function(callback),
  getPeers: function(callback),
  importChain: function(),
  removePeer: function(),
  sleep: function github.com/ethereum/go-ethereum/console.(*bridge).Sleep-fm(),
  sleepBlocks: function github.com/ethereum/go-ethereum/console.(*bridge).SleepBlocks-fm(),
}

```

连接完成，联盟链搭建成功。

下一步工作

在测试网络中开发和部署智能合约