# 私链节点

私链节点包括2个部分：以太坊私链节点、伴生程序

## 以太坊私链节点

| 名称 | 说明 | 备注 |
| --- | --- | --- |
| 安装平台 | centos7-64 | |
| 程序包 | 以太坊geth，linux-X64可执行包 | |
| | | |

## 1，私链的搭建

**文件结构：**

```
#默认安装路径/opt/ether-data
/opt/ether-data/
|-- bin
|    |-- attach-geth.sh
|    |-- geth
|    |-- bootnode
|    |-- puppeth
|    |-- init-geth.sh
|    |-- start-geth.sh
|    `-- stop-geth.sh
|-- config.toml
|-- genesis.json
|-- log
|-- keystore
`-- passwd
#使用方法
cd /opt/ether-data/bin
#初始化
sh init-geth.sh
#导出配置文件
sh dump-geth.sh
#启动
sh start-geth.sh
```

```
#停止
sh stop-geth.sh
#连接
sh ttach-geth.sh
```

**私链POA以太坊节点搭建**

**a，下载geth程序包[1.8.7]，并创建账户**

下载链接地址：[https://ethereum.github.io/go-ethereum/downloads/](https://ethereum.github.io/go-ethereum/downloads/)，并解压安装包到/opt/ether-data/bin文件夹下

```
mkdir -p /opt/ether-data/bin
#解压安装包到/opt/ether-data/bin文件夹下
```

账号创建(>=1个记账账户[设置为挖矿地址]+>=1个普通资金账户)

| 名称 | 数量 | 其他 |
| --- | --- | --- |
| 记账账户【私链节点挖矿账户】 | >=3 | 开发的时候可以设置为1，简易版方便开发 |
| 初始普通账户【资金比较充裕】 | >=1 | |
| | | |

```
#记录密码到文件password
echo "box123456" > /opt/ether-data/passwd
#密码可以设置相同，重复下边指令完成4个账户创建
$/opt/ether-data/bin/geth --datadir /opt/ether-data/ account new --password /opt/ether-data/passwd
...
Address: {f051dd9e9d409edd0c56440f3b728ec9806bec9d}
#记录下账户的地址到文件accounts,后边要用，Address: {f051dd9e9d409edd0c56440f3b728ec9806bec9d}
echo "f051dd9e9d409edd0c56440f3b728ec9806bec9d" >> account
#node 自定义：
$/opt/ether-data/bin/bootnode -genkey /opt/ether-data/node-key
```

**b，创建genesis.json文件，文件存放到/opt/ether-data/目录下**

使用puppeth生成genesis.json文件

```
$/opt/ether-data/bin/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, please)
> genesis

Sweet, you can set this via --network=boxethprv next time!
INFO [05-18|13:04:58] Administering Ethereum network name=boxethprv
WARN [05-18|13:04:58] No previous configurations found path=/Users/john/.puppeth/boxethprv

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

Which consensus engine to use? (default = clique)
 1. Ethash - proof-of-work
 2. Clique - proof-of-authority
> 2

How many seconds should blocks take? (default = 15)
> 5

Which accounts are allowed to seal? (mandatory at least one)
> 0x7dbab39da083e927fd5dbe433634abf28fb8dbf5
> 0x8da53d17a192bd1a8cd505df64b739e62fe66aad
> 0xf9e04155a092ba9788ea34d4c4c91467cf2f71d1
> 0x

Which accounts should be pre-funded? (advisable at least one)
> 0x7dbab39da083e927fd5dbe433634abf28fb8dbf5
> 0x8da53d17a192bd1a8cd505df64b739e62fe66aad
> 0xf9e04155a092ba9788ea34d4c4c91467cf2f71d1
> 0x[资金账户]


Specify your chain/network ID if you want an explicit one (default = random)
```

```
> 20180518

Anything fun to embed into the genesis block? (max 32 bytes)
>

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 fork rules
 2. Export genesis configuration
> 2

Which file to save the genesis into? (default = genesis.json)
> genesis.json
INFO [05-18|13:09:14] Exported existing genesis block

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
> ^C
```

### c，初始化私链节点

初始化私链

```
$/opt/ether-data/init-geth.sh
```

/opt/ether-data/init-geth.sh脚本内容

```
#!/bin/bash
DATADIR=/opt/ether-data
GETH=${DATADIR}/bin/geth
GENESISFILE=${DATADIR}/genesis.json


echo "Warning: run this script will delete all exists block data!"
read -p "Do you want do this job?(yes/no default no): " answer

answer=${answer:-no}

if [ $answer = "yes" ];then
  rm -rf ${DATADIR}/geth
```

```
  # init genesis block
  ${GETH} --datadir ${DATADIR} init ${GENESISFILE}
else
  echo "Nothing to happend."
fi
```

导出私链配置文件

```
$/opt/ether-data/dump-geth.sh
```

/opt/ether-data/dump-geth.sh脚本内容

```
#!/bin/bash
DATADIR=/opt/ether-data
GETH=${DATADIR}/bin/geth
GENESISFILE=${DATADIR}/genesis.json
CONFIGFILE=${DATADIR}/config.toml
IPCFILE=${DATADIR}/geth.ipc

LOCALIP="localhost"
NETWORKID=`grep -i chainId ${GENESISFILE} | grep -o "[0-9]*[0-9]"`

${GETH} --datadir "${DATADIR}" --ethash.dagdir ${DATADIR} --ipcpath "${IPCFILE}"\
    --syncmode 'full' \
    --networkid ${NETWORKID} --nodiscover  --gasprice "1" \
    --rpc  --rpcaddr ${LOCALIP}    \
    --ws --wsaddr ${LOCALIP}  --wsorigins "*" \
    --cache 256 --minerthreads 1 dumpconfig > ${CONFIGFILE}
```

启动节点

```
$/opt/ether-data/start-geth.sh
```

/opt/ether-data/start-geth.sh脚本内容

```
#!/bin/bash

DATADIR=/opt/ether-data
GETH=${DATADIR}/bin/geth
LOGDIR=${DATADIR}/log
ACC="0x`cat ${DATADIR}/account`"
ACCPASS=${DATADIR}/passwd
NODEKEY=`cat ${DATADIR}/node-key`

mkdir -p "${LOGDIR}"
nohup ${GETH} --config ${DATADIR}/config.toml --nodekeyhex ${NODEKEY} --etherbase ${ACC} --unlock ${ACC} --password ${ACCPASS} --mine  > ${LOGDIR}/log_geth_start.log 2>&1 &
```

```
#添加节点后查看/opt/ether-data/log下的日志，看到有（number=112）字样，并且数字在不断更新，则说明节点已经正常连接并已经开始工作
#或者[bash /opt/ether-data/attach-geth.sh],执行eth.blockNumber,查看区块高度，如果区块高度有更新则说明节点连接正常。

#4，添加StaticNodes,修改配置文件config.toml,添加3个节点的StaticNodes=[]，避免每次启动后


```

停止节点

```
$/opt/ether-data/stop-geth.sh
```

停止脚脚本/opt/ether-data/stop-geth.sh

```
#!/bin/bash

function query_pid() {
  echo `ps aux|grep "[g]eth --config"|awk '{print $2}'`
}

PID=`query_pid`
if [ ! -z "${PID}" ]; then
  kill -9 ${PID}
  echo "The geth process stopped. PID: ${PID}"
fi
```

## 2，发布合约【参考】

**目标：**

1，oracle合约的发布

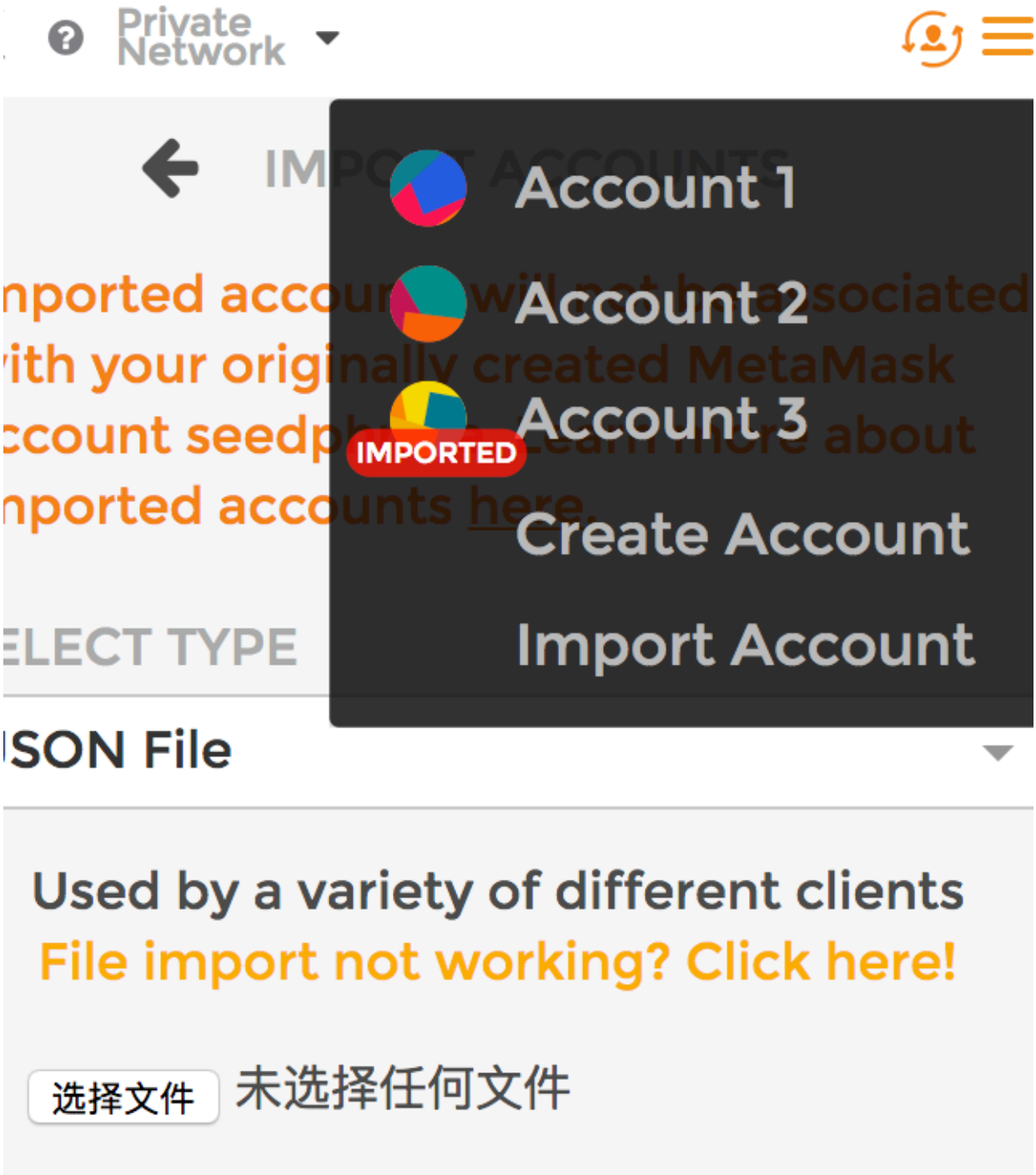
2，signer账户添加

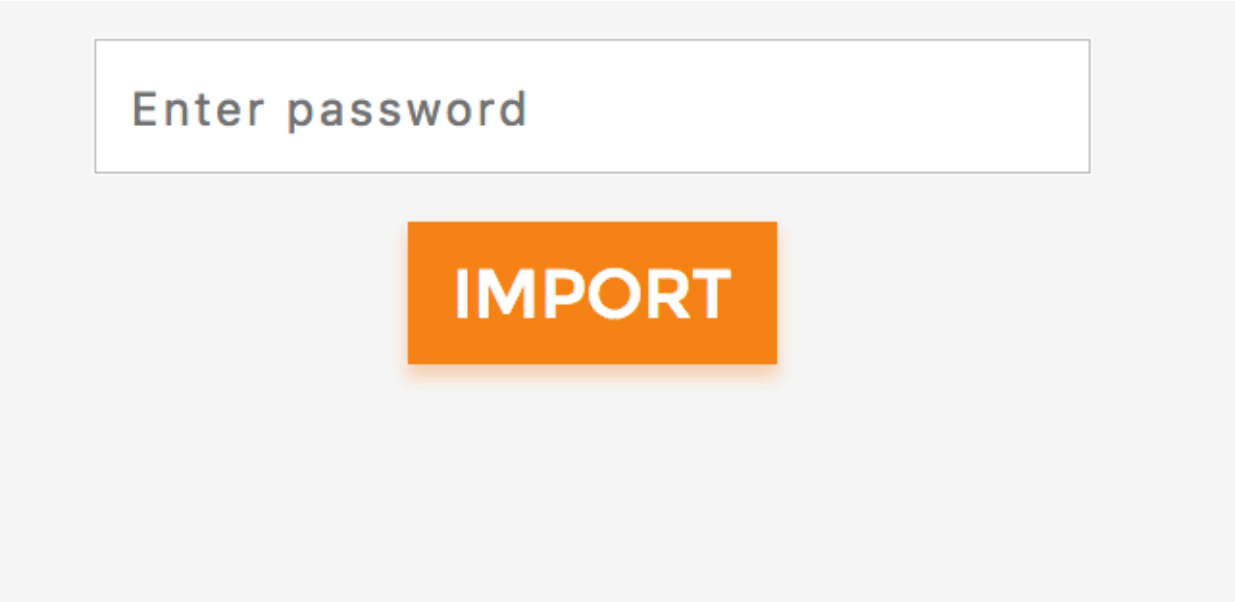
3，sink合约的发布

**准备：**

工具：浏览器插件(chrome,firefox)metamask，[https://metamask.io/](https://metamask.io/)

**a，准备一个 一个发布oracle合约的账户，可以导入上一步的资金账户或者在私链节点上新创建的账户，或者使用metamask插件产生一个新的账户地址【注意保存账户的密码】**






**b，在私链上给账户充值（如果有资金则跳过此步骤）参考指令：**

```
#解锁用户如果需要
personal.unlockAccount(eth.accounts[0],"密码",10);
#向指定用户充入1ETH
eth.sendTransaction({from: eth.accounts[0] , to: "0xa079cb32e31e23c32c5412295fba3373ad43323c", value: web3.toWei(1)})
#查看到账情况
web3.fromWei(eth.getBalance("0xa079cb32e31e23c32c5412295fba3373ad43323c"),'ether')
```

充值到账后可在插件中看到对应的金额。

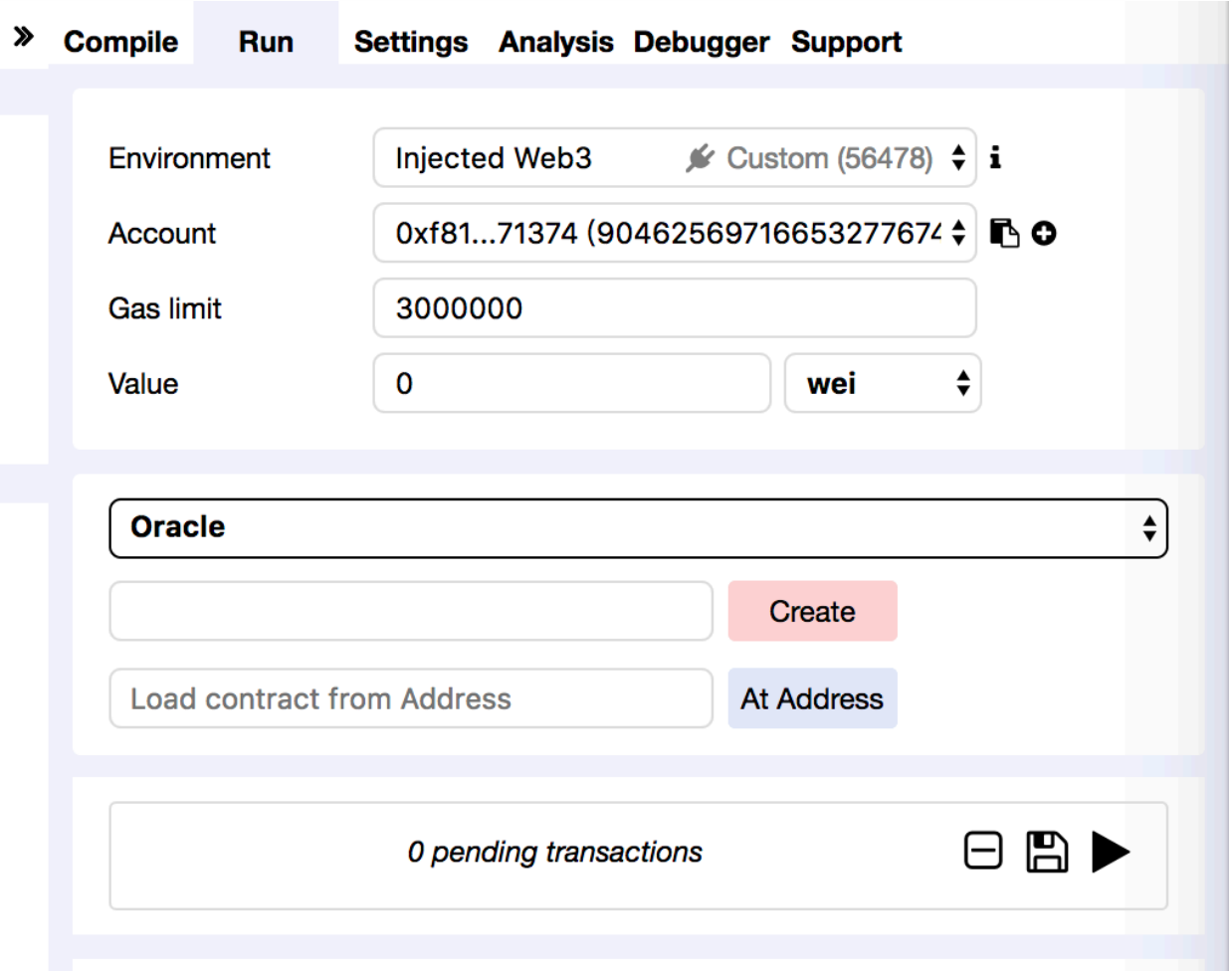
**c，发布合约**

https://ethereum.github.io/browser-solidity/

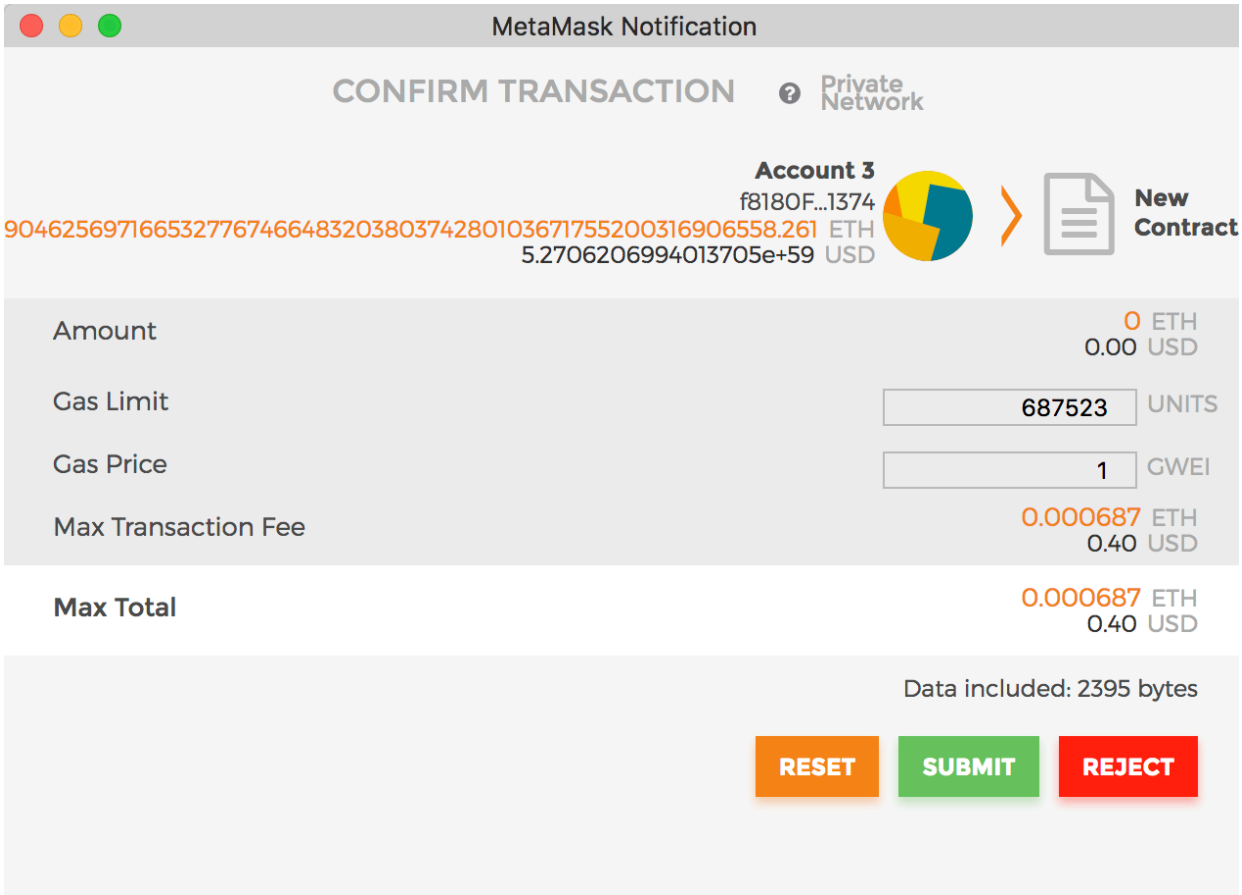
http://remix.ethereum.org/#optimize=false&version=soljson-v0.4.24+commit.e67f0147.js

导入合约文件【oracle.sol,sink.sol】，并编译合约

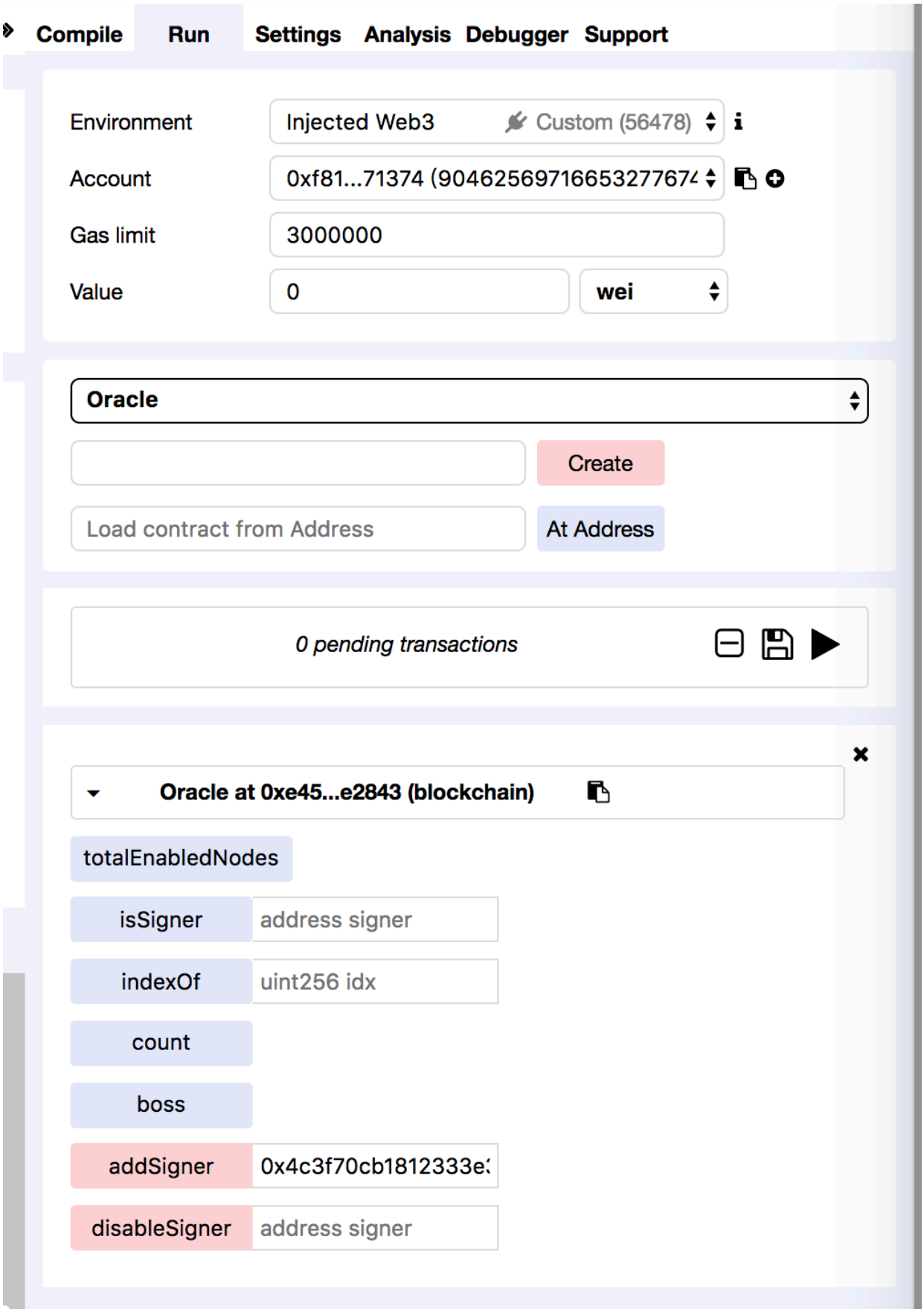

切换侧边页，进入【RUN】页面，选择“oracle”点解“Create”进行合约发布




点击“Create”后会弹出提交窗口，点击“SUBMIT”完成合约的发布。



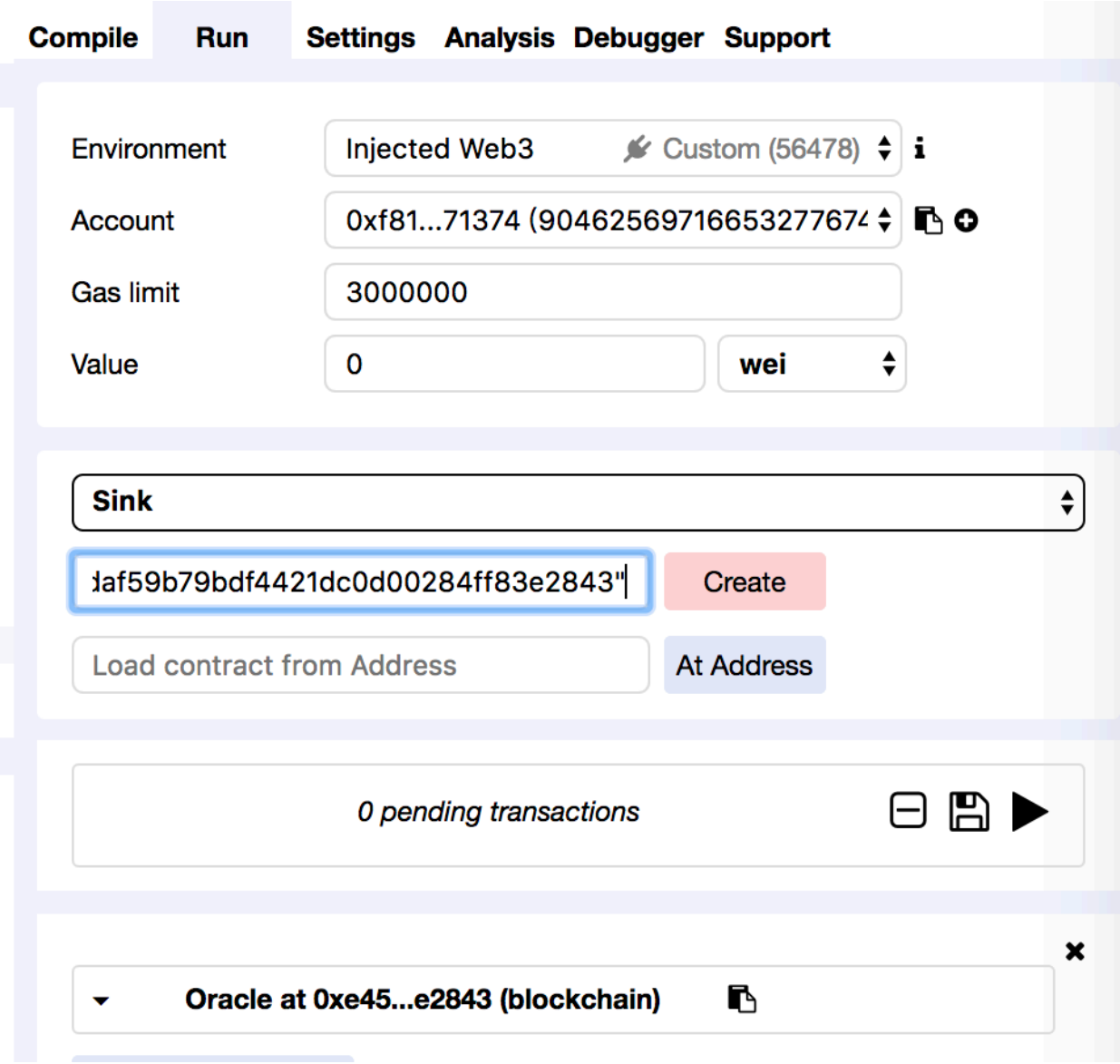

等待私链确认后会在侧边出现关于发布ORACLE合约信息，以及操作接口。




记录下oracle的合约地址，并使用“addSigner”接口添加投票者，此处使用添加3个有记账权的账户，点击“addSigner”后也会弹出窗口需要用户来确认提交，点击“SUBMIT” 即可。依次添加3个有记账权的账户完成此步骤

**【注】oracle合约中signer添加后的验证，使用“indexOf”方法进行验证**

sink合约发布：复制上一步骤获取的oracle合约地址“0xe457ccddaf59b79bdf4421dc0d00284ff83e2843”，切换合约为Sink，在“Create”栏写入oracle的合约地址【记住要加上双引号】，然后点击“Create”完成合约发布。




sink合约发布后会在侧边栏发现sink的合约地址“0x259bec50f95ac5fdee20254c4b2b1ce81ddbaa98”，记录下来

Compile | Run | Settings | Analysis | Debugger | Support

| | |
|---|---|
| Environment | Injected Web3 — Custom (56478) |
| Account | 0xf81...71374 (9046256971665327767 |
| Gas limit | 3000000 |
| Value | 0 wei |

**Sink**

"0xe457ccddaf59b79bdf4421dc0d00284 — Create

Load contract from Address — At Address

*0 pending transactions*

**Oracle at 0xe45...e2843 (blockchain)**

**Sink at 0x259...baa98 (blockchain)**

| | |
|---|---|
| txExists | bytes32 hash, bytes32 |
| available | bytes32 hash |
| approve | bytes32 txHash, uint25 |
| addHash | bytes32 hash |
| enable | bytes32 hash |
| disable | bytes32 hash |
| changeOracle | address newOracle |

# 伴生程序

| 名称 | 说明 | 备注 |
| --- | --- | --- |
| 安装平台 | centos7-64 | |
| 编译环境 | golang,git | 如需要进行源码编译 |
| | | |

文件结构

```
companion/
├── companion                #伴生程序可执行文件
├── config.json              #伴生程序配置文件
├── cursor.txt               #私链节点块高度检索记录文件
├── leveldb                  #伴生程序db文件
├── certs                    #代理RPC证书
|   ├── client.key
|   └── client.pem
├── log                      #伴生程序日志文件
└── log.xml                  #伴生程序日志配置文件
```

配置文件：

/opt/box/companion/config.json

```
{
  "pri_eth": {
    "creator":"0x1db6dec5731130d6b5d2e6789194e1391ca05754",
    "creator_passphrase": "box123456",
    "creator_keystore_path": "/opt/ether-data/keystore/UTC--2017-12-07T05-50-56.854752329Z--1db6dec5731130d6b5d2e6789194e1391ca05754",
    "geth_api": "ws://localhost:8546",
    "check_block_before": 0,
    "cursor_file_path":"/opt/box/companion/cursor.txt",
    "nonce_file_path":"/opt/box/companion/nonce.txt",
    "scan_interval": 5,
    "gas_price":2,
    "wallet_gas": 291654,
    "factory_gas": 513617
  },
  "router_info":{
```

```
    "ser_voucher":"voucher",
    "ser_companion":"companion",
    "companion_name":"comp-001"
  },
  "level_db_path":"/opt/box/companion/leveldb",
  "sink_address":"0x78202ee297826b2d3798db4300340d69893746f5",
  "client_cert":"/opt/box/companion/certs/client.pem",
  "client_key":"/opt/box/companion/certs/client.key",
  "grpc_ser_host":"192.168.199.165:50502"
}
```

log.xml

主要更改如下两个地方，详细的使用方法参考[log4go]

```
....
<level>DEBUG</level>
.....
<property name="filename">log/agent.log</property>
...
```

启动：

```
cd /opt/box/companion
./companion start
#后台启动
nohup ./companion start &
```