

← 返回作业列表

发布于：2019-09-29 19:00

随堂测试

截止日期：2019-10-11 00:00

作业内容：
1、作业内容：
1. 以太坊的安装、私有链创世区块搭建、私有链节点的加入
2. 对 getBlock 中所得区块的各个字段进行解释
3. 对日志输出进行解释
4. 编写简单的智能合约，在 remix 下进行调试，并部署在链上进行调用
5. 对交易的字段进行解释

2、作业要求：
完成作业内容并写成实验报告，部署、搭建、合约调用等过程需要截图证明。
截止提交时间10月10日24:00。

我的提交(点击下载):

最新学员



以太坊有很多版本，这里我选择了 geth 进行了安装：

```
Python3 (666) [Geth]geth.exe
INFO [10-10 06:53:36.695] Bumping default cache on mainnet
WARN [10-10 06:53:36.699] Sanitizing cache to Go's GC limits
INFO [10-10 06:53:36.706] Maximum peer count
INFO [10-10 06:53:36.891] Starting peer-to-peer node
INFO [10-10 06:53:36.896] Allocated trie memory caches
INFO [10-10 06:53:36.899] Allocated cache and file handles
INFO [10-10 06:53:37.240] Opened ancient database
INFO [10-10 06:53:37.257] Writing default main-net genesis block
INFO [10-10 06:53:37.539] Persisted trie from memory database
INFO [10-10 06:53:37.545] Initialised chain configuration
tium: 4370000 Constantinople: 7280000 Petersburg: 7280000 Istanbul:
INFO [10-10 06:53:37.554] Disk storage enabled for ethash caches
INFO [10-10 06:53:37.558] Disk storage enabled for ethash DAGs
INFO [10-10 06:53:37.562] Initialising Ethereum protocol
WARN [10-10 06:53:37.563] Upgrade blockchain database version
INFO [10-10 06:53:37.604] Loaded most recent local header
INFO [10-10 06:53:37.608] Loaded most recent local full block
INFO [10-10 06:53:37.612] Loaded most recent local fast block
INFO [10-10 06:53:37.613] Regenerated local transaction journal
INFO [10-10 06:53:37.682] Allocated fast sync bloom
INFO [10-10 06:53:37.725] Initialized fast sync bloom
INFO [10-10 06:53:37.806] New local node record
INFO [10-10 06:53:37.821] Started P2P networking
37f4402585f48778528127.0.0.1:30303
INFO [10-10 06:53:37.824] IPC endpoint opened

provided=1024 updated=4096
provided=4096 updated=4096
ETH=50 LBS=0 total=50
instance=Geth/v1.9.6-stable-bd059680/windows-amd64/go1.13
clean=1015.00MiB dirty=1015.00MiB
database=C:\Users\Ailisa\AppData\Local\Ethereum\geth\chaindata cache=1.98GiB handles=8192
database=C:\Users\Ailisa\AppData\Local\Ethereum\geth\chaindata\ancient
nodes=12356 size=1.79MiB time=49.8666ms gnodes=0 gcsize=0.00B gctime=0s livenodes=1 liveness=0.00B
config={ChainID: 1 Homestead: 1150000 DAO: 1920000 DAOsupport: true EIP150: 2463000 EIP155: 2675000 EIP158: 2675000 Byzantium:
from=<nil> to=7
number=0 hash=d4e567...cb8fa3 td=17179869184 age=50y5mo4w
number=0 hash=d4e567...cb8fa3 td=17179869184 age=50y5mo4w
number=0 hash=d4e567...cb8fa3 td=17179869184 age=50y5mo4w
transaction=0 account=0
size=1.98GiB
items=12356 errorrate=0.000 elapsed=69.514ms
seq=1 id=c2901146ea787758 ip=127.0.0.1 udp=30303 tcp=30303
self=enode://c72396823ed6b85eac34b079f1dfe3c56488967614709030251097847591bb4772c63d58d686ad84fcd1dcfa6e7d77bdc6b3e6af2f3a
url=\\\\.\\pipe\\geth.ipc
```



命令行验证：

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.18362.388]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Ailsa>git --version
git version 2.23.0.windows.1

C:\Users\Ailsa>geth --help
NAME:
  geth - the go-ethereum command line interface

  Copyright 2013-2019 The go-ethereum Authors

USAGE:
  geth [options] command [command options] [arguments...]

VERSION:
  1.9.6-stable-bd059680

COMMANDS:
  account          Manage accounts
  attach           Start an interactive JavaScript environment (connect to node)
  console         Start an interactive JavaScript environment
  copydb          Create a local chain from a target chaindata folder
  dump            Dump a specific block from storage
  dumpconfig      Show configuration values
  export          Export blockchain into file
  export-preimages Export the preimage database into an RLP stream
  import          Import a blockchain file
  import-preimages Import the preimage database from an RLP stream
  init           Bootstrap and initialize a new genesis block
```

安装位置: `--datadir value` Data directory
for the databases and keystore (default:
"C:\\Users\\Ailsa\\AppData\\Local\\Ethereum")

然后配置环境:

```
Select Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\WINDOWS\system32> set-ExecutionPolicy RemoteSigned

Execution Policy Change
The execution policy helps protect you from scripts that you do not trust. Changing the
execution policy might expose you to the security risks described in the
about_Execution_Policies help topic at https://go.microsoft.com/fwlink/?LinkID=135170. Do
you want to change the execution policy?
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "N"): Y
PS C:\WINDOWS\system32> iwr https://chocolatey.org/install.ps1 -UseBasicParsing | iex
Getting latest version of the Chocolatey package for download.
Getting Chocolatey from https://chocolatey.org/api/v2/package/chocolatey/0.10.15.
Extracting C:\Users\Ailsa\AppData\Local\Temp\chocolatey\chocInstall\chocolatey.zip to C:\U
sers\Ailsa\AppData\Local\Temp\chocolatey\chocInstall...
Installing chocolatey on this machine
Creating ChocolateyInstall as an environment variable (targeting 'Machine')
Setting ChocolateyInstall to 'C:\ProgramData\chocolatey'
WARNING: It's very likely you will need to close and reopen your shell
before you can use choco.
Restricting write permissions to Administrators
We are setting up the Chocolatey package repository.
The packages themselves go to 'C:\ProgramData\chocolatey\lib'
(i.e. C:\ProgramData\chocolatey\lib\yourPackageName).
A shim file for the command line goes to 'C:\ProgramData\chocolatey\bin'
and points to an executable in 'C:\ProgramData\chocolatey\lib\yourPackageName'.

Creating Chocolatey folders if they do not already exist.

WARNING: You can safely ignore errors related to missing log files when
upgrading from a version of Chocolatey less than 0.9.9.
'Batch file could not be found' is also safe to ignore.
'The system cannot find the file specified' - also safe.
WARNING: Not setting tab completion: Profile file does not exist at
'C:\Users\Ailsa\Documents\WindowsPowerShell\Microsoft.PowerShell_profile.ps1'.
Chocolatey (choco.exe) is now ready.
You can call choco from anywhere, command line or powershell by typing choco.
Run choco /? for a list of functions.
You may need to shut down and restart powershell and/or consoles
first prior to using choco.
Ensuring chocolatey commands are on the path
Ensuring chocolatey.nupkg is in the lib folder
PS C:\WINDOWS\system32> choco install git
Chocolatey v0.10.15
Installing the following packages:
git
By installing you accept licenses for the packages.
```

```
Select Administrator: Windows PowerShell

Progress: Downloading chocolatey-core.extension 1.3.3... 100%
Progress: Downloading git 2.23.0... 100%

chocolatey-core.extension v1.3.3 [Approved]
chocolatey-core.extension package files install completed. Performing other installation steps.
  Installed/updated chocolatey-core extensions.
  The install of chocolatey-core.extension was successful.
  Software installed to 'C:\ProgramData\chocolatey\extensions\chocolatey-core'

git.install v2.23.0 [Approved]
git.install package files install completed. Performing other installation steps.
The package git.install wants to run 'chocolateyInstall.ps1'.
Note: If you don't run this script, the installation will fail.
Note: To confirm automatically next time, use '-y' or consider:
choco feature enable -n allowGlobalConfirmation
Do you want to run the script?([Y]es/[A]ll - yes to all/[N]o/[P]rint): Y

Using Git LFS
Installing 64-bit git.install...
git.install has been installed.
git.install installed to 'C:\Program Files\Git'
  git.install can be automatically uninstalled.
Environment Vars (like PATH) have changed. Close/reopen your shell to
see the changes (or in powershell/cmd.exe just type refreshenv).
The install of git.install was successful.
  Software installed to 'C:\Program Files\Git\'

git v2.23.0 [Approved]
git package files install completed. Performing other installation steps.
The install of git was successful.
  Software install location not explicitly set, could be in package or
  default install location if installer.

Chocolatey installed 3/3 packages.
  See the log for details (C:\ProgramData\chocolatey\logs\chocolatey.log).
PS C:\WINDOWS\system32> choco install golang
Chocolatey v0.10.15
Installing the following packages:
golang
By installing you accept licenses for the packages.
Progress: Downloading golang 1.13.1... 100%

golang v1.13.1 [Approved]
golang package files install completed. Performing other installation steps.
The package golang wants to run 'chocolateyinstall.ps1'.
Note: If you don't run this script, the installation will fail.
Note: To confirm automatically next time, use '-y' or consider:
choco feature enable -n allowGlobalConfirmation
Do you want to run the script?([Y]es/[A]ll - yes to all/[N]o/[P]rint): Y
```

```
Select Administrator: Windows PowerShell

choco feature enable -n allowGlobalConfirmation
Do you want to run the script?([Y]es/[A]ll - yes to all/[N]o/[P]rint): Y

Downloading golang 64 bit
  from 'https://dl.google.com/go/gol.13.1.windows-amd64.msi'
Progress: 100% - Completed download of C:\Users\Ailsa\AppData\Local\Temp\chocolatey\golang
\1.13.1\gol.13.1.windows-amd64.msi (111.7 MB).
Download of gol.13.1.windows-amd64.msi (111.7 MB) completed.
Hashes match.
Installing golang...
golang has been installed.
  golang may be able to be automatically uninstalled.
Environment Vars (like PATH) have changed. Close/reopen your shell to
see the changes (or in powershell/cmd.exe just type refreshenv ).
The install of golang was successful.
  Software installed as 'msi', install location is likely default.

Chocolatey installed 1/1 packages.
See the log for details (C:\ProgramData\chocolatey\logs\chocolatey.log).
PS C:\WINDOWS\system32> choco install mingw
Chocolatey v0.10.15
Installing the following packages:
mingw
By installing you accept licenses for the packages.
Progress: Downloading mingw 8.1.0... 100%

mingw v8.1.0 [Approved]
mingw package files install completed. Performing other installation steps.
The package mingw wants to run 'chocolateyinstall.ps1'.
Note: If you don't run this script, the installation will fail.
Note: To confirm automatically next time, use '-y' or consider:
choco feature enable -n allowGlobalConfirmation
Do you want to run the script?([Y]es/[A]ll - yes to all/[N]o/[P]rint): Y

Downloading mingw 64 bit
  from 'https://sourceforge.net/projects/mingw-w64/files/Toolchains%20targetting%20Win64/P
ersonal%20Builds/mingw-builds/8.1.0/threads-posix/seh/x86_64-8.1.0-release-posix-seh-rt_v6
-rev0.7z/download'
Progress: 100% - Completed download of C:\Users\Ailsa\AppData\Local\Temp\chocolatey\mingw\
8.1.0\x86_64-8.1.0-release-posix-seh-rt_v6-rev0.7z (47.08 MB).
Download of x86_64-8.1.0-release-posix-seh-rt_v6-rev0.7z (47.08 MB) completed.
Hashes match.
Extracting C:\Users\Ailsa\AppData\Local\Temp\chocolatey\mingw\8.1.0\x86_64-8.1.0-release-p
osix-seh-rt_v6-rev0.7z to C:\ProgramData\chocolatey\lib\mingw\tools\install...
C:\ProgramData\chocolatey\lib\mingw\tools\install
PATH environment variable does not have C:\ProgramData\chocolatey\lib\mingw\tools\install\
mingw64\bin in it. Adding...
Environment Vars (like PATH) have changed. Close/reopen your shell to
see the changes (or in powershell/cmd.exe just type refreshenv ).
```

打开 win+r 输入`refreshenv` 保存修改

Powershell 按下回车:

```
Administrator: Windows PowerShell
ShimGen has successfully created a shim for strings.exe
ShimGen has successfully created a shim for strip.exe
ShimGen has successfully created a shim for widl.exe
ShimGen has successfully created a shim for windmc.exe
ShimGen has successfully created a shim for windres.exe
ShimGen has successfully created a shim for x86_64-w64-mingw32-c++.exe
ShimGen has successfully created a shim for x86_64-w64-mingw32-g++.exe
ShimGen has successfully created a shim for x86_64-w64-mingw32-gcc-8.1.0.exe
ShimGen has successfully created a shim for x86_64-w64-mingw32-gcc-ar.exe
ShimGen has successfully created a shim for x86_64-w64-mingw32-gcc-nm.exe
ShimGen has successfully created a shim for x86_64-w64-mingw32-gcc-ranlib.exe
ShimGen has successfully created a shim for x86_64-w64-mingw32-gcc.exe
ShimGen has successfully created a shim for x86_64-w64-mingw32-gfortran.exe
ShimGen has successfully created a shim for ccl.exe
ShimGen has successfully created a shim for cclplus.exe
ShimGen has successfully created a shim for collect2.exe
ShimGen has successfully created a shim for f951.exe
ShimGen has successfully created a shim for lto-wrapper.exe
ShimGen has successfully created a shim for ltol.exe
ShimGen has successfully created a shim for fixincl.exe
ShimGen has successfully created a shim for gdbmtool.exe
ShimGen has successfully created a shim for gdbm_dump.exe
ShimGen has successfully created a shim for gdbm_load.exe
ShimGen has successfully created a shim for python.exe
ShimGen has successfully created a shim for python2.7.exe
ShimGen has successfully created a shim for python2.exe
ShimGen has successfully created a shim for wininst-6.0.exe
ShimGen has successfully created a shim for wininst-7.1.exe
ShimGen has successfully created a shim for wininst-8.0.exe
ShimGen has successfully created a shim for wininst-9.0-amd64.exe
ShimGen has successfully created a shim for wininst-9.0.exe
ShimGen has successfully created a shim for ar.exe
ShimGen has successfully created a shim for as.exe
ShimGen has successfully created a shim for dlltool.exe
ShimGen has successfully created a shim for ld.bfd.exe
ShimGen has successfully created a shim for ld.exe
ShimGen has successfully created a shim for ld.gold.exe
ShimGen has successfully created a shim for nm.exe
ShimGen has successfully created a shim for objcopy.exe
ShimGen has successfully created a shim for objdump.exe
ShimGen has successfully created a shim for ranlib.exe
ShimGen has successfully created a shim for readelf.exe
ShimGen has successfully created a shim for strip.exe
The install of mingw was successful.
Software installed to 'C:\ProgramData\chocolatey\lib\mingw\tools\install'
Chocolatey installed 1/1 packages.
See the log for details (C:\ProgramData\chocolatey\logs\chocolatey.log).
PS C:\WINDOWS\system32>
```

创建工作环境，及克隆源


```
Administrator: Windows PowerShell

Mode                LastWriteTime         Length Name
-----
d-----          9/29/2019    5:44 PM                Dell
d-----         10/10/2019    6:24 AM                Go
d-----          9/28/2019    5:45 PM                Intel
d-----          9/29/2019   11:52 AM                MinGW
d-----          3/19/2019   12:52 PM                PerfLogs
d-r-----        10/10/2019    6:21 AM            Program Files
d-r-----        10/7/2019    8:12 AM            Program Files (x86)
d-----         10/6/2019    4:52 PM                python
d-r-----          9/29/2019    2:56 AM                Users
d-----         10/6/2019    2:53 PM                Windows
d-----          9/29/2019    3:10 AM            Windows.old

PS C:\> cd Users
PS C:\Users> dir

Directory: C:\Users

Mode                LastWriteTime         Length Name
-----
d-----         10/7/2019    8:33 AM                Ailsa
d-r-----          9/29/2019    5:42 PM                Public

PS C:\Users> cd Ailsa
PS C:\Users\Ailsa> set "GOPATH=%USERPROFILE%"
PS C:\Users\Ailsa> set "Path=%USERPROFILE%\bin;%Path%"
PS C:\Users\Ailsa> setx GOPATH "%GOPATH%"

SUCCESS: Specified value was saved.
PS C:\Users\Ailsa> setx Path "%Path%"

SUCCESS: Specified value was saved.
PS C:\Users\Ailsa> mkdir src\github.com\ethereum

Directory: C:\Users\Ailsa\src\github.com

Mode                LastWriteTime         Length Name
-----
d-----        10/10/2019    6:33 AM                ethereum

PS C:\Users\Ailsa> git clone https://github.com/ethereum/go-ethereum src\github.com\ethere
```

配置自己的创世块是为了区分公有链，同一个网络中，创世块必须是一样的，否则无法联通。在刚刚 Geth 安装目录下放置初始化创世块文件名字为 `genesis.json`

文件内容：

```
1 {
2
3   "config": {
4     "chainId": 7878,
5     "homesteadBlock": 0,
6     "eip155Block": 0,
7     "eip158Block": 0
8   },
9   "difficulty": "200",
10  "gasLimit": "4294967295",
11  "alloc": {
12    "7df9a875a174b3bc565e6424a0050ebc1b2d1d82": { "balance": "300000" },
13    "f41c74c9ae690c1aa78f42e5647a62f353b7bdde": { "balance": "400000" }
14  }
15 }
```

参数名称	参数描述
mixhash	与nonce配合用于挖矿，由上一个区块的一部分生成的hash。注意他和nonce的设置需要满足以太坊的Yellow paper, 4.3.4. Block Header Validity, (44)章节所描述的条件。
nonce	nonce就是一个64位随机数，用于挖矿，注意他和mixhash的设置需要满足以太坊的Yellow paper, 4.3.4. Block Header Validity, (44)章节所描述的条件。
difficulty	设置当前区块的难度，如果难度过大，cpu挖矿就很难，这里设置较小难度
alloc	用来预置账号以及账号的以太币数量，因为私有链挖矿比较容易，所以我们不需要预置有币的账号，需要的时候自己创建即可。
coinbase	矿工的账号，随便填
timestamp	设置创世块的时间戳
parentHash	上一个区块的hash值，因为是创世块，所以这个值是0
extraData	附加信息，随便填，可以填你的个性信息
gasLimit	该值设置对GAS的消耗总量限制，用来限制区块能包含的交易信息总和，因为我们是私有链，所以填最大。

<https://blog.csdn.net/qq35094046>

执行 geth 的 init 命令初始化私链节点：

```
C:\Users\Ailsa\AppData\Local\Ethereum>geth --datadir "%cd%\chain" init genesis.json
INFO [10-10|07:04:08.882] Maximum peer count          ETH=50 LES=0 total=50
INFO [10-10|07:04:08.934] Allocated cache and file handles database=C:\\Users\\Ailsa\\AppData\\Local\\Ethereum\\chain\\geth\\chaindata cache=16.00MiB handles=16
INFO [10-10|07:04:09.044] Writing custom genesis block
INFO [10-10|07:04:09.054] Persisted trie from memory database nodes=3 size=399.00B time=1.002ms gcnodes=0 gcsizes=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [10-10|07:04:09.098] Successfully wrote genesis state database=chaindata hash=f26adf...0f025c
INFO [10-10|07:04:09.106] Allocated cache and file handles database=C:\\Users\\Ailsa\\AppData\\Local\\Ethereum\\chain\\geth\\lightchaindata cache=16.00MiB handles=16
INFO [10-10|07:04:09.232] Writing custom genesis block
INFO [10-10|07:04:09.242] Persisted trie from memory database nodes=3 size=399.00B time=0s gcnodes=0 gcsizes=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [10-10|07:04:09.291] Successfully wrote genesis state database=lightchaindata hash=f26adf...0f025c
```

这会在当前目录下创建 data 目录，用来保存区块数据及账户信息。

打开目录发现多了一些文件：

chain	10/10/2019 7:04 AM	File
data	10/10/2019 7:06 AM	File
data0	10/10/2019 7:44 AM	File
geth	10/10/2019 7:02 AM	File
keystore	10/10/2019 6:53 AM	File
genesis.json	10/10/2019 7:04 AM	JSO
geth.log	10/10/2019 7:44 AM	Text

启动私有链节点

执行如下命令：

```
C:\Users\Ailsa\AppData\Local\Ethereum>geth --rpc --datadir .\data --networkid 7878 console
INFO [10-10|07:06:48.245] Maximum peer count                      ETH=50 LES=0 total=50
INFO [10-10|07:06:48.309] Starting peer-to-peer node               instance=Geth/v1.9.6-stable-bd059680/windows-amd64/gol.13
INFO [10-10|07:06:48.314] Allocated trie memory caches              clean=256.00MiB dirty=256.00MiB
INFO [10-10|07:06:48.317] Allocated cache and file handles          database=C:\Users\Ailsa\AppData\Local\Ethereum\data\geth\chaindata
INFO [10-10|07:06:48.317] Opened database                          cache=512.00MiB handles=8192
INFO [10-10|07:06:48.712] Opened ancient database                   database=C:\Users\Ailsa\AppData\Local\Ethereum\data\geth\chaindata\ancient
INFO [10-10|07:06:48.734] Writing default main-net genesis block
INFO [10-10|07:06:49.041] Persisted trie from memory database        nodes=12356 size=1.79MiB time=54.8562ms gcnodes=0 gcsz=0.00B gctime=0s livenodes=1 liveness=0.00B
INFO [10-10|07:06:49.047] Initialised chain configuration            config="{ChainID: 1 Homestead: 1150000 DAO: 1920000 DAOSupport: true EIP150: 2463000 EIP155: 2675000 EIP158: 2675000 Byzantium: 4370000 Constantinople: 7280000 Petersburg: 7280000 Istanbul: <nil> Engine: ethash}"
INFO [10-10|07:06:49.056] Disk storage enabled for ethash caches     dir=C:\Users\Ailsa\AppData\Local\Ethereum\data\geth\ethash count=3
INFO [10-10|07:06:49.062] Disk storage enabled for ethash DAGs       dir=C:\Users\Ailsa\AppData\Local\Ethereum\data\geth\ethash count=2
INFO [10-10|07:06:49.072] Initialising Ethereum protocol            versions=[63] network=7878 dbversion=63
WARN [10-10|07:06:49.075] Upgrade blockchain database version        from=<nil> to=7
INFO [10-10|07:06:49.104] Loaded most recent local header            number=0 hash=d4e567...cb8fa3 td=1717986 age=50y5mo4w
INFO [10-10|07:06:49.109] Loaded most recent local full block        number=0 hash=d4e567...cb8fa3 td=1717986 age=50y5mo4w
INFO [10-10|07:06:49.116] Loaded most recent local fast block        number=0 hash=d4e567...cb8fa3 td=1717986 age=50y5mo4w
INFO [10-10|07:06:49.123] Regenerated local transaction journal       transactions=0 accounts=0
INFO [10-10|07:06:49.144] Allocated fast sync bloom                 size=512.00MiB
INFO [10-10|07:06:49.205] Initialized fast sync bloom                items=12356 errorrate=0.000 elapsed=57.872ms
INFO [10-10|07:06:49.338] New local node record                      seq=1 id=fa2181eb8d464d87 ip=127.0.0.1 udp=30303 tcp=30303
INFO [10-10|07:06:49.346] Started P2P networking                     self=enode://317d9e32837f2f241765b5ff90a67afc944222a0765546bf7591288c668d1f4f010dc205eb0b6992a953462bala5ab59ba0f0515a43b59d00f61e98558d263d7@127.0.0.1:30303
INFO [10-10|07:06:49.348] IPC endpoint opened                        url=\\.\pipe\geth.ipc
INFO [10-10|07:06:49.363] HTTP endpoint opened                       url=http://127.0.0.1:8545 cors= vhosts=localhost
WARN [10-10|07:06:49.442] Served eth_coinbase                       reqid=3 t=0s err="etherbase must be explicitly specified"
Welcome to the Geth JavaScript console!

instance: Geth/v1.9.6-stable-bd059680/windows-amd64/gol.13
at block: 0 (Thu, 01 Jan 1970 08:00:00 CST)
datadir: C:\Users\Ailsa\AppData\Local\Ethereum\data
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
```

接着启动私有链，运行以下命令：

```
geth --datadir data0 --networkid 1108 --nodiscover console
```

```
2>>geth. log
```

--datadir 代表文件夹地址，

--nodiscover 代表该链条不希望被其他节点发现，

--networkid 示这个私有链的网络 id 为 1108，网络 id 在连接到其他节点的时候会用到

console >> geth. log 代表将控制台输出到文件 geth. log 中去

```
C:\Users\Ailsa\AppData\Local\Ethereum>geth --datadir data0 --networkid 1108 --nodiscover console 2>>geth. log
Welcome to the Geth JavaScript console!

instance: Geth/v1.9.6-stable-bd059680/windows-amd64/go1.13
at block: 0 (Thu, 01 Jan 1970 08:00:00 CST)
datadir: C:\Users\Ailsa\AppData\Local\Ethereum\data0
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
```

添加节点 00

在节点 00 的控制台，查看 00 节点的 enode

我们通过分享 enode 地址的方式来让两个节点建立链接：

```
> admin.nodeInfo.enode
enode://54488488515ba0f3e5597de48617d4b5b7bd055e6f787f7bb2b036838ca0e75e16cca3a5dba244ba23fcb3716e10e30f9f8d832bb316896c153bbaae3ab2b19e@127.0.0.1:30303?discport=0
```

在节点 01 的控制台，添加节点 00

```
> admin.nodeInfo.enode
enode://317d9e32837f2f241765b5ff90a67af94422a0765546bf7591288c668d1f4f010dc205eb0b6992a953462ba1a5ab59ba0f0515a43b59400f61e985584263d78127.0.0.1:30303"
> admin.nodeInfo
{
  enode: "enode://317d9e32837f2f241765b5ff90a67af94422a0765546bf7591288c668d1f4f010dc205eb0b6992a953462ba1a5ab59ba0f0515a43b59400f61e985584263d78127.0.0.1:30303",
  enr: "enr:-Jg4GJ3C0013a1uadRCv4CN-44j2JhG2a081Cva1d4600wG25_YlyNtaiYVq-94d-e-Jhg8i11Q58xyoQZayca3g2V0aMrJhPpk7ASDEYwgn1kgY0gn1whH8AAAGJc2VjcD11NesxxoQWxf24rg38vJbdltrf-Qqer81E1icH2YR",
  id: "fa2181eb93464d7837716b3c2d4884ac89045fa6b477f7f16ee8ae679cd1084fc",
  ip: "127.0.0.1",
  listenAddr: "[::]:30303",
  name: "geth.v1.9.6-stable-bd059680/windows-amd64/go1.13",
  ports: {
    discovery: 30303,
    listener: 30303
  },
  protocols: {
    eth: {
      config: {
        byzantiumBlock: 4370000,
        chainId: 1,
        constantinopleBlock: 7280000,
        daoForkBlock: 1920000,
        daoForkSupport: true,
        eip150Block: 2403000,
        eip150Hash: "8ac3866699ae8eae135c246c65021c2b4e15a2c431340993aaccf42781886514f0",
        eip158Block: 2670000,
        eip158Block: 2670000,
        ethash: {},
        homesteadBlock: 1150000,
        petersburgBlock: 7280000
      },
      difficulty: 1717989184,
      genesis: "0xd4e56740f876aef8c010b86a40d5f56745a118a0990434469bbcc30db1c8fca3",
      head: "0xd4e56740f876aef8c010b86a40d5f56745a118a0990434469bbcc30db1c8fca3",
      network: 7878
    }
  }
}
> admin.addPeer("enode://317d9e32837f2f241765b5ff90a67af94422a0765546bf7591288c668d1f4f010dc205eb0b6992a953462ba1a5ab59ba0f0515a43b59400f61e985584263d78127.0.0.1:30303")
true
```

连接成功之后，节点 01 会自动快速同步节点 00 的区块

查看连接的节点

可以在节点 00 和 01 控制台中, 查看连接的节点数量和连接的节点列表

```
net.peerCount
admin.peers
geth --networkid 14 --nodiscover --datadir /home/blockChain/data/01 --port 61911 --rpcapi net,eth,web3,personal --rpc --rpcaddr ip_address --rpcport 8101 console
(anonymous): Line 1:8 Unexpected identifier (and 1 more errors)
```

getBlock() 所得区块的各个字段解释

[illegible]

difficulty: 大小 4 字节，存储格式为 难度系数的 HASH 值，该字段标记着当前区块被“挖”出来的难度（哈希碰撞出来的难度）；

ExtraData: 与此区块相关的附加数据

asLimit: 当前区块允许使用的最大 gas。

gasUsed: 当前区块累计使用的 gas。

Hash: 区块的哈希值。如果区块没有被确认, 这个字段会是 null 值。

LogsBloom: 区块日志的布隆过滤器, 区块没被确认是值为 null

Miner:取得该区块记账权的矿工。

mixhash: 与 nonce 配合用于挖矿, 由上一个区块的一部分生成的 hash。

Nonce: 大小 4 字节，当前区块工作量证明 (Proof of Work) 的参数

(是以一坨 0 开头的数), 存储格式为 Hash 值。hash 计算的目标值, 改值随机。当“矿机”节点经过 Hash 计算出的值为该随机数时, 即为“挖矿”初步成果(经过后续六个区块的认证之后, 才是真正的成果, 终态)

Number: 区块号

parentHash: 前一个区块的哈希值

receiptsRoot: 收据树的根哈希值

Sha3Uncles: 数据块的哈希值

size: 区块大小

stateRoot: 区块状态树的根哈希

Timestamps: 大小 4 字节, 核心字段, 自 1970-01-01T00:00 UTC 之后开始的秒数;

totalDifficulty: 截止到本块的链上总难度

transactions: 交易对象数组

transactionsRoot: 交易的区块数根哈希值

Uncles: 叔哈希的数组

对日志输出进行解释

```
miner.start()
INFO [10-10 08:53:11.154] Updated mining threads          threads=4
INFO [10-10 08:53:11.157] Transaction pool price threshold updated price=1000000000
INFO [10-10 08:53:11.160] Etherbase automatically configured address=0x163528c75d2f6cE7d1f263Ca8f8c73c33dad
INFO [10-10 08:53:11.165] Commit new mining work          number=1 sealhash=8a85ba...3cc491 uncles=0 txs=0 gas=0 fees=0 elapsed=0s
INFO [10-10 08:53:18.223] Generating DAG in progress      epoch=1 percentage=0 elapsed=2.862s
INFO [10-10 08:53:18.127] Generating DAG in progress      epoch=1 percentage=1 elapsed=5.766s
INFO [10-10 08:53:21.571] Generating DAG in progress      epoch=1 percentage=2 elapsed=9.210s
INFO [10-10 08:53:24.619] Generating DAG in progress      epoch=1 percentage=3 elapsed=12.257s
INFO [10-10 08:53:27.222] Generating DAG in progress      epoch=1 percentage=4 elapsed=15.061s
INFO [10-10 08:53:30.141] Generating DAG in progress      epoch=1 percentage=5 elapsed=17.780s
INFO [10-10 08:53:33.004] Generating DAG in progress      epoch=1 percentage=6 elapsed=20.643s
INFO [10-10 08:53:33.675] Generating DAG in progress      epoch=1 percentage=7 elapsed=23.314s
INFO [10-10 08:53:38.293] Generating DAG in progress      epoch=1 percentage=8 elapsed=29.118s
INFO [10-10 08:53:41.299] Generating DAG in progress      epoch=1 percentage=9 elapsed=28.938s
INFO [10-10 08:53:44.103] Generating DAG in progress      epoch=1 percentage=10 elapsed=31.742s
INFO [10-10 08:53:47.101] Generating DAG in progress      epoch=1 percentage=11 elapsed=34.740s
INFO [10-10 08:53:49.811] Generating DAG in progress      epoch=1 percentage=12 elapsed=37.450s
INFO [10-10 08:53:52.871] Generating DAG in progress      epoch=1 percentage=13 elapsed=40.510s
INFO [10-10 08:53:55.703] Generating DAG in progress      epoch=1 percentage=14 elapsed=43.342s
INFO [10-10 08:53:58.385] Generating DAG in progress      epoch=1 percentage=15 elapsed=46.224s
INFO [10-10 08:54:01.545] Generating DAG in progress      epoch=1 percentage=16 elapsed=49.184s
INFO [10-10 08:54:04.252] Generating DAG in progress      epoch=1 percentage=17 elapsed=51.891s
INFO [10-10 08:54:06.983] Generating DAG in progress      epoch=1 percentage=18 elapsed=54.607s
INFO [10-10 08:54:09.713] Generating DAG in progress      epoch=1 percentage=19 elapsed=57.352s
INFO [10-10 08:54:12.489] Generating DAG in progress      epoch=1 percentage=20 elapsed=60.128s
INFO [10-10 08:54:15.349] Generating DAG in progress      epoch=1 percentage=21 elapsed=62.988s
INFO [10-10 08:54:18.288] Generating DAG in progress      epoch=1 percentage=22 elapsed=65.877s
INFO [10-10 08:54:20.886] Generating DAG in progress      epoch=1 percentage=23 elapsed=68.525s
INFO [10-10 08:54:23.677] Generating DAG in progress      epoch=1 percentage=24 elapsed=71.316s
INFO [10-10 08:54:26.410] Generating DAG in progress      epoch=1 percentage=25 elapsed=74.049s
INFO [10-10 08:54:29.205] Generating DAG in progress      epoch=1 percentage=26 elapsed=76.844s
INFO [10-10 08:54:32.092] Generating DAG in progress      epoch=1 percentage=27 elapsed=79.731s
INFO [10-10 08:54:34.851] Generating DAG in progress      epoch=1 percentage=28 elapsed=82.490s
INFO [10-10 08:54:37.244] Generating DAG in progress      epoch=1 percentage=29 elapsed=85.183s
INFO [10-10 08:54:40.310] Generating DAG in progress      epoch=1 percentage=30 elapsed=87.949s
INFO [10-10 08:54:43.048] Generating DAG in progress      epoch=1 percentage=31 elapsed=90.687s
INFO [10-10 08:54:45.822] Generating DAG in progress      epoch=1 percentage=32 elapsed=93.461s
INFO [10-10 08:54:48.602] Generating DAG in progress      epoch=1 percentage=33 elapsed=96.241s
INFO [10-10 08:54:51.386] Generating DAG in progress      epoch=1 percentage=34 elapsed=99.025s
INFO [10-10 08:54:54.168] Generating DAG in progress      epoch=1 percentage=35 elapsed=101.807s
INFO [10-10 08:54:56.983] Generating DAG in progress      epoch=1 percentage=36 elapsed=104.507s
INFO [10-10 08:54:59.673] Generating DAG in progress      epoch=1 percentage=37 elapsed=107.312s
INFO [10-10 08:55:02.430] Generating DAG in progress      epoch=1 percentage=38 elapsed=110.069s
INFO [10-10 08:55:05.160] Generating DAG in progress      epoch=1 percentage=39 elapsed=112.799s
INFO [10-10 08:55:07.520] Generating DAG in progress      epoch=1 percentage=40 elapsed=115.458s
```

编写简单的智能合约, 在 remix 下进行调试

用 remix 编译一个简单的智能合约

```
pragma solidity ^0.4.24;
```

```
contract HelloWorld {

    function HelloWorld() public pure returns (string) {

        return ("hello world");

    }

}
```

```
1 pragma solidity ^0.4.24;
2 contract HelloWorld{
3     function say() public pure returns(string){
4         return "Hello Wrold";
5     }
6 }
```

HelloWorld

Run: 在 run 中点击 deploy 后就会显示可以调用的函数

HelloWorld at 0x692...77b3a (memory)

say ▶ 0: string: Hello Wrold

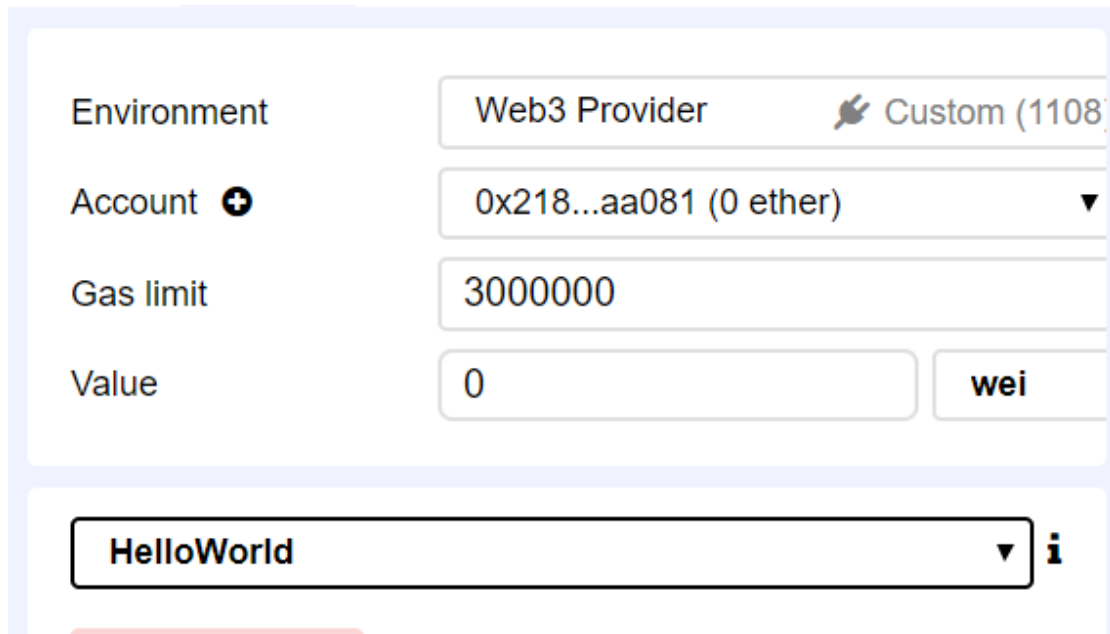
部署在链上进行调用

重新启动私有链，要在 geth 上加上 rpc，将 environment 改成 web3

```
geth --datadir data0 --networkid 1108 --rpc --rpcaddr 0.0.0.0
--rpcport 8545 --rpcapi
"admin, debug, eth, miner, net, personal, shh, txpool, web3" --
```

```
rpccorsdomain "*" --nodiscover --ipcdisable console
```

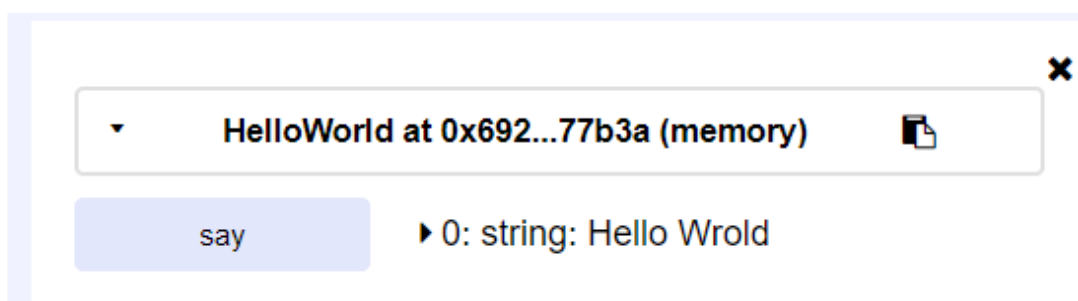
```
2>>geth.log
```



The screenshot shows a web interface for deploying a contract. It includes fields for Environment (Web3 Provider), Account (0x218...aa081 (0 ether)), Gas limit (3000000), and Value (0 wei). Below these fields is a dropdown menu showing 'HelloWorld'.

解锁账号，否则不能部署

```
personal.unlockAccount(eth.accounts[0], "密码", 0)
```



The screenshot shows the result of the deployment. It displays 'HelloWorld at 0x692...77b3a (memory)' with a button labeled 'say'. Below the button, it shows the output: '0: string: Hello Wrold'.

要先挖矿赚钱，否则没钱部署


```

> eth.getBalance("0x21817935bbacca
0
> miner.start()
null
> eth.getBalance("0x21817935bbacca
7.115e+21

```

然后点击 deploy，需要注意的是 remix 似乎不支持带有 library 的合约。

<div> <div></div> <div> <div>[block:1431 txIndex:0]</div> <div>from:0x218...aa081 to:HelloWorld. (constructor) va</div> <div>data:0x608...80029 logs:0 hash:0x619...d3cbe</div> </div> </div>	
status	Status not available at the moment
transaction hash	0x6198b6a80be77d9b68fba80ad0fd2a1436ecf24b8877390245a4d897ccc
from	0x21817935bbacca1ae01824d5b6e06b5da53aa081
to	HelloWorld. (constructor)
gas	136299 gas
transaction cost	136299 gas

Deployed Contracts

▼

HelloWorld at 0x56b...ea456 (blockchain)

|

say

在 geth 上调用合约

remix 有一个复制 abi 的按钮（见下图），地址就在上图可以找到，

abi 要化为一行（百度查看在线 json 压缩）

Current
version:0.4.25+commit.59dbf8f1.Emscripten.clang

Select new compiler version ▼

☒ Auto compile ☐ Enable Optimization
☒ Hide warnings

🔄 Start to compile

SafeMath ▼ ⬆ Swarm

Details ABI Bytecode

```
>  
> var abi = [{"constant": true,"inputs": [],"name": "say","outputs": [{"name":  
","type": "string"}],"payable": false,"stateMutability": "pure","type": "functi  
n"}];  
  
> var contract = web3.eth.contract(abi).at("0x56b5834e0a1401d4faf70cf3684828  
7ea456")  
  
undefined  
> contract.say()  
  
"Hello world"
```

对交易的字段进行解释

在两个账号间发送一笔交易

```
"0x21817935bbacca1ae01824d5b6e06b5da53aa
> eth.getBalance(eth.accounts[0])

7.275e+21
> eth.getBalance(eth.accounts[1])

(anonymous): Line 1:32 Unexpected token
```

```
true
> eth.sendTransaction({from:eth.accounts[0],to:eth.accounts[1],value:web3.toWei(3,'ether')})

> eth.getTransaction("0x2149e8120734c5fc9dc61545d414ddb21cbe75983af03487648d1255cd435e20")

{
  blockHash: "0x0000000000000000000000000000000000000000000000000000000000000000",
  blockNumber: null,
  from: "0x21817935bbacca1ae01824d5b6e06b5da53aa081",
  gas: 90000,
  gasPrice: 1000000000,
  hash: "0x2149e8120734c5fc9dc61545d414ddb21cbe75983af03487648d1255cd435e20",
  input: "0x",
  nonce: 1,
  r: "0x1529bce53b71e24cb66dc8d23fe15a5b9b2651ed8e33ed636bb0662c8b125cc9",
  s: "0x3bc22f5b6d690218ef6fc03486c1b9da79a26b5cf76fc532d7074e18e3fc5b0c",
  to: "0x20fa68b80368213958646d9e320e09a6a7f07486",
  transactionIndex: 0,
  v: "0x38",
  value: 3000000000000000000
}
```

通过

```
web3.eth.getTransaction( '0x2149e8120734c5fc9dc61545d414ddb21cbe75983af03487cd435e20' );
```

也可以获得交易的信息，通过获得消息回执，可以获得事件信息：

```
web3.eth.getTransactionReceipt( '0x2149e8120734c5fc9dc61545d414ddb21cbe75983af03487cd435e20' );
```

这是一条事件日志， 执行的合约是：EOSTokenContract

调用的函数是 transfer 函数。