آموزش کار باGETH(Go Ethereum)

یکی از پیاده سازی های رسمی اتریوم به زبان Go است. (علاوه بر زبان های C++و پایتون .(در واقع به کمک واسط کاربری برخط Geth می توان یک full node اتریوم ایجاد کرد. هنگامی که شما نود اتریوم خصوصی خود را دارید، دیگر نیاز به استفاده از یک سرویس ثالث برای اتصال به شبکه اتریوم وجود ندارد و می توانید به نود خود اعتماد کنید.

پیش از شروع به کار با این ابزار، به این نکته توجه داشته باشید که بیشترین کاربرد geth برای ایجاد یک نود کامل و یا به اصطلاح کلاینت در شبکه اتریوم است. ایجاد یک نود کامل که تمام بلاک های زنجیره را در خود داشته باشد و مدام در حال بروزرسانی وضعیت خود باشد؛ می تواند مورد استفاده ماینرها قرار بگیرد؛ و یا می تواند صرفا به عنوان تامین کننده سرویس بلاکچین اتریوم استفاده گردد و یا توسط صرافی های آنلاین برای افزایش سرعت بروزرسانی تراکنش های خود استفاده شود.

بنابراین می توان گفت geth در فرآیند توسعه اپلیکیشن های غیرمتمرکز معمولا استفاده نمی شود. به دلیل فرآیند پیچیده و زمان بر بودن کار با آن .در واقع geth بیشتر جنبه آموزشی دارد و کار با آن موجب عمیق تر شدن دانش ما نسبت به بلاکچین خواهد شد.

نصب

آخرین نسخه geth رو می تونید از https://geth.ethereum.org/docs/install-and-build/installing-geth

برای اطمینان از نصب موفق یک پنجره ترمینال باز کنید و دستور زیر را اجرا کنید:

```
$ geth
```

در صورت اجرای دستور بالا، چنین چیزی مشاهده خواهید کرد:

```
INFO [07-04|10:25:18.313] Disk storage enabled for ethash DAGs       dir=C:\Users\harry\AppData\Local\Ethash count=2
INFO [07-04|10:25:18.322] Initialising Ethereum protocol             network=1 dbversion=8
INFO [07-04|10:25:18.365] Loaded most recent local header            number=0 hash=d4e567..cb8fa3 td=17,179,869,184 age=52
y3mo2d
INFO [07-04|10:25:18.374] Loaded most recent local full block        number=0 hash=d4e567..cb8fa3 td=17,179,869,184 age=52
y3mo2d
INFO [07-04|10:25:18.383] Loaded most recent local fast block        number=0 hash=d4e567..cb8fa3 td=17,179,869,184 age=52
y3mo2d
WARN [07-04|10:25:18.395] Snapshot maintenance disabled (syncing)
INFO [07-04|10:25:18.400] Loaded local transaction journal           transactions=0 dropped=0
INFO [07-04|10:25:18.408] Regenerated local transaction journal      transactions=0 accounts=0
INFO [07-04|10:25:18.416] Gasprice oracle is ignoring threshold set threshold=2
INFO [07-04|10:25:18.425] Starting peer-to-peer node                 instance=Geth/v1.10.4-stable-aa637fd3/windows-amd64/g
o1.16.4
INFO [07-04|10:25:18.501] New local node record                      seq=5 id=78c14be3c8ac3d16 ip=127.0.0.1 udp=30303 tcp=
30303
INFO [07-04|10:25:18.519] Started P2P networking                     self=enode://e62aef4c89be2c2ce09a40a801ece03a74dffa8e
ea65ccc95b484e48f3e16cbe666d5b1c7a47737cb87420c76c76d4129f4e4b0e9df155294b56196ec0454194@127.0.0.1:30303
INFO [07-04|10:25:18.526] IPC endpoint opened                        url=\\.\pipe\geth.ipc
INFO [07-04|10:25:21.363] Mapped network port                        proto=tcp extport=30303 intport=30303 interface="UPNP
 IGDv1-IP1"
INFO [07-04|10:25:21.519] New local node record                      seq=6 id=78c14be3c8ac3d16 ip=2.191.96.243 udp=30303 t
cp=30303
INFO [07-04|10:25:21.818] Mapped network port                        proto=udp extport=30303 intport=30303 interface="UPNP
 IGDv1-IP1"
INFO [07-04|10:25:28.580] Looking for peers                          peercount=0 tried=9 static=0
INFO [07-04|10:25:32.042] Block synchronisation started
WARN [07-04|10:25:32.049] Enabling snapshot sync prototype
INFO [07-04|10:25:39.379] Looking for peers                          peercount=1 tried=29 static=0
```

با اجرای این دستور، درواقع geth شروع به جستجو و اتصال به سایر نودهای اتریوم می کند. زمانی که geth یک نود در همسایگی خود پیدا کند؛ شروع به سینک کردن داده های بلاکچین می کند. برای متوقف کردن این فرآیند، می توان از کلید Ctrl+Cاستفاده کرد.
همانطور که گفته شد با اجرای دستور geth به صورت پیش فرض یک نود mainnet اتریوم ایجاد می شود. محل ذخیره سازی داده های مربوط به این زنجیره به صورت پیش فرض با توجه به نوع سیستم عامل متفاوت است.

- Windows

```
C:\Users\[USERNAME]\AppData\Local\Ethereum
or
%APPDATA%\Ethereum
```

در مسیر مشخص شده دو پوشه با نام های geth و keystore وجود دارد. پوشه gethاطلاعات مربوط به وضعیت زنجیره و بلاک ها را ذخیره می کند؛ و پوشه keystoreمحل نگه داری اکانت های مربوط به بلاکچین است.

هچنین geth از طریق ایجاد یک بستر IPC امکان اتصال به نود مشخصی را فراهم می کند.

```
$ geth attach [path to the ipc endpoint]
```

به طور مثال برای اتصال به نود ایجاد شده برروی سیستم خود دستور زیر استفاده می شود:

```
geth attach geth.ipc
```

مسیر فایل تنظیمات برای اتصال به نود ایجاد شده، بر روی سیستم عامل های مختلف متفاوت است. به عنوان مثل مسیر پیشفرض برای فایل سوکت geth در سیستم عامل های لینوکس و مک در اینجا قرار دارد:

```
~/.ethereum/geth.ipc
```

و در ویندوز از این طریق در دسترس است:

```
\\.\pipe\geth.ipc
```

از طریق دستور زیر می توان هم زمان با اجرای geth وارد کنسول جاوااسکریپت نیز شد:

```
$ geth console
```

در این صورت هم زمان کنسول جاوااسکریپت اجرا خواهد شد. نمایش لاگ ها در پنجره کنسول هم چنان ادامه دارد، که این ی تواند در هنگام کار با کنسول آزاردهنده باشد؛ برای حذف نمایش لاگ ها دستور زیر را در کنسول تایپ کنید:

```
debug.verbosity(0)
```

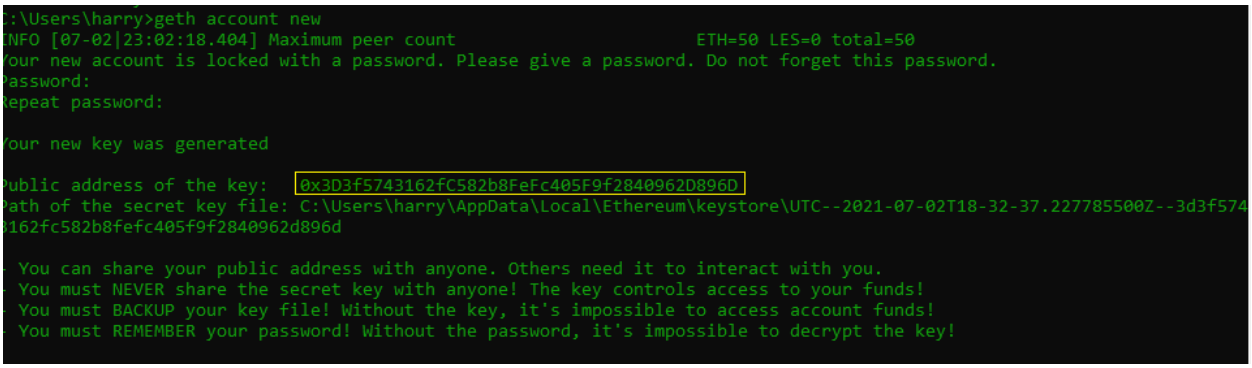
همچنین برای خروج از کنسول جاوااسکریپت در کنسول دستور exitرا اجرا کنید.
هنگامی که دستور geth را اجرا می کنیم، به طور پیش فرض با شبکه اصلی اتریوم ارتباط برقرار می شود ولی به کمک ابزار geth می توان به هر شبکه دلخواهی وصل شد.

ایجاد اکانت اتریوم

قبل از ایجاد یک بلاکچین خصوصی نیاز داریم تا یک حساب اتریوم ایجاد کنیم با استفاده از دستور زیر:

```
$ geth account new
```

با اجرای این دستور، چنین تصویری مشاهده خواهید کرد. توجه داشته باشید که رمز عبور را به خاطر بسپارید.



همان طور که در تصویر مشاهده می کنید در قسمت `path of the secret key file` آدرس محل ذخیره سازی کلید خصوصی مربوط به آدرس ایجاد شده به صورت رمزنگاری شده مشخص شده است. محل ذخیره سازی این کلید در پوشه `keystore` مربوط به نود شبکه اصلی است.

ساخت بلاکچین خصوصی

قبل از هر چیز یک پوشه برای بلاکچین خصوصی خود بر روی سیستم ایجاد کنید.
ساخت هر بلاکچین با یک بلاک آغازین یا بلاک genesis آغاز می شود. برای این کار می بایست یک فایل با فرمت json ایجاد کنیم و تنظیمات بلاک آغازین را در آن قرار دهیم. یک فایل با نام `genesis.json` داخل پوشه بلاکچین خود ایجاد کنید و محتویات زیر را داخل آن کپی کنید.

```
{
  "config": {
        "chainId": 4568,
        "homesteadBlock": 0,
        "eip150Block": 0,
        "eip155Block": 0,
        "eip158Block": 0
    },
  "alloc"      : {},
  "difficulty" : "400",
  "extraData"  : "",
  "gasLimit"   : "15000000",
  "parentHash" :
"0x0000000000000000000000000000000000000000000000000000000000000000",
  "timestamp"  : "0x00"
}
```

در ادامه بخش های مختلف فایل genesis توضیح داده می شود:

۱. config: تنظیمات مربوط به بلاکچین خصوصی در این بخش تعریف می شود:

chainId: مشخص کننده بلاکچین است. زنجیره اصلی اتریوم با شماره ۱ مشخص می شود؛ شبکه تست Ropstenبا شماره زنجیره ۳ مشخص می شود؛ Rinkeby شماره ٤ و Kovan شماره ٤٢ است. شبکه خصوصی ای که ما ایجاد می کنیم دارای شماره زنجیره ٤٥٦٨ است. این عدد می تواند هر شماره دلخواهی که قبلا استفاده نشده است؛ باشد. با یک کلاینت مشترک می توان به انواع شبکه های اتریوم متصل شد.
homesteadBlock: این فیلد و سه فیلد بعدی مشخص کننده این است که از کدام شماره بلاک تغییرات مربوط به هارد فورک ها آغاز می شود. در اینجا ما بلاکچین خود را داریم و از بلاک آغازین شروع می کنیم، بنابراین تمام فیلدها را صفر قرار ی دهیم.

۲. alloc: در این بخش می توانیم به محض ایجاد بلاکچین، آدرس ایجاد کنیم و آن ها را شارژ کنیم. ما با این قسمت کاری نداریم. برای شارژ اکانت ها از ماین کردن کمک می گیریم.
۳. difficulty: میزان سختی برای یافتن یک هش بلاک معتبر را مشخص می کند. و یا به عبارت دیگر هدف ماینینگ را شخص می کند. این مقدار در حال نوسان است تا زمان تولید یک بلاک را تقریبا در ۱۵ ثانیه نگه دارد. در اینجا برای اینکه سریع به جواب برسیم میزان سختی را پایین درنظر می گیریم.
٤. extraData: یک مقدار اختیاری به اندازه ۳۲ بایت است که هر عبارت دلخواهی می تواند باشد.
٥. gasLimit: حداکثر مقدار مجاز gas برای هر بلاک را مشخص می کند. عدد ۱۵,۰۰۰,۰۰۰ مقدار در حال حاضر بر روی شبکه اصلی است.
٦. parentHash: مقدار هش هدر بلاک قبلی یا والد را مشخص می کند .درواقع همین اشاره گر به بلاک قبلی، زنجیره بلاک ها را ایجاد می کند.
۷. timestamp: زمان به فرمت یونیکس در هنگام ساخت بلاک را مشخص می کند. این پارامتر به ما کمک می کند چه زمانی سطح سختی می بایست تغییر کند تا میانگین زمان تولید یک بلاک ثابت بماند. همچنین به کمک این پارامتر می توان صحت ترتیب بلاک ها را داخل یک زنجیره تایید کرد.

سپس زنجیره خصوصی خود را با استفاده از فایل genesis ساخته شده، ایجاد می کنیم. دستور زیر را در پوشه مربوط به بلاکچین خصوصی که ساخته اید، اجرا کنید.

```
$ geth --datadir . init genesis.json
```

پیش از آنکه سراغ اجرای نود برویم با استفاده از دستور geth دو اکانت ایجاد می کنیم. اکانت اول به عنوان اکانت اصلی یا آدرس ماینر و اکانت دوم برای انجام تراکنش استفاده می شود. در پوشه مربوط به شبکه خصوصی دستور زیر را اجرا کنید، فراموش نکنید که رمز عبور را به خاطر بسپارید و همچنین آدرس های تولید شده را در جایی نگه دارید.

```
geth --datadir . account new
```

حالا می توانیم بلاکچین خود را اجرا کنیم و بلاک های زنجیره را ماین کنیم. برای اجرای نود وارد پوشه شبکه خود شوید و دستور زیر را اجرا کنید:

```
$  geth --allow-insecure-unlock --datadir . --keystore keystore --networkid
4568 --http --http.corsdomain "*" --http.port 8502 --http.api
personal,eth,net,web3,txpool,miner --mine --
miner.etherbase=0xD04E55037Cab53E4d711BBC81f275e9d37629C9c
```

با اجرای دستور، چنین چیزی می بایست مشاهده کنید:

```
INFO [07-06|09:39:35.849] HTTP server started                      endpoint=127.0.0.1:8502 prefix= cors=* vhosts=localho
st
INFO [07-06|09:39:35.860] Transaction pool price threshold updated price=1,000,000,000
INFO [07-06|09:39:35.865] Updated mining threads                   threads=0
INFO [07-06|09:39:35.871] Transaction pool price threshold updated price=1,000,000,000
INFO [07-06|09:39:35.877] Commit new mining work                   number=95 sealhash=430493..8e6e20 uncles=0 txs=0 gas=
0 fees=0 elapsed="531µs"
INFO [07-06|09:39:38.334] New local node record                    seq=4 id=b633a4ae0e65fe13 ip=2.191.22.147 udp=30303 t
cp=30303
INFO [07-06|09:39:38.643] Mapped network port                      proto=tcp extport=30303 intport=30303 interface="UPNP
 IGDv1-IP1"
INFO [07-06|09:39:39.090] Mapped network port                      proto=udp extport=30303 intport=30303 interface="UPNP
 IGDv1-IP1"
INFO [07-06|09:39:46.229] Looking for peers                        peercount=0 tried=66 static=0
INFO [07-06|09:39:56.802] Looking for peers                        peercount=0 tried=55 static=0
```

حالا می توانیم به زنجیره خصوصی خود متصل شویم. با توجه به اینکه نود مورد نظر بر روی سیستم خودمان است می توانیم از پروتکل IPC استفاده کنیم .پیش از آن در هنگام اجرای دستور قبل عبارت `IPC endpoint opened`را می توان مشاهده کرد. این آدرس بر روی سیستم عامل های مختلف ممکن است متفاوت باشد. به عنوان مثال بر روی سیستم عامل ویندوز به شکل زیر وجود دارد:

```
INFO [07-06|10:16:43.088] IPC endpoint opened                      url=\\.\pipe\geth.ipc
INFO [07-06|10:16:43.118] HTTP server started                      endpoint=127.0.0.1:8502 prefix= cors=* vhosts=localho
st
```

می توانیم یک ترمینال جدید باز کنیم و از طریق ایجاد یک بستر IPC به نود مورد نظر متصل شویم.
پس از اتصال به نود می بایست چنین چیزی شاهده کنیم:

```
Welcome to the Geth JavaScript console!

instance: Geth/v1.10.4-stable-aa637fd3/windows-amd64/go1.16.4
coinbase: 0x3d3f5743162fc582b8fefc405f9f2840962d896d
at block: 0 (Thu Jan 01 1970 03:30:00 GMT+0330 (+0330))
 datadir: C:\MyFiles\Blockchain\CoinIran\chapter4\privateBlockchain
 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
>
```

به پارامترهای `datadir` و`coinbase`دقت کنید که به ترتیب آدرس محل ذخیره سازی بلاکچین خصوصی و آدرس اتریوم معرفی شده توسط شما به عنوان ماینر یا همان آدرس اصلی هستند.
در کنسول باز شده می توانید دستورات مورد نظر را برای ارتباط با نود خصوصی، اجرا کنید.

بررسی اکانت

شما می توانید به تمام اکانت های geth از طریق `eth.accounts`دسترسی داشته باشید.

```
> eth.accounts
["0x1c29b7832ad2e4731d816e60f767777cbf374e15",
"0x01058d7d56a216b3f1ff8f41c20ad58888424de7"]
```

همچنین می توانید موجودی هر حساب را با استفاده از متد `eth.getBalance()` مشاهده کنید.

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

اولین حساب ایجاد شده یا حساب اصلی از طریق `eth.coinbase` نیز در دسترس است.

ماینینگ

از آنجایی که شبکه خصوصی ایجاد شده همانند شبکه اتریوم از مکانیزم اجماع اثبات کار استفاده می کند، برای اجرای تراکنش ها می بایست بلاک ها را ماین کنیم.
برای شروع ماینینگ کافیست دستور `miner.start()` را اجرا کنیم. می توانید فرآیند ماینینگ را بر روی ترمینال نود مشاهده کنید. بعد از آنکه تعدادی بلاک ماین شد با دستور `miner.stop()` می توانید فرآیند را متوقف کنید. حالا می توانید موجودی حساب اصلی خود را مجددا چک کنید که باید شامل مقداری اتر باشد.

ارسال تراکنش

باتوجه به اینکه مقداری اتر در حساب خود داریم می توانیم یک تراکنش انجام دهیم و به یک حساب دیگر اتر واریز کنیم.
برای انجام تراکنش در کنسول gethاز متد `eth.sendTransaction()` استفاده می کنیم. ولی پش از آن ی بایست بدانیم یک تراکنش اتریوم شامل چه داده های است:

- from:از نوع `String`، آدرس ارسال کننده را مشخص می کند.
- to:از نوع `String`، آدرس گیرنده یا مقصد را شخص می کند.
- value:فیلد اختیاری، از نوع `String`, `Number`و یا `BigNumber`می تواند باشد. مبلغی که در تراکنش منتقل می شود بر حسب wei.
- gas:فیلد اختیاری، از نوع `String`, `Number`و یا `BigNumber`می تواند باشد. میزان gas ای که برای تراکنش مصرف می شود.
- gasPrice:فیلد اختیاری، از نوع `String`, `Number`و یا `BigNumber`. هزینه gas برای این تراکنش برحسب wei.مقدار پیشفرض براساس حداقل هزینه gas شبکه در نظر گرفته می شود.
- data:فیلد اختیاری، از نوع `String`.
- nonce:فیلد اختیاری، از نوع `Number`. یک عدد صحیح است به ما اجازه میدهد تراکنشی را که هنوز در حال انجام است جایگزین کنیم با تراکنشی که دارای همان عدد nonce باشد.

برای ارسال اتر به یک حساب دیگر تنها نیاز دارید ارسال کننده، دریافت کننده و مقدار را مشخص کنید. مانند عبارت زیر:

```
> eth.sendTransaction({to: eth.accounts[1], from: eth.accounts[0], value:
100})
```

در هنگام اجرای این دستور با خطای `Error: authentication needed` مواجه می شوید. بنابراین می بایست اصطلاحا حسابی که می خواهید با آن اتر ارسال کنید یعنی تراکنش انجام دهید را unlock کنید با وارد کردن رمز عبور به صورت زیر:

```
> personal.unlockAccount(eth.coinbase)
```

```
Unlock account 0xabc...
true
```

حال می توانید دستور اجرای تراکنش را مجددا اجرا کنید. بعد از انجام تراکنش موجودی حساب ۱ را چک کنید:

```
> eth.getBalance(eth.accounts[1])
0
```

مشاهده می کنید که موجودی حساب صفر است به این خاطر که تراکنش انجام شده هنوز ماین نشده و داخل بلاکچین ثبت نشده است. با اجرای دستور ماینینگ و سپس توقف آن بعد از ماین یک بلاک حالا می توانیم دوباره موجودی حساب ۱ را چک کنیم:

```
> eth.getBalance(eth.accounts[1])
100
```

موجودی حساب می بایست مقدار ۱۰۰ باشد.