

Crypto Currencies CA#2

**Shiva Vafadar
810899074**

بخش دوم :

کام اول :

چون سیستم بنده macOS میباشد، طبق مراحل لینک قرار داده شده، ابتدہ homebrew را نصب کردم و سپس geth را نصب کردم.

```
shivavafadar@Shivas-MBP ~ % brew -v
zsh: command not found: brew
shivavafadar@Shivas-MBP ~ % /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
==> Checking for `sudo` access (which may request your password)...
Password:
==> This script will install:
/opt/homebrew/bin/brew
/opt/homebrew/share/doc/homebrew
/opt/homebrew/share/man/man1/brew.1
/opt/homebrew/share/zsh/site-functions/_brew
/opt/homebrew/etc/bash_completion.d/brew
/opt/homebrew
==> The following new directories will be created:
/opt/homebrew/bin
/opt/homebrew/etc
/opt/homebrew/include
/opt/homebrew/lib
/opt/homebrew/sbin
/opt/homebrew/share
/opt/homebrew/var
/opt/homebrew/opt
/opt/homebrew/share/zsh
```

```
shivavafadar@Shivas-MBP ~ % brew -v
Homebrew 4.3.1
shivavafadar@Shivas-MBP ~ % brew tap ethereum/ethereum
==> Tapping ethereum/ethereum
Cloning into '/opt/homebrew/Library/Taps/ethereum/homebrew-ethereum'...
remote: Enumerating objects: 10750, done.
remote: Counting objects: 100% (249/249), done.
remote: Compressing objects: 100% (154/154), done.
remote: Total 10750 (delta 144), reused 180 (delta 91), pack-reused 10501
Receiving objects: 100% (10750/10750), 1.46 MiB | 804.00 KiB/s, done.
Resolving deltas: 100% (5663/5663), done.
Tapped 7 formulae (20 files, 1.8MB).
shivavafadar@Shivas-MBP ~ % brew install ethereum
==> Downloading https://ghcr.io/v2/homebrew/core/ethereum/manifests/1.14.3
curl: (28) Failed to connect to ghcr.io port 443 after 76076 ms: Operation timed out
Warning: Problem : timeout. Will retry in 1 seconds. 3 retries left.
#####
 100.0%
==> Fetching ethereum
==> Downloading https://ghcr.io/v2/homebrew/core/ethereum/blobs/sha256:9149d5039
#####
 100.0%
==> Pouring ethereum--1.14.3.arm64 Ventura.bottle.tar.gz
🍺 /opt/homebrew/Cellar/ethereum/1.14.3: 19 files, 256.8MB
==> Running `brew cleanup ethereum`...
```

```

shivavafadar@Shivas-MBP ~ % brew update
brew upgrade
brew reinstall ethereum
==> Updating Homebrew...
Already up-to-date.
==> Downloading https://ghcr.io/v2/homebrew/core/ethereum/manifests/1.14.3
Already downloaded: /Users/shivavafadar/Library/Caches/Homebrew/downloads/b4be7b
7e42ed1dd16e1562cb53de4c848c0db5428ad90e9e08fd649951dc66a4--ethereum-1.14.3.bott
le_manifest.json
==> Fetching ethereum
==> Downloading https://ghcr.io/v2/homebrew/core/ethereum/blobs/sha256:9149d5039
Already downloaded: /Users/shivavafadar/Library/Caches/Homebrew/downloads/d1ac34
4844ec6c95e0576e4ef228837eb9a702c93d197406870f4091824cf365--ethereum--1.14.3.arm
64_ventura.bottle.tar.gz
==> Reinstalling ethereum
==> Pouring ethereum--1.14.3.arm64_ventura.bottle.tar.gz
🍺 /opt/homebrew/Cellar/ethereum/1.14.3: 19 files, 256.8MB
==> Running `brew cleanup ethereum`...
Disable this behaviour by setting HOMEBREW_NO_INSTALL_CLEANUP.
Hide these hints with HOMEBREW_NO_ENV_HINTS (see `man brew`).

```

```

shivavafadar@Shivas-MBP ~ % geth --help
NAME:
  geth - the go-ethereum command line interface

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

VERSION:
  1.14.3-stable

COMMANDS:
  account           Manage accounts
  attach            Start an interactive JavaScript environment (connect t
o node)
  console           Start an interactive JavaScript environment
  db                Low level database operations
  dump              Dump a specific block from storage
  dumpconfig        Export configuration values in a TOML format
  dumpgenesis       Dumps genesis block JSON configuration to stdout
  export            Export blockchain into file
  export-history    Export blockchain history to Era archives
  import            Import a blockchain file
  import-history   Import an Era archive
  import-preimages  Import the preimage database from an RLP stream

```

از دستور help، میتوان ب موارد زیر اشاره کرد :

.1. account: مدیریت حسابهای کاربری اتر. شامل ایجاد، حذف، قفل کردن و باز کردن قفل حسابها.

.2. attach: شروع یک محیط تعاملی JavaScript و اتصال به نود (گره) اتریوم برای اجرای دستورات.

.3. console: شروع یک محیط تعاملی JavaScript بدون نیاز به اتصال به نود.

.4. db: انجام عملیات‌های سطح پایین پایگاه داده، شامل عملیات‌های بازیابی و مدیریت داده‌ای پایگاه داده.

.5. dump: استخراج و نمایش یک بلاک خاص از حافظه.

.6. dumpconfig: صادرات تنظیمات نرم‌افزار به قالب TOML.

.JSON: نمایش پیکربندی بلاک جنسیس به صورت dumpgenesis .7

8. export: صادرات بلاکچین به یک فایل.

9. Era: صادرات تاریخچه بلاکچین به بایگانی .Era

10. import: وارد کردن یک فایل بلاکچین.

11. Era: وارد کردن یک بایگانی .Era

12. import-preimages: وارد کردن پایگاه داده پیش تصاویر از یک جریان RLP.

13. init: بوتاسترپ و راه اندازی یک بلاک جنسیس جدید.

14. js: (نسخه شده) اجرای فایلهای JavaScript مشخص شده.

15. license: نمایش اطلاعات مجوز نرم افزار.

16. removedb: حذف بلاکچین و پایگاه دادهای وضعیت.

17. show-deprecated-flags: نمایش فلگهایی که منسوخ شده اند.

18. snapshot: مجموعه ای از دستورات مبتنی بر عکس فوری (snapshot).

19. verkle: مجموعه ای از دستورات مدیریت درخت ورکل تجربی.

20. version: نمایش شماره نسخهای نرم افزار.

21. version-check: بررسی آنلاین برای آسیب پذیری های امنیتی شناخته شده Geth.

22. wallet: مدیریت کیف پولهای پیش فروش اتریوم.

23. help, h: نمایش لیست دستورات یا کمک برای یک دستور خاص.

کام دوم :

```
shivavafadar@Shivas-MBP ~ % {
  "config": {
    "chainId": 15,
    "homesteadBlock": 0,
    "eip155Block": 0,
    "eip158Block": 0
  },
  "difficulty": "400000",
  "gasLimit": "2100000",
  "alloc": {
    "[Account #1 Address)": { "balance": "1000000000810899074" },
    "[Account #2 Address)": { "balance": "2000000000810899074" },
    "[Account #3 Address)": { "balance": "1500000000810899074" }
  }
}
```

کام سوم :
mkdir node01 node02 node03

سوال ۱ :

کارکرد هر نود در یک شبکه رمزارز

تایید تراکنشها: نودها تراکنشهای جدید را دریافت کرده و صحت آنها را بررسی می‌کنند. این شامل اطمینان از صحبت امضاهای دیجیتال و موجود بودن موجودی کافی در حساب فرستنده است.

ذخیره بلاکچین: نودها یک کپی کامل یا جزئی از بلاکچین را نگهداری می‌کنند و با دیگر نودها در شبکه همگام می‌شوند تا اطمینان حاصل شود که تمامی نسخهای یکسان و بهروز هستند.

مشارکت در اجماع: نودها در فرایند اجماع شبکه شرکت می‌کنند تا در مورد اضافه کردن بلاکهای جدید به بلاکچین توافق کنند. بسته به نوع الگوریتم اجماع (مانند اثبات کار یا اثبات سهام)، نقش نودها متفاوت خواهد بود.

پخش اطلاعات: نودها اطلاعات مربوط به تراکنشها و بلاکها را به سایر نودها پخش می‌کنند و از این طریق اطلاعات جدید را در سراسر شبکه منتشر می‌کنند.

Full Node

یک نود کامل (Full Node) تمامی وظایف ذکر شده را انجام می‌دهد و علاوه بر آن:

ذخیره‌سازی کل بلاکچین: یک نود کامل دارای یک کپی کامل از تمام بلاکهای بلاکچین از آغاز تا به امروز است.

تایید مستقل تراکنشها و بلاکها: نود کامل قادر است تراکنشها و بلاکهای جدید را به طور مستقل تایید و بررسی کند.

بالا بردن امنیت شبکه: نودهای کامل نقش مهمی در امنیت و عدم تمرکز شبکه ایفا می‌کنند. هر چه تعداد نودهای کامل بیشتر باشد، شبکه مقاومت و امنیت خواهد بود.

پشتیبانی از سایر نودها: نودهای کامل می‌توانند به نودهای سبک (Light Nodes) کمک کنند تا اطلاعات تراکنشها و بلاکها را دریافت و تایید کنند.

Light Node

یک نود سبک (Light Node) عملکردی کم‌جهت و ساده‌تر دارد:

ذخیره‌سازی جزئی بلاکچین: نودهای سبک به جای نگهداری کل بلاکچین، تنها بخشی از آن (مانند هدرهای بلاکها) را نگهداری می‌کنند.

وابستگی به نودهای کامل: نودهای سبک برای تایید و بررسی تراکنشها به نودهای کامل متکی هستند. آنها درخواست اطلاعات جزئی و تاییدها را از نودهای کامل می‌کنند.

کاهش منابع مصرفی: نودهای سبک به منابع کمتری نیاز دارند و می‌توانند روی دستگاههای کمقدرت مانند گوشی‌های هوشمند یا کامپیوترهای شخصی اجرا شوند.

مناسب برای کاربران معمولی: نودهای سبک برای کاربرانی که نیاز به تایید مستقیم تراکنشها ندارند و بیشتر برای تراکنشهای روزمره و کاربردهای سبک از رمزارزها استفاده می‌کنند، مناسب هستند.

Light Node و Full Node مقایسه

حجم داده‌ها: نودهای کامل حجم زیادی از داده‌ها (کل بلاکچین) را نگهداری می‌کنند، در حالی که نودهای سبک تنها داده‌ای ضروری را ذخیره می‌کنند.

استقلال: نودهای کامل به طور مستقل کار می‌کنند و می‌توانند تراکنشها و بلاکها را بدون نیاز به نودهای دیگر تایید کنند. نودهای سبک برای این کار به نودهای کامل نیاز دارند.

امنیت و اعتماد: نودهای کامل امنیت بیشتری به شبکه اضافه می‌کنند، زیرا به دلیل استقلال و نگهداری کل بلاکچین، احتمال تقلب و حملات کمتر است. نودهای سبک به دلیل وابستگی به نودهای کامل، کمی کمتر امن هستند اما هنوز برای اکثر کاربردها کافی‌اند.

در نتیجه، انتخاب بین نود کامل و نود سبک بستگی به نیازها و منابع کاربر دارد. نودهای کامل برای کاربرانی که می‌خواهند در امنیت و تایید تراکنشها نقش فعالی داشته باشند، مناسب هستند، در حالی که نودهای سبک برای کاربرانی که به دنبال راه حلی ساده و کم‌جهت برای استفاده از شبکه رمزارز هستند، کافی می‌باشند.

گام چهارم:

```
[sh-3.2# geth --datadir "~/node01" account new
INFO [05-24|14:15:08.513] Maximum peer count                                     ETH=50 total=50
Your new account is locked with a password. Please give a password. Do not forget this
password.
>Password:
[Repeat password:

Your new key was generated

Public address of the key: 0xe58308a5a108679bc70b5286Fc2fFcc96ACA3A8F
Path of the secret key file: /var/root/node01/keystore/UTC--2024-05-24T10-45-19.342093
000Z--e58308a5a108679bc70b5286fc2ffcc96aca3a8f

- You can share your public address with anyone. Others need it to interact with you.
- You must NEVER share the secret key with anyone! The key controls access to your fun
ds!
- You must BACKUP your key file! Without the key, it's impossible to access account fu
nds!
- You must REMEMBER your password! Without the password, it's impossible to decrypt th
e key!
```

```
{  
  "config": {  
    "chainId": 15,  
    "homesteadBlock": 0,  
    "eip155Block": 0,  
    "eip158Block": 0  
  },  
  "difficulty": "400000",  
  "gasLimit": "2100000",  
  "alloc": {  
    "[Account #1 0xe58308a5a108679bc70b5286Fc2fFcc96ACA3A8F]": { "balance": "10000000000810899074" },  
    "[Account #2 0x149869e18fcffd5862C5A541643C8062a35dc259)": { "balance": "20000000000810899074" },  
    "[Account #3 0x6a9740f4ED8F33F63DD9F77c162224DfB7B055D3)": { "balance": "15000000000810899074" }  
  }  
}
```

گام چهارم initialized کردن نودها) :

```

INFO [05-24|17:24:40.258] Maximum peer count
INFO [05-24|17:24:40.261] Set global gas cap
INFO [05-24|17:24:40.261] Initializing the KZG library
INFO [05-24|17:24:40.277] Allocated trie memory caches
dirty=256.00MiB
INFO [05-24|17:24:40.278] Using pebble as the backing database
INFO [05-24|17:24:40.278] Allocated cache and file handles
ivavafadar/node01/geth/chaindata cache=512.00MiB handles=5120
INFO [05-24|17:24:40.326] Opened ancient database
ivavafadar/node01/geth/chaindata/ancient/chain readonly=false
INFO [05-24|17:24:40.326] State scheme set to already existing
Fatal: Failed to register the Ethereum service: only PoS networks are supported, please transition old ones with Geth v1.13.x
shivavafadar@Shivas-MBP ~ %

```

Your new key was generated

Public address of the key: **0xb119ECA59dE97Fab9CE02A1c6a612533D5a108E2**
Path of the secret key file: /Users/shivavafadar/Projects/project3/node01/keystores/UTC--2024-05-27T07-36-29.490854000Z--b119eca59de97fab9ce02a1c6a612533d5a108e2

- You can share your public address with anyone. Others need it to interact with you.
- You must NEVER share the secret key with anyone! The key controls access to your funds!
- You must BACKUP your key file! Without the key, it's impossible to access account funds!
- You must REMEMBER your password! Without the password, it's impossible to decrypt the key!

```

shivavafadar@Shivas-MacBook-Pro project3 % geth --identity "MyNode1" --http --http.port "8000" --http.corsdomain "*" --datadir "/Users/shivavafadar/Projects/project3/node01" --port "30303" --nodiscover --http.api "db,eth,net,web3,personal,miner,admin" --networkid 1900 --nat "any"

INFO [05-27|11:56:10.277] Maximum peer count
INFO [05-27|11:56:10.282] Set global gas cap
INFO [05-27|11:56:10.282] Initializing the KZG library
INFO [05-27|11:56:10.315] Allocated trie memory caches
cleanse=154.00MiB dirty=256.00MiB
INFO [05-27|11:56:10.315] Defaulthing to pebble as the backing database
INFO [05-27|11:56:10.315] Allocated cache and file handles
database=/Users/shivavafadar/Projects/project3/node01/geth/chaindata cache=512.00MiB handles=5120
INFO [05-27|11:56:10.399] Opened ancient database
database=/Users/shivavafadar/Projects/project3/node01/geth/chaindata/ancient/chain readonly=false
INFO [05-27|11:56:10.400] State scheme set to default
scheme=hash
INFO [05-27|11:56:10.400] Initiating peer-to-peer protocol
version=1900 diversion=<nil>
INFO [05-27|11:56:10.406] Writing default main-net genesis block
INFO [05-27|11:56:10.721] Persisted trie from memory database
nodes=12356 size=1.79MiB time=140.027208ms gonode=0 gcslice=0.00B gctime=0s livenodes=0 livesize=0.00B
INFO [05-27|11:56:10.789]
INFO [05-27|11:56:10.789] -----
INFO [05-27|11:56:10.789] Chain ID: 1 (mainnet)
INFO [05-27|11:56:10.789] Consensus: Beacon (proof-of-stake), merged from Ethash (proof-of-work)
INFO [05-27|11:56:10.789]
INFO [05-27|11:56:10.789] Pre-Merge hard forks (block based):
INFO [05-27|11:56:10.789] - Homestead:
#1150000 (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/homestead.md)
INFO [05-27|11:56:10.789] - DAO Fork:
#1920000 (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/dao-fork.md)
INFO [05-27|11:56:10.789] - Tangerine Whistle (EIP 150):
#2463000 (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/tangerine-whistle.md)
INFO [05-27|11:56:10.789] - Spurious Dragon/I (EIP 155):
#2605000 (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/spurious-dragon.md)
INFO [05-27|11:56:10.789] - Spurious Dragon/II (EIP 158):
#26575000 (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/spurious-dragon-2.md)
INFO [05-27|11:56:10.789] - Byzantium:
#4370000 (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/byzantium.md)
INFO [05-27|11:56:10.789] - Constantinople:
#7280000 (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/constantinople.md)
INFO [05-27|11:56:10.789] - Petersburg:
#7280000 (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/petersburg.md)
INFO [05-27|11:56:10.789] - Istanbul:
#9069000 (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/istanbul.md)
INFO [05-27|11:56:10.789] - Muir Glacier:
#9200000 (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/muir-glacier.md)
INFO [05-27|11:56:10.789] - London:
#12240000 (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/london.md)
INFO [05-27|11:56:10.789] - Arrow Glacier:
#13980000 (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/arrow-glacier.md)
INFO [05-27|11:56:10.789] - Gray Glacier:
#15050000 (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/gray-glacier.md)
INFO [05-27|11:56:10.789]
INFO [05-27|11:56:10.789] Merge configured:
INFO [05-27|11:56:10.789] - Hard-fork specification: https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/paris.md
INFO [05-27|11:56:10.789] - Network known to be merged: true
INFO [05-27|11:56:10.789] - Total terminal difficulty: 58750000000000000000000000000000
INFO [05-27|11:56:10.789]
INFO [05-27|11:56:10.789] Post-Merge hard forks (timestamp based):
INFO [05-27|11:56:10.789] - Shanghai:
#1681338455 (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/shanghai.md)
INFO [05-27|11:56:10.789] - Cancun:
#1710338135 (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgrades/cancun.md)
INFO [05-27|11:56:10.789]
INFO [05-27|11:56:10.790] -----
INFO [05-27|11:56:10.790] Loaded most recent local block
number=0 hash=d4e567..cb8fa1 td=17,179,869,184 age=55y2mo1w
WARN [05-27|11:56:10.790] Failed to load snapshot
err="missing or corrupted snapshot"
INFO [05-27|11:56:10.799] Rebuilding state snapshot
INFO [05-27|11:56:10.803] Initialized transaction indexer
INFO [05-27|11:56:10.803] Range: last 2350000 blocks"
INFO [05-27|11:56:10.803] range="last 2350000 blocks"
INFO [05-27|11:56:10.803] root=d7f897..0f0544 accounts=0 slots=0 storage=0.00B dangling=0 elapsed=4.373ms
INFO [05-27|11:56:10.803] accounts=0 slots=0 storage=409.64KiB dangling=0 elapsed=93.91ms
INFO [05-27|11:56:10.893] Generated state snapshot
INFO [05-27|11:56:10.977] Enabled snap sync
INFO [05-27|11:56:10.977] Chain post-merge, sync via beacon client
INFO [05-27|11:56:10.978] Gasprice oracle is ignoring threshold set threshold=2
INFO [05-27|11:56:10.983] Engine API enabled
INFO [05-27|11:56:10.983] Starting peer-to-peer node
INFO [05-27|11:56:10.983] Peer-to-peer open
INFO [05-27|11:56:11.013] Unavailable modules in HTTP API list
INFO [05-27|11:56:11.014] Generated JWT secret
INFO [05-27|11:56:11.015] New HTTP server started
INFO [05-27|11:56:11.015] New local node record
INFO [05-27|11:56:11.015] Started P2P networking
0.0.1:30303!tcpport=0
INFO [05-27|11:56:11.019] WebSocket enabled
url=wss://127.0.0.1:30303

```

```
[shivavafadar@Shivas-MacBook-Pro ~ % geth attach http://127.0.0.1:8000
WARN [05-27|11:58:21.525] Enabling deprecated personal namespace
Welcome to the Geth JavaScript console!

instance: Geth/MyNode1/v1.13.15-stable-c5ba367e/darwin-amd64/go1.21.6
at block: 0 (Thu Jan 01 1970 03:30:00 GMT+0330 (+0330))
datadir: /Users/shivavafadar/Projects/project3/node01
modules: admin:1.0 eth:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 web3:1.0

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

```
> admin.nodeInfo
{
  enode: "enode://01f65f1ea872e9a2de8d023f7910d8d253429dd56b235b24ae7ac4800f23ca45f6ca5abffcd820c5f29e45313f126f6225f510b8e1647af2faaacb9e94d06b8127.0.0.1:30303?discport=0",
  enr: "-I=4QGIdQJNoyRF4UEg9hsYY8JXplImbwg9x14Fn-XOn_e8gRpQuT_9Rg2hUHayIiHhdZG@a78wCsooseRcaAkAhQsCGAY-5KDTeg2V@aMrJhPxk7ASDEYwwgmlkgny0gm!whH8AAAGj2VjcD11MsxaQM8918epocumi3a0CP3kQ8NJTQp3VayNbJKS6xIAPI8pIRzbmFwW1N9Y3Ccd18",
  id: "639f567cd9684f8f26cb536495d50718ef50b28250c69e3507373c01beb81d",
  ip: "127.0.0.1",
  listenAddr: "[::]:30303",
  name: "Geth/MyNode1/v1.13.15-stable-c5ba367e/darwin-amd64/go1.21.6",
  ports: {
    discovery: 0,
    listener: 30303
  },
  protocols: {
    eth: {
      config: {
        arronGlacierBlock: 13773000,
        berlinBlock: 12244000,
        byzantiumBlock: 4378000,
        cancanTime: 1710338135,
        chainid: 1,
        constantinopleBlock: 7280000,
        dawsonkebBlock: 1920000,
        daoForkSupport: true,
        eip158Block: 2463000,
        eip158Block: 2675000,
        eip158Block: 2675000,
        ethash: {},
        grayFujisBlock: 10000000,
        homesteadBlock: 11500000,
        istanbulBlock: 90659000,
        londonBlock: 12965000,
        muirGlacierBlock: 9200000,
        petersburgBlock: 7280000,
        shanghaiTime: 1681338455,
        terminalTotalDifficulty: 5.875e+22,
        terminalTotalDifficultyPassed: true
      },
      difficulty: 17179869184,
      genesis: "0xd4e56740f876afe8c010b86a40d5f56745a118d0906a34e69aec8c0db1cb8fa3",
      head: "0xd4e56740f876afe8c010b86a40d5f56745a118d0906a34e69aec8c0db1cb8fa3",
      network: 1900
    }
  },
  snap: {}
}
>
```

```
Last login: Mon May 27 13:36:36 on ttys004
[shivavafadar@Shivas-MacBook-Pro ~ % geth attach http://127.0.0.1:8008
Welcome to the Geth JavaScript console!

instance: Geth/MyNode2/v1.13.15-stable-c5ba367e/darwin-amd64/go1.21.6
at block: 0 (Thu Jan 01 1970 03:30:00 GMT+0330 (+0330))
modules: eth:1.0 net:1.0 rpc:1.0 web3:1.0

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

```
londonBlock: 12965000,
muirGlacierBlock: 9200000,
petersburgBlock: 7280000,
shanghaiTime: 1681338455,
terminalTotalDifficulty: 5.875e+22,
terminalTotalDifficultyPassed: true
},
difficulty: 17179869184,
genesis: "0xd4e56740f876aef8c010b86a40d5f56745a118d0906a34e69aec8c0db1cb8fa3",
head: "0xd4e56740f876aef8c010b86a40d5f56745a118d0906a34e69aec8c0db1cb8fa3"
,
network: 1
},
snap: {}
}
]
> admin.addPeer("enode://01f65f1ea6872e9a2de8d023f7910d0d253429dd56b235b24ae7ac4800f23ca45f6ca6abffcdb420c5f29e45313f126f6225f510b8e1647af2faacab9e04d06b@127.0.1:30303?discport=0")
true
> net.peerCount
2
> 
```

```
[> eth.accounts
["0x0ce15bb697ca278c9498bc56f5bfce899568dd6f"]
> 
```

```
[> eth.accounts
["0x0ce15bb697ca278c9498bc56f5bfce899568dd6f"]
[> eth.getBalance(eth.accounts[0])
0
[> personal.unlockAccount(eth.accounts[0])
Unlock account 0x0ce15bb697ca278c9498bc56f5bfce899568dd6f
[Passphrase:
true
> 
```

```
[> miner.setEtherbase(eth.accounts[0])
true
[> miner.start()
null
> 
```

سوال : ۲

این دستور حسابی را تنظیم می‌کند که پاداش‌های استخراج به آن اختصاص داده می‌شود. [۰] `eth.accounts[0]` به اولین حساب در لیست حسابهای مدیریت شده توسط نود اتریوم شما اشاره دارد. خروجی `true` نشان می‌دهد که اتریس با موفقیت به این حساب تنظیم شده است.

نشان‌دهنده خطأ نیست؛ برای دستورات کنسول `null` دستور دوم، فرایند استخراج را در نود اتریوم شما آغاز می‌کند. خروجی صرفاً نشان می‌دهد که `null` معمول است. در این زینه `null` نود که نتیجه مستقیم یا خروجی خاصی ندارند، خروجی دستور دریافت شده و فرایند استخراج بدون خروجی خاصی آغاز شده است.

سوال : ۳

نه، شما نمی‌توانید با ایجاد ۰۰۰.۷۲۲.۱۹ بلاک در یک بلاکچین محلی و سپس تلاش برای پخش آن در شبکه اینترنت، بلاکچین اتریوم اصلی را جایگزین کنید. دلایل اصلی که مانع از انجام این کار می‌شوند، شامل موارد زیر می‌باشند:

۱. **اجماع شبکه (Network Consensus):** بلاکچین‌ها مانند اتریوم بر اساس مکانیزم اجماع کار می‌کنند. این به این معنی است که برای اینکه یک بلاک به زنجیره بلاکها اضافه شود، باید توسط اکثریت نودهای شبکه تأیید شود. بلاکچین شما که در محیط محلی ایجاد شده و قادر تأییدات و تاریخچه معتبر نودهای دیگر است، به راحتی توسط نودهای دیگر در شبکه رد خواهد شد.

۲. **طول زنجیره و کار محاسباتی (Proof of Work):** در شبکه‌های مبتنی بر اثبات کار (PoW)، زنجیره با بیشترین کار محاسباتی انجام شده (یعنی بیشترین تجمع تلاش ماینینگ) به عنوان زنجیره معتبر شناخته می‌شود. ایجاد ۰۰۰.۷۲۲.۱۹ بلاک به صورت معتبر، نیازمند توان محاسباتی بسیار زیادی است که در یک محیط محلی تقریباً غیرممکن است.

۳. **معیارهای امنیتی و اعتبارسنجی:** نودهای شبکه اتریوم بلاکهای جدید را بر اساس پروتکلهای دقیق و مشخصی بررسی می‌کنند که شامل اعتبارسنجی تراکنش‌ها، تأیید امضاهای رعایت قوانین شبکه است. هر تلاش برای پخش بلاکچینی با داده‌های ساختگی یا دستکاری شده به سرعت تشخیص داده شده و رد می‌شود.

۴. **همگام‌سازی بلاکها و تاریخچه تراکنش‌ها:** برای اینکه بلاکچین جدیدی به عنوان جایگزین بلاکچین فعلی پذیرفته شود، باید تمام تاریخچه تراکنش‌های قبلی را که بر روی شبکه اصلی ثبت شده‌اند، شامل شود و به درستی همگام‌سازی شود. این کار نیز در یک محیط محلی بدون دسترسی به داده‌های کامل شبکه، غیرممکن است.

به طور خلاصه، تلاش برای جایگزین کردن بلاکچین اتریوم اصلی با یک بلاکچین محلی ایجاد شده، نه تنها فنی ناممکن است بلکه به دلیل مکانیزم‌های امنیتی و اجماع موجود در شبکه، عمل ناکارآمد است.

سوال 1: الگوریتم اجماع Proof of Burn و تأثیر اقتصادی آن

(اثبات سوزاندن) یک الگوریتم اجماع است که در آن کوینها به طور داوطلبانه توسط کاربران سوزانده می‌شوند تا حق ماین کردن بلاکها یا شرکت در فرایندهای تصمیم‌گیری در شبکه را به دست آورند. در این روش، کوین‌هایی به آدرس‌هایی ارسال می‌شوند که قابل دسترسی نیستند و به این ترتیب، به طور دائم از چرخه عرضه خارج می‌شوند.

تأثیرات اقتصادی:

1. کاهش عرضه: سوزاندن کوینها به طور موثری عرضه آنها را کاهش می‌دهد. این کاهش عرضه می‌تواند به افزایش ارزش باقیمانده کوینها منجر شود، به شرطی که تقاضا ثابت بماند یا افزایش یابد.
2. تحریک سرمایه‌گذاری: با سوزاندن کوینها، کاربران نشان می‌دهند که به آینده شبکه اعتقاد دارند و حاضرند برای اطمینان از امنیت و پایداری آن سرمایه‌گذاری کنند.
3. پیشگیری از حملات: از آنجایی که سوزاندن کوینها هزینه واقعی برای ماینرها ایجاد می‌کند، احتمال حملات مخرب بر شبکه کاهش می‌یابد، زیرا مهاجمان نیز باید هزینه قابل توجهی بپردازنند.

سوال 2: زبان Scripting بیتکوین و تورینگ کامل بودن

زبان **Scripting بیتکوین** یک زبان برنامه‌نویسی ساده است که برای تعریف شرایط استفاده از بیتکوینها در تراکنش‌ها استفاده می‌شود. این زبان **Turing Incomplete** است، به این معنا که امکان اجرای تمام عملیات‌های ممکن که یک زبان **Turing Complete** مانند JavaScript یا Python را ندارد.

عدم تورینگ کامل بودن به دلایل امنیتی است. به این ترتیب، جلوی اجرای حلقات یا برنامه‌هایی که می‌توانند به نامحدود اجرا شوند و منابع شبکه را مصرف کنند، گرفته می‌شود. این محدودیت از حملات حجمی (Denial-of-service attacks) که می‌توانند شبکه را فلک کنند، جلوگیری می‌کند.

مزایای استفاده از زبان Turing Complete

1. انعطاف‌پذیری بیشتر: زبان‌های برنامه‌نویسی تورینگ کامل امکان نوشتگر برنامه‌های پیچیده‌تر و کاربردی‌تر را فراهم می‌کنند.
 2. پیاده‌سازی قراردادهای هوشمند: قراردادهای هوشمند نیازمند زبان‌های برنامه‌نویسی هستند که قابلیت اجرای عملیات‌های متعدد و پیچیده را داشته باشند.
- در حالی که بیتکوین از یک زبان ساده و ایمن استفاده می‌کند، پلتفرم‌های دیگر مانند اتریوم از زبان‌های Turing Complete مانند Solidity برای ایجاد قراردادهای هوشمند استفاده می‌کنند، که این امکان را می‌دهد تا قراردادها و عملیات‌های مالی پیچیده‌تری پیاده‌سازی شوند.