Innovating With Bitcoin

Joe Bender - Developer Evangelist

October 2020
I’m over four months into my Developer Evangelist journey at Blockstack. My focus is empowering our community to build robust tools & useful applications.

Favorite Quarantine Activity?

Browsing Bitcoin memes!
Overview

During this talk, I’ll explain:

1. Blockchain Basics
2. Bitcoin & Lightning
3. Blockstack
4. The Stacks 2.0 Testnet
5. Clarity Smart Contracts
6. Overview of Running a Node & Miner
7. Why Run a Node?
8. How to Get Involved
Catch-Up

Principle Foundations of Bitcoin
Shreemoy Mishra

How Ethereum Works
John Long
“No company on the internet should have so much power that they get to debate if they should be evil today or not..”

- Muneeb Ali, Co-Founder of Blockstack
What the heck is a Web 3.0?
Early Internet Decentralization
Web 1.0
read-only
static

Web 2.0
read-write
dynamic

Web 3.0
read-write-trust
verifiable
The best place to get started with crypto

coinbase
METAMASK

A crypto wallet & gateway to blockchain apps
First-Movers
First-Movers

**Coinbase**

- 30+ Million Users
- Password Login, Coinbase has Private Key
- Buy Crypto directly within app
- Implemented Staking Rewards (Tezos)
- Coinbase Pro & Coinbase Wallet
- Verification Levels
  - Level 1 - Phone #
  - Level 2 - Personal Info
  - Level 3 - Verify Photo ID

**Metamask**

- 1+ Million Extension Downloads
- Password Login, User stores seed phrase
- Recommends Wyre or CoinSwitch to purchase crypto
- Allows user to connect to testnets or local networks
- Mobile app added dApp browser
Networks & Nodes

- In telecommunication networks, **nodes** act as redistribution points or as a communication endpoints.
- Full node, master node, light node, miner, super node.
- Download particular software to your machine that outlines rules for connecting to **network**.

- **Node** = Building blocks of blockchain.
- **Blockchain** = blocks of data.
  - Data is stored in nodes.
- All **nodes** on a **blockchain** are connected to each other and they constantly exchange the latest blockchain data with each other so all nodes stay up to date.
Networks & Nodes

- In telecommunication networks, **nodes** act as redistribution points or as a communication endpoints.
- Full node, master node, light node, miner, super node.
- Download particular software to your machine that outlines rules for connecting to **network**.
- Node = Building blocks of blockchain.
- Blockchain = blocks of data.
  - Data is stored in nodes.
- All **nodes** on a **blockchain** are connected to each other and they constantly exchange the latest blockchain data with each other so all nodes stay up to date.
Enter, Bitcoin
Launched 03 January 2009 by “Satoshi Nakamato”
Bitcoin

- Launched 03 January 2009 by “Satoshi Nakamoto”
- First widespread implementation of decentralized cryptocurrency
- Current Market Cap > $200 Billion USD
Lightning Network
Lightning Network

- [https://lightning.network/](https://lightning.network/)
- Scalability solution.
- Transactions are sent over a network of micropayment channels.
- Moves small and frequent transactions off-chain, allowing for fast peer-to-peer transactions and low fees.
- Uses native smart-contract scripting language.
- Creates a two-party ledger entry.
- Bi-directional.
- Makes cross-chain atomic swaps possible so long as the chains can support the same cryptographic hash function.
- Channel closes upon completion.
Lightning Network vs. DeFi

12.1 Million USD

11.23 Billion USD
At Blockstack, we’re building the tools needed for a user-owned internet.
Blockstack is an open-source effort to develop software that provides an alternative to traditional (centralized) web applications.

We’ve developed a full-stack decentralized computing network that enables a new generation of applications where developers and users can interact fairly and securely.

Blockstack uses blockchain technology to build protocols and developer tools designed to enable a fair and open Internet that returns digital rights to developers and consumers.
At Blockstack, we believe Web 3 will introduce true internet ownership anchored to the most secure blockchain: **Bitcoin**.

Stacks 2.0 represents the design by which Web 3 can emerge and scale.

Designed with security, scalability, and speed in mind.

**Proof of Transfer (PoX):** A novel mining mechanism that leverages Bitcoin to secure a new network. PoX enables benefits not possible with just proof-of-work or proof-of-stake.

**Stacking:** A novel value transfer mechanism that we have proposed that allows Stacks holders to earn Bitcoin for actively participating in the consensus algorithm.

**Clarity:** A smart contract language that optimizes for predictability and security. Enables developers to write expressive smart contracts and experiment with new business models.
Smart Contracts

- Smart contracts **encode** and **enforce** rules for modifying a particular set of data that is shared among people and entities who don’t necessarily trust each other.
- Exist in a **blockchain**, anyone can query them, and anyone can submit transactions to execute them.
- A smart contract execution can result in new **transactions** being written to the blockchain.
- Apps can take advantage of smart contracts to manage a **global state** that is visible to the public.
- Anyone can **audit** the blockchain in order to independently verify that an app’s global shared state has been managed correctly according to the smart contracts’ rules.

**Use Cases**
- Access control (e.g. pay to access)
- Non-fungible (e.g. collectibles) and fungible tokens (e.g. stablecoins)
- Business model templates (e.g. subscriptions)
- App-specific blockchains
- Decentralized Autonomous Organizations
Clarity Programming Language

- Clarity is a programming language for writing smart contracts on the Stacks 2.0 blockchain.
- Differs from other SC languages:
  - **Interpreted**: Human-readable and auditable
  - **Decidable**: Determine precisely what code is being executed, for any function.
- Ability to write fully expressive smart contracts that anchor to Bitcoin.
- The Clarity language uses a strong static type system. LISP-based.
- A Clarity smart contract is composed of two parts — a data space and a set of functions.
  - Only the associated smart contract may modify its corresponding data space on the blockchain.
- Users call smart contracts’ public functions by broadcasting a transaction on the blockchain which invokes the public function.
- Function arguments and database schemas require specified types, and use of types is checked during contract launch.
These are impossible in Clarity

- Breaking news: 0x down.
- DAO Hack. $50M
- Multi-Sig hack(s). $200M
- Hundreds of txs aborting: ___$/ day.
- Audits are hard: [$_k: $_$_k] / contract.
Basic Example - Counter

1. define-data-var: initializes a new integer variable counter with the value set to 0.
   - The counter variable is stored in the data space associated with this particular smart contract.

2. define-public: provides access to the counter variable from outside of the current smart contract.
   - The var-get statement looks for a variable in the contract’s data space and returns it.

3. begin statement evaluates the multi-line expressions and returns the value of the last expression. In this case, it is used to set a new value and return the new value.
### Testing - Counter

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔️</td>
<td>Should be successfully deployed with valid syntax</td>
</tr>
<tr>
<td></td>
<td>Deploying an instance of the contract</td>
</tr>
<tr>
<td>✔️</td>
<td>Should start at zero</td>
</tr>
<tr>
<td>✔️</td>
<td>Should increment (133ms)</td>
</tr>
<tr>
<td>✔️</td>
<td>Should decrement (177ms)</td>
</tr>
</tbody>
</table>

4 passing (586ms)
Notice how the instance of the smart contract is created on line 8

- Where to find `counter.clar`

Creates a transaction query that tests the valid syntax.

Passes first test!
1. Gets the counter variable, checks it is equal to zero.
2. Checks if variable can be incremented by calling `increment` then `getCounter` twice.
3. Checks if variable can be decremented by calling `decrement` then `getCounter` twice.
(define-data-var meal-cost int 0)
(define-data-var tip int 0)
(define-data-var rating int 0)

(define-public (reserve-meal-cost (cost int))
  (ok
   (begin
    (var-set meal-cost cost)
    (calculate-tip)
    (var-get meal-cost))))

(define-public (get-meal-cost)
  (ok (var-get meal-cost)))

(define-public (get-tip-value)
  (ok (var-get tip)))

(define-public (get-rating)
  (ok (var-get rating)))

(define-public (finish-meal (mealRating int))
  (ok
   (begin
    (var-set rating mealRating)
    (calculate-tip)
    (var-get rating))))

;; if rating is greater than 3 then the user is very satisfied
;; if rating is less than or equals 3 then the user is dissatisfied
;; tip would be the minimum 15%
;; tip would be 20%
;; Support these workers they are literally putting their lives on the line
;; if you'd like to support organizations helping in the corona effort
;; help people get food through https://www.cityharvest.org/
;; I hope this doesn't disqualify me xD
(define-private (calculate-tip)
  (begin
   (if
    (> (var-get rating) 3)
     (var-set tip (/ (* (var-get meal-cost) 20) 100))
     (var-set tip (/ (* (var-get meal-cost) 15) 100))))
Types of Smart Contracts

- Simple exchange
- DAOs
- Dapp
- Counter
- Supply chain
- Deeds
- Access Restriction
- Withdrawals
- State Machines
- Balance Address Checker

- Contracts call another Contract
- Fundraising
- Simple Marketplace
- Basic Provenance
- Asset transfer
- Lottery
- Delegated Voting
- Smart contract IOT - devices
- Legal Agreements
- Payment Splitting
Stacks 2.0 Testnet Rollout

Testnet launch broken into four phases:
- Neon, Argon, Krypton, Xenon → Mainnet!

Benefits of testnet:
- Testnet $BTC & $STX.
  - No monetary value or attack incentive.
- Parallel network built for testing purposes.
  - Not losing value with txn fees.
- Let community tinker before final product.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Phase 1</th>
<th>Phase 2</th>
<th>Phase 3</th>
<th>Phase 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple Proof of Transfer mining</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Send and receive STX</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deploy Clarity contracts</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>New Stacks Explorer</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>stacks-transactions-js</td>
<td></td>
<td>●</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transaction signing</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Proof of Transfer mining</td>
<td></td>
<td>●</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stacking</td>
<td></td>
<td></td>
<td></td>
<td>●</td>
</tr>
<tr>
<td>New Stacks Wallet</td>
<td></td>
<td></td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Microblocks</td>
<td></td>
<td></td>
<td></td>
<td>●</td>
</tr>
<tr>
<td>Testing upgrade to Stacks 2.0</td>
<td></td>
<td></td>
<td></td>
<td>●</td>
</tr>
<tr>
<td>Integration with Bitcoin testnet</td>
<td></td>
<td></td>
<td></td>
<td>●</td>
</tr>
</tbody>
</table>
Testnet Activities

- Download the stacks-blockchain software and spin up a node.
- Turn your node into a miner and participate in PoX consensus.
- Create a wallet with the JavaScript CLI.
- Try out earning $BTC rewards with stacking.
- Build a Clarity smart contract and deploy it to testnet.
- Send testnet transactions between accounts.
- Play with the new testnet explorer to analyze transactions.
  - Sandbox: Deploy & Call Contracts, get testnet $STX, transfer $STX, and more.
- Submit bug bounties to earn $BTC.
Running a Testnet Node

Running a testnet node
Learn how to set up and run a Stacks 2.0 testnet node.

Introduction
The Stacks 2.0 testnet is currently in development. As part of the testnet, you can run a node and connect it to a public network. This guide will walk you through downloading and running your own node in the testnet network.

Prerequisites
Note: If you use Linux, you may need to manually install `libssl-dev` and other packages. In your command line, run the following to get all packages:

```
  sudo apt-get install build-essential cmake libssl-dev pkg-config
```

https://docs.blockstack.org/stacks-blockchain/testnet-node

Starting a Miner

Mine Stacks tokens
Set up and run a miner on the Stacks 2.0 testnet

Introduction
Make sure you’ve followed our guide for getting a Stacks 2.0 Testnet node up and running, once completed it’s only a few more steps to run a proof-of-burn miner on the testnet.

Running a miner
First, we need to generate a keychain. With this keychain, we’ll get some testnet BTC from a faucet, and then use that BTC to start mining.

https://docs.blockstack.org/mining
Prerequisites

- **Command Line Interface (CLI)**
  - Used to interact with the blockchain and enter commands.

- **RUST**
  - Programming language built for performance, reliability, and productivity. Needed for your machine to run node software successfully.

- **Linux Users: Libssl-dev**
  - API needed for Linux OS to process stacks-blockchain software.

- **Stacks-blockchain Github Repository**
  - Source code for running a node on the Stacks 2.0 testnet.

```bash
sudo apt-get install build-essential cmake libssl-dev pkg-config
```
Downloading RUST & stacks-blockchain

- Installing RUST
  ```shell
curl --proto 'https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
  ```
  1) Proceed with installation (default)
  2) Customize installation
  3) Cancel installation

- Installing ‘stacks-blockchain’
  - [https://github.com/blockstack/stacks-blockchain](https://github.com/blockstack/stacks-blockchain)
  - From Github Website

- From Command Line
  ```shell
git clone https://github.com/blockstack/stacks-blockchain.git
  ```
Installing stacks-blockchain

- Install the ‘stacks-blockchain’ software
  ```
cargo install --path ./testnet/stacks-node
  
  cd stacks-blockchain
  ```

- **Important!** Change your active directory to the stacks-blockchain folder.

- Start your node!
  ```
  stacks-node argon
  ```

- Evaluate Log Output

```
INFO [1588108047.585] [src/chainstate/stacks/index/marf.rs:732]

First-ever block 0f9188f13cb7b2c71f2a335e3a4fc328bf5beb436012afca590b1a11466e2206
```
Stacks-blockchain Output

Processing Blocks

Contract Call

INFO [1598225435.320] [src/net/relay.rs:800] [ThreadId(5)] Processing newly received blocks: 6
INFO [1598225435.741] [src/net/relay.rs:800] [ThreadId(5)] Processing newly received blocks: 6
INFO [1598225436.480] [src/net/relay.rs:800] [ThreadId(5)] Processing newly received blocks: 6
INFO [1598225436.856] [src/net/relay.rs:800] [ThreadId(5)] Processing newly received blocks: 6
INFO [1598225437.225] [src/net/relay.rs:800] [ThreadId(5)] Processing newly received blocks: 6
INFO [1598225437.694] [src/net/relay.rs:800] [ThreadId(5)] Processing newly received blocks: 6
INFO [1598225438.142] [src/net/relay.rs:800] [ThreadId(5)] Processing newly received blocks: 6
INFO [1598225438.545] [src/net/relay.rs:800] [ThreadId(5)] Processing newly received blocks: 6
INFO [1598225439.071] [src/net/relay.rs:800] [ThreadId(5)] Processing newly received blocks: 6
0x4576656e742120a8656c6c6f20776f726c64
INFO [1598225439.157] [src/chains/statestacks/db/transactions.rs:563] [ThreadId(5)] Contract-call 1 to ST8QVG6WC82C54AYPCVNN5VVQZVDCBD9S709VZ4Y.hello_world.ClarityName("set-value") args [Buffer (666f6f), Buffer(626172)] returned Response(ResponseData { committed: true, data: UInt(1) })
INFO [1598225439.157] [src/chains/statestacks/db/transactions.rs:564] [ThreadId(5)] Contract-call 1 to ST8QVG6WC82C54AYPCVNN5VVQZVDCBD9S709VZ4Y.hello_world.ClarityName("set-value") args [Buffer (666f6f), Buffer(626172)] cost ExecutionCost { write_length: 107, write_count: 1, read_length: 1573, read_count: 2, runtime: 1780 }
INFO [1598225439.184] [src/chains/statestacks/db/transactions.rs:563] [ThreadId(5)] Contract-call 1 to ST8QVG6WC82C54AYPCVNN5VVQZVDCBD9S709VZ4Y.hello_world.ClarityName("set-value") args [Buffer (666f6f), Buffer(626172)] returned Response(ResponseData { committed: true, data: UInt(1) })
INFO [1598225439.184] [src/chains/statestacks/db/transactions.rs:564] [ThreadId(5)] Contract-call 1 to ST8QVG6WC82C54AYPCVNN5VVQZVDCBD9S709VZ4Y.hello_world.ClarityName("set-value") args [Buffer (666f6f), Buffer(626172)] cost ExecutionCost { write_length: 107, write_count: 1, read_length: 1573, read_count: 2, runtime: 1780 }
Creating Keychain

- Create a keychain
  - With this keychain, we'll get some testnet BTC from a faucet, and then use that BTC to start mining.
- Simplest way is using ‘blockstack-cli’

```bash
npx blockstack-cli@1.1.0-beta.1 make_keychain -t
```

- After this runs, you'll probably see some installation logs, and at the end you should see some JSON that looks like this:

```json
```
```
```
Funding the miner

- Get your BTC address
  - The ‘btcAddress’ field from the JSON snippet

```json
"mnemonic": "exhaust spin topic distance hole december impulse gate century absent breeze ostrich armed clerk oak peace want scrap auction sniff cradle siren blur blur",
"privateKey": "2833269b55826ff2edda0f6de2e5693877fde9d697af8fe8f057bb5962894d5501",
"address": "STXZV6nCOMMSET72MC4CE4YQ9DDNTGCFX3KEtXK",
"btcAddress": "mrkRY7kPBjw3NjVz3HByqAvVz8c4B6ND",
"index": 0
```

- Request testnet BTC from the Stacks 2.0 Testnet Faucet
  - “Get testnet Bitcoin Tokens (BTC)” Field
  - [https://www.blockstack.org/testnet/faucet](https://www.blockstack.org/testnet/faucet)
  - You'll be sent 0.5 testnet BTC to that address. Don't lose this information - we'll need to use the ‘privateKey’ field later on.
Configure Node

- We have to configure the node so it knows to use the provided BTC keychain.
- In the ‘stacks-blockchain’ folder, create a new file called:
  ```toml
  [node]
  rpc_bind = "0.0.0.0:20443"
  p2p_bind = "0.0.0.0:20444"
  bootstrap_node = "048dd4f26101715853533"  # Enter your private key here!
  seed = "replace-with-your-private-key"
  miner = true
  ```
### Configuration File Explained

- **Configuring Burnchain**

```ini
[burnchain]
chain = "bitcoin"
mode = "argon"
peer_host = "argon.blockstack.org"
rpc_port = 18443
peer_port = 18444
```

- **Configuring Balances**

```ini
[[mstx_balance]]
address = "STB44HYPYAT2BB2Q5E513NS81HTMYWBPQ2HPGK6"
amount = 100000000000000000
[[mstx_balance]]
address = "ST11NJTTKGVT6D1HY4NJRQWMQM7TVAR091EJBPY2Y"
amount = 100000000000000000
[[mstx_balance]]
address = "ST1183WKNBYBOY377WZS89KKB3R3EG9EPJKR"
amount = 100000000000000000
[[mstx_balance]]
address = "STRYYQ9M8KAF4N5WNZQYY5993X6K31JP64CP"
amount = 100000000000000000
```
Start Running Your Miner!

Enter this command to begin mining:

```
stacks-node start --config=./testnet/stacks-node/conf/testnet-miner-conf.toml
```
Creating an Optimized Binary

- If you want to host a node on a server somewhere, you might want to generate an optimized binary.
  - Run this command to create the binary
    ```
    cd testnet/stacks-node
    cargo build --release --bin stacks-node
    ```
- Now that your optimized binary is compiled, go ahead and run it.
  ```
  cd ../../
  ./target/release/stacks-node start --config=./testnet/conf/argon-follower-conf.toml
  ```
Enable Debug Logging

- In case you are running into issues, or would like to see more detailed logging, you can run your node with debug logging enabled.
- In the command line, run:

```
BLOCKSTACK_DEBUG=1 stacks-node argon
```
Why Running a Node is Important

- **Trust**
  - To have a copy of the blockchain that you have validated yourself, rather than having to trust a third party to be honest about the state of the chain.

- **Control**
  - You do not depend on third parties for broadcasting your transactions to the network.

- **Scalability**
  - More nodes & miners = faster network!

- **It’s inexpensive!**
  - Unlike Proof-of-Work, Stacks 2.0 miners don’t need overpriced hardware to begin participating.

- **Increase Network Security**
  - By adding a node that can validate the state of the chain, you are actively making the blockchain safer.

- **Participate in Mining**
  - Running a node is the first step to mining.

- **Autonomy**
  - When a blockchain undergoes a hard fork, lightweight nodes will automatically follow the chain with the biggest accumulated difficulty.

- **It’s easy!**
  - All you need is a computer, and a few commands in the terminal.
How to get involved!

- Checkout our updated Documentation
  - https://docs.blockstack.org/
- Join the Forum
  - https://forum.blockstack.org/
- Join our Discord
  - http://discord.gg/unFGwwu
- Follow Blockstack on Twitter
  - https://twitter.com/blockstack
- Download the Stacks Wallet
  - https://wallet.blockstack.org/
- Read our Whitepapers
  - https://www.blockstack.org/papers
- Contribute Code
  - github.com/blockstack
- Host meetups, give talks, and spread the word!
  - https://community.blockstack.org/
- Complete Testnet Bounties
  - https://www.blockstack.org/testnet/bounties
- Participate in the Stacks 2.0 Hackathon Series
  - https://community.blockstack.org/stacks-series
- Sign-up for the Blockstack Newsletter
  - http://newsletters.blockstack.org/
- Experiment with the Stacks 2.0 Testnet
  - https://www.blockstack.org/testnet
<table>
<thead>
<tr>
<th>Topic</th>
<th>Replies</th>
<th>Views</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Faucet working?</td>
<td>0</td>
<td>14</td>
<td>11h</td>
</tr>
<tr>
<td>testnet</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Forum Maintenance (expect downtime of less than an hour)</td>
<td>1</td>
<td>27</td>
<td>17h</td>
</tr>
<tr>
<td>Community</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The Clarity Hackathon has begun!</td>
<td>0</td>
<td>28</td>
<td>22h</td>
</tr>
<tr>
<td>Blockstack + Flutter?</td>
<td>3</td>
<td>43</td>
<td>1d</td>
</tr>
<tr>
<td>Apps</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shared user data in a blockstack app</td>
<td>14</td>
<td>1.6k</td>
<td>1d</td>
</tr>
<tr>
<td>Apps</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>How do i Create Shared Data on Gaia Storage</td>
<td>5</td>
<td>330</td>
<td>1d</td>
</tr>
<tr>
<td>Storage (Gaia)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Documentation

All you need to build decentralized apps and smart contracts.

Get started

Building decentralized apps
Overview and guides for getting started building decentralized applications.

Write smart contracts
Overview and guides for getting started with Clarity

Mine Stacks tokens
Set up and run a miner on the Stacks 2.0 testnet
# HackStacks

SEPTEMBER 30 - NOVEMBER 11

#HackStacks is a virtual hackathon focused on leveraging the power of PoX, a novel, Bitcoin-secured mining mechanism enabling endless potential for new business models. PoX will enable mining and stacking, unique functionality that offers all stakeholders the opportunity to share in value creation on the Stacks 2.0 network. #HackStacks supports and rewards developers for building tools and products that make mining and stacking more accessible to the Stacks community.

# HackDeFi

OCTOBER 20 - NOVEMBER 20

Taking Defi to the next level with Bitcoin. Build safe, smart, secure DeFi projects anchored to Bitcoin by leveraging Proof of Transfer, Stacking, and Clarity smart contracts on the Stacks 2.0 testnet and reap the rewards.
● What do you think the biggest hurdle would be for a non-crypto enthusiast?
● What balance of blockchain education vs. obfuscation do you think is best?
● What will be the catalyst that drives the mainstream public to try crypto?
● Do you believe in the ‘Crypto already has a 1%’ assertion?
● Will digital assets eventually be valued in the same way as physical assets?
● Does blockchain need more developers, or better marketing?
● What decentralized application would you most desire implemented into your life?