

## CET 4007B -Blockchain Technology Unit 4 Ethereum Blockchains

### **Ethereum Blockchains**

Ethereum Virtual Machine, Smart contract, wallets for Ethereum, Ethereum Programming Language – Solidity, Mining in Ethereum, uses and benefits of Ethereum, Introduction to Ethereum Development Tools, Ethereum Clients, Ethereum Languages, Ethereum Wallets, Ethereum Accounts, Ethereum Keypairs, Ethereum Platform.

# Ethereum Blockchain Blockchain



Ethereum is a decentralized, open-source blockchain with smart contract functionality.



Ether is the native cryptocurrency of the platform.



After Bitcoin, it is the second-largest cryptocurrency by market capitalization. Ethereum is the most actively used blockchain

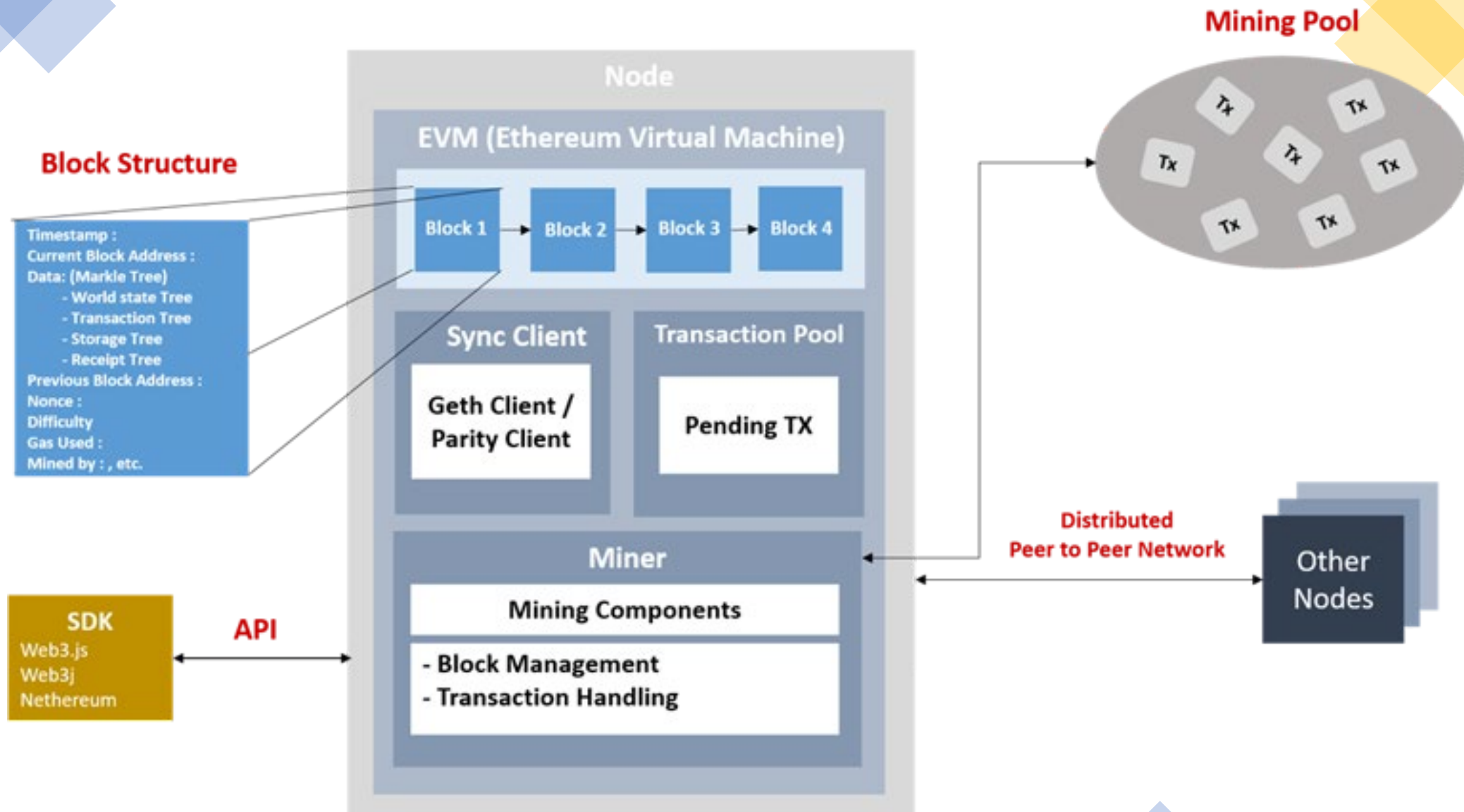
# Ethereum

---

Ethereum is a Powerful and Trustful Blockchain environment used to deploy Smart Contracts.

---

Vitalik Buterin is the co-founder and inventor of Ethereum, he describes it as a "decentralised mining network and software development platform rolled into one that facilitates create new crypto currencies and programs that share a single Blockchain".



# Ethereum Blocks / Components

- **Node/Client:** A node is a device/program that communicates with the Ethereum network. Nodes are also known as clients. Software that can act as an Ethereum node include Parity, Go-Ethereum, etc.
- **Block:** A block is a package of data that has zero or more transactions, the hash of the previous block, and optionally other data.
- **Miners:** Miners will add the block to the Blockchain. Miners are simply nodes in the ethereum network who find new block, confirm transactions and commit new transactions in a block
- **Proof of work:** Proof of work is an activity that miners undertake to write transactions to a new block. It refers to a mathematical value that can act as the proof of having solved a resource and time-consuming computational problem such as finding nonce in Ethereum (Ethash POW Algorithm by Ethereum).
- **Ethereum Virtual Machine:** Ethereum Virtual Machine is the decentralized computing platform which forms the core of the Ethereum platform.

# Ethereum Blocks / Components

- **Smart Contract:** A persistent piece of code on the Ethereum Blockchain that has a set of data and executable functions. These functions execute when Ethereum transactions are made to them with certain input parameters. Based on the input parameters, the functions will execute and interact with data within and outside of the contract.
- **Ether:** Ether is the name of the currency used within Ethereum Network. Miners are rewarded with ethers for providing computing power for finding a new block and confirming new transaction of Smart contracts. Ether is used to pay for computations within the EVM.
- **Gas:** Refers to the pricing value required to successfully perform a transaction or execute a smart contract on the Ethereum Blockchain platform. It is also the name for crypto-fuel that is consumed when code is executed by the EVM. The gas is paid as execution fee for every operation made on an Ethereum Blockchain.
- **Gas Limit:** The gas limit represents the maximum amount of gas you are willing to pay for a smart contract transaction execution.
- **Mining Pool:** Mining pool are simply groups of miners that work together to mine blocks for ethereum network.
- **Main-net:** A main-net is a main Ethereum Blockchain network.

# A fairer financial system

Today, billions of people can't open bank accounts, others have their payments blocked.

Ethereum's decentralized finance (DeFi) system never sleeps or discriminates.

With just an internet connection, you can send, receive, borrow, earn interest, and even stream funds anywhere in the world.

# The internet of assets



Ethereum isn't just for digital money. Anything you can own can be represented, traded and put to use as non-fungible tokens (NFTs).



You can tokenise your art and get royalties automatically every time it's re-sold.



Or use a token for something you own to take out a loan. The possibilities are growing all the time.



# A new frontier for development

---

- ❑ Ethereum and its apps are transparent and open source.
- ❑ You can fork code and re-use functionality others have already built.
- ❑ If you don't want to learn a new language you can just interact with open-sourced code using JavaScript and other existing languages.



# Ethereum Virtual Machine

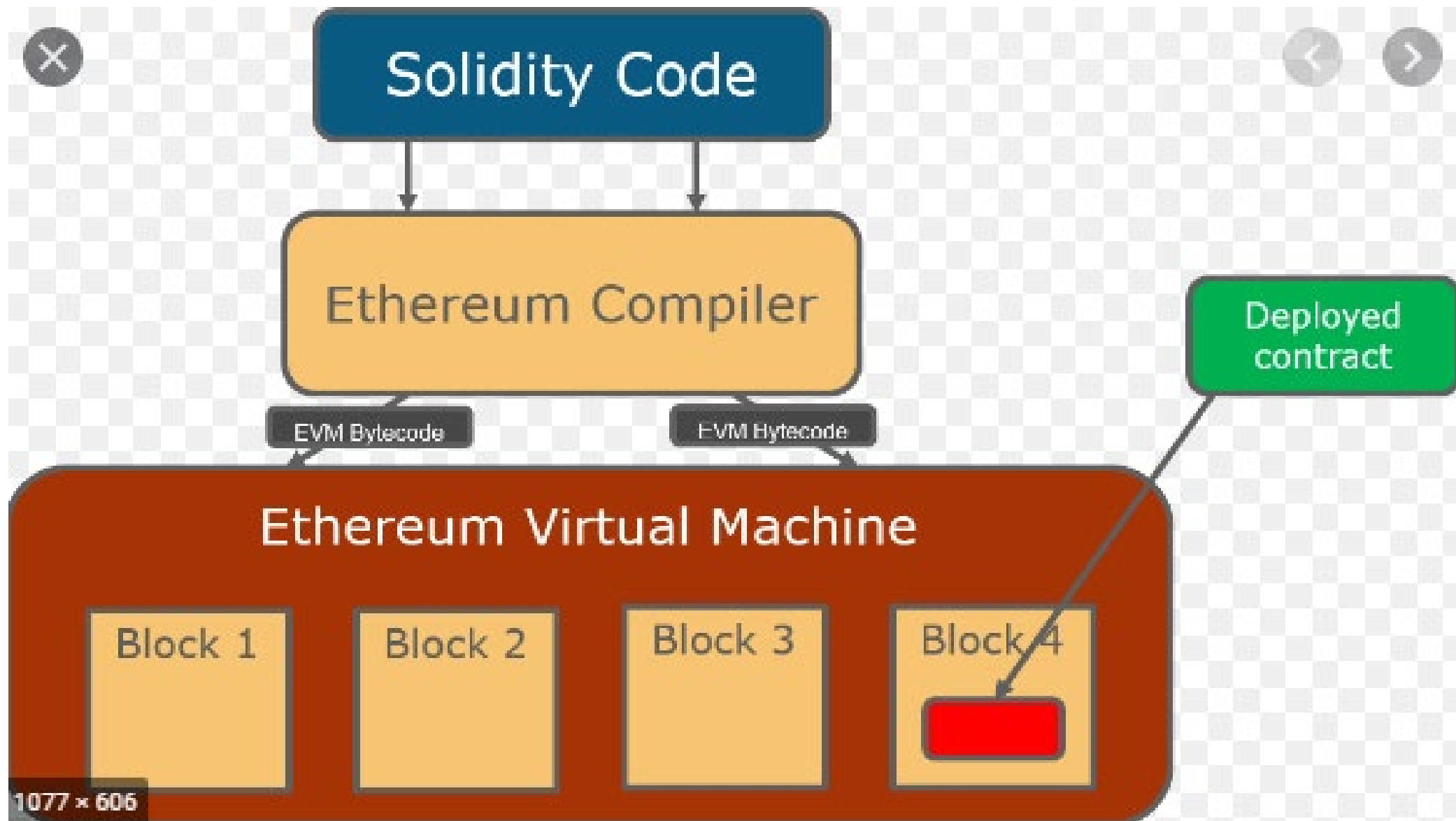
---

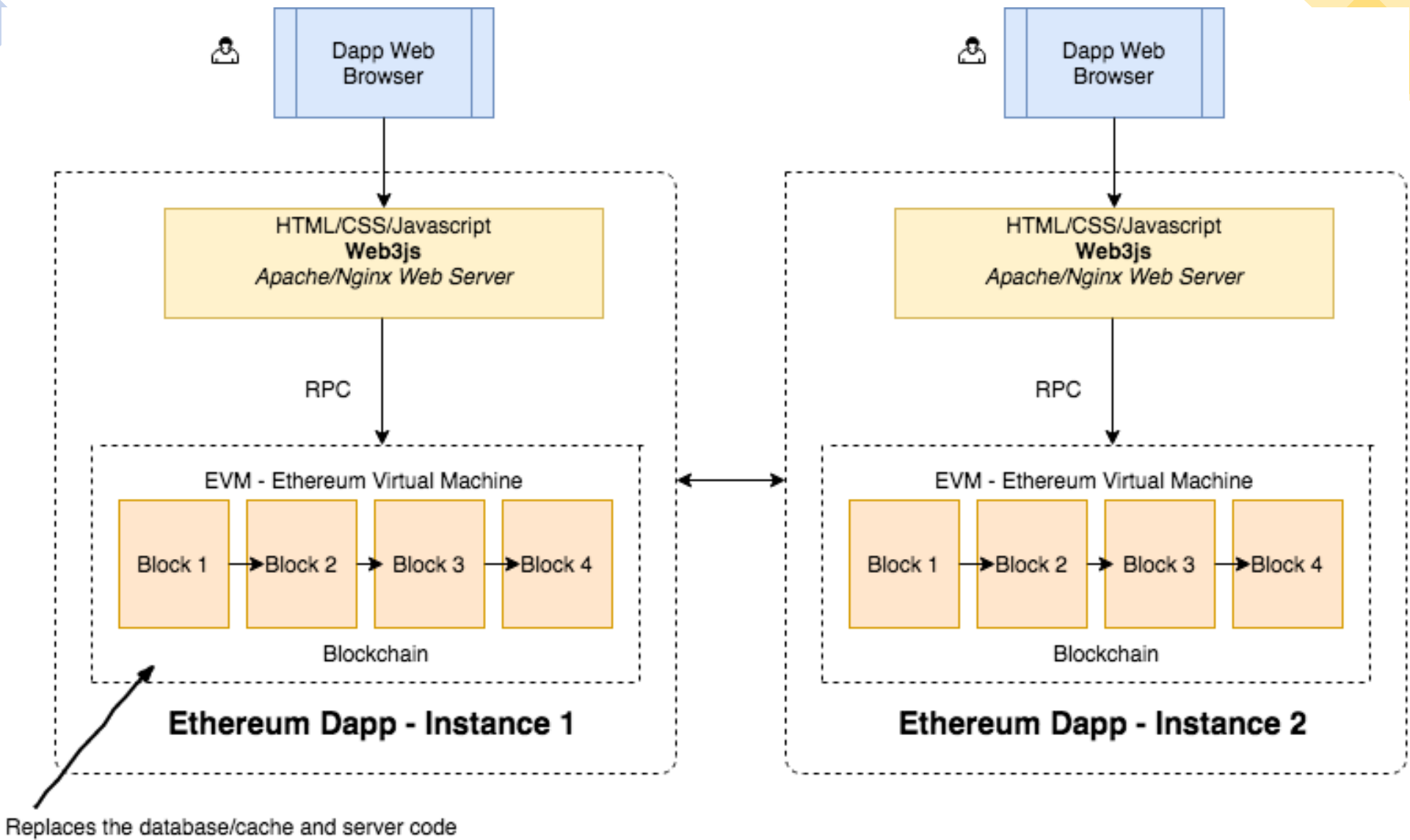
- Ethereum is a decentralized platform that runs smart contracts, applications that run exactly as programmed without possibility of downtime, censorship, fraud or third party interference.
- Go Ethereum is one of the three original implementations (along with C++ and Python) of the Ethereum protocol.
- It is written in Go, fully open source and licensed under the GNU LGPL v3.

# Ethereum Virtual Machine

---

- ❑ At a high level, the EVM running on the **Ethereum** blockchain can be thought of as a global decentralized **computer** containing millions of executable objects, each with its own permanent data store. ... The EVM has a stack-based **architecture**, storing all in-memory values on a stack.
- ❑ Ethereum, in a rustic way, defines a set of generalized protocols which have become the pillars of the development of decentralized applications. At the heart of this, lies the Ethereum Virtual Machine.
- ❑ It is important to note that, the Ethereum Virtual Machine is not only completely sandboxed, but also completely isolated. This means that code that is currently running on the EVM has no access to the network or the file-system and can sparingly access other contracts.





# The Ethereum Network

---

The ethereum network is a public blockchain network. It forms the basis of all decentralized peer-to-peer applications and organizations run on the network.

---

The network is comprised of two types of nodes namely, full nodes and light-weight-nodes.

---

**Full nodes** contain the entire history of transactions since the genesis block. They are a full-fledged proof of the integrity of the blockchain network. Full nodes have to contain each and every transaction that has been verified according to the rules set up by Ethereum's specifications.

# The Ethereum Network

- **Light-weight nodes** on the other hand only contain a subset of the entire blockchain.
- These types of nodes are mostly used in e-wallets which have to be light-weight in nature and hence the entire blockchain cannot be stored on them.
- These nodes, in contrast, do not verify every block or transaction and may not have a copy of the current blockchain state.
- They rely on full nodes to provide them with missing details (or simply lack particular functionality).
- The advantage of light nodes is that they can get up and running much more quickly, can run on more computationally/memory constrained devices, and don't eat up nearly as much storage.



# Ether and Gas

- Ether is the name of the crypto-currency used to pay for transactions on the ethereum network. Besides from paying for general transactions and services, Ether is also used to buy Gas, which in turn is used to pay for computation within the EVM.
- Ether is the metric unit and has a lot of denominations which help accurately pay for transactions and gas. The smallest denomination a.k.a base unit is called Wei. The denominations along with their specific names can be seen in the table below:



# Ether and Gas

Units	Wei Value	Wei
wei	1 wei	1
Kwei	1e3 wei	1,000
Mwei	1e6 wei	1,000,000
Gwei	1e9 wei	1,000,000,000
microEther	1e12 wei	1,000,000,000,000
milliEther	1e15 wei	1,000,000,000,000,000
Ether	1e18 wei	1,000,000,000,000,000,000

# Ether and Gas

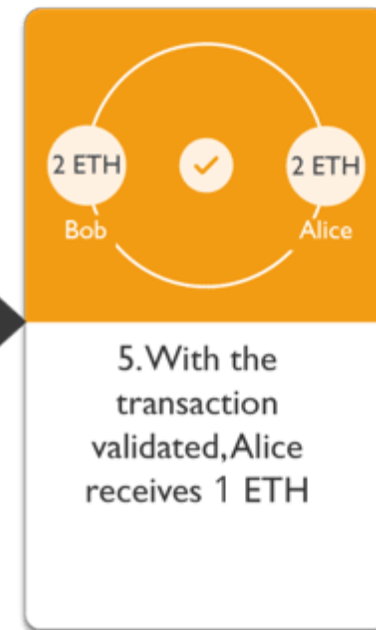
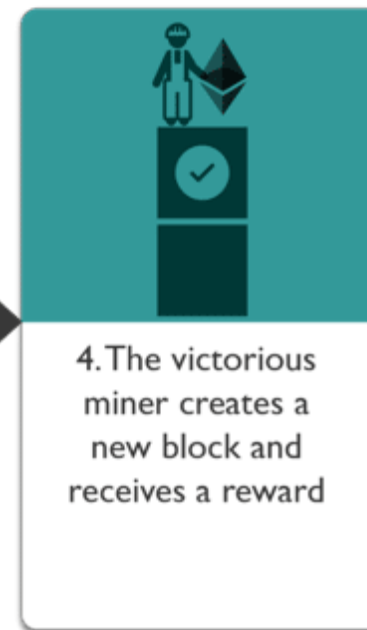
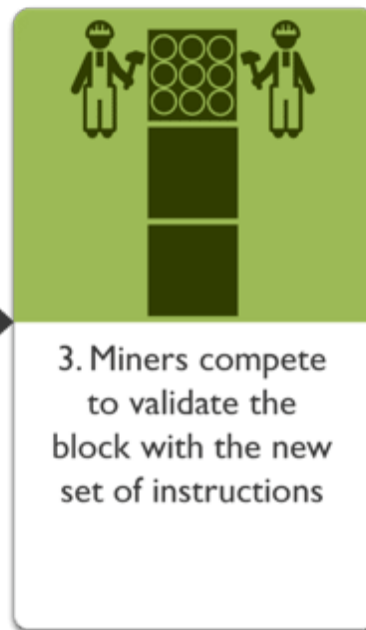
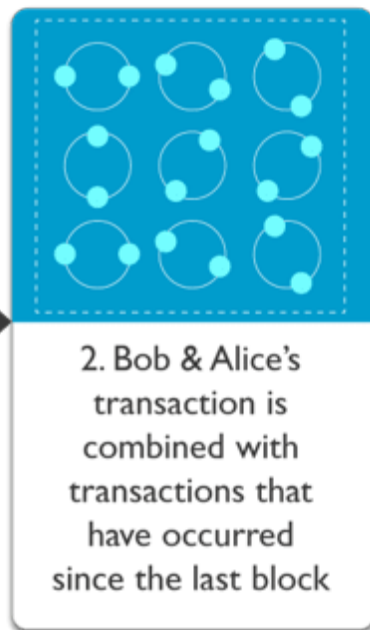
- ❑ As discussed earlier, we know that EVM is responsible for running code that is deployed on its network.
- ❑ So what's stopping someone from running an infinite loop on the EVM and completely overloading its memory? This is where the concept of Gas comes in.
- ❑ Gas is used as a metric for paying for computational resources on the network. Every contract on the network has a set maximum amount of gas that it can use for its computations. This is known as the **"Gas Limit"**

# Ether and Gas

- **Gas Price:** This is the cost of gas in terms of tokens like Ether and its other denominations. To stabilize the value of gas, the Gas Price is a floating value such that if the cost of tokens or currency fluctuates, the Gas Price changes to keep the same real value.
- **Gas Fee:** This is effectively the amount of Gas needed to be paid to run a particular transaction or program (called a contract).
- Hence, if someone tries to run a piece of code that runs forever, the contract will eventually exceed its gas limit and the entire transaction that invoked the contract will be rolled back to its previous state.

# Ethereum Mining

- **Gas Price:** This is the cost of gas in terms of tokens like Ether and its other denominations. To stabilize the value of gas, the Gas Price is a floating value such that if the cost of tokens or currency fluctuates, the Gas Price changes to keep the same real value.
- **Gas Fee:** This is effectively the amount of Gas needed to be paid to run a particular transaction or program (called a contract).
- Hence, if someone tries to run a piece of code that runs forever, the contract will eventually exceed its gas limit and the entire transaction that invoked the contract will be rolled back to its previous state.



# Ethereum Mining

- ❑ Ethereum, much like other public blockchain technologies ensures security through an incentive-based model. This is called a proof-of-work mechanism. The figure below shows how ethereum mining works:
- ❑ From a more technical perspective, the proof-of-work algorithm used is called Ethash, which is a hashing algorithm inspired by the Dagger-Hashimoto Algorithm.
- ❑ Now that we've seen the working architecture of ethereum and discussed its essential elements, let's see a real-world problem and the ethereum approach to solve the same.

# Decentralized Application (DApp)

---

It is an Application that operates without a central trusted party.

---

An application that enables direct interaction/agreements/communication between end users and/or resources without a middleman.

---

To develop a Dapp we need to know Solidity (learn more about solidity <https://solidity.readthedocs.io>).

# Ethereum Client

---

Ethereum Client Refers to any node able to parse and verify new transaction, execute smart contract and find new block for the blockchain.

---

It also allows you to provide interfaces to create transactions and mine blocks which is the key for any Blockchain interaction.



# Ethereum Client

Tool	Language +
go-ethereum	Go
Parity	Rust
cpp-ethereum	C++
pyethapp	Python
ethereumjs-lib	Javascript
Ethereum(J)	Java
ruby-ethereum	Ruby
ethereumH	Haskell

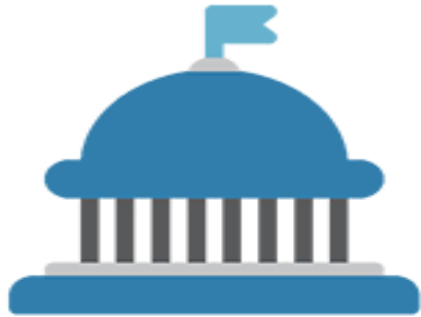
# How Ethereum Network works:

---

- Key components of Ethereum blockchain constitutes: Clients, Transaction Pool, Miners, Mining Pool, Ethereum Virtual Machine, SDK's etc.
- Journey started by Vitalik Buterin (co-founder of Ethereum) by creating first ethereum block called as Genesis Block (First block in Blockchain).
- Ethereum client/nodes can be used to interact with the Ethereum Blockchain via Ethereum SDK's like: Web3.js, Web3j etc. Using clients one can create transaction, broadcast the transaction, read the data of block, get the ethereum account balance etc.
- Client broadcast's transaction to its peer.
- Peers can be another node or miners.
- When the transaction is broadcasted to the network, transaction goes to every peer's Transaction Pool and Mining pool.
- Miners pick the transaction from Mining pool having greater transaction solving fees (Gas), Miner solves the mathematical computation and broadcast the proof of work.
- Other miners validate the nonce given by miner and if the nonce is correct then miner add the block to the Blockchain.

# Decentralized Crowd Funding Use Case

- ❑ **Problem Statement:** A good 'idea' isn't everything in today's world to start a successful business. A lot of funding and effort is needed to implement an idea. This is where organizations like "Kickstarter" come into the picture. They provide projects with the public exposure needed for donations towards their project to get it up and running, but the centralized architecture of such a motive has its downsides, mainly in the way the rewards are handled. Since the centralized authority makes all the decisions, systems are prone to rules like:
  - ❑ anyone who missed the deadline for the campaign cannot get in any more
  - ❑ any donor who changed their mind can't get out



1. A centralised crowdfunding has a lot of fundanmental probelems with money management and reward system



2. A DAO is setup to simulate crowdfunding of projects



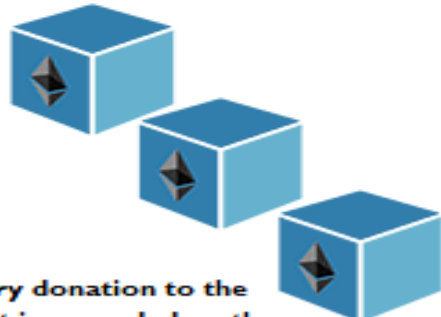
3. Conditions are set in a contract to simulate crowdfunding logic



4. Every person who donates is issued a token

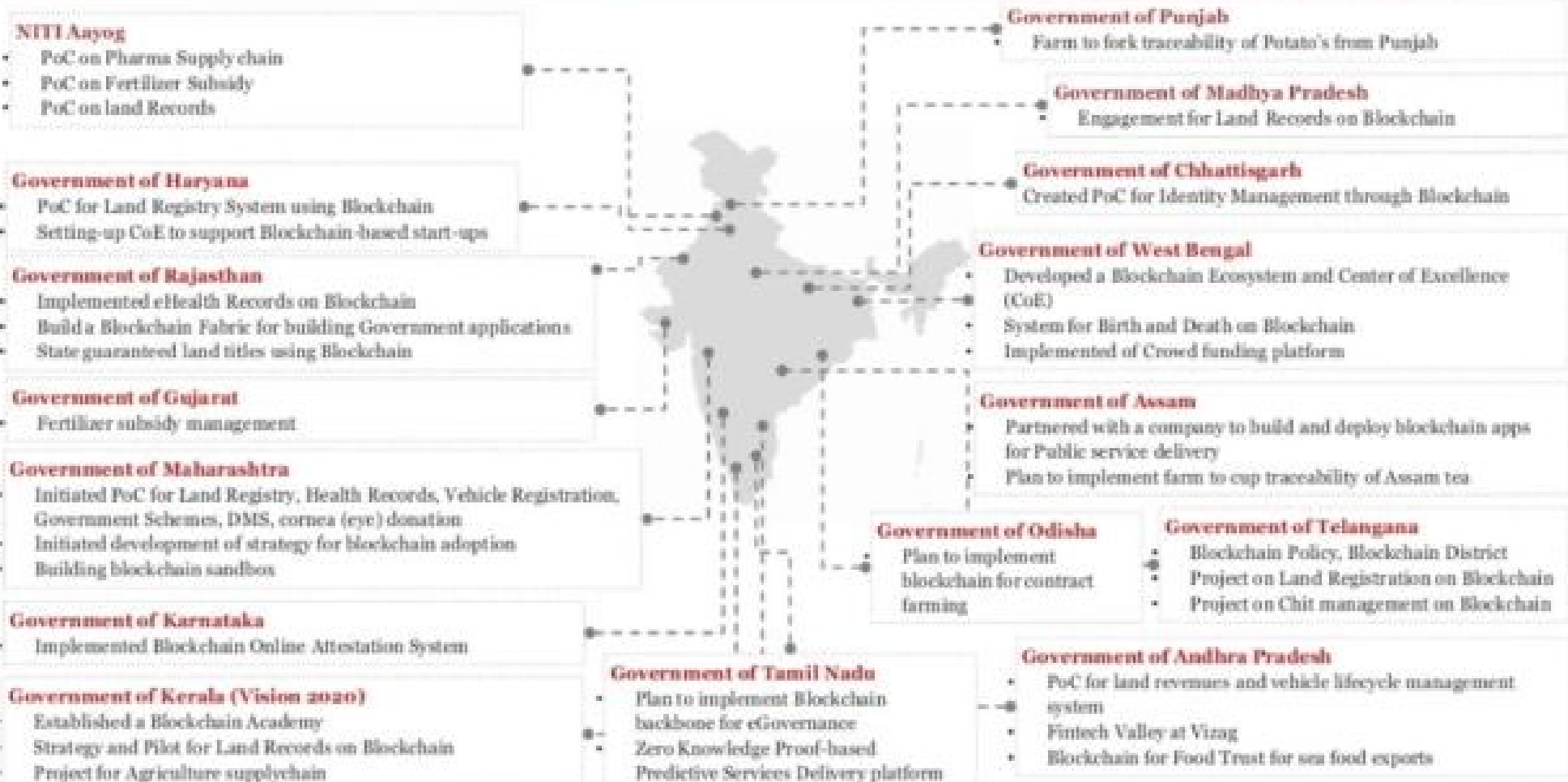


5. People donate via a contract



6. Every donation to the project is recorded on the blockchain

# Blockchain Implementations in India



<b>Blockchain characteristics comparison</b>			
<b>Characteristics</b>	<b>Bitcoin</b>	<b>Ethereum</b>	<b>Hyperledger</b>
<b>Permission restrictions</b>	Permissionless	Permissionless	Permissioned
<b>Restricted public access to data</b>	Public	Public or private	Private
<b>Consensus</b>	Proof-of-Work	Proof-of-Work	PBFT
<b>Scalability</b>	High node-scalability, Low performance-scalability	High node-scalability, Low performance-scalability	Low node-scalability, High performance-scalability
<b>Centralized regulation (governance*)</b>	Low, decentralized decision making by community/miners	Medium, core developer group, but EIP process	Low, open-governance model based on Linux model
<b>Anonymity</b>	Pseudonymity, no encryption of transaction data	Pseudonymity, no encryption of transaction data	Pseudonymity, encryption of transaction data
<b>Native currency</b>	Yes, bitcoin, high value	Yes, ether	No
<b>Scripting</b>	Limited possibility, stack-based scripting	High possibility, Turing-complete virtual machine, high-level language support (Solidity)	High possibility, Turing-complete scripting of chaincode, high-level Go-language

Characteristics	Ethereum	Hyperledger Fabric	R3 Corda
Programming Language	Solidity	Go, Java	Kotlin
Governance	Distributed among all participants	Linux foundation and organisation in the Chain	R3 and organisations involved.
Smart Contract	Not legally bounded	Not legally bounded	Legally bounded
Consensus Algorithm	PoW. Casper implementation PoS.	PBFT	Notary nodes can run several consensus algorithm
Scalability	Existing scalability issue	Not prevalent	Not prevalent
Privacy	Existing privacy issue	Not prevalent	Not prevalent
Currency	Ether	None Can be made using chaincode	None





# Ethereum Developer Tools

Ref: <https://media.consensys.net/>

- For any developer — whether you're a wide-eyed Web3 novice or a grizzled OG crypto-anarchist tech overlord — Github is your friend.
- The ConsenSys Github in particular features perhaps the most definitive repository for Ethereum dev tools you'll find on the entire Internet.
- Whether it's the basics, coding languages, IDEs, APIs, ABIs, frameworks, best practices, smart contract standards, testing tools, or faucets, the ConsenSys Github has what you need to start building decentralized apps on the Ethereum blockchain now.



# Tools for Ethereum Developer

- [Solidity](#) — The most popular smart contract language.
- [Truffle](#) — Most popular smart contract development, testing, and deployment framework. Install the CLI via NPM and start here to write your first smart contracts.
- [Metamask](#) — Chrome extension wallet to interact with Dapps.
- [Truffle boxes](#) — Packaged components for the Ethereum ecosystem
- [EthHub.io](#) — Comprehensive crowdsourced overview of Ethereum- its history, governance, future plans and development resources.
- [Infura](#) — Scalable, secure, and reliable access to the Ethereum network.

# Solidity Programming Language

- [Solidity](#) — Ethereum smart contracting language
- [Bamboo](#) — A morphing smart contract language
- [Vyper](#) — New experimental pythonic programming language
- [LLL](#) — Low-level Lisp-like Language
- [Flint](#) — New language under development with security features including asset types, state transition, and safe integers

# Frameworks

- [Truffle](#) — Most popular smart contract development, testing, and deployment framework. The Truffle suite includes Truffle, [Ganache](#), and [Drizzle](#). [Deep dive on Truffle here](#)
- [Embark](#) — Framework for DApp development
- [Waffle](#) — Framework for advanced smart contract development and testing, small, flexible, fast (based on ethers.js)
- [Dapp](#) — Framework for DApp development, successor to DApple
- [Populus](#) — The Ethereum development framework with the most cute animal pictures
- [Etherlime](#) — ethers.js based framework for Dapp deployment
- [Parasol](#) — Agile smart contract development environment with testing, INFURA deployment, automatic contract documentation and more. It features a a flexible and unopinionated design with unlimited customizability
- [Oxcert](#) — JavaScript framework for building decentralized applications

# Integrated Development Environments

- [Remix](#) — Web IDE with built in static analysis, test blockchain VM.
- [Superblocks Lab](#) — Web IDE. Built in browser blockchain VM, Metamask integration (one click deployments to Testnet/Mainnet), transaction logger and live code your WebApp among many other features.
- [Atom](#) — Atom editor with [Atom Solidity Linter](#), [Etheratom](#), [autocomplete-solidity](#), and [language-solidity](#) packages
- [Pragma](#) — Very simple web IDE for solidity, and auto-generated interfaces for smart contracts.
- [Vim solidity](#) — Vim syntax file for solidity
- [Visual Studio Code](#) — Visual Studio Code extension that adds support for Solidity
- [IntelliJ Solidity Plugin](#) — Open-source plug-in for [JetBrains IntelliJ Idea IDE](#) (free/commercial) with syntax highlighting, formatting, code completion etc.
- [YAKINDU Solidity Tools](#) — Eclipse based IDE. Features context sensitive code completion and help, code navigation, syntax coloring, build in compiler, quick fixes and templates.
- [Eth Fiddle](#) — IDE developed by [The Loom Network](#) that allows you to write, compile and debug your smart contract. Easy to share and find code snippets.

# Test Blockchain Networks

- [Ganache](#) — App for test Ethereum blockchain with visual UI and logs
- [Kaleido](#) — Use Kaleido for spinning up a consortium blockchain network. Great for PoCs and testing
- [Pantheon Private Network](#) — Run a private network of Pantheon nodes in a Docker container  
\*\* [Orion](#) — Component for performing private transactions by PegaSys \*\* [Artemis](#) — Java implementation of the Ethereum 2.0 Beacon Chain by PegaSys
- [Cliquebait](#) — Simplifies integration and accepting testing of smart contract applications with docker instances that closely resembles a real blockchain network
- [Local Raiden](#) — Run a local Raiden network in docker containers for demo and testing purposes
- [Private networks deployment scripts](#) — Out-of-the-box deployment scripts for private PoA networks
- [Local Ethereum Network](#) — Out-of-the-box deployment scripts for private PoW networks
- [Ethereum on Azure](#) — Deployment and governance of consortium Ethereum PoA networks
- [getho](#) — DApp development platform including PoA private blockchain and Smart Contract testing tool.
- [Ethereum on Google Cloud](#) — Build Ethereum network based on Proof of Work

# Test Ether Faucet

- [Rinkeby faucet](#)
- [Kovan faucet](#)
- [Ropsten faucet](#)
- [Universal faucet](#)

# Communicating with Ethereum Frontend Ethereum APIs

- [Web3.js](#) — Javascript Web3
- [Eth.js](#) — Javascript Web3 alternative
- [Ethers.js](#) — Javascript Web3 alternative, useful utilities and wallet features
- [Web3Wrapper](#) — Typescript Web3 alternative
- [Ethereumjs](#) — A collection of utility functions for Ethereum like [ethereumjs-util](#) and [ethereumjs-tx](#)
- [flex-contract](#) and [flex-ether](#) Modern, zero-configuration, high-level libraries for interacting with smart contracts and making transactions.
- [ez-ens](#) Simple, zero-configuration Ethereum Name Service address resolver.
- [web3x](#) — A TypeScript port of web3.js. Benefits includes tiny builds and full type safety, including when interacting with contracts.
- [Drizzle](#) — Redux library to connect a frontend to a blockchain
- [Tasit SDK](#) — A JavaScript SDK for making native mobile Ethereum dapps using React Native
- [Subproviders](#) — Several useful subproviders to use in conjunction with [Web3-provider-engine](#) (including a LedgerSubprovider for adding Ledger hardware wallet support to your dApp)
- [web3-react](#) — React framework for building single-page Ethereum dApps
- [Vortex](#) — A Dapp-ready Redux Store. Smart and Dynamic background data refresh thanks to WebSockets. Works with [Truffle](#) and [Embark](#).

# Backend Ethereum APIs

- [Web3.py](#) — Python Web3
- [Web3.php](#) — PHP Web3
- [Ethereum-php](#) — PHP Web3
- [Web3j](#) — Java Web3
- [Nethereum](#) — .Net Web3
- [Ethereum.rb](#) — Ruby Web3
- [Web3.hs](#) — Haskell Web3
- [KEthereum](#) — Kotlin Web3
- [Pyethereum](#) — The Python core library of the Ethereum project
- [Eventem](#) — A bridge between Ethereum smart contract events and backend microservices, written in Java by Kauri
- [Ethereumex](#) — Elixir JSON-RPC client for the Ethereum blockchain
- [EthContract](#) — A set of helper methods to help query ETH smart contracts in Elixir








# Ethereum clients

- An Ethereum client is a software application that implements the Ethereum specification and communicates over the peer-to-peer network with other Ethereum clients.
- Different Ethereum clients *interoperate* if they comply with the reference specification and the standardized communications protocols.
- While these different clients are implemented by different teams and in different programming languages, they all “speak” the same protocol and follow the same rules.
- As such, they can all be used to operate and interact with the same Ethereum network.
- Ethereum is an open source project, and the source code for all the major clients is available under open source licenses (e.g., LGPL v3.0), free to download and use for any purpose.
- *Open source* means more than simply free to use, though. It also means that Ethereum is developed by an open community of volunteers and can be modified by anyone. More eyes means more trustworthy code.

# Ethereum Languages

- Select your programming language of choice to find projects, resources, and virtual communities:
- [Ethereum for Dart developers](#)
- [Ethereum for Delphi developers](#)
- [Ethereum for .NET developers](#)
- [Ethereum for Go developers](#)
- [Ethereum for Java developers](#)
- [Ethereum for JavaScript developers](#)
- [Ethereum for Python developers](#)
- [Ethereum for Ruby developers](#)
- [Ethereum for Rust developers](#)

# Ethereum Wallets

- There are a few ways to interface and interact with your account:
-  Physical hardware wallets are devices that let you keep your crypto offline – very secure
-  Mobile applications that make your funds accessible from anywhere
-  Browser wallets are web applications that let you interact with your account directly in the browser
-  Browser extension wallets are extensions you download that let you interact with your account and applications through the browser
-  Desktop applications if you prefer to manage your funds via macOS, Windows or Linux

# Ethereum Wallets

- “A **digital wallet**, also known as **e-wallet**, is an electronic device, online service, or software program that allows one party to make electronic transactions with another party bartering digital currency units for goods and services.”
- Let us simplify it again. Some people spend money and hoard gold, as it is precious, and they might resell it when the price is high and might store it and wait for a better price to sell it and make a profit. Similarly, instead of gold, we buy virtual coins like Ethereum, Bitcoin, Dogecoin, etc., and money. The wallet facilitates the process.
- This rise in popularity leads many investors to buy this Ethereum currency and make a fortune out of it. When you buy a digital currency like Ethereum, you'll get a private key representing your ownership of it.
- To manage these transactions and keys associated with each coin, crypto wallets are used. A wallet is a software application or electronic chip used to store, buy, and manage your cryptocurrencies.
- Wallets store details associated with each transaction and keys which can be used to exchange the currencies.

# Wallets examples

- [List of the Top 7 Best Ethereum Wallets of 2021](#)
  - [Trezor One](#)
  - [Metamask](#)
  - [Ledger Nano](#)
  - [Exodus](#)
  - [Electrum](#)
  - [Coinbase](#)
  - [Mist](#)

# Ethereum Accounts

- An Ethereum account is an entity with an ether (ETH) balance that can send transactions on Ethereum. Accounts can be user-controlled or deployed as smart contracts.

# Ethereum Accounts

- **ACCOUNT TYPES**

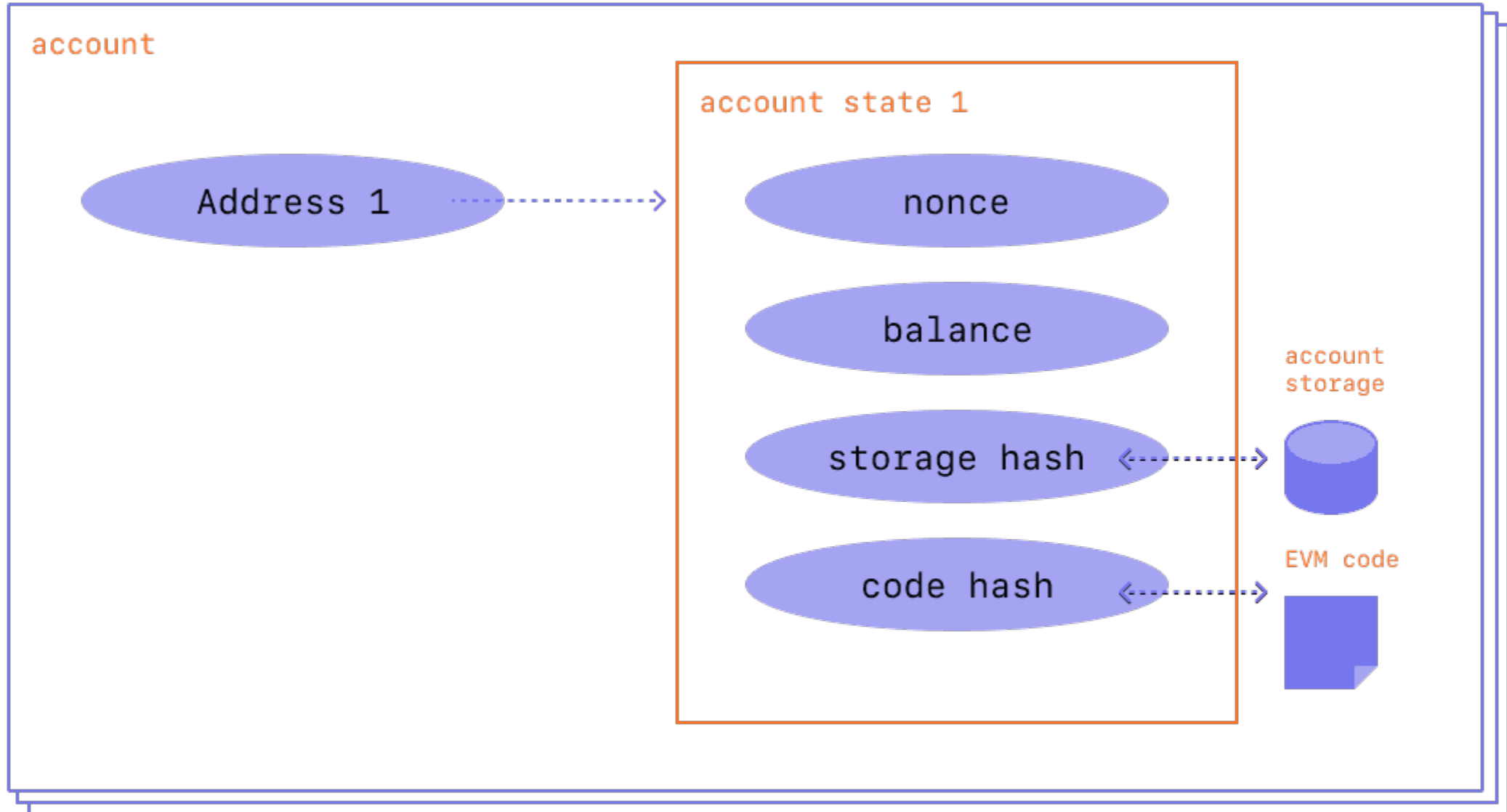
- Ethereum has two account types:
  - Externally-owned account (EOA) – controlled by anyone with the private keys
  - Contract account – a smart contract deployed to the network, controlled by code. Learn about [smart contracts](#)
- Both account types have the ability to:
  - Receive, hold and send ETH and tokens
  - Interact with deployed smart contracts

# Ethereum Accounts

- **Externally-owned**
- Creating an account costs nothing
- Can initiate transactions
- Transactions between externally-owned accounts can only be ETH/token transfers
- Made up of a cryptographic pair of keys: public and private keys that control account activities
- **Contract**
- Creating a contract has a cost because you're using network storage
- Can only send transactions in response to receiving a transaction
- Transactions from an external account to a contract account can trigger code which can execute many different actions, such as transferring tokens or even creating a new contract
- Contract accounts don't have private keys. Instead, they are controlled by the logic of the smart contract code



# Ethereum Account



# Ethereum Account

- Ethereum accounts have four fields (Refer account diagram)
  1. nonce – A counter that indicates the number of transactions sent from an externally-owned account or the number of contracts created by a contract account. Only one transaction with a given nonce can be executed for each account, protecting against replay attacks where signed transactions are repeatedly broadcast and re-executed.
  2. balance – The number of wei owned by this address. Wei is a denomination of ETH and there are  $1e+18$  wei per ETH.

<https://ethereum.org/en/developers/docs/accounts/>

# Ethereum Account

**codeHash** – This hash refers to the code of an account on the Ethereum virtual machine (EVM). Contract accounts have code fragments programmed in that can perform different operations. This EVM code gets executed if the account gets a message call. It cannot be changed, unlike the other account fields. All such code fragments are contained in the state database under their corresponding hashes for later retrieval. This hash value is known as a **codeHash**. For externally owned accounts, the **codeHash** field is the hash of an empty string.

**storageRoot** – Sometimes known as a storage hash. A 256-bit hash of the root node of a Merkle Patricia trie that encodes the storage contents of the account (a mapping between 256-bit integer values), encoded into the trie as a mapping from the Keccak 256-bit hash of the 256-bit integer keys to the RLP-encoded 256-bit integer values. This trie encodes the hash of the storage contents of this account, and is empty by default.

# Ethereum Keypairs

- An account is made up of a cryptographic pair of keys: public and private. They help prove that a transaction was actually signed by the sender and prevent forgeries. Your private key is what you use to sign transactions, so it grants you custody over the funds associated with your account. You never really hold cryptocurrency, you hold private keys – the funds are always on Ethereum's ledger.
- This prevents malicious actors from broadcasting fake transactions because you can always verify the sender of a transaction.
- If Alice wants to send ether from her own account to Bob's account, Alice needs to create a transaction request and send it out to the network for verification. Ethereum's usage of public-key cryptography ensures that Alice can prove that she originally initiated the transaction request. Without cryptographic mechanisms, a malicious adversary Eve could simply publicly broadcast a request that looks something like "send 5 ETH from Alice's account to Eve's account," and no one would be able to verify that it didn't come from Alice

# Ethereum Keypairs

When you want to create an account most libraries will generate you a random private key.

A private key is made up of 64 hex characters and can be encrypted with a password.

Example:

```
fffffffffffffffffffffffffffffebaaedce6af48a03bbfd25e8cd036415f
```

The public key is generated from the private key using the Elliptic Curve Digital Signature Algorithm([opens in a new tab](#))<sup>7</sup>. You get a public address for your account by taking the last 20 bytes of the Keccak-256 hash of the public key and adding 0x to the beginning.

The following example shows how to use a signing tool called Clef([opens in a new tab](#))<sup>7</sup> to generate a new account. Clef is an account management and signing tool that comes bundled with the Ethereum client, Geth([opens in a new tab](#))<sup>7</sup>. The clef newaccount command creates a new key pair and saves them in an encrypted keystore.

# Ethereum Keypairs

- It is possible to derive new public keys from your private key but you cannot derive a private key from public keys. This means it's vital to keep a private key safe and, as the name suggests, **PRIVATE**.
- You need a private key to sign messages and transactions which output a signature. Others can then take the signature to derive your public key, proving the author of the message. In your application, you can use a javascript library to send transactions to the network

# Ethereum Platform

<https://www.coding-bootcamps.com/blog/ethereum-architecture-and-components.html>

