Blockchain Honor Degree Sem VII

HBCC 601 : Blockchain Platforms

Module - 3 : Ethereum Blockchain
(10 Hours)

Instructor : Mrs. Lifna C S

Topics to be covered

- Introduction to Ethereum (Already covered in Module 1)
- **Ethereum and Its Components:**
  - **Mining, Gas, Ethereum, Ether, Ethereum Virtual Machine, Transaction, Accounts.**
- **Architecture of Ethereum,**
- Smart Contract (Already covered in Module 1)
  - Remix IDE,
  - Developing smart contracts for Ethereum blockchain,
  - Applications using smart contracts (Already covered in Module 1)
- **Dapp Architecture.**
- **Types of test networks used in Ethereum,**
- **Transferring Ethers Using MetaMask,**
- Mist Wallet (Deprecated)
- **Case study of Ganache for Ethereum blockchain.**
- Ethereum Frameworks
- Ethereum 2.0
  - POS (Proof of Stake),
  - Sharding of Chain
  - Concept of Beacon chain,

Self-learning Topics: Study case study on any Ethereum blockchain

Benefits of Ethereum

1. **Availability:**
   - As the Ethereum network is decentralized so **there is no downtime**.
   - Even if one node goes down other computing nodes are available.
2. **Privacy:** Users don't need to enter their personal credentials while using the network for exchanges, thus **allowing them to remain anonymous**.
3. **Security**: Ethereum is **designed to be unhackable,** as the hackers have to get control of the majority of the network nodes to exploit the network.
4. **Less ambiguity**: The **smart contracts that are used as a basis for trade and agreement** on Ethereum ensure stronger contracts than traditional contracts which require follow-through and interpretation.
5. **Rapid deployment:** On Ethereum decentralized networks, enterprises can easily deploy and manage private blockchain networks instead of coding blockchain implementation from scratch.
6. **Network size**: Ethereum network **can work with hundreds of nodes and millions of users.**
7. **Data coordination:** Ethereum **decentralized architecture better allocates information** so that the **network participants don't have to rely on a central entity** to manage the system and mediate transactions.

Drawbacks of Ethereum

- **Complicated programming language:** Learning solidity from programming smart contracts on Ethereum can be challenging and one of the main concerns is the scarcity of beginner-friendly classes.
- **Volatile cryptocurrency:** Ethereum investing can be risky as the price of Ether is very volatile, resulting in significant gains as well as a significant loss.
- **Low transaction rate**:
  a. **Bitcoin** has an average transaction rate of **7TPS**
  b. **Ethereum** has an average speed of **15 TPS** which is almost double that of bitcoin but it is still not enough.

Components of Ethereum Network

1. Ethereum Node
2. Ethereum Client
3. Ether
4. Gas
5. Ethereum Accounts
6. Nonce
7. Storage Root
8. Ethash
9. Transactions
10. Ethereum Virtual Machine (EVM)

Components of Ethereum Network - (1) Ethereum Node

- a **computer that is running the software client**.
- Nodes **communicate with one another in order to validate transactions and record data** about the status of the blockchain.
- **Responsible for storing, validating, and trading data.**
- Each node keeps its **own copy of the blockchain** and **strives to verify** that it matches the copies of all the other nodes.
- Every node on the network **must process any action that requires a new block to be added** to the blockchain.
- This network of continually communicating nodes **allows us to avoid relying on a single source of truth and all of the challenges it entails.**
- A new block is **added based on whether or not the majority of nodes accept it.**

Components of Ethereum Network - (1) Ethereum Node (Types)

1. **Full Node**:
   - verify and validate each and every transaction that takes place inside the network
   - **maintain the state / a full copy** of the blockchain.
   - When a smart contract transaction occurs,
     - Full nodes also **execute all of the instructions in the smart contract**.
     - It determines whether or not the smart contract execution is **producing the expected results.**
   - It **keeps receiving copies of the entire blockchain** including its transactions which are stored locally and keeps the latest state of transaction with itself.
   - Eg: **Person A performs a transaction to person B**,
     - transaction is added to the blockchain,
     - Full nodes verify whether the transaction complies with all the Ethereum specifications,
     - Maintain the latest state of the blockchain by storing or removing the specification if it does not comply.
   - **Example of a discarder transaction** is when a person transfers X ETH to another person but their account contains less ETH.

Components of Ethereum Network - (1) Ethereum Node (Types)

2. **Archive Nodes:**
   - **complete nodes that have the "archive mode" option enabled.**
   - While a **Full Node only stores the latest state of the transaction,**
   - the Archive nodes **hold all of the blockchain's history data dating back to the genesis block**.
   - **used when blocks prior to the latest 128 blocks are required.**
   - Eg : using functions like **eth_getBalance** of a historic address would require an archive node, as will interacting with smart contracts launched far earlier in the blockchain.
   - Archive Nodes **memory requirement** :
     - **require more than 6 Terabytes of space**, contrary to Full node which only requires a little over 500 Gigabytes of disk space.
     - **are not useful for average people**,
     - they are **effective in the application of block exploring, wallet vending, and chain analytics**.

Components of Ethereum Network - (1) Ethereum Node (Types)

3. **Light Nodes:**

- does not hold the complete current blockchain state
- stores only the block header.
- suitable for low memory and computational devices since maintaining a light node involve the least investment in hardware, running costs, and technical skill.
- Light nodes **rely on full nodes to function.**
- These nodes **do not need to run continuously or read and publish a large amount of data on the blockchain.**
- It provides **an easy way to create a wallet, especially for beginners.**
- Eg : **Solid-State Drives (SSD)** cannot afford to store the gigabytes of data that other nodes take.
- But there are some limitations of light nodes which cannot be denied, there is no guarantee that the light wallet provider will be online when it is needed.

Components of Ethereum Network - (2) Ethereum Client

- **software program** that is used **to implement the Ethereum specification**
- **connect itself with other Ethereum clients** over a peer-to-peer network.
- Different Ethereum clients can communicate with one another if they follow the reference specification and the defined communication protocols.
- These interactions among different clients in the network **take place using various programming languages**
  - like Geth (Go), OpenEthereum (Rust),
  - Nethermind (C#, .NET), Besu (Java),
  - Erigon (Go/Multi).
- The **yellow paper** is the Ethereum protocol that allows anybody to run a client to construct a node.
- **Ethereum sets standard behaviors that all Ethereum clients must adhere to**
- **Ethereum's specs** enabled the blockchain to allow for different, but interoperable, software implementations of an Ethereum client by providing standard documentation and simple language.

Components of Ethereum Network - (2) Ethereum Client (Types)

1. **Full Client:**
   - **save the complete Ethereum blockchain**,
   - which **might take several days to synchronize** and
   - takes a **massive amount of disc space** – more than 1 Terabyte, according to the most recent estimates.
   - **Enable connected nodes to conduct all network functions**, including as
     - mining,
     - transaction
     - block-header validation,
     - smart contract execution.

Components of Ethereum Network - (2) Ethereum Client (Types)

2. **Light Client:**
   - **do not always need to necessary keep all of the data,**
   - when data storage and performance are concerns, developers utilize the “light clients”.
   - Light clients **provide a portion of full client capability.**
   - they can provide quick delivery and free up data storage space.
   - The functionality of a light client is adapted to the purposes of the Ethereum client.
   - **widely used within wallets to maintain private keys and Ethereum addresses.**
   - **manage smart contract interactions and transaction broadcasts**.
   - **useful for web3 instances** within JavaScript objects, Dapp browsers
   - **obtaining the exchange rate data.**

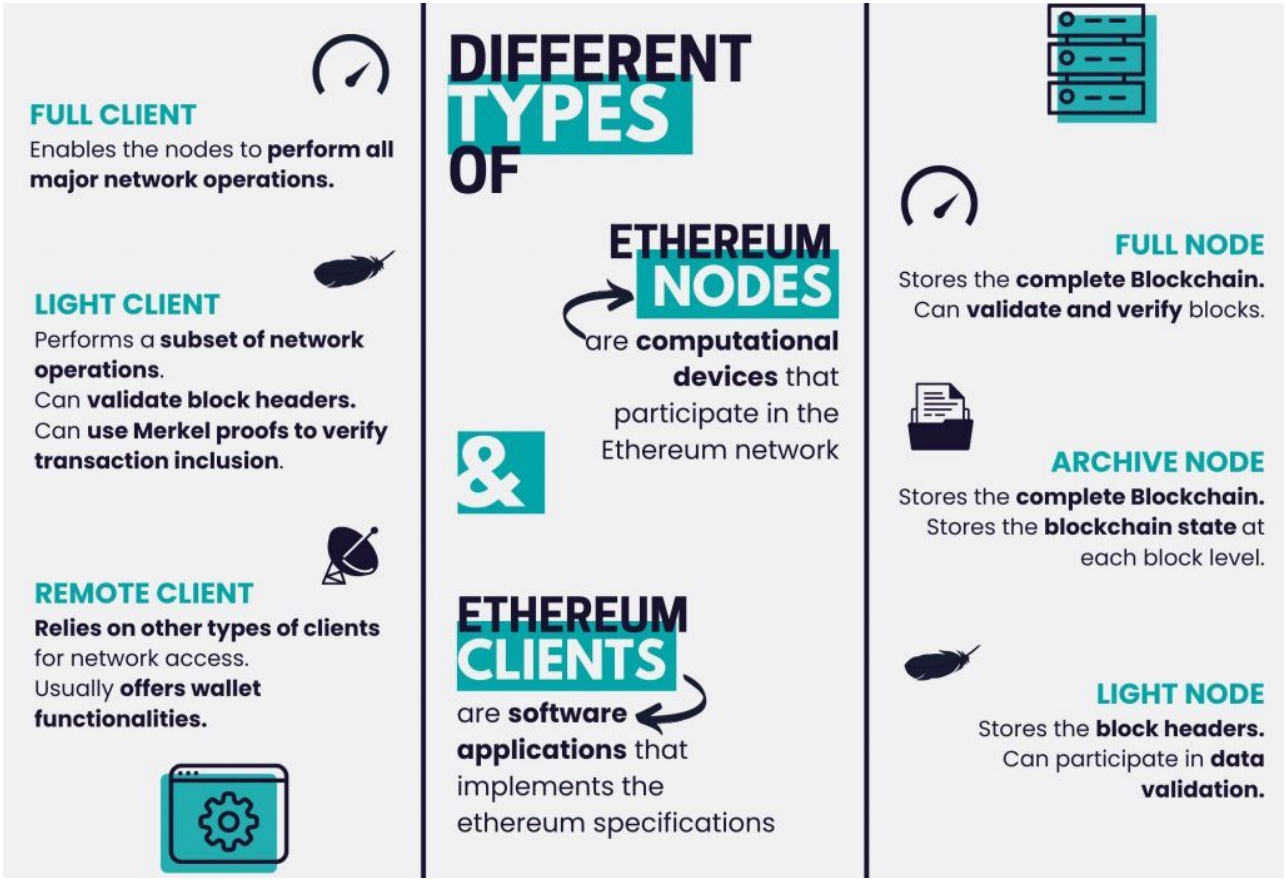
Components of Ethereum Network - (2) Ethereum Client (Types)

3. **Remote Client:**
   - A remote client is **much like a light client**.
   - The primary distinction is that a **remote client does not keep its own copy of the blockchain or validate transactions or block headers.**
   - Remote clients, on the other hand, rely entirely on a full or light client to have access to the Ethereum blockchain network.
   - These clients are **mostly used as wallets for transmitting and receiving transactions.**

Components of Ethereum Network - Node Vs Client

Ethereum Node	Ethereum Client
A machine running Ethereum client software is referred to as an “Ethereum Node”	A client is an Ethereum implementation that validates all transactions in each block, ensuring the network’s security and data accuracy
The three types of Ethereum Nodes are Full, Light, Archive, and Miner Nodes.	The three types of Ethereum Clients are Full, Light, and Remote Clients
The Ethereum node operating system allows us to access the internet	The Ethereum client computer allows a user to access the node operating system

Components of Ethereum Network - Node Vs Client



Courtesy : Chainstack

Components of Ethereum Network -

	Pros	Cons
Light nodes	• Portable • Resource-efficient • User-friendly	• Do not validate the network • Do not propagate blocks • Do not maintain consensus • Less secure
Full nodes	• Validate the network • Propagate blocks • Maintain consensus • More secure	• Resource-heavy • Harder to maintain • Less user-friendly
Pruned	Flexible storage	Need to revalidate old blocks
Archive	Carry full history	Resource and storage heavy
Mining	• Easily trackable involvement • Can pool with others to increase reward rate	• High and wasteful energy consumption • High equipment cost and barrier to entry
Staking	• Low barrier to entry • Low energy consumption	• Reward system based on luck • Low transparency in staking pools
Masternodes	• Balanced network benefits and rewards • Lower maintenance costs	• High initial investment • Difficult setup process

Courtesy : Chainstack

Components of Ethereum Network - (3) Ether

- type of **cryptocurrency used in the Ethereum network**
- It is a **peer-to-peer currency**
- It **tracks and promotes each transaction** in the network.
- It is the **second-largest cryptocurrency in the world.**
- Other cryptocurrencies can be used to get ether tokens, but vice versa is not true.
    - It means that **ether tokens can't be interchanged by other cryptocurrencies** to render computing power for Ethereum transactions.
- paid as a commission for any execution that affects the state in Ethereum.
- **used** in the Ethereum algorithm **as an incentive for miners to blocks to the blockchain using PoW**
- only currency that can be **used to pay transaction costs**, which go to miners as well.
- Aside from paying for transactions, **ether is often used to purchase gas**, which is used to pay for the computation of any transaction on the Ethereum network.

Components of Ethereum Network - (4) Gas

- **An internal currency** of the Ethereum network.
- We need gas to run applications on the Ethereum network, much as we need gas to run a vehicle.
- To complete every transaction on the Ethereum network, a consumer must first make a payment—send out ethers—and **the intermediate monetary value is known as gas.**
- Gas is a unit of measurement on the Ethereum network **for the computing power used to execute a smart contract or a transaction.**
- The price of gas is **very low compared to Ether.**
- **The execution and resource utilization costs are predetermined in Ethereum** in terms of Gas units, called **gwei.**

Components of Ethereum Network - Ether Denominations

Value (in wei)	Exponent	Common Name	SI Name
1	1	wei	wei
1,000	10^3	babbage	kilowei or femtoether
1,000,000	10^6	lovelace	megawei or picoether
1,000,000,000	10^9	shannon	gigawei or nanoether
1,000,000,000,000	10^{12}	szabo	microether or micro
1,000,000,000,000,000	10^{15}	finney	milliether or milli
1,000,000,000,000,000,000	*10^{18}*	*ether*	*ether*
1,000,000,000,000,000,000,000	10^{21}	grand	kiloether
1,000,000,000,000,000,000,000,000	10^{24}		megaether

Courtesy : Zastrin

Components of Ethereum Network - (5) Ethereum Accounts

- similar to a bank account,
- but for ethers or ETH, where Ethereum can be held, transferred to other accounts,
- can also be **used to execute smart contracts**.
- An entity that is composed of **an Ethereum address along with a private key.**
  - The first 20 bytes of the SHA3 hashed public key is the Ethereum address.
- Ethereum has two types of accounts:
  - **Externally owned account (EOA):**
  - **Contract Account**:

Components of Ethereum Network - (5) Ethereum Accounts (Types)

Ethereum has two types of accounts:

1. **Externally owned account (EOA):**
   - **controlled by private keys**.
   - Each EOA **has a public-private key pair**.
   - The **users can send messages by creating and signing transactions**.

Advantages of EOA

1. Transactions from an external account to a contract account **can trigger code that can execute many different actions,** such as transferring tokens or even creating a new contract.
2. Externally Owned Accounts cannot list incoming transactions.

Components of Ethereum Network - (5) Ethereum Accounts (Types)

2. **Contract Account**:
   - **controlled by contract codes**.
   - These **codes are stored with the account**.
   - Each contract account **has an ether balance associated with it**.
   - The contract code of these accounts **gets activated every time a transaction from an EOA or a message from another contract is received by it**.
   - When the contract code activates, **it allows to read/write the message to the local storage, send messages and create contracts.**
   - **Types of Contract Accounts :**
     i. Simple Account: The account is *created and owned by a single account holder*.
     ii. Multisig (multisignature) Account: A Multisig Wallet *contains several owner Accounts*, one of which is also the creator Account.

Components of Ethereum Network - (5) Ethereum Accounts (Types)

Advantages of CA:

1. A contract account can list incoming transactions.
2. Contract accounts can be set up as Multisig Accounts.
3. A Multisig Account can be structured such that it has a daily limit that you specify, and only if the daily limit is exceeded will multiple signatures be required.

Disadvantages of CA:

1. Creating contract accounts costs gas because they use the valuable computational and storage resource of the network.
2. Contract accounts can't initiate new transactions on their own. Instead, contract accounts can only fire transactions in response to other transactions they have received either from an externally owned account or from another contract account.

Components of Ethereum Network - (5) Ethereum Account (EOA Vs CA)

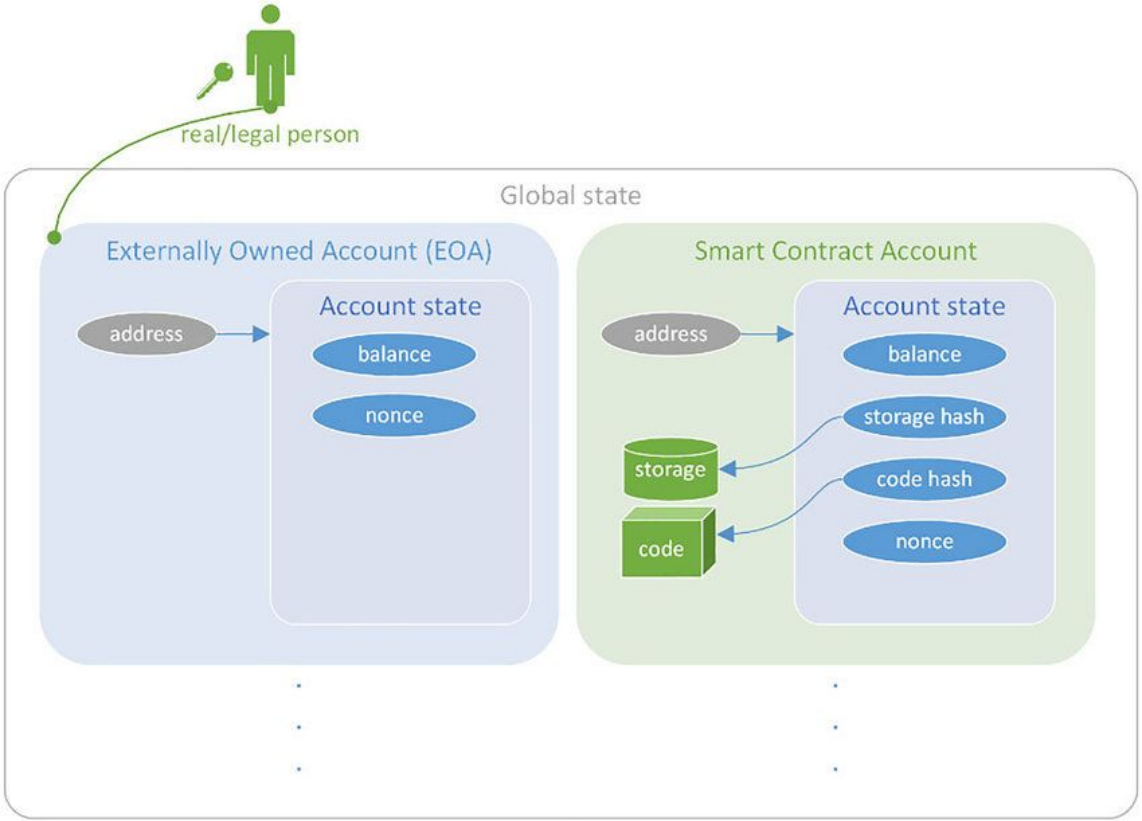
Externally Owned Accounts	Contract Account
Controlled by third party	Controlled by Contract Code
Private Key is needed to access EOAs	No key is needed to access Contract Accounts
EOAs are created automatically on creating wallet	CA require EOAs to be activated
EOA doesn’t have a code associated with it	CA have their own associated code
No execution fee is associated with EOAs	Execution fee is associated with CAs
Code hash = empty string	Code hash represents the code associated with tha account

Components of Ethereum Network - (5) Ethereum Account (Field)

Different Fields in Ethereum Accounts

1. **Nonce:**
   - The nonce in an Ethereum account indicated the number of transactions that have been sent from that account.
   - This ensures that each transaction is made only once by taking count every time it takes place.
2. **Ether Balance:** the amount of ether present in an ether repository of the current ether account.
3. **Contract Code**:
   - This is non-mandatory to fill, in case it is present since not all accounts have a contract code.
   - But note, that they cannot be altered once executed.
4. **Storage:** This field remains not filled unless mentioned.
5. **Code Hash**:
   - hash that refers to the code present in that Ethereum account
   - since no code is associated with externally owned Ethereum accounts, ⇒ **code hash = empty string.**

Components of Ethereum Network - (5) Ethereum Account (Field)



Courtesy : ResearchGate

Components of Ethereum Network - (5) Ethereum Accounts & Key Pairs

An Ethereum account is a **private-public key pair** that may be **linked to a blockchain address**.

Private Key

- It is a “owned” or “externally owned” account if the **private key is known and controlled by someone**.
- **Contract accounts** do not have a private key connected with them, although EOAs have.
- Control and access to one’s assets and contracts are granted through the EOA private key.
- The **user keeps the private key safe.**

Public Key

- Account’s **public key is open**.
- This key serves as the account’s identity.
- A one-way cryptographic function is **used to produce the public key from the private key.**

For example, if you create an account on Ethereum,

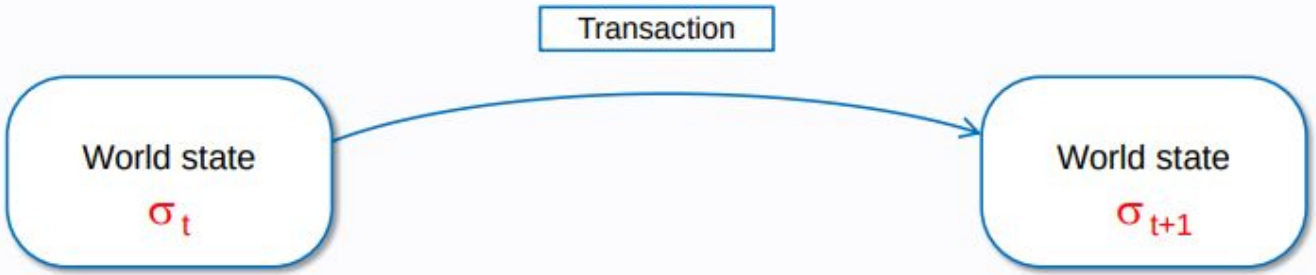
- **Retain the private key**
- **Share the public key.** As transactions between accounts are completed using public keys.

Components of Ethereum Network - Nonce, Storage Root, Ethash

6. **Nonce**
   - **For EOAs,** nonce means the number of transactions via this account.
   - **For CA,** nonce means the number of contracts generated via this account.
7. **Storage Root**
   - It is the **main root node of a Merkle tree**.
   - **Hash of all details of the account** is stored here.
   - The **root of the Merkle tree** is the verification of all transactions.
8. **Ethash**
   - **PoW algorithm for Ethereum 1.0**
   - most **recent version of Dagger-Hashimoto**

Components of Ethereum Network - Transactions

A transaction-based state machine



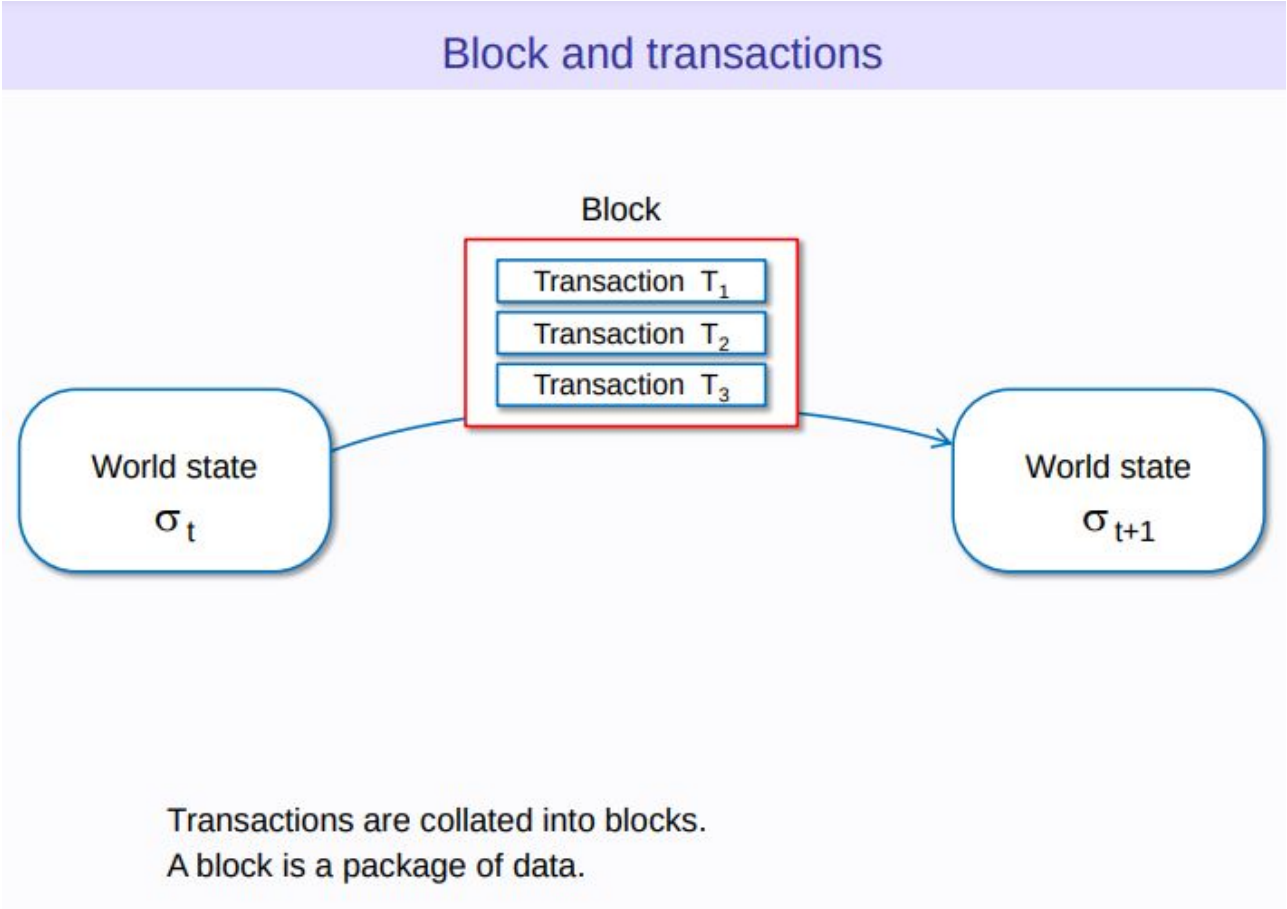
Ethereum can be viewed as a transaction-based state machine.

Components of Ethereum Network - Transactions

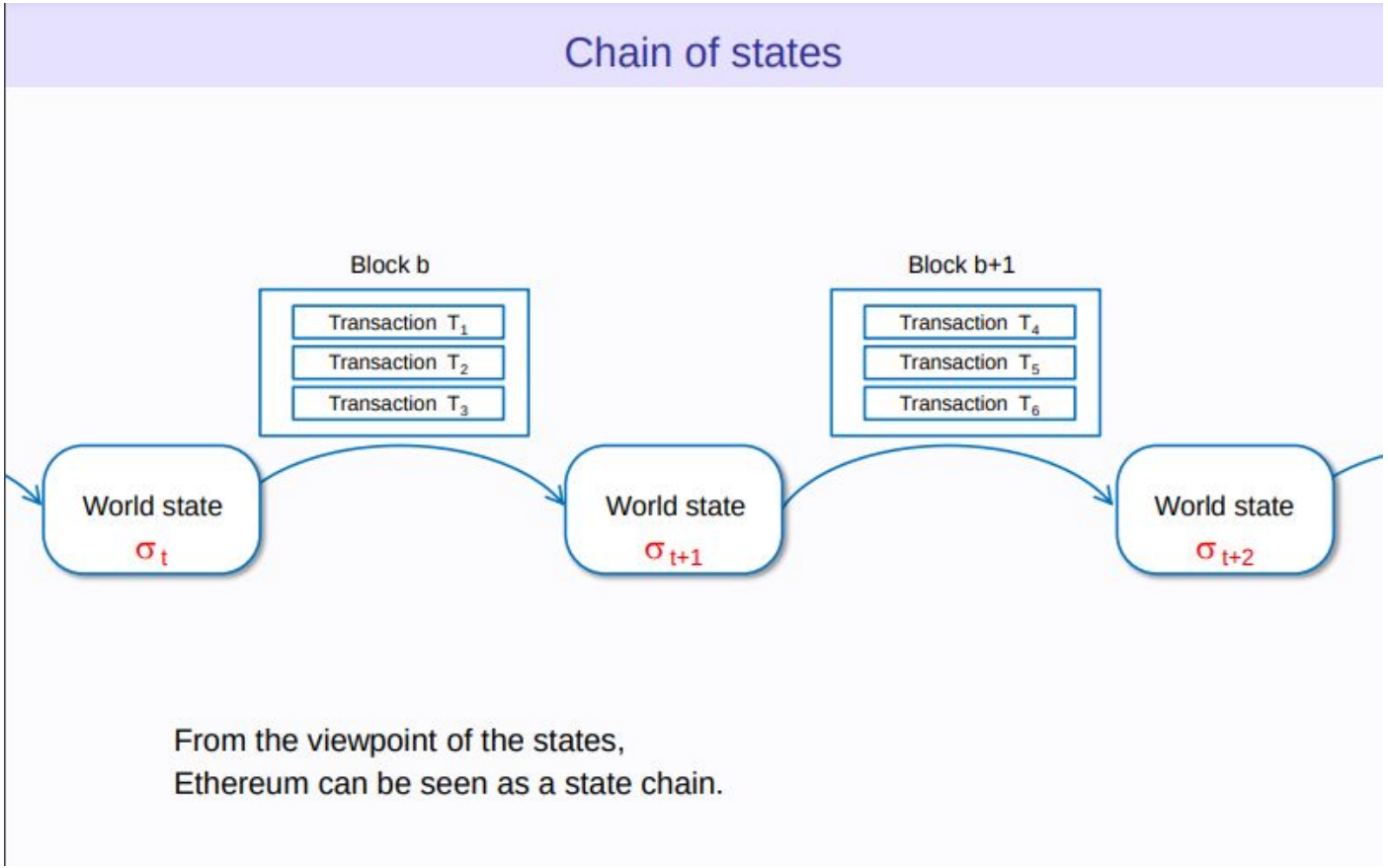
A transaction-based state machine

A transaction represents a valid arc between two states.

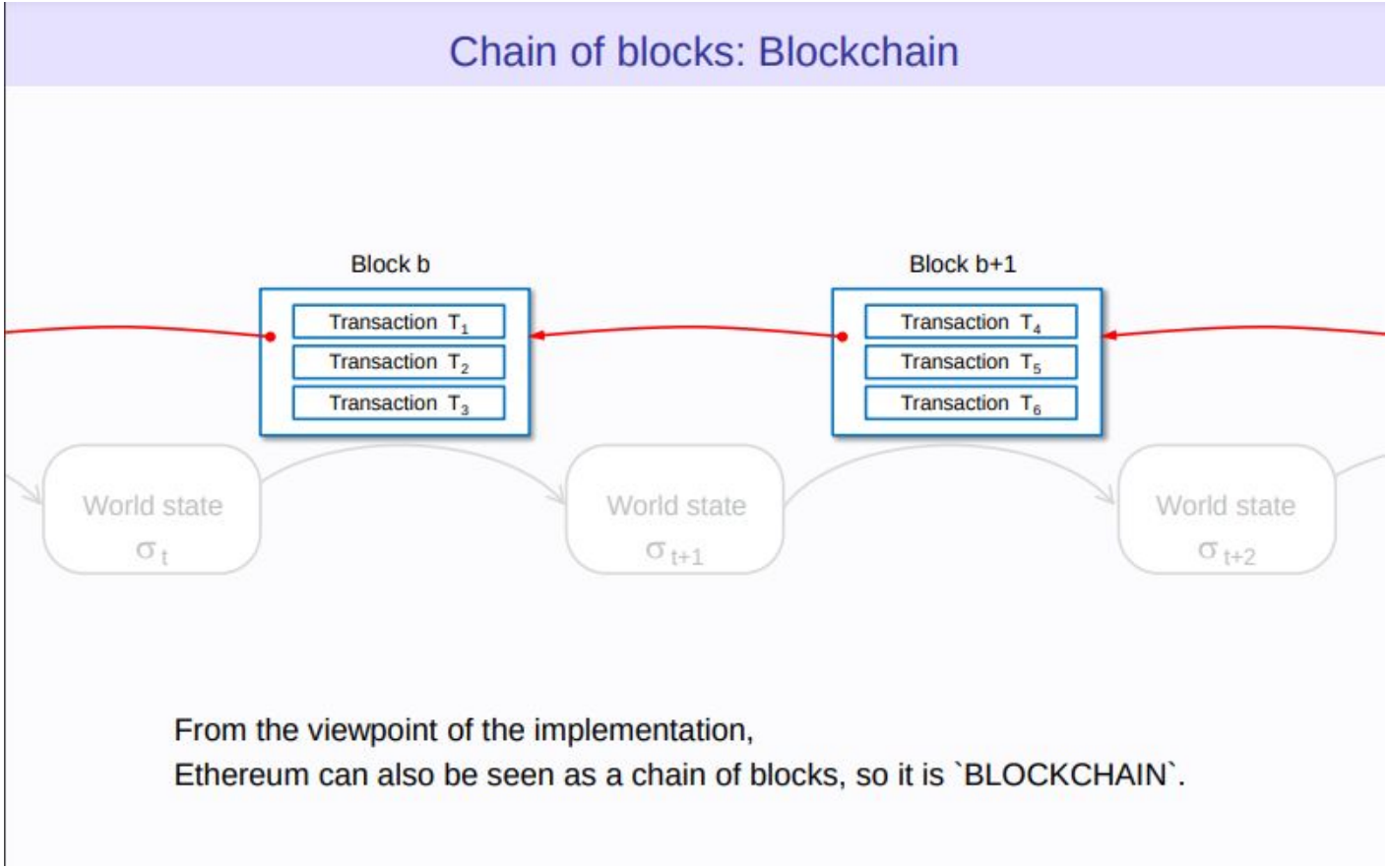
Components of Ethereum Network - Transactions



Components of Ethereum Network - Transactions

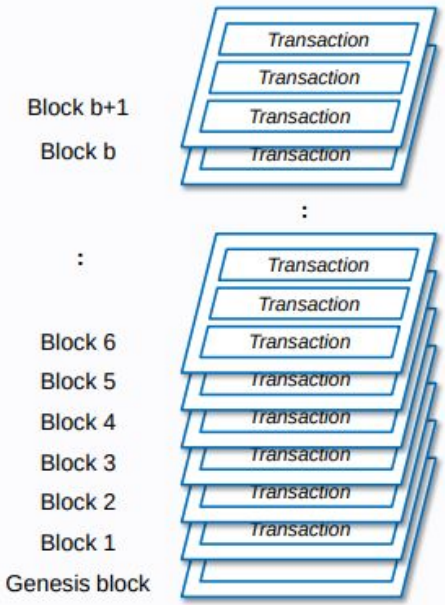


Components of Ethereum Network - Transactions



Components of Ethereum Network - Transactions

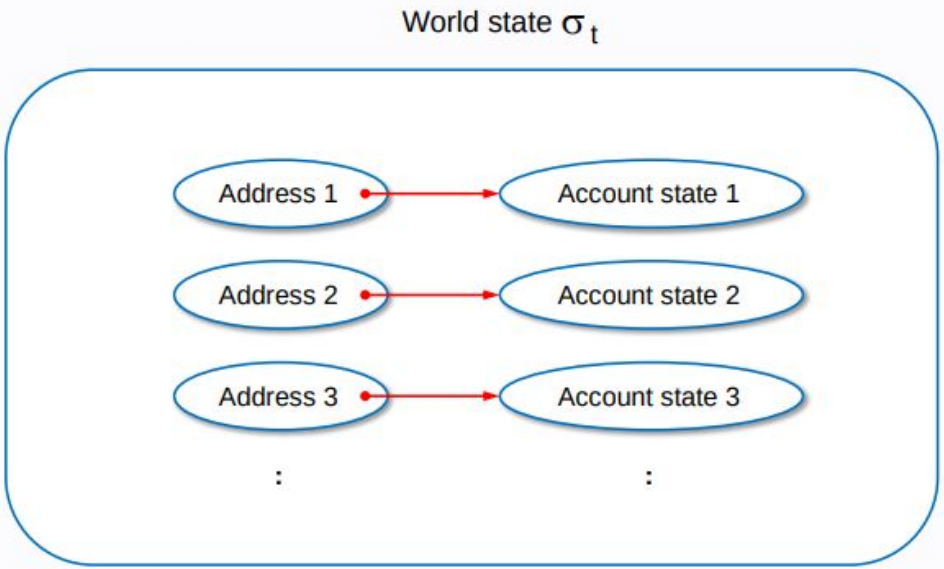
Stack of transactions : Ledger



From the viewpoint of the ledger,
Ethereum can also be seen as a stack of transactions.

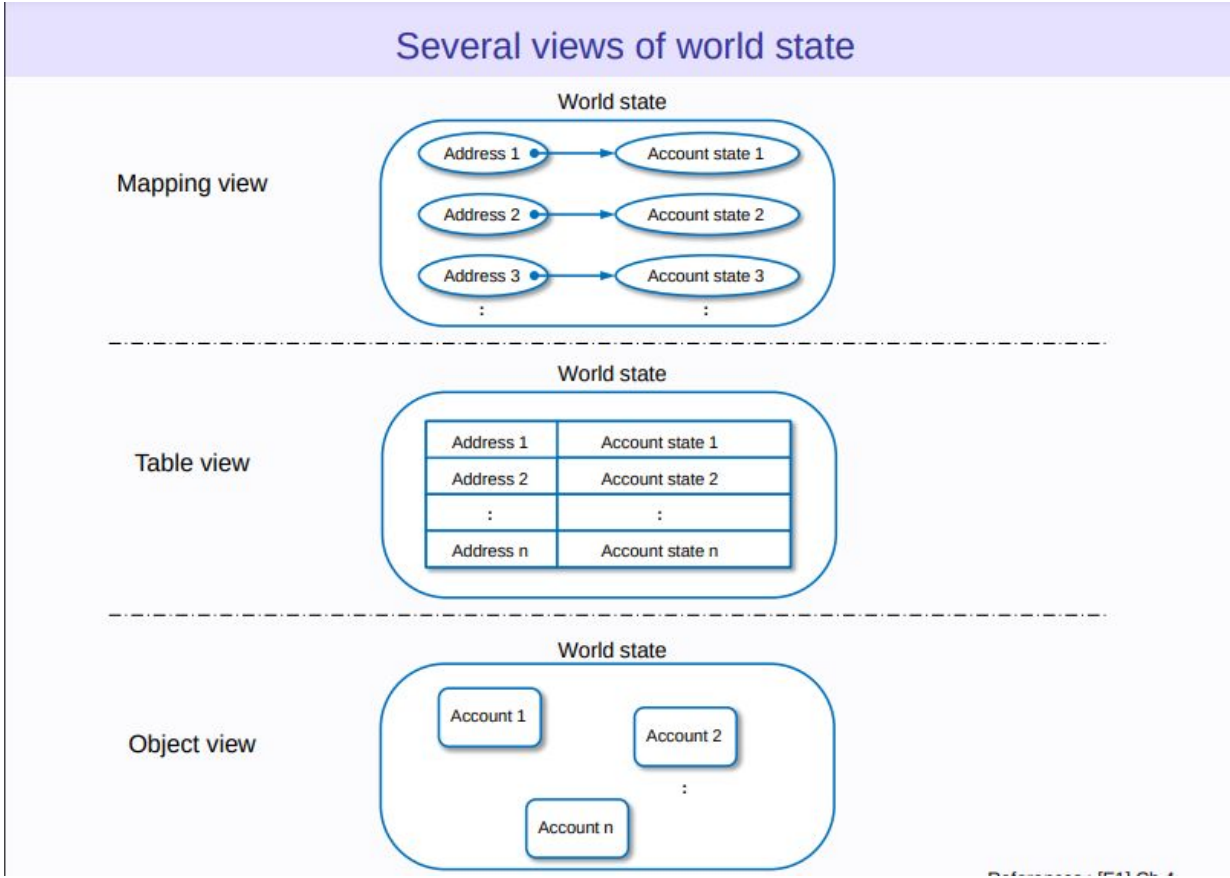
Components of Ethereum Network - Transactions

World state

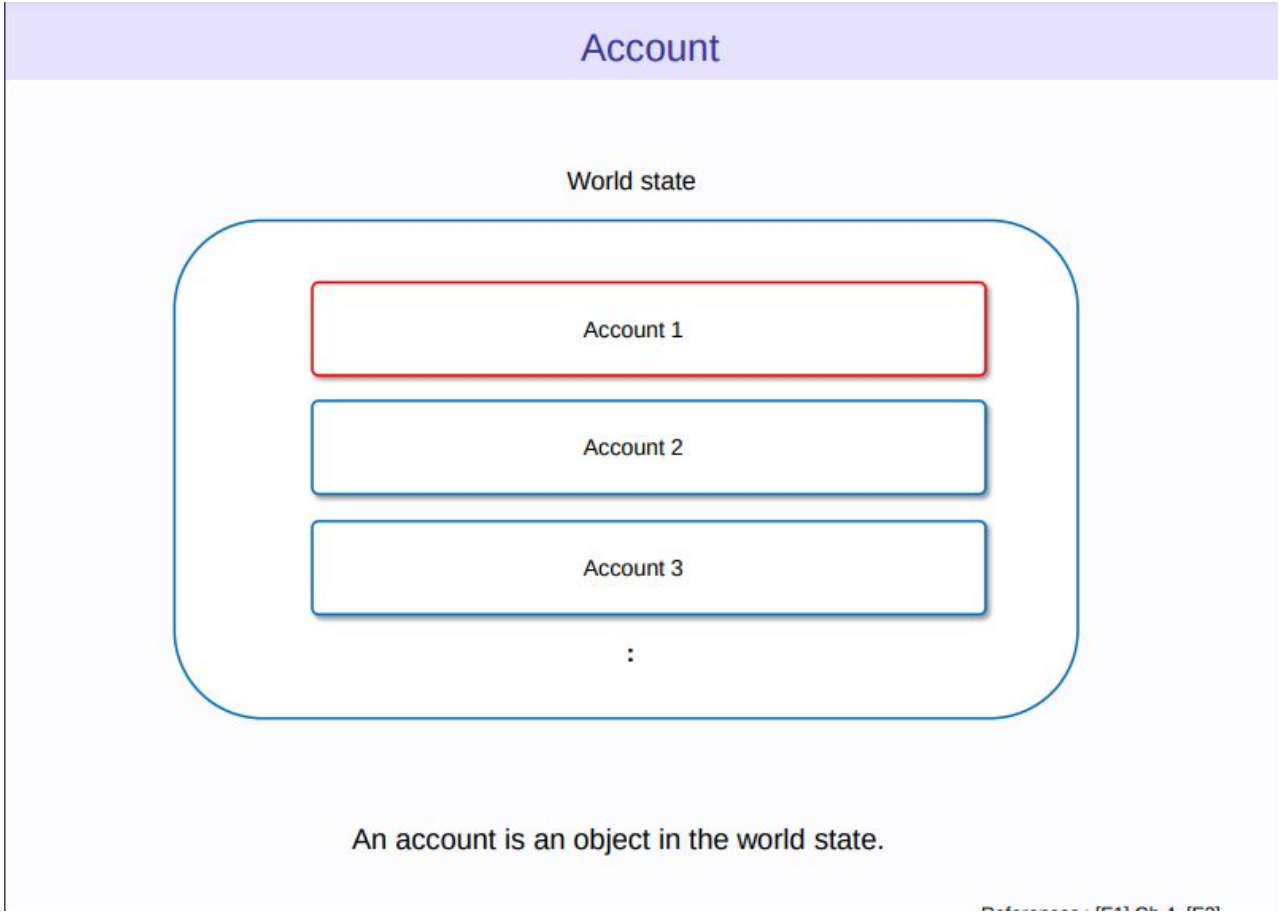


The world state is a mapping between address and account state.

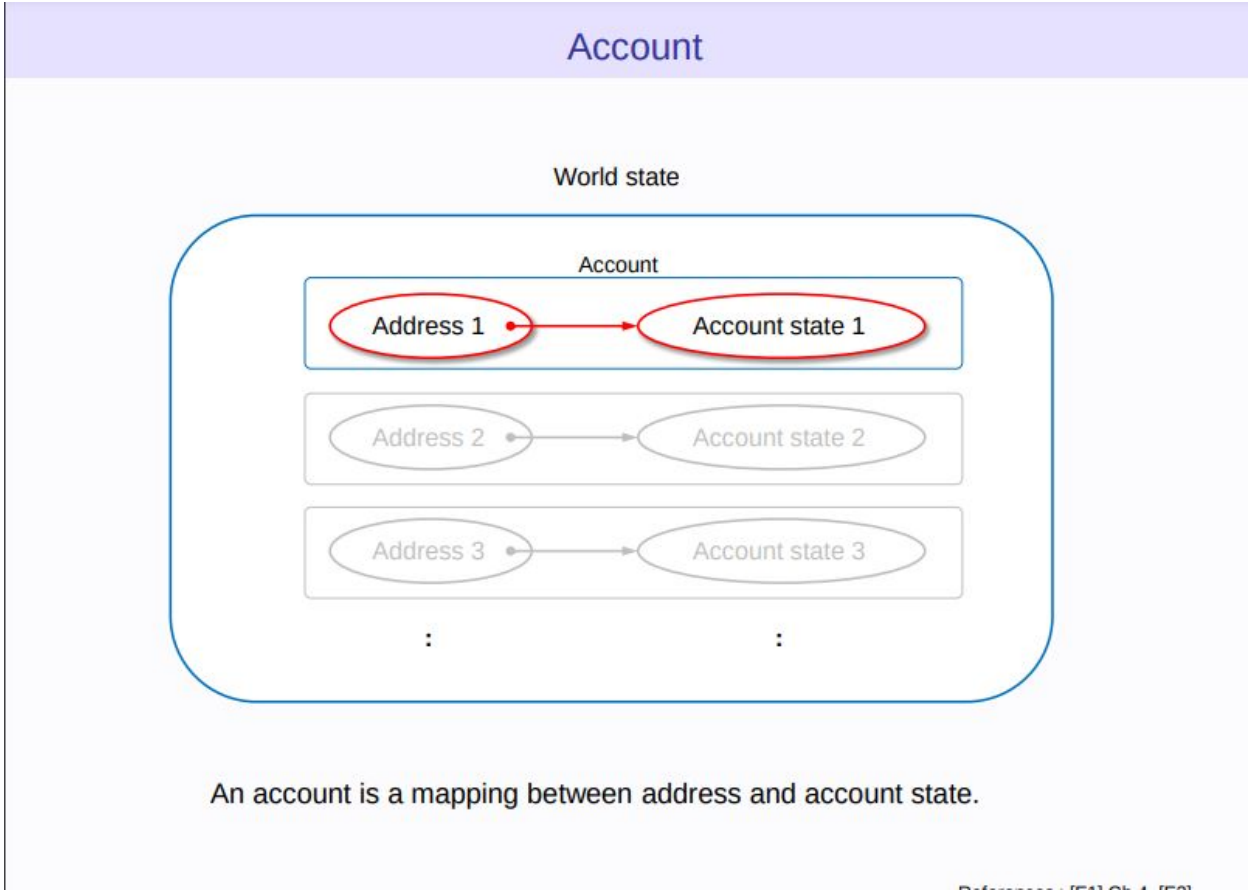
Components of Ethereum Network - Transactions



Components of Ethereum Network - Transactions

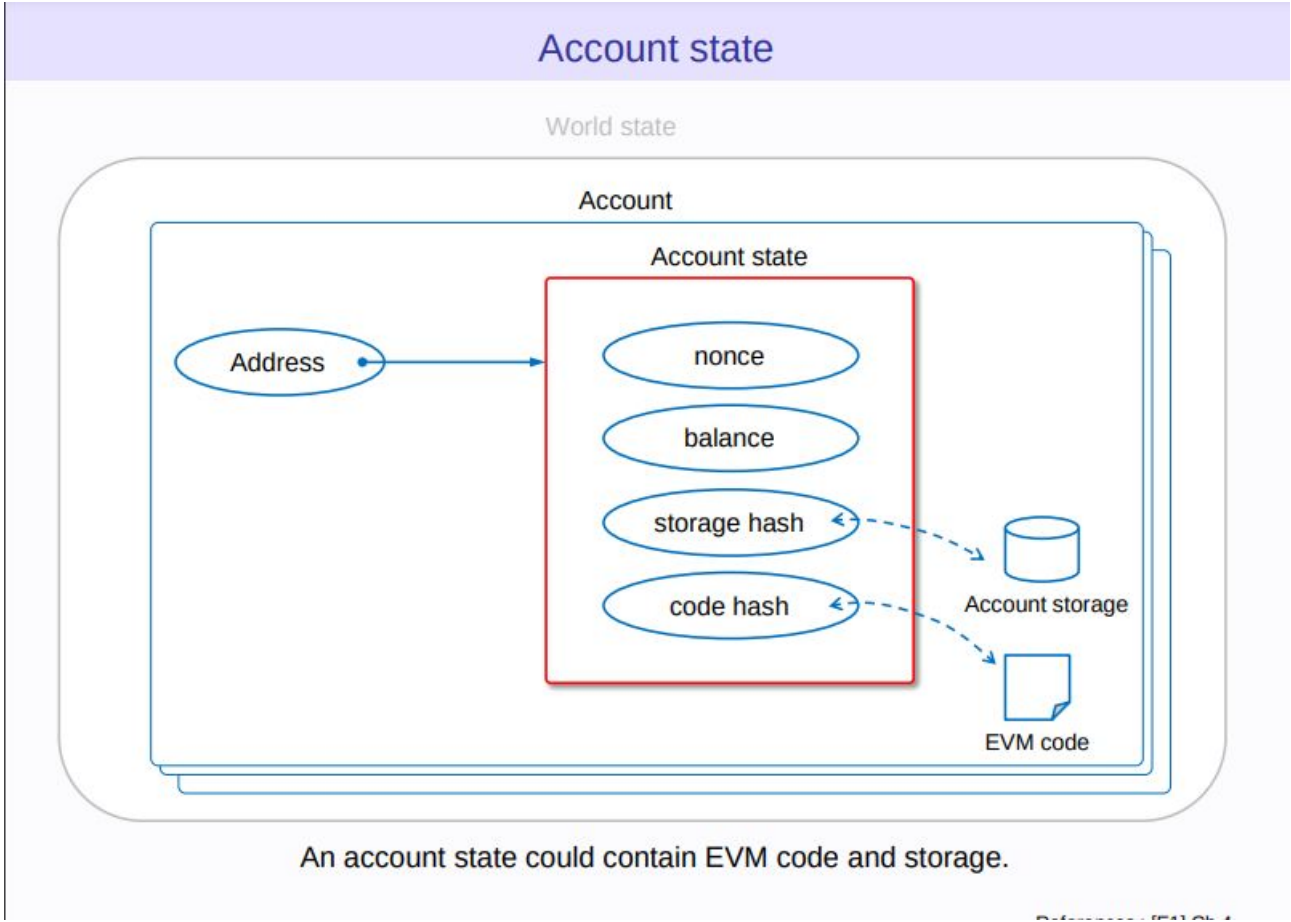


Components of Ethereum Network - Transactions

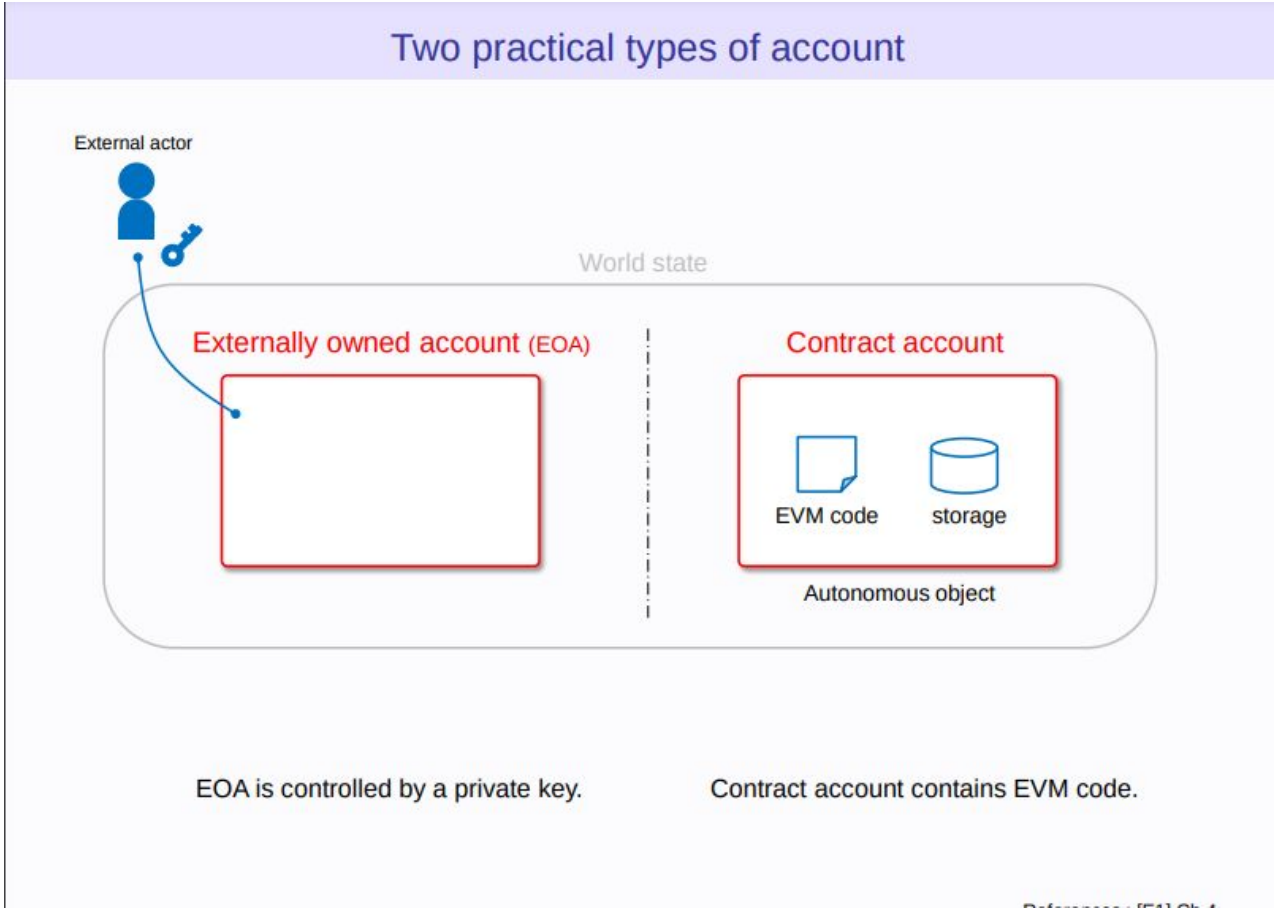


Courtesy : Takenobu T

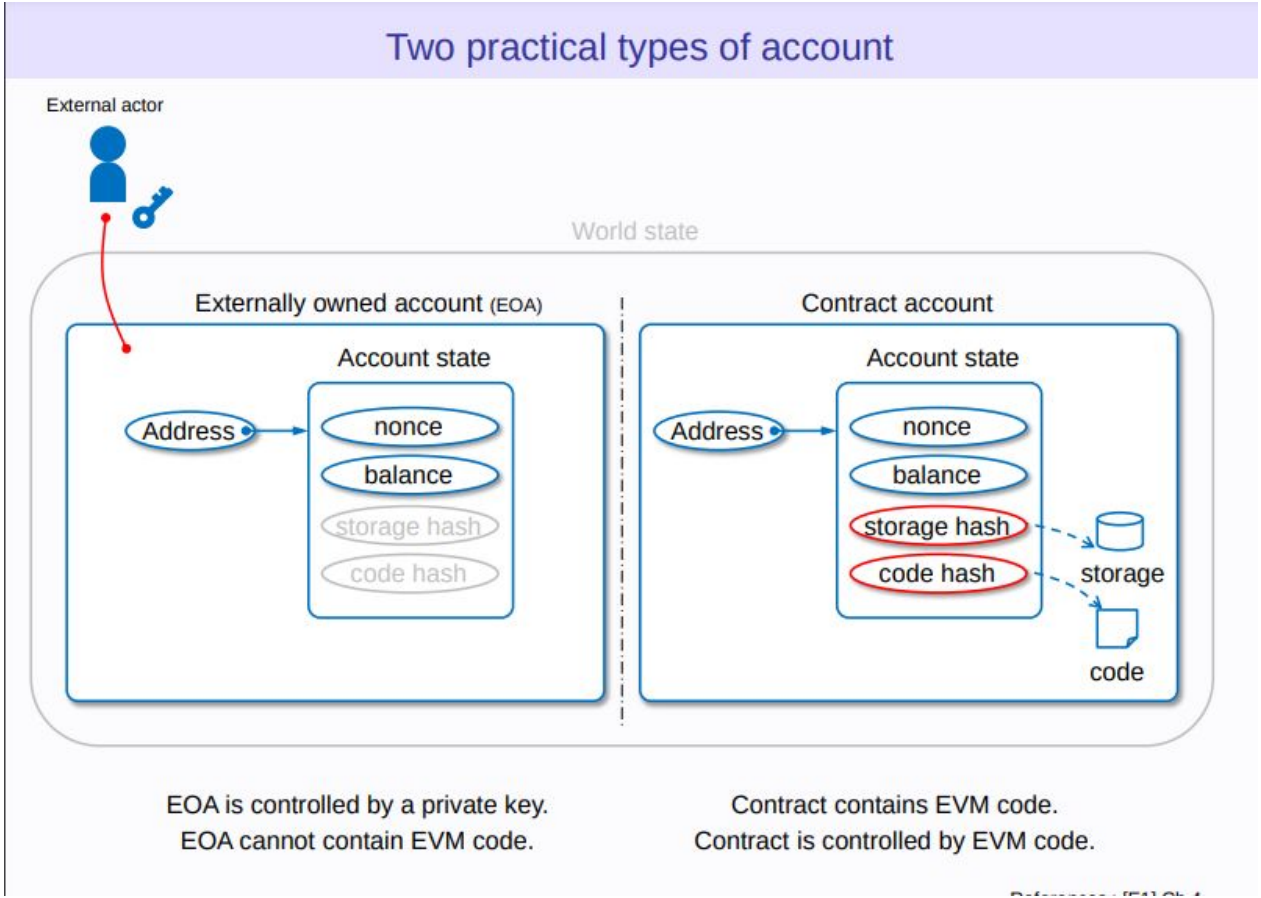
Components of Ethereum Network - Transactions



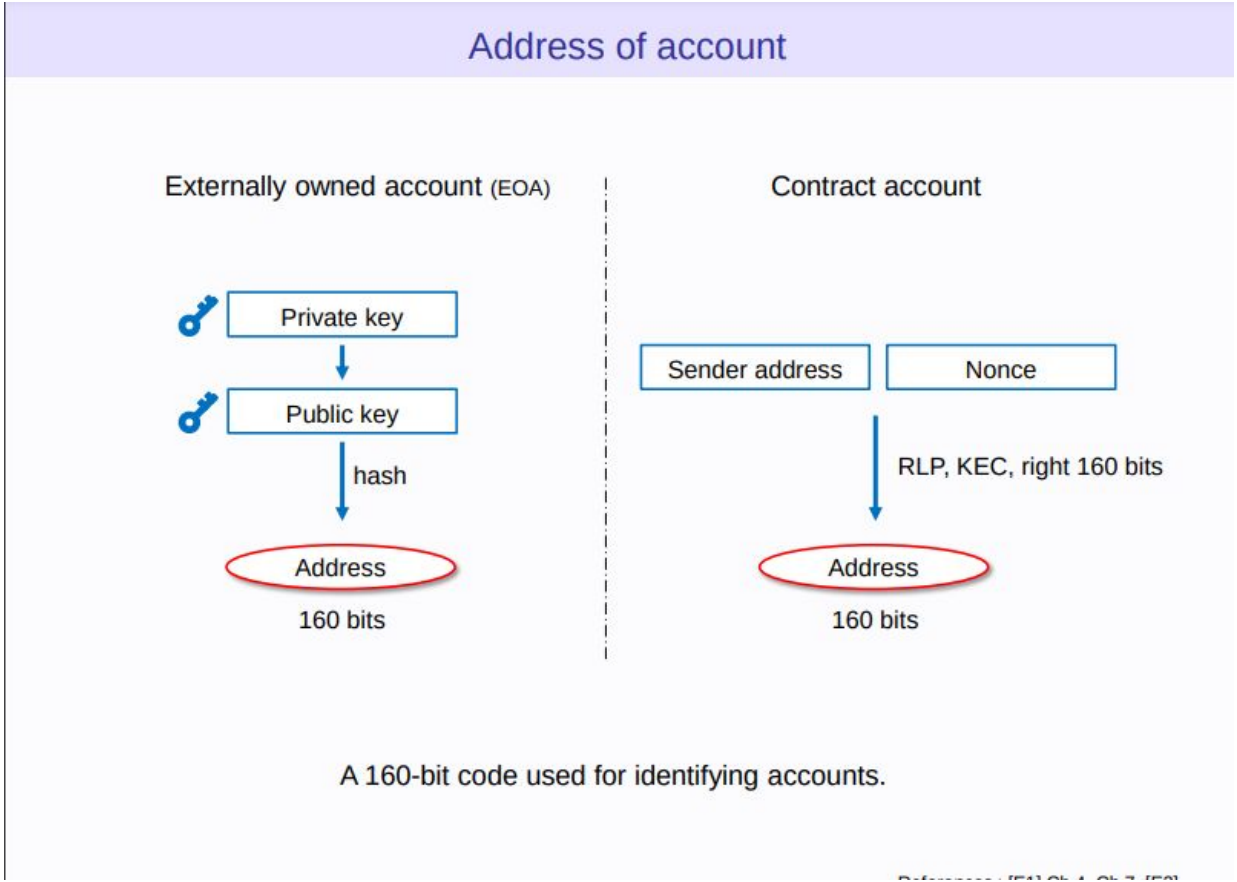
Components of Ethereum Network - Transactions



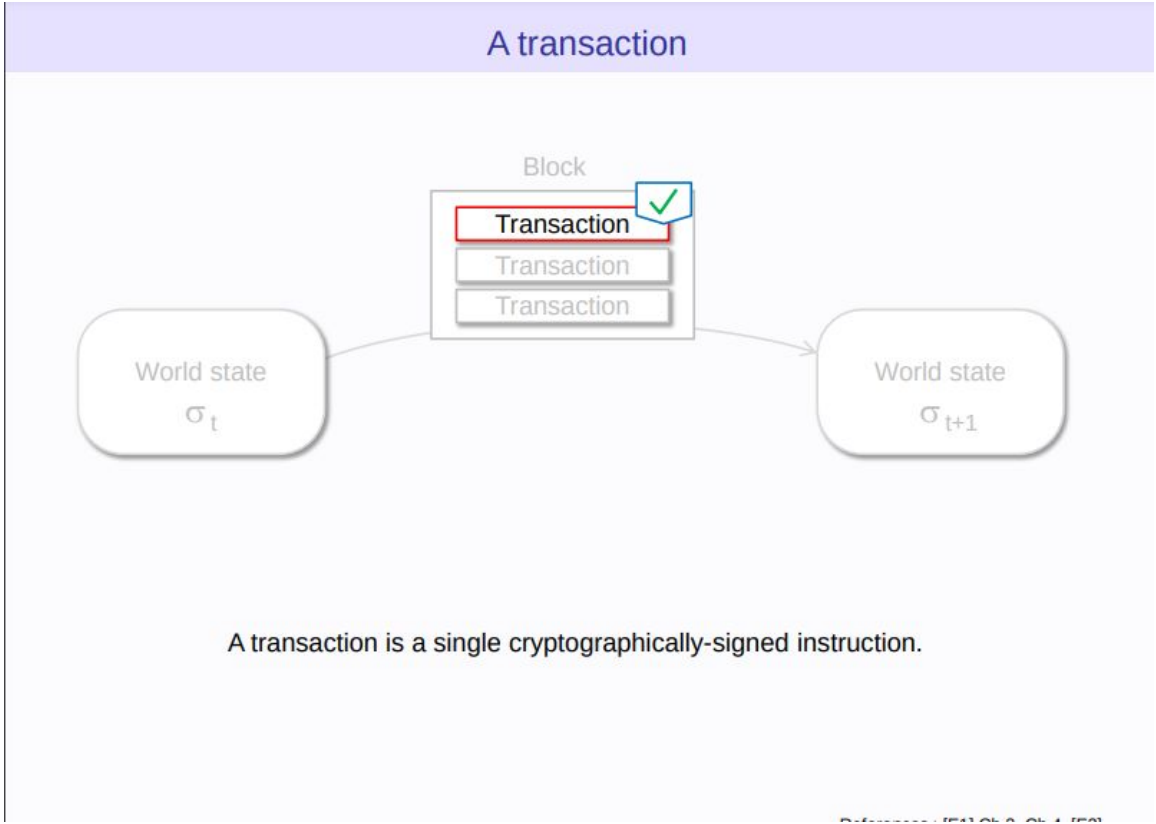
Components of Ethereum Network - Transactions



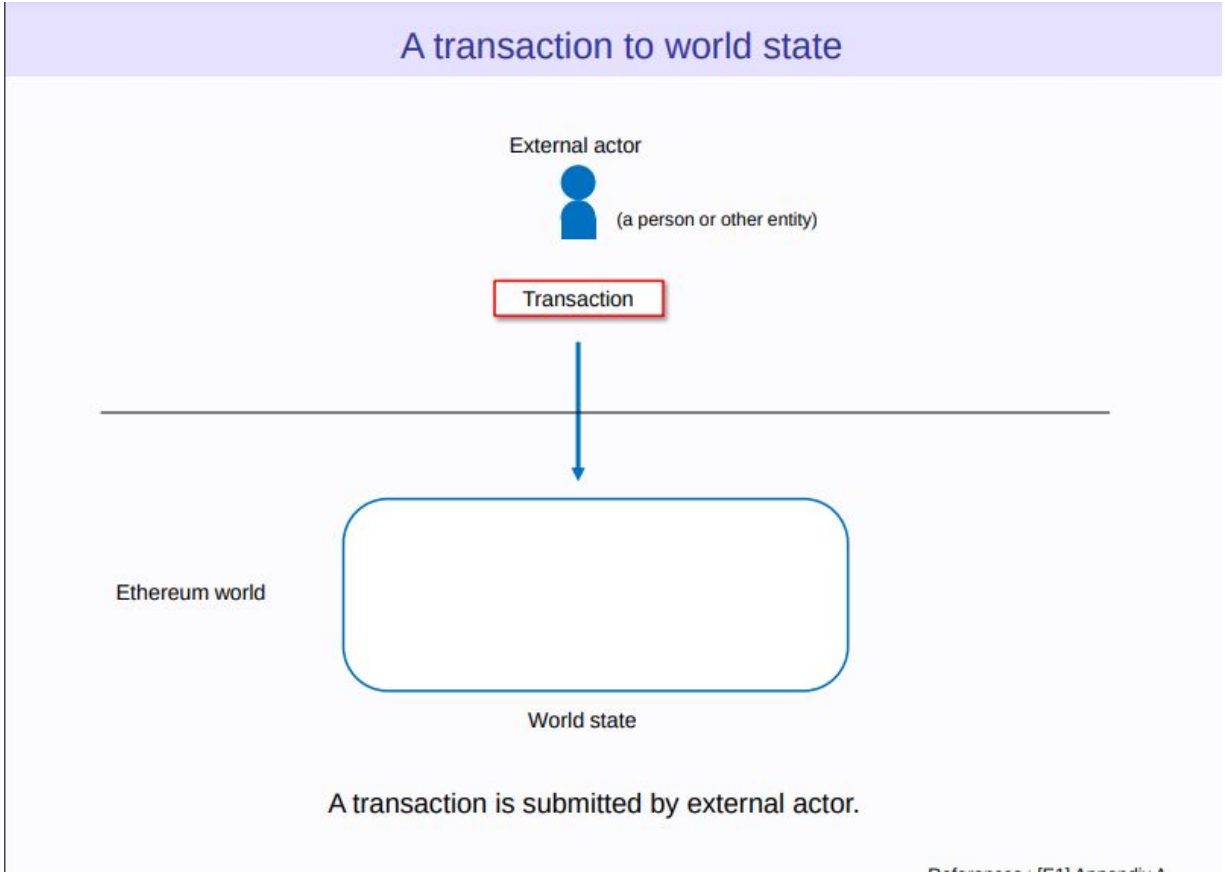
Components of Ethereum Network - Transactions



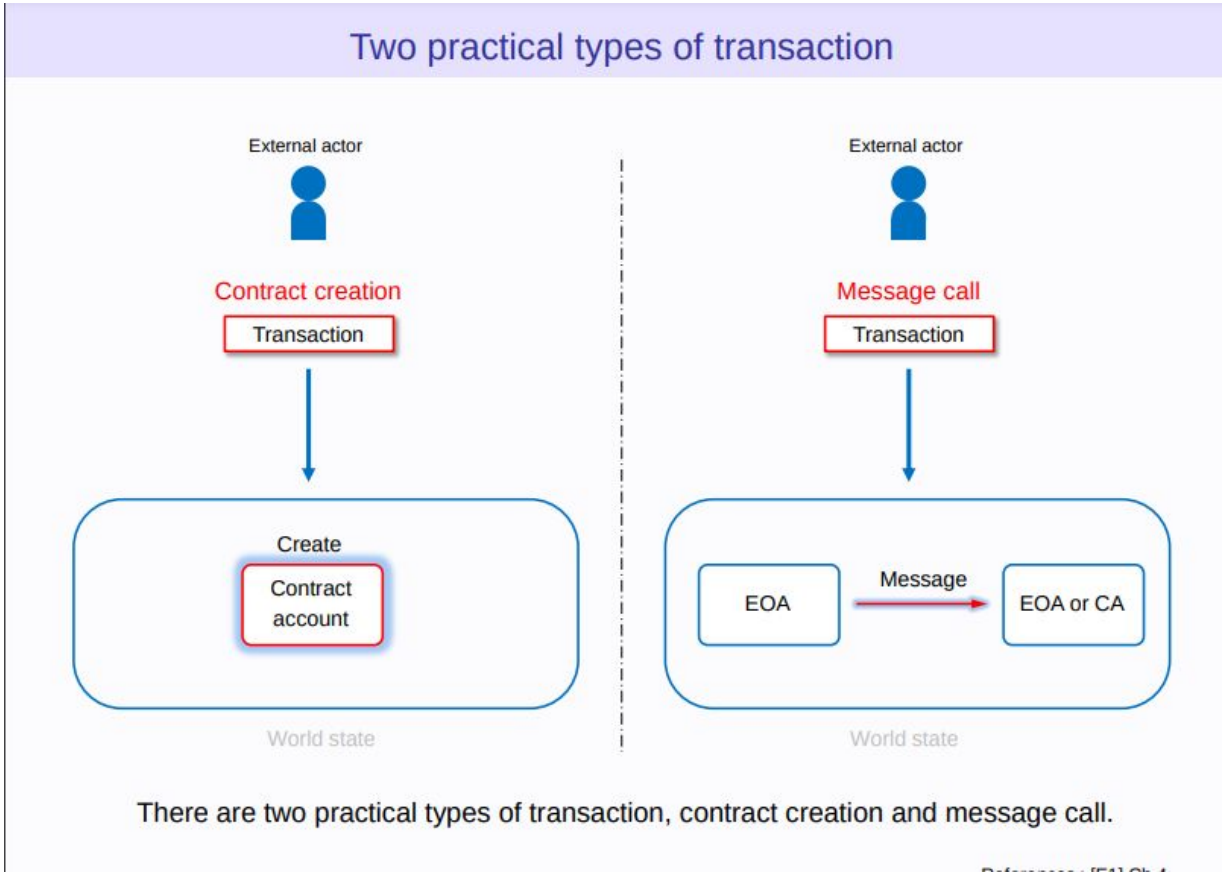
Components of Ethereum Network - Transactions



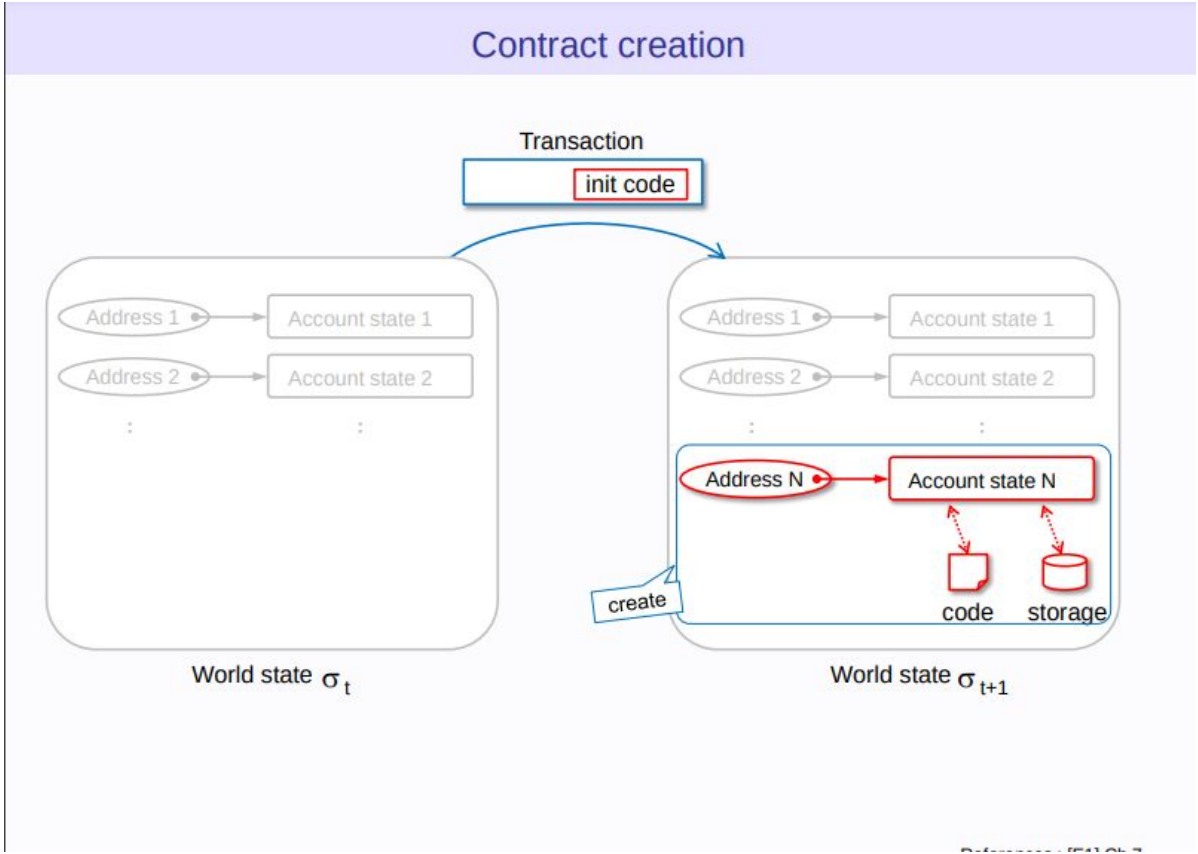
Components of Ethereum Network - Transactions



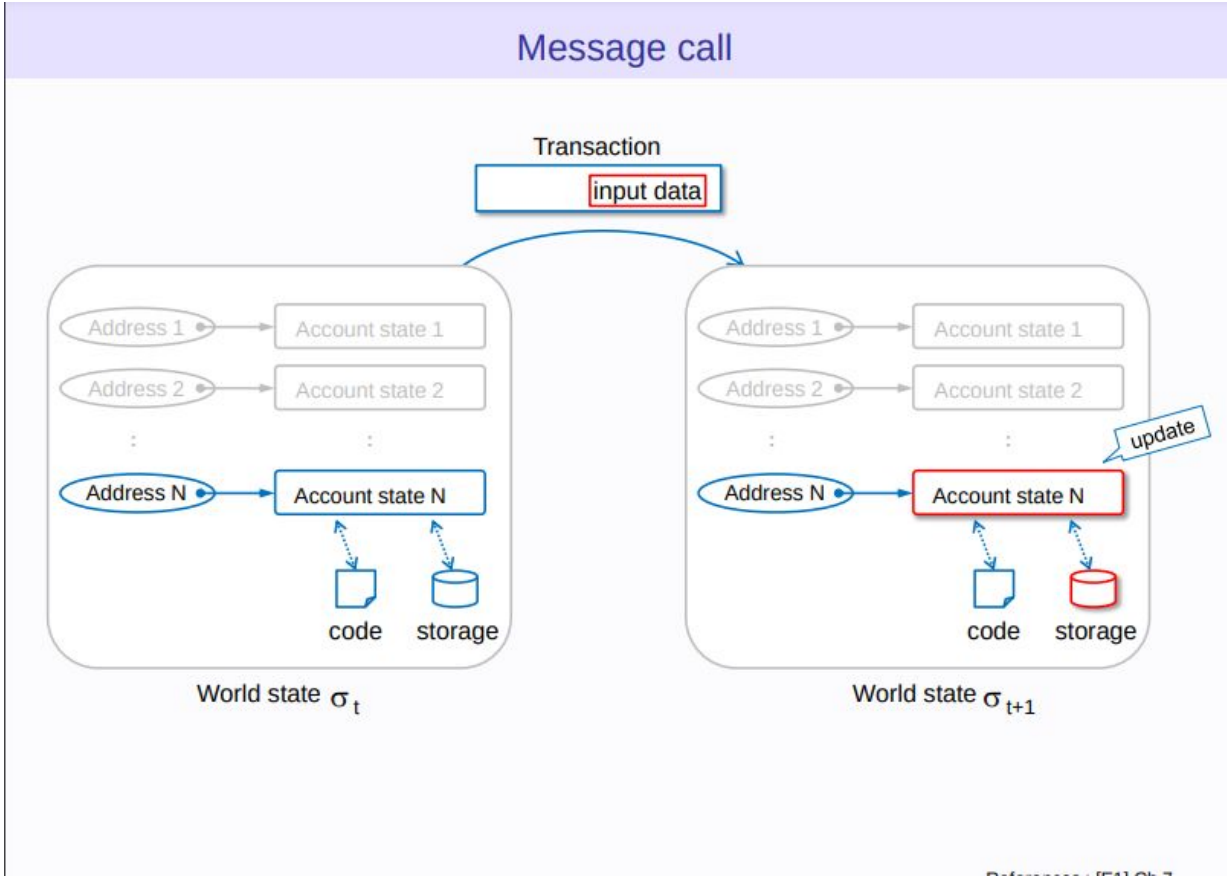
Components of Ethereum Network - Transactions



Components of Ethereum Network - Transactions

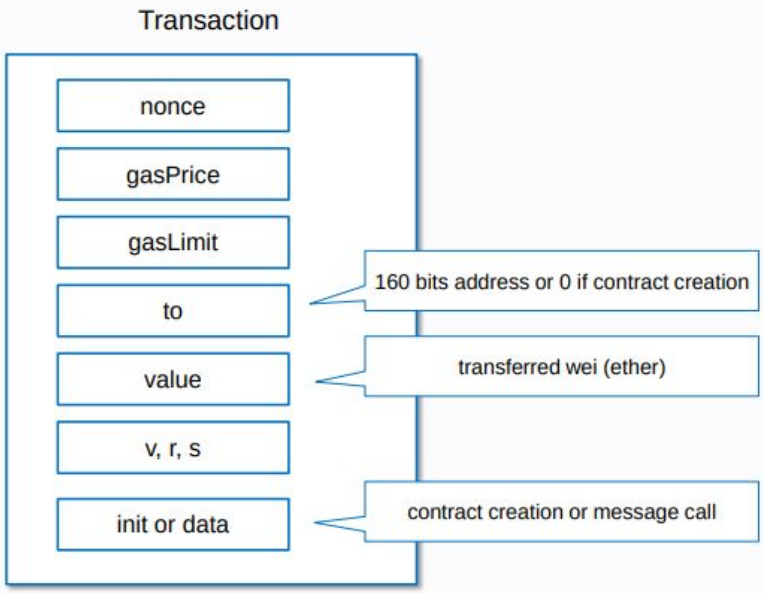


Components of Ethereum Network - Transactions



Components of Ethereum Network - Transactions

Field of a transaction



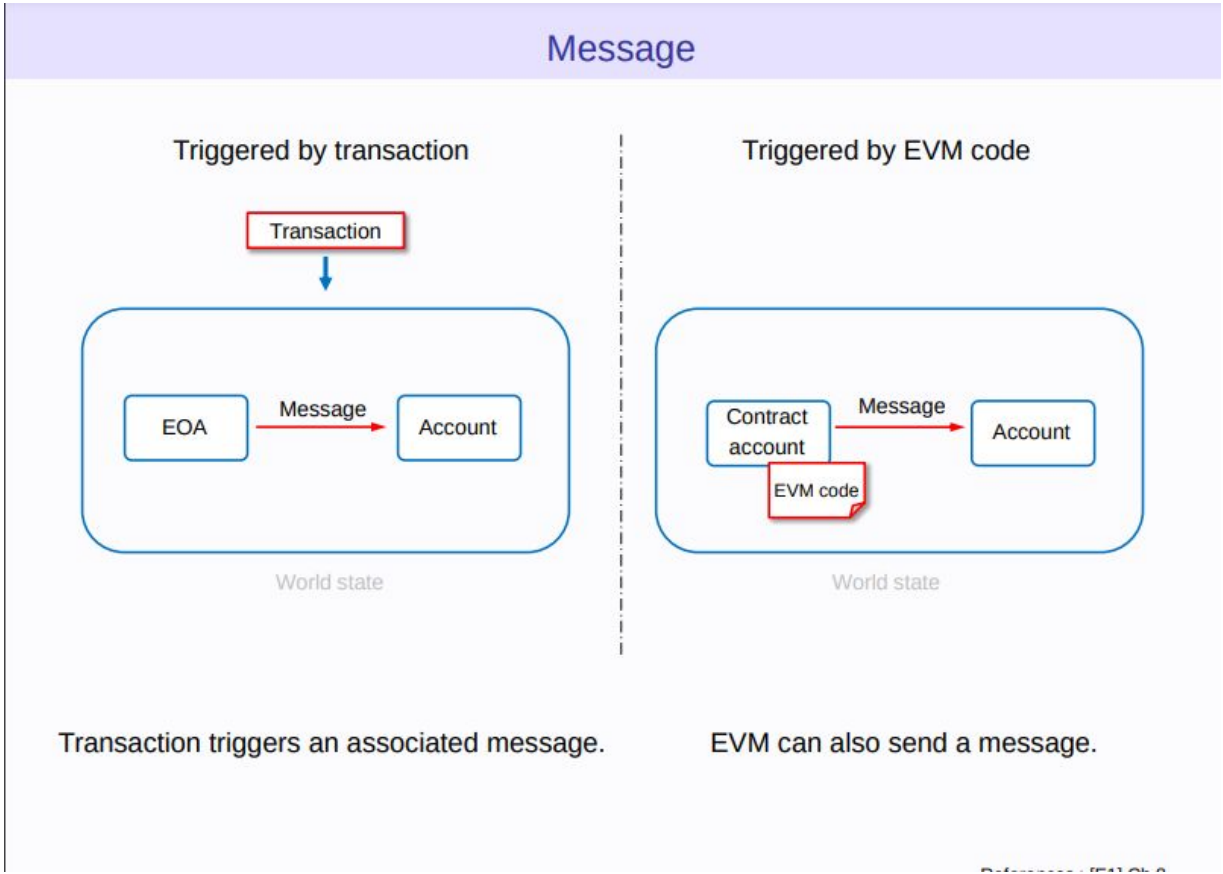
Components of Ethereum Network - Transactions

Message

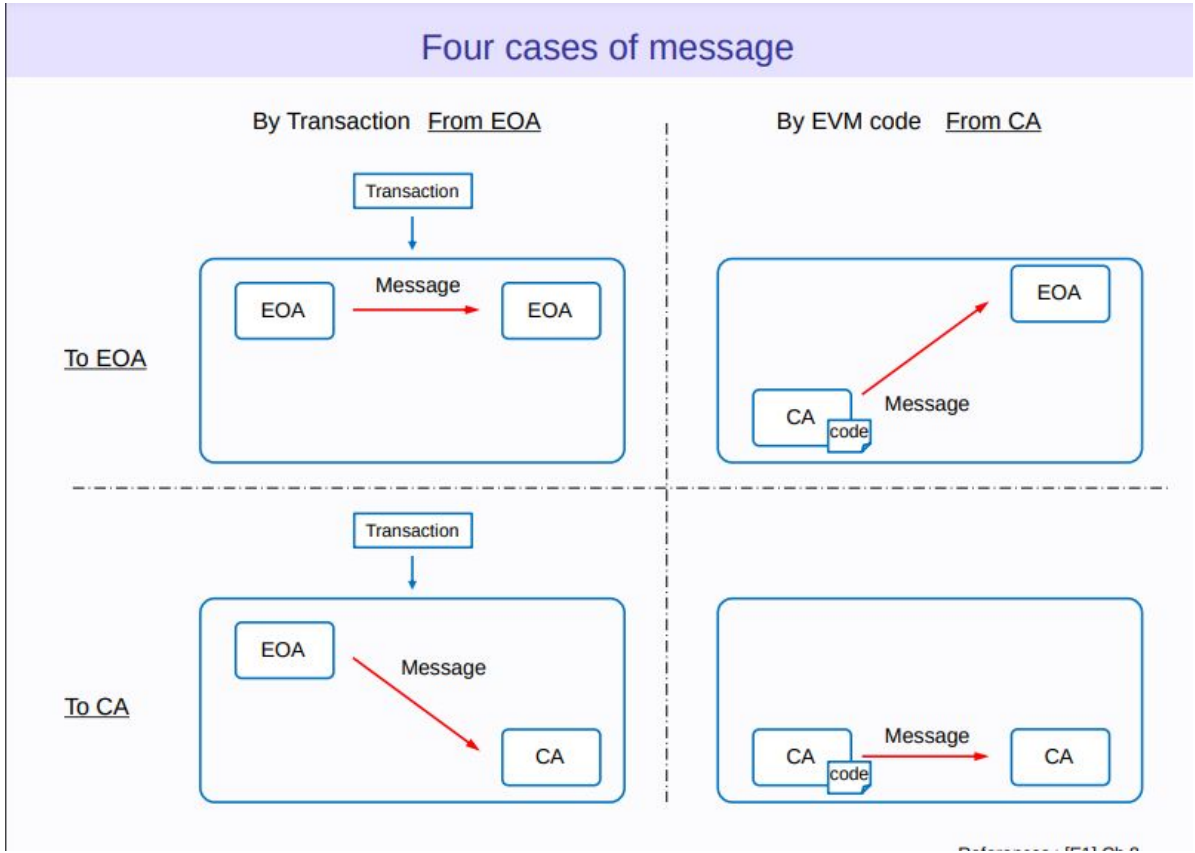
Message is passed between two Accounts.

Message is Data (as a set of bytes) and Value (specified as Ether) .

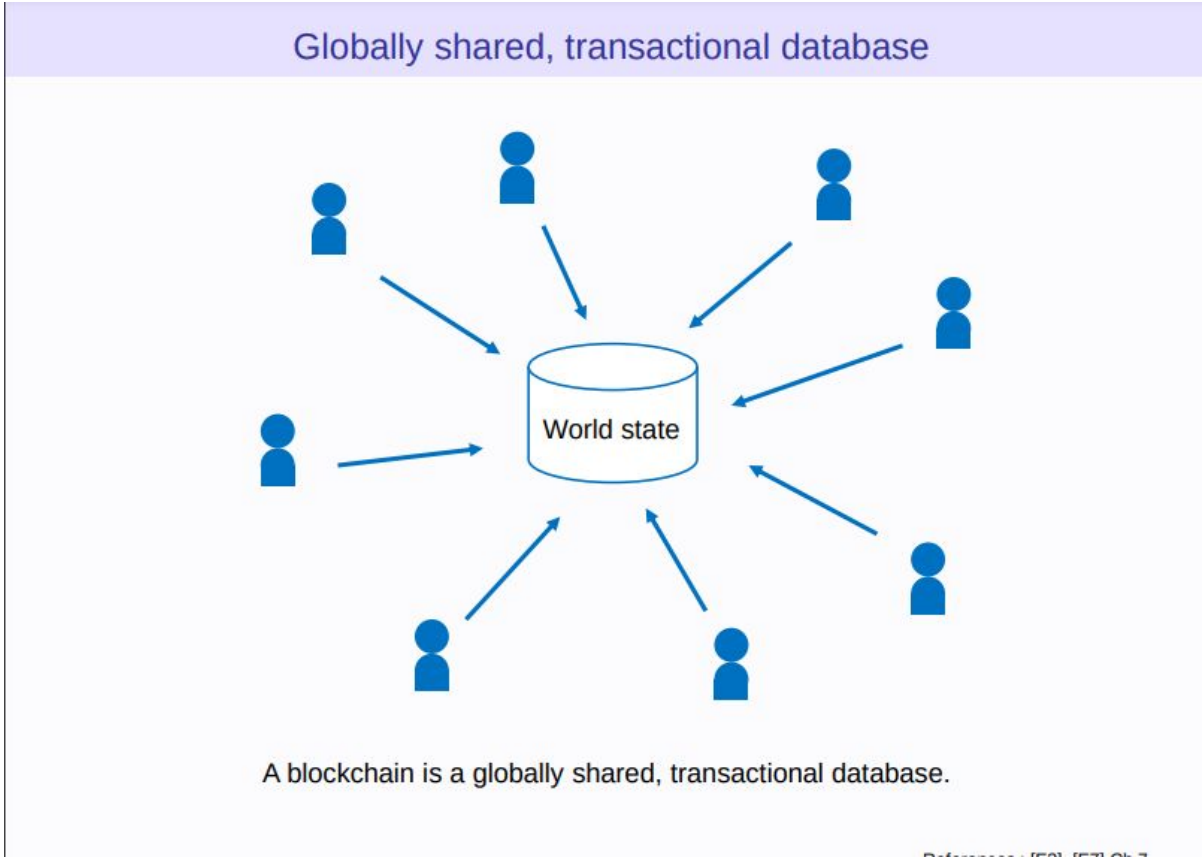
Components of Ethereum Network - Transactions



Components of Ethereum Network - Transactions

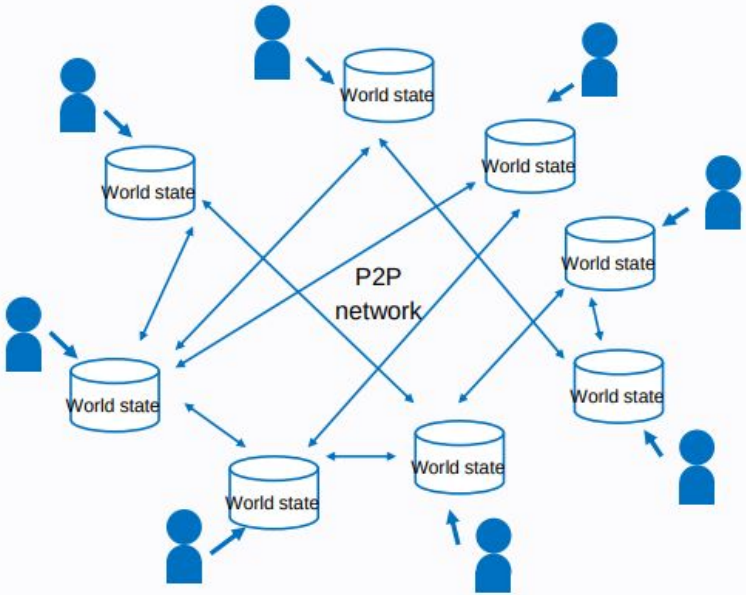


Components of Ethereum Network - Transactions



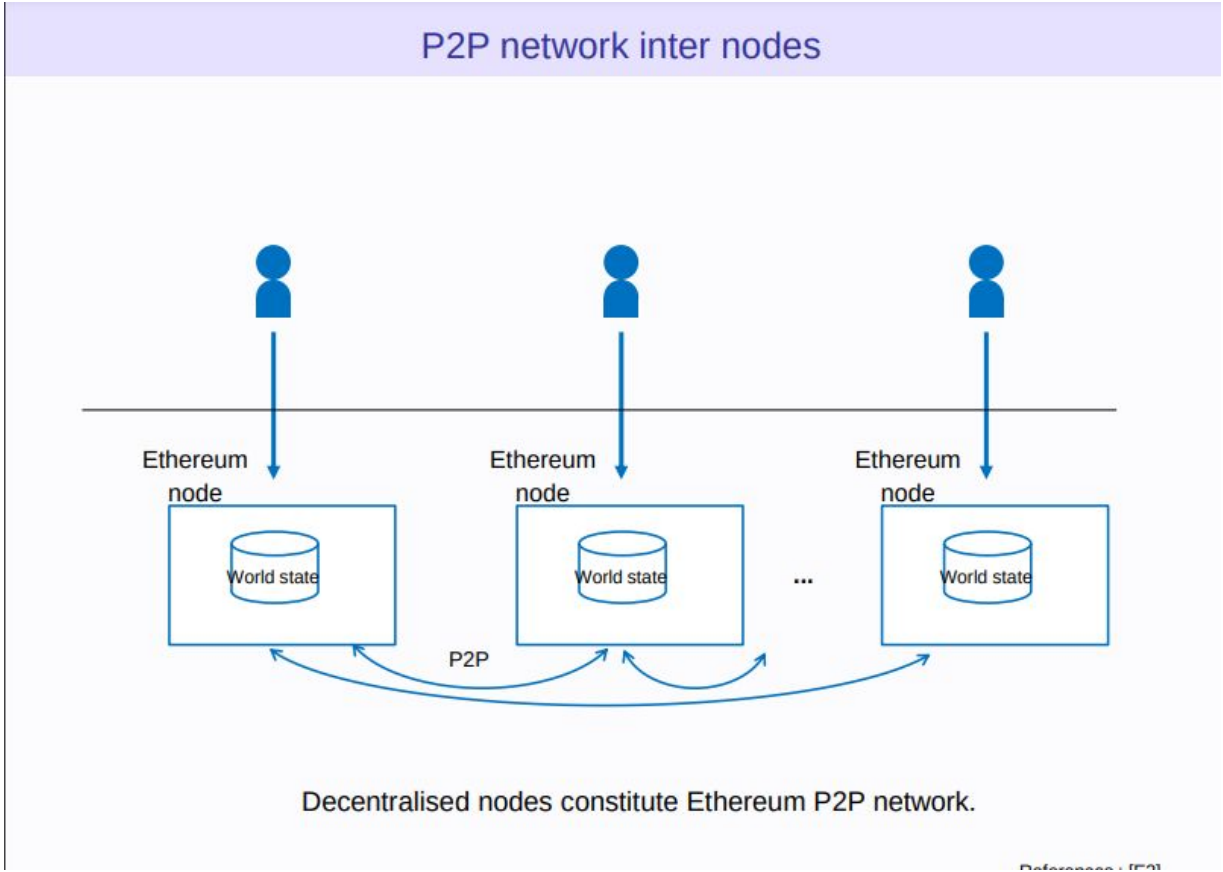
Components of Ethereum Network - Transactions

Decentralised database

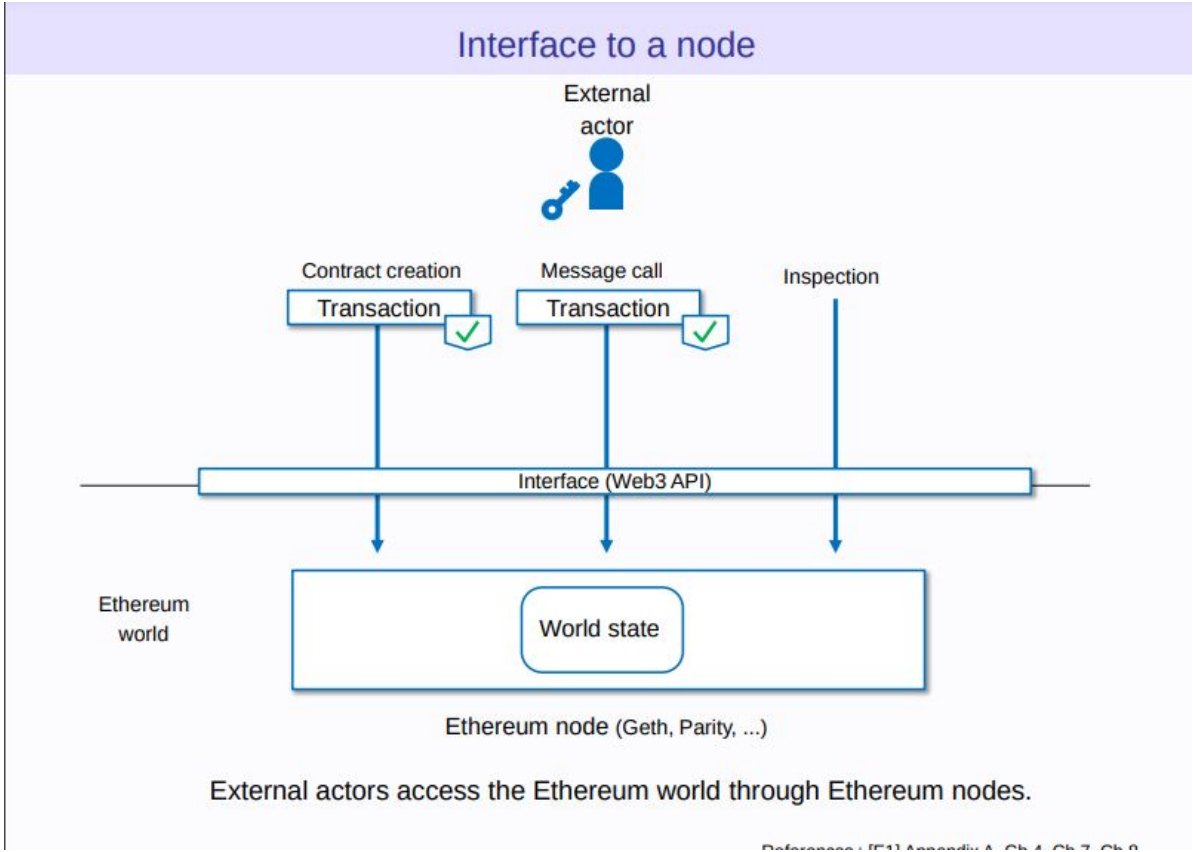


A blockchain is a globally shared, decentralised, transactional database.

Components of Ethereum Network - Transactions



Components of Ethereum Network - Transactions



Components of Ethereum Network - Transactions

Atomicity of transaction

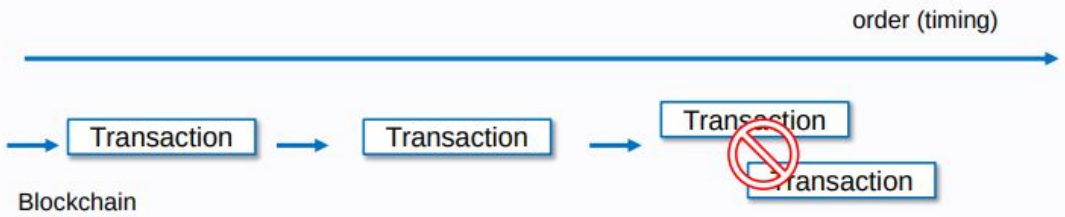
A transaction is an atomic operation. Can't divide or interrupt.

Transaction or Transaction

That is, All (complete done) or Nothing (zero effect).

Components of Ethereum Network - Transactions

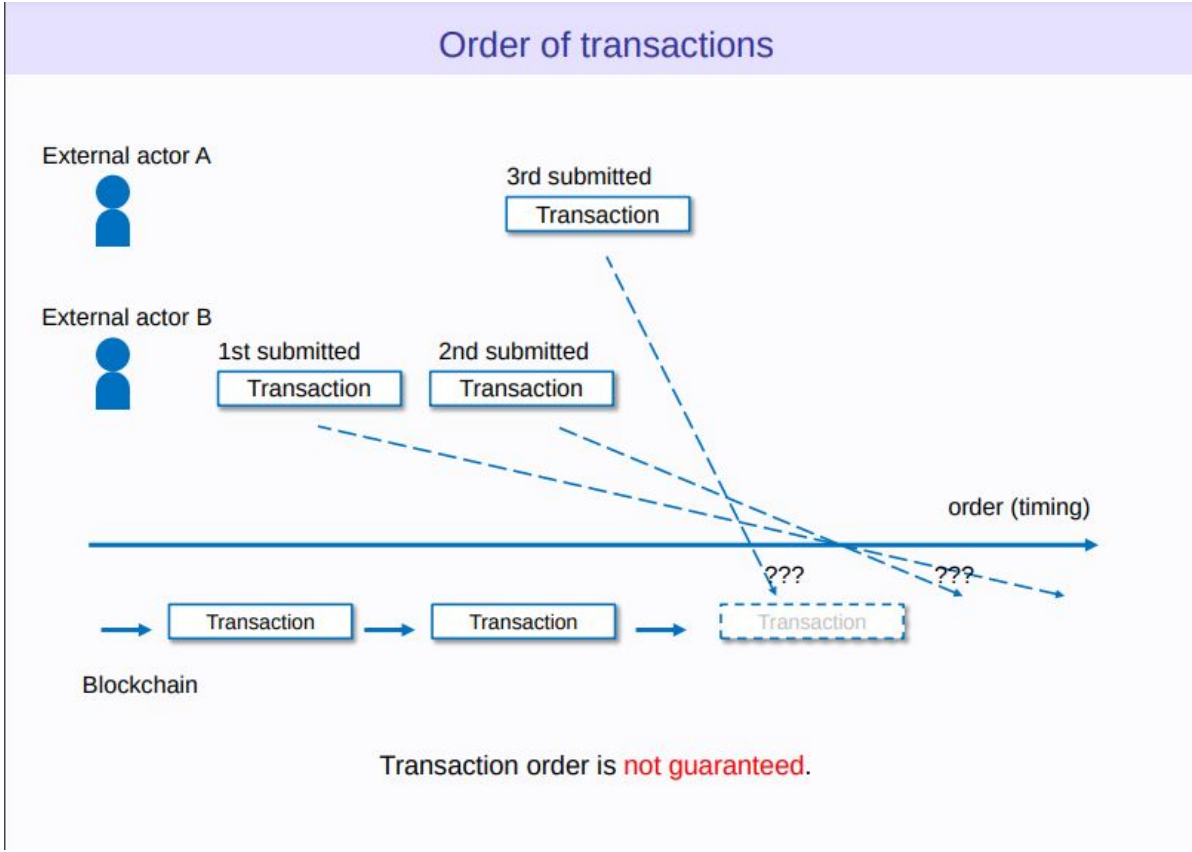
Order of transactions



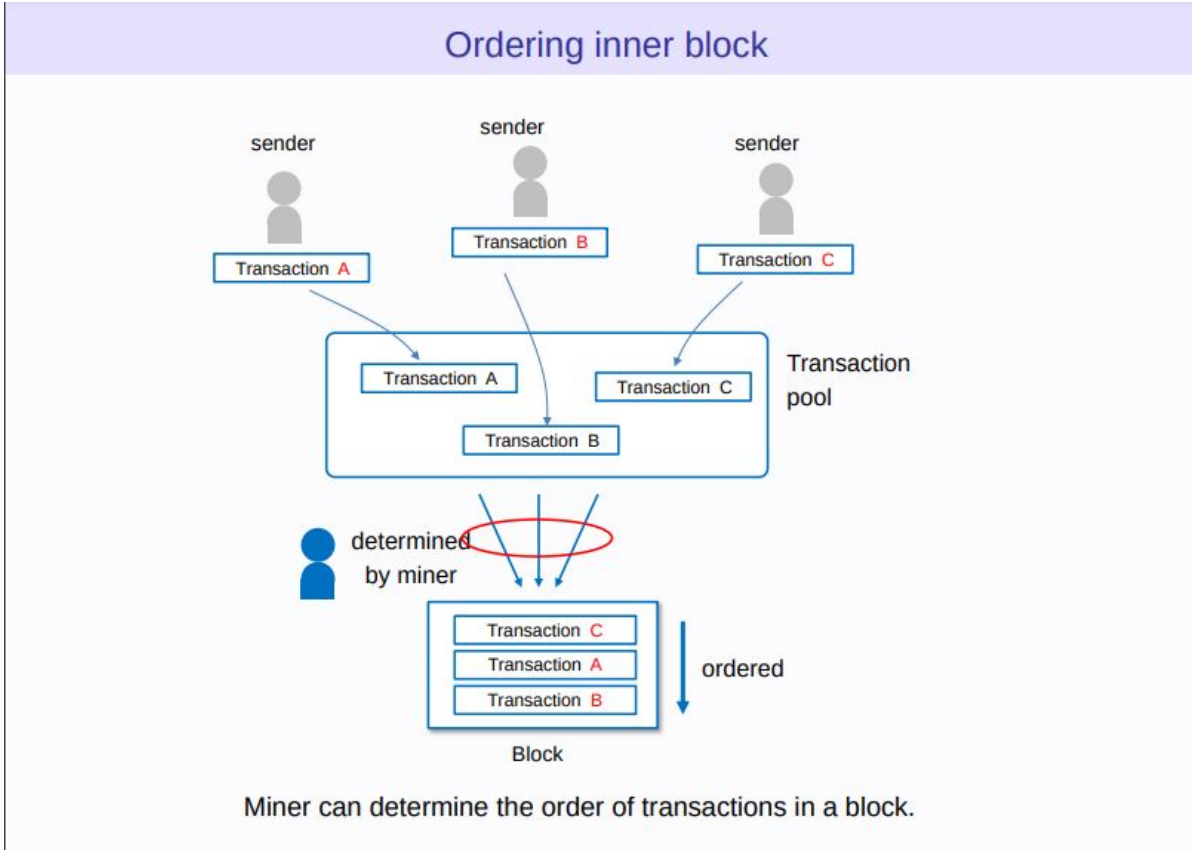
Transactions cannot be overlapped.

Transactions must be executed sequentially.

Components of Ethereum Network - Transactions

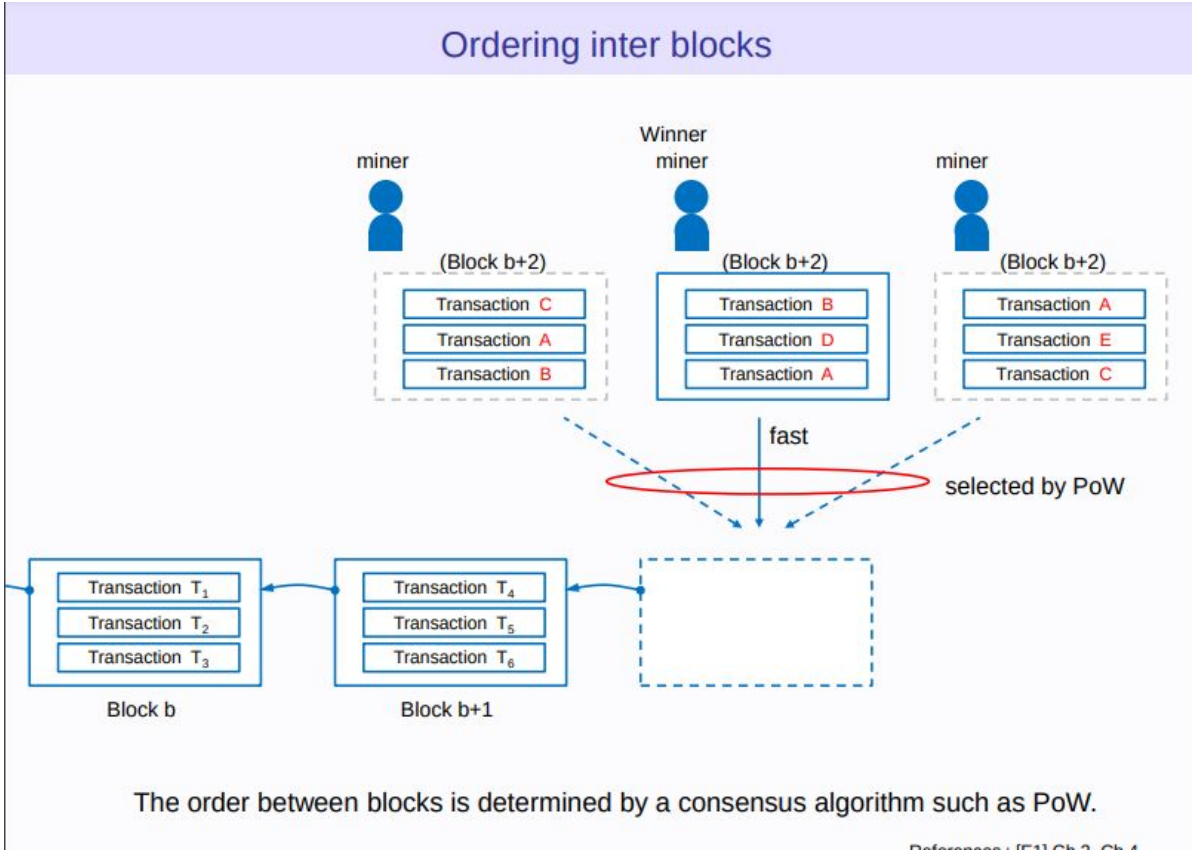


Components of Ethereum Network - Transactions

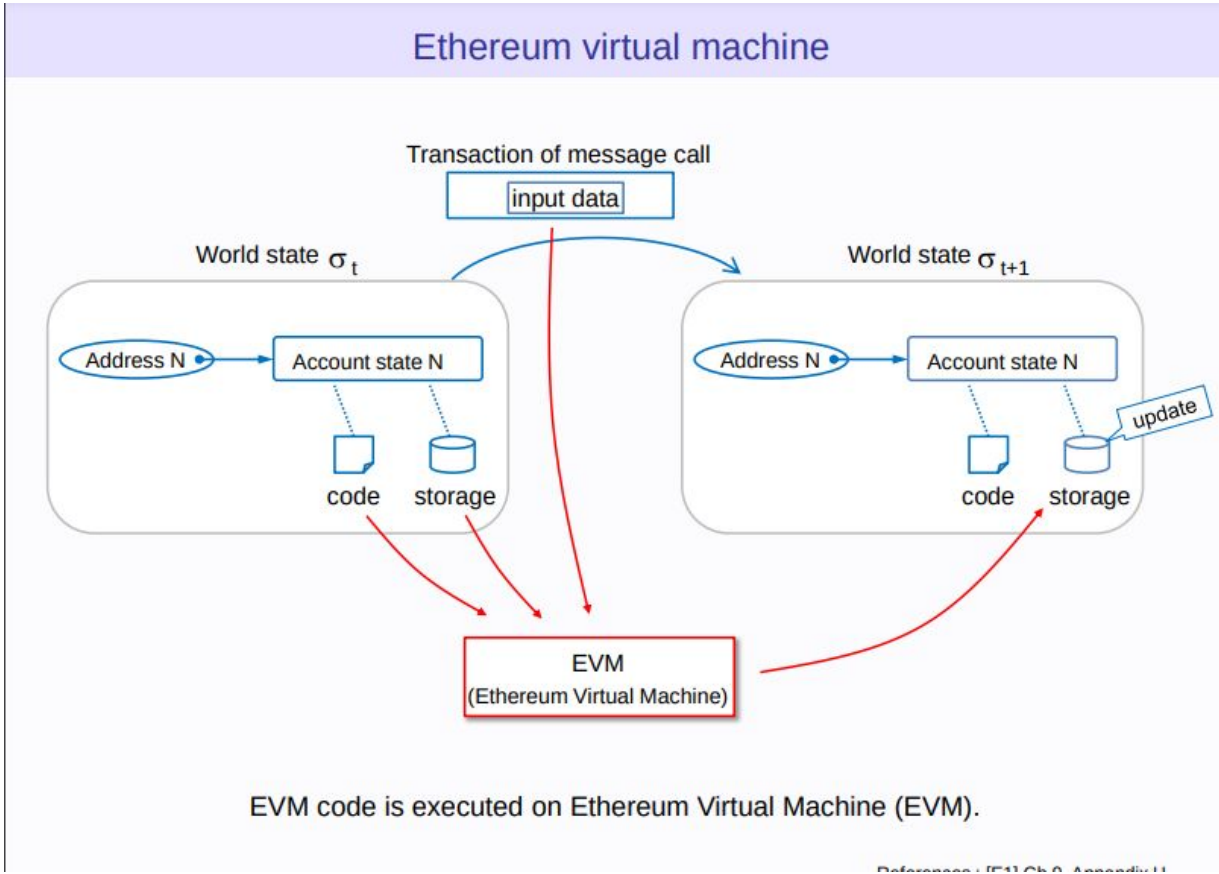


Courtesy : Takenobu T

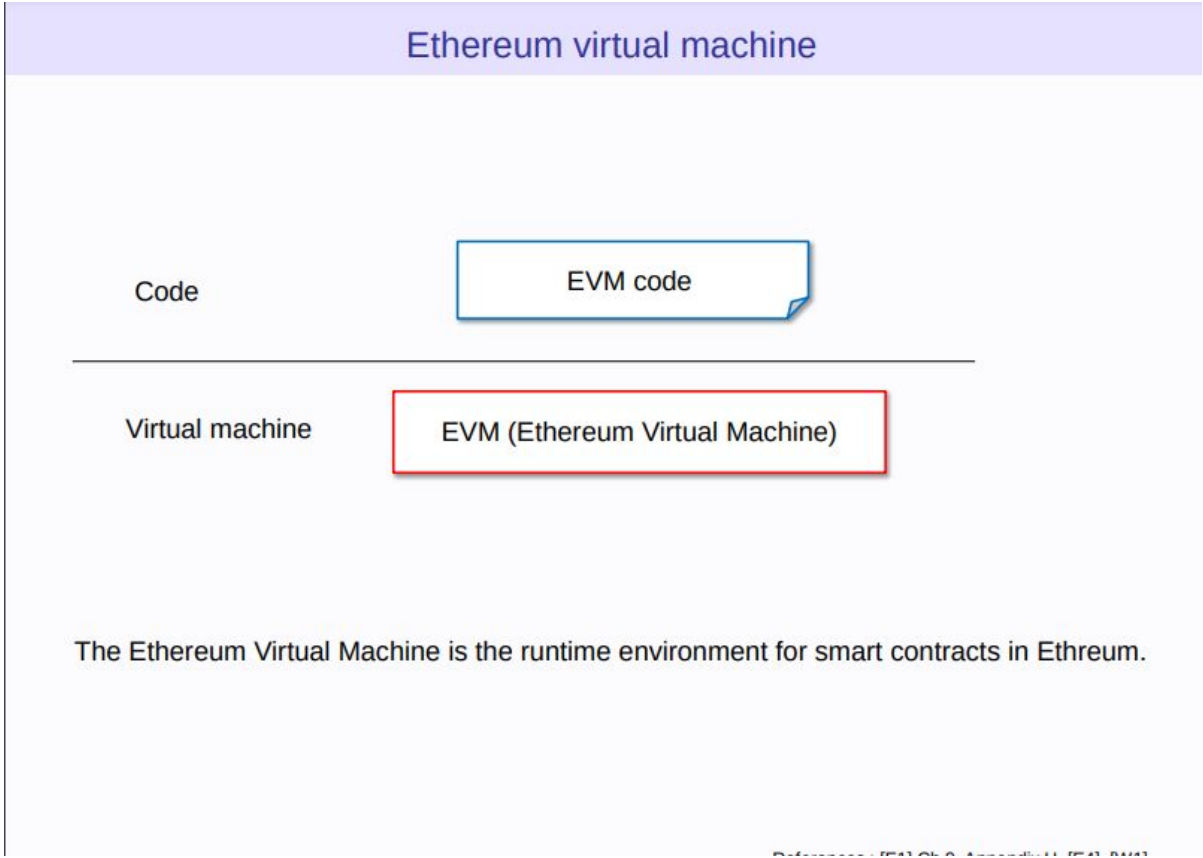
Components of Ethereum Network - Transactions



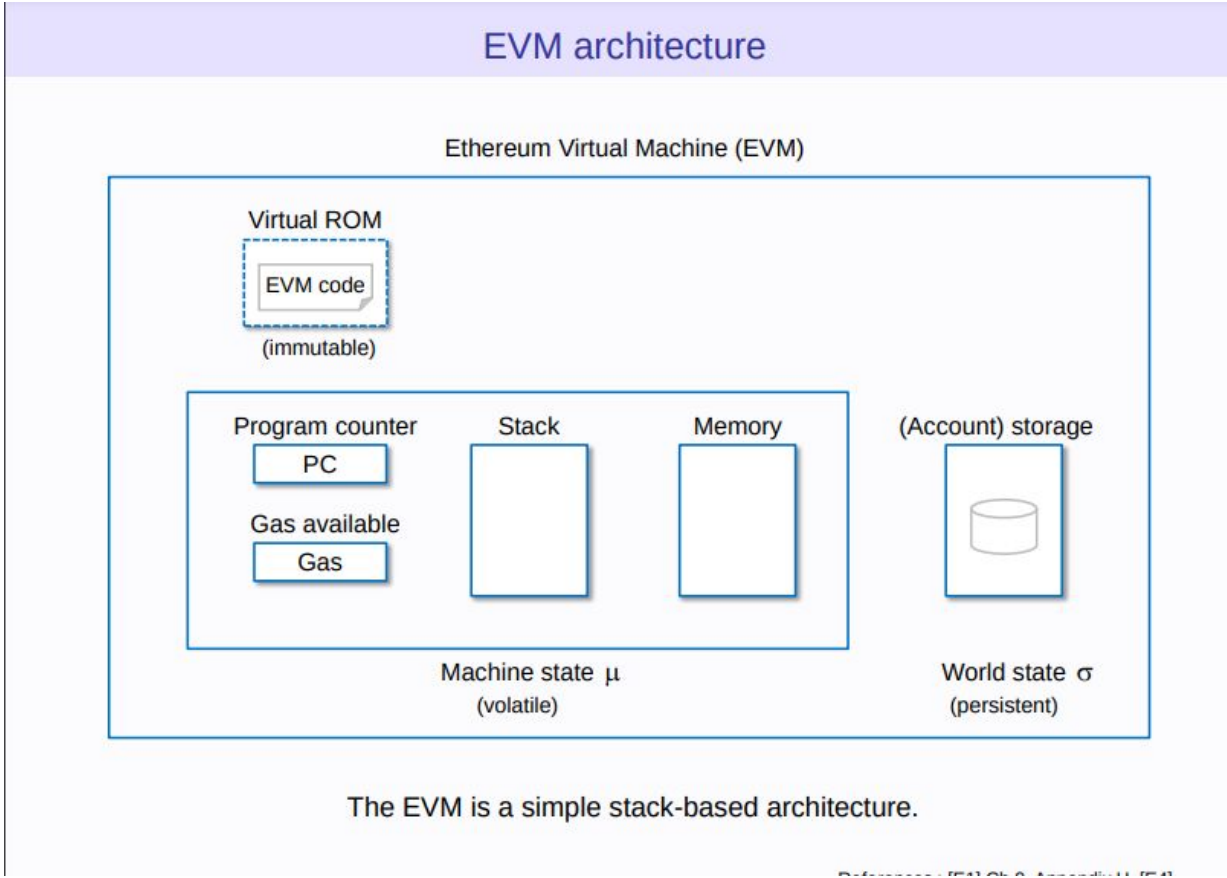
Components of Ethereum Network - Transactions & EVM



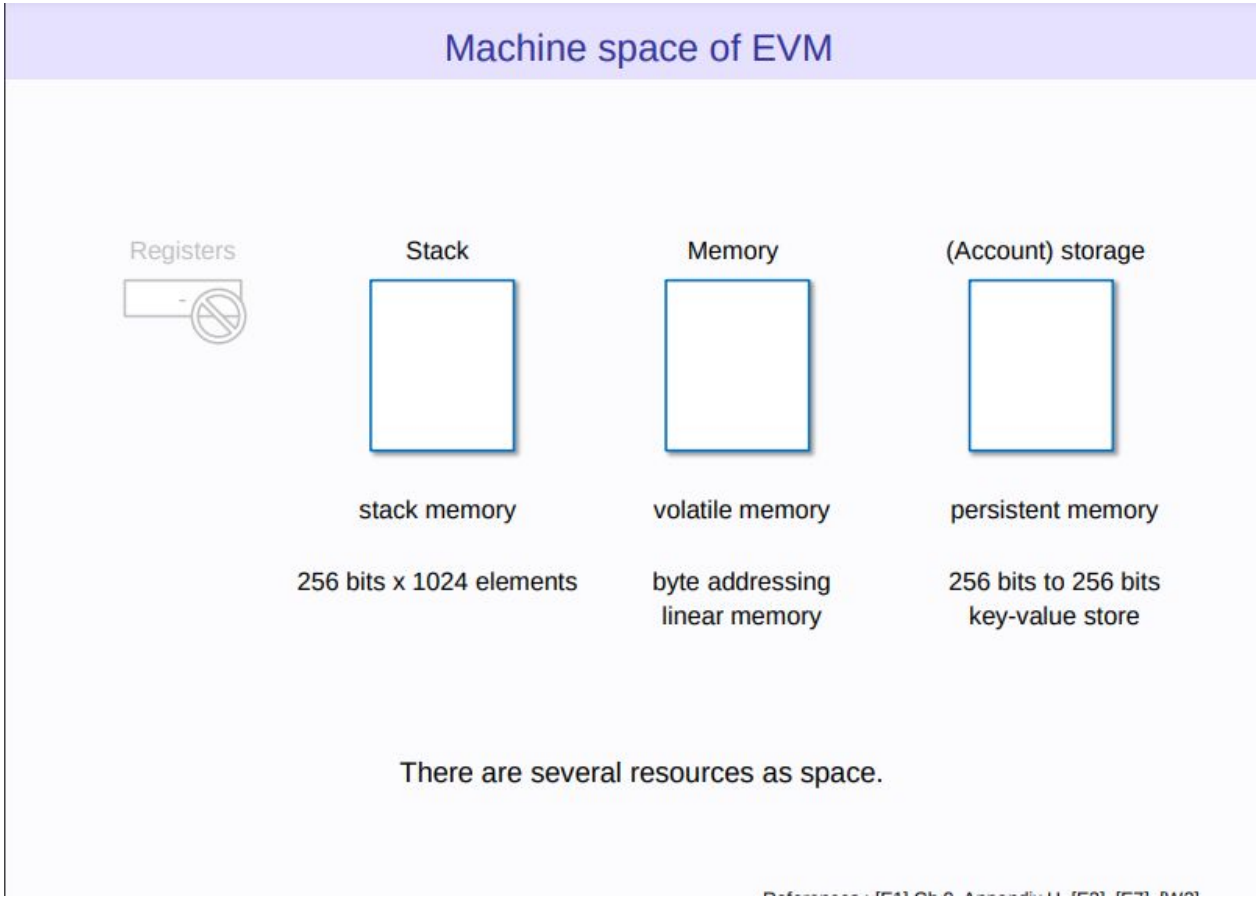
Components of Ethereum Network - EVM



Components of Ethereum Network - EVM Architecture

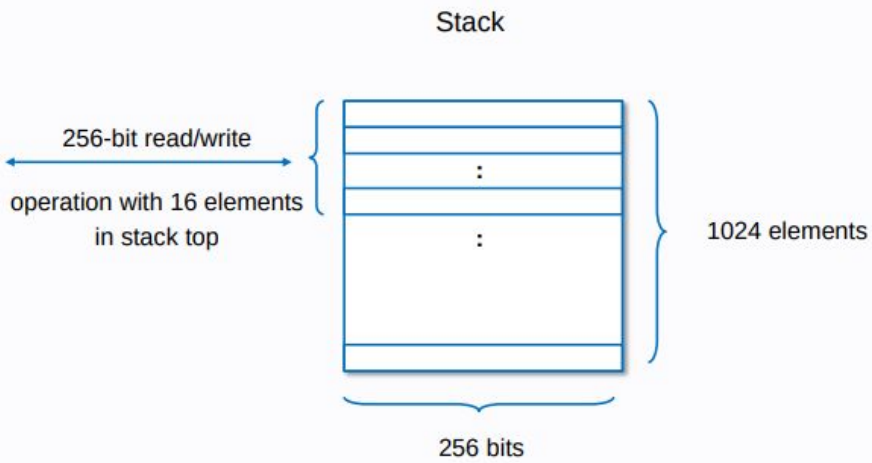


Components of Ethereum Network - EVM Architecture



Components of Ethereum Network - EVM Architecture

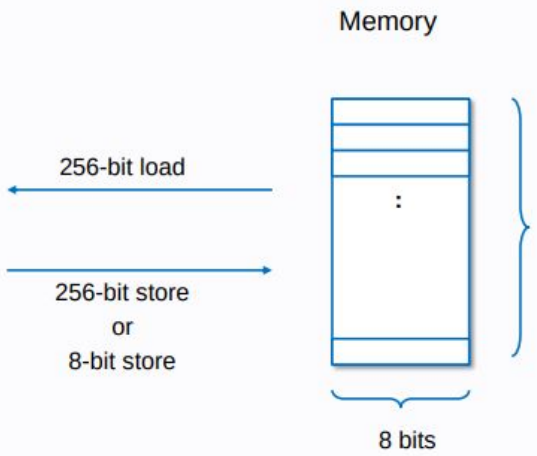
Stack



All operation are performed on the stack.
Access with many instructions such as PUSH/POP/COPY/SWAP, ...

Components of Ethereum Network - EVM Architecture

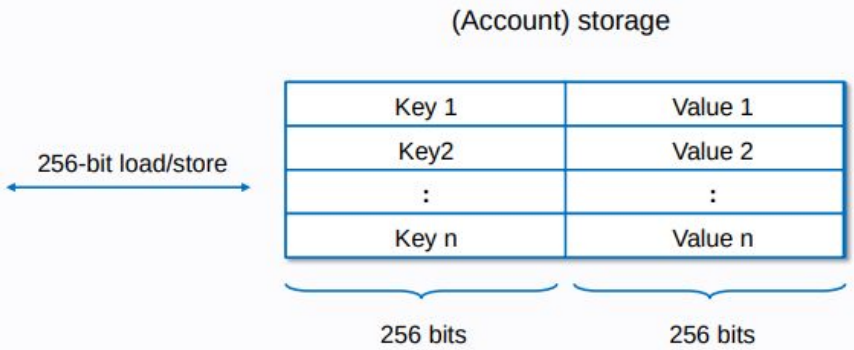
Memory



Memory is linear and can be addressed at byte level.
Access with MSTORE/MSTORE8/MLOAD instructions.
All locations in memory are well-defined initially as zero.

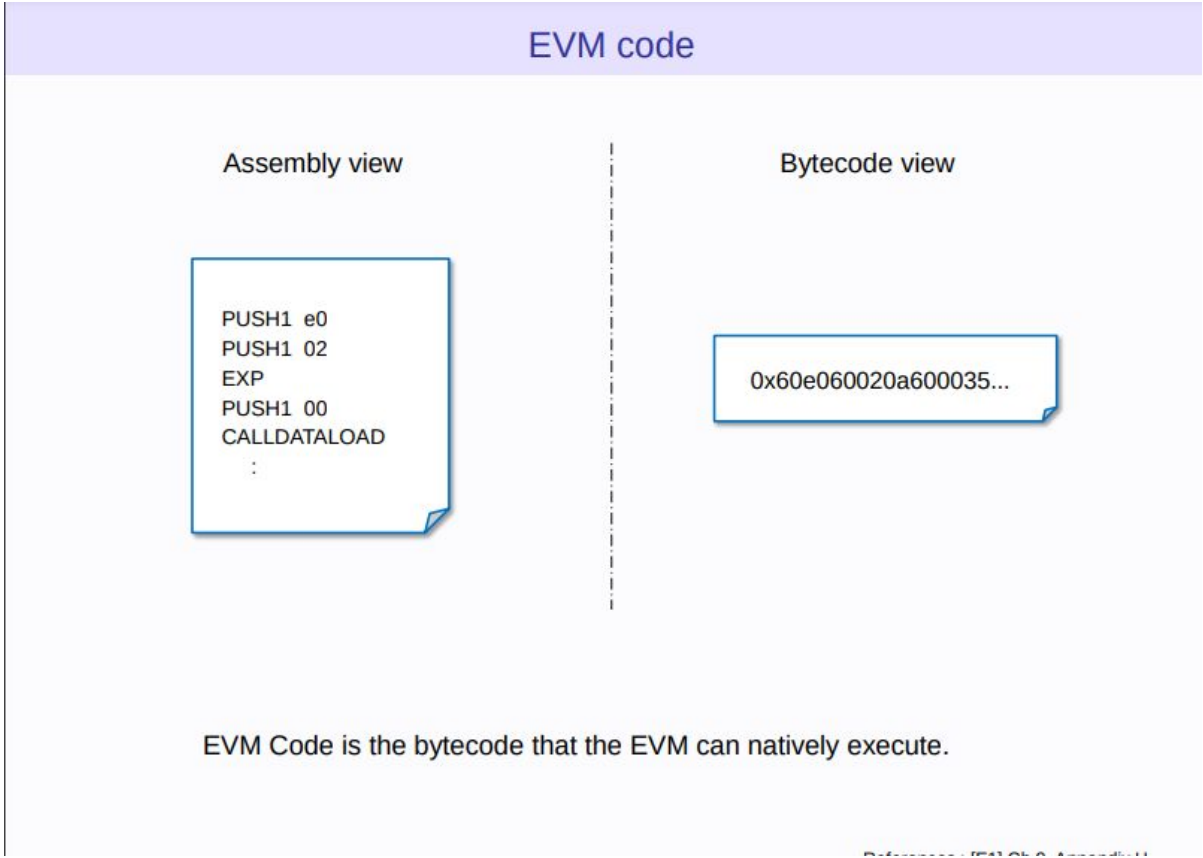
Components of Ethereum Network - EVM Architecture

Account storage

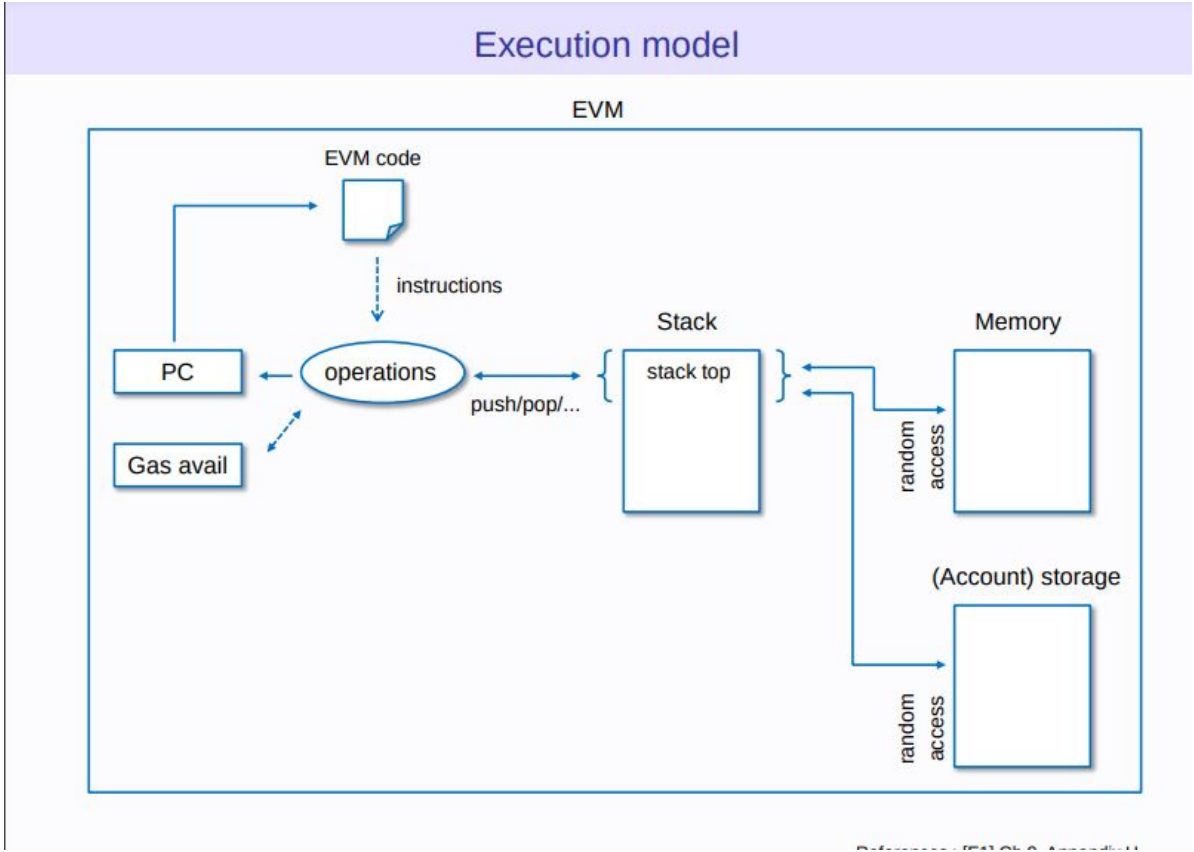


Storage is a key-value store that maps 256-bit words to 256-bit words.
Access with SSTORE/SLOAD instructions.
All locations in storage are well-defined initially as zero.

Components of Ethereum Network - EVM Architecture

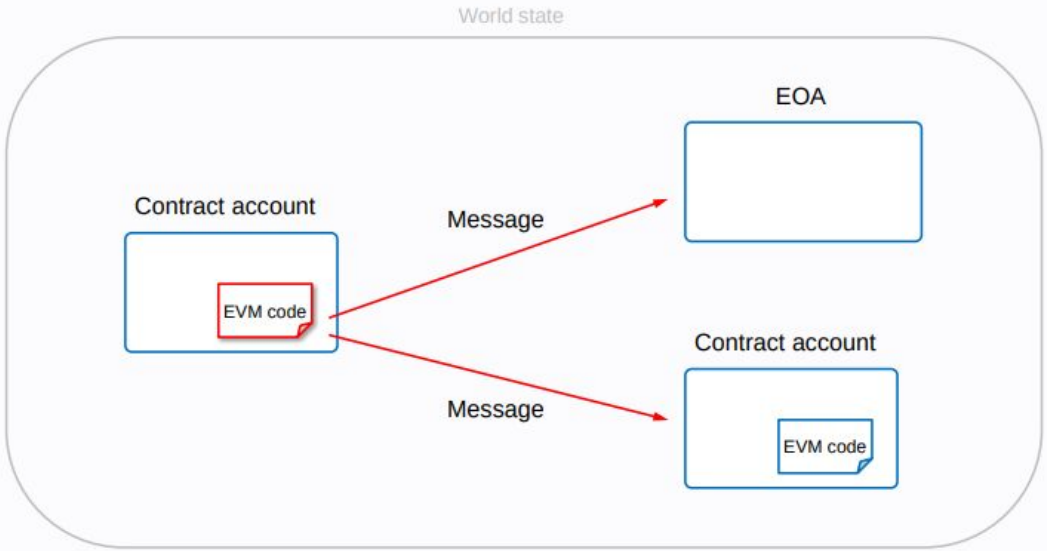


Components of Ethereum Network - EVM Architecture



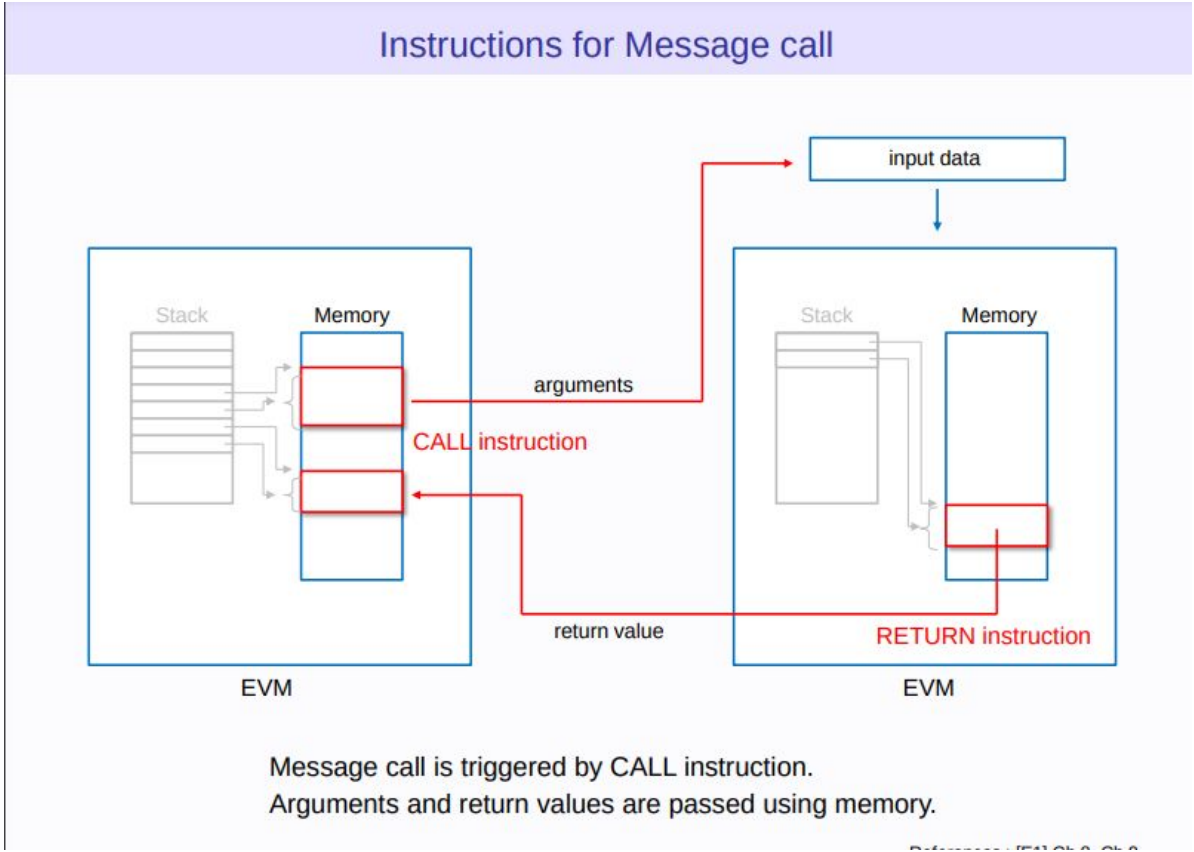
Components of Ethereum Network - Transactions

Message call

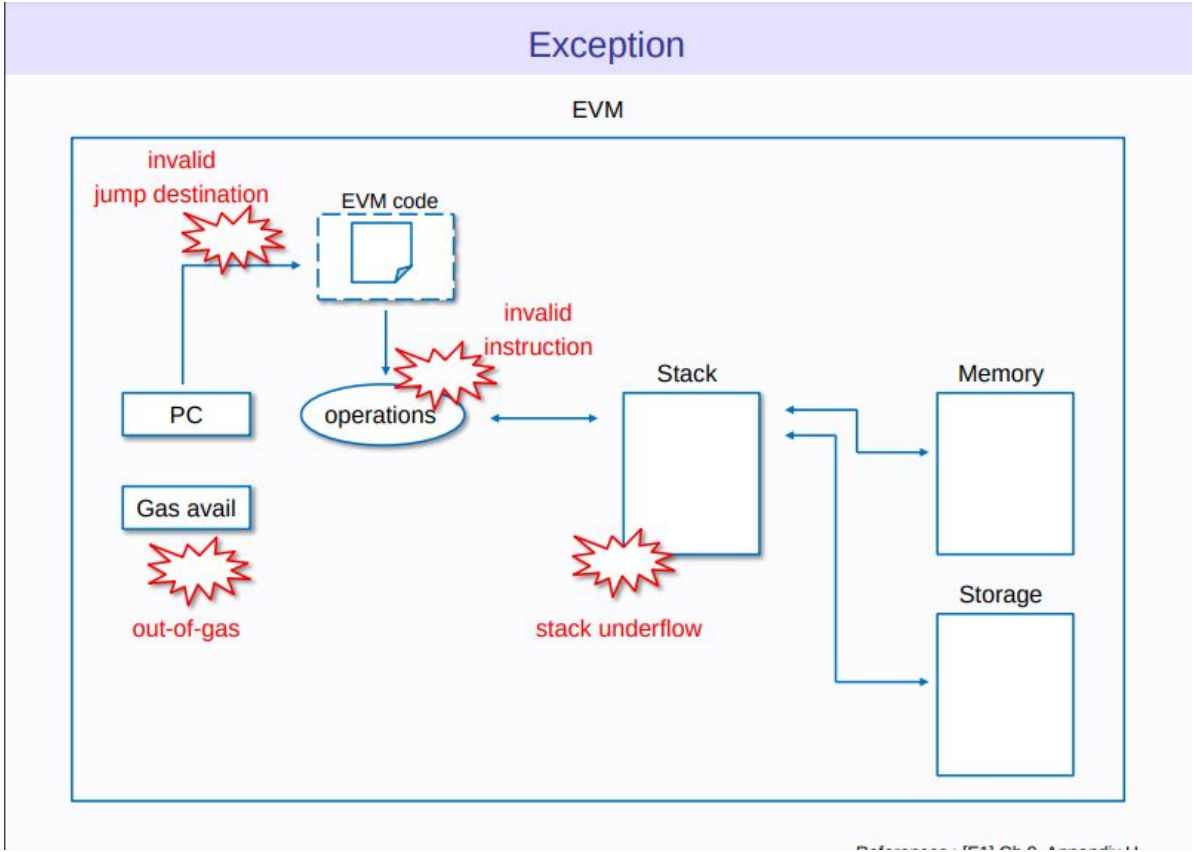


EVM can send a message to other account.
The depth of message call is limited to less than 1024 levels.

Components of Ethereum Network - Transactions

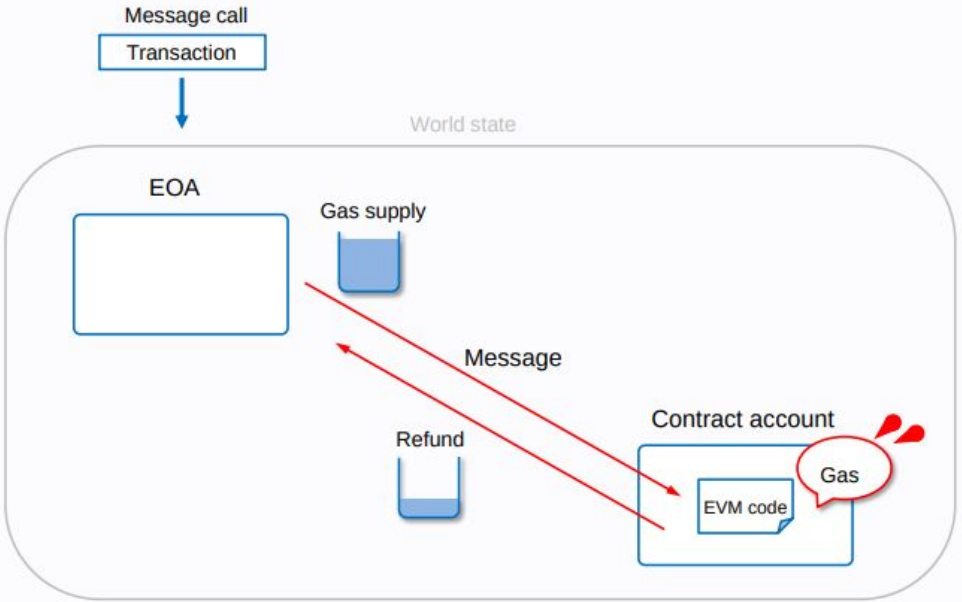


Components of Ethereum Network - Transactions



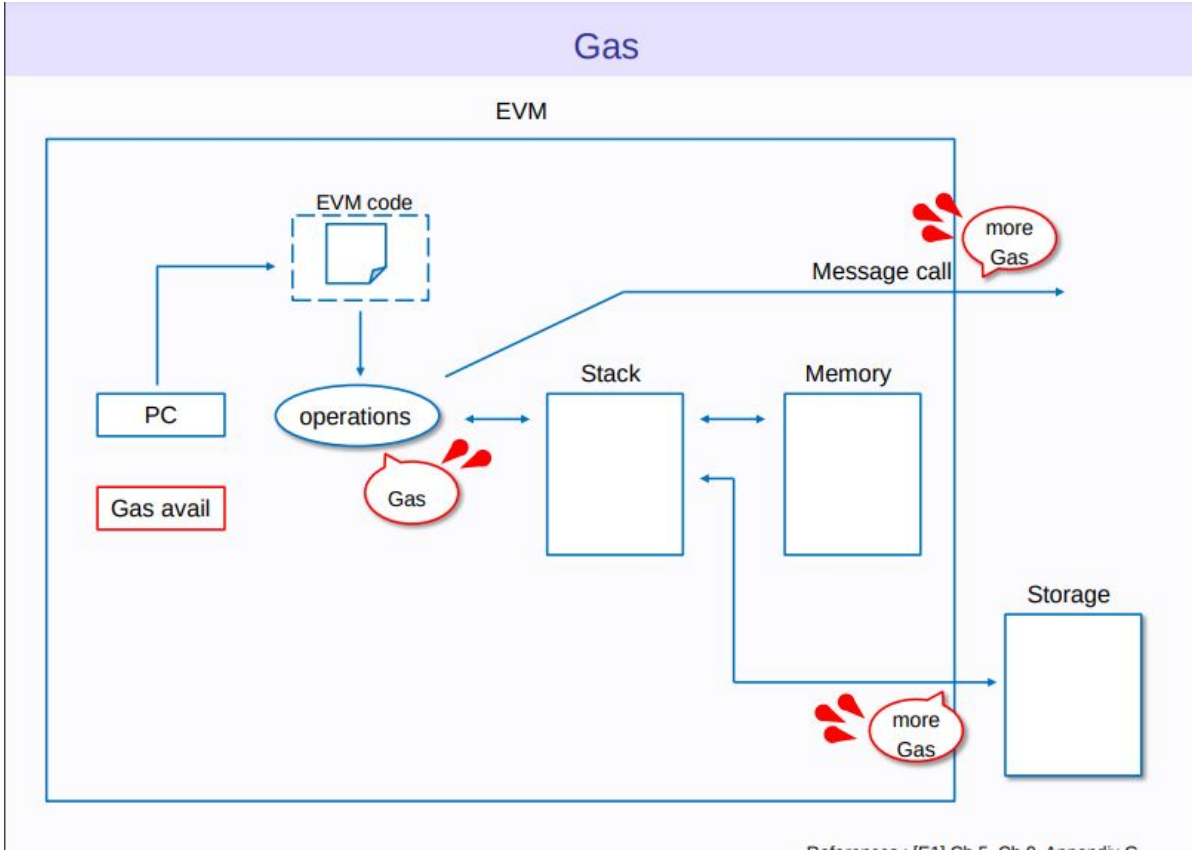
Components of Ethereum Network - Transactions

Gas and fee



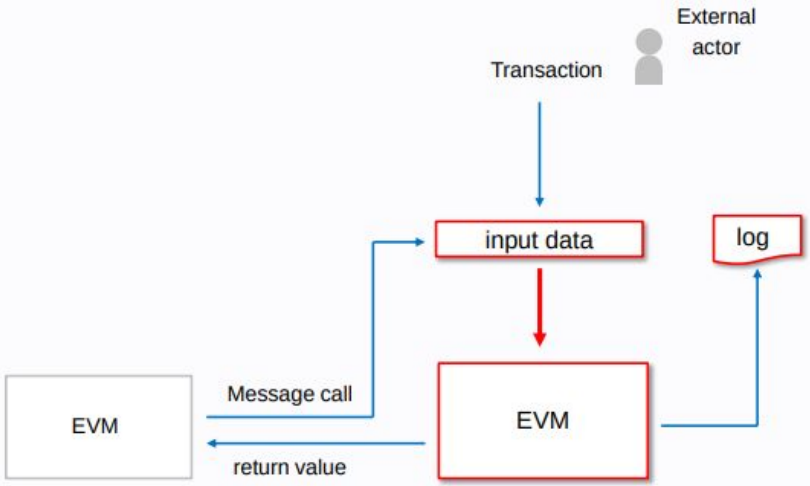
All programmable computation in Ethereum is subject to fees (denominated in gas).

Components of Ethereum Network - Transactions



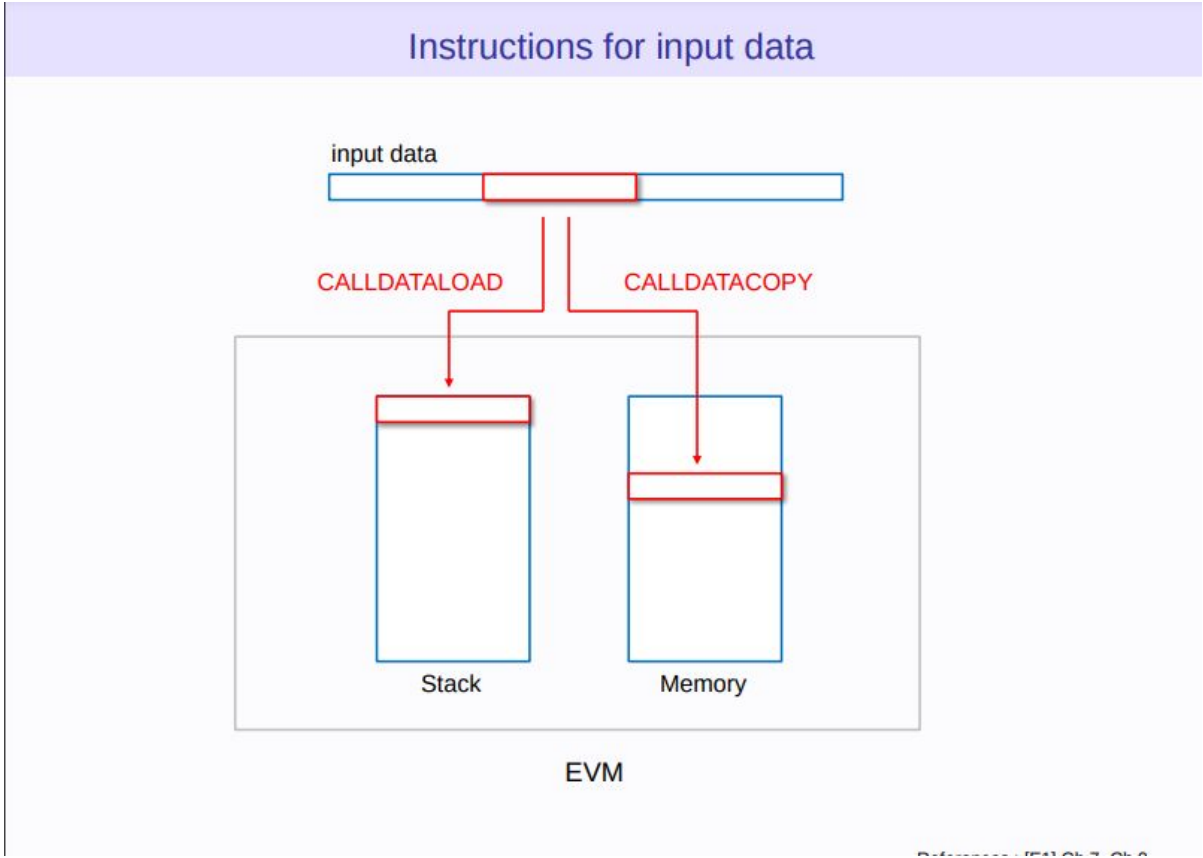
Components of Ethereum Network - Transactions

Input and Output of EVM

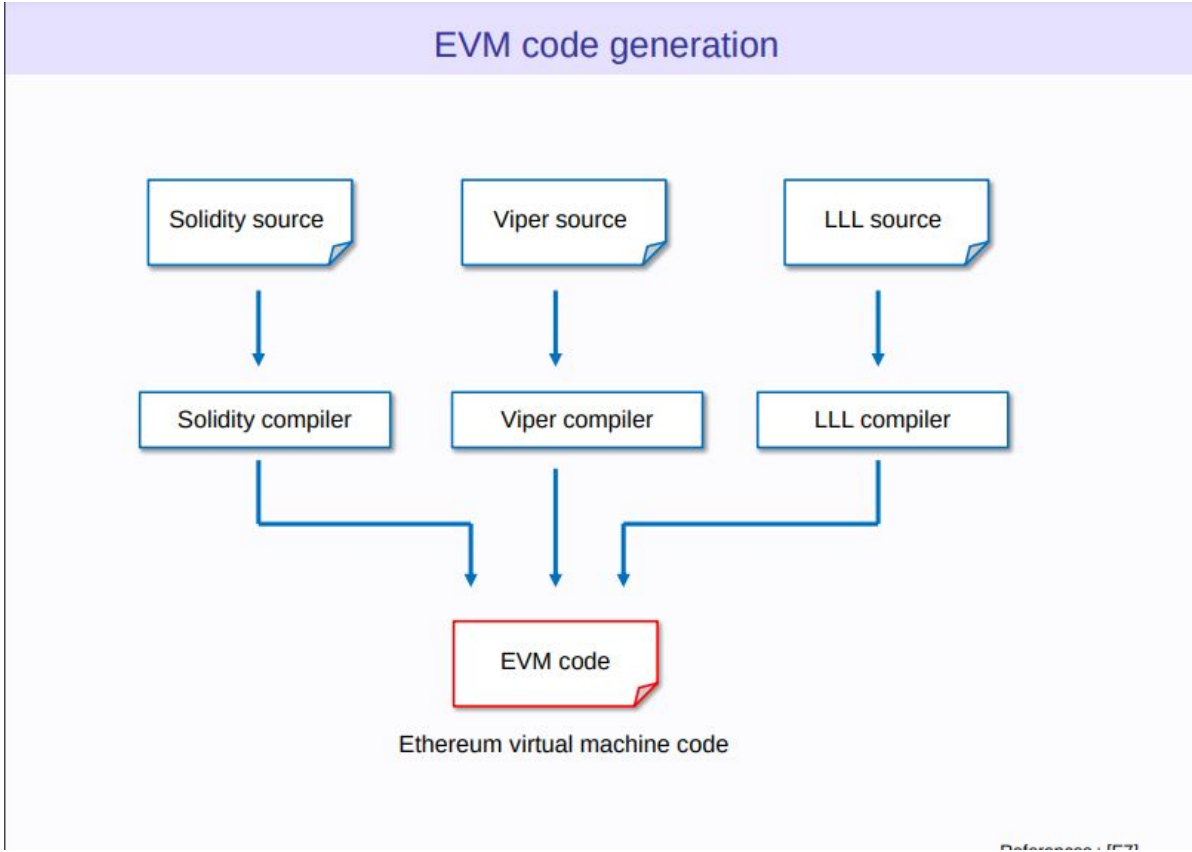


EVM can input external data from a message call.
EVM can output log. EVM can also return values to Caller EVM.

Components of Ethereum Network - Transactions

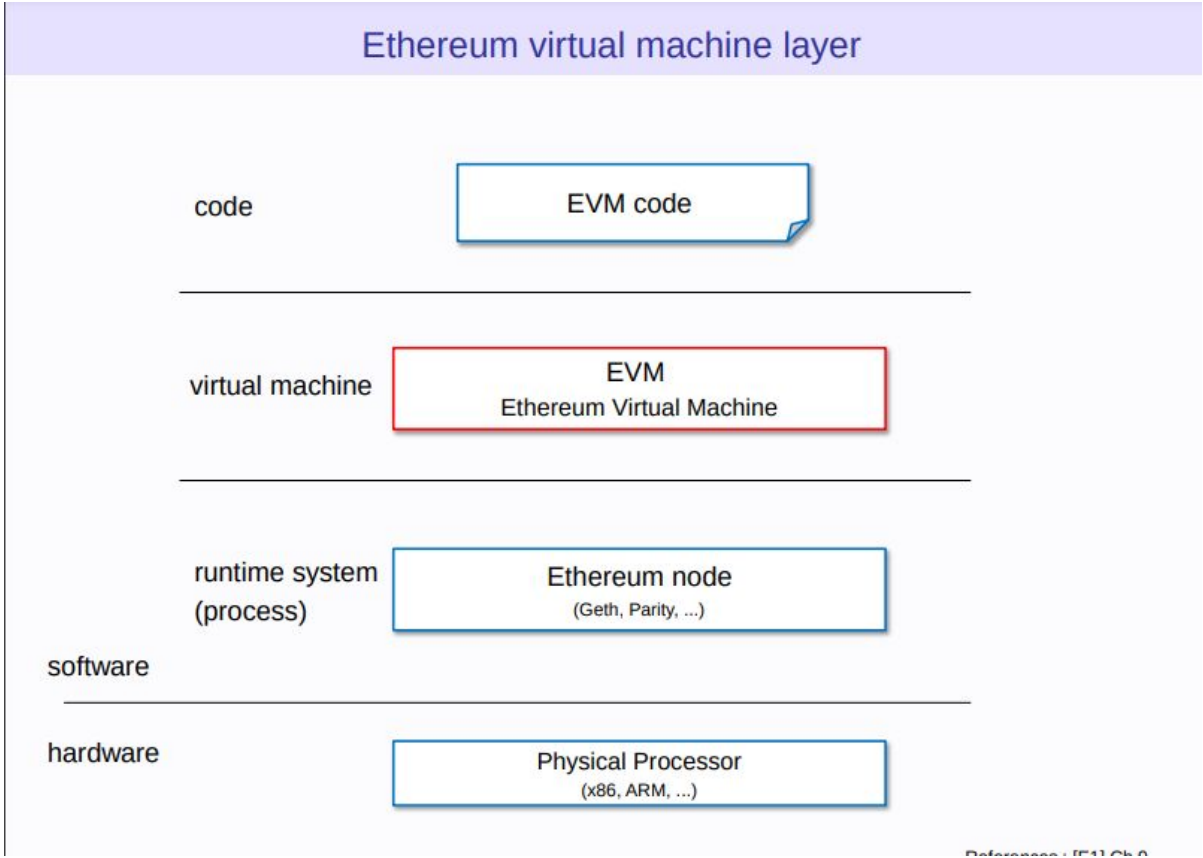


Components of Ethereum Network - EVM



Courtesy : Takenobu T

Components of Ethereum Network - EVM



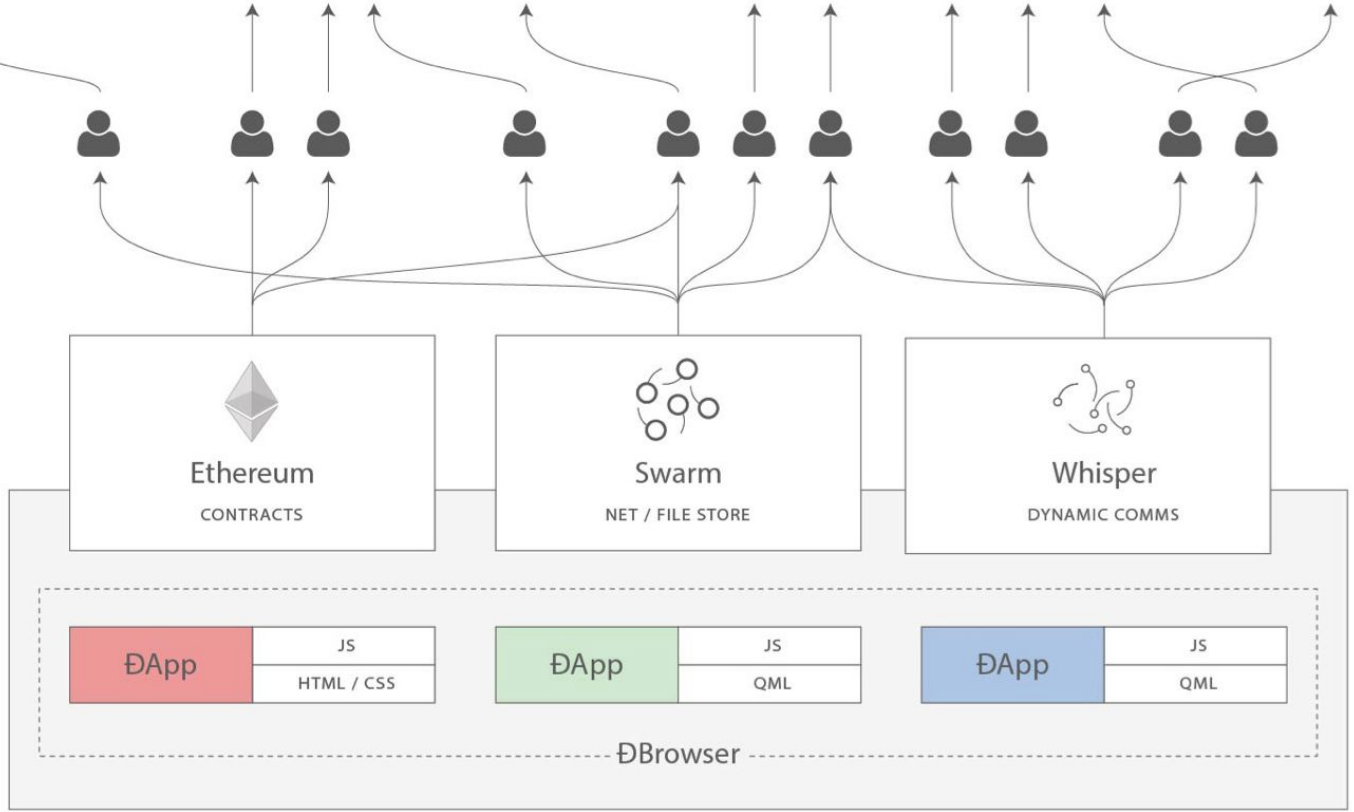
Courtesy : Takenobu T

Components of Ethereum Network - Swarm & Whisper

- Swarm and Whisper are complementary technologies contributing to the vision of Ethereum as a "world computer".
- When imagining Ethereum as a metaphor for a shared computer, it should be noted that computation alone is not enough.
- For a computer to be fully useful, it also needs storage to "remember" things and bandwidth to "communicate" them. This could be summarised as such:
  - **Contracts**: decentralized logic
  - **Swarm**: decentralized storage
  - **Whisper**: decentralized messaging

Courtesy : Ethereum Stackexchange

Components of Ethereum Network - Swarm & Whisper



Interaction including Ethereum contracts, Swarm storage, Whisper comms

Courtesy : Ethereum Stackexchange

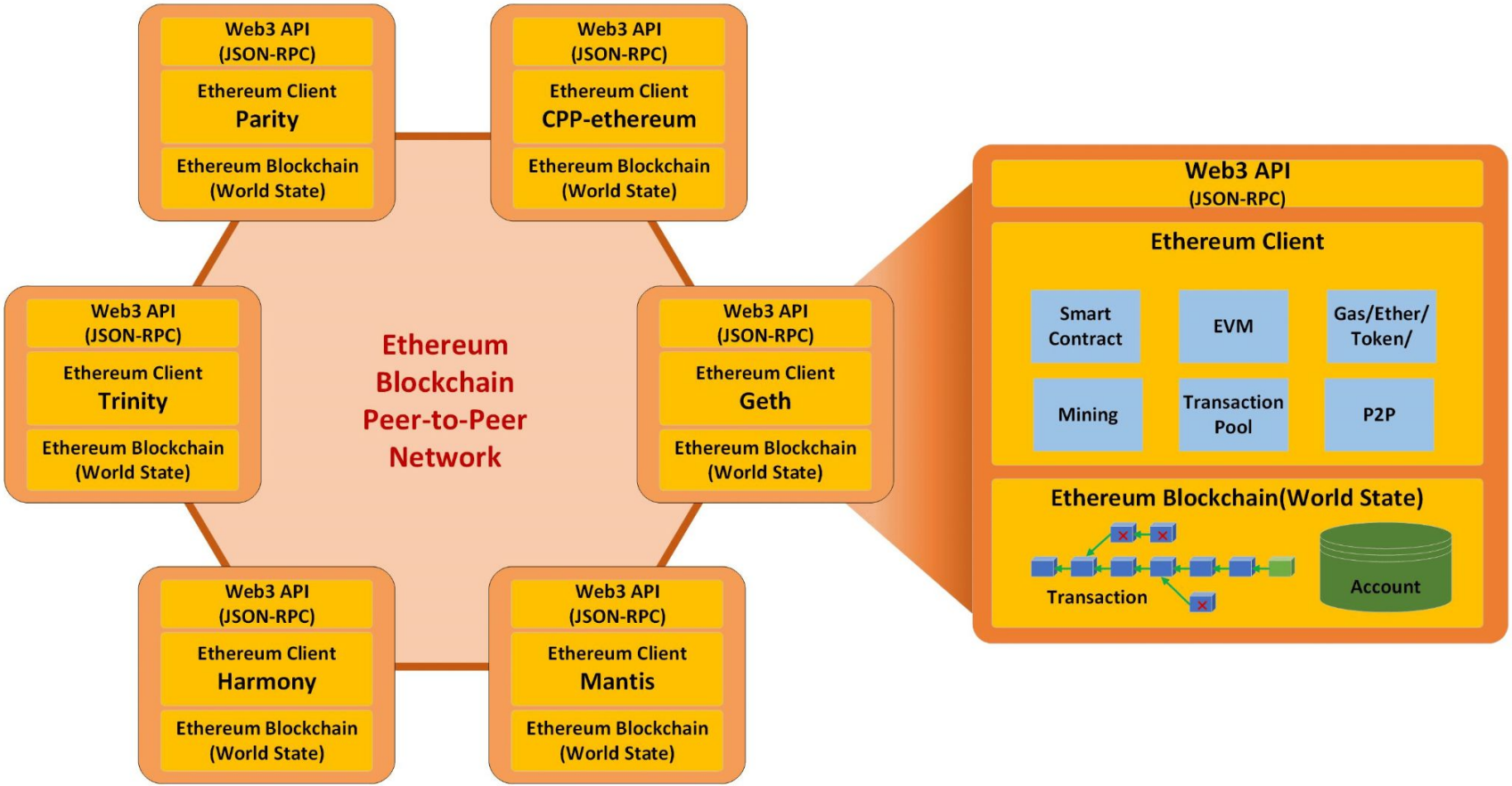
How Does Ethereum Work?

Ethereum **implements an execution environment** called **Ethereum Virtual Machine (EVM).**

1. When a **transaction triggers a smart contract** all the nodes of the network will execute every instruction.
2. **All the nodes** will **run EVM** for **block verification**, where the nodes will go through the transactions listed in the block and runs the code as triggered by the transaction in the EVM.
3. **All the nodes** on the network must **perform the same calculations for keeping their ledgers in sync.**
4. Every transaction must include:
   a. **Gas limit.**
   b. **Transaction Fee** that the sender is willing to pay for the transaction.
5. If **total amount of gas needed to process the transaction < = the gas limit** then the *transaction will be processed*
6. if the **total amount of the gas > the gas limit** then the *transaction will not be processed* *and* the fees are still lost.

Thus it is **safe to send transactions with the gas limit above the estimate to increase the chances of getting it processed**

Ethereum Architecture



Courtesy : [Ethereum Architecture]

Ethereum Architecture

Ethereum blockchain network :

- decentralized Peer-to-Peer (P2P) network of Ethereum clients, representing network nodes.

Ethereum client

- any node that verify the new transaction, execute the smart contract, and process new blocks of the chain.
- EVM residing in thousands of computers or devices on the internet, and connected through the Ethereum P2P network.
- Ethereum Virtual Machine or EVM and the runtime environment in the P2P network for smart contract execution.
- Ethereum clients run the EVM and can technically be written in any popular programming language.
- There are many different implementations of Ethereum clients.
- Ethereum client implementations are defined in the Ethereum Yellow Paper.
- provide **a set of web3 APIs over JSON-RPC for DApps** interacting with an Ethereum blockchain.

Ethereum Architecture

Advantages with a variety of Ethereum client implementations

- makes the network more resilient against bugs.
- prevents the centralization of developer resources.
- competitions between teams help to find the best solutions to common and challenging issues.
- Each client may have a different focus, strength, and weakness in mining, prototyping, DApp development, and more.

From web or wallet application,

- one can use the web3 object provided by the web3.js library to communicate with the Ethereum network.
- It works with any Ethereum client.
- it connects to a local or remote Ethereum node and makes RPC calls.

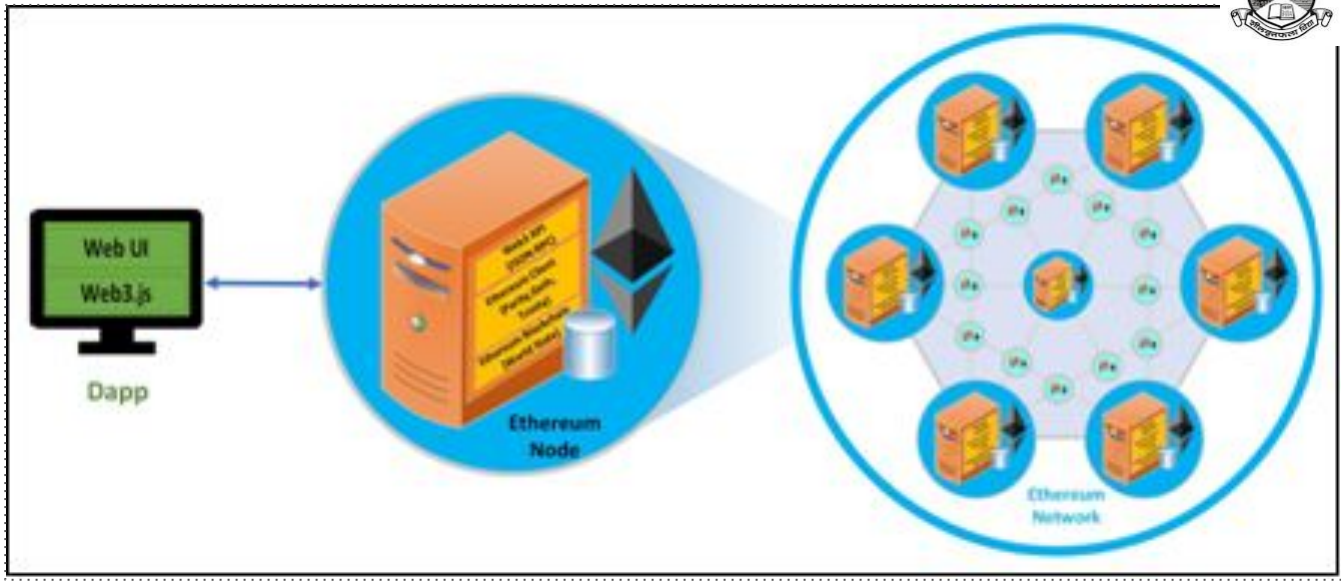
Ethereum Architecture

DApp developers or private Ethereum blockchain operators may choose the ones fitting their own special needs.

Client	Language	Developers	Where to download?
Geth	Go	Ethereum Foundation	https://geth.ethereum.org/downloads/
Parity	Rust	Parity	https:/www.parity.io
cpp-ethereum	C++	Ethereum Foundation	https://github.com/ethereum/aleth
Trinity	Python	Ethereum Foundation	https://github.com/ethereum/trinity
Harmony	Java	Ether Camp	https://github.com/ether-camp/ethereum-harmony
Mantis	Scala	IOHK's Team Grothendieck	https://mantis.readthedocs.io/en/latest/

Courtesy : Ethereum Architecture

Ethereum Architecture



In short, **client-server model,**

- Client : **DApps**
- Server : **entire Ethereum network**

- To DApps, the Ethereum network is just like a giant world computer, assembled together with thousands of computing devices throughout the internet.
- Once connected to the network, one could connect to any node in the decentralized network,

Courtesy : Ethereum Architecture

Ethereum Architecture

Ethereum client provides all blockchain components to maintain

- Managing transaction and state transition with the Ethereum blockchain
- Maintaining world state and account state
- Managing P2P communication Block finalization with mining
- Managing transaction pool
- Managing cryptoassets, gas, ether, and tokens

Ethereum design based on the white paper intends

- to build a simple, efficient and extensible blockchain platform,
- have a Turing-complete program language to support more sophisticated and complex computations.
- Incorporate all of the benefits of a blockchain
- serve as the framework for supporting all types of digital assets and value transfers as well.

Courtesy : Ethereum Architecture

Ethereum Architecture

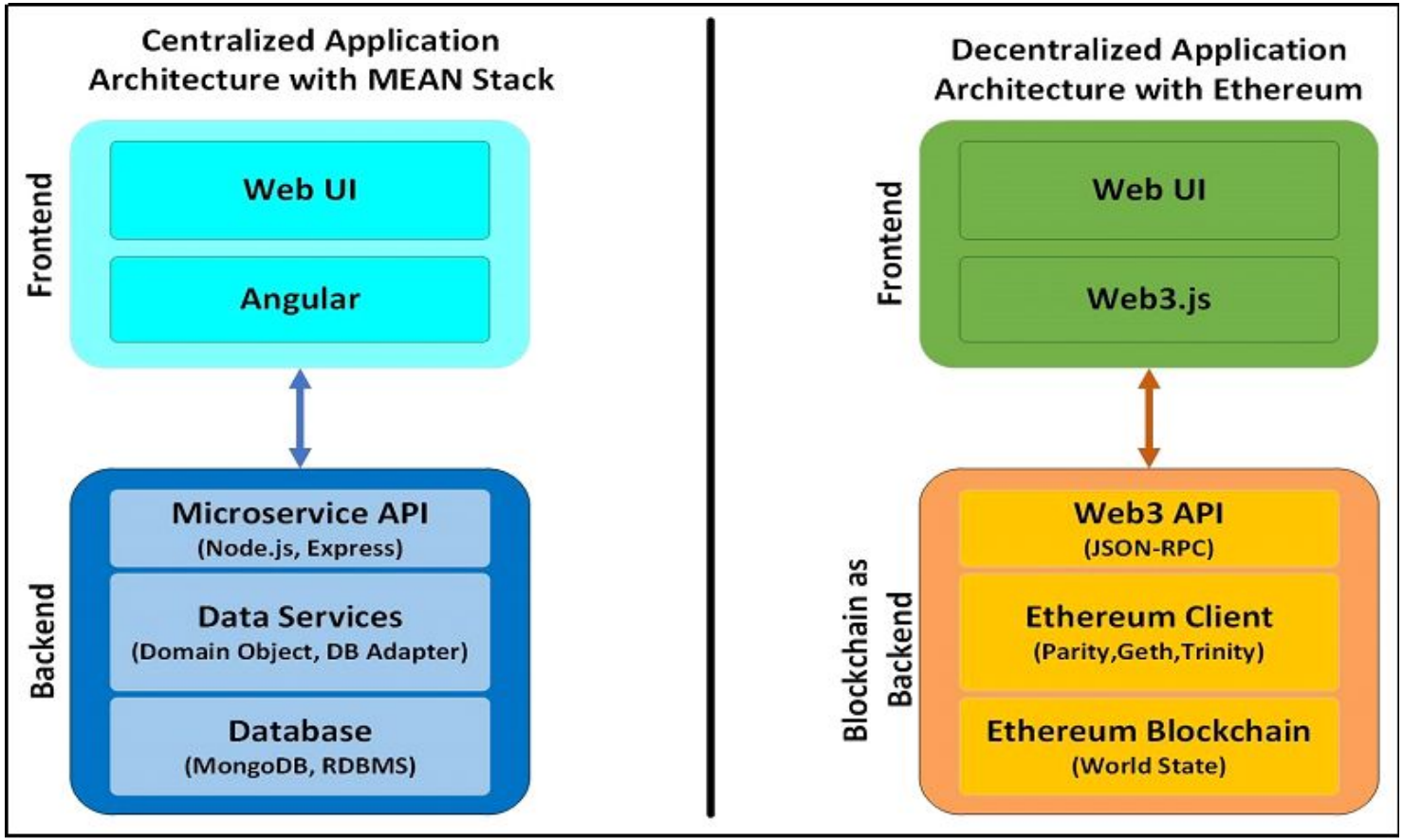
DApp (Decentralized Applications)

- application or service that runs on a blockchain network
- enables direct interaction between consumers and providers,
- Eg: Connecting buyers and sellers in a decentralized marketplace.
- use and generate cryptographic tokens.
- often start with a whitepaper and a working prototype.

DApp involves

- Decentralized backend that runs on the blockchain network
- Centralized frontend that allows end users to access their wallets and make a transaction

Ethereum Architecture - Centralized Vs Decentralized Applications

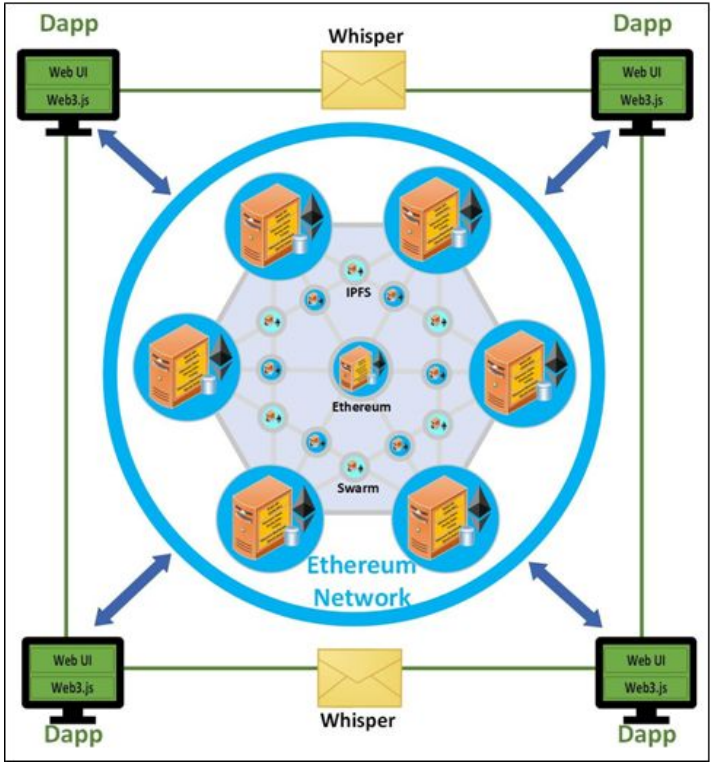


Courtesy : Ethereum Architecture

Ethereum Architecture

Ethereum provides

- a platform for anyone to write smart contracts and decentralized applications based on your business needs and value propositions.
- It is intended as the world computer for the decentralized world.
- Ethereum provides **four decentralized computing facilities**,
  1. Ethereum blockchain for decentralized state
  2. Smart contracts for decentralized computing
  3. Swarm and IPFS for decentralized storage
  4. Whispers for P2P messaging



Courtesy : Ethereum Architecture

Smart Contract - Remix IDE Environment - Key Features

- popular web-based integrated development environment (IDE) for smart contract development on blockchain platforms, especially Ethereum.

1. **Web-Based Environment:**
   - Accessible through a web browser, making it platform-independent
   - Easy to use without the need for local installations.
2. **Solidity Support:**
   - Solidity is the primary programming language for writing smart contracts on Ethereum
3. **Code Editor:**
   - with syntax highlighting and auto-completion features,
   - making it easier for developers to write and edit Solidity code.
4. **Smart Contract Compilation:**
   - allows users to compile their Solidity smart contracts directly within the IDE.
   - It provides information about compilation status, errors, and warnings.

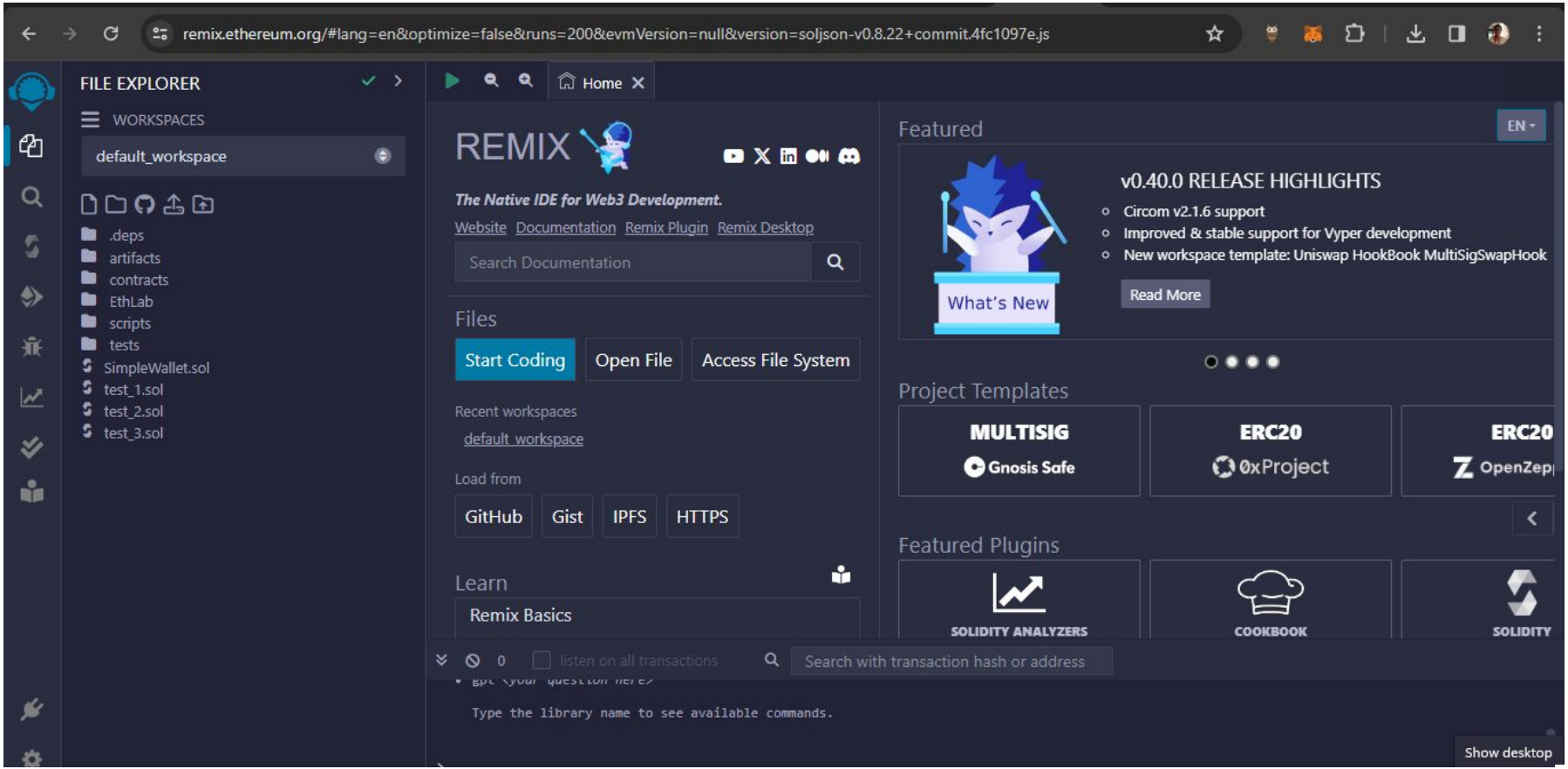
Smart Contract - Remix IDE Environment - Key Features

5. **Contract Deployment:**
   - Developers can deploy their smart contracts to various Ethereum networks (local, testnet, or mainnet) directly from Remix.
6. **Debugging Tools:**
   - allows developers to step through their code, set breakpoints, inspect variables, and troubleshoot issues.
7. **Test Environment:**
   - enables developers to write and run unit tests for their smart contracts.
8. **Plugin System:**
   - allows developers to extend its functionality by integrating additional tools and features.
9. **Version Control Integration:**
   - Developers can integrate Remix with version control systems (such as GitHub) to manage and track changes to their smart contract projects.

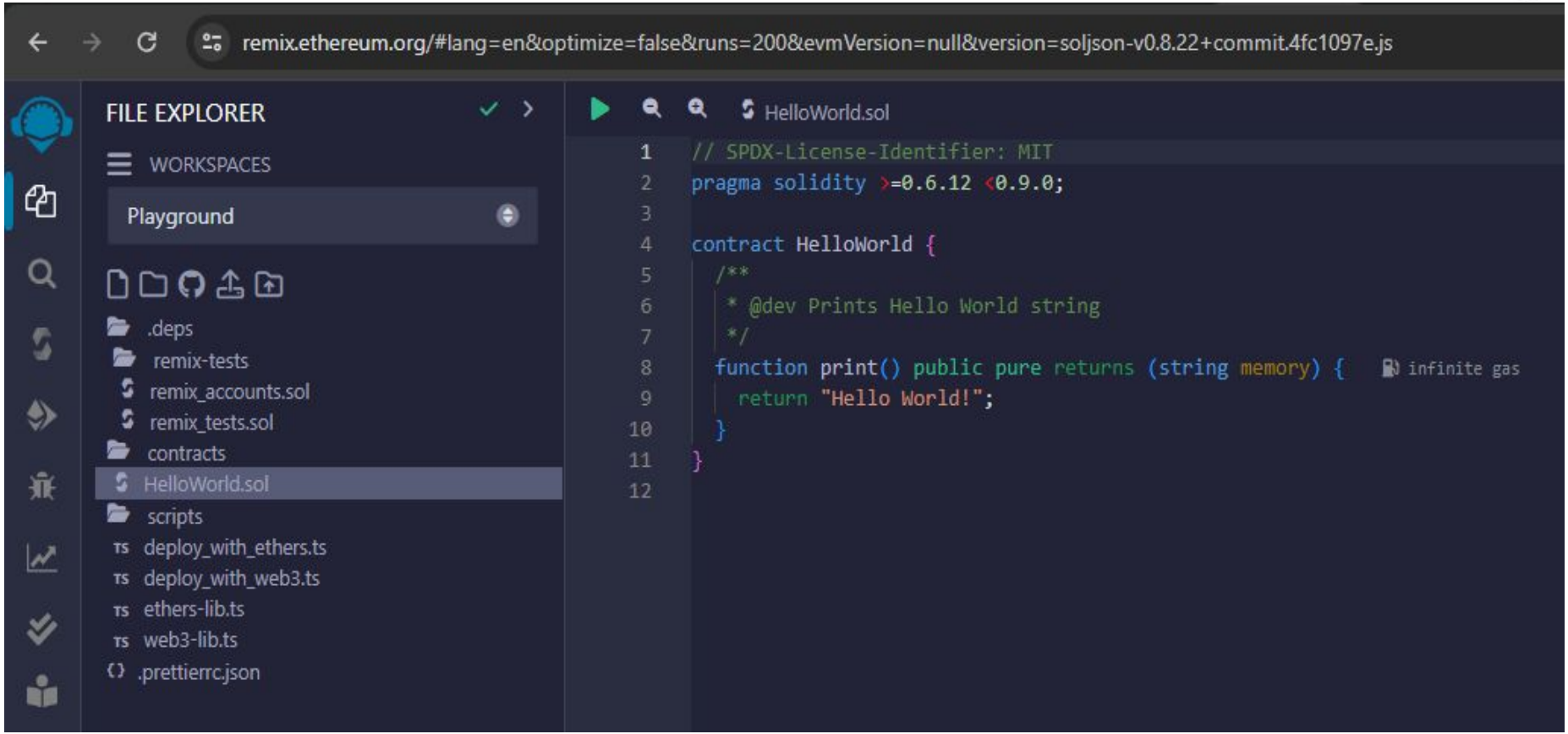
Smart Contract - Remix IDE Environment - Key Features

10. **Ethereum Virtual Machine (EVM) Integration:**
    - allows developers to test and deploy contracts on Ethereum-based networks.
11. **Security Analysis:**
    - helps developers identify potential vulnerabilities in their smart contracts.
12. **Documentation and Community Support:**
    - makes it easier for developers to learn and troubleshoot issues.

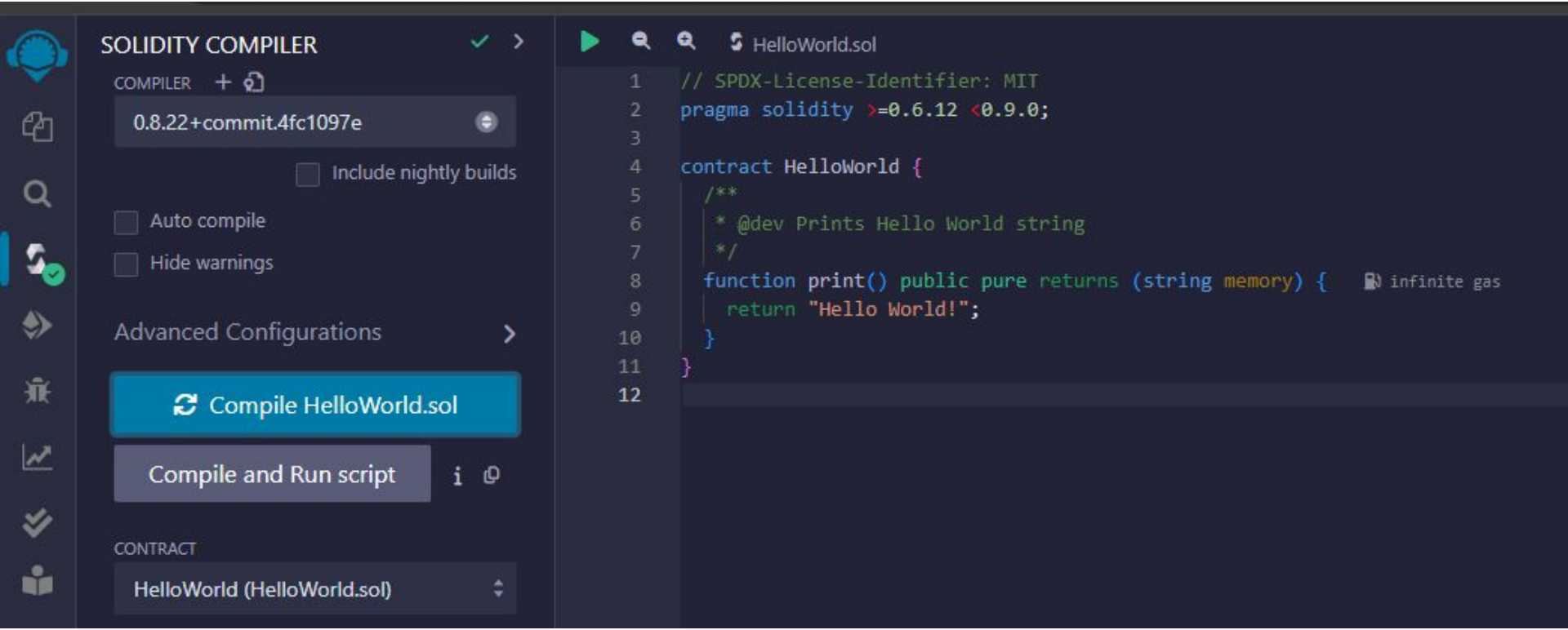
Smart Contract - Remix IDE Environment



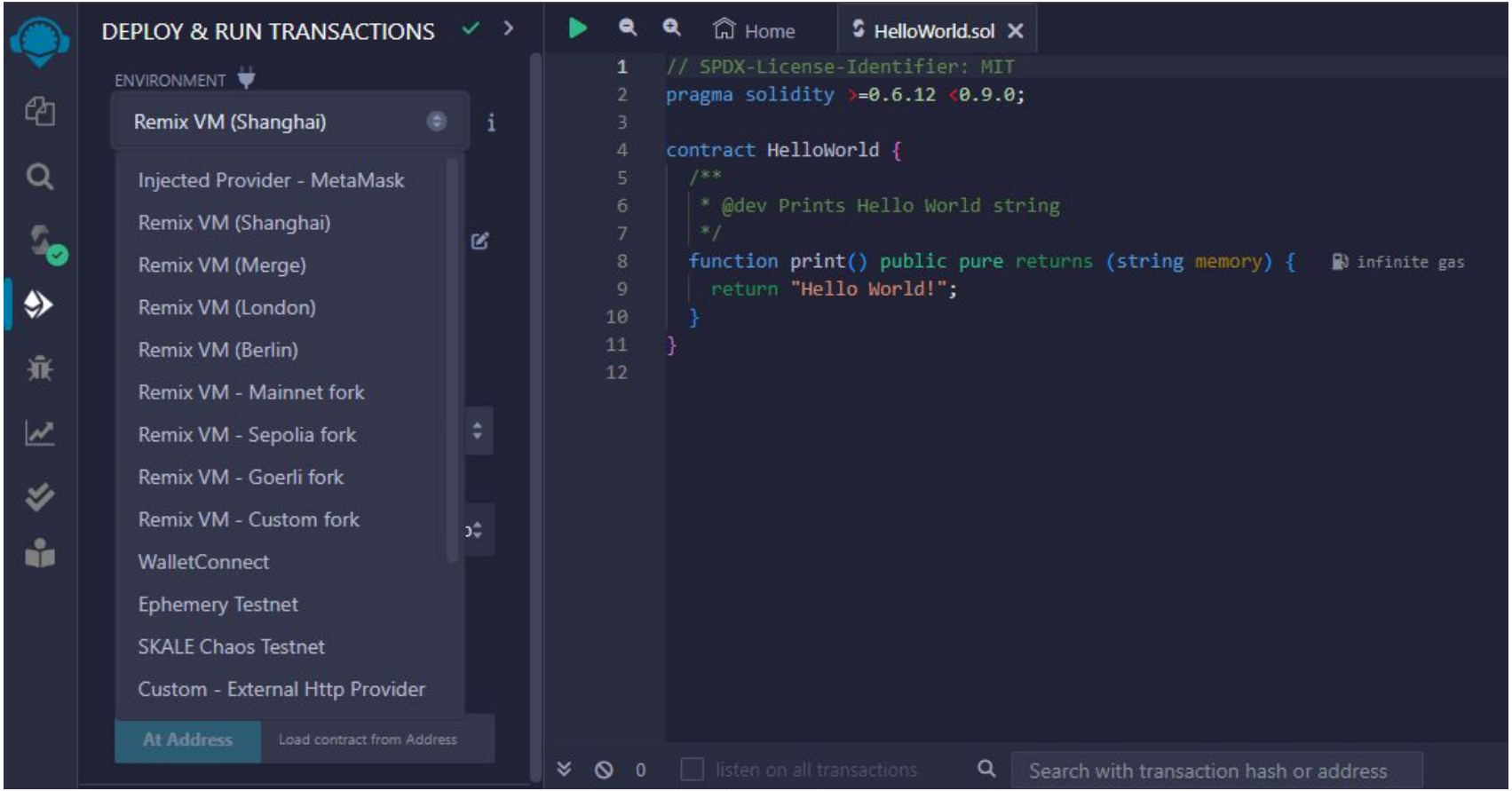
Smart Contract - Remix IDE (HelloWorld.sol)



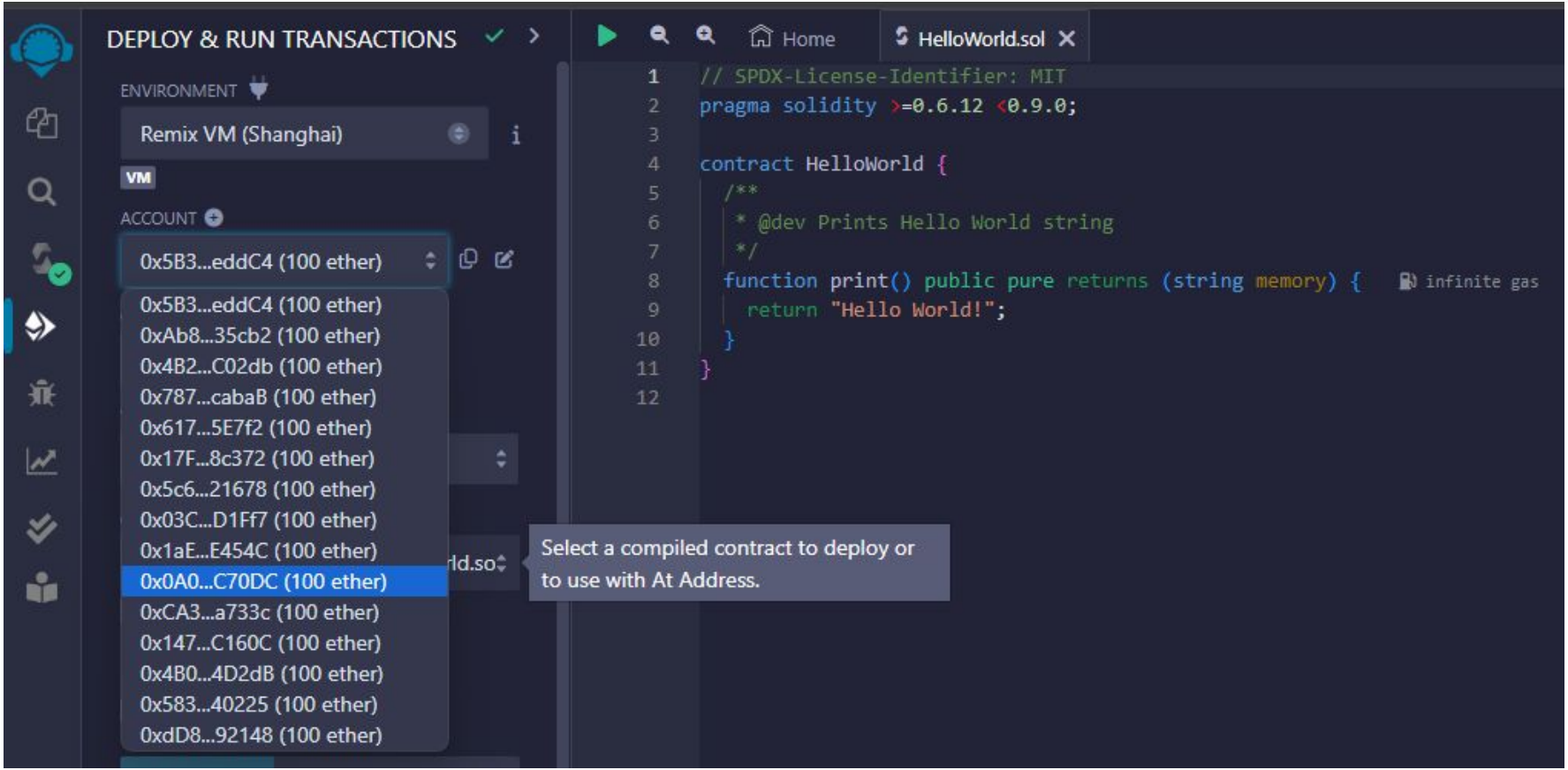
Smart Contract - Remix IDE (Compile HelloWorld.sol)



Smart Contract - Remix IDE (Deploy HelloWorld.sol)

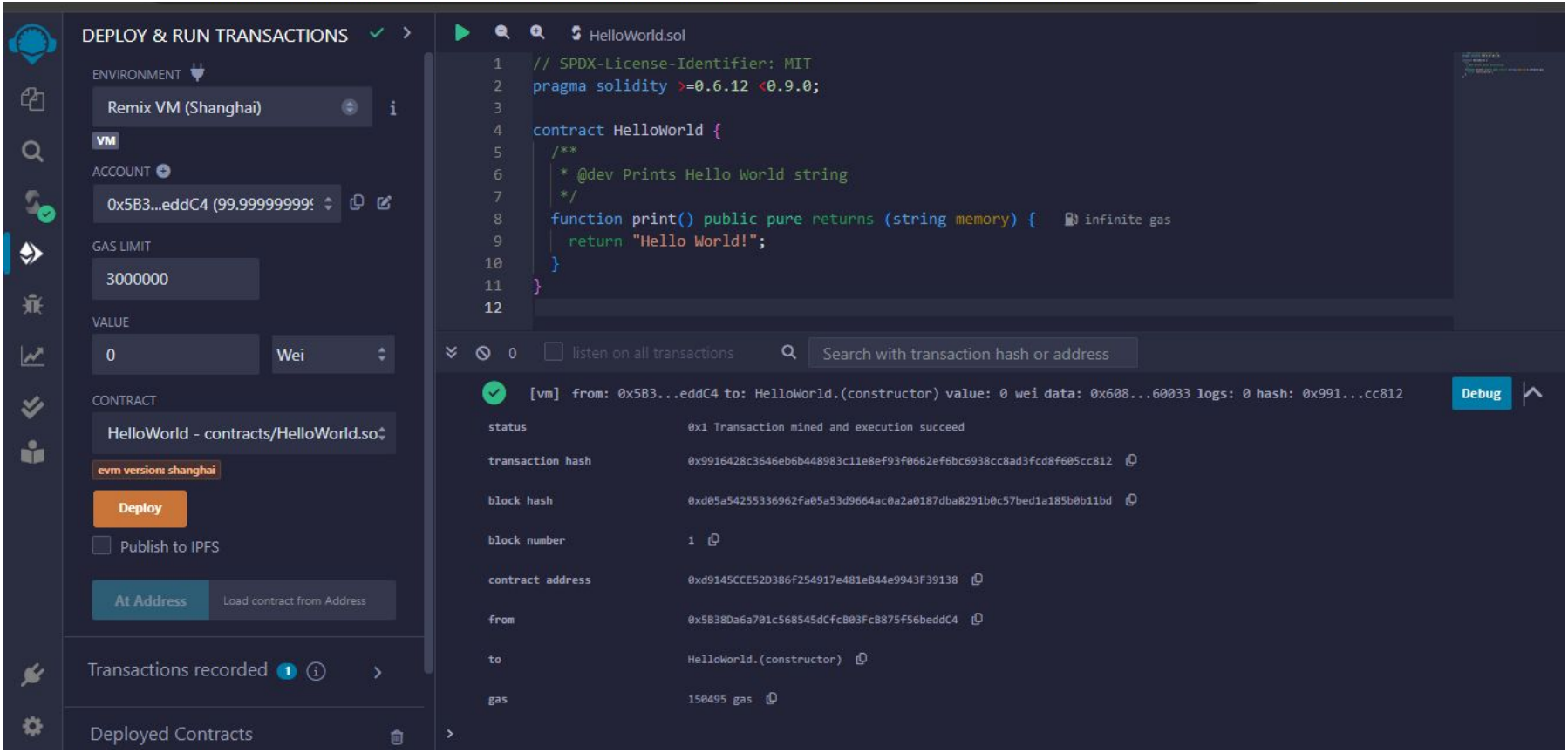


Smart Contract - Remix IDE (Deploy HelloWorld.sol)

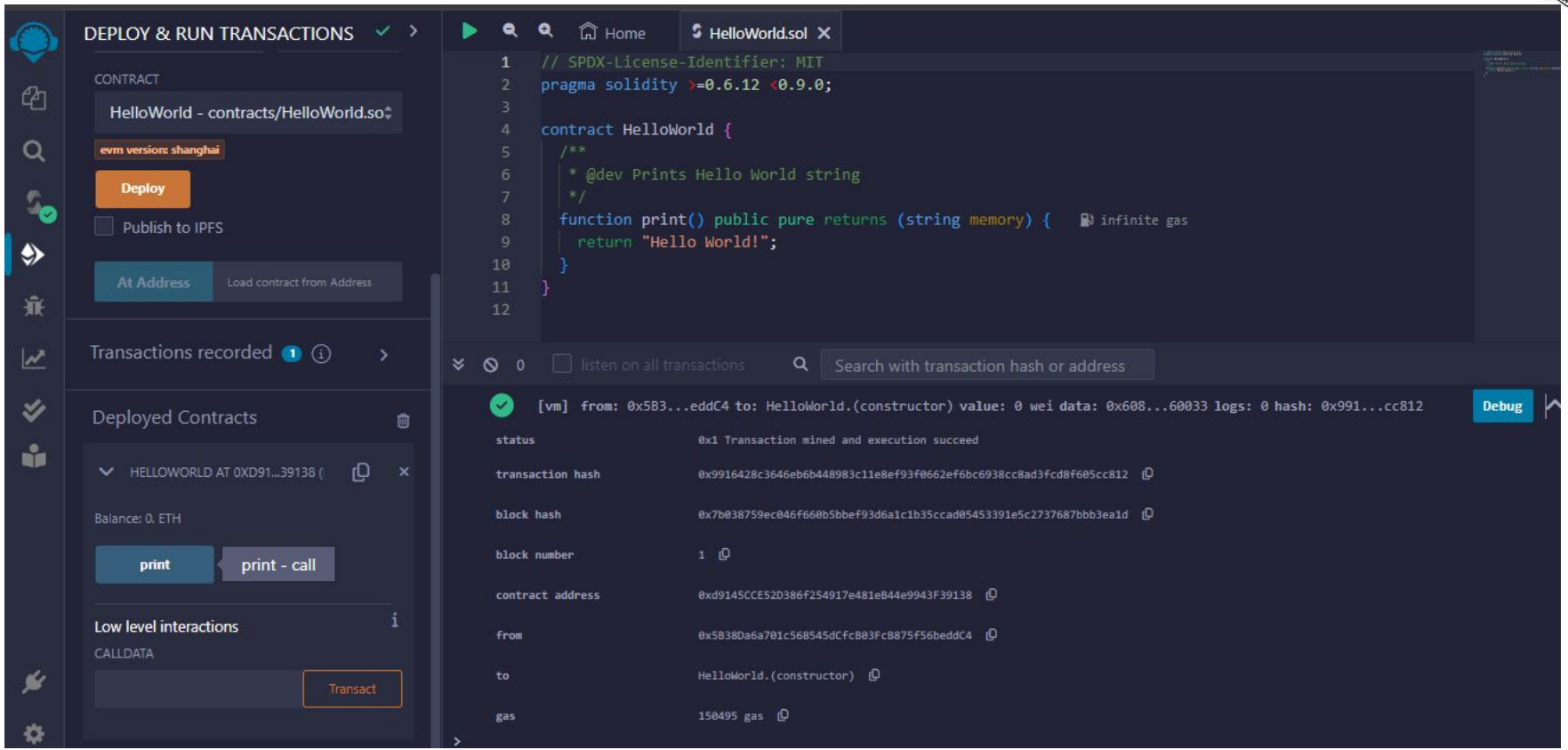


Courtesy : Remix IDE

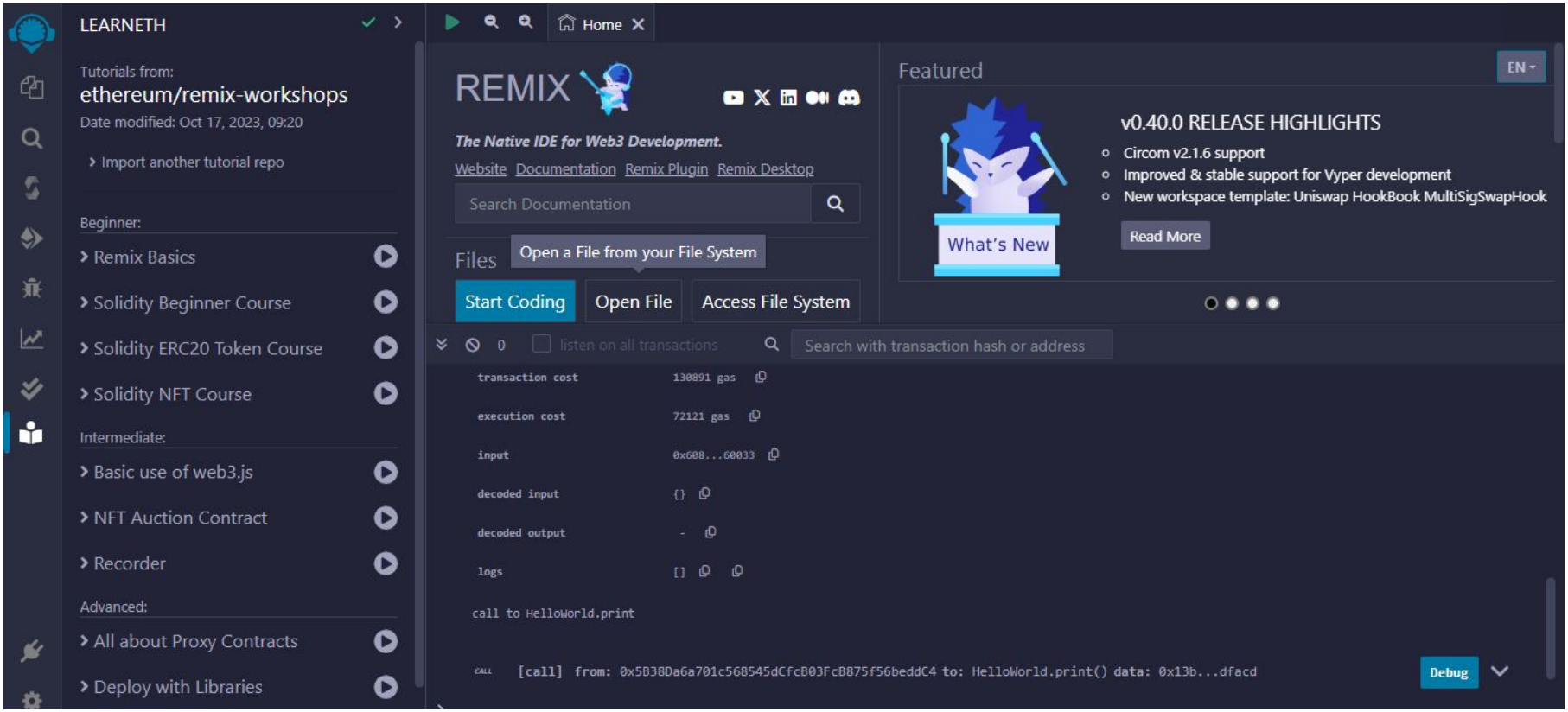
Smart Contract - Remix IDE (Deploy HelloWorld.sol)



Smart Contract - Remix IDE (Deploy HelloWorld.sol)



Smart Contract - Remix IDE (LearnETH)



Introduction to Dapp (Decentralized Application)

- A software application that operates on a decentralized network, typically using blockchain technology.
- Compared to traditional applications that rely on centralized servers, Dapps leverage the security, transparency, and trustlessness of blockchain to function.
- Salient Features of DApps
  1. **Decentralization:**
     - Dapps run on a decentralized network of computers, often referred to as a blockchain.
     - This network is distributed across many nodes, making it resistant to single points of failure and censorship.
  2. **Smart Contracts**:
     - Dapps typically use smart contracts, which are self-executing contracts with the terms of the agreement directly written into code.
     - Smart contracts automatically execute actions when specific conditions are met, without the need for intermediaries.
  3. **Transparency:**
     - All transactions and actions within a Dapp are recorded on a public ledger (the blockchain).
     - This transparency ensures that anyone can verify transactions and data, enhancing trust in the application.

Introduction to Dapp (Decentralized Application)

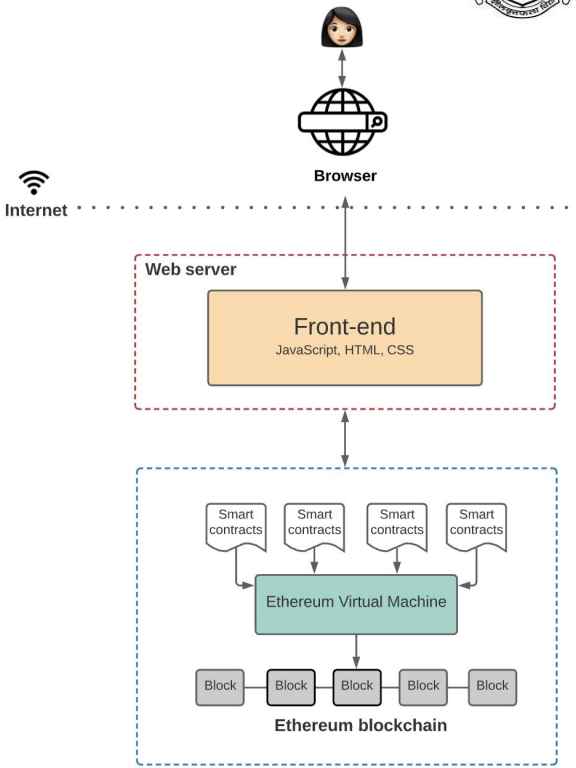
4. **Security:**
   - Blockchain technology, with its cryptographic techniques, provides a high level of security.
   - Transactions are immutable, meaning once recorded on the blockchain, they cannot be altered.
   - This makes Dapps resilient to fraud and hacking.
5. **Trustlessness:**
   - Dapps aim to operate without relying on a central authority or intermediary.
   - Users can interact with the application and each other directly without needing to trust a third party.
6. **Token Economy:**
   - Many Dapps have their own native tokens or use established cryptocurrencies like Ethereum's Ether.
   - These tokens can be used for various purposes within the application, such as paying for services, participating in governance, or earning rewards.
7. **Use Cases:**
   - Dapps have a wide range of use cases, including decentralized finance (DeFi), non-fungible tokens (NFTs), supply chain management, voting systems, gaming, and more.
   - Each Dapp is designed to solve specific problems or provide new functionalities in a decentralized manner.

Introduction to Dapp (Decentralized Application)

8. **User Interface:**
   - Dapps often have user interfaces (UIs) similar to traditional apps or websites, making them accessible to a broader user base.
   - Users may not even be aware that they are interacting with blockchain technology.
9. **Challenges:**
   - While Dapps offer numerous advantages, they also face challenges, including scalability issues, user adoption barriers, and regulatory compliance.
   - These challenges are actively being addressed by the blockchain community.
10. **Development:**
   - Building a Dapp typically involves programming smart contracts using languages like Solidity (for Ethereum) and developing a frontend interface.
   - Popular Dapp development frameworks like Truffle and web3.js make the development process more accessible.

Dapp Architecture

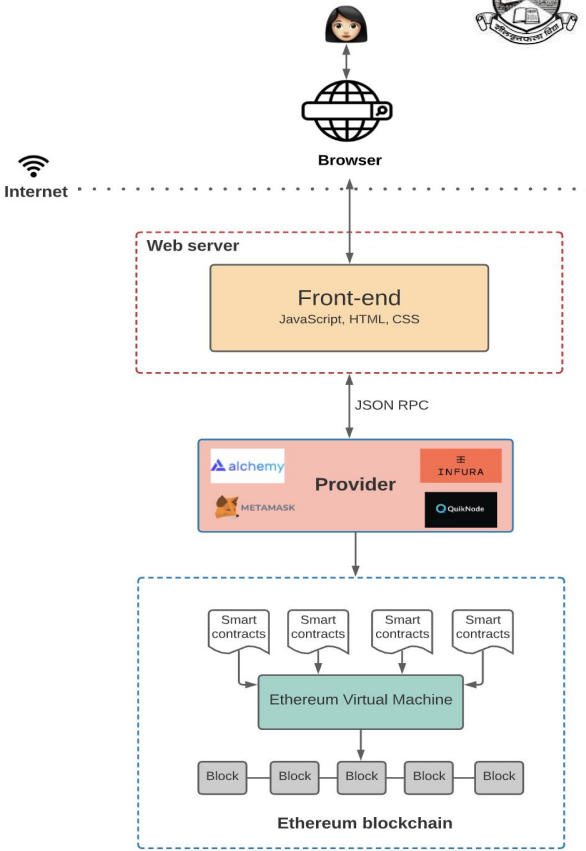
1. **Ethereum blockchain:**
   - Securely executes and verifies application code, called smart contracts.
   - DApps use the Ethereum blockchain for data storage.
2. **Smart Contracts:**
   - DApps use smart contracts to define the state changes happening on the blockchain.
   - A smart contract is a collection of code and data that resides at a specific address on the Ethereum Blockchain and runs on the Ethereum blockchain.
3. **Ethereum Virtual Machine(EVM):**
   - Global virtual computer that executes the logic defined in the smart contracts and processes the state changes that happen on this Ethereum network.
4. **Front-end:**
   - The part of the DApps, that users can see and interact with such as the graphical user interface(GUI),
   - Communicates with the application logic defined in smart contracts.



Courtesy : Preeti - DApp

DApp - How Frontend Code Communicate ?

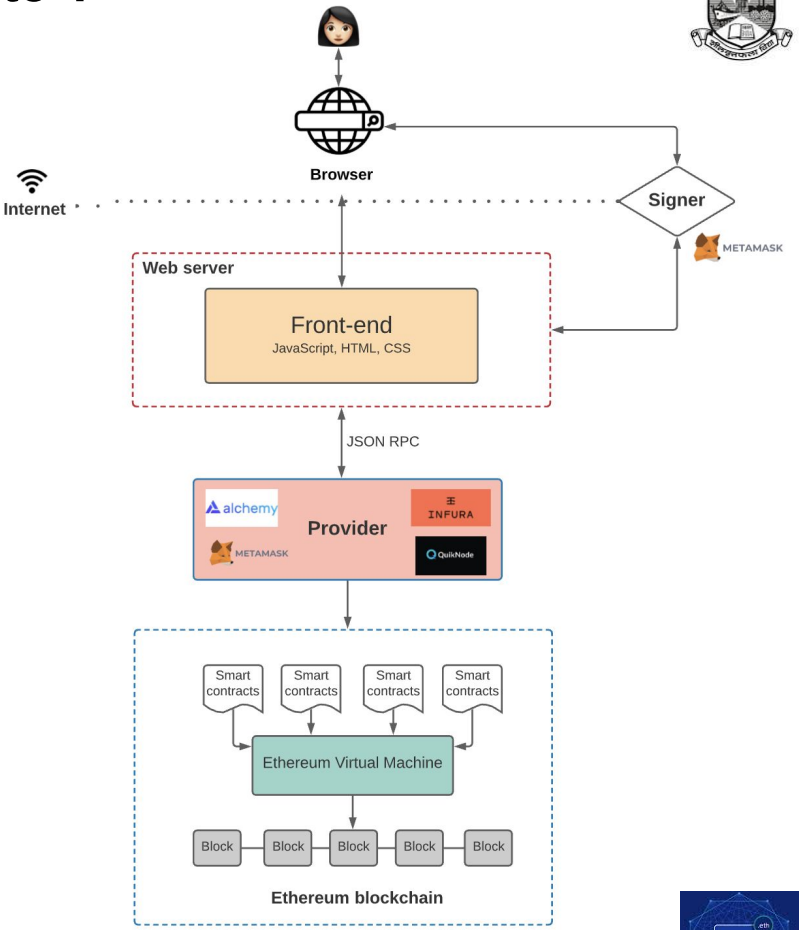
- When we need to interact with the data and code on a blockchain, we need to interact with one of these nodes.
- As any node can broadcast a request for a transaction to be executed on the EVM.
- A miner will then execute the transaction and propagate the resulting state change to the rest of the network.
- There are two ways to broadcast a new transaction:
  1. Set up your own node which runs the Ethereum blockchain software
  2. Use nodes provided by third-party services like Infura, Alchemy, and Quicknode
- Every Ethereum client (i.e. provider) implements a JSON-RPC specification. This ensures that there's a uniform set of methods when frontend applications want to interact with the blockchain.



Courtesy : Preeti - DApp

DApp - How Frontend Code Communicate ?

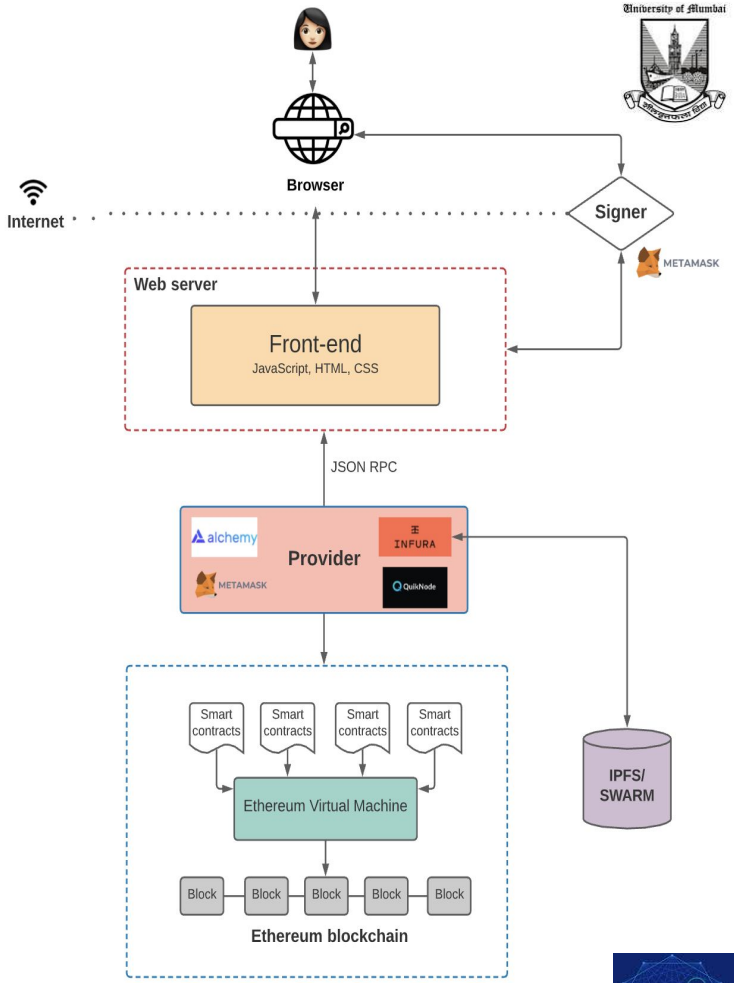
- When a user wants to publish a new post onto the chain, our DApp would ask the user to “sign” the transaction using their private key — only then would the DApp relay the transaction to the blockchain.
- Otherwise, the nodes wouldn’t accept the transaction.
- This “signing” of transactions is where Metamask typically comes in.



Courtesy : Preeti - DApp

DApp - Storage on the Blockchain

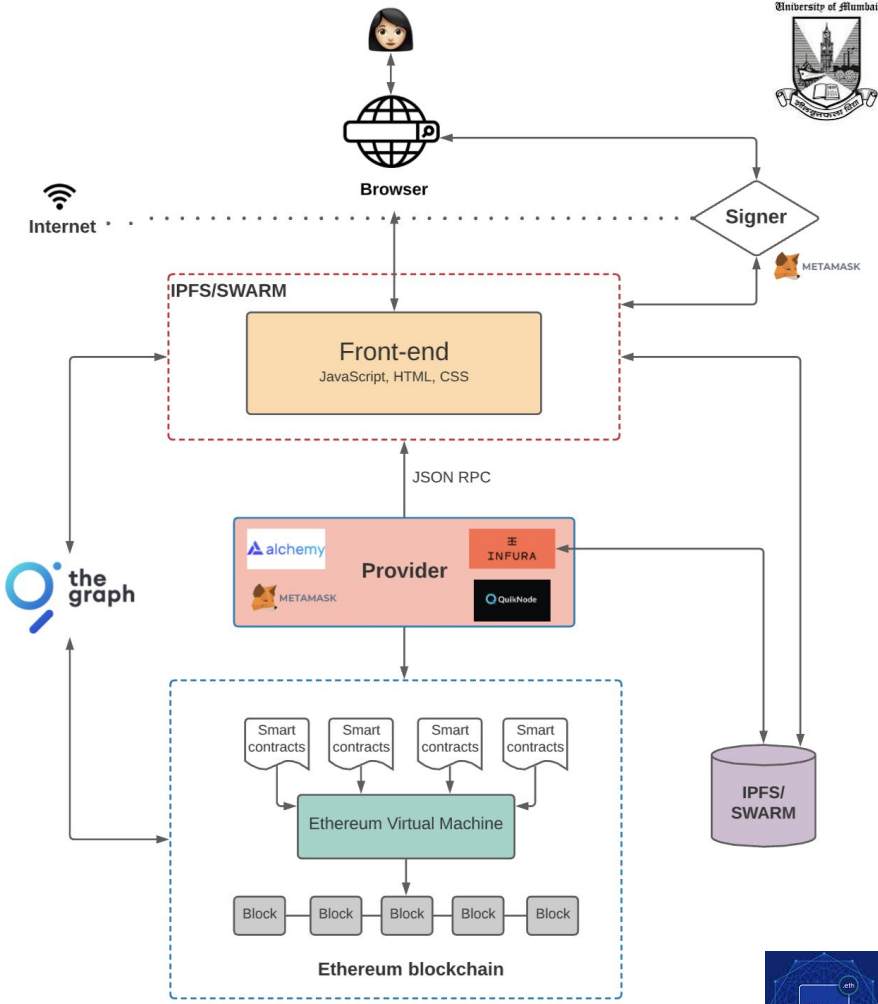
- Storing everything on the blockchain gets really expensive.
- Users to pay extra for using your DApp every time their transaction requires adding a new state is not the best user experience.
- One way to combat this is to **use a decentralized off-chain storage solution**, like IPFS or Swarm.
- IPFS : Distributed file system for storing and accessing data.
  - distributes and stores the data in a peer-to-peer network.
  - Has an incentive layer known as “Filecoin.” This layer incentivizes nodes around the world to store and retrieve this data.
  - IPFS providers
    - **Infura** (which provides you with an IPFS node) or
    - Pinata (“pin” your files to IPFS and take the IPFS hash and store that on the blockchain).
- **Swarm** : A decentralized storage network,
  - **Difference :** Swarm’s incentive system is built-in and enforced through smart contracts on the Ethereum blockchain for storing and retrieving data.



Courtesy : Preeti - DApp

DApp Querying the Blockchain

1. **Smart Contract Events :**
   - Use the Web3.js library to query and listen for smart contract events.
   - Here, we need to listen to specific events and specify a callback every time the event is fired.
   - using callbacks to handle various UI logic gets very complex
   - Issue: when you deploy a smart contract and later realize you need an event emitted that you didn't originally include.
2. **The Graph**
   - An off-chain indexing solution that makes it easier to query data on the Ethereum blockchain.
   - Allows to define which smart contracts to index, which events and function calls to listen to, and how to transform incoming events into entities that the frontend logic can consume.
   - It uses **GraphQL** as a query language,
   - By indexing blockchain data, The Graph lets us query on-chain data in our application logic with low latency

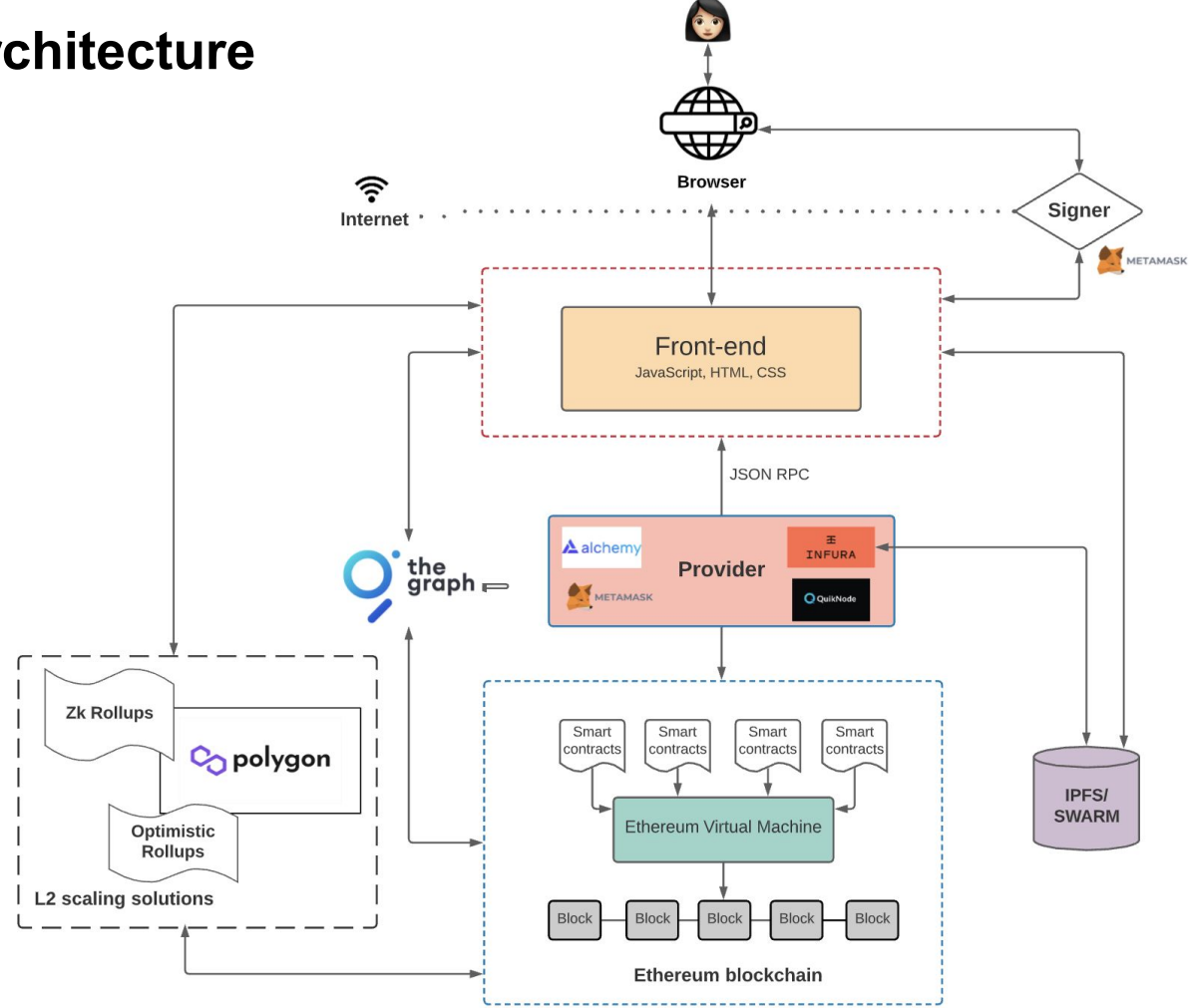


Courtesy : Preeti - DApp

Overall DApp Architecture

Addressing

- Storage,
- Querying
- Scalability



Courtesy : Preeti - DApp

Ethereum Networks

- groups of connected computers that communicate using the Ethereum protocol.
- There is **only one Ethereum Mainnet,**
- But **independent networks**
  - conforms to the **same protocol rules** can be created
  - for testing and development purposes.
  - These independent "networks" that conform to the protocol **without interacting with each other.**

Note :

- One can even start one locally on your own computer for testing your smart contracts and web3 apps.
- Eg : Geth & Ganache Private Networks created during the Lab Experiments
- Yout **Ethereum account will work across the different networks**,
- But your account balance and transaction history won't carry over from the main Ethereum network.

For testing purposes :

- it's useful to know which networks are available
- how to get testnet ETH to play around with.

For security considerations : it's not recommended to reuse mainnet accounts on testnets or vice versa.

Courtesy : Ethereum

Ethereum Networks

- **PUBLIC NETWORKS**
  - Public networks are accessible to anyone in the world with an internet connection.
  - Anyone can read or create transactions on a public blockchain and validate the transactions being executed.
  - The consensus among peers decides on the inclusion of transactions and the state of the network.
- **Ethereum Mainnet**
  - Mainnet is the primary public Ethereum production blockchain,
  - where actual-value transactions occur on the distributed ledger.
  - When people and exchanges discuss ETH prices, they're talking about Mainnet ETH.

Courtesy : Ethereum

Ethereum Networks

- **Ethereum Testnets**
  - These are networks used by protocol developers or smart contract developers to test both protocol upgrades as well as potential smart contracts in a production-like environment before deployment to Mainnet.
  - Test any contract code you write on a testnet before deploying to Mainnet.
  - Among dapps that integrate with existing smart contracts, most projects have copies deployed to testnets.
  - Most testnets started by using a permissioned proof-of-authority consensus mechanism.
  - This means a small number of nodes are chosen to validate transactions and create new blocks – staking their identity in the process.
  - Alternatively, some testnets feature an open proof-of-stake consensus mechanism where everyone can test running a validator, just like Ethereum Mainnet.
  - **ETH on testnets is supposed to have no real value**;
  - Most people get testnet ETH for free from faucets.
  - Most faucets are webapps where you can input an address which you request ETH to be sent to.

Ethereum Networks

- **Which Testnet should I use?**
  - The two public testnets that client developers are **Sepolia** and **Goerli**.
  - **Sepolia**
    - recommended default testnet for application development.
    - Features :
      - Closed validator set, controlled by client & testing teams
      - New testnet, less applications deployed than other testnets
      - Fast to sync and running a node requires minimal disk space
  - **Goerli (long-term support)**
    - the Goerli testnet is deprecated and will be replaced by Holesovice in 2023.
    - Features :
      - Open validator set, stakers can test network upgrades
      - Large state, useful for testing complex smart contract interactions
      - Longer to sync and requires more storage to run a node

Courtesy : Ethereum

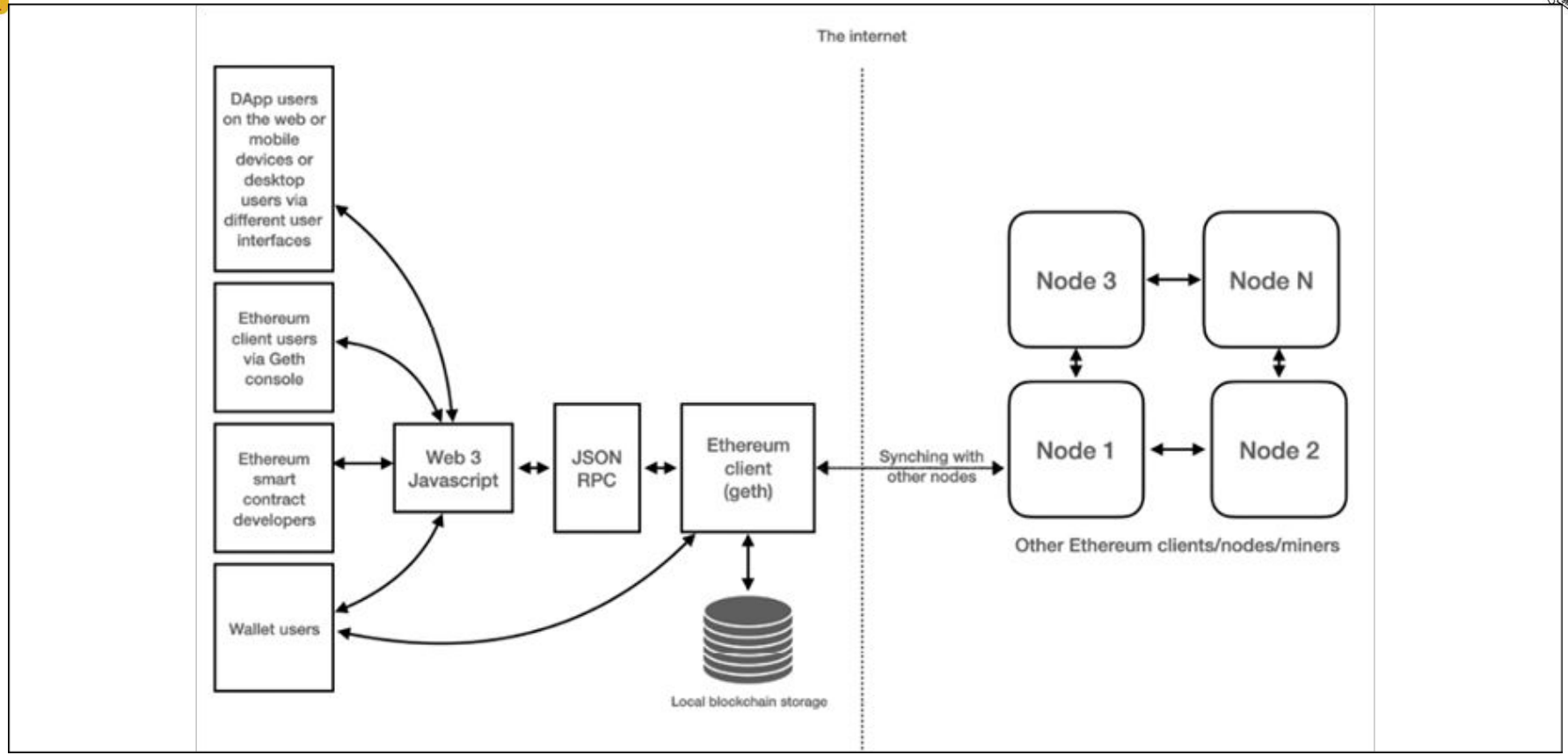
Ethereum Networks

- **PRIVATE NETWORKS**
  - An Ethereum network is a private network if its nodes are not connected to a public network (i.e. Mainnet or a testnet).
  - Private only means reserved or isolated, rather than protected or secure.
- **Development networks**
  - To develop an Ethereum application, run it on a private network to see how it works before deploying.
  - Create a local blockchain instance to test your dapp.
  - This allows for much faster iteration than a public testnet.
- **Consortium networks**
  - The consensus process is controlled by a pre-defined set of nodes that are trusted.
  - For example :
    - a private network of known academic institutions that each govern a single node,
    - and blocks are validated by a threshold of signatories within the network.

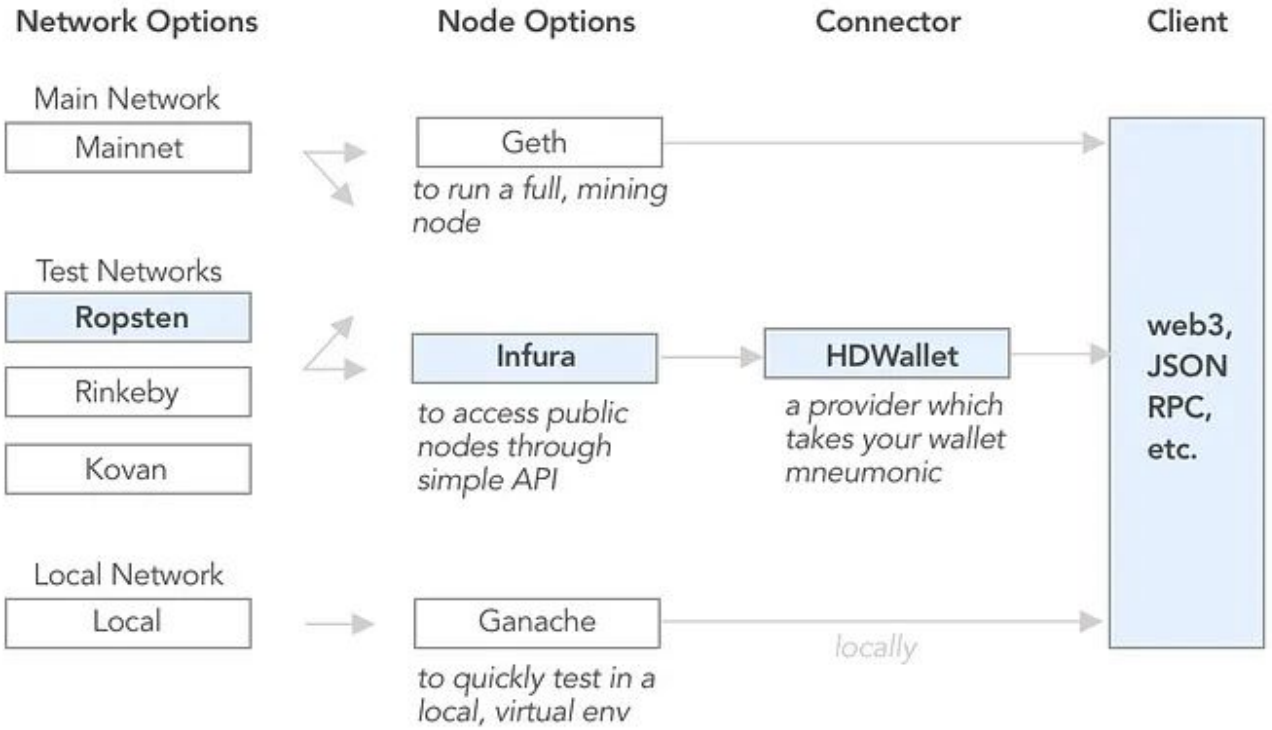
Note :

If a **public Ethereum network ⇒ public internet**, a **consortium network ⇒ private intranet.**

Ethereum High level Ecosystem

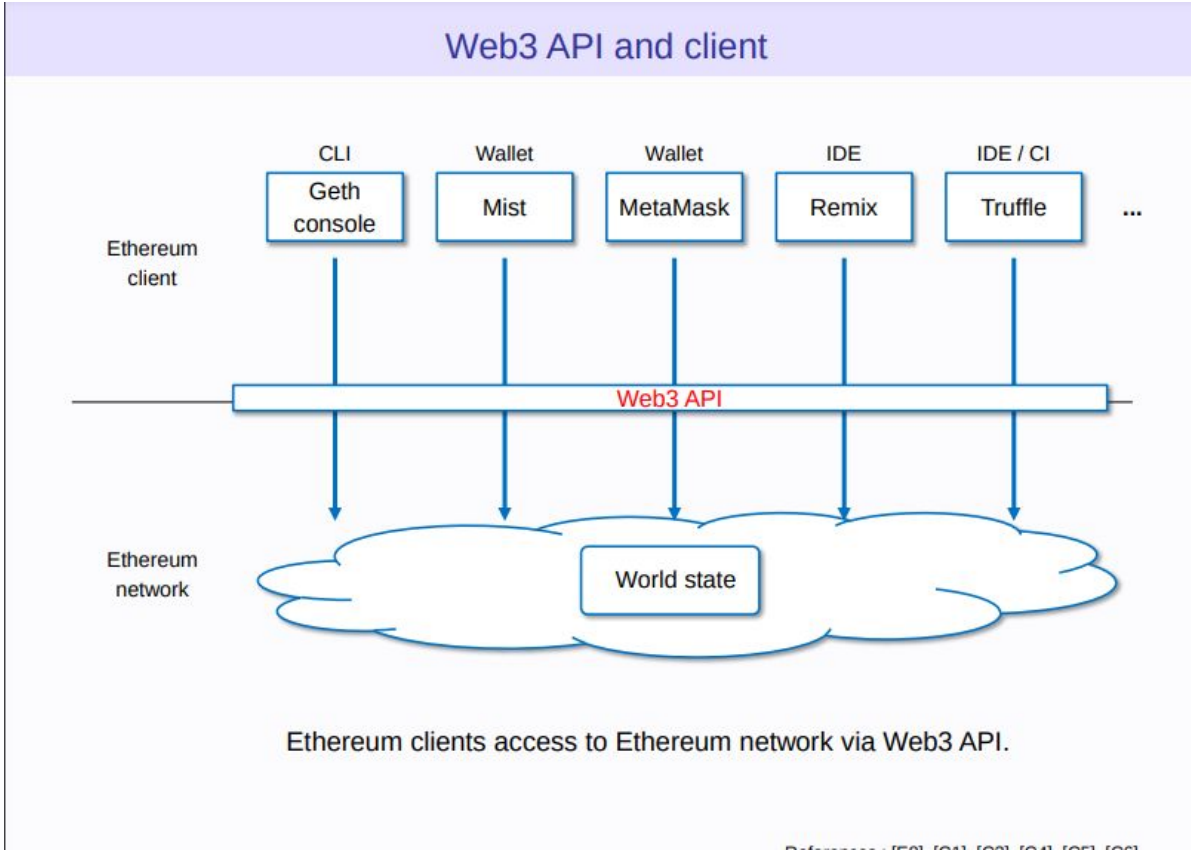


Ethereum Smart Contract Deployment Options



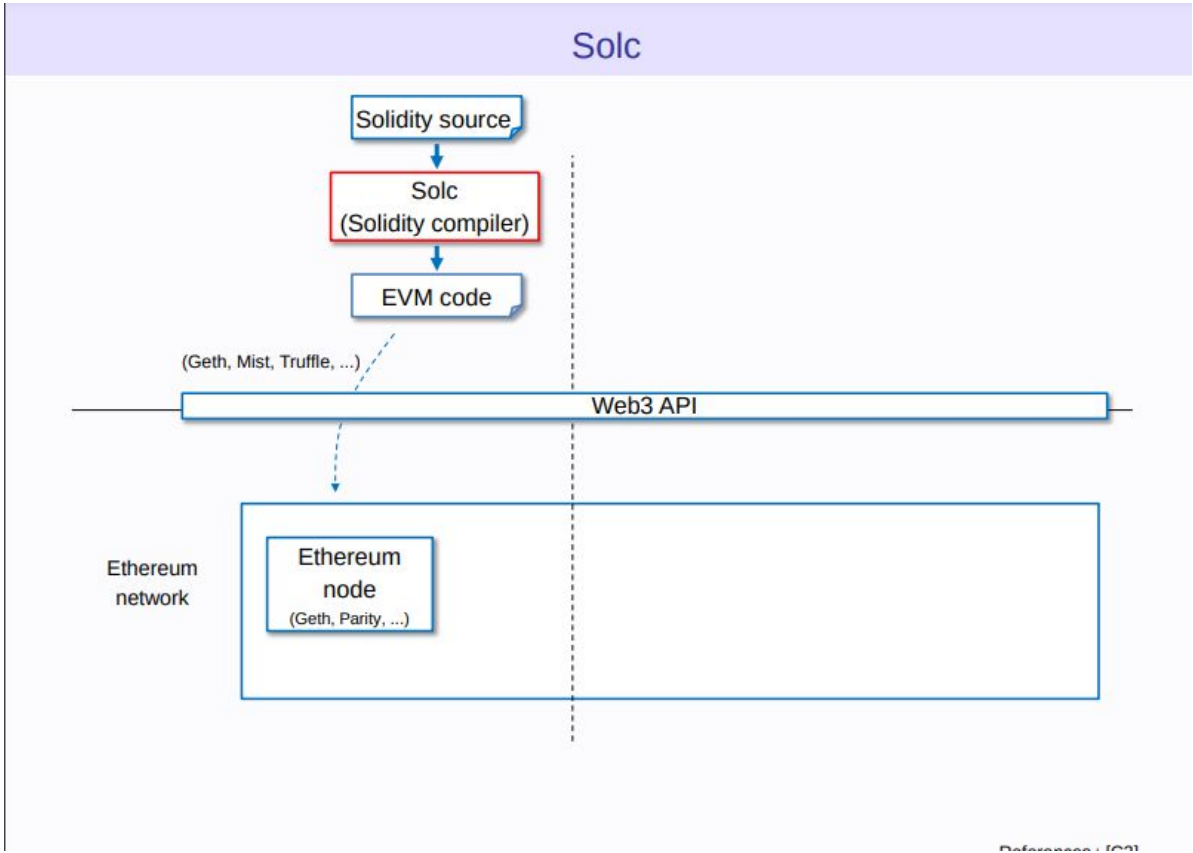
Courtesy : Medium - Coinmonks

Ethereum Blockchain Deployment via Web3 API

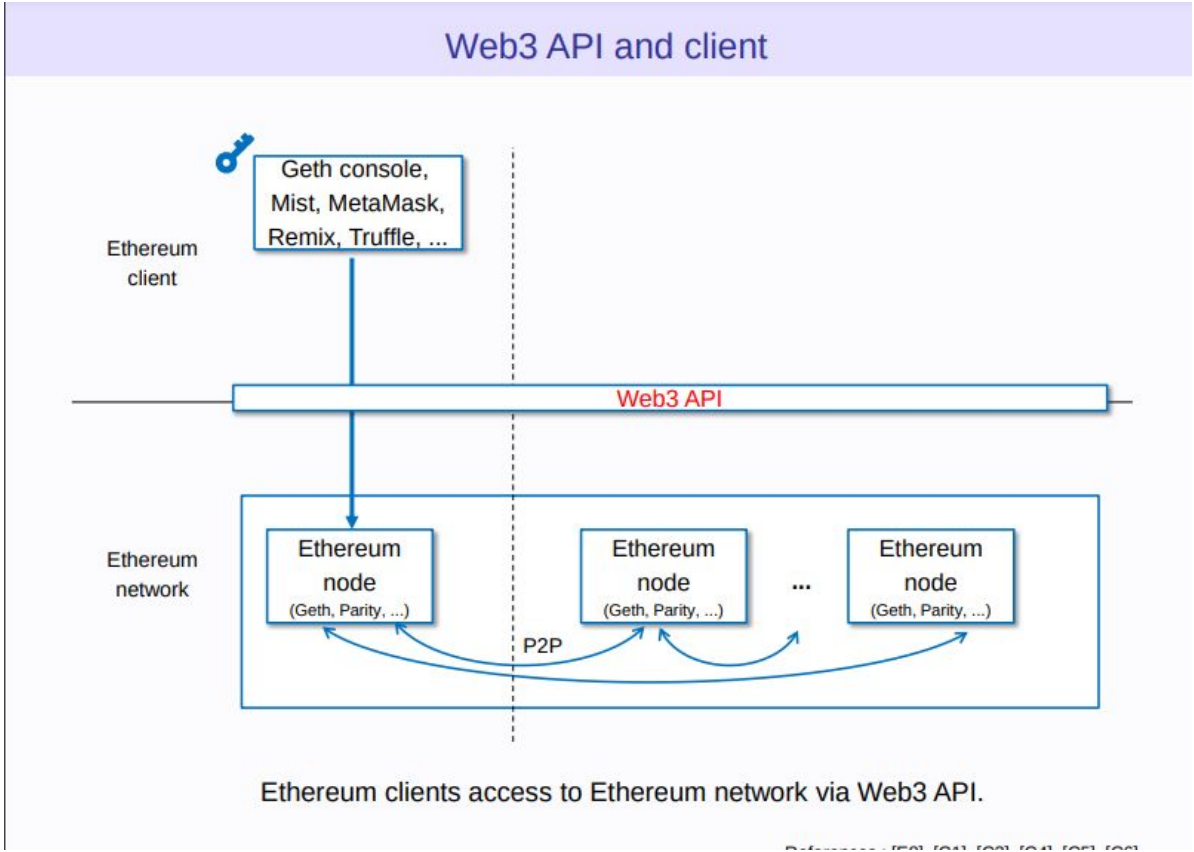


Courtesy : Takenobu T

Ethereum Blockchain Deployment via Web3 API

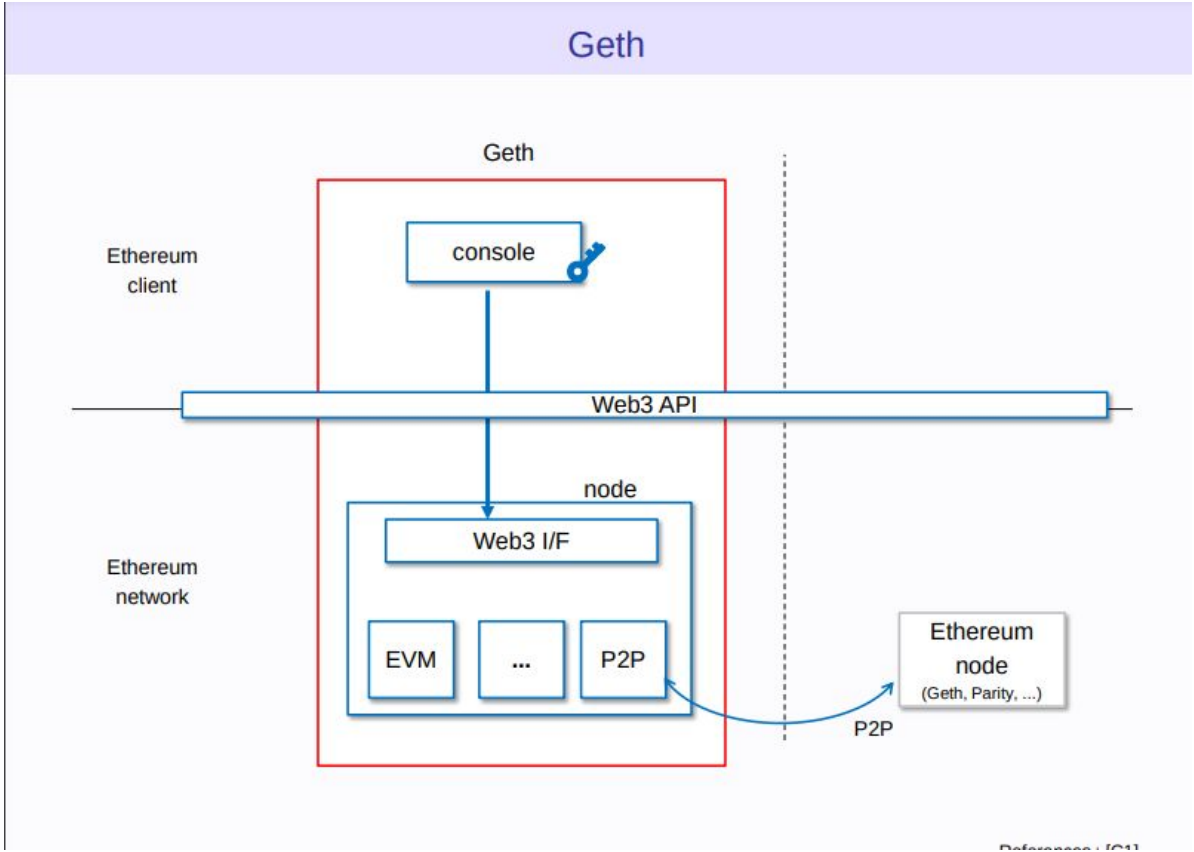


Ethereum Blockchain Deployment via Web3 API and Client



Courtesy : Takenobu T

Ethereum Blockchain Deployment via Web3 API and Geth



Courtesy : Takenobu T

STO Metamask (Ethereum Wallet)

- STO Metamask - Security Token Offerings using Metamask (Similar to ICO)
- a **free crypto wallet software** that people can use to interact in the crypto world.
- **free to use** and **can be installed as an extension on internet browsers**, Google Chrome, Firefox, Brave, and Edge, or downloaded as a smartphone application both on iOS and Android.
- With over **30 million users, MetaMask is one of the most popular cryptocurrency** wallets today.
- used for managing Ethereum and Ethereum-based tokens.
- Metamask allows users to:
  - Buy, receive, send and swap Ether, the main token on Ethereum
  - Buy, receive, send and swap nonfungible tokens (NFTs) in marketplaces
  - Connect to Ethereum dapps
  - Connect to other crypto wallets
  - Play blockchain-based games
  - Access different networks such as the BNB Smart Chain and other testnets

Courtesy : Cointelegraph - Metamask

METAMASK

Benefits of using Metamask (Ethereum Wallet)

Its massive global user and follower base make it a vital part of the Ethereum community.

Utilizes local key storage, giving users full control over their keys. No personal information is stored either.

Allows in-app coin purchasing – users can buy Ether and ERC-20 directly from Coinbase and ShapeShift.

Allows multiple accounts, which is great for people who want to keep their finances separate.

Backs up the user account with hierarchical deterministic settings.

Has an easy-to-use interface that is good for crypto beginners.

Easily connects with popular exchanges like Coinbase and Binance.

Courtesy : Cointelegraph - Metamask

METAMASK

Advantages of Metamask (Ethereum Wallet)

1. **User-Friendly Interface:** MetaMask provides a relatively intuitive and user-friendly interface, making it accessible to both beginners and experienced users.
2. **Security:** MetaMask is considered secure, with features like seed phrases for wallet recovery, password protection, and integration with hardware wallets like Ledger and Trezor for enhanced security.
3. **Cross-Platform:** It is available as a browser extension for Chrome, Firefox, Brave, and Edge, as well as a mobile app for iOS and Android, ensuring accessibility across multiple platforms.
4. **Compatibility:** MetaMask supports Ethereum and Ethereum-based tokens (ERC-20 and ERC-721 tokens), making it versatile for managing various assets within the Ethereum ecosystem.
5. **DApp Integration:** It allows users to interact with a wide range of Ethereum-based DApps directly through their browser extension, simplifying the DApp experience.
6. **Privacy:** While Ethereum is a public blockchain, MetaMask enables users to have a degree of privacy by not revealing their real-world identity when transacting or interacting with DApps.
7. **Community Support:** MetaMask has a strong community and active development, which means regular updates, bug fixes, and improvements.

Courtesy : Cointelegraph - Metamask

METAMASK

Disadvantages of Metamask (Ethereum Wallet)

1. **Security Risks:** While MetaMask itself is secure, users can still fall victim to phishing scams and malicious websites. Users must be cautious and ensure they are using the official MetaMask extension or app.
2. **Limited to Ethereum:** As of my last knowledge update in September 2021, MetaMask primarily supports Ethereum and Ethereum-based tokens. While there are efforts to expand to other blockchains, it may not be the best choice for managing assets on other networks.
3. **Transaction Fees:** Ethereum transaction fees (gas fees) can be high during times of network congestion, making it costly to perform transactions or interact with DApps.
4. **Learning Curve:** For individuals new to cryptocurrencies and blockchain technology, MetaMask may have a learning curve, particularly when it comes to understanding gas fees, wallet management, and security practices.
5. **Lack of Customer Support:** MetaMask is an open-source project, and while it has a strong community, there is no official customer support. Users must rely on community resources for assistance.
6. **Centralization Concerns:** Some users may have concerns about the centralization of DApp interactions through a browser extension like MetaMask, as it relies on centralized infrastructure to communicate with the Ethereum network.

Courtesy : Cointelegraph - Metamask

METAMASK

Setup a Development Environment using Metamask

1. Install MetaMask:

- install the MetaMask browser extension or mobile app on your preferred browser or device.
- You can find MetaMask on Chrome, Firefox, Brave, Edge, and as a mobile app for iOS and Android.

2. Create or Import a Wallet:

- Follow the on-screen instructions to set up a wallet, including creating a strong password and securely storing your recovery seed phrase.
- If you already have a MetaMask wallet, you can import it using your existing seed phrase or private key.

3. Switch to a Test Network:

- In a development environment, it's common to use test networks like the Ropsten, Rinkeby, or Kovan testnets to avoid spending real Ether on testing transactions.
- To switch to a test network in MetaMask:
  - Click on your MetaMask extension or app to open it.
  - Click on the network name (usually "Main Ethereum Network" by default) at the top.
  - Select "Custom RPC" if your desired test network is not listed, and then enter the network's details (such as the network name, RPC URL, and chain ID).

Courtesy : Cointelegraph - Metamask

METAMASK

Setup a Development Environment using Metamask

4. Fund Your Test Wallet:

- To interact with the test network, you'll need test Ether (tEther) instead of real Ether.
- tEther can be obtained from a faucet specific to the test network you're using.
- **Faucets** are websites that provide free test Ether for development purposes. Search online for the test network's faucet and follow the instructions to get some tEther.

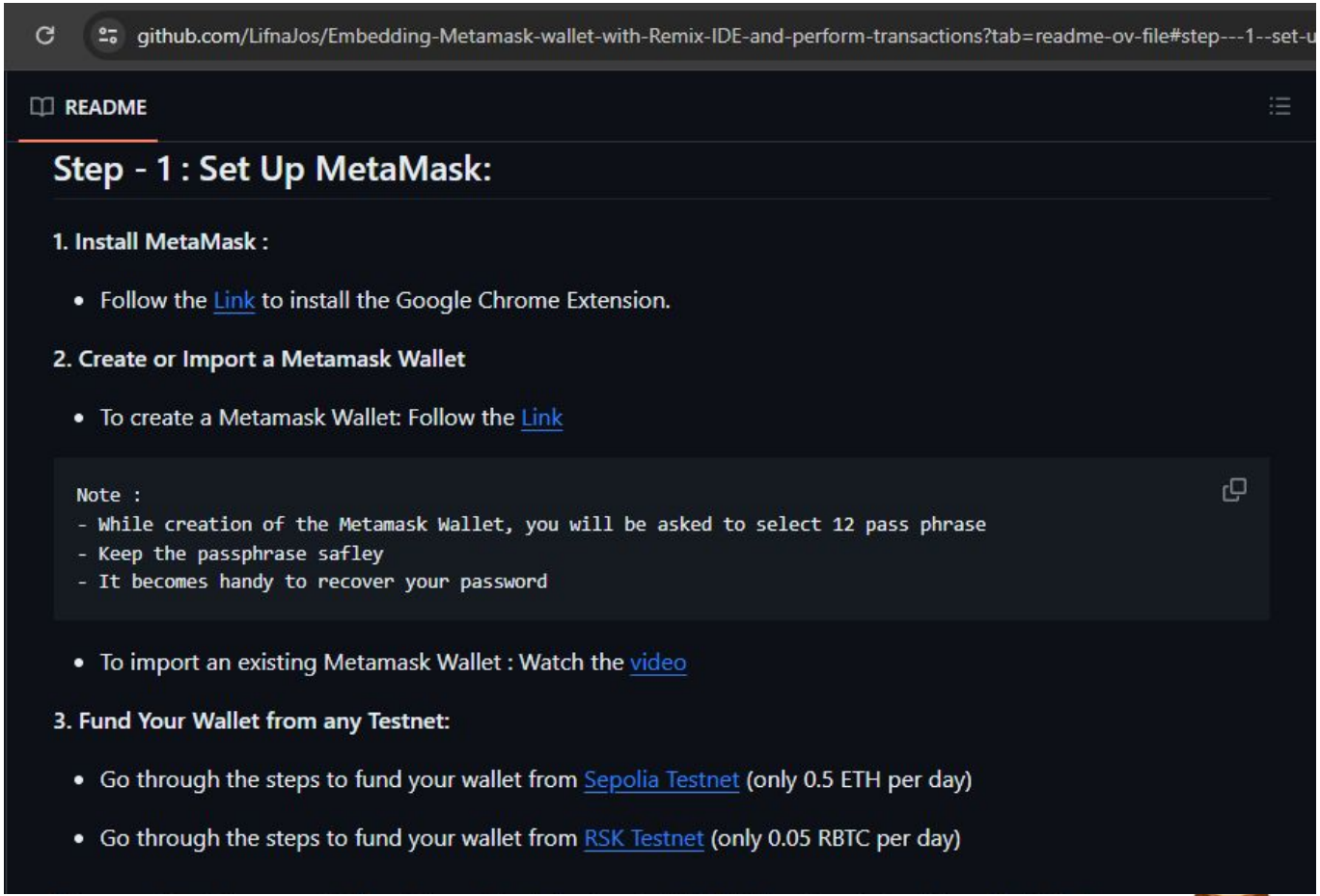
5. Start Developing and Testing:

- With your MetaMask wallet configured for the test network and funded with tEther, you can now start developing and testing your decentralized applications (DApps) or smart contracts.
- Use tools like Remix, Truffle, or Hardhat for Ethereum development and connect them to MetaMask to interact with your wallet and the blockchain.
- When interacting with your DApps or smart contracts, MetaMask will prompt you to confirm transactions and sign messages as needed. Remember that transactions on the test network are not real and won't affect your actual cryptocurrency holdings.
- By setting up a development environment with MetaMask, you can test your applications and smart contracts in a safe and controlled environment before deploying them to the Ethereum mainnet.

Courtesy : Cointelegraph - Metamask

METAMASK

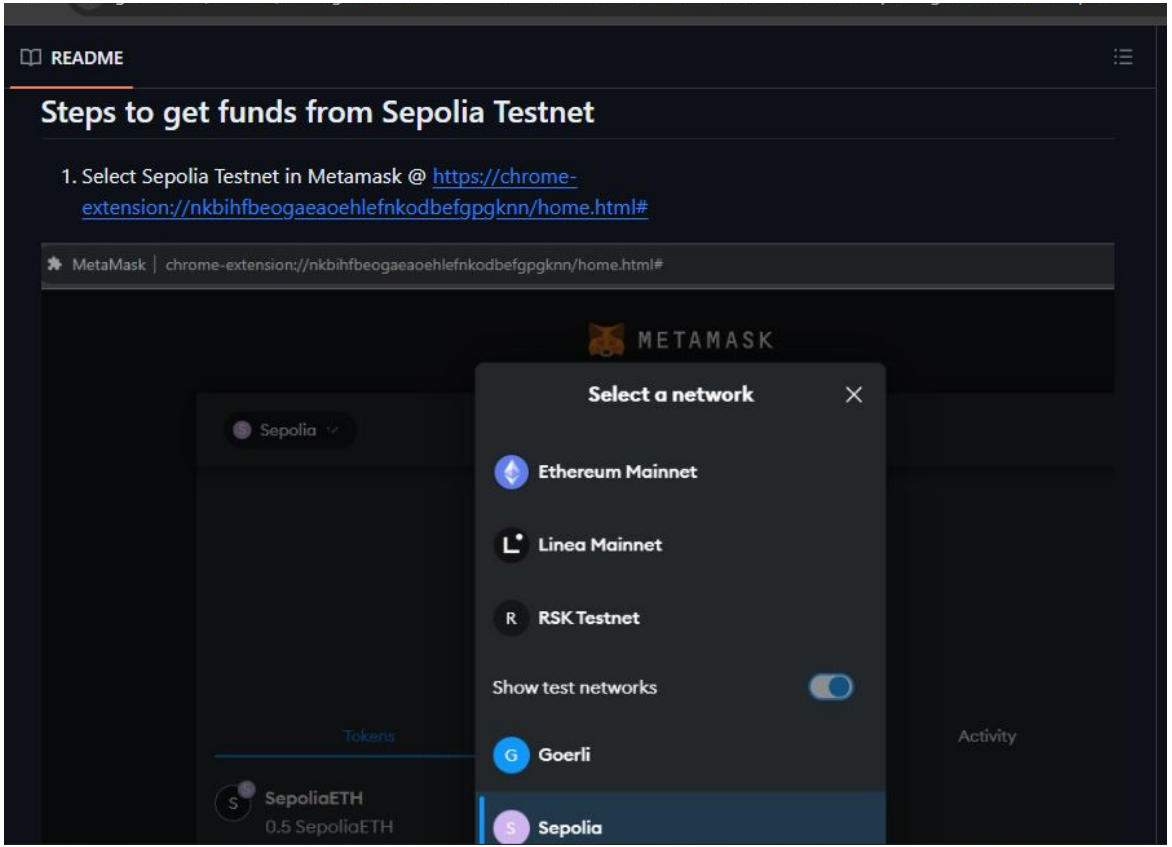
Setup a Development Environment using Metamask



Courtesy : Connect Metamask with Remix

METAMASK

Getting Funds from testnets to Metamask Wallet



Courtesy : Getting Funds from Testnets to Metamask Wallet

METAMASK

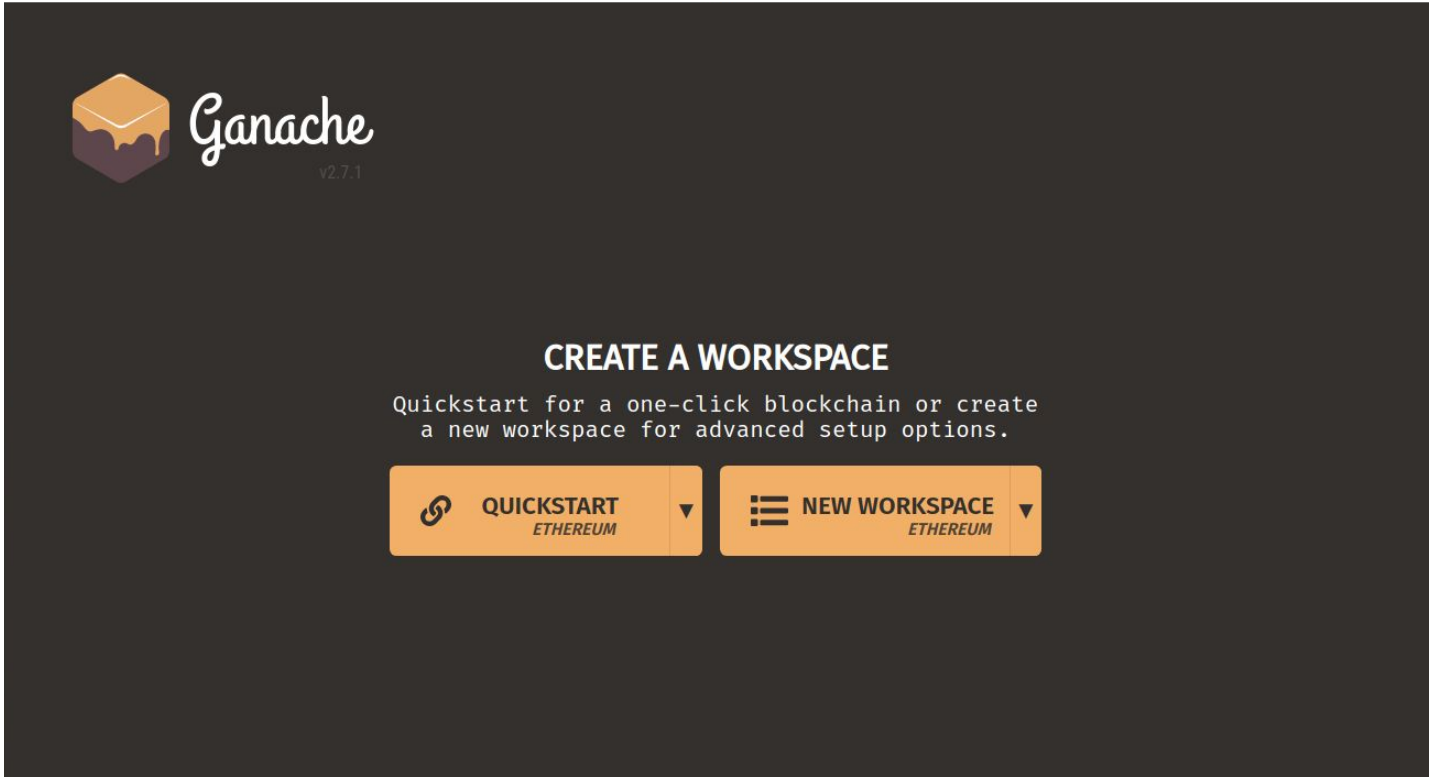
Introduction to Ganache for Ethereum blockchain

- Ganache is a personal blockchain for rapid Ethereum and Filecoin distributed application development.
- One can use Ganache across the entire development cycle;
- enabling you to develop, deploy, and test your dApps in a safe and deterministic environment.
- Ganache comes in two flavors: a UI and CLI.
    - Ganache UI is a desktop application supporting Ethereum and Filecoin technology.
    - More robust command-line tool, ganache, is available for Ethereum development. It offers:
        - console.log in Solidity
        - Zero-config Mainnet and testnet forking
        - Fork any Ethereum network without waiting to sync
        - Ethereum JSON-RPC support
        - Snapshot/revert state
        - Mine blocks instantly, on demand, or at an interval
        - Fast-forward time
        - Impersonate any account (no private keys required!)
        - Listens for JSON-RPC 2.0 requests over HTTP/WebSockets
        - Programmatic use in Node.js
        - Pending Transactions

All versions of Ganache are available for Windows, Mac, and Linux.

Introduction to Ganache for Ethereum blockchain

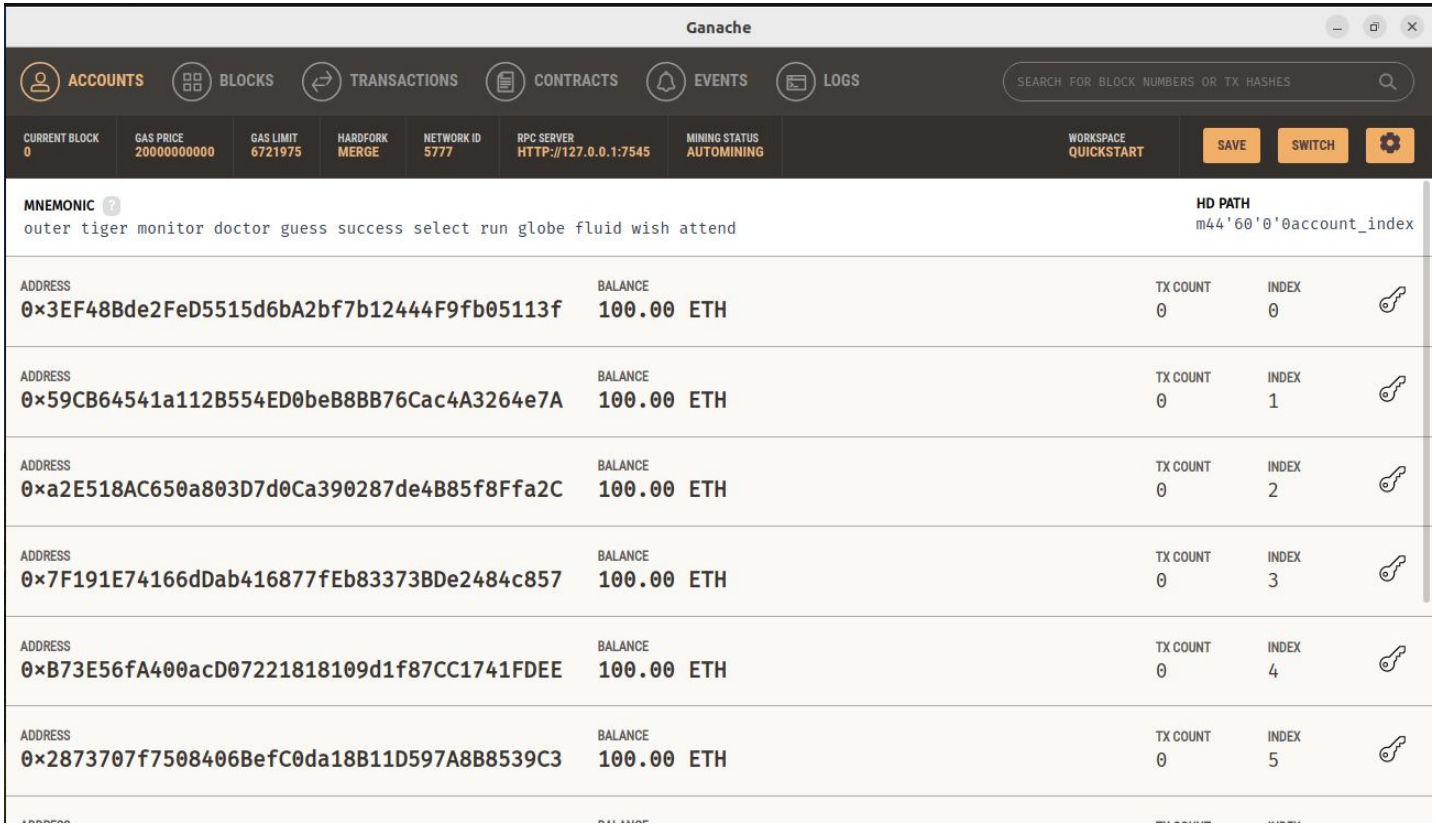
Flash screen of the Ganache after installation, Click on **Quickstart** to create a blockchain



Courtesy : Github - Ganache

Introduction to Ganache for Ethereum blockchain

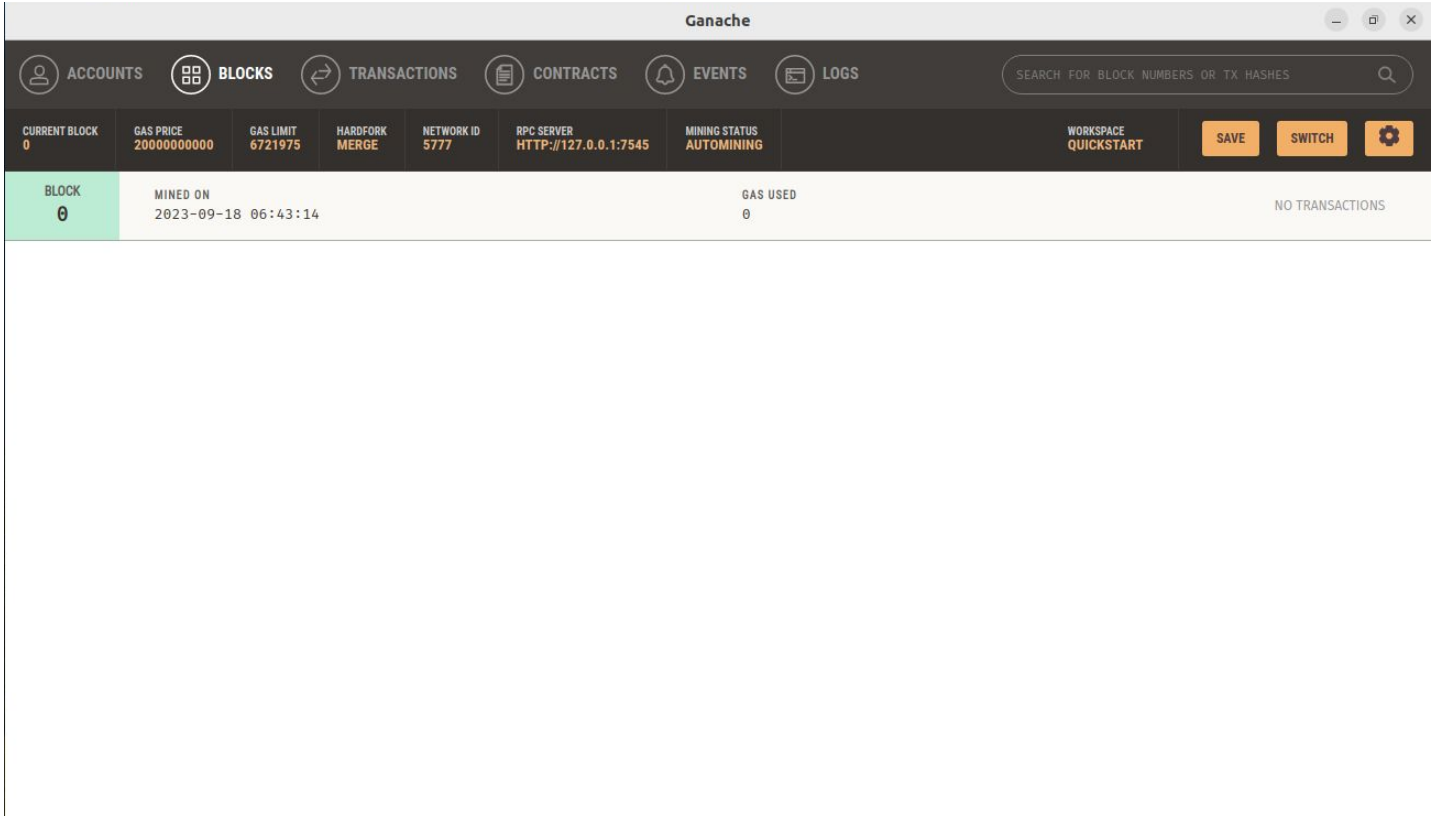
On the Accounts tab, enlists the 10 Accounts created each with 100 Ethers



Courtesy : Github - Ganache

Introduction to Ganache for Ethereum blockchain

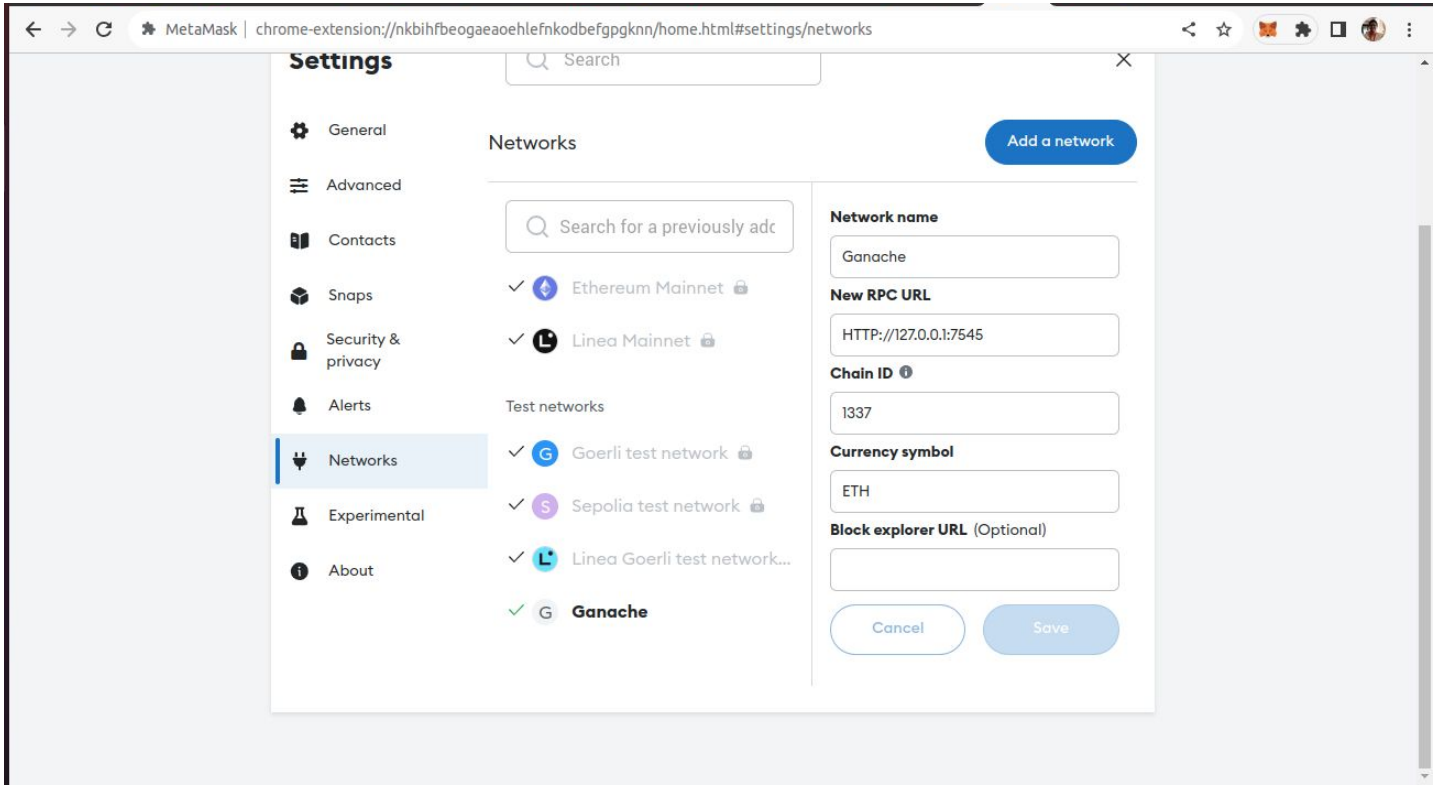
On the Blocks Tab, we could see the Genesis Block created



Courtesy : Github - Ganache

Introduction to Ganache for Ethereum blockchain

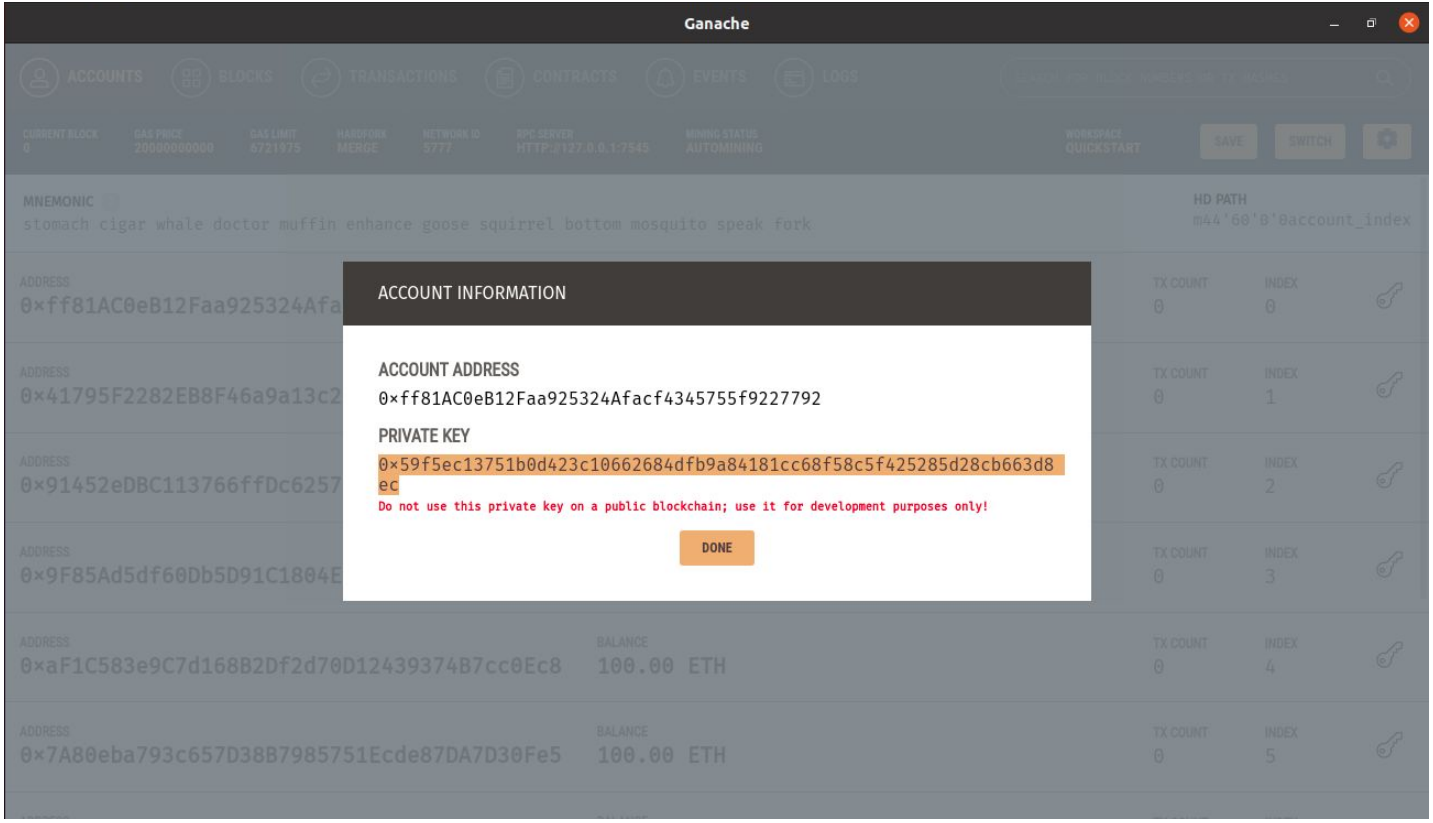
On the Metamask, add Ganache Network as follows:



Courtesy : Github - Ganache

Introduction to Ganache for Ethereum blockchain

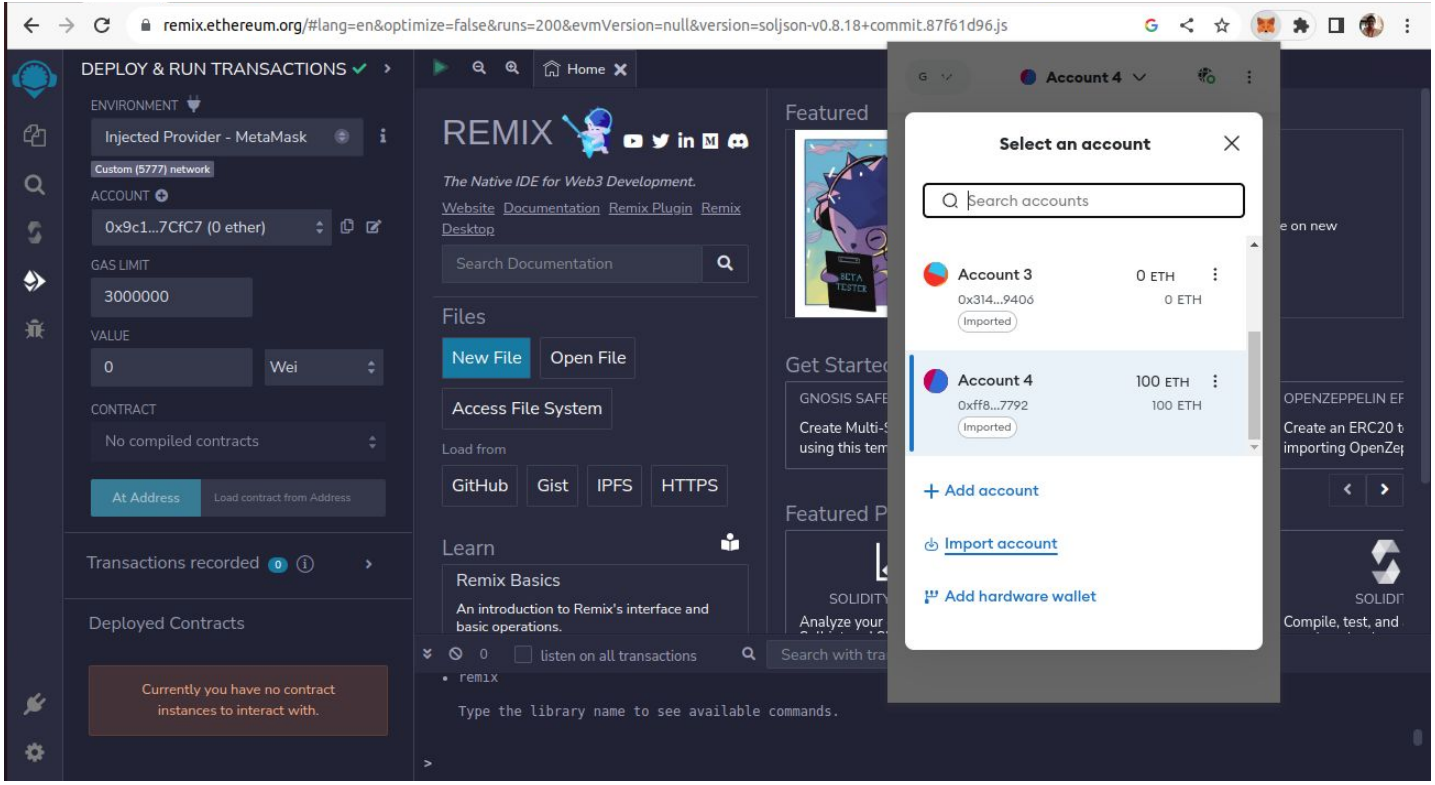
From on account, copy the Private key



Courtesy : Github - Ganache

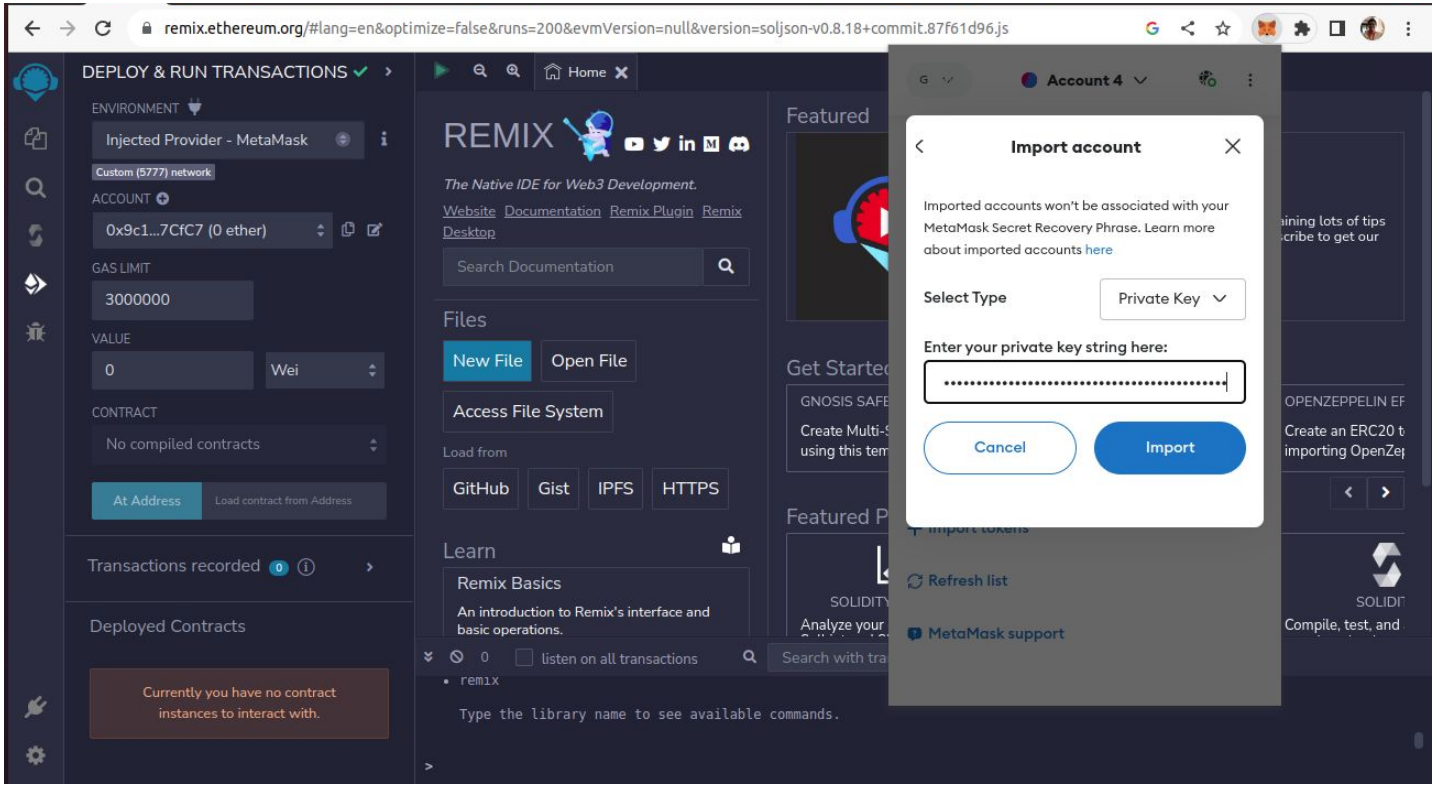
Introduction to Ganache for Ethereum blockchain

On the Metamask, select **Import Account**



Introduction to Ganache for Ethereum blockchain

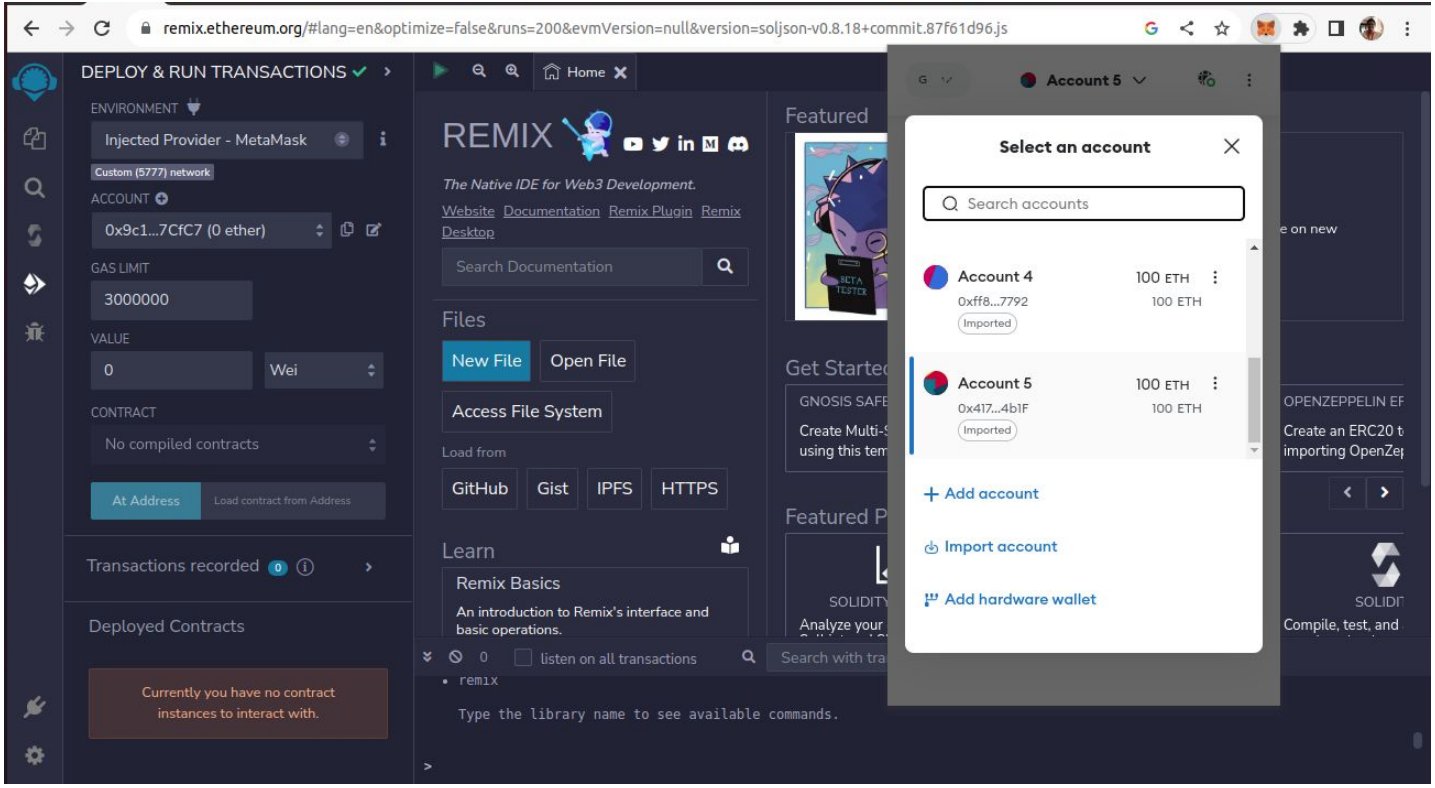
While importing Account on the Metamask, paste the Private key



Courtesy : Github - Ganache

Introduction to Ganache for Ethereum blockchain

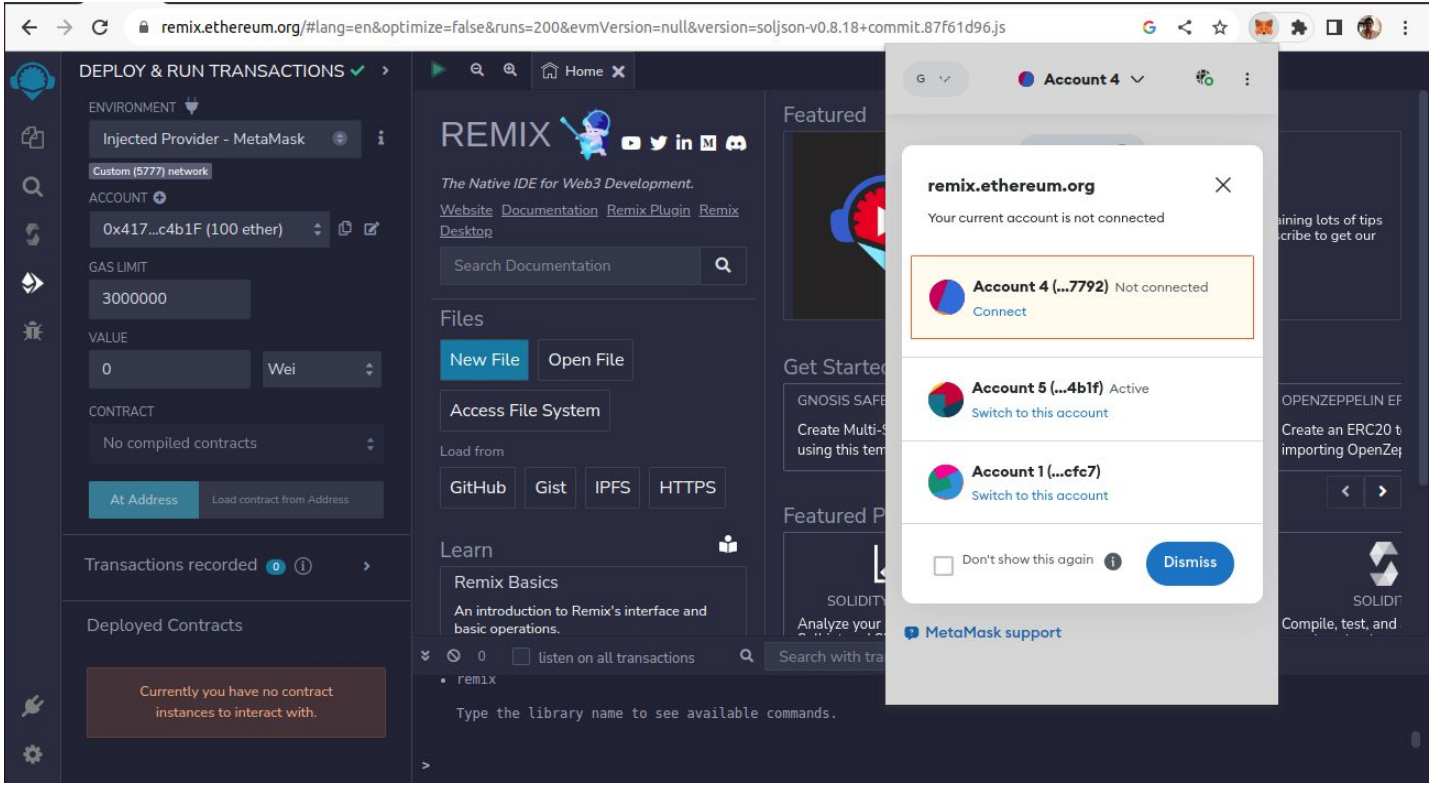
Enlists the Account imported



Courtesy : Github - Ganache

Introduction to Ganache for Ethereum blockchain

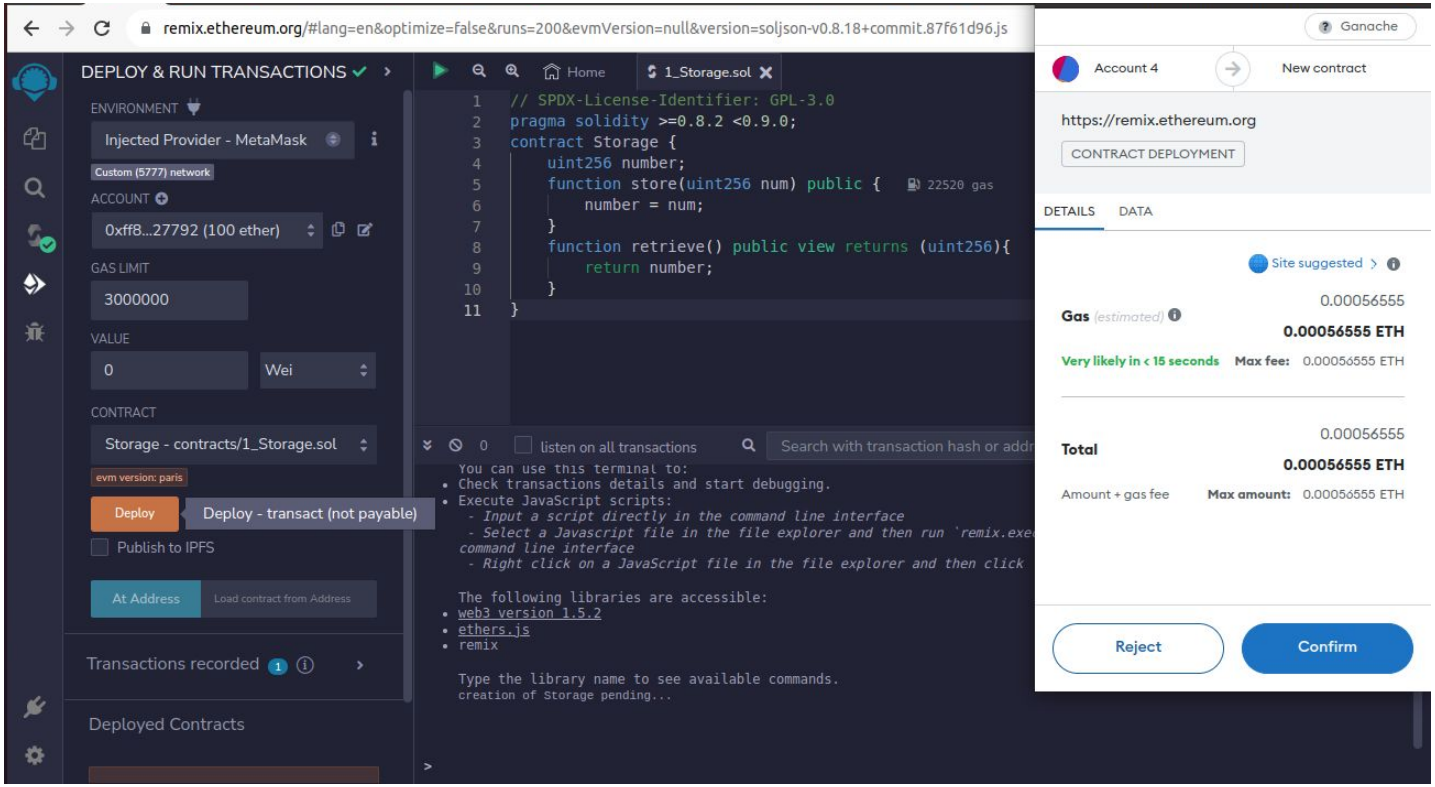
Select the Account from Metamask to be connected with Remix IDE



Courtesy : Github - Ganache

Introduction to Ganache for Ethereum blockchain

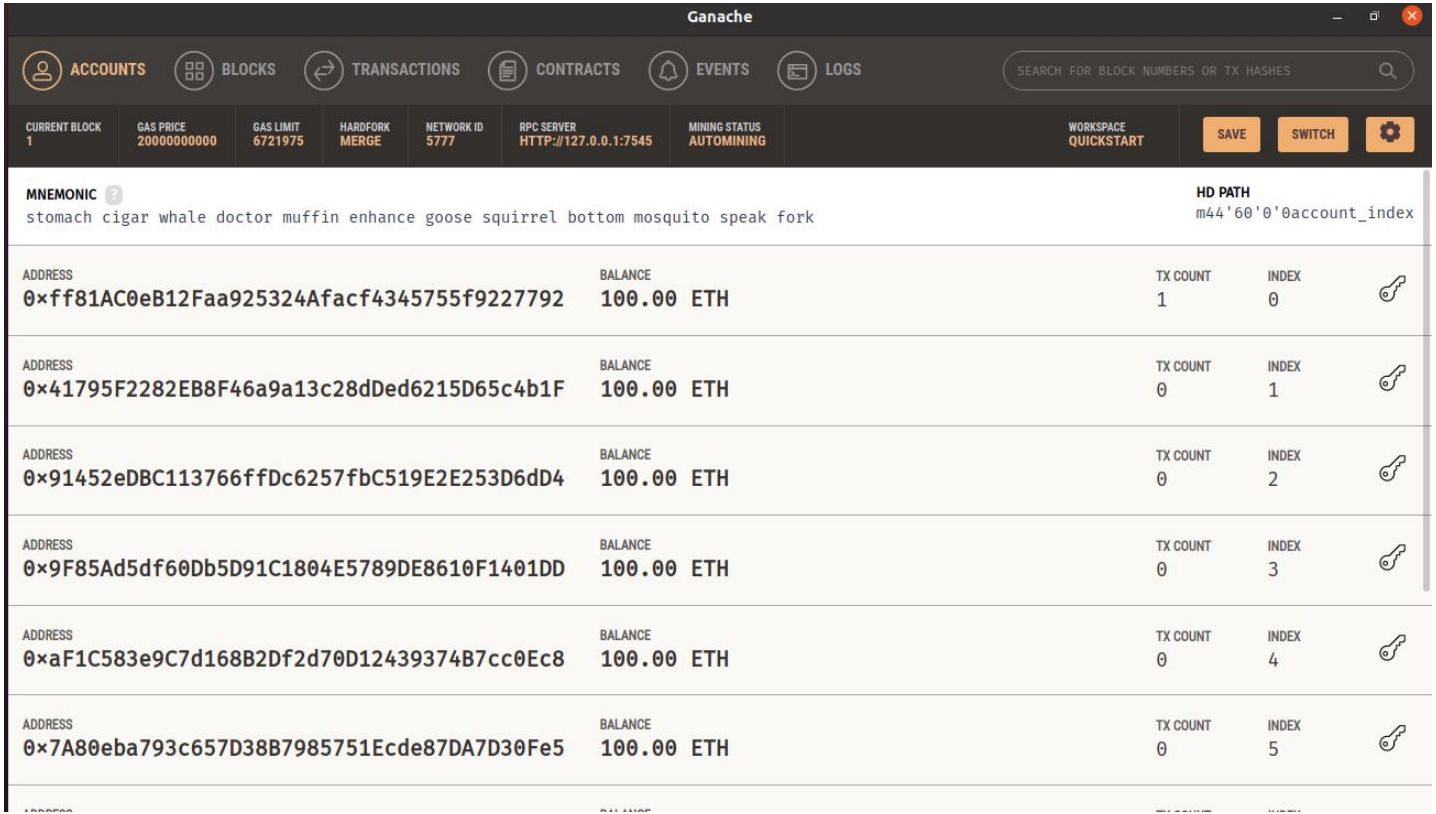
Deploy the Smart Contract



Courtesy : Github - Ganache

Introduction to Ganache for Ethereum blockchain

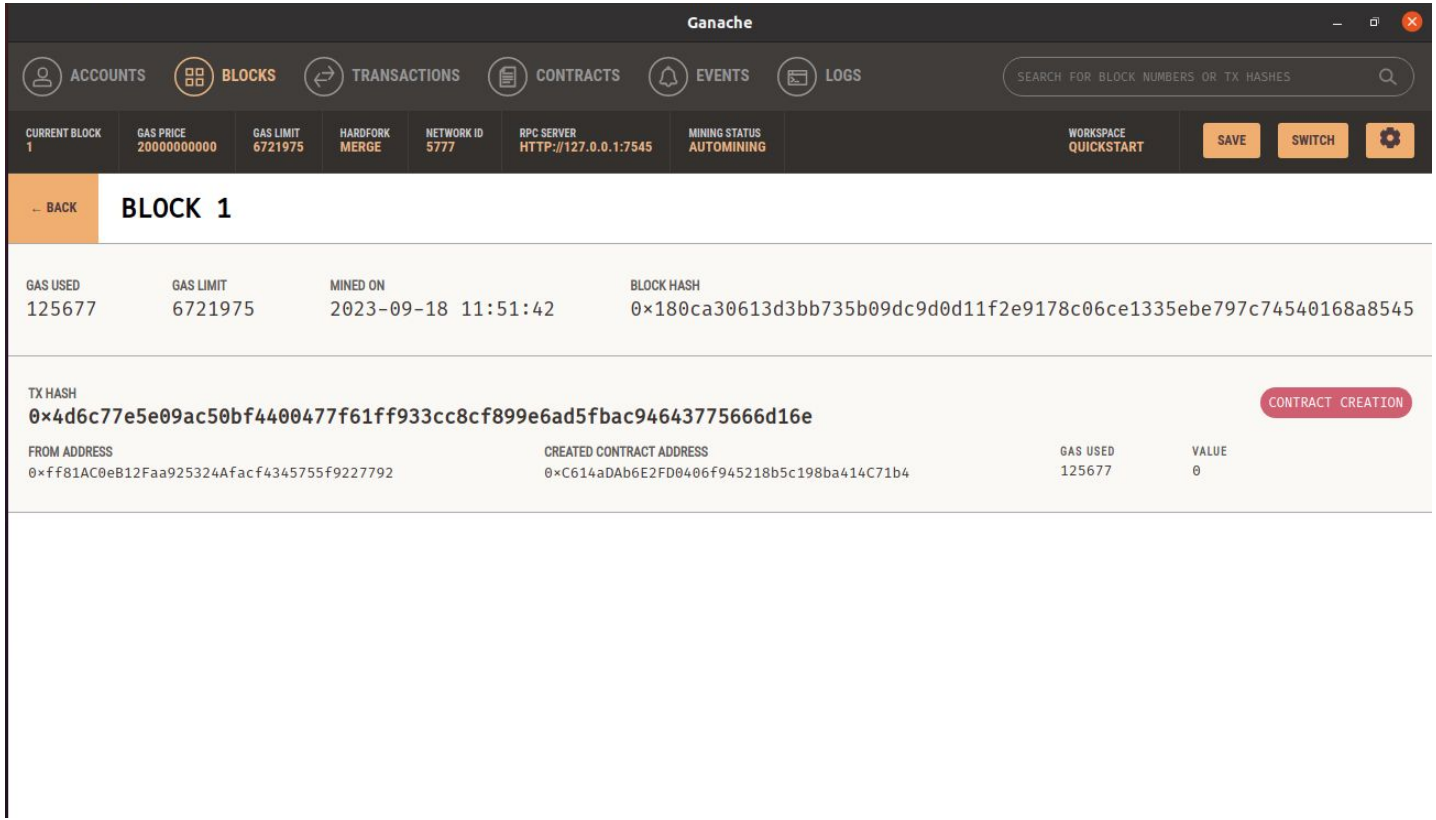
On the Ganache Environment, the Transaction count is updated



Courtesy : Github - Ganache

Introduction to Ganache for Ethereum blockchain

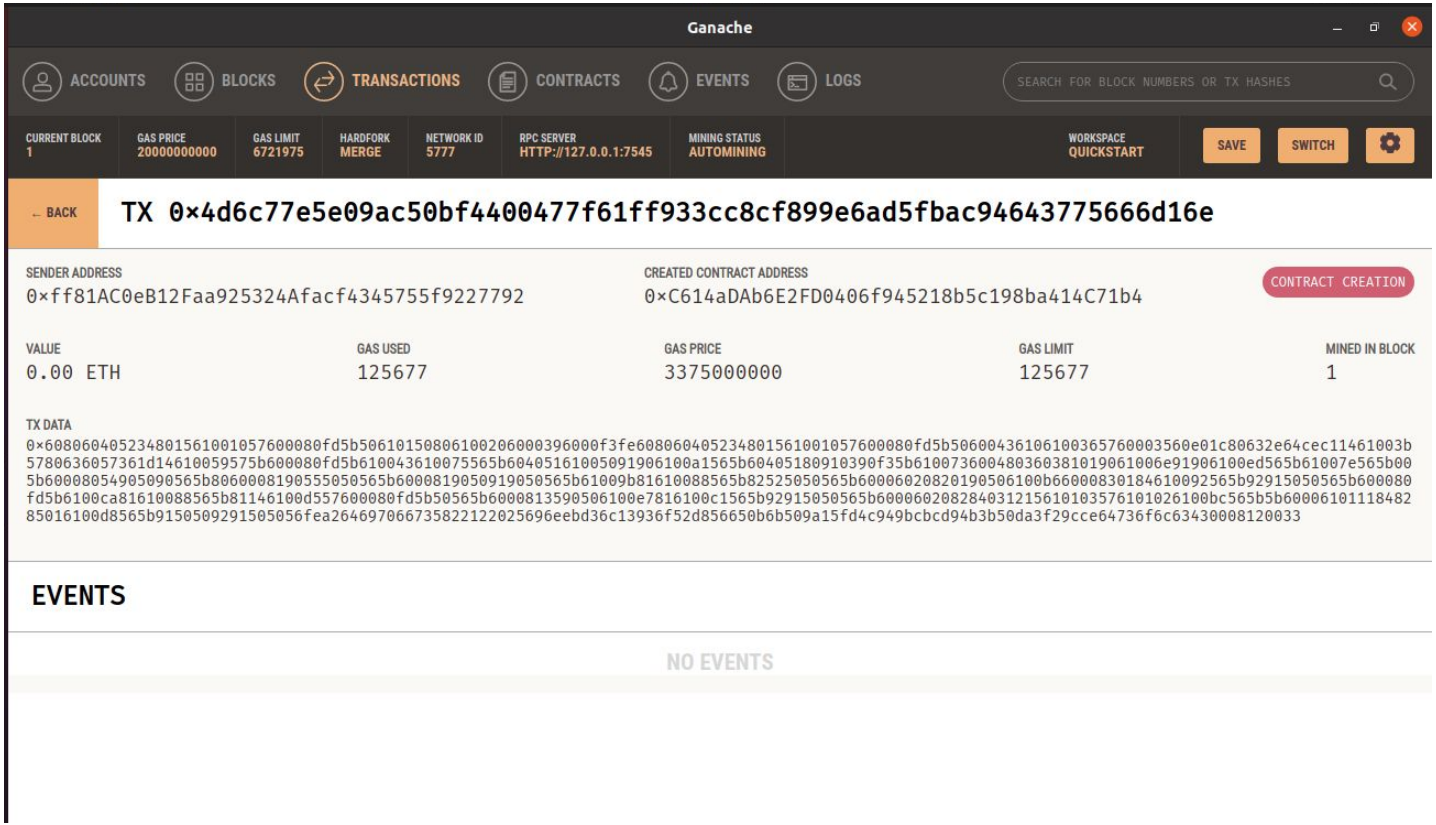
Block 1 is added to the Blockchain which displays the Contract Created



Courtesy : Github - Ganache

Introduction to Ganache for Ethereum blockchain

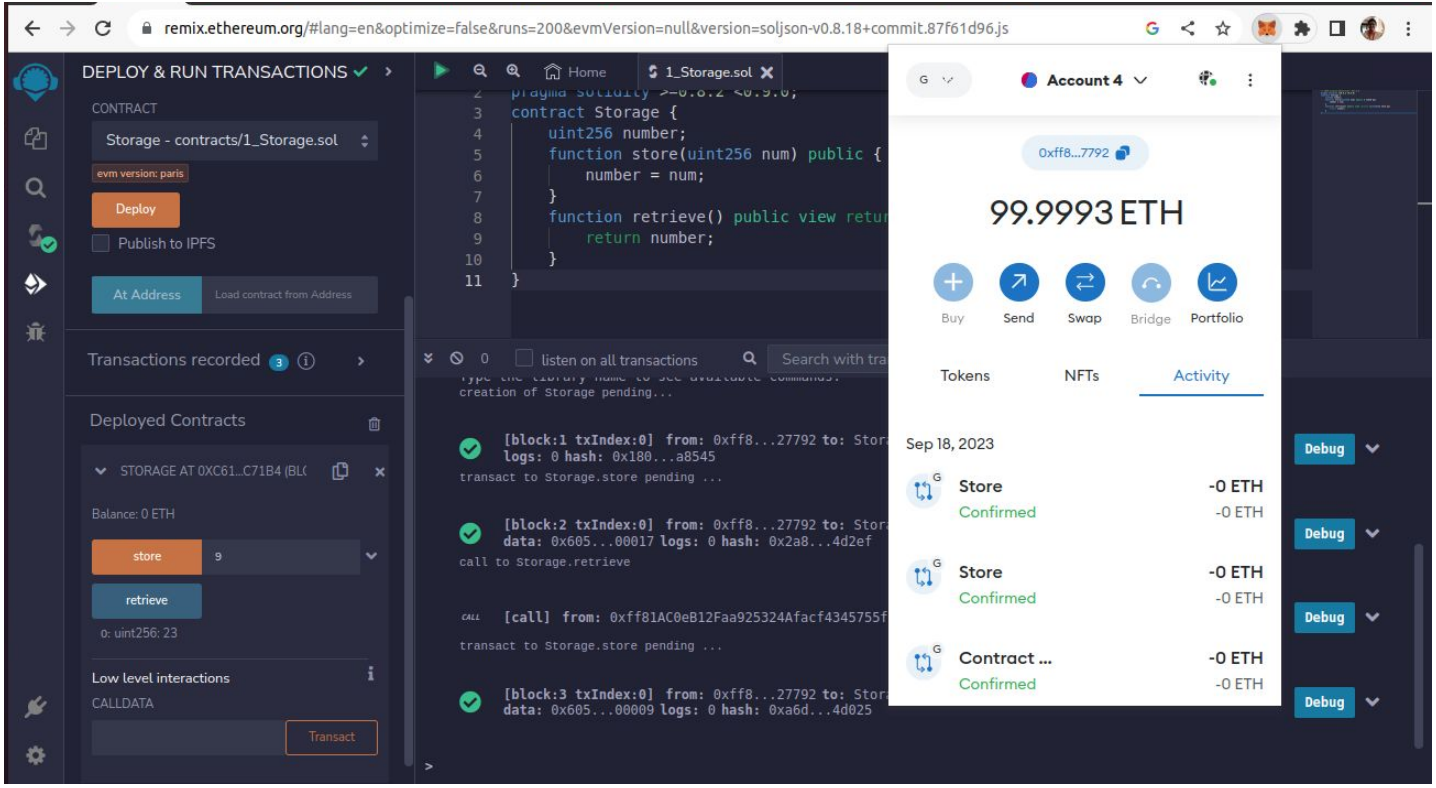
Transaction details of the Contract is displayed on Ganache Environment



Courtesy : Github - Ganache

Introduction to Ganache for Ethereum blockchain

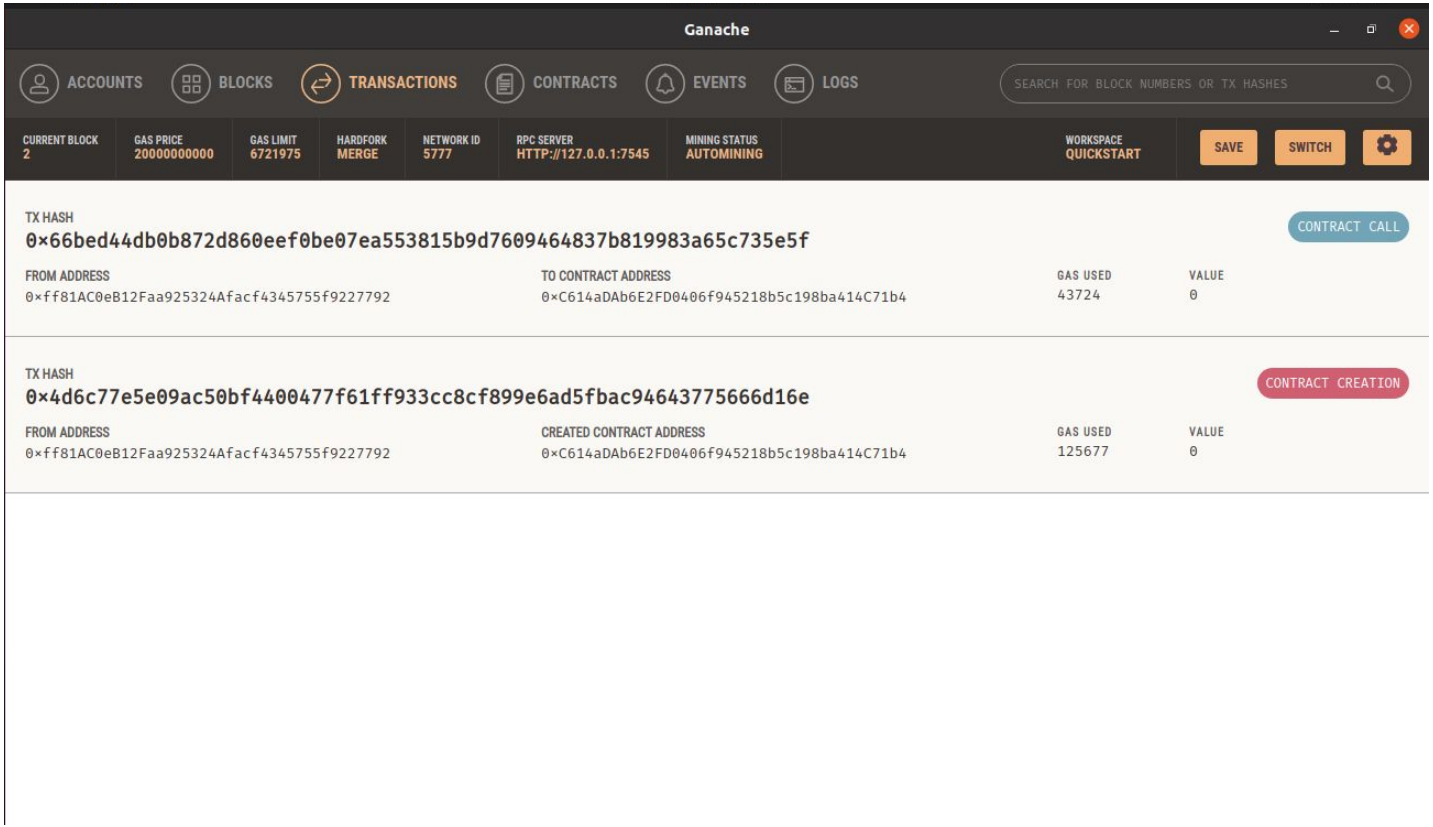
After interacting with the Smart Contract, funds are updated on the Metamask



Courtesy : Github - Ganache

Introduction to Ganache for Ethereum blockchain

On the Ganache, the Contract call is listed



Courtesy : Github - Ganache

Ethereum Frameworks

Courtesy : Ethereum Frameworks

Ethereum Frameworks

1. **Truffle Suite:**
   a. **Truffle:** A development environment, testing framework, and asset pipeline for Ethereum.
   b. **Ganache:** A personal blockchain for Ethereum development & used to deploy contracts, develop your applications, and run tests.
2. **Hardhat:** A development environment for Ethereum that allows to compile, deploy, test, and debug your smart contracts.
3. **Embark:** A framework for building and deploying decentralized applications (DApps) on Ethereum.
4. **Brownie:** A Python-based framework for Ethereum smart contract development, testing, and deployment.
5. **Waffle:** A testing and development framework for Ethereum smart contracts that focuses on simplicity and extensibility.
6. **OpenZeppelin:**
   a. A library for secure smart contract development.
   b. It provides reusable, secure smart contract components.

Ethereum Frameworks

7. **ethers.js:**
   - A JavaScript library for interacting with the Ethereum blockchain.
   - It simplifies the process of interacting with smart contracts and sending transactions.
8. **Web3.js:**
   - Another JavaScript library for interacting with the Ethereum blockchain.
   - It is widely used for building decentralized applications.
9. **Drizzle:**
   - A front-end library for integrating Ethereum decentralized applications with React.
10. **Remix:**
    - An open-source web and desktop application that helps developers write, test, and deploy smart contracts.
11. **Infura:**
    - A cloud infrastructure service that provides Ethereum nodes as a service.
    - It allows developers to connect to the Ethereum network without running their own nodes.
12. **ENS (Ethereum Name Service):**
    - A decentralized domain name system built on the Ethereum blockchain.
    - It simplifies the process of associating human-readable names with Ethereum addresses.

Ethereum Frameworks

1. **Truffle Suite:**
   - **Features:**
     - Development environment, testing framework, and asset pipeline for Ethereum.
     - Built-in smart contract compilation, linking, deployment, and binary management.
     - TestRPC for local testing and debugging.
   - **Applications:**
     - Rapid smart contract development and testing.
     - Automated testing of Ethereum contracts.
   - **Pros:**
     - Comprehensive development suite.
     - Large community support.
   - **Cons:**
     - Steeper learning curve for beginners.

Courtesy : Ethereum Frameworks

Ethereum Frameworks

2. **Embark:**
   - **Features:**
     - Framework for building and deploying decentralized applications (dApps).
     - Smart contract development, deployment, and testing.
     - Integrated web server for easy dApp development.
   - **Applications:**
     - Full-stack decentralized application development.
     - Simplified deployment process.
   - **Pros:**
     - Easy to use for both beginners and experienced developers.
     - Built-in web server for streamlined development.
   - **Cons:**
     - Smaller community compared to Truffle.

Courtesy : Ethereum Frameworks

Ethereum Frameworks

3. **Hardhat:**
   - **Features:**
     - Ethereum development environment with task automation.
     - Advanced development environment with support for TypeScript.
     - Extensive plugin system for additional functionality.
   - **Applications:**
     - Ethereum smart contract development.
     - Task automation and extensibility.
   - **Pros:**
     - Solid TypeScript support.
     - Flexible and extensible through plugins.
   - **Cons:**
     - May be less beginner-friendly.

Courtesy : Ethereum Frameworks

Ethereum Frameworks

4. **Brownie:**
   - **Features:**
     - Python-based framework for Ethereum development.
     - Simple and Pythonic syntax for smart contract development.
     - Built-in testing and deployment tools.
   - **Applications:**
     - Ethereum smart contract development using Python.
     - Rapid prototyping and testing.
   - **Pros:**
     - Python-friendly syntax.
     - Easy to learn for Python developers.
   - **Cons:**
     - Smaller community compared to more established frameworks.

Courtesy : Ethereum Frameworks

Ethereum Frameworks

5. **OpenZeppelin:**
   - **Features:**
     - Open-source framework for building secure smart contracts.
     - Provides reusable, community-audited smart contract components.
     - Follows best practices for secure contract development.
   - **Applications:**
     - Building secure and audited smart contracts.
     - Reusing tested and secure components.
   - **Pros:**
     - Emphasis on security.
     - Large library of reusable smart contract components.
   - **Cons:**
     - Not a complete development suite; often used in conjunction with other frameworks.

Courtesy : Ethereum Frameworks

Ethereum 2.0 / ETH 2.0 / Serenity - Key features

- aims to address scalability, security, and sustainability issues.
- **Beacon Chain:**
  - first phase of Ethereum 2.0 and was launched in December 2020.
  - Introduces a PoS consensus mechanism
    - where validators are chosen to create new blocks and validate transactions based on the amount of cryptocurrency they "stake" as collateral.
- **Shard Chains:**
  - will run in parallel to the existing Ethereum mainnet.
  - aim to improve scalability by processing transactions and smart contracts in smaller groups, or shards, rather than on a single chain.

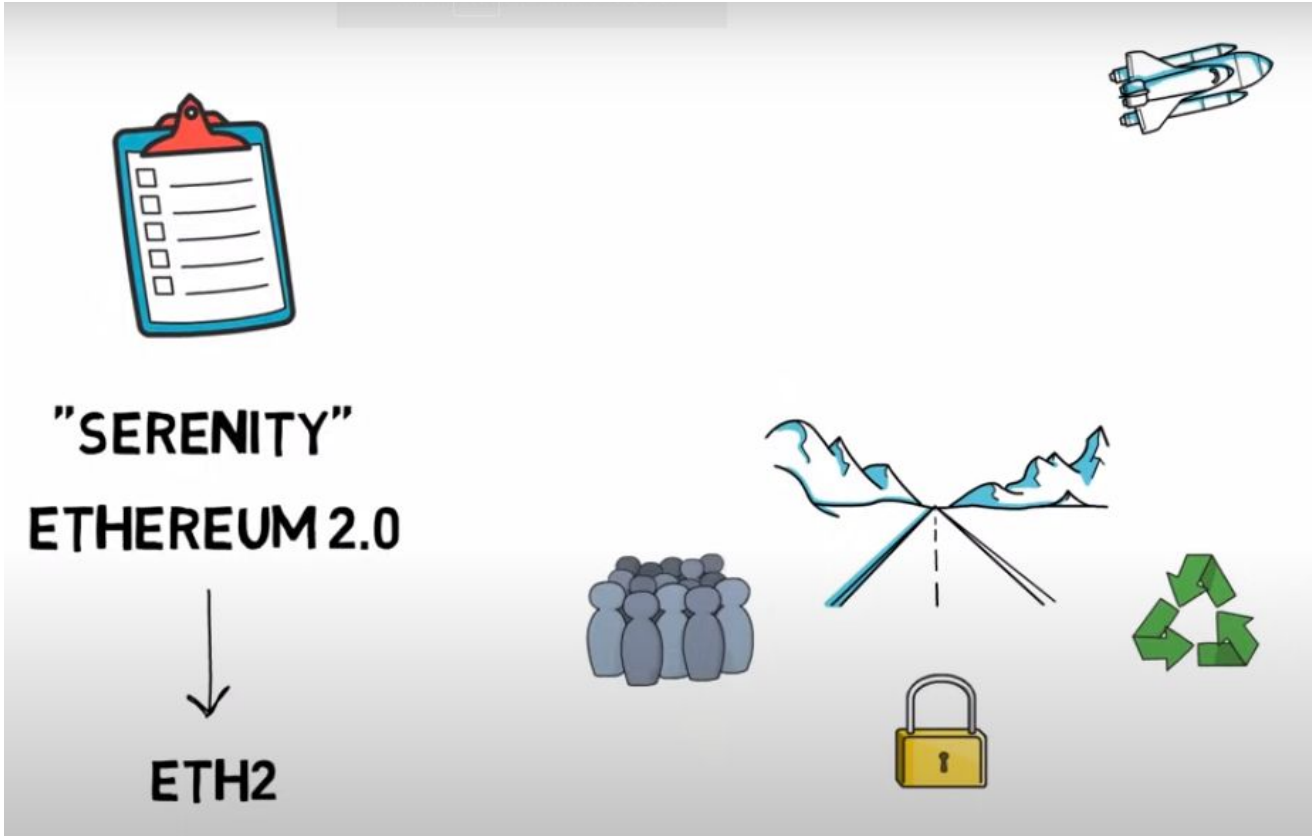
Ethereum 2.0 / ETH 2.0 / Serenity - Key features

- **eWASM (Ethereum WebAssembly):**
  - replace the current Ethereum Virtual Machine (EVM) with eWASM.
  - expected to bring improved efficiency, faster execution of smart contracts, and greater flexibility for developers.
- **Crosslinks:**
  - connect shard chains to the Beacon Chain,
  - ensuring communication and data transfer between different shards.
  - ***Data Availability:*** Helps maintain the security of the network by providing a mechanism for validators to agree on the state of different shard chains.
- **Migration from PoW to PoS:**
  - The transition from PoW to PoS will occur in multiple phases.
  - PoS is more energy-efficient and allows users to earn rewards by locking up cryptocurrency as collateral, contributing to network security.

Ethereum 2.0 / ETH 2.0 / Serenity - Key features

- **Docking and Withdrawal:**
  - Allows Ethereum users to "dock" their existing ETH into the Ethereum 2.0 system, ***converting it into a special token called "BETH."***
  - Users can withdraw their ETH from the Ethereum 2.0 system back to the Ethereum 1.0 network.
- **Timeline:**
  - ***Ongoing Development:*** Ethereum 2.0 is a complex and multi-phase upgrade, and development is ongoing. Different phases will be rolled out incrementally as they are ready.
  - ***Full Deployment:*** Full deployment of Ethereum 2.0 is expected to take several years, with continuous updates and improvements.
- **Community and Research:**
  - open source, with contributions from a large community of developers and researchers.
  - Research-Oriented: The upgrade is the result of extensive research and collaboration within the Ethereum community.

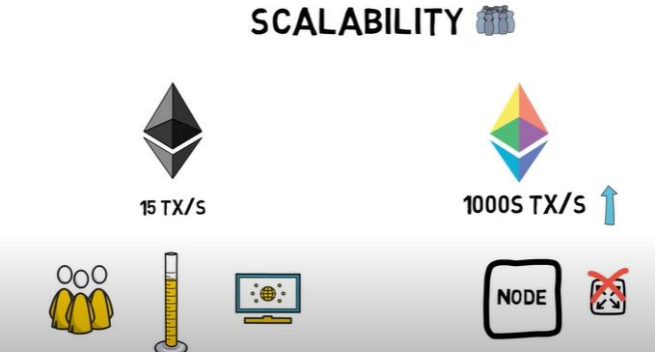
Ethereum 2.0 / ETH 2.0 / Serenity - Objectives

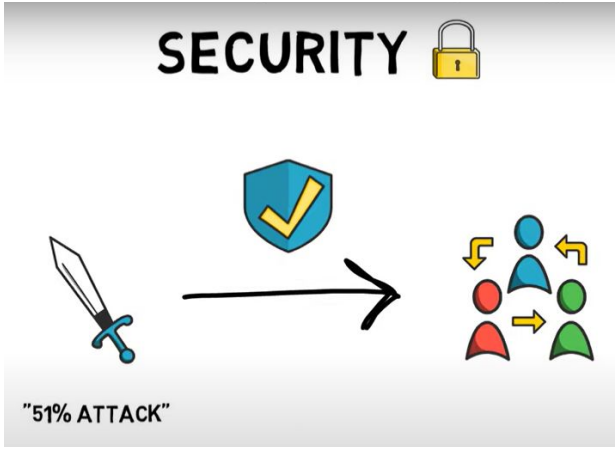


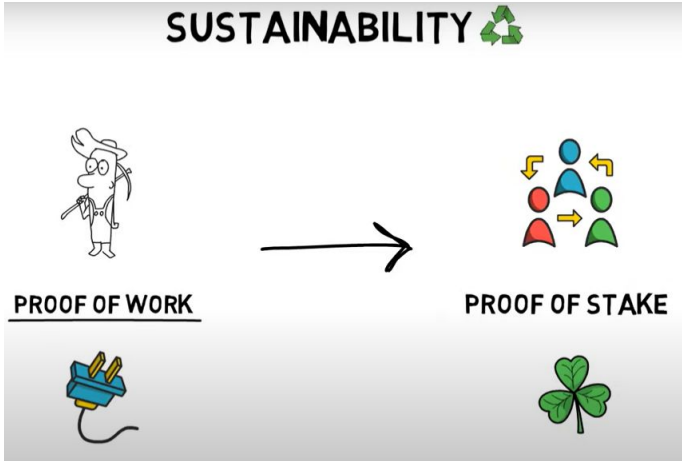
Scalability
Security
Eco-friendly

Courtesy : Ethereum 2.0

Ethereum 2.0 / ETH 2.0 / Serenity - Objectives

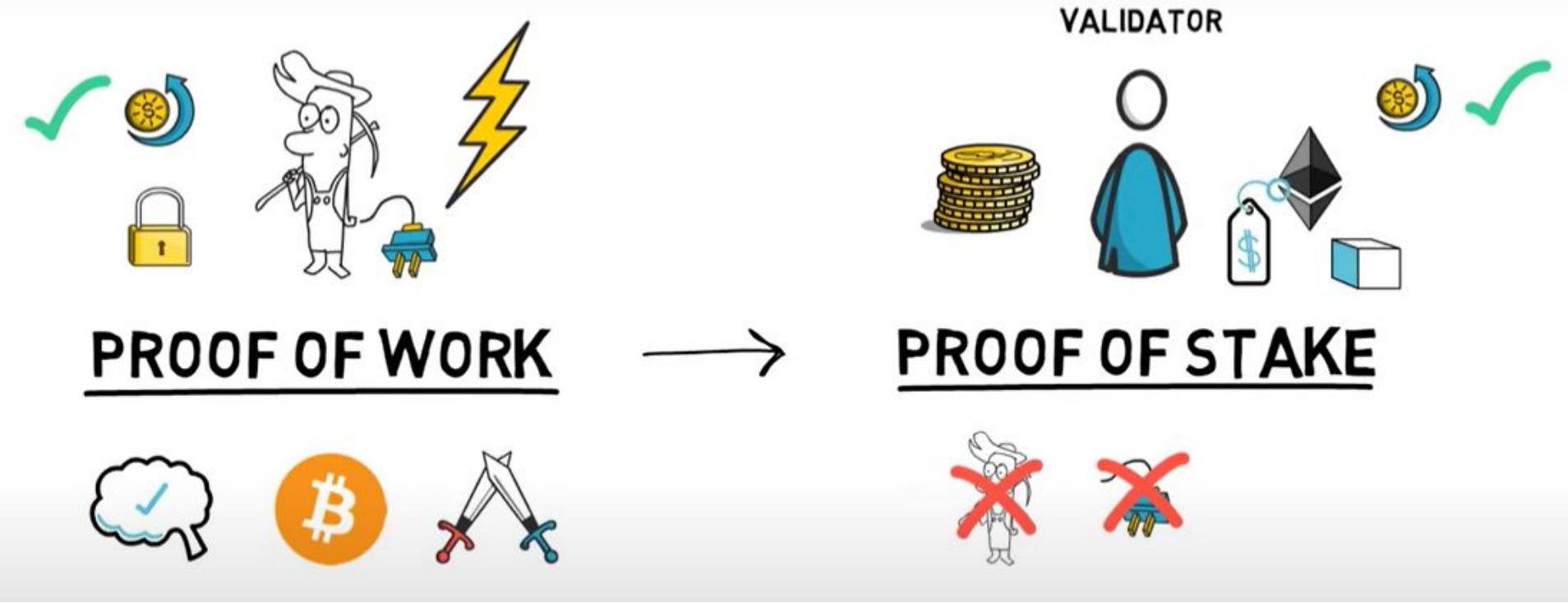






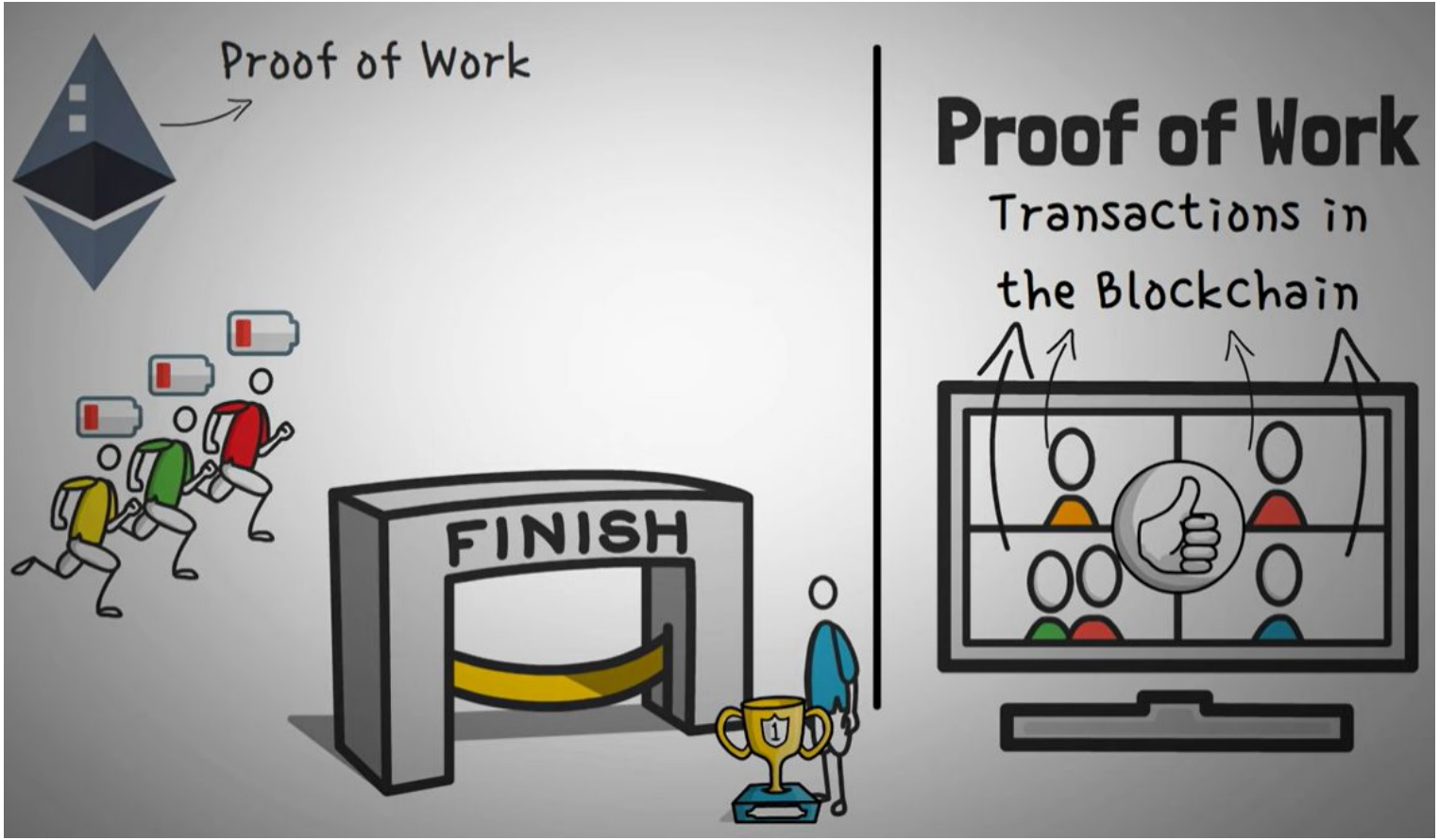
Courtesy : Ethereum 2.0

Ethereum 2.0 / ETH 2.0 / Serenity - Key features : PoS



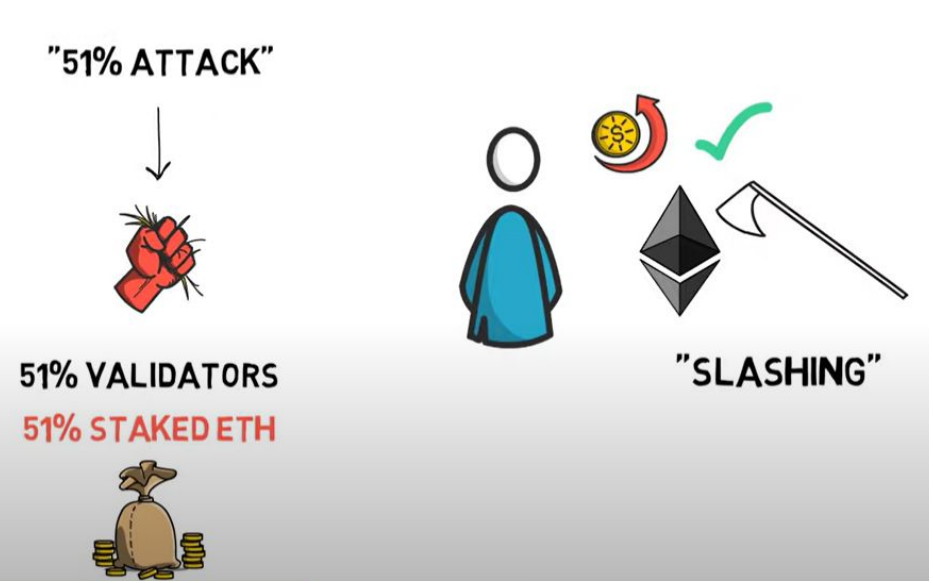
Courtesy : Ethereum 2.0

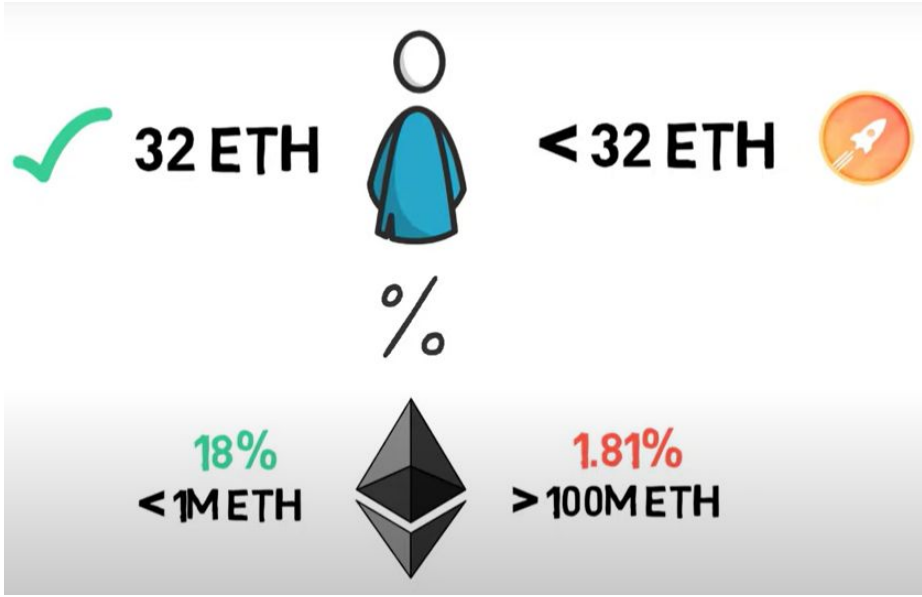
Ethereum 2.0 / ETH 2.0 / Serenity - Key features : PoS



Courtesy : Ethereum 2.0

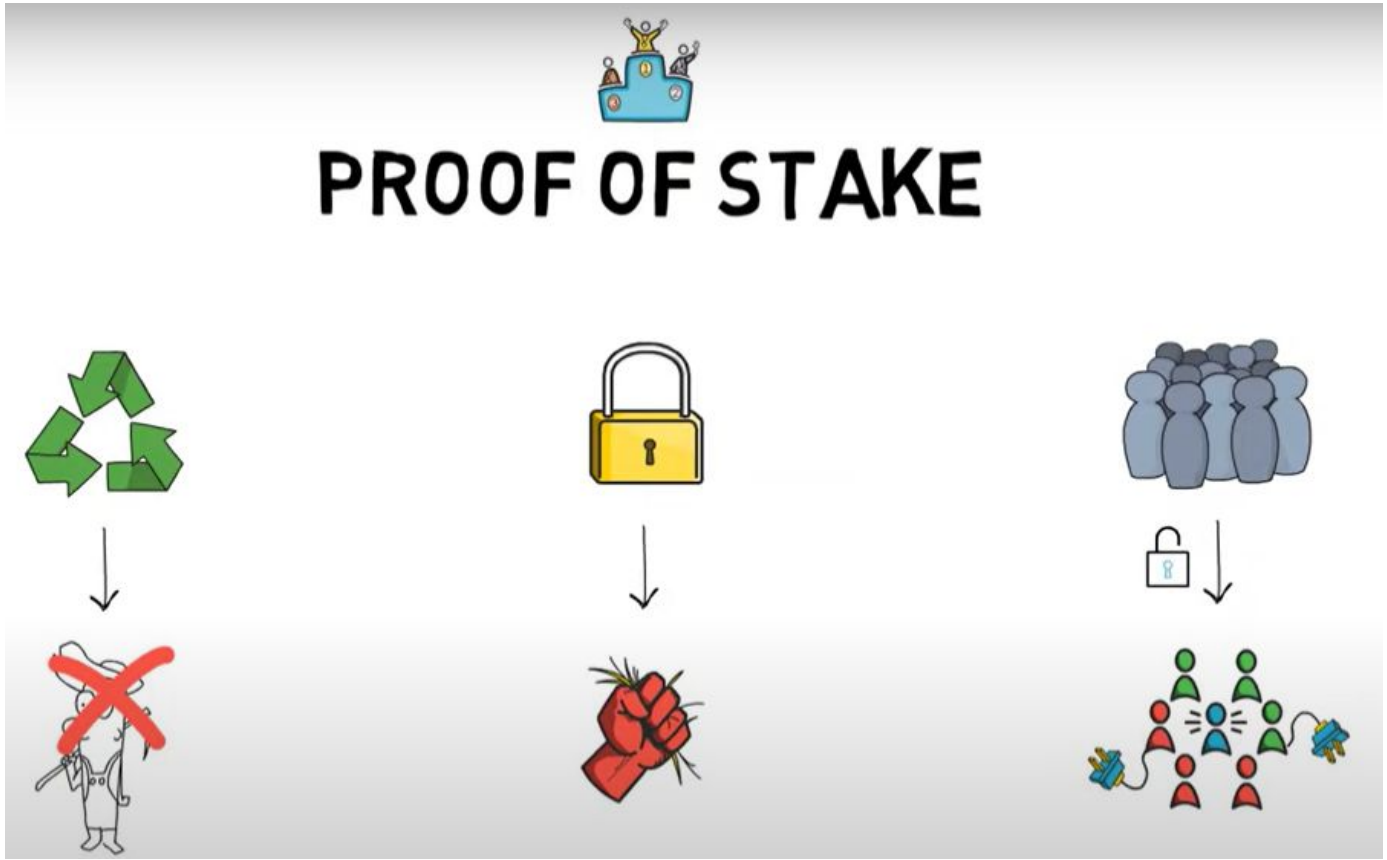
Ethereum 2.0 / ETH 2.0 / Serenity - Key features : PoS





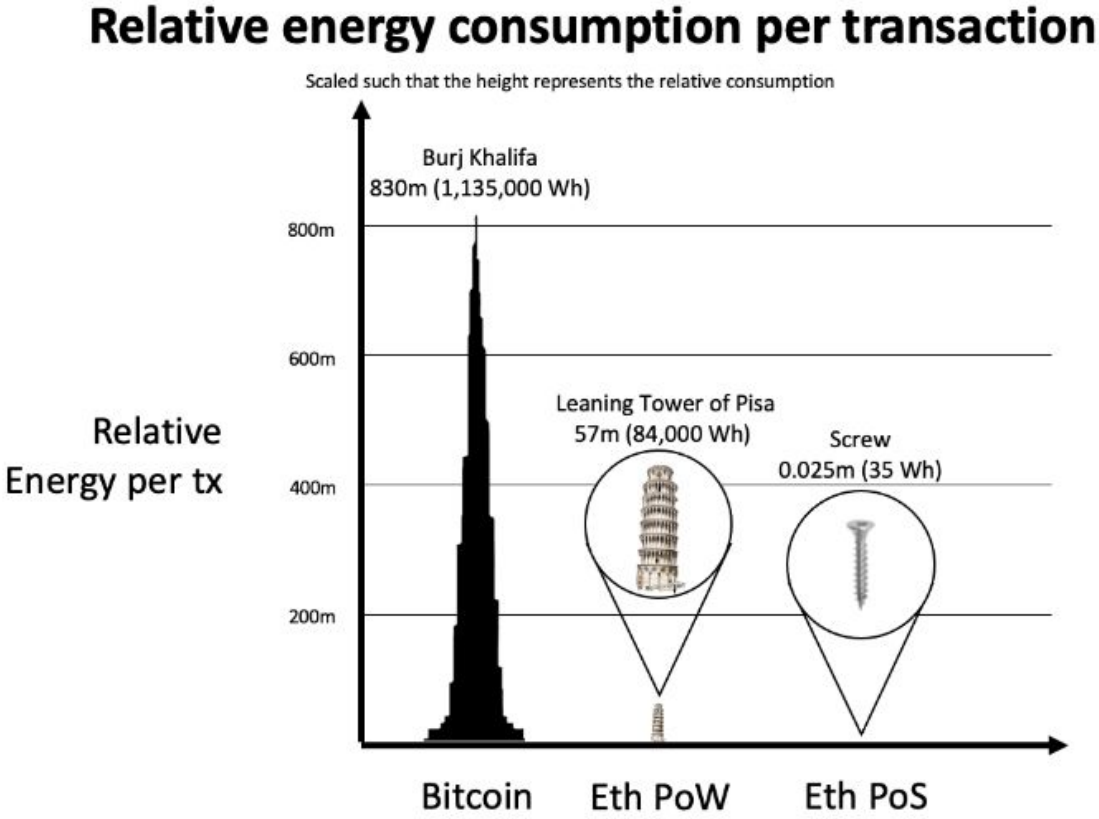
Courtesy : Ethereum 2.0

Ethereum 2.0 / ETH 2.0 / Serenity - Key features : PoS



Courtesy : Ethereum 2.0

Ethereum 2.0 / ETH 2.0 / Serenity - Key features : PoS



Courtesy : Working of Ethereum 2.0

Ethereum 2.0 / ETH 2.0 / Serenity - Key features : PoS

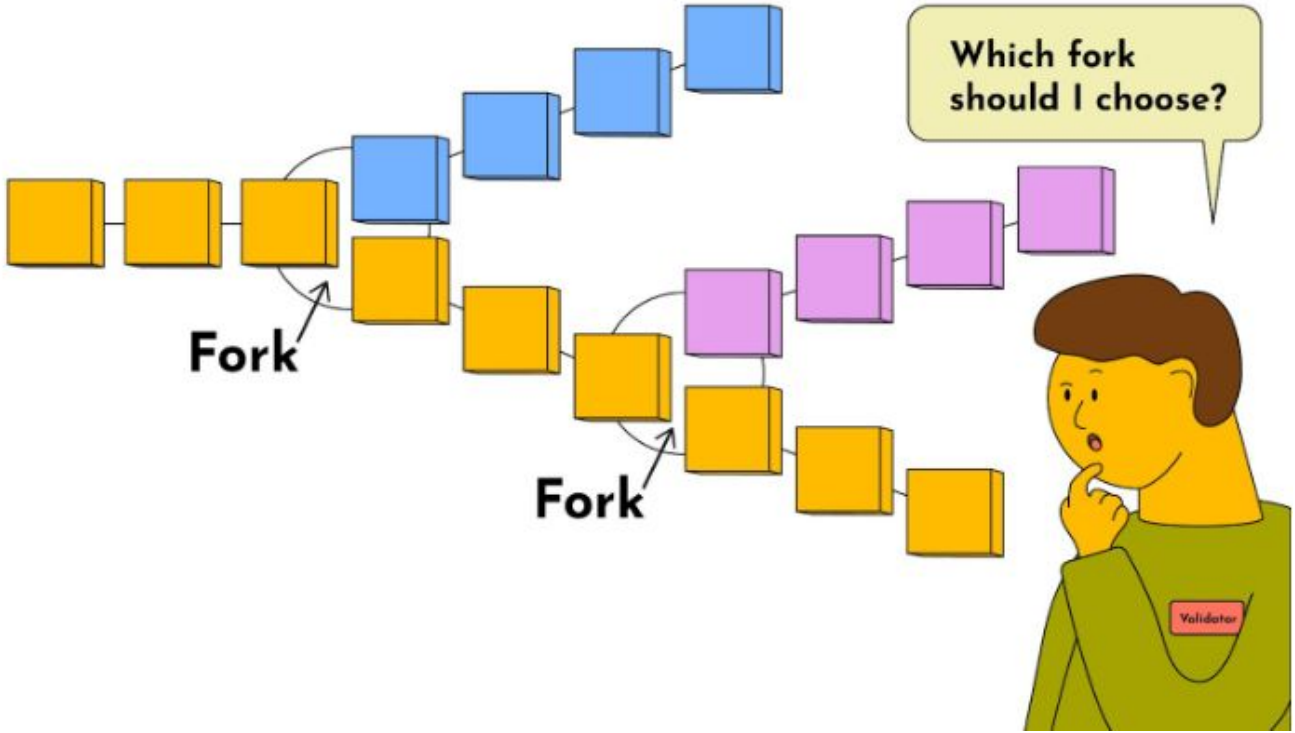
Proof-of-Work hash,

- each "vote" is backed by hardware units.

Proof-of-Stake,

- uses money as a Sybil resistance mechanism.
- each vote is backed by money.

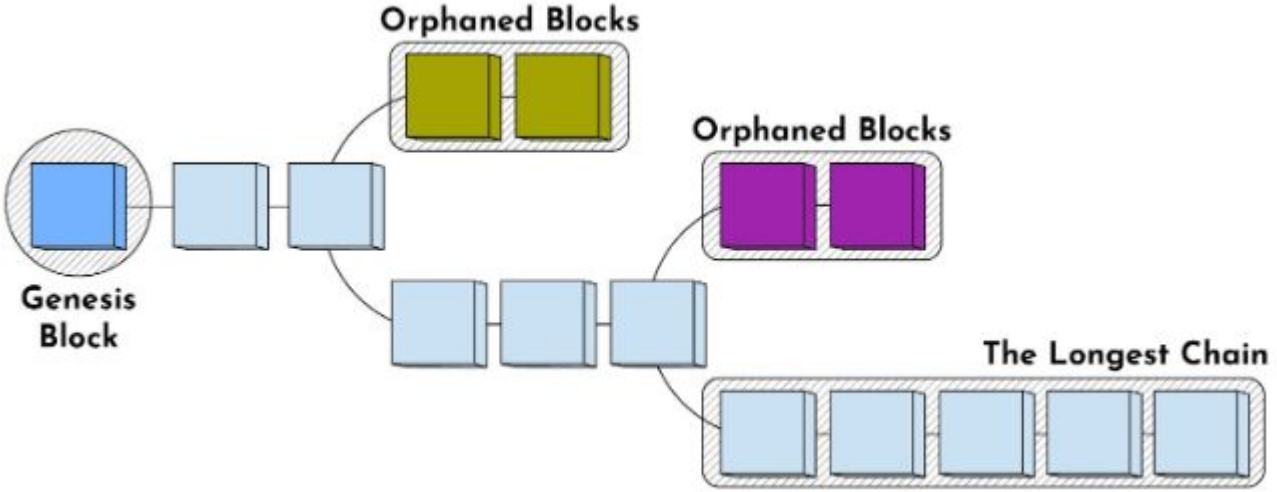
Fork-choice rule is how a consensus algorithm chooses the "correct" chain.



Courtesy : Working of Ethereum 2.0

Ethereum 2.0 / ETH 2.0 / Serenity - Key features : PoS

Bitcoin uses the "longest chain" rule, which means that whichever blockchain is the longest will be the one the rest of the nodes accept as the valid chain. The longest chain is determined by the cumulative Proof-of-Work difficulty on that chain.



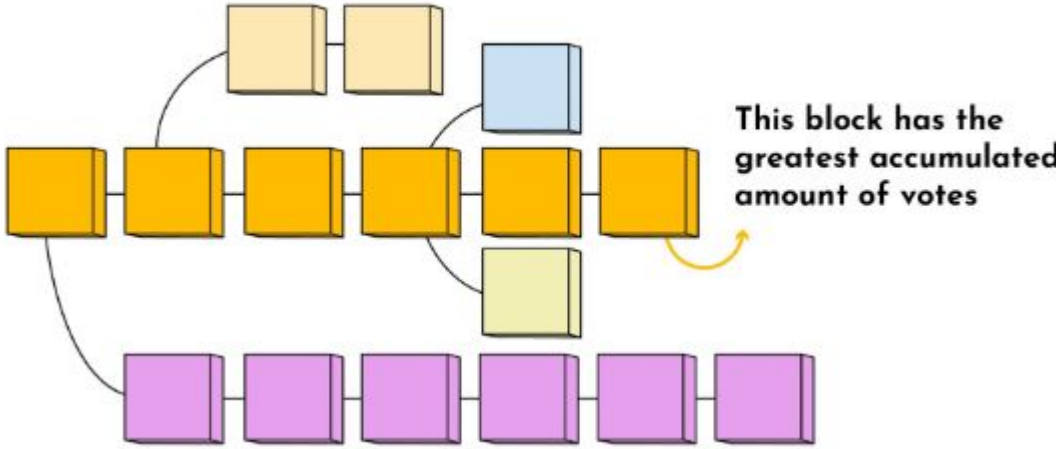
Courtesy : Working of Ethereum 2.0

Ethereum 2.0 / ETH 2.0 / Serenity - Key features : PoS

Proof-of-Stake,

- Ethereum now looks at the “weight” of the chain, which is determined by the accumulated sum of validator votes.

In Gasper, the fork-choice rule is called "LMD-GHOST," which stands for "latest message-driven greedy heaviest observed sub-tree." This means we select the fork with the greatest accumulated weight of attestations to be the canonical one.

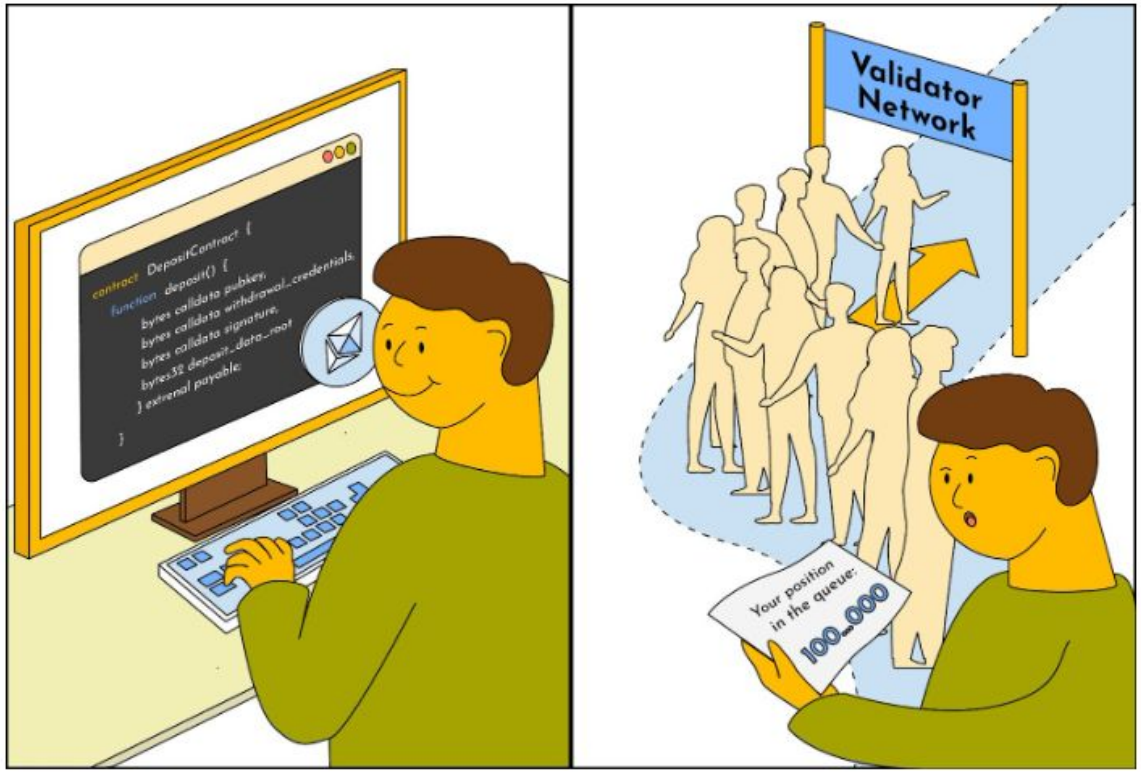


Courtesy : Working of Ethereum 2.0

Ethereum 2.0 / ETH 2.0 / Serenity - Key features : PoS

"Validators" are the users that are responsible for voting on which block to add next. Anyone can become a validator by depositing 32 ETH into the "deposit contract" (which is a special contract designated to collect and track validators) and then running two separate pieces of software: an execution client and a consensus client.

After depositing their ETH into the deposit contract, the user joins an "activation queue" where they are basically in line to become a validator.



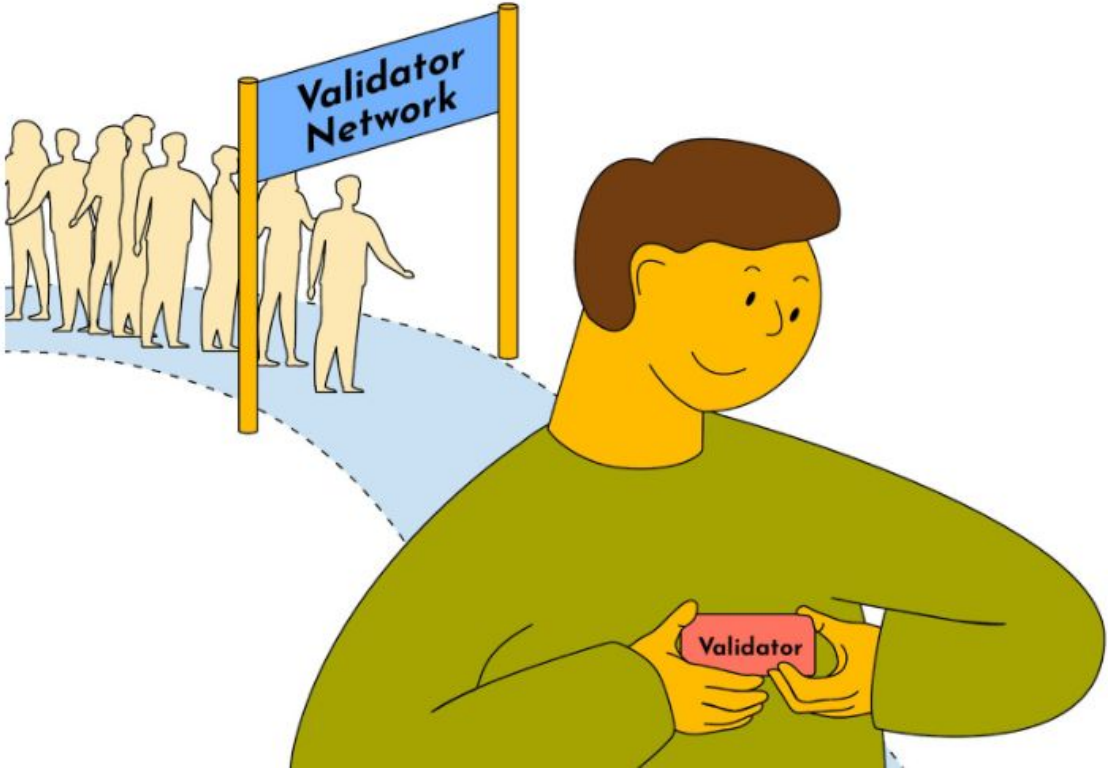
Courtesy : Working of Ethereum 2.0

Ethereum 2.0 / ETH 2.0 / Serenity - Key features : PoS

This is done to limit the rate of new validators entering the network at any one time. The goal of rate-limiting entry and exit is two-fold:

1. The communication overhead of validators casting their votes is very high, and there's a limit to how many validators the protocol can handle while ensuring votes are propagated on time.
2. To prevent a large number of malicious validators from performing some malicious action and then leaving the network to escape penalties.

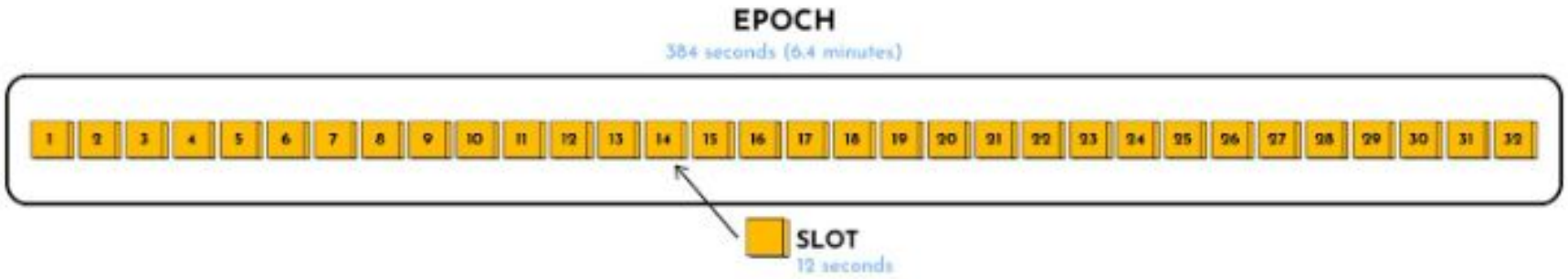
Once a validator is activated, the validator's responsibilities begin.



Courtesy : Working of Ethereum 2.0

Ethereum 2.0 / ETH 2.0 / Serenity - Key features : PoS

Every consensus algorithm needs a sense of time to order events in the network, such as votes. For Gasper, the algorithm divides time into "**slots**" and "**epochs**." **Slots are 12 seconds long**, and **each epoch consists of 32 slots** (thus 6.4 minutes).

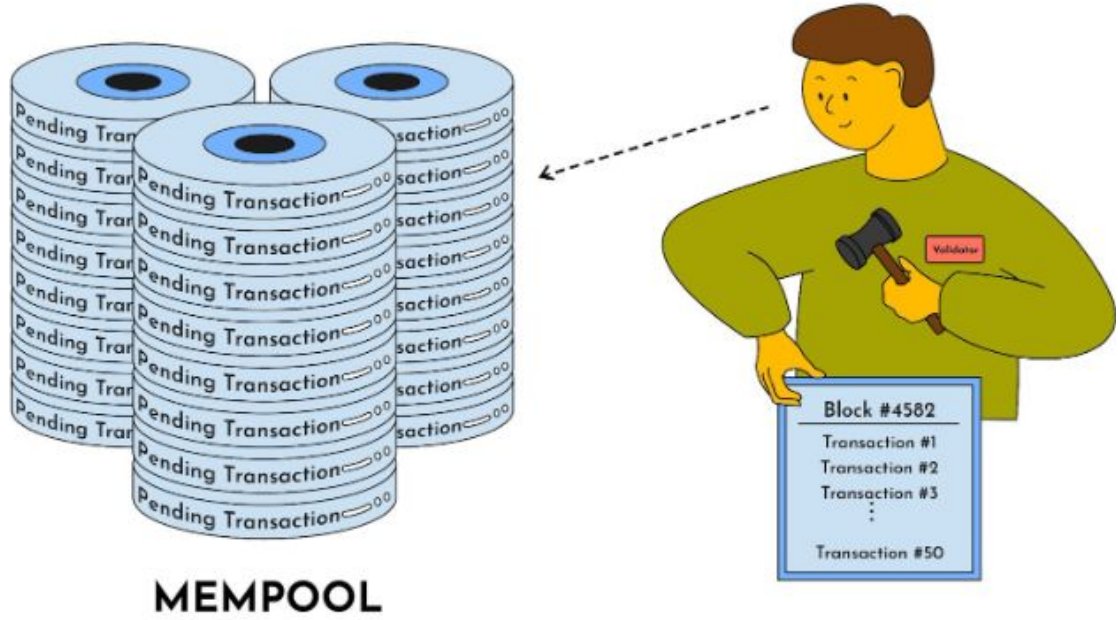


Note: Although not used as much, a period of 2048 epochs (~9.1 days) is called an eek ("Ethereum week"). Operations that take a long time can be measured in eeks.

Courtesy : Working of Ethereum 2.0

Ethereum 2.0 / ETH 2.0 / Serenity - Key features : PoS

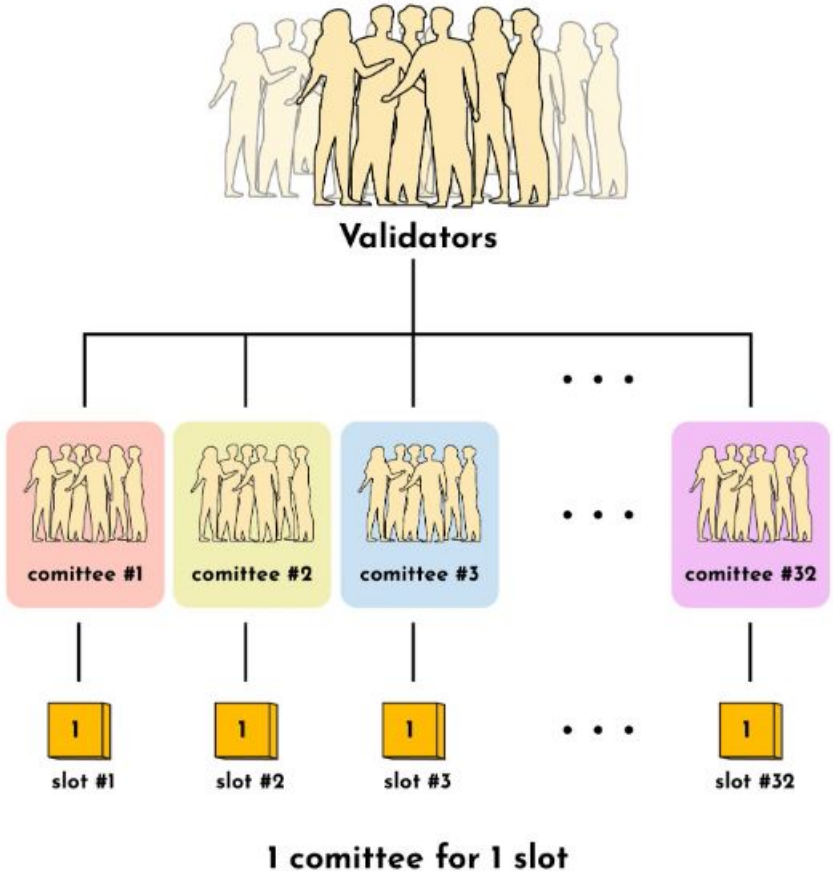
For every slot, one validator is randomly selected to be a "block proposer." The block proposer validator is responsible for constructing a new block out of pending transactions. It then sends the block out to other validators in the network, which vote on whether that block is valid.



Courtesy : Working of Ethereum 2.0

Ethereum 2.0 / ETH 2.0 / Serenity - Key features : PoS

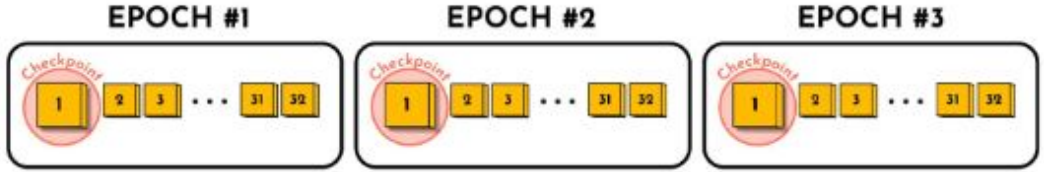
Not every validator votes for every block, though. Instead, for each epoch, a validator is assigned to a "committee." Each committee is randomly assigned one slot where they need to attest to determine whether the proposed block is valid. Committees are spread among the 32 slots in an epoch, which means 1/32 of the total validator set attests to the validity of each slot or block, with the maximum allowed validators per committee being 2048. In other words, each validator only attests one block per epoch for the slot their committee was assigned to.



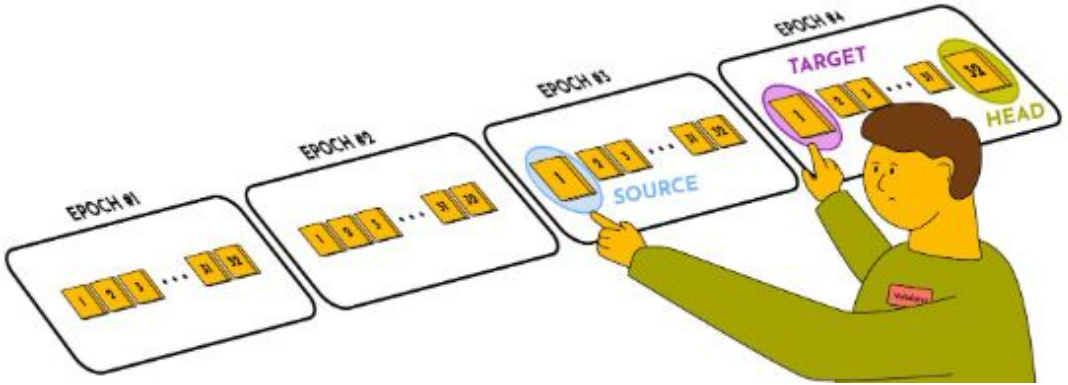
Courtesy : Working of Ethereum 2.0

Ethereum 2.0 / ETH 2.0 / Serenity - Key features : PoS

Note that a small set of validators are also randomly chosen to join sync committees, which are different from the committees mentioned above. Being in a sync committee requires validators to help light clients sync up and determine the head of the chain, which they earn additional rewards for. However, as a validator, you are only part of a sync committee once every ~22 months, so it's not a responsibility carried on a daily basis. Validators don't only attest to the current head block. They also attest to two others called "checkpoint" blocks. Every epoch has one checkpoint block that identifies the latest block at the start of that epoch.



Validators attest to their view of two checkpoints every epoch: the "source" and "target" blocks.



Courtesy : Working of Ethereum 2.0

Ethereum 2.0 / ETH 2.0 / Serenity - Key features : PoS

In addition to the head, source, and target blocks, validators also include the following information in their attestation:

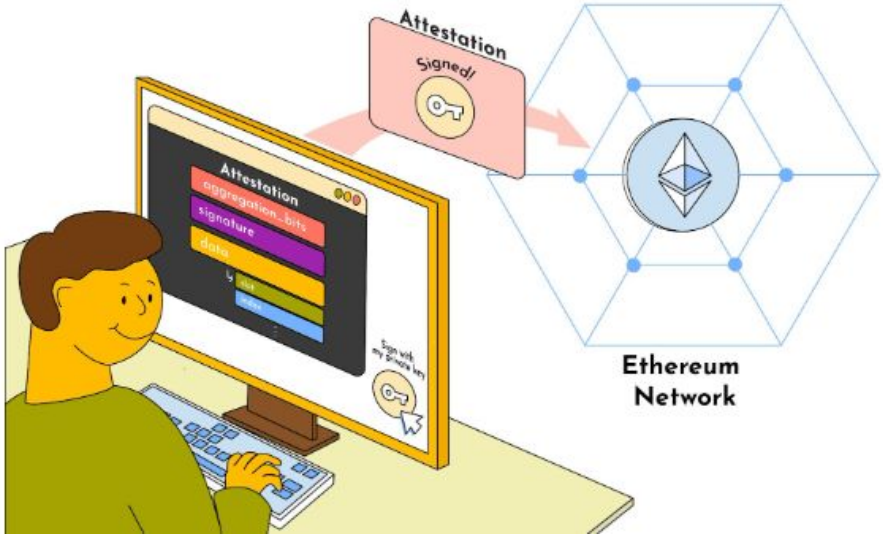
- aggregation_bits: a bitlist of validators where the position maps to the validator index in their committee; the value (0/1) indicates whether the validator signed the data (i.e., whether they are active and agree with the block proposer)
- data: details relating to the attestation (defined below)
- signature: a BLS signature that aggregates the signatures of individual validators

The data contains the following information:

- slot: the slot number that the attestation refers to
- index: the committee # that the validator belongs to in a given slot
- beacon_block_root: root hash of the block the validator sees at the head of the chain
- source: what the validators see as the most recent justified block
- target: what the validators see as the first block in the current epoch

Ethereum 2.0 / ETH 2.0 / Serenity - Key features : PoS

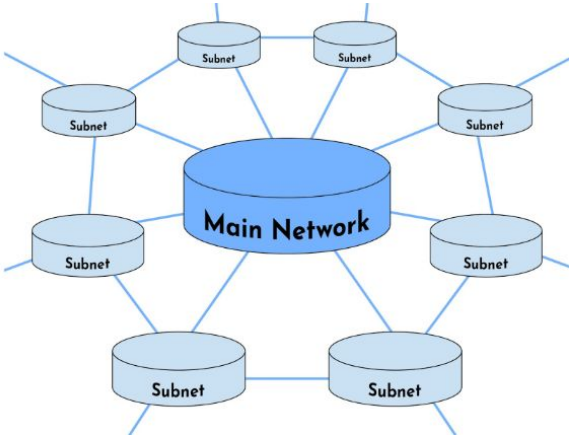
Once the data is built, the validator can flip the bit in aggregation_bits corresponding to their own validator index from 0 to 1 to show that they participated in the vote.
Finally, the validator signs the attestation using a private key and broadcasts it to the network.
But wouldn't passing attestation data created by every validator to every other validator in the network create a lot of overhead? Yes. That's why, instead of having every validator listen to every other validator, attestations from individual validators are aggregated within "subnets" before they're broadcasted.
If you divide a network into smaller, individual networks, those networks become subnets. It's just a network in a network. Subnets make communication much more efficient by collecting and shortening the route that data has to travel to get to its destination.

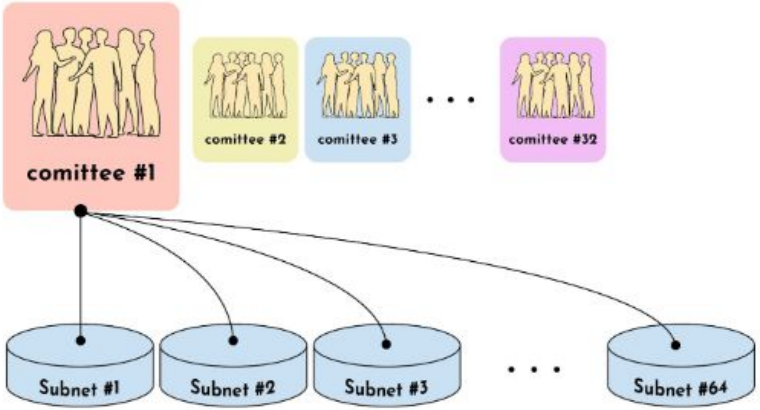


Courtesy : Working of Ethereum 2.0

Ethereum 2.0 / ETH 2.0 / Serenity - Key features : PoS

Remember when we learned that validators are divided into committees? Each of those committees is further divided into **64 subnets**. This lets aggregation happen.





Courtesy : Working of Ethereum 2.0

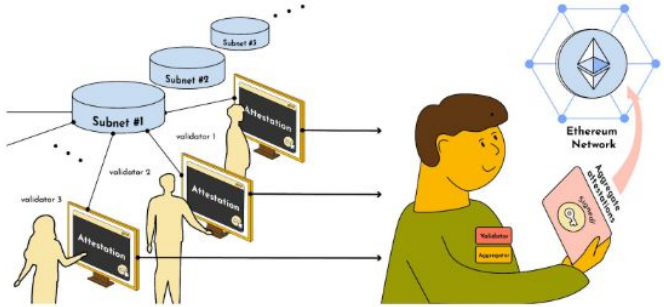
Ethereum 2.0 / ETH 2.0 / Serenity - Key features : PoS

In each epoch, a validator in each subnet is selected to be the aggregator. The aggregator collects all the attestations that have equivalent data to its own. The sender of each matching attestation is recorded in the aggregation_bits. That means a single signature is formed by combining the signatures of all validators that agree with the aggregator's data. The aggregator then broadcasts the aggregate attestations to the broader network.

When a validator is selected to be a block proposer, they package aggregate attestations from the subnets up to the latest slot in the new block.
So, in summary, the responsibilities of a validator include the following:

- Attesting on a block (once per epoch)
- Aggregating attestations from validators in the same committee (occasionally when selected to be an aggregator)
- Creating blocks when selected (infrequently when selected to be a validator)

As a result of these attestations, the blockchain comes to a consensus.

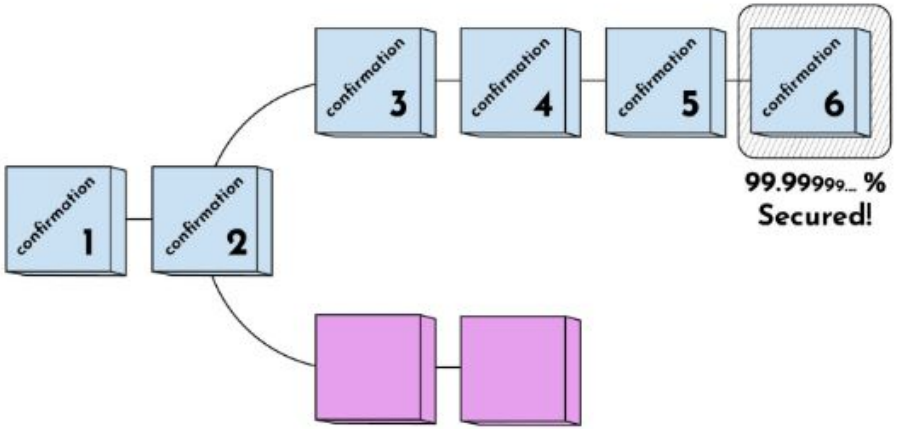


Courtesy : Working of Ethereum 2.0

Ethereum 2.0 / ETH 2.0 / Serenity - Key features : PoS

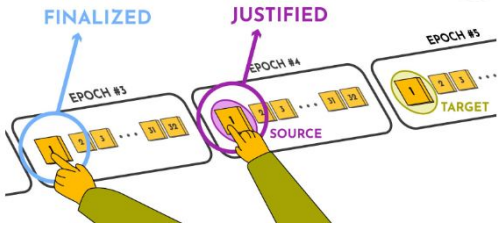
A quick recap: validators vote on blocks that a block proposer creates — if they believe it's a valid block. But at what point do we determine that a block should be included in the chain? This is what we mean when we talk about "finality." If a block reaches "finality," it cannot be reverted unless there has been a critical consensus failure.

In Bitcoin, we know a block is "final" by waiting for six confirmations. In other words, if six blocks get built on top of the current block, we assume the current block is finalized. The probability of a block being reorged after six confirmations is so small that it’s statistically negligible.



Courtesy : Working of Ethereum 2.0

Ethereum 2.0 / ETH 2.0 / Serenity - Key features : PoS



In Gasper, we determine a finalized block (or "finality") differently. Rather than using six confirmations, we use the following logic:

- If ⅔ of the total staked ether votes in favor of that block, then it becomes "justified." Justified blocks are unlikely to be reverted, but they can be under certain conditions.
- When another block is justified on top of a justified block, it's upgraded to "finalized." Finalizing a block is a commitment to include the block in the canonical chain. It can't be reverted unless an attacker destroys millions of ether.

Blocks do not reach the "justified" and "finalized" states in every slot. Instead, they happen to the first block in every epoch, which is called a "checkpoint" block.

When a checkpoint block gets upgraded to justified, it must have a link to the previous checkpoint. That is, two-thirds of the total staked ether must vote that checkpoint B is the correct descendant of checkpoint A. As a result, the previous checkpoint block becomes finalized, and the more recent block is justified.

Courtesy : Working of Ethereum 2.0

Ethereum 2.0 / ETH 2.0 / Serenity - Key features : PoS

Therefore, an attacker cannot create an alternative finalized chain without:

- Owning or manipulating ⅔ of the total staked ether (because ⅔ of total ether is required to finalize a chain)
- Destroying at least ⅓ of the total staked ether (because if ⅔ of total ether was used to finalize two different chains, that means ⅓ of the ether voted maliciously, which would be slashed)

Now that you understand how validator votes help determine which blocks reach finality, we can move on to the fork-choice rule.

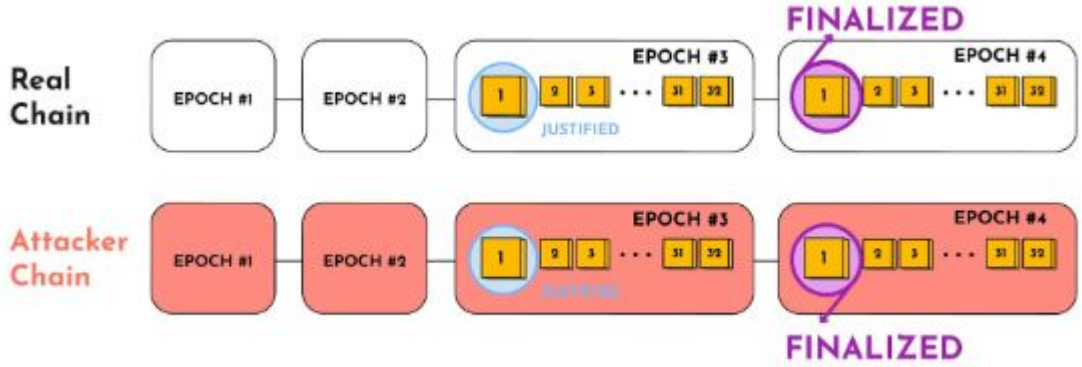
Courtesy : Working of Ethereum 2.0

Ethereum 2.0 / ETH 2.0 / Serenity - Key features : PoS

So far, we understand how blocks are voted on, how they become "finalized," and how validators know which fork to build on top of. This gets us a blockchain where a decentralized set of actors can create and agree on what block to add to the blockchain next.

But the big issue with Proof-of-Stake consensus mechanisms is that they're susceptible to "**long-range attacks**" — that is, to a group of attackers can build a fork from genesis, hide it from users for a while, and then suddenly publish it. This could lead to **two different chains with "finalized" blocks** and end users wouldn't know which is the correct chain.

This wouldn't be possible in the Proof-of-Work protocol because it would require an insurmountable amount of hash power to generate all of the previous blocks and outpace the main chain. But in Proof-of-Stake, there's no such thing as hash power since votes are just digital representations of data, so it's entirely possible for a subset of validators to create a new blockchain that looks and acts like the main chain and deceive others into believing in this alternative blockchain.



Courtesy : Working of Ethereum 2.0

4. Weak Subjectivity

There are certain checkpoints where every node in every network agrees that a block (and each that came before it) belongs in the canonical chain. This is called a "**weak subjectivity checkpoint.**" If a node sees a block that conflicts with a weak subjectivity checkpoint, then it rejects that block. The latest weak subjectivity checkpoint is like a new genesis block for the entire network.

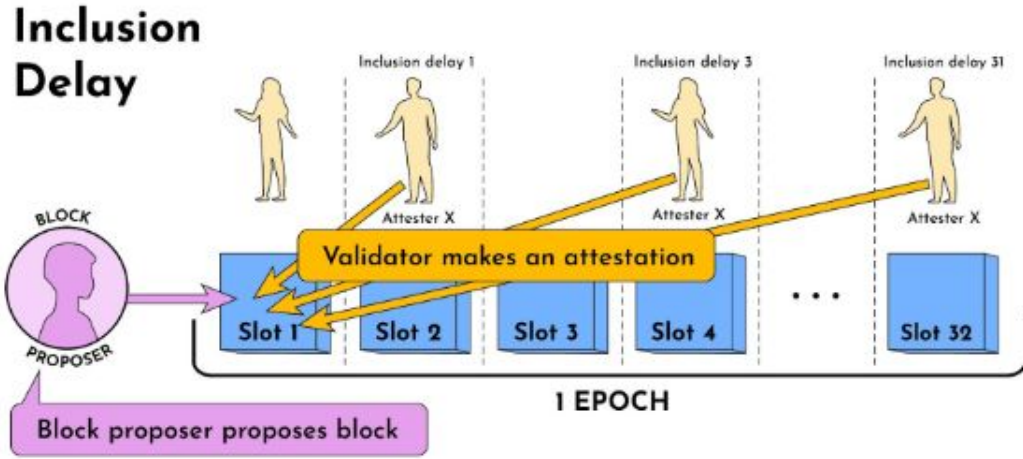
Recall that in any blockchain, the genesis block is typically the first block that everyone builds future blocks on top of. Similarly, weak subjectivity checkpoints are like genesis blocks that everyone agrees belongs to the canonical chain — as such, everyone builds on top of that weak subjectivity point.

New nodes that are trying to sync to the canonical head of the Beacon chain must use the latest weak subjectivity checkpoint as the starting point of their sync.

This is where the major criticism of weak subjectivity comes in: that nodes just have to trust that they're hearing about the latest weak subjectivity checkpoint from trusted sources (e.g., block explorers or other nodes). For those interested, Vitalik goes into his rationale for why he's okay with this in this post.

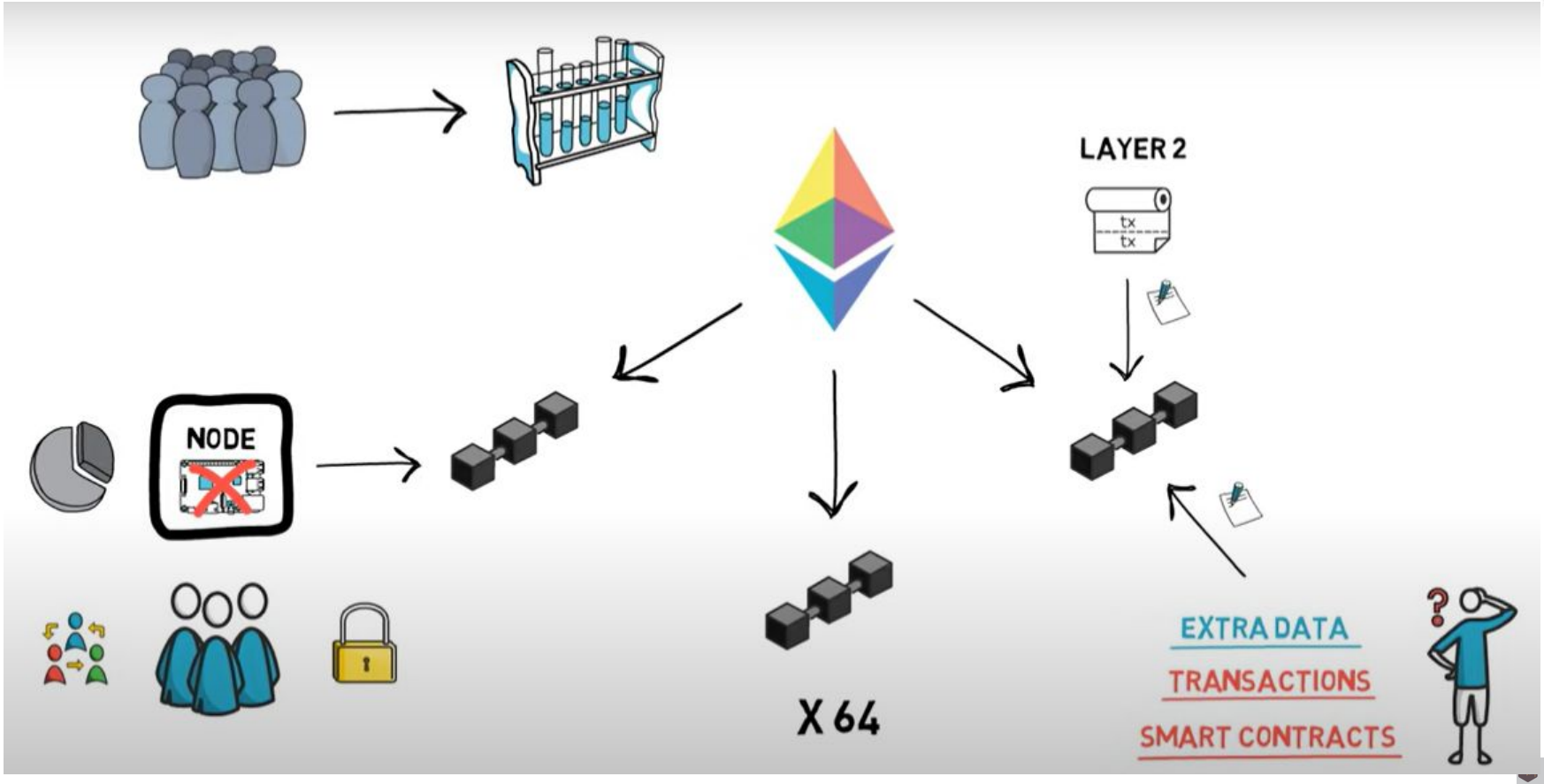
Ethereum 2.0 / ETH 2.0 / Serenity - Key features : PoS

Recall that an attestation contains three votes (source, target, head). Each vote is eligible for a reward. Validators only receive rewards for "correct" attestations — meaning their votes agree with the current block proposer. Moreover, there is an additional weight based on how quickly a validator makes attestations after a block has been proposed, where the base reward gets divided by the “inclusion delay.”



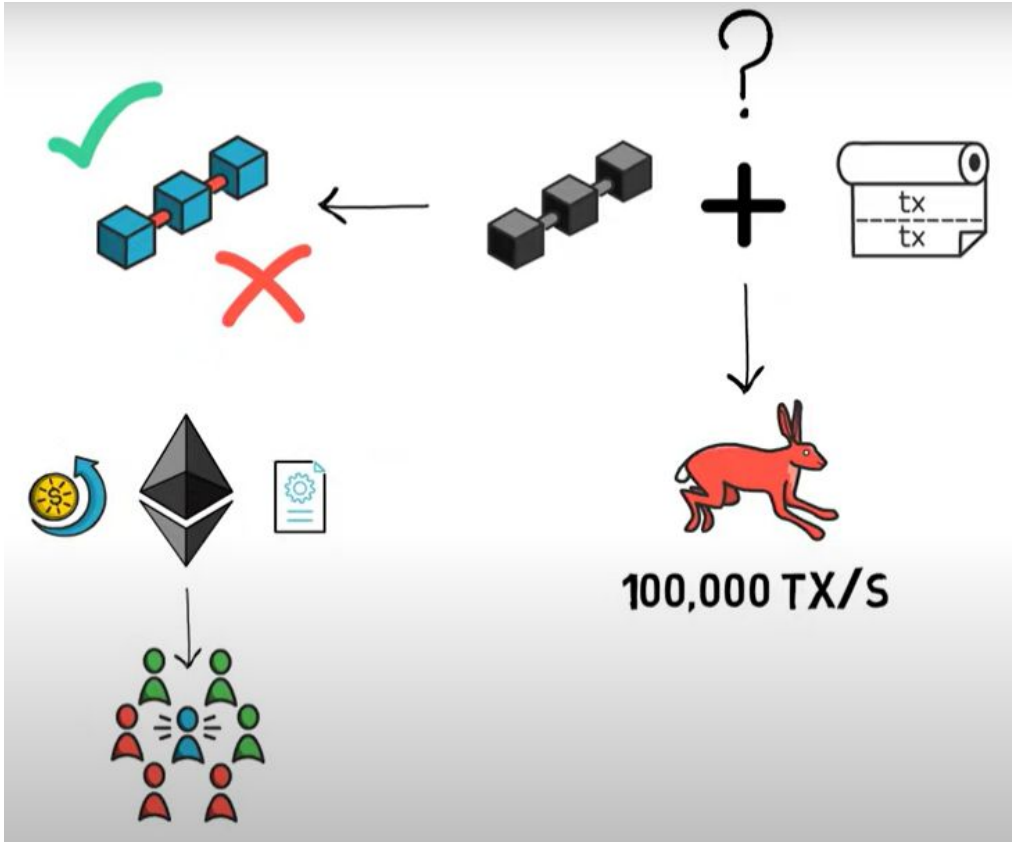
Courtesy : Working of Ethereum 2.0

Ethereum 2.0 / ETH 2.0 / Serenity - Key features : Sharding



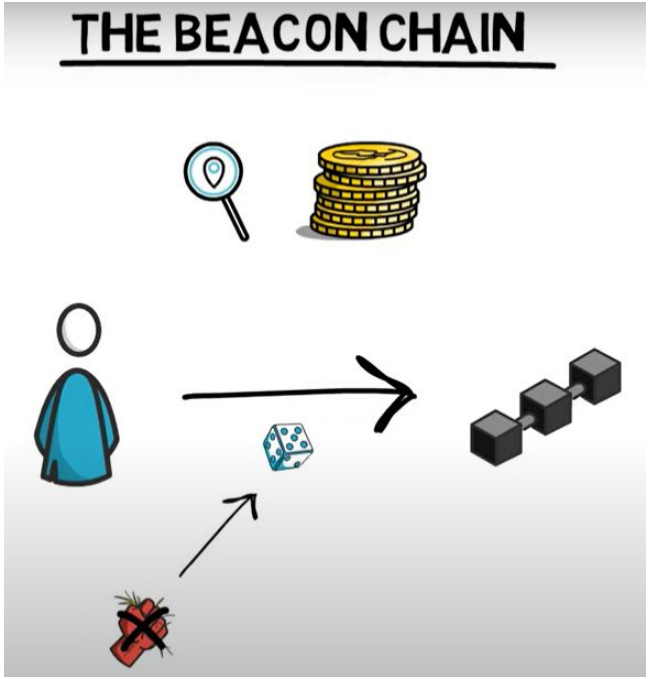
Courtesy : Ethereum 2.0

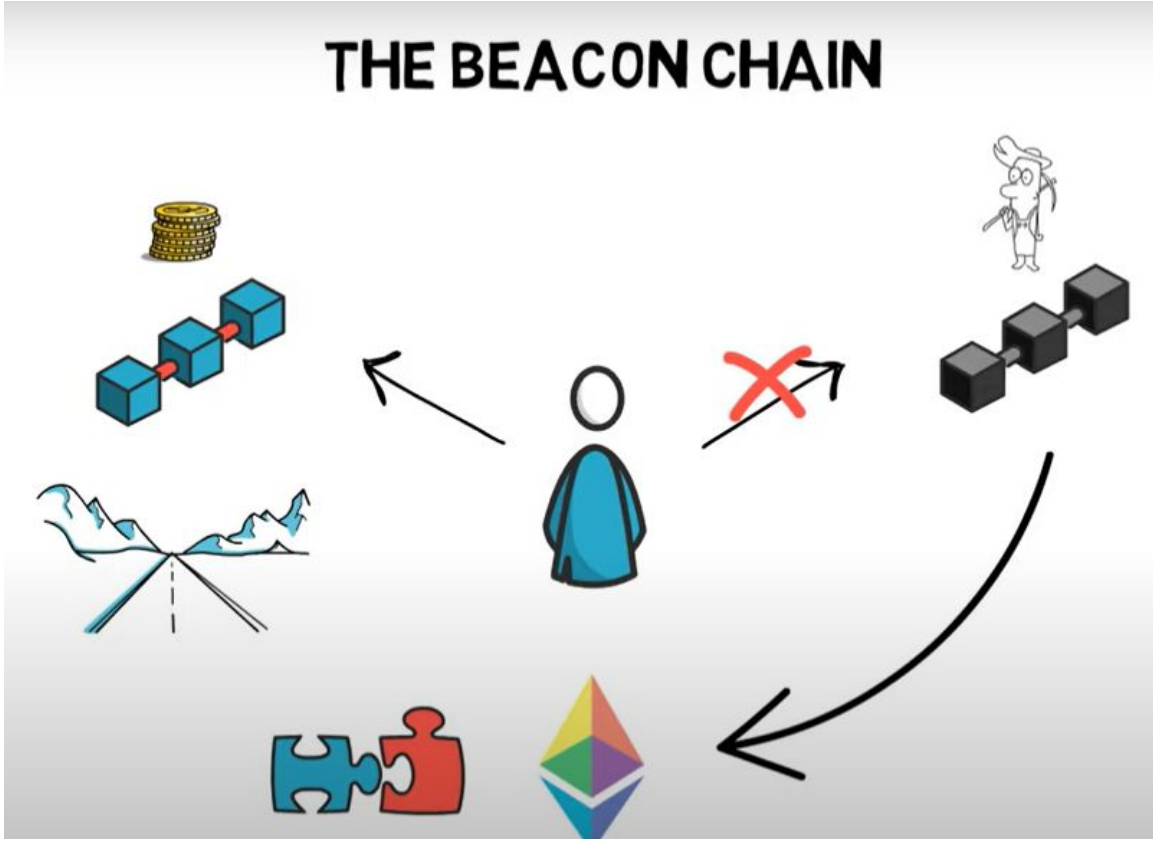
Ethereum 2.0 / ETH 2.0 / Serenity - Key features : Sharding



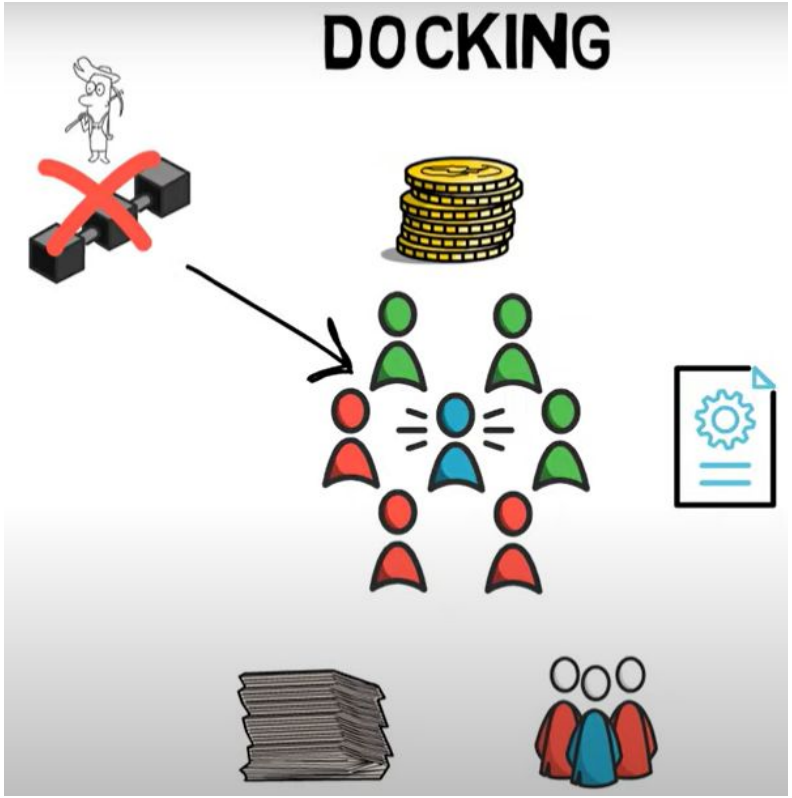
Courtesy : Ethereum 2.0

Ethereum 2.0 / ETH 2.0 / Serenity - Key features : Beacon Chain



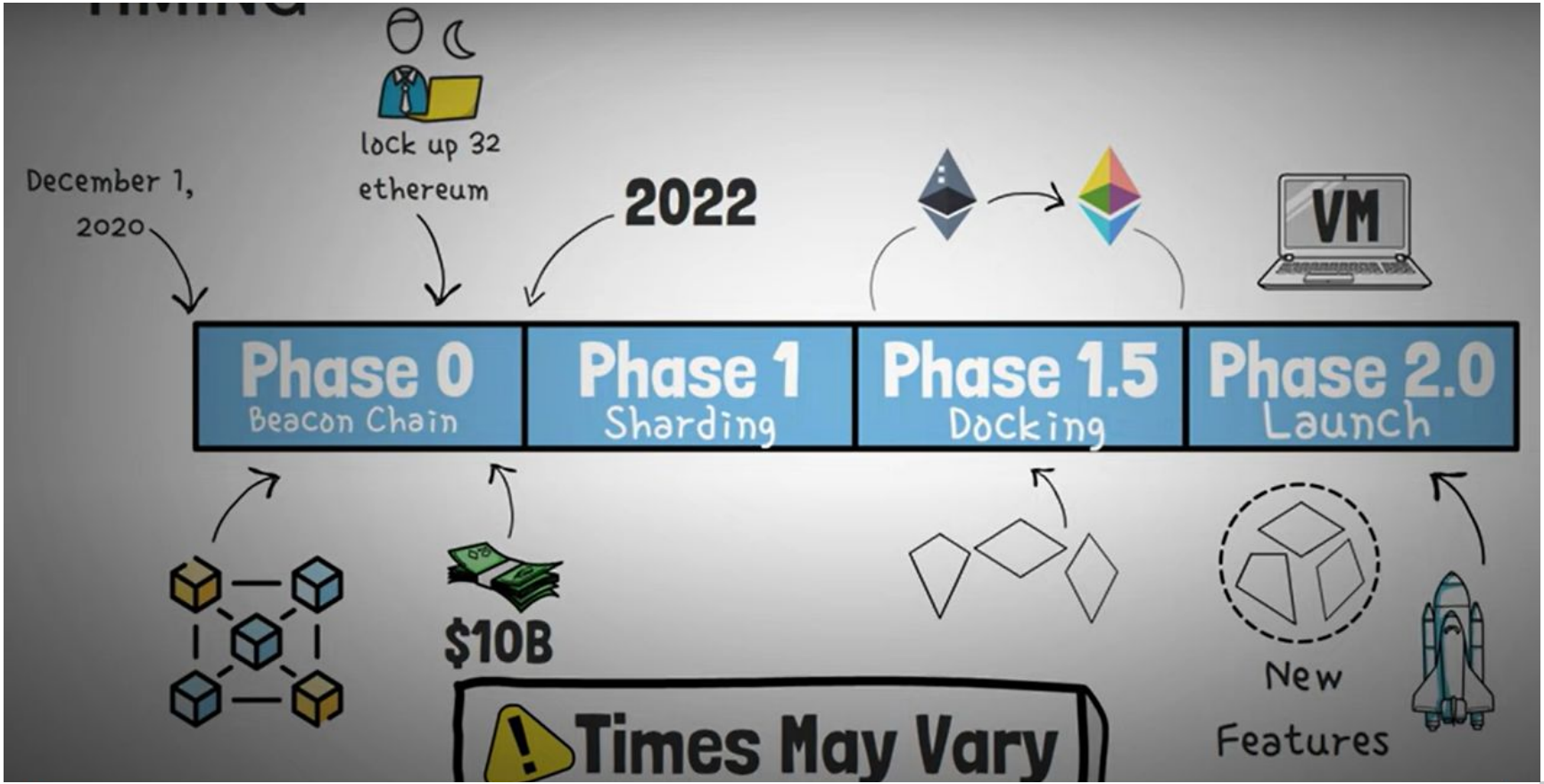


Ethereum 2.0 / ETH 2.0 / Serenity - Key features : Docking



Courtesy : Ethereum 2.0

Timeline for Ethereum 2.0 / ETH 2.0 / Serenity



Courtesy : Ethereum 2.0