UNIT IV

HYPERLEDGER FABRIC & ETHEREUM

ARCHITECTURE OF HYPERLEDGER FABRIC V1.1

Hyperledger Fabric is an open-source platform for building distributed ledger solutions, with a modular architecture that delivers high degrees of confidentiality, flexibility, resiliency, and scalability. This enables solutions developed with fabric to be adapted for any industry. This is a private and confidential blockchain framework managed by the Linux Foundation. The article focuses on discussing Hyperledger Fabric in Blockchain. The following topics will be discussed here:

1. What is Hyperledger Fabric ?
2. How does Hyperledger Fabric Work ?
3. Hyperledger Fabric Consensus Algorithm.
4. Industry Use Cases For Hyperledger Fabric.
5. Benefits Of Hyperledger Fabric.
6. Limitation of Hyperledger Fabric.

What is Hyperledger Fabric ?

Hyperledger Fabric is designed for use in enterprise-level applications, and it is characterized by its modular architecture, permissioned network, and smart contract functionality, known as "chaincode".

- The platform provides a high degree of security, privacy, and scalability, and it supports the development of custom blockchain solutions for various use cases across industries such as finance, supply chain, and healthcare.
- Hyperledger Fabric operates as a network of nodes, where each node performs a specific function, such as validating transactions, maintaining the ledger, and executing chaincode.
- Transactions are validated and ordered by a consensus mechanism, which ensures the integrity and consistency of the ledger.

How does Hyperledger Fabric Work?

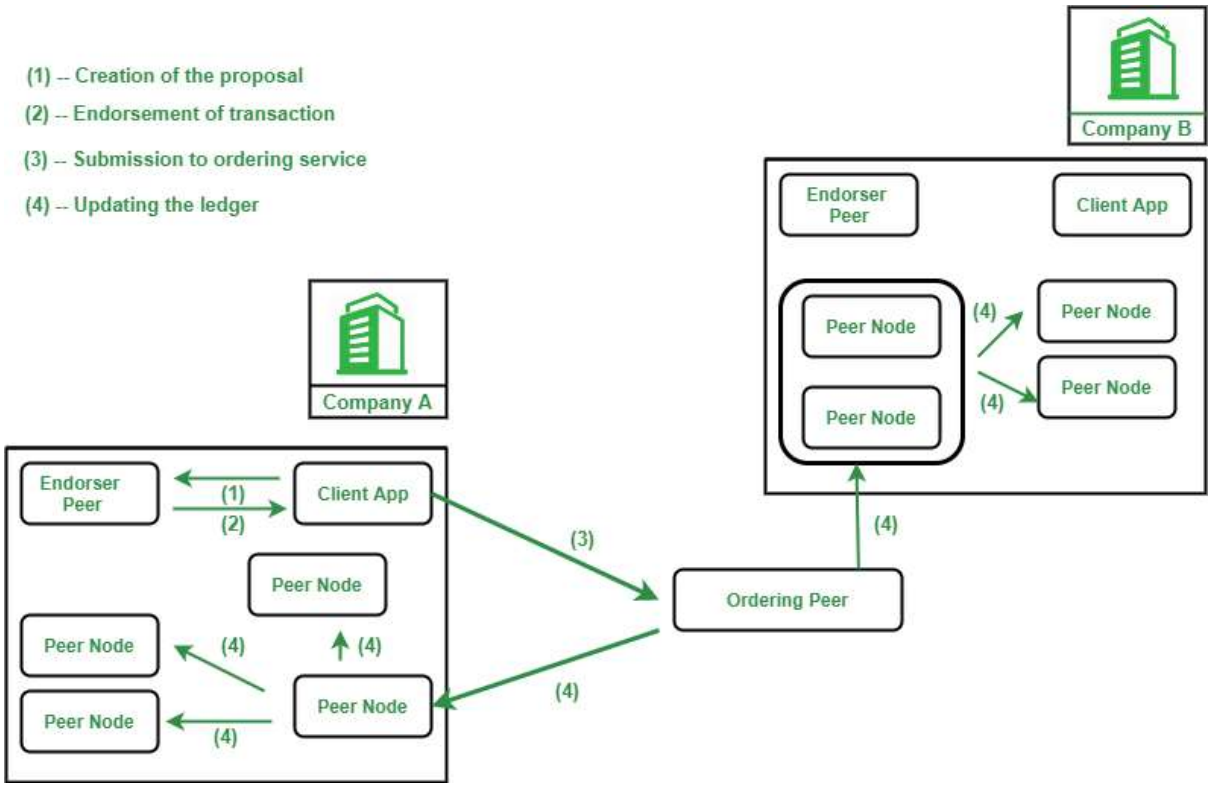
Components:

- Hyperledger fabric is an enterprise-level permission blockchain network. It is made up of various unique organizations or members that interact with each other to serve a specific purpose. For example, these organizations can be a bank, financial institution, or a supply chain network. Each organization is identified and they have a fabric certificate authority. These organizations are called members.
- Each member of the fabric can set up one or more authorized peers to participate in the network using the fabric certificate authority. All of these peers must be authorized properly.
- There is a client-side application connected to the network written with the software development kit (SDK) of any particular programming language.

Workflow:

For each and every transaction in the fabric, the following steps are followed-

1. **Creation of the proposal:** Imagine a deal between a smartphone manufacturer company and a smartphone dealership. The transaction begins when a member organization proposes or invokes a transaction request with the help of the client application or portal. Then the client application sends the proposal to peers in each organization for endorsement.
2. **Endorsement of the transaction:** After the proposal reaches the endorser peers (peers in each organization for endorsement of a proposal) the peer checks the fabric certificate authority of the requesting member and other details that are needed to authenticate the transaction. Then it executes the chain code (a piece of code that is written in one of the supported languages such as Go or Java) and returns a response. This response indicates the approval or rejection of the following transaction. The response is carried out to the client.
3. **Submission to ordering service:** After receiving the endorsement output, the approved transactions are sent to the ordering service by the client-side application. The peer responsible for the ordering service includes the transaction into a specific block and sends it to the peer nodes of different members of the network.
4. **Updating the ledger:** After receiving this block the peer nodes of such organizations update their local ledger with this block. Hence the new transactions are now committed.



Hyperledger Fabric Consensus Algorithm

Hyperledger Fabric uses a consensus algorithm to achieve agreement among the participants in a network on the contents of the shared ledger. The consensus algorithm in Hyperledger Fabric is pluggable, which means that it can be replaced with a different algorithm as needed.

The most commonly used consensus algorithms in Hyperledger Fabric are:

- **Practical Byzantine Fault Tolerance (PBFT):** PBFT is a consensus algorithm that provides fault tolerance and reliability in a network. It is well-suited for networks with a limited number of participants who are trusted and well-known.
- **RAFT**: RAFT is a consensus algorithm that is used to maintain a consistent state across multiple nodes. It is well-suited for networks where the participants are unknown and potentially untrusted.
- **Solo**: Solo is a consensus algorithm that is used for testing purposes in a single-node network. It is not suitable for production use.

Industry Use Cases For Hyperledger Fabric

1. Supply Chain: Supply chains are global or regional webs of suppliers, manufacturers, and retailers of a particular product. Hyperledger Fabric networks can improve the transaction processes of the supply chain by increasing the clarity and traceability of transactions within the fabric. On a Fabric network, enterprises having authentication to access the ledger can view the data of the previous transactions. This fact increases accountability and reduces the risk of counterfeiting of the transactions. Real-time production and shipping updates can be updated to the ledger. Which can help us to track the product condition in a much faster, simpler, and efficient way.
2. Trading and Asset Transfer: Trading and asset transfer requires many organizations or members like importers, exporters, banks, brokers. They work with one another. And even in the era of digitalization a lot of paperwork is going on in this sector. But using Hyperledger they can transact and interact with each other in a paperless way. The Hyperledger fabric can add the same layer of trust as the document signed by a trusted authority. This also increases the performance of the system. Another benefit of Hyperledger fabric is that assets can be dematerialized on the blockchain network with the help of Hyperledger fabric. Due to this traders or stakeholders will be able to have direct access to their financial securities and they can trade it anytime.
3. Insurance: The insurance industry spends billions to avoid insurance frauds or falsified claims. With the help of Hyperledger fabric, the Insurance company can refer to the transaction data that is stored inside the ledger. Hyperledger Fabric can also make the processing of claims faster using the chain code and automate the payment. This process will be also helpful for multi-party subrogation claims processing. Where it can automate repayment from the fault party back to the insurance company. Verification of identity or KYC process will be easy using this private blockchain.

Benefits Of Hyperledger Fabric

1. Open Source: Hyperledger fabric is an open-source blockchain framework hosted by the Linux foundation. It has an active community of developers The code is designed to be publicly accessible. Anyone in the community can see, modify, and distribute the code as they see fit. People across the world can come and help to develop the source code.
2. Private and Confidential: In a public blockchain network each and every node in the network is receiving a copy of the whole ledger. Thus keeping privacy becomes a much bigger concern as everything is open to everyone. In addition to this one, the identities of all the participating members are not known and authenticated. Anyone can participate as it is a public blockchain. But in the case of Hyperledger fabric, the identities of all participating members are authenticated. And the ledger is only exposed to the

authenticated members. This benefit is the most useful in industry-level cases, like banking, insurance, etc where customer data should be kept private.

3. Access Control: In the Hyperledger fabric, there is a virtual blockchain network on top of the physical blockchain network. It has its own access rules. It employs its own mechanism for transaction ordering and provides an additional layer of access control. It is especially useful when members want to limit the exposure of data and make it private. Such that it can be viewed by the related parties only. As an example when two competitors are on the same network. The fabric also offers private data collection and accessibility, where one competitor can control the access to its own data such that the data do not get exposed to the other competitor.

4. Chaincode Functionality: It includes a container technology to host smart contracts called *chain code* that defines the business rules of the system. And it's designed to support various pluggable components and to accommodate the complexity that exists across the entire economy. This is useful for some of the specific types of transactions like asset ownership change.

5. Performance: As the Hyperledger fabric is a private blockchain network, There is no need to validate the transactions on this network so the transaction speed is faster, resulting in a better performance.

Limitation of Hyperledger Fabric

Hyperledger Fabric is a robust and flexible platform for developing blockchain applications, but like any technology, it has certain limitations:

1. **Scalability:** Hyperledger Fabric is designed for permissioned networks, where the participants are known and trusted, which can limit its scalability for large-scale public networks.
2. **Performance:** The performance of Hyperledger Fabric can be impacted by factors such as network size, network configuration, and the complexity of chaincode, which can limit its ability to handle high volumes of transactions.
3. **Complexity:** Setting up and configuring a Hyperledger Fabric network can be complex, requiring a deep understanding of the technology and its components.
4. **Compatibility:** Hyperledger Fabric is designed to be used with specific programming languages, such as Go and JavaScript, which can limit its compatibility with other technologies and programming languages.
5. **Cost:** Running a Hyperledger Fabric network requires infrastructure and resources, which can add costs to the deployment and operation of blockchain applications.
6. **Interoperability:** Hyperledger Fabric is designed to be used within a single network, and its interoperability with other blockchain platforms is limited.

CHAIN CODE

Blockchain is the technology that has been making a lot of news lately. It's a hot topic because it's believed to be revolutionary by many. Hyperledger Fabric is one of the leading blockchain frameworks out there, and this article will explain what chain code is within Hyperledger Fabric. The following topics will be discussed here:

1. **What is chain code In Hyperledger Fabric?**
2. **Types of chain code In Hyperledger Fabric?**
3. **System Chain codes**

4. **Chain codes For Developers In Blockchain**
5. **Chain codes For Operators In Blockchain**
6. **Conclusion**

Let's start discussing each of these topics in detail.

What is chain code in Hyperledger Fabric?

A chain code (or "smart contract") is an application that runs on top of the underlying architecture to enforce business rules and maintain the state. This application can define its own data structures, or use the types of services exposed by other chain codes running in parallel on the same peer node.

- The node that participates in the consensus protocol validates blocks and distributes them to other peers. Each node has access to all information required for this task and runs an internal chain code that enforces business rules.
- chain code is the program that is running on a peer node.
- There are many chain codes out there and they all run the same way, enforcing business rules on transactions.
- chain code can be written in any language supported by the underlying platform.

Types of Chain codes in Hyperledger Fabric

In Hyperledger Fabric, there are two different types of chain codes:

- **Orderer chain code:** This type of chain code is executed by an Orderer node (a peer with special tagging configured in the network identity). The Orderer executes this chain code for each transaction and subsequently enforces the business logic associated with this transaction by representing it as an "Order" object. The order is then sent to the participants in the peer-to-peer network ("Peer") and they are responsible for processing it (as long as they confirm that the transaction is valid).
- **Core chain code:** This type of chain code is executed by a core node (a peer that runs multiple chain codes). The core node has special tagging configured in the network identity. The core node executes this chain code for each transaction but represents it as a new message that is sent using the Messaging service. As mentioned above, chain code will run on the peer nodes within the Hyperledger Fabric network. When running a chain code, there are several things that will be enforced on transactions by chain code. A peer group can be seen as a logical cluster of peers that are connected to each other through a leader. Each peer in this group will be running the chain code of the same type.

What are System Chain codes?

- A system chain code is a set of functions that provides the blockchain runtime with information about the status of a transaction, while still allowing the transaction to execute. It can be used to validate or reject transactions, or to obtain additional information about Blockchain interactions.
- In Hyperledger Fabric, system chain codes are usually written in a programming language based on JavaScript (e.g., Nodejs) and run in a container as an isolated process. A developer can write custom application-specific chain codes by using one of many available SDKs for systems programming languages such as C++, Go, Rust and Python.

- Hyperledger Fabric is a blockchain-based distributed ledger with a modular architecture, which is, in general, compatible with Hyperledger Fabric.
- There are several organizations that provide support for the Hyperledger Fabric project, and Hyperledger is one of the key contributors to this effort. It is said to be an open-source distributed ledger framework designed by the Linux Foundation.
- It has been inspired by the Bitcoin Blockchain as well as the Ethereum Platform and its purpose is to have an open source blockchain system in which each participating enterprise can design and build their own custom applications to interact with the blockchain "to improve their operational processes or business model".

Chain code For Developers in Blockchain

chain code is a program that lives in the blockchain and it executes transactions when instructed. It offers an alternative to using scripts to create smart contracts.

- Chain code is written in JavaScript which allows developers without a background in crypto economics to understand the codebase and use it in their applications.
- Chain code for developers has been pioneered by Hyperledger Fabric, an open-source framework from Linux Foundation that implements a blockchain architecture with permission ledgers and multi-party computation (MPC).
- As per the spec, chain code will be used for "stateful business logic", which does not require any state change by a consensus algorithm.
- Chain code is code that is invoked by participating nodes in order to validate transactions that are part of a block's blockchain; it can also invoke other transaction types like event subscription or Coinbase transactions. Chain codes also execute scriptable data-management operations on the blockchain's state database.
- Chain code is also referred to as smart contracts due to its ability to interact with the blockchain and perform data manipulation functions.
- Several different types of chain codes are available from Coda Blockchain's Smart Contracts Library, including oracle, Mult signature, timestamping, and Keystore. Coda also provides various coinbase operations for creating/composing transactions that include transaction value and identifying remarks on the blockchain.
- As a means of simplifying development on the blockchain and facilitating interop with legacy systems, Coda provides a number of features such as the use of native libraries like libevent in order to provide access to APIs like libbitcoin or libledger

***C*hain code For Operators**

Chain code is a fabric-specific script written to perform operations within the framework. Hyperledger Fabric's chain code enables a user with no knowledge of blockchain technology to build and deploy smart contracts and transactions.

- These scripts can be defined as executable pieces of software that act on input data generated by other code.
- The chain code creates and runs transactions using consensus across nodes in the decentralized system; this helps to enforce the rules set within entities'

business logic while taking into account all contributions from other participants in the network, known as actors.

- This includes both transactions between actors, such as customers or suppliers, but also interactions with fabric-specific functions like transaction ordering or reconciliation.
- Within the context of blockchain development, smart contracts also refer to contract code that is enforced by the distributed blockchain ledger and executed automatically when a condition specified in the contract code is triggered.
- In this context, smart contracts are executable programmable logic (programs) that are stored on the blockchain, which can only be run by authorized parties. For operators, they can be used as a means of upgrading their distributed ledger systems to provide new functionalities, such as allowing users to issue assets and make transactions with them.
- Chain code provides an interface for smart contracts to be programmed via Chain code. Coda's Chain code performs the following functions:
- Chain code is launched in one of two ways in Coda: Either by invoking it via the CLI or programmatically via SDK calls. In order for the node operator to launch chain code, one must first authorize launch permissions by providing a list of node IDs for which the chain code is authorized to run.
- Chain codes should primarily be written using Go, but other languages such as Python and NodeJS can also be used.

ETHEREUM: ETHEREUM NETWORK

Ethereum is a decentralized global software platform powered by blockchain technology. It is most commonly known by investors for its native cryptocurrency, ether (ETH), and by developers for its use in blockchain and decentralized finance application development.

Anyone can use Ethereum—it's designed to be scalable, programmable, secure, and decentralized—to create any secured digital technology. Its token is designed to pay for work done supporting the blockchain, but participants can also use it to pay for tangible goods and services if accepted.

KEY TAKEAWAYS

- Ethereum is a blockchain-based development platform known for its cryptocurrency, ether (ETH).
- The blockchain technology that powers Ethereum enables secure digital ledgers to be publicly created and maintained.
- Bitcoin and Ethereum have many similarities but different long-term visions and limitations.
- Ethereum uses a proof-of-stake transaction validation mechanism.
- Ethereum is the foundation for many emerging technological advances based on blockchain.

History of Ethereum

Vitalik Buterin, credited with conceiving Ethereum, published a white paper introducing it in 2014. The Ethereum platform was launched in 2015 by Buterin and Joe Lubin, founder of the blockchain software company ConsenSys.

The founders of Ethereum were among the first to consider the full potential of blockchain technology beyond just enabling a secure virtual payment method.

Since the launch of Ethereum, ether as a cryptocurrency has risen to become the second-largest cryptocurrency by market value. It is outranked only by Bitcoin.

A Historic Split

One notable event in Ethereum's history is the hard fork, or split, of Ethereum and Ethereum Classic. In 2016, a group of network participants gained control of the smart contracts used by a project called The DAO to steal more than $50 million worth of ether.

The raid's success was attributed to the involvement of a third-party developer for the new project. Most of the Ethereum community opted to reverse the theft by invalidating the existing Ethereum blockchain and approving a blockchain with a revised history.

However, a fraction of the community chose to maintain the original version of the Ethereum blockchain. That unaltered version of Ethereum permanently split to become Ethereum Classic (ETC).

Ethereum vs. Bitcoin

Ethereum is often compared to Bitcoin. While the two cryptocurrencies have many similarities, there are some important distinctions.

Ethereum is described by founders and developers as “the world’s programmable blockchain,” positioning itself as a distributed virtual computer on which applications can be developed.The Bitcoin blockchain, by contrast, was created only to support the bitcoin cryptocurrency as a payment method.

The maximum number of bitcoins that can enter circulation is 21 million.The amount of ETH that can be created is unlimited, although the time it takes to process a block of ETH limits how much can be minted each year.The number of Ethereum coins in circulation as of May 2024 is just over 120 million.

Another significant difference between Ethereum and Bitcoin is how the respective networks treat transaction processing fees. These fees, known as gas on the Ethereum network, are paid by the participants in Ethereum transactions and burned by the network. The fees associated with Bitcoin transactions are paid to Bitcoin miners.

Ethereum uses a proof-of-stake consensus mechanism. Bitcoin uses the energy-intensive proof-of-work consensus, which requires miners to compete for rewards.

The Future of Ethereum

Ethereum’s transition to the proof-of-stake protocol, which enabled users to validate transactions and mint new ETH based on their ether holdings, was part of a significant upgrade to the Ethereum platform. However, Ethereum now has two layers. The first layer is the execution layer, where transactions and validations occur. The second layer is the consensus layer, where attestations and the consensus chain are maintained.

The upgrade added capacity to the Ethereum network to support its growth, which will eventually help to address chronic network congestion problems that have driven up gas fees

EVM

Ethereum Virtual Machine (EVM) is designed as the runtime environment for smart contracts in Ethereum. It is sandboxed and isolated from the other parts of the system. This means that any operation on EVM should not affect your data or programs in any way, no matter how many times you call a particular function on it.

- An EVM is the runtime environment that executes Ethereum smart contracts.
- Ethereum contains its own Turing-complete scripting language, called Solidity, and with this comes a need to execute this code.
- A program called the Ethereum Virtual Machine (EVM) can do this task.
- It runs on top of the Ethereum network, meaning that all nodes reach a consensus about what code should be executed at every given time.

Purpose of EVM

The Ethereum Virtual Machine (EVM) is a Turing complete programmable machine, which can execute scripts to produce arbitrary outcomes. It has been built with the purpose of being a "world computer" and has immense power.

- It is the computer that stores data on blockchain, like bitcoin, but it also executes code in smart contracts on the Ethereum network.
- The machine is made to be able to run any kind of Crypto-contract that can be built on Ethereum's blockchain. It does this by using a programming language called Solidity, which is compiled into the EVM for execution.
- The intention behind writing code on the Ethereum network is to create smart contracts and programs that automatically execute things when certain conditions are met. If a terms or condition is not met, the system can execute it in an "exit" function as well.
- For example, if an account has been hacked, the hacker cannot steal money from the system, because they don't have the budget or authority to do so.

How Does EVM Works?

Ethereum Virtual Machine (EVM) is a program which executes scripts used to implement certain operations usually in Ethereum blockchain.

The Ethereum Virtual Machine makes the process of creating new tokens on Ethereum Blockchain easy. Here, script means a set of instructions or an algorithm which tells the computer what it needs to do in order for something to work properly. The EVM requires that one has access over any network node so as to be able to execute the desired commands and create new tokens on the blockchain without any difficulties.

- In Ethereum, there is something called a smart contract. These contracts have some computer code which facilitates the exchange of money and information.

- These contracts are predefined by the creator of the smart contract, in order to ensure that a certain outcome will happen based on either what happens or doesn't happen.
- Ethereum Virtual Machine provides Turing complete environment for execution of scripts and smart contracts. This means that anything that can be implemented with a computer can be run on EVM.

In the Ethereum ecosystem, EVM plays a vital role by providing a platform for decentralized applications (DApps) to be built on top of it.

Ethereum Virtual Machine ensures that all transactions and smart contracts made on the Ethereum blockchain are executed in correct and expected manner as desired by the smart contract code. It serves as a platform for applications to be executed on. In simple words, it can be said that Ethereum Virtual Machine facilitates DApp creation and execution on the blockchain.

Ethereum Virtual Machine (EVM) has two parts:

- **EVM (the part that runs solidity source code):** The EVM is written in C++ and uses LLVM as its compiler. It is a full-featured virtual machine with all the features that you would want in a general purpose Smart Contract Virtual Machine, such as support for multiple programming languages, security features, runtime environments and more. It also allows you to write custom EVM bytecode .
- **Uncles:** These are small pieces of smart contracts or data stored on the blockchain. This is a useful feature because it allows for you to store metadata about your program. EVM Assembly: This is the bytecode of EVM, which you can use as your programming language.

These are small pieces of smart contracts or data stored on the blockchain. This is a useful feature because it allows one to store metadata about the program.

- **Actions:** These are basic operations that one can perform on assets stored in memory (and not on the blockchain), such as multiplication, addition, and so on.
- **Balance:** This is the amount of Ether that one have at any time. So, if there is a balance of 100 Ether and spend 10 Ether, the balance would be 90 Ether. Note that this is not actually a variable, it's just a part of memory where EVM stores the data. This means that whenone tries to modify or read from it, it will return execution with an error.
- **Block:** This is an immutable storage for all actions and transactions related to Ethereum in its lifetime up until this block in particular. These blocks can be only 65,000 so this is not going to change.
- **Blockhash:** This is a hash of the block in question. So, if you are looking at stored on the blockchain under another name, this would be a hash of the data stored there.
- **Block Number:** This is a number indicating which block this particular blockhash belongs to. It always starts from zero and increases every time there's a new block added to the chain. Note that blocks have timestamps associated with them so you can tell how much time passed between two blocks.
- **Code:** This is code executed in EVM that determines what action is going to be taken when an input happens (such as transferring money).

- **CodeHash:** This is a hash of the code itself. If one looks at a contract on Etherscan, the CodeHash is what one will see. When functions are executed on EVM, this number changes because the code itself changes based on the input.
- **CodeSize:** This is the actual size of the code in bytes.
- **GasLimit:** This is a part of EVM that allows for users to specify how much gas they are willing to spend in order to execute something. If this number is zero, then nothing will happen (this rarely happens).

How Does Gas RelateTo Performance Of EVM?

- Gas is a measure of computational power. It determines how much time each transaction and contract takes to execute.
- Because there is so much code already in the system, it uses a limited amount of Gas to run all of this code. It sets the default gas limit to 250,000 gas units.
- In general, the more complicated your transaction, the more gas it takes to execute.

Benefits of EVM

- **Execute untrusted code without risking data:** One can execute untrusted code without putting the data at risk. EVM guarantees that its computations will not interfere with anything else happening in the system or with the personal files.
- **Can run complex smart contracts:** One can run complex smart contracts in EVM without worrying about how they interact with each other. One can write them once and then run them on multiple platforms, which allows for the creation of a single contract that runs on multiple computing environments.
- **Deterministic processing:** Smart contracts written on EVM have access to all of Ethereum's states at any given time, allowing for processing to happen in a deterministic way and giving more guarantees about their correctness. For example, one cannot make an infinite loop in EVM by calling the same function twice. It would stop executing and return a finite value.
- **Distributed consensus:** One of the potential applications of Ethereum is to allow for distributed consensus where everyone is running the same program but from their own computers.
- **Robust against failure:** This is a complex process because the network needs to be able to come to a consensus at any given time. This way, the system becomes more robust against failures of individual nodes and you can update several nodes simultaneously without worrying that they might end up disagreeing with each other because of how code was written.
- **Easy to write stateful contracts:** From a developer perspective, EVM is designed for writing smart contracts as well as for creating DApps (decentralized applications), which are programs running on distributed networks in a way that ensures all of them are seeing the same version. It also makes it incredibly easy to write stateful contracts, which need access to some kind of persistent storage.

TRANSACTION FEE

Transaction fees serve two essential purposes when it comes to blockchain networks. They reward miners or validators who help confirm transactions and help protect the network from spam attacks.

Transaction fees can be both small or large, depending on the network activity. Market forces can also influence the fees you pay. While high fees can hinder wider blockchain adoption, very low fees could potentially bring security concerns.

Transaction fees are and have been an essential part of most blockchain systems since their inception. You are most likely to have come across them when sending, depositing, or withdrawing crypto.

The majority of cryptocurrencies use transaction fees for two important reasons. First of all, fees reduce the amount of spam on the network. It also makes large-scale spam attacks costly and expensive to implement. Secondly, transaction fees act as an incentive for users that help verify and validate transactions. Think of it as a reward for helping the network.

For most blockchains, transaction fees are reasonably cheap, but they can get quite expensive depending on network traffic. As a user, the amount you choose to pay in fees determines your transaction's priority in being added to the next block. The higher the fee paid, the quicker the confirmation process.

Bitcoin Transaction Fees

As the world's first blockchain network, Bitcoin set the standard for transaction fees used by many cryptocurrencies today. Satoshi Nakamoto realized that transaction fees could protect the network from large-scale spam attacks and incentivize good behavior.

Bitcoin miners receive transaction fees as part of the process of confirming transactions to a new block. The pool of unconfirmed transactions is called the memory pool (or mempool). Naturally, miners will prioritize transactions with higher fees, which users agreed to pay when sending their BTC to another bitcoin wallet.

Malicious actors who wish to slow down the network must therefore pay a fee associated with each transaction. If they set the fee too low, miners will likely ignore their transactions. If they put them at a suitable level, they incur a high economic cost. So, transaction fees also act as a simple but effective spam filter.

How Are BTC Transaction Fees Calculated?

On the Bitcoin network, certain crypto wallets allow users to set their transaction fees manually. It's also possible to send BTC with zero fees, but miners will most likely ignore such transactions, meaning they won't be validated.

Unlike some tend to believe, Bitcoin fees are not dependent on the amount sent but on the transaction size (in bytes). For example, imagine your transaction size is 400 bytes, and the average transaction fee is now at 80 satoshis per byte. In that case, you would have to pay around 32,000 satoshis (or 0.00032 BTC) for a good chance of having your transaction added to the next block.

When network traffic is high, and there is a great demand for sending BTC, the transaction fee needed for speedy confirmation rises as other bitcoin users try to do the same. This may occur during periods of intense market volatility.

As such, the high fees can make it challenging to use BTC in day-to-day situations. Buying a $3 cup of coffee might not be practical if the fees are much higher than that.

Only a certain number of transactions can be included within a block, which has a limit of 1MB (i.e., block size). Miners add these blocks to the blockchain as quickly as possible, but there is still a limit to how fast they can go.

The scalability of cryptocurrency networks is a crucial issue here in deciding network fees. Blockchain developers are making continuous efforts to address the problem. Previous network updates have helped improve scalability, such as the implementation of SegWit and the Lightning Network.

Ethereum Transaction Fees

Ethereum transaction fees work differently in comparison to Bitcoin's. The fee takes into account the amount of computing power needed to process a transaction, known as gas. Gas also has a variable price measured in ether (ETH), the network's native token.

While the gas needed for a specific transaction can stay the same, gas prices can rise or fall. This gas price is directly related to network traffic. If you pay a higher gas price, miners will likely prioritize your transaction.

How Are Ethereum Transaction Fees Calculated?

The total gas fee is simply a price that covers the cost, plus an incentive to process your transaction. However, you should also consider the gas limit, which defines what's the maximum price paid for that transaction or task.

In other words, the gas cost is the amount of work required, and the gas price is the price paid for “each hour” of work. The relation between these two and the gas limit defines the total fee for an Ethereum transaction or smart contract operation.

Let's pick a random transaction on Etherscan.io as an example. The transaction cost 21,000 gas, and the gas price was 71 Gwei. So, the total transaction fee was 1,491,000 Gwei or 0.001491 ETH.

MIST BROWSER

The Mist browser was intended to be an integral part of the Ethereum network's dApps (decentralized applications) ecosystem. It was the first graphical user interface that enabled users to access the blockchain at a time when you could only access it via the command line. Its developers wanted to offer a one-stop-shop for running and executing various Ethereum applications and projects.

Unfortunately, the technical requirements of a fully decentralized dApp browser system were too far beyond what the technology allowed at the time. As a result, the Mist browser project was abandoned, and the software was taken out of circulation in March of 2019

- The Mist browser was a decentralized app on the Ethereum network from 2015 to 2019.
- Mist was the first browser that allowed users to browse dApps, and it had an Ethereum wallet built in. It was also the first desktop crypto wallet with a graphical user interface (GUI)

Mist was deprecated in March 2019 after developers decided other browser developers and wallet makers were better able to create products for this quickly evolving space Understanding the Mist Browser
The Mist browser was an Ethereum interface intended to allow users to access the various dApps available on the Ethereum network. It was also known as the Ethereum dApp Browser.Ethereum is a popular blockchain optimized for smart contracts and other decentralized applications.

As a dApp browser, Mist was a standalone application with a graphical user interface (GUI) that allowed users to sync to the blockchain. It also provided an easy way for users to create their own dApps and deploy tokens and other smart contracts in a non-technical way. The Mist Ethereum wallet itself would run on a user's computer, which meant it had to be downloaded, installed, and run locally.

Mist Browser?
The Mist browser was an attempt to develop a graphical user interface for the Ethereum blockchain and virtual machine. It was sunsetted in 2019

What Replaced the Mist Browser?
Mist browser was essentially a wallet, so it was replaced by many other wallets that allow you to access cryptocurrency, blockchains, dApps, and even trade on a cryptocurrency exchange.

ETHER

Ethereum is a decentralized blockchain platform that enables the development and execution of smart contracts and decentralized applications (DApps). It provides a

foundation for building various blockchain-based solutions and allows for the creation of programmable digital assets.

What is Ether?

Ether (ETH) is the native cryptocurrency of the Ethereum network. It serves as the fuel that powers the execution of smart contracts, enables transactions, and facilitates interactions within the Ethereum ecosystem.

Is Ether the same as Ethereum?

No, Ether and Ethereum are not the same. Ethereum is the blockchain platform, whereas Ether is the cryptocurrency native to that platform. Ethereum provides the infrastructure and capabilities for decentralized applications, while Ether is the digital asset used within that ecosystem.

Ether Uses

- **Transaction Fees and Gas:** Ether is used to pay transaction fees on the Ethereum network. Users must include a fee, known as "gas," to incentivize miners or validators to process and validate transactions.
- **Smart Contract Execution:** Ether is required to execute smart contracts on the Ethereum network. It serves as the currency for deploying and interacting with these self-executing agreements.
- **Decentralized Finance (DeFi):** Ether plays a significant role in DeFi applications. It can be used as collateral for loans, liquidity provision, yield farming, and governance within DeFi protocols.
- **Token Creation and ICOs:** Ether has been instrumental in the creation of tokens on the Ethereum platform. Many projects conduct Initial Coin Offerings (ICOs), raising funds by selling their tokens in exchange for Ether.
- **Interoperability and Integration:** Ether can be tokenized as Wrapped Ether (WETH) and used on other blockchain platforms, enabling interoperability and seamless integration between Ethereum and other ecosystems.

How to Buy Ether?

To buy Ether, you can follow these steps:

1. Set up a cryptocurrency wallet that supports Ether.
2. Choose a cryptocurrency exchange that allows you to buy Ether.
3. Create an account on the exchange and complete the necessary verification process.
4. Deposit funds into your account using fiat currency or other cryptocurrencies.
5. Place a buy order for Ether on the exchange, specifying the desired amount and price.
6. Once the order is filled, the purchased Ether will be credited to your wallet.

How is Ether used to power DApps?

Ether plays a crucial role in powering DApps on the Ethereum network. It is used for:

- **Accessing DApp features:** Ether can be required to access specific functionalities, premium features, or services within a DApp.
- **In-App Purchases:** DApps may allow users to make purchases using Ether. This can include buying digital goods, virtual assets, or in-game items.
- **Decentralized Governance:** Ether can be staked or used for voting within decentralized autonomous organizations (DAOs) to participate in decision-making processes and shape the direction of the organization or network.

Ether vs Bitcoin

Below is the difference between Ethereum and Bitcoin:

	Ether (ETH)	Bitcoin (BTC)
Purpose	Fuel for the Ethereum network	Digital currency/store of value
Blockchain	Ethereum	Bitcoin
Functionality	Executes smart contracts	Enables peer-to-peer transactions
Supply	Uncapped supply	Capped supply of 21 million
Consensus Mechanism	Transitioning to Proof of Stake (PoS)	Proof of Work (PoW)
Transaction Speed	Faster	Slower
Use Cases	DApps, DeFi, Tokenization	Store of value, transactions

Value Benefits of Ether

- **Utility and Versatility:** Ether offers utility beyond being a traditional cryptocurrency. It serves as the fuel for executing smart contracts and enables the creation of decentralized applications (DApps). This versatility positions Ether as a valuable asset in the evolving blockchain landscape.
- **Thriving Ecosystem:** Ethereum boasts a vibrant and rapidly growing ecosystem. It attracts developers, entrepreneurs, and investors, fostering innovation and creating numerous opportunities for collaboration and value creation. The thriving ecosystem contributes to the long-term value potential of Ether.
- **Interoperability:** Ether's tokenization capability, such as Wrapped Ether (WETH), allows it to seamlessly interact with other blockchain platforms. This interoperability opens doors for cross-chain applications, liquidity pools, and decentralized finance (DeFi) protocols, expanding the reach and utility of Ether.
- **Liquidity and Market Presence:** Ether is one of the most widely recognized and traded cryptocurrencies in the market. Its high liquidity facilitates easy buying, selling, and conversion, enabling users to access and move value efficiently within the Ethereum ecosystem and beyond.
- **Network Effects and Community:** Ethereum benefits from a strong and active community of developers, enthusiasts, and stakeholders. The collective efforts and collaborations within the Ethereum community contribute to the network effects, driving adoption, innovation, and the overall value of Ether.
- **Decentralization and Security:** As a cryptocurrency native to a decentralized blockchain platform, Ether offers security and immutability. It operates on a distributed network of nodes, reducing the risk of censorship, single points of

failure, and unauthorized alterations, providing users with increased trust and confidence.

- **Potential for Growth and Adoption:** Ether's early adoption and continued development make it well-positioned for potential future growth. As more industries embrace blockchain technology and decentralized applications, Ether's value may increase as it becomes an integral part of innovative solutions across various sectors.

GAS

Ethereum Gas is a section that calculates the quantity of calculation action that it takes to perform specific functions. Every function that carries position in Ethereum like transactions and smart contracts etc. performance needs some part of gas. It is essential to the blockchain P2P network because it is the power that authorizes it to accomplish exactly what an automobile needs fuel to drive. Gas refers to the cost required to complete a deal on the Ethereum network.

Facts about Ethereum Gas

- Gas prices are delivered in the form of Ethereum's born money, the currency of Ethereum is ETH.
- Gas costs are indicated in "gwei" which is a movement of Ethereum, every single "gwei" is equivalent to 0.000000001 ETH.
- For sample, rather than stating that the gas costs 0.000000001 ether, say, the gas costs 1 "gwei".
- The term "gwei" means "Giga-wei", which is equivalent to 1,000,000,000 "wei".
- "Wei" is called after "Wei Dai" that is the creator of b-money (least unit of ETH).

Gas Fees

The gas fees include Base, Priority, Max, and calculating fees. Let's discuss these terms in detail.

1. Base fees:

- Each alliance has a ground fee which serves as a fund price.
- The ground fee is figured alone of the existing alliance and is rather specified by the alliances before it pushes trade prices better for users.
- When the block is excavated the base fee is "burned", dragging it from regulation.
- The ground fee is evaluated by instructions that compare the length of the earlier block with the marked size.
- The base fee intention rise by a max of 12.5% per block if the marked block size is surpassed.

2. Priority fee:

- Apart from the base fee obtained, the advancement presented a priority fee i.e. tip to miners to contain a trade in the alliance.

- Without tips, miners can see it financially possible to drill cleared blocks because they can obtain the exact alliance prize.
- Under normal circumstances, a little tip provides miners with the slightest encouragement to maintain a transaction.
- For dealings that require getting preferentially conducted ahead of different trades in the same block, a more elevated tip will be essential to endeavor to outbid contending trades.

3. Max fee:

- To conduct trade on the P2P network, users can select the greatest limit they are inclined to spend for their dealing to be completed.
- This additional circumference is called the max fee/Gas.
- For a trade to be completed, the max fee must surpass the aggregate of the base and priority fees.
- The trade sender is repaid the contrast between the max fee and the total of the ground(base) fee and tip.

4. Calculating fees:

- The main benefit of the upgrade is enhancing the user's knowledge when charging trade fees.
- In the wallets that sustain the upgrade rather than explicitly commenting to pay the transaction from a wallet, providers will automatically set a suggested transaction fee (base fee + suggested priority fee) to decrease the quantity of complexness loaded on their users.

Importance of Gas Fees

Below are some of the reasons why gas fess is important:

- Gas fees help support the Ethereum peer-to-peer network security.
- By demanding a price for every analysis performed on the web grid, it prevents evil performers from spamming the web grid.
- To evade unexpected or malicious endless circles or different computational wastage in principle, every trade is needed to set a boundary to how multiple computational efforts of code implementation can operate.
- The basic division of accounting is "gas".
- A trade organizes a boundary and any unit of gas not utilized in trade is replaced (i.e. max fee – (base fee + tip)).

Why gas fees can go high?

- Increased gas prices exist because of the fame of Ethereum.
- Conducting any function on Ethereum needs depleting gas, and the gas area is determined per alliance.
- Prices contain measures, holding or exploiting data, and swallowing various parts of "gas" units.
- As app functionality produces additional complexity, the digit of functions a wise contract achieves also develops, representing each trade brings up more additional area of a fixed measure block.
- If the demand is quite high, users must present a more elevated tip payment to try and outbid other users' dealings.

- A more increased tip can construct it additional possible that your trade will convey into the subsequent block.

Initiative to reduce gas costs:

- The ETH efficiency advancements should eventually manage some of the gas cost points, allowing the medium to process thousands of trades globally per moment.
- Second layer scaling is a direct ambition to significantly enhance gas prices, user knowledge, and scalability.
- The recent proof-of-stake(POS) sample based on the Beacon Chain, should decrease heightened control consumption and dependence on technological hardware.
- This chain resolve permits decentralized ETH P2P networks to compromise and support network security while restricting gas consumption by rather demanding an economic responsibility.

Strategies to reduce gas costs:

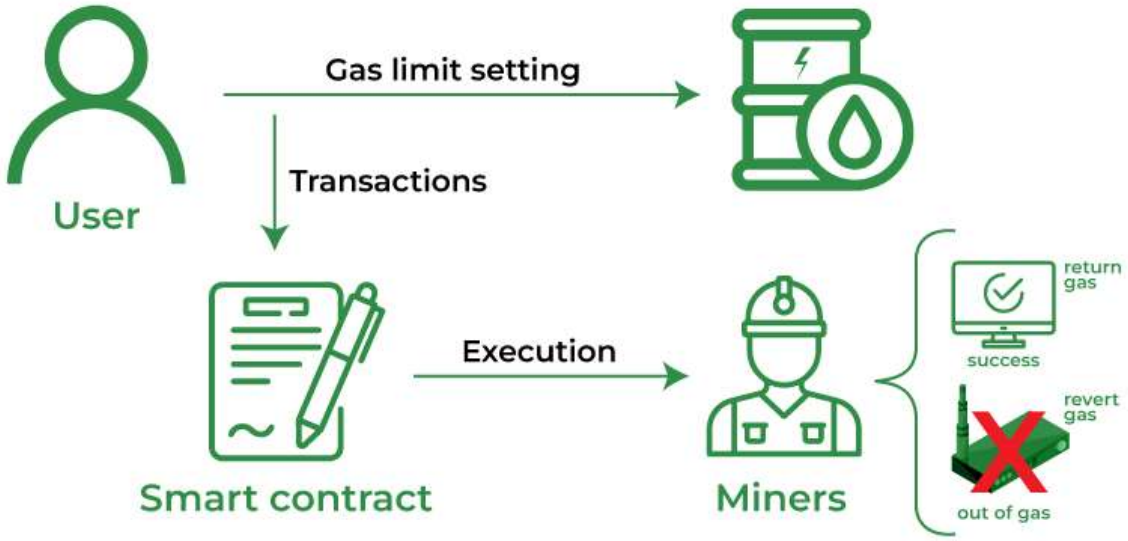
- Strategies to decrease gas costs for ETH, Users can select a direction to display the important status of the user's transaction.
- Miners intention perform trades that suggest a more increased tip per gas, as they hold the tips that users spend and resolve be slightly tilted to conduct transactions with more down tips specified.
- If users like to observe gas costs, then they can send ETH for smaller, user can utilize multiple other tools given below:
- " Etherscan Transaction gas price estimator ", and " Blocknative ETH Gas Estimator Gas " for evaluating Chrome attachment backing both Class 0 inheritance trades and Class 2 EIP-1559 trades, and" ETH Gas Station " which is client instructed metrics for the Ethereum gas demand, " Cryptoneur Gas Fees Calculator " is used for calculating gas prices in the local currency for various trade.

Why do gas fees exist?

- Gas prices assist maintain the Ethereum grid safe. By demanding a price for every analysis performed on the grid, it controls harmful attackers from spamming the grid.
- To bypass unexpected code, every trade needs to set a limitation to the multiple computational efforts of regulation performance.
- The absolute unit of analysis is "gas".
- Moreover, a trade contains a limitation, any gas not operated in a transaction is produced to the user (i.e. max fee – (base fee + tip) is produced).

How does block size affect the base fees?

The base fee is computed by a procedure that resembles the dimensions of the last block (the part of gas utilized for all the trades) with the target size. The base fee will rise by a most of 12.5% per alliance if the target block size is overextended.



SOLIDITY

Solidity is a brand-new programming language created by **Ethereum** which is the second-largest market of cryptocurrency by capitalization, released in the year 2015 and led by Christian Reitwiessner. Some key features of solidity are listed below:

- Solidity is a high-level programming language designed for implementing smart contracts.
- It is a statically typed object-oriented(contract-oriented) language.
- Solidity is highly influenced by Python, c++, and JavaScript which run on the Ethereum Virtual Machine(EVM).
- Solidity supports complex user-defined programming, libraries, and inheritance.
- Solidity is the primary language for blockchains running platforms.
- Solidity can be used to create contracts like voting, blind auctions, crowdfunding, multi-signature wallets, etc.

Ethereum

Ethereum is a decentralized open-source platform based on the blockchain domain, used to run smart contracts i.e. applications that execute the program exactly as it was programmed without the possibility of any fraud, interference from a third party, censorship, or downtime. It serves as a platform for nearly 2,60,000 different cryptocurrencies. Ether is a cryptocurrency generated by Ethereum miners, used to reward for the computations performed to secure the blockchain.

Ethereum Virtual Machine(EVM)

Ethereum Virtual Machine abbreviated as EVM is a runtime environment for executing smart contracts in ethereum. It focuses widely on providing security and execution of untrusted code using an international network of public nodes. EVM is specialized to prevent Denial-of-service attack and confirms that the program does not have any access to each other's state, also ensures that the communication is established without any potential interference.

Smart Contract

Smart contracts are high-level program codes that are compiled to EVM byte code and deployed to the ethereum blockchain for further execution. It allows us to perform credible transactions without any interference of the third party, these transactions are trackable and irreversible. Languages used to write smart contracts are Solidity (a language library with similarities to C and JavaScript), Serpent (similar to Python, but deprecated), LLL (a low-level Lisp-like language), and Mutan (Go-based, but deprecated).

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >= 0.4.16 < 0.9.0;
/// @title A contract for demonstrate how to write a smart contract
/// @author Jitendra Kumar
/// @notice For now,this contract just show to set the value of state variable,calculate the sum and get this sum value from the smart contract

contract Test
{

   // Declaring state variables
   uint public var1;
   uint public var2;
   uint public sum;

   // Defining public function
   // that sets the value of
   // the state variable
   function set(uint x, uint y) public
   {
      var1 = x;
      var2 = y;
      sum=var1+var2;
   }

   // Defining function to
   // print the sum of
   // state variables
   function get(
   ) public view returns (uint) {
      return sum;
   }
```

}



Explanation:

1. Version Pragma:

```
pragma solidity >=0.4.16 <0.9.0;
```

Pragmas are instructions to the compiler on how to treat the code. All solidity source code should start with a "version pragma" which is a declaration of the version of the solidity compiler this code should use. This helps the code from being incompatible with the future versions of the compiler which may bring changes. The above-mentioned code states that it is compatible with compilers of version greater than and equal to 0.4.16 but less than version 0.9.0

2. The contract keyword:

```
contract Test

{

//Functions and Data

}
```

The contract keyword declares a contract under which is the code encapsulated.

3. State variables:

```
uint public var1;

uint public var2;

uint public sum;
```

State variables are permanently stored in contract storage that they are written in Ethereum Blockchain. The line *uint public var1* declares a state variable called var1 of type uint (unsigned integer of 256 bits). Think of it as adding a slot in a database. Similarly, goes with the declaration *uint public var2* and *uint public sum*.

4. A function declaration:

```
function set(uint x, uint y) public

function get() public view returns (uint)
```

- This is a function named *set* of access modifier type *public* which takes a variable *x* and variable y of datatype *uint* as a parameter.
- This was an example of a simple smart contract that updates the value of var1 and var2. Anyone can call the function set and overwrite the value of var1 and var2 which is stored in the Ethereum blockchain. This is an example of a decentralized application that is censorship-proof and unaffected to the shutdown of any centralized server. As long as someone is running a single node of the Ethereum blockchain, this smart contract will be accessible.
- The variable sum is calculated by adding the values of the variables var1 and var2.
- Function get will retrieve and print the value of the state variable sum.

How to Execute The Code:

There are practically two ways to execute a solidity smart contract:

1. Offline Mode: Running a solidity smart contract in Offline mode requires three prerequisites and 4 major steps to be followed to get the smart contract running:

- **Prerequisites:**
  - Download and install node.js.
  - Install Truffle globally.
  - Install ganache-cli.
- **Objectives:**
  - Create a truffle project and configure a development network
  - Create and deploy smart contracts
  - Interact with the smart contract from Truffle console
  - Write tests for testing the main features offered by Solidity.

2. Online Mode: Remix IDE is generally used to compile and run Solidity smart contracts in the Online Mode