What is Ethereum?

Ethereum is a blockchain-based computing platform that enables developers to build and deploy decentralized applications—meaning not run by a centralized authority. You can create a decentralized application for which the participants of that particular application are the decision-making authority.

Ethereum Features

- Ether: This is Ethereum's cryptocurrency.
- Smart contracts: Ethereum allows the development and deployment of these types of contracts.
- Ethereum Virtual Machine: Ethereum provides the underlying technology—the architecture and the software—that understands smart contracts and allows you to interact with it.
- Decentralized applications (Dapps): A decentralized application is called a Dapp (also spelled DAPP, App, or DApp) for short. Ethereum allows you to create consolidated applications, called decentralized applications.
- Decentralized autonomous organizations (DAOs): Ethereum allows you to create these for democratic decision-making.

These are Ethereum's essential features. Before going deep into the Ethereum tutorial, let's discuss each of these features in more detail.

1. Ether

Ether (ETH) is Ethereum’s cryptocurrency. It is the fuel that runs the network. It is used to pay for the computational resources and the transaction fees for any transaction executed on the Ethereum network. Like Bitcoins, ether is a peer-to-peer currency. Apart from being used to pay for transactions, ether is also used to buy gas, which is used to pay for the computation of any transaction made on the Ethereum network.

Also, if you want to deploy a contract on Ethereum, you will need gas, and you would have to pay for that gas in ether. So gas is the execution fee paid by a user for running a transaction in Ethereum. Ether can be utilized for building decentralized applications, building smart contracts, and making regular peer-to-peer payments.

2. Smart Contracts

Smart contracts are revolutionizing how traditional contracts work, which is why you need to use the tutorial to become more familiar with them. A smart contract is a simple computer program that facilitates the exchange of any asset between two parties. It could be money, shares, property, or any other digital asset that you want to exchange. Anyone on the Ethereum network can create these contracts. The contract consists primarily of the terms and conditions mutually agreed on between the parties (peers).

The smart contract’s primary feature is that once it is executed, it cannot be altered, and any transaction done on top of a smart contract is registered permanently—it is immutable. So even if you modify the smart contract in the future, the transactions correlated with the original contract will not get altered; you cannot edit them.

The verification process for the smart contracts is carried out by anonymous parties in the network without the need for a centralized authority, and that’s what makes any smart contract execution on Ethereum a decentralized execution.

The transfer of any asset or currency is done in a transparent and trustworthy manner, and the identities of the two entities are secure on the Ethereum network. Once the transaction is successfully done, the accounts of the sender and receiver are updated accordingly, and in this way, it generates trust between the parties.

Smart Contracts Vs. Traditional Contract Systems

In conventional contract systems, you sign an agreement, then you trust and hire a third party for its execution. The problem is that in this type of process, data tampering is possible. With smart contracts, the agreement is coded in a program.

A centralized authority does not verify the result; it is confirmed by the participants on the Ethereum blockchain-based network. Once a contract is executed, the transaction is registered and cannot be altered or tampered, so it removes the risk of any data manipulation or alteration.

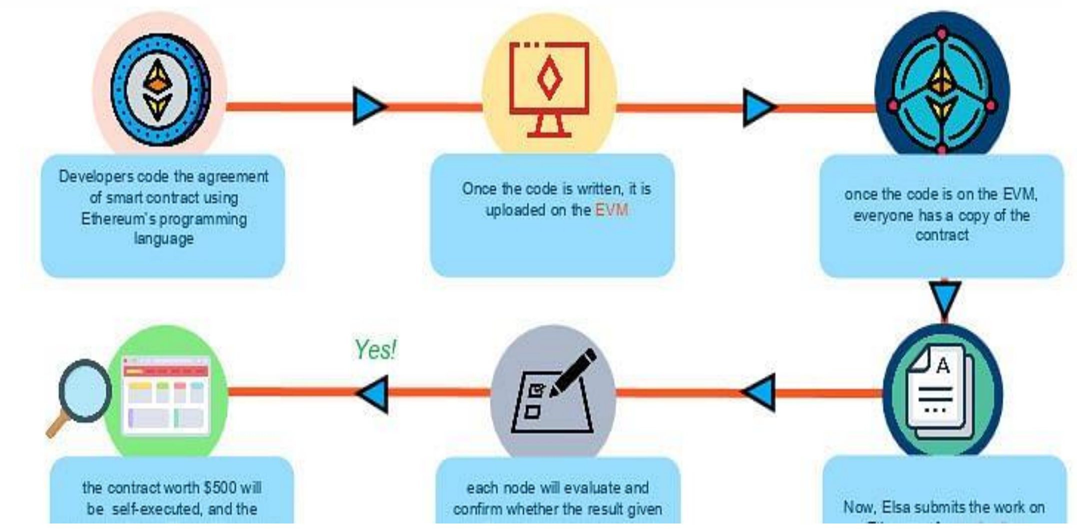
Let’s take an example in which someone named Zack has given a contract of $500 to someone named Elsa for developing his company’s website. The developers code the agreement of the smart contract using Ethereum’s programming language.

The smart contract has all the conditions (requirements) for building the website. Once the code is written, it is uploaded and deployed on the Ethereum Virtual Machine (EVM).

EVM is a runtime compiler to execute a smart contract. Once the code is deployed on the EVM, every participant on the network has a copy of the contract. When Elsa submits the work on Ethereum for evaluation, each node on the Ethereum network will evaluate and confirm whether the result given by Elsa has been done as per the coding requirements.

Once the result is approved and verified, the contract worth $500 will be self-executed, and the payment will be paid to Elsa in ether. Zack's account will be automatically debited, and Elsa will be credited with $500 in ether.

The Ethereum tutorial video includes a demo on the deployment of an Ethereum smart contract.


Developers code the agreement of smart contract using Ethereum's programming language
Once the code is written, it is uploaded on the EVM
once the code is on the EVM, everyone has a copy of the contract
Now, Elsa submits the work on
each node will evaluate and confirm whether the result given
Yes!
the contract worth $500 will be self-executed, and the

Take a deep dive on Bitcoins, Hyperledger, Ethereum, and Multichain Blockchain platforms with the Blockchain Certification Training Course!

3. Ethereum Virtual Machine

EVM, as mentioned above in this Ethereum tutorial, is designed to operate as a runtime environment for compiling and deploying Ethereum-based smart contracts. EVM is the engine that understands the language of smart contracts, which are written in the Solidity language for Ethereum. EVM is operated in a sandbox environment—basically, you can deploy your stand-alone environment, which can act as a testing and development environment. You can then test your smart contract (use it) “n” number of times, verify it, and once you are satisfied with the performance and the functionality of the smart contract, you can deploy it on the Ethereum main network.

Any programming language in the smart contract is compiled into the bytecode, which the EVM understands. This bytecode can be read and executed using the EVM. Solidity is one of the most popular languages for writing a smart contract. Once you write your smart contract in Solidity, that contract gets converted into the bytecode and gets deployed on the EVM, thereby guaranteeing security from cyberattacks.

a) How Does EVM Work?

Suppose person A wants to pay person B 10 ethers. The transaction will be sent to the EVM using a smart contract for a fund transfer from A to B. To validate the transaction; the Ethereum network will perform the proof-of-work consensus algorithm.

The miner nodes on Ethereum will validate this transaction—whether the identity of A exists or not, and if A has the requested amount to transfer. Once the transaction is confirmed, the ether will be debited from A’s wallet and will be credited to B’s wallet, and during this process, the miners will charge a fee to validate this transaction and will earn a reward.

All the nodes on the Ethereum network execute smart contracts using their respective EVMs.

b) Proof of Work

Every node in the Ethereum network has:

- The entire history of all the transactions—the entire chain
- The history of the smart contract, which is the address at which the smart contract is deployed, along with the transactions associated with the smart contract
- The handle to the current state of the smart contract

The goal of the miners on the Ethereum network is to validate the blocks. For each block of a transaction, miners use their computational power and resources to get the appropriate hash value by varying the nonce. The miners will vary the nonce and pass it through a hashing algorithm—in Ethereum, it is the Ethash algorithm.

This produces a hash value that should be less than the predefined target as per the proof-of-work consensus. If the hash value generated is less than the target value, then the block is considered to be verified, and the miner gets rewarded.

When the proof of work is solved, the result is broadcast and shared with all the other nodes to update their ledger. If other nodes accept the hashed block as valid, then the block gets added to the Ethereum main blockchain, and as a result, the miner receives a reward, which as of today stands at three ethers. Plus, the miner gets the transaction fees that have been generated for verifying the block. All the transactions that are aggregated in the block—the cumulative transaction fees associated with all the transactions are also rewarded to the miner.

c) Proof of Stake

In Ethereum, a process called proof of stake is also under development. It is an alternative to proof of work and is meant to be a solution to minimize the use of expensive resources spent on mining using proof of work. In proof of stake, the miner—who is the validator—can validate the transactions based on the number of crypto coins he or she holds before actually starting the mining.

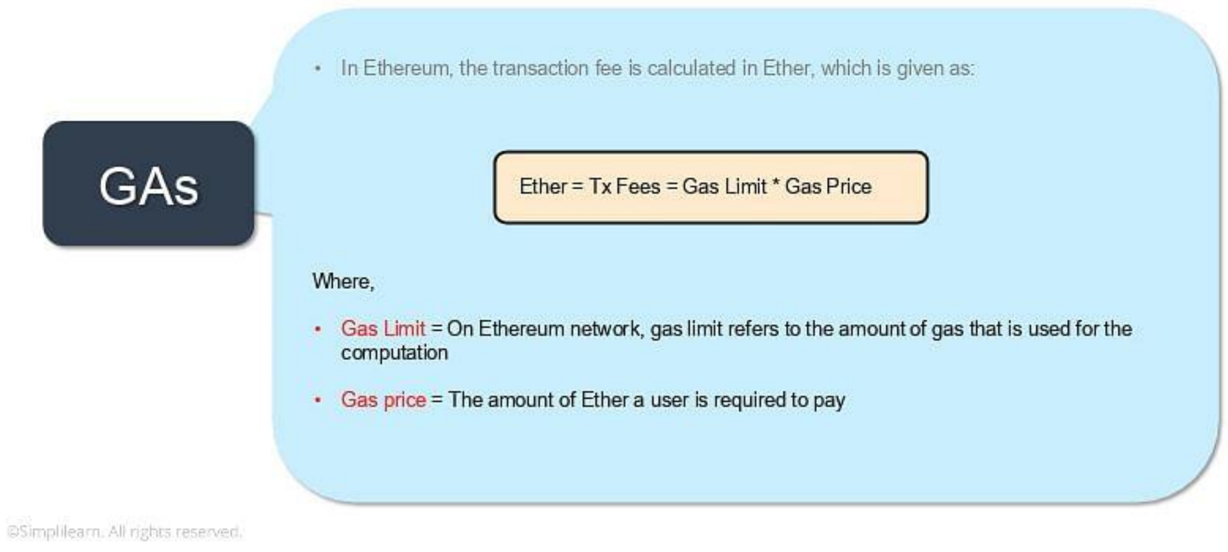
So, based on the accumulation of crypto coins the miner has beforehand, he or she has a higher probability of mining the block. However, proof of stake is not widely used as of now compared to proof of work.

d) Gas

Just like we need fuel to run a car, we need gas to run applications on the Ethereum network. To perform any transaction within the Ethereum network, a user must make a payment, in this case paying out ethers, to get a transaction done, and the intermediary monetary value is called gas.

On the Ethereum network, gas is a unit that measures the computational power required to run a smart contract or a transaction. So, if you must do a transaction that updates the blockchain, you would have to shell out gas, and that gas costs ethers.

In Ethereum, the transaction fees are calculated using a formula (see screenshot below). For every transaction, there is gas and its correlated gas price. The transaction fees equal the amount of gas required to execute a transaction multiplied by the gas price. “Gas limit” refers to the amount of gas used for the computation and the amount of ether a user is required to pay for the gas.



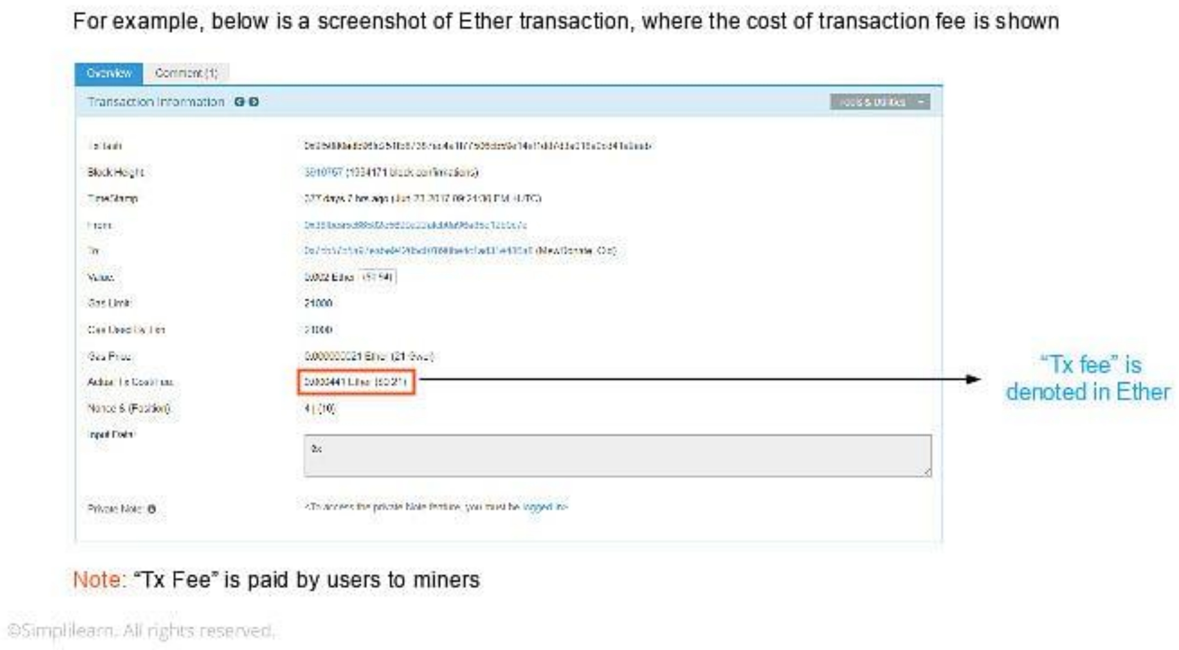
- In Ethereum, the transaction fee is calculated in Ether, which is given as:

Ether = Tx Fees = Gas Limit * Gas Price

Where,

- Gas Limit = On Ethereum network, gas limit refers to the amount of gas that is used for the computation
- Gas price = The amount of Ether a user is required to pay

Below is a screenshot from the Ethereum network showing the transaction cost. You can see for this particular transaction, the gas limit was 21,000, the gas used by the transaction was 21,000, and the gas price was 21 Gwei, which is the lowest denomination of ether. So, 21 Gwei * 21,000 gave the actual transaction fees: 0.000441 ethers, or about 21 cents as of today. As mentioned, the transaction fee goes to the miner, who has validated the transaction.



For example, below is a screenshot of Ether transaction, where the cost of transaction fee is shown

Overview | Comment (1)

Transaction Information

TxHash:	[illegible]
Block Height:	3910767 (1034171 block confirmations)
TimeStamp:	227 days 7 hrs ago (Jun 21 2017 09:24:06 PM +UTC)
From:	[illegible]
To:	[illegible] (MewDonate, Ox)
Value:	0.002 Ether ($1.54)
Gas Limit:	21000
Gas Used By Txn:	21000
Gas Price:	0.000000021 Ether (21 Gwei)
Actual Tx Cost/Fee:	0.000441 Ether ($0.21)
Nonce & {Position}:	4 | {10}
Input Data:	0x
Private Note:	<To access the private Note feature, you must be logged in>

Note: "Tx Fee" is paid by users to miners

To understand the gas limit and price, let's consider an example using a car. Suppose your vehicle has a mileage of 10 kilometers per liter and petrol costs $1 per liter. Under these parameters, driving a car for 50 kilometers would cost you five liters of petrol, which is worth $5. Similarly, to perform an operation or to run code on Ethereum, you need to obtain a certain amount of gas, like petrol, and the gas has a per-unit price, called gas price.