Ethereum Client

An Ethereum client is a software program that is used to implement the Ethereum specification and connect itself with other Ethereum clients over a peer-to-peer network.

Handle tasks such as

Synchronizing with the network:

Connects and validates to peers

Executing smart contracts:

Run Smart contracts

Processing transactions:

Provide RPC access to apps and wallets

Client Types:

1)Execution Layer:

Handles **smart contracts**,**transactions**, and **the Ethereum Virtual Machine.**

Acts as the “brain” that runs code and processes user activity.

Example:GEth, Nethermind, Besu, Erigon

Languages used:Go, C#, Java, Go

2)Consensus Layer (CL):

Manages the **Proof of Stake consensus**,which block is accepted as the next valid one.

Handles **validators**, **staking**, **attestations**, and **finality** (making blocks irreversible).

It does **not** run smart contracts — it just ensures all nodes agree on the same chain head.

Example:Prysm, Lighthouse, Teku, Nimbus, Lodestar

Languages used:Go, Rust, Java, Nim, TypeScript

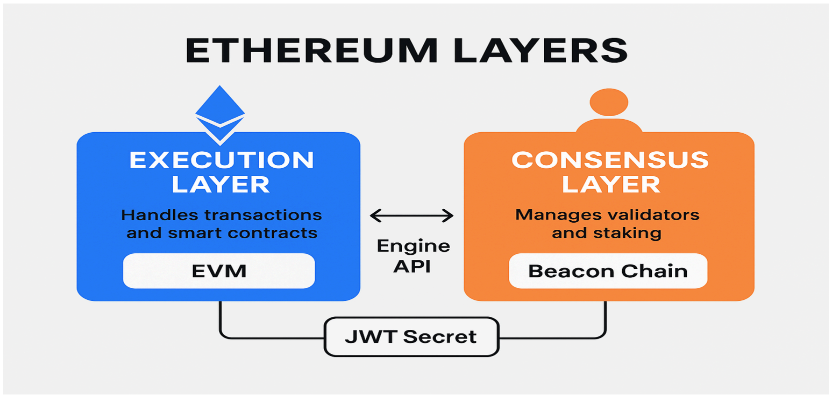
Need of both Clients Execution layer and Consensus layer:

After The Merge:

Execution Layer (e.g., Geth) = runs the Ethereum logic (transactions, contracts).

Consensus Layer (e.g., Prysm) = runs the Beacon Chain and validator consensus.

They talk to each other through a secure **Engine API** (via a JWT secret file).



Example setup:

Geth (EL) ←→ Prysm (CL)

- Geth executes transactions and provides block data.
- Prysm decides which block becomes part of the canonical chain.