POLKADOT
BLOCKCHAIN
ACADEMY

Introduction to Substrate

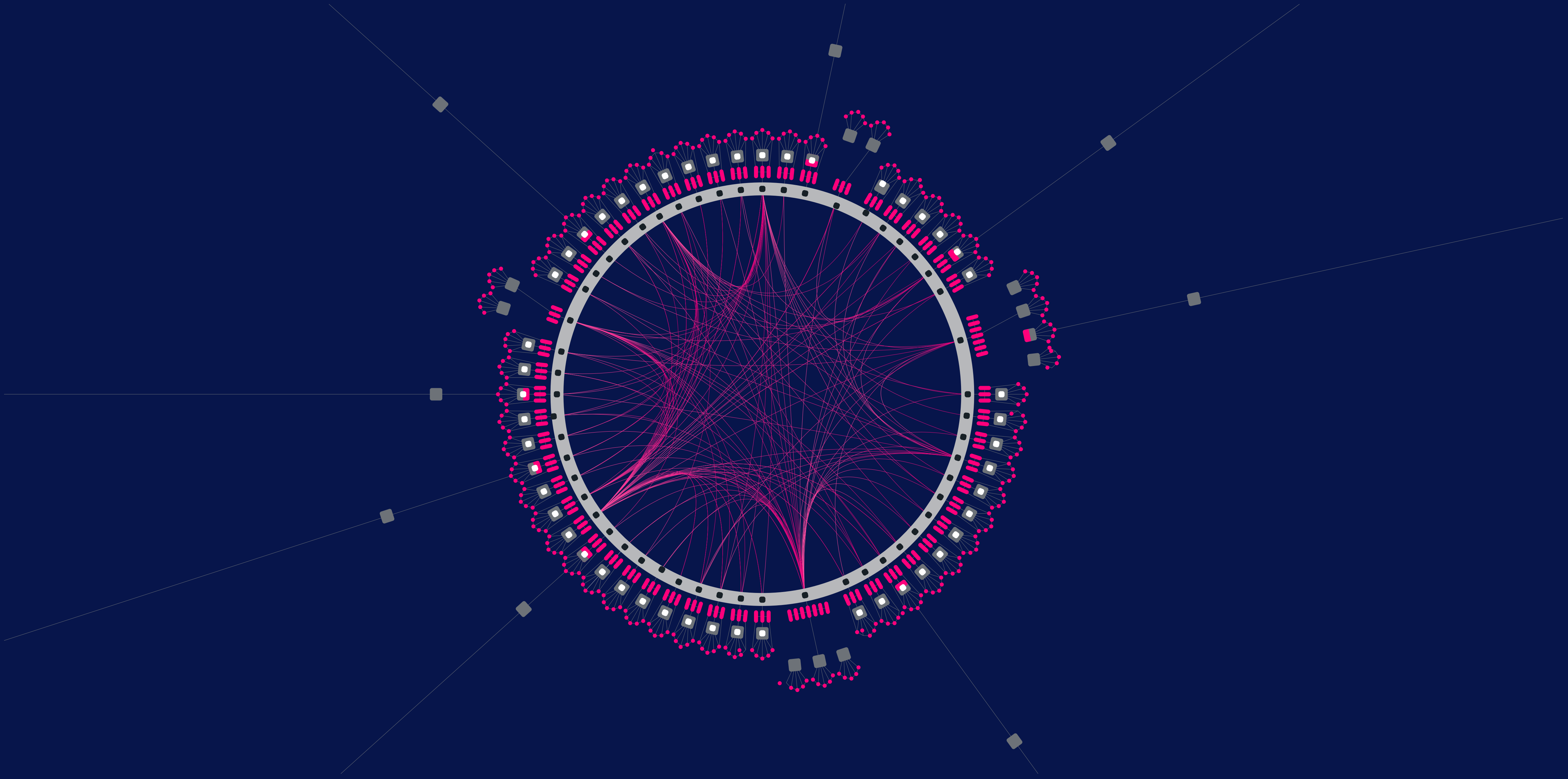
What is Substrate?

Substrate is a Rust framework for building modular and extensible blockchains.

Why Substrate?

Building a blockchain is hard.
Like... really hard.

The Multi-Chain Future

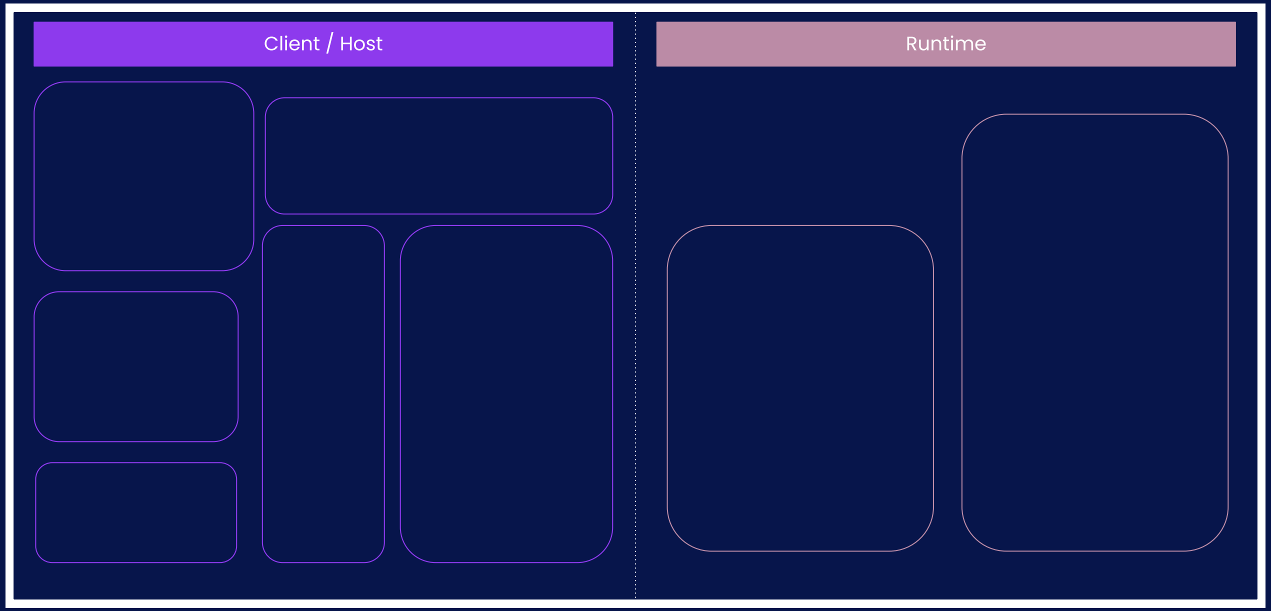


Substrate Architecture

At a very high level, a Substrate node has two parts:

The Client - A Wasm executor.

A Wasm runtime.



What is the Runtime?

The runtime contains all of the business logic for executing the **state transition function** of the blockchain.

Turing-Complete State Machine

- Turing completeness basically means you can implement any computer algorithm.
- Besides limitations from execution time, memory size, or storage limitations... the runtime is a turing-complete state machine.
- The state machine itself is broken into two components:
    - The state itself
    - The state transition function

Runtime as a VM

The Runtime is designed as a Virtual Machine within the Substrate client.

Why?

- Runtime code must execute deterministically.
- Runtime code should be sandboxed.

VMs enable this.

The runtime is always a Wasm binary.

Wasm

- Wasm is short for WebAssembly
- It is a binary instruction format for a stack-based virtual machine.
- Originally built for the web as a faster and better alternative to technologies like JavaScript.
- The open standards for WebAssembly are developed by W3C groups.

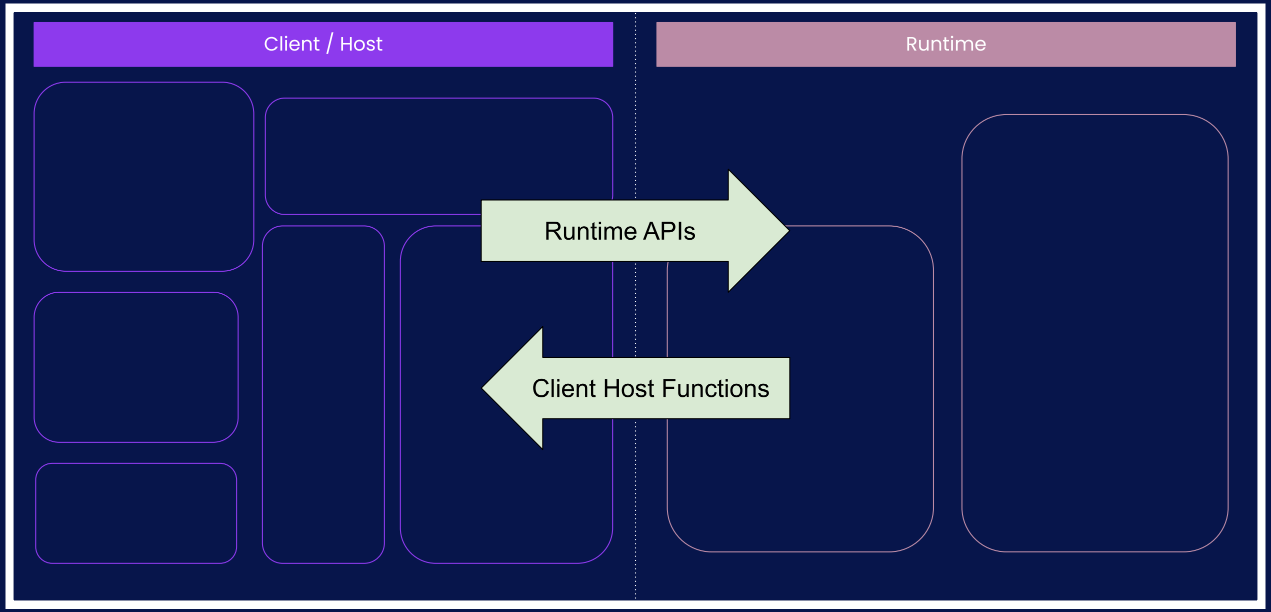
Why Substrate chose Wasm?

- Compact: Designed to be easily transferred over the web.
- Sandboxable: Keeps Wasm Safe, as capabilities have to be exposed explicitly to the Wasm environment.
- Deterministic(-ish): assuming all outputs are defined given some instruction set.
- Performance: direct mapping of operations to machine code.
- Well Supported: WASM is on its way to become a core component of the web, just like JavaScript did.

Runtime Assumptions

To make a Substrate compatible runtime, our only assumptions are:

- It exposes a specific Runtime APIs.
- It has access to specific client-side host functions.

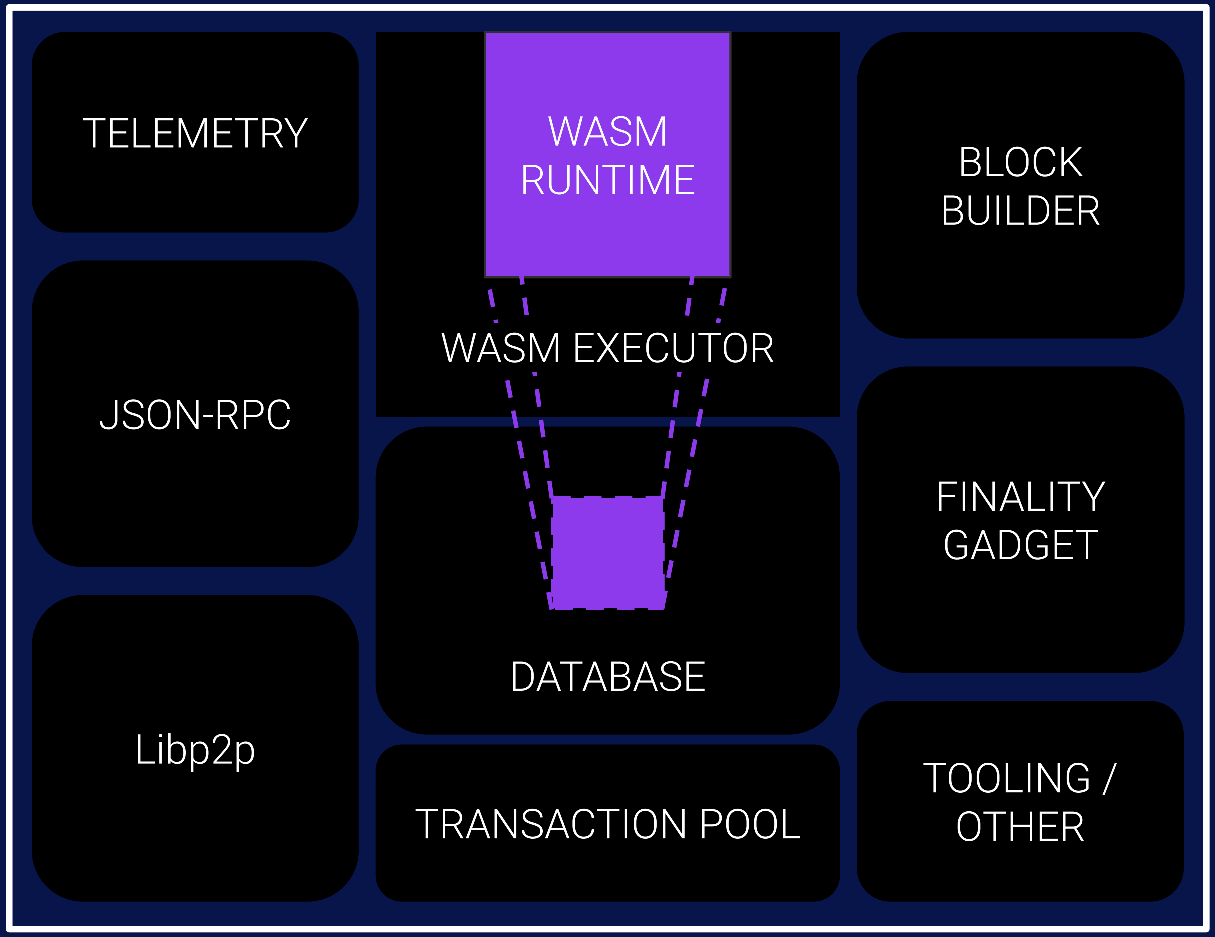


The Client

- This is simply the natively compiled binary which runs on your computer.
- It has access to do much more things then the restrictive Wasm VM environment.
- Generally, determinism is not important at this level.
  - Allows multiple implementations to be created.
  - Allows for compilation to different targets.

Client Components

- Networking
- Database
- Transaction Queue
- Consensus
- Telemetry
- Runtime
- Tools
- And more!

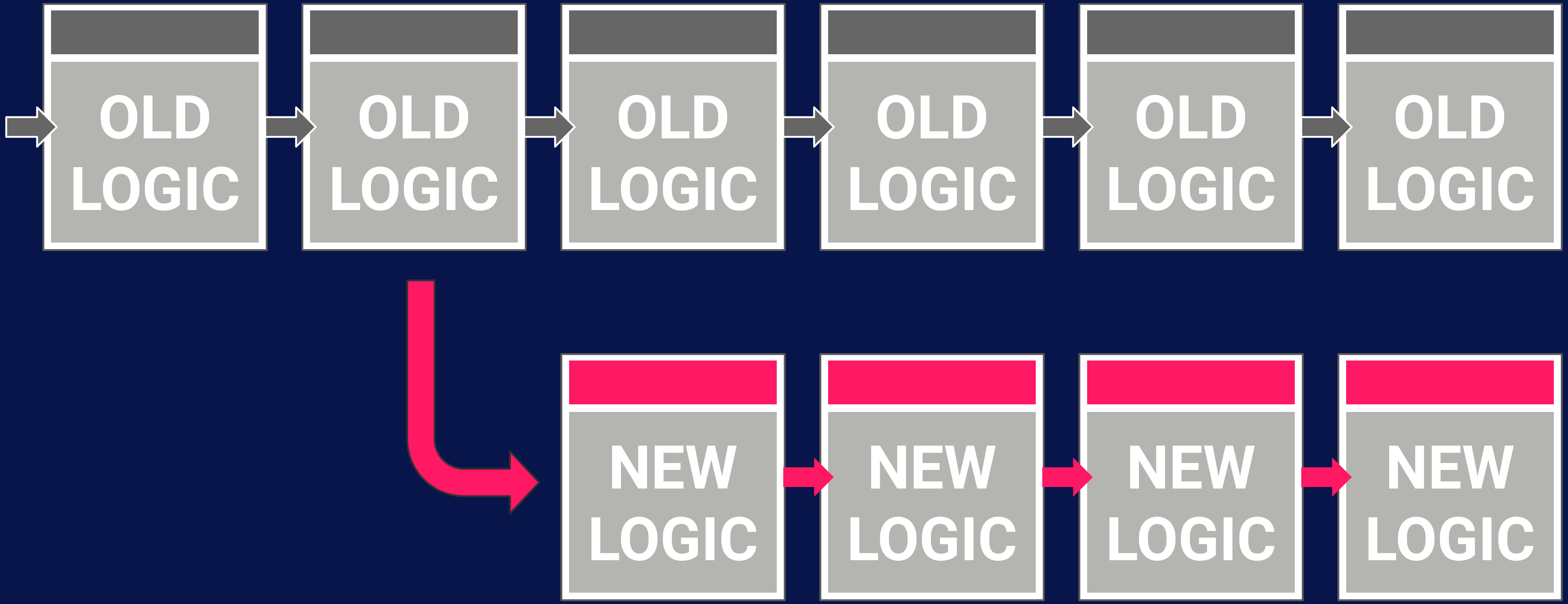


SUBSTRATE CLIENT

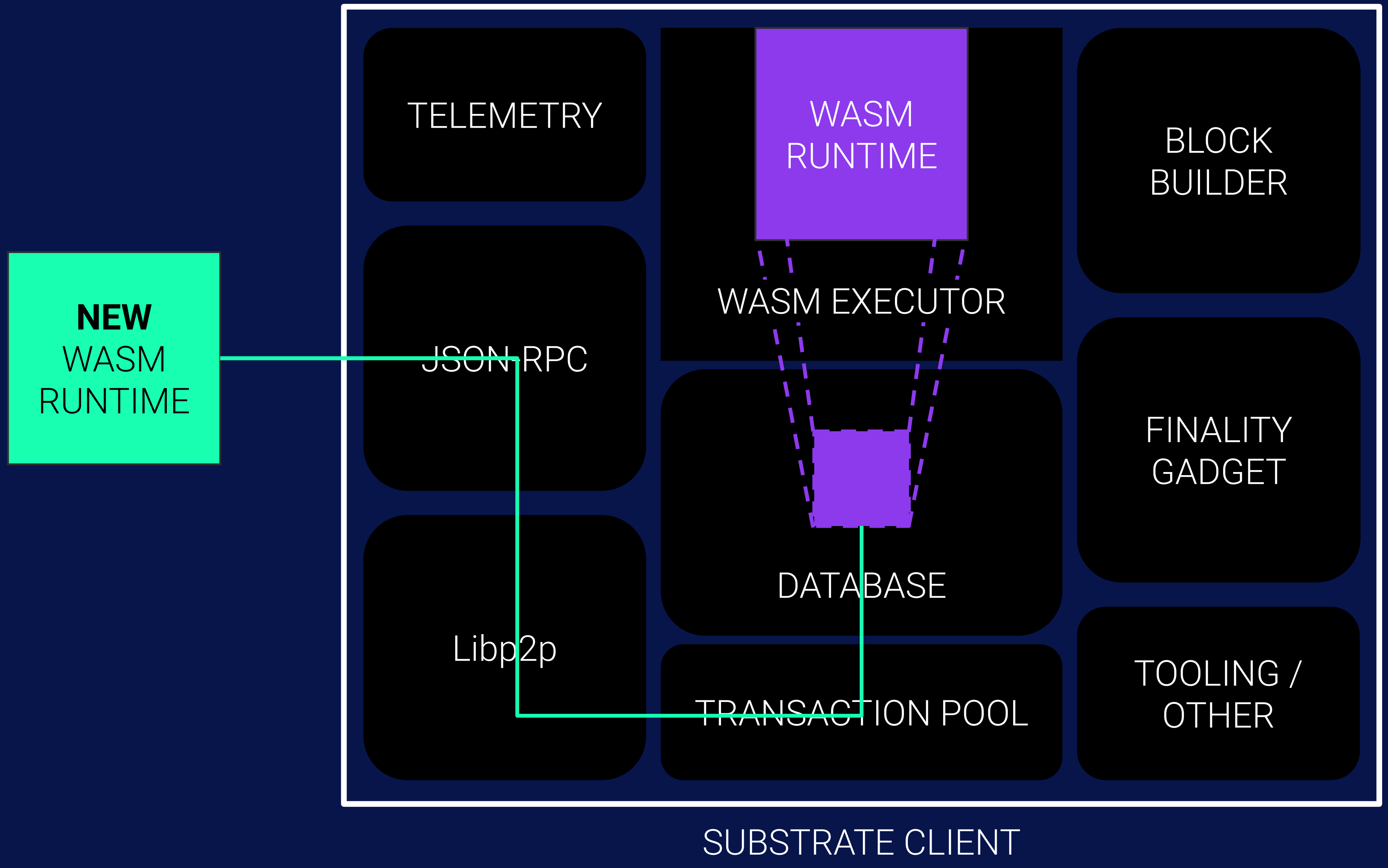
Wasm is stored on chain!

Problems with Hard Forks

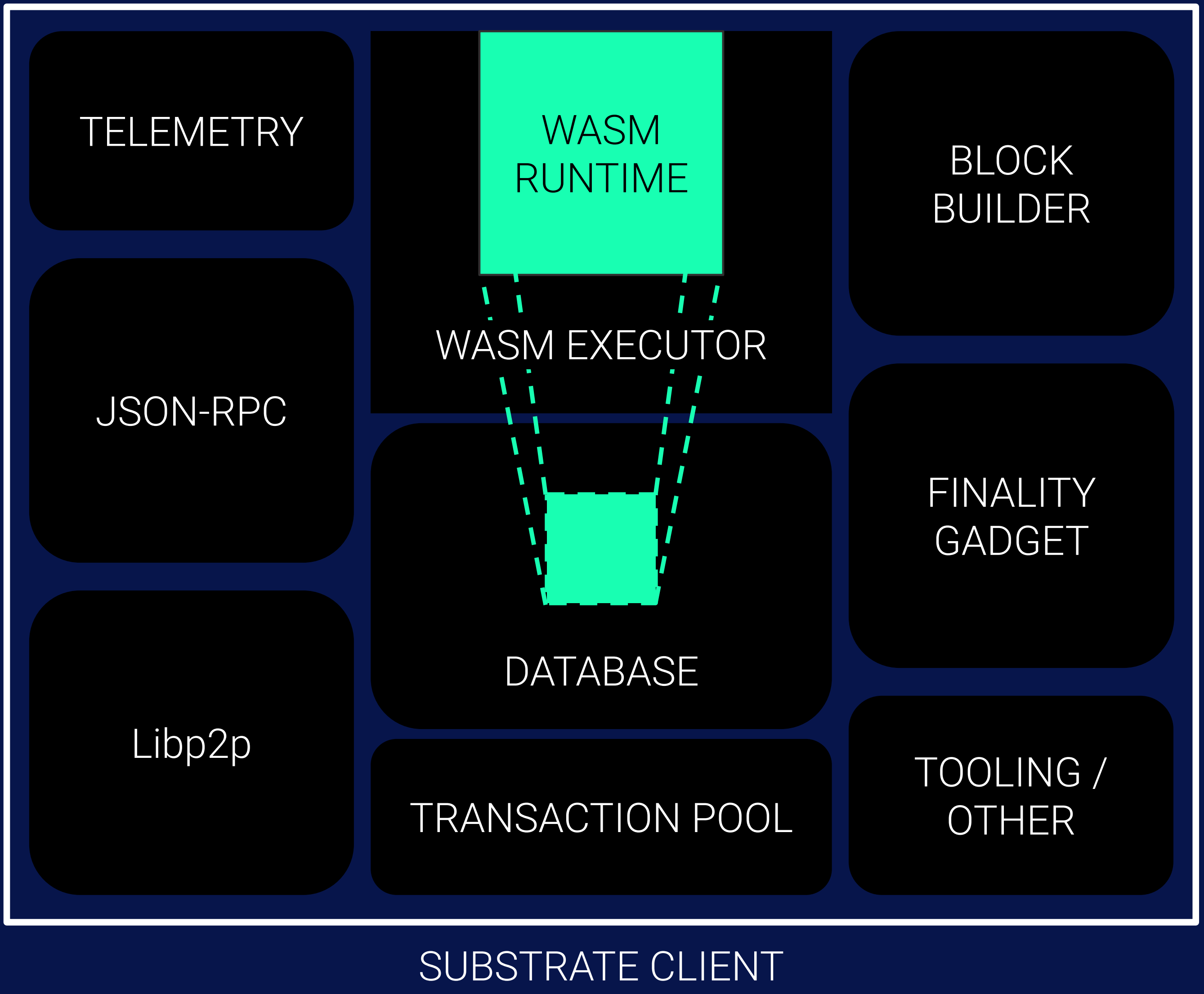
Not everyone updates their client software in time.



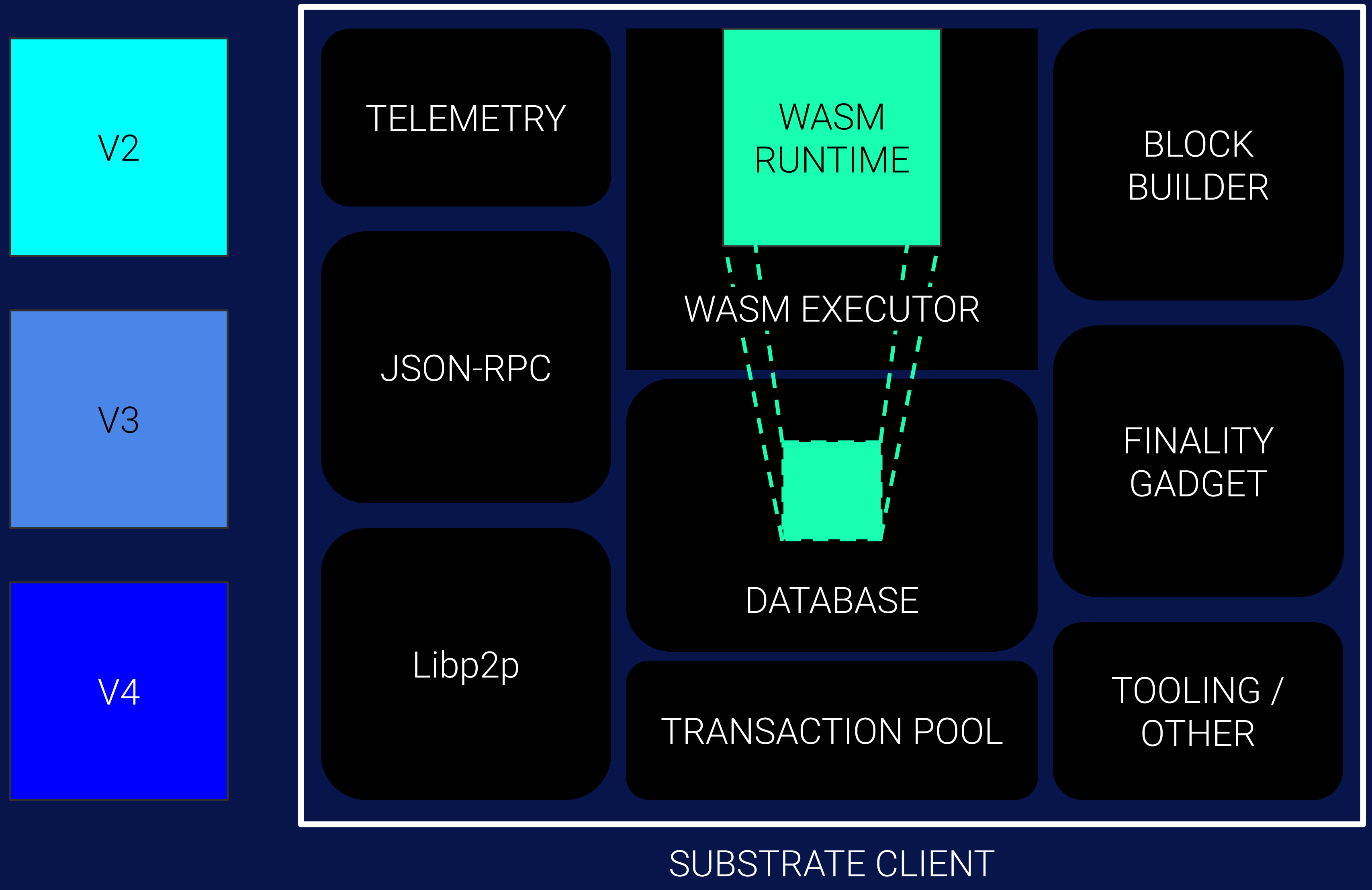
Upgrading the Substrate Runtime



Upgrading the Substrate Runtime



Upgrading the Substrate Runtime

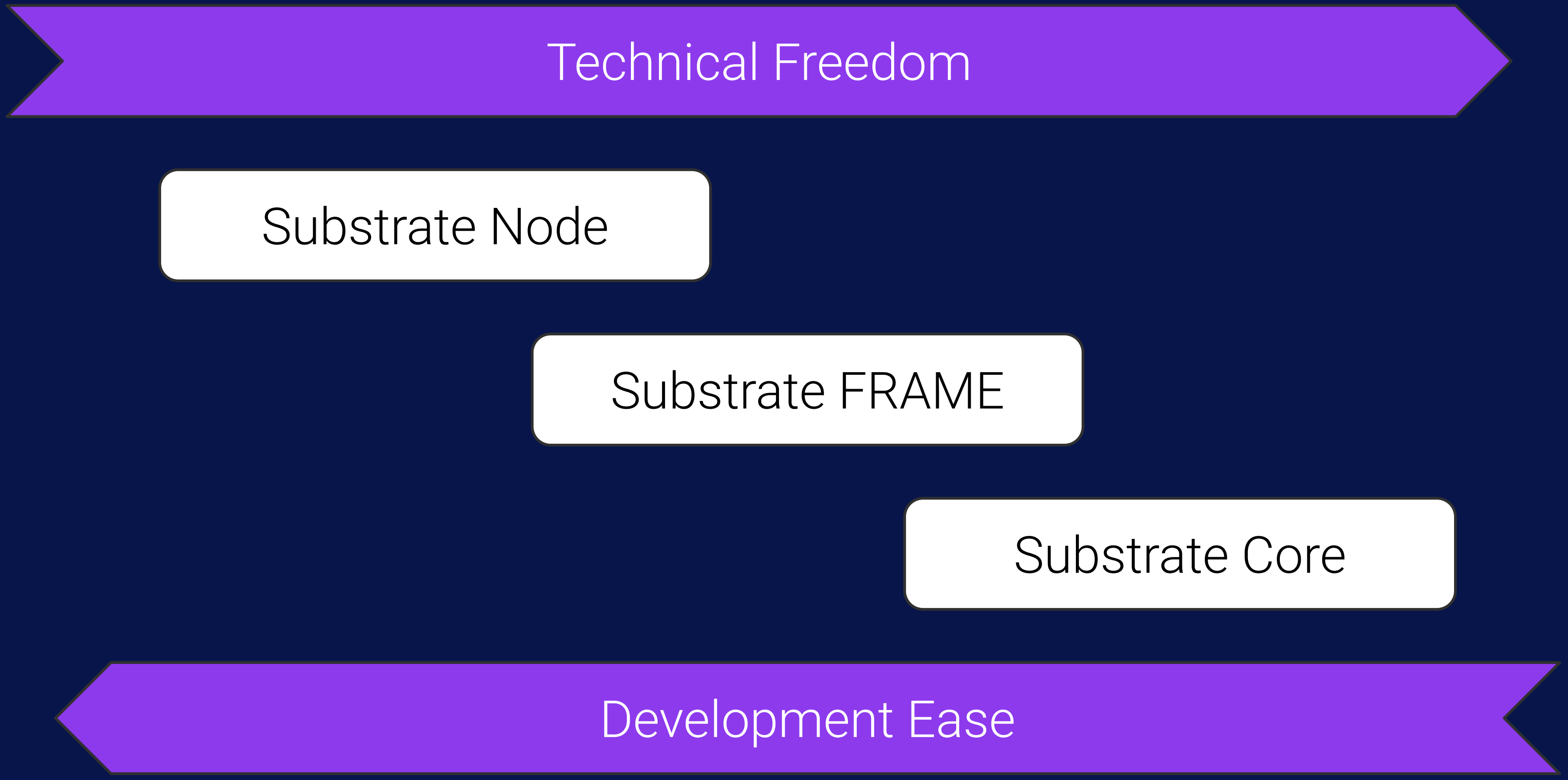


Game Console Analogy

Substrate Client

Substrate Runtime

Technical Freedom vs Ease



License

Substrate Primitives (`sp-*`), Frame (`frame-*`) and the pallets (`pallets-*`), binaries (`/bin`) and all other utilities are licensed under Apache 2.0.
Substrate Client (`/client/*` / `sc-*`) is licensed under GPL v3.0 with a classpath linking exception.

- Apache2 allows teams full freedom over what and how they release, and giving licensing clarity to commercial teams.
- GPL3 ensures any deeper improvements made to Substrate's core logic (e.g. Substrate's internal consensus, crypto or database code) to be contributed back so everyone can benefit.