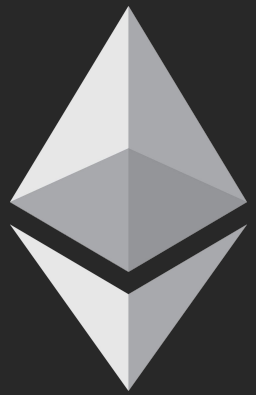
substrate_

web3 foundation

Parity has a lot of blockchain building experience...



github.com/paritytech/
parity-ethereum

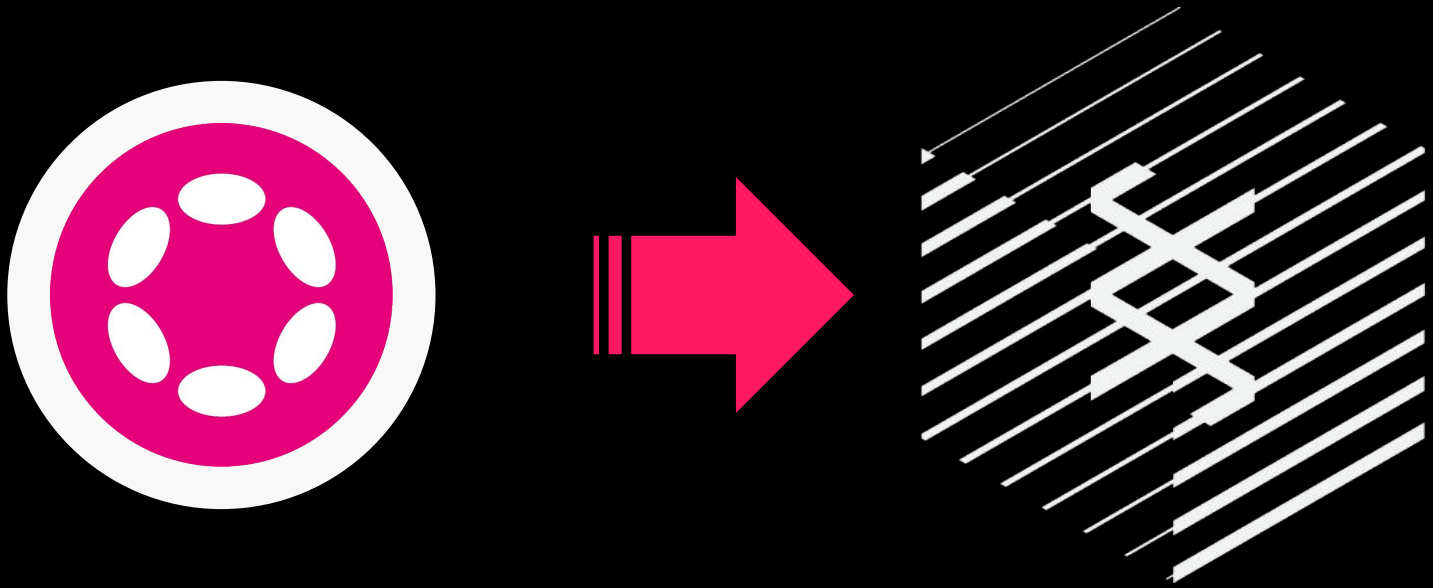
github.com/paritytech/
parity-bitcoin

github.com/paritytech/
polkadot

parity

web3
foundation

From Polkadot, came Substrate.

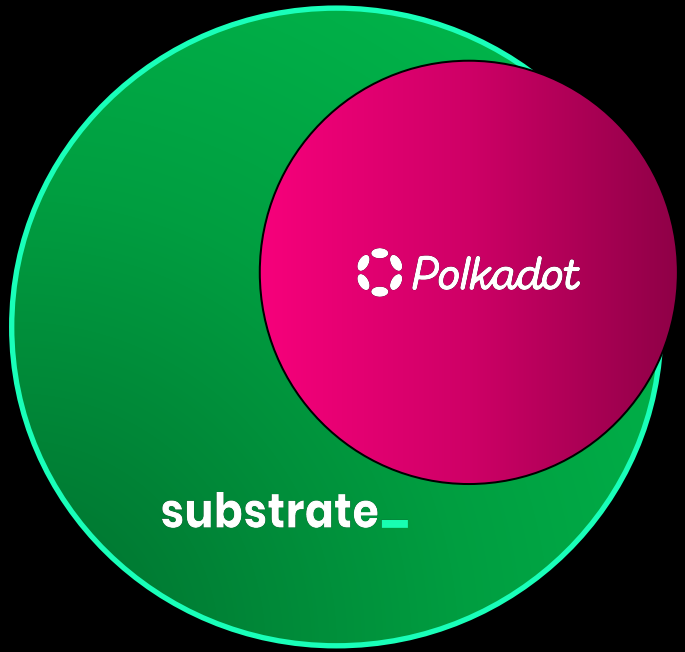


SUBSTRATE

Substrate is a modular, flexible, extensible framework for building blockchains.

Re-use battle-tested libraries while building
the custom components that matter most.

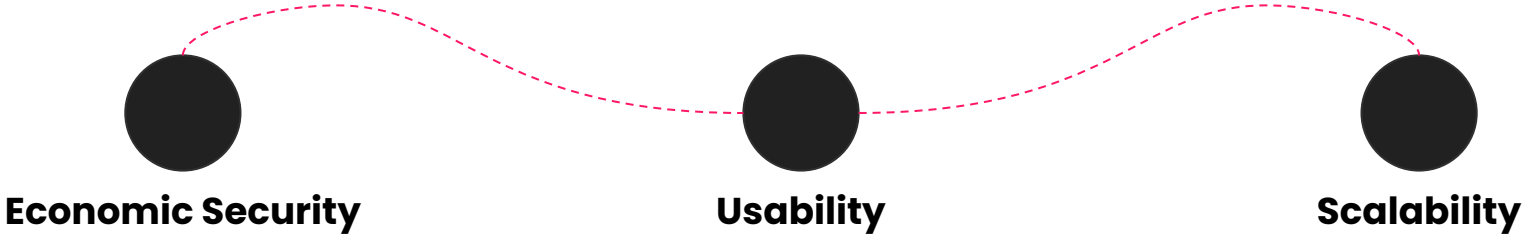
Polkadot and Substrate



EVM != complex innovations

Operation	Gas	Description
ADD/SUB	3	Arithmetic operation
MUL/DIV	5	
ADDMOD/MULMOD	8	
AND/OR/XOR	3	Bitwise logic operation
LT/GT/SLT/SGT/EQ	3	Comparison operation
POP	2	Stack operation
PUSH/DUP/SWAP	3	
MLOAD/MSTORE	3	Memory operation
JUMP	8	Unconditional jump
JUMPI	10	Conditional jump
SLOAD	200	Storage operation
SSTORE	5,000/ 20,000	
BALANCE	400	Get balance of an account
CREATE	32,000	Create a new account using CREATE
CALL	25,000	Create a new account using CALL

Limited Ethereum OpCodes


Economic Security
Usability
Scalability

The case for application specific blockchains

Performance

Single app optimised state machine in contrast to bottleneck virtual machine.

Security

Attack-surface of virtual machine is large (mostly due to complexity).

Sovereignty

Not dependent on platform governance.

The challenges of custom blockchains

Overhead

It takes years and $$$ to build a blockchain from scratch.

Isolation

Blockchains couldn’t talk to each other without going through centralised services.

Security

Blockchains would naturally compete with each other over security resources; leaving each one of them less secure.

substrate_

The Polkadot Stack for a Multichain Future

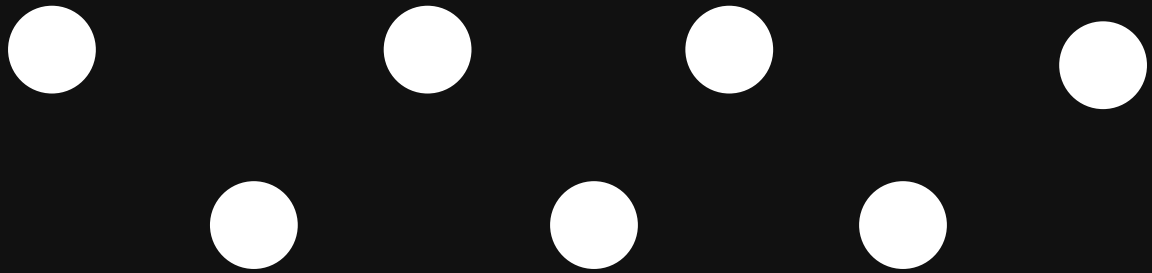
What is Substrate?

Substrate is a framework **for building blockchains,** initially created for building Polkadot.

Modular

Dynamic

Scalable



Focus on business logic and application-specific design, without needing to rewrite complex blockchain modules.

Substrate is modular

GOVERNANCE SYSTEMS

Substrate allows you specify rules for on-chain governance, extending existing modules or building new ones.

ASSETS

Substrate unlocks the next generation of programmable assets and payment systems.

CONSENSUS

Consensus mechanisms can be swapped out for different ones or adapted for specific use-cases.

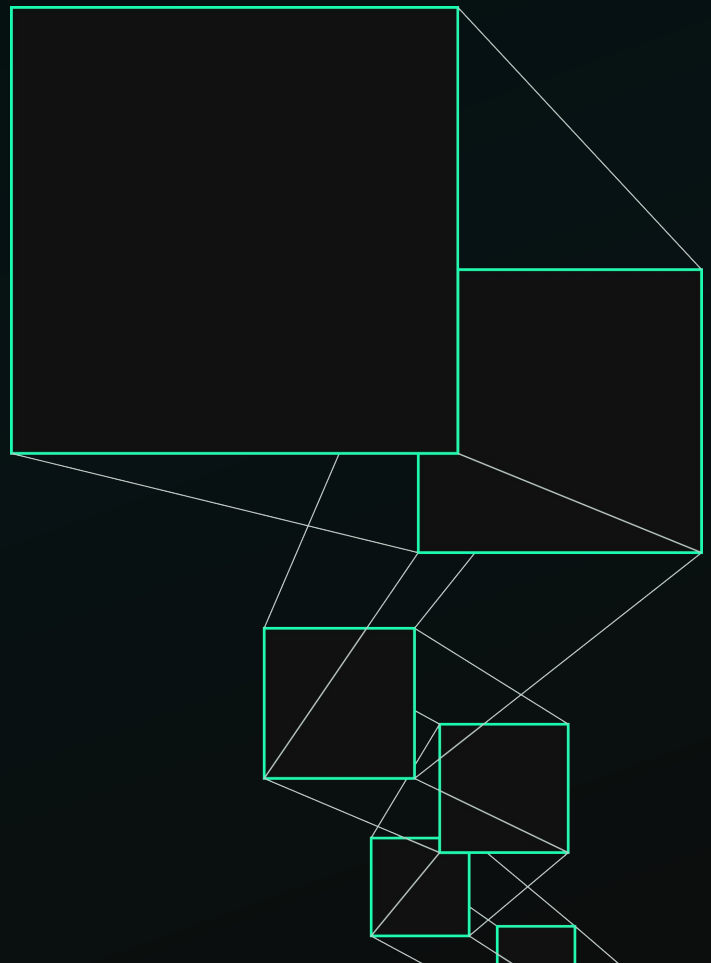
INFRASTRUCTURE

Architectural components can be changes for new ones if needed, from database and networking layers to transaction pool logic.

Substrate is dynamic and adaptable

FORKLESS UPGRADES

Substrate is designed to build blockchains that evolve. If bugs are found in production, Substrate allows enables forkless upgrades.



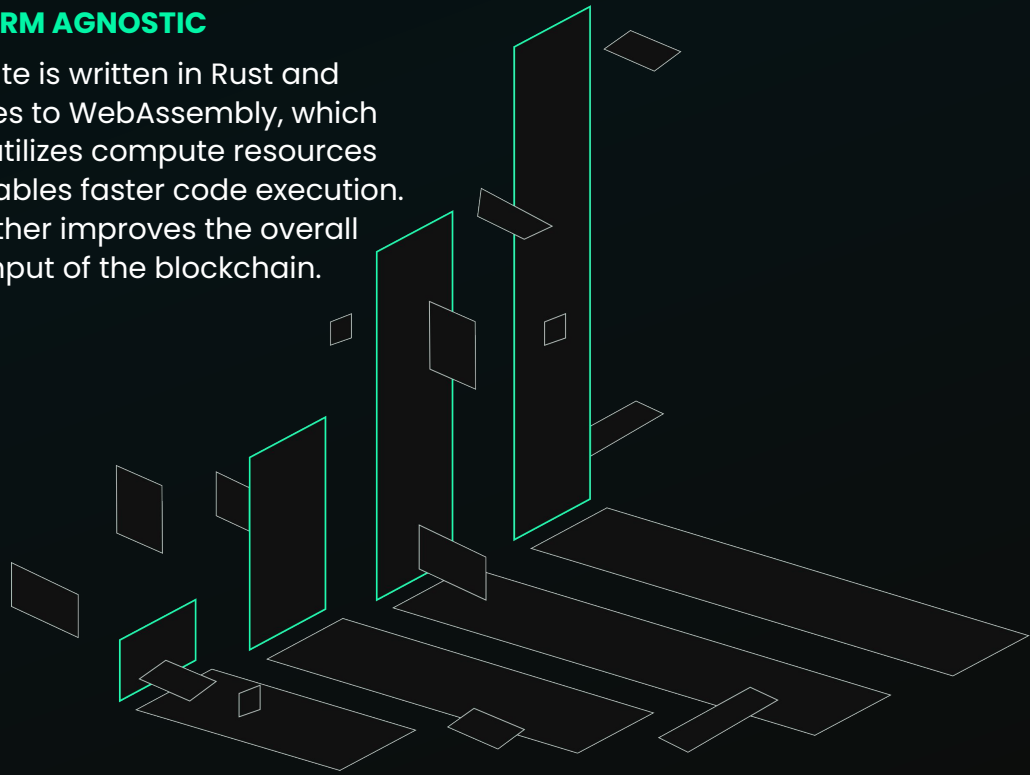
Substrate is built to scale

~1500 TPS

This load test was done on a multi-node, geographically distributed network, with transactions that are non-correlated and independently signed-and-executed.

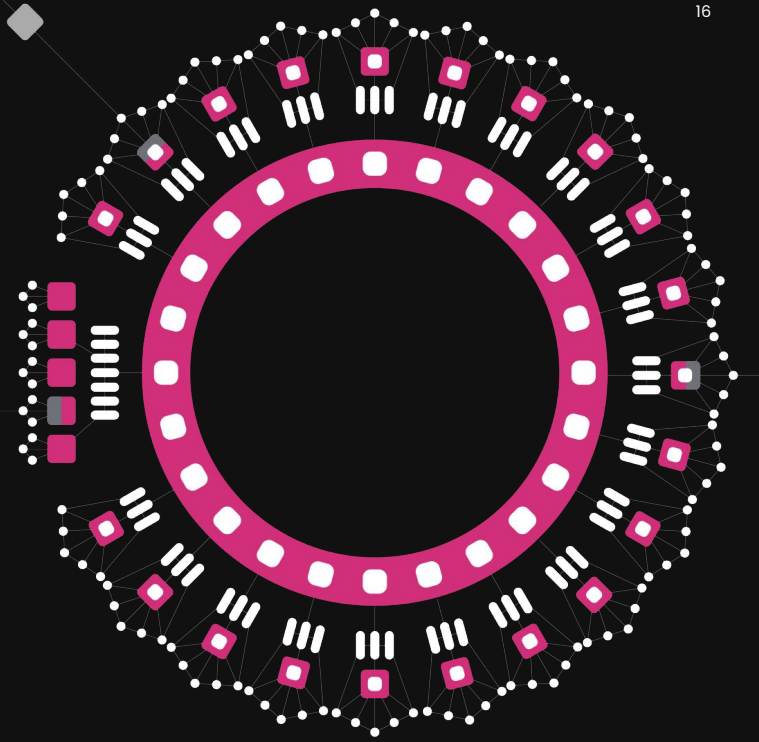
PLATFORM AGNOSTIC

Substrate is written in Rust and compiles to WebAssembly, which better utilizes compute resources and enables faster code execution. This further improves the overall throughput of the blockchain.



What does Substrate provide?

- Database Layer
- Networking Layer
- Consensus Engine
- Transaction Queue
- Library of Runtime Modules



... which are all the core components of a Blockchain.

How it works

Runtime and Storage

State Transition Function

The runtime defines possible state transitions and APIs around them.

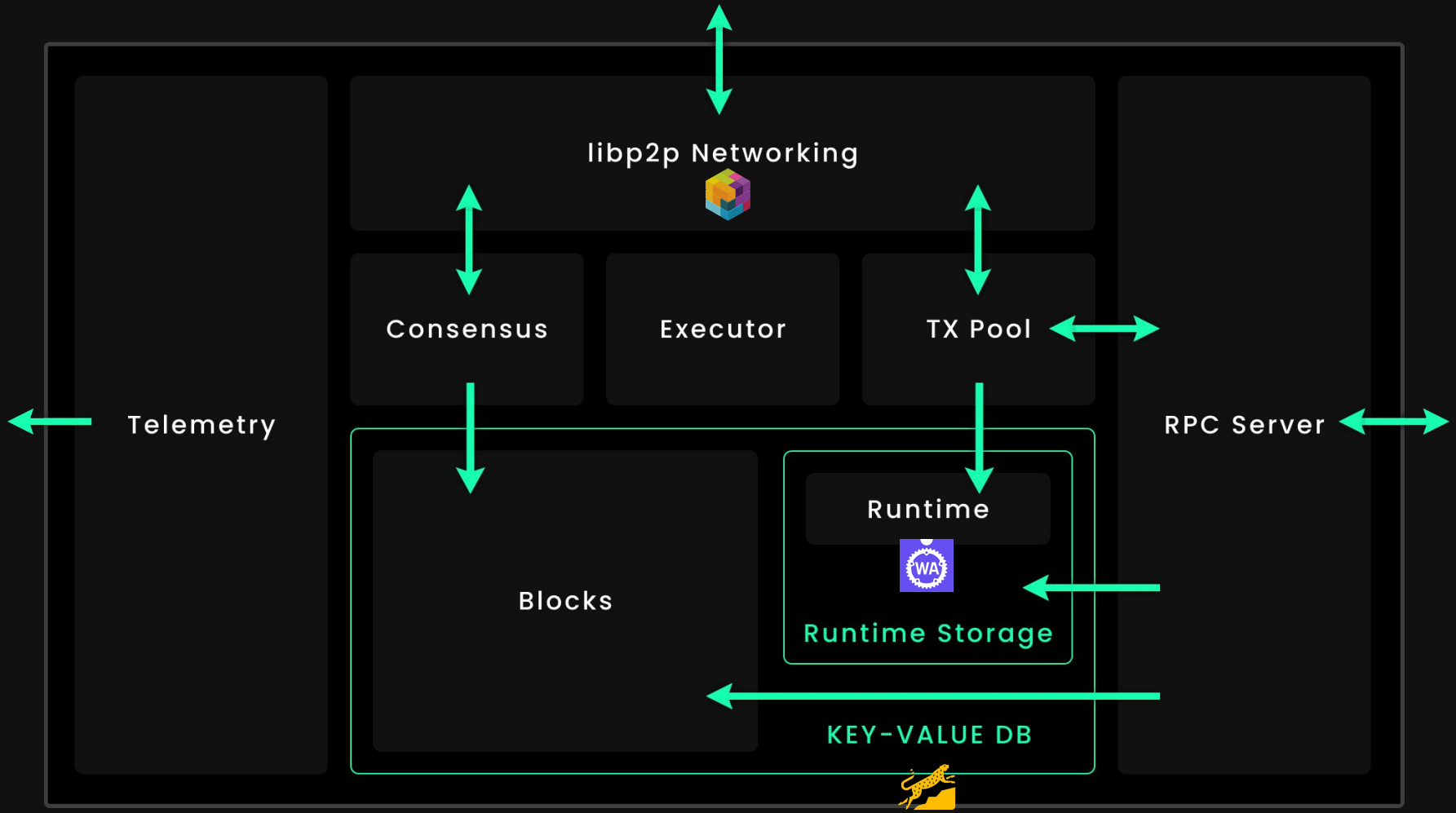
Persistence

State transitions persist to storage, using Rocks DB's key-value storage layer.

Upgradability

The runtime compiles to Wasm, resulting in a byte-code blob containing all storage and STF logic.

The definition of the runtime is an element in storage.


libp2p Networking
Consensus
Executor
TX Pool
Telemetry
RPC Server
Blocks
Runtime
WA
Runtime Storage
KEY-VALUE DB

FRAME Pallets

Framework for Runtime Aggregation of Modularized Entities

Substrate runtimes are made of a collection of Rust modules we call **“pallets”**.

- Simplifies runtime development.
- Enables composability.

SUBSTRATE FRAME PALLETS

Aura	BABE	GRANDPA	Elections
Utility	Atomic Swap	Sudo	Multisig
Identity	Assets	Contracts	EVM
Collective	Treasury	Elections Phragmen	Democracy
Randomness	Timestamp	Staking	and more...

RUNTIME

Aura	GRANDPA	Sudo	Assets
Collective	Treasury	Elections Phragmen	Timestamp

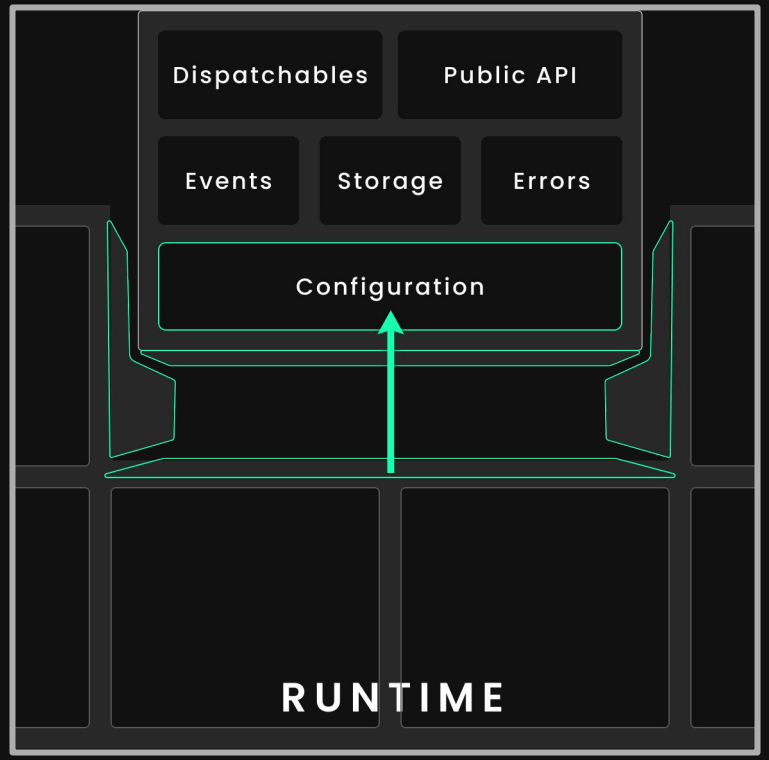
How FRAME works

0—Config

The Configuration trait for all FRAME pallets. Allows developers to configure their pallet by specifying the parameters and types on which it depends.

1—Storage, Events & Errors

Provides an interface to runtime storage, as well as emitting runtime events and errors.



2—Dispatchables

Allows users to make calls into the runtime, such as transactions and other types of extrinsics.

Substrate features a rich set of tools for all stages of your blockchain journey.

With the tools available in the Substrate ecosystem, you can build, deploy and monitor blockchains, smart contracts, and even user interfaces! Here is a quick overview of some of these tools:

BUILD BLOCKCHAINS

- Substrate Framework
- Substrate Node Template
- Benchmarking Framework
- Ink! Smart Contracts
- Frame Pallets
- Zombie Net

BUILD CLIENT APPS

- Polkadot JS API & Apps
- Parity Scale Codec
- Substrate Connect
- Substrate Archive
- Browser Extension
- Signer App
- RPC Libraries

DEPLOY

- End-to-End Deployment Solution
- Docker and Kubernetes Support
- Polkadot Secure Validator (Infra as a code)

MONITOR

- Network Telemetry
- Network Analytics Aggregator
- Built-in Instrumentation
- Prometheus Metrics Support
- Grafana dashboards and visualisation
- Validator Alert System

Developer community

Builder community & Rust developer talent pool

Cross-chain collaboration on tackling market verticals

Language Choice

Substrate is built in Rust, but the runtime is in Wasm
Theoretically, any language that compiles to Wasm can be used
Currently there is a project working to provide support for runtime development in Go
  -> Potentially others in the future (AssemblyScript, C++)

Comparative overview

corda

COSMOS SDK

polygon

Comparison

	Ethereum	Hyperledger Besu	Hyperledger Fabric	Corda R3	Cosmos SDK	Polygon	Substrate
Language	EVM bytecode (Solidity, Vyper)	EVM bytecode (compatible w/ Ethereu	Go, Node (Javascript), Java	JVM (Java, Kotlin, etc.)	Go	EVM bytecode (Solidity, Vyper)	Rust (Go in development)
Granularity	Smart contract	Smart contract	Chain code (similar to smart contracts)	CorDapps (similar to smart contracts)	Runtime (not upgradeable w/o fork)	Smart contracts	Runtime (w/ forkless upgrades)
Sybil resistance / Block production	PoS	PoA	PoA	PoA	PoS	PoA on top of Ethereum PoS	Customizable (out-of-box support for PoA, PoS or PoW)
Permissions	Public	Public, permissioned, or private	Public, permissioned, or private	Private	Public, permissioned, or private	Public or permissioned	Public, permissioned, or private

Comparison

	Ethereum	Hyperledger Besu	Hyperledger Fabric	Corda R3	Cosmos SDK	Polygon	Substrate
Runtime Upgrades	Hard fork	Hard fork	Hard fork	Hard fork	Hard fork	Hard fork	Forkless (runtime code on-chain)
Governance	Off-chain coordir	Off-chain coordination	Off-chain coordination	Chain-customizable, but major changes require off-chain coordina	Chain-customizable, but major changes require off-chain coordin	Off-chain coordir	On-chain (can be configured)
Enactment	Opt-in	Opt-in	Opt-in	Opt-in	Opt-in	Opt-in	Automated
Interoperability	Smart contract-based bridges	Smart contract-based bridges	Smart contract- based	Under development	IBC	Via smart contra and L1->L2 asse movement	XCM (arbitrary code execution, asset movement fungible and non-fungible assets)

Questions?_