

Introductions

- Who are you
- What blockchain experience do you have
- What students get out of training
- Unique facts about PolkaDot and Substrate

Course Overview

Blockchain Overview

Defining the technology, tools and how they work

- Kusama: Introduction
- Polkadot vs. Kusama vs. Rococo vs. Westend
- DOT, KSM and their use cases
- Understanding the Polkadot Architecture
- Interoperability and parachians
- BABE and GRANDPA
- Nominated proof of stake
- On-chain governance
- Shared security
- Transaction fees
- On-chain treasury
- Polkadot Host
- Forkless upgrades
- Bridges
- Parathreads
- Common good parachains
- Parachain slot auctions
- Parachain crowdloans
- PolkadotJS

Cross-Consensus Messaging(XCM)

- Defined
- Format and use

BUILDING

- Getting started: overview of building options, Substrate intro
- Intro to Substrate
- Intro to PolkadotJS
- Apps UI, extension
- Architecture
- Setting up your environment
- Substrate Node template

•

FRAME

- Storage
- Events and errors
- Origins and calls
- Pallet coupling
- Metadata
- Scale codec
- Testing
- Weights
- Benchmarking
- Polkadot JS API

Hands on Workshop

Blockchain Defined

"Blockchain is a shared, immutable ledger for recording transactions, tracking assets and building trust." - [IBM](#)

"Cryptocurrencies like Bitcoin and Ethereum are powered by a technology called the blockchain. At its most basic, a blockchain is a list of transactions that anyone can view and verify." - [Coinbase](#)

"Blockchain technology is an advanced database mechanism that allows transparent information sharing within a business network. A blockchain database stores data in blocks that are linked together in a chain. The data is chronologically consistent because you cannot delete or modify the chain without consensus from the network. As a result, you can use blockchain technology to create an unalterable or immutable ledger for tracking orders, payments, accounts, and other transactions. The system has built-in mechanisms that prevent unauthorized transaction entries and create consistency in the shared view of these transactions." - [Amazon AWS](#)

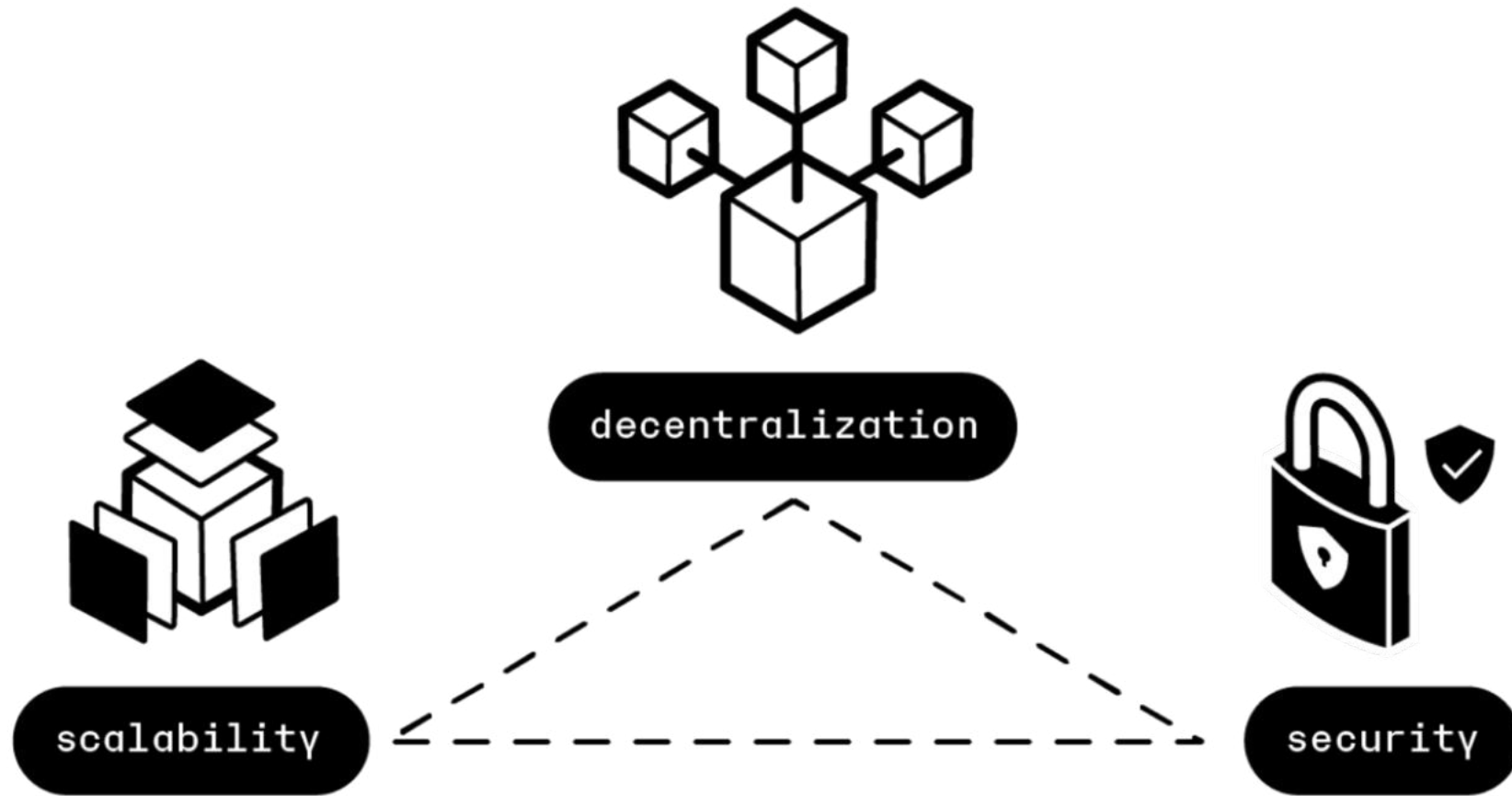
Why is Digital money so difficult to achieve?

Double Spend problem

Mitigate the risk of trust

Reconciliation issues

Blockchains Feature Trilemma: Decentralized, Scalable, Secure

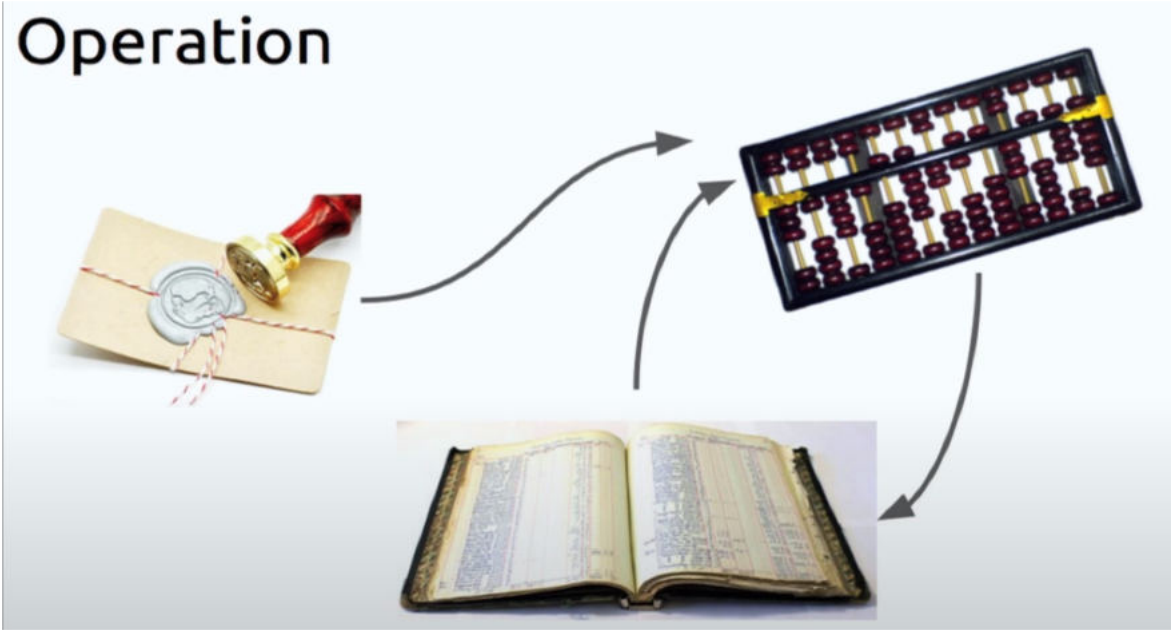


Types of Blockchains

- Public Blockchain networks
- Private Blockchain networks
- Permissioned Blockchain networks
- Consortium Blockchain networks

Why Blockchain

Operation



Authority



Baked in Not Bolted on in blockchain
“Less Trust | More Truth” -Parity Labs

Why Blockchain?

Trust

- trust protocol not participants. Protocols modeled for adversarial environments.

Security

- Data accuracy is required.
- Validated transactions are immutable & permanent.

Efficiency

- Near real time reconciliation
- Built in Auditing
- Auto Execution of state rules

Optional Activity:

Substrate:Run your Own Blockchain

run a local blockchain

Digital mitigation of trust

When communicating digitally how do we know who we are communicating with?

What methods can be used to verify identity and state of something?

How do we secure messaging and communications?

How do you mitigate the risk of trust digitally?

The answer to all of these questions is quite simple: Cryptography!

Cryptography: One way functions, Hash functions vs Cryptographic Hash

One-way function(Trapdoor): $y=f(x)$ given y , it's computationally infeasible to calculate x but given x , it's possible to calculate y .

Hash Function: Function that accepts some arbitrary input and returns a fixed-length digest or a summary value. Loads of usages(Hash map, data distribution, nearest neighbor...)

Cryptographic Hash: One way and collision resistant hash functions. They're hiding yet they're puzzle friendly.

Money, Commodity Money and Fiat

Money should do 3 things

- Store Value
- Medium of exchange
- Unit of account

Commodity Money: Money backed by some commodity of value

Fiat Money: Relies on the reputation of the issuer to maintain value.
Many times this is a country or nation state.

Digital migration to blockchain prior to bitcoin

Digicash: Chaum Signatures

E-Gold: G&S reserves inc. Instant transfer of precious metal backed value at fractional levels.

Hashcash: Back, Proof of work to verify funds. eliminate spam.

B-Money: Dai, Use Computers to generate money outside of governments.

Bit Gold: Szabo, project goal: Make gold digital.

Bitcoin: A Peer-to-Peer Electronic Cash System

Decentralized Ledger

Proof Of Work Consensus

Node Rewards for blocks created

Code dictates the rules of the ledger

Limited state changes.

Not a general purpose computer. No Smart Contracts at this level

What Bitcoin Did?

- Ledger has cryptographic methods backed into the architecture
- Intent of hardening the data set
- Baked-in Cryptography
- pseudonymous founder or founders
- Solved the Double Spend problem(Byzantine General)
- Deflationary Monetary Supply

Blockchain & Distributed Consensus

Distributed Consensus Protocol: There are n nodes that each have an input value. Some of the nodes are faulty or bad actors. The protocol should have the following:

- Value must have derived from an honest node
- come to agreement with other honest nodes on the value or update in state.

Ethereum: Next Generation Blockchain

Ledger and Compute

Turing complete virtual machine: EVM

EVM provides for logic and data that can be stored and ran via an instrument called a “smart contract”

Consensus

Proof-of-Work to start Migrated to Proof of Stake

Founders include: Gavin Wood, Charles Hoskinson, Joe Lubin and Vitalik Buterin

Decentralized compute that anyone can use?

How to mitigate the risk of Denial of Service or over requesting?

Gas

What's Gas?

Payment for computation and storage based on opp codes

Smart Contracts: Defined

Digital Contracts stored on Blockchain that can auto execute code based on conditions

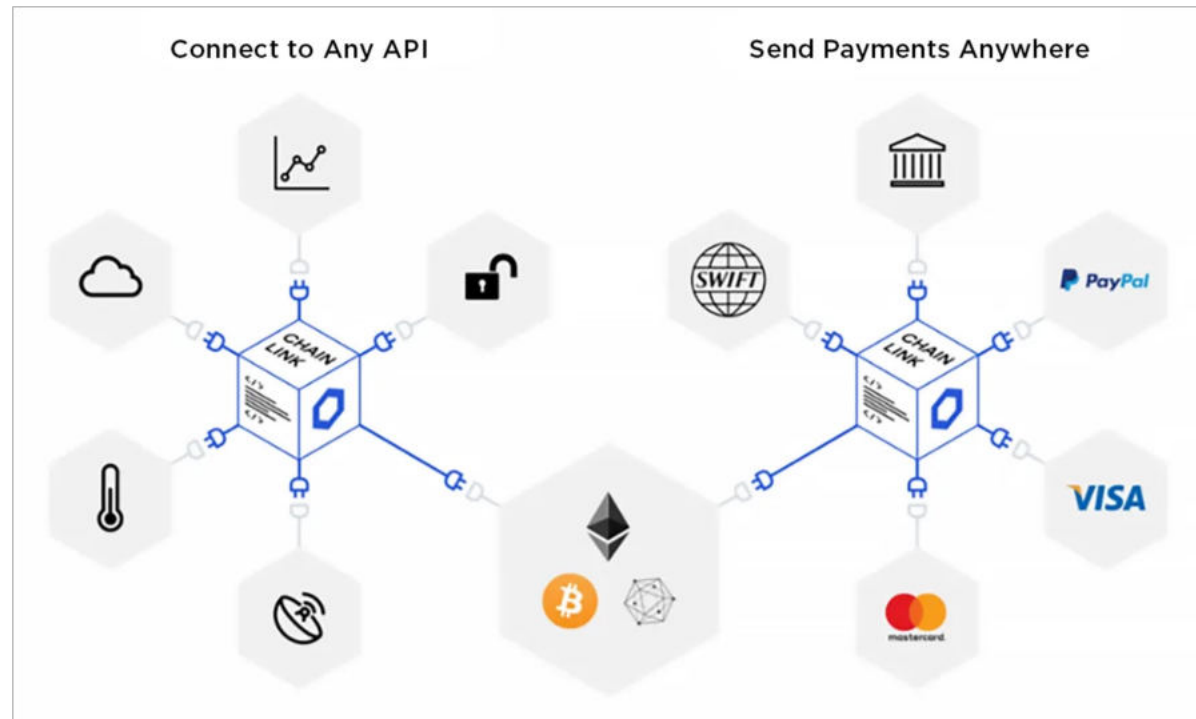
Contracts have all the characteristics of a blockchain

- Decentralized
- Immutable
- Cryptographic Primitives baked in
- Autonomy in the code
- Psudanomy in the users identity
- Transparency in the state and the history of the transactions

DAO's & Oracles

DAO: Decentralized Autonomous Organizations

Oracle: Trusted Authority of reporting of information



Blockchains & Interoperability

Single Blockchains:

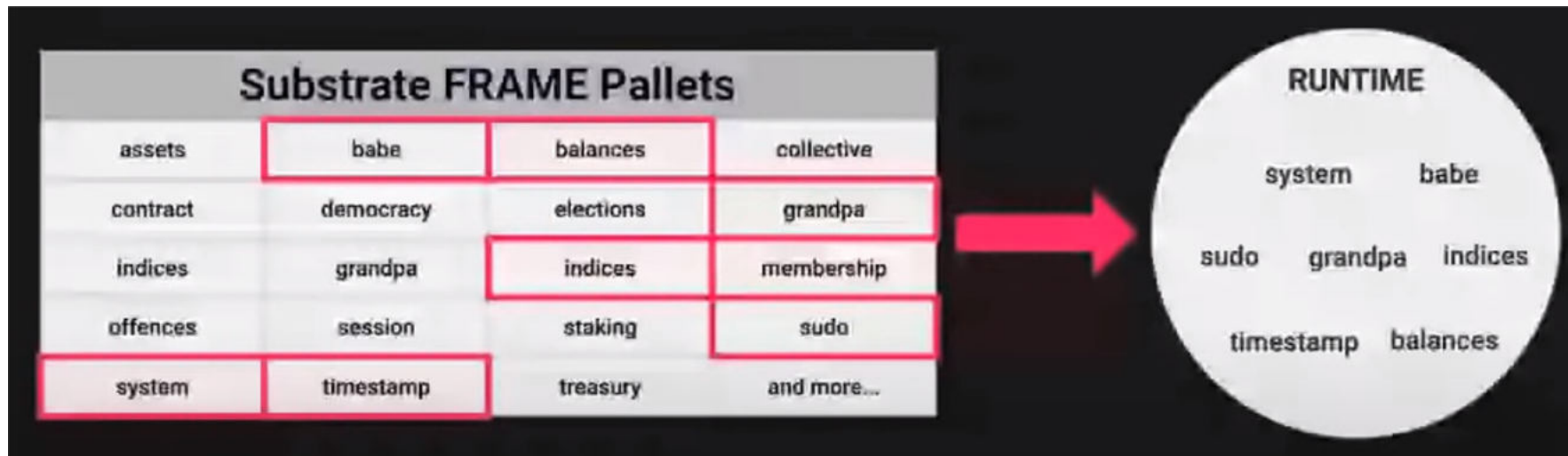
Do one thing well(Bitcoin)

do a lot of things okay(Ethereum)

Polkadot connects a network of **Custom-built blockchains** into one

- gaming
- IoT
- Identity
- Music

These **Custom-built blockchains** are built with a tool called substrate that provides the framework by leveraging pallets into their runtime systems.



Blockchains and Governance

Common Methods and Networks

Little Governance: Changes happen very seldom if ever	Bitcoin
One Person or small group decides: Usual founders	Ethereum
Community proposals and votes	Polkadot

PolkaDot: Referendum, Dot holder contributions

Propose a public referendum	Prioritize public referenda	Vote on all active referenda	Vote for council members	Become a council member
-----------------------------	-----------------------------	------------------------------	--------------------------	-------------------------

Pluggable security & Self-upgrades

Systems built with substrate can plug into existing security systems and network.

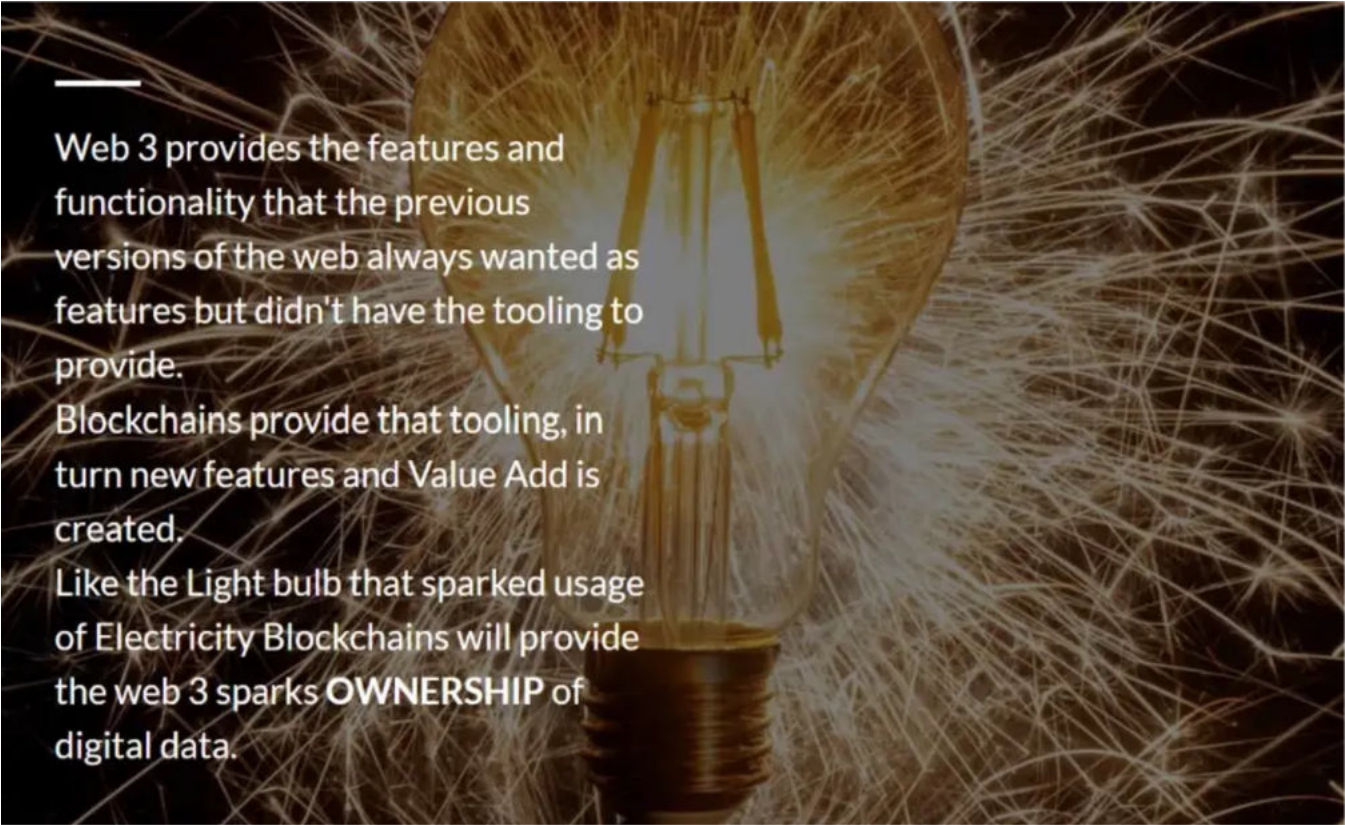
Eliminates the need to build a network of validators.

Self Upgrades provide future proofing



Web 3

<https://medium.com/@tommycooksey/the-progression-of-the-web-web1-to-web3-4f4733c99f2d>



Web 3 provides the features and functionality that the previous versions of the web always wanted as features but didn't have the tooling to provide.

Blockchains provide that tooling, in turn new features and Value Add is created.

Like the Light bulb that sparked usage of Electricity Blockchains will provide the web 3 sparks **OWNERSHIP** of digital data.

Web 2 stack

Application: Facebook, Email, Browser, youtube

TCP/IP SMTP HTTP STREAMING BROWSER

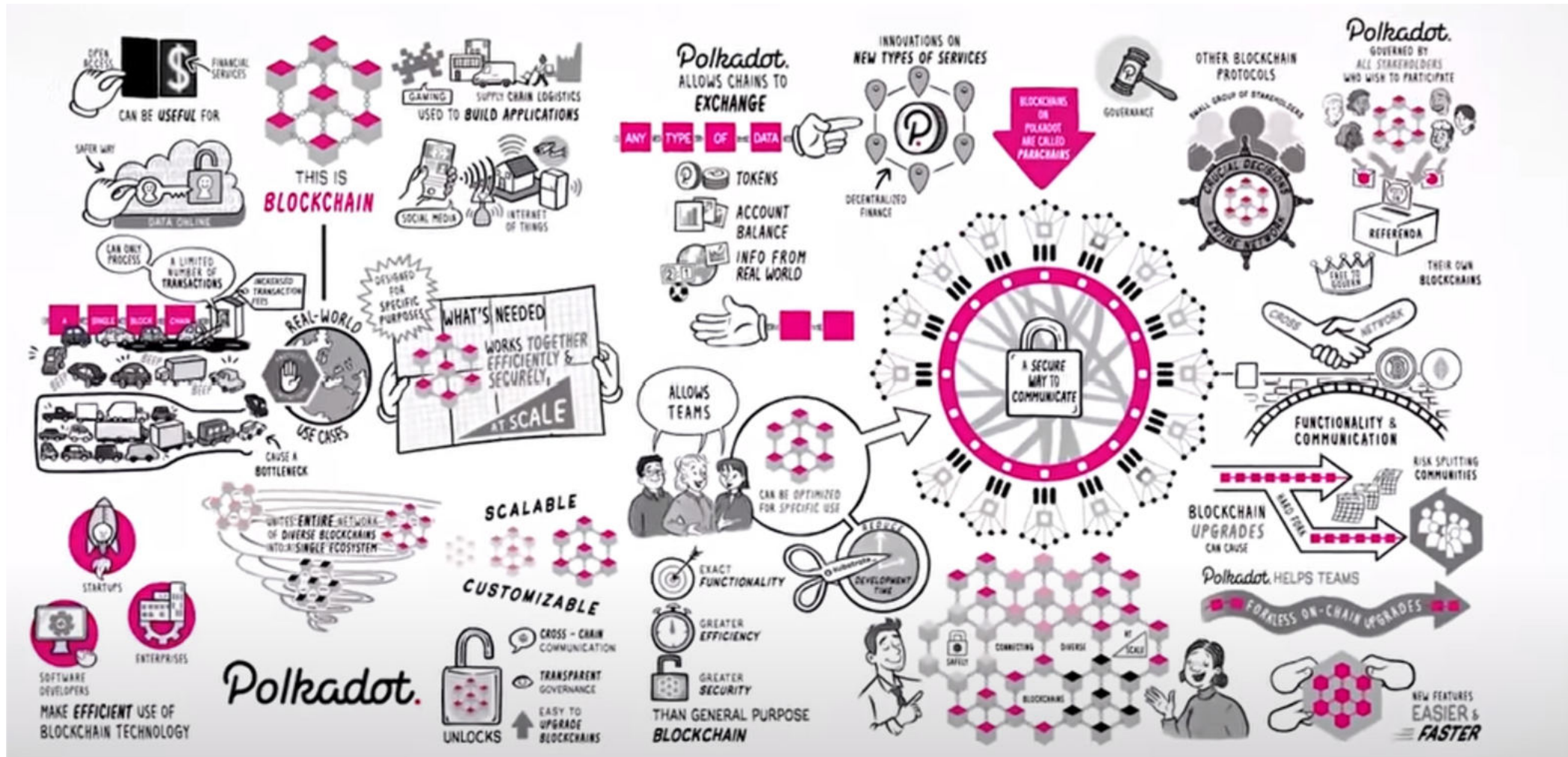
Web 3 stack

SubSocial(DC social network) Acala(Defi) Brave

Substrate Polkadot Consensus, XCM, Bridges

**Defining the technology, tools and
how they work**

PolkaDot Ecosystem



Polkadot

Polkadot as a protocol has 2 main goals

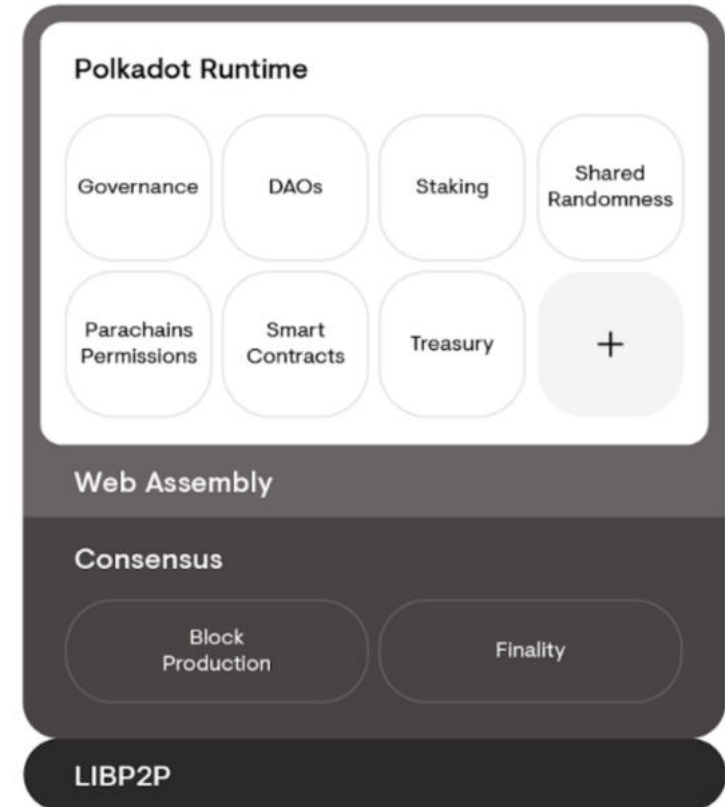
- Provide **shared security** among all connected parachains.
- Allow all connected chains to **interoperate** by using [XCM](#).
 - Cross-Consensus Message Format provides:
 1. **Asynchronous**: XCM messages in no way assume that the sender will be blocking on its completion.
 2. **Absolute**: XCM messages are guaranteed to be delivered and interpreted accurately, in order and in a timely fashion.
 3. **Asymmetric**: XCM messages out of the box do not have results that let the sender know that the message was received. Any results must be separately communicated to the sender with an additional message.
 4. **Agnostic**: XCM makes no assumptions about the nature of the Consensus Systems between which the messages are being passed.

Polkadot Runtime & Ecosystem Networks

- Mainnet: **Polkadot**
- Canary network: **Kusama**

Test Networks:

- **Westend:**
- **Canvas**
- **Rococo**



Kusama: Introduction

Kusama is a scalable network of specialized blockchains built using Substrate and nearly the same codebase as Polkadot. The network is an experimental development environment for teams who want to move fast and innovate on Kusama, or prepare for deployment on Polkadot.

Kusama was founded in 2019 by Gavin Wood, founder of Polkadot and co-founder and former CTO of Ethereum.

Polkadot vs. Kusama

Polkadot is a blockchain platform that aims to enable a decentralized and secure network of networks, where different blockchain networks can interoperate and exchange data and value. It is designed to support a wide range of use cases, including decentralized finance (DeFi), governance, and identity management.

Polkadot is and always will be the primary network for deploying enterprise-level applications and those that entail high-value transactions requiring bank-level security, stability, and robustness.

Polkadot vs. Kusama

Kusama is a blockchain network that is similar to Polkadot, but it is designed for more experimental and risk-tolerant use cases. It is intended to serve as a testbed for new technologies and ideas that may eventually be incorporated into Polkadot. It is important to note that Kusama is NOT a testnet for Polkadot, it is a standalone blockchain that currently operates as a much faster and cheaper canary network.

Projects that require high-throughput but don't necessarily require bank-like security, such as some gaming, social networking, and content distribution applications, are more inclined to use Kusama.

Rococo vs. Westend

- **Rococo** is a testnet for the Polkadot ecosystem that is designed for developers to test and refine their applications before deploying them to the main Polkadot network. It is essentially a development and testing environment for the Polkadot network.
- **Westend** is a testnet for the Polkadot ecosystem that is designed to simulate the main Polkadot network as closely as possible. It is intended to be used for testing and evaluating the performance and scalability of applications and services that are intended to run on the main Polkadot network.

Polkadot vs. Kusama vs. Rococo vs. Westend

Overall, each of these networks serves a different purpose within the Polkadot ecosystem.

Polkadot is the main network for decentralized applications and services.

Kusama is a testbed for experimental and high-risk use cases.

Rococo is a development and testing environment.

Westend is a simulation of the main Polkadot network for performance testing.

DOT & KSM

DOT is the native cryptocurrency of the Polkadot network. It is used to facilitate transactions and secure the network through a proof-of-stake (PoS) consensus mechanism.

KSM is the native cryptocurrency of the Kusama network. It is used to facilitate transactions and secure the network through a PoS consensus mechanism, similar to DOT.

Use Cases for DOT and KSM

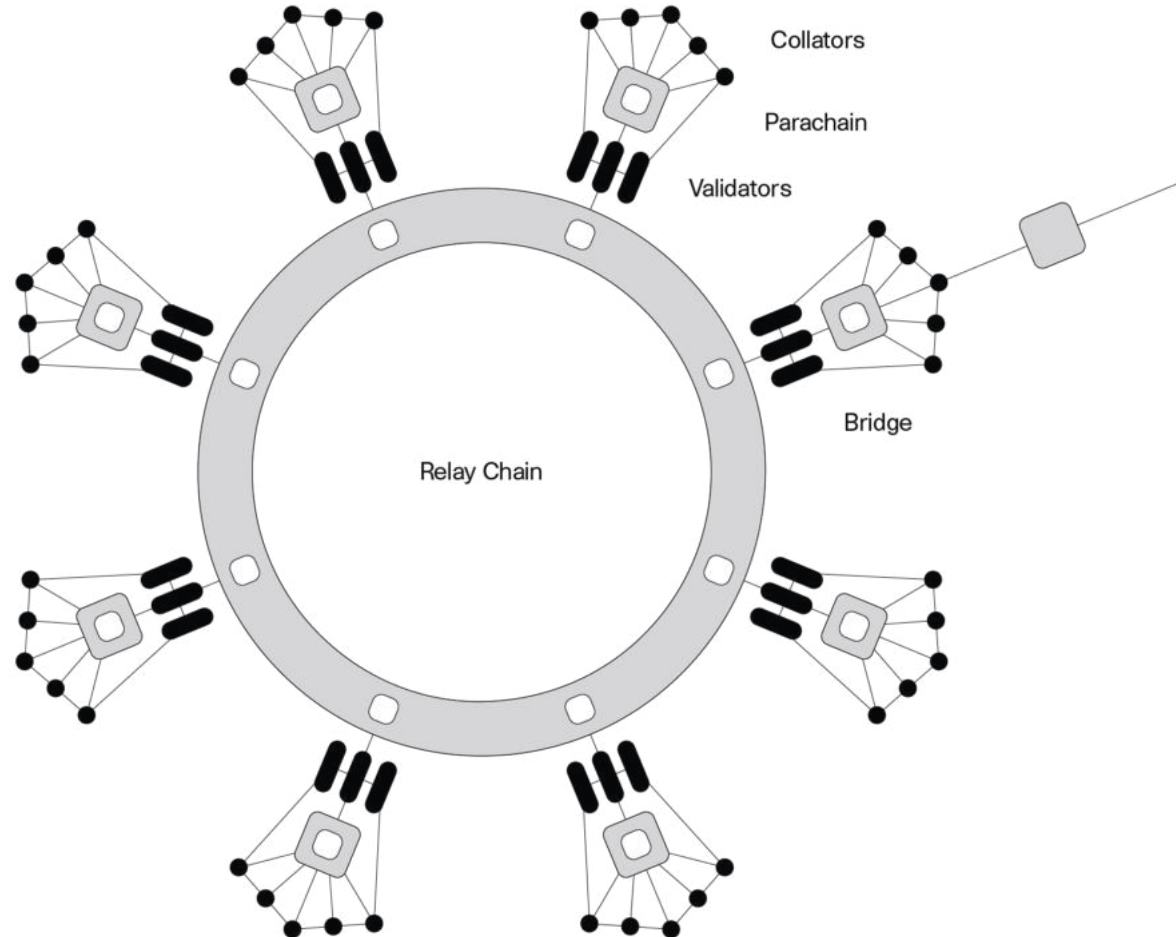
- Staking: Holders of DOT/KSM can earn rewards by participating in the network's PoS consensus mechanism. The staking of DOT/KSM acts as a disincentive for malicious participants who will be punished by the network by getting their DOT/KSM slashed.
- Governance: DOT/KSM holders can participate in decision-making processes on the network, such as determining the fees of the network, the addition or removal of parachains, and exceptional events such as upgrades and fixes to the Polkadot/Kusama platform.
- Payment: DOT/KSM can be used as a form of payment within the Polkadot/Kusama network.
- Parachain Slot Acquisition: DOT/KSM can be locked for a duration in order to secure a parachain slot in the network. The DOT/KSM will be reserved during the slot lease and will be released back to the account that reserved them after the duration of the lease has elapsed and the parachain is removed.

Key Similarities

- Nominated Proof of Stake (NPoS) system;
- Parachains connected to the Relay chain;
- Parachains talking to each other and to the Relay chain via XCM;
- Forkless upgrades that don't require validators to upgrade in advance;
- On-chain governance that gives every DOT or KSM owner a say in how the network will change.

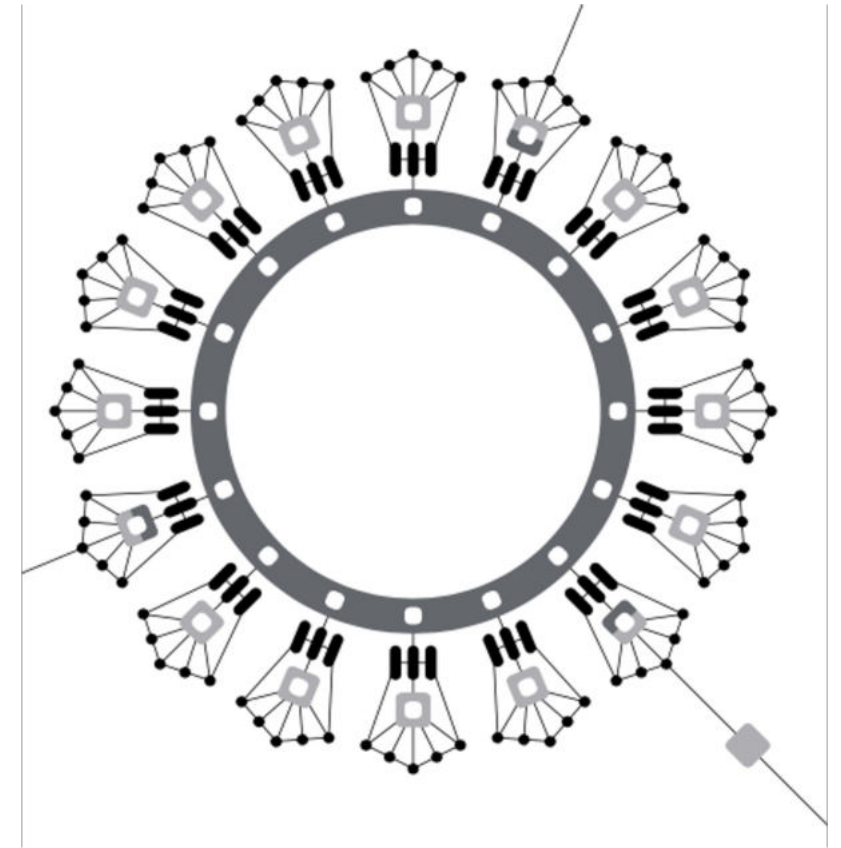
Polkadot	Kusama
High security	Low barriers to entry for parachain deployment
High stability	Low bond requirements for validators and parachains
More conservative governance and upgrade	Latest technology
High validator rewards	Low slashing penalties
	Fast iteration
Practical differences	
Existential deposit: 1 DOT	Existential deposit: 0.000333333 KSM
Minimum stake: 100 DOT	Minimum stake: 0.1 KSM
Unbonding and slash defer: 28 days	Unbonding and slash defer: 7 days
Minimum crowdloan contribution: 5 DOT	Minimum crowdloan contribution: 1 KSM
Parachain slot lease: up to 96 weeks	Parachain slot lease: up to 48 weeks
Era: 24 hours, Epoch: 4 hours	Era: 6 hours, Epoch: 1 hour
Democracy voting/launching/enactment: 28 days	Democracy voting/launching: 7 days, enactment: 8 days
Council: up to 13 members, up to 20 runners up	Council: up to 19 members, up to 19 runners up
Council term and voting: 7 days	Council term and voting: 1 day
Treasury: 24 days between spends	Treasury: 6 days between spends

The Polkadot Architecture



Relay Chain

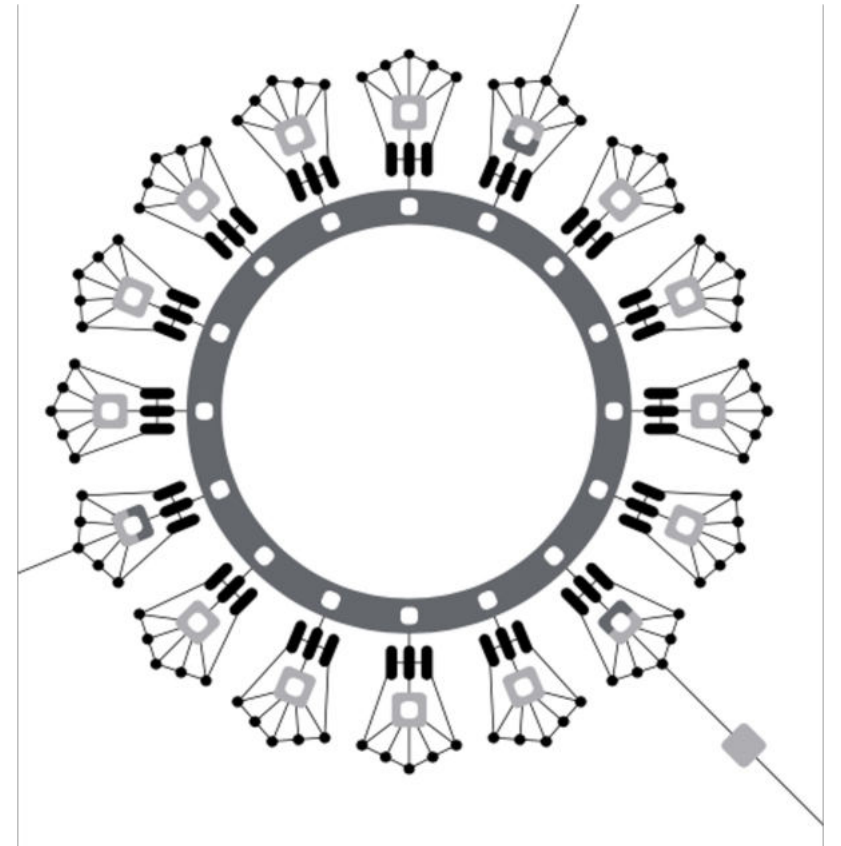
The core component of the Polkadot architecture, the relay chain is a proof-of-stake (PoS) blockchain that serves as the backbone of the network. It is responsible for coordinating and validating transactions, as well as facilitating interchain communication.



Parachains

The relay chain and parachains work together to facilitate interoperability by allowing for the exchange of data and information between different blockchain networks.

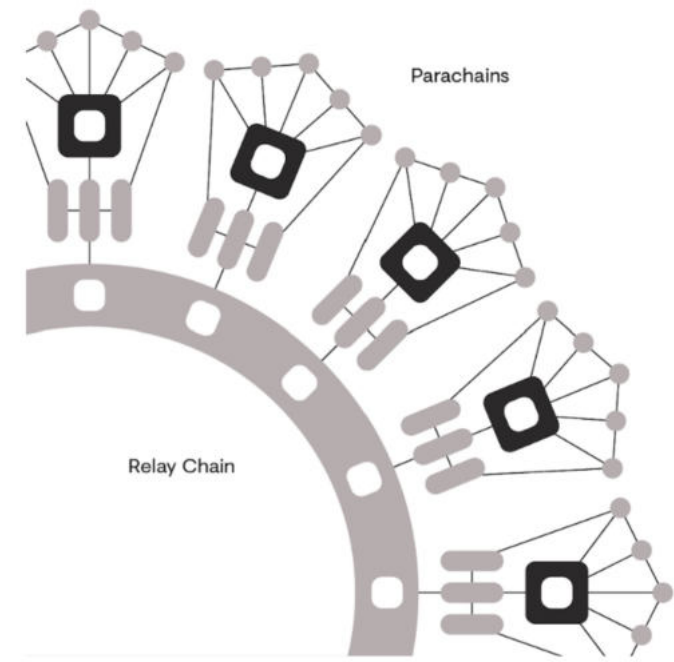
Parachains allow for the creation of specialized blockchain networks that can support specific types of transactions and applications.



Parachains, Parathreads

A **Parachain** is an application-specific data structure that is globally coherent and validatable by the validators of the Relay Chain. They are connected to and secured by the Relay Chain. The Polkadot network can host up to 100 Parachains.

Parathreads are on demand Parachains that temporarily participate (on a block by block basis) in Polkadot security without needing to lease a dedicated parachain slot. The Polkadot network can host up to 10,000 Parathreads



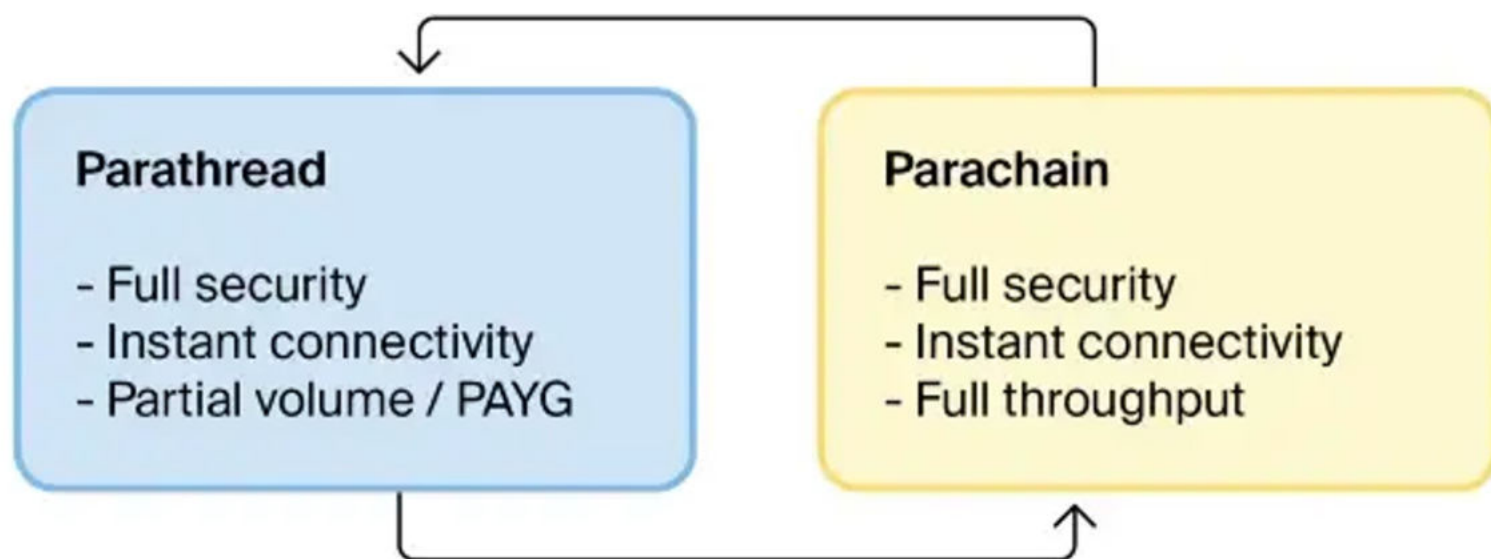
Benefits of Parachain

- **Scalability:** The sharded multichain network approach allows for what is essentially parallel computation (processing power) that can process several transactions in parallel.
- **Interoperability:** Any decentralised application or chain that wants to enable trustless messaging to other parachains already connected to Polkadot would want to become a parachain.
- **Isolatability:** Allows for the needs of many to be accounted for under the framework.
- **Dependability:** As a parachain, the blocks you submit are verified by validators with a Wasm runtime. Since the validators have the Wasm runtime for all the parachains, your parachain shares the security of the validator pool with everyone on the relay chain.
- **Governance:** Most governance systems in blockchains use an off-chain governance mechanism. Polkadot's on-chain governance encourages maximum participation of token holders and is frictionless and transparent.

Benefits of Parathreads

- **Parathreads** have a fixed fee for registration that would realistically be much lower than the cost of acquiring a parachain slot.
- It will be possible for parachain to swap their parachain slot with a **parathread** so that the parathread “upgrades” to a full parachain and the parachain becomes a parathread.
 - a. This allows for parachains to gracefully exit lease agreements and continue as parathreads that only produce blocks when necessary
- **Parathreads** help ease the sharp stop of the parachain slot term by allowing parachains that are still doing something useful to produce blocks, even if it is no longer economically viable to rent a parachain slot.
- **Parathreads** are great for applications that do not frequently update state and therefore do not need to continuously produce blocks.
- Like parachains, **parathreads** are secured under Polkadot’s shared security and can send and receive messages over XCMP.

Downgrade to parathread to have **deposit returned**.



Upgrade to parachain for **maximum throughput**.

Common Good Parachains

"Common Good" parachains are parachain slots reserved for functionality that benefits the ecosystem as a whole.

Instead of going through the parachain auction process, Common good parachains are allocated by Polkadot's on-chain governance system.

These types of parachain do not expire and can only be removed via governance.

There are two types of common good parachains: system level chains and public utility chains.

System Level Chains

System level chains move functionality from the Relay Chain into parachains, minimizing the administrative use of the Relay Chain.

These chains are generally uncontroversial because they merely move functionality that stakeholders already agreed was useful from the relay chain to the the parachain.

Moving logic from the relay chain to a parachain frees up processing power and leads to the entire network becoming more efficient.

Public Utility Chains

Public utility chains add functionality that doesn't exist yet, but that the stakeholders believe will add value to the entire network.

These chains must stay fully aligned with the Relay Chain, adopt it's native token (i.e DOT or KSM) as their native token and take messages from the Relay Chains and system level; parachains at face value.

Some examples of potential public utility chains are bridges, DOT/KSM-denominated smart contract platforms, and generic asset chains.

Examples of Common Good Parachains

Statemint and its Kusama equivalent **Statemine** are the first common good parachains. They both are public utility chains that add the ability to create and manage fungible and non-fungible assets. Due to low fees on parachains Statemint is well suited for handling DOT balances and transfers as well as managing on-chain assets.

Encointer is a blockchain platform for self-sovereign ID and a global universal basic income. It is registered as the second common good parachain on Kusama's network. The functionality of Encointer adds logic to the Relay Chain that aims to bring financial inclusivity to Web 3 and mitigate Sybil attacks with a novel Proof of Personhood (PoP) system for unique identity.

Parachain Slot Auctions

Parachain slot auctions are a mechanism for determining which blockchain projects get to run on the Polkadot network.

In order for a blockchain to operate as a parachain on the relay chain they must lease a slot.

Parachain slot auctions are an important part of the Polkadot ecosystem, enabling the network to efficiently and fairly allocate its resources to the most valuable projects.

How do they work?

Parachain slot auctions are a form of candle auction, where participants submit a bid for a parachain slot and the winner is selected at a random time during the ending period of an auction.

The bid represents the amount of DOT tokens that the participant is willing to pay for the slot.

The highest bidder of the randomly selected block wins the auction and gets to run their blockchain on the Polkadot network.

Benefits of Parachain Slot Auctions

- Parachain slot auctions provide a fair and transparent mechanism for allocating parachain slots on the Polkadot network.
- They incentivize participants to bid competitively, ensuring that the slots are allocated to the projects that value them the most.
- They also provide a source of revenue for the Polkadot network, as the DOT tokens from winning bids are used to fund its operations.

Parachain Crowdlans

- Parachain crowdlans are a way for projects to raise funds to launch and maintain their parachains on the Polkadot network.
- These loans are typically funded by the community, with investors receiving a share of the parachain's transaction fees as repayment.
- Crowdlans are a unique fundraising mechanism that allows projects to tap into the Polkadot community and secure the funding needed to launch and operate their parachains.
- Once the crowdlan is live, the parachain configuration will be locked and will be deployed as the parachain's runtime.

Benefits of Parachain Crowdlans

- One of the key benefits of parachain crowdlans is that they provide a way for projects to raise funds without having to go through traditional venture capital channels.
- This allows for a more decentralized and community-driven approach to funding, which aligns with the values of the broader blockchain ecosystem.
- Additionally, investors in parachain crowdlans can earn a return on their investment through the share of transaction fees they receive, providing a potential additional incentive for participation.

How to Participate in Parachain Crowdlans

- Interested investors can participate in parachain crowdlans through the Polkadot network(PolkadotJS).
- This typically involves purchasing DOT tokens, which are the native currency of the Polkadot network, and using them to fund the crowdlan of a specific parachain project.
- It is important to carefully research and evaluate the different parachain projects before participating in a crowdlan, as with any investment.

Crowdloan Participation (Cont.)

If the campaign was successful, then the parachain will enter a retirement phase at the end of its lease. During this phase, participants can withdraw the tokens with which they participated.

If the campaign was unsuccessful, then this retirement phase will begin at the campaign's configured end, and participants can likewise withdraw their tokens.

Smart Contracts

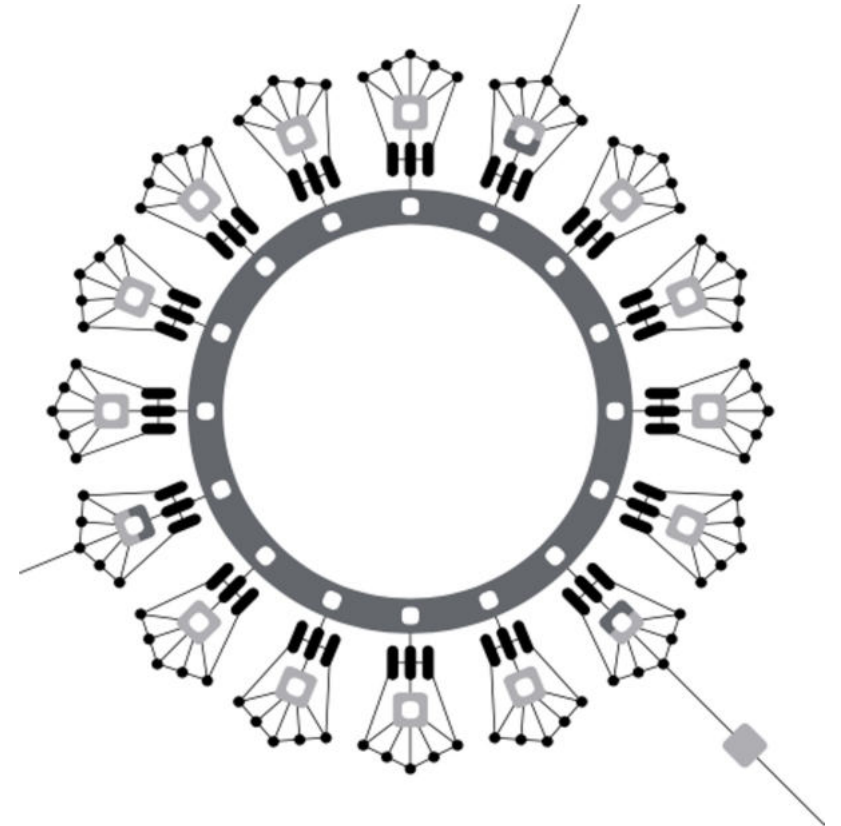
- Defined: Smart contracts are executable programs that exist on only a single chain and are limited in complexity. It's the code that exists at an address on a chain and is callable by external actors.
- Deployment: Parachain Specific by sending a special transaction that will create the smart contract on the parachain ledger.
- Paying for the smart contract:
 - Transaction fee, subscription fee, access token, trial or promotion or gas
- Consider storage and complexity

Contrasting Parachains & Smart Contracts

	Parachain	Smart Contract
Speed of Development	-	+
Ease of Deployment	-	+
Complexity of Logic	+	-
Maintenance Overhead	-	+
Level of customization	+	-
Strict resource control	-	+
Native chain features	+	-
Scalability	+	-

Bridges

Bridges are specialized parachains that are used to connect the Polkadot network to other blockchain networks. They enable interoperability between different blockchain networks.



Bridging Methods

- Bridge pallets
- Smart contracts
- Higher-order protocols

via Bridge Pallets

Receiving messages on Polkadot from an external, non-parachain blockchain can be possible through a Substrate pallet.

The Substrate instance can then be deployed to Polkadot either as a system-level parachain (native extension to the core Polkadot software) or as a community-operated parachain.

An example of this is the Kusama to Polkadot bridge.

via Smart Contracts

Given the generality of blockchain platforms with Turing-complete smart contract languages, it is possible to bridge Polkadot and any other smart contract capable blockchain.

Ethereum's Parity Bridge is an example of this. It consists of two smart contracts, one on each chain and allows for cross-chain transfers of value.

via Higher-Order Protocols

Higher-order protocols (like XCLAIM) can be used to bridge but should only be used when other options are not available. XCLAIM, in particular, requires any swappable asset to be backed by a collateral of higher value than the swappable assets, which adds additional overhead.

An example of a network that would be well-suited for higher-order protocols would be Bitcoin, since it does not support smart-contracts and it's not based on Substrate.

Nominators

In the Polkadot ecosystem, nominators are individuals or entities who participate in the network by staking their tokens on a validator.

The purpose of nominators in the Polkadot ecosystem is to support the validators and contribute to the security and stability of the network, while also participating in its governance and decision-making processes.

Nominating	Joining a Pool
Minimum 100 DOT to nominate.	Minimum 1 DOT to be a member.
Rewards can be compounded automatically or sent to any account.	Rewards can be manually claimed to the pool member's account and be bonded in the pool again to compound them.
If the active validator gets slashed, all active nominators are subjected to slashing, also those that do not receive rewards due to the oversubscription issue.	If the active validator gets slashed, all pool members are subjected to slashing.
Can bond and stake DOT indefinitely.	Can bond and stake DOT until the pool exists.
Unbonding period of 28 days. Can switch validators without unbonding.	Unbonding period of 28 days. Need to unbond before switching to a different pool.
Maximum uncapped.	Maximum uncapped.
Should bond more than the minimum active nomination in an era to be eligible to earn staking rewards, although it can depend on multiple other factors outlined in the linked document.	A nomination pool earns rewards in an era if it satisfies all the conditions mentioned for the nominator (as the nomination pool is just a nominator from the NPoS system perspective).
Staked tokens can be used for participation in Governance.	Staked tokens cannot be used for participation in Governance.
Rewards payout can be triggered permissionlessly by anyone (typically done by the validator).	Rewards must be claimed by the pool member.

Nominated proof of stake

Polkadot implements Nominated Proof-of-Stake (NPoS), a relatively novel and sophisticated mechanism to select the validators who are allowed to participate in its consensus protocol.

NPoS encourages DOT holders to participate as nominators.

Nominating Validators

Nominating on Polkadot requires 2 actions:

- Locking a minimum of 100 DOT tokens on-chain.
- Selecting a set of validators, to whom these locked tokens will automatically be allocated to.

The action of locking tokens is also known as bonding

Being a Nominator

Nominators have far fewer responsibilities than validators. These include:

- Selecting validators and monitoring their performance,
- Keeping an eye on changing commission rates
- General health monitoring of their validators' accounts.

Staking Rewards

Validators who produce a block are rewarded with tokens, and they can share rewards with their nominators.

Both validators and nominators can stake their tokens on chain and receive staking rewards at the end of each era.

The staking system pays out rewards equally to all validators regardless of stake. Thus, having more stake in a validator does not influence the amount of block rewards it receives.

Slashing

Slashing will happen if a validator misbehaves (e.g. goes offline, attacks the network, or runs modified software) in the network.

In this scenario, both validators and nominators lose a percentage of their staked DOT.

Levels of Slashing

- Level 1: isolated unresponsiveness
- Level 2: concurrent unresponsiveness or isolated equivocation
- Level 3: misconducts unlikely to be accidental, but which do not harm the network's security to any large extent
- Level 4: misconduct that poses serious security or monetary risk to the system, or mass collusion

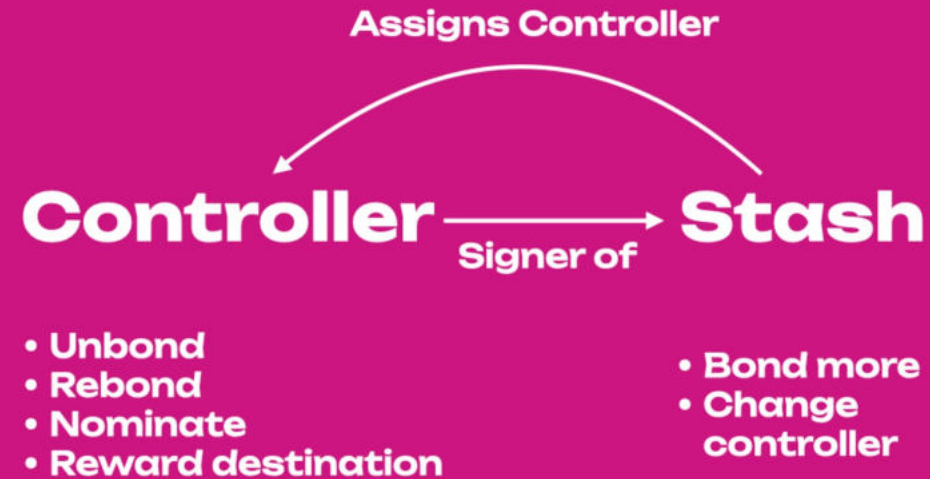
Pros of Staking

- Earn rewards for contributing to the network's security through staking.
- Low barrier of entry through Nomination Pools.
- Can choose up-to 16 validators which can help to decentralize the network through the sophisticated NPoS system
- 10% inflation/year of the tokens is primarily intended for staking rewards.

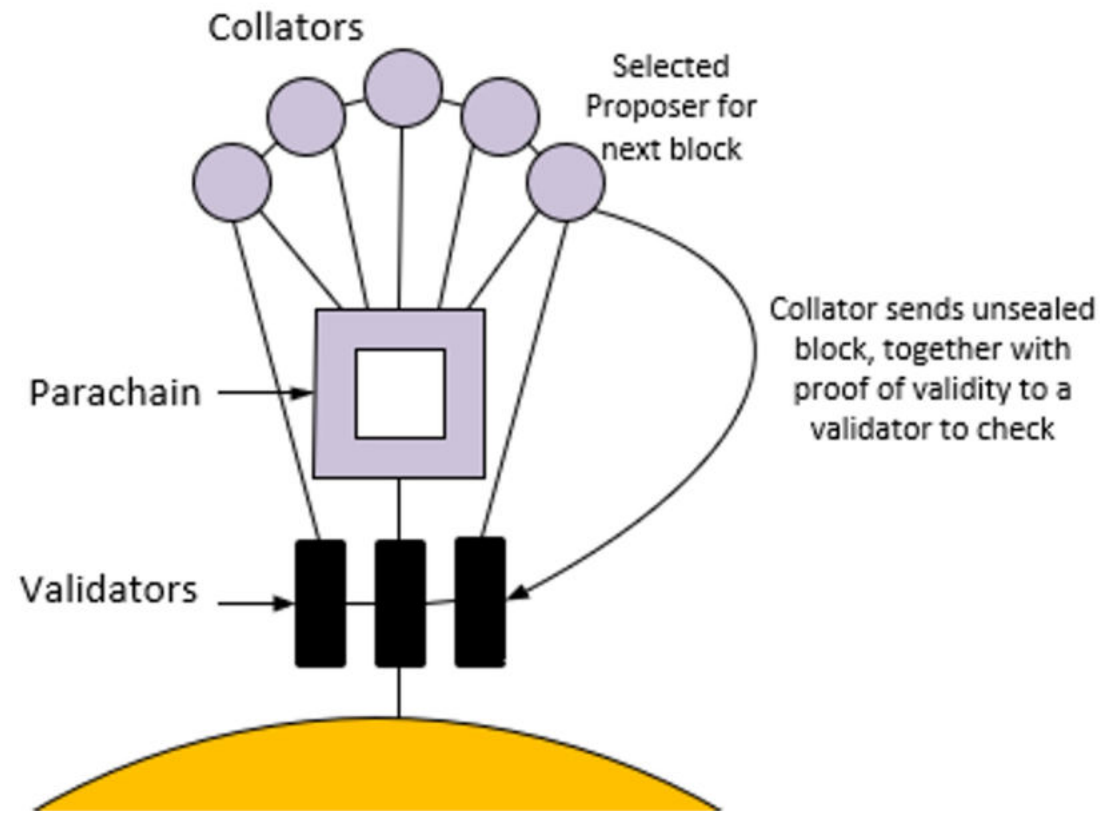
Cons of Staking

- Tokens will be locked for about 28 days on Polkadot. No rewards will be earned during the unbonding period.
- Possible punishment in case of the active validator found to be misbehaving.
- Lack of liquidity i.e. You would not be able to use the tokens for participating in crowdloans or transfer them to different account etc.

Stash and Controller Accounts for Staking



Validators and Collators



Polkadot Host

A Polkadot host is a server or computer that is running a full node of the Polkadot blockchain.

A full node is a complete copy of the blockchain ledger and is responsible for validating and relaying transactions and blocks on the network.

Components of the Polkadot Host

- Networking components such as Libp2p that facilitates network interactions.
- State storage and the storage trie along with the database layer.
- Consensus engine for GRANDPA and BABE.
- Wasm interpreter and virtual machine.
- Low level primitives for a blockchain, such as cryptographic primitives like hash functions.

Functions of a Host node

- Must populate the state storage with the official genesis state.
- Should maintain a set of around 50 active peers at any time. New peers can be found using the discovery protocols.
- Should open and maintain the various required streams with each of its active peers.
- Should send block requests to these peers to receive all blocks in the chain and execute each of them.
- Should exchange neighbor packets.

Polkadot Runtime



Web Assembly

Consensus



LIBP2P

On-chain Governance

On-chain governance refers to the use of smart contracts and decentralized autonomous organizations (DAOs) to facilitate decision-making and voting processes on a blockchain

In Polkadot, on-chain governance is facilitated through a combination of validators, token holders, and the Council.

On-chain Governance in Polkadot

Validators help secure the network and participate in decision-making processes, while token holders can participate in voting processes and have a say in the governance of the network.

The Council is a group of elected representatives that serve as the decision-making body for the network.

Referenda

Referenda are simple, inclusive, stake-based voting schemes.

They can be started in one of several ways:

- Publicly submitted proposals
- Proposals submitted by the council, either through a majority or unanimously
- Proposals submitted as part of the enactment of a prior referendum
- Emergency proposals submitted by the Technical Committee and approved by the Council

Voting Timetable

Every 28 days, a new referendum will come up for a vote.

The top proposal from the queue of council-approved or publicly submitted proposals is chosen to be the referendum for that month.

Multiple referenda cannot be voted upon in the same period, excluding emergency referenda.

Voting on a referendum

To vote, a voter generally must lock their tokens up for at least the enactment delay period beyond the end of the referendum.

It is possible to vote without locking at all, but your vote is worth a small fraction of a normal vote, given your stake.

Voluntary locking provides a way for voters with small amount of tokens to influence the referendum result.

Tallying

Depending on which entity proposed the proposal and whether all council members voted yes, there are three different scenarios. We can use the following table for reference.

Entity	Metric
Public	Positive Turnout Bias (Super-Majority Approve)
Council (Complete agreement)	Negative Turnout Bias (Super-Majority Against)
Council (Majority agreement)	Simple Majority

Voluntary Locking

*votes = tokens * conviction_multiplier*

Peter: Votes **No** with 10 DOT for a 128 week lock period => $10 \times 6 = 60$ Votes

Logan: Votes **Yes** with 20 DOT for a 4 week lock period => $20 \times 1 = 20$ Votes

Kevin: Votes **Yes** with 15 DOT for a 8 week lock period => $15 \times 2 = 30$ Votes

Even though combined both Logan and Kevin vote with more DOT than Peter, the lock period for both of them is less than Peter, leading to their voting power counting as less.

Council

An on-chain entity comprising several actors, each represented as an on-chain account.

On Polkadot, the council currently consists of 13 members.

The council is called upon primarily for three tasks of governance:

1. Proposing sensible referenda
2. Cancelling uncontroversially dangerous or malicious referenda
3. Electing the technical committee

Benefits of On-Chain Governance

- Transparency
- Decentralization
- Flexibility

By enabling transparent and decentralized decision-making and voting processes, on-chain governance helps ensure the long-term success and stability of the network.

On-Chain Treasury in Polkadot

The on-chain treasury is managed by the Council and funded through a portion of block production rewards, transaction fees, slashing, staking inefficiencies, etc..

Key features:

- Decentralized control
- Transparency
- Funding for grants and projects

Decision-Making and Funding

Decisions regarding the use of the on-chain treasury are made through a combination of on-chain voting and offline deliberation by the Council.

- The Council is responsible for considering proposals and making decisions on behalf of the network
- Approved spending proposals enter a waiting period before distribution

Types of Spending Proposals

Proposals may consist of (but are not limited to):

- Infrastructure deployment and continued operation
- Network security operations
- Ecosystem provisions
- Marketing activities
- Community events and outreach
- Software development

Funding the Treasury

The Treasury is funded from different sources:

- Slashing
- Transaction fees
- Staking inefficiency
- Parathreads

Benefits of On-Chain Treasury

- Decentralized control
- Transparency
- Funding for grants and projects
- Long-term sustainability
- Ecosystem development

Transaction fees

Transaction fees are small amounts of cryptocurrency that users pay to have their transactions included in a block and added to the blockchain.

Polkadot uses a **weight-based fee model** as opposed to a **gas-metering model**. As such, fees are charged prior to transaction execution; once the fee is paid, nodes will execute the transaction.

Fee Calculation

Fees on the Polkadot Relay Chain are calculated based on three parameters:

- A Weight fee
 - Base weight
 - Call(s) weight
- A Length fee
- A Tip (optional)

Parameters: weight fee, length fee, tip, targeted fee adjustment

Inclusion fee: sum of base fee, length fee, and adjusted weight fee

Allocation of fees

- Inclusion fee deducted from sender's account
- Portion to block author, remainder to Treasury
- Tips go directly to block author

Refunds

- Final weights based on worst case scenario of functions
- Refunds for overestimated weights
- Amount determined by final weight

Best practices for optimizing fees

- Reduce transaction size
- Add a tip
- Monitor network congestion

Forkless upgrades

Runtime upgrades allow Polkadot to change the logic of the chain, without the need for a hard fork.

They allow for the seamless integration of new features and improvements to the network.

How do forkless upgrades work?

By using Wasm in Substrate, Polkadot is able to upgrade it's runtime logic without hard forking.

This is generally a two step process:

- The existing runtime logic is followed to update the Wasm runtime stored on the blockchain to a new version.
- The upgrade is then included in the blockchain itself, meaning that all the nodes on the network execute it

Benefits of forkless upgrades on Polkadot

- Forkless upgrades on Polkadot allow for seamless integration of new features and improvements to the network.
- They improve the stability and security of the network by reducing the risk of network disruption.
- They also enable a more gradual and controlled rollout of updates, making it easier to identify and fix any issues that may arise.

Shared security

Parachains connected to the Polkadot Relay Chain all share in the security of the Relay Chain.

Polkadot has a shared state between the Relay Chain and all of the connected parachains.

The shared state ensures that the trust assumptions when using Polkadot parachains are only those of the Relay Chain validator set and no other.

Cross-Consensus Messaging(XCM)

Cross-consensus communication

Cross-consensus communication relies on a message format- XCM that is designed to provide a generalized and extensible set of instructions for completing transactions across boundaries created by different systems, transaction formats and transport protocols.

Interoperability and parachains

Parachains offer a promising solution for achieving interoperability between different blockchain networks

By enabling communication and exchange between different blockchain networks, parachains can help create a more cohesive and connected ecosystem for decentralized applications and decentralized finance (DeFi).

Interoperability and parachains

Parachains can achieve interoperability by connecting to the relay chain, through a process called "parachain bonding".

This allows the parachain to interact with the main chain and access its security and network resources, while still maintaining its own independent functionality.

To achieve interoperability parachains utilize cross-channel messaging to seamlessly communicate with each other while leveraging the security of the relay chain.

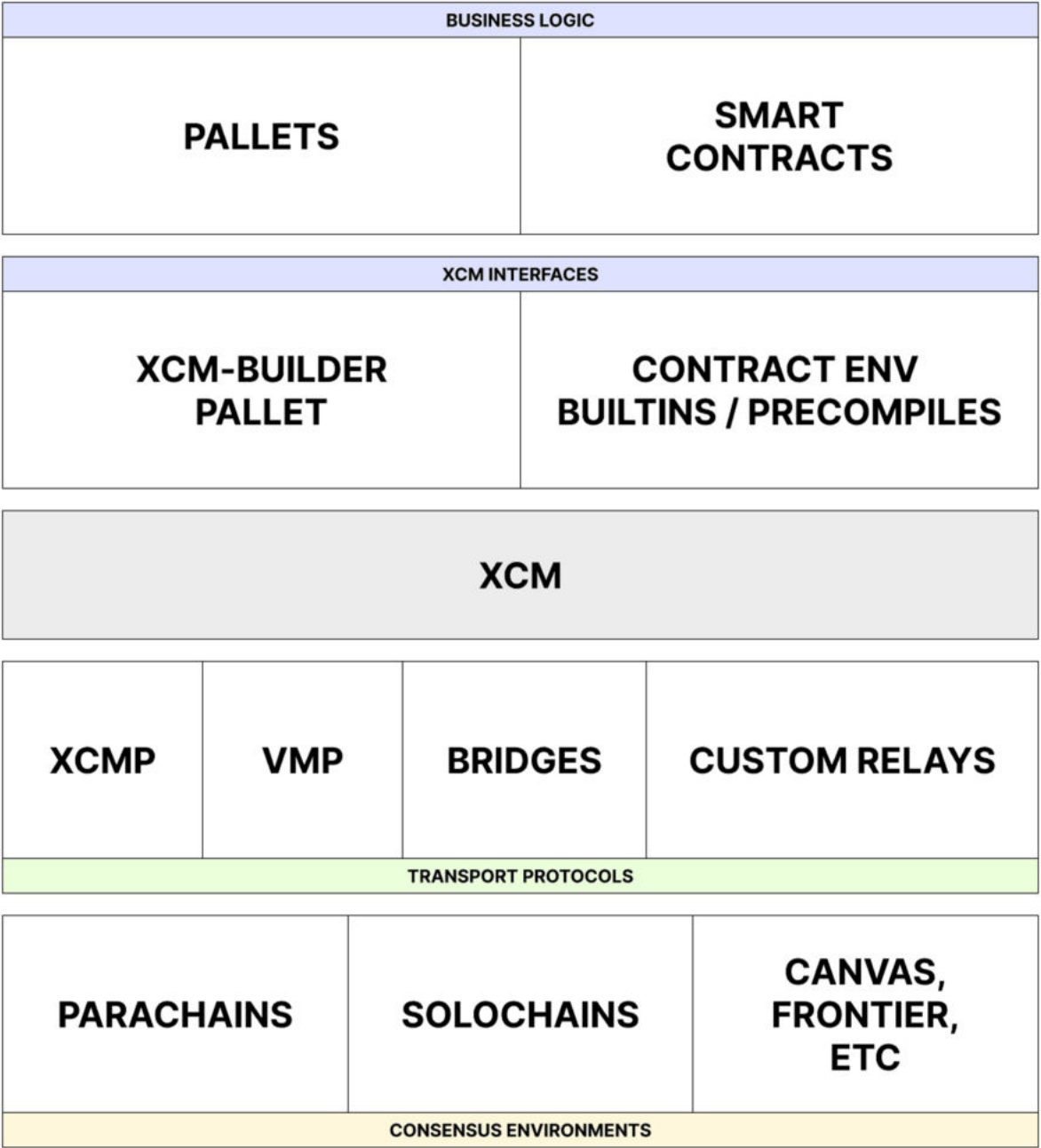
Benefits of Interoperability

- Increased scalability
- Increased flexibility
- Increased security

Message Protocols

In the Polkadot ecosystem, there are three main communication channels used to transfer messages between chains:

- Upward message passing (UMP)
- Downward message passing (DMP)
- Cross-consensus message passing (XCMP)



Messages in the XCM format

Four important principles about messages that use the XCM format:

- Messages are asynchronous
- Messages are absolute
- Messages are asymmetric
- Messages are agnostic

Locations

XCM is a language for communication between different consensus systems, it must have an abstract way to express locations in a general and flexible way:

- Where an instruction should be executed
- Where an asset should be withdrawn from
- Where an account to receive assets can be found

XCM format

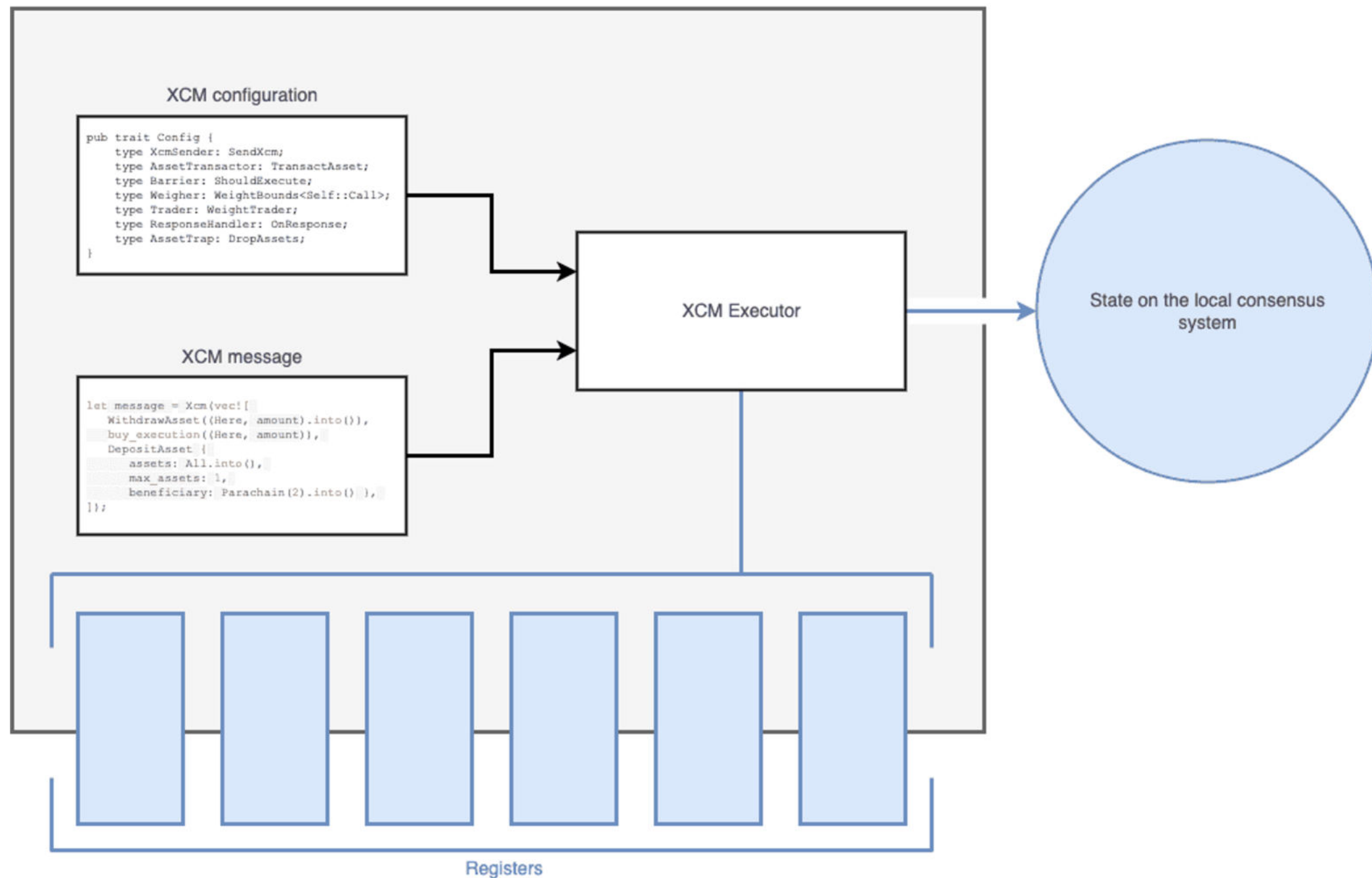
In Rust, message is defined like:

```
pub struct Xcm<Call>(pub Vec<Instruction<Call>>);
```

Execution in a virtual machine

- The cross-consensus virtual machine(XCVM) is a high level virtual machine with an XCM executor program that executes the XCM instructions it receives
- The program executes the instructions in order until it runs to the end or encounters an error and stops execution

Cross-consensus virtual machine



Address formats

- Overview of SS58 Format
 - Address Format
- Derivation Paths
- Address Generation Tools
- Address Validation in Polkadot
- Conclusion

SS58 Format

- Addresses are **unique identifiers** for accounts Polkadot
- Substrate-based chains use the **SS58 address format**
- SS58 includes an **address type prefix** that identifies which network an address belongs to
- Derivation paths allow for the creation and management of **multiple accounts** using the same seed
- Every networks format represents **the same public key** in a private-public key pair
- Substrate-based chains are compatible as long as the format is converted correctly

Ethereum Address

- `0xfd1472d6f10826df5fe5769990c6080523a30a35`

Polkadot Address

- `12uxb9baJaiHhCvMzijncybkiXpGQ24jhj4AmhNvrMEzWuoV`
- `Evh78gP5ATklKjHonVpxM8c1W6rWPKn5cAS14fXn4Ry5NxK`
- `2q5qF1LqDpINWGC1JJaczmPQMGPZPKQ76f2XqzxMBjwmadxW`
- `5DyfSpLWSoSpFfur35gn4PmbrupchiWbdEKgcQPaJGDULHGd`
- `jRaQ6PPzCqNnckcLStwqrTjEvpKnJUP2Jw65Ut36LQQUycd`
- `4dFhts6694CTKKV4btQdnzB3yzxrNcjUVaztvJXmX8eYeXox`

Derivation Paths

- Derivation paths allow for the creation and management of multiple accounts using the same seed
- Soft vs. Hard Derivation
 - Hard derivation uses "//" after the mnemonic phrase
 - e.g. "caution juice atom organ advance problem want pledge someone senior holiday very//0"
 - Soft derivation uses "/" after the mnemonic phrase
 - e.g. "caution juice atom organ advance problem want pledge someone senior holiday very/0"
- Password derivation uses "///" after the mnemonic phrase and requires a password in addition to the mnemonic phrase and derivation path to access the account
- Things to remember
 - Use well-defined sequences or write down derivation paths for easy recreation
 - Use password derivation for added security
 - Unless there is a specific need for a soft derivation, it is recommended to use hard derivation for added security

Address Generation Tools in Polkadot

- Toolkit
 - Polkadot JS Apps Suite
 - Polkadot CLI
- Create private-public key pairs to generate unique addresses
- Note: it's good to use

BUILDING

Substrate

- Substrate is the primary blockchain SDK used by developers to create the parachains that make up the Polkadot network. It allows to create specialized blockchains for any use case.
- It was used by Parity Technologies to create Polkadot itself, which attest to its high level performance, flexibility and robustness.

Substrate & it's vision

- True Interoperability
 - Substrate chains interoperate without bridge requirements
- Shared Security
 - Security is shared without compromise to all chains connected to it
- Sovereign Governance
 - Each chain also maintains independent governance for maximum freedom

Substrate Advantages

Unlike other distributed ledger platforms, Substrate is:

- **Flexible**
- **Open**
- **Interoperable**

Substrate Future-proof:

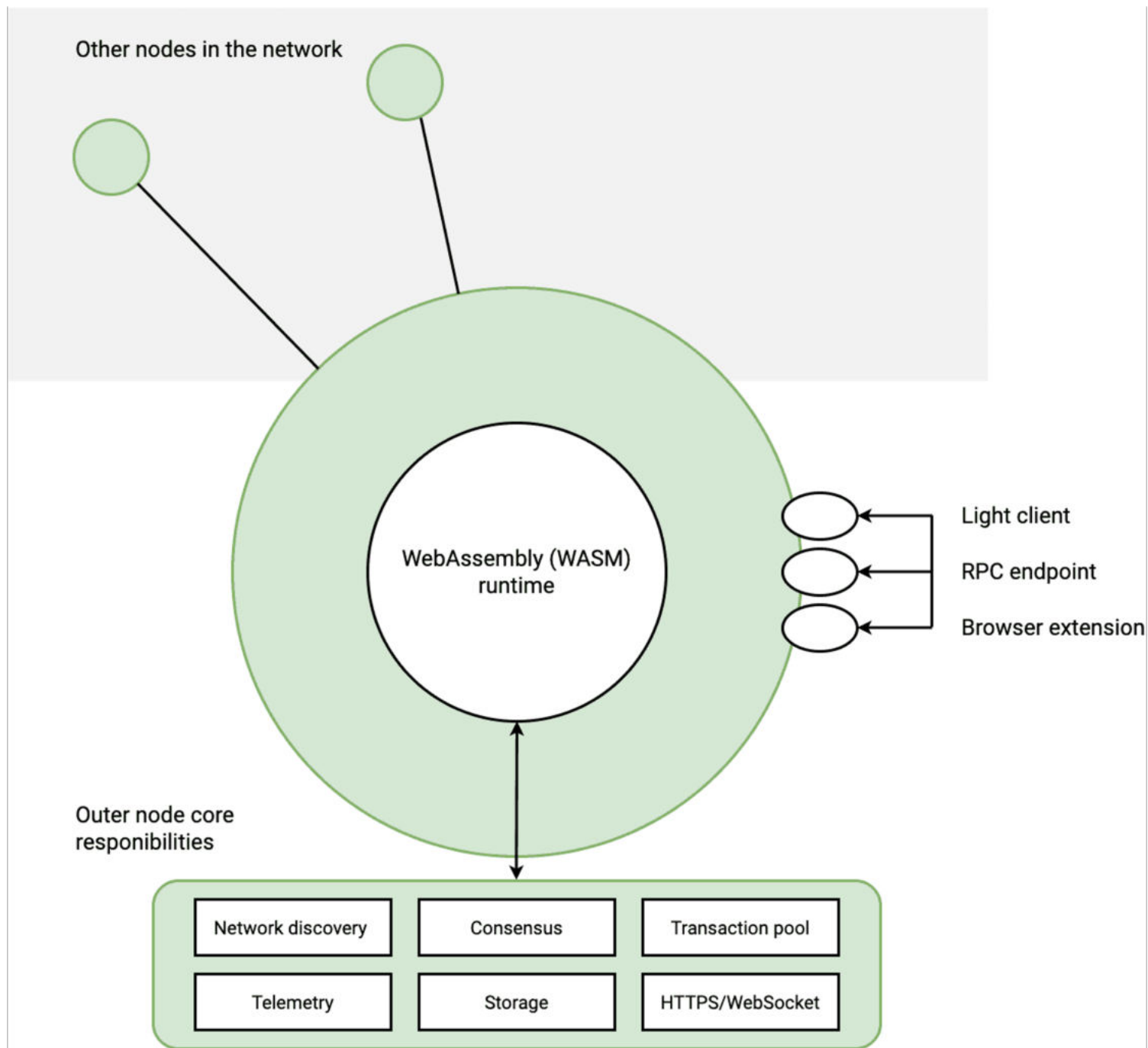
Substrate is built to be upgradeable, composable and adaptable.

The substrate runtime is a self-contained WebAssembly object. The nodes in the blockchain can be given the ability to completely change the runtime itself under specific conditions, inducing a runtime upgrade network wide.

Thus “forkless” upgrade are possible, as no action is required in most cases for the nodes to operate with this new runtime.

Substrate Architecture

- The substrate architecture divides operational responsibilities just like any other decentralized networks where all nodes act as both clients that request data and as servers that reasoning to requests for data.
- A substrate node provides a layered environment with two main elements:
 - An **outer node** that handles network activity such as peer discovery, managing transaction requests, reaching consensus with peers and responding to RPC calls
 - A **runtime** that contained all of the business logic for executing the state transition functions of the blockchain



Outer Node

- Outer node is responsible for activity that takes place outside of the runtime like handling peer discovery, managing the transaction pool, communicating with other nodes to reach consensus and answering RPC calls
- Some of the most important activities that are handled by outer node are:
 - Storage
 - Peer-to-peer networking
 - Consensus
 - Remote procedure calls
- Performing these task often requires the outer node to query the runtime for information or to provide information to the runtime which is handled by calling specialized runtime APIs

Substrate Runtime

- Runtime is responsible for handling the changes to the blockchain's state transition function and everything that happens on-chain. It determines whether a transaction is valid or invalid
- Runtime controls how transactions are included in blocks and how blocks are returned to the outer node for communicating with other nodes
- It is the core component of the node for building Substrate blockchains

Runtime

The substrate runtime is designed to compile to WebAssembly (wasm) byte code which enables:

- Support for forkless upgrades
- Multi-platform compatibility
- Runtime validity checking
- Validation proofs for relay chain consensus mechanisms

Networks and Blockchains

- Substrate enables the development of any type of blockchain and define its boundaries bases on the application specific requirements
- Substrate-bases blockchains can be used in different types of network architectures. For example:
 - Private networks
 - Solo chains
 - Relay chains
 - Parachains

Network types

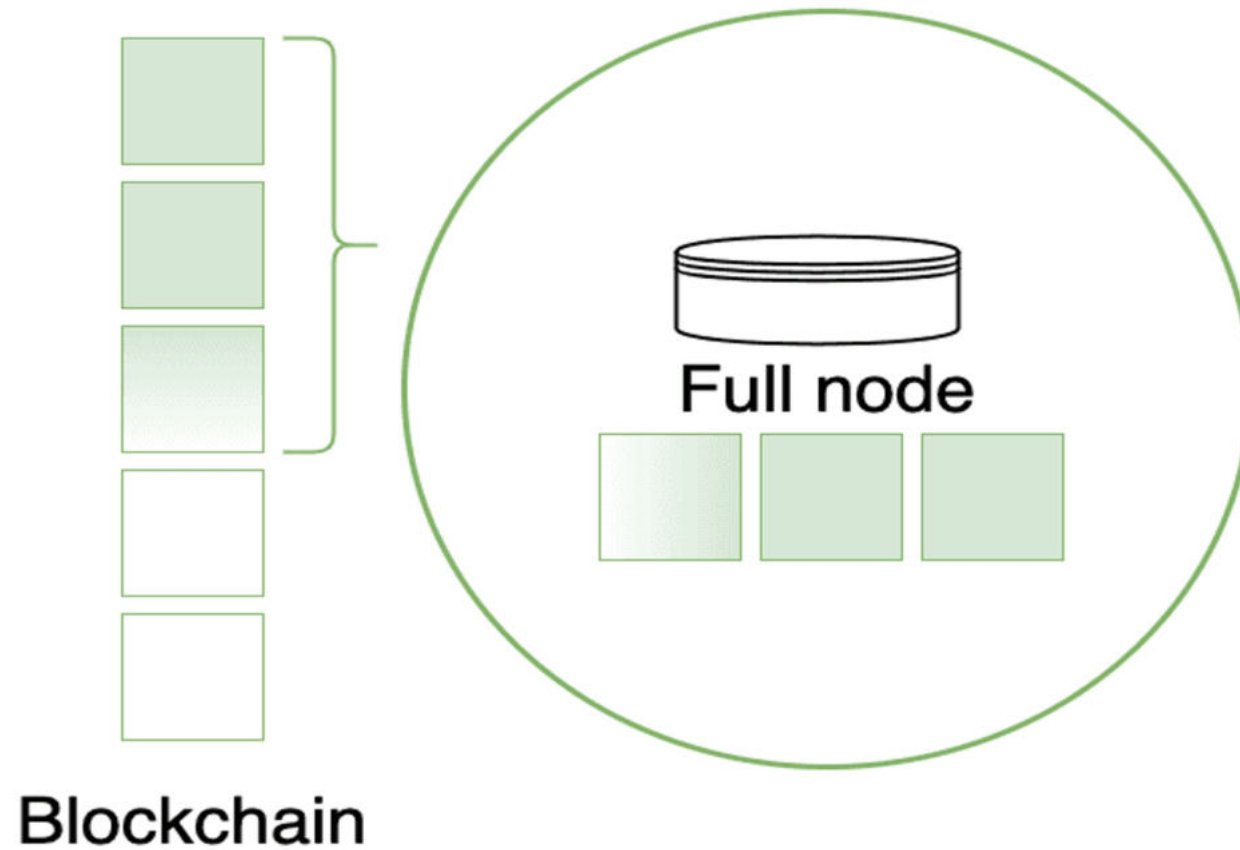
- **Private networks:** They limit access to a restricted set of nodes
- **Solo chains:** They implement their own security protocol and don't connect or communicate with any other chains
- **Relay chain:** Provide decentralized security and communication for other chains that connect to them
- **Parachain:** They are built to connect to a relay chain and have the ability to communicate with other chains that use the same relay chain

Node types

- Blockchain network nodes are synchronised to provide up-to-date view of the blockchain state. Each synchronized node stores a copy of the blockchain and keep track of the incoming transactions which requires a lot of storage and computing resource
- To make it easier to maintain the security and integrity, there are different types of nodes that can interact with the chain:
 - Full nodes
 - Archive nodes
 - Light client nodes

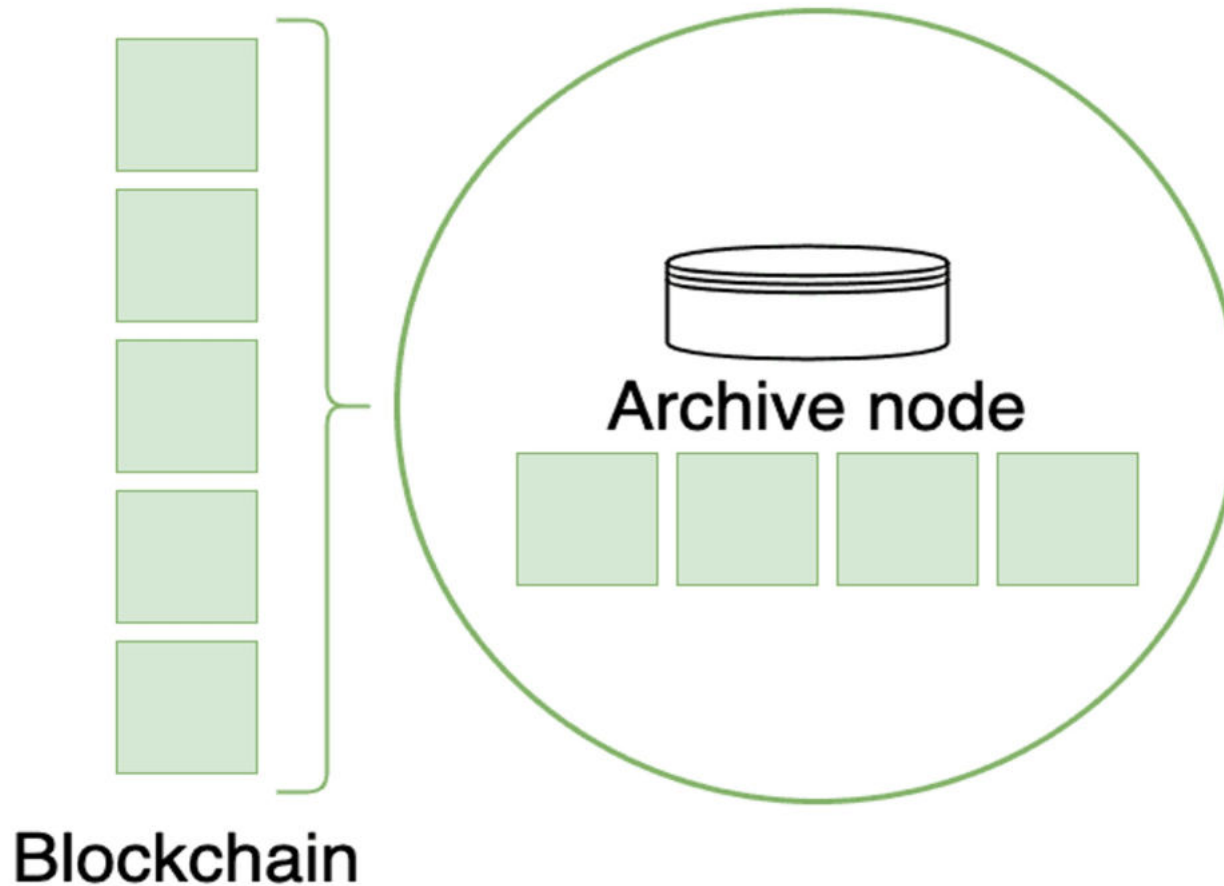
Full Nodes

Synchronize recent blocks



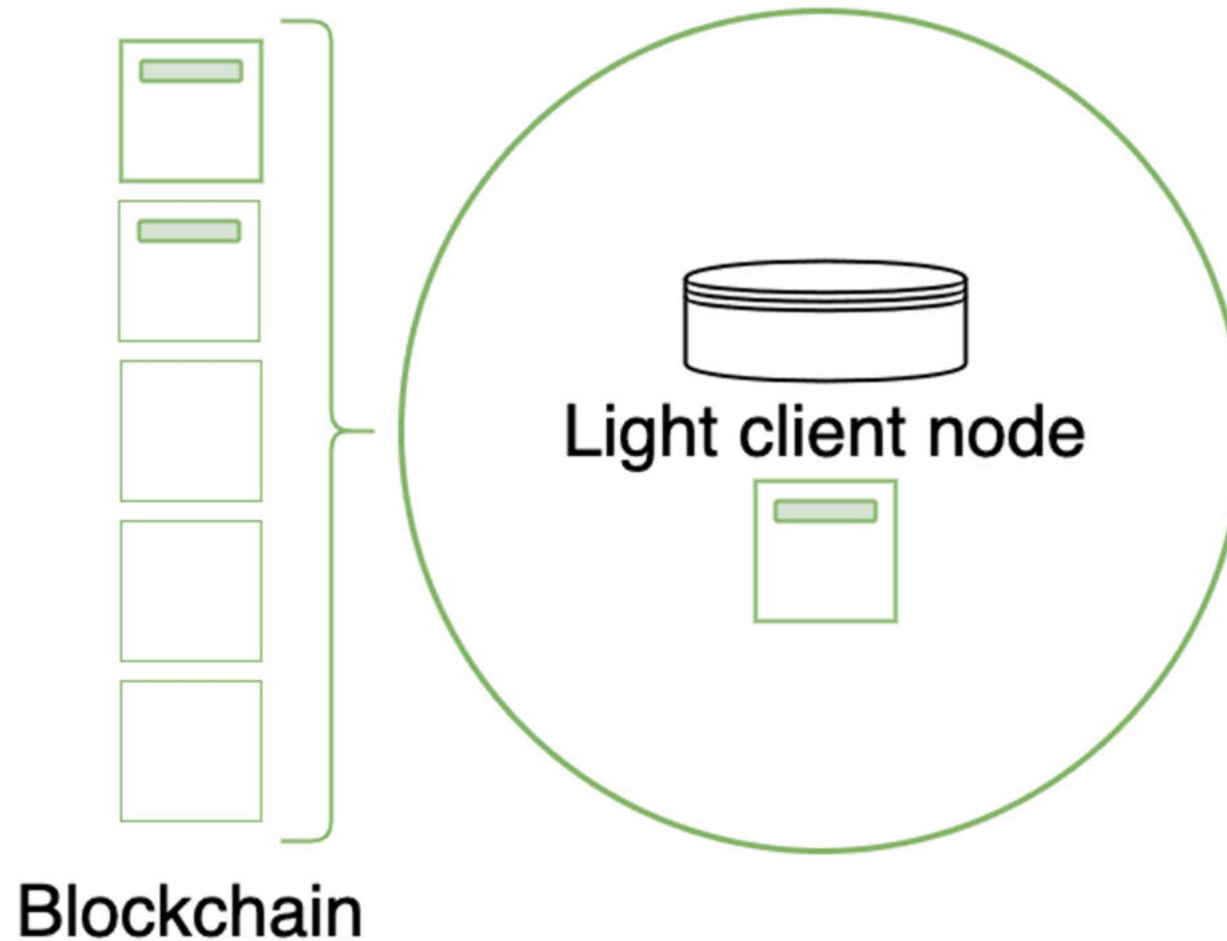
Archive Nodes

Archive all blocks



Light client nodes

Synchronize without storage



Consensus

Substrate split the consensus into two separate phases:

- Block authorizing: It is the process nodes use to create new blocks
- Block finalizing: It is the process used to handle forks and choose the canonical chain

Default Consensus models:

- Auro
- BABE
- GRANDPA

BABE

- BABE provide slot-based block authoring with a known set of validators and is typically used in proof-of-stake blockchains
- Slot assignment is based on the evaluation of a Verifiable Random Function (VRF)
- Each validators is assigned a weight for an epoch which is broken up into slots and the validators evaluates its VRF at each slot

GRANDPA

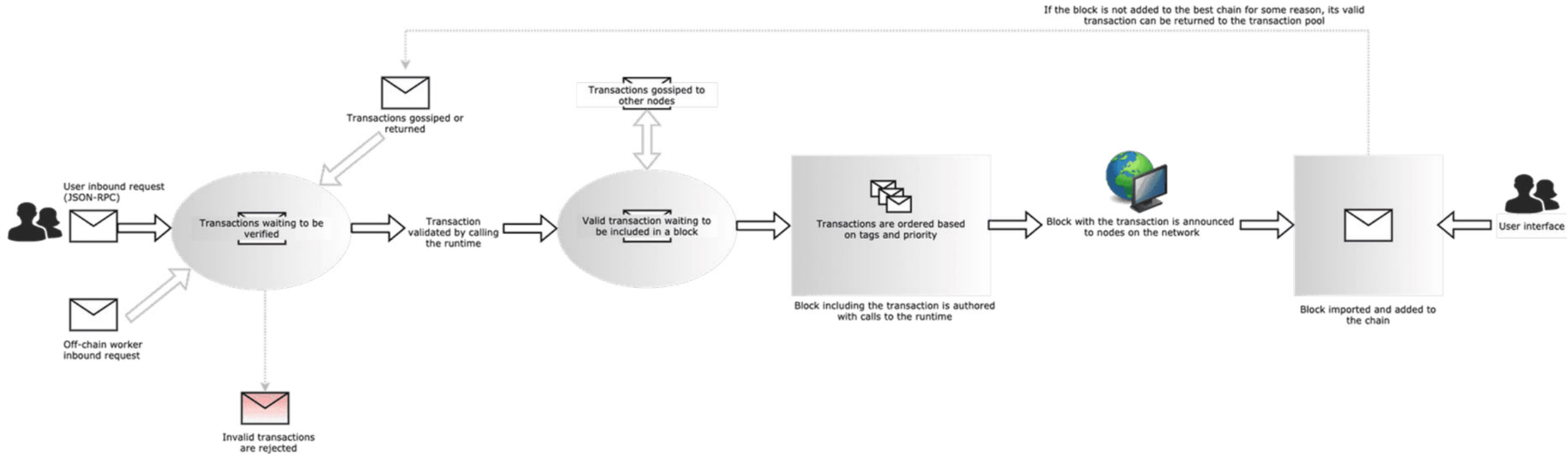
- GRANDPA provides block finalization
- GRANDPA does not author blocks, it just listens to gossip about blocks that have been produced by block authoring nodes
- GRANDPA validators vote on chains, not blocks that they consider best and their votes are applied transitively to all previous blocks.

Transaction and Blocks basics

Transaction provide a mechanism for making changes to state that can be included in a block. Substrate has three distinct transactions:

- Signed transaction: Signed transaction must include the signature of an account sending an inbound request
- Unsigned transaction: Unsigned transaction don't require a signature and don't include any information about who submitted the transaction
- Inherent transaction: Inherent transaction are special type of unsigned transaction. They can only be inserted into a block by the block authoring nodes

Transaction lifecycle



Why Substrate?

Guiding principles: Reduce complexity & Lower Barrier to entry to blockchain technology

Designed with goals in mind:

- Flexible
- Open
- Interoperable
- Future-proof

Now what?

- You can get started with a pre-configured node and start a solo chain with no code.
- You can use Substrate to build a custom solo chain that runs independent of any other blockchains.
- You can couple your Substrate chain to a relay chain like Polkadot or Kusama.

Substrate: Setting up a LOCAL dev environment

To set up Substrate for blockchain development, you will need to follow these basic steps:

1. Install the latest version of Rust, a programming language that Substrate is built on. You can install Rust by following the instructions on the Rust website (<https://www.rust-lang.org/tools/install>).
2. Install Substrate by running:

2. Install Substrate by running the following command:

```
cargo install --git https://github.com/paritytech/substrate.git
```

3. Create a new Substrate project by running the following command: This will create a new directory with the default Substrate node template.

```
substrate --template node
```

Substrate: Setting up a LOCAL dev environment

4. We then build the node using cargo from the project directory.

A terminal window with a dark background. The text 'cargo build' is written in a light blue monospace font. In the top right corner, there is a small icon of a document and the text 'Copy code' in a light gray font.

```
cargo build
```

5. After the node is built, it can be started by running:

A terminal window with a dark background. The text './target/debug/substrate --dev' is written in a light blue monospace font. In the top right corner, there is a small icon of a document and the text 'Copy code' in a light gray font.

```
./target/debug/substrate --dev
```

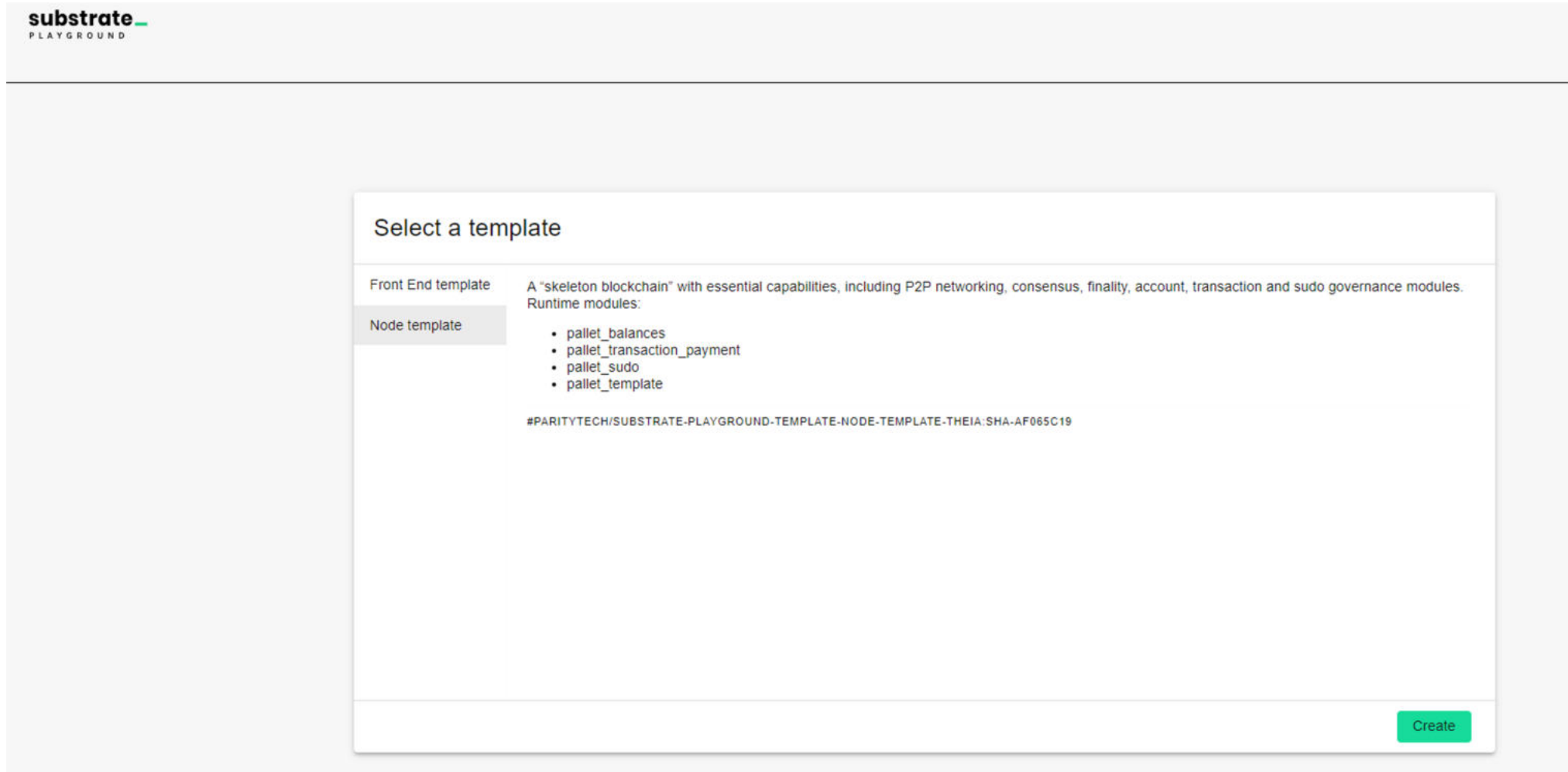
This starts a local dev chain that can be leveraged for testing and ideation.

Substrate Playground: Online IDE

Provides an online dev environment for quick ideation and skills building that provides to templates to start with:

- **Node template:** A “skeleton blockchain” with essential capabilities, including P2P networking, consensus, finality, account, transaction and sudo governance modules. Runtime modules:
 - pallet_balances
 - pallet_transaction_payment
 - pallet_sudo
 - pallet_template
- **Front End template:** A modular UI built with ReactJS to act as a front-end to the Substrate Node Template. It contains all necessary components to interact with the Node Template runtime. Components include:
 - Pallet interactor
 - Events
 - Balances
 - Upgrade runtime

Substrate Playground: Online IDE



Substrate Node Template

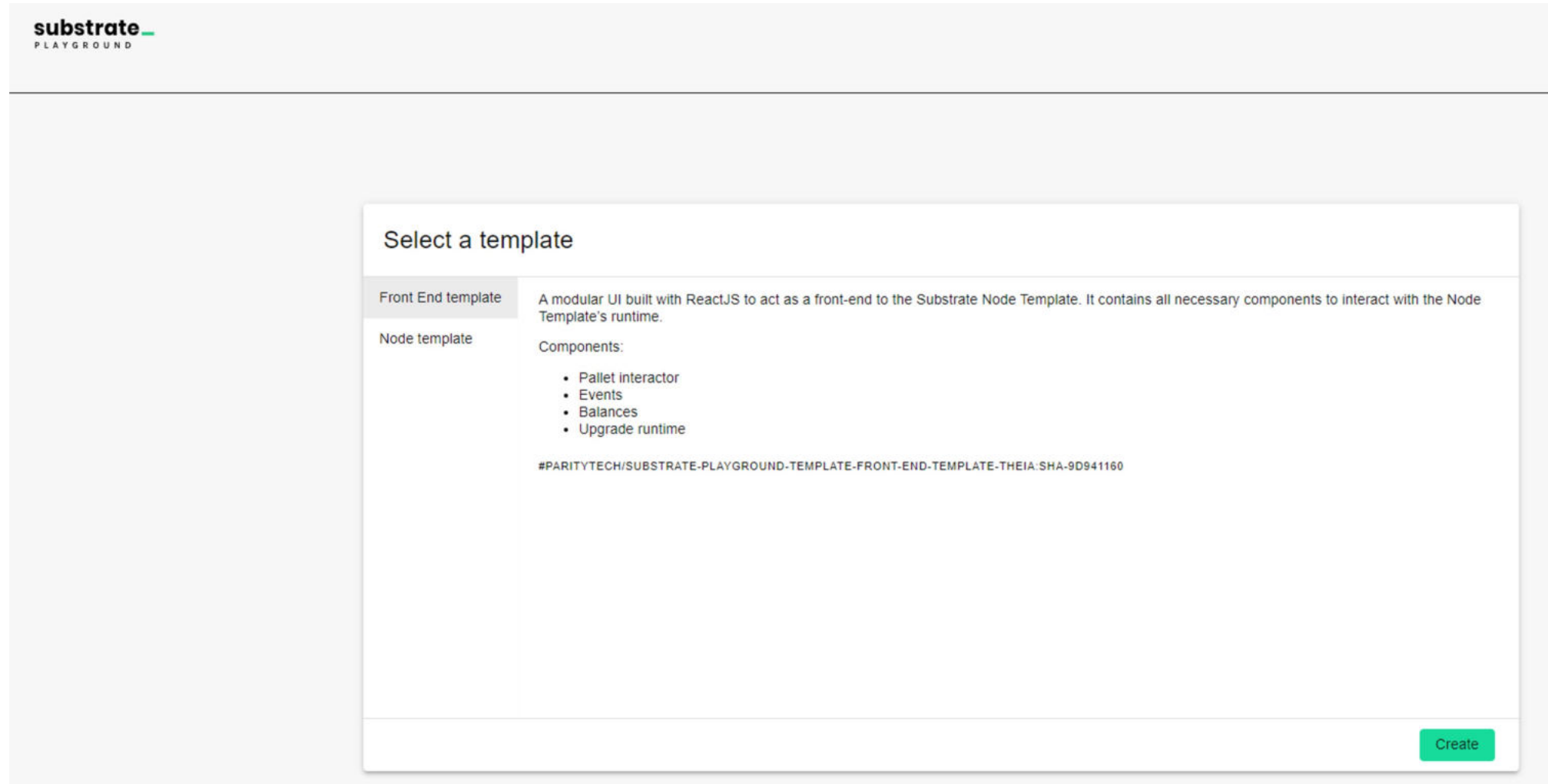
The screenshot displays the Substrate Playground interface. On the left, the Explorer panel shows the project structure for the 'runtime' directory, including 'src' (containing 'lib.rs'), 'Cargo.toml', and 'README.md'. The main editor area shows the 'lib.rs' file with the following content:

```
1  #![cfg_attr(not(feature = "std"), no_std)]
2  // 'construct_runtime!' does a lot of recursion and requires us to increase the limit to 256.
3  #![recursion_limit = "256"]
4
5  // Make the WASM binary available.
6  #![cfg(feature = "std")]
7  include!(concat!(env!("OUT_DIR"), "/wasm_binary.rs"));
8
9  use pallet_grandpa::{
10     fg_primitives, AuthorityId as GrandpaId, AuthorityList as GrandpaAuthorityList,
11 };
12 use sp_api::impl_runtime_apis;
13 use sp_consensus_aura::sr25519::AuthorityId as AuraId;
14 use sp_core::{crypto::KeyTypeId, OpaqueMetadata};
15 use sp_runtime::{
16     create_runtime_str, generic, impl_opaque_keys,
17     traits::{AccountIdLookup, BlakeTwo256, Block as BlockT, IdentifyAccount, NumberFor, Verify},
```

Below the editor, the terminal shows the execution output for the 'playground@session-tcooksey1972: ~/workspace' session, displaying block preparation and consensus session details for proposals at 37 and 38.

```
2022-12-22 21:28:36 [Prepared block for proposing at 37 (3 ms) [hash: 0x5b610c694fd9da5985f56725c00e324c04d274484d971f5e592a2ae2c2414be8; parent_hash: 0x7f33_34f8; extrinsi
cs (1): [0xddde8_d4eb]]
2022-12-22 21:28:36 Idle (0 peers), best: #36 (0x7f33_34f8), finalized #34 (0xee02_0b84), I 0 I 0
2022-12-22 21:28:36 Pre-sealed block for proposal at 37. Hash now 0x25919e51c10256c674ec4e27e87ece75c1d82c84e8a5d4a59343e4d35332e62e, previously 0x5b610c694fd9da5985f56725
c00e324c04d274484d971f5e592a2ae2c2414be8.
2022-12-22 21:28:36 Imported #37 (0x2591_e62e)
2022-12-22 21:28:41 Idle (0 peers), best: #37 (0x2591_e62e), finalized #35 (0xb1a8_9890), I 0 I 0
2022-12-22 21:28:42 Starting consensus session on top of parent 0x25919e51c10256c674ec4e27e87ece75c1d82c84e8a5d4a59343e4d35332e62e
2022-12-22 21:28:42 Prepared block for proposing at 38 (3 ms) [hash: 0x6055e80ba502219cd9ed2d15d6fb1ed4ef186d70c1c4c29d7e73113c038d60c0; parent_hash: 0x2591_e62e; extrinsi
cs (1): [0x5e20_1000]]
2022-12-22 21:28:42 Pre-sealed block for proposal at 38. Hash now 0x533580fb67f218f60437c18f2ee6e3adee1e97fbcbbc56cfb1783d86d8632f58, previously 0x6055e80ba502219cd9ed2d15
d6fb1ed4ef186d70c1c4c29d7e73113c038d60c0.
2022-12-22 21:28:42 Imported #38 (0x5335_2f58)
2022-12-22 21:28:46 Idle (0 peers), best: #38 (0x5335_2f58), finalized #36 (0x7f33_34f8), I 0 I 0
2022-12-22 21:28:48 Starting consensus session on top of parent 0x533580fb67f218f60437c18f2ee6e3adee1e97fbcbbc56cfb1783d86d8632f58
2022-12-22 21:28:48 Prepared block for proposing at 39 (3 ms) [hash: 0x4ce14feedd1519ba52689e40e4154cd1d0ddb61002b009e02d9025333c84bd80; parent_hash: 0x5335_2f58; extrinsi
cs (1): [0x7770_3f11]]
2022-12-22 21:28:48 Pre-sealed block for proposal at 39. Hash now 0x37f7aef6feabc6ca044271ad18634900bb4e1b22095a472c29db34f3ac04366b, previously 0x4ce14feedd1519ba52689e40
e4154cd1d0ddb61002b009e02d9025333c84bd80.
2022-12-22 21:28:48 Imported #39 (0x37f7_366b)
```

Substrate Playground: Online IDE



Substrate Playground: Existing Sessions

substrate
PLAYGROUND

Existing session

Node template
Started 27min ago (17min left)
Phase: *Running*

Environment		Ports		
Name	Value	Name	Path	Value
		wss	/wss	9944

Stop Connect

Intro to PolkadotJS

Polkadot JS is a collection of tools for interacting with the Polkadot network

It allows developers to build applications that can read and write to the Polkadot blockchain

Key Features of Polkadot JS


- Support for multiple blockchain runtime environments (e.g. Substrate, Ethereum)
- Interoperability with other blockchains on the Polkadot network
- A user-friendly API for building decentralized applications

Apps UI

The Polkadot-JS UI, also known as the Apps UI, is a hosted application that loads in your browser.

Polkadot-JS Apps has many capabilities that go beyond basic wallet functions such as account creation and sending or receiving transactions.

The UI has a standard DNS hosted version, which always has the latest features, and an IPFS version that is less frequently updated but is more decentralized.



Polkadot
polkadot/9230
#10,975,307

Accounts

Network

Governance

Developer

Settings 2

GitHub

Wiki

Chain state

Storage

Constants

Raw storage

selected constant query ?

balances

▼

existentialDeposit: u128

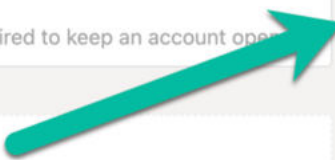
The minimum amount required to keep an account open

+

const balances.existentialDeposit: u128

10,000,000,000


×



Extension

The Polkadot-JS extension is a simple tool for managing accounts in a browser extension and allowing the signing of extrinsics using these accounts.

The Polkadot-JS extension is not made for users to interact with on-chain functions as one would find through a wallet app. The extension acts as a robust key-store and thus acts as an account manager for Substrate-based accounts. It also allows for the signing of transactions with those accounts.



Polkadot
polkadot/9230
#10,975,616

Accounts

Network

Governance

Developer

Settings 2

GitHub

Wiki

Parity Polkadot v0.9.24
api v8.10.1
apps v0.116.2-72-x


Settings

General

Metadata 2

Translate

extensions

upgradable extensions
polkadot-js 0.44.1

Update metadata

Polkadot JS Resources

- Official documentation: <https://polkadot.js.org/>
- GitHub repository: <https://github.com/polkadot-js/api>
- Join the community on Discord: <https://discord.gg/6FpUKcK>

FRAME

Runtime Storage

The FRAME storage data structures can be used to read or write any value that can be encoded by the SCALE codec. The storage module provides the following types of storage structures:

- Storage value: to store any single value such as u64
- StorageMap: to store a single key value mapping
- StorageDoubleMap: to store values in a storage map with two keys
- StorageNMap: to store value in a map with any arbitrary number of keys

Weights

- A convention used in Substrate-based blockchains to measure and manage the time it takes to validate a block
- The maximum block weight should be equivalent to one-third of the target block time with an allocation of:
 - one third for the block construction
 - one third for network propagation
 - one third for import and verification

Pallet coupling

In Substrate, tight pallet coupling and loose pallet coupling are used to describe how a pallet can call functions in another pallet.

- Tight coupling is when two groups of classes are dependent on each other
- Loose coupling is when a class uses an interface that another class exposes

Tightly couples pallets

Tight pallet coupling is only used if a pallet requires inheriting its coupled counterpart as a whole rather than specific types or methods.

All FRAME pallets are tight coupled to the frame_system pallet.

```
pub trait Config: frame_system::Config +  
some_pallet::Config {  
    // --snip--  
}
```

Loosely coupled pallets

In loose pallet coupling, the traits and function interfaces that certain types need to be bound by can be specified.

Assume a pallet that can access account balances and make transfers to another account. This defines a Currency trait, which has an abstract function interface that will implement the actual transfer logic later

```
pub trait Currency<AccountId> {  
    // -- snip --  
    fn transfer(  
        source: &AccountId,  
        dest: &AccountId,  
        value: Self::Balance,  
        // don't worry about the last parameter for now  
        existence_requirement: ExistenceRequirement,  
    ) -> DispatchResult;  
}
```

Events

A pallet can emit events when it wants to send notification about the changes or conditions in the runtime to external entities like users, chain explorers or dApps

Errors

Runtime code should explicitly handle all error cases.

Each FRAME pallet can define a custom `DispatcherError` by using the

```
# [pallet::error]
# [pallet::error]
pub enum Error<T> {
    /// Error names should be descriptive.
    InvalidParameter,
    /// Errors should have helpful documentation
    associated with them.
    OutOfSpace,
}
```

Origins and calls

The runtime origin is used by disposable functions to check where a call has come from

Substrate defines three raw origins which can be used in the runtime pallets:

```
pub enum RawOrigin<AccountId> {  
    Root,  
    Signed(AccountId),  
    None,  
}
```

Metadata

- Substrate nodes provide an RPC call, `state_getMetadata` that returns a complete description of all the types in the current runtime
- Client applications use the metadata to interact with the node, to parse responses, and to format message payloads sent to the node
- This metadata includes information about a pallet's storage items, transactions, events, errors and constants.

Testing

- Testing is the key to making the chain accessible to a wider audience and economically viable in the network
- Tools and techniques for testing the blockchain logic:
 - Unit testing
 - Debug
 - Benchmark
 - Simulate Parachains
 - Check runtime

SCALE

- Substrate uses a lightweight and efficient encoding and decoding program to optimize how data is sent and received over the network
- The program used to serialize and deserialize data is called the SCALE codec, with SCALE being an acronym for **S**imple **C**oncatenated **A**ggregate **L**ittle-**e**ndian
- The SCALE codec is a critical component for communication between the runtime and the outer node

Benchmark

- Substrate and FRAME provide a flexible framework for developing custom logic for the blockchain
- This flexibility enables to design complex and interactive pallets and implement sophisticated runtime logic
- Determining the appropriate weight to assign to the functions in the pallets is difficult. Benchmarking enables to measure the time it takes to execute different functions in the runtime and under different conditions.

Hands on Workshops