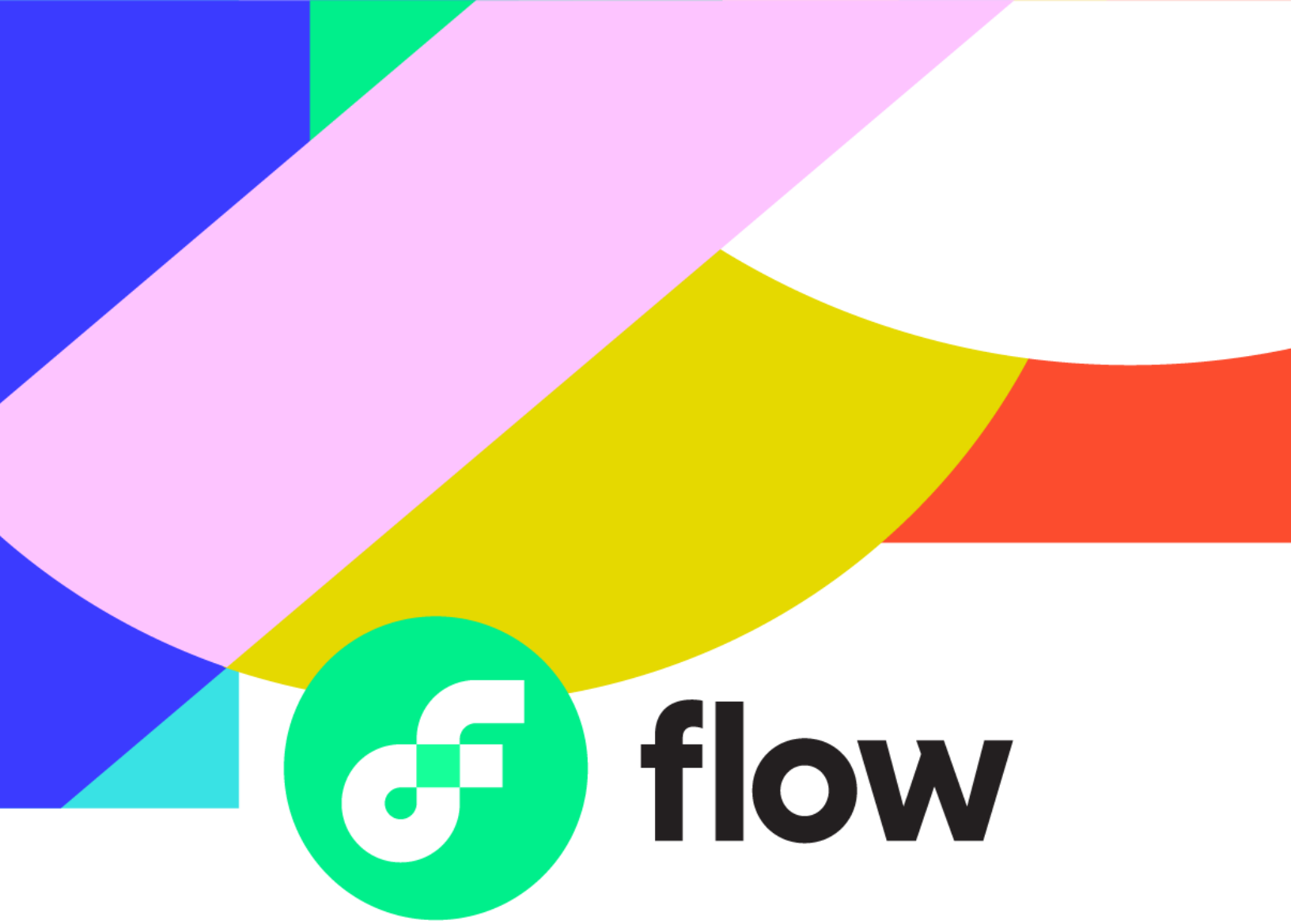


flow

Primer

Certain statements in this presentation constitute forward-looking statements. When used in this presentation, the words “may,” “will,” “should,” “project,” “anticipate,” “believe,” “estimate,” “intend,” “expect,” “continue,” and similar expressions or the negatives thereof are generally intended to identify forward-looking statements. Such forward-looking statements, including the intended actions and performance objectives of the Company and its affiliates involve known and unknown risks, uncertainties, and other important factors that could cause the actual results, performance, or achievements of the Company in its development of Flow, the blockchain, the network and the tokens as well as the features of Flow, the blockchain, the network and the tokens described herein to differ materially from any future results, performance, achievements, functionality of features expressed or implied by such forward-looking statements. No representation or warranty is made as to future performance or such forward-looking statements. All forward-looking statements in this presentation speak only as of the date this presentation was provided to you. The Company expressly disclaims any obligation or undertaking to disseminate any updates or revisions to any forward-looking statement contained herein to reflect any change in its expectation with regard thereto or any change in events, conditions, or circumstance on which any such statement is based.

Flow: Blockchain for Open Worlds

Flow is a fast, decentralized, and developer-friendly blockchain, designed as the foundation for a new generation of games, apps, and the digital assets that power them.

This makes Flow the platform for a new digital economy, open for anyone to join, participate in, and benefit from. Digital assets on Flow are fully interoperable, and applications on Flow can be assembled like Lego blocks to power apps serving billions of people, from sports fans to businesses with mission-critical requirements.

For creators, developers, and artists, Flow opens up new business models and organic ways to engage and reward communities. For users, applications built on Flow offer portable data, real ownership of digital assets, and the opportunity to share in the value they help create.

Flow was developed by Dapper Labs, the creators of some of the most successful blockchain applications to date including CryptoKitties, Dapper wallet, and NBA Top Shot. The Flow network is powered by FLOW token, a low-inflation native cryptocurrency. The FLOW token is designed to enable value exchange across the network and to secure the value of everything built on top. For more on the FLOW token, read our token economics primer: **hoo.ps/tep**

There are three principles that together make Flow different than anything else out there:

- **Pipelining:** unique among crypto networks, Flow has a four-node architecture that uses pipelining to drive massive improvements in speed, throughput, and cost, while preserving decentralization. No sharding or “layer two” needed.
- **Composability:** Flow’s developer-first design is made for a world of open applications and permissionless composability. On Flow, creators can build and remix new products and services at a moment’s notice to add value to their communities.
- **Community:** with easy payments integrations and protocol-level features to keep consumers safe, Flow lets developers focus on the core value of their products while benefiting from the vibrant ecosystem developing around the network.

These principles are outlined on the following pages along with high-level outlines of the Flow system as a whole. For more details refer to our technical papers: **hoo.ps/tecp**

Pipelining: Four-Node Architecture

In first-generation smart contract blockchains like Ethereum, every node in the network performs all of the work associated with processing every transaction in the chain – and stores the entire network history or state (account balances, smart contract code, etc). This is highly secure but also incredibly inefficient.

Most second-generation crypto networks focus on improving performance in one of two ways: either compromising decentralization by limiting the nodes that can participate to powerful servers; or compromising composability by breaking up the network through mechanisms such as sharding. The former is vulnerable to platform risk and cartel-like behavior; the latter effectively saddles the hardest part of scaling to application developers.

Flow offers a new path: pipelining applied to crypto networks.

From manufacturing to CPU design, pipelining is a common technique for dramatically scaling up productivity. Flow applies pipelining to blockchains by separating the jobs of a validator node into four different roles: Collection, Consensus, Execution, and Verification. This separation of labor between nodes is vertical (across the different validation stages for each transaction) rather than horizontal (across different transactions, as with sharding).

In other words, every validator node still participates in the validation of every transaction, but they do so only at one of the stages of validation. They can therefore specialize — and greatly increase the efficiency — for their particular stage of focus.

This allows Flow to scale to thousands of times higher throughput and lower cost while maintaining a **shared execution environment** for all operations on the network. In database terms, smart contracts and user accounts on Flow can always interact with each other in one atomic, consistent, isolated, and durable (ACID) transaction. This ensures full composability and the optimal user experience, letting developers easily build on each other's work.

Background: Problems with Sharding

Most proposals aim to improve the scalability of blockchains by fragmenting them into inter-connected networks: commonly referred to as shards or sidechains. These approaches remove serializability (i.e., ACID) guarantees common in database systems.

Loss of ACID guarantees makes building apps that need to access data across fragments far more difficult and error-prone. Interactions between smart contracts become very complicated, and even individual large-scale applications would have to resort to complex mechanics to scale across shards due to latency issues and higher transaction failure rates.

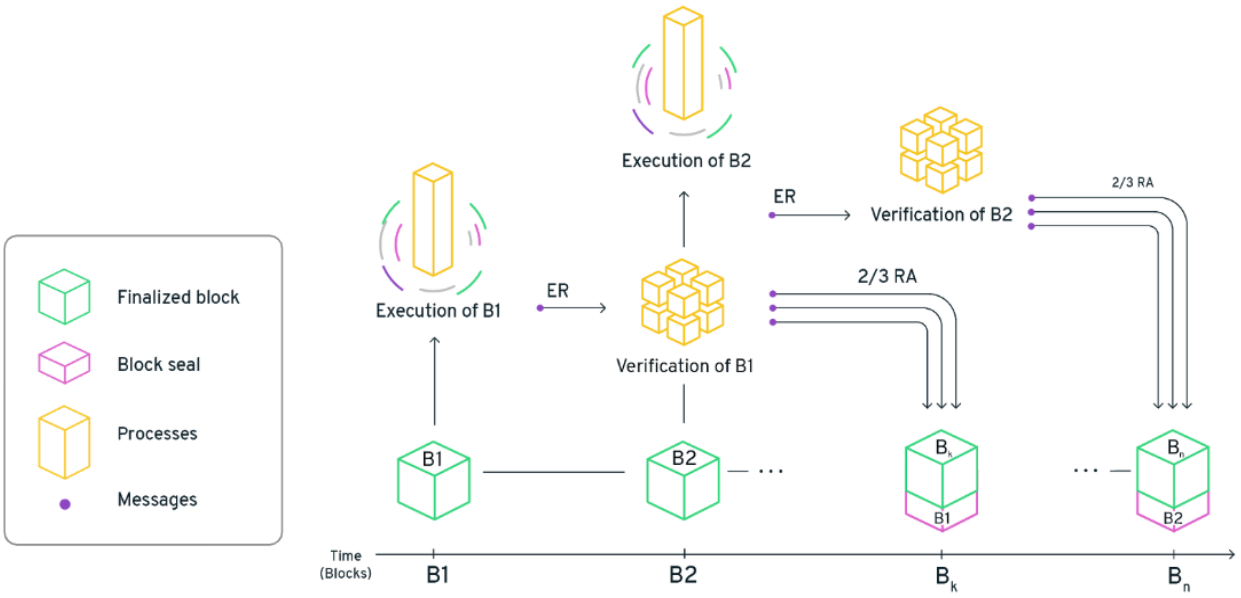
The combination dramatically limits the kinds of applications possible on the network as well as their network effects. **Sharding effectively saddles the hardest part of scaling the blockchain onto application developers, rather than solving it at the protocol level.**

A simple user action (purchasing a hat for a CryptoKitty using a stablecoin like TUSD) can take twelve transactions and seven blocks on a sharded blockchain. In an unsharded, ACID-compliant environment like Flow, the same action, and many more complex than it, can be handled by one atomic transaction in a single block. Even more troubling is the increased attack surface and complexity: it will be much harder to design, test, and harden the smart contract code on a sharded blockchain.

Separating Consensus from Compute

The core insight that led to architecture of Flow is that we can separate non-deterministic processes from deterministic ones and assign each to different types of nodes based on their technical capabilities to dramatically increase the blockchain throughput and solve several user and developer experience problems with existing networks at the same time. Our realization was that tasks within a blockchain can be divided into two types:

- Non-deterministic (or “subjective”) tasks, such as determining the presence and order of transactions in the blockchain
- Deterministic (or “objective”) tasks, such as computing the result of those ordered transactions once it has been determined



Non-deterministic tasks require a coordinated consensus process (like Proof of Work or Proof of Stake). Deterministic tasks, on the other hand, always have a single, objectively correct outcome. The critical insight behind Flow's architecture was that the single biggest bottleneck to blockchain performance is the deterministic task of executing transactions after they've already been included in a block, and not the subjective process that requires consensus, i.e. the formation of the block itself. This insight is outlined in our first technical paper: Separating Consensus and Compute.

Multi-Role Validator Node Architecture

Flow pipelines the work of a blockchain miner or validator across four different validator node roles that all require staking, a separation of concerns that significantly reduces redundancy:

- Consensus Nodes decide the presence and order of transactions on the blockchain
- Execution Nodes perform the computation associated with each transaction
- Verification Nodes are responsible for keeping the Execution Nodes in check
- Collection Nodes enhance network connectivity and data availability for applications

Flow is designed such that even a single honest validator node can punish and trigger recovery from invalid data introduced by dishonest Collection or Execution Nodes.

Consensus and Verification Nodes together are the foundation of security in the Flow network and leverage cryptoeconomic incentives to hold the rest accountable. These validators can optimize for security and decentralization: the roles of Consensus and Verification are streamlined to allow high levels of participation, even by individuals with consumer-grade hardware running on home internet connections. Consensus nodes currently run a variant of HotStuff, one of the most proven proof of stake algorithms.

Execution and Collection Nodes, on the other hand, do work that is fully deterministic – making them less vulnerable to attack. The work of these nodes is also verified and held accountable by the other node types. These node roles can therefore safely optimize for security and scalability, allowing the network to scale. Operating these nodes requires dedicated server hardware in a professionally managed data center.

Specialized Proofs of Confidential Knowledge (SPoCKs)

Specialized Proofs of Confidential Knowledge (SPoCKs) are a new cryptographic technique developed by the Flow team and formally defined in our Technical Papers. SPoCKs allow any number of participants ("provers") to demonstrate to a third-party observer that they each have access to the same *confidential knowledge*. These proofs are non-interactive and don't leak any information about the confidential knowledge itself. Each prover's SPoCK is *specialized* to them and can't be copied or forged by any other prover.

Flow uses SPoCKs to address the Verifier's Dilemma by requiring Execution and Verification Nodes to "show their work" to Consensus Nodes. In order to get paid, the former nodes need to provide a SPoCK showing access to confidential knowledge that can only be obtained by executing all of the transactions assigned to them.

The construction of SPoCKs is based on the BLS signature scheme, using the pairing properties underlying that technique in a novel way to provide a proof of knowledge that doesn't leak any data about the underlying secret. Flow nodes also use BLS keys to authenticate protocol messages because BLS signatures allow for highly efficient aggregation. The same public and private key pairs can be used in each case, ensuring that SPoCK generation is tightly coupled to node identity for enhanced security.

Composability: Developer-First Experience

Our experience developing blockchain applications like CryptoKitties and the Dapper Smart Contract wallet has led us to incorporate a number of improvements to developer ergonomics directly into the protocol layer on Flow. Several are outlined below.

Cadence

Cadence is the first ergonomic, resource-oriented smart contract programming language.

While existing programming environments can be used to keep track of asset ownership, they are typically used in scenarios where they are *reflecting* ownership rather than defining it directly. Public blockchains are unique in that they are explicitly designed to manage ownership of digital assets with scarcity and full access control. Digital assets on public blockchains behave like physical assets: they cannot be copied or counterfeited, only moved.

Last year, the Flow team was investigating the use of Linear Types in the context of blockchains, following academic research into better smart contract languages. At just about the same time, the Libra team defined a new programming model for Move based around a new ownership model inspired by Linear Types: resources. Resources are a new way of representing asset ownership and the properties of crypto-enabled digital assets directly in the programming language. From the Move paper's introduction:

> *The key feature of Move is the ability to define custom resource types. Resource types are used to encode safe digital assets with rich programmability.*

We were so struck by the power of Resource-Oriented Programming that it's one of the defining features of Cadence, a programming language designed specifically for the new paradigm of crypto-enabled applications.

Resource-oriented programming is a new paradigm, designed to be secure and easy-to-use. For the first time, developers can create uniquely durable digital artifacts where ownership is tracked by the language itself, enabling a powerful new category of applications.

As the first high-level resource-oriented programming language, Cadence has a comfortable, ergonomic syntax making it very easy to read. It uses a strong, static type system to minimize runtime errors, and allows all methods, interfaces, and transactions to include pre- and post-conditions to enforce expected behaviour. This has resulted in a language that is easier to learn, significantly easier to audit, and ultimately much more productive than any current alternatives. You can start learning Cadence on Flow Playground: **play.onflow.org**

Open source tooling

The Flow team has open sourced a series of tools to help developers get started:

Flow Go SDK: the Go SDK is a great tool for developers looking for backend integration with scalability in mind. Go is one of the most popular backend programming languages when performance is a top priority and has been the go-to choice for Dapper Labs.

Flow JavaScript SDK: for frontend developers, our JavaScript SDK will allow you to easily integrate and interact with Flow. Develop without using ABIs, construct composable interactions and create dapps that delight your users. We think you're going to love building with our JavaScript SDK.

Visual Studio Code Extension: interact with Flow and use the Cadence language natively in Visual Studio Code. Statically check your Cadence code for errors and test your smart contracts without leaving the comfort of this industry-leading IDE.

Flow Playground GUI: the hosted, in-browser development environment where users can learn and try out Cadence smart contract language without any setup needed. We make it easy for any new developer to get a taste of Cadence, the powerful new language for smart contract development.

Standards Proposals: FTs (Fungible tokens) and NFTs (Non-fungible tokens) are the Flow equivalent of Ethereum's ERC-20 and ERC-721 tokens, respectively.

Upgradable Smart Contracts

One of the most important promises made by smart contract platforms is that users can trust the smart contract code instead of trusting the smart contract authors. This aspect of blockchains unlocks use cases that we are only beginning to explore, the most impactful of which might be the concept of open services and composability.

In their first incarnation, smart contract platforms were designed such that contract code could never be changed after it was released. This is the most straightforward method to achieve the goal: If the code can't be changed, even by the original authors, you clearly don't need to trust the authors after the code is launched.

Unfortunately, software is hard to get right the first time. There are no shortage of examples of smart contracts that – even with incredibly talented teams and motivated communities – had subtle problems that led to a massive loss of funds.

Many developers have expressed the desire to fix or improve a smart contract after it has been deployed, and several have gone to a lot of time and trouble to build some mechanism into their smart contract to allow for upgrades or migrations. But having each developer "roll their own" mechanism for upgradability adds complexity and makes those smart contracts harder to trust.

On Flow, we allow smart contracts to be deployed to the mainnet in a "beta state", where the code can be incrementally updated by the original authors. Users will be alerted to the unfinished nature of this code and can choose to wait until the code is finalized before trusting it. Once authors are confident that their code is safe, they can irrevocably release their control on the contract, and it becomes perfectly immutable for the rest of time.

This system balances the needs of users to be informed about what kind of code they are dealing with – whether or not an application or smart contract is truly trustless – while allowing developers the flexibility to tweak their code for a limited time after shipping.

Fast, Deterministic Finality

From the standpoint of end users, the speed of a blockchain is most practically measured by the time it takes before they (or their client software) can be confident their transaction is permanently included in the chain. This is commonly referred to as “finality”. In Bitcoin, most people define finality as six block confirmations which can take more than an hour. Ethereum improves on this by achieving **probabilistic finality** after about 6 minutes.

On Flow, deterministic finality is achieved within seconds: once Consensus Nodes determine which block a transaction will be a part of, user agents can, in most cases, execute the transaction locally and give feedback to the user almost immediately. In cases where results may be influenced by other transactions in the network, users will either choose to trust an Execution Node, using modern APIs to get feedback within seconds, or wait until the results of the transaction are sealed into the blockchain along with all the relevant execution and verification receipts. This process of block sealing and formal observation also happens within seconds.

Built-in Logging Support

Sometimes, the only way to be sure that a complicated piece of software is working as expected is to log its behaviour, in detail, over a long period of time. Existing smart contract platforms don’t include a logging facility for the simple fact that storing a complete log record of the entire blockchain is completely intractable. Too much data!

Flow recognizes that since all smart contract transactions are fully deterministic, there is no need to store the actual logs for every transaction inside the network. Instead, Flow simply marks which transactions would have produced log messages for which topics. If someone wants to “examine” the logs, they can query the blockchain for the subset of transactions tagged with a particular topic and then re-run the transactions locally to generate those logs for analysis. This technique also makes event logging dramatically more efficient.

Community: Onboarding and Rewards

In addition to mainstream-ready payment on-ramps (from other crypto tokens as well as fiat currencies), the Flow network makes it easy to build applications that people want to use:

Smart User Accounts: no more seed words or lost keys

Flow is designed with flexibility in mind. Over the past year, Dapper Labs has pioneered a variety of usability enhancements to the Ethereum account model as part of the Dapper Smart Contract Wallet. Those enhancements are part of the native account model on Flow:

- Optional, modular, smart contract functionality built into every Flow wallet
- Supports automated processes or more sophisticated authorization controls, in turn enabling good user experience. For example, dapps can easily make sure consumers never lose their assets – or access to their accounts – with secure account recovery flows
- Added security through optional multiple signature support, with the ability to cycle out old keys regularly to avoid security leaks

Human Readable Security

On current networks, it's nearly impossible for an app or wallet software to provide a human-readable message clearly outlining what permissions they're giving when authorizing a transaction. The Flow transaction format makes strong guarantees about what kinds of changes a transaction can and can not make. This makes it easy for the wallet to ensure users are making informed decisions about what they are approving.

It will be up to wallet software to display this information to the users, but by making the Flow transaction format easy to statically analyze, we create the possibility for a more transparent transaction approval process.

Engagement and Reward Programs

Flow is committed to a world of open ecosystems: a world where software developers, content creators, and consumers alike are appropriately incentivized and rewarded for the value they contribute to the network.

The technical architecture is just one example of how Flow will ensure inclusivity and participation at the protocol level: our community evangelism and efforts toward governance are equally if not more important.

Node Operator Fees and Rewards

On Flow, validator node operators supporting the network receive a portion of the transaction fees that pass through the system proportional with the work they do and their associated stake. Unlike miners or validators for other blockchains, validators on Flow can get started as Consensus or Verification Nodes with relatively cheap hardware, ensuring broad, equitable participation and decentralization.

In the early years of the network when fees are low, the network will provide additional rewards to node operators proportional with their efforts and associated stake.

Developer Ecosystem

A healthy and vibrant ecosystem is the most important long-term determinant of success for a blockchain. It is a fundamental requirement that Flow engages with a large and diverse set of stakeholders. Most importantly, this extends beyond the token holders and includes the developers and ecosystem partners that choose to build on top of the network.

In addition to a technical design optimized for developer experience and performance, the Flow team is taking additional steps to ensure a healthy ecosystem:

- **Developer Alpha Program:** over the coming months, the Flow team will begin demonstrating the capabilities of the network to interested blockchain developers for

technical feedback. Whether you're an independent hobbyist or a venture-backed powerhouse, the Flow team wants to hear from you.

- **Ecosystem Development:** a portion of Flow tokens will be set aside for ecosystem development to bootstrap adoption and reward early participants in the network. These participation rewards will be distributed via a number of different programs including competitions, hackathons, and contributions to open source development. In addition to accelerating adoption, setting aside a portion of tokens for long-term ecosystem development also ensures a path to diversifying and decentralizing network participation and governance, ensuring global access from a variety of participants.

Content Partners

Blockchain lets brands and influencers connect directly with their fans in new ways. Digital scarcity and true ownership of assets create the space for gamified social experiences that go beyond individual apps, creating collector economies around every unique IP. Flow is working with independents that are breaking the mold as well as some of the world's leading entertainment studios, IP holders, and publishers to ensure our platform serves their needs.

Flow will sponsor the creation of an entertainment industry council for C-level executives from global IP holders, game publishers, entertainment studios, and cultural influencers. The council will help identify risks and opportunities, remove friction-points for consumer adoption, and help catalyze a healthy global entertainment ecosystem.

Join the community

Together with our community Flow can power an open and trustworthy internet for billions of consumers. We're looking for developers, companies, and ambassadors to bring this new digital world to life. These teams and individuals will work closely with our team and receive mentorship in order to establish sustainable products, services, businesses, and communities on Flow. To get started building on Flow, **head to the docs** and join **discord.gg/flow**.

Further Technical Reading

The architecture of Flow is specified in a series of **Technical Papers**.

Technical Paper 1: Separating Consensus & Compute

The first paper describes the approach at the core of the Flow architecture: splitting consensus (selection and ordering of transactions) from compute (executing each transaction and recording its output) and proves this can dramatically increase throughput without compromising security. In this first paper we analyze how the Flow architecture increases performance, preserves ACID guarantees, and proves that it does not compromise security. The result is a throughput increase by a factor of 56 compared to conventional architectures without loss of safety or decentralization. The paper also notes that a working system based on these ideas must verify the computation (the subject of technical paper 3), but that its key result is applicable regardless of how that problem is addressed.

Technical Paper 2: Block Formation

The second in the series of technical papers formalizes the process of block formation and the Proof of Stake based consensus process in Flow. Flow adapts a variant of the HotStuff consensus algorithm for Consensus Nodes to come to consensus on the blocks they will honor at every block height. This paper also addresses the Consensus role's responsibility in mitigating challenges submitted to the network.

Technical Paper 3: Execution Verification

The third technical paper answers questions posed by the first whitepaper around the verification of computation results. The paper provides a formalization of our verifiable computation scheme with proofs of safety and liveness under reasonable Byzantine assumptions. Although the paper does not explore the possibility, we believe this result could be adapted to other scenarios where Bulletproofs, TrueBit, TEEs, and other verifiable computation schemes are applicable.