

Tab 1

# Gossamer: Core Contributions to Polkadot [Q3/Q4 2025]

**Proposal Date:** 4th June 2025

**Requested amount:** 815\_358 USDC

**Beneficiary address:** 149mJjdQjEBMHHbWDjbLJ7X4e95ps6L35DZVaBzn1raR1EVQ

**Short description:** Gossamer is a team dedicated to implementing the Polkadot Relay Chain Validator, which was developed using Go. As an alternative client implementation, it serves decentralization and robustness of the network by being a full node and a validator node. Ultimately, the heart of Gossamer is its team. This is not just a group of developers writing lines of code. It's a dedicated team of Polkadot protocol engineers and Web3 enthusiasts who have a broad portfolio of projects and have spent years building toward a shared vision: the importance of creating a truly decentralized Web3 future.

**Project Category/Type:** Software development ▾

**GitHub:** [Gossamer](#)

**Website:** [ChainSafe](#)

**Previous treasury proposals:**

- [Treasury Gossamer grant in 2022 \(passed\)](#)
- [Treasury grant in 2024 \(passed\)](#)

**Forum post:** [polkadot.forum](#)



GOSSAMER.



ChainSafe

# 1. Context of the Proposal

The Gossamer team at ChainSafe is a group of technical experts, engaging in protocol-level implementations for the Polkadot ecosystem. The team has formerly received funding from the Web3 Foundation, the Polkadot Treasury and has also been temporarily self-funded through ChainSafe.

This proposal is a request for six months of funding from the Polkadot Treasury for the continued work of the Gossamer team. The ecosystem is rapidly evolving, and a lot of change is to be expected. We are therefore adopting a six-month proposal cycle to maintain agility and responsiveness to best serve the ecosystem.

In this funding period, as detailed in Section 4.2 (Scope of Current Proposal), we will dedicate resources to critical ecosystem needs, specifically, continue building alternative implementations of Polkadot validator, JAM implementation, and enhancing Collator functionality to improve Parachain resilience. We understand the team's commitment as a fundamental technological pillar for advancing the Polkadot ecosystem.

Even though this proposal describes concrete development milestones for our alternative Polkadot client, we're already weaving in bridges into the future. The team is dedicated to fundamentally accompanying and supporting the journey and development of the ecosystem, from Polkadot to Polkadot 2.0 to required JAM implementations, now and in the future.

## 1.1 About ChainSafe and the Gossamer Team

At ChainSafe, we are dedicated to pioneering the development of decentralized and community-oriented technologies that empower users globally. Our mission is to advance web3 potential through open-source innovation, making it more accessible, secure, and sustainable.

Outside Gossamer, some of ChainSafe's development contributions to the Polkadot ecosystem are:

- [Multix](#)
  - An interface to easily manage complex multisigs.
- [Phala SubBridge](#)
  - Bridging data and assets from/to Dotsama and Ethereum.
- [Sygma Substrate Pallets](#)
  - Enables native connectivity for assets and messages, leveraging the Sygma protocol, between EVM and Substrate-based chains.
- [Cypress Plugin - Polkadot Wallet](#)
  - A plugin that enables integration tests with wallets using the popular testing framework, Cypress.
- [Metamask Snap](#)
  - Plugin for interacting with Polkadot dApps and other Substrate-based chains.
- [Chainlink Pallet](#)
  - Integration of Chainlink feed in Substrate-based chains.

Outside development contributions and as part of our multifaceted approach, we are enhancing the Polkadot ecosystem through targeted infrastructure improvements:

- [Snapshot Hosting Services](#)
  - Snapshots are free and allow efficient network synchronization of both Polkadot and Kusama.
- Running Validator Nodes
  - Entering the 1K Validator Program on Kusama, with plans to expand to Polkadot with both Parity and KAGOME validators, and Gossamer once it is ready.
  - These nodes will eventually become a testing ground for Gossamer to demonstrate its capabilities.

Additionally, we continue to be active in our non-technical contributions, such as participating and presenting at Polkadot events like Decoded ([2022](#), [2023](#)), Sub0 ([2024](#)), ParisDot, [Web3 Summit](#), Polkadot Blockchain Academy, JAM Experience, and so on. We are also organizers of Polkadot meetups, both physical and online ([Croatia meetup](#), [CSCON](#)).

## 2. Problem Statement

Client diversity is crucial for robust decentralized networks, ensuring resilience, security, and sustained growth. In decentralized systems, this refers to the use of multiple independent software implementations of a protocol. Recent events, such as the [Kusama dispute storm](#) and the [Ethereum Holesky hard fork bug](#), highlight its importance. In the latter case, multiple client implementations proved critical: while one client encountered a bug, others continued to function correctly, allowing the network to remain operational. Beyond crisis recovery, this diversity also fosters a more robust development environment where additional expertise can prevent bugs before they emerge.

Polkadot Collators are essential infrastructure components whose resilience is often neglected; given the Relay Chain's shared security, uptime is their most critical parameter. Collator downtime leads to missed parachain blocks, a concern amplified as their responsibilities increase with more AssetHub logic. Consequently, with shorter block times also demanding greater performance and reliability, highly dependable collators are increasingly crucial.

Beyond RelayChain validation, research is also being conducted to enhance the resilience of Polkadot Collators. This research leverages the considerable overlap between existing validator functionality and collator requirements, with the goal of ensuring higher uptime and reliability (as detailed in section 4.2.2).

## 3. Correlation with JAM

### 3.1 Gossamer's Role in the JAM Evolution

While Polkadot 2.0 lays a solid foundation, it also sets the stage for JAM, the next evolution of the network. This shift doesn't signal Gossamer's departure from Polkadot, but rather strengthens its commitment to the ecosystem through client diversity and ongoing contribution to security and resilience.

Since JAM's announcement in Q1 2024, part of the Gossamer team has been involved in JAM development. This was completely outside our previous proposal deliverables, and currently, our JAM initiative is funded by ChainSafe. The team's efforts include:

- **JAM Specification Analysis and Development:** Studying the Gray Paper and related material, participation in JAM Implementers meetings, community test vectors and foremost building JAM implementation.
- **PVM Research & Implementation:** Developed a Go-based Polkadot Virtual Machine (PVM) to test features for both Polkadot 2.0 and JAM. Additionally, experiments are being done to compile Go to PVM to build a Go JAM Service SDK.
- **Community Engagement:** Actively facilitating JAM Implementer calls and events, the JAM Implementers DAO, and running an external contributor program to onboard developers from other ecosystems.

Rather than reacting passively, the Gossamer team is proactively building for JAM, awaiting the opening of Milestone 1 submissions to contribute their implementation.

### 3.2 Why This Work is Essential for JAM

#### **Progressive Protocol Development**

JAM shows considerable promise, and we are very optimistic about it. While this is encouraging, we must be pragmatic about the immense undertaking required for full adoption. This extends beyond intricate protocol development to the creation of crucial

tooling and comprehensive SDKs necessary for migrating Polkadot parachains. Given these significant challenges, the full adoption could be at least two years away. Hence, work on Polkadot 2.0 will, for now, continue to be valuable for a formidable amount of time.

### **Current Building Blocks Will Be Reused**

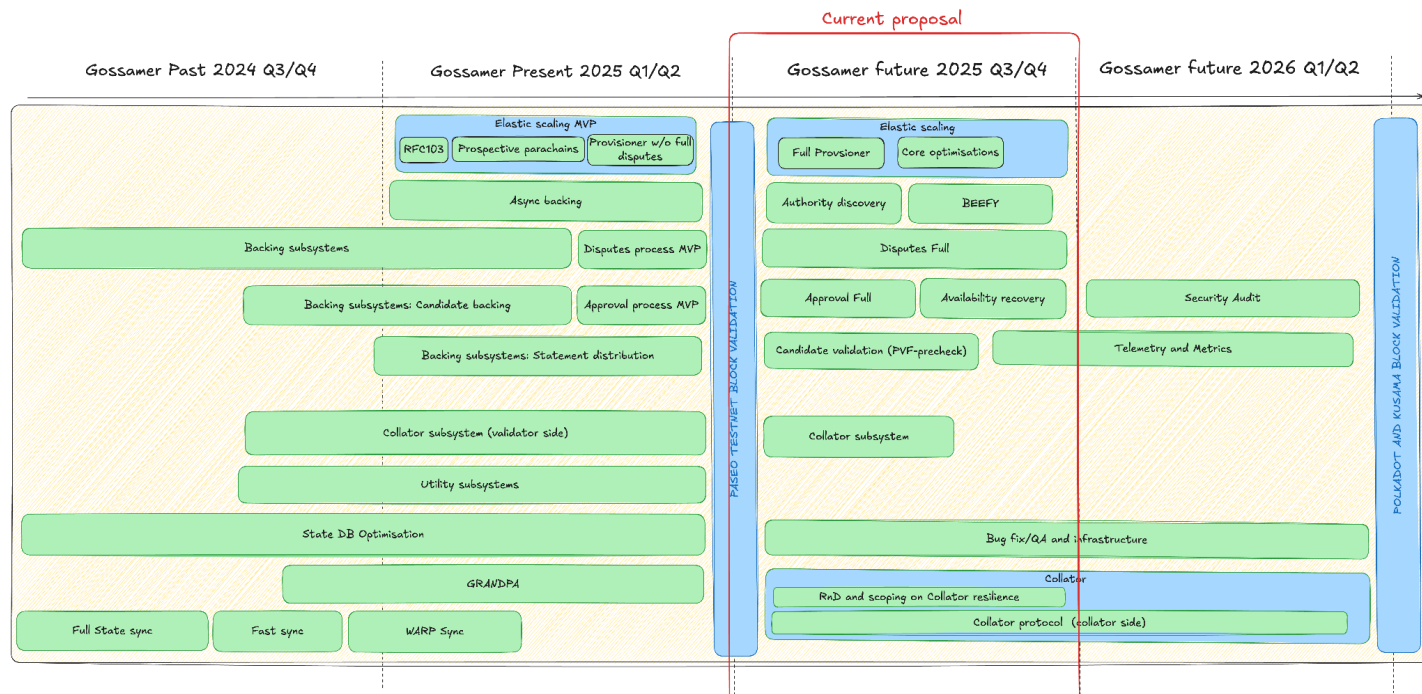
JAM, in essence, is an evolution of the Polkadot network, sharing substantial common DNA with the current Relay Chain. Key modules such as ELVES, GRANDPA, and Safrole are slated for use within JAM without requiring significant modifications. Critically, the expertise derived from these established Polkadot components directly benefits and accelerates the creation of a timely and robust JAM client.

### **Parachains Need Resilient Collators**

Nevertheless, assuming unstoppable evolution of JAM, the Gossamer team is looking for an additional way to contribute to the protocol in the long term. During the next 6 months, we will also dedicate research toward designing more resilient collators for parachains. Despite all the changes that JAM brings, parachains will still require collators to interface with the proposed parachains JAM service. PoV blocks for parachains will still need to be collated and submitted to ensure shared security of the parachain. More details are explained in section [#4.2.2 Collator resilience R&D](#).

## **4. Scope of Work**

Below is the Gossamer roadmap undertaken after the previous treasury proposal and future plan for achieving validating functionality on Polkadot and collating for parachains.



Pic 4.1 Gossamer mid-2024 - mid-2026 Roadmap

## 4.1 Progress Report for the Previous Grant Period

The previous year, the Gossamer team was mainly focused on implementing the ELVES protocol, refactoring our storage abstraction layer, integrating GRANDPA, and starting work on the BEEFY protocol. As described in section 4.1.1, we also encountered several issues and challenges that slowed our progress.

Here is a report of the work done within the previous report:

| Feature                             | Description                                                                                                                                                                                                                                       | Status   |
|-------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------|
| <b>Full, Fast, and WARP Syncing</b> | The fast sync and WARP sync modes are complete, and the Gossamer node successfully syncs using these modes with all Polkadot ecosystem networks. As of now, Gossamer nodes can fully synchronize with all Polkadot networks. <a href="#">Spec</a> | Complete |



|                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                                                                |
|----------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------|
| <b>ELVES: Collator protocol</b>                          | The Collator Protocol implements the network protocol by which collators and validators communicate. It is used by collators to distribute collations to validators, and used by validators to accept collations from collators and communicate backing results. <a href="#">Design</a> , <a href="#">Specification</a> .<br>We have finished the design phase and started implementation with an ETA at the end of Q2 2025. ( <a href="#">Epic</a> ) | In progress                                                    |
| <b>ELVES: Backing subsystem</b>                          | The subsystem is responsible for backing (validating by re-execution, in simple terms) candidates submitted by parachain collators. Logically, this can be broken down into two main components: <b>Candidate backing</b> and <b>Statement distribution</b> .                                                                                                                                                                                         | See the following two sub-items.                               |
| <b>Backing subsystem: Candidate backing</b>              | <a href="#">Design</a> , <a href="#">Specification</a>                                                                                                                                                                                                                                                                                                                                                                                                | Complete ( <a href="#">Epic</a> )                              |
| <b>Backing subsystem: Statement distribution</b>         | <a href="#">Design</a> , <a href="#">Specification</a>                                                                                                                                                                                                                                                                                                                                                                                                | In Progress ( <a href="#">Epic</a> )                           |
| <b>ELVES: Availability subsystem</b>                     | The availability subsystems are responsible for ensuring that Proofs of Validity of backed candidates are widely available within the validator set, without requiring every node to retain a full copy. It consists of availability recovery, distribution recovery, and bitfield signing and distribution.                                                                                                                                          | See the following sub-items                                    |
| <b>Availability subsystem: Availability distribution</b> | <a href="#">Design</a> , <a href="#">Specification</a>                                                                                                                                                                                                                                                                                                                                                                                                | Complete ( <a href="#">Epic</a> )                              |
| <b>Availability subsystem: Availability recovery</b>     | <a href="#">Design</a> , <a href="#">Specification</a>                                                                                                                                                                                                                                                                                                                                                                                                | Implementation planned for Q3 of 2025 ( <a href="#">Epic</a> ) |
| <b>Availability subsystem: Bitfield distribution</b>     | <a href="#">Design</a> , <a href="#">Specification</a>                                                                                                                                                                                                                                                                                                                                                                                                | Complete ( <a href="#">Epic</a> )                              |

|                                                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                                                                    |
|----------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------|
| <b>Availability subsystem: Bitfield signing</b>    | <a href="#">Design</a> , <a href="#">Specification</a>                                                                                                                                                                                                                                                                                                                                                                                                      | Complete<br>( <a href="#">Epic</a> )                               |
| <b>GRANDPA Integration</b>                         | GRANDPA voter module implemented.<br>Integration with networking is in progress<br>Block Import is in progress.<br><a href="#">Design</a>                                                                                                                                                                                                                                                                                                                   | In progress<br>( <a href="#">Epic</a> )                            |
| <b>BEEFY: Merkle Mountain Ranges</b>               | This is an underlying trie abstraction used in BEEFY.                                                                                                                                                                                                                                                                                                                                                                                                       | Complete<br>( <a href="#">Epic</a> )                               |
| <b>State storage trie and DB abstraction layer</b> | We have implemented a lazy loading storage trie and optimized our State DB abstraction layer, generally focusing on making it more substrate-compatible.                                                                                                                                                                                                                                                                                                    | Complete<br>( <a href="#">Epic</a> )                               |
| <b>Authority discovery</b>                         | The discovery mechanism enables Polkadot nodes to both publish their local addresses and learn about other nodes' identifiers and addresses. Currently, the Gossamer node uses the bootstrap mechanism to connect to nodes.                                                                                                                                                                                                                                 | <a href="#">Designed</a> . Not started<br>( <a href="#">Epic</a> ) |
| <b>Async Backing</b>                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Complete<br>( <a href="#">Epic</a> )                               |
| <b>Elastic scaling</b>                             | Elastic Scaling in Polkadot enhances throughput by allowing parachains to use multiple relay chain cores simultaneously. This enables the inclusion of multiple parablocks within the same relay chain block, significantly reducing block inclusion time.<br><a href="#">Design</a> .<br>Although most of the work is done, Elastic Scaling is still being tested and improved by Parity and is expected to be finished <a href="#">by the end of 2025</a> | In Progress<br>( <a href="#">Epic</a> )                            |

#### 4.1.1 Issues or Challenges Encountered

- **Challenge 1: SASSAFRAS**

- SASSAFRAS was updated to SAFROLE and is no longer on the Polkadot roadmap. However, we have already partially implemented SAFROLE for JAM as well.

- **Challenge 2: PVM**

- Our team has dedicated a significant amount of time to implementing the PVM in Go, assuming that it will be part of JAM and play a role in Polkadot 2.0 by late 2025. However, as things stand, it appears that the PVM will be integrated into Polkadot only through the Revive pallet.

- **Challenge 3: BEEFY**

- We have implemented Merkle Mountain Ranges for BEEFY, but general integration with GRANDPA has not yet finished and is planned for Q3-Q4, 2025.

- **Challenge 4: Approval and Disputes**

- Although these are essential components of network security, we've chosen to implement them at a later stage, as they do not currently block our ability to begin testing validation capabilities, specifically block building and finalization. This decision is largely based on the observation that disputes on Polkadot have been practically nonexistent to date. Planned for Q3-Q4 of 2025

- **Challenge 5: Zombienet Integration**

- We have implemented part of the Zombienet integration but decided to pause the initiative and resume after full functionality is in place.

- **Challenge 6: JAM**

- Right after JAM was announced, part of our team has dedicated resources to researching and implementing, as well as facilitating community engagement around JAM. We see this as a long-term investment to drive the future of Polkadot. Our main motivation is to make JAM a reality, we are not just in it for a potential JAM prize.

## 4.2 Scope of Current Proposal

### 4.2.1 Gossamer Validator Feature Set

| Feature                      | Description                                                                                                                                                                                                                                                                         | SoW                  |
|------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------|
| ELVES: Approval Subsystem    | Responsible for revalidating all the transactions included in a block and raising a dispute in case any of them is malicious.<br><a href="#">Epic</a> , <a href="#">Specification</a>                                                                                               | <a href="#">Epic</a> |
| ELVES: Disputes subsystems   | If approval voting finds an invalid candidate, a dispute is raised.<br>It was partially done to suffice as an MVP of the Polkadot Relay Chain validator, but further adjustments need to be made.<br><a href="#">Epic</a> , <a href="#">Specification</a>                           | <a href="#">Epic</a> |
| ELVES Availability recovery: | This subsystem is responsible for recovering the data necessary for candidate validation during the approval/disputes processes.<br><a href="#">Epic</a> , <a href="#">Specification</a> , <a href="#">Design</a>                                                                   | <a href="#">Epic</a> |
| Authority Discovery          | The discovery mechanism enables Polkadot nodes to both publish their local addresses and learn about other nodes' identifiers and addresses. Currently, the Gossamer node uses the bootstrap mechanism to discover nodes.<br><a href="#">Design</a> , <a href="#">Specification</a> | <a href="#">Epic</a> |
| Elastic Scaling              | Though most of the work for elastic scaling was done in the previous proposal ( <a href="#">Epic</a> ). Based on our investigations, it is still being iterated on the Parity side. Hence, we will be closely following all optimizations and upgrades of the protocol.             | <a href="#">Epic</a> |
| BEEFY                        | A light client protocol for Polkadot that foremost facilitates efficient bridging between Polkadot relay chains (such as Polkadot and Kusama) and external blockchains like Ethereum.                                                                                               | <a href="#">Epic</a> |

|                                    |                                                                                                                                          |                      |
|------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------|----------------------|
| Finishing Collator protocol        | After June, we would still have around 30% of collateral protocol to finish.                                                             | <a href="#">Epic</a> |
| Telemetry and Metrics              | Compliance with standard Polkadot Validator metrics as well as integration of Open Telemetry for tracing and better bottlenecks analysis | TBA                  |
| Bugfix, QA and performance testing | E2E testing, including Zombinet tests, performance load tests, and other optimizations.                                                  | TBA                  |
| Project/product management         | Engineering and Product management support for the team and ecosystem                                                                    | TBA                  |

***\*Estimation is a tricky aspect of protocol development because of the complexity and constant protocol upgrades. We aim for precision using various techniques, but still, some of our predictions don't turn out as expected. SoW above is expected to be finished within the next 6 months with our team.***

#### 4.2.2 Collator Resilience R&D

As outlined in our Roadmap (Pic 4.1), over the next six months, our main focus will be progressing toward becoming a validator while also pursuing a vital initiative: *enhancing the resilience of Polkadot Collators*.

Collators are often neglected, but are an important part of the Polkadot infrastructure. Since every parachain is using shared security granted by the RelayChain, security assumptions for collators are pretty low; generally, the most critical parameter is resilience. The leading indicator of resilience in this context is ***Uptime***. When the collator is down, parachain will be missing its blocks. As more logic is migrated to AssetHub, the responsibilities of its collators are increasing. Additionally, with shorter block times, the demands on performance and reliability are growing, making it more critical than ever to have highly dependable collators in place.

The only real path to sustained scalability and reliability is *horizontal scaling*. It removes single points of failure, distributes load, and allows your system to grow with demand.

Blockchain nodes do not scale horizontally by default. Each node must independently process all transactions and have a unique identifier, creating inherent bottlenecks that demand different scaling strategies.

The hardest part of horizontal scaling is *managing state*. The more state your system holds, or needs to synchronize across nodes, the harder it becomes to scale and maintain consistency. Systems that minimize shared state or rely on stateless components scale more easily and yield more predictable, consistent behavior. That is precisely the problem ChainSafe once solved in Lodestar, a consensus beacon node and validator client for the Ethereum blockchain. Refactored state management allowed Lodestar to achieve [100% uptime when running Lodestar nodes for Lido](#) and consistently be among the top 15% of Lido validators.

During the next 6 months, we plan to dedicate some of our capacity to perform research and implementation on collator state separation and minimization. The planned outcome is to create the initial version of a collator node and assess it against being hosted in the managed high-load environments.

This is our general R&D plan to achieve scalable collator logic:

- **Isolate the collator logic**

Decouple it from other system components to make it independently deployable and scalable.

- **Minimize the required state**

Identify and reduce the volume of state each collator instance needs to operate. The fewer states, the easier it is to scale and synchronize.

- **Research and choose a state management strategy**

Two approaches to consider:

- **Decentralized state per instance**

Each collator keeps its own local state, running at maximum speed with

minimal coordination. Occasional non-BFT synchronization (e.g., via gossip or checkpoints) may be used to ensure convergence over time.

- **Centralized ACID-compliant state store**

A single fast, reliable storage layer holds the shared state, with strong consistency guarantees using distributed locks, transactions, etc. Easier to reason about consistency, but introduces potential bottlenecks.

## 5. Budget

### 5.1 Previous Proposal Budget Report

- **Allocated Budget:** 368K DOT
- **Period:** 16 months
- **Spent So Far:** \$1,65M (including 240K DOT ~ \$1.1M with average price of sale ~\$4.6)
- **Remaining Budget:** (negative \$550K) and 128K DOT left (including 110K in Staking)

### 5.2 Current Proposal Budget Description

#### 5.2.1 Gossamer Software Development

The Gossamer core team currently consists of:

- 1x Full-Time Protocol Product Manager/Senior Engineering Manager
- 8x Full-Time Protocol Engineers

This leads to an overall cost breakdown of the total requested funds:

**July 2025–December 2025 (6 months):**

Our average Engineering salary rate is **94.37 USD/hour** including overhead. The real rate is based on the amount of experience and responsibility within the team. Assuming

the nature of Protocol Development, some of the engineers also perform QA and DevOps roles.

As outlined in the Statement of Work (section 4.2.1), providing exact estimates is challenging. However, our team commits to delivering the work completed within a six-month timeframe. This assumes an average number of working hours per month, factoring in public holidays, vacations, and sick leave.

**(8 Engineers + 1 Eng. Manager) \* 6 months (160h \* 6) \* 94.37 h/rate USD = 815\_358 USD**

***ADDED on 18.06:***

We were asking for a total of 815.358 USD for 6 months of development work for a 9-person team.

This means: 135.000 USD / month

This number covers two major blocks:

- On average 110.000 USD/month salary costs for the 9-person team
- about 25.000 USD/month in ancillary costs

The ancillary costs include:

- Travel to visit the conference, hackathons, and to stay in touch with the community
- software subscriptions
- Infrastructure costs for CI/CD pipelines, e2e tests, testnets, etc.
- Additional jurisdictional company costs like local taxes, social, and health coverage (this varies vastly for different countries)

Projecting the monthly salary costs to yearly average salary costs of our protocol engineers, we are still below 150.000 USD/year/protocol engineer according to the following resources:

- <https://web3.career/web3-salaries> (Go Engineer table entry)
- <https://metana.io/blog/web3-developer-salary-2025-what-you-can-earn-in-blockchain/>
- <https://web3.career/protocol-engineer-jobs>

## 5.3 Total Budget of This Proposal

**Beneficiary account:** 149mJjdQjEBMHbWDjbLJ7X4e95ps6L35DZVaBzn1raR1EVQ



Cost (in USDC): 815\_358

## Progress and Updates:

- Monthly progress reports (GitHub [discussions](#), [X](#))
- GitHub code progress (see [insights](#))
- Presentations at Polkadot events
- Documentation updates (see [docs](#))
- New releases every 2–4 weeks once the production state is reached (see [releases](#))
- New (Gossamer) nodes on [Telemetry](#)

## Contact:

- Kyrylo Pysariev, Senior Engineering Manager, [kyrylo@chainsafe.io](mailto:kyrylo@chainsafe.io), @p1sar:[matrix.org](https://matrix.org)
- Peter Kalambet, Head of Protocol, [peter@chainsafe.io](mailto:peter@chainsafe.io), @peter:[dod.ngo](https://dod.ngo)
- Belma Gutlic, VP of Engineering, [belma@chainsafe.io](mailto:belma@chainsafe.io), @morrigan.iv:[matrix.org](https://matrix.org)