Title: Aztecscan – the first privacy focused block explorer

Proposer

: Walnut

Contact details:

roman[at]walnut[dot]dev

Summary:

Walnut team, experienced with building devtools on Starknet and Optimism, is interested in delivering this project. We plan to build an open source explorer that anyone can run locally, and also offer a hosted version. We hope this will mark our first project for the Aztec ecosystem and spark the beginning of a long-term collaboration.

Estimated start and end date:

August 2024 – November 2024 for the first launch

Grant amount requested:

$45k

Grant budge rationale:

- $40k: One senior full stack developer at $10k per month for four months.
- $5k: Occasional support from a designer and frontend engineer.

We intend to cover hosting costs ourselves initially. If the hosting costs grow large due to demand, we would like the option to discuss other grant options or retroactive funding in the future.

Details

gm Aztec!

We at the Walnut team are excited about building Aztecscan, an explorer for Aztec. We hope that it will mark the beginning of our long journey with Aztec and we will have the chance to deliver more tools and infrastructure for Aztec in the future.

Open Source and Privacy Focused

To meet Aztec's high privacy standards, we propose to make the explorer fully open source, available under an open licence such as MIT, allowing anyone to run it locally. We will also run and manage a hosted version at aztecscan.co for those users who prefer convenience. The app will not collect any user data such as IP addresses.

About Walnut and why are we the right team for successfully delivering this project

We are a team of five (and hiring!) developers experienced in Ethereum, the EVM and some rollups around, mostly Starknet and Optimism.

Some examples of our work

- cairovm.codes: A web app for compiling and debugging Cairo. This project required extensive low-level work to map CairoVM instructions to Cairo code. The tool is nearly complete and will soon be integrated into the official Cairo website. The work was supported by Starkware and Starknet Foundation.
- walnut.dev: A debugger and simulator for transactions on Starknet, currently in development. We presented an early version at ETHPrague and plan to launch during Starknet CC in Brussels. Check out some screenshots on our changelog. The development of this tool is also supported by a grant from the Starknet Foundation.
- opscan: An explorer for the OP Stack. Work started this week, and the repository is available here. Supported by a grant from Optimism, you can view our accepted proposal here.

Our extensive experience with transaction decoding, simulations, VM instructions and memory mapping, and transaction debugging makes us an excellent fit for this project.

Design Philosophy

We propose a design similar to Etherscan, incorporating Aztec's color scheme. This will make the explorer familiar and accessible to developers experienced with Etherscan. Our designs are mobile-first, focusing on clarity and ease of use.

Tech stack

- Shadcn + Tailwind UI
- React
- Node JS
- PostreSQL
- Docker
- AWS for the hosted version

Progress Reporting

We are committed to transparency and accountability. We will provide detailed updates on deliverables and any potential roadblocks after completing each milestone, as a comment under this post. The project will be developed publicly on a GitHub repository, allowing anyone to track our progress.

Note on Business Motivation

This is a proposal for an open-source tool, but that doesn't mean there is no business motivation behind it.

Here are some areas we plan to explore for monetisation:

- RPC Service or Paid APIs for large and enterprise customers. Those might include wallets, data analytics companies or large dApps.
- Transaction Simulation API to preview outcomes of transactions before they happen
- Tx debugger and simulator similar to Tenderly

Note on other work for Aztec

In the RFP, it is mentioned that Aztec is also looking for builders to deliver other projects, namely viewing key tool, programmatic access to node data, indexer, RPC provider, bootstrap nodes.

We are very interested to get involved in building those tools. We are very interested in running an RPC service for Aztec and offering programmatic access to the node and explorer data, but also the other projects that are mentioned.

Note on technical support

We would appreciate, if possible, to get access to technical people familiar with Aztec's internals. We do not intend to bother them a lot, but we envision it could help us go through certain potential roadblocks faster.

Milestones

Starting with Milestone 1, we will make the explorer connected to the aztec sandbox. We will extend the explorer with support for testnet and mainnet once those are launched.

Milestone 1: The Basics + Block page (due end of August 2024)

- Homepage with a list of recent blocks, recent transactions, and a search bar to search for blocks, transactions, and contracts.
- Block page with information about each block, reflecting the specification in the RFP, specifically:
- Block hash
- Timestamp
- Number of transactions and their transaction hashes
- Stats on logs
- Block header data
- User-friendly view of transaction state effects

- Block hash
- Timestamp
- Number of transactions and their transaction hashes
- Stats on logs
- Block header data
- User-friendly view of transaction state effects
- Simplistic transaction page with transaction hash and block number
- Simplistic contract page with contract class ID
- Simplistic class page with class hash ID

Note: information about gas will be added in a later milestone.

Milestone 2: Transaction page & fee data (due end of September 2024)

- Expand the transaction page with more details, as specified in the RFP, specifically:
- Transaction hash (identifier)
- Status (mining/preconf/proving/success/dropped)
- Block number
- L1 batch transaction
- Timestamp
- Transaction fee (in the native fee-paying asset)
- Transaction effects:
- Number of notes + note hashes and note encrypted logs
- Number of nullifiers + hashes
- L1->L2 message hash
- L2->L1 message hash
- Encrypted logs
- Unencrypted logs
- Public state reads/writes
- Public function call(s) (similar to how one can view function calls on Etherscan)
- Number of notes + note hashes and note encrypted logs
- Number of nullifiers + hashes
- L1->L2 message hash
- L2->L1 message hash
- Encrypted logs
- Unencrypted logs
- Public state reads/writes
- Public function call(s) (similar to how one can view function calls on Etherscan)
- Fee-related information:
- FPA used

- Gas price
- Gas
- Fee divided per dimensions (DA, L1, L2)
- Coinbase (recipient of block reward)
- Fee recipient (Address to receive fees)
- Any public teardown call
- Any public fee information (depends on if the user/dapp pays the fee in public or not):
- Fee payer/fee payment contract or if paid from L1
- Tokens paid (if paid in a custom token)
- Initial fee bid in token or FPA
- Public refund received
- Fee payer/fee payment contract or if paid from L1
- Tokens paid (if paid in a custom token)
- Initial fee bid in token or FPA
- Public refund received
- FPA used
- Gas price
- Gas
- Fee divided per dimensions (DA, L1, L2)
- Coinbase (recipient of block reward)
- Fee recipient (Address to receive fees)
- Any public teardown call
- Any public fee information (depends on if the user/dapp pays the fee in public or not):
- Fee payer/fee payment contract or if paid from L1
- Tokens paid (if paid in a custom token)
- Initial fee bid in token or FPA
- Public refund received
- Fee payer/fee payment contract or if paid from L1
- Tokens paid (if paid in a custom token)
- Initial fee bid in token or FPA
- Public refund received
- Transaction hash (identifier)
- Status (mining/preconf/proving/success/dropped)
- Block number
- L1 batch transaction
- Timestamp
- Transaction fee (in the native fee-paying asset)

- Transaction effects:
- Number of notes + note hashes and note encrypted logs
- Number of nullifiers + hashes
- L1->L2 message hash
- L2->L1 message hash
- Encrypted logs
- Unencrypted logs
- Public state reads/writes
- Public function call(s) (similar to how one can view function calls on Etherscan)
- Number of notes + note hashes and note encrypted logs
- Number of nullifiers + hashes
- L1->L2 message hash
- L2->L1 message hash
- Encrypted logs
- Unencrypted logs
- Public state reads/writes
- Public function call(s) (similar to how one can view function calls on Etherscan)
- Fee-related information:
- FPA used
- Gas price
- Gas
- Fee divided per dimensions (DA, L1, L2)
- Coinbase (recipient of block reward)
- Fee recipient (Address to receive fees)
- Any public teardown call
- Any public fee information (depends on if the user/dapp pays the fee in public or not):
- Fee payer/fee payment contract or if paid from L1
- Tokens paid (if paid in a custom token)
- Initial fee bid in token or FPA
- Public refund received
- Fee payer/fee payment contract or if paid from L1
- Tokens paid (if paid in a custom token)
- Initial fee bid in token or FPA
- Public refund received
- FPA used
- Gas price
- Gas

- Fee divided per dimensions (DA, L1, L2)
- Coinbase (recipient of block reward)
- Fee recipient (Address to receive fees)
- Any public teardown call
- Any public fee information (depends on if the user/dapp pays the fee in public or not):
- Fee payer/fee payment contract or if paid from L1
- Tokens paid (if paid in a custom token)
- Initial fee bid in token or FPA
- Public refund received
- Fee payer/fee payment contract or if paid from L1
- Tokens paid (if paid in a custom token)
- Initial fee bid in token or FPA
- Public refund received

Milestone 3: Contract and Class pages (due end of October 2024)

- Contract public code (or bytecode)
- Contract private code (or bytecode)
- Contract class ID
- Version
- Hash of the initialization function that ran
- Deployer address
- Salt
- Note tagging scheme registered
- Public keys associated with the address (e.g., nullifier, viewing, etc.)
- Hash of public keys deployed with the contract
- Any public transactions where this address was either a from

or a to

Milestone 4: Launch of aztecscan.co (due end of November 2024)

- Polish and finish everything that has not yet been done.
- Deploy the project to aztecscan.co and set up a CI pipeline on GitHub for automatic deployments from the main branch.
- Set up a public uptime monitoring service like Uptime.com or Pingdom to report the liveness of the web app. Monitor and fix potential bottlenecks to provide reliable service.
- Publish a post on the Aztec governance forum announcing the new explorer. Encourage everyone to use it and offer feedback. Invite developers to explore the project's repository on GitHub and contribute.

Milestone 5: post launch learnings and next steps (due end of January 2025)

- Publish a post on the Aztec governance forum with learnings from the launch:
- Uptime report: Status on the liveness of the web app
- GitHub report: Number of stars and contributions from external contributors

- Summary of all feedback collected since the launch in November
- Uptime report: Status on the liveness of the web app
- GitHub report: Number of stars and contributions from external contributors
- Summary of all feedback collected since the launch in November
- Suggest next steps:
- Offer our thoughts on what feedback should be incorporated
- Suggest new features to enhance the project, such as support for viewing keys to decrypt any notes or logs with a viewing key, and support for smart contract verification.

Other Not Time-Bound Milestones

- Once the Aztec Testnet is live: Connect the deployed version of the explorer, hosted at aztecscan.co, to the Aztec Testnet.
- Once Aztec Mainnet is live: Connect the deployed version of the explorer, hosted at aztecscan.co, to the Aztec Mainnet. Maintain a switch to testnet so that users can explore both.

Info on how to verify the delivery of each milestone:

- As soon as with Milestone 1, our GitHub repository will contain a README.md with instructions for running the explorer locally.
- Anyone will be able to run it, verify the milestone was completed, and provide feedback.

Link to the repository will be posted as a comment under this post when our work starts, so early August 2024.

Closing thoughts

We are very excited to begin this work and enter the Aztec ecosystem. We look forward to building a long-lasting collaboration and developing numerous tools around RPCs, transactions, simulations, and more.

We want to extend our gratitude for the detailed RFP, which has significantly facilitated our grant submission process. A special thanks to Lauri for bringing this request to our attention.