

Hello there SN community!

Since its been 20+ days since we've posted our [Github grant proposal](#) and we've seen no activity on it we decided to post it here. We hope we reach better visibility and gather more feedback.

Fadroma - Secret Network smart contracts development tooling

Project Description

The purpose of the Fadroma suite of tools is to facilitate the rapid prototyping, continued development, and maintenance of decentralized services and applications. It consists of a Rust library to serve as a foundation for building smart contracts, and a TypeScript-based build system for deploying smart contracts and interacting with them from JavaScript. The result is a portable and accessible development environment for smart contracts that abstracts away a large part of the grunt-work and lets you focus on your product logic.

Problem / Solution

As a smart contract developer

, I need to focus on the business logic of what I'm writing. For that, I need to iterate on a tight feedback loop. Therefore, I start my project with Fadroma because it lets me automate the repetitive workflow steps and provides me with reusable building blocks to streamline contract development. This empowers me to maintain high development velocity and deliver a reliable product.

- Fadroma has integrated the LocalSecret local network;
- Fadroma allows me to test contract and cross-contract interactions natively (Rust);
- Fadroma allows me to write deploy scripts;
- Fadroma keeps track of deployments and allows deployments to continue from where they stopped if interrupted;
- Fadroma provides me with receipts for all deployments.

As a full-stack Web 2.0 developer being onboarded to blockchain development

, I expect a streamlined, comprehensive development environment that builds on top of the similarities between the Web 2.0 and Web 3.0 operational model, provides clear affordances towards approaching the areas where the two differ (e.g. viewing smart contracts as a serverless deployment platform that is metered per operation rather than per CPU hour). I want to easily write abstractions to communicate with smart contracts from JavaScript.

- Fadroma provides me with building blocks for writing client interfaces for contracts.

Detailed product description

Over the past year and a half, we've been developing Fadroma as our in-house repository of Secret Network-related capabilities. It has been instrumental to realising the Sienna Network project. From the start, we decided to keep a common library to facilitate the rapid prototyping, continued development, and maintenance of decentralized services and applications based on CosmWasm.

Fadroma currently consists of:

- a Rust library to serve as a foundation for smart contracts (e.g. composable contracts, derive macro)
- a TypeScript build system for deploying smart contracts and interacting with them from JavaScript, which is nearly ubiquitous

Fadroma's architecture went through several changes since its inception. We are currently approaching a 1.0.0 stable release with support for secret-cosmwasm-std 0.10, secretjs 1.4.0 and a legacy mode based on secretjs 0.17.5. It is capable of building smart contracts in a Docker container, and deploying them to either a containerised local devnet, the pulsar-2 testnet, or the secret-4 mainnet using a scripting API that allows us to declaratively express the complex deployment and configuration procedures of an evolving DeFi system using concise TypeScript. There is also a Mocknet mode which runs smart contract binaries in the standard WASM runtime of Node.js, and allows their behaviours to be validated without having to spin up.

Fadroma code is covered by Rust test suites, and lcof reports. For testing smart contracts, we provide the Ensemble feature (for integration testing from pure Rust) and the Mocknet (which runs smart contract binaries in a simulated environment based on Node.js's built-in WASM runtime, enabling fast-forwarded exploration of dynamic multi-user, multi-contract system behaviour). Fadroma's main dependencies are Node.js >=16 and Docker; containerised builds make it possible to build a Fadroma project without having a Rust toolchain at all.

It is feasible to add rootless builds (with Podman), containerless reproducible builds (with Riff) and/or remote builds (using a custom, build service).

[

fadroma-architecture

563×623 30.3 KB

](https://global.discourse-cdn.com/standard17/uploads/enigma1/original/2X/6/649db300b5bb2cb806c583bfb026d607e079022c.png)

Go-to-Market plan

Fadroma is already open-sourced and even used in production, powering major apps on Secret Network.

As an open-source project, Fadroma is intended to become a common foundation for the community to build on, and a standard component of Cosmos developers' workflows in the way Truffle and Hardhat are for EVM ecosystems.

Regarding announcements, marketing and community:

The product is already live and it is in use by major apps on Secret Network.

- We intend to build and support a community for developers using Fadroma
- Announcements, blog posts, tech guides and examples are on our radar for easier user onboarding

Value capture for Secret Network ecosystem

Fadroma improves day-to-day operations of developers on Secret Network.

To continually improve the viability of the blockchain paradigm as a foundation for the decentralized software ecosystem of tomorrow, the tooling used for developing smart contract-based products needs to be qualitatively better than those currently being used for mainstream Web development.

We'd like for Fadroma to provide a streamlined workflow for rapid development of Secret Network-based products, guiding developers' attention towards extensive high-level validation of smart contracts' behaviour, and preventing them from getting bogged down in platform details.

Eventually, we'd like for Fadroma to become the frontend for an extensive library of pre-validated templates for smart contracts that can be customised and deployed with little to

no coding, similar to what OpenZeppelin provides for Ethereum, or CosmWasm Plus provides for Cosmos.

Additionally, Fadroma IBC support will make it easier to use Secret Network as a privacy provider for unencrypted Cosmos blockchains.

In short:

The more users Fadroma has → the more developers work on Secret Network → the more adoption of Secret Network.

Statistics

The project is used by two of the major projects on Secret Network:

- Sienna
- Shade (based on feature requests, PRs, and feedback received)

We don't have any telemetry built-in the project, but based on GitHub statistics for project clone we have ~1000 unique clones for this month. These are mainly Secret Network users (or soon to be users) as the project only supports SN.

[
68747470733a2f2f6e6f7465732e6861636b2e62672f75706c6f6164732f75706c6f61645f39393364646261376166323037383437363565393031373134303836323439332e6a7067
1822x826 118 KB
](https://global.discourse-cdn.com/standard17/uploads/enigma1/original/2X/a/ac77c9f9bda149b539e60b6505e9b535dcb3dc9a.jpeg)

Demo

In this demo we demonstrate some of the features of Fadroma.

[
](https://asciinema.org/a/L9RePY2AWJHmzZbWUW7FNldg8)

Team members

- Adam Avramov - architect [egasimus · GitHub](#)
- Asparuh Kamenov - core member - [aakamenov · GitHub](#)
- Denis Maximov - core member - [denismaxim0v \(Denis Maximov\) · GitHub](#)
- Milen Radkov - core member - [mradkov · GitHub](#)

Team Website


- <https://fadroma.tech>
- <https://hack.bg>

Team's experience

We have extensive experience developing smart contracts, tooling and infrastructure for blockchain networks and blockchain based projects for the past 6yrs.

Some of our previous work can be seen on our website and GitHub org.

Team Code Repos

- [GitHub - hackbg/fadroma: Industrial strength components and workflows for smart contract development in Rust.](#) 
- [GitHub - hackbg/secret-multicall: Secret Network Multicall Query Batcher](#)
- [GitHub - hackbg/chainlink-terra-cosmwasm-contracts: Chainlink Protocol implementation in CosmWasm \(Rust\) for Terra blockchain](#)
- [GitHub - hackbg/terra-chainlink-demo: Chainlink <> Terra Integration Demo.](#)
- [GitHub - hackbg/terra-fm-metrics-exporter: A Prometheus exporter for terra contracts](#)
- [GitHub - SiennaNetwork/SiennaNetwork: Sienna Network Monorepo](#)
- This is part of the relevant ones. Whoever is interested can easily find our other contributions.

Team LinkedIn Profiles

- <https://bg.linkedin.com/company/hack-blockchain-development>
- not all of us are on LinkedIn.
- not all of us are on LinkedIn.

Development Roadmap

Delivery

- Publish JS components to NPM
- @hackbg/fadroma

yet to be published

- @hackbg/fadroma

yet to be published

- Publish Rust components to [Crates.io](#)
- Install the whole toolkit with nix-shell <https://advanced.fadroma.tech>

Contract development and testing

- Ensemble - write inter-contract-communication tests in Rust, by simulating blockchain behaviour and state without spinning up an actual node
- Mocknet - which runs smart contract binaries in a simulated environment based on Node.js's built-in WASM runtime, enabling fast-forwarded exploration of dynamic multi-user, multi-contract system behaviour
- View test coverage reports
- Toolkit for common types and common functionality
- Composable trait that allows you to compose self-contained subsystems into a single contract

Macros

- Derive contract macro
- Generate CosmWasm boilerplate
- Generate interfaces to write contracts as composable components. Some of the built-in components:
 - Killswitch
 - Admin
 - Auth (viewing key, permits)
- Killswitch
- Admin
- Auth (viewing key, permits)
- Generate CosmWasm boilerplate
- Generate interfaces to write contracts as composable components. Some of the built-in components:
 - Killswitch
 - Admin
 - Auth (viewing key, permits)
- Killswitch
- Admin
- Auth (viewing key, permits)
- Derive canonize macro - easily convert between structures containing addresses (Humanized/Canonized form)

Contract operation

- Integration with secretcli
- Rust library to programmatically interact with secretcli
- Support system keystore in the same way as secretcli

to import mainnet/testnet private keys

Distribution

- Versioning scheme:

1.0.0 upcoming

- Documentation

at <https://fadroma.tech>: guides need another refresh

- Decide contribution policy and governance mechanisms

that allows the community to have their needs met while preserving hack.bg's prerogative on the design and direction of the project

Milestones:

(

what's been done so far)

- 0.1. Declarative macro trying to make CosmWasm look like Redux (dropped)
- 0.2. First project built and deployed with Fadroma on CW0.10.
- 0.3. Localnet container management.
- 0.4. Procedural macro for flexible generation of composable smart contracts and components.
- 0.5. Receipts system - record of uploaded and instantiated contracts
- 0.6. Ensemble - Rust multi-contract integration testing environment.
- 0.7. Mocknet - JavaScript/WASM integration testing environment. Test production builds, fast.
- 0.8. Pluggable builder interface supporting containerised and raw builds, incl. from past commits
- 0.9. Pluggable uploader interface, currently supporting uploads from local files.
- 0.10. Comprehensive domain model of smart contract operations workflow, providing a foundation for idiomatic client libraries and ops scripts.

(

we're here)

- 1.0. CW1.0 support
- Stabilize current code:
- Publish stable version of build system (Fadroma Ops 1.0)
- Publish current state of contract framework (Fadroma Engine 0.10 with legacy secret-cosmwasm 0.10)
- Publish stable version of build system (Fadroma Ops 1.0)
- Publish current state of contract framework (Fadroma Engine 0.10 with legacy secret-cosmwasm 0.10)
- Implement CosmWasm 1.0 support:
- Publish beta version of contract framework (Fadroma Engine 1.0 with secret-cosmwasm 1.0 and cosmwasm 1.0)
- Update existing library features to support the newest CW1.0
- Support CW1.0 IBC features
- Update existing library features to support the newest CW1.0
- Support CW1.0 IBC features
- Publish beta version of build system with CW1.0 support
- Support latest secretjs and cosmwasm-js
- Support multiple devnet containers for local testing of IBC transactions
- Support CW1.0 Mocknet
- Support latest secretjs and cosmwasm-js
- Support multiple devnet containers for local testing of IBC transactions
- Support CW1.0 Mocknet
- Publish beta version of contract framework (Fadroma Engine 1.0 with secret-cosmwasm 1.0 and cosmwasm 1.0)
- Update existing library features to support the newest CW1.0
- Support CW1.0 IBC features
- Update existing library features to support the newest CW1.0
- Support CW1.0 IBC features
- Publish beta version of build system with CW1.0 support
- Support latest secretjs and cosmwasm-js
- Support multiple devnet containers for local testing of IBC transactions
- Support CW1.0 Mocknet
- Support latest secretjs and cosmwasm-js
- Support multiple devnet containers for local testing of IBC transactions
- Support CW1.0 Mocknet
- Stabilize current code:
- Publish stable version of build system (Fadroma Ops 1.0)
- Publish current state of contract framework (Fadroma Engine 0.10 with legacy secret-cosmwasm 0.10)
- Publish stable version of build system (Fadroma Ops 1.0)
- Publish current state of contract framework (Fadroma Engine 0.10 with legacy secret-cosmwasm 0.10)
- Implement CosmWasm 1.0 support:
- Publish beta version of contract framework (Fadroma Engine 1.0 with secret-cosmwasm 1.0 and cosmwasm 1.0)
- Update existing library features to support the newest CW1.0
- Support CW1.0 IBC features
- Update existing library features to support the newest CW1.0
- Support CW1.0 IBC features
- Publish beta version of build system with CW1.0 support
- Support latest secretjs and cosmwasm-js
- Support multiple devnet containers for local testing of IBC transactions
- Support CW1.0 Mocknet
- Support latest secretjs and cosmwasm-js
- Support multiple devnet containers for local testing of IBC transactions
- Support CW1.0 Mocknet
- Publish beta version of contract framework (Fadroma Engine 1.0 with secret-cosmwasm 1.0 and cosmwasm 1.0)
- Update existing library features to support the newest CW1.0
- Support CW1.0 IBC features

- Update existing library features to support the newest CW1.0
- Support CW1.0 IBC features
- Publish beta version of build system with CW1.0 support
- Support latest secretjs and cosmwasmd-js
- Support multiple devnet containers for local testing of IBC transactions
- Support CW1.0 Mocknet
- Support latest secretjs and cosmwasmd-js
- Support multiple devnet containers for local testing of IBC transactions
- Support CW1.0 Mocknet

Rationale: The Secret Network upgrade to CW1.0 will be a major milestone reached. Having support of CW1.0 in Fadroma will make development more efficient while preserving the already useful toolkit provided so far and make CW1.0 contracts from other chains easily transferrable to Secret Network to be tested and deployed. Cross-chain IBC testing will be easier.

(

what we need support for)

- 1.1. Milestone: Stability (week 00-06)
- Improve test coverage and documentation of existing features
- Improve smart contract boilerplate macros (Fadroma Derive)
- Fix known issues
- There is an edge-case where an interface can go out of sync with the current implementation
- There is an edge-case where an interface can go out of sync with the current implementation
- Optimize the API
- Fix known issues
- There is an edge-case where an interface can go out of sync with the current implementation
- There is an edge-case where an interface can go out of sync with the current implementation
- Optimize the API
- Publish stable version
- Improve test coverage and documentation of existing features
- Improve smart contract boilerplate macros (Fadroma Derive)
- Fix known issues
- There is an edge-case where an interface can go out of sync with the current implementation
- There is an edge-case where an interface can go out of sync with the current implementation
- Optimize the API
- Fix known issues
- There is an edge-case where an interface can go out of sync with the current implementation
- There is an edge-case where an interface can go out of sync with the current implementation
- Optimize the API
- Publish stable version

Rationale: After reaching the milestone for Fadroma - CW1.0 support, more time has to be put on stability, performance, fixing known issues and polishing the experience for developers using it. We aim to make the project even more lightweight and well structured. Better up-to-date documentation and more examples are in the scope of this milestone.

- 1.2. Milestone: Additional features (week 06-12)
- Project scaffolding (Fadroma Create)
- Contracts identify themselves by default (standard query API for retrieving version, source URL, API schema)
- Improve performance of reproducible builds (bypass Docker/Mac IO bottleneck)
- JavaScript REPL for interacting with deployed contracts
- Backtraces/source maps/coverage when testing from Mocknet
- Project scaffolding (Fadroma Create)
- Contracts identify themselves by default (standard query API for retrieving version, source URL, API schema)
- Improve performance of reproducible builds (bypass Docker/Mac IO bottleneck)
- JavaScript REPL for interacting with deployed contracts
- Backtraces/source maps/coverage when testing from Mocknet

Rationale: Project setup has to be quick for seamless development experience. We want to work on improving this by adding project scaffolding for Fadroma. Additionally we've spotted some tricky workflows while working on Secret contracts - such as the need of keeping track of old builds and deployments so we want to improve on that.

The REPL is a handy tool we believe will improve dev UX - whether it is a quick contract config that has to be made or an experiment.

- 1.3. Milestone: Additional features 2.0 (week 12-18)

- Create devnet from mainnet snapshot
- Allow selection of mainnet block number to get blockchain state from to run your contracts against;
- Configure custom nodes to interface for this feature.
- Allow selection of mainnet block number to get blockchain state from to run your contracts against;
- Configure custom nodes to interface for this feature.
- Upload from remote URL - build contracts in CI, deploy manually
- Reliable gas profiling.
- Create devnet from mainnet snapshot
- Allow selection of mainnet block number to get blockchain state from to run your contracts against;
- Configure custom nodes to interface for this feature.
- Allow selection of mainnet block number to get blockchain state from to run your contracts against;
- Configure custom nodes to interface for this feature.
- Upload from remote URL - build contracts in CI, deploy manually
- Reliable gas profiling.

Rationale: In this milestone we want to focus on handy features we've used in other tools while developing on other blockchain ecosystems. We believe these are the logical next step in developing Fadroma.

Grant request

The total grant request is \$89,597 (or the equivalent of SCRT at daily exchange rate on payment or USDT).

\$13,337 awarded at the time of grant approval and payments on each milestone completion after.

We propose this grant timeline and funding breakdown:

Milestone

Estimated Date

Amount

1. Grant Accepted

Present (T0)

\$13,337

1. M1.1 Completed

T0 + 6 weeks

\$25,420

1. M1.2 Completed

T0 + 12 weeks

\$25,420

1. M1.3 Completed

T0 + 18 weeks

\$25,420

Total

\$89,597

Additional Information

- Tech:
- <https://fadroma.tech>
- Documentation:
- [@hackbg/fadroma](#)
- [fadroma - Rust](#)
- [@hackbg/fadroma](#)
- [fadroma - Rust](#)
- Coverage:
- [Code coverage report for All files](#)
- [Code coverage report for All files](#)
- Documentation:
- [@hackbg/fadroma](#)
- [fadroma - Rust](#)
- [@hackbg/fadroma](#)
- [fadroma - Rust](#)

- Coverage:
- [Code coverage report for All files](#)
- [Code coverage report for All files](#)
- <https://fadroma.tech>
- Documentation:
- [@hackbg/fadroma](#)
- [fadroma - Rust](#)
- [@hackbg/fadroma](#)
- [fadroma - Rust](#)
- Coverage:
- [Code coverage report for All files](#)
- [Code coverage report for All files](#)
- Documentation:
- [@hackbg/fadroma](#)
- [fadroma - Rust](#)
- [@hackbg/fadroma](#)
- [fadroma - Rust](#)
- Coverage:
- [Code coverage report for All files](#)
- [Code coverage report for All files](#)
- Social:
- [Introducing Fadroma](#)
- [Secret Feature: Fadroma](#)
- [Introducing Fadroma](#)
- [Secret Feature: Fadroma](#)