Implement a CLI for node interactions in Hyperledger Solang

LFX Mentorship 2023

Project plan by: Tarek Elsayed

Mentor: Cyrill Leutwiler

Description

The project aims to add new commands to the Command Line Interface (CLI) for Hyperledger Solang, a Solidity compiler written in Rust. The CLI will enable node interactions with Substrate and Solana blockchains, including uploading and deploying contracts. The CLI will provide a more efficient and streamlined process for uploading, deploying, and interacting with contracts during iterative contract development.

Deliverables

Solang CLI for Substrate:

- `solang upload --target substrate <contract_file>`: Uploads the compiled contract artifact to a Substrate blockchain node.
    - Flags:
        - `--suri`: The Secret URI used for signing the extrinsic (mandatory).
        - `--password`: Optional password for the `--suri`.
        - `--url`: The websockets URL of an RPC node on the target chain. Defaults to a locally running node at "ws://localhost:9944".
        - `-x` or `--execute`: Optional flag to execute the extrinsic on-chain.
        - `--storage-deposit-limit`: Optional limit on the maximum balance that can be charged from the caller to pay for the storage consumed.
- `solang instantiate --target substrate <contract_file>`: Deploys and instantiates the contract on a Substrate blockchain node.
    - Flags: Same as the `upload` command.
- `solang call --target substrate <contract_address> <function_name> [arguments...]`: Calls a function on a deployed contract in a Substrate blockchain node.

    - Flags: Same as the `upload` command.

Solang CLI for Solana:

- `solang upload --target solana <contract_file>`: Deploys and instantiates the contract on a Solana blockchain.
    - Flags: To be determined during the research phase.
- `solang call --target solana <contract_file>`: Calls a function on a deployed contract in a Solana blockchain node.
    - Flags: To be determined during the research phase.
    - This should ideally be implemented in rust. If not, a 3rd party program can be called as a backup plan.

Project Plan

To start, the project will focus on Substrate, as there is an existing tool called `cargo-contract` that can be leveraged. The plan is to extract the relevant parts of `cargo-contract` into a separate crate called `contract-extrinsics`. This crate will include functionalities for uploading, instantiating, and calling contracts on a Substrate blockchain node.

The first step is to extract the necessary code from `cargo-contract` into a new crate called `contract-extrinsics`. This will involve identifying and separating the components responsible for contract upload, instantiation, and function calling. By creating a dedicated library (crate), these functionalities can be made accessible within Solang CLI.

Once `contract-extrinsics` is available, the Solang CLI for Substrate can be implemented.

Throughout the implementation process, unit tests should be written to ensure the correctness of the CLI functionalities. The Solang CLI for Substrate should be thoroughly tested with different contract scenarios to validate its reliability and effectiveness.

As part of the implementation process, it is essential to include documentation for the newly added commands in the Solang CLI. The documentation should give clear instructions on how to use the commands, including the available flags and usage examples.

While Substrate interactions are being developed and tested, the project will also include research and exploration of Solana's node interaction possibilities. As of now, the details and available tools for Solana CLI interactions are vague. The plan is to dedicate the latter half of the project period to researching and understanding Solana's deployment and interaction mechanisms. Once sufficient knowledge is acquired, the Solang CLI for Solana can be implemented accordingly.

It is important to note that Solana has a different structure and set of commands compared to Substrate. Unlike Substrate, Solana does not have a separate command specifically for contract instantiation. Instead, the `solana deploy` command in Solana CLI is responsible for both deploying and instantiating the contract on the blockchain. Therefore, when implementing the Solang CLI for Solana, it will be necessary to accommodate this difference in command structure. The focus will be on implementing the `solang upload --target solana <contract_file>` command, which will handle the contract deployment and instantiation process on the Solana blockchain. The specific flags and options for this command will be determined during the research phase to ensure alignment with Solana's deployment mechanism.

Timeline

- June 12- July 14th (Exploration Period):
    - Spend the first two weeks diving into the Solang relevant codebase, including the existing Solang CLI and `cargo-contract`, to gain a deep understanding of the architecture and functionality.
    - Extract Relevant Parts: Extract the necessary components from `cargo-contract` into the new crate, `contract-extrinsics`, allowing for their use in the Solang CLI.
    - Research and Exploration: Dedicate time to research Solana's deployment and interaction mechanisms.

- July 15th - August 31st (Substrate Development Phase):
    - Add Substrate Commands: Implement the commands in the Solang CLI, utilizing the functionalities provided by `contract-extrinsics`.
    - Designing an generic interface for node interactions, abstracting away blockchain specifics.
    - Documentation: Add documentation for the new Substrate commands, ensuring clear instructions, usage examples, and explanations of available flags.

- September 1st - November 30th (Solana Development and Research Phase):
    - Solana Upload Command: Implement the `upload` command in the Solang CLI to handle contract deployment and instantiation on the Solana blockchain.
    - Solana Call Command: Implement the `call` command in the Solang CLI to handle calling functions on the deployed contracts on the Solana blockchain.

- Documentation: Update the documentation to include the newly added Solana command, providing users with instructions on deploying contracts to the Solana blockchain.