Welcome to Solang, a new Solidity compiler written in rust which uses llvm as the compiler backend. Solang can compile Solidity for Solana and Polkadot Parachains with the contracts pallet. Solang is source compatible with Solidity 0.8, with some caveats due to differences in the underlying blockchain.

Solang is under active development right now, and has extensive documentation.

Solana

Please follow the Solang Getting Started Guide.

Solang is part of the Solana Tools Suite (version v1.16.3 and higher). There is no need to install it separately.

Installation

Solang is available as a Brew cask for MacOS, with the following command:

```
brew install hyperledger/solang/solang
```
For other operating systems, please check the installation guide.

**Build for Polkadot**

Run the following command, selecting the flipper example available on Solang's repository:

```
solang compile --target polkadot examples/polkadot/flipper.sol
```

Alternatively if you want to use the solang container, run:

```
docker run --rm -it -v $(pwd):/sources ghcr.io/hyperledger/solang compile -v -o /sources --target polkadot /sources/flipper.sol
```

You will have a file called flipper.contract. You can use this directly in the Contracts UI, as if your smart contract was written using ink!.

**Tentative roadmap**

Solang has a high level of compatibility with many blockchains. We are trying to ensure the compiler stays up to date with the newest Solidity syntax and features. In addition, we focus on bringing new performance optimizations and improve developer experience. Here is a brief description of what we envision for the next versions.

<table>
<thead>
<tr>
<th>V0.4 Feature</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Improve management over optimization passes</td>
<td>Not started</td>
</tr>
<tr>
<td>Adopt single static assignment for code generation</td>
<td>In progress</td>
</tr>
<tr>
<td>Support openzeppelin on Polkadot target</td>
<td>In progress</td>
</tr>
<tr>
<td>Provide Solidity -&gt; Polkadot porting guide</td>
<td>Not started</td>
</tr>
<tr>
<td>Declare accounts for a Solidity function on Solana</td>
<td>In progress</td>
</tr>
<tr>
<td>Tooling for calls between ink! &lt;&gt; solidity</td>
<td>In progress</td>
</tr>
<tr>
<td>Provide CLI for node interactions</td>
<td>Done</td>
</tr>
</tbody>
</table>

**License**

Apache 2.0

The Solidity language is the most popular programming language for smart contracts. However, the existing Solidity compiler only targets the Ethereum virtual machine. The Solang project aims to make Solidity available for other blockchains, and focuses on maintaining source code compatibility with Solc, so that developers can use their existing codebase for blockchains other than Ethereum with minimal modifications.

The purpose of Solang is two-fold: first of all, developers with Solidity language knowledge can develop new smart contracts for non-Ethereum blockchains in Solidity, so they do not have to learn a new language. Secondly, there is a large amount of existing Solidity contracts available, which can be recompiled for a different blockchain.

Currently, Solang targets the following blockchains:

- Solana
- Substrate

At the time of writing, these chains are the 9th, and 11th by market capitalization according [coinmarketcap.com](http://coinmarketcap.com) (the coins are listed by market cap).

Any other blockchain that wishes to have Solidity language support is welcome to add a new target to the Solang project. The cosmos blockchain [grant foundation](https://cosmofoundation.io) has said it would support a grant for adding CosmWasm support.

The goal of Solang is to bring the Solidity language to as many blockchains as possible. Writing a production quality compiler is a complex task, so collaboration between blockchains will be hugely beneficial.

The success of the project can be measured by the number of projects that use Solang as a compiler.
Key Characteristics

Hyperledger Solang is a compiler: it knows how to transform Solidity source code into a binary program (or contract) which can be directly deployed on a blockchain. Hyperledger Solang does not provide tooling for this, however we do provide documentation on how to deploy and interact with your Solidity contracts, see the block-chain specific documentation below.

Hyperledger Solang has the following stages:

1. Parser stage. This parser Solidity. This is in the `solang-parser` crate. This crate is used by another project: foundry uses it for its Solidity code formatter.
2. Semantic Analysis Stage (sema). This validates that the source code is valid, and produces the AST. This is used by the Language Server.
3. Code generation (codegen). This transforms the AST into a CFG (control flow graph). This deals with contract inheritance too, and includes a few code optimization passes.
4. LLVM IR emit (emit). This transforms the CFG into LLVM IR.
5. LLVM Backend. The LLVM Libraries further optimize and compile the code into optimized binary files.
6. LLD Linker. The LLD Linker produces the final file.

Documentation


How to run Solang on command line: https://Solang.readthedocs.io/en/latest/running.html

Blockchain-specific instructions for:

- Solana: https://solang.readthedocs.io/en/latest/targets/solana.html
- Substrate: https://solang.readthedocs.io/en/latest/targets/substrate.html

Project Management and Issue Tracking

All Solang projects use GitHub for receiving issues, receiving pull requests and tracking releases. The links to the GitHub repos for the project are below.

Repositories

https://github.com/hyperledger/solang
https://github.com/hyperledger/homebrew-solang

Communication

Mailing List

- https://lists.hyperledger.org/g/solang
  - subscribe
  - messages

Chat (for questions and ephemeral discussions)

Questions are welcome and best asked in Hyperledger Discord.

Learn more about Hyperledger Discord here, get the invite and check out one of the many Aries project channels.

Meeting

People who want to learn about or contribute to Solang should join this call. This does not replace our asynchronous collaboration, but should help us keep everyone up-to-date and moving together.
Discussion items: upcoming releases, current PRs, work that will generate future PRs, architecture changes that will impact downstream teams, project standards, best practices, design, etc.

Calendar

- Hyperledger Calendar of Public Meetings

History

- Proposed by
  - Sean Young
  - Lucas Steuernagel
  - Tracy A. Kuhrt
  - Cyrill Leutwiler
- Approved by the TSC on 2022-08-18

Recent space activity

- Ry Jones
  - Hyperledger Solang updated Apr 23, 2024
- David Boswell
  - Hyperledger Solang updated Oct 06, 2022
- Sean W. Bohan
  - Hyperledger Solang updated Sep 29, 2022
- Lucas Steuernagel
  - Hyperledger Solang updated Sep 24, 2022
- Sean Young
  - Hyperledger Solang updated Sep 22, 2022

Space contributors

- Ry Jones (45 days ago)
- David Boswell (610 days ago)
- Sean W. Bohan (617 days ago)
- Lucas Steuernagel (623 days ago)
- Sean Young (624 days ago)
- ...