Introduction to Smart Contracts and Solidity

Part 7 - Installation of the Solidity Compiler via npm

Versioning

Semantic versioning

- Solves *“dependency hell”*
  - Version lock
  - Version promiscuity
- Formed as a simple set of rules and requirements
- MAJOR.MINOR.PATCH
  - MAJOR= incompatible API changes
  - MINOR = backward-compatible adding of functionality
  - PATCH = backward-compatible bug fixes

Solidity versioning details

Solidity patch-level releases

- Code compiling with version A.x.y will always compile with A.x.z

Nightly development releases are also available

- No stability guaranteed / bleeding-edge
- Must not be included in a production system

Stable releases

- Always should include them in a production system

Remix

Integrated development environment for

- Developing...
- Deploying...
- Administering...
- ... smart contracts for Ethereum-like blockchains
- Suitable for small contracts and learning purposes

Remix in two forms

- Online edition
- Offline edition

Solidity compiler software

Several variants of Solidity compiler *solc* (by origin):

- npm / Node.js
- Docker container
- Linux packages / repository compiler
- Precompiled solidity compiler / static binary
- Source code for local compilation

Solidity compiler via npm / Node.js

npm = Node.js package manager

solc → solc-js (only in this variant)

Emscripten = compiler toolchain...

- ... to webAssembly
- ... using LLVM

Node.js installation/update procedure on Ubuntu Linux

Solidity compiler installation on Ubuntu Linux