Human-friendly Contract Interactions with sourcify Verification

Kaan Uzdoğan
Ethereum Foundation

Devcon VI
11.10.2022 - Bogotá
Just a normal day in web3...
null
A typical web3 interaction nowadays is *still* a YOLO-Signing nightmare
FUNCTION TYPE: Contract Interaction

HEX DATA: 195 BYTES

FUNCTION: register(string name, uint duration)

ARGUMENTS:
- name: "meetup-berlin.eth"
- duration: 1 year

DESCRIPTION:
Registers the domain "meetup-berlin.eth" for 1 year.

show source

✅ Fully verified on Sourcify
What can you do to achieve this...
What can you do to achieve this...

...as a smart contract developer?  
...as a wallet developer?
What can you do to achieve this...

...as a smart contract developer?  ...as a wallet developer?
As a contract developer:

1) Use NatSpec documentation
2) Source code verification on Sourcify
As a contract developer:

1) Use NatSpec documentation
2) Source code verification on Sourcify
Natspec

https://docs.soliditylang.org/en/develop/natspec-format.html

“Ethereum Natural Language Specification Format (NatSpec)”
contract Wallet {

/// @dev Allows to swap/replace an owner from the Safe with another address.
/// This can only be done via a Safe transaction.
/// @notice Replaces the owner `oldOwner` in the Safe with `newOwner`.
/// @param prevOwner Owner that pointed to the owner to be replaced in the linked list
/// @param oldOwner Owner address to be replaced.
/// @param newOwner New owner address.

function swapOwner(
    address prevOwner,
    address oldOwner,
    address newOwner
) public authorized {

contract Wallet {
...

/// @dev Allows to swap/replace an owner from the Safe with another address.
///     This can only be done via a Safe transaction.
/// @notice Replaces the owner `oldOwner` in the Safe with `newOwner`.
/// @param prevOwner Owner that pointed to the owner to be replaced in the linked list
/// @param oldOwner Owner address to be replaced.
/// @param newOwner New owner address.

function swapOwner(
    address prevOwner,
    address oldOwner,
    address newOwner
) public authorized {...
contract Wallet {

    /// @dev Allows to swap/replace an owner from the Safe with another address.
    /// This can only be done via a Safe transaction.
    /// @notice Replaces the owner `oldOwner` in the Safe with `newOwner`.
    /// @param prevOwner Owner that pointed to the owner to be replaced in the linked list
    /// @param oldOwner Owner address to be replaced.
    /// @param newOwner New owner address.
    function swapOwner(
        address prevOwner,
        address oldOwner,
        address newOwner
    ) public authorized {...
contract Wallet {

    /// @dev Allows to swap/replace an owner from the Safe with another address.
    /// This can only be done via a Safe transaction.
    /// @notice Replaces the owner `oldOwner` in the Safe with `newOwner`.
    /// @param prevOwner Owner that pointed to the owner to be replaced in the linked list
    /// @param oldOwner Owner address to be replaced.
    /// @param newOwner New owner address.

    function swapOwner(
        address prevOwner,
        address oldOwner,
        address newOwner
    ) public authorized {...

https://docs.soliditylang.org/en/develop/natspec-format.html
Natspec: Dynamic Expressions

/// @notice Replaces the owner `oldOwner` in the Safe with `newOwner`.
Natspec: Dynamic Expressions

@notice Replaces the owner `oldOwner` in the Safe with `newOwner`
Natspec: Dynamic Expressions

@notice Replaces the owner `oldOwner` in the Safe with `newOwner`

Becomes:

Replaces the owner 0xcC60F45e0507032036033b361d3a6457b9F0283D in the Safe with 0x83D0360050703233b361d3a6457b9F2cC60F45e0
Where to find userdoc (@notice) and devdoc (@dev)?
In Solidity **Contract Metadata:**

https://docs.soliditylang.org/en/latest/metadata.html
Solidity Contract Metadata

Introduced in Solidity v0.4.7 (2016-12-15) but was not picked up by the community
Solidity Contract Metadata

Introduced in Solidity v0.4.7 (2016-12-15) but was not picked up by the community

JSON file generated by the Solidity compiler which contains... metadata:
Solidity Contract Metadata

Introduced in Solidity v0.4.7 (2016-12-15) but was not picked up by the community

JSON file generated by the Solidity compiler which contains... metadata:

- ABI
- Userdoc + devdoc
- Compilation info
- Source file info
Solidity Contract Metadata

Introduced in Solidity v0.4.7 (2016-12-15) but was not picked up by the community

JSON file generated by the Solidity compiler which contains... metadata:

- ABI
- Userdoc + devdoc
- Compilation info
- Source file info

How to interact with the contract?
Solidity Contract Metadata

Introduced in Solidity v0.4.7 (2016-12-15) but was not picked up by the community

JSON file generated by the Solidity compiler which contains... metadata:

- ABI
- Userdoc + devdoc
- Compilation info
- Source file info

How to interact with the contract?

How to reproduce a contract compilation?
"userdoc": {
  "kind": "user",
  "methods": {
    "addOwnerWithThreshold(address,uint256)": {
      "notice": "Adds the owner `owner` to the Safe and updates the threshold to `_threshold`.
    },
    "changeThreshold(uint256)": {
      "notice": "Changes the threshold of the Safe to `_threshold`.
    },
    "disableModule(address,address)": {
      "notice": "Disables the module `module` for the Safe.
    },
    "enableModule(address)": {
      "notice": "Enables the module `module` for the Safe.
    },
    "removeOwner(address,address,uint256)": {
      "notice": "Removes the owner `owner` from the Safe and updates the threshold to `_threshold`.
    },
    "requiredTxGas(address,uint256,bytes,uint8)": {
      "notice": "Deprecated in favor of common/StorageAccessible.sol and will be removed in next major version.
    },
    "swapOwner(address,address,address,address)": {
      "notice": "Replaces the owner `oldOwner` in the Safe with `newOwner`.
    }
  }
}
Solidity Contract Metadata

$ solc --metadata MyContract.sol
$ truffle compile
$ cd build/contracts
$ cat MyContract.json
{
    "contractName": "MyToken",
    "abi": [...]
    "metadata": "{"compiler":{"version":"0.8.4+commit.c7e474f2"},"language":{
        ...
        ,"version":1}"
    ,"bytecode": "..."
...}
Solidity Contract Metadata

```json
$ hardhat compile
$ cd artifacts/build-info
$ cat 901568e56d422b1e1e3f64004cb4dd6e.json
{
  "id": "901568e56d422b1e1e3f64004cb4dd6e",
  "_format": "hh-sol-build-info-1",
  "solcVersion": "0.8.6",
  "solcLongVersion": "0.8.6+commit.11564f7e",
  "input": {
    ...
  },
  "output": {
    "contracts": {
      "MyContract.sol": {
        "MyContract": {
          "abi": [
            ...
          ],
          "evm": {
            "metadata": "{"compiler":{"version":"0.8.6+commit.11564f7e"},"language":"Solidity","output(...
```

Solidity Contract Metadata

Contract Bytecode

919050565b600082610b9a57634e487b7160e01b81526012600452602481fd5b506490565b80825b6001808611610bb157
50610bd5c565b81874821115610bc357610bc3610cd5565b80861615610bd057918102915b9490941c938002610ba2565b
9459492505050565b600061046860001960ff851684600082610c0157506001610468656b81610c0e575600061046856
5b8160018114610c245760028114610c2e57610c5b565b6001915050619468565b60ff841115610c3f57610c3f610cd556
5b6001841b915084821115610c5557610c55610cd5565b50610468565b5060208319610133831016604e8410600b841061
1715610c8e575081810a83811115610c8957610c89610cd5565b610468565b610c9b84d84846001610b9f565b8086048211
15610cad57610cad610cd5565b02949350505050565b6000816000190483118215151615610cd057610cd0610cd5565b50
0290565b634e487b7169e01b600052601160045260246000fdfe0264697066735822122978b530288f1cffe879bb7d9062
e904deb3aa9b9c8d27ea4ecaafa987583c18a6e64736f6c63430008010033
Solidity Contract Metadata

- The compiler takes the **IPFS hash** of the metadata file and appends to the bytecode alongside the compiler version.

```
ipfs://QmWTqspM5B1quNvdhXbS6TbXzyLZ5cUGHnTV8ZWJPrqQq
```

---

Contract Bytecode

```
919050565b600082610b9a157534e487b7160e01b8152601260045262481f5b50690565b6082b600180611610bb157
50610bdcc565b81870482115610bc357610bc3610cd556b80861615610bd57918102915b4949941c938092610b266b
94509942505050565b690061960ff851664600082610c015750001610468655b81610c0e5756009610468656
5b8160018114610c24576028114610c2e57610c5b56b6001915050610468656b0ff84115610c3f57610c3f610cd556
5b6001841b91508482115610c555610c355610cd556b5610468656b5602061061013831016604e841060b841016
171560c8e575081810a83811115610c8957610c89610cd556b610468656b610c9d8484846001610b9f565b8086048211
15610cad57610cad610cd556b0294935605050556b600081600019048311821515215610cd57610cd610cd556b50
0290565b634e487b7160e01b6000526016004526524600f6fe0256057610cd5555
```
Solidity metadata.json playground

Enter Contract Address or ENS
0x34a...456d

Decide

Click to decode some example contracts:
- SynthetixAMM
- Uniswap
- ENS
- XMON
- ShibaSwap
- PODK
- CollateralShort
- SushiSwap LP
- BUSD
- Wonderland DAO
- CollateralShort
- some-old-contract
- BinanceSC
- Avalanche
- without appended CBOR

or paste contract bytecode
0x608060405234801561001057600080fd5b5061012f8061002060...

Decide

Try it out in Playground: playground.sourcify.dev
As a contract developer:

1) Use NatSpec documentation
2) Source code verification on Sourcify
What is source code verification?
### Contract Overview

**Ethereum 2.0 Deposit Contract**

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Balance</td>
<td>7,906,082,000000000000000000000000000000000000 Ether</td>
</tr>
<tr>
<td>Value</td>
<td>$28,229,693,571.91 (@ $3,570.63/ETH)</td>
</tr>
<tr>
<td>Token</td>
<td>&gt;736,822.02 &gt;107</td>
</tr>
</tbody>
</table>

### More Info

- **My Name Tag:** Not Available, login to update
- **Creator:** 0xb20a608c824ca500093... at txn 0xe75f554e433e03763...

---

**Contract Source Code Verified (Exact Match)**

- **Contract Name:** DepositContract
- **Compiler Version:** v0.6.11+commit.5ef660b1
Source Code Verification

- Smart contract code lives in **bytecode** on the blockchain
- Not human-readable, machine readable
Source Code Verification

MyContract.sol

Ownable.sol

ERC20.sol

...
Source Code Verification

- MyContract.sol
- Ownable.sol
- ERC20.sol

Compilation Settings

```
version: "0.8.7+commit.e28d00a7",
optimizer: {
  enabled: true,
  runs: 200
},
```
Source Code Verification

- MyContract.sol
- Ownable.sol
- ERC20.sol
- ...
Solidity Contract Metadata

JSON file generated by the Solidity compiler which contains...

- ABI
- Userdoc + devdoc
- Compilation info
- Source file info

How to interact with the contract?

How to reproduce a contract compilation?
Source Code Verification

- MyContract.sol
- Ownable.sol
- ERC20.sol

Compilation Settings

```
version: "0.8.7+commit.e28d00a7",
optimizer: {
  enabled: true,
  runs: 200
},
```
Source Code Verification

- MyContract.sol
- Ownable.sol
- ERC20.sol

bytecode

0x6080604052600436106200010b5760003560e01c806379ba5097116200009b578063add50991162000069578063add509914620002d1578063d0e30db0146200113578063d4ee1d901462000305578063dce0b4e41462000327578063f2...
Source Code Verification

bytecode

0x60806040526004361062
00010b5760003560e01c80
6379ba5097116200009b57
8063add50991162000069
578063add509914620002
d1578063d0e30db0146200
0113578063d4ee1d901462
000305578063dce0b4e414
62000327578063f2...
Source Code Verification

bytecode

0x60806040526004361062
00010b5760003560e01c80
6379ba5097116200009b57
8063add50991162000069
578063add509914620002
d1578063d0e30db0146200
0113578063d4ee1d901462
00305578063dce0b4e414
62000327578063f2...
Source Code Verification

bytecode

```solidity
0x60806040526004361062
00010b5760003560e01c80
6379ba5097116200009b57
8063add50991162000069
578063add509914620002
d1578063d0e30db0146200
0113578063d4ee1d901462
000305578063dce0b4e414
6200327578063f2...
```

```
eth_getCode("0x011fDBf...64cc90BB26D0C")
```

```solidity
0x60806040526004361062
00010b5760003560e01c80
6379ba5097116200009b57
8063add50991162000069
578063add509914620002
d1578063d0e30db0146200
0113578063d4ee1d901462
000305578063dce0b4e414
6200327578063f2...
```
Source Code Verification

bytecode

```
0x60806040526004361062 00010b5760003560e01c80 6379ba5097116200009b57 8063addd50991162000069 578063addd509914620002 d1578063d0e30db0146200 0113578063d4ee1d901462 000305578063dce0b4e414 62000327578063f2...
```

```
eth_getCode("0x011fDBf...64cc90BB26D0C")
```

match?
# Source Code Verification

😊 **Partial match** = bytecodes match

😊 **Full / Perfect Match** = bytecode + metadata match

### Contract Bytecode

```
919050565b600082610b9a57634e487b7160e01b81526012600452602481fd5b500490565b80825b60018086161610bb157
50610bdc565b818704821115610bc357610bc3610cd5565b80861615610bd057918102915b9490941c938002610ba2565b
9450949250b050565b60061046866001960ff85168460082610c0157506001610468565b81610c0e575060061046856
5b8160018114610c245760028114610c2e57610c5b55b600191505610468565b60ff841115610c3f57610c3f610cd556
5b6001841b91508482115610c5557610c55610cd5565b50610468565b5060208310610133831016604e8410600b841016
1715610c8e575081810a83811115610c8957610c89610cd5565b610468565b610c9b8484846001610bf56b8086048211
15610cad57610cad610cd5565b02949350505050565b6000816000190483118215151615610cd55610cd0610cd5565b50
0299565b634e487b7160e01b600052601160045260246000fde264697066735822122078b530288f1cffe879bb7d962
e904deb3aa9b9c8d27ea4ecafa987583c18a6e64736f6c63430008010033
```
Source Code Verification

😊 **Partial match** = **bytecodes** match

😢 **Full / Perfect Match** = **bytecode** + **metadata** match
Source Code Verification

.partial match = bytecode match

.full / perfect match = bytecode + metadata match (compilation fingerprint)
Source Code Verification

😊 **Partial match** = bytecodes match

😊 **Full / Perfect Match** = bytecode + metadata match

Full matches **cryptographically guarantee** the whole compilation including the Solidity files are exactly the same as when deployed - even comments, variable names etc.

**How?**
Full/Perfect Verification: How?

- MyContract.sol
- Ownable.sol
- ERC20.sol
Full/Perfect Verification: How?

- MyContract.sol: Hashed → 0xb6ee9d...
- Ownable.sol: Hashed → 0x41e281...
- ERC20.sol: Hashed → 0x9fd73f...

...
Full/Perfect Verification: How?

- MyContract.sol: Hashed 0xb6ee9d...
- Ownable.sol: Hashed 0x41e281...
- ERC20.sol: Hashed 0x9fd73f...

Metadata

```javascript
{
  "compiler": {...}, // 1 item
  "language": "Solidity",
  "output": {...}, // 3 items
  "settings": {...}, // 6 items
  "sources": {
    "contracts/MyContract.sol": {
      "keccak256": "0xb6ee9d528b336942dd70d3b41e2811...
      "license": "GPL-3.0",
      "urls": [
        "bzz-raw://fe52c6e3c04ba5d83ede6cc1a43c45f...
        "dweb:/ipfs/QmawU3NMIWNwKbMarUdYCIvFvuFElTL...
      ]
    },
    "contracts/Ownable.sol": {...}, // 3 items
    "contracts/ERC20.sol": {...} // 3 items
  }
}
```
Full/Perfect Verification: How?

- MyContract.sol  
  Hashed  
  0xb6ee9d...

- Ownable.sol  
  Hashed  
  0x41e281...

- ERC20.sol  
  Hashed  
  0x9fd73f...

...
Full/Perfect Verification: How?

- MyContract.sol
  - Hashed: 0xb6ee9d...

- Ownable.sol
  - Hashed: 0x41e281...

- ERC20.sol
  - Hashed: 0x9fd73f...

---

Metadata

```json
{
  "compiler": {"..."}, // 1 item
  "language": "Solidity",
  "output": {"..."}, // 3 items
  "settings": {"..."}, // 6 items
  "sources": {
    "contracts/MyContract.sol": {
      "keccak256": "0xb6ee9d528b336942dd70d3b41e2811f5b67e9d8a5166480d4ef7d465211b4517",
      "license": "GPL-3.0",
      "urls": [
        "bzz-raw://fe52c6e3c04ba5d83ede6c1a43c45f4fa3a",
        "dweb:/ipfs/QmawU3NM1WNKbAUrUyCiFeuFE1TL...
      ]
    },
    "contracts/Ownable.sol": {"..."}, // 3 items
    "contracts/ERC20.sol": {"..."} // 3 items
  }
}
```
<table>
<thead>
<tr>
<th>Contract Bytecode</th>
</tr>
</thead>
<tbody>
<tr>
<td>919050565b600082610b9a57634e487b7160e01b81526012600452602481fd5b500490565b80825b6001808611610bb5750610bdc565b81870421115610bc357610bc3610cd5565b80861615610bd057918102915b9490941c9380092610b02565b945934925050565b600061046860001960ff851684600082610c015750600161046856b81610c0e5750600610468565b8160018114610c245760028114610c2e57610c5b565b6001915050610468656b60ff84111610c3f57610c3f5610c55610c5557610c55610cd5565b50610468565b506020831061013831016694e8410600b8410161715610c8e57508181a8381115610c8957610c89610cd5565b10468565b610c9b848484846001610b9f565b806048211015610cad57610cad610cd5565b029435050505656b60008260000e98311821511516510cd057610cd0610cd5565b500290565b635e4e87b7160e01b6009526011600452602460000fdefea264697066735822122078b530288f1cffe879bb7d9062e904deb3aa9b9c8d27e4e5c3fa987583c18a6e64736f6c634308010033</td>
</tr>
</tbody>
</table>
encoded match?

```
500490565b80825b600180861610bb157918102915b9490941c938002610ba2565b610468565b81610c0e57560060610468565b60ff841115610c3f57610c3f610c556831061013031016604e8410600b8410169c9848846001610b9f565b808604821115151615610cd57610cd610cd5565b505822122078b530288f1cffe879bb7d9062
```

```
Contract Bytecode
919050565b600082610b9a57634e487b7160e01b815b50610bdc565b818704821115610bc357610bc3610cd94509492505050565b600061046869001960ff8516845b8160018114610c245760028114610c2e57610c5565b601841b915084821115610c557610c55610cd5561715610c8e57508180a83811115610c8957610c89615610cad57610cad610cd5565b029493505050505050290565b634e487b7160e01b6000526011600452602a

```

match?
Full/Perfect Verification: How?

- MyContract.sol: Hashed → 0xb6ee9d...
- Ownable.sol: Hashed → 0x41e281...
- ERC20.sol: Hashed → 0x9fd73f...

Metadata
Full/Perfect Verification: How?

- MyContract-diff.sol  Hashed  0xb6ee9d...
- Ownable.sol  Hashed  0x41e281...
- ERC20.sol  Hashed  0x9fd73f...

Metadata
Full/Perfect Verification: How?

- **MyContract-diff.sol**
  - Hashed: 0xa2fc16...

- **Ownable.sol**
  - Hashed: 0x41e281...

- **ERC20.sol**
  - Hashed: 0x9fd73f...

...
Full/Perfect Verification: How?

- MyContract-diff.sol
  - Hashed
  - 0xa2fc16...

- Ownable.sol
  - Hashed
  - 0x41e281...

- ERC20.sol
  - Hashed
  - 0x9fd73f...

...
<table>
<thead>
<tr>
<th>Contract Bytecode</th>
</tr>
</thead>
<tbody>
<tr>
<td>919050565b6000082610b9a57634e487b7160e01b81526012600452602481fd5b500490565b80825b6001808611610bb157</td>
</tr>
<tr>
<td>50610bdc565b81870482115610bc357610bc3610cd5565b80861615610bd657918102915b9490941c938002610ba2565b</td>
</tr>
<tr>
<td>94509492505050565b6000610468660001960ff8516846000082610c015750600161046856b81610c0e575060061046856</td>
</tr>
<tr>
<td>5b8160018114610c245760028114610c2e57610c5b565b000191505061046856b600ff841115610c3f57610c3f610cd556</td>
</tr>
<tr>
<td>5b6001841b91508482115610c5557610c555610cd5565b5061046856b506020831061013831016684e8410600b841016</td>
</tr>
<tr>
<td>1715610c8e575081810a8381115610c8957610c89610cd5565b1046856b610c9b8498464001610b9f565b8086048211</td>
</tr>
<tr>
<td>15610cad57610cad610cd5565b02943530505050565b600081600204831821515161510cd057610cd0610cd5565b50</td>
</tr>
<tr>
<td>0290565b634e487b7160e01b600052601160045260246000f6f6e264697066735822122078b530288f1cffe879bb7d9062</td>
</tr>
<tr>
<td>e904deb3aa9b9c8d27ea4ecafa987583c18a6e64736f6e36430008010033</td>
</tr>
<tr>
<td>Contract Bytecode</td>
</tr>
<tr>
<td>-------------------</td>
</tr>
<tr>
<td>919050565b600082610b9a57634e487b7160e01b81526012600452602481fd5b500490565b8082b60018086161610bb157</td>
</tr>
<tr>
<td>50610bdc565b81870421115610bc357601bc3610cd556b80861615610bd05791802915b9490941c938002610ba2565b</td>
</tr>
<tr>
<td>945094925050505056b6000610468600001960f85169460002610c0157506001161468656b18610ce0570600601046856</td>
</tr>
<tr>
<td>5b8160018114610c245760028114610c2e57610c5b565b6001915050601468656b60ff841115610c3f57610c3f610cd5565</td>
</tr>
<tr>
<td>5b6001841b95760182115610c557610c55610cd5566b0160045260246000f00000000000000000000000000000000</td>
</tr>
<tr>
<td>288f1c904debff8d27ea4eca18a6e6fa987553c4736f6c63430908010033</td>
</tr>
</tbody>
</table>
match?
How to verify?
sourcify.dev UI

Verifier
Verify smart contracts by recompiling with the Solidity source code and metadata.

Old verifier UI available at legacy.sourcify.dev

File Add Zone
Add the Solidity source files and metadata of all contracts you want to verify.

- Import from remote
- Import from Etherscan
- Import from GitHub

Contracts

Storage

Added Files (2)
- 0x00878Ac0D6B8d981ae72BA7c9C967eA0Fae69df4/metadata.json
- 0x00878Ac0D6B8d981ae72BA7c9C967eA0Fae69df4/sources/contracts/1_Storage.sol
API

Sends provided files for verification.

**URL**: `/verify` or `/`

**Method**: `POST`

**Content-Type**: `multipart/form-data` or `application/json`

```json
{
    "address": ..., 
    "chain": ..., 
    "files": {
        "file-name1.sol": ..., 
        "file-name2.sol": ...
    }
}
```
API

- Check by addresses (full match): `GET /check-by-addresses?addresses={addresses}&chainIds={chainIds}`
- Check by addresses (full or partial match): `GET /check-by-all-addresses?addresses={addresses}&chainIds={chainIds}`
- Get file tree (full match): `GET /files/tree/:chain/:address`
- Get file tree (full or partial match): `GET /files/tree/any/:chain/:address`
- Get source files (full match): `GET /files/:chain/:address`
- Get source files (full or partial match): `GET /files/any/:chain/:address`
- Get contract addresses (full or partial match): `GET /files/contracts/:chain`

[docs.sourcify.dev](https://docs.sourcify.dev)
Tooling

@wighawag/hardhat-deploy

$ hardhat --network mainnet sourcify
Tooling

Remix Plugin

Upload, verify & publish contract metadata and sources.

Note:
1. Compile contracts in Remix, before attempting verification
2. The metadata must be exactly the same as at deployment time

Search repository for verified contract here.

Ethereum Mainnet

0x2738d13E81e30bC615766A041

Verify

// SPDX-License-Identifier: 0.7.0 < 0.9.0
pragma solidity >=0.7.0 < 0.9.0;

/**
 * @title Storage
 * @dev Store & retrieve value
 * @custom:dev-run-script ./s
 */

contract Storage {

  uint256 number;

  /**
   * @dev Store value in variable
   * @param num value to store
   */
  function store(uint256 num)
  {
    number = num;
  }

  /**
   * @dev Return value
   * @return value of 'number'
   */
  function retrieve() public
  {
    return number;
  }
}
Tooling

Foundry

```shell
$ forge create Contract --verify --verification-provider sourcify
$ forge verify-contract <address> --verifier sourcify
$ forge verify-check <address> --verifier sourcify
```
Automatic Verification

- i.e. **Monitor**
- Catches contract creations
- Tries to fetch
  1) Metadata
  2) Source files from IPFS
     - Metadata file contains source file IPFS hashes too
- **Automatically compiles and verifies**
As a contract developer:

1) Use NatSpec documentation
2) Source code verification on Sourcify
As a contract developer:

1) Use NatSpec documentation
2) Source code verification on Sourcify
As a contract developer:

1) Use NatSpec documentation
2) Publish (pin) metadata and source files on IPFS
As a contract developer:

1) Use NatSpec documentation
2) Publish (pin) metadata and source files on IPFS

If you publish on IPFS we verify for you! 🎉
Contract repository

- Served over HTTP ([repo.sourcify.dev](http://repo.sourcify.dev)) and IPFS
- We pin the verified contract source files and metadata so that they are accessible by decoding the bytecode.
  
```
  15610cad57610cad610cd5565b0294935050505565b6008160001904831182151516510cd057610cd0610cd5565b50
  0290565b634e487b766e01b60052601160045260246000fdefa2646970667358221222278b530288f1c9e879bb7d9862
  e904deb3aa9b9c8d27ea4ecafa987583c18a6e64736f6c63430008618033
```

**Metadata Hash (decoded)**

```
ipfs://QmWTqspM5B1quNvdhXbS6TbxzLY5cUGHnTV8ZWPq5pQqj
```

- IPNS updated regularly

- [/ipns/repo.sourcify.dev](http://ipns/repo.sourcify.dev)
What can you do to achieve this...

...as a smart contract developer?

...as a wallet developer?
What can you do to achieve this...

...as a smart contract developer?

...as a wallet developer?
What is the goal again?

Function
depositETH(address, address, uint)

Arguments
pool: 0xd9Db270c1B5E3Bd161E8c8503c55cEABeE709552

onBehalfOf: 0x68b3465833fb72A70ecDF485E0e4C7bD8665Fc45

referralCode: 0

deposits WETH into the pool 0xd9Db270c1B5E3Bd161E8c8503c55cEABeE709552, on behalf of the account 0x68b3465833fb72A70ecDF485E0e4C7bD8665Fc45, using native ETH. A corresponding amount of the overlying asset is minted.
As a wallet developer

1) Fetch metadata

https://repo.sourcify.dev/contracts/full_match/{chainId}/{address}/metadata.json
As a wallet developer

1) Fetch metadata

https://repo.sourcify.dev/contracts/full-match/{chainId}/{address}/metadata.json

Remember you can get it through the contract bytecode!
As a wallet developer

1) Get contract bytecode  `eth_getCode("0x1fa47...b2c53f")`
As a wallet developer

1) Get contract bytecode  `eth_getCode("0x1fa47...b2c53f")`

```text
15610cad57610cad610cd5565b02949350505050565b6000816000199483118215151615610cd057610cd0610cd5565b500290565634e487b7160e01b600052601160045260246000f0dfeaea26469706673582212207853088f1cffe879bb7d9062e904deb3aa9b9c8d27ea4ecafa987583c18a6e64736f6c63430008010033
```

2) Fetch metadata via the IPFS hash

We pinned it for you!

**Metadata Hash (decoded)**

`ipfs://QmWTqspM5B1quNvdhXbS6TbXzyLZ5cUGHnTV8ZWJPqrQqj`
Solidity Contract Metadata

JSON file generated by the Solidity compiler which contains... metadata:

- ABI
- Userdoc + devdoc
- Compilation info
- Source file info

How to interact with the contract?

How to reproduce a contract compilation?
As a wallet developer

1) Get contract bytecode  
   `eth_getCode("0x1fa47...b2c53f")`

   15610cad57610cad610cd5565b029493505050565b6000816000190483118215151615610cd057610cd0610cd5565b50
   0290565b634e487b7160e01b600052601160045260246000fdfe364697066735822122078b530288f1cffe879bb7d9062
   e904deb37a9b9c8d27ea4eccafa987583c18a6e64736f6c634300008010033

2) Fetch metadata via the IPFS hash
   
   We pinned it for you!

   Metadata Hash (decoded)
   `ipfs://QmWTqspM5B1quNvdhXbS6TbXzyLZ5cUGHnTV8ZWJPqrQqj`

3) Decode **ABI** + populate **NatSpec** comments for the method
FUNCTION: register(string name, uint duration)

ARGUMENTS:
- name: "meetup-berlin.eth"
- duration: 1 year

DESCRIPTION:
Registers the domain "meetup-berlin.eth" for 1 year.

show source
✅ Fully verified on Sourcify
Ways to “human-friendliness”

- **Sourcify**
  - Idea: show human readable descriptions via the NatSpec comments found in the metadata

- **EIP-4430 & EIP-3224**
  - Richard Moore (@ricmoo) and Nick Johnson (@arachnid)
  - Idea: Include a `eipXXXdescribe` function inside the contract that will return a human readable description
  - Can decode things like an ENS `commit(bytes32 hash)` but extra data (gas) in contract.

- **EIP: Rich Site-Proposed Contract Metadata**
  - Dan Finlay (@danfinlay)
  - Idea: The first point of contact with the contract address proposes metadata to the wallet
    - visit app.uniswap.org → receive ABI + method describers from the website → save to the wallet
  - Backwards compatible and flexible but not bound to the contract itself
Ways to better UX:

- How many times was this contract interacted with?
- When was this contract deployed?
- Is this contract audited and by whom?
Recap: What is Sourcify?

Technically:

- An open-sourced **automatic** smart contract verification service (**Monitor**)
- A user interface (**UI**), API (**Server**), and tooling (**Plugins**) to verify contracts manually
- A public, decentralized content-addressed storage (IPFS) of verified contracts (**Repo**)

More generally:

- A **base layer** and **public good** for other tools to build on top
- An initiative to foster the use of Solidity contract metadata, NatSpec, and full verification
- An ongoing effort to improve smart contract UX, safety, and transparency
Thank you