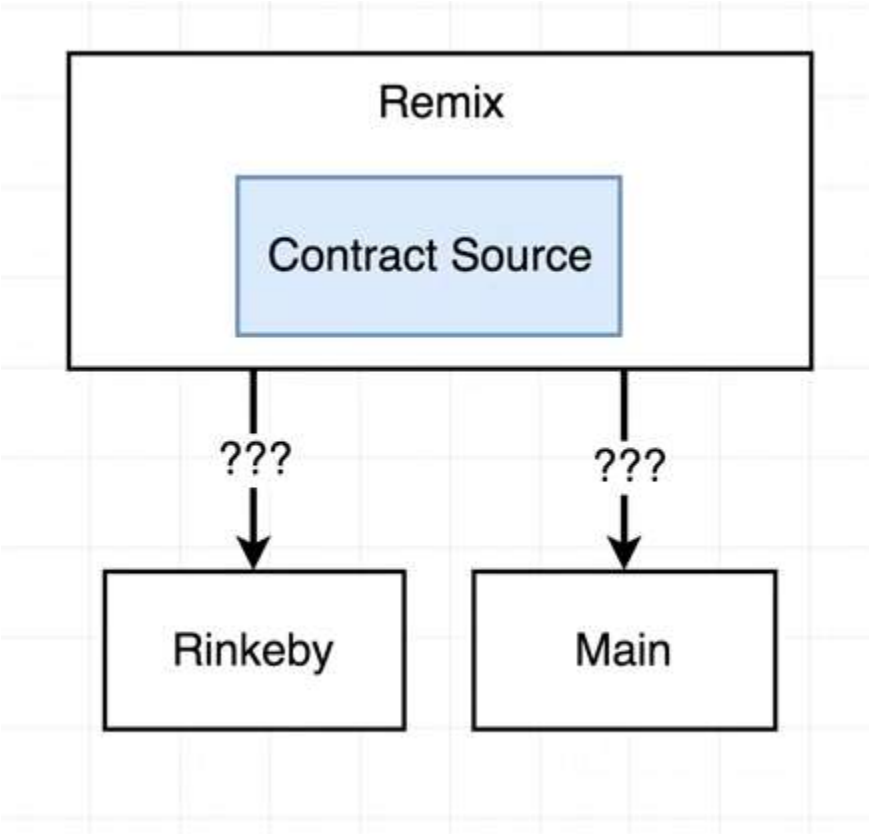
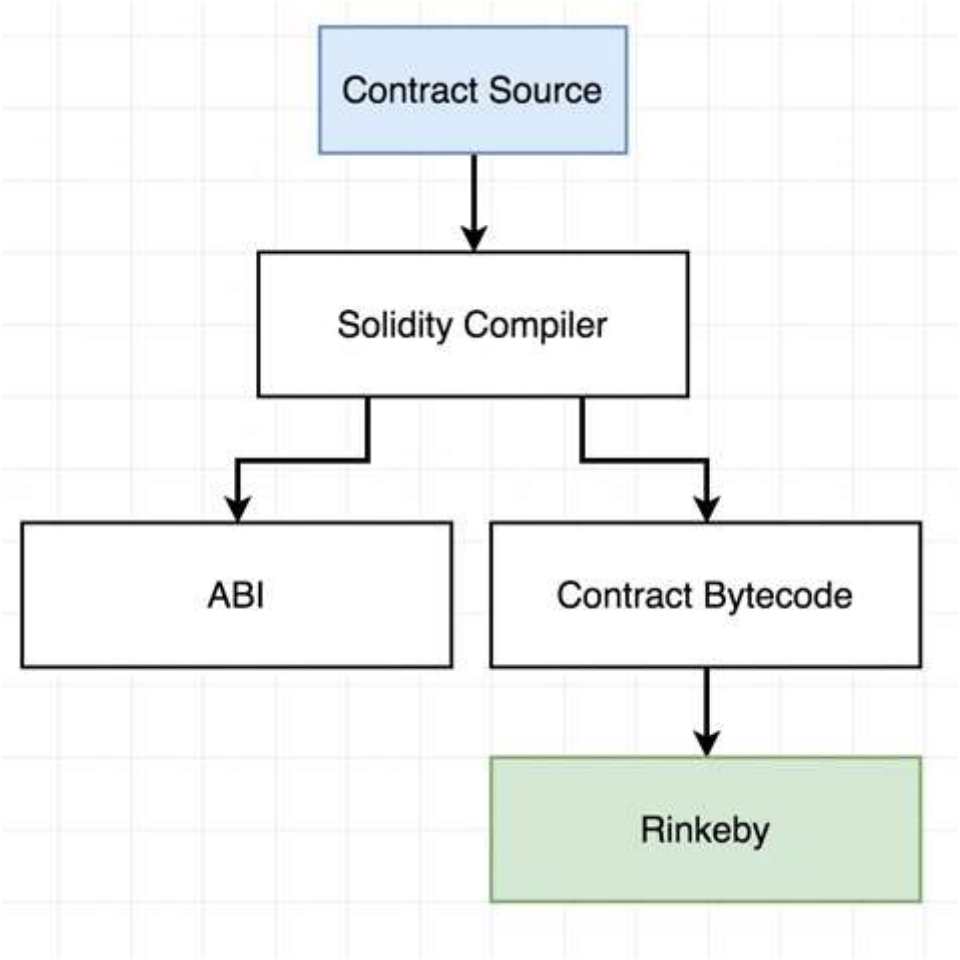
ethereum

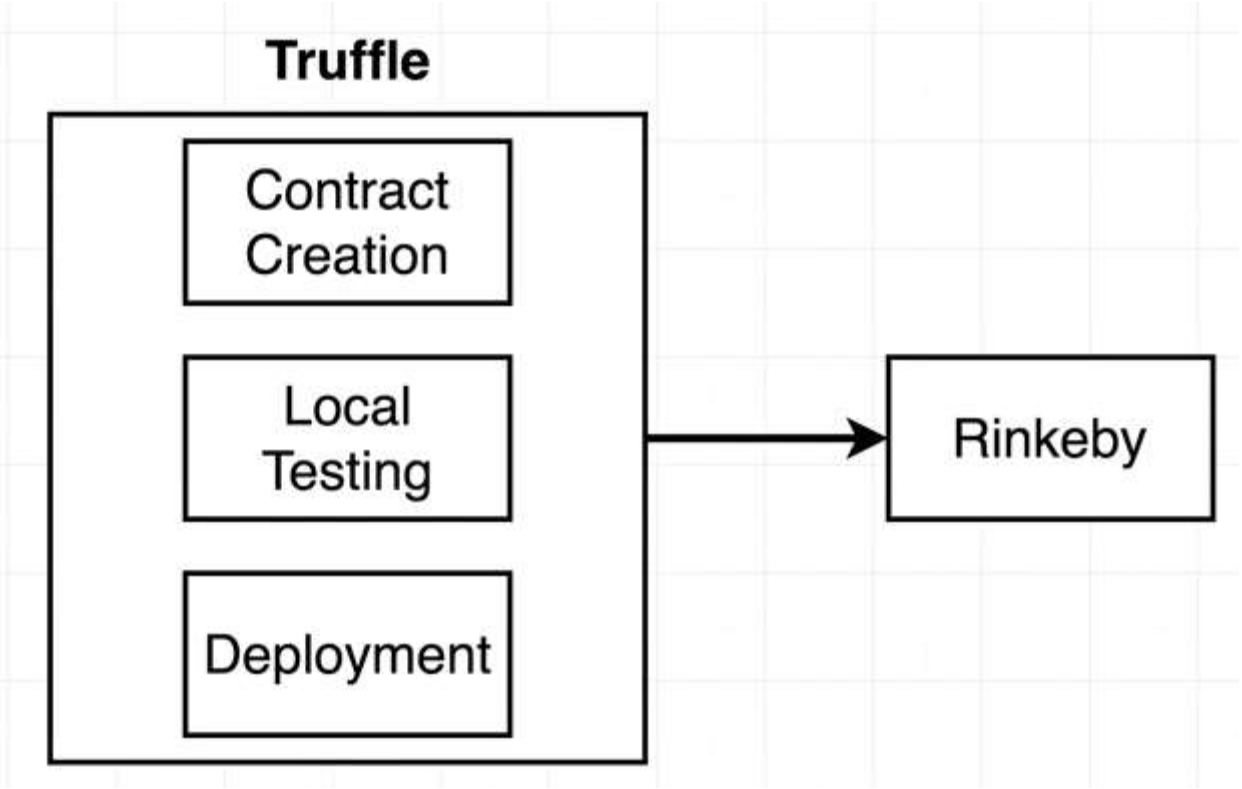
# How to deploy the contract on real network



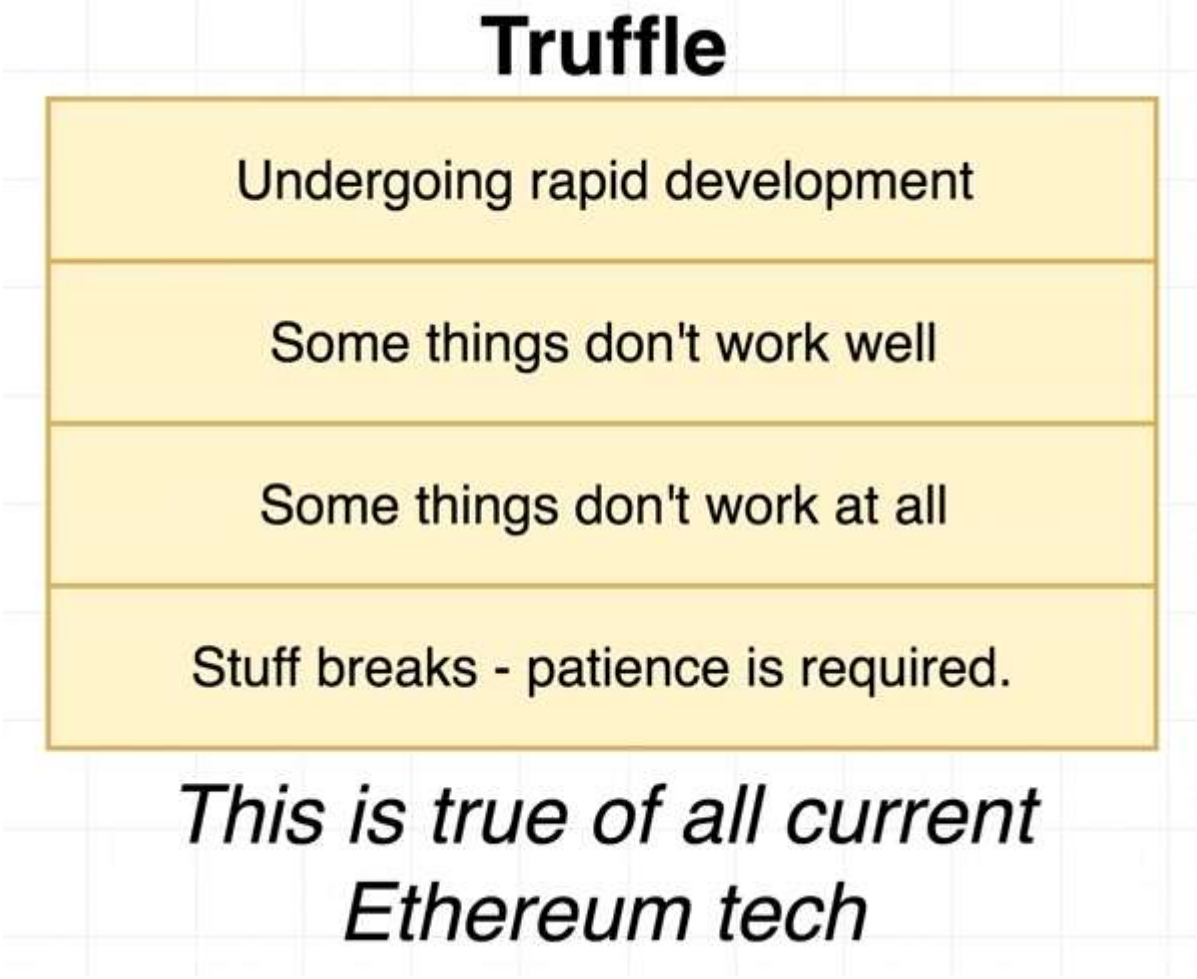
# At core it will follow same process



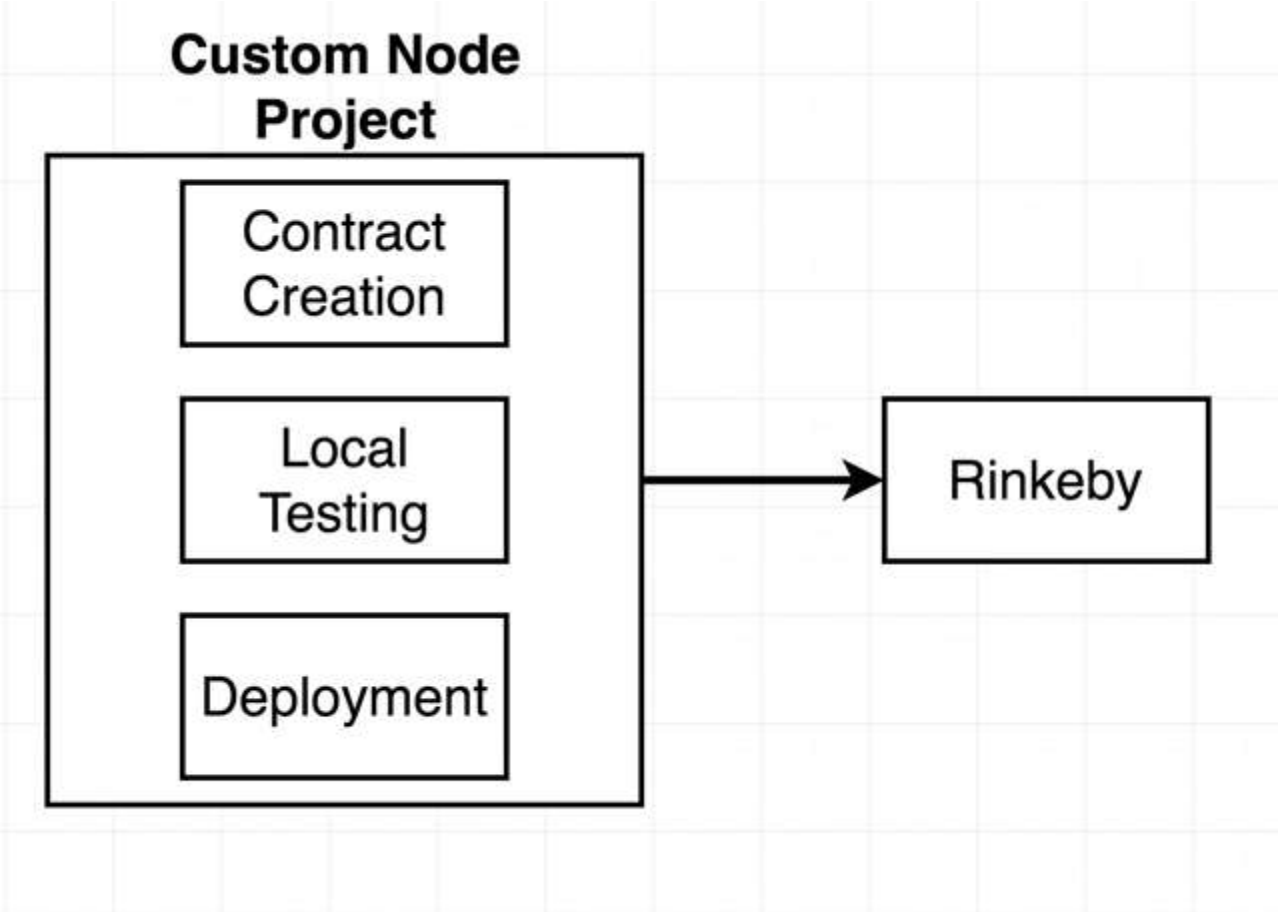
# For that we need truffle



# Issues with truffle



# Other option



# Boilerplate

| Boilerplate Design | |
|---|---|
| **Issue** | **Solution** |
| Need to be able to write Solidity code in a Javascript project | Set up the Solidity compiler to build our contracts |
| Need some way to rapidly test contracts without doing the manual testing we were doing with Remix | Set up a custom Mocha test runner that can somehow test Solidity code |
| Need some way to deploy our contract to public networks | Set up a deploy script to compile + deploy our contract |

# First Project from Scratch

- Open VS code (or any other editor)
- Open terminal
- Go to the location where you want to create project

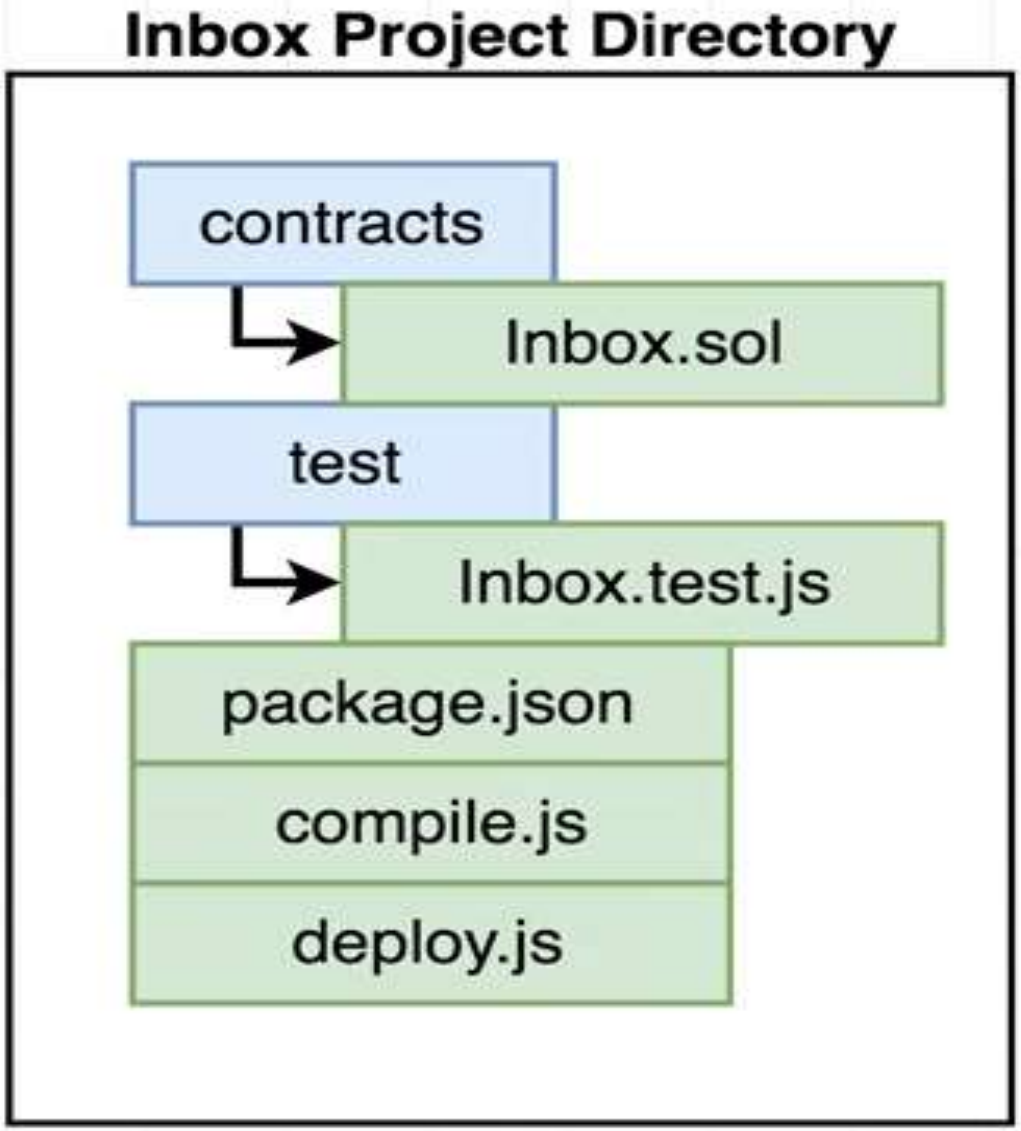
**mkdir Inbox**

**cd Inbox**

**npm init**

- //Do not provide any detail, press enter multiple times
- Now, the **package.json** file has been created

# Project structure



# First file


Project
inbox
contracts
package.json
Inbox.sol
1 pragma solidity ^0.4.17;
2
3 contract Inbox {
4 string public message;
5
6 function Inbox(string initialMessage) public {
7 message = initialMessage;
8 }
9
10 function setMessage(string newMessage) public {
11 message = newMessage;
12 }
13 }
14

# Inbox.sol (new version)

```
pragma solidity ^0.8.13;

contract Inbox {
    string public message;

    constructor (string memory initialMessage) {
        message = initialMessage;
    }

    function setMessage(string memory newMessage) public {
        message = newMessage;
    }
}
```

# Solidity compiler

- npm install - - save solc
- Make new file compile.js

compile.js

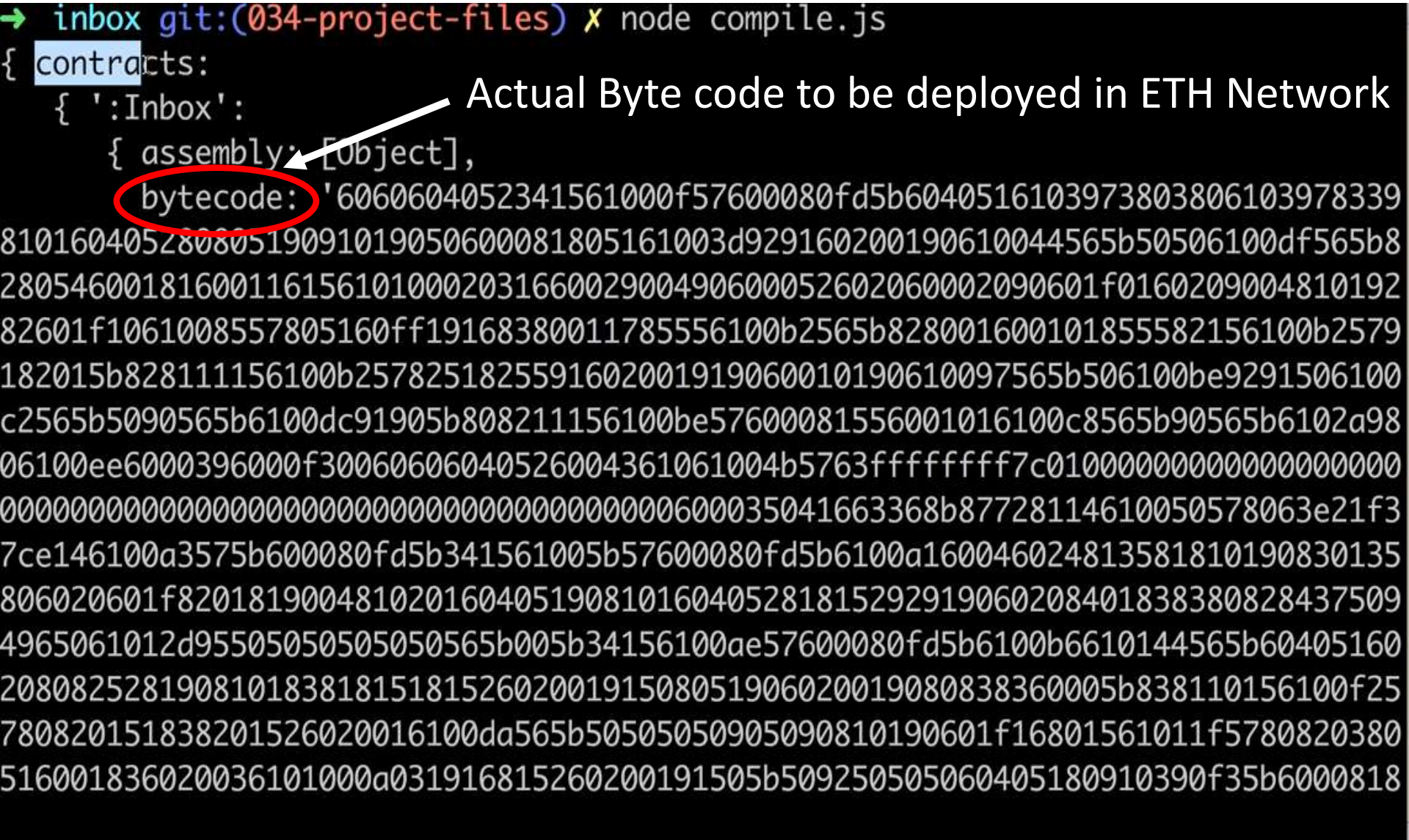
```
const path = require('path');
const fs = require('fs');
const solc = require('solc');

const inboxPath = path.resolve(__dirname, 'contracts', 'Inbox.sol');
const source = fs.readFileSync(inboxPath, 'utf8');

console.log(solc.compile(source, 1));
```

# Solidity compiler (compiler.js)

```
//Path from compiler to .sol file & provide cross platform compatibility
const path = require('path');

const fs = require('fs');       // import file system
const solc = require('solc');       //Solidity Compiler

const inboxPath = path.resolve(__dirname, 'contracts', 'Inbox.sol');
//__dir → current working directory, contracts → directory

const source = fs.readFileSync(inboxPath, 'utf8');

// console.log(solc.compile(source, 1));
// 1 → number of components to compile
// log → generating log of compilation process
```

```
console.log (solc.compile(
    JSON.stringify({
        language: "Solidity",
        sources: {
            "Inbox.sol": {
                content: source
            }
        },
        settings: {
            outputSelection: {
                "*": {
                    "*": ["evm", "bytecode"]
                }
            }
        }
    })
));
```

# Compile the code

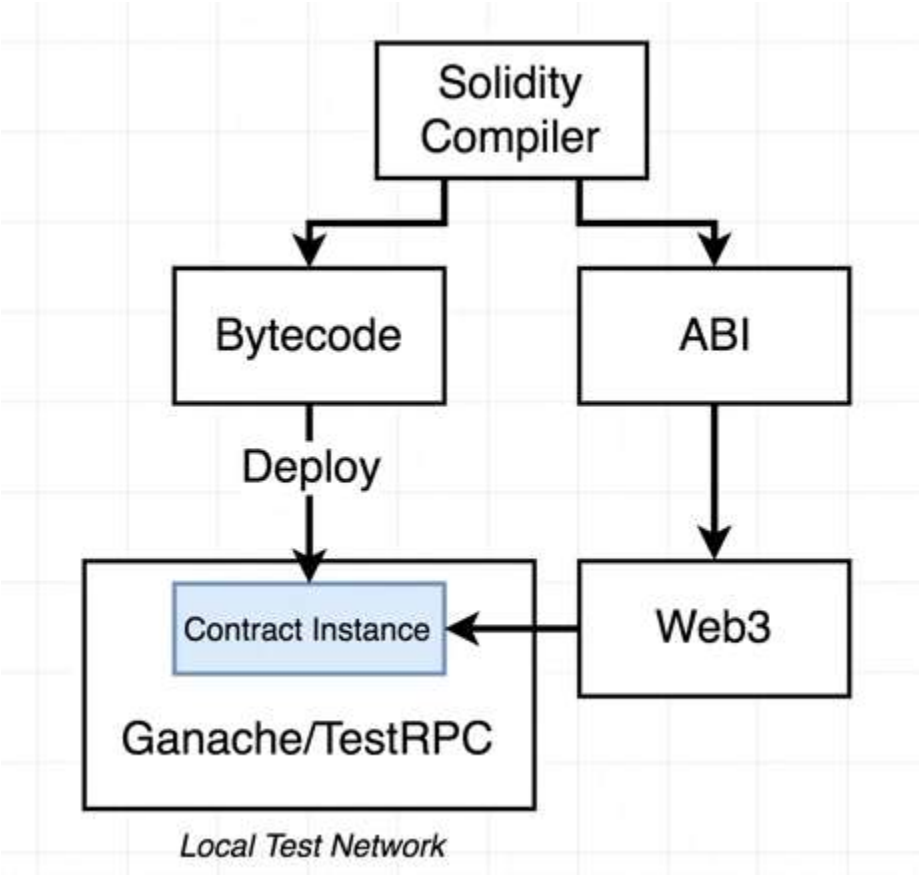


# ABI (Application Binary Interface)


Interface contains: Arguments, type of arguments, return value etc.

**module.exports = solc.compile(source, 1).contracts[':Inbox'];**

# Testing setup



# Installations

**npm install --save mocha ganache-cli web3@ 1.0.0-beta.26**

**npm install --save mocha ganache-cli web3**

//specific version of web3

Create a new folder test and create **inbox.test.js** file in that folder.

**const assert = require('assert');** //assertion of tests

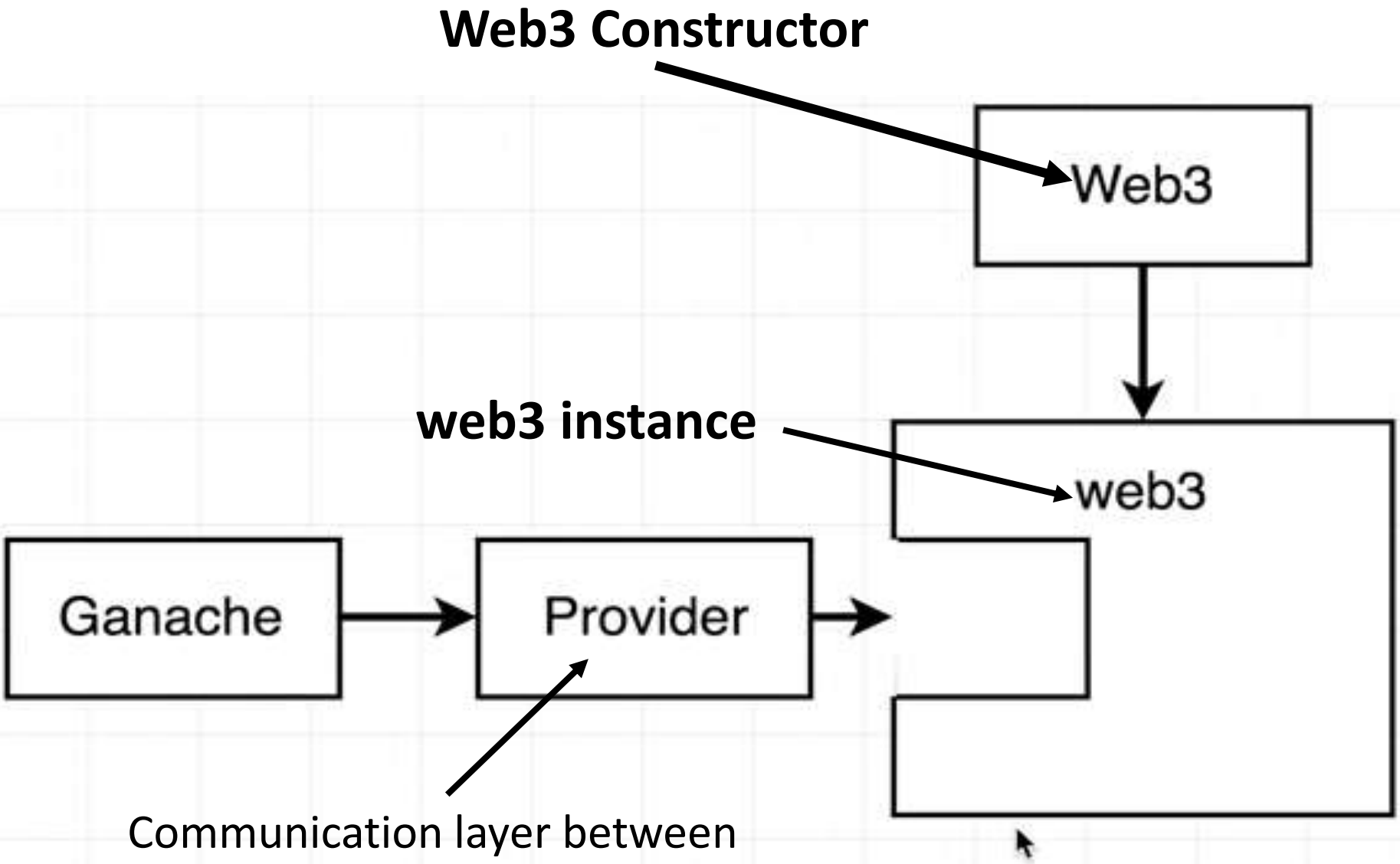
**const ganache = require('ganache-cli');**

**const Web3 = require('web3');** // Web3 is constructor

**const web3 = new Web3(ganache.provider());**

//web3 → instance of Web3

# Web3 Providers



# Mocha

| Mocha Functions | |
|---|---|
| Function | Purpose |
| it | Run a test and make an assertion. |
| describe | Groups together 'it' functions. |
| beforeEach | Execute some general setup code. |

# Inbox.test.js

```
→  inbox git:(040-providers) ✗ npm run test

> inbox@1.0.0 test /Users/stephengrider/workspace/
inbox
> mocha



  Car
    ✓ can park


  1 passing (34ms)
```

```
class Car {
  park() {
    return 'stopped';
  }

  drive() {
    return 'vroom';
  }
}
```

In package.json

```
"scripts": {
  "test": "mocha"
},
```

```
describe('Car', () => {
  it('can park', () => {
    const car = new Car();
    assert.equal(car.park(), 'stopped');
  });
});
```

# Using before each

```
let car;

beforeEach(() => {
  car = new Car();
});

describe('Car', () => {
  it('can park', () => {
    assert.equal(car.park(), 'stopped');
  });

  it('can drive', () => {
    assert.equal(car.drive(), 'vroom');
  });
});
```

```
const assert = require('assert');
const ganache = require ('ganache-cli')
const Web3 = require ('web3')
const web3 = new Web3(ganache.provider());

class Car {
    park() {
        return 'stopped';
    }

    drive() {
        return 'vroom';
    }
}
```

```
/* let car;

beforeEach(() => {
    car = new Car();
}); */

describe('Car1', () => {
    it('can park', () => {
        const car = new Car();
        assert.equal(car.park(), 'stopped');
    });

    it('can drive', () => {
        const car = new Car();
        assert.equal(car.drive(), 'vroom');
    });
});
```