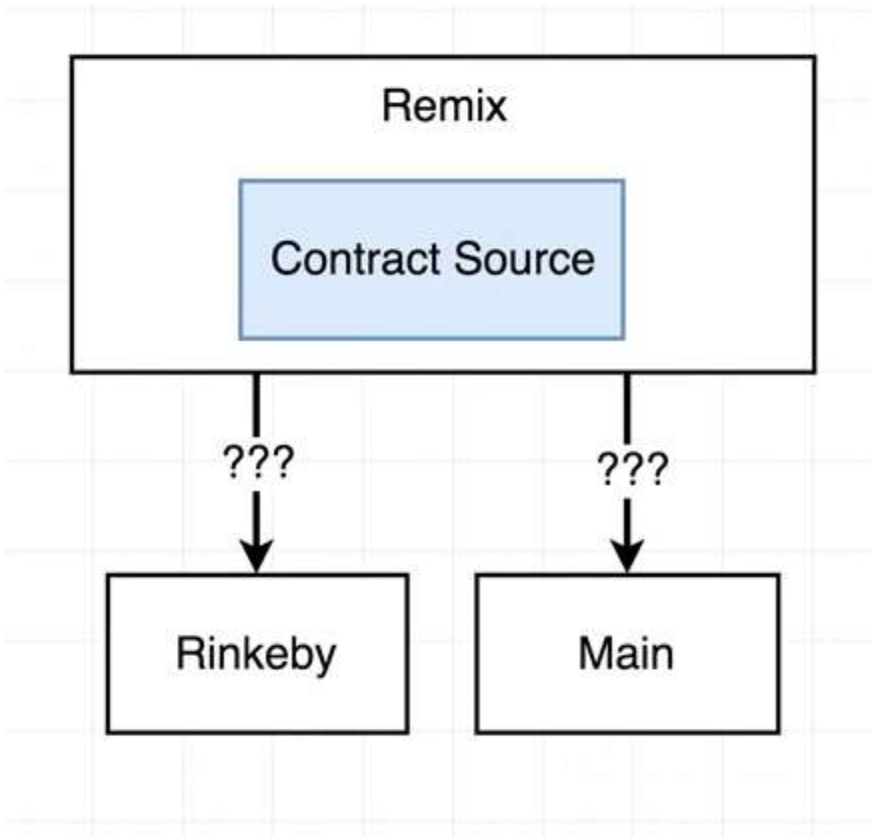
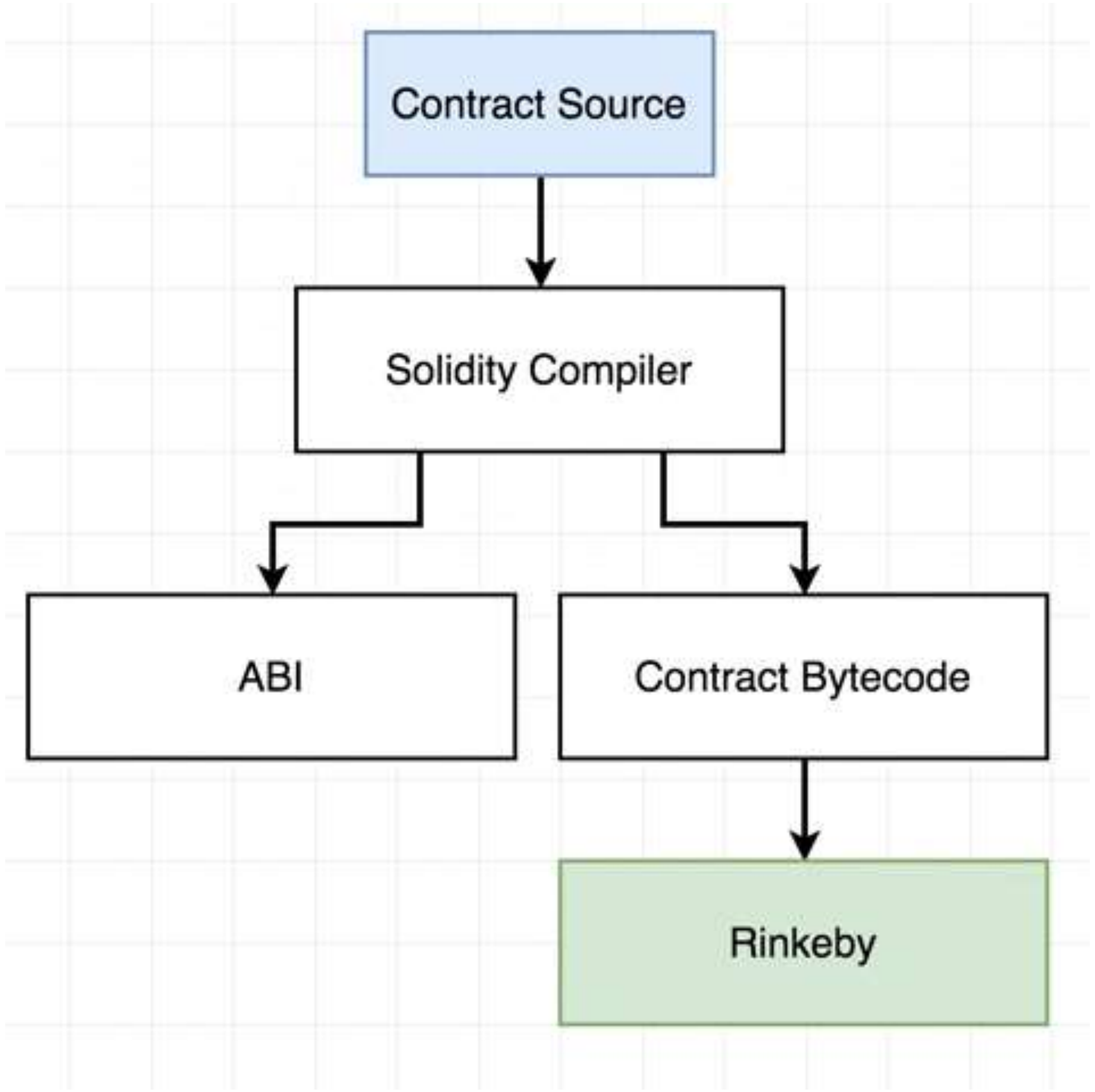
ethereum

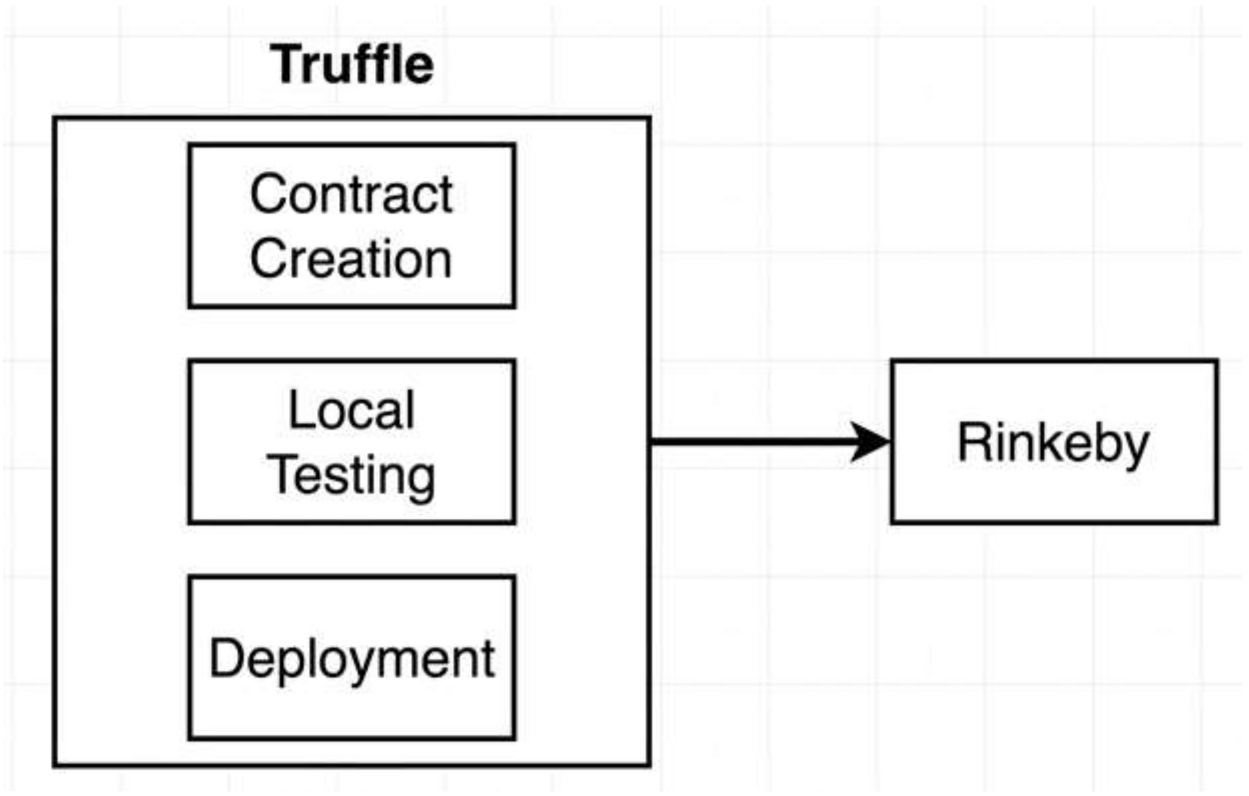
How to deploy the contract on real network



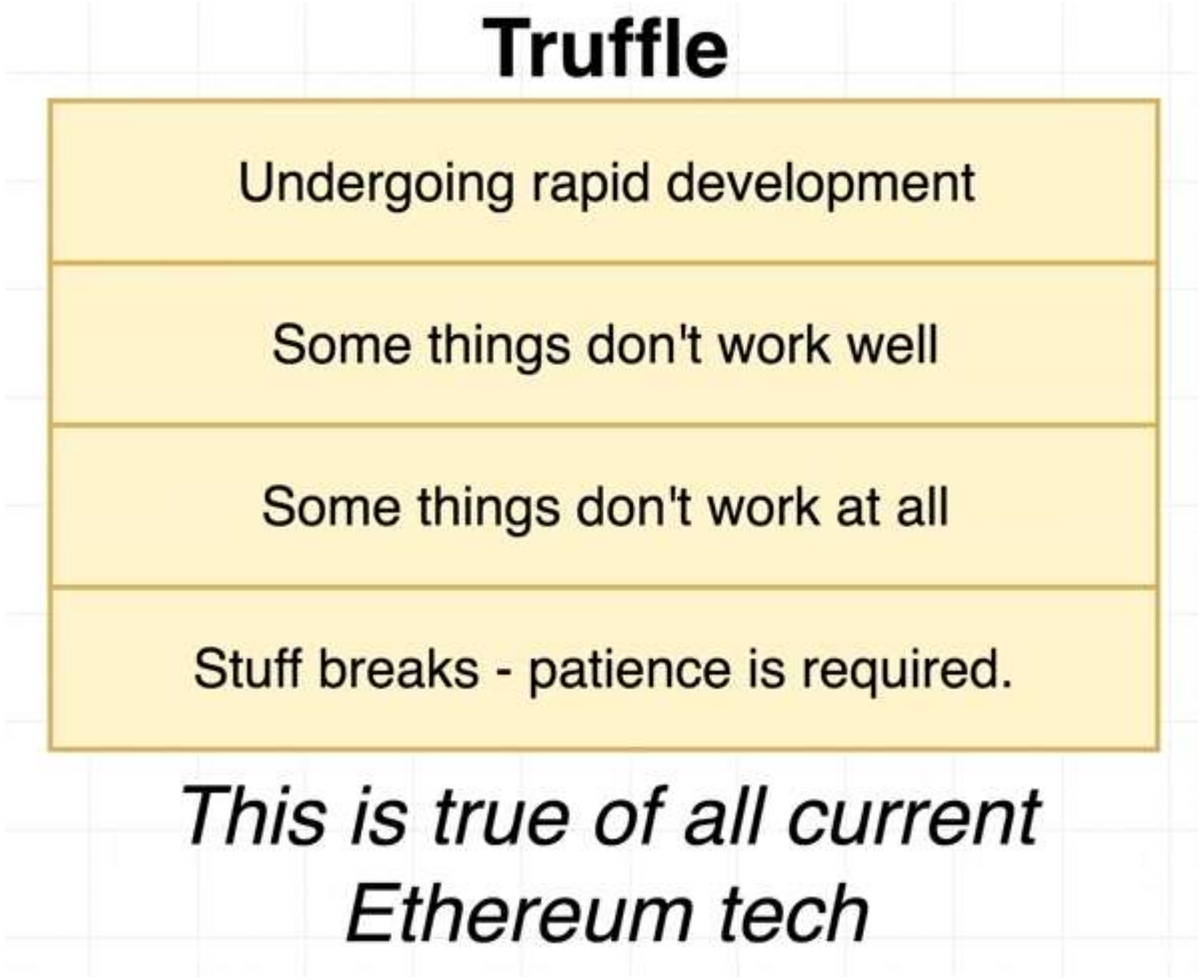
At core it will follow same process



For that we need truffle



Issues with truffle



Truffle

Undergoing rapid development

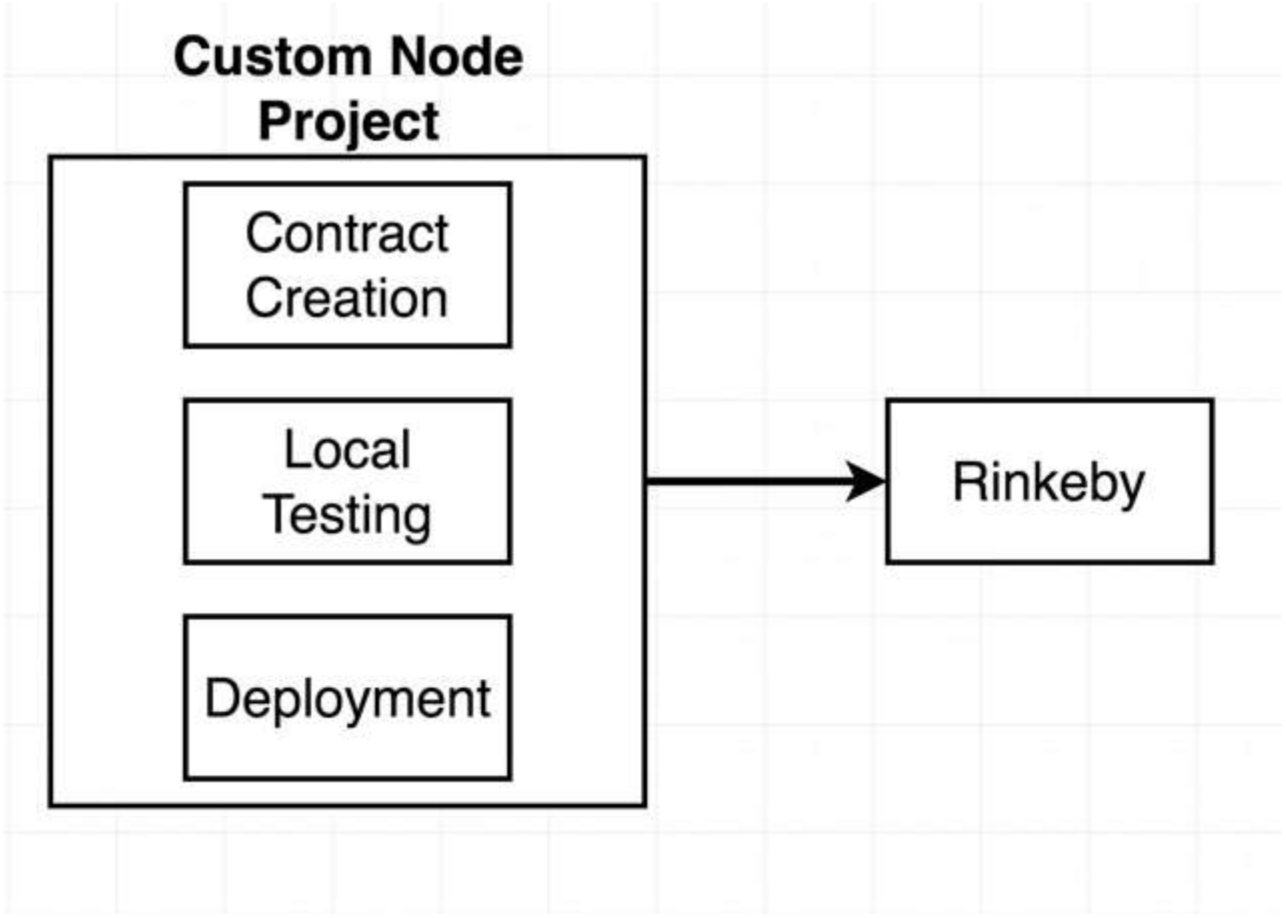
Some things don't work well

Some things don't work at all

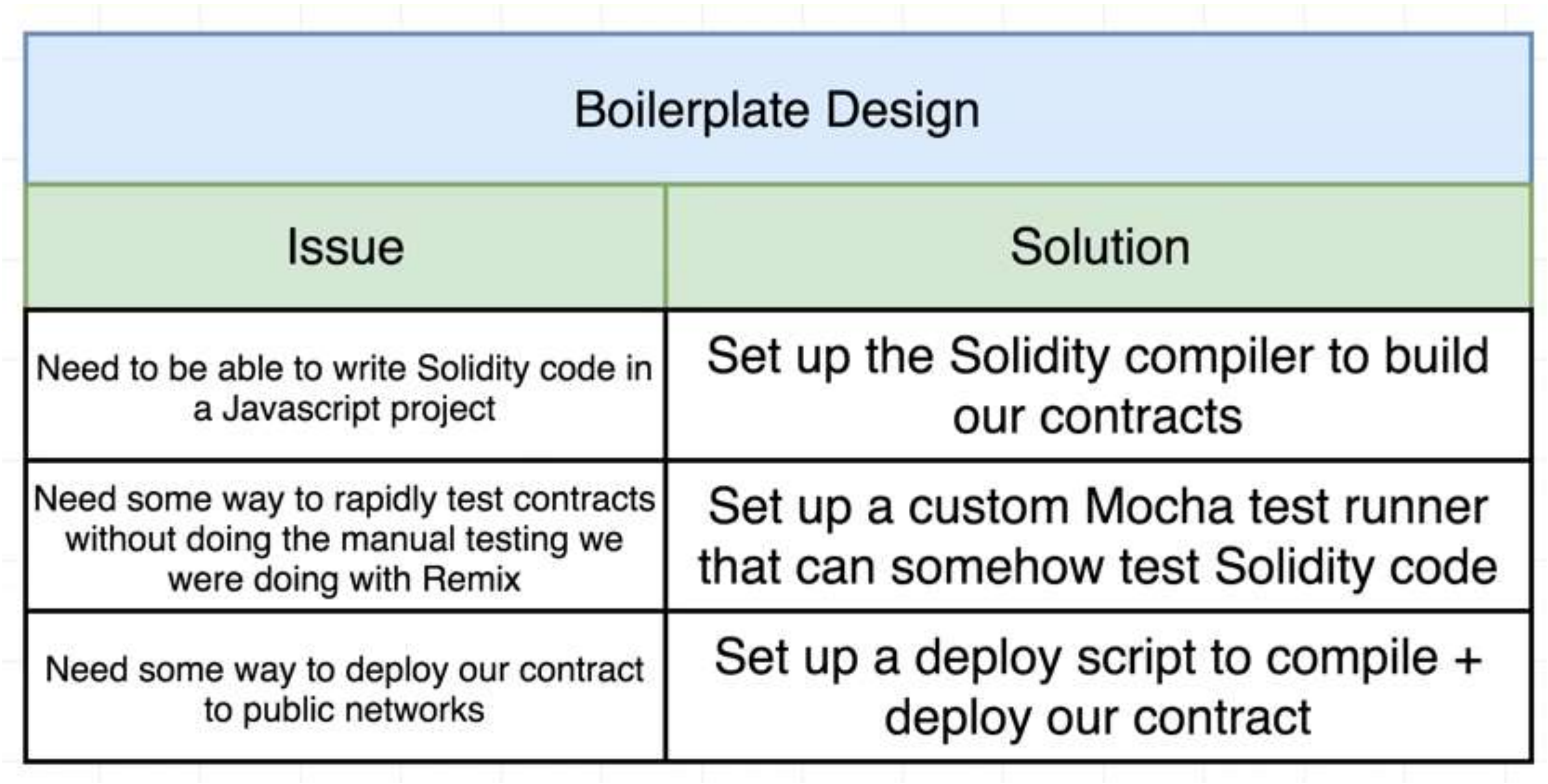
Stuff breaks - patience is required.

This is true of all current Ethereum tech

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

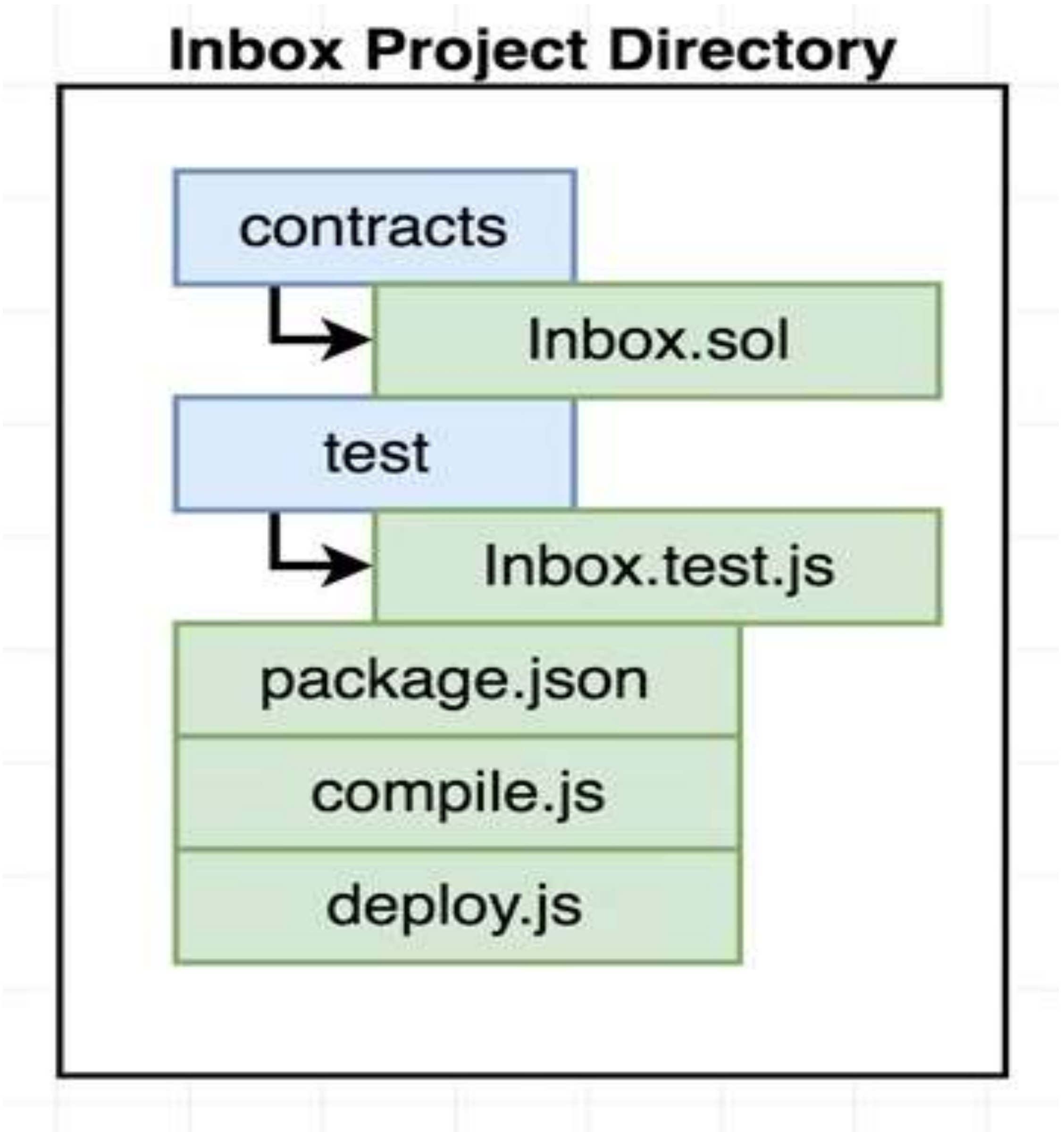
First Project from Scratch

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

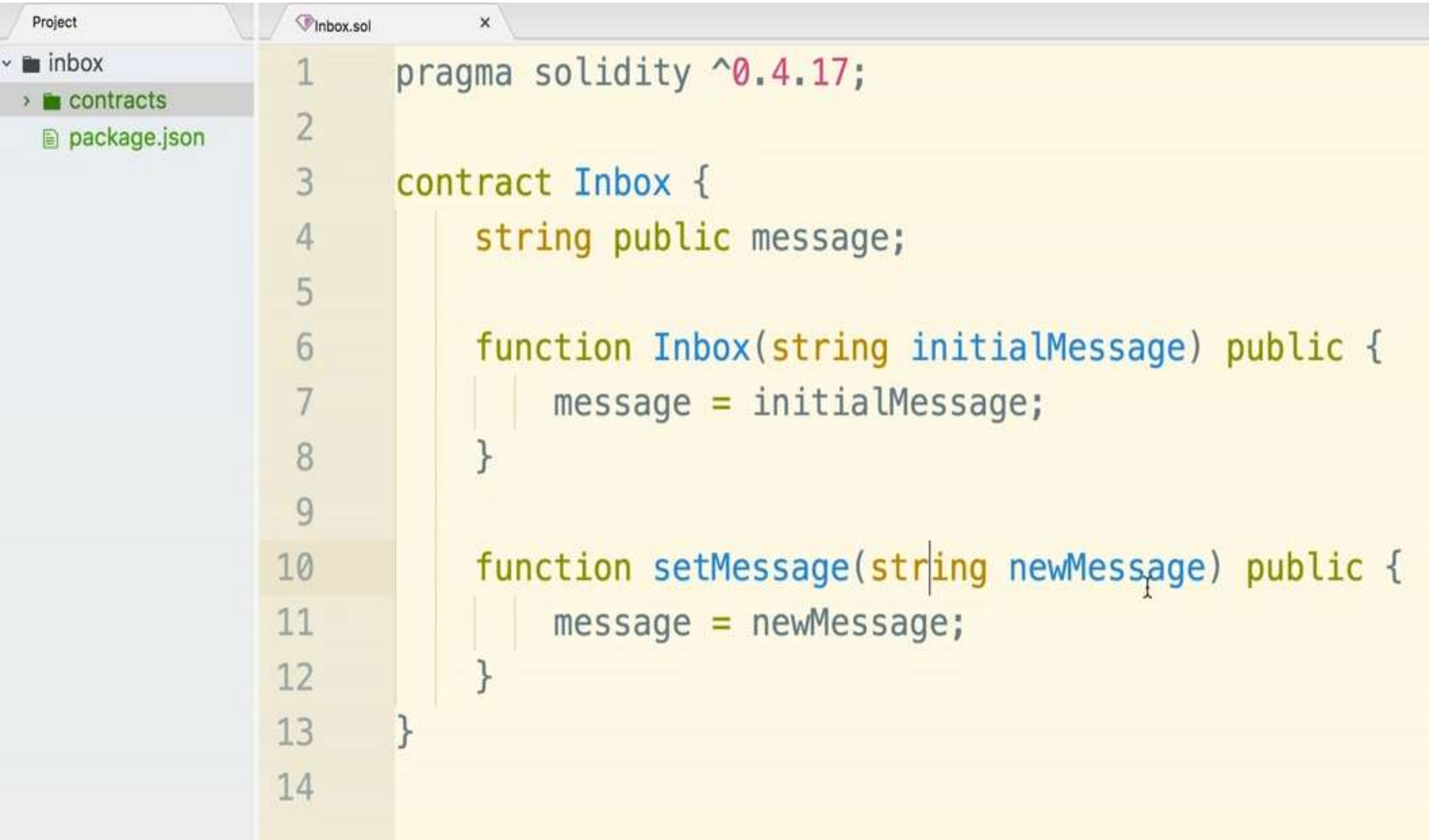
```
mkdir Inbox
cd Inbox
npm init
```

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

Project structure



First file



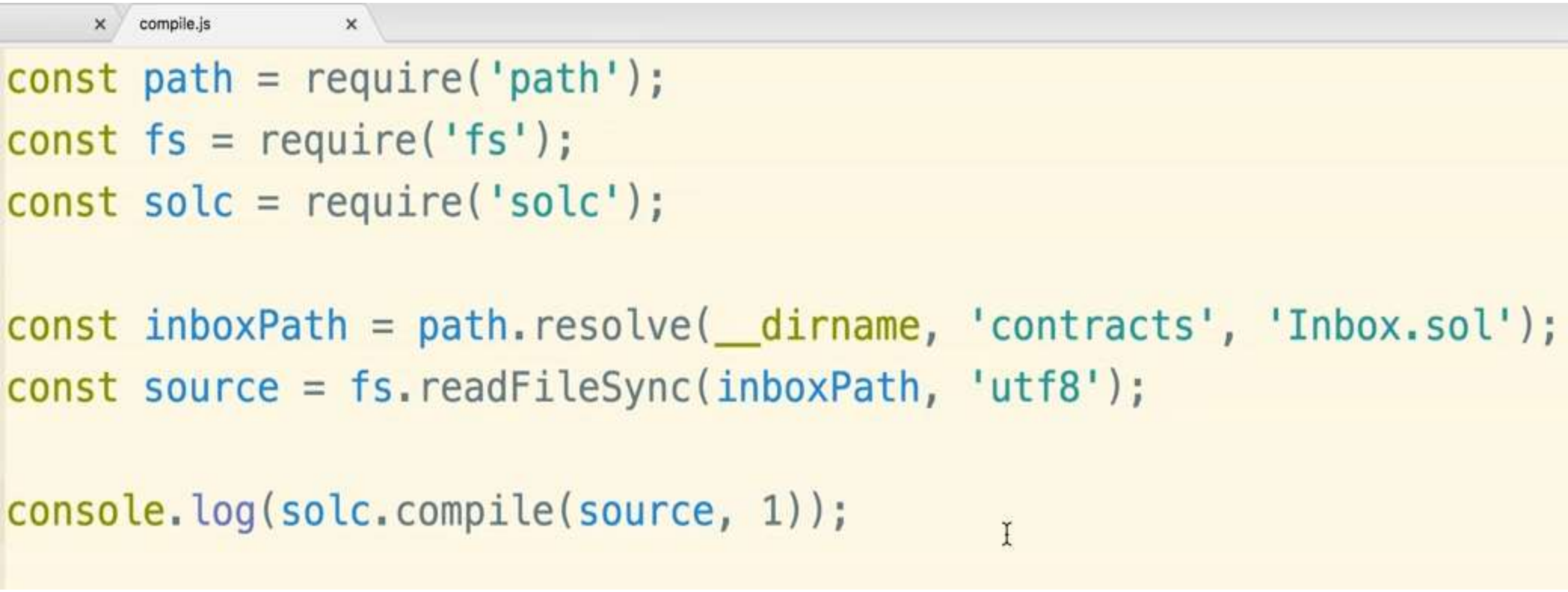
```
pragma solidity ^0.4.17;

contract Inbox {
    string public message;

    function Inbox(string initialMessage) public {
        message = initialMessage;
    }

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

```

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



```
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)

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

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, 'utf-8');
```

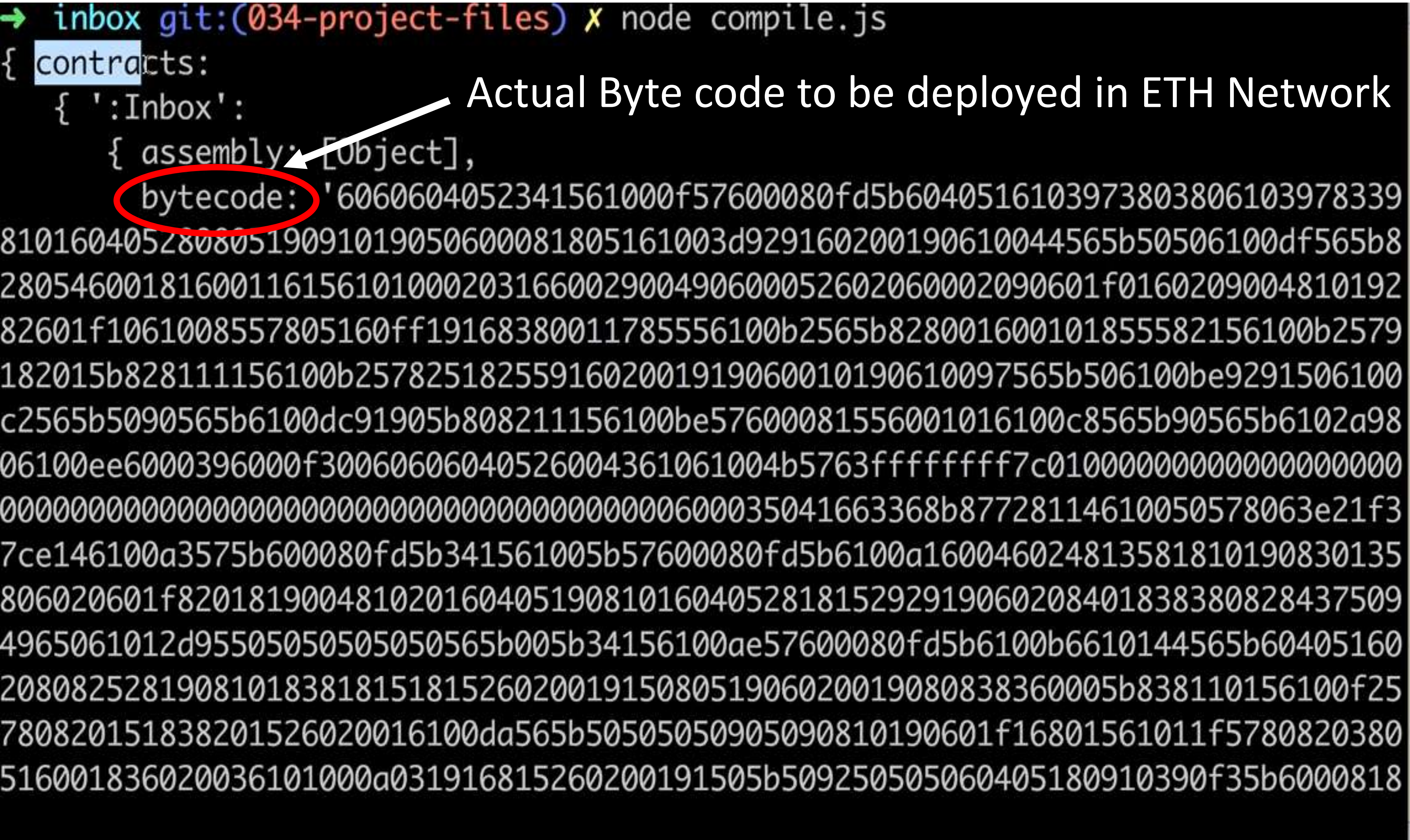
```
var input = {
        language: "Solidity",
        sources: {
            "Inbox.sol": {
                content: source
         } },
        settings: {
            outputSelection: {
                "*": {
                    "*": ["*"]
                } }
        }    };
// parses solidity to English and strings
var output = JSON.parse(solc.compile(JSON.stringify(input)));
var outputContracts = output.contracts['Inbox.sol']['Inbox']
// exports ABI interface
module.exports.abi = outputContracts.abi;
// exports bytecode from smart contract
module.exports.bytecode = outputContracts.evm.bytecode.object;
```

To run compile.js file

$\downarrow$

```
node compile.js
```

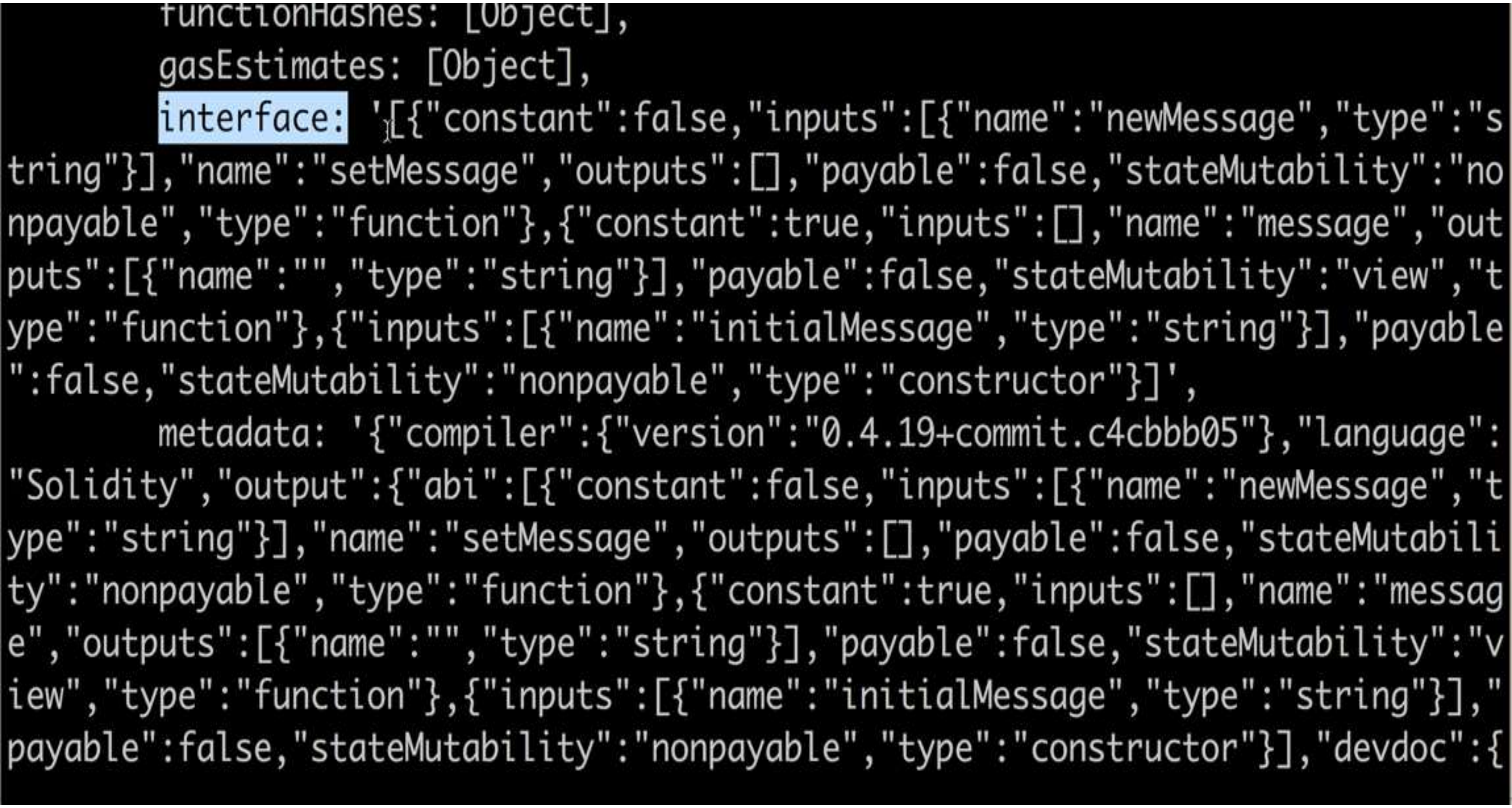
Compile the code



Actual Byte code to be deployed in ETH Network

```
→  inbox git:(034-project-files) ✗ node compile.js
{ contracts:
   { ':Inbox':
      { assembly: [Object],
        bytecode: '6060604052341561000f57600080fd5b6040516103973803806103978339
81016040528080519091019050600081805161003d929160200190610044565b50506100df565b8
2805460018160011615610100020316600290049060005260206000209060 1f0160209004810192
82601f10610085578051 60ff1916838001178555 6100b2565b828001600101855582156100b2579
182015b828111156100b257825182559160200191906001019061009756 5b506100be9291506100
c2565b5090565b6100dc91905b808211156100be5760008155600101 6100c8565b90565b6102a98
06100ee6000396000f30060606040526004361061004b5763ffffffff7c0100000000000000000000
000000000000000000000000000000000000000000600035041663368b877281146100505780 63e21f3
7ce146100a3575b600080fd5b341561005b57600080fd5b6100a160046024813581810190830135
806020601f820181900481020160405190810160405281815292919060208401838380828437509
4965061012d9550505050505050565b005b34156100ae57600080fd5b6100b6610144565b60405160
208082528190810183818151815260200191508051906020019080838360005b838110156100f25
78082015183820152602001 6100da565b50505050905090810190601f168015610 11f5780820380
5160018360200361010000a03191681526020019150 5b509250505060405180910390f35b6000818
```

ABI (Application Binary Interface)

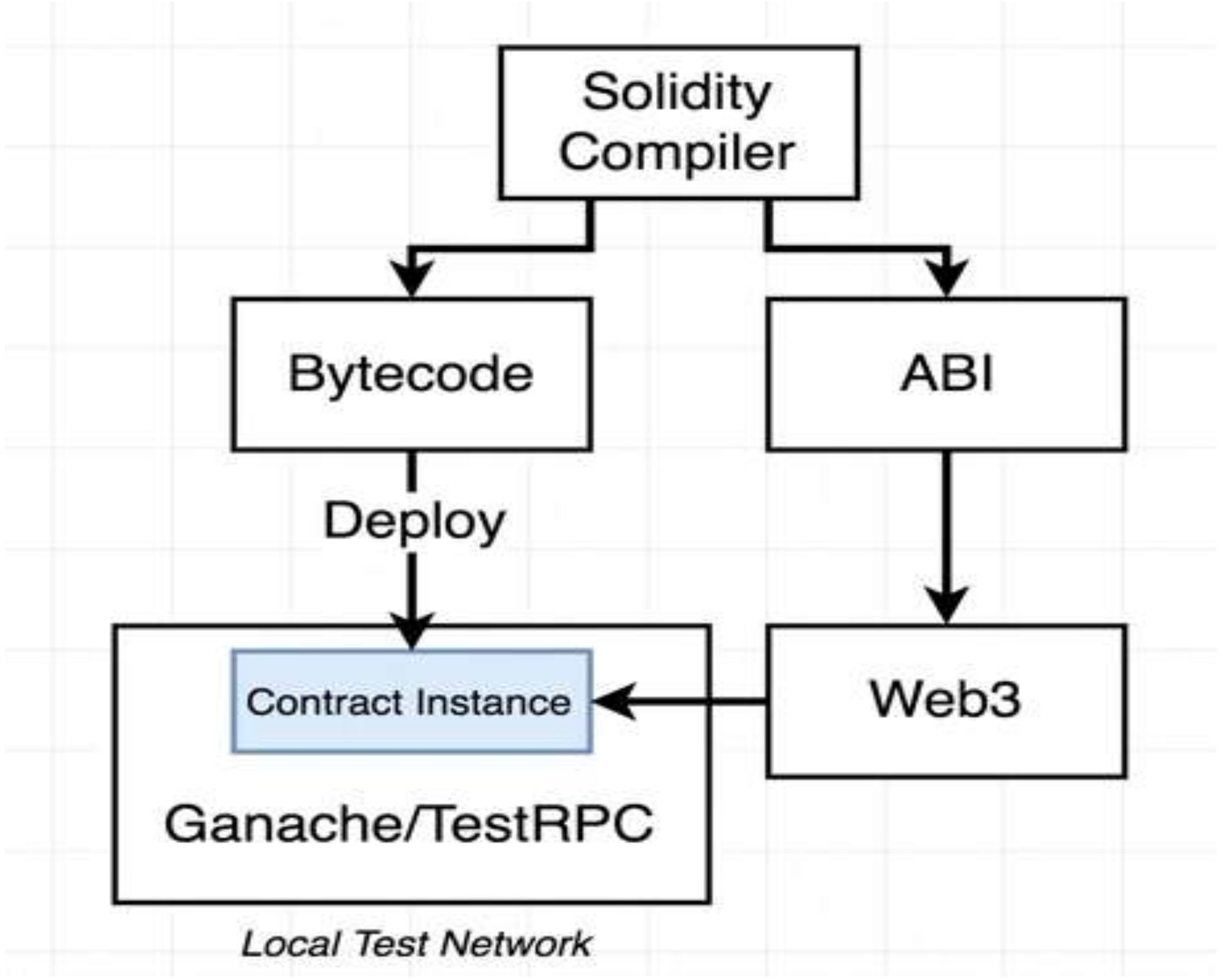


```
        functionHashes: [Object],
        gasEstimates: [Object],
        interface: '[{"constant":false,"inputs":[{"name":"newMessage","type":"s
tring"}],"name":"setMessage","outputs":[],"payable":false,"stateMutability":"no
npayable","type":"function"},{"constant":true,"inputs":[],"name":"message","out
puts":[{"name":"","type":"string"}],"payable":false,"stateMutability":"view","t
ype":"function"},{"inputs":[{"name":"initialMessage","type":"string"}],"payable
":false,"stateMutability":"nonpayable","type":"constructor"}]',
        metadata: '{"compiler":{"version":"0.4.19+commit.c4cbbb05"},"language":
"Solidity","output":{"abi":[{"constant":false,"inputs":[{"name":"newMessage","t
ype":"string"}],"name":"setMessage","outputs":[],"payable":false,"stateMutabili
ty":"nonpayable","type":"function"},{"constant":true,"inputs":[],"name":"messag
e","outputs":[{"name":"","type":"string"}],"payable":false,"stateMutability":"v
iew","type":"function"},{"inputs":[{"name":"initialMessage","type":"string"}],"
payable":false,"stateMutability":"nonpayable","type":"constructor"}],"devdoc":{
```

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 **<u>inbox.test.js</u>** file in that folder.

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

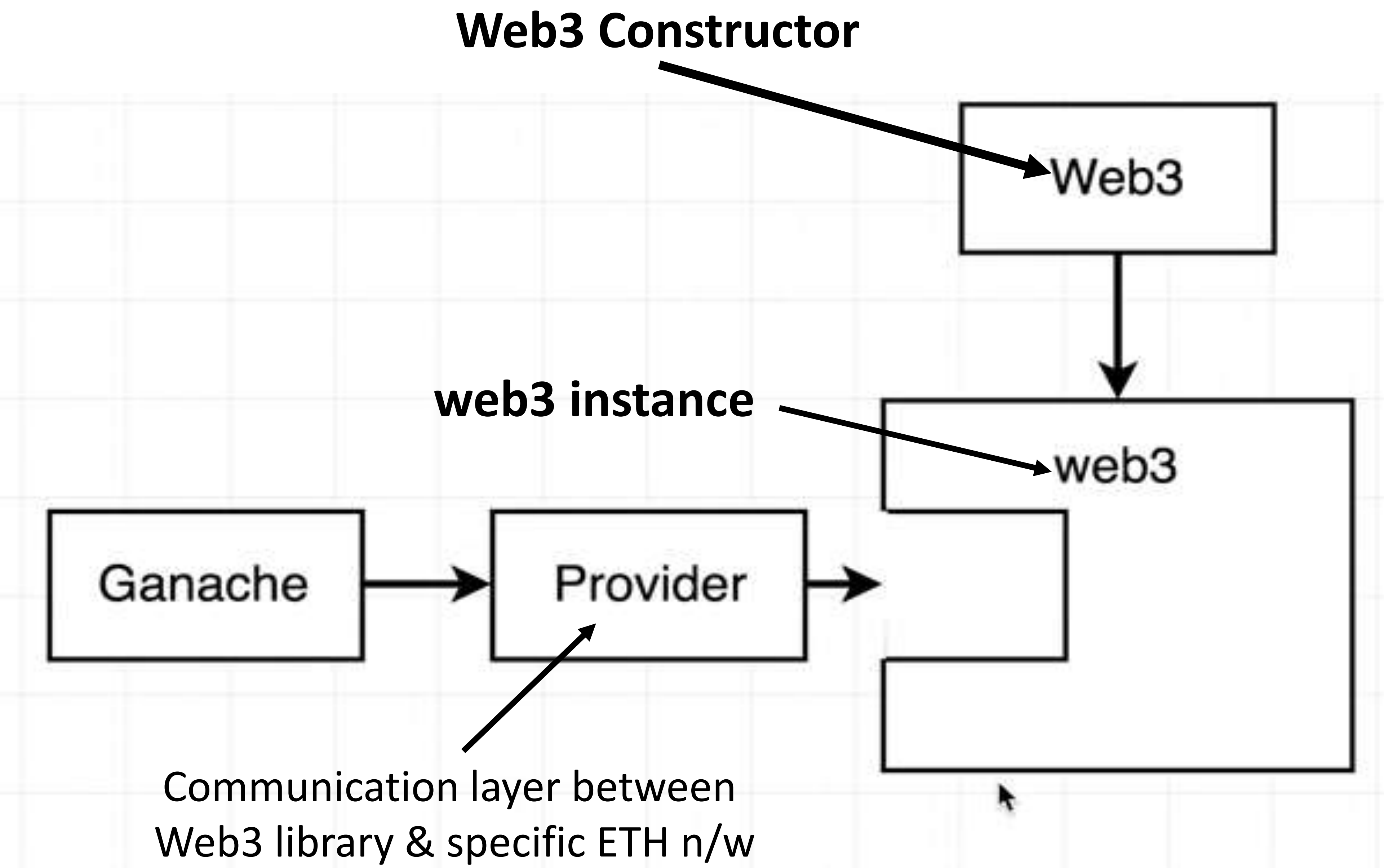
const ganache = require('ganache-cli');

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

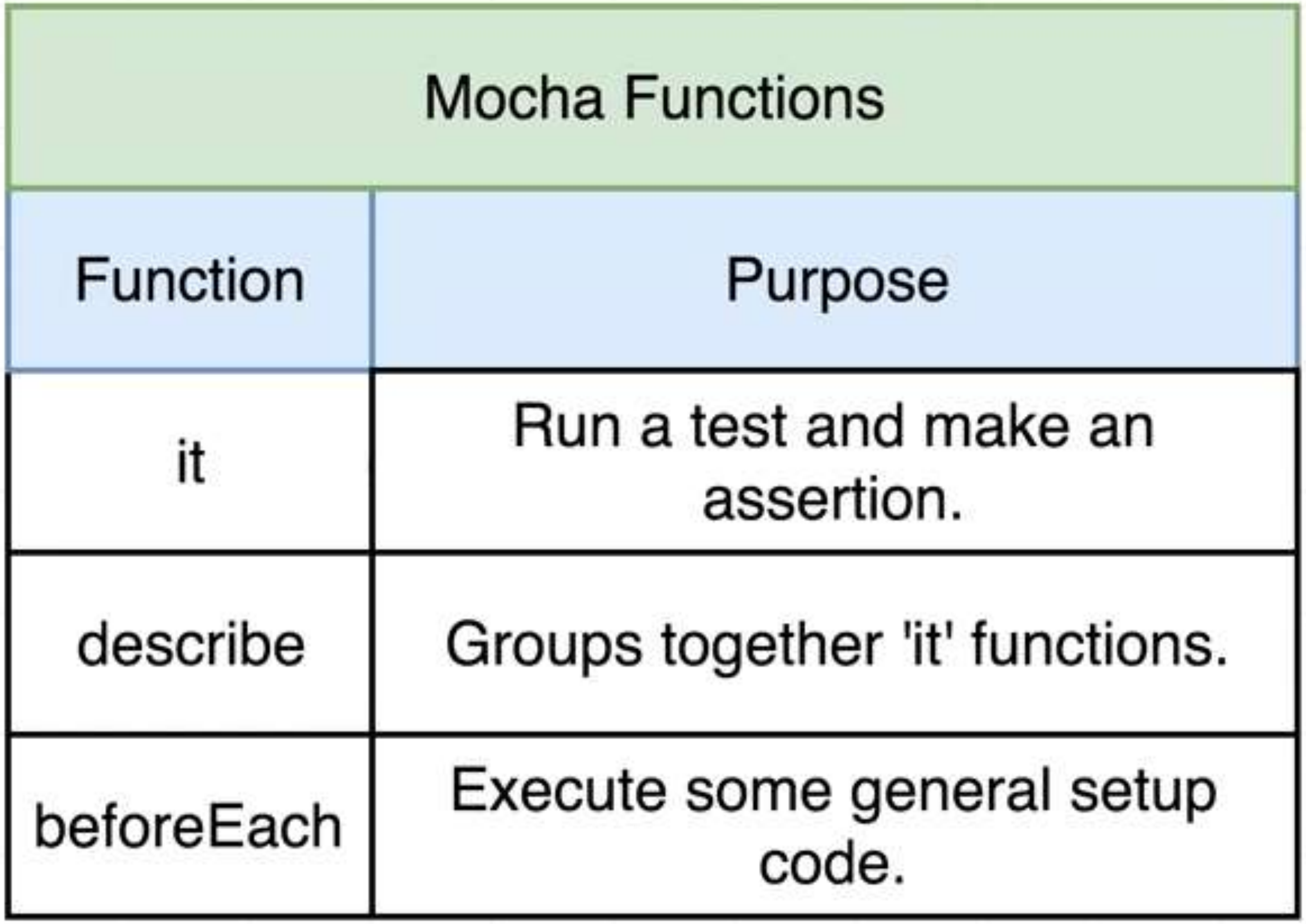
const web3 = new Web3(ganache.provider());

//web3 → instance of Web3

Web3 Providers

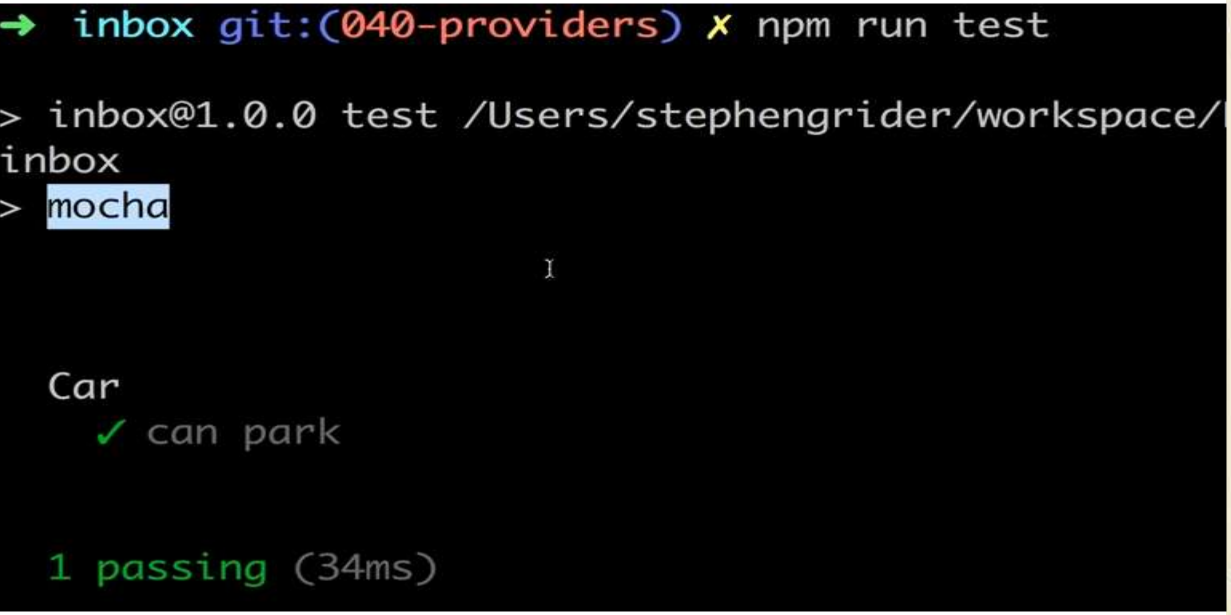


Mocha

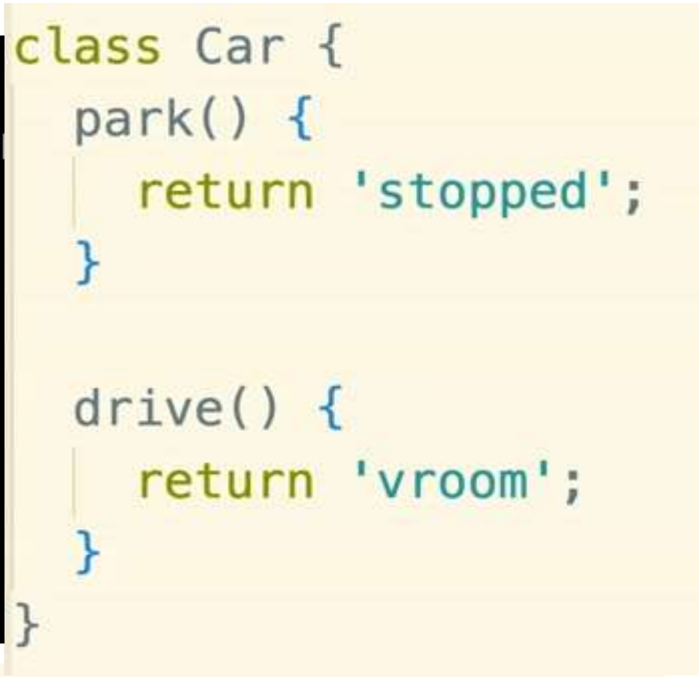


Mocha Functions	
Function	Purpose
it	Run a test and make an assertion.
describe	Groups together 'it' functions.
beforeEach	Execute some general setup code.

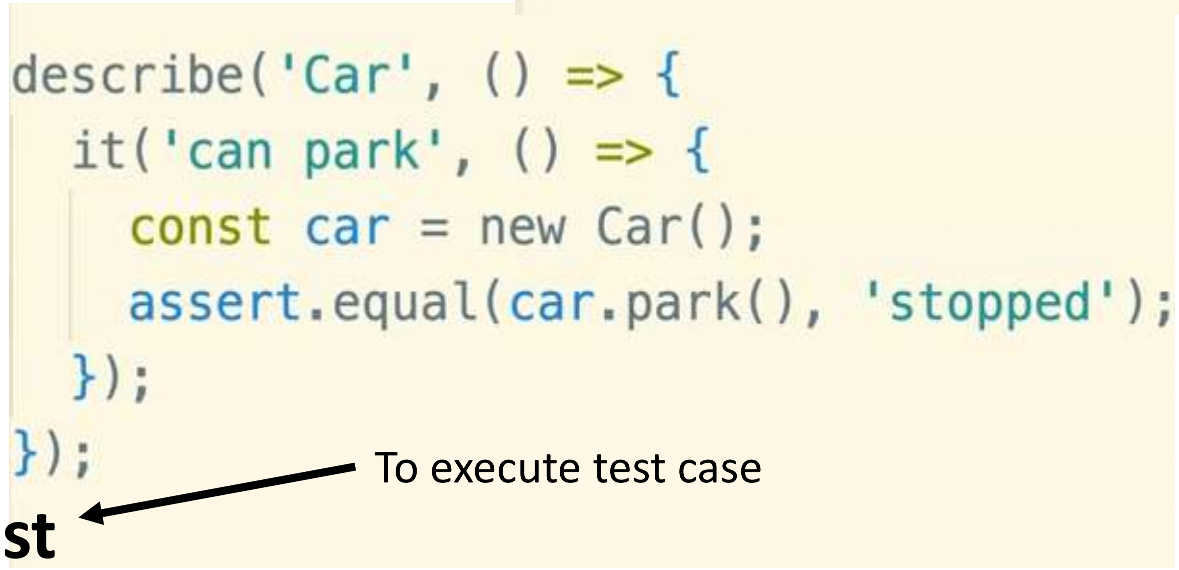
Inbox.test.js



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

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



  Car
    ✓ can park


  1 passing (34ms)
```

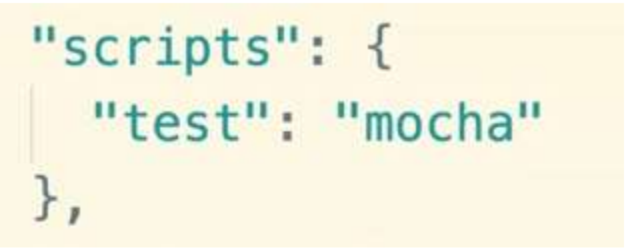


```
class Car {
  park() {
    return 'stopped';
  }

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



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

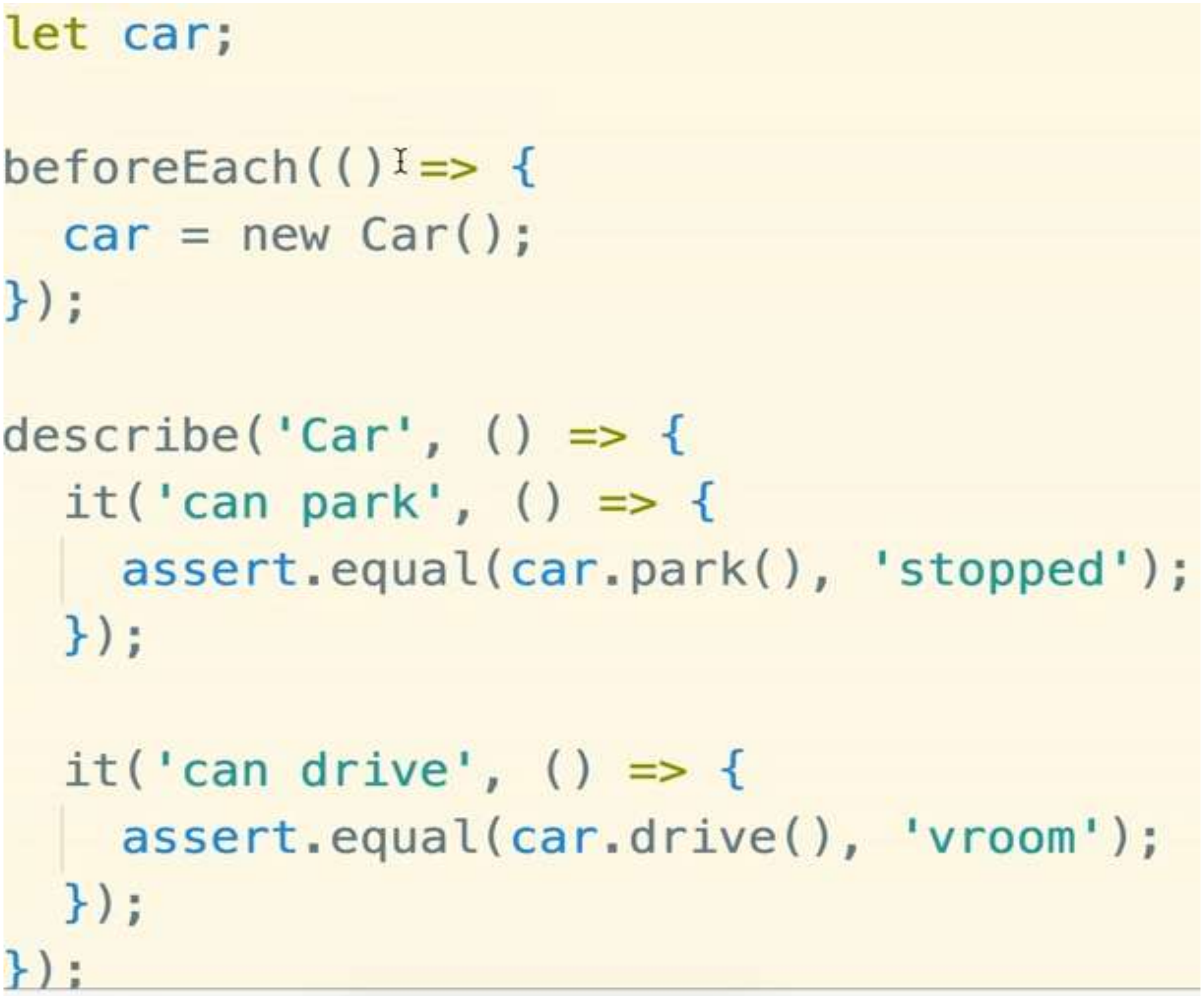
In package.json



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

npm run test ← To execute test case

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');
  });
});
```

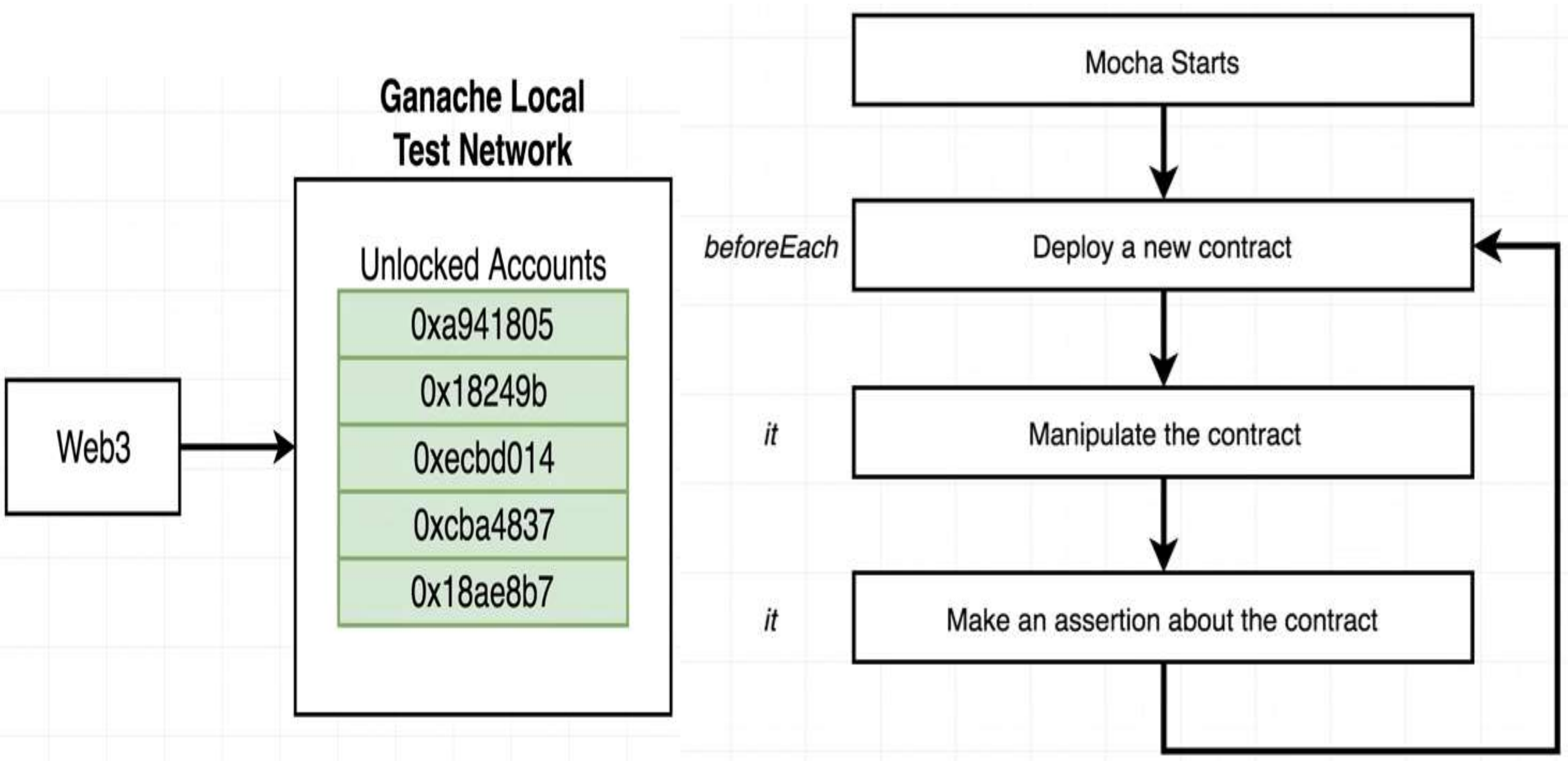
```
const assert = require('assert');
const ganache = require ('ganache-cli')
const Web3 = require ('web3')
const web3 = new Web3(ganache.provider());

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

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

npm run test ← To execute test case

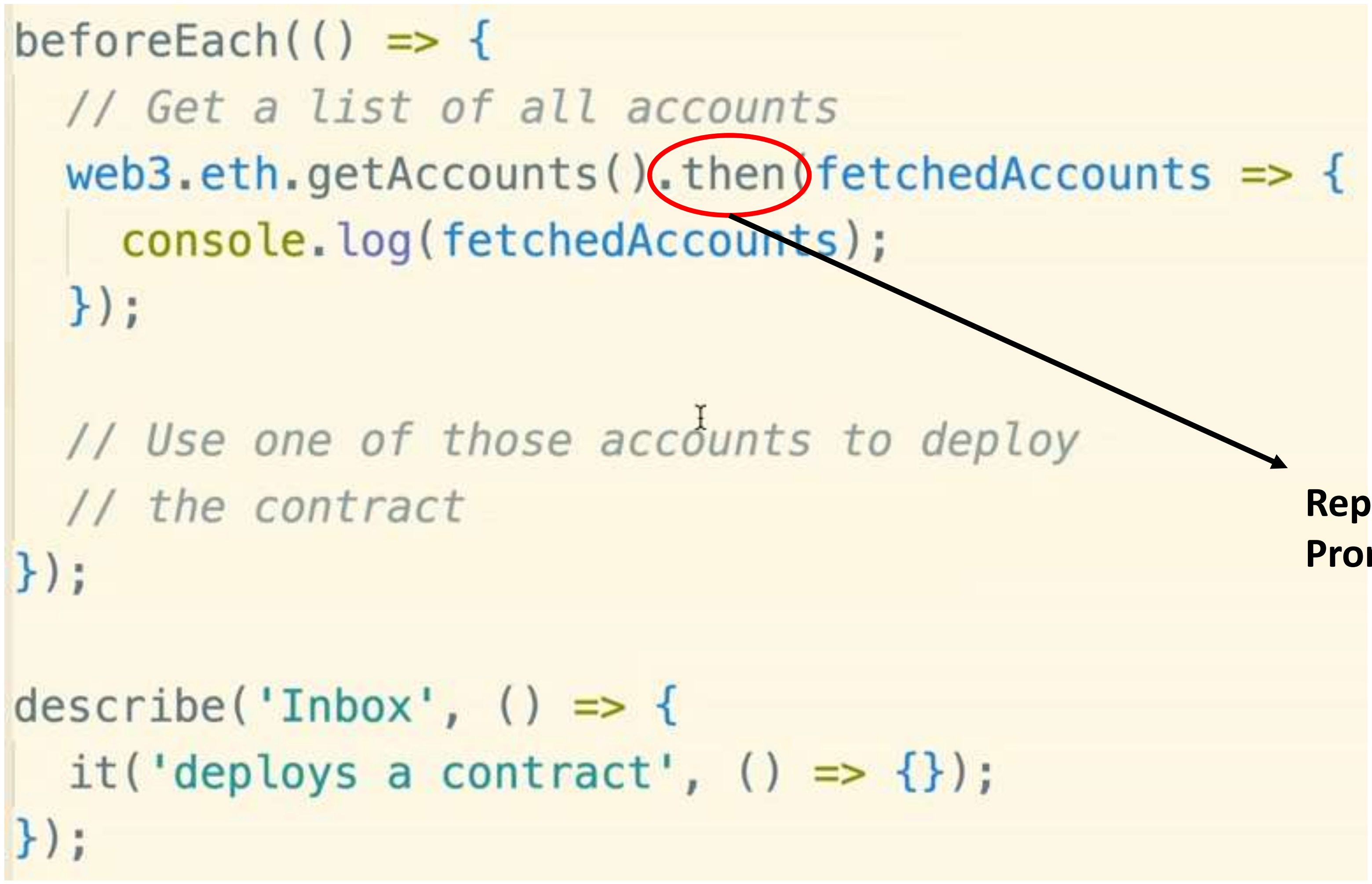
```
/* let car;

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');
    });
});
```

npm run test $\leftarrow$ To execute test case

Mocha for contracts



Deploying contract (using Promises)

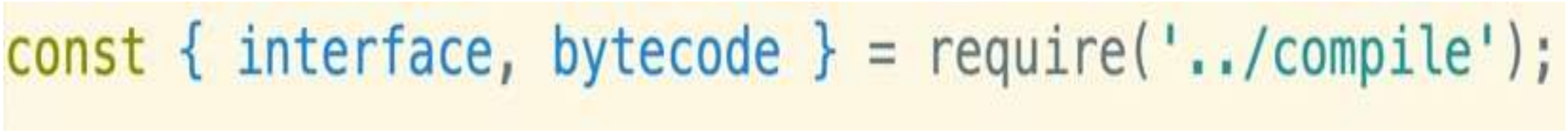


```
beforeEach(() => {
  // Get a list of all accounts
  web3.eth.getAccounts().then(fetchedAccounts => {
    console.log(fetchedAccounts);
  });

  // Use one of those accounts to deploy
  // the contract
});

describe('Inbox', () => {
  it('deploys a contract', () => {});
});
```

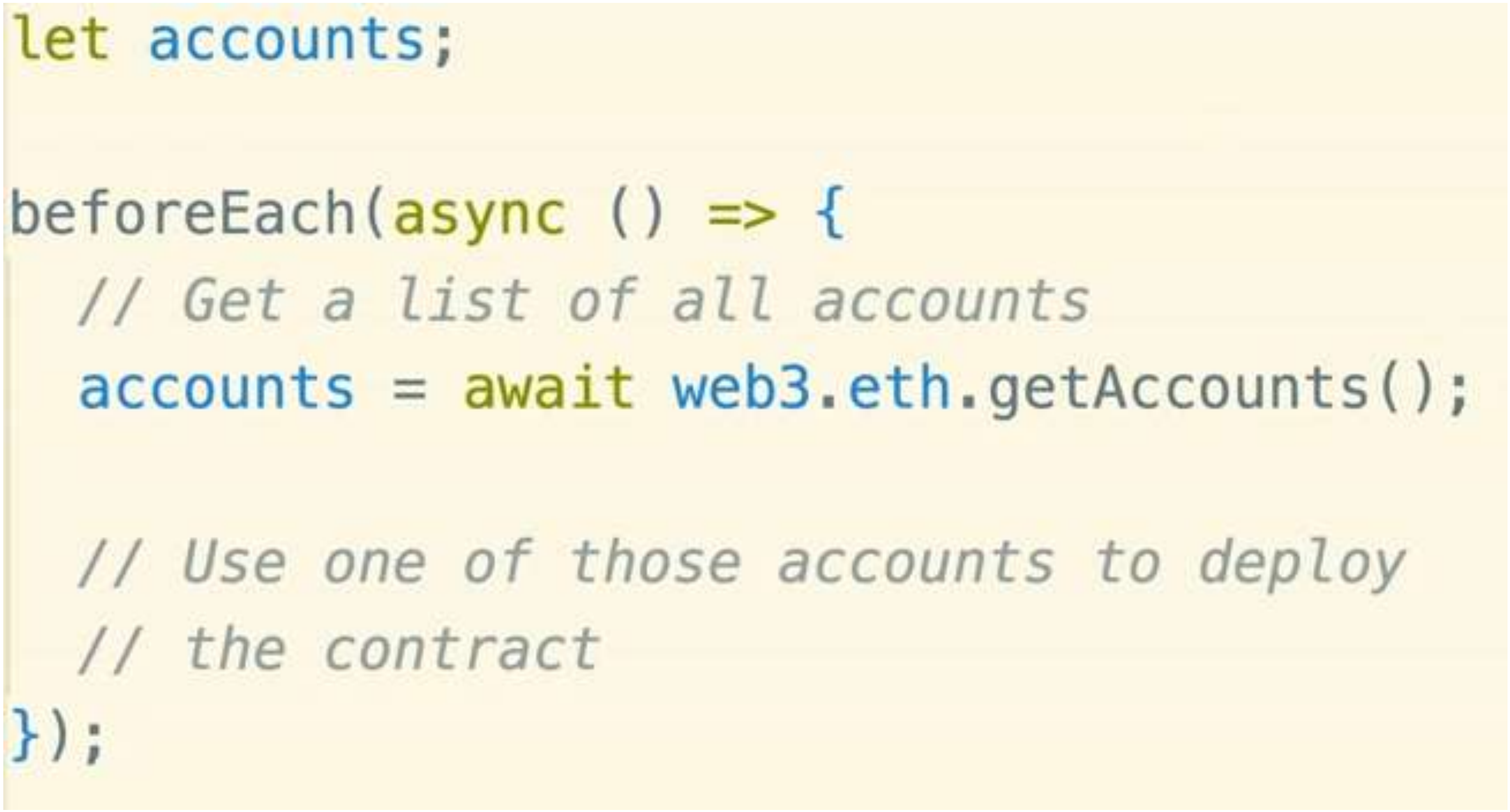
Represents Promise

```
beforeEach(() => {
    //Get a list of accounts
    web3.eth.getAccounts()
     .then(fetchedAccounts => {
        console.log(fetchedAccounts);
    });

    //Use one of those accounts to deploy the
contract
});

describe('Inbox', () => {
    it('deploys a contract', () => {});
});
```

Callback vs promise vs async await

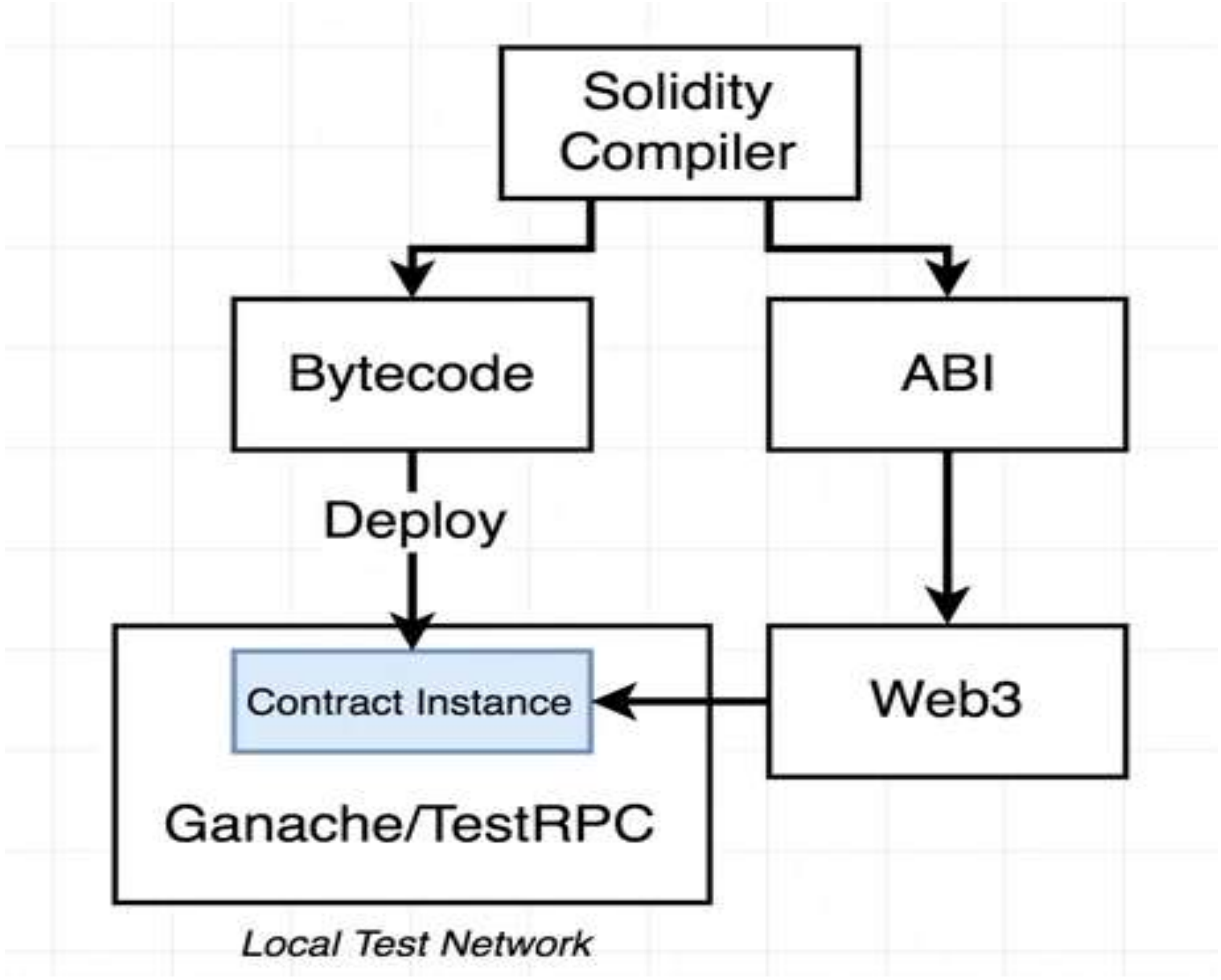


```
const { interface, bytecode } = require('../compile');
```



```
let accounts;

beforeEach(async () => {
  // Get a list of all accounts
  accounts = await web3.eth.getAccounts();

  // Use one of those accounts to deploy
  // the contract
});
```

Testing setup



Test contract (Prior to deploy)

Inbox.test.js

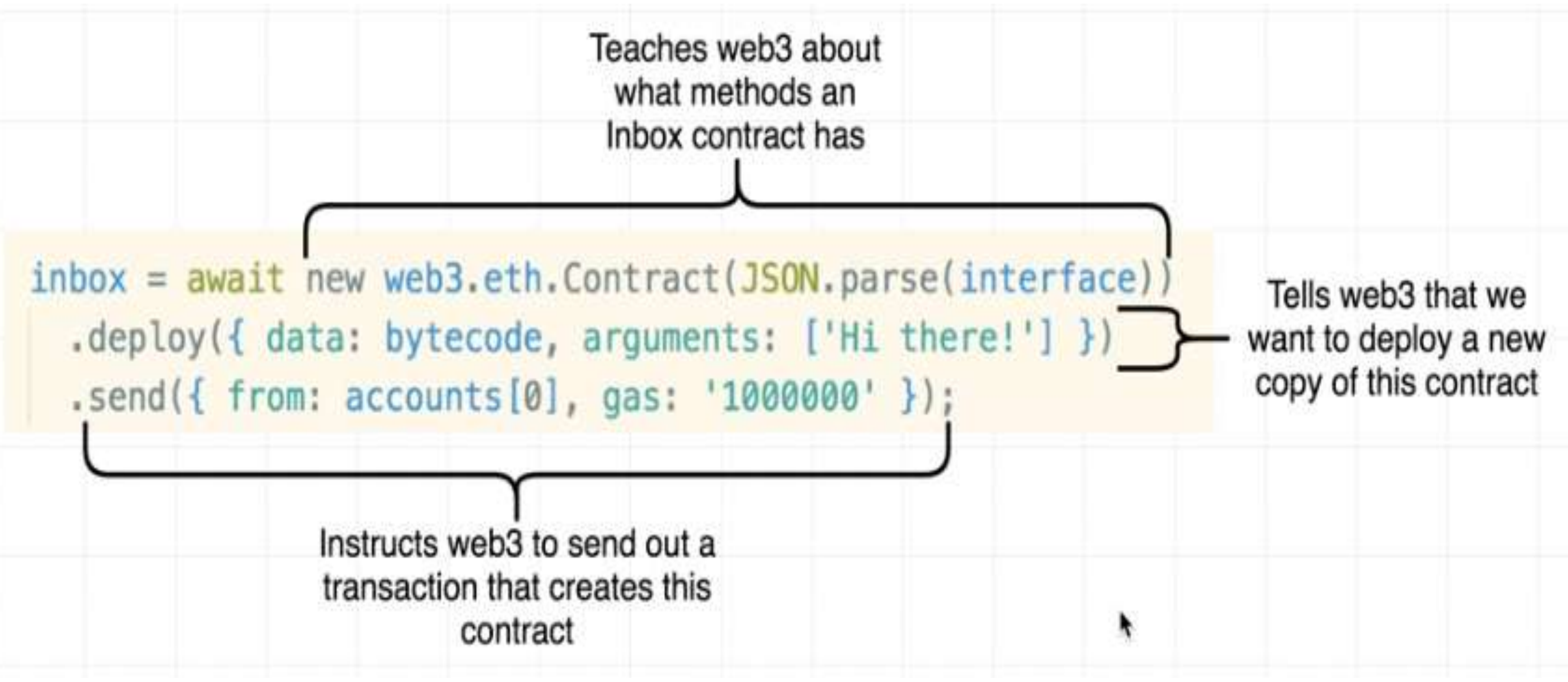
```
const assert = require('assert');
const ganache = require ('ganache-cli');

const Web3 = require ('web3');
const web3 = new Web3(ganache.provider());

const {abi, bytecode} = require('../compile’);

let accounts;
let inbox;
```

Test contract (Prior to deploy)

Inbox.test.js

```
beforeEach(async () => {
    //Get a list of accounts
    accounts = await web3.eth.getAccounts();

    //Use one of those accounts to deploy the contract
    inbox = await new web3.eth.Contract((abi))
       .deploy({data: bytecode,
                arguments: ['Hi there!'] })
       .send({from: accounts[0], gas: '1000000'});
});

describe('Inbox', () => {
    it('deploys a contract', () => {
    //console.log(accounts);
    console.log(inbox);
    });
});
```

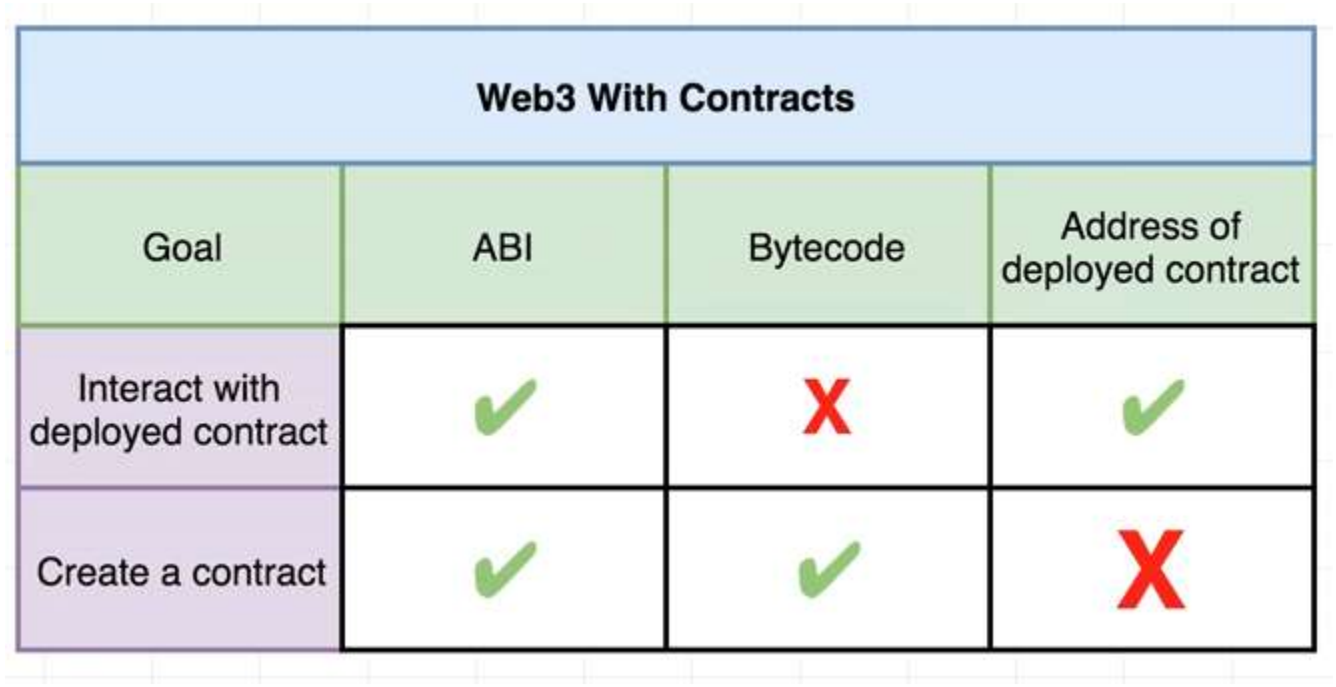
Deploy contract



```
inbox = await new web3.eth.Contract(JSON.parse(interface))
  .deploy({ data: bytecode, arguments: ['Hi there!'] })
  .send({ from: accounts[0], gas: '1000000' });
```

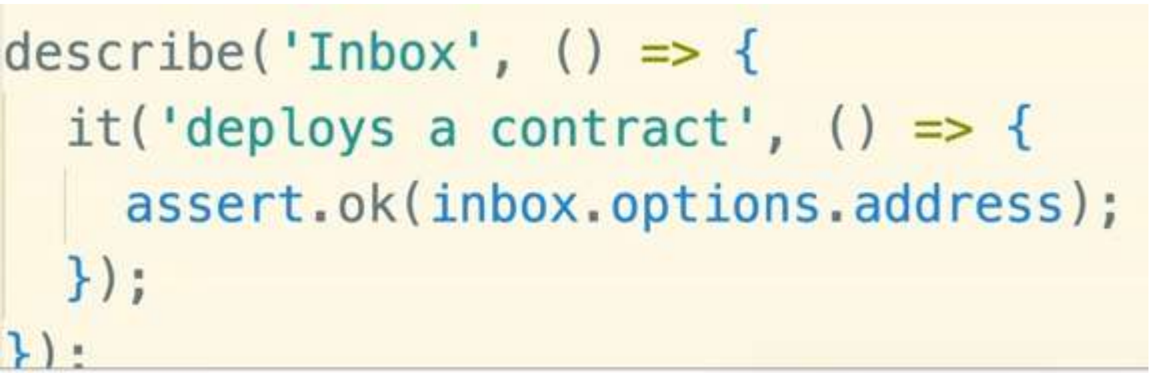
Teaches web3 about what methods an Inbox contract has
Tells web3 that we want to deploy a new copy of this contract
Instructs web3 to send out a transaction that creates this contract

Web3 with contracts

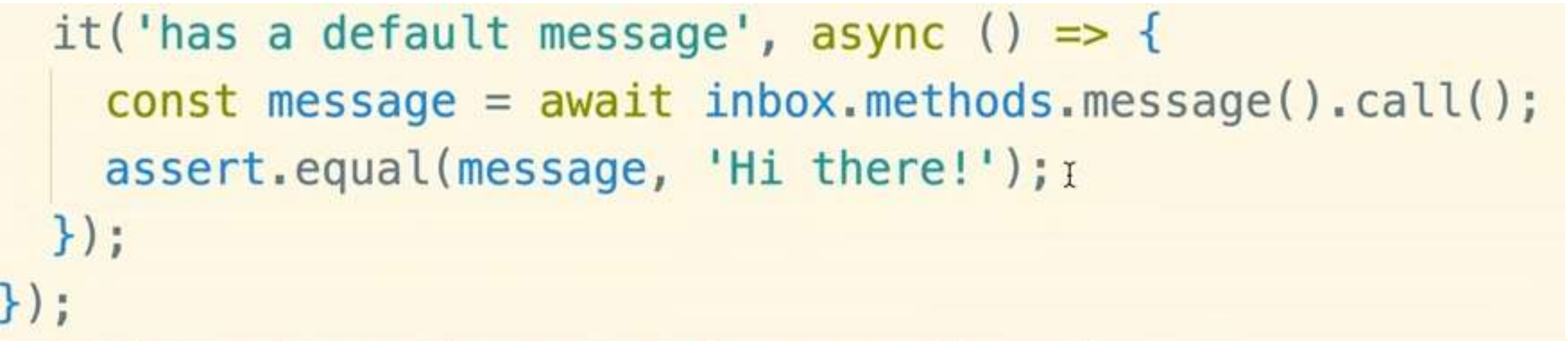


Web3 With Contracts			
Goal	ABI	Bytecode	Address of deployed contract
Interact with deployed contract	✓	X	✓
Create a contract	✓	✓	X

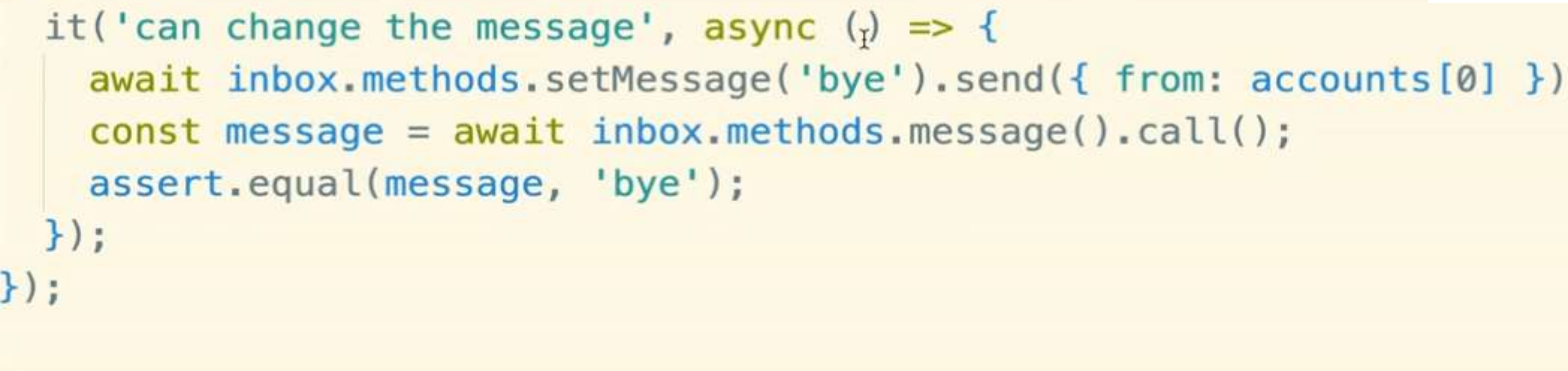
Actual Tests on Inbox



```
describe('Inbox', () => {
  it('deploys a contract', () => {
    assert.ok(inbox.options.address);
  });
});
```



```
  it('has a default message', async () => {
    const message = await inbox.methods.message().call();
    assert.equal(message, 'Hi there!');
  });
});
```

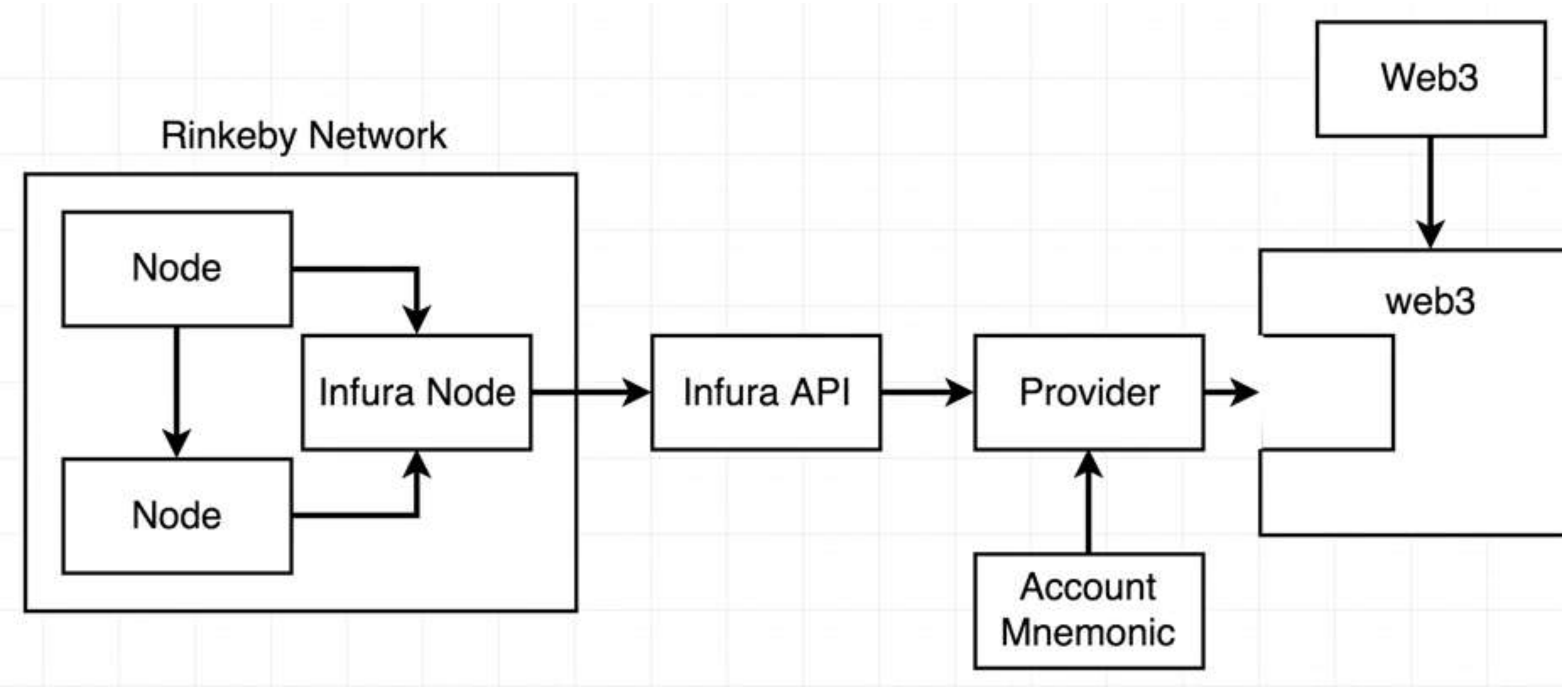


```
  it('can change the message', async () => {
    await inbox.methods.setMessage('bye').send({ from: accounts[0] })
    const message = await inbox.methods.message().call();
    assert.equal(message, 'bye');
  });
});
```

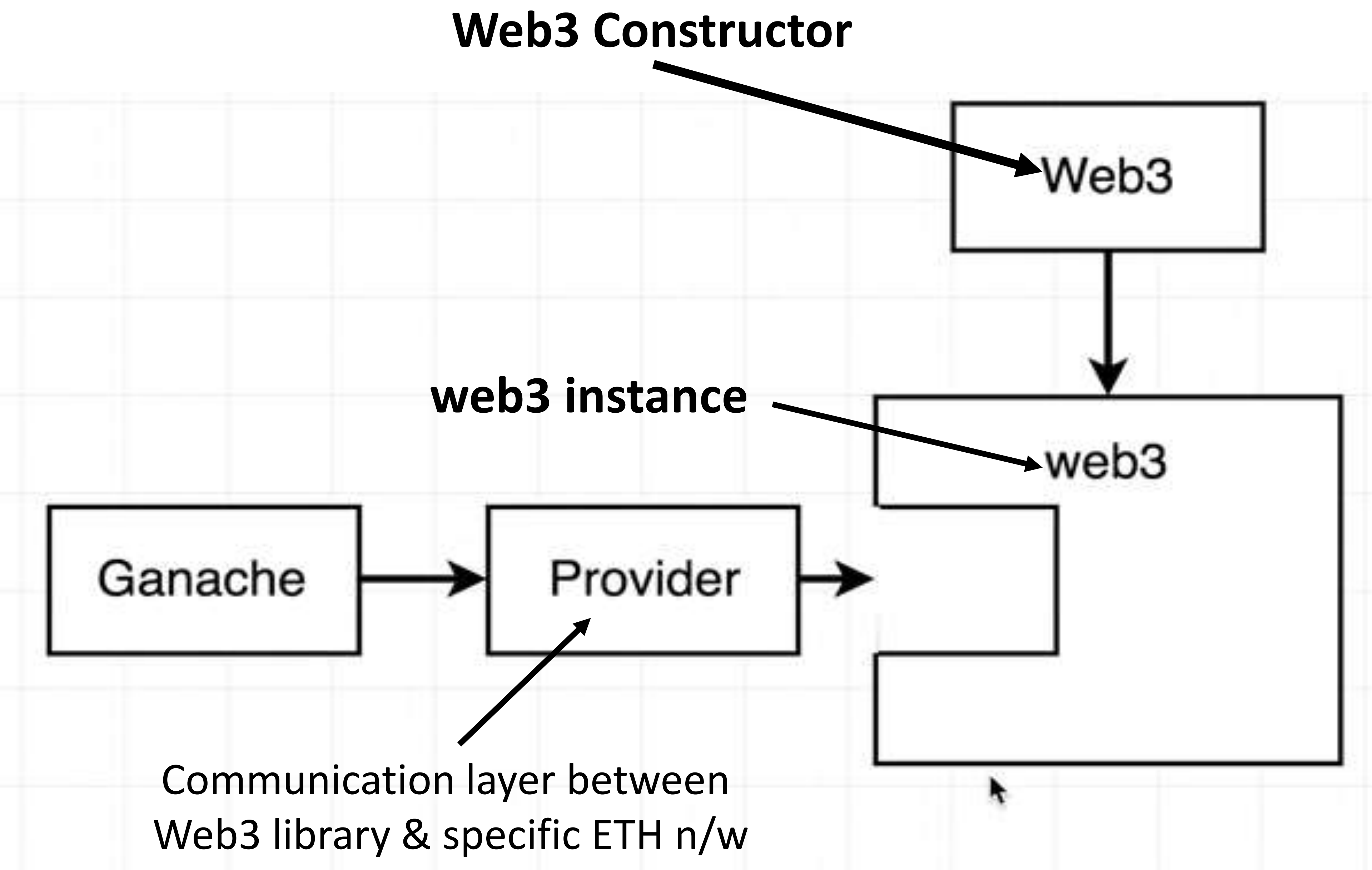
Deploying to real network

- **Till now we were simply using the already created accounts by ganache.**
- **These account were open and had ethers as well.**

- **Check for sufficient ethers**
- **The provider must have an account with ethers for deployment purpose.**



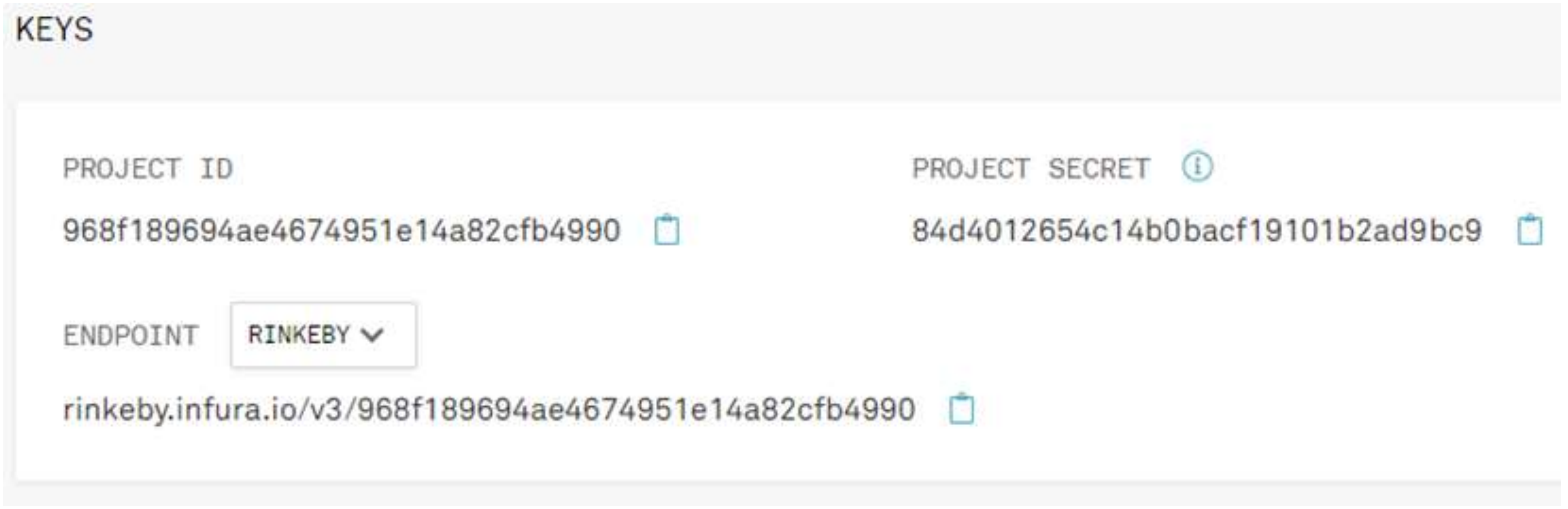
Web3 Providers



Infura.io

http://infura.io/

https://rinkeby.infura.io/v3/ee229d8330b643599f7129b8761ba865
wss://rinkeby.infura.io/ws/v3/ee229d8330b643599f7129b8761ba865



KEYS

PROJECT ID
968f189694ae4674951e14a82cfb4990

PROJECT SECRET
84d4012654c14b0bacf19101b2ad9bc9

ENDPOINT RINKEBY

rinkeby.infura.io/v3/968f189694ae4674951e14a82cfb4990

```
npm install  - - save truffle-hdwallet-provider
```

Deploy.js

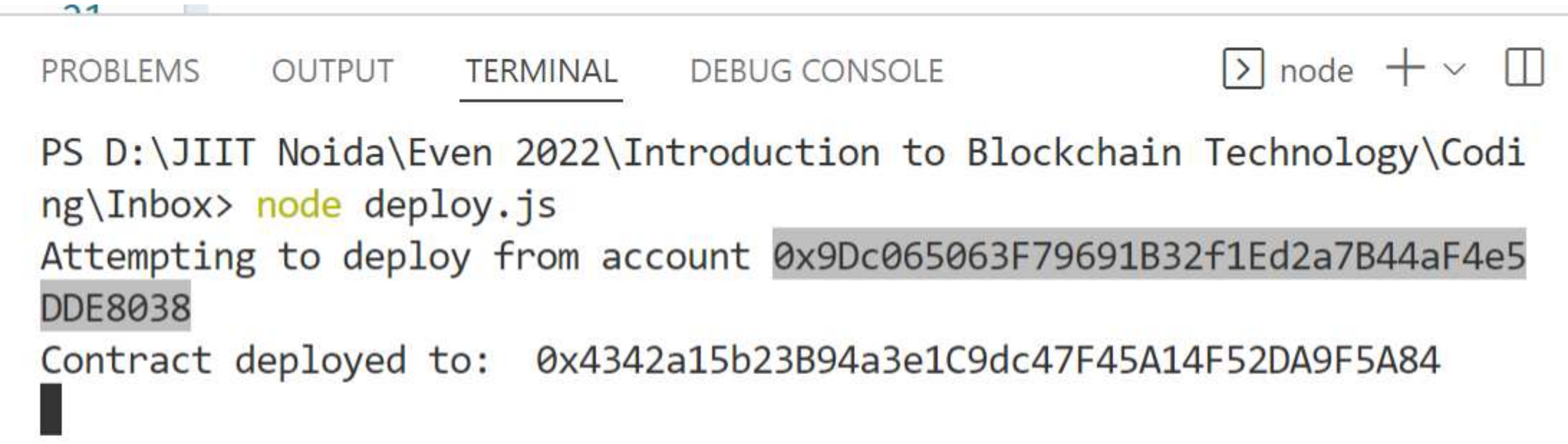
```
const HDWalletProvider = require('truffle-
hdwallet-provider');
const Web3 = require('web3');
const {abi, bytecode} = require('./compile');

const provider = new HDWalletProvider(
    'witness daughter carry valve snake room
hat such couple taste dutch panther',
    'https://rinkeby.infura.io/v3/ee229d8330b
643599f7129b8761ba865’
);

const web3 = new Web3(provider);
```

Deploy.js (new web3 instance to deploy contract on rinkeby)

```
const deploy = async() => {
    const accounts = await web3.eth.getAccounts();

    console.log('Attempting to deploy from account', accounts[0]);

  //Use one of those accounts to deploy the contract
  const result = await new web3.eth.Contract((abi))
  .deploy({data: bytecode, arguments: ['Hi there!'] })
  .send({from: accounts[0], gas: '1000000'});

  console.log('Contract deployed to: ‘, result.options.address);
};

deploy();
```

node deploy.js

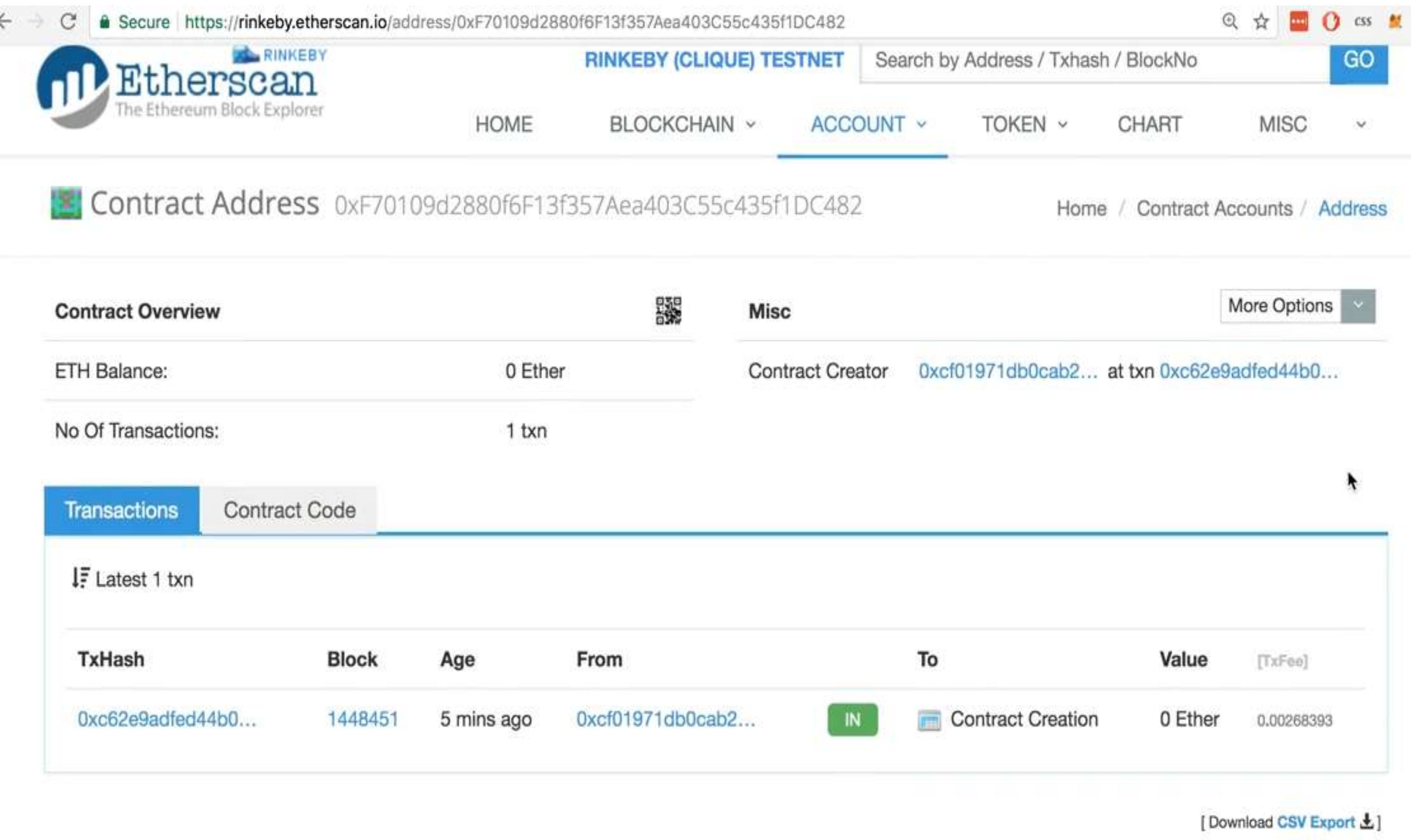
Get the address where contract was deployed



```
PS D:\JIIT Noida\Even 2022\Introduction to Blockchain Technology\Codi
ng\Inbox> node deploy.js
Attempting to deploy from account 0x9Dc065063F79691B32f1Ed2a7B44aF4e5
DDE8038
Contract deployed to:  0x4342a15b23B94a3e1C9dc47F45A14F52DA9F5A84
```

```
D:\JIIT Noida\Even 2022\Introduction to Blockchain Technology\Coding\Inbox> node deploy.js
Attempting to deploy from account 0x9Dc065063F79691B32f1Ed2a7B44aF4e5DDE8038DDE8038
Contract deployed to:  0x4342a15b23B94a3e1C9dc47F45A14F52DA9F5A84
```

Rinkeby.etherscan.io



Secure https://rinkeby.etherscan.io/address/0xF70109d2880f6F13f357Aea403C55c435f1DC482
RINKEBY Etherscan The Ethereum Block Explorer
RINKEBY (CLIQUE) TESTNET
Search by Address / Txhash / BlockNo
GO
HOME
BLOCKCHAIN
ACCOUNT
TOKEN
CHART
MISC
Home / Contract Accounts / Address
More Options
Transactions
Contract Code
[Download CSV Export]

Contract Address 0xF70109d2880f6F13f357Aea403C55c435f1DC482

Contract Overview

ETH Balance:	0 Ether
No Of Transactions:	1 txn

Misc

Contract Creator 0xcf01971db0cab2... at txn 0xc62e9adfed44b0...

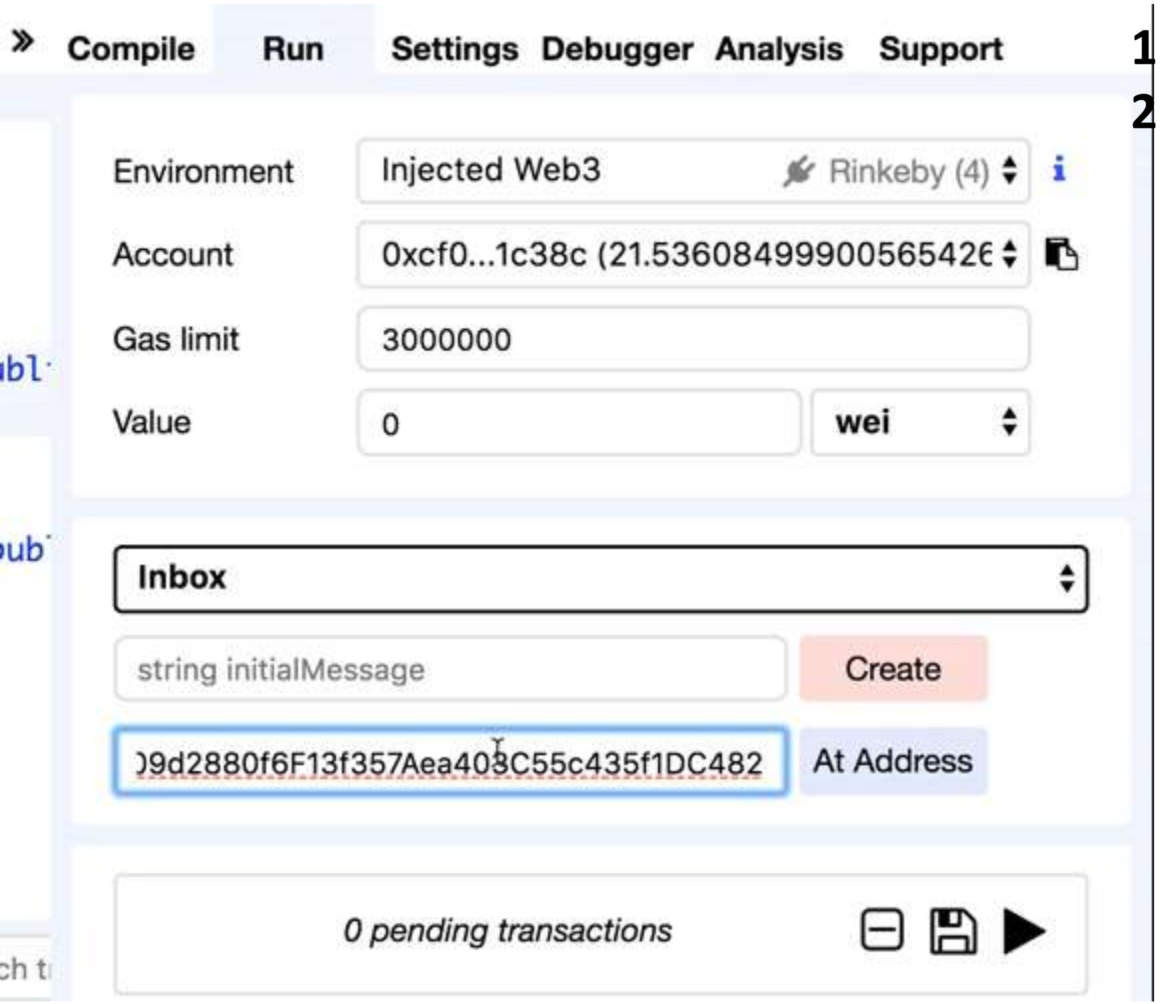
Latest 1 txn

TxHash	Block	Age	From		To	Value	[TxFee]
0xc62e9adfed44b0...	1448451	5 mins ago	0xcf01971db0cab2...	IN	Contract Creation	0 Ether	0.00268393

Remix Vs Code Editor

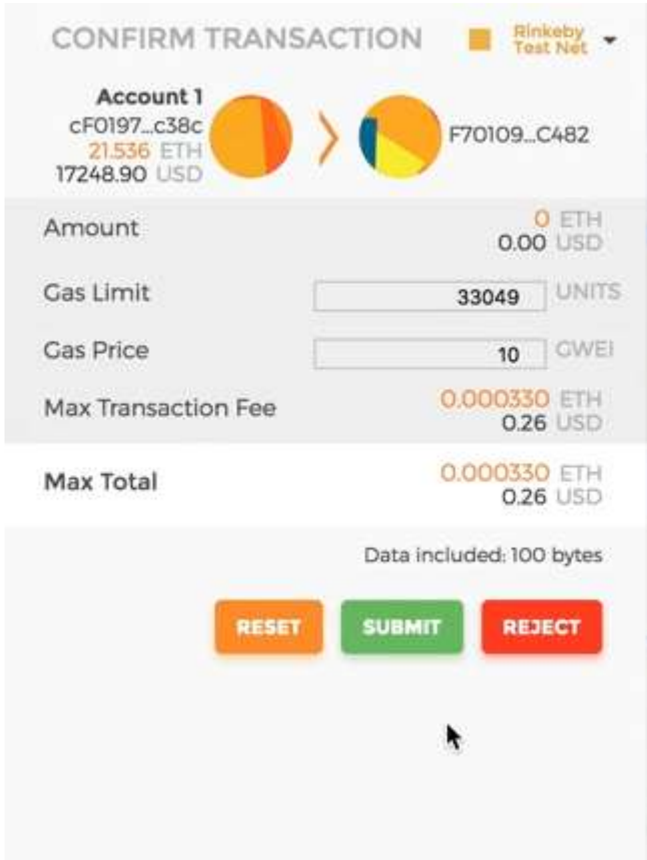
- Remix:
  - Easy to write the code and test on local network
  - Great tool for beginners
- Code Editor:
  - Interact with other front end applications
  - Compile multiple .sol files at same time
  - Support from GIT for version control and other queries
  - Ganache may be used for ready-made accounts
  - Different ‘it’ statements can be used to test cases

Assignment: Interacting with contracts deployed on rinkeby



Environment: Injected Web3 Rinkeby (4)
Account: 0xcf0...1c38c (21.5360849990056542
Gas limit: 3000000
Value: 0 wei
Inbox
string initialMessage
)9d2880f6F13f357Aea403C55c435f1DC482
0 pending transactions

1. **Interact with deployed contract**
2. **Deploy a new contract**



Account 1
cF0197...c38c
21.536 ETH
17248.90 USD
F70109...C482

Amount	0 ETH 0.00 USD
Gas Limit	33049 UNITS
Gas Price	10 GWEI
Max Transaction Fee	0.000330 ETH 0.26 USD
Max Total	0.000330 ETH 0.26 USD

Data included: 100 bytes

Thank You!