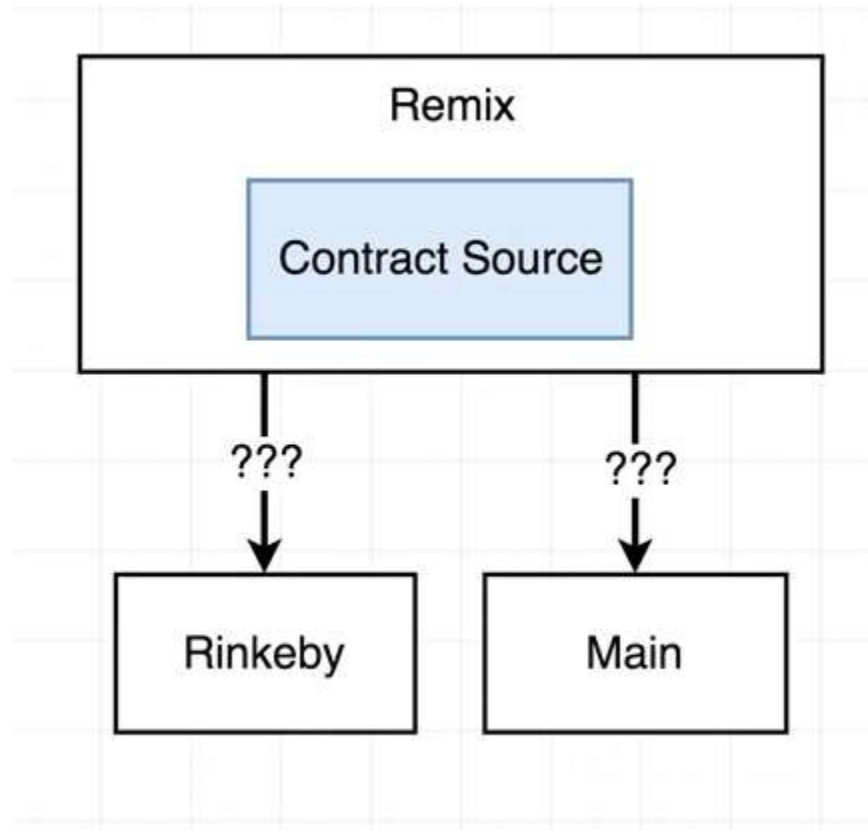


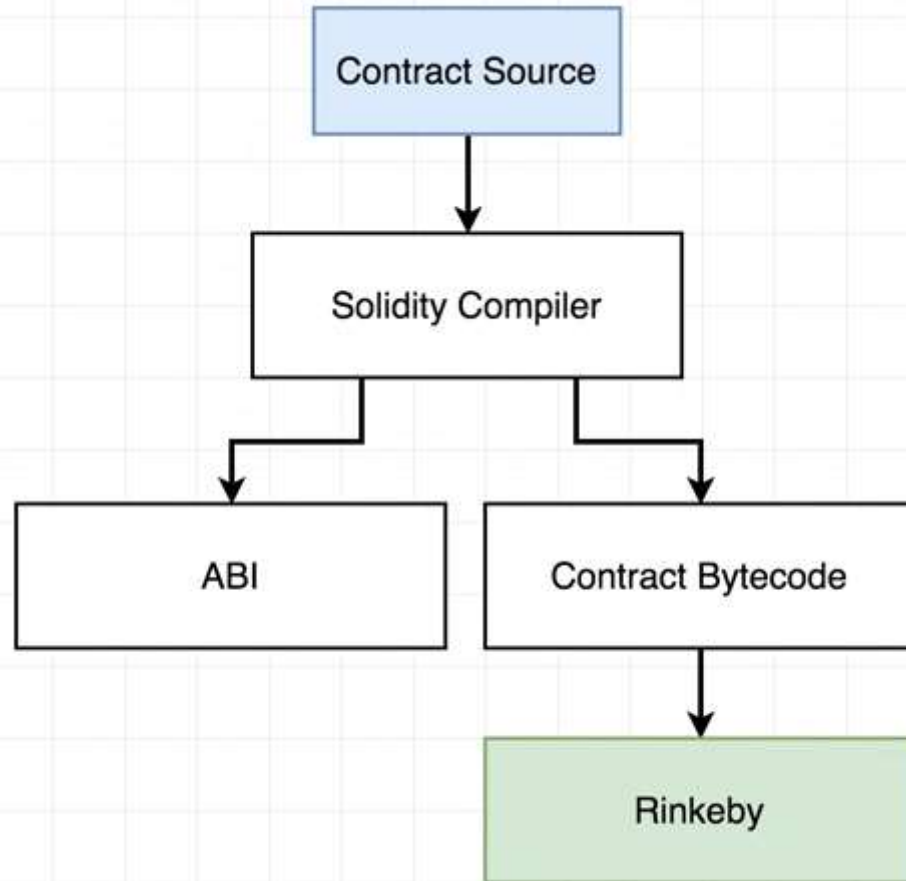


ethereum

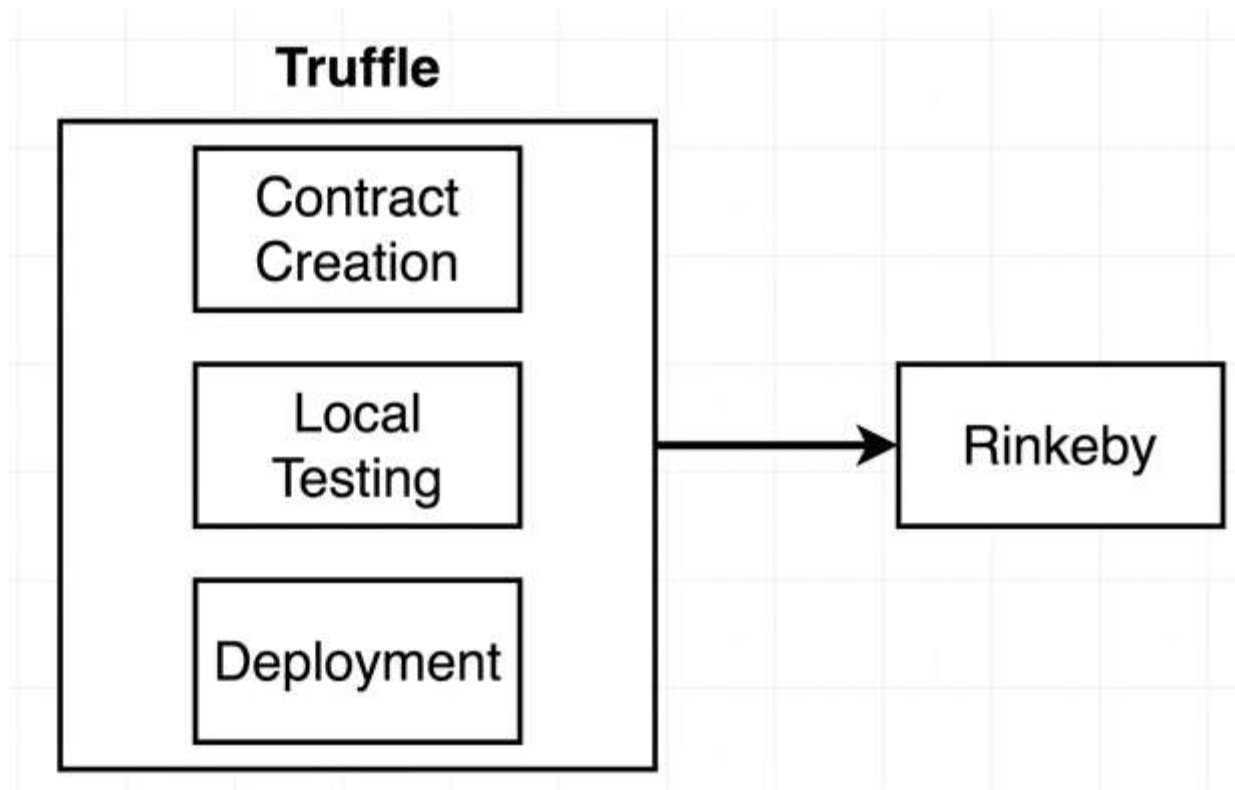
How to deploy the contract on real network



At core it will follow same process



For that we need truffle



Issues with truffle

Truffle

Undergoing rapid development

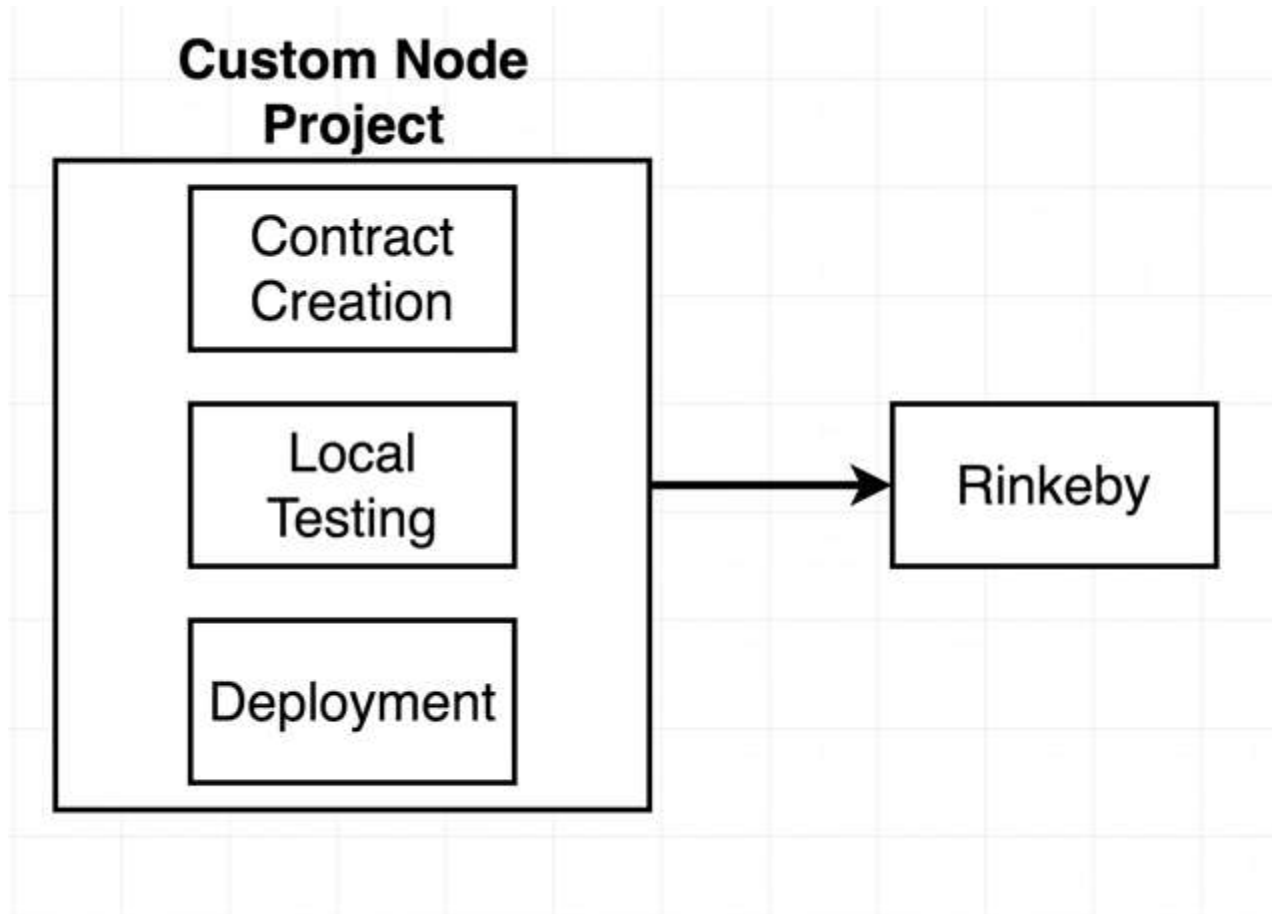
Some things don't work well

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

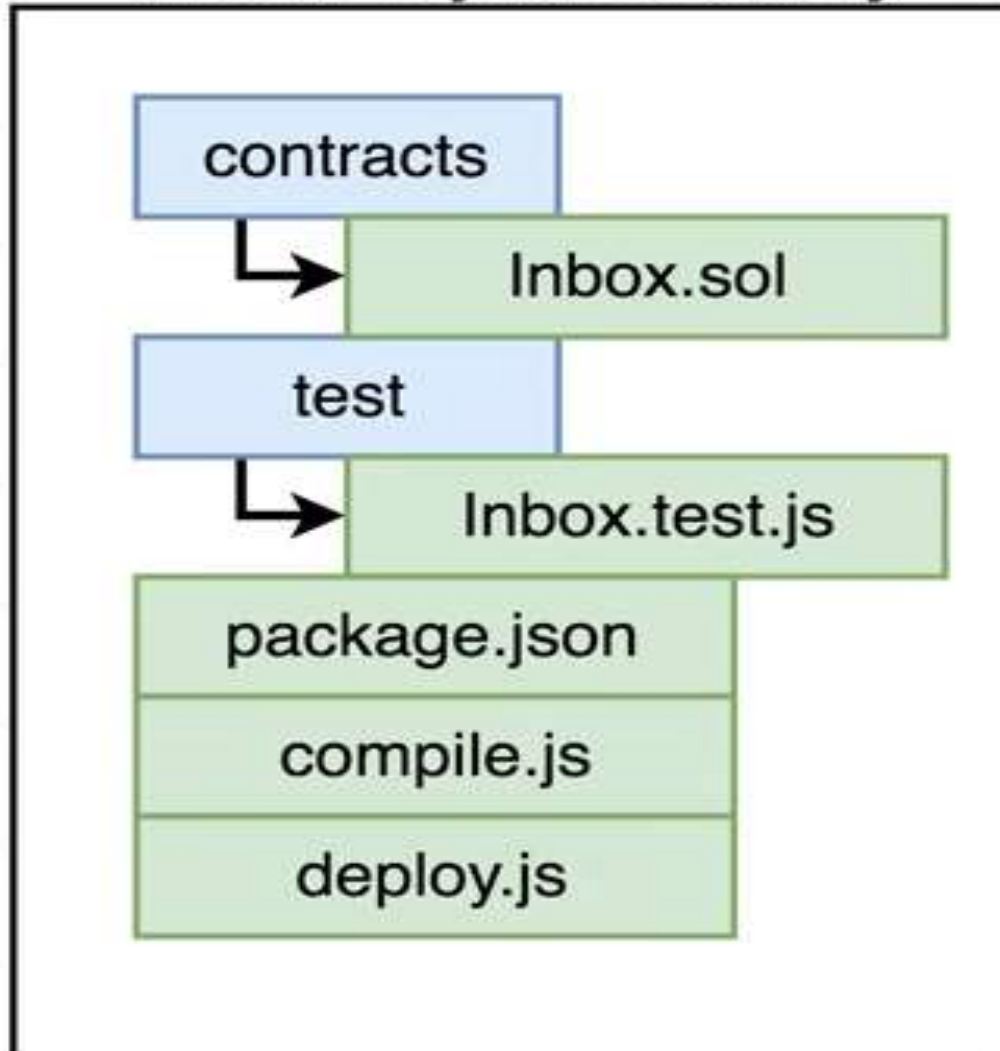
cd Inbox

npm init

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

Project structure

Inbox Project Directory



First file



```
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

```
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

```
→ inbox git:(034-project-files) ✗ node compile.js
```

```
{ contracts:
```

```
  { ':Inbox':
```

```
    { assembly: [Object],
```

```
      bytecode: '6060604052341561000f57600080fd5b6040516103973803806103978339
```

```
81016040528080519091019050600081805161003d929160200190610044565b50506100df565b8
```

```
28054600181600116156101000203166002900490600052602060002090601f0160209004810192
```

```
82601f1061008557805160ff19168380011785556100b2565b828001600101855582156100b2579
```

```
182015b828111156100b2578251825591602001919060010190610097565b506100be9291506100
```

```
c2565b5090565b6100dc91905b808211156100be57600081556001016100c8565b90565b6102a98
```

```
06100ee6000396000f30060606040526004361061004b5763ffffffff7c01000000000000000000
```

```
0000000000000000000000000000000000000000000000000000000000000000000000000000
```

```
0000000000000000000000000000000000000000000000000000000000000000000000000000
```

```
7ce146100a3575b600080fd5b341561005b57600080fd5b6100a160046024813581810190830135
```

```
806020601f820181900481020160405190810160405281815292919060208401838380828437509
```

```
4965061012d955050505050505050565b005b34156100ae57600080fd5b6100b6610144565b60405160
```

```
208082528190810183818151815260200191508051906020019080838360005b838110156100f25
```

```
780820151838201526020016100da565b50505050905090810190601f16801561011f5780820380
```

```
516001836020036101000a031916815260200191505b509250505060405180910390f35b6000818
```

Actual Byte code to be deployed in ETH Network

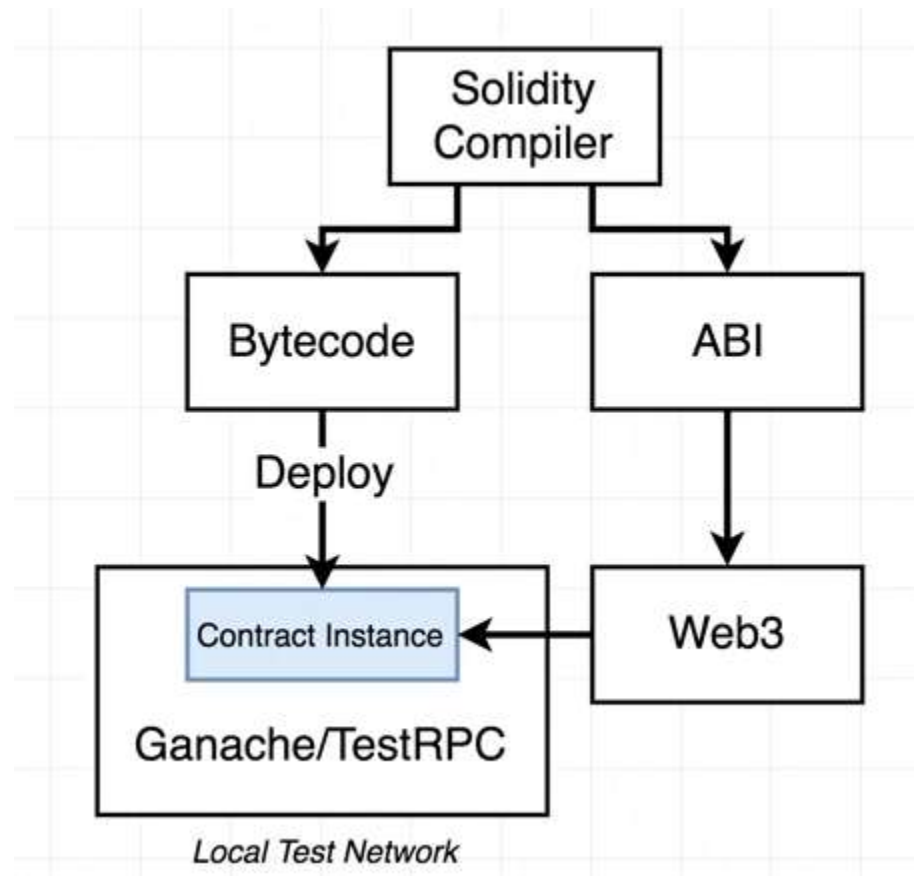
ABI (Application Binary Interface)

```
functionHashes: [Object],
gasEstimates: [Object],
interface: '[{"constant":false,"inputs":[{"name":"newMessage","type":"string"}],"name":"setMessage","outputs":[],"payable":false,"stateMutability":"nonpayable","type":"function"}, {"constant":true,"inputs":[],"name":"message","outputs":[{"name":"","type":"string"}],"payable":false,"stateMutability":"view","type":"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","type":"string"}],"name":"setMessage","outputs":[],"payable":false,"stateMutability":"nonpayable","type":"function"}, {"constant":true,"inputs":[],"name":"message","outputs":[{"name":"","type":"string"}],"payable":false,"stateMutability":"view","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 **inbox.test.js** file in that folder.

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

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

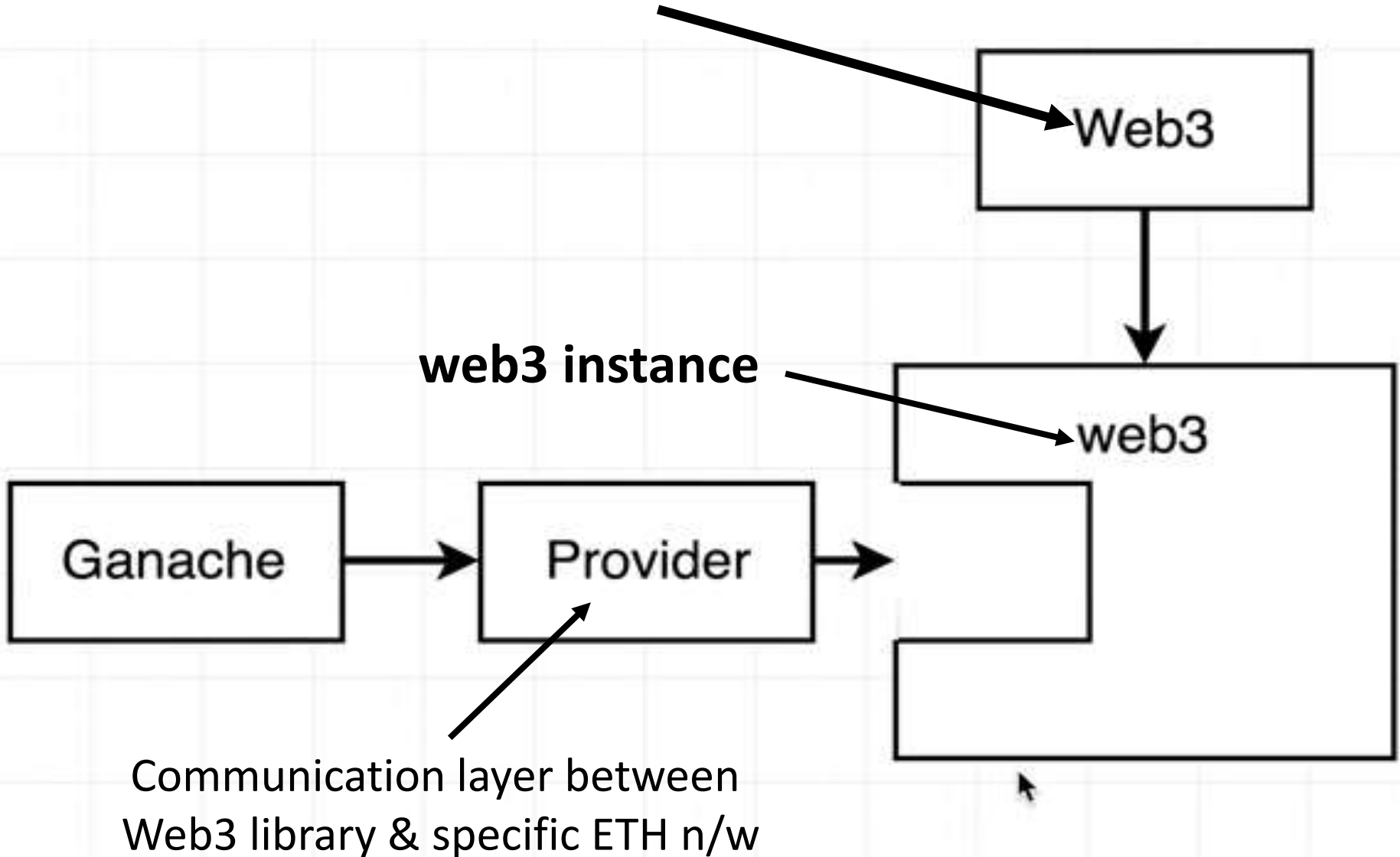
```
const Web3 = require('web3');           // Web3 is constructor
```

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

```
//web3 → instance of Web3
```

Web3 Providers

Web3 Constructor



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