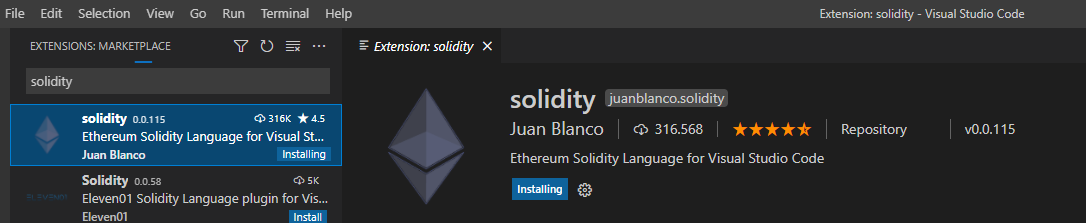
**Setup ambiente**

1. Installare un editor, vscode è ottimo <https://code.visualstudio.com/download>
2. In vscode aggiungere l’estensione per solidity



1. Installare nodejs <https://nodejs.org/it/download/> qualsiasi versione, meglio se LTS.
2. Installare Ganache <https://www.trufflesuite.com/ganache> che sarà la blockchain di test locale
3. Al termine dell’installazione aprire terminale.
4. Installare truffle framework con:

npm install -g truffle

1. Creare una cartella di test

mkdir test

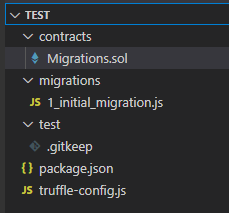
cd test

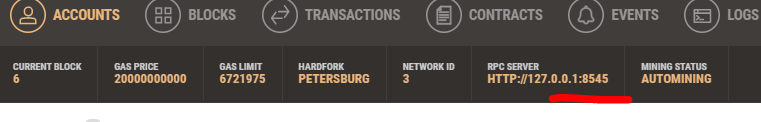
1. Avviare un nuovo progetto node

npm init

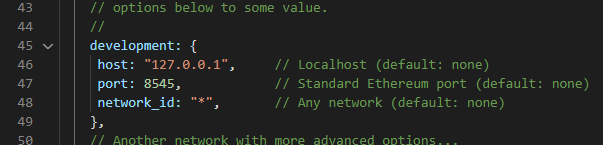
1. Inserire tutti i dati richiesti. Essendo un test si può anche premere enter fino al termine.
2. Avviare un nuovo progetto truffle

truffle init

1. Verrà creata una struttura di cartelle e file
   1. La cartella contracts contiene gli smart contract sviluppati che vanni poi caricati in chain. Migrations.sol è un contract di base che truffle usa per fare il deploy dei successivi, questo smart contract è utile quando è richiesto un processo di aggiornamento continuo del contratto. Per il momento lo lasciamo poi verificheremo se utilizzare questo metodo per il deploy oppure se utilizzare remix.
   2. La cartella migrations contiene gli script js che truffle userà per fare il deploy degli smart contract.
   3. La cartella test serve a contenere i file js che esguiranno test automatizzati sui contract.
   4. package.json è il file di node per il pacchetti aggiuntivi, le versioni ecc.
   5. truffle-config.js è il file di configurazione di truffle.
2. Apriamo ganache e prendiamo nota della porta utilizzata per la connessione

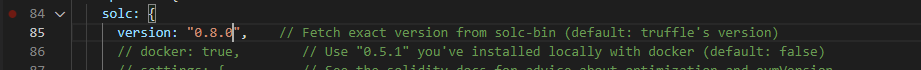


1. Con vscode apriamo il file truffle-config.js e decommentiamo le righe 45-49 (ctrl + k + u)



Verificare che la porta alla riga 47 sia uguale a quella usata da ganache.

1. Decommentiamo anche la riga 85 e settiamo la versione del compilatore a 0.8.0.



1. Salvare il file.
2. Da terminale importare openzeppelin con:

npm install --save @openzeppelin/contracts

1. Verrà create una cartella nodemodules con all’interno la raccolta di contracts di openzeppelin. Questo repository contiene molteplici classi e preset di contract già validati e funzionabili, utilizzabili per estendere il nostro contract di base.

**Test setup**

1. Creare il file test.sol nella cartella contracts.
2. Incollare questo contenuto

pragma solidity ^0.8.0;

import "@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol";

contract testContract is ERC721URIStorage{

uint256 public tokenCounter;

constructor (string memory \_name, string memory \_symbol) ERC721 (\_name, \_symbol){

tokenCounter = 0;

}

function test(string memory input) public pure returns (string memory) {

return input;

}

}

1. Salvare il file.
2. Creare un nuovo file nella cartella migrations chiamato 2\_deploy.js. Può essere assegnato qualsiasi nome a patto che inizi con il numero progressivo + \_.
3. Incollare questo contenuto

var testContract = artifacts.require("testContract");

module.exports = function(deployer) {

deployer.deploy(testContract, "Test Contract", "TC");

};

1. Questo file verrà utilizzato da truffle per fare il deploy dello smart contract creato in precedenza.
2. Dal terminale accediamo alla console di truffle con

truffle console

1. Se la configurazione di truffle-config.js e ganache è corretta, il teminale si connetterà e mostrerà la shell

truffle(development)>

1. Per ottenere l’indirizzo di un account basta

accounts[0]

dove [0] è l’indice da 1 a 9 degli account disponibili in ganache.

1. Per ottenere il balance di un account si può richiamare la relativa funzione di web3 basta

web3.eth.getBalance(accounts[0])

il risultato sarà ovviamente in wei.



1. Possiamo ora fare il deploy del nostro contract con

truffle migrate --reset

1. Truffle compilerà i contracts e ne farà il deploy sulla chain di ganache.
2. Per accedere al contract appena deployato assegnamolo a una variabile locale

let test = await testContract.deployed()

1. Possiamo quindi recuperare l’address del contratto con

test.address

1. Oppure il nome

test.name()

1. Oppure il simbolo

test.symbol()

1. Oppure l’abi

test.abi

1. Possiamo anche chiamare la funzione test definita nel contract con

await test.test.call("this is a test")

1. Per chiudere la console usiamo

.exit

**Test automatici**

Verranno usate mochajs e chai assertion library.

1. Installiamo mochajs

npm install --global mocha

1. Installiamo chai

npm install chai chai-as-promised

1. All’interno della cartella test creare un nuovo file autotest.js
2. Incollare nel file

const { assert } = require("chai")

const test = artifacts.require("testContract")

require("chai")

.use(require("chai-as-promised"))

.should()

contract("testContract", (accounts) => { //accounts from ganache

before(async () => { //get contract reference global

contract = await test.deployed()

})

describe("deployment", async () => {

it("deploys successfully", async () => {

const address = contract.address

assert.notEqual(address, 0x0)

assert.notEqual(address, "")

assert.notEqual(address, null)

assert.notEqual(address, undefined)

})

it("has a name", async () => {

const name = await contract.name()

assert.equal(name, "Test Contract")

})

it("has a symbol", async () => {

const symbol = await contract.symbol()

assert.equal(symbol, "TC")

})

it("check test function", async () => {

const text = await contract.test.call("Test Function Text")

assert.equal(text, "Test Function Text")

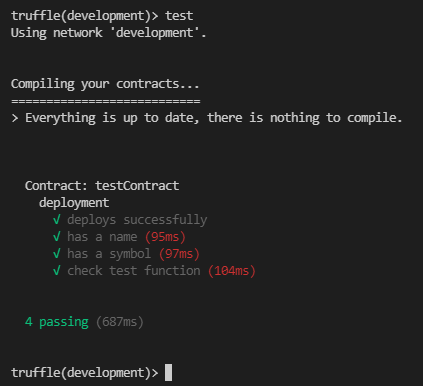
})

})

})

1. Salvare il file.
2. Dalla console truffle

test



**Riferimenti**

OpenZeppelin

* <https://openzeppelin.com/>
* <https://openzeppelin.com/contracts/>
* <https://github.com/OpenZeppelin/openzeppelin-contracts>
* <https://wizard.openzeppelin.com/>
* <https://docs.openzeppelin.com/contracts/3.x/erc1155>
* <https://docs.openzeppelin.com/contracts/3.x/api/token/erc1155>

Truffle

* <https://www.trufflesuite.com/>
* <https://www.trufflesuite.com/ganache>
* <https://www.trufflesuite.com/docs/truffle/getting-started/installation>
* <https://www.trufflesuite.com/docs/truffle/getting-started/using-truffle-develop-and-the-console>

Testing

* <https://mochajs.org/>
* <https://mochajs.org/#installation>
* <https://www.chaijs.com/>

Tutorial

* <https://www.youtube.com/watch?v=YPbgjPPC1d0>
* <https://www.youtube.com/watch?v=ZH_7nEIJDUY>