**Practical No: 08**

**Aim: Smart Contract using Truffle Framework.**

**Theory:**

* **Truffle framework:**
* Truffle is a world-class development environment, testing framework and asset pipeline for blockchains using the Ethereum Virtual Machine (EVM), aiming to make life as a developer easier.
* Truffle is widely considered the most popular tool for blockchain application development with over 1.5 million lifetime downloads. Truffle supports developers across the full lifecycle of their projects, whether they are looking to build on Ethereum, Hyperledger, Quorum, or one of an ever-growing list of other supported platforms.
* Paired with Ganache, a personal blockchain, and Drizzle, a front-end dApp development kit, the full Truffle suite of tools promises to be an end-to-end dApp development platform.

1. Built-in smart contract compilation, linking, deployment and binary management.
2. Automated contract testing for rapid development.
3. Scriptable, extensible deployment & migrations framework.
4. Network management for deploying to any number of public & private networks.
5. Package management with EthPM & NPM, using the ERC190 standard.
6. Interactive console for direct contract communication.
7. Configurable build pipeline with support for tight integration.
8. External script runner that executes scripts within a Truffle environment.

**You can install Truffle with NPM in your command line like this:**

**$ npm install -g truffle**

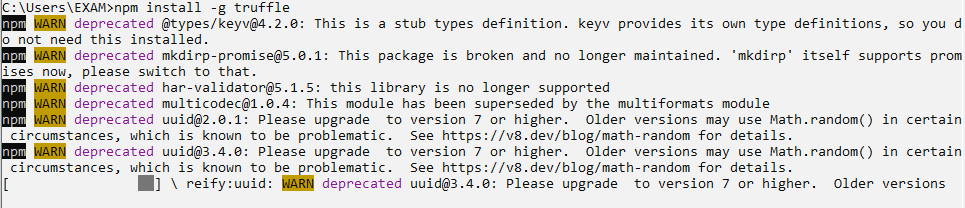
* **Smart Contract:**

A smart contract is a stand-alone script usually written in Solidity and compiled into binary or JSON and deployed to a specific address on the blockchain. In the same way

that we can call a specific URL endpoint of a RESTful API to execute some logic through an HttpRequest, we can similarly execute the deployed smart contract at a specific address by submitting the correct data along with the necessary Ethereum to call the deployed and compiled Solidity function.

1. **Install Node JS and Truffle Suite to develop and migrate the smart contracts into the Private Blockchain network. Make use of Truffle tools like compile, migrate and test for compilation, migration and testing the smart contracts through Blockchain**

**> npm install -g truffle**



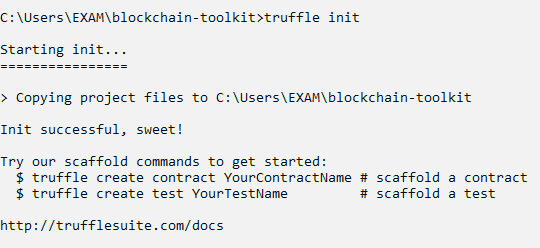
**> mkdir blockchain-toolkit**



**> cd blockchain-toolkit**



**> truffle init**



**> touch package.json**



**copy-and-pasting the below code into package.json file**

{

"name": "blockchain-toolkit",

"version": "1.0.0",

"description": "The Complete Blockchain Developer Toolkit for 2019 & Beyond",

"main": "truffle-config.js",

"directories": {

"test": "test"

},

"scripts": {

"dev": "lite-server",

"test": "echo \"Error: no test specified\" && sexit 1"

},

"author": "gregory@dappuniversity.com",

"license": "ISC",

"devDependencies": {

"bootstrap": "4.1.3",

"chai": "^4.1.2",

"chai-as-promised": "^7.1.1",

"chai-bignumber": "^2.0.2",

"dotenv": "^4.0.0",

"ganache-cli": "^6.1.8",

"lite-server": "^2.3.0",

"nodemon": "^1.17.3",

"solidity-coverage": "^0.4.15",

"truffle": "5.0.0-beta.0",

"truffle-contract": "3.0.6",

"truffle-hdwallet-provider": "^1.0.0-web3one.0"

}

}

**Save package.json file**

**Start developing a smart contract using solidity**

**> touch ./contracts/MyContract.sol**

****

**Copy the below contract code and save in Mycontract.sol**

pragma solidity >=0.4.2 <=0.8.17;

contract MyContract {

string value;

constructor() public {

value = "myValue";

}

function get() public view returns(string memory) {

return value;

}

function set(string memory \_value) public {

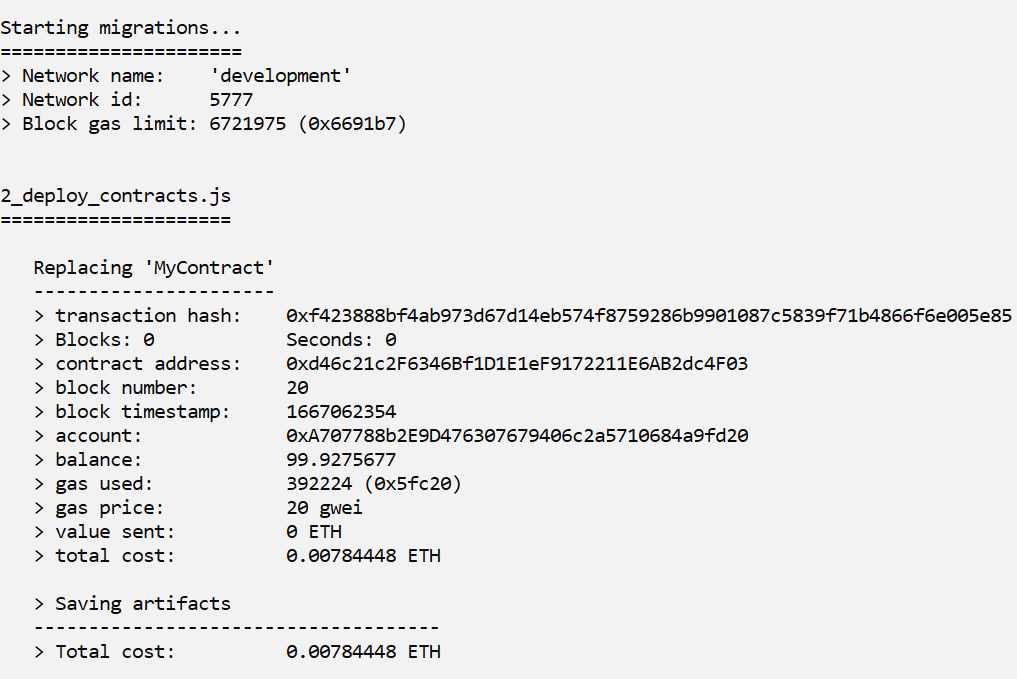
value = \_value;

}

}

**> truffle compile**

****

****

**Update the project configuration file (Find the file truffle-config.js and paste the following code)**

module.exports = {

networks: {

development: {

host: "127.0.0.1",

port: 7545,

network\_id: "\*" // Match any network id

}

},

solc: {

optimizer: {

enabled: true,

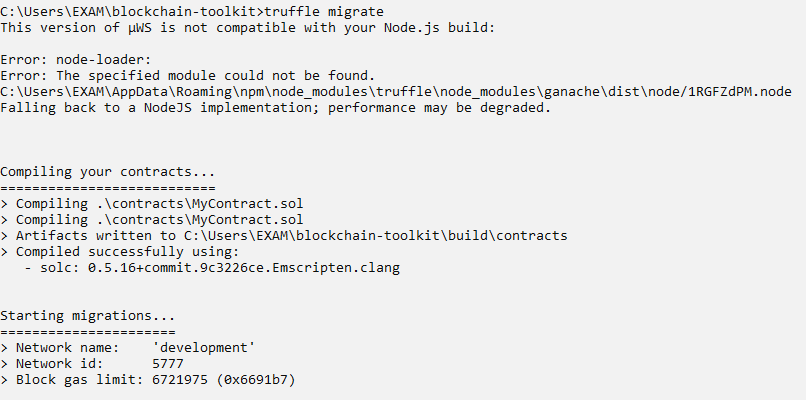
runs: 200

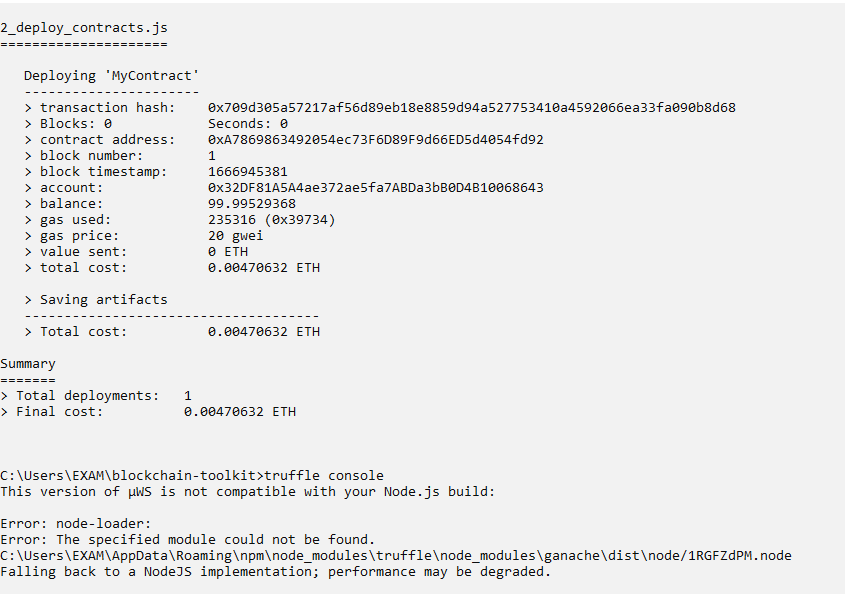
}

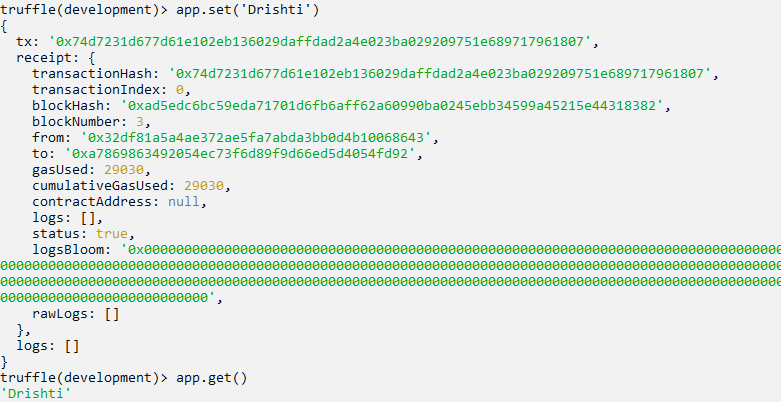
}

}



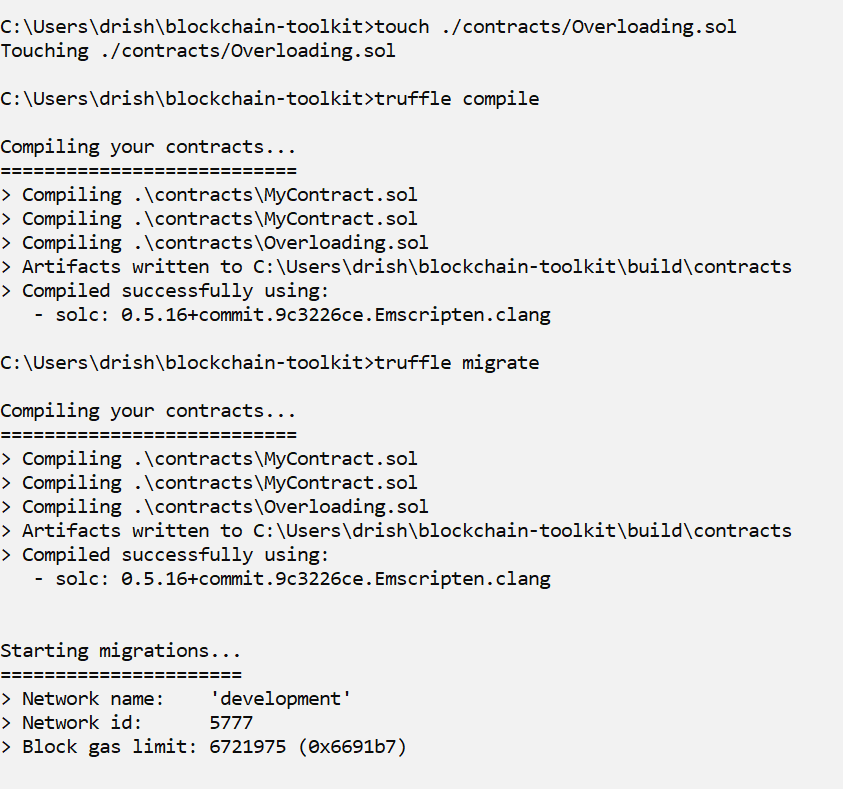
****

****

****

**2. Create a Smart contract to simulate function overloading . Execute the contract using truffle framework.**

**🡪** **touch ./contracts/Overloading.sol**

****

**Code:**

pragma solidity >=0.4.2 <=0.8.0;

contract Overloading {

function getCal(uint a, uint b,uint c) public pure returns(uint){

return a\*b\*c;

}

function getCal(uint a, uint b) public pure returns(uint){

return a + b;

}

function callValueWithTwoArguments() public pure returns(uint){

return getCal(1,2);

}

function callValueWithThreeArguments() public pure

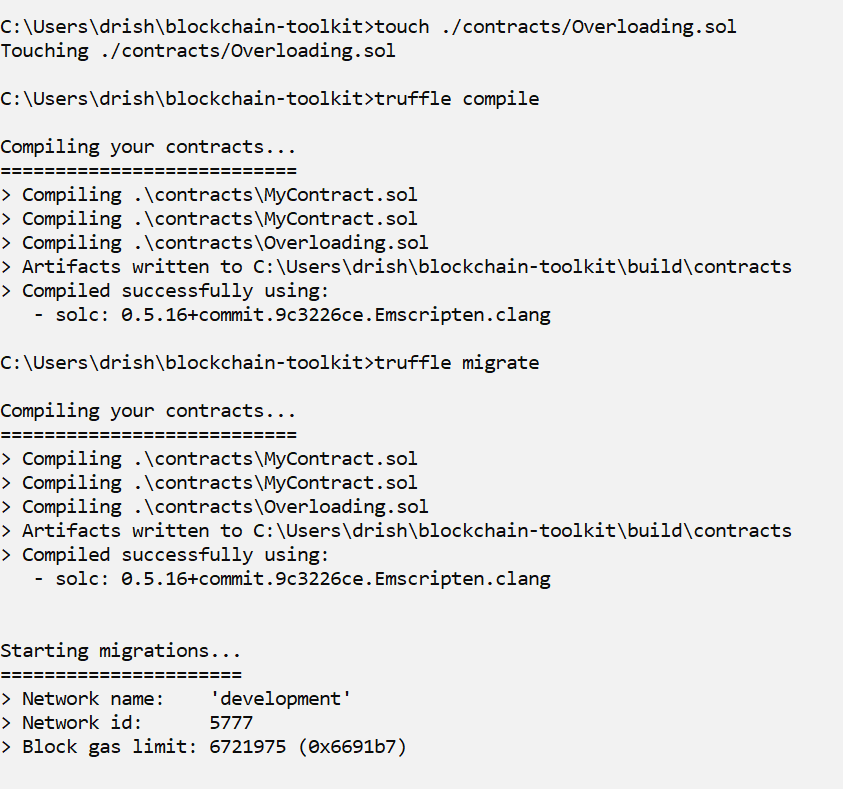
returns(uint){

return getCal(1,2,3);

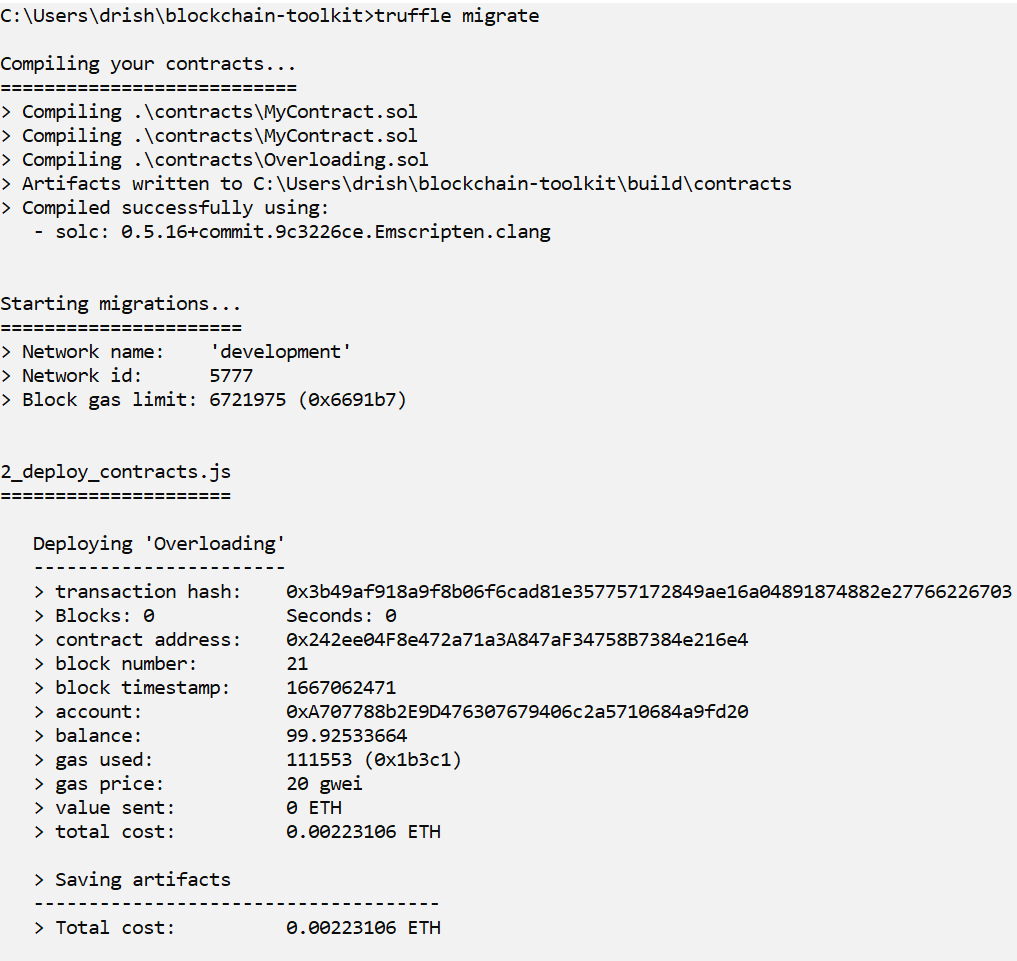
}

}

**🡪truffle compile**

****

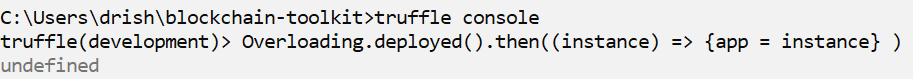
**🡪truffle migrate**

****

****

**🡪truffle console**

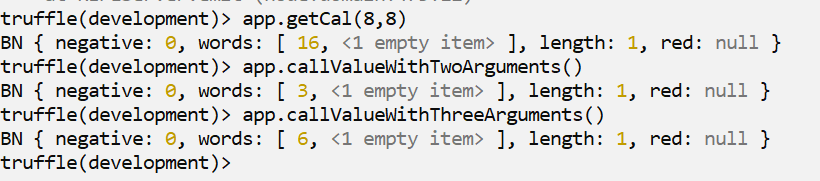
**🡪 Overloading.deployed().then((instance) => {app = instance} )**

****

**🡪app.getCal(8,8)**

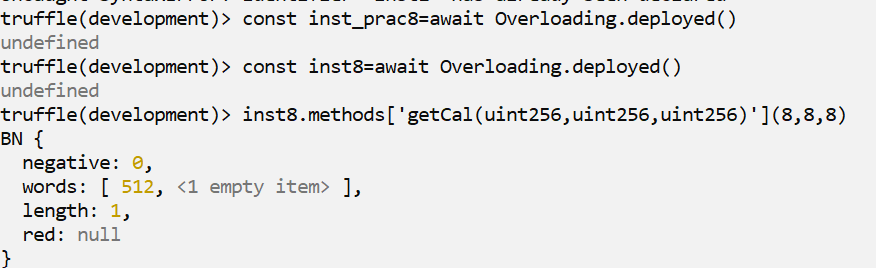
**🡪app.callValueWithTwoArguments()**

**🡪callValueWithThreeArguments ()**

****

**🡪 const inst8=await Overloading.deployed()**

**🡪inst8.methods['getSum(uint256,uint256,uint256)'](8,8,8)**

****

**🡪inst8.methods['getSum(uint256,uint256)'](8,8)**

****

**Transaction Loss:**

****

**Conclusion:** We have successfully executed the smart contract using the truffle framework.