

PREPARANDO ENTORNO PARA DESARROLLAR SOLIDITY EN RED ETHEREUM

Fuente: <https://archive.trufflesuite.com/docs/truffle>

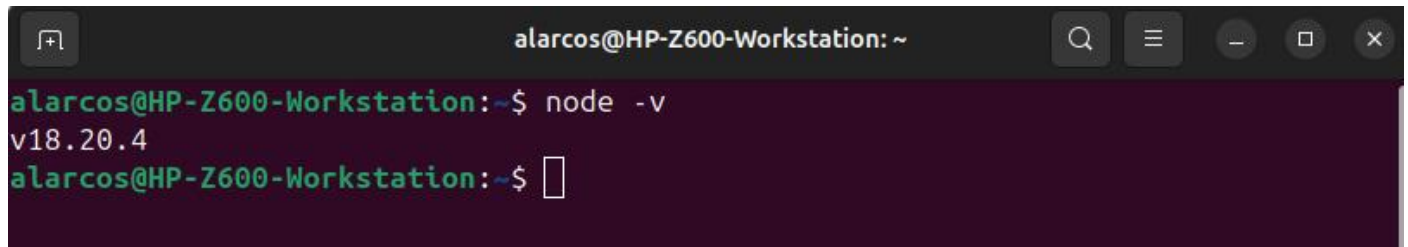
INSTALACIÓN DE COMPONENTES

Instalamos CURL, para obtener el Node Version Manager (NVM): (<https://github.com/nvm-sh/nvm>):

```
sudo apt-get install curl  
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.1/install.sh | bash
```

Una vez tenemos NVM instalado, podemos proceder con la instalación de NodeJS. Necesitamos la versión 14 como mínimo (en este caso trabajamos con la 18):

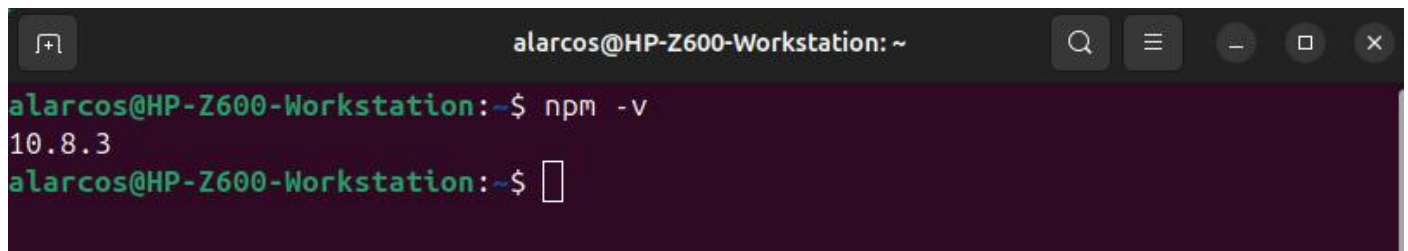
```
nvm install 18  
nvm use 18  
node -v
```



A terminal window titled 'alarcos@HP-Z600-Workstation: ~' with search, menu, and window control icons. The prompt is 'alarcos@HP-Z600-Workstation:~\$'. The command 'node -v' is entered and executed, returning 'v18.20.4'. The prompt returns to 'alarcos@HP-Z600-Workstation:~\$'.

Instalamos la última versión de NPM:

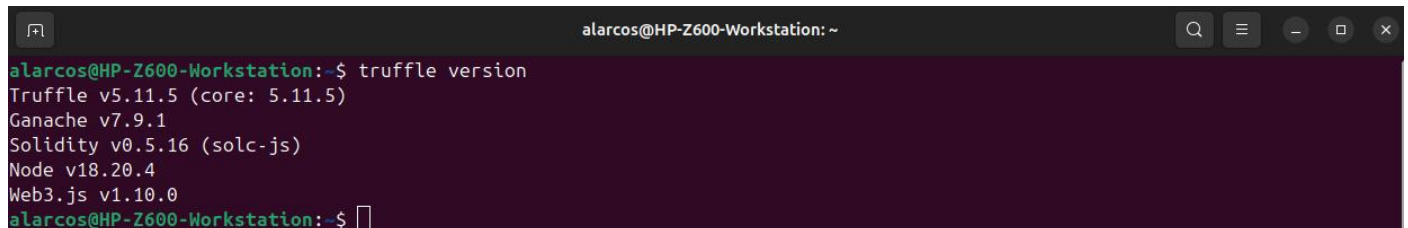
```
npm i -g npm  
npm -v
```



A terminal window titled 'alarcos@HP-Z600-Workstation: ~' with search, menu, and window control icons. The prompt is 'alarcos@HP-Z600-Workstation:~\$'. The command 'npm -v' is entered and executed, returning '10.8.3'. The prompt returns to 'alarcos@HP-Z600-Workstation:~\$'.

Ahora instalamos Truffle y el cliente Ethereum:

```
npm install -g truffle  
truffle version
```



A terminal window titled 'alarcos@HP-Z600-Workstation: ~' with search, menu, and window control icons. The prompt is 'alarcos@HP-Z600-Workstation:~\$'. The command 'truffle version' is entered and executed, returning the following output:
Truffle v5.11.5 (core: 5.11.5)
Ganache v7.9.1
Solidity v0.5.16 (solc-js)
Node v18.20.4
Web3.js v1.10.0
The prompt returns to 'alarcos@HP-Z600-Workstation:~\$'.

CREACIÓN DE UN NUEVO PROYECTO

Como primer proyecto vamos a realizar un “Hello World” para comprobar que todo está instalado y configurado correctamente.

En primer lugar en una nueva consola mantenemos el cliente Ethereum abierto:

```
ganache-cli
```

```
alarcos@HP-Z600-Workstation: ~$ ganache-cli
ganache v7.9.2 (@ganache/cli: 0.10.2, @ganache/core: 0.10.2)
Starting RPC server

Available Accounts
=====
(0) 0xb136730AD3C66996f04271BB480F89B9A9C13896 (1000 ETH)
(1) 0x97dddA214b82179b614b65479A82aa9fA6603176 (1000 ETH)
(2) 0xaEb817DD056a2603AB4B3201344886546b45e019 (1000 ETH)
(3) 0x9b7C1A673253486dfB9749d6e57c6fAA91156567 (1000 ETH)
(4) 0x84ed84B5628EC2ee2F98dcAF0056D60254f3B8a0 (1000 ETH)
(5) 0x2Ae2E25D4a49589449ed2c81Ee1Cc5d53D21D8E5 (1000 ETH)
(6) 0x9C96A12ce78701230dFd0E8894F81065E674cAEa (1000 ETH)
(7) 0x49187c37b044847D0B05ab0D1AcB5803f9f10898 (1000 ETH)
(8) 0xAAb0775A0ab8797338daaEdB92a0Fab7E74E1570 (1000 ETH)
(9) 0xc17fD8050A191bB913F44291E738dcb57fb4F3E5 (1000 ETH)

Private Keys
=====
(0) 0x60be9a57247255653f2e22aabc013f2f91330fd9f41e115c1023e2d13765e214
(1) 0xd23bc67f7e2660c96f476a6227b35b9b51b6a02f8758c0c1fe72719b0ac2819d
(2) 0x3bc8e7ca80b57ab7391cd2a186c773025329cf1ecfb0b89a6cf6396694b29e15
```

Dejaremos esta consola ejecutándose y comenzaremos desde otra a crear nuestro proyecto:

```
mkdir HelloWorld
cd HelloWorld
truffle init
```

```
alarcos@HP-Z600-Workstation: ~/Documentos/GitHub/ALBA-ethereum/HelloWorld$ mkdir HelloWorld
alarcos@HP-Z600-Workstation: ~/Documentos/GitHub/ALBA-ethereum$ cd HelloWorld/
alarcos@HP-Z600-Workstation: ~/Documentos/GitHub/ALBA-ethereum/HelloWorld$ truffle init

Starting init...
=====

> Copying project files to /home/alarcos/Documentos/GitHub/ALBA-ethereum/HelloWorld

Init successful, sweet!

Try our scaffold commands to get started:
$ truffle create contract YourContractName # scaffold a contract
$ truffle create test YourTestName        # scaffold a test

http://trufflesuite.com/docs

alarcos@HP-Z600-Workstation: ~/Documentos/GitHub/ALBA-ethereum/HelloWorld$
```

Esto nos genera la estructura de un proyecto con los directorios: contracts, migrations, test y el archivo: truffle-config.js

```
alarcos@HP-Z600-Workstation: ~/Documentos/GitHub/ALBA-ethereum/HelloWorld$ ls
contracts migrations test truffle-config.js
alarcos@HP-Z600-Workstation: ~/Documentos/GitHub/ALBA-ethereum/HelloWorld$
```

Añadimos nuestro proyecto a Ganache (nuestro cliente Ethereum):

```
???????????
```

Editamos el archivo “truffle-config.js” y completamos la red de development, la información la necesaria la tenemos en nuestra consola con ganache-cli, al final:

```
300000000
Call Gas Limit
=====
500000000
Chain
=====
Hardfork: shanghai
Id:      1337
RPC Listening on 127.0.0.1:8545
█
```

```
60 networks: {
61   // Useful for testing. The `development` name is special - truffle uses it by default
62   // if it's defined here and no other network is specified at the command line.
63   // You should run a client (like ganache, geth, or parity) in a separate terminal
64   // tab if you use this network and you must also set the `host`, `port` and `network_id`
65   // options below to some value.
66   //
67   development: {
68     host: "127.0.0.1",    // Localhost (default: none)
69     port: 8545,          // Standard Ethereum port (default: none)
70     network_id: "*",     // Any network (default: none)
71   },
72   //

```

Dentro de la carpeta “contracts” creamos un nuevo contrato (con extensión .sol):

```
touch contracts/HelloWorld.sol
```

El código necesario para este programa es el siguiente:

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.8.0;

contract HelloWorld {
    string private message;

    constructor() {
        message = string("Hello, World!");
    }

    function getMessage() public view returns (string memory) {
        return message;
    }
}
```

A continuación, compilamos el proyecto:

```
truffle compile          #Compilará los ficheros modificados desde la última compilación
truffle compile --all    #Compilará todos los ficheros
```

```
alarcos@HP-Z600-Workstation: ~/Documentos/GitHub/ALBA-ethereum/HelloWorld
alarcos@HP-Z600-Workstation:~/Documentos/GitHub/ALBA-ethereum/HelloWorld$ truffle compile --all

Compiling your contracts...
=====
> Compiling ./contracts/HelloWorld.sol
> Artifacts written to /home/alarcos/Documentos/GitHub/ALBA-ethereum/HelloWorld/build/contracts
> Compiled successfully using:
   - solc: 0.8.21+commit.d9974bed.Emscripten.clang
alarcos@HP-Z600-Workstation:~/Documentos/GitHub/ALBA-ethereum/HelloWorld$
```

Creemos un archivo de migración dentro del directorio “migrations”, siempre con el formato:

N_migration_file_name.js donde N es un número entero, por ejemplo, el siguiente:

```
touch migrations/1_first_deploy.js
```

```
alarcos@HP-Z600-Workstation: ~/Documentos/GitHub/ALBA-ethereum/HelloWorld
alarcos@HP-Z600-Workstation:~/Documentos/GitHub/ALBA-ethereum/HelloWorld$ touch migrations/1_first_deploy.js
alarcos@HP-Z600-Workstation:~/Documentos/GitHub/ALBA-ethereum/HelloWorld$
```

Incluimos el siguiente código para que el “deployer” despliegue el contract que acabamos de crear:

```
const HelloWorld = artifacts.require("HelloWorld");

module.exports = function (deployer) {
  deployer.deploy(HelloWorld);
};
```

Migramos el proyecto compilado a la red “development”:

```
truffle migrate --network development
truffle migrate --network development --reset #Para nuevas migraciones si se hizo una anterior
```

```
alarcos@HP-Z600-Workstation: ~/Documentos/GitHub/ALBA-ethereum/HelloWorld
alarcos@HP-Z600-Workstation:~/Documentos/GitHub/ALBA-ethereum/HelloWorld$ truffle migrate --network development --reset

Compiling your contracts...
=====
> Everything is up to date, there is nothing to compile.

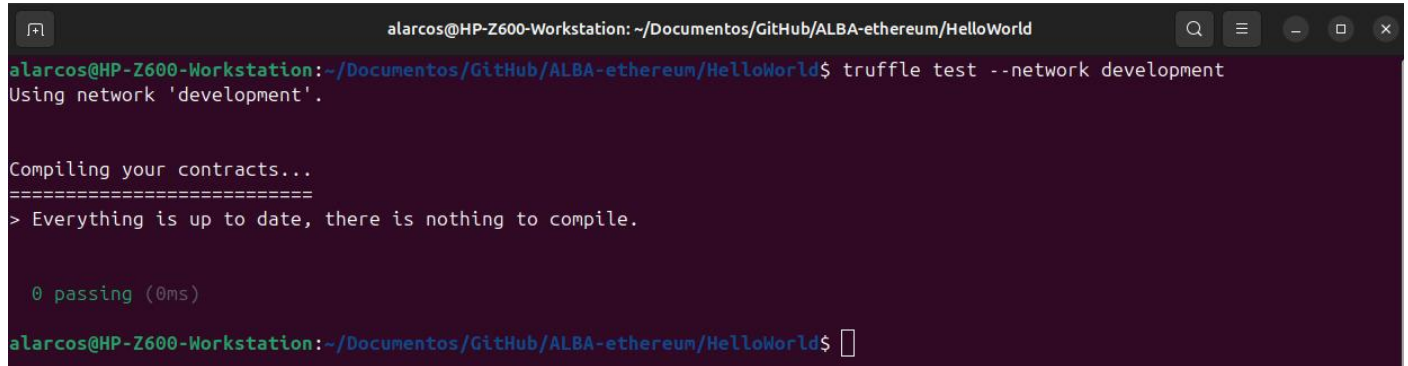
Starting migrations...
=====
> Network name:      'development'
> Network id:        1727174527667
> Block gas limit:   30000000 (0x1c9c380)

1_first_deploy.js
=====

Deploying 'HelloWorld'
-----
> transaction hash:  0xe4f0730cbe7b8caaf0e4260893eaf3c1e0eda862fd63d1d1d4a2069ec6627050
> Blocks: 0         Seconds: 0
```


Y finalmente testamos el despliegue del contrato:


```
truffle test --network development
```

A terminal window titled 'alarcos@HP-Z600-Workstation: ~/Documentos/GitHub/ALBA-ethereum/HelloWorld'. The prompt is 'alarcos@HP-Z600-Workstation:~/Documentos/GitHub/ALBA-ethereum/HelloWorld\$'. The command 'truffle test --network development' is entered. The output shows 'Using network 'development''. Then 'Compiling your contracts...' is displayed, followed by a separator line and the message '> Everything is up to date, there is nothing to compile.' Below this, it says '0 passing (0ms)'. The prompt returns to 'alarcos@HP-Z600-Workstation:~/Documentos/GitHub/ALBA-ethereum/HelloWorld\$'.

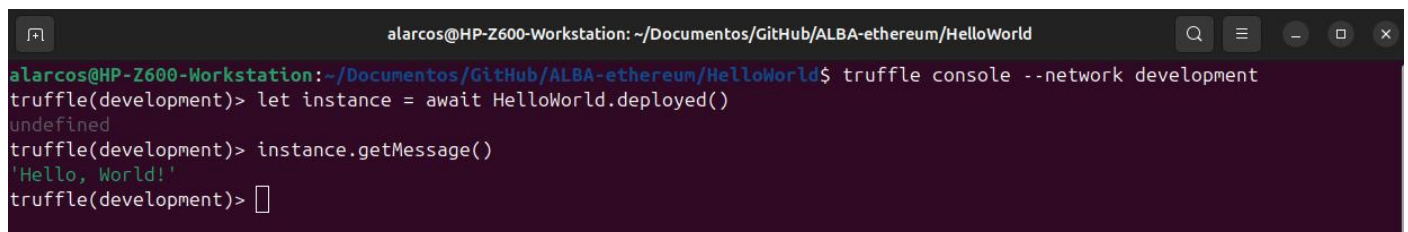
Ahora podemos interactuar con el contrato desde la consola de truffle:

```
truffle console --network development
```

Si dentro de la consola, llamamos por su nombre a nuestro contrato “HelloWorld”, nos aparecerán todas las funciones que podemos llamar:

A terminal window titled 'alarcos@HP-Z600-Workstation: ~/Documentos/GitHub/ALBA-ethereum/HelloWorld'. The prompt is 'alarcos@HP-Z600-Workstation:~/Documentos/GitHub/ALBA-ethereum/HelloWorld\$'. The command 'truffle console --network development' is entered. The prompt changes to 'truffle(development)>'. The user enters 'HelloWorld'. The output shows '[Function: TruffleContract] {' followed by a list of methods: '_constructorMethods: {', 'configureNetwork: [Function: configureNetwork]', 'setProvider: [Function: setProvider]', 'new: [Function: new]', 'at: [AsyncFunction: at]', 'deployed: [AsyncFunction: deployed]', 'defaults: [Function: defaults]', 'hasNetwork: [Function: hasNetwork]', 'isDeployed: [Function: isDeployed]', 'detectNetwork: [AsyncFunction: detectNetwork]', 'setNetwork: [Function: setNetwork]', 'setNetworkType: [Function: setNetworkType]', 'setWallet: [Function: setWallet]', 'resetAddress: [Function: resetAddress]', 'link: [Function: link]', 'clone: [Function: clone]', 'addProp: [Function: addProp]', 'toJSON: [Function: toJSON]'.

Vamos a realizar una llamada sobre la función “getMessage()”:

A terminal window titled 'alarcos@HP-Z600-Workstation: ~/Documentos/GitHub/ALBA-ethereum/HelloWorld'. The prompt is 'alarcos@HP-Z600-Workstation:~/Documentos/GitHub/ALBA-ethereum/HelloWorld\$'. The command 'truffle console --network development' is entered. The prompt changes to 'truffle(development)>'. The user enters 'let instance = await HelloWorld.deployed()'. The output shows 'undefined'. The user enters 'instance.getMessage()'. The output shows ''Hello, World!'. The prompt returns to 'truffle(development)>'.

Ahora, para ejecutar un script lo haremos de la siguiente forma (utilizando JavaScript como lenguaje):

```
mkdir scripts
touch script/miScript.js
```

Editamos el script para interactuar con nuestro contrato HelloWorld:

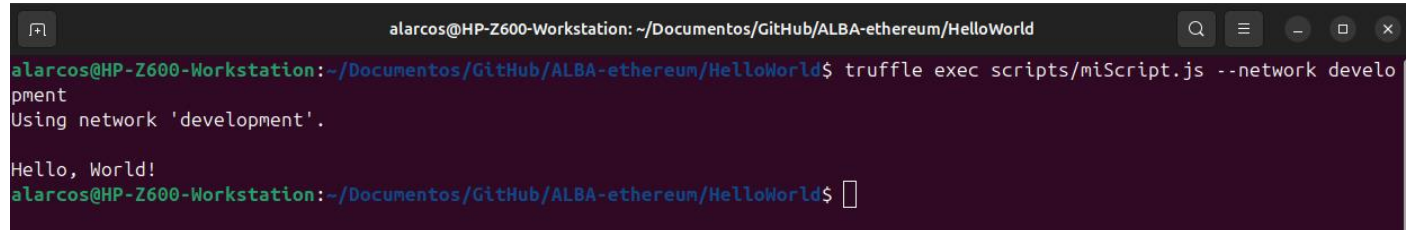
```
const HelloWorld = artifacts.require("HelloWorld");

module.exports = async function(callback){
```

```
try{
    const instance = await HelloWorld.deployed();
    const result = await instance.getMessage();
    console.log(result);
} catch(error) {
    console.error(error);
}
callback();
};
```

Y lo ejecutamos con:

```
truffle exec scripts/miScript.js --network development
```

A terminal window with a dark background and light green text. The window title is 'alarcos@HP-Z600-Workstation: ~/Documentos/GitHub/ALBA-ethereum/HelloWorld'. The prompt is 'alarcos@HP-Z600-Workstation:~/Documentos/GitHub/ALBA-ethereum/HelloWorld\$'. The command entered is 'truffle exec scripts/miScript.js --network development'. The output is 'Using network 'development'.' followed by a blank line and 'Hello, World!'. The prompt is now 'alarcos@HP-Z600-Workstation:~/Documentos/GitHub/ALBA-ethereum/HelloWorld\$' with a cursor.

```
alarcos@HP-Z600-Workstation:~/Documentos/GitHub/ALBA-ethereum/HelloWorld$ truffle exec scripts/miScript.js --network development
Using network 'development'.

Hello, World!
alarcos@HP-Z600-Workstation:~/Documentos/GitHub/ALBA-ethereum/HelloWorld$
```