Truffle

Que es:

Es un framework de desarrollo de ethereum que facilita a la hora de desarrollar, testear y desplegar los smart contracts.

FrameworK: conjunto estandarizado de conceptos, prácticas y criterios para enfocar un **tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.**

Truffle nos ofrece la posibilidad de:

- Compilación, enlace y despliegue de Smart contracts desde el propio framework
- Depuración y testing automatizado de contratos
- Framework con scripts de despliegue y migraciones en redes públicas y privadas
- Acceso a cientos de paquetes externos y gestión con EthPM & NPM
- Consola interactiva para comunicación directa con los contratos
- Interacción con contratos mediante scripts externos

Una capacidad de Truffle es desplegar nuestros SmartContrats de manera muy sencilla a distintos entornos. Se podrán modificar desde el truffle-config.js

Como instalar:

Descargaremos el siguiente elemento:

- https://nodejs.org/es/

Una vez que tenemos estas herramientas instaladas, instalaremos Truffle con el comando, desde la pantalla de CMD o desde el visual code pulsando (ctrl +ñ):

- npm install -g truffle

Si deseas instarlar una versión concreta será

- npm install -g truffle@y la versión que desees. -g (global)

Comprobamos que truffle se ha instalado correctamente ejecutando, ejecutando lo siguiente, también lo ejecutaremos desde CMD:

- truffle version

Y veremos que sale algo parecido a:

- Truffle v4.1.13 (core: 4.1.13)
- Solidity v0.4.24 (solc-js)
- Node v11.7.0

A continuación, abrimos una nueva terminal (cmd).

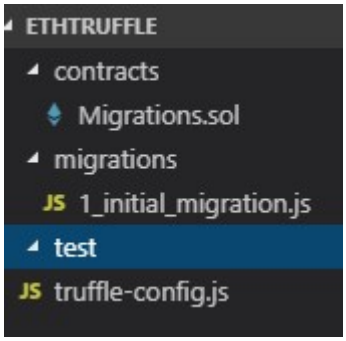
No colocamos mediante los comandos donde deseemos realizar las siguientes operaciones, y a continuación creamos una nueva carpeta con el comando mkdir y una vez creada nos colocamos dentro.

Ahora procedemos a iniciar nuestro proyecto:

- Una vez que estemos dentro de la carpeta creada con el mkdir introducimos: truffle init

```
C:\node\pruebas>truffle init

√ Preparing to download
√ Downloading
√ Cleaning up temporary files
√ Setting up box

Unbox successful. Sweet!

Commands:

  Compile:        truffle compile
  Migrate:        truffle migrate
  Test contracts: truffle test
```

Quedando algo por este estilo:



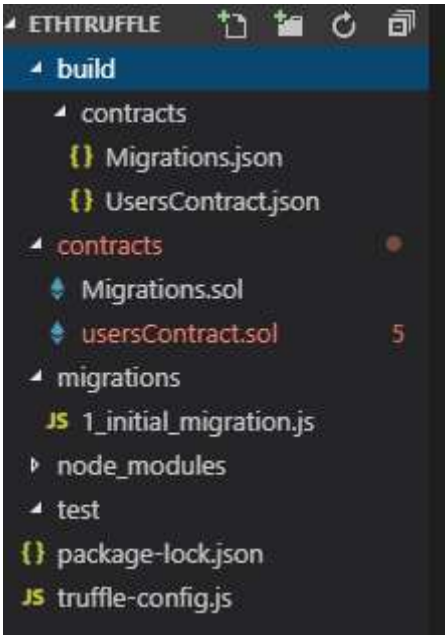
Compilación de smartContracts.

Realizamos un truffle compile

```
PS C:\Users\edu\Desktop\ethereum-pruebas\ethTruffle> truffle version
Truffle v4.1.13 (core: 4.1.13)
Solidity v0.4.24 (solc-js)
PS C:\Users\edu\Desktop\ethereum-pruebas\ethTruffle> truffle compile
(node:25356) V8: C:\Users\edu\AppData\Roaming\npm\node_modules\truffle\node_modules\solc\soljson.js:3 Invalid asm.js: Invalid member of stdlib
Compiling .\contracts\Migrations.sol...
Compiling .\contracts\usersContract.sol...
Writing artifacts to .\build\contracts
```

Esto nos creará una carpeta bild con las mismas extensiones pero con el formato JSON.

En caso de borrar la carpeta bild, si volvemos a realizar el compile nos la volverá a crear con los mismos archivos



A continuación, iremos a truffle-config.js y borraremos todo el contenido que tiene escrito.

Aquí declaramos los que va a exportar el módulo, será un objeto donde vamos a definir nuestras redes:

```
module.exports = {
  networks: {
    development: {
      host: 'localhost',
      port: 7545,
      network_id: '*',
      gas: 5000000 //limite de gas por trasaccion
    }
  }
}
```

Una vez cambiado y guardado esto.

Abriremos la Ganache y ejecutaremos el comando truffle deploy.

```
PS C:\Users\edu\Desktop\ethereum-pruebas\ethTruffle> truffle deploy
(node:4976) V8: C:\Users\edu\AppData\Roaming\npm\node_modules\truffle\node_modules\solc\soljson.js:3 Invalid asm.js: Invalid member of stdlib
Using network 'development'.

Network up to date.
```

Ahora iremos a la consola intereactiva de truffle

Iremos a truffle console –network development

Esta consola ya nos da la librería Web3.

Si indicamos el programa con el que estamos trabajando:

- Nos dará ayuda de la network en la que estamos trabajando o de donde viene el smartContrac y el host contra el que estamos trabajando (entre otras cosas).

Con el programa que trabajamos .address

- Nos indicará donde la dirección del smartContract que se encuentra desplegada en esos momentos en la aplicación (nuestro caso ganache)

Si deseamos saber las cuentas que tenemos activas en este momento (en Ganache) Ejecutamos lo siguiente:

```
truffle(development)> web3.eth.getAccounts(function(err, acc) {accounts = acc});
undefined
truffle(development)> accounts
[ '0xa02a21ff991b1e53be262eb1f6d77f26073b829a',
  '0xb793a001d0a9d96903d75f8ee831b95aa82a0c8f',
  '0xc320465ceec24dc75a057c9fd39b44d34a66bd46',
  '0xfcbbe6d4146a64dea3a929ae1d781950241233cf',
  '0x964b11ec3ace5dfb436f5dd12e40c8fb849f2cda',
  '0x6f90da4dada08cc7d00e5036121563f8f5836230',
  '0x41faee24886e7cff84b7d50267ec079f60c98680',
  '0x7324f5f825d6e582be1b8930203f142c0541af19',
  '0x5b4d702ec7422809e6c1efe61cfa046585a2321a',
  '0x872da0412101943ca5c2ec9cc0dfd3ab866d2123' ]
truffle(development)>
```

Para saber el balance de una cuenta usaremos:

```
truffle(development)> web3.eth.getBalance(account[1]);
BigNumber { s: 1, e: 19, c: [ 999947, 92980000000000 ] }
truffle(development)> web3.eth.getBalance(account[1]);
BigNumber { s: 1, e: 19, c: [ 999947, 92980000000000 ] }
truffle(development)> web3.eth.getBalance(account[2]).toNumber();
100000000000000000000
truffle(development)>
```

Nos lo indicará en numero grande pero llamando a toNumber() nos lo pondrá en número normal.

Podemos observar que la cuenta 1 a sido usada mientras que la cuenta dos aun no se ha llegado a usar

Así sacaríamos el número de ether que le queda en la cuenta a ese usuario

```
truffle(development)> web3.fromWei(web3.eth.getBalance(account[1]).toNumber(), 'ether');
'99.99479298'
```

Ahora enviaremos ether de un usuario a otro:

Indicando el usuario inicial, el usuario final, y el valor que queremos transferir

El resultado será la cuenta a la que hemos pasado en este caso los 15 de ether

```
truffle(development)> web3.eth.sendTransaction({from: account[0], to: accounts[1], value: web3.toWei(15, 'ether')});
'0x7fb23845e0e470eb96cd03af0b8a1b626f61906cc675fcd2e7ff1d59e99d289b'
truffle(development)>
```

← BACK

TX 0×7fb23845e0e470eb96cd03af0b8a1b626f61906cc675fcd2e7ff1d59e99d289b

SENDER ADDRESS	TO CONTRACT ADDRESS				
0×A02a21fF991B1e53Be262eB1F6D77f26073B829A	0×B793a001d0a9d96903D75F8Ee831b95aa82a0C8F	VALUE TRANSFER			
VALUE	GAS USED	GAS PRICE	GAS LIMIT	MINED IN BLOCK	
15.00 ETH	21000	20000000000	90000	13	
TX DATA					
0x					

Comprobamos que la cuenta 0xA02…… tienen 15 unidades menos de ether.

Comprobamos que la cuenta 0xB79… tiene 15 unidades más de ether.

```
truffle(development)> web3.eth.getBalance(accounts[0]).toNumber();
84827745200000000000
truffle(development)> web3.eth.getBalance(accounts[1]).toNumber();
115000000000000000000
truffle(development)>
```

Para enviar transacciones emplaremos el siguiente comando:

```
truffle(development)> web3.eth.getAccounts(function(e,d) {accounts = d});
undefined
truffle(development)> accounts
[ '0xa02a21ff991b1e53be262eb1f6d77f26073b829a',
  '0xb793a001d0a9d96903d75f8ee831b95aa82a0c8f',
  '0xc320465ceec24dc75a057c9fd39b44d34a66bd46',
  '0xfcbbe6d4146a64dea3a929ae1d781950241233cf',
  '0x964b11ec3ace5dfb436f5dd12e40c8fb849f2cda',
  '0x6f90da4dada08cc7d00e5036121563f8f5836230',
  '0x41faee24886e7cff84b7d50267ec079f60c98680',
  '0x7324f5f825d6e582be1b8930203f142c0541af19',
  '0x5b4d702ec7422809e6c1efe61cfa046585a2321a',
  '0x872da0412101943ca5c2ec9cc0dfd3ab866d2123' ]
truffle(development)> web3.eth.sendTransaction({from: accounts[1], to: accounts[2], value: web3.toWei(2,'ether')});
'0x75b86fc3601ef1a001e8b9017aa39de20d3a89ebd31699c18abe77a953f33963'
```

Y para comprobar que a sido bien recibida usaremos:

Con el digito que nos de del has del comando del sendTransaction realizado anteriormente.

```
truffle(development)> web3.eth.getTransactionReceipt('0x75b86fc3601ef1a001e8b9017aa39de20d3a89ebd31699c18abe77a953f33963')
```

Podremos ejecutar
web3.eth.getTransactionReceipt(‘valor en verde el comando realizado anteriormente’)

Nos fijaremos en el parámetro status si tien 0x1 (todo correcto) y si por el contrario es 0x0 (hubo algún error en el proceso del ether.