**Contratos en Raspichain**

**Compilar contratos**

Para interactuar con los contratos, necesitamos instalar "*solc*" en Windows o Linux, lo que permite compilarlos y generar los archivos *.abi* y *.bin*.

**Linux**

Escribimos lo siguiente en la consola para instalar *solc*:

*$ sudo add-apt-repository ppa:ethereum/ethereum*

*$ sudo apt-get update*

*$ sudo apt-get install solc*

Tras crear el archivo con el contrato, por ej, *Greeter.sol*, escribimos lo siguiente para compilarlo:

*$ solc -o target --bin --abi Greeter.sol*

Así tendremos los archivos necesarios para crear un script.

**Windows**

Descargamos el binario de la siguiente dirección:

https://github.com/ethereum/solidity/releases

Lo extraemos en una carpeta, en la cual metemos el archivo *.sol*, y desde la consola vamos a dicha dirección y escribimos:

*solc -o target --bin --abi Greeter.sol*

Se creará una carpeta con los archivos *.abi* y .*bin*.

**Scripts para desplegar contratos**

A la hora de desplegar un contrato en la cadena, la forma más sencilla es ejecutando un script con los resultados de la compilación anterior realizada anteriormente. Para ello, en un archivo de texto escribiríamos lo siguiente (tomando como ejemplo el contrato *Trivial.sol*):

*var trivialFactory = eth.contract([{"constant":true,"inputs":[],"name":"getValue","outputs":[{"name":"","type":"uint256"}],"payable":false,"stateMutability":"view","type":"function"}])*

*var trivialCompiled = "0x" + "6080604052348015600f57600080fd5b5060a18061001e6000396000f300608060405260043610603f576000357c0100000000000000000000000000000000000000000000000000000000900463ffffffff16806320965255146044575b600080fd5b348015604f57600080fd5b506056606c565b6040518082815260200191505060405180910390f35b6000604d9050905600a165627a7a723058201ffe4ff707172fe52349718a3d74ae1bd48db237bd643ae25cfe38dc9c44443b0029"*

*var trivial = trivialFactory.new({from:eth.accounts[0],data:trivialCompiled,gas:1000000}, function(e, contract){*

*if(e) {*

*console.error(e);*

*return;*

*}*

*if(!contract.address) {*

*console.log("Contract transaction send: TransactionHash: " + contract.transactionHash + " waiting to be mined...");*

*} else {*

*console.log("Contract mined! Address: " + contract.address);*

*console.log(contract);*

*}*

*})*

Este es el script para inicializar uno de los contratos que hemos creado. En primer lugar, creamos una variable que contendrá el código del archivo *.abi* al completo. El siguiente paso es crear una nueva variable que contendrá el código compilado del contrato en hexadecimal que encontramos en el archivo *.bin*. Por último, tenemos la función que permite el despliegue y que nos avisará cuando se haya minado la dicha operación, mostrando además la dirección del contrato en la cadena.

**Manejo de contratos**

1) **Trivial**

Address: 0x6d53783be2ef1ca75b3303bea7880c77617830ae

*> personal.unlockAccount(eth.coinbase, "")*

*> loadScript("trivial.js")*

*> eth.getCode(trivial.address)*

*> trivial.getValue()*

2) **Greeter**

Address: 0xfbaea1317a5fef6dedeec53fcc1c95c49fdb6d2c

*> personal.unlockAccount(eth.coinbase, "")*

*> loadScript("greeter.js")*

*> eth.getCode(greeter.address)*

*> greeter.greet()*

3) **Counter**

Address: 0x058efb55c34900e0fd75220f39a471f97d9b0f5d

*> personal.unlockAccount(eth.coinbase, "")*

*> loadScript("counter.js")*

*> eth.getCode(counter.address)*

*> counter.getCount()*

*> counter.increment()*

*> counter.decrement()*

*> counter.incAmount(400)*

*> counter.decAmount(100)*

*> counter.getCount()*

Si se produce fallo de dirección al invocar a las funciones que alteran el estado de la variable escribiremos:

*> eth.defaultAccount=eth.accounts[0]*

Para interactuar con contratos desplegados en la blockchain necesitamos el abi del contrato y su dirección. De esta forma, creando una instancia del contrato, podemos usar sus funciones desde los otros nodos. Para crear una instancia escribiremos lo siguiente desde el nodo desde el cual queremos ejecutarlo:

*var contractInstance = web3.eth.contract(ContractAbi).at(contractAddress)*

Ej: desplegamos el contrato *Trivial* desde raspi1, conseguimos la dirección del contrato y escribimos en raspi2:

*> var trivDemo = web3.eth.contract([{"constant":true,"inputs":[],"name":"getValue","outputs":[{"name":"","type":"uint256"}],"payable":false,"stateMutability":"view","type":"function"}]).at('*0*x6d53783be2ef1ca75b3303bea7880c77617830ae')*

*> trivDemo.getValue()*

Para el resto de contratos:

*> var greetDemo = web3.eth.contract([{"constant":false,"inputs":[],"name":"kill","outputs":[],"payable":false,"stateMutability":"nonpayable","type":"function"},{"constant":true,"inputs":[],"name":"greet","outputs":[{"name":"","type":"string"}],"payable":false,"stateMutability":"view","type":"function"},{"inputs":[{"name":"\_greeting","type":"string"}],"payable":false,"stateMutability":"nonpayable","type":"constructor"}]).at('0xfbaea1317a5fef6dedeec53fcc1c95c49fdb6d2c')*

*> greetDemo.greet()*

*> var countDemo = web3.eth.contract([{"constant":false,"inputs":[],"name":"decrement","outputs":[],"payable":false,"stateMutability":"nonpayable","type":"function"},{"constant":true,"inputs":[],"name":"getCount","outputs":[{"name":"","type":"uint256"}],"payable":false,"stateMutability":"view","type":"function"},{"constant":false,"inputs":[{"name":"amount","type":"uint256"}],"name":"incAmount","outputs":[],"payable":false,"stateMutability":"nonpayable","type":"function"},{"constant":false,"inputs":[],"name":"increment","outputs":[],"payable":false,"stateMutability":"nonpayable","type":"function"},{"constant":false,"inputs":[{"name":"amount","type":"uint256"}],"name":"decAmount","outputs":[],"payable":false,"stateMutability":"nonpayable","type":"function"},{"inputs":[{"name":"\_count","type":"uint256"}],"payable":false,"stateMutability":"nonpayable","type":"constructor"}]).at('0x058efb55c34900e0fd75220f39a471f97d9b0f5d')*

*> countDemo.getCount()*

Las siguientes funciones no son sólo de consulta, ya que alteran la cadena, por lo que para ejecutarlas debemos introducir la contraseña de la cuenta personal e indicar, dentro de ella, la cuenta de ether sobre la que vamos a trabajar (en el caso de que tengamos sólo una cuenta, simplemente usaremos esa) para que, al ejecutar las funciones, pueda sustraerse de ella el ether que cuesta realizar la transacción:

*> personal.unlockAccount(eth.coinbase, "")*

*> eth.defaultAccount=eth.accounts[0]*

*> countDemo.increment()*

*> countDemo.decrement()*

*> countDemo.incAmount(400)*

*> countDemo.decAmount(100)*