

Coloreando tu propia criptodivisa con Ethereum.

Pedro Bonilla Nadal

Primeros Pasos

Instalación

Antes de empezar con la implementación, hemos de tener instalada la línea de comandos de ethereum. Puedes encontrar una guía de instalación en inglés y más completa aquí (pincha).

Hay tres implementaciones fundamentales hechas para ethereum: Geth, Eth y Pyethapp. La primera de esta se implemento en GO, la segunda en C++ y la tercera en Python, tres de los lenguajes con más fuerza de los últimos tiempos. Existen otras implementaciones de ethereum hechas en lenguajes como RUST, Haskell, javascript entre otras muchas. Nosotros nos centraremos en la primeras. Aun así, explicaremos como instalar las dos primeras, pues se recomienda usar varias implementaciones por razones de fiabilidad de la aplicación instalada.

Geth

Para instalar geth en **Linux** escriba en su terminal:

```
sudo apt-get install software-properties-common
sudo add-apt-repository -y ppa:ethereum/ethereum
sudo apt-get update
sudo apt-get install ethereum
```

Eth

Para instalar eth en **Linux** escriba en su terminal:

```
sudo apt-get install cpp-ethereum
```

Ejecutandolo

Tanto Geth como Eth son cli multiproposito que ejecutan un modo completo de ethereum. A pesar de ofrecer múltiples interfaces, como un servidor JSONRPC, nosotros nos centraremos en la consola interactiva. Para abrir la consola de Geth escribimos:

```
geth console
```

Conectandose a un area privada para test

Como podremos imaginar, en nuestros primeros pasos, para hacer programas (o contratos) simples no nos interesa estar en la red general de ethereum.

Como eres el único miembro de la red eres el responsable de encontrar todos los bloques, validar todas las transacciones y ejecutar todos los contratos. También podrías hacer trampas y hacerte a ti mismo un ataque del 51%, sin mucho beneficio de todos modos. Esto hará el desarrollo más fácil (y barato), dado que tienes la flexibilidad de controlar tu propia cadena de bloques. Para generarla:

```
geth --datadir ~/.ethereum_private init ~/dev/genesis.json

geth --fast --cache 512 --ipcpath ~/Library/Ethereum/geth.ipc
--networkid <id-network> --datadir ~/.ethereum_private console
```

Para usuarios precavidos y responsables es conveniente cambiar el bloque de inicio, para que no se considere un fork “ilegal” de la cadena principal (que puede ser tomada por defecto). Aquí te explican cómo. Nosotros no lo explicaremos por motivos de tiempo.

Por último, para poner un minero local ejecutamos:

```
geth <usual-flags> --mine --minerthreads=1
etherbase=0x0000000000000000000000000000000000000000000000000000000000000000
```

Ahora tomaremos una serie de medidas para evitar que otro usuario que no conozca tu nonce, network-id, y genesis file se conecte a tu red. Para esto primero encontraremos la URL de tu nodo con:

```
admin.nodeInfo.NodeUrl
```

Para que otros usuarios se unan diles que usen el comando:

```
admin.addPeer(<tu_id>)
```

Creando cuentas

Para hacer una cuenta nueva escribimos el comando:

```
personal.newAccount("Una frase de contraseña es mejor que una palabra")
```

Para revisar las cuentas creadas usamos:

```
web3.eth.accounts
```

Por último, para mirar el ahorro de una cuenta. Para ello haremos uso de una variable y miraremos la cuenta de índice *i*.

```
var primaryAccount = web3.eth.accounts[i]  
web3.eth.getBalance(primaryAccount)
```

Para revisar todos los balances podemos utilizar esta función, la cual además es muy entretenida de leer e ir comprendiendo poco a poco.

```
function checkAllBalances() {  
  web3.eth.getAccounts(function(err, accounts) {  
    accounts.forEach(function(id) {  
      web3.eth.getBalance(id, function(err, balance) {  
        console.log("" + id + ":\tbalance: " + web3.fromWei(balance, "ether") + " ether");  
      });  
    });  
  });  
};
```

Contratos y transacciones

Ahora tenemos que entender como hacemos un envío simple de ether. Solo consistiría en escribir:

```
eth.sendTransaction({from: 'cuenta1', to: 'cuenta2', value: web3.toWei(1, "moneda")})
```

Contrato “Hola mundo”

Vamos a escribir ahora el ejemplo clásico de todo lenguaje de programación para que se pueda saber la base de como hacer un contrato inteligente, con la funcionalidad mínima de escribir por pantalla “hola mundo”. Como bien sabemos por la memoria, con un contrato se puede hacer basicamente cualquier cosa, por el lenguaje turing completo.