ETSIT UPM

dit UPM

Blockchain: Desarrollo de Aplicaciones

Trabajar en el Nodo Geth

BCDA 2018

Versión: 2018-10-03

Clientes Disponibles

- En la documentación de Ethereum:

  `http://www.ethdocs.org/en/latest/ethereum-clients/index.html`

  - Geth
  - cpp_ethereum
  - parity
  - ...

- Vamos a utilizar Geth
  - Implementación en go (Conocido como cliente go-ethereum).

- Clientes ligeros para el desarrollo: Ganache.

Detalles

- Instalación de Geth:
  - macOS (usando HomeBrew):

```
$ brew update
$ brew upgrade
$ brew tap ethereum/ethereum
$ brew install ethereum
```

  - macOS (descargarse binarios):

```
https://geth.ethereum.org/downloads/
```

  - Ubuntu:

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

  - Windows:

Geth

- Ayuda:

```
$ geth --help
```

- Ejemplo: Lanzar un nodo y su consola conectado a la red Ropsten:

```
$ geth --testnet console
```

  - Conviene redirigir la salida de error a otro tty o fichero.

- El nodo proporciona un entorno de ejecución javascript interactivo (consola) o no interactivo (scripts).

  https://github.com/ethereum/go-ethereum/wiki/JavaScript-Console

- APIs disponibles:
  - web3 javascript Dapp, admin , net, personal, ...

  https://github.com/ethereum/wiki/wiki/JavaScript-API

  https://github.com/ethereum/go-ethereum/wiki/JavaScript-Console

  https://github.com/ethereum/go-ethereum/wiki/Management-APIs

Ejemplos: Crear Cuenta

- Desde la consola Geth:
  - Crear una cuenta desde la consola Geth:

```
> personal.newAccount("12345678")
"0x4f3cae2df5a8a1062e483d108ffe0ea495786a60"
```

  - Consultar las cuentas existentes:

```
> eth.accounts
["0x795ff0852b2f79bf4652f29cf65bd192d3252aa2",
"0x4f3cae2df5a8a1062e483d108ffe0ea495786a60",
"0x08392b6f8100100920978 2ddf0e09dd0e08651ac"]
```

  - Desbloquear una cuenta:

```
> var addr = eth.accounts[0]
> personal.unlockAccount(addr, "1234", 3600)
> personal.unlockAccount(addr)
Unlock account
0x795ff0852b2f79bf4652f29cf65bd192d3252aa2
Passphrase:
```

- Consultar el balance de una cuenta:

```
> var addr = eth.accounts[0]
> eth.getBalance(addr)
2.65890625e+21
> web3.fromWei(eth.getBalance(addr),"ether")
2658.90625
```

- Desde la linea de comandos:
  - Crear una cuenta:

```
$ geth account new
Passphrase:
Repeat passphrase:
Address: {08392b6f81001009209782ddf0e09dd0e08651ac}
$ geth --password <file> account new
```

  - Consultar las cuentas existentes:

```
$ geth account list
```

  - Desbloquear una cuenta:

```
$ geth --unlock 0
Passphrase:
```

Ejemplos: peers

- Consultar si se están escuchando las peticiones de conexión de otros nodos.

```
> net.listening
true
```

- Consultar el número de nodos con los que está conectado el nodo

```
> net.peerCount
2
```

- Consultar datos sobre los nodos a los que está conectado el nodo:

```
> admin.peers
[ {
  ID: 'a4de274d3a...60514ab336d4acdc312db671b',
    Name: 'Geth/v0.9.14/linux/go1.4.2',
    Caps: 'eth/60',
    RemoteAddress: '5.9.150.40:30301',
    LocalAddress: '192.168.0.28:39219'
 },{
    ID: 'f4642fa...e18a8ecba66fbaf6416c0',
    Name: '++eth/Zeppelin/Rascal/v0.9.14/Re/Darwin/clang/int',
    Caps: 'eth/60, shh/2',
    RemoteAddress: '129.16.191.64:30303',
    LocalAddress: '192.168.0.28:39705'
  } ]
```

- Consultar datos del propio nodo:

```
> admin.nodeInfo
{
  Name: 'Geth/v0.9.14/darwin/go1.4.2',
  NodeUrl: 'enode://3414c0...d049dbf4c17694@[::]:30303',
  NodeID: '3414c0...a2d049dbf4c17694',
  IP: '::',
  DiscPort: 30303,
  TCPPort: 30303,
  Td: '2044952618444',
  ListenAddr: '[::]:30303'
}
```

Ejemplos: Bloques y Tx

- Consultar el estado de la sincronización:

```
> eth.syncing
{
  currentBlock: 0,
  highestBlock: 281668,
  knownStates: 0,
  pulledStates: 0,
  startingBlock: 0
}
```

- Arrancar y parar la minería:

```
> miner.start(2)
true
> miner.stop()
true
```

- Estado de las transacciones:

```
> txpool.status
{
  pending: 0,
  queued: 0
}
```

- Número de transacciones pendientes:

```
> eth.getBlockTransactionCount("pending")
0
```

- Ver todas las transacciones pendientes:

```
> eth.getBlock("pending", true).transactions
[]
```

- Consultar una transacción:

```
> eth.getTransaction("0xbd3167258370e20a...84b6ef890e9")
{
  blockHash:
"0xb33515e5afbbd7e2fca8d6e7b1bf3f672bb7bed57b7acd86bbb06a39c9e86212",
  blockNumber: 5366,
  from: "0xd40c37307364501926ef7833a7bb022ee031f45b",
  gas: 500000,
  gasPrice: 20000000000,
  hash:
"0xbd3167258370e20af2fc9bbe5a21205784ab1c40e6a5039aef99584b6ef890e9",
  input: "0x60606 . . . 000000",
  nonce: 1,
  r:
"0x3b049ae35e83fb8d7d5cb7bd093bdde2daca869ad0f8391fdbc226d3f07f5c9b",
  s:
"0x66e5b3c617ff17f20c802c61352f002b51bcf8af08e49364dda0242292ecaa17",
  to: null,
  transactionIndex: 0,
  v: "0x29",
  value: 0
}
```

- Consultar una Transaction Receipt:

```
> eth.getTransactionReceipt("0xb9defa...84621c79f8928f3")
{
  blockHash:
"0xb33515e5afbbd7e2fca8d6e7b1bf3f672bb7bed57b7acd86bbb06a39c9e86212",
  blockNumber: 5366,
  contractAddress: "0x419cdb13175808c816ea9547359c758d3409cd9e",
  cumulativeGasUsed: 464566,
  from: "0xd40c37307364501926ef7833a7bb022ee031f45b",
  gasUsed: 464566,
  logs: [],
  logsBloom: "0x00000 . . . 0000",
  root:
"0xa24e7b029e319ce9716ff401c06d811828c30400130b39b587b05cb302993201",
  to: null,
  transactionHash:
"0xbd3167258370e20af2fc9bbe5a21205784ab1c40e6a5039aef99584b6ef890e9",
  transactionIndex: 0
}
```

Compilador de Solidity

- El compilador de solidity es solc.
- Consultar forma de uso:

```
$ solc --help
solc, the Solidity commandline compiler.
Usage: solc [options] [input_file...]
Example:
    solc --bin -o /tmp/solcoutput contract.sol

Allowed options:
  --help                Show help message and exit.
  --version             Show version and exit.
  -o [ --output-dir ] path

Output Components:
  --bin                 Binary of the contracts in hex.
  --abi                 ABI specification of the contracts.
  --hashes              Function signature hashes of the contracts.
```

- Compilar el contrato Contador.sol desde linea de comandos:

```
$ solc -o salida --abi --bin --hashes Contador.sol
$ ls salida
Contador.abi    Contador.bin    Contador.signatures
$ cat salida/Contador.abi
[{"constant":false,"inputs":[],"name":"incr","outputs":
[],"payable":false,"stateMutability":"nonpayable","type":"function"}
,{"constant":true,"inputs":[],"name":"valor","outputs":
[{"name":"","type":"uint8"}],"payable":false,"stateMutability":"view
","type":"function"},
{"payable":false,"stateMutability":"nonpayable","type":"fallback"},
{"anonymous":false,"inputs":
[{"indexed":false,"name":"msg","type":"string"},
{"indexed":false,"name":"out","type":"uint8"}],"name":"Tic","type":"
event"}]
$ cat salida/Contador.bin
608060405260008060006101000a81548160ff021916908360ff1602179055503480
1561002b57600080fd5b5061019b8061003b6000396000f3006080604052600436106
61004c576000357c0100000000000000000000000000000000  ...etc...
2001828103825260008152602001807f41637475616c697a61646f0000000000000
0000000000000000000000000000000815250602001925050506040518091039a1565b
60008090549061010a900460ff16815600a165627a7a723058206f3f3f1b3275d1
f83c6ecff99a4a971f0af10f7532a6c651cb8d6c013b9a89fc0029
$ cat salida/Contador.signatures
119fbbd4: incr()
ecbac7cf: valor()
```

- Compilar el contrato Contador.sol desde la consola de Geth:

```
$ geth OTRAS_OPCIONES --solc /usr/local/bin/solc console
> var source = "pragma solidity ^0.4.4; contract { ... etc ...}"
> var contract = eth.compile.solidity(source)
> contract
{
  Contador: {
    code: "0x60806040526000...etc...89fc0029",
    info: {
      abiDefinition: [... etc ...],
      compilerOptions: "bin,abi, ... etc...",
      compilerVersion: "0.4.25",
      developerDoc: {
        methods: {}
      },
      language: "Solidity",
      languageVersion: "0.4.25",
      source: "pragma solidity ^0.4.25; contract Counter { ... etc ...}",
      userDoc: {
        methods: {}
      }
    }
  }
}
```

Desplegar el Contrato

- Lanzamos el nodo:

```
$ geth --testnet --unlock 0 console
```

- Creamos una variable con el contenido del ABI:

```
> var abi = [{"constant":false,"inputs":
[],"name":"incr","outputs":
[],"payable":false,"stateMutability":"nonpayable","type":"fun
ction"},{"constant":true,"inputs":
[],"name":"valor","outputs":
[{"name":"","type":"uint8"}],"payable":false,"stateMutability
":"view","type":"function"},
{"payable":false,"stateMutability":"nonpayable","type":"fallb
ack"},{"anonymous":false,"inputs":
[{"indexed":false,"name":"msg","type":"string"},
{"indexed":false,"name":"out","type":"uint8"}],"name":"Tic","
type":"event"}];
undefined
```

- Creamos una variable con el bytecode del contrato:

```
> var code =
"0x6080604052600080600061010000a81548160ff021916908360ff160217
90555034801561002b57600080fd5b5061019b8061003b6000396000f3006
0806040526004361061004c576000357c0100000000000000000000000000
0000000000000000000000000000900463ffffffff168063119fbbd4146
1005e578063ecbac7cf14610075575b34801561005857600080fd5b506000
80fd5b34801561006a57600080fd5b506100736100a6565b005b348015610
08157600080fd5b5061008a61015d565b604051808260ff1660ff16815260
200191505060405180910390f35b60008081819054906101000a900460ff1
68092919060010191906101000a81548160ff021916908360ff1602179055
50507f278733a8e0534f74d81486a11876429bb0d35a6968fa576ec403ad7
aecfa2e6e60008090549061010000a900460ff166040518080602001836 0ff
1660ff16815260200182810382526000b8152602001807f41637475616c697
a61646f000000000000000000000000000000000000000081525060200 1
92505050604051809103 90a1565b60008090549061010000a900460ff16815
600a165627a7a723058206f3f3f1b3275d1f83c6ecff99a4a971f0af10f75
32a6c651cb8d6c013b9a89fc0029";
undefined
```

- Guardo la dirección desde la que voy a desplegar en una variable:

```
> var primaryAddress = eth.accounts[0];
undefined
```

- Estimo cuanto gas necesitaré para desplegar el contrato:

```
> eth.estimateGas({from: primaryAddress,
data:code});
167308
```

- Construimos un objeto que representa nuestro contrato (de acuerdo al ABI que estamos usando):

```
> var MyContract = eth.contract(abi);
undefined
```

- Me pongo a minar:

```
> miner.start(2);
null
```

- Creo una transacción para desplegar una nueva instancia del contrato en la red:

```
> var contractInstance = MyContract.new({from: primaryAddress,
data: code, gas: 500000});
undefined
> contractInstance
{
  abi: [{
    constant: false,
    inputs: [],
    name: "incr",
    outputs: [],
    payable: false,
    stateMutability: "nonpayable",
    type: "function"
  }, {
    constant: true,
    inputs: [],
    name: "valor",
    outputs: [{...}],
    payable: false,
    stateMutability: "view",
    type: "function"
  }, {
    payable: false,
    stateMutability: "nonpayable",
    type: "fallback"
  }, {
    anonymous: false,
    inputs: [{...}, {...}],
    name: "Tic",
    type: "event"
  }],
  address: undefined,
  transactionHash:
"0x37f9bec2cebeb5adc3e9655f4a851fd74e68c431e5e11971da5dafa7ccabb
bc5"
}
```

- La transaccion creada es:

```
> eth.getTransaction("0x37f9bec2cebeb5adc3e9655f4a851fd74e68c431e5e11971da5dafa7ccabbbc5")
{
  blockHash: "0x0000000000000000000000000000000000000000000000000000000000000000",
  blockNumber: null,
  from: "0x795ff0852b2f79bf4652f29cf65bd192d3252aa2",
  gas: 500000,
  gasPrice: 18000000000,
  hash: "0x37f9bec2cebeb5adc3e9655f4a851fd74e68c431e5e11971da5dafa7ccabbbc5",
  input: "0x6080604 ...etc ...a89fc0029",
  nonce: 6,
  r: "0xc3c3f25d50f56e6083b07c4640cd7adaa1cb0364a12c5b9092c959b322fbfc80",
  s: "0x1b5164ff72ab73971fee554517e2c81d19595bec616dd46b625e0e30b9a3653d",
  to: null,
  transactionIndex: 0,
  v: "0x29",
  value: 0
}
```

- Espero hasta que se mine la transacción:

```
> txpool.status;
{
  pending: 0,
  queued: 0
}
```

- La transacción ya se ha minado:

```
> eth.getTransaction("0x37f9bec2cebeb5adc3e9655f4a851fd74e68c431e5e11971da5dafa7ccabbbc5")
{
  blockHash: "0x4b81503604b9b5fb43338112177bc2c1eb03270403175832d5ec5c0a53ac5853",
  blockNumber: 525,
  from: "0x795ff0852b2f79bf4652f29cf65bd192d3252aa2",
  gas: 500000,
  gasPrice: 18000000000,
  hash: "0x37f9bec2cebeb5adc3e9655f4a851fd74e68c431e5e11971da5dafa7ccabbbc5",
  input: "0x6080604 ...etc ...a89fc0029",
  nonce: 6,
  r: "0xc3c3f25d50f56e6083b07c4640cd7adaa1cb0364a12c5b9092c959b322fbfc80",
  s: "0x1b5164ff72ab73971fee554517e2c81d19595bec616dd46b625e0e30b9a3653d",
  to: null,
  transactionIndex: 0,
  v: "0x29",
  value: 0
}
```

- Obtengo la dirección del contrato del transaction receipt:

```
>
eth.getTransactionReceipt("0x37f9bec2cebeb5adc3e9655f4a851fd74e68c431e5e11
971da5dafa7ccabbbc5")
{
  blockHash:
"0x4b81503604b9b5fb43338112177bc2c1eb03270403175832d5ec5c0a53ac5853",
  blockNumber: 525,
  contractAddress: "0xb4124b2c32d1ce4d61cc558b169510ee9ad9bf63",
  cumulativeGasUsed: 167308,
  from: "0x795ff0852b2f79bf4652f29cf65bd192d3252aa2",
  gasUsed: 167308,
  logs: [],
  logsBloom:
"0x000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000",
  root:
"0xdd6ceb3efb4b9d44b76ec6d0542db9d1d51366c029aecb8cf0f70179e0b1d68c",
  to: null,
  transactionHash:
"0x37f9bec2cebeb5adc3e9655f4a851fd74e68c431e5e11971da5dafa7ccabbbc5",
  transactionIndex: 0
}
```

- En la llamada a MyContract.new puede añadirse una callback que podemos usar para saber cuando se ha desplegado en contrato:

```
> var contractInstance = MyContract.new({from: primaryAddress,
                                         data: code,
                                         gas: 500000},
        function(err, contract) {
              if (!err && contract.address) {
                    console.log("Contrato desplegado en",
                                contract.address);
                }
            });
undefined
>
>
>  EN ALGUN MOMENTO DEL FUTURO:
>
> Contrato desplegado en 0x18adf6a1eb3b024c0822cb4468cecacd55b5af4d
```

Usar el Contrato

- Lanzamos el nodo:

```
$ geth --testnet console
```

- Ejecutamos:

```
> var abi = [{"constant":false,"inputs":
[],"name":"incr","outputs":
[],"payable":false,"stateMutability":"nonpayable","type":"fun
ction"},{"constant":true,"inputs":
[],"name":"valor","outputs":
[{"name":"","type":"uint8"}],"payable":false,"stateMutability
":"view","type":"function"},
{"payable":false,"stateMutability":"nonpayable","type":"fallb
ack"},{"anonymous":false,"inputs":
[{"indexed":false,"name":"msg","type":"string"},
{"indexed":false,"name":"out","type":"uint8"}],"name":"Tic","
type":"event"}];
undefined
```

```
> var MyContract = eth.contract(abi);
undefined
> var primaryAddress = eth.accounts[0];
undefined
> personal.unlockAccount(primaryAddress, "1234", 3600)
true
```

- Creamos una variable con la dirección donde se deplegó alguna instancia del contrato:

```
> var addr = "0xe5c52e5fa9bd9cd8e78010ef53131e754bffa144";
undefined
```

- Obtenemos la instancia del contrato desplegado en esa dirección:

```
> var contractInstance = MyContract.at(addr);
undefined
> contractInstance
{
  abi: [{ ...etc... }],
  address: "0xe5c52e5fa9bd9cd8e78010ef53131e754bffa144",
  transactionHash: null,
  Tic: function(),
  allEvents: function(),
  incr: function(),
  valor: function()
}
```

- Consultar el valor actual del contador:

```
> contractInstance.valor();
0
> contractInstance.valor.call();
0
```

- Un detalle que conviene destacar: podemos llamar a los métodos del contrato usando sendTransaction o llamando a call.
  - sendTransaction crea una transacción que deben ejecutar todos los nodos de la red, llegar a un consenso, y finalmente se guardará en la cadena de bloques junto con el cambio de estado que se haya realizado.
  - call se usa para que el método se ejecute de forma local, no se genera transacción, no se ejecuta nada en ningún otro nodo de la red, y no se cambia el estado.
  - También puede llamarse a los métodos del contrato sin usar explícitamente ni sendTransaction, ni call. Se elige automáticamente lo más adecuado según los calificadores (ej: constant) usados al definir el método.

- Incrementar el valor actual del contador

```
> contractInstance.incr.sendTransaction({from: primaryAddress, gas:
200000})
"0x4a4498ff18e0cb6013d25aa928b0a408fe537de6c23147036e061488c378c824"


> contractInstance.incr({from: primaryAddress, gas: 200000})
"0xb648c9a455f256fb06d55299230bf9af6832f0aeb513a94dabccdb7fe3a241a3"
```

- Si el método tiene parámetros, se colocan al principio:, tanto para call como para sendTransaction:

```
> contractInstance.incrDIFERENTE("hola",23,{from: primaryAddress,
gas: 200000})
```

```
> eth.getTransaction("0x4a4498ff18e0cb6013d25aa928b0a408fe537de6c23147036e061488c378c824")
{
  blockHash: "0xf6acdac9a8bf48b6b87cbd412555fe022486ca605d94cfe25711978dbe0f3c27",
  blockNumber: 100,
  from: "0x795ff0852b2f79bf4652f29cf65bd192d3252aa2",
  gas: 200000,
  gasPrice: 1000000000,
  hash: "0x4a4498ff18e0cb6013d25aa928b0a408fe537de6c23147036e061488c378c824",
  input: "0x119fbbd4",
  nonce: 2,
  r: "0x7a835d15316c58c05799954e452575acdfdf55d06767d6bc4c76d6310b7f125b",
  s: "0xc0fe509ed4d1ba696f9a3e9811da1429656ea1a77fbec324be376e793d6e6ab",
  to: "0xe5c52e5fa9bd9cd8e78010ef53131e754bffa144",
  transactionIndex: 0,
  v: "0x2a",
  value: 0
}
```

```
> eth.getTransactionReceipt("0x4a4498ff18e0cb6013d25aa928b0a408fe537de6c23147036e061488c378c824")
{
  blockHash: "0xf6acdac9a8bf48b6b87cbd412555fe022486ca605d94cfe25711978dbe0f3c27",
  blockNumber: 100,
  contractAddress: null,
  cumulativeGasUsed: 44087,
  from: "0x795ff0852b2f79bf4652f29cf65bd192d3252aa2",
  gasUsed: 44087,
  logs: [{
      address: "0xe5c52e5fa9bd9cd8e78010ef53131e754bffa144",
      blockHash: "0xf6acdac9a8bf48b6b87cbd412555fe022486ca605d94cfe25711978dbe0f3c27",
      blockNumber: 100,
      data: "0x0000000000000000000000000000000000000000000000000000000000000040000000000000000000000000000000000000000000000000000000000000001000000000000000000000000000000000000000000000000000000000000000b41637475616c697a61646f0000000000000000000000000000000000000000",
      logIndex: 0,
      removed: false,
      topics: ["0x278733a8e0534f74d81486a11876429bb0d35a6968fa576ec403ad7aecfa2e6e"],
      transactionHash: "0x4a4498ff18e0cb6013d25aa928b0a408fe537de6c23147036e061488c378c824",
      transactionIndex: 0
  }],
  logsBloom: "0x000000 muchos ceros 0000000",
  root: "0xf682e5d4cddae1a4505a6bff1f148326411f8375b7879c1b2e3a2122fa9036a8",
  to: "0xe5c52e5fa9bd9cd8e78010ef53131e754bffa144",
  transactionHash: "0x4a4498ff18e0cb6013d25aa928b0a408fe537de6c23147036e061488c378c824",
  transactionIndex: 0
}
```

- Consultar el valor actual del contador:

```
> contractInstance.valor();
1
> contractInstance.valor.call();
1
```

- El campo **logs** contiene la información de los eventos generados.
  - Lo podemos comprobar calculando en valor sha3 de la signatura de este evento, que debe coincidir con el valor del campo **topics**:

```
> web3.sha3("Tic(string,uint8)")
"0x278733a8e0534f74d81486a11876429bb0d35a69
68fa576ec403ad7aecfa2e6e"
```

  - Y los argumentos de este evento están contenidos en el campo **data**:

```
> web3.toHex("Actualizado")
"0x41637475616c697a61646f"
> web3.toHex(1)
"0x1"
```

- Registramos callbacks para que se ejecuten cuando se produzcan eventos:

```
> contractInstance.Tic(function(err, data) {
      console.log("Se ha producido un evento Tic:");
      if (err){
        console.log(err);
       } else {
        var msg = data.args.msg;
        var out = data.args.out;
        console.log(" * Msg =", msg);
        console.log(" * Out =", out.valueOf());
       }
  });
```

- Ahora invocamos a los métodos del contrato:

```
> contractInstance.incr.sendTransaction({from: primaryAddress,
"gas":200000 })
"0x7fd7fc92eb1a5861a91d3649023ee7978e345cc7a13d3e4bc0b1e84396c77f09
"
```

- Pasado un rato se generarán algunos eventos, se llamarán a las callbacks registradas, y veremos la siguiente salida:

```
Se ha producido un evento Tic:
 * Msg = Actualizado
 * Out = 4
```

Transferir Fondos

```
> var origen = "0x0c9f1b37cc9caa4199a2fb9df70cc7ab40588728";
> var destino  = "0xb29f4cb64ae82d634206a836b9d3dbe1f5b305b2";
> var amount = web3.toWei(0.001, "ether");

> eth.getBalance(origen);
1962496850826222080
> eth.getBalance(destino);
0

> personal.unlockAccount(origen, "mipassword", 300);
true

> eth.sendTransaction({from: origen, to: destino, value:
amount});
"0x390b5c56d71559cea6dd6f7f0743ed175db54b32f2869701f7a892b6b818
2061"
```

Pendientes

- Hablar de los modos de sincronización: full, fast,
  - light para cliente ligero.
- Poner ejemplos de observar los cambios en la cadena.

```
eth.filter("latest", function(err, block) {
algo(); });
eth.filter("pending", function(err, block)
{ algo(); });
```