¡Bienvenidos, desarrolladores de Blockchain!

Estas son herramientas para desarrolladores de las cadenas de bloques. Las herramientas de línea de comandos le permitirán conectar su servidor o ejecutar su aplicación en la cadena de bloques de Ethereum - o su propia cadena de bloque privada.

Clientes

Por motivos de seguridad, se crearon tres implementaciones independientes para Ethereum: Geth, Eth y Phyton. Los clientes tienen una funcionalidad casi idéntica, por lo que cualquiera que se elija se deja a elección personal al igual que la plataforma, el idioma y cuál es su uso previsto para la red.

Si se está construyendo un negocio que necesita tener las máximas garantías de disponibilidad para la red Ethereum, se recomienda ejecutar al menos una instancia de varios clientes para garantizar la fiabilidad. No obstante en este manual vamos a tratar con el cliente Geth.

Geth (Go-Ethereum)

Geth es la interfaz de línea de comandos para ejecutar un nodo Ethereum completo implementado en Go. Es el principal producto de la etapa Frontier, una de las cuatro fases marcadas en la hoja de ruta en Ethereum. Geth ha sido auditado por seguridad y será la base futura para el navegador Mist, orientado al usuario final. Así que si se tiene experiencia con el desarrollo web y se está interesado en construir portadas para aplicaciones descentralizadas, se debería experimentar con Geth.

Instalación en Mac

Instalación con Homebrew

La manera más fácil de instalar Go-Ethereum es usar Homebrew. Si no se tiene Homebrew, es necesario descargarlo e [instálelo primero](). Después hay que asegurarse de que esté actualizado:

```
brew update
brew upgrade
```

Luego hay que ejecutar los siguientes comandos para agregar el *tap* e instalar *geth*:

```
brew tap ethereum/ethereum
brew install ethereum
```

Se puede instalar la rama de desarrollo ejecutando *---devel*:

```
brew install ethereum --devel
```

Después de la instalación, hay que ejecutar *run geth account* para crear una cuenta en su nodo.

Ahora se debería poder ejecutar *geth* y conectarse a la red.

Hay que asegurarse de chequear las diferentes opciones y comandos con *geth --help*

Para ver opciones y parches, consulte: https://github.com/ethereum/homebrew-ethereum

Construyendo Geth desde el origen (cliente de línea de comandos)

Se debe clonar el repositorio a un directorio de su elección:

```
git clone https://github.com/ethereum/go-ethereum
```

Construir *geth* requiere el compilador Go, por lo que hay que instalarlo:

```
brew install go
```

Finalmente, se construye el programa *geth* usando el siguiente comando.

```
cd go-ethereum
make geth
```

Ahora puede ejecutar build/bin/geth para iniciar su nodo.

Instalación en Windows

Binarios: Descargar binarios estables

Todas las versiones de Geth están construidas y disponibles para su descarga en https://geth.ethereum.org/downloads/.

La página de descarga proporciona un instalador y un archivo de extensión .zip. El instalador pone geth en su ruta automáticamente. El archivo .zip contiene los archivos command.exe y se puede utilizar sin necesidad de instalación.

1. Descargar archivo zip
2. Extraer geth.exe desde el archivo .zip
3. Abrir una línea de comandos
4. Teclear “chdir” (sin comillas)
5. Abrir geth.exe

Fuente: Compilar geth con herramientas de chocolatey

El gestor de paquetes Chocolatey proporciona una forma fácil de instalar las herramientas de compilación necesarias. Si aún no se tiene chocolatey, es necesario seguir las instrucciones en https://chocolatey.org para instalarlo primero.

A continuación, se debe abrir un símbolo de comandos de Administrador e instalar las herramientas de compilación que necesitamos:

```
C:\Windows\system32> choco install git
C:\Windows\system32> choco install golang
C:\Windows\system32> choco install mingw
```

La instalación de estos paquetes configurará la variable de entorno *Path* (Ruta). Abra una nueva línea de comandos para obtener la nueva *Path* (Ruta). Los siguientes pasos no necesitan privilegios de administrador.

Hay que asegurarse de que la versión Go instalada sea 1.7 (o cualquier versión posterior).

Primero crearemos y configuraremos un diseño de directorio de espacio de trabajo Go, luego clonaremos el código fuente.

Observación: Si, durante los comandos de abajo, aparece el siguiente mensaje:

```
WARNING: The data being saved is truncated to 1024 characters.
```

Entonces eso significa que el comando *setx* fallará, y procediendo se truncará el *Path/GOPATH*. Si esto sucede, es mejor abortar e intentar hacer más espacio en la ruta *Path* antes de intentarlo de nuevo.

```
C:\Users\xxx> set "GOPATH=%USERPROFILE%"
C:\Users\xxx> set "Path=%USERPROFILE%\bin;%Path%"
C:\Users\xxx> setx GOPATH "%GOPATH%"
C:\Users\xxx> setx Path "%Path%"
C:\Users\xxx> mkdir src\github.com\ethereum
C:\Users\xxx> git clone https://github.com/ethereum/go-ethereum
src\github.com\ethereum\go-ethereum
C:\Users\xxx> cd src\github.com\ethereum\go-ethereum
C:\Users\xxx> go get -u -v golang.org/x/net/context
```

Finalmente, el comando para compilar *geth* es:

```
C:\Users\xxx\src\github.com\ethereum\go-ethereum> go install -v ./cmd/...
```

Instalación en Linux

Primero hay que descargar Geth para Linux.

Después, en Ubuntu, hay que ejecutar estos comandos:

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

Ejecución de Geth

Geth es una herramienta de línea de comandos multipropósito que ejecuta un nodo Ethereum completo. Ofrece múltiples interfaces: los subcomandos y opciones de la línea de comandos, un servidor JSON-RPC y una consola interactiva.

A los efectos de esta guía, nos centraremos en la consola, un entorno JavaScript que contiene todas las características principales que probablemente se deseen. Hay que copiar y pegar los siguientes comandos:

```
geth console
```

Luego, usar *geth attach*.

```
Geth attach
```

La primera vez que inicie la línea de comandos se presentará una licencia. Antes de poder utilizarlo, hay que aceptar esta licencia, después de leerla detenidamente.

ATENCIÓN: Si el objetivo de instalar Geth es la de probar la tecnología y jugar, NO SE DEBE UTILIZAR LA RED PRINCIPAL. Leer más información para saber cómo implementar una red privada de pruebas sin gastar Ether.

Conexión a una red de prueba privada (private testnet)

A veces es posible que no se quiera conectarse a la red pública en vivo, sino que se puede optar por crear una red privada de prueba. Esto es muy útil si no se necesita probar los contratos públicos y sólo se quiere probar o desarrollar la tecnología. Al crear esta red privada de prueba, el creador es el único responsable de encontrar todos los bloques, validar todas las transacciones y ejecutar todos los contratos inteligentes. Esto hace que el desarrollo sea más barato y fácil, ya que cabe la posibilidad de controlar de forma flexible la inclusión de transacciones en la propia cadena de bloques personal.

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

geth --fast --cache 512 --ipcpath ~/Library/Ethereum/geth.ipc
--networkid 1234 --datadir ~/.ethereum_private  console
```

Se debe sustituir “12345” por cualquier número aleatorio que se desee utilizar como identificador de red (networkID). Es una buena idea cambiar el contenido del bloque génesis (el primer bloque de la red) porque si alguien accidentalmente se conecta a esa red de prueba usando la cadena real, la copia local creada será considerada un fork rancio y será actualizado al fork “real". Cambiando el *datadir* también cambia esa copia local de la cadena de bloques, de lo contrario, para poder extraer un bloque exitosamente, se necesitaría minar contra la dificultad del último bloque presente en la copia local de la cadena de bloques - lo que puede tomar varias horas.

Si se desea crear una red privada se debería, por razones de seguridad, utilizar un bloque génesis diferente (una base de datos que contenga todas las transacciones de las ventas de Ether).

Estos comandos evitan que cualquier persona que no conozca el archivo secreto, identificador de red y bloque génesis, se conecte a esa nueva testnet y proporcione datos no deseados. En el caso de querer conectarse a otros compañeros y crear una pequeña red privada de múltiples ordenadores, todos necesitarán usar el mismo networkID y un bloque génesis idéntico. También tendrás que ayudar a cada nodo a encontrar los otros. Para ello, primero es necesario crear una URL de nodo propia:

```
admin.nodeInfo.NodeUrl
```

Lo cual devolverá la propia URL de nodo. Es necesario anotar esta URL y luego añadir en los otros clientes tu par ejecutando este comando:

```
admin.addPeer("YOURNODEURL")
```

No es necesario que se agreguen todos los clientes entre sí, ya que una vez conectados compartirán información sobre cualquier otro cliente con el que estén conectados.

Registros

Si se está ejecutando Geth posiblemente se notará que hay muchas entradas de registro apareciendo en la consola - a veces mientras se escribe. Esto se debe a que todas las advertencias e información de progreso son registradas en vivo en el terminal usado por el cliente. Si se desea guardar los registros en un archivo para verlos más adelante, se debe utilizar este comando:

```
geth console 2>>geth.log
```

Geth soporta múltiples ventanas de terminal y se puede iniciar una nueva con los registros en una y la consola en otra. Esto nos dará exactamente la misma funcionalidad que la consola original, pero sin el desorden. Para ello, es necesario abrir una nueva ventana de terminal y escribir:

```
geth attach
```

La consola tiene soporte de autocompletado e historial que persiste entre sesiones. Es posible completar un comando presionando la tecla de tabulación, geth entonces completará automáticamente la sentencia actual o mostrará una lista de las terminaciones disponibles cuando múltiples terminaciones son posibles. Se puede navegar por el historial de comandos utilizando las teclas de flecha arriba y abajo.

Más información sobre cómo ejecutar un nodo

Copia de seguridad y restauración
Conexión a la red

Ejemplos de uso

Creación de cuentas

Para poder hacer cualquier cosa en una red Ethereum se necesita Ether, y para conseguirlo, se necesita crear una cuenta. Hay varias maneras de hacer esto, pero la más sencilla es a través de la consola.

ATENCIÓN: Si se estaba ejecutando Ethereum durante la fase Olimpic o antes en el desarrollo, no se debe reutilizar las claves generadas antes de la liberación del software cliente Frontier 1.0, ya que de lo contrario podrían ser vulnerables a los ataques de repetición. Es conveniente realizar copias de seguridad de esas claves y crear otras nuevas utilizando los clientes de lanzamiento de Frontier.

```
personal.newAccount("Escribir aquí una buena y aleatoria frase de contraseña")
```

Nota: Una vez seleccionada la frase de contraseña se recomienda anótala. Si se pierde la contraseña que se utilizó para cifrar la cuenta, no se podrá acceder a ella. Repito: ¡No hay

redes de seguridad!. NO es posible acceder a la cuenta sin una frase de contraseña válida y no hay opción de "Olvidé mi contraseña" aquí.

** ¡No olvide su contraseña! **

Se pueden crear tantas cuentas como se desee. Por convención llamamos a la primera cuenta que se crea la cuenta principal. Se pueden ver todas las cuentas con el comando:

```
web3.eth.accounts
```

El orden de las cuentas refleja el momento de su creación. Los archivos de clave se almacenan en el directorio DATADIR/keystore y se pueden transferir entre clientes copiando los archivos contenidos en el interior. Los archivos están encriptados con su contraseña y deben ser respaldados si contienen cualquier cantidad de Ether. Sin embargo, hay que tener en cuenta que si se transfieren archivos de clave individuales, el orden de las cuentas presentadas puede cambiar y no puede terminar la misma cuenta en la misma posición. Por lo tanto, hay que tener en cuenta que confiar en el índice de la cuenta sólo se da mientras no se copien archivos de clave externos en el almacén de claves.

Obtener el saldo de cualquier cuenta

Todos los comandos de la consola están en JavaScript, por lo que se pueden crear variables y funciones en cadena. También se puede escribir cualquier función "eth" como "web3.eth" ya que forma parte del objeto "web3" principal.

Se puede intentar esto por ejemplo:

```
var primaryAccount = web3.eth.accounts[0]
```

A partir de ahora se tendría una variable llamada *PrimaryAccount* que puede utilizar en otras llamadas. Para obtener el saldo de cualquier cuenta, se debe utilizar la función *eth.getBalance*, así:

```
web3.eth.getBalance(primaryAccount)
```

El saldo debe devolver 0, ya que se acaba de crear. Para poder hacer los siguientes pasos, es necesario tener un poco de Ether en su cuenta para poder pagar los costes del Gas. En la siguiente sección aprenderás qué es el Gas y cómo puedes interactuar con la red.

Consultar todos los saldos a la vez

Las herramientas de línea de comandos son entornos JavaScript, lo que significa que se pueden crear funciones como lo haría en JavaScript. Por ejemplo, si se desea comprobar el saldo de todas las cuentas a la vez, se debe utilizar este fragmento de código JavaScript.

Se repetirá sobre cada una de las cuentas e imprimirá el saldo en Ether. Se puede utilizar el siguiente código:

```
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");
   });
  });
 });
};
```

Una vez ejecutada la línea anterior, todo lo que se necesita para comprobar todos los saldos es llamar a la siguiente función:

```
checkAllBalances()
```

Sugerencia: si se tienen muchos scripts pequeños y prácticos como este que son usados con frecuencia, se pueden guardar en un archivo para cargarlos todos a la vez usando loadScript:

```
loadScript('/some/script/here.js')
```