



MANUAL ETHEREUM

Ismael Rabanal Martín

INDICE

1.Instalación en Ubuntu.

2.Creación de nodos.

3.Consola en los nodos.

4.Dapps con truffle.

5.Remix IDE.

6.Ganache.

7.Contratos

8.Links de Interés

1.Instalación en Ubuntu

Para empezar instalamos Geth, la implementación en Go de Ethereum, para lo cual seguimos las instrucciones de la [página oficial](#), en mi caso lo he instalado vía PPAs.

Primero añadimos el repositorio:

```
sudo add-apt-repository -y ppa:ethereum/ethereum
```

Y después actualizamos e instalamos:

```
sudo apt-get update
```

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

Ahora instalamos el framework Truffle, igual que antes encontramos las instrucciones en su [página oficial](#).

El comando es el siguiente:

```
sudo npm install -g truffle
```

En caso de que no tengamos instalado npm tendremos que instalarlo previamente:

```
sudo apt install npm
```

Con la instalación del framework Truffle también se instala Solidity que es el lenguaje de programación utilizado para los contratos, además de Web3.js y node.

```
gsya@gsya-VirtualBox:~$ truffle version
You can improve web3's performance when running Node.js versions older than 10.5
.0 by installing the (deprecated) scrypt package in your project
Truffle v5.1.24 (core: 5.1.24)
Solidity v0.5.16 (solc-js)
Node v8.10.0
Web3.js v1.2.1
```

2.Creación de Nodos

Utilizando Geth podemos tanto generar un nodo para conectarnos a alguna de las redes Ethereum existentes como crear una red propia.

2.1.Red existente.

Geth nos proporciona opciones para distintas redes ya existentes, ya sea la red principal denominada Ethereum o las [redes de testeo](#) como Rinkeby, Kovan, Ropsten...

En Rinkeby puedes solicitar fondos de manera gratuita para llevar a cabo tus pruebas.

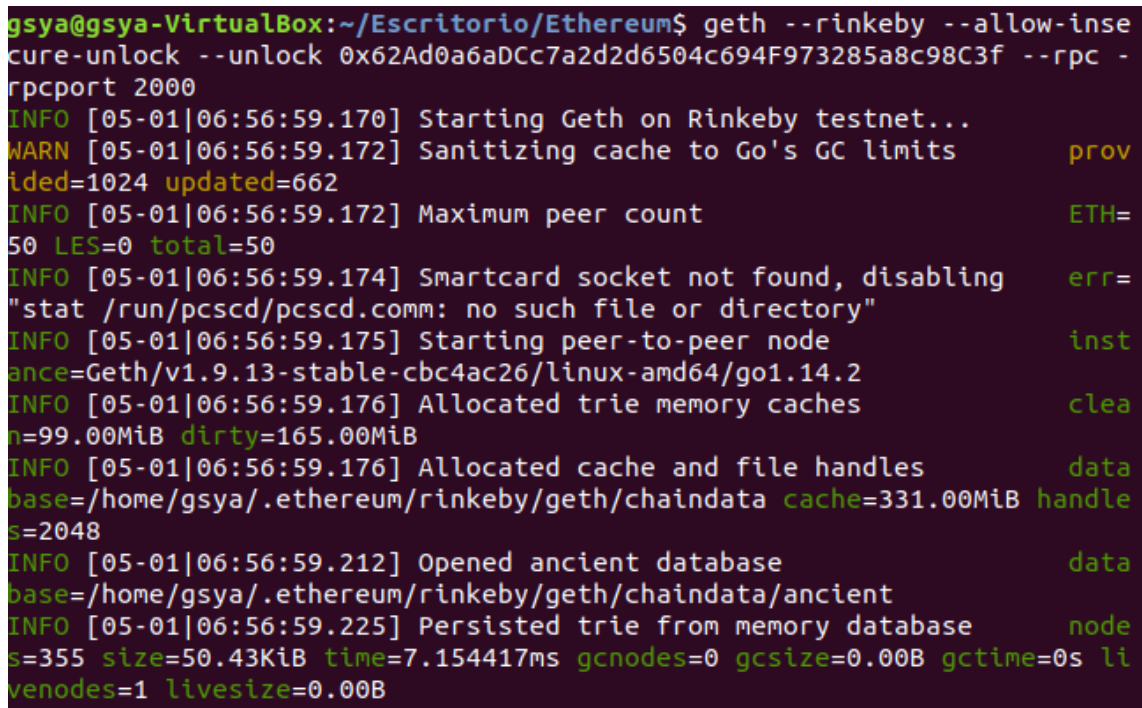
Por ejemplo para [conectarnos a rinkeby](#), primero nos creamos una cuenta:

```
geth --rinkeby account new
```

Y nos conectamos con:

```
geth --rinkeby --allow-insecure-unlock --unlock 0x<your_public_address> --  
rpc --rpcport <anyport>
```

Aquí vemos como comienza a conectarse...



```
gsya@gsya-VirtualBox:~/Escritorio/Ethereum$ geth --rinkeby --allow-inse  
cure-unlock --unlock 0x62Ad0a6aDCc7a2d2d6504c694F973285a8c98C3f --rpc -  
rpcport 2000  
INFO [05-01|06:56:59.170] Starting Geth on Rinkeby testnet...  
WARN [05-01|06:56:59.172] Sanitizing cache to Go's GC limits      prov  
ided=1024 updated=662  
INFO [05-01|06:56:59.172] Maximum peer count      ETH=  
50 LES=0 total=50  
INFO [05-01|06:56:59.174] Smartcard socket not found, disabling      err=  
"stat /run/pcscd/pcscd.comm: no such file or directory"  
INFO [05-01|06:56:59.175] Starting peer-to-peer node      inst  
ance=Geth/v1.9.13-stable-cbc4ac26/linux-amd64/go1.14.2  
INFO [05-01|06:56:59.176] Allocated trie memory caches      clea  
n=99.00MiB dirty=165.00MiB  
INFO [05-01|06:56:59.176] Allocated cache and file handles      data  
base=/home/gsy/.ethereum/rinkeby/geth/chaindata cache=331.00MiB handle  
s=2048  
INFO [05-01|06:56:59.212] Opened ancient database      data  
base=/home/gsy/.ethereum/rinkeby/geth/chaindata/ancient  
INFO [05-01|06:56:59.225] Persisted trie from memory database      node  
s=355 size=50.43KiB time=7.154417ms gcnodes=0 gcsiz=0.00B gctime=0s li  
venodes=1 livesize=0.00B
```

Y comienza a importar los bloques de la red existente:

```

INFO [05-01|06:57:11.724] Imported new block receipts      coun
t=192 elapsed=20.278ms  number=960 hash=413833...8d126d age=3y1mo4d size=
120.38KiB
INFO [05-01|06:57:12.138] Imported new state entries      coun
t=708 elapsed=60.652µs  processed=1365 pending=18050 retry=0 duplicate=
0 unexpected=0
INFO [05-01|06:57:12.225] Imported new state entries      coun
t=384 elapsed=611.696µs processed=1749 pending=18361 retry=0 duplicate=
0 unexpected=0
INFO [05-01|06:57:12.333] Imported new block headers      coun
t=576 elapsed=214.389ms  number=1536 hash=ebcd68...10b559 age=3y1mo4d
INFO [05-01|06:57:12.434] Imported new block headers      coun
t=192 elapsed=95.365ms   number=1728 hash=7eff72...2707a2 age=3y1mo4d
INFO [05-01|06:57:12.466] Imported new block receipts      coun
t=576 elapsed=101.867ms  number=1536 hash=ebcd68...10b559 age=3y1mo4d size
=361.12KiB
INFO [05-01|06:57:12.495] Imported new block receipts      coun
t=192 elapsed=26.348ms   number=1728 hash=7eff72...2707a2 age=3y1mo4d size
=120.38KiB
INFO [05-01|06:57:12.694] Imported new state entries      coun
t=384 elapsed=2.328ms    processed=2133 pending=18838 retry=0 duplicate=
0 unexpected=0
INFO [05-01|06:57:12.812] Imported new block headers      coun
t=384 elapsed=111.149ms  number=2112 hash=cc3a9a...11d37c age=3y1mo4d
INFO [05-01|06:57:12.861] Imported new block receipts      coun
t=384 elapsed=34.198ms   number=2112 hash=cc3a9a...11d37c age=3y1mo4d size
=240.75KiB

```

2.2.Red propia.

En primer lugar es conveniente crear una o varias cuentas con el cliente geth para poder más tarde establecerla con autoridad en nuestra red propia. Usamos el siguiente comando:

```
geth account new
```

Nos va a pedir que le asignemos una contraseña y luego nos dará la clave pública:

```

gsya@gsya-VirtualBox:~$ geth account new
INFO [05-01|06:14:15.190] Maximum peer count
      ETH=50 LES=0 total=50
INFO [05-01|06:14:15.190] Smartcard socket not found, disabling
      err="stat /run/pcscd/pcscd.comm: no such file or directory"
Your new account is locked with a password. Please give a passw
ord. Do not forget this password.
Password:
Repeat password:

Your new key was generated

Public address of the key:  0xA3F9d442b9ff05721E8e6Fd82fFaa684
BE9364A0
Path of the secret key file: /home/gsyas/.ethereum/keystore/UTC-
-2020-05-01T04-14-22.964546639Z--a3f9d442b9ff05721e8e6fd82ffaa6
84be9364a0

- You can share your public address with anyone. Others need it
  to interact with you.
- You must NEVER share the secret key with anyone! The key cont
  rols access to your funds!
- You must BACKUP your key file! Without the key, it's impossib

```

Después tenemos que generar un bloque generador utilizando la herramienta puppeth de geth. Simplemente escribimos puppeth en la terminal y nos aparece el asistente.

```
gsya@gsya-VirtualBox:~$ puppeth
+-----+
| Welcome to puppeth, your Ethereum private network manager |
|
| This tool lets you create a new Ethereum network down to |
| the genesis block, bootnodes, miners and ethstats servers |
| without the hassle that it would normally entail.         |
|
| Puppeth uses SSH to dial in to remote servers, and builds |
| its network components out of Docker containers using the |
| docker-compose toolset.                                   |
+-----+

Please specify a network name to administer (no spaces, hyphens or capital letters please)
```

Indicamos que queremos crearlo desde 0:

```
What would you like to do? (default = create)
 1. Create new genesis from scratch
 2. Import already existing genesis
> 1
```

Nos deja elegir el tipo de consenso:

```
Which consensus engine to use? (default = clique)
 1. Ethash - proof-of-work
 2. Clique - proof-of-authority
> 1
```

El tiempo de bloque:

```
How many seconds should blocks take? (default = 15)
>
```

Cuentas que pueden validar:

```
Which accounts are allowed to seal? (mandatory at least one)
> 0xA3F9d442b9fF05721E8e6Fd82fFaa684BE9364A0
```

A que cuentas se le dan fondos inicialmente:

```
Which accounts should be pre-funded? (advisable at least one)
> 0xA3F9d442b9fF05721E8e6Fd82fFaa684BE9364A0
```

Muy importante es el Network ID que va a ser el identificador de nuestra red:

```
Specify your chain/network ID if you want an explicit one (default = random)
> 1516
```

Una vez configurado elegimos la opción exportar lo que nos va a generar un archivo JSON con la información de la red:

```
What would you like to do? (default = stats)
```

1. Show network stats
2. Manage existing genesis
3. Track new remote server
4. Deploy network components

```
> 2
```

1. Modify existing configurations
2. Export genesis configurations
3. Remove genesis configuration

```
> 2
```

```
{
  "config": {
    "chainId": 1518,
    "homesteadBlock": 0,
    "eip150Block": 0,
    "eip150Hash":
"0x0000000000000000000000000000000000000000000000000000000000000000",
    "eip155Block": 0,
    "eip158Block": 0,
    "byzantiumBlock": 0,
    "constantinopleBlock": 0,
    "petersburgBlock": 0,
    "istanbulBlock": 0,
    "clique": {
      "period": 15,
      "epoch": 30000
    }
  },
  "nonce": "0x0",
  "timestamp": "0x5eabad8e",
  "extraData":
"0x0000000000000000000000000000000000000000000000000000000000000000a3f9d442b9ff05",
  "gasLimit": "0x47b760",
  "difficulty": "0x1",
  "mixHash":
"0x0000000000000000000000000000000000000000000000000000000000000000",

```

Este archivo JSON es muy importante porque es necesario para crear todos los nodos que van a pertenecer a esta red.

Para crear un nodo nuevo con este archivo génesis usamos el siguiente comando:

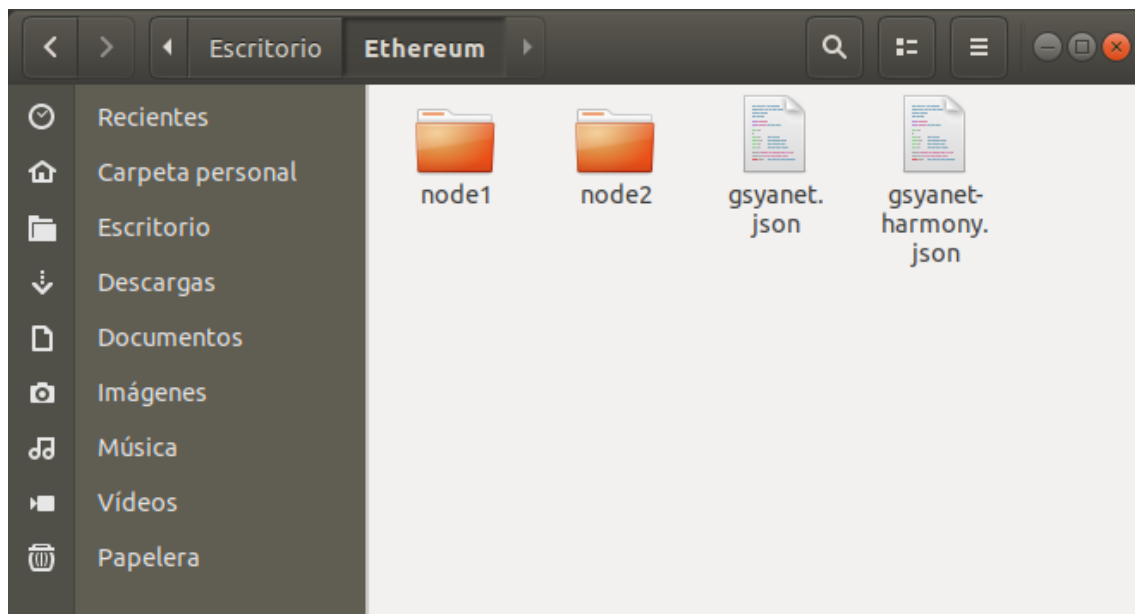
```
geth --datadir <dirección> init gsyonet.json
```

```

gsya@gsya-VirtualBox:~/Escritorio/Ethereum$ geth --datadir node1 init g
syonet.json
INFO [05-01|06:33:15.764] Maximum peer count                      ETH=
50 LES=0 total=50
INFO [05-01|06:33:15.765] Smartcard socket not found, disabling  err=
"stat /run/pcscd/pcscd.comm: no such file or directory"
INFO [05-01|06:33:15.778] Allocated cache and file handles      data
base=/home/gsyas/Escritorio/Ethereum/node1/geth/chaindata cache=16.00MiB
handles=16
INFO [05-01|06:33:15.807] Writing custom genesis block
INFO [05-01|06:33:15.825] Persisted trie from memory database    node
s=356 size=50.51KiB time=1.557944ms gcnodes=0 gcsiz=0.00B gctime=0s li
venodes=1 livesize=0.00B
INFO [05-01|06:33:15.826] Successfully wrote genesis state      data
base=chaindata hash=cefdd2...c757d6
INFO [05-01|06:33:15.829] Allocated cache and file handles      data
base=/home/gsyas/Escritorio/Ethereum/node1/geth/lightchaindata cache=16.
00MiB handles=16
INFO [05-01|06:33:15.836] Writing custom genesis block
INFO [05-01|06:33:15.843] Persisted trie from memory database    node
s=356 size=50.51KiB time=1.87599ms gcnodes=0 gcsiz=0.00B gctime=0s li
venodes=1 livesize=0.00B
INFO [05-01|06:33:15.844] Successfully wrote genesis state      data
base=lightchaindata hash=cefdd2...c757d6
gsya@gsya-VirtualBox:~/Escritorio/Ethereum$

```

He creado 2 nodos distintos para poder ejecutarlos y que interactúen entre sí:

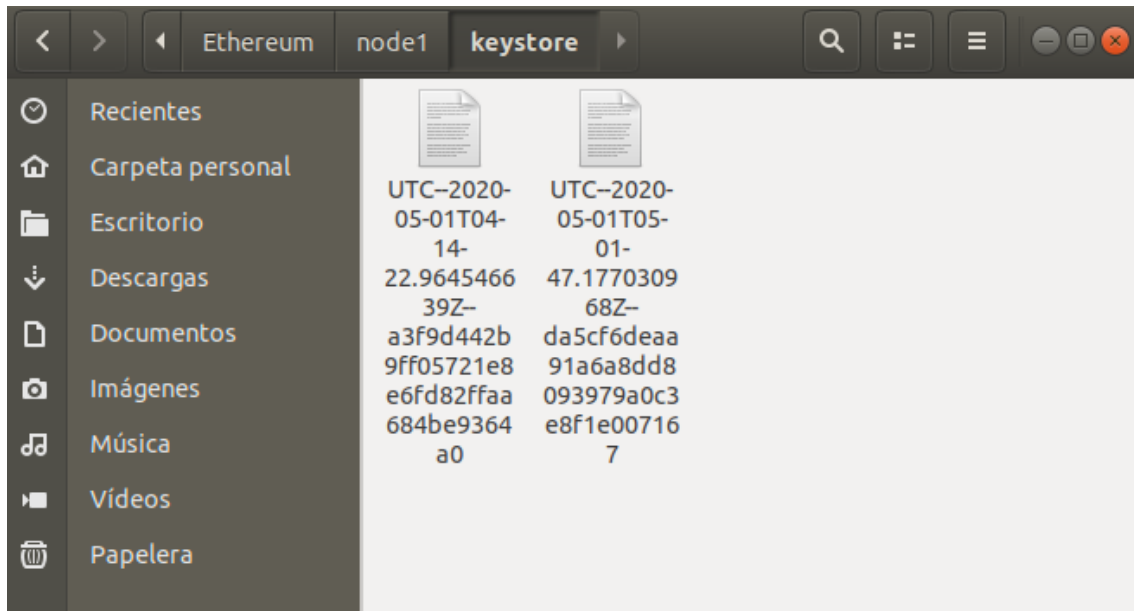


El comando base para ejecutar los nodos es:

geth --datadir <dirección>

Pero para que el nodo empiece a minar, o lo que es lo mismo, pueda procesar transacciones es necesario asignarle una cuenta y conectarlo con al menos otro nodo.

Primero copiamos el archivo Key de las cuentas que hemos creado, por defecto están en ‘~/ethereum/keystore’, a las keystore de nuestros nodos.



Para ejecutar los nodos con las opciones de conectividad:

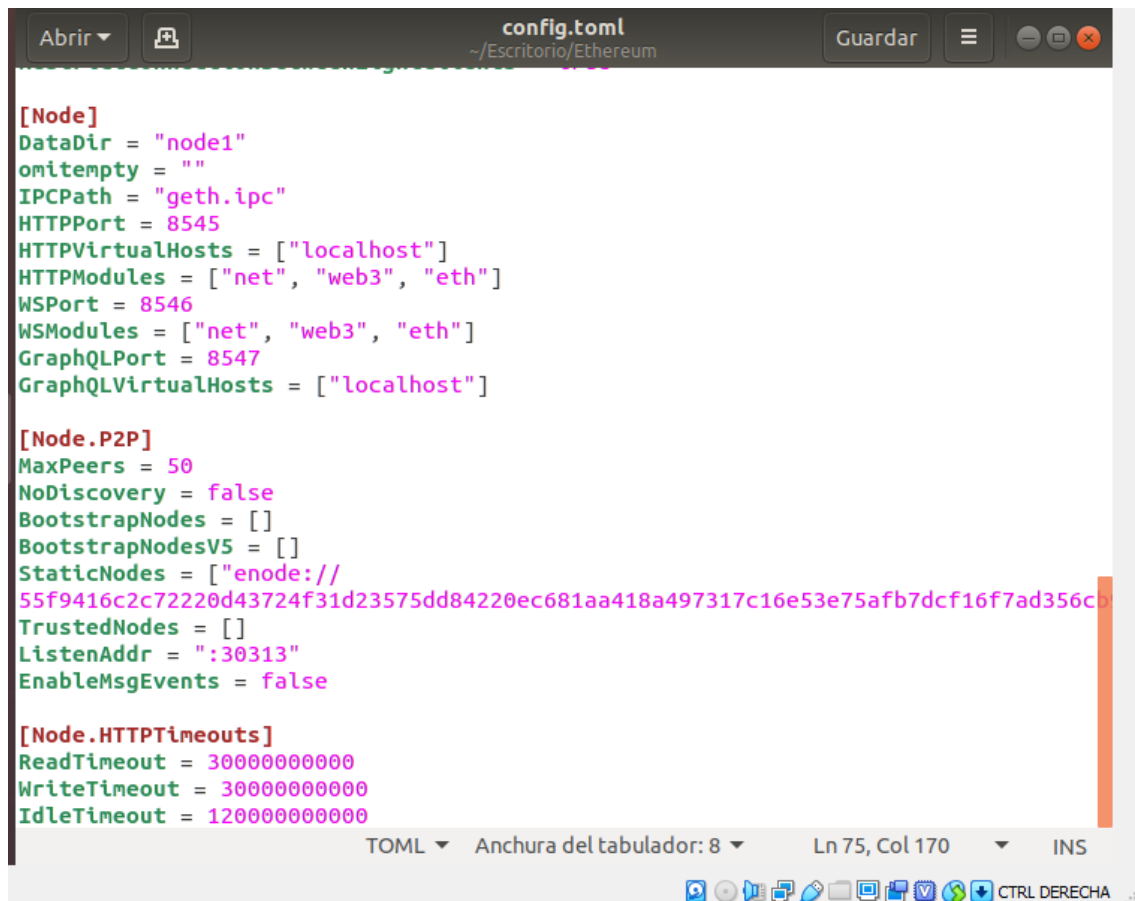
```
geth --datadir node1/ --syncmode 'full' --port <puerto> --rpc --rpcaddr  
'localhost' --rpcport <puerto> --rpcapi 'personal,db,eth,net,web3,txpool,miner' --  
rpccorsdomain "*" --networkid 1518 --allow-insecure-unlock -unlock  
'0x<direccionpublica> --password <direccioncontraseña> --mine
```

Con esto los nodos van a ejecutarse, pero todavía no se van a encontrar, podríamos conectarnos por ipc o rpc (como voy a explicar más adelante) para añadir los peers a mano en ejecución, pero es mejor opción crear un archivo de configuración y añadirlos de manera estática.

Para ello ejecutamos:

```
geth --datadir <dirección> dumpconfig > config.toml
```

Esto nos va a generar un archivo de configuración estándar que podemos editar añadiendo las opciones que queramos como los peers.

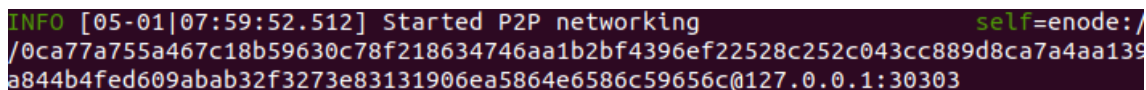


```
[Node]
DataDir = "node1"
omitempty = ""
IPCPath = "geth.ipc"
HTTPPort = 8545
HTTPVirtualHosts = ["localhost"]
HTTPModules = ["net", "web3", "eth"]
WSPort = 8546
WSModules = ["net", "web3", "eth"]
GraphQLPort = 8547
GraphQLVirtualHosts = ["localhost"]

[Node.P2P]
MaxPeers = 50
NoDiscovery = false
BootstrapNodes = []
BootstrapNodesV5 = []
StaticNodes = ["enode://
55f9416c2c72220d43724f31d23575dd84220ec681aa418a497317c16e53e75afb7dcf16f7ad356cb
TrustedNodes = []
ListenAddr = ":30313"
EnableMsgEvents = false

[Node.HTTPTimeouts]
ReadTimeout = 30000000000
WriteTimeout = 30000000000
IdleTimeout = 120000000000
```

Puedes encontrar la dirección “enode” ejecutando el nodo con el comando **geth --datadir <dirección>** si no quieres complicarte:



```
INFO [05-01|07:59:52.512] Started P2P networking                      self=enode:/
/0ca77a755a467c18b59630c78f218634746aa1b2bf4396ef22528c252c043cc889d8ca7a4aa139
a844b4fed609abab32f3273e83131906ea5864e6586c59656c@127.0.0.1:30303
```

En el archivo de configuración he realizado los siguientes cambios:

- Asignar el DadaDir correspondiente.
- Añadir el StaticNode de los nodos que van a ser peers.
- Poner NoDiscovery a True.
- Eliminar nodos preestablecidos.
- Asignar puertos.

Una vez tenemos estos archivos de configuración (1 por nodo) podemos ejecutar con:

geth --config config.toml --verbosity 4 --rpcorsdomain "*" --allow-insecure-unlock -unlock '< direccionpublica >' --password <direccioncontraseña> --mine

Si queremos reducir aún mas el esfuerzo podemos crear un pequeño script bash por cada nodo llamado start.sh como:

#!/bin/bash

#!/*- ENCODING: UTF-8 -*-

geth --config config.toml --verbosity 4 --rpccorsdomain "*" --allow-insecure-unlock -
unlock '<direccionpublica>' --password <direccioncontraseña> --mine

y ejecutamos así:

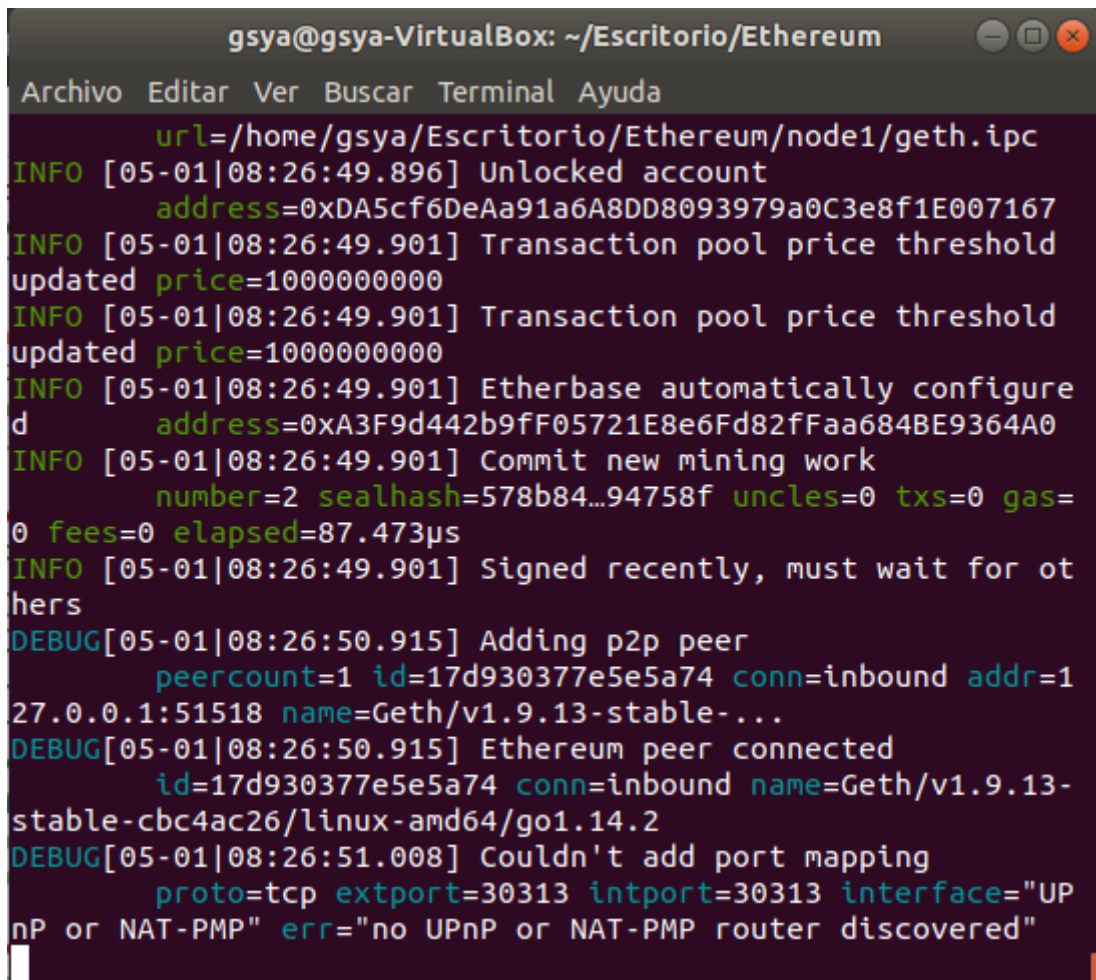
./start.sh

Resultado

Terminal 1:

```
gsya@gsya-VirtualBox: ~/Escritorio/Ethereum
Archivo Editar Ver Buscar Terminal Ayuda
INFO [05-01|08:26:50.898] IPC endpoint opened
  url=/home/gsyas/Escritorio/Ethereum/node2/geth.ipc
INFO [05-01|08:26:50.905] New local node record
  seq=12 id=17d930377e5e5a74 ip=127.0.0.1 udp=0 tcp=30312
INFO [05-01|08:26:50.907] Started P2P networking
  self="enode://55f9416c2c72220d43724f31d23575dd84220ec681aa418
a497317c16e53e75afb7dcf16f7ad356cb9df100b2933dfb31f84e1fd01cfa46
dc8b1f2d74c35b11e@127.0.0.1:30312?discport=0"
DEBUG[05-01|08:26:50.915] Adding p2p peer
  peercount=1 id=ea94d6557e8dc62b conn=staticdial addr=127.0.0.
1:30313 name=Geth/v1.9.13-stable-...
DEBUG[05-01|08:26:50.915] Ethereum peer connected
  id=ea94d6557e8dc62b conn=staticdial name=Geth/v1.9.13-stable-
cbc4ac26/linux-amd64/go1.14.2
INFO [05-01|08:26:51.904] Unlocked account
  address=0xA3F9d442b9fF05721E8e6Fd82fFaa684BE9364A0
INFO [05-01|08:26:51.904] Transaction pool price threshold updat
ed price=1000000000
INFO [05-01|08:26:51.904] Transaction pool price threshold updat
ed price=1000000000
INFO [05-01|08:26:51.904] Etherbase automatically configured
  address=0xA3F9d442b9fF05721E8e6Fd82fFaa684BE9364A0
INFO [05-01|08:26:51.904] Commit new mining work
  number=2 sealhash=020b59...b61e04 uncles=0 txs=0 gas=0 fees=0 e
```

Terminal 2:



```
gsya@gsya-VirtualBox: ~/Escritorio/Ethereum
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
url=/home/gsys/Escritorio/Ethereum/node1/geth.ipc
INFO [05-01|08:26:49.896] Unlocked account
address=0xDA5cf6DeAa91a6A8DD8093979a0C3e8f1E007167
INFO [05-01|08:26:49.901] Transaction pool price threshold
updated price=1000000000
INFO [05-01|08:26:49.901] Transaction pool price threshold
updated price=1000000000
INFO [05-01|08:26:49.901] Etherbase automatically configure
d address=0xA3F9d442b9fF05721E8e6Fd82fFaa684BE9364A0
INFO [05-01|08:26:49.901] Commit new mining work
number=2 sealhash=578b84...94758f uncles=0 txs=0 gas=
0 fees=0 elapsed=87.473µs
INFO [05-01|08:26:49.901] Signed recently, must wait for ot
hers
DEBUG[05-01|08:26:50.915] Adding p2p peer
peercount=1 id=17d930377e5e5a74 conn=inbound addr=1
27.0.0.1:51518 name=Geth/v1.9.13-stable-...
DEBUG[05-01|08:26:50.915] Ethereum peer connected
id=17d930377e5e5a74 conn=inbound name=Geth/v1.9.13-
stable-cbc4ac26/linux-amd64/go1.14.2
DEBUG[05-01|08:26:51.008] Couldn't add port mapping
proto=tcp extport=30313 intport=30313 interface="UP
nP or NAT-PMP" err="no UPnP or NAT-PMP router discovered"
```

Ambos nodos se han reconocido entre sí como peers.

3.Consola en los Nodos.

Podemos conectarnos a nuestros nodos de 2 maneras: IPC(Interprocess communication) y RPC(remote procedure call).

geth attach <direcciónnodo>/geth.ipc para conexión IPC.

geth attach 'http://<ip>:<puerto>' por ejemplo **geth attach 'http://localhost:8501'** para conexión rpc.

Los comandos que puedes usar son los correspondientes a las apis que permitas a la hora de ejecutar los nodos como por ejemplo --rpcapi 'personal,db,eth,net,web3,txpool,miner'

Si ejecutamos el comando `admin.peers` podemos ver si los peers se han añadido correctamente y están online:

```
gsya@gsya-VirtualBox: ~/Escritorio/Ethereum/node1
Archivo Editar Ver Buscar Terminal Ayuda
modules: admin:1.0 clique:1.0 debug:1.0 eth:1.0 miner:1.0 net:1.0 personal:1.0
rpc:1.0 txpool:1.0 web3:1.0

> admin.peers
[
  {
    caps: ["eth/63", "eth/64", "eth/65"],
    enode: "enode://55f9416c2c72220d43724f31d23575dd84220ec681aa418a497317c16e53e75afb7dcf16f7ad356cb9df100b2933dfb31f84e1fd01cfa46dc8b1f2d74c35b11e@127.0.0.1:51624",
    id: "17d930377e5e5a743b336b97913dbcbb63b4f7058006d6f8c7ff9b92515304a1",
    name: "Geth/v1.9.13-stable-cbc4ac26/linux-amd64/go1.14.2",
    network: {
      inbound: true,
      localAddress: "127.0.0.1:30313",
      remoteAddress: "127.0.0.1:51624",
      static: false,
      trusted: false
    },
    protocols: {
      eth: {
        difficulty: 2,
        head: "0xff826cfa8f0af017dff9178cf6920c8da7e739303be30a1c75ba2e4219b0d349",
        version: 65
      }
    }
  }
]
```

4.Dapps con Truffle

Para iniciar un proyecto truffle y empezar a crear Dapps (Distributed apps) ejecutamos el siguiente comando:

truffle init

Nos va a crear un entorno por defecto:

```
gsya@gsya-VirtualBox:~/Escritorio/Proyecto$ ls
contracts migrations test truffle-config.js
gsya@gsya-VirtualBox:~/Escritorio/Proyecto$
```

Tenemos las siguientes carpetas:

- contracts** donde vamos a guardar los contratos en el lenguaje solidity,
- migrations** que viene con 1 solo archivo para una migración inicial a la blockchain pero es importante saber que hay que crear archivos de migración para subir los contratos
- tests** para los posibles tests que hagamos.

Y 1 archivo muy importante **truffle-config.js**, en el que vamos a especificar la red que se va a usar, el compilador y el resto de opciones.

Los comandos mas importantes son:

- truffle compile**: para compilar el proyecto.
- truffle migrate**: para compilar y subir los contratos a la blockchain especificada en truffle-config.js
- truffle test**: para llevar a cabo los tests.
- truffle console**: para ejecutar transacciones desde la consola

Truffle también trae plantillas para Frontend para distintos frameworks como React, Angular y Vue. Por ejemplo para descargar la de React:

truffle unbox react

5.Remix IDE.

Remix IDE es un entorno de desarrollo muy útil al cual puedes acceder directamente en el navegador para realizar tus primeros contratos. Un contrato es simplemente un trozo de código que se ejecuta en la blockchain.

Remix te permite compilar y ejecutar el código que desarrolles y llevar a cabo transacciones en un entorno simulado.

The screenshot displays the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel is visible, featuring a 'JavaScript VM' environment, an account address '0x1a5...9D6C2 (100 ether)', a gas limit of '3000000', and a value of '0 wei'. The contract selected is 'RBAC - browser/RBAC.sol'. Below this, there are buttons for 'Deploy', 'PUBLISH TO IPFS', and 'OR At Address Load contract from Address'. A message at the bottom states: 'Currently you have no contract instances to interact with.'

The right side of the interface shows the code editor with the 'RBAC.sol' file open. The code is written in Solidity and includes a pragma statement for Solidity version 0.6.0, an experimental ABI Encoder V2, and a contract named 'RBAC'. The contract defines several events: 'UserCreated', 'RoleCreated', 'ProtectedObjectCreated', 'PermissionCreated', 'UserDeleted', 'RoleDeleted', 'ProtectedObjectDeleted', and 'PermissionDeleted'. It also defines two structs: 'Role' and 'Permission'. The 'Role' struct has fields for 'name', 'membersNumber', and 'members'. The 'Permission' struct has fields for 'ProtectedObject', 'role', and 'permissionTypes'.

At the bottom of the interface, there is a terminal window with the following content:

```
remix.debugHelp(): Display help message for debugging

- Welcome to Remix v0.10.1 -

You can use this terminal for:
• Checking transactions details and start debugging.
• Running JavaScript scripts. The following libraries are accessible:
  o web3 version 1.0.0
  o ethers.js
  o swarmgw
  o remix (run remix.help() for more info)
• Executing common command to interact with the Remix interface (see list of commands above). Note that these commands can also be included and run from a JavaScript script.
• Use exports.register(key, obj).remove(key).clear() to register and reuse object across script executions.
```

En la consola puedes ejecutar comandos y ver los resultados de las transacciones como este deploy del contrato RBAC.sol para lo cual he tenido que subir el límite de gas a 30000000 debido al tamaño del contrato:

creation of RBAC pending...

[vm] from:0x928...fc189 to:RBAC.(constructor) value:0 wei data:0x608...60033 logs:0 hash:0xe2d...54a08

status	0x1 Transaction mined and execution succeed
transaction hash	0xe2d8195c371f124445a8a72dd30c5062e7c741b110f9ac7aad42da7eb8f54a08
contract address	0x2069ab4c0be74807a5415603a782f3687e62dfc8
from	0x928750021fdc0d57c51ba7cbf7f592aba63fc189
to	RBAC.(constructor)
gas	30000000 gas
transaction cost	5112108 gas
execution cost	3864620 gas
hash	0xe2d8195c371f124445a8a72dd30c5062e7c741b110f9ac7aad42da7eb8f54a08
input	0x608...60033
decoded input	{ }

Ejecutar transacciones mientras estas llevando a cabo el desarrollo ha sido una de sus funciones más útiles porque te ahorras migrar cada vez a una blockchain.

addPermission

string objectName, string roleName

addProtectedO...

string objectName

addRole

string roleName

addUser

string userName, address id

addUser

string userName, address id, strin

changeUserRole

address userAddress, string roleName

6.Ganache.

Ganache es una herramienta que te permite montar una blockchain simulada con una serie de cuentas ya creadas y listas para usar.

```
gsya@gsya-VirtualBox:~/Escritorio/React$ ganache
ganache: orden no encontrada
gsya@gsya-VirtualBox:~/Escritorio/React$ ganache-cli
Ganache CLI v6.9.1 (ganache-core: 2.10.2)

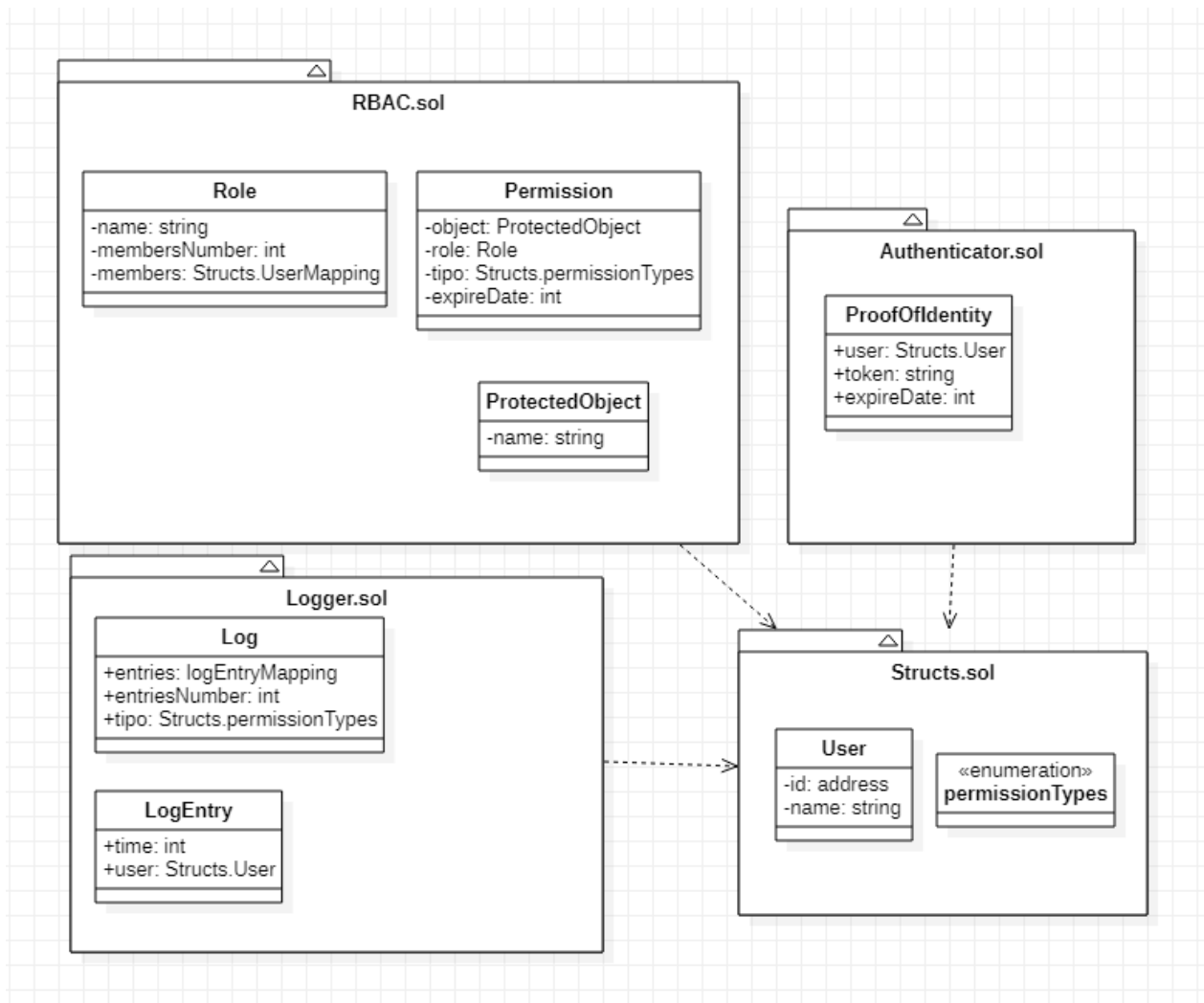
Available Accounts
=====
(0) 0x28d31F416aA8549561AC738a278A5015F5832275 (100 ETH)
(1) 0xE759249eBc64998ac640521Aec057A0eC3123ba2 (100 ETH)
(2) 0x49498Ab51a372CcEE07E300A48EC21E7E5240c99 (100 ETH)
(3) 0xF68E775CBB7B45A754232BCf293C3C636E5b98E8 (100 ETH)
(4) 0xA466960b26ee8aB467f71c0ed9F26B8D61547211 (100 ETH)
(5) 0xcc295615A549b0fdCAc4600f2a8b55E45D97BEd8 (100 ETH)
(6) 0x54a943BEFCe225b451F851EdcC4596035Fb10e00 (100 ETH)
(7) 0x95eA538dCD3A3E2A4f2C672fA03166Af3eDA05b4 (100 ETH)
(8) 0x1fA1E9127102B9FD759e12606B2555F15d3448E8 (100 ETH)
(9) 0x783d634272FABF2df65E254F7F54c3Da639f16d7 (100 ETH)

Private Keys
=====
(0) 0x9afac7a61e5551b96632b8294db53821eda1c0710b3374684e9fe6a4992ce763
(1) 0x5313ca993815bb0c5af254e536dd2f79424e84465867fb36f1707399977ba375
(2) 0xaf84829956cf40af7747d87473b58b7c0e3d04c7411f83c3f30d811ffe30dab9
(3) 0xcdc2c91c305f48009f9ec0a1dc61f011aebf78833d057cc35a99b5a877f09ba5
(4) 0xe959222ac296a75a196ff515c683d32f1820ab8133e6e59775319507c953377f
(5) 0xb1807918bad6676f63b84aef80303e8f2575d688fedf84441243c6146fa4a8cb
(6) 0x4be50714603c22ad5037af463d0f9212392999aa84bdf097d17ea116b38897b8
(7) 0x3ba71364aa8144cc7cbd8c0a3536c4a3566f588ec41ef907475665716778f936
```

Puedes ser una buena alternativa si te gusta desarrollar en tu propio entorno y probar las transacciones en un entorno seguro.

7. Contratos.

Esta es actualmente la estructura de los contratos:



Estas son las funciones de cada contrato actualmente, en naranja las transacciones y en azul las vistas:

RBAC

addPermission	string objectName, string roleNar	▼
addProtectedO...	string objectName	▼
addRole	string roleName	▼
addUser	string userName, address id	▼
addUser	string userName, address id, strin	▼
changeUserRole	address userAddress, string roleN	▼
deletePermissi...	string objectName, string roleNar	▼
deleteProtecte...	string objectName	▼
deleteRole	string roleName	▼

deleteUser	address userAddress	▼
askForPermissi...	string objectName, string roleNar	▼
getAllPermissi...		
getAllProtecte...		
getAllRolesDes...		
getAllUsers		
getAllUsersFro...	string roleName	▼

Logger

addEntryByType	uint8 tipo, uint256 time, tuple use	▼
addLog	uint8 tipo	▼
getEntriesByTy...	uint8 tipo	▼
logs	uint256	▼

Authenticator

addToken	tuple user, string token, uint256 e	▼
getAllTokens		
getTokenByUser	tuple user	▼

8. Links de Interés

<https://ethereum.org/developers/#getting-started>

<https://kauri.io/full-stack-dapp-tutorial-series/5b8e401ee727370001c942e3/c>

<https://medium.com/@david.chou93/connect-to-rinkeby-testnet-a18098ce2ea0>

<https://remix.ethereum.org/#optimize=false&evmVersion=null&version=soljson-v0.6.6+commit.6c089d02.js>

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