Nodo Testnet

En Bitcoin existen tres distintas redes:

- Mainnet
  Esta red es la principal y donde se guardan las direcciones y circulante de Bitcoin en bloques.
- Testnet
  Es una red especial para probar el funcionamiento de aplicaciones (evitando los costos de transacción real) antes de lanzarlas en mainnet. Tiene su propia blockchain y mineros que generar sus bloques de manera separada al mainnet.
- Regtest
  De igual manera que testnet, regtest es un modo de prueba de un solo punto o nodo. Dejando de lado la sincronización con otros nodos para hacer tests.

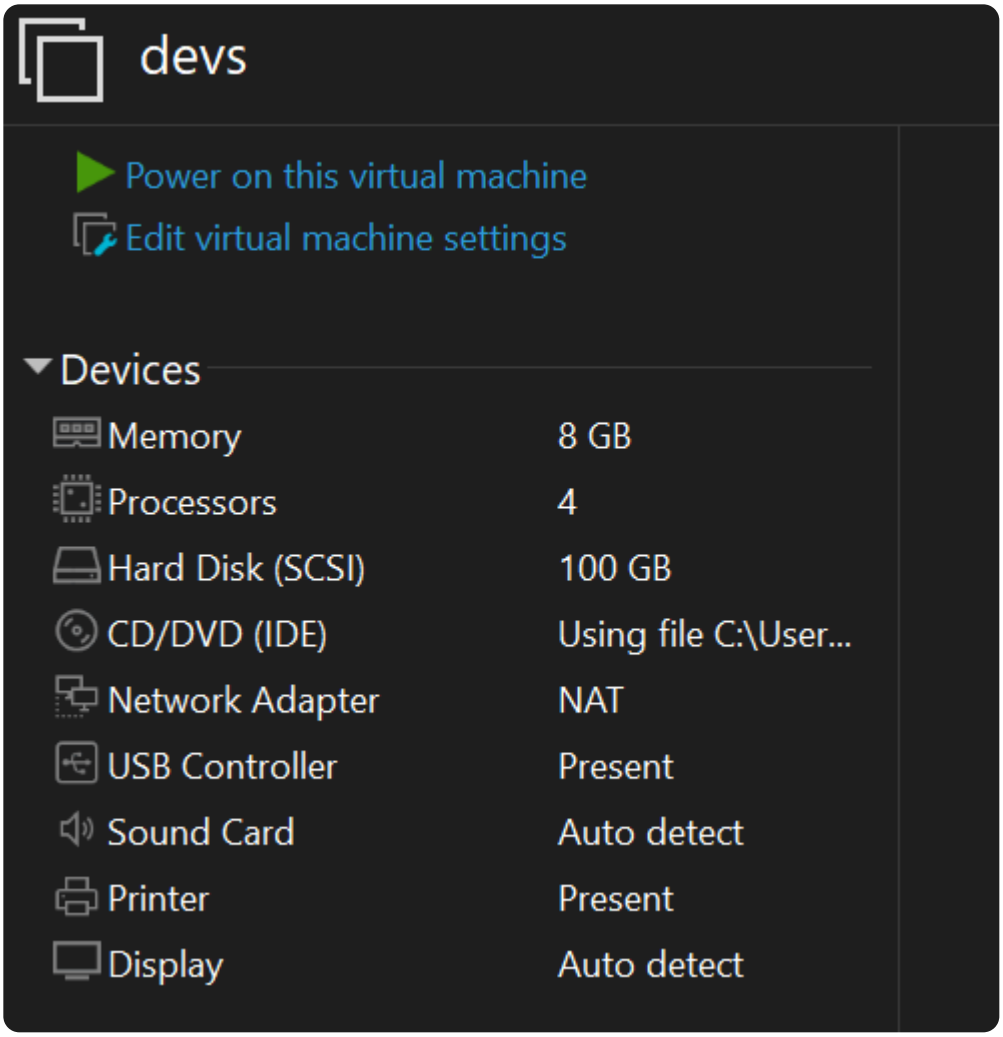
En Bitcoin/Nodo Bitcoin detallamos el proceso para construir un nodo completo Bitcoin en una raspberry.

En esta publicación estudiamos el montar un nodo Bitcoin en `testnet` a partir de un ordenador portátil.

Consideraciones

El ordenador portátil virtualizará una máquina virtual con Parrot Os que es un sistema Linux basado en Debian. En Setup Linux detallamos como instalar un sistema desde cero.

El ordenador asignado es de baja potencia teniendo este esquema:



El blockchain de Testnet no es tan pesado como el mainnet. Ronda a 12-08-2022 (mm-dd-aa) por los 30 GB en total. Por lo que una máquina VM con 100 GB es más que suficiente.

Tutorial paso a paso

Instalar un nodo Bitcoin es igual en ambos casos, main y test net. La diferencia se encuentra en el archivo de configuración `bitcoind.conf`

En Bitcoin/Nodo Bitcoin se explica a detalle los pasos para instalar un nodo en un Raspberry que tiene un procesador arm de 64 bits. Para hacerlo en una máquina que virtualiza un x86 repasamos el procedimiento.

Pre instalación

Instalamos paquetes necesarios

```
sudo apt update && sudo apt install ca-certificates gnupg gpg wget jq --no-install-recommends -y
```

Y seguimos rápidamente:

- Configurar ssh
- Firewal ufw
- Instalar Fail2ban
- Instalar nginx
- Instalar Tor

En un directorio descargamos los siguientes archivos:

1. Bitcoin-core
2. El checksum
3. Firmas gpg que validan checksum y core

En la página https://bitcoincore.org o en otros repositorios se puede encontrar los binarios del core de Bitcoin. Se debe descargar el formato según sea el caso: windows, mac, linux y la arquitectura ya sea arm o x86. En el caso nuestro usamos `/bitcoin-24.0.1-x86_64-linux-gnu.tar.gz` .
De igual forma se descarga los binarios del sha256 y las firmas digitales gpg en una carpeta temporal `tmp` .

```
# Bitcoin Core binary
wget https://bitcoincore.org/bin/bitcoin-core-24.0.1/bitcoin-24.0.1-x86_64-linux-gnu.tar.gz

# cryptographic checksum
wget https://bitcoincore.org/bin/bitcoin-core-24.0.1/SHA256SUMS

# signatures checksums
wget https://bitcoincore.org/bin/bitcoin-core-4.0.1/SHA256SUMS.asc

# signature timestamp ots
wget https://bitcoincore.org/bin/bitcoin-core-24.0.1/SHA256SUMS.ots
```

Verificando autenticidad de Binarios

Checksum256

El checksum hace referencia al hash sha256 que se hace al binario original para constatar que no existe manipulación.
El checksum que descargamos de la web tiene la siguiente característica.

```
cat SHA256SUMS
>2cca490c1f2842884a3c5b0606f179f9f937177da4eadd628e3f7fd7e25d26d0
```

y coincide al realizar el hash sha256 del bitcoin-core.

```
sha256sum bitcoin-23.0-x86_64-linux-gnu.tar.gz
>2cca490c1f2842884a3c5b0606f179f9f937177da4eadd628e3f7fd7e25d26d0  bitcoin-23.0-x86_64-linux-gnu.tar.gz
```

Firma digital

Como sabemos el desarrollo de Bitcoin-core esta a cargo por un grupo exclusivo de programadores que dejan su propia firma al publicar un versión final.

Bajamos de un repositorio las llaves públicas que se tienen de estos desarrolladores e importamos las llaves.

```
wget https://raw.githubusercontent.com/bitcoin/bitcoin/master/contrib/builder-keys/keys.txt
while read fingerprint keyholder_name; do gpg --keyserver hkps://keyserver.ubuntu.com --recv-keys ${fin
```

Verificamos que el SHA256SUMS.asc es válido con cada una de estas firmas.

```
gpg --verify SHA256SUMS.asc
```

Basta con ver que cumple con varias firmas.

```
>gpg: Good signature from ...
>Primary key fingerprint:...
```

Verificando Timestamp

Usando el estándar Opentimestamps se ha dejado un timestamp del archivo `SHA256SUMS` que se puede descargar.

```
wget https://bitcoincore.org/bin/bitcoin-core-23.0/SHA256SUMS.ots
```

Como se desarrolla en OpenTimeStamps, se utiliza el blockchain de Bitcoin para verificar que existía el archivo previo stamp.

```
sudo pip3 install opentimestamps-client
ots --version
```

Se puede usar la versión web para verificar, pero usando el nodo que instalamos se puede verificar propiamente.

```
ots verify SHA256SUMS.ots -f SHA256SUMS
>Got 1 attestation(s) from https://alice.btc.calendar.opentimestamps.org
>Got 1 attestation(s) from https://finney.calendar.eternitywall.com
>Got 1 attestation(s) from https://btc.calendar.catallaxy.com
>Got 1 attestation(s) from https://bob.btc.calendar.opentimestamps.org
>Success! Bitcoin block 733490 attests existence as of 2022-04-25 -04
```

Instalando el Core

Descomprimimos el archivo y se instala.

```
tar -xvf bitcoin-24.0.1-x86_64-linux-gnu.tar.gzz
rm bitcoin-24.0.1-x86_64-linux-gnu.tar.gz
sudo install -m 0755 -o root -g root -t /usr/local/bin bitcoin-24.0.1/bin/*
bitcoind --version
>Bitcoin Core version v24.0.1
Copyright (C) 2009-2022 The Bitcoin Core developers

Please contribute if you find Bitcoin Core useful. Visit
<https://bitcoincore.org/> for further information about the software.
The source code is available from <https://github.com/bitcoin/bitcoin>.

This is experimental software.
Distributed under the MIT software license, see the accompanying file COPYING
or <https://opensource.org/licenses/MIT>
```

Nuevo user: bitcoin

Por razones de seguridad se desea que el core de bitcoin corra en segundo plano bajo un usuario con permisos limitados. Para lograr esto se crea un nuevo usuario `bitcoin` y se añade al usuario `admin` al grupo del nuevo user `bitcoin`. Esto para configurar permisos por grupo.

```
sudo adduser --gecos "" --disabled-password bitcoin
sudo adduser admin bitcoin
```

Data Folder

En la red testnet de Bitcoin el blockchain completo esta a fecha 01-20-2023 (mm-dd-aa) alrededor de los 33 GB. Por esta razón se monta el nodo en el mismo disco duro del sistema.
Se crea una carpeta que contenga la información para sincronizar.

```
mkdir /home/ghost/btc
sudo chown bitcoin:bitcoin /home/ghost/btc
```

Desde el usuario bitcoin `sudo su - bitcoin` creamos un enlace simbólico de la carpeta /btc/ en la carpeta del usuario bitcoin.

```
ln -s /home/ghost/btc/ /home/bitcoin/.bitcoin
ls -la
```

Archivo Bitcoin.conf

Como se estudio a detalle en Bitcoin.conf usaremos un template para testnet. En este caso usaremos RPCAuth para no dejar el password explícitamente, sin embargo, funciona correctamente son rpcuser y rpcpassword.

```
cd .bitcoin
wget https://raw.githubusercontent.com/bitcoin/bitcoin/master/share/rpcauth/rpcauth.py
python3 rpcauth.py nodo test
>rpcauth=nodo:00d8682ce66c9ef3dd9d0c0a6516b10e$c31da4929b3d0e092ba1b2755834889f888445923ac8fd69d8eb73ef
```

Con este dato creamos el archivo `nano /home/bitcoin/.bitcoin/bitcoin.conf`

```
# /home/bitcoin/.bitcoin/bitcoin.conf

server=1
txindex=1
listen=1
testnet=1

[test]
dbcache=1024
bind=127.0.0.1
rpcbind=0.0.0.0
rpcauth=nodo:sdasdasdsaca8bba088079857a3ce96c6ae$70bd90f4764efda696f3555a96c9033c3
rpcport=18332
```

Y le asignamos permisos adecuados.

```
chmod 640 /home/bitcoin/.bitcoin/bitcoin.conf
```

Corriendo Bitcoin

Basta con iniciar `bitcoind` (aún en el usuario bitcoin) y empieza a correr.

```
❯ ls -la
drwxr-xr-x bitcoin bitcoin  60 B  Fri Jan 20 15:43:50 2023 .
drwxr-xr-x ghost   ghost   504 B  Fri Jan 20 15:44:42 2023 ..
.rw-r----- bitcoin bitcoin 326 B  Fri Jan 20 15:43:50 2023 bitcoin.conf
.rw-r--r-- ghost   ghost   1.5 KB Fri Jan 20 11:55:25 2023 rpcauth.py
drwxr-xr-x bitcoin bitcoin 226 B  Fri Jan 20 15:42:02 2023 testnet3
```

Se crea una carpeta `/testnet3/` donde se almacenan todos los archivos cuando sincroniza.

```
❯ ls -la
drwxr-xr-x bitcoin bitcoin 226 B  Fri Jan 20 15:42:02 2023 .
drwxr-xr-x bitcoin bitcoin  60 B  Fri Jan 20 15:43:50 2023 ..
.rw-r----- bitcoin bitcoin  75 B  Fri Jan 20 12:36:57 2023 .cookie
.rw------- bitcoin bitcoin   0 B  Fri Jan 20 12:26:07 2023 .lock
.rw------- bitcoin bitcoin  31 B  Fri Jan 20 12:26:07 2023 {} banlist.json
drwx------ bitcoin bitcoin 8.6 KB Fri Jan 20 15:45:37 2023 blocks
drwx------ bitcoin bitcoin  11 KB Fri Jan 20 15:35:08 2023 chainstate
.rw-r----- bitcoin bitcoin 402 MB Fri Jan 20 15:46:06 2023 debug.log
.rw------- bitcoin bitcoin 242 KB Fri Jan 20 12:31:15 2023 fee_estimates.dat
drwx------ bitcoin bitcoin  14 B  Fri Jan 20 12:26:07 2023 indexes
.rw------- bitcoin bitcoin  18 B  Fri Jan 20 12:31:15 2023 mempool.dat
.rw------- bitcoin bitcoin 487 KB Fri Jan 20 15:42:02 2023 peers.dat
.rw-r----- bitcoin bitcoin   4 B  Fri Jan 20 12:36:56 2023 {} settings.json
drwxr-xr-x bitcoin bitcoin   0 B  Fri Jan 20 12:26:07 2023 wallets
```

> **Note**
>
> Si en el bitcoin.conf se quiere usar un método de autentificación distinto al declarar rpcuser y rpcpassword el método rpcauth crea un archivo `.cookie`.
>
> Este archivo (al igual que debug.log) tiene los permisos limitados solo al user `bitcoin` así que basta con agregar un permiso de lectura al grupo `chmod g+r` para ser accesible bitcoin-cli desde otros usuarios.

Cambiamos permisos adecuados al debug.log

```
chmod g+r /home/bitcoin/.bitcoin/debug.log
exit
```

Automatizando Bitcoind

Para que Bitcoin Core se ejecute tras un reinicio o algún error de manera automática añadimos un servicio a systemctl.

```
sudo nano /etc/systemd/system/bitcoind.service
```

```
[Unit]
Description=Bitcoin daemon
After=network.target

[Service]

# Service execution
###################

ExecStart=/usr/local/bin/bitcoind -daemon \

                                  pid=/run/bitcoind/bitcoind.pid \
```

```
                                    -pid=/run/bitcoind/bitcoind.pid \
                                    -conf=/home/bitcoin/.bitcoin/bitcoin.conf \
                                    -datadir=/home/bitcoin/.bitcoin \
                                    -startupnotify="chmod g+r /home/bitcoin/.bitcoin/testnet3/.cookie

# Process management
####################
Type=forking
PIDFile=/run/bitcoind/bitcoind.pid
Restart=on-failure
TimeoutSec=300
RestartSec=30

# Directory creation and permissions
####################################
User=bitcoin
UMask=0027

# /run/bitcoind
RuntimeDirectory=bitcoind
RuntimeDirectoryMode=0710

# Hardening measures
####################
# Provide a private /tmp and /var/tmp.
PrivateTmp=true

# Mount /usr, /boot/ and /etc read-only for the process.
ProtectSystem=full

# Disallow the process and all of its children to gain
# new privileges through execve().
NoNewPrivileges=true

# Use a new /dev namespace only populated with API pseudo devices
# such as /dev/null, /dev/zero and /dev/random.
PrivateDevices=true

# Deny the creation of writable and executable memory mappings.
MemoryDenyWriteExecute=true

[Install]
WantedBy=multi-user.target
```

Luego habilitamos el servicio y reiniciamos

```
sudo systemctl enable bitcoin.service
sudo reboot
```

Revisamos que todo este funcionando correctamente.

```
sudo systemctl status bitcoin.service
```

```
❯ sudo systemctl status bitcoin.service
● bitcoin.service - Bitcoin daemon
     Loaded: loaded (/etc/systemd/system/bitcoin.service; enabled;
     Active: active (running) since Fri 2023-01-20 12:36:56 -04; 3h
    Process: 709 ExecStart=/usr/local/bin/bitcoind -daemon -pid=/ru
   Main PID: 771 (bitcoind)
      Tasks: 16 (limit: 4548)
     Memory: 3.3G
        CPU: 1h 27min 7.595s
     CGroup: /system.slice/bitcoin.service
             └─771 /usr/local/bin/bitcoind -daemon -pid=/run/bitcoi

ene 20 12:36:56 parrot systemd[1]: Starting Bitcoin daemon...
ene 20 12:36:56 parrot bitcoind[709]: Bitcoin Core starting
ene 20 12:36:56 parrot systemd[1]: bitcoin.service: Can't open PID
ene 20 12:36:56 parrot systemd[1]: Started Bitcoin daemon.
lines 1-15/15 (END)
```

Verificando Bitcoin Core

Cada acción que vaya se realizando al sincronizar el blockchain se va escribiendo en el archivo `debug.log`. Para monitorear y ver en pantalla lo que realiza en segundo plano usamos el siguiente comando.

```
tail -f /home/bitcoin/.bitcoin/testnet3/debug.log
```

```
2023-01-20T20:06:19Z UpdateTip: new best=000000000000216493f87068612
94da69e1756517034b4853277df3a36a17a09 height=2026130 version=0x20600
000 log2_work=74.348830 tx=60427447 date='2021-07-08T11:17:00Z' prog
ress=0.931300 cache=703.1MiB(4957207txo)
2023-01-20T20:06:19Z UpdateTip: new best=0000000000002e2e967869f7297
619240bd5b0fc7f91e82cfa9436acea6300e2 height=2026131 version=0x20000
000 log2_work=74.348830 tx=60427448 date='2021-07-08T11:17:05Z' prog
ress=0.931300 cache=703.1MiB(4957208txo)
2023-01-20T20:06:19Z UpdateTip: new best=000000000002007553d16567b1
7b6627453fef52e210fca143d8131ff17159e height=2026132 version=0x20200
000 log2_work=74.348830 tx=60427450 date='2021-07-08T11:17:06Z' prog
ress=0.931300 cache=703.1MiB(4957209txo)
2023-01-20T20:06:19Z UpdateTip: new best=0000000000000f0614fa6eb5510
077dda44a1376277ba0abaeabd3aca8e17863 height=2026133 version=0x20800
000 log2_work=74.348830 tx=60427451 date='2021-07-08T11:17:08Z' prog
ress=0.931300 cache=703.1MiB(4957210txo)
```

si buscamos interactuar con `bitcoin-cli` desde terminal, por ejemplo, con el comando `bitcoin-cli getblockchaininfo` obtenemos la siguiente respuesta.

```
❯ sudo su - bitcoin
┌─[bitcoin@parrot]─[~]
└──╼ $bitcoin-cli getblockchaininfo
{
  "chain": "test",
  "blocks": 2097260,
  "headers": 2417246,
  "bestblockhash": "000000000000009628c034f96e8474cdb79095
7c6c425b9fd",
  "difficulty": 61681740.29071233,
  "time": 1633132423,
  "mediantime": 1633131049,
  "verificationprogress": 0.943778822541906,
  "initialblockdownload": true,
  "chainwork": "000000000000000000000000000000000000000000000
5805000",
  "size_on_disk": 29985477788,
  "pruned": false,
  "warnings": ""
}
```

Si se quiere acceder desde otros usuarios distintos a `bitcoin` basta con crean un link simbólico de la carpeta en `/home/ghost/btc` a `/home/admin/.bitcoin`

```
ln -s /home/ghost/btc /home/ghost/.bitcoin
```

Note

- Se pueden correr distintas instancias de bitcoind en un mismo ordenador. Esto para tener en un mismo lugar mainnet y testnet por ejemplo.
- Para que no exista errores se debe tener especial cuidado con los puertos declarados en bitcoin.conf.
- Se pueden tener varios archivos de configuración, uno para cada red. Pero lo mejor es usar las etiquetas [main] [test] para ordenarlo todo en un mismo archivo de configuración.
- Al interactuar con bitcoin-cli con más de una red sincronizando al mismo tiempo, se debe explicitar la red al invocar un comando:
  mainnet -> bitcoin-cli getblockchaininfo
  testnet -> bitcoin-cli -testnet getblockchaininfo

```
sudo apt-get install python3 python3-dev python3-pip python3-setuptools python3-wheel
```