

Gerald Kaszuba

Programming, Technology, and Science.

Creating your own experimental Bitcoin network

Bitcoin is a decentralised digital currency which has been growing rapidly in popularity and use. You can send bitcoins to people and businesses around the world quickly, easily, and with significantly less fees than international wire transfers, PayPal, or credit cards. You can even buy beer almost instantly; Here is [a photo of me buying beer using Bitcoin](#).



If you're developing a Bitcoin application, or even want to just play around with it, you can do so without costing anything and use a minimal amount of system resources. This blog post covers some technical topics about the Bitcoin block chain, which you might want to [read up on the subject](#) beforehand.

The different Bitcoin networks

The standard Bitcoin client, Bitcoin Core has three networks it can run on: mainnet, testnet, and regtest.

mainnet

This is the real Bitcoin network, with the block chain that everyone uses. You can of course experiment with this network, but there are a few downsides:

- It's generally frowned upon to experiment on a production network. Even though it has been proven to be solid for 5 years, you might risk damaging it, or be wasting space on the block chain for no good reason.
- It costs real money—not only to buy bitcoins in the first place—but also transaction fees, even when sending bitcoins to yourself. If you were writing software that made transactions within unit tests, you could run out of bitcoins quite quickly.
- Block mining rate. Blocks are mined around every 10 minutes. It would be a waste of time to have to wait an hour for your 6 confirmations in a test environment.
- Synchronising the block chain. This behemoth is (as of April 2014) just under **20GB** in size and growing.

testnet

This is the test network that runs in parallel with mainnet, except that the value of these coins are negligible. It exists to experiment with a block chain that won't harm the mainnet block chain, e.g. with new features to Bitcoin Core. It also has the intention to be easier to mine blocks, and therefore it is essentially free to acquire testnet bitcoins.

Every now and then the developers nuke the ledger to prevent selling off testnet bitcoins, and also to reset the mining difficulty low enough so a CPU can effectively mine. Currently it is called **testnet3**.

The testnet runs on a different TCP port, has a different **genesis block**, and the Bitcoin addresses start with different letters. This is to make sure users don't accidentally shift between the two networks. Many consideration. Super developer. Wow.

Now for the downsides of using this for your own testing:

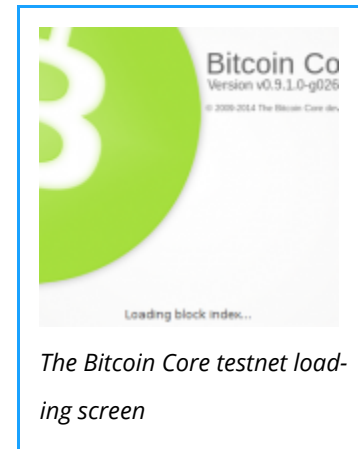
- It is difficult to mine testnet coins. While the Bitcoin Core developers suggest not to, I suspect people are using GPUs/ASICs to mine testnet blocks in order to get their test bitcoins faster. This causes the difficulty to go up so a standard laptop is not able to mine in a small amount of time. I ran mine for around 24 hours hoping to get one block, but nothing came through.
- The other way to get testnet coins is asking people and using testnet faucets. This is a great alternative, but doesn't really give you many bitcoins to test the new large-scale Bitcoin exchange you just made.
- The testnet3 block chain is also a mini-behemoth currently at **1.1GB**. You will need to download this but even at this smaller size it is *not cheap* when your ADSL+ is down because of incompetent telecommunications companies, and have to rely on expensive mobile data.

regtest

New in **Bitcoin Core 0.9.0**, the "Regression Test" mode allows mining with practically zero difficulty and is initially disconnected from any existing network. It also has its own genesis block, and when you start your client, no peers are registered, and nothing is initially mined by anyone.

The great thing about the regtest network is that you don't need to use a chunky amount of disk space, nor a massive amount of CPU power to mine thousands of test bitcoins. After the initialisation of the Bitcoin Core daemon, i.e. **bitcoind**, the whole regtest data directory is **17MB**. It takes around a minute and around **2MB** of extra disk space to mine 100 blocks.

This is the network we'll be using for testing.



Building the Bitcoin docker image

I have created a **Docker** image which simply allows you to run Bitcoin Core in a Docker container. You just need **docker installed** and to **clone the repository** I have set up.

Building the image takes a few minutes, so if you're doing this interactively, let's multi-task. After cloning the repository, hop into it and run:

```
1 # make build
```

make build hosted with ❤ by GitHub

[view raw](#)

The **Dockerfile** sets up the **Bitcoin PPA** which at the time of writing is version **0.9.0**. There is a version **0.9.1** which was released specifically to handle the **Heartbleed bug**. Apparently version **0.9.0** is safe from the vulnerability due to the fact that **it symbolically links to the OpenSSL library**. As long as your OpenSSL library is up to date, and you don't expose your RPC port, you should be safe.

We are going to run two docker instances, one for Alice and one for Bob. Both containers will run their own Bitcoin peer using **bitcoind**, and initially the only thing in common will be the regtest genesis block which is hard-coded in Bitcoin Core.

Your Own Bitcoin Network

After the docker image is built, load up two shells/terminals, preferably with a window split so you can see both at once. Run the following commands in their own windows:

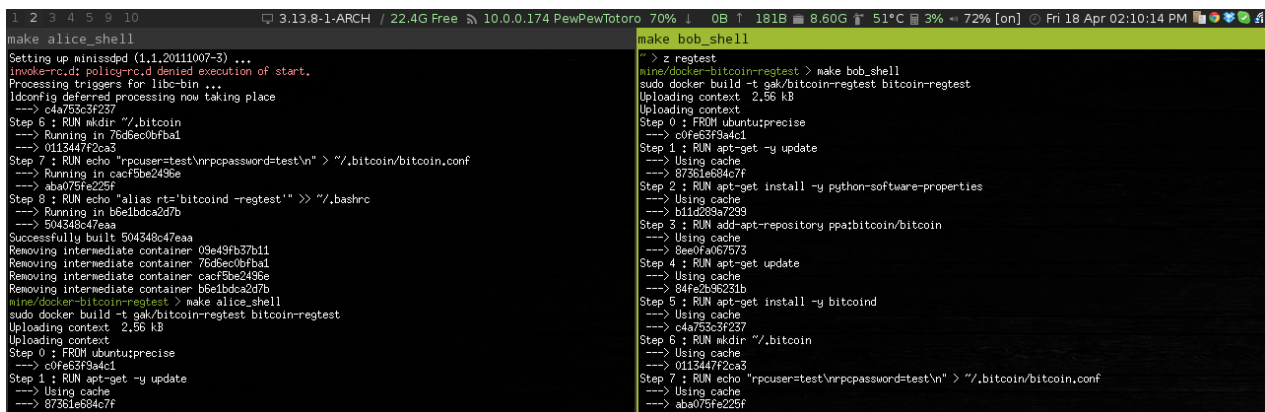
```
1 # make alice_shell
```

```
2 # make bob_shell
```

make shells hosted with ❤ by GitHub

[view raw](#)

Here's a screenshot of approximately what you should be seeing:



```
1 2 3 4 5 9 10
make alice_shell
Setting up minissdpd (1.1.20111007-3) ...
invoke-rc.d: policy-rc.d denied execution of start.
Processing triggers for libc-bin ...
ldconfig deferred processing now taking place
--> c4a753c3f237
Step 6 : RUN mkdir "/.bitcoin"
--> Running in 76d6ec0bfb1
--> 0113447f2ca3
Step 7 : RUN echo "rpcuser=test/vrpcpassword=test" > "/.bitcoin/bitcoin.conf"
--> Running in cacf9be2496e
--> aba079fe225f
Step 8 : RUN echo "alias rt='bitcoind -regtest'" > "/.bashrc"
--> Running in b6e1bdc2d7b
--> 504348c47aaa
Successfully built 504348c47aaa
Removing intermediate container 09e49fb37b11
Removing intermediate container 76d6ec0bfb1
Removing intermediate container cacf9be2496e
Removing intermediate container b6e1bdc2d7b
wine/docker-bitcoin-regtest > make alice_shell
sudo docker build -t gak/bitcoin-regtest bitcoin-regtest
Uploading context 2.56 kB
Uploading context
Step 0 : FROM ubuntu:precise
--> c0fe63f94d1
Step 1 : RUN apt-get -y update
--> Using cache
--> 87361e84c7f
Step 2 : RUN apt-get install -y python-software-properties
--> Using cache
--> b11d289a7299
Step 3 : RUN add-apt-repository ppa:bitcoin/bitcoin
--> Using cache
--> 8ee0f4d67573
Step 4 : RUN apt-get update
--> Using cache
--> 84fe2b36231b
Step 5 : RUN apt-get install -y bitcoind
--> Using cache
--> c4a753c3f237
Step 6 : RUN mkdir "/.bitcoin"
--> Using cache
--> 0113447f2ca3
Step 7 : RUN echo "rpcuser=test/vrpcpassword=test" > "/.bitcoin/bitcoin.conf"
--> Using cache
--> aba079fe225f
Step 8 : RUN echo "alias rt='bitcoind -regtest'" > "/.bashrc"
```

```

Step 2 : Run apt-get install -y python3-software-properties
--> Using cache
--> b11d289a7293
Step 3 : RUN add-apt-repository ppa:bitcoin/bitcoin
--> Using cache
--> 8ee9fa06573
Step 4 : RUN apt-get update
--> Using cache
--> 84fe2b96231b
Step 5 : RUN apt-get install -y bitcoind
--> Using cache
--> c4a753c3f237
Step 6 : RUN mkdir ~/.bitcoin
--> Using cache
--> 0113447f2ca3
Step 7 : RUN echo "rpcuser=test\nrpcpassword=test\n" > ~/.bitcoin/bitcoin.conf
--> Using cache
--> aba075fe225f
Step 8 : RUN echo "alias rt='bitcoind -regtest' >> ~/.bashrc
--> Using cache
--> 504348c47eaa
Successfully built 504348c47eaa
sudo docker rm -f alice
alice
sudo docker run -t -p 18444:18444 -p 18332:18332 --name=alice --hostname=alice -i galk/bitcoin-regtest bash
root@alice:/#

```

Once you're in, you should have **bitcoind** in your path. Start **bitcoind** on both containers like so:

```
1 root@alice:/# bitcoind -regtest -daemon -printtoconsole
```

run bitcoind hosted with ❤ by GitHub

[view raw](#)

I have set up an alias in the container's **.bashrc** called "rt" which is a short-cut to "bitcoind -regtest". The rest of the examples will use this. Using rt, this is the equivalent of the previous command:

```
1 root@alice:/# rt -daemon -printtoconsole
```

rt shortcut hosted with ❤ by GitHub

[view raw](#)

Here's what you should be seeing after running the bitcoind daemon with the **printtoconsole** option. The screenshot also shows that Alice and Bob's balances are both zero.

```

1 2 3 4 5 9 10
make alice_shell
keypool added key 70, size=70
keypool added key 79, size=79
keypool added key 80, size=80
keypool added key 81, size=81
keypool added key 82, size=82
keypool added key 83, size=83
keypool added key 84, size=84
keypool added key 85, size=85
keypool added key 86, size=86
keypool added key 87, size=87
keypool added key 88, size=88
keypool added key 89, size=89
keypool added key 90, size=90
keypool added key 91, size=91
keypool added key 92, size=92
keypool added key 93, size=93
keypool added key 94, size=94
keypool added key 95, size=95
keypool added key 96, size=96
keypool added key 97, size=97
keypool added key 98, size=98
keypool added key 99, size=99
keypool added key 100, size=100
keypool added key 101, size=101
keypool reserve 1
keypool keep 1
wallet 6002ms
init message: Loading addresses...
ERROR: CAddrman::Read() : open failed
Invalid or missing peers.dat: recreating
Loaded 0 addresses from peers.dat 0ms
mapBlockIndex.size() = 1
nBestHeight = 0
setKeyPool.size() = 100
mapWallet.size() = 0
mapAddressBook.size() = 1
ext-ip thread start
dnsseed thread start
Loading addresses from DNS seeds (could take a while)
0 addresses found from DNS seeds
dnsseed thread exit
net thread start
dumpaddr thread start
addcon thread start
opencn thread start
maghand thread start
init message: Done loading
GetMyExternalIP() received
GetMyExternalIP() returned
AddLocal(18444,4)
ext-ip thread exit
root@alice:/# rt getbalance
0.00000000
root@alice:/#

```

```

make bob_shell
keypool added key 70, size=70
keypool added key 79, size=79
keypool added key 80, size=80
keypool added key 81, size=81
keypool added key 82, size=82
keypool added key 83, size=83
keypool added key 84, size=84
keypool added key 85, size=85
keypool added key 86, size=86
keypool added key 87, size=87
keypool added key 88, size=88
keypool added key 89, size=89
keypool added key 90, size=90
keypool added key 91, size=91
keypool added key 92, size=92
keypool added key 93, size=93
keypool added key 94, size=94
keypool added key 95, size=95
keypool added key 96, size=96
keypool added key 97, size=97
keypool added key 98, size=98
keypool added key 99, size=99
keypool added key 100, size=100
keypool added key 101, size=101
keypool reserve 1
keypool keep 1
wallet 6113ms
init message: Loading addresses...
ERROR: CAddrman::Read() : open failed
Invalid or missing peers.dat: recreating
Loaded 0 addresses from peers.dat 0ms
mapBlockIndex.size() = 1
nBestHeight = 0
setKeyPool.size() = 100
mapWallet.size() = 0
mapAddressBook.size() = 1
ext-ip thread start
addcon thread start
opencn thread start
dumpaddr thread start
dnsseed thread start
Loading addresses from DNS seeds (could take a while)
0 addresses found from DNS seeds
dnsseed thread exit
net thread start
maghand thread start
init message: Done loading
GetMyExternalIP() received
GetMyExternalIP() returned
AddLocal(18444,4)
ext-ip thread exit
root@bob:/# rt getbalance
0.00000000
root@bob:/#

```

You can run some commands to check out the state of the regtest network. Right now there are no mined blocks, and no peers connected. For example:

```
1 root@alice:/# rt getinfo
2 keypool reserve 2
3 keypool return 2
4 {
5   "version" : 90000,
6   "protocolversion" : 70002,
7   "walletversion" : 60000,
8   "balance" : 0.00000000,
9   "blocks" : 0,
10  "timeoffset" : 0,
11  "connections" : 0,
12  "proxy" : "",
13  "difficulty" : 0.00000000,
14  "testnet" : false,
15  "keypoololdest" : 1397794246,
16  "keypoolsize" : 101,
17  "paytxfee" : 0.00000000,
18  "errors" : ""
19 }
20
21 root@alice:/# rt getpeerinfo
22 [
23 ]
24
25 root@alice:/# rt getblockhash 0
26 0f9188f13cb7b2c71f2a335e3a4fc328bf5beb436012afca590b1a11466e2206
```

initial regtest status hosted with ❤ by GitHub

[view raw](#)

The output of **getpeerinfo** above shows that there are no connected peers. Let's change that and make Bob get in touch with Alice. To tell bitcoind about Alice, we tell Bob to manually add Alice as a peer using Alice's IP address.

```
1 root@bob:/# rt addnode 172.17.0.2 onetry
```

add alice node hosted with ❤ by GitHub

[view raw](#)

A log entry in both peers should appear similar to the following. If not, try it again or wait a bit.

```
1 receive version message: /Satoshi:0.9.0/: version 70002, blocks=0, us=0.0.0.0:0, tl
```

new node log entry hosted with ❤ by GitHub

[view raw](#)

Now, when running **getpeerinfo** on Alice, we can see Bob as a listed peer:

```
1 root@alice:/# rt getpeerinfo
2 [
3   {
4     "addr" : "172.17.0.3:57631",
5     "services" : "00000001",
6     "lastsend" : 1397794619,
7     "lastrecv" : 1397794619,
8     "bytessent" : 297,
9     "bytesrecv" : 321,
10    "conntime" : 1397794619,
11    "pingtime" : 0.00000000,
12    "version" : 70002,
13    "subver" : "/Satoshi:0.9.0/",
14    "inbound" : true,
15    "startingheight" : 0,
16    "banscore" : 0,
17    "syncnode" : true
18  }
19 ]
```

getpeerinfo node hosted with ❤ by GitHub

[view raw](#)

Now that we have a two node Bitcoin network, let's mine some blocks!

Just on Alice, run **setgenerate**, and once that is done, check out your new, freshly mined **50** bitcoins:

```
1 root@alice:/# rt setgenerate true 101
2
3 ...
4
5 root@alice:/# rt getbalance
6 50.00000000
```

generate blocks hosted with ❤ by GitHub

[view raw](#)

Block rewards are released after 100 confirmations ([why?](#)), so on the 101st block mined, we should

have **50 BTC** available to spend. Here is a screenshot of the output of Alice and Bob, after Alice mining, and after Bob syncing the mined blocks.

Let's send some of this to Bob! On Bob's instance, let's grab a new address to receive to:

```
1 root@bob:/# rt getnewaddress
2 keypool added key 104, size=101
3 keypool reserve 4
4 keypool keep 4
5 mm47sczsEDYAGF2fp7xHSRpWQzCvpE1eZt
```

generate address hosted with ❤ by GitHub **view raw**

Notice how the address “**mm47sczsEDYAGF2fp7xHSRpWQzCvpE1eZt**” starts with an “m” instead of a “1”? This is because that extra safety feature I mentioned earlier to not to mix mainnet with the testing networks.

On Alice's instance, we can use this address to send **42** bitcoins to Bob:

```
1 root@alice:/# rt sendtoaddress mm47sczsEDYAGF2fp7xHSRpWQzCvpE1eZt 42
```

send coin hosted with ❤ by GitHub **view raw**

Now Alice has **8** bitcoins remaining, but Bob's **getbalance** command still shows **0**. This is because

```
1 root@alice:/# setgenerate true 6
```

[illegible]

Apart from unit testing, continuous integration, and staging environments, a configuration like this leads to many interesting things you can do. For example you can try the same experiment above without initially connecting the two nodes together. The beauty of the block chain is that Bob gets paid even if he's not connected to the network. He will be able to spend his new coins when he synchronises with the network.

The Makefile also has **alice_daemon** and **bob_daemon** commands which runs the bitcoind daemon detached as RPC servers, so you can experiment with an RPC client too. Please note that it will listen

to RPC on all ports, unencrypted, with the username test and password test. I highly recommend at least changing the username/password, and restricting remote access to the port.

The files used in this blog post [are on github](#) for your pleasure and I'm happy to accept pull requests for any issues or handy new features.

Happy hacking!

This entry was posted in Hacking and tagged bitcoin on April 18, 2014 [<https://geraldkaszuba.com/creating-your-own-experimental-bitcoin-network/>] by Gerald Kaszuba.