Qtum Blockchain Development Environment Setup

cryptominder (38) (/@cryptominder)in #qtum (/trending/qtum) • 3 years ago (edited)

Intro

In this post, I will describe how to quickly set up a private 3-node regtest environment for the Qtum blockchain on a **single** machine (e.g. on your laptop) using Docker.

This type of environment is useful for people who want to experiment (e.g. with staking) and/or develop **Ethereum smart contracts** on the Qtum blockchain.

The reason I chose to use Docker is to (later) simplify the installation of development tools such as **solc** and **ethabi** on different OSs (e.g. Windows, macOS/OSX, Linux, etc.)

I will be making follow-up posts aimed at prospective **Qtum blockchain software developers** which will require that you have this environment in place.

Assumptions

I will be making the following assumptions:

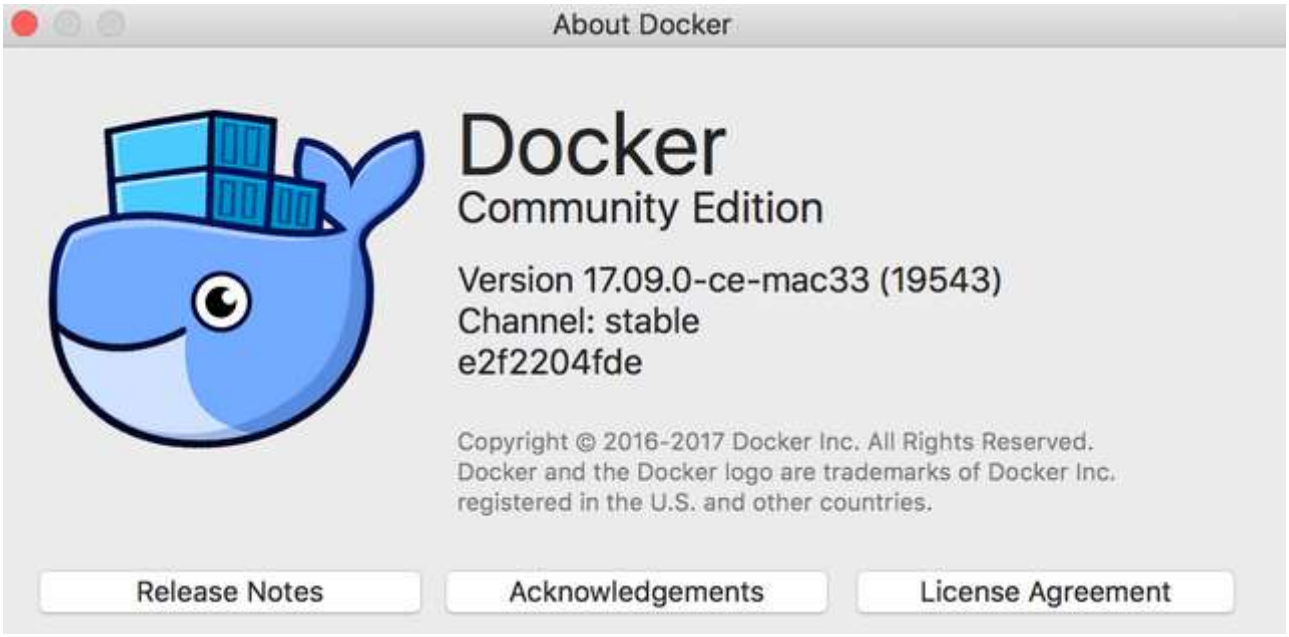
- You are a software developer (or, you at least know how to edit files, run commands in a shell/Terminal, have or can get `git` installed, etc.)
- You know that if I say "CTRL-D", that I mean pressing the Control key and the 'd' key at the same time, one after the other.
- My commands and output will be shown from a macOS/OSX environment, but you can translate the command/output to your own OS (e.g. Windows, Linux, etc.)
- You have approx. 200MB of free disk space to host the data for the 3 nodes.

I will be using the `$` character as a prompt for the commands given below. Your OS may have a different prompt (e.g. `>`). You will need to enter these commands in a shell/Terminal session (e.g. "Command Prompt" in Windows, etc.)

Step 1 - Installing Docker

There are many guides for installing Docker Community Edition (CE) on the web. For most people, the documentation at https://docs.docker.com/engine/installation/ should be sufficient.

I have the following Docker (CE) version installed and running on my laptop running macOS Sierra 10.12.6:



Running the following Docker command:

```
$ docker version
```

I get:

```
Client:
 Version:      17.09.0-ce
 API version:  1.32
 Go version:   go1.8.3
 Git commit:   afdb6d4
 Built:        Tue Sep 26 22:40:09 2017
 OS/Arch:      darwin/amd64

Server:
 Version:      17.09.0-ce
 API version:  1.32 (minimum version 1.12)
 Go version:   go1.8.3
 Git commit:   afdb6d4
 Built:        Tue Sep 26 22:45:38 2017
 OS/Arch:      linux/amd64
 Experimental: true
```

To ensure that you have Docker installed and running properly, run the following command:

```
$ docker run hello-world
```

You should see the following output:

```
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
5b0f327be733: Pull complete
Digest: sha256:b2ba691d8aac9e5ac3644c0788e3d3823f9e97f757f01d2ddc6eb5
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working corre

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker
 3. The Docker daemon created a new container from that image which r
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, whic
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://cloud.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/engine/userguide/
```

If you received an error, then ask the community for help.

Step 2 - Create a Qtum Docker Bridge Network

To allow the qtumd nodes (running in separate Docker containers) to communicate with each other, we'll create a Docker bridge network for them:

```
$ docker network create --driver bridge qtum_network
```

You'll be able to see this Docker network was properly created by running:

```
$ docker network ls
```

The output will look something like this:

```
NETWORK ID          NAME                DRIVER              SCOPE
6ed968118fba        bridge              bridge              local
07f2b230641a        host                host                local
4275590e2165        none                null                local
056c85d3e4a3        qtum_network        bridge              local
```

We'll reference the `qtum_network` network in commands used further below.

Step 3 - Pull the Qtum Docker Image

I created the following Qtum Docker Image: https://hub.docker.com/r/cryptominder/qtum/ . It uses the latest (as of Oct. 6th, 2017) Qtum Ignition (mainnet) v1.0.2 node/wallet binary .

Pull this image onto your machine:

```
$ docker pull cryptominder/qtum
```

After a few moments, you should see that it has downloaded the image, and its dependencies onto your machine. You can run:

```
$ docker images
```

to verify that the image (i.e. `cryptominder/qtum`) is indeed available locally.

Qtum Daemon Datadir as a Docker Volume

The `cryptominder/qtum` Docker image has a volume exposed at `/data`. We'll use a local directory (one for each node) to contain the Qtum **datadir** (e.g. where the blockchain data, the wallet file, the debug log, etc. is stored). This will allow you to stop and restart the Docker container without losing any data.

We'll also map a **local** `qtumd` **config file** to `/home/qtum/qtum.conf` (as a read-only file).

The above is achieved by using the following 2 command-line arguments to the `docker run` command (used in sections further below):

```
-v ${PWD}/node1_qtumd.conf:/home/qtum/qtum.conf:ro
-v ${PWD}/node1_data:/data
```

The `${PWD}` environment variable contains the current directory under Linux and macOS/OSX -- feel free to substitute this for your actual directory path. On **Windows**, this is typically equivalent to `%cd%`.

Step 4 - Obtaining Qtum Daemon Config Files

Before starting the Qtum regtest environment, let's first get some `qtumd` config files -- one for each of the 3 nodes.

I've made the 3 configuration files available at https://github.com/crypt0m1nd3r/docker-qtum-config . You can either manually download the 3 files from Git, or you can simply clone the repository:

```
$ git clone https://github.com/crypt0m1nd3r/docker-qtum-config.git
```

You should now have 3 files:

1. node1_qtumd.conf
2. node2_qtumd.conf
3. node3_qtumd.conf

The files are nearly identical, except for how they get each node to connect to another one (i.e. node1 connects to node2, node2 connects to node3, and node3 connects to node1).

You should take a moment to inspect these files. In particular, the `regtest=1` option activates the `regtest` blockchain mode.

Feel free to modify the config files if you prefer different settings.

Step 5 - Starting up

Now we're ready to start the 3 nodes to form a Qtum blockchain regtest network.

Start node #1 (all on 1 line):

```
$ docker run -d --rm --name qtumd_node1 --network=qtum_network -v
${PWD}/node1_qtumd.conf:/home/qtum/qtum.conf:ro -v
${PWD}/node1_data:/data cryptominder/qtum:latest qtumd
```

It is **very important** that the paths in the `-v` parameters are **absolute paths**. Again, on **Windows** you can replace `${PWD}` with `%cd%`.

You can check that node #1 is running:

```
$ docker ps
```

it should output (you can scroll this output to the left):

```
CONTAINER ID        IMAGE                          COMMAND
e1dd22bdf9c9        cryptominder/qtum:latest       "/entrypoint.sh qtumd"
```

You'll notice (in the last column above, at the very far right) that we named the Docker container for qtumd node #1: `qtumd_node1`.

`qtumd_node1` is using the config file `node1_qtumd.conf` (make sure that the file is present in the path provided), and it will use the **datadir** at `node1_data` in the current directory.

Check that the `${PWD}/node1_data` directory contains a sub-directory called `regtest` -- and that below the `regtest` sub-directory you find `wallet.dat`, `debug.log`, etc. Running:

```
$ ls -1FA node1_data/regtest
```

should output:

```
.cookie
.lock
banlist.dat
blocks/
chainstate/
database/
db.log
debug.log
peers.dat
qtumd.pid
stateQtum/
vm.log
wallet.dat
```

Perform the same steps as above (including the checks) for nodes #2 and #3, i.e.:

Start node #2 (all on 1 line):

```
$ docker run -d --rm --name qtumd_node2 --network=qtum_network -v
${PWD}/node2_qtumd.conf:/home/qtum/qtum.conf:ro -v
${PWD}/node2_data:/data cryptominder/qtum:latest qtumd
```

and

Start node #3 (all on 1 line):

```
$ docker run -d --rm --name qtumd_node3 --network=qtum_network -v
${PWD}/node3_qtumd.conf:/home/qtum/qtum.conf:ro -v
${PWD}/node3_data:/data cryptominder/qtum:latest qtumd
```

If you run:

```
$ docker ps
```

again, you should see (NOTE: the output below can be scrolled to the left):

```
CONTAINER ID        IMAGE                          COMMAND
6a9cf2ea8a6c        cryptominder/qtum:latest       "/entrypoint.sh qtumd"
7bc649de7649        cryptominder/qtum:latest       "/entrypoint.sh qtumd"
e1dd22bdf9c9        cryptominder/qtum:latest       "/entrypoint.sh qtumd"
```

Congratulations! -- you've just created a 3-node regtest Qtum blockchain environment.

Feel free to create a shell/batch script to automate this process.

Step 6 - Verification

To further verify that your regtest network is set up properly, use the `qtum-cli` command to invoke the RPC interface of **qtumd** on one or more of the nodes.

To use `qtum-cli` to call the RPC interface on **qtum_node1**, run:

```
$ docker run -i --network container:qtumd_node1 -v
${PWD}/node1_qtumd.conf:/home/qtum/qtum.conf:ro -v
${PWD}/node1_data:/data cryptominder/qtum:latest qtum-cli getinfo
```

You should see output similar to:

```
{
  "version": 140301,
  "protocolversion": 70016,
  "walletversion": 130000,
  "balance": 0.00000000,
  "stake": 0.00000000,
  "blocks": 0,
  "timeoffset": 0,
  "connections": 0,
  "proxy": "",
  "difficulty": {
    "proof-of-work": 4.656542373906925e-10,
    "proof-of-stake": 4.656542373906925e-10
  },
  "testnet": false,
  "moneysupply": 0,
  "keypoololdest": 1507353938,
  "keypoolsize": 100,
  "paytxfee": 0.00000000,
  "relayfee": 0.00400000,
  "errors": ""
}
```


There are 2 things to note about the command above:

1. The `-i` parameter is used. It is sometimes preferable to use the `-stdin` command-line argument to `qtum-cli` to input certain values (e.g. an account name that has a space, or a passphrase with spaces). The `-i` parameter to `docker run` makes it possible to accept this input.
2. The `--network container:qtumd_node1` parameter. This tells Docker to use the network associated with the running `qtumd_node1` container.

Here are other `qtum-cli` commands that you can run:

help:

```
$ docker run -i --network container:qtumd_node1 -v
${PWD}/node1_qtumd.conf:/home/qtum/qtum.conf:ro -v
${PWD}/node1_data:/data cryptominder/qtum:latest qtum-cli help
```

The above will show all available RPC commands.

getconnectioncount:

```
$ docker run -i --network container:qtumd_node1 -v
${PWD}/node1_qtumd.conf:/home/qtum/qtum.conf:ro -v
${PWD}/node1_data:/data cryptominder/qtum:latest qtum-cli
getconnectioncount
```

The above should output `2` (assuming the nodes have been up for a few minutes so they have a chance to connect to each other).

getaccountaddress:

```
$ docker run -i --network container:qtumd_node1 -v
${PWD}/node1_qtumd.conf:/home/qtum/qtum.conf:ro -v
${PWD}/node1_data:/data cryptominder/qtum:latest qtum-cli -stdin
getaccountaddress
```

After running the command above, hit enter (to get the default/empty "" account address of this node's wallet) followed by CTRL-D. This is an example of using the `-stdin` option to specify more advanced command-line parameters (one per line, until you press CTRL-D).

There are of course many other commands (you can see them when running `help`, see above).

The commands above used `qtumd_node1`. You can run these on the other nodes by changing the relevant parameter options on the command-line (i.e. turn the `1`'s into `2`s, etc.) You can check that each node are giving identical results for things like the block height (i.e. `"blocks": 0` from the `getinfo` command).

Again, feel free to create a shell/batch script (or alias) to simplify calling these commands.

Step 7 - Advancing to Proof-of-Stake (POS) Blocks

You may have noticed that the `getinfo` command in the previous step output: `"blocks": 0`. This means there are no blocks created yet on the our Qtum regtest blockchain.

In a `regtest` blockchain, you'll need to manually generate blocks. You can do this with the `generate` RPC command. Run `help generate` to see how it works:

```
$ docker run -i --network container:qtumd_node1 -v
${PWD}/node1_qtumd.conf:/home/qtum/qtum.conf:ro -v
${PWD}/node1_data:/data cryptominder/qtum:latest qtum-cli help
generate
```

TIP: You can use `help <command>` for other RPC commands too.

Now, let's generate 1 block:

```
$ docker run -i --network container:qtumd_node1 -v
${PWD}/node1_qtumd.conf:/home/qtum/qtum.conf:ro -v
${PWD}/node1_data:/data cryptominder/qtum:latest qtum-cli generate 1
```

Now, run the `getinfo` command again:

```
$ docker run -i --network container:qtumd_node1 -v
${PWD}/node1_qtumd.conf:/home/qtum/qtum.conf:ro -v
${PWD}/node1_data:/data cryptominder/qtum:latest qtum-cli getinfo
```

You should see that it outputs:

```
"blocks": 1,
```

In Qtum, **the first 5000 blocks are Proof-of-Work (POW) blocks.** I recommend that you advance the block height to (at least) 5000 so that you enter into the **Proof-of-Stake (POS) blocks.** If you generated 1 block above, you'll need to now generate 4999 more blocks:

```
$ docker run -i --network container:qtumd_node1 -v
${PWD}/node1_qtumd.conf:/home/qtum/qtum.conf:ro -v
${PWD}/node1_data:/data cryptominder/qtum:latest qtum-cli generate
4999
```

Running the command above will take **2 or 3 minutes** to complete. When it's done, you'll see a lengthy output (i.e. a JSON document containing one line for each of the 4999 generated block hashes).

UPDATE: Jordan Earls , the leader developer for Qtum informed me of the following: "*This is actually not quite true. PoS and PoW blocks are mixed below block 5000, and after block 5000 PoW is banned. However, in regtest mode, there is no PoW or PoS limit. So, as long as you generate at least 500 blocks (because coins can't be used for staking until 500 blocks of maturity), then you can generate pretty much any amount. I usually use 600. Also, at very large block generations like 5000 sometimes you can have trouble getting PoS to start for a couple of hours due to some technicalities related to timestamp enforcement.*"

If we again run the `getinfo` command:

```
$ docker run -i --network container:qtumd_node1 -v
${PWD}/node1_qtumd.conf:/home/qtum/qtum.conf:ro -v
${PWD}/node1_data:/data cryptominder/qtum:latest qtum-cli getinfo
```

You should see that it now outputs:

```
"blocks": 5000,
```

We're now in **Proof-of-Stake (POS)** territory.

IMPORTANT NOTE: In the Qtum **regtest** environment, the POS blocks will automatically generate themselves. It takes a little while (e.g. about 10 minutes) to generate the 5001st block -- but afterwards the blocks will be created fairly quickly. You can still manually generate POS blocks if you want.

After you generated blocks, you'll also notice that your wallet balance has increased (use `getbalance` to check).

Step 8 - Stopping and Resetting

To stop a node, simply run:

```
$ docker stop qtumd_node1
```

(obviously change `qtumd_node1` to the node you want to stop)

If you wish to completely reset the blockchain, you can simply delete each of the 3 data directories (e.g. `node1_data/regtest`, etc.) after you have stopped the nodes.

Troubleshooting

In case you want to "login" to a running container, the following command is helpful:

```
$ docker exec -it <CONTAINER ID> /bin/sh
```

The `<CONTAINER ID>` part can be the container id (as shown with `docker ps`) or the name we've given our containers (e.g. `qtumd_node1`).

When you're done looking around, just type:

```
# exit
```

to leave the container (it will still keep running).

What's Next?

This is just the beginning. I will be writing follow-up guides for developing Ethereum (EVM) **smart contracts** over Qtum using this 3 node regtest environment. Docker will also be used for compling Solidity code using **solc**. I'm also looking at creating a Docker image for **ethabi**.

The `qtum-cli` tool (as we used in this guide) will also be used for creating and invoking smart contracts.

I may also add (in Github) shell/batch scripts to help simplify calling the commands above, as well as providing instructions for using Docker Compose to bring up the 3 node Qtum regtest environment.

I hope this guide was helpful to you. If you need help, please use the comment section below, or look for me on the Qtum subreddit. I'm also active on Qtum's Slack (although Slack invitations are closed as the Qtum team prepares to move to another collaboration/messaging platform).

If you found this guide useful, and you are so inclined, my Qtum tip wallet address (running on my Raspberry Pi 3 (https://steemit.com/qtum/@cryptominder/qtum-staking-tutorial-using-qtumd-on-a-raspberry-pi-3)) is: **QUa3yA8ALfQyM5eEb9sDPRWkX6sSurMs6D** . I appreciate your support!

#blockchain (/trending/blockchain) #smartcontract (/trending/smartcontract) #docker (/trending/docker) #regtest (/trending/regtest)

3 years ago in #qtum (/trending/qtum) by **cryptominder** (38) (/@cryptominder)

$0.15 | 16 votes

Reply | 8 (/qtum/@cryptominder/qtum-blockchain-development-environment-setup)

Sort: Trending

piotr-galas (48) (/@piotr-galas) 2 years ago (/qtum/@piotr-galas/re-cryptominder-qtum-blockchain-development-environment-setup-20180128t220747676z#@piotr-galas/re-cryptominder-qtum-blockchain-development-environment-setup-20180128t220747676z) [-]

I am stuck on Step related to generate blocks. I get error

```
error code: -1
error message:
CreateNewBlock: TestBlockValidity failed:  (code 0)
```

I suppose that this occure becouse after start my container I have about 90k blocks. If somebody know how to solve this problem please reply.

$0.00 | 1 vote | Reply

reblogger (-2) (/@reblogger)(1) 3 years ago (/qtum/@reblogger/re-cryptominder-qtum-blockchain-development-environment-setup-20171007t072055717z#@reblogger/re-cryptominder-qtum-blockchain-development-environment-setup-20171007t072055717z) [-]

Hi, cryptominder! I just resteemed your post!
I am a new, simple to use and cheap resteeming bot.
If you want to know more about me, read my introduction post (https://steemit.com/resteembot/@reblogger/reblogger-new-resteeming-bot-based-on-resteembot).
Good Luck!

$0.00 | Reply

rahulsharmaitm (37) **(/@rahulsharmaitm)** 3 years ago (/qtum/@rahulsharmaitm/re-cryptominder-qtum-blockchain-development-environment-setup-20171007t073232974z#@rahulsharmaitm/re-cryptominder-qtum-blockchain-development-environment-setup-20171007t073232974z) [-]

Very informative

$0.00 Reply

ta-da (25) **(/@ta-da)** 3 years ago (/qtum/@ta-da/re-cryptominder-qtum-blockchain-development-environment-setup-20171014t051726803z#@ta-da/re-cryptominder-qtum-blockchain-development-environment-setup-20171014t051726803z) [-]

Thanks man. It is exactly what I was looking for. +1

$0.00 1 vote Reply

elbapu (25) **(/@elbapu)** 3 years ago (/qtum/@elbapu/re-cryptominder-qtum-blockchain-development-environment-setup-20171017t174836946z#@elbapu/re-cryptominder-qtum-blockchain-development-environment-setup-20171017t174836946z) (edited) [-]

Hi cryptominder! First, again thanks for these tutorials!
I'm using the raspberry pi3 for setting up this envieronment. while following the guide, I'm having trouble on step3: this part:

"Qtum Daemon Datadir as a Docker Volume
-v ${PWD}/node1_qtumd.conf:/home/qtum/qtum.conf:ro
-v ${PWD}/node1_data:/data"

When I type that (replacing the PWD for my home/pi, i get this message: docker run requires at least one argument. Run a command in a new container.

What am I doing wrong? Is it the same procedure for the raspberry?
Thanks in advanced

$0.00 Reply

piotr-galas (48) **(/@piotr-galas)** 2 years ago (/qtum/@elbapu/re-cryptominder-qtum-blockchain-development-environment-setup-20171017t174836946z#@piotr-galas/re-elbapu-re-cryptominder-qtum-blockchain-development-environment-setup-20180128t214649975z) [-]

This is not valid command to run. Full command to copy/paste is under # step 5 section

$0.02 1 vote Reply

skyline-team (25) **(/@skyline-team)** 2 years ago (/qtum/@skyline-team/re-cryptominder-qtum-blockchain-development-environment-setup-20180105t035932473z#@skyline-team/re-cryptominder-qtum-blockchain-development-environment-setup-20180105t035932473z) [-]

hi, guys, I am running into some basic issues here - after in Step#3, pull down the docker image, but I just can NOT find this directory location

/home/qtum

I am on Mac Air.

can one of you give some hints - what went wrong ?

thanks for helping !!

$0.00 | Reply

piotr-galas (48) **(/@piotr-galas)** 2 years ago (/qtum/@skyline-team/re-cryptominder-qtum-blockchain-development-environment-setup-20180105t035932473z#@piotr-galas/re-skyline-team-re-cryptominder-qtum-blockchain-development-environment-setup-20180128t212936163z) [-]

In step#3 author only inform about some directories. Follow him to net step and just copy/paste his commands. directory `/home/qtum` live inside a docker container.

$0.02 | 1 vote | Reply