FunctionX testnet node setup tutorial

1 Introduction

FunctionX is a new generation blockchain based on Cosmos SDK and Tendermint, which currently serves as a support to the PundiX blockchain and terminals and point-of-sales.

Reference: https://functionx.gitbook.io/home/

2 FunctionX testnet node

2.1 Setup a FunctionX testnet node on a Raspberry Pi

2.1.1 Before you start

The components required below are good items to enjoy your node for a quite long time.

Note that you can opt for a cheaper RPi4 (4 GB recommended) or for a smaller SSD (500GB recommended).

Also, it is highly recommended to have a good enclosure as well as a fan for the RPi, and to always store it in a cool environment with enough space to dissipate heat.

However, the FunctionX blockchain might grow faster and faster and smaller or cheaper components might not be fitted for a long-term use.

The MicroSD card will just be used temporarily : you will be able to reuse it somewhere else later on.

Once everything is between your hands, please note the following procedure can take 2 to 4 hours to execute depending on your skills and your internet speed. The synchronization of your node with the FxCore network might take longer though !

Please note that this setup incurs only non-recurring costs of 250-350$ (at time of writing) + 15W electricity consumption and Internet access. This is especially interesting compared to the Cloud setup that will incur recurring costs of approximately 80-100$ per month. However, this Raspberry Pi setup will require regular updates and might need to be restarted properly in case of unexpected electrical shutdowns and/or unjailed.

2.1.2 Requirements

1) Hardware (~ 250-350 $ of non-recurring costs) :
    a. Raspberry Pi 4 8GB with 5V 3A power supply ($140)

    ⚠ This tutorial will not work with previous versions of Raspberry Pis, unless you already reprogrammed the bootloader to boot from USB via Raspbian OS.

    b. HDMI (female) to Micro-HDMI (male) Adapter (included in (1.a))
    c. HDMI cable (< $10)
    d. Crucial BX500 2TB Solid-State drive ($180)
    e. USB 3.0 to SATA3 cable (< $10)
    f. A MicroSD card with adapter (> 8GB) (< $10)
2) Software:
    a. File to MicroSD card image writer : BalenaEtcher : to transfer the 64-bits Linux distribution file to the SDCard
        i. Can be replaced by any software capable to "burn" a Linux distribution file to a hard drive or SDCard.
    b. Linux distribution Ubuntu Server 20.04.x LTS image archive : currently 20.04.3 recommended
    c. The FxCore latest snapshot (to be downloaded later on) : https://fx-testnet.s3.amazonaws.com/fxcore-snapshot-2021-11-08.tar.gz (or any later monday)
3) A computer with Internet access
4) Access to Internet (through a Wifi network – 2.4 GHz or 5 GHz – or an Ethernet cable network)

2.1.3 Setup your SDCard

Everything has to be done from your local computer:

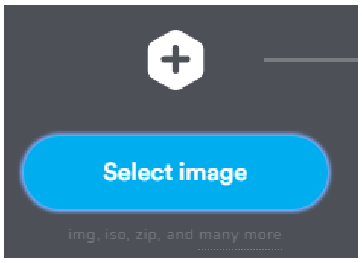
a. Download BalenaEtcher (2.a) and Linux distribution – Ubuntu Server 20.04.x LTS for ARM64 – (2.b).
b. Insert your MicroSD card directly, or using the SDCard adapter, into your computer.

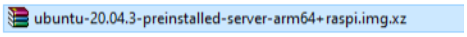
⚠ Do not plug your SSD at this time!

c. Run BalenaEtcher :
    1. Click "*Select image*" : select the Linux image file (e.g. "*ubuntu-20.04.3-preinstalled-server-arm64+raspi.img.xz*") you just downloaded

    ⚠ Make sure it's finished downloading.

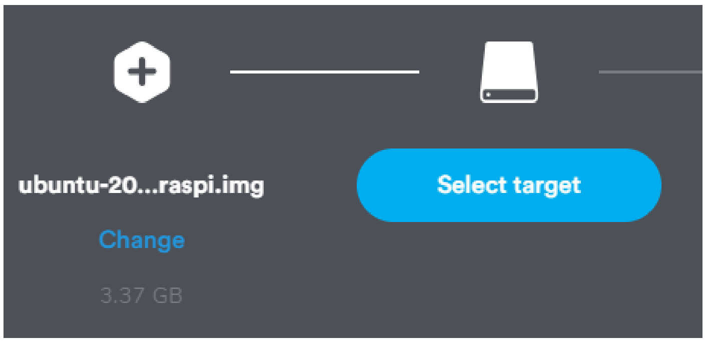
    ⚠ Make sure you downloaded the "-arm64-" version

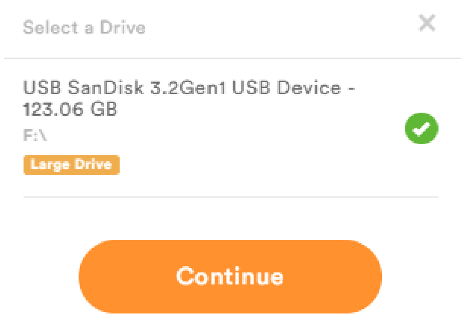




2. Select the drive where you want this file to be written and click "*Continue*"

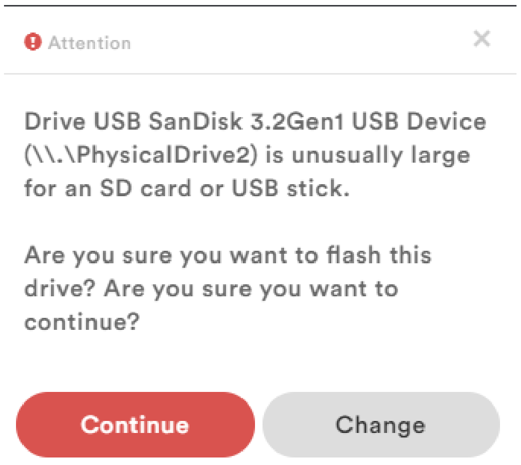
⚠ It should be the drive where your MicroSD card has been found.
⚠ Make sure you do not write this file on another drive : it might erase your data or system partition on your computer otherwise!





3. Click *"Flash!"*

⚠ If you get the message below, check again that you did select the SDCard (the one you want to install FxCore onto) : if that's OK, click "Continue". Otherwise, click "Change".



Attention

Drive USB SanDisk 3.2Gen1 USB Device (\\.\PhysicalDrive2) is unusually large for an SD card or USB stick.

Are you sure you want to flash this drive? Are you sure you want to continue?

4. Once you clicked “Continue” (and eventually allowed for execution on “*User Account Control*” feature of Microsoft Windows), the software will start flashing your SDCard and will validate it.
5. Once finished and verified, close BalenaEtcher and “extract” your Micro/SDCard from your computer.

2.1.4 Finalize setup of the Ubuntu system on your Raspberry Pi

At first start, it will be possible to access your Rapsberry Pi through your local network using SSH (if it’s plugged using a wired network). Also, it will only work when booting from a MicroSD card.

However, a MicroSD card is not adapted to “speedy” read/write transactions as required by a FunctionX testnode.

Moreover, your (WiFi) network is not configured yet.

We are going to setup everything : update the Raspberry Pi, change the keyboard language, the Wifi network (if you need it), allow it to start from a SSD, etc.

1. **Before anything**

   ⚠ Your Raspberry Pi must be **<u>powered off</u>** before you start. Power-on will happen later.

   Insert the MicroSD card into the Raspberry.

2. **Plug the Raspberry Pi to your monitor or TV**

   Plug the MicroHDMI to HDMI adapter onto your Raspberry, the HDMI cable to any TV

3. **Power on your Raspberry Pi**

   · You should soon see the rainbow screen and a lot of scrolling lines.

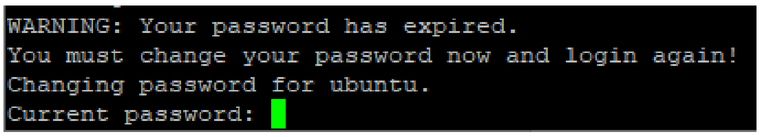
   · The default login/password used by Linux Ubuntu is : “ubuntu” / “ubuntu”

   ⚠ Do NOT forget to change the default password later on.

4. **Change Raspberry Pi main password**

   The following line appears after the Raspberry Pi initialized (remember the default login/password are ubuntu/ubuntu), you will need to change the Ubuntu password.

   Follow the instructions below and replace the stuff in brackets with the current password and the new password.



```
WARNING: Your password has expired.
You must change your password now and login again!
Changing password for ubuntu.
Current password:
```

```
Current password: [type “ubuntu” without the quotes]
New password: [type your new password]
Retype new password: [type your new password again]
```

   If the “*Your new password has expired*” line does not show up, you will have to provoke it by executing the following command and following instructions above.

```
passwd
```

5. **Update the Raspberry Pi**

   Execute the following commands in the terminal:

```
sudo apt update && sudo apt upgrade -y && sudo apt full-upgrade
sudo reboot
```

6. **Install libraries**

   Execute the following commands in the terminal:

```
sudo apt install -y make gcc net-tools atop iotop libraspberrypi-bin
rpi-eeprom
```

7. **Change the system language (optional)**

   The default language of Ubuntu is English. You can change the entire system language or just the keyboard language. If you want to change the system language, language packages must first be installed.

   To find the correct files for you, execute the following command by replacing the word "*French*" (keep the quotes) by the language name (in English) you're looking for :

```
sudo apt-cache search language-pack-* | grep "French"
```

When you found the correct file, execute the following command (here, those are the 2 files to install for "French" language with "fr" keyword) :

```
sudo apt install language-pack-fr language-pack-fr-base
```

- Others are "de" for German, "nl" for Dutch, "af" for "Afrikaans", "id" for Indonesian, "ja" for Japanese

To install the language:

```
sudo dpkg-reconfigure locales
```

This opens a list of all available languages. Select the one you want to activate (an "X" will show up beside the language name) and unselect the ones (the "X" will disappear beside the language name) you don't want anymore.

To activate the new language setting, the system must be restarted:

```
sudo reboot
```

8. **Change the keyboard language**

   If you only want to change the keyboard layout, you can do this with the following command:

```
sudo dpkg-reconfigure keyboard-configuration
```

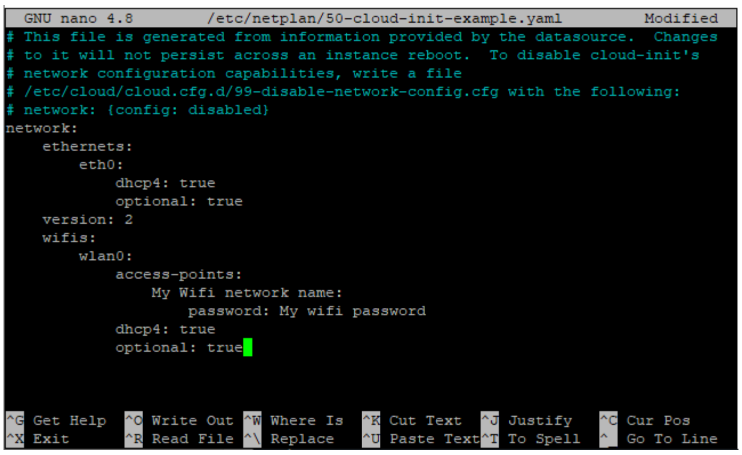
Here too, the system needs to be restarted after making the changes.

```
sudo reboot
```

9. **Configure your Wifi (optional if already connected by Ethernet)**
Execute the following command:

```
sudo nano /etc/netplan/50-cloud-init.yaml
```

The following screen will give you an example of what will show up. Complete the file as necessary, using 4-spaces indentation (very important!!).
Replace "*My Wifi network name*" with your own Wifi network name.
Replace "*My wifi password*" with your own Wifi password.



```
  GNU nano 4.8          /etc/netplan/50-cloud-init-example.yaml          Modified
# This file is generated from information provided by the datasource.  Changes
# to it will not persist across an instance reboot.  To disable cloud-init's
# network configuration capabilities, write a file
# /etc/cloud/cloud.cfg.d/99-disable-network-config.cfg with the following:
# network: {config: disabled}
network:
    ethernets:
        eth0:
            dhcp4: true
            optional: true
    version: 2
    wifis:
        wlan0:
            access-points:
                My Wifi network name:
                    password: My wifi password
            dhcp4: true
            optional: true

^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text^T To Spell    ^_ Go To Line
```

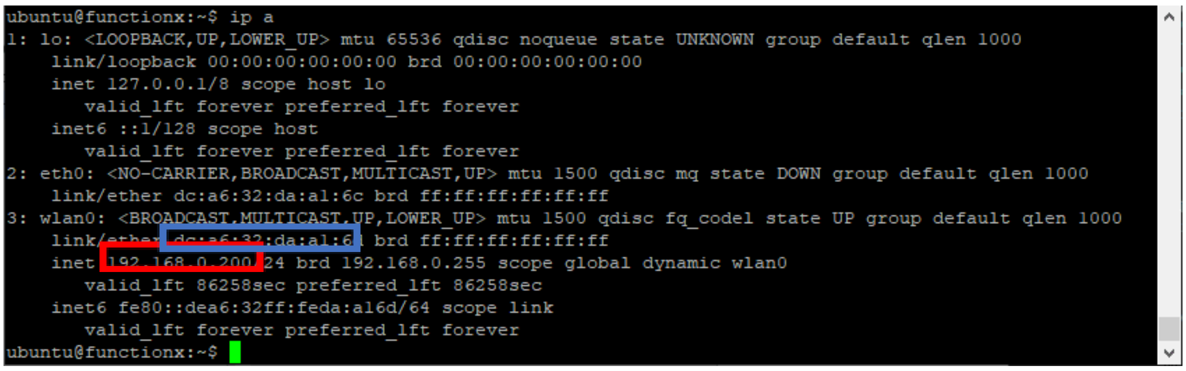
Then type « CTRL » + X then "Y" + "ENTER" : this will save the file and exit.
Execute the following command to check your configuration :

```
sudo netplan --debug apply
```

If everything went fine, you should have no errors : the wifi connection should be established and be connected to the wifi network you specified. To confirm the wifi connection, follow the next section which should show you your IP address in the WLAN (wireless) section.

10. **Accessing Raspberry Pi and Ubuntu with SSH**
The SSH server is installed and activated by default on Linux Ubuntu. SSH is very practical if you don't want to plug your Raspberry Pi to a screen but still want to control it from a simple computer on your local network. If you don't know the IP address of your Pi, you can find out what it is with the following command:

```
ip a
```



```
ubuntu@functionx:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN group default qlen 1000
    link/ether dc:a6:32:da:a1:6c brd ff:ff:ff:ff:ff:ff
3: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether dc:a6:32:da:a1:6d brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.200/24 brd 192.168.0.255 scope global dynamic wlan0
       valid_lft 86258sec preferred_lft 86258sec
    inet6 fe80::dea6:32ff:feda:a16d/64 scope link
       valid_lft forever preferred_lft forever
ubuntu@functionx:~$
```

The IP address of your Pi is the one in the red box.
The MAC address of your Pi is the one in the blue box.
You can also find the IP address through your router or DHCP server, by looking at the IP address assigned to your Pi's MAC address.
Please note the IP address your Raspberry was assigned on your local network might change over time, and the best way to find it will be using your router or DHCP server.

11. Allow the Raspberry Pi to boot from USB

Reference: website

⚠ If you're using a Raspberry Pi < 4 AND you already reprogrammed the bootloader to boot from USB, you can jump this step 11 !

a) Check current running version of bootloader EEPROM image

First thing to check is your current running version of the bootloader EEPROM image:

```
vcgencmd bootloader_version
```

USB booting was supported starting with version 9th march 2020. You should see that version "*Mar 9 2020*" or later is installed. Now check to see if there is an update available.

```
sudo rpi-eeprom-update
```

You should see that the bootloader is up-to-date. By default, you will only see updates from the "default" directory. Default updates are not released too often. Even if you see "***UPDATE AVAILABLE***", you can ignore it as we are going to check for an update in a more frequently released channel or directory called "stable".

b) Upgrade to the latest "stable" EEPROM version

You will need to change the firmware directory from critical to stable to find the latest updates. To do this, you need to edit a file called "*/etc/default/rpi-eeprom-update*".

```
sudo nano /etc/default/rpi-eeprom-update
```

Change the word "*critical*" or "*default*" right after "*FIRMWARE_RELEASE_STATUS*" to "*stable*" and save the file.

Check once again for an EEPROM update and you should notice a newer version.

```
sudo rpi-eeprom-update
```

c) Apply the update and reboot

```
sudo rpi-eeprom-update -a
sudo reboot
```

You can check the running EEPROM version again to ensure the correct version.

```
vcgencmd bootloader_version
```

d) Check BOOT ORDER configuration

```
sudo rpi-eeprom-config
```

Take note of the BOOT_ORDER code. The default code is 0xf41 and is read **right-to-left** to determine the boot order.

1 = Check SD card

4 = Check USB drive

f = Start again

This boot order is exactly what we want. If the boot order code is anything other than 0xf41, you can change the boot configuration using this command.

```
sudo -E rpi-eeprom-config --edit
```

2.1.5 Install FxCore testnode

2.1.5.1 Install Go

Let’s start by installing Go 1.17.2 for Raspberry Pi and configure it.

```
wget https://dl.google.com/go/go1.17.2.linux-arm64.tar.gz
sudo rm -rf /usr/local/go && sudo tar -C /usr/local -xzf
go1.17.2.linux-arm64.tar.gz
export PATH=$PATH:/usr/local/go/bin
mkdir -p $HOME/go/bin
echo "export PATH=$PATH:/usr/local/go/bin" >> ~/.bash_profile
echo "export PATH=$PATH:$(go env GOPATH)/bin" >> ~/.bash_profile
source ~/.bash_profile
```

2.1.5.2 Install the binaries

Next, let's install the latest version of f(x)Core. Type the following command:

```
git clone -b testnet https://github.com/functionx/fx-core.git
cd fx-core
make go.sum && make install
```

That will install the “*fxcored*” binary. Verify that everything is OK by typing the following command:

```
fxcored version --long
```

For instance, it should output something similar to:

```
name: fxcore
server_name: fxcored
version: testnet-76e025af60f1cff541413c8521c8893f6e2680d2
commit: 76e025af60f1cff541413c8521c8893f6e2680d2
build_tags: netgo,ledger
go: go version go1.17.2 linux/arm64
build_deps:
[...]
```

2.1.5.3 Setup the full testnode (with Binaries)

Initialize fxcore by typing the following command :

```
fxcored init fx-zakir
```

Fetch the configuration files by typing the following 3 commands:

```
wget
https://raw.githubusercontent.com/functionx/fx-core/testnet/public/gen
esis.json -O ~/.fxcore/config/genesis.json
wget
https://raw.githubusercontent.com/functionx/fx-core/testnet/public/con
fig.toml -O ~/.fxcore/config/config.toml
wget
https://raw.githubusercontent.com/functionx/fx-core/testnet/public/app
.toml -O ~/.fxcore/config/app.toml
```

2.1.6 Download the latest FX Testnet snapshot

```
cd ~
wget -c
https://fx-testnet.s3.amazonaws.com/fxcore-snapshot-2021-11-08.tar.gz
tar -xzvf fxcore-snapshot-2021-11-08.tar.gz -C ~/.fxcore/
```

⚠ The snapshot file is pretty heavy (~20GB) : be patient (6-12 hours for up to 7 days syncing from the latest snapshot) and make sure it's fully downloaded before proceeding to the next step.

2.1.7 Register FxCore to start as a service (daemon)

First, create a service definition file:

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

Then, power on your Raspberry Pi.

Copy the following lines into the editor :

```
[Unit]
Description=f(x)Core NodeAfter=network.target
[Service]
Type=simple
User=ubuntu
WorkingDirectory=/home/ubuntu
ExecStart=/home/ubuntu/go/bin/fxcored start --home
/home/ubuntu/.fxcore
Restart=on-failure
RestartSec=3
LimitNOFILE=4096
[Install]
WantedBy=multi-user.target
```

When done, type [CTRL+X] then type "Y" (for yes).

After creating that service definition file, you need to enable the service to start.

```
sudo systemctl daemon-reload
sudo systemctl enable fxcored
```

It is always better to sync f(x)Core using the Daemon method because this ensures stability and that your syncing is continuously running in the background.

2.1.8 Transfer content of the SDCard onto the SSD drive

You first need to identify your USB device name under Linux.

Type the following command:

```
lsblk
```

You should get something like the following:

```
NAME         MAJ:MIN RM   SIZE  RO TYPE MOUNTPOINT
loop0          7:0    0  48,9M  1 loop /snap/core18/2127
loop1          7:1    0    49M  1 loop /snap/core18/2248
loop2          7:2    0  57,4M  1 loop /snap/core20/1171
loop3          7:3    0  60,7M  1 loop /snap/lxd/21804
loop4          7:4    0  28,2M  1 loop /snap/snapd/13643
loop5          7:5    0  36,5M  1 loop /snap/snapd/13830
loop6          7:6    0  60,7M  1 loop /snap/lxd/21843
sda            8:0    1 114,6G  0 disk
└─sda1          8:1    1 114,6G  0 part
mmcblk0      179:0    0  58,9G  0 disk
├─mmcblk0p1 179:1     0   256M  0 part /boot/firmware
└─mmcblk0p2 179:2     0  58,6G  0 part /
```

You notice there are 2 disks : one is a “sdX” (your USB drive) and the other a “mmcblk0” (your micro SDCard). Retain the name of the “sdX” for use in the following command line.

```
sudo dd if=/dev/mmcblk0 of=/dev/sda bs=1M status=progress
```

⚠ Do not process any operation on the Raspberry Pi while this operation is ongoing.

```
sudo shutdown now
```

2.1.9 Switch to USB

Once your Raspberry Pi is shut down, remove your SD card and make sure your USB SSD is still plugged in.

Then, power on your Raspberry Pi. It should boot as it used to boot before.

2.1.10 Wait for the testnet full node to fully sync

Check the status of the synchronization regularly:

```
curl localhost:26657/status
```

⚠ “catching_up” will be true for as long as your node is synchronizing.
Do not proceed the next step until “catching_up” shows up as **false**.

2.1.11 Configure your own full testnet full node

Now that your node is synced, you can configure it.

In the following lines of command, replace “my-node-name” with your node name (keeping the quotes), “my-http-link” with a URL pointing to a description of your node or yourself, and “My node description” with a description of your node (e.g. where it’s located, etc.)

```
export NODE_NAME="my-node-name"
export HTTP_LINK="my-http-link"
export NODE_DESCRIPTION="My node description"
fxcored keys add "$NODE_NAME"
```

The terminal will return you something like this. Make sure to backup all parameters figuring in blue and red below.

```
- name: test
  type: local
  address: fx1lpckl9eekevmsg34c9g0ea93j2yju73tu6vd4x
  pubkey:
fxpub1addwnpepq2yhl00ht0yhpwj6a2sv3fevv9velyd8h0kczcltjs98skkcyhv37am6
jla
  mnemonic: ""
  threshold: 0
  pubkeys: []
**Important** write this mnemonic phrase in a safe place.
It is the only way to recover your account if you ever forget your
password.

ship huge hungry hair timber mansion issue company napkin clay lounge
social wild sign stick humor idle increase remember execute use panel
insect blanket
```

⚠ Data figuring in **red** above will only be accessible once. Please make sure to back it up somewhere safe. And never share the 24-words seed-phrase with anyone ! Never ! Noone !!!!

Before proceeding with the next step, make sure to transfer at least 100 TESTNET FX (and not real FX!) to the testnet account address you just got (**fx1lpckl9...73tu6vd4x** above).

You can get those Testnet here : https://aabbcc-faucet.function.io

You can check if your testnet account has the necessary testnet FX by checking here : https://aabbcc-explorer.functionx.io and then entering your testnet account address.

Once the transfer is done, we are going to bind our validator full node with the key we just created:

```
fxcored tx staking create-validator --chain-id=fxcore --gas="auto"
--gas-adjustment=1.2 --gas-prices="6000000000000FX"
--from="$NODE_NAME" --amount=100000000000000000000FX
--pubkey=$(fxcored tendermint show-validator) --moniker=" $NODE_NAME"
--commission-rate="0.001" --commission-max-rate="0.200"
--commission-max-change-rate="0.003"
```

```
--min-self-delegation="1000000000000000000000" --moniker="$NODE_NAME"
--website="$HTTP_LINK" --details="$NODE_DESCRIPTION"
```

2.2 Frequently asked questions

2.2.1 What's the status of my node ?

Solution (a) - Type the following command on the Linux terminal hosting your node:

```
sudo systemctl status fxcored
```

You will get something like this:

```
● fxcored.service - f(x)Core Node
     Loaded: loaded (/etc/systemd/system/fxcored.service; enabled;
vendor preset: enabled)
     Active: inactive (dead)
```

If you see `inactive (dead)`, then that means your service is not started : you should reboot your Raspberry Pi and see if it's still happening.

Solution (b) - You can also check by entering the following command:

```
ps -ef | grep fxcored
```

You should see 2 lines: the grep command itself, and the fxcored daemon:

```
ubuntu      3217    9999 99 18:59 pts/2    01:23:45 fxcored start
ubuntu      3218    9999  0 19:00 pts/0    00:00:00 grep fxcored
```

If you don't see the line containing "*fxcored start*", that means your FX validator node is not started : you should reboot your Raspberry Pi and see if it's still happening.

Solution (c) – You can also check by entering the following command:

```
curl localhost:26657/status
```

You will get in return:

```
{
  "jsonrpc": "2.0",
  "id": -1,
  "result": {
    "node_info": {
      "protocol_version": {
        "p2p": "8",
        "block": "11",
        "app": "0"
      },
      "id": "**********",
      "listen_addr": "tcp://0.0.0.0:26656",
      "network": "fxcore",
      "version": "v0.34.9",
      "channels": "**********",
      "moniker": "moniker",
      "other": {
        "tx_index": "on",
        "rpc_address": "tcp://0.0.0.0:26657"
      }
    },
    "sync_info": {
      "latest_block_hash": "**********",
      "latest_app_hash": "************",
      "latest_block_height": "**********",
      "latest_block_time": "2021-11-11T21:17:32.56232646Z",
      "earliest_block_hash": "**********",
      "earliest_app_hash": "**********",
      "earliest_block_height": "1",
      "earliest_block_time": "2021-07-12T07:42:23.292429Z",
      "catching_up": false
    },
    "validator_info": {
      "address": "**********",
      "pub_key": {
        "type": "tendermint/PubKeyEd25519",
        "value": "**********"
      },
      "voting_power": "1"
    }
  }
```

If you have `"catching_up": true`, your node is still synchronizing. Wait until it's `false`

If you have `"voting_power": "0"`, your node is inactive or jailed.

2.2.2 My node does not start

How do you know that ? Do you have an error message ?

Contact us on the FunctionX forum (or me in private message) and we'll do our best to help you.

Don't forget to send us your node name, its public address, the error message and your logs.

2.2.3 I need to stop my node

Type the following command on the Linux terminal hosting your node:

```
sudo systemctl stop fxcored
```

2.2.4 I need to restart my node

Type the following command on the Linux terminal hosting your node:

```
sudo systemctl restart fxcored
```

2.2.5 My node is inactive

Is your 'fxcored' process running ? Check it out first using information in §2.2.1.

If everything is alright, your node is probably “jailed”. A node is jailed if it fails to validate blocks for a long period of time, or if a misbehavior of your node was detected on the blockchain. Follow information in §2.2.6.

2.2.6 My node is jailed

Type the following command (replace “my-node-name” with your node name that you noted previously) on the Linux terminal hosting your node:

```
fxcored tx slashing unjail --from="my-node-name" --fees
1200000000000000000FX
```

If it’s not too bad, your node should be unjailed quickly (within a minute). Otherwise, contact the FunctionX team on the forum.

2.2.7 I want to access FxCore logs

You can access the ‘fxcored’ logs in multiple ways:

Entire log : type the following command on the Linux terminal hosting your node

```
journalctl -t fxcored
```

Entire log reversed : type the following command

```
journalctl -t fxcored -r
```

Latest and continuous : type the following command

```
journalctl -t fxcored -f
```

2.2.8 This tutorial contains glitches

Well : nobody’s perfect !

Contact me on the FunctionX forum (https://forum.functionx.io/u/mdmdmd83) and I’ll handle that as soon as possible !

3 Donations

Any donation will be greatly appreciated :

- **FxCore :** fx1y8yygve3q5sdsh5rw56q86m2udnxlap3ned0v8
- **PundiX :** px1y8yygve3q5sdsh5rw56q86m2udnxlap3gcah2z
- **Ethereum :** 0xCD6BF294Ee776615fAa0C4c8885902802995f8A5

4 Sources

- FunctionX Mainnet explorer : https://explorer.functionx.io/
- FunctionX Testnet explorer : https://aabbcc-explorer.functionx.io/
- FunctionX Testnet faucet : https://aabbcc-faucet.functionx.io/ (to get free testnet FX)
- FunctionX forum : https://forum.functionx.io/
- FunctionX documentation : https://functionx.gitbook.io/home/
- FunctionX website: https://functionx.io/

5 Thanks

Special thanks to Richard Ng and ClaudioxBarros for their help and support.

And of course, special greetings to Zac Cheah, David Ben Kay and the rest of the team for their continuous effort to bring this project to life !

6 Contact

Although I'm not part of the team, feel free to contact me for any question :

- Web: https://forum.functionx.io/u/mdmdmd83
- Twitter: @FunctionxNerd