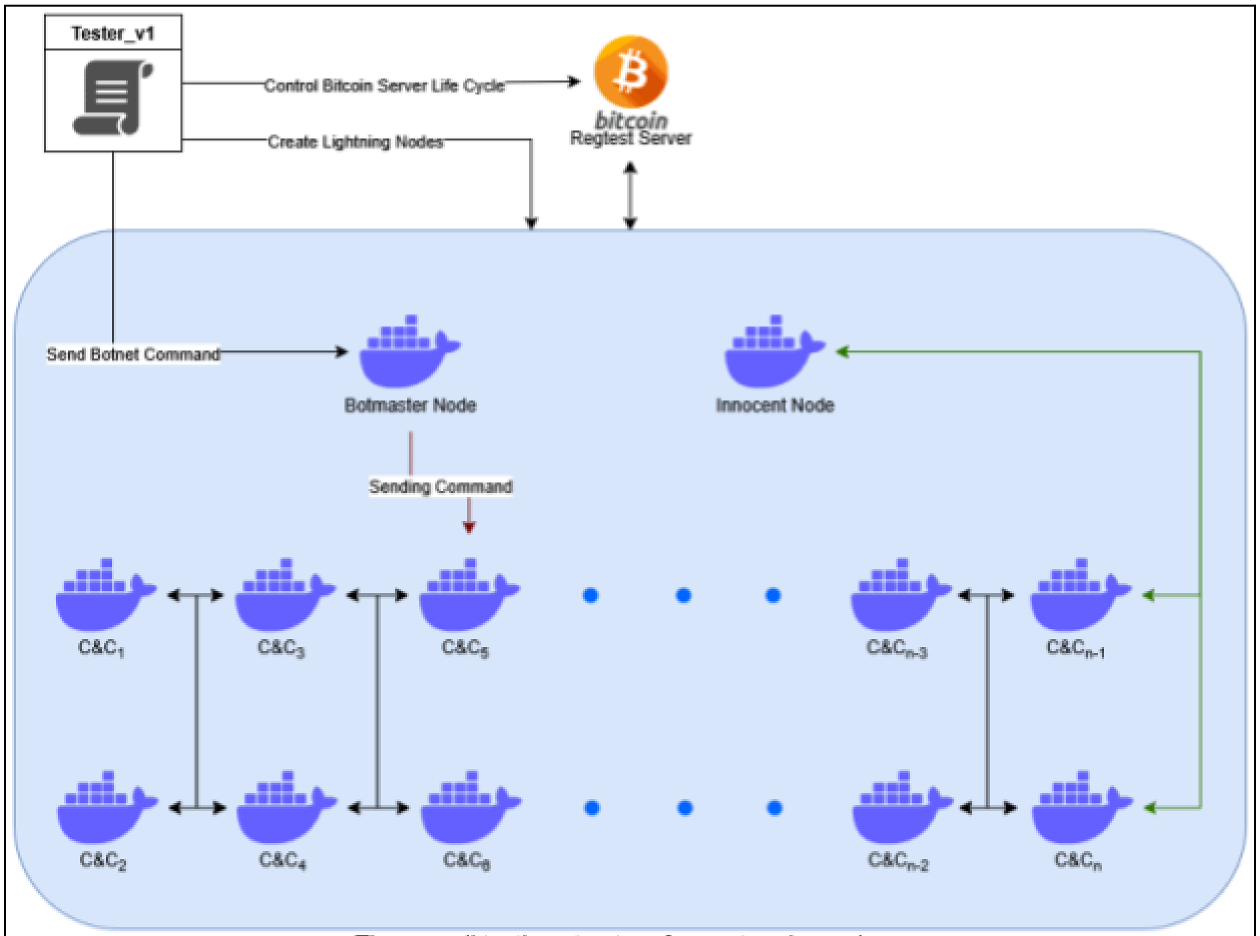
Architecture

The botnet has 5 essential components. We have the main tester script that runs on the host machine, the Innocent Node, Botmaster Node and the CC Server Nodes, all of which run on individual docker containers running the “elementsproject/lightningd:latest” image. Finally we have bitcoin core running a regtest server on the host machine to simulate the bitcoin network.

Each test automatically restarts the bitcoin server with a fresh wallet. In an effort to minimize the impact of previous tests on the next, all nodes and their associated resources are taken down after every test.

This setup was done with a linux ubuntu distribution.



The overall testing structure from a top down view.

Pre-requisites

This guide assumes that the user is starting from a fresh install of Ubuntu.

Update Ubuntu

Be sure your system is up to date.

```
sudo apt update
Sudo apt upgrade
```

Python

This project uses bash and python scripts. We will specifically create a virtual environment for this project.

Install venv for python so we can create the virtual environments.

```
sudo apt install python3-venv
```

Docker

All lightning nodes will be individual docker containers. Reference the following guide, the pertinent instructions have been provided.

- https://docs.docker.com/engine/install/ubuntu/#install-using-the-repository

Install the Apt resources.

```
# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o
/etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}")
stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

Actual installation of Docker.

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin
docker-compose-plugin
```

Check the status of Docker, it should be running at this point.

```
sudo systemctl status docker
```

Git

The repository containing the project will need to be cloned.

Install git using apt.

```
sudo apt install git
```

Set-up

Create directory for LNBot and the Bitcoin-Core files

Navigate to the `Documents` directory in your home directory.

```
cd ~/Documents
```

Create a directory for LNBot and Bitcoin core to live in. In our testing environment we named it "`LNBot_research_project`".

```
mkdir LNBot_research_project
cd LNBot_research_project
```

Note: If you name this directory something else, remember to modify the "`config.env`" file in the cloned LNBot repo.

Install Bitcoin-core

Download the bitcoin core tar file from https://bitcoin.org/en/download and move this file into the `LNBot_research_project` directory.

Extract the bitcoin core tar file here. You should be able to tab to complete the name.

```
tar -xvzf bitcoin-*
```

Rename the folder to bitcoin.

```
mv bitcoin-* bitcoin
```

Do not run bitcoin-core just yet.

Create bitcoin and lightning configs

Create bitcoin config

We will create a config file for the bitcoin core server to use. This way the bitcoin server will start as a regtest with the same server everytime. Change your directory to */.bitcoin*, if the directory doesn't exist create it.

```
mkdir ~/.bitcoin #if the directory doesn’t exist
cd ~/.bitcoin
nano bitcoin.conf
```

We will set a few rules for the bitcoin server. This will define the network settings, how the server listens for incoming RPC connections and other settings to ensure smooth interactions with tools like the Lightning Network.

Copy and paste this into the bitcoin.conf file.

```
#Global Settings

regtest=1 #run as regtest
server=1 #enable rpc control
daemon=1 #run bitcoincore in the background
txindex=1 #index transactions for faster lookups
# Increase the system resources available
rpcworkqueue=512
rpcthreads=64
```

```
prune=n #keeps full blockchain for lightning compatibility

# REMEMBER to use the same bitcoin credentials
rpcuser=YourRpcUsername
rpcpassword=YourRpcPassword

[regtest]

rpcport=8332
rpcallowip=127.0.0.1
rpcallowip=10.0.0.0/8
rpcallowip=172.0.0.0/8
rpcallowip=192.0.0.0/8
zmqpubrawblock=tcp://0.0.0.0:28332
zmqpubrawblock=tpc://0.0.0.0:28333
zmqpubhashblock=tcp://0.0.0.0:28334
whitelist=127.0.0.1

fallbackfee=0.00001
```

Remember the **username** and **password** you used here since we have to use the same credentials for the configs being passed into the lightning containers.

Create lightning config

We will create a config file for the lightning daemons to use. We will set the rpcusername and rpcpassword that these C&C nodes will be using to connect to the bitcoin regtest network.

Change your directory to */.lightning*, if the directory doesn't exist create it.

```
mkdir ~/.lightning #if the directory doesn't exist
cd ~/.lightning
```

Here we create the configuration file for lighting.

```
nano lightning.conf
```

Copy and paste the following:

```
network=regtest #we are on a regtest network

# REMEMBER to use the same bitcoin credentials
bitcoin-rpcuser=YourRpcUsername
bitcoin-rpcpassword=YourRpcPassword
bitcoin-rpcconnect=127.0.0.1
bitcoin-rpcport=8332

log-level=debug
```

The **username** and **password** here should match the credentials placed into bitcoin.conf.

At this point the configuration for the testing setup on the host machine is complete. The last requirement is to modify the testing files and double check that the path files are correct relative to your directory paths.

Test Bitcoin environment

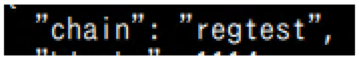
We will start the bitcoin server and verify that it is starting as a regtest environment.

Change directories to where the bitcoin-core files are located. This assumes you followed the exact same file structure as we did.

```
cd ~/Documents/LNBot_research_project/bitcoin/bin
./bitcoind
```

This should start the bitcoin server. Now we verify that this is running in a regtest environment. Run getblockchaininfo to get information on the currently running server.

```
./bitcoin-cli getblockchaininfo
```

We're looking for this specifically. 

```
"chain": "regtest",
```

That "chain" is "regtest". Once this is verified we can stop the server and we can now set up LNBot.

```
./bitcoin-cli stop
```

Note: You can add bitcoind and bitcoin-cli to your path environment if you want to start and stop bitcoin-core without having to run them in this directory specifically.

Setting up LNBot

Using git, we will download the repo containing LNBot. Keep in mind that this is a self contained testing suite and will not communicate outside of the host machine, hence why we did not need to open up any ports in the previous steps. We will then modify the variables inside the “config.env” file to match your unique system.

It is important to note that the script uses hard paths to find the necessary files to run. This is because we use sudo to run the tester files (docker requires sudo unless you are part of the docker group) and so the tester file will look in “root”s home directory for those files unless a correct path is used instead.

Clone the repo using git

As of 9/30/25 this is a private repo, you will need a private access token (PAT) from Thomas Bakaysa Jr if you wish to clone the repo. First we move into the LNBot_research_project directory and then clone the git repo into this directory.

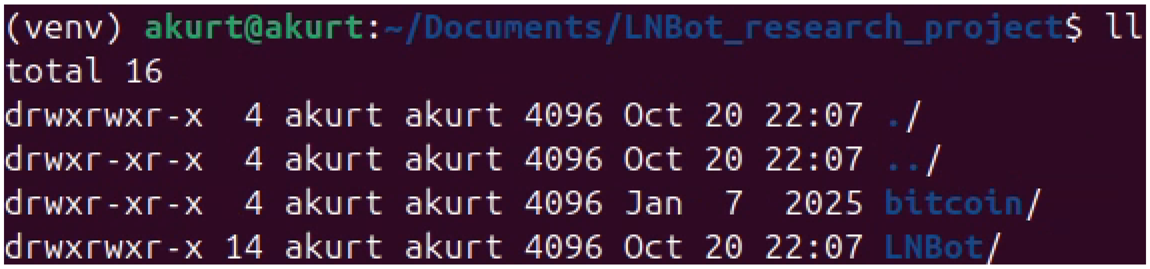
```
Cd ~/Documents/LNBot_research_project
git clone https://github.com/ThomasBakaysaJr/LNBot_Research.git
```

It will then ask for a username and password. The user name is ThomasBakaysaJr and the password will be the provided PAT. Unzip the file containing the PAT and open the text file, copy and paste the entire line as the password.

Rename the cloned repo to LNBot

```
mv [repo] LNBot
```

The final directory structure inside the LNBot_research_project should appear as follows:

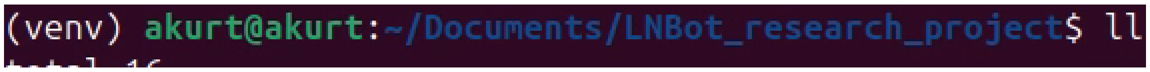


```
(venv) akurt@akurt:~/Documents/LNBot_research_project$ ll
total 16
drwxrwxr-x  4 akurt akurt 4096 Oct 20 22:07 ./
drwxr-xr-x  4 akurt akurt 4096 Oct 20 22:07 ../
drwxr-xr-x  4 akurt akurt 4096 Jan  7  2025 bitcoin/
drwxrwxr-x 14 akurt akurt 4096 Oct 20 22:07 LNBot/
```

Modify config.env

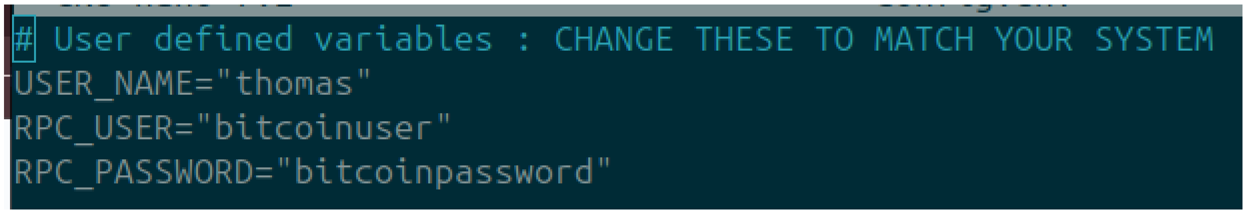
With LNBot now on your system, modify the config.env file located in the root folder of LNBot. The first three variables are the most important, ensure these match the system you're currently on.

USER_NAME : Needs to match the current user account.
In this example USER_NAME should be "akurt".



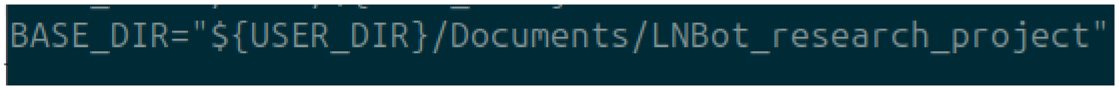
```
(venv) akurt@akurt:~/Documents/LNBot_research_project$ ll
```

RPC_USER: Should match what's in bitcoin.conf and lightning.conf
RPC_PASSWORD: Should match what's in bitcoin.conf and lightning.conf



```
# User defined variables : CHANGE THESE TO MATCH YOUR SYSTEM
USER_NAME="thomas"
RPC_USER="bitcoinuser"
RPC_PASSWORD="bitcoinpassword"
```

Change the "BASE_DIR" to match your directory structure if you did not use the same naming conventions.
This should be the path to where the bitcoin and LNBot directories are living.



```
BASE_DIR="${USER_DIR}/Documents/LNBot_research_project"
```

Downloading python dependencies

From here we're going to create a virtual environment for python, activate it and then install the required libraries.

Inside the LNBot directory:

```
python3 -m venv venv
source venv/bin/activate
pip install -r requirements.txt
```

Running tester_v1

The python script “tester_v1” is the main script for testing the Lightning Botnet. It is responsible for creating the containers, managing memory, sending the initial botnet commands for propagation and then tearing everything down for subsequent tests.

Because this script manages memory and deals with running docker files, it must be run as `sudo`, however doing so reset the path for python, which means that we lose the dependencies we just installed. To use the correct interpreter, we need to run `sudo` with an absolute path to the correct python interpreter. Run tester_v1.

```
sudo venv/bin/python tester_v1.py --small
```

This will start a small gamut of tests to see if everything is set up correctly. Progress can be monitored in the logs in the log directory, with the status of each node stored in the status directory.

Data collected will be in the “data/” directory, remember to save your data before each run since the script will overwrite any data that is stored there.

Commands and Script

An overview of the useful scripts contained in this setup. This section will be written as:

Script

```
How to run script
```
Description of what this script does.

tester_v1

```
sudo venv/bin/python tester_v1
```
Use “-h” to bring up the help screen to see the possible commands.
Can choose between “--full”, “--small” and “--test”.
“--full”
- Run all tests from start to finish. Change the parameters to change how to the tests runs

“--small”
- Run a small gamut of tests to make sure that the scripts are working properly

"--test"

- Run a specific series of tests. Choose between:
    - 1: Changing number of cc nodes
    - 2: Changing number of active nodes
    - 3: Changing number of cc nodes the botmaster will connect to
    - 4: Changing number of locations the botmaster will connect to (fixed to top, middle and bottom)

Optional parameters that can be changed for the tests. Does not affect "--small" tests.
Behavior:
Optional parameters will affect all tests. All tests will start iterations at the provided values and will hold at provided values for all other tests.
Example:
Having "--num_cc 30" will start num_cc tests at 30 and will create 30 C&C servers when testing other parameters.

"--num_cc #"

- Number of CC nodes for the tests

"--active_nodes #"

- Number of active nodes

"--bm_cc #"

- Number of channels the botmaster will create

"--bm_pos #"

- Position where the botmaster will connect to in the network

"--max_msg #"

- Number of messages to send per test

kill_nodes.sh

```
sudo ./kill_nodes
```

Stops and removes all docker nodes created during the test.
Clears out shared memory.
Removes the persistent docker directories so that no files interfere with further tests.
Does not remove logs in the NodeManagerComms/logs directory.

cleanup_lightning_nodes.sh

```
sudo ./cleanup_lightning_nodes.sh
```

Kill nodes except it also clears out the logs.
This is the script that the tester_v1 script calls after recording each test.

restart_bitcoin.sh

```
sudo ./restart_bitcoin.sh
```

Stops bitcoin-core.
Deletes regtest data so we start fresh.
Creates a new wallet for the tests, since by default bitcoind does not create a wallet.
Starts a mineBlocks bitcoin miner in the background.

Note: Does not kill any bitcoin miner that may be still running. That is done in the tester_v1 script.

Common Problems and Fixes

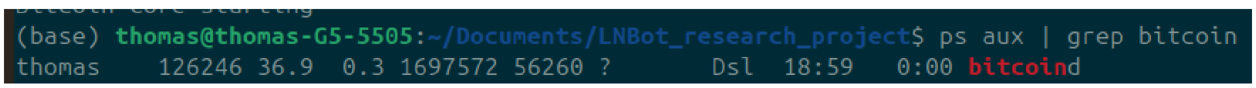
Some common problems that can pop up from time to time.

Bitcoin error

If you run into bitcoin errors as the testing starts, usually with a description of loading wallet or some such. This is usually because bitcoin core was already running and the script couldn't shut it down properly or the device was shutdown while bitcoin-core was still running.

You will need to find the pid of bitcoin-core and kill it, sometimes forcefully if it will not exit out with a normal kill command.

So find the bitcoin-core server and then kill it.



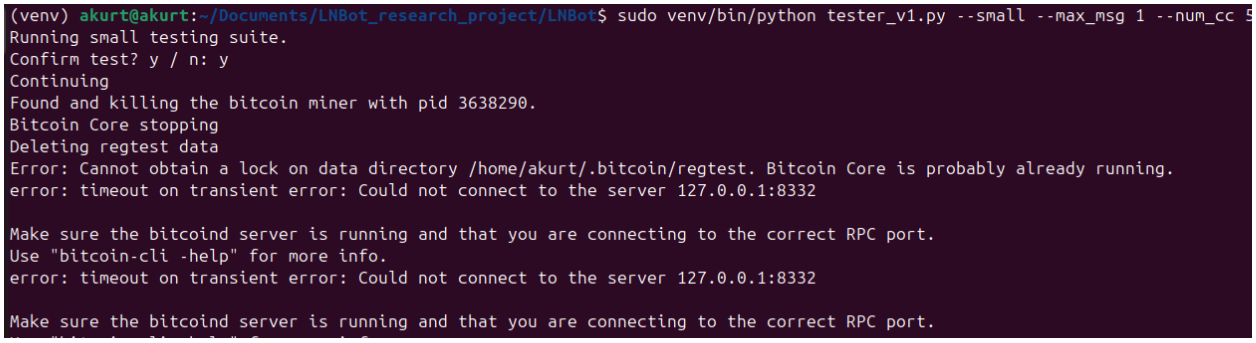
```
(base) thomas@thomas-G5-5505:~/Documents/LNBot_research_project$ ps aux | grep bitcoin
thomas    126246 36.9  0.3 1697572 56260 ?       Dsl  18:59   0:00 bitcoind
```

```
kill 126246
```

Note: I will be adding more to this document as I run into them, this is the biggest problem that I persistently run into.

Bitcoin lock error

If you start the tester and it states that it can't get a lock on the regtest folder, that means bitcoin-core was not shutdown automatically by the scripts. Crtl+c to exit the tester and retry, it usually clears up immediately.


```
(venv) akurt@akurt:~/Documents/LNBot_research_project/LNBot$ sudo venv/bin/python tester_v1.py --small --max_msg 1 --num_cc 5
Running small testing suite.
Confirm test? y / n: y
Continuing
Found and killing the bitcoin miner with pid 3638290.
Bitcoin Core stopping
Deleting regtest data
Error: Cannot obtain a lock on data directory /home/akurt/.bitcoin/regtest. Bitcoin Core is probably already running.
error: timeout on transient error: Could not connect to the server 127.0.0.1:8332

Make sure the bitcoind server is running and that you are connecting to the correct RPC port.
Use "bitcoin-cli -help" for more info.
error: timeout on transient error: Could not connect to the server 127.0.0.1:8332

Make sure the bitcoind server is running and that you are connecting to the correct RPC port.
```

Last updated: Oct 20, 2025