Bitcoin Core Setup Guide

Step 1: Install Dependencies

Ubuntu:

```
sudo apt update
sudo apt install build-essential libtool autotools-dev automake
pkg-config bsdmainutils python3
sudo apt install libevent-dev libboost-system-dev
libboost-filesystem-dev libboost-test-dev libboost-thread-dev
sudo apt install libminiupnpc-dev libzmq3-dev libprotobuf-dev
protobuf-compiler libqrencode-dev
sudo apt install libdb5.3-dev libdb5.3++-dev
sudo apt install libqt5gui5 libqt5core5a libqt5dbus5 qttools5-dev
qttools5-dev-tools libprotobuf-dev protobuf-compiler libqrencode-dev
```

- **Explanation**:
  - `build-essential`, `libtool`, `autotools-dev`, `automake`: Development tools for compiling Bitcoin.
  - `pkg-config`, `bsdmainutils`: System utilities required by Bitcoin.
  - `libevent-dev`: Event notification library.
  - `libboost-*`: Boost libraries for various functionalities in Bitcoin.
  - `libminiupnpc-dev`: UPnP support.
  - `libzmq3-dev`: ZeroMQ support for network communication.
  - `libdb5.3-dev`, `libdb5.3++-dev`: Berkeley DB libraries for wallet storage.
  - `libqt5*`, `qttools5*`: Dependencies for building the Qt GUI.

macOS:

```
xcode-select --install
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
brew install autoconf automake libtool boost openssl@1.1 berkeley-db@5
libevent zeromq protobuf miniupnpc qrencode pkg-config
brew install qt@5
```

- **Explanation**:
  - Xcode command-line tools: Required for compiling software.
  - Homebrew: macOS package manager.
  - `autoconf`, `automake`, `libtool`: Used to set up build systems.
  - `boost`, `openssl@1.1`, `berkeley-db@5`, `libevent`, `zeromq`, `protobuf`, `miniupnpc`: Dependencies similar to Ubuntu.
  - `qt@5`: Required for building Bitcoin's GUI.

Step 2: Clone and Compile Bitcoin

Clone Bitcoin Repository:

```
git clone https://github.com/bitcoin/bitcoin.git
cd bitcoin
git checkout v27.x
```

Compile Bitcoin:

```
./autogen.sh
./configure --with-incompatible-bdb --prefix=/usr/local/bitcoin
make -j$(nproc)   # On Linux
make -j$(sysctl -n hw.logicalcpu)  # On macOS
sudo make install
```

- **Explanation**:
  - `./autogen.sh`: Generates configuration files for the build system.
  - `./configure`: Configures the build environment. `--with-incompatible-bdb` allows the use of Berkeley DB 5.3+ for wallet storage.
  - `make -j`: Compiles the code using parallel processing. `$(nproc)` or `$(sysctl -n hw.logicalcpu)`determines the number of CPU cores to use.

Step 3: Configure Bitcoin

Create .bitcoin/bitcoin.conf:

```
mkdir -p ~/.bitcoin
nano ~/.bitcoin/bitcoin.conf
```

Add the following configuration:

```
signet=1
rpcuser=myrpcuser
rpcpassword=myrpcpassword
rpcallowip=127.0.0.1
rpcport=8332
server=1
daemon=1
txindex=1
```

- **Explanation**:
  - `signet=1`: Enables the use of the Signet test network.
  - `rpcuser`, `rpcpassword`: Credentials for RPC access.
  - `rpcallowip=127.0.0.1`: Restricts RPC access to localhost for security.
  - `rpcport=8332`: Sets the port for RPC connections.
  - `server=1`: Enables running the node in server mode to accept RPC commands.
  - `daemon=1`: Runs Bitcoin Core as a background daemon.
  - `txindex=1`: Maintains a full transaction index, useful for certain queries.

Step 4: Running Bitcoin

Run the Bitcoin Daemon:

```
bitcoind -conf=~/.bitcoin/bitcoin.conf -datadir=~/.bitcoin
```

Run the Bitcoin GUI:

```
bitcoin-qt -conf=~/.bitcoin/bitcoin.conf -datadir=~/.bitcoin
```

- **Explanation**:
    - `bitcoind`: Runs the Bitcoin daemon in the background.
    - `bitcoin-qt`: Opens the Bitcoin GUI for interaction.

To Stop Bitcoin Daemon:

```
bitcoin-cli stop
```

- **Explanation**:
    - `bitcoin-cli stop`: Safely stops the running `bitcoind` daemon.

Step 5: Query Bitcoin via RPC

Test with bitcoin-cli:

```
bitcoin-cli -conf=~/.bitcoin/bitcoin.conf -datadir=~/.bitcoin
getblockchaininfo
```

- **Explanation**:
    - `getblockchaininfo`: Retrieves information about the current state of the blockchain.

Test with a Bash Script:

```
curl --user myrpcuser:myrpcpassword --data-binary
'{"jsonrpc":"1.0","id":"curltest","method":"getblockchaininfo","params
":[]}' -H 'content-type: text/plain;' http://127.0.0.1:8332/
```

- **Explanation**:
    - This script queries the `getblockchaininfo` method using RPC.

Step 6: Enabling RPC for Scripting

Ensure your `bitcoin.conf` has the following settings:

```
rpcuser=myrpcuser
rpcpassword=myrpcpassword
rpcallowip=127.0.0.1
rpcport=8332
server=1
daemon=1
```

- **Explanation**:
  - The `rpcuser` and `rpcpassword` settings allow a script or external program to query your Bitcoin node over RPC.
  - `rpcallowip` restricts access to the local machine, while `rpcport` defines the port for the RPC server.
  - Ensure that the `server` and `daemon` settings are enabled for proper server functionality and background execution.