

Install guide

Deploy your stack - install guide

This is the software you run yourself, on your own server. Nothing here phones home; getcryptocoin operates none of it. When it's running, **you are the operator** of your chain. Read OPERATOR-LICENCE.md before you start.

Everything ships configured for a testnet - the coins have **no monetary value**. Going further than a testnet is your decision and your responsibility.

What you need

Requirement	Why
A VPS (Ubuntu 22.04 / 24.04), 2 vCPU, 4 GB RAM, 40 GB disk	The chain, explorer and pool are long-running services - they need a server that's always on, with root/Docker.
Root access on first login	To run <code>bootstrap.sh</code> , which creates a safe user account and installs Docker.
Docker Engine + Compose plugin	Installed for you by <code>bootstrap.sh</code> .
(Optional) a domain + DNS	For a public explorer/site with HTTPS.

Shared hosting will not work - even with terminal access. A blockchain node, an electrs indexer, a mining pool and an explorer are background daemons that listen on their own TCP ports. cPanel/FTP-style shared hosting cannot run them. You need a VPS (or a dedicated/home server) where you control Docker and the firewall. This is a property of the software, not a limitation of this package.

Step 0 - first-time server setup (run once as root)

Your VPS gives you root access by default. Running the stack as root is unsafe - if anything goes wrong, it can affect the whole server. Before doing anything else, run `bootstrap.sh` as root. It creates a safe non-root user, installs Docker, and copies your files into the new user's home folder.

```
# While logged in as root, inside the unpacked bundle folder:
bash bootstrap.sh
```

It prompts for a username and password, explains what it is doing at each step, and tells you exactly what to type next. When it finishes:

1. Type `exit` to log out of root.
 2. Log back in as your new user.
 3. Continue from Step 1 below.
-

The full bundle, at a glance

```

getcryptocoin-stack-full/
|-- OPERATOR-LICENCE.md      <- read first - you become the operator
|-- START-HERE.md           <- plain-English quickstart (read if new to this)
|-- bootstrap.sh             <- run first as root - creates safe user + installs Docker
r
|-- setup.sh                 <- run second as your new user - starts chain + website
|-- check-license.sh         <- licence validation (paid bundles; see OPERATOR-LICENCE
$6)
|-- license.env              <- your order ID + licence key (paid bundles only)
|-- INSTALL.md               <- this file (technical detail)
|-- VERIFY.txt               <- sha256 of every package
|-- chain/                   <- the blockchain node (Litecoin/Scrypt fork) + electrs +
miner
|   |-- coin-params.env       <- your rebrand knobs (name, ticker, ports, magic...)
|   |-- docker-compose.yml
|   |-- bin/build-coin.sh
|-- explorer/                <- block explorer (web UI)
|-- pool/                    <- stratum mining pool (pool_server.py)
|-- faucet/                  <- optional test-coin faucet / forms handler
|-- website/                 <- your marketing + dashboard site (Astro source)
|-- wallet/                  <- Android wallet APK + how-to-verify

```

Each folder is also published as its own download, so you can take just the pieces you want (see "Individual packages" at the end).

Step 1 - make the chain yours (optional for testnet)

Open chain/coin-params.env. To run the reference GCC testnet as-is, change nothing. To make it your chain, edit the rebrand knobs - coin name, ticker, address prefix, network magic, ports, block reward, halving. These are consensus-critical: decide them before first start.

*If this bundle was branded for your coin (see BUILD-INFO.txt in the bundle root), coin-params.env is already filled in with your name, ticker, address prefix and a mined genesis - leave the consensus values as they are. Only change a *_PORT if it clashes with something else on your server. Changing the genesis/magic/prefix now would mean a different, incompatible chain.*

```

cd chain
nano coin-params.env      # COIN_NAME, COIN_TICKER, MAGIC, RPC_PORT, BLOCK_REWARD_COIN
...

```

Step 2 - build & start the chain

```
cd chain
docker compose build      # compiles the node from source (~10-15 min first time)
docker compose up -d      # starts: node-a, node-b, electrs, miner
docker compose logs -f miner  # watch blocks being mined
```

You now have a live testnet: two nodes, an electrs index, and a miner producing one block a minute.

Step 3 - start the explorer

```
cd ../explorer
docker compose up -d
```

The explorer reads from the chain's electrs. Browse to `http://YOUR_SERVER_IP:<explorer-port>` (the compose file documents the port). Put it behind nginx + Let's Encrypt for a public HTTPS explorer.

Step 4 - start the mining pool

```
cd ../pool
python3 pool_server.py \
  --rpc-url http://127.0.0.1:<RPC_PORT> \
  --rpc-auth <user>:<pass> \
  --port 3333 \
  --reward <your-miner-address> \
  --stats 3004 \
  --label "YOURCOIN pool"
```

Miners point their Scrypt miner at `stratum+tcp://YOUR_SERVER:3333`. Pool stats are served as JSON on `--stats` for your site to read. A systemd/ unit is included so the pool survives reboots.

Step 4b - registration portal + automatic payouts (the full pool)

`pool_server.py` is the bare stratum relay; the portal turns it into a real pool - miners register, get a dashboard (hashrate, workers, balance, payment history), set a payout address, and are paid automatically by the payout daemon. This is the experience your miners see.

1. Install the portal's Python deps (the stratum + payout daemon are pure stdlib; only the portal needs Flask):

```
cd /opt/getcryptocoin/pool
pip3 install --user -r requirements.txt
```

2. Configure the portal:

```
cd portal
cp .env.example .env
```

```
nano .env      # set PORTAL_SECRET (the file shows the one-liner) + STRATUM_HOST
```

3. Start the portal + payout daemon (templates are in pool/systemd/, set to /opt/getcryptocoin - edit the paths if you installed elsewhere):

```
mkdir -p ~/.config/systemd/user
cp /opt/getcryptocoin/pool/systemd/getcryptocoin-pool-portal.service.template ~/.confi
ig/systemd/user/getcryptocoin-pool-portal.service
cp /opt/getcryptocoin/pool/systemd/getcryptocoin-pool-payout.service.template ~/.confi
g/systemd/user/getcryptocoin-pool-payout.service
systemctl --user daemon-reload
systemctl --user enable --now getcryptocoin-pool-portal getcryptocoin-pool-payout
```

4. Create your operator account (sign in at /operator for pool config, miners, blocks, payments, the Coin Manager):

```
cd /opt/getcryptocoin/pool/portal
python3 init_operator.py you@yourdomain.example "a-strong-password"
```

5. Put it behind HTTPS. The portal listens on 127.0.0.1:3005 - point nginx/Caddy at it for pool.yourdomain.example with TLS, and open your stratum port (3333) in the firewall.

That's the full pool: stratum + portal + auto-payouts. Miners register at your portal, point a Script miner at stratum+tcp://YOUR_SERVER:3333 -u <mining-username>.<worker>, and are paid to their wallet once their balance passes the threshold.

Step 5 - the faucet (optional)

faucet/ is a small HTTP handler that hands out test coins so people can try the chain without mining. It has no value implications - testnet coins are free and worth nothing by design. Configure and run it per faucet/README.

Step 6 - build & serve your website

setup.sh handles this automatically when you run the full bundle - it builds the site inside a Docker container (no npm required on the host) and serves it on port 8080. If you want to build it manually instead:

```
cd ../website
npm install
npm run build      # static site -> ./dist
```

Serve dist/ with any static web server (nginx, Caddy, or npx serve dist). Edit the content first - it's your brand, your copy. Keep the **testnet / no monetary value** framing and avoid any profit/return/yield language; that framing is what keeps a software demo out of financial-promotion territory (see the licence, and take your own legal advice for your jurisdiction).

Step 7 - the wallet

wallet/ contains the Android APK. Sideload it, then point it at your chain's electrum endpoint (set when you built the chain). Verify the APK checksum against VERIFY.txt before installing.

Verify your download

Every package is checksummed. From the folder that contains the archives:

```
sha256sum -c VERIFY.txt
```

All lines should say OK. If any doesn't, do not deploy it - re-download.

Individual packages

If you don't want the whole stack, each component is downloadable on its own:

Package	What it is
`chain`	the blockchain node + electrs + miner
`pool`	the stratum mining pool
`explorer`	the block explorer web app
`wallet`	the Android wallet APK
`website`	the marketing + operator-dashboard site (Astro)
`faucet`	the optional test-coin faucet

Each carries its own copy of OPERATOR-LICENCE.md. The same rule applies to all of them: ****you run it, you operate it, you're responsible for what you do with it.****

getcryptocoin · software you run yourself. Testnet - no monetary value.