

1. Clone code

1.1 \$ git clone <https://github.com/swordlet/xdagPool.git>

1.2 \$ cd **xdagPool**

\$ git checkout master

\$ git checkout -b develop

\$ git branch --set-upstream-to=origin/develop develop

\$ git config user.email "you@example.com"

\$ git config user.name "Your Name"

\$ git config pull.rebase false

\$ git pull

We run the code in the develop branch

2. Configure redis

\$ sudo apt update

\$ sudo apt install redis-server

2.1 Modify redis password

\$ sudo nano /etc/redis/redis.conf

Find **#requirepass foobared** in the configuration file and modify it to:

requirepass your_new_password

Such as:

```

#
# +-----+ +-----+
# |      Master      | ---> |   Replica   |
# | (receive writes) |     | (exact copy) |
# +-----+ +-----+
#
# 1) Redis replication is asynchronous, but you can configure a master to
#    stop accepting writes if it appears to be not connected with at least
#    a given number of replicas.
# 2) Redis replicas are able to perform a partial resynchronization with the
#    master if the replication link is lost for a relatively small amount of
#    time. You may want to configure the replication backlog size (see the next
#    sections of this file) with a sensible value depending on your needs.
# 3) Replication is automatic and does not need user intervention. After a
#    network partition replicas automatically try to reconnect to masters
#    and resynchronize with them.
#
# replicaof <masterip> <masterport>
#
# If the master is password protected (using the "requirepass" configuration
# directive below) it is possible to tell the replica to authenticate before
# starting the replication synchronization process, otherwise the master will
# refuse the replica request.
requirepass 123456
# masterauth <master-password>
#
# However, this is not enough if you are using Redis ACLs (for Redis version
# 6.0 and later). Still, a read only replica exports by default all the administrative commands
# such as CONFIG, DEBUG, and so forth. To a limited extent you can improve
# security of read only replicas using 'rename-command' to shadow all the
# administrative / dangerous commands.
replica-read-only no
#
# Replication SYNC strategy: disk or socket.
#
# New replicas and reconnecting replicas that are not able to continue the

```

Then save and close the file.

2.2 Restart the Redis service for the changes to take effect:

```
$ sudo systemctl restart redis-server
```

3. Install the environment related to running code

3.1 \$ sudo apt install golang-go

3.2 \$ sudo apt install cmake make

4. Change configuration in config.json

4.1 Enter the **xdagPool** file

```
$ cp config.example.json ./config.json
```

Such as:

```
ubuntu@10-35-80-195:~/xdagPool$ ls
-rw-r--r-- config.example.json  config.json  go.mod  go.sum  kvstore  LICENSE  main.go  payouts  pool  random  README.md  screenshot.png  store.txt  stratum  util  ws  ws.txt  xdag  xdagpool
```

4.2 Modify the **config.json** configuration file

First click <https://anycript.com/> (or other AES encryption tool URL) to encrypt your wallet address and redis password.

Configure **addressEncrypted** parameter

Anycript Online Encryption Tool.

Anycript is a free tool for AES online encryption and decryption. This tool performs ECB and CBC encryption modes and supports the key length of 128/192/256 bits. Anycript provides additional JSON formatting for decrypted raw data (only if the data is in raw JSON Format). The data you enter on Anycript is safe and secure. We are not saving your data and not sending it to a server. Anycript is a client-side tool. All the processes are executed in your web browser.

AES Encryption

Encryption Text	Encrypted Text
<div>LW2PGwYk4eovUttAn64ApS6nQ29yKVBhU</div> <div>your pool wallet address</div>	<div>G6LLHMvWi6HiysT+PuCWxhuaTW0xbHIEocNf5ilWAY+e7KsjAGPVQe1</div> <div>PBgtxeFD</div> <div>addressEncrypted</div>

Secret Key

12345678*****

128 Bits

Select mode

CBC

ECB

IV (optional)

12345678*****

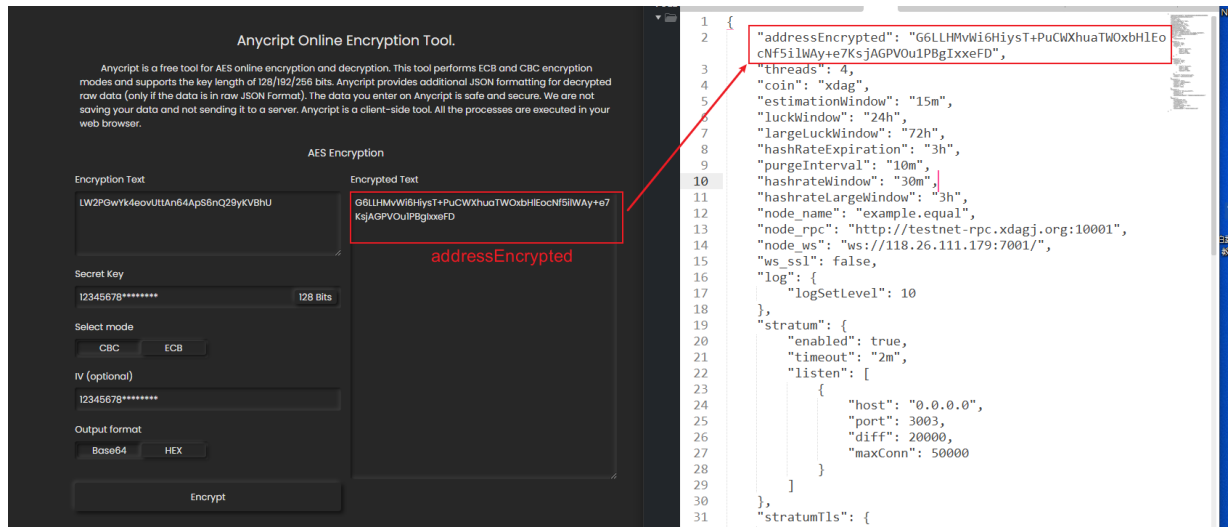
Output format

Base64

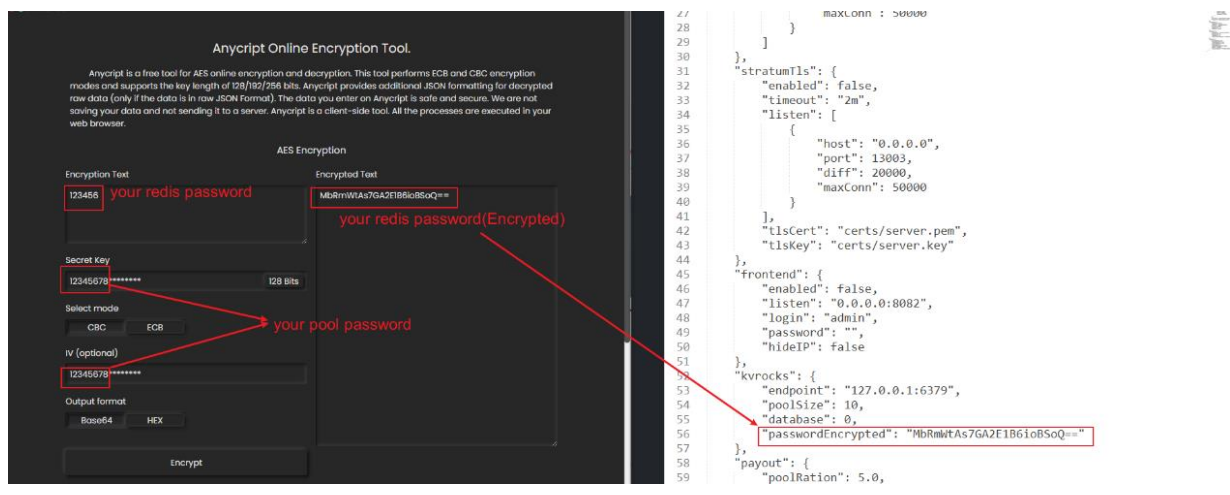
HEX

Encrypt

your pool password, if it is less than 16 characters, fill it with *. For example, the mining pool password here is 12345678



Configure **passwordEncrypted** (Encrypted redis password) parameter



5. Import wallet file

5.1 Open 0.6.6XDAGJ wallet

5.2 Copy the **xdagj_wallet** folder to the **xdagPool** folder



6. Compile project

6.1 \$ cd ./xdagPool/clib/randomx

6.2 \$ cmake .

```
ubuntu@10-35-80-195:~/xdagPool/clib/randomx$ cmake .
```

6.3 \$ make

```
ubuntu@10-35-80-195:~/xdagPool/clib/randomx$ make
```

6.4 \$ cd ./xdagPool

6.5 \$ go build

```
ubuntu@10-35-80-195:~/xdagPool$ go build
```

6.6 After the compilation is successful, you can see a compiled executable file

```
ubuntu@10-35-80-195:~/xdagPool$ ls
clib  config.example.json  config.json  go.mod  go.sum  kvstore  LICENSE  logs  main.go  payouts  pool  randomx  README.md  screenshot.png  store.txt  stratum  util  ws  ws.txt  www  xdagj_wallet  xdagj  xdagpool
```

7. Run your mining pool

7.1 \$ cd xdagPool

7.2 \$ sudo ./xdagpool

```
ubuntu@10-35-80-195:~/xdagPool$ sudo ./xdagpool
2024/01/04 02:58:35 Loading config: /home/ubuntu/xdagPool/config.json
2024/01/04 02:58:35 logSetLevel: 10
2024/01/04 02:58:35 infoFile: logs/info.log
2024/01/04 02:58:35 errorFile: logs/error.log
2024/01/04 02:58:35 shareFile: logs/share.log
2024/01/04 02:58:35 blockFile: logs/block.log
[I] 2024/01/04 02:58:35.791801 platform_linux.go:13: Rlimit Current: 1024
[I] 2024/01/04 02:58:35.791892 platform_linux.go:25: Setting Rlimit: 800000
[I] 2024/01/04 02:58:35.791901 platform_linux.go:31: Rlimit Final: 800000
Enter Security Password:

```

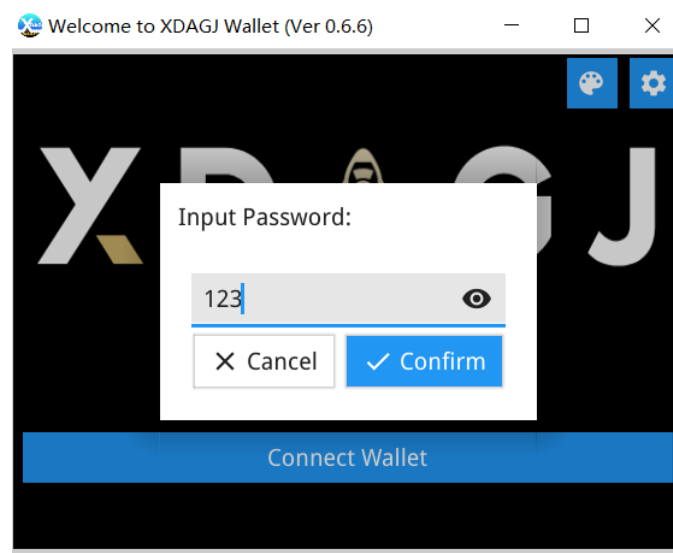
The pool password is the key used for your AES encryption. For example, **12345678******* was mentioned before as the key for encrypting the wallet address and redis password, then **12345678** is your pool password for unlocking the pool.

```
ubuntu@10-35-80-195:~/xdagPool$ sudo ./xdagpool
2024/01/04 02:58:35 Loading config: /home/ubuntu/xdagPool/config.json
2024/01/04 02:58:35 logSetLevel: 10
2024/01/04 02:58:35 infoFile: logs/info.log
2024/01/04 02:58:35 errorFile: logs/error.log
2024/01/04 02:58:35 shareFile: logs/share.log
2024/01/04 02:58:35 blockFile: logs/block.log
[I] 2024/01/04 02:58:35.791801 platform_linux.go:13: Rlimit Current: 1024
[I] 2024/01/04 02:58:35.791892 platform_linux.go:25: Setting Rlimit: 800000
[I] 2024/01/04 02:58:35.791901 platform_linux.go:31: Rlimit Final: 800000
Enter Security Password:

Enter Wallet Password:

```

The wallet password is the password used when logging into the wallet. For example:



Then **123** is your wallet password.

7.3 Running success result


```

ubuntu@10-35-39-255:~/xdagPool$ sudo ./xdagpool
2024/01/04 11:11:23 Loading config: /home/ubuntu/xdagPool/config.json
2024/01/04 11:11:23 logSetLevel: 10
2024/01/04 11:11:23 infoFile: logs/info.log
2024/01/04 11:11:23 errorFile: logs/error.log
2024/01/04 11:11:23 shareFile: logs/share.log
2024/01/04 11:11:23 blockFile: logs/block.log
[I] 2024/01/04 11:11:23.272781 platform_linux.go:13: Rlimit Current: 1048576
Enter Security Password:
Enter Wallet Password:
[I] 2024/01/04 11:11:33.379102 main.go:305: Initializing cryptography...
[I] 2024/01/04 11:11:33.379902 main.go:306: Reading wallet...
[I] 2024/01/04 11:11:33.742646 main.go:278: Backend check reply: PONG
[I] 2024/01/04 11:11:33.748336 client.go:94: Connected to server
[I] 2024/01/04 11:11:33.748365 rpc.go:36: Connected to server
[I] 2024/01/04 11:11:33.748387 main.go:54: Running with 4 threads
[I] 2024/01/04 11:11:33.748435 stratum.go:120: Set purge interval to 10m0s
[I] 2024/01/04 11:11:33.749672 stratum.go:188: Stratum listening on 0.0.0.0:3003
[I] 2024/01/04 11:12:00.024190 client.go:146: recv: {
  "msgType": 1,
  "msgContent": {
    "task": {
      "preHash": "1fac2c24318017e1265d92be8a903f30a8ab4fa34902b9fcec3c2f8dc8c4d631",
      "taskSeed": "ed7baf6eb9739e14e12a1f7b1a42fb4695f9179f9cd234e00000000000000000",
    },
    "taskTime": 26630280,
    "taskIndex": 11670
  }
}
[I] 2024/01/04 11:12:00.024421 rpc.go:47: Recieved text message {
  "msgType": 1,
  "msgContent": {
    "task": {
      "preHash": "1fac2c24318017e1265d92be8a903f30a8ab4fa34902b9fcec3c2f8dc8c4d631",
      "taskSeed": "ed7baf6eb9739e14e12a1f7b1a42fb4695f9179f9cd234e00000000000000000",
    },
    "taskTime": 26630280,
    "taskIndex": 11670
  }
}
[I] 2024/01/04 11:12:00.024565 blocks.go:65: New block to mine on example.equal at jobHash 1fac2c24318017e1265d92be8a903f30a8ab4fa34902b9fcec3c2f8dc8c4d631, timestamp: 26630280

```

Then you can use your mining machine to connect to this mining pool and start mining. The default configured pool port is **3003**. Please do not use a proxy, miners can directly connect to the mining pool for mining.

8. Other configurations

```

node_name : "example.equal"
"node_rpc": "http://118.26.111.179:10001", node rpc
"node_ws": "ws://118.26.111.179:7001/", node IP
"ws_ssl": false,

"payout": {
  "poolRation": 5.0,
  "rewardRation": 5.0,
  "directRation": 5.0,
  "threshold": 3,
  "paymentInterval": "5m",
  "mode": "equal",
  "paymentRemark": "http://testpool_01_equal.com"
}

```

mode: equal or solo

threshold: Indicates the payment threshold for issuing rewards to miners, here are 3 XDAG

paymentInterval: payment interval, here is 5 minutes, indicating that miner rewards are issued every five minutes

paymentRemark: URL of the mining pool

The two test node IPs currently running on the test network are **118.26.111.179** and **152.32.129.160**, and the port numbers are both **7001**. Currently, both nodes have opened the mining pool whitelist, **welcome to build your pool!**