Ark v0.7.x

- https://github.com/arkade-os/arkd/releases/tag/v0.7.0
- APIインターフェース
  - APIドキュメントの方はまだ更新されていない

2025/07/17

Arkを動かしてみようとしたときの最新バージョンが v0.7.0 だった。

arkインストール

https://github.com/arkade-os/arkd

ビルドはしないでreleaseページのものを使う。

- Nigiri
  - は、Raspberry Piに対応していなかった。。。
  - Setup
    - Electrs: http://localhost:3000
    - Esplora: http://localhost:5000

Bitcoin regtest

Blockstream/electrs

外部DBを使わない設定にもできる

- PostgreSQL($ apt install postgresql)
- Redis($ apt install redis)

wallet を unlock するとログが出始める

```
round f6a448f1-c071-4bed-a024-520629599513 aborted  error="not enough intents registered 0/1"
```

round が abort してまた新しい round が始まり、同じようなログが出る

エラーログではなくデバッグログなので、これは問題ないのか？

`wallet balance` で"main account"と"connectors account"がある。"main account"はwallet自身のようだが"connectors account"はよくわからない。

v0.7系から daemon側の機能がウォレットとdaemonで別プロセスになってそれぞれ常駐するようになった。

データは、bitcoind/electrs は `~/.bitcoin/regtest/` 内、ark 関係はカレントディレクトリの `./data/` 内に保存している。新たに作業するときは削除すると良い。
v0.6.x で作った環境をいくつか使い回しているので、そちらでのスクリプトなども参考にできるだろう。 `generate.sh` でブロックを自動生成させるのはときには便利だと思う。

Ark v0.6.x#75fd77

~/.bitcoin/bitcoin.conf

```
server=1
txindex=1
regtest=1

zmqpubrawblock=tcp://127.0.0.1:28332
zmqpubrawtx=tcp://127.0.0.1:28333

[regtest]
#rpcuser=testuser
#rpcpassword=testpass
rpcauth=testuser:90d538109436dcea4d3da67f65d6aa00$21214960fe9d1bbd9d5f40ab16212fe9a
a3d87a59e2cfef91232729c5de00657
fallbackfee=0.000001

# SPV
blockfilterindex=1
peerblockfilters=1
```

`bitcoin-cli estimatesmartfee` を行うAPIがあるので

start-new-bitcoind.sh

```
#!/bin/bash -e

# トランザクション数と生成回数の設定
TX_COUNT=30
BLOCKS=20

rm -rf $HOME/.bitcoin/regtest
bitcoind -daemon

while :
do
```

```
  bitcoin-cli getblockcount > /dev/null 2>&1
  if [ "$?" -eq 0 ]; then
    break
  fi
  echo -n "."
  sleep 1
done
echo

bitcoin-cli createwallet ""
./generate.sh 150

# 複数のfeeレベルを作成（1〜30 sat/vB 相当）
echo "Sending $TX_COUNT transactions with varying fees..."
for i in $(seq 1 $TX_COUNT); do
  DEST=$(bitcoin-cli getnewaddress)
  TXID=$(bitcoin-cli -named sendtoaddress address="$DEST" amount=0.0001
fee_rate="$i")
  echo "Sent $i $TXID"
done

# 少し時間を置く（mempoolに反映されるまでのため）
sleep 2

# ブロックを複数生成してTXを順次承認
echo "Mining $BLOCKS blocks..."
for i in $(seq 1 $BLOCKS); do
  echo "generate block $i"
  ./generate.sh
  sleep 0.2
done

# estimatesmartfee を試す
echo "Estimating fee for confirmation within 2 blocks..."
bitcoin-cli estimatesmartfee 2
```

start-electrs.sh

```
#!/bin/bash
```

```
while :
do
        dl=`bitcoin-cli getblockchaininfo | jq .initialblockdownload`
        if [ "$dl" == "false" ]; then
                break
        fi
        sleep 3
done

ELECTRS=electrs

DATADIR="$HOME/.bitcoin/regtest/electrs"
NETWORK="regtest"
RPCUSER="testuser"
RPCPASS="testpass"
ZMQ_ADDR="localhost:28332"
ELECTRUM_URL="localhost:50001"
REST_URL="0.0.0.0:3002"
ESPLORA_URL="*"

$ELECTRS --db-dir="$DATADIR" \
    --network="$NETWORK" \
    --cookie="${RPCUSER}:${RPCPASS}"  \
    --zmq-addr="$ZMQ_ADDR" \
    --electrum-rpc-addr="$ELECTRUM_URL" \
    --http-addr="$REST_URL" \
    --cors="$ESPLORA_URL"
```

Esploraは `bitcoin-cli` で確認するのが面倒なので使っているだけなので必須ではない。
直接 `npm` を使って起動しているだけなのでスクリプトファイルは `git clone` したディレクトリの中に置いて使う。

start-esplora.sh

```
#!/bin/bash

HOST=http://localhost
export CORS_ALLOW="*"
export STATIC_ROOT=$HOST:5000/
export API_URL=http:$HOST:3002/
npm run dev-server
```

`bitcoind` と `electrs` を起動させておく。

まずは `arkd-wallet` の起動。常駐する。ログはあまり出ない。

start-wallet.sh

```
#!/bin/bash

ARKW=../build/arkd-wallet-linux-amd64

#export ARKD_WALLET_BITCOIND_RPC_USER=testuser
#export ARKD_WALLET_BITCOIND_RPC_PASS=testpass
#export ARKD_WALLET_BITCOIND_RPC_HOST=localhost:18443
#export ARKD_WALLET_BITCOIND_ZMQ_BLOCK=tcp://localhost:28332
#export ARKD_WALLET_BITCOIND_ZMQ_TX=tcp://localhost:28333
export ARKD_WALLET_NEUTRINO_PEER=localhost:18444

# https://github.com/arkade-os/arkd/blob/master/docker-compose.regtest.yml
export ARKD_WALLET_ESPLORA_URL=http://localhost:3002
export ARKD_WALLET_DATADIR=./data/regtest
export ARKD_WALLET_NETWORK=regtest
export ARKD_WALLET_LOG_LEVEL=6
export ARKD_WALLET_PORT=6060

$ARKW
```

続けて `arkd` の起動。ログの記録をするのは主にこちら。

`arkd` はdaemonだけでなくdaemonへの制御を行うCLIコマンドでもある。

環境変数のいくつかは設定ファイルに書き込まれるようで、間違った引数で起動しようとしてエラー終了すると次回修正してもエラーになってしまう。データディレクトリを削除すること。

`ARKD_VTXO_TREE_EXPIRY` は512未満だと単位が秒ではなくブロック高になるので注意。

arkd.sh

```
#!/bin/bash

ARKD_FILE=arkd-linux-amd64
export ARKD=../build/$ARKD_FILE
export PASSWD=aaaaaaaa
```

```
export ARKD_PORT=7170
export ARKD_LOG_LEVEL=6
export ARKD_NO_MACAROONS=true
export ARKD_VTXO_TREE_EXPIRY=10 # ARKD_VTXO_TREE_EXPIRY < 512 は単位がブロック高になる
export ARKD_ROUND_INTERVAL=30
export ARKD_SCHEDULER_TYPE=block
#export ARKD_SCHEDULER_TYPE=gocron # 単位が秒の時はこちら(エラーになるので分かる)
export ARKD_UNILATERAL_EXIT_DELAY=512
export ARKD_BOARDING_EXIT_DELAY=1024
export ARKD_DATADIR=./data/regtest
export ARKD_WALLET_ADDR=localhost:6060
export ARKD_ESPLORA_URL=http://localhost:3002
export ARKD_VTXO_MIN_AMOUNT=1
export ARKD_LIVE_STORE_TYPE=inmemory
export ARKD_EVENT_DB_TYPE=badger
export ARKD_DB_TYPE=sqlite
export ARKD_UNLOCKER_TYPE="env"
export ARKD_UNLOCKER_PASSWORD="$PASSWD"
export ARKD_ALLOW_ZERO_FEES="true"

ARKD_URL=http://localhost:$ARKD_PORT

if [ "$1" == "init" ]; then
        $ARKD --url=$ARKD_URL wallet create --password $PASSWD
        $ARKD --url=$ARKD_URL wallet unlock --password $PASSWD
        sleep 10
        killall $ARKD_FILE
        echo "arkd is killed!"
elif [ "$1" == "unlock" ]; then
        $ARKD --url=$ARKD_URL wallet unlock --password $PASSWD
else
        $ARKD --url=$ARKD_URL $@
fi
```

ここまでで、 `bitcoind` 、 `electrs` 、 `arkd-wallet` 、 `arkd` が常駐している。
クライアント CLI の準備もしておく。複数使いたいこともあるので共通部分は `cli.sh` にまとめて違うところだけそれぞれスクリプトファイルを作る。

`cli.sh`

```
#!/bin/bash

ARK=../pkg/ark-cli/build/ark-linux-amd64
CLI="$ARK --network=regtest --datadir=$ARK_WALLET_DATADIR"

if [ "$1" == "init" ]; then
        $CLI init --password $PASSWD --server-url localhost:7170 --explorer http://
localhost:3002
elif [ "$1" == "pay" ]; then
        $CLI send --to $2 --amount $3 --password $PASSWD
elif [ "$1" == "settle" ] || [ "$1" == "recover" ]; then
        $CLI $1 --password $PASSWD
else
        $CLI $@
fi
```

1.sh

```
#!/bin/bash

export ARK_WALLET_DATADIR=./data/regtest/cli1
export PASSWD=bbbbbbbb

./cli.sh $@
```

初期化

```
$ ./arkd.sh init
```

ウォレットの作成とunlockを行い、**10秒後に arkd プロセスを殺す**。

~~10秒は特に根拠がなく、バックグラウンドで何かしているようなので少し待っているだけだ。~~

~~プロセスを殺す理由は、そうしないとなぜかVTXOの期限切れ処理がうまくいかないからである。理由は不明。~~

(2025/07/31：これをしてもだめになった。タイミング的なもののように思えてきた。)

入金

```
$ ./arkd.sh wallet address
```

入金用のBitcoinアドレス(boarding address)が出力されるので、そこに送金する。

クライアントが入金するときに同等のBTCが必要になるので多めに入れておいた方がよい。

1confすると残高になるようだがREADMEには最低2confと書かれていた。

balance

```
$ ./arkd.sh wallet balance
main account
   available: 9.99998586
   locked: 0.00000000
connectors account
   available: 0.00000000
   locked: 0.00000000
```

CLI1 の初期化など

クライアント側は常駐しない。

```
$ ./1.sh init
```

```
$ ./1.sh receive
```

これで boarding address と offchain address が出力されるので boarding address に送金する。

```
$ ./1.sh settle
```

boarding address への送金を offchain balance にするにはおそらく明示的にコマンドを打ち込む必要がある。

1conf は必要。 `arkd.sh` 側が confirm を確認している状態で `settle` コマンドを実行する。

settlementというページがあるが、実際はBoarding じゃないかと思っている。

APIには "settle" のようなコマンドはないので、いろいろ組み合わせて実現していると思われる。

batch outputの例

これは `ARKD_VTXO_TREE_EXPIRY=10` だから、OP_CSVで10ブロック以上経過しているのを確認して、シュノア署名の検証をして終わりというだけだ。シングル鍵だが全員の署名がいる」はずなのでMuSig2かもしれない。

```
script
   10
   OP_CHECKSEQUENCEVERIFY
   OP_DROP
   c73b6ff93a7ed7305105d8077a697b860af48012528935d7b2132dfa85491ffb
   OP_CHECKSIG
```

INPUT は CLI 1に送金した 1 BTC が使われている。

OUTPUT は 1つである。sweep tx の場合は connector output はいらないのか？ また、OUTPUT は P2WPKH である。つまりシングル鍵だ。

BuildBatchOutput()では MuSig。下の方にBuildConnectorOutput()もある。
BuildCommitmentTx()でそれらを使っているので、Batch Output と Connector Output を持つのは Commitment Transaction という考え方でよいと思う。