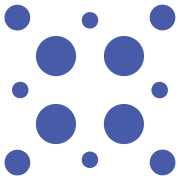


adax

CARDANO NODE DEPLOYMENT TOKEN MINTING AND BURNING (MAINNET)

SUMMARIZED BY
ADAX.PRO

NODE DEPLOYMENT:

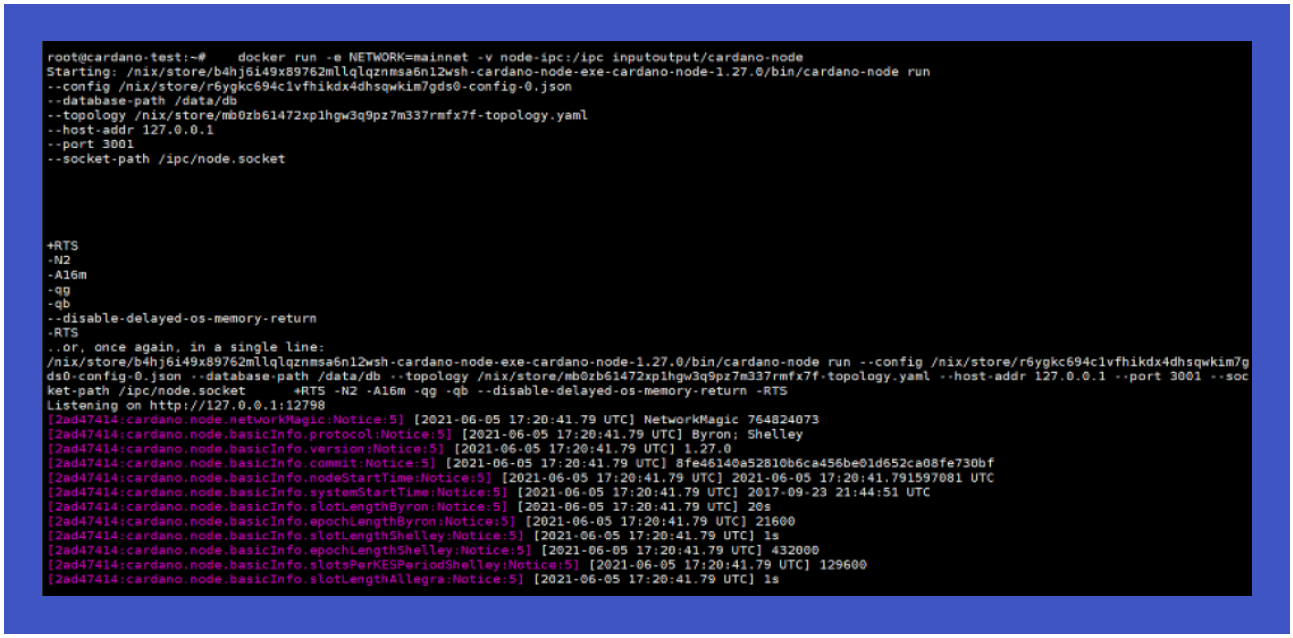
01 Get the latest Cardano Docker image.

```
docker pull inputoutput/cardano-node:latest
```

02 Run the node.

To run the node start a new tmux session paste the first command and start the node by pasting second command. Logs are also attached below.

```
1. tmux new -s cardano-node
2. docker run -e NETWORK=mainnet -v node-ipc:/ipc inputoutput/cardano-node
```



03 After the node starts syncing.

Once node starts syncing then detach the tmux session it will sync in background.

04 Check node sync status.

To check node sync status first check running docker container name by first command and then paste the container id in second command, then add node socket file path to the environment by third command. Paste fourth command to check the node sync status cli output attached below. Copy the block number and paste it to the cardanoscan it will show sync status. Explorer link also attached.

```
1. docker ps
2. docker exec -it {CONTAINER ID} bin/bash
3. export CARDANO_NODE_SOCKET_PATH=ipc/node.socket
4. cardano-cli query tip --mainnet
5. https://cardanoscan.io/(EXPLORER LINK)
```

```
root@cardano-test:~# docker ps
CONTAINER ID   IMAGE                    COMMAND                  CREATED          STATUS          PORTS     NAMES
2ad47414a1f5   inputoutput/cardano-node   "/nix/store/p435ajna…"   29 minutes ago   Up 29 minutes             compassionate_hawking
root@cardano-test:~# docker exec -it 2ad47 bin/bash
bash-4.4# export CARDANO_NODE_SOCKET_PATH=ipc/node.socket
bash-4.4# cardano-cli query tip --mainnet
{
    "epoch": null,
    "hash": "24effadda449cc04713e397e5963acb5b12b1a367abf96c2f7312b20eb264e8d",
    "slot": 538496,
    "block": 538430,
    "era": "Byron"
}
bash-4.4#
```

PRE-REQUISITES:

The “cardano-cli” command-line tool will be provided by the node. It supports many wallet functionalities that include address generation, transaction creation, signing, broadcasting, etc.

To mint, the token/ asset on the Cardano, issuer account, and script policy is required. Therefore, “cardano-cli” can be used for this.

01 Create a key pair:

```
cardano-cli address key-gen \
--verification-key-file policy.vkey \
--signing-key-file policy.skey
```

Note: “policy.vkey” and “policy.vkey” are the names of public key and private key.

Using the above-described command, two files for a key pair are created under the names “policy.skey” and “policy.vkey”. The contents in these files can be displayed using the “cat” command.

```
cat policy.skey
{
"type": "PaymentSigningKeyShelley_ed25519",
"description": "Payment Signing Key",
"cborHex":
"**********************************************************************"
}
```

```
cat policy.vkey
{
"type": "PaymentVerificationKeyShelley_ed25519",
"description": "Payment Verification Key",
"cborHex": "58201786da5ff6d53600cc03364e1301a93368186f21621159152107b459b2ff8451"
}
```

02 Derive the payment address from using the “cardano-cli address build” command.

```
cardano-cli address build \
--payment-verification-key-file policy.vkey \
--out-file account.addr \
--mainnet
```

Note the “pay.addr” is the output file where the payment address is stored.

Print the payment address

```
cat pay.addr
> addr1v9v8luaetjng8s6usgy364wqcgp73d37u6j4fqw5st2amcsq762ee
```

Also, query this address using the command “cardano-cli query utxo”:

```
cardano-cli query utxo \
--address addr1v9v8luaetjng8s6usgy364wqcgp73d37u6j4fqw5st2amcsq762ee \
--mainnet
```

```
bash-4.4# cardano-cli query utxo --address addr1v9v8luaetjng8s6usgy364wqcgp73d37u6j4fqw5st2amcsq762ee --mainnet
                           TxHash                                 TxIx        Amount
--------------------------------------------------------------------------------------
bash-4.4#
```

03 Retrieve the current Cardano protocol parameters using this command

```
cardano-cli query protocol-parameters \
--mainnet \
--out-file protocol.json
```

MINTING THE CARDANO ASSET:

Minting the assets on the Cardano network involves the following steps:

01 Create the policy

```
mkdir policy
cd policy
cardano-cli address key-gen \
--verification-key-file policy.vkey \
--signing-key-file policy.skey
touch policy/policy.script && echo "" > /policy.script
echo "{" >> policy.script
echo " \"keyHash\": \"$(./cardano-cli address key-hash
--payment-verification-key-file policy.vkey)\"," >> policy.script
echo " \"type\": \"sig\"" >> policy.script
echo "}" >> policy.script
```

Note: The directory name is “policy” and a fresh key pair is required for the policy. Therefore, “policy.vkey” is the public key and the “policy.skey” is the private key.

Display the policy.script:

```
cat policy.script
{
"keyHash": "e2c3e007410d4a0cb2a6a9c25ecd803ec170a2b5b022cc6fc5f2ebff",
"type": "sig"
}
```

02 Mint the policy id using the “cardano-cli transaction policyid” command.

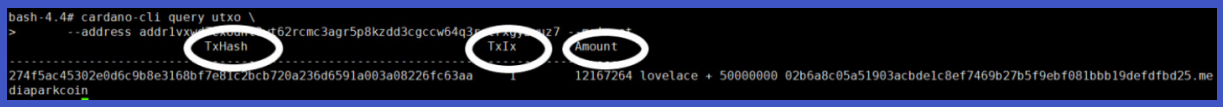
```
cardano-cli transaction policyid --script-file ./policy.script
```

This command will return a policyid copy this policyid.

03 Query UTXO

Querying UTXO is required to build raw transaction.

```
cardano-cli query utxo \
--address {Address} \
--mainnet
```



04 Building raw transaction(without fee):

“ADA” tokens are required to create transactions from an address. Lovelace is the smallest unit of ADA and one ADA is equal to one million Lovelace (1000000).

Cardano is UTXO based, now using this transaction hash id as a transaction input. . In the transaction one million tokens is set to mint.

```
cardano-cli transaction build-raw \
--mary-era \
--tx-in {TxHash}#{TxIx} \
--out-file transaction.raw \
--fee 0 \
--tx-out{Receiving Address}+{ADA amount}+"{Tokens amount to mint}
{Policyid}.{tokenname}" \
--mint "{Tokens amount to mint} {Policyid}.{tokenname}" \
--minting-script-file ./policy.script
```

Note: ADA has to be set in the lovelace. Whereas the asset amount is allowed without any decimal placing or further units.

Display the raw transaction, it is stored in “transaction.raw”:

```
cat transaction.raw
{
"type": "TxBodyMary",
"description": "",
"cborHex":
"84a40081825820bbafb10df4ccd8fe380cfd0cde40fdf891384af1e512a58fe65f7eea29a55068000
18182581d60133160c
7eef0beb94206e771ad5381df350413b49a2c349beda541cf821a3b9aca00a1581c38ead71bf59ef67
9593685f406554b7a
c26f3007802b6bd433f396b2a14d6d656469617061726b636f696e1a000f4240020009a1581c38ead7
1bf59ef679593685
f406554b7ac26f3007802b6bd433f396b2a14d6d656469617061726b636f696e1a000f42409f820058
1ce2c3e007410d4
a0cb2a6a9c25ecd803ec170a2b5b022cc6fc5f2ebfffff6f6"
}
```

05 Calculate the minimum fee:

While creating the raw transaction the fee value was set to “0” now using the “cardano-cli transaction calculate-min-fee” command, the transaction fee will be calculated as:

```
cardano-cli transaction calculate-min-fee \
--tx-body-file transaction.raw \
--tx-in-count 1 \
--tx-out-count 1 \
--witness-count 2 \
--mainnet \
--protocol-params-file protocol.json
>{feevalue in lovelace}
```

06 Build the transaction again (With fee):

Build the transaction again using the same parameters except “fee”. Total fee of {**fee**}Lovelace is set.

Moreover, utxo “{**TxHash**}” contains {**balance**} ADA, which is equal to “{**balance *10^6**} Lovelace”. According to the utxo model of Cardano, the total amount of ADA in the input should be equal to the total amount of ADA. The fee is required for every transaction to be processed so that it can be successful in the blockchain. Therefore, creator of the transaction should follow this:

```
Input amounts = Output amounts
Total received ADA = Fee + Amount changed

Now, compute this, {totallovelace} - {fee} = {remaining}.
```

So, build the raw transaction according to total fee, sending ADA and receiving back ADA:

```
cardano-cli transaction build-raw \
--mary-era \
--tx-in {TxHash}#{TxIx} \
--out-file transaction.raw \
--fee {fee} \
--tx-out{Receiving Address}+{ADA amount}+"{Tokens amount to mint}
{Policyid}.{tokenname}" \
--mint "{Tokens amount to mint} {Policyid}.{tokenname}" \
--minting-script-file ./policy.script
```

07 Sign the transaction:

Sign the transaction using the “cardano-cli transaction sign” command. For this, two secret keys are required: “policy.skey” and “policy/policy.skey”. First secret key belongs to the asset issuer. Second secret key belongs to the policy script which is placed under the directory “policy” and it also contains the same name as “policy.skey”.

```
cardano-cli transaction sign \
--signing-key-file policy.skey \
--signing-key-file policy/policy.skey \
--mainnet \
--tx-body-file transaction.raw \
--out-file transaction.signed
```

08 Submit the transaction:

Submit the transaction to the cardano network using the “cardano-cli transaction submit” command.

```
cardano-cli transaction submit \
--tx-file transaction.signed \
--mainnet

> Transaction successfully submitted
```

The message “Transaction successfully submitted” is returned by the “cardano-cli transaction submit” command. This means that the asset name “{**tokenname**}” has successfully minted with this supply “{**supply**}”.

Verify it by querying the utxo of the address used in the transaction build process.

```
cardano-cli query utxo \
--address {address} --mainnet

TxHash TxIx Amount
--------------------------------------------------------------------------------------
274f5ac45302e0d6c9b8e3168bf7e81c2bcb720a236d6591a003a08226fc63aa                    1 12167264 lovelace + 50000000
02b6a8c05a51903acbde1c8ef7469b27b5f9ebf081bbb19defdfbd25.mediaparkcoin
```

The last line is the utxo, which shows that an asset named mediaparkcoin with “1000000” is created. This changed amount is “999817867” and is returned to the same address.

We can check tokens by pasting policyid to explorer like.

```
https://cardanoscan.io/tokenPolicy/02b6a8c05a51903acbde1c8ef7469b27b5f9ebf081bbb19defdfbd25
```

MINTING THE CARDANO ASSET:

01 Building raw transaction:

```
cardano-cli transaction build-raw \
--mary-era \
--tx-in {TxHash}#{TxIx} \
--out-file transaction.raw \
--fee 0 \
--tx-out{Receiving Address}+{ADA amount}+"{Tokens amount to transfer}
{Policyid}.{tokenname}"
```

02 Calculate the minimum fee:

```
cardano-cli transaction calculate-min-fee \
--tx-body-file transaction.raw \
--tx-in-count 1 \
--tx-out-count 1 \
--witness-count 2 \
--mainnet \
--protocol-params-file protocol.json
>{feevalue in lovelace}
```

03 Build the transaction again:

```
cardano-cli transaction build-raw \
--mary-era \
--tx-in {TxHash}#{TxIx} \
--out-file transaction.raw \
--fee {fee} \
--tx-out{Receiving Address}+{ADA amount}+"{Tokens amount to transfer}
{Policyid}.{tokenname}"
```

04 Sign the transaction:

```
cardano-cli transaction sign \
--signing-key-file policy.skey \
--signing-key-file policy/policy.skey \
--mainnet \
--tx-body-file transaction.raw \
--out-file transaction.signed
```

05 Submit the transaction:

```
cardano-cli transaction submit \
--tx-file transaction.signed \
--mainnet

> Transaction successfully submitted
```

After submitting, query the recipient address on cardanoscan (it may take some time).

DESTROYING THE TOKEN:

The minted token e.g. "{**tokenname**}" over the Cardano blockchain can be destroyed. The token destroying process is the same as asset minting. Only the asset "--mint" argument needs to be set to a negative asset amount.

01 Building raw transaction:

Value to tokens should be negative in order to destroy tokens.

```
cardano-cli transaction build-raw \
--mary-era \
--tx-in {TxHash}#{TxIx} \
--out-file transaction.raw \
--fee 0 \
--tx-out{Receiving Address}+{ADA amount}+"{-Tokens amount to destroy}
{Policyid}.{tokenname}" \
--mint "{Tokens amount to mint} {Policyid}.{tokenname}" \
--minting-script-file ./policy.script
```

02 Calculate the minimum fee:

Fee calculation process is the same as used in asset minting.

```
cardano-cli transaction calculate-min-fee \
--tx-body-file transaction.raw \
--tx-in-count 1 \
--tx-out-count 1 \
--witness-count 2 \
--mainnet \
--protocol-params-file protocol.json
>{feevalue in lovelace}
```

03 Build the transaction again:

Fee is set to {**fee**}.

```
cardano-cli transaction build-raw \
--mary-era \
--tx-in {TxHash}#{TxIx} \
--out-file transaction.raw \
--fee 0 \
--tx-out{Receiving Address}+{ADA amount}+"{-Tokens amount to destroy}
{Policyid}.{tokenname}" \
--mint "{Tokens amount to mint} {Policyid}.{tokenname}" \
--minting-script-file ./policy.script
```

04 Sign the transaction:

Signing the transaction for asset destruction is the same process.

```
cardano-cli transaction sign \
--signing-key-file policy.skey \
--signing-key-file policy/policy.skey \
--mainnet \
--tx-body-file transaction.raw \
--out-file transaction.signed
```

05 Submit the transaction

Submitting the transaction is also the same.

```
cardano-cli transaction submit \
--tx-file transaction.signed \
--mainnet

> Transaction successfully submitted
```

After submitting, query the recipient address on cardanoscan (it may take some time).