RSK Truffle Token Box¶

Truffle box with everything you need to start creating a token using Open Zeppelin smart contracts library in Truffle framework, connected to a RSK network. It includes network configurations for local node (regtest), Testnet and Mainnet.

Requirements¶

There are a few technical requirements before we start. To useTruffle boxes , you need to have installed in your computer:

- Git
- a POSIX compliant shell
- cURL
- Node.js and NPM
- a code editor

If you don't have any of them installed, go to the tutorialTruffle boxes prerequisites which have all the instructions to setup these requirements.

Truffle framework

Once you have those requirements installed, you only need one command to installTruffle . It is better to do it globally:

npm install -g truffle

Installation¶

1. Create a new folder. For example, create the folderrsk-token
2. . Navigate to the folder in the terminal.

mkdir rsk-tokencd

rsk-token 1. Run the unbox command. This also takes care of installing the necessary dependencies and it can take some time.

truffle unbox rsksmart/rsk-token-box This is the result using Windows OS:

Development console¶

Truffle has an interactive console that also spawns a development blockchain. This is very useful for compiling, deploying and testing locally.

1. Run the development console. This command is successful if you see a list of 10 accounts, a mnemonic and the command prompt is nowtruffle(develop)>

truffle develop You will now be in the truffle develop console with seeded accounts and their associated private keys listed.

C:\RSK\rsk-next>truffle develop

Truffle Develop started at http://127.0.0.1:8545/

Accounts: (0) 0x1056f747cf4bc7710e178b2aeed4eb8c8506c728 (1) 0x45a71c00382c2898b5d6fae69a6f7bfe6edab80c (2) 0x1596384706dc9ac4cca7f50279a4abe591d6c3fe (3) 0x9576d0a496b645baa64f22aceb2328e7468d4113 (4) 0xd431572eef7d77584d944c1809398a155e89f830 (5) 0x92c111839718fe0800fadccc67068b40b8524a0f (6) 0x6da22b5a027146619bfe6704957f7f36ff029c48 (7) 0x2c3a82d8c3993f8c80dcaf91025437bd057df867 (8) 0xc43ae7a44f7deb759177b7093f06512a0a9ff5d7 (9) 0xe61bf00cd7dce248449cfe58f23a4ef7d542bc0b

Private Keys: (0) f32f32839fe27ad906b63eafb326f26fed95c231e3c5e33c7cdd08f62db63167 (1) ebef990088f27f6ef13b5e52a77d5dcc5a76862a701908c586d01b6fe93562b3 (2) 598ccae5e4436fedeb0e798c0d254789c55a63401ebfc3ae8ddde29634ddfcde (3) 09934b80f391e0024b8cb00cd73790fdf64c4d0509e144766414fee317cd3f4e (4) ac745b84b6574b5738d364b43e0d471c9d5107504acc709c90f6f091b78c751b (5) 449654cde095f2349113ef12a93e139b4302bc95adb3619d08adf53dde9b8847 (6) c217f12a89c352fc70b5f1bd5742314b4fb1bb1e35cb779fdb3c2390106355db (7) 1d4c74dfa4e99e161130c18cc63938bb120a128cefbf1b9188efc678bf5722cb (8) 0f44e0becf2e090db498a1b747d2a758fcc81fb0241f350d61117a9c6b1fa82e (9) 85218c5eec657470dafeb09e6f7101f91d21bfe822fbeeecfc9275f798662a63

Mnemonic: virtual valve razor retreat either turn possible student grief engage attract fiber

⚠ Important ⚠ : This mnemonic was created for you by Truffle. It is not secure. Ensure you do not use it on production blockchains, or else you risk losing funds.

truffle(develop)>

Token.sol¶

Take a look at the smart contractToken.sol . You can check it out in foldercontracts .

Token.sol has only 7 code lines! This smart contract is a mintableERC20 token. This means that, in addition to the standard ERC20 specification, it has a function for issuing new tokens.

To create our ERC20 Token, we will importERC20Mintable from Open Zeppelin. This library itself imports several other libraries such asSafeMath.sol , the standards for this kind of token, and the capability to mint tokens.

Inside the token, we define some basic information about the token:name ,symbol , and number ofdecimals for the precision.

To inherit the library's attributes and functions, we simply define our contract as aERC20Mintable using theis keyword in this way.

1. Compile the smart contract.

Note inside the development console we don't preface commands withtruffle .

To make sure you're in the development console, the command prompt must betruffle(develop)> compile Thecompile output should be similar to:

1. Deploy (migrate) the smart contract.

migrate And themigrate output should be similar to:

1. Running contract tests.

This Truffle box also comes with the fileTestToken.js which include some examples for testing the smart contract. You can check it out in thetest folder.

There are many other tests which can be done to check an ERC20 token.

Run this command in the development console:

test Thistest output should be similar to:

Note the command varies slightly if you're in or outside of the development console.

// inside the development console. test // outside the development console. truffle

test

Interact with the token using Truffle console¶

1. Get your accounts in Truffle console.

In the Truffle console, enter:

const

accounts

=

await

web3 . eth . getAccounts () Don’t worry about theundefined return, it is ok. See the addresses after it by entering the command below:

accounts And to view each account:

accounts [0] accounts [1] Take a look in the results:

1. Interact with the token using Truffle console.

First of all, connect with your token

const

token

=

await

Token . deployed () 1. Confirm if our instance is OK.

Enter the instance's name:token , then. , without space hit the TAB button twice to trigger auto-complete as seen below. This will display the published address of the smart contract, and the transaction hash for its deployment, among other things, including all public variables and methods available.

token .

[TAB]

[TAB]

1. Check the total supply

To check if we have tokens already minted, call thetotalSupply function:

(await

token . totalSupply ()). toString () The returned value is 0, which is expected, since we did not perform any initial mint when we deployed the token.

1. Check the token balance

To check the balance of an account, call thebalanceOf function. For example, to check the balance of account 0:

(await

token . balanceOf (accounts [0])). toString () Take a look in the results of total supply and balanceOf:

The returned value is also 0, which is expected, since we did not make any initial mint when we deployed the token, and by definition no accounts can have any tokens yet.

1. Mint tokens

Run this command:

token . mint (accounts [0],

10000) This command sent a transaction to mint 100 tokens for account 0.

You can also mint to a specific address,0xa52515946DAABe072f446Cc014a4eaA93fb9Fd79 :

token . mint ("0xa52515946DAABe072f446Cc014a4eaA93fb9Fd79" ,

10000)

1. Reconfirm the token balance

Check the balance of account 0 again:

(await

token . balanceOf (accounts [0])). toString () The returned value is 10000, which is 100 with 2 decimal places of precision. This is exactly what we expected, as we issued 100 tokens

Also, you can get the balance of a specific address, for example,0xa52515946DAABe072f446Cc014a4eaA93fb9Fd79 :

(await

token . balanceOf ("0xa52515946DAABe072f446Cc014a4eaA93fb9Fd79")). toString () Take a look in the results:

1. Check the total supply (again)

Check the total supply again:

(await

token . totalSupply ()). toString ()

The returned value is 20000, which is 200 with 2 decimal places of precision. After minting 100 tokens for 2 accounts, this is perfect!

1. Transfer tokens

To transfer 40 tokens from account 0 to account 2. This can be done by calling thetransfer function.

token . transfer (accounts [2],

4000 ,

{ from :

accounts [0]})

Account 2 had no tokens before the transfer, and now it should have 40. Account 1 must have 60 tokens. Also the total supply will be the same.

Let's check the balance of each account and the total supply:

(await

token . balanceOf (accounts [2])). toString () (await

token . balanceOf (accounts [0])). toString () (await

token . totalSupply ()). toString () Take a look in the results:

Great! The balances of both accounts and the total supply are correct.

Exit Truffle console¶

In the Truffle console, enter this command to exit the terminal:

.exit

Using RSK networks¶

This Truffle box is already configured to connect to three RSK networks:

1. regtest (local node)
2. testnet
3. mainnet

Testnet will be used here.

We need to do some tasks:

- Setup an account and get R-BTC
- Update RSK network gas price
- Connect to an RSK network
- Deploy in the network of your choose

Setup an account & get R-BTC¶

1. Create a wallet

The easy way to setup an account is using a web3 wallet injected in the browser. Some options are:-Metamask -Nifty

Select the RSK Network in the web wallet. - Nifty: select in the dropdown list - Metamask: go toRSK Testnet to configure it inCustom RPC

You can learn more aboutaccount based RSK addresses .

Take a looktruffle-config.js file to realize that we are usingHDWalletProvider with RSK Networks derivations path: - RSK Testnet dpath:m/44'/37310'/0'/0 - RSK Mainnet dpath:m/44'/137'/0'/0

For more information checkRSKIP57 .

1. Update.secret
2. file

After create your wallet, update your mnemonic in the file.secret , located in the folder project, and save it.

1. Get some R-BTCs:
2. For the RSK Testnet, get tR-BTC fromour faucet
3. .
4. For the RSK Mainnet, get R-BTC froman exchange
5. .

Setup the gas price¶

Gas is the internal pricing for running a transaction or contract. When you send tokens, interact with a contract, send R-BTC, or do anything else on the blockchain, you must pay for that computation. That payment is calculated as gas. In RSK, this is paid inR-BTC . TheminimumGasPrice is written in the block header by miners and establishes the minimum gas price that a transaction should have in order to be included in that block.

To update theminimumGasPrice in our project run this query using cURL:

Testnet

curl https://public-node.testnet.rsk.co/ -X POST -H "Content-Type: application/json"

\

--data '{"jsonrpc":"2.0","method":"eth_getBlockByNumber","params":["latest",false],"id":1}'

\

> .minimum-gas-price-testnet.json Mainnet

curl https://public-node.rsk.co/ -X POST -H "Content-Type: application/json"

\

--data '{"jsonrpc":"2.0","method":"eth_getBlockByNumber","params":["latest",false],"id":1}'

\

> .minimum-gas-price-mainnet.json This query saved the details of latest block to file .minimum-gas-price-testnet.json or .minimum-gas-price-mainnet.json, respectively.

In thetruffle-config.js , we are reading the parameterminimumGasPrice in each json file.

For more information about theGas andminimumGasPrice please go to thegas page .

Connect to RSK Testnet or Mainnet¶

Run the development console for any RSK network.

Console for Testnet

truffle console --network testnet# Console for Mainnet truffle console --network mainnet

Test the connection to RSK network¶

On any of the networks, run this commands in the Truffle console:

Block number¶

Shows the last block number.

(await

web3 . eth . getBlockNumber ()). toString ()

Network ID¶

To get the network ID, run this command:

(await

web3 . eth . net . getId ()). toString () List of network IDs: - mainnet: 30 - testnet: 31 - regtest (local node): 33

Exit the Truffle console:

.exit

Compile and migrate the smart contracts.¶

We will do it running the below commands directly in the terminal, without using the truffle console, this is to show you an alternative.

On any of the networks, run this commands in a terminal (not in Truffle console). To use Testnet or Mainnet, you need to specify this using the parameter-- network :

truffle migrate --network testnet The migrate process in a real blockchain takes more time, because Truffle creates some transactions which need to be mined on the blockchain.

Where to go from here¶

Interact with the token published on an RSK network using Truffle console, doing the same steps which was done before:

- Get your accounts
- Connect with your token
- Check the total supply or the token balance
- Mint tokens
- Transfer tokens

At this point, we have installed all requirements and created a token using Truffle framework and Open Zeppelin smart contracts library, connected to an RSK local node (Regtest), the RSK Testnet and the RSK Mainnet.

Find more documentation

Check out theRSK developers portal .

Do you have questions?

Ask in theRSK chat .