Thunder Token Documentation

Thunder Token Inc.

Jun 14, 2018

Contents:

CHAPTER 1

Introduction to Thunder Token

CHAPTER 2

Tutorials

2.1 Build a DApp on Thunder

In this tutorial, we will be developing a decentralized application (DApp) and deploy it on the Thunder network.

2.1.1 Install Truffle and ganache

Truffle is a popular DApp development framework. Ganache is used for launching a local blockchain for development purposes. They are both written in JavaScript and can be installed with `npm`, the canonical JavaScript package manager.

If you don't have `npm` installed, follow the instructions here to install it.

Once you have `npm` ready, run the following command:

```
npm install -g truffle ganache-cli
```

If you run into permission errors, prepend the commands above with `sudo`.

2.1.2 Install MetaMask

MetaMask is browser extension for using DApps.

To install MetaMask, go to the website and select an installation method depending on your browser.

Once you installed MetaMask, click on the fox icon in your browser and go through the account creation process. You will be asked to save a seed phrase. Make sure to save it somewhere secure; we will need to use it later.

2.1.3 Download the Thunder Truffle box

To make it easier to get started with Thunder, we've created a template for Thunder DApps that can be downloaded with Truffle.

Go ahead and create an empty directory. Then inside the directory, run:

```
truffle unbox thundertoken/dapp-tutorial
```

It may take a minute or two. Once it's done, feel free to explore the directory to see what we have got.

In the following sections, unless otherwise specified, all commands should be run within this directory.

2.1.4 Launching a local blockchain

As aforementioned, during development it's recommended to use a local blockchain. To launch a local blockchain, simply run the following command in a new terminal:

```
ganache-cli
```

You are going to see some console printout. One of the lines towards the end is going to look something like:

```
Mnemonic:      urge club joy arrow finish false then ocean stereo stock this giraffe
```

Go ahead and copy the mnemonic. We will need to use it later.

Now that we have a local blockchain running, let's try deploying our smart contracts on it. In a new terminal, run:

```
truffle deploy
```

If the command finishes running without any errors, the smart contracts have been deployed!

Lastly, we will now configure MetaMask so it uses the local blockchain:

- Click the MetaMask fox icon in your browser.
- Click the network dropdown and select "Localhost 8545"
- Click the menu icon on the upper right corner.
- Click "Log Out".
- Click "Restore from seed phrase".
- Paste the mnemonic that you just copied, then put in some password. Then click "OK".

If everything goes right, you should see that you have around 100ETH. Woah! But keep in mind these are just tokens on your local blockchain and don't have real value.

2.1.5 Launching the DApp

Now that we've launched a local blockchain, deployed a smart contract, and connected MetaMask with the local blockchain, we can finally launch our DApp!

In a new terminal window, run:

```
cd client
npm run start
```

Now, go to `http://localhost:3000`. You should be seeing a page with the title `MyToken`.

Congrats, you've created your first DApp!

So what's this DApp about? It's essentially launching your own ICO, or initial coin offering, where people get tradable tokens that live on the blockchain. In this case, we are calling our token `MyToken`.

If you look at your balance, you should see that you have `1000000` tokens already. We will get to the reason why later. For now, let's try sending the tokens to someone else. Since this is a local blockchain, we don't really have other people to send to. Fortunately it's easy enough to create a new address on a blockchain, so let's do just that:

- Click the MetaMask icon in your browser.
- Click the accounts button, which is the first button in the upper right corner.
- Click "Create Account".

You've created a new account! Now, click the three dots next to the header "Account 2", and click "Copy Address to clipboard." Then click the accounts button again to switch back to "Account 1".

Now go back to the DApp. In the `To` field, paste in the address you just copied. In the `Amount` field, put in the amount of tokens you'd like to send. Finally, click `Send` and confirm the transaction through the MetaMask pop-up.

Now, you should see that the balance shown in the web page has been reduced by the correct amount. Now try switching the account to "Account 2" and refresh the page. You should see that the balance reflects the amount you just sent.

2.1.6 Updating the DApp

So we've just launched our first token, which is very cool. However, it'd be cooler if we could give the token a real name as opposed to a generic name like `MyToken`.

Open the file `contracts/Token.sol` in your favorite editor. Even if you are not already familiar with Solidity (the smart contract programming language), you can probably make sense of the code:

- On line 6, we are setting the name of the token to `MyToken`.
- On line 7, we are giving our token the symbol `MT`, similar to how USD is short for the US dollars.
- On line 9, we are setting the initial supply of the money to 1 million.

Go ahead and change all of that! Give your token a better name, a cooler symbol, and a different initial supply.

Once you've done that, run the following commands again:

```
truffle compile
truffle deploy --reset
```

Now go back to the DApp and make sure you are using Account 1 in MetaMask. You should see your new token with the amount you specified as the initial supply.

2.1.7 Connecting to the Thunder testnet

It was very cool launching your own token. However, it's no fun if you can only send it to yourself. Now let's "go live" by deploying the DApp to the Thunder testnet, so other people can use your tokens.

The first thing we need to do is to connect MetaMask with the Thunder testnet. Here are the steps:

- Click the MetaMask fox icon in your browser.
- Log out of your current account (which is provided by the local blockchain) by clicking the menu button, and then click "Log Out".
- Click "Restore from seed phrase" again and restore your original account with your original seed phrase.
- Click the dropdown that says "Private Network".
- Select "Custom RPC".

- In the field that says "New RPC URL", enter `https://testnet.thundertoken.com`.

With any luck, you should see that your account has been refreshed with a balance of 0. This is because you don't own any Thunder tokens yet. Let's go ahead and get some Thunder tokens!

2.1.8 Getting Thunder testnet tokens

Head straight to "https://testnet.thundertoken.com"...

2.1.9 Deploying the DApp on Thunder testnet

We've made it very easy to deploy on the Thunder testnet. Go back to the code directory and open `truffle.js` in your favourite editor. Edit the first line to use your actual mnemonic, such as this:

```
// This is just an example!!  You want to use your actual mnemonic.
let mnemonic = "candy maple cake sugar pudding cream honey rich smooth crumble sweet␣
↪treat";
```

Once that's done, simply run the following command:

```
truffle deploy --network thunder
```

The command is self-explanatory: we are deploying the smart contracts on Thunder.

When the command has finished running, go back to your DApp running on `http://localhost:3000`.

Again, you should see that you already have `1000000` tokens. But this time, the tokens are live on the Thunder testnet, and you can send tokens to anyone with a Thunder testnet address!

But wait, you might say, how do people use my tokens if the DApp is running on `localhost`? That's an excellent question. To make sure others can actually access your DApp, we will be using the free hosting service Github pages.

2.1.10 Deploying the frontend on Github pages

Head to github and create a new repo. You should name the repo after your token, since your DApp will eventually be available at `https://<username>.github.io/<repo-name>`.

Once you've created the repo, push your code into the repo. The steps are roughly as follows:

```
git init
git add .
git commit -m "Init"
git remote add origin <your-repo-url>
git push -u origin master
```

Once that's done, open up `package.json` in your favorite editor and edit the `homepage` field to look like the following:

```
{
  "homepage": "https://<username>.github.io/<repo-name>",
}
```

You want to replace `<username>` with your actual github username and `<repo-name>` with your actual repo name.

Finally, go into the `client/` subdirectory and run:

```
npm run deploy
```

Once the command finishes running, you can head to your `homepage` URL and you should see your DApp! This time, the DApp is running on the public internet and available for everyone to use.

2.2 Migrate a DApp to Thunder

In this tutorial, we will be migrating an existing decentralized application (DApp) to Thunder. We assume that your project has been set up using the popular tool Truffle, though the workflow should be easily adeptable to other tools. If you are having trouble deploying an existing DApp to Thunder, drop a line at dev@thundertoken.com and we will be happy to help.

Since Thunder is 100% compatible with Ethereum, migrating an existing DApp to Thunder is as easy as installing a package and changing a couple lines in your config.

2.2.1 Installing dependencies

Run the following command in your project's root directory (where the `contracts` directory is located):

```
npm install truffle-hdwallet-provider
```

2.2.2 Update Truffle config

Open up `truffle.js` (or `truffle-config.js` if you are using Windows) and add the following code to the top of the file:

```
let HDWalletProvider = require('truffle-hdwallet-provider')
let mnemonic = "<the mnemonic for your private key>"
```

Note that the `mnemonic` variable needs to be set to a real mnemonic.

Now, in your module exports, include a `thunder` section under the `networks` section, like the following:

```
module.exports = {
  networks: {
    thunder: {
      provider: function() {
        return new HDWalletProvider(mnemonic, "https://testnet.thundertoken.com")
      },
      network_id: "*"
    }
  }
}
```

2.2.3 Deploy smart contracts on Thunder

To deploy your smart contracts on Thunder, simply run:

```
truffle migrate --network thunder
```

If you didn't see any error messages, you should be all set. It was that easy!

Note that smart contracts are only a part of the overall DApp. For instance, you still need to customize your frontend so that it uses the smart contracts deployed on Thunder. We won't detail the steps here since they are very much dependent on your specific project, but feel free to drop us a line at dev@thundertoken.com and we will be happy to help.

CHAPTER 3

Getting Thunder testnet tokens

CHAPTER 4

Configuring MetaMask

CHAPTER 5

Frequently Asked Questions

CHAPTER 6

DApp Developer Resources