




Liquidity Pool

Definiere & deploye eine ERC20 basierte Währung, lege bei uniswap.org einen liquidity pool für diese an und führe einen automatisierten airdrop durch.

Neuen Coin erstellen

Um die Aufgabe zu lösen muss zuerst ein neuer Coin erstellt werden. Dies wird auf dem "Ropsten" Netzwerk passieren.

- Auf "MetaMask" das Testnetzwerk auswählen.
- Webseite "Remix" besuchen:

Remix - Ethereum IDE
 <http://remix.ethereum.org>

- Dort neue ".sol" Datei im "contracts" Ordner erstellen

```
// SPDX-License-Identifier: GNU GPL
pragma solidity >=0.8.0 < 0.9.0;

import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v4.4/contracts/token/ERC20/ERC20.sol";

contract RandomCoin is ERC20 {

    constructor () ERC20("RandomCoin", "RAND") {
        _mint(msg.sender, 1000 * 10 ** 18 );
    }

}
```

In diesem Codebeispiel werden 1000 RandomCoins mit 18 Nachkommastellen erstellt.

- Diesen Code compillieren

Da das deployen eines Smart-Contracts nicht kostenlos ist benötigen wir zu Beginn ein wenig ETHER um das deployment zu bezahlen.

- Auf der Website <https://faucet.egorfine.com/> seine Wallet Adresse eingeben
 - Dadurch wird dem Wallet 0.3 Ether zugeschrieben (Auf dem Ropsten Testnetzwerk)
- Unter Deploy & Run Transactions:
 - Environment: Injected Web3
 - Account: Eigenen Wallet angeben

- Deploy
- Link von Testcoin kopieren

Hinzufügen von Coin zu eigenem Wallet

Damit mit den eigenen Coins gehandelt werden kann, müssen diese in das eigene Wallet übertragen werden.

- Auf <https://ropsten.etherscan.io/> gehen und Wallet Adresse eingeben
 - Dort sollte Transaction des Smart-Contracts zu sehen sein
- In der Transaktion die Adresse des Contracts kopieren
- In Metamask unter "Import Tokens" eingeben
- Innerhalb des Metamask Wallets sollten die erstellten Coins zu sehen sein

Liquidität hinzufügen

Nachdem ein Testcoin erstellt wurde, muss diesem auch eine Liquidität zugeschrieben werden.

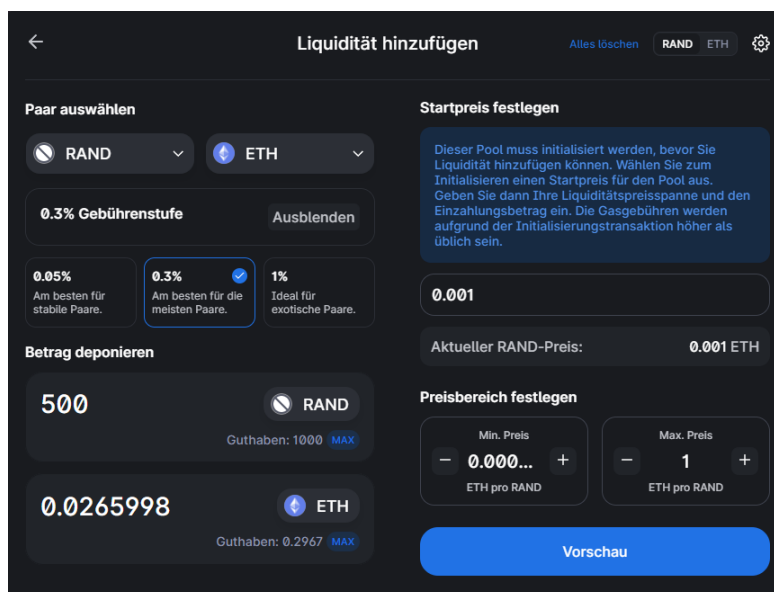
- Diese Aufgabe kann auf "app.uniswap.org" erledigt werden

Uniswap Interface

Swap or provide liquidity on the Uniswap Protocol

 <http://app.uniswap.org>

- Auf Website zu "Pool" wechseln
- Neue Position hinzufügen



← Liquidität hinzufügen Alles löschen RAND ETH ⚙️

Paar auswählen

RAND ETH

0.3% Gebührenstufe Ausblenden

0.05% Am besten für stabile Paare. **0.3%** Am besten für die meisten Paare. **1%** Ideal für exotische Paare.

Betrag deponieren

500 RAND Guthaben: 1000 MAX

0.0265998 ETH Guthaben: 0.2967 MAX

Startpreis festlegen

Dieser Pool muss initialisiert werden, bevor Sie Liquidität hinzufügen können. Wählen Sie zum Initialisieren einen Startpreis für den Pool aus. Geben Sie dann Ihre Liquiditätspreisspanne und den Zahlungsbetrag ein. Die Gasgebühren werden aufgrund der Initialisierungstransaktion höher als üblich sein.

0.001

Aktueller RAND-Preis: **0.001 ETH**

Preisbereich festlegen

Min. Preis: **0.000...** ETH pro RAND + -

Max. Preis: **1** ETH pro RAND + -

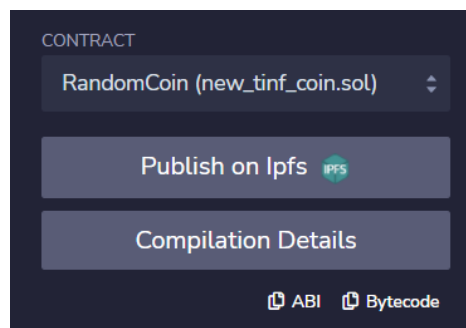
Vorschau

Um eine Liquidität hinzuzufügen muss eine Menge von Testcoins eingezahlt werden. Andere Nutzer können diesen Coin dann für einen festgelegten Preis abkaufen. Bei jeder Transaktion werden außerdem eine kleine Gebühr gezahlt.

Automatischer Airdrop

Um Coins auf andere Wallets zu übertragen kann ein automatisierter Airdrop durchgeführt werden. Dabei wird vom Ursprungswallet ein bestimmter Coin ausgewählt der versendet werden soll. In einer Liste werden die Zielwallets angegeben.

- Zu Beginn muss eine Provider URL erstellt werden
 - Dies wird auf <https://infura.io/> erstellt
 - Die `wss://ropsten.infura.io/...` Adresse kopieren
 - **WICHTIG:** "Endpoints" muss auf "ROPSTEN" gesetzt werden
- Falls die Deno Laufzeitumgebung noch nicht auf dem System installiert ist, sollte das nun auf deno.land gemacht werden
- Das Deno Repository unter <https://deno.land/x/airdrop> herunterladen
- Auf <https://remix.ethereum.org> die ABI Datei für den zu versendenden Coin herunterladen
 - Im Solidity Compiler sollte nach dem Compilieren ein Button für die ABI erscheinen



- Die Datei dann als .ts Datei im Projekt Ordner abspeichern
 - Die erste Zeile der Datei anpassen

```
export const randomCoin = [  
  {  
    "inputs": [],  
    "stateMutability": "nonpayable",  
    ...
```

- Zum Ausführen des Airdrops muss die `launch-airdrop.ts` Datei angepasst werden
 - `ropsten.ts` einbinden

- Empfänger der Coins angeben
- airdropservice Instanz anpassen

```
import { yourCoinName } from "../path/to/your/abi/file/ropsten.ts"
```

```
const airdropRecipients = [
  "wallet_empfänger_1",
  "wallet_empfänger_2",
  "wallet_empfänger_3"
]
```

```
const airdropService = new AirdropService(providerURL,
"smart_coin_address", yourCoinName, 1, privateKeySender)
```

- Eigenes Wallet in airdrop-service.ts einbinden

```
private walletofSender: string = "your_wallet_address"
```

- Nachdem die Dateien angepasst wurden, kann das Programm gestartet werden
- Deno Befehl in Kommandozeile
 - Eigenen Privatekey aus Metamask kopieren

```
deno run --allow-net launch-airdrop.ts
wss://ropsten.deine.infura.is/adresse dein_private_key
```

Contributors

- Tobias Barz
- Florian Ott
- Jens Wölpert
- Daniel Vallelonga