Simulating Uniswap - HardHat and Mainnet Forking

---

1. Introduction

This project delves into the world of Uniswap, a leading decentralized exchange built on the Ethereum blockchain. We embark on a journey to understand its functionalities and capabilities, with a specific focus on using Hardhat and a Mainnet Fork for a risk-free, yet realistic learning experience.

Directly interacting with Uniswap on the mainnet often poses a significant barrier for beginners including but not limited to:

- The high cost of tokens
- Obtaining these tokens can be challenging for beginners

To overcome this challenge, we leverage Hardhat, a powerful development environment, and a Mainnet Fork, a replica of the existing Mainnet blockchain. This combination allows us to simulate real-world scenarios and transactions without jeopardizing actual assets.

Throughout this document, we will examine various aspects of Uniswap, including

Trading: Understanding the mechanics of swapping tokens on the platform.

Liquidity Pools: Delving into the concept of providing liquidity and earning rewards.

By leveraging Hardhat and the Mainnet Fork, we can intricately explore each of these facets without financial risk. This project aims to provide a comprehensive understanding of Uniswap and its functionalities.

2. So what is Uniswap (https://app.uniswap.org/swap)

Uniswap is a decentralized exchange (DEX) that facilitates peer-to-peer trading of cryptocurrency tokens on the Ethereum blockchain. Unlike centralized exchanges like Coinbase, Uniswap does not rely on an intermediary to control and manage user funds. Instead, it employs an automated market maker (AMM) model, which utilizes smart contracts to automate the pricing and execution of trades.

Key Features of Uniswap:

Decentralization: Uniswap is not controlled by any single entity, making it resistant to censorship and manipulation.

Non-custodial: Users retain full control of their private keys and cryptocurrency assets.
Peer-to-peer trading: Trades are directly between users, without the need for an intermediary.
Automated market maker (AMM) model: Smart contracts manage the pricing and execution of trades based on the supply and demand of tokens in liquidity pools.
Wide range of token listings: Uniswap supports a vast array of Ethereum-based tokens, including ERC-20 tokens and ERC-721 NFTs.

3. Hardhat

What is Hardhat?
Hardhat is an Ethereum development environment designed for professionals. It's a flexible, extensible, and fast framework that helps developers manage and automate the repetitive tasks inherent in building smart contracts and dApps (decentralized applications), while also enabling them to easily introduce custom functionalities.

Key Features:

- **Smart Contract Compilation, Deployment, and Testing:** Hardhat simplifies the entire lifecycle of smart contracts, from writing and testing to deploying them on the Ethereum network.
- **Hardhat Runtime Environment (HRE)**: Provides a runtime environment for scripting and automating development tasks, offering a consistent environment across all tasks.
- **Interactive Console**: Offers an interactive JavaScript console for direct interaction with smart contracts and the Ethereum network.
- **Network Management**: Easy configuration for different networks, allowing for smooth deployment on mainnet, testnets, or local development networks.
- **Hardhat Network**: A built-in local Ethereum network designed for development, which provides features like automatic mining, error stack traces, and console.log in contracts.
- **Stack Traces**: When a transaction fails, Hardhat provides detailed stack traces, pointing exactly where the failure occurred.
- **Ethers and Web3.js Integration**: Built-in integrations with popular Ethereum libraries like Ethers.js and Web3.js.
- **Plugin Ecosystem**: Hardhat can be extended through plugins. This allows developers to easily integrate other tools and services into their development workflow.
- **Waffle and Truffle Support**: Compatibility with testing frameworks like Waffle and Truffle, allowing developers to write and run tests for their contracts in a flexible environment.

Benefits of Using Hardhat:

- **Enhanced Developer Productivity**: Automates common tasks, provides detailed error messages, and integrates with other tools, all of which speed up the development process.
- **Ease of Learning and Use:** Designed with simplicity in mind, making it accessible for newcomers while still being powerful for experienced developers.
- **Customizable and Extendable:** The ability to extend its functionality through plugins allows developers to tailor the environment to their specific needs.
- **Active Community and Support**: A growing community that contributes to its development, offers support and creates a wide range of plugins.

- **Realistic Development Environment**: The Hardhat Network simulates the real Ethereum network, providing a realistic environment for testing and development.

4. Hardhat Configuration (hardhat.config)

- **Networks Configuration:** This section allows you to define settings for different Ethereum networks. You can specify local networks for development, test networks like Rinkeby or Ropsten, and the main Ethereum network. Each network configuration can include parameters such as the network's URL, accounts to use for deployment, gas price, and other network-specific settings.
- **Solidity Configuration:** Here, you specify the version of Solidity you are using. Hardhat allows you to compile contracts with different Solidity versions at the same time, and you can define specific compiler settings like optimizations.
- **Paths**: You can customize the paths to your sources, tests, artifacts, and cache directories. This is useful if you are integrating Hardhat into an existing project with a non-standard directory structure.
- Plugins: Hardhat's functionality can be extended through plugins. In this section, you can list and configure the plugins your project uses.
- **Tasks**: Hardhat allows the creation of custom tasks. You can define these tasks in your configuration file, scripting complex workflows specific to your project's needs.
- **Mocha**: For testing, Hardhat uses Mocha under the hood. You can configure Mocha settings like timeout limits, enabling or disabling colors in the output, and other testing behaviors.

Example of a Basic Hardhat Configuration

```
require("@nomiclabs/hardhat-waffle");

module.exports = {
  solidity: "0.8.4",
  networks: {
    hardhat: {
      chainId: 1337
    },
    ropsten: {
      url: "https://eth-ropsten.alchemyapi.io/v2/your-api-key",
      accounts: ['your-private-key']
    }
  }
};
```

5. Requirements for Forking the Mainnet

Forking the Ethereum mainnet is a process that creates a local copy of the entire state of the Ethereum network at a specific block, allowing developers to interact with this copy as if it were the real network. This process is incredibly valuable for testing purposes. To successfully fork the Ethereum mainnet, several key requirements need to be met:

Node.js and npm

1. Node.js: A JavaScript runtime environment that is essential for running the Hardhat environment and other Ethereum development tools. It executes JavaScript code outside of a web browser.
2. npm (Node Package Manager): Accompanies Node.js and is used for managing JavaScript packages. It's used to install Hardhat and other dependencies.

Ethereum Node or Provider API

1. To fork the mainnet, you need access to the state and history of the Ethereum network. This can be achieved either by running your own Ethereum node or by using a service that provides access to Ethereum nodes.
2. Services like Infura or Alchemy offer API access to Ethereum nodes. These services provide a URL that will be used in Hardhat configuration to connect to the Ethereum network.

Hardhat

1. Hardhat is the development environment that facilitates main net forking. It's crucial to have Hardhat installed in your project.
2. Hardhat can be installed using npm by running npm install --save-dev hardhat.

Sufficient Disk Space and RAM

1. Forking the mainnet involves replicating the state of the entire Ethereum blockchain at a particular block. This requires significant computational resources.
2. Ensure your development machine has enough disk space (several gigabytes at minimum) and RAM (at least 8GB, preferably more) to handle the data involved in this process.

An IDE or Code Editor

1. A suitable Integrated Development Environment (IDE) or code editor (like Visual Studio Code) is necessary for writing and managing your smart contracts and scripts.

A Stable Internet Connection

1. Forking the mainnet requires a continuous and stable internet connection, as it involves interacting with an Ethereum node or a third-party node provider service over the network.

MetaMask

1. For interacting with the forked network via a user interface, MetaMask or a similar Ethereum wallet can be used. This allows for transactions and interactions with the local version of the mainnet.

6. Forking Mainnet Using Hardhat (Step-by-Step Guide)

Setting Up Hardhat

First, you need to have a Hardhat project set up. If you haven't already done so, you can create a new Hardhat project by following these steps:

1. Install Hardhat: Run `npm install --save-dev hardhat` in your project directory.
2. Initialize Hardhat: Run `npx hardhat` and follow the prompts to create a basic Hardhat project.

Configuring Hardhat for Mainnet Forking

In your hardhat.config.js file, you need to add configurations for the mainnet fork:

Specify the Network: Under the networks section, add a configuration for the Hardhat network. Here, you will specify the URL of an Ethereum node (or a service like Infura or Alchemy) and the block number you want to fork from.

```
require("@nomiclabs/hardhat-waffle");

module.exports = {
  networks: {
    hardhat: {
      forking: {
        url: "https://eth-mainnet.alchemyapi.io/v2/your-api-key",
        blockNumber: 12345678 // Optional: Specify a block number to fork from
      }
    }
  },
  solidity: "0.8.4",
};
```

In this configuration, replace "https://eth-mainnet.alchemyapi.io/v2/your-api-key" with the URL provided by your Ethereum node service and set block number to the specific block you want to fork from (or omit it to fork from the latest block).
https://www.alchemy.com/

Starting the Hardhat Network
Run your Hardhat network with the command `npx hardhat node`. This will start a local Ethereum network that forks from the specified block on the mainnet.
The console will display a list of accounts and private keys. These accounts are pre-funded with Ether in your local fork and can be used for testing.

7. Connecting MetaMask to the Local Hardhat Network

MetaMask is a popular Ethereum wallet and a gateway to blockchain apps. It's commonly used for interacting with Ethereum networks, including custom networks like a local Hardhat environment. Here's a detailed guide on connecting MetaMask to your local Hardhat network:

Setting Up the Local Hardhat Network
First, ensure that your local Hardhat network is running. If you've forked the mainnet (or set up a local network), Hardhat should be running on your local machine. Typically, Hardhat runs on http://127.0.0.1:8545.

Installing and Opening MetaMask
If you haven't already, install MetaMask as a browser extension from the official MetaMask website. Once installed, open the MetaMask extension in your browser.

Adding a Custom RPC Network

1. In MetaMask, click on the network selection dropdown at the top (which might be set to Ethereum Mainnet by default).
2. Select "Custom RPC" at the bottom of the list to add a new network.
3. In the "Network Name" field, enter a name for your local network (e.g., "Hardhat Local").
4. In the "New RPC URL" field, enter the URL of your local Hardhat network, which is usually http://127.0.0.1:8545.
5. Leave the "Chain ID" field with the default value provided by Hardhat, which is usually 31337. However, this can be configured in your hardhat.config.js file.
6. The other fields like "Symbol" (which can be set to "ETH") and "Block Explorer URL" can be left blank as they are not relevant for a local network.

Connecting and Testing the Network

1. Once you've added the custom network, select it from the network dropdown.
2. You should now be connected to your local Hardhat network.

Importing Accounts

1. Hardhat provides a list of accounts and private keys when it's running. You can import one of these accounts into MetaMask to interact with the local network.
2. In MetaMask, click on the profile icon and select "Import Account".
3. Copy and paste the private key of one of the Hardhat accounts and click "Import".
4. This account will now be added to MetaMask and should have a balance of Ether, which is prefilled by Hardhat for testing purposes.

Interacting with Your Local Network

You can now use this MetaMask account to deploy contracts, send transactions, and interact with your local Hardhat environment just as you would on the main Ethereum network.

8. Using Uniswap with Forked Mainnet on HardHat

Post all the above steps, after you have your Metamask connected to your hardhat - you can go to https://app.uniswap.org/swap and perform all the swaps and NFT transactions.

Steps:

- Now that you have imported an account into your Metamask from your HardHat with 10000ETH, you can go ahead and swap ETH with any other pair at any given time.
- As visible in the below image, you can enter:
    - Pay = 1ETH
    - Select a token you want to exchange it with
    - You will see a conversion rate based on the current market rate of the pair
    - Click on the Swap button
    - Your MetaMask wallet will open
    - Verify the swap information
    - Click on **Confirm** and confirm the transaction

- As of this point, you will see the transaction getting confirmed on your local hardhat blockchain instead of the Ethereum mainnet.
- However, let's keep in mind - since we have forked the mainnet the exchange rates and all the related information about the transaction will be based on real-time data on the Ethereum mainnet.