Lesson 7

Hardhat Fund Me

Hikmah Nisya - 1103184094
Radzis Araaf Jaya Jamaludin - 1103184234
Raudhatul Rafiqah Assyahiddini - 1103180225

Hardhat Fund Me

- When Hardhat is run, it searches for the closest hardhat.config.js file starting from the Current Working Directory. This file normally lives in the root of your project. An empty hardhat.config.js is enough for Hardhat to work.

- The entirety of your Hardhat setup (i.e. your config, plugins and custom tasks) is contained in this file.

Config Hardhat

To set up your config, you have to export an object from `hardhat.config.js`.

This object can have entries like `defaultNetwork`, `networks`, `solidity`, `paths` and `mocha`. For example:

```
module.exports = {
  defaultNetwork: "rinkeby",
  networks: {
    hardhat: {
    },
    rinkeby: {
      url: "https://eth-rinkeby.alchemyapi.io/v2/123abc123abc123abc123abc123abcde",
      accounts: [privateKey1, privateKey2, ...]
    }
  },
  solidity: {
    version: "0.5.15",
    settings: {
      optimizer: {
        enabled: true,
        runs: 200
      }
    }
  },
  paths: {
    sources: "./contracts",
    tests: "./test",
    cache: "./cache",
    artifacts: "./artifacts"
  },
  mocha: {
    timeout: 40000
  }
}
```

Network Configuration

The networks config field is an optional object where network names map to their configuration.

There are two kinds of networks in Hardhat: JSON-RPC based networks, and the built-in Hardhat Network.

You can customize which network is used by default when running Hardhat by setting the config's defaultNetwork field. If you omit this config, its default value is "hardhat".

Hardhat Network

Hardhat comes built-in with a special network called hardhat. When using this network, an instance of the Hardhat Network will be automatically created when you run a task, script or test your smart contracts.

Hardhat Network has first-class support of Solidity. It always knows which smart contracts are being run and exactly what they do and why they fail. Learn more about it here.

See the Hardhat Network Configuration Reference for details on what can be configured.

JSON-RPC Network

These are networks that connect to an external node. Nodes can be running in your computer, like Ganache, or remotely, like Alchemy or Infura.

his kind of network is configured with objects with the following fields:

url: The url of the node. This argument is required for custom networks.

chainId: An optional number, used to validate the network Hardhat connects to. If not present, this validation is omitted.

from: The address to use as default sender. If not present the first account of the node is used.

gas: Its value should be "auto" or a number. If a number is used, it will be the gas limit used by default in every transaction. If "auto" is used, the gas limit will be automatically estimated. Default value: "auto".

gasPrice: Its value should be "auto" or a number. This parameter behaves like gas. Default value: "auto".

gasMultiplier: A number used to multiply the results of gas estimation to give it some slack due to the uncertainty of the estimation process. Default value: 1.

accounts: This field controls which accounts Hardhat uses. It can use the node's accounts (by setting it to "remote"), a list of local accounts (by setting it to an array of hex-encoded private keys), or use an HD Wallet. Default value: "remote".

httpHeaders: You can use this field to set extra HTTP Headers to be used when making JSON-RPC requests. It accepts a JavaScript object which maps header names to their values. Default value: undefined.

timeout: Timeout in ms for requests sent to the JSON-RPC server. If the request takes longer than this, it will be cancelled. Default value: 40000 for the localhost network, 20000 for the rest.

Default networks object

```
{
    localhost: {
      url: "http://127.0.0.1:8545"
    },
    hardhat: {
      // See its defaults
    }
}
```

HD WALLET CONFIG

HD Wallet config

To use an HD Wallet with Hardhat you should set your network's `account` with the following fields:

- `mnemonic` : A required string with the mnemonic phrase of the wallet.
- `path` : The HD parent of all the derived keys. Default value: `"m/44'/60'/`
- `initialIndex` : The initial index to derive. Default value: `0` .
- `count` : The number of accounts to derive. Default value: `20` .
- `passphrase` : The passphrase for the wallet. Default value: empty string.

For example:

```
module.exports = {
  networks: {
    rinkeby: {
      url: "...",
      accounts: {
        mnemonic: "test test test test test test test test test t
        path: "m/44'/60'/0'/0",
        initialIndex: 0,
        count: 20,
        passphrase: "",
      },
    },
  },
};
```

Solidity Configuration

The solidity config is an optional field that can be one of the following:

A solc version to use, e.g. "0.7.3".

An object which describes the configuration for a single compiler. It contains the following keys:

version: The solc version to use.

settings: An object with the same schema as the settings entry in the Input JSON.

An object which describes multiple compilers and their respective configurations. It contains the following:

compilers: A list of compiler configuration objects like the one above.

overrides: An optional map of compiler configuration override objects. This maps file names to compiler configuration objects. Take a look at the compilation guide to learn more.

Path Configuration

- You can customize the different paths that Hardhat uses by providing an object to the paths field with the following keys:
- root: The root of the Hardhat project. This path is resolved from hardhat.config.js's directory. Default value: the directory containing the config file.
- sources: The directory where your contract are stored. This path is resolved from the project's root. Default value: './contracts'.
- tests: The directory where your tests are located. This path is resolved from the project's root. Default value: './test'.
- cache: The directory used by Hardhat to cache its internal stuff. This path is resolved from the project's root. Default value: './cache'.
- artifacts: The directory where the compilation artifacts are stored. This path is resolved from the project's root. Default value: './artifacts'.