DOP: / /2023 DOS: / / 2023

Experiment No: 03

Aim: To deploy and publish smart contract on Ethereum test network.

Theory:

Ethereum

Ethereum is a decentralized blockchain platform that enables developers to build and deploy decentralized applications (DApps). It was created by Vitalik Buterin in 2015 and has since become one of the most popular blockchain platforms.

Ethereum's native cryptocurrency is called Ether (ETH), which is used to pay for transaction fees and smart contract execution fees on the Ethereum network. Smart contracts are self-executing contracts that live on the Ethereum blockchain and can be programmed to automatically perform actions based on predefined conditions.

Ethereum test networks are separate blockchain networks that are designed to mimic the main Ethereum network. They are used for testing and development purposes, allowing developers to test their DApps and smart contracts without having to spend real Ether on the main network. The most popular Ethereum test networks are Rinkeby, Ropsten, and Kovan.

Test networks use test Ether, which can be obtained for free from faucets or other sources. Transactions on test networks are usually faster and cheaper than on the main network, making them ideal for testing and development. However, test networks are also less secure than the main network, and transactions on test networks are not final and can be easily reversed or censored.

Steps:

1. First, make sure you have a valid Ethereum account and some test ether on the test network you want to deploy to. You can get test ether from a faucet for the test network you want to use.

2. Create a new Truffle project using the following command:

truffle init

3. Create a new Solidity smart contract inside the contracts directory. Here's an example SimpleStorage.sol contract that allows you to set and get a single value:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract SimpleStorage {
    uint256 private _value;

    function set(uint256 value) public {
        _value = value;
    }

    function get() public view returns (uint256) {
        return _value;
    }
}
```

4. Add the @truffle/hdwallet-provider package to your project using the following command:

npm install @truffle/hdwallet-provider

5. Add the following configuration to your truffle-config.js file to specify the network you want to deploy to and your Infura project ID:

```
const HDWalletProvider = require('@truffle/hdwallet-provider');
const infuraProjectId = '<YOUR_INFURA_PROJECT_ID>';
const mnemonic = '<YOUR_METAMASK_MNEMONIC>';

module.exports = {
  networks: {
    rinkeby: {
      provider: () =>
        new HDWalletProvider({
          mnemonic: mnemonic,
          providerOrUrl: `https://rinkeby.infura.io/v3/${infuraProjectId}`,
          chainId: 4,
          gasPrice: 10000000000, // 10 gwei
        }),
      network_id: 4, // Rinkeby's id
      gas: 5500000, // Gas limit used for deploys
    },
  },
};
```

Make sure to replace **<YOUR_INFURA_PROJECT_ID>** with your Infura project ID and **<YOUR_METAMASK_MNEMONIC>** with your MetaMask mnemonic.

6. Compile your smart contract using the following command:

truffle compile

```
Compiling your contracts...
===========================
> Compiling ./contracts/SimpleStorage.sol
> Compiling @openzeppelin/contracts/access/Ownable.sol
> Compiling @openzeppelin/contracts/token/ERC20/IERC20.sol
> Compiling @openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol
> Compiling @openzeppelin/contracts/token/ERC20/ERC20.sol
> Artifacts written to /path/to/your/truffle-project/build/contracts
> Compiled successfully using:
   - solc: 0.8.7+commit.e28d00a7.Emscripten.clang
```

7. Deploy your smart contract to the test network using the following command:

truffle migrate --network rinkeby

```
Starting migrations...
======================
> Network name:    'rinkeby'
> Network id:      4
> Block gas limit: 0x989680


1_initial_migration.js
======================

   Deploying 'Migrations'
   ----------------------
   > transaction hash:    0x12aaf84c8610b7fa57cfe9340cb7197ec3c3f91d7c4e4f4dc8f221b
   > Blocks: 2            Seconds: 13
```

Conclusion: You have deployed and published smart contract on Ethereum test network.