ERC-20 Token Deployment on Ethereum Mainnet using Infura

1. Install Dependencies

Before starting, ensure you have Node.js and npm installed from Node.js.

```
mkdir my-erc20-token
cd my-erc20-token
npm init -y
npm install --save-dev hardhat @nomiclabs/hardhat-ethers ethers
npm install @openzeppelin/contracts
```

2. Hardhat Setup

Initialize a new Hardhat project:

```
npx hardhat
```

Select **"Create an empty hardhat.config.js"**.

3. Create the ERC-20 Contract

Create a file `contracts/MyToken.sol` with the following code:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
import "@openzeppelin/contracts/access/Ownable.sol";
import "@openzeppelin/contracts/utils/Pausable.sol";

contract MyToken is ERC20, Ownable, Pausable {
    constructor() ERC20("MyToken", "MTK") Ownable(msg.sender) {
        _mint(msg.sender, 1_000_000 * 10 ** decimals()); // 1 million
tokens
    }

    function mint(address to, uint256 amount) public onlyOwner {
        _mint(to, amount);
    }

    function burn(uint256 amount) public {
        _burn(msg.sender, amount);
    }

    function pause() public onlyOwner {
        _pause();
    }

    function unpause() public onlyOwner {
        _unpause();
    }
```

```
}
```

4. Configure Hardhat for Deployment

Edit `hardhat.config.js`:

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

module.exports = {
  solidity: "0.8.20",
  networks: {
    mainnet: {
      url: "https://mainnet.infura.io/v3/YOUR_INFURA_PROJECT_ID", // 
Infura URL
      accounts: ["YOUR_WALLET_PRIVATE_KEY"] // Your Metamask private 
key
    }
  }
};
```

Replace `YOUR_INFURA_PROJECT_ID` with your Infura API project ID from Infura and `YOUR_WALLET_PRIVATE_KEY` with your Ethereum wallet private key.

5. Create the Deployment Script

Create a file `scripts/deploy.js` with the following content:

```
async function main() {
  const [deployer] = await ethers.getSigners();
  console.log("Deploying contracts with the account:", 
deployer.address);

  // Obtain contract
  const Token = await ethers.getContractFactory("MyToken");
  console.log("Deploying MyToken...");
  const token = await Token.deploy();
  await token.deployed();

  console.log("MyToken deployed to:", token.address);
}

main()
  .then(() => process.exit(0))
  .catch((error) => {
    console.error(error);
    process.exit(1);
  });
```

6. Compile the Contract

Run the following command:

```
npx hardhat compile
```

If successful, you should see a message like:

```
Compiled 8 Solidity files successfully (evm target: paris).
```

7. Deploy to Ethereum Mainnet

To deploy the contract, run:

```
npx hardhat run scripts/deploy.js --network mainnet
```

If successful, you will see:

```
Deploying contracts with the account: 0xYourWalletAddress
Token deployed to: 0xYourTokenAddress
```

8. Verify the Contract on Etherscan

To verify the contract on Etherscan, use:

```
npx hardhat verify --network mainnet 0xYourTokenAddress
```

Replace `0xYourTokenAddress` with the actual deployed contract address.

9. Testing and Usage

Your contract is now deployed on Ethereum Mainnet. You can interact with it using ethers.js, web3.js, or explorers like Etherscan.

10. Tree

this is how your file organization should look like