

Using Hardhat for Deploying Smart Contracts on BSC

In this tutorial, we explain step-by-step how to create, compile and deploy a simple smart contract on the BSC Testnet using Hardhat.

What is Hardhat

Hardhat is a development environment to compile, deploy, test, and debug your smart contract.

Setting up the development environment

There are a few technical requirements before we start.

Pre-requisites

There are a few technical requirements before we start as listed below:

- **Node.js v10+ LTS and npm** (comes with Node)
- **Git**
- Create an empty project `npm init --yes`
- Once your project is ready, run `npm install --save-dev hardhat` to install Hardhat.
- Install hardhat toolbox `npm install @nomicfoundation/hardhat-toolbox`
- To use your local installation of Hardhat, you need to use `npx` to run it (i.e. `npx hardhat`).

Create A Project

- To create your Hardhat project run `npx hardhat` in your project folder to initialize your project.
- Select `Create an empty hardhat.config.js` with your keyboard and hit enter.

```
$ npx hardhat
888 888                888 888                888
888 888                888 888                888
888 888                888 888                888
888888888888 8888b. 888d888 .d888888 88888b. 8888b. 888888
888 888      "88b 888P" d88" 888 888 "88b "88b 888
888 888 .d8888888 888 888 888 888 .d8888888 888
888 888 888 888 888 Y88b 888 888 888 888 Y88b.
888 888 "Y888888 888 "Y888888 888 888 "Y888888 "Y888

Welcome to Hardhat v2.10.1

✓ What do you want to do? · Create a JavaScript project
✓ Hardhat project root: · Project-Directory
✓ Do you want to add a .gitignore? (Y/n) · y

You need to install these dependencies to run the sample project:
npm WARN config global '--global', '--local' are deprecated. Use '--location=global' instead.
  npm install --save-dev "hardhat@^2.10.1" "@nomicfoundation/hardhat-toolbox@^1.0.1"

Project created

See the README.md file for some example tasks you can run

Give Hardhat a star on Github if you're enjoying it!

https://github.com/NomicFoundation/hardhat
```

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 and an empty `hardhat.config.js` is enough for Hardhat to work. The entirety of your setup is contained in this file.

Create Smart Contract

You can write your own smart contract or download the [BEP20 token smart contract template](#), place it in the `contracts` directory of your project and rename it as `BEP20Token.sol`.

Configure Hardhat for BSC

- Go to `hardhat.config.js`
- Update the config with bsc-network-credentials.

```
require("@nomicfoundation/hardhat-toolbox");

const { mnemonic } = require('./secrets.json');

// This is a sample Hardhat task. To learn how to create your own go to
// https://hardhat.org/guides/create-task.html
task("accounts", "Prints the list of accounts", async () => {
  const accounts = await ethers.getSigners();
```

What is Hardhat

Setting up the development environment

Pre-requisites

Create A Project

Create Smart Contract

Configure Hardhat for BSC

Compile Smart Contract

Deploy Smart Contract on BSC Network

Verify with Hardhat

Install the plugin

Configure the Etherscan plugin in
hardhat.config.js

Verify Command

Conclusion

```

    for (const account of accounts) {
      console.log(account.address);
    }
  });

  // You need to export an object to set up your config
  // Go to https://hardhat.org/config/ to learn more

  /**
   * @type import('hardhat/config').HardhatUserConfig
   */
  module.exports = {
    defaultNetwork: "mainnet",
    networks: {
      localhost: {
        url: "http://127.0.0.1:8545"
      },
      hardhat: {
      },
      testnet: {
        url: "https://data-seed-prebsc-1-s1.binance.org:8545",
        chainId: 97,
        gasPrice: 20000000000,
        accounts: {mnemonic: mnemonic}
      },
      mainnet: {
        url: "https://bsc-dataseed.binance.org/",
        chainId: 56,
        gasPrice: 20000000000,
        accounts: {mnemonic: mnemonic}
      }
    },
    solidity: {
      version: "0.8.9",
      settings: {
        optimizer: {
          enabled: true
        }
      }
    },
    paths: {
      sources: "./contracts",
      tests: "./test",
      cache: "./cache",
      artifacts: "./artifacts"
    },
    mocha: {
      timeout: 20000
    }
  };

```

NOTE

It requires mnemonic to be passed in for Provider, this is the seed phrase for the account you'd like to

Sample secrets.json

```

{
  "mnemonic": "Your_12_Word_MetaMask_Seed_Phase"
}

```

Compile Smart Contract

To compile a Hardhat project, change to the root of the directory where the project is located and then type the following into a terminal:

```
npx hardhat compile
```

Deploy Smart Contract on BSC Network

- Copy and paste the following content into the `scripts/deploy.js` file.

```

async function main() {
  const [deployer] = await ethers.getSigners();

  console.log("Deploying contracts with the account:", deployer.address);

  console.log("Account balance:", (await deployer.getBalance()).toString());

  const Token = await ethers.getContractFactory("BEP20Token"); //Replace with name of your smart contract
  const token = await Token.deploy();

  console.log("Token address:", token.address);
}

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

```

```
});
```

- Run this command in root of the project directory:

```
$ npx hardhat run --network testnet scripts/deploy.js
```

- Sample Output

```
$ npx hardhat run --network testnet scripts/deploy.js
Deploying contracts with the account: 0x27cf2CEAcddce834f1673005Ed1C60efA63c081
Account balance: 100721709119999208161
Token address: 0xbF39886B4F91F5170934191b0d96Dd277147FBB2
```

Remember your address, transaction_hash and other details provided would differ, Above is just to provide an idea of structure.

Congratulations! You have successfully deployed BEP20 Smart Contract. Now you can interact with the Smart Contract.

You can check the deployment status here: <https://bscscan.com/> or <https://testnet.bscscan.com/>

Verify with Hardhat

Hardhat has an Etherscan plugin: [Hardhat Etherscan plugin](#)

Note: Hardhat was previously Buidler.

Install the plugin

```
npm install --save-dev @nomiclabs/hardhat-etherscan
```

Configure the Etherscan plugin in hardhat.config.js

- Step1: Add `require("@nomiclabs/hardhat-etherscan");`
- Step2: Add your Bscscan API key. Register and obtain your API key from <https://bscscan.com/myapikey>.
- Step3: Always remember to set the solidity compiler version to match what was used for deploying the smart contract.

!!! warning Keep your API key as secret and never it commit to version control

```
// hardhat.config.js
const { mnemonic, bscscanApiKey } = require('./secrets.json');

require('@nomiclabs/hardhat-ethers');
require("@nomiclabs/hardhat-etherscan");
/**
 * @type import('hardhat/config').HardhatUserConfig
 */
module.exports = {

  networks: {
    testnet: {
      url: 'https://data-seed-prebsc-1-s1.binance.org:8545',
      accounts: {mnemonic: mnemonic}
    },
    mainnet: {
      url: 'https://bsc-dataseed.binance.org/',
      accounts: {mnemonic: mnemonic}
    }
  },

  etherscan: {
    // Your API key for Etherscan
    // Obtain one at https://bscscan.com/
    apiKey: bscscanApiKey
  },
  solidity: "0.8.9"
};
```

Verify Command

!!! warning Remove any unnecessary contracts and clear the artifacts otherwise these will also be part of the verified contract.

Run the following command:

```
npx buidler verify --network mainnet DEPLOYED_CONTRACT_ADDRESS "Constructor argument 1"
```


- Example

```
$ npx hardhat verify --network testnet 0xbF39886B4F91F5170934191b0d96Dd277147FBB2
Nothing to compile
Compiling 1 file with 0.5.16
Successfully submitted source code for contract
contracts/BEP20Token.sol:BEP20Token at 0xbF39886B4F91F5170934191b0d96Dd277147FBB2
for verification on Etherscan. Waiting for verification result...
```

```
Successfully verified contract BEP20Token on Etherscan.
https://testnet.bscscan.com/address/0xbF39886B4F91F5170934191b0d96Dd277147FBB2#code
```

Conclusion

This tutorial guided you through the basics of creating and deploying a simple smart contract using the Hardhat IDE. It also provides step-by-step guide on how to verify your deployed smart contract. This tutorial uses testnet, however, the exact same instructions and sequence will work on the mainnet as well.

 [Edit this page](#)


Previous
[« Using Truffle](#)

Next
[Using Replit »](#)

Docs

[Getting Started](#)

Community

[BNB Chain Forum](#) 

[Blog](#) 

[GitHub](#) 

[Discord](#) 

[Twitter](#) 

[Telegram](#) 

Copyright © 2023 Build N Build.