

Dashboard

What are Layer 2 blockchains?

Digging into non-EVM blockchains

Setup and integrate ENS (.eth) domains into your dApp

Looking into decentralized Github (Radicle.XYZ)

Testing your smart contracts locally (100x faster than testnets)

Verifying your smart contracts' code on Etherscan

Learning about IPFS - the decentralized file system

Build your own NFT collection and store metadata on IPFS

Building sovereign user-owned data profiles using Ceramic Network

Building a lottery game on-chain using Chainlink's VRF

Indexing your lottery game data using The Graph's indexers

Lesson Type: Quiz

Estimated Time: 45-60 minutes

Current Score: 100%

Testing contracts on a local blockchain node

This tutorial should familiarize you with starting a local blockchain using Hardhat, deploying a sample smart contract to the local blockchain and interacting with that blockchain with Metamask and Remix.

Why local blockchain?

It is very useful to run a local blockchain because testing becomes very fast and efficient. It is only your machine which is running the blockchain and thus consensus is fast and you don't have to wait for other nodes to sync or validate. You can also use many specialized modules specially built for local testing like [Hardhat console.log](#) which helps you to add printing inside your contract.

🤔 Why is testing on local blockchain faster?

Because only your computer is running the node so consensus happens faster

Because the code is more efficient when run locally

Build

To build the smart contract we would be using [Hardhat](#). Hardhat is an Ethereum development environment and framework designed for full stack development in Solidity. In simple words you can write your smart contract, deploy them, run tests, and debug your code.

•

To setup a Hardhat project, Open up a terminal and execute these commands

```
npm init --yes
npm install --save-dev hardhat
```

•

In the same directory where you installed Hardhat run:

```
npx hardhat
```

- Select [Create a JavaScript project](#)
- Press enter for the already specified [Hardhat Project root](#)
- Press enter for the question on if you want to add a [.gitignore](#)
- Press enter for [Do you want to install this sample project's dependencies with npm @nomicfoundation/hardhat-toolbox?](#)

Now you have a hardhat project ready to go!

If you are not on mac, please do this extra step and install these libraries as well :)

```
npm install --save-dev @nomicfoundation/hardhat-toolbox
```

and press [enter](#) for all the questions.

The following code is a simple smart contract. We'll be using this smart contract in our example.

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

import "hardhat/console.sol";

contract Greeter {
    string private greeting;

    constructor(string memory _greeting) {
        console.log("Deploying a Greeter with greeting:", _greeting);
        greeting = _greeting;
    }

    function greet() public view returns (string memory) {
        return greeting;
    }

    function setGreeting(string memory _greeting) public {
        console.log("Changing greeting from '%s' to '%s'", greeting, _greeting);
        greeting = _greeting;
    }
}
```

This contract declares a string - `greeting`. There are also two methods and a constructor. The constructor initiates the greeting variable with the provided string value.

The `greet` method returns the greeting string. Since this is a `view` function, it costs no gas, and requires no signing to execute.

The `setGreeting` method sets the greeting string with a provided user value. Since this updates the smart contract state, it costs gas, and requires signing. One interesting thing to note about the `setGreeting` method is that it uses the Hardhat's `console.log` contract, so we can actually debug and see to what values was `greeting` changed to!

🤔 Which contract did we use for debugging?

hardhat/debug.sol

hardhat/debugging.sol

hardhat/console.sol

Now to actually start running your local blockchain in your terminal pointing to your directory execute this command:

```
npx hardhat node
```

(Keep this terminal running)

This command starts a local blockchain node for you. You should be able to see some accounts which have already been funded by hardhat with 10000 ETH

🤔 Which command did you use to start your local blockchain?

npx hardhat local start

npx hardhat node

npx hardhat local chain

```
Account #19: 0x8626f6940e2eb28930efb4cef49b2d1f2c9c1199 (10000 ETH)
Private Key: 0xdf57089febbacf7ba0bc227dafbffa9fc08a93fdc68e1e42411a14efcf23656e
```

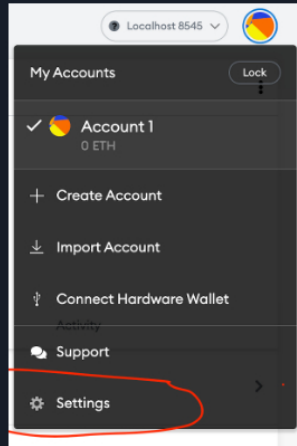
Now, you can continue by deploying the contract to the local blockchain using Hardhat, by running

```
npx hardhat run scripts/sample-script.js
```

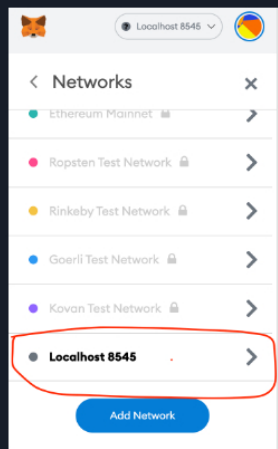
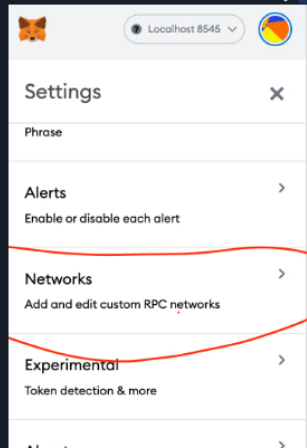
Alternatively, you can also use something like Remix and have it deploy contracts to your local blockchain. The second method will also involve setting up Metamask to work with your local blockchain, and will give you an idea of how to locally test your React/Next.js apps using contracts running on the local blockchain as well, so let's do that.

Metamask Connection

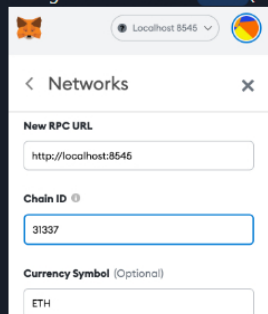
To use metamask to connect to this network, click on your profile and then click on settings



Then click on Networks, followed by [Localhost 8545](#)



Change the Chain ID to [31337](#) (this is the chainId for the local blockchain you are running) and then click [Save](#)



Block Explorer URL (Optional)

Cancel Save

Awesome now your MetaMask has a connection to your local blockchain, we will now add the accounts that Hardhat gave to us. In the node terminal, you should see several accounts displayed. Let's grab one of those:

```
Account #0: 0xf39fd6e51aad88f6f4ce6ab8827279cfff92266 (10000 ETH)
Private Key: 0xac0974bec39a17e36ba4a6b4d238ff944bacb478cbed5efcae784d7bf4f2ff80
```

Go to Metamask --> click on your profile --> Import Account. Select private key in the dropdown and paste the private key from the account you wish. You should now see an account with 10000 ETH (wow!)

🤔 What is the chain id for a local hardhat node?

1337

31337

3133

42069

Remix

We will now deploy our contract to local blockchain and interact with it using Remix

Go to remix.ethereum.org and create a new file inside the contracts folder named `Greeter.sol`

- Copy this code into it:

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

import "hardhat/console.sol";

contract Greeter {
    string private greeting;

    constructor(string memory _greeting) {
        console.log("Deploying a Greeter with greeting:", _greeting);
        greeting = _greeting;
    }


    function greet() public view returns (string memory) {
        return greeting;
    }


    function setGreeting(string memory _greeting) public {
        console.log("Changing greeting from '%s' to '%s'", greeting, _greeting);
        greeting = _greeting;
    }
}
```

This is the same code, we explained above

Compile `Greeter.sol`


SOLIDITY COMPILER

COMPILER 


0.8.7+commit.28d00a7 

☐ Include nightly builds

LANGUAGE


Solidity 

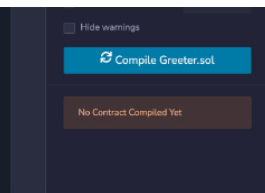
EVM VERSION

compiler default 

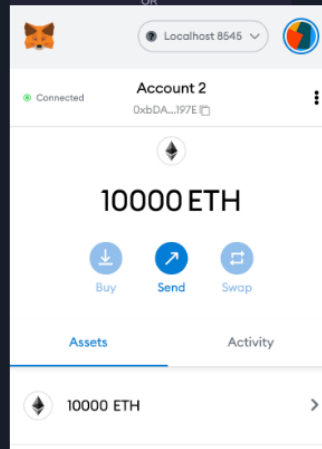
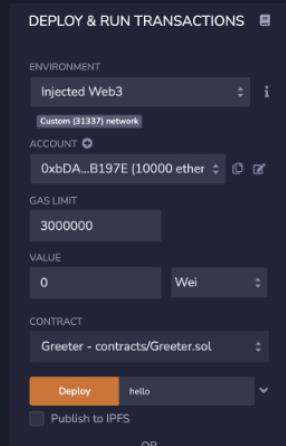
COMPILER CONFIGURATION

☐ Auto compile

☐ Enable optimization `700` 

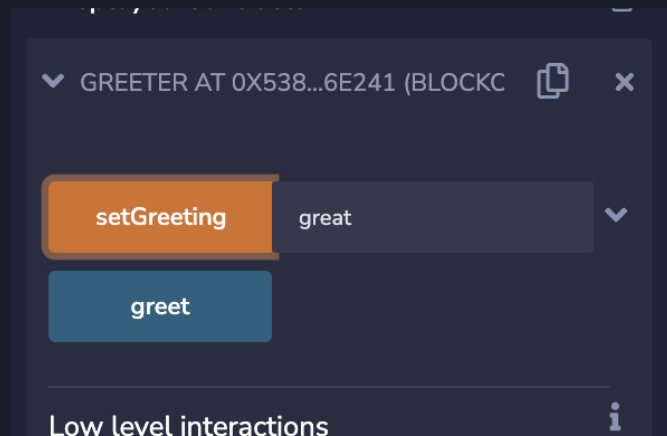


Now to deploy, go to deployment tab and in your environment select **Injected Provider - Metamask**, make sure that the account connected is the one that you imported above and the network is **Localhost 8545** on your MetaMask



Set a greeting and click on deploy. Wait for it to finish. Your contract is now deployed 🎉

Set a greeting and click on **setGreeting**



Check your terminal which was running your hardhat node, it should have the console.log

```
Contract call: <UnrecognizedContract>
Transaction: 0xb40efb253dff27894e0db1bbac71acf1dc0dcefd9bb2505a77f63a3a0384cccc
From: 0x8626f6940e2eb28930efb4cef49b2d1f2c9c1199
To: 0x73511669fd4de447fed18bb79bafec93ab7f31f
Value: 0 ETH
Gas used: 35342 of 35618
Block #2: 0x339446d48bbfc8d3f7de60a033c56384917abaf96ce821f8bd21f0a7f457a086
```

```
console.log:  
Changing greeting from 'hello' to 'great' ←
```

🤔 Can MetaMask and Remix connect with your local blockchain?

Yes

No

Contributors

🤔 Can you trust a dApp running on someone's local node?

Yes

No

🤔 Can you use Hardhat's console.sol library on regular Ethereum networks? Select the best option

No it only works with the local node

Yes we can use it anywhere

No it will give a compilation error

Yes but it is much harder to read those logs

Submit Quiz

0 / 7

Submit



© 2022 LearnWeb3 DAO.



Product

[Dashboard](#)

[Courses](#)

[Community](#)

[Blog](#)

Company

[About](#)

[Work With Us](#)

[We're Hiring](#)

[Buy us a coffee](#)

Stay up to date

Your email address

