Deploying Your First Contract on OP Mainnet

This tutorial is meant for developers who are new to OP Mainnet and Solidity. If you're already familiar with Solidity, consider reading theOP Mainnet Differences guide to understand how OP Mainnet differs from Ethereum. OP Mainnet is designed to "just work" with existing Ethereum tooling, so you can use the tools you already know likeHardhat(opens in a new tab) orFoundry(opens in a new tab) just like you would on Ethereum. This tutorial walks you through the process of deploying your first smart contract to OP Mainnet using theRemix(opens in a new tab) in-browser Solidity IDE.

Dependencies

- Firefox or any Chromium-based browser (Chrome, Brave, Edge, etc.)

Wallet Setup

You'll need access to anEthereum wallet(opens in a new tab) if you want to deploy a smart contract. This tutorial usesMetaMask(opens in a new tab) , a popular browser extension wallet, just to get you started. MetaMask is also available as a mobile application, but it's typically easier to use the browser extension for development. If you already have MetaMask installed, you can skip this section.

Browser-based wallets like MetaMask are convenient for development, but they're not the most secure option. You should never store large amounts of ETH or ERC-20 tokens in a browser-based wallet. If you're planning to store a significant amount of ETH or ERC-20 tokens, consider using a hardware wallet instead.

Install the MetaMask extension for your browser

Head over tometamask.io(opens in a new tab) and click the "Download" button. You'll then be given an option to install the MetaMask extension for your browser. If using a Chromium-based browser, you may be prompted to install the MetaMask extension from the Chrome Web Store. An onboarding page will pop up once the extension is installed.

Continue to create a new wallet

MetaMask will ask you to accept their terms of service. Read them carefully, then tick the box to accept them. Once you've accepted the terms of service, you can create a new wallet by clicking the "Create a new wallet" button.

Decide if you want to share usage data

After starting the wallet creation process you'll be asked if you'd like to share usage data with MetaMask. This is entirely up to you. Select "I agree" or "No thanks" depending on your preference.

Create a password

You'll be asked to create a password for your wallet. This password will be used to encrypt your wallet, so make sure it's a strong one that you won't forget.

You'll be asked to confirm that you understand MetaMask cannot recover your password if you lose it. You may want to store this password in a password manager. Remember, browser-based wallets are great for development but are not the most secure option. Consider looking into a hardware wallet if you're planning to store a significant amount of ETH or ERC-20 tokens.

Once you've written down your password, click "Create a new wallet" to continue.

Watch the short security video

MetaMask will now show you a short video about wallet security. If this is your first time using MetaMask or a browser-based wallet, it's recommended that you watch the video.

Secure your wallet

You'll be presented with the option to "Secure my wallet (recommended)" or "Remind me later (not recommended)". It's strongly recommended to continue with the "Secure my wallet (recommended)" option. This will prompt you to back up your wallet and might prevent you from losing access to your wallet down the line.

Write down your secret recovery phrase

After continuing with "Secure my wallet (recommended)", you'll be asked to write down a 12 word secret recovery phrase. This phrase can be used to recover your wallet if you lose your password or the browser extension somehow becomes corrupted. Write these words down somewhere safe and keep them secret. For a development wallet, it's fine to keep the phrase in a password manager. Just don't place too much importance in this wallet.

Confirm your secret recovery phrase

After writing down your secret recovery phrase, you'll be asked to confirm it. This is to make sure you wrote it down correctly. MetaMask will ask you to input a few of the words you wrote down. Make sure to input them in the correct order.

Complete the wallet creation process

You're all done! Read through the final page to get some more tips from MetaMask, then click "Got it!" when you're ready to continue. Want to explore wallets other than MetaMask? There are many different kinds of wallets out there. You can useEthereum.org's "Find a wallet" feature(opens in a new tab) to find one that works for you.

Add OP Sepolia to MetaMask

This tutorial will show you how to deploy a smart contract to OP Sepolia, the testnet for OP Mainnet. Once you know how to deploy a smart contract to OP Sepolia, you'll be able to deploy a smart contract to OP Mainnet in exactly the same way. In order to interact with OP Sepolia using MetaMask, you'll need to add it as a custom network.

Open the OP Sepolia connection link

Clickthis link(opens in a new tab) to open the OP Sepolia connection link. This link will show you the connection details for OP Sepolia and give you the option to add it to MetaMask. Once you're ready to connect to OP Sepolia, click the "connect" button.

Allow the site to add the network

You'll be presented with a popup from MetaMask asking if you'd like to add the network. Click "Approve" to add OP Sepolia to MetaMask.

Switch to the OP Sepolia network

After you approve the network, MetaMask will automatically try to switch to the OP Sepolia network. Click "Switch network" to continue.

Get ETH on OP Sepolia

You'll need some ETH on OP Sepolia to pay for thegas fees(opens in a new tab) associated with deploying a smart contract. You can use theOptimism Superchain Faucet(opens in a new tab) to get some free ETH on OP Sepolia.

Having issues with the Optimism Superchain Faucet? You can try usingother available OP Sepolia faucets instead.

Check Your Wallet Balance

After you get some ETH on OP Sepolia, you can check your wallet balance in MetaMask. Make sure that your balance has updated before continuing. If you don't see any ETH in your wallet, double check that your MetaMask is connected to the OP Sepolia network.

Write Your First Contract

The most popular smart contract development language today isSolidity(opens in a new tab). In this tutorial, you'll be using a browser-based integrated development environment (IDE) calledRemix(opens in a new tab). Remix is a great tool for learning Solidity because it requires minimal setup and runs in your browser.

Open Remix

Head over toremix.ethereum.org(opens in a new tab) to open Remix. You'll be presented with a welcome screen and a popup asking if you'd like to share usage data with Remix. Accept or decline this request depending on your preferences.

Step through the tutorial

Remix has a small built-in tutorial that will walk you through the basics of the IDE. Click the "Next" button to step through the tutorial. Read through the tips to get a sense of the IDE.

Create an empty contract file

Click on the "File" icon in the left sidebar to create a new file. Name the file "MyFirstContract.sol" and hit enter to create it.

Write your first contract

This tutorial will show you how to deploy a simple contract that has a storage variable you can read and write. You'll be able to update this variable with a transaction and then retrieve the updated value. This is just a simple example to get you started. Solidity is a powerful language that can be used to write complex smart contracts, check out theNext Steps section below after you've finished this tutorial for some more advanced examples!

Copy and paste the following code into the file you just created. Remix will detect copy/pasted code and give you a warning about it. This is meant to prevent you from accidentally running malicious code. Always be careful when copy/pasting code from the internet!

// SPDX-License-Identifier: MIT pragma

solidity ^0.8.0;

contract MyFirstContract { string

public message;

function

setMessage (string

memory

_message) public { message = _message; } }

Compile your contract

By default, Remix will automatically compile your contract when you save it. You can also manually compile your contract by clicking the "Solidity Compiler" icon in the left sidebar and then clicking the "Compile" button. It's usually easier to leave automatic compilation enabled. You shouldn't see any compilation errors for this contract.

Deploy Your Contract

Now that you've written your first contract, you can deploy it to OP Sepolia. Deploying contracts with Remix is pretty straightforward.

Open the Deploy tab

Click on the "Deploy & run transactions" icon in the left sidebar to open the Deploy tab (it looks like an Ethereum logo with an arrow pointing to the right). You'll see some deployment options and a list of contracts that are currently compiled. Since you only have one contract, you should see a single contract called "MyFirstContract" in the list.

Change your environment

By default, Remix will try to deploy your contract to a local, in-memory blockchain. This is useful for testing, but you'll need to change your environment to deploy to OP Sepolia for this tutorial. Click on the dropdown underneath the "Environment" heading and select "Injected Provider - MetaMask".

Accept the connection request

Once you've selected the "Injected Provider - MetaMask" option, MetaMask will show you a popup asking if you'd like to connect to Remix. Accept this request by clicking the "Connect" button.

Deploy your contract

You're now ready to deploy your contract! Click the orange "Deploy" button to deploy your contract to OP Sepolia. You'll be presented with another MetaMask popup asking you to confirm the transaction. Click the "Confirm" button to continue.

Wait for your transaction to confirm

OP Sepolia is relatively fast, so transactions should confirm within just a few seconds. Remix will automatically detect when your transaction has confirmed and will show you your newly deployed contract under the "Deployed Contracts" heading within the Deploy tab.

Interact With Your Contract

Now that you've deployed your contract, you can interact with it. Remix makes it easy to interact with your contract by providing a built-in user interface. You can use this interface to call functions on your contract and read its state.

Expand your contract

Click on the arrow next to the name of your contract under the "Deployed Contracts" heading to expand it. You should see a list of functions that you can call on your contract.

Call the setMessage function

Now you'll update the message variable in your contract. Type a message into the input box next to the orange "setMessage" button and click on the "setMessage" button. Just like when you deployed your contract, you'll be presented with a MetaMask popup asking you to confirm the transaction. Click the "Confirm" button to continue.

Wait for your transaction to confirm

Once again, Remix will automatically detect when your transaction has confirmed. You'll see a green checkmark appear in the console at the bottom of the screen.

Read the updated message variable

Click on the "message" button to read the updated message variable. You should see the message you set in the previous step appear underneath the "message" button. Congrats, you've successfully deployed and interacted with your first smart contract on OP Sepolia!

How Your Contract Works

Now that you've deployed your contract, you might be wondering how it works. Let's take a closer look at the code you wrote.

License Identifier

The first line of most Solidity files is the license identifier. This line is used to specify the license under which the code is released. In this case, the code written is released under theMIT license(opens in a new tab) .

```
// SPDX-License-Identifier: MIT
```

Pragma Directive

The next line is apragma directive(opens in a new tab). This line tells the Solidity compiler which version of the Solidity language to use. In this case, the code is written for Solidity version 0.8.0 or higher.

```
pragma

solidity ^0.8.0;
```

Contract Definition

The next line defines a contract calledMyFirstContract . A contract is a collection of code and data that is stored at a specific address on the blockchain. You can think of a contract as a class in an object-oriented programming language. The contract definition is followed by a pair of curly braces that contain the contract's code.

```
contract MyFirstContract {
```

Message Variable

The first thing you'll notice inside the contract definition is a variable calledmessage . This variable is declared as astring , which is a Solidity type that represents a string of characters. Thepublic keyword means that this variable can be read from outside the contract.

```
string

public message;
```

Message Update Function

The next thing you'll notice is a function calledsetMessage . This function takes a single argument called_message of typestring . Thepublic keyword means that this function can be called from outside the contract by anyone. Since this function doesn't have any access control, anyone can update themessage variable.

function

setMessage (string

memory

_message) public { message = _message; }

Next Steps

To learn more about Solidity, check outthis list of resources on the Optimism Help Center(opens in a new tab). If you learn best by reading source code, check outthis annotated code for an ERC-20 token contract(opens in a new tab). You can also check out theSolidity documentation(opens in a new tab) for more information about the language itself.

Now that you know how to deploy a smart contract to OP Sepolia, you can deploy a smart contract to OP Mainnet in exactly the same way. If you're up for a challenge, trycreating a token on Sepolia and OP Sepolia and thenusing the Optimism SDK to bridge some tokens between the two networks.

Overview Communicating Between Chains in Solidity