SMART CONTRACT SERIES

Smart Contract Deployment on Sepolia Testnet using Hardhat

How to deploy smart contract on Sepolia Testnet using hardhat

Chikku George · Follow

Published in BLOCK6 · 4 min read · Jul 11, 2022

In this article context, we will be covering how we can deploy smart contract on Sepolia Testnet using hardhat configuration.

Sepolia Testnet

Sepolia is the latest testnet of Ethereum Blockchain. It is based on the Proof of Work(PoW) consensus mechanism.

Hardhat

Hardhat is a development environment where we can compile, run, deploy, debug and test our smart contracts.

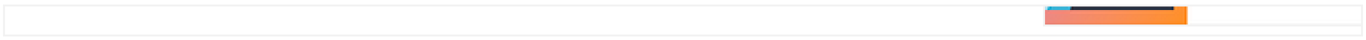
. . .

Step 1: Add Sepolia Testnet to your Metamask and get some fake Sepolia Ethers

Sepolia Testnet

How to add Sepolia Testnet to metamask? How to get some fake Sepolia ethers?

learn.block6.tech



Step 2: Initialize the project

```
npm init
```

It will create a new package.json file which you can edit accordingly.

```
{
 "name": "MySepoliaNFT",
 "version": "1.0.0",
 "description": "",
 "main": "index.js",
 "engines": {
    "node": "12.19.0"
  },
 "scripts": {
   "test": "echo \"Error: no test specified\" && exit 1",
   "start": "nodemon index.js"
 },
 "keywords": [],
 "author": "",
 "license": "ISC"
}
```

Step 3: Write smart contract

Create a folder ***contracts/*** and create a new file called ***MySepolia.sol.***

Make sure to install OpenZeppelin library as we are extending classes from OpenZeppelin Contracts library. ***npm install @openzeppelin/contracts.***

Copy the following code to your smart contract file.

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.8.0 <0.9.0;
import “@openzeppelin/contracts/token/ERC721/ERC721.sol”;
import “@openzeppelin/contracts/utils/Counters.sol”;
import “@openzeppelin/contracts/access/Ownable.sol”;

contract MySepolia is ERC721, Ownable {
 using Counters for Counters.Counter;
 Counters.Counter private _tokenIds;
 using Strings for uint256;

 mapping(uint256 => string) private _tokenURIs;
 string private _baseURIextended;

 constructor() ERC721(“MySepolia”, “MySepolia”) {}

 function setBaseURI(string memory baseURI_) external onlyOwner {
   _baseURIextended = baseURI_;
 }

 function _setTokenURI(uint256 tokenId, string memory _tokenURI)
 internal
 virtual
 {
   require(
    _exists(tokenId),
    “ERC721Metadata: URI set of nonexistent token”
   );
   _tokenURIs[tokenId] = _tokenURI;
 }

 function _baseURI() internal view virtual override returns
(string memory) {
    return _baseURIextended;
 }

 function tokenURI(uint256 tokenId)
 public
 view
 virtual
 override
 returns (string memory)
 {
   require(
   _exists(tokenId),
   “ERC721Metadata: URI query for nonexistent token”
   );

   string memory _tokenURI = _tokenURIs[tokenId];
   string memory base = _baseURI();
   if (bytes(base).length == 0) {
     return _tokenURI;
   }
```

```
    if (bytes(_tokenURI).length > 0) {
      return string(abi.encodePacked(base, _tokenURI));
    }
    return string(abi.encodePacked(base, tokenId.toString()));
  }

  function mintNFT(address recipient, string memory _tokenURI)
  public onlyOwner
  returns (uint256)
  {
    _tokenIds.increment();
    uint256 newItemId = _tokenIds.current();
    _mint(recipient, newItemId);
    _setTokenURI(newItemId, _tokenURI);
    return newItemId;
  }
}
```

Step 4: Install Hardhat & Ethers.js

```
npm install --save-dev hardhat
npm install --save-dev @nomiclabs/hardhat-ethers 'ethers@^5.0.0'
```

Step 5: Create Hardhat project

```
npx hardhat
```

You will get a prompt like below and select “create an empty hardhat.config.js”. It will create an empty hardhat.config.js file in your project folder.

```
888    888                      888 888               888
888    888                      888 888               888
888    888                      888 888               888
8888888888  8888b.  888d888 .d88888 88888b.   8888b.  888888
888    888     "88b 888P"  d88" 888 888 "88b     "88b 888
```

```
888    888 .d888888 888    888  888 888  888 .d888888 888
888    888 888  888 888    Y88b 888 888  888 888  888 Y88b.
888    888 "Y888888 888     "Y88888 888  888 "Y888888
"Y888Welcome to Hardhat v2.8.0? What do you want to do? ...
  Create a basic sample project
  Create an advanced sample project
  Create an advanced sample project that uses TypeScript
> Create an empty hardhat.config.js
  Quit
```

Step 6: Update hardhat.config.js

```
/**
 * @type import(‘hardhat/config’).HardhatUserConfig
 */
require(“dotenv”).config();
require(“@nomiclabs/hardhat-ethers”);
const { API_URL, PRIVATE_KEY } = process.env;
module.exports = {
   solidity: "0.8.9",
   defaultNetwork: "sepolia",
   networks: {
     hardhat: {},
     sepolia: {
      url: API_URL,
      accounts: [`0x${PRIVATE_KEY}`],
     }
   }
};
```

Make sure your .env file contains ***API_URL*** and ***PRIVATE_KEY***. Your .env file should look like below:

```
API_URL = https://rpc.sepolia.online
PRIVATE_KEY = 'Metamask Private Key'

Other RPC URLs which can be used as API_URL
--------------------------
https://rpc.sepolia.dev
https://rpc.sepolia.online
https://www.sepoliarpc.space
```

```
https://rpc.sepolia.org
https://rpc-sepolia.rockx.com
```

Step 7: Compile smart contract

```
npx hardhat compile
```

Step 8: Write deploy script

Create another folder called ***scripts/*** and create a new file called ***deploy.js*** and add the following content to it.

```
async function main() {
  const MySepolia = await ethers.getContractFactory("MySepolia");
  const MySepoliaContract = await MySepolia.deploy();
  console.log("Contract deployed to address:",
MySepoliaContract.address);
}
main().then(() =>
  process.exit(0)
).catch((error) => {
   console.log(error);
   process.exit(1);
});
```

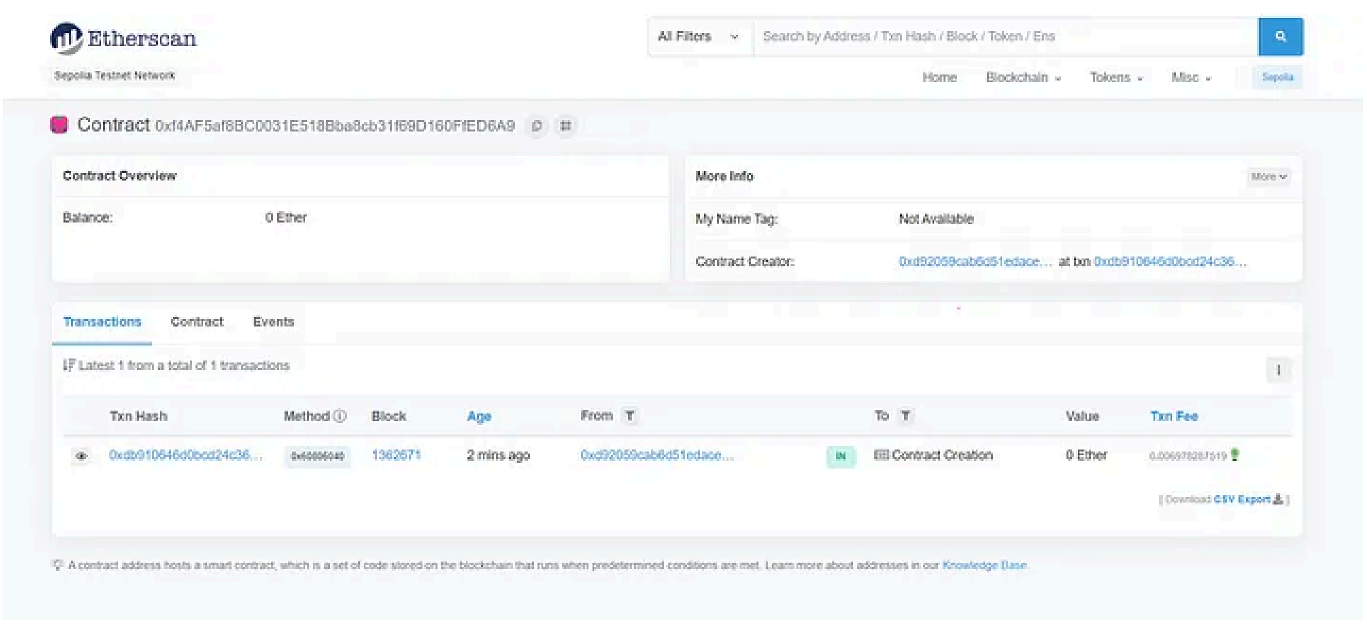
Step 9: Deploy smart contract

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

You will get a console output like below:

```
Contract deployed to address:
0xf4AF5af8BC0031E518Bba8cb31f69D160FfED6A9
```

Now you can verify the deployed contract address on sepolia block explorer. The transaction will look something like below:



Contract 0xf4AF5af8BC0031E518Bba8cb31f69D160FfED6A9

Contract Overview

Balance: 0 Ether

More Info

My Name Tag: Not Available

Contract Creator: 0xd92059cab6d51edace... at txn 0xdb910646d0bcd24c36...

Transactions | Contract | Events

Latest 1 from a total of 1 transactions

	Txn Hash	Method	Block	Age	From		To	Value	Txn Fee
	0xdb910646d0bcd24c36...	0x60806040	1362671	2 mins ago	0xd92059cab6d51edace...	IN	Contract Creation	0 Ether	0.006978287519

[Download CSV Export]

A contract address hosts a smart contract, which is a set of code stored on the blockchain that runs when predetermined conditions are met. Learn more about addresses in our Knowledge Base.

Credit: Author

Great! You have successfully deployed your smart contract on Sepolia Testnet.

. . .

Happy DEPLOYing!

Contents distributed by Learn.Block6.tech

👉 Discord — Live Talks

👉 Twitter — Latest articles

👉 LinkTr.ee

Sepolia Testnet Smart Contract Blockchain Hard Hat Blockchain Ethereum Blockchain

Written by Chikku George

Follow

115 Followers · Writer for BLOCK6

Software Engineer | ReactJs | NodeJs | Blockchain Enthusiast

More from Chikku George and BLOCK6

Chikku George in Coinmonks

How to Interact with Blockchain using Ethers.js

Utilizing ethers, read data and send transactions on the blockchain

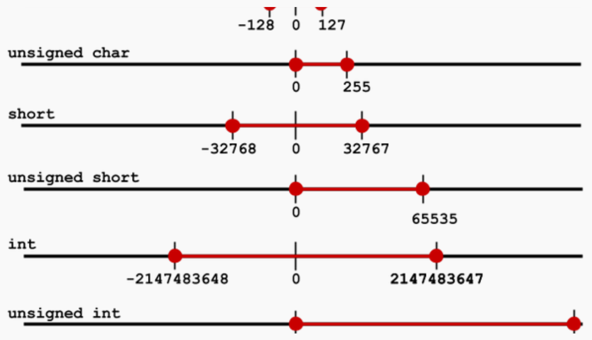
Oct 15, 2022 11

BitcoinShrimps in BLOCK6

How to earn 82% in 2 weeks with this simple trading strategy

Yesterday, I reviewed a strategy which combined two indicators, a day later, I've don...

May 3, 2022 157 3

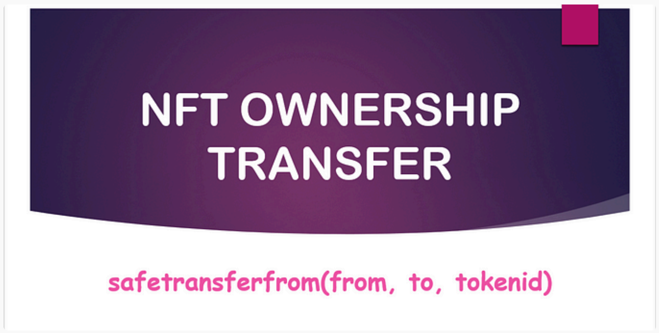


0xPredator in BLOCK6

Solidity Hacking: Integer Overflow and Underflow

Background

May 6, 2022 48



Chikku George in Coinmonks

Transfer Ownership of an NFT

A guide on how to transfer the ownership of an NFT from one wallet address to another wallet

Mar 16, 2022 107

See all from Chikku George

See all from BLOCK6

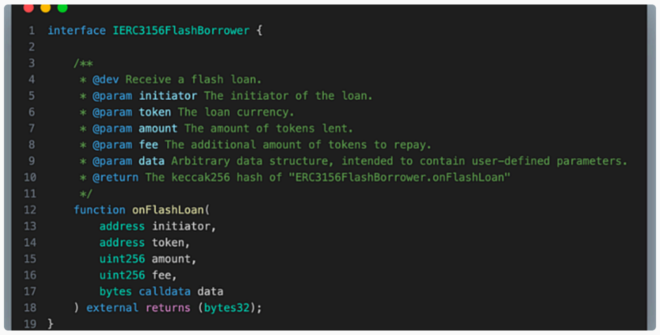
Recommended from Medium

Nate Lapinski

Run Your Own Ethereum Testnet using Anvil and Python

Looking to test your smart contracts or prototype your application but don't want to...

Jun 1 40 1



```
interface IERC3156FlashBorrower {

    /**
     * @dev Receive a flash loan.
     * @param initiator The initiator of the loan.
     * @param token The loan currency.
     * @param amount The amount of tokens lent.
     * @param fee The additional amount of tokens to repay.
     * @param data Arbitrary data structure, intended to contain user-defined parameters.
     * @return The keccak256 hash of "ERC3156FlashBorrower.onFlashLoan"
     */
    function onFlashLoan(
        address initiator,
        address token,
        uint256 amount,
        uint256 fee,
        bytes calldata data
    ) external returns (bytes32);
}
```

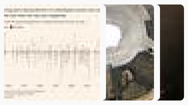
RareSkills in RareSkills

Flash Loans and how to hack them: a walk through of ERC-3156

Flash loans are loans between smart contracts that must be repaid in the same transaction....

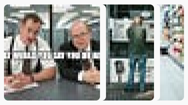
May 27 230 2

Lists



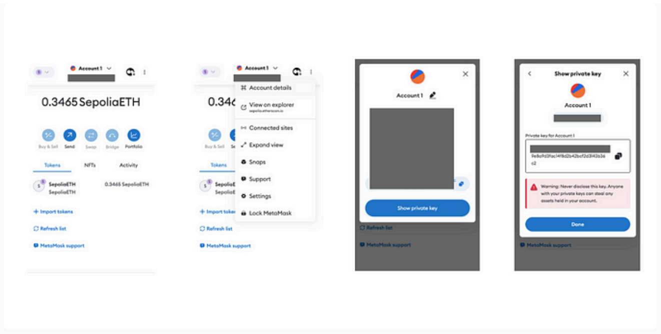
My Kind Of Medium (All-Time Faves)

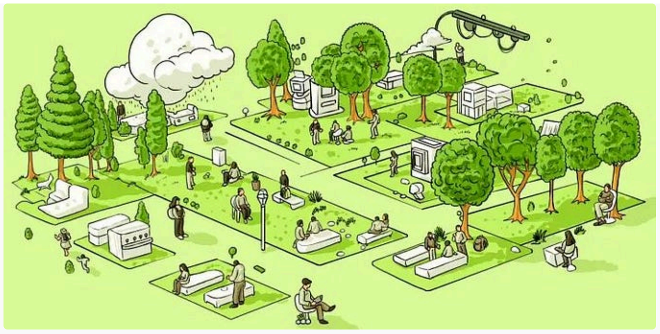
100 stories · 561 saves



MODERN MARKETING

195 stories · 928 saves





Samson JK

How to Verify Smart Contracts Using Hardhat: A Step-by-Step...

Ready to take your smart contract skills to the next level? This guide will show you how to...

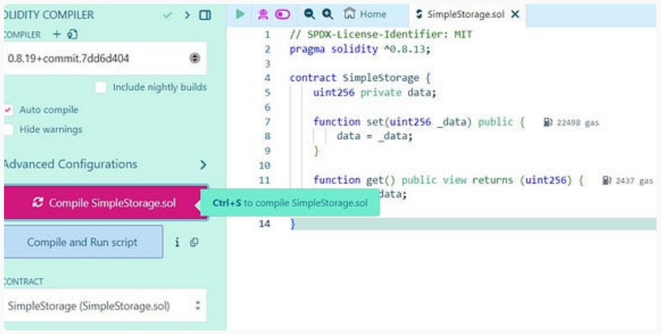
May 24 3 1

Pavlos Giorkas in Digital Currency Traders

Why Grass Deserves a Spot on Your Crypto x AI Portfolio

Remember When I Told You About Grass? Now It's The Most Anticipated Crypto x Ai...

Nov 7 102



```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.13;

contract SimpleStorage {
    uint256 private data;

    function set(uint256 _data) public {      22498 gas
        data = _data;
    }

    function get() public view returns (uint256) {      2437 gas
        data;
}
```

Magda Jankowska

How to Code and Deploy a Smart Contract on the Sepolia Testnet

Introduction

Jun 22 1

Aplha Drops

Title: Deploying a Smart Contract using Hardhat: A Step-by-Step...

Introduction: Hardhat is a popular development environment for Ethereum...

Aug 19

See more recommendations