

# **Internship Report**

## **Blockchain & Crypto Internship**

**Task 02: Create and Deploy an ERC-20 Token on Polygon Amoy**

Submitted by: Vaibhav pratap singh

Intern at Future Interns

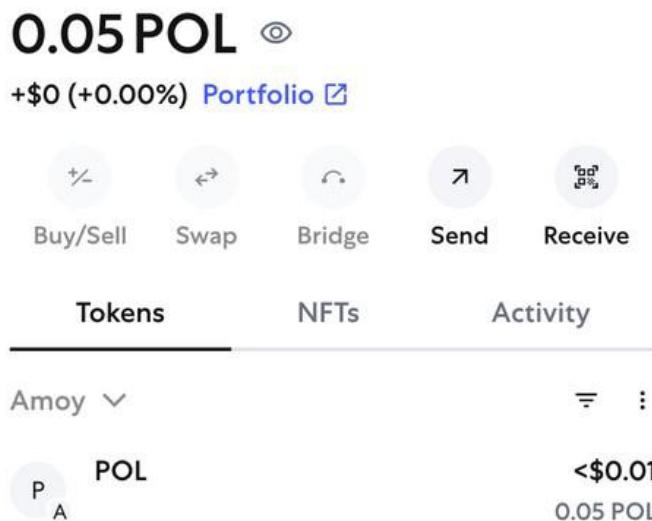
Date: July 26, 2025



## Task Overview:

The goal of Task 02 was to create a custom ERC-20 token using Solidity and deploy it on a public blockchain testnet. Due to the deprecation of the Polygon Mumbai Testnet, this task was completed using the upgraded and actively supported Polygon Amoy Testnet. The token was written in Solidity, compiled and deployed using Remix IDE, and verified via MetaMask and the Amoy explorer.

## Step 1: Setup MetaMask with Amoy Testnet and Fund with 0.05 POL



## Step 2: Write the ERC-20 Token Smart Contract in Remix

The image shows the Remix IDE interface. On the left is a "FILE EXPLORER" sidebar with "WORKSPACES" and "default\_workspace" selected. It lists files like ".deps", "npm", "hardhat", ".states", "vm-prague", "artifacts", "build-info", "PolyCoin\_metadata.json", "PolyCoin.json", "contracts", "Storage.sol", "Owner.sol", "Ballot.sol", "PolyCoin.sol", "scripts", "deploy\_with\_ethers.ts", "deploy\_with\_web3.ts", "ethers-lib.ts", and "web3-lib.ts". The main workspace shows a Solidity code editor with the following content:

```
1 import Web3 from 'web3'
2 import { Contract, ContractSendMethod, Options } from 'web3-eth-contract'
3
4 /**
5  * Deploy the given contract
6  * @param {string} contractName name of the contract to deploy
7  * @param {Array<any>} args list of constructor's parameters
8  * @param {string} from account used to send the transaction
9  * @param {number} gas gas limit
10 * @return {Options} deployed contract
11 */
12 export const deploy = async (contractName: string, args: Array<any>, from?: string, gas?: number): Promise<Options> => {
13
14  const web3 = new Web3(web3Provider)
15  console.log(`Deploying ${contractName}`)
16  // Note that the script needs the ABI which is generated from the compilation artifact.
17  // Make sure contract is compiled and artifacts are generated
18  const artifactsPath = `browser/artifacts/${contractName}.json`
19
20  const metadata = JSON.parse(await remix.call('fileManager', 'getFile', artifactsPath))
21
22  const accounts = await web3.eth.getAccounts()
23
24  const contract: Contract = new web3.eth.Contract(metadata.abi)
25
26  const contractSend: ContractSendMethod = contract.deploy({
27    data: metadata.data.bytecode.object,
28    arguments: args
29  })
```

At the bottom, the footer says "Welcome to Remix 0.68.3" and "Your files are stored in indexedDB, 2.65 KB / 285.5 GB used".

The screenshot shows the Remix IDE interface. On the left is the File Explorer sidebar with various project files listed, including .deps, npm, hardhat, .states, artifacts, contracts, scripts, tests, and configuration files like .prettierrc.json and README.txt. The code editor on the right contains a TypeScript file named ethers-lib.ts. The code is a deployment script for a Ethereum contract:

```
import { ethers } from 'ethers'

/**
 * Deploy the given contract
 * @param {string} contractName name of the contract to deploy
 * @param {Array<any>} args list of constructor' parameters
 * @param {Number} accountIndex account index from the exposed account
 * @return {Contract} deployed contract
*/
export const deploy = async (contractName: string, args: Array<any>, accountIndex?: number): Promise<ethers.Contract> => {
  console.log(`deploying ${contractName}`)
  // Note that the script needs the ABI which is generated from the compilation artifact.
  // Make sure contract is compiled and artifacts are generated
  const artifactsPath = `browser/artifacts/${contractName}.json` // Change this for different path

  const metadata = JSON.parse(await remix.call('fileManager', 'getFile', artifactsPath))
  // 'web3Provider' is a remix global variable object

  const signer = (new ethers.providers.Web3Provider(web3Provider)).getSigner(accountIndex)

  const factory = new ethers.ContractFactory(metadata.abi, metadata.data.bytecode.object, signer)

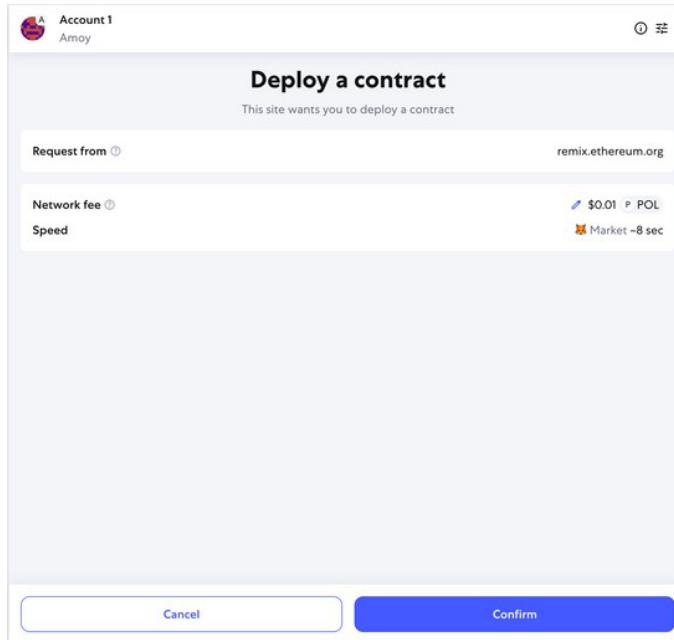
  const contract = await factory.deploy(...args)

  // The contract is NOT deployed yet; we must wait until it is mined
  await contract.deployed()
  return contract
}
```

The Remix interface includes a bottom bar with tabs for different contracts (1\_Storage.sol, 2\_Owner.sol, 3\_Ballot.sol, PolyCoin.sol) and a status bar indicating 0 transactions and a welcome message.

This screenshot is similar to the first one, showing the Remix IDE interface. The File Explorer sidebar is identical. The code editor now displays a different TypeScript file, ethers-lib.ts, which contains the same deployment script as before. The tabs at the top of the code editor are now ts ethers-lib.ts and ts web3-lib.ts. The status bar at the bottom includes a Scam Alert button, an Initialize as git repo button, a Did you know? link, and a RemixAI Copilot (enabled) message.

## Step 3: Confirm Contract Deployment via MetaMask on Polygon Amoy



## Step 4: Use Remix's Injected Web3 to Deploy Contract to Amoy

The image shows the Remix IDE interface. On the left, there's a sidebar with various icons. The main area is titled "DEPLOY & RUN TRANSACTIONS". It includes sections for "ENVIRONMENT" (set to "Remix VM (Prague)"), "ACCOUNT" (showing address 0x5B3...eddC4), "GAS LIMIT" (set to "Estimated Gas" with value 3000000), "VALUE" (set to 0 Wei), "CONTRACT" (selected "PolyCoin - contracts/PolyCoin.sol"), "DEPLOY" (with "initialSupply" set to 2), and "TRANSACTION" (which is empty). On the right, a "Debug" panel shows the transaction details for a successful deployment:

Value	Description
[vm]	from: 0x5B3...eddC4 to: PolyCoin.(constructor) value: 0 wei data: 0x608...0000d logs: 1 hash: 0x8af...a2cdf
status	0x1 Transaction mined and execution succeed
transaction hash	0xaefb93bd36e72bb4c64f3ee3bdc271adcd4ec95a11fd5956ef5ac47f3d0a2cdf
block hash	0x7ae9cc4a564dd5a29845e4ac2315d2d0100d555c695c4f277f12c210733cf3c
block number	2
contract address	0xd8b934580fcE35a11B58C6D73a0eE468a2833fa8
from	0x5B30a6a701c568545dFc803FcB875f56beddC4
to	PolyCoin.(constructor)
gas	1083085 gas
transaction cost	941813 gas
execution cost	804065 gas

## Step 5: View Deployed Contract on Amoy Block Explorer

The screenshot shows the Amoy Block Explorer interface. At the top, it displays a contract address: **Contract 0xA856dCd63aebd6526f2e7e659e8756582faE8489**. Below this, there are two main sections: **Overview** and **Contract info**.

**Overview:** Shows POL holdings (0 POL), Token holdings (0 tokens), and NFT holdings (0 NFTs).

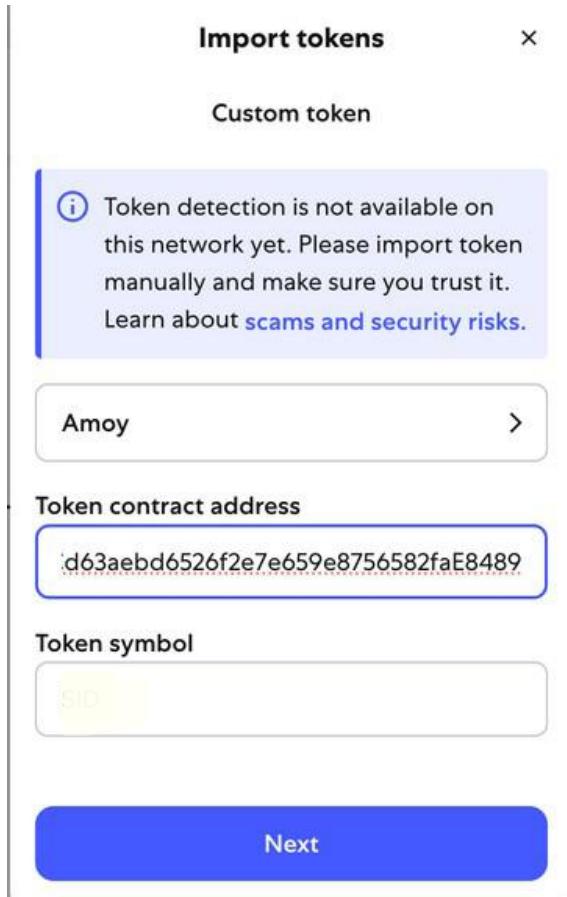
**Contract info:** Shows the contract creator (**0x3972f31c4532ea67216b720710bcd7630e59e790**) and the corresponding token (**SID (SiddToken)**).

Below these sections, there are tabs for **Transactions**, **Contracts**, and **Events**. The **Transactions** tab is selected, showing a single record:

Txn hash	Method	Block	Date time	From	To	Amount	Txn fee
0x5ddd31b82c5...	sanleo	23212263	06/26/2025, 09:53:11	0x3972...0e59e790	In 0xa856...ae8489	0 POL	0.02943144 POL

At the bottom, there are filters for **Start date**, **End date**, **From/To**, **Address**, **Amount**, **Method**, **Txn status: All**, **Txn type: All**, and a toggle for **Hide zero-amount txns**.

## Step 6: Manually Import the Token into MetaMask



## Step 7: Verify Token is Added with Correct Balance

The screenshot shows the MetaMask wallet interface. At the top, it says "Your balance". Below that, there's a list of tokens: SID with a balance of \$0.00 and 1,000,000 SID. Under "Token details", it shows the network as Amoy, contract address as 0xA856d.E8489, token decimal as 8, and spending caps with a "Copy to clipboard" button and a link to "Edit in Portfolio".

Network	A Amoy
Contract address	0xA856d.E8489
Token decimal	8
Spending caps	<a href="#">Edit in Portfolio</a>

### SolidityCode:

```
// SPDX-License-Identifier: MIT pragma
solidity ^0.8.0;
import "@openzeppelin/contracts/token/ERC20/ERC20.sol"; contract
MyToken is ERC20 {
    constructor() ERC20("VaiCoin", "VAION") {
        _mint(msg.sender, 1000000 * 10 ** decimals());
    }
}
```

### Learnings

- Mastered ERC-20 token deployment via OpenZeppelin and Remix IDE.
- Understood MetaMask interaction with custom networks.
- Became aware of Polygon Mumbai deprecation and Amoy migration.
- Gained confidence working with smart contracts and testnets.