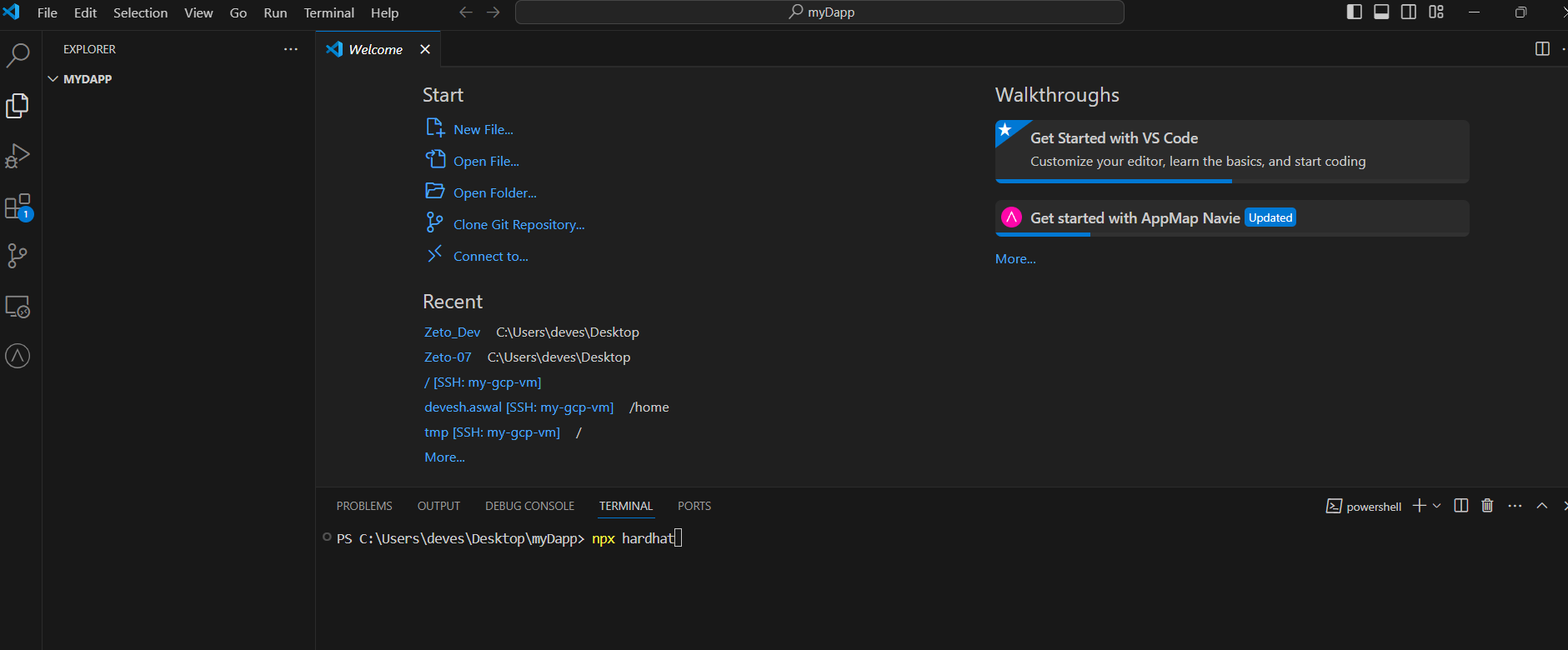
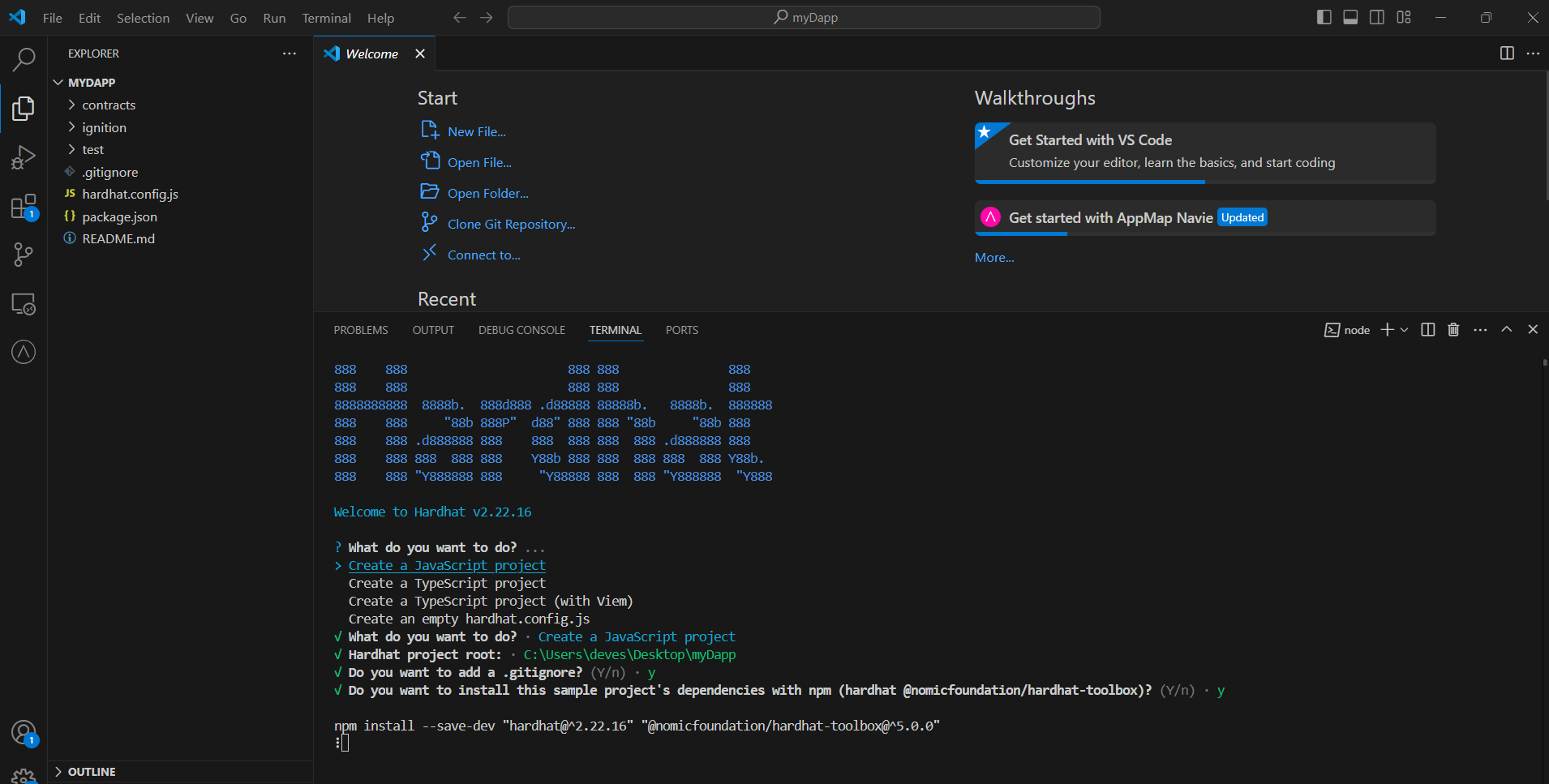
**Type npx hardhat or npx hardhat init in terminal**

****

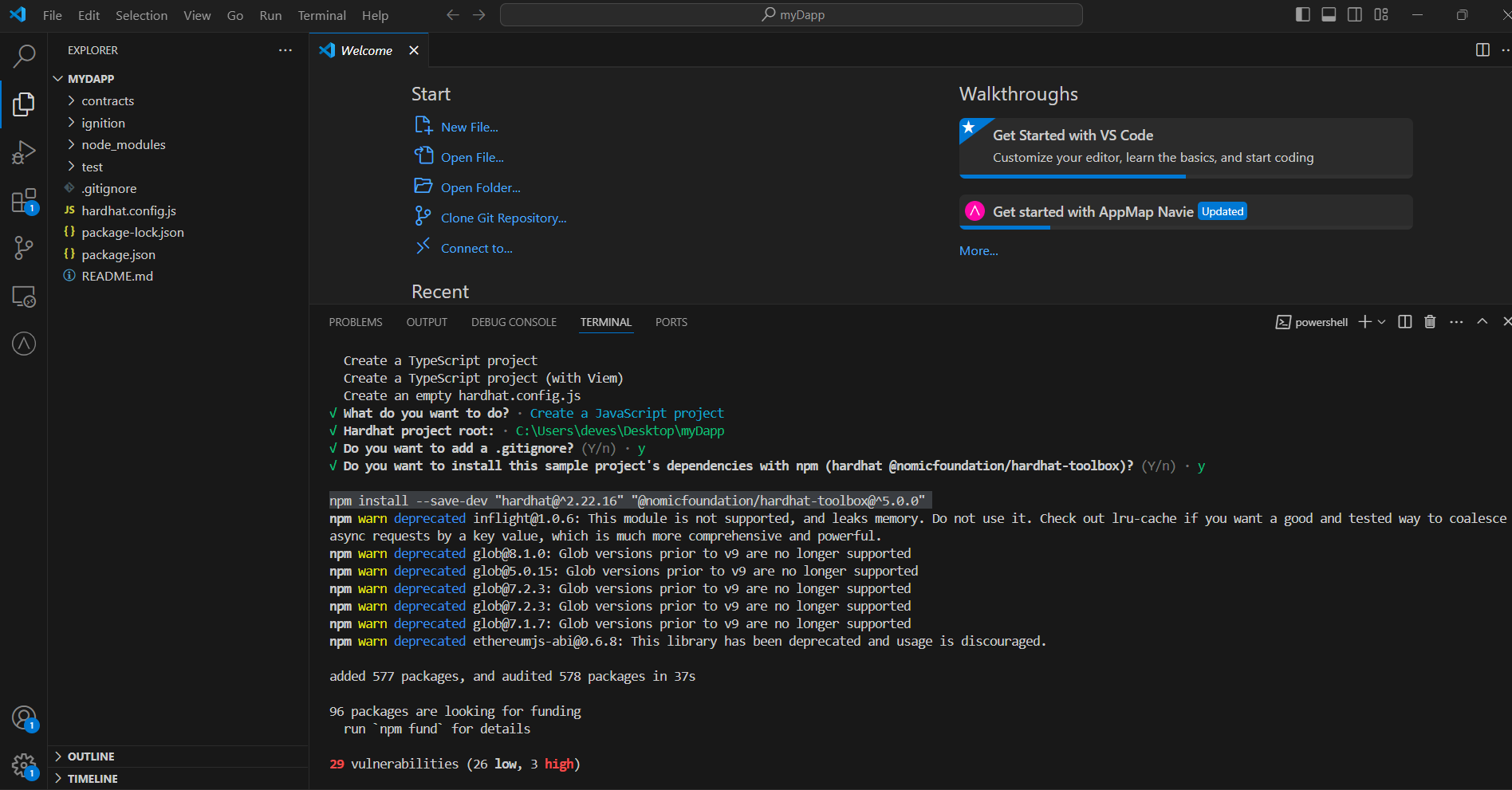
**Select below options**

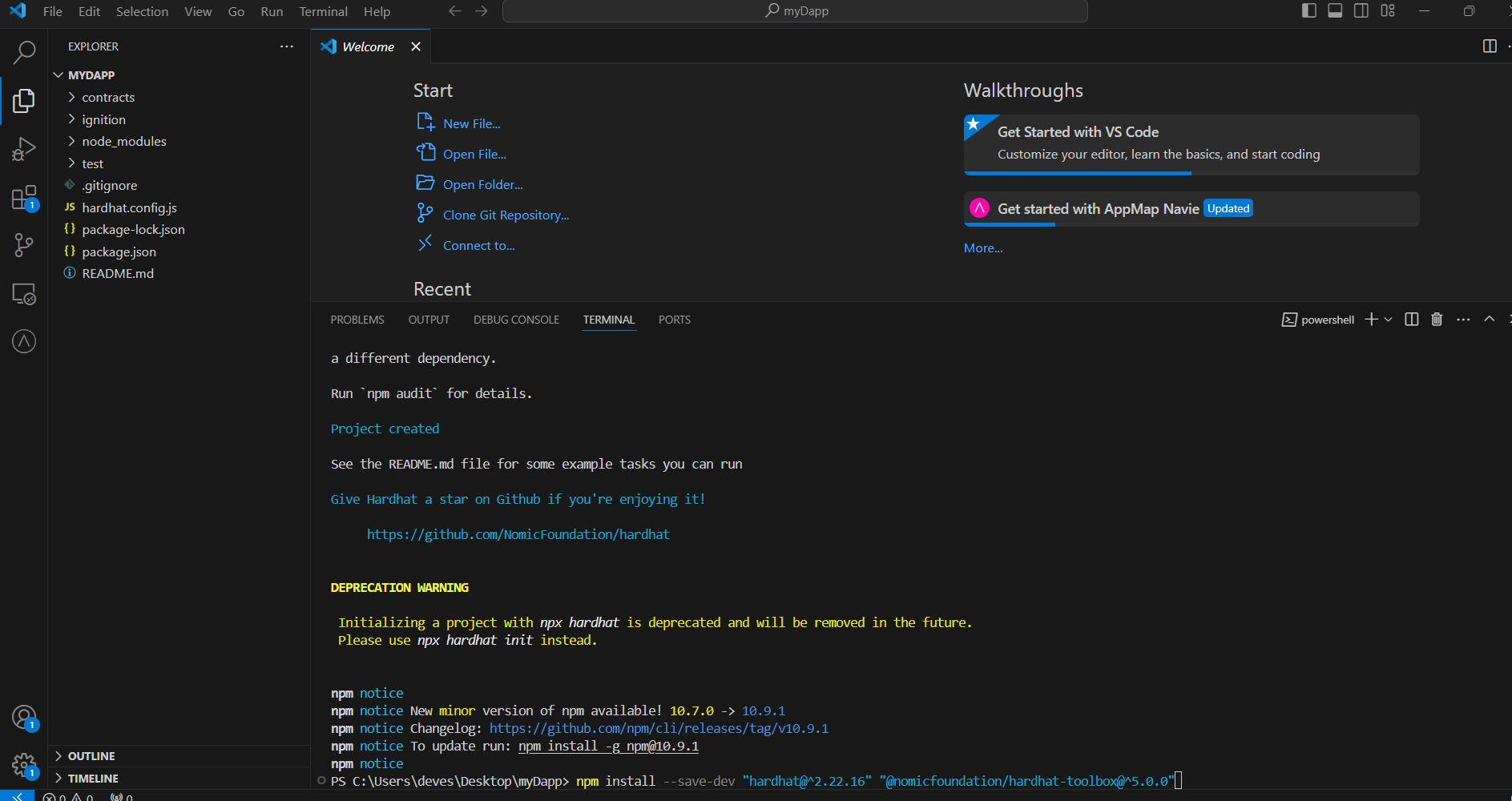
****

**All the related files will be created.**

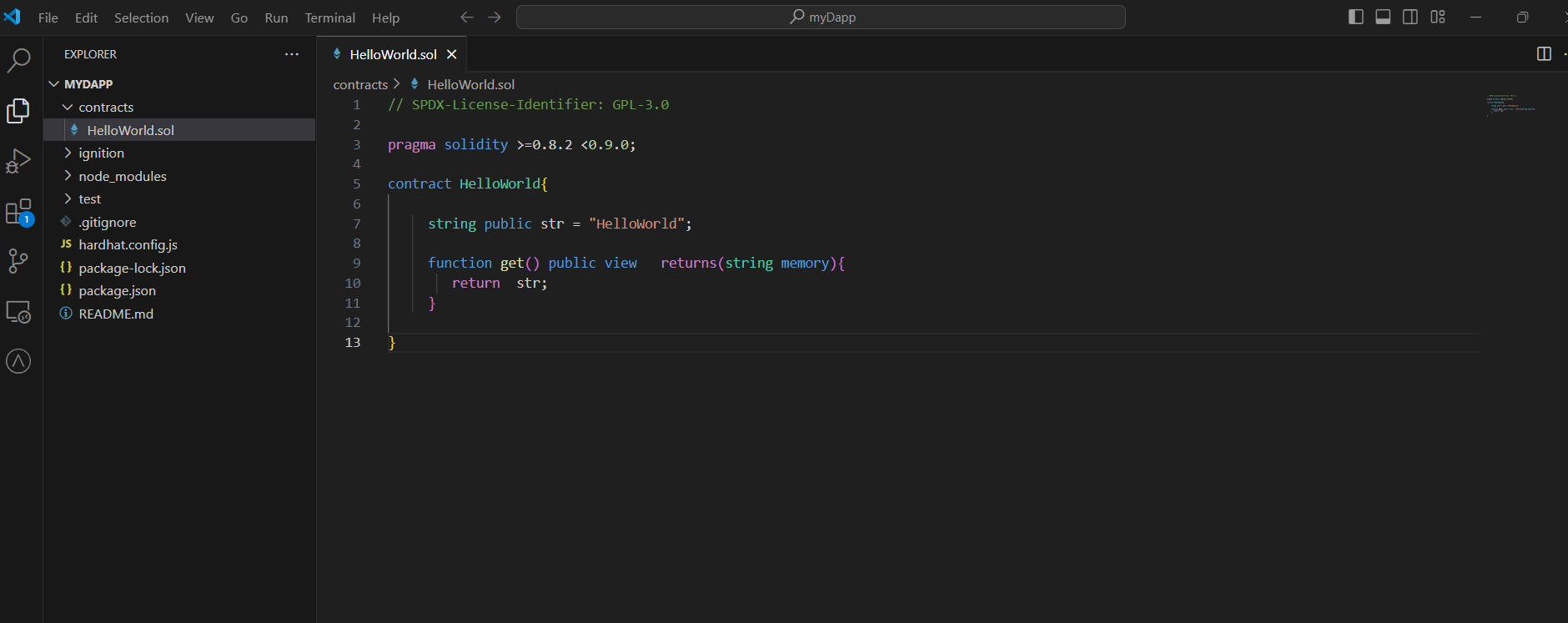
****

**Install extra dependency which you got from previous command**

****

****

**Now under contracts section you will find lock.sol ,it’s a sample smart contract. Change the name and code of it as below.**

****

**And under script folder modify deploy.js. Change smart contract name accordingly in script. If script folder is not available, please create manually.**

****

**const hre = require("hardhat");**

**async function main() {**

**const HelloWorld = await hre.ethers.getContractFactory("HelloWorld"); //fetching bytecode and ABI**

**const helloworld = await HelloWorld.deploy(); //creating an instance of our smart contract**

**await helloworld.waitForDeployment();//deploying your smart contract**

**console.log("Deployed contract address:",`${await helloworld.getAddress()}`);**

**}**

**// We recommend this pattern to be able to use async/await everywhere**

**// and properly handle errors.**

**main().catch((error) => {**

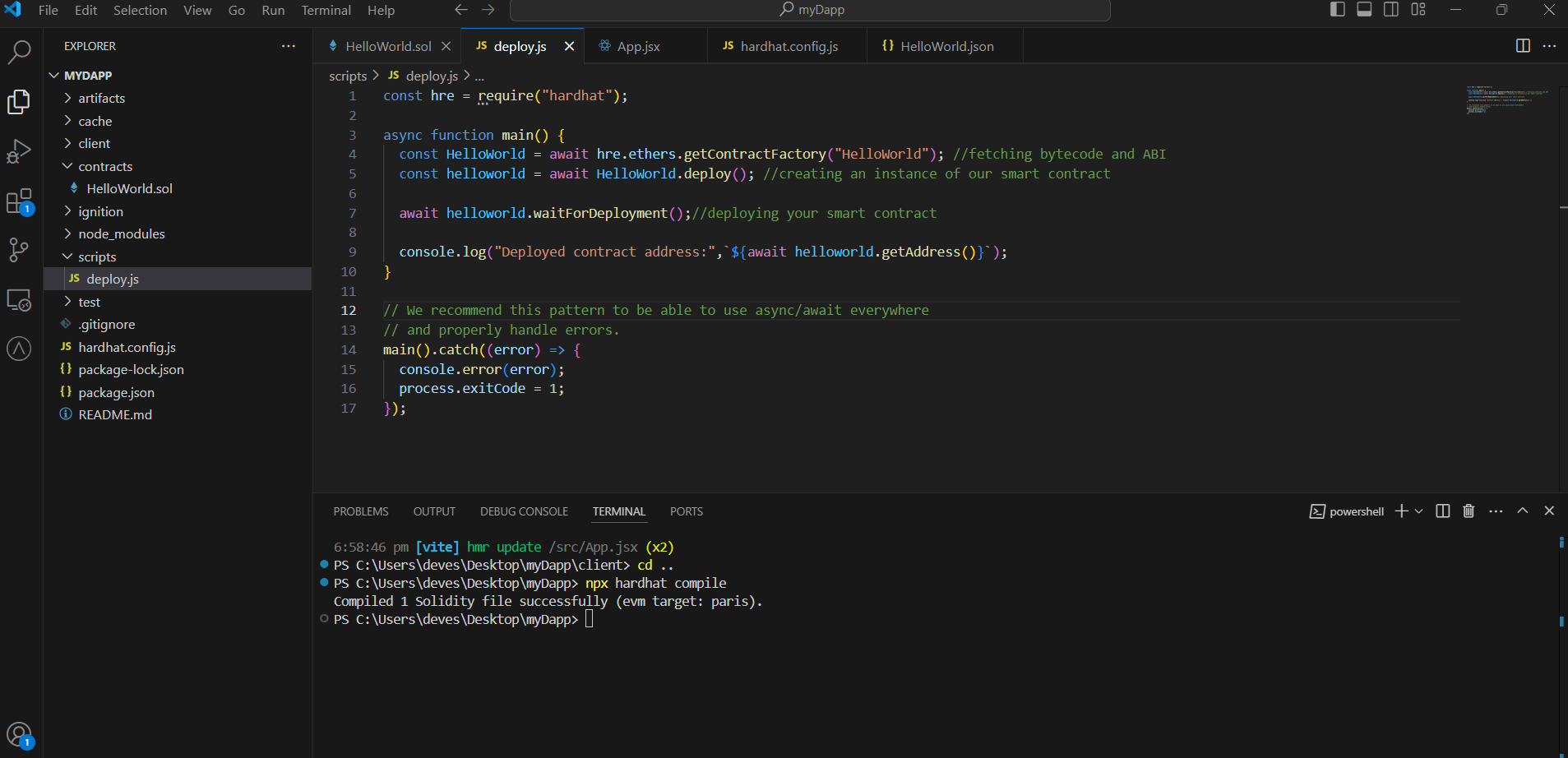
**console.error(error);**

**process.exitCode = 1;**

**});**

**Compile smart contract**

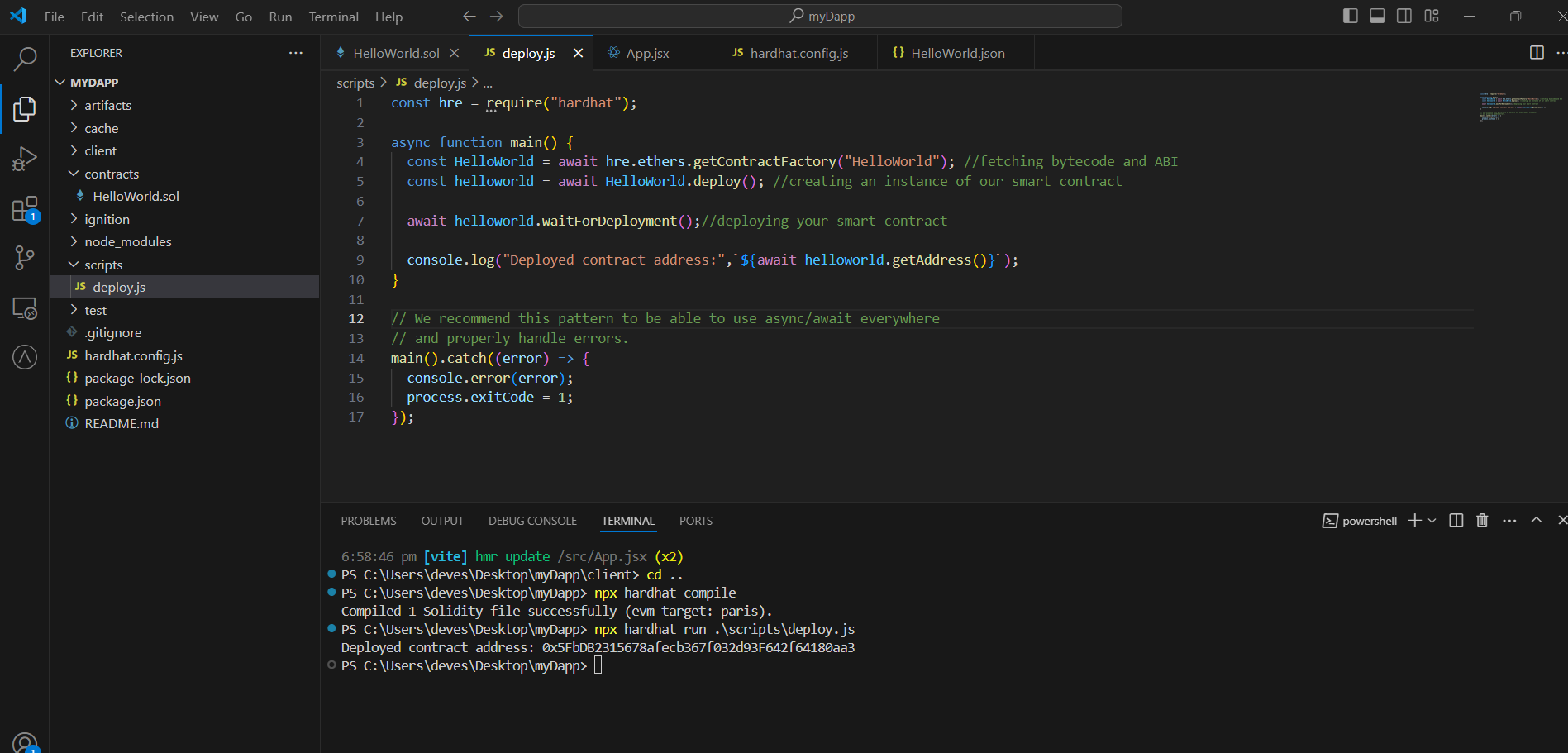
**npx hardhat compile**

****

**You will observe artifacts folder is created. It will have Json file which contain bytecode and abi.**

**Now we need to run the deploy script to deploy contract on blockchain. To deploy run below command**

**npx hardhat run scripts/deploy.js**

****

**Contract is deployed. Please make a note of contract address**

**0x5FbDB2315678afecb367f032d93F642f64180aa3**

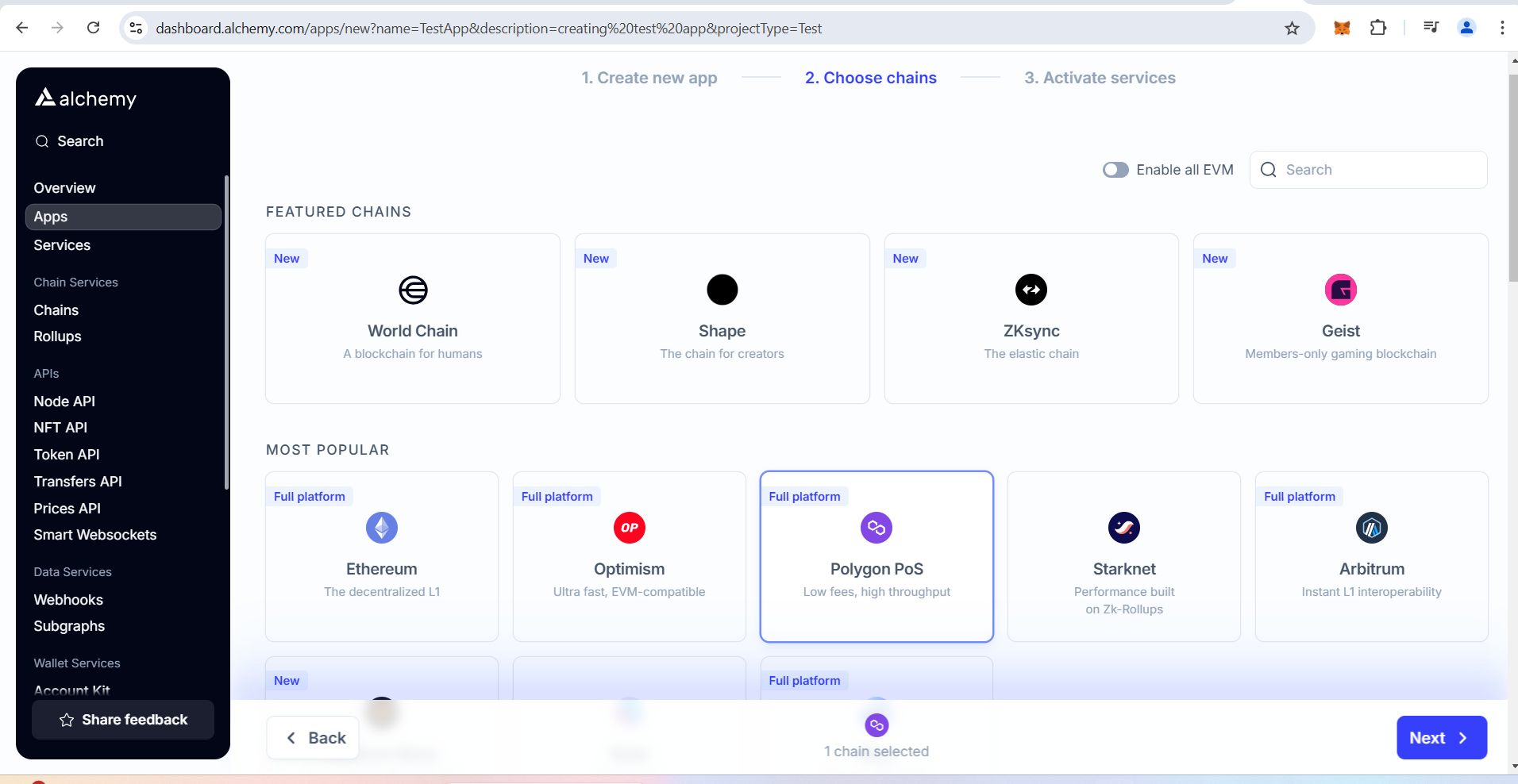
**Currently we have deployed our smart contract in local hardhat blockchain. Now we will deploy our smart contract in test net.**

**Go to Alchemy** [**https://www.alchemy.com/**](https://www.alchemy.com/) **and create your profile. Now click on create app.**

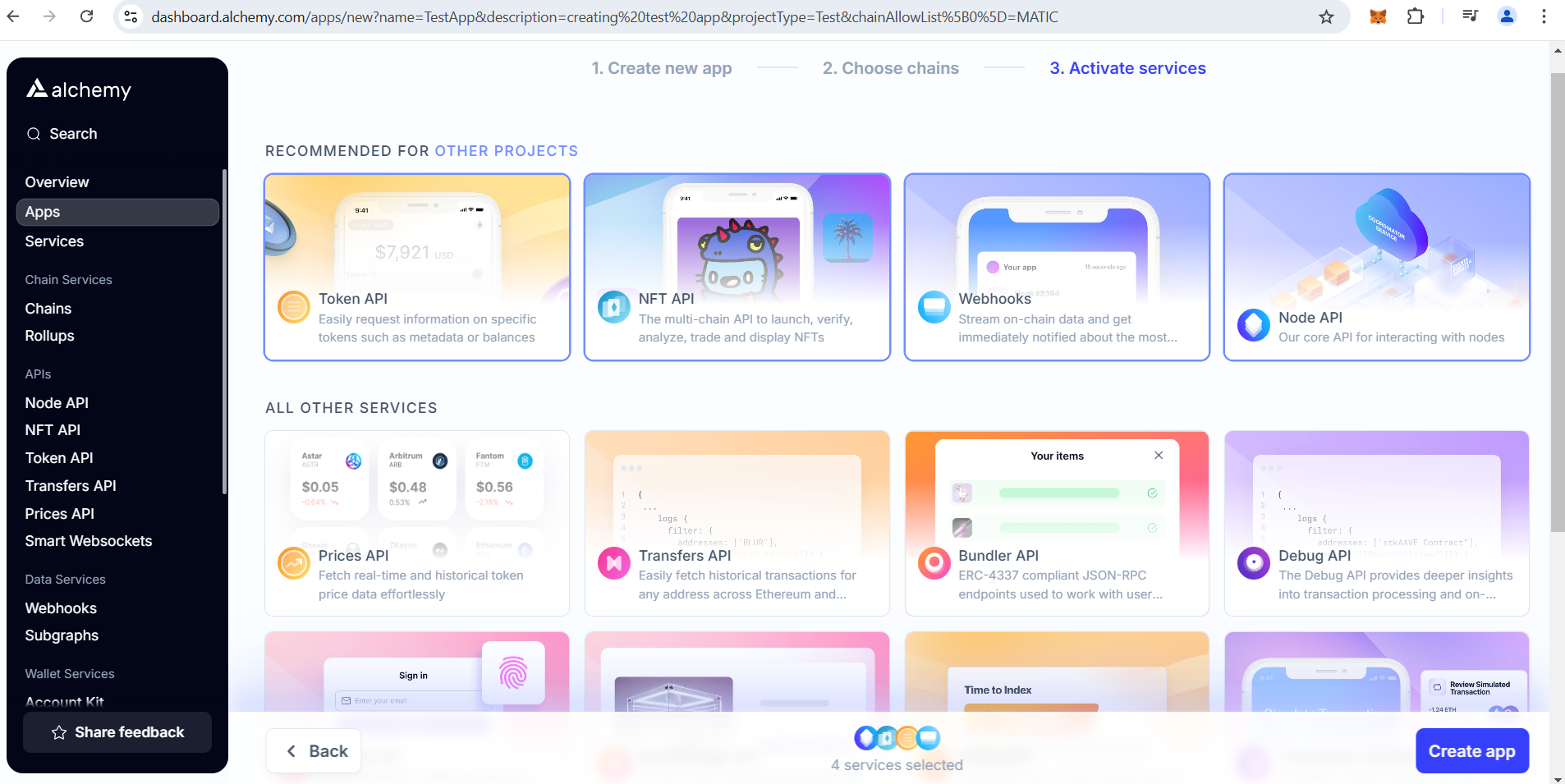
**Fill the required information**

****

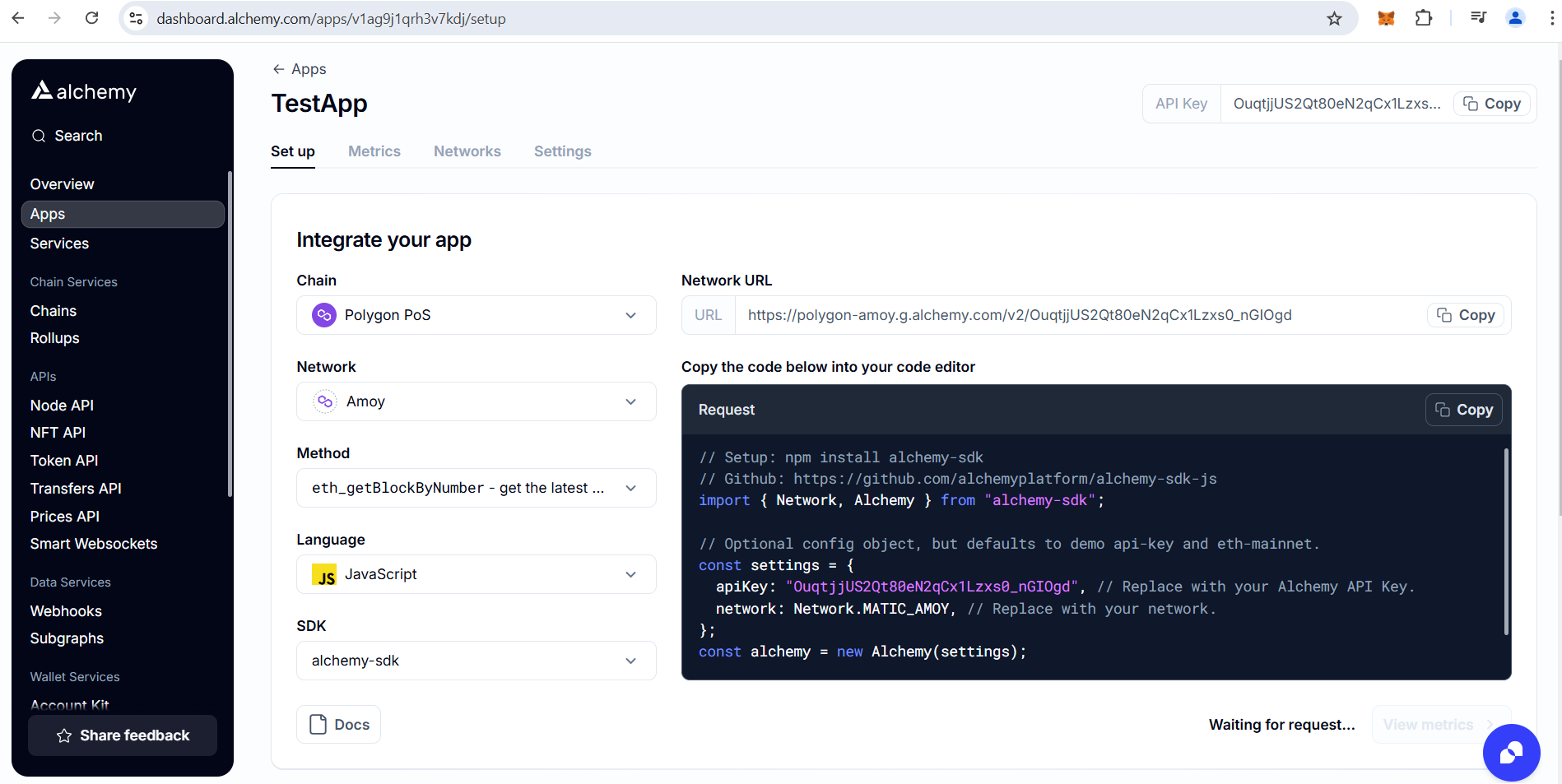
**Select the blockchain whose node you want to host. I have selected polygon.**

****

**Click on create app**

****

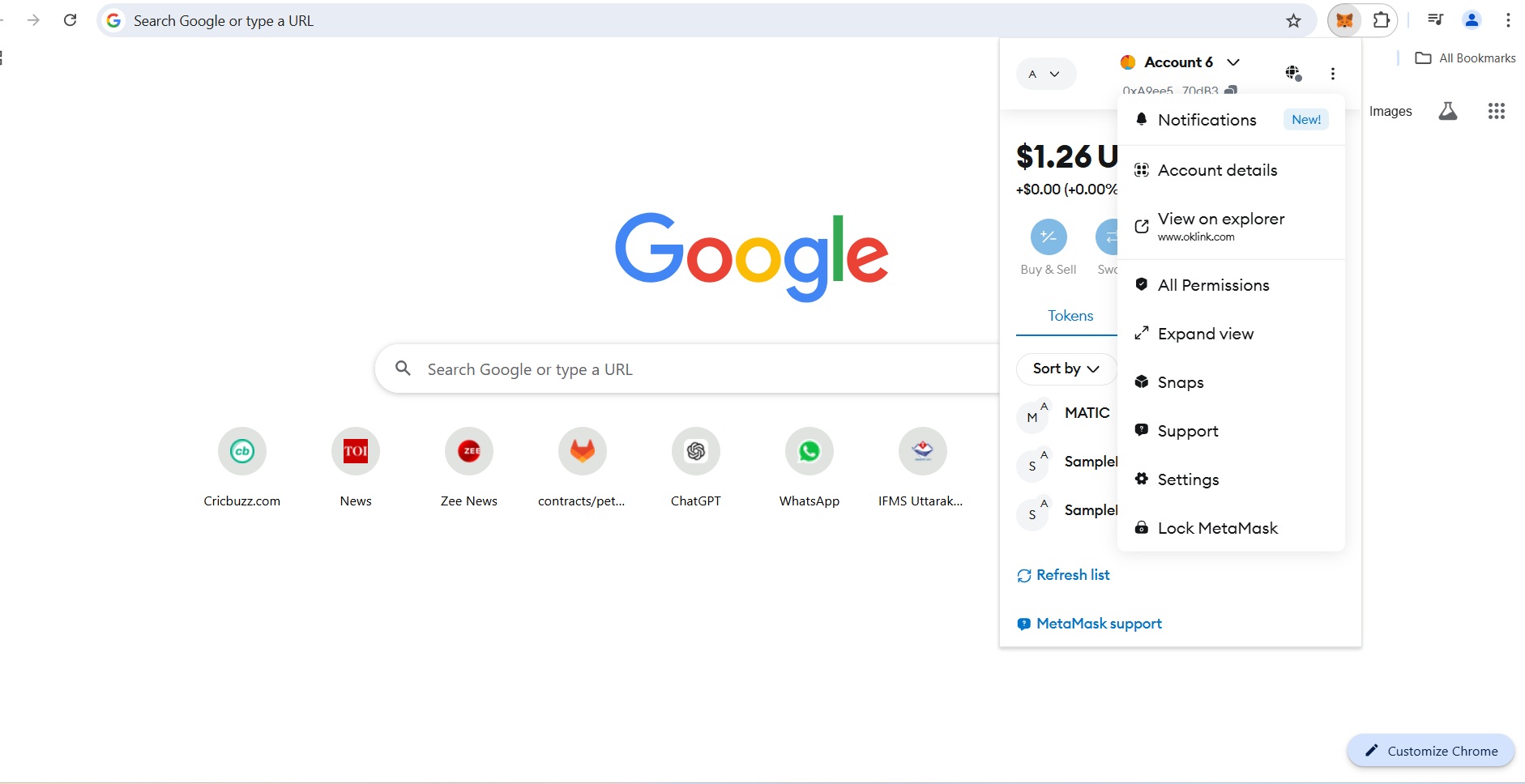
**Change network to amoy from mainnet**

****

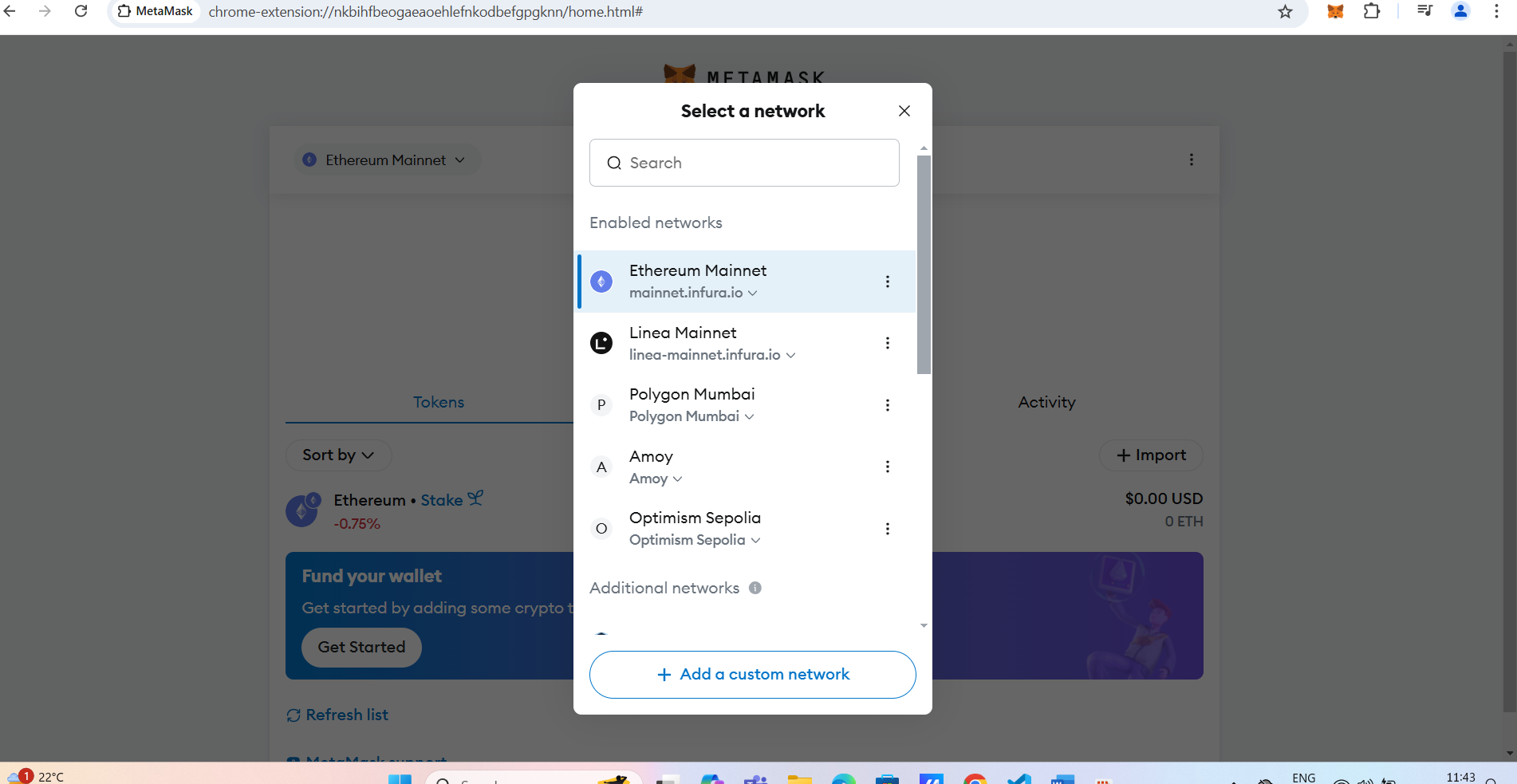
**Go to networks tab and copy url**

****

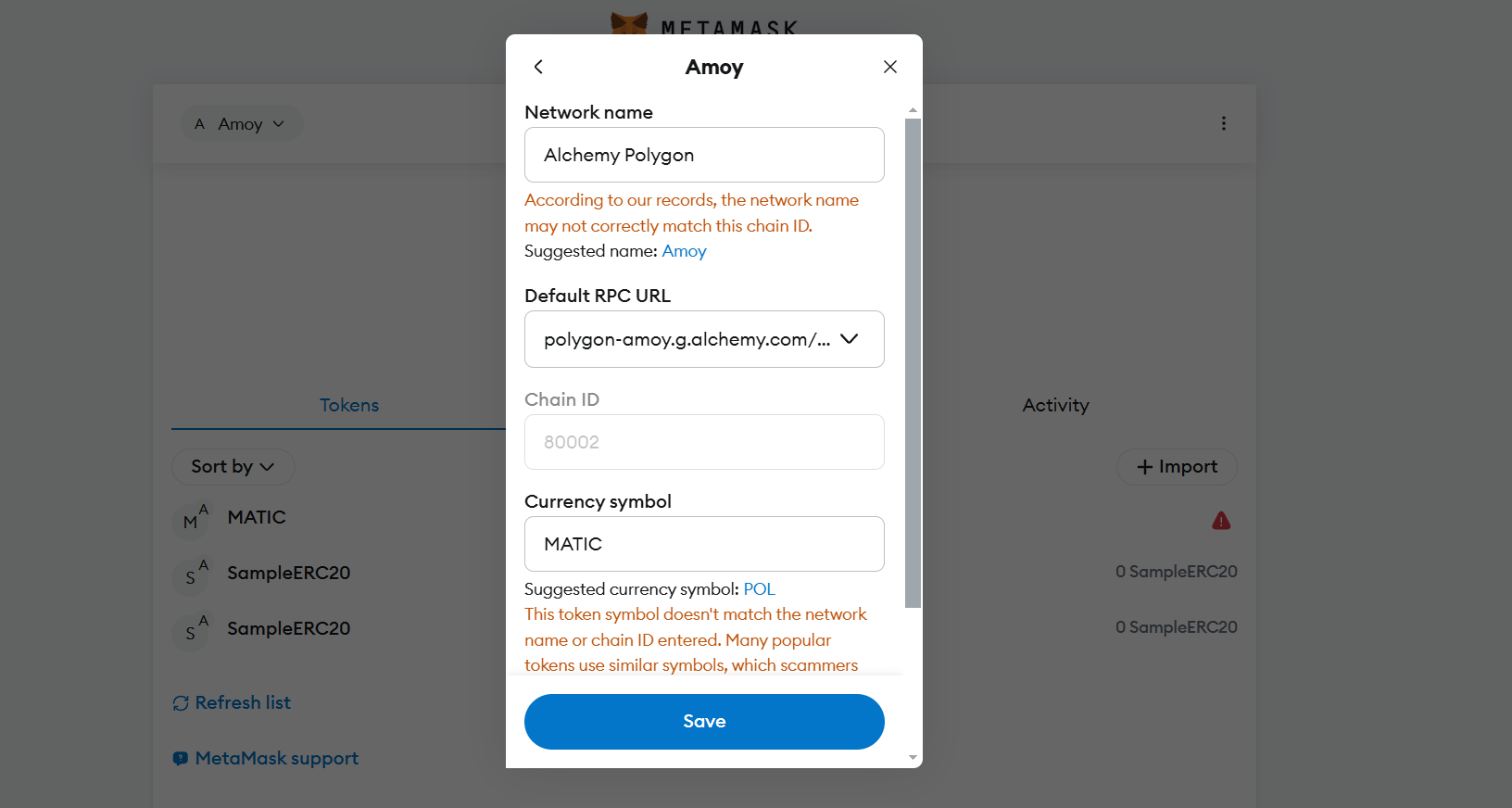
**Now go to metamask and open it in expanded view for better experience. If you don’t have please add it as chrome extension.**

****

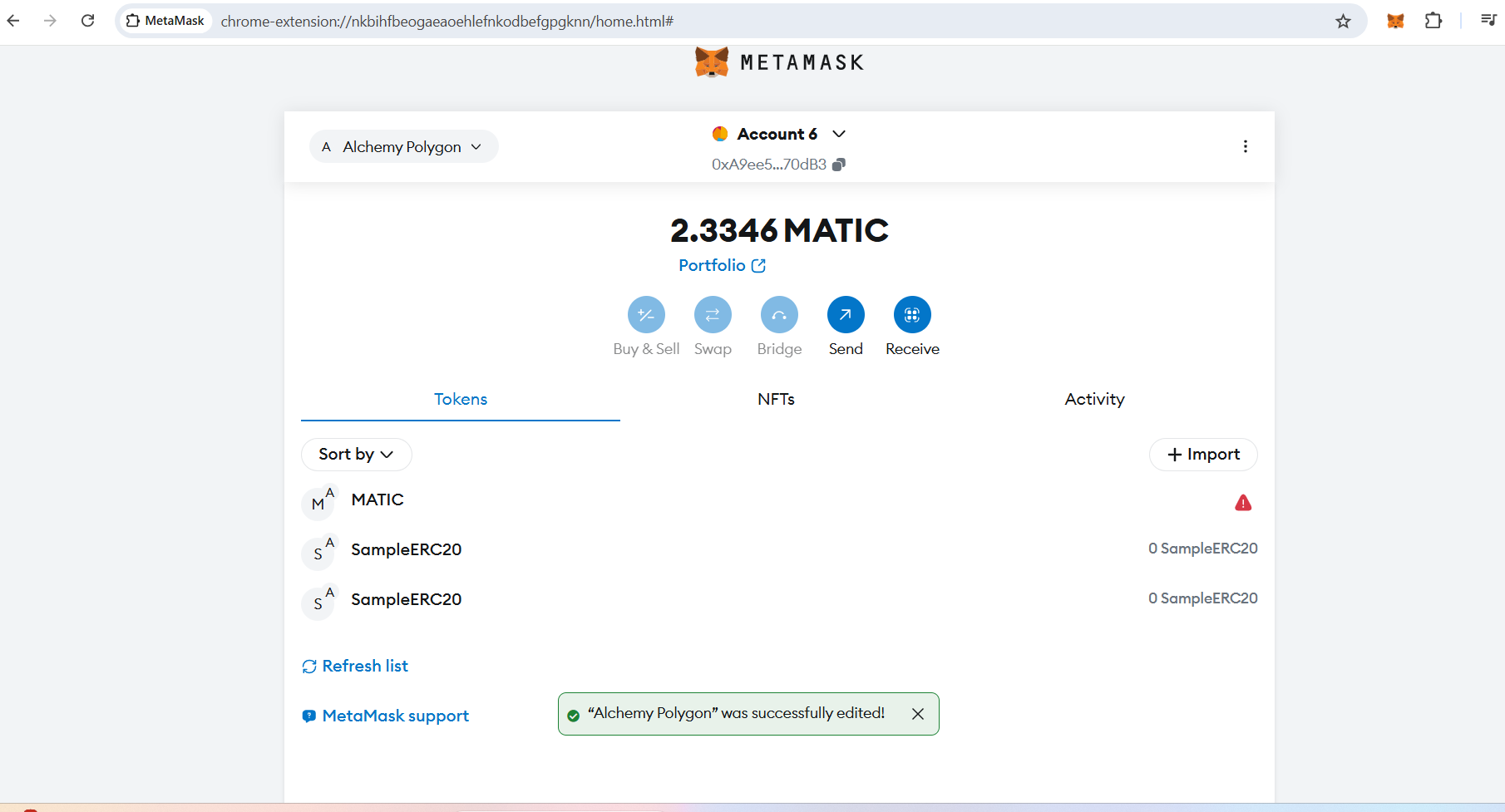
**Now click on the networks dropdown. Click on add a custom network**

****

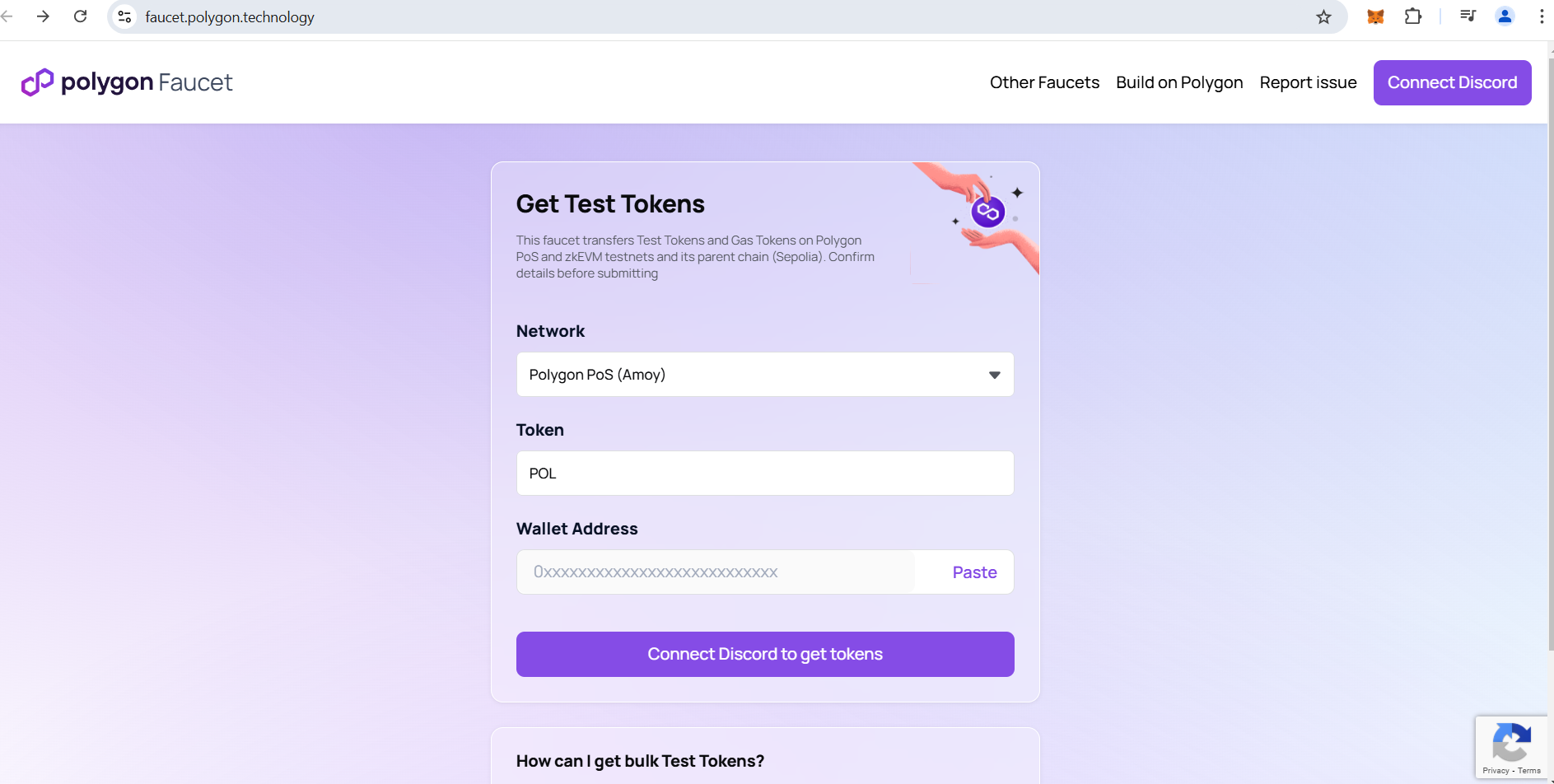
**Now fill the details. You can provide any name of your choice. RPC url you will get it from Alchemy under networks tab. Chain Id anf currency symbol will be auto populate if RPC url is correct.**

****

**Click save and your network is setup**

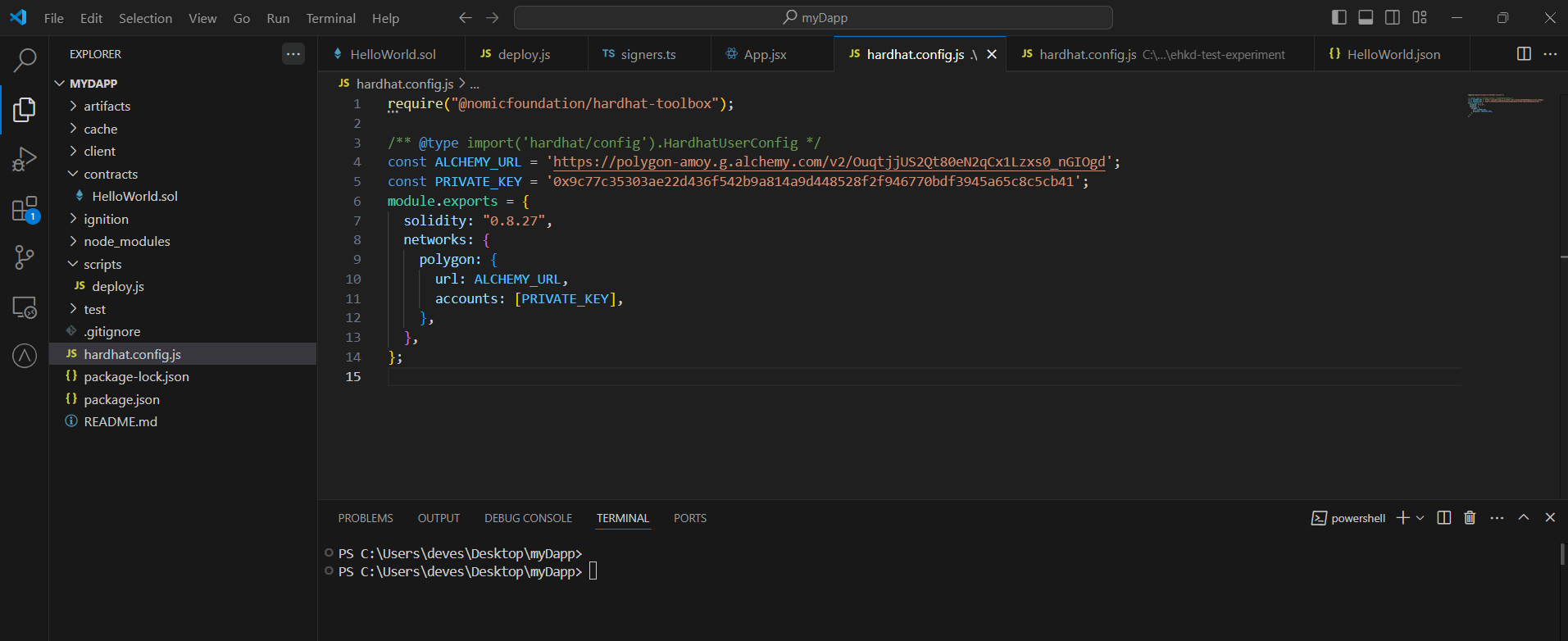
****

**Initially your balance will be zero. You can get test matic from faucet :- https://faucet.polygon.technology/**

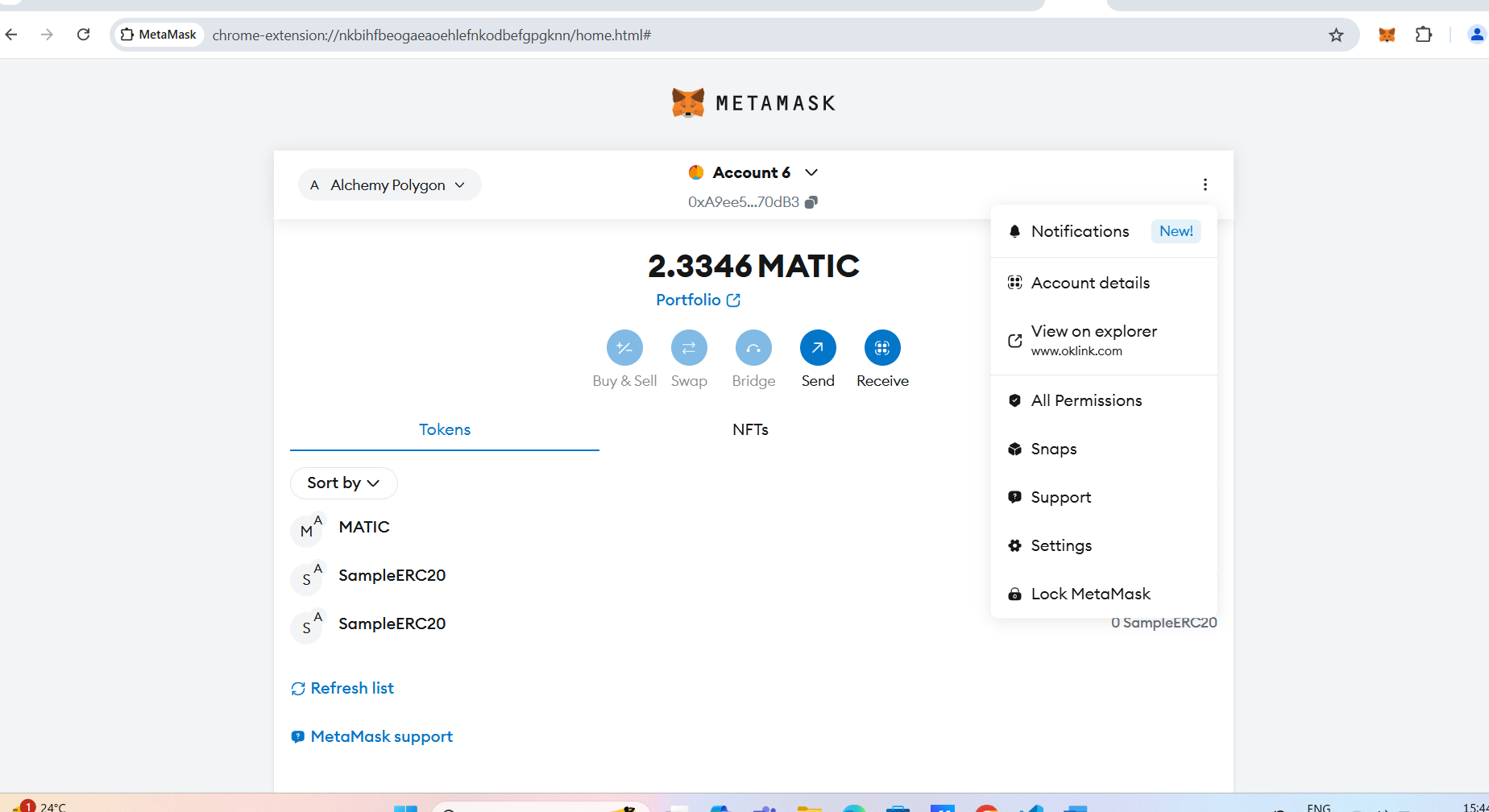
****

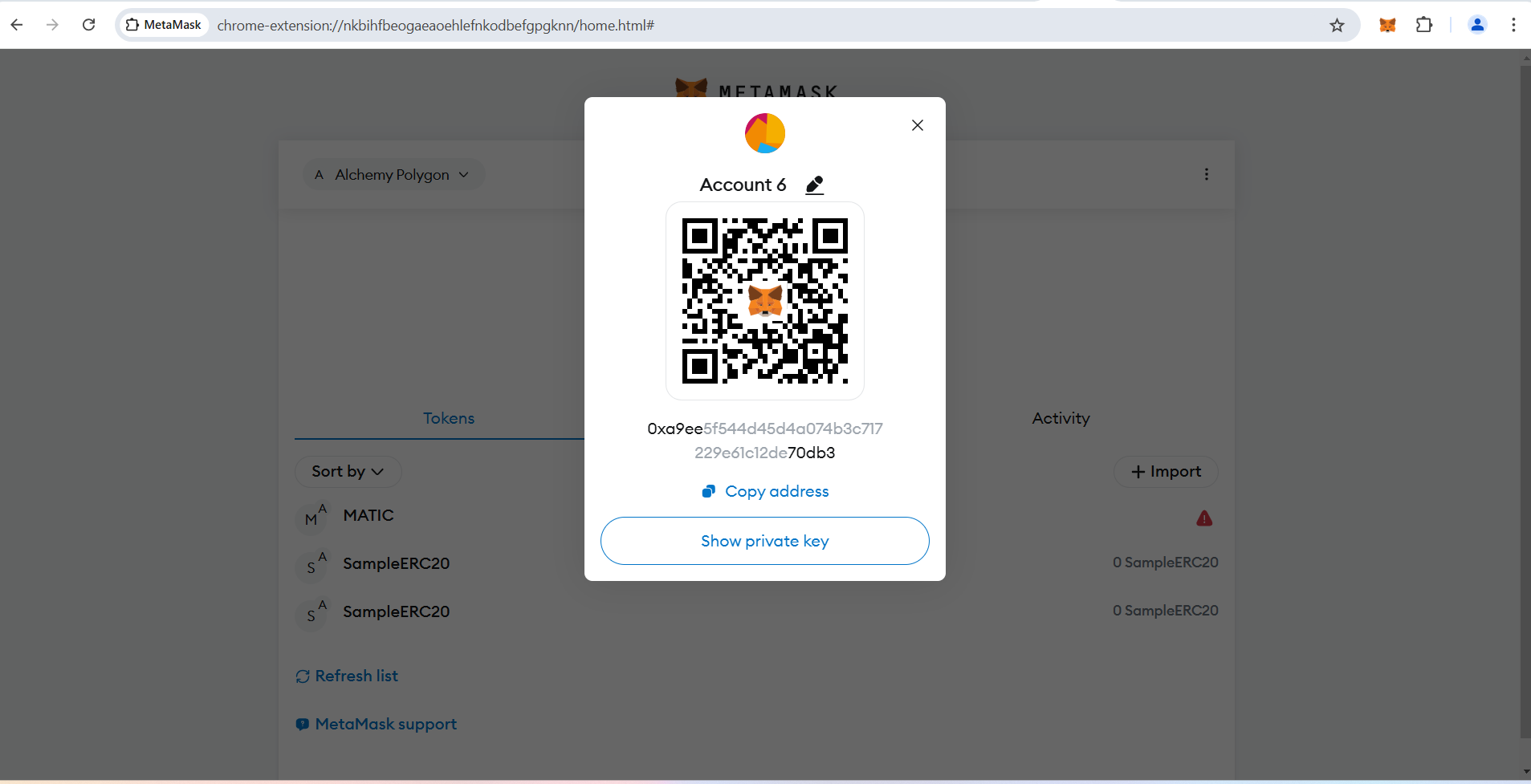
**copy your account address from metamask wallet and paste in wallet address field. Faucet will send .2MATIC.**

**Now go back to your project setup and update hardhat.config.js**

****

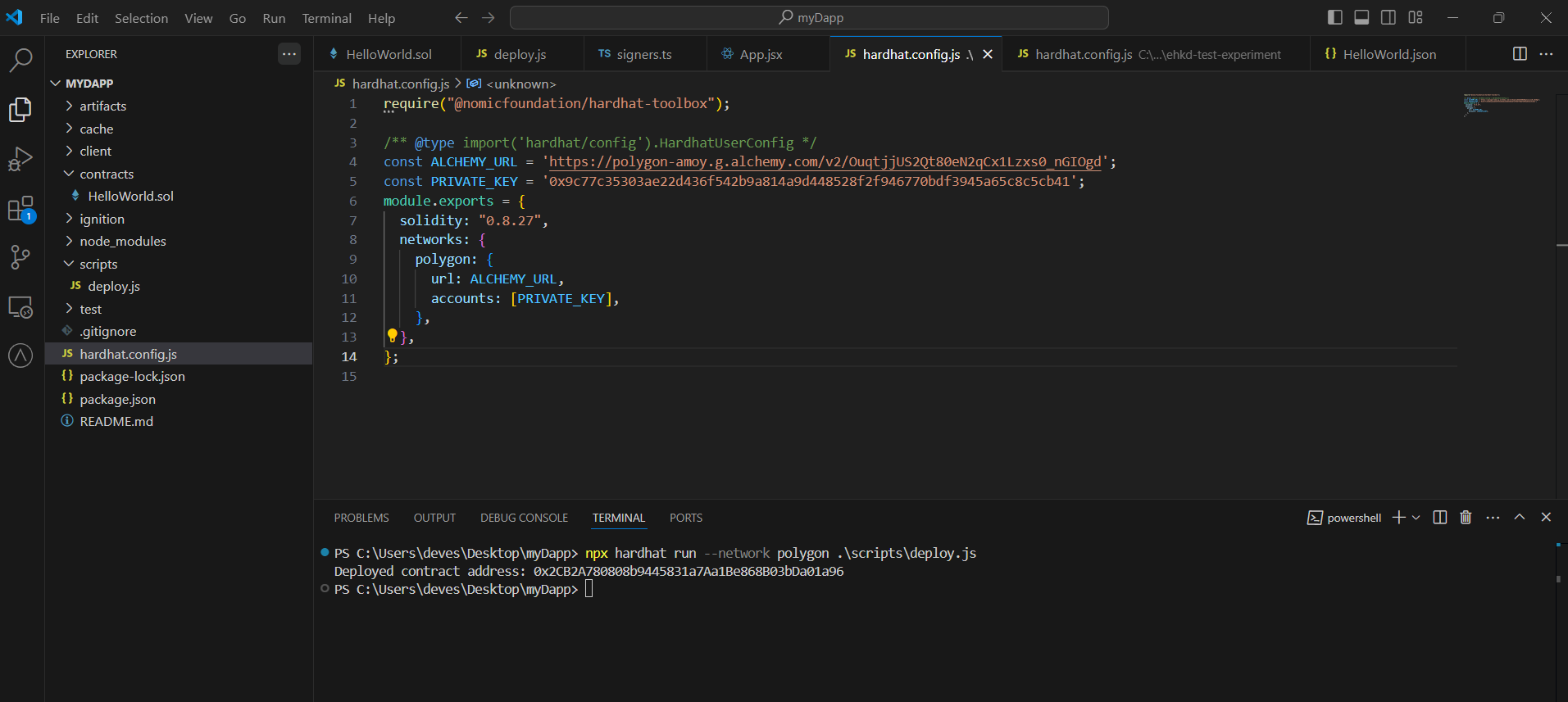
**You will get private key from Metamask wallet. Account details->Show private key**

****

****

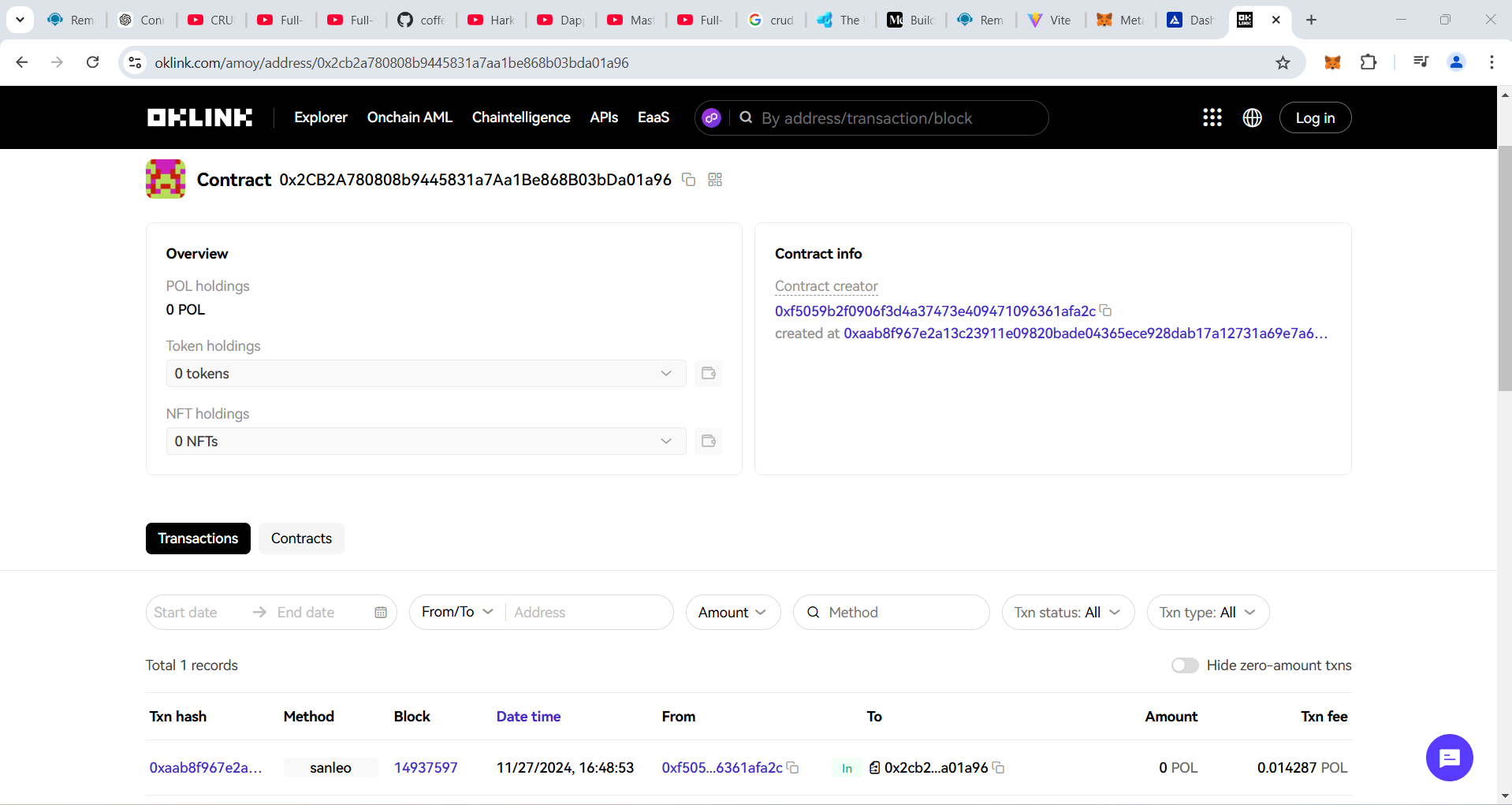
**Now execute below command in terminal to deploy smart contract on polygon network**

**npx hardhat run --network polygon .\scripts\deploy.js**

****

**Contract will be deployed and you can find contract address in console.**

**To confirm if the contract is deployed successfully, go to polygon amoy explorer and search with contract address :-** [**www.oklink.com/amoy**](http://www.oklink.com/amoy)

****

**You can verify time and to address, it should be your account address.**

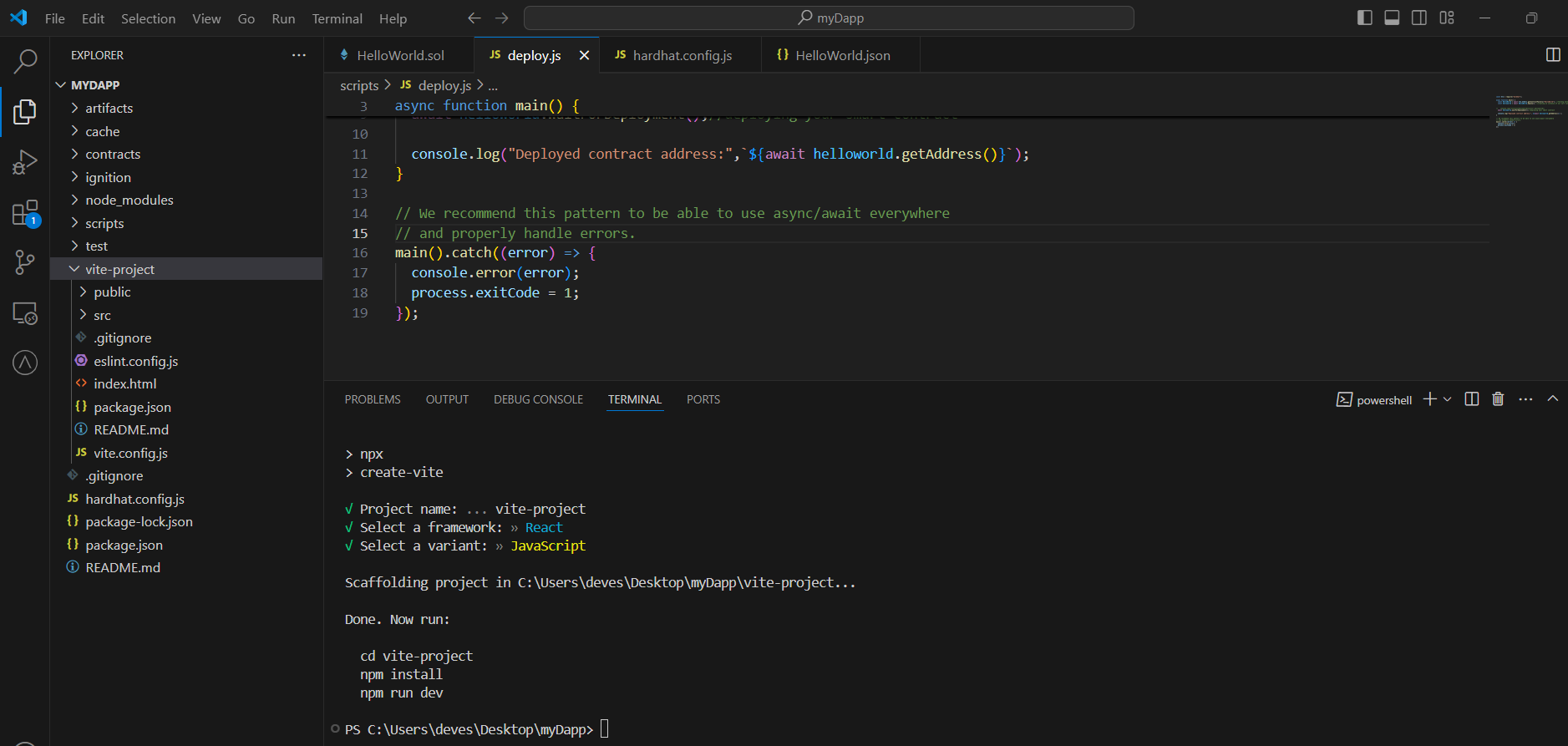
**Now we need to develop front end app.**

**To create react app type below command**

**Npm create vite@latest**

****

**Vite-project folder is created. I have renamed it to client(optional).**

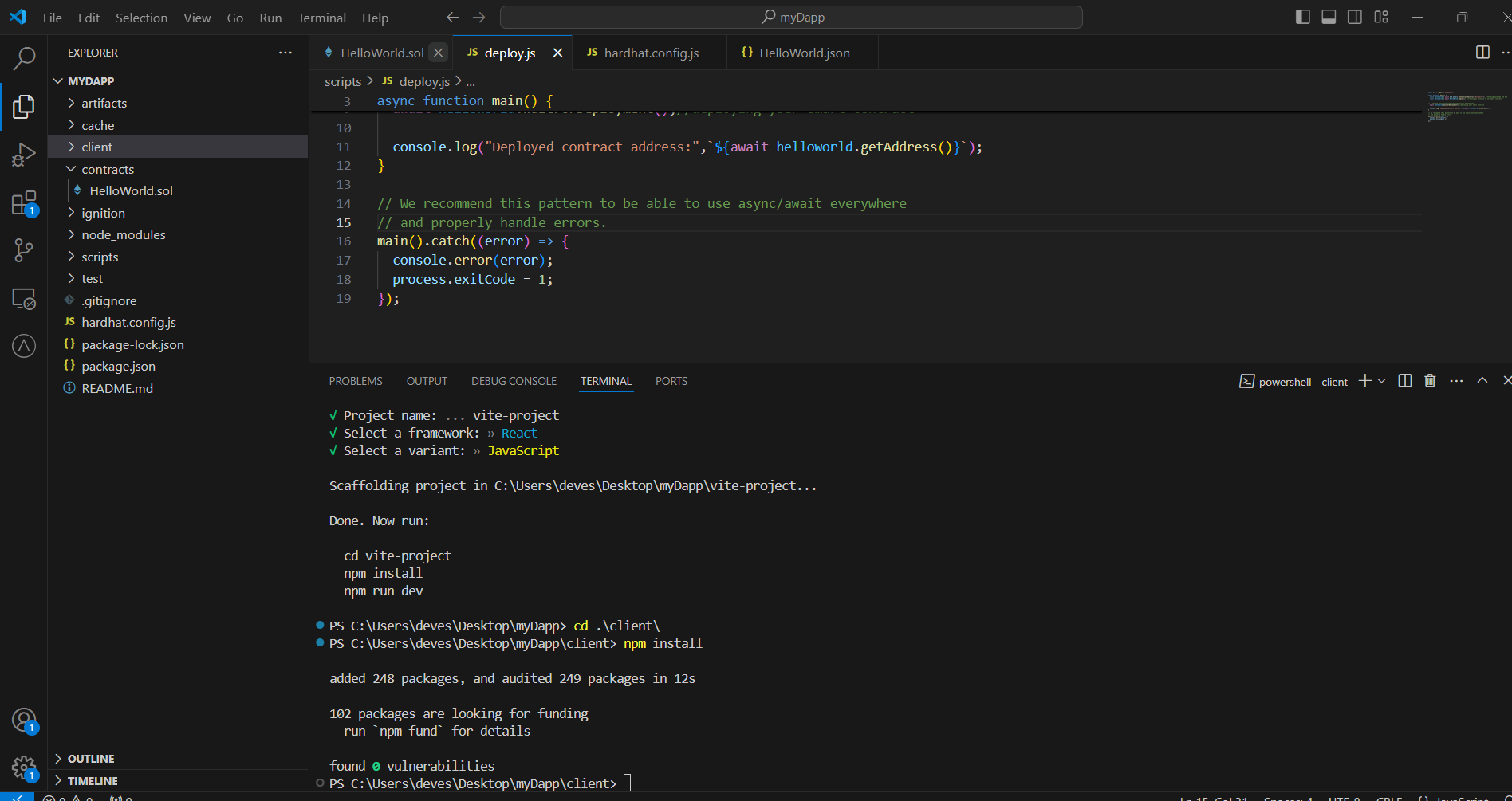
****

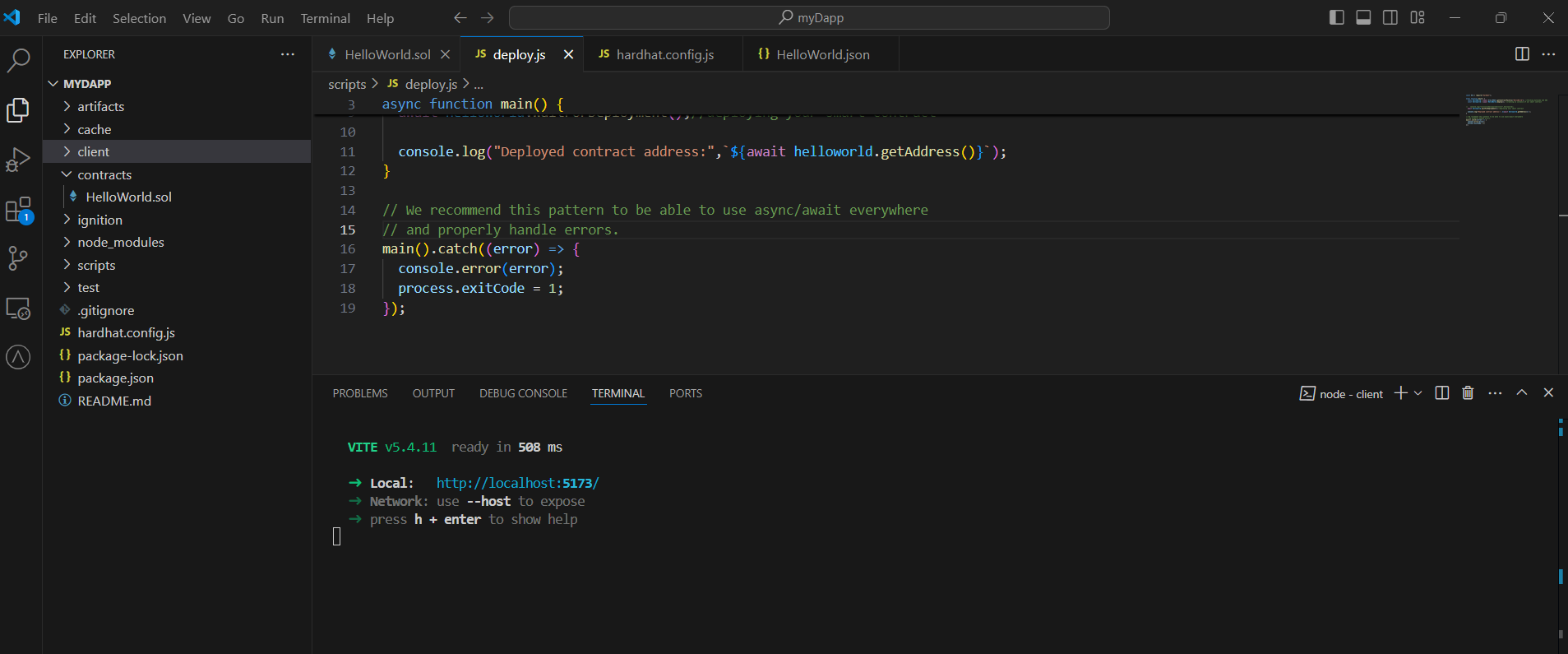
**Now run below commands**

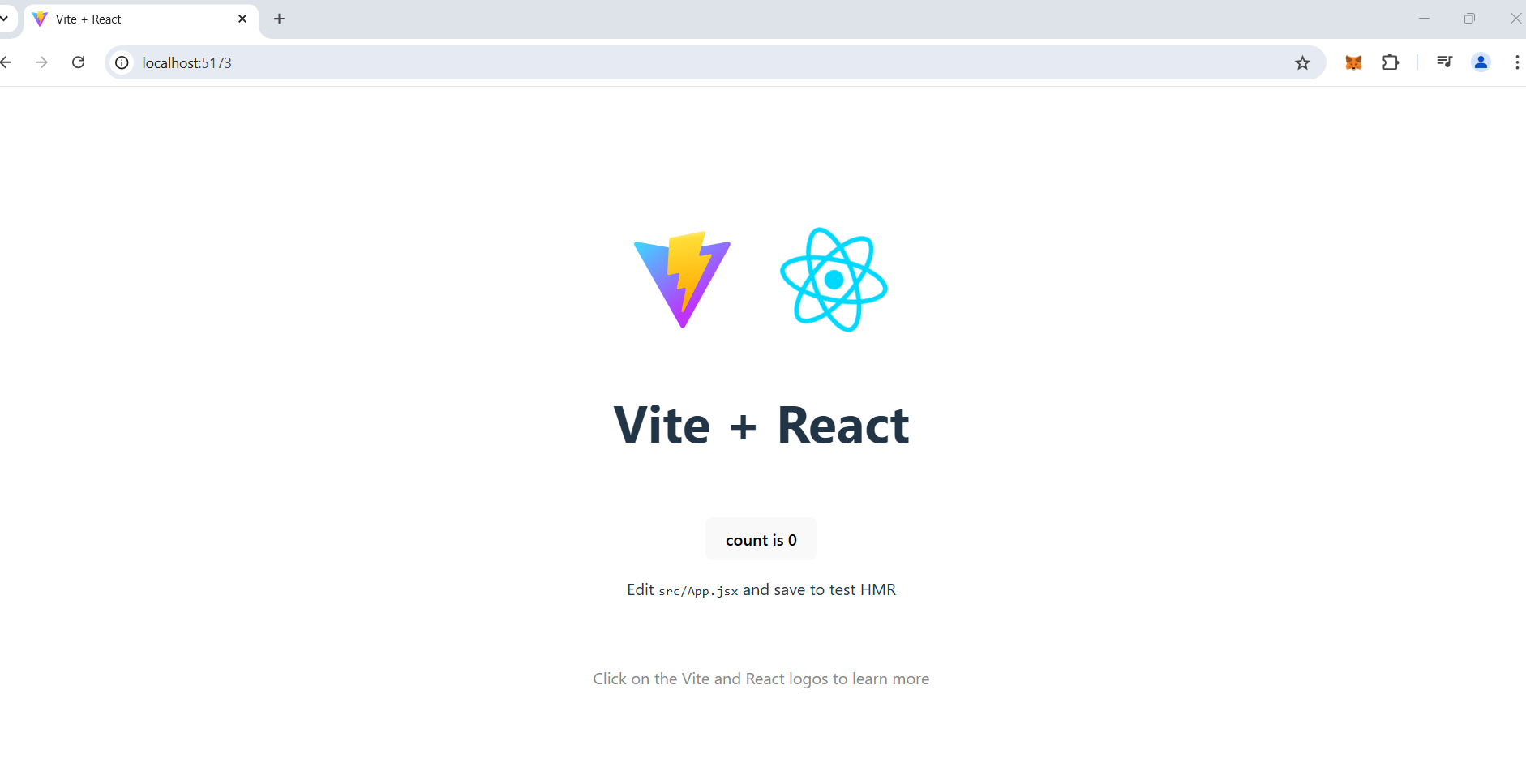
**cd client**

**npm install**

**npm run dev**

****

****

****

**Now we need to create two button and two textboxes for get and set string in blockchain.**

**Please add below code in app.jsx**

**import React, { useState } from "react";**

**import { ethers } from "ethers";**

**import abi from "../../artifacts/contracts/HelloWorld.sol/HelloWorld.json";**

**// Replace these with your contract details**

**const CONTRACT\_ADDRESS = "0x2CB2A780808b9445831a7Aa1Be868B03bDa01a96";**

**const CONTRACT\_ABI = abi;**

**function App() {**

**const [getValue, setGetValue] = useState(""); // For the "Get" text box**

**const [setValue, setSetValue] = useState(""); // For the "Set" text box**

**// Function to handle setting a value on the blockchain**

**const handleSetClick = async () => {**

**try {**

**// Connect to MetaMask**

**if (!window.ethereum) {**

**alert("MetaMask not detected!");**

**return;**

**}**

**const provider = new ethers.BrowserProvider(window.ethereum);**

**// Request accounts from MetaMask (to ensure the user is connected)**

**await provider.send("eth\_requestAccounts", []);**

**const signer = await provider.getSigner();**

**const contract = new ethers.Contract(CONTRACT\_ADDRESS, CONTRACT\_ABI.abi, signer);**

**// Interact with the contract**

**const tx = await contract.set(setValue);**

**await tx.wait(); // Wait for the transaction to be mined**

**alert("Value successfully set on blockchain!");**

**setSetValue(""); // Clear the input box**

**setGetValue("");**

**} catch (error) {**

**console.error("Error setting value:", error);**

**alert("Failed to set value on blockchain.");**

**}**

**};**

**// Function to handle getting a value from the blockchain**

**const handleGetClick = async () => {**

**try {**

**// Connect to MetaMask**

**if (!window.ethereum) {**

**alert("MetaMask not detected!");**

**return;**

**}**

**const provider = new ethers.BrowserProvider(window.ethereum);**

**// Ensure the user is connected to the wallet**

**//await provider.send("eth\_requestAccounts", []);**

**const contract = new ethers.Contract(CONTRACT\_ADDRESS, CONTRACT\_ABI.abi, provider);**

**// Interact with the contract**

**const value = await contract.get();**

**//  console.log("\*\*\*\*\*\*\*\*\*value\*\*\*\*\*\*\*\*",value);**

**setGetValue(value); // Update the "Get" text box**

**} catch (error) {**

**console.error("Error getting value:", error);**

**alert("Failed to fetch value from blockchain.");**

**}**

**};**

**return (**

**<div style={{ padding: "20px", fontFamily: "Arial" }}>**

**<h2>Blockchain Get and Set Example</h2>**

**{/\* Get Section \*/}**

**<div style={{ marginBottom: "10px" }}>**

**<button**

**onClick={handleGetClick}**

**style={{**

**padding: "10px 20px",**

**backgroundColor: "#007BFF",**

**color: "#fff",**

**border: "none",**

**borderRadius: "5px",**

**cursor: "pointer",**

**}}**

**>**

**Get**

**</button>**

**<input**

**type="text"**

**value={getValue}**

**readOnly**

**style={{**

**marginLeft: "10px",**

**width: "300px",**

**padding: "5px",**

**}}**

**/>**

**</div>**

**{/\* Set Section \*/}**

**<div>**

**<button**

**onClick={handleSetClick}**

**style={{**

**padding: "10px 20px",**

**backgroundColor: "#28A745",**

**color: "#fff",**

**border: "none",**

**borderRadius: "5px",**

**cursor: "pointer",**

**}}**

**>**

**Set**

**</button>**

**<input**

**type="text"**

**value={setValue}**

**onChange={(e) => setSetValue(e.target.value)}**

**style={{**

**marginLeft: "10px",**

**width: "300px",**

**padding: "5px",**

**}}**

**/>**

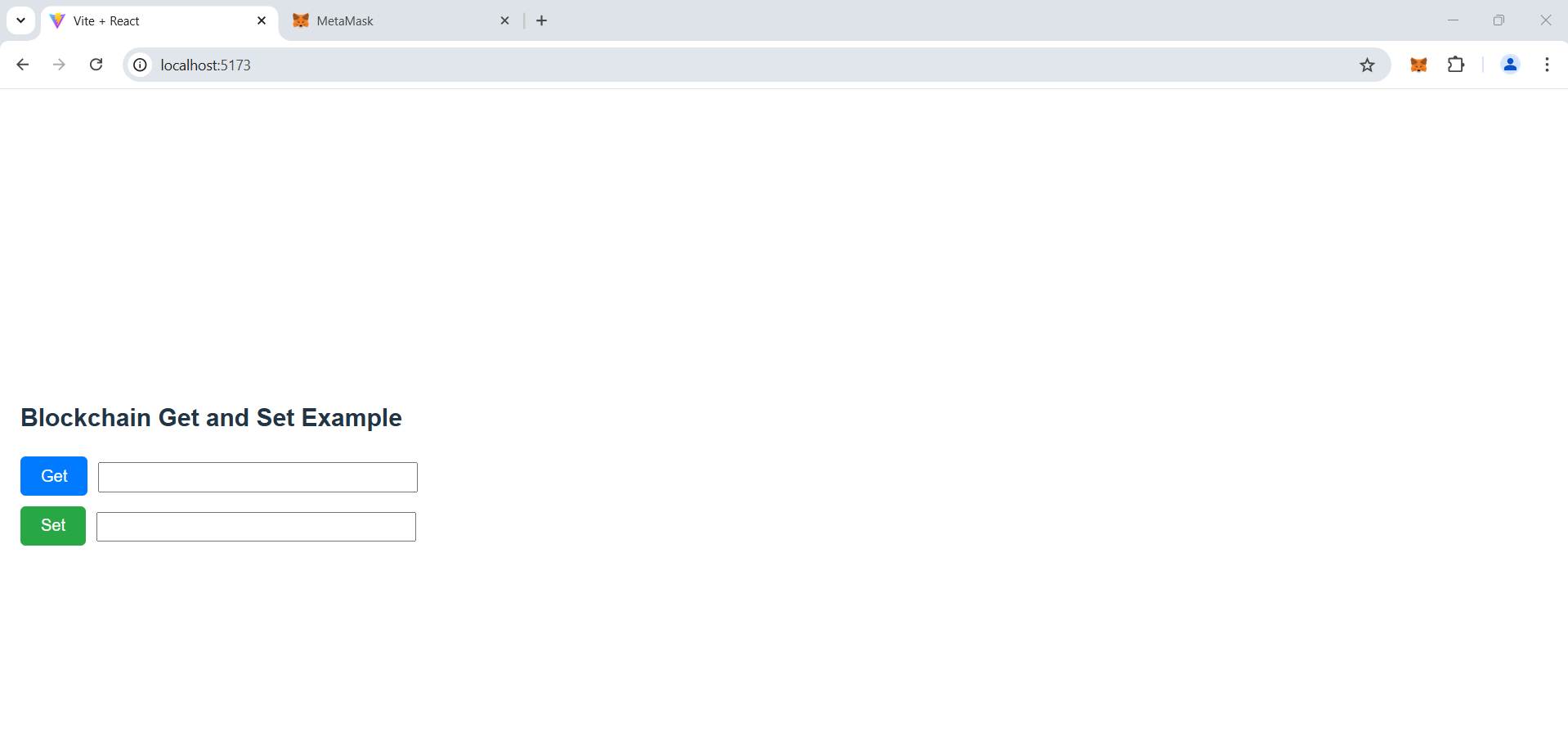
**</div>**

**</div>**

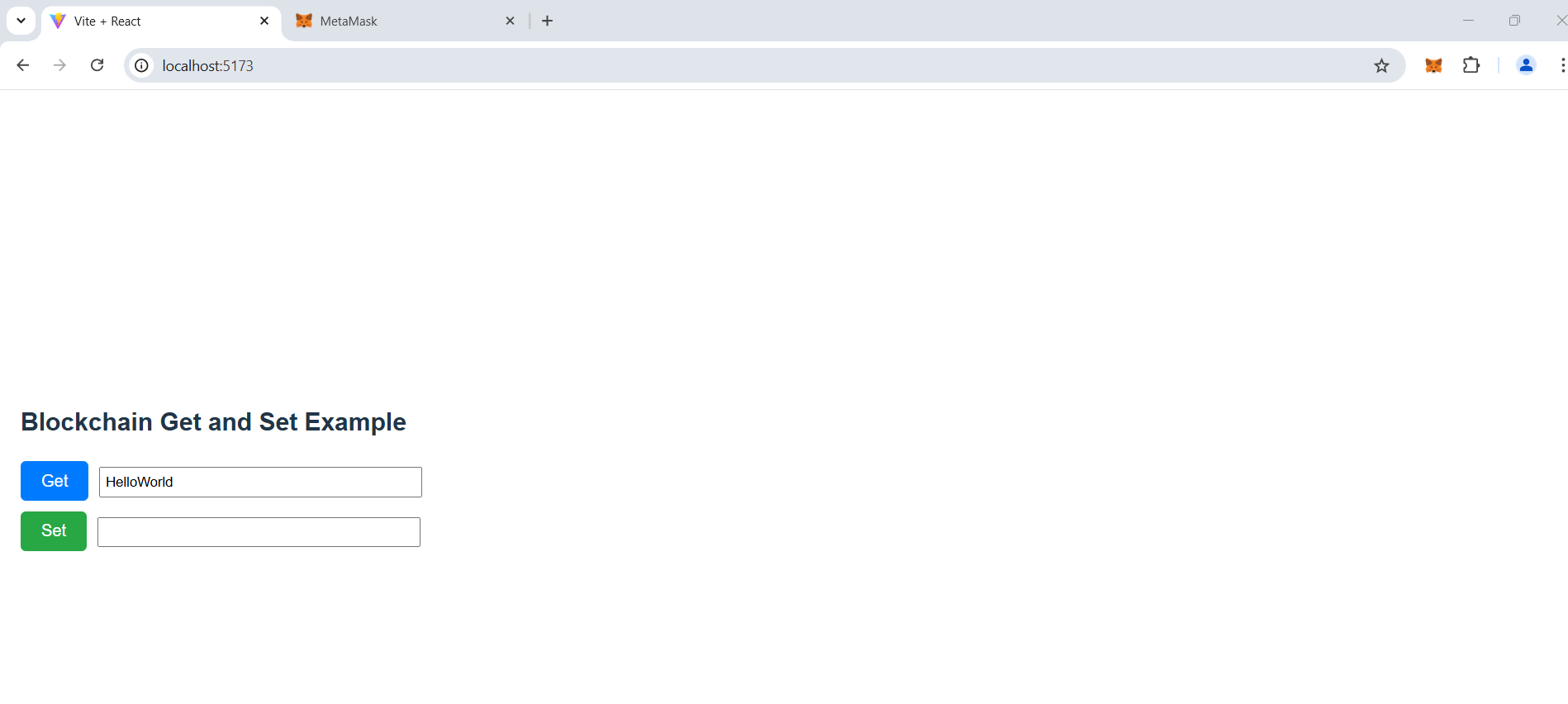
**);**

**}**

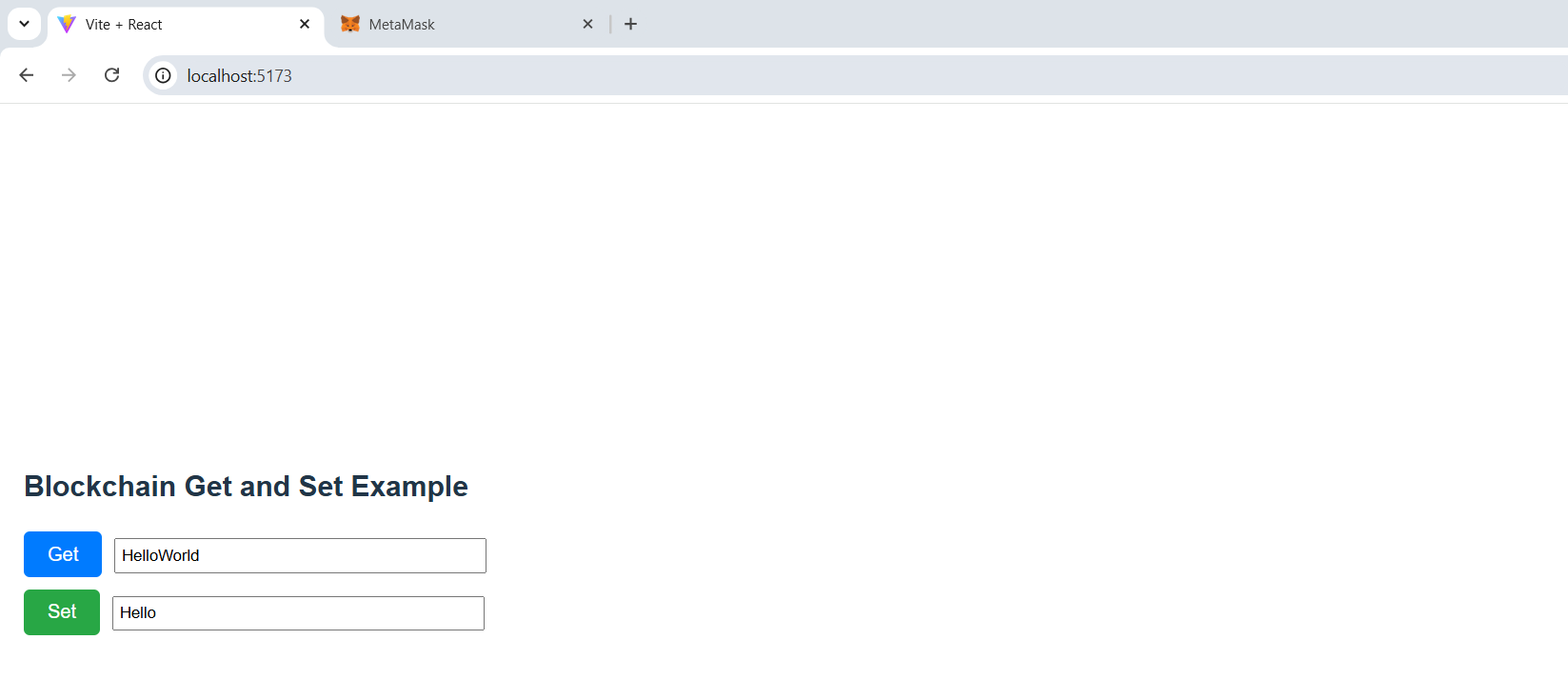
**export default App;**

****

**Click on Get**

****

**Now enter new text in set textbox and click on set**

****

**As on set we are modify blockchain so we need to pay some gas. So as soon as you click on set metamask will open and request for confirmation .**

****

**Once you confirm it will take few seconds to update on blockchain. Once successfully updated you will get success message.**

**You can click on get method to check updated value.**