

## Project Setup and Testing Guide

1. Install dependencies by running the following command:

```
npm install
```

2. Add Network Configuration to `hardhat.config.js`

```
networks: {  
  sepolia: {  
    url: "https://eth-sepolia.g.alchemy.com/v2/YOUR-PROVIDER-API-KEY",  
    accounts: ["YOUR-WALLET-PRIVATE-KEY" ]  
  }  
}
```

In this project, we use Alchemy as a blockchain infrastructure provider.

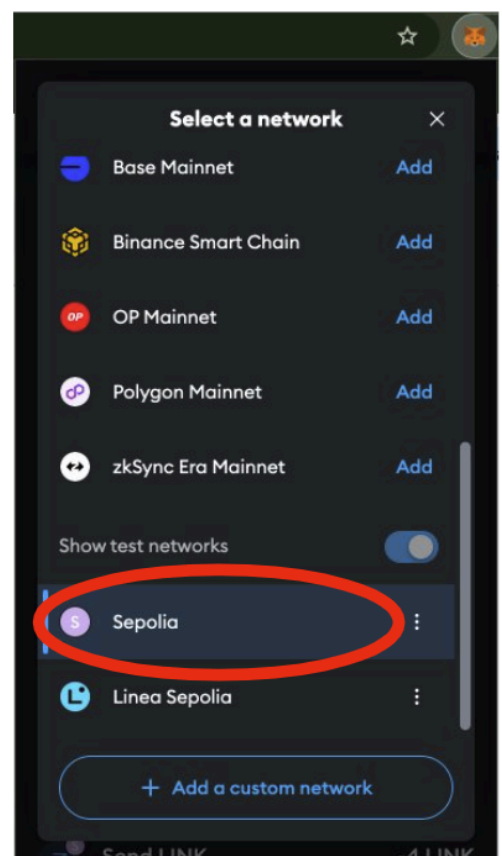
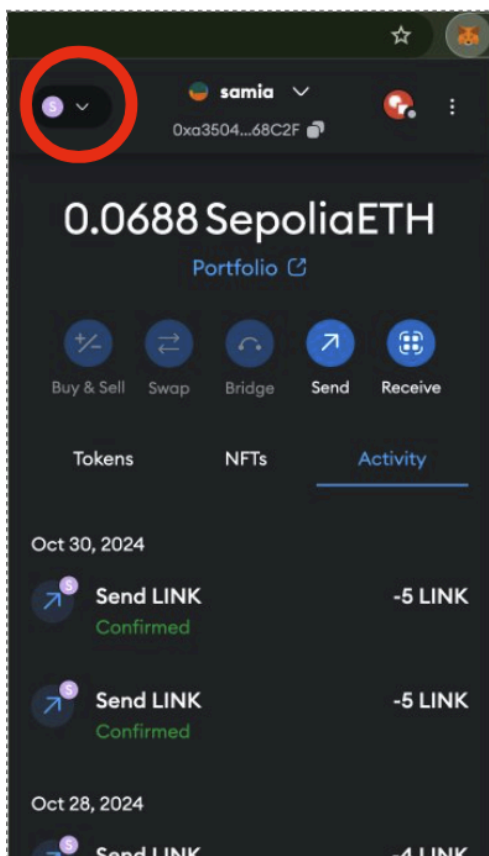
2.1 Sign up on [Alchemy's website](#) and create a new project, this process will generate an API key, copy this API key in

url:

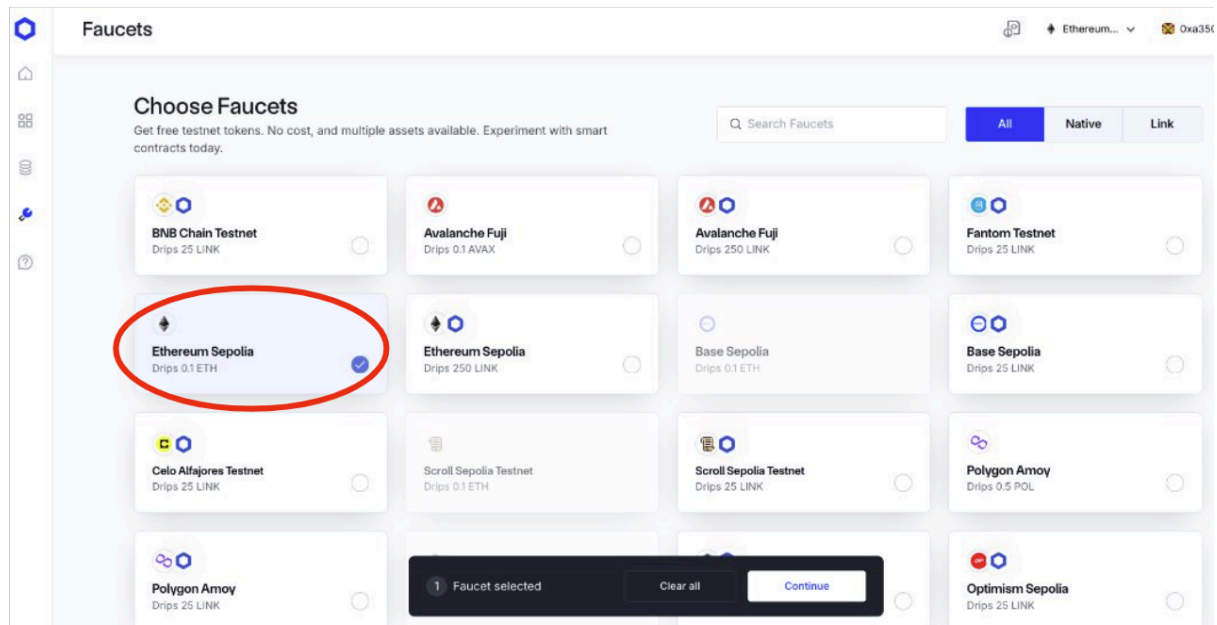
"https://eth-sepolia.g.alchemy.com/v2/**YOUR-PROVIDER-API-KEY**",

2.2 Create wallet e.g MetaMask wallet (add Metamask extension to your browser) and set the Ethereum Sepolia Testnet as a deployment network.

2.3 Enter your wallet private key to the accounts: [ "**YOUR-WALLET-PRIVATE-KEY**" ]



2.4 Fund your wallet with Sepolia Ether (e.g., using the [Chainlink faucet](#)). Select Sepolia Ethereum and click "Continue," and then enter your wallet's public address.



3. Compile the contracts, run in your terminal

```
npx hardhat compile
```

Upon successful compilation, you should see a message similar to:

Compiled 27 Solidity file successfully (EVM target: Paris)

## Compilation Output

A new directory called `/artifacts` will be created, storing the compiled contract files. Each compiled contract will have two files in this folder:

`contract.json`: Contains the contract's ABI (Application Binary Interface) and bytecode.

`contract.dbg.json`: Provides debugging information for the contract.

These files provide the essential data for local testing

```
✓ CHAINLINKORACLE_BETCONTRACT
  ✓ artifacts
    > @chainlink
    > @openzeppelin
    > build-info
  ✓ contracts
    ✓ AMM.sol
      {} AMM.dbg.json
      {} AMM.json
    ✓ Bet.sol
      {} Bet.dbg.json
      {} Bet.json
    ✓ TCOIN.sol
      {} TCOIN.dbg.json
      {} TCOIN.json
```

#### 4. Deploy ./contracts/TCOIN.sol Contract

Run the following command

```
npx hardhat ignition deploy ./ignition/modules/deployTcoin.js --network sepolia
```

TCOIN compiling output: The terminal shows the address of the newly deployed contract.

```
apple@MacBook-Pro-di-Apple chainlinkOracle_BETContract % npx hardhat ignition deploy ./ignition/modules/deployTcoin.js --network sepolia
✓ Confirm deploy to network sepolia (11155111)? ... yes
Hardhat Ignition 🚀

Deploying [ TCOIN ]

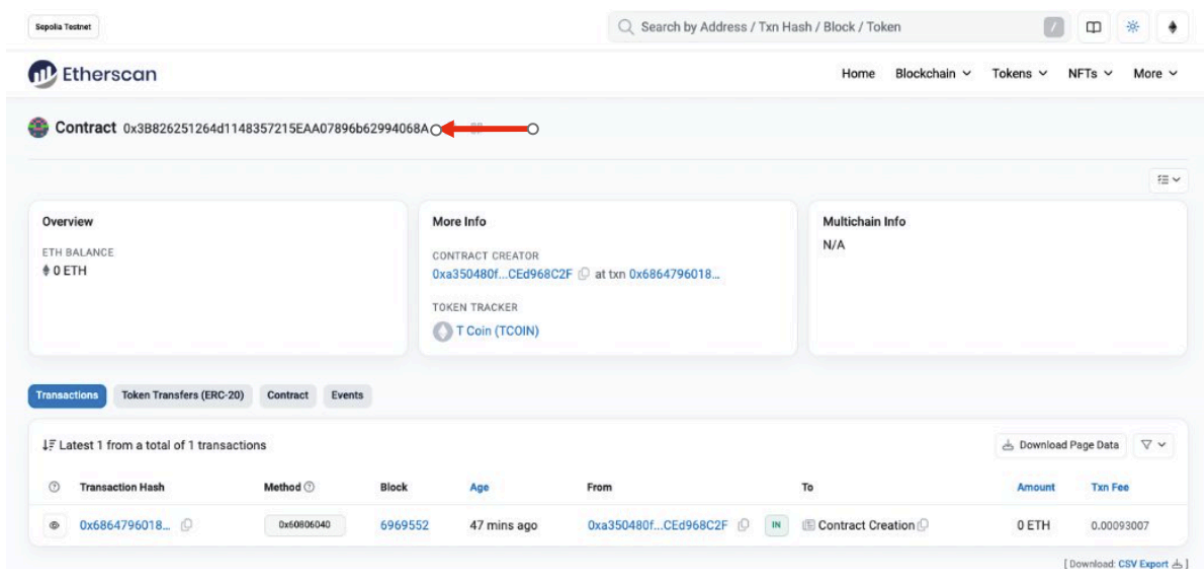
Batch #1
  Executed TCOIN#TCOIN

[ TCOIN ] successfully deployed 🚀

Deployed Addresses

TCOIN#TCOIN - 0x3B826251264d1148357215EAA07896b62994068A ←
apple@MacBook-Pro-di-Apple chainlinkOracle_BETContract %
```

You can also see the contract on [Etherscan](https://etherscan.io) by entering this address in the search bar.



Deployment output:

You'll find two JSON files in ./ignition/deployments/chain-11155112/artifacts/ after deployment:

TCOIN#TCOIN.dbg.json: Debugging information.

TCOIN#TCOIN.json: ABI and bytecode of the deployed contract on the Sepolia testnet.

Note: A constants.js file is located in the ./tests folder, where we save the smart contract properties (address and ABI) for future interactions. After deployment, copy the TCOIN contract address from the terminal output and update the tcoinAddress variable. Do not change the deployedTcoin variable, as it automatically updates with each deployment. **We'll use the same process for any future deployed contracts.**

```
JS constant.js M X
tests > JS constant.js > [0] betAddress
1 //TCOIN
2 const deployedTcoinJson = require("../ignition/deployments/chain-11155111/artifacts/TCOIN#TCOIN.json");
3 const tcoinAddress = "TCOIN-ADDRESS";
4 const tcoinAbi = deployedTcoinJson.abi;
```

Unexpected Error Handling Possible: If you want to modify the contract and redeploy it, delete only the entries in the `journals.json` file located in `/ignition/deployments/chain-11155111/artifacts/` that are related to this contract. This step is essential to prevent confusion during future deployments.

## 5. Deploy `./contract/AMM.sol` contract

**5.1** Paste the TCOIN address into the `contract` function of the `buildModule` class in `./ignition/modules/deployAMM.js`. This address is required for the AMM constructor to use TCOIN as a token for exchanging with Ether.

```
JS deployAMM.js M ●
ignition > modules > JS deployAMM.js > ...
1  const { buildModule } = require("@nomicfoundation/hardhat-ignition/modules");
2
3  module.exports = buildModule("AMM", (m) => {
4    const tokenaddress = "TCOIN-ADDRESS"; // TCOIN address
5    const ammContract = m.contract("AMM", [tokenaddress]);
6    return { ammaddress: ammContract.address };
7  });
```

## 5.2 Run the following command

```
npx hardhat ignition deploy ./ignition/modules/deployAMM.js -- network sepolia
```

The screenshot shows the Etherscan interface for a contract on the Sepolia Testnet. The contract address is `0xe1E89120C62a0EF01Fc379849f44572c73dc3583`. The 'Overview' section shows an ETH balance of 0. The 'More Info' section shows the contract creator as `0xa350480f...CEd968C2F` at transaction `0x70cfd2af6e...`. The 'Transactions' tab is selected, showing a list of transactions. The latest transaction is a contract creation for 0 ETH with a fee of 0.00835245 ETH.

Transaction Hash	Method	Block	Age	From	To	Amount	Txn Fee
<a href="#">0x70cfd2af6e...</a>	<code>0x0806040</code>	<a href="#">6969888</a>	6 mins ago	<a href="#">0xa350480f...CEd968C2F</a>	<a href="#">Contract Creation</a>	0 ETH	0.00835245

## 6. Deploy `./contract/Bet.sol` contract

**6.1** Paste the TCOIN address and AMM address into the `contract` function of the `buildModule` class in `./ignition/modules/deployBet.js`.

```

JS deployBet.js M
ignition > modules > JS deployBet.js > ...
1  const {ethers} = require("hardhat");
2  const { buildModule } = require("@nomicfoundation/hardhat-ignition/modules");
3
4  function convertDateToTimestamp(dateString) {
5      const date = new Date(dateString);
6      const timestamp = Math.floor(date.getTime() / 1000); // Convert to seconds
7      return timestamp;
8  }
9  module.exports = buildModule("Bet", (m) => {
10     const humanReadableDate = "31 october 2024 23:00:00";
11     const timestamp1= convertDateToTimestamp(humanReadableDate);
12
13     const oracle = m.contract("Bet", ["AMM-ADDRESS", "TOKEN-ADDRESS", 1, timestamp1]);
14     return { contractaddress: oracle.address };
15 });

```

6.2 Run the following command

`npx hardhat ignition deploy ./ignition/modules/deployBet.js -- network sepolia`

The screenshot displays the Etherscan interface for a contract on the Sepolia Testnet. The contract address is 0xD6Cb85a913B873e25f9fED456E0F0adEe7E7b3ac. The Overview section shows an ETH balance of 0 ETH. The More Info section lists the contract creator as 0xa350480f...CEd968C2F. The Transactions section shows a single transaction (0x24ceed4315...) for contract creation, with a block number of 6969971 and a timestamp of 1 min ago. The interface also includes a search bar, navigation links, and a footer with a knowledge base link.

Deployments output:

```

✓ ignition
  ✓ deployments / chain-11155111
    ✓ artifacts
      {} AMM#AMM.dbg.json
      {} AMM#AMM.json
      {} Bet#Bet.dbg.json
      {} Bet#Bet.json
      {} TCOIN#TCOIN.dbg.json
      {} TCOIN#TCOIN.json
    > build-info
      {} deployed_addresses.json
      {} journal.jsonl

```

## 7. Testing Contracts

The test scripts for contracts are in the `ignition/modules` folder. To test a smart contract on the Ethereum Sepolia network, we need to interact with it and call its functions. This requires creating a contract instance using the contract's ABI, bytecode, and a connected wallet. We will use the variables from the `constants.js` file to create each contract instance.

`constant.js`:

```
JS constant.js M X
tests > JS constant.js > [?] betAddress
1 //TCOIN
2 const deployedTcoinJson = require("../ignition/deployments/chain-11155111/artifacts/TCOIN#TCOIN.json");
3 const tcoinAddress = "TCOIN-ADDRESS";
4 const tcoinAbi = deployedTcoinJson.abi;
5
6 //AMM
7 const deployedAmmJson = require("../ignition/deployments/chain-11155111/artifacts/AMM#AMM.json");
8 const ammAddress = "AMM-ADDRESS";
9 const ammAbi = deployedAmmJson.abi;
10
11 //Bet
12 const deployedBetJson = require("../ignition/deployments/chain-11155111/artifacts/Bet#Bet.json");
13 const betAddress = "BET-ADDRESS";
14 const betAbi = deployedBetJson.abi;
15
16 module.exports = {
17   tcoinAddress,
18   tcoinAbi,
19   ammAddress,
20   ammAbi,
21   betAddress,
22   betAbi
23 };
```

7.1 Create `.env` file under `./tests/` and set `YOUR-PROVIDER_API_KEY` to your Alchemy API key and `YOUR-WALLET-PRIVATE_KEY` to your wallet's private key.

```
.env M X
tests > .env
1 ALCHEMY_API_KEY = https://eth-sepolia.g.alchemy.com/v2/YOUR-PROVIDER-API-KEY
2
3 PRIVATE_KEY_WALLET = YOUR-WALLET-PRIVATE-KEY
```

## Test for Fair Player: No MEV Attempt and No Interaction with Bet Contract Dependency (AMM Contract)

### 1. Interact with AMM contract

Note: Comment Out the swap() Function Call IB `./tests/interactWithAmm.js`

Call `addLiquidity()` function with 0.0006 Ether and 600 TCOIN, run the following command:

```
node tests/interactWithAmm.js
```

`addLiquidity()` transaction in [etherscan](https://etherscan.io)

This screenshot shows the 'Transaction Details' page on Etherscan for a Sepolia Testnet transaction. The transaction is successful and has 6195 block confirmations. It was executed 22 hours ago on October 29, 2024, at 04:00:48 PM UTC. The transaction action is a call to the 'Add Liquidity' function of the AMM contract (0xe1E89120c62a0EF01Fc379849f44572c73dc3583) by the user 0xa350480f...CEd968C2F. The transaction transferred 600 TCOIN (TCOIN) and 0.0006 ETH from the user to the contract. The transaction fee was 0.00028546897766544 ETH.

Field	Value
Transaction Hash	0xc163748dc69c1c56263e8602f19c1ae6ffde50c92e6517a62739f2341f263a2e
Status	Success
Block	6970843 (6195 Block Confirmations)
Timestamp	22 hrs ago (Oct-29-2024 04:00:48 PM UTC)
Transaction Action	Call Add Liquidity Function by 0xa350480f...CEd968C2F on 0xe1E89120...c73dc3583
From	0xa350480f236e13199e3D502FA02d30dCEd968C2F
To	0xe1E89120C62a0EF01Fc379849f44572c73dc3583
ERC-20 Tokens Transferred	2
Value	0.0006 ETH
Transaction Fee	0.00028546897766544 ETH

This screenshot shows the 'Contract' page on Etherscan for the AMM contract (0xe1E89120c62a0EF01Fc379849f44572c73dc3583). The contract is created by 0xa350480f...CEd968C2F. The overview shows an ETH balance of 0.0006 ETH and token holdings of \$0.00 (1 Tokens). The contract creator is 0xa350480f...CEd968C2F at transaction 0x70cfbd2af6e... The contract has two events: 'Add Liquidity' and 'Contract Creation'. The 'Add Liquidity' event shows 600 TCOIN being added to the contract by 0xa350480f...CEd968C2F on October 29, 2024, at 04:00:48 PM UTC, with a value of 0.0006 ETH and a fee of 0.00028546 ETH. The 'Contract Creation' event shows the contract being created by 0xa350480f...CEd968C2F on October 29, 2024, at 04:00:48 PM UTC, with a value of 0 ETH and a fee of 0.00835245 ETH.

Age	From	To	Amount	Txn Fee
15 hrs ago	0xa350480f...CEd968C2F	0xe1E89120...c73dc3583	0.0006 ETH	0.00028546
19 hrs ago	0xa350480f...CEd968C2F	Contract Creation	0 ETH	0.00835245



## 2. Interact with Bet Contract

Note: Comment Out the win() Function Call in tests/interactWithBet.js

Call bet() function, run the following command in your terminal:

```
node tests/interactWithBet.js
```

The screenshot shows the Etherscan Sepolia Testnet interface. At the top, there's a search bar and navigation links. The main header identifies the page as a 'Contract' for address 0xD6Cb85a913B873e25f9fED456E0F0adEe7E7b3ac. Below this, there are three tabs: 'Overview', 'More Info', and 'Multichain Info'. The 'Overview' tab shows an 'ETH BALANCE' of 0.0000001 ETH. The 'More Info' tab shows the 'CONTRACT CREATOR' as 0xa350480f...CED968C2F at transaction 0x24ceed4315... The 'Multichain Info' tab shows 'N/A'. Below the tabs, there are four sub-tabs: 'Transactions', 'Token Transfers (ERC-20)', 'Contract', and 'Events'. The 'Transactions' sub-tab is active, showing a table of transactions. The table has columns for Transaction Hash, Method, Block, Age, From, To, Amount, and Txn Fee. Two transactions are listed: one with hash 0x8c5526c07f5... (Method: Bet, Block: 6975266, Age: 5 mins ago, From: 0xa350480f...CED968C2F, To: 0xD6Cb85a9...Ee7E7b3ac, Amount: 0.0000001 ETH, Txn Fee: 0.00034975) and another with hash 0x24ceed4315... (Method: 0x60806040, Block: 6969971, Age: 18 hrs ago, From: 0xa350480f...CED968C2F, To: Contract Creation, Amount: 0 ETH, Txn Fee: 0.00955874).

## 3. Interact with Bet Contract : Call win () function

Note: Comment Out the bet() Function Call in tests/interactWithBet.js

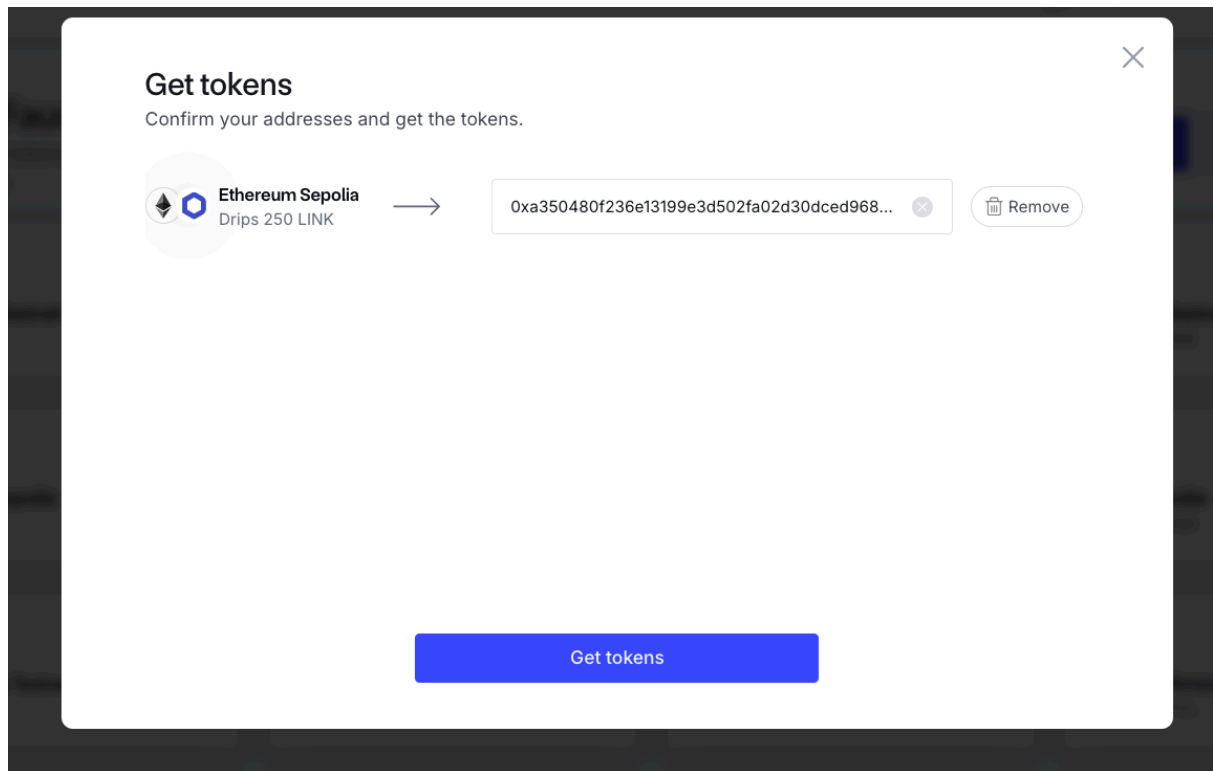
### 3.1 Ensure your wallet has sufficient LINK Tokens

The win() function requires a LINK token balance in the Bet contract to cover fees for interacting with the Chainlink oracle for that you need to fund the Bet contract with enough LINK, to do that first Fund Your Wallet with LINK from [Chainlink faucet](#), then send a sufficient amount of LINK (e.h 10 LINK) from your wallet to the Bet contract address. Select Link token for Ethereum Sepolia testnet in [Chainlink faucet](#):

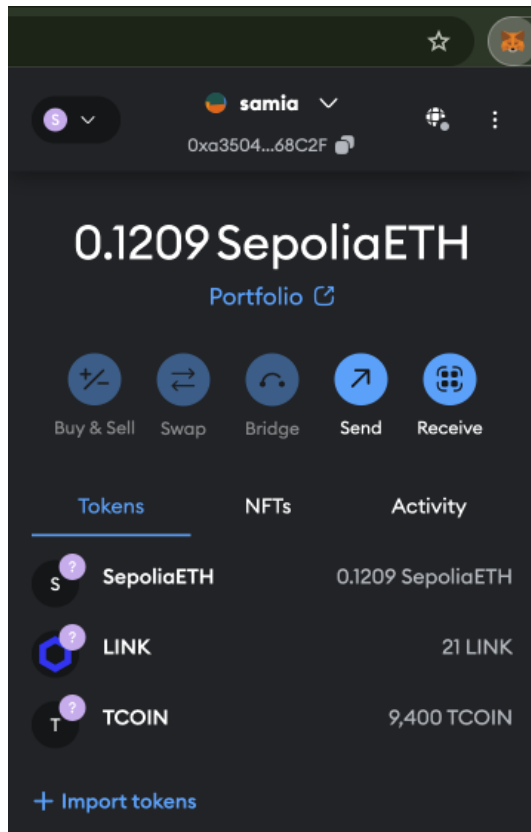
The screenshot shows the Chainlink Faucets website. At the top, there's a header with the Chainlink logo and the word 'Faucets'. Below this, there's a section titled 'Choose Faucets' with a sub-header 'Get free testnet tokens. No cost, and multiple assets available. Experiment with smart contracts today.' There's a search bar and three tabs: 'All', 'Native', and 'Link'. The 'Link' tab is selected. Below the tabs, there's a grid of faucet options. Each option is a card with a Chainlink logo, the faucet name, and the amount of LINK tokens. The options are: BNB Chain Testnet (Drips 25 LINK), Avalanche Fuji (Drips 0.1 AVAX), Avalanche Fuji (Drips 250 LINK), Fantom Testnet (Drips 25 LINK), Ethereum Sepolia (Drips 0.1 ETH), Ethereum Sepolia (Drips 250 LINK), Base Sepolia (Drips 0.1 ETH), Base Sepolia (Drips 25 LINK), Celo Alfajores Testnet (Drips 25 LINK), Scroll Sepolia Testnet (Drips 0.1 ETH), Scroll Sepolia Testnet (Drips 25 LINK), Polygon Amoy (Drips 0.5 POL), Polygon Amoy (Drips 25 LINK), and Optimism Sepolia (Drips 25 LINK). The 'Ethereum Sepolia (Drips 250 LINK)' option is selected with a blue checkmark. At the bottom, there's a dark bar with a '1 Faucet selected' message, a 'Clear all' button, and a 'Continue' button.



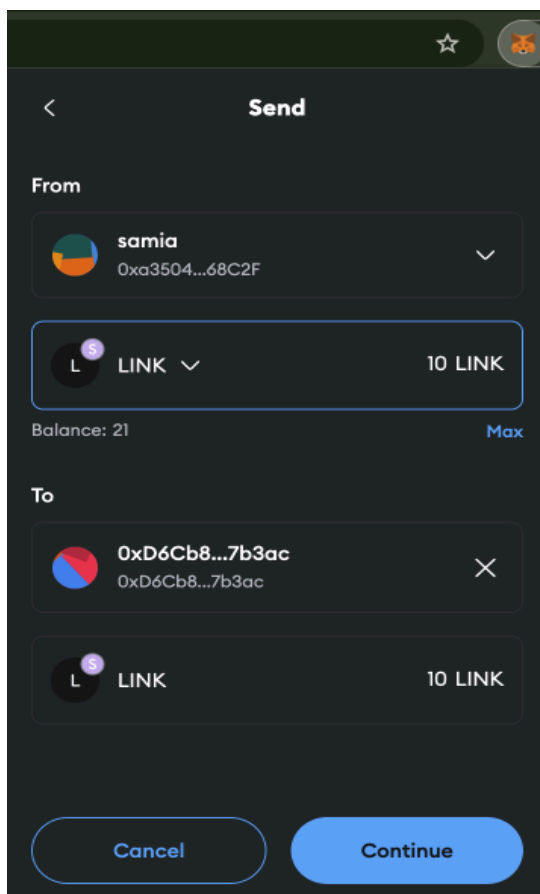
copy your wallet address:



Import the LINK token contract address (0x779877A7B0D9E8603169DdBd7836e478b4624789) into your wallet for proper visibility.



### 3.2 Fund the Bet Contract with LINK Tokens: Transfer 10 LINK



```

    invocation: null,
    revert: {
      signature: 'Error(string)',
      name: 'Error',
      args: [ 'ERC20: transfer amount exceeds balance' ]
    },
  },
}

```

Note: Comment Out the bet() Function Call in tests/interactWithBet.js

Transaction of the win() function call:

The Bet contract after executing win() function:

Sepolia Testnet

Search by Address / Txn Hash / Block / Token

Etherscan

Home

Blockchain

Tokens

NFTs

More

Contract

0xD6Cb85a913B873e25f9ED456E0F0adEe7E7b3ac

Overview

ETH BALANCE

0 ETH

TOKEN HOLDINGS

\$0.00 (1 Tokens)

More Info

CONTRACT CREATOR

0xa350480f...CEd968C2F at txn 0x24ceed4315...

Multichain Info

N/A

Transactions

Internal Transactions

Token Transfers (ERC-20)

Contract

Events

Latest 3 from a total of 3 transactions

Download Page Data

Transaction Hash	Method	Block	Age	From	To	Amount	Txn Fee
0x99dd4db887...	Win	6975675	1 min ago	0xa350480f...CEd968C2F	0xD6Cb85a9...Ee7E7b3ac	0 ETH	0.00513219
0x8c5526c07f5...	Bet	6975266	1 hr ago	0xa350480f...CEd968C2F	0xD6Cb85a9...Ee7E7b3ac	0.0000001 ETH	0.00034975
0x24ceed4315...	0x60806040	6969971	20 hrs ago	0xa350480f...CEd968C2F	Contract Creation	0 ETH	0.00955874

As shown in the picture above, the Bet contract transfers its entire balance to the player upon executing the win() function. This transfer is made only after verifying, through the Chainlink oracle, that no MEV attempt has occurred.

### Test for UnFair Player: No MEV Attempt and No Interaction with Bet Contract Dependency (AMM Contract)

1. redeploy the Bet contract and update the necessary changes in the `constante.js` file

Note: Before redeploying the Bet contract delete only the entries in the `journals.json` file located in `/ignition/deployments/chain-11155111/artifacts/` that are related to this contract. This step is crucial to avoid any confusion during future deployments.

Run the following command:

```
npx hardhat ignition deploy ./ignition/modules/deployBet.js -- network sepolia
```

2. Interact with Bet contract

Call `bet()` function, run the following command in your terminal:

Note: Comment Out the win() Function Call in `tests/interactWithBet.js`

```
node tests/interactWithBet.js
```

Sepolia Testnet

Search by Address / Txn Hash / Block / Token

Etherscan

Home Blockchain Tokens NFTs More

Contract 0x823120a161B0Ad6271B4e276554a6C4690385BE2

Overview

ETH BALANCE  
0.0000001 ETH

More Info

CONTRACT CREATOR  
0xa350480f...CEd968C2F at txn 0xb82521fe41d...

Multichain Info  
N/A

Transactions Token Transfers (ERC-20) Contract Events

Latest 2 from a total of 2 transactions

Transaction Hash	Method	Block	Age	From	To	Amount	Txn Fee
0x9a76b80921...	Bet	6975856	2 mins ago	0xa350480f...CEd968C2F	0x823120a1...690385BE2	0.0000001 ETH	0.00028147
0xb82521fe41d...	0x60806040	6975845	4 mins ago	0xa350480f...CEd968C2F	Contract Creation	0 ETH	0.01077836

[ 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.

### 3. Interact with AMM contract

Call `swap()` function with 0.0003 Ether, run the following command in your terminal:

Note: Comment Out the `addliquidity()` Function Call in `tests/interactWithBet.js`

```
node tests/interactWithAmm.js
```

Transaction of the `swap()` function call:

The screenshot shows the Etherscan interface for a transaction on the Sepolia Testnet. The transaction is successful and has 2 block confirmations. It is a swap function call on the 0xa350480f...CEd968C2F contract, initiated by 0xe1E89120...c73dc3583. The transaction value is 0.0003 ETH.

**Transaction Details**

[ This is a Sepolia Testnet transaction only ]

Transaction Hash: 0x4232ceae3523792c401ca7c8aa260dc86d88fd7a41f2a683684ae374950a733d

Status: Success

Block: 6975882 (2 Block Confirmations)

Timestamp: 19 secs ago (Oct-30-2024 09:54:12 AM UTC)

Transaction Action: Call Swap Function by 0xa350480f...CEd968C2F on 0xe1E89120...c73dc3583

From: 0xa350480f236e13199e3D502FA02d30dCEd968C2F

To: 0xe1E89120C62a0EF01Fc379849f44572c73dc3583

ERC-20 Tokens Transferred: All Transfers Net Transfers

From 0xe1E89120...c73dc3583 To 0xa350480f...CEd968C2F For 200 T Coin (TCOIN)

Value: 0.0003 ETH

The AMM contract after executing `swap()` function:

The screenshot shows the Etherscan interface for the AMM contract (0xe1E89120C62a0EF01Fc379849f44572c73dc3583). The contract has an ETH balance of 0.0009 ETH and token holdings of \$0.00 (1 Tokens). The contract creator is 0xa350480f...CEd968C2F. The transactions tab shows the latest 3 transactions, including a swap, an add liquidity, and a contract creation.

**Contract** 0xe1E89120C62a0EF01Fc379849f44572c73dc3583

**Overview**

ETH BALANCE  
0.0009 ETH

TOKEN HOLDINGS  
\$0.00 (1 Tokens)

**More Info**

CONTRACT CREATOR  
0xa350480f...CEd968C2F at txn 0x70cfbd2af6e...

**Multichain Info**  
N/A

**Transactions** Internal Transactions Token Transfers (ERC-20) Contract Events

Latest 3 from a total of 3 transactions

Transaction Hash	Method	Block	Age	From	To	Amount	Txn Fee
0x4232ceae35...	Swap	6975882	39 secs ago	0xa350480f...CEd968C2F	0xe1E89120...c73dc3583	0.0003 ETH	0.00065448
0xc163748dc6...	Add Liquidity	6970843	17 hrs ago	0xa350480f...CEd968C2F	0xe1E89120...c73dc3583	0.0006 ETH	0.00028546
0x70cfbd2af6e...	0x60806040	6969888	21 hrs ago	0xa350480f...CEd968C2F	Contract Creation	0 ETH	0.00835245

[ Download: CSV Export ]

#### 4. Interact with Bet contract

Note: Comment Out the bet() Function Call in `tests/interactWithBet.js`

Call `win()` function, run the following command in your terminal:

```
node tests/interactWithBet.js
```

The Bet contract before execution the `win()` function:

Sepolia Testnet

Search by Address / Txn Hash / Block / Token

Etherscan

Home Blockchain Tokens NFTs More

Contract 0x823120a161B0Ad6271B4e276554a6C4690385BE2

Overview

ETH BALANCE

0.0000001 ETH

More Info

CONTRACT CREATOR

0xa350480f...CEd968C2F at txn 0xb82521fe41d...

Multichain Info

N/A

Transactions Token Transfers (ERC-20) Contract Events

Latest 2 from a total of 2 transactions

Transaction Hash	Method	Block	Age	From	To	Amount	Txn Fee
0x9a76b80921...	Bet	6975856	2 mins ago	0xa350480f...CEd968C2F	0x823120a1...690385BE2	0.0000001 ETH	0.00028147
0xb82521fe41d...	0x60806040	6975845	4 mins ago	0xa350480f...CEd968C2F	Contract Creation	0 ETH	0.01077836

[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.

The Bet contract after execution the `win()` function:

Sepolia Testnet

Search by Address / Txn Hash / Block / Token

Etherscan

Home Blockchain Tokens NFTs More

Contract 0x823120a161B0Ad6271B4e276554a6C4690385BE2

Overview

ETH BALANCE

0.0000001 ETH

TOKEN HOLDINGS

\$0.00 (1 Tokens)

More Info

CONTRACT CREATOR

0xa350480f...CEd968C2F at txn 0xb82521fe41d...

Multichain Info

N/A

Transactions Internal Transactions Token Transfers (ERC-20) Contract Events

Latest 3 from a total of 3 transactions

Transaction Hash	Method	Block	Age	From	To	Amount	Txn Fee
0xd81fe04306a...	Win	6975972	39 secs ago	0xa350480f...CEd968C2F	0x823120a1...690385BE2	0 ETH	0.00152012
0x9a76b80921...	Bet	6975856	25 mins ago	0xa350480f...CEd968C2F	0x823120a1...690385BE2	0.0000001 ETH	0.00028147
0xb82521fe41d...	0x60806040	6975845	27 mins ago	0xa350480f...CEd968C2F	Contract Creation	0 ETH	0.01077836

[Download: CSV Export]

As shown in the picture above, the Bet contract does not transfer its entire balance to the player upon executing the win() function. It verifies through the Chainlink oracle that a MEV attempt has occurred in the player's historical transactions.