

Blockchain & Solidity Lab4 – Voting dApp Development

S2BC



Lab 4: Run a dApp and Consider Next Steps

- BUILD / TEST / INTEGRATE / **RUN**

So far, you've followed the steps in Labs 1 to 3, gaining valuable insights into the core components of blockchain development. Now, in Lab 4, we will discuss crucial considerations for running a dApp in Morpheus.

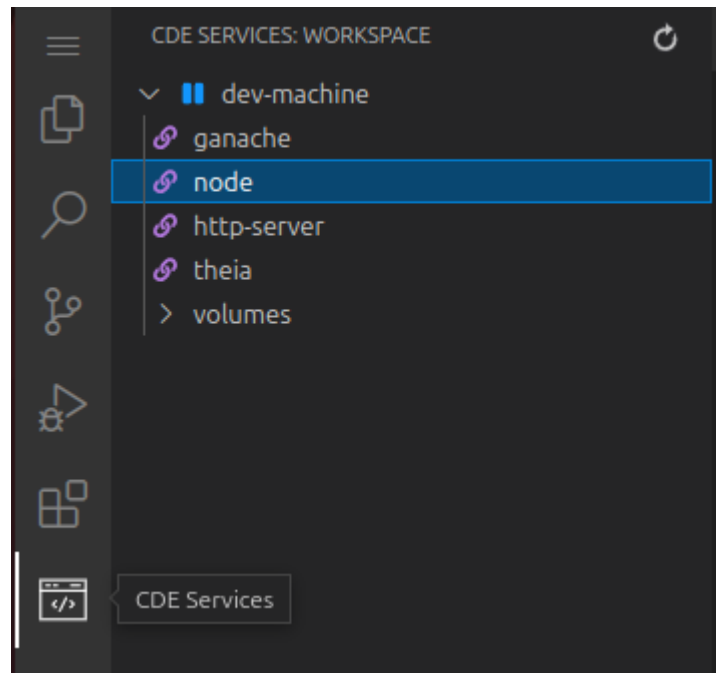
1. Run the front end

in voting-dapp-2023/frontend run:

```
node server.js (or npm start)
```

This will start the server for your dApp's frontend.

Then, in your IDE interface from Morpheus, find in the left menu the CDE menu and click on the node service in order to open your web app.



2. Try the Voting Dapp

- Click on the connect button, a metamask pop up will ask you if you want to connect and then accept with the deployer account imported.
- You can check the Voting status in the Vote Panel:

Voting dApp 2023

Connected

0x65d493425fD6d67993FF90375375139FCd2D36E0

Vote Panel

Admin Panel

Election Title: No title yet

Election ID: Not started yet

Timer: 0 s left

ID	Candidates	Votes count	Vote Button
----	------------	-------------	-------------

No candidates yet

- Then go the Administrator Panel to Start an election:

Vote Panel

Admin Panel

Start an Election:

Election Title

Candidate 1

Candidate 2

Candidate 3

+ candidates separated by (,)

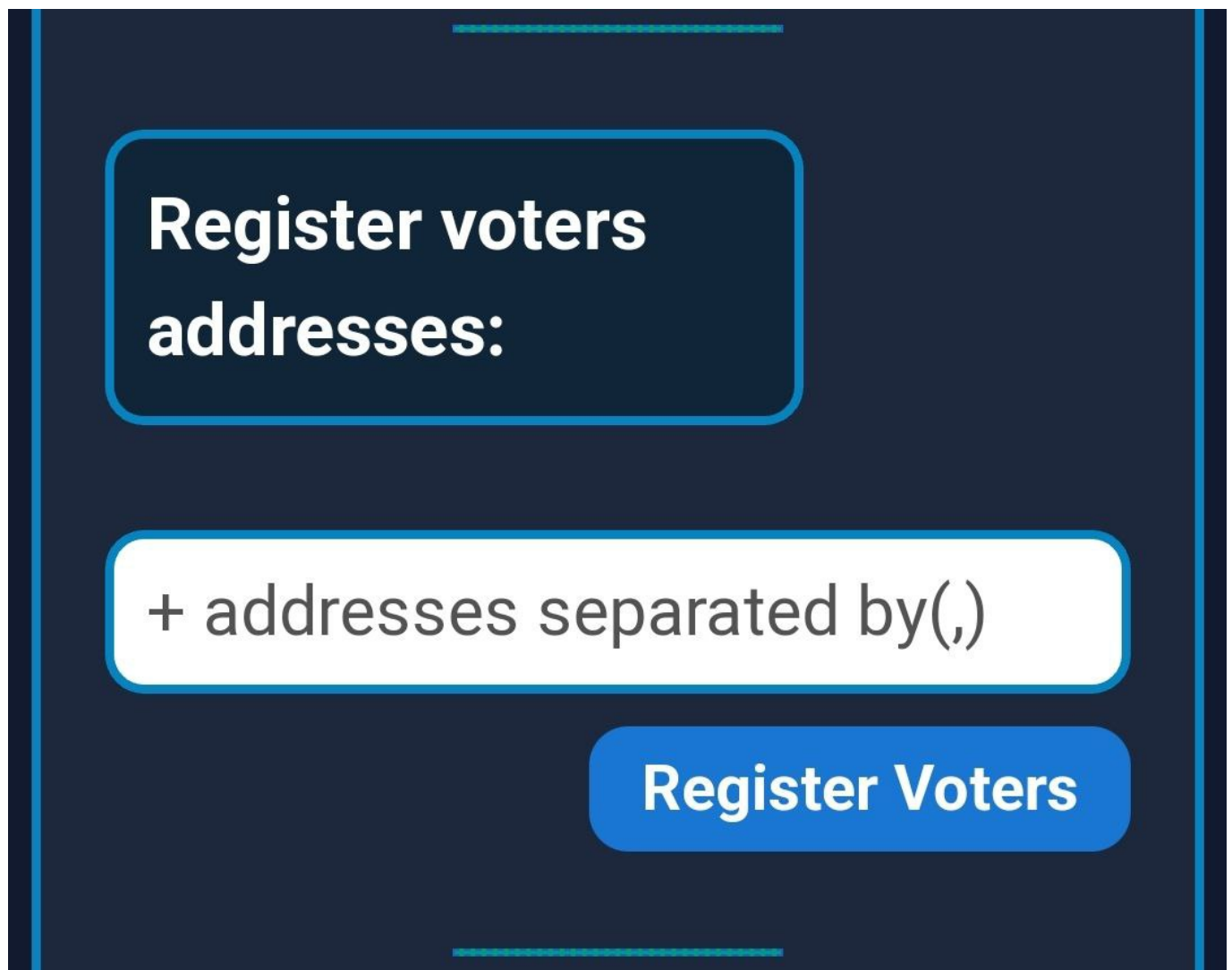
How long the election will last?:

Duration in minutes

Start Election

add election title, candidates and election duration, then click start election button

Next register voters:



**Register voters
addresses:**

+ addresses separated by(,)

Register Voters

And you should be able to return to Vote Panel and see the election ongoing information.

Cast a vote.

Then you can end the election by click on "End Election" button in the admin panel.

When election is finished, you can now mint the result with the next "Mint results" Buttun.

When everything is finished, you can reinitialise the Election by clicking on "Rehinitialise" election.

You can as well open you console in the devoloper options of your explorer (F12) to see relative console logs while navigating the app.

3. Migrating to Sepolia Testnet and Utilizing Etherscan

To successfully migrate your dApp to the Sepolia Testnet and leverage Etherscan for enhanced visibility, follow the steps below:

Step 1: Obtain RPC_URL and Etherscan API Key

1.1 Obtain the RPC_URL for Sepolia from Morpheus, Alkemy's website, or Infura.

1.2 Obtain a free Etherscan API Key from the Etherscan website.

Step 2: Update Configuration Files

2.1 Open your **.env** file and modify the values as follows:

```
RPC_URL="https://eth-sepolia.g.alchemy.com/v2/APIKEY"
PRIVATE_KEY="00000...000"
API_KEY="APIKEYFROMETHERSCAN"
```

Ensure the private key corresponds to the deployer account on Sepolia. You can use any account created with Metamask, and acquire testnet ETH from a faucet like Alkemy faucet.

2.2 Update the **chainID** in your **hardhat.config.js** file from 1303 to 11155111. Then change the network name "votingchain" to "sepolia"

```
require("@nomicfoundation/hardhat-toolbox");

require("dotenv").config();

/** @type import('hardhat/config').HardhatUserConfig */
module.exports = {
  solidity: "0.8.22",
  networks: {
    sepolia: {
      chainId: 11155111,
      url: process.env.RPC_URL,
      accounts: [process.env.PRIVATE_KEY],
    },
  },
  etherscan: {
    apiKey: process.env.API_KEY,
  },
  paths: {
    artifacts: "./src/artifacts",
    contracts: './src/contracts',
  }
};
```

2.3 In your **frontend/public/script.js** file, replace all occurrences of "1303" with "11155111" to ensure the frontend connects to the Sepolia chainID.

Tip: You can select **1305**, then do Ctrl+D several times to get all occurrences selected, and then past 11155111.

Certainly! Here's an organized and clear version of your tutorial:

2.3 Update ChainID in `frontend/public/script.js` File

To ensure that your frontend connects to the Sepolia chainID, follow these steps to replace all occurrences of "1303" with "11155111" in the `frontend/public/script.js` file.

Manual Method:

1. Open your text editor and navigate to the `frontend/public/script.js` file.
2. Locate the first occurrence of "1303" and position your cursor at the beginning of the number.
3. Press `Ctrl + D` (or `Cmd + D` on macOS) to select the current occurrence.
4. Continue pressing `Ctrl + D` until all instances of "1303" are selected.
5. Type "11155111" to replace the selected occurrences.
6. Save the file.

Alternative Method using Find and Replace:

1. Open the `frontend/public/script.js` file in your text editor.
2. Use the find function (`Ctrl + F` or `Cmd + F`) to search for "1303."
3. Click on "Replace" or "Replace All."
4. Enter "11155111" as the replacement and confirm the action.
5. Save the file.

Step 3: Redeploy the Contract on Sepolia

3.1 Navigate to the hardhat folder in your terminal.

3.2 Run the following command to redeploy the contract on Sepolia:

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

Step 4: Update Contract Address in Frontend

4.1 Once the deployment is complete, locate the voting contract address.

4.2 Copy the contract address and update the variable in `frontend/public/script.js` as follows:

```
const contractAddress = 'votingcontractaddress';
```

Step 5: Restart the Server

5.1 Start or restart your server using the following command:

```
node frontend/server.js
```

Step 6: Verification on Etherscan

6.1 If you have chosen to verify your contract on Etherscan, you have two methods available:

Method 1: Using Hardhat

To verify your contract using Hardhat, follow these steps:

1. Navigate to your Hardhat directory in the terminal.
2. Run the following command, replacing `<votingcontractaddress>` with the actual address of your deployed contract:

```
npx hardhat verify <votingcontractaddress> --network sepolia
```

3. Review the response in the terminal to confirm the success or any output related to the verification process.
4. Check Etherscan to verify if the contract has been successfully verified.

Method 2: Using Etherscan Interface

An alternative method is to use the Etherscan interface directly. Provide the following information to Etherscan:

- Contract Code
- Compiler Version
- ABI (Application Binary Interface) of the contract

This method involves interacting with the Etherscan website to manually input the required details for verification.

Choose the method that best fits your workflow or preference. Successful verification ensures transparency and allows users to explore transactions and events within the voting contract on Etherscan, providing detailed insights at each step of the election.

Step 7: Test the New Setup

7.1 Retry launching a new election on this updated setup to ensure seamless functionality.

By following these steps, your dApp should now be successfully migrated to the Sepolia Testnet, utilizing the specified `RPC_URL` and providing enhanced insights through Etherscan verification.

4. Uploading Your dApp on Morpheus app library to share with community

For detailed steps on uploading your dApp, refer to the [documentation](#).

