**Unit 4 – Blockchain and Decentralized Applications**

**Lab 8 Manual**

**Objective:**

By the end of this lab, students will be able to:

* Understand what Hyperledger Besu is and its purpose.
* Set up and configure a Hyperledger Besu node.
* Deploy and interact with a simple smart contract on the Sepolia testnet

**8.1. Introduction to Hyperledger Besu**

Hyperledger Besu is an Ethereum client designed for public and private blockchain networks. It supports Ethereum Mainnet and testnets like Sepolia. It is compatible with Ethereum tools like Remix and Truffle. It provides enterprise-grade blockchain capabilities. It supports both permissioned and permissionless networks.

**8.2. Setting Up Hyperledger Besu**

**Step 1: Install Hyperledger Besu**

1. Open a terminal and install Besu via its latest release on GitHub:

* Download and extract Besu: <https://github.com/hyperledger/besu>
* curl -O <https://hyperledger.jfrog.io/artifactory/besu-binaries/releases/besu-23.10.0.tar.gz>
* tar -xvzf besu-23.10.0.tar.gz
* cd besu-23.10.0
* ./bin/besu –version

1. This verifies that Besu is installed.

**Step 2: Run Besu and Connect to Sepolia**

1. Create a directory for Besu data:

mkdir besu\_data

1. Start Besu with the following command:

./bin/besu --network=sepolia \

--data-path=besu\_data \

--rpc-http-enabled \

--rpc-http-api=ETH,NET,WEB3

1. This connects Besu to the Sepolia testnet and enables the RPC interface.

**Step 3: Verify the Node Connection**

1. Open a terminal and use the curl command to verify Besu and to test the connection:

curl -X POST –data '{"jsonrpc":"2.0","method":"web3\_clientVersion","params":[],"id":1}' http://127.0.0.1:8545

1. A successful response confirms that Besu is running.

**8.3. Deploying a Smart Contract**

**Step 1: Create a Project Directory Structure**

1. Set up a Truffle project in VS Code:
   * mkdir lab8\_besu\_project
   * cd lab8\_besu\_project
   * truffle init
2. Ensure the directory has the following structure:

lab8\_besu\_project/

|-- contracts/

| `-- SimpleStorage.sol

|-- migrations/

| `-- 2\_deploy\_contracts.js

|-- test/

|-- truffle-config.js

**Step 2: Write a Simple Contract**

In the contracts/ folder, create a file SimpleStorage.sol:

pragma solidity ^0.8.0;

contract SimpleStorage {

uint256 private storedData;

function set(uint256 x) public {

storedData = x;

}

function get() public view returns (uint256) {

return storedData;

}

}

**Step 3: Configure Truffle**

Open truffle-config.js and edit it to include:

module.exports = {

networks: {

besu: {

host: "127.0.0.1", // Localhost (default: none)

port: 8545, // Standard Ethereum port (default: none)

network\_id: "1337", // Any network (default: none)

},

},

compilers: {

solc: {

version: "^0.8.0", // Fetch exact version from solc-bin (default: truffle's version)

},

},

};

**Step 4: Deploy the Contract**

1. Create a migration script in migrations/2\_deploy\_contracts.js:

const SimpleStorage = artifacts.require("SimpleStorage");

module.exports = function (deployer) {

deployer.deploy(SimpleStorage);

};

1. Deploy the contract:

truffle migrate --network besu (Note the deployed contract address)

**Step 5: Interact with the Contract**

1. Open the Truffle console:

truffle console --network besu

1. Run the following commands to interact with the contract:

let instance = await SimpleStorage.deployed();

await instance.set(42);

let value = await instance.get();

console.log("Stored Value:", value.toString());