# **🟢 E. Lawnmusk Coin — Full Gnosis Safe & Defender Workflow**

## **Overview**

This repository provides a "bulletproof" starter pack for deploying and managing the ELMCCoin, an ERC20 token on Binance Smart Chain. It integrates Hardhat for development, OpenZeppelin Defender for secure automation, and Gnosis Safe for multi-signature control of administrative actions.

## **✅ 1. Project Structure**

ELMCCoin-Project/

├── contracts/

│ └── ELMCCoin.sol # Your main token contract

├── scripts/

│ ├── deploy.js # Script to deploy ELMCCoin and create PancakeSwap pair

│ ├── transferOwnership.js # Script to transfer contract ownership to Gnosis Safe

│ └── defender/ # Scripts to be run via OpenZeppelin Defender Relayer

│ ├── enableTrading.js # Enable trading on the ELMCCoin contract

│ ├── burnLP.js # Burn LP tokens

│ └── rescueBNB.js # Rescue stuck BNB from the contract

├── defender-actions/ # Configuration files for OpenZeppelin Defender Actions

│ └── enableTrading.json # Defender Action to enable trading

├── hardhat.config.js # Hardhat configuration file

├── .env # Environment variables (KEEP PRIVATE!)

├── README.md # This guide

└── package.json # Project dependencies

## **✅ 2. Setup**

1. Clone the Repository:

git clone [YOUR\_REPO\_URL] ELMCCoin-Project

cd ELMCCoin-Project

1. Install Dependencies:

npm install

1. Configure .env:  
   Create a .env file in the root of your project based on the .env Example provided. Crucially, fill in all YOUR\_... placeholders.
   * PRIVATE\_KEY: The private key of your deployer wallet (funded with BNB for gas).
   * DEFENDER\_API\_KEY, DEFENDER\_API\_SECRET: Your OpenZeppelin Defender API credentials.
   * PANCAKESWAP\_ROUTER\_ADDRESS: The correct PancakeSwap Router V2 address for your target network (e.g., BSC Mainnet: 0x10ED43C718714eb63d5aA57B78B54704E256024E).
   * MARKETING\_WALLET\_ADDRESS, COMMUNITY\_WALLET\_ADDRESS, TEAM\_WALLET\_ADDRESS: Your designated wallet addresses for initial token distribution.
   * ELMC\_CONTRACT\_ADDRESS: Leave empty initially. You'll fill this after deployment.
   * PANCAKESWAP\_PAIR\_ADDRESS: Leave empty initially. You'll fill this after pair creation.
   * GNOSIS\_SAFE\_ADDRESS: The address of your Gnosis Safe (create one if you don't have one).
   * DEFENDER\_RELAYER\_ID: Your Defender Relayer ID (from Defender dashboard).
   * RESCUE\_BNB\_AMOUNT: Optional, for rescueBNB.js.
2. Configure hardhat.config.js:  
   The provided hardhat.config.js is set up. Ensure your accounts array in network configurations uses process.env.PRIVATE\_KEY.

## **✅ 3. Deployment Workflow**

Follow these steps carefully for a secure and controlled launch.

### **3.1. Deploy ELMCCoin**

This script deploys your ELMCCoin contract and automatically creates the PancakeSwap pair.

npx hardhat run scripts/deploy.js --network bscMainnet # or bscTestnet

* Output: The script will print the deployed ELMCCoin address and the PancakeSwap Pair address.
* Action: Update your .env file with ELMC\_CONTRACT\_ADDRESS and PANCAKESWAP\_PAIR\_ADDRESS immediately after this step.

### **3.2. Add Initial Liquidity (Manual)**

This is a CRITICAL manual step for a fair launch.

* Go to PancakeSwap's "Add Liquidity" UI on the correct network (Mainnet or Testnet).
* Select your deployed ELMC token (using ELMC\_CONTRACT\_ADDRESS) and BNB.
* Provide the desired amount of ELMC (from your deployer's remaining 60% balance) and an equivalent value of BNB.
* Confirm the transaction.
* Your deployer wallet will receive LP (Liquidity Provider) tokens.

### **3.3. Lock or Burn Liquidity (CRITICAL for Trust!)**

Immediately after adding liquidity, you MUST secure the LP tokens.

* Option A: Burn LP Tokens (Recommended for Meme Coins):  
  This script uses OpenZeppelin Defender to securely burn all LP tokens held by the ELMCCoin contract owner (which will be your Gnosis Safe after ownership transfer).
  + Set up Defender Relayer: In OpenZeppelin Defender, create a new Relayer and fund it with enough BNB for gas.
  + Set up Defender Action:
    - Go to Defender Actions.
    - Create a new Action.
    - Copy the content of scripts/defender/burnLP.js into the Action's code editor.
    - Configure the Action:
      * Name: Burn ELMC LP Tokens
      * Network: bsc-mainnet (or bsc-testnet)
      * Relayer: Select the Relayer you created.
      * Trigger: Manual.
      * Autotask: Enable.
    - Important: The script expects ELMC\_CONTRACT\_ADDRESS and PANCAKESWAP\_PAIR\_ADDRESS to be set in your .env file (which Defender will use if configured).
    - Run: Execute this Defender Action. It will require Gnosis Safe signatures if ownership has been transferred.
  + Alternative: Manual Burn (less secure for multi-sig):  
    If ownership is still with the EOA, you can run a script directly. (Not provided in this pack, Defender is preferred).
* Option B: Lock LP Tokens (via Third-Party Service):  
  Use a reputable liquidity locker service (e.g., Unicrypt, PinkSale, Team.Finance). Transfer your LP tokens to their locker contract for a long duration (e.g., 69 years). This is done via their platform's UI.

### **3.4. Enable Trading**

Once liquidity is added and secured (locked/burned), you can enable trading.

* Via Defender Action (Recommended):
  1. Import Action: Import defender-actions/enableTrading.json into OpenZeppelin Defender Actions.
  2. Configure: Update address and gnosisSafe.address placeholders in the JSON with your deployed ELMC contract address and Gnosis Safe address.
  3. Link Relayer: Link the Action to your Defender Relayer.
  4. Run: Execute this Defender Action. It will require Gnosis Safe signatures if ownership has been transferred.
* Via Script (if EOA is still owner):  
  If you haven't transferred ownership yet, you can call enableTrading() directly from your deployer EOA. (Not provided as a separate script, Defender is preferred).

### **3.5. Transfer Ownership to Gnosis Safe**

For enhanced security and decentralized control, transfer ownership of the ELMCCoin contract to your Gnosis Safe.

npx hardhat run scripts/transferOwnership.js --network bscMainnet # or bscTestnet

* Action: Update safeAddress and contractAddress in scripts/transferOwnership.js before running.
* Gnosis Safe: Ensure you have created and configured your Gnosis Safe (e.g., on app.safe.global).

## **✅ 4. Admin Actions (Controlled by Gnosis Safe via Defender)**

Once ownership is transferred to your Gnosis Safe, all onlyOwner functions will require multi-signature approval. You can automate and schedule these actions using OpenZeppelin Defender.

* enableTrading(): (One-time, after liquidity is secured)
* burnLPTokens(amount): To burn excess LP tokens or manage liquidity.
* pause() / unpause(): Emergency functions to halt/resume transfers.
* setFees(...): Adjust transaction taxes if needed.
* rescueStuckTokens(...) / rescueStuckBNB(...): Last resort functions for accidentally sent funds. (See scripts/defender/rescueBNB.js for an example).

## **✅ 5. Verify Your Contract**

After successful deployment, verify your contract on BscScan (or Testnet BscScan). This makes your code public and auditable.

npx hardhat verify --network bscMainnet 0xYourELMCAddress "routerAddress" "marketingWallet" "communityWallet" "teamWallet"

* Action: Replace placeholders with your actual deployed ELMC address and the constructor arguments used during deployment.

## **✅ 6. Done!**

Your ELMCCoin is now set up for a secure and controlled launch:

* Decentralized: Ownership can be managed by a multi-signature Gnosis Safe.
* Gated by Safe: All critical admin actions require multi-sig approval.
* Managed with Defender: Automation and secure execution of admin tasks.
* Public & Auditable: Contract code is verified on the blockchain explorer.
* Meme-Proof: Built-in anti-whale and fee mechanisms.

🚀 Happy Launching!