Hardhat 3 (TypeScript + Mocha + Ethers)

Usage instructions:

1. Launch the product via 1-click from AWS Marketplace. **Wait** until the instance status changes to 'Running' and passes all health checks. Then, connect to your instance using your Amazon private key and the '**ubuntu**' user."

To update software, use: **sudo apt update && sudo apt upgrade -y**

2) Verify toolchain

node -v # expect v22.x

npx hardhat --version # expect 3.x

3) Project location

cd /opt/hardhat

4) Quick start

npm run compile

npm test

5) Local development (recommended)

Terminal A - start a local JSON-RPC node (localhost only)

npm exec -- hardhat node --hostname 127.0.0.1 --port 8545

The node prints funded accounts + private keys (for local dev only).

6) Deploy options (choose ONE)

A) Standalone Ethers (simplest; no plugins)

Terminal B

Deploy Counter and print the address:

**LOCAL_PRIVATE_KEY=<copy a key from Terminal A> **

npm exec -- ts-node scripts/deploy-standalone.ts

Example output:

Counter deployed at: 0xABC...

Interact (increments x() to 6 total):

**LOCAL_PRIVATE_KEY=<same key> **

**COUNTER_ADDR=<0x... from deploy> **

npm exec -- ts-node scripts/poke-nonce.ts

Expect: x before: 0 → x after inc()+incBy(5): 6

B) Hardhat Ignition (repeatable deployments)

Terminal B

npx hardhat ignition deploy ./ignition/modules/Lock.ts --network localhost

C) Classic Hardhat script (advanced users)

If you have scripts/deploy.ts:

npx hardhat run --network localhost scripts/deploy.ts

7) Sepolia (optional testnet)

You only need this if you want to deploy to Sepolia.

Option 1 — .env file:

cd /opt/hardhat

cp .env.example .env

edit and add:

SEPOLIA_RPC_URL=...

SEPOLIA_PRIVATE_KEY=0x...

Option 2 — Hardhat keystore (encrypted on disk):

npx hardhat keystore set SEPOLIA_RPC_URL

npx hardhat keystore set SEPOLIA_PRIVATE_KEY

Deploy:

npx hardhat run --network sepolia scripts/deploy.ts

or with Ignition:

npx hardhat ignition deploy ./ignition/modules/Lock.ts --network sepolia

See for documentation: https://hardhat.org/docs/learn-more/whats-new

8) Notes & troubleshooting

• Local node is ephemeral — if you restart it, redeploy and use the new address.

• “Nonce too low / has already been used”: run the provided nonce-safe poke script, or restart the local node and redeploy.

• “Not a contract” or decode errors: you’re calling an address from a previous run; redeploy and use the fresh address.

• Security: keep port 8545 closed; only SSH (22/tcp) should be open and ideally restricted to your IP.

• The included Counter.sol exposes: x(), inc(), incBy(uint). Scripts show read/write patterns.

9) Where things live

```
/opt/hardhat
  ├─ contracts/              # Solidity contracts (Counter.sol, etc.)
  ├─ scripts/
  │   ├─ deploy-standalone.ts   # Deploy via ethers (no HRE)
  │   ├─ poke-nonce.ts          # Safe read+write example using NonceManager
  │   └─ deploy.ts (optional)   # Classic Hardhat script, if you prefer
  ├─ ignition/modules/Lock.ts   # Example Ignition module
  ├─ hardhat.config.ts          # Hardhat 3 config
  ├─ .env.example               # Example env for Sepolia (no secrets)
  └─ README-AMI.txt             # This quick-start as shipped
```

10) Security defaults

• SSH (22/tcp) only.

• Local RPC: 127.0.0.1:8545 (not exposed).

• No private keys are stored in the AMI. Only use keys printed by your local dev node or your own keys for testnets.

Happy building!

AWS Data

- Data Encryption Configuration: This solution does not encrypt data within the running instance.
- User Credentials are stored: /root/.ssh/authorized_keys & /home/ubuntu/.ssh/authorized_keys
- Monitor the health:
    - Navigate to your Amazon EC2 console and verify that you're in the correct region.
    - Choose Instance and select your launched instance.
    - Select the server to display your metadata page and choose the Status checks tab at the bottom of the page to review if your status checks passed or failed.

Extra Information: (Optional)

Allocate Elastic IP

To ensure that your instance **keeps its IP during restarts** that might happen, configure an Elastic IP. From the EC2 console:

1. Select ELASTIC IPs.
2. Click on the ALLOCATE ELASTIC IP ADDRESS.
3. Select the default (Amazon pool of IPv4 addresses) and click on ALLOCATE.
4. From the ACTIONS pull down, select ASSOCIATE ELASTIC IP ADDRESS.
5. In the box that comes up, note down the Elastic IP Address, which will be needed when you configure your DNS.
6. In the search box under INSTANCE, click and find your INSTANCE ID and then click ASSOCIATE.
7. Your instance now has an elastic IP associated with it.
8. For additional help: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/elastic-ip-addresses-eip.html