Rudalph Gonsalves
B. Tech Computer Engineering
Fr. Conceicao Rodrigues College of Engineering (Bandra, Mumbai)

HARDHAT

Smart Contracts in Web3

1. Smart Contracts are the building blocks of decentralized applications (dApps).
2. They are self-executing programs deployed on the blockchain.

Web2 Architecture Components

1. Frontend (HTML/CSS/JS or frameworks like React)
2. Backend (Node.js, Express.js, etc.)
3. Database (MongoDB, MySQL, Firebase, etc.)

Transitioning from Web2 to Web3

1. To transition a Web2 app into a Web3 app, we integrate smart contracts into the existing architecture.
2. This allows our Web2 frontend/backend to interact with the blockchain.

Need for Deployment

1. To allow interaction, the smart contract must be deployed to a blockchain network (testnet, mainnet, or local network).
2. Once deployed, the contract is live and can be interacted with using its address and ABI.

Methods of Smart Contract Deployment

1. Remix IDE	**2. Hardhat**
3. Truffle	4. Ganache (With Truffle or Hardhat)

Deployment Tools: Pros and Cons

REMIX	
Pros	Cons
1. Good for beginners and small contracts.	1. Difficult to integrate external libraries 2. Limited test Ether (depends on testnet faucets). 3. Low testing and debugging flexibility. 4. Not suitable for automation or complex deployment. 5. Not ideal for integrating into larger projects or pipelines.

HARDHAT
Pros
1. Provides a local blockchain network (`npx hardhat node`). 2. Easy integration of libraries and NPM packages. 3. Powerful debugging and error tracking. 4. Supports multi-network deployment (localhost, testnets, mainnet). 5. Scripting support for custom deployment logic.

Web3 Architecture Components

1. Frontend: HTML, CSS, JS, React, etc.
2. Backend: Node.js / Express.js (optional).
3. Hardhat Workspace:
    - Contracts (Solidity code)
    - Scripts (Deployment logic)
    - Artifacts (Compiled ABI & bytecode)

Setting Up Hardhat Project (3 Commands)

```
1. npm init -y                      # Initialize Node.js project
2. npm install --save-dev hardhat   # Install Hardhat
3. npx hardhat                      # Initialize Hardhat project
```

Rudalph Gonsalves
Linkedin: https://linkedin.com/in/rudalphgonsalves
Github: https://github.com/Rudalph
Portfolio: https://rudalph.vercel.app/

Key Project Folders

1. contracts/: Contains your .sol smart contracts (e.g., MyContract.sol)
2. scripts/deploy.js: (You create this.) Script to deploy your smart contract.
3. artifacts/: Generated after compilation. Contains:
    - ABI
    - Bytecode
    - Metadata used to interact with your contract.

Deployment Process (Step-by-Step)

Compile Contract	npx hardhat compile
Start Local Blockchain	npx hardhat node
Deploy Contract	npx hardhat run scripts/deploy.js --network localhost Deploys contract to local network. Displays contract address.
Artifacts Folder	Automatically created after deployment. Contains **ABI** and **bytecode** needed for Web3 integration.

Web2 ⇔ Web3 Integration

1. Use the contract address and ABI (from artifacts/) in your Web2 frontend or backend (via Web3.js or Ethers.js) to:
    - Read/write data
    - Trigger functions
    - Listen for events

Hardhat Test Accounts

1. You can hardcode one of the 20 private keys from Hardhat in your backend or script: To avoid wallet popups (useful in backend automation).
2. You can import the private key into MetaMask to simulate wallet transactions for testing with popup.

Rudalph Gonsalves
Linkedin: https://linkedin.com/in/rudalphgonsalves
Github: https://github.com/Rudalph
Portfolio: https://rudalph.vercel.app/