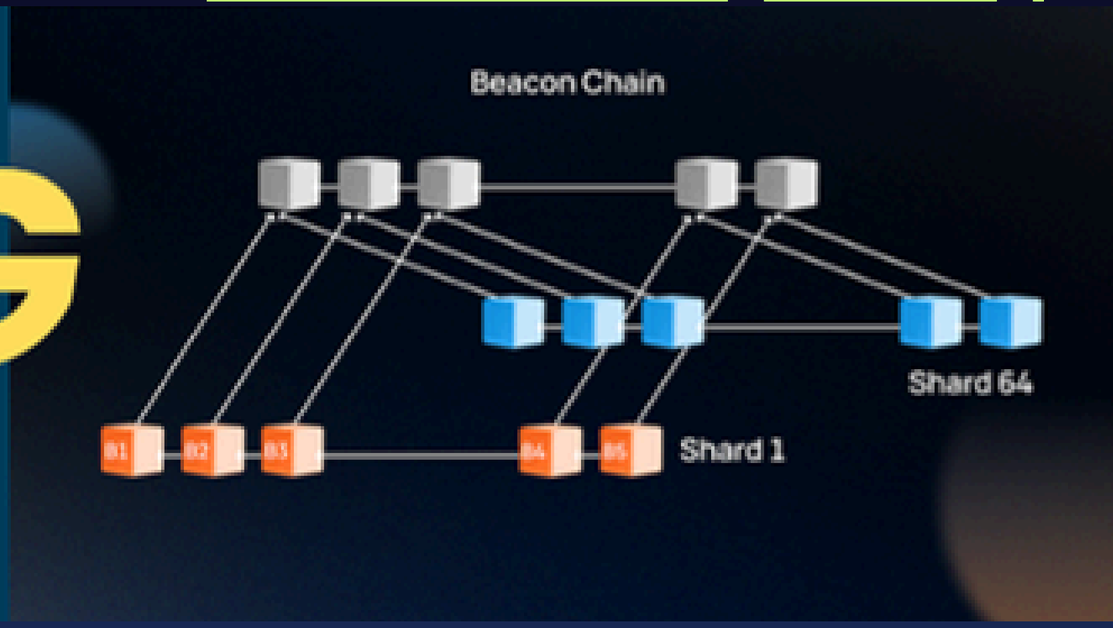
ETHEREUM 2.0 MASTERY PROGRAM

Instructor: Raja Rizwan Saleem

MODULE (6-10)

SHARDING
& ETHDO TOOLS



Raja Rizwan Saleem
Lead Blockchain Trainer

Class-3

METHODS OF ACCOUNT MANAGEMENT, TRANSACTION GENERATION AND SIGNING

Methods of Account Management, Transaction Generation and Signing

In a blockchain system, **account management**, **transaction generation**, and **signing** are critical processes that ensure secure, decentralized interaction between users and the blockchain network.

Methods of Account Management, Transaction Generation and Signing

1. Methods of Account Management

Account management refers to the creation, maintenance, and security of blockchain user accounts (or addresses). Each account typically controls a balance of tokens or holds information and smart contracts on the blockchain.

Methods of Account Management, Transaction Generation and Signing

1. Methods of Account Management

a. Externally Owned Accounts (EOA) vs. Contract Accounts

- **Externally Owned Accounts (EOA)**: Managed by users directly and secured with private keys. EOAs are the most common type of account used by individuals in blockchain systems like Ethereum.
- **Contract Accounts**: Represent smart contracts deployed on the blockchain and are controlled by their internal logic, not by a private key.

Methods of Account Management, Transaction Generation and Signing

1. **Methods of Account Management**

b. Private and Public Keys

- **Public Key**: Derives from the private key and is associated with the account address. It is used to receive assets.
- **Private Key**: Used to access the account and sign transactions. Protecting the private key is essential for maintaining account security.

Methods of Account Management, Transaction Generation and Signing

1. Methods of Account Management

c. Hierarchical Deterministic (HD) Wallets

HD wallets are a widely used method for managing multiple accounts securely and efficiently.

- **Master Seed**: HD wallets generate all private and public keys from a single master seed. This seed allows users to back up and restore multiple accounts with one phrase.
- **Key Derivation**: HD wallets use the BIP-32 standard to derive multiple public/private key pairs from the master seed, giving users control over numerous blockchain addresses while needing to back up only one seed.

Methods of Account Management, Transaction Generation and Signing

1. Methods of Account Management

d. Multi-Signature Accounts

A multi-signature (multi-sig) account requires more than one private key to authorize transactions, enhancing security.

- **Usage**: Multi-sig accounts are commonly used for organizations or joint accounts, where multiple parties must sign off on a transaction before it is executed.
- **Advantages**: Prevents unilateral actions by a single party, reducing the risk of theft if one private key is compromised.

Methods of Account Management, Transaction Generation and Signing

1. **Methods of Account Management**

e. Non-Custodial Wallets (Metamask, Trust wallet etc) - Decentralized wallets

- **Description**: Non-custodial wallets allow users full control over their private keys. Users are responsible for managing their own private keys and backups, giving them complete ownership of their blockchain assets.
- **Advantages**: Provides security and control, as no third party holds access to the private keys.

Methods of Account Management, Transaction Generation and Signing

1. Methods of Account Management

f. Custodial Wallets

- **Description**: Custodial wallets are managed by a third party (e.g., exchanges or wallet services), which stores the private keys on behalf of the users.
- **Advantages**: Users don’t need to manage private keys directly, making it easier for less experienced users. However, users have to trust the custodian with their assets.

ACCOUNT MANAGEMENT

Account Management with reference to Eth. 2.O

1. **Validator Accounts:**
    - **Staking to Become a Validator:** In Ethereum 2.0, users manage accounts by staking 32 ETH to become validators. Validators are responsible for securing the network by proposing and validating new blocks. Managing this account involves maintaining an active validator status by adhering to the network's rules, avoiding slashing penalties (for malicious behavior or inactivity), and collecting rewards for participating correctly.
    - **Validator Keys:** Validators need two types of keys:
        - **Withdrawal Key:** This key manages the funds that can be withdrawn after the validator has exited.
        - **Signing Key:** This key is used for signing attestations and blocks. It should be kept online.

Account Management

2. Eth1 and Eth2 Accounts:

- **Eth1 Account for Deposits:** The initial deposit (32 ETH) to become a validator comes from an Ethereum 1.0 account, which is managed through the Ethereum 1.0 (Eth1) chain. Users manage their existing Eth1 accounts through standard wallets like MetaMask, Ledger, etc.
- **Eth2 Staking Interface:** After the deposit, the management of validator accounts shifts to the Ethereum 2.0 network. Interfaces like BeaconChain explorers or staking services help track and manage these accounts.

Account Management

3. Staking Services:

- **Custodial Staking Services:** Some users prefer using third-party custodial services for account management. These services manage staking and validator keys on behalf of the user, reducing the complexity of self-managing validator accounts.
- **Non-Custodial Staking:** Users can also opt for decentralized and non-custodial staking services, retaining full control of their keys while leveraging a service to assist with the staking process.

Account Management

4. Slashing and Penalties:

- **Self-Management of Validator Performance:** Validators need to manage their accounts by keeping their nodes online and operating correctly. If they go offline or engage in malicious activities, they risk being slashed, which leads to penalties. Monitoring tools and alerts can be set up to help manage this process.

Account Management

5. Withdrawal Management:

- **Account Exit and Withdrawal Process:** After a validator exits, the withdrawal of staked ETH (including rewards) needs to be managed. Users must keep track of their withdrawal credentials and manage the process of transferring funds from their validator account back to their Eth1 account.

Github

GitHub

GitHub is a web-based platform used for storing, sharing, and collaborating on code using a system called Git. It's essentially a cloud-based repository where developers can store their projects, track changes, and work together on the same codebase

What is GitHub?

- A web-based platform for version control using Git.
- Hosts code, manages projects, and enables collaboration.

Why Use GitHub?

- - Version control
- - Collaboration
- - Open-source community
- - Project management

Why Use GitHub?

Version Control:

GitHub uses Git, a version control system that allows developers to track every change made to a project. This means you can revert to previous versions, see who made changes, and compare different versions of the code

Why Use GitHub?

Collaboration:

GitHub facilitates collaboration by allowing multiple developers to work on the same project simultaneously. They can create branches of the code, work on them independently, and then merge their changes into the main codebase

Why Use GitHub?

Storing and Sharing:

GitHub provides a central location for storing code, making it easy to share projects with others, whether they are individual developers or entire teams.

Code Hosting:

GitHub is a code hosting platform, meaning it provides the infrastructure for storing, managing, and accessing code repositories.

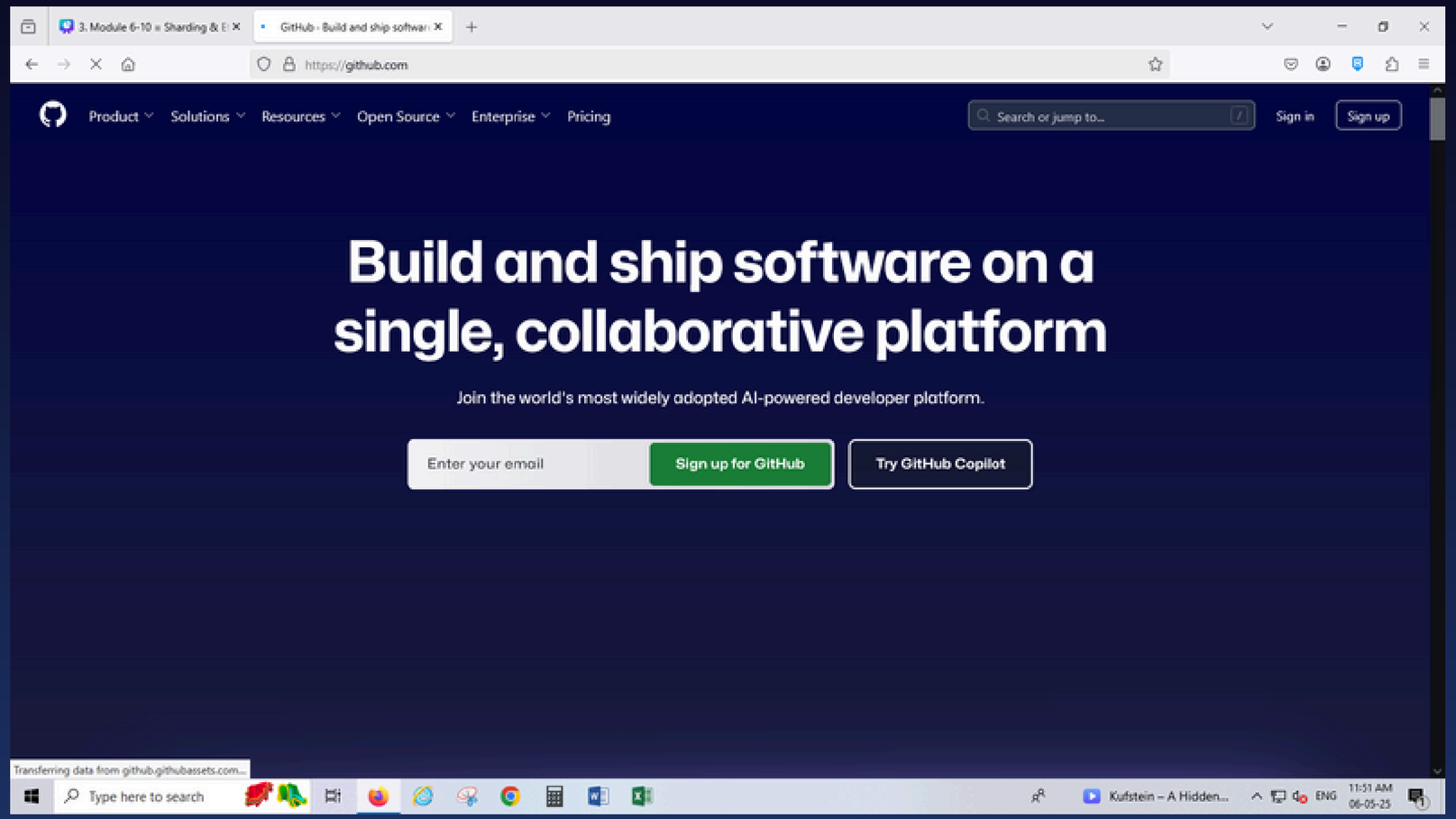
Beyond Code:

While primarily used for code, GitHub can also be used to store other files, documentation, and project resources.

Setting Up GitHub

- 1. Create a GitHub account
- 2. Install Git on your system
- 3. Configure Git:
- git config --global user.name "Your Name"
- git config --global user.email "you@example.com"

Setting Up GitHub



Creating a Repository

- 1. Click 'New' in GitHub Repos
- 2. Name your repository
- 3. Initialize with README (optional)

Cloning a Repository

- Command:
- git clone https://github.com/username/repo.git
- Creates a local copy of the remote repo.

Making Changes

- 1. Edit files locally
- 2. Use commands:
- git add .
- git commit -m "Your message"
- git push

Branching and Merging

- - Use branches for new features
- - Command:
- git checkout -b new-feature
- - Merge using Pull Requests

Understanding Pull Requests

- - Way to propose changes
- - Enables review and discussion before merging

Issues and Bug Tracking

- - Create and manage issues
- - Assign labels, milestones, and collaborators

Project Boards

- - Visualize progress using Kanban-style boards
- - Track tasks and issues

Actions and Automation

- - Automate workflows using GitHub Actions
- - CI/CD pipelines, testing, deployment

GitHub Pages

- - Host static websites from GitHub repos
- - Free and easy deployment

Forking Repositories

- - Copy of a repo under your account
- - Use to contribute to others’ projects

Watching, Starring, and Following

- - Watch: Notifications for changes
- - Star: Bookmark for later
- - Follow: Updates from developers

Managing Collaborators

- - Invite team members to your repo
- - Set roles and permissions

Security Features

- - Dependabot alerts
- - Branch protection rules
- - Secrets management

Integrations

- - Connect with tools like Slack, VS Code, Trello, Jenkins

GitHub Desktop & CLI

- - GitHub Desktop: GUI for GitHub
- - GitHub CLI: Command-line access to GitHub features

Summary & Resources

- GitHub is essential for modern development.
- Docs: https://docs.github.com
- Learning Lab: https://lab.github.com

Development Tools

Mastering the Hardhat Development Environment

- A Developer's Toolkit for Ethereum Smart Contract Development

Introduction to Hardhat

- Hardhat is a development environment to compile, deploy, test, and debug Ethereum software. Built with JavaScript and designed for EVM-based chains.

Why Use Hardhat?

- - Developer-friendly and flexible
- - Built-in local blockchain (Hardhat Network)
- - Better error messages and stack traces
- - Powerful plugins ecosystem

Prerequisites

- - Node.js and npm
- - Basic JavaScript and Solidity
- - IDE like VS Code
- - Terminal/CLI knowledge

Installing Hardhat

- mkdir hardhat-project && cd hardhat-project
- npm init -y
- npm install --save-dev hardhat

Setting Up a Hardhat Project

- npx hardhat
- Choose template
- Project structure includes contracts/, scripts/, test/, hardhat.config.js

Project Structure Explained

- - contracts/: Solidity contracts
- - scripts/: Deployment scripts
- - test/: Test files
- - hardhat.config.js: Configuration

Writing a Simple Contract

- contract HelloWorld {
- string public greet = "Hello, World!";
- }

Compiling Contracts

- npx hardhat compile
- Output: artifacts/ folder with ABI and bytecode

Deploying Contracts

- Example deploy script using ethers.getContractFactory and deploy()
- npx hardhat run scripts/deploy.js --network localhost

Hardhat Network

- Local Ethereum node
- Auto mines transactions
- Pre-funded accounts
- Fast and resettable

Testing with Hardhat

- Use Mocha/Chai
- Example test checks the greeting from HelloWorld contract

Using Ethers.js

- Integrated by default
- Handles wallet, provider, and contract interactions

Debugging with Hardhat

- Detailed stack traces
- console.log in Solidity via hardhat/console.sol

Hardhat Plugins

- Popular plugins:
- - hardhat-ethers
- - hardhat-waffle
- - hardhat-gas-reporter
- - hardhat-etherscan

Deploying to Testnets

- Add network config
- Use Infura/Alchemy
- Deploy with: npx hardhat run --network goerli

Integrating Frontend

- Use Ethers.js to connect UI to smart contracts
- Import ABI and contract address

CI/CD with Hardhat

- Automate tests and deployment
- Use GitHub Actions or Jenkins

Hardhat vs Truffle

- Hardhat: faster, better debugging, flexible plugins
- Truffle: integrated suite (Ganache, Drizzle)

Summary & Resources

- Docs: https://hardhat.org
- GitHub: github.com/NomicFoundation/hardhat

THANK-YOU