email : techcenter@trainocate.com.my
contact : +603-2281-1111

BTA-HEDB: ETHEREUM TRAINING: HANDS-ON ETHEREUM DEVELOPMENT BOOTCAMP

Course Code: BTA-HEDB
Duration: 3 days
Instructor-led Training (ILT) | Virtual Instructor-led Training (VILT)

OVERVIEW

This 3 day instructor-led course is designed for programmers and developers who want to take a comprehensive deep dive in writing smart contracts and building applications that interact with them. This course provides detailed overviews of Ethereum, smart contracts, and the development language, Solidity.

The student will be exposed to Ethereum's adaptable feature set which allows the developer to design decentralized applications for countless applications. Students will also participate in hands-on programming lab sessions to learn, develop, and advance their skills in Ethereum development.

The Ethereum Development Training Course is designed for those seeking an in-depth understanding and development experience of the Ethereum Blockchain platform. Students will participate in approximately 50% programming lab time providing practical experience, enhancing their knowledge and existing skill set. Due to the technical programming lab content covered in this course, it is not recommended for those without programming knowledge and experience.

SKILLS COVERED

- An excellent overall understanding of the Ethereum architecture and Solidity language.
- All functional components (including smart contracts) required to develop an Ethereum Blockchain.
- The understanding of how to instantiate an Ethereum application on the network.
- An in-depth understanding of how transactions are created and implemented on an Ethereum network.

WHO SHOULD ATTEND?

Target Course Audience Include:

- Programmers
- Application Developers
- System Architects
- Network Architects
- Network Security Architects
- IT Professionals w/programming experience

PRE-REQUISITES

- Absolutely Necessary: Basic knowledge in JavaScript / HTML
- Advantageous: Basic knowledge in C ++ / Java, data types
- Advantageous: Basic knowledge with git repositories

MODULES

Module 1: Blockchain and Smart Contract Basics

- What is Blockchain and how does it work?
- Centralized vs. Decentralized vs. Distributed
- Blockchain vs. Databases
- Bitcoin vs Ethereum
- What are Smart Contracts?
- How are Smart Contracts used?

Module 2: Smart Contract Programming Basics

- Advantages and Drawbacks of Smart Contracts
- Layer 1 vs. Layer 2
- High-Level Language vs. Low-Level
- Languages in Comparison Solidity, Vyper, others
- Smart Contracts with Solidity
- The Layout of a Solidity File
- LAB TASKS (Lab 1)
    - Types of Variables in Solidity
    - Function/Variable Visibility
    - Smart Contract Constructors
    - Setter- and Getter-Functions

Module 3: Understanding Decentralized Information and Web3

- Blockchain Access structures and Architectures
    - Remote Blockchain Nodes vs. Local Blockchain Nodes
- Blockchain Access vs. centralized RESTful API
- Understanding Web3.js API
- Understanding Transactions and Consensus
- Private Keys, Public Keys and Signatures
- Understanding privacy on pubic Blockchains
- Understanding the architecture of KeyStore's such as MetaMask or MIST
- LAB TASKS (Lab 2 – Ropsten Test-Ether and MetaMask)
    - Installing and Configuring MetaMask
    - Obtaining Ropsten (or Testnet) Ether
    - Tracing Ether through Block-Explorers
    - Understanding Infura

Module 4: Basics of Ethereum and the EVM

- Ethereum Denominations
- Understanding EVM and the ABI Interface
- Calls vs. Transactions
- Concurrency and Events
- Use cases of Events
- LAB TASKS (Lab 3 Web3JS Operations + Lab 4 Events)
    - Install and Use Ganache
    - Work with Web3.js
    - Work with Infura
    - Define Events
    - Listen and React to Events

Module 5: Solidity Advanced: Modifiers, Mappings, Structs and Inheritance

- Understanding Functions, Mappings and Structs
- When to use Modifiers
- Libraries vs. Inheritance
- LAB TASKS (Lab 5 Modifiers, 6 Mappings and Structs, Lab 7 Inheritance)
    - Understand and use Modifiers
    - Add Mappings and Structs
    - Use Inheritance to increase auditability

Module 6: Understanding Deployment and Costs

- Understand Development and Deployment Cycles
- Understanding Solidity Compilation and Deployment
- Gas and Gas-Costs
- Upgradeability and Data Migration Techniques
- Understand the moving Parts: Compiler, Blockchain, API, KeyStore
- LAB TASKS (Lab 8 Deployment, Lab 9 Tie it together)

  - Changing from strings to bytes (save gas)
  - Deploy using Ropsten Test-Ether
  - Understand the difference when using a real Blockchain
  - Use a fully functioning distributed Application

Module 7: Mining, Proof of Work vs. Proof of Authority

- What is Mining in PoW?
  - How blocks are generated
  - PoW vs. PoA (vs. PoS)
- Understanding Go-Ethereum or Ganache/TestRPC for local development
- Understanding Private Blockchains vs. Public Blockchains
- LAB TASKS (Lab 10 Mining)
  - Installing and using Ganache
  - Installing and using Go-Ethereum
  - Connecting to Ganache/Go-Ethereum from Remix and Web3.js
  - Interact with the Blockchain from HTML/JS

Module 8: Current Problems, Solutions, Outlook, Serenity

- Ethereum Now and Ethereum Future
- Where we are at with Ethereum
  - Homestead
  - PoW
- Where Ethereum is heading to
  - Serenity
  - PoS + PoW to PoS
  - Sharding
- Recommended Newsletters/Groups

Module 9: Working in Teams, Testing and Versioning

- Understanding what Truffle is
- Comparison to Embark
- How Manage Code for Teams
- Understanding Migrations
- Understanding Unit-Testing with Truffle
- LAB TASKS (Lab 11 Truffle setup, Lab 12 Truffle Unit Testing)
  - Download and Setup Truffle
  - Adapt the standard Truffle-Project
  - Write A Unit Test

Module 10: IPFS and distributed File-Storage

- What is IPFS
- Comparison between IPFS, FileCoin, Swarm, Sia, Storj
- LAB TASK (Lab 13 – IPFS)
  - Install and work with IPFS
  - Upload and retrieve a fully decentralized file

Bonus Module: Compilation

- LAB TASK (Lab 14 – Compilation)
  - Compile a Solidity File with a command-line compiler

END OF PAGE