BTDS Practicals - Step-by-Step Guide

Practical 1: Ethereum Environment Setup

- Use Git Bash (Windows) or terminal (Linux/Mac).
- Run: node -v and npm -v to check Node.js.
- Install Truffle: npm install -g truffle
- Install Ganache: npm install -g ganache
- Create project: mkdir MyEthereumProject && cd MyEthereumProject && truffle init
- Launch Ganache: ganache (CLI) or open GUI.
- In truffle-config.js, set development network port to 7545.

Practical 2: Simple Smart Contract Deployment

- File: SimpleStorage.sol goes in contracts/
- File: 2_deploy_simple_storage.js goes in migrations/
- Run in terminal: truffle compile
- Start Ganache (on port 7545), then run: truffle migrate --network development
- Run truffle console: truffle console --network development
- Interact:

```
const instance = await SimpleStorage.deployed();
await instance.set(42);
const value = await instance.get();
console.log(value.toString()); // Output: 42
```

Practical 3: Hyperledger Fabric Setup

- Go to fabric-samples/test-network
- Run: ./network.sh up createChannel -c mychannel -ca
- Run: ./network.sh deployCC -ccn basic -ccp ../asset-transfer-basic/chaincode-go -ccl go
- Use Docker CLI: docker exec -it cli bash
- Query: peer chaincode query -C mychannel -n basic -c '{"Args":["GetAllAssets"]}'
- Shutdown: ./network.sh down

Practical 4: IPFS Data Storage

- Install IPFS (go-ipfs), then run:

```
ipfs init
ipfs daemon
```

- To add a file: ipfs add example.txt
- Note the CID. To retrieve: ipfs cat <CID> > output.txt
- JS file uses IPFS API (via ipfs-http-client) to upload and retrieve content.

BTDS Practicals - Step-by-Step Guide

Practical 5: Data Integrity on Blockchain

- Deploy DataIntegrity.sol via Truffle.
- Compile & migrate using truffle CLI.
- registerDataHash.js reads file (e.g. dataset.csv), hashes it (SHA-256), and sends the hash to the blockchain.
- Output: transaction hash if successful.

Practical 6: Data Provenance Tracking

- Deploy DataProvenance.sol via Truffle.
- From console:

```
const instance = await DataProvenance.deployed();
const id = web3.utils.sha3("dataset1.csv");
await instance.recordEvent(id, "UPLOAD", "First version");
const event = await instance.getProvenanceEvent(id, 0);
```

- Output: Provenance history of each data ID.

Practical 7: Decentralized Marketplace

- Deploy DataMarketplace.sol via Truffle.
- Add data listing: await contract.listData("QmCID", priceWei)
- Buy listing: await contract.buyData(id, { value: price })
- Output: Transaction logs, ownership change.

Practical 8: Fabric Chaincode Deployment

- Place chaincode.go in asset-transfer-basic
- Run: ./network.sh up createChannel -c mychannel -ca
- Deploy: ./network.sh deployCC -ccn asset_transfer -ccp ../asset-transfer-basic -ccl go
- Query: peer chaincode query -C mychannel -n asset_transfer -c '{"function":"ReadAsset","Args":["asset1"]}'

Practical 9: Blockchain Voting System

- Deploy Voting.sol via Truffle.
- In console:

```
await contract.addCandidate("Alice")
await contract.vote(1, { from: user })
const count = await contract.getVoteCount(1);
console.log(count.toString());
```

- Output: Number of votes per candidate.

BTDS Practicals - Step-by-Step Guide

Practical 10: IoT and Blockchain

- Deploy IoTDataRegistry.sol via Truffle.
- Use registerIoTData.js script:
  - Hash sensor file
  - Register hash on blockchain
- Output: Transaction hash proving IoT data was recorded securely.