Stellar DeFi Workshop

A comprehensive educational workshop demonstrating Stellar blockchain development with DeFi protocols including Soroswap and DeFindex integrations.

Presentation

Overview

This repository contains complete workshops that demonstrate DeFi operations on the Stellar blockchain and Soroban smart contract platform:

1. **Soroswap Workshop** (`src/soroswap.ts`) - DEX integration and token swapping
2. **DeFindex Workshop** (`src/defindex.ts`) - Vault creation and asset management
3. **Direct Contract Calls** (`src/soroswap-typescript.ts`, `src/defindex-typescript.ts`) - Same flows without SDKs, using direct Stellar SDK contract calls

The workshops demonstrate professional DeFi development patterns using official SDKs on Stellar testnet. The direct contract call versions show how to interact with contracts at the lowest level without SDK/API dependencies.

What You'll Learn

Soroswap Workshop

- Creating and funding wallets on Stellar testnet
- Using the Soroswap SDK for DEX operations
- Swapping tokens (XLM to USDC) through Soroswap
- Quote → Build → Sign → Submit workflow for trading
- Adding liquidity to decentralized exchange pools
- Checking token balances via Soroban contracts
- Integration with Soroban smart contracts

PROF

DeFindex Workshop

- Creating multi-asset vaults with configurable strategies
- Understanding vault roles (Emergency Manager, Fee Receiver, Manager, Rebalance Manager)
- Deploying vaults with initial deposits
- Managing vault permissions and fees
- Making deposits to existing vaults
- Working with asset management strategies
- Using the DeFindex SDK for vault operations

Prerequisites

- Node.js 18+ installed
- TypeScript knowledge

- Basic understanding of blockchain and DeFi concepts
- Stellar testnet account (automatically created in workshops)

Installation

1. Clone the repository:

```
git clone https://github.com/paltalabs/stellar-workshop.git
cd stellar-workshop
```

2. Install dependencies:

```
npm install
```

3. Create a `.env` file with API keys (optional but recommended):

```
cp .env.example .env
# Edit .env and add your API keys:
# SOROSWAP_API_KEY=your_soroswap_api_key
# DEFINDEX_API_KEY=your_defindex_api_key
```

Usage

Run the Soroswap Workshop

```
npm run soroswap
```

PROF

This workshop demonstrates:

- Wallet creation and funding via Friendbot
- Token swapping (XLM to USDC) using Soroswap DEX
- Checking token balances
- Adding liquidity to trading pools

Run the DeFindex Workshop

```
npm run defindex
```

This workshop demonstrates:

- Vault manager wallet creation

- Configuring vault parameters with roles and strategies
- Creating a DeFindex vault with initial deposit
- Creating additional depositor wallets
- Making deposits to existing vaults

Run Direct Contract Call Workshops (No SDKs)

Soroswap Direct (`src/soroswap-typescript.ts`):

```
npm run soroswap-direct
```

DeFindex Direct (`src/defindex-typescript.ts`):

```
DEFINDEX_FACTORY=your_factory_address npm run defindex-direct
```

These versions execute the same flows but interact directly with smart contracts using only the Stellar SDK—no Soroswap/DeFindex SDKs or APIs. Perfect for understanding contract interactions at the lowest level and building custom transaction logic.

Workshop Structure

Soroswap Workshop Flow

1. **Wallet Creation**: Generate keypair and fund with testnet XLM
2. **Token Swap**: Swap XLM for USDC using Soroswap SDK
3. **Balance Check**: Query token balances via Soroban contracts
4. **Add Liquidity**: Provide liquidity to XLM/USDC pool

DeFindex Workshop Flow

PROF

1. **Vault Manager Setup**: Create and fund vault manager wallet
2. **Configure Vault**: Define roles, fees, assets, and strategies
3. **Create Vault**: Deploy vault with initial deposit
4. **Create Depositor**: Generate separate depositor wallet
5. **Make Deposit**: Deposit additional funds to the vault

Key Features

- **Educational Focus**: Comprehensive logging and step-by-step explanations
- **Production SDKs**: Uses official Stellar, Soroswap, and DeFindex SDKs
- **Testnet Safe**: All operations run on Stellar testnet
- **Real-world Patterns**: Demonstrates professional DeFi development workflows
- **Type-Safe**: Full TypeScript implementation with proper error handling

Technical Architecture

TypeScript Dependencies

- `@stellar/stellar-sdk` (v14.3.2): Official Stellar SDK for blockchain operations
- `@soroswap/sdk` (v0.3.8): Soroswap SDK for DEX operations
- `@defindex/sdk` (v0.1.1): DeFindex SDK for vault management
- `dotenv`: Environment variable management
- `typescript`: Type-safe development

Rust Smart Contracts

The repository includes Soroban smart contracts written in Rust:

- **DeFindex Zap Contract**: Advanced vault interaction contract
- **Soroswap Integration Contracts**: DEX router interfaces
- Built with `soroban-sdk` v23.0.2
- Uses OpenZeppelin Stellar contracts for security patterns

Network Configuration

- **Horizon Server**: `https://horizon-testnet.stellar.org`
- **Soroban RPC**: `https://soroban-testnet.stellar.org`
- **Soroswap API**: `https://api.soroswap.finance`
- **Network**: Stellar Testnet

Project Structure

```
stellar-workshop/
├── src/
│   ├── soroswap.ts           # Soroswap DEX workshop (uses SDK)
│   ├── defindex.ts           # DeFindex vault workshop (uses SDK)
│   ├── soroswap-typescript.ts   # Direct contract calls (no SDK)
│   └── defindex-typescript.ts   # Direct contract calls (no SDK)
├── contracts/           # Soroban smart contracts (Rust)
│   ├── defindex-zap/   # DeFindex integration contract
│   ├── soroswap-auth/  # Soroswap authorization contract
│   └── soroswap-simple/# Simple Soroswap integration
├── scripts/            # Deployment and utility scripts
└── Cargo.toml         # Rust workspace configuration
```

PROF

Educational Outcomes

By completing these workshops, you will understand:

1. **Stellar Blockchain**: Account creation, funding, and transaction management
2. **DeFi Protocols**: DEX trading, liquidity provision, and vault management
3. **Soroban Smart Contracts**: Interacting with deployed contracts on Stellar
4. **SDK Integration**: Using production-grade SDKs for DeFi development
5. **Transaction Workflows**: Quote → Build → Sign → Submit patterns

6. **Asset Management**: Vault strategies, roles, and permissions
7. **Best Practices**: Error handling, transaction signing, and security

Smart Contract Development

Building Contracts

```
# Build all Soroban contracts
make build

# Or build with Cargo
cargo build --target wasm32-unknown-unknown --release
```

Deploying Contracts

The `scripts/deploy/` directory contains deployment scripts for:

- DeFindex vault interactions
- Soroswap router integrations
- Custom token implementations

Security Notes

- All operations use testnet XLM (no real value)
- Private keys are generated randomly for each workshop run
- Never commit private keys or secrets to version control
- API keys should be stored in `.env` file (not committed)
- Transactions include proper timeout and fee settings
- Error handling demonstrates production patterns

Troubleshooting

PROF

Common issues and solutions:

1. **Network Issues**: Ensure stable internet connection to Stellar testnet
2. **Rate Limiting**: Wait between workshop runs if hitting Friendbot/API limits
3. **Transaction Failures**: Check Stellar Expert for transaction details
4. **SDK Errors**: Verify API keys in `.env` file
5. **Balance Errors**: Ensure wallet has sufficient XLM for transactions and fees
6. **Contract Errors**: Check Soroban RPC connection and contract IDs
7. **Build Errors**: Ensure Rust toolchain is installed for contract compilation

Resources

Official Documentation

- Stellar Documentation - Core Stellar blockchain docs

- Soroban Documentation - Smart contract platform
- Soroswap Documentation - DEX protocol docs
- DeFindex Documentation - Vault management docs

Tools & Explorers

- Stellar Expert - Testnet blockchain explorer
- Stellar Laboratory - Transaction builder and tester
- Soroswap Interface - DEX trading interface

SDKs & Libraries

- @stellar/stellar-sdk - Official JavaScript SDK
- @soroswap/sdk - Soroswap integration SDK
- @defindex/sdk - DeFindex vault SDK
- soroban-sdk - Rust smart contract SDK

Contributing

This is an educational workshop maintained by Palta Labs. Contributions are welcome!

To contribute:

1. Fork the repository
2. Create a feature branch
3. Make your changes
4. Submit a pull request

For bugs or feature requests, please create an issue.

License

Apache-2.0 License - See LICENSE file for details

PROF

Acknowledgments

Built by Palta Labs for the Stellar developer community.

Ready to start? Run `npm install` and then `npm run soroswap` to begin your Stellar DeFi journey!