Liquidity Timelock contract

How to test?

```
bash scripts/quickstart.sh standalone
```

in another terminal

```
bash scripts/run.sh
```

first build the contract

```
cd liquidity_timelock
make build

# to test run
make test
```

Audit With Scout

```
cd /workspace/liquidity_timelock
cargo scout-audit
```

1.- Deployment

PROF

1.- Be sure to have your secrets

```
cp .env.example .env
```

Set your `ADMIN_SECRET_KEY` and `MAINNET_RPC_URL`.

2.- Set your `configs.json Specifically, youll need to set,l for your desired network:` soroswap_router, end_timestamp, xlm_address, usdc_address`.

3.- Deploy and initialize the Timelock contract, with the desired Soroswap Router and End Timestamp

```
# install dependencies
yarn
```

```
# to deploy the contract
yarn deploy testnet liquidity_timelock
```

Or in Mainnet

```
yarn deploy mainnet liquidity_timelock
```

2.- Publish Deployed Addresses and Timestamps

Publish the deployed contract into your `public` folder, together with the admin account, soroswap router address and end_timestamp that was used when intializing the contract

```
yarn publish_addresses testnet
```

3.- Add Liquidity

0.- Use the same account defined on the `ADMIN_SECRET_KEY`
1.- Swap XLM for USDC in your favourite AMM or DEX.
2.- Be sure to have 100 XLM extra, because our script will keep 100 XLM
3.- Check on https://app.soroswap.finance/liquidity that youll be able to provide liquidity with all your USDC.
Dont worry, if not all tokens will be used, they will be returned to your wallet.

4.- Provide Liquidity using the Timelock Contract

```
yarn add_xlm_usd_liquidity testnet liquidity_timelock
```

PROF

4.- Claim and Remove Liquidity

When the time defined in end_timestamp has passed, recover your liquidity with

```
yarn remove_xlm_usd_liquidity testnet liquidity_timelock
```

This will
1.- claim the LP tokens
2.- withdraw all the liquidity of those LP tokens

If time as not passed, yet, you will get a `Error(Contract, #906)` error