NFTY Lending Final Release

Overview

We arrive at the final release of the agreed features and components related to NFTY finance. This components include:

- Smart Contracts: A set of smart contracts which implements all the required features targeting Ethereum blockchain.
- Backend: used mainly as cache and to process all the offchain flows.
- dApp: frontend used as a client to interact with the platform.

Code

The released version of the code is located in the DEVELOP branch, ready to be merged to MAIN by NFTY's team at their convenience.

- **Smart Contract:**
  - Repository: https://github.com/NFTYLabs/nftyfinance-contracts
  - Commit Hash: 9f528e9e1fdac2bc2646fa568901b5e21806559b
- **Backend:**
  - Repository: https://github.com/NFTYLabs/nftyfinance-backend
  - Commit Hash: 28a9512b3130c4eb9830ec217268bc63aaf9614f
- **dApp:**
  - Repository: https://github.com/NFTYLabs/nftyfinance-dapp
  - Commit Hash: 0a7bea5606c61e62103a9c70cb05348c2fd613fb

Documentation

- Smart contracts:
  - All the smart contracts have comments describing each of the different parts (structures, functions, etc).
  - Class diagram: see attachment named "NFTY class diagrams.drawio.pdf"
  - Flow diagram: see attachment named "NFTY flow diagrams.drawio.pdf"
- Backend: The endpoints are documented using swagger syntax

Platform configuration

The NFTY finance platform has different parameters which can affect the protocol behavior.

Platform fees:

Total fee: 1% (in MainNet this fee will be charged in NFTY tokens, since NFTY token doesn't exist on TestNet we will use a dummy ERC20 compliant token). This fee is paid by the borrower when he requests a loan.

Distribution of the fees (how this fee is distributed):

- Lender: 10%
- Borrower: 30% (in the case the borrower paid the loan back in time, in other case it goes to the platform)
- Platform: 60%

All the above values can be changed by the owner of the contracts.

Oracle prices expiration time:

The maximum amount of time, in seconds, that can pass between the last time oracle prices were updated, after that time the smart contract will consider the price invalid and will not be able to calculate the fees to be paid in NFTY token by a loan.
In MainNet we recommend to have a price expiration time in the order of a few minutes.
Due to a change in the business rules (as a consequence of the security audit), the function which allows to change this time in the smart contract has changed to reject any time bigger than 24 hours.

NFTY will have to assure the price in the oracle contract is updated with the frequency selected. If this does not happen (and the price in the oracle contract gets old) any attempt to create or accept a new loan will be rejected.

Whitelisted ERC20s:

The ERC20 to be used as liquidity by the lender in their shops MUST be whitelisted in the platform by the administrator. In this TestNet deployment we recommend whitelist at least the official USDC token:

- USDC Goerli contract address: 0x07865c6e87b9f70255377e024ace6630c1eaa37f

To assure the correct functioning of the protocol, the whitelisted ERC20 must implement the decimals and symbol functions.

Backend refresh time

The backend works a cache for the dApp and to process all the offchain flows.
The refresh rates used for the events listener on the backend should be taken carefully. We suggest using around 30 seconds. Remember this may be the minimum time required to a dApp user to see some action impacted on the frontend (for example, display a new liquidity shop after a successful creation).

Parameters that CAN'T be changed after the deployment

The following parameters CAN'T be changed after the deployment of the contracts, in order to change this a new deployment will be required:

- NFTY token address
- Smart NFT addresses (obligation and promissory)
- Oracle address
- Loan expiration durations. Right now these durations are hardcoded in the code and are specified as: 30, 60 and 90 days as requested by NFTY team.

Preconditions

NFTs

- The token must be an ERC721 compliant
- The tokenURI can't be empty. tokenURI should return an URI for each minted token.

If the tokens whitelisted in the platform do not meet the preconditions it can cause errors in the system.

Addresses

NFTY TOKEN contract address:

NFTY's team will be in charge of the deployment

ERC20 used for liquidity contract address:

NFTY's team will whitelist the ERC20 tokens to be used as liquidity

NFTs whitelisted collections addresses:

NFTY's team will whitelist the collection

NFTY LENDING contract:

NFTY's team will be in charge of the deployment

dApp & backend

NFTY's team will be in charge of the deployment

Comments and Suggestions

- We recommend designing a high availability infrastructure to avoid service interruptions in the dApp and/or backend. This should include but is not limited to:
    - Redundancy
    - Load balancing
    - Backups and restore policies
    - Auto-scaling
    - Etc.
- Infrastructure security is very important, we suggest taking care of this aspect in the infrastructure layer. Examples:
    - Use of bastion
    - DMZ
    - Isolation of resources
    - Software updates
    - Etc.
- Monitoring: this is a crucial aspect in any platform, there should be some monitoring policy in different components:
    - Infrastructure
    - Smart contracts
    - Oracles
    - Etc.
- For MainNet deployment we recommend transferring the ownership of the contracts to a multisig with timelock (at least) or a DAO (the best).
- The backend needs access to an Ethereum Full Node in order to work correctly. In this case we suggest NFTY to run their own node. In case NFTY decides to use a node provider like Infura (which we don't recommend), it has to be sure to hire a layer which supports the daily volume of requests required by the backend. In case the node rejects requests by the backend, the dApp and the backend can start to behave incorrectly.