finance.vote

Christopher Smith, Nick Almond, Naved Ali

Nov 02, 2022

CONTENTS:

mint is a merkle root based eligibility system for NFT issuance and price discovery.

CHAPTER

ONE

SETTING UP

1.1 Frontend

Repository link: Mint

1. In the influence terminal:

```
git checkout develop
```

2. Install dependencies:

```
npm install
```

3. Run application:

```
npm start
```

1.2 Backend

Repository link: Mint backend

In the minting-backend repo:

```
nvm use
npm i
docker-compose up --build
```

CHAPTER

TWO

DEPLOYING SMART CONTRACTS ON TESTNET

To set up a mint, several contracts are needed to be deployed in the right order.

1. MerkleIdentity.sol:

```
address _mgmt atribute -> creator wallet address
```

2. VoterID.sol:

```
ooner atribute -> creator wallet address
minter atribute -> MerkleIdentity contract address
nomen atribute -> name of your choice
symbowl atribute -> symbol of your choice
```

3. AmaluEligibility.sol or MerkleEligibility.sol:

```
AmaluEligibility:
_mgmt atribute -> creator wallet address
_gatemaster atribute -> MerkleIdentity contract address

MerkleEligibility:
_gatemaster atribute -> MerkleIdentity contract address
```

4. AmaluPriceGate.sol or FixedPricePassThruGate.sol or SpeedBumpPriceGate.sol

Attention: **NFT** address = **VoterId** contract address

GATE address = **MerkleIdentity** contract address

CHAPTER

THREE

CREATE NEW MINT

3.1 Form type

1. Series

   Form to create another tree under an already existing mint. Choose that type if you need to add at least 2 new addresses to the whitelist, new NFT series or grant another minting permissions (eligibility) to users in the tree.

2. Mint

   Form to create a brand new mint. Choose that type if you want to create minting for a whole new NFT address.

To set up a mint or to add series you have to fill in the tokenID (numbers) and uri (for example, ipfs links with metadata containing all NFT attributes and graphics). Each tokenID must be an unique number and have its own uri link. Uri link could be duplicated, provided metadata for all tokens is the same.

3.2 Mint type

1. Selectable

   NFTs aren't minted in an order, user can choose what number wants to mint

2. Sequential

   NFTs are minted in an order, user cannot choose freely the number

3.3 Eligibility type

1. Amalu

   Everyone are allowed to mint. It's an open mint. No addresses list needed.

2. Merkle

   Only users in the tree are eligible to mint. Tree is calculated from the uploaded addresses list.

3. Whitelist

   -

4. Tokenmapping

   -

3.4 Price type

1. Amalu

   Price is always 0. It's a free mint.

2. Fixed

   NFT's price is set.

3. Speedbump

   NFT's price changes with interest. The more frequently NFTs are minted, the higher the price. Price increases with each mint, and decays in time.

   - price decay parameter - price drop multiplier
   - price increase parameter - price increase multiplier
   - price denominator parameter - time in seconds that specifies how quickly the price should decrease

3.5 Mint configures

1. unboxing video

   Displaying after the minting transaction successfully passes. Once the unboxing video box is checked, there is an additional option that specifies whether the video should be played for the minter only or for everyone who has the minting page open during the minting process.

2. placeholder image

   An image that displays above the minting component when 'show next media' is unchecked.

3. tokens externally hosted

   If token uri are hosted on external server (e.g. on ipfs), this box needs to be chcecked.

4. show next media

   If the user may see upfront what token will be minted, this box should be checked. This option shouldn't be used in case the NFT has a predefined rarity.

5. reveal

   Decide whether minted NFT should be shown after successful minting or not.

6. show traffic lights

   Three lights that show the current state of the token (**green** - available to mint, **yellow** - transaction pending, token reserved, **red** - token unavailable)

7. show progress bar

   Line under the Mint button with a live view of the minting progress.

8. stages based

   Option allowing to split minting into stages and to decide on the strategy of that.

   - sold out - the next stage opens once the previous one sold out
   - timebased - the next stage opens at a specific time

- simultaneous - the next stage opens either on the previous stage sold out or at a specific time (whichever comes first)

3.6 Set up mint steps

For testing purposes, it may be needed to deploy the contracts first.

1. Select chain.
2. Choose 'mint' on the switch.
3. Pass the addresses (whitelist) and token metadata (tokenID and uri).
4. Create Group (key) - it's the address of the mint (i.e. **https://mint.factorydao.org/Group(key)**).
5. Pass VoterID contract address as NFT address.
6. Pass MerkleIdentity contract address as Gate address.
7. Select mint type (Sequential or Selectable).
8. Select the start date and time.
9. Select the eligibility type (Amalu or Merkle).
10. Pass AmaluEligibility or MerkleEligibility contract addresses as Eligibility address.
11. Decide on maximum withdrawals per address and maximum total withdrawals. The latter must always be bigger than the first, regardless of the whitelist size.
12. Pass the eligibility gate index or uncheck the box if you don't have one.
13. Choose price type (Amalu, Fixed or Speedbump).
14. Pass AmaluPriceGate or FixedPricePassThruGate or SpeedBumpPriceGate contract address as the Price address.
    - for a fixed price, pass the price gate index if it exists, or uncheck the box and pass the beneficiary wallet address that will gain the interest from the minting process.
    - for speedbump, pass the price gate index if it exists, or uncheck the box and pass price decay, price increase, price denominator and beneficiary wallet address that will gain the interest from the minting process.
15. Specify token range (what is the tokenID range, e.g. 1-100)
16. Upload placeholder image (File type), or cloudflare video ID (cloudflare type).
17. Decide on minting options:
    - unboxing video - if yes, pass the cloudflare video ID and decide if unboxing will be streamable for the minter or for everyone (if for everyone, check the box),
    - check the 'Tokens externally hosted' box if tokens are hosted on ipfs,
    - optionally choose reveal, traffic lights, progress bar boxes,
    - decide if mint will be split into stages and choose stages strategy.

3.7 Add series steps

1. Select chain.
2. Choose ‘series’ on the switch.
3. Select Group key you will be adding to from the dropdown.
4. Pass the addresses (whitelist) and token metadata (tokenID and uri).
5. Select the start date and time.
6. Select the eligibility type (Amalu or Merkle).
7. Pass AmaluEligibility or MerkleEligibility contract addresses as Eligibility address.
8. Decide on maximum withdrawals per address and maximum total withdrawals. The latter must always be bigger than the first, regardless of the whitelist size.
9. Pass the eligibility gate index or uncheck the box if you don’t have one.
10. Choose price type (Amalu, Fixed or Speedbump).
11. Pass AmaluPriceGate or FixedPricePassThruGate or SpeedBumpPriceGate contract address as the Price address.
    - for a fixed price, pass the price gate index if it exists, or uncheck the box and pass the beneficiary wallet address that will gain the interest from the minting process.
    - for speedbump, pass the price gate index if it exists, or uncheck the box and pass price decay, price increase, price denominator and beneficiary wallet address that will gain the interest from the minting process.
12. Specify token range (what is the tokenID range, e.g. 101-170)

CHAPTER

FOUR

1 OF 1 AUCTION SETTING UP

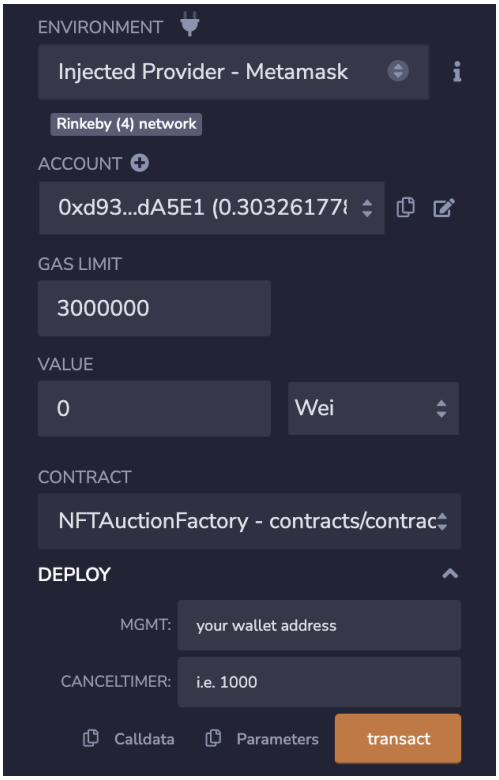
Steps:

```
1. contracts deploy
2. setting up auction
3. bidding
4. concluding auction
```

4.1 Contracts deploy

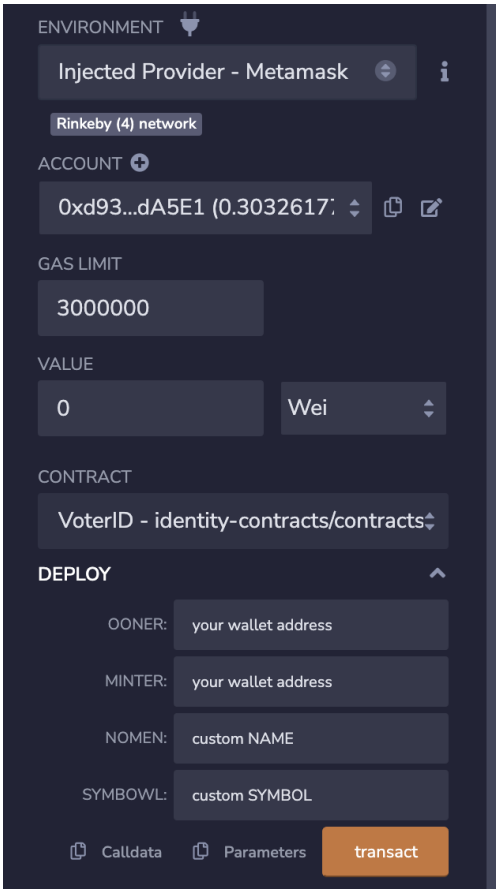
Within remix set up two contracts:

1. *finance.vote/contracts/contracts/auction/NFTAuctionFactory.sol*
    - compile
    - deploy with settings:
        - Envirnoment: Injected Provider - Metamask on Rinkeby
        - Deploy/MGMT: your wallet addres
        - Deploy/CANCELTIMER: i.e. 1000



- hit 'Transact' button and confirm transaction via Metamask
- open tx hash transaction and copy **contract address** and **block number**

2. *finance.vote/identity-contracts/contracts/VoterID.sol*
   - compile
   - deploy with settings:
     - Envirnoment: Injected Provider - Metamask on Rinkeby
     - Deploy/OONER: your wallet address
     - Deploy/MINTER: your wallet address
     - Deploy/NOMEN: custom name
     - Deploy/SYMBOWL: custom symbol name



- hit 'Transact' button and confirm transaction via Metamask

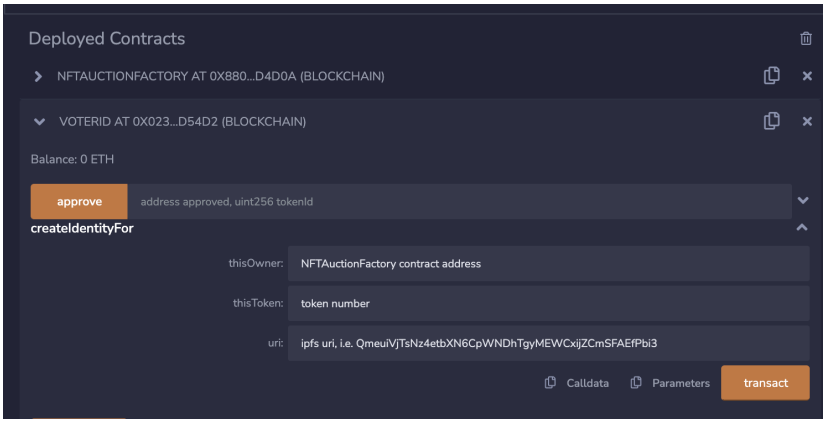
3. In the *./factory-dao-mint/src/contracts/NFTAuctionFactory.json* file change lines:
   - 386 with deployed NFTAuctionFactory contract address
   - 387 with deployed NFTAuctionFactory contract block number

```
384   "networks": {
385     "4": {
386       "address": "0x8804924504fc68215A7cBc956e1f659Cbb0D4D0A",
387       "fromBlock": 11292114
388     }
389   }
```

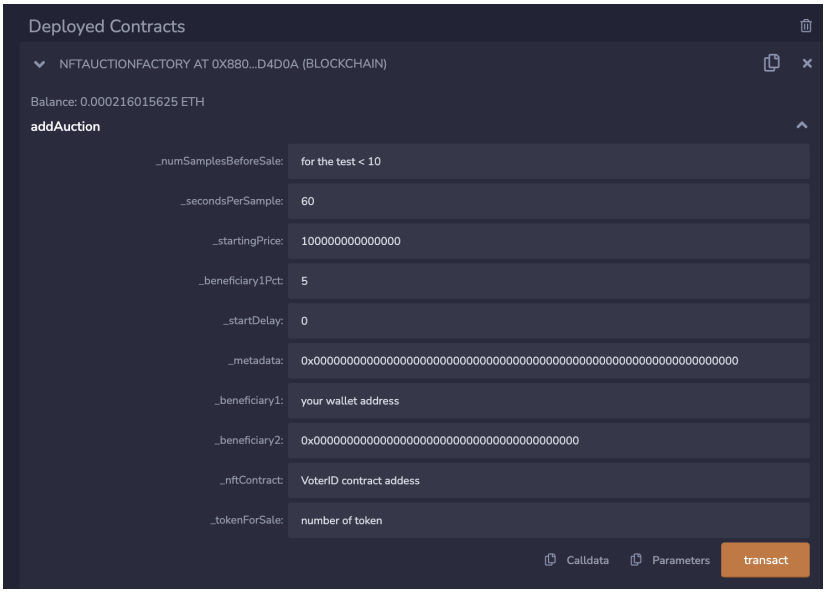
4.2 Set up auction

1. Load both contracts into the remix

2. In the VoterID fill in **createIdentityFor** field:
    - thisOwner - **NFTAuctionFactory contract address**
    - thisToken - token number, for example **1**
    - uri - token ipfs uri, for example **QmeuiVjTsNz4etbXN6CpWNDhTgyMEWCxijZCmSFAEfPbi3**



3. In the NFTAuctionFactory in **addAuction** fill in those fields:
    - _numSamplesBeforeSale - for test purposes put here **number < 10**
    - _secondsPerSample - recommended **60**
    - _startingPrice - set in Wei, for example **100000000000000**
    - _beneficiary1Pct - recommended **5**
    - _startDelay - recommended **0**
    - _metadata - **0x0000000000000000000000000000000000000000000000000000000000000000**
    - _beneficiary1 - your wallet address
    - _beneficiary2 - **0x0000000000000000000000000000000000000000**
    - _nftContract - **VoterID contract addess**
    - _tokenForSale - number of token that will be selled in the auction, for example **1** (same as *thisToken* in the previous step)



4. In the NFTAuctionFactory hit **startAuction**

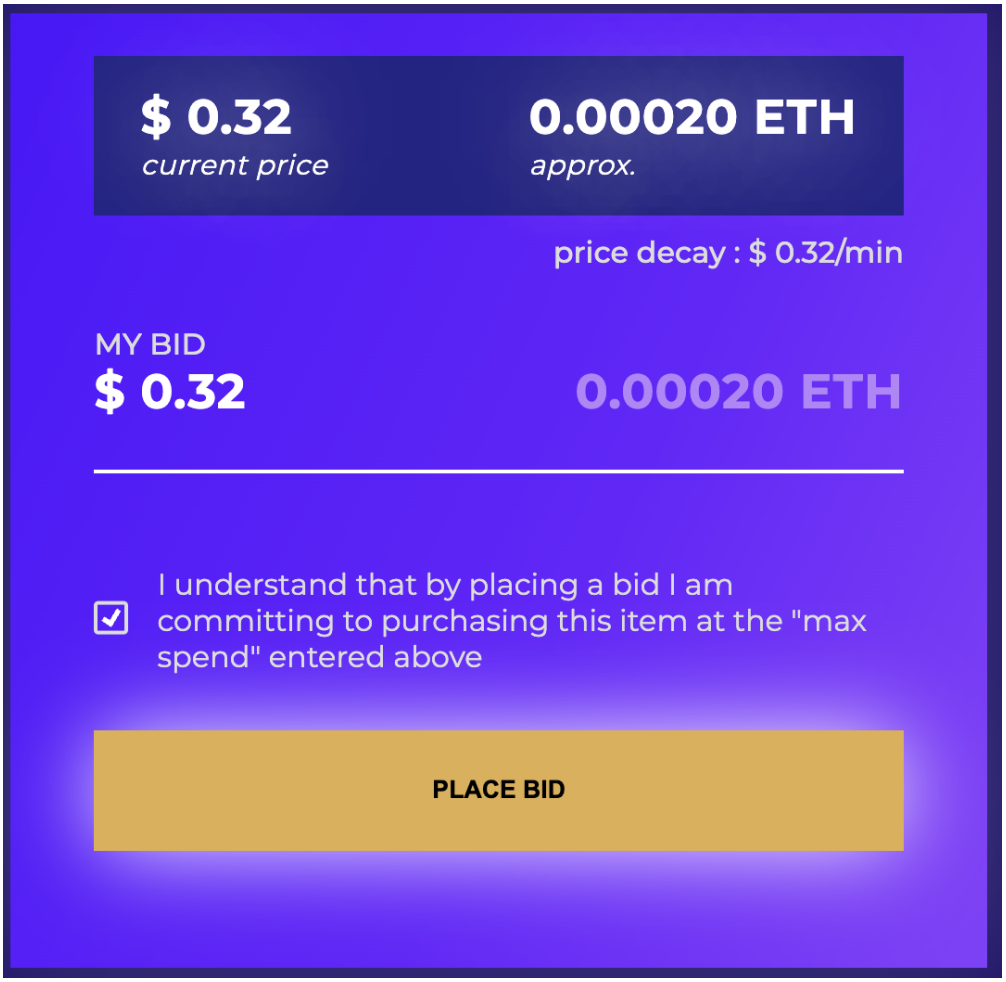
4.3 Bidding

Go to the **localhost:3000/oneofone** and place a bid.
If everything was set up correctly, you should be able to see your NFT on the main page.

Check box and hit 'place bid' button.

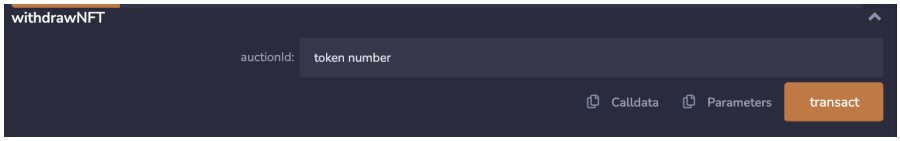
Accept transaction in Metamask.

After time set in the NFTAuctionFactory contract the auction will end:
(time of auction running = `_numSamplesBeforeSale` * `_secondsPerSample`)



4.4 Concluding auction

In **NFTAuctionFactory** contract hit `concludeAuction` then `withdrawNFT`.



Auction is finished, you shouldn't be able to place a bid anymore.

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

CHAPTER

SIX

OUR PRODUCTS

- bank
- influence
- launch
- markets
- yield

6.1 Back to main page

- Home