Build a Dapp on Optimism

And conquer L2 Bridging

Emily Lin

Developer Evangelist, Truffle @ ConsenSys

Disclaimer: I do not work for or represent the opinions of Optimism

I just think the project is super cool 😎
And Truffle X Optimism may be the most ambitious crossover of all time

Agenda

Optimistic rollups

- L1s vs. L2s
- What is an optimistic rollup?

Bridges

- What is a bridge?
- Bridge hacks

Contract walkthrough

- ICrossDomainMessenger
- Standard Token Bridge

Let's build a DApp!

- Set up
- Unbox the Truffle box
- Build a marketplace
- Add Bridget the bridge widget!

Section 1

Optimistic roll ups

A crash course

Let's talk L1s and L2s

What is layer 1?

Layer 1 (L1) is the **underlying foundation and base blockchain** that various layer 2 (L2) networks build on top of. For example, **Ethereum is an L1** that is comprised of node operators to secure and validate the network, block producers, the history of transaction data, the consensus mechanism the blockchain itself and

What is layer 2?

L2 is a **separate blockchain that increases transaction speed and throughput**, while fully or partially **deriving its security from Ethereum**. Additionally, Layer 2 projects **rely on Ethereum for data availability** by posting their transaction data onto Ethereum.

source: https://ethereum.org/

Why should we care?

The blockchain trilemma

The blockchain trilemma is a **“pick 2” situation between decentralization, security, and scalability**. Ethereum’s growth has greatly affected scalability: it can only process ~15 transactions/sec. Between the 3, the **Ethereum community has chosen decentralization and security**. **Sacrificing scalability means increasing gas prices, which jeopardizes the adoption of Ethereum** by pricing out users.

Layer 2 to the rescue!

As the demand to use Ethereum grows, the network becomes congested. L2s **decrease L1 congestion by bundling transactions to be submitted to Ethereum**, thus increasing scalability while **inheriting Ethereum’s data availability, security, and decentralization**.

source: https://ethereum.org/

What is an optimistic rollup?

Rollups "**roll up" and execute hundreds of transactions outside of the L1 into a single transaction that is then submitted to the L1**. Rollups come in 2 flavors: optimistic and zero-knowledge.

With **optimistic** rollups….

- Transactions are **assumed to be valid** (feeling optimistic 😊) and **don't publish proofs of validity**
- Before publishing, there's a **challenge period** - if a transaction is **suspected to be invalid, a fault proof is ran** to see if this has taken place

Whereas with **zero-knowledge** rollups…

- Transactions **are published with proofs of validity**

source: https://ethereum.org/

Section 2

Bridges

Another crash course

What is a bridge?

Bridges **connect 2 blockchain ecosystems** (i.e., L1 to L1, L2 to L1, L1 to L2) by enabling….

- **Cross-chain transfer** of **assets and information**
- Dapps and users to **take the "pros"** of various blockchains and platforms to **alleviate the "cons"**
- **L1 contracts can trigger functions on L2 contracts** and **vice versa**

A user locks assets on L1 and receives equivalent assets on L2. When bridging back to L1, the assets are burned on L2, releasing the locked assets to the user on L1.

NOTE: A set bridge standard does not yet exist!

source: https://ethereum.org/

OH NO, A BRIDGE HACK!

"**$2 billion in cryptocurrency has been stolen** across 13 separate cross-chain bridge hacks, the majority of which was stolen this year. **Attacks on bridges account for 69% of total funds stolen in 2022** so far. " - **Aug 2, 2022**

(and literally the BSC bridge recently RIP)

OH MY,WHY??

There's not an established bridge design, but because users typically transfer funds to a bridge protocol, which are then locked into the contract, **bridges lock up a lot of liquidity, acting as a prime centralized point for attack**.

Ser, how fix?

- **Current processes identify and authenticate target and source blockchains by chainId** - **ensure that this is checked during the crosschain message** (recent BSC hack!!)
- **Safeguard a contract's private key** through a Hardware Security Module (HSM), Key Management System (KMS), hardware wallet, offline wallet, or secure vault technology, or share ownership through a multi-signature wallet
- Strengthen **security for the web layer**, which is centralized

So many more…
https://entethalliance.github.io/crosschain-interoperability/crosschainsecurityguidelines.html

source: https://entethalliance.github.io/

My hero 😗

Optimism has **provided contracts and an SDK** that abstract away the need for us to write our own bridge contracts. These deployed contracts are what **allows us to do the cross-chain transfer** of data and assets.

Writing a bridge contract can be **complex and requires thorough security auditing**. That's why we'll be **utilizing Optimism's provided bridge contracts instead of writing our own**.

Section 3

Contract walkthroughs

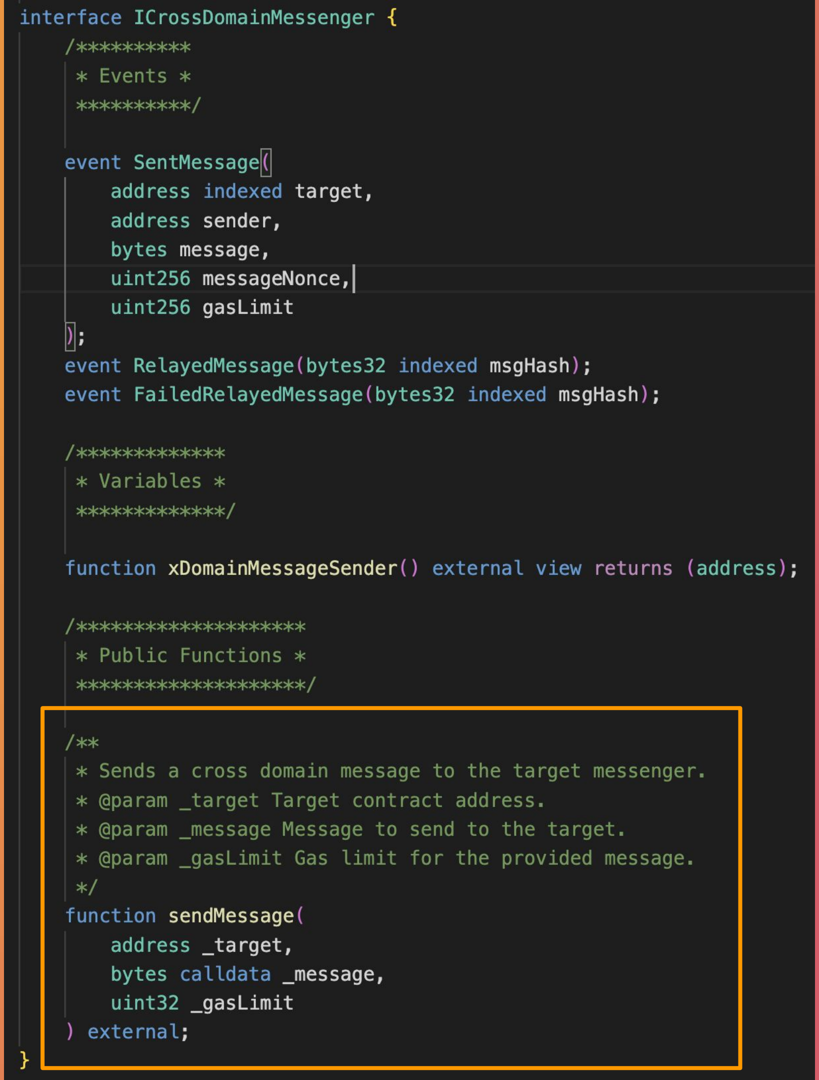
One more crash course 😱

Transfer data

To transfer data, each layer has its own messenger contract that is responsible for calling the other contract's function. The main points are…

- There are two pre-deployed contracts: **L1CrossDomainMessenger.sol** and **L2CrossDomainMessenger.sol**
- They are pre-deployed, and you can find their addresses in the Optimism monorepo: https://github.com/ethereum-optimism/optimism/tree/develop/packages/contracts/deployments

source: https://community.optimism.io/docs/developers/bridge/messaging/

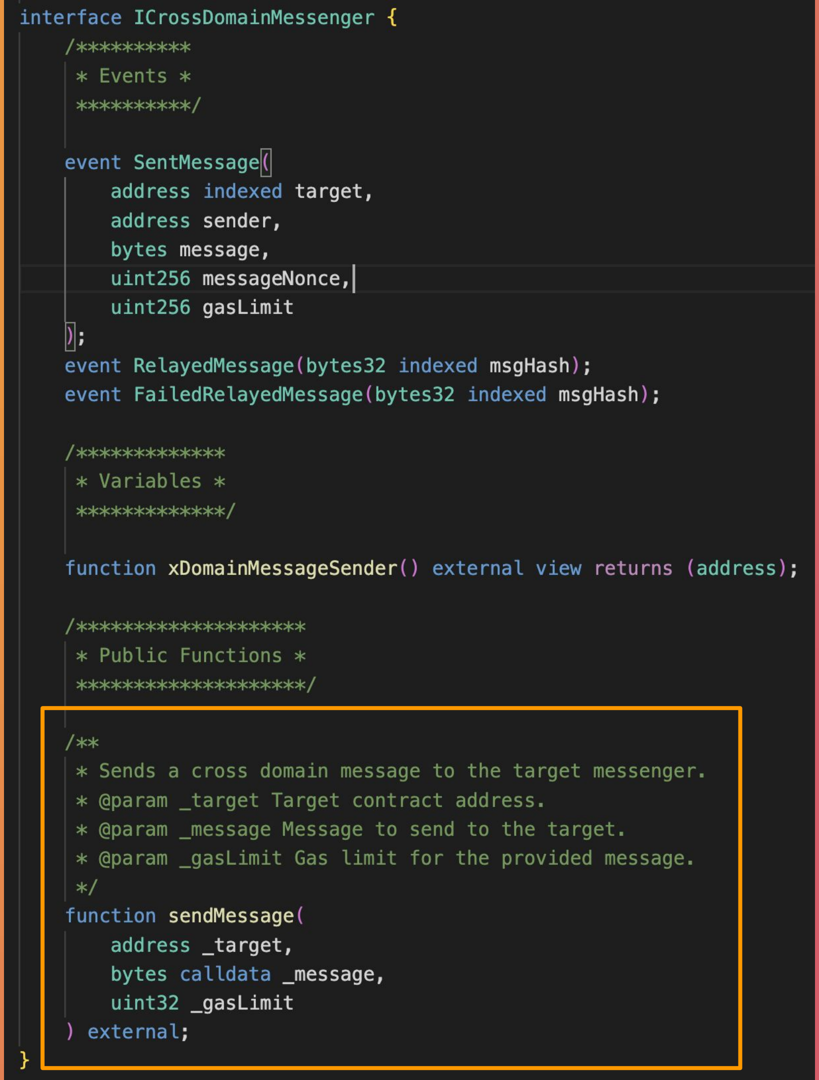

```
interface ICrossDomainMessenger {
    /**********
     * Events *
     **********/

    event SentMessage(
        address indexed target,
        address sender,
        bytes message,
        uint256 messageNonce,
        uint256 gasLimit
    );
    event RelayedMessage(bytes32 indexed msgHash);
    event FailedRelayedMessage(bytes32 indexed msgHash);

    /*************
     * Variables *
     *************/

    function xDomainMessageSender() external view returns (address);

    /********************
     * Public Functions *
     ********************/

    /**
     * Sends a cross domain message to the target messenger.
     * @param _target Target contract address.
     * @param _message Message to send to the target.
     * @param _gasLimit Gas limit for the provided message.
     */
    function sendMessage(
        address _target,
        bytes calldata _message,
        uint32 _gasLimit
    ) external;
}
```

sendMessage

- Allows us to call an L1 contract's function from the L2 contract and vice versa

- To call a function on an Optimism contract from Ethereum:
    - _target is the Optimism contract address
    - _message is the encoded version of the function and inputs
    - _gasLimit is how much gas you are willing to pay

source: https://community.optimism.io/docs/developers/bridge/messaging/

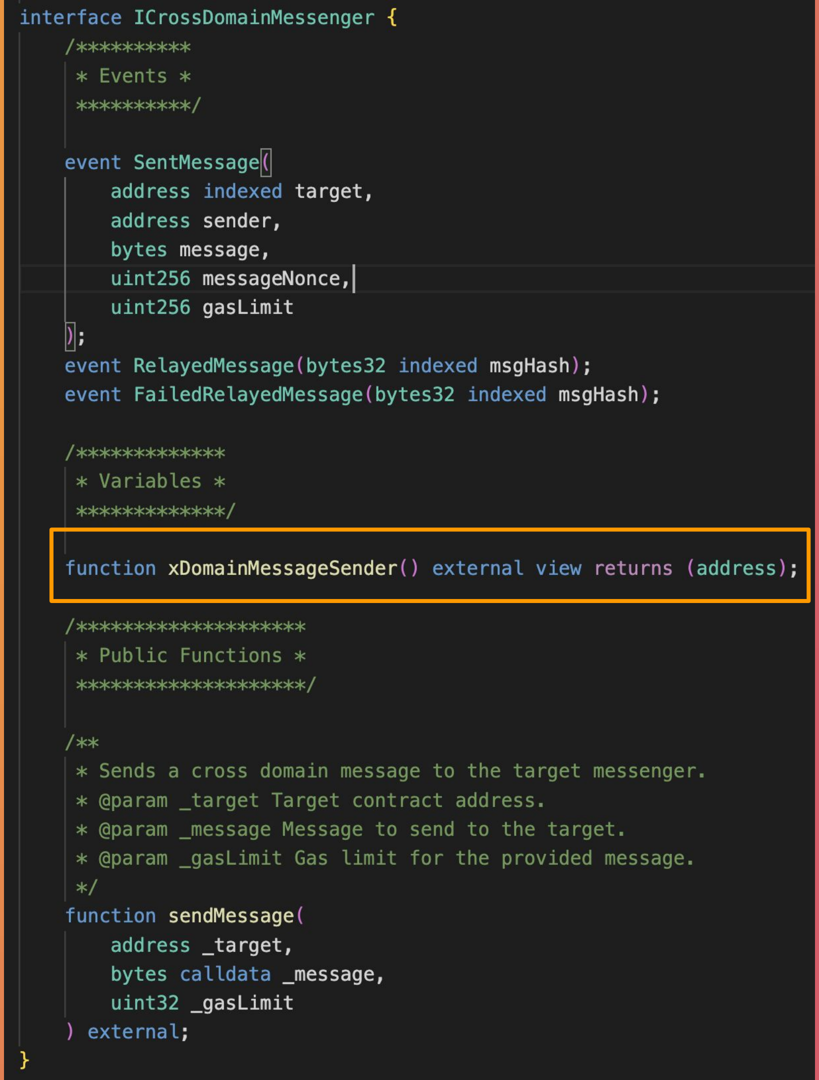


```
interface ICrossDomainMessenger {
    /**********
     * Events *
     **********/

    event SentMessage(
        address indexed target,
        address sender,
        bytes message,
        uint256 messageNonce,
        uint256 gasLimit
    );
    event RelayedMessage(bytes32 indexed msgHash);
    event FailedRelayedMessage(bytes32 indexed msgHash);

    /*************
     * Variables *
     *************/

    function xDomainMessageSender() external view returns (address);

    /********************
     * Public Functions *
     ********************/

    /**
     * Sends a cross domain message to the target messenger.
     * @param _target Target contract address.
     * @param _message Message to send to the target.
     * @param _gasLimit Gas limit for the provided message.
     */
    function sendMessage(
        address _target,
        bytes calldata _message,
        uint32 _gasLimit
    ) external;
}
```

sendMessage

- _gasLimit fees explained
    - L1 -> L2 requires L2 gas
        - First 1.92 million L2 gas is free
        - If you need more, specify on _gasLimit at a 1:32 ratio
        - BUT if you specify 80k over 1.92 million, but spend less than 1.92 million, you still have to pay 80k/32 = 2500 L1 gas
    - L2 -> L1 requires 2 transactions
        - L2 fee to initiate transaction
        - L1 fee to finalize transaction

source: https://community.optimism.io/docs/developers/bridge/messaging/



```
interface ICrossDomainMessenger {
    /**********
     * Events *
     **********/

    event SentMessage(
        address indexed target,
        address sender,
        bytes message,
        uint256 messageNonce,
        uint256 gasLimit
    );
    event RelayedMessage(bytes32 indexed msgHash);
    event FailedRelayedMessage(bytes32 indexed msgHash);

    /*************
     * Variables *
     *************/

    function xDomainMessageSender() external view returns (address);

    /********************
     * Public Functions *
     ********************/

    /**
     * Sends a cross domain message to the target messenger.
     * @param _target Target contract address.
     * @param _message Message to send to the target.
     * @param _gasLimit Gas limit for the provided message.
     */
    function sendMessage(
        address _target,
        bytes calldata _message,
        uint32 _gasLimit
    ) external;
}
```

xDomainMessageSender

- Accessing msg.sender will return the calling contract, which is the messenger contract
- xDomainMessengerSender allows us to identify the “actual” contract who is calling the function (i.e. Optimism contract using the messenger contract to call Ethereum contract function)

source: https://community.optimism.io/docs/developers/bridge/messaging/

Transfer ETH and ERC20s

Transferring ETH and ERC20s are so common that Optimism has provided a “Standard Bridge” for us to do that, which are powered by the messengers we had talked about in the previous slides. The main points to know about the Standard Bridge are…

- There are 2 main contracts: **L1StandardBridge.sol** and **L2StandardBridge.sol**
    - Find a detailed walkthrough of these contracts here: https://ethereum.org/en/developers/tutorials/optimism-std-bridge-annotated-code/
- You **CANNOT bridge every ERC20**, but must make a PR to add it to Optimism’s token list
    - Find the token list here: https://github.com/ethereum-optimism/optimism-tutorial/tree/main/standard-bridge-standard-token
- **Deposit** means moving ETH and tokens **from L1 to L2**
- **Withdraw** means moving ETH and tokens **from L2 to L1**

source: https://ethereum.org/en/developers/tutorials/optimism-std-bridge-annotated-code/

Section 4

BUIDL BUIDL BUIDL

Not a crash course, but maybe a crash and burn

What are we building?

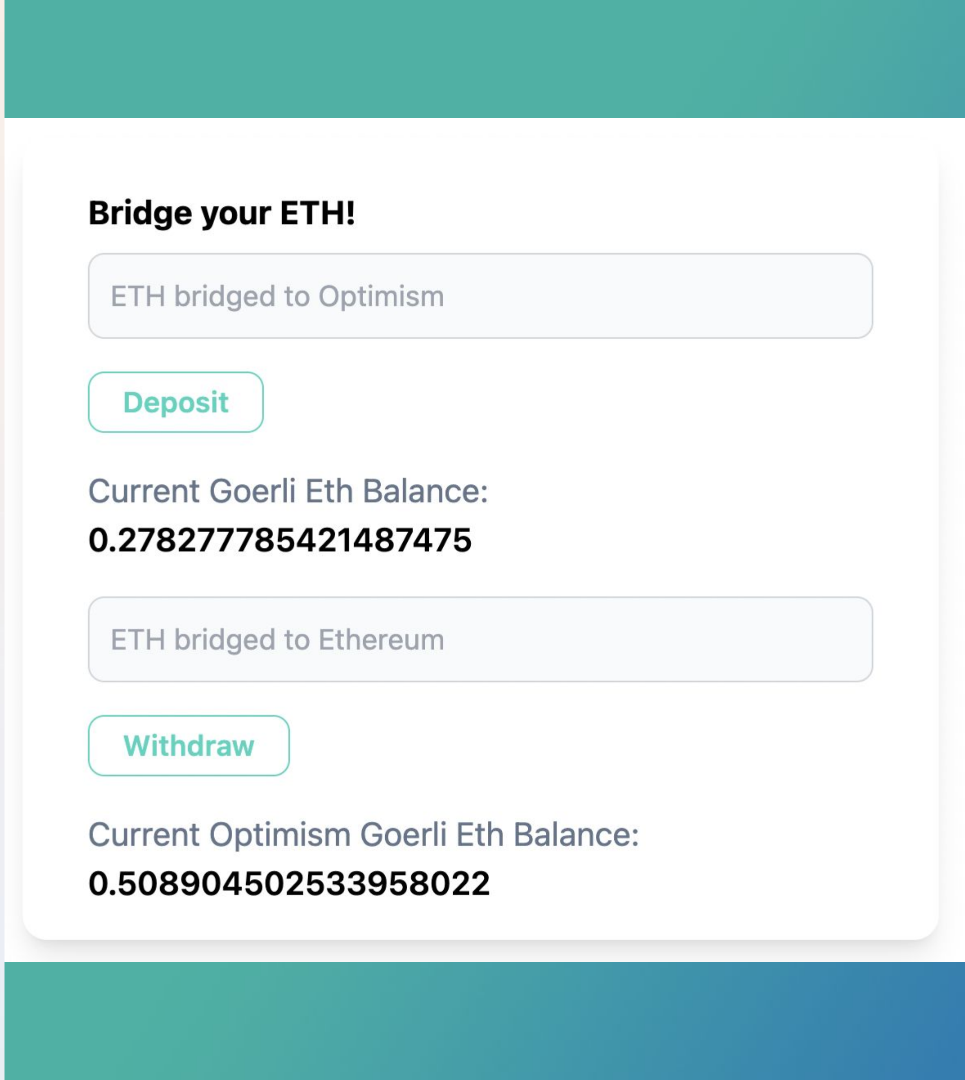
A NFT Marketplace on Optimism

UX Problem:

- Users have to go to a separate website to bridge their ETH

Solution:

- Bridget the bridge widget!
- Embed L2 bridging directly in your dapp



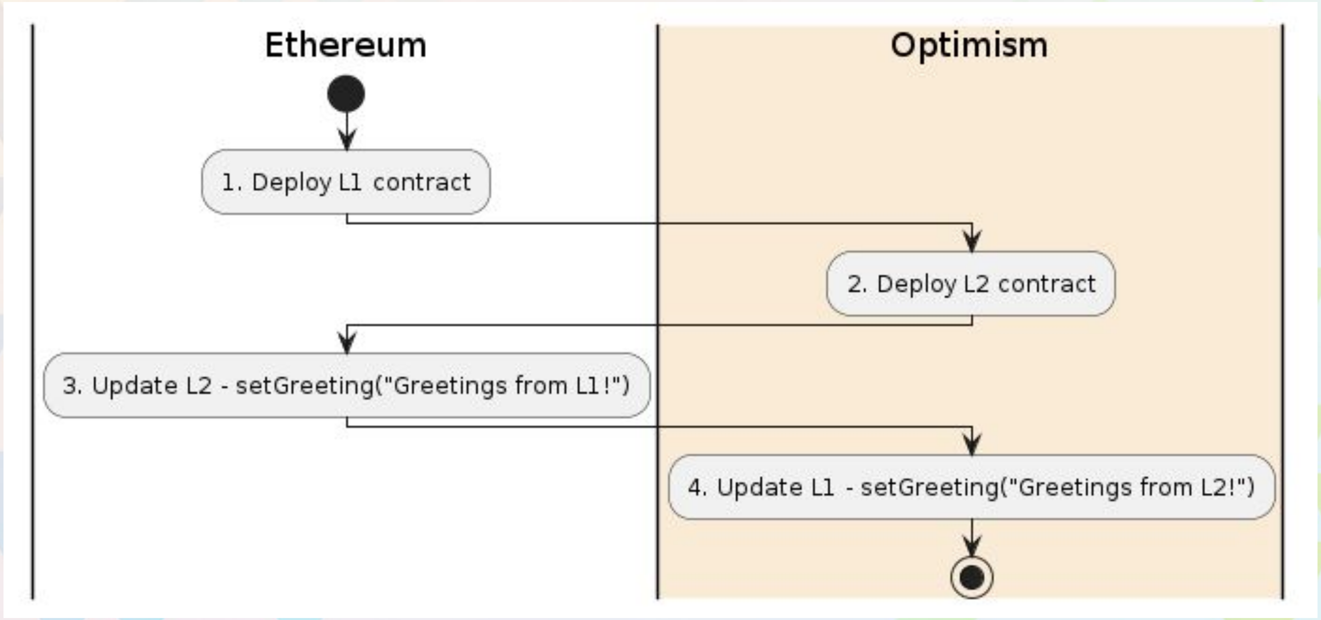
Bridge your ETH!

Current Goerli Eth Balance:
0.27827785421487475

Current Optimism Goerli Eth Balance:
0.508904502533958022

What you'll need

- Truffle
    - Optimism Bridge Box - example code for bridging in Optimism


Ethereum
Optimism
1. Deploy L1 contract
2. Deploy L2 contract
3. Update L2 - setGreeting("Greetings from L1!")
4. Update L1 - setGreeting("Greetings from L2!")

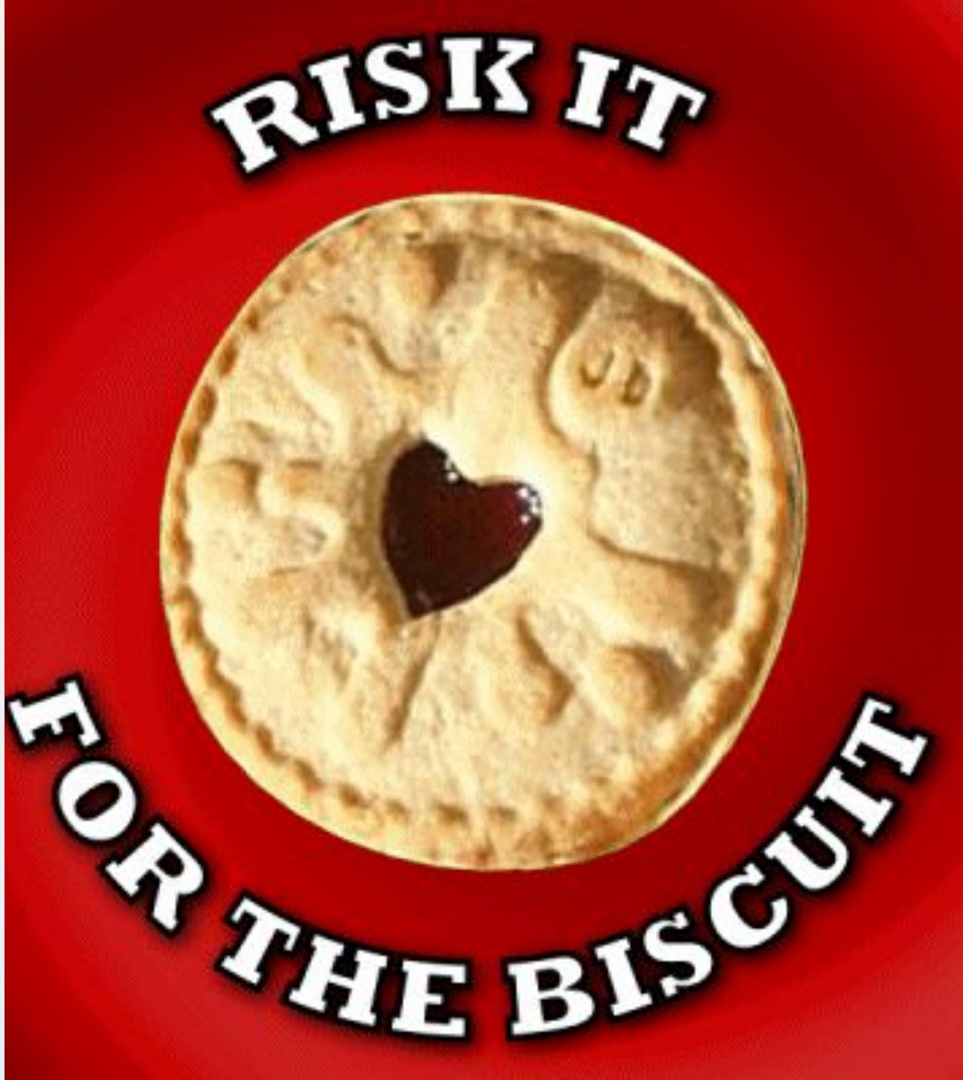
What you’ll need

- **Infura**
  - Connect to the Goerli and Optimism Goerli testnets
  - IPFS project for uploading NFT metadata
- **MetaMask**
  - Deploy your contracts and interact with your dapp
- **Goerli ETH, Optimism Goerli ETH, and Goerli DAI**

Are we doing this live?

Hehehehehe YES.

Fingers crossed nothing goes wrong 😅



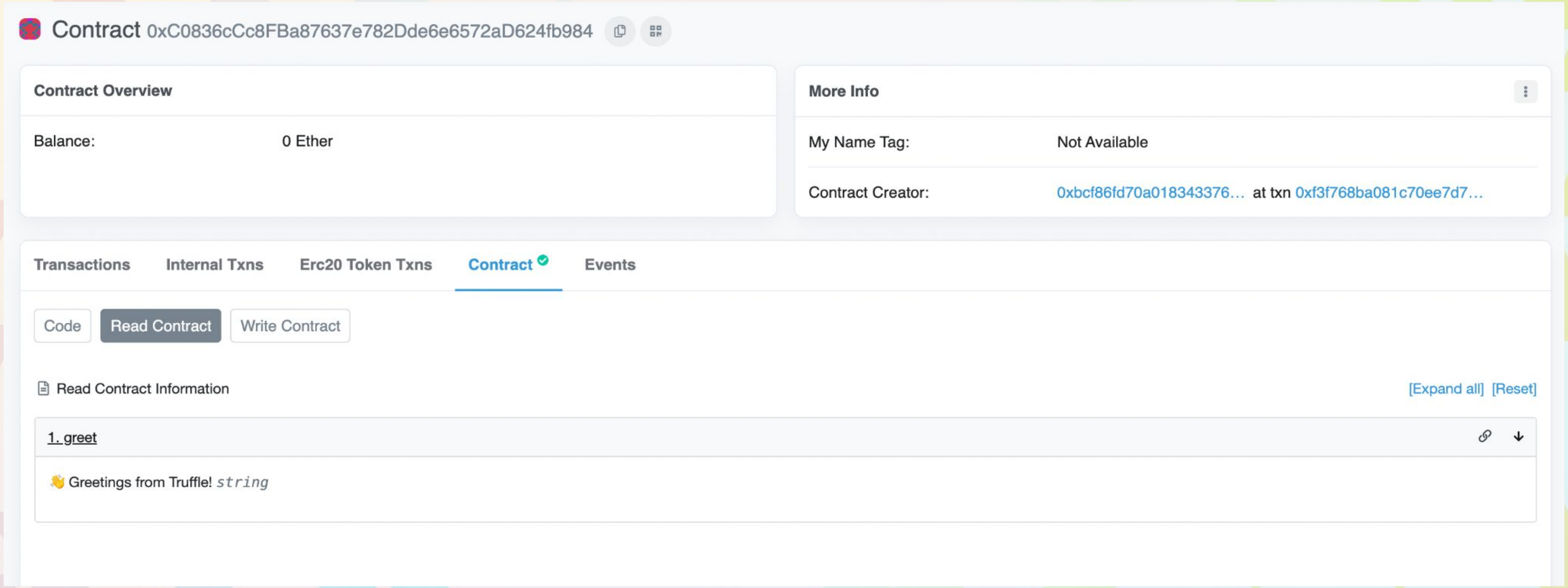
Please enjoy this low quality res pic instead

Are we do

Hehehehehe Y

Fingers cros
wrong 😅

L1 setGreeting on L2



Contract 0xC0836cCc8FBa87637e782Dde6e6572aD624fb984

Contract Overview

Balance:	0 Ether

More Info

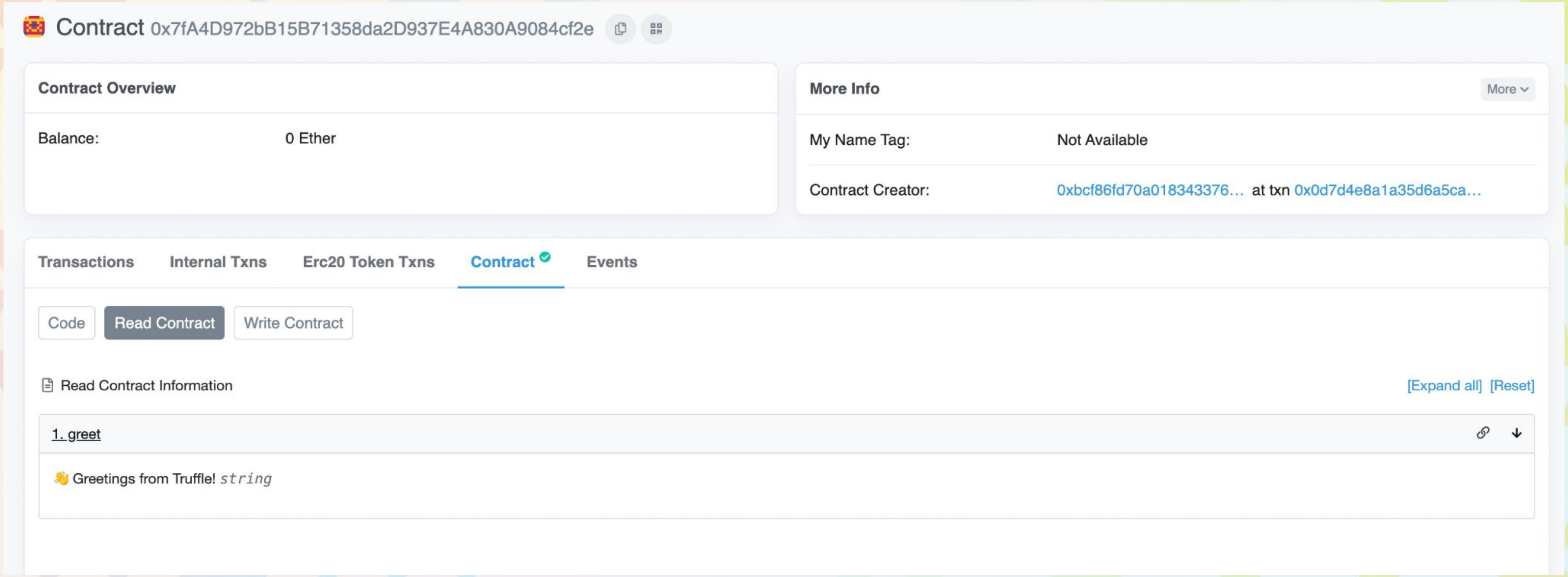
My Name Tag:	Not Available
Contract Creator:	0xbcf86fd70a018343376… at txn 0xf3f768ba081c70ee7d7…

Read Contract Information

1. greet

👋 Greetings from Truffle! *string*

L2 setGreeting on L1



Contract 0x7fA4D972bB15B71358da2D937E4A830A9084cf2e

Contract Overview

Balance:	0 Ether

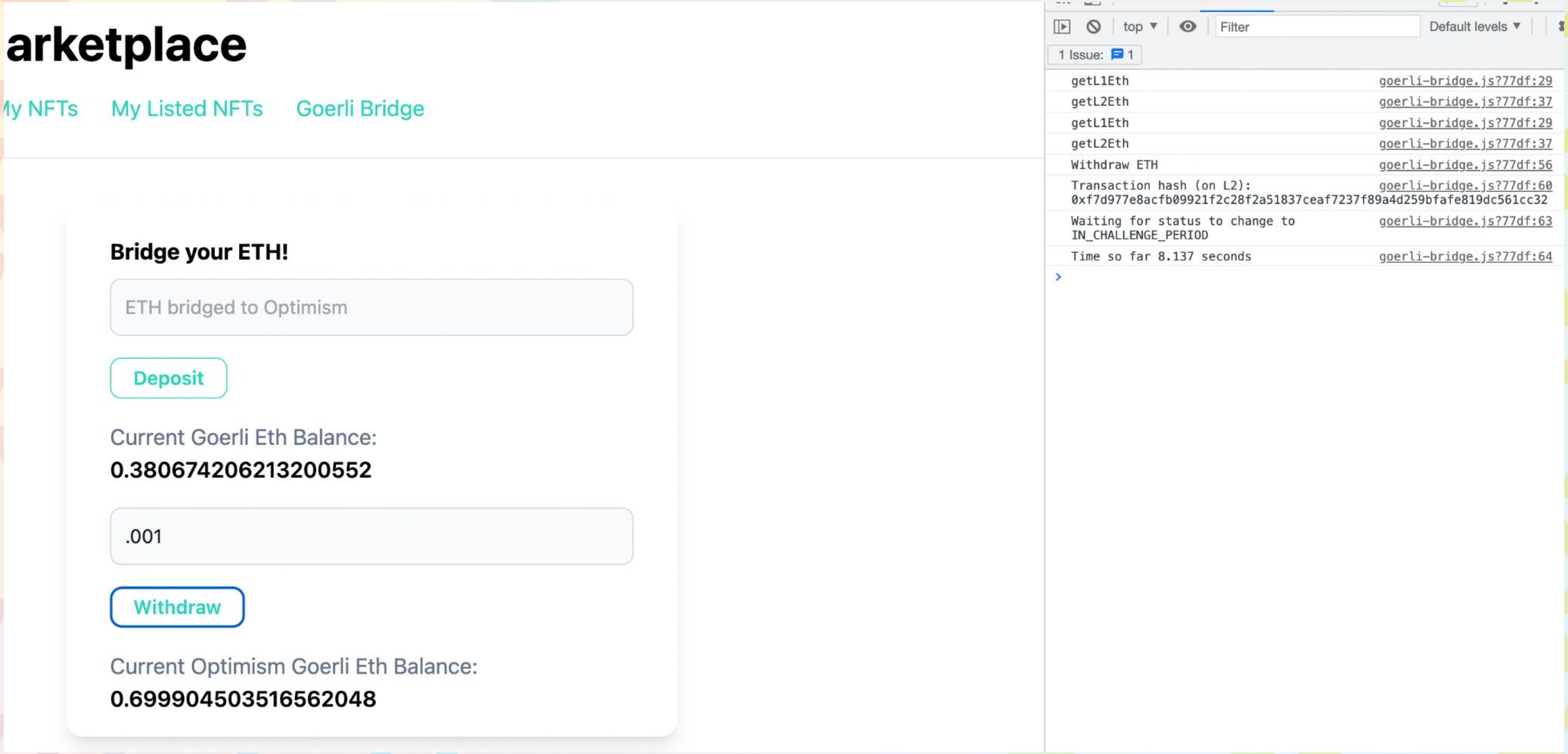
More Info

My Name Tag:	Not Available
Contract Creator:	0xbcf86fd70a018343376... at txn 0x0d7d4e8a1a35d6a5ca...

Read Contract Information

1. greet

👋 Greetings from Truffle! *string*



Bridge your ETH!

ETH bridged to Optimism

Deposit

Current Goerli Eth Balance:
0.380674206213200552

.001

Withdraw

Current Optimism Goerli Eth Balance:
0.699904503516562048

```
getL1Eth                                                     goerli-bridge.js?77df:29
getL2Eth                                                     goerli-bridge.js?77df:37
getL1Eth                                                     goerli-bridge.js?77df:29
getL2Eth                                                     goerli-bridge.js?77df:37
Withdraw ETH                                                 goerli-bridge.js?77df:56
Transaction hash (on L2):                                    goerli-bridge.js?77df:60
0xf7d977e8acfb09921f2c28f2a51837ceaf7237f89a4d259bfafe819dc561cc32
Waiting for status to change to                              goerli-bridge.js?77df:63
IN_CHALLENGE_PERIOD
Time so far 8.137 seconds                                    goerli-bridge.js?77df:64
>
```

Section 5

What's next?

A crash course

Section 5

What's next?

JK not a crash course

Potential Extensions

- Build a chat messenger using the Greeter contracts
  - In this case, you'll need Truffle to deploy the contracts to be utilized in the client (starting with the Optimism Bridge Box)
- Make Bridget compatible with other networks and ERC20s
- Tinker around with our other L2 boxes: Arbitrum and StarkNet (coming soon)

Truffle's plans for multi-chain

Multi-chain dapps are hard - complex deployment scripts and testing. What is Truffle doing to solve this problem?

- Declarative deployments
  - Declare your end state and Truffle shall grant your wish
  - Environment configs - test, dev, prod environments
- Ganache plugins
  - "L2 flavored" versions of Ganache - imagine forking Optimism to interact with its mainnet deployed contracts rather than just on Goerli
- Education
  - Nov 3rd: Web3 Unleashed livestream with Annie from Optimism to dive deeper into rollups, bridges, and Optimism Bedrock in the words of an ***actual expert***

Thank you!

Emily Lin

Developer Evangelist, Truffle@ConsenSys

https://beacons.ai/_emjlin

@_emjlin