National College of Ireland

Configuration Manual

MSc Research Project
Cloud Computing

Joan Carlo Lopez Marin
Student ID: 20232535

School of Computing
National College of Ireland

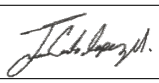
Supervisor: Shivani Jaswal

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Joan Carlo Lopez Marin
Student ID:	20232535
Programme:	Cloud Computing
Year:	2021
Module:	MSc Research Project
Supervisor:	Shivani Jaswal
Submission Due Date:	15/08/2022
Project Title:	Configuration Manual
Word Count:	2335
Page Count:	38

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	14th August 2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☑
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☑

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Joan Carlo Lopez Marin
20232535

1 Preparing Environment

1.1 Wallet

To interact with most of the blockchain we need tokens, this tokens are saved in an account, and one tool used to managage that accounts are the wallets, in this case the wallet selected is Metamask as is the mos popular and easy to use wallet by far. We have multiple options to install the wallet, in this case we will be installing the Firefox extension.

1. https://metamask.io/download/
2. click on "install Metamask for firefox"
3. you will be redirected to the mozilla ADD-ONS store, and here we have to click on "Add to Firefox 1".
4. After the plugin is added to the browser, a fox icon will be displayed on the top right corner 2.
5. Click on the icon and start the wallet configuration to create a new wallet account. It will be very intuitive .
6. During the configuration you will be configuring a password, remember this password because you need it to interact with blockchain.

1.2 Connect to test net

After the configuration of our wallet the next steps is connect it to a some test net bacause is dosent requiere real money to get tokens. The test net selected to this research is KOVAN (https://kovan.etherscan.io/).

1. Click on the fox icon 2
2. click on the top button "Ethereum Mainnet" 3
3. Click on "Show/hide" test networks 3 and turn on the feature.
4. In the networks displayed select KOVAN
5. The main view of the wallet should be chainged the Eth to kovanEth

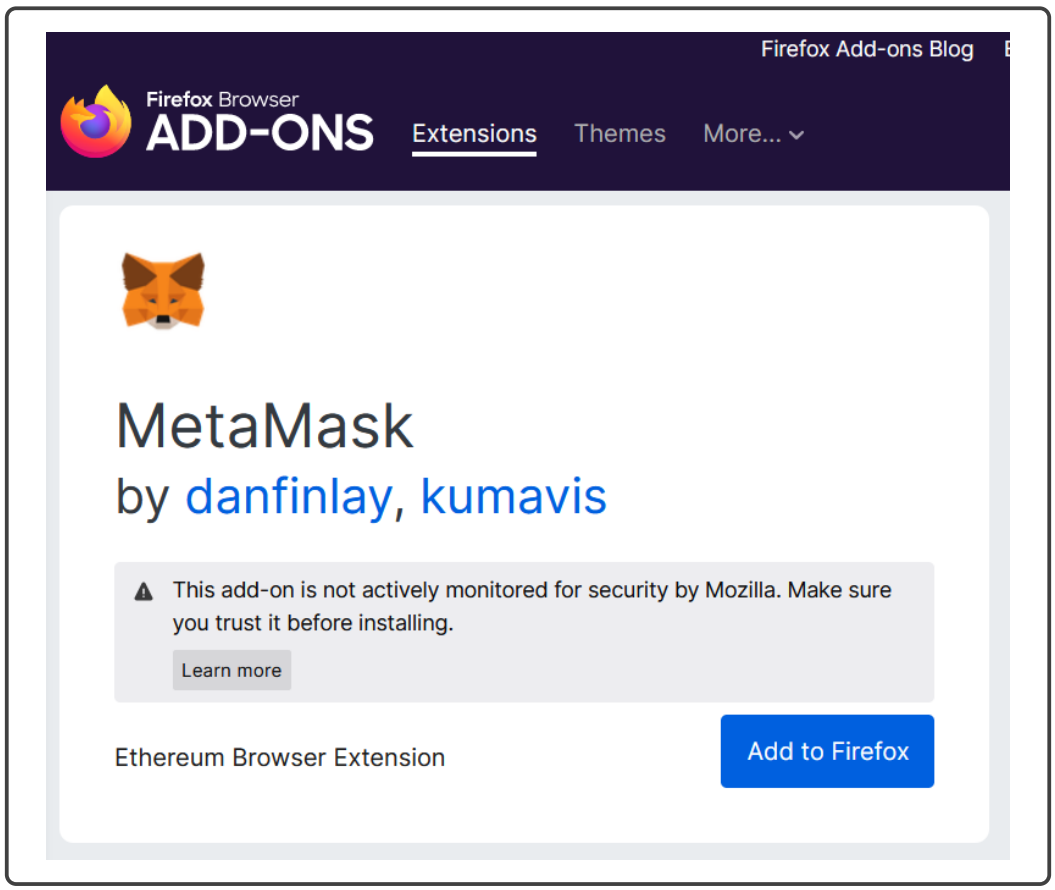


Figure 1: Add Metamask to firefox

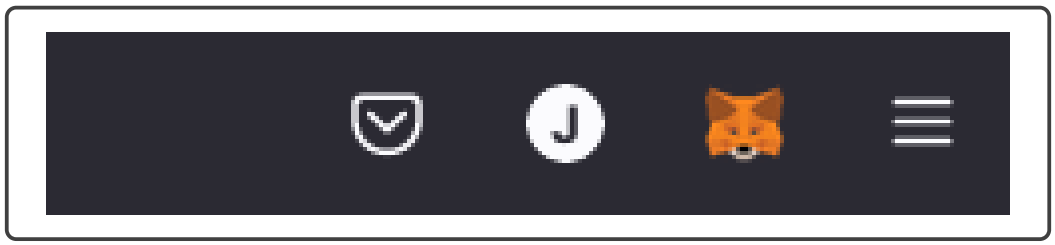


Figure 2: Fox icon in the browser

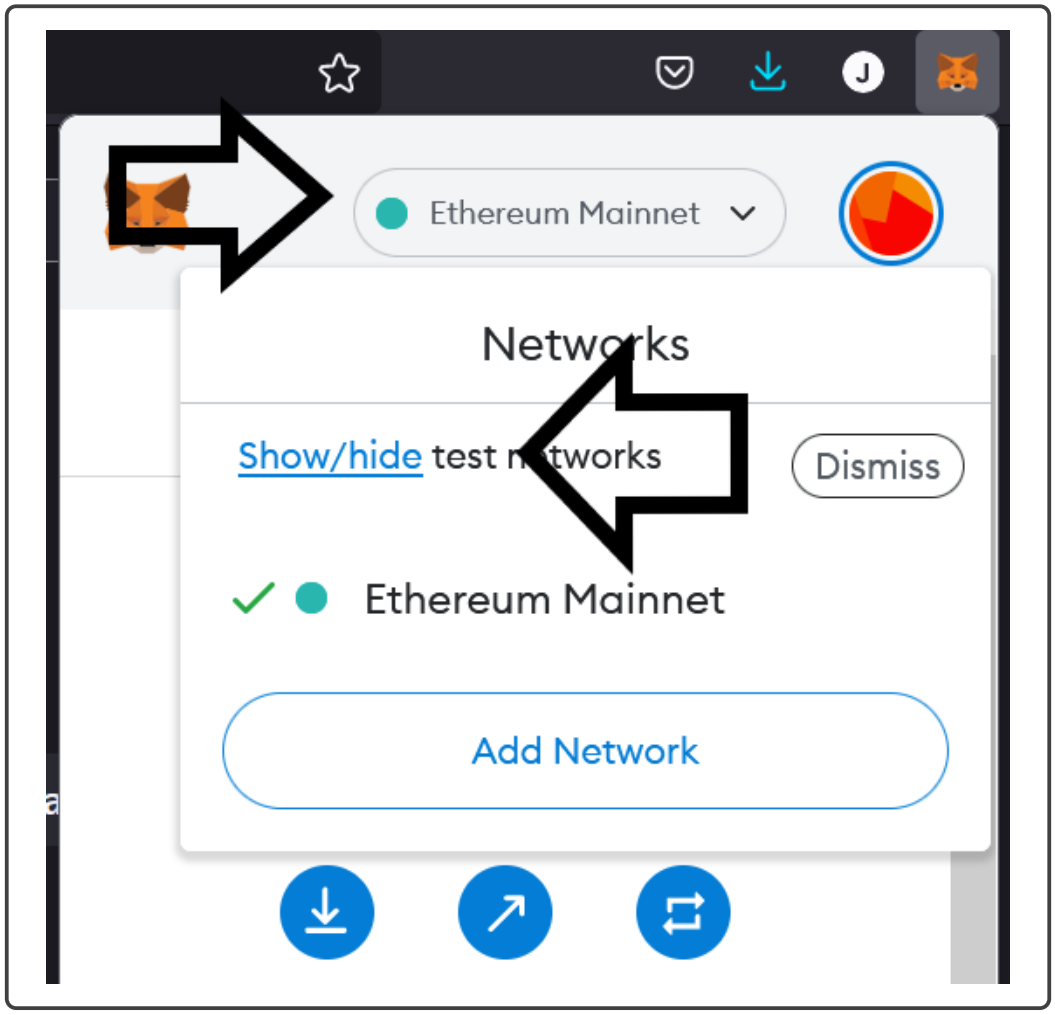


Figure 3: Display test net

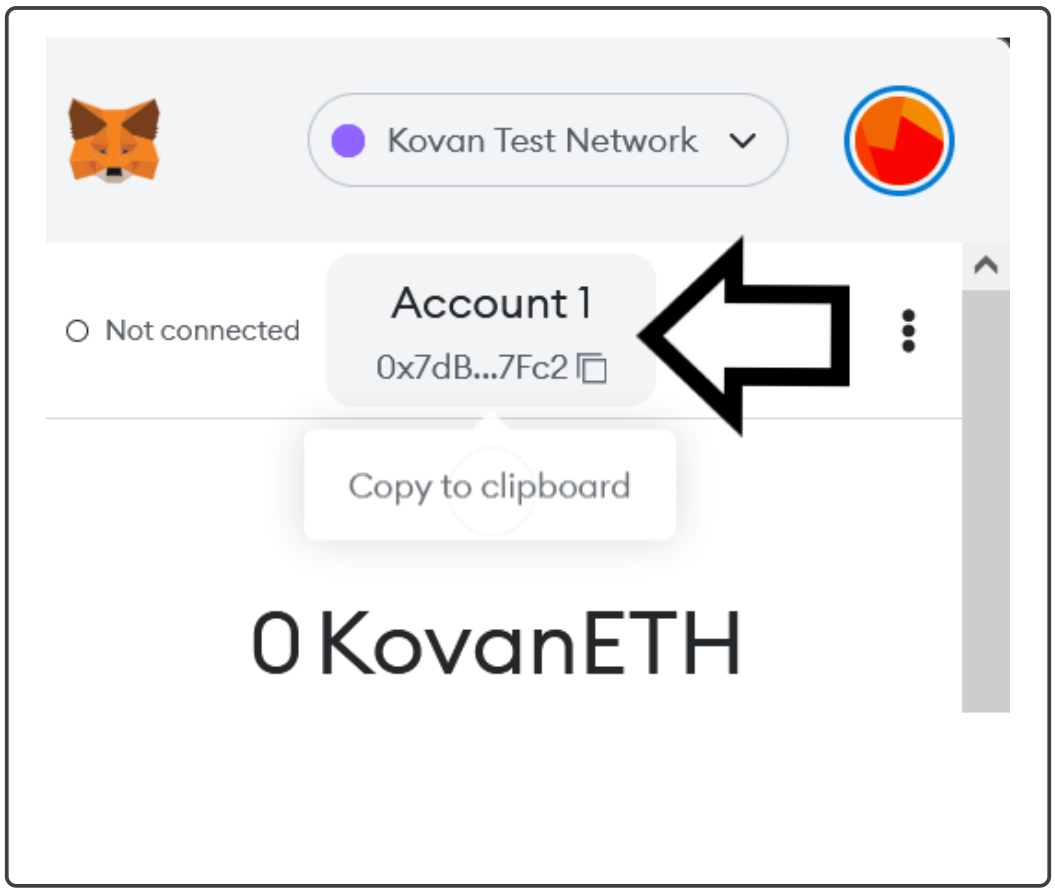


Figure 4: Wallet Address

1.3 Getting KovanEth

To interact with any ethereum network we need Ethers from that network, in the case of the tests nets there are some faucets where we can get the tokens for free. As well kovanETH we need the tokens needed to interact with the Oracles, in this case we will be using ChainLink, and the tokens are Links.

1. Identify and copy our wallet address 4
2. Go to https://faucets.chain.link/ and connect your wallet to get kovanETH and Link tokens. To get more kovanETH you can use the following faucets.
   (a) https://ethdrop.dev/
   (b) https://gitter.im/kovan-testnet/faucet
3. Follow the instructions.
4. After the request we have to wait a few minutes and the results will be kovanEthers and Links displayed in our wallet 5

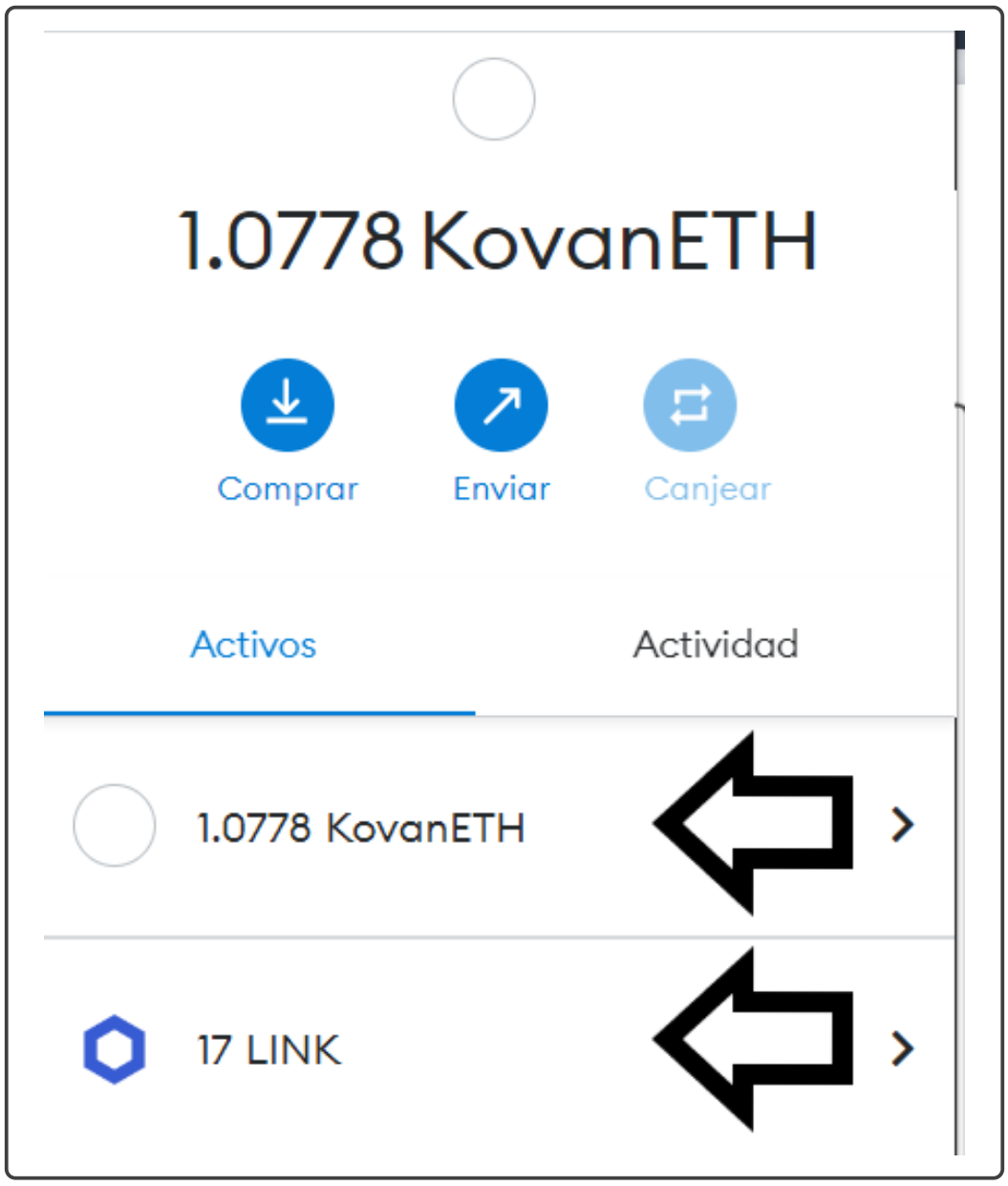


Figure 5: Wallet Tokens

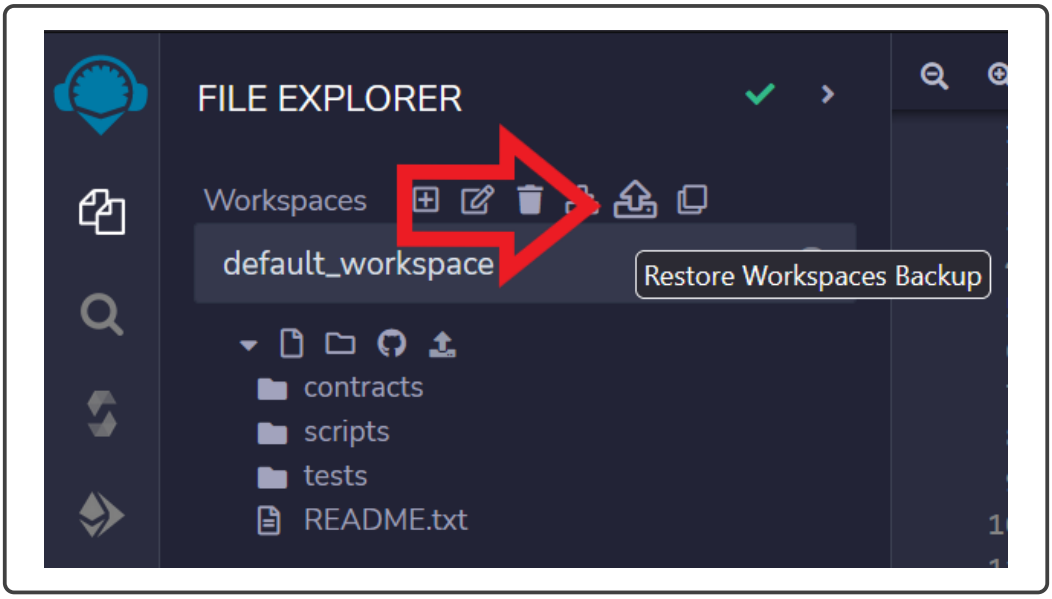


Figure 6: Restore Backups button

1.4 Code Editor

There are multiple ways to deploy smart contracts in Ethereum, the most popular code editors has available extensions but the most easy way is using Remix that is a code editor created to interact with blockchains.

Option 1: With this option all the patterns will be imported at the same time in the workspace.

1. Go to https://remix.ethereum.org/
2. in the File explorer click on "Restore Workspace Backup" 6
3. select the design_patterns_Dapps.zip file
4. Click on "import Design_patterns_dapps 7

Option 2: The second option is copy all the .sol files into the file explorer in remix. At the end of this section everything is ready to start running the code.

2 Running the patterns code

At this point all the technologies, tokens and process are ready to deploy the patterns in the blockchain.

2.1 Proxy Delegate Pattern [1]

To deploy the proxy patterns follow the instructions:

1. Verify that you are located in the correct workspace 8.
2. Go to the path contracts /Design_patterns_Dapps/proxy_pattern

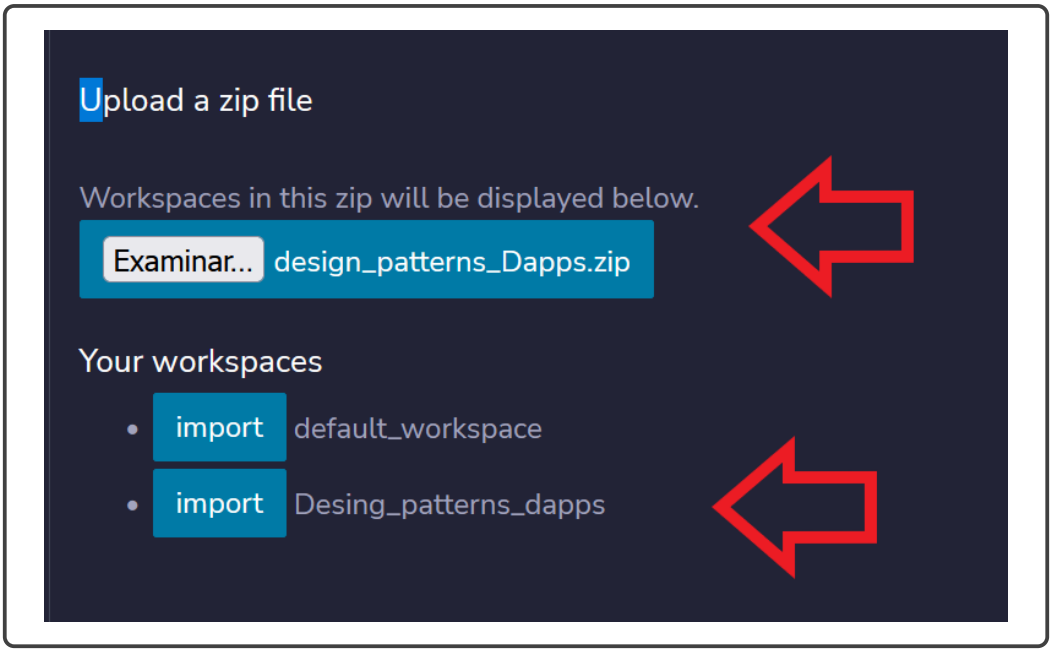


Figure 7: Import Design Patterns

3. Open the file proxy_contract.sol
4. In the left panel click on the solidity icon (compiler) 9
5. Verify that the version displayed in the compiler is the same version as the version displayed at the beginning of the file "pragma solidity 0.8.7".
6. Click on the "Compile proxy_contract.sol"
7. After the compilation is ready go to the "Deploy & run transactions " section in the left panel 10.
8. In the first option "Environment" select "Injected Provider - Metamask"
9. Connect metamask with remix, clicking on Next and Connect.
10. Then go to the contract section and select the contract Example
11. Click on "Deploy"
12. The gas cost will be displayed in the metamask view, click on confirm to deploy the contract in Kovan Test net11.
13. wait until the terminal return a green check 12
14. Then the contract deployed is displayed in the bottom of the left panel.
15. Copy the contract address deployed 13.
16. Then we have to select the next contract that is "Dispatcher"

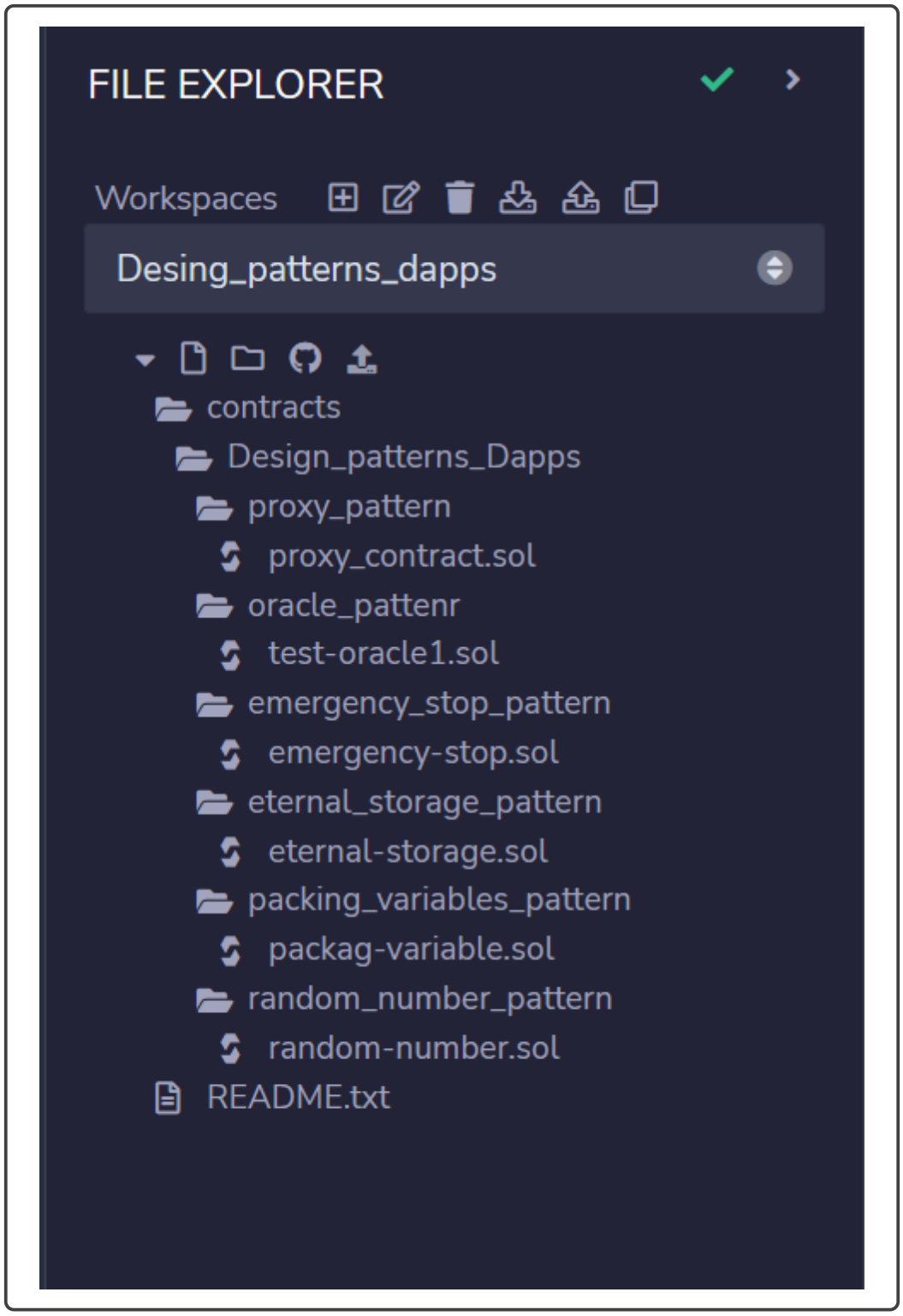


Figure 8: Patterns Folders and files

17. After select the Dispatcher an input will be displayed next to the deploy button, in that input paste the addres copyed from the other contract.

18. Click on deploy and confirm the transaction 14

19. Copy the address of the dispatcher after the deployment, in the contract select again the Example, And in the "At Address input paste the dispatcher address" and click on At Address15.

20. Now in the Deployed contracts section, there are 3 contracts, click on the last one.

21. Test the functionality: Write a number in the input next to the setUint and then click on the setUint Button and confirm the transaction16.

22. In the last step we stored a number, now we can get the number clicking in the button getUint.

23. If the number is not returned it could be because the gas limit is to low and the transaction is not performed. To increase the gas limit see the Appendix A.

24. At this point the Example contract is returning the number introduces multiplied for 2, lets modify the example contract to deploy a new one and test the proxy.

25. open the contract and modify the line 46 and change the number 2 to 4.

26. Compile the contract again, then change to the deploy view, select the contract in the contract section and deploy with the ORANGE button.

27. Now copy the new address of the example contract deployed and paste it in the dispatcher contract17.

28. Confirm the transaction.

29. We can still using the same instance that we deploy before, the 3rd contract deployed to test the updated function19.

2.2 Randomness Pattern[2]

To deploy the Randomness patterns follow the instructions:

1. Verify that you are located in the correct workspace 8.

2. Go to the path contracts /Design_patterns_Dapps/random_number_pattern

3. Open the file random-number.sol

4. In the left panel click on the solidity icon (compiler) 9

5. Verify that the version displayed in the compiler is the same version as the version displayed at the beginning of the file "pragma solidity 0.8.7".

6. Click on the "Compile random-number.sol"

Figure 9: Solidity Compiler

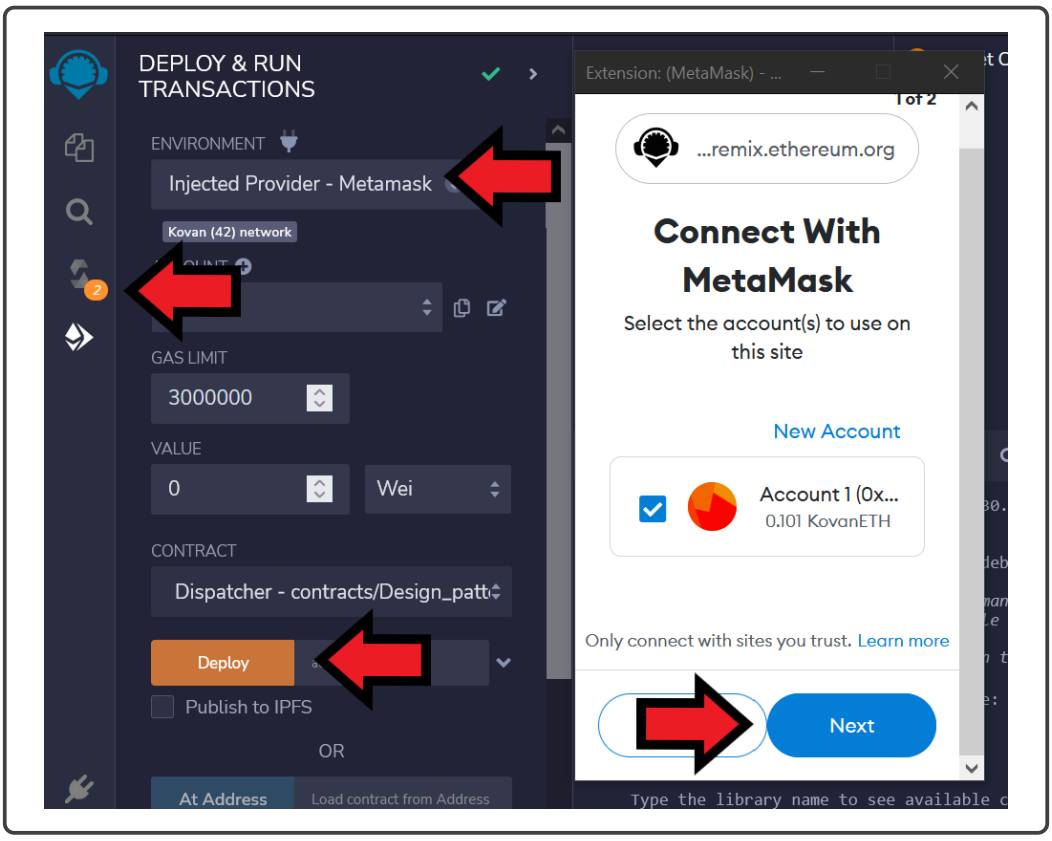


Figure 10: Deploy Contract

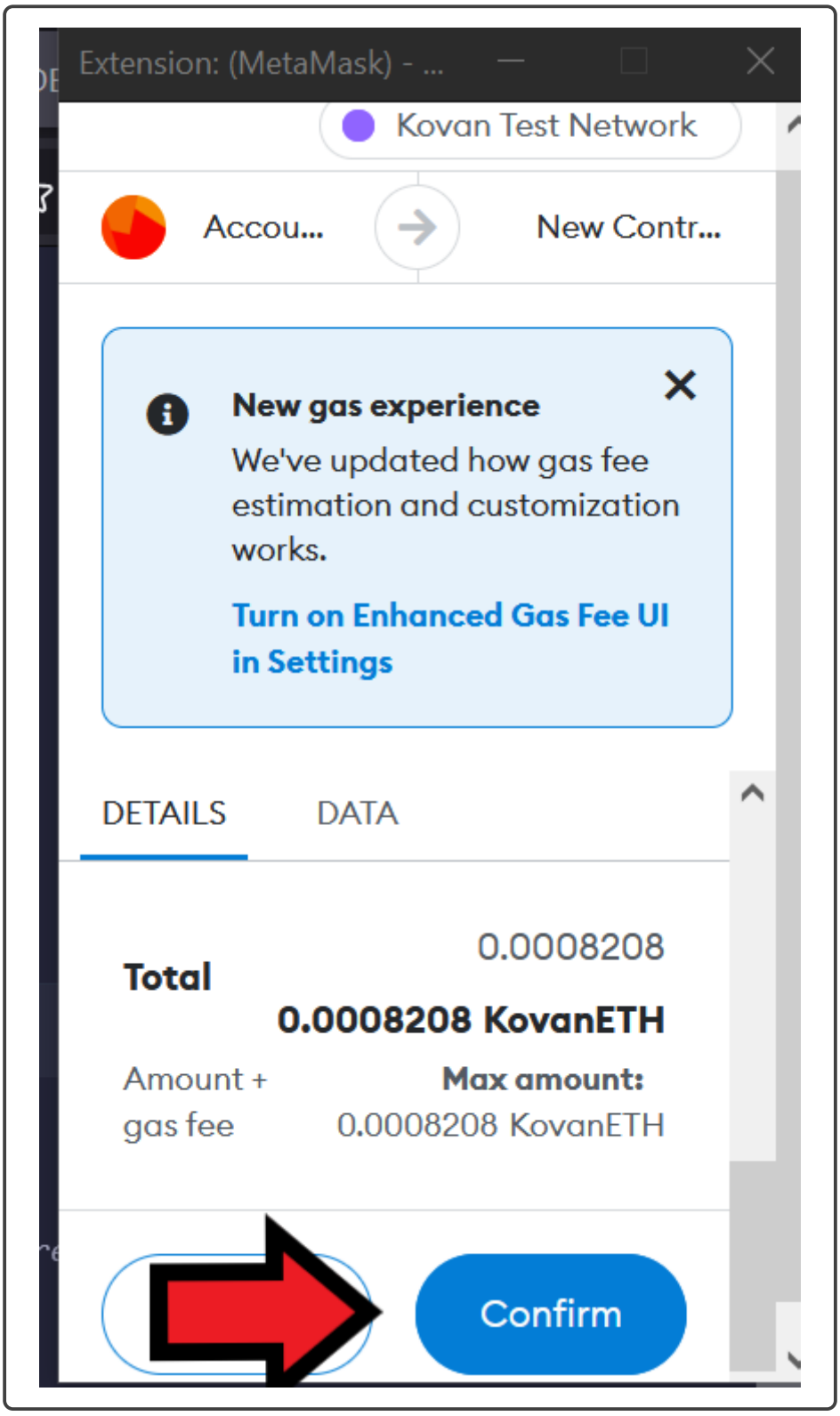


Figure 11: Confirm Deploy proxy - Example Contract

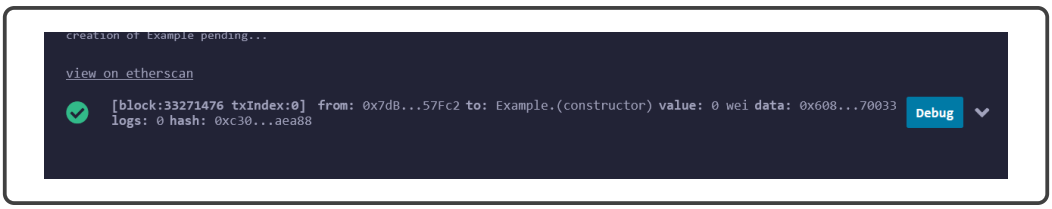


Figure 12: Green check confirm - Example Contract

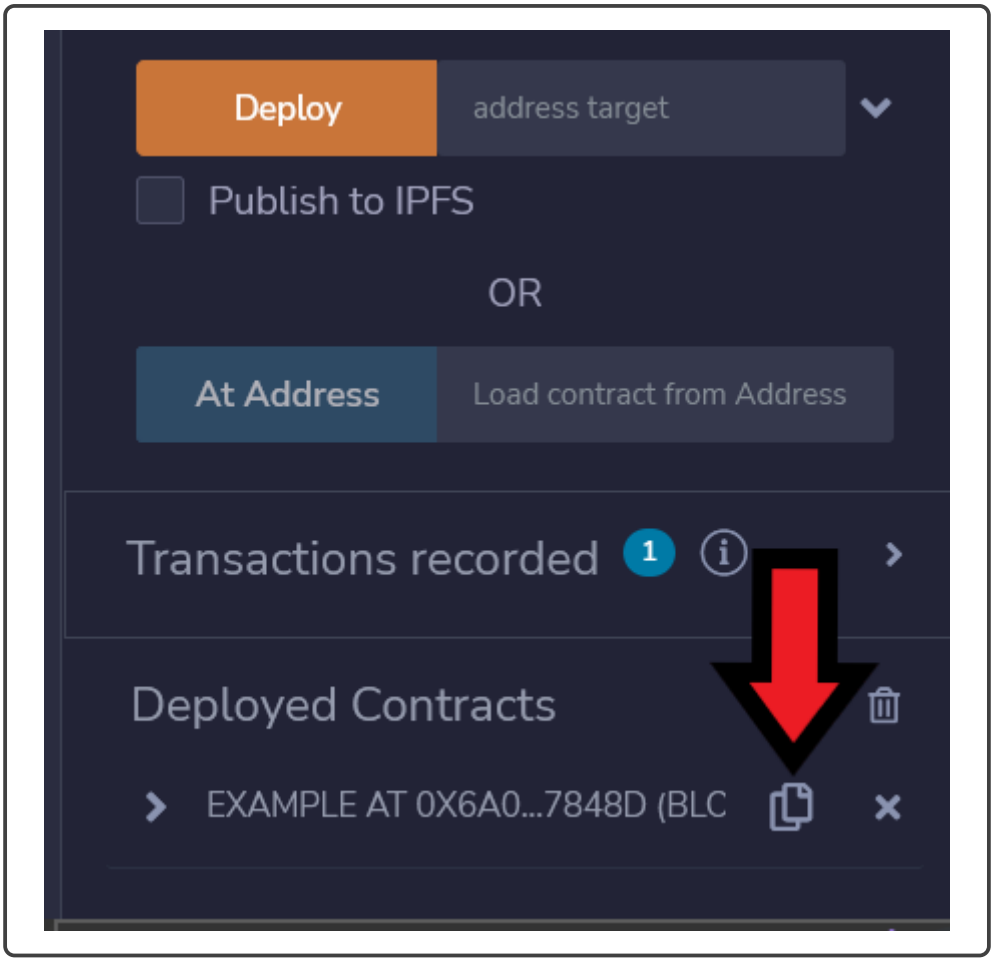


Figure 13: Copy Address Contract

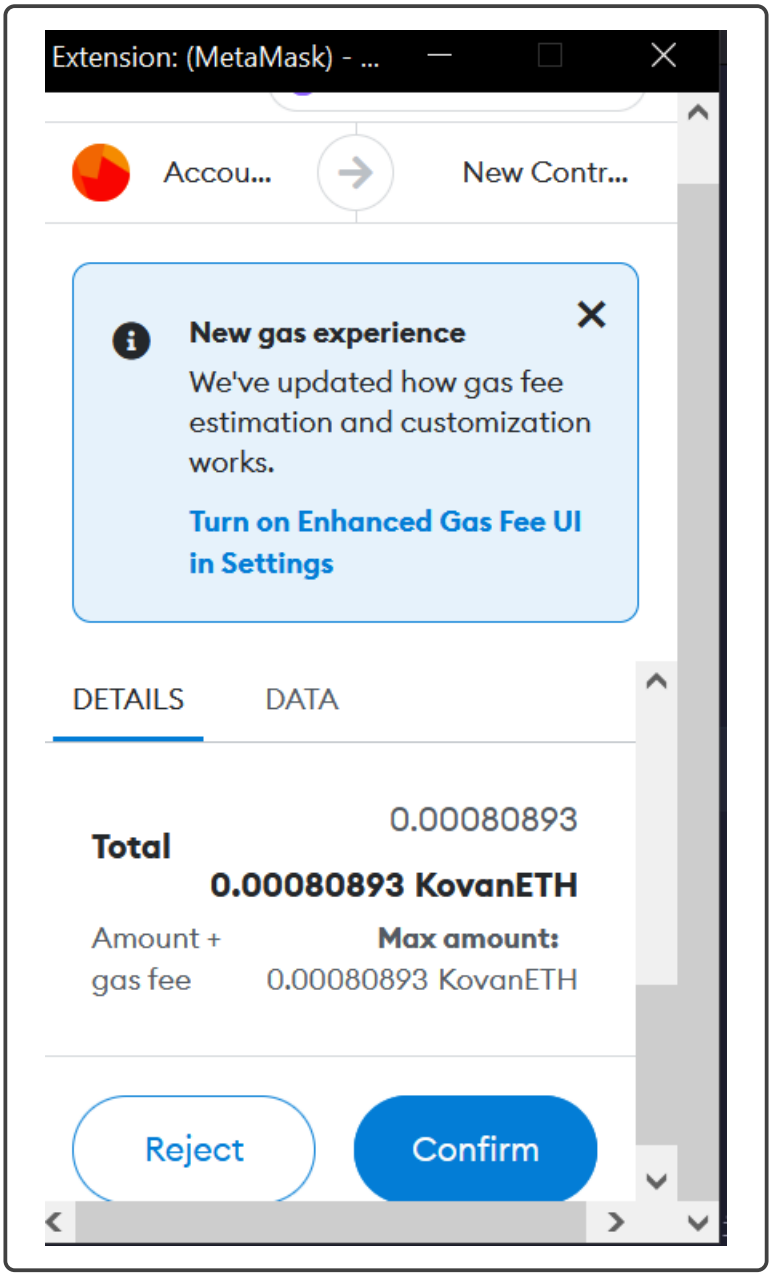


Figure 14: Dispatcher fees deploy

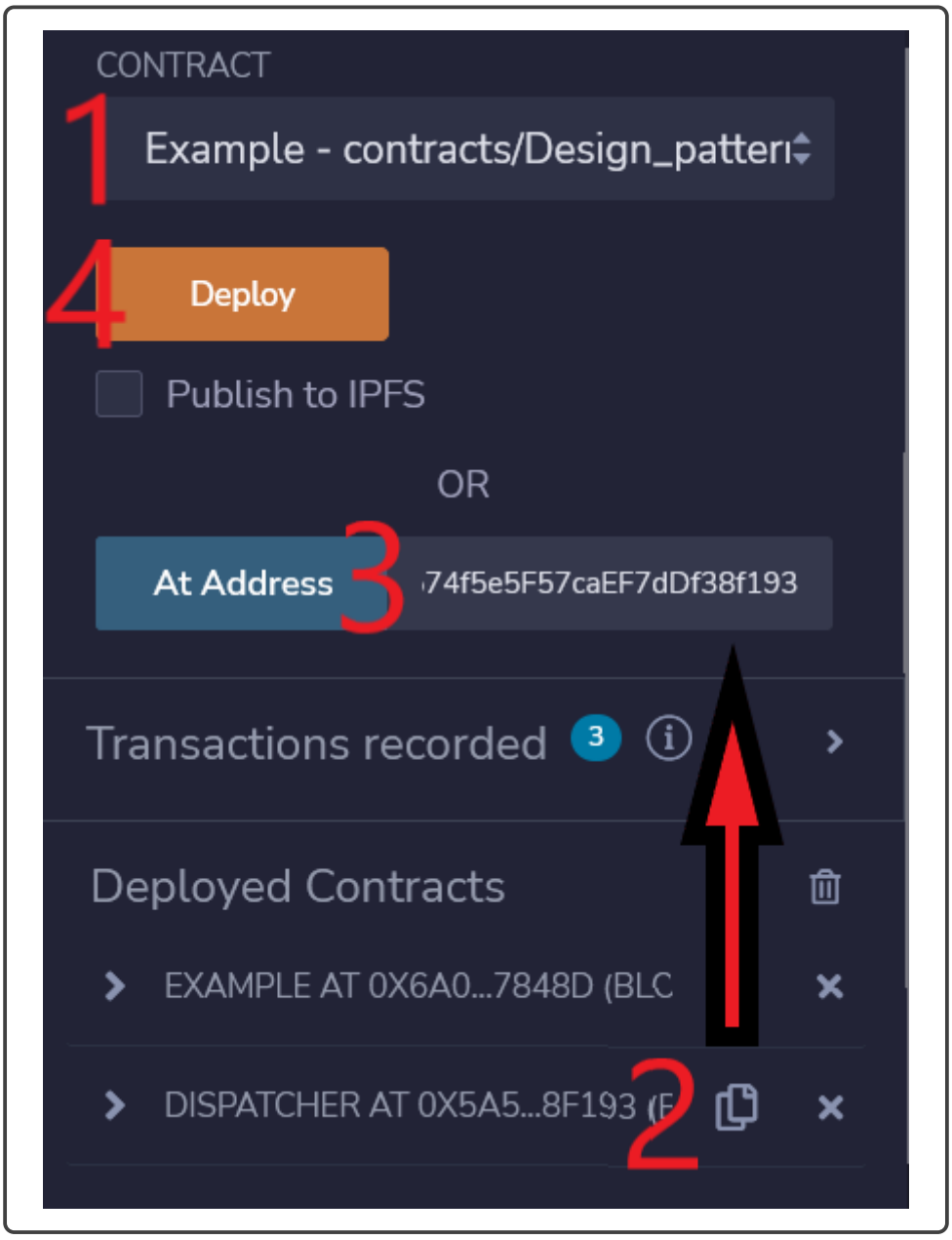


Figure 15: Deploy Example on the Dispatcher address

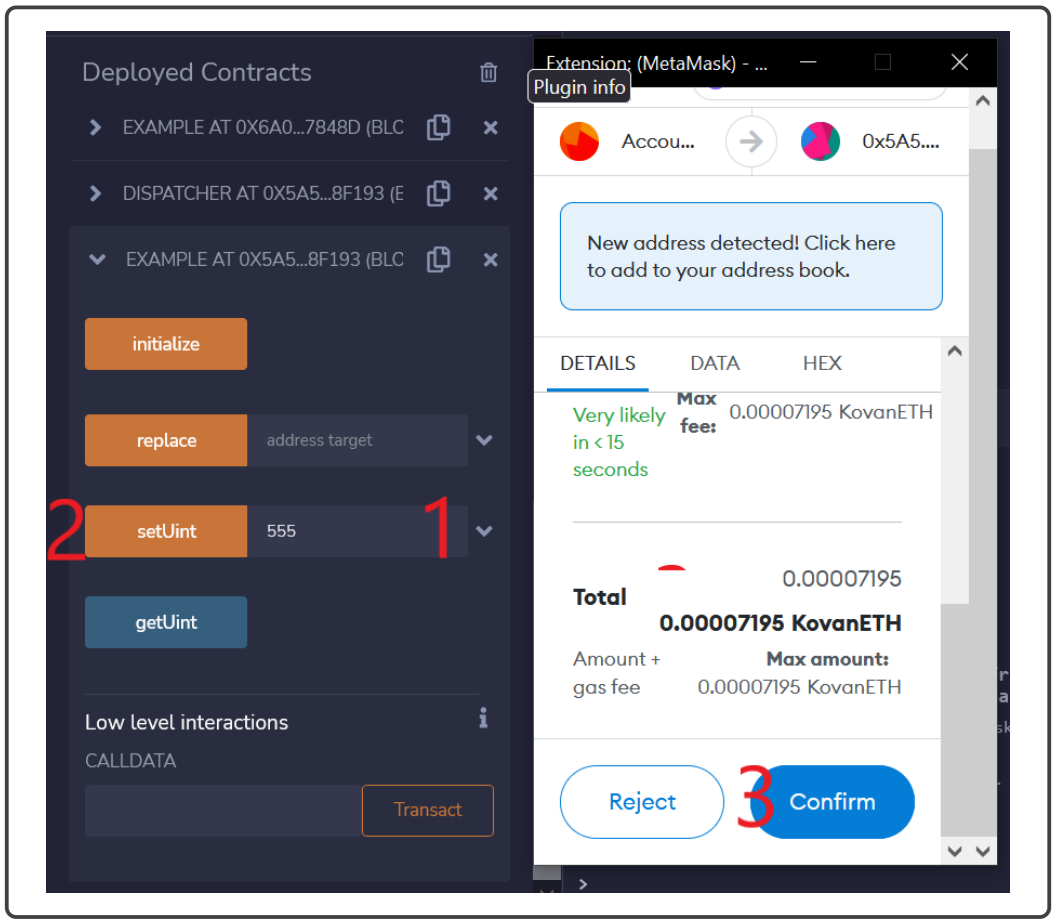


Figure 16: Test contract example function

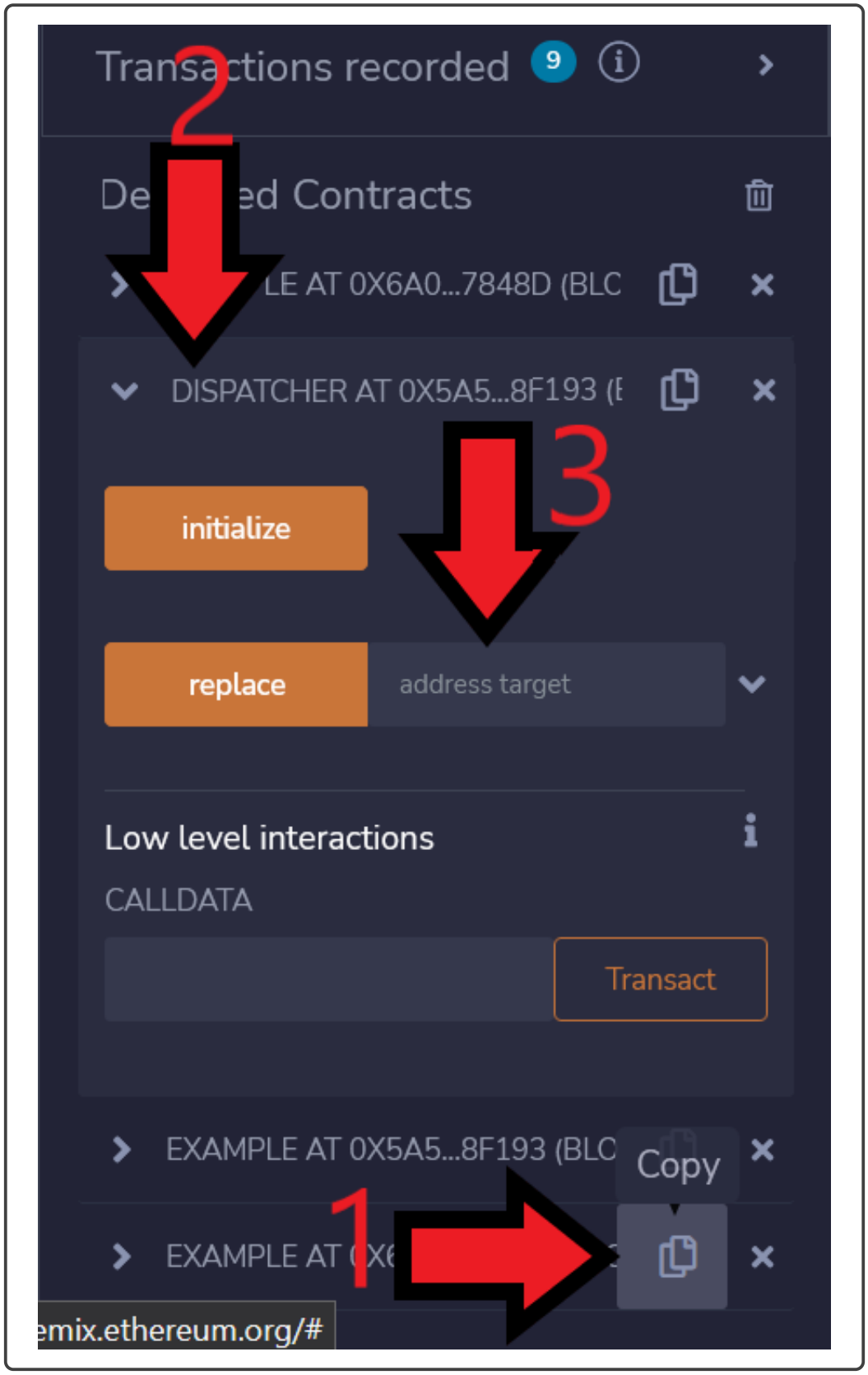


Figure 17: Replace the example address in the proxy contract

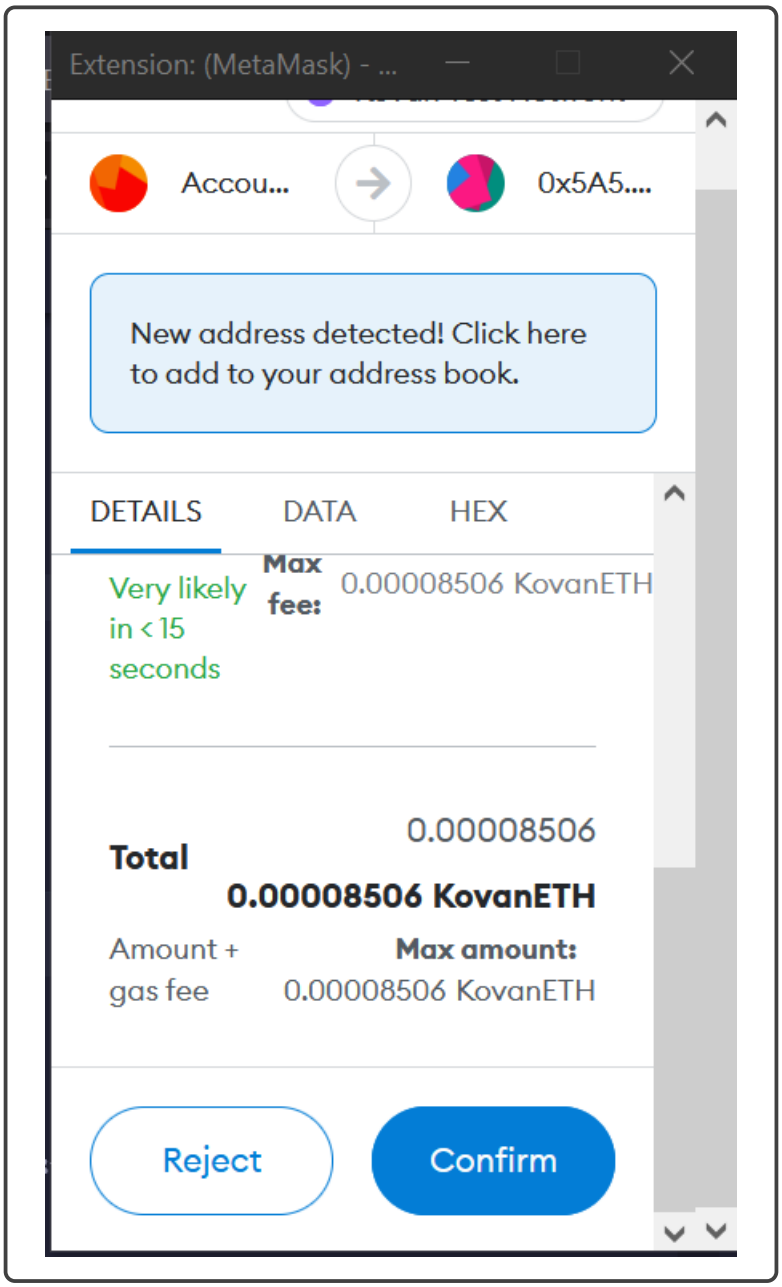


Figure 18: Replace address fee

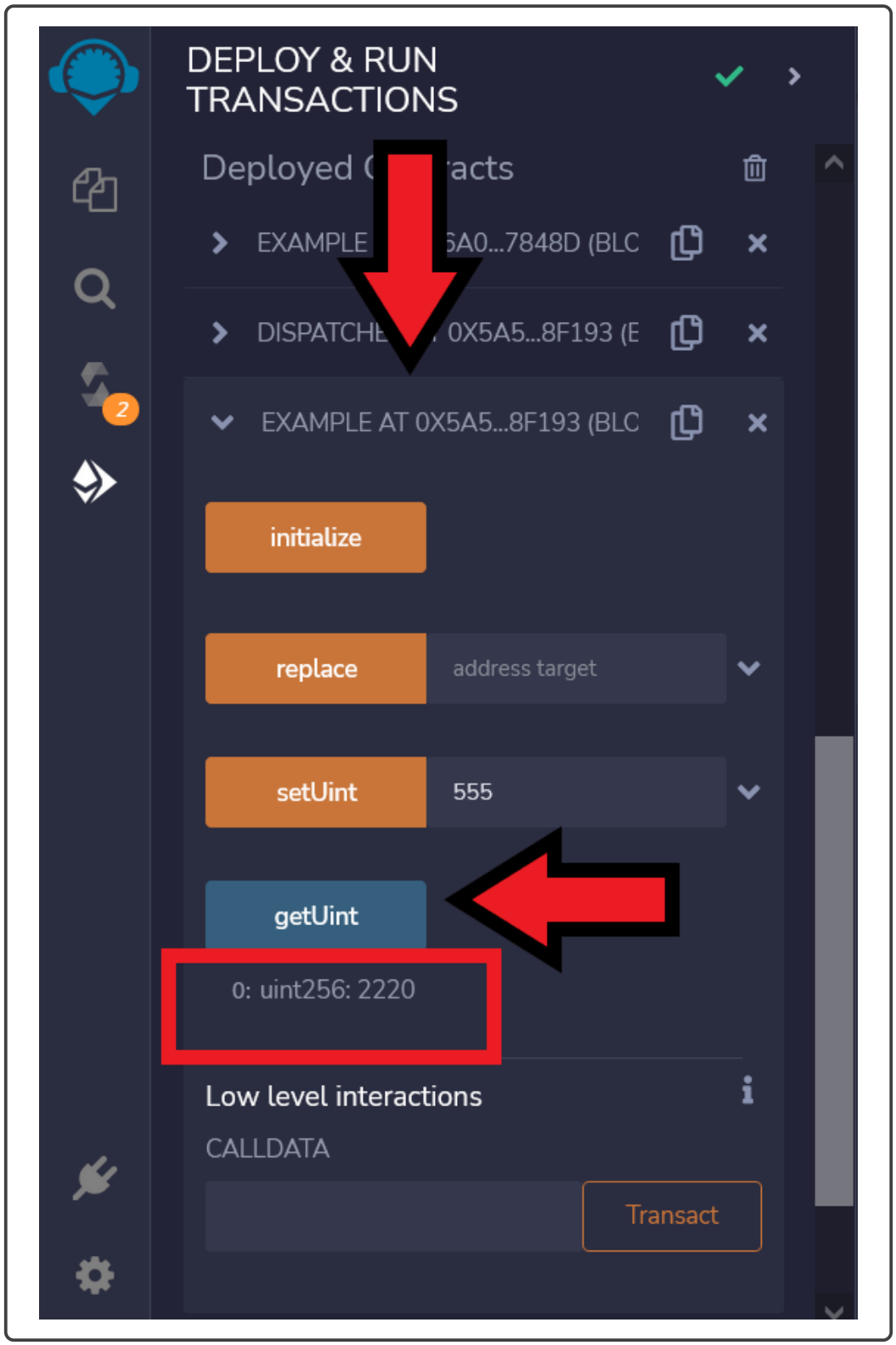


Figure 19: Updated contract address

7. After the compilation is ready go to the "Deploy & run transactions " section in the left panel 10.
8. In the first option "Environment" select "Injected Provider - Metamask"
9. Connect metamask with remix, clicking on Next and Connect.
10. Then go to the contract section and select the contract randomNumberTest
11. Click on "Deploy"
12. The gas cost will be displayed in the metamask view, click on confirm to deploy the contract in Kovan Test net20.
13. wait until the terminal return a green check 12
14. Then the contract deployed is displayed in the bottom of the left panel.
15. Click on the contract and there will be 3 options to generate the random number.
    (a) one seed
    (b) two seeds
    (c) tree seeds
16. the simple random number generator just needs one seed but to add more complexity to the random method it can be added multiple seeds.
17. The execution of this contract does not generate price because it does not modify the blockchain. Is created as provider contract only.

2.3 Oracle Pattern[2]

To deploy the oracle pattern follow the instructions:

1. Verify that you are located in the correct workspace 8.
2. Go to the path contracts /Design_patterns_Dapps/ oracle_pattern
3. Open the file test-oracle.sol
4. In the left panel click on the solidity icon (compiler) 9
5. Verify that the version displayed in the compiler is the same version as the version displayed at the beginning of the file "pragma solidity 0.8.7".
6. Click on the "Compile test-oracle"
7. After the compilation is ready go to the "Deploy & run transactions " section in the left panel 10.
8. In the first option "Environment" select "Injected Provider - Metamask"
9. Connect metamask with remix, clicking on Next and Connect.

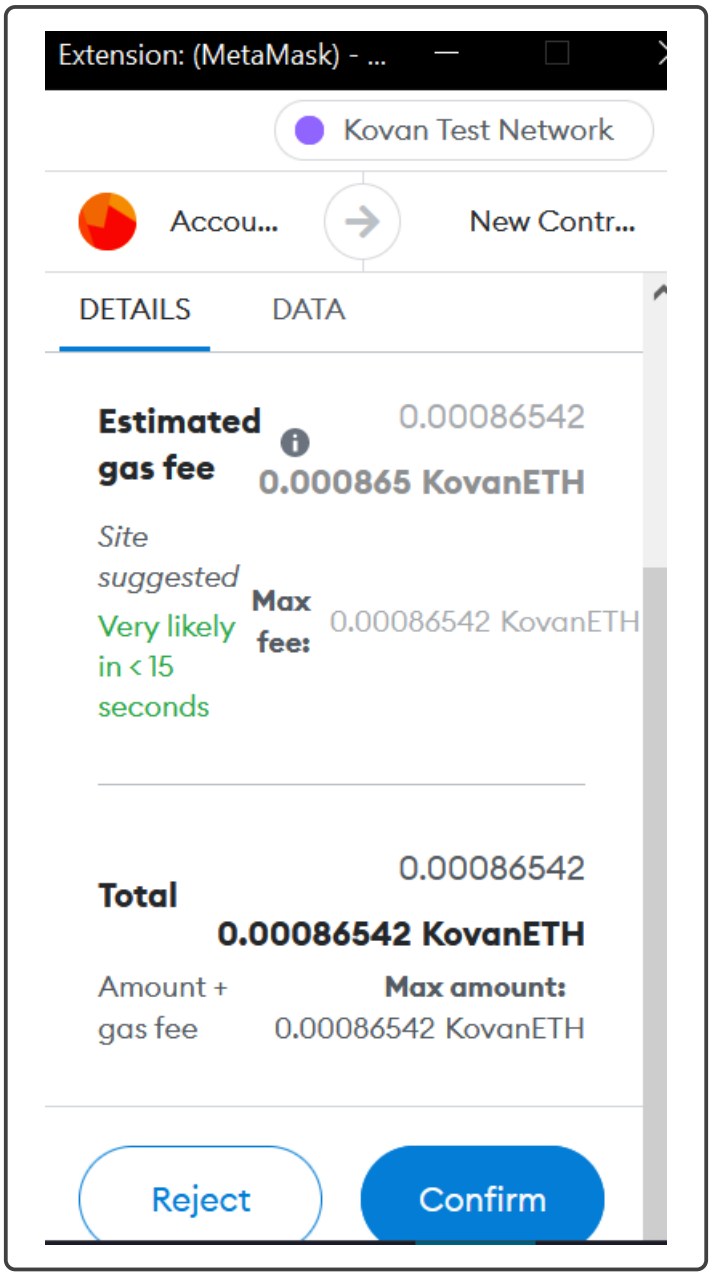


Figure 20: Random contract fee deploy

10. Then go to the contract section and select the contract FetchPriceFromCovalentAPI
11. Click on "Deploy"
12. The gas cost will be displayed in the metamask view, click on confirm to deploy the contract in Kovan Test net.
13. wait until the terminal return a green check 12
14. Then the contract deployed is displayed in the bottom of the left panel, Copy the contract address 21.
15. To use the oracle we need to send LINK tokens to the contract, because the oracle will be asking for that tokens to the contract.
16. Open Metamask and click on the Send button22
17. Paste the contract address in the input, and change the Asset to LINK and in the amount change to 3. **Note:** if the token is no displayed see the appendix B.
18. Confirm the transaction.
19. No go to the contract and click on "requestETHPrice", then click on price, this will be getting the price of an ETH outside of the blockchain 23

2.4 Emergency Stop Pattern[2]

To deploy the Emergency Stop pattern follow the instructions:

1. Verify that you are located in the correct workspace 8.
2. Go to the path contracts /Design_patterns_Dapps/emergency_stop_pattern
3. Open the file emergency-stop.sol
4. In the line 6 the variable address needs to be equals to your wallet address, because only that address could modify the state of the contract and save it.
5. In the left panel click on the solidity icon (compiler) 9
6. Verify that the version displayed in the compiler is the same version as the version displayed at the beginning of the file "pragma solidity 0.8.7".
7. Click on the "Compile Emergency_stop"
8. After the compilation is ready go to the "Deploy & run transactions" section in the left panel 10
9. In the first option "Environment" select "Injected Provider - Metamask"
10. Connect metamask with remix, clicking on Next and Connect.
11. Then go to the contract section and select the contract Emergency_stop
12. Click on "Deploy" 24

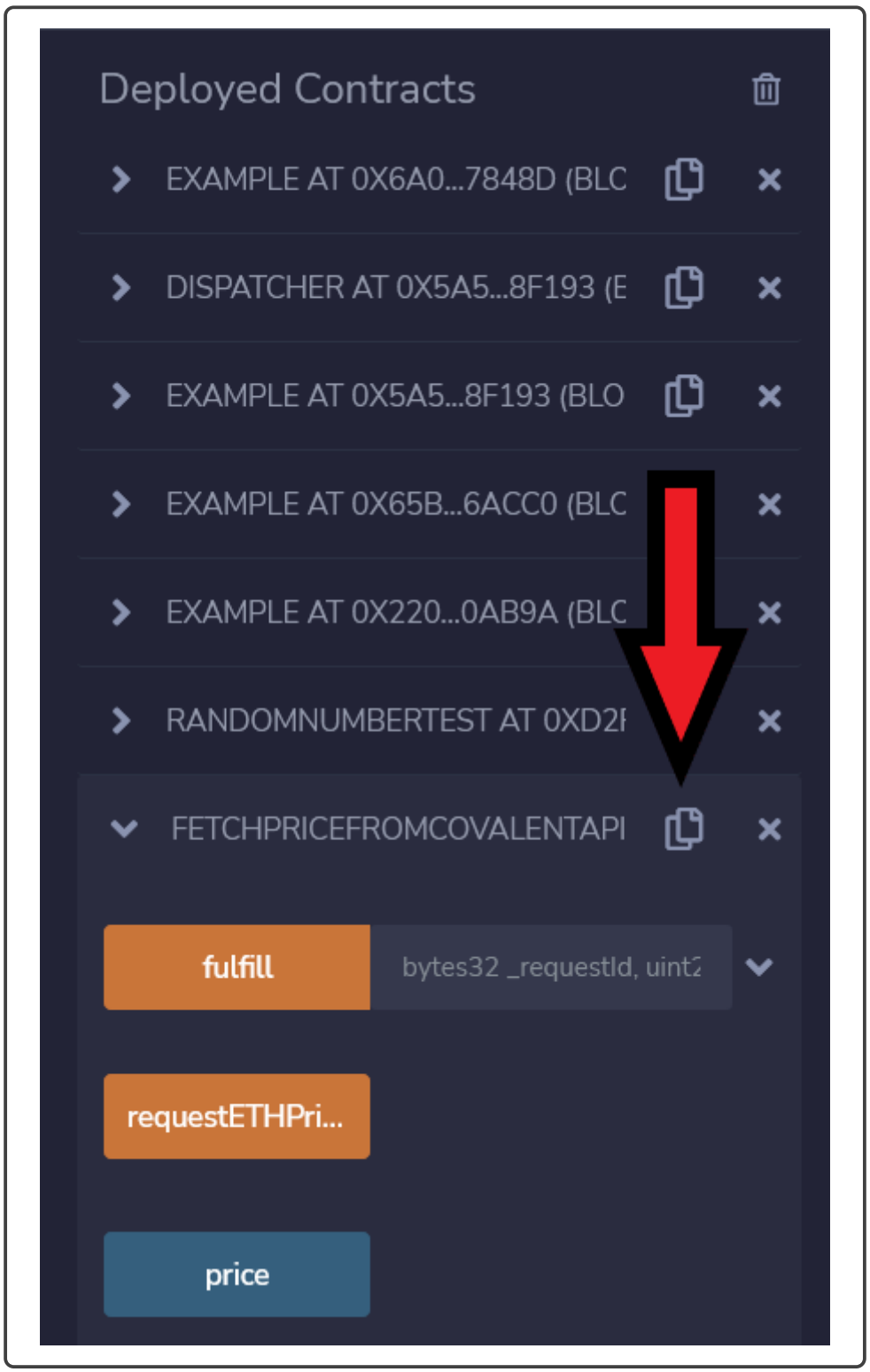


Figure 21: Copy oracle contract address

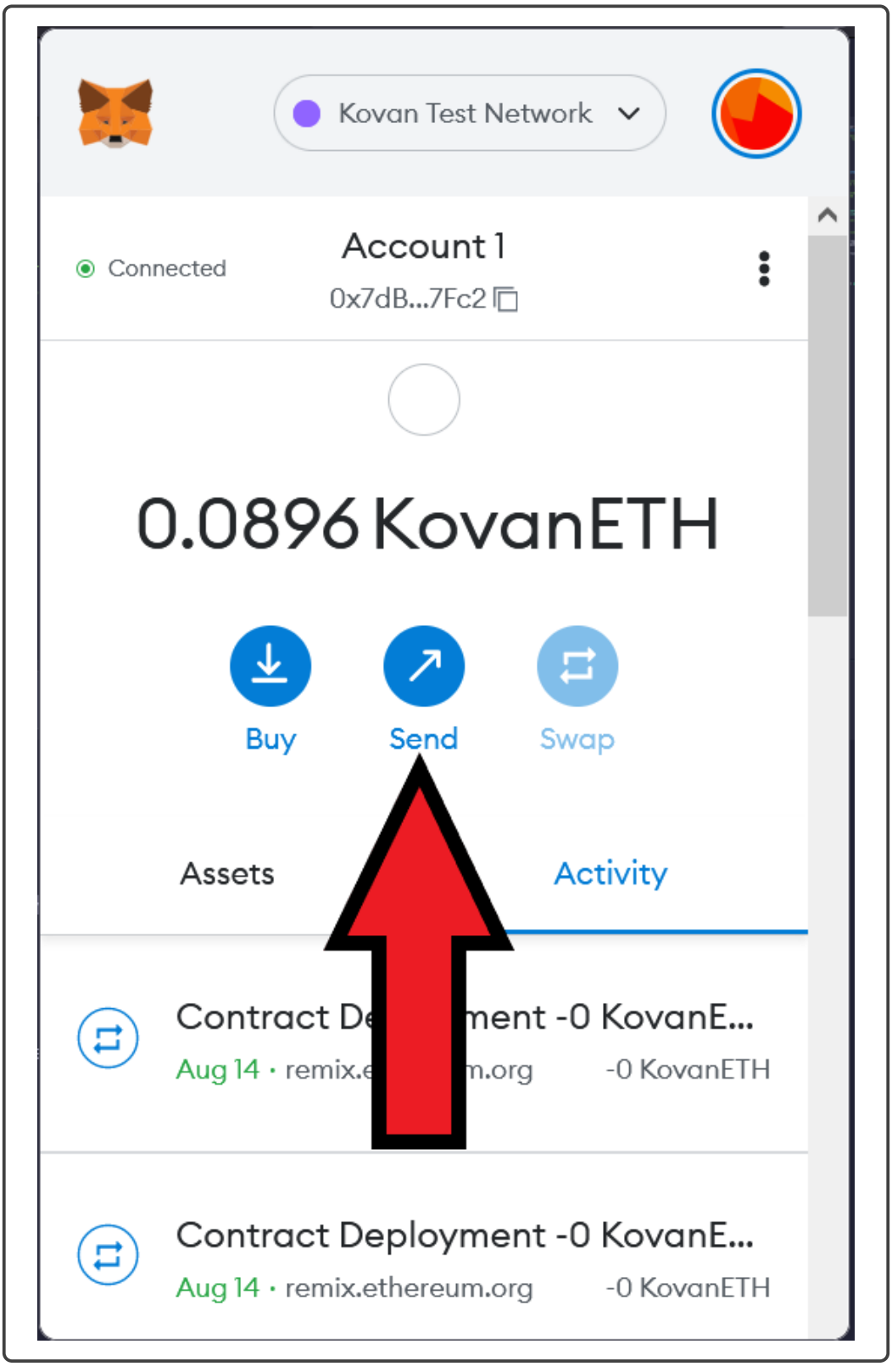


Figure 22: Send tokens to the contract

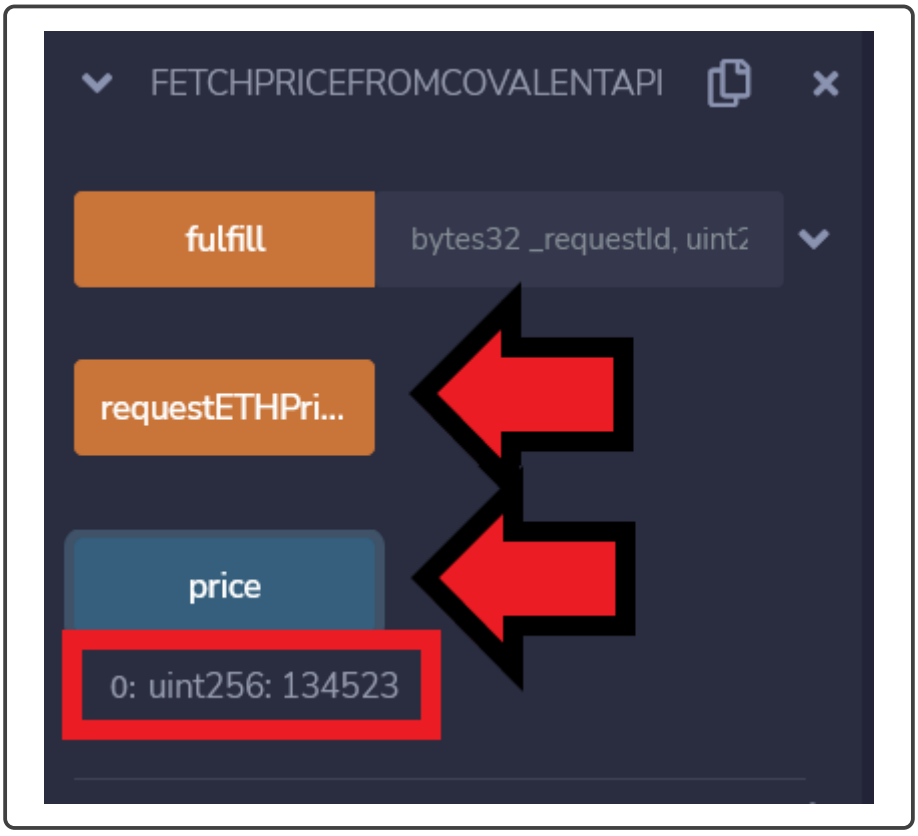


Figure 23: Run Oracle contract

13. The gas cost will be displayed in the metamask view, click on confirm to deploy the contract in Kovan Test net
14. Wait until the terminal return a green check 12
15. No you can try the stop function and the resume function 25.
16. Stop fees in image 26 and the resume fee in image 27

2.5 Eternal storage Pattern[1]

To deploy the Eternal storage patterns follow the instructions:

1. Verify that you are located in the correct workspace 8.
2. Go to the path contracts /Design_patterns_Dapps/eternal_storage_pattern
3. Open the file eternal-storage.sol
4. In the left panel click on the solidity icon (compiler) 9
5. Verify that the version displayed in the compiler is the same version as the version displayed at the beginning of the file "pragma solidity 0.8.7".
6. Click on the "Compile eternal-storage.sol"
7. After the compilation is ready go to the "Deploy & run transactions " section in the left panel 10.
8. In the first option "Environment" select "Injected Provider - Metamask"
9. Connect metamask with remix, clicking on Next and Connect.
10. Then go to the contract section and select the contract Eternal_Storage
11. Click on "Deploy"
12. The gas cost will be displayed in the metamask view, click on confirm to deploy the contract in Kovan Test net28.
13. Wait until the terminal return a green check 12
14. Then the contract deployed is displayed in the bottom of the left panel.
15. Copy the address of the deployed eternal storage contract.
16. In the contract section select the Ballot contract, and in the input next to the deploy button paste the address of the eternal storage29.
17. Deploy the contract
18. In the deployed contract section test the functionality making a vote.

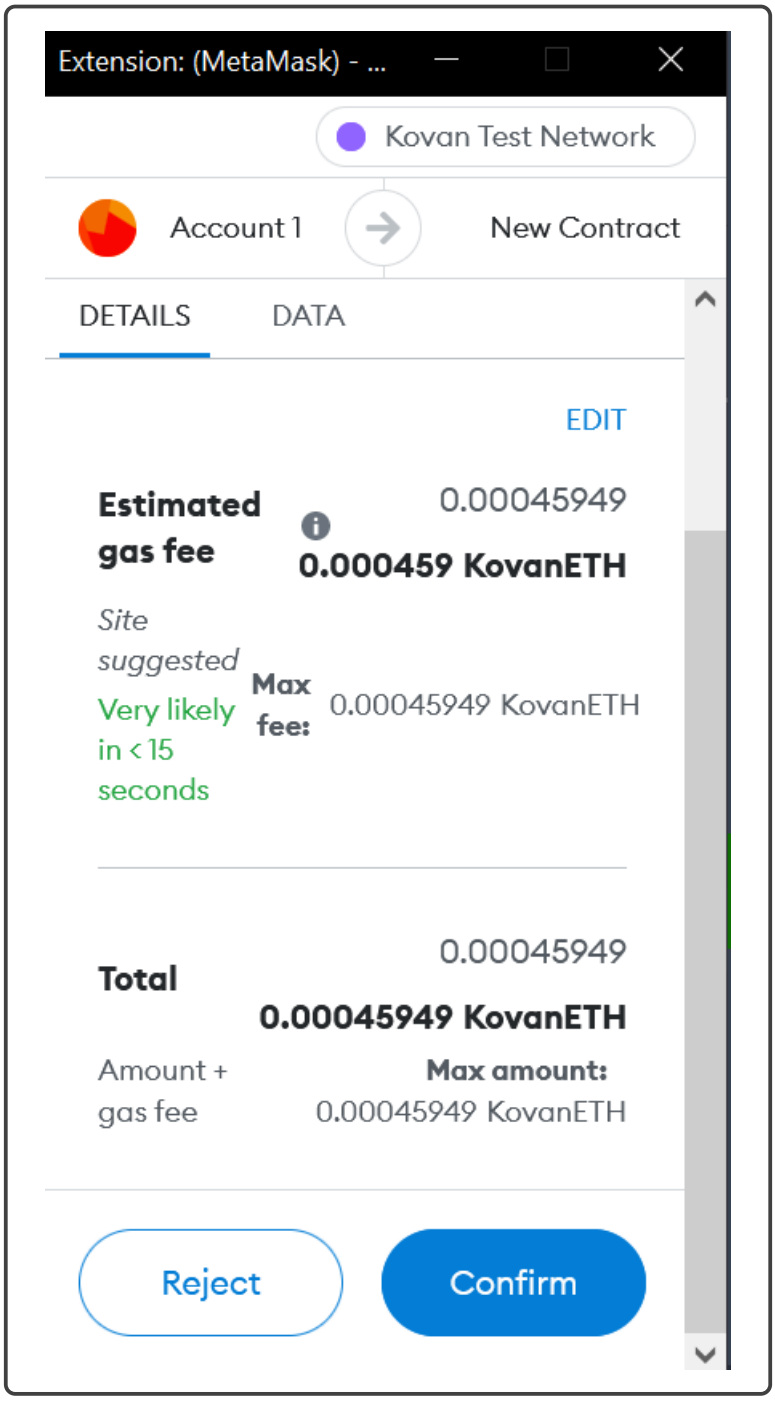


Figure 24: Emergency stop deployment fees

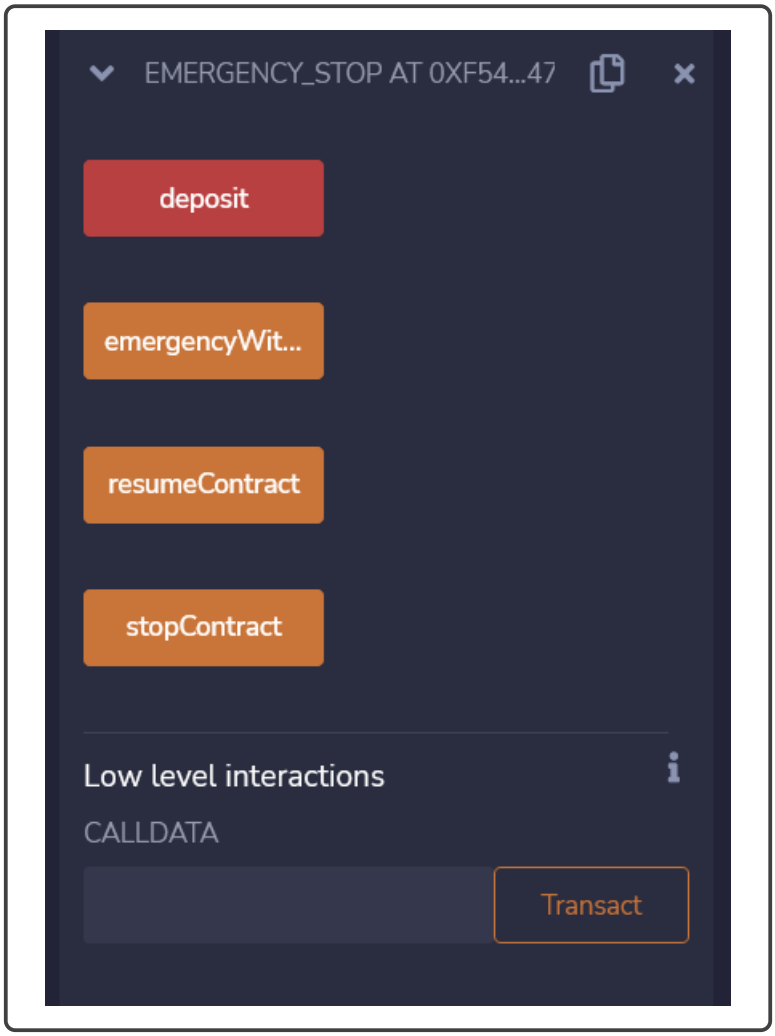


Figure 25: Stop emergency functionalities

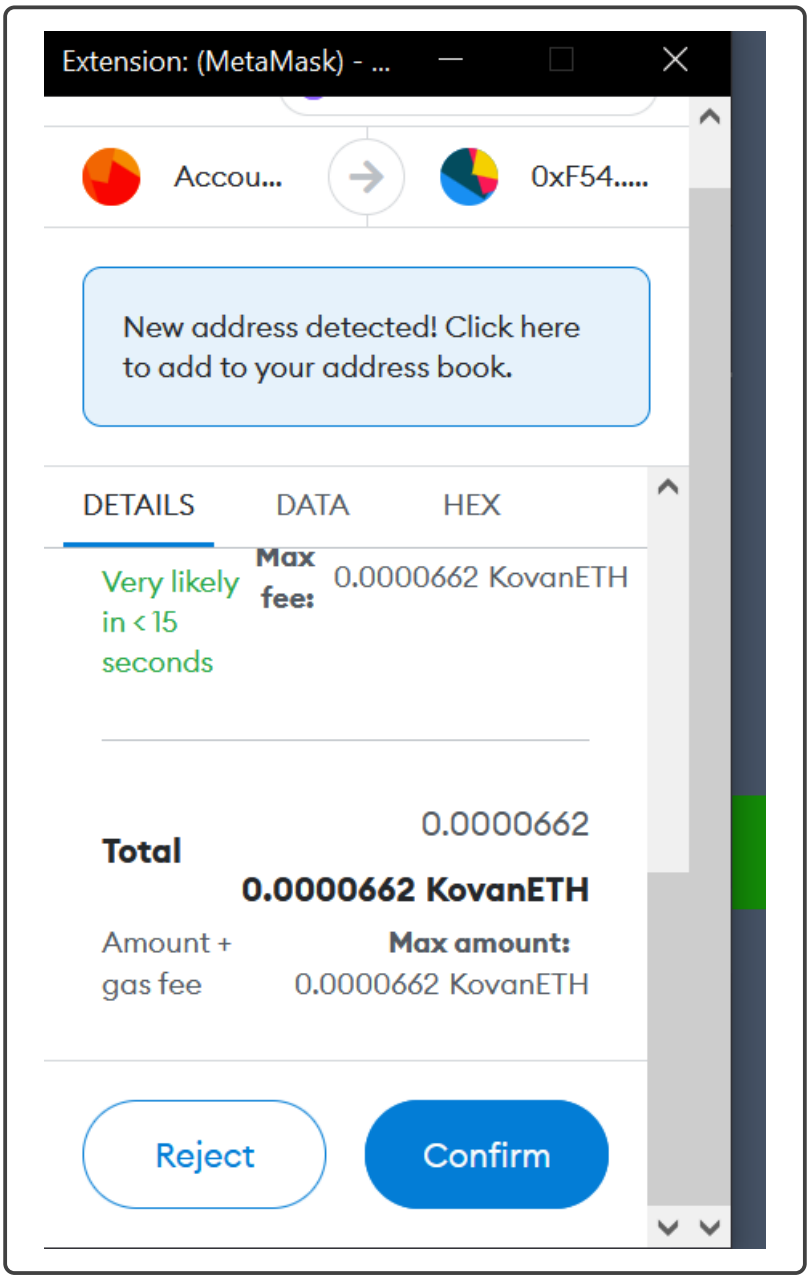


Figure 26: Emergency stop execution fees

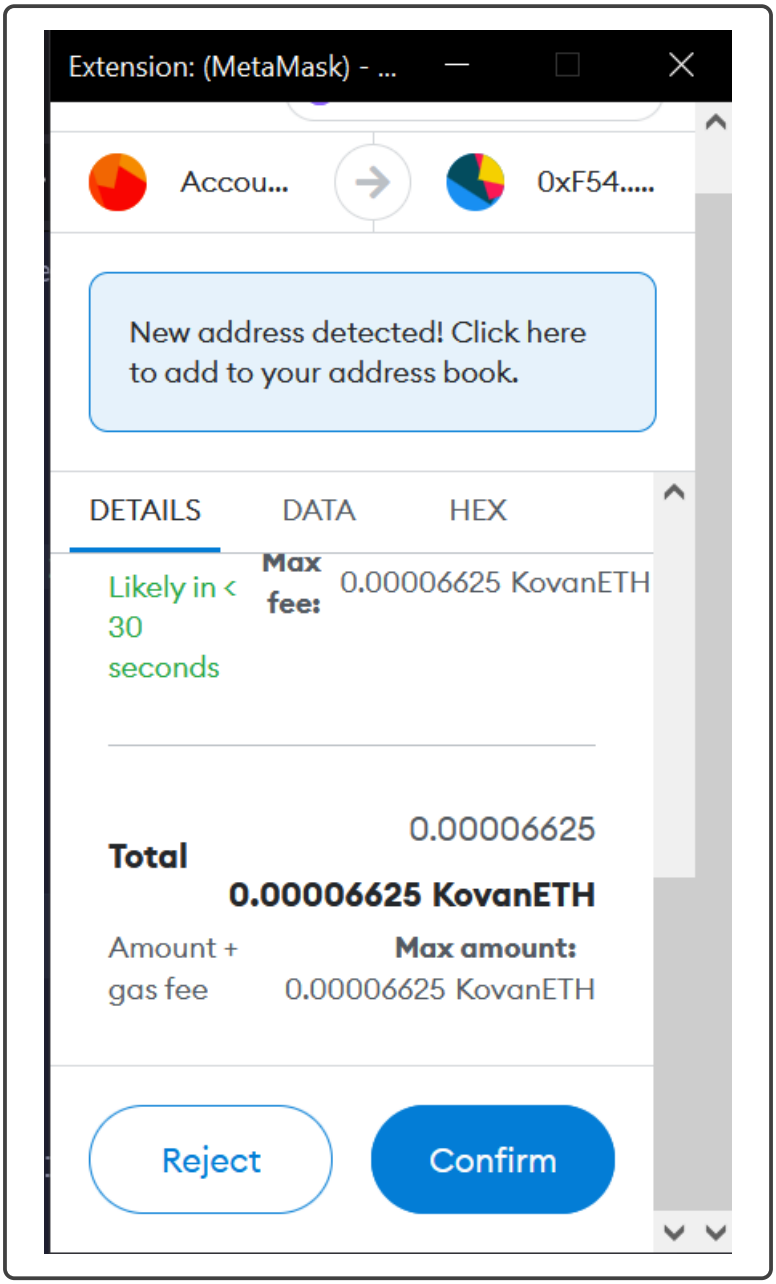


Figure 27: Emergency stop Resume execution fees

19. Now we have to modify the contract to test the storage functionality, to do this we have to modify the line 58 in the contract change the end of the line from 1 to 4, now when the user make a votes it will be adding 4 votes.

20. Compile the contract and deploy it again, remember to add the eternal storage address to the deployment.

21. Then you can get directly the number of votes and you will be getting the votes make by the old contract, but if we make a vote then we will be adding 4 votes at the same time.

2.6 Variable Packing Pattern[3]

In this patter we have to make the comparison between using packing variables and not using it. To deploy the Variable Packing follow the instructions:

1. Verify that you are located in the correct workspace 8.

2. Go to the path contracts /Design_patterns_Dapps/packing_variables_pattern

3. Open the file packing-variables.sol

4. In the left panel click on the solidity icon (compiler) 9

5. Verify that the version displayed in the compiler is the same version as the version displayed at the beginning of the file "pragma solidity 0.8.7".

6. Click on the "Compile packag-variable.sol"

7. After the compilation is ready go to the "Deploy & run transactions " section in the left panel 10.

8. In the first option "Environment" select "Injected Provider - Metamask"

9. Connect metamask with remix, clicking on Next and Connect.

10. Then go to the contract section and select the contract cheapPackExample

11. Click on "Deploy"

12. The gas cost will be displayed in the metamask view, click on confirm to deploy the contract in Kovan Test net30.

13. wait until the terminal return a green check 12

14. Then the contract deployed is displayed in the bottom of the left panel.

15. Dow we can deploy a structure without using the packing pattern.

16. Deploy ExpensiveWithoutpackExample 31

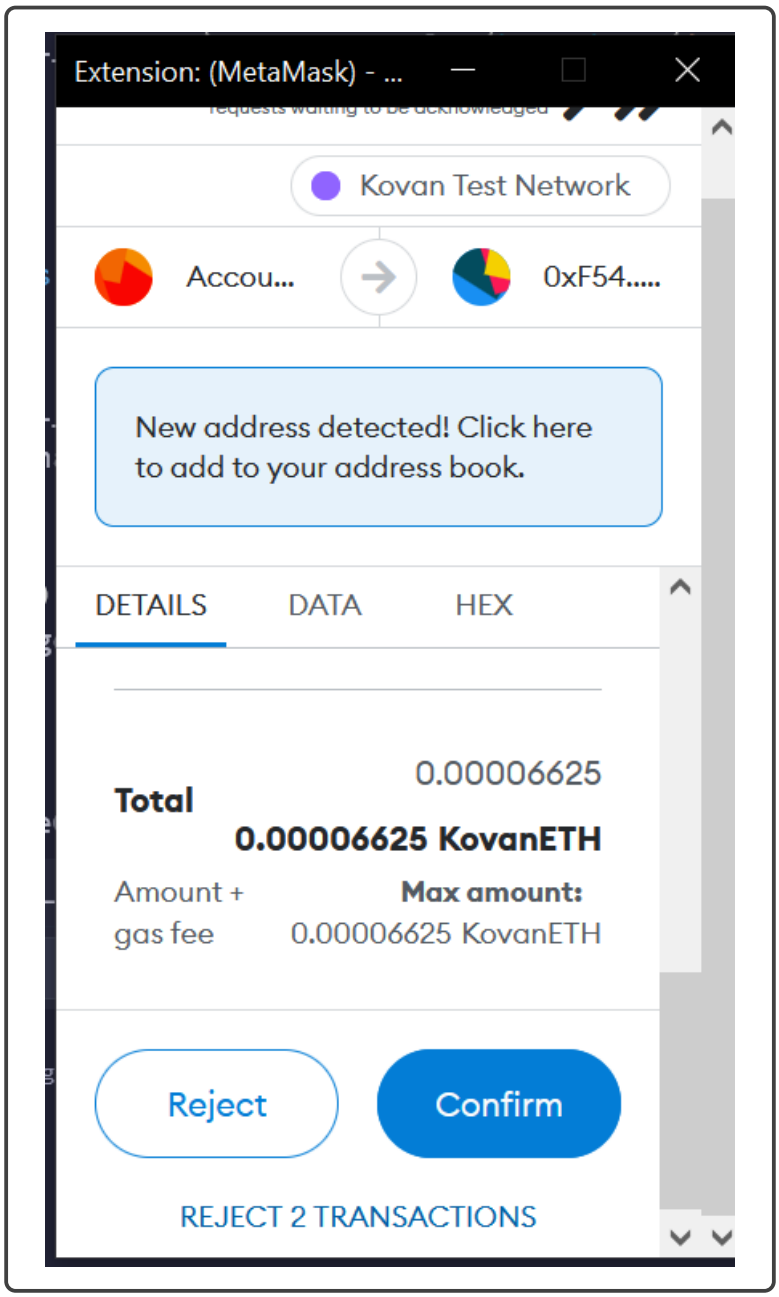


Figure 28: Eternal storage fees

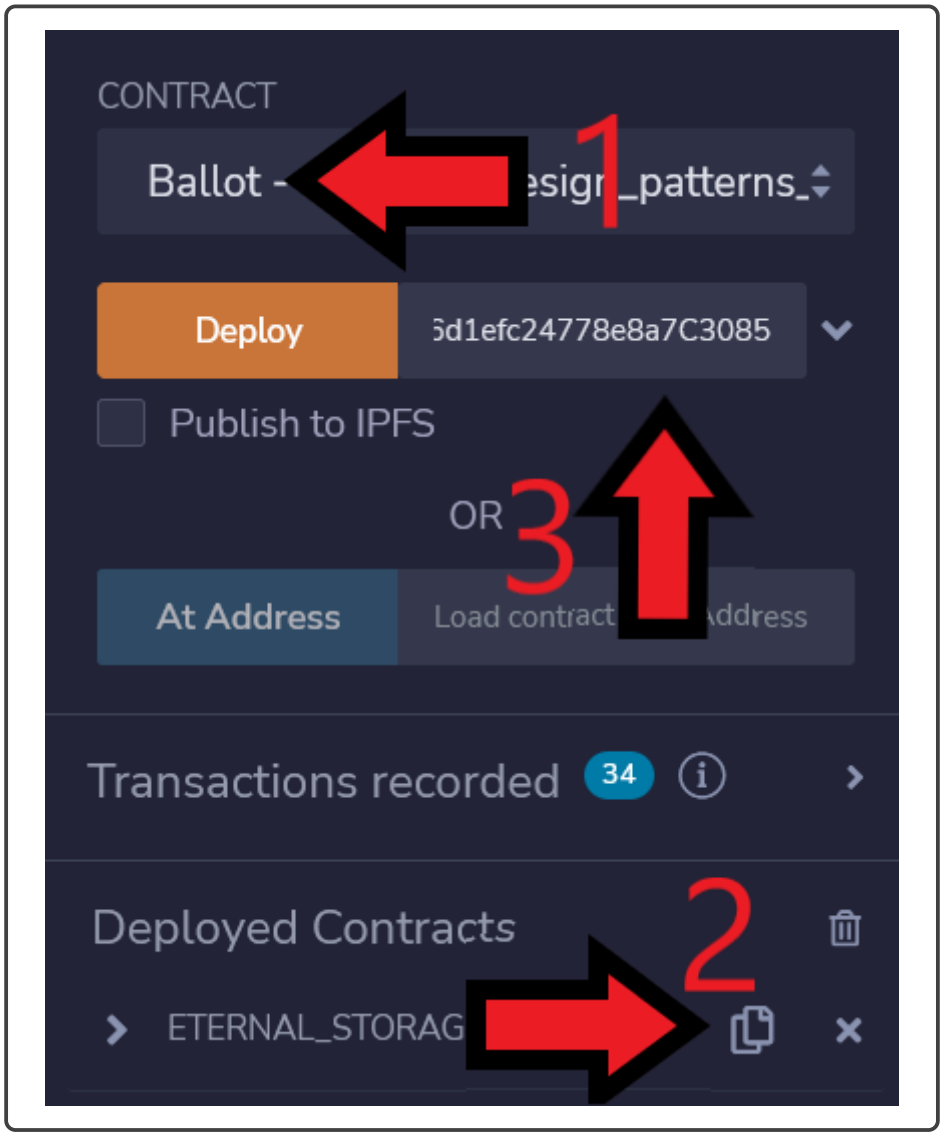


Figure 29: Deploy process contract Ballot

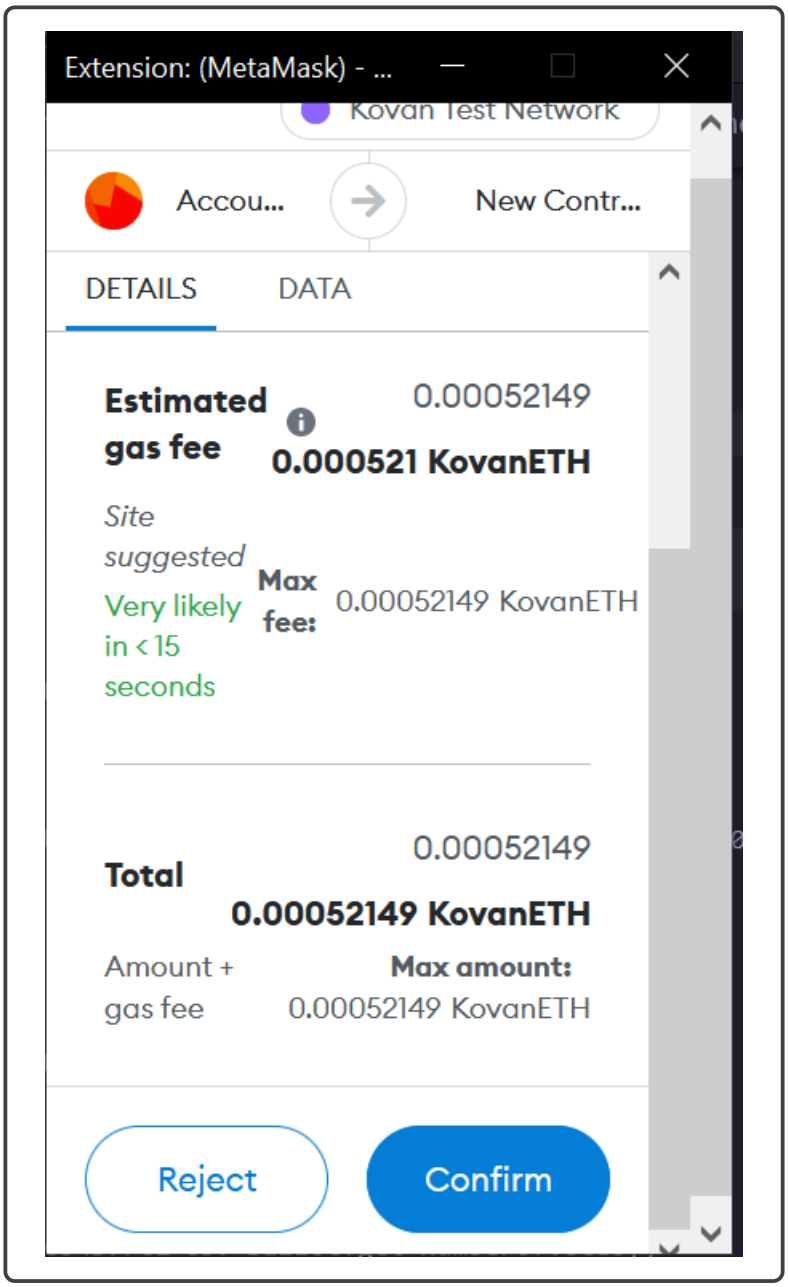


Figure 30: Variable Packing fees

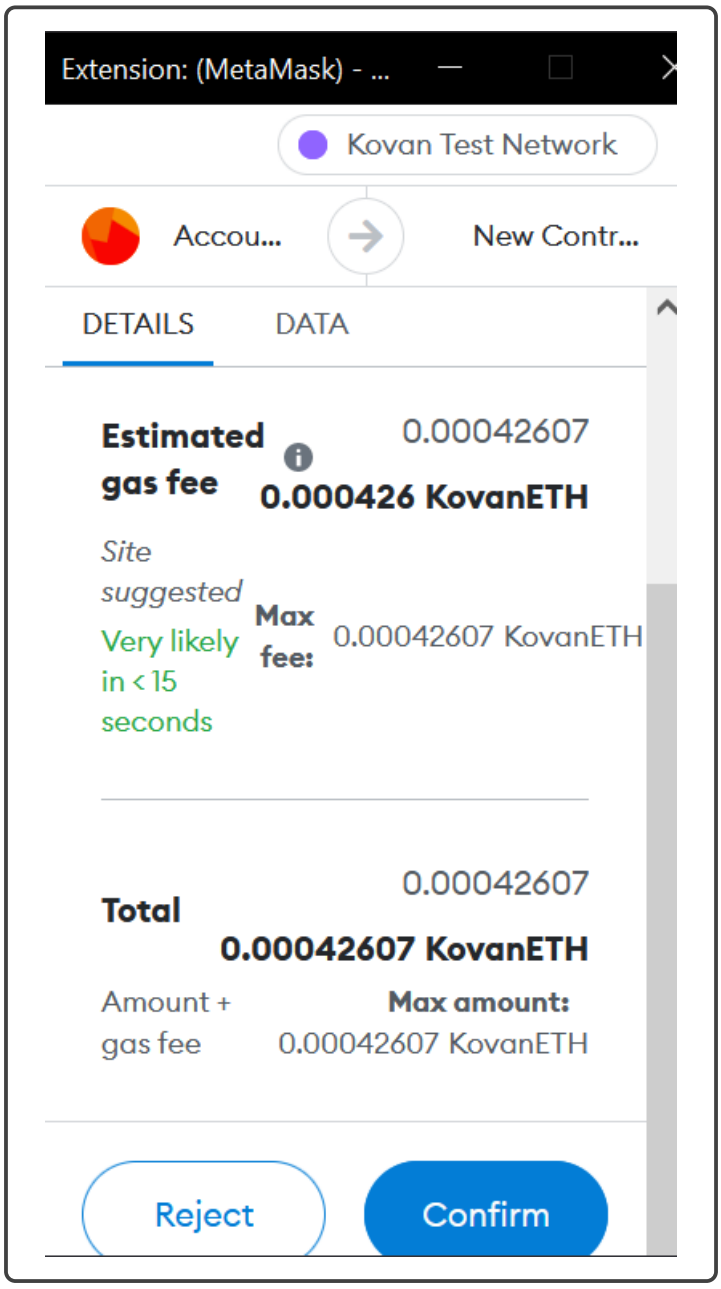


Figure 31: Expensive variable fees

A Increase the gas limit

To increase the gas limit follow the instructions:

1. Before confirming the transaction in the metamask view, click on the "EDIT" label 32
2. Click on the button "Edit Suggested gas fee"
3. Increase the gas limit to 50000.

B Add Link Tokens to the wallet

In the case of the Link tokens are not in your wallet after you requested in the chainlink faucet follow the instructions to added manually.

1. Open Metamask
2. Click on import Tokens 33
3. Add the address contract of the token (the contract where the token was created) 0xa36085F69e2889c224210F603D836748e7dC0088 and click on Add Custom Token.
4. After that icon of LINK tokens will be displayed, click on import token 34.

References

[1] V. C. Bui, S. Wen, J. Yu, X. Xia, M. S. Haghighi, and Y. Xiang, "Evaluating upgradable smart contract," in *2021 IEEE International Conference on Blockchain (Blockchain)*. Melbourne: IEEE, 2021, pp. 252–256, jCR Impact factor 2021: 3.28.

[2] B. Massimo, P. Livio, R. Kurt, B. Joseph, M. Andrew, P. Y. Ryan, T. Vanessa, B. Andrea, S. Massimiliano, P. Federico, and J. Markus, *An Empirical Analysis of Smart Contracts: Platforms, Applications, and Design Patterns*. Cham: Springer International Publishing, 2017, CORE2021 Rank:A.

[3] L. Marchesi, M. Marchesi, G. Destefanis, G. Barabino, and D. Tigano, "Design patterns for gas optimization in ethereum," in *2020 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*, London, 2020, pp. 9–15, CORE2021 Rank: C.

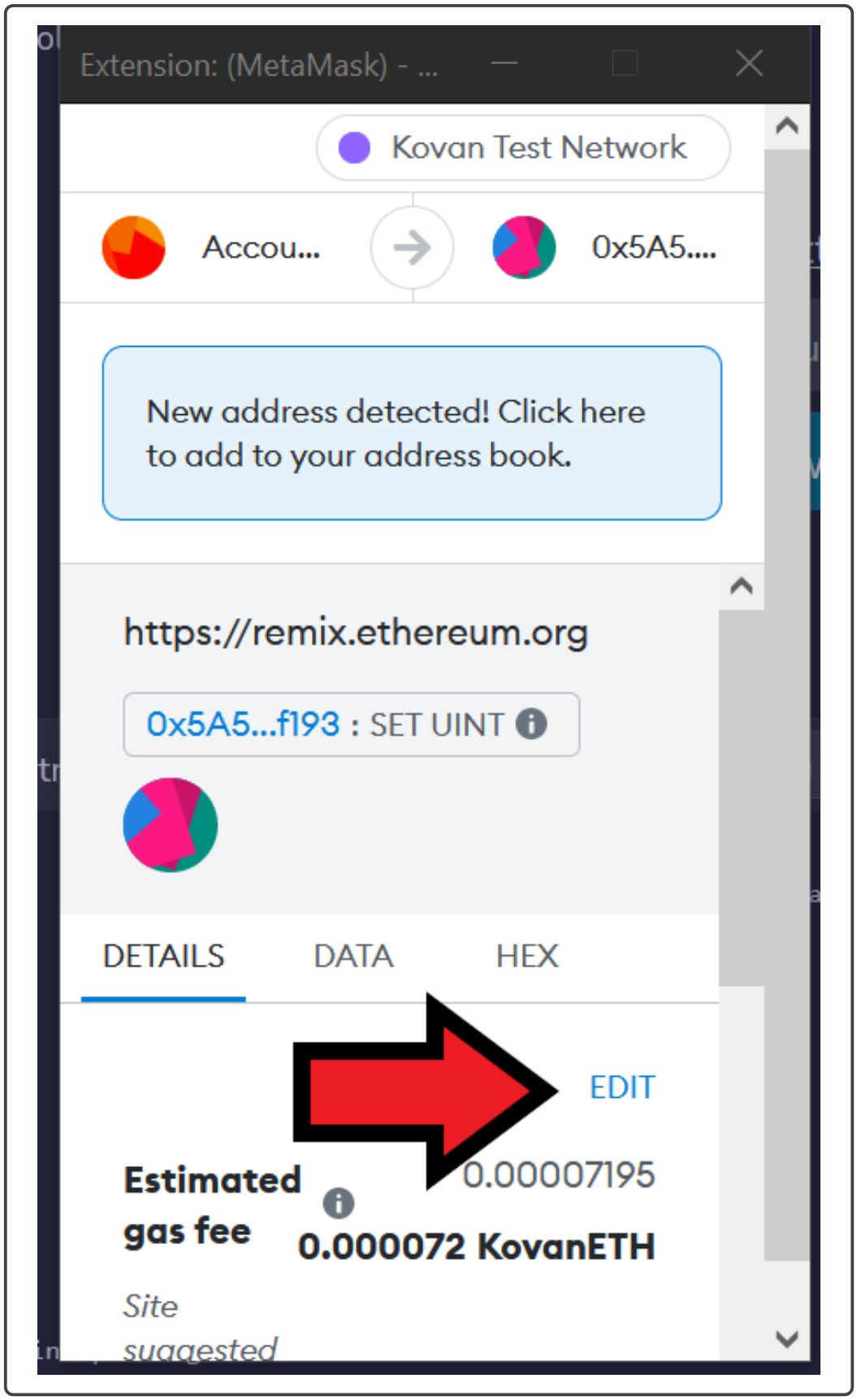


Figure 32: Edit Gas button

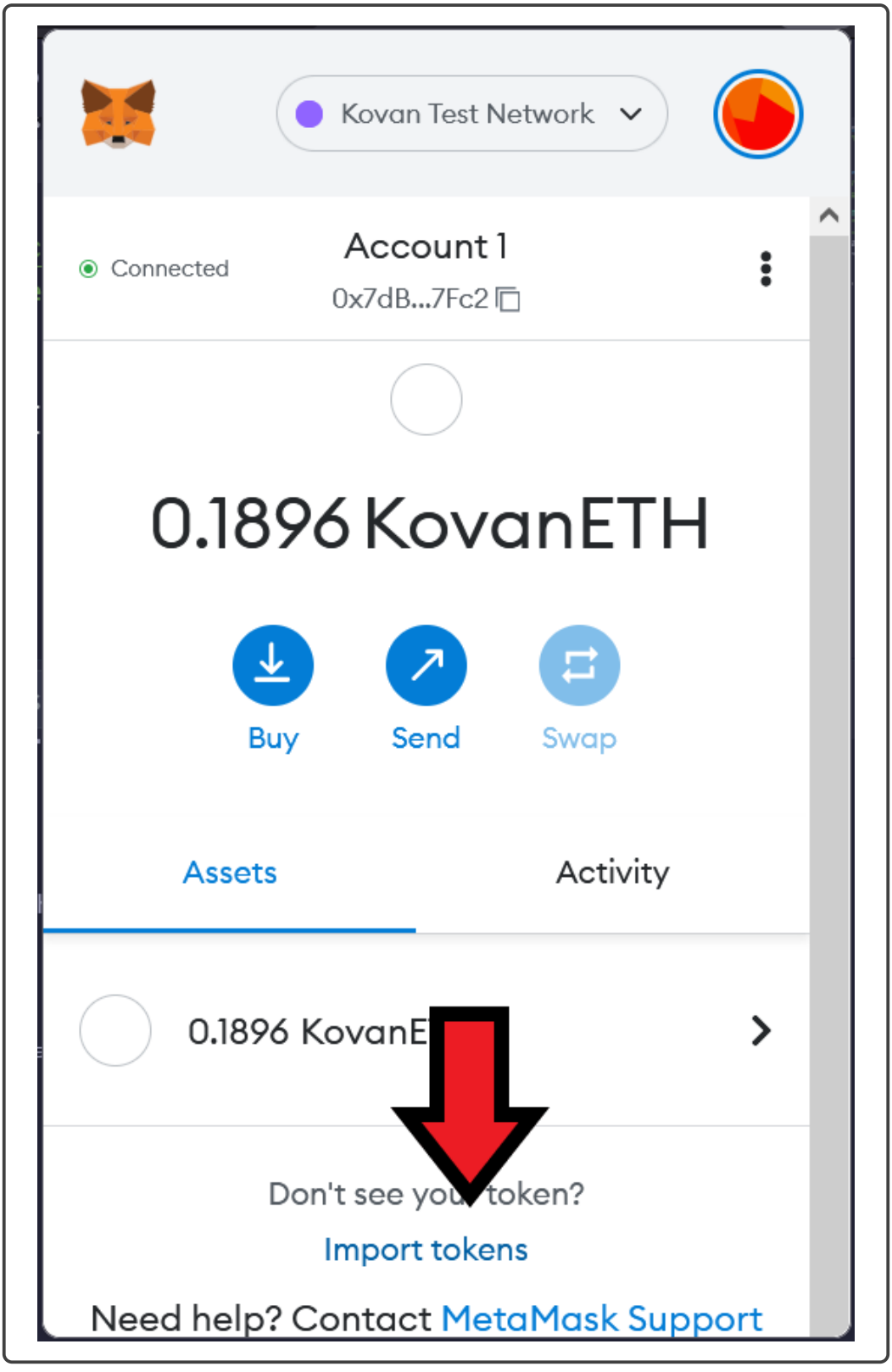


Figure 33: Import Tokens

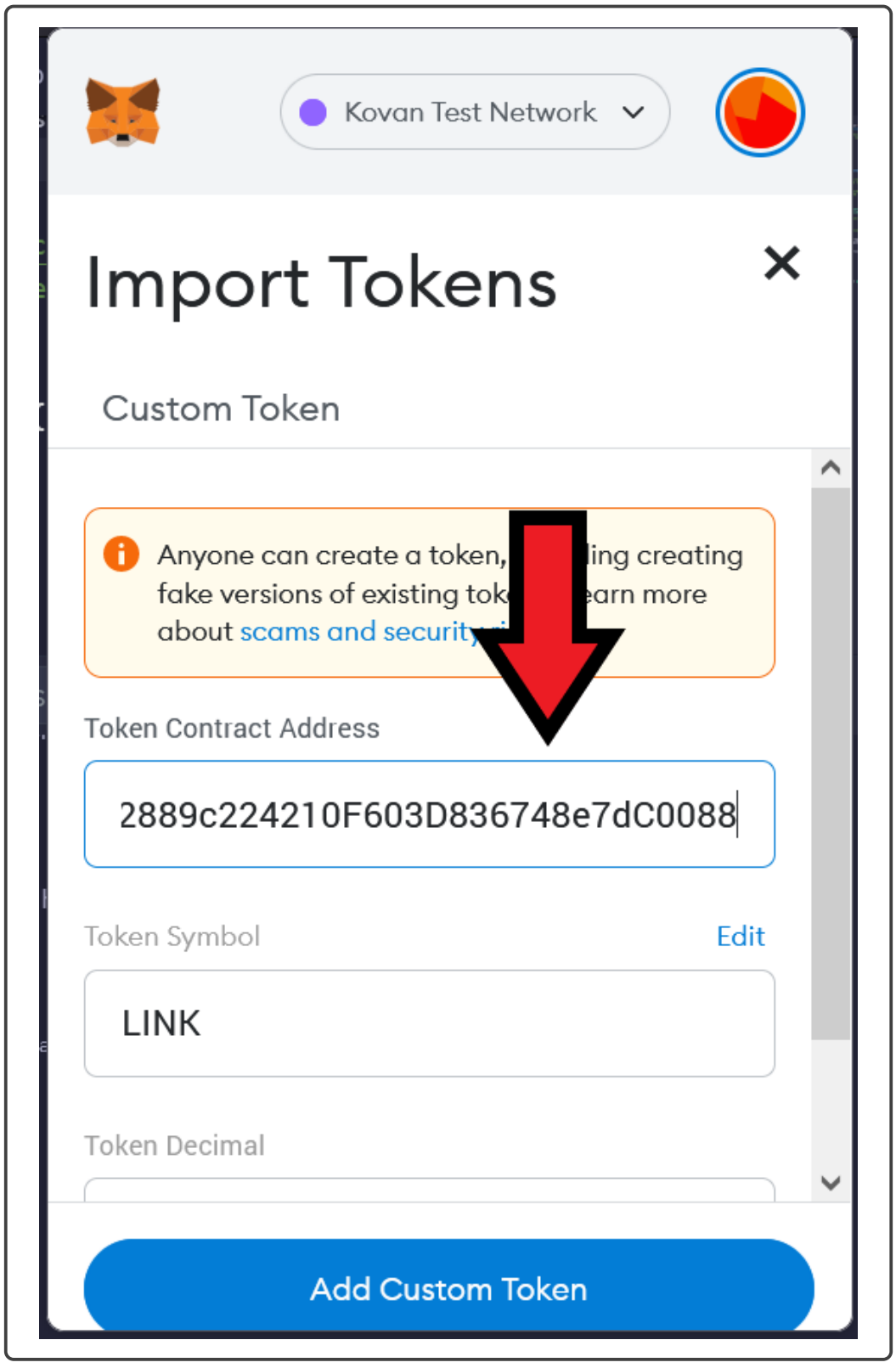


Figure 34: Edit Gas button