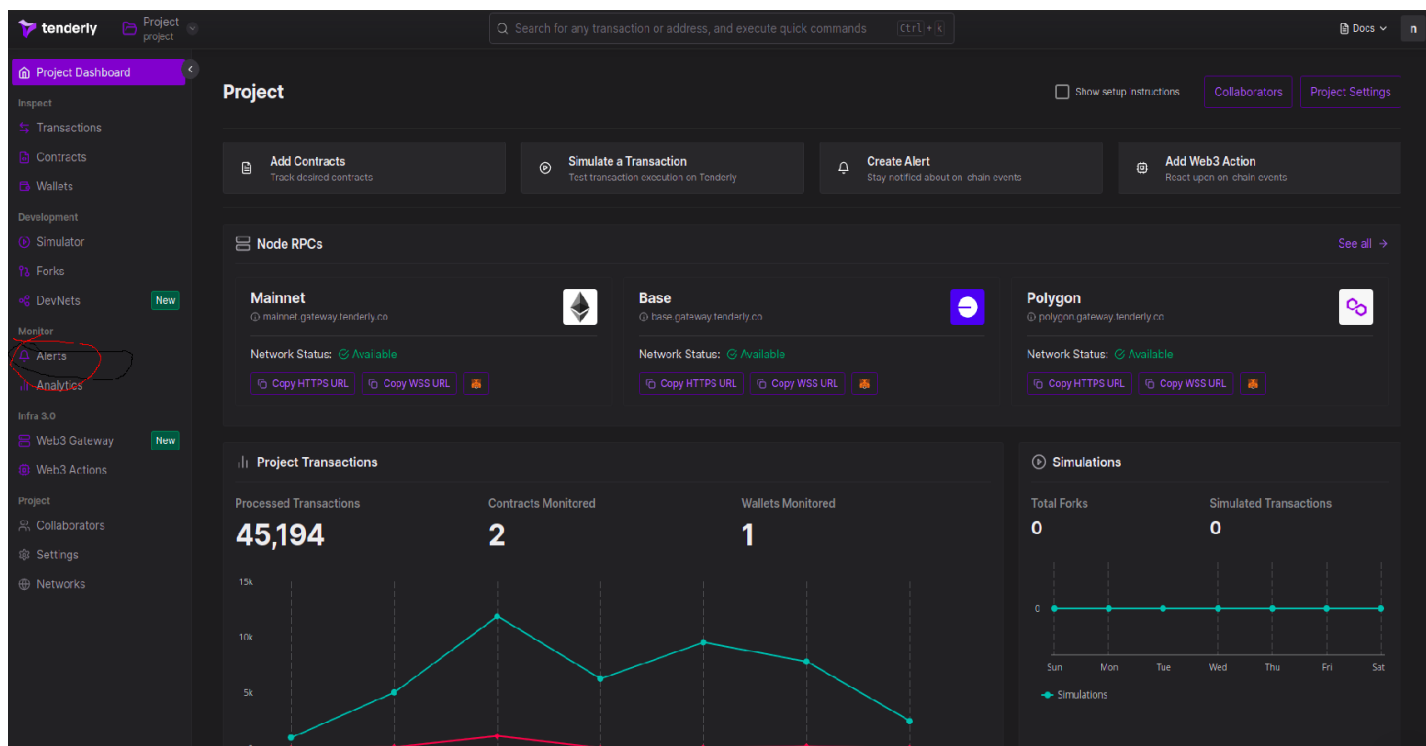


Tenderly Documentation

The purpose of this document is to detail how and why the third-party tool Tenderly is used in the system, and how to go about making use of its facilities.

Tenderly is a Web3 development platform that helps build, monitor, test and operate smart contracts. It offers several features including contract debuggers, simulators and analytics. The feature that we make use of is Tenderly Alerts, which sends notification for specific events on the blockchain.



We require a method to get notified everytime a certain event on a specific smart contract is emitted.

More information on its emitted events of the contract can be found on arbiscan.

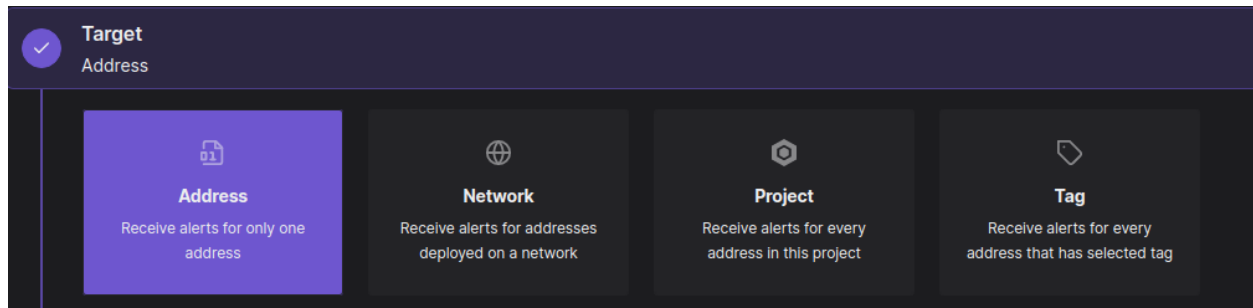
As discussed in the documentation of the dashboard code, the 'IncreasePosition' event notifies us when a position increase (including position openings) is made, and the 'DecreasePosition' event notifies us when a position decrease/close is made.

For example, here we can see an 'IncreasePosition' event emitted here in the 'events' section of the [arbisacan](#) page of the contract.

Next we must choose the addresses for which the alert will be triggered.

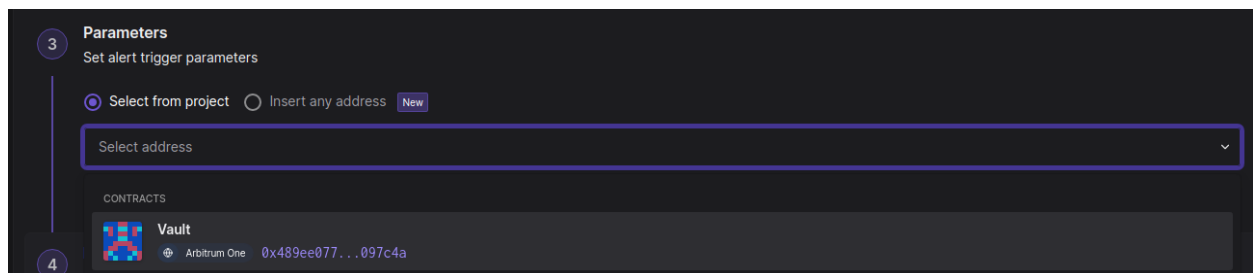
Due to only the singular gmx address

'0x489ee077994b6658eafa855c308275ead8097c4a' being involved, we choose to receive alerts for only one address.



As the trigger parameter, we choose the address

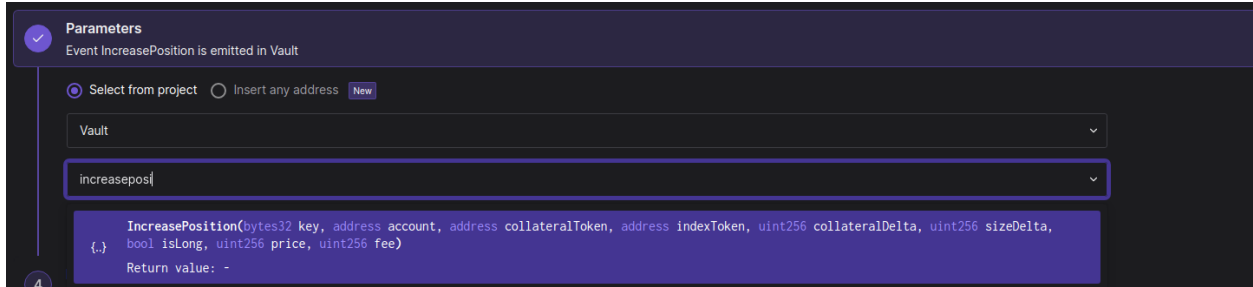
'0x489ee077994b6658eafa855c308275ead8097c4a'.



Once the contract is selected, one must choose the event they are interested in listening for.

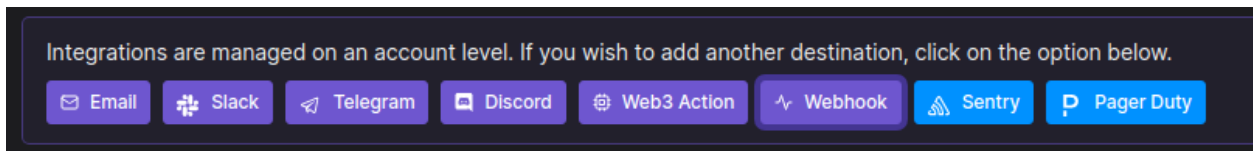
Tenderly automatically extracts all the events from the smart contract and displays them in a drop down list.

One can then simply choose the desired event. For instance, one can see the 'IncreasePosition' event below.



Next, we must choose the destination to which the notification will be sent.

Here, we choose the 'webhook' option.



The 'webhook' option allows us to send alert data to a custom webhook endpoint. This endpoint can be anything that can receive http requests.

So when an alert is triggered, Tenderly will send a JSON formatted payload to the webhook endpoint. One can see the structure of the json payload in the dashboard documentation.

In Tenderly, webhook receive data about any event related to a smart contract as it happens. A main advantage of using webhooks is that they receive real-time notification about alerts, allowing one to take actions quickly as they happen.

To create a new webhook, one needs to provide a webhook name and a webhook url.

The name can be anything that would help one identify the webhook.

Now the url to be given would be the url to which the notification and json payload would be sent.

In our current system, let us take the open trades dashboard code for example (more info in the AWS documentation).

We are running the code on the AWS lightsail instance with the static ip address '3.109.24.178'.



And the code is running on port 443.

```
if __name__ == '__main__':  
    #app.run(host='0.0.0.0', port=443)  
    http_server = WSGIServer('', 443), app)  
    http_server.serve_forever()
```

Our URL webhook to use will thus be '<http://3.109.24.178:443>'

So in this case, one can set the webhook as follows and add the webhook

Add Destination

Select Destination

Webhook

Set up your webhook endpoint to receive live events from Tenderly or [learn more about Webhooks](#).

new_instance_443

<> http://3.109.24.178:443

Send a test webhook to see if the provided URL can receive Tenderly events.

Send test webhook





Description (Optional)

Add webhook







Advanced Configuration

Currently Active Alerts:

Currently, there are two alerts active to ensure the correct functioning of the system.

Alerts			Have suggestions or feedback?
	Event DecreasePosition emitted in Vault	Enabled	
	Event Emitted 2 months ago (05/07/2023 16:21:26)	<input type="checkbox"/> Default	
	Event IncreasePosition emitted in Vault	Enabled	
	Event Emitted 2 months ago (05/07/2023 02:03:00)	<input type="checkbox"/> Default	

The 'IncreasePosition' alert has two destinations - to port 443 and port 80 of the AWS instance.






	new_instance_443		
	Webhook Id		
	new_instance_80		
	Webhook Id		
Add more destinations			

Port 443 is where the open dashboard code is running, and port 80 is where the live trading code is running. Both of these require alerts on when the 'IncreasePosition' event is emitted.

The 'DecreasePosition' alert has one destination, to port 20, where the closing trades dashboard is being run.

In the 'History' section' one can view the notification history of all the alerts.

One issue observed with Tenderly is that at times it sends multiple notifications for the same event in the same transaction

Alert	Tx Hash	When	Network
Event IncreasePosition emitted in Vault	0xd7c31743ab...9e7348	17/08/2023 09:25:16	 Arbitrum One
Event IncreasePosition emitted in Vault	0xd7c31743ab...9e7348	17/08/2023 09:25:16	 Arbitrum One
Event IncreasePosition emitted in Vault	0xd7c31743ab...9e7348	17/08/2023 09:25:16	 Arbitrum One
Event IncreasePosition emitted in Vault	0x9bf22c72fe...a1b029	17/08/2023 09:17:45	 Arbitrum One
Event IncreasePosition emitted in Vault	0x9bf22c72fe...a1b029	17/08/2023 09:17:45	 Arbitrum One
Event IncreasePosition emitted in Vault	0x9bf22c72fe...a1b029	17/08/2023 09:17:45	 Arbitrum One

This case may have to be dealt with. One solution which has been implemented in the dashboard code is to store the processed transaction hashes, so if the code receives the same transaction again, it is skipped.


```
tx_hash=data["transaction"]["hash"]

with open("tx_hash_store.txt", "r") as f:
    content = f.read()
    values = content.splitlines()
    if tx_hash in values:
        return 'OK', 200 # the transaction has already been processed before

    else:
        with open("tx_hash_store.txt", "a") as f2:
            f2.write(tx_hash+"\n")
```

The above segment has been discussed in the dashboard documentation as well.

-

The document thus sheds light on where and why Tenderly is utilized in the system in general, and how one can go about setting up alerts to receive notifications on blockchain events in particular.

-----X-----