Hydra POC First Run

This document is related to cPoker Hydra Case Study: Implement Interactive dApp, and explains how to do the first run of the Hydra POC on MacOS.

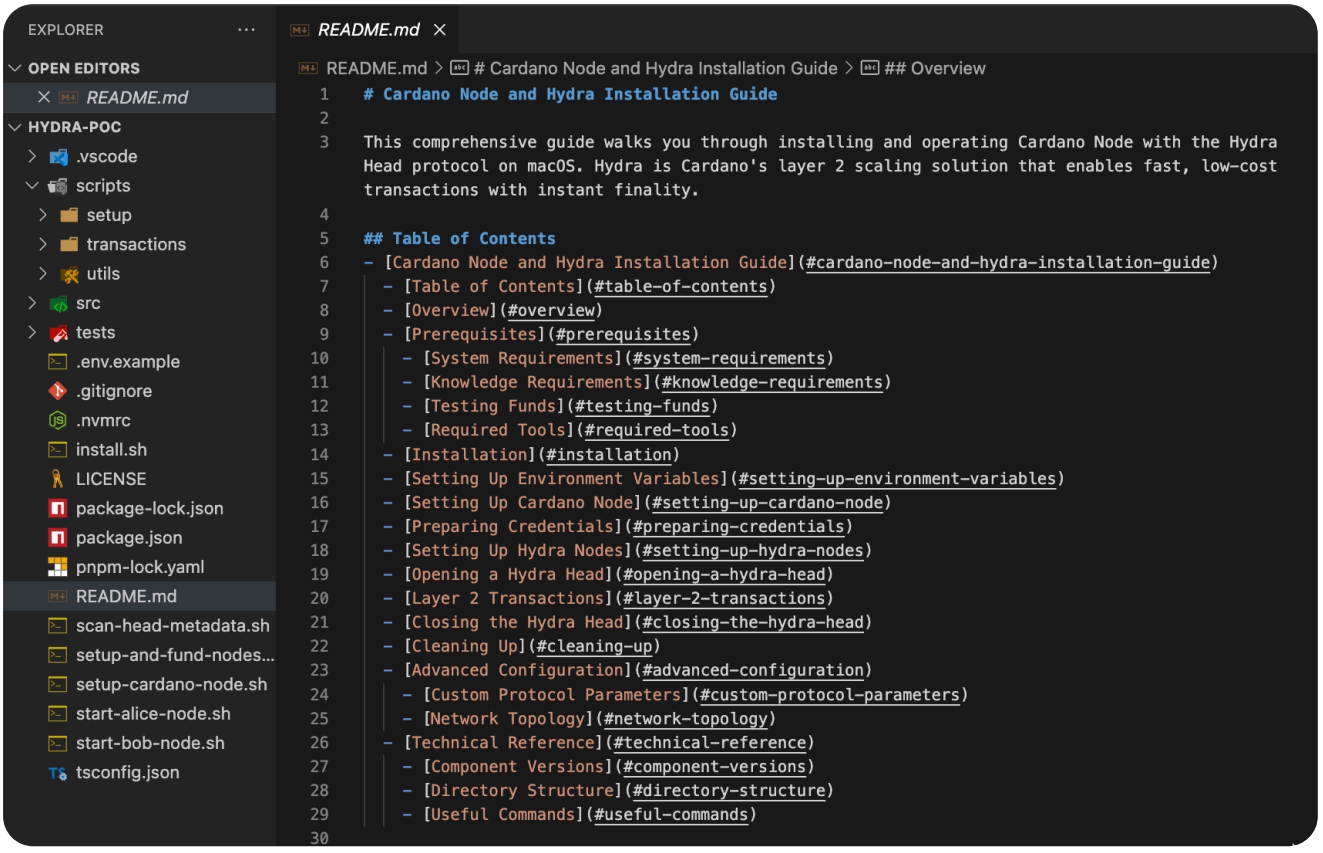
The contents of this article are coming from https://github.com/Cardano-After-Dark/hydra-poc > main branch (commit b7f813c).

Getting started

Prepare a folder and clone the project

```
mkdir ~/temp && cd ~/temp
git clone git@github.com:Cardano-After-Dark/hydra-poc.git
cd hydra-poc
```

Open the project to consult the readme for the details



Installing and Setting Up Env Variables

Run the installation scripts and set up the environment variables to have a running environment

```
./install.sh
```

```
./scripts/utils/set-env-vars.sh
```

Your output should look like this

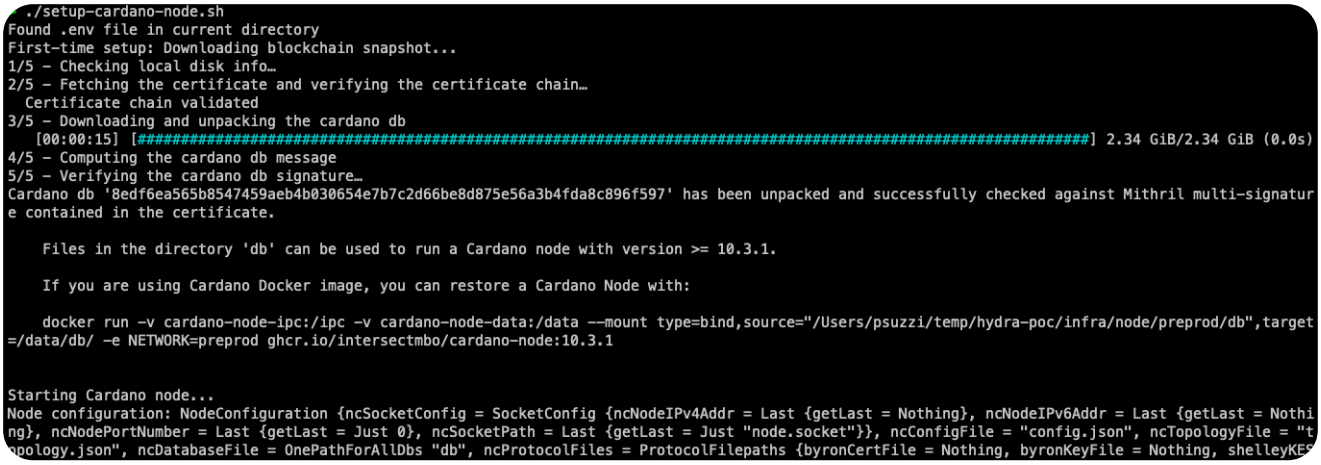
Setting up Cardano Node

Set up and start the Cardano node:

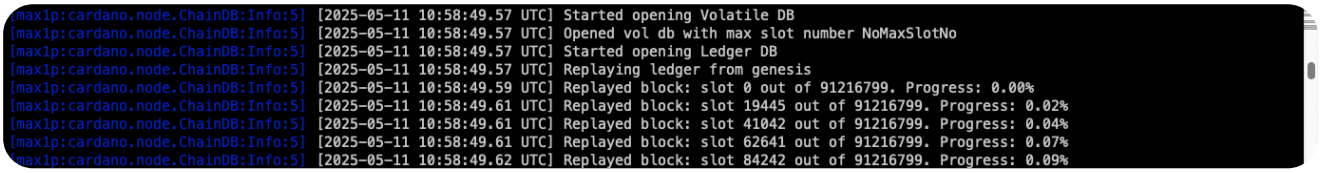
```
./setup-cardano-node.sh
```

The setup should be relatively quick:

The initial download and unpack of the cardano db should be done in a minute, if you have a fast internet connection.



The replay of blocks from 0 to 100% should take 5-10 minutes, on a good machine with enough RAM.



...

When completed, leave the terminal open with the cardano node running

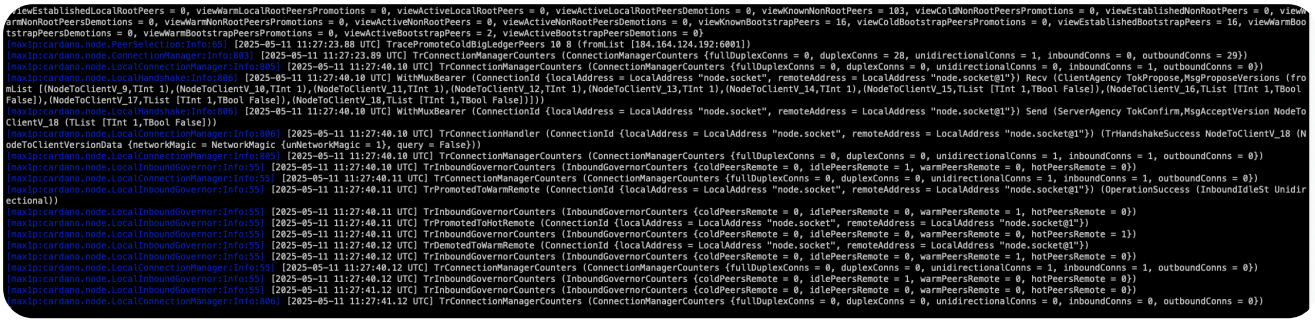
Note: if requested, grant terminal the permission to access local network.

Open a new terminal, and query the tip of the running node

```
./scripts/utils/query-tip.sh
```

In your new terminal, you should see something like this

While in the terminal running the cardano node, you should notice that the other process queried the node.



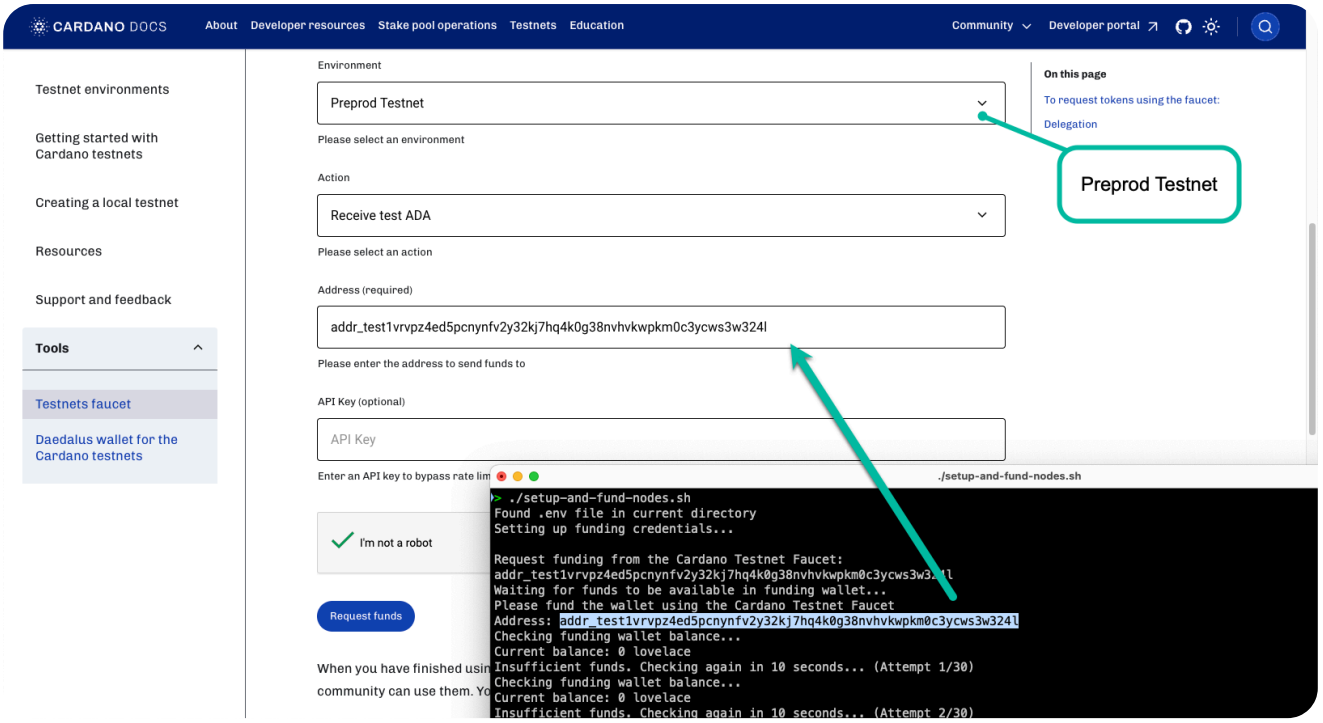
Prepare credentials for A and B

We're going to add two hydra nodes connected to the cardano node. Each of these node will be owned by a different party, let's call them Alice (A) and Bob (B).

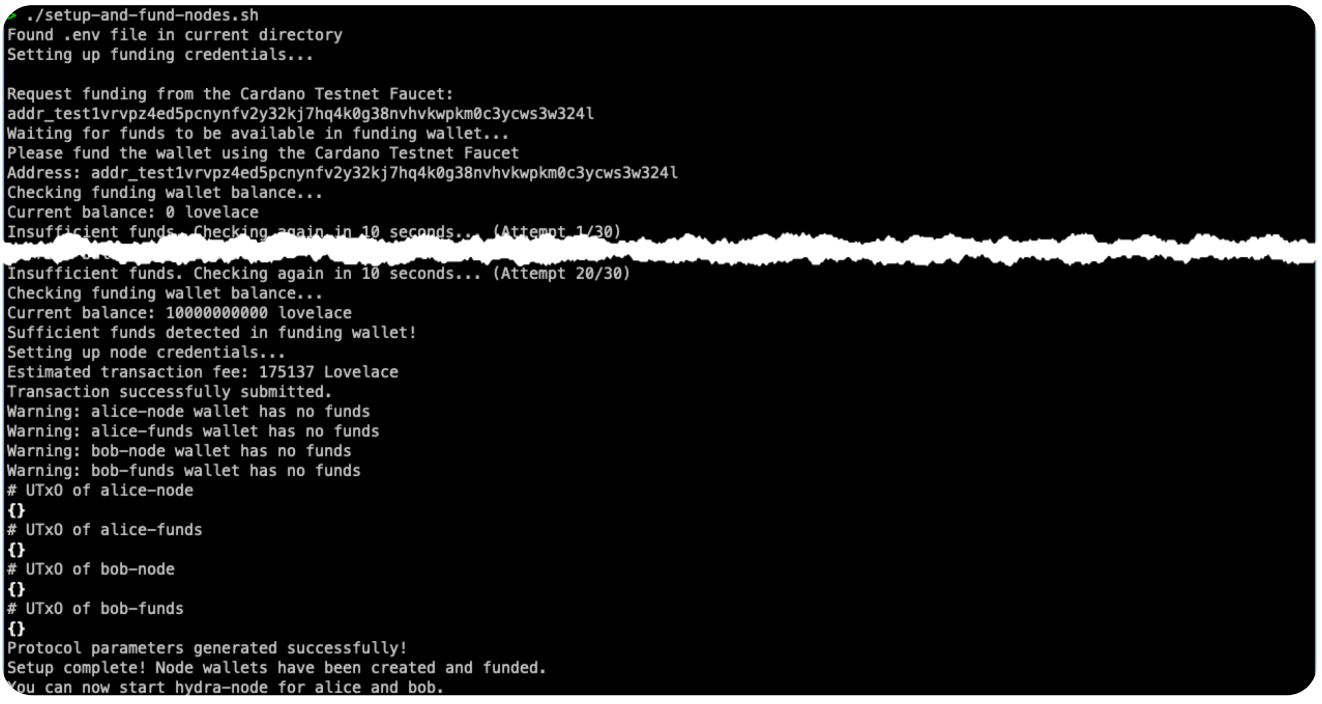
Run the script

```
./setup-and-fund-nodes.sh
```

The script will tell you to open Cardano Testnet Faucet and select to send test funds into a specific address in Preprod Testnet. With this operation, we'll send funds needed to the test operations we're running later.



When complete, the script will tell you the funds are sent.



Let's complete this step by verifying the funds are in place

```
./scripts/utils/query-node-funds.sh
```

The output should look like the one below

Launch two Hydra Nodes for Alice and Bob

Open a new terminal and start the Alice Hydra node

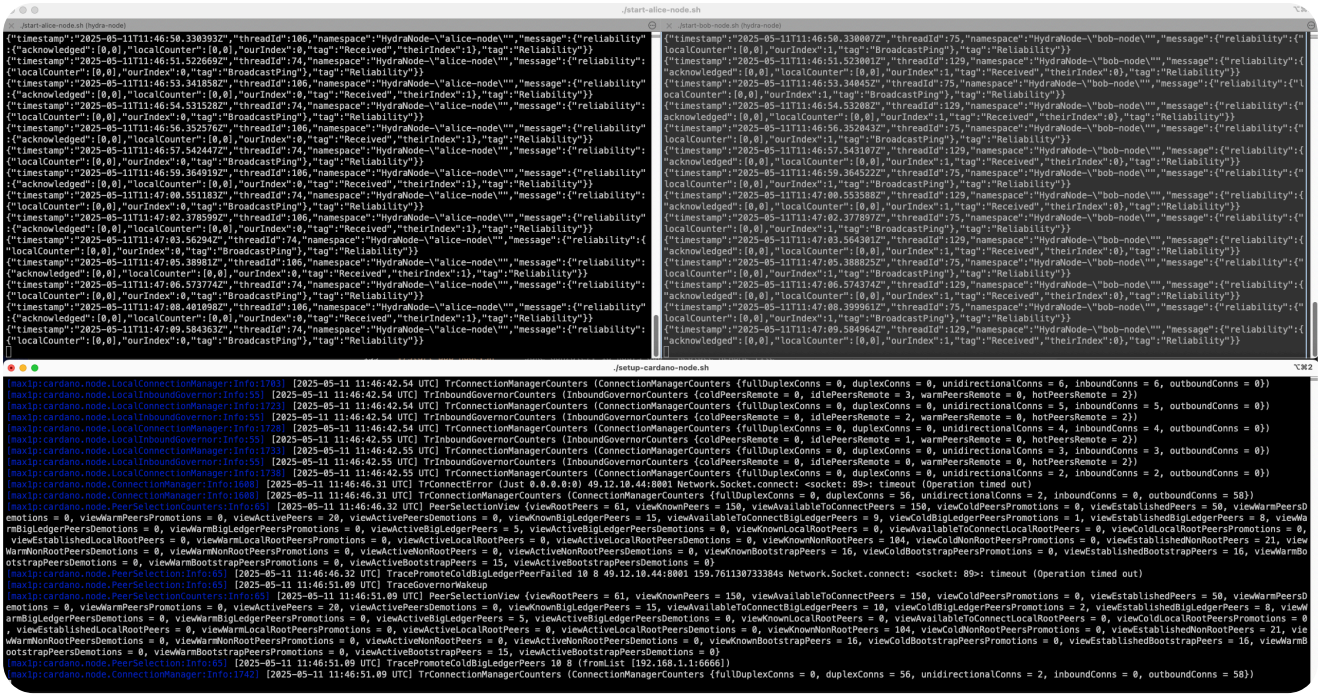
```
./start-alice-node.sh
```

Open a new terminal and start the Bob's Hydra node

```
./start-bob-node.sh
```

Now, you should have

- a terminal running the cardano node
- a terminal running the Hydra node for Alice
- a terminal running the Hydra node for Bob



Let's keep all this infrastructure running, and launch two more terminals to verify the nodes are running by connecting to their websocket APIs

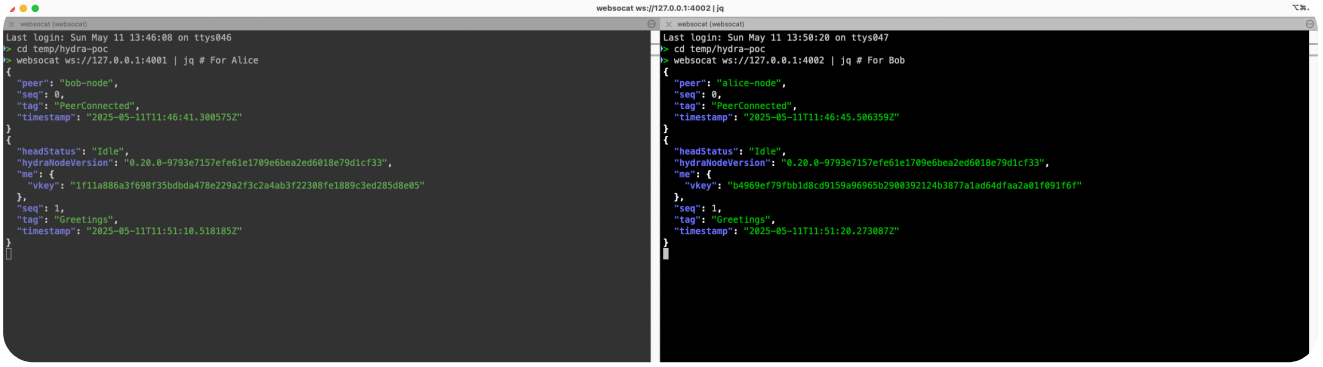
Open a terminal

```
websocat ws://127.0.0.1:4001 | jq # For Alice
```

Open another terminal

```
websocat ws://127.0.0.1:4002 | jq # For Bob
```

You should see greeting messages from both



Opening a Hydra Head

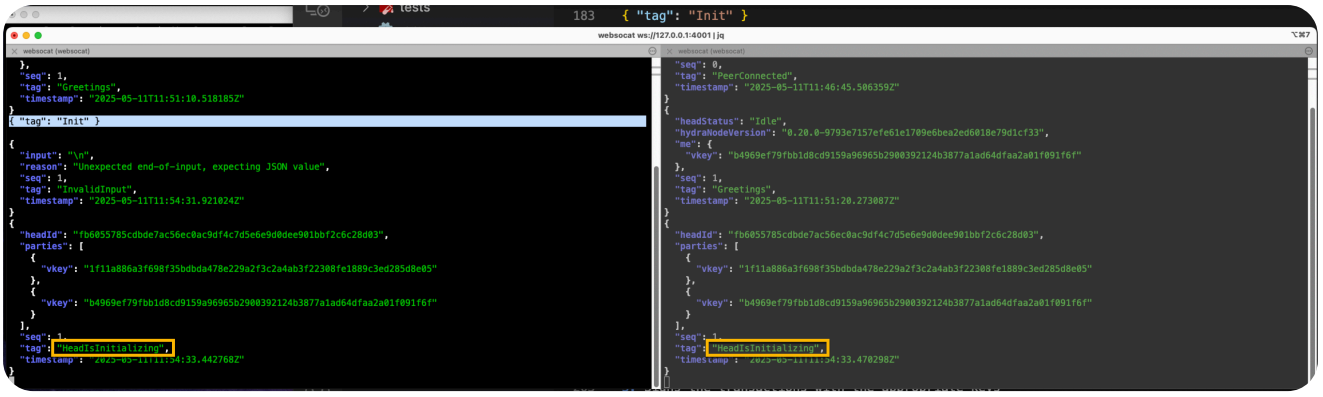
With all the nodes running, we initiate a Hydra head through the Websocket connection

In one of the two websocket sessions, we send the initiation message.

```
{ "tag": "Init" }
```

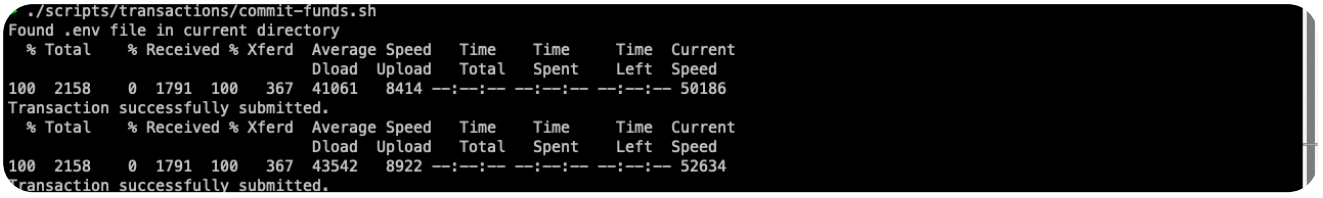
Note: after entering the message and pressing enter, if nothing happens press enter again. It should succeed in a few seconds. If it fails, just send the message again.

If working, you'll see a `HeadIsInitializing` message in the WebSocket output
When working, you'll see a HeadsIsInitializing message like below



Next, in a new terminal, commit the funds to the head

```
./scripts/transactions/commit-funds.sh
```



Layer 2 transactions

Open a new terminal, navigate to the project and make sure you are using nodejs v. 23+ and pnpm v 9+. If not install and use node 23 and pnpm 9.

Then, run

Then, run the install

```
pnpm install
```

```
pnpm install
Lockfile is up to date, resolution step is skipped
Packages: +96
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Progress: resolved 96, reused 96, downloaded 0, added 96, done

dependencies:
+ @emurgo/cardano-serialization-lib-nodejs 14.1.1
+ @helios-lang/codec-utils 0.3.4
+ @helios-lang/ledger 0.7.8
+ @helios-lang/tx-utils 0.6.7
+ @hyperionbt/helios 0.16.7
+ blessed 0.1.81
+ dotenv 16.5.0
+ ink 4.4.1
+ keypress 0.2.1
+ node-pty 1.0.0
+ prompt-sync 4.2.0
+ react 18.3.1
+ websocket 1.0.35
+ ws 8.18.2

devDependencies:
+ @types/blessed 0.1.25
+ @types/node 22.15.17
+ @types/prompt-sync 4.2.3
+ @types/react 18.3.21
+ @types/ws 8.18.1
+ tsx 4.19.4
+ typescript 5.8.3

Done in 1s using pnpm v9.15.9
```

Now, open two terminals: one for the Terminal User Interface (TUI)m and the other for the transaction stream viewer:

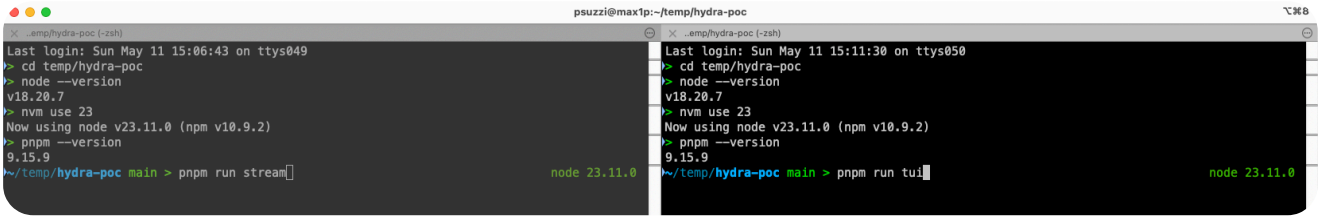
In the TUI terminal

```
pnpm run tui
```

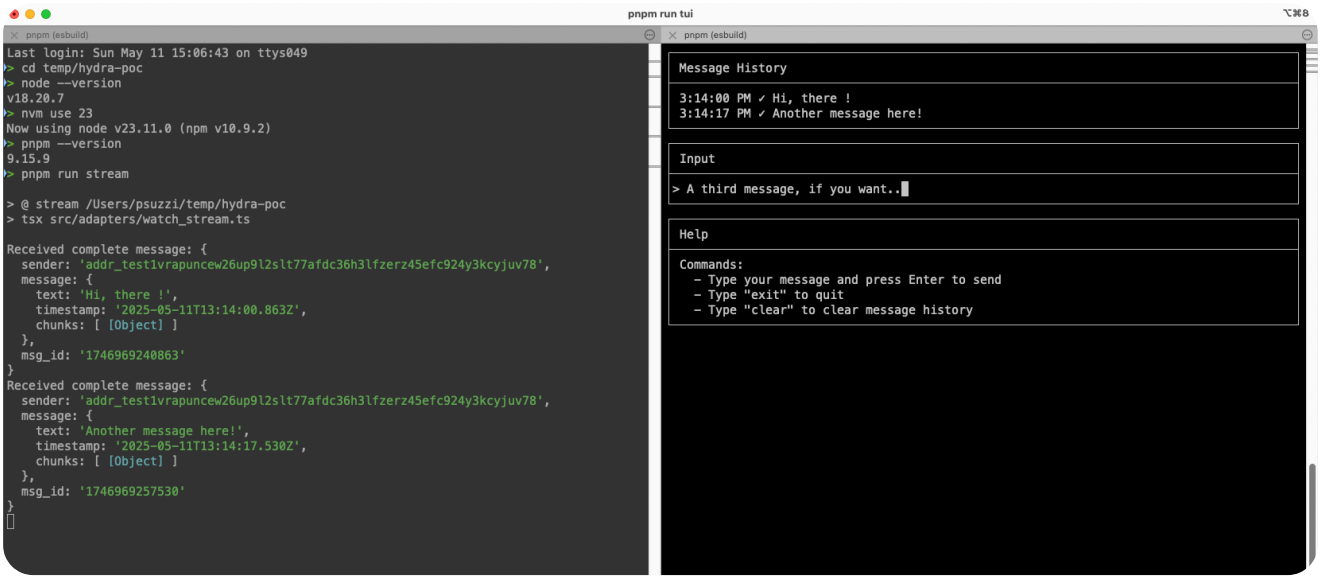
In the Stream terminal

```
pnpm run stream
```

Note: make sure you check the node version in each terminal before launching the commands.



TUI is a simplified interface to send and view sent messages, while the Stream Viewer provides real time monitoring in the hydra head. So, for instance, after sending a couple of messages through TUI, we'll see a situation like the one in the image



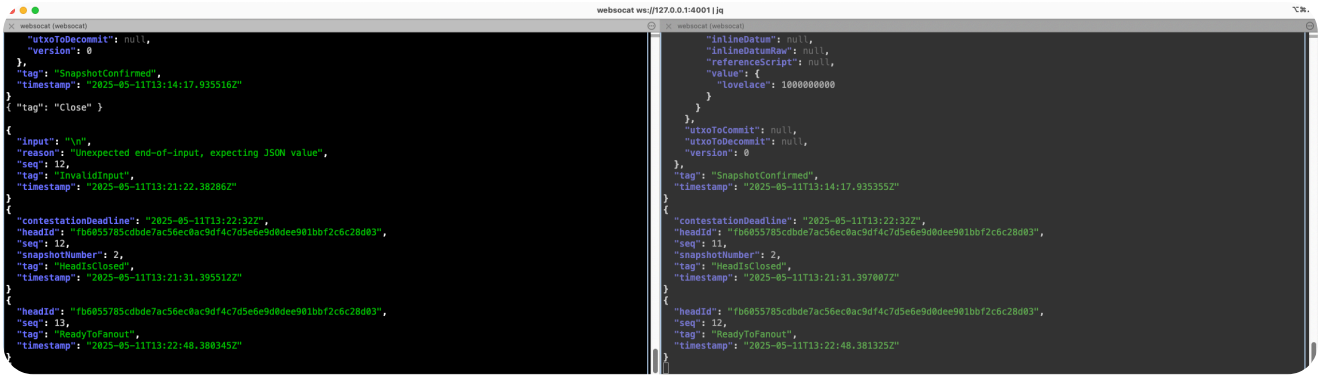
Of course, when sending messages through TUI, you can also check the status on the websockets and hydra nodes.

Closing Hydra Head

When done transacting on Hydra, and settle back to layer 1, send this message through the websocket connecttion:

```
{ "tag": "Close" }
```

If all goes as expected, you shall see a `HeadIsClosed` response message.
Then, if you wait patiently the contestation period expiration(e.g. 30-60 sec), you shall see a message `ReadyToFanout`



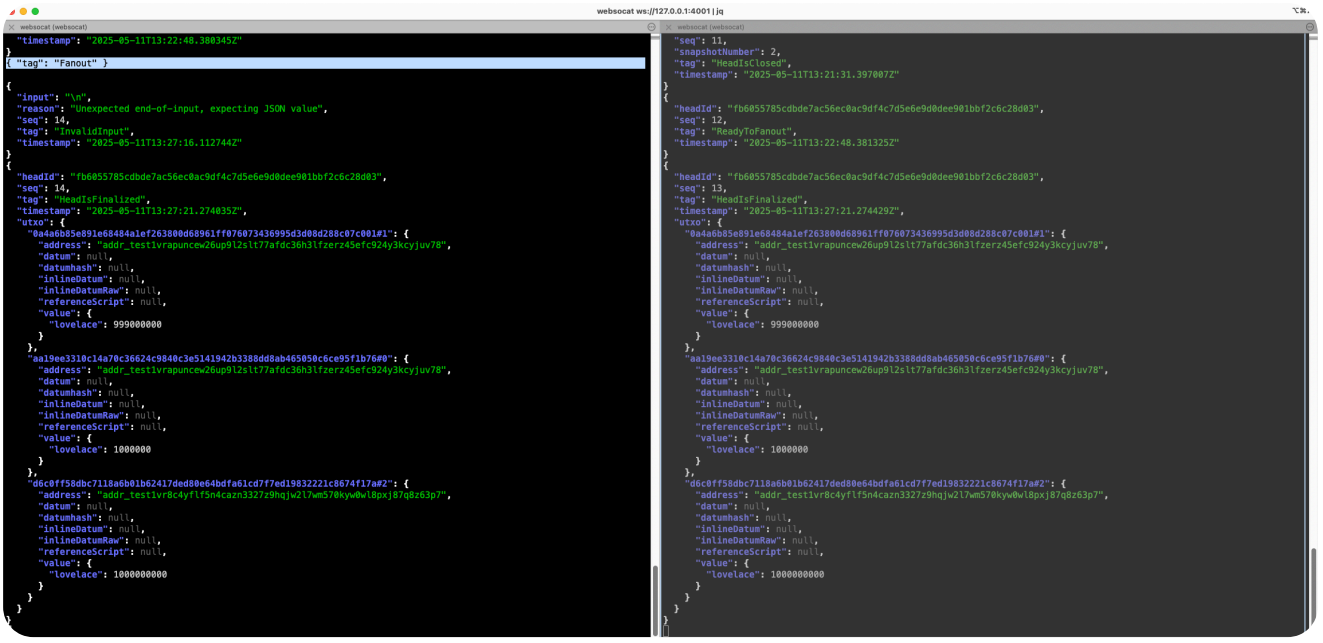
At this point you can finalize the head closure

```
{ "tag": "Fanout" }
```

With this:

- the fanout transaction is submitted to Layer1
- funds are distributed according to the final head state.

- The head transitions to `HeadIsFinalized` state



Open a new terminal and verify the final balances on Layer1 with this script:

```
./scripts/utils/query-node-funds.sh
```

The result shall look like the image

Now that the testing sequence is complete, return any unused test ada to the funding wallet to be used again later:

```
./scripts/utils/refund-funding-wallet.sh
```

And this concludes your initial walkthrough running a hydra head