

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.

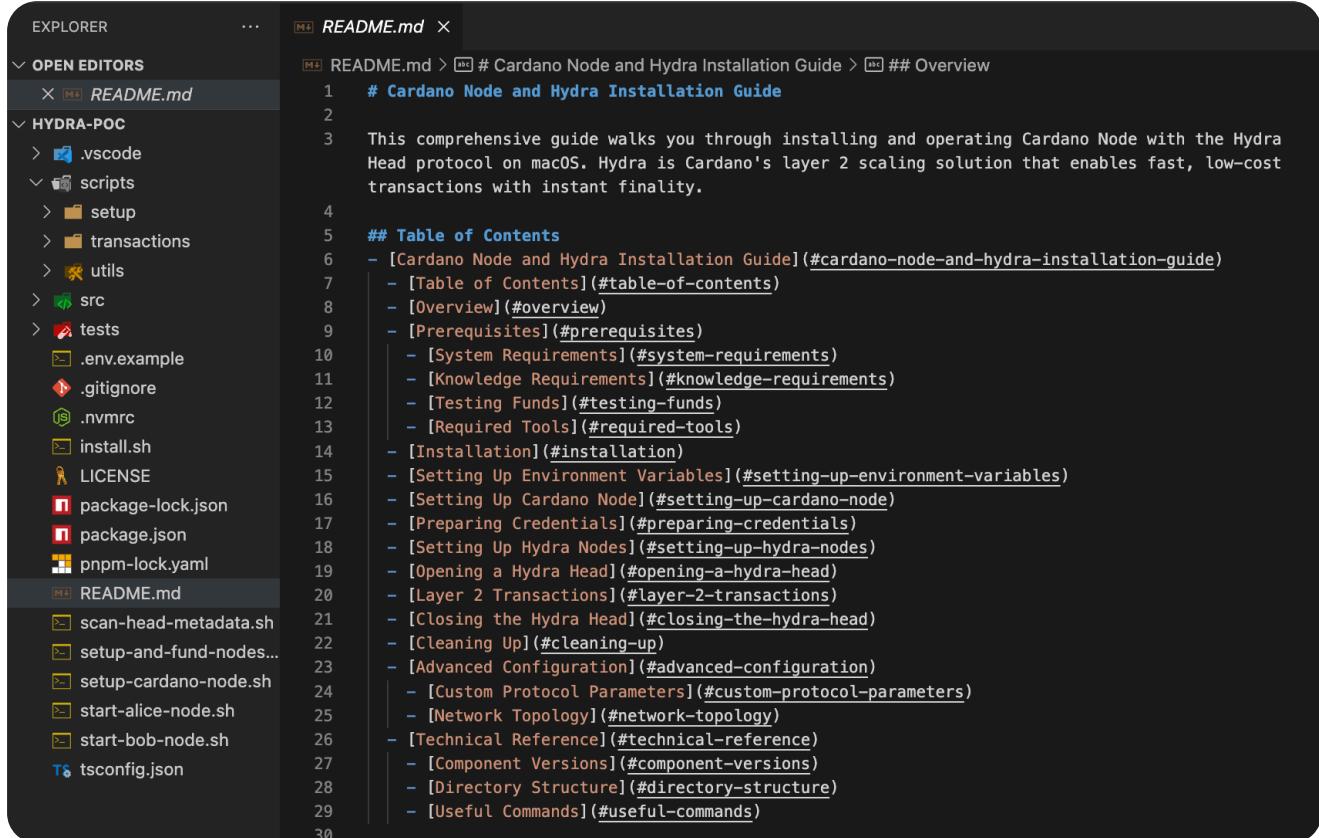
The contents of this article are coming from <https://github.com/Cardano-After-Dark/hydra-poc> > main branch (commit b7f813c).

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



The screenshot shows a code editor interface with two panes. The left pane, titled 'EXPLORER', displays a file tree for the 'HYDRA-POC' project. The right pane, titled 'README.md', shows the content of the README.md file. The file starts with a header and a table of contents, followed by detailed sections on system requirements, knowledge requirements, testing funds, required tools, installation, environment variables, cardano node setup, preparing credentials, hydra nodes setup, opening a hydra head, layer 2 transactions, closing the hydra head, cleaning up, advanced configuration, custom protocol parameters, network topology, technical reference, component versions, directory structure, and useful commands.

```
EXPLORER      ...      README.md ×  
OPEN EDITORS  README.md  
HYDRA-POC  
.vscode  
scripts  
setup  
transactions  
utils  
src  
tests  
.env.example  
.gitignore  
.nvmrc  
install.sh  
LICENSE  
package-lock.json  
package.json  
pnpm-lock.yaml  
README.md  
scan-head-metadata.sh  
setup-and-fund-nodes...  
setup-cardano-node.sh  
start-alice-node.sh  
start-bob-node.sh  
tsconfig.json  
  
README.md > # Cardano Node and Hydra Installation Guide > ## Overview  
# Cardano Node and Hydra Installation Guide  
This comprehensive guide walks you through installing and operating Cardano Node with the Hydra Head protocol on macOS. Hydra is Cardano's layer 2 scaling solution that enables fast, low-cost transactions with instant finality.  
## Table of Contents  
- [Cardano Node and Hydra Installation Guide](#cardano-node-and-hydra-installation-guide)  
- [Table of Contents](#table-of-contents)  
- [Overview](#overview)  
- [Prerequisites](#prerequisites)  
- [System Requirements](#system-requirements)  
- [Knowledge Requirements](#knowledge-requirements)  
- [Testing Funds](#testing-funds)  
- [Required Tools](#required-tools)  
- [Installation](#installation)  
- [Setting Up Environment Variables](#setting-up-environment-variables)  
- [Setting Up Cardano Node](#setting-up-cardano-node)  
- [Preparing Credentials](#preparing-credentials)  
- [Setting Up Hydra Nodes](#setting-up-hydra-nodes)  
- [Opening a Hydra Head](#opening-a-hydra-head)  
- [Layer 2 Transactions](#layer-2-transactions)  
- [Closing the Hydra Head](#closing-the-hydra-head)  
- [Cleaning Up](#cleaning-up)  
- [Advanced Configuration](#advanced-configuration)  
- [Custom Protocol Parameters](#custom-protocol-parameters)  
- [Network Topology](#network-topology)  
- [Technical Reference](#technical-reference)  
- [Component Versions](#component-versions)  
- [Directory Structure](#directory-structure)  
- [Useful Commands](#useful-commands)
```

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

```
./install.sh
% Total    % Received % Xferd  Average Speed   Time   Time   Time  Current
                                         Dload  Upload   Total Spent   Left Speed
0     0      0      0      0      0      0  --::--  --::--  --::--   0
100 140M 100 140M 0     0 45.1M 0 0:00:03 0:00:03 0:00:03 62.5M
Archive: hydra-aarch64-darwin-0.20.0.zip
  inflating: bin/hydra-tui
  inflating: bin/hydra-node
% Total    % Received % Xferd  Average Speed   Time   Time   Time  Current
                                         Dload  Upload   Total Spent   Left Speed
0     0      0      0      0      0      0  --::--  --::--  --::--   0
100 162M 100 162M 0     0 45.9M 0 0:00:03 0:00:03 0:00:03 59.1M
Fetching release information from https://api.github.com/repos/input-output-hk/mithril/releases/latest...
Downloading mithril-client to latest from https://github.com/input-output-hk/mithril/releases/download/2517.1/mithril-2517.1-macos-arm64.tar.gz...
Congrats! mithril-client has been upgraded to 0.12.1-b1a2fa from distribution latest and installed at bin!
Installation complete. Check that all components were downloaded successfully in the 'node' directory.
> ./scripts/utils/set-env-vars.sh
Environment variables set:
- Project root: /Users/psuzzi/temp/hydra-poc/
- Node socket: /Users/psuzzi/temp/hydra-poc/infra/node/preprod/node.socket
- Network ID: 1
  Hydra version: 0.20.0
```

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.

```
./setup-cardano-node.sh
Found .env file in current directory
First-time setup: Downloading blockchain snapshot...
1/5 - Checking local disk info...
2/5 - Fetching the certificate and verifying the certificate chain...
  Certificate chain validated
3/5 - Downloading and unpacking the cardano db
  [00:00:15] [########################################] 2.34 GiB/2.34 GiB (0.0s)
4/5 - Computing the cardano db message
5/5 - Verifying the cardano db signature...
Cardano db '8edf6ea565b8547459aeb4b030654e7b7c2d66be8d875e56a3b4fda8c896f597' has been unpacked and successfully checked against Mithril multi-signature contained in the certificate.

  Files in the directory 'db' can be used to run a Cardano node with version >= 10.3.1.

  If you are using Cardano Docker image, you can restore a Cardano Node with:

  docker run -v cardano-node-ipc:/ipc -v cardano-node-data:/data --mount type=bind,source="/Users/psuzzi/temp/hydra-poc/infra/node/preprod/db",target=/data/db/ -e NETWORK=preprod ghcr.io/intersectmbo/cardano-node:10.3.1

Starting Cardano node...
Node configuration: NodeConfiguration {ncSocketConfig = SocketConfig {ncNodeIPv4Addr = Last {getLast = Nothing}, ncNodeIPv6Addr = Last {getLast = Nothing}, ncNodePortNumber = Last {getLast = Just 0}, ncSocketPath = Last {getLast = Just "node.socket"}}, ncConfigFile = "config.json", ncTopologyFile = "topology.json", ncDatabaseFile = OnePathForAllDbs "db", ncProtocolFiles = ProtocolFilepaths {byronCertFile = Nothing, byronKeyFile = Nothing, shelleyKEKFile = Nothing}}
```

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

```
[max1@cardano node]# ./cardano-node --datadir /tmp/cardano-node-testnet --port 30000 &
[1] 11888 pts/0 00:00:00 bash
```

[max1@cardano node]# cardano-node status

```
[max1@cardano node]# cardano-node status
[2025-05-11 10:58:49.57 UTC] Started opening Volatile DB
[2025-05-11 10:58:49.57 UTC] Opened vol db with max slot number NoMaxSlotNo
[2025-05-11 10:58:49.57 UTC] Started opening Ledger DB
[2025-05-11 10:58:49.57 UTC] Replaying ledger from genesis
[2025-05-11 10:58:49.59 UTC] Replayed block: slot 0 out of 91216799. Progress: 0.0%
[2025-05-11 10:58:49.61 UTC] Replayed block: slot 19445 out of 91216799. Progress: 0.02%
[2025-05-11 10:58:49.61 UTC] Replayed block: slot 41042 out of 91216799. Progress: 0.04%
[2025-05-11 10:58:49.61 UTC] Replayed block: slot 62641 out of 91216799. Progress: 0.07%
[2025-05-11 10:58:49.62 UTC] Replayed block: slot 84242 out of 91216799. Progress: 0.09%
```

1

```
[maxip:cardano.node.ChainDB:Info:5] [2025-05-11 11:19:21.86 UTC] Replayed block: slot 91130396 out of 91216799. Progress: 99.91%
[maxip:cardano.node.ChainDB:Info:5] [2025-05-11 11:19:22.19 UTC] Replayed block: slot 91151988 out of 91216799. Progress: 99.93%
[maxip:cardano.node.ChainDB:Info:5] [2025-05-11 11:19:22.59 UTC] Replayed block: slot 91173581 out of 91216799. Progress: 99.95%
[maxip:cardano.node.ChainDB:Info:5] [2025-05-11 11:19:22.92 UTC] Replayed block: slot 91195190 out of 91216799. Progress: 99.98%
[maxip:cardano.node.ChainDB:Info:5] [2025-05-11 11:19:23.24 UTC] Opened lgr db
[maxip:cardano.node.ChainDB:Info:5] [2025-05-11 11:19:23.24 UTC] Started initial chain selection
[maxip:cardano.node.ChainDB:Info:5] [2025-05-11 11:19:23.24 UTC] Initial chain selected
[maxip:cardano.node.ChainDB:Info:5] [2025-05-11 11:19:23.24 UTC] Opened db with immutable tip at 8901e454081e4448bb4304a28de0911a983c355d08acc9d2bacb70
e8eac18f4a at slot 91216799 and tip 8901e454081e4448bb4304a28de0911a983c355d08acc9d2bacb701e0ca18f4a at slot 91216799
```

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:

```
./scripts/utils/query-tip.sh
Found .env file in current directory
{
  "block": 3466949,
  "epoch": 215,
  "era": "Conway",
  "hash": "2248ddd7fafda32de9aec1ff55f780bfae32603bb16b3af6be0ff920b97fdc8",
  "slot": 91279643,
  "slotInEpoch": 41243,
  "slotsToEpochEnd": 390757,
  "syncProgress": "100.00"
```

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

```
viewLearnLocalRootPeersPromotions = 0, viewActiveLocalRootPeers = 0, viewLearnLocalRootPeersPromotions = 0, viewKnownNonRootPeersPromotions = 0, viewEstablishedNonRootPeers = 0, viewNonRootPeersDemotions = 0, viewWarnNonRootPeersPromotions = 0, viewActiveNonRootPeers = 0, viewKnownBootstrapPeersPromotions = 0, viewEstablishedBootstrapPeers = 16, viewWarnBootstrapPeersDemotions = 0, viewWarnBootstrapPeersPromotions = 0, viewActiveBootstrapPeers = 0, viewKnownBootstrapPeersDemotions = 0  
maxipicardino_node_PeerConnectionManager:Info@0x [2025-05-11 17:27:48.11 UTC] TrcPromoteColdDiggerLeapers 10 (from List [184.164.124.192:6001])  
maxipicardino_node_ConnectionManager:Info@0x [2025-05-11 17:27:48.11 UTC] TrcConnectionHandler (fullDuplexConn = 0, duplexConn = 0, unidirectionalConn = 0, inboundConn = 0, outboundConn = 0)  
maxipicardino_node_ConnectionManager:Info@0x [2025-05-11 17:27:48.11 UTC] TrcConnectionHandler (fullDuplexConn = 0, duplexConn = 0, unidirectionalConn = 0, inboundConn = 1, outboundConn = 0)  
maxipicardino_node_LocalHandleChange:Info@0x [2025-05-11 17:27:48.10 UTC] TrcWithMuBeader (ConnectionId = LocalAddress "node.socket@1") Recv (ClientAgcy TokPropose,MsgProposeVersion) (from NodeClient_V_9,Tint_11,TList [Tint_11,Bool False])  
maxipicardino_node_ConnectionManager:Info@0x [2025-05-11 17:27:48.10 UTC] TrcWithMuBeader (ConnectionId = LocalAddress "node.socket@1") Send (ServerAgcy TokConfirm,MsgAcceptVersion NodeToClient_V_18 (TList [Tint_11,TBool False]))  
maxipicardino_node_LocalConnectionManager:Info@0x [2025-05-11 17:27:48.10 UTC] TrcConnectionHandler (ConnectionId = LocalAddress "node.socket@1") (TrHandshakeSuccess NodeToClient_V_18 (NodeToClientVersionData {networkMagic = NetworkMagic {unetworkMagic = 1}, query = False}))  
maxipicardino_node_LocalConnectionManager:Info@0x [2025-05-11 17:27:48.10 UTC] TrcConnectionManagerCounters (ConnectionManagerCounters {fullDuplexConn = 0, duplexConn = 0, unidirectionalConn = 1, inboundConn = 1, outboundConn = 0})  
maxipicardino_node_LocalConnectionManager:Info@0x [2025-05-11 17:27:48.10 UTC] TrcConnectionManagerCounters (ConnectionManagerCounters {fullDuplexConn = 0, duplexConn = 0, unidirectionalConn = 1, inboundConn = 0, outboundConn = 0})  
maxipicardino_node_LocalConnectionManager:Info@0x [2025-05-11 17:27:48.10 UTC] TrcConnectionManagerCounters (ConnectionManagerCounters {fullDuplexConn = 0, duplexConn = 0, unidirectionalConn = 1, inboundConn = 1, outboundConn = 0})  
maxipicardino_node_LocalInboundGovernor:Info@0x [2025-05-11 17:27:48.11 UTC] TrPromotedToWarmRemote (ConnectionId = LocalAddress "node.socket@1") (OperationSuccess (InboundIdleSt Unidirectional))  
maxipicardino_node_LocalInboundGovernor:Info@0x [2025-05-11 17:27:48.11 UTC] TrInboundGovernorCounters (InboundGovernorCounters {coldPeerRemote = 0, idlePeerRemote = 0, warnPeerRemote = 0, hotPeerRemote = 0})  
maxipicardino_node_LocalInboundGovernor:Info@0x [2025-05-11 17:27:48.11 UTC] TrInboundGovernorCounters (InboundGovernorCounters {coldPeerRemote = 0, idlePeerRemote = 0, warnPeerRemote = 0, hotPeerRemote = 1})  
maxipicardino_node_LocalInboundGovernor:Info@0x [2025-05-11 17:27:48.11 UTC] TrInboundGovernorCounters (InboundGovernorCounters {coldPeerRemote = 0, idlePeerRemote = 1, warnPeerRemote = 0, hotPeerRemote = 0})  
maxipicardino_node_LocalInboundGovernor:Info@0x [2025-05-11 17:27:48.11 UTC] TrInboundGovernorCounters (InboundGovernorCounters {coldPeerRemote = 0, idlePeerRemote = 1, warnPeerRemote = 0, hotPeerRemote = 0})  
maxipicardino_node_LocalInboundGovernor:Info@0x [2025-05-11 17:27:48.11 UTC] TrInboundGovernorCounters (InboundGovernorCounters {coldPeerRemote = 0, idlePeerRemote = 0, warnPeerRemote = 0, hotPeerRemote = 0})  
maxipicardino_node_LocalInboundGovernor:Info@0x [2025-05-11 17:27:48.11 UTC] TrInboundGovernorCounters (InboundGovernorCounters {coldPeerRemote = 0, idlePeerRemote = 0, warnPeerRemote = 0, hotPeerRemote = 0})  
maxipicardino_node_LocalConnectionManager:Info@0x [2025-05-11 17:27:48.11 UTC] TrcDottedWarmRemote (ConnectionId = LocalAddress "node.socket@1")
```

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.

The screenshot shows the 'Testnets faucet' section of the Cardano Docs website. On the left sidebar, under 'Tools', 'Testnets faucet' is selected. The main form has 'Preprod Testnet' selected in the 'Environment' dropdown and 'Receive test ADA' selected in the 'Action' dropdown. The 'Address (required)' field contains the testnet address: `addr_test1vrpz4ed5pcnynfv2y32kj7hq4k0g38nvhvkwpkm0c3ycws3w324l`. A red arrow points from the text 'Preprod Testnet' in the 'On this page' sidebar to the 'Preprod Testnet' button in the dropdown. Another red arrow points from the text 'Preprod Testnet' in the 'On this page' sidebar to the 'Address (required)' input field.

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

```
./setup-and-fund-nodes.sh
Found .env file in current directory
Setting up funding credentials...

Request funding from the Cardano Testnet Faucet:
addr_test1vrpz4ed5pcnynfv2y32kj7hq4k0g38nvhvkwpkm0c3ycws3w324l
Waiting for funds to be available in funding wallet...
Please fund the wallet using the Cardano Testnet Faucet
Address: addr_test1vrpz4ed5pcnynfv2y32kj7hq4k0g38nvhvkwpkm0c3ycws3w324l
Checking funding wallet balance...
Current balance: 0 lovelace
Insufficient funds. Checking again in 10 seconds... (Attempt 1/30)

Insufficient funds. Checking again in 10 seconds... (Attempt 2/30)
Insufficient funds. Checking again in 10 seconds... (Attempt 3/30)
Insufficient funds. Checking again in 10 seconds... (Attempt 4/30)
Insufficient funds. Checking again in 10 seconds... (Attempt 5/30)
Insufficient funds. Checking again in 10 seconds... (Attempt 6/30)
Insufficient funds. Checking again in 10 seconds... (Attempt 7/30)
Insufficient funds. Checking again in 10 seconds... (Attempt 8/30)
Insufficient funds. Checking again in 10 seconds... (Attempt 9/30)
Insufficient funds. Checking again in 10 seconds... (Attempt 10/30)
Insufficient funds. Checking again in 10 seconds... (Attempt 11/30)
Insufficient funds. Checking again in 10 seconds... (Attempt 12/30)
Insufficient funds. Checking again in 10 seconds... (Attempt 13/30)
Insufficient funds. Checking again in 10 seconds... (Attempt 14/30)
Insufficient funds. Checking again in 10 seconds... (Attempt 15/30)
Insufficient funds. Checking again in 10 seconds... (Attempt 16/30)
Insufficient funds. Checking again in 10 seconds... (Attempt 17/30)
Insufficient funds. Checking again in 10 seconds... (Attempt 18/30)
Insufficient funds. Checking again in 10 seconds... (Attempt 19/30)
Insufficient funds. Checking again in 10 seconds... (Attempt 20/30)

Checking funding wallet balance...
Current balance: 10000000000 lovelace
Sufficient funds detected in funding wallet!
Setting up node credentials...
Estimated transaction fee: 175137 Lovelace
Transaction successfully submitted.
Warning: alice-node wallet has no funds
Warning: alice-funds wallet has no funds
Warning: bob-node wallet has no funds
Warning: bob-funds wallet has no funds
# UTXO of alice-node
{
# UTXO of alice-funds
{
# UTXO of bob-node
{
# UTXO of bob-funds
}
Protocol parameters generated successfully!
Setup complete! Node wallets have been created and funded.
You can now start hydra-node for alice and bob.
```

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

```
./scripts/utils/query-node-funds.sh
Found .env file in current directory
# UTxO of alice-node
{
  "d6c0ff58dbc7118a6b01b62417ded80e64bdfa61cd7f7ed19832221c8674f17a#1": {
    "address": "addr_test1vp63853zzlqt8fwpp9kk7uxz02krrvvzhncryeefdfgglzgpuzqxw",
    "datum": null,
    "datumhash": null,
    "inlineDatum": null,
    "inlineDatumRaw": null,
    "referenceScript": null,
    "value": {
      "lovelace": 1000000000
    }
  }
}
# UTxO of alice-funds
{
  "d6c0ff58dbc7118a6b01b62417ded80e64bdfa61cd7f7ed19832221c8674f17a#0": {
    "address": "addr_test1vrapuncew26up9l2slt77afdc36h3lfzerz45efc924y3kcyjuv78",
    "datum": null,
    "datumhash": null,
    "inlineDatum": null,
    "inlineDatumRaw": null,
    "referenceScript": null,
    "value": {
      "lovelace": 1000000000
    }
  }
}
# UTxO of bob-node
{
  "d6c0ff58dbc7118a6b01b62417ded80e64bdfa61cd7f7ed19832221c8674f17a#3": {
    "address": "addr_test1vqu9uqq3g9h9ecjukkndtvaeayrr7antpuzphlv30ahzmuqhceesc",
    "datum": null,
    "datumhash": null,
    "inlineDatum": null,
    "inlineDatumRaw": null,
    "referenceScript": null,
    "value": {
      "lovelace": 1000000000
    }
  }
}
# UTxO of bob-funds
{
  "d6c0ff58dbc7118a6b01b62417ded80e64bdfa61cd7f7ed19832221c8674f17a#2": {
    "address": "addr_test1vr8c4yflf5n4cazn3327z9hjqw2l7wm570kyw0wl8pxj87q8z63p7",
    "datum": null,
    "datumhash": null,
    "inlineDatum": null,
    "inlineDatumRaw": null,
    "referenceScript": null,
    "value": {
      "lovelace": 1000000000
    }
  }
}
```

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

```
websocat (websocat)
Last login: Sun May 11 13:46:08 on ttys046
cd temp/hydra-poc
websocat ws://127.0.0.1:4001 | jq # For Alice
{
  "peer": "bob-node",
  "seq": 1,
  "tag": "PeerConnected",
  "timestamp": "2025-05-11T11:46:41.300575Z"
}
{
  "headStatus": "Idle",
  "hydradevVersion": "0.20.0-0793e7157fe61e1709e6bea2ed6018e79d1cf33",
  "me": {
    "key": "1f11a86a3f698f35bd8da478e229a2f3c2a4ab3f22308fe1889c3ed285d8e05"
  },
  "seq": 1,
  "tag": "Greetings",
  "timestamp": "2025-05-11T11:51:10.518185Z"
}
websocat (websocat)
Last login: Sun May 11 13:58:28 on ttys047
cd temp/hydra-poc
websocat ws://127.0.0.1:4002 | jq # For Bob
{
  "peer": "alice-node",
  "seq": 1,
  "tag": "PeerConnected",
  "timestamp": "2025-05-11T11:46:45.506359Z"
}
{
  "headStatus": "Idle",
  "hydradevVersion": "0.20.0-0793e7157fe61e1709e6bea2ed6018e79d1cf33",
  "me": {
    "key": "a9696ef79bb1d8cd9159a69695b2900392124b38771ad64dfa2a0f8091f6f"
  },
  "seq": 1,
  "tag": "Greetings",
  "timestamp": "2025-05-11T11:51:20.273087Z"
}
```

Opening a Hydra Head

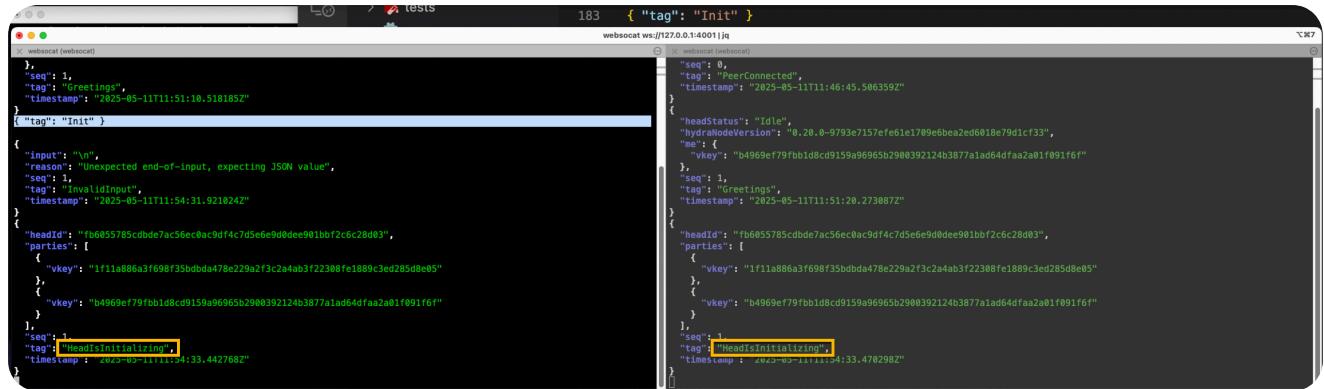
With all the nodes running, we initiate a Hydra head through the Websocket connection

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

```
{ "tag": "Tinit" }
```

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



```
ws://127.0.0.1:4001/q
{
  "tag": "Init"
}

{
  "seq": 1,
  "tag": "Greetings",
  "timestamp": "2025-05-11T11:51:10.518185Z"
}
{
  "tag": "Init"
}

{
  "input": "\n",
  "reason": "Unexpected end-of-input, expecting JSON value",
  "seq": 1,
  "tag": "InvalidInput",
  "timestamp": "2025-05-11T11:54:31.921024Z"
}
{
  "headId": "f8605785dbde7ac56ec0ac9df4c7d5e6e9d0dee901bbf2c6c28d83",
  "parties": [
    {
      "vkey": "If11a886a3f698f35bdbd478e229a2f3c2e4ab3f22388fe1889c3ed285d8e05"
    },
    {
      "vkey": "b4969ef79fb1d8cd9159a96965b2900392124b3877a1ad64dfa2a01f091f6f"
    }
  ],
  "seq": 1,
  "tag": "HeadIsInitializing",
  "timestamp": "2025-05-11T11:54:33.442768Z"
}

{
  "seq": 0,
  "tag": "PeerConnected",
  "timestamp": "2025-05-11T11:46:45.506359Z"
}
{
  "headStatus": "idle",
  "hydranodeVersion": "0.20.0-0793e7157ef61e1709e6bea2ed6818e791c133",
  "me": {
    "vkey": "b4969ef79fb1d8cd9159a96965b2900392124b3877a1ad64dfa2a01f091f6f"
  },
  "seq": 1,
  "tag": "Greetings",
  "timestamp": "2025-05-11T11:51:20.273087Z"
}
{
  "headId": "f8605785dbde7ac56ec0ac9df4c7d5e6e9d0dee901bbf2c6c28d83",
  "parties": [
    {
      "vkey": "If11a886a3f698f35bdbd478e229a2f3c2e4ab3f22388fe1889c3ed285d8e05"
    },
    {
      "vkey": "b4969ef79fb1d8cd9159a96965b2900392124b3877a1ad64dfa2a01f091f6f"
    }
  ],
  "seq": 1,
  "tag": "HeadIsInitializing",
  "timestamp": "2025-05-11T11:54:33.470298Z"
}
```

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

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

```
./scripts/transactions/commit-funds.sh
Found .env file in current directory
% Total    % Received % Xferd  Average Speed   Time   Time     Time Current
          Dload  Upload Total Spent   Left Speed
100  2158    0  1791  100    367  41061   8414 --:-- --:-- --:-- 50186
Transaction successfully submitted.
% Total    % Received % Xferd  Average Speed   Time   Time     Time Current
          Dload  Upload Total Spent   Left Speed
100  2158    0  1791  100    367  43542   8922 --:-- --:-- --:-- 52634
Transaction successfully submitted.
```

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

```
cd temp/hydra-poc
> node --version
v18.20.7
> nvm use 23
Now using node v23.11.0 (npm v10.9.2)
> pnpm --version
9.15.9
> node --version
v23.11.0
```

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:

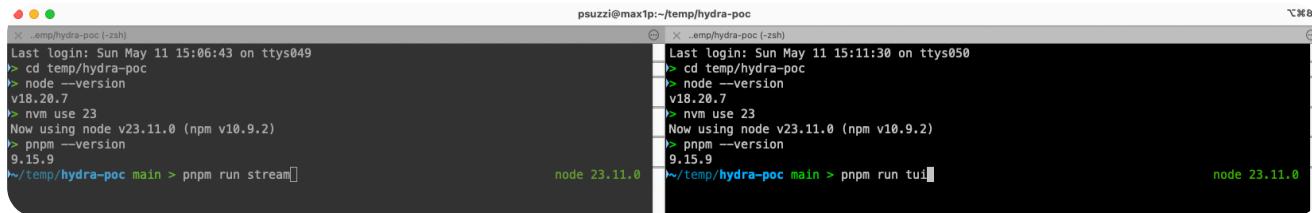
In the TUI terminal

```
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

The left terminal window shows the command sequence:

```

> cd temp/hydra-poc
> node --version
v18.20.7
> nvm use 23
Now using node v23.11.0 (npm v10.9.2)
> pnpm --version
9.15.9
> pnpm run stream

```

The right terminal window shows a message history application with the following logs:

```

Message History
3:14:00 PM ✓ Hi, there !
3:14:17 PM ✓ Another message here!

Input
> A third message, if you want..■

Help
Commands:
- Type your message and press Enter to send
- Type "exit" to quit
- Type "clear" to clear message history

```

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 connection:

```
{ "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`

The left tab shows the initial JSON message sent to close the head:

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

The right tab shows the response from the websocket, which includes the head's final state and a confirmation message:

```

{
  "utxoToCommit": null,
  "version": 0,
  "tag": "SnapshotConfirmed",
  "timestamp": "2025-05-11T13:14:17.935516Z"
}
{ "tag": "Close" }

{
  "input": "\n",
  "reason": "Unexpected end-of-input, expecting JSON value",
  "seq": 12,
  "tag": "InvalidInput",
  "timestamp": "2025-05-11T13:21:22.382862Z"
}
{
  "contestationDeadline": "2025-05-11T13:22:32Z",
  "headId": "fb6055785cdbe7ac56ec0ac9df4c7d5e69d0dee901bbf2c6c28d03",
  "seq": 12,
  "snapshotNumber": 2,
  "tag": "HeadIsClosed",
  "timestamp": "2025-05-11T13:21:31.395512Z"
}
{
  "headId": "fb6055785cdbe7ac56ec0ac9df4c7d5e69d0dee901bbf2c6c28d03",
  "seq": 13,
  "tag": "ReadyToFanout",
  "timestamp": "2025-05-11T13:22:48.388345Z"
}

```

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

```

{
  "timestamp": "2025-05-11T13:22:48.380345Z",
  "tag": "Fanout"
}

{
  "input": "\n",
  "reason": "Unexpected end-of-input, expecting JSON value",
  "seq": 14,
  "tag": "InvalidInput",
  "timestamp": "2025-05-11T13:27:16.112744Z"
}

{
  "headid": "fb605785cdbe7ac56ec0ac9df4c7d5e6e9d0dee901bbf2c6c28d03",
  "seq": 14,
  "tag": "HeadIsFinalized",
  "timestamp": "2025-05-11T13:27:21.2748352Z",
  "utxo": {
    "0xa4a6b85e81e68484a1ef263800d68961ff076073436995d3d8d288c07c001": {
      "address": "addr_test1vraruncew26up9l2slt77afdc36h3lfzerz45efc924y3kcyjuv78",
      "datum": null,
      "datumhash": null,
      "inlineDatum": null,
      "inlineDatumRaw": null,
      "referenceScript": null,
      "value": {
        "lovelace": 999000000
      }
    },
    "aa19ee3310c14a70c36624c9840c3e5141942b3388dd8ab465050c6ce95f1b76#0": {
      "address": "addr_test1vraruncew26up9l2slt77afdc36h3lfzerz45efc924y3kcyjuv78",
      "datum": null,
      "datumhash": null,
      "inlineDatum": null,
      "inlineDatumRaw": null,
      "referenceScript": null,
      "value": {
        "lovelace": 100000000
      }
    },
    "d6cff58dbc7118a6b01b62417ded88e64bdfa61cd7f7ed19832221c8674f17a#2": {
      "address": "addr_test1vraruncew26up9l2slt77afdc36h3lfzerz45efc924y3kcyjuv78",
      "datum": null,
      "datumhash": null,
      "inlineDatum": null,
      "inlineDatumRaw": null,
      "referenceScript": null,
      "value": {
        "lovelace": 100000000
      }
    }
  }
}

```

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

```
./scripts/utils/query-node-funds.sh
Found .env file in current directory
# UTx0 of alice-node
{
  "e2399c6ea657434eea597d54e7ef4f4dc38d68d6b8e937283ce5e079525cbdd5#3": {
    "address": "addr_test1vp63853zzlqt8fwpp9kk7uxz02krrwvzhncryeefdfgglzgpuzqxw",
    "datum": null,
    "datumhash": null,
    "inlineDatum": null,
    "inlineDatumRaw": null,
    "referenceScript": null,
    "value": {
      "lovelace": 992107572
    }
  }
}
# UTx0 of alice-funds
{
  "e2399c6ea657434eea597d54e7ef4f4dc38d68d6b8e937283ce5e079525cbdd5#0": {
    "address": "addr_test1vraruncew26up9l2slt77afdc36h3lfzerz45efc924y3kcyjuv78",
    "datum": null,
    "datumhash": null,
    "inlineDatum": null,
    "inlineDatumRaw": null,
    "referenceScript": null,
    "value": {
      "lovelace": 999000000
    }
  },
  "e2399c6ea657434eea597d54e7ef4f4dc38d68d6b8e937283ce5e079525cbdd5#1": {
    "address": "addr_test1vraruncew26up9l2slt77afdc36h3lfzerz45efc924y3kcyjuv78",
    "datum": null,
    "datumhash": null,
    "inlineDatum": null,
    "inlineDatumRaw": null,
    "referenceScript": null,
    "value": {
      "lovelace": 1000000
    }
  }
}
# UTx0 of bob-node
{
  "b6362fa4cb8d993ae3d9af567e2239e5c273be97af211af5a2aab7700307fc7#1": {
    "address": "addr_test1vqu9uqg3g9h9ecjukkndtvvaeayrr7antpuzphlv30ahzmuqhceesc",
    "datum": null,
    "datumhash": null,
    "inlineDatum": null,
    "inlineDatumRaw": null,
    "referenceScript": null,
    "value": {
      "lovelace": 996260979
    }
  }
}
# UTx0 of bob-funds
{
  "e2399c6ea657434eea597d54e7ef4f4dc38d68d6b8e937283ce5e079525cbdd5#2": {
    "address": "addr_test1vr8c4yflf5n4cazn3327z9hjqjw2l7wm570kyw0wl8pxj87q8z63p7",
    "datum": null,
    "datumhash": null,
    "inlineDatum": null,
    "inlineDatumRaw": null,
    "referenceScript": null,
    "value": {
      "lovelace": 1000000000
    }
  }
}
~/temp/hydra-poc main > █
```

5s node 23.11.0

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
```

```
./scripts/utils/refund-funding-wallet.sh
Found .env file in current directory
Estimated transaction fee: 185125 Lovelace
Transaction successfully submitted.
Estimated transaction fee: 179097 Lovelace
Transaction successfully submitted.
```

And this concludes your initial walkthrough running a hydra head