THE TRUST LAYER OF THE INTERNET

hedera.com

Abi Castro

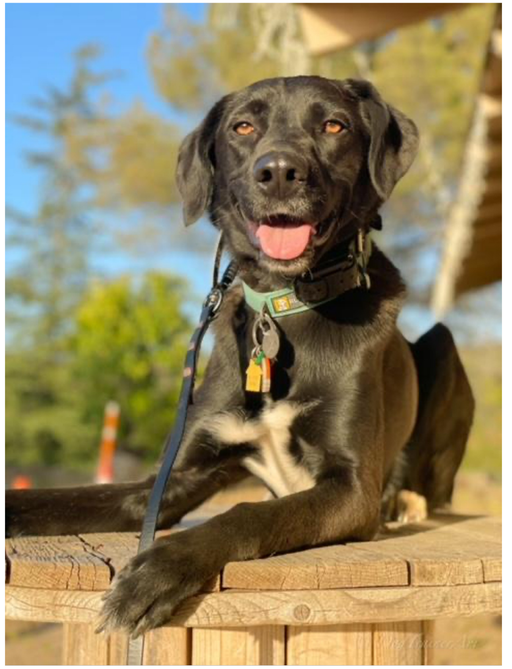
Developer Evangelist at Swirlds Labs

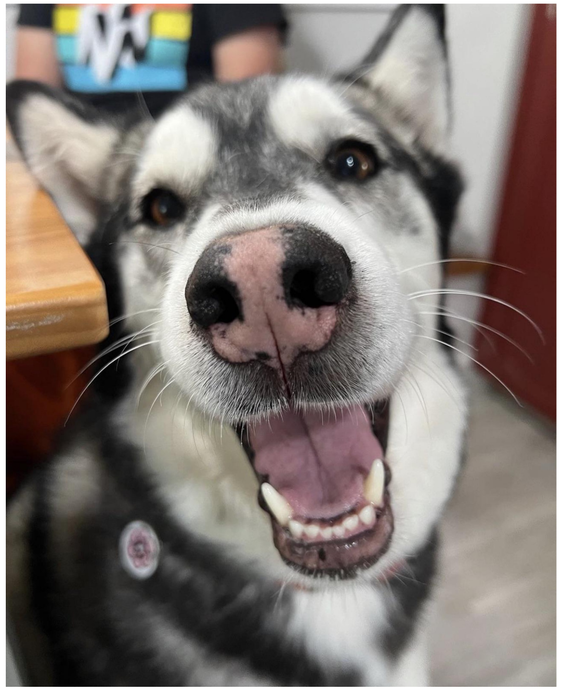
Connect with me!

@ridley___

/in/a-ridley

hedera.com





Francesco Coacci

Developer Evangelist at Swirlds Labs

Connect with me!

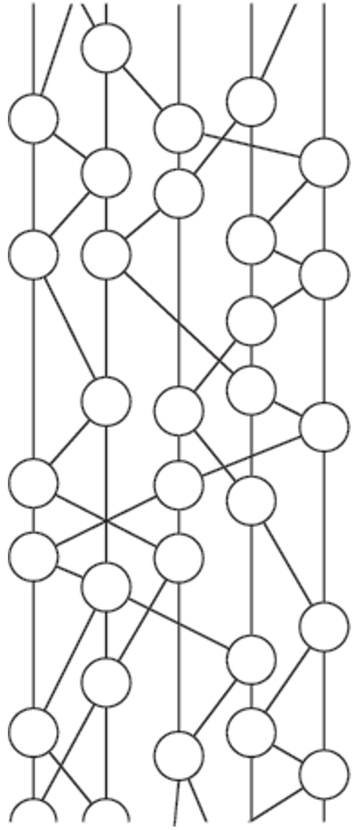
@francescocoacci

/in/francesco-coacci

“Passionate about Distributed Systems and Web3, I enjoy experimenting with all kinds of technologies to find new paths that lead to better and faster results. My love for sailing led me to an endless search for freedom, even in the tech world.”

In this workshop, we'll cover:

- ❑ An overview of the benefits of building on Hedera
- ❑ A tour of Hedera Networks
- ❑ An introduction to Hedera Development Core Concepts
- ❑ The 3 steps to building on the Hedera network
  - ○ Create a free development account
  - ○ Choose your SDK
  - ○ Set up your environment



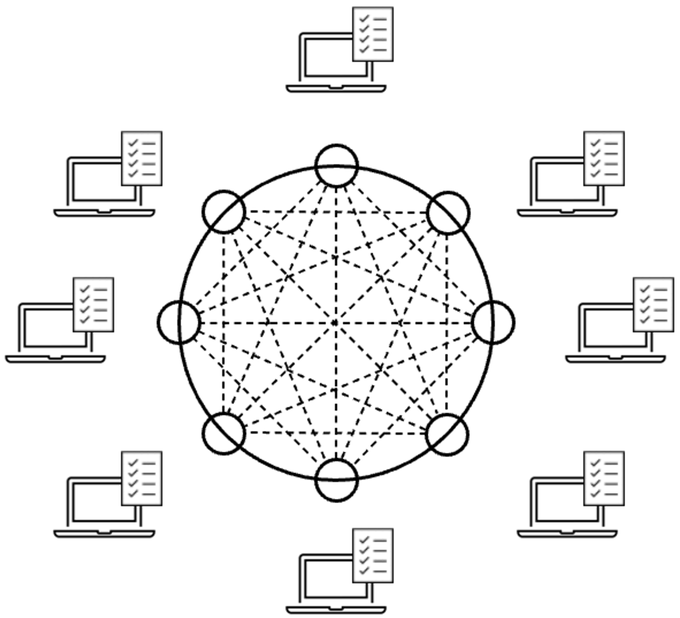
Hedera is a third-generation public distributed ledger (DLT)

	1ST GENERATION	2ND GENERATION	3RD GENERATION
	Bitcoin Blockchain	Ethereum Blockchain	Hedera™ Hashgraph
TRANSACTIONS PER SECOND	3+ TPS	12+ TPS	10,000+ TPS
AVERAGE TRANSACTION FEE	$22.57 USD	$19.55 USD	$0.0001 USD
AVERAGE TRANSACTION CONFIRMATION	10-60 MINUTES	10-20 SECONDS	3-5 SECONDS (W/ FINALITY)

• Cryptocurrency transactions. For Hedera, range shown for transactions not requiring a transaction record, but can receive a transaction receipt.
• Avg. Bitcoin tx fee from 6/26/20 - 9/24/20 from https://blockchair.com/bitcoin/charts/average-transaction-fee-usd?interval=3m
• Avg. Ethereum tx fee from 6/26/20 - 9/24/20 from https://blockchair.com/ethereum/charts/average-transaction-fee-usd?interval=3m

Distributed ledgers are a key component of Web 3.0 because of their qualities

DISTRIBUTED LEDGER



Entire network records and validates each transaction

CENTRALIZED LEDGER

Single authority verifies, records, and executes transactions

Distributed ledgers are a key component of Web 3.0 because of their qualities

No central point of failure to attack

Reliant on strong cryptography to prove data integrity and tamper resistance

Rules for the ledger are determined by a governing concept of some sort

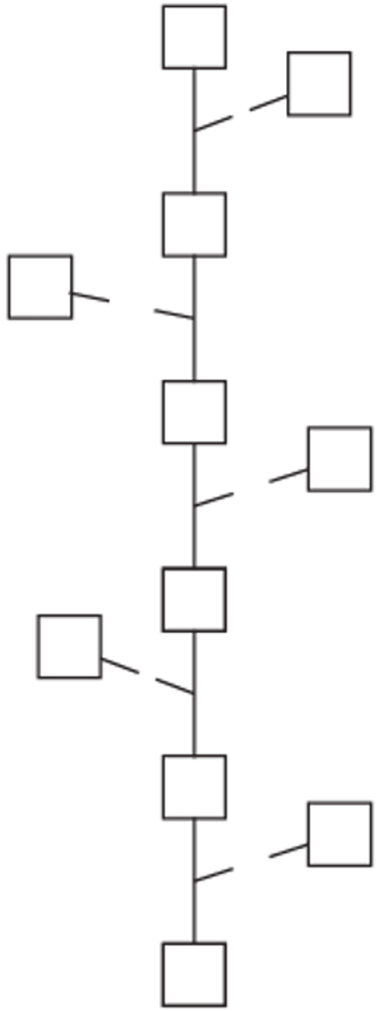
Require a consensus mechanism to determine the rules for adding new transactions to the state of the ledger

A comparison of 3rd Generation DLTs

	POLYGON	ALGORAND	(ETHEREUM)	3RD GENERATION
	polygon	Algorand		Hedera™ Hashgraph
TRANSACTIONS PER SECOND	6500* (CLAIMED)	1,000† (APPROX)	12+ TPS	10,000+ TPS
AVERAGE TRANSACTION FEE	$0.002 USD (VARIABLE)	0.001 ALGO (FIXED)	$19.55 USD (VARIABLE)	$0.0001^ USD (FIXED)
AVERAGE TRANSACTION CONFIRMATION	5 - 10 SECONDS (LEADER BLOCK CREATION)	5 SECONDS (LEADER BLOCK CREATION)	10-20 SECONDS (LEADER BLOCK CREATION)	3-5 SECONDS (W/ FINALITY)
ENERGY USE PER TRANSACTION	90+ KWH	0.00534 KWH	102+ KWH	0.00017 KWH

* Actual TPS closer to 53 over a rolling 30 day period - https://thecoinweekly.com/not-so-fast-are-eth-killers-as-speedy-as-they-claim
^ Hedera Transaction fees - https://hedera.com/fees
† https://www.algorand.com/resources/algorand-announcements/algorand-2021-performance

Blockchains



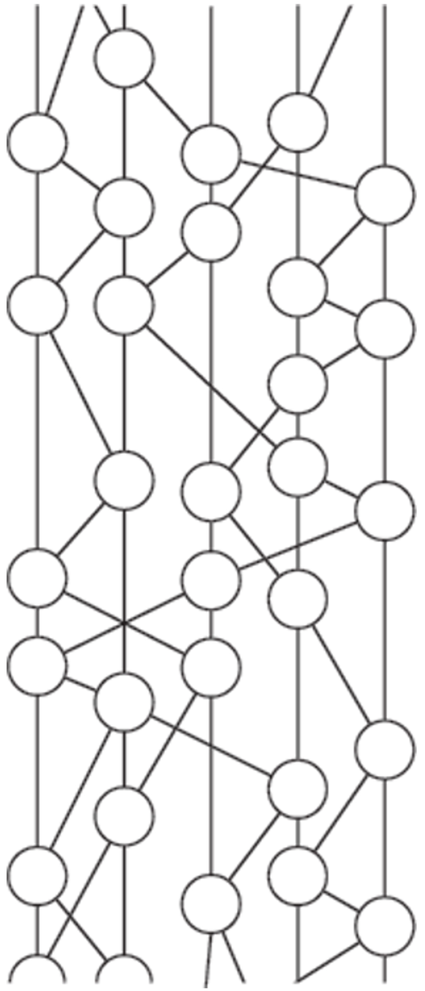
Proof-of-Work (PoW) typically has heavy electrical / computational requirements

No guaranteed consensus, just a “good enough” probability

Inefficient - “stale” blocks are pruned

Can be difficult to scale beyond ~1,000 tps

Hashgraph



Hashgraph is an **open-source** consensus algorithm and data structure

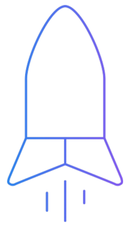
Uses a directed acyclic graph (DAG) and novel inventions:

- Gossip about Gossip protocol
- Virtual voting

Hedera currently supports 10,000+ cryptocurrency transactions per second

Finality within 3-5 seconds

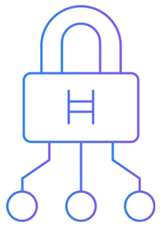
You will build fast, fair, secure, applications in a cost-effective and sustainable manner



Performance

10,000 TPS+

Finality in seconds

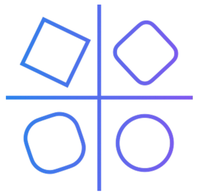


Security

Asynchronous Byzantine Fault Tolerance

Not leader-based

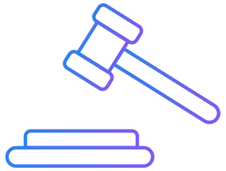
Proof of Stake



Stability

No forking of the network

Fees stable in USD



Governance

Run initial network nodes

Roadmap

Treasury management

Sustainability

Carbon negative infrastructure

Lowest carbon footprint of any public network

Networks

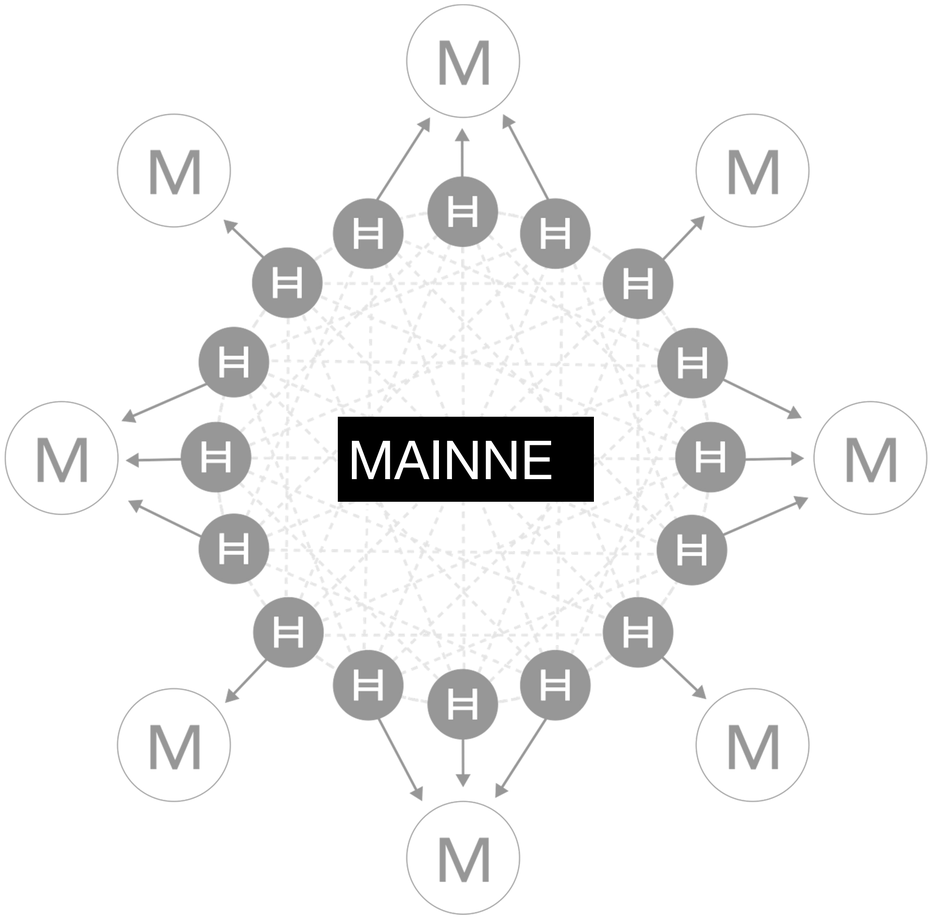
FEATURES AND SETUP

MAINNET

- Can submit HAPI (Hedera API) transactions to the Hedera network
- Contributes to consensus on transactions
- Creates events on the Hedera network
- Requires HBAR cryptocurrency payment for transactions & queries

MIRRORNET

- Maintains a history of some or all the Hedera network state and ledger of transactions
- Value-added services (managed read-only node, etc.)
- Enables analytical insight into an application's state / transactions
- Publish and subscribe capabilities



L LOCALNET

- Personal consensus and mirror node
- Avoid being interrupted by anyone throttling the network

T TESTNET

- Run same code as Hedera Mainnet nodes
- Suitable for pre-production environment
- Reset on a quarterly basis
- 10.000 HBARs refill every 24 hours

P PREVIEWNET

- Run code currently under development
- Not suitable pre-production environment

Setup a Local Network

```
$git clone https://github.com/hashgraph/hedera-local-node.git
```

```
$cd hedera-local-node
```

```
$docker-compose up -d
```

```
//Create your local client
const node = {"127.0.0.1:50211": new AccountId(3)};
const client = Client.forNetwork(node).setMirrorNetwork("127.0.0.1:5600");
```

```
client.setOperator(AccountId.fromString("0.0.2"),PrivateKey.fromString("302e020100300506032b65700422042091132178e72057a1d7528025956fe39b0b847f200ab59b2fdd367017f3087137"));
```

Connect to a Network

Step 1: Define **.env** file

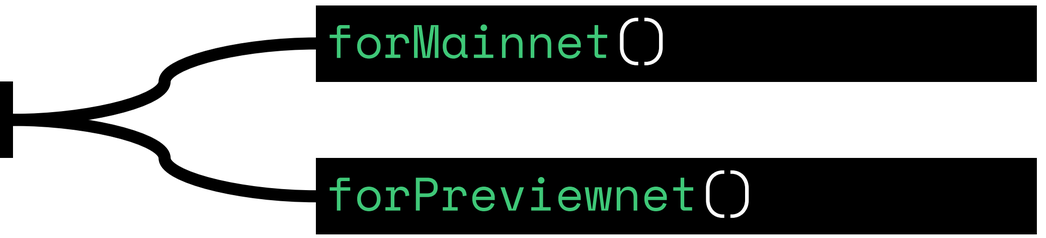
```
OPERATOR_ID = 0.0.351…
OPERATOR_KEY = 302e…
```

Step 2: Import credentials

```
const operatorId = AccountId.fromString(process.env.OPERATOR_ID);
const operatorKey = PrivateKey.fromString(process.env.OPERATOR_KEY);
```

Step 3: Configure client

```
const client = Client.forTestnet()
```




```
client.setOperator(operatorId, operatorKey);
```

Accounts

DEFINITION AND STRUCTURE

Accounts

- Two ways to create one: using an existing account or by using an alias
- Accounts are composed of:
  - Account ID (eg 0.0.3452)
  - Keys (single key or key list)
  - Token balance(s) (HBAR and any other token created using HTS)
  - Automatic associations
  - Account Alias (an immutable key once set)
  - Livehash(es)
- Accounts like files and smart contracts have an expiration (not enabled yet)

Create an Account using JS SDK

```
const { AccountCreateTransaction, Hbar, PrivateKey } = require("@hashgraph/sdk");

const newKey = PrivateKey.generateED25519();
const response = await new AccountCreateTransaction()
    .setInitialBalance(new Hbar(10))
    .setKey(newKey.publicKey)
    .execute(client);

const receipt = await response.getReceipt(client);
```

Transactions and Queries

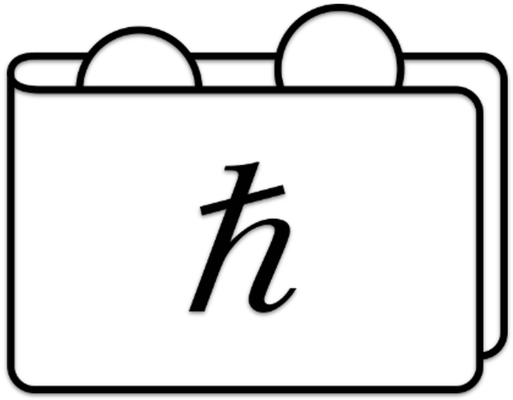
FEATURES AND STRUCTURE

Transactions

- Requests sent from client to consensus node
- Each transaction has an associated fee
- Users can set a max transaction fee
- Transactions are composed of:
  - Node Account (consensus node the tx is being sent to)
  - Transaction ID (eg 0.0.9401@1598924675.82525000)
  - Transaction Fee
  - Valid Duration
  - Memo (a string of text up to 100 bytes)
  - Transaction (type of transaction)
  - Signatures

Scheduled Transactions:

- Queue a transaction to wait signatures
- Ideal for multi-sign transactions



Topic Create Transaction

```
const { Client, TopicCreateTransaction } = require("@hashgraph/sdk");

const main = async () => {
    const client = Client.forTestnet().setOperator(yourAccountId,yourPrivateKey);

    const tx = await new TopicCreateTransaction().execute(client);
    const receipt = await tx.getReceipt(client);
    const newTopicId = receipt.topicId;
}

main();
```

Queries

- Processed by a single node
- Useful to get some aspects of the current consensus state (e.g. Account Balance)
- Some are free but generally subject to fees
- No refund if query is overpaid
- As an alternative there’s the Mirror Node API (no fees)

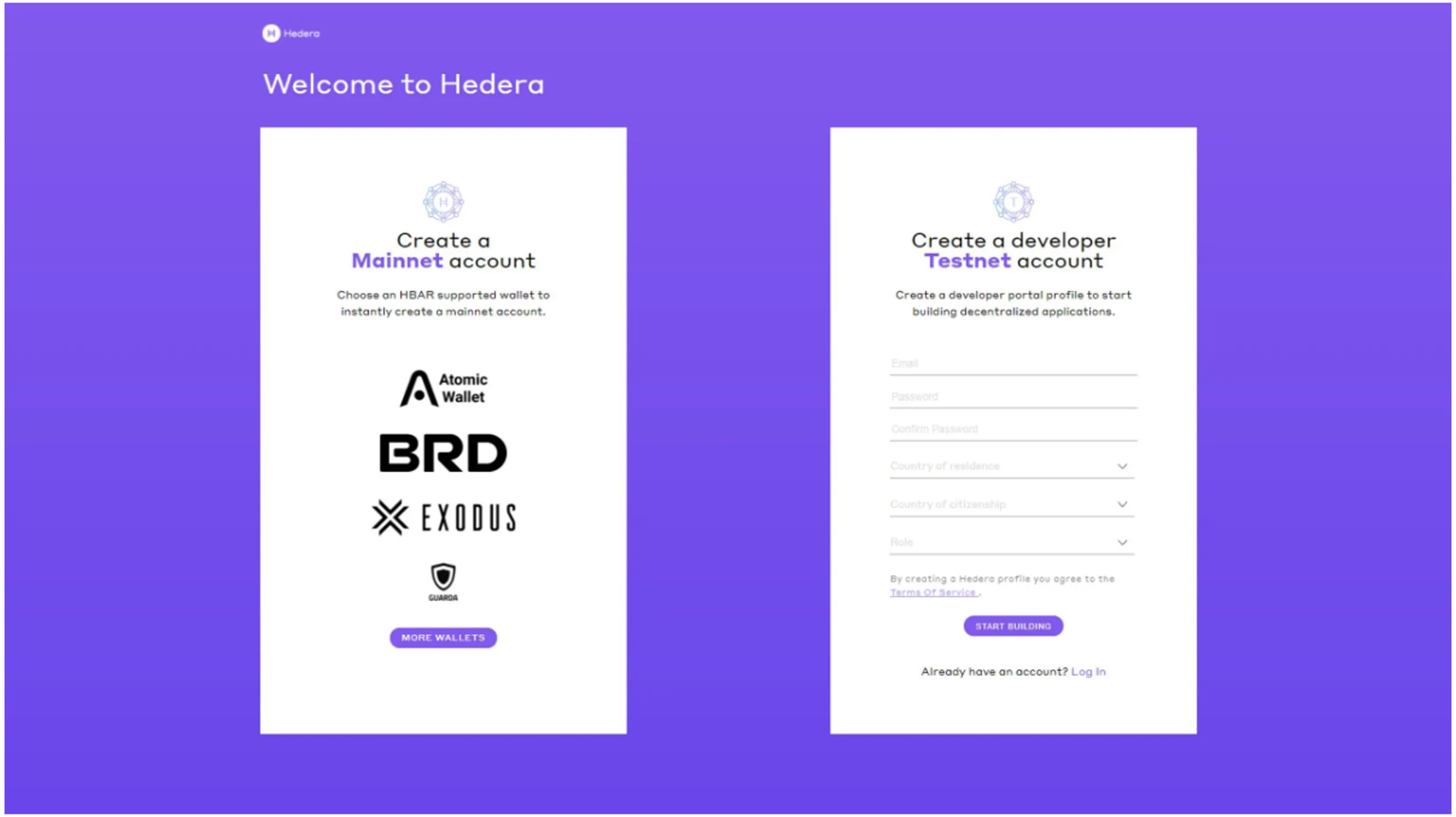
Account Balance Query

```
const { Client, AccountBalanceQuery } = require("@hashgraph/sdk");

const main = async () => {
    const client = Client.forTestnet().setOperator(yourAccountId,yourPrivateKey);

    const tx = new AccountBalanceQuery()
         .setAccountId(yourAccountId);
    const accountBalance = await tx.execute(client);
    const hbarBalance = accountBalance.hbars;
}

main();
```

Let’s Build!

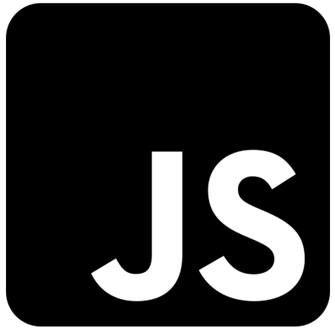
START CREATING ON THE HEDERA NETWORK

Step 1: Create a free development account

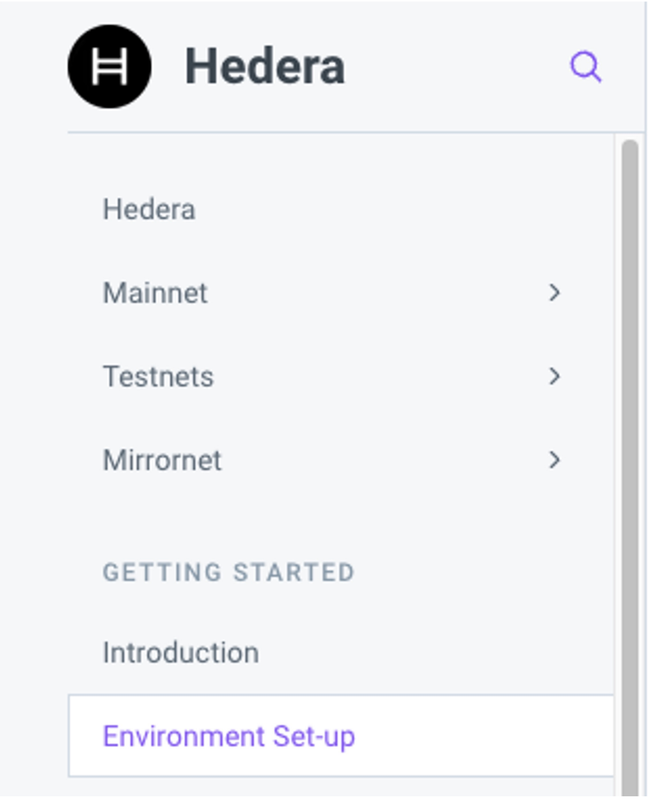


https://portal.hedera.com/register

Step 2: Choose your SDK!



Step 3: Set up your environment



Environment Set-up

Summary

In this section you will complete the following:

- Create your project directory
- Create a `.env` file and store your Hedera testnet account ID and keys
- Set-up your Hedera testnet client

https://docs.hedera.com/guides/getting-started/environment-set-up

In this workshop, we have:

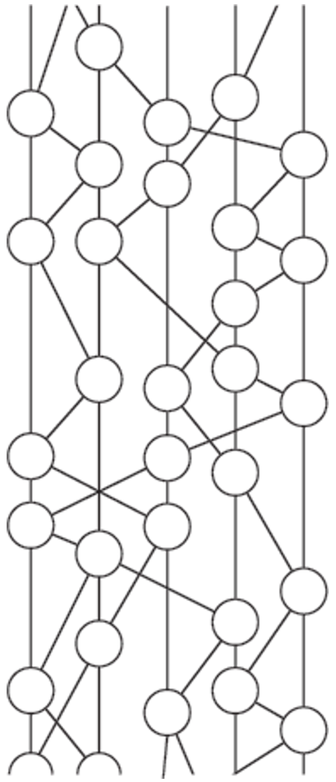
An overview of the benefits of building on Hedera

A tour of Hedera Networks

An introduction to Hedera Development Core Concepts

The 3 steps to building on the Hedera network

- Create a free development account
- Choose your SDK
- Set up your environment



Questions / Discussion

We’d love to help you further at:

DOCUMENTATION

Get Started: https://hedera.com/get-started

Documentation: https://docs.hedera.com/guides/

CODING TUTORIALS

Articles: https://docs.hedera.com/guides/resources/tutorials

Videos: https://www.youtube.com/playlist?list=PLcaTa5RR9SuA__8rzCKru8Y_F6iMJPEUD

COMMUNITY

Developer Discord: https://hedera.com/discord