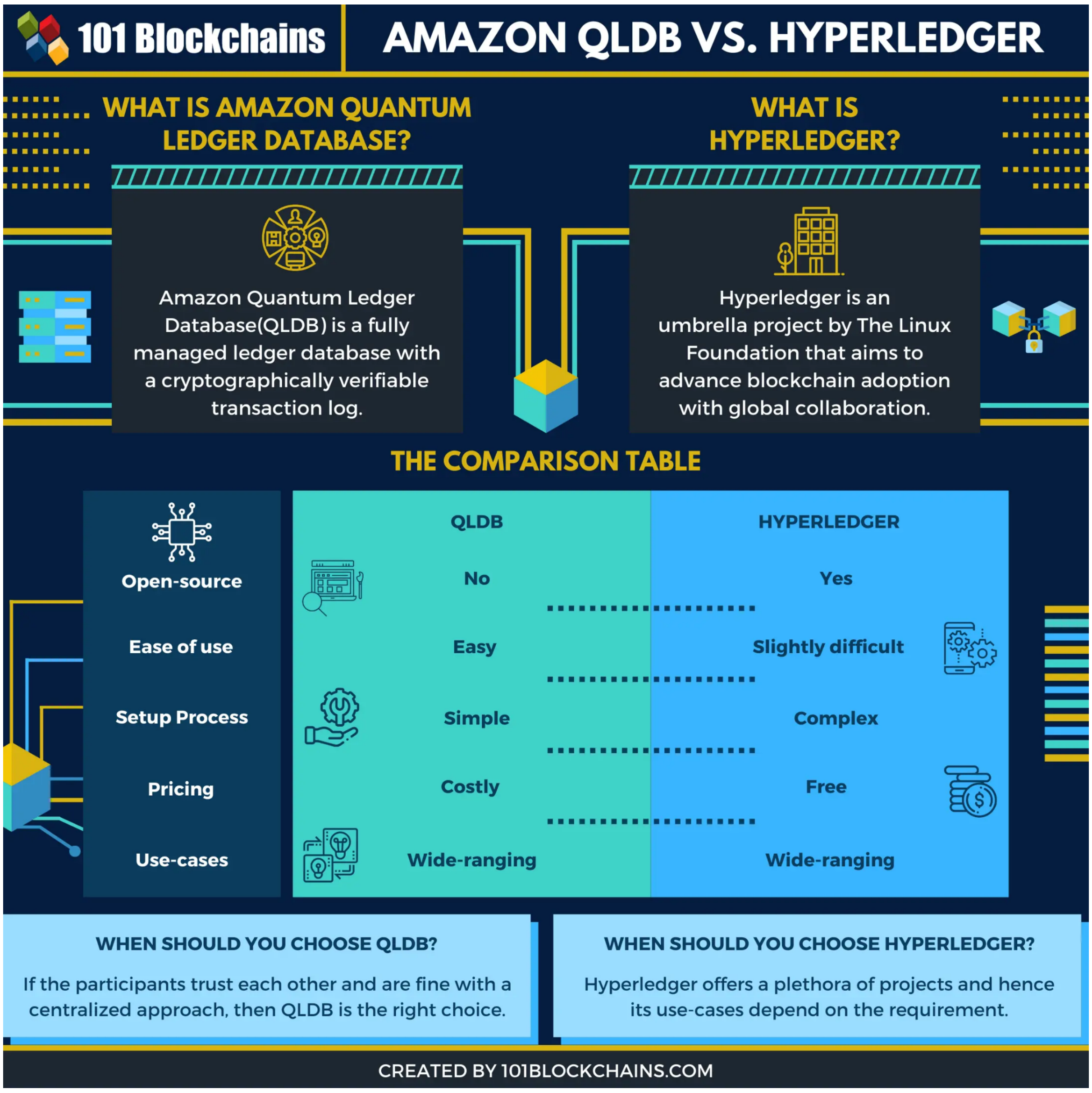
CloudSeals

Amazon Quantum Ledger DataBase (QLDB) vs Hyperledger:



101 Blockchains

AMAZON QLDB VS. HYPERLEDGER

WHAT IS AMAZON QUANTUM LEDGER DATABASE?

Amazon Quantum Ledger Database(QLDB) is a fully managed ledger database with a cryptographically verifiable transaction log.

WHAT IS HYPERLEDGER?

Hyperledger is an umbrella project by The Linux Foundation that aims to advance blockchain adoption with global collaboration.

THE COMPARISON TABLE

	QLDB	HYPERLEDGER
Open-source	No	Yes
Ease of use	Easy	Slightly difficult
Setup Process	Simple	Complex
Pricing	Costly	Free
Use-cases	Wide-ranging	Wide-ranging

WHEN SHOULD YOU CHOOSE QLDB?

If the participants trust each other and are fine with a centralized approach, then QLDB is the right choice.

WHEN SHOULD YOU CHOOSE HYPERLEDGER?

Hyperledger offers a plethora of projects and hence its use-cases depend on the requirement.

CREATED BY 101BLOCKCHAINS.COM

QLDB Pricing:

	Price
Write I/Os	$0.70 per 1 million requests
Read I/Os	$0.136 per 1 million requests
Journal Storage Rate	$0.03 per GB-month
Indexed Storage Rate	$0.25 per GB-month

Price	
Data Transfer IN To Amazon QLDB From Internet	
All data transfer in	$0.00 per GB
Data Transfer OUT From Amazon QLDB To Internet	
AWS customers receive 100 GB of data transfer out to the internet free each month, aggregated across all AWS Services and Regions (except China and GovCloud). The 100 GB free tier data transfer out to the internet is global and does not apply separately or individually to AWS regions.	
First 10 TB / Month	$0.09 per GB
Next 40 TB / Month	$0.085 per GB
Next 100 TB / Month	$0.07 per GB
Greater than 150 TB / Month	$0.05 per GB

Pricing example #1

A financial services company has an application for their customers that provides information on bank branches, including hours of operation, branch-specific services (e.g., mortgage broker), and bulletins (e.g., holidays, special offers) for their entire network of offices. This application makes 0.15 write IO requests per second and 180 read IO requests per second. This application has 30 GB of data in its journal and 30 GB of indexed storage.

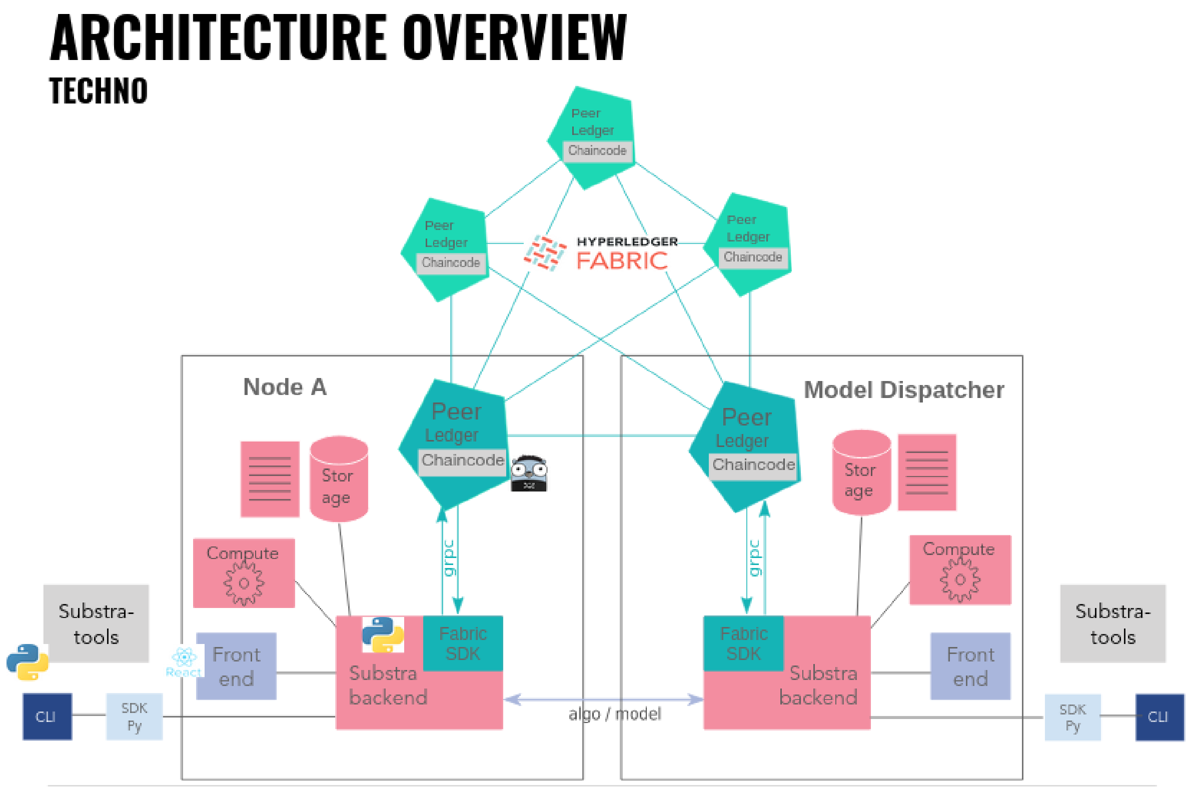
Dimension	Usage	Price	Monthly Total
Write I/O Requests	0.39 million	$0.70/million	$0.28
Read I/O Requests	473.4 million	$0.136/million	$64.38
Journal Storage	30 GB	$0.03/GB-month	$0.90
Indexed Storage	30 GB	$0.25/GB-month	$7.50
Data Transfer Out	0.5 GB	$0.00 (<100 GB)	$0.00
			$73.06

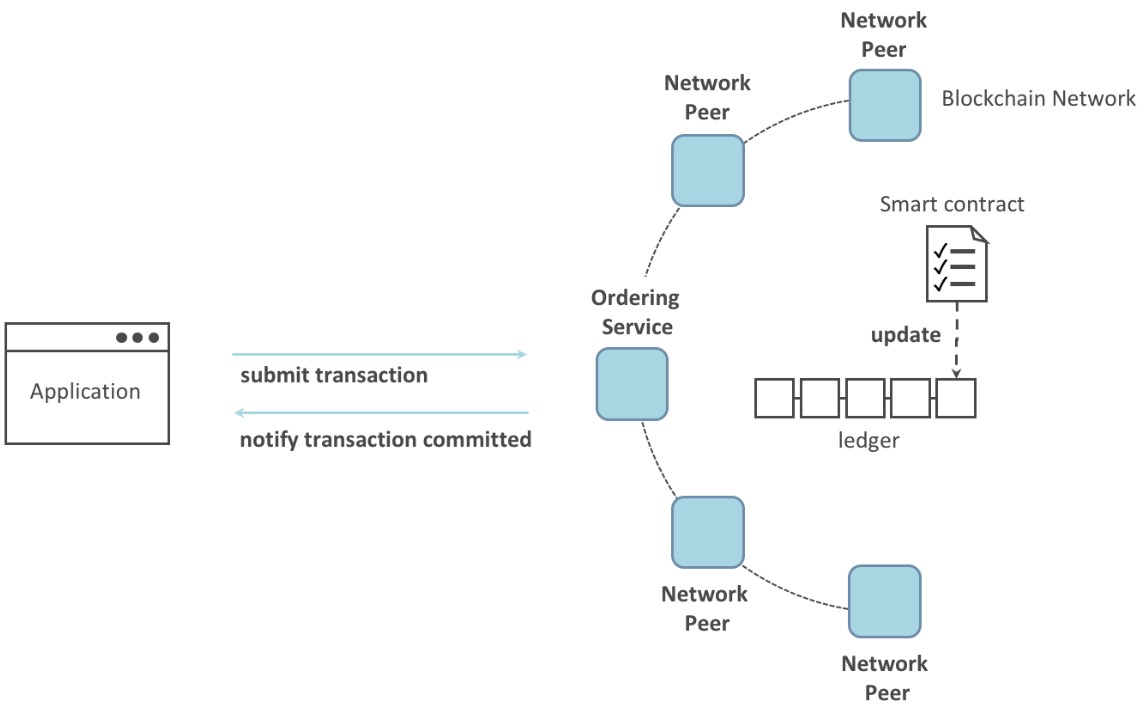
Pricing example #2

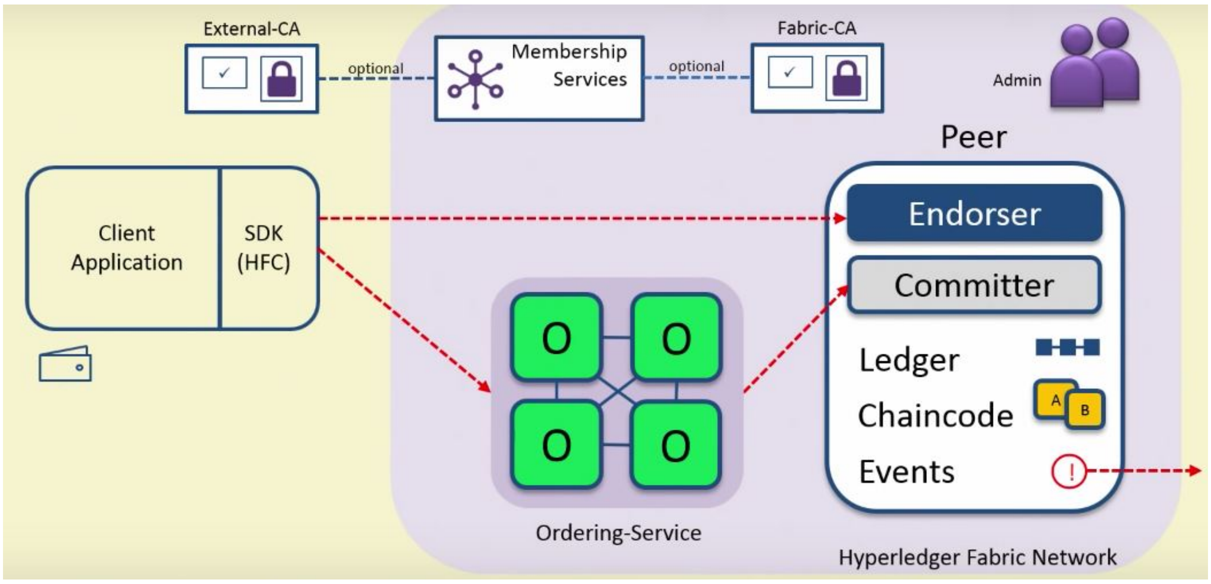
A supply chain logistics company develops a platform that provides their customers the full data lineage of items moving through the supply chain. This application makes 30 write IO requests per second and 150 read IO requests per second. The journal stores 2 TB of data and indexed storage keeps 600 GB of data between its current tables and history tables.

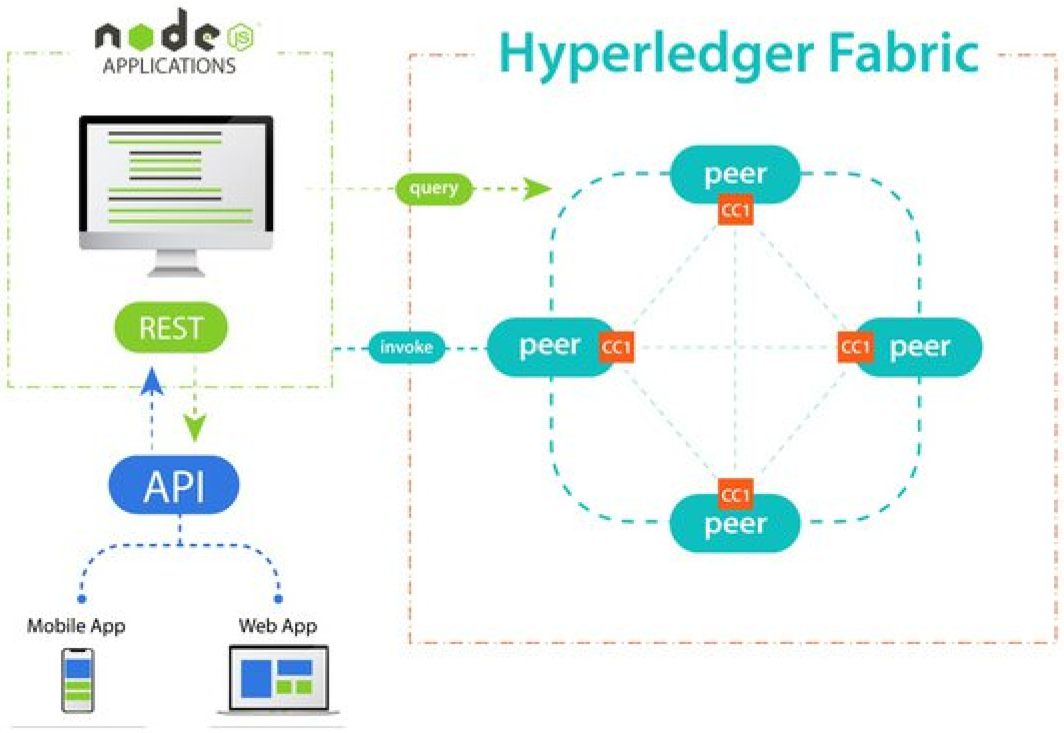
Dimension	Usage	Price	Monthly Total
Write I/O Requests	78.89 million	$0.70/million	$55.23
Read I/O Requests	394.47 million	$0.136/million	$53.65
Journal Storage	2 TB	$0.03/GB-month	$60.00
Indexed Storage	600 GB	$0.25/GB-month	$150.00
Data Transfer Out	15 GB	$0.00 (<100 GB)	$0.00
			$320.14

Exploring the master slave model and addition of node2:









1. Setting up a development environment. Our application needs a network to interact with, so we'll download one stripped down to just the components we need for registration/enrollment, queries and updates

2. the parameters of the sample smart contract our app will use. Our smart contract contains various functions that allow us to interact with the ledger in different ways. We'll go in and inspect that smart contract to learn about the functions our applications will be using.

3. Developing the applications to be able to query and update assets on the ledger. We'll get into the app code itself and manually manipulate the variables to run different kinds of queries and updates

An application has to follow six basic steps to submit a transaction:

- Select an identity from a wallet
- Connect to a gateway
- Access the desired network
- Construct a transaction request for a smart contract
- Submit the transaction to the network
- Process the response

Alerts / Notifications:

If you like to send a notification to an external legacy system every time a new transaction is successfully committed to the ledger.

To achieve this I thought about two different ways:

1. **HTTP Request** - Is it possible to execute an http request directly from the chaincode? If yes, is it possible to send an http request to the endpoint of the legacy system to notify the transaction?
    a. **HTTP Request:** Yes, this is possible, but definitely not recommended. The chaincode is used to endorse transaction proposals, but one peer's endorsement of a transaction proposal does not necessarily mean that transaction itself will be committed. Sending the request from chaincode execution would be premature.

   b. Additionally, the chaincode is not necessarily (and not supposed to be) executed on only one peer. If you have the request being sent from the chaincode and you have 10 peers executing the chaincode, you're going to have 10 requests going to your legacy system

2. **Event** - I understood that there is the possibility to create events in the chaincode. Is it possible to listen to these events without using the Fabric SDK (I can't integrate the SDK in the legacy system)?
   a. **Event:** You can set custom events in the chaincode, but you won't need them as each sdk supports notification of transaction commitment. While I understand it is not reasonable to embed the sdk within your legacy service, the sdk is most likely the best place for you to listen for events

Solution

You will probably need to cobble together the HTTP Request and Event solutions. Have a separate service use the sdk to send transaction proposals and listen for notifications of transaction commitment. Once a transaction is committed, use this service to send a request to your legacy system's endpoint(webhooks, Alert request etc.,).