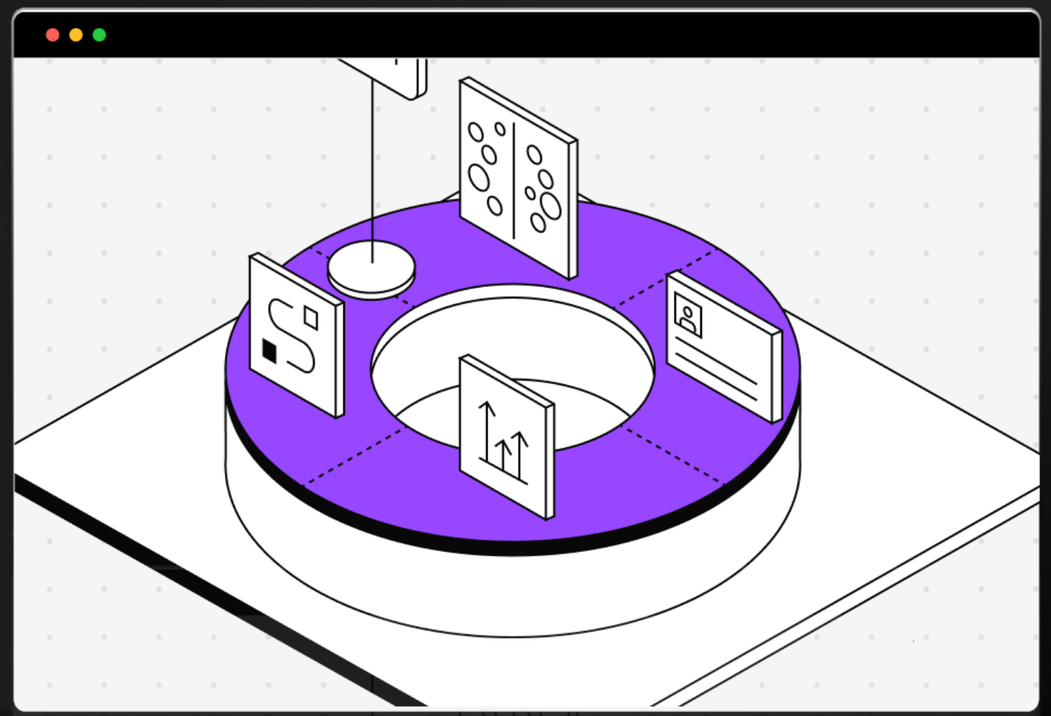
Cockroach Labs

[CASE STUDY]

Starburst

Reach for the stars:
Eliminating data silos for distributed workloads



Starburst

Reach for the stars: Eliminating data silos for distributed workloads

INDUSTRY:
Software (data analytics and data storage)

CHALLENGES:
Building a centralized product with a distributed architecture.

SOLUTION:
A multi-region application running on CockroachDB that guarantees high availability and delivers low latency.

About Starburst

Like many other SaaS companies based on open-source projects, Starburst has its origins in Trino (formerly PrestoSQL) that was founded to help data engineers make better use of their data.

Around 10 years ago, Facebook started to run into the issue of having too much data (300 petabytes) to analyze, just sitting in a data warehouse. As a company that thrives on analytics, they needed to find a better way to run more queries and get results faster. In the fall of 2012, a small group of Facebook Data Infrastructure engineers released Presto, an interactive query system that could operate fast, at petabyte scale.

After a few successful years, the Presto community split off from Facebook, forking it to form PrestoSQL, and the original product became PrestoDB. But this naming convention caused confusion so they changed the name from PrestoSQL to Trino.

Today, Trino is a leading query engine that allows users to query data across distributed data sources and aggregate the results. Starburst delivers an open hybrid data lakehouse that addresses data silos, speed of access problems for their customers, and price-performant SQL analytics. It's the fastest and most efficient SQL-based MPP query engine for lakehouse, data lake, data warehouse, or hybrid environments. It unlocks the value of distributed data in and around the data lake by making it fast and easy to access.

In order to support their global customer base, Starburst had to build an application that could span across multiple geographies, achieve massive scale, and ensure 24/7 uptime – which is why they built Starburst Galaxy, their fully managed cloud lakehouse, on CockroachDB.

Watch YouTube**:** How Starburst uses CockroachDB to eliminate data silos for distributed workloads

"Why would I have my engineering team try to solve the multi-region problem when there's an engineering team out there that's already built a solid solution? We needed to make defensive, smart technology solution choices because we are on the hook for our customer's data."

– Ken Pickering
VP of Engineering, Starburst

Challenges with handling distributed data

First, it's important to understand that Starburst's architecture is massively decentralized because they serve customers that have distributed data. Starburst runs everywhere – across AWS, GCP, Azure – in the EU, US, APAC – and so on.

The Starburst team was dealing with a unique type of wide-scale global distribution which is when their biggest challenge became clear: how do you build a centralized product with a distributed architecture?

Since they run managed Trino in multiple clouds and in multiple regions, they need their customers to be able to access their data in multiple clouds and regions. The team wanted to manage and scale this setup, and they needed to move data close to their customers to reduce latency. Speed of access and time to insight would be critical components to the success of the product.

They also needed a primary application database for the Starburst Galaxy web application which stores standard data like account information and protected passwords. They also built their own metastore service and needed a reliable backend that could be used to access and query data anywhere globally. Replicating this metadata globally would be necessary, yet difficult to achieve without the right infrastructure to support them.

They needed a backend solution for this data, and they required the ability to

- Deploy the application across multiple regions
- Ensure high availability
- Bring data close to customers
- Guarantee low latency (especially for reads)

Given these requirements, their options were limited. They didn't want to use Spanner since they didn't want to be tied to a cloud provider. They could have used Postgres, but they would have had to replicate it themselves and figure out the supporting tech that would allow them to make deploying across multiple regions possible.

And they certainly didn't want to write it from scratch. They needed a cloud-agnostic solution that solved the multi-region challenge for them, which led them to CockroachDB.

6

CLOUD REGIONS

2TB

QUERY HISTORY TABLE

40–80ms

ROUNDTRIP FOR QUERIES

Keep read latency low & meet survival goals

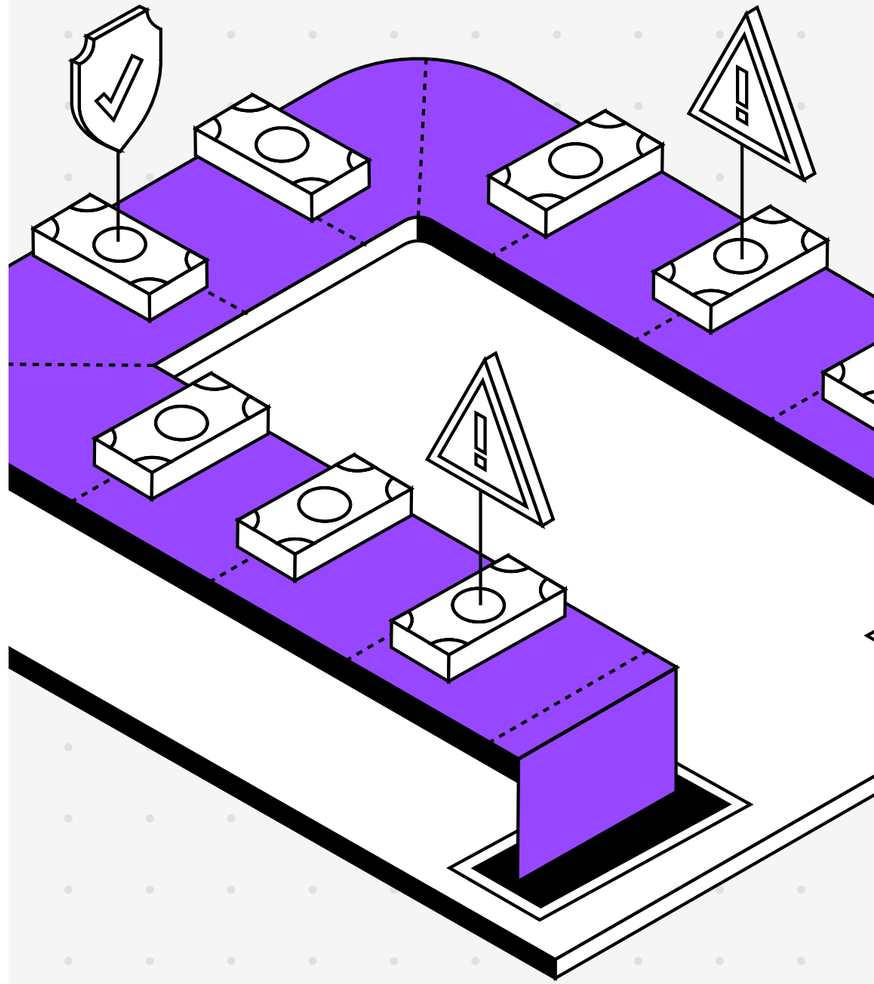
Once they selected CockroachDB Cloud (the managed offering) to support their cloud lakehouse platform (Starburst Galaxy) they needed to figure out the best way to set up regions close to their customers to reduce latency and meet data compliance regulations. They also needed to ensure that if (and when) failover happens, their customers can still get access to their data.

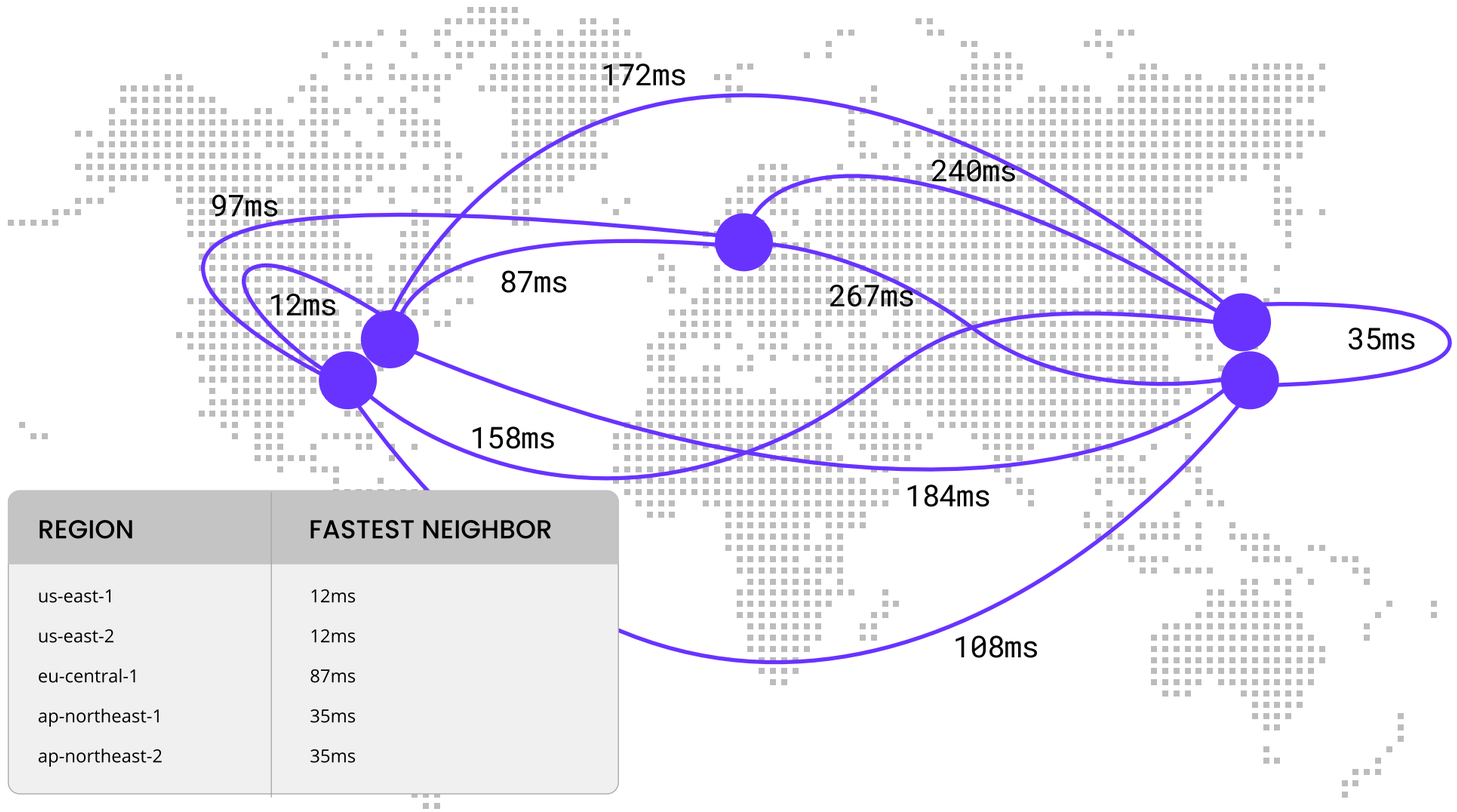
They intended to serve customers in the United States, Europe, Asia Pacific, and eventually South America. If they did not set up regions correctly, their platform could be slow. Their tables are mostly read-heavy, and when it came to performance optimizations, it was really important to them that their lakehouse provides optimal price-performance.

Ideally, when a customer queries their data, they would hit their local read node and not have to cross the world to access the data. This meant that their reads needed to be near-instant, while writes could be somewhat slower. Over time, the team thought that their workload would become increasingly read-heavy (as most writes occur at the initial setup), so they needed to tackle this challenge now.

Given the nature of this data, they decided that they would leverage CockroachDB's global table locality to ensure that read latency was really low. They were willing to compromise write latency in this scenario.

Prior to going into production, they worked with the Cockroach Labs team to determine the best scenario for minimizing latency. For example, they could have deployed their application across 4 AWS regions spread out among us-west-1, us-east1, eu-central, and ap-northeast-1.





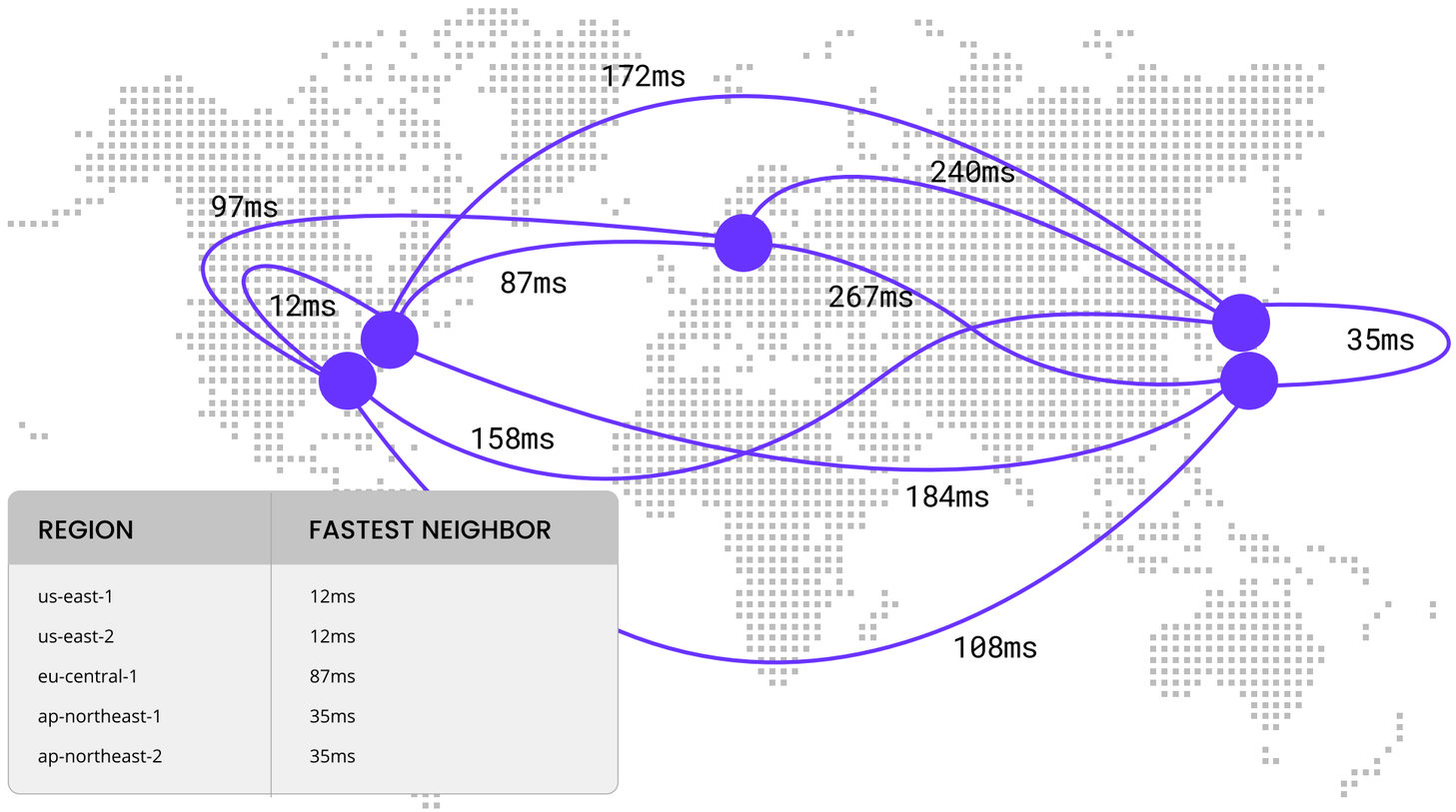
REGION	FASTEST NEIGHBOR
us-east-1	12ms
us-east-2	12ms
eu-central-1	87ms
ap-northeast-1	35ms
ap-northeast-2	35ms

However, their survival goal = region failure, meaning that if an entire region went down (and assuming the app in that region is also down) their customers would be able to access their data from a replica in a region nearby. Given this survival goal, it was best to rearrange their setup and add an additional region.

"Starburst runs an enterprise grade distribution of Trino where the customer's data resides in and around their data lake. We don't ask a customer to move their data, so we have to be present in every cloud and every region. Having CockroachDB from the onset made it so we never had to invest in trying to figure out regional replication and keeping our online transactional database consistent across all of the different corners of the world."

– Matt Stephenson

Senior Principal Software Engineer, Starburst



REGION	FASTEST NEIGHBOR
us-east-1	12ms
us-east-2	12ms
eu-central-1	87ms
ap-northeast-1	35ms
ap-northeast-2	35ms

In the event that a region goes down, they have a plan:

- US users can have replicas in us-east-1, us-east-2 and eu-central-1
- APAC users can have replicas in ap-northeast-1, ap-northeast-2 and us-east-2
- EU users can have replicas in eu-central-1, us-east-1, and us-east-2)

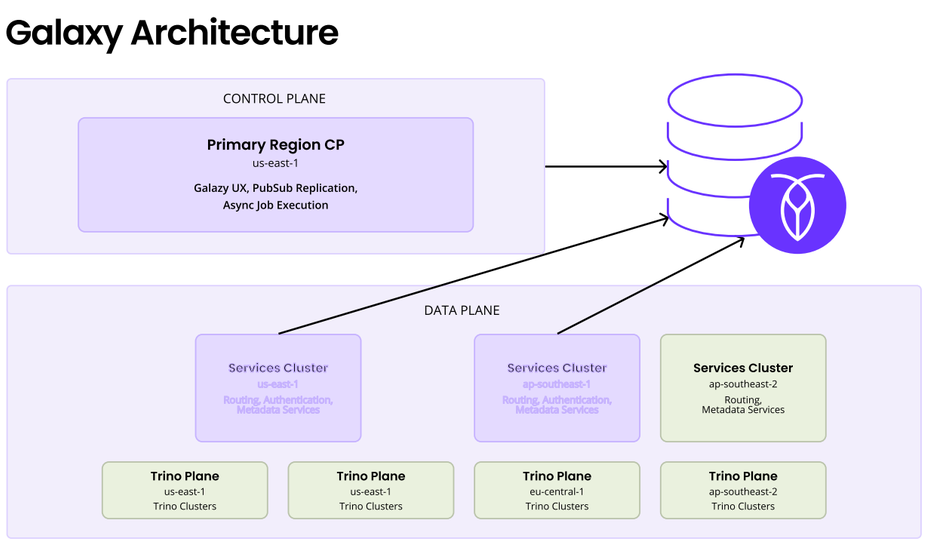
Since Starburst went into production in November 2021, their approach to achieve their survival goals and minimize latency for their customers has been successful. They've also added a sixth region (ap-southeast-1) and will continue to add more in the future to meet their customers where they are.

A backbone for mission-critical use cases

Since going into production, CockroachDB has served as the backbone of Starburst's entire control plan and services plan (a substrate layer that's deployed cross-regional). Starburst also keeps a massive 2TB table of their entire query history in CockroachDB. This means whenever queries are submitted inside of Starburst clusters, those queries come through an event bus (router) back into CockroachDB, and then are made visible to the users in the UI. All of their RBAC systems, internal authentication system, and authorization system are also running on CockroachDB.

The team says that at some point, they have to assume a region will go down. But even if their primary control plane goes down and a failover hasn't happened, the customers can still get access to their data because CockroachDB makes it available in another region.

Here's an overview of what the Starburst Galaxy architecture looks like:



Because Starburst is running across multiple regions and can guarantee SLAs around latency, they say they are able to simplify complexity for their developers. For example, they can guarantee a 40-80 millisecond round trip time for their substrate layer, meaning queries coming from the service plane. Because they don't have to think about or program around a high latency link to the database, it makes developing applications much more straightforward.

> *"The reason we selected [CockroachDB] in the first place was that it was an obvious choice to be able to cut away a whole lot of engineering effort and time, and today we still see that has paid off."*
>
> – **Matt Stephenson**
> Senior Principal Software Engineer, Starburst

Multi-region now, multi-cloud later?

There are certain assumptions within a tech stack that push you to operate a certain way. For example, you select one cloud provider, your ecosystem becomes static, and you have a certain amount of instances you are running at a certain threshold. But it doesn't have to be this way.

Yes, there can be technical challenges associated with building a multi-region application, but for companies like Starburst, the benefits greatly outweigh the challenges. One decision that can be difficult is choosing the right cloud vendor to partner with.

All the vendors offer different regions and availability zones, and have nuances when it comes to operating their systems. It's tough to find engineers who can run systems in different environments because it requires specialized expertises. Plus, network security is handled differently within each cloud environment which adds an additional layer of complexity.

Developing a plan for when (not if) an outage happens, is a responsibility Starburst takes very seriously. Right now their primary control plane is AWS and they have designed a multi-region application on CockroachDB so that they can failover to an adjacent region and ensure availability.

Their architecture is pretty portable and because CockroachDB is cloud-native and cloud-agnostic, they have options for the future should they want to change this setup. For example, they could choose to failover to a different cloud provider which is a good strategy to minimize downtime. Or they could choose to deploy across multiple clouds which would ensure availability even in the face of a total cloud failure. Regardless of their strategy, knowing that they have options is key.

Another reason why Starburst likes to be able to control where and when they spin up a new region (or even cloud) is so they can maintain flexibility to meet the unique and evolving needs of their customers. For example, if they have a new or existing customer running a proof of concept (POC) in South Africa, they can spin up a CockroachDB replica in South Africa, enabling the customer to experience Starburst Galaxy in the region. If that POC doesn't turn into a customer, or they decide not to run that workload in that region, Starburst can easily shut the replica down.

> *"In the future, we don't want one database for AWS, one for GCP, and one for Azure. Our on-call and observability teams will have to look in a bunch of different places to find information. So CockroachDB's flexibility to allow us to build the way that we did was really important."*
>
> – **Matt Stephenson**
> Senior Principal Software Engineer, Starburst

What's Next

Today, as organizations adopt lakehouse architectures for their data, analytics, and AI needs, many companies still struggle with developing a single source of truth to access their data and they are growing increasingly tired of data management.

Starburst helps all types of companies that have distributed data (running workloads across multiple business units or multiple clouds) link it all together through secure data access. Their customer use cases span from embedded applications, customer analytics, anti-money laundering and stock trade violations, supply chain analytics, log analytics, and more at massive scale.

Over the next few years, Starburst is focused on bringing the lakehouse to meet the data, analytics, and AI needs of customers across hybrid, mulit-cloud, and on-premises by improving their products and growing their customer base. They believe that there's a future in separating storage and compute, and if they can make this easy to use, it will be a game changer. Along the way they will be relying on the power of Trino and CockroachDB to help make hybrid, multi-region, and multi-cloud use cases a reality.

To learn more about Starburst, check out their website: starburst.io.

Ready to get started?

Go hands-on with 100% free CockroachDB Serverless. Spin up your first cluster in just a few clicks.

Get Started →

Compare Products →

"Yeah the tech is really cool, and it works great. But even more importantly, we trust our business to run on CockroachDB. It's storing our critical assets for us which is our customer data. We also know that there is a talented team behind the product that's there to help us every step of the way."

– Matt Stephenson

Senior Principal Software Engineer, Starburst