

FROM RESILIENCE TO SCALABILITY:

12 Mission-Critical CockroachDB Use Cases

Why (and how) global enterprises rely on CockroachDB to drive business growth and meet customer expectations



FROM RESILIENCE TO SCALABILITY: 12 Mission-Critical CockroachDB Use Cases

Why (and how) global enterprises rely on
CockroachDB to drive business growth
and meet customer expectations

If you've paid one of your bills online, ordered from your favorite ecommerce site, or navigated a series of security steps to verify your identity, then you've interacted with the types of services powered by CockroachDB.

Use case interactions are part of a broader experience that customers expect to be instant, seamless, and always available. In order to meet these expectations, applications require a resilient underlying infrastructure that can effortlessly scale and serve its users regardless of their location or time of day.

Not only do organizations need to quickly adapt to evolving customer expectations, but they also need to differentiate their product and continue innovating ahead of their competition. They must accomplish these business-level objectives without adding more complexity to their infrastructure, which requires a database built for modern application development.

The right database can provide a great experience for users without having to build resiliency, consistency, and data replication in the application layer. However, if the

wrong database is used for the wrong use case, it can put applications at risk for downtime, cause latency issues for users, prevent organizations from meeting compliance regulations, and limit the ability to scale.

CockroachDB addresses modern application development challenges by delivering a distributed SQL foundation with built-in resilience and scale. We've worked with thousands of customers over the years who have architected mission-critical use cases on CockroachDB so that they can future-proof their infrastructure and deliver new and exciting experiences for their customers.

Oftentimes, organizations use CockroachDB as a transactional data store for several use cases that are critical to their survival. In this guide, we will discuss 12 common use cases for CockroachDB and dive deeper into the four most popular: User accounts and metadata, identity access management, payments, and orders and inventory management. These use cases span all different types of verticals, so we have organized the guide into three groups: use cases serving People, Money, and Things.

People 4

User Accounts and Metadata 5

Identity Access Management (IAM) 7

Gaming 9

Money 10

Payments 11

Banking and Wallet 13

Trading 13

Betting 14

Things 15

Orders and Inventory Management 16

Routing and Logistics 18

IoT and Device Management 18

Digital Storage Management 19

AI/ML Feature Store 19

People

People are what power the economy and can make or break any company's success. The actions customers take within a particular application or with a service directly impact performance and can test the limits of the software's infrastructure.

User Accounts and Metadata

User accounts and metadata refers to the identity created for the user when accessing various account types (i.e. service or system accounts), along with managing the metadata related to users and associated objects.

This use case is a critical component of any business application and required to run SaaS services. It involves a strategy that ensures the data is identified, captured, maintained, and retrieved, wherever it's needed.

For example, when a user signs up for a service, they will create an account, profile, objects, and have actions they take which all need to be recorded in a database. Choosing the right database to support a user metadata use case is crucial and can help automate many workflows.

Challenges

After a user signs up for a service, they will start to generate metadata that has to be tied to their particular account. This includes information such as accounts, bills and invoices, user profile and preferences, permissions, assets, and other related user information. These relationships need to be tracked correctly, meaning the user's data must always be available for them to access. If an inconsistency is introduced, it can be very difficult to recover the correct information, which puts the business at risk during compliance audits.

Accounts and metadata availability are crucial to an application's functionality. For example, if your metadata database goes offline, users won't be able to access their

profile and associated information which degrades the customer experience.

Another challenge is being able to scale user accounts and metadata workloads as your business acquires new customers. If your organization expands its reach into new regions, you need to be able to partition user data to live within certain geographies for low latency or regulatory compliance.

Database requirements

Data consistency is a requirement for architecting user accounts and metadata systems. Metadata is often duplicated across multiple databases (e.g. for user access or for internal analytics) so it's imperative that the data is always correct. Leveraging a multi-region database foundation can help when it comes to scale, data locality, and fast user access.

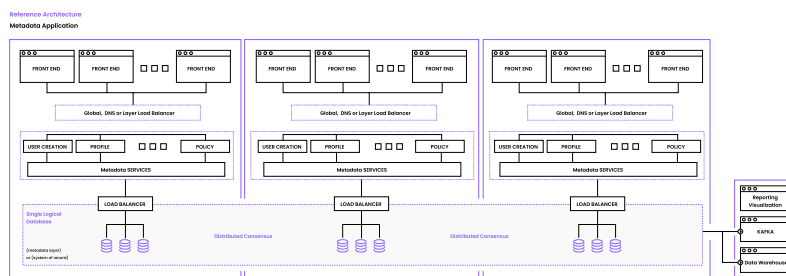
An active-passive system puts too much at risk when it comes to data loss and availability. If the primary database goes offline, there's a period of time before failing over to the secondary where data is not available and could potentially be lost. Choosing an active-active or multi-active system will better ensure your user's metadata is available and consistent.

Why CockroachDB for user accounts and metadata?

Not only does CockroachDB meet the requirements for this use case, but it also makes it significantly easier to build multi-region services and automate metadata management using built-in features like change data capture (CDC).

Reference architecture

This diagram shows how an application might integrate CockroachDB as a user metadata store. The multi-region setup offers inherent benefits in terms of global access and impact to user latency.



Customer Story: Starburst guarantees user metadata availability with CockroachDB

Starburst offers an enterprise and cloud solution that addresses data silos and speed of access problems for their customers. Because their customer base is global, Starburst needed to build an application that could span across multiple geographies, achieve massive scale, and ensure 24/7 uptime. These requirements lead them to select CockroachDB as a backend to replicate their metadata.

Their survival goal = region failure, so they run CockroachDB across 5 regions in order to survive the failure of an entire region, and to minimize latency for their distributed customers.

"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."



Ken Pickering
VP of Engineering

To learn more about Starburst's use case and read other customer stories, visit cockroachlabs.com/customers ↗

PEOPLE

Identity Access Management (IAM)

Identity access management (IAM) is the process of authenticating and authorizing users (or services) to securely access systems and data.



IAM systems are mission critical because they allow users or services to gain access to the right systems and data at the right time. This use case is often connected to user metadata since they both deal with the management of user activity.

Usually IAM comes in two forms: authentication or authorization. Authentication verifies the identity of a user/service, whereas authorization allows the user/service to gain access to parts of a system and data. This use case requires a database that can deliver fast, consistent access to users regardless of their location.

Challenges

IAM systems cannot go offline because the business impact is too substantial. Every minute that authentication or authorization systems are down, the company loses money and their customers' trust. Even planned downtime for maintenance or software upgrades is unacceptable, since many businesses serve a global audience spanning different time zones.

Catering to global users also creates challenges when it comes to maintaining high availability. You want your database to be physically located close to users for fast access, yet you also need to make sure that you replicate

your authentication data to a backup region. In the event of a node, zone, or region failure, chaos can ensue.

Additionally, data inconsistencies can cause major security issues when authorizing users (i.e. the right permissions aren't attached to their profile). As you continue to scale the volume of users, more inconsistencies can be introduced if you aren't using a resilient database system that guarantees data correctness through synchronous data replication.

Database requirements

Because most IAM systems cater to users or systems that are distributed in different locations, they require a database that can span multiple regions and/or data centers and have the ability to domicile data in a particular location to aid with compliance. A multi-region database also allows you to maintain high availability in the event of a total region failure.

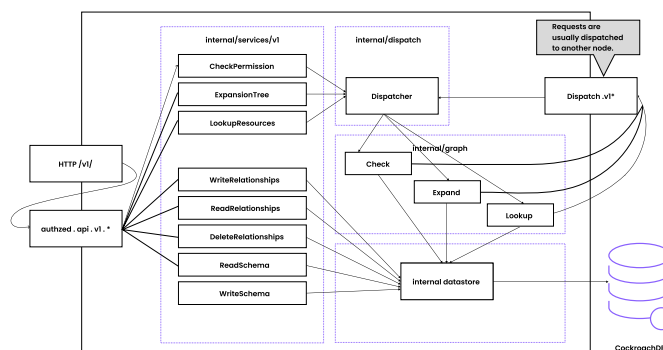
Additionally, IAM systems must be able to handle an increase in users and/or services added to the system. They require a distributed database that can seamlessly adapt to an uptick in workloads without causing outages or the need for complex manual sharding configurations. Even as you scale, your data must always be consistent, which requires a database with ACID guarantees to ensure data integrity is maintained.

Why CockroachDB for identity access management?

IAM systems require CockroachDB's built-in replication, geo-distribution, and ACID guarantees to quickly authenticate and authorize secure access to systems and data anywhere, every time.

Reference architecture

This diagram is a high-level overview of a SpiceDB deployment (an open-source system for managing security-critical app permissions with CockroachDB as the distributed backend store). The request is handled by the dispatch interface and persisted into the datastore.



SALTO

Customer Story: Salto's IAM system powered by CockroachDB

SALTO is a leader in cloud-based access management technology and needed a highly available database to back their user access management system.

They previously built systems on SQL Server where they often experienced downtime for schema changes/database upgrades, and knew they were at risk for outages on an active-passive system. They built a new product from scratch on CockroachDB which runs across 3 regions and delivers 24/7 availability on an multi-active system. They no longer need to take the database offline for maintenance and were able to simplify their architecture by leveraging CDC.

"We wanted to decouple the developer from the infrastructure team. We didn't want to make the developer responsible for knowing that there's a difference between primary, failover, or read replicas. We just wanted them to use the database; that's all... we were impressed with how CockroachDB just worked."



Gorka Lerchundi
Engineering Manager & Tech Lead

To learn more about Salto's use case and read other customer stories, visit cockroachlabs.com/customers ↗



Gaming

Gaming platforms involve the management of users, assets, and state of a game.

Aside from game developers, there are several different types of companies that leverage a gaming platform. For example, software publishers, hospitality groups, and media companies all need to monitor users' actions within an application.

Similar to the other “people” use cases, a gaming platform relies heavily on the database's ability to scale up to accommodate an increase in traffic. However, this type of scale is often unpredictable, and a dramatic increase in workload volume often knocks gaming platforms offline.

Why CockroachDB for gaming platforms?

CockroachDB allows organizations building gaming platforms to prepare for unexpected scale and support millions of concurrent players in real-time without downtime.

Money

Aside from fintechs, banks, and various financial institutions, other industries rely on the management of people's money – from retailers with point of sale systems, to sports betting apps with real-money gaming, to logistics companies that need to bill their customers. People might make or break a company's success, but money makes the world go round. The movement of money naturally generates heavy transactional workloads which requires the right underlying foundation.

Payments

Payments is the process of initiating, authorizing, capturing, and exchanging a monetary value.

Virtually every business in one way or another has to deal with payment processing. It's easy to understand how a missed or incorrect payment impacts a company's bottom line and credibility. It's also easy to understand how frustrating it is when a payment system isn't working or charges you incorrectly.

Typically there are two types of payment use cases: payment processing and payment gateways. A payment processing system resolves money movement between financial institutions, while a payment gateway accepts payments. There are also subcategories within the payments ecosystem which include use cases such as card authorization, fraud, rewards, and gift cards.

Challenges

The speed and correctness of a payment matters during the time in which a payment transaction takes place. However, because of the possibility of things like returns, refunds, disputes, and audits, payment data typically needs to remain available for years.

Customers (and auditors) are not forgiving when it comes to money which means data correctness and availability is critical. Even a very small amount of downtime or a missed calculation could equate to financial fines, lost revenue, brand deterioration, and customer churn.

Without question, security and audit capabilities are table stakes for these systems. Successful businesses utilize key audit information generated from payment transactions to

ensure quality, protect against fraud, conduct root cause analysis, and instrument the overall system. Ensuring that logging, security and auditing doesn't overwhelm the system is a constant trade off with system performance.

Lastly, relying on your application (as opposed to your infrastructure) to handle consistency, resiliency, and scale creates challenges for your operations and development teams. Building consistency checks in your application takes effort and creates excess code. Building highly resilient applications without an active-active setup takes a lot of toil and complex operational runbooks. Manually sharding in order to scale requires teams to re-architect the application so it will work in the future.

Database requirements

Security and auditing of all interactions, including data mutations and data access, need to be available and integrated into the organizational authentication, authorization, auditing, and encryption systems. Security and audit is the one feature that gives business owners a peace of mind about the database.

The actual process of storing payment data in a database isn't complex or particularly different from storing other types of data. At smaller scales, the consistency and correctness requirements for payments can be met with the ACID transactional guarantees offered by most SQL databases. For larger systems, it's wise to build a payment infrastructure with correctness in mind which means using a database that allows you to achieve idempotency.

As a company scales up, the regular transactional load placed on the database, as well as the ever-increasing total volume of data, can quickly become a performance bottleneck. Maintaining performance as your business grows requires a solution that allows you to scale up your

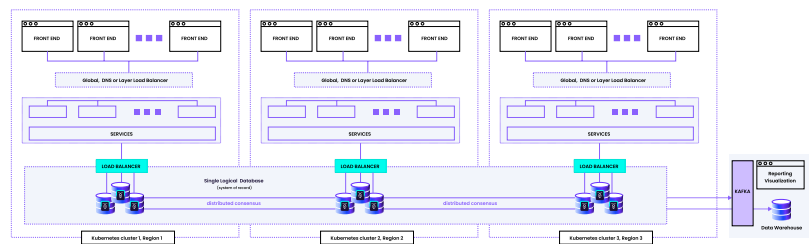
payments database without sacrificing correctness or availability. You'll also want a database that doesn't require manual sharding.

Why CockroachDB for payments?

Payments systems rely on CockroachDB to handle the complexity in the database layer so that they can deliver customers real-time transactional correctness all the time, every time.

Reference architecture

Visualize how CockroachDB's dramatically simple multi-region architecture enables payment systems to be highly available and strongly consistent with greatly reduced operational upkeep.



Customer Story: Shipt relies on CockroachDB to deliver correct payments

Shipt, a grocery ecommerce company owned by Target, maintains a suite of payment services that are crucial to their business model. Here's a basic model of how a transaction flows through their payment service and how they set up idempotency.

For Shipt, it was crucial to have concurrency control in a distributed system so that they could block concurrent requests for the same payment. They built their system on CockroachDB which allows them to achieve guaranteed atomicity, consistency, isolation, and durability (ACID) all the way down to the row level.

When working with CockroachDB you have to develop a distributed mindset. We knew we wanted to build a multi-region application so we started to ask ourselves, what part of the application layer do we put in the complexity to make this work? But with CockroachDB, the complexity is not in the application layer, a majority of it is handled by the database. This makes it substantially easier to build multi-region services.



David Templin
Senior Software Engineer

To learn more about Shipt's use case and read other customer stories, visit cockroachlabs.com/customers





Banking and Wallet

Banking and wallet refers to the tracking of financial transactions such as debits and credits at a transactional and account level. This use case may also be referred to as a financial ledger or system of record which means it is an authoritative source of data for a particular type of financial information.

Companies have to modernize their banking systems so that they can keep up with the agile fintechs who are targeting the younger generations. Banking and wallet use cases may include several different types of applications and services including deposits, debit cards, savings, rewards, crypto, fraud detection, and more.

All of these services depend on a series of requirements for the movement of money that's organized into accounts with computed balances, business logic, and workflows. If the requirements are not met, they might have a tremendous impact on the customer experience and ultimately impact the company's bottom line.

Why CockroachDB for banking and wallet?

Banking and wallet use cases require a mission-critical database backend that's never down and always ensures data consistency.



Trading

Trading is the ordering and exchange of assets such as equities, fixed income, commodities, retail trades, 401K, and more.

Similar to payments and banking and wallet use cases, trading deals with the movement of money and sensitive data. Interruptions to these services can cost companies severe reputational damage that results in massive regulatory fines and loss of customer trust.

At market open, a trading system needs to sequentially process queued orders received prior to open. This peak activity can result in hundreds of thousands of database transactions that must be executed consistently within a regulated timeframe. During the trading day, the volumes of trade typically level off, however there's always a possibility for sudden spikes.

Why CockroachDB for trading?

Trading use cases require CockroachDB's ability to horizontal scale to accommodate a spike in transactional workloads without causing downtime or creating data inconsistencies.



Betting

Betting involves the coordination, management, and processing of bids, bets, and/or auctions in the exchange for goods, services, and/or money.

While the core component of betting software is the management of purchases, payments, and digital wallets, it also requires the handling of bet placements, metadata, and user accounts. Customer loyalty is won by ensuring your application and services are always performant, stats are updated in real-time, and every payout is penny-perfect.

Users expect bet placements and settlements to happen instantly which requires an infrastructure that supports concurrent distributed transactions. Sports betting and gambling are highly regulated industries and organizations must adhere to compliance requirements, which means they have to control the location of data.

Why CockroachDB for betting?

CockroachDB handles bet placement and settlement in a serializable fashion to ensure data consistency while delivering a highly available foundation that can control the location of data.

Things

Like people and money, businesses have many “things” that need to be managed, tracked, and stored. Some of these use cases have been around for decades – railroad routing systems, factory order management, shipping logistics – and are rapidly evolving to fit into the modern era. Others have emerged more recently to keep pace with the increased volume of internet-connected devices and machine intelligence.

Orders and Inventory Management

Orders and inventory management involves managing the status, flow, and state of inventory that is purchased through orders.

Orders and inventory management is used in any industry that needs a heavily optimized process for moving users or goods along a funnel to be purchased. Without this system of movement, purchases can not be completed and the business can not make money.

This use case is heavily dictated by the consumer's experience. They have high demands when it comes to receiving their purchases in a timely manner and they are easily frustrated when items go out of stock.

Challenges

Orders and inventory management is often cyclical, meaning that there are predicted moments in time for when there's a spike in orders (holiday shopping season, yearly sales, etc). Capacity planning can become a challenge when budgets are tight and companies don't want to overspend on infrastructure. At the same time, spikes in traffic can knock a system offline if the supporting infrastructure is not prepared to handle it. Downtime (unplanned or planned) is detrimental to this use case.

This use case naturally creates a distributed system since the customers, factories, headquarters, physical stores, etc. are all in different locations. This distributed nature creates challenges when it comes to achieving a consolidated view of your data, meeting data privacy regulations, and ensuring end-users experience minimal latency.

Additionally, the consistency of data is important to this use case because there needs to be several workflows in place to ensure that the placement and modification of orders is propagated downstream. If data consistency is only eventually consistent, it can create a lot of operational challenges.

Database requirements

A horizontally scalable database is needed to manage fluctuating workloads and reduce the overall infrastructure spend. Additionally, a database that can scale across multiple regions aids in reaching a distributed customer base and lowering end-user latency.

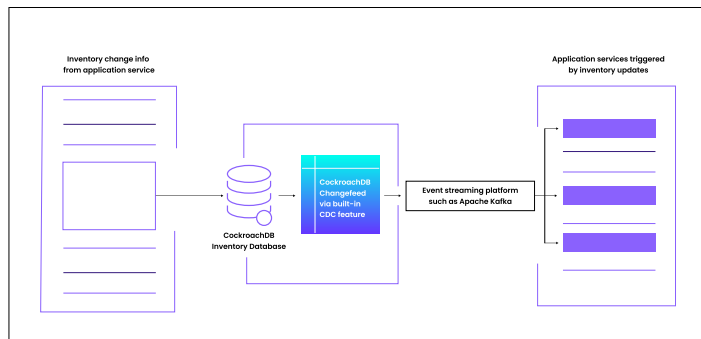
To make orders and inventory management systems less complex, it's useful if a database has built-in features such as ACID guarantees, replication, and change data capture (CDC). Data accuracy should be guaranteed and data should be replicated so it's never lost. CDC helps ensure all changes to the system are made to the data wherever it lives.

Why CockroachDB for orders and inventory management?

Orders and inventory management benefits from the distributed, resilient nature of CockroachDB and leverages the ability to horizontally scale up/down to save on yearly infrastructure costs.

Reference architecture

In the diagram, an application microservice updates the CockroachDB table with an inventory change. Since other services need that information, CockroachDB's CDC feature generates a changefeed that sends the inventory update to Kafka which triggers other application workflows.



Customer Story: CockroachDB powers Booksy's order management

Booksy is one of the most widely used scheduling apps for instant online appointment booking in hair salons, beauty salons, or barber shops.

They originally built their platform on PostgreSQL, but decided to switch to CockroachDB so that they could achieve 10x growth over the next 2 years. Since making the switch, Booksy's service level objective (SLO) is at the highest level of five nines. Their customers greatly benefit from a platform that's available 24/7, delivers low latency access, and can automatically scale to accommodate increases in traffic.

"Before migrating to CockroachDB we had challenges with resiliency, scaling, and keeping up with customer's needs... now we have a globally available platform."



Pawel Sobkowiak
Chief Technology Officer, Booksy

To learn more about Booksy's use case and read other customer stories, visit cockroachlabs.com/customers ↗



Routing and Logistics

Routing and logistics tracks and sequences the operations of a physical good. It involves real-time tracking of vehicles, monitoring inventory levels, and optimizing routes for efficient delivery.

Like orders and inventory management, this use case demands a sophisticated infrastructure to manage the complexities of supply chain operations. If this use case is not implemented correctly, it can cause massive headaches for operations teams that will spend countless hours trying to maintain the system.

The technology stack for this use case needs to support a distributed architecture and ensure data synchronization across various locations. Geospatial capabilities are essential for accurate mapping and route optimization. Strong consistency is required to maintain data integrity even as you scale the system.

Why CockroachDB for routing and logistics?

CockroachDB is a fitting solution for routing and logistics due to its distributed nature, global consistency, and support for geospatial data types.



IoT and Device Management

IoT and device management involves the deployment, control, management, and monitoring of devices over the internet.

As the number of connected devices continues to grow, so does the need for a scalable, yet flexible, infrastructure. This use case spans several industries such as consumer electronics, manufacturing, healthcare, oil and gas, smart cities, etc.

IoT and device management use cases typically deal with two types of application data: management data (i.e. information about the device itself such as location and status) and event data. Management data is often updated while event data is measured in near real-time. Both types can generate data quickly, however, and management data is especially reliant on the correctness of data.

Why CockroachDB for IoT and device management?

CockroachDB delivers a zero-downtime foundation for IoT and device management and can handle increases in workloads without impacting data correctness.



Digital Storage Management

Digital storage management is about managing the metadata related to objects which is required to find where a specific file or digital asset is stored.

All sorts of companies have to store digital media in various forms such as files, videos, documents, and objects. This media needs to be quickly located and accessed for it to be of any value. It's also useful, from a latency standpoint, if the media is stored close to the end user.

As the volume of a company's digital storage increases, the infrastructure must scale (compute and storage) to support it. Without the right database foundation in place, application performance and availability can take a hit as you scale.

Why CockroachDB for digital storage management?

CockroachDB enables organizations to easily scale digital storage management systems without sacrificing data consistency or availability.



AI/ML Feature Store

AI / ML feature store involves the storage and lookup of model data that is required to generate a response or recommendation.

This use case supports applications requiring real-time, accurate data to feed into generative AI models for content generation, decision-making processes, or personalized user experiences. Maintaining an AI/ML feature store can create several operational and technical challenges.

Traditional relational databases often struggle with the volume and velocity of data requests generated by AI models, leading to performance bottlenecks, data inconsistencies, and potential downtime. Organizations need a well-architected solution to support high-performance AI models that is efficient, secure, and cost effective.

Why CockroachDB for AI/ML feature store?

CockroachDB provides horizontal scalability to ensure that the database can grow alongside the AI application without impacting performance or data accuracy.

Thanks for reading

We hope you found this information useful for understanding how CockroachDB can be applied to use cases at your organization.

Cockroach Labs has dozens of experts that specialize in use case development. To get in touch, visit cockroachlabs.com/contact. ↗