Product FAQs

# Frequently Asked Questions

CONTRIBUTE

## On This Page

## What is CockroachDB?

CockroachDB is a distributed SQL database built on a transactional and strongly-consistent key-value store. It **scales** horizontally; **survives** disk, machine, rack, and even datacenter failures with minimal latency disruption and no manual intervention; supports **strongly-consistent** ACID transactions; and provides a familiar **SQL** API for structuring, manipulating, and querying data.

CockroachDB is inspired by Google's Spanner (http://research.google.com/archive/spanner.html) and F1 (http://research.google.com/pubs/pub38125.html) technologies, and it's completely open source (https://github.com/cockroachdb/cockroach).

## When is CockroachDB a good choice?

CockroachDB is well suited for applications that require reliable, available, and correct data regardless of scale. It is built to automatically replicate, rebalance, and recover with minimal configuration and operational overhead. Specific use cases include:

- Distributed or replicated OLTP
- Multi-datacenter deployments
- Cloud-native infrastructure initiatives

## When is CockroachDB not a good choice?

CockroachDB is not a good choice when very low latency reads and writes are critical; use an in-memory database instead.

Also, CockroachDB is not yet suitable for:

- Complex SQL Joins (the feature still needs optimization (https://www.cockroachlabs.com/blog/cockroachdbs-first-join/))
- Heavy analytics / OLAP

## How easy is it to install CockroachDB?

It's as easy as downloading a binary on OS X and Linux or running our official Docker image on Windows. There are other simple install methods as well, such as running our Homebrew recipe on OS X or building from source files on both OS X and Linux.

For more details, see Install CockroachDB (install-cockroachdb.html).

## How does CockroachDB scale?

CockroachDB scales horizontally with minimal operator overhead. You can run it on your local computer, a single server, a corporate development cluster, or a private or public cloud. Adding capacity (start-a-node.html) is as easy as pointing a new node at the running cluster.

At the key-value level, CockroachDB starts off with a single, empty range. As you put data in, this single range eventually reaches a threshold size (64MB by default). When that happens, the data splits into two ranges, each covering a contiguous segment of the entire key-value space. This process continues indefinitely; as new data flows in, existing ranges continue to split into new ranges, aiming to keep a relatively small and consistent range size.

When your cluster spans multiple nodes (physical machines, virtual machines, or containers), newly split ranges are automatically rebalanced to nodes with more capacity. CockroachDB communicates opportunities for rebalancing using a peer-to-peer gossip protocol (https://en.wikipedia.org/wiki/Gossip_protocol) by which nodes exchange network addresses, store capacity, and other information.

## How does CockroachDB survive failures?

CockroachDB is designed to survive software and hardware failures, from server restarts to datacenter outages. This is accomplished without confusing artifacts typical of other distributed systems (e.g., stale reads) using strongly-consistent replication as well as automated repair after failures.

**Replication**

CockroachDB replicates your data for availability and guarantees consistency between replicas using the Raft consensus algorithm (https://raft.github.io/), a popular alternative to Paxos (http://research.microsoft.com/en-us/um/people/lamport/pubs/paxos-simple.pdf). You can define the location of replicas (configure-replication-zones.html) in various ways, depending on the types of failures you want to secure against and your network topology. You can locate replicas on:

- Different servers within a rack to tolerate server failures
- Different servers on different racks within a datacenter to tolerate rack power/network failures
- Different servers in different datacenters to tolerate large scale network or power outages

When replicating across datacenters, be aware that the round trip latency between datacenters will have a direct effect on your database's performance. Latency in cross-continent clusters will be noticeably worse than in clusters where all nodes are geographically close together.

**Automated Repair**

For short-term failures, such as a server restart, CockroachDB uses Raft to continue seamlessly as long as a majority of replicas remain available. Raft makes sure that a new “leader” for each group of replicas is elected if the former leader fails, so that transactions can continue and affected replicas can rejoin their group once they’re back online. For longer-term failures, such as a server/rack going down for an extended period of time or a datacenter outage, CockroachDB automatically rebalances replicas from the missing nodes, using the unaffected replicas as sources. Using capacity information from the gossip network, new locations in the cluster are identified and the missing replicas are re-replicated in a distributed fashion using all available nodes and the aggregate disk and network bandwidth of the cluster.

## How is CockroachDB strongly-consistent?

CockroachDB guarantees the SQL isolation level "serializable", the highest defined by the SQL standard. It does so by combining the Raft consensus algorithm for writes and a custom time-based synchronization algorithms for reads. See our description of strong consistency (strong-consistency.html) for more details.

## How is CockroachDB both highly available and strongly consistent?

The CAP theorem (https://en.wikipedia.org/wiki/CAP_theorem) states that it is impossible for a distributed system to simultaneously provide more than two out of the following three guarantees:

- Consistency
- Availability
- Partition Tolerance

CockroachDB is a CP (consistent and partition tolerant) system. This means that, in the presence of partitions, the system will become unavailable rather than do anything which might cause inconsistent results. For example, writes require acknowledgements from a majority of replicas, and reads require a lease, which can only be transferred to a different node when writes are possible.

Separately, CockroachDB is also Highly Available, although "available" here means something different than the way it is used in the CAP theorem. In the CAP theorem, availability is a binary property, but for High Availability, we talk about availability as a spectrum (using terms like "five nines" for a system that is available 99.999% of the time).

Being both CP and HA means that whenever a majority of replicas can talk to each other, they should be able to make progress. For example, if you deploy CockroachDB to three datacenters and the network link to one of them fails, the other two datacenters should be able to operate normally with only a few seconds' disruption. We do this by attempting to detect partitions and failures quickly and efficiently, transferring leadership to nodes that are able to communicate with the majority, and routing internal traffic away from nodes that are partitioned away.

## Why is CockroachDB SQL?

At the lowest level, CockroachDB is a distributed, strongly-consistent, transactional key-value store, but the external API is Standard SQL with extensions. This provides developers familiar relational concepts such as schemas, tables, columns, and indexes and the ability to structure, manipulate, and query data using well-established and time-proven tools and processes. Also, since CockroachDB supports the PostgreSQL wire protocol, it's simple to get your application talking to Cockroach; just find your PostgreSQL language-specific driver (install-client-drivers.html) and start building.

For more details, learn our basic CockroachDB SQL statements (learn-cockroachdb-sql.html), explore the full SQL grammar (sql-grammar.html), and try it out via our built-in SQL client (use-the-built-in-sql-client.html). Also, to understand how CockroachDB maps SQL table data to key-value storage and how CockroachDB chooses the best index for running a query, see SQL in CockroachDB (https://www.cockroachlabs.com/blog/sql-in-cockroachdb-mapping-table-data-to-key-value-storage/) and Index Selection in CockroachDB (https://www.cockroachlabs.com/blog/index-selection-cockroachdb-2/).

## Does CockroachDB support distributed transactions?

Yes. CockroachDB distributes transactions across your cluster, whether it's a few servers in a single location or many servers across multiple datacenters. Unlike with sharded setups, you don't need to know the precise location of data; you just talk to any node in your cluster and CockroachDB gets your transaction to the right place seamlessly. Distributed transactions proceed without downtime or additional latency while rebalancing is underway. You can even move tables – or entire databases – between data centers or cloud infrastructure providers while the cluster is under load.

## Do transactions in CockroachDB guarantee ACID semantics?

Yes. Every transaction (transactions.html) in CockroachDB guarantees ACID semantics (https://en.wikipedia.org/wiki/ACID) spanning arbitrary tables and rows, even when data is distributed.

- **Atomicity:** Transactions in CockroachDB are "all or nothing." If any part of a transaction fails, the entire transaction is aborted, and the database is left unchanged. If a transaction succeeds, all mutations are applied together with virtual simultaneity. For a detailed discussion of atomicity in CockroachDB transactions, see How CockroachDB Distributes Atomic Transactions (https://www.cockroachlabs.com/blog/how-cockroachdb-distributes-atomic-transactions/).
- **Consistency:** SQL operations never see any intermediate states and move the database from one valid state to another, keeping indexes up to date. Operations always see the results of previously completed statements on overlapping data and maintain specified constraints such as unique columns. For a detailed look at how we've tested CockroachDB for correctness and consistency, see DIY Jepsen Testing of CockroachDB (https://www.cockroachlabs.com/blog/diy-jepsen-testing-cockroachdb/).
- **Isolation:** By default, transactions in CockroachDB use serializable snapshot isolation (SSI). This means that even concurrent read-write transactions will never result in anomalies. We also provide snapshot isolation (SI), which is more performant with high-contention workloads, although it exhibits anomalies not present in SSI (write skew). For a detailed discussion of isolation in CockroachDB transactions, see Serializable, Lockless, Distributed: Isolation in CockroachDB (https://www.cockroachlabs.com/blog/serializable-lockless-distributed-isolation-cockroachdb/).
- **Durability:** In CockroachDB, every acknowledged write has been persisted consistently on a majority of replicas (by default, at least 2) via the Raft consensus algorithm (https://raft.github.io/). Power or disk failures that affect only a minority of replicas (typically 1) do not prevent the cluster from operating and do not lose any data.

## Since CockroachDB is inspired by Spanner, does it require atomic clocks to synchronize time?

No. CockroachDB was designed to work without atomic clocks or GPS clocks. It's an open source database intended to be run on arbitrary collections of nodes, from physical servers in a corp development cluster to public cloud infrastructure using the flavor-of-the-month virtualization layer. It'd be a showstopper to require an external dependency on specialized hardware for clock synchronization.

However, CockroachDB does require moderate levels of clock synchronization for correctness. If clocks drift past a maximum threshold, nodes will be taken offline. It's therefore highly recommended to run NTP (http://www.ntp.org/) or other clock synchronization software on each node.

For more details on how CockroachDB handles unsychronized clocks, see Clock Synchronization (recommended-production-settings.html#clock-synchronization). And for a broader discussion of clocks, and the differences between clocks in Spanner and CockroachDB, see Living Without Atomic Clocks (https://www.cockroachlabs.com/blog/living-without-atomic-clocks/).

## What languages can I use to work with CockroachDB?

CockroachDB supports the PostgreSQL wire protocol, so you can use any available PostgreSQL client drivers. We've tested it from the following languages:

- Go
- Python
- Ruby
- Java
- JavaScript (node.js)
- C++/C
- Clojure
- PHP
- Rust

See Install Client Drivers (install-client-drivers.html) for more details.

## Why does CockroachDB use the PostgreSQL wire protocol instead of the MySQL protocol?

CockroachDB uses the PostgreSQL wire protocol because it is better documented than the MySQL protocol, and because PostgreSQL has a liberal Open Source license, similar to BSD or MIT licenses, whereas MySQL has the more restrictive GNU General Public License.

Note, however, that the protocol used doesn't significantly impact how easy it is to port applications. Swapping out SQL network drivers is rather straightforward in nearly every language. What makes it hard to move from one database to another is the dialect of SQL in use. CockroachDB's dialect is based on PostgreSQL as well.

## What is CockroachDB's security model?

You can run a secure or insecure CockroachDB cluster. When secure, client/node and inter-node communication is encrypted, and SSL certificates authenticate the identity of both clients and nodes. When insecure, there's no encryption or authentication.

Also, CockroachDB supports common SQL privileges on databases and tables. The `root` user has privileges for all databases, while unique users can be granted privileges for specific statements at the database and table-levels.

For more details, see our documentation on privileges (privileges.html) and the `GRANT` (grant.html) statement.

## How does CockroachDB compare to MySQL or PostgreSQL?

While all of these databases support SQL syntax, CockroachDB is the only one that scales easily (without the manual complexity of sharding), rebalances and repairs itself automatically, and distributes transactions seamlessly across your cluster.

For more insight, see CockroachDB in Comparison (cockroachdb-in-comparison.html).

## How does CockroachDB compare to Cassandra, HBase, MongoDB, or Riak?

While all of these are distributed databases, only CockroachDB supports distributed transactions and provides strong consistency. Also, these other databases provide custom APIs, whereas CockroachDB offers standard SQL with extensions.

For more insight, see CockroachDB in Comparison (cockroachdb-in-comparison.html).

## Can a MySQL or PostgreSQL application be migrated to CockroachDB?

The current version of CockroachDB is intended for use with new applications. The initial subset of SQL we support is small relative to the extensive standard, and every popular database implements its own set of extensions and exhibits a unique set of idiosyncrasies. This makes porting an existing application non-trivial unless it is only a very lightweight consumer of SQL functionality.

## Does Cockroach Labs offer a cloud database as a service?

Not yet, but this is on our long-term roadmap.

## Can I use CockroachDB as a key-value store?

CockroachDB is a distributed SQL database built on a transactional and strongly-consistent key-value store. Although it is not possible to access the key-value store directly, you can mirror direct access using a "simple" table of two columns, with one set as the primary key:

## Keep up-to-date with CockroachDB software releases and usage best practices

```
> CREATE TABLE kv (k INT PRIMARY KEY, v BYTES);
```

Email **SUBMIT**

When such a "simple" table has no indexes or foreign keys, `INSERT` (insert.html)/`UPSERT` (upsert.html)/`UPDATE` (update.html)/`DELETE` (delete.html) statements translate to key-value operations with minimal overhead (single digit percent slowdowns). For example, the following `UPSERT` to add or replace a row in the table would translate into a single key-value Put operation:

```
> UPSERT INTO kv VALUES (1, b'hello')
```

This SQL table approach also offers you a well-defined query language, a known transaction model, and the flexibility to add more columns to the table if the need arises.

## Have questions that weren't answered?

- CockroachDB Community Forum (https://forum.cockroachlabs.com): Ask questions, find answers, and help other users.
- Join us on Gitter (https://gitter.im/cockroachdb/cockroach): This is the most immediate way to connect with CockroachDB engineers. To open Gitter without leaving these docs, click **Chat with Developers** in the lower-right corner of any page.
- SQL FAQs (sql-faqs.html): Get answers to frequently asked questions about CockroachDB SQL.
- Operational FAQS (operational-faqs.html): Get answers to frequently asked questions about operating CockroachDB.

Was this page helpful? YES NO

## Keep up-to-date with CockroachDB software releases and usage best practices

Email SUBMIT

**PRODUCTS (HTTPS://WWW.COCKROACHLABS.COM/PRODUCT/COCKROACHDB-CORE)**

CockroachDB Core (https://www.cockroachlabs.com/product/cockroachdb-core/)

CockroachDB Enterprise (https://www.cockroachlabs.com/product/cockroachdb-enterprise/)

FAQ (https://www.cockroachlabs.com/docs/stable/frequently-asked-questions.html)

Pricing (https://www.cockroachlabs.com/pricing/)

Comparison (https://www.cockroachlabs.com/docs/stable/cockroachdb-in-comparison.html)

**RESOURCES (HTTPS://WWW.COCKROACHLABS.COM/COMMUNITY/)**

Events (https://www.cockroachlabs.com/community/events/)

Tech Talks (https://www.cockroachlabs.com/community/tech-talks)

Forum (https://forum.cockroachlabs.com/)

**COMPANY (HTTPS://WWW.COCKROACHLABS.COM/ABOUT/)**

Blog (https://www.cockroachlabs.com/blog/)

About (https://www.cockroachlabs.com/about/)

Careers (https://www.cockroachlabs.com/careers/)

Press (https://www.cockroachlabs.com/press/)

**DOCUMENTATION (HTTPS://WWW.COCKROACHLABS.COM/DOCS/STABLE)**

Get Started (https://www.cockroachlabs.com/docs/stable/start-a-local-cluster.html)

Build an App (https://www.cockroachlabs.com/docs/stable/build-an-app-with-cockroachdb.html)

Develop (https://www.cockroachlabs.com/docs/stable/install-client-drivers.html)

Deploy (https://www.cockroachlabs.com/docs/stable/clouddeployment.html)

Manage (https://www.cockroachlabs.com/docs/stable/monitor-cockroachdb-with-prometheus.html)

## Keep up-to-date with CockroachDB software releases and usage best practices

Email **SUBMIT**

**CONNECT WITH US (HTTPS://WWW.COCKROACHLABS.COM/CONTACT/)**

Contact Us (https://www.cockroachlabs.com/contact/)

Github (https://github.com/cockroachdb/cockroach/)

Forum (https://forum.cockroachlabs.com/)

Gitter (https://gitter.im/cockroachdb/cockroach)

Twitter (https://twitter.com/cockroachdb)