DZone A DEVADA MEDIA PROPERTY

BROUGHT TO YOU IN PARTNERSHIP WITH Cockroach Labs

Multi-Region Database Deployments

Patterns and Anti-Patterns for Database Deployments Across Multiple Regions

JIM WALKER
VP OF PRODUCT MARKETING, COCKROACH LABS

CONTENTS

INTRODUCTION

Deployed in the cloud, our modern applications and services demand an always-on, low latency experience for users no matter where they are on the planet. Whether you're building a startup from the ground up or you are a member of a massive Fortune 500 development organization, these demands are typically the same.To meet these requirements, many deploy multiple instances of their applications and services across multiple cloud regions. Some will deploy a database and synchronize it across multiple regions so that they can survive a regional outage as well. It's not enough. This legacy approach leaves room for downtime and even worse, inconsistencies in data.

Deployment of an active-active database with multi-region capabilities that can be applied down to the table and row level of your data will allow you to not only survive a region failure without downtime, but also ensure consistent and low latency access to data no matter where you do business.

WHAT IS A MULTI-REGION APPLICATION?

A multi-region application is deployed across multiple cloud availability zones or regions. This deployment pattern is designed to meet the expectations of a distributed audience while also meeting high availability requirements. Given the nature of this setup, we also consider this application to be "global," meaning that it may serve audiences across different geographic zones.

We typically deploy our applications across multiple regions so that we have redundancies and the ability to survive a regional outage. Regional outages may seem unlikely, but they actually occur quite often. Usually, an active-passive setup is used for databases, but that deployment style is risky. A new generation of database technologies have emerged that allow for more robust active-active redundancies while also offering the geographical performance improvements required of a truly multi-region application.

Using a multi-region database delivers several benefits for building distributed applications. Typically, organizations turn to multi-region databases to solve problems such as:

- **Minimizing transactional latency** by placing data physically close to end users
- **Eliminating outages** with redundancies that can survive entire region or cloud failures
- **Aiding with data privacy compliance** by storing data within local boundaries
- **Delivering a multi/hybrid-cloud deployment** to eliminate risk and costs of an "all-in" single cloud approach

CockroachDB

The most highly evolved database on the planet.

CockroachDB makes it easy for every developer to create apps that scale fast, survive anything,

Avoid complexity with automated scale

Scale up or down effortlessly without sharding or manual operations.

Eliminate outages with bulletproof resilience

Operate worry-free and guarantee access to data without interruption.

Say goodbye to transactional latency

Deliver low latency reads and writes to anyone, anywhere by locking data to a location.

Get started for free today

cockroachlabs.com

Trusted by innovators

DOORDASH COMCAST SPACEX LUSH ebay MYTHICAL

While these benefits address several challenges that global organizations typically face, there's even more value to embracing a distributed cloud database system than meets the eye.

WHAT DO I NEED TO DEPLOY A MULTI-REGION DATABASE?

First, you should already have a cloud strategy. That might be all in on a single public cloud provider, a mix of several, or perhaps a hybrid, multi cloud deployment. Regardless, you should be familiar with cloud-based consumption and deploying your apps as a service. Second, you need a database that was built to deliver on multi-region requirements. Unfortunately, most of the cloud databases offered by the public cloud providers present significant challenges for multi-region deployments and struggle to deliver on the stringent latency and resilience requirements of modern consumers. So you may have to look elsewhere.

You should look for a database that can deliver highly available and low latency access to consistent, reliable data at efficient scale. While many traditional databases can meet some of these requirements, they may struggle when deployed across multiple regions.

WHAT IS THE VALUE FOR DEVOPS TEAMS?

Deploying an application across multiple regions is not a simple task, and it is worthwhile to consider the impact it might have on your development and operations teams.

Having five nines (99.999%) high availability and no downtime will contribute to the health and happiness of your teams, and will keep your operations people sharp and attentive to optimizing. Using a multi-region database with active-active capabilities will automate distribution of data across various failure domains, eliminating database downtime due to regional outages because the data is always accessible somewhere.

Additionally, a multi-region database will help you minimize transactional latency by anchoring data physically closer to whatever region your users might be in. This means that developer teams can spend less time worrying about the performance of the application and more time building new features.

HOW CAN THIS HELP ACHIEVE COMPANY GOALS?

A multi-region database will help you with three key top level business goals:

1. **Business growth:** Increase your user base in different countries more efficiently where you may have never conducted business before. If you deploy your database across multiple clouds in multiple regions and have the ability to keep data close to users, you can guarantee a great end-user experience by ensuring data is nearby.
2. **Avoid regulatory fines:** As organizations grow globally, they encounter a complex web of current and emerging local data regulation and compliance laws. The ability to tie data to a particular region can help meet some of these regulations and minimize your risk associated with them.
3. **Eliminate operational overhead and costs:** Using a single logical database across multiple regions will not only reduce the costs associated with asynchronous disconnect active/passive systems, but it will also allow you to do more with less as it takes fewer people to manage a single multi-region database.

COMMON PATTERNS AND ANTI-PATTERNS

There are several considerations to take into account when building a multi-region application. Most importantly, we will cover: availability model, databases, and physical deployment/topology patterns. Once you have the right strategy and tools in place, you can successfully adopt a multi-region database.

PATTERN 1: PICK THE BEST AVAILABILITY MODEL

An active-active availability model enables the database to scale beyond single machines by letting nodes in a cluster serve reads and writes. With this kind of replication, you set up at least two active sites — each of which must contain all of the cluster's data.

Client reads and writes from the node in one of the active sites will generate any modifications to other active sites. For many workloads, active-active replication is difficult and expensive to instrument in a database. For that reason, you should consider the type of database you are using to achieve this replication model (more on this below).

AVAILABILITY MODEL ANTI-PATTERNS

By default, active-passive replication allows one node to receive all the requests and replicate data to a single follower. There are two ways to set this up. Synchronous replication requires that each write to the active node propagates to the passive node. However, if the passive node can't be reached because it's down or there's a network partition, the active node can't progress.

Asynchronous replication lets the active node send data to the passive node without guaranteeing that the two nodes' states mirror one another. When the active node fails, any data that hasn't been committed to the passive node is lost forever.

While an active-passive configuration is OK for some use cases, it will struggle to deliver low-latency, consistent transactions and high availability for writes. Simply put, the active-passive availability model is not fit for modern applications.

PATTERN 2: SELECT THE RIGHT DATABASE FOR MULTI-REGION APPLICATIONS

Ultimately, multi-region applications should be backed by a truly distributed database which is purpose-built to store data and deliver efficient reads and writes across different physical locations. Even though an active-active replication model is a critical capability, that alone is not enough. Not all distributed databases were created equal.

Distributed databases should deliver and ease scaling, but they must also provide consistent transactions at scale. Additionally, they should be resilient and have the capability to replicate data across different geographical locations. They should also speak SQL since it is the language of data and will be familiar to virtually every developer.

However, there are two critical requirements for a distributed database that will help you get the most out of a multi-region application.

The first is data locality. The database should understand the location of participants (in various regions or data centers) and be able to store data that is relevant to them close to their location.

The second is multi-cloud capability. Support for more than one public cloud provider mitigates risk and eliminates the need to rely on a single network to accomplish distribution. It also prevents vendor lock-in and gives you more regions to select from.

DATABASE FOR MULTI-REGION ANTI-PATTERNS

A traditional database often lacks the capabilities required to shine in a multi-region deployment. These database systems were typically architected for vertical scale. Horizontal, cross-region capabilities are secondary add-ons or modifications. In order to get any sense of horizontal scale, we often have to depend on an error-prone, costly, and operationally complex manual sharding approach.

Further, high availability and disaster recovery for a legacy database will often employ an active-passive topology that still leaves room for downtime and may cause inaccuracies in data.

NoSQL databases such as Cassandra and MongoDB deliver an active-active architecture. However, when you want to scale transactions from a distributed user base using a NoSQL database, they sacrifice consistency, which means you also sacrifice a consistent user experience. Further, the challenge of managing objects within the document store framework can be an issue as business complexity rises.

PATTERN #3: CHOOSE A PHYSICAL DEPLOYMENT METHOD OR TOPOLOGY

When using a distributed database, it is important to extend your focus from the simple logical model of the data and start to consider the physical model, or where data will live. In this section, we will review some physical topology concerns for a multi-region database.

Topology patterns help map the physical shape of your cluster. The placement of data throughout the distributed nodes will allow you to optimize your database for surviving failures (server, rack, region) or for delivering low latency access to users by tying data to nodes that are physically close to the requestor. The physical model of the database is a critical concern.

Using these patterns will force you to ask: Are you deployed in a single region across multiple availability zones? Are you deployed across multiple regions? And where is the data located across that cluster, across those nodes?

In order to achieve a multi-region cluster, your application and data layer will be distributed across multiple machines in multiple regions. There are a few things you need to carefully choose:

- **Cluster regions:** geographic regions that a user specifies at node start time.
- **Database regions:** geographic regions that a given database operates within. A database region must be chosen from the available cluster regions. Each region is broken into multiple zones. (These terms correspond directly to the region and zone terminology used by cloud providers.)
- **Table locality:** a setting to determine how to optimize access to the table's data from that locality.
- **Survivability goals:** rules that dictate how many failures a database can survive.

Once you pick the pattern that is the best fit for your use case, you can begin to set up your multi-region cluster.

PHYSICAL DEPLOYMENT OR TOPOLOGY ANTI-PATTERNS

With single-region clusters, you do not have to spread data across multiple geographic locations, so latency isn't an issue. However, single region-clusters are not fit for building multi-region applications since they physically limit the distribution of data.

With a multi-region cluster, you have a cluster spread across multiple geographic locations and the distribution of data can also become a key performance bottleneck. Latency can skyrocket if it's not configured correctly.

For that reason, it's important to think about the latency requirements of each table and then use the appropriate data topologies to locate data for optimal performance.

PATTERN #4: ADOPT A MULTI-REGION DATABASE

At a high level, the simplest process for running a multi-region cluster is the following:

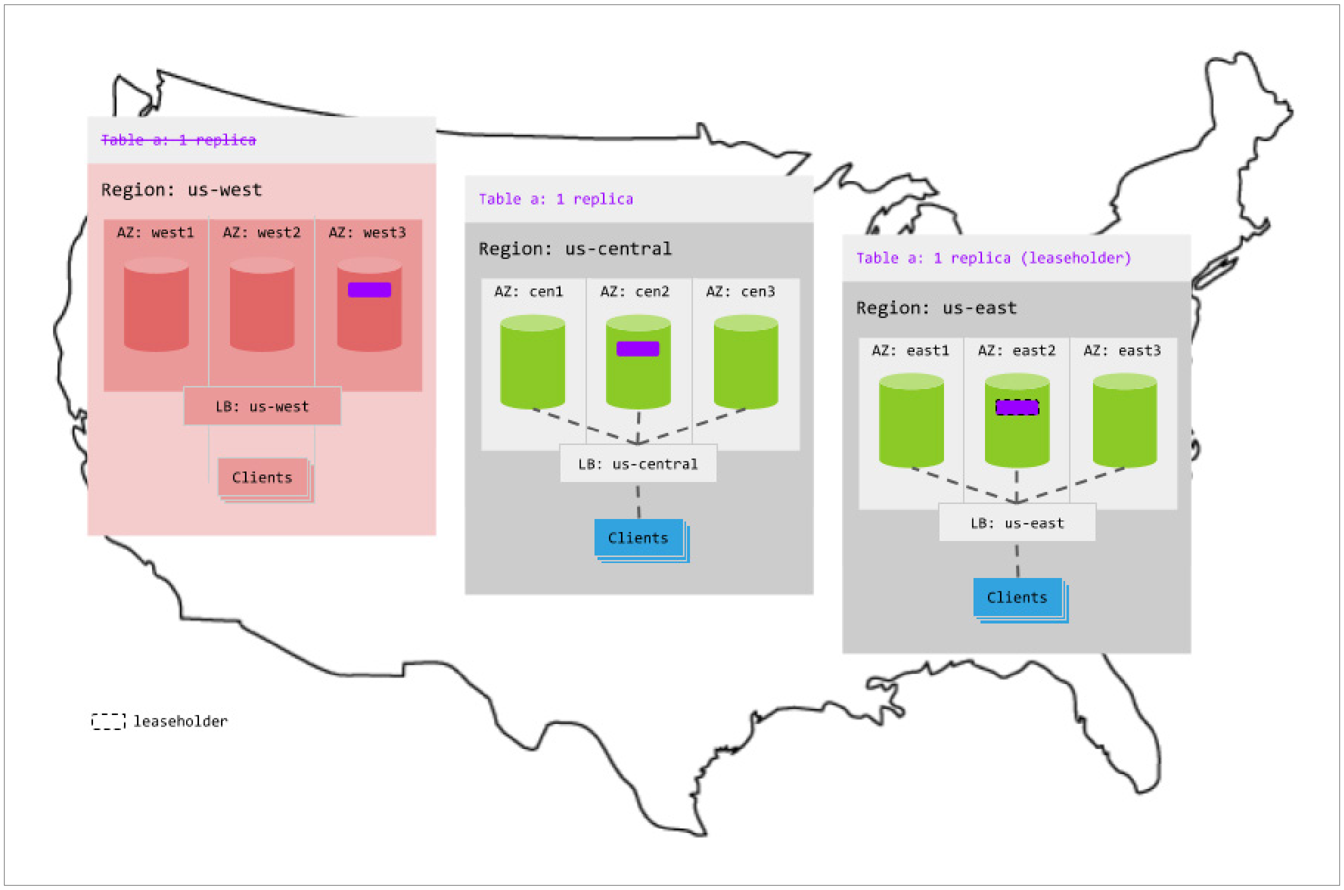
1. Set region information for each node in the cluster at startup using node startup locality options.
2. Add one or more regions to a database, making it a "multi-region" database. One of these regions must be the primary region.
3. (Optional) Change table localities (global, regional by table, regional by row). This step is optional because, by default, the tables in a database will be located in the database's primary region (as set during Step 1).
4. (Optional) Change the database's survival goals (zone or region). This step is optional; by default, multi-region databases will be configured to survive zone failures.

These steps describe the simplest case, where you accept all of the default settings. We've included a diagram below that showcases how this setup will allow you to survive a region failure. In this instance, there are nine nodes, all working together to deliver a single logical database and rows of data for the tables are spread across the three regions. If a region goes down, then there are still copies of the data available in other regions and the database can continue to function.

Surviving zone failures is the default setting for active-active multi-region databases, and this is sufficiently resilient for many multi-region applications. However, if your application has performance or availability needs that are different than what the default settings provide, you should explore the other customization options for that database.

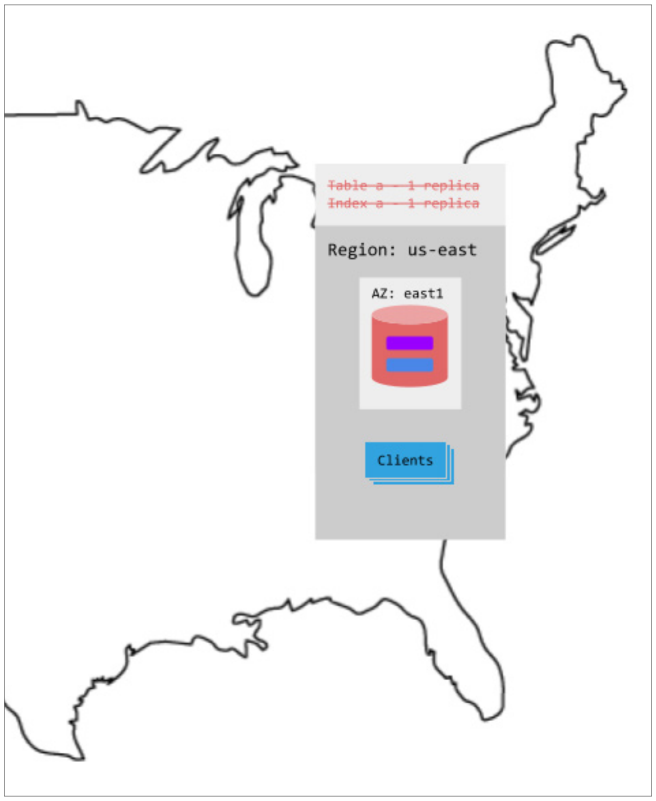
For example, imagine a financial services company needs a payment transfer system that is easily accessed by a global customer base (even when they are traveling). A multi-region database will automatically move data close to users when they are accessing the application, regardless of their location. This will reduce latency and allow the company to adhere to data regulations.

It's crucial for organizations in highly regulated industries like financial services to protect their customers' private data and ensure high availability at all times.



MULTI-REGION DATABASE ANTI-PATTERNS

As we mentioned above, using an active-passive architecture is the wrong approach for a multi-region database. But there's a few other patterns that are also ineffective or risky. The first is a single-region deployment using two availability zones (AZs), or multi-region deployments using two regions. In these situations, the cluster could not survive the loss of a single AZ or a region, respectively (see image below). Single-region deployments are also not suitable when you are catering to users far from the application's deployment region (causes latency, issues with data integrity).



The second anti-pattern is a broadly distributed multi-region deployment (e.g., US-West, Asia, and Europe) using the Follow-the-Workload default pattern. In this case, latency can be unacceptably high. When a table is active in multiple regions at the same time and you have concurrent reads and writes, latency will quickly skyrocket.

For example, let's say there's an online platform in the EU that is expanding to the US. The platform is popular during overlapping hours, which will cause latency for US users communicating with the EU datacenter. Instead, the company should use a multi-region database and add a region in the US. This way, US users can have low latency access, and their data will be in the same region as they are, which is often necessary for regulatory compliance.

FINAL THOUGHTS

This Refcard is designed to give you an overview of the benefits of building a multi-region application. As mentioned before, setting up a multi-region deployment is no easy feat. If you are a global organization catering to a global audience, you should take this methodology into consideration.

WRITTEN BY JIM WALKER,
VP OF PRODUCT MARKETING, COCKROACH LABS

Jim is a recovering developer turned product marketer and has spent his career in emerging tech and open source. He believes product marketing is a core strategic function for early-stage companies and helps them translate complex concepts into a compelling, effective core narrative and market strategy. He is an advocate of the developer and an active participant in several open source communities.