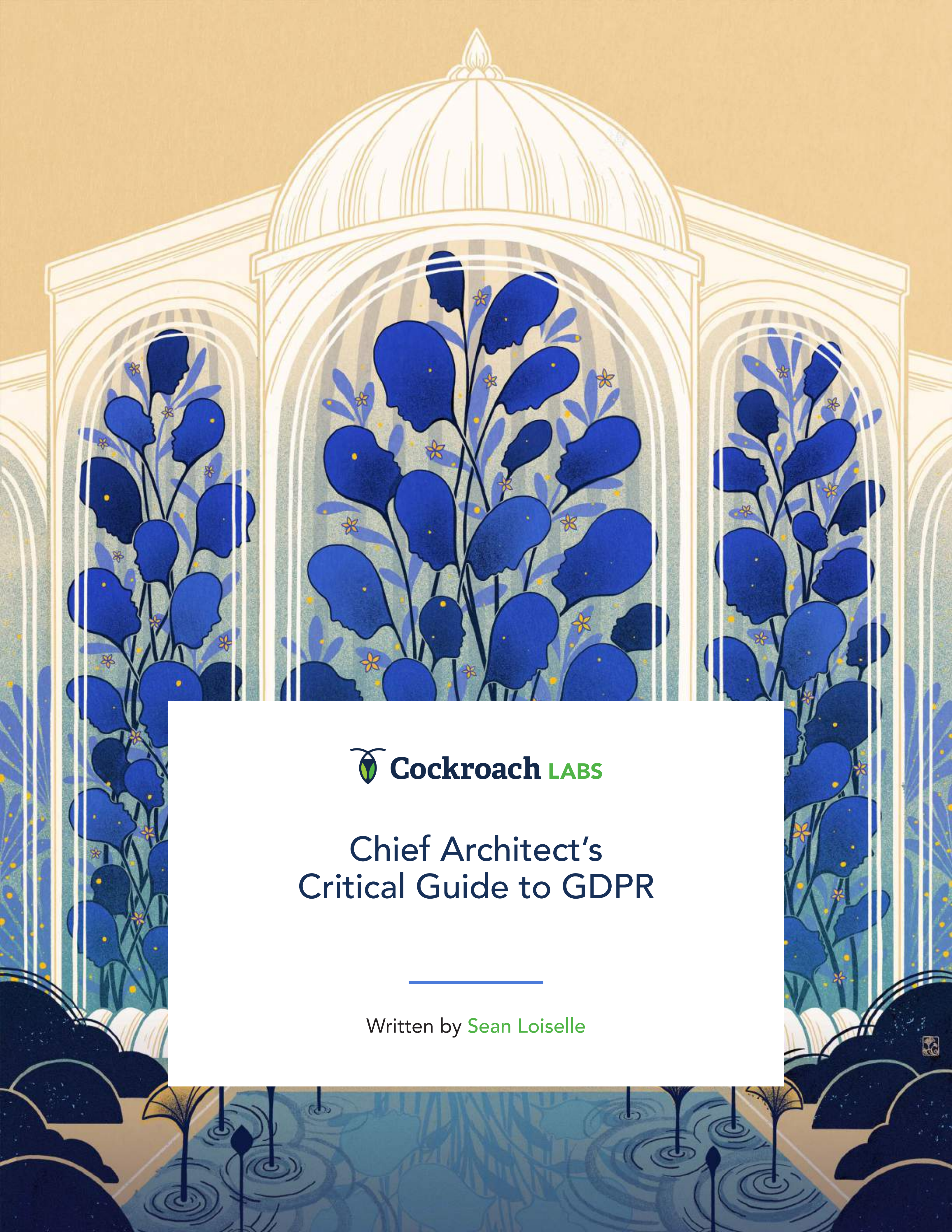


Cockroach LABS

Chief Architect's Critical Guide to GDPR

Written by Sean Loiselle

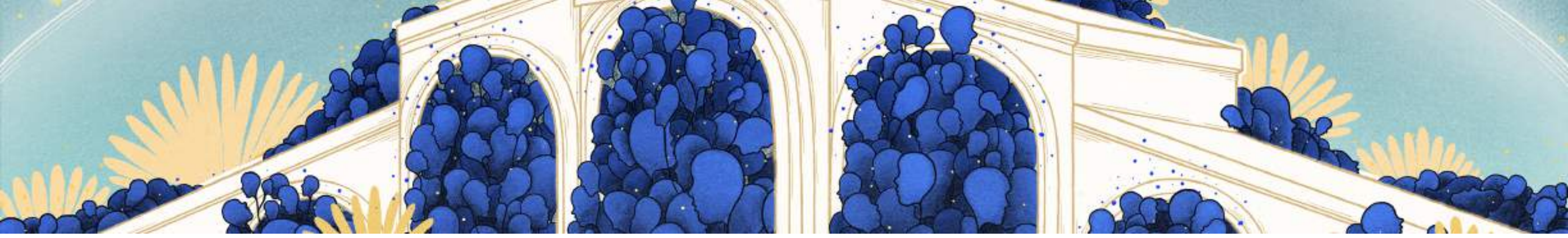
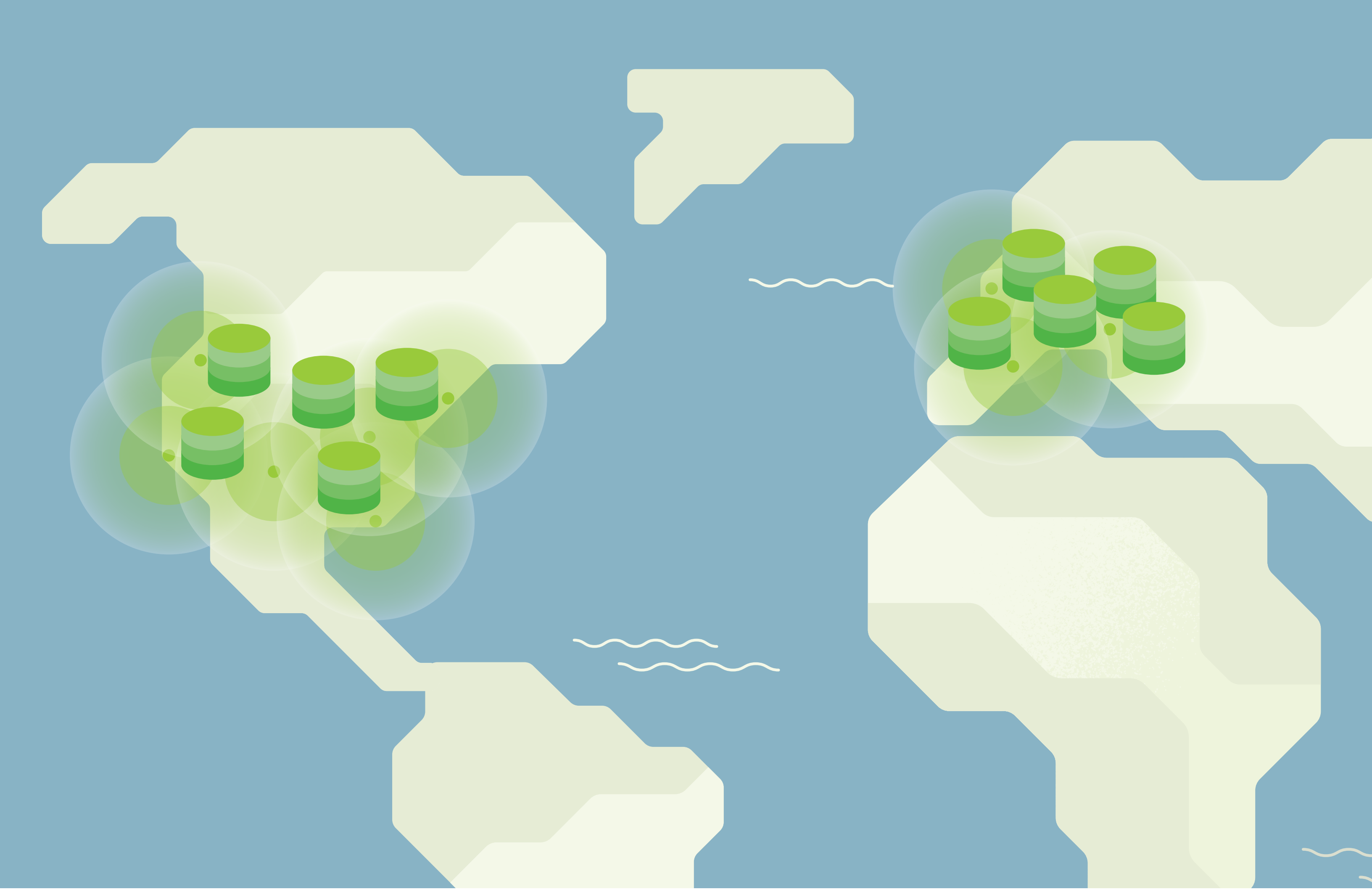


Table of Contents



Introduction

General Data Protection Regulation means a lot of things for your business—one of the most important things is answering the question, "How can my database comply with GDPR?"

Growth is exciting, but it always brings new challenges. For businesses—especially those in North America—after you grow to a certain point, it's very tempting to look at the EU as a great next opportunity.

However, moving into the EU is no longer simple because of the General Data Protection Regulation (GDPR). In the past, you just needed to stand up servers close to those new users and that was it. With GDPR in place, though, there are now a number of new, complex considerations you need to make to make sure you comply.

What You'll Learn

To make cogent choices that take the complexity of GDPR compliance into account, this guide answers some crucial questions:

- What is GDPR?
- What rights does GDPR provide my users?
- How can my database comply with GDPR?
- How can I set up CockroachDB to comply with GDPR?
- How can the rest of my application support GDPR compliance?

GDPR Overview

What Is GDPR?

Just to make sure we're all on the same page, it's important to understand what the GDPR is and how it affects your business' software infrastructure—particularly as it relates to your database.

GDPR is a regulation that took effect on May 25, 2018 to improve business practices in handling user data: both in the security they provide, and the transparency with which they communicate how the data is used.

To boil it down to its simplest components, GDPR is focused on protecting user data. For businesses (and their Ops teams), that means receiving explicit consent from EU users before storing or even **processing** their data outside the EU. If the user declines, their data must only reside within the EU. If you're caught failing to comply with GDPR, you'll face fines of either 4% of annual global turnover or €20 Million, whichever is greater.

GDPR Jargon

When dealing with GDPR, there are a few key pieces of terminology that are helpful to understand:

- **Data Subjects** are users.
- **Controllers** are businesses or other entities that have Data Subjects.
- **Processors** are entities who process data, which includes those who provide software to controllers (e.g. your database or any services you use to process data).

GDPR Guidelines aka Data Subject Rights

The substance of GDPR is a number of regulations (known as Data Subject Rights) that you must meet to comply with the regulation. These regulations, when taken in total, attempt to ensure that businesses treat EU citizens' data with the utmost care and are totally transparent about what they do with it.

WARNING: These are Cockroach Labs' interpretations of GDPR and do not constitute legal advice. Before taking any action with this information, consult your own lawyers.

- **Data Location & Consent:** To make sure that data stays within the EU, controllers must acquire explicit consent from users to transfer user's personal data outside the EU.
- **Right to access:** On request, controllers must provide data subjects confirmation as to whether or not they are processing any of the subject's data, where and for what purpose. The controller must also freely provide a copy of any personal data in an electronic format.
- **Data portability:** Data subjects can transmit personal data given to them by a controller to another controller.
- **Right to rectification:** Data subjects can correct any erroneous personal data controllers store.
- **Right to be forgotten:** Data subjects can have controllers erase all of their personal data, cease distributing it, and potentially have any processors from continuing to use it. This not only applies to an explicit request from the subject, but also when the data is no longer relevant to the original purposes of processing.
- **Privacy by design:** Controllers and processors must design their services considering the "state of the art" to protect their customer's data. Additionally, controllers and processors must notify users of any data compromises.

GDPR & Your Database

With an understanding of what GDPR is, let's examine what each Data Subject Right means for your database, as well as potential strategies for complying with each.

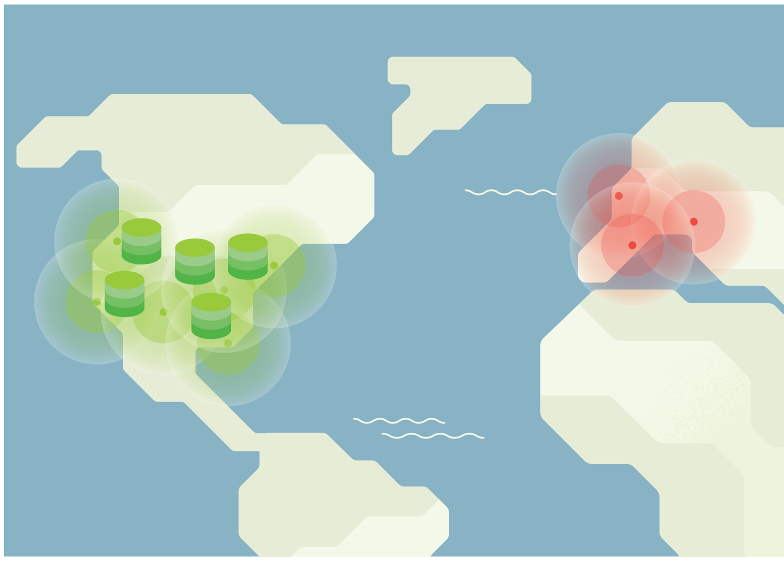
Data Location & Consent

Overview

To make sure that data stays within the EU, controllers must acquire explicit consent from users to transfer their personal data outside the EU.

What It Means for Your Database

Because this guide focuses primarily on scaling an application to meet GDPR compliance, we assume you're in the most difficult version of this scenario: you currently have infrastructure outside of the EU, but want to serve customers within it.



Scenario: You currently have infrastructure outside of the EU, but want to serve customers within it.

>> DID YOU KNOW

Data Location & Consent does not prohibit sending user data outside the EU. Instead, it requires users be alerted as to which data is sent and how it will be used.

One simple misgiving with this, though, is easy to dispel: this right doesn't mean you can't send EU user data outside of the EU. Instead, it means you must let users know what data you're sending and how it's to be used.

What this means for your business depends on the strategy you adopt to comply with GDPR. User notifications could range from minimal (e.g. you're only temporarily reading data in the US for business analytics), to very upfront (e.g. primary copies of their data will be stored and read outside the EU).

The problem with the latter statement is not a technological one, but one of sentiment. For a lot of users who are understandably leery of businesses sharing and storing their data, this puts your business at a disadvantage against competitors who are more privacy-focused.

Compliance Strategies

There are two extremes you can use to deal with data portability with many options between them.

- Much like many EU websites simply notify users that the site leverages cookies, one tactic is to notify users that their data might be processed outside of the EU—continuing to use the service is explicit consent of this situation.

  Of course, this runs the risk of alienating users who are concerned about the implications of their data leaving the EU at all. For a small-to-medium-sized startup that is eager to attract as many customers as possible, this can be a difficult position to adopt.

- Create a data architecture that stores and accesses EU data only within the EU. This is a technical hurdle to clear, but is likely to receive the most positive sentiment from your users. This kind of architecture can also provide improved performance by only storing data close to users (similar to a CDN).

  For B2B SasS platforms in particular, this choice is far superior. If you are a processor of other companies' data, they must receive consent from their users to process data outside of the EU—even if they themselves have a totally domiciled data architecture. That's a hard sell and is likely to increase both deflection and attrition on your services.

Like we mentioned, there is a gradient between these two choices. Because of the rash of privacy issues that have continued to arise over the last year, many companies are eager to give their customers the strong data protection they're demanding and are opting for something closer to the second choice.

There are many paths to domiciling data, but one of the most powerful and simple to use is CockroachDB's enterprise geo-partitioning feature, which we'll cover in much greater depth below.

Right To Access

Overview

On request, controllers must provide data subjects with confirmation as to whether or not they are processing any of the subject's data, where, and for what purpose. The controller must also freely provide a copy of any personal data in an electronic format.

What It Means for Your Database

To comply with this right, you must find a way to aggregate all data that contains a user's personally identifiable information (PII). For a database whose responsibility is finding data, this isn't too onerous a requirement.

Compliance Strategies

How you comply with this largely depends on the type of database you're using:

In a relational database, you simply need to be very thorough in creating foreign key relationships that reference your user table—with that in place, it's simple to write a single transaction that aggregates all of a user's data.

In a document-oriented database, you're more likely to have a single document with all of a user's data that you'll need to send them. However, if you keep analytic-type documents elsewhere, you'll need to find a references to an individual user in them.

Data Portability

Overview

Data subjects can transmit personal data given to them by a controller to another controller.

What It Means for Your Database

From your database's perspective, this is essentially a similar feature to Right to access. You simply must be able to provide the user the data that you have about them, and they must be able to move that data to another company. That doesn't mean you have to export/import for them, but you must make it an achievable feat.

Right To Rectification

Overview

Data subjects can correct any erroneous personal data controllers store.

What It Means for Your Database

Right to rectification is also similar to **Right to access**. It requires you to be able to find all of a user's data, but instead of simply writing it to a file, you must be able to perform updates on it.

Right To Be Forgotten

Overview

Data subjects can have controllers erase all of their personal data, cease distributing it, and potentially stop any processors from continuing to use it. This not only applies to an explicit request from the subject, but also when the data is no longer relevant to the original purposes of processing.

What It Means for Your Database

Much like **Right to access** requires you to be able to read all of a user's PII, the right to be forgotten means you need to be able to delete it.

Importantly, though, this doesn't mean that you need to completely remove all data the user generated—for instance, if you created a GUID for that user, analytics tables that include the GUID do not need to be deleted; there just shouldn't be any way to correlate that data to any of the user's PII.

In addition, you also need to consider how your database handles backups. Because it's infeasible for most database engines to delete individual

>> DID YOU KNOW

A data architecture that stores and accesses EU data within the EU delivers improved performance to users because their data is stored nearby, similar to a CDN.

values from their backups, you'll need a process to ensure that any backups that are used also remove any data from users who've requested to be forgotten.

Compliance Strategies

The compliance strategy for the **Right to be forgotten** is more involved than the **Right to access**, because you must provide users details about exactly which steps you are taking to remove their data and any contingencies they should be aware of.

Here are some examples of what you may need to notify customers about:

- Primary data sources in production systems will be deleted immediately.
- Data stored in backups must be retained for a longer period of time, but will not be restored back into production systems, except in rare cases such as a full system restore from a point in time before the user exercised their right to be forgotten. In this case, the controller will take all available steps to honor the initial request and delete the primary data sources.
- All data will be retained for as short a time as necessary before being automatically deleted.

To achieve the goal of removing users who've requested to be forgotten from any backups, you might consider storing that data within the database, but backing it up at much more regular intervals or in different locations to ensure it's available even after an event that necessitates disaster recovery.

Privacy By design

Overview

Controllers and processors must design their services considering the "state of the art" to protect their customers' data. Additionally, controllers and processors must notify users of any data compromises.

What It Means for Your Database

This right has broad, sweeping implications that touch many facets of your database, but we'll cover those that we're aware of being most impactful:

- Connections to your database must be encrypted via SSL.
- You should limit access to your data in ways that minimizes the number of services accessing them.
- If possible, the data and its backups should be stored using encryption-at-rest.Note that in the regulation itself refers to encryption as "pseudonymization", but they're the same thing.

Compliance Strategies

Privacy by design has both required and negotiable tenants; we'll cover strategies for both.

- Non-negotiably, your services must leverage "pseudonymisation," which is legalese for encryption. This means that data sent to and from your services, because it contains sensitive information, must be protected with SSL encryption.

  Fortunately, every contemporary database that you would choose to put into production provides support for SSL certificates, so this is simple to achieve.
- Negotiably, you can consider many other angles of controlling access to your user's data.
  - Role-based access controls to minimize the number of users who can access data to only those who justifiably need it.
  - Encryption at rest, which means that backups are generated in an encrypted format and require a key to decrypt, making sure that compromised data is more difficult for hackers to access.

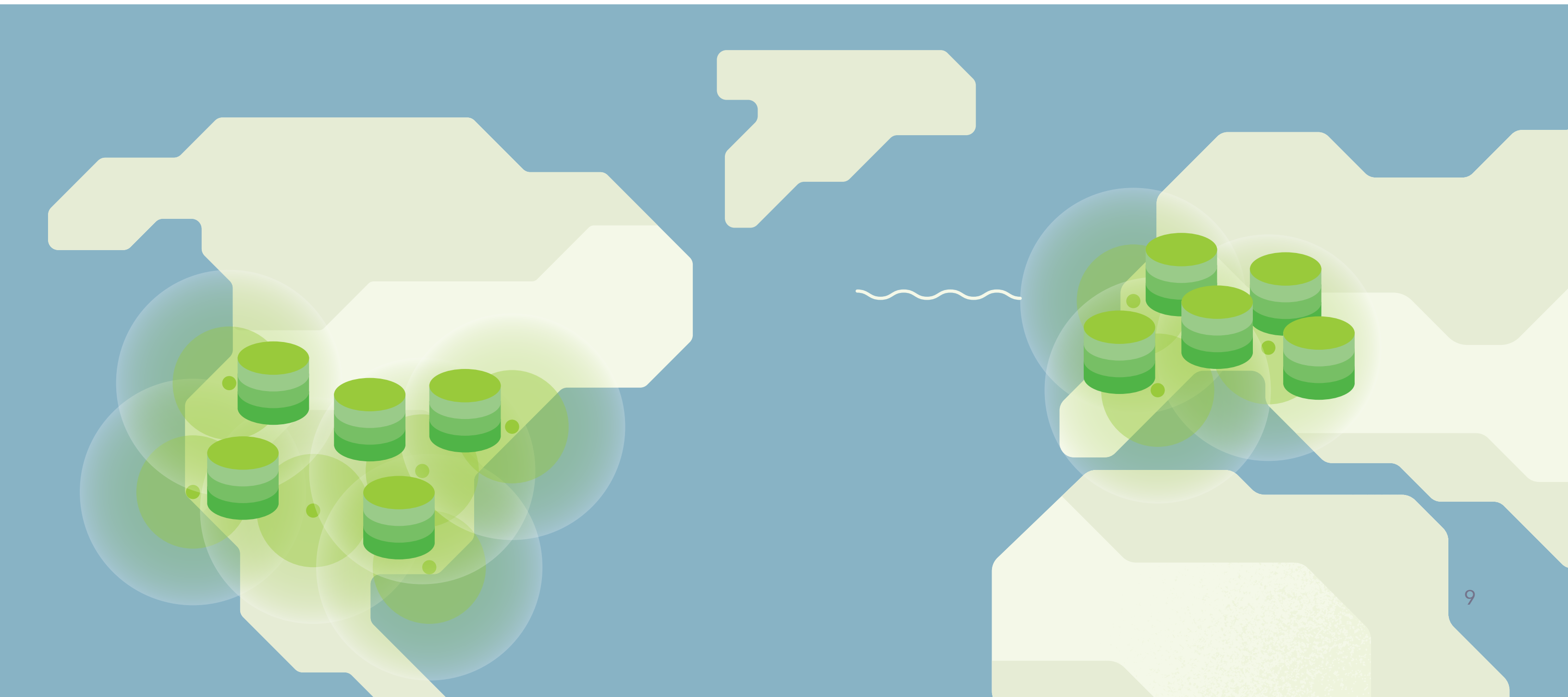
How CockroachDB Can Support GDPR Compliance

Because of CockroachDB's cloud native design, it offers a number of features that simplify GDPR compliance.

Of course, a database alone cannot make your team entirely GDPR compliant, but the rest of this guide will show you how to create a strong foundation which you can build the rest of your GDPR stack on top of.

To give you a sense of what these features are and how they would work in practice, the rest of this guide is going to use an fictitious startup, Movr.

One GDPR compliance strategy is a data architecture that stores and accesses EU data only within the EU.



Movr

Movr is SaaS platform that offers vehicle sharing (bicycles, scooters, and skateboards). Users open the Movr app, find a nearby vehicle, book it, ride it, and then release it when they're done.

Movr's Current Architecture

Movr uses a 5-node CockroachDB cluster centered on the east coast of the US to service their user base in NYC. They don't need to worry about serving users anywhere else in the US because their physical assets are only in NYC.

To manage their infrastructure, they rely on configuration management tools (e.g. Chef).

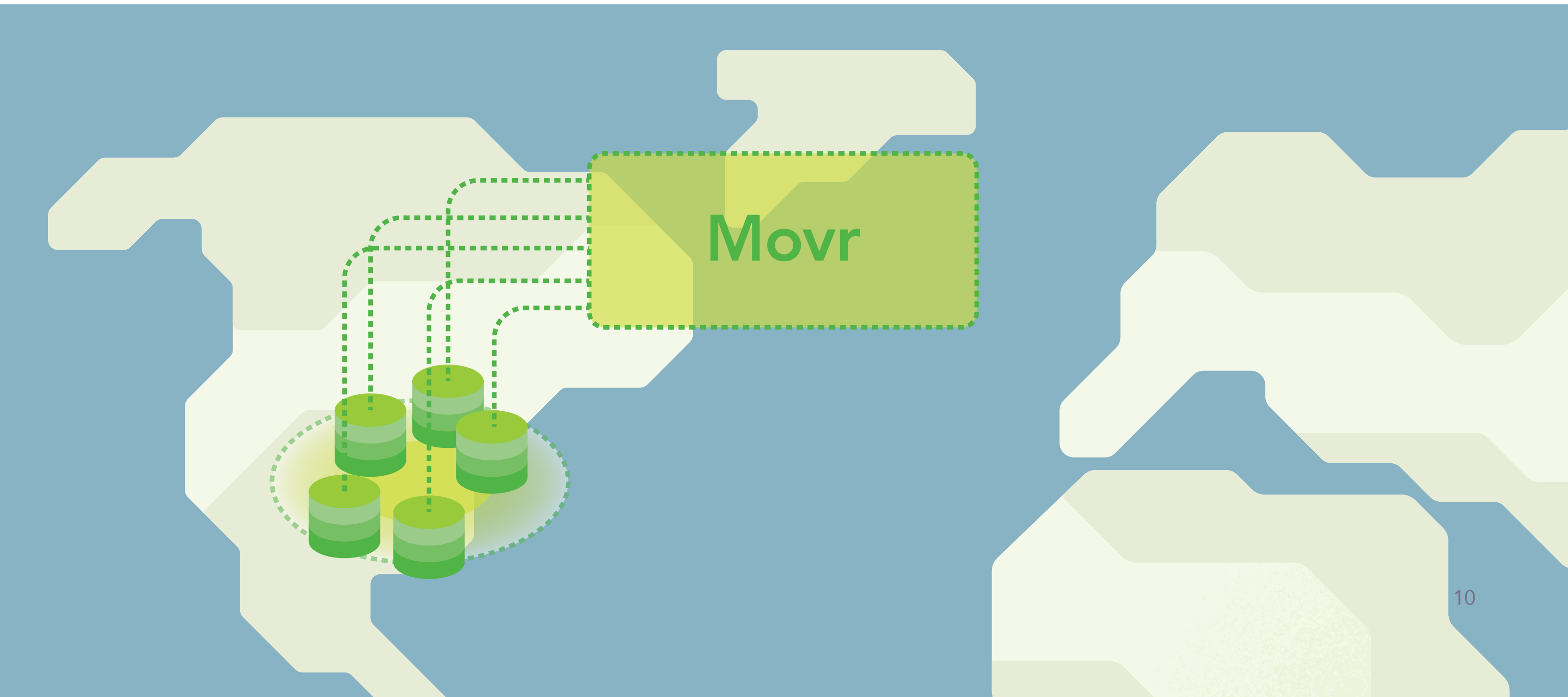
Movr's Data

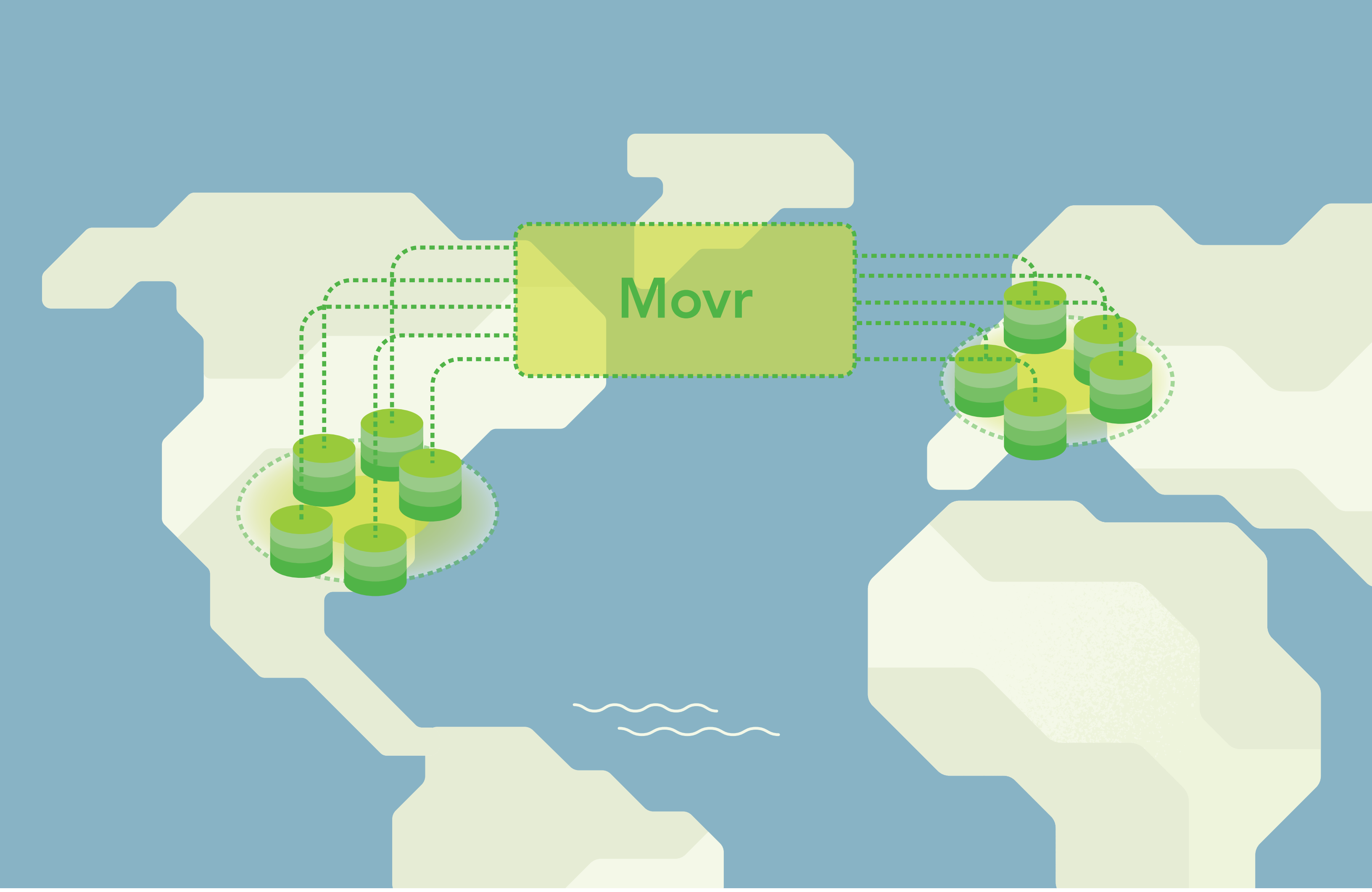
Movr's application is elegant and simple, consisting of only a few tables:

- Users
- Rides
- Vehicles

Thanks to an exorbitant amount of VC funding, they aren't charging users at all for their service, so they don't need to deal with anything thorny like payments.

A 5-node CockroachDB cluster distributed in the eastern US serves MovR's user base in NYC.





MovR expands to Amsterdam by spinning up a 5-node replicated CockroachDB cluster in the EU.

Movr's Plans

Movr's had a lot of success in NYC and wants to expand to the EU. After a wonderful spring trip, their team decides that an expansion into Amsterdam is a great fit.

Because their infrastructure has been such a success, Movr wants to replicate it in the EU—ultimately having another 5-node CockroachDB cluster running nearby.

Note that this setup of a 10-node CockroachDB clusters with a replication factor of 5 can still only handle 2 simultaneous outages without compromising availability.

Movr is in a great situation here with its use of CockroachDB, which has enterprise features that will make GDPR compliance much simpler:

- Geo-partitioning lets Movr easily domicile EU users' data in the EU.
- Role-based access control provides granular control over which users can access which tables.

Compliance Strategies

Now that we understand who Movr is and what they want to accomplish, let's look at the strategies they chose for GDPR compliance. After that, we'll examine their actual implementation in CockroachDB.

Data Location & Consent

Movr plans to leverage CockroachDB's geo-partitioning feature to keep each user's data stored in their region of origin, i.e. New York users in US data centers and Amsterdam users in the EU.

We have more detail about this below, but to use CockroachDB's geo-partitioning feature, each table needs nothing more than the first column in its primary key to identify which partition it belongs to. This does mean Movr will need to rebuild their tables, but that is a one-time cost they're willing to pay for a dramatically simplified future—both in terms of reduced latencies and data domiciling.

For simplicity's sake, Movr plans to keep its full table backups in the EU. If it chose to keep the backups in the US, though, it would require explicit consent from users.

However, because Movr is based in the US and it still wants it business analytics to read data from all users—including those in the EU— they will need to obtain consent from users for that level of access.

Right To Access

To provide users the right to access all of their data, Movr creates an application-level query to aggregate all of a user's individual data from both the user and ride table.

Movr also needs to include detail about what it's doing with that data, which they determine is:

- User data uniquely identifies each user.
- Ride data contains records of the users who took the right to examine demand patterns so Movr can continue to improve their service.

To make this simple, Movr simply needs to include a foreign key in the Rides table that references the user.

Data Portability

Movr complies with this through the work they invest in complying with **Right to access**. If users request a copy of their data, Movr can easily supply it. Once the user has the data in hand, they're free to do with it as they please.

Right To Rectification

Movr freely lets users edit any information about their account, so users can rectify any data Movr stores in the User table.

Right To Be Forgotten

To achieve this, Movr enters NULL (or a "NULL" string) for all of a user's PII data, e.g. their name, email addresses, address, etc.

They also create another table called "forgotten_users", in which they only store the user's ID. This way, if there is ever a disaster recovery restore, they're able to similarly expunge these users' data as quickly as possible. However, it's important that Movr keep a separate backup of this table so it can be retrieved even if there is a catastrophic failure in one datacenter. Because this backup doesn't contain sensitive information, they decide to replicate it to both the EU and US data centers.

Privacy By Design

Movr only uses SSL connections to connect to CockroachDB, so their users' data is protected.

Using CockroachDB's enterprise role feature, they also fastidiously manage groups who can access the cluster's data. For example, their vehicle maintenance team doesn't need access to the User table, so their application's client is placed in a role that can't access it.

For increased security, Movr stores their backups on servers with SSH authentication.

CockroachDB Deployment Strategies

Now, let's look at what Movr needs to do to scale their application to support their GDPR goals.

Schema Design

To begin with, Movr needs to redesign their schemas to support their GDPR initiatives. Here's what their pre-GDPR schema looks like:

```
CREATE TABLE user (
   id UUID,
   username STRING,
   password STRING,
   city STRING,
   name STRING,
   address STRING,
   ountry STRING,
   CONSTRAINT PRIMARY KEY (id),
   INDEX (username)
);

CREATE TABLE vehicle (
   id UUID,
   city STRING,
   country STRING,
   type STRING,
   owner_user_id UUID,
   status STRING,
   CONSTRAINT PRIMARY KEY (id),
   INDEX (owner_user_id),
   CONSTRAINT FOREIGN KEY (owner_id)
REFERENCES users (id)
);
```

```
CREATE TABLE forgotten_users (
    id UUID PRIMARY KEY,
    date_forgotten TIMESTAMP
)
```

Note that Movr doesn't index any user personally identifiable information (PII). This works well for them because doing so can theoretically store user PII in one of CockroachDB's meta ranges, which are used to identify which node contains a given row's data. Similarly, if a PII value ended up in the meta range, it could not be deleted if a user exercised their right to be forgotten.

To mitigate this issue while still making it efficient to look up users, you can simply create usernames, which are not PII, and both index them and leverage them in cookies in your app.

Partitioning Support

Now that the tables have their partitions defined, we can define the partitions on the physical machines through these steps.

1. Start nodes with their locations identified with the—locality flag.
2. Create and apply zone configs.

>> DID YOU KNOW

There are many paths to domiciling data, but one of the most powerful and simple to use is CockroachDB's enterprise geo-partitioning feature.

Node Location

CockroachDB nodes need to be told where they're running, which is done with the locality flag. In Movr's case, they chose to add a region label, which is what they will use to control partitions.

For their US-based nodes, which are all on the east coast, they restart their nodes with the following command:

```
# Start the node in the US datacenter:

$ cockroach start --locality=region=us-
east,datacenter=us-east-1  [... other
flags]
```

Similarly, for their new EU nodes, which are based in the western part of the EU:

```
# Start the node in the EU datacenter:

$ cockroach start --insecure
--locality=region=eu-
west,datacenter=eu-west-1 [... other
flags]
```

Note that the locality flags can have an arbitrary number of levels, but that the levels should use the same hierarchy, such as region > datacenter > availability zone.

Create and Apply Zone Configs

To manage where data is replicated, CockroachDB uses replication zones, controlled by what we call "zone configs." With these, you can identify the desired replication behavior of nodes based on their --locality flags.

For geo-partitioning, you'll want to create zone configs for each partition you want; in Movr's case, it's one for the US and one for the EU.

1. Create a zone config for the US called north_america.zone.yml with the following contents:

   ```
   constraints: [+region=us-east]
   ```

2. Create a zone config for the EU called europe.zone.yml with the following contents:

   ```
   constraints: [+region=eu-west]
   ```

3. Apply the zone configs to the table's partitions (these are namespaces defined in the PARTITION BY LIST clause):

   ```
   $ cockroach zone set movr.users.
   north_america -f north_america.zone.
   yml

   $ cockroach zone set movr.rides.
   north_america -f north_america.zone.
   yml

   $ cockroach zone set movr.users.
   europe -f europe.yml

   $ cockroach zone set movr.rides.
   europe -f europe.yml
   ```

With that done, writes that come into any table with a country value of NL will be stored on their servers in eu-west!

SSL Certificates

CockroachDB always requires production systems to run with SSL certificates. For more information on that, check out our security certificate documentation.

Role-Based Access Control

To limit the exposure of user's data, Movr creates a system wherein their vehicle maintenance team doesn't need access to their user's data.

With CockroachDB's enterprise role-based access controls, they just need to create a distinct role for that team:

```
> CREATE ROLE vehicle_maintenance;

> GRANT SELECT, UPDATE ON vehicle, ride
TO vehicle_maintenance

> GRANT vehicle_maintenace TO
maintenance_app
```

Now, clients connecting as maintenance_app can select and update only from the vehicle and ride tables.

Bonus: Application-Level Considerations

Like we mentioned at the beginning of the paper, there are many more considerations to getting your application to comply with GDPR than your database alone can handle.

Routing

To make sure your users connect to the services in the right locale, you need to handle two layers of routing:

- **Gateways** are necessary when you need to ensure clients are routed to a specific datacenter. For example, Netflix's open-source project Zuul is a popular (and powerful) gateway.
- **Routers & load balancers** are used to connect the vertical layers of your services and distribute load among the service's replicas.

Zuul (Gateway Service)

Zuul is designed as an edge microservice; it should be the first thing all users "hit" once they enter your data center, which gives you the option of routing data however you see fit.

To control data flow, Zuul offers three stages of filtering:

- "Pre" filters give you a chance to weed out traffic before actually routing it to a service.
- "Routing" filters direct users to a particular service.
- "Post" filters can manipulate responses from services before sending them to clients.

>> DID YOU KNOW

With CockroachDB, you can create a routing filter that sends any user with an EU account to an EU datacenter (even if they're vacationing in Dubai).

To comply with GDPR, you can create a routing filter that sends any user with an EU account to an EU datacenter (even if they're vacationing in Dubai).

You can get more information on Zuul at GitHub.

Routing & load balancing

For a user to traverse your stack, you need to connect its vertical layers—e.g. forwarding application requests to your database.

To ensure you're GDPR compliant, this requires more dedication and attention to maintain two distinct instances of your stack and that the two don't intermingle in ways you don't intend—you have to make sure EU users don't get routed to US services.

For instance, this means your applications might use different connection strings, which ultimately lead them to load balancers differentiated by region. Something like this is relatively simple to instrument with a tool like HAProxy, but requires care and consideration to prevent data leakage.

Restore Procedures

While your backup and restore processes interface with your database, you must have an application outside of the database controlling how things like restores actually happen.

Because you'll need a tool to ensure you fastidiously weed out any data from users who've exercised their right to be forgotten, it's best to bake this directly into your restore process.

Using CockroachDB's transactional support you can run a restore as a transaction, immediately followed by a series of DELETE for all users in your forgotten_users table.

CockroachLabs is the company behind CockroachDB, the SQL database for building global cloud services.

With a mission to Make Data Easy, Cockroach Labs is led by a team of former Google engineers who have had front row seats to nearly two decades of database evolution. The company is headquartered in New York City and is backed by an outstanding group of investors including Benchmark, G/V, Index Ventures, Redpoint, and Sequoia.

Cockroach LABS

Run your business on a database built right.
Learn more at:
www.cockroachlabs.com