How to Solve Data Residency Challenges with a Data Privacy Vault

skyflow

Contents

Abstract

Authored by Manish Ahluwalia, Field CTO, Skyflow

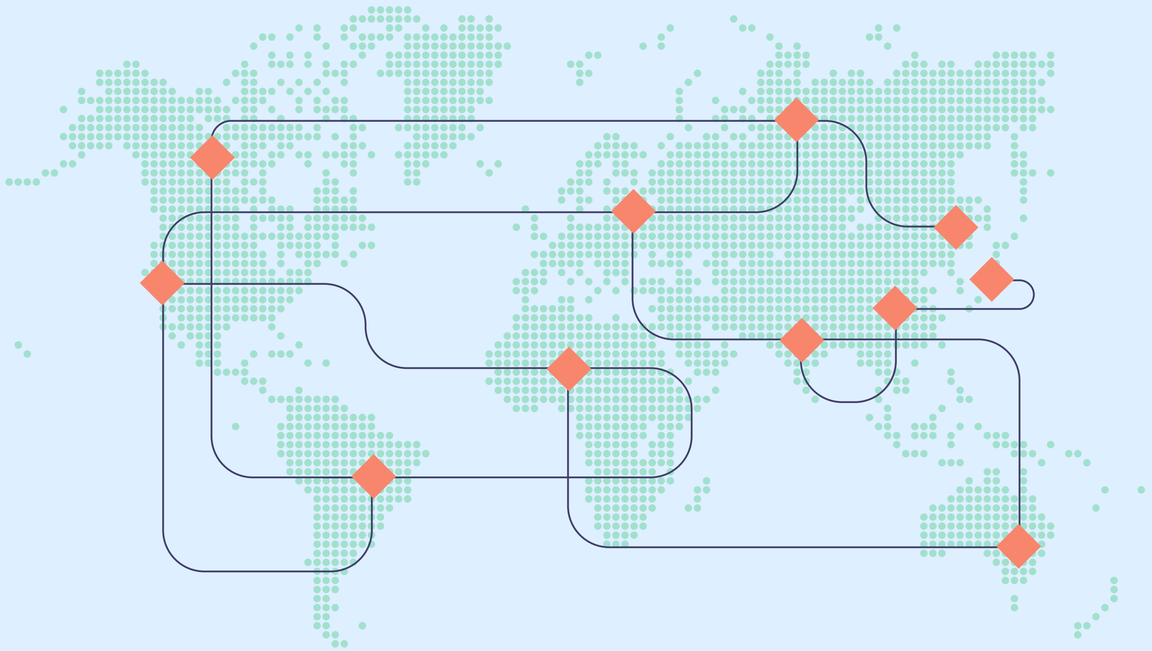
If your business has a global presence - you have customers or employees across the globe, or your customers store their globally-distributed customers' data - you almost certainly have to deal with requirements related to data residency compliance. You can use a data privacy vault to help you keep your sensitive data secure and in compliance with applicable laws and regulations, all while continuing to use this data for business-critical workflows.

Introduction

Let's say that you operate a business that needs to store global data. You have your datacenter in a centralized location[1]. The data you collect, store, and process will likely contain sensitive, private information that will expose you to compliance risk. For example, your European customers have rights under GDPR and your Brazilian customers have rights under LGPD. And the scope of this sensitive data can be wider than you might guess - it includes PII like names and email addresses that fall under the scope of GDPR, LGPD, and similar laws cropping up worldwide, as well as PHI (healthcare data). And while it's not always within the scope of data residency concerns, sensitive data also includes any other type of data that should be handled with extra care, like unreleased financial results.

The various data privacy laws and their rights frameworks are vast and varied, so when learning about a topic like GDPR you should not only read summaries and officially-issued guidance, but also consult legal counsel.

In this white paper, I'll focus on data residency requirements that are included in many data privacy laws. These requirements mean that certain individuals' data must be *localized*, or restricted to specific locations governed by that law. To get a better sense of how this works, consider an example company with customers in several geolocations.



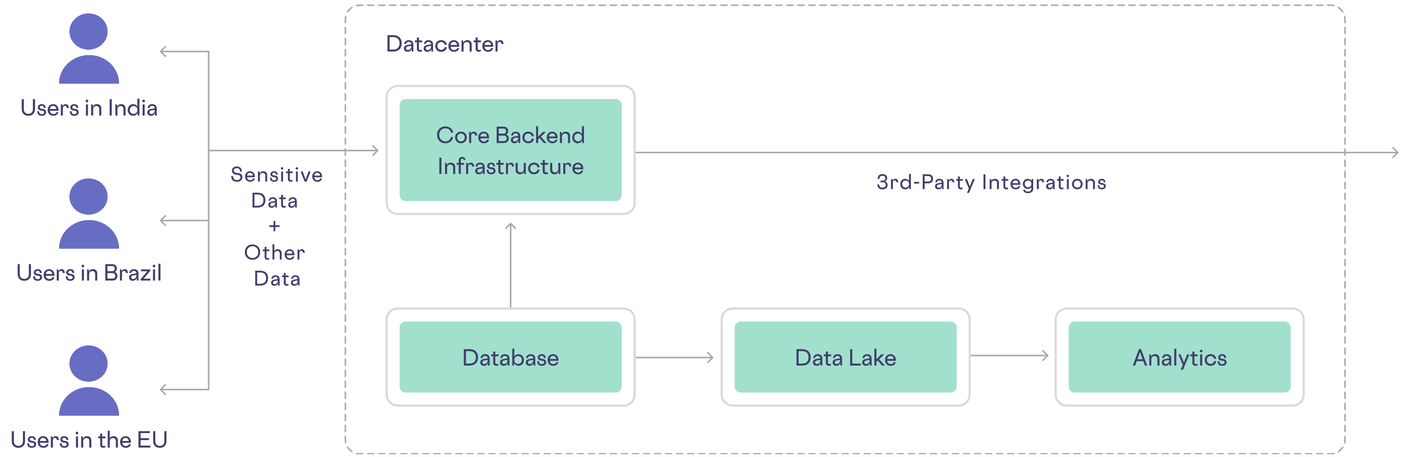
[1] Strictly speaking, the following discussion applies not just to a single physical datacenter, but equally to a single logical datacenter split physically into several locations that store and process copies of the entire dataset in multiple locations - for availability, BCDR / redundancy, performance, etc.

A Data Residency Example

Let's say that you work for a global company that has customers in the EU, India, and Brazil. GDPR requires that European citizens' personal, sensitive data remain in Europe, or is managed under rules compatible with those of Europe; India has requirements that sensitive payment data remain in India, and some countries (such as Brazil) have requirements that sensitive health data stays in the country it was collected in. This sensitive data is usually mixed with other, non-sensitive data that doesn't require special handling.

While there are supplemental measures available to transfer sensitive data to other countries from the country where it was originally collected, some of these measures are very time consuming, costly, and difficult to automate. Even if you don't have a regulator forcing you to separate sensitive data by geography, you might have internal compliance requirements or contractual requirements that force you to separate sensitive data from different sources. This is especially common for companies that provide outsourced data processing services.

A simplified centralized architecture for your business (before addressing any data residency issues) might look like the following:

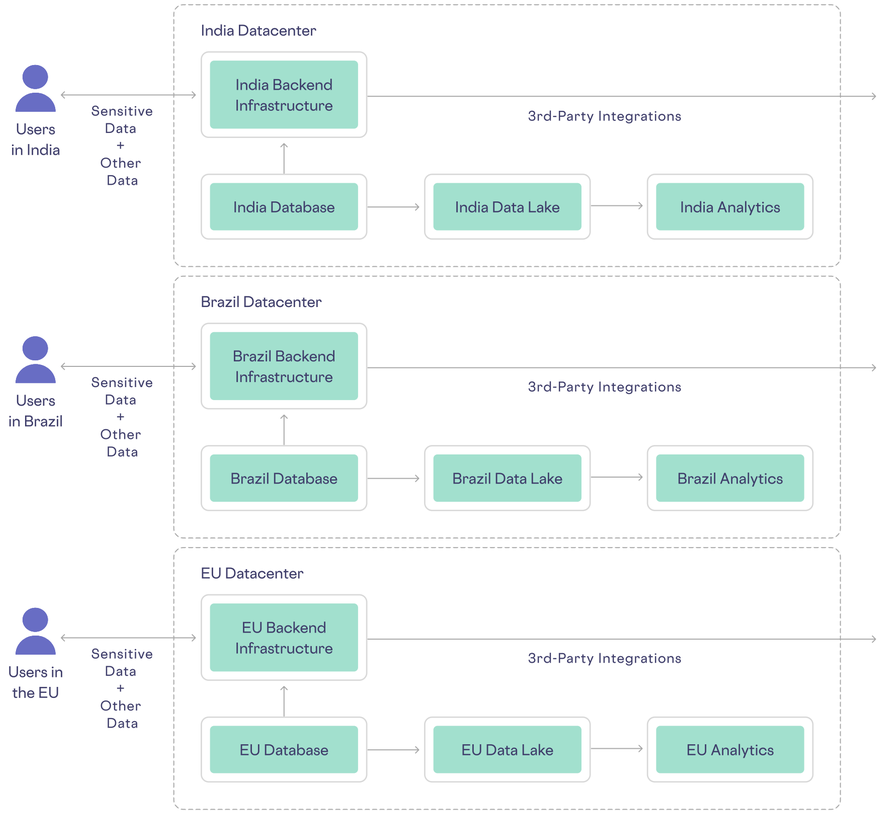


A Simple Centralized Architecture that Doesn't Meet Data Residency Requirements

Continued

In this scenario, you have one centralized datacenter. As noted above, this applies whether you run your infrastructure in a single physical datacenter or whether you're using a modern, cloud-based datacenter: a group of related services that operate as the cloud equivalent of a traditional on-prem datacenter. The data you collect from your global userbase flows to this datacenter, creating compliance headaches for your company.

One common, but limiting, approach to solving this problem is to replicate your infrastructure in multiple geographies. For example, you could create a geo-duplicated architecture, where you have a replica of your infrastructure in each region where you have users, as follows:



Geo-Duplicated Architecture: Infrastructure is Replicated Per Region

Continued

But, unless there are business reasons that you need to have an architecture like the one shown above, it's best avoided. This approach can get needlessly expensive, both in terms of infrastructure costs and operational overhead.

Generally, it's better to separate data residency concerns from the rest of your architectural decision making for a few reasons: it lets you scale to more regions without adding yet more duplication, and it means that you can improve your architecture without replicating that work across each region.

So, how can you best address data residency needs and minimize duplication beyond what's needed for high availability, even if you have a single datacenter where you centralize your analysis and use of sensitive data?

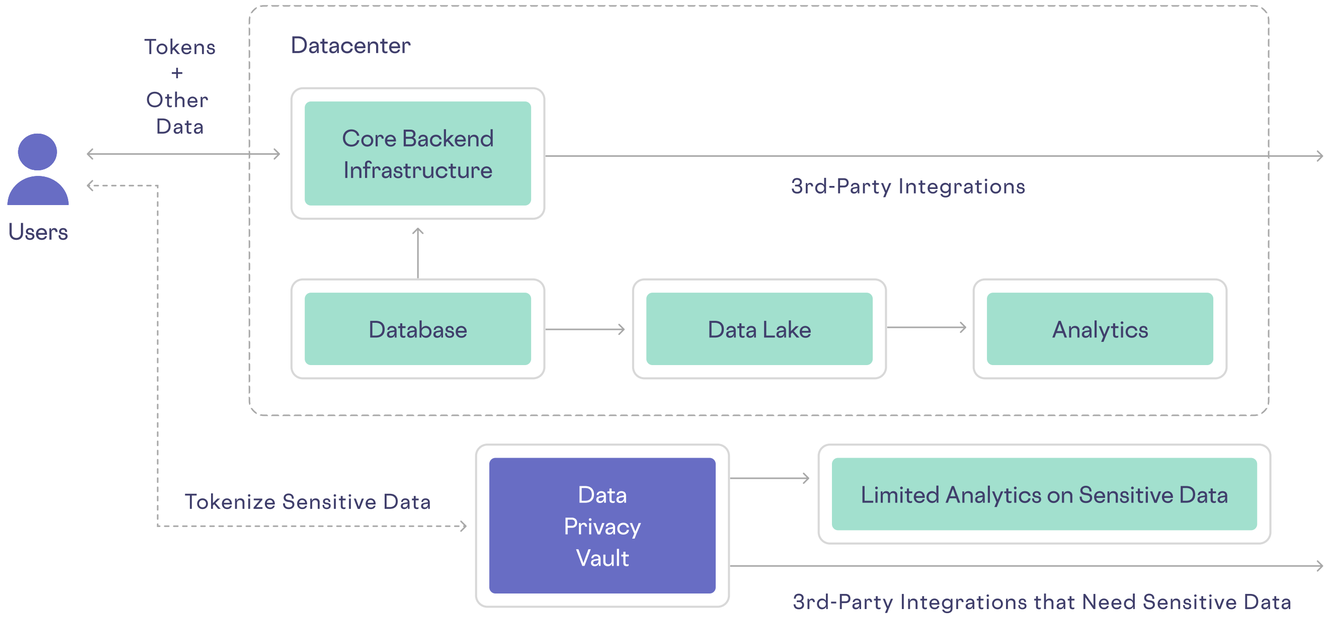
Introducing the Data Privacy Vault

A data privacy vault is an architectural pattern where your sensitive data is taken out of your general purpose systems and databases and placed in a separate, hardened, environment - the vault - that provides the cross-dataset linkages, data usage, and governance features you need to continuously execute your business-critical processes. And, it does this while improving the privacy and security of sensitive data, because that data is centralized in a vault that descopes the rest of your infrastructure from privacy, security, and compliance concerns.

When you enter PII or other sensitive data into a data privacy vault, it exchanges each sensitive data element for a de-identified "pointer" back to the original data (also known as a *token*). With PII isolated in the vault, you only need to store and process the tokens in the rest of your infrastructure. This increases the privacy and security of the sensitive data values that you've *tokenized* (exchanged for these tokens) because sensitive PII is only located in the vault, not scattered across your systems.

Continued

Leaving aside data residency concerns for now, if you started with the earlier simplified architecture and added tokenization, you'd end up with an architecture that looks like the following:



Backend Infrastructure with Tokenization

With this architecture, sensitive data that customers provide using a native client, web client, or APIs is sent directly to the vault instead of your backend infrastructure. The vault stores this data, exchanging sensitive data for tokens[2]. These tokens and any non-sensitive data are passed to your backend and from there to your other systems or to third party integrations.

There are many different options when tokenizing sensitive data, and many of these options support a wide range of operations. Using the right types of tokens to support your workflows lets you avoid the need to *detokenize* sensitive data (swap tokens for sensitive data) in most cases.

[2] In practice, switching sensitive data with tokens can happen on the edge of your network as well, making this transparent to your users. For this paper, the distinction is not material and we'll ignore it.

Continued

For example, suppose your business uses payment card numbers, or PANs – a sensitive field, regulated by PCI. Let's say that you want to tokenize PANs, while being able to analyze which banks issued which cards using just the tokens. Because the first several digits of a PAN is a bank identification number (BIN), you'll want to tokenize PANs while leaving the first four to six digits – the BIN – unchanged. By exposing this limited amount of data in your tokens, you let your analytics department do things like analyze aggregate spending per issuing bank. This reduces how frequently you need to detokenize these PANs, and removes your analytics pipeline from the scope of PCI compliance. To learn more about tokenization, see Demystifying Tokenization: What Every Engineer Should Know.

You can't perform all operations with tokens; and, in some cases, you might not be willing to make the tradeoff of exposing your systems to tokens that contain some amount of plaintext information. To support use cases where you need direct access to sensitive data and can't use tokens, a data privacy vault gives you a governance framework to securely manage limited access to this data.

How Does a Data Privacy Vault Help with Data Residency?

The crucial thing to note is that many of the requirements around which data you are allowed to store in which geolocation concern exactly the kinds of data elements that belong in a data privacy vault. This makes sense because a data privacy vault is designed to help you meet your security and compliance requirements for sensitive data.

Isolating and centralizing sensitive data in a data privacy vault and exchanging that data for tokens lets you avoid sensitive data sprawl and protect sensitive data – all while logically retaining the relationships between sensitive data and the other types of data that your business uses. You can use this capability to create architectures that meet localization requirements while avoiding the need for wholesale duplication of your architecture in each region or putting all of your infrastructure within the scope of data residency requirements.

Addressing Common Data Residency Scenarios

Let's take a look at a few data residency scenarios and see how you can address them using a data privacy vault.

Localization Respecting Architecture #1: Easily Siloed Data

In some cases, you can cleanly shard your dataset according to attributes that have data residency implications. Let's say you work for a B2B company with infrastructure deployed in the US, that manages payroll for a Germany-only employer and a Brazil-only employer. Or, you help companies to manage payments and your customer's websites operate exclusively within national boundaries - only in Germany, only in Brazil, etc. In either scenario, you would rarely, if ever, need to access multiple customers' privacy-related data in a single operation.

For an example B2B payroll company whose customers each operate within a single country, your schema and database might logically look like the following, although your database or databases could look quite different[3]:

Employee Data					
ID	Cust ID	Cust Geo	Emp Name	Bank Acct #	Pay Rate
1	A32	EU	Boris	34297	25
2	B96	Brazil	Paulo	34268	27
3	C06	EU	Olaf	25347	31
4	D42	EU	Margaret	96748	52
5	E95	Brazil	Dilma	53765	21
6	F89	EU	Angela	79309	40

A Logical View of B2B Payroll Company Customer Data, with Sensitive Data Columns in Orange

You could split this logical schema by customer geography (**Cust Geo**): The key idea is that you create a vault in each region where you operate. The vault stores the "columns" or "fields" that are subject to data residency requirements.

In this example, you secure names and bank account numbers in the vault. The central Employee data is still stored as it was previously - except that instead of storing names and account numbers, it stores tokens received in exchange for

[3] Please keep in mind that this is merely a logical view, and has nothing to do with how you store your data in your database, which could be a document DB, a key-value store, graph database, etc.

Localization Respecting Architecture #1: Easily Siloed Data

Continued

the sensitive data that's stored in the vault.

Here's a look at how the same data shown above is stored in this new architecture. First, let's look at how your database looks with sensitive data tokenized.

Tokenized Employee Data					
ID	Cust ID	Cust Geo	Token Name	Token Acct #	Pay Rate
1	A32	EU	fhhwlsu	f3-42-452d6e	25
2	B96	Brazil	hwuala	e1-48-a238b	27
3	C06	EU	sjsywku	3f-41-73e2da	31
4	D42	EU	qodkwu	1f-47-e723aa	52
5	E95	Brazil	uqejaiq	1e-44-87aa6	21
6	F89	EU	owjegse	f1-44-a42fa7	40

B2B Payroll Company Data, with Sensitive Data Tokenized

Now, let's look at two of your regional vaults that contain sensitive data and map that data to tokens. Here's the EU vault, which contains sensitive information and token mapping for EU residents:

Vault located in EU				
Cust ID	Name	Token Name	Bank Acct #	Token Acct #
A32	Boris	fhhwlsu	34297	f3-42-452d6e
C06	Olaf	sjsywku	25347	3f-41-73e2da
D42	Margaret	qodkwu	96748	1f-47-e723aa
F89	Angela	owjegse	79309	f1-44-a42fa7

EU-based Data Privacy Vault

Localization Respecting Architecture #1: Easily Siloed Data

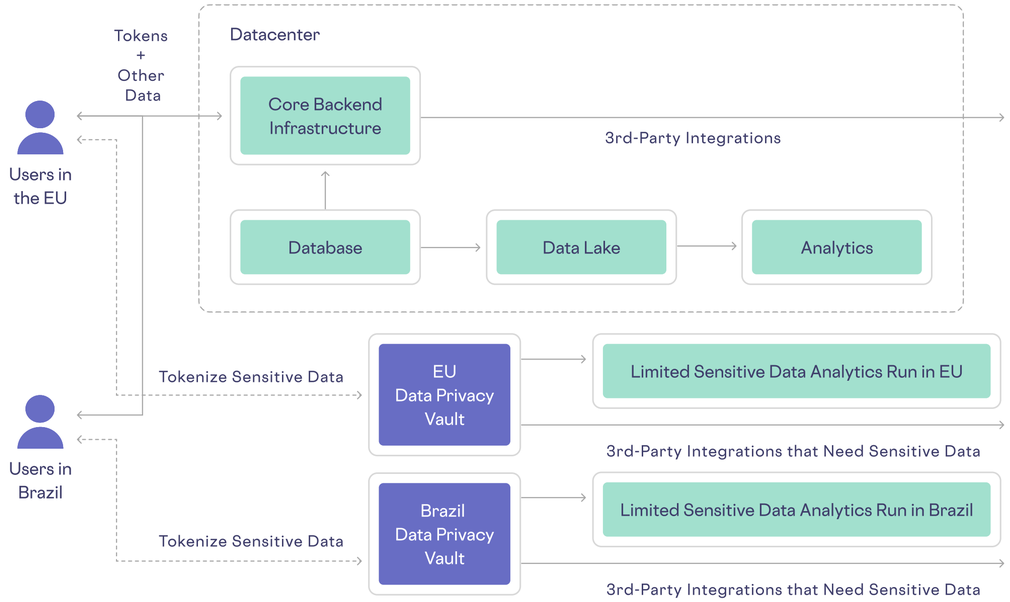
Continued

And here's the equivalent vault that contains sensitive information and token mapping for residents of Brazil:

Vault located in Brazil				
Cust ID	Name	Token Name	Bank Acct #	Token Acct #
B96	Paulo	hwuala	34268	e1-48-a238b
E95	Dilma	uqejaiq	53765	1e-44-87aa6

Brazil-based Data Privacy Vault

In this case, the vaults store the sensitive data for customers in the designated region only. The central database in the US (or your preferred geolocation) has no sensitive data. The tokens stored there come from the regional vaults, and can therefore be used by authorized parties to recreate the full record, only if needed. All other operations can just use the tokens. So a diagram of your centralized architecture with local vaults would now look like the following:



Centralized Architecture with Local Data Privacy Vaults and Tokenization

Localization Respecting Architecture #1: Easily Siloed Data

Continued

With this architecture, you're still running one centralized datacenter. The sensitive data elements are sent to the data privacy vault instance in the correct geolocation[4], returning tokens in exchange. These tokens, along with the non-sensitive data you handle, are all that your centralized datacenter needs to deal with. For use cases that require sensitive data, vault policies allow a local execution of these processes, all subject to policies based on data protection requirements in that jurisdiction - for example, under LGPD sensitive customer data from Brazil can only be accessed from within the borders of Brazil.

Localization Respecting Architecture #2: Non-splittable Global Dataset

Your data residency challenges might be more complicated than in the hypothetical example described above - you might not have a natural way to split your global dataset into partitions (or shards) with distinct localization requirements. If that's the case, you'll need a different architecture than what's shown above to avoid unnecessarily sharding your dataset. Let's look at a few examples of how you might build an architecture to support situations like this.

Analytics on Your Global Dataset

Sharded datasets can make some queries more challenging. For example, if you want to run a query on all your global customers and your customer data is sharded by geography, first, you have to run pieces of the same operation in multiple datacenters. Then, you have to integrate those results without losing accuracy, and all while remaining in compliance with data export policies in each region.

◆ Exporting only Aggregates from the Local Vaults

Let's say that you run an international ecommerce site and you want to get a demographic breakdown of your users grouped by age and annual spending to help inform product and marketing choices. If you have a large number of users in each age group in each geolocation, you can do this operation by performing the computation independently in each vault

[4] This is a subtle but important step. A variety of legal and policy decisions need to be made here to support various scenarios, like what happens when a citizen of one country travels to another country with incompatible localization requirements. Executing the transfer of data to the vault also has complications due to data being possibly routed through other, non-compliant regions. A general-purpose solution will allow you to specify policy while taking care of how the subjects' data travels in a secure and compliant manner to the correct vault.

Exporting only Aggregates from the Local Vaults

Continued

(the *local phase* of query execution), and exporting the aggregate information[5] to a central location where you can combine it to get the final, global query result (the *global phase* of query execution).

More complex aggregations will require that you export multiple types of information from each local query to complete the global query without losing accuracy. For example, to compute the average spending within each age group across multiple countries, you don't just need to export the average spending for each age group within each geolocation. Instead, you'd need to export totals (i.e., the total number of users by age group) and counts (i.e, their aggregate spending) from each local query to run the global query. Without these totals and counts, you would know the average spending by age group within each geolocation, but you wouldn't have an accurate way to combine these to get the global query result – average spending per user by age group across all locations.

◆ Queries that Join Local and Global Data

This can get complicated fast, so we'll just highlight a few use cases and their solutions.

Example 1: Basket Analysis by Credit-card Issuer

Storing and processing credit card data has security, privacy, and compliance implications. So this is a great example of data that belongs in a data privacy vault.

Let's say your marketing team wants your help answering a question like "across the world, what are the shopping preferences of Amex Centurion cardholders vs Visa Retail cardholders?"

While it might look like you can't answer this question without accessing your users' full card numbers, an *information exposing token*[6] makes this possible. With an information exposing token – in this example, a token that contains a few unchanged digits from the full credit card number and tokenizes the remaining digits – you can run this query using only tokens.

A well-designed "information exposing" token for this scenario exposes only the BIN number of your card, keeping your

[5] Aggregate, de-personalized information (if aggregated with techniques like differential privacy) may not be directly identifying information. But depending on the technique used and how the data was collected, the ability to re-identify individuals with this information can mean it's still sensitive according to the applicable regulations. So, it may still have privacy or regulatory implications - check with your local privacy expert.

[6] A token that contains in it a part of the information being pointed to by the token. This creates a small information leak with the tradeoff of usability and requiring fewer users / systems to have vault access. Please see our tokenization whitepaper (linked above) for details.

Queries that Join Local and Global Data - Example 1

Continued

customers' full card numbers secure and keeping you compliant with PCI regulations. This means that you can answer the question from the marketing team using a simple query that works just as well with tokenized data as it would with full credit card numbers - no query modifications necessary.

Here's how this workflow might look - note that the BIN portion of each PAN is in **bold** for emphasis:

B2B Payroll Company Data, with Sensitive Data Tokenized

Example 2: User Behavior Analysis by Age

Most databases don't store age (since that changes over time), but compute it on the fly using a **date of birth** field, which is sensitive information. If your users' **date of birth** data is stored in local vaults, but their behavior data is in the global data lake, you now have to join these datasets before you can perform any analytics or workflows.

Depending on the exact nature of the problem you're solving, this might be a query that you can split into a local phase and a global phase, as described above. Or, maybe you could export non-vaulted data to the local vaults to perform a join on age, before finishing the processing globally (essentially a three phase computation: export to local vault, local query phase, global query phase).

Sensitive Data Integrations with Residency-Respecting 3rd Parties

Imagine you want to send out an email to all of your customers on their birthday. This email campaign needs to know the

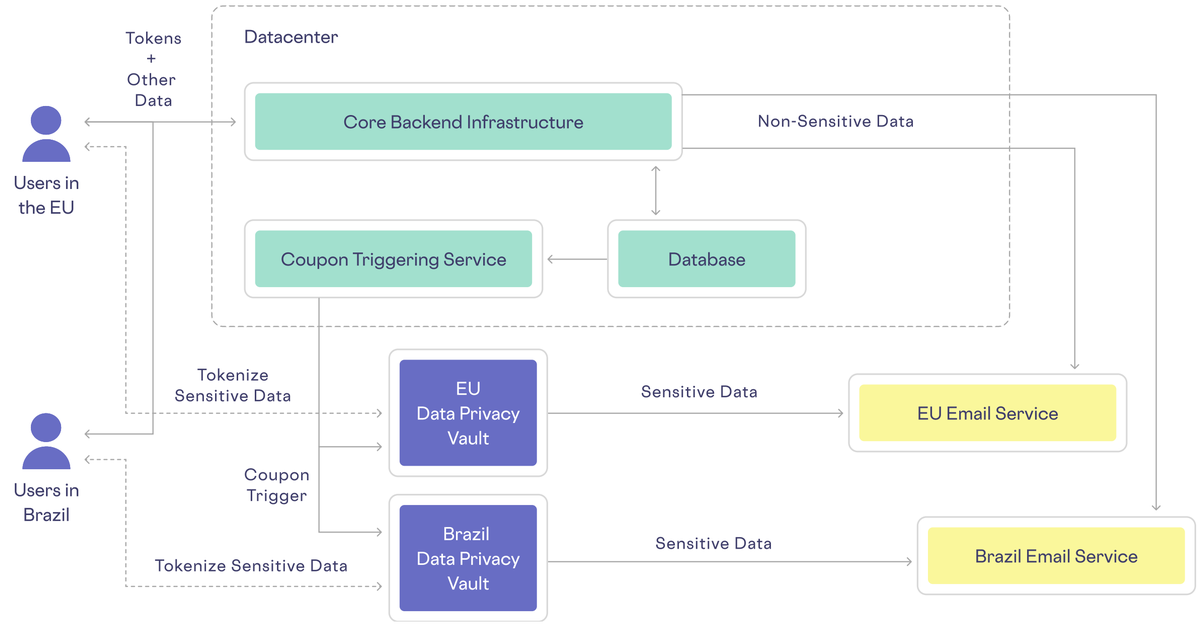
Sensitive Data Integrations with Residency-Respecting 3rd Parties

Continued

date of birth and email address for each customer, both of which are sensitive, personal data that belong in local data privacy vaults.

To send out these emails while staying in compliance with various data residency requirements, you've decided to work with multiple email providers that respect local data residency constraints and that share the geolocations of your customers. So, for example, you have a Brazilian email provider and an EU email provider to send email to your customers in those geolocations. And of course, you're looking to automate your work with these email providers.

To support this scenario, you could set up these email campaigns to trigger from your local data privacy vaults. Your users' **date of birth** data is used in each of the vaults to determine when to send each email (and to whom). The vault integrates with the local email providers to send out the emails as each customers' birthday occurs, as shown below:



A Privacy-preserving Workflow for Sending Email Messages on Customer Birthdays

Sensitive Data Integrations with Residency-Respecting 3rd Parties

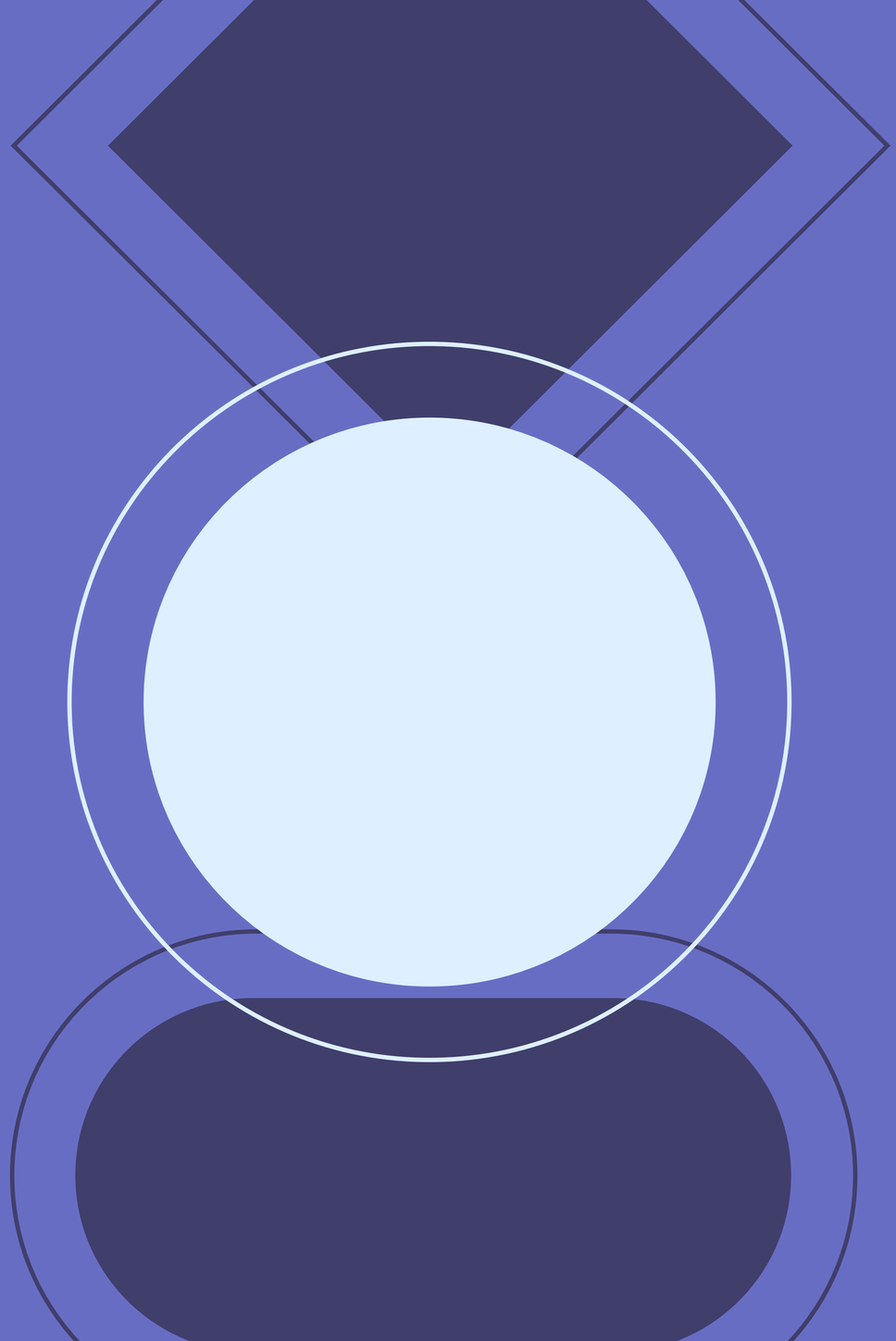
Continued

Of course, part of vetting these local email service providers includes ensuring that they protect the data that you're sharing with them as required by data residency laws and any other data protection requirements. The fact that an email was sent to a user on a specific date becomes a good proxy for their birthday (although without the year). So, in this scenario you'd want to ensure that these providers handle any email audit logs in a way that protects your users' data privacy.

How Skyflow Eases Data Residency Compliance

Ever-increasing data residency obligations can be burdensome for architects, data engineers, systems engineers, and compliance professionals. However, careful use of a data privacy vault can ease the burden of complying with GDPR and other laws without sacrificing functionality, or speed of execution.

To do this, you need a data privacy vault designed with in-depth support for a variety of scenarios and third-party integrations, and the right solution architecture. For more on data privacy vaults, see, What is a Data Privacy Vault? To learn more about the Skyflow Data Privacy vault, give it a try or contact us.



About Skyflow

Founded in 2019, Skyflow is a data privacy vault for sensitive data. The company was founded by former Salesforce executives Anshu Sharma and Prakash Khot to radically transform how businesses handle users' financial, healthcare, and other personal data that powers the digital economy. Skyflow is based in Palo Alto, California, with offices in Bangalore, India. For more information, visit skyflow.com or follow on Twitter and LinkedIn.

About the Author

Manish Ahluwalia has over 2 decades of experience in the software industry, with over 10 years in information-security. Most recently he was running security for NerdWallet. He currently works to help Skyflow's customer's find the right architecture for their data protection needs.

skyflow