S9S GUIDE

Developers Guide to

Sovereign DBaaS

several9s

Traditional DBaaS (*database-as-a-service*) is falling short in meeting the evolving needs of businesses today. While DBaaS has certainly made it easier for companies to set up and manage databases, it has also introduced a number of new challenges and risks. The standard approach is increasingly inflexible, leaving companies at the mercy of cloud and other 3rd-party vendors, an ever-changing regulatory landscape, license instability, and unreliable cost structures.

Many organizations opt for a traditional **Database as a Service (DBaaS)** because it's easily implemented and managed. They have made this choice because it allows them to focus their resources on building and improving their products and driving results for their business, instead of the administration of the database layer.

DBaaS lets organizations test ideas more quickly, cheaply, and with a lot less risk. They don't have to make large, upfront commitments to infrastructure and they don't have to recruit for and build out large teams of infra and database experts. These freedoms enable smaller startups and even non-profits that otherwise wouldn't have the resources to bring their ideas to life in days past. Cloud and the implementation schemes tied to it have heralded an unprecedented age of innovation due to improving access to compute, storage, and lower-level systems that applications are built on.

But there are several business risks associated with the standard DBaaS model: vendor, environment, and ecosystem lock-in, managed database license instability, key person dependencies, and vulnerability to changes in data protection regulations. These risks can be costly, disruptive, and difficult to mitigate.

Sovereign DBaaS is a new implementation model that addresses the business risks of traditional DBaaS in a number of ways, such as by providing true control of the database layer through minimizing multiple forms of lock-in. For instance, it minimizes environment and ecosystem lock-in because the database is decoupled from the underlying infrastructure, so an organization can place their workloads where it makes most sense.

The license stability angle doesn't get much play when open-source licenses are changed. But this instability begs the question, "Can an organization using managed database services, open-source or not, be said to have true control over their databases if they can lose access to it when the vendor changes their license, as in the cases of MongoDB and Elasticsearch?"

The decoupling of infrastructure and database, and the ability to freely and confidently deploy open-source and source-available databases, ultimately enable organizations to build out a more resilient database layer that is aligned with the needs of the business.

True sovereignty means intelligently utilizing technologies to maintain the ownership and control of your database layer, which extends to your actual data. Sovereign DBaaS allows for ultimate influence over the full spectrum of business risks related to your data.

We created this guide to help developers understand:

- ✓ The importance and power of data sovereignty.
- ✓ The differences between traditional DBaaS and Sovereign DBaaS.
- ✓ The pillars of sovereign data and the principles of Sovereign DBaaS.
- ✓ Application use cases for Sovereign DBaaS.
- ✓ How to become data sovereign by rolling your own DBaaS.

First, we should define traditional DBaaS and Sovereign DBaaS. The key distinction comes down to this question: who owns your data?

What is DBaaS?

Before we dive deeper, let's define DBaaS by starting with the goal — reliable database ops at scale — and a DBaaS implementation achieves this via automation. From there, we can define two implementation concepts, managed and Sovereign DBaaS:

- ✓ **Managed DBaaS** - DBaaS run on a cloud-based provider as a consumption-based service.
- ✓ **Sovereign DBaaS** - DBaaS deployed on infrastructure you control via homegrown and purchased tools.

The concept of DBaaS is usually associated with the idea of a managed database service that allows you to create and operate databases without having to build and manage underlying infrastructure or database software. A DBaaS provider takes care of all these things for you, so you can simply provision and deploy your databases using the provider's interface or other tooling, such as an API or CLI. The provider handles the technical details of the implementation for the database infrastructure and manages the database software's operations, including failover, recovery, backups, and patches.

But while DBaaS is commonly associated with managed service providers who handle operations and assume responsibility for the database layer, **DBaaS is ultimately an implementation concept that is about ensuring reliable database operations at scale through the use of automation**. A DBaaS implementation concept doesn't necessarily have to be tied to who is actually delivering it. You can implement DBaaS using third-party providers,

yourself, or both, e.g., depending on workload requirements. That optionality naturally allows for different implementation concepts. In this guide, we are breaking down DBaaS into two distinct concepts: Traditional and Sovereign.

Managed DBaaS

A traditional DBaaS concept is mediated through service providers in the public cloud and covers database layer responsibility, control, and location. For example, Oracle MySQL Cloud Service, AWS RDS, and Aiven for MySQL are fully managed services wherein all operations, including provisioning, failover, recovery, and deprecation, are handled by the vendor's control plane and backed by an SLA from the provider. There are benefits to this model, which is why it's exploded in popularity. In short, these benefits come down to leveraging focus and expertise. By offloading database layer operations, organizations are able to move on and focus on developing the applications that deliver business value. But, leveraging the focus and expertise of a traditional DBaaS provider is a trade-off for additional business risks that become more acute over time as an organization grows in sophistication and the landscape changes around it.

Sovereign DBaaS

Sovereign DBaaS is primarily distinguished from traditional DBaaS by allowing for organizations to affect greater degrees of control over their database layer.

In contrast to traditional DBaaS, Sovereign DBaaS is an implementation concept that entails organizations themselves automating their database layer operations, either fully or partially, using homegrown code, open-source tooling, and/or off-the-shelf proprietary software in any environment, e.g.

cloud, on-premises, or hybrid. Historically, this would be almost impossible to do without Herculean effort. But it is now possible for any organization to implement their own Sovereign DBaaS, enabling organizations to get the benefits of DBaaS while mitigating the looming business risks associated with traditional DBaaS, such as license instability and undesirable dependencies.

Sovereign DBaaS lets you deploy databases in an environment-agnostic manner to vendor-neutral environments so that you retain complete control over your databases and data. You control the locations of your dedicated tenants. The location of the infrastructure may be hosted on-premises, in a colocation facility, or in a hyperscale cloud provider facility as infrastructure-as-a-service (IaaS). The important distinction is that the choices are yours, and you can change them at any time for any reason.

Beyond compliance and legal controls, Sovereign DBaaS enables ultimate control for all business risks related to data. Specifically, sovereign control mitigates vendor, environment, and ecosystem lock-in, open-source license instability, key person dependencies, data regulation changes, and infrastructure cost unpredictability.

Traditional DBaaS vs. Sovereign DBaaS

	Managed DBaaS	Sovereign DBaaS
Control	Vendor controls and manages the database infrastructure and software for you — you have limited control over fundamental choices.	You own and can manage your infrastructure and databases to suit your needs. You aren't limited in how you can change your service and have the power to fine-tune for your use cases.

	Managed DBaaS	Sovereign DBaaS
Access	In general, a black box — you can't get to the deeper levels. When something goes wrong, you are reliant on the provider to fix it because you lack visibility and access.	You have root access to the database and the infrastructure layer to varying degrees. You can make core operational changes to database systems to better align them with your use case. When something goes wrong, you have direct access to fix it.
Portability	Little to no portability — designed so that you are dependent on a service or vendor ecosystem. Porting to another provider can be very difficult.	No vendor or environment lock-in. With true open-source databases and tooling that can be implemented anywhere, you can easily migrate to another cloud provider or even on-premises.
Environment	Environment choices are limited to vendor offerings, usually the big 3 public clouds. Hybrid environments are far and few between and often lock you into an ecosystem.	Unrestricted choice of environment, dependent only on your own requirements. Utilize any combination of clouds, on-premise implementations, and hybrid setups.
User experience	A single user experience but limited flexibility. Using multiple vendors fractures user experience and can be prohibitively challenging.	You can combine homegrown work with off-the-shelf products that accommodate multiple DBs and environments. This allows for a more flexible and customizable solution and better user experience.
Security and privacy	You share responsibility with the vendor and adopt the security and privacy frameworks they offer.	You can configure security and privacy based on your own requirements and frameworks.

	Managed DBaaS	Sovereign DBaaS
Licensing	Offers access to multiple license types, but also subject to changes in licenses which may result in losing access to the product or some of its key features, posing sustainability risks and vendor lock-in risks.	You are not subject to the same license restrictions as DBaaS vendors, with the freedom to choose the database technologies that best suit your organization's needs without concern over license instability.
Costs	Traditional DBaaS operates under an hourly consumption model that is billed monthly. Few of these services are straightforward to cost out. While they are cost-efficient at small scale, many customers are caught by surprise when they reach large scale implementations.	You have the ability to determine the level of control you want to assert over your deployment stack, so you can better control operating costs. Also allows for cost tracking and build more accurate models.

Control

Traditional DBaaS using a managed service means the vendor manages your database infrastructure and software. You don't have to do any of the operational work, giving you more time to focus on getting to a finished product or application. This sounds great, except you have no control over your database infrastructure and software.

Sovereign DBaaS offers much more control than traditional managed service providers by giving you the ability to affect change at every level. With Sovereign DBaaS, you have the flexibility to self-manage your databases and determine the level of control you have over your infrastructure. This allows you to have more choice in the type of infrastructure and databases you use, as well as how they are configured and tuned.

Complete ownership gives you greater control over your data and empowers you to make decisions that best suit your needs. In contrast, using a managed service provider limits choice, offering little to no control over how infrastructure and databases are coupled and tuned, and you ultimately own nothing.

For example, you may need to place workloads in specific environments and locations based on business and regulatory requirements, such as those applicable in the banking sector. That doesn't include access requirements for the databases that underlie critical apps. Sovereign DBaaS gives you complete control over where and how you administer your database layer. With a traditional DBaaS implementation, you will have limited to no control over their location and administration.

Sovereign DBaaS enables full control of database technologies, enabling any custom combination of proprietary, open-source, or mixed/gray licensed databases. Rather than being forced to use certain database technologies based on a vendor's offerings, you have the option to roll your own individually optimized solutions.

Access

Traditional DBaaS offerings are often referred to as a "black box" because users have little visibility and access into the deeper levels of the service. This means that when something goes wrong with the service, users are

reliant on the provider to fix the issue. This can result in delays and a lack of control over the resolution of the problem.

Likewise, users are not able to intervene in lower-level operations, such as configuration and tuning, which can limit their ability to optimize the service for their specific use case. This lack of visibility and control can make it difficult for users to understand and troubleshoot issues, and can ultimately impact the performance and reliability of their application.

In contrast, Sovereign DBaaS provides users with root access to the database layer, as well as the infrastructure layer to varying degrees. This means that users have the ability to make core operational changes to their database systems, such as configuring and tuning, in order to better align them with their specific use case. This level of control and access allows users to optimize the implementation for their needs, resulting in better performance and reliability. Additionally, when something goes wrong, users have direct access to the underlying systems, which allows them to quickly troubleshoot and resolve issues. This empowers users to take ownership of their data and manage it in a way that best suits their needs.

Portability

Companies that use traditional DBaaS will inevitably face the dilemma of vendor lock-in. Whether you choose an open-source, mixed, or proprietary licensed database, once you deploy it with a traditional DBaaS, you face a considerable challenge if you one day want to leave. Traditional DBaaS is designed so that you are fully dependent on that service — they offer little to no portability. Many providers offer proprietary databases with proprietary data formats. So, moving would likely require that you transform your data into a new format for the new target database.

The difficulty of porting databases comes down to two key considerations: provider and environment. In terms of providers, migrating away from

proprietary databases can be the most difficult, as it often requires a complete refactoring of the data and application. Open-source databases are the next most difficult, as migrating away would require partial refactoring, dealing with different APIs, nomenclature, and semantics. For example, transferring from Aiven for Kafka to AWS MSK would require such refactoring. Or migrating from AWS RDS would require a logical backup, i.e. reading data from the database, as AWS does not provide access to the actual backup files. However, if you implement open-source via a Sovereign DBaaS scheme, it's closer to a 1:1 transfer because you're not having to plug into the above.

In terms of environments, traditional DBaaS is in the cloud by default, so you can migrate from cloud to cloud, but you are unlikely to find a provider that delivers their services on-premises or in a mixed environment. Additionally, cloud-to-cloud migrations typically only entail major cloud providers like AWS, GCP, and Azure. Even if they do provide on-premises deployment options, they are still likely tied to an ecosystem, such as in the case of AWS Outpost. The key difference with Sovereign DBaaS is that you are truly free to set up a managed environment anywhere you want, without the limitations of traditional DBaaS providers.

Overall, a key advantage of Sovereign DBaaS is that it provides more flexibility and control over your databases, which can help to avoid vendor lock-in and improve portability. By using open-source databases and providing root access, you can configure, tune and manage the databases to better align with your use case and easily migrate to another cloud provider or even on-premises.

Environment

When it comes to choosing a DBaaS solution, one of the key considerations is the environment in which the service will be deployed. Traditional DBaaS

providers often limit your environment choices to their own offerings, making it difficult to implement a hybrid environment that meets your specific requirements.

With traditional DBaaS, your options are limited to the vendor's ecosystem, which can be restrictive and make it difficult to achieve a tailored solution that meets your business needs. For example, if your organization requires a hybrid environment that includes on-premises and cloud deployments, you may find that traditional DBaaS providers do not offer this option or have very restrictive implementations. This can make it difficult to achieve the desired level of control and flexibility over your data infrastructure.

On the other hand, Sovereign DBaaS offers unrestricted choice of environment, dependent only on your own requirements. With Sovereign DBaaS, you have the flexibility to utilize any combination of cloud providers, on-premise implementations, or hybrid setups, to best align with your use case. This enables you to tailor your environment to meet the specific needs of your organization, whether that's for cost savings, performance, or security. For example, you could use a cloud provider for your production environment and an on-premise implementation for your development and testing environments.

Sovereign DBaaS gives you the freedom to choose the environment that best suits your organization's needs, without the limitations imposed by traditional DBaaS providers. This level of control and flexibility over your data infrastructure can help you achieve a tailored solution that meets your specific requirements and improve the performance, security and cost-effectiveness of your DBaaS solution.

User experience

Traditional DBaaS from a managed service provider offers a simplified user experience with a unified interface out-of-the-box in trade for control and

flexibility. This is appealing for organizations that are looking for a simple and easy-to-use solution, but it means that they have to make major sacrifices in terms of any customization and flexibility they want to use that isn't part of the vendor's services.

The control and portability offered by a Sovereign DBaaS implementation mean that you can choose to implement one control plane across your database layer, and thereby achieve a more consistent user experience. You can combine homegrown work with off-the-shelf products that accommodate multiple DBs and environments. This allows for a more unified, customizable solution, although doing it on your own does require more work initially.

Companies often look to build their database layer with 3rd-party managed open-source databases. A rising trend over the last few years is open-source database vendors restricting their licenses so that they are unavailable as services outside of the actual vendor. Mongo and Elasticsearch are prominent examples. If your architecture calls for using services restricted in this way, then you'd potentially have to go with multiple vendors. Multiple vendors and different control planes fracture user experience. Additionally, organizations that are trying to span environments, i.e. implement a hybrid scheme, may be limited in their options with traditional DBaaS.

Security and Privacy

Security is a crucial benefit of using a DBaaS. A good DBaaS will use encryption technologies and multiple layers of security to protect data. A traditional DBaaS from a managed service provider uses a form of shared responsibility model for some areas of security, usually involving keys. Traditional DBaaS providers manage every aspect of the underlying infrastructure and control the handling and management of your data.

This is part of the reason why there are security and privacy compliance standards that many of the providers carry, such as ISO 27001, SOC 2, etc.

Depending on how you implement a Sovereign DBaaS concept, your privacy posture will be derived from what policies you have in place, what you build, and what off-the-shelf software you buy.

Privacy-wise, the location of the cloud hosting the DBaaS plays a significant role. Many Western workloads are hosted by cloud providers headquartered in the U.S., which are subject to U.S. laws that may conflict with **GDPR** and other international privacy regulations. This problem will become even more complex as more countries implement their own regulations, each with its own unique requirements and standards.Therefore, organizations that have operations and customers in multiple countries will face an increasingly complicated regulatory landscape.

Additionally, there are certain workloads, such as those related to national security, that may not be suitable for hosting in a public cloud. Sovereign DBaaS offers a solution by giving users complete control over where they place their workloads, making complying with multiple regulatory standards more easily achievable.

Licensing

When implementing Traditional DBaaS, you will have access to multiple license types such as proprietary, open-core, and open-source. However, it also means that you will be subject to changes in licenses which may result in losing access to the product or some of its key features, such as InfluxDB, Kafka, TimescaleDB, Elasticsearch, etc. This poses sustainability risks, particularly when building the database layer around managed services.

One option is to go with the database vendor themselves, which poses vendor lock-in risks and fragmentation issues. The other option is to replace it with another database, which can be a nightmare.

With Sovereign DBaaS, you are not necessarily subject to the same license restrictions as a vendor. Therefore, you can still freely use databases that are no longer available as 3rd-party managed services such as MongoDB and Elasticsearch, and you don't have to worry about license changes rendering your implementation strategy obsolete. You have more control over your data infrastructure and can choose the environment that best suits your organization's needs without the limitations imposed by traditional DBaaS providers.

Costs

Traditional DBaaS operates under an hourly consumption model that is generally billed monthly. While this may seem straightforward at first, it can quickly become complex and difficult to cost out, especially for large-scale implementations. One of the biggest challenges with traditional DBaaS is the lack of transparency around costs. Many customers are caught by surprise when they reach large scale, as they discover that the costs of running their database have skyrocketed.

In contrast, with Sovereign DBaaS, organizations can better control operating costs because they have the ability to determine the level of control they want to assert over their deployment stack, e.g. tools, databases, environments, providers, etc. Additionally, with Sovereign DBaaS, they can better track costs, helping them better predict and manage their expenses.

Other ways Sovereign DBaaS reduces costs include:

- ✓ The use of open-source databases.
- ✓ Adding guardrails to improve cost predictability and transparency.
- ✓ Distributing workloads across more cost-efficient hybrid multi-cloud deployment options.

As you can see, the differences between traditional DBaaS and Sovereign DBaaS impact many aspects of deploying and operating databases. But

the most critical benefit Sovereign DBaaS brings is true data ownership or data sovereignty. In the next section of this guide, we'll expand on key concepts and define the principles and pillars of data sovereignty.

II. Sovereign DBaaS means true data ownership

Data sovereignty as a conceptual framework means leveraging all available technologies and following the best-known practices to manage all the business risks associated with your data. We explained above the concept of Sovereign DBaaS and how it differs from traditional DBaaS solutions. In brief, it comes down to the level of control you have over your database layer. Possessing control over your database layer is ideal for many reasons, not least of which is the ability to handle increasing compliance requirements and mitigating assorted business risks. This derives from the flexibility to freely adapt to ever-changing conditions.

Data legislation, privacy, and residency

Most countries have laws regarding the handling and processing of data, and those laws mostly revolve around consumer privacy rights. If your business operates in multiple countries or handles data for citizens in certain areas (e.g., European Economic Area (EEA), California, Canada), you need to manage your data effectively to comply with all applicable privacy laws and regulations.

Managing data against multiple different data regulations simultaneously presents a major challenge that becomes especially difficult without sovereign control. Let's take a look at some notable data laws:

- ✓ **GDPR** — The EU General Data Protection Regulation (GDPR) requires that companies have processes to ensure the security and privacy of the personal information of citizens located in the EEA. The law has numerous requirements based on data concepts such as fairness, transparency, accuracy, accountability, and confidentiality. GDPR is consent-based (opt-in); companies must have consent before collecting and processing users' personal information. Companies that want to operate in the EU must carefully adhere to GDPR rules or they may face severe consequences. Many well-known organizations have already been forced to pay large fines for non-compliant data practices. Bottom line: if you want European customers, you must comply with GDPR. Sovereign DBaaS makes this much easier.
  - **Schrems II** — This court ruling invalidated the EU-US Privacy Shield, delegitimizing the process of transferring personal data from servers in Europe to the U.S. and other countries worldwide. However, had the shield remained in place, it wouldn't have guaranteed that European citizens' data would remain private. Whether the data is stored in Europe or the U.S., American companies must comply with intelligence agency requests. So, American companies can't guarantee data privacy to European citizens.

- ✓ **CCPA** — The California Consumer Privacy Act is a California state law requiring businesses to follow specific guidelines in collecting, handling, and selling Californians' personal information. The law establishes six central data rights for consumers, including the right to delete data collected about the consumer and the right for data disclosure (e.g., when and why data was collected, and if the data was sold to a third party). CCPA is opt-out; consumers can opt out of businesses collecting, processing, or selling their personal information.
  - **CPRA** — California Privacy Rights Act (CPRA) is an amendment to the CCPA adding two rights to be enforced starting in July 2023.

These two rights are the ability to correct inaccurate information and limit the use and disclosure of sensitive personal information.

Regarding data regulations, you need to know which laws include data residency rules, including localization. Data residency is the place or location where the data is stored and data localization requirements specify the storage of data in the region from which it came. For example, if you collect data from Canada and then process and store that data in Canada, you are practicing data localization. Organizations increasingly face tremendous challenges regarding data regulations, and many must navigate multiple complex compliance requirements.

Data sovereignty via a Sovereign DBaaS can help you more easily comply with privacy regulations because you are able to determine the level of control you want over every step of the data collection, handling, and storage process. With traditional DBaaS vendors, you don't. By embracing the pillars of sovereign data and implementing a Sovereign DBaaS, you will be well situated in remaining compliant with current laws and able to quickly adapt as needed.

The Principles of Sovereign DBaaS

A Sovereign DBaaS implementation requires three key principles:

1) End-user independence
2) Environment agnosticism
3) Open-source / source-available software (OSS)

First principle: End-user independence

End-user Independence arises from two conditions: visibility and control of the database layer.

The first factor required for end-user independence is full visibility and transparency into the database layer. This includes end-to-end visibility into the technologies and software the DBaaS uses. Sovereign DBaaS can offer complete data transparency with no intermediaries (e.g., vendors) withholding information about the components and processes being used to implement the stack. Traditional DBaaS is a veritable black box — you can't see into it, i.e. the data management software, security configurations, or privacy protocols, etc., just the output.

Arising out of visibility is the second condition, control, which is a function of the following:

- ✓ **DB and infrastructure access** — You can modify the database/infra configuration and everything that the configuration entails. This is made possible by the direct use of open-source software, unmediated by a vendor's implementation, enabling you to better tune your databases to support your workloads.

- ✓ **Location choice** — You decide where and how data is processed and stored. For instance, you can place workloads with stringent requirements in one environment, such as on-premises, and those with fewer in another, such as public cloud. These requirements don't just have to revolve around compliance and security, but performance, cost, and other variables that influence your workloads as well.

Second principle: Environment/ecosystem agnosticism

Sovereign DBaaS enforces the idea of environment agnosticism and extends it to the ecosystem. It means that end users have the freedom to choose different infrastructure environments and the ability to combine multiple underlying environments into a unified control plane. They get environment

agnosticism – which enables location control. You can choose one environment or select from a mix of environments such as private cloud (e.g., VMware, Nutanix, OpenStack), public cloud (e.g. AWS, GCP, Azure, etc.), on-premises, co-location, and hybrid.

Sovereign DBaaS means having the freedom to go beyond any one ecosystem. For example, AWS Outposts lets you run on-premises. However, this setup is not truly Sovereign because, aside from the managed service aspect, you're locked into the AWS ecosystem.

Third principle: Embracing Open-Source Software (OSS)

A crucial principle of Sovereign DBaaS is the unrestricted use of open-source software. OSS allows you to avoid many of the issues you see with proprietary cloud vendor solutions, such as vendor lock-in. You have the ability to freely utilize the best OSS databases available, without worrying about managed providers' APIs, nomenclature and semantics, e.g. interacting with managed PostgreSQL from one provider is a different experience than another, or license changes that render a database unavailable for third-party offering, such as Elasticsearch.

Open-source software also potentially unlocks cost efficiency because, a) it's free, b) it decouples your databases from the infrastructure, enabling you to place them where you want, and c) you have full access to be able to tune and optimize their configuration. You are buying a packaged solution from vendors that is more **open-source adjacent**, it's tied to the infrastructure which they determine, often only available in one environment (typically a handful of clouds), and you aren't given full access to the database because of their SLA requirement.

The above principles support greater sovereignty over your database layer.

But what does their expression look like? The following four markers will help you determine your sovereignty.

The markers of a Sovereign DBaaS

When we talk about the markers of sovereign data, we are referring to these five main concepts:

- ✓ **Control** — You are able to own and assert control over the data pipeline according to your needs through your Sovereign DBaaS implementation — from the underlying infrastructure, databases and their operations, to the location of your workloads.

- ✓ **Access** — You have the level of access your use case requires to your data and the technologies that handle that data. You can access the data plane, the underlying infrastructure, and the data management system. You get root access, allowing you to install, configure, and manage your stack components regardless of the infrastructure.

- ✓ **Portability** — You can take your data from one vendor and port it to another with minimal difficulty and without substantial overhead costs. The third-party approach to database management inevitably leads to organizations becoming wholly dependent on a data service, effectively trapping them in that particular ecosystem.

  Conversely, being data sovereign ensures you don't get locked in with a specific vendor or environment. You can efficiently and cost-effectively move databases from one cloud service to another, or from an on-premise environment to a cloud environment and vice versa.

- ✓ **Licensing stability** — A fundamental principle of sovereign data is the ability to roll your own optimized DBaaS solutions without being subject

to vendors' licensing restrictions. You can include source-available options in your sovereign DBaaS solution like MongoDB and Elasticsearch that third-party service providers are unable to.

- ✓ **Budget control** - Using a provider for your database allows you to get started quickly and easily, with the ability to scale up and down as needed. However, this scaling comes at a cost. If not monitored, and managed, then this can create bills that are an order of magnitude higher than expected. Enforcing limits, both at the billing and deployment level, is crucial to limit these surprises. Within a hybrid/multi-cloud, the complexity increases exponentially.

  With Sovereign DBaaS, you can more easily form a clear understanding of costs because you have greater visibility into and control over inputs, e.g. infrastructure, databases, tools, etc. You can better track and manage them because you can consolidate your database layer into a true single pane of glass.

Now that we understand the principles and markers of a Sovereign DBaaS implementation, it's important to understand its true purpose, i.e. business risk mitigation, which we cover now.

Sovereign DBaaS mitigates top business risks

The pillars of Sovereign DBaaS enable you to mitigate and neutralize many business risks:

Vendor lock-in

Sovereign DBaaS mitigates vendor lock-in due to incompatibilities and prohibitive costs of switching databases or services.

Environment/ecosystem lock-in

Sovereign DBaaS decouples the database from the infrastructure, allowing you to place your databases where and in what configuration you want, e.g. on-prem, cloud, hybrid, and manage them from an environment-spanning single pane of glass.

Key person dependency

Sovereign DBaaS mitigates reliance on one or a few individuals who understand the system and whose absence may cause catastrophic failure. Redundancy, whether via personnel or systems, is built into the database layer so that you can continue to operate as needed.

Managed database license instability

Sovereign DBaaS mitigates issues related to database licenses becoming unavailable or changing terms, forcing a change in your usage. Sovereign DBaaS means you are free from vendor license instability.

Regulatory changes

Sovereign DBaaS mitigates challenges with data privacy regulations by providing the ability to choose where workloads are placed, manage access to them, and create a record of all changes made. This allows organizations to stay compliant with ever-changing regulatory landscapes.

III. How to become data sovereign

In the previous sections of this guide, we introduced the concept of Sovereign DBaaS and its pillars and principles. In this third and final section, we will get into practical applications and elaborate on how you can implement a Sovereign DBaaS of your own.

We'll start with examples of two industries that see substantial benefits from a Sovereign DBaaS implementation and the needs that a sovereign approach solves. Then we'll describe exactly how you too can become data sovereign. Let's dive in!

Part 1: Sovereign DBaaS application examples

The needs solved by a Sovereign DBaaS implementation

Applications that have strict data requirements benefit the most from Sovereign DBaaS. These requirements typically center around regulatory compliance, high performance, and/or high availability.

Regulatory compliance

Organizations that are subject to strict regulatory requirements need to ensure that they are in compliance with data protection laws and regulations. Failing to comply with these regulations can result in hefty fines, loss of reputation, and even legal action. Sovereign DBaaS offers a way to process and store data in environments and locations that align with regulatory requirements and data protection laws.

High performance

Applications with high expectations for speed and reliability suffer if their data services are too slow or unreliable. Sovereign DBaaS offers fast,

reliable, and scalable database services that can fine-tune the performance and availability of customer-facing applications. This, in turn, can help organizations to retain customers and gain a competitive advantage.

High availability

High availability is especially for mission-critical applications such as those used in the finance and healthcare sectors. These applications need to be available 24/7 and any downtime can result in significant financial and reputational damage. Sovereign DBaaS enables organizations to deploy database instances in multiple regions, providing high availability and low latency for customers in different locations. Additionally, automatic failover and load balancing features help to ensure that the application remains available even in the event of a failure.

How Sovereign DBaaS enables regulatory compliance for consumer banks

Consumer banks are responsible for handling Personal Identifiable Information (PII) of customers, sensitive data related to payment processing, accounting ledgers, and customer product portfolios such as loans and insurance. As a result, they are subject to very strict regulatory requirements from government authorities in terms of the ways they handle this sensitive customer data and financial information. Sovereign DBaaS is an ideal solution for consumer banks as it provides a range of benefits that can help them meet their regulatory requirements and protect their sensitive data.

✓ **Key Features of Sovereign DBaaS for consumer banks:**

- **Data Sovereignty**: Allows the bank to process and store data in environments and locations that align with regulatory requirements and data protection laws.
- **Compliance**: Helps the bank to meet regulatory requirements

by providing a way to manage and track access to data, as well as keeping an auditable log of all changes.

- **Security**: Enhances the security of sensitive customer data and financial information by providing encryption and secure access controls.
- **Performance**: Improves the performance and availability of the bank's customer-facing applications by providing a fast, reliable, and scalable DBaaS.
- **Flexibility**: Allows the bank to choose the right type of server and storage, and configure the DBaaS based on its specific needs.
- **Cost-efficiency**: Reduces the cost of maintaining and scaling a DBaaS by providing a self-managed option that can be automated and optimized.
- **Disaster recovery / Cloud Exit Planning**: Gives the bank peace of mind and full control over how they mitigate the impact of unforeseen events.

The first and foremost benefit of Sovereign DBaaS is **data sovereignty**. This allows a consumer bank to process and store data in environments and locations that align with their regulatory requirements and data protection laws. By doing so, the bank can ensure that its data is protected and that it is complying with all relevant regulations.

Another benefit of Sovereign DBaaS is **compliance**. The solution helps the bank to meet regulatory requirements by providing a way to manage and track access to data, as well as keeping an auditable log of all changes. This ensures that the bank can demonstrate compliance with regulations and that it is taking all necessary steps to protect its data.

Security is also a key benefit of Sovereign DBaaS. The solution enhances the security of sensitive customer data and financial information by providing

encryption and secure access controls. This ensures that only authorized personnel can access the data and that it is protected from unauthorized access and cyber threats.

In addition to these benefits, Sovereign DBaaS also offers improved **performance** and availability for the bank's customer-facing applications. The solution provides a fast, reliable, and scalable DBaaS that can help improve the user experience for customers. It also offers flexibility, allowing the bank to choose the right type of server and storage, and configure the DBaaS based on its specific needs. This can help the bank to optimize its operations and reduce costs.

Cost-efficiency is one of the key benefits of Sovereign DBaaS because it allows for self-managed options. By providing the ability to automate and optimize the management of the database, a self-managed option can significantly reduce the cost of maintaining and scaling a DBaaS. This makes it an attractive option for a consumer bank business that is looking to save money while still maintaining a high level of performance and reliability.

Disaster recovery features of a DBaaS are important for consumer banks to safeguard sensitive financial data, ensure regulatory compliance, maintain business continuity, and uphold customer trust. Critically, a Sovereign DBaaS lets the bank retain control and autonomy over their infrastructure. Having complete control of their data enables banks to gain a competitive advantage, improve operational efficiency, and maintain peace of mind; while being able to run Disaster Recovery scenarios without requesting it from a third party. Financial services regulators also provide specific guidelines on cloud exit planning. The European Banking Authority, for example, requires an exit strategy for outsourced critical or important functions in its guidelines on outsourcing arrangements, including the use of the public cloud.

Finally, as with all use cases, Sovereign DBaaS provides monitoring, failover, and load balancing capabilities that can help ensure availability and reliability.

How Sovereign DBaaS enables high availability and high performance for telecoms

Telecommunications companies provide essential connectivity services to their customers, making high availability and high performance crucial for their operations.

✓ **Key features of Sovereign DBaaS for telecoms:**

- **Multi-Region Deployment**: Allows the telecommunications company to deploy database instances in multiple regions, providing high availability and low latency for customers in different locations.
- **Scalability**: Enables the DBaaS to scale vertically and / or horizontally as needed to handle changes in traffic and data volume.
- **Automatic failover**: Allows the DBaaS to automatically route traffic to a healthy instance in the event of a failure, ensuring high availability.
- **Load balancing**: Enables the distribution of traffic across multiple instances of the DBaaS, reducing the risk of a single point of failure and improving performance.
- **Performance optimization**: Provides tools and techniques to optimize the performance of the services managed by the platform, such as caching, memory management, and replication.
- **Monitoring and management**: Provides monitoring and management tools that allow the telecommunications company to track the health and performance of the DBaaS and take action as needed.

- **Security and compliance guarantees**: In addition to high availability and performance, a Sovereign DBaaS enables the telecom to implement granular security policies and ensure compliance with data regulations surrounding user data.

The **multi-region deployment** capability of a Sovereign DBaaS implementation allows telecommunications companies to deploy database instances in each of the regions they operate within, providing high availability and low latency for customers across all their locations. With multiple instances of the database distributed geographically, the risk of a single point of failure is reduced and customers can access their data more quickly and reliably.

Scalability is another important feature of Sovereign DBaaS for telecoms. It enables their DBaaS to scale vertically and/or horizontally as needed to handle changes in traffic and data volume. This means that the telecommunications company can easily adjust the resources allocated to the database to meet the changing needs of their customers.

Automatic failover is a critical feature of Sovereign DBaaS that ensures high availability. In the event of a failure, the DBaaS can automatically route traffic to a healthy instance, minimizing downtime and ensuring that applications can access and act on data without interruption.

Load balancing is another feature that improves performance and reduces the risk of a single point of failure. By distributing traffic across multiple instances of the DBaaS, the load on each instance is reduced, and customers can access their data quickly and reliably.

Sovereign DBaaS also offers **performance optimization** tools and techniques such as caching, memory management, and replication. These

tools help to ensure that the services managed by the platform are running efficiently and effectively.

Monitoring and management tools are also provided by Sovereign DBaaS. These tools allow the telecommunications company to track the health and performance of the DBaaS and take action as needed to ensure that it is running smoothly.

In addition to high availability and performance, Sovereign DBaaS also offers granular **security policies and compliance with data regulations** surrounding user data. This ensures that the telecommunications company can maintain the security and privacy of their customers' data, which is essential in the telecommunications industry.

Overall, a Sovereign DBaaS implementation is an excellent choice for telecommunications companies that need to maintain high availability and performance for their customers while also ensuring the security and privacy of their data. With features such as multi-region deployment, scalability, automatic failover, load balancing, performance optimization, and monitoring and management tools, Sovereign DBaaS is a comprehensive solution that can meet the needs of even the most demanding telecommunications companies.

Part 2: How to become data sovereign

Roll your own DBaaS

The rest of this guide will show you how to create your own custom DBaaS. You will learn how to choose the right database engine, set up a cloud infrastructure, deploy and manage your database instances and monitor their performance. By creating your own custom DBaaS, you will have more control and flexibility over your data and applications.

We'll cover the two most common use cases: one where you're building a DBaaS for your own team or project, and another where you're building a DBaaS for an entire organization.

Use Case: Providing DBaaS for your team or project

Implementing your own DBaaS for projects saves time and money, improves performance and reliability, enhances security, and increases flexibility. At this scope, rolling your own DBaaS is very similar to buying into a traditional DBaaS offering from a vendor.

Building a DBaaS for your team can help them succeed in building applications by providing several benefits:

- ✓ **Enhanced productivity**: A custom DBaaS is designed to meet the specific requirements and workflows of your engineering team, enabling them to work more efficiently and focus on application development rather than database management tasks.

- ✓ **Simplified infrastructure management**: A well-designed DBaaS abstracts the underlying complexities of database infrastructure, allowing your engineering team to focus on building applications without worrying about infrastructure setup, maintenance, and scaling.

- ✓ **Consistency and standardization**: A custom DBaaS enforces consistent database management practices, configurations, and security policies across all applications, reducing the risk of errors and misconfigurations, and ensuring a stable development environment.

- ✓ **Faster development cycles**: A DBaaS provides engineers with easy access to database resources and automates provisioning, scaling, and

backup processes, reducing the time spent on these tasks and allowing for faster development cycles and quicker releases.

- ✓ **Improved collaboration**: A DBaaS can serve as a central platform for your engineering team, facilitating collaboration, knowledge sharing, and streamlined communication, ultimately leading to better applications and faster problem-solving.

- ✓ **Scalability and performance optimization**: A custom DBaaS can be designed to scale efficiently based on the demands of your applications, ensuring optimal performance and resource utilization, and enabling your team to build applications that can grow with your business.

- ✓ **Increased application reliability**: By using a DBaaS with built-in high availability, backup, and disaster recovery features, your engineering team can build applications that are more resilient and less prone to downtime, ensuring a better user experience for your customers.

  Home-built solutions can be developed to provide cloud management functionality using tools and APIs which enable developers to create interfaces that interact with the DBaaS. These solutions can be tailored to the specific needs of the company, allowing for a more customized experience.

Setting up a DBaaS for your team involves several steps, which can be broken down into the following phases:

1. **Planning and design**: Identify your requirements within the larger context of your project or organization, evaluate the DBaaS options available to you, select database technologies that best apply to what you're working on, and define the architecture.

2. **Implementation and deployment**: Set up infrastructure, install database software, configure the system, and integrate with existing tools.

3. **Testing and optimization**: Test the system, optimize performance, and establish monitoring and alerting.
4. **Management and maintenance**: Create documentation, train your team, implement backup and disaster recovery plans, and perform regular maintenance.

By following these steps, you can set up a DBaaS that meets the specific needs of your team, provides a stable and efficient development environment, and enables your engineers to focus on building applications rather than managing database infrastructure. We'll go into more detail about each step of building a DBaaS later in this guide.

Use Case: Providing DBaaS internally across your entire organization

When the scope of the DBaaS extends to an entire organization there are additional components required for an optimal user experience. These components include workflow orchestration tools and cloud management tools.

Workflow orchestration tools are essential if you want to provide internal customers a centralized platform for managing complex workflows and automating tasks. Since a DBaaS involves several components, including compute, storage, and networking, these all need to be coordinated and managed effectively together. Workflow orchestration tools provide a framework for managing these components and streamlining the deployment and management of the DBaaS.

Cloud management tools enable users across the organization to interface with the DBaaS. This is where solutions like ServiceNow or home-built GUI solutions come in. The cloud management module in ServiceNow provides a graphical user interface (GUI) that enables internal customers to interface with the DBaaS. Through this GUI, users can deploy and manage instances, as well as manage access and security. The GUI is intuitive, allowing internal customers to quickly and easily interact with the DBaaS without requiring specialized technical knowledge.

Large organizations are increasingly leveraging **Internal Developer Platforms** (IDPs) to offer DBaaS for internal users, streamlining and automating various processes for both platform teams and application developers. For example, Backstage is a popular open-source platform originally created by Spotify which aims to unify infrastructure tooling, services, and documentation to create a streamlined development environment from end to end.

Platform teams focus on building and maintaining the IDP by incorporating standardization, infrastructure management, service level agreements, and workflow optimization. They configure the IDP to automate repetitive tasks, such as spinning up resources or environments, and establish baseline templates for application configurations and permissions governance.

Application developers gain autonomy in modifying configurations, deploying applications, and managing environments. By integrating with existing workflows, IDPs empower developers to request resources, spin up fully provisioned environments, rollback changes, and deploy applications with ease. This cohesive approach ensures a seamless and efficient DBaaS experience for the entire organization.

Your Use Case

Now that you've got a sense for the scope of Sovereign DBaaS use cases, you're probably considering where yours fits in and how to get started building. The remainder of this guide will step through each part of the process for creating your own DBaaS:

1. **Choose Your Infrastructure**
2. **Select and Configure Your Databases**
3. **Decide Where and How to Deploy**
4. **Set up Security, Access Controls, and Compliance Rules**
5. **Monitor and Manage Performance**
6. **Plan for Disaster and Recovery**

Choose Your Infrastructure

DBaaS infrastructure covers three domains: compute, storage, and networking.

Compute: Choosing the right type of server and instance size

Creating your own custom DBaaS means choosing the right type of server and instance size for your database instances. There are three main options to consider: virtual machines, bare metal servers, and containerized environments.

Virtual machines
Virtual machines (VMs) are software-based emulations of physical servers that run on a hypervisor. VMs offer a high level of abstraction and isolation from the underlying hardware and other VMs. They also allow you to easily

provision, resize and migrate your database instances across different regions and zones. However, VMs also introduce some overhead and performance degradation due to the hypervisor layer.

Bare metal servers

Bare metal servers are physical servers that run without any hypervisor or virtualization layer. They offer a low level of abstraction and isolation from the underlying hardware and other servers. They also provide you with full access and control over your database instances and their resources. However, bare metal servers also require more upfront investment and maintenance costs. They also limit your ability to scale up or down your database instances on demand.

Containerized environments

Containerized environments are software-based packages that bundle together your database instances and their dependencies into isolated units that run on a container engine. Containers offer a medium level of abstraction and isolation from the underlying hardware and other containers. They also enable you to deploy, update and scale your database instances faster and easier than VMs or bare metal servers. However, containers also require some additional tools and skills to manage them effectively.

The best option for your custom DBaaS depends on several criteria, such as:

✓ **Existing architecture** and constraints: Products, tech stacks, and other resources that your organization has already committed to using. For example if your org already uses OpenStack, or has leased data centers, these prior commitments should factor into each subsequent criterion for your DBaaS

✓ **Hardware needs**: The type and amount of CPU cores, memory, disk space, network bandwidth and other resources that your database instances require.

- ✓ **Workloads**: The nature and characteristics of your database operations, such as I/O intensive (e.g., OLTP), RAM intensive (e.g., in-memory databases), GPU intensive (e.g., training models) or mixed workloads.
- ✓ **Scale of traffic**: The volume and variability of requests that your database instances need to handle at peak times or during spikes.
- ✓ **Budget**: The amount of money that you are willing to spend on your compute infrastructure in terms of upfront costs (e.g., capital expenditure) or ongoing costs (e.g., operational expenditure).

Based on these criteria, you can compare the pros and cons of each option and choose the one that best suits your needs.

Storage: Selecting the appropriate storage solution

Selecting the right storage solutions is crucial for any bespoke DBaaS. Which types you'll want to use depends on your use case. When choosing a storage solution for your custom DBaaS, consider the following criteria:

- ✓ **The type of data being stored (structured or unstructured)**
- ✓ **The scale of the data**
- ✓ **Durability and availability requirements**
- ✓ **Access patterns (sequential or random)**
- ✓ **Your budget**

Evaluate these criteria against the available storage solutions below and select the storage solution that best meets your needs and provides the necessary level of performance, scalability, and cost-effectiveness. Storage solutions include block storage, object storage, and file storage.

Block storage

Block storage solutions such as local NVMe disks, SAN (Storage Area Network), and NAS (Network Attached Storage) offer high performance

and low latency. This makes them ideal for use cases that require fast and reliable access to data. They are best suited for storing structured data that is accessed randomly. Block storage is typically more expensive than other storage solutions, but it offers better performance and reliability.

Object storage

Object storage solutions such as Amazon S3, Azure Blob, and Google Cloud Storage are designed to store large amounts of unstructured data. They offer high durability, scalability, and availability at a lower cost compared to block storage. Object storage is best suited for storing large files, backups, and archival data. Access patterns are usually sequential, as object storage is optimized for reading and writing large files.

File storage

File storage solutions such as NFS (Network File System) and SMB (Server Message Block) are commonly used for storing files that need to be shared across multiple systems. They are best suited for storing unstructured data that is accessed randomly. File storage is usually slower than block storage, but it offers better compatibility with various operating systems and applications.

Networking: Setting up network topology and security

In the context of creating a DBaaS, networking infrastructure offers on-demand access to network capabilities and resources. It dictates how services communicate with each other across any combination of public and private clouds. When building your DBaaS networking infrastructure, you'll want to consider network topology, security, load balancing, accessibility, performance, and compliance.

Network topology

The network topology defines the structure of the network and how devices

are connected. There are several options to choose from, including single VPC (Virtual Private Cloud), multiple VPCs, and VPC peering. A single VPC is the simplest option, but it may not be scalable enough to meet the needs of a growing DBaaS. Multiple VPCs and VPC peering allow for more complex and scalable network topologies, but they require more configuration and management.

Network security

Implementing strong security measures is essential to protect your DBaaS from unauthorized access and attacks. Some security measures to consider include firewalls, VPNs (Virtual Private Networks), and security groups. Firewalls can be used to control incoming and outgoing traffic to your DBaaS. VPNs can provide a secure connection between your DBaaS and remote clients. Security groups can be used to define access rules for specific IP addresses and ports.

Load balancing

Setting up load balancers can distribute traffic across multiple instances of your DBaaS, ensuring high availability and performance. Load balancers can be configured to automatically route traffic to the least busy instance or the one that is geographically closest to the client.

DNS

Configuring DNS (Domain Name System) records is essential to ensure that your DBaaS is easily accessible to your clients. DNS allows clients to access your DBaaS using a domain name instead of an IP address. Configuring DNS records can also provide failover and load balancing capabilities.

Performance and compliance

In addition to the key areas discussed above, consider how your networking architecture may affect performance. Network latency and bandwidth will affect the performance of your DBaaS, so it's important to choose a network

solution that provides adequate speed and reliability. Additionally, it's essential to ensure that the chosen network configuration complies with any regulatory requirements for data storage and management.

By evaluating each of the aspects of networking and choosing an appropriate networking solution, you can ensure that your custom DBaaS is secure, highly available, and easily accessible to your clients, while also meeting performance and compliance requirements.

Select and configure your databases

After you've established your DBaaS infrastructure you can move on to the stage of selecting and configuring your choice of databases.

Database Selection

There are myriad database technologies available covering the full spectrum of data types and uses. Consider the following factors to determine which databases may be good choices for your DBaaS.

Data model
Decide whether to use a relational or non-relational data model. Relational databases store data in tables with a predefined schema, while non-relational databases use flexible schemas to store data in various formats. Thus, relational databases are well suited for structured data, while non-relational databases are better for unstructured data.

Query language
Once you have determined the type of database you want to use, you'll need to evaluate the query languages available for the database. SQL is the standard query language used by relational databases, while non-relational databases often use a variety of query languages. It's important to evaluate

the query language options for each database and determine which one will work best for your organization.

Scale

The ability of the database to handle the scale of data and traffic may be an important consideration for your use case. Relational databases are known for their scalability with structured data, and non-relational databases are designed to scale up with large volumes of unstructured data. Nowadays, there are database options in both classes that can handle vertical and horizontal scaling. It's important to determine the expected volume of data and traffic and choose a database that can handle it.

Topology

Topology refers to the structure or layout of a database system, which can significantly impact performance, scalability, and availability. Let's outline some of the more common options below:

- ✓ **Single node**: This is a single instance running your database. This is often used for development environments but (almost) never suitable for production environments.
- ✓ **Active-Passive**: In Active-Passive topologies, there is one primary node for read and write operations, and secondary nodes that replicate data from the primary. Active-Passive provides improved availability and data redundancy, but limited write scalability.
- ✓ **Active-Active**: In Active-Active topologies, multiple nodes can handle read and write operations, improving performance and fault tolerance, but increasing data consistency and management complexity.
- ✓ **Sharding**: Dividing a large database into smaller, more manageable parts called shards, which are distributed across multiple nodes.

The licensing implications of the chosen topology and any required third-party solutions or libraries may also impact other choices for the

DBaaS. For example, some vendors may require specific licensing for clustered or HA database configurations, and some database solutions may require third-party libraries or solutions to implement specific topologies.

Concurrency

Concurrency refers to the ability of the database to handle multiple concurrent requests. Relational databases typically handle concurrency well, while non-relational databases may struggle with it. It's important to evaluate the concurrency requirements of the business and choose a database that can handle the expected load.

Durability

Durability refers to the ability of the database to recover data in the event of a failure or outage. Relational databases typically have strong durability guarantees and are designed to handle failures, while non-relational databases may have weaker durability guarantees.

Cost

Cost is always a factor to consider. Some databases may be expensive to license, while others may require significant maintenance resources. It's important to evaluate the total cost of ownership for each database and determine which one provides the best value for the business. Evaluate the cost of using and maintaining the database, including licensing fees, hosting costs, and ongoing maintenance costs.

Your existing stack

Consider whether you have existing tools and applications that use specific technologies, such as web applications that use MySQL while your mobile apps use CouchDB. Will these be replaced or are you looking for a more reliable way to run the same stack? If you have existing applications that rely on a particular database technology, it may be necessary to choose a database that can integrate with these tools.

Database configuration

Once you have selected the appropriate database for your custom DBaaS, the next step is to configure it to meet your needs. This process involves several key areas that are essential to ensuring the database is secure, performs well, and can recover from any potential failures.

Instance configuration

Instance config involves configuring the instance size and type, storage, and security settings. It is essential to properly size the instance to ensure it can handle the expected workload while staying within budget. The storage and security settings should also be carefully configured to protect the data and ensure it is only accessible to authorized users.

Database initialization

The next area to consider is database initialization. This involves importing data, creating tables, and defining indexes. Importing data can be a complex process, so it is important to plan and test this carefully to avoid any potential data loss or corruption. Creating tables and defining indexes should be done with care to ensure optimal performance.

Performance tuning

Configure settings such as caching, memory management, and replication, so you can optimize performance and ensure that the database can handle the expected workload. This process can be time-consuming, but it is essential to ensure the database performs well and meets your needs.

Monitoring and management

Monitoring and management are critical to maintaining the health and performance of the database. Set up monitoring and management tools to track performance metrics, alert you to potential issues, and provide visibility

into the database's health. This allows you to proactively address any potential issues before they impact users.

Backup and recovery

Backup and recovery procedures ensure that data is protected and can be easily restored in the event of a failure. This can be a complex process, and goes beyond a backup script scheduled with cron. A backup and recovery solution would need to be able to do point in time restore, automatic verification of the backup files, retention, compression, encryption and upload to external storage. So it is important to plan and test this carefully to avoid any potential data loss or corruption.

Additional considerations

- ✓ High availability is important for ensuring the database is always available. This can be achieved by configuring the database for high availability, such as using replication or clustering.
- ✓ Security is essential, so it is important to ensure the database is properly configured for security, such as encrypting data at rest and in transit.
- ✓ Compliance should be considered to ensure that the chosen configuration complies with any regulatory requirements for data storage and management.

Configuring a database for a custom DBaaS involves several critical areas that must be carefully planned and executed to ensure optimal performance, security, and availability. By carefully considering each of these areas, you can create a database that meets your specific needs and provides reliable, high-performance data storage and management.

Decide where and how to deploy

Now that you've established your infrastructure and selected and configured your databases, you can decide where and how to deploy.

Availability zones

Availability zones are discrete locations within a cloud region that are designed to be isolated from failures in other availability zones. Using availability zones can provide increased availability and redundancy for your database.

Geographic diversity

It is essential to ensure that the availability zones are located in different geographic locations to reduce the risk of a single event affecting all zones. Further, the geographic diversity of your availability zones should correspond to where your users are located.

Redundancy

Implementing redundancy within the availability zones is important to ensure that there are multiple copies of data and resources available. This redundancy ensures that in case of any failures, the system continues to run without interruption, and the data remains intact.

Automatic failover

Automatic failover is another essential factor in availability zone configuration. Configuring automatic failover ensures that traffic is automatically routed to a healthy availability zone in case of a failure. This setup ensures that your database is available and accessible to your users without any disruption.

Testing

Test your availability zones regularly to ensure that they are properly configured and can handle failover scenarios. Regular testing helps identify and fix any issues before they cause downtime or affect user experience.

Cost

The cost of setting up and maintaining availability zones can be high. Weigh the benefits of using more availability zones against the added cost.

Compliance

Compliance is another important factor to consider when configuring availability zones. Ensure that the chosen availability zone configuration complies with the local regulatory requirements for data storage and management.

In summary, deploying your database in multiple availability zones provides increased availability, redundancy, and automatic failover, ensuring that your database is always available and accessible to your users. Be sure to consider cost, compliance, and how you'll conduct regular testing.

Deployment models

Choosing the appropriate deployment model is crucial when building your own DBaaS. There are two primary deployment models: single-region and multi-region.

Single-region deployment model

In a single-region deployment model, a single instance of the DBaaS is deployed in one region. This model is best suited for small or medium-sized applications with low traffic. It is relatively simple to deploy and manage, and the cost is lower. However, this model offers less availability and disaster recovery options compared to a multi-region model. A single region may

have more than one availability zone, which makes it more robust. If the application is small and has low traffic, a single-region deployment model is sufficient.

Multi-region deployment model

In a multi-region deployment model, a DBaaS instance is deployed in multiple regions, providing higher availability and better performance for users in different locations. This model is also used for compliance reasons when data must be stored in a particular region. Multi-region deployment is complex to deploy and manage, and the cost is higher compared to a single-region model. However, it offers high availability and better performance for users in different locations.

Additional considerations

- ✓ Latency can impact application performance, so deploying multiple availability zones in each region is recommended. If latency is a priority, it is essential to incorporate it into cost calculations and thoroughly map the network architecture. By examining overlapping networks and potential cross-network interactions, you can optimize the architecture to minimize latency, either by adding caching mechanisms or strategically colocating infrastructure to reduce secondary costs.

- ✓ Ensuring data consistency between instances in different regions can be a challenge and needs to be considered in a multi-region deployment model.

- ✓ The cost of deploying and maintaining instances in multiple regions can be high. Ingress and egress costs should be considered in public cloud and leased / colocation environments, and the networking setup can greatly impact the monthly cost.

- ✓ Regulatory requirements for data storage and management must be complied with for each region in which you operate.

Selecting the appropriate deployment model for your DBaaS makes a big difference in how your DBaaS meets various requirements. A single-region deployment model is suitable for small or medium-sized applications with low traffic, while a multi-region deployment model is better for applications that require higher availability and better performance for users in different locations. There are pros and cons to each model, and additional considerations such as latency, data consistency, cost, and compliance must be taken into account when making a decision.

Set up Security, Access Controls, and Compliance Rules

Security is a top priority when setting up a DBaaS, and it is essential to ensure that it is secure and compliant with regulations. This section will discuss the various security measures that should be taken to protect the DBaaS, including authentication and authorization, encryption, network security, vulnerability management, incident response, compliance, third-party access, auditing, and risk assessment.

Authentication and authorization

Authentication and Authorization are the primary security measures used to ensure that only authorized users can access the DBaaS. Authentication is the process of verifying the identity of a user or system, while Authorization is the process of granting access to a user or system based on their identity. Tools like LDAP, Active Directory, and OAuth are used for this purpose. These tools provide a centralized database of user credentials that can be used to authenticate users and determine their level of access to the DBaaS.

Encryption

Encryption should be used to protect data in transit and at rest. LUKS, dm-crypt, and BitLocker are some of the encryption tools that can be used for this purpose. Encryption protects against data breaches by ensuring that data cannot be accessed by unauthorized users even if it is intercepted.

Network security

Network security measures such as firewalls, VPNs, and security groups should be implemented to secure any DBaaS. Firewalls like iptables and UFW can be used to block unauthorized access to the DBaaS, while VPNs like OpenVPN and IPSec can be used to encrypt and secure network traffic. Security groups can be used to control inbound and outbound traffic to and from the DBaaS.

Vulnerability management

Vulnerability management is essential to ensure that the DBaaS remains secure over time. Regular monitoring and patching of vulnerabilities in the DBaaS should be done using tools such as Nessus, OpenVAS, and Nessus Agent.

Incident response

Incident response plans should be put in place to respond quickly and effectively to any security incidents that may occur. Tools such as SOC Prime, ThreatConnect, and IBM Resilient can be used for this purpose.

Compliance

As with every part of a DBaaS, compliance is a critical aspect of DBaaS security. It is essential to ensure that the DBaaS complies with relevant regulations such as HIPAA, PCI-DSS, GDPR, and any other applicable regulatory schemes in the areas you operate. Compliance can be achieved through the use of various security measures such as encryption, access controls, and auditing.

Additional considerations

- ✓ Third-party access should be carefully managed and monitored to ensure that only authorized third parties can access the DBaaS.
- ✓ Auditing should be used to keep track of all access and changes to the DBaaS. This can help identify any security incidents that may occur and provide a record of the events that led up to the incident.
- ✓ Regular risk assessments are helpful to identify and mitigate any security risks to the DBaaS. This can involve identifying vulnerabilities in the system, assessing the potential impact of a security incident, and taking steps to prevent the incident from occurring. Regular risk assessments can help ensure that the DBaaS remains secure and compliant with regulations over time.

Setting up a secure and compliant DBaaS requires implementing various security measures such as authentication and authorization, encryption, network security, vulnerability management, incident response, compliance, third-party access management, auditing, and risk assessment. By taking these measures, you can ensure that your DBaaS remains secure and compliant with regulations and continues to be secure and compliant over time.

Monitor and manage performance

Once your DBaaS is up and running you'll want to ensure that it's doing its job and doing it well. These are the kinds of tools you'll want to implement for monitoring and managing performance.

Monitoring and logging

Establishing robust logging and monitoring practices ensure the DBaaS is running optimally and will help to identify and troubleshoot issues. Effective logging and monitoring also play a vital role in meeting regulatory

requirements, such as HIPAA, PCI-DSS, and GDPR, which mandate that all access to sensitive data must be tracked and auditable. Here are some considerations to keep in mind when setting up logging and monitoring for your DBaaS:

Data collection

Collecting log data from the DBaaS is the first step in establishing a comprehensive logging and monitoring strategy. There are different types of logs that can be collected, such as query logs, error logs, and transaction logs. These logs provide valuable information about the system's performance and any errors or issues that may be affecting it. There are several tools available to collect log data from a DBaaS, such as Syslog, Rsyslog, and Fluentd. These tools allow administrators to collect log data from different sources and transmit it to a centralized location for analysis.

Storage

Once log data is collected, it must be stored in a centralized location for easy access and analysis. Elasticsearch, Kibana, and Logstash are some of the popular tools used for storing log data. These tools provide a powerful search engine that enables administrators to quickly search through large volumes of log data.

Analysis

Analyzing log data is critical in identifying patterns, troubleshooting issues, and monitoring for security threats. Tools such as Splunk, Loggly, and Sumo Logic provide advanced analytics and visualization features that allow administrators to gain insights from log data.

Alerting

Setting up alerts to notify administrators of issues or suspicious activity is essential in keeping the DBaaS running smoothly. Tools such as PagerDuty,

VictorOps, and Nagios enable administrators to set up alerts based on specific criteria, such as CPU usage, memory utilization, and disk space.

Compliance

Ensuring that the logging and monitoring configuration complies with relevant regulations such as HIPAA, PCI-DSS, and GDPR is essential. Audit logging is a must-have feature for compliance, as it allows administrators to track all access and changes to the DBaaS.

Additional considerations

- ✓ Retention policies should be set to ensure that log data is kept for the appropriate amount of time.
- ✓ Searchability of log data is critical in ensuring easy access and analysis.
- ✓ Third-party access to log data should be monitored and managed to maintain system security and compliance.

Setting up logging and monitoring is a critical component of building a DBaaS infrastructure. Collecting and storing log data in a centralized location, analyzing it, and setting up alerts can help administrators quickly identify and troubleshoot issues, monitor for security threats, and ensure compliance with regulatory requirements.

Managing performance

Managing performance is one of the most rewarding parts of running a successful DBaaS. Here are some tips to help you ensure optimal performance for your DBaaS:

Load testing

Regularly perform load testing to determine the capacity of your DBaaS and identify any scalability issues that may need to be addressed. Load testing

simulates user traffic and helps you identify potential bottlenecks and areas for improvement. This ensures that your DBaaS can handle the expected workload and scale when necessary.

Capacity planning
Analyze performance metrics to understand resource utilization trends and proactively plan for future resource needs. Capacity planning helps you ensure that your DBaaS has enough resources to handle current and future workloads. It involves monitoring resource utilization and identifying any trends that suggest you may need to scale up or down.

Optimizing configurations
Regularly optimize database configurations to ensure optimal performance, such as fine-tuning parameters like query performance, indexing strategies, and memory management. Tuning the database configurations helps you optimize the use of resources, reduce overhead, and improve overall performance.

Performance tuning
Regularly perform performance tuning to identify areas for improvement and optimize database performance, such as optimizing queries, indexes, and caching strategies. Performance tuning helps you identify bottlenecks and other issues that impact performance and allows you to take corrective actions to improve database performance.

Performance optimization tools
Utilize performance optimization tools to automate performance tuning and help identify performance issues, such as database profiling and performance analysis tools. These tools provide valuable insights into the performance of your DBaaS and help you optimize database performance more efficiently.

Regularly performing load testing, capacity planning, optimizing configurations, performance tuning, and utilizing performance optimization tools can help you ensure optimal performance, scalability, and efficiency for your DBaaS.

Plan for disaster and recovery

When things go wrong, it pays to have a plan. Here are some points to consider when creating a disaster and recovery plan for your DBaaS:

Disaster Recovery Planning
Developing a comprehensive disaster recovery plan is critical for maintaining business continuity in the event of a disaster. A disaster recovery plan should include a clear and documented process for recovering from a disaster, including the steps required to restore backups, recover data, and resume operations. Regular testing of your disaster recovery plan is also essential to ensure that it is effective and up-to-date.

Offsite Backup Storage
Consider storing your backups offsite or in a remote location to ensure that they are protected from local disasters. This could include backing up your data to a cloud-based storage service or physically transporting backup media to a secure offsite location. It is important to regularly test the process of restoring backups from offsite storage to ensure that backups are accessible and usable in the event of a disaster.

Automated Backup Management
Implementing automated backup management tools can simplify the backup process and ensure that backups are completed regularly and efficiently. These tools can also provide additional features, such as compression, encryption, and incremental backups, that can help optimize storage and minimize backup time. It is important to regularly test the process of restoring backups created by automated backup management tools to ensure that backups are complete and accurate.

Creating a comprehensive disaster recovery plan, storing backups offsite, and implementing automated backup management tools are all essential steps to ensure that your DBaaS is resilient and can recover quickly from a disaster. Regularly testing these processes and making updates as needed is also critical to ensure that your disaster and recovery plan is effective and up-to-date.

Conclusion: A DBaaS of your own

Building a Sovereign DBaaS can be an intimidating task. However, by considering the factors mentioned in each section of this article, it is possible to roll your own DBaaS that is tailored to meet the specific needs of your organization. From choosing the right database technology and designing a scalable architecture to implementing robust logging and monitoring and creating a comprehensive disaster recovery plan, every aspect of building a successful DBaaS has been covered here.

By following these guidelines and best practices, you can ensure that your DBaaS is resilient, secure, performant, and compliant with regulatory requirements. With careful planning and execution, you can build a DBaaS infrastructure that meets the unique needs of your organization while providing flexibility, scalability, reliability, and cost-effectiveness.

www.severalnines.com